SIEMENS

SIMATIC

STEP 7 Professional V13.0

System Manual

System overview of STEP 7 and WinCC 1

Readme 2

Installation 3

Migrating projects and programs 4

First steps 5

Introduction to the TIA Portal 6

Editing projects 7

Editing devices and networks 8

Programming the PLC 9

Visualize processes 10

Using technology functions 11

Using Team Engineering 12

Using online and diagnostics functions 13

Hardware documentation 14

Printout of the online help

02/2014

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠ DANGER
indicates that death or severe personal injury **will** result if proper precautions are not taken.

⚠ WARNING
indicates that death or severe personal injury **may** result if proper precautions are not taken.

⚠ CAUTION
indicates that minor personal injury can result if proper precautions are not taken.

NOTICE
indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

⚠ WARNING
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Siemens AG
Industry Sector
Postfach 48 48
90026 NÜRNBERG
GERMANY

℗ 02/2014 Subject to change

Copyright © Siemens AG 2014.
All rights reserved

Table of contents

1 System overview of STEP 7 and WinCC....27

1.1 Scaling of STEP 7 and WinCC in the TIA Portal....27

1.2 Options for STEP 7 Engineering System....28

1.3 Options for WinCC Engineering and Runtime systems....28

2 Readme....31

2.1 General notes....31

2.2 STEP 7 Professional....35
2.2.1 Security information....35
2.2.2 News....37
2.2.3 Notes on use....37
2.2.4 Editing devices and networks....39
2.2.4.1 General information on devices and networks....39
2.2.4.2 Use of modules on the S7-1200....40
2.2.4.3 Replacing ET 200S positioning modules....41
2.2.4.4 CP 343-2 on SIMATIC S7 Embedded Controller EC31-RTX....41
2.2.4.5 Notes on online and diagnostics....42
2.2.4.6 Network components....42
2.2.5 Programming a PLC....45
2.2.5.1 General notes on PLC programming....45
2.2.5.2 Instructions....49
2.2.5.3 Testing the user program....51
2.2.6 Inter Project Engineering (IPE)....52
2.2.6.1 Notes on IPE....52
2.2.7 Technological functions....53
2.2.7.1 Notes on technological functions....53
2.2.7.2 Notes on technological functions....54
2.2.7.3 Notes on technological functions....54
2.2.7.4 SIMATIC STEP 7 Easy Motion Control....54
2.2.7.5 SIMATIC STEP 7 PID Professional....54

2.3 WinCC Basic....55
2.3.1 Security information....55
2.3.2 News....56
2.3.3 Notes on use....57
2.3.4 Migration....59
2.3.5 Engineering System....62
2.3.5.1 Screens and Screen Objects....62
2.3.5.2 Tags and connections....66
2.3.5.3 Alarm system and alarm displays....68
2.3.5.4 System functions....69
2.3.5.5 Recipes....69
2.3.5.6 User administration....69
2.3.5.7 Communication....70
2.3.6 System-wide functions....72

2.3.7 Compiling and loading....72
2.3.8 Runtime....75
2.3.8.1 Notes on operation in Runtime....75
2.3.8.2 Notes on operation of panels in Runtime....77
2.3.9 HMI devices....78
2.3.9.1 Notes on HMI devices....78

3 Installation....83

3.1 Notes on the installation....83

3.2 System requirements for installation....84
3.2.1 Notes on licenses....84
3.2.2 Notes on the system requirements....85
3.2.3 System requirements STEP 7 Professional....86
3.2.3.1 Licensing of STEP 7....86
3.2.3.2 Handling licenses and license keys....88
3.2.3.3 STEP 7 Professional for PC-CP....90
3.2.3.4 Software and hardware requirements STEP 7....91
3.2.4 System requirement for WinCC Basic....94
3.2.4.1 Software and hardware requirements....94
3.2.4.2 Parallel installation....98
3.2.4.3 Licenses and Powerpacks....99

3.3 Installation log....103

3.4 Starting installation....104

3.5 Checking availability of updates and support packages and installing them....106

3.6 Displaying the installed software....108

3.7 Modifying or updating installed products....109

3.8 Repairing installed products....110

3.9 Starting to uninstall....112

3.10 Installing and uninstalling the migration tool....114
3.10.1 System requirements....114
3.10.2 Installing the migration tool....114
3.10.3 Uninstalling the migration tool....115

4 Migrating projects and programs....117

4.1 Migrating projects in a TIA portal project....117
4.1.1 Migration of projects with the TIA Portal....117
4.1.2 Preparing projects with the migration tool....118
4.1.2.1 Migrating projects with the migration tool....118
4.1.2.2 Calling the migration tool....120
4.1.2.3 Creating a migration file....120
4.1.3 Migrating projects....121
4.1.4 Displaying the history of the migration....123
4.1.5 Display migration log....123
4.1.6 Migrating STEP 7 projects....124
4.1.6.1 Migration of STEP 7 projects....124
4.1.6.2 Requirements for the migration of STEP 7 projects....125
4.1.6.3 Checking whether STEP 7 projects can be migrated....126
4.1.6.4 Consistency check in initial project prior to the migration....127

4.1.6.5 Removing unsupported hardware components....127
4.1.6.6 Special points to observe during migration....128
4.1.7 Migrating WinCC flexible projects (Basic)....142
4.1.7.1 Basics (WinCC flexible)....142
4.1.7.2 Migrating engineering data (WinCC flexible)....147
4.1.7.3 Migrating runtime data (WinCC flexible)....158
4.1.7.4 Migrating integrated projects (WinCC flexible)....160
4.1.7.5 Reference (WinCC flexible)....164
4.1.8 Migrating integrated projects....172
4.1.8.1 Migrating an integrated project....172
4.1.8.2 Post-editing integrated projects....174
4.1.8.3 Converting unspecified CPUs into specified CPUs....175
4.1.8.4 Creating an integrated HMI connection....177
4.1.8.5 Re-linking HMI tags....179
4.1.8.6 Deleting an unspecified connection....180

4.2 Migrating PLC programs to a S7-1500 CPU....180
4.2.1 Basic information on migrating PLC programs....180
4.2.2 Performing migration....182
4.2.3 Displaying a log file....183
4.2.4 Special considerations for migrating PLC programs....184
4.2.4.1 Information on migrating PLC programs....184
4.2.4.2 Migration of instructions to S7-1500....190
4.2.4.3 Migration of LAD/FBD programs to S7-1500....213
4.2.4.4 Migration of STL programs to S7-1500....216
4.2.4.5 Migration of SCL programs to S7-1500....228
4.2.4.6 Migration of GRAPH programs to S7-1500....233

4.3 Migrating S7-1200 to firmware as of V4....238
4.3.1 Basic information on upgrading to V4....238
4.3.2 Migrating to V4....240
4.3.3 Special considerations after migrating to V4....241

4.4 Programming recommendations....243
4.4.1 The new S7-1500 CPU functions at a glance....243
4.4.2 Symbolic addressing....247
4.4.3 Using IEC timers and counters....249
4.4.4 Flexibly using enable output ENO....250
4.4.5 Using MOVE instructions in STL....252
4.4.6 Implementing array access with a variable index....254
4.4.7 Using PLC data types (UDT)....256
4.4.8 Querying and fixing errors in the program code....258
4.4.9 Using VARIANT instructions....263

5 First steps....271

5.1 Getting Started Documentation....271

6 Introduction to the TIA Portal....273

6.1 User interface and operation....273
6.1.1 Starting, setting and exiting the TIA Portal....273
6.1.1.1 Starting and exiting the TIA Portal....273
6.1.1.2 Overview of the program settings....273
6.1.1.3 Overview of the script and text editor settings....275
6.1.1.4 Overview of the print settings....276

6.1.1.5 Changing the settings....276
6.1.2 Layout of the user interface....277
6.1.2.1 Views....277
6.1.2.2 Portal view....277
6.1.2.3 Project view....279
6.1.2.4 Library view....281
6.1.2.5 Project tree....282
6.1.2.6 Work area....285
6.1.2.7 Inspector window....293
6.1.2.8 Task cards....295
6.1.2.9 Reference projects....297
6.1.2.10 Details view....299
6.1.2.11 Overview window....300
6.1.2.12 User interface layout....304
6.1.3 Keyboard operation in the TIA Portal....308
6.1.3.1 Operating the TIA Portal with the keyboard....308
6.1.3.2 Displaying an overview of all keyboard shortcuts....308
6.1.3.3 Basic functions of the TIA Portal....308
6.1.3.4 Using project-related functions....310
6.1.3.5 Arranging windows....310
6.1.3.6 Navigating through the program interface....311
6.1.3.7 Customizing editors....312
6.1.3.8 Editing objects....314
6.1.3.9 Text editing....315
6.1.3.10 Editing tables....316
6.1.3.11 Using online functions....317
6.1.3.12 Using the on-screen keyboard....318
6.1.4 Special features specific to the operating system....318
6.1.4.1 Influence of user rights....318
6.1.4.2 Expanding user rights....319

6.2 Help on the information system....320
6.2.1 General remarks on the information system....320
6.2.2 Opening the supplied Help system....323
6.2.3 Searching the Help system for keywords....324
6.2.4 Full-text searches....324
6.2.5 Using favorites....325
6.2.6 Printing help topics....326
6.2.7 Configuring the display of tooltips and tooltip cascades....327
6.2.8 Safety Guidelines....327
6.2.9 Assembling customized documentation....329

7 Editing projects....331

7.1 The basics of projects....331

7.2 Using logs....331

7.3 Creating and managing projects....332
7.3.1 Creating a new project....332
7.3.2 Compatibility of projects....333
7.3.3 Opening projects....334
7.3.4 Upgrading projects....335
7.3.5 Displaying properties of the project....337
7.3.6 Saving projects....338

7.3.7 Closing projects....339
7.3.8 Deleting projects....339
7.3.9 Working with multi-language projects....340
7.3.9.1 Project text basics....340
7.3.9.2 Select project languages....342
7.3.9.3 Setting the editing language....342
7.3.9.4 Translating all project texts in tabular form....343
7.3.9.5 Translating text associated with individual objects....344
7.3.9.6 Translating texts using reference texts....344
7.3.9.7 Exporting and importing project texts....345
7.3.9.8 Application examples for multilanguage projects....347
7.3.10 Archiving and retrieving projects....348
7.3.10.1 Working with project archives....348
7.3.10.2 Archiving projects....349
7.3.10.3 Retrieving projects....350

7.4 Using reference projects....351
7.4.1 Basics of reference projects....351
7.4.2 Opening and closing a reference project....351
7.4.3 Comparing reference projects....352

7.5 Editing project data....353
7.5.1 Compiling and loading project data....353
7.5.1.1 Compiling project data....353
7.5.1.2 Loading project data....355
7.5.2 Comparing project data....363
7.5.2.1 Basics of project data comparison....363
7.5.2.2 Carrying out an online/offline comparison....364
7.5.2.3 Carrying out offline/offline comparisons....364
7.5.2.4 Using the compare editor 365
7.5.3 Protecting project data....378
7.5.3.1 Protection concept for project data....378
7.5.3.2 Revoking access rights for devices....379
7.5.4 Printing project contents....380
7.5.4.1 Printing project documentation....380
7.5.4.2 Printing module labels....399

7.6 Undoing and redoing actions....402
7.6.1 Basics of undoing and redoing actions....402
7.6.2 Undoing an action....404
7.6.3 Redoing an action....405

7.7 Finding and replacing in projects....406
7.7.1 Information on the search function....406
7.7.2 Search and replace....407

7.8 Working with text lists....409
7.8.1 Text lists....409
7.8.2 Creating user-defined text lists....410
7.8.3 Editing user-defined text lists....411
7.8.4 Editing system-defined text lists....411

7.9 Using memory cards....412
7.9.1 Basics about memory cards....412
7.9.2 Adding a user-defined card reader....413

7.9.3 Accessing memory cards....414
7.9.4 Displaying properties of memory cards....414

7.10 Using libraries....415
7.10.1 Library basics....415
7.10.2 Using the "Libraries" task card....417
7.10.2.1 Overview of the "Libraries" task card....417
7.10.2.2 Using the element view....419
7.10.3 Using the library view....420
7.10.3.1 Overview of the library view....420
7.10.3.2 Opening and closing the library view....422
7.10.4 Using library management....423
7.10.4.1 Overview of the library management....423
7.10.4.2 Opening library management....425
7.10.4.3 Filtering the display of types....426
7.10.4.4 Displaying instances in the project....426
7.10.4.5 Displaying cross references of an instance....427
7.10.4.6 Displaying relations to other library objects....427
7.10.5 Using global libraries....428
7.10.5.1 Creating a global library....428
7.10.5.2 Compatibility of global libraries....429
7.10.5.3 Opening a global library....430
7.10.5.4 Upgrading global libraries....431
7.10.5.5 Displaying properties of global libraries....433
7.10.5.6 Displaying logs of global libraries....433
7.10.5.7 Saving a global library....434
7.10.5.8 Closing a global library....435
7.10.5.9 Deleting a global library....436
7.10.5.10 Archiving and disabling global libraries....436
7.10.5.11 Using global corporate libraries....439
7.10.6 Creating folders in a library....441
7.10.7 Using master copies....441
7.10.7.1 Basics on master copies....441
7.10.7.2 Adding master copies....442
7.10.7.3 Filtering master copies....443
7.10.7.4 Using master copies....444
7.10.8 Using types and their versions....444
7.10.8.1 Basics on types....444
7.10.8.2 State of type versions....446
7.10.8.3 Displaying a released type version....447
7.10.8.4 Displaying properties of a type or version....448
7.10.8.5 Working with types in the project library....449
7.10.8.6 Working with types in global libraries....460
7.10.8.7 Assigning a version....463
7.10.9 Editing library elements....465
7.10.10 Updating a library with the contents of another library....467
7.10.11 Harmonizing names and path structure....469
7.10.12 Clean up library....470
7.10.13 Comparing library elements....471

7.11 Using cross-references....472
7.11.1 Using cross-references....472

7.12 Simulating devices....472

7.12.1 Simulation of devices....472
7.12.2 Starting the simulation....473

8 Editing devices and networks....475

8.1 Configuring devices and networks....475
8.1.1 Hardware and network editor....475
8.1.1.1 Overview of hardware and network editor....475
8.1.1.2 Network view....477
8.1.1.3 Device view....479
8.1.1.4 Topology view....482
8.1.1.5 Overview of settings for hardware configuration....483
8.1.1.6 Printing hardware and network configurations....484
8.1.1.7 Activating the page break preview for printout....486
8.1.1.8 Changing the print options....486
8.1.1.9 Inspector window....487
8.1.1.10 Hardware catalog....488
8.1.1.11 Enabling product support....490
8.1.1.12 Displaying product support for hardware components....491
8.1.1.13 Keyboard operation: Navigation in the editor....492
8.1.1.14 Keyboard operation: Editing objects....493
8.1.2 Configuring devices....494
8.1.2.1 Basics....494
8.1.2.2 Configuring individual devices....503
8.1.2.3 Comparing devices....517
8.1.3 Configure networks....519
8.1.3.1 Networking devices....519
8.1.3.2 Communication via connections....541
8.1.3.3 Displaying and configuring topology....709
8.1.3.4 Industrial Ethernet Security....725
8.1.4 Creating configurations....862
8.1.4.1 Information about the web server....862
8.1.4.2 Things you should know about PROFIBUS DP operating modes....863
8.1.4.3 Configuring automation systems (S7-300, S7-400, S7-1500)....864
8.1.4.4 Configuring automation systems (S7-1200)....937
8.1.4.5 Configurations for point-to-point connections (S7-1500)....973
8.1.4.6 Configurations for point-to-point connections (S7-300, S7-400)....1027
8.1.4.7 Configuring PC stations....1154
8.1.4.8 S7-1200 CM/CP....1164
8.1.4.9 SCALANCE X, W and M....1179
8.1.4.10 Configuring PROFIBUS DP....1339
8.1.4.11 Configuring PROFIBUS DP (S7-1200)....1413
8.1.4.12 Configurations for PROFINET IO....1436
8.1.4.13 Bus coupling with PN/PN coupler....1550
8.1.4.14 Integrating external tools....1551
8.1.4.15 Loading a configuration....1553
8.1.5 Additional information on configurations....1563
8.1.5.1 Functional description of S7-1200 CPUs....1563
8.1.5.2 Functional description of S7-300/400 CPUs....1639
8.1.5.3 Functional description of S7-1500 CPUs....1831
8.1.5.4 CPs, PC stations and network components....1922
8.1.5.5 Identification systems....2012
8.1.5.6 Distributed I/O....2019

8.1.5.7 IPv6 configuration....2072
8.1.6 Displaying alarms....2074
8.1.6.1 Overview of the alarm display....2074
8.1.6.2 Archive view....2074
8.1.6.3 Layout of the alarms in the archive view....2075
8.1.6.4 Receiving alarms....2075
8.1.6.5 Export archive....2076
8.1.6.6 Clear archive....2076
8.1.6.7 "Active alarms" view....2076
8.1.6.8 Layout of the alarms in the "Active alarms" view....2076
8.1.6.9 Status of the alarms....2077
8.1.6.10 Acknowledging alarms....2077
8.1.6.11 Ignoring alarms....2078
8.1.6.12 Keyboard commands in the alarm display....2078

8.2 Device and network diagnostics....2079
8.2.1 Hardware diagnostics....2079
8.2.1.1 Overview of hardware diagnostics....2079
8.2.1.2 Showing non-editable and current values of configurable module properties....2089
8.2.1.3 Showing the current values of dynamic modules properties....2096
8.2.1.4 Checking a module for defects....2100
8.2.1.5 Changing the properties of a module or the programming device/PC....2107
8.2.1.6 Diagnostics in STOP mode....2123
8.2.1.7 Online accesses in the Online and Diagnostics view....2125
8.2.1.8 Checking PROFIBUS DP subnets for faults....2128
8.2.2 Connection diagnostics....2131
8.2.2.1 Overview of connection diagnostics....2131
8.2.2.2 Displaying the connection status using icons....2132
8.2.2.3 Detailed connection diagnostics....2133
8.2.3 System diagnostics with 'Report System Errors'....2136
8.2.3.1 Introduction to system diagnostics with 'Report System Errors'....2136
8.2.3.2 Basics of system diagnostics....2137
8.2.3.3 Modules with limited diagnostic possibilities....2137
8.2.3.4 Diagnostics blocks for reporting system errors....2139
8.2.3.5 Properties of the blocks....2140
8.2.3.6 Supported error OBs....2140
8.2.3.7 Diagnostics status DB....2142
8.2.3.8 Displaying settings of system diagnostics....2145
8.2.3.9 Settings....2145
8.2.4 System diagnostics for S7-1500 PLCs....2147
8.2.4.1 Introduction....2147
8.2.4.2 Basics of system diagnostics....2147
8.2.4.3 Displaying settings of system diagnostics....2148
8.2.4.4 Settings....2148
8.2.5 Using additional diagnostics options....2149
8.2.5.1 Hardware diagnostics....2149
8.2.5.2 Connection diagnostics....2165

9 Programming the PLC....2167

9.1 Creating a user program....2167
9.1.1 Programming basics....2167
9.1.1.1 Operating system and user program....2167
9.1.1.2 Blocks in the user program....2168

9.1.1.3 Block calls....2180
9.1.1.4 Using and addressing operands....2197
9.1.1.5 Data types....2236
9.1.1.6 Program flow control....2447
9.1.2 Declaring PLC tags....2473
9.1.2.1 Overview of PLC tag tables....2473
9.1.2.2 Structure of the PLC tag tables....2474
9.1.2.3 Rules for PLC tags....2475
9.1.2.4 Creating and managing PLC tag tables....2479
9.1.2.5 Declaring PLC tags....2481
9.1.2.6 Grouping PLC tags for inputs and outputs in structures....2485
9.1.2.7 Declaring global constants....2488
9.1.2.8 Editing properties....2490
9.1.2.9 Monitoring of PLC tags....2493
9.1.2.10 Editing PLC tag tables....2493
9.1.3 Creating and managing blocks....2500
9.1.3.1 Creating blocks....2500
9.1.3.2 Specifying block properties....2512
9.1.3.3 Managing blocks....2518
9.1.4 Programming blocks....2523
9.1.4.1 Program editor....2523
9.1.4.2 Programming code blocks....2544
9.1.4.3 Programming data blocks....2825
9.1.4.4 Programming PLC data types....2854
9.1.4.5 Using external source files....2863
9.1.5 Comparing PLC programs....2869
9.1.5.1 Basic information on comparing PLC programs....2869
9.1.5.2 Comparing blocks....2873
9.1.5.3 Comparing PLC tags....2895
9.1.5.4 Comparing PLC data types....2896
9.1.6 Compiling and downloading blocks....2898
9.1.6.1 Compiling blocks....2898
9.1.6.2 Downloading blocks....2903
9.1.6.3 Downloading blocks....2921
9.1.7 Protecting blocks....2929
9.1.7.1 Protecting blocks....2929
9.1.7.2 Setting up and removing block copy protection....2931
9.1.7.3 Setting up block know-how protection....2932
9.1.7.4 Opening know-how protected blocks....2933
9.1.7.5 Printing know-how protected blocks....2933
9.1.7.6 Changing a password....2935
9.1.7.7 Removing block know-how protection....2935

9.2 Displaying program information....2936
9.2.1 Overview of available program information....2936
9.2.2 Displaying an assignment list....2937
9.2.2.1 Introduction to the assignment list....2937
9.2.2.2 Layout of the assignment list....2938
9.2.2.3 Symbols in the assignment list....2939
9.2.2.4 Displaying an assignment list....2940
9.2.2.5 Setting the view options for the assignment list....2941
9.2.2.6 Filter options in the assignment list....2942
9.2.2.7 Defining filters for assignment list....2942

9.2.2.8 Filtering an assignment list....2943
9.2.2.9 Defining retentive memory areas for bit memories....2944
9.2.2.10 Enabling the display of retentive bit memories....2945
9.2.3 Displaying the call structure....2945
9.2.3.1 Introduction to the call structure....2945
9.2.3.2 Symbols in the call structure....2947
9.2.3.3 Layout of the call structure....2948
9.2.3.4 Displaying the call structure....2949
9.2.3.5 Setting the view options for the call structure....2950
9.2.3.6 Introducing the consistency check in the call structure....2950
9.2.3.7 Checking block consistency in the call structure....2951
9.2.4 Displaying the dependency structure....2952
9.2.4.1 Introduction to the dependency structure....2952
9.2.4.2 Layout of the dependency structure....2953
9.2.4.3 Symbols in the dependency structure....2954
9.2.4.4 Displaying the dependency structure....2954
9.2.4.5 Setting the view options for the dependency structure....2955
9.2.4.6 Introducing the consistency check in the dependency structure....2956
9.2.4.7 Checking block consistency in the dependency structure....2956
9.2.5 Displaying CPU resources....2957
9.2.5.1 Introducing resources....2957
9.2.5.2 Layout of the "Resources" tab....2959
9.2.5.3 Displaying resources....2960
9.2.5.4 Selecting the maximum load memory available....2961

9.3 Displaying cross-references....2961
9.3.1 General information about cross references....2961
9.3.2 Structure of the cross-reference list....2962
9.3.3 Displaying the cross-reference list....2963
9.3.4 Displaying cross-references in the Inspector window....2965

9.4 Testing the user program....2966
9.4.1 Basics of testing the user program....2966
9.4.2 Testing with program status....2967
9.4.2.1 Introduction to testing with program status....2967
9.4.2.2 Setting the call environment....2968
9.4.2.3 Testing in single step mode....2969
9.4.2.4 Switching test with program status on/off....2977
9.4.2.5 Editing blocks during the program test....2978
9.4.2.6 Modifying tags in the program status....2979
9.4.2.7 Switching display formats in the program status....2980
9.4.2.8 Special considerations when testing GRAPH programs....2980
9.4.2.9 Examples of program status display....2987
9.4.3 Testing with the watch table....2994
9.4.3.1 Introduction to testing with the watch table....2994
9.4.3.2 Layout of the watch table....2995
9.4.3.3 Basic mode and expanded mode in the watch table....2996
9.4.3.4 Icons in the watch table....2997
9.4.3.5 Creating and editing watch tables....2998
9.4.3.6 Entering tags in the watch table....3000
9.4.3.7 Monitoring tags in the watch table....3008
9.4.3.8 Modifying tags in the watch table....3013
9.4.4 Testing with the force table....3021

9.4.4.1 Introduction for testing with the force table....3021
9.4.4.2 Safety precautions when forcing tags....3023
9.4.4.3 Layout of the force table....3023
9.4.4.4 Basic mode and expanded mode in the force table....3024
9.4.4.5 Icons in the force table....3025
9.4.4.6 Open and edit force table....3026
9.4.4.7 Entering tags in the force table....3027
9.4.4.8 Monitoring tags in the force table....3034
9.4.4.9 Forcing tags in the force table....3038
9.4.4.10 Stop forcing tags....3045

9.5 Configuring alarms....3047
9.5.1 Introduction to alarm configuration....3047
9.5.2 Assigning alarm numbers....3047
9.5.3 Components of an alarm....3048
9.5.4 Available alarm blocks....3048
9.5.5 Alarm type and alarms....3050
9.5.6 Formal parameters, alarm data types and alarm blocks....3051
9.5.7 Layout of the alarm editor....3053
9.5.8 Creating and editing alarms....3054
9.5.8.1 Creating program alarms....3054
9.5.8.2 Editing program alarms in the alarm editor....3054
9.5.8.3 Editing program alarms in the program editor....3055
9.5.8.4 Deleting program alarms....3056
9.5.8.5 Creating and editing an alarm type....3056
9.5.8.6 Creating and editing an instance DB....3056
9.5.8.7 Creating user diagnostic alarms....3057
9.5.8.8 Editing user diagnostic alarms....3057
9.5.8.9 Deleting user diagnostic alarms....3058
9.5.8.10 Insert associated values in alarms....3058
9.5.8.11 Structure of associated values....3059
9.5.8.12 Examples of associated values....3060
9.5.8.13 Deleting associated values....3061
9.5.9 Creating and editing alarms....3062
9.5.9.1 Creating program alarms....3062
9.5.9.2 Editing program alarms in the alarm editor....3062
9.5.9.3 Editing program alarms in the program editor....3063
9.5.9.4 Deleting program alarms....3064
9.5.9.5 Creating and editing an alarm type....3064
9.5.9.6 Creating and editing an instance DB....3064
9.5.9.7 Insert associated values in alarms....3065
9.5.9.8 Settings when entering associated values....3066
9.5.9.9 Examples of associated values....3067
9.5.9.10 Deleting associated values....3067
9.5.10 Texts and attributes....3068
9.5.10.1 Keywords for alarm texts....3068
9.5.10.2 Entering texts....3068
9.5.10.3 Locking texts....3069
9.5.10.4 Locking attributes....3070
9.5.11 Text lists for alarms....3070
9.5.11.1 Basics of alarm text lists....3070
9.5.11.2 Editing text lists for alarms....3071
9.5.11.3 Integrating texts from text lists in alarms....3071

9.5.11.4 Example of integrating texts from text lists in alarms....3072
9.5.12 Alarm classes....3073
9.5.12.1 Creating alarm classes....3073
9.5.12.2 Editing alarm classes....3073

9.6 Using Team Engineering....3074
9.6.1 Exchanging data with Inter Project Engineering (IPE)....3074
9.6.1.1 Basics of Inter Project Engineering (IPE)....3074
9.6.1.2 Requirements for Inter Project Engineering (IPE)....3076
9.6.1.3 Procedure for working with Inter Project Engineering (IPE)....3077
9.6.1.4 Creating device proxy data....3079
9.6.1.5 Opening device proxy data....3080
9.6.1.6 Exchange of controller data via IPE file....3080
9.6.1.7 Exchange of controller data via project file....3081

9.7 Programming examples....3082
9.7.1 Programming examples (S7-1200, S7-1500)....3082
9.7.1.1 LAD programming examples....3082
9.7.1.2 FBD programming examples....3090
9.7.1.3 STL programming examples....3099
9.7.1.4 SCL programming examples....3110
9.7.2 Programming examples (S7-300, S7-400)....3115
9.7.2.1 LAD programming examples....3115
9.7.2.2 FBD programming examples....3124
9.7.2.3 STL programming examples....3134
9.7.2.4 SCL programming languages....3145

9.8 References....3150
9.8.1 References (S7-1200, S7-1500)....3150
9.8.1.1 General parameters of the instructions....3150
9.8.1.2 Basic instructions....3156
9.8.1.3 Extended instructions....4502
9.8.1.4 Technology....4823
9.8.1.5 Communication....5033
9.8.2 References (S7-300, S7-400)....5367
9.8.2.1 General parameters of the instructions....5367
9.8.2.2 Basic instructions....5374
9.8.2.3 Extended instructions....6407
9.8.2.4 Technology....6793
9.8.2.5 Communication....7351

10 Visualize processes....7961

10.1 Creating screens....7961
10.1.1 Basics....7961
10.1.1.1 Screen basics....7961
10.1.1.2 Device-specific functional scope of screens....7962
10.1.1.3 Basics....7964
10.1.1.4 Working with screens....7967
10.1.1.5 Working with Templates....7970
10.1.1.6 Working with styles....7976
10.1.2 Working with objects....7978
10.1.2.1 Overview of objects....7978
10.1.2.2 Options for Editing Objects....7981
10.1.2.3 Inserting an object....7982

10.1.2.4 Deleting an Object 7984
10.1.2.5 Positioning an object....7985
10.1.2.6 Resizing an object....7986
10.1.2.7 Selecting multiple objects....7988
10.1.2.8 Aligning objects....7990
10.1.2.9 Moving an object forward or backward....7991
10.1.2.10 Show objects outside the screen area....7992
10.1.2.11 Rotating objects....7993
10.1.2.12 Flipping objects....7995
10.1.2.13 Designing an object....7995
10.1.2.14 Designing the fill pattern....7996
10.1.2.15 Formatting graphics in an object....7999
10.1.2.16 Formatting text in an object....8000
10.1.2.17 Designing table-based objects....8001
10.1.2.18 Designing a border....8002
10.1.2.19 Defining color gradients....8004
10.1.2.20 Applying predefined styles....8005
10.1.2.21 Inserting multiple objects of the same type (stamping tool)....8006
10.1.2.22 Repositioning and resizing multiple objects....8008
10.1.2.23 External graphics....8008
10.1.2.24 Managing external graphics....8009
10.1.2.25 Storing an external image in the graphics library....8011
10.1.2.26 Working with object groups....8013
10.1.2.27 Configuring the keyboard access....8020
10.1.2.28 Examples....8022
10.1.3 Working with text lists and graphics lists....8026
10.1.3.1 Working with text lists....8026
10.1.3.2 Working with graphics lists....8033
10.1.4 Dynamizing screens....8042
10.1.4.1 Basics on dynamization....8042
10.1.4.2 Dynamization in the inspector window....8043
10.1.4.3 Dynamization with animations....8045
10.1.4.4 Dynamize with system functions....8054
10.1.5 Working with function keys....8056
10.1.5.1 Working with function keys 8056
10.1.5.2 Assigning function keys globally....8058
10.1.5.3 Local assignment of function keys....8060
10.1.5.4 Assigning a function key to a function....8061
10.1.5.5 Assigning operator authorization for a function key....8062
10.1.5.6 Assigning a function key to a graphic....8064
10.1.5.7 Configuring LED tags....8066
10.1.5.8 Example: Using function keys for screen navigation....8068
10.1.6 Working with layers....8069
10.1.6.1 Basics on working with layers....8069
10.1.6.2 Moving objects between layers....8070
10.1.6.3 Setting the active layer....8071
10.1.6.4 Show and hide layers....8072
10.1.6.5 Renaming layers....8073
10.1.7 Display and operating objects....8075
10.1.7.1 Device-Specific Nature of the Objects....8075
10.1.7.2 Objects....8076
10.1.8 Configuring screen navigation....8105

10.1.8.1 Basics for screen navigation....8105
10.1.8.2 Assign button with screen change....8106
10.1.8.3 Assign screen change to function key....8107

10.2 Working with Tags....8108
10.2.1 Basics....8108
10.2.1.1 Basics of tags....8108
10.2.1.2 Overview of HMI tag tables....8109
10.2.1.3 External tags....8110
10.2.1.4 Addressing external tags....8112
10.2.1.5 Internal tags....8114
10.2.2 Working with tags....8115
10.2.2.1 Creating tags....8115
10.2.2.2 Editing tags....8119
10.2.2.3 Configuring tags....8125
10.2.3 Working with arrays....8142
10.2.3.1 Basics on arrays....8142
10.2.3.2 Creating array tags....8144
10.2.3.3 Examples of arrays....8145
10.2.4 Working with cycles....8145
10.2.4.1 Cycle basics....8145
10.2.5 Displaying tags....8146
10.2.5.1 Outputting tag values in screens....8146
10.2.5.2 Configuring trend displays for values from the PLC....8147

10.3 Working with alarms....8148
10.3.1 Basics....8148
10.3.1.1 Alarm system in WinCC....8148
10.3.1.2 Alarm types....8149
10.3.1.3 Alarm states....8151
10.3.1.4 Alarm classes....8153
10.3.1.5 Acknowledgment....8154
10.3.1.6 Alarm groups....8156
10.3.1.7 Alarm number....8157
10.3.2 Working with alarms....8157
10.3.2.1 Alarm components and properties....8157
10.3.2.2 Configuring alarms....8159
10.3.2.3 Configuring alarm output....8171
10.3.2.4 Acknowledging alarms....8177
10.3.3 Operating alarms in Runtime....8180
10.3.3.1 Alarms in Runtime....8180
10.3.3.2 Simple alarm view, simple alarm window in runtime....8181
10.3.3.3 Alarm indicator in Runtime....8183
10.3.3.4 Acknowledging alarms....8184
10.3.4 Reference....8185
10.3.4.1 System functions for alarms....8185
10.3.4.2 System events....8185
10.3.5 Configuring system diagnostics....8207
10.3.5.1 System diagnostics basics....8207
10.3.5.2 System diagnostics views....8208
10.3.5.3 Configuring the system diagnostic view....8210

10.4 Working with recipes....8211
10.4.1 Basics....8211

10.4.1.1 Definition and applications....8211
10.4.1.2 Examples for using recipes....8212
10.4.1.3 Structure of recipes....8212
10.4.1.4 Displaying recipes....8214
10.4.1.5 Flow of data for recipes....8214
10.4.1.6 Synchronization of recipe data records with the PLC....8215
10.4.2 Elements and basic settings....8216
10.4.2.1 "Recipes" editor....8216
10.4.3 Displaying and editing recipes in Runtime....8219
10.4.3.1 Simple recipe view....8219
10.4.3.2 Behavior of the recipe view in Runtime....8220
10.4.4 Configuring recipes....8221
10.4.4.1 General configuration procedure....8221
10.4.4.2 Creating and Editing Recipes....8222
10.4.4.3 Configuring the display of recipes....8230
10.4.5 Using recipes in Runtime....8233
10.4.5.1 Using the simple recipe view....8233
10.4.6 Example....8239
10.4.6.1 Example of creating a recipe....8239

10.5 Configuring user administration....8241
10.5.1 Field of application of the user administration....8241
10.5.2 Form of the user administration....8241
10.5.3 Basics....8242
10.5.3.1 Users....8242
10.5.3.2 Users work area....8243
10.5.3.3 User groups....8244
10.5.3.4 User groups work area....8244
10.5.3.5 Settings for the user administration....8245
10.5.4 Setting up the user administration....8247
10.5.4.1 Basics on user administration....8247
10.5.4.2 Administering users for Runtime....8248
10.5.4.3 Managing users in Runtime....8255
10.5.4.4 Configuring access protection....8263
10.5.5 Reference....8264
10.5.5.1 Objects with access protection....8264
10.5.5.2 Default user groups and authorizations....8265
10.5.6 Examples....8265
10.5.6.1 Example: Configuring a button with logon dialog box....8265
10.5.6.2 Example: Logging the logon and logoff events....8266
10.5.6.3 Example of user management....8267

10.6 Working with system functions....8274
10.6.1 Basics....8274
10.6.1.1 System functions....8274
10.6.1.2 Use of system functions....8276
10.6.2 Working with function lists....8276
10.6.2.1 Basic of the functions list....8276
10.6.2.2 Properties of a function list....8277
10.6.2.3 Configuring a function list....8278
10.6.2.4 Editing a function list....8280
10.6.2.5 Executing a function list in Runtime....8281
10.6.3 Example....8281

10.6.3.1 Changing the operating mode on the HMI device with the current display....8281
10.6.4 Reference....8284
10.6.4.1 Function list....8284
10.6.4.2 Events....8348

10.7 Planning tasks....8364
10.7.1 Field of application of the Scheduler....8364
10.7.2 Working with tasks and triggers....8366
10.7.3 Basics....8367
10.7.3.1 Work area of the "Scheduler" editor....8367
10.7.3.2 Function list....8368
10.7.3.3 Triggers....8368
10.7.3.4 Planning tasks with event triggers....8369
10.7.4 Examples....8371
10.7.4.1 Example: Update user following change of user....8371

10.8 Communicating with PLCs....8373
10.8.1 Basics of communication....8373
10.8.1.1 Communication between devices....8373
10.8.1.2 Devices and networks in the automation system....8374
10.8.1.3 Data exchange using tags....8379
10.8.1.4 Data exchange using area pointers....8380
10.8.1.5 Communication drivers....8380
10.8.2 Editors for communication....8381
10.8.2.1 "Devices & networks" editor....8381
10.8.2.2 Network view....8382
10.8.2.3 Network data....8385
10.8.2.4 Diagnostics of online connections....8387
10.8.2.5 Device view....8388
10.8.2.6 Topology view....8390
10.8.2.7 Inspector window....8392
10.8.2.8 Hardware catalog....8394
10.8.2.9 Information on hardware components....8396
10.8.3 Networks and connections....8397
10.8.3.1 SIMATIC communication networks....8397
10.8.3.2 Configuring networks and connections....8402
10.8.4 Data exchange....8410
10.8.4.1 Data exchange using tags....8410
10.8.4.2 Data exchange using area pointers....8417
10.8.5 Device dependency....8423
10.8.5.1 Basic Panel....8423
10.8.6 Communicating with SIMATIC S7 1500....8429
10.8.6.1 Communication with SIMATIC S7 1500....8429
10.8.6.2 Communication via PROFINET....8430
10.8.6.3 Communication via PROFIBUS....8443
10.8.6.4 Data exchange....8452
10.8.6.5 Performance features of communication....8470
10.8.6.6 Configuring connections in the "Connections" editor....8472
10.8.7 Communicating with SIMATIC S7 1200....8477
10.8.7.1 Communication with SIMATIC S7 1200....8477
10.8.7.2 Communication via PROFINET....8478
10.8.7.3 Communication via PROFIBUS....8492
10.8.7.4 Data exchange....8501

10.8.7.5 Performance features of communication....8520
10.8.7.6 Creating connections in the "Connections" editor....8522
10.8.8 Communicating with SIMATIC S7 300/400....8531
10.8.8.1 Communication with SIMATIC S7 300/400....8531
10.8.8.2 Communication via PROFINET....8532
10.8.8.3 Communication via PROFIBUS....8545
10.8.8.4 Communication via MPI....8553
10.8.8.5 Data exchange....8561
10.8.8.6 Performance features of communication....8578
10.8.8.7 Creating connections in the "Connections" editor....8579
10.8.9 Communicating with SIMATIC S7 200....8591
10.8.9.1 Communication with SIMATIC S7 200....8591
10.8.9.2 Creating a connection to SIMATIC S7 200....8591
10.8.9.3 Parameters for the connection....8593
10.8.9.4 Data exchange 8600
10.8.9.5 Performance features of communication....8618
10.8.10 Communicating with SIMATIC LOGO!....8618
10.8.10.1 Communication with SIMATIC LOGO!....8618
10.8.10.2 Creating a connection to SIMATIC LOGO!....8619
10.8.10.3 Connection parameters....8621
10.8.10.4 Data exchange....8625
10.8.10.5 Performance features of communication....8629
10.8.11 Communication with other PLCs....8630
10.8.11.1 Communication with other PLCs....8630
10.8.11.2 Distinctive features when configuring....8631
10.8.11.3 Communication drivers....8631
10.8.11.4 Data exchange using area pointers....8745

10.9 Using global functions....8760
10.9.1 HMI device wizard basics....8760
10.9.2 Working with libraries....8761
10.9.2.1 Basics on libraries....8761
10.9.2.2 Overview of the library view....8763
10.9.2.3 Copy templates and types....8765
10.9.2.4 Libraries in WinCC....8766
10.9.2.5 Managing libraries....8767
10.9.2.6 Managing objects in a library....8775
10.9.2.7 Using types and their versions....8779
10.9.3 Using controller data from other projects....8783
10.9.3.1 Basics of Inter Project Engineering (IPE)....8783
10.9.3.2 Controller data....8784
10.9.3.3 Initializing a device proxy via an IPE file....8784
10.9.3.4 Initializing a device proxy via a project file....8790
10.9.4 Importing and exporting project data....8795
10.9.4.1 Importing and exporting project data....8795
10.9.4.2 Importing and exporting recipes....8796
10.9.4.3 Importing and exporting alarms....8800
10.9.4.4 Importing and exporting tags....8807
10.9.4.5 Importing and exporting text lists....8812
10.9.4.6 Importing and exporting project texts....8817
10.9.5 Using cross-references....8820
10.9.5.1 General information about cross references....8820
10.9.5.2 Displaying the cross-reference list....8820

10.9.5.3 Structure of the cross-reference list....8821
10.9.5.4 Displaying cross-references in the Inspector window....8822
10.9.5.5 Rewiring tags in the screens....8823
10.9.6 Managing languages....8826
10.9.6.1 Languages in WinCC....8826
10.9.6.2 Language settings in the operating system....8827
10.9.6.3 Operating system settings for Asian languages....8828
10.9.6.4 Setting project languages....8829
10.9.6.5 Creating one project in multiple languages....8832
10.9.6.6 Using language-specific graphics....8840
10.9.6.7 Languages in runtime....8844
10.9.6.8 Example of multilingual configuration....8851
10.9.7 Replacing devices....8854
10.9.7.1 Basics....8854
10.9.7.2 Device-specific functions....8855
10.9.7.3 Adjusting screens to the new device....8858
10.9.8 Copying between devices and editors....8863
10.9.8.1 Basics....8863
10.9.8.2 Copying and pasting....8865
10.9.8.3 Copying between different RT and ES versions....8868
10.9.9 Using WinCC version compatibility....8870
10.9.9.1 Basics on version compatibility....8870
10.9.9.2 Editing projects of a previous WinCC version....8872
10.9.9.3 Upgrading projects....8873
10.9.9.4 Upgrading a global library....8874
10.9.9.5 Changing between device versions....8875
10.9.9.6 Changing the device version....8876
10.9.10 Viewing memory card data....8877
10.9.10.1 Basics....8877
10.9.10.2 Working with backups....8878

10.10 Compiling and loading....8880
10.10.1 Compiling and loading projects....8880
10.10.1.1 Overview of compiling and loading projects....8880
10.10.1.2 Compiling a project....8882
10.10.1.3 Loading projects....8883
10.10.1.4 Runtime start....8886
10.10.2 Simulating projects....8887
10.10.2.1 Simulation basics....8887
10.10.2.2 Simulating a project....8888
10.10.2.3 Working with the tag simulator....8890
10.10.2.4 Simulation restrictions....8892
10.10.3 Servicing the HMI device....8892
10.10.3.1 Overview of HMI device maintenance (Basic Panels)....8892
10.10.3.2 ProSave....8893
10.10.3.3 Backup of HMI data....8894
10.10.3.4 Backing up and restoring data of the HMI device....8895
10.10.3.5 Updating the operating system....8896
10.10.3.6 Updating the HMI device operating system....8897
10.10.4 Reference....8899
10.10.4.1 Error messages during loading of projects....8899
10.10.4.2 Adapting the project for another HMI device....8900
10.10.4.3 Establishing a connection to the HMI device....8901

10.11 Operating in Runtime....8902
10.11.1 Basics....8902
10.11.1.1 Overview....8902
10.11.1.2 Tags in Runtime....8903
10.11.1.3 System functions in Runtime....8904
10.11.2 Commissioning projects....8904
10.11.2.1 Settings in the Runtime software....8904
10.11.2.2 Loading projects....8906
10.11.2.3 Starting Runtime on the Engineering Station....8907
10.11.2.4 Starting Runtime on the HMI device....8907
10.11.2.5 Testing a project....8908
10.11.2.6 Closing a project....8909
10.11.2.7 Backing up and restoring data of the HMI device....8910
10.11.3 Languages in runtime....8911
10.11.3.1 Languages in runtime....8911
10.11.3.2 Setting a runtime language....8911
10.11.3.3 Setting the font for a runtime language....8912
10.11.3.4 Configuring language switching....8913
10.11.3.5 Specific features of Asian and Eastern languages in runtime....8914
10.11.4 Operating projects....8915
10.11.4.1 Basics....8915
10.11.4.2 Operating objects....8922
10.11.4.3 Project security....8935
10.11.4.4 Operating alarms....8942
10.11.4.5 Operating recipes....8949

10.12 Performance features....8959
10.12.1 Engineering system....8959
10.12.2 Basic Panel....8961
10.12.3 Basic Panel 2nd Generation....8965
10.12.4 General technical data....8968
10.12.4.1 Permitted characters....8968
10.12.4.2 Recommended printers....8969
10.12.4.3 Memory requirements of recipes for Basic Panels....8970

10.13 Migrating to WinCC in the TIA Portal....8971
10.13.1 Overview of migration to WinCC V12....8971
10.13.2 Libraries....8971
10.13.3 Screens and templates....8973

11 Using technology functions....8977

11.1 Counting, measurement and position input....8977
11.1.1 Counting, measurement and position input (S7-1500)....8977
11.1.1.1 The basics of counting, measurement and position detection....8977
11.1.1.2 Using the High_Speed_Counter....9024
11.1.1.3 Using the technology module....9045
11.1.2 Counting and measurement (S7-300, S7-400)....9080
11.1.2.1 Counting and measuring....9080

11.2 Cam control (S7-300, S7-400)....9340
11.2.1 Cam control....9340
11.2.1.1 Cam control basics....9340
11.2.1.2 Functions of the FM x52....9354

11.2.1.3 Using FM 352....9358
11.2.1.4 Using FM 452....9433

11.3 PID control....9511
11.3.1 Principles for control....9511
11.3.1.1 Controlled system and actuators....9511
11.3.1.2 Controlled systems....9512
11.3.1.3 Characteristic values of the control section....9515
11.3.1.4 Pulse controller....9517
11.3.1.5 Response to setpoint changes and disturbances....9521
11.3.1.6 Control Response at Different Feedback Structures....9522
11.3.1.7 Selection of the controller structure for specified controlled systems....9530
11.3.1.8 PID parameter settings....9531
11.3.2 Configuring a software controller....9531
11.3.2.1 Overview of software controller....9531
11.3.2.2 Steps for the configuration of a software controller....9533
11.3.2.3 Add technology objects....9533
11.3.2.4 Configure technology objects....9534
11.3.2.5 Call instruction in the user program....9535
11.3.2.6 Downloading technology objects to device....9536
11.3.2.7 Commissioning software controller....9537
11.3.2.8 Save optimized PID parameter in the project....9538
11.3.2.9 Comparing values....9538
11.3.2.10 Parameter view....9541
11.3.2.11 Display instance DB of a technology object....9558
11.3.3 PID control (S7-1200, S7-1500)....9559
11.3.3.1 Using PID_Compact....9559
11.3.3.2 Using PID_3Step....9589
11.3.4 PID control (S7-300, S7-400, S7-1500)....9622
11.3.4.1 Using PID basic functions....9622
11.3.5 PID control (S7-300, S7-400)....9653
11.3.5.1 Using standard PID control....9653
11.3.5.2 Using modular PID control....9751
11.3.5.3 Using PID Self Tuner....9784
11.3.5.4 Basics on controller modules....9799
11.3.5.5 Using FM 355....9836
11.3.5.6 Using FM 355-2....9868
11.3.5.7 Using FM 455....9916
11.3.5.8 Using ET 200S 2PULSE....9950

11.4 Motion Control....10005
11.4.1 Motion Control (S7-1200, S7-1500)....10005
11.4.1.1 Using S7-1200 Motion Control....10005
11.4.1.2 Using S7-1500 Motion Control....10207
11.4.2 Motion Control (S7-300, S7-400)....10416
11.4.2.1 Position feedback basics....10416
11.4.2.2 Using SM 338....10419
11.4.2.3 Using ET 200S 1SSI....10428
11.4.2.4 Basics of positioning with function modules....10451
11.4.2.5 Using FM 351....10458
11.4.2.6 Using FM 451....10548
11.4.2.7 Using ET 200S 1PosU....10646
11.4.2.8 Using ET 200S 1STEP 5V....10711

11.4.2.9 Using the IM 174 10748
11.4.2.10 Using Easy Motion Control 10763

12 Using Team Engineering 10831

12.1 Shared commissioning of projects 10831
12.1.1 Basics for shared commissioning 10831
12.1.2 Requirements for shared commissioning 10833
12.1.3 Procedure for shared commissioning 10834
12.1.4 Rules for shared commissioning 10837

12.2 Exchanging data with Inter Project Engineering (IPE) 10842
12.2.1 Basics of Inter Project Engineering (IPE) 10842
12.2.2 Requirements for Inter Project Engineering (IPE) 10844
12.2.3 Procedure for working with Inter Project Engineering (IPE) 10845

13 Using online and diagnostics functions 10849

13.1 Displaying accessible devices 10849

13.2 Changing the device configuration online 10850

13.3 Connecting devices online 10851
13.3.1 General information about online mode 10851
13.3.2 View in online mode 10853
13.3.3 Establishing and canceling an online connection 10854
13.3.4 Connecting online with several devices 10856
13.3.5 Disconnecting online connections of multiple devices 10856

13.4 Backing up the software and hardware configuration of a device 10857
13.4.1 Creating a backup of a device 10857
13.4.2 Backing up a device configuration 10858
13.4.3 Restoring the software and hardware configuration of a device 10859

13.5 Configuring the PG/PC interface 10860
13.5.1 Online access 10860
13.5.2 Basics of assigning parameters for the PG/PC interface 10862
13.5.3 Showing or hiding interfaces 10863
13.5.4 Displaying and modifying interface properties 10863
13.5.5 Adding interfaces 10864
13.5.6 Setting parameters for the Ethernet interface 10865
13.5.6.1 Setting parameters for the Industrial Ethernet interface 10865
13.5.6.2 Displaying operating system parameters 10866
13.5.6.3 Connecting the PG/PC interface to a subnet 10866
13.5.6.4 Setting parameters for the Ethernet interface 10867
13.5.6.5 Assigning a temporary IP address 10868
13.5.6.6 Managing temporary IP addresses 10868
13.5.6.7 Resetting the TCP/IP configuration 10869
13.5.7 Setting parameters for the MPI and PROFIBUS interfaces 10869
13.5.7.1 Setting parameters for the MPI and PROFIBUS interfaces 10869
13.5.7.2 Setting MPI or PROFIBUS interface parameters automatically 10870
13.5.7.3 Setting parameters for the MPI interface 10871
13.5.7.4 Setting parameters for the PROFIBUS interface 10873
13.5.7.5 Overview of the bus parameters for PROFIBUS 10875
13.5.7.6 Resetting the MPI or PROFIBUS configuration 10877

13.6 Using the trace and logic analyzer function 10877

Preface....10877
13.6.1 Security information....10878
13.6.1 Description....10879
13.6.1.1 Supported hardware....10879
13.6.1.2 Recording of measured values with the trace function....10879
13.6.1.3 Trace configuration, recording and measurement....10880
13.6.1.4 Data storage....10881
13.6.2 Software user interface....10882
13.6.2.1 Project navigator....10884
13.6.2.2 Working area....10884
13.6.2.3 Device-specific area....10892
13.6.3 Operation....10892
13.6.3.1 Quick start....10892
13.6.3.2 Using the trace function - overview....10897
13.6.3.3 Calling the trace editor....10898
13.6.3.4 Trace handling....10898
13.6.3.5 Signal table....10902
13.6.3.6 Curve diagram....10903
13.6.4 Devices....10904
13.6.4.1 S7-1200/1500 CPUs....10904

13.7 Establishing a remote connection with TeleService....10917
13.7.1 Basics of working with TeleService....10917
13.7.1.1 Introduction to TeleService....10917
13.7.1.2 TeleService functionality....10918
13.7.1.3 Telephone book at TeleService....10919
13.7.2 Working with the phone book....10920
13.7.2.1 Basics on working with the phone book....10920
13.7.2.2 Structure of the phone book....10920
13.7.2.3 Symbols in the phone book....10921
13.7.2.4 Manage phone book....10922
13.7.3 Remote connections as dial-up connections....10928
13.7.3.1 Basics for establishing a dial-up connection....10928
13.7.3.2 Telephone networks and modems....10929
13.7.3.3 Access protection for dial-up connections....10932
13.7.3.4 TS adapter MPI....10937
13.7.3.5 TS adapter IE....10944
13.7.3.6 Establishing a dial-up connection to a remote system....10950
13.7.4 Remote VPN connections....10952
13.7.4.1 Basics for establishing a VPN connection....10952
13.7.4.2 Basics of CA certificates....10953
13.7.4.3 Installing CA certificates for VPN connections....10955
13.7.4.4 Deleting CA certificates for VPN connections....10958
13.7.4.5 Establishing a VPN connection to a remote system....10958
13.7.4.6 TS Adapter IE Advanced....10960
13.7.5 CPU controlled TeleService remote connections 10965
13.7.5.1 Overview of CPU controlled remote connections....10965
13.7.5.2 Establishing a connection from and to remote systens (PG-AS-remote coupling)....10966
13.7.5.3 Data exchange between remote systems (AS-AS-remote coupling)....10967
13.7.5.4 Send SMS from a system....10969
13.7.5.5 Send an email from a system....10971
13.7.6 Notes on troubleshooting....10973
13.7.6.1 General information on troubleshooting for modem problems....10973

13.7.6.2 Recording a log file for the modem....10973
13.7.6.3 Dial-up connection to the TS Adapter is not established....10974
13.7.6.4 Dial-up connection from the TS Adapter is not established....10975
13.7.6.5 Modem connection is interrupted....10976
13.7.6.6 Checklist for troubleshooting the modem....10977
13.7.6.7 Modem alarms....10977
13.7.6.8 Possible error messages with VPN connections....10978

14 Hardware documentation....10981

14.1 General information on the hardware documentation....10981

14.2 PLC....10981
14.2.1 SIMATIC S7-1200....10981
14.2.1.1 CPU....10981
14.2.1.2 Signal boards....10985
14.2.1.3 Communication boards....10986
14.2.1.4 Battery boards....10987
14.2.1.5 Digital input modules....10987
14.2.1.6 Digital output modules....10988
14.2.1.7 Digital input and digital output modules....10989
14.2.1.8 Analog input modules....10990
14.2.1.9 Analog output modules....10992
14.2.1.10 Analog input and analog output modules....10992
14.2.1.11 Communications modules....10993
14.2.1.12 Technology modules....10995
14.2.2 SIMATIC S7-1500....10995
14.2.2.1 CPU....10995
14.2.2.2 Additional modules for S7-1500....10997

14.3 Distributed I/O....10998
14.3.1 ET 200MP....10998
14.3.1.1 Interface modules....10998
14.3.1.2 Digital input modules....10998
14.3.1.3 Digital output modules....10999
14.3.1.4 Digital input and digital output modules....11001
14.3.1.5 Analog input modules....11001
14.3.1.6 Analog output modules....11002
14.3.1.7 Analog input and analog output modules....11002
14.3.1.8 Communications modules....11003
14.3.1.9 Power supply module....11004
14.3.1.10 Technology modules....11005
14.3.2 ET 200SP....11006
14.3.2.1 CPU....11006
14.3.2.2 Interface modules....11006
14.3.2.3 Digital input modules....11007
14.3.2.4 Digital output modules....11008
14.3.2.5 Analog input modules....11010
14.3.2.6 Analog output modules....11011
14.3.2.7 Communication modules....11012
14.3.2.8 Power modules....11012
14.3.2.9 Special modules....11012
14.3.2.10 Technology modules....11013

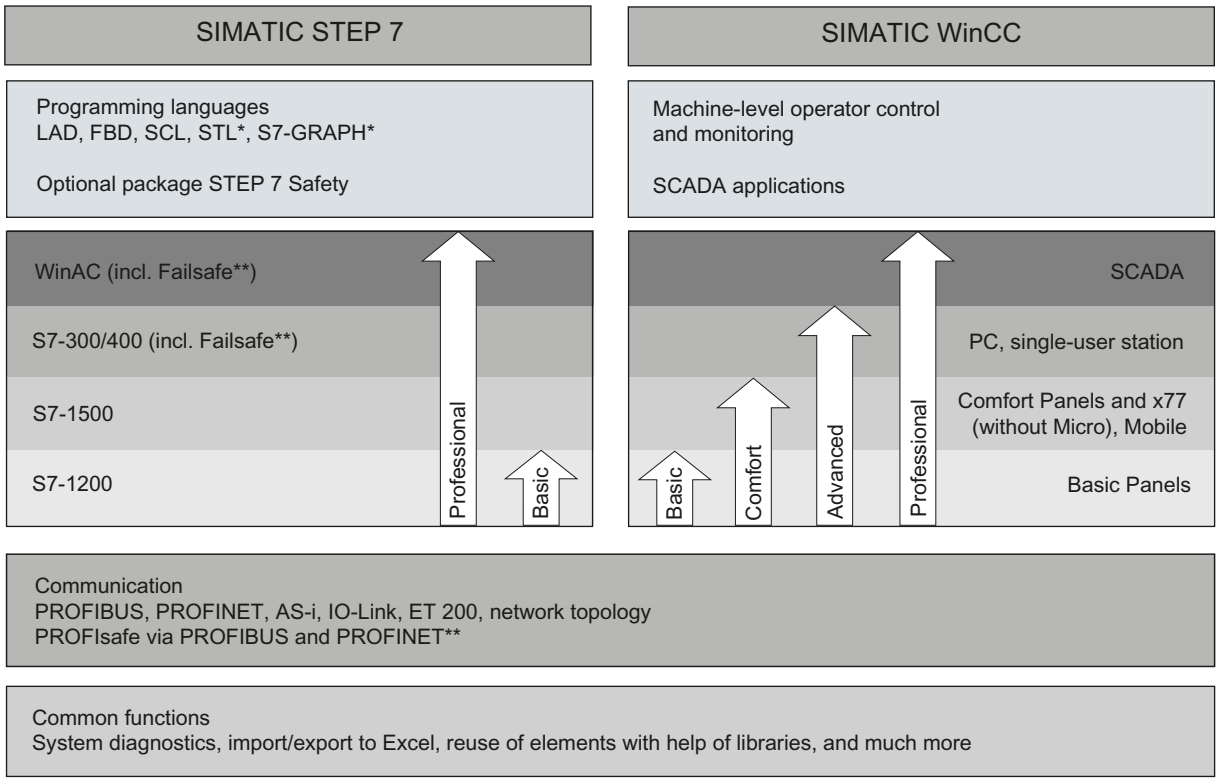
Index....11015

System overview of STEP 7 and WinCC 1

1.1 Scaling of STEP 7 and WinCC in the TIA Portal

Scope of performance of the products

The following graphic shows the scope of performance of the individual products of STEP 7 and WinCC:



* Only with STEP 7 Professional for S7-300/400/WinAC and S7-1500
** With installed optional package "STEP 7 Safety Advanced"

STEP 7

STEP 7 (TIA Portal) is the engineering software for configuring the SIMATIC S7-1200, S7-1500, S7-300/400 and WinAC controller families. STEP 7 (TIA Portal) is available in two editions, depending on the configurable controller families:

- STEP 7 Basic for configuring the S7-1200
- STEP 7 Professional for configuring S7-1200, S7-1500, S7-300/400 and WinAC

WinCC

WinCC (TIA Portal) is an engineering software for configuring SIMATIC Panels, SIMATIC Industrial PCs, and Standard PCs with the WinCC Runtime Advanced or the SCADA System WinCC Runtime Professional visualization software.

WinCC (TIA Portal) is available in four editions, depending on the configurable operator control systems:

- WinCC Basic for configuring Basic Panels
  WinCC Basic is included with every STEP 7 Basic and STEP 7 Professional product.
- WinCC Comfort for configuring all panels (including Comfort Panels, Mobile Panels)
- WinCC Advanced for configuring all panels and PCs with the WinCC Runtime Advanced visualization software
  WinCC Runtime Advanced is a visualization software for PC-based single-station systems. WinCC Runtime Advanced can be purchased with licenses for 128, 512, 2k, 4k as well as 8k PowerTags (tags with a process interface).
- WinCC Professional for configuring panels and PCs with WinCC Runtime Advanced or SCADA System WinCC Runtime Professional. WinCC Professional is available in the following editions: WinCC Professional for 512 and 4096 PowerTags as well as "WinCC Professional max. PowerTags".
  WinCC Runtime Professional is a SCADA system for structuring a configuration ranging from single-station systems to multi-station systems including standard clients or web clients. WinCC Runtime Professional can be purchased with licenses for 128, 512, 2k, 4k, 8k, and 64k PowerTags (tags with a process interface).

With WinCC (TIA Portal), it is also possible to configure a SINUMERIK PC with WinCC Runtime Advanced or WinCC Runtime Professional and HMI devices with SINUMERIK HMI Pro sl RT or SINUMERIK Operate WinCC RT Basic.

1.2 Options for STEP 7 Engineering System

Additional STEP 7 products

For applications with increased safety requirements, STEP 7 Professional can be supplemented with the STEP 7 Safety Advanced option.

When using the STEP 7 Safety Advanced option, you can configure failsafe I/O and program safety programs for F-CPUs in LAD and FBD.

1.3 Options for WinCC Engineering and Runtime systems

SIMATIC Panels as well as WinCC Runtime Advanced and WinCC Runtime Professional contain all essential functions for operator control and monitoring of machines or plants. Additional options allow you to extend the functionality in some cases to increase the range of available tasks.

Options for Comfort Panels, Mobile Panels, Multi Panels

The following possible extensions are available for Comfort Panels, Mobile Panels, and Multi Panels:

- WinCC SmartServer (remote operation)
- WinCC Audit (audit trail and electronic signature for regulated applications)

Note

In contrast to WinCC flexible 2008, functions from the WinCC flexible /Sm@rtService, WinCC flexible /Sm@rtAccess options as well as the WinCC flexible /OPC Server option are incorporated into the basic functionality.

Options for WinCC Runtime Advanced

The following possible extensions are available for WinCC Runtime Advanced:

- WinCC SmartServer (remote operation)
- WinCC Recipes (recipe system)
- WinCC Logging (logging of process values and alarms)
- WinCC Audit (audit trail for regulated applications)
- WinCC ControlDevelopment (extension by means of customer-specific controls)

Note

In contrast to WinCC flexible 2008, functions from the WinCC flexible /Sm@rtService, WinCC flexible /Sm@rtAccess options as well as the WinCC flexible /OPC Server option are incorporated into the basic functionality.

Options for WinCC Runtime Professional

The following possible extensions are available for WinCC Runtime Professional:

- WinCC Client (standard client for structuring multi-station systems)
- WinCC Server (supplements WinCC Runtime to include server functionality)
- WinCC Recipes (recipe system, formerly WinCC /UserArchives)
- WinCC WebNavigator (Web-based operator control and monitoring)
- WinCC DataMonitor (display and evaluation of process states and historical data)
- WinCC ControlDevelopment (extension by means of customer-specific controls)

Note

In contrast to WinCC V7, functions from the WinCC /OPC-Server and WinCC / ConnectivityPack options are incorporated into the basic functionality. Likewise, the basic functionality includes the Runtime API from WinCC /ODK.

Readme 2

2.1 General notes

The information in this readme file supersedes statements made in other documents.

Read the following notes carefully because they include important information for installation and use. Read these notes prior to installation.

Notes on the information system

The following function is already described in the information system, but is not available in TIA Portal V13:

- Hardware comparison of devices

Functions for S7-1200 as of firmware version V4

In the information system for the TIA Portal V13, functions are documented for S7-1200 as of firmware version V4 which are not yet available in the software.

The 64-bit data types will be available for an S7-1200 as of firmware version V4.

User-defined documentation

On the product DVD under "Documents\UserDocumentation", you will find instructions for creating and using user-defined documentation.

Display of interfaces via online access

If the Ethernet interfaces for online access are not displayed sporadically, install hotfix KB2588507 (for Microsoft Windows) from the Microsoft Support website.

Internet link: http://support.microsoft.com/kb/2588507 (http://support.microsoft.com/kb/2588507/en-us)

Display of Asian characters in the TIA Portal

If the Chinese TIA Portal is installed on a different Asian Windows operating system (for example Korean) and texts are not displayed correctly in the TIA Portal, select English as the "Language for non-Unicode programs" in the Windows Control Panel.

Instruction "GET_NAME: Read out name of a module"

According to the online help, the read name depends on the IO device type.

For an HMI Panel, however, it is not the name of the station that is output – as described in the online help – but the name of the interface module.

Online operation in hibernate mode

We recommend that you do not use the two options "Hibernate" and "Sleep" in online operation; if you do, communication problems could occur. If necessary, adapt the computer's energy options.

Installing new .Net versions or .Net service packs

- Close the TIA Portal before installing a new .Net version or a new .Net service pack on your programming device/PC.
- Restart the TIA Portal only after successful installation of the new .Net version or the new .Net service pack.

Notes on handling

- If a project in the list of projects last used is located on a network drive that is not connected, you may experience delays when opening the "Project" menu.
- When you insert a CPU, you may need to wait for some time if the project editor is open at the same time. This generally takes longer when you insert the first CPU in a newly created project. To be able to continue working more quickly, you should close the project editor before inserting a CPU.
- The alarm "Application is not responding" may appear in Windows 7 with functions that take a long time to run (loading the CPU for example). If this occurs, wait until the function has correctly finished.
- If you have installed a Microsoft mouse with IntelliPoint, you may find that it superimposes components over the buttons of the title bar. If this is the case, uninstall the IntelliPoint software from Microsoft.
- Enabling the "Virtual Desktop" options with NVIDIA graphics cards can cause problems. In this case, disable the "nView virtual desktop manager" of your NVIDIA graphics driver.

Using the TIA Portal via a remote desktop

In principle, it is possible to use the TIA Portal via a remote desktop connection. During configuration, you should, however, avoid disconnecting the connection to the desktop client. In rare cases, this can lead to the software user interface being blocked.

If you experience this blockage, follow these steps on the desktop client.

1. Open the Windows Task-Manager and close the "rdpclip.exe" process.
2. Type in "rdpclip.exe" in the command prompt to restart the process.

Note that the current content of the clipboard will be lost. You can, however, then continue configuration as usual. To be on the safe side, you should restart the TIA Portal at the next opportunity.

Notes on SIMATIC memory cards

The SIMATIC memory cards have been formatted and set up by Siemens for use with S7-1200 and S7-1500 modules. This format must not be overwritten; otherwise, the card will no longer be accepted by the modules. Formatting with Windows tools is therefore not permitted.

Behavior in case of open force job

Note that an active force job is retained even after you have loaded a new project to the SIMATIC memory card. This means you should first delete the active force job before you remove a SIMATIC memory card from the CPU and before you overwrite the card in the PC with a new project. If you use a SIMATIC memory card with unknown content, you should format the SIMATIC memory card before the next download.

Subnet addressing for CP 1613 and CP 1623

CP 1613 and CP 1623 are communication modules with microprocessor. To ensure secure management of communication links, these are processed on the module. The protocol stack in your PC is used for diagnostic purposes (SNMP, DCP). To allow both protocol stacks (i.e. CP 1613/23 Firmware and CP 1613/23 NDIS access) access to the same partners, is recommended to place both stacks of a module in the same subnet.

Editing a device IP address

Do not use the address range from 192.168.x.241 to 192.168.x.250 when editing a device IP address. If necessary, this address range is automatically assigned by the system to a programming device. Depending on the subnet mask, this applies also for all network classes.

Migrating projects with the TIA Portal

After the migration of hardware configurations and program blocks from earlier automation solutions, first check the functionality of the migrated project before you use it in productive operation.

Working with automatically synchronized network drives

Automatic synchronization after a network interruption can result in current (local) project data being stored as a "backup" on the network drive through user interactions. This could cause outdated project data to be loaded from the network drive when opening the project. For this reason, we do not recommend that you store TIA Portal projects on synchronized network drives.

If, however, you do work on synchronized drives, you can continue working locally in the event of a network interruption. In this case, you must always ensure that the TIA Portal application is closed while data is synchronized. The synchronization itself must be implemented in such a way that the current (local) project data replaces the project data on the network drive.

Entry of decimal places

With certain Windows language settings, it may occur that the entry of values with a comma as decimal place is not recognized (entering "1,23" leads to an error). Instead, use the international format ("1.23").

Access protection for memory cards in USB card readers

By improving the security mechanisms for online access and engineering of S7-1500 CPUs, the data storage on memory cards has been changed. For this reason, this version of STEP 7 cannot evaluate the passwords of the configured protection level when reading project data from memory cards that is accessed via a USB card reader. The changed behavior affects the memory cards for CPUs of the S7-1200/1500 series. Therefore, use physical safeguards to protect critical project data on memory cards for these devices.

Note

This restriction is not related to online access to devices or the know-how protection of program blocks.

Information on the TIA Portal in online support

Overview of the most important technical information and solutions for the TIA Portal in the Siemens Industry online support.

Internet link: TIA Portal in Siemens Industry online support (www.siemens.com/industry/onlinesupport/tiaportal)

All information on service and support in the Siemens Industry online support:

Internet link: Service and support in Siemens Industry online support (http://support.automation.siemens.com)

Here, you can also subscribe to the newsletter that provides you with latest information relating to your products.

Starting the TIA Portal

When you start the TIA Portal, Windows attempts to update the Certificate Revocation List (CRL) of "windowsupdate.com".

If no Internet access is available and there are multiple DNS servers, a timeout may occur and delay the start of the TIA Portal.

Compatibility with V12 SP1

An empty V12 SP1 project with the name "TIA_Portal_Project_V12.0.1.3.ap12" is installed in the installation directory under ..\Portal V13\SampleProjects to allow TIA Portal V13 to be opened in compatibility mode V12 SP1. This project must be copied to a local directory with full access before it can be used. For more information on this, refer to FAQ ID 835846.

FAQs on the TIA Portal

FAQs on the TIA Portal are available at FAQs (http://support.automation.siemens.com/WW/view/en/28919804/133000).

2.2 STEP 7 Professional

2.2.1 Security information

Upgrades and updates

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mind, Siemens' products and solutions undergo continuous development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Third-party products that may be in use should also be considered. For more information about industrial security, visit

http://www.siemens.com/industrialsecurity (http://www.industry.siemens.com/topics/global/en/industrial-security/Pages/Default.aspx)

To stay informed about product updates as they occur, sign up for a product-specific newsletter. For more information, visit

http://support.automation.siemens.com (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=99&lang=en&referer=%2fWW%2f&func=cslib.csinfo2&siteid=csius&extranet=standard&viewreg=WW&groupid=4000002)

Network settings

The following tables show the network settings of each product you need to analyze the network security and to configure external firewalls:

STEP 7 Professional					
Name	Port number	Transport protocol	Direction	Function	Description
ALM	4410*	TCP	Inbound/ outbound	License service	This service provides the complete functionality for software licenses and is used by both the Automation License Manager as well as all license-related software products.
RFC 1006	102	TCP	Outbound	S7 communicatio n	Communication to the S7 controller via Ethernet/PROFINET for programming and diagnostic purposes.

STEP 7 Professional					
DCP	---	Ethernet	Outbound	PROFINET	The DCP protocol (Discovery and Basic Configuration Protocol) is used by PROFINET and provides the basic functionality for locating and configuring PROFINET devices.
SNMP	161	UDP	Outbound	PROFINET	The SNMP client functionality is used by STEP 7 to read status information from PROFINET devices.
* Default port that can be changed by user configuration					

WinCC ES Basic (without simulation)					
Name	Port number	Transport protocol	Direction	Function	Description
ALM	4410*	TCP	Inbound/ outbound	License service	This service provides the complete functionality for software licenses and is used by both the Automation License Manager as well as all license-related software products.
HMI Load	1033	TCP	Outbound	HMI Load (RT Basic)	This service is used to transmit images and configuration data to Basic Panels.
* Default port that can be changed by user configuration					

Simulation RT Basic					
Name	Port number	Transport protocol	Direction	Function	Description
HMI Load	1033	TCP	Inbound	HMI Load (RT Basic)	This service is used to transmit images and configuration data to Basic Panels.
Ethernet/ IP	44818	TCP	Outbound	Ethernet/IP channel	The Ethernet/IP protocol is used for connections to Allen Bradley PLCs.
	2222	UDP	Inbound	Ethernet/IP channel	The Ethernet/IP protocol is used for connections to Allen Bradley PLCs.
Modbus TCP	502	TCP	Outbound	Modbus TCP channel	The Modbus TCP protocol is used for connections to Schneider PLCs.
RFC 1006	102	TCP	Outbound	S7 channel	Communication to the S7 controller via Ethernet/PROFINET
Mitsubishi MC	5002	TCP	Outbound	Mitsubishi MC channel	The Mitsubishi protocol is used for connections to Mitsubishi PLCs.

2.2.2 News

News in the TIA Portal

To keep you up-to-date with the latest news, Siemens Industry Online Support now offers a website with the latest news on a variety of topics of the TIA Portal.

All news on STEP 7 Professional is available at a glance at this link:

http://support.automation.siemens.com//WW/view/84047138 (http://support.automation.siemens.com/WW/view/en/84047138)

2.2.3 Notes on use

Contents

Information that could not be included in the online help and important information about product characteristics.

Online operation

The simultaneous online operation of STEP 7 V5.5 or earlier and STEP 7 Professional V13 has not been approved.

Simultaneous online connections on an S7-1200 CPU

It is not possible to establish an online connection from multiple TIA Portal instances simultaneously to the same S7-1200 CPU.

Parallel use

If the term "Combo" appears in the name or license key of the software after installation, the use of the following products/versions is permitted in accordance with Article 1.6 of the General Terms and Conditions (see also setup text):

- STEP 7 Professional 2010
- STEP 7 Professional 2006

Configuring and assigning module parameters

You will find an overview of the modules that can be configured and assigned parameters with STEP 7 Professional V13 at http://support.automation.siemens.com (http://support.automation.siemens.com/WW/view/en/28919804/133000).

Documentation on S7-PLCSIM

S7-PLCSIM is installed together with STEP 7 Professional. The documentation is saved centrally.

You can find the manuals under: C:\Programs\Siemens\Documentation\Product notes

You can find the readme files under: C:\Programs\Siemens\Documentation\Readme

PLC migration

No instruction profile should be activated during PLC migration.

Deactivate your instruction profile before you perform the PLC migration. You can then reactivate the profile again.

Removing/inserting the memory card

After removing or inserting a memory card, always perform a memory reset on the CPU in order to restore the CPU to a functional condition.

Deleting data logs on the memory card with S7-1500 CPUs

If you delete data logs (csv files that were generated with data log applications) on the memory card via the file explorer on the PC, this could result in undefined behavior of the data log during further operation. The data logs might not be removed completely by the deletion procedure given above.

Remedy: To delete data logs reliably, use the "DataLogDelete" instruction in the user program or format the memory card with the help of the online functions in the CPU.

Deleting a WinAC RTX from a SIMATIC PC

A controller application WinAC RTX with an assigned onboard CP interface in an interface slot (IF) cannot be deleted from a SIMATIC Box PC/Panel PC/Rack PC. If you wish to delete the WinAC RTX, you must first remove the onboard CP interface from the interface slot. To do this, the onboard CP interface must be selected in the network or device view and a different option button (for example, "Windows only" or "SIMATIC PC station") must be activated in the Inspector window under "CP assignment" instead of the option "SIMATIC WinAC".

Removing and inserting Ethernet modules

If Ethernet modules are removed and re-inserted during operation, you must boot the PC; otherwise, the "Accessible devices" functionality in STEP 7 or NCM PC will not display all devices. While the PC boots, Ethernet modules must be activated.

Comparing project data

The comparison functions (online/offline, offline/offline) currently do not take hardware into consideration.

Loading project data with TIA Portal V12 and V13 (S7-1200)

If you load the project data of an S7-1200 CPU with the TIA Portal V13, you can no longer use TIA Portal V12 to access this data. To do this, first restore the factory settings of the CPU. Read the additional information on this in the online help under "How to reset a CPU to factory settings".

Using project data of distributed IO-Link master modules from TIA Portal V12 in V13

The following procedure applies if you are using distributed IO-Link master modules in TIA Portal V12 that are not GSD devices and were configured with PCT: To continue using your project data from TIA Portal V12.0 in TIA Portal V13.0, you have to export them in the PCT before you upgrade the project. After the upgrade, you must import the project data once again using the PCT.

Compatibility

The device configuration and program of an S7-1200 CPU must always be configured with the same STEP 7 version. Usually, the TIA Portal makes sure that no version conflicts occur by outputting appropriate notifications during loading to the device.

This automatic verification is not possible with S7-1200 CPUs with firmware version V1.x. In this case, users themselves must ensure that no version conflicts occur.

2.2.4 Editing devices and networks

2.2.4.1 General information on devices and networks

Contents

General information on devices and networks.

PROFIBUS connection of the I/O to S7-300/S7-400/WinAC

When connecting the distributed I/O of the ET 200MP/SP via PROFIBUS DP to S7-300/S7-400/WinAC controller, no I&M data and therefore no diagnostics can be displayed.

To work round these restriction, the two slaves can also be connected using GSD media.

Operation of IM 155-6 DP HF with S7-300/400, ET 200S, ET 200pro or WinAC

The ET 200SP interface module IM 155-6 DP HF with the order number 6ES7 155-6BU00-0CN0 cannot be operated behind an S7-300/400, ET 200S, ET 200pro or WinAC CPU on PROFIBUS. This configuration cannot be compiled.

S7 PCT IO-Link

The S7 Port Configuration Tool is available for free download at the following link.

http://support.automation.siemens.com/WW/view/37936752 (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=0&lang=en&referer=%2fWW%2f&func=cslib.csinfo&siteid=csius&groupid=4000002&extranet=standard&viewreg=WW&nodeid0=33102519&objaction=csopen)

2.2.4.2 Use of modules on the S7-1200

Contents

Information that could not be included in the online help and important information about product characteristics.

Use of modules on the S7-1200

The modules listed below are not supported on the S7-1200.

Family	Module	Order number
S7-300 FMs	SM 338	6ES7 338-4BC01-0AB0
	FM 350-1	6ES7 350-1AH03-0AE0
	FM 350-2	6ES7 350-2AH00-0AE0, 6ES7 350-2AH01-0AE0
	FM 351	6ES7 351-1AH01-0AE0, 6ES7 351-1AH02-0AE0
	FM 352	6ES7 352-1AH02-0AE0
	FM 355 S	6ES7 355-1VH10-0AE0
	FM 355 C	6ES7 355-0VH10-0AE0
	FM 355-2 C	6ES7 355-2CH00-0AE0
	FM 355-2 S	6ES7 355-2SH00-0AE0
S7-300 PtP-CP	CP 340	6ES7 340-1AH02-0AE0, 6ES7 340-1BH02-0AE0, 6ES7 340-1CH02-0AE0
	CP 341	6ES7 341-1AH01-0AE0, 6ES7 341-1AH02-0AE0, 6ES7 341-1BH01-0AE0, 6ES7 341-1BH02-0AE0, 6ES7 341-1CH01-0AE0, 6ES7 341-1CH02-0AE0
Network component	Diagnostics repeater	6ES7 972-0AB01-0XA0
ET 200S	1 Count 24 V	6ES7 138-4DA04-0AB0
	1 Count 5 V	6ES7 138-4DE02-0AB0
	1 Step 5 V	6ES7 138-4DC00-0AB0, 6ES7 138-4DC01-0AB0
	2 pulses	6ES7 138-4DD00-0AB0, 6ES7 138-4DD01-0AB0
	1 SI	6ES7 138-4DF01-0AB0
	1 SI Modbus	6ES7 138-4DF11-0AB0
	1 SSI	6ES7 138-4DB02-0AB0, 6ES7 138-4DB03-0AB0
	1 Pos Universal	6ES7 138-4DL00-0AB0

Family	Module	Order number
	SIWAREX	7MH4910-0AA01, 7MH4912-0AA01, 7MH4920-0AA01
ET 200M	SIWAREX	7MH4 900-2AA01, 7MH4 900-3AA01, 7MH4 950-1AA01, 7MH4 950-2AA01

Loading S7-1200 module comments to the PG/PC

In central configurations with S7-1200, comments of modules, submodules and signal boards are not loaded. With CPs/CMs, only the comments of the IE interface or DP interface are loaded. In distributed configurations with ET 200SP or ET 200MP, only the comment of the channels is loaded from the I/O modules.

2.2.4.3 Replacing ET 200S positioning modules

Contents

Information that could not be included in the online help and important information about product characteristics.

Replacing ET 200S positioning modules

This information relates to the positioning module "1 Step 5V" (6ES7 138-4DC00-0AB0) from a project which was created with TIA Portal V11.0. When replacing these modules from the TIA Portal V11.0 with a new version of these modules, the parameter settings are reset to the default values.

This is the case with one of the following procedures:

- Replace the positioning module 6ES7 138-4DC00-0AB0 with its successor module 6ES7 138-4DC01-0AB0 by means of a device exchange.
- Updating the module version using the appropriate button in the device properties in the Inspector window.

2.2.4.4 CP 343-2 on SIMATIC S7 Embedded Controller EC31-RTX

Contents

Information that could not be included in the online help and important information about product characteristics.

CP 343-2 on SIMATIC S7 Embedded Controller EC31-RTX

The module AS-Interface CP 343-2 (order no.: 6GK7 343-2AH01) can be inserted in an expansion rack of the SIMATIC S7 Embedded Controller EC31-RTX (order no.: 6ES7 677-1DDxx-0BB0), but the CP 343-2 cannot be operated with the EC31-RTX.

2.2.4.5 Notes on online and diagnostics

Contents

Information that could not be included in the online help and important information about product characteristics.

Hardware detection followed by online connection

When the "Online > Hardware detection" command is performed for an unspecified CPU, the online configuration is not loaded from the CPU. If you do not load the configuration resulting from the hardware detection to the CPU, the device and network views will always show a difference between the offline and online configurations. It will appear that there are different configurations in the online and diagnostic views, although the MLFBs are identical in the actual CPU and the offline CPU.

2.2.4.6 Network components

Network components

CP 1242-7

Copying the CP 1242-7 into another project

If you copy a CP 1242-7 from one project into another project, the following parameters in the parameter group "CP identification" are changed on the target station:

- Project number of the CP
- Station number of the CP

Download to device

Only use the "Download to device" function with the CP 1242-7 via a TeleService connection as follows:

1. Select the CP in STEP 7.
2. Select the "Online" > "Download to device" menu.
3. In the "Extended download" dialog that appears, select the TeleService interface.
4. Download the project data from the "Extended download" dialog.

Upload from device

The function "Upload from device" is not supported by the CP 1242-7.

S7-1500 CM/CP

CM 1542-1: Remove retentive storage of the IP address if there are duplicate addresses

The IP address and the device name of the CM 1542-1 remain installed retentively:

If, for example during startup, the CM detects a duplicate address in another network, the CM is not connected to the network. The CM changes to RUN and cannot be reached via the Ethernet interface.

To be able to connect the CM to the network, you can remove the retentively stored IP address as follows:

1. Remove the memory card of the CPU.
2. Using DCP with the Primary Setup Tool (PST) and with the CPU in STOP, set the IP address of the CM to 0.0.0.0 without configuration.
   You have removed the retentively stored IP address of the CM. The CM can be connected into the network.
3. Insert the memory card in the CPU again.

PC CPs

PC CPs for VMware vSphere Hypervisor ESXi

The following communications processors have been released for operation with VMware vSphere 5.1 Hypervisor ESXi:

- CP 5622
- CP 5711
- VMware E1000 network adapter

The following communications processors have not been released for operation with VMware vSphere 5.1 Hypervisor ESXi:

- CP 5611
- CP 5612
- CP 5621
- CP 5512

Other communications processors such as the CP 1613, CP 1623, CP 5613, CP 5614, CP 5623, CP 5624 are released following installation of the SIMATIC NET PC software. You will find information on this in the readme file of the SIMATIC NET PC software.

Security modules

Migration of projects with Ethernet CPs and activated security functions

In STEP 7 V5.5 projects that contain Industrial Ethernet CPs with activated security functions, the security settings are disabled during migration to STEP 7 Professional.

If necessary, follow these steps after the migration:

1. Activate the security functionality.
2. Configure the required security settings.

Use of symbolic names when configuring security functions

The use of symbolic names during the configuration of Security functions is not supported.

Migration of IP access protection lists when activating the security functions

An active IP access protection list is converted to firewall rules when security functions are activated. These rules are visible in advanced firewall mode and can be adjusted there. The advanced firewall mode is activated automatically.

Security online diagnostics of S7 CPs

Security online diagnostics of a CP with security capability is only possible if the online connection is established directly via the CP. If the online connection from STEP 7 to the station was established via the CPU, no security online diagnostics of the CP is possible.

To establish an online connection directly via the CP, you need to enter the IP address of the CP in the "Station address" input box under the entry "Online access" in online diagnostics. Only then is security online diagnostics of the CP possible.

IKE mode

When negotiating the key in phase 1 the IKE mode "Main" should be preferred. This mode normally provides a reliable procedure compared with the "Aggressive" mode. One reason for using the "Aggressive" setting is when you have VPN groups with different pre-shared keys since only one pre-shared key is supported with the "Main" setting.

The IKE "Aggressive" mode should not be used in conjunction with certificates. In the IKE "Aggressive" mode, only use pre-shared keys.

The SOFTNET Security Client only supports the IKE "Main" mode.

A security module must not be used in VPN groups that use different IKE modes.

Downloading a configuration to an S7-300/S7-400 station via a VPN tunnel

When you download a configuration via the gigabit interface of a CP x43-1 Advanced to an S7-300/S7-400 station, the path via which the download takes place is stored in the project. If the project is then downloaded via a VPN tunnel established between a SCALANCE S module and the CP x43-1 Advanced, the download fails due to the changed download path.

To download via the VPN tunnel, follow the steps below:

1. Using the "Go online" button, connect to the gigabit interface of the CP x43-1 Advanced.
2. Disconnect the online connection to the CP x43-1 Advanced.
3. Download the project to the station via the gigabit interface of the CP x43-1 Advanced.

VPN tunnel establishment by the CP 1543-1

The CP 1543-1 can only establish VPN tunnel connections with the authentication method "pre-shared key" if its VPN connection partner is also a CP 1543-1. The establishment of VPN tunnel connections by the CP 1543-1 to all other VPN connection partners is only possible with the "certificate" authentication method.

Media converters

When you migrate a project, you need to reconfigure the media converters.

2.2.5 Programming a PLC

2.2.5.1 General notes on PLC programming

Contents

Information that could not be included in the online help and important information about product characteristics.

Information about network security

For communications access between the TIA Portal and CPU or between HMI (except for HMI access using "GET/PUT communication") and CPU, there are integrated security functions. The provide greater protection from manipulation and higher access protection. To protect against unauthorized network access to a CPU with standardized communications access such as "GET/PUT", "TSEND/TRCV", "Modbus", "FETCH/WRITE", you should also take suitable additional measures (e.g. cell protection concept).

Renumbering PLC data types

For better performance, the CPUs of the S7-1200 module process PLC data types with numbers. The user is not aware of these numbers, because the system processes the numbers independently. This means number conflicts are automatically resolved. When using know-how protected blocks that use PLC data types, it may be useful to set up your own numbering system (recommended > 5000) because the automatic resolution of numbering conflicts may result in a password prompt (compilation required).

To renumber the default number of a PLC data type, follow these steps:

1. Open the project library in the "Libraries" task card.
2. Drag the compilable PLC data type to the "Types" folder.
   The "Add type" dialog opens.
3. Enter the properties of the new type.
4. Click "OK" to confirm.

5. Right-click the PLC data type.
   The shortcut menu opens.
6. Select "Edit type" in the shortcut menu.
7. Open the "PLC programming" task card once again.
   The extension "in testing" is now added to the name of the PLC data type.
8. Right-click the PLC data type.
   The shortcut menu opens.
9. Select "Properties" in the shortcut menu.
10. Change the number of the PLC data type in the "General" section.
11. Select "Release version" from the shortcut menu of the library.

The PLC data type now has the new number and can be used. The assigned number is retained even if the type of the PLC data type is revoked.

Generating external sources from blocks

When generating external sources from blocks, the changes made directly in the block interface to the default values of PLC data types are not exported to the sources. This means these values are not available when the sources are imported once again. The default values are applied instead. To prevent this loss of data for the modified default values, the changes must be made directly in the PLC data type and not in the block interface. In this case, the changes are also exported when generating external sources.

Indirect indexing of ARRAY components of data type bit string in SCL

In TIA Portal V13, you can address the components of an ARRAY for a CPU of the S7-1200/1500 series in SCL with a tag of the data type BYTE, WORD, DWORD or LWORD as index in addition to a tag of the integer data type if the IEC check is not set. This is not possible for a CPU of the S7-300/400 series.

The result is an incompatibility with TIA Portal V12 SP1 in which indirect indexing with the data type bit string is also permitted on a CPU of the S7-300/400 series.

Declaring internal parameters of a GRAPH block as retentive (S7-1500 < V1.5)

When you declare the internal parameters of a GRAPH block as retentive, the sequence control remains in the current status and continues from the point at which it was interrupted after a change to "RUN". Please note that supervisions should not be used if you activate the new parameter "Retentive internal parameters" in a GRAPH block.

Explicit data type conversion in SCL (S7-1200/1500)

As of TIA Portal V13, the string is displayed with a leading sign during explicit data type conversion of SINT/INT/DINT/LINT_TO_STRING or WSTRING in SCL and transferred aligned to the left.

The result is an incompatibility with TIA Portal V12 SP1, in which the string was transferred aligned to the right during conversion.

Functions from the global library "Long Functions" with PLCSIM S7-1500

The compilation process links the functions of the S7-1500 CPU family to the CPU to protect them from manipulation. Function from the global library "Long Functions" that are used in the CPU and have such a link cannot be loaded to a simulation. However, it is possible to replace such elements with their originals from the global library. These are not linked to a CPU.

Use of the I/O access error OB with an S7-1500 CPU

If you are using the I/O access error OB with an S7-1500 CPU and the check box "Optimized block access" under "Properties > Attributes" is selected (default setting), the following applies to the "Area" tag of the start information: "Area" is the ID for the area in which the incorrect access occurred and can have the following values:

- B#16#01: Direct access to input
- B#16#02: Direct access to output
- B#16#81: Access to process image input
- B#16#82: Access to process image output

Loading inconsistent programs to an S7-1500 CPU

In TIA Portal, it is not possible to download inconsistent programs to an S7-1500 CPU without a consistency check. During the loading process, all blocks of the program are implicitly checked and are compiled again in the event of inconsistencies. If, however, there are programs on your CPU which were loaded with earlier versions of STEP 7, these programs could demonstrate inconsistencies.

In this case, note the following:

If you load an inconsistent program from a device, you will not be able to load the program back to an S7-1500 CPU unchanged afterwards, because a consistency check always takes place during the loading process and existing inconsistencies are corrected.

Loading data blocks from a device

Please note that the two CPU families handle the values of tags differently when downloading blocks from a device.

- S7-1200/1500
  CPUs of the S7-1200 series store the start values with which a data block is loaded to the CPU in the load memory. If you load a data block from an S7-1200 CPU, these start values are applied from the load memory to the offline data block again. Even if you have made value changes in the load memory during runtime by means of the "WRIT_DBL" instruction, the values originally loaded to the device are applied to the offline data block.
  In order to apply the current monitored values from the CPU, you can use the function "Snapshot of the monitored values". You can find detailed information on this in the information system, in the section "Programming data blocks".
- S7-300/400
  CPUs of the series S7-300/400 also store the start values with which a data block is loaded to the CPU in the load memory. However, they cannot read the values back from the load memory. If you load a data block from a device, the current monitored values are loaded from the CPU and entered in the "Start values" column in the offline data block.

Start value behavior during "Upload from device" (S7-1500)

The start values that you have changed with the instruction "WRIT_DBL: Write to data block in the load memory" will be lost during execution of the action "Upload from device".

Importing organization blocks from external STL/SCL sources

CPUs of the S7-300/400 series accept organization blocks with any name during import. If you import an OB with a standard name (for example, CYCL_EXC or CYC_INT5), the correct OB class is automatically assigned. If you import an OB with an unknown name, an OB0 is created. You can change this number after the import and hereby assign it to the desired OB class. To change to the OB number, select "Properties" in the shortcut menu.

When importing from external sources to a CPU of the series S7-1200/1500, only organization blocks with standard names are accepted. An OB with an unknown name is invalid after the import and is identified with a no parking character. You should, therefore, use a text editor to change the name of the organization block in the title bar of the external source before the import into the TIA Portal.

Process image of PTO/PWM outputs

Do not use PTO/PWM outputs in the process image (for example, for access in the user program, for online functions or in HMI). The update rate of the process image is much slower than the rate of the signal changes. The display in the process image therefore does not reflect the signal flow.

Moving system blocks

Moving objects from the "System blocks" folder to the "Program blocks" folder is not permitted.

Monitoring blocks in LAD and FBD

If the start of the current path is outside the visible range, it may not be possible to determine the input value. In this case, the current path is shown grayed out.

Resetting registration requirements for GRAPH blocks

When errors and faults are reset via the "REG_EF" input parameter, the following differences exist between the CPU families with respect to resetting the registration requirement:

- S7-300/400
  The registration requirement is retained until an interlock or supervision error occurs in a step. Only then is the "R1" event triggered, the associated action executed, and the registration requirement reset.
- S7-1500
  The registration requirement is always reset, regardless of whether interlock or supervision errors occur in the current cycle.

Avoid using PLC data types generated by the system in libraries

Some instructions generate their own PLC data types during instancing which are saved in the "PLC data types" project folder. However, you should not use these system-generated PLC data types in any library, because they may be recreated by the system at any time and may result in an unfavorable system behavior.

Note on data consistency in S7 communication and Open User Communication

In the S7-1500 module series, you can change parameters during ongoing operation with the instructions of S7 communication as well as with Open User Communication. Changed parameters are used immediately, even if a job is still active. Both can lead to data inconsistency! To avoid data inconsistencies, you should never change parameters while a job is running.

Using global data blocks in assignments

It is not possible to assign the contents of a global data block to a structurally identical data block, e.g. using a move box.

2.2.5.2 Instructions

Contents

Information that could not be included in the online help and important information about product characteristics.

Notes on the information system

The following function is already described in the information system, but is not available in TIA Portal V13:

- "ASI_CTRL" instruction for CPUs of the S7-1200/1500 modules

Instruction "PE_WOL: Starting and stopping energy-saving mode via WakeOnLan" (S7-1500)

This instruction only supports devices with a device number lower than 256. Devices with a higher device number are ignored.

Instruction "TMAIL_C: Transfer email"

In addition to the values described in the online help, the RET_VAL parameter can also have the following value:

Error code (W#16#...)	Explanation
8555	If multiple addresses were entered, these need to be separated by ",".

Instructions "BLKMOV: Move block", "UBLKMOV: Move block uninterruptible" and "FILL: Fill block"

In addition to the values described in the online help, the RET_VAL parameter can also have the following value:

Error code (W#16#...)	Explanation
8352	The WSTRING, WCHAR and BOOL data types are not supported.

Instruction "GET_IM_DATA: Read the identification and maintenance data" (S7-1500)

The "IM0_DATA" structure is not available as a system structure for reading out I&M0 data in TIA Portal V13. Instead, you can use the data structure "ARRAY of BYTE" to store the data.

Error code 0x807F at parameter RET_VAL

The S7-1500 CPUs can execute a maximum of 10 instances of the "Read the identification and maintenance data" instruction in parallel. Once this number has been exceeded, the output parameter RET_VAL assumes the value 0x807F and the instruction is not executed.

Remedy:

- Avoid more than 10 parallel calls of the instruction.
- Repeat the call at a later time, as this is a temporary error.

Instruction "Gen_UsrMsg: Generate user diagnostic alarms" (S7-1500)

The "Gen_UsrMsg" instruction described in the help is not available in V13.

Using instructions with parameters of type VARIANT in code blocks with different access types (S7-1500)

Code blocks (FBs/FCs) and data blocks (DBs) can be created with different access types ("standard" and "optimized"). In code blocks, you can call any instructions. Certain instructions (for example, "WRIT_DBL" and "READ_DBL") use pointers of type VARIANT at input and output parameters to address data blocks.

Ensure that you do not use these instructions in programs in which code blocks of different access types are called reciprocally. This could cause the following to occur:

- A structure from a standard data block is directly or indirectly passed to an optimized code block, which forwards this structure directly or indirectly to one of the blocks mentioned above.
- The reverse scenario, whereby a structure from an optimized code block is directly or indirectly passed to a standard data block, which forwards this structure directly or indirectly to one of the blocks mentioned above.

Output parameters of the S7-1500 instructions

If an access error or a programming error occurs while the instruction is being executed, the value of the output is undefined.

2.2.5.3 Testing the user program

Testing with the watch table

Contents

Information that could not be included in the online help and important information about product characteristics.

Multiple access to the same CPU (S7-300/400/1200)

Access to a CPU from a PG/PC is permitted only when a TIA Portal is open. Multiple access to the same CPU is not permitted and can lead to errors.

Loading data blocks during an active control job

Note

Loading changed data blocks during an active control job can result in unforeseen operating states. The control job continues to control the specified address, although the address allocation may have changed in the data block. Complete active control jobs before loading data blocks.

"Enable peripheral outputs" function

The function "Enable peripheral outputs" in not available for CPUs of the S7-1500 series in the TIA Portal V13.

This function can only be executed with an S7-300, S7-400 or S7-1200 CPU in TIA Portal V13.

Testing programs converted from STEP 7 V10.5/V11.0

To monitor and test a program converted from STEP 7 V10.5, you have to first compile and load with STEP 7 V13.0.

Testing with the force table

Contents

Information that could not be included in the online help and important information about product characteristics.

Forcing tags for direct I/O access

If you use direct I/O access for an S7-300 CPU in your user program, forcing this I/O address is not permitted.

Example

If I/O access to the address "IB0:P" takes place in the user program, it is not permitted to force the following I/O address areas: I0.0:P, IB0:P, IW0:P and ID0:P.

2.2.6 Inter Project Engineering (IPE)

2.2.6.1 Notes on IPE

Contents

Information that could not be included in the online help and important information about product characteristics.

Comment box of the device proxy

Contrary to the description in the information system for the TIA Portal V13, with the "Inter Project Engineering" functionality, the comment field for a device proxy cannot be edited.

Restricted functionality

In the TIA-Portal V13, the "S7 GRAPH overview" and "PLC code display" functionality cannot be used in conjunction with the IPE device proxy.

2.2.7 Technological functions

2.2.7.1 Notes on technological functions

S7-1500 technology modules

Contrary to the information provided in the online help, the commissioning editor for the S7-1500 technology modules is not yet available.

S7-1500 Motion Control - Configuration of a positioning axis via an HMI device

If a tag exists both at the speed axis and at the positioning axis (e.g. <TO>.Velocity), you cannot access the tag of the positioning axis via an HMI device.

Copy the tag of the positioning axis via your user program into a data block which you have created. You can access the copy of the tag via an HMI device.

S7-1500 Motion Control - FAQs with remedies for technology alarms

You can find the FAQs with remedies for technology alarms on the Internet (http://support.automation.siemens.com/WW/view/en/66958052).

S7-1500 Motion Control - tuning

After a test step for tuning is started, the axis is slowed down with the configured maximum delay after the measuring duration has elapsed.

S7-1500 Motion Control - Changing the configuration offline

If you change the configuration of the axis offline and download the CPU in RUN operating mode, the modified parameters are displayed, but only take effect after the technology object has been restarted.

To do this, call MC_Reset with Restart = TRUE.

S7-1500 Motion Control and S7-1200 Motion Control

S7-1200 Motion Control and S7-1500 Motion Control are different. The operating modes of the motion control instruction MC_Home are not compatible.

You have to modify your user program. Programs which are transferred to another CPU series always have to be recommissioned.

S_USSI on S7-1500

Note the following when using the instruction S_USSI of the distributed I/O ET 200S 1SI on an S7-1500.

The parameter ANZ shows the value 0 even in the case of an error.

If the CPU is switched to stop with the S_USSI instruction, check all possible errors listed at the parameter ANZ.

2.2.7.2 Notes on technological functions

There are no notes about the technology functions.

2.2.7.3 Notes on technological functions

Parameter assignment for the FM x51 and FM x52

In contrast to the information in the Online Help, if you change the measuring system, the previously entered parameter values are not converted to the new measuring system.

Programming FM x50-1, FM x51 and FM x52

The "initial values" specified in the parameter tables in the online help regarding the instructions correspond to the "default value" in the software interface of the user programming.

IM 174 migration

During the migration of projects with IM 174, the parameter assignment of IM 174 is set to default values. It is absolutely necessary to set parameters for IM 174 again after the migration.

2.2.7.4 SIMATIC STEP 7 Easy Motion Control

Notes on SIMATIC STEP 7 Easy Motion Control

There are no notes available on SIMATIC STEP 7 Easy Motion Control.

2.2.7.5 SIMATIC STEP 7 PID Professional

Notes on SIMATIC STEP 7 PID Professional

There are no notes available on SIMATIC STEP 7 PID Professional.

2.3 WinCC Basic

2.3.1 Security information

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mind, Siemens' products and solutions undergo continuous development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Any third-party products that may be in use must also be taken into account. For more information about industrial security, visit

http://www.siemens.com/industrialsecurity (http://support.automation.siemens.com)

To stay informed about product updates as they occur, sign up for a product-specific newsletter. For more information, visit

http://support.automation.siemens.com (http://support.automation.siemens.com)

Passwords

Various passwords are set by default in WinCC. For security reasons, you should change these passwords.

- For the user "Administrator", the default password is "administrator".

Communication via Ethernet

In Ethernet-based communication, end users themselves are responsible for the security of their data network. The proper functioning of the device cannot be guaranteed in all circumstances; targeted attacks, for example, can lead to overload of the device.

Ending Runtime automatically

If automatic transfer is enabled on the HMI device and a transfer is started on the configuration PC, the running project is automatically stopped on the HMI device. The HMI device then switches autonomously to "Transfer" mode. Transfer mode may cause undesired reactions in the system.

After the commissioning phase, disable the automatic transfer function to prevent the HMI device from switching inadvertently to transfer mode. To block access to the transfer settings and thus avoid unauthorized changes, assign a password in the Control Panel.

Network settings

The following tables show the network settings of each product which you need in order to analyze the network security and for the configuration of external firewalls:

WinCC Basic (without simulation)					
Name	Port number	Transport protocol	Direction	Function	Description
ALM	4410*	TCP	Inbound, Outbound	License service	This service provides the complete functionality for software licenses and is used by both the Automation License Manager as well as all license-related software products.
HMI Load	1033	TCP	Outbound	HMI Load (RT Basic)	This service is used to transmit images and configuration data to Basic Panels.
* Default port that can be changed by user configuration					

WinCC Simulation for Basic Panels					
Name	Port number	Transport protocol	Direction	Function	Description
HMI Load	1033	TCP	Inbound	HMI Load (RT Basic)	This service is used to transmit images and configuration data to Basic Panels.
EtherNet/IP	44818	TCP	Outbound	Ethernet/IP channel	The Ethernet/IP protocol is used for connections to Allen Bradley PLCs.
	2222	UDP	Inbound	Ethernet/IP channel	The Ethernet/IP protocol is used for connections to Allen Bradley PLCs.
Modbus TCP	502	TCP	Outbound	Modbus TCP channel	The Modbus TCP protocol is used for connections to Schneider PLCs.
RFC 1006	102	TCP	Outbound	S7 channel	Communication with the S7 controller via Ethernet/PROFINET
Mitsubishi MC	5002	TCP	Outbound	Mitsubishi MC channel	The Mitsubishi protocol is used for connections to Mitsubishi PLCs.

See also

http://www.siemens.com/industrialsecurity (http://www.industry.siemens.com/topics/global/en/industrial-security/Pages/Default.aspx)

2.3.2 News

News about the TIA Portal

You can learn about the latest news about various topic areas of the TIA Portal in Siemens Industry Online Support.

All news concerning WinCC can be found here: News (http://support.automation.siemens.com/WW/view/en/88360672)

2.3.3 Notes on use

Contents

Information that could no longer be included in the online help and important information about product features.

Copying HMI devices with HMI connections

If you copy an HMI device with HMI connections to a PLC, the HMI connection in the new HMI device is not automatically connected to an existing PLC with the same name. This applies to copying within a project as well as copying across projects.

To access the PLC tag via HMI tag in the new HMI device, you have to complete the configuration of the HMI connection immediately after copying. Proceed as follows:

1. Open the "Devices & Networks" editor.
2. Connect the new HMI device to the desired network.
3. Open the connection table.
4. Select the HMI connection of the new HMI device.
5. Select the desired PLC under "Partner".

If you compile the new HMI device or connect additional PLC tags in between copying the HMI device and completing the connection, there may be some instances in which an additional HMI connection to the same PLC is created. This is especially true if you connect HMI tags with DB array elements.

Device replacement

After an HMI device has been replaced, you should check the appearance of the configured screens. Changing the size of the display may result in changes to the position and appearance of screen objects, e.g. recipe view and alarm view.

Device replacement - communication

If an HMI device is replaced, error messages of the type "... are not supported in the new configuration. and will therefore be removed" may be generated. These alarms refer to configured connections of the device and are triggered, for example, if the HMI devices have a different number of interfaces. These connections are marked red after a device replacement. If you would like to continue to use these connections, you have change the configuration of the connection. Proceed as follows:

1. Open the "Devices and Networks" editor.
2. Click "Network" in the toolbar of the network view.
3. Network the interface of the HMI device with the interface of the CPU.
4. Click in the table area of the network view on the "Connections" table.

5. Select the connection marked red.
6. Enter the new interface under "Properties > General > Interface" in the Inspector window.

Specifying the time of modification in the overview window

The times of modification displayed in the overview window only refer to changes to the object itself. Changes to subordinate objects, e.g. screen objects in a screen, do not cause the time of the last change to the screen to change in the overview window.

HMI device wizard

When you create a device with a color display using the HMI device wizard, the graphics of the navigation buttons may be displayed in black and white. This error only occurs, however, if the new device is created with the same name as a device with a monochrome display which has been deleted in the meantime.

You can avoid this error by always deleting the associated graphics in the Graphics collection whenever you delete a device from the project.

Objects with object references in the project library

Two copying methods can be used in WinCC flexible.

- With "simple copy" a WinCC flexible screen including an IO field, for example is copied. Only the object name of a tag configured on the IO field is copied, as this is a reference.
- With "copy", a screen, an IO field contained there and a tag configured on the IO field together with its properties are copied.

These two methods can also be used for storing an object in a library. Project libraries and the objects contained there are migrated during migration and can be used in WinCC.

In WinCC, however, only one copying method is available. With regard to tags, it functions like "simple copy" in WinCC flexible. With regard to graphics, graphics lists and text lists, it functions like "copy" in WinCC flexible.

If you stored objects with references to tags in a library in WinCC flexible, you must reconfigure the referenced objects when using them in WinCC.

Installing East Asian project languages on a PC without Asian operating system

If you select an East Asian project language on a PC that does not have an Asian operating system installed, the default font is marked as invalid in the "Runtime settings > Language & font" editor.

To resolve this problem, open the "Regional and Language Options > Languages" dialog in the Control Panel and enable the "Install files for East Asian languages" option.

Installation sequence for Startdrive

When you install Startdrive on a PC, adhere to the following installation sequence:

- Install STEP 7 V13.0.
- Install Startdrive.

Compatibility with V12

An empty V12.0.1.3 project with the name "TIA_Portal_Project_V12.0.1.3.ap12" is installed in the installation directory under ..\Portal V13\SampleProjects to allow TIA Portal V13 to be opened in compatibility mode V12. This project must be copied to a local directory with full access before it can be used. For more information on this, refer to FAQ ID 66027369.

File browser on a Windows 8 PC with touch screen

You can only operate the file browser dialog with a mouse, keyboard or on-screen keyboard (without using the touch function) on a Windows 8 PC with touch screen. We recommend using the file browser dialog of the Windows operating system with the help of a script on a touch screen PC with Windows 8.

2.3.4 Migration

Contents

Information that could no longer be included in the online help and important information about product features.

Project languages in WinCC

WinCC V13 does not support all project languages that were available in WinCC flexible, such as Arabic. If you receive an empty project as the result of your migration, you may want to check the set editing language. Do not set the project languages that are not supported as editing language in the source project. Proceed as follows:

1. Open the project with WinCC flexible.
2. Change the editing language to English, for example.
3. Save the project.
4. Restart the migration.

Migrating an integrated project with ProTool objects

The "PROTOOL option package(s) missing in STEP 7" error message output during migration of a WinCC flexible project that is integrated in STEP 7 indicates that WinCC flexible 2008 SP3 is installed on your system. Moreover, the project still contains objects that were configured using ProTool. Do not open the project with WinCC flexible 2008 SP3! Proceed as follows to migrate the project:

1. Copy the project to a computer on which WinCC flexible 2008 SP2 and STEP 7 are installed.
2. Open the project in the SIMATIC Manager.
3. Remove all ProTool objects from the project.
4. Execute the "Save as" command in the "File" menu.
5. Enable the option "With reorganization" in the "Save project as" dialog.
6. Click OK.
7. Copy the project back to the original computer.
8. Restart the migration.

Migrating a WinCC V7 project: Border line of rectangles

In a WinCC V7 project, you have configured a rectangle with the settings "Line weight = 1" and "Draw insider border = yes".

You then migrate the WinCC V7 project to WinCC V13. To have the rectangle displayed correctly, follow these steps.

1. Open the Inspector window of the rectangle.
2. Open the property list.
3. Disable "Widen border line inwards".

Progress bar

As long as the progress bar still shows a value of 100%, the software is still busy running remaining tasks such as the closing of references. The software will not respond to user input while this status is given.

Open a project created with WinCC V11

When you open a V11 project with a WinCC V13 version, it is no longer be possible to open this project with an older version afterwards.

Managing third-party ActiveX controls

The migration also supports third-party ActiveX controls. However, the controls must be registered in the operating system. If an ActiveX control is not registered, migration is canceled.

If you save a project with the migration tool and perform the migration yourself on another PC, the controls must also be registered on this PC.

Migrating integrated projects with alarm views

An alarm view is enabled with all alarm classes in an integrated project. The alarm classes are disabled during migration of the project.
Once the migration of the project is completed, check the settings in the alarm view.
Enable the require alarm classes in the Inspector window of the alarm view if needed under "Properties > General".

Migrating more extensive projects from WinCC V7

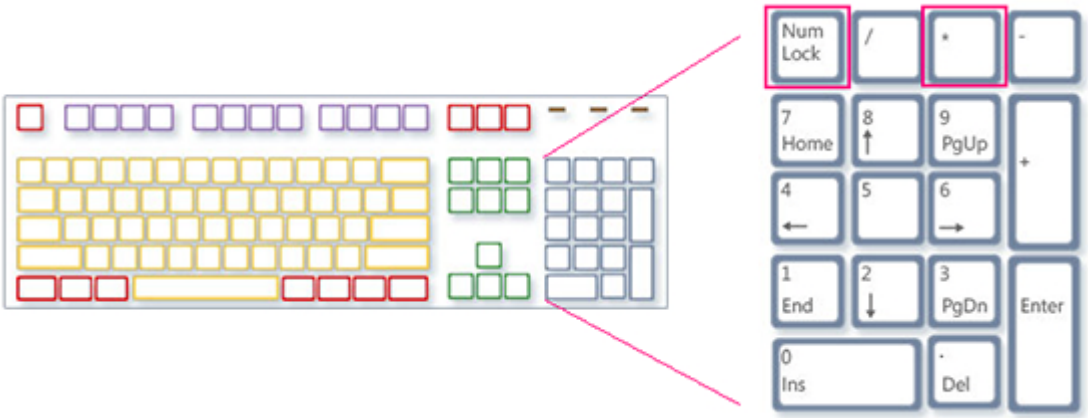
We recommend the use of a 64-bit operating system for the migration of more extensive projects from WinCC V7.

Migrating projects from WinCC V7

In TIA Portal V13, you can continue to use projects from WinCC V7.2 after migration. Projects from earlier WinCC versions cannot be migrated directly to WinCC TIA Portal version V13. If you wish to continue using such projects in TIA Portal V13, you must first migrate them to the WinCC V7.2 Classic page. To do so, use WinCC V7.2 with the latest update.

Migration log

As of TIA Portal V13.0 alarms are sorted into a tree structure in the migration log. This means all alarms that are part of a specific subsystem are stored in one folder. The result is an increase in the number of folders in the tree structure. Press the shortcut <NUM+ X> to expand the currently selected folder and all subfolders in one single step.



Restrictions for user-specific project data

1. Folders and files you have created in the WinCC V7.2 project directory are not copied to the new project directory during the migration. You must adapt all scripts that access such directories and files after the migration.
2. C standard functions of WinCC V7 are not migrated. If you have made changes to the C standard functions in the WinCC V7 project, you must apply these changes manually to the TIA Portal after the migration.

Migration of texts in Spanish (international sorting) and Spanish (traditional sorting)

If the WinCC V7 project includes texts in Spanish (traditional sorting), these texts are migrated as Spanish (Spain) in WinCC V13.

If the WinCC V7 project includes texts in Spanish (international sorting), these texts are migrated as Spanish (Spain) in WinCC V13.

If the WinCC V7 project includes texts in Spanish (international sorting) as well as Spanish (traditional sorting), only the texts from Spanish (traditional sorting) are migrated as Spanish (Spain) in WinCC V13. The texts from Spanish (international sorting) are not taken into consideration.

See also

Object support during migration

2.3.5 Engineering System

2.3.5.1 Screens and Screen Objects

Contents

Information that could no longer be included in the online help and important information about product features.

Display objects after HMI device replacement

If you upgrade a device to the new HMI device versions, you should check the screens contained in the project. Because of the new appearance and improved operability, texts of symbolic I/O fields may not be completely legible and may be concealed by operator controls.

Copying display objects between two projects or two devices

In Project_1 configure an alarm window in the Global Screen, for example. You copy the alarm window and paste it in the Global Screen in Project_2.

The enabled alarm classes are partly not enabled in the alarm window after pasting.

This behavior applies to the following display objects:

- Alarm window
- Alarm indicator
- Alarm view

Display of the cross-references in the Inspector window

The Inspector window displays objects used by a screen object in the "About > Cross-reference" tab.

A screen is open and an object selected. You are using an HMI tag at the object as process tag.

The object and the linked HMI tag are displayed in the cross-references. All locations of use of the object and the HMI tags are listed.

If the HMI tag is interconnected with a PLC tag or a DB tag, the locations of use of the interconnected PLC tag or DB tag are displayed.

Event names in case of alarms in the "Info" tab of the Inspector window

In some alarms of the Inspector window the event names in the "Info" tab will deviate from the names in the "Properties" tab.

Name in the "Properties" tab of the Inspector window	Name in the "Info" tab of the Inspector window
Cleared	ClearScreen
Loaded	GenerateScreen
Enable	Activate
Change	Change
When a dialog is opened	ONMODALBEGIN
When a dialog is closed	ONMODALEND
User change	PASSWORD
Screen change	SCREEN
Disable	Deactivate
Press	Press
Outgoing	Going
Incoming	Coming
Limit "high limit error" violated	AboveUpperLimit
Limit "low limit error" violated	BelowLowerLimit
Click	Click
Loop-in alarm	LoopInAlarm
Release	Release
Alarm buffer overflow	OVERFLOW
Acknowledge	Acknowledgement
Runtime stop	Shutdown
Press key	KeyDown
Release key	KeyUp
Toggle ON	SwitchOn
Toggle OFF	SwitchOff
Value change	Change value

Faceplates

Faceplates cannot be rotated or mirrored.

Tab sequence in screens with faceplates

If you have configured a tab sequence in screens with faceplates in WinCC V12 or WinCC V12 SP1, you should check the tab sequence of these screens in WinCC V13. The tab sequence may have been changed in both the screen and the faceplate.

Tag prefix of a screen window in WinCC Runtime Professional

The objects of the "Controls" palette do not support the tag prefix that can be configured for a screen window.

I/O field with "decimal" display format and format pattern without "s" prefix

You have linked a process tag to an I/O field. The I/O field is assigned the "decimal" display format.

You may select a signed or an unsigned display format.

A "Format pattern" setting without "s", e.g., "999" has the following effects:

1. You cannot set negative values using the I/O field in Runtime.
2. If the tag assumes a negative value, the I/O field generates a two's complement and a corrupted positive value is output.

Trend view on Basic Panels

The trend view buttons are not displayed on Basic Panels. You can operate the trend view using the function keys of the HMI device that are assigned corresponding system functions.

Grouping of screen objects

When you group screen objects in WinCC, performance problems can arise in WinCC in the case of large nesting depths.

ActiveX and .NET controls

ActiveX and .NET controls are always positioned in the foreground in runtime.

The configuration of ActiveX and .NET controls on levels is not supported.

Use of bitmaps as icons

In Windows 8 and Windows 8.1, the use of bitmaps with a size of 48x48 pixels and a color depth of 32 bits as icons is not supported.

Border line of rectangles

In a WinCC V7 project, you have configured a rectangle with the settings "Line weight = 1" and "Draw insider border = yes".

You then migrate the WinCC V7 project to WinCC V12. To have the rectangle displayed correctly, follow these steps.

1. Open the Inspector window of the rectangle.
2. Open the property list.
3. Disable "Widen border line inwards".

Border line of “Graphic I/O field" object

In WinCC V13, the dynamization of the border line of the "Graphic I/O field" object in "Two states" mode has no effect during runtime.

Dynamization of instances of a faceplate type in a group

You are using the instance of a faceplate type in an object group. The properties of the instance are also displayed as properties of the group. Any dynamization with tags, scripts or animations of the group is not displayed in Runtime.

Preview in screen window

You use your own designs with shadows for screen objects. The screen objects can be displayed in a screen window.
The shadows of the screen objects are not displayed in the preview of the screen window. The response occurs only in the engineering system. It is displayed correctly in Runtime.

Assigning graphics to a softkey

A graphic can only be assigned to a softkey if the bottom edge of the permanent window does not conceal the area of the softkey graphic.

Renaming a PLC in Runtime

If a PLC is connected to the PLC code display in WinCC Professional and Runtime is running, changing the name of the PLC during runtime will trigger a error. Do not change the PLC name, the IP address, or other properties of the HMI connection during runtime.

Panels and RT Advanced with device version V13: many visually different screen objects

The use of screen objects with very many visually different properties (e.g., very many different styles) can reduce the performance of the user interface in Runtime and can increase the amount of available memory space used. Avoid using, for example, very many different corner radii: 0 pixel, 1 pixel, 2 pixels, 3 pixels, etc.

The use of many differently sized “Gauge” objects can have the same effect. Avoid “Gauge” of height 48 pixels, 49 pixels, 51 pixels, etc. Instead, use sizes such as 50 pixels, 70 pixels, 100 pixels.

Dynamization of grouped objects

For groupings with multiple nestings (group in group, faceplate in group, group in faceplate, etc.), only the events of the outermost group and the innermost objects can be used for dynamization with system functions. System functions that are configured at events of the lower-level group or lower-level faceplate are not executed.

Number of characters in text fields, lists and alarm texts

The number of available characters in the text of a screen object is dynamic and depends on the HMI device and the memory format. Control instructions and formatting are taken into consideration when entering text data and the maximum number of characters is reduced accordingly.

Transparency in WinCC V13

Transparent screens can be displayed without any problems in Runtime. This is true for all Comfort Panels and WinCC Runtime Advanced with version 13.0.0.0.
To use the transparency in a graphic view or in a graphic I/O field, the "Fill pattern" property must be set to "Transparent" and the property "Use transparent color" must be disabled.

If the property "Use transparent color" is enabled in a device with version 13.0.0.0, the transparency of the screen is lost and the transparent screen areas are displayed in black. We recommend that you check the "Use transparent color" property at the points of use of transparent screens after upgrading existing projects to device version 13.0.0.0. If the screens are not displayed correctly, disable this property.

2.3.5.2 Tags and connections

Contents

Information that could no longer be included in the online help and important information about product features.

Display of deleted array elements at location of use of HMI tags

The locations of use of HMI tags, such as the process value of IO fields, are usually indicated by the tag name. If the element of an array tag is used, then the tag name will be extended by the index of the array element indicated in brackets.

If a used tag is no longer included in the project, then the tag name will still be displayed at the location of use. The field will be displayed with a red background to indicate the missing tag. If a used array element or the array tag is no longer present, then only the index of the array element will be displayed in brackets. The tag name will not be displayed. The field is highlighted in red. You can no longer identify the name of the associated array tag based on the location of use in this instance.

If you do not know which array tag was linked to this location of use, then it may be necessary to link the array element once again.

If a tag or array tag was created based on a reference, then the selected reference will be closed automatically.

If an HMI tag is connected with an array element of a PLC tag and the PLC tag does no longer exist in the project, then the same behavior will take place in the "HMI tags" editor.

Array tags as list entry of multiplex tags

You can use the array tags of the Char data type just like the tags of the String data type.

The use of an array tag of the Char data type as list entry of a multiplex tag in the "HMI tags" editor is not supported.

Multiplexing tags on a Basic Panel

If you multiplex a tag with an external tag on a Basic Panel, the address is read from the PLC in runtime during the first read cycle. The value of the address read is not available until the second read cycle.

Runtime Advanced and Panels: Importing array elements and structure elements

Array tags and structure tags are always imported in full with all elements. The elements of the array tags and structure tags are not filled further during import.

A new tag is created if the name of a tag corresponds to the name of an array or structure element in the import file.

Example:

The import file contains an array tag called "Otto" with 10 array elements. The array elements are then called Otto[1], Otto[2], for example.

If the import file contains a tag named "Otto[1]", the first element of the array tag will not be filled. Instead, a new tag will be created in the engineering system.

Local ID of HMI connections

You cannot edit the "Local ID" value in the HMI connection properties. You need the local ID, for example, for communication by way of AR_SEND. To enable usage of the "Local ID" for communication, proceed as follows:

1. Open the network view in the "Devices & Networks" editor.
2. Click "Connections".
3. Select an S7 connection.
4. Select the "Add new connection" command in the shortcut menu of the PLC.
5. Click on the interface.
6. Specify the "Local ID (hex)".
7. Click "Add" and then "Close".
8. Select "Properties > General" from the partner area of the Inspector window and enter the IP address of the HMI device for the new connection.
9. Configure the necessary raw data tags for communication in the HMI device.

Tags with the DTL data type

Tags that use the "DTL" data type element by element, can only be used as read-only.

Tag names in faceplates

Use of the "." or "@" character in names of tags in faceplate types is not permitted. Do not use these special characters in the tag names in faceplates.

RT Professional: Tags with symbolic addressing and "Char Array" data type

Tags with symbolic addressing and the "Char Array" data type are not released for communication of RT Professional and SIMATIC S7-1200 V3.

Array elements in WinCC

If you have connected an HMI tag with an array from a STEP 7 data block which does not start with a low limit of 0, the array elements are mapped in WinCC to the low limit of 0.

To ensure that you do not have to rethink between the STEP 7 indices and the WinCC indices when accessing the individual array elements, the low limits of arrays should also start at 0 in STEP 7.

2.3.5.3 Alarm system and alarm displays

Contents

Information that could not be included in the online help and important information about product properties.

Displaying special characters in alarm texts

When configuring alarm texts, a fixed character set is used in the Engineering System. This character set allows you to use numerous special characters in alarm texts.

Language-dependent fonts are used in runtime to display the texts, for example MS PGothic, SimSun. The fonts used in runtime do not support all special characters. As a result, some special characters are not displayed in runtime.

Use of multiplex tags in output boxes with alarm texts

It is also possible to use multiplex tags in the output boxes of alarm texts in the engineering system. During runtime, this leads to an incorrect display of the alarm, because the use of multiplex tags is not supported by the basic panels.

Parameters in user alarms

Contrary to the information in the online help, it is not possible to configure parameters for user alarms.

The menu command "Properties > Properties > Alarm parameters" is not available in the Inspector window.

Tags in alarm texts of RT Advanced

Tags of data type WChar or WString cannot be displayed in the alarm view in RT Advanced.

2.3.5.4 System functions

Content

Information that could not be included in the online help and important information about product properties.

The "SimulateTag" system function on Basic Panels

If you use the system function "SimulateTag" with a short cycle time on a Basic Panel, the HMI device may be overloaded.

2.3.5.5 Recipes

Contents

Information that could not be included in the online help and important information about product features.

Arrays in recipe elements

If you have configured both an array as well as the elements of this array for recipe elements of a recipe, the loading of data records aborts with the following error message: "290055: Import of data records aborted with error"

Use either the array or just the array elements for recipe elements of a single recipe.

2.3.5.6 User administration

Contents

Information that could not be included in the online help and important information about product features.

2.3.5.7 Communication

Contents

Information that could no longer be included in the online help and important information about product features.

Connection interruptions with Mitsubishi PLCs

After multiple connection interruptions, a situation may arise where all the connection resources of the Mitsubishi PLC are in use and the connection can no longer be established. It is recommended to check these connection resources in the PLC program of the PLC and also to enable them again.

Limited number of possible HMI connections

An error message is displayed during compilation of a device indicating that the configuration of the HMI connection in the "Devices & Networks" editor is invalid. The reason may be that the maximum number of possible connections of the HMI device or PLC has been exceeded.

Check the maximum number of available connections. Consult the device manuals of the devices you are using.

Communication by means of routing with S7 300/400

Communication of connection partners in different subnets is possible via routing with the following connections: PROFINET, PROFIBUS, MPI.

Use of PROFINET IO with panel HMI devices

When using PROFINET IO to connect the direct keys and LEDs of HMI devices to the PLC, you can define an offset for the address area of the inputs and outputs during configuration in HW Config.

The following restriction applies when a PROFINET IO-capable S7-400 CPU is used with one of the HMI devices listed below:

The offset for the start of the address area of the inputs must not be greater than the offset for the start of the address area of the outputs.

The restriction applies to the following HMI devices:

- OP 177B
- OP 277
- Mobile Panel 177

For the configuration of the address parameters, open the PLC with the S7-400 CPU in HW Config. Select the HMI device connected via PROFINET IO in the station window of HW Config. A table with the properties of the HMI device is displayed at the bottom of the station window in the detail view. Select the line containing the addresses of the HMI device in the table and open the object properties using the shortcut menu.

Enable the "Addresses" tab in the "Object properties" dialog. Configure the offset for the inputs under "Inputs > Start". Configure the offset for the outputs under "Outputs > Start".

Exceeding value ranges with Mitsubishi MC and Mitsubishi FX

With some data types, the Mitsubishi MC and Mitsubishi FX communication drivers do not check whether the value of a recipe tag exceeds the value range of the PLC tags. The data types affected are:

- 4-bit block
- 12-bit block
- 20-bit block
- 24-bit block
- 28-bit block

Coordination area pointer in an OPC connection

In principle, the coordination area pointer can be used eight times in an OPC connection. If you have configured an OPC connection and you automatically create another OPC connection using "Add", the coordination area pointer is only displayed once in the newly created connection. In this case, you should change the communication driver of the connection. If you then set OPC again as the communication driver, the coordination area pointer can again be used eight times.

Changing IP settings and device name of a PLC in the Control Panel of the HMI device

The Control Panel is open in the "Service and Commissoning > IP-Adaptation" menu on the HMI device. If you want to change the IP settings or the device name of a PLC, note the following:

In the Engineering System, you need to have activated the following options in the Inspector window of the PLC under "Properties > General > PROFINET interface > Ethernet addresses" beforehand:
- "Set IP address using a different method" and
- "Set PROFINET device name using a different method".

HMI connections in WinCC V13

HMI connections to SIMATIC S7-1200 PLCs with firmware versions earlier than V2.0 are not possible in WinCC V13.

Connections via PROFIBUS DP

When a connection between a PLC and an HMI device via PROFIBUS DP is interrupted and then re-established, sporadically all other PROFIBUS DP connections in the communication network are interrupted and re-established.

De-energize the disconnected station before reconnecting it.

"Set the IP suite (address) of the PLC in the Control Panel" with SIMATIC S7-1200 V1

The function "Set the IP suite (address) of the PLC in the Control Panel" has not been approved for the following PLCs:

- SIMATIC S7-1200 V1

Switching a connection

A connection may be interrupted when it is switched from an HMI device to a SIMATIC S7-300/400, to a SIMATIC S7-1500 or to a SIMATIC S7-1200 PLC.

Note the following settings in the SIMATIC S7-1500 or SIMATIC S7-1200 controllers:

- Absolute addressing of tags
- The "Disable PUT-GET communication" option must be selected
- The "Complete protection" protection level may not be set

Raw data communication in redundant projects

Simatic.NET, Named Connections and various communication blocks, such as BSEND/BRCV, for example, can only be used to a limited extent in a redundantly configured PC station because the connection parameters for the redundant partner server cannot be configured.

2.3.6 System-wide functions

Contents

Information that could no longer be included in the online help and important information about product features.

Inter Project Engineering: Displaying alarms of system diagnostics

If you do not enable "PLC alarms" when you create device proxy data, system diagnostic alarms are not applied to the device proxy data.

System diagnostic alarms of the device proxy are not displayed after initialization of device proxy data without "PLC alarms".

2.3.7 Compiling and loading

Contents

Information that could no longer be included in the online help and important information about product features.

Compiling and loading

If internal errors or warnings occur during compiling, compile the complete project using the command "Compile > Software (rebuild all)" in the shortcut menu of the HMI device.

Before you start productive operation with your project, compile the entire project using the "Compile > Software (rebuild all)" command from the shortcut menu of the HMI device.

If you are using HMI tags that are connected to the control tags in your project, compile all modified blocks with the command "Compile > Software" in the shortcut menu before you compile the HMI device.

Settings for update of operating system

If you select the command "Online > HMI device maintenance > Update operating system" from WinCC, you cannot change the settings such as the type of PG/PC interface or baud rate. The settings used during the last download are always used.

To make changes to the settings, open the "Extended download" dialog using the "Online > Extended download to device" command and change the settings. When you click the "Load" button the changed settings are saved.

Alternatively, you can perform an update of the operating system with changed settings with ProSave. You start ProSave via the Windows Start menu "Siemens Automation > Options and Tools > HMI Tools > SIMATIC ProSave".

Incorrect installation of ProSave

If you receive an error message during installation of ProSave when loading data to a target device or maintenance of an HMI device, then you cannot remedy this error using the repair function of setup. Remove ProSave via the Control Panel. Then start setup and install the "ProSave" component again.

Checking the address parameters

During compilation of an HMI device in the project tree with the command "Compile > Software" in the shortcut menu, the address parameters of the HMI device, such as the IP address, will not be checked. If you want to ensure that the address parameters are checked as well, you will have to compile the HMI device using the "Compile" button in the "Devices & Networks" editor of the toolbar.

Error message when downloading data to the PLC

A panel and a PLC are connected and communicating with other.

If a tag is accessed while downloading data from the panel to the PLC, an error message is displayed on the panel.

Delayed reaction in the "Extended download to device" dialog

If the settings in the "Extended download to device" dialog for "Type of the PG/PC interface" and "PG/PC interface" do not match the settings on the HMI device, this can result in the application not responding for up to a minute.

Extended download with an S7-1200 and a Comfort Panel

An S7-1200 PLC and a Comfort Panel are located in the same physical network as the PG/PC. You open the "Extended download to device" dialog for the Comfort Panel.

If you enable the option "Show all accessible devices", it may occur that the application stops responding.

OP77A, OP73, TP177A: Loading projects

When loading a project to an HMI device, it can happen that Runtime is not automatically ended, even though "Remote Transfers" is activated in the Panel.

If this happens, stop Runtime and manually set the transfer mode on the HMI device.

Loading a SIMATIC HMI application to a PC station

The following circumstances can lead to an error message during the first load of a SIMATIC PC station:

- A SIMATIC HMI application is configured in a PC station in the project
  - WinCC Runtime Advanced
  - WinCC Runtime Professional
  - WinCC Standby
  - or WinCC Client
- The property "S7RTM is installed" is activated.

Before you load a SIMATIC PC station for the first time, select the configured device HMI_RT (WinCC...) in the project tree. Open the "Extended download to device" dialog and select the appropriate interface and parameter settings. Click "Load".

You then load the PC station as normal.

Project transfer via USB

If you have connected more than one HMI device via USB to your configuration PC, project transfer is only possible to the last connected device.

Opening project files

When you run "HmiIRTm.exe", a dialog opens asking if you want to open the project file (.fwc).

The following options are available to you:

- "Yes": A dialog opens allowing you to select a project file (.fwc).
- "No": The dialog closes.

Tag simulator in WinCC Professional does not start

If the tag simulator does not start in WinCC Professional, it may be because the fm20.dll file has not been installed in the system yet. The fm20.dll file cannot be distributed and must be installed as part of a Microsoft application:
http://support.microsoft.com (http://support.microsoft.com/default.aspx?scid=kb;EN-US;224305)

You have the following two options for installing the fm20.dll file:

- Install an application, such as Microsoft Office 97, on the destination system. The installation routine also installs the fm20.dll file.
- You can also download and install the Microsoft ActiveX Control Pad free of charge. The fm20.dll file is also installed in this case. Additional information on the ActiveX Control Pad is available on the Microsoft Developer Network website:
http://msdn.microsoft.com/en-us/library/ms968493.aspx (http://support.microsoft.com/default.aspx?scid=kb;EN-US;224305)

2.3.8 Runtime

2.3.8.1 Notes on operation in Runtime

Contents

Information that could no longer be included in the online help and important information about product features.

Focus in Runtime

If you have configured a low-contrast combination of focus color and border color in a V12 project, the focus may no longer be identifiable after changing the HMI device version in Runtime. Change one of the two colors.

Language behavior - Layout of on-screen keyboard

The layout of the on-screen keyboard is not switched when the runtime language changes.

Tag values exceed the maximum length

You enter a character string in a string tag via an I/O field. If the character string exceeds the configured number of tags, the character string will be shortened to the configured length.

Empty message texts

Runtime is running with a project. The project is saved on a network drive.

In the event of interruptions to the network drive connection, Runtime may attempt to load message texts from the network drive.

In the event of disconnection, the alarm window or the alarm view remains empty.

To avoid this, copy the project to a local drive before the starting the project in Runtime.

Complete download in Service mode

If you need to perform a "complete download" to the OS in Service mode from the engineering station, Runtime automatically stops and then starts again.

The project is then no longer in Service mode.

In this state, the power supply is interrupted and WinCC Runtime no longer starts automatically on the OS.

Remedy:

1. Switch the project manually to Service mode after you have performed the "complete download".
2. Close the project manually.
3. Enable Service mode.
4. Start Runtime again using the surrogate icon in the taskbar.

Slow response from the screen keyboard and SmartServer

The following programs may start and respond very slowly under Windows 7 and Windows 2008 servers:

- Microsoft OSK screen keyboard and HMI TouchInputPC
- SmartServer: <Ctrl+Alt+Del> shortcut in the logon dialog

The delay is caused by the callback for the Internet certificate validation.

Remedy:

You can find the following files
on the product DVD under:
Support\Windows7\CRL_Check or CD_RT\ Support\Windows7\CRL_Check\:

- DisableCRLCheck_LocalSystem.cmd
- DisableCRLCheck_CurrentUser.cmd

1. Run the "DisableCRLCheck_LocalSystem.cmd" file with administrator rights. Select the command "Run as administrator" from the shortcut menu of the file.
2. Reboot the PC.

If the problem persists, follow these steps:

1. Double-click the file and run the "DisableCRLCheck_CurrentUser.cmd" file with user rights.
2. Reboot the PC.

Note

The callback for the certificate validation is disabled for all users or PCs. To restore the original state, perform the following files:

- RestoreDefaults_LocalSystem.cmd
- RestoreDefaults_CurrentUser.cmd

You can find the files in the following directory of the product DVD:

- Support\Windows7\CRL_Check or CD_RT\Support\Windows7\CRL_Check\

Starting Runtime

Only WinCC Runtime V13 can be started in TIA Portal V13. WinCC Runtime V11.02, V12 or V13 can be simulated in TIA Portal V13.

2.3.8.2 Notes on operation of panels in Runtime

Contents

Information that could not be included in the online help and important information about product features.

License transfer via S7USB

You always need to run WinCC to transfer a license to a panel via S7USB.

Transferring licenses to a panel on 64-bit operating systems

If you are running a 64-bit operating system and the "Edit > Connect target systems > Connect HMI device" menu command is not available in Automation License Manager, open command line input and run the following command with administrator rights:

"%WINDIR%\system32\RegSvr32.exe" "%CommonProgramFiles%\siemens\AlmPanelPlugin\ALMPanelParam.dll"

Using the mouse wheel in Runtime

The use of the mouse wheel in Runtime is not supported on all panels.

Basic Panels, OP73, OP77A and TP177A: Displaying texts in runtime

The default font selected in the "Runtime settings > Languages & font" editor has an effect on the display of texts in runtime.

Text entries may be truncated if you selected an unfavorable font size or style.

This setting possibly has an effect on the following text entries:

- Tooltips
- long alarm text
- text in the dialogs

Basic Panels: Connections to S7-1200 and S7-1500 with Backup/Restore

If you use the "Backup/Restore" function, a maximum of two connections from Basic Panels to the controllers are possible at any given time.

- SIMATIC S7-1200
- SIMATIC S7-1500

Basic Panels: Backup on the memory card of the PLC

Create the backup file "A.psb" on the memory card of the PLC. An error, for example a connection break, occurs when creating the backup.
This will create a corrupt file on the memory card of the PLC. Such a file has "~$" as prefix. Delete the file with the prefix "~$" if you want to save a backup again under the same name "A.psb".

Basic Panels: Panel Data Storage and S7-1500F

The "Panel Data Storage" PDS function cannot be used on Basic Panels in conjunction with S7-1500F when the password for the protection level "Full access incl. fail-safe" is used.

2.3.9 HMI devices

2.3.9.1 Notes on HMI devices

Contents

Information that could no longer be included in the online help and important information about product features.

If the PC goes into standby or hibernate mode while the transfer is in progress, the panel status after interruption of the transfer is not defined.

Multi-key operation

Unintentional actions can be triggered by multi-key operation:

- When you are using a key device, you cannot press more than two function keys at the same time.
- When you are using a touch device, a standard PC or a panel PC, you can only press one function key or button at the same time.

TS Adapter with Ethernet interface

If an HMI device is connected via Ethernet and a TS adapter, it can not be reset to factory settings.

Simulation of the Basic Panels

Use an output field in an alarm text to output an external tag. The content of the output field will always be displayed with "0" during simulation.

Simulation with real PLC connection

The access point used by the simulation is independent from the settings of the engineering system and can only be altered in the Control Panel with the "Setting PG/PC Interface" tool. If the PLC connection is terminated right after the start of the simulation with alarm 140001, you should check the access point used by the simulation with "Setting PG/PC Interface".

1. Double-click "Setting PG/PC Interface" in the Control Panel. A dialog opens.
2. Select" "S7ONLINE" in the "Access point of application" field as standard for HMI.
3. Select the interface in the "Interface Parameter Assignment Used" area.
4. Exit the dialog "Setting PG/PC Interface" with OK

Loading of projects without recipe data records

You are using recipes in a project. You transfer the project to a Basic Panel without recipe data records.

You may encounter inconsistencies if you have altered the structure of the recipe in the engineering system and the device already held recipe data records.

Check the consistency of the data records in this case. The device will not issue a note for all structural changes.

Floating point numbers on MP 277, MP 377, TP 177B 4" and CP4

Only floating point numbers in the range from 10^{-293} ... 10^{+307} are displayed correctly on the HMI devices MP 277, MP 377, TP 177B 4" and CP4. If the tag value is outside this range, it is displayed as 0.

Mobile Panels V2

If you use Mobile Panels V2 in a project, it is not possible to open the project with WinCC V11 SP1. This affects projects with the following devices:

- Mobile Panel 277F IWLAN (RFID Tag)
- Mobile Panel 277F IWLAN V2
- Mobile Panel 277 IWLAN V2

"Zone ID/Connection point ID" tag of a Mobile Panel 277 IWLAN V2

The tag used for the "Zone ID/Connection point ID" must be of data type INT for Mobile Panel 277 IWLAN V2 devices. Adapt this data type if necessary when migrating a project.

HMI devices with operating system Windows CE 5.0 or higher

Owing to a modified client-server communication security setting, the time difference between the HMI device (client) and PC (server) must not exceed 1 day. If you back up recipe data from the HMI device on a network drive, for example, make sure that the time is set correctly on the PC (server) and the HMI device (client).

HMI devices with high communication load

S7 Diagnostics should be enabled if a Panel is assigned many connections to PLCs or other HMI devices. Otherwise, you will risk overload on the Panel.

Device replacement in the engineering system

In the engineering system, you replace a device with configured LED keys with a device without LED keys. Runtime start fails after you have transferred the project data to the device.

For this reason, delete the LED key configuration before you replace the device.

Restrictions for the HMI device, MP 377 15" Touch daylight readable

The following functions are not supported in WinCC V12 for the MP 377 15" Touch daylight readable HMI device:

- Option: Sm@rtServer
- System function: SetAndGetBrightness
- Direct keys

Upgrading Basic Panels to WinCC V13

Before you upgrade Basic Panels from version V12 to version V13, transfer the image of the V12 SP2 Update 5 or higher to the devices.
In the "SIMATIC ProSave [OS Update]" dialog, select the setting "Reset to factory settings".

In this way, you always start a functional update of the image.

Affected devices:

- KP300 Basic mono PN
- KP400 Basic color PN
- KTP400 Basic color PN

Connection switch in the Control Panel with Basic Panels

If you use the "Override protected connection information" function, the following restriction applies:

You cannot perform a connection switch in the Control Panel of a Basic Panel from a PLC without a protection level to a PLC with a "Complete protection" level.

Mobile Panels 277F 8" IWLAN

Mobile Panels 277F 8" IWLAN V2 and 277F 8" IWLAN (RFID Tag) cannot be configured with WinCC V13.

Basic Panels 2nd Generation

Basic Panel KTP1200 Basic can currently not be configured with WinCC V13.

The following maximum values are in effect for Basic Panels 2nd Generation.

Alarms

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of discrete alarms	1000	1000	1000	1000

Screens

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of complex objects per screen 1)	This information is not relevant for Basic Panels 2nd Generation.			
Number of system diagnostic displays per screen	5	5	5	5
Number of array elements per screen 2)	100	100	100	100

1) Complex objects include: Bars, sliders, symbol library, clock, and all objects from the Controls area.

2) Array elements contained in recipes are included in the count.

Logs

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of entries per log (including all log segments) [1)]	10000	10000	10000	10000
Number of log segments	400	400	400	400
Number of tags that can be logged per log	10	10	10	10
Cyclic trigger for tag logging	1 s	1 s	1 s	1 s

Installation 3

3.1 Notes on the installation

Contents

Information that could not be included in the online help and important information about product characteristics.

Automated Installation

A description of automated installation is available on the product DVD in the directory "Documents\Readme\<Language directory>".

Using the same versions of TIA Portal products

The TIA Portal products STEP 7 (incl. PLCSIM), WinCC and Startdrive must be upgraded to the same version (V13). If the versions are different, you will no longer be able to start the TIA Portal. Even uninstalling a product will not restore an operational status. Please ensure that you have the service packs for these products to hand before you start the installation. You can download the service packs from the Internet under Siemens Industry online support (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=99&lang=en&referer=%2fWW%2f&func=cslib.csinfo2&siteid=csius&extranet=standard&viewreg=WW&groupid=4000002).

Target directory of the installation

Do not use any UNICODE characters (for example, Chinese characters) in the installation path.

Use of antivirus programs

During the installation, read and write access to already installed files is necessary. Some antivirus programs block this access. We therefore recommend that you disable antivirus programs during the installation of the TIA Portal and enable them again afterwards.

3.2 System requirements for installation

3.2.1 Notes on licenses

Availability of licenses

The licenses for the products of the TIA Portal are usually supplied on the installation data medium and installed automatically by the Automation Licence Manager during the installation process of the TIA Portal.

Before you uninstall the TIA Portal, you must transfer and back up the licenses still required. Use the Automation License Manager for this purpose.

Provision of the Automation License Manager

The Automation License Manager is supplied on the installation data medium and is transferred automatically during the installation process.

If you remove the TIA Portal, the Automation License Manager remains installed on your system.

Working with the Automation License Manager

The Automation License Manager is a product of Siemens AG, which is used for handling license keys (technical representatives of licenses).

Software products that require license keys for operation, such as the TIA-Portal, register the required license key automatically with the Automation License Manager . If the Automation License Manager finds a valid license key for this software, the software can be used according to the license usage terms associated with this license key.

Note

For additional information on how to manage your licenses with the Automation License Manager , refer to the documentation supplied with the Automation License Manager .

See also

Notes on the system requirements (Page 85)

Starting installation (Page 104)

Displaying the installed software (Page 108)

Modifying or updating installed products (Page 109)

Repairing installed products (Page 110)

Starting to uninstall (Page 112)

Installation log (Page 103)

3.2.2 Notes on the system requirements

System requirements for individual products

The system requirements may differ depending on the products you want to install. You should therefore check the individual system requirements of your products.

If you want to install several products, make sure that your system meets the demands of the product with the highest requirements.

Displaying PDF files

To be able to read the supplied PDF files, you require a PDF reader that is compatible with PDF 1.7 e.g. Adobe (R) Reader version 9.

Displaying the Welcome Tour

You require the Adobe (R) Reader as of version 9 to start the Welcome Tour for the TIA portal.

See also

Notes on licenses (Page 84)

Starting installation (Page 104)

Displaying the installed software (Page 108)

Modifying or updating installed products (Page 109)

Repairing installed products (Page 110)

Starting to uninstall (Page 112)

3.2.3 System requirements STEP 7 Professional

3.2.3.1 Licensing of STEP 7

Introduction

You require a License Key to license the following STEP 7 editions:

- STEP 7 Basic
- STEP 7 Professional

You can install the corresponding License Key when you are installing STEP 7 or transfer it using the Automation License Manager after the installation has been completed.

Licenses for STEP 7

The following licenses with the corresponding license keys are available:

- STEP 7 Basic
- STEP 7 Professional
- STEP 7 Professional Combo

Validity of license keys for older versions of STEP 7

With a valid License Key for V13.x of STEP 7 Professional and STEP 7 Professional Combo, you can also operate older versions of STEP 7 without restrictions. The following table provides more detailed information about this:

Edition	License	Valid for
STEP 7 Basic V13.x	STEP 7 Basic	• STEP 7 Basic V13.x • STEP 7 Basic V12.x • STEP 7 Basic V11.x • STEP 7 Basic V10.5
STEP 7 Professional V13.x	STEP 7 Professional	• STEP 7 Professional V13.x • STEP 7 Professional V12.x • STEP 7 Professional V11.x • STEP 7 Basic V13.x • STEP 7 Basic V12.x • STEP 7 Basic V11.x • STEP 7 Basic V10.5
STEP 7 Professional V13.x	STEP 7 Professional Combo	• STEP 7 Professional V13.x • STEP 7 Professional V12.x • STEP 7 Professional V11.x • STEP 7 Basic V13.x • STEP 7 Basic V12.x • STEP 7 Basic V11.x • STEP 7 Basic V10.5 • STEP 7 V5.5 • STEP 7 V5.4 • STEP 7 Professional 2010 • STEP 7 Professional 2006

Starting without a valid license key

If you start an edition of STEP 7 without a valid License Key , the system alerts you that you are working in non-licensed mode. You have the one-time option of activating a Trial License. However, this license is valid for a limited period only and expires after 21 days.

When the Trial License expires, the following scenarios can occur:

- STEP 7 has never been licensed on the PC in question:
  - Operations requiring a license can no longer be performed in STEP 7.
- STEP 7 was already licensed on the PC in question:
  - A window requiring acknowledgment presents an alert for non-licensed mode every 10 minutes and for every action requiring a license.

License requirements for simulation

You do not require additional licenses when you use the menu command "Online > Simulation" to start the simulation in STEP 7.

If the following conditions are met, the appropriate licenses for the edition of STEP 7 you have installed are also required for the simulation.

- The engineering station is connected to a PLC.
- The connection to the PLC is configured and active.

See also

Handling licenses and license keys (Page 88)

3.2.3.2 Handling licenses and license keys

Introduction

In each case, you require a valid License Key to use STEP 7 Basic and STEP 7 Professional.

Installing license keys

When you install STEP 7 Basic, the License Key is installed automatically during setup. When you install STEP 7 Professional you will be prompted at the end of the setup to transfer the license from the supplied storage medium to your PC.

If you want to install additional License Keys, you have to use the Automation License Manager to do this.

When you install a license, the associated license key is removed from the license key storage location.

NOTICE
Destruction of license keys by copying It is not possible to copy a License Key. The copy protection prevents the license keys from being copied. If you attempt to copy a License Key this will be destroyed.

Uninstalling license keys

License keys are always uninstalled using the Automation License Manager. You uninstall a License Key in the following cases:

- When backing up data.
- If you no longer require the license.

You can then use a valid license on another PC or HMI device.

Data backup

When backing up data on the HMI device or creating a backup during device replacement, remove the License Keys on the HMI device. To do this, open the Automation License Manager and back up the uninstalled license key to another storage location.

NOTICE
Destruction of license keys on PCs Start by removing all license keys in the following situations: • Before you format the hard disk. • Before you compress the hard disk. • Before you restore the hard disk. • Before you start an optimization program that moves fixed blocks. • Before you install a new operating system. Read the description of Automation License Manager ("Start > Siemens Automation > Documentation"). Observe all warnings and notices.

On PC-based HMI devices and on non-PC-based HMI devices where Automation License Manager is used, the license key storage location may contain multiple license keys. This capability means you can store multiple licenses of the same type at one location. Save all license keys of an HMI device to the same storage location.

NOTICE
Always keep the original storage location of the license keys.

Invalid license after time zone change

The installed license no longer functions in the following case.

- If you change the time zone on a PC as follows:
  From a time based on a complete hour to a time not based on a complete hour.
  Example: You change the time zone from GMT +3:00 to GMT +3:30.

To avoid this inconvenience, uninstall the license key using the Automation License Manager under the time zone setting that was set when the license key was installed.

This behavior does not apply to the Trial License.

Defective license

A license is defective in the following cases:

- If the license key is no longer accessible at the storage location.
- If the license key disappears during its transfer to the destination drive.

You can use the Automation License Manager to repair the defective license. To do this, use the "Restore" function or the "Restore wizard" of the Automation License Manager. To perform this restore operation you must contact Customer Support.

You can find more detailed information on the Internet: http://support.automation.siemens.com (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=99&lang=en&referer=%2fWW%2f&func=cslib.csinfo2&siteid=csius&extranet=standard&viewreg=WW&groupid=4000002)

See also

Licensing of STEP 7 (Page 86)

3.2.3.3 STEP 7 Professional for PC-CP

Functions of STEP 7 Professional for PC-CP

STEP 7 Professional for PC-CP is a full version of STEP 7 Professional that was installed without license. This version only supports configuration and diagnostic options for the following components:

- PC-CPs and applications (OPC server) for PC stations
- Network components and routers: SCALANCE X, SCALANCE W, SCALANCE M
- Security functions: SCALANCE S, Security configuration (SCT)
- PROFINET IO devices and PROFIBUS DP slaves
- RF and MV Reader

SIMATIC stations and WinAC components are only available in the network and topology view (no device view). Programming functions are not available.

Documentation

The information system for PC-CP available in STEP 7 Professional describes the full scope of STEP 7 Professional functions. You should therefore remember that the functions are restricted to the configuration of the previously listed device groups.

Upgrading to the range of functions of STEP 7 Professional

The full scope of STEP 7 Professional functions is already available on your engineering station with the installation of STEP 7 Professional for PC-CP, although its use is restricted.

By installing a license later, you can upgrade at any time to the full functionality of STEP 7 Professional. There is no need for a new installation. Any projects created previously in STEP 7 Professional for PC-CP can still be used without restrictions once a license is installed.

3.2.3.4 Software and hardware requirements STEP 7

Minimum software and hardware requirements

The following table shows the minimum software and hardware requirements that have to be met for installation of the "SIMATIC STEP 7 Professional" software package:

Hardware/software	Requirement
Processor	Intel® Celeron® Dual Core 2.2 GHz (Ivy/Sandy Bridge)
RAM	4 GB
Available hard disk space	5 GB
Operating systems *	**Windows 7 (32-bit)** • Windows 7 Professional SP1 • Windows 7 Enterprise SP1 • Windows 7 Ultimate SP1 **Windows 7 (64-bit)** • Windows 7 Professional SP1 • Windows 7 Enterprise SP1 • Windows 7 Ultimate SP1 **Windows 8.1 (64-bit)** • Windows 8.1 Professional • Windows 8.1 Enterprise **Windows Server (64-bit)** • Windows Server 2008 R2 StdE SP1 (full installation) • Windows Server 2012 R2 StdE (full installation)
Screen resolution	1024 x 768

* For more detailed information on operating systems, refer to the help on Microsoft Windows or the Microsoft homepage.

Recommended software and hardware requirements

The following table shows the recommended software and hardware for the operation of STEP 7.

Hardware/software	Requirement
Computer	From SIMATIC FIELD PG M4 PREMIUM (or similar PC)
Processor	From Intel® Core™ i5-3320M 3.3 GHz
RAM	8 GB or more
Hard disk	300 GB SSD

Hardware/software	Requirement
Monitor	15.6" Wide Screen Display (1920 x 1080)
Operating systems *	**Windows 7 (64-bit)** • Windows 7 Professional SP1 • Windows 7 Enterprise SP1 • Windows 7 Ultimate SP1 **Windows 8.1 (64-bit)** • Windows 8.1 • Windows 8.1 Professional • Windows 8.1 Enterprise **Windows Server (64-bit)** • Windows Server 2008 R2 StdE SP1 (full installation) • Windows Server 2012 R2 StdE (full installation)

* For more detailed information on operating systems, refer to the help on Microsoft Windows or the Microsoft homepage

Supported virtualization platforms

You can install the "SIMATIC STEP 7 Professional" software package on a virtual machine. For this purpose, use one of the following virtualization platforms in the specified version or a newer version:

- VMware vSphere Hypervisor (ESXi) 5.5
- VMware Workstation 10
- VMware Player 6.0
- Microsoft Windows Server 2012 R2 Hyper-V

These virtualization platforms can use the following operating systems as host operating system:

- Windows 7 Professional/Ultimate/Enterprise (64-bit)
- Windows Server 2008 R2 (64-bit)
- Windows Server 2012 R2 (64-bit)
- Windows 8.1 Professional/Enterprise (64-bit)

You can use the following host operating systems to install "SIMATIC STEP 7 Professional" within the selected virtualization platform:

- Windows 7 Professional/Ultimate/Enterprise (64-bit)
- Windows 8.1 Professional/Enterprise (64-bit)

Note

- The same hardware requirements apply to the host operating system as for the respective TIA products.
- The plant operator must ensure that sufficient system resources are available for the host operating systems.
- The hardware certified by the manufacturers is recommended for the use of HyperV server and ESXi.
- When you use Microsoft Hyper-V, accessible stations cannot be displayed.

Supported security programs

The following security programs are compatible with "SIMATIC STEP 7 Professional":

- Antivirus programs:
  - Symantec Endpoint Protection 12.1
  - Trend Micro Office Scan Corporate Edition 10.6
  - McAfee VirusScan Enterprise 8.8
  - Kaspersky Anti-Virus 2014
  - Windows Defender (as of Windows 8.1)
- Encryption software:
  - Microsoft Bitlocker
- Host-based Intrusion Detection System:
  - McAfee Application Control 6.0

3.2.4 System requirement for WinCC Basic

3.2.4.1 Software and hardware requirements

Introduction

Specific requirements for the operating system and software configuration must be met for the installation.

Note

WinCC is generally authorized for use in a domain or workgroup.

However, be aware that domain group policies and restrictions of the domain may hinder the installation. If this happens, remove the computer from the domain prior to installing Microsoft Message Queuing, Microsoft SQL Server 2005 and WinCC. Log onto the computer in question locally with administrative rights. Then perform the installation. After successful installation, you can enter the WinCC computer back into the domain. If the domain group policies and restrictions of the domain do not impede the installation, the computer need not be removed from the domain during the installation.

Be aware that domain group policies and restrictions of the domain may also hinder operation. If you cannot avoid these restrictions, run the WinCC computer in a workgroup.

Consult with the domain administrator if needed.

Installation requirements

The following table shows the minimum software and hardware requirements that have to be met for the installation of the "SIMATIC WinCC Basic" software package:

Hardware/software	Requirement
Processor type	Intel® Celeron® Dual Core 2.2 GHz (Ivy/Sandy Bridge)
RAM	4 GB
Free hard disk space	5 GB

Hardware/software	Requirement
Operating systems *	**Windows 7 (32 bit)** • Windows 7 Home Premium SP1 • Windows 7 Professional SP1 • Windows 7 Enterprise SP1 • Windows 7 Ultimate SP1 **Windows 7 (64 bit)** • Windows 7 Home Premium SP1 • Windows 7 Professional SP1 • Windows 7 Enterprise SP1 • Windows 7 Ultimate SP1 **Windows 8 (64 bit)** • Windows 8.1 • Windows 8.1 Professional • Windows 8.1 Enterprise **Windows Server (64 bit)** • Windows Server 2012 R2 Standard Edition
Screen resolution	1024x768
Network	Ethernet 10 Mbps or faster
Optical drive	DVD-ROM
Software	Microsoft .Net Framework 4.5

* For additional information on the operating systems, refer to the Microsoft Windows help or the Microsoft homepage.

Simultaneously opening multiple instances of WinCC on a configuration PC can also increase the hardware capacity required.

Note

"Aero Glass Style" of Microsoft Windows 7

A powerful graphics card is required for "Aero Glass Style". It requires DirectX9 capabilities and 128 MB of dedicated graphics memory.

The performance of the architecture of the graphics system can significantly influence the performance of WinCC.

Recommended hardware

The following table shows the recommended hardware for the operation of SIMATIC WinCC.

Hardware	Requirement
Computer	SIMATIC FIELD PG M4 PREMIUM or higher (or comparable PC)
Processor	Intel® Core™ i5-3320M 3.3 GHz or higher
RAM	8 GB or more
Hard disk	300 GB SSD

Hardware	Requirement
Monitor	15.6" Wide Screen Display (1920 x 1080)
Optical drive	DL MULTISTANDARD DVD RW

Supported virtualization platforms

You can install the "SIMATIC WinCC Basic" software package on a virtual machine. To do this, use one of the following virtualization platforms:

- VMware vSphere Hypervisor (ESXi) 5.5
- VMware Workstation 10
- VMware Player 6.0
- Microsoft Windows Server 2012 R2 Hyper-V

The following operating systems can serve as a host operating system for these virtualization platforms:

- Windows 7 Professional/Ultimate/Enterprise (64-bit)
- Windows 8.1 Professional/Enterprise (64-bit)
- Windows Server 2008 R2 (64-bit)
- Windows Server 2012 R2 (64-bit)

You can use the following guest operating systems to install "SIMATIC WinCC Basic" on the selected virtualization platform:

- Windows 7 Professional/Ultimate/Enterprise (64-bit)
- Windows 8.1 Professional/Enterprise (64-bit)

Note

- The same hardware requirements apply to the guest operating system as to the respective TIA products.
- The plant operator must provide sufficient system resources for the guest operating systems.
- The hardware certified by the manufacturers is recommended for the use of HyperV server and ESXi.

Supported security programs

The following security programs are compatible with "SIMATIC WinCC Basic":

- Virus scanners:
  - Symantec Endpoint Protection 12.1
  - Trend Micro Office Scan Corporate Edition 10.6
  - McAfee VirusScan Enterprise 8.8
  - Kaspersky Anti-Virus 2014
  - Windows Defender (Windows 8.1 and higher)
- Encryption software:
  - Microsoft Bitlocker
- Host-based Intrusion Detection System:
  - McAfee Application Control 6.0

Installing Microsoft .Net Framework

The software requires .Net Framework 4.5; it is automatically installed and activated.

You will be prompted to install the required .Net Framework version if this cannot be performed by the software package installation. A restart may be necessary after installation of .Net Framework.

Online help for Windows 7 / Windows Server 2008

Windows 7 and Windows Server 2008 no longer support all online help formats by default. These online help formats are used in WinCC in the following scenarios:

- Call-up of WinCC Direct Help
- Call-up of the WinCC Information System from the WinCC editors or via the Direct Help links

In order to still call WinCC Direct Help, the following component is installed during the installation of WinCC:

- Microsoft Help Engine

You can also call the WinCC Information System in Windows 7 and Windows Server 2008 from the Windows Start menu or from the installation folder.

To call the WinCC Information System from the WinCC editors or via the Direct Help links, some changes have to be made to the operating system. You can find more information on this in the section "More information for advanced users" of Microsoft Support article "917607": http://support.microsoft.com/kb/917607 (http://support.microsoft.com/kb/917607)

Security settings during installation

When you install WinCC V13, security settings are changed in your operating system.

The security settings in question are listed during the installation.

You must confirm the changes to the security settings.

If you make changes to your operating system after the installation, the changes to the security settings made by the installation of the TIA Portal could be changed.

You can restore the changes to the security settings made by the installation of the TIA Portal: "Start > All Programs > Siemens Automation > Security Controller > Restore settings".

SQL instance of WinCC V13

If you have installed a WinCC V11 product and you wish to install WinCC V13, you must uninstall the WinCC instance of SQL Server 2005 before the installation.

A new WinCC SQL 2008 instance is installed with the installation of WinCC V13.

See also

Licensing of WinCC Engineering System (Page 99)

3.2.4.2 Parallel installation

Parallel installations in TIA Portal V13

You will be prevented from starting the TIA Portal if you perform a non-permitted parallel installation of STEP 7 and WinCC. The following parallel installations are permitted in the TIA portal:

- STEP 7 V13 and WinCC V13

A dialog opens during installation to inform you of any inconsistencies in your parallel installation. The following parallel installations are permitted:

- WinCC V13 and RT Advanced V13
- WinCC V13 and RT Professional V13

The engineering system and Runtime must always be of the same version after an installation.

Parallel installation of WinCC V13 and other SIMATIC HMI products

Parallel installation of WinCC V13 and versions of WinCC flexible earlier than WinCC flexible 2008 SP1 is not allowed.

Parallel installation of WinCC V13 with versions of WinCC earlier than WinCC V7.0 SP2 is not allowed. Parallel installation of WinCC V13 with WinCC V7.0 SP2 or WinCC V7.0 SP3 is only allowed for:

- WinCC V 13 Professional
- WinCC V 13 Professional

Parallel use

If the term "Combo" appears in the name or license key of the software after installation, the use of the following products/versions is permitted in accordance with paragraph 1.6 of the General Terms and Conditions (see also setup text):

- With license for "WinCC V13 Comfort Combo": WinCC flexible 2008 Standard
- With license for "WinCC V13 Advanced Combo": WinCC flexible 2008 Advanced

3.2.4.3 Licenses and Powerpacks

Licensing of WinCC Engineering System

You require a license key for the following:

- WinCC Engineering System, for example, WinCC Professional
- Add-ons for WinCC Engineering System

You can install the license key during the installation of WinCC. You transfer licenses for the WinCC add-ons after installation with the Automation License Manager.

Starting without a valid license key

If you start WinCC without a valid license, the system alerts you that you are working in non-licensed mode. You have the one-time option of activating a trial license. The trial license for Engineering editions WinCC Basic, Comfort/Advanced, and Professional expires after 21 days.

When the trial license has expired, the following scenarios may occur:

- WinCC was never licensed on the PC in question.
  - Operations requiring a license can no longer be performed in WinCC.
- WinCC was already licensed on the PC in question.
  - Non-licensed mode is indicated every 10 minutes and at every action requiring a license by a window, which you must acknowledge.

License requirements for simulation

When you want to start simulation in WinCC using the "Online > Simulation> With tag simulator" menu command, you do not need licenses for WinCC Runtime or for add-ons that require a license.

If the following conditions are met, you will need the appropriate licenses for WinCC Runtime and add-ons that require a license even for simulation:

- The engineering station is connected to a PLC.
- The connection to the PLC is configured and active.

You start the simulation with the "Online > Simulation > Start" menu command.

See also

Software and hardware requirements (Page 94)

Licensing of HMI devices (Page 100)

Working with license keys (Page 101)

Licensing of HMI devices

Non-PC-based HMI devices are always equipped with "Runtime licenses". A license key is not required for runtime operation.

A license may be required for an add-on for non-PC-based HMI devices. The license key of the respective license always activates one add-on for use.

License key

In order to license non-PC-based HMI devices with license keys, the "SIMATIC HMI License Manager Panel Plugin" add-on is needed.

WinCC Setup installs this add-on by default. You open the License Manager Panel Plugin in the Automation License Manager with the "Edit > Connect Target System > Connect HMI Device" menu command.

If WinCC is not installed, an installation of ProSave 7.2 or higher is required.

Note

Further information on license handling can be found in the Automation License Manager help.

Note

Verify that the current version of the operating system (or higher) is installed on the HMI device before you start licensing. If necessary, update the operating system using ProSave.

Non-licensed mode

Runtime add-ons can be used for a short time without restrictions without a valid license. Non-licensed mode is indicated every 10 minutes by a window, which you must acknowledge.

See also

Licensing of WinCC Engineering System (Page 99)

Working with license keys

Introduction

You transfer a license key to the HMI device in the following cases:

- To use the WinCC engineering system
- To operate WinCC Runtime
- To use add-ons for WinCC Runtime on PC-based HMI devices
- To use add-ons on non-PC-based HMI devices

You transfer a license key from the HMI device in the following cases:

- When backing up data
- If you no longer require the license

You can then use this license on another PC or HMI device.

When you transfer a license to an HMI device, the associated license key is removed from the license key storage location.

Note

A license key cannot be copied. The copy protection employed prevents the license keys from being copied.

Data backup

You transfer the license keys from the HMI device for data backup on the HMI device or as a backup for device replacement.

You use the Automation License Manager to back up license keys from an HMI device to the storage area of the license keys.

NOTICE

Destruction of license keys on non-PC-based HMI devices

License keys transferred as a result of backup/restore operations are destroyed in the case of the following HMI devices.

- 270 series
- 370 series

Carry out the following before beginning restoring:

- Use the Automation License Manager and ProSave to check whether license keys are on the HMI device.
- Transfer the license keys present on the HMI device to a storage location.
  After restoring has been carried out, transfer the license keys back to the HMI device.

NOTICE
Destruction of license keys on PCs Transfer all license keys to a storage location in the following cases: • Before you format the hard disk • Before you compress the hard disk • Before you restore the hard disk • Starting an optimization program that moves fixed blocks • Installing a new operating system Read the description of Automation License Manager ("Start > Simatic Automation > Documentation"). Observe all warnings and notices.

The license key storage location on PC-based HMI devices and on non-PC-based HMI devices where Automation License Manager is used may contain multiple license keys. This capability means you can store multiple licenses of the same type at one location. For the backup, use a single storage location for all license keys present on an HMI device.

Note

Always keep the original storage location of the license keys.

Invalid license after time zone change

The transferred license no longer functions in the following case.

- If you change the time zone on a WinCC PC as follows:
  - From a time based on a complete hour to a time not based on a complete hour.
    Example: You change the time zone from GMT +3:00 to GMT +3:30.

To work around this, transfer the license key from the HMI device with a time zone setting that was set when the license key was transferred to the HMI device.

Example:

You transferred the license key to the HMI device with a time zone setting based on a full hour. Then, also transfer the license key from the HMI device with a time zone setting based on a full hour.

This behavior does not apply to the trial license.

Defective license

A license is defective in the following cases:

- If the license key is no longer accessible at the storage area.
- If the license key disappears during its transfer to the destination drive.

Note

Resetting of the system date to an earlier time causes all licenses to be become defective.

You can use the Automation License Manager to repair the defective license. Use the "Restore" function or the "Restore Wizard" of the Automation License Manager for this purpose. You must contact Customer Support in order to restore the license. For additional information see: http://support.automation.siemens.com (http://support.automation.siemens.com)

Note

The Runtime software can also be operated without errors if the license is missing or defective. The system alerts you at brief intervals that you are working in non-licensed mode.

Note

If you start the WinCC Engineering System without a valid license key, the system alerts you that you are working in non-licensed mode. You have the one-time option of activating a trial license. The trial license expires after 21 days.

When the trial license has expired, the following scenarios may occur:

- WinCC was never licensed on the PC in question.
  WinCC can no longer be started.
- WinCC was already licensed on the PC in question.
  WinCC can be started. Non-licensed mode is indicated every 10 minutes by a window, which you must acknowledge.

See also

Licensing of WinCC Engineering System (Page 99)

3.3 Installation log

Function of the installation log

The progress during the following installation processes is logged in a file:

- Installing products
- Modifying or updating already installed products
- Repairing an existing installation
- Uninstalling products

If errors occur during the installation process or warnings are issued, these can be evaluated with the help of the log file. You can do this yourself or contact product support.

Installation logs storage location

The log file is the most recent file with the file extension ".log" and the name of which that starts with "SIA".

The location of the log file is stored in the environment variable "%autinstlog%". You can enter this environment variable in the address bar of Windows Explorer to open the folder with the log files. Alternatively, you can navigate to the corresponding directory by entering "CD %autinstlog%" in the command line.

The location depends on the operating system, e.g. "C:\Documents and Settings\All Users \Application Data\Siemens\Automation\Logfiles\Setup" in the English version of Windows XP.

Setup_Report (CAB file)

To make it easier to provide Product Support with all necessary files, an archive file that contains the installation log and all other required files is saved in CAB format. This archive can be found at "%autinstlog%\Reports\Setup_report.cab". Send this CAB file to Product Support if you need assistance with installation. With this information, Product Support can determine whether the installation was executed properly. CAB files that were generated during earlier installation processes are saved with a date ID in the "Reports" directory.

See also

Notes on licenses (Page 84)

Starting installation (Page 104)

Checking availability of updates and support packages and installing them (Page 106)

Displaying the installed software (Page 108)

Modifying or updating installed products (Page 109)

Repairing installed products (Page 110)

Starting to uninstall (Page 112)

3.4 Starting installation

Introduction

Software packages are installed automatically by the setup program. The setup program starts once the installation medium has been inserted in the drive.

Requirement

- Hardware and software of the programming device or PC meet the system requirements.
- You have administrator privileges on your computer.
- All running programs are closed.

Procedure

To install the software packages, follow these steps:

1. Insert the installation medium in the relevant drive.
   The setup program starts automatically unless you have disabled Autostart on the programming device or PC.
2. If the setup program does not start up automatically, start it manually by double-clicking the "Start.exe" file.
   The dialog for selecting the setup language opens.
3. Choose the language in which you want the setup program dialogs to be displayed.
4. To read the information on the product and installation, click the "Read Notes" or "Installation Notes" button.
   The help file containing the notes opens.
5. Once you have read the notes, close the help file and click the "Next" button.
   The dialog for selecting the product languages opens.
6. Select the languages for the product user interface, and click the "Next" button.

   Note

   "English" is always installed as the basic product language.

   The dialog for selecting the product configuration opens.
7. Select the products you want to install:
   - If you wish to install the program in a minimal configuration, click on the "Minimal" button.
   - If you wish to install the program in a typical configuration, click on the "Typical" button.
   - If you wish to personally select the products to be installed, click on the "User-defined" button. Then select the check boxes for the products you wish to install.
8. If you want to create a shortcut on the desktop, select the "Create desktop shortcut" check box.
9. Click the "Browse" button if you want to change the target directory for the installation. Note that the length of the installation path must not exceed 89 characters.
10. Click the "Next" button.
    The dialog for the license terms opens.
11. To continue the installation, read and accept all license agreements and click "Next".
    If changes to the security and permissions settings are required in order to install the TIA Portal, the security settings dialog opens.
12. To continue the installation, accept the changes to the security and permissions settings, and click the "Next" button.
    The next dialog displays an overview of the installation settings.
13. Check the selected installation settings. If you want to make any changes, click the "Back" button until you reach the point in the dialog where you want to make changes. Once you have completed the desired changes, return to the overview by clicking on "Next".

14. Click the "Install" button.
    Installation is started.

    Note

    If no license key is found during installation, you have the chance to transfer it to your PC. If you skip the license transfer, you can register it later with the Automation License Manager.

    Following installation, you will receive a message indicating whether the installation was successful.

15. It may be necessary to restart the computer. If this is the case, select the "Yes, restart my computer now." option button. Then click "Restart".
16. If the computer does not reboot, click "Exit".

Result

The TIA Portal along with the products and licenses you have ordered and the Automation License Manager have been installed on your computer.

See also

Installation log (Page 103)

Notes on the system requirements (Page 85)

Notes on licenses (Page 84)

Displaying the installed software (Page 108)

Modifying or updating installed products (Page 109)

Repairing installed products (Page 110)

Starting to uninstall (Page 112)

3.5 Checking availability of updates and support packages and installing them

In the TIA Portal, you have the option of checking whether new software updates or support packages are available, for example, Hardware Support Packages (HSPs). If this is the case, you can then install the software.

Note

Updates and support packages from TIA Portal V13 or higher are supported.

Procedure

To check the availability of software updates and support packages and then install these, follow these steps:

1. Use one of the two following methods to start the installation process:
   - Click "Installed software" in the "Help" menu in the TIA Portal.
     The "Installed software" dialog opens.
   - Open the "Siemens TIA Updater" program using the program link on your computer.
     The "Siemens TIA Updater" dialog is opened and the available software packages are displayed.
2. Click "Check for updates". This step is optional if the "Siemens TIA Updater" dialog is already open.
   If you started the installation procedure in the TIA Portal, the "Siemens TIA Updater" dialog is opened and the available updates are displayed.
3. Click "Download" in the line of the update or support package that you want to install.
   The update or support package is downloaded. The associated "Install" button becomes active as soon as the download process has been completed.

   Note

   Please note the following:

   1. You can initiate multiple download processes simultaneously.
   2. You can log off or also shut down the computer while the download process is running. In such cases, the download process continues in the background as soon as you log on again.

4. Close the TIA Portal if it is still open.
5. In the "Siemens TIA Updater" dialog, click the "Install" button of the software package you want to install.
   The installation dialog appears.

   Note

   Please note the following:

   1. It is not possible to install multiple updates simultaneously.
   2. Do not log off during an installation and do not shut down the computer. This avoids inconsistent versions of the software on your computer.

6. Click "Next".
   The selected product is installed.

Alternative procedures for the installation of support packages

Another procedure is available for the installation of a support package. To do this, follow these steps:

1. Click "Support packages" in the "Options" menu in the TIA Portal.
   The "Detailed information" dialog opens. A table lists all support packages from the directory that you selected as the storage location for support packages in the settings.
2. If you want to install a support package that is not in the list, you have the following options:
   – If the support package is already on your computer, you can add it to the list by selecting "Add from the file system".
   – If you add a support package from the "Service & Support" page on the Internet, first you download it by selecting "Download from the Internet". Then you can add it from the file system.
3. Select the support package that you want to install.
4. Click "Install".
5. Close and then restart the TIA Portal.

See also

Installation log (Page 103)

3.6 Displaying the installed software

You can find out which software is installed at any time. In addition, you can display more information on the installed software.

Procedure

To display an overview of the software installed, follow these steps:

1. Click "Installed software" in the "Help" menu.
   The "Installed software" dialog opens. You will see the installed software products in the dialog. Expand the entries to see which version is installed in each case.
2. If you would like to display additional information on the installed automation software, click the link on the "Detailed information about installed software" dialog.
   The "Detailed information" dialog opens.
3. Chose the topic you want more information about in the area navigation.

See also

Notes on the system requirements (Page 85)

Notes on licenses (Page 84)

Starting installation (Page 104)

Modifying or updating installed products (Page 109)
Repairing installed products (Page 110)
Starting to uninstall (Page 112)
Installation log (Page 103)

3.7 Modifying or updating installed products

You have the option to modify installed products using the setup program or to update to a new version.

Requirement

- Hardware and software of the programming device or PC meet the system requirements.
- You have administrator privileges on your computer.
- All running programs are closed.

Procedure

To modify or update installed products, follow these steps:

1. Insert the installation medium in the relevant drive.
   The setup program starts automatically unless you have disabled Autostart on the programming device or PC.
2. If the setup program does not start up automatically, start it manually by double-clicking the "Start.exe" file.
   The dialog for selecting the setup language opens.
3. Choose the language in which you want the setup program dialogs to be displayed.
4. To read the information on the product and installation, click the "Read Notes" or "Installation Notes" button.
   The help file containing the notes opens.
5. Once you have read the notes, close the help file and click the "Next" button.
   The dialog for selecting the installation variant opens.
6. Select the "Modify/Upgrade" option button and click the "Next" button.
   The dialog for selecting the product languages opens.
7. Select the check boxes of the product languages that you want to install. You can remove previously installed product languages by clearing the corresponding check boxes.

   Note

   Note that the product language "English" cannot be removed.

8. Click the "Next" button.
   The dialog for selecting the product configuration opens.

9. Select the check boxes of the components that you want to install. You can remove previously installed components by clearing the corresponding check boxes.
10. Click the "Next" button.

Note

Note that you cannot change the target directory because the existing installation is being modified.

If changes to the security and permissions settings are required in order to install the TIA Portal, the security settings dialog opens.

11. To continue the installation, accept the changes to the security and permissions settings, and click the "Next" button.
    The next dialog displays an overview of the installation settings.
12. Click the "Modify" button.
    This starts the installation of the additional components.

Note

Following installation, you will receive a message indicating whether the existing installation was successfully changed.

13. It may be necessary to restart the computer. If this is the case, select the "Yes, restart my computer now." option button. Then click "Restart".
14. If the computer does not reboot, click "Exit".

Result

The existing installation has been modified on your computer.

See also

Notes on the system requirements (Page 85)

Notes on licenses (Page 84)

Starting installation (Page 104)

Displaying the installed software (Page 108)

Repairing installed products (Page 110)

Starting to uninstall (Page 112)

Installation log (Page 103)

3.8 Repairing installed products

You have the option to repair installed products by completely reinstalling them using the setup program.

Requirement

- Hardware and software of the programming device or PC meet the system requirements.
- You have administrator privileges on your computer.
- All running programs are closed.

Procedure

To repair installed products, follow these steps:

1. Insert the installation medium in the relevant drive.
   The setup program starts automatically unless you have disabled Autostart on the programming device or PC.
2. If the setup program does not start up automatically, start it manually by double-clicking the "Start.exe" file.
   The dialog for selecting the setup language opens.
3. Choose the language in which you want the setup program dialogs to be displayed.
4. To read the information on the product and installation, click the "Read Notes" or "Installation Notes" button.
   The help file containing the notes opens.
5. Once you have read the notes, close the help file and click the "Next" button.
   The dialog for selecting the installation variant opens.
6. Select the "Repair" option button, and click the "Next" button.
   The next dialog displays an overview of the installation settings.
7. Click the "Repair" button.
   This starts the repair of the existing installation.

   Note

   Following installation, you will receive a message indicating whether the existing installation was successfully repaired.

8. It may be necessary to restart the computer. If this is the case, select the "Yes, restart my computer now." option button. Then click "Restart".
9. If the computer does not reboot, click "Exit".

Result

The installed products have been reinstalled.

See also

Notes on the system requirements (Page 85)

Notes on licenses (Page 84)

Starting installation (Page 104)

Displaying the installed software (Page 108)

Modifying or updating installed products (Page 109)
Starting to uninstall (Page 112)
Installation log (Page 103)

3.9 Starting to uninstall

Introduction

Software packages are removed automatically by the setup program. Once started, the setup program guides you step-by-step through the entire removal procedure.

You have two options for removing:

- Removing selected components via the Control Panel
- Removing a product using the installation medium

Note

The Automation License Manager will not be removed automatically when you remove the software packages, because it is used for the administration of several license keys for products supplied by Siemens AG.

Removing selected components via the Control Panel

To remove selected software packages, follow these steps:

1. Open the Control Panel with "Start > Settings > Control Panel".
2. Double click on "Add or Remove Programs" in the control panel.
   The "Add or Remove Programs" dialog opens.
3. Select the software package to be removed in the dialog "Add or Remove Programs", and click "Remove".
   The dialog for selecting the setup language opens.
4. Select the language in which you want the dialogs of the setup program to be displayed and click "Next".
   The dialog for selecting the products you want to remove opens.
5. Select the check boxes for the products that you want to remove and click "Next".
   The next dialog displays an overview of the installation settings.
6. Check the list with the products to be removed. If you want to make any changes, click the "Back" button.
7. Click the "Uninstall" button.
   Removal begins.
8. It may be necessary to restart the computer. If this is the case, select the "Yes, restart my computer now." option button. Then click "Restart".
9. If the computer does not reboot, click "Exit".

Removing a product using the installation medium

To remove all software packages, follow these steps:

1. Insert the installation medium in the relevant drive.
   The setup program starts automatically unless you have disabled Autostart on the programming device or PC.
2. If the setup program does not start up automatically, start it manually by double-clicking the "Start.exe" file.
   The dialog for selecting the setup language opens.
3. Choose the language in which you want the setup program dialogs to be displayed.
4. To read the information on the product and installation, click "Read product information" or "Read installation notes".
   The help file containing the notes opens.
5. Once you have read the notes, close the help file and click the "Next" button.
   The dialog for selecting the installation variant opens.
6. Select the "Uninstall" option button and click the "Next" button.
   The next dialog displays an overview of the installation settings.
7. Click the "Uninstall" button.
   Removal begins.
8. It may be necessary to restart the computer. If this is the case, select the "Yes, restart my computer now." option button. Then click "Restart".
9. If the computer does not reboot, click "Exit".

See also

Installation log (Page 103)

Notes on the system requirements (Page 85)

Notes on licenses (Page 84)

Starting installation (Page 104)

Displaying the installed software (Page 108)

Modifying or updating installed products (Page 109)

Repairing installed products (Page 110)

3.10 Installing and uninstalling the migration tool

3.10.1 System requirements

System requirements for the migration tool

The following system requirements apply to the use of the migration tool:

- All products used to create the source project must be installed. The following products are supported:
  - WinCC flexible 2008 SP2 and SP3
  - WinCC V7.0 SP3
  - STEP 7 V5.5
  - Integrated projects from STEP 7 V5.5 and the WinCC products listed above
  - SINUMERIK
  - STARTER and Startdrives
  - SIMOTION SCOUT V4.4
    You need the SCOUT Migration Tool PlugIn V4.4 to migrate SIMOTION SCOUT V4.4 projects.
- All optional packages needed to process the STEP 7 project are installed. For example, all HSPs for the devices used in the source project are required.

3.10.2 Installing the migration tool

Distribution of the migration tool

You can find the migration tool in the "Support" directory on the installation DVD of the TIA Portal. Alternatively, it can be downloaded from the Service & Support area of the Siemens website. For some products, (such as SIMATIC Failsafe or SIMOTION) additional plug-ins are required for the migration tool. These plug-ins can also be downloaded from the Service & Support page or installed from the installation DVD of the specific products.

Normally, the migration tool is installed without the TIA Portal. Because the TIA Portal has its own integrated migration function, a separate installation of the migration tool is not necessary.

Procedure

To install the migration tool, proceed as follows:

1. Download the installation DVD from the Service & Support area of the Siemens website or use the installation file from the "Support" directory of the installation DVD of the TIA portal to perform the installation.
2. Run the installation file.
   The setup program for the migration tool will open.
3. First, select the language in which the setup should be displayed and click the "Next" button.
   The page for selecting the software language is displayed.
4. Since the migration tool is provided exclusively in English, you cannot choose any other language for the installation. Therefore, click "Next" to proceed to the next step.
   The page for selecting the product is displayed.
5. The migration tool consists solely of a software component. Therefore, the migration tool is already selected.
   To create a Desktop icon for starting the migration tool, select the check box "Create Desktop icon". Then click the "Next" button.
   The page for confirming the licensing terms is shown.
6. Click on an entry in the list of license terms to read the selected license term. If you agree with all license terms, select the check box "I accept the terms of the displayed license agreement". Then click the "Next" button.
   An overview of the installation is displayed.
7. Click the "Install" button.
   The installation is performed with the displayed settings.

3.10.3 Uninstalling the migration tool

The migration tool can be removed using the Control Panel.

Procedure

To remove the migration tool, follow these steps:

1. Open the Control Panel.
2. Double click on "Add or Remove Programs" in the Control Panel.
   The "Add or Remove Programs" dialog opens.
3. Select the entry for the migration tool in the "Add or Remove Programs" dialog, and click the "Remove" button.
   A security prompt appears.
4. Click the "Uninstall" button to confirm this prompt.
   The migration tool will be removed.

Migrating projects and programs 4

4.1 Migrating projects in a TIA portal project

4.1.1 Migration of projects with the TIA Portal

Migration of existing projects

You can migrate projects from earlier automation solutions to the TIA Portal. Each time you migrate, a new project is created for the migrated data with which you can then work. Any TIA Portal projects already open are closed first.

The migration is then displayed in the table of the project history. From there, you have access to the migration log that is created automatically for the migration.

Supported products for migration

The chapter "System overview STEP 7 and WinCC" includes information on the products that are available for the TIA Portal. In principle, all products listed there are supported by the TIA Portal during migration.

Any additional requirements that must be met depend on the initial products that were used and the currently installed products. For more information on the migration options for your products, you can, for example, refer to the Siemens Industry Online Support and the documentation of your software products.

See also: Scaling of STEP 7 and WinCC in the TIA Portal (Page 27)

Procedure during migration

The migration process is divided into the following basic steps:

1. Preparing the initial project
   If the software for editing the initial project is not or not fully installed on the programming device/PC with the TIA Portal, or if the initial project is an integrated project, the initial project must first be converted into a migration file. To do this, install the migration tool on a programming device/PC on which the required software for editing the initial project is installed. Then, use the migration tool to convert the initial project, and copy the file to the programming device/PC on which the TIA Portal is installed. You can omit this step if the initial project and its associated software are on the same programming device/PC as the TIA Portal, and if the initial project is not an integrated project.
2. Performing migration
   Perform the actual migration within the TIA Portal. For the migration, specify as source either the migration file which you created with the migration tool or the initial project when all required software has been installed.

3. Checking the migration log
   A migration log is created for each migration. It contains information about modified project parts. You can call the log under "Common data > Logs" in the project tree or in the project history. After completion of the migration, the migration log will be displayed in the TIA Portal. Check the log following completion of the migration.
   If the migration failed, an XML file is created as a log under "\Logs" in the project directory. You can use any XML editor to open this log and view the reasons why the migration failed.
4. Correcting the migrated project
   Because the configurations of the initial project may not always be completely compatible with the TIA Portal, not all configurations are transferred in identical form in the migrated project. You should therefore work through the points in the migration log systematically. If you have not included the hardware configuration in the migration, you also have to convert the unspecified devices to the appropriate hardware.

Including the hardware configuration in the migration

By default, only the software parts of the project are included in the migration. An unspecified device is generated in the migrated project for the devices contained in the initial project. The hardware and network configurations and the connection are not migrated. You can convert the unspecified devices into suitable devices after the migration and create any network configurations and connections manually.

If you are certain that the hardware used in the initial project has a corresponding equivalent in the TIA Portal, you can include the hardware configuration in the migration. In this case, both the hardware configuration and the software are migrated.

See also

Display migration log (Page 123)

Scaling of STEP 7 and WinCC in the TIA Portal (Page 27)

4.1.2 Preparing projects with the migration tool

4.1.2.1 Migrating projects with the migration tool

Preparation for migration

In many cases, a project that you wish to migrate will not be located on the same programming device/PC on which the latest version of the TIA Portal is installed. Therefore, the initial project must first be converted to a compatible format for the migration. The same applies to integrated projects.

After creation of the migration file, you copy the migration file to the programming device/PC on which the current version of the TIA Portal is installed. In the TIA Portal, enter the migration file as source for the migration. You can now create a project in the current file format of the TIA Portal.

Procedure for migration with the migration tool

The following steps are necessary to prepare a migration with the migration tool:

1. Install the migration tool on the programming device/PC where the source project is located. To do this, download the installation file from the Siemens Industry Online Support or install the migration tool from the setup DVD of the TIA Portal.
2. Start the migration tool, and use it to convert the source project to the migration file format with file extension ".am13".
   For this step, make sure that all software needed to process the source project is installed on the programming device/PC. This also includes all necessary service packs, hardware support packages and all expansion software that is needed to process the initial project. If individual products are not installed it may not be possible to perform the migration or the migration may be incomplete.
3. Copy the migration file to the target system on which a current version of the TIA Portal is installed.
   Note that the target system must have been installed with all software needed to configure the complete set of devices contained in the migration.
4. Perform the migration within the TIA Portal, and specify the migration file with the extension ".am13" as the source.
5. Once migration is complete, check the migration log and systematically work through the information provided there for the newly created project. Read the information in the Inspector window with special care after the first compilation of the configuration.

Including the hardware configuration in the migration

By default, only the software parts of the project are included in the migration. An unspecified device is generated in the migrated project for the devices contained in the initial project. The hardware and network configurations and the connection are not migrated. You can convert the unspecified devices into suitable devices after the migration and create any network configurations and connections manually.

If you are certain that the hardware used in the initial project has a corresponding equivalent in the TIA Portal, you can include the hardware configuration in the migration. In this case, both the hardware configuration and the software are migrated.

See also

Migration of projects with the TIA Portal (Page 117)

Migrating projects (Page 121)

Calling the migration tool (Page 120)

Creating a migration file (Page 120)

4.1.2.2 Calling the migration tool

Starting the migration tool

During the installation, a "Migration to TIA Portal V13" shortcut is created as standard in the Start menu under "Siemens Automation > Migration Tool". Click this shortcut.

Alternatively, you can call the migration tool directly in Windows Explorer. During the installation, the migration tool is saved by default in one of the following folders:

- On a 64-bit operating system:
  C:\Program Files (x86)\Siemens\Automation\MIGTOOL_V13\Bin
- On a 32-bit operating system:
  C:\Program Files\Siemens\Automation\MIGTOOL_V13\Bin

To start the migration tool, click the "Siemens.Automation.MigrationApplication.exe" file in one of the directories.

See also

Creating a migration file (Page 120)

4.1.2.3 Creating a migration file

The section below describes how you can use the migration tool to convert the initial project into a migration file that can be read by the TIA Portal. Following conversion, this file is transferred to the target system and migrated there.

You can specify whether the migration file should contain the entire project, including the complete hardware configuration and the associated software, or whether you want to migrate the software only.

Requirements

- The suitable, original software with a valid license is installed for all configurations used in the initial project.
- The initial project is not provided with access protection.
- The initial project must be consistent, otherwise problem-free migration cannot be assured.

Procedure

To create the migration file, proceed as follows:

1. Choose the path of the source file for the migration in the "Storage Location (Path)" field.
2. Specify which project parts are to be migrated:
   - Select the "Include HW and Network data during the migration" check box to migrate not only the software but also the complete hardware parts and the network configuration of the project.
   - Select the "Copy SCADA runtime data" check box if you also want to migrate the runtime data, such as alarm archives, tag archives and user archives, in addition to the data of the engineering system.
3. Choose the path and the file name for the migration file in the "Intermediate file" field.
4. Click the "Migrate" button.

Result:

A migration file is created. Finally, copy this file to the target system and migrate this file in the TIA Portal.

See also

Migrating projects (Page 121)

Calling the migration tool (Page 120)

Migrating projects with the migration tool (Page 118)

4.1.3 Migrating projects

Requirements

- A converted file in the format ".am13" is already available or the original software with a valid license is installed for all configurations used in the initial project.
- The initial project is not provided with access protection.
- The initial project must be consistent, otherwise problem-free migration cannot be assured.

Read the additional information on the requirements in the help for the respective products installed.

Note

System hibernation during the migration

While a migration is running, the system should not be changed to the standby or hibernate mode. Otherwise the migration will be aborted.

Procedure

To migrate a project, follow these steps:

1. Select the "Migrate project" command in the "Project" menu.
   The "Migrate project" dialog opens.
2. Specify the path and the file name for the project to be migrated in the "Source path" field.
   Choose either a project in the ".am12" migration format or in the format of the initial project.
3. To include the hardware configuration in the migration, select the "Include hardware configuration" check box.
   If you have selected a migration file that was created with the migration tool, the check box cannot be selected. In this case, you must specify if you wish to include the hardware configuration in the migration before the conversion with the migration tool .
4. Select the "Copy WinCC Runtime Professional data" check box, if you also want to migrate the runtime data, such as alarm archives, tag archives and user archives, in addition to the data of the engineering system.
   If you have selected a migration file that was created with the migration tool, the check box cannot be selected. In this case, you must specify if you wish to include the SCADA runtime data in the migration before the conversion with the migration tool .
5. Choose a name for the new project in the "Project name" box.
6. Choose a path in the "Target path" box where the new project will be created.
7. Enter your name or the name of another person responsible for the project in the "Author" field.
8. Enter a comment in the "Comment" box, if you require one.
9. Click "Migrate".

Result

The initial project is converted and a message appears after conversion is complete. The newly created project is then opened in the project view and the migration log is opened in the TIA Portal.

Even if the migration failed, a project directory is created and a migration log in the form of an XML file is generated in this directory. The completion message that appears after the migration contains a link to this XML file. Click the link to open the XML file. Alternatively, you can find the XML file in the project directory under "\Logs".

See also

Post-editing integrated projects (Page 174)

Display migration log (Page 123)

Using logs (Page 331)

Migrating projects with the migration tool (Page 118)

Creating a migration file (Page 120)

4.1.4 Displaying the history of the migration

If a project was created by migration, the migration will be listed in the table of the project history. You can open the migration log in the table. The time of the migration is also shown.

Procedure

To display the migration in an overview table, follow these steps:

1. Select the open project in the project tree.
2. Select "Properties" in the shortcut menu of the project.
   The dialog with the properties of the project opens.
3. Select the "Project history" group in the area navigation.
   The overview table is displayed.

See also

Displaying properties of the project (Page 337)

4.1.5 Display migration log

A log is created for each successful migration. The log contains the following information:

- Migrated objects
- Modifications to objects made during migration
- Errors that occurred during migration
- In certain cases a link to more help with specific events.
  In this case, click the question mark to obtain more help.

Procedure

To display the log file of the migration, follow these steps:

1. Open the "Common data > Logs" folder in the project tree.
2. Double-click the desired log in the list.
   The contents of the log are displayed in the work area.

See also

Migration of projects with the TIA Portal (Page 117)

Using logs (Page 331)

4.1.6 Migrating STEP 7 projects

4.1.6.1 Migration of STEP 7 projects

Introduction

You can migrate a project from SIMATIC STEP 7 V5.4 SP5, or from SIMATIC STEP 7 V5.5 to be able to continue to use it in the TIA Portal. If you want to migrate projects from older STEP 7 versions, you have to open, compile and save these projects first in STEP 7 V5.4 SP5 or V5.5. It is also possible to migrate integrated projects with devices from other software products.

Scope of migration

Generally, you can migrate all of the configurations and objects from SIMATIC STEP 7 V5.4 SP5 or V5.5 which also support the installed version of the TIA Portal. This includes, for example, the following devices and configurations:

- Devices of the S7-300 and S7-400 family.
- PROFIBUS configurations with connected distributed I/O. These include GSD-based slaves, I-slaves as well as direct data exchange.
- PROFINET configurations with distributed I/O. These include GSD-based devices and I-devices.
- Network configurations
- Connections
- Blocks created in the programming languages LAD, FBD or STL, S7-SCL, S7-GRAPH
- PLC tags
- User-defined data types (UDT)
- Alarms and alarm classes
- Interrupts
- User-defined attributes (UDA)
- User text libraries

Including the hardware configuration in the migration

By default, only the software parts of the project are included in the migration. An unspecified device is generated in the migrated project for the devices contained in the initial project. The hardware and network configurations and the connection are not migrated. You can convert the unspecified devices into suitable devices after the migration and create any network configurations and connections manually.

If you are certain that the hardware used in the initial project has a corresponding equivalent in the TIA Portal, you can include the hardware configuration in the migration. In this case, both the hardware configuration and the software are migrated.

See also

Reporting system errors (Page 132)

General notes on migrating program blocks (Page 132)

4.1.6.2 Requirements for the migration of STEP 7 projects

Various requirements for the software installed on the original PG/PC and for the initial project apply to the migration.

Requirements for the original PG/PC

The following requirements apply to the PG/PC:

- A licensed version of STEP 7 V5.4 SP5 or V5.5 must be installed. For projects that only contain PC stations, it is sufficient if the SIMATIC NET PC software is installed.
- For all of the configurations used in the project, the corresponding additional software must be installed with a valid license, for example, optional packages.
- All HSPs for modules not included in the hardware catalog must be installed.
- All GSD files used in the project must comply with the specifications.
- You must be logged on to the operating system with administrative rights.
- One of the following software products must be installed:
  - The TIA Portal with STEP 7 Professional V13 or higher
    If you do not migrate with the most recent version of the TIA Portal to the original programming device/PC, the functional scope of the project in the more recent version of the TIA Portal is initially limited. You first have to upgrade the project before you can use the full functional scope of more recent versions.
  - The migration tool

Requirements for the initial project

- The initial project must not be configured with access protection.
- The hardware and software must be consistent.
- The alarm number assignment must be set to CPU-oriented.
- It must contain no protected blocks with time stamp conflict.
- It must be possible to compile all programs and their sources without errors.
- All called blocks must be available in the block folder.

- The block folder must contain no blocks that are not called, in particular no instance data blocks.
- We recommend that you migrate blocks unencrypted, because blocks encrypted with KNOW_HOW_PROTECT (from STEP 7 V5.x) or S7-Block Privacy (from STEP 7 V5.5) cannot be read or modified in the TIA Portal.

Requirements for the PG/PC with STEP 7 Professional

- One of the following software products must be installed:
  - STEP 7 Professional V13 or later
  - STEP 7 Professional V13 or later (without license) when the project to be migrated only contains components that have been configured with the SIMATIC NET PC software.
- For all of the configurations used in the project, the corresponding additional software must be installed with a valid license, for example, optional packages.
- All HSPs for modules that are not contained in the hardware catalog and all GSD files used in the project must be installed.

4.1.6.3 Checking whether STEP 7 projects can be migrated

Before you launch migration, you can check if all necessary requirements for migration have been met.

Note

Note that the actual values are reset in the original project when the following steps are carried out. Also note that running the block consistency check in STEP 7 V5.x resets the initial values in the instance DBs to the default values. If you are using technology functions, this behavior is not desired.

Procedure

To check whether a project can be migrated, follow these steps:

1. Open the original project in SIMATIC STEP 7 V5.4 SP5 or V5.5. For a correct update, it is necessary that only the optional packages and modules, including HSPs and GSD files, that are contained in the TIA Portal are installed.
2. Open the individual stations. If no messages indicating missing components are displayed when the stations are opened, all the components necessary for migration are available.
3. Perform a block consistency check for each block container contained in the project.
   For more information on the consistency check, refer to Chapter "Consistency check in initial project prior to the migration (Page 127)".
4. Compile the entire project. If no errors are displayed during the compilation, the programs can be migrated.
5. Check whether all modules with an identical MLFB and the same firmware version are contained in the TIA Portal hardware catalog.

6. Check whether the CPU-oriented message number assignment is activated for each CPU.
7. Compile the project in NetPro. Here too, no errors should be reported.

See also

Consistency check in initial project prior to the migration (Page 127)

FAQ entry on consistency check (http://support.automation.siemens.com/WW/view/en/5416540)

4.1.6.4 Consistency check in initial project prior to the migration

Consistent initial project

A consistent initial project is a requirement for the successful migration of a SIMATIC STEP 7 V5.x project to the TIA Portal. It is therefore advisable to perform a consistency check on the initial project before migration. You can find a guide to the consistency check in SIMATIC STEP 7 V5.x in an FAQ entry (http://support.automation.siemens.com/WW/view/en/5416540) on the Service and Support page or in the SIMATIC STEP 7 V5.x help.

Migrate start values in instance DBs for technology blocks

The compilation in the block consistency check causes the tags in instance DBs to be reset to the default values. The set start values are lost in the process.

To migrate the start values anyway, we recommend one of the following procedures:

- For smaller projects: Backup the instance DBs outside the block folder.
  Backup the instance DBs before you start the consistency check. After the consistency check has been completed, copy the instance DBs back to the block folder.
- For larger projects: Create a copy of the project
  Make a copy of the project and then carry out the check on the copy of the project. Correct any errors found during the consistency check in the initial project.

You can then migrate the project with the configured values. In this way, the configuration of the technology functions is also retained.

See also

Basic information on start values (Page 2836)

4.1.6.5 Removing unsupported hardware components

If you have created configurations in the initial project for which not all of the right software is installed, then you can still migrate the project by manually deleting the unsupported configurations from the project. Alternatively, you can use the "Reorganize" function.

Procedure

To clean the original project, follow these steps:

1. Use an Installation of SIMATIC STEP 7 V5.4 or V5.5 that contains only the optional packages contained in the TIA Portal and all required modules.
2. Save the project again and select the "With reorganization" option when saving. Any unsupported configurations will be removed from the project.

For modules that are only available in a new version or with a new firmware version in the TIA Portal, use the "Exchange object" function in the station to exchange an older module for one that can be migrated.

4.1.6.6 Special points to observe during migration

Device names

Device names of PROFINET IO devices

In STEP 7 V5.5 the device name of a PROFINET IO device is used as the name of the IO devices. After the migration to a TIA Portal project, the device name of the PROFINET IO device from the initial project is used as follows:

- As the PROFINET device name
- As the name for the interface module
- As the name of the station
  A maximum of 24 characters is permitted for the name of the station in the TIA Portal. The name of the station is therefore truncated to a maximum of 24 characters where required. If truncating of the station name would result in several stations with the same name, the last characters of the station name are used to number the stations consecutively.

Names of modules and racks

In SIMATIC STEP 7 V5.5, the names of modules and racks do not have to be unique. In the TIA Portal, however, unique names are required. Therefore, the names of the modules and racks are incremented during the migration to make sure that unique names are assigned. This does not apply to PROFINET IO devices and PROFINET CPs. You must provide unique names for these yourself.

PROFINET IO configurations

Unique name assignment required

The names of PROFINET IO devices must be unique within the subnet. The same applies to the names of PROFINET CPs. For this reason, ensure that the name assignment is unique before the migration into the initial project or provide unique names after migration and before the first compilation.

Media redundancy with sync domains

If a sync domain with activated media redundancy is found in the initial project, the sync domain configuration is imported during migration. If, however, several sync domains with activated media redundancy are found in the initial project, they are removed and the project is migrated without the sync domains. In any case, you will receive a message which sync domains were imported.

In any case, use SIMATIC STEP 7 V5.5 to configure the media redundancy in the initial project.

Isochronous mode with PROFINET IO

Observe the following for PROFINET IO configurations in isochronous mode:

- IRT with high performance
  If you have configured IRT with high performance when planning the topology in the initial project, the option is changed to "RT" during migration. You will receive a message informing you of the change.
- Use of OB 61
  If the isochronous mode for PROFINET IO with OB 61 is activated in the original configuration in the STEP 7 project, it will be deactivated by the migration. The devices will still be imported. PROFINET IRT configurations with the option "high flexibility" without the use of OB 61 can be migrated.

CBA for IO controllers

If you have used the cyclical service CBA (Component Based Automation) in the initial project, it is deactivated during migration. You will receive a message informing you of the change.

Use of Shared Devices

If Shared Devices are configured in the initial project, they are removal during migration. You will receive a message informing you of their removal.

Transfer areas

If transfer areas for data exchange between the IO controller and I-device are configured in the initial project, they are removed during migration. You will receive a message informing you when a transfer area is removed during migration.

Update groups for PROFINET IO

If necessary, also remove update groups from the STEP 7 project before you start the migration. If any update groups are still configured, then the migration can not be carried out.

I slaves and I-devices

Only connected and consistent I slave and I-device configurations can be migrated. If the configuration is inconsistent, then the configurations in question will be excluded from the migration.

PROFINET IO systems that are not connected

Devices with PROFINET IO systems that are not connected can be migrated. The IO devices and the PROFINET IO systems are located in the network view without connection after migration. Connect these IO devices after migration with an IO controller or delete any unnecessary IO devices.

PROFIBUS configurations

Isochronous PROFIBUS

It is possible to migrate isochronous PROFIBUS configurations. You should compile the project again after the migration, however, to rule out any possible inconsistencies. Also pay attention to slightly changed SDBs.

Orphaned master systems

Devices with orphaned DP master systems can be migrated. The DP slaves of the abandoned DP master system are located in the network view without connection after migration. Connect these DP slaves after migration with a DP master or delete any unnecessary DP slaves.

GSD and GSDML devices

For devices that were configured with PROFIBUS GSD or PROFINET GSD files, the corresponding GSD files must also be installed in the TIA Portal. If GSD files are not yet installed in the TIA Portal, they will be installed automatically during the migration. The GSD files must comply with the specifications (e.g., GSDML Specification for PROFINET IO, see www.profinet.com (http://www.profinet.com)), in order for the project to be migrated successfully. If GSD files of a device manufacturer do not comply with the specification, the project cannot be migrated. In the migration log, you receive alarms notifying you of non-compliant GSD files.

If the GSD files do not comply with specifications, check whether a newer version of the GSD file is available from the manufacturer. Install the current version. If a version of the GSD file that complies with the specification is not available, contact the device manufacturer.

Connections

Connections from SIMATIC STEP 7 projects can usually be migrated if they are also supported by the installed version of the TIA Portal.

Migratable types of connection

The following connection types can be migrated:

- S7 connection one-way
- S7 connection two-way
- Failsafe S7 connection

- TCP connection
- ISO-on-TCP connection
- ISO connection
- UDP connection
- E-mail connection
- FDL connection
- Point-to-point connection

FMS connections on the PROFIBUS are not accepted during the migration. Adapt the connections or modify the user program.

Multiprojects

Multiprojects in SIMATIC STEP 7

In SIMATIC STEP 7, you can organize projects into a multiproject if, for example, a project is very large or if several people are working on it. The subprojects may contain cross-project references, for example connections. All subprojects of a multiproject and the associated libraries are stored in the same directory.

Migration of parts of a multiproject

You can migrate the subprojects of a multiproject. To do this, select one of the subprojects during the migration. All of the devices and configurations contained in the respective subproject will be imported during the migration.

Observe the following when migrating a subproject:

- Cross-project connections
  Cross-project connections between subprojects are created as unspecified connections.
- Cross-project networking
  Information on networking of devices across project boundaries will be lost during the migration. Links between devices in the same subproject will be retained however.

Grouping a multiproject for the migration.

It is only possible to migrate an entire multiproject with all of its associated subprojects if you manually regroup the different subprojects prior to the migration. This is possible, for example, by copying the individual devices and pasting them back into a single project.

Messages

PLC alarms are migrated with all parameters and including all alarm classes. Alarm types and alarm instances are also migrated and stored under the PLC alarms in the TIA Portal.

Requirement

Requirement is that the CPU-wide assignment of alarm numbers is selected in the initial project. Before the migration, select the "Always assign CPU-oriented unique message numbers" option in the initial project, if necessary, and perform "Save As > With Reorganization".

Restrictions

- Symbol-based alarms cannot be migrated.
- The keywords "$$AKZ$$", "$$BlockComment$$", and "$$Area$$" are transferred but no replacement takes place during runtime.

Reporting system errors

In principle, you can migrate system error messages that you have configured using SIMATIC STEP 7.

Restrictions

When migrating the "Report system errors" application, only the settings that were set by the user in the SIMATIC STEP 7 project will be migrated (e.g. basic settings, block numbers, block names).

The following elements are not migrated:

- SFM blocks
- SFM messages and the settings that were made to configure the message texts
- System text libraries

After migration, the settings can be manually adjusted once again. The missing elements will be recreated during the first compilation of the hardware configuration.

Migrating program blocks

General notes on migrating program blocks

You can generally migrate all blocks created in LAD, FBD, SCL, STL, and GRAPH. Please note the following:

Programs with absolute addressing

In the TIA Portal, symbolic operands are automatically declared for all absolute addresses. If absolute addressing was used in the STEP 7 program to be migrated, symbolic operands are also declared for the absolute addresses during the migration. This automatic declaration can lead to data type conflicts, particularly if the IEC check is activated in the TIA Portal. A data type conflict could occur, for example, if a 32-bit data value is declared as a DWORD type tag but a REAL data type tag is expected later in the program.

In this case, it is not enough to deactivate the IEC check. Instead, you must correct the declaration in the PLC tag table.

Migration of communication blocks

During the migration of communication instructions from STEP 7 V5.x, pointers of the data type ANY are replaced by PLC tags within the own S7 station.

- After the migration a warning is displayed at the affected parameters.
- Replace the PLC tags with a pointer at the affected parameters.
  For absolute addressing of a data block with data type ANY, the data block must have been created with standard access, since symbolic addressing with data type ANY is not possible.
  Example: To input values, use the appropriate parameters of instruction P#M10.0 BYTE 4 instead of %MD10.

Symbolic names for blocks and user-defined data types (UDT)

In the TIA Portal, each block or UDT has a number and a name. It is therefore no longer necessary to declare symbolic names. If the migrated program contains symbol declarations for blocks or UDTs, they will be used as names after the migration.

Blocks from STEP 7 libraries

In the TIA Portal, standard functions and function blocks are not provided in libraries. Instead, they are available as instructions in the "Instructions" task card. The instructions are sorted by function and can be found under their symbolic name.

If calls of standard functions or function blocks are contained in the migrated program, they are replaced during the migration by the instructions that correspond to the original standard function or the original standard function block.

If the library block is no longer supported in the TIA Portal, you have the following options:

- Replace the library block with a compatible instruction. In this case, you will receive an alarm informing you that you have to compile the program after migration. Note that the calling block may not carry any know-how protection, as the block call has to be adapted during compilation.
- If there is no compatible instruction, the block is migrated as user block. It is then available as know-how protected block in the folder "Program blocks" with the extension "_LF" (legacy function).

User-defined block libraries

User-defined block libraries are no longer available in the familiar form in the TIA Portal. However, you can migrate user-defined libraries by integrating these into a project prior to migration and then migrating the project. After this, you can copy these from this project in the TIA Portal and use them.

Instructions in a modified display

The display of some of the LAD, FBD, STL, SCL and GRAPH commands in the TIA Portal differs from the display in earlier versions of STEP 7. Thus, for example, the mathematical functions and the comparators are no longer specific for data types. Instead, there is a central instruction, which can be used for all types of data. The "ADD_I" command is no longer permitted, for example; the instruction "ADD" is used instead.

Some other commands also have a new display in the TIA Portal, e.g., edge commands, word logic operations, conversions, IEC timers, IEC counters, etc.

The commands will be converted for the new display during the migration.

Stricter IEC check

Stricter rules for checking data type compatibility will be used in the TIA Portal. In addition, errors that can lead to runtime errors are already detected during the syntax check. It is no longer permitted to write your own input parameters or to read your own output parameters in functions (FC). Observe the following rules to avoid syntax errors:

- Only use tags with compatible data types when transferring the parameters.
- In comparison instructions or arithmetic instructions, only use tags with compatible data types.
- You may not write to input parameters or read from output parameters.

For more information on the block interface and on data type conversion with IEC check, refer to "See also".

Peripheral access ID ":P"

In the TIA Portal, the access ID ":P" is used for direct addressing of the peripherals. The following notation is not permitted, for example:

```
%PIW3
```

The following notation is used instead:

```
%IW3:P //absolute display
MyTag:P //symbolic display
```

The accesses are converted into the new notation during the migration. However, symbolic names, which were declared to be peripheral tags in the original program, cannot be accepted. New declarations will be created instead. For additional information on peripheral access, refer to "See also".

No distinction is made between upper and lower case for jump marks

No distinction is made between upper and lower case for jump marks in the TIA Portal. If the source program contains jump labels that differ only through upper/lower case, these will be converted into unique jump labels. The log file provides information on the modified jump labels.

German special character "ß" is not distinguished from "ss"

In tag names, the German special character "ß" is not distinguished from "ss" in the TIA Portal. If, for example, the source program contains tag names that differ only in these two characters, they will be converted into unique tag names.

Migration of global data blocks

The display and handling of data values in data blocks is different in the TIA Portal than in SIMATIC STEP 7. In global data blocks which are not derived from a high-level object, for example a UDT, the tags are always assigned the default value of the data type as the default value, for example FALSE for the data type BOOL. This default value is not editable. If offline initial values were assigned in the declaration view, they are not imported during the migration. If you require user-specific default values in the program, use a global data block based on a PLC data type. You can assign user-specific default values for tag values in a PLC data type.

For additional information on data values, refer to "See also".

See also

Programming data blocks (Page 2825)

Accessing I/O devices (Page 2210)

General rules for declaring the block interface (Page 2547)

Data types (Page 2236)

Migration of LAD/FBD/STL programs

LAD, FBD and STL blocks can run after the migration without being recompiled or reloaded.

Know-how protection

Blocks that were know-how protected before the migration remain know-how protected after the migration. Because the sources are not migrated, the know-how protection cannot be canceled. The blocks can thus not be opened or edited. It is possible, however, to load the blocks into the CPU and to run them.

The following options are available for disabling the a block's know-how protection:

- Prior to the migration, remove the KNOW_HOW_PROTECT attribute from the source and create a block without know-how protection from this. Then migrate this block.
- Import the source into the TIA Portal and create a block without know-how protection from this. Then use the "Edit > Know-how protection" menu command to provide the block with a password.

Switch-over from LAD/FBD to STL

As determined by the system, programs that were created in LAD or FBD and contain jump operations to subsequent networks will be displayed in STL after the migration. In the block properties, however, the language is still set as LAD or FBD.

In the block properties, change the program again to STL. Then reset the language to LAD or FBD.

Migrating GRAPH programs

Requirement

The prerequisite for migrating GRAPH blocks is to have the "S7-GRAPH" optional package, version 5.3 SP6 or later, installed on the original device.

GRAPH-specific block settings

The GRAPH-specific block settings have been significantly reduced in the TIA Portal. The following table gives an overview of how the original block settings will be transferred to the TIA Portal.

Setting in the initial project		Setting in the TIA Portal
FB parameter	Minimum	Existing parameters will be migrated
	Standard	
	Maximum	
	Definable	
Interface description	Minimized memory space	Generate DB with minimized memory space
	Structure fields	Individual structures are always created
	Individual structures	
Run capability	FC70/71	FC 73 if the option "Generate DB with minimized memory space" is active; otherwise FC 72
	FC72	
	FC73	
Criteria analysis in the DB		Setting not relevant in the TIA Portal
Skip steps		Skip steps
Acknowledgment of errors required		Acknowledge supervision errors
Synchronization		Always activated in the TIA Portal
Permanent processing of all interlocks in manual mode		Permanent processing of all interlocks in manual mode
Disable operating mode selection		Disable operating mode selection
Safeguarded switching behavior		Always activated in the TIA Portal
Warnings	None	Always activated in the TIA Portal
	Any	
Activate messages		Activate messages
Messages with WR_USMSG		Messages in the TIA Portal always with ALARM_SQ/ALARM_S

Step name extension

Step name extensions are not supported in the TIA Portal. They are converted to step-specific text during the migration.

Compiling the migrated block

GRAPH blocks must be recompiled and reloaded after the migration.

The following errors may occur when compiling:

- Data type conflicts may occur due to the stricter IEC check. In this case, change the declaration.
- Since the "structure fields" setting for the interface is no longer available in the TIA Portal, interface elements can no longer be addressed with G7S[1].X, for example. Access to the element must therefore be corrected, e.g. to STEP1.X.

Changing the interface of the GRAPH block

The changed settings can lead to interface changes of the block. Therefore, it may be necessary to regenerate the instance DB or to update block calls.

Migrating SCL programs

The prerequisite for migrating SCL blocks is to have the "S7-SCL" option package, version 5.3 SP5 or later, installed on the original device.

SCL blocks must be recompiled and reloaded after the migration.

Basic procedure for the migration

A complete migration of SCL blocks is only carried out if the associated sources exist in the initial project.

The table below gives you an overview of the basic procedure.

Existing in the initial project	Existing after migration
SCL blocks with source	Editable SCL blocks
Know-how protected SCL blocks with source	Editable SCL blocks. After the migration, the blocks no longer have know-how protection but can be re-protected, if necessary.
SCL blocks without source	Know-how protected SCL blocks
Know-how protected SCL blocks without source	Know-how protected SCL blocks

SCL-specific compiler options

Only the compiler options that are defined directly in an SCL source are adopted as block properties in the TIA Portal during migration. If no compiler options are defined in the original SCL source, the options in the properties of the migrated block are deactivated.

Compiler options defined as settings in the SCL Editor or in an SCL compile control file are not migrated.

The following table gives an overview of the compiler options that are transferred as block property to the TIA Portal.

Compiler option in the SCL source	Block property in the TIA Portal
scl_monitorarraylimits	Check ARRAY limits
scl_createdebuginfo	Create extended status information
scl_setokflag	Set ENO automatically

New EN/ENO mechanism

SCL uses the EN/ENO mechanism of the TIA Portal. For this reason, all usages of the OK flag are replaced by ENO during the migration. The points where ENO was used in the original program are highlighted. You must check these points after the migration and adapt them to the new mechanism.

For additional information on the EN/ENO mechanism of the TIA Portal, refer to "See also".

Operator "DIV"

The "DIV" operator is no longer available in the TIA Portal. All usages of "DIV" will be converted to the standard notation "/" during the migration.

Nested ARRAYs

In the TIA Portal it is not possible to nest ARRAYs. The following declaration is not permitted, for example:

```
ARRAY[1..5] OF ARRAY[0..3] OF INT
```

Nested ARRAYs are converted into multi-dimensional ARRAYs during migration. The example would look as follows after the migration:

```
ARRAY[1..5, 0..3] OF INT
```

Declaration of jump labels (LABEL)

The declaration of jump labels is not possible in the TIA Portal. The following declaration would not be accepted from the initial project, for example:

```
LABEL
MARKE1, MARKE2, MARKE3 ;
END_LABEL
```

Set jump labels, however, will be retained in the program code and can be used for GOTO operations.

Declaration of symbolic constants in the block interface

In the TIA Portal, symbolic constants always have a data type. If symbolic constants are declared without data type in the source program, an appropriate data type will be assigned to them during the migration.

Indexed peripheral accesses

In the TIA Portal, the syntax with rounded parentheses is used for indexed peripheral accesses. The following notation is not permitted, for example:
`PEB[1]`

The following notation is used instead:
`EB(1):P`

The accesses are converted into the new notation during the migration. For additional information on peripheral access, refer to "See also".

Indexed memory accesses

In the TIA Portal, the syntax with rounded parentheses is used for indexed memory accesses. The following notation is not permitted, for example:
`EB[2]`

The following notation is used instead:
`EB(2)`

The accesses are converted into the new notation during the migration. For additional information on addressing, refer to "See also".

Indexed access to data blocks

In the TIA Portal, the syntax with rounded parentheses is used for indexed data block accesses. The following notation is not permitted, for example:
`%DB100.DW[5]`

The following notation is used instead:
`%DB100.DW(5)`

The accesses are converted into the new notation during the migration. For additional information on addressing, refer to "See also".

Absolute access to data blocks

For an absolute access, the absolute designator of the data block must be used. Access using the symbolic designator is not permitted in the TIA Portal.

The following notation is not permitted, for example:
`DB100.DW3`

The following notation is used instead:
`%DB100.DW3`

During the migration, the marker "%" will be added for a detected absolute data block access. For additional information on addressing, refer to "See also".

Logarithmic functions "EXPD" and "LOG"

Logarithmic functions "EXPD" and "LOG" are no longer available in the TIA Portal. All usages of "EXPD" are converted to the standard notation "10**(<expression>)" during the migration. All usages of "LOG" are converted to the standard notation "LN(<expression>)" during the migration.

Mathematical instruction "MAX"

The "MAX" instruction cannot process any time data types in the TIA Portal. However, you can use comparators instead.

The following example shows how you can find the "larger" of two time data types using comparators.

SCL

```
#xs_tod := #xi_tod + #xi_time;
   // statt #xstod1 := MAX(IN1:=#xtod1, IN2:=#xstod1);
   IF #xs_tod < #xi_tod THEN
         #xs_tod := #xi_tod;
   END_IF;

   // statt #xsd1 := MAX(IN1:=#xd1, IN2:=#xsd1);
   IF #xi_date <#xs_date THEN
         #xs_date := #xi_date;
   END_IF;
#xs_date_time := #xi_date_time + #xi_time;
   // statt #xs_date_time := MAX(IN1:=#xi_date_time, IN2:=#xs_date_time);
   IF #xi_date_time <#xs_date_time THEN
          #xs_date_time := #xi_date_time;
   END_IF;
```

Floating point numbers - exponential display

The exponential notation of floating point numbers without a decimal point is no longer permitted in the TIA Portal. The following notation is not permitted, for example:
`3E10`

The following notation is used instead:
`3.0E10`

During the migration, all usages of "3E10" will be converted to the standard notation "3.0E10". For additional information on floating point numbers, refer to "See also".

Blocks from STEP 7 libraries

The functions FC 1 to FC 40 from the STEP 7 classic library "IEC standard functions" are no longer available in the TIA Portal. Calls to the standard functions are converted to calls to the

corresponding extended instructions during the migration. If there is no clear conversion possibility, a system error is issued.

In this case, change the SCL block after the migration and use SCL instructions or operators instead of IEC standard functions.

Examples:

Use S5TIME_TO_TIME instead of S5TI_TIM (FC 33)

```
"TAG_TIME" := S5TIME_TO_TIME ("TAG_S5TIME");
```

Use the comparator operator <> instead of NE_DT (FC 9):

```
IF #D1 <> #D2 THEN "MyOutput" : = 10;
END_IF;
```

Use the comparator operator = instead of EQ_STRING (FC 10):

```
IF #My_String1 = #MyString2 THEN "MyOutput" : = 10;
END_IF;
```

Interruption of strings with "$"

In STEP 7 Classic, strings in SCL could be interrupted by special characters, e.g., control characters or non-printable characters. These characters were identified by the interruption sequences "$>" and "$<".

Example:

```
#myString := 'Test1$>     $<Test1';
```

This option no longer exists in the TIA Portal. String interruptions will be deleted during the migration.

NULL pointer

Null pointers are no longer specified in the TIA Portal with "NIL" but instead with "NULL". The notation is replaced automatically during the migration.

Comments

Comments from the migrated source are transferred when possible. The following rules apply:

- Comments in the output block directly after the block definition are converted as block comments.
- Comments that appear as line comments after a tag declaration are transferred to the "Comment" column in the block interface.
- All other comments are not transferred.

Example:

```
// Comment that is not transferred.
FUNCTION_BLOCK FB57
// Comment that is transferred to the block properties.
(*Comment that is transferred to the block properties. *)
VAR_TEMP
    myVar1 : INT;// Comment that is transferred to the interface.
```

```
    myVar2 : INT;(* Comment that is not transferred because it is a
comment segment. *)
END_VAR
// Comment that is not transferred.
BEGIN
    ;
END_FUNCTION_BLOCK
```

See also

Declaring global constants (Page 2488)

Basics of constants (Page 2204)

EN/ENO mechanism (Page 2447)

Accessing I/O devices (Page 2210)

Floating-point numbers (Page 2252)

Addressing operands (Page 2209)

4.1.7 Migrating WinCC flexible projects (Basic)

4.1.7.1 Basics (WinCC flexible)

Migration (WinCC flexible)

Introduction

You can continue to use projects in WinCC from WinCC flexible. The following version of WinCC flexible is supported:

- WinCC flexible 2008 SP2

The following sections describe the operating devices that are supported and the required conditions for a successful migration.

Projects from ProTool and from earlier versions of WinCC flexible cannot be migrated directly to WinCC. If you wish to continue to use such projects in WinCC, you must first migrate them to a supported version of WinCC flexible.

See also

Object support during migration (WinCC flexible) (Page 148)

Projects from Migrating WinCC flexible projects (WinCC flexible) (Page 144)

Compiling and loading a migrated project (WinCC flexible) (Page 146)

Migrating runtime data (WinCC flexible) (Page 158)

Migrating integrated projects (WinCC flexible) (Page 160)

Supported HMI devices (WinCC flexible) (Page 147)

Migration of data types (WinCC flexible) (Page 164)

Basics on migration (WinCC flexible)

Introduction

During migration, the project data from a WinCC flexible project are converted to the new format of WinCC. The data will not be evaluated to see if they are consistent in the project you want to migrate. If errors or warnings are output in a source project during compilation, these will not be resolved as part of the migration. This means you should be able to compile the project without errors prior to migration. Note the scope of a project during migration. The features of WinCC apply for migration. For more information on this topic, refer to the "Visualize Processes > Performance Features > Engineering System" chapter in the online help.

Unique object names

The objects are clearly identified by the folders in which they are contained in WinCC flexible. Screen elements in groups are clearly identified by the group name.

In WinCC, an object name must be unique within an HMI device. The name of screen elements must be unique within a screen.

The uniqueness of the name is verified during migration. If a name is not unique according to the new rule, the object in question will be renamed. A renamed object will receive the suffix "#Mign", where "n" stands for a sequential number.

Example:

In WinCC flexible, tags located in different folders may have the same name. In WinCC, the tag name must be unique on the configured HMI device. This means tags with the same name from different folders will be renamed during migration.

Tags are renamed as follows:

Before migration	After migration
Folder_1/Tag_1	Folder_1/Tag_1
Folder_1/Tag_2	Folder_1/Tag_2
Folder_2/Tag_1	Folder_2/Tag_1#Mig1
Folder_2/Tag_2	Folder_2/Tag_2#Mig1
Folder_3/Tag_1	Folder_3/Tag_1#Mig2
Folder_3/Tag_2	Folder_3/Tag_2#Mig2

Affected objects

The following objects are renamed if necessary:

- Screens
- Screen objects
- Recipes
- Tags

Cancelling migration

The migration is cancelled in the following cases:

- If the project to be migrated is opened in the engineering system or in Runtime.
- If not enough memory space is available on the hard disk to create a a copy for migration of the project.
- If the migration cannot address the project database due to problems with the installed SQL-Server.
- If the migration cannot address the project database due to missing user authorization.
- If you select the "*.hmi" file for the migration in an integrated project. You must select the "*.s7" file for the migration in an integrated project.
- If the project was created with a version not supported by the migration.

Saving the project in the migration format

You do not have to execute the migration of a WinCC flexible project completely on the PC on which the project is available. You can prepare the migration by saving the project in the migration format. The migration tool is available for saving a WinCC flexible project in the migration format. The migration tool exports the engineering data from the WinCC flexible project and saves the data in the migration format "*.AM11".

For the actual migration, copy the data in the migration format to a PC on which the TIA-Portal is installed.

See the section "Auto-Hotspot" for more information on the migration tool.

Projects from Migrating WinCC flexible projects (WinCC flexible)

Introduction

When you migrate a project, data from a WinCC flexible project is loaded into a new project for WinCC. A new project is therefore created automatically for project migration. You cannot migrate to an existing project.

The migration can be started in both the Portal view and the Project view.

You should only migrate a project in a newly started TIA portal.

Information on the migration of an integrated project can be found in the section Migrating integrated projects (WinCC flexible) (Page 160).

If you only want to save the project in migration format, you can use the migration tool. See Basics on migration (WinCC flexible) for additional information.

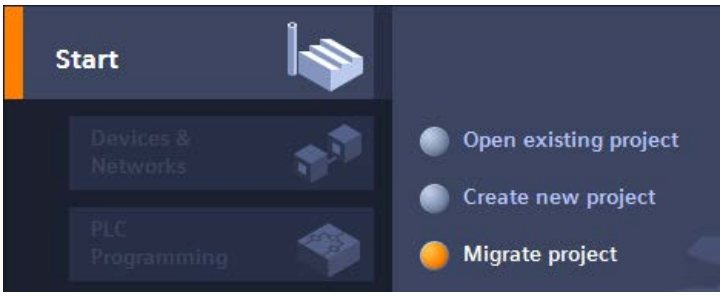
Requirement

- A project from WinCC flexible is available.
- The project is not open in WinCC flexible.

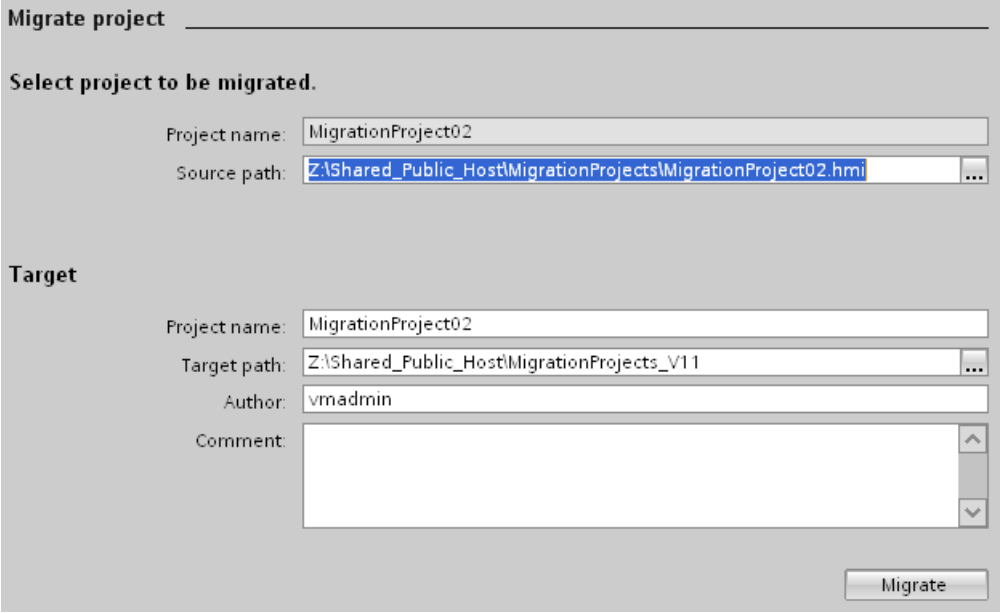
Procedure

Migrate a project in the Portal view as follows:

1. Select the action "Start > Migrate Project".



2. In the "Source path" box, navigate to the project you want to migrate.



3. Select the WinCC flexible project file "*.hmi".

4. Change the information for the project to be created, if necessary. For example, change the project name or project path. The data to be migrated is created in the new project.
5. Click "Migrate".
   A new project is created and migration of the data is started:
   - The Project view opens.
   - The progress of the migration is shown in a migration window.
   - Warnings and errors about the migration process are displayed in the Inspector window under "Info > General".
   - All information about the migration is saved in a log file.
   - The project is saved and a message displayed upon completion of the migration. The message contains a link that you can use to open the log.

When migration is complete, you will find a newly created device for each migrated HMI device in the project tree. These devices contain the migrated data, such as screens, alarms and tags.

Opening the migration log at a later point in time

The migration log is saved together with the migrated project. You can view the log at a later point in time. Open the log as follows:

1. Open the "Common data > Logs" folder in the project tree. It contains the logs of all previously performed migrations.
2. Double-click the required migration log.
   The log is opened.

See also

Migrating integrated projects (WinCC flexible) (Page 160)

Compiling and loading a migrated project (WinCC flexible)

Compiling a migrated project

Once you have successfully migrated a WinCC flexible project, you need to recompile it before loading it to the HMI device. The project will only compile successfully if it was capable of error-free compiling prior to migration.

If errors occur during compilation of the migrated project, they have to be eliminated.

Once compiling is successfully completed, load the project to the HMI device.

Settings for download to the HMI device

The settings for loading the HMI device are not included in the migration. Once you have migrated the project, you must configure the settings for loading.

Select the HMI device in the project tree and select "Loading in device > Software (complete loading)" from the shortcut menu.The dialog "Advanced Loading" is opened. Configure the

required settings for the interface. Click the "Load" button. The project is recompiled and the dialog "Load preview" is opened.

Expand the "Overwrite" entry and verify the settings for the following options:

- Would you like to overwrite the existing user administration data from this device
- Would you like to overwrite the existing recipe data on HMI system

Configure the options as you want to use them in the project in the future. Subsequently, load the project to the HMI device.

4.1.7.2 Migrating engineering data (WinCC flexible)

Supported HMI devices (WinCC flexible)

Introduction

Note that WinCC only supports the following HMI device types when migrating projects from WinCC flexible:

- KTP400 Basic mono PN
- KTP400 Basic mono PN Portrait
- KTP600 Basic DP
- KTP600 Basic DP Portrait
- KTP600 Basic PN
- KTP600 Basic PN Portrait
- KTP600 Basic mono PN
- KTP600 Basic mono PN Portrait
- KTP1000 Basic DP
- KTP1000 Basic PN
- TP1500 Basic PN

WinCC only supports the functions provided by these HMI device types.

If your WinCC flexible project contains an HMI device that is not supported by WinCC, then the migration process will abort. To migrate the project, you must change the HMI device in WinCC flexible to a HMI device type supported by WinCC.

There may be some functions in a WinCC flexible project that are not supported by a Basic Panel, for example, because the device type has been switched. These unsupported functions are not migrated.

Adaptations before migration

If the HMI device has changed in the project being migrated, the project needs to be recompiled before migration. The compilation process will adjust the size of the screens and screen elements.

See also

Object support during migration (WinCC flexible) (Page 148)
Migration (WinCC flexible) (Page 142)
Migration of alarm classes and alarm groups (WinCC flexible) (Page 152)
Migration of language-dependent contents (WinCC flexible) (Page 154)
Migrating libraries (WinCC flexible) (Page 157)
Migration tags (WinCC flexible) (Page 151)
Changes of values of object properties by the migration (WinCC flexible) (Page 150)

Object support during migration (WinCC flexible)

Introduction

When migrating projects from WinCC flexible, all configuration data involving an HMI device supported by WinCC will be migrated. Basically, all object types and functions that are available and can be mapped to the new project environment will be fully migrated.

Some global object types are not migrated, for example, dictionaries and global libraries.

Supported object types

The following object types are supported for migration:

- Animations
- Scheduler
- User administration
- Area pointer
- Screens
- Screen template
- Data types
- Function lists
- Graphics lists
- Display and operating elements
  Migration supports all display and operating elements available on the supported HMI devices.
- Alarms
- Alarm classes
- Alarm groups
- Project library
- Project languages

- Recipes
- Runtime languages
- Runtime scripting
- System functions
- Texts
- Text lists
- Tags
- Connections

Unsupported object types

The following object types are not supported by migration:

- Global libraries
- Dictionaries
- Project versions
- Change log

Migration of the screen template

WinCC offers an extended concept for working with screen templates. WinCC offers a global screen and several templates for each device. During migration of a template from WinCC flexible, the objects contained there and the properties configured in the template are migrated to different templates of WinCC.

The following objects are migrated to the "global screen" of WinCC:

- Alarm window
- Alarm indicator
- Function keys of HMI devices with function keys

All other objects and properties are migrated to a template of WinCC.

The connection of the objects and properties to the respective template is automatically adapted.

Migration of system functions

The names of some system functions have changed in WinCC.

System functions which have changed their names are renamed.

This concerns the following system functions:

Function name in WinCC flexible	Function name in WinCC
IncreaseValue	IncreaseTag
DecreaseValue	DecreaseTag
SetValue	SetTag

See also

Supported HMI devices (WinCC flexible) (Page 147)

Changes of values of object properties by the migration (WinCC flexible) (Page 150)

Changes of values of object properties by the migration (WinCC flexible)

Introduction

The standardization of object properties from WinCC V7 and WinCC flexible requires changes to the object properties during the migration process. The migration calculates the changes in such a way that the representation of the objects after migration is the same as prior to migration. Changes made during migration result in different units of measurements and values in the configuration for some object properties.

Migrating the font settings of an object

In WinCC V7 and WinCC flexible, the unit of measurement "point" is used to denote the size of the fonts used for an object. In WinCC, the unit of measurement "pixel" is used to denote the size of the fonts used for an object. During migration, the font size is converted accordingly to ensure that the representation of the font is the same size at zoom level 100%. The different units of measurement result in changes to the numerical values for the font sizes after migration.

Example:

Font style before migration	Font style after migration
Arial 10 points	Arial 13 pixels
Arial 16 points	Arial 21 pixels
Tahoma 10 points	Tahoma 13 pixels
Tahoma 16 points	Tahoma 21 pixels

Migration of object margins

In WinCC flexible, some objects permit the entry of values <0 and >127 for setup of the object margins for the configuration of the representation. In WinCC, the range of values for object margins is limited to values between 0 and 127. The migration changes values <0 to the value "0" and values >127 to the value "127".

See also

Supported HMI devices (WinCC flexible) (Page 147)

Object support during migration (WinCC flexible) (Page 148)

Migration tags (WinCC flexible)

Introduction

You need to make some special considerations when migrating tags. The following aspects should be distinguished:

- Migrating data types of tags
- Migrating internal tags
- Migrating external tags
- Tag names

Migrating data types

WinCC features some other data types and uses different data type names than WinCC flexible. When migrating a relevant tag, the data type from WinCC flexible is mapped to the corresponding data type in WinCC. You can find additional details on this in the section "Migration of data types (WinCC flexible) (Page 164)".

Migrating tags

Tags are always fully migrated. Only the data type names and tag names may change due to migration.

Migrating names of tags

In WinCC flexible, tags located in different folders can have the same name. In WinCC, the tag name must be unique on the configured HMI device. This means tags with the same name from different folders will be renamed during migration. You can find additional details on this in the section "Basics on migration (WinCC flexible) (Page 143)".

See also

Basics on migration (WinCC flexible) (Page 143)

Migration of data types (WinCC flexible) (Page 164)

Supported HMI devices (WinCC flexible) (Page 147)

Migration of alarm classes and alarm groups (WinCC flexible)

Changing the names of alarm classes

In contrast to WinCC flexible, the names of the predefined alarm classes are not dependent on the user interface language currently in use. During migration, the names of the alarm classes are assigned as follows:

WinCC flexible	WinCC
Error	Alarms
System	System
Warnings	Events

The names of the alarm classes can be changed as necessary after migration.

Migrating alarm groups

Migration will migrate only those alarm groups actually in use.

Alarm groups with an ID from 1-31 will be migrated 1:1.

A corresponding alarm group is created in WinCC for each alarm class in the system. These alarm groups created by the system are assigned IDs beginning with the number 32 and consecutively incremented. The 4 pre-defined message classes in every WinCC project are automatically given IDs 32-35 by their alarm groups. Additionally created alarm group and an additional ID is assigned to each user-defined alarm class. Therefore, the IDs for alarms groups with IDs > 31 may be changed after migration. This step also changes the assignment of the alarm group names to the IDs.

Example:

In the example, you can see the assignment of the IDs in WinCC for the migration.

Alarm groups	ID in WinCC flexible	ID in WinCC	
Alarm group 1-16	1-16	1-16	Default for alarm groups from system alarms
Alarm group 17-31	17-31	17-31	Custom alarm groups
		32-35	Default in WinCC for alarm groups of predefined alarm classes.
Alarm group 32	32	36	Changed assignment of ID to alarm group in WinCC
Alarm group 33	33	37	Changed assignment of ID to alarm group in WinCC

Also note:

When migrating alarm groups that supposedly have the same group name, the migration adapts the name. This occurs, for example, when a group name contains a space at the end of the name. The migration deletes all existing spaces at the end of names. If two groups obtained the same group names due to this deletion, the migration adds the suffix "# Mign" to the group name of the following alarm groups, where "n" stands for a sequential number.

Example:

The following alarm groups exist in WinCC flexible:

"AlarmGroup_18"

"AlarmGroup_18 " - group name contains one space

"AlarmGroup_18 " - group name contains two spaces

"AlarmGroup_18" is the alarm group with the highest number.

Result after migration:

"AlarmGroup_18"

"AlarmGroup_18#Mig1"

"AlarmGroup_18#Mig1.1"

Changing the names of alarm classes

In contrast to WinCC flexible, the names of the predefined alarm classes are not dependent on the user interface language currently in use. During migration, the names of the alarm classes are assigned as follows:

WinCC flexible	WinCC
Error	Errors
System	System
Warnings	Warnings

The names of the alarm classes can be changed as necessary after migration.

Display of ALARM_S messages and SIMATIC SFM messages

In WinCC flexible you can activate the display classes for ALARM_S messages in integrated projects. In WinCC flexible, you activate the display of SIMATIC SFM messages via a separate setting. The separate setting for activating the display of SIMATIC SFM messages is not required in WinCC. You control the display of SIMATIC SFM messages, and also the display of ALARM_S messages in WinCC only by activating the corresponding display class.

The changed concept may cause the display of messages to change following migration.

If all the display classes for ALARM_S messages are activated and the display of SIMATIC SFM messages is deactivated in the WinCC flexible project , ALARM_S messages and SIMATIC SFM messages are displayed following migration.

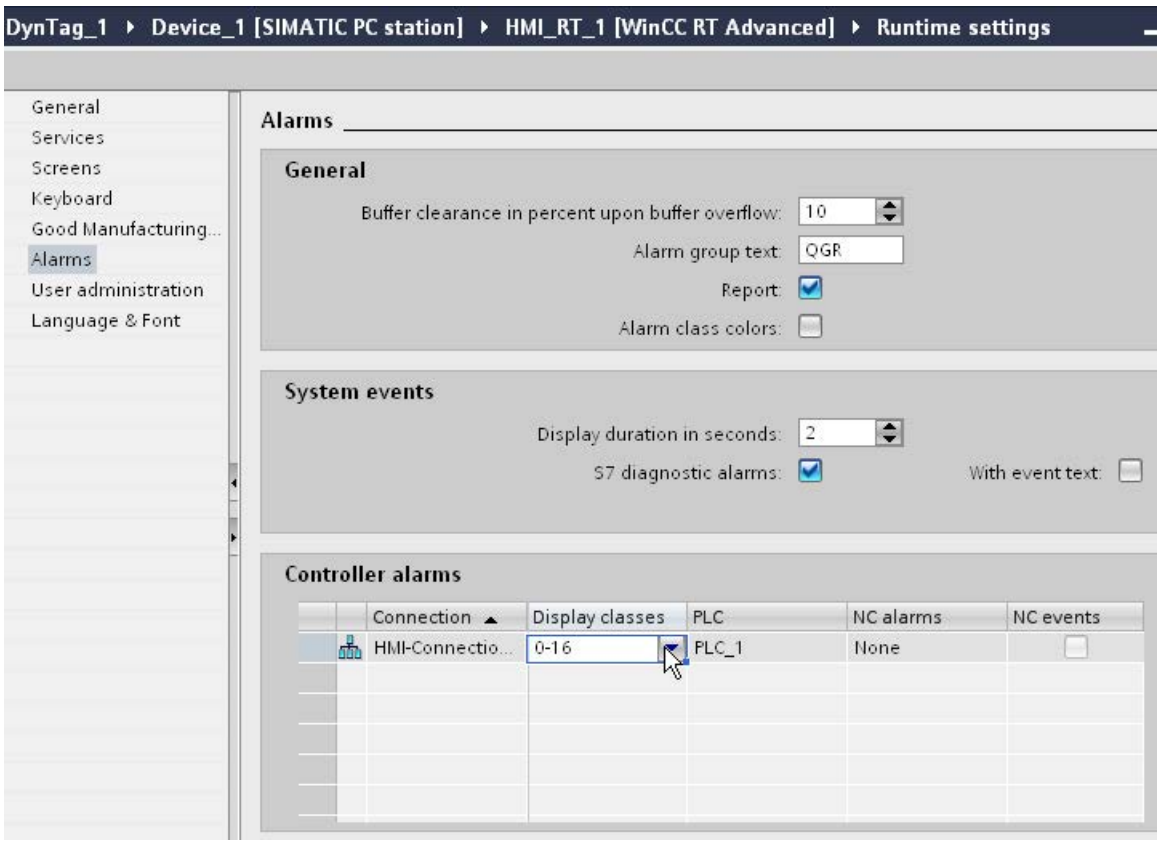
To ensure that only ALARM_S messages are displayed following migration, you have to assign the SIMATIC SFM messages to an unused display class after migration to STEP 7. You then have to deactivate this display class in WinCC.

If all the display classes for ALARM_S messages are deactivated and the display of SIMATIC SFM messages is activated in the WinCC flexible project , ALARM_S messages and SIMATIC SFM messages are not displayed following migration.

To ensure that only SIMATIC SFM messages are displayed following migration, you have to assign the SIMATIC SFM messages to an unused display class after migration to STEP 7. You then have to activate this display class in WinCC.

The display class is dependent on the settings in STEP 7. The default setting for SIMATIC SFM messages in Step 7 is the display class "0". To activate the display in WinCC, the display class "0" must be activated.

You activate the display class in WinCC in the Runtime settings of the respective HMI device in the "Messages" category.



See also

Supported HMI devices (WinCC flexible) (Page 147)

Migration of language-dependent contents (WinCC flexible)

Introduction

WinCC offers the same options for configuring projects in different languages as those available in WinCC flexible. All languages supported by WinCC are included in the migration of a project.

Migrating language-dependent content

The following language-dependent content is migrated:

- Project languages
- Project texts
- Fonts for display in runtime
- Language-dependent graphics

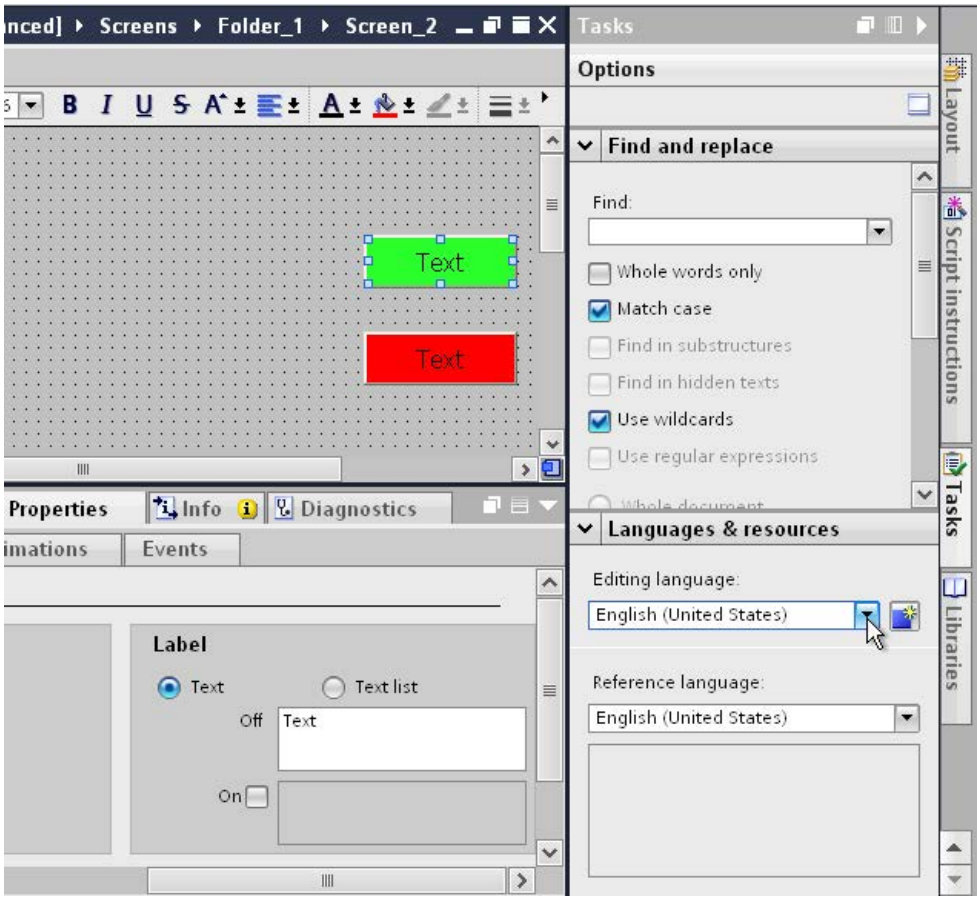
You need to consider the following when migrating language-dependent content:

- The operating system on the PC performing the migration must support the project languages used in the project.
- The fonts used for runtime display must be installed on the PC performing the migration.
- Dictionaries are not supported by the migration.

Editiing language of integrated projects following migration

During migration of an integrated project, the project components to be migrated from STEP 7 and WinCC flexible also bring their respective settings for the editing language. In WinCC there is only one editing language for all project components. Migration activates for the mgrated project the editing language which was set in STEP 7 prior to migration. If this setting is not the same as the setting from WinCC flexible, the configured texts are no longer visible in WinCC. No text is displayed at the usage locations, or only the entry "Text" can be seen. To make the texts visible, you must change the editing language. Click the "Tasks" taskcard at

the right-hand edge of the TIA portal and select the correct editing language in the "Language & Resources" area.



Unsupported languages

The migration of language-dependent content depends on whether or not WinCC supports the respective language.

If a project only contains project languages not supported by WinCC, the project will not be migrated.

If a project contains supported and unsupported project languages, only the supported languages will be migrated. The editing language and reference language are set to a supported language.

The following languages are not supported by WinCC:

- Arabic
- Hebrew
- Dhivehi
- Gujarati
- Kannada

- Tamil
- Telugu
- Urdu
- Punjabi
- Persian
- Syrian

See also

Supported HMI devices (WinCC flexible) (Page 147)

Migrating libraries (WinCC flexible)

Introduction

You need to consider two different cases when migrating from libraries:

1. Migrating a project library
2. Migrating a global library

Migrating a project library

A project library is stored together with the project data in the project file. For this reason, a project library is migrated with the same restrictions as the project data.

Migrating a global library

Global libraries are not supported by the migration. The library objects used in the project will be migrated, however. The library objects are copied when used in the project and then no longer have a connection to the library.

To migrate a global library, you must copy or move the objects contained in the library to the project library. The objects are then included in the migration. In WinCC, you move the migrated objects to a new global library that is created. You can copy or move both individual objects or entire library categories.

See also

Supported HMI devices (WinCC flexible) (Page 147)

4.1.7.3 Migrating runtime data (WinCC flexible)

Migrating runtime data (WinCC flexible)

Introduction

When migrating a project, only the configuration data will be migrated. The runtime data are not affected. You need to update the runtime data following migration.

The runtime data consists of the following:

- Runtime project
  The runtime project contains the compiled project data.
- Recipe data and user administration
  The recipe data and user administration are data that can be changed in runtime.

Migrating runtime data

You update the runtime project by compiling the project in WinCC again and loading it to the HMI device.

If the recipe data and user administration were changed in runtime, you need to back up this information from the HMI device before you load the migrated project. You can then load the migrated project to the HMI device. Finally, you load the saved recipe data and user administration back to the HMI device. You can find additional details on this in the section "Auto-Hotspot".

See also

Migration (WinCC flexible) (Page 142)

Backing up recipe data and user administration (WinCC flexible) (Page 158)

Restoring recipe data and user administration (WinCC flexible) (Page 159)

Backing up recipe data and user administration (WinCC flexible)

Introduction

To continue using the recipe data and user administration in a migrated project, you first need to back up this data from the HMI device. Then load the data into the migrated WinCC project. Use ProSave to back up the data.

Requirement

- The WinCC flexible project is running on the HMI device in Runtime.
- The HMI device is connected to a PC on which ProSave is installed.

Procedure

Proceed as follows to back up the recipe data and user administration:

1. Start ProSave.
2. Select the device type and the connection parameters in the "General" tab.
3. Open the "Backup" tab.
4. Select the "Recipes from the device memory" entry in the "Data type" box.
   Do not select "Complete backup" because otherwise you will not be able to select separately when restoring the recipe data.
5. Navigate to the desired location in the "Save as" box and click "Start Backup".
   The recipe data are saved.
6. Select "User administration" in the "Data type" box and click "Start Backup".
   The user administration is saved.

For additional information refer to the online help for ProSave.

Alternative procedure

ProSave is automatically installed with WinCC flexible. The entire functional range of ProSave is available on the configuration PC within WinCC flexible via the menu command "Project > Transfer".

Alternatively, you can back up the recipe data and user administration via the ProSave integrated in WinCC flexible. Start WinCC flexible and select the menu command "Project > Transfer > Backup". Back up the recipe data and user administration as described in steps 4-6.

See also

Migrating runtime data (WinCC flexible) (Page 158)

Restoring recipe data and user administration (WinCC flexible) (Page 159)

Restoring recipe data and user administration (WinCC flexible)

Introduction

To continue using saved recipe data and user administration after the migration, you first need to compile the migrated project and load it to the HMI device. You can then transfer the saved data to the HMI device. Use ProSave to restore the data.

Requirement

- The migrated project has been transferred to the HMI device and is running in runtime.
- The HMI device is connected to a PC on which ProSave is installed.

Procedure

Proceed as follows to load the saved recipe data and user administration to the HMI device:

1. Start ProSave.
2. Select the device type and the connection parameters in the "General" tab.
3. Open the "Restore" tab.
4. Navigate to the location of the saved recipe data in the "Opening..." box and select the file.
5. Click "Start Restore".
   The recipe data will be transferred to the HMI device..
6. Repeat steps 4-5 to restore the user administration.
   The user administration will be transferred to the HMI device.

For additional information refer to the online help for ProSave.

Alternative procedure

ProSave is automatically installed with WinCC. The entire functional range of ProSave is available on the configuration PC within WinCC flexible via the menu command "Project > Transfer".

You can also restore the recipe data and user administration via the ProSave integrated in WinCC. Start WinCC and select the menu command "Online > Device maintenance > Restore". Restore the recipe data and user administration as described in steps 4-6.

See also

Migrating runtime data (WinCC flexible) (Page 158)

Backing up recipe data and user administration (WinCC flexible) (Page 158)

4.1.7.4 Migrating integrated projects (WinCC flexible)

Migrating integrated projects (WinCC flexible)

Introduction

The controllers and HMI devices contained in a project integrated in STEP 7 are linked together by the configuration. The configuration data of WinCC flexible and STEP 7 are also connected. When an integrated project is migrated, the complete project will be migrated with components from WinCC flexible and STEP 7. The connections remain intact.

Note

It is advisable to compile and save an integrated project in WinCC flexible before you migrate it. You can be sure that the data in WinCC flexible and STEP 7 is synchronized if compilation was completed without errors.

Migrating an integrated project

When migrating an integrated project, the same requirements apply for the WinCC flexible component as those for the migration of a non-integrated WinCC flexible project. The objects and properties contained in the WinCC flexible component must be supported by WinCC, for example, the HMI device or the communication driver. The "Online" property must be activated on the configured connection. A connection with deactivated "Online" property is not migrated.

In addition to the requirements for the WinCC flexible component, there are also requirements for the STEP 7 component of the integrated project. The objects and properties contained in the STEP 7 V5.4 SP5 or V5.5 component must be supported in STEP 7. For detailed information, refer to the documentation for STEP 7.

To fully migrate an integrated project and then edit it, the following components must be installed on the PC performing the migration:

- STEP 7 V5.4 SP5 or STEP 7 V5.5
- WinCC flexible 2008 SP2 or WinCC flexible 2008 SP3
- STEP 7

If you only want to save the project in migration format, you can use the migration tool. See Basics on migration (WinCC flexible) (Page 143) for additional information.

An integrated project is always fully migrated. If you only want to migrate the WinCC flexible project it contains, you need to separate it from the STEP 7 project before the migration. To separate the project from the integrated form, open the project in STEP 7 V5.4 SP5 or V5.5. Open the WinCC flexible project in the SIMATIC Manager. The project is opened with WinCC flexible. In WinCC flexible, select the menu command "Project > Copy project from STEP 7". WinCC flexible saves a non-integrated copy of the project.

See also

Basics on migration (WinCC flexible) (Page 143)

Migrating an integrated project

Introduction

When migrating an integrated project, the components from both the WinCC flexible project and the STEP 7 project will be migrated. This means you need to select the project file with the file extension "*.s7p" for migration. During migration, the data is copied from the existing project and migrated to a new project. You cannot migrate to an existing project.

The migration can be started in both the Portal view and the Project view.

You should only migrate a project in a newly started TIA portal.

If you only want to save the project in migration format, you can use the migration tool. For more information, refer to basics of migration (WinCC flexible).

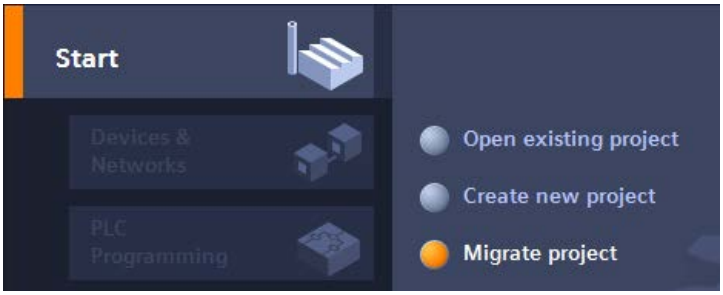
Requirement

- STEP 7 V5.4 SP5 or STEP 7 V5.5 and all add-on packages used are installed.
- STEP 7 and all add-on packages used are installed.
- The TIA portal is newly started.
- No project is open in WinCC.
- An integrated project is available.
- The integrated project is not open.

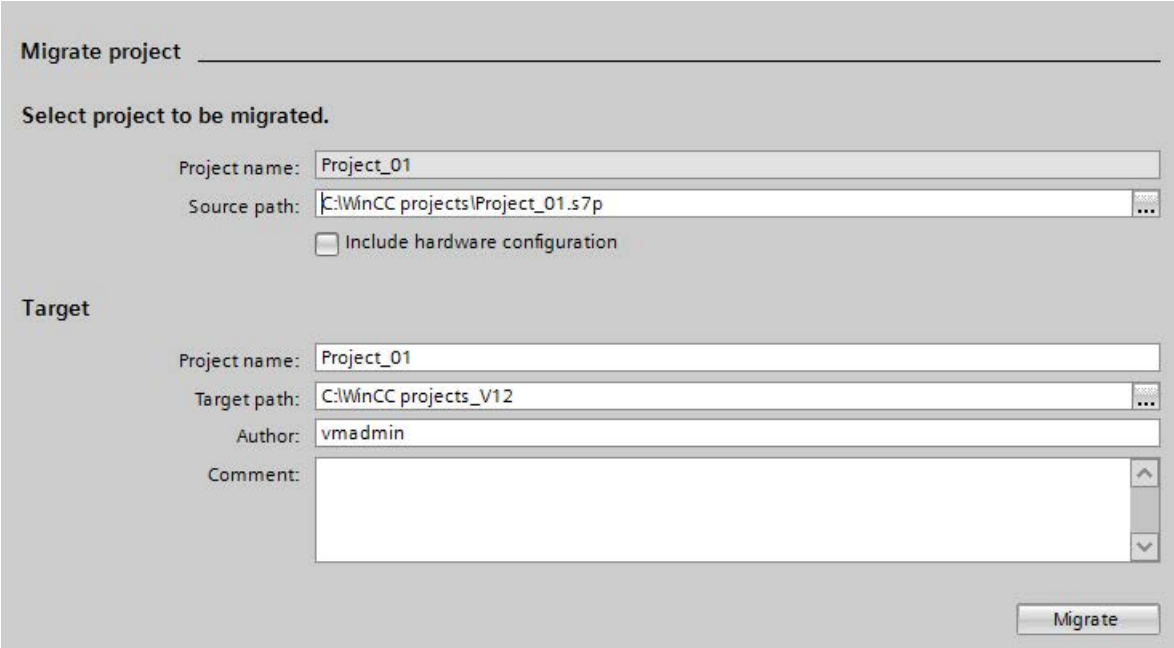
Procedure

To migrate an integrated project in the portal view, follow these steps:

1. Select the action "Start > Migrate Project".



2. In the "Source path" box, navigate to the project you want to migrate.



3. Select the "*.s7p" project file.
4. Change the information for the project to be created, if necessary. For example, change the project name or project path. The data to be migrated is created in the new project.
5. To migrate the project with hardware configuration, activate "Include hardware configuration".

6. Click "Migrate".
   A new project is created and migration of the data is started:
   - The Project view opens.
   - The progress of the migration is shown in a migration window.
   - Warnings and errors about the migration process are displayed in the Inspector window under "Info > General".
   - All information about the migration is saved in a log file.
   - A message is displayed upon completion of the migration. The message contains a link that you can use to open the log.
7. Once migration is completed, save the project.

Once the migration is complete, you will find a newly created device for each migrated HMI device and controller in the project tree. These devices include the migrated data.

Opening the migration log at a later point in time

The migration log is saved together with the migrated project. You can view the log at a later point in time. Open the log as follows:

1. Open "Shared data > Logs" in the project navigation.
2. Double-click the log file. The migration log opens.

See also

Basics on migration (WinCC flexible) (Page 143)

4.1.7.5 Reference (WinCC flexible)

Migration of data types (WinCC flexible)

Introduction

To harmonize the data types used by controllers and HMI systems, some types of internal HMI tags are renamed. The naming takes place in accordance with IEC conventions. Because only the names change, there are no changes to the internal tags for the configuration.

The following table describes the mapping of data types from WinCC flexible to the data types in WinCC.

Migration of data types

The internal data types are mapped as follows during migration:

Internal data types WinCC flexible	Internal data types WinCC
Bool	Bool
Char	SInt
Byte	USInt
Int	Int
UInt	UInt
Long	DInt
ULong	UDInt
Float	Real
Double	LReal
String	WString
DateTime	DateTime

Migrating external data types

See the following pages for how to map the available communication drivers.

See also

Migration (WinCC flexible) (Page 142)
Migrating data types of Allen-Bradley DF1 (WinCC flexible) (Page 166)
Migrating data types of Allen-Bradley Ethernet IP (WinCC flexible) (Page 166)
Migrating data types of Mitsubishi FX (WinCC flexible) (Page 167)
Migrating data types of Modicon Modbus (WinCC flexible) (Page 167)
Migrating data types of Modicon Modbus TCP/IP (WinCC flexible) (Page 168)
Migrating data types of Omron Hostlink/Multilink (WinCC flexible) (Page 168)
Migrating data types of SIMATIC S7 200 (WinCC flexible) (Page 169)
Migrating data types of SIMATIC S7 300/400 (WinCC flexible) (Page 169)

Migrating data types of Allen-Bradley DF1 (WinCC flexible)

Migrating data types Allen-Bradley DF1

The data types of the Allen-Bradley DF1 communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
ASCII	ASCII
BCD4	UInt
BCD8	UDInt
Bit	Bool
Int	Int
Long	DInt
Real	Real
UInt	UInt
ULong	UDInt

See also

Migration of data types (WinCC flexible) (Page 164)

Migrating data types of Allen-Bradley Ethernet IP (WinCC flexible)

Migrating data types Allen-Bradley Ethernet IP

The data types of the Allen-Bradley Ethernet IP communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
Bool	Bool
DInt	DInt
Int	Int
Real	Real
SInt	SInt
String	String
UDInt	UDInt
UInt	UInt
USInt	USInt

See also

Migration of data types (WinCC flexible) (Page 164)

Migrating data types of Mitsubishi FX (WinCC flexible)

Migrating data types Mitsubishi FX

The data types of the Mitsubishi FX communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
12 Bit Block	12-Bit Block
16 Bit Block	16-Bit Block
20 Bit Block	20-Bit Block
24 Bit Block	24-Bit Block
28 Bit Block	28-Bit Block
32 Bit Block	32-Bit Block
4 Bit Block	4-Bit Block
8 Bit Block	8-Bit Block
Bit	Bool
Double	DWord
IEEE-Float	Real
String	String
Word	Word

See also

Migration of data types (WinCC flexible) (Page 164)

Migrating data types of Modicon Modbus (WinCC flexible)

Migrating data types Modicon Modbus

The Modicon Modbus communication driver is not supported by WinCC, it is replaced by the Modicon Modbus RTU driver. The data types of the Modicon Modbus communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
+/-Double	+/- Double
+/-Int	+/- Int
16 Bit Group	16 Bit Group
ASCII	ASCII
Bit	Bit
Double	Double
Float	Float
Int	Int

See also

Migration of data types (WinCC flexible) (Page 164)

Migrating data types of Modicon Modbus TCP/IP (WinCC flexible)

Migrating data types Modicon Modbus TCP/IP

The data types of the Modicon Modbus TCP/IP communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
+/-Double	+/- Double
+/-Int	+/- Int
16 Bit Group	16 Bit Group
ASCII	ASCII
Bit	Bit
Double	Double
Float	Float
Int	Int

See also

Migration of data types (WinCC flexible) (Page 164)

Migrating data types of Omron Hostlink/Multilink (WinCC flexible)

Migrating data types Omron Hostlink/Multilink

The Omron Hostlink/Multilink communication driver is not supported by WinCC, it is replaced by the Omron Host Link driver. The data types of the Omron Hostlink/Multilink communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
+/-DEC	Int
+/-LDEC	DInt
ASCII	String
BIN	Bool
BYTE	Byte
DEC	UInt
IEEE	Real
LDEC	UDInt

See also

Migration of data types (WinCC flexible) (Page 164)

Migrating data types of SIMATIC S7 200 (WinCC flexible)

Migrating data types SIMATIC S7 200

The data types of the SIMATIC S7 200 communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
Bool	Bool
Byte	Byte
Char	Char
DInt	DInt
DWord	DWord
Int	Int
Real	Real
StringChar	StringChar
Timer	Timer
Word	Word

See also

Migration of data types (WinCC flexible) (Page 164)

Migrating data types of SIMATIC S7 300/400 (WinCC flexible)

Migrating data types SIMATIC S7 300/400

The data types of the SIMATIC S7 300/400 communication driver are mapped as follows in the migration to WinCC:

Data type in WinCC flexible	Data type in WinCC
Bool	Bool
Byte	Byte
Char	see below
Counter	see below
Date	Date
Date and Time	Date_And_Time
DInt	DInt
DWord	DWord
Int	Int

Data type in WinCC flexible	Data type in WinCC
Real	Real
String	String
StringChar	see below
Time	Time
Time of Day	Time_Of_Day
Timer	see below
Word	Word

Special considerations for some data types

There are special considerations to be made when migrating external tags that contain data types of a SIMATIC S7-300/400 PLC.

Mapping of the S7 data type "Char"

The S7 data type "Char" is a data type for mapping characters according to the specification. However, since this data type is often used for reading and writing numerical values, it is mapped in WinCC to the S7 data type "Byte". If this should be the case during migration, an alarm will appear in the output window.

If the S7 data type "Char" is used for numerical values and negative numbers were configured at the point of use, the result is an error in mapping to the S7 data type "Byte". The S7 data type "Byte" cannot map any negative numbers. You have to adapt the configuration accordingly to correct the error. Use a signed data type, such as the data type "Int", for processing positive and negative numerical values.

If the S7 data type "Char" is used for mapping characters, you must change the configuration after migration. To represent characters, use the data type "String".

When an integrated project is migrated, the data type "Char" in WinCC is also migrated to the data type "Byte". With a connected PLC tag, the data type "Char" remains "Char". As a result of changing the data type of the HMI tag, symbolic addressing of the tags in question is not migrated. After migration. the tags are interconnected by absolute addresses and continue to work. If you want to restore symbolic addressing, you have to change the configuration accordingly after the migration.

Mapping an array of the S7 data type "Char"

An array of the S7 data type "Char" is mapped to an array of the data type "Byte" during migration.

If an array of the S7 data type "Char" is used for numerical values and negative numbers were configured at the point of use, the result is an error in mapping to an array of the S7 data type "Byte". The S7 data type "Byte" cannot map any negative numbers. You have to adapt the configuration accordingly to correct the error. Use a signed data type, such as the data type "Int", for processing positive and negative numerical values.

Mapping of the S7 data type "Counter"

An external tag with the S7 data type "Counter" with counter address is mapped to the S7 data type "Counter". The address will be retained.

If an external tag with the S7 data type "Counter" addresses a data block or a bit memory address, it is is mapped to the S7 data type "Word". The address will be retained. The migration sets the coding to "SimaticBCDCounter".

The S7 data type "Counter" has a value range of 0-999. When supplied by the S7 data type "Word" the value range may be exceeded on the PLC side. Ensure that you are observing the value range.

Example:

WinCC flexible

Tag	S7 data type	Address	Comment
Counter_Actual_Value	Counter	C10	BCD coded counter value
Counter_Setpoint_Value	Counter	DB10.DBW200	BCD coded counter value
Counter_Setpoint_Value#2	Counter	MW20	BCD coded counter value

WinCC

Tag	S7 data type	Address	Coding	Comment
Counter_Actual_Value	Counter	%C10	<Standard>	BCD coded counter value
Counter_Setpoint_Value	Word	%DB10.%DBW200	SimaticBCDCounter	BCD coded counter value
Counter_Setpoint_Value#2	Word	%MW20	SimaticBCDCounter	BCD coded counter value

Mapping of the data type "StringChar"

In WinCC there is no corresponding data type to which the "StringChar" data type can be mapped. Mapping in WinCC depends on the property "Length" of the S7 data type.

A tag of the "StringChar" data type with the "Length" property > 1 is migrated to an array of the S7 data type "Char". The length of the array corresponds to the length of the originally configured data type "StringChar".

If the property "Length" = 1, the data type in WinCC is migrated to an array of the S7 data type "Char" with length = 1. The expression for an array with an element is "Array[0 ..0] of Char".

Mapping of the S7 data type "Timer"

An external tag with the S7 data type "Timer" with timer address is mapped to the S7 data type "Timer". The address will be retained.

If an external tag with the S7 data type "Timer" addresses a data block or a bit memory address, it is is mapped to the S7 data type "S5 Time". The address will be retained.

Example:

WinCC flexible

Tag	S7 data type	Address	Comment
Timer_Actual_Value	Timer	T10	BCD coded timer value
Timer_Setpoint_Value	Timer	DB10.DBW200	BCD coded timer value
Timer_Setpoint_Value#2	Timer	MW20	BCD coded timer value

WinCC

Tag	S7 data type	Address	Comment
Timer_Actual_Value	Timer	%T10	BCD coded timer value
Timer_Setpoint_Value	S5Time	%DB10.%DBW200	BCD coded timer value
Timer_Setpoint_Value#2	S5Time	%MW20	BCD coded timer value

See also

Migration of data types (WinCC flexible) (Page 164)

4.1.8 Migrating integrated projects

4.1.8.1 Migrating an integrated project

Introduction

In integrated projects, you use SIMATIC controllers and WinCC components together in a project. When an integrated project is migrated, the complete project will be migrated with components from WinCC and STEP 7. Configured connections between control and visualization remain intact.

Migrating an integrated project

When migrating an integrated project, the same requirements apply for the STEP 7 component as those for migration of a non-integrated STEP 7 project. The objects and properties contained in the WinCC component must also be supported in WinCC (TIA Portal). For detailed information, refer to the documentation for WinCC.

Also note that the initial project must be compiled before the migration.

To fully migrate an integrated project, the following components must be installed on the PG/PC performing the migration:

- STEP 7 V5.4 SP5 or STEP 7 V5.5
- WinCC V7.0 SP3 or WinCC Flexible 2008 SP2 and SP3

To be able to fully post-edit an integrated project, the following components must be installed on the PC for post-editing:

- STEP 7 Professional V12 SP1
- WinCC Basic, WinCC Comfort/Advanced or WinCC Professional, depending on the components used

Using the migration tool

It is necessary to use the migration tool under the following circumstances:

- The initial project is not located on the same PG/PC as the installation of the TIA Portal.
- SCADA devices are included in the initial project. These can only be migrated with the migration tool.
- WinCC Professional V12 and STEP 7 with WinCC V7.0 SP3 cannot be installed on the same PG / PC. Therefore, integrated projects with WinCC V7.0 SP3 parts must be prepared for migration with the migration tool.

Migration of the STEP 7 part of an integrated project

An integrated project is always fully migrated. Individual components cannot be migrated on their own. You can only migrate the included STEP 7 project alone, if you have previously deleted all HMI stations in the SIMATIC stations in the SIMATIC Manager and then recompiled the project in NetPro.

Alternatively, you can open the project in an installation of STEP 7 V5.4 SP5 or V5.5 without an installation of WinCC. Then, save the project again and select the "Reorganize" function during saving. The WinCC parts are then automatically removed when the copy is saved.

You can then migrate the STEP 7 project without the WinCC project.

Migration of an integrated project with the hardware configuration

In integrated projects, HMI devices are migrated even if the hardware configuration is not included in the migration. The STEP 7 part of the hardware configuration, including network configurations, and connections and interrupts, is migrated only if you include the hardware configuration in the migration. Otherwise, unspecified modules will be created for the STEP 7 devices and you will need to convert them into suitable modules after the migration.

HMI modules that are plugged into a PC station are converted to a separate Station during the migration. If you perform the migration without including the hardware configuration, the migrated project then contains a non-specified SIMATIC PC Station and a SIMATIC PC Station with the HMI devices. References to HMI devices are not imported during migration. When the hardware configuration is included, the migrated project contains two separate stations: the HMI Station and the PC Station.

Storage location of an integrated WinCC project

If you migrate an integrated project, the HMI part of it must be on the same PG/PC as the STEP 7 part of the project. If the HMI part is on a different PG, then only the STEP 7 part will be migrated.

Unsupported objects

The following components are not supported for migration:

- STEP 7 multiproject
  A STEP 7 multiproject cannot be migrated. Migration will be canceled.
- Central Archive Server - CAS
  If a CAS is part of an integrated project, then the migration will be carried out but the CAS data will not be migrated.

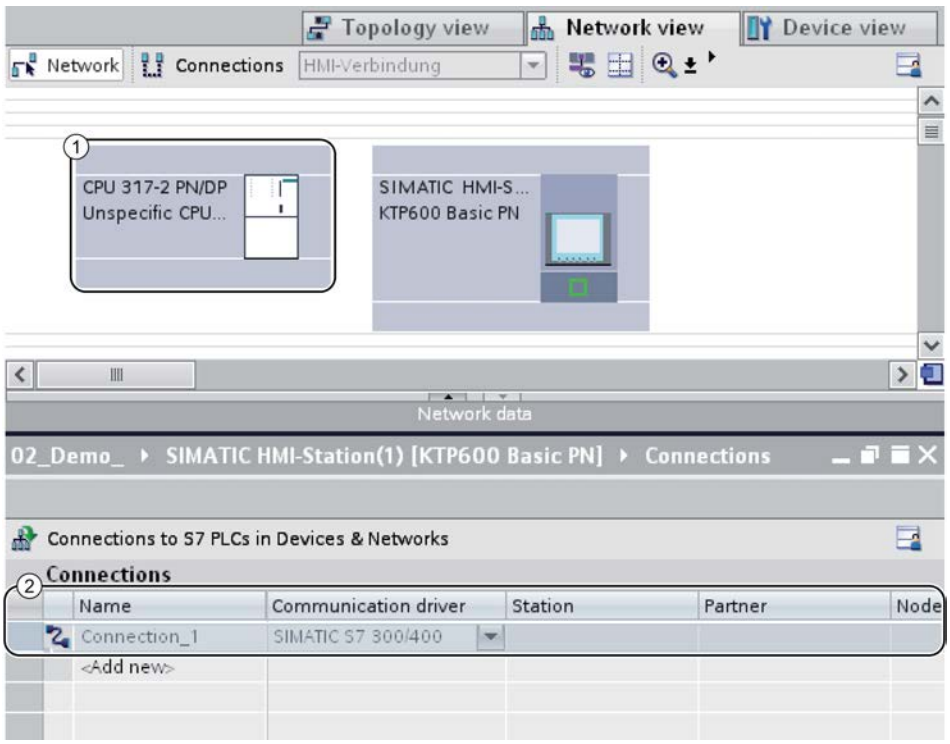
See also

Post-editing integrated projects (Page 174)

4.1.8.2 Post-editing integrated projects

If you have migrated an integrated project without hardware configuration, unspecified CPUs are used instead of the CPUs of the original project. Since no connection can exist between an unspecified CPU and an HMI device, connections from the source project are also imported only unspecified.

The following figure shows the state after a Migration without hardware configuration in an example project:



① The original CPU 317-2 PN/DP was replaced with an unspecified CPU during migration.

② The link between the CPU and HMI device is also unspecified and must be renewed.

Procedure

To continue to use an integrated project after the migration, follow these steps:

1. Convert the unspecified devices into suitable devices again.
2. Restore the integrated HMI connection between the HMI device and the PLC.
3. Connect all HMI tags to the newly created integrated connection.
4. Restore the connection between HMI tags and PLC tags.
5. Delete the non-integrated HMI connection.

In the following chapters a sample project is used to describe the individual steps in more detail.

See also

Converting unspecified CPUs into specified CPUs (Page 175)

Creating an integrated HMI connection (Page 177)

Re-linking HMI tags (Page 179)

Deleting an unspecified connection (Page 180)

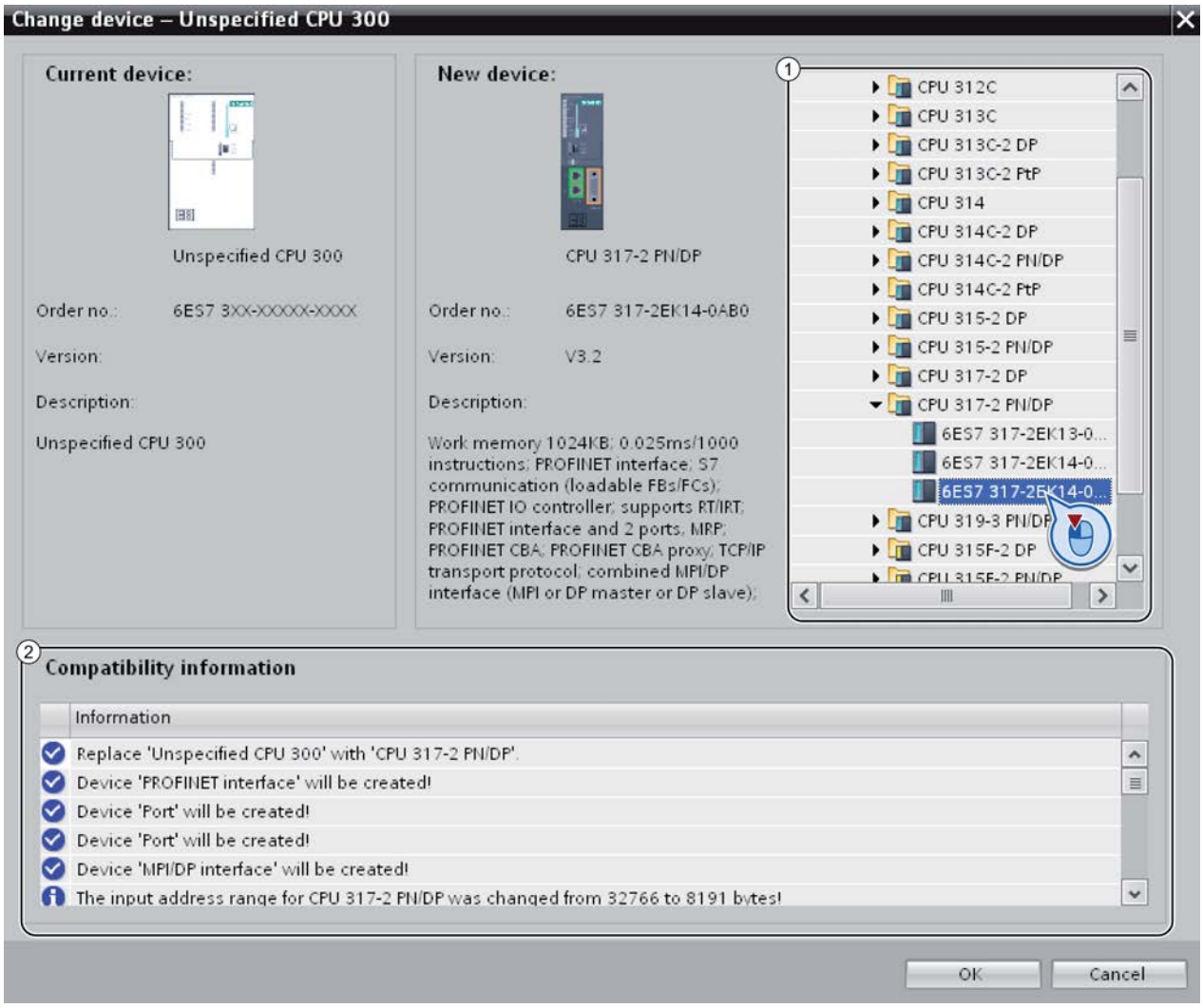
4.1.8.3 Converting unspecified CPUs into specified CPUs

The first step after the migration without hardware configuration is the conversion of the unspecified CPUs into specified CPUs. Unspecified CPUs are placeholders for certain CPUs from the hardware catalog that are not currently known. You can define general parameters and home the CPUs already in the user program. However, the project is not fully functional until the unspecified CPU has been specified.

Specifying a CPU using module replacement

To use module replacement to specify an unspecified CPU, follow these steps:

1. Select the unspecified CPU in the network or device view.
2. Select the "Replace device" command in the shortcut menu.
   The "Replace device" dialog opens.



3. Under "New device" in the tree structure, select the module with which you want to replace the unspecified CPU. (Area 1)
   "Compatibility information" provides you with information on the extent to which the selected CPU is compatible with the configuration in source project. (Area 2)
4. Click "OK".
5. Perform the above-described steps for all unspecified CPUs.

See also

Creating an integrated HMI connection (Page 177)

4.1.8.4 Creating an integrated HMI connection

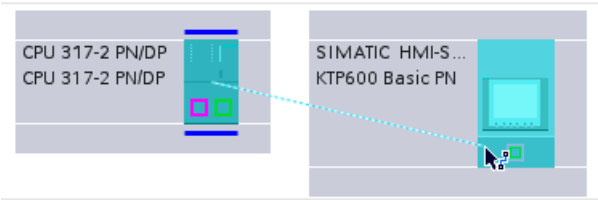
After you have specified the unspecified CPU, establish the connection to the HMI-device.

Procedure

To create a connection graphically, follow these steps:

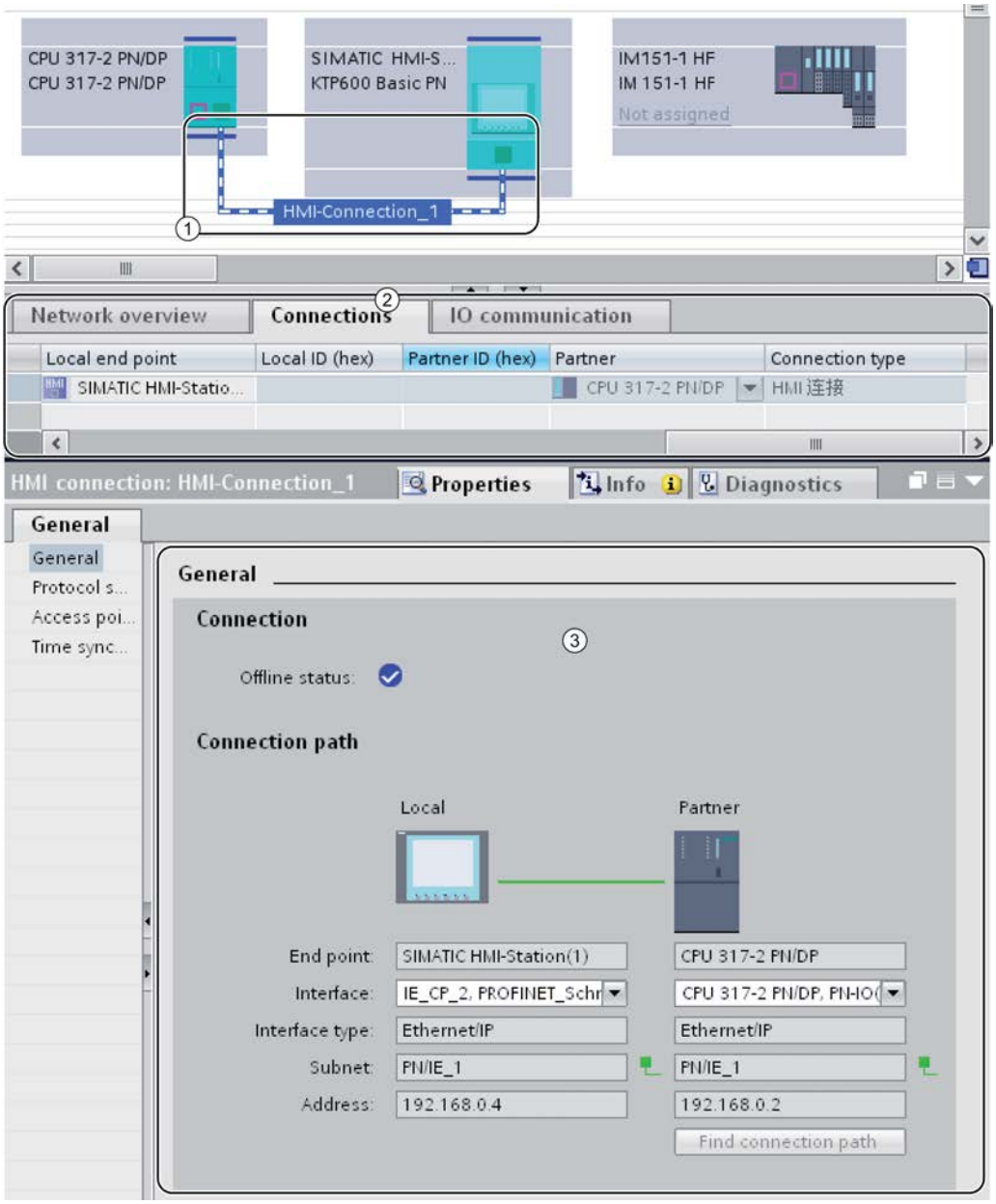
1. On the toolbar, click the "Connections" icon. This activates connection mode.

2. Select the connection type "HMI connection" in the adjacent drop-down list.
   The network view highlights in color all CPUs and HMI devices that can be used for an HMI connection.
3. You can now have the connection path automatically determined, or explicitly select a connection path via specific interfaces:
   - Allow connection path to be automatically determined
     Select the source CPU for a connection. Drag the mouse to the target components. Confirm the connection endpoint with another mouse click.
     Alternatively: While holding down the shift button, select the target components and with the right mouse button select the "Add new connection" command.
   - Selecting an explicit connection path from interface to interface
     Click on the subnet interface in the device for which you want to create a connection. Hold down the mouse button, drag the cursor to the relevant interface in the target device and then release the mouse button.



Result

The following figure shows the state after the integrated connection has been created:



① An integrated HMI connection is created and highlighted in the network view.

② The connection is shown in the connection table of the components.

③ The connection can be edited in the connection properties.

See also

Re-linking HMI tags (Page 179)

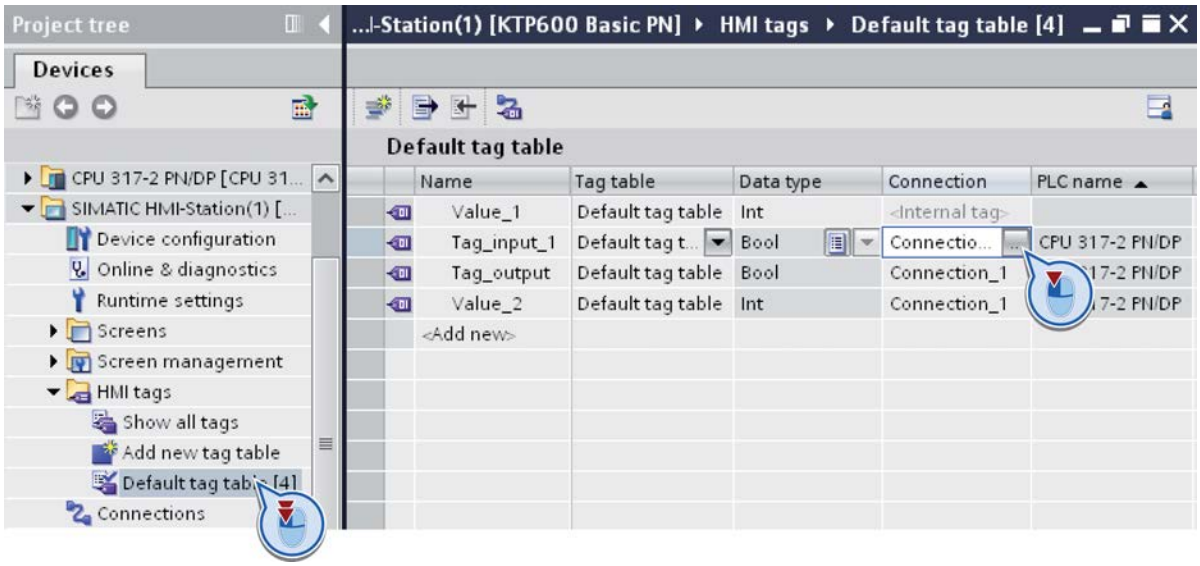
4.1.8.5 Re-linking HMI tags

When you have created a new HMI connection between the CPU and HMI device, you have to assign the existing HMI tags to the new connection. Perform the following steps for each line in the relevant tag table.

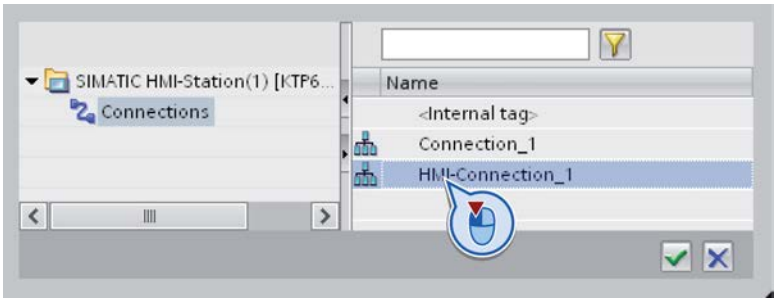
Procedure

To re-link HMI tags, follow these steps:

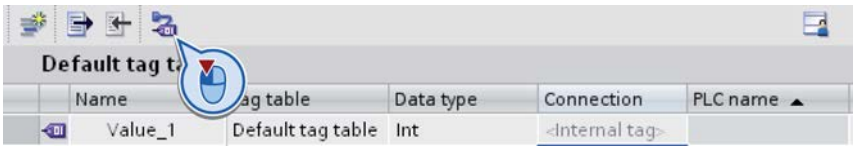
1. In the project tree, navigate to the HMI tags and double-click the relevant tag table to show this in the work area.
   The tag table opens.



2. Click the " ... " button in the "Connection" column.
   A dialog box for selecting the connection opens.
3. Select the newly established HMI connection.



4. Click the "✓" button to apply the selected connection.
5. On the toolbar, click the "Re-connect PLC tag" button.



See also

Deleting an unspecified connection (Page 180)

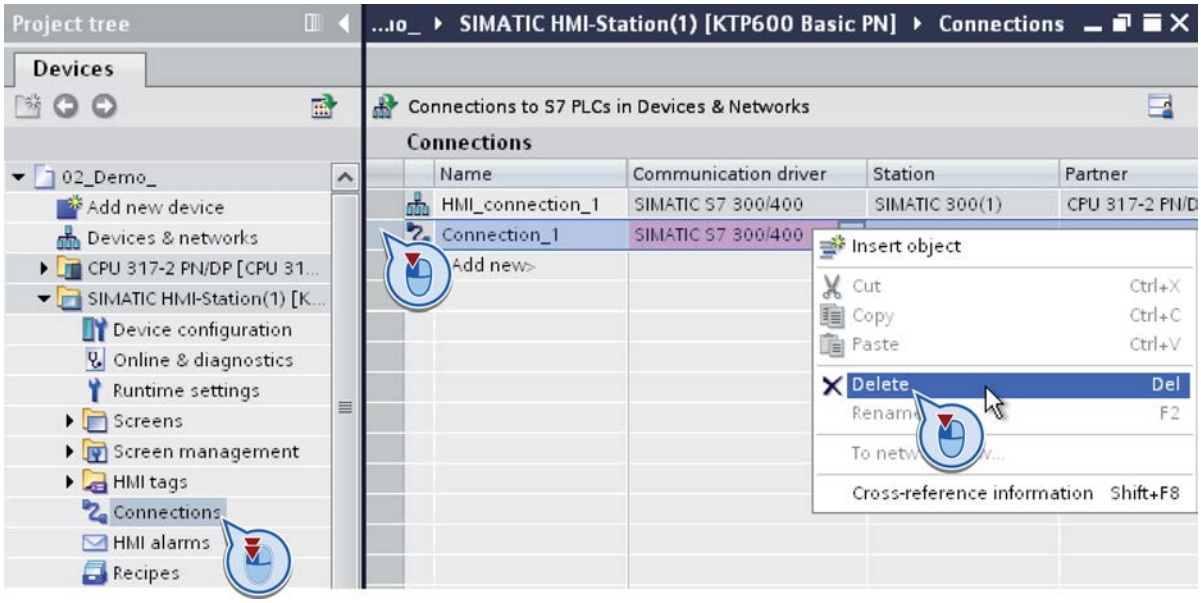
4.1.8.6 Deleting an unspecified connection

Finally, you can remove unspecified connections that still remain from the source project.

Procedure

To delete unspecified connections, follow these steps:

1. In the project tree, open the HMI device and double-click the "Connections" entry.
   The connection table opens.



2. Select the row with the old connection in the table.
3. Select the "Delete" command in the shortcut menu of the connection line.
4. Perform the above-described steps for all unspecified connections of the source project.

4.2 Migrating PLC programs to a S7-1500 CPU

4.2.1 Basic information on migrating PLC programs

Introduction

You can migrate PLC programs within the TIA Portal from a CPU of the S7-300/400 series to a CPU of the S7-1500 series. Migration within the TIA Portal is called "PLC migration". With

this migration, the new CPU will be created in the project and the existing PLC program copied to the new CPU. The old CPU remains available unchanged in the project.

The advantage of migration compared to copying is that the program is automatically adapted to the new CPU family as much as possible. Program structures that are no longer permitted are updated. Instructions that are not available on the new CPU are replaced with corresponding S7-1500 instructions.

Scope of migration

During PLC migration, all components of the PLC program are copied to the newly created CPU. The following objects are included, for example:

- Program blocks, whose programming language is supported by S7-1500 and that have no know-how protection.
- PLC tag tables
- Watch and force tables
- PLC data types
- Technological objects
- User-defined groups in the project tree

The hardware configuration of the original CPU is not automatically transferred to the new CPU of the S7-1500 series.

WARNING

Preventing personal injury and material damage

Changes are made to the program during PLC migration in some cases. It is therefore very important that you thoroughly review the program in a test environment after migration before putting it in operation.

Note

Additional support for PLC migration

In the Siemens Industry Online Support, you will find the latest FAQs on the topic "PLC migration" (http://support.automation.siemens.com/WW/view/en/67858106).

If you need further assistance for PLC migration, please contact SIMATIC Customer Support.

Note

Additional information on the TIA Portal

Additional information on all aspects of the TIA Portal can be found under TIA Portal in the Siemens Industry Online Support (http://support.automation.siemens.com/WW/view/en/65601780).

There you will find information on our training offers. For experienced users of the classic SIMATIC S7 system who want to upgrade efficiently to SIMATIC S7-1500, we recommend "TIA-SYSUP - SIMATIC TIA Portal system retraining course for SIMATIC S7-1500".

4.2.2 Performing migration

Requirement

- A consistent reference program is available.
- All blocks are compiled up-to-date. This also applies to blocks that are included in the project, but will not be called during program execution.
- The program was created with TIA Portal V12 or later or upgraded to this version.

Preparing for migration

Proceed as follows to prepare the program for migration:

1. Open the CPU with the reference program in the project tree.
2. Open the folder "Program blocks" and check whether it contains know-how protected blocks.
3. Remove the know-how protection from the blocks.
4. Open the folder "Program blocks > System blocks" and check whether the block contained therein include the extension "_LF" (legacy functions).
   These are library blocks from STEP 7 that are no longer supported in the TIA Portal. These blocks are not transferred to S7-1500 during PLC migration because they are know-how protected.
5. If possible, replace the blocks with an instruction from the "Instructions" task card.

Procedure

To migrate a PLC program within the TIA Portal to S7-1500, follow these steps:

1. Open the CPU with the reference program in the project tree.
2. Open the device configuration.

3. Select the CPU containing the reference program in the device view or network view.
4. Select the "Migrate to S7-1500" command in the shortcut menu.
   The "Migrate to S7-1500" dialog opens.
5. In the "New device" area, select the CPU to which you want to migrate your program and confirm with "OK".
   A security message appears informing you that the program will be changed during the migration.
6. Confirm the security prompt.
   A dialog opens in which you can specify which instructions you want to use for point-to-point communication in the migrated program.
7. Select the required set of instructions. You must select one of the two options even if you do not use point-to-point communication. The selection does not have an effect on the program in this case.
   The PLC program is migrated. An alarm then appears notifying you whether errors occurred during the migration. The alarm also contains a link to the migration log.
8. Open the migration log. It contains details on the migration and informs you about program changes which you must perform to make your program executable on the new CPU.
9. Process all the information contained in the migration log.
10. Compile the migrated project.

Result

A new CPU is created next to the original CPU in the project tree. It contains the migrated PLC program.

To access the full functionality of the S7-1500, we recommend that you enable optimized block access for the migrated blocks. For more information on this, refer to "See also".

Note

Note that only the PLC program is converted by the PLC migration. The hardware configuration of the original CPU is not automatically transferred to a CPU of the S7-1500 series. Perform the hardware configuration of the new CPU after the migration manually.

See also

Using logs (Page 331)

Blocks with optimized access (Page 2175)

4.2.3 Displaying a log file

A log file is created for each migration. The log file contains the following information:

- Modifications to objects made during migration.
- Information on necessary adaptations you need to make to the program.

Procedure

To display the log file of the migration, follow these steps:

1. Open the "Common data > Logs" folder in the project tree. It contains the logs of all previously performed migrations.
2. Double-click the required migration log.
   The log opens.
3. Messages that relate to a certain position in the program are marked with an arrow in the "Go to" column. Double-click on the arrow to jump to the relevant position in the program.
4. Messages for which additional information is available are marked with a question mark in the "?" column. To obtain additional information on the message, click the question mark.

See also

Using logs (Page 331)

4.2.4 Special considerations for migrating PLC programs

4.2.4.1 Information on migrating PLC programs

Organization blocks

The following rules apply to migration of organization blocks:

- Block names and numbers are transferred unchanged.
- The block interface is also accepted unchanged.
- The OB is assigned to the event corresponding to the OB type.
- The OB parameters, such as the priority, remain unchanged. If additional parameters can be assigned in the new CPU, they will be assigned default values.

See also:

Basics of error handling (Page 2821)

Some organization blocks (OBs) are not supported by S7-1500 and therefore cannot be migrated. The following table provides an overview of these organization blocks and contains information on how you can reproduce the functionality in your program.

Non-migratable OB	Notes
OB60 (multicomputing interrupt)	The OB cannot be replaced. Multicomputing is not available on S7-1500.
OB65 (technology synchronization interrupt)	Check whether the OB in your program can be replaced by the MC servo OB or the MC interpolator OB.
OB7x (redundancy error)	The OB cannot be replaced. Redundancy is not available on S7-1500.
OB81 (power supply error)	Check whether the OB in your program can be replaced by the diagnostic interrupt OB.
OB84 (CPU hardware fault)	Check whether the OB in your program can be replaced by the diagnostic interrupt OB.
OB85 (program execution error)	Check whether the OB in your program can be replaced by the pull/plug OB or the rack error OB.
OB87 (communications error)	Check whether the OB in your program can be replaced by the diagnostic interrupt OB.
OB88 (processing interrupt)	Check whether the OB in your program can be replaced by the programming error OB.
OB90 (background cycle)	The OB cannot be replaced. Background processing is not available on S7-1500.
OB101 (hot restart)	The OB cannot be replaced. Hot restart is not available on S7-1500.
OB102 (cold restart)	Check whether the OB in your program can be replaced by the startup OB.

See also:

Auto-Hotspot

Auto-Hotspot

Status word

The status word is no longer fully supported in S7-1500. It contains less information and can now be evaluated only in STL. LAD and FBD no longer support the evaluation of the status word. The chapters below provide information on the migration of status word access in the individual programming languages.

See also:

Migration of LAD/FBD programs to S7-1500 (Page 213)

Migration of STL programs to S7-1500 (Page 216)

Basic information on the status word (Page 2466)

Using block parameters

The following rules apply to the use of block parameters within the block in S7-1500:

- Input parameters may only be read.
- Output parameters may only be written.
- In/out parameters may be read and written.
- Function values (Ret_Val) may only be written.

If your program does not conform to these rules, a warning is output during compilation. In this case, convert the relevant input or output parameter to an in/out parameter.

Additional rules apply to the parameter assignment of functions (FC).

See also:

Parameter assignment of functions (Page 2187)

Access to block parameters during program execution (Page 2190)

Block parameter as edge memory bit in functions (FC)

The above-mentioned rules for parameter assignment must also be observed when edge evaluations are programmed in functions (FC):

As edge memory bit, you require a data value that may be both read and written and that is retained for several cycles. Since input parameters (input) may only be read and output parameters (output) may only be written, these cannot be used as edge memory bits. Temporary local data (Temp) can also not be used as edge memory bits, because it is only available for one cycle.

Therefore, use only in/out parameters (InOut) as edge memory bits in functions (FC). You can also use a single bit of an in/out parameter as edge memory bit, if you address this via a slice access.

The following example shows you how to use an in/out parameter as edge memory bit.

STL

```
FP #InOutFlagStore.x0
```

"InOutFlagStore" is an in/out parameter of the BYTE data type. The address "#InOutFlagStore.x0" addresses the bit address 0, which is used as edge memory bit.

Automatic initialization of block parameters in functions (FC)

In S7-300/400, temporary local data (Temp) of a function must be pre-assigned with a value. Otherwise, it may happen that the program operates with undefined values.

In S7-1500, the risk of continuing to process undefined values is considerably reduced because the parameters below are automatically initialized when the block is called:

- Temporary local data of the STRING data type is always pre-assigned with the correct length information (maximum length and actual length).
- Temporary local data with the elementary data type is automatically initialized in functions (FC) with optimized access. It is then assigned the value that has been predefined for the specified data type. For example, the value "false" is predefined for BOOL. Elements of the PLC data types are pre-assigned with the default value that is specified in the declaration of the PLC data type (UDT). ARRAY elements are pre-assigned with the value "0", even if they are used within a PLC data type.

In S7-300/400, output parameters (Output) in functions had to be assigned a value, because unwanted reactions could otherwise have occurred in the program. When jump instructions or RLO instructions were used, each supported program path then had to be checked.

In S7-1500, the risk of undefined output parameters is considerably reduced because the output parameters of the elementary data type are automatically initialized when blocks are called. They are then pre-assigned the value that was predefined for the specified data type. For example, the value "false" is predefined for BOOL. However, structured output parameters are not pre-assigned with a value. They are transferred as pointer when a block is called and therefore cannot be undefined.

See also:

Parameter assignment to functions (Page 2187)

CPU data blocks

Blocks that were generated by the CREAT_DB or CREATE_DB instructions in the CPU and are only available online are not migrated.

See also:

Migration of data block instructions (Page 198)

CPU data blocks (Page 2174)

Multiple instance capability

All function blocks in the CPUs of the S7-1500 series have multiple instance capability. This means FBs that were labeled as not having multiple instance capability in the reference program are converted to blocks with multiple instance capability by the migration. A note is output in the migration log for each converted FB. If you continue to use the FB as a single instance after migration, your program runs unchanged.

See also:

Call function blocks as single instance or multiple instance (Page 2182)

Blocks from STEP 7 libraries

Some library blocks from STEP 7 are no longer supported in the TIA Portal. If it is not possible to migrate such blocks in the TIA Portal, these blocks are labeled as know-how protected user blocks in the TIA Portal and given with the name suffix "_LF" (legacy function). These blocks are not transferred to S7-1500 during PLC migration within the TIA Portal because they are know-how protected.

If possible, check if you can replace the blocks with the "_LF" name suffix with an instruction from the "Instructions" task card prior to migration.

Invalid floating-point numbers

The handling of invalid floating-point numbers in S7-1500 differs from S7-300/400. Example:

- S7-300/400: The result of the expression "Invalid floating point number <> 1.0" is FALSE.
- S7-1500: The result of the expression "Invalid floating point number <> 1.0" is TRUE.

Note that the instructions in your program can produce other results due to this difference.

See also:

Floating-point numbers (Page 2252)

Absolute access to local data

Absolute access to the local data of a block was possible in S7-300/400 without declaring a symbol for the address.

Because pure absolute access in S7-1500 is no longer supported, migration deals with this as follows:

- If a temporary tag is declared in the block interface for an area within the local data, the absolute address is converted to a symbolic address.
- If there is no suitable temporary tag, the absolute address is retained. Because additional local tags may be created in some circumstances during migration, addresses may shift within the local data. Therefore, check to see if the absolute address you are using is still correct and if the correct data is being addressed. If possible, create a tag in the "Temp" section and use it for addressing. To specifically address areas within declared tags, you can overlay existing tags with AT or address individual areas with the syntax .X, .B, .W or .D.

See also:

Overlaying tags with AT (Page 2218)

Addressing areas of a tag with slice access (Page 2216)

The following examples show migration of absolute access to local data.

In the first table, you can see a program section prior to migration:

STL

```
L %LW20
L %LW3
+I
T %LW5
```

In the second table, you can see a program section after migration:

STL

```
L "MyTempVar1"
L "MyTempVar2"
+I
T %LW5
```

- Temporary tags were already declared for "LW20" and "LW3". These are used in the program after migration.
- No tag was declared for "LW5". You need to check if access is still correct after the migration.

Downloading software changes

CPUs of the S7-1500 series behave differently than CPUs of the S7-300/400 series with regard to the downloading of software changes. For example, they allow changes to be downloaded in STOP and RUN without affecting the actual values of previously loaded tags.

See also:

Auto-Hotspot

System status lists

CPUs of the S7-1500 series do not provide system status lists. However, you can read some of the information using the "GET_DIAG", "Geo2Log", "DeviceStates", or "ModuleStates" instruction.
Additional information on migration of system status lists is available under: Migration of the RDSYSST instruction (Page 200)

System diagnostics with "Report System Errors"

The CPUs of the S7-1500 series have integrated system diagnostics. "Report System Errors" is therefore no longer supported. If the program contains blocks for "Report System Errors", disable system diagnostics in the reference program prior to migration and recompile the program. The blocks in question are deleted during this process.

System diagnostics for the S7-1500 CPUs is enabled by default. You do not have to make any additional settings.
To change the settings for the system diagnostics, select the CPU in the device configuration. The settings for the system diagnostics are then shown in the Inspector window under "Properties > System diagnostics". The alarms of the system diagnostics are enabled and can be deselected if they are not needed.

PROFINET/PROFIBUS diagnostics

The following rules apply to migration of PROFINET/PROFIBUS diagnostics:

- "Report system errors" is replaced by the integrated system diagnostics.
- FB 126 is not supported and can therefore not be migrated.
- Use the "DeviceStates" and "ModulStates" instructions for application-specific evaluation.

Note

Further support

In the Siemens Industry Online Support, you will find the latest FAQs on the topic "PLC migration" (http://support.automation.siemens.com/WW/view/en/67858106).

If you need further assistance for PLC migration, please contact SIMATIC Customer Support.

See also

Parameter transfer at block call (Page 2185)

4.2.4.2 Migration of instructions to S7-1500

Information on migrating instructions

Automatic migration of instructions

During migration, the instructions used in the initial program are carried over to the new program as much as possible. In the process, necessary adaptations are made automatically whenever possible without altering the semantics of the initial program. If multiple versions of an instruction exist, the migration always uses the newest instruction version.

Instructions that are not available in S7-1500 are replaced automatically whenever possible by a compatible or similar instruction.

Migration scenarios

The following table shows the possible scenarios for migrating instructions:

Class	Scenario	System reaction	Manual adaptation of the program
1	The instruction is identical in both CPU families.	The instruction is migrated.	No adaptation is required. The semantics of the migrated program is unchanged and can be compiled.
2	The instruction is no longer available in S7-1500, but there is a compatible, new instruction.	The instruction is replaced by the new instruction. Additional instructions that are necessary to preserve the semantics of the initial program may be inserted. The migration log reports the replacement.	No adaptation is required. The semantics of the migrated program is unchanged and can be compiled.

Class	Scenario	System reaction	Manual adaptation of the program
3	The instruction is no longer available in S7-1500, but there is a similar instruction.	The instruction is replaced by the similar instruction. The migration log reports the replacement. Locations to be checked are marked in the program. You receive information regarding necessary adaptations in the form of a comment at the appropriate location in the program or when compiling.	You must check the program and possibly adapt it.
4	The instruction is no longer available in S7-1500 and there is no similar instruction.	The instruction is highlighted in red. The program cannot be compiled.	You must check the program and possibly adapt it.

Information regarding instructions of migration class 3

Some instructions that were available in S7-300/400 are no longer offered in S7-1500 because there are now easier or more efficient methods available for solving certain tasks. The PLC migration supports you during the migration by replacing these instructions with appropriate new instructions. However, the instructions from scenario 3 cannot be completely migrated automatically. For example, new formal parameters may need to be added. The locations that cannot be migrated automatically are marked in the program. You receive information regarding necessary adaptations in the form of a comment directly at the appropriate location in the program, in the migration log, or when compiling.

The following manual adaptations may be necessary:

- If the new instruction has additional formal parameters, assign these parameters with suitable actual parameters.
- If the new instruction has modified parameter names, change the parameter assignment.
- If the migration cannot unequivocally ensure that the parameter assignment is semantically identical to the reference program, the actual parameters to be checked are marked. Check the semantics at these locations in the program after migration and correct it if necessary.
- If the reference program contains hardware identifiers, for example at the "LADDR" parameter, these are also marked. Enter the new HW identifier at the "LADDR" parameter.
- If the new instruction has modified error codes, adapt the error handling in the program.

Some examples of instructions of migration class 3 can be found in the chapter "Migration of data block instructions (Page 198)".

Information regarding instructions of migration class 4

Some instructions are no longer offered in S7-1500 because the functionality has fundamentally changed. These instructions are not replaced by the migration. They are shown

in red in the program and prevent the program from being compiled. You receive information regarding necessary adaptations in the form of a comment directly at the appropriate location in the program, in the migration log, or when compiling.

In the following chapters, you can find an overview of the non-migratable instructions of migration class 4 as well as information on alternative solutions.

Note

Further support

In the Siemens Industry Online Support, you will find the latest FAQs on the topic "PLC migration" (http://support.automation.siemens.com/WW/view/en/67858106).

If you need further assistance for PLC migration, please contact SIMATIC Customer Support.

Overview of non-migratable instructions

Introduction

Some instructions from S7-300/400 cannot be migrated to S7-1500. This is the case, for example, when the instructions can be used only in connection with certain modules or when the functionality of the instruction in S7-1500 needs to be implemented differently.

Information on replacing non-migratable instructions (Class 4)

The following are examples of instructions that cannot be migrated and information on their replacement:

Instruction group	Instruction which cannot be migrated	Notes
Additional instructions	SETP	Check whether the instruction in your program can be replaced by the "SET_BF" instruction.
	RESETP	Check whether the instruction in your program can be replaced by the "RESET_BF" instruction.
	DRUM_X	Check whether the instruction in your program can be replaced by the "DRUM" instruction.
	TONR_X	Check whether the instruction in your program can be replaced by the "TONR" instruction.
	RESET	Check whether the instruction in your program can be replaced by the "RESET_BF" instruction.
	SET	Check whether the instruction in your program can be replaced by the "SET_BF" instruction.
	WSR	The instruction cannot be replaced. For more assistance, please contact SIMATIC Customer Support.
	SHRB	The instruction cannot be replaced. For more assistance, please contact SIMATIC Customer Support.
	RESETI	Check whether the instruction in your program can be replaced by the "RESET_BF" instruction.

Instruction group	Instruction which cannot be migrated	Notes
	SETI	Check whether the instruction in your program can be replaced by the "SET_BF" instruction.
Runtime control	COMPRESS	Delete the instruction. In S7-1500, it is no longer necessary to compress the work memory of the CPU or the load memory.
	PROTECT	Configure the protection level and password via the front panel of the S7-1500 CPU or in the module properties of the CPU.
	CIR	The instruction cannot be replaced. For more assistance, please contact SIMATIC Customer Support.
Date and time	SET_CLKS	Check whether the instruction in your program can be replaced by the "WR_SYS_T" instruction.
	LOC_TIME	Check whether the instruction in your program can be replaced by the "RD_SYS_T" instruction.
	BT_LT	Check whether the instruction in your program can be replaced by the "RD_SYS_T" instruction.
	LT_BT	Check whether the instruction in your program can be replaced by the "RD_SYS_T" instruction.
	S_LTINT	Check whether the instruction in your program can be replaced by the "SET_TINTL" instruction.
	SET_SW	Check whether the instruction in your program can be replaced by the "WR_SYS_T" or "RD_SYS_T" instruction.
	SET_SW_S	Check whether the instruction in your program can be replaced by the "WR_SYS_T" or "RD_SYS_T" instruction.
	TIMESTMP	Check whether the instruction in your program can be replaced by the "WR_SYS_T" or "RD_SYS_T" instruction.
	WS_RULES	The instruction cannot be replaced. For more assistance, please contact SIMATIC Customer Support.
Distributed I/O	D_PRAL	The instruction cannot be replaced. For more assistance, please contact SIMATIC Customer Support.
	ASi_3422	The instruction cannot be replaced. For more assistance, please contact SIMATIC Customer Support.
	SALRM	The instruction cannot be replaced. For more assistance, please contact SIMATIC Customer Support.
Module parameter assignment	PARM_MOD	The instruction cannot be replaced. For more assistance, please contact SIMATIC Customer Support.
Interrupts	MP_ALM	Multicomputing is not available on S7-1500.
	REPL_VAL	The instruction cannot be replaced. For more assistance, please contact SIMATIC Customer Support.
Alarms	EN_MSG	The instruction cannot be replaced. For more assistance, please contact SIMATIC Customer Support.
	DIS_MSG	The instruction cannot be replaced. For more assistance, please contact SIMATIC Customer Support.
	ALARM_SQ	Check whether the instruction in your program can be replaced by the "Program_Alarm" instruction.
	ALARM_S	Check whether the instruction in your program can be replaced by the "Program_Alarm" instruction.

Instruction group	Instruction which cannot be migrated	Notes
	ALARM_SC	Check whether the instruction in your program can be replaced by the "Program_Alarm" instruction.
	WR_USMSG	The instruction cannot be replaced. For more assistance, please contact SIMATIC Customer Support.
	READ_SI	The instruction cannot be replaced. For more assistance, please contact SIMATIC Customer Support.
	DEL_SI	The instruction cannot be replaced. For more assistance, please contact SIMATIC Customer Support.
	ALARM_DQ	Check whether the instruction in your program can be replaced by the "Program_Alarm" instruction.
	ALARM_D	Check whether the instruction in your program can be replaced by the "Program_Alarm" instruction.
	NOTIFY_8P	Check whether the instruction in your program can be replaced by the "Program_Alarm" instruction.
	ALARM	Check whether the instruction in your program can be replaced by the "Program_Alarm" instruction.
	ALARM_8	Check whether the instruction in your program can be replaced by the "Program_Alarm" instruction.
	ALARM_8P	Check whether the instruction in your program can be replaced by the "Program_Alarm" instruction.
	NOTIFY	Check whether the instruction in your program can be replaced by the "Program_Alarm" instruction.
	AR_SEND	Check whether the instruction in your program can be replaced by the "Program_Alarm" instruction.
Diagnostics	RDSYSST	The instruction "RDSYSST: Read system status list" is no longer available in S7-1500 as the CPUs of this product series do not provide system status lists. However, you can read some of the information using the "GET_DIAG", "Geo2Log", "DeviceStates", or "ModuleStates" instruction. You can find additional information in the chapter "Migration of the instruction RDSYSST (Page 200)" and from SIMATIC Customer Support.
	OB_RT	Check whether the instruction in your program can be replaced by the "RD_SYS_T" instruction.
Table functions	ATT	The instructions cannot be replaced. If you need the instructions in your program, you have to implement them yourself. For additional support, refer to the chapter "Migration of table instructions (Page 202)" and to SIMATIC Customer Support.
	FIFO	
	TBL_FIND	
	LIFO	
	TBL	
	TBL_WRD	
	WRD_TBL	
	DEV	
	CDT	
	TBL_TBL	

Instruction group	Instruction which cannot be migrated	Notes
	PACK	
iSlave	SET_ADDR	The instruction cannot be replaced. For more assistance, please contact SIMATIC Customer Support.
PID Control	TUN_EC	Check whether the instructions in your program can be replaced by the "PID_Compact" or "PID_3Step" technology objects.
	TUN_ES	
Function modules	CNT_CTRL	Check whether the instructions in your program can be replaced by the "High_Speed_Counter" technology object in conjunction with the "S7-1500 TM Count 2x24V" or "TM PosInput 2" counter module. Alternatively, check whether the required functionality can be implemented with the "ET 200S 1Count24V" or "1Count5V" module.
	DIAG_INF	
	CNT_CTL1	
	CNT_CTL2	
	CNT2WRPN	
	CNT2RDPN	
	CNT2_CTR	
	DIAG_RD	
	ABS_INIT	Check whether the functions can be replaced by "S7-1500 Motion" or "ET 200S 1PosU".
	ABS_CTRL	
	ABS_DIAG	
	CAM_INIT	Check whether the encoder acquisition is possible via the modules "S7-1500 TM Count 2x24V" or "TM PosInput 2" and the use of the two DQs for faster reactions. Additional DQs of standard DQs can be switched depending on the encoder value from the application program.
	CAM_CTRL	
	CAM_DIAG	
	PID_FM	Check whether the instructions in your program can be replaced by the technology objects "PID_Compact", "PID_3Step" or PID basic functions.
	FUZ_355	
	FORCE355	
	READ_355	
	CH_DIAG	
	PID_PAR	
	CJ_T_PAR	
	FMT_PID	
	FMT_PAR	
	FMT_CJ_T	
	FMT_DS1	
	FMT_TUN	
	FMT_PV	

Instruction group	Instruction which cannot be migrated	Notes
300C functions	ANALOG_300C	Check whether the instructions in your program can be replaced by the instructions from the "Motion Control" group.
	DIGITAL_300C	
	COUNT_300C	Check whether the instructions in your program can be replaced by the "High_Speed_Counter" technology object in connection with counter modules.
	FREQUENC_300C	
	PULSE_300C	Check whether the instruction in your program can be replaced by the "CTRL_PWM" instruction.
	SEND_RK_300C	The communication instructions of the S7-1500 do not support RK 512. If you need RK functionality, you must implement it yourself.
	FETCH_RK_300C	
	SERVE_RK_300C	
Communication with iSlave/iDevice	I_GET	The instructions cannot be replaced. For more assistance, please contact SIMATIC Customer Support.
	I_PUT	
	I_ABORT	
S7 communication	PRINT	Check whether the instructions in your program can be replaced by the instructions of S7 communication for S7-1500. The "CONTROL" and "C_CNTRL" instructions, for example, can be replaced by the "T_DIAG" instruction.
	START	
	STOP	
	RESUME	
	STATUS	
	USTATUS	
	CONTROL	
	C_CNTRL	
Open user communication	TCON_PAR	These instructions are not required for the Open User Communication in S7-1500. The connection parameters are set in the block properties.
	TADDR_PAR	
	TCP_conn_active	
	TCP_conn_passive	
	ISOonTCP_conn_active	
	SOonTCP_conn_passive	
	ISOonTCP_conn_CP_active	
	ISOonTCP_conn_CP_passive	
	UDP_local_open	
	UDP_rem_address_and_port	
Communications processor	Instructions of the SIMATIC NET CPs group Instructions of the PTP CP 340, PTP CP341, PTP CP 440, PTP CP 441 groups	You can find more detailed information on these instructions in the chapter "Migration of communication instructions (Page 206)".

Instruction group	Instruction which cannot be migrated	Notes
PROFINET / CBA	PN_IN	PROFINET/CBA communication is not supported by S7-1500. Use S7 communication or open user communication.
	PN_OUT	
	PN_DP	
MPI communication	X_SEND	MPI communication is not supported by S7-1500. Use S7 communication or open user communication.
	X_RCV	
	X_GET	
	X_PUT	
	X_ABORT	
Global data communication	GD_SND	Global data communication is not supported by S7-1500. Use S7 communication or open user communication.
	GD_RCV	
Point-to-point communication	P_PRINT	The print function is not supported by the point-to-point communication modules.
	P_PRINT341	The print function is not supported by the point-to-point communication modules.
	MODB_341	These blocks for CP 341/CP 441 are not supported by S7-1500. Use the instructions for MODBUS (RTU) instead. • Modbus_Comm_Load • Modbus_Master • Modbus_Slave
	MODB_441	
	S_MODB	These blocks for ET 200S 1SI are not supported by S7-1500.
	S_USST	
	S_USSR	
	S_USSI	
Teleservice	PG_DIAL	The MPI interface is not supported by S7-1500 as the default interface. The instruction is no longer available.
	AS_DIAL	The MPI interface is not supported by S7-1500 as the default interface. The instruction is no longer available.
	SMS_SEND	The MPI interface is not supported by S7-1500 as the default interface. Check whether the instruction in your program can be replaced by the "TMAIL_C" instruction.

Instruction group	Instruction which cannot be migrated	Notes
	AS_MAIL	The MPI interface is not supported by S7-1500 as the default interface. Check whether the instruction in your program can be replaced by the "TMAIL_C" instruction.
User library	PNIO_DIAG	The "PNIO_DIAG" instruction could be downloaded to a user library for S7-300/400 and used for diagnostic purposes. "PNIO_DIAG" has been replaced in S7-1500 by integrated diagnostics and is thus no longer needed. In S7-1500, the system diagnostics is enabled by default. You can use the system diagnostics without making any additional settings. If required, you can edit the settings for the system diagnostics in the device configuration of the S7-1500 on the "Properties > Diagnostics" tab. You can also configure the interrupts of the system diagnostics there. The interrupts are enabled by default. Following the migration, you can delete the diagnostic screens used in conjunction with "PNIO_DIAG" and the associated tags in the HMI Panel. Use the "System diagnostics display" control in your diagnostic screens instead.

Note

Support for instructions that cannot be migrated

If you need further assistance for instructions that cannot be migrated, please contact SIMATIC Customer Support.

In the Siemens Industry Online Support, you will find the latest FAQs on the topic "PLC migration" (http://support.automation.siemens.com/WW/view/en/67858106).

Migration of data block instructions

Procedure for migration of data block instructions (migration class 3)

S7-1500 provides a revised set of instructions of the "Data blocks" category. These instructions are replaced automatically during the migration. Afterwards you still have to make manual adaptations.

The table below provides an overview of the migration of data block instructions:

Instruction in S7-300/400	Instruction in S7-1500	Migration class
CREA_DBL	CREATE_DB	3
CREAT_DB	CREATE_DB	3
CREA_DB	CREATE_DB	3

Instruction in S7-300/400	Instruction in S7-1500	Migration class
DEL_DB	DELETE_DB	3
TEST_DB	ATTR_DB	3

NOTICE
Asynchronous processing The new instructions work asynchronously in S7-1500, which means their execution is spread across several programming cycles.

Manual adaptations after migration

The instructions have additional formal parameters in S7-1500 which are required for asynchronous execution, for example . You need to provide these additional parameters with appropriate actual parameters following migration.

The block number of the created DB must be between 60000 and 60999 in S7-1500. Therefore, adapt the "LOW_LIMIT" and "UP_LIMIT" parameters to the new number range.

You can find a detailed description of the parameters in the reference help for S7-1500.

See also:

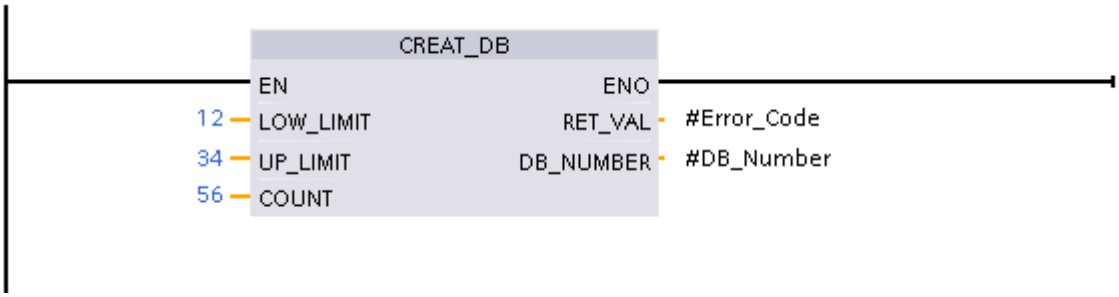
Data block functions (Page 4798)

Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 3150)

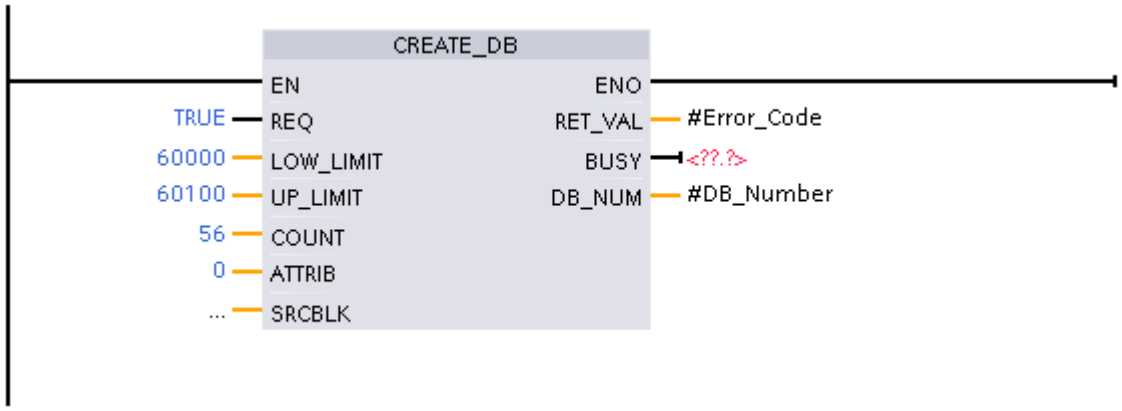
Example

The following example shows the migration of the "CREAT_DB" instruction to the "CREATE_DB" instruction.

In the first figure, you can see the call for "CREAT_DB" before migration:



In the second figure, you can see the call for "CREATE_DB" after migration:



- The "REQ" parameter was not available for "CREAT_DB" and was amended by the migration. It gets the default value "TRUE".
- The "LOW_LIMIT" and "UP_LIMIT" parameters were outside the permitted number range of 60000 to 60999. The values "12" and "34" were therefore marked as invalid during the migration. The permitted values "60000" and "60100" were then entered manually.
- The "ATTRIB" parameter was not available for "CREAT_DB" and was amended by the migration. It gets the default value "0". You can specify the properties of the DB to be created at this parameter.
- The "SRCBLK" parameter was not available for "CREAT_DB" and was amended by the migration. Here you can specify the data area whose contents are to be written to the DB to be generated.
- The "BUSY" parameter was not available for "CREAT_DB" and was amended by the migration. The parameter retains the signal "1" until the process for generating the data block is complete. To evaluate the signal, you need to interconnect the output parameter with an operand.

Migration of the instruction RDSYSST

Procedure for migration of RDSYSST (migration class 4)

The instruction "RDSYSST: Read system status list" is no longer available in S7-1500 as the CPUs of this product series do not provide system status lists. However, the option exists to use one of the following instructions to read out some of the information you received previously via the system status list:

- "RDREC (Page 4557)" (Extended instructions > Distributed I/O)
- "GET_DIAG (Page 4761)" (Extended instructions > Diagnostics)
- "GEO2LOG (Page 4808)" (Extended instructions > Diagnostics)
- "DeviceStates (Page 4753)" (Extended instructions > Diagnostics)
- "ModuleStates (Page 4756)" (Extended instructions > Diagnostics)

- "RALRM (Page 4564)" (Extended instructions > Distributed I/O)
- "LED (Page 4740)" (Extended instructions > Diagnostics)

Reading system status information in S7-1500

The following table shows which information about the system status can be read in S7-1500.

You will find the IDs that you have previously read with the RDSYSST instruction in the "SSL-ID" column. The two final digits are relevant. The "xy" placeholder varies depending on the module used.

The "Migration" column indicates whether the information is still available in S7-1500 and how you can read it out.

Partial list	SZL-ID	Migration
Module ID	W#16#xy11	This information can be read with the "RDREC" instruction. Read data record 16#AFF0 in the "INDEX" parameter.
CPU characteristics	W#16#xy12	This information is not currently available in S7-1500.
User memory areas	W#16#xy13	This information is not currently available in S7-1500.
System areas	W#16#xy14	This information is not currently available in S7-1500.
Block types	W#16#xy15	This information is not currently available in S7-1500.
Identification of one component	W#16#xy1C	This information can be read with the "RDREC" instruction for central modules and PROFINET IO. Read data record 16#AFF0 in the "INDEX" parameter.
Interrupt status	W#16#xy22	This information is not available in S7-1500.
Assignment of partial process images and OBs	W#16#xy25	This information is not available in S7-1500.
Communication: Status data	W#16#xy32	This information is not available in S7-1500.
H-CPU group information	W#16#xy71	Not relevant
Status of the module LEDs	W#16#xy74	This information can be read with the "LED" instruction. Read data record W#16#xy74 in the "LED" parameter.
Switched DP slaves in the H-system	W#16#xy75	Not relevant
DP master system information	W#16#xy90	This information is not available in S7-1500.
Module status information	W#16#xy91	This information can be read with the "GET_DIAG" or "ModuleStates" instruction.
Rack/station status information	W#16#xy92	This information can be read with the "GET_DIAG" or "DeviceStates" instruction.

Partial list	SZL-ID	Migration
Rack/station status information	W#16#xy94	This information can be read with the "GET_DIAG" or "DeviceStates" instruction.
Extended DP master system information	W#16#xy95	This information is not available in S7-1500.
Module status information PROFINET IO and PROFIBUS DP	W#16#xy96	This information can be read with the "GET_DIAG", "ModuleStates" or "DeviceStates" instruction.
Tool changer information (PROFINET IO)	W#16#xy9C	This information is not available in S7-1500.
Diagnostic buffer	W#16#xyA0	This information is not available in S7-1500.
Module diagnostic information (data record 0)	W#16#00B1	This information can be read with the "GET_DIAG" or "RDREC" instruction. Within the interrupt OB, you also get the diagnostic information using "RALRM".
Module diagnostic information (data record 1), physical address	W#16#00B2	This information can be read with the "RDREC" or "GEO2LOG" instruction. Within the interrupt OB, you also get the diagnostic information using "RALRM".
Module diagnostic information (data record 1), logical address	W#16#00B3	This information can be read with the "GET_DIAG" or "RDREC" instruction. Within the interrupt OB, you also get the diagnostic information using "RALRM".
Diagnostics data of a DP slave	W#16#00B4	This information can partially be read with the "ModuleStates" or "GET_DIAG" instruction. Within the interrupt OB, you also get the diagnostic information using "RALRM". Some of the information is not available in S7-1500.

Note

Support on information that cannot be read

If you need further assistance on information that cannot be read, please contact SIMATIC Customer Support.

In the Siemens Industry Online Support, you will find the latest FAQs on the topic "PLC migration" (http://support.automation.siemens.com/WW/view/en/67858106).

Migration of table instructions

Example for migration of "ATT: Add value to table"

Procedure for migration of table instructions (migration class 4)

The instructions from the "Table instructions" group are no longer available in S7-1500 and cannot be replaced during the migration because they are based on pure absolute addressing.

However, S7-1500 provides convenient functions that allow you to implement the instructions yourself. Below is an example for the implementation of the instruction "ATT": Add value to table".

Implementation

The following SCL program shows how to implement the "ATT" instruction. The function in this example has been implemented as an FC in SCL.

The following table shows the declaration of the tags used:

Operand		Declaration	Data type	Description
val		Input	INT	Entered value
tab		InOut	PLC data type "Table10"	Table with 10 rows
	x	InOut	ARRAY[0..9] of INT	Each ARRAY element represents a row. An ARRAY of INT is used in the example. However, you can create ARRAYs with any data type. An ARRAY of STRUCT or an ARRAY of a PLC data type is also possible.
	lng	InOut	INT	Length of the buffer. The value depends on the number of ARRAY elements of "tab.x".
	first	InOut	INT	First written value
	last	InOut	INT	Last written value
h		Temp	INT	Auxiliary tag
Ret_Val			BOOL	Function value

The following table shows the FC "Attend":

The function writes the value of the input parameter "#val" to the last row of the table.

SCL

```
#h:=(#tab.last+1) MOD #tab.lng;    // increment index
IF #h <> #tab.first THEN          // Scan to see if indices are overtaking each
other
   #tab.x[#tab.last] := #val;      //
Write value
   #tab.last := #h;                // Save index of the next free element
   #Attend:= false;
ELSE
  #Attend:= true;  // Error >> The table is full; the value cannot be entered
END_IF;
```

Example for migration of "FIFO: Output first value of the table"

Procedure for migration of table instructions (migration class 4)

The instructions from the "Table instructions" group are no longer available in S7-1500 and cannot be replaced during the migration because they are based on pure absolute addressing. However, S7-1500 provides convenient functions that allow you to implement the instructions yourself. Below is an example for the implementation of the instruction "FIFO": Output first value of the table".

Implementation

The following SCL program shows how to implement the "FIFO" table function. The function in this example has been implemented as an FC in SCL.

The following table shows the declaration of the tags used:

Operand		Declaration	Data type	Description
val		Output	INT	Value returned
tab		InOut	PLC data type "Table10"	Table with 10 rows
	x	InOut	ARRAY of INT [0..9]	Each ARRAY element represents a row. An ARRAY of INT is used in the example. However, you can create ARRAYs with any data type. An ARRAY of STRUCT or an ARRAY of a PLC data type is also possible.
	lng	InOut	INT	Length of the buffer. The value depends on the number of ARRAY elements of "tab.x".
	first	InOut	INT	First written value
	last	InOut	INT	Last written value
h		Temp	INT	Auxiliary tag
Ret_Val			BOOL	Function value

The following table shows the FC "Fifo":

The function returns the first value written to the table (first in, first out). The function value "true" indicates that the table is empty.

SCL

```
#h:=(#tab.first+1) MOD #tab.lng; // increment index
  IF #h <> #tab.last THEN        // Scan to see if indices are
overtaking each other
    #tab.first := #h;   ·        // Save index of the next value
    #val := #tab.x[#h];      //
Output desired value
    #Fifo:= false;
  ELSE
```

```
SCL
    #Fifo:= true; // Error >> The table is empty; no value is
output
    #val := #tab.x[#tab.first]; //
Output last valid value
  END_IF;
```

Example for migration of "LIFO: Output last value in the table"

Procedure for migration of table instructions (migration class 4)

The instructions from the "Table instructions" group are no longer available in S7-1500 and cannot be replaced during the migration because they are based on pure absolute addressing. However, S7-1500 provides convenient functions that allow you to implement the instructions yourself. Below is an example for the implementation of the instruction "LIFO": Output last value in the table".

Implementation

The following SCL program shows how to implement the "LIFO" table function. The function in this example has been implemented as an FC in SCL.

The following table shows the declaration of the tags used:

Operand		Declaration	Data type	Description
val		Output	INT	Value returned
tab		InOut	PLC data type "Table10"	Table with 10 rows
	x	InOut	ARRAY of INT [0..9]	Each ARRAY element represents a row. An ARRAY of INT is used in the example. However, you can create ARRAYs with any data type. An ARRAY of STRUCT or an ARRAY of a PLC data type is also possible.
	lng	InOut	INT	Length of the buffer. The value depends on the number of ARRAY elements of "tab.x".
	first	InOut	INT	First written value
	last	InOut	INT	Last written value
h		Temp	INT	Auxiliary tag
Ret_Val			BOOL	Function value

The following table shows the FC "Lifo":

The function returns the last value written in the table (last in, first out). The function value "true" indicates that the table is empty.

SCL

```
#h:=(#tab.last-1) MOD #tab.lng;           // decrement index
  IF #h<0 THEN #h:=#h + #tab.lng; END_IF; // allow only
positive indices
   IF #h <> #tab.first THEN               // Scan to see if
indices are overtaking each other
    #tab.last := #h;                      // Save index of
the next value
    #val := #tab.x[#h];            //
Output desired value
    #Lifo:= false;
  ELSE
    #Lifo:= true;  // Error >> The table is empty; no value
is output
    #val := #tab.x[#tab.last];           //
Output last valid value
  END_IF;
```

Migration of communication instructions

SIMATIC NET CP/CM

Migrating program blocks

Introduction

S7-300/400 uses specific program blocks for communication functions via SIMATIC NET CPs. S7-1500 provides a revised set of instructions for these communication functions.

The following tables provide an overview of the migration of the program blocks used in S7-300/400.

You can find more detailed information about the specification in the "Class" column in chapter Information on migrating instructions (Page 190).

Program blocks for industrial Ethernet / PROFINET

Communication service / functional area	Type of instruction	Library		Instruction in S7-1500	Class
		CP 300	CP 400		
SEND / RECEIVE (Open communication services)	AG_SEND Migration AG_SEND/ AG_LSEND (Page 209)	x	x	T_SEND	3
	AG_LSEND Migration AG_SEND/ AG_LSEND (Page 209)	x	x	T_SEND	3
	AG_SEND (via UDP) Migration AG_SEND/ AG_LSEND (Page 209)	x	x	TUSEND	3
	AG_SEND (e-mail) Migration AG_SEND/ AG_LSEND (Page 209)	x	x	TMAIL_C	3
	AG_RECV Migration AG_RECV/ AG_LRECV (Page 210)	x	x	T_RCV	3
	AG_LRECV Migration AG_RECV/ AG_LRECV (Page 210)	x	x	T_RCV	3
	AG_LOCK	x	x	-	4
	AG_UNLOCK	x	x	-	4
	AG_CNTRL	x	x	T_DIAG, T_RESET, TCON, TDISCON	3
Programmed communication connections	IP_CONFIG	x	x	T_CONFIG	3
FTP	FTP_CMD	x	x	FTP_CMD	3
PROFINET IO	PNIO_SEND	x		-	4 *)
	PNIO_RECV	x		-	4 *)
	PNIO_RW_REC	x		-	4 *)
	PNIO_ALARM	x		-	4 *)

*) Other access mechanisms should be used.

Program blocks for PROFIBUS

Communication service / functional area	Type of instruction	Library		Instruction in S7-1500	Case
		CP 300	CP 400		
SEND / RECEIVE (Open communication services)	AG_SEND Migration AG_SEND/ AG_LSEND (Page 209)	x	x	T_SEND	3
	AG_LSEND Migration AG_SEND/ AG_LSEND (Page 209)	-	x	T_SEND	3
	AG_RECV Migration AG_RECV/ AG_LRECV (Page 210)	x	x	T_RCV	3
	AG_LRECV Migration AG_RECV/ AG_LRECV (Page 210)	-	x	T_RCV	3
PROFIBUS DP	DP_SEND	x	-	-	4 *)
	DP_RECV	x	-	-	4 *)
	DP_DIAG	x	-	-	4 *)
	DP_CTRL	x	-	-	4 *)

*) Other access mechanisms should be used.

Additional formal parameters

The instructions have additional formal parameters in S7-1500 which are required for asynchronous execution, for example . You need to provide these additional parameters with appropriate actual parameters following migration.

You can find a detailed description of the parameters in the reference help for S7-1500.

See also:

Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 3150)

Example

You can find a basic example of the representation of a migrated instruction in the chapter Migration of data block instructions (Page 198)

Migration AG_SEND/AG_LSEND

Migration of the call parameters - communication via ISO Transport, ISO-on-TCP, TCP

Table 4-1 Migration AG_SEND/AG_LSEND

AG_SEND / AG_LSEND		T_SEND		Description / notes
Parameter	Type	Parameter	Type	
ACT	BOOL	REQ	BOOL	
ID	INT	ID	CONN_OUC (WORD)	
LADDR	WORD	--	--	
SEND	ANY	DATA	VARIANT	
--	--	ADDR	VARIANT	
LEN	INT	LEN	UINT	
DONE	BOOL	DONE	BOOL	
--	--	BUSY	BOOL	
ERROR	BOOL	ERROR	BOOL	
STATUS	WORD	STATUS	WORD	

Additional steps

- Configure the corresponding connection type with the relevant address parameters.

Migration of the call parameters - communication via UDP

Table 4-2 Migration AG_SEND/AG_LSEND

AG_SEND / AG_LSEND		TUSEND		Description / notes
Parameter	Type	Parameter	Type	
ACT	BOOL	REQ	BOOL	
ID	INT	ID	CONN_OUC (WORD)	
LADDR	WORD	--	--	
SEND	ANY	DATA	VARIANT	
--	--	ADDR	VARIANT	
LEN	INT	LEN	UINT	
DONE	BOOL	DONE	BOOL	
--	--	BUSY	BOOL	
ERROR	BOOL	ERROR	BOOL	
STATUS	WORD	STATUS	WORD	

Additional steps

- Configure the corresponding connection type with the relevant address parameters.

Migration of the call parameters - communication via E-MAIL

Table 4-3 Migration AG_SEND/AG_LSEND

AG_SEND / AG_LSEND		TMAIL_C		Description / notes
Parameter	Type	Parameter	Type	
ACT	BOOL	REQ	BOOL	
ID	INT	--	--	
LADDR	WORD	--	--	
SEND	ANY	TO_S	STRING	
SEND	ANY	CC	STRING	
SEND	ANY	SUBJECT	STRING	
SEND	ANY	TEXT	STRING	
SEND	ANY	ATTACHMENT	VARIANT	
--	--	ATTACHMENT _NAME	STRING	
--	--	MAIL_ADDR_PARAM	VARIANT	
LEN	INT	--	--	
DONE	BOOL	DONE	BOOL	
--	--	BUSY	BOOL	
ERROR	BOOL	ERROR	BOOL	
STATUS	WORD	STATUS	WORD	

Additional steps

- Configure the corresponding connection type with the relevant address parameters.

Migration AG_RECV/AG_LRECV

Migration of call parameters

Table 4-4 Migration AG_RECV/AG_LRECV

AG_RECV / AG_LRECV		TRCV / TURCV		Description / notes
Parameter	Type	Parameter	Type	
		EN_R	BOOL	
ID	INT	ID	CONN_OUC (WORD)	
LADDR	WORD	--	--	

AG_RECV / AG_LRECV		TRCV / TURCV		Description / notes
Parameter	Type	Parameter	Type	
--	--	LEN	UDINT	
RECV	ANY	DATA	VARIANT	
--	--	ADDR	VARIANT	
LEN	INT	LEN	UINT	
NDR	BOOL	NDR	BOOL	
--	--	BUSY	BOOL	
ERROR	BOOL	ERROR	BOOL	
STATUS	WORD	STATUS	WORD	

Additional steps

- Configure the corresponding connection type with the relevant address parameters.

Point-to-point CP/CM

Migrating point-to-point program blocks

Introduction

S7-300/400 uses specific program blocks for communication functions via point-to-point CPs.

S7-1500 provides a revised set of instructions for these communication functions.

During migration, you are asked whether you want to use the revised set of instructions for the communication modules of the S7-1500 or the program blocks for the communication processors of the S7-300/400.

The table below provides an overview of the migration of the program blocks used in S7-300/400 on instructions of the S7-1500.

You can find more detailed information about the specification in the "Class" column in chapter Information on migrating instructions (Page 190).

Program blocks for point-to-point communication

Communication service / functional area	Block	Library			Instruction in S7-1500	Class
		CP 300	CP 400	ET 200		
Blocks for CP 340	P_RCV	x	-	-	Receive_P2P	3 [1]
	P_SEND	x	-	-	Send_P2P	3 [1]
	P_RESET	x	-	-	Receive_Reset	3 [1]
	V24_STAT_340	x	-	-	Signal_Get	3 [1]

Communication service / functional area	Block	Library			Instruction in S7-1500	Class
		CP 300	CP 400	ET 200		
	V24_SET_340	x	-	-	Signal_Set	3 [1]
	P_PRINT	x	-	-	-	4
Blocks for CP 341	P_RCV_RK	x	-	-	Receive_P2P	3 [1]
	P_SND_RK	x	-	-	Send_P2P	3 [1]
	V24_STAT	x	-	-	Signal_Get	3 [1]
	V24_SET	x	-	-	Signal_Set	3 [1]
	P-PRT341	x	-	-	-	4
Blocks for CP 440	RECV_440	-	x	-	Receive_P2P	3 [1]
	SEND_440	-	x	-	Send_P2P	3 [1]
	RES_RECV	-	x	-	Receive_Reset	3 [1]
Blocks for CP 441	V24_STAT_441	-	x	-	Signal_Get	3 [1]
	V24_SET_441	-	x	-	Signal_Set	3 [1]
Blocks for Modbus	MODB_341	x	-	-	-	4
	MODB_441	-	x	-	-	4
Blocks for ET 200S 1SI	S_RCV	-	-	x	Receive_P2P	3 [1]
	S_SEND	-	-	x	Send_P2P	3 [1]
	S_VSTAT	-	-	x	Signal_Get	3 [1]
	S_VSET	-	-	x	Signal_Set	3 [1]
	S_XON	-	-	x	Port_Config	3 [1]
	S_RTS	-	-	x	Port_Config	3 [1]
	S_V24	-	-	x	Send_Config	3 [1]
	S_MODB	-	-	x	-	4
	S_USST	-	-	x	-	4
	S_USSR	-	-	x	-	4
	S_USSI	-	-	x	-	4
Blocks for compact CPUs S7 300	SEND_PTP	x	-	-	Send_P2P	3 [1]
	RCV_PTP	x	-	-	Receive_P2P	3 [1]
	RES_RCVB	x	-	-	Receive_Reset	3 [1]

[1] You can find information on migrating this instruction at this link (Page 190).

Additional formal parameters

The instructions have additional formal parameters in S7-1500. You need to provide these additional parameters with appropriate actual parameters following migration.

The Siemens Industry Online Support offers a detailed description of the parameters under FAQs on the topic "PLC migration" (http://support.automation.siemens.com/WW/view/en/67858106).

4.2.4.3 Migration of LAD/FBD programs to S7-1500

Information on migrating LAD/FBD programs

---(SAVE): Save RLO to BR memory

The instruction is not available in S7-1500. Migration handles the instruction as follows:

- If the "---(SAVE)" instruction was used in the last network of a block, it is replaced by the "---(RET)" instruction with the parameter "RLO".
- If the "---(SAVE)" instruction was used in a network that is not the last network of the block, it cannot be replaced. Change the program manually. For example, save the RLO in a PLC tag, a DB tag or a local tag and return it to the calling block at the end of the block with the --(RET) instruction.

IEC timer and IEC counter with EN/ENO interconnection

The instructions have been adapted to the IEC 1131-3 standard for S7-1500. They can now be connected with the current path via the "IN" parameter, and therefore no longer have the "EN" and "ENO" parameters. Migration reports an error if the reference program includes an IEC timer or IEC counter with preceding logic that is evaluated using the "EN" input parameter.

Insert a jump instruction between the preceding logic and the IEC timer or IEC counter to call the IEC timer or IEC counter based on the RLO. The following jump instructions are available:

- ---(JMP): Jump if RLO = 1
- ---(JMPN): Jump if RLO = 0

---(CALL) Call block without parameters

The instruction is not available in S7-1500. Migration adapts the program so that the call is implemented via a box.

Master control relay

The master control relay is not available on S7-1500. Migration reports an error. Change the program manually. Use temporary tags, for example, to conditionally execute instructions or networks.

Language change within a block

The contents of the registers, accumulators and the status word are only available within STL networks. When an LAD or FBD network follows an STL network, you cannot access the register contents that were previously set in STL from the LAD or FBD network. However, the register contents are again available in a downstream STL network. The result of logic operation is an exception: It is set to "undefined" at a language change and is no longer available in downstream networks.

See also: Passing values using registers in case of language change (Page 219)

See also

--(RET): Return (Page 3380)

RET: Return (Page 3680)

EN/ENO mechanism (Page 2447)

Partially qualified addresses in LAD/FBD

Introduction

The addressing of DB tags without specification of the DB name is referred to as partially qualified addressing. Partially qualified addressing is not supported in S7-1500 for LAD/FBD. The "---(OPN)" and "---(OPNI)" instructions, which were often used in conjunction with partially qualified addressing, are not supported in S7-1500 either.

Migration of partially qualified addresses

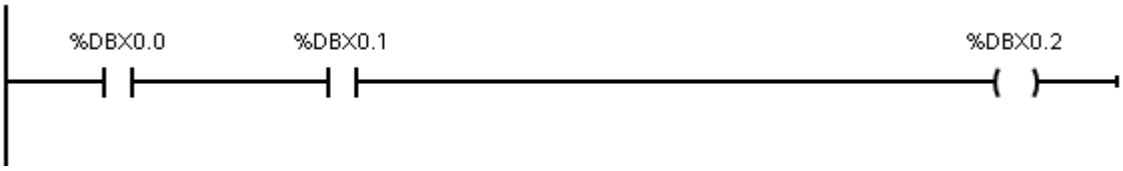
Migration handles partially qualified addresses as follows, if possible:

- If the data block can be uniquely determined, migration replaces partially qualified addresses with full addresses and specifies the DB.
- If the DB is specified indirectly by the data block register, the data block cannot be clearly determined. In this case, migration adds a parameter of the data type "DB_Any" to the block interface. The data block name is passed to this parameter. Migration then replaces all partially qualified addresses with addresses using the following syntax: `#<Parameter name>.%<absolute address>.`

Example

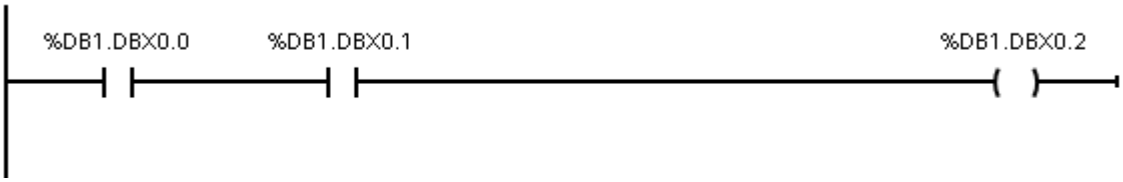
The following example shows migration of partially qualified addresses. In the first figure, you can see the program before migration:

Network 1: Tags are addressed partially qualified.



The following figure shows the program after migration:

Network 1: The DB containing the tags could be determined unambiguously. The program was migrated automatically. The tags in "DB1" are addressed fully qualified.



Note

Migration of partially qualified block parameters

The migration also converts partially qualified DB parameters into fully qualified parameters. Note that the type of parameter transfer to the called block can change because of this: The called block may no longer access the actual parameter directly using the fully qualified address but works with a copy that was transferred during the block call instead.
Make sure that the semantics of the migrated program matches that of the reference program. Additional information is available under: Access to block parameters during program execution (Page 2190)

If this adaptation is not desired, you can, for example, use a parameter with structured data type rather than an elementary block parameter.
Define PLC data type (UDT) as formal parameter and transfer a tag of this type or a DB that was derived from the PLC data type (UDT).

See also

Addressing operands (Page 2209)

Access to status bits in LAD/FBD

Introduction

To achieve the highest possible program execution performance on CPUs of the S7-1500 series, values are only passed between blocks by means of the block interface, by using global data blocks or PLC tags. It is not possible to pass values by using the status word in LAD and FBD.

Migration of the instructions "--| |--: Get status bit" / "--|/|--: Get negated status bit"

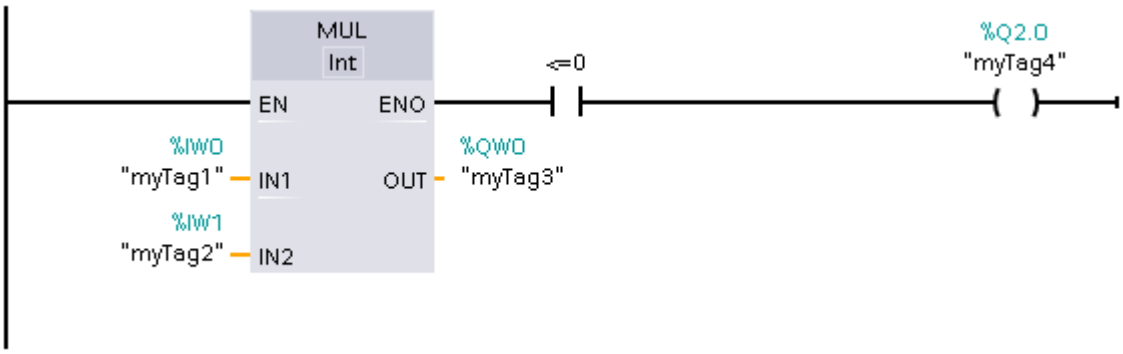
The instructions are not available in S7-1500. The migration distinguishes between the following cases:

- If a status bit query is placed directly after a mathematical instruction in the same network, a comparator is used instead.
- In all other cases the program is faulty after migration. You have to change your program in this case. For example, use a query of the ENO output instead of a query of the OV status bit. If you link the negation of several ENO queries with OR, you can replace the query of the OS status bit.

Example

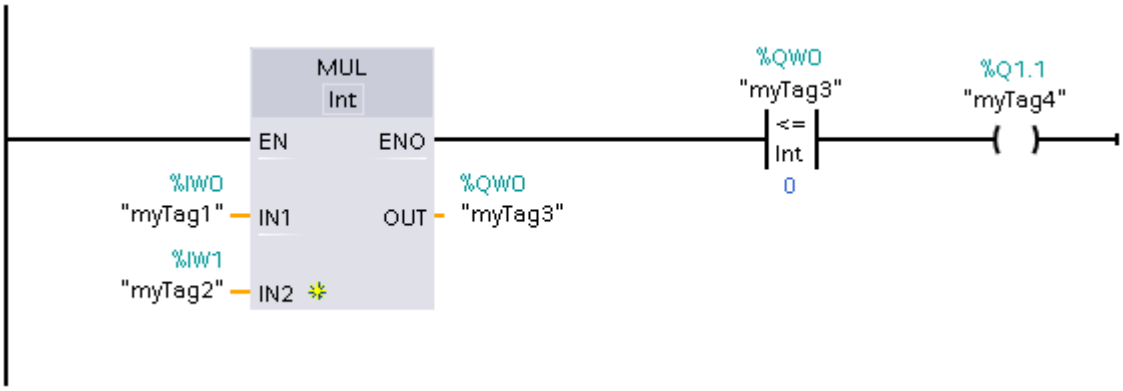
The following example shows the migration of a status bit query. In the first figure, you can see the program before migration.

Network 1: The "<=0" instruction queries the status bit "A1". It indicates whether the result of multiplication is less than or equal to zero.



The following figure shows the program after migration.

Network 1: The program was able to be migrated automatically. The "<=0" instruction was removed. Instead, the result of the multiplication is queried by using a comparator with the "myTag3" tag.



4.2.4.4 Migration of STL programs to S7-1500

Information on migrating STL programs

Passing values using registers and status words

To achieve the highest possible program execution performance on CPUs of the S7-1500 series, values are only passed between blocks by means of the block interface, by using global data blocks or PLC tags.

There is no way to pass values using registers (for example, accumulators, AR1, AR2, DB, DI) or the status word in LAD and FBD. In STL, values can be passed with some restrictions. Keep in mind that program execution slows down when you use these areas to pass values between various blocks.

The following rules apply for STL:

- The contents of the registers, accumulators and the status word are only available within STL networks. When an LAD or FBD network follows an STL network, you cannot access the register contents that were previously set in STL from the LAD or FBD network. However, the register contents are available again in a downstream STL network.
  The RLO bit is an exception: It is set to "undefined" at a language change and is no longer available in downstream networks.
- The values from the registers, accumulators and the status word are never transferred to called blocks. The instructions "CC" and "UC" are the only exceptions. If you use "UC" or "CC" and want to pass parameters to the called block by using registers, the status word or accumulators, you have to enable the "Parameter passing via registers" option in the properties of the called block. Note that this option is only available for STL blocks with standard access and the block can have no formal parameters. When the option is enabled, you can transfer register contents between blocks. The RLO bit is an exception here as well: It is set to "undefined" when the block is exited and is no longer available after a block call.
- To pass an error message to the calling block, you can use the BR bit. You first need to store the error message in the calling block with the instruction "SAVE" in BR bit. You can then read the BR bit in the calling block.
- The data block register DB is set to "0" after each access to a data block with specification of a fully qualified address (for example, `%DB10.DBW10`). Subsequent partially qualified access leads to an error during compilation.
- If you symbolically address a local formal parameter (e.g. with the instruction L #myIn) from the block interface of an FB in S7-1500, you always access the data block that you specified as an instance in the block call. Although the OPN DI, L AR2, +AR2, TDB, TAR instructions change the contents of the DI or address register, the registers are not evaluated when addressing local formal parameters.

You can find programming examples in the sections below.

Master control relay

The master control relay is not available on S7-1500. Migration reports an error. Change the program manually. Formulate conditions, for example, in the form of block parameters to conditionally execute instructions or networks.

LEAVE and ENT

The "LEAVE" and "ENT" instructions are not available in S7-1500 because only two accumulators are available. Migration reports an error. Change the program manually. Use temporary tags, for example, to store intermediate results.

Block parameters of the "Block_DB" parameter type

The "Block_DB" parameter type is not available in S7-1500. Migration changes this parameter and assigns it the "DB_Any" data type instead.

It is not possible to match library instructions with the call of an instance in the form of a parameter of the "DB_Any" type in S7-1500.

The following example shows the call of a library block with a variable instance on a CPU of the S7-300/400 family. This sequence cannot be migrated to S7-1500.

STL	Explanation
CALL GET, #myBlock_DB	/ / The library block "GET" is called with the instance data block that is currently applied at the "myBlock_DB" block parameter.
REQ := <Operand>	
ID := <Operand>	
NDR := <Operand>	
ERROR := <Operand>	

See also

STL Basics (Page 2650)

Basic information on the status word (Page 2466)

Logic operation sequences and jumps

Introduction

In S7-300/400, it is possible to place jump labels in STL at any instruction to enable a program-controlled jump to the labeled position. However, if you define the jump destination within a logical program unit in which the RLO does not have a defined state, unexpected results can occur during program execution. The CPU may switch to STOP state. You must therefore take steps during program creation to ensure that jump labels are used only at the start of a logical program sequence.

In S7-1500, a compilation error is output if programs contain jumps from or to logic operation sequences. This allows you to avoid runtime errors during program execution.

The following example shows a jump to a logic operation sequence. At jump label n_OK in line 6, the result of logic operation does not have a defined state:

```
1        A     #Motor_Enabled
2        AN    #Motor_EmergencyStop
3        JC    n_OK
4        =     #Motor_Start
5        AN    #Motor_SpeedOK
6  n_OK: AN    #Motor_BrakesEnabled
7        =     #Motor_Stop
```

Migration of jumps to logic operation sequences

The migration does not change the jump labels of your STL program. However, an error is reported when compiling if jump labels within logic operation sequences are detected. Change the program. Use the SET or CLR instructions in order to label the start of a new “logical unit” and to set the result of logic operation to a defined state.

The following example shows how you can use "SET" to change the program following the migration:

```
1        A     #Motor_Enabled
2        AN    #Motor_EmergencyStop
3        JC    n_OK
4        =     #Motor_Start
5        AN    #Motor_SpeedOK
6        AN    #Motor_BrakesEnabled
7        =     #Motor_Stop
8        JU    End
9  n_OK: SET
10       AN    #Motor_BrakesEnabled
11       =     #Motor_Stop
12 End:  NOP 0
```

Passing values using registers in case of language change

Introduction

The contents of the registers, accumulators and the status word are only available within STL networks. When an LAD or FBD network follows an STL network, you cannot access the register contents that were previously set in STL from the LAD or FBD network. However, the register contents are available again in a downstream STL network.

The RLO bit is an exception: It is set to "undefined" at a language change and is no longer available in downstream networks.

Migration of value passing using registers in case of language change

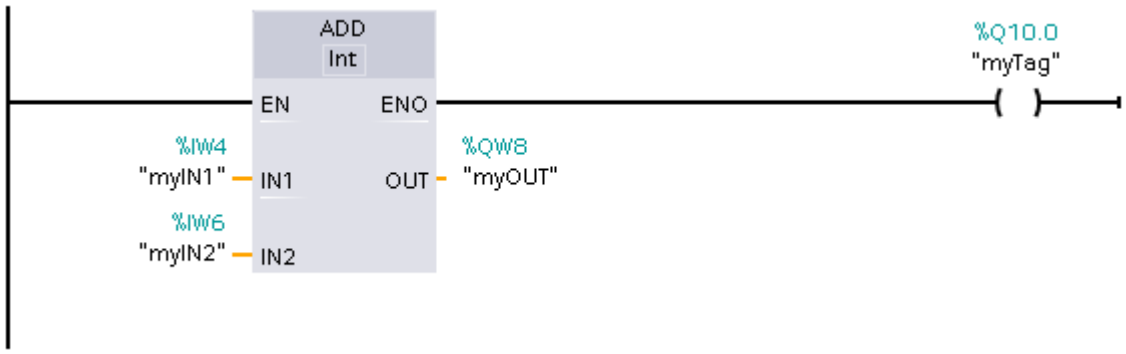
When a register is accessed in a migrated LAD or FBD network, an error is reported during compilation. Change the program in such a way that registers are only set and read in STL networks.

Example

The following example shows the migration of an accumulator access. In the first two figures, you can see the program before migration:

Network 1: The "myIN1" operand is downloaded to accumulator 1. Then "myIN2" is downloaded to accumulator 1, which transfers "myIN1" to accumulator 2. Now both values are

added together. The result is stored in accumulator 1 and from there it is assigned to the "myOUT" operand.



Network 2: The "myIN3" operand is downloaded to accumulator 1. The existing "myOUT" operand is thereby transferred to accumulator 2 and can be immediately added to the "myIN3" operand. The "myOUT" operand does not need to be explicitly downloaded again.

```
1    L    "myIN3"      %IW12
2    +I
3    T    "myResult"   %IW14
4
```

After the migration, network 2 is faulty because values are no longer written to the accumulators by the LAD network. The program must be adapted manually so that both values are explicitly loaded to the accumulators in STL.

The following figure shows network 2 after correction of the error.

Network 2: The two values are explicitly loaded to the accumulator in STL before they are added together.

```
1    L    "myIN3"      %IW12
2    L    "myOUT"      %QW8
3    +I
4    T    "myResult"   %QW14
5
```

See also

Basic information on the status word (Page 2466)

STL Basics (Page 2650)

Passing values using registers or the status word with a block call

Migration of passing values using registers or the status word with a block call

The values from registers, accumulators and the status word are set to "0" or given the status "undefined" at a block change. This means they cannot be passed to called blocks.

The instructions "CC" and "UC" are the only exceptions. If you use "UC" or "CC" and want to pass parameters to the called block by using registers, the status word or accumulators, you have to enable the "Parameter passing via registers" option in the properties of the called block. Note that this option is only available for STL blocks with standard access and the block can have no formal parameters. When the option is enabled, you can transfer register contents between blocks. The RLO bit is an exception: It is always set to "undefined" at a block transition and is no longer available after a block call.

To pass an error message to the calling block, you can use the BR bit. You first need to store the error message in the calling block BR bit. To do this, use the instructions "SAVE" or "JNB", for example. You can then read the BR bit in the calling block.

An error is reported if you access register contents that were set in the called block after a block call. In this case, change your program. Use the tags in data blocks or PLC tags, for example, to return values to the calling block.

Examples

The following example shows the changes you need to make to your program in order to pass values to a calling block via registers.

In the first table, you can see the program prior to migration:

STL	Explanation
CALL "MyFB", "MyFB_DB"	The RLO of the "MyFB" block is assigned to the "MyBit" operand after execution.
= #MyBit	

The second table shows you which changes you need to make to the program

STL	Explanation
CALL "MyFB", "MyFB_DB"	In the called "MyFB" block, write the currently pending RLO to the BR bit with the instruction "SAVE" at any location.
A BR	The BR bit is read in the calling block.
= #MyBit	The value of the BR bit is assigned to the "MyBit" operand.

See also

Basic information on the status word (Page 2466)

Parameter assignment via registers during block call in STL (Page 2652)

Basics of block access (Page 2175)

Protecting blocks (Page 2929)

STL Basics (Page 2650)

Passing values using registers with the CC and UC instructions

Introduction

In S7-300/400, block calls can be programmed using the instructions UC and CC. The parameters are not passed to the calling block via the interface, but rather via registers, for example, the AR1, AR2, DB, DI, accumulators or the status word.

These calls slow down program execution and can therefore no longer be set as the default in S7-1500. The CALL instruction is used instead in S7-1500. But the CALL instruction cannot be used to implement indirect block calls. If you wish to use UC or CC for indirect block calls, you need to select the "Parameter passing via registers" option in the properties of the called block. It is then possible for you to transfer register contents between different blocks. The RLO bit is an exception: It is set to "undefined" at a block transition and is no longer available after a block call.

Note that this option is only available for STL blocks with standard access and the block can have no formal parameters. When the option is enabled, you can transfer register contents between blocks. The RLO bit is an exception here as well: It is set to "undefined" at a block transition and is no longer available after a block call.

It is no longer necessary to use indirect block calls for the creation of know-how protected library elements. The TIA Portal supports you here with an automatism: Any number conflicts that occur with blocks that are already in the user program are automatically resolved when a block is inserted from a library.

Migration of block calls via "UC" or "CC"

The migration handles block calls via "UC" or "CC" as follows:

- The "UC FC" instruction with specification of a block number is replaced by the "CALL" instruction.
- The "CC FC" instruction with specification of a block number is replaced by the "CALL" instruction. A jump instruction is also added that implements the conditional call.
- The "UC FC" and "CC FC" instructions with indirect specification of the block number remain unchanged.
- The "UC FB" and "CC FB" instructions with indirect specification of a block number remain unchanged.

Example

The following example shows migration of block calls via "UC".

In the first table, you can see the program prior to migration:

STL
UC FC 10
UC FC[#temp0]
UC FB 10
UC FB [#temp0]

In the following table, you can see the program after migration:

STL	Explanation
CALL FC 10	
UC FC[#temp0]	The "Parameter passing via registers" option must be set at the called block.
UC FB 10	The "Parameter passing via registers" option must be set at the called block.
UC FB [#temp0]	The "Parameter passing via registers" option must be set at the called block.

See also

Basic information on the status word (Page 2466)

Parameter assignment via registers during block call in STL (Page 2652)

STL Basics (Page 2650)

Fully qualified addresses in STL

Introduction

The addressing of DB tags with specification of the DB name or DB number is referred to as fully qualified addressing. The data block register is set to "0" after each access to the data block with specification of a fully qualified address. If you want to access the DB register again after fully qualified access, you must first re-assign a value using the OPEN DB command.

Migration of fully qualified addresses

Migration inserts the "OPN" instruction after fully qualified access, if needed, to reload the current data block into the data block register.

Example

The following example shows migration of a fully qualified address.

In the first table, you can see the program prior to migration:

STL	Explanation
OPN "MyDB"	"MyDB" is loaded to the data block register.

STL	Explanation
L %DBW1	Data elements from "MyDB" are addressed partially qualified.
L "Global_DB".Data1	The elements from a global DB are then addressed fully qualified. "Global_DB" is loaded implicitly to the data block register by this step.
L "Global_DB".DBW2	
T DBW[AR1, P#0.0]	A subsequent access addresses the "Global_DB", because it is in the DB register.

In the following table, you can see the program after migration:

STL	Explanation
OPN "MyDB"	
L "MyDB".DBW1	The partially qualified access is converted into a fully qualified access.
L "Global_DB".Data1	
L "Global_DB".DBW2	The DB register is reset after fully qualified access.
OPN "Global_DB"	The "Global_DB" must be loaded to the DB register again with OPN.
T DBW[AR1, P#0.0]	

See also

STL Basics (Page 2650)

Basic information on the status word (Page 2466)

Partially qualified addresses in STL

Introduction

The addressing of DB tags without specification of the DB name or DB number is referred to as partially qualified addressing. Partially qualified addressing accesses a defined value in the data block that is currently stored in the DB register.

The following restrictions apply to partially qualified addresses in S7-1500:

- In S7-1500, partially qualified addresses are only allowed if the DB register has been set explicitly in the current block. You set the DB register with the instruction "OPN", for example. In data blocks with standard access, you can only address tags partially qualified.
- The data block register is set to "0" with the block call in S7-1500. This means it is not possible to open a data block in a block and to address data elements from the data block in a subordinate block partially qualified. Set the DB register in the current block before you address a DB tag partially qualified.
- The DB data block register is also to "0" with each fully qualified access (for example, %DB10.DBW10). Set the DB register again after fully qualified access before you address a DB tag partially qualified.
- The use of partly qualified addresses slows down the processing of the program during runtime.

Migration of partially qualified addresses

Migration handles partially qualified addresses in STL as follows:

- The partially qualified address is converted into a fully qualified address, if possible.
- If the data block is opened in the calling code block and the DB cannot be clearly identified, migration inserts a parameter of the "DB_Any" data type in the interface of the called block. The data block name is passed to this parameter. Migration inserts the "OPN" instruction in the called block to open the data block.

Example

The following example shows the migration of partially qualified addresses.

In the first table, you can see the program prior to migration:

STL	Explanation
L DBW10 L DBW12 +I T DBW14	A data block is opened in the calling block and transferred to the DB register. In the current block, the values "DBW10", "DBW12" and "DBW14" are adopted from the data block that is currently open in the DB register.

In the following table, you can see the program after migration:

STL	Explanation
OPN "PlcmigTempBlockDB" L DBW10 L DBW12 +I T DBW14	Migration inserts a parameter of the data type "DB_Any" in the interface of the called block. The data block name is passed to this parameter. Migration inserts the "OPN" instruction in the called block to open the data block passed in the interface.

Note

Migration of partially qualified block parameters

The migration also converts partially qualified DB parameters into fully qualified parameters. Note that the type of parameter transfer to the called block can change because of this: The called block may no longer access the actual parameter directly using the fully qualified address but works with a copy that was transferred during the block call instead.
Make sure that the semantics of the migrated program matches that of the reference program. Additional information is available under: Access to block parameters during program execution (Page 2190)

If this adaptation is not desired, you can, for example, use a parameter with structured data type rather than an elementary block parameter.
Define PLC data type (UDT) as formal parameter and transfer a tag of this type or a DB that was derived from the PLC data type (UDT).

Examples:

```
CALL  "MyFC"
          InStruct :="DBofUDT"
```

or

```
    CALL  "MyFC"
          InStruct := "DBArrayOfUDT".a[#i]
```

See also

Parameter assignment via registers during block call in STL (Page 2652)

STL Basics (Page 2650)

Basic information on the status word (Page 2466)

Access to the instance DB in STL

Introduction

The "OPN DI" or "CDB" instructions place a data block in the DI register. In S7-300/400, the block opened there is considered an instance data block. The subsequent symbolic addressing of a local formal parameter from the block interface of an FB (IN, OUT, InOut, Static) no longer addresses the data block that was specified as an instance in the block call, but rather the data block that is located in the DI register. To subsequently symbolically address a local formal parameter from the block interface, the instance DB must be loaded to the DI register.

Even after the instructions "L AR2", "+ AR" and "TAR", it is not possible in S7-300/400 to symbolically address a formal parameter from the block interface, because the basis for the parameter access has been destroyed by the instructions.

This behavior has been corrected in S7-1500: If you symbolically address a local formal parameter (e.g. with the instruction L #myIn) from the block interface in S7-1500, you always access the data block that you specified as an instance in the block call. Although the OPN DI, L AR2, +AR2, TDB, TAR instructions change the contents of the DI or address register, the registers are not evaluated when addressing local formal parameters.

Migration of accesses to local tags in the instance DB

Migration does not change the programmed accesses. However, if you have changed the instructions OPN DI, L AR2, +AR2, TDB, TAR, etc. in the reference program, the semantics of the program may have changed.
In order to achieve the original semantics again, you must change the program manually. It is often no longer necessary to address data via registers. Instead, avail of the possibility to use ARRAYs within the instance DB and index the ARRAY elements indirectly.

Example 1

The following example shows the modified semantics of the instruction "OPN DI":

STL	Explanation
L #MyIn1 L #MyIn2 +I T #MyOut3	The "L" and "T" instructions address local tags that were declared in the block interface. The values of the tags are located in the instance DB that was specified in the block call.
OPNDI "MyDB" L #MyIn1 L #MyIn2 +I T #MyOut3	The global data block "MyDB" is written to the DI register. The "L" and "T" instructions address tags that were declared in "MyDB" prior to migration. After migration, the "L" and "T" instructions address tags that were declared in the block interface. The DI register is not evaluated for access in S7-1500.

Example 2

The following example shows the modified semantics of the instruction "LAR2":

STL	Explanation
L P#M23.0	

STL	Explanation
LAR2	Prior to migration, the basis for parameter access is destroyed with the assignment to AR2.
L #MyIn1	Prior to migration, access to "MyIn1" is not possible or results in an error. After migration, access to the formal parameter "MyIn1" is correctly executed.
L IW [AR2, P#1.0]	Access to %IW24.0
A [AR2, P#0.4]	Access to %M23.4

Example 3

The following example shows how a DB tag can be addressed indirectly in S7-1500 without using the address register:

STL	Explanation
OPN "MyDB" L #index LAR1 L %DBW [AR1 , P#10.0]	Prior to migration, area-internal indirect register addressing is used. A variable value (#index) is hereby loaded to the address register 1. Depending on this value, a data word from "MyDB" is loaded to accumulator 1.
L "MyDB".MyArray1[#index]	After migration, you can store the data values in "MyDB" in an ARRAY. The individual ARRAY elements can be variably indexed using the input parameter "#index".

4.2.4.5 Migration of SCL programs to S7-1500

Information on migrating SCL programs

Representation of SCL blocks after migration

SCL blocks that were created with S7-300/400 are automatically migrated to S7-1500, if their semantics are clearly interpretable. The program positions that are not clearly interpretable are marked with a question mark after the migration, which results in syntax errors. Check the semantics at these positions in the program after migration and correct them if necessary.

In some cases, the program code is changed by the migration. The corresponding section of code from the original program is then transferred to the migrated block as a comment section. This makes it easy for you to track the changes that were made by the migration.

See also

Basic information on migrating PLC programs (Page 180)

Information on migrating PLC programs (Page 184)

Indirect addressing in SCL

Introduction

The options for indirect addressing were standardized for all programming languages in S7-1500. Indirect addressing that was possible in S7-300/400 is therefore partially converted during the migration.

The following table provides an overview of the conversion options. The migration of individual language constructs is explained below in more detail.

Indirect addressing	S7-300/400	S7-1500
Indirect addressing of a DB	Data type "BLOCK_DB"	"DB_ANY" data type
Indirect addressing of DB tags	#block.%DBW3	"PEEK"/"POKE" instructions
	WORD_TO_BLOCK_DB(#myWord).%DBW3	"PEEK"/"POKE" instructions
	#block.DW(IDX := #myInt)	"PEEK"/"POKE" instructions
	WORD_TO_BLOCK_DB(#myWord).DW(IDX:=#myInt)	"PEEK"/"POKE" instructions
	%DB1.DW(IDX :=#myInt)	"PEEK"/"POKE" instructions
Indirect addressing of I/O	QB(IDX :=#myInt):P	"PEEK"/"POKE" instructions
Indirect addressing of PLC tags	IX(IDX :=#myInt1,Bit:=#myInt2)	"PEEK"/"POKE" instructions
	QB(IDX :=#myInt)	"PEEK"/"POKE" instructions
	MW(IDX :=#myInt)	"PEEK"/"POKE" instructions

Migration of the "BLOCK_DB" data type to "DB_ANY"

The "BLOCK_DB" data type is no longer available in S7-1500. Tags of this data type are converted to the "DB_ANY" data type by the migration. The "BLOCK_DB_TO_WORD" and "WORD_TO_BLOCK_DB" conversion functions are migrated to "UINT_TO_WORD(DB_ANY_TO_UINT)" and "UINT_TO_DB_ANY(WORD_TO_UINT)".

The following examples show the migration of the "BLOCK_DB" data type.

In the first table, you can see three program sections prior to migration:

SCL

```
#myBlock  //Data type  BLOCK_DB;
//...
#myWord :=BLOCK_DB_TO_WORD(#myBlock);
//...
#myBlockDB := WORD_TO_BLOCK_DB(#myWord);
```

In the second table, you can see three program sections after migration:

SCL

```
#myBlock  //Datentyp DB_Any;
//...
#myWord := UINT_TO_WORD(DB_ANY_TO_UINT(#myBlock));
//...
#myDBANY:= UINT_TO_DB_ANY(WORD_TO_UINT(#myWord));
```

Migration of indirect addressing of DB tags

Indirect specification of data blocks or DB tags is implemented with the "PEEK"/"POKE" instructions in S7-1500. The migration automatically converts them to indirect addressing.

The following examples show the migration of indirect addressing of DB tags.

In the first table, you can see some program sections prior to migration:

SCL

```
#myWord := %DB1.DW(IDX :=#myInt);
//...
#myBool := %DB1.DX(IDX :=#myByteOffset,Bit:=#myBitOffset);
//...
%DB1.DW(IDX := #myInt) := 12;
//...
```

In the second table, you can see the program sections after migration:

SCL

```
#myWord := PEEK_WORD(area:=16#84, dbNumber:=1, byteOffset:=#myInt);
//...
#myBool:=PEEK_BOOL(area:=16#84, dbNumber:=1, byteOffset:=#myByteOffset, bitOffset:=#myBitOffset);
//...
POKE(area:=16#84,dbNumber:=1,byteOffset:=#myInt,value:=12);
//...
```

Migration of indirect addressing of PLC tags

Indirect specification of PLC tags is implemented with the "PEEK"/"POKE" instructions in S7-1500. The migration automatically converts them to indirect addressing.

The following examples show the migration of indirect addressing of DB tags.

In the first table, you can see a program section prior to migration:

SCL

```
#myWord := MW(IDX := #myInt);
```

In the second table, you can see the program sections after migration:

SCL

```
#myWord := PEEK(area:=16#83,dbNumber:=0,byteOffset:=#myInt);
```

See also

Addressing operands indirectly (Page 2220)

POKE: Write memory address (Page 4213)

POKE_BOOL: Write memory bit (Page 4214)

PEEK: Read memory address (Page 4209)

PEEK_BOOL: Read memory bit (Page 4211)

POKE_BLK: Write memory area (Page 4216)

Instructions in SCL

Migration of instructions

S7-1500 has a slightly different set of instructions. Some instructions have been innovated or replaced by new instructions.

During migration, the instructions used in the program are retained as much as possible. If an instruction is not available in S7-1500, an attempt is made to replace it with a compatible or similar instruction. You then have to check the migrated program again and adapt it if necessary.

The following changes may occur, for example:

- The new instruction has additional formal parameters. Configure the additional parameters.
- A formal parameter of the new instruction has a different data type. In this case, the migration automatically adds an instruction to convert the data types.

If the migration cannot unequivocally ensure that the parameter assignment is semantically identical to the initial program, the relevant parameters are commented out. Check the semantics at these locations in the program after migration and correct it if necessary.

The following examples show the migration of the "AG_SEND" instruction to the "TSEND" instruction.

In the first table, you can see a call of "AG_SEND" prior to migration:

SCL

```
AG_SEND (ACT:=#myBool,
         ID:=#myInt1,
         LADDR:=#myWord,
         SEND:=#myAny,
         LEN:=#myInt2,
         DONE=>#myBool,
         ERROR=>#myBool,
```

SCL

```
          STATUS=>#myWord);
```

In the first table, you can see the call of "TSEND" after migration:

SCL

```
"TSEND_DB"((* ToReplace: REQ:=#myBool *)
          (* ToReplace: ID:=#myInt1 *)
          LEN:=INT_TO_UDINT(#myInt2),
          DONE=>#myBool,
          ERROR:=#myBool,
          STATUS:=#myWord,
          DATA:=#myAny;
```

The actual parameter of "REQ" was applied from the "ACT" parameter of the "AG_SEND" instruction. It is commented out because the program must be checked at this location.

The actual parameter of "ID" was applied from "AG_SEND" and also needs to be tested.

The "LEN" parameter has another data type with "AG_SEND". The migration automatically adds a conversion.

Migration of instructions in expressions

In some cases, migration generates several instructions from a single instruction, for example, when an output parameter has another data type after migration and the formal parameter must be converted. The conversion instruction is inserted after the migrated instruction. This means, however, that the original instruction can no longer be used as an expression. In this case, the migration adds a temporary tag in the interface of the migrated block and assigns it the result of the instruction. This temporary tag can then be used as an expression. Automatic migration cannot always be performed for complex expressions. Expressions that are not clearly interpretable are marked with a question mark after the migration, which results in syntax errors. Check the semantics at these locations in the program after migration and correct it if necessary.

The following examples shows the migration of the "READ_RTM" instruction to the "RTM" instruction.

In the first table, you can see a call of "READ_RTM" prior to migration:

SCL

```
IF READ_RTM(NR:=#myByte, CQ=>#myBool, CV=>#myOutInt)= 1 THEN
...;
END_IF;
```

In the second table, you can see a call of "RTM" after migration.

SCL

```
(* classic code: IF READ_RTM(NR:=#myByte, CQ=>#myBool, CV=>#myOutInt)= 1 THEN*)
#SCL_MIGRA_TEMP_INT_1:=RTM(NR:=#myByte,
```

SCL

```
                            MODE:=0,
                            PV:=#Migra_PV,
                            CQ:=#myBool,
                            CV:=#Migra_CV);
#myOutInt:=DINT_TO_INT(#Migra_CV);

IF #SCL_MIGRA_TEMP_INT_1= 1 THEN;
...
END_IF;
```

The instruction from the original program is transferred as a comment section to the migrated block. This makes it easy for you to track the changes that were made by the migration.

Because CV has the DINT data type for "RTM", a conversion instruction is inserted that converts the actual parameter from INT to DINT. This means the result of RTM can no longer be used as an expression within the IF instruction.

The temporary tag "SCL_MIGRA_TEMP_INT_1" is inserted in the interface of the migrated block. The result of the RTM instruction is assigned to the temporary tag.

The temporary tag is used as the expression in the IF instruction instead of the "RTM" instruction.

4.2.4.6 Migration of GRAPH programs to S7-1500

Information on migration of GRAPH programs

Introduction

GRAPH programs that were created with S7-300/400 can be migrated automatically to S7-1500. The PLC migration optimizes the programs for running on a CPU of the S7-1500 series. The migration log reports all automatic program changes made during the migration and informs you of any necessary manual changes you still have to perform.

Adaptation of the GRAPH program by the PLC migration

The following list shows some examples of automatic adaptations that are made during the PLC migration. All adaptations are described in detail in the sections belows.

- Adaptation of the block interface
  The interface of the GRAPH blocks is changed slightly. This enables use of the optimized block access of the S7-1500 for GRAPH blocks.
- Integrated symbolic addressing
  SIMATIC S7-1500 is characterized by its integrated symbolic programming. This makes the program more efficient and reduces the risk of access errors.
  For this reason, if the initial program uses addresses for which a symbolic name is not declared, the migration assigns a symbolic name for the address whenever possible.
- Memory model of the GRAPH DB
  Based on the performance data of the S7-1500, it is no longer necessary to use GRAPH DBs with minimized memory space requirements. For this reason, GRAPH no longer offers different memory space models. You have access to the full parameter set and thus the complete functionality in all GRAPH DBs.
  During migration, DBs with minimized memory space requirements are expanded to include the full parameter set.

New functions of the S7-1500

Following migration of a GRAPH program, you can make full use of the functionality of the S7-1500. The following list shows some examples of the new functions:

- Optimized block access
  With optimized block access, the declared data elements are arranged automatically in the available memory area of the block in such a way as to make optimal use of its capacity. The data is structured and saved in a manner that is optimal for the CPU used. This allows you to increase the performance of the CPU. Access errors, from the HMI, for example, are not possible.
  The "Optimized block access" attribute is always enabled for GRAPH blocks in S7-1500 and cannot be deselected.
  See also: Blocks with optimized access (Page 2175)
- New instructions
  The "Instructions" task card contains the complete set of instructions of the S7-1500. Numerous instructions have been newly developed or innovated. For example, the "Program control" folder in the "Basic instructions" pane contains the "GET_ERROR" instruction for programming local troubleshooting.
  See also: GET_ERROR: Get error locally (Page 4433)
- LAD/FBD functions
  For S7-300/400, GRAPH only supported the programming of a preceding logic operation at the first Boolean input of a box. In S7-1500, preceding logic operations can be programmed for all Boolean inputs.
  The enable output (ENO) can be enabled or disabled selectively for individual instructions.
  See also: Enabling and disabling the EN/ENO mechanism (Page 2600)

- New data types
  S7-1500 provides an expanded set of data types and new data type conversion options.
  See also: Overview of the valid data types (Page 2236)
- Constants
  S7-1500 provides additional options for constant notation. Thus you can also assign data types to constants.
  See also: Auto-Hotspot

Changes in the GRAPH program

Introduction

The PLC migration automatically adapts the GRAPH program to the new CPU family as much as possible. In so doing, both the actions as well as the LAD or FBD networks that you used for programming of conditions are analyzed. Program structures that are no longer permitted are updated. Instructions that are not available on the new CPU are replaced with appropriate S7-1500 instructions.

Symbolic addressing

SIMATIC S7-1500 is characterized by its integrated symbolic programming. This makes the program more efficient and reduces the risk of access errors.

If the initial program uses absolute addresses for which a symbolic name is not declared, the absolute address is marked as invalid. You must then declare a symbolic name before you can compile the migrated program.

If the initial program uses absolute addresses for which a symbolic name is declared, the migration uses the symbolic name instead of the absolute address. Accesses to individual bits within a declared symbolic operand are replaced by slice accesses.

The following example shows an absolute access before the migration:

```
%DB1.DBx1.1
```

The following shows you how the migration replaces the absolute access. Since the "FirstDataByte" tag was declared for the first data byte in "DataBlock1", the access to the first bit within this tag can be replaced by a slice access.

```
"DataBlock1".FirstDataByte.x1
```

Fully qualified addressing

S7-300/400 programs frequently use partially qualified addresses to access addresses indirectly. In this case only the address of a DB tag within a data block is specified. The DB name or DB number is not specified in the address (e.g., `%DBX0.2`). At runtime, the program accesses the data block that is currently stored in the DB register. Partially qualified addressing is not supported in S7-1500. In order to address DB tags indirectly, use block parameters of data type "DB_ANY".

If you have used partly qualified addresses in GRAPH programs, these are reported as errors after the migration. The block cannot be compiled. Then replace the partly qualified address with a fully qualified address.

LAD/FBD elements in conditions

The LAD or FBD networks in Transition, Interlock, Supervision, and Permanent instructions are migrated according to the same rules as the networks in pure LAD or FBD blocks.

See also: Migration of LAD/FBD programs to S7-1500 (Page 213)

GRAPH actions

SIMATIC S7-1500 provides a slightly modified set of instructions. Some instructions have been innovated or replaced by new instructions.

If you have used instructions in GRAPH actions, the migration checks whether these are still supported on S7-1500. The instructions are carried over as much possible. If an instruction is not available in S7-1500, an attempt is made to replace it with a compatible or similar instruction.

Examples of this are the "RLDA: Rotate left by status bit CC 1" and "RRDA: Rotate right by status bit CC 1" instructions. These are replaced by the "ROR: Rotate right" and "ROL: Rotate left" instructions.

If instructions are replaced during the migration, this is noted in the migration log. You then have to check the migrated program again and adapt it if necessary.

See also: Migration of instructions to S7-1500 (Page 190)

Adaptation of the interface of GRAPH blocks

Modification of the block interface during migration

The interface of the GRAPH blocks is adapted during the migration. The adaptations enable you to use the optimized memory concept of the S7-1500 with GRAPH. Optimized block access is always active for GRAPH blocks on the S7-1500 and cannot be deselected. Changes are made in the following areas:

- Static parameters: "STATIC"
- Output parameters: "OUTPUT"

See also: Blocks with optimized access (Page 2175)

If you have used parameter names in the initial program that are reserved in S7-1500, these are renamed during the migration.

Static parameters

Static parameters are adapted as follows during the migration:

- The interface contains a structure in the "Static" section for each step and transition of the GRAPH FB. In S7-300/400, the "GraphStep", "GraphStepMin", "GraphTransition", and "GraphTransitionMin" data types were used. In S7-1500, the new "G7_StepPlus" and "G7_TransitionPlus" data types are introduced for reproducing the step and transition structures.
  The migration proceeds as follows here:
  - The "SNO" and "TNO" parameters are transferred together with their default values from the old structures; all other default values are not transferred. GRAPH programs with a DB with minimized memory space requirements do not contain the "SNO" and "TNO" parameters. If you migrate a DB with minimized memory space requirements, the migration assigns default values for these parameters starting with "1".
  - The locations of use of the modified parameters in the GRAPH program are not adapted automatically but rather are marked as errors. You must manually adapt the locations of use after the migration.
- All other static parameters are transferred to the "RT_DATA" structure. The default values are thus carried over as much as possible and the locations of use in the program are adapted automatically.

Output parameters

Because S7-1500 no longer supports the criteria analysis, the following output parameters are removed during the migration:

- T_CRIT
- T_CRITOLD
- T_CRITFLT
- S_CRITLOC
- S_CRITLOCERR
- S_CRITSUP

The locations of use in the GRAPH program are not adapted automatically but rather are marked as errors. You must manually adapt the locations of use after the migration.

See also

Keywords (Page 2197)

4.3 Migrating S7-1200 to firmware as of V4

4.3.1 Basic information on upgrading to V4

Introduction

If you have used a CPU with firmware version V3 in your project and want to upgrade to a CPU with firmware V4.0 or later, you can easily replace the device.

The TIA Portal offers the "Change device" function for this purpose. The project remains unchanged when the device is replaced. You can continue using the programs that you created with firmware version V3.

Rules

The following basic rules apply when replacing a device:

- Device replacement is only possible if the CPU used to date has been operating with firmware version V3.0. S7-1200 CPUs with firmware version V1.0 or V2.0 must first be replaced with a V3.0 CPU.
- It is not possible to replace a V4 CPU with a V3 CPU. If you want to continue using the existing V3 CPU, create a copy of this CPU before replacing the device.
- The program cannot be transferred to the new CPU via a memory card. Instead, use the "Change device" function, which is described in the following sections.

HMI panels

Configured HMI panels are treated differently during device replacement, depending on the firmware version of the panel and the communication mode.

The following table shows the HMI connections that are supported with the migration:

Firmware version of the panel	PUT/GET communication	Migration to V4
V11 or later	No	S7-1200 does not support this configuration. Upgrade the firmware of the HMI panel to V12.0. Then compile and load the configuration.
V11 or later	Yes	S7-1200 supports this configuration. The connection is established automatically while you compile and load the project after replacing the device.
V12 or later	No	S7-1200 supports this configuration. The connection is established automatically while you compile and load the project after replacing the device.

Note

HMI TP 177B 4"

The HMI TP 177B 4" with firmware version V11.0.2 cannot be operated with S7-1200 V4. Replace the panel with a new device, if necessary.

S7-1200 expansion modules

If you are already using the following centrally plugged S7-1200 modules in your system, you must perform a firmware update for these modules in order to guarantee operation with S7-1200 V4.

- ASi - Master - CM 1243
- DP - Master - CM 1243-5
- WAN CP - CP1243-1

Newly shipped S7-1200 modules have the latest firmware installed ex works.

Protected blocks

Blocks equipped with know-how protection or copy protection cannot be converted to V4. If the project contains protected blocks, you must remove the protection prior to migration.

If these are supplied blocks and you do not know the password, ask your supplier either for the password or for a V4-compatible block.

WARNING

Preventing personal injury and material damage

Changes are made to the program during device replacement in some cases. Therefore, thoroughly check the program in a test environment after replacing the device and before putting it in operation.

Note

Additional support

You can find the latest FAQs about migrating to S7-1200 V4 (http://support.automation.siemens.com/WW/view/en/82140966) in Siemens Industry Online Support.

If you need further assistance with migrating to S7-1200 V4, please contact SIMATIC Customer Support.

See also

Migrating to V4 (Page 240)
Special considerations after migrating to V4 (Page 241)

4.3.2 Migrating to V4

Requirement

- A CPU with firmware version V3 is available in the project.
- The project contains no protected blocks.

Procedure

Proceed as follows to replace a CPU:

1. Select the V3 CPU you want to replace.
2. Select the "Change device" command in the shortcut menu.
   The "Change device" dialog box appears.
3. Under "New device" in the tree structure, select the V4 CPU with which you want to replace your current V3 CPU.
4. Click "OK".
   The existing CPU is replaced by the new one.
5. Select the new CPU and select the "Compile > Hardware and software (only changes)" command in the shortcut menu.
   The device configuration and the user program are compiled again.
6. Optional: If necessary, apply know-how protection or copy protection to individual blocks in the program.
7. Select the new CPU and select the "Download to device > Hardware and software (only changes)" command in the shortcut menu.
   The device configuration and the user program are loaded into the new CPU.
   This completes the device replacement.

WARNING

Preventing personal injury and material damage

Changes are made to the program during device replacement in some cases. Therefore, thoroughly check the program in a test environment after replacing the device and before putting it in operation.

Note

Additional support

You can find the latest FAQs about migrating to S7-1200 V4 (http://support.automation.siemens.com/WW/view/en/82140966) in Siemens Industry Online Support.

If you need further assistance with migrating to S7-1200 V4, please contact SIMATIC Customer Support.

See also

Basic information on upgrading to V4 (Page 238)

Special considerations after migrating to V4 (Page 241)

4.3.3 Special considerations after migrating to V4

Functional changes in V4

S7-1200 V4 offers significantly enhanced functionality. The most important functional changes that you need to consider after migrating from V3 are described briefly below.

You can find more information on S7-1200 in the "SIMATIC S7-1200 Programmable controller" system manual.

See also:

TIA Portal in Siemens Industry Online Support (http://support.automation.siemens.com/WW/view/en/65601780)

Organization blocks

With S7-1200 V4, you can specifically set the interruptibility of each organization block used. When a device is replaced, all the organization blocks are configured as non-interruptible, to ensure that the executability of your V3 program remains unchanged. The OB priorities from the V3 program also remain unchanged. After migration, you can change the settings for priority and interruptibility as needed.

The behavior of diagnostic interrupts in V4 has changed as follows:

In V3, the start information always contained information on the triggering module, including the channel number. In V4, this information is only generated for a pending diagnostics event. If no diagnostic event is pending, for example, because the fault has already been corrected, only the triggering module is indicated.

Access levels

S7-1200 V4 offers an extended access level concept. The following table shows how the protection levels of the V3 firmware are indicated in V4:

V3 protection level	V4 access level
No protection	Full access (no protection)
Read-only	Read access
Write/read protection	HMI access

The access level "No access (complete protection)" has been newly introduced with V4. It was not available in previous versions.

Instruction libraries

After migrating to S7-1200 V4, instructions from the libraries of the V3 firmware version are still available. This ensures that you can continue to use your program unchanged. In addition, S7-1200 V4 offers many new instructions which are also compatible with the instructions of the S7-1500.

You can find more information on the instruction libraries of S7-1200 in the "SIMATIC S7-1200 Programmable controller" system manual.

See also:

TIA Portal in Siemens Industry Online Support (http://support.automation.siemens.com/WW/view/en/65601780)

Motion Control

When a device is replaced, the Motion Control objects from the libraries of the firmware versions V1 and V2 are replaced with the corresponding objects from the V3 libraries. The objects from V3 libraries are compatible so that you can continue to use the programs unchanged.

The libraries of the S7-1200 V4 offer many new Motion Control functions, which are compatible with the functions of the S7-1500. If you want to use V4 libraries, select them on the "Instructions" task card after replacing the device.

You can find more information on the new Motion Control functions in the "SIMATIC S7-1200 Programmable controller" system manual.

See also:

TIA Portal in Siemens Industry Online Support (http://support.automation.siemens.com/WW/view/en/65601780)

Web server

The following settings for operation via a web server are transferred from the V3 CPU to the V4 CPU during device replacement:

- Activate web server on this module
- Permit access only with HTTPS

If you want to operate the V4 CPU via a web server, you need to set up user accounts with assigned user rights in the user management. Only the standard web pages are available to standard users without any additional rights.

Note

Additional support

You can find the latest FAQs about migrating to S7-1200 V4 (http://support.automation.siemens.com/WW/view/en/82140966) in Siemens Industry Online Support.

If you need further assistance with migrating to S7-1200 V4, please contact SIMATIC Customer Support.

Communication via PUT/GET

Communication via PUT/GET is enabled after the device replacement. Note that the new integrated connection types offer a security standard higher than PUT/GET communication. If you do not use PUT/GET communication, you should disable it.

See also

Basic information on upgrading to V4 (Page 238)

Migrating to V4 (Page 240)

4.4 Programming recommendations

4.4.1 The new S7-1500 CPU functions at a glance

Higher performance

The S7-1500 represents a CPU series that provides much higher performance than the CPUs of the S7-300/400 series. When programming with STEP 7 V5.x, you were probably used to working with programming methods such as absolute addressing to achieve higher performance from the CPU and leaner program code.

Due to the high performance that the S7-1500 provides, these programming methods are made obsolete.

In the paragraphs below we would like to introduce some new programming options of the S7-1500.

Universal symbolism

The S7-1500 allows you to use the symbolism throughout the entire project. Using the auto-complete function, you are given context-dependent support for programming with symbols within the programming editors. The data elements, for example those in a data block, are assigned only a symbolic name in the declaration but no fixed address within the data block. This allows you to fully exploit the high performance of the S7-1500 when accessing these data elements. The absolute addresses of operands need not be known and access errors are avoided.

Your program code will be clearer due to the symbolism and you have to comment less. All points of use are automatically updated when a correction is made to the symbolism.

An example of the use of universal symbols is available at: Symbolic addressing (Page 247)

Optimized block access

With optimized block access, the declared data elements are arranged automatically in the available memory area of the block in a way that makes optimal use of its capacity. The data is structured and saved in way that is optimal for the CPU used. The storage is left to the system. The data elements are assigned only a symbolic name in the declaration by which the

tag within the block can be addressed. This allows you to increase the performance of the CPU. Access errors, from the HMI, for example, are not possible.

Comparison of block accesses
Standard < > Optimized

SIEMENS

	Standard block access (S7-1200/1500 compatible with S7-300/400)	Optimized block access (S7-1200/1500 only)
Data management	You can address tags using both symbolic (memory-optimized) and absolute (user-defined) addresses.	The system manages and optimizes the data storage. This results in optimal use of the memory capacity.
Performance	The access to a CPU of the S7-1200/1500 series does not always occur as fast as possible because the data storage can be inefficient due to absolute addressing.	The access always occurs as fast as possible because the data storage is optimized by the system and no fixed addresses are assigned.
Susceptibility to errors	Absolute addressing (e.g., from HMI or when indirect addressing is used) can lead to inconsistencies after a change to the fixed address.	Access errors, e.g., from HMI or when indirect addressing is used, are not possible because the access is symbolic.
Retentivity	Valid for all tags of a data block	Valid for individual tags

Recommendation:
To achieve the best-possible performance, the combining of different block access types within your program is not recommended.

You can find additional information on blocks with optimized access under "See also".

New data types

The new data types LWORD, LINT, ULINT, LTIME, LTOD, LDT and the array (32-bit limit) offer much higher calculation accuracy when using mathematical functions. In the area of implicit and explicit data type conversion, you have more options in comparison to the CPUs of the S7-300/400 series.

You can find additional information on the new data types under "See also".

PLC data types

PLC data types (UDT) are data structures that you define and that can be used multiple times within the program. The structure of a PLC data type is made up of several components, each of which can contain different data types. You define the type of components in the declaration of the PLC data type.

You can use the PLC data type as a base data type for defining tags and as a template for creating global data blocks. If you later make changes to the PLC data type, the changes are automatically updated at all points of use.

You can also symbolically access individual elements of an array within a PLC data type.

You can find additional information on the new data types under "See also".

You can find an example for using PLC data types under: Using PLC data types (UDT) (Page 256)

Uniform instructions in all programming languages

You are provided with a uniform set of instructions in all programming languages (LAD, FBD, STL, SCL, and GRAPH).

Slice access

Slice access gives you the option of specifically addressing areas within declared tags. You can implement symbolic access to a single bit up to the level of the tag. The single bit is then addressed absolutely.

You can find additional information on slice access under "See also".

Indirect addressing

Indirect addressing offers you the option of addressing operands whose address is not calculated until runtime. All programming languages provide general methods for indirect addressing, for example, via POINTER. In the SCL programming language, you can also use PEEK and POKE instructions.

You can find additional information on indirect addressing under "See also".

Additional programming recommendations

You can find more information about programming recommendations and a programming guide in the Siemens Industry online support under:

- FAQs: Programming recommendations (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=0&lang=en&referer=%2fWW%2f&func=cslib.csinfo&siteid=csius&Datakey=47071380&extranet=standard&groupid=4000002&viewreg=WW&nodeid0=29156492&objaction=csopen)
- Background and system descriptions in the Programming Guideline (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=100&lang=en&referer=%2fWW%2f&func=cslib.cssearch&nodeid0=4000024&viewreg=WW&siteid=csius&extranet=standard&groupid=4000002&objaction=cssearch&content=adsearch%2Fadsearch%2Easpx)

See also

Blocks with optimized access (Page 2175)

Data types (Page 2236)

PLC data types (Page 2280)

Addressing areas of a tag with slice access (Page 2216)

Indirect addressing (Page 2220)

4.4.2 Symbolic addressing

Advantages of symbolic addressing

The use of universally applied and meaningful symbols in the overall project makes the program code easier to read and understand.

This gives you the following advantages:

- You do not have to write detailed comments.
- Data access is faster.
- No errors occur when accessing data.
- You no longer have to work with absolute addresses.
- The assignment of the symbol to the memory address is monitored by STEP 7, which means all points of use are automatically updated when the name or the address of a tag changes.

Programming in STEP 7 V5.x

STEP 7 V5.x already gave you the option to write a program that is easier to read by using descriptive names for operands and blocks. You did this by assigning the symbolic operands to memory addresses and blocks in the symbol table. In order for a change in the symbolism to also have an affect on the program code in the programming editor, you had to use the "Operand priority" property to specify whether the symbol or the absolute value had priority.

The use of symbolic addressing allowed you to create a program with greater clarity. However, in some cases, such as when programming with user-defined data types (UDT), this could impair performance.

You could increase the performance by ignoring the symbolism in the UDT and using absolute addressing. This made in necessary, however, to know the data storage. Changes to the UDT were not automatically updated. Using absolute addressing, you could also access parts of a tag and edit these. The only drawback of absolute addressing, however, was that after a certain level the program code was cluttered and you had to insert additional comments for better orientation.

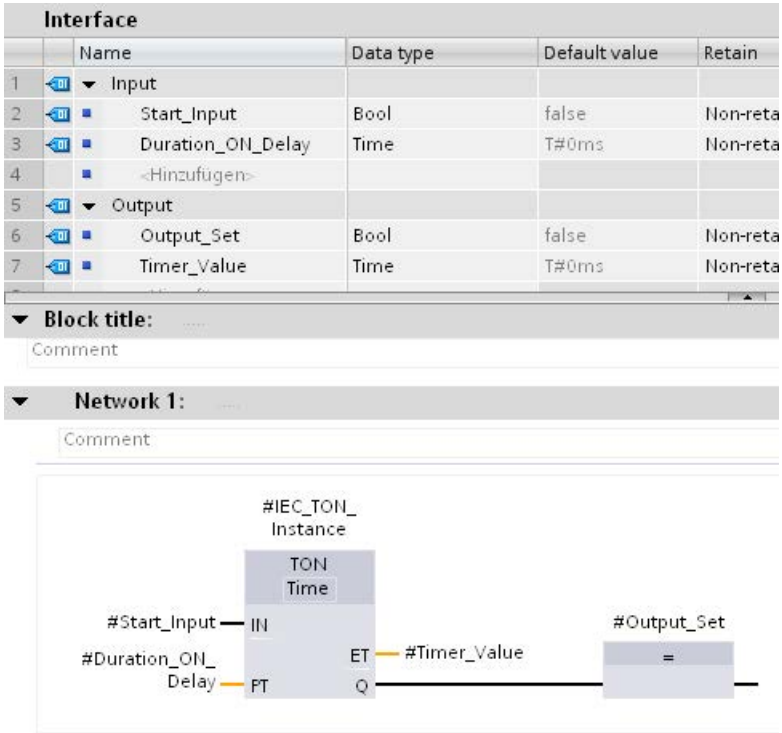
Procedure in STEP 7 TIA Portal

The S7-1500 CPU offers much higher performance than the S7-300/400 CPUs. To fully use this high performance, we recommend you enable optimized block access for all blocks and use symbolic addressing in the program code.

The programming editor helps you work with symbols through context-sensitive input help, for example, auto-complete. Using it, you can easily access existing tags or instructions during programming.

Programming example

The following example shows you how to symbolically access individual elements:



You can use the tag names that you have defined in the block interface directly at the parameters of the TON instruction without knowing the absolute address of the tag.

See also

Setting up block access (Page 2177)

Basics of block access (Page 2175)

Using autocompletion (Page 2540)

Addressing areas of a tag with slice access (Page 2216)

Using VARIANT instructions (Page 263)

4.4.3 Using IEC timers and counters

Advantages of IEC timers and counters

The universal use of IEC timers and counters makes your program code more efficient.

This gives you the following advantages:

- The blocks can be called multiple times with newly generated instance data blocks.
- The IEC counters have a large counting range.
- Compared to S5 timers, IEC timers have better performance and greater time accuracy.

Programming in STEP 7 V5.x

The S5 timers and counters in STEP 7 V5.x were addressed absolutely via a number. Due to this dependency on numbers, program blocks were not reusable with S5 timers and counters.

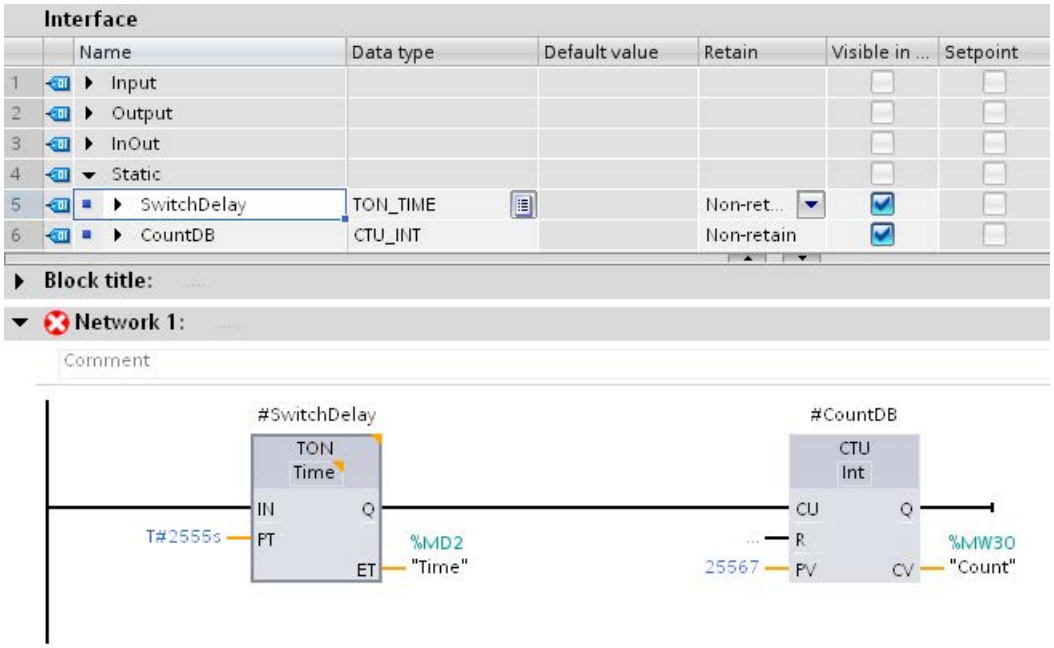
The value range of a timer was limited to 9990 seconds and that of a counter limited to a maximum of 999.

Procedure in STEP 7 TIA Portal

You declare IEC timers and counters in the program block where they are called or needed. The IEC timer is a structure of data type IEC_TIMER, IEC_LTIMER, or TON_TIME and TON_LTIME, for example, which you can also declare as a local tag in a block. The IEC counter is a structure of data type IEC_SCOUNTER, IEC_USCOUNTER, etc.

Programming example in the TIA Portal

The following example shows you how to declare an IEC timer and IEC counter as a local tag:



The data of the TON IEC timer and the CTU IEC counter is stored as a local tag (multi-instance) in the block interface.

See also

Timers (Page 2255)

IEC Timers (Page 3177)

4.4.4 Flexibly using enable output ENO

Advantages

You can to detect runtime errors and avoid program termination using the EN/ENO mechanism for each instruction and block call. Overflow, for example, is reported via the ENO enable output for mathematical functions.

In STEP 7 TIA Portal, the ENO enable output is disabled by default in the programming languages LAD and FBD. If needed, you can activate the enable output and thereby deliberately target the instructions for which you want to have error evaluation.

This gives you the following advantages:

- The performance increases with ENO disabled.
- Runtime errors do not cause the CPU to go to STOP when ENO is enabled.

Procedure in STEP 7 TIA Portal

Proceed as follows to activate the EN/ENO mechanism of an instruction:

1. In your program, right-click the instruction at which you want to activate the EN/ENO mechanism.
2. Select the "Generate ENO" command from the shortcut menu.
   The ENO value is generated for the instruction. Other instructions are inserted with the enable output.

The tables below list the instructions for which ENO enable output can be disabled:

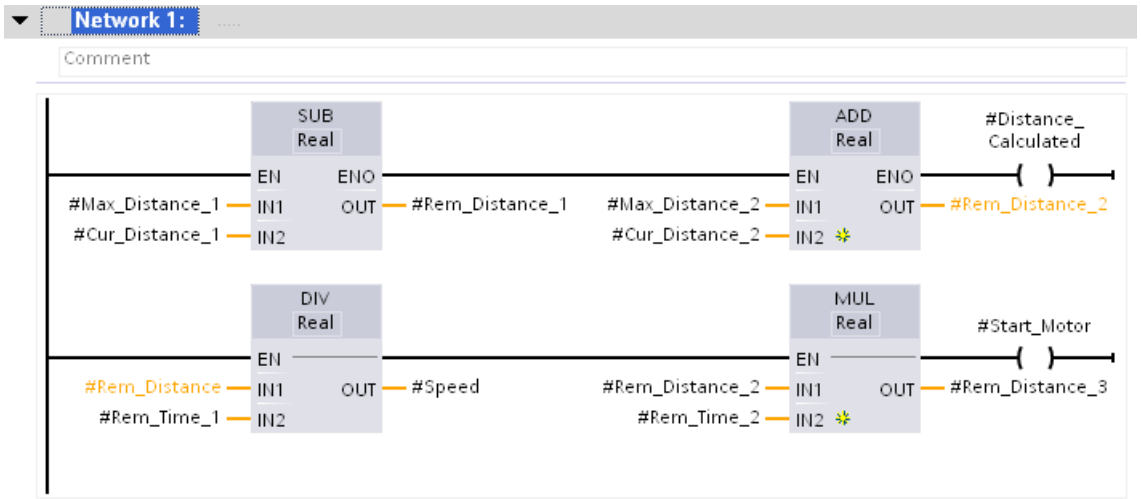
Basic instructions	
Math functions	ADD, SUB, MUL, DIV, MOD, INC, DEC, ABS, NEG, SQR, SQRT, LN, EXP, SIN, COS, TAN, ASIN, ACOS, ATAN, FRAC, EXPT, MIN, MAX, LIMIT, CALCULATE
Move operations	MOVE, SWAP, MOVE_BLK, UMOVE_BLK, FILL_BLK, UFILL_BLK, MOVE_BLK_VARIANT
Conversion operations	CONVERT, ROUND, CEIL, TRUNC, FLOOR, NORM_X, SCALE_X
Word logic operations	AND, OR, XOR, INV, DECO, ENCO
Shift and rotate	SHR, SHL, ROR, ROL

Extended instructions	
String + Char	CONCAT, LEFT, RIGHT, MID, DELETE, INSERT, REPLACE, FIND, LEN, S_CONV
Date and time	T_CONV

You can find additional information on the EN/ENO mechanism in each programming language under "See also".

Programming example

The following example shows you how to use instructions with the ENO enable output enabled and disabled:



If you have activated the ENO enable output, for example with the SUB instruction, all subsequent instructions are also applied with an activated enable output. If an arithmetic error occurs during the execution of the SUB instruction, the ADD instruction is not executed.

The ENO enable output is disabled for the DIV instruction in the second branch. If a runtime error occurs during execution, the MUL instruction is executed anyway.

See also

Enabling and disabling the EN/ENO mechanism (Page 2600)

Basics of the EN/ENO mechanism (Page 2447)

4.4.5 Using MOVE instructions in STL

Possible applications

You can now program with MOVE instructions in STL on an S7-1500 CPU.

This gives you the following advantages:

- The program structure is easier to create.
- The performance of the CPU increases.

Programming in STEP 7 V5.x

In STEP 7 V5.x, you used the "BLKMOV: Move block" and "UBLKMOV: Move block uninterruptible" system functions to implement the MOVE functionality.

Procedure in STEP 7 TIA Portal

The following new MOVE instructions are available in STEP 7 TIA Portal:

- MOVE: Move value
- MOVE_BLK: Move block
- MOVE_BLK_VARIANT: Move block
- UMOVE_BLK: Move block uninterruptible

You can find additional information on the new MOVE instructions under "See also".

Programming example

The following example shows you how the "MOVE_BLK instruction works: Move block". An array block is copied into another array block:

Data_DB

	Name	Data type
1	Static	
2	Array_1	Array [0..10] of Int
3	Array_1[0]	Int
4	Array_1[1]	Int
5	Array_1[2]	Int
6	Array_1[3]	Int
7	Array_1[4]	Int
8	Array_1[5]	Int
9	Array_1[6]	Int
10	Array_1[7]	Int
11	Array_1[8]	Int
12	Array_1[9]	Int
13	Array_1[10]	Int
14	Array_2	Array [0..10] of Int
15	Array_2[0]	Int
16	Array_2[1]	Int
17	Array_2[2]	Int
18	Array_2[3]	Int
19	Array_2[4]	Int
20	Array_2[5]	Int
21	Array_2[6]	Int
22	Array_2[7]	Int
23	Array_2[8]	Int
24	Array_2[9]	Int
25	Array_2[10]	Int

```
Network 1:
Comment
1   CALL  MOVE_BLK
2       Int  UInt
3       IN    :="Data_DB".Array_1[0]
4       COUNT :=10
5       OUT   :="Data_DB".Array_2[0]
6
7
```

Using the MOVE_BLK instruction, ten elements from "Array_1" of the "Data_DB" data block are copied into "Array_2" of the same data block.

See also

MOVE: Move value (Page 3782)

MOVE_BLK: Move block (Page 3793)

MOVE_BLK_VARIANT: Move block (Page 3795)

UMOVE_BLK: Move block uninterruptible (Page 3800)

4.4.6 Implementing array access with a variable index

Advantages of the variable index

For addressing the components of an array, you can specify tags of the integer data type as well as constants as the index. Integers with lengths up to 32 bits are allowed here. When tags are used, the index is calculated during runtime. You can, for example, use a different index for each cycle in program loops. You can also access an array within a PLC data type.

This gives you the following advantages:

- No addressing with address registers or with self-assembled pointers, for example, an ANY pointer, is necessary.
- More flexibility within your program.
- The variable index is available in all STEP 7 programming languages.
- It uses the existing names of data blocks and array tags. This improves readability of the program code.
- The start address of the array does not have to be known.
- The program code is easier to write.
- The compiler generates optimized program code.

Procedure in STEP 7 V5.x

In STEP 7 V5.x, you had to use an address register with the help of a self-configured POINTER for indexed addressing of array elements. The SCL programming language already supported indirect addressing with variable index.

Programming example in STEP 7 V5.x

The "Data_classic" data block is required for the following STL example. To address an element of the "Quantities" array, the following commands must be used:

STL	Explanation
`OPN "Data_classic"`	`// The "Data_classic" data block is called.`
`L #index`	`// The value of the local tag #index is loading in accumulator 1.`
`SLD 3`	`// Move bits 0 to 31 of accumulator 1 by 3 positions to the left.`
	`// Fill the now empty bit places with zeros.`
`LAR1`	`// Load address register 1 with contents of accumulator 1.`
`L DBW [AR1, P#10.0]`	`// Load the array element addressed with #index into accumulator 1.`
	`// P#10.0 = Start address of the array`

You need to consider the following with this procedure, however:

- The array name is not used. This reduces the readability of the program code and makes it necessary to add a comment.
- The start address of the array (P#10.0) must be known to perform the addressing.
- The indexed addressing must involve the use of the address register.

Programming example in STEP 7 TIA Portal

The following syntax is used for indirect indexing of the "Quantities" array that was declared in the "Data_DB" data block:

```
"Data_DB".Quantities["i"]          // One-dimensional array
"Data_DB".Quantities["i"].a        // One-dimensional array of STRUCT
"Data_DB".Quantities["i","j"]      // Multi-dimensional array
"Data_DB".Quantities["i","j"].a    // Multi-dimensional array of STRUCT
```

Part	Description
Data_DB	Name of the data block in which the array is located
Quantities	Tag of the Array data type
i, j	PLC tags of the integer data type that are used as pointers
a	Additional partial tag of the structure

The following example shows indirect indexing of an array component in STL.

You need another program line for addressing an array element:

Addressing in STL	Explanation
`L "Data_DB".Quantities[#index]`	`// The value of the array element #index is loaded directly from the data block into accumulator 1.`

Note the following to get the best performance:

- Declare tags that are used as an array index as an integer of less than or equal to 32 bits.
- Store intermediate results and array indexes in the temporary local data area.

See also

Addressing operands indirectly (Page 2220)

Array (Page 2267)

4.4.7 Using PLC data types (UDT)

Advantages of the PLC data types

PLC data types (UDT) are data structures that you define and that can be used multiple times within the program. The structure of a PLC data type is made up of several components, each of which can contain different data types. You define the type of components in the declaration of the PLC data type. By using PLC data types, you can optimally exploit the high performance of the S7-1500 CPU.

Many programs require contiguous data records that have to be processed by different locations in the program. They are optimized or replaced as the program runs. These are, for example:

- Data records for material tracking
- A parameter set for a motor setting
- Various recipes

This gives you the following advantages:

- PLC data type elements can also be addressed indirectly, which means the address is variable and is first calculated in runtime.
- Tags based on a PLC data type inherit all properties of the PLC data type.
- Modifying the PLC data type results in automatic adjustment of all tags derived from it.
- Using universal symbolism makes the program easier to read, because the names of the individual elements of a PLC data type are displayed in the program.
- Increased performance because optimized program code is generated.

- The PLC data type can be passed as a complete structure for a block call.
- Simplified calling interface due to fewer parameters that have to be supplied.

Procedure in STEP 7 V5.x

STEP 7 V5.x already gave you the option to create a data record as a structured tag using the STRUCT data type or a PLC data type (UDT). However, the performance was adversely affected by the use of symbolic addressing.

The declaration in the data blocks was mostly implemented as an anonymous structure. The blocks themselves were then programmed to pass the values of the structure as actual parameters and the calculated values were copied back into the structure. This enabled you to also pass the data block number and use absolute addressing in the block. The number of parameters that you had to supply was often very large. The actual data was stored in the data blocks and the calculated values were passed to other blocks. But no symbolism was available when passing the data block tags.

Procedure in STEP 7 TIA Portal

If you work with PLC data types in your program, you can assign them to both a formal parameter and an actual parameter. This means you no longer have to declare each individual parameter. If a block has an input parameter that is based on a PLC data type, you must transfer a tag that has the same PLC data type as an actual parameter.

To use a PLC data type, follow these steps:

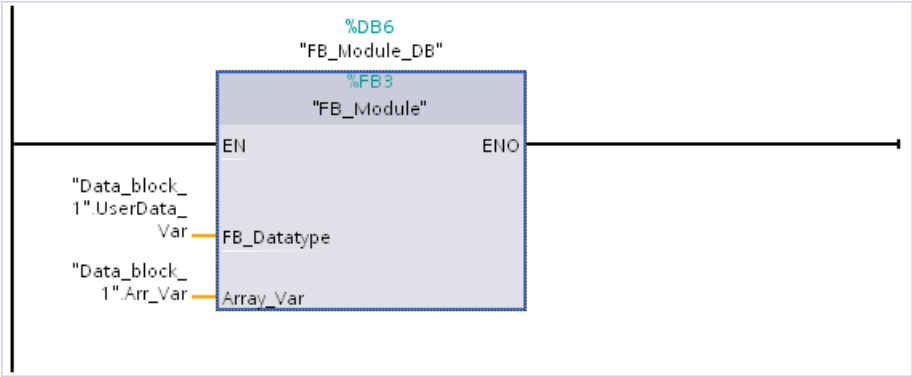
1. Declare the data records as PLC data type with a suitable name, for example, "Material tracking".
2. Use this data type to declare a data block or a DB tag.
3. Define the formal parameter as VAR_IN_OUT / VAR_IN or VAR_OUT tag in the block interface.
4. Assign the PLC data type to the formal parameter in the "Data type" column.

Programming example in the TIA portal

The following example shows the call and the parameter assignment of a function block (FB) with two formal parameters. The data type of the DB tag (Array [1 .. 10] of BOOL) and the PLC data type (UDT) are identical to the data types of the FB parameters.

Data_block_1

	Name	Data type	Start value
1	Static		
2	Arr_Var	Array [1..10] of Bool	
3	UserData_Var	"UserDataType"	
4	Var1	Bool	false
5	Var2	Bool	false
6	ByteVar1	Byte	16#0
7	WordVar1	Word	16#0
8	IntVar1	Int	0
9	RealVar1	Real	0.0



Declare the "UserDataType" PLC data type in the project tree in the "PLC data types" folder. Then use it as a data type for the "UserData_Var" tag. You declare both tags "UserData_Var" and "Arr_Var" in the "Data_block_1" data block and use them to supply the parameters of the "FB_Module" block.

See also

Other useful information regarding structured PLC tags (Page 2485)

Creating structured PLC tags (Page 2487)

PLC data types (Page 2280)

General rules for assigning parameters (Page 2186)

4.4.8 Querying and fixing errors in the program code

Error handling in the TIA Portal

Various error scenarios can occur within the program code during runtime. These can involve, for example, an access error or an overflow in mathematical operations. If you do not catch such a runtime error using a program code sequence within your program, the CPU reacts as follows in the event of an error, depending on the model:

- CPU S7-1200:
  - The CPU remains in RUN mode and writes an entry to the diagnostic buffer.
- CPU S7-1500:
  - When a programming error occurs, the CPU goes to STOP mode and writes an entry to the diagnostic buffer.
  - With an I/O access error, the CPU remains in RUN mode and writes an entry to the diagnostic buffer.

To be able to respond appropriately to possible errors, in addition to global error handling (for example, the use of the organization blocks "Programming error OB" or "I/O access error OB"), we recommend that you implement local error handling in your program (for example, use an

EN/ENO mechanism, the parameters RET_VAL, STATUS and ERROR, or the instructions "GET_ERROR"/"GET_ERR_ID").

Note

STL programming language

In STL, you use the BR bit of the status word instead of the EN/ENO mechanism.

Global error handling intervenes at the end of each program cycle, while local error handling is capable of intervening immediately after an error occurs.

Note

BR bit and EN/ENO mechanism as first indicator

A first indicator for an error can be either the BR bit of the status word or the ENO enable output. If these return signal state "0", there is an error in the execution of the instruction. Signal state "1" indicates that there is no error and no further error analysis is necessary.

Options for local error handling

The following options for local error handling of programming and access errors are available:

Error handling method	Validity	Explanation
EN/ENO mechanism	S7-1200/1500	You can use the ENO enable output to detect and handle runtime errors. Execution of subsequent instructions depends on the signal state of this parameter. You can avoid program crashes by using the EN/ENO mechanism. The block status is passed on in the form of a Boolean tag.
Output parameter RET_VAL	S7-1200/1500	Using the RET_VAL parameter as the return value of system functions (SFCs). you can display general error codes that may relate to any instruction or specific error codes that apply only to the instruction at hand. A maximum of one tag of the data type INT or WORD can be output. This error handling method is suitable for querying a communication error or incorrect data access, for example.
Output parameters STATUS and ERROR	S7-1200/1500	Using the STATUS and ERROR parameters as return values of system function blocks (SFBs), you can query device-specific error information. The error information is output in a pre-defined structure.
Instructions "GET_ERROR" and "GET_ERR_ID"	S7-1200/1500	The instructions give you the opportunity to obtain an error ID or detailed error information. You can use the error information for an access error, for example, to determine the parameter that caused the access error. In order for the instructions to output the required error information, they must be programmed in the user program for each individual block from which potential errors are to be evaluated. This option is ideal especially for custom libraries with error handling. If you work with the "GET_ERROR" or "GET_ERR_ID" instructions, no error OB is called and there is no entry in the diagnostic buffer. With this method of error handling, you actively intervene in the program sequence when an error occurs.
BR bit of the status word	S7-1200/1500	Using the BR bit of the status word, you can determine whether an error occurred when the instruction was executed. (BR bit = "0" => An error has occurred; BR bit = "1" => No error has occurred)

> **WARNING**
>
> **Output parameter RET_VAL**
>
> When errors occur in the supply of the input parameters during the execution of the instruction that contains the RET_VAL parameter, an invalid error code is output at the RET_VAL parameter and the output parameters of the instruction cannot be evaluated.

Example

The above-mentioned local error handling options can be programmed individually or in combination with one another. To ensure that each error scenario that can occur within your program is recognized, we recommend a combination of local error handling options, as shown in the example below.

For a more precise error analysis, in addition to the RET_VAL output parameter you can also use the instructions "GET_ERROR" or "GET_ERR_ID". These options provide you with error codes, the detailed explanations of which are available in the descriptions of the respective instructions.

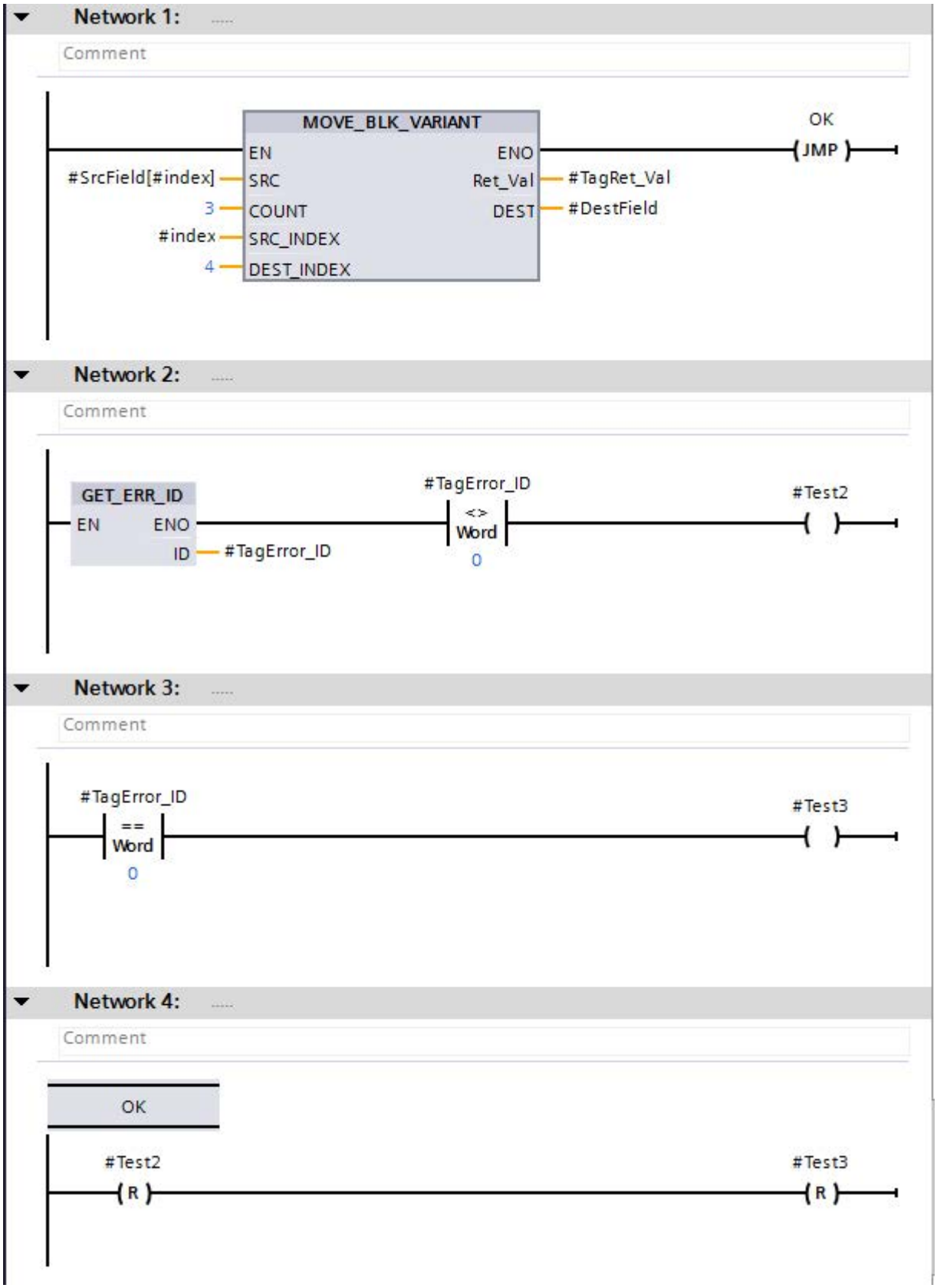
There are also error scenarios in which the RET_VAL output parameter does not output a valid error code. If an access error occurs while reading an input parameter, for example, the outputs of the instruction are no longer written, because the execution of the instruction is interrupted. In this case, we recommend that you integrate the two instructions "GET_ERROR" and "GET_ERR_ID" in your program because they provide reliable error information even when this type of error occurs.

> **WARNING**
>
> **Access error when reading an input parameter**
>
> The RET_VAL parameter does not return a valid error code and no detailed error information is output to the diagnostic buffer.

The following example shows you how you can recognize an access error when reading an input parameter:

Interface

	Name	Data type	Default value
1	Input		
2	SrcField	Array [1..100] of Real	
3	index	DInt	0
4	<Add new>		
5	Output		
6	TagOutput	Real	0.0
7	Test2	Bool	false
8	TagRet_Val	Int	0
9	DestField	Array [0..20] of Real	
10	Test3	Bool	false
11	<Add new>		
12	InOut		
13	Static		
14	Temp		
15	TagError_ID	Word	


Network 1:
Comment
MOVE_BLK_VARIANT
EN
ENO
#SrcField[#index]
SRC
Ret_Val
#TagRet_Val
3
COUNT
DEST
#DestField
#index
SRC_INDEX
4
DEST_INDEX
OK
JMP
Network 2:
Comment
GET_ERR_ID
EN
ENO
ID
#TagError_ID
#TagError_ID
<>
Word
0
#Test2
Network 3:
Comment
#TagError_ID
==
Word
0
#Test3
Network 4:
Comment
OK
#Test2
R
#Test3
R

In network 1, the instruction "MOVE_BLK_VARIANT": Move block" is called. The "SrcField" source area is accessed at the SRC parameter using a variable index. If no errors occur during the execution of the instruction, the ENO enable output returns signal state "1", and program execution jumps to network 4 and continues there.

If an access error occurs during the execution of the instruction, for example, due to the variable index, the instruction "GET_ERR_ID: Get error ID locally" in network 2 returns an error ID. The comparison of the error ID for "NOT EQUAL" with the value "0" returns the result "Test2" = TRUE.

The #TagRet_Val operand at the RET_VAL output parameter returns no valid error code in this case.

See also

Handling program execution errors (Page 2821)

GET_ERROR: Get error locally (Page 3388)

GET_ERR_ID: Get error ID locally (Page 3392)

Evaluating errors with output parameter RET_VAL (Page 3152)

Basics of the EN/ENO mechanism (Page 2447)

Parameter STATUS (Page 4566)

4.4.9 Using VARIANT instructions

The VARIANT data type

The VARIANT data type can be used to create generic function blocks or functions. It is used to interconnect the parameters of the block to tags of any type when called. Parameters of the VARIANT data type can be declared in the sections VAR_IN, VAR_IN_OUT and VAR_TEMP.

When the block is called, the type information of the tag is passed in addition to a pointer to the tag. The code of the block can then be executed according to its type with regard to the tags passed during runtime.

If, for example, a block parameter of a function has the VARIANT data type, then a tag of the integer data type can be passed at one point in the program, and a tag of the PLC data type can be passed at another point in the program. With the help of the VARIANT instructions, the function is then in a position to react to the situation without errors.

Procedure in the S7-300/400

The ANY data type can be used as follows for CPUs of the S7-300/400 series:

- In SCL and STL, memory of any kind can be passed at a block call when a pointer of the ANY data type has been programmed at a block parameter.
  However, important information about the structure of the memory is lost because the ANY pointer does not store the information that it points to a tag of the PLC data type, for example. The ANY pointer sees this as an ARRAY of BYTE.
- Parameters of the ANY data type can be passed to system function blocks (SFBs) or system functions (SFCs).
- In combination with the ANY data type, you can use the instruction "BLKMOV: Move block" to copy data of any data type in the program.

The ANY data type has the following restrictions:

- The ANY pointer has to manually constructed.
- No data with optimized block access can be passed.
- The ANY pointer cannot point to structures.

Procedure in S7-1500

In the TIA Portal, we recommend that you use the VARIANT data type instead of the ANY pointer.

Advantages of the VARIANT data type:

- Using the VARIANT data type, it is possible to write programs that perform dynamic type definition, i.e. integrated type checking during runtime of the program.
  The VARIANT data type loses the information about the data type of the tag to which it points. In other words, if the block parameter of a function of the VARIANT data type is supplied with a PLC data type, you can check within the function during runtime whether VARIANT points, for example, to "myUDTa" or "myUDTb".
- The VARIANT pointer can also be constructed symbolically. This improves the readability of your program code. You can find more information on the advantages of symbolic addressing here: Symbolic addressing (Page 247)
- Any data can be passed at a parameter of the VARIANT data type.
- The data type of the actual parameter to which the VARIANT points is determined during runtime. This gives you the possibility to program different responses in the program code depending on the data type of the actual parameter.

You can find more information about working with the VARIANT data type in the Siemens Industry online support under: Indirect addressing with the VARIANT data type (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=0&lang=en&referer=%2fWW%2f&func=cslib.csinfo&siteid=csius&groupid=4000002&extranet=standard&viewreg=WW&nodeid0=29156492&objaction=csopen)

The TIA Portal provides, among others, the following instructions of the VARIANT data type as parameters:

Basic instructions		
Programming language	Instruction	Description
LAD/FBD comparator operations	EQ_Type	Compare data type for EQUAL with the data type of a tag
	NE_Type	Compare data type for UNEQUAL with the data type of a tag
	EQ_ElemType	Compare element data type for EQUAL with the data type of a tag
	NE_ElemType	Compare element data type for UNEQUAL with the data type of a tag
	IS_NULL	Compare for EQUALS ZERO
	NOT_NULL	Compare for UNEQUALS ZERO
	IS_ARRAY	Check for ARRAY
SCL comparator operations	TypeOf	Check data type of a VARIANT tag
	TypeOfElements	Check element data type of a VARIANT tag
	IS_ARRAY	Check for ARRAY
LAD/FBD/SCL/GRAPH move operations	VariantGet	Read out VARIANT tag value
	VariantPut	Write VARIANT tag value
	CountOfElements	Get number of ARRAY elements
LAD/FBD/STL/SCL/GRAPH move operations	ReadFromArrayDB	Read from the array data block
	WriteToArrayDB	Write to the array data block
	ReadFromArrayDBL	Read from the array data block in the load memory
	WriteToArrayDBL	Write to array data block in the load memory
	BLKMOV	Move block
	UBLKMOV	Move block uninterruptible
	FILL	Fill block
	Deserialize	Move data type from ARRAY of BYTE (Deserialize)
	Serialize	Move data type to ARRAY of BYTE (Serialize)
	MOVE_BLK_VARIANT	Move block
SCL move operations	READ_LITTLE	Read data in little endian format
	WRITE_LITTLE	Write data in little endian format
	READ_BIG	Read data in big endian format
	WRITE_BIG	Write data in big endian format
LAD/FBD/STL/SCL conversion operations	VARIANT_TO_DB_ANY	Convert VARIANT to DB_ANY
	DB_ANY_TO_VARIANT	Convert DB_ANY to VARIANT

You can find additional instructions with the VARIANT data type in the online help under "Extended instructions".

Example

The following example in the SCL programming language shows you a queue (FIFO, First In First Out), which works with elements of any data type. Programmed using the instructions described above.

Declaration of the block interface of the "FIFOQueue" function block:

Declaration	Parameters	Data type	Comment
Input	request	BOOL	The instruction is executed when a positive signal edge is detected at the "request" parameter.
	mode	BOOL	0 = The first entry in the ring buffer is returned. 1 = An entry is written at the last position in the ring buffer.
	initialValue	VARIANT	Value with which the ARRAY of the ring buffer is initialized.
Output	error	INT	Error information
InOut	item	VARIANT	The entry that is either returned from the ring buffer or written to the ring buffer.
	buffer	VARIANT	An ARRAY that is used as a ring buffer.
Static	edgeupm	BOOL	Edge memory bit in which the RLO of the previous query is saved.
	firstItemIndex	INT	Index of the oldest entry in the ring buffer
	nextEmptyItemIndex	INT	Index of the next free entry in the ring buffer
Temp	edgeup	BOOL	Result of edge evaluation
	internalError	INT	Error information
	newFirstItemIndex	INT	Variable index
	newNextEmptyItemIndex	INT	Variable index
	bufferSize	UDINT	Number of ARRAY elements in the ring buffer

Program code of the "FIFOQueue" function block:

```
(* This program code section is only executed once after a positive signal edge. If there
is no change in the signal state of the RLO, the program execution of the "FIFOQueue" FB
ends. *)
#edgeup := #request & NOT #edgeupm;
#edgeupm := #request;
IF NOT (#edgeup) THEN
   RETURN;
END_IF;


// ------Validation of whether all parameter inputs are valid.----
(* This program code section checks whether the ring buffer is an ARRAY. If so, the number
of the ARRAY elements is read out. If it is not an ARRAY, the program execution is
terminated at this point and the error code "-10" is output. *)
IF NOT (IS_ARRAY(#buffer)) THEN
```

```
   #error := -10;
   RETURN;
ELSE
   #bufferSize := CountofElements(#buffer);
END_IF;

(* This program code section checks whether the data type of the ARRAY elements matches the
data type of the entry (tag #item). If the data types do not match, the program execution
is terminated at this point and the error code "-11" is output. *)
IF NOT (TypeOf(#item) = TypeOfElements(#buffer)) THEN
   #error := -11;
   RETURN;
END_IF;

(* This program code section checks whether the initial value of the ring buffer matches
the entry (tag #item). If the data types do not match, the program execution is terminated
at this point and the error code "-12" is output. *)
IF NOT (TypeOf(#item) = TypeOf(#initialValue)) THEN
   #error := -12;
   RETURN;
END_IF;

(* This program code section checks whether the variable indices are within the ARRAY
limits. If they are not, the program execution is terminated at this point and either error
code "-20" or "-21" is output depending on the index. *)
IF (#nextEmptyItemIndex >= #bufferSize) THEN
   #error := -20;
   RETURN;
END_IF;
IF (#firstItemIndex >= #bufferSize) THEN
   #error := -21;
   RETURN;
END_IF;


//-----------Program code execution, depending on the Mode parameter-------------
// The execution of the instructions depends on the signal state of the Mode parameter.
IF #mode = 0 THEN

// If the Mode parameter has the signal state "0", the first entry from the passed ring
buffer is returned.
(* This program code section checks whether the ring buffer is empty. If this is the case,
program execution is terminated at this point and the error code "-40" is output. *)
   IF (#firstItemIndex = -1) THEN
      #error := -40;
      RETURN;
```

```
   END_IF;

// This program code section returns the first entry of the ring buffer.
   #internalError := MOVE_BLK_VARIANT(SRC := #buffer,
                                      COUNT := 1,
                                      SRC_INDEX := #firstItemIndex,
                                      DEST_INDEX := 0,
                                      DEST => #item);

   IF (#internalError = 0) THEN
(* This program code section checks whether the ring buffer contains ARRAY elements. If it
does, the first entry is passed further on and the index is incremented by 1. *)
      #internalError := MOVE_BLK_VARIANT(SRC := #initialValue,
                                         COUNT := 1,
                                         SRC_INDEX := 0,
                                         DEST_INDEX := #firstItemIndex,
                                         DEST => #buffer);

// This program code section calculates the new index of the first entry.
      #newFirstItemIndex := #firstItemIndex +1;
      #newFirstItemIndex := #newFirstItemIndex MOD #bufferSize;

// This program section checks whether the ring buffer is empty.
      IF (#nextEmptyItemIndex = #newFirstItemIndex) THEN
// If the ring buffer is empty, the index is set to 0.
         #firstItemIndex := -1;
         #nextEmptyItemIndex := 0;
      ELSE
// The index of the first entry is changed.
         #firstItemIndex := #newFirstItemIndex;
      END_IF;
   END_IF;
ELSE


// If the Mode parameter has the signal state "1", the entry is written to the passed ring
buffer.
(* This program code section checks whether the ring buffer is full. If this is the case,
program execution is terminated at this point and the error code "-50" is output. *)
   IF (#nextEmptyItemIndex = #firstItemIndex) THEN
      #error := -50;
      RETURN;
   END_IF;

// This program code section writes the entry to the ring buffer.
```

```
   #internalError := MOVE_BLK_VARIANT(SRC := #item,
                                      COUNT := 1,
                                      SRC_INDEX := 0,
                                      DEST_INDEX := #nextEmptyItemIndex,
                                      DEST => #buffer);

   IF (#internalError = 0) THEN
// This program code section increments the index by 1 and calculates the new empty entry
index.
      #newNextEmptyItemIndex := #nextEmptyItemIndex +1;
      #newNextEmptyItemIndex := #newNextEmptyItemIndex MOD #bufferSize;
      #nextEmptyItemIndex := #newNextEmptyItemIndex;

(* This program code section checks which index the "#firstItemIndex" tag has. If the number
= -1, the ring buffer is initialized and the entry is written to the ring buffer. Therefore,
"0" must be assigned to the tag. *)
      IF (#firstItemIndex = -1) THEN
         #firstItemIndex := 0;
      END_IF;
   END_IF;
END_IF;

//-------------------------Local error handling----------------------------
(* This program code section checks whether a local error has occurred. If this is the case,
program execution is terminated at this point and the error code "-100" is output. *)
IF (#internalError > 0) THEN
   #error := -100;
   RETURN;
END_IF;

// If no error occurred during program execution, the error code "0" is output.
#error := 0;
```

See also

VARIANT (Page 2277)

TypeOf: Query data type of a VARIANT tag (Page 4159)

TypeOfElements: Check element data type of a VARIANT tag (Page 4160)

IS_ARRAY: Check for ARRAY (Page 4161)

MOVE_BLK_VARIANT: Move block (Page 4189)

First steps 5

5.1 Getting Started Documentation

Getting started with the TIA Portal

The Getting Started documentation is available to help you begin using the TIA portal.

The Getting Started documentation contains instructions which show you, step-by-step how to create a project in the TIA portal and give you the chance to get a quick overview of all the possibilities the TIA portal offers you.

Contents

The Getting Started documents describe the creation of a single, continuous project for STEP 7 and WinCC that is extended with each chapter. You start with simple basic functions, and use more complex ones as you continue with the creation of the project.

In addition to the step-by-step instructions, the Getting Started documents also give you background information that explains the functions used and illustrate how they relate to each other.

Target audience

The Getting Started documents are intended for beginners, but are also useful for users migrating from a previous version of SIMATIC STEP 7 and WinCC.

Download

The documentation is available, free of charge at the Service&Support (https://support.automation.siemens.com) portal in PDF form.

You can download the documents here:

- STEP 7 Basic and WinCC Basic (V10.5) (http://support.automation.siemens.com/WW/view/en/40263542/0/en)
- STEP 7 Professional and WinCC Advanced (V11) (http://support.automation.siemens.com/WW/view/en/28919804/133300)
- STEP 7 Professional and WinCC Advanced (V12) (http://www.automation.siemens.com/salesmaterial-as/interactive-manuals/getting-started_simatic-s7-1500/_content/EN/content_en.html)

Introduction to the TIA Portal 6

6.1 User interface and operation

6.1.1 Starting, setting and exiting the TIA Portal

6.1.1.1 Starting and exiting the TIA Portal

Starting the TIA Portal

Follow the steps below to start the TIA Portal:

1. In Windows, select "Start > Programs > Siemens Automation > TIA Portal V13".
   The TIA Portal opens with the last settings used.

Exiting the TIA Portal

Follow the steps below to exit the TIA Portal:

1. In the "Project" menu, select the "Exit" command.
   If the project contains any changes that have not been saved, you will be asked if you wish to save them.
   - Select "Yes" to save the changes in the current project and close the TIA Portal.
   - Select "No" to close the TIA Portal without saving the most recent changes in the project.
   - Select "Cancel" to cancel the closing procedure. The TIA Portal will remain open if you select this option.

6.1.1.2 Overview of the program settings

Overview

The following table shows the application settings that you can make:

Group	Setting	Description
General settings	User name	The user name of the user. The user name is stored in the project properties when a new project is created.
	User interface language	Language for the program interface

Group	Setting	Description
	Mnemonic	Specifies the mnemonics for programming: "German" uses the German mnemonics, for example, "E1.0". "International" uses international mnemonics, for example, "I1.0". For information on the differences in the mnemonics of the individual commands, refer to the description of the relevant programming language.
	Show list of recently used projects	Number of entries in the list of recently used projects in the "Project" menu
	Load most recent project during startup	The last opened project is opened automatically after the TIA Portal starts.
	Displaying truncated text in full	Texts which are truncated due to their length are displayed in a tooltip.
	Show tooltips (context-sensitive help is available)	Tooltips are displayed and you get context-sensitive help. If this function is disabled, you can open the tooltip with <F1>.
	Open cascade automatically in tooltips	After a brief time, the tooltips automatically expand to display a cascade containing additional help. If this option is cleared, the tooltips must be manually expanded.
Reset to default	All application settings	All changes that you made in the TIA Portal after installation are reversed.
	Layout of the editors	Resets the complete layout of the application to the factory state.
	Show all message windows	All message windows whose appearance was manually suppressed are displayed again.
Start view	Most recent view	Starts the program in the last view that was used. This can be either the portal view or the project view.
	Portal view	The TIA Portal will always be started in the portal view, irrespective of the last view you worked in.
	Project view	The TIA Portal will always be started in the project view, irrespective of the last view you worked in.
View for objects in overview	Details	When several views are available, the detail view opens by default, for example, in the overview window.
	List	When several views are available, the list view opens by default, for example, in the overview window.
	Thumbnails	When several views are available, the symbol view opens by default, for example, in the overview window.
Storage settings	Recently used storage location	When a project is saved the first time, the most recently used file path is set by default.
	Default storage location	Enables the specification of file paths for: • Projects • Libraries • Configuration file for corporate libraries
Data exchange	Storage location for data import	Imported files are searched for at this storage path by default.
	Storage location for data export	This is the default storage path for the data export.
	Storage location for support packages	When support packages are downloaded, they are stored in the specified storage path and can be installed from there.
	Storage location for log files	Log files are stored at the location specified here.

See also

Starting and exiting the TIA Portal (Page 273)
Resetting the user interface layout (Page 307)
Changing the settings (Page 276)
Configuring the display of tooltips and tooltip cascades (Page 327)

6.1.1.3 Overview of the script and text editor settings

Overview

The following table shows the available settings for script and text editors:

Group	Setting	Description
Font	Font type and size	Sets the font type and size for the text in text editors.
Font colors	Color settings	You can choose the colors for individual text elements from the respective drop-down lists in the text editors. Optional settings are available for the following text elements: • Text • Keywords • Comments • Operands • Scripts • Standard functions • Instructions/system functions • Constant strings • Numeric constants • Constant tags • Tags • Object models • Formal parameters
	Reset to default	Resets all font colors in editors to their factory settings.
Tabs	Tab width	Sets the width of tabs.
	Use tabs	Enables the use of tabs.
	Use spaces	Specifies use of space characters instead of tabs.
Indent	None	Text entries are not automatically indented.
	Block	The lines or the selected paragraph are automatically indented.
	Smart	The lines or the selected paragraph are automatically indented. All unnecessary spaces are also removed.
View	Show white spaces	Shows control characters within a text.
STL (Statement List)	Font type and size	Sets the font type and size for STL program code.
SCL (Structured Control Language)	Show line numbers	Shows the row numbers in SCL programs.

See also

Changing the settings (Page 276)

6.1.1.4 Overview of the print settings

Overview

The following table shows the available settings for printing:

Group	Setting	Description
General	Always print table data as pairs of values	Tables are printed as a list and not in tabular form. The corresponding values are listed for each column. Enable this option, for example, if you want to print a table that is too large for the print area.
Hardware configuration	Active graphic view	The graphics of network and device view are included in the printout.
	Active table	A table associated with an editor is included when printing with the editor.
PLC Programming	Zoom factor	Specifies the size in which blocks are to be printed out.
	With interface	The interfaces of blocks are included in the printout.
	With comments	Comments on blocks are included in the printout.
	With line numbers	The line numbers of the program code are printed for text-based programming languages.
Motion & Technology	Dialog display/graphic	The content of the editor is printed out as a graphic if the editor supports this.
	Table	The parameters of the technology objects are printed out in the form of a table.
HMI screens	Show tab order	In the printout you can specify the order in which the runtime objects can be selected with the TAB key.

See also

Changing the settings (Page 276)

6.1.1.5 Changing the settings

Procedure

To change the settings, proceed as follows:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "General" group in the area navigation to change the settings described in the previous sections. Or click on one of the other entries in area navigation to make settings for your installed products.
3. Change the settings.

Result

The change will be adopted immediately, there is no need to save it explicitly.

See also

Overview of the program settings (Page 273)
Overview of the script and text editor settings (Page 275)
Overview of the print settings (Page 276)

6.1.2 Layout of the user interface

6.1.2.1 Views

Views

Three different views are available for your automation project:

- The portal view is a task-oriented view of the project tasks.
- The project view is a view of the components of the project, as well as the relevant work areas and editors.
- The library view (Page 281) shows the elements of the project library and the open global libraries.

You can change over between the two views using a link.

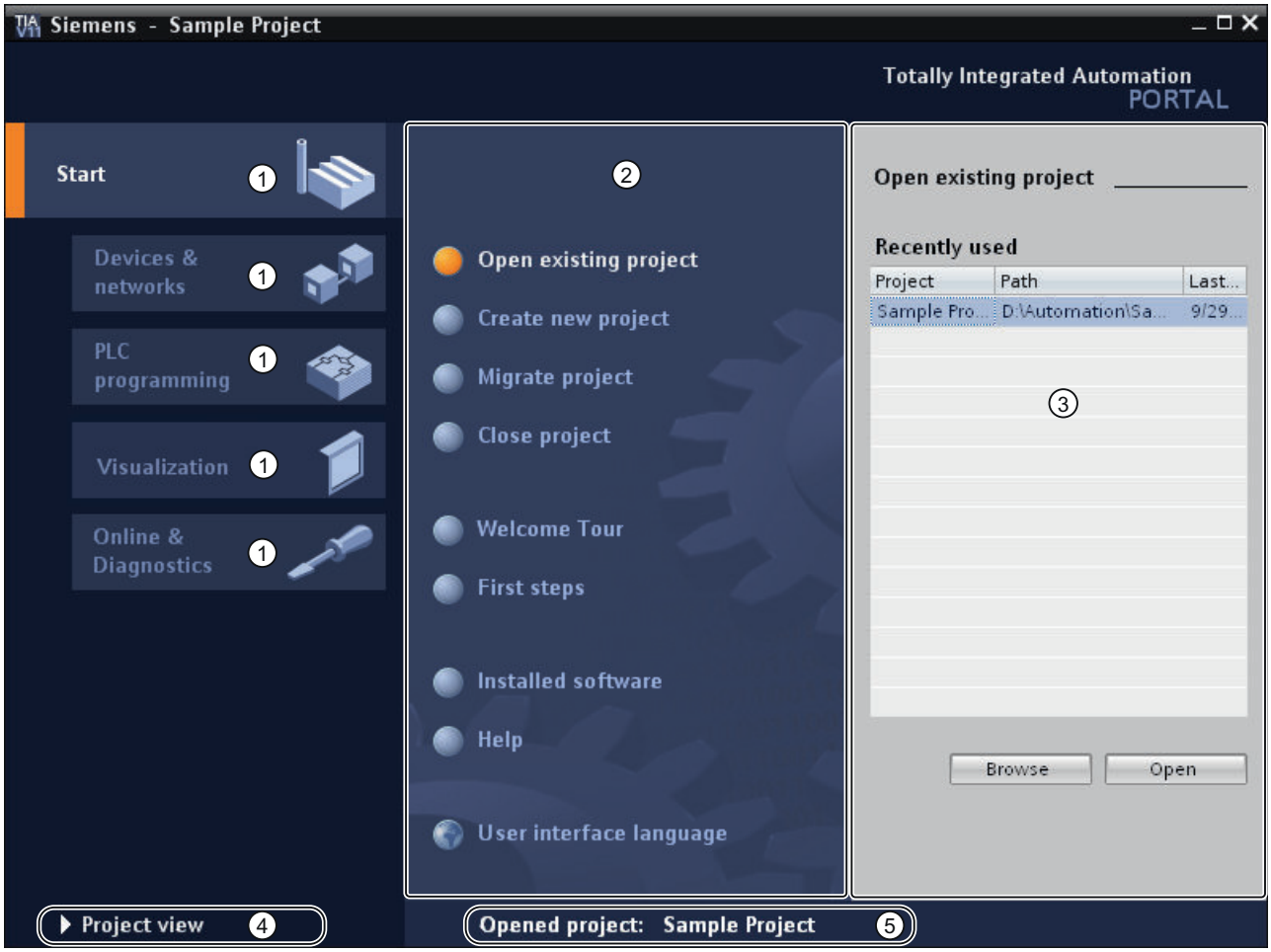
6.1.2.2 Portal view

Purpose of the portal view

The portal view provides you with a task-oriented view of the tools. Here, you can quickly decide what you want to do and call up the tool for the task in hand. If necessary, the view changes automatically to the project view (Page 279) for the selected task.

Layout of the portal view

The following figure shows an example of the components in the portal view:



① Portals for different tasks
② Actions for the selected portal
③ Selection panel for the selected action
④ Change to the project view
⑤ Display of the project that is currently open

Portals

The portals provide the basic functions for the individual task areas. The portals that are provided in the portal view depends on the products that have been installed.

Actions for the selected portal

Here, you will find the actions available to you in the portal you have selected. You can call up the help function in every portal on a context-sensitive basis.

Selection panel for the selected action

The selection panel is available in all portals. The content of the panel adapts to your current selection.

Change to the project view

You can use the "Project view" link to change to the project view.

Display of the project that is currently open

Here, you can obtain information about which project is currently open.

See also

Project tree (Page 282)
Basics of the work area (Page 285)
Inspector window (Page 293)
Basics on task cards (Page 295)
Details view (Page 299)

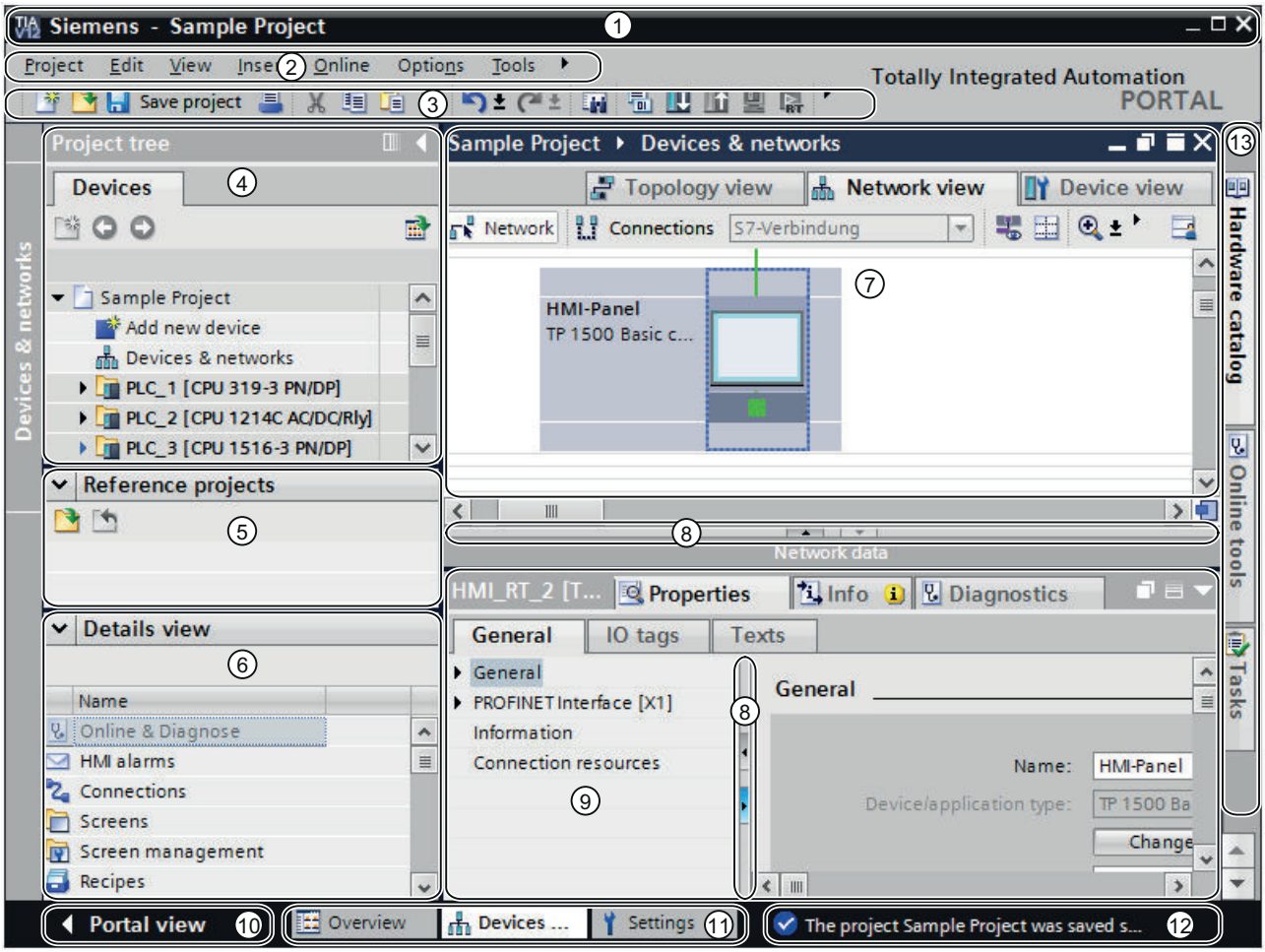
6.1.2.3 Project view

Purpose of the project view

The project view is a structured view of all components of the project.

Layout of the project view

The following figure shows an example of the components of the project view:



①	Title bar
②	Menu bar
③	Toolbar
④	Project tree (Page 282)
⑤	Reference projects (Page 297)
⑥	Details view (Page 299)
⑦	Work area (Page 295)
⑧	Dividers
⑨	Inspector window (Page 293)
⑩	Changing to the Portal view (Page 277)
⑪	Editor bar
⑫	Status bar with progress display
⑬	Task cards (Page 295)

Title bar

The name of the project is displayed in the title bar.

Menu bar

The menu bar contains all the commands that you require for your work.

Toolbar

The toolbar provides you with buttons for commands you will use frequently. This gives you faster access to these commands.

Dividers

Dividers separate individual components of the program interface. The arrows on the dividers allow you to display and hide the adjacent sections of the user interface.

Changing to the portal view

You can use the "Portal view" link to change to the portal view.

Editor bar

The Editor bar displays the open editors. If you have opened a lot of editors, they are shown grouped together. You can use the Editor bar to change quickly between the open elements.

Status bar with progress display

In the status bar, you will find the progress display for processes that are currently running in the background. This also includes a progress bar that shows the progress graphically. Hover the mouse pointer over the progress bar to display a tooltip providing additional information on the active background process. You can cancel the background processes by clicking the button next to the progress bar.

If no background processes are currently running, the status bar displays the last generated alarm.

See also

Basics of the work area (Page 285)

6.1.2.4 Library view

Function of the library view

The library view provides an overview of the elements in the project library and the open global libraries. You can switch to the library view using the "Libraries" task card.

See also: Overview of the library view (Page 420)

6.1.2.5 Project tree

Function of the project tree

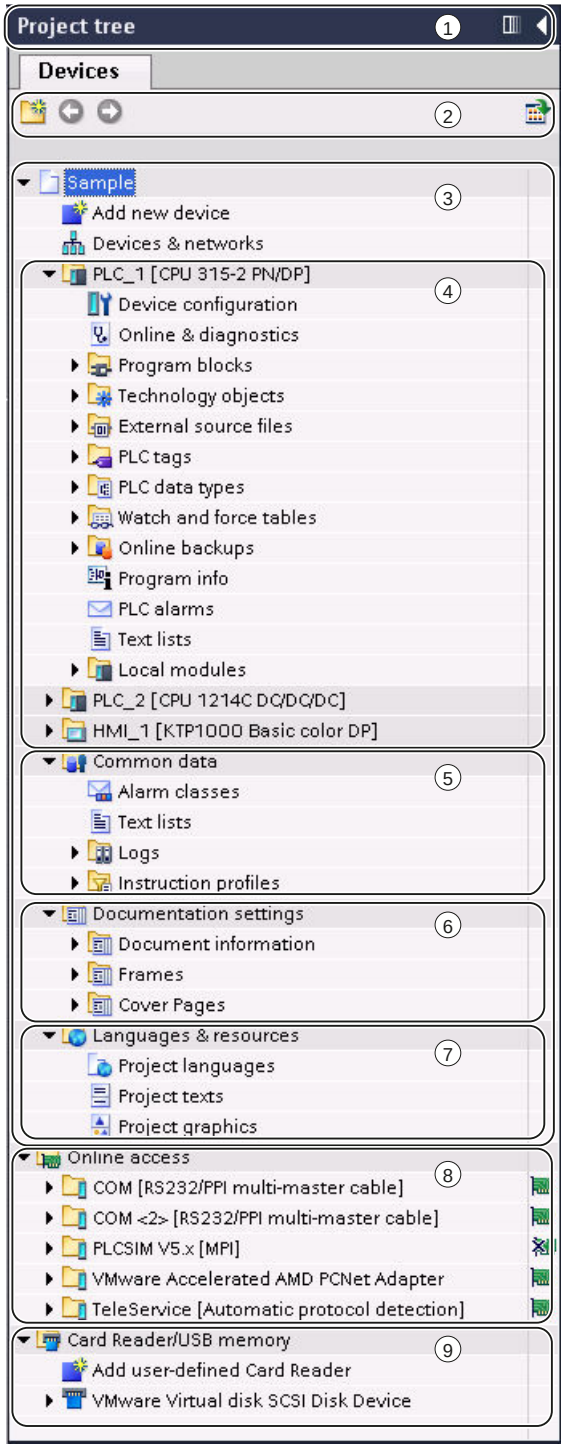
Using the project tree features gives you access to all components and project data. You can perform the following tasks in the project tree:

- Add new components
- Edit existing components
- Scan and modify the properties of existing components

You can select the objects of the project tree either with the mouse or via the keyboard by typing the first letter of the desired object. If more than one object begins with the same letter, the next lower object is selected. The project tree must be the focused user interface element in order for you to select an object with its initial letter.

Layout of project tree

The following figure shows an example of the project tree components:



① Title bar
② Toolbar
③ Project
④ Devices
⑤ Common data
⑥ Documentation settings
⑦ Languages & resources
⑧ Online access
⑨ Card Reader / USB memory

Title bar

The title bar of the project tree has a button for automatically and manually collapsing the project tree. Once it is collapsed manually, the button is "Reduced" to the left-hand margin. It changes from an arrow pointing left to one that is pointing right, and can now be used to reopen the project tree. You can use the "Reduce automatically" button collapse to project tree automatically when you do not need it.

See also: Maximizing and minimizing the work area (Page 287)

Toolbar

You can do the following tasks in the toolbar of the project tree:

- Create a new user folder; for example, in order to group blocks in the "Program blocks" folder.
- Navigate forward to the source of a link and back to the link itself.
  There are two buttons for links in the project tree. You can use these to navigate from the link to the source and back.
- Show an overview of the selected object in the work area.
  When the overview is displayed, the lower-level objects and actions of the elements in the project tree are hidden.

Project

You will find all the objects and actions related to the project in the "Project" folder, e.g.:

- Devices
- Languages & resources
- Online access

Device

There is a separate folder for each device in the project, which has an internal project name. Objects and actions belonging to the device are arranged inside this folder.

Common data

This folder contains data that you can use across more than one device, such as common message classes, logs, scripts and text lists.

Documentation settings

In this folder, you can specify the layout for project documentation to be printed at a later point.

Languages & resources

You can determine the project languages and texts in this folder.

Online access

This folder contains all the interfaces of the programming device / PC, even if they are not used for communication with a module.

Card Reader / USB memory

This folder is used to manage all card readers an other USB storage media connected to the programming device / PC.

See also

Portal view (Page 277)

Project view (Page 279)

Basics of the work area (Page 285)

Inspector window (Page 293)

Basics on task cards (Page 295)

Details view (Page 299)

6.1.2.6 Work area

Basics of the work area

Function of the work area

The objects that you can open for editing purposes are displayed in the work area. These objects include, for example:

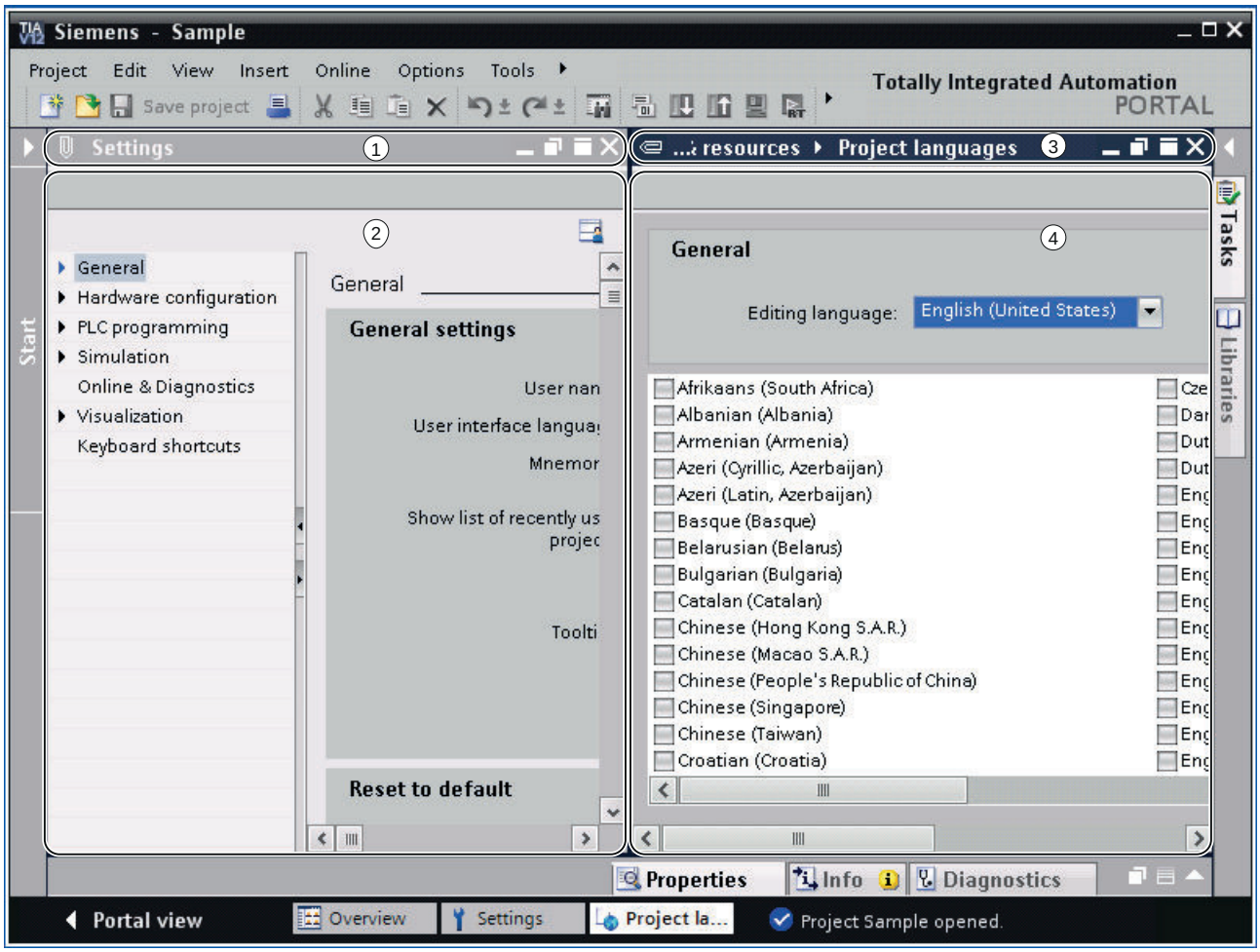
- Editors and views
- Tables

You can open several objects. However, normally it is only possible to see one of these at a time in the work area. All other objects are displayed as tabs in the Editor bar. If, you would

like to view two objects at the same time when performing certain tasks, you can tile the work area either horizontally or vertically or undock elements of the work area. If no objects are open, the work area will be empty.

Layout of the work area

The following figure shows an example of a vertically split work area:



① Title bar of left-hand editor
② Work area of left-hand editor
③ Title bar of right-hand editor
④ Work area of right-hand editor

See also

Maximizing and minimizing the work area (Page 287)

Splitting the work area (Page 289)

Floating the work area elements (Page 289)

Using grouped elements of the work area (Page 290)

Minimizing and maximizing elements of the work area (Page 292)

Switching between the elements in the work area (Page 292)

Saving a layout of editors and tables (Page 306)

Save user interface layout (Page 304)

Maximizing and minimizing the work area

You have the option to adapt the work area to make it as large as possible. You can use the following function for this:

- Maximizing the work area
  You can close the task cards, project tree and inspector window with a single click. This increases the size of the work area. You can minimize the work area again at any time in order to return to the previous view.
- Collapsing task cards, project tree, and Inspector window automatically
  You can use the "Collapse automatically" option for the task cards, project tree, and Inspector window. This function causes these items to collapse automatically when you don't need them.

Maximizing and minimizing the work area

To maximize the work area, follow these steps:

1. Open an element such as an editor or a table.
   The element appears in the work area.
2. Click the "Maximize" button in the title bar of the element.
   The task cards, project tree and inspector window collapse, and the work area is shown with its maximum dimensions.

To minimize the work area again, follow these steps:

1. Click the "Embed" button in the title bar of the displayed element.
   This restores the view that existed before the work area was maximized. That is, if the task cards, project tree, or Inspector window were expanded before, they will be expanded again.

Collapsing task cards, project tree, and Inspector window automatically

To collapse the task cards automatically, follow these steps:

1. Click "Collapse automatically" in the title bar of the task cards.
   The task cards collapse when you click anywhere outside the task cards.
2. To use the task cards, click the collapsed task cards.
3. The task cards expand and are available for use. The "Collapse automatically" option remains enabled.

To collapse the project tree automatically, follow these steps:

1. Click "Collapse automatically" in the title bar of the project tree.
   The project tree collapses when you click anywhere outside the project tree.
2. To use the project tree, click the collapsed project tree.
   The project tree expands and is available for use. The "Collapse automatically" option remains enabled.

To collapse the Inspector window automatically, follow these steps:

1. Click "Collapse automatically" in the title bar of the Inspector window.
   The Inspector window collapses when you click anywhere outside the Inspector window.
2. To use the Inspector window, click the collapsed Inspector window.
   The Inspector window expands and is available for use. The "Collapse automatically" option remains enabled.

To disable the automatic collapse option, follow these steps:

1. Click "Expand permanently" again in the relevant window.
   The Collapse automatically" option is disabled, and the window remains expanded.

See also

Basics of the work area (Page 285)

Splitting the work area (Page 289)

Floating the work area elements (Page 289)

Using grouped elements of the work area (Page 290)

Minimizing and maximizing elements of the work area (Page 292)

Switching between the elements in the work area (Page 292)

Saving a layout of editors and tables (Page 306)

Splitting the work area

You can split the work area vertically or horizontally.

Procedure

To split the work area vertically or horizontally, follow these steps:

1. In the "Window" menu, select the "Split editor space vertically" or "Split editor space horizontally" command.
   The element you have clicked and the next element in the Editor bar will be displayed either next to one another or one above the other.

Note

If no elements are open in the work area, the "Split editor space vertically" and "Split editor space horizontally" functions will not be available.

See also

Basics of the work area (Page 285)

Maximizing and minimizing the work area (Page 287)

Floating the work area elements (Page 289)

Using grouped elements of the work area (Page 290)

Minimizing and maximizing elements of the work area (Page 292)

Switching between the elements in the work area (Page 292)

Saving a layout of editors and tables (Page 306)

Floating the work area elements

You can float work area elements in their own separate window:

- Editors
- Tables
- Setting windows
- Task cards
- Inspector window

You can embed floating elements again in the work area at any time.

Note

Properties of elements in a floating window

The properties of elements that you have selected in a floating window are only displayed in the Inspector window if the Inspector window is floating as well.

Floating the work area elements

To float work area elements, follow these steps:

1. Click the "Float" button in the title bar of the element.
   The element will be released from the work area and displayed in its own window. You can now place the window wherever you wish. If you have minimized the window, you can restore it via the editor bar.

Embedding elements in the work area

To embed elements in the work area again, follow these steps:

1. Click the "Embed" button in the title bar of the element.
   The element will appear in the work area again.

See also

Basics of the work area (Page 285)

Maximizing and minimizing the work area (Page 287)

Splitting the work area (Page 289)

Using grouped elements of the work area (Page 290)

Minimizing and maximizing elements of the work area (Page 292)

Switching between the elements in the work area (Page 292)

Saving a layout of editors and tables (Page 306)

Using grouped elements of the work area

If you open more than five elements of the same type, e.g., editors or tables, they are grouped in the editor bar. You can use these groups as follows:

- Displaying individual elements of a group
- Displaying all elements of a group in separate windows
- Embedding all displayed elements of a group in the work area
- Minimizing all displayed elements
- Closing all elements of a group

Displaying individual elements of a group

To display individual elements of a group, follow these steps:

1. In the editor bar, click the group containing the element you want to display.
   All list of all available elements of the group is displayed.
2. Click the element that you want to display.

Displaying all elements of a group in separate windows

To display all elements of a group in separate windows, follow these steps:

1. In the editor bar, right-click the group whose elements you want to display.
2. Select "Restore group" in the shortcut menu.
   All elements of the group are displayed in separate, overlapping windows. Move the windows in order to see the individual element, or choose an element via the group in the editor bar.

Embedding all displayed elements of a group in the work area

To embed all elements of a group displayed in separate windows in the work area again, follow these steps:

1. In the editor bar, right-click the group whose elements you want to embed.
2. Select "Embed group" in the shortcut menu.
   All elements of the group are embedded in the work area again.

Minimizing all displayed elements

To minimize all elements of a group, follow these steps:

1. In the editor bar, right-click the group whose elements you want to minimize.
2. Select "Minimize group" in the shortcut menu.
   All elements of the group are minimized. However, the minimized elements remain open and can be quickly maximized again via the group in the editor bar.

Closing all elements of a group

To close all elements of a group, follow these steps:

1. In the editor bar, right-click the group whose elements you want to close.
2. Select "Close group" in the shortcut menu.
   All elements of the group are closed. The group is removed.

See also

Basics of the work area (Page 285)

Maximizing and minimizing the work area (Page 287)

Splitting the work area (Page 289)

Floating the work area elements (Page 289)

Minimizing and maximizing elements of the work area (Page 292)

Switching between the elements in the work area (Page 292)

Saving a layout of editors and tables (Page 306)

Minimizing and maximizing elements of the work area

You can minimize the elements that are open in the work area, such as editors or tables, as needed. However, an element remains open even if it has been minimized, and can quickly be maximized again using the editor bar.

Minimizing elements in the work area

To minimize elements in the work area, follow these steps:

1. Click the "Minimize" button in the title bar of the element.
   The element is minimized, but can still be accessed via the editor bar.

To minimize all elements at the same time, follow these steps:

1. In the "Window" menu, select the "Minimize all" command.

Maximizing elements in the work area

To maximize elements in the work area again, follow these steps:

1. Click the required element in the editor bar.
   The element is maximized and appears in the work area.

See also

Basics of the work area (Page 285)

Maximizing and minimizing the work area (Page 287)

Splitting the work area (Page 289)

Floating the work area elements (Page 289)

Using grouped elements of the work area (Page 290)

Switching between the elements in the work area (Page 292)

Saving a layout of editors and tables (Page 306)

Switching between the elements in the work area

You can switch between the elements in the work area at any time.

Switching between the elements in the work area

To switch to the previous or next editor, follow these steps:

1. In the "Window" menu, select the "Next editor" or "Previous editor" command.
   The next or previous editor will be displayed.

See also

Basics of the work area (Page 285)

Maximizing and minimizing the work area (Page 287)

Splitting the work area (Page 289)
Floating the work area elements (Page 289)
Using grouped elements of the work area (Page 290)
Minimizing and maximizing elements of the work area (Page 292)
Saving a layout of editors and tables (Page 306)

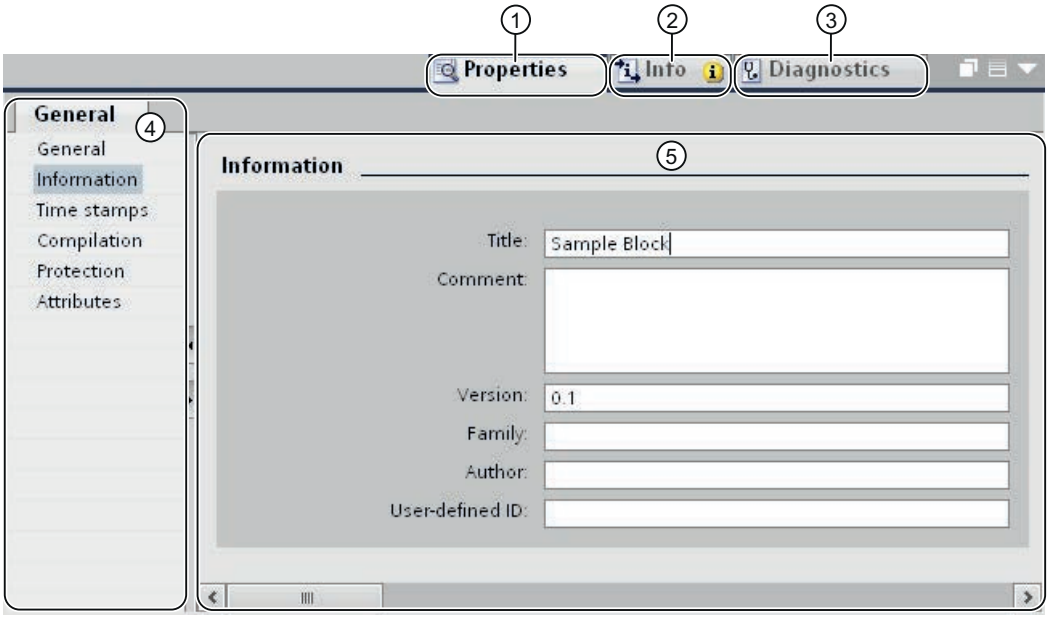
6.1.2.7 Inspector window

Function of the Inspector window

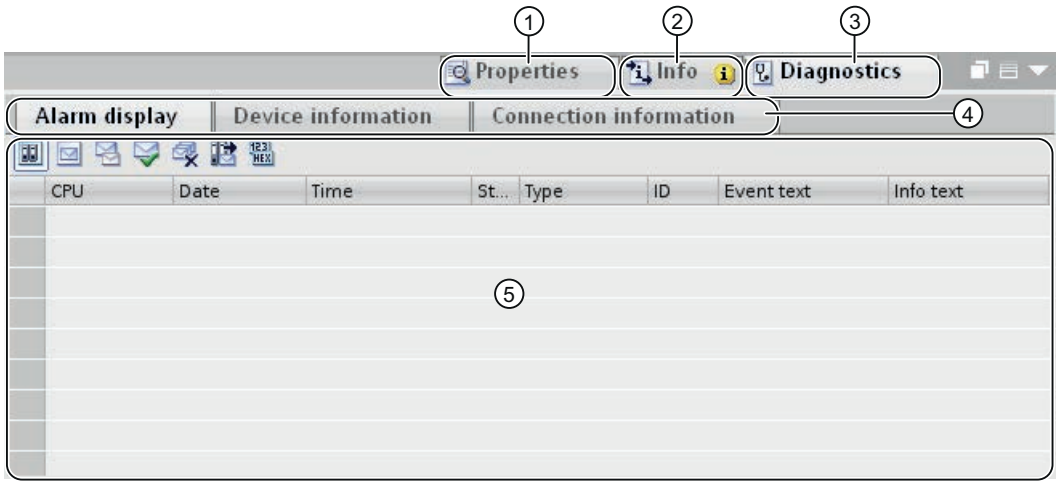
Additional information on an object selected or on actions executed are displayed in the inspector window.

Layout of the Inspector window

The following figures show the components of the Inspector window:



① "Properties" tab
② "Info" tab
③ "Diagnostics" tab
④ Area navigation within the "Properties" tab
⑤ Content of the "Properties" tab



① "Properties" tab
② "Info" tab
③ "Diagnostics" tab
④ Navigation within the tabs (only available in the "Info" and "Diagnostics" tabs)
⑤ Content of the "Diagnostics" tab

"Properties" tab

This tab displays the properties of the object selected. You can change editable properties here.

"Info" tab

This tab displays additional information on the object selected, as well as alarms on the actions executed (such as compiling).

"Diagnostics" tab

This tab provides information on system diagnostics events, configured alarm events, and connection diagnostics.

Navigation within the tabs

You can use area navigation and the lower-level tabs to display the information you require within the tabs.

See also

Project tree (Page 282)

Basics of the work area (Page 285)

Portal view (Page 277)

Project view (Page 279)

Basics on task cards (Page 295)
Details view (Page 299)

6.1.2.8 Task cards

Basics on task cards

Function of task cards

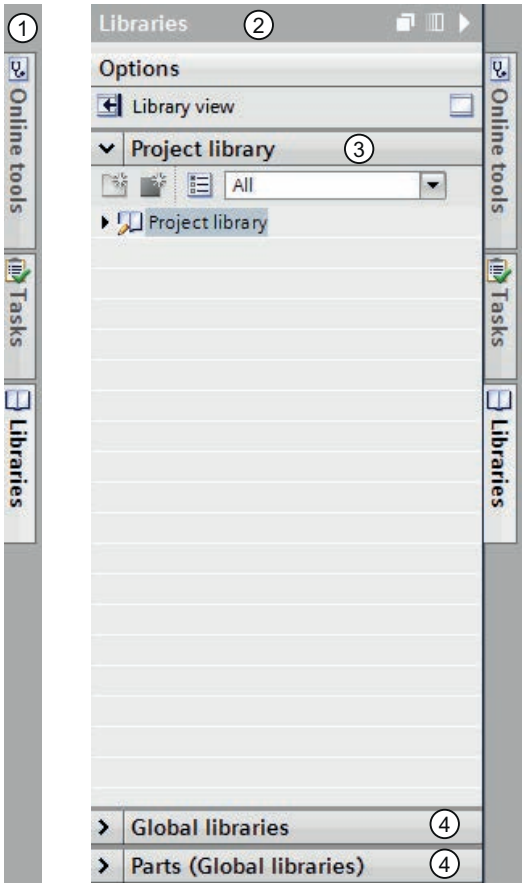
Depending on the edited or selected object, task cards that allow you perform additional actions are available. These actions include:

- Selecting objects from a library or from the hardware catalog
- Searching for and replacing objects in the project
- Dragging predefined objects to the work area

The task cards available can be found in a bar on the right-hand side of the screen. You can collapse and reopen them at any time. Which task cards are available depends on the products installed. More complex task cards are divided into panes that you can also collapse and reopen.

Layout of task cards

The following figure shows an example of the bar with the task cards:



① Task cards closed
② Task card open
③ Opened palette of a task card
④ Closed palette of a task card

See also

Changing the pane mode (Page 297)
Project tree (Page 282)
Basics of the work area (Page 285)
Inspector window (Page 293)
Portal view (Page 277)
Project view (Page 279)
Details view (Page 299)

Changing the pane mode

You can choose between two pane modes:

- Single pane mode:
  Only one pane is open at any given time. If you open another pane, the previously opened pane is closed automatically.
- Multi-pane mode:
  You can open several panes at the same time.

Procedure

To change the pane mode, follow these steps:

1. Click the "Change pane mode" button above the panes inside a task card.

See also

Basics on task cards (Page 295)

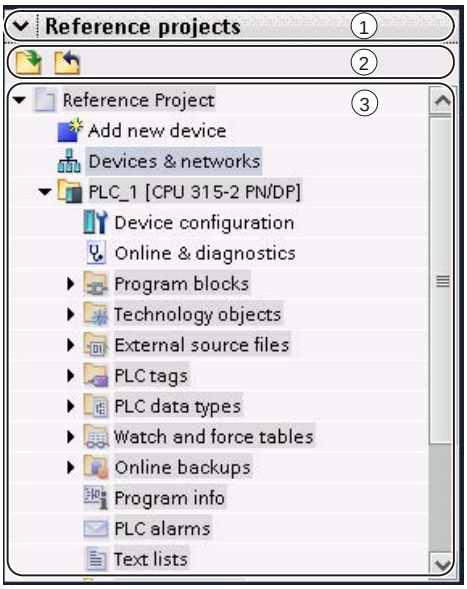
6.1.2.9 Reference projects

Function of reference projects

In the "Reference projects" palette, you can open other projects in addition to the current project. These reference projects are write-protected and cannot be edited. However, you can drag the objects of a reference project into your current project and further edit them there. You can also compare the objects of a reference project to the objects of your current project.

Layout of the "Reference projects" palette

The following figure shows the layout of the "Reference projects" palette:



① Title bar

② Toolbar

③ Opened reference projects

Title bar

The arrow for closing the palette is located in the title bar of the "Reference projects" palette. Once it is closed, the direction in which the arrow is pointing changes from downwards to right. It can now be used to reopen the palette.

Toolbar

The toolbar contains buttons for opening and closing reference projects.

Opened reference projects

Opened reference projects are displayed as read-only with their objects and their hierarchical structure.

See also

Basics of reference projects (Page 351)

Opening and closing a reference project (Page 351)

6.1.2.10 Details view

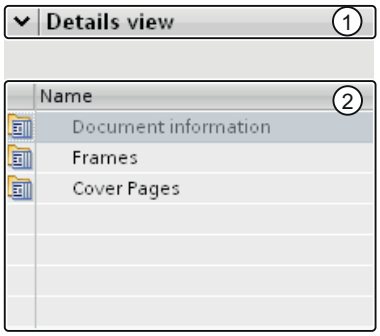
Purpose of the details view

The detail view shows certain content of the selected object is in the overview window or in the project tree. This might include text lists or tags.

The content of folders is not shown, however. To display the content of folders, use the project tree or the Inspector window.

Layout of the details view

The following figure shows an example of the details view:



① Title bar

② Content of the selected object

Title bar

The arrow for closing the details view is located in the title bar of the details view. After it has closed, the direction in which the arrow is pointing changes from left to right. It can now be used to reopen the details view.

Objects

The displayed content varies depending on the selected object. You can move the content of objects from the details view to the required location using drag-and-drop.

See also

Project tree (Page 282)

Basics of the work area (Page 285)

Inspector window (Page 293)

Basics on task cards (Page 295)

Portal view (Page 277)
Project view (Page 279)

6.1.2.11 Overview window

Overview window

Functions of the Overview window

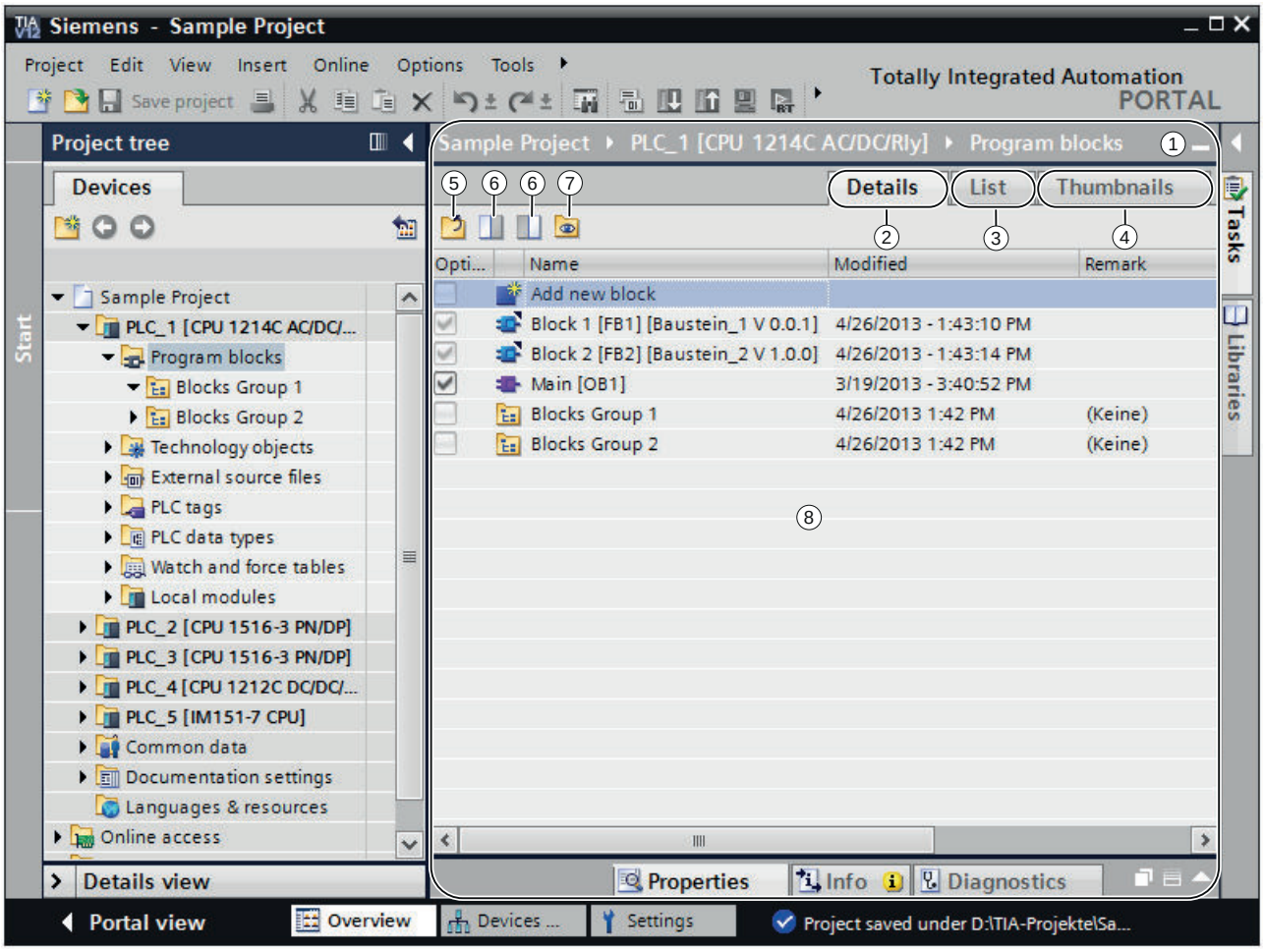
The Overview window supplements the project tree. The Overview window shows the contents of the folder currently selected in the project tree.

In addition, you can perform the following actions in the Overview window:

- Open objects
- Display and edit the properties of objects in the Inspector window
- Rename objects
- Call object-specific actions from the shortcut menu
- Compare objects side by side
- Perform various object operations, such as inserting objects from the library via drag-and-drop and moving, copying, pasting, and deleting objects

Layout of the Overview window

The following figure shows the components of the Overview window:



① Overview window

② Switch to the Details view

③ Switch to the List view

④ Switch to the Icon view

⑤ Move to higher level

⑥ Split the overview window in two. Either the right or left half of the overview window is synchronized. Clicking again cancels the split.

⑦ All elements within a selected folder are displayed even if these are located in lower-level groups. This option is only available in details view.

⑧ Contents of the object selected in the project tree.

Display forms of the Overview window

The content of the Overview window can be displayed as follows:

- Details view
  The objects are displayed in a list with additional information, such as the date of the last change.
- List view
  The objects are displayed in a simple list.
- Icon view
  The objects are displayed as icons according to category.

See also

Comparing objects in the overview window (Page 302)

Sorting the details view of the overview window (Page 303)

Overview of the library view (Page 420)

Comparing objects in the overview window

You can display the contents of two folders or objects side by side in the Overview window. The Overview window is split in half and you can display different contents on the left and right sides.

In addition, you can use a drag-and-drop operation to move objects between the split windows. Thus, for example, you can move contents from one window to the other.

Procedure

To split the Overview window in half or cancel the split, follow these steps:

1. In the toolbar, click on the "Synchronize left side" or "Synchronize right side" icon to split the overview window. Either the left or the right side of the overview window synchronized with the contents of the selected object in the project tree.
2. To cancel the split, click again on the previously selected icon.

See also

Overview window (Page 300)

Sorting the details view of the overview window

You have several options for adapting the display in the details view of the overview window:

- Adding additional columns
  Some columns are hidden by default to increase clarity. You can display hidden columns if needed. The columns available depend on the selected object.
- Display of folder contents in a flat hierarchy
  Folder contents can be displayed in a flat hierarchy. All the content is displayed at once even if it is located in different groups.
- Sorting the table columns
  You can sort individual columns of the table in ascending or descending order.

Showing or hiding columns

To show or hide additional table columns, follow these steps:

1. Right-click the title bar of the table.
2. Select the "Show/Hide" command in the shortcut menu, and select the columns you want to display.

Displaying folder contents in flat hierarchy

To display the content of a folder in a flat hierarchy, follow these steps:

1. Select the required folder in the project tree or in the library navigation of the library view.
2. Click the "Show subordinate elements" icon in the toolbar.
   All elements are displayed at once in the table even if they are located in subfolders.

Sorting a table in ascending or descending order

To sort the table by a column in ascending or descending order, follow the steps below:

1. Click the table header of a column if you want to sort the column in ascending order.
2. Click again on the same column of the table header to sort the column in descending order.
3. Click a third time on the table header of the same column to cancel the sorting.

See also

Overview window (Page 300)

6.1.2.12 User interface layout

Save user interface layout

Options for saving the user interface layout

When you make a change to the user interface, this is retained even after a restart of the TIA portal. A change to the user interface layout includes, for example, moving a window or adjusting the size of an editor.

In addition to the automatic saving of the user interface layout, you have the option of saving certain layouts:

- Saving the window layout
  You can save the layouts of the windows and editors of the TIA portal manually and restore these at a later time. It is possible to call five window layouts using a key combination. Use this function, for example, if you are work with a notebook which you connect to an external monitor when necessary. You can create a window layout for mobile use on the notebook display and another layout for when you work at the office with an external monitor.
- Save the layout within editors
  With some editors, you can adjust the display. You can, for example, adjust the width of tables or show or hide individual table columns.

See also

Save window layout (Page 304)

Load window layout (Page 305)

Managing window layouts (Page 306)

Saving a layout of editors and tables (Page 306)

Resetting the user interface layout (Page 307)

Basics of the work area (Page 285)

Save window layout

You can save the current window layout in order to call it again in the same form at a later time.

Procedure

To save a window layout, follow these steps:

1. Arrange all windows in the way in which you want to save them.
2. In the "Window" menu, select the "Save window layout as" command.
   The "Save window layout" dialog box appears.
3. Enter a name for the window layout in the "Name" field.

4. Enter a description of the window layout in the "Description" field in order to be able to identify the window layout more easily later.
5. Click "Save".

Result

The new window layout is saved in the last position after the existing saved window layouts. The first five window layouts can be called using a key combination.

See also

Save user interface layout (Page 304)

Load window layout

If you have already saved a window layout, you can load this, allowing you to quickly adjust your work environment to the respective conditions. You can load the first five window layouts using quick access via the "Window" menu or via a key combination.

If you load a window layout and then make changes to the arrangement of the window, you can restore the originally saved window layout.

Using quick access to load window layouts 1 to 5

To load one of the first five saved window layouts, follow these steps:

1. In the "Window" menu, select a window layout or select the key combination <Alt+Shift+[1 ... 5]>.

Loading additional window layouts

To load a window layout that is not among the first five window layouts, follow these steps:

1. In the "Window" menu, select the "Additional window layouts" command.
   The "Manage window layouts" dialog box appears.
2. Select the desired window layout.
3. Click "OK".

Restore window layout

To go back to a saved window layout, follow these steps:

1. In the "Window" menu, select the "Restore window layout" command or select the key combination <Alt+Shift+0>.

See also

Save user interface layout (Page 304)

Managing window layouts

You can carry out the following actions with existing window layouts:

- Changing the order of window layouts
  The order of the window layouts is important, as the first five window layouts can be called directly via the "Window" menu and via a key combination.
- Select a window layout
  If a window layout is not one of the first five window layouts, you can call it using the "Manage window layouts" dialog box.
- Deleting window layouts

Procedure

To manage the existing window layouts, follow these steps:

1. In the "Window" menu, select the "Manage window layouts" command.
   The "Manage window layouts" dialog box appears.
2. Select the window layout which you want to modify.
3. Click the "Up" or "Down" symbol to move the window layout up or down.
4. Click the "Delete" symbol to delete the selected window layout.
5. Click "OK".
   The selected window layout is activated.

See also

Save user interface layout (Page 304)

Saving a layout of editors and tables

You have the option of adapting editors and tables to meet your requirements. For example, you can hide columns in tables that you don't need. You can then save your customized view.

Procedure

To save the layout of editors and tables in the work area, follow these steps:

1. Adapt the editor or table according to your requirements.
2. Click the "Remember Layout" button in the editor or table.

Result

The layout is saved. When you reopen the editor or table, this layout will be used.

See also

Basics of the work area (Page 285)

Maximizing and minimizing the work area (Page 287)

Splitting the work area (Page 289)

Floating the work area elements (Page 289)

Using grouped elements of the work area (Page 290)

Minimizing and maximizing elements of the work area (Page 292)

Switching between the elements in the work area (Page 292)

Save user interface layout (Page 304)

Resetting the user interface layout

Every change you make to the layout of the user interface is saved. The changes are thus available even after a restart of the TIA Portal. For example, if you change the height and width of a text editor or the division of a table, your changes are retained so that you don't have to re-customize elements every time.

In some cases, however, it may be helpful to restore the original layout settings; for example, if another user prefers a different arrangement of the user interface.

Procedure

To reset the user interface settings to the default, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "General" group in the area navigation.
3. Click the "Reset to default" button under "Reset to default > Editor layout".

Result

The default settings for the user interface are restored.

See also

Overview of the program settings (Page 273)

Save user interface layout (Page 304)

6.1.3 Keyboard operation in the TIA Portal

6.1.3.1 Operating the TIA Portal with the keyboard

You can navigate through the TIA Portal using the keyboard, for example if you do not have a mouse available at the given moment. Many functions are also accessible via keyboard shortcuts. You can find an overview of all keyboard shortcuts in the settings for the TIA Portal.

In the following sections, you will learn how to navigate in the TIA Portal using the keyboard, edit objects and customize the TIA Portal to your needs.

See also

Displaying an overview of all keyboard shortcuts (Page 308)

6.1.3.2 Displaying an overview of all keyboard shortcuts

You can display an overview of all keyboard shortcuts.

Procedure

To display an overview of all available keyboard shortcuts, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The settings of the TIA Portal are displayed.
2. Open the "Keyboard shortcuts" entry in the area navigation.
   You can see an overview of all keyboard shortcuts, which are valid for the currently installed products.

6.1.3.3 Basic functions of the TIA Portal

Below, you will learn how you can use the basic functions of the TIA Portal using only your keyboard.

Operating basic functions of the TIA Portal with a keyboard

The following table shows how you can access basic functions of the TIA Portal with keyboard shortcuts:

Function	Keyboard shortcuts	Menu command
Switch between the project view and the portal view	<Alt+F7>	
Open the Help system If you need help on the TIA Portal, press <F1>.	<F1>	Help > Show help
Cancel the current action	<Esc>	
Find	<Ctrl+F>	
Replace an object	<Ctrl+H>	

Function	Keyboard shortcuts	Menu command
Find next If you have started a search, you can jump to the next hit with <F3>	<F3>	
Print object	<Ctrl+P>	Project > Print

Operating menus

The following table shows how you can navigate through menus using the keyboard:

Function	Keyboard shortcuts
Start keyboard operation in the menu You can access the menu using the <Alt> key and then continue to navigate with the arrow keys to scroll through the menu. Confirm your selection of the menu command with <Return>.	<Alt>
Go directly to a specific menu You can go directly to an individual menu command by holding down the <Alt> key. There is an underlined letter for each menu command. Press the <Alt> key along with the underlined letter.	<Alt+underlined letter in respective menu>
Open shortcut menu of an object With the shortcut menu key (on Microsoft Windows compatible keyboards), you can open the shortcut menu of the selected object. Alternatively, you can use <Shift+F10> if you are not using a Microsoft Windows compatible keyboard. You can use the arrow keys to scroll through the shortcut menu and select a menu command with <Return>.	<Shortcut menu key> Alternative: <Shift+F10>

Operating expandable elements

The following table shows how you can operate expandable elements using the keyboard:

Function	Keyboard shortcuts
Expand a folder in a tree With <Arrow right>, for example, you expand a folder in the project tree.	<Arrow right>
Close a folder in a tree With <Arrow left>, for example, you collapse a folder in the project tree.	<Arrow left>
Open a drop-down list You can open drop-down lists with <F4> and then navigate with the arrow keys to scroll through the drop-down list. Finally, press the <Return> key to confirm your selection.	<F4>
Open autocompletion	<Ctrl+Space> <Ctrl+I>
Show object selection	<Ctrl+J>

6.1.3.4 Using project-related functions

Editing a project

Function	Keyboard shortcuts	Menu command
Open a project	<Ctrl+O>	Project > Open
Close a project	<Ctrl+W>	Project > Close
Save a project	<Ctrl+S>	Project > Save
Save a project under a different name	<Ctrl+Shift+S>	Project > Save as
Delete a project	<Ctrl+E>	Project > Delete project
Print project	<Ctrl+P>	Project > Print
Undo last action	<Ctrl+Z>	Edit > Undo
Redo last action	<Ctrl+Y>	Edit > Redo

Calling up the help function

Function	Keyboard shortcuts	Menu command
Calling up the help function	<F1> or <Shift+F1>	Help > Show help

6.1.3.5 Arranging windows

Below, you will learn how to open and close individual windows of the TIA Portal and save window layouts using the keyboard .

Opening and closing windows

The following table shows how you can open and close windows with keyboard shortcuts:

Function	Keyboard shortcuts	Menu command
Open/close project tree	<Ctrl+1>	View > Project tree
Opening/closing the detailed view	<Ctrl+4>	View > Details view
Opening/closing the overview	<Ctrl+2>	View > Overview
Opening/closing a task card	<Ctrl+3>	View > Task card
Open libraries	<Ctrl+Shift+L>	
Open hardware catalog If you are in the device or network view, the hardware catalog opens.	<Ctrl+Shift+C>	
Open/close inspector window	<Ctrl+5>	View > Inspector window
Open the "Properties" tab in the inspector window	<Ctrl+6>	
Open the "Info" tab in the Inspector window	<Ctrl+7>	
Open the "Diagnostics" tab in the Inspector window	<Ctrl+8>	
Display or hide reference projects	<Ctrl+9>	

Function	Keyboard shortcuts	Menu command
Display the on-screen keyboard You can display a keyboard on the screen, for example, to operate with a touch screen.	<Ctrl+Shift+K>	
Close all editors	<Ctrl+Shift+F4>	Window > Close all

Using saved window layouts

You can save individual window arrangements and restore them at a later point in time. The following table shows how you to access saved window layouts with keyboard shortcuts:

Function	Keyboard shortcuts	Menu command
Restore active window layout If you use a saved window layout and have made changes to the program interface in the meantime, you can restore the original state of the active window layout with <Alt+Shift+0>.	<Shift+Alt+0>	Window > Restore window layout
Load window layout You can use <Alt+Shift+[number of the window layout]> to activate the first of the five saved window layouts.	<Shift+Alt+[Number of the window layout]>	Window > Window layout 1 to 5

6.1.3.6 Navigating through the program interface

The TIA Portal is divided into several interface areas, for example, individual windows, toolbars and editors. If you want to work with the keyboard within an interface area, you first have to place the focus on it. Below, you will learn how to place the focus on individual interface areas using the keyboard. You will also learn how to move within an interface area in the TIA Portal using the keyboard.

Switching between interface areas and editors

The following table shows how to move between individual interface areas of the TIA Portal:

Function	Keyboard shortcuts
Move clockwise between the interface areas You can use the <F6> key to move clockwise between the various interface areas of the TIA Portal. The interface area currently in focus is highlighted with a blue title bar. If you are in the project tree, for example, and press the <F6> key, you jump to the currently open editor. If you press <F6> again, and the task cards are in focus. If you press <Shift+F6>, on the other hand, you move counterclockwise between the work areas.	<F6>
Move counter-clockwise between the interface areas With <Shift+F6> you move counter-clockwise between the interface areas of the TIA Portal.	<Shift+F6>
Go to the next open editor With <Ctrl+Alt+Arrow right> you move to the next open editor. You can see the open editors in the editor toolbar.	<Ctrl+Alt+Arrow right> Alternative: <Ctrl+F6>

Function	Keyboard shortcuts
Go to the previously open editor With <Ctrl+Alt+Arrow left> you move to the most recently open editor.	<Ctrl+Alt+Arrow left> Alternative: <Ctrl+Shift+F6>
Go to the next higher section of the interface area With <Shift+Esc>, you move to the next higher section of the program interface. If, for example, you have selected a device in the project tree and you press <Shift+Esc>, the entire project navigation is put into focus.	<Shift+Esc> Alternative: <Alt+Arrow up>
Go to the next lower section of the interface area With <Return>, you place the focus on the next lower section of the program interface. For example, if you have just opened the properties of a device in the Inspector window in order to assign parameters to the device, press <Return> to go one level deeper in the program interface. You can then navigate to the desired parameter using the Tab key.	<Return> Alternative: <Alt+Arrow down>

Navigation within interface areas and editors

The following table shows how you can navigate within an interface area using the keyboard:

Function	Keyboard shortcuts
Jump to the next element within an interface area You can use the Tab key to jump from one element to the next within a work area. If, for example, you have opened the properties of a device and want to jump from one field to the next, press the Tab key. Any changes you have made to the current text box are applied in this case.	<Tab>
Jump to the previous element within an interface area With <Shift+Tab>, you can jump to the previous element in a work area, for example, a previous text box. Any changes you have made to the current text box are applied in this case.	<Shift+Tab>
Move to the next tab within an interface area If an interface area is divided into separate tabs, you can move between the tabs with the shortcut keys <Ctrl+Tab>. If, for example, you are in the "Properties" tab of the Inspector window and you want to jump to the "Info" tab, press the shortcut keys <Ctrl+Tab>.	<Ctrl+Tab>
Go to the previous tab With <Ctrl+Shift+Tab>, you move to the most recently open tab within an interface area.	<Ctrl+Shift+Tab>
Jumping to the toolbar of an editor You can use the <Alt+F10> key to jump to the toolbar of an editor. For example, if you have opened the print preview and want to switch to the next page of the printout in the toolbar, press <Alt+F10>. Then, use the arrow keys to navigate to the appropriate icon in the toolbar and confirm the selection with <Return>.	<Alt+F10>
Use arrows on the divider to display or hide user interface components The table in the work area can be minimized and maximized. First, navigate to the work area and use the Tab key to place the focus on one of the little arrows on the separator line above the table. The arrows have the focus as soon as they are highlighted in blue. Then, press the space bar to minimize or maximize the table.	<Space>

6.1.3.7 Customizing editors

Below, you will learn how to arrange editors using the keyboard. You will also learn how to select the display size and the area within a graphical editor.

Arranging and customizing editors

The following table shows how to arrange open editors above and below each other or side-by-side, and how to close an open editor:

Function	Keyboard shortcuts	Menu command
Close active editor	<Ctrl+F4>	
Split editor space vertically If, for example, you have opened the overview window and the network view and want to display them side-by-side, press the <F12> key.	<F12>	Window > Split editor space vertically
Split editor space horizontally You can display two open editors in the work area above and below each other.	<Ctrl+F12>	Window > Split editor space horizontally
Remove window split If you have displayed two editors in the work area horizontally or vertically in split mode, you can remove the split with <Alt+Shift +F12>.	<Alt+Shift+F12>	Window > Unsplit editor space

Customizing the display in an editor

The following table shows how you how to zoom in and out in graphical editors and how to move the area selection in an editor:

Function	Keyboard shortcuts
Zoom in step-by-step in an editor With <Ctrl> and the <Plus> key on the numeric keypad of the keyboard, you can zoom in on the display of the editor.	<Ctrl+Plus> Alternative: <Ctrl+mousewheel up>
Zoom out step-by-step in an editor Use the <Ctrl> and the <Minus> key on the numeric keypad of the keyboard to zoom out of the display of the editor.	<Ctrl+Minus> Alternative: <Ctrl+mousewheel down>
Set view in editor to 100% You can zoom in or out of the display in a graphical editor to set the current view to 100% by pressing <Ctrl+0>.	<Ctrl+0>
Moving the area selection of the editor If you hold down the spacebar, you can move the displayed section of an editor using the mouse.	<Space>

6.1.3.8 Editing objects

Selecting objects

The following table shows how to select individual objects, for example, devices in the project tree:

Function	Keyboard shortcuts	Menu command
Select an object located at the left, right, above or below	<Arrow keys>	
Jump to the first object within the current interface area The first object in the interface area currently in focus is selected. In the project tree, this would be the project node at the top, for example.	<Home>	
Jump to the last object within the current interface area The last object in the interface area currently in focus is selected, for example, the last item in the project tree.	<End>	
Select all objects in an area All objects in the work area currently in focus are selected.	<Ctrl+A>	Edit > Select all
Select multiple objects If you want to select several objects that are not located directly next to each other, you first have to move the focus (gray outline of an object) to the next desired object using <Ctrl+Arrow keys>. The current selection is maintained. Then, press the space bar to select the new focused object as well. Repeat this process until all desired objects are selected.	<Ctrl+Arrow keys> + <Space>	

Editing objects

The following table provides an overview of all the keyboard shortcuts required for editing objects:

Function	Keyboard shortcuts	Menu command
Insert new object A new object is inserted depending on your current context. If, for example, you are in the device view, the "Add Device" dialog opens for creating a new device.	<Ctrl+N>	
Open object	<Return>	
Rename an object	<F2>	Edit > Rename
Copy an object	<Ctrl+C> Alternative: <Ctrl+Ins>	Edit > Copy
Cut an object	<Ctrl+X> Alternative: <Shift+Del>	Edit > Cut
Paste an object	<Ctrl+V> Alternative: <Shift+Ins>	Edit > Paste
Delete an object	<Del>	Edit > Delete

Function	Keyboard shortcuts	Menu command
Compile an object	<Ctrl+B>	Edit > Compile
Open properties of an object Many objects in the TIA Portal have editable properties. Press the shortcut keys <Alt+Enter> to display the properties of an object.	<Alt+Return>	-

6.1.3.9 Text editing

Below, you will learn how to operate text editing functions using only the keyboard.

Editing text

The following table shows the basic editing functions for text:

Function	Keyboard shortcuts
Switch to insert or overwrite mode	<Insert>
Exit edit mode	<Esc>
Delete	<Del>
Delete characters	<Backspace>
Confirm entry in a text box and leave the text box	<Return>
Line break in a multiline text box In a multiline text box, hold down the <Shift> button to create a line break.	<Shift+Return>
Reset input in a text box If you are in an text box and press <Esc>, you exit the box and the changes are discarded.	<Esc>

Navigating within a text area

The following table shows how to navigate in a text area with the keyboard:

Function	Keyboard shortcuts
Jump to start of line	<Home>
Jump to end of line	<End>
Jump to start of text	<Ctrl+Home>
Jump to end of text	<Ctrl+End>
Jump to the previous page	<PgUp>
Jump to the next page	<PgDn>
Confirm entry in a text box and leave the text box	<Return>
Line break in a multiline text box	<Shift+Return>
Reset input in a text box If you are in an text box and press <Esc>, you exit the box and the changes are discarded.	<Esc>

Selecting text

The following table shows how to select text with the keyboard:

Function	Keyboard shortcuts
Expand selection to the word at the left or right The text or current text selection is marked up to the end of word. If you are at the beginning or end of a word, the previous or next word is selected.	<Ctrl+Shift+Arrow left or Arrow right>
Expand selection to beginning of line	<Shift+Home>
Expand selection to end of line	<Shift+End>
Expand selection to beginning of text The text is selected up to the beginning or the end.	<Ctrl+Shift+Home>
Expand selection to end of text The text is selected up to the beginning or the end.	<Ctrl+Shift+End>

6.1.3.10 Editing tables

Below, you will learn how to navigate with the keyboard in tables, edit individual fields, and select parts of tables.

General keyboard operation in tables

The following table shows how you can edit tables using only the keyboard:

Function	Keyboard shortcuts
Place a cell in edit mode	<F2> or <Return>
Confirm entry and exit edit mode	<Return>
Cancel editing and discard changes	<Esc>
Open drop-down list in a cell Open the drop-down list with <F4>. Use the arrow keys to select the desired entry and then confirm the selection with <Return>.	<F4>
Close drop-down list in a cell and discard changes	<Esc>

Navigate in tables

The following table shows how you can navigate within a table using the keyboard:

Function	Keyboard shortcuts
Go to the next cell	<Arrow keys>
Go to the next editable cell on the right	<Tab>
Go to the next editable cell on the left	<Shift+Tab>
Move a screen upwards	<PgUp>
Move a screen downwards	<PgDn>
Go to the first cell in the row	<Home>

Function	Keyboard shortcuts
Go to the last cell in the row	<End>
Go to the first cell in the table	<Ctrl+Home>
Go to the last cell in the table	<Ctrl+End>
Go to the top cell in the column	<Ctrl+Arrow up>
Go to the bottom cell in the column	<Ctrl+Arrow down>

Selecting areas in tables

The following table shows how you can select areas within a table using the keyboard:

Function	Keyboard shortcuts
Select column	<Ctrl+Space>
Select line	<Shift+Space>
Select all cells	<Ctrl+A>
Expand selection by one cell	<Shift+arrow keys>
Extend selection up one page	<Shift+PgUp>
Extend selection down one page	<Shift+PgDn>
Expand selection up to the first row	<Ctrl+Shift+Arrow up>
Expand selection down to the last row	<Ctrl+Shift+Arrow down>
Expand selection to the first cell in the row	<Ctrl+Shift+Arrow left>
Expand selection to the last cell in the row	<Ctrl+Shift+Arrow right>

6.1.3.11 Using online functions

Controlling online functions with the keyboard

The following table provides an overview of the shortcut keys that you can use for the online functions of the TIA Portal:

Function	Keyboard shortcuts	Menu command
Establish an online connection	<Ctrl+K>	Online > Go online
Go offline	<Ctrl+M>	Online > Go offline
Download project data to the device	<Ctrl+L>	Online > Download to device
Show accessible devices This opens a dialog showing all devices that are connected to the PG/PC interface of the PG/PC.	<Ctrl+U>	Online > Show accessible devices
Start CPU The CPU is set to "RUN" mode. The CPU must be online for this.	<Ctrl+Shift+E>	Online > Start CPU

Function	Keyboard shortcuts	Menu command
Stop CPU The CPU is set to "STOP" mode. The CPU must be online for this.	<Ctrl+Shift+Q>	Online > Stop CPU
Start simulation The hardware and software of the project can be tested in a simulated online environment, without the modules actually being online.	<Ctrl+Shift+X>	Online > Simulation > Start

6.1.3.12 Using the on-screen keyboard

Introduction

When working with the TIA portal, you also have the Microsoft on-screen keyboard available.

Displaying the on-screen keyboard

To display the on-screen keyboard, follow these steps:

1. In the "View" menu, select the "Screen keyboard" command.

Exiting the on-screen keyboard

To exit the on-screen keyboard, follow these steps:

1. In the "File" menu of the on-screen keyboard, select the "Exit" command.

6.1.4 Special features specific to the operating system

6.1.4.1 Influence of user rights

Restrictions when user rights are limited

The software provides several functions that require direct access to the hardware of the programming device / PC and therefore also to the installed operating system. To make full use of the range of functions, the software must cooperate closely with the operating system. To ensure problem-free interaction, you should therefore be logged on to the operating system with adequate user rights.

In particular, you may not be able to use functions requiring an online connection or those that change the settings of interface cards if you work with limited user rights.

Recognizing restricted functions

You can recognize functions requiring special rights as follows:

- A shield icon is displayed beside the function.
  The function can be used but is regulated by the user account control.
- A box is grayed out and cannot be accessed.
  You require administrator privileges to access the box. In some operating system environments, you can obtain administrator privileges by entering an administrator password.

Note

A box being grayed out does not necessarily mean a lack of rights. You should also check the additional information in the tooltip cascades to find out the conditions for editing the box.

6.1.4.2 Expanding user rights

Counteracting restrictions due to user rights

Certain functions may not be available if you are not logged on to the operating system with adequate rights. You can counteract these restrictions in the following ways:

- Enabling of extended rights using Windows user account control
- Logging on to the operating system with administrator privileges
- Using temporary administrator rights

Enabling extended rights using the Windows user account control

To be able to use a function indicated by the shield icon of the Windows user account control, follow these steps:

1. Click on the box or button with the shield icon.
   The security prompt of the Windows user account control opens.
2. Follow the instructions of the Windows user account control and, when prompted enter an administrator password, if possible.

The function can now be used once without restrictions.

Logging on to the operating system with administrator privileges

To be able to use a function that is disabled due to lack of user rights, follow these steps:

1. Close the software.
2. Log off from the operating system.
3. Log on to the operating system with administrator privileges.
4. Restart the software.

Using temporary administrator rights

To obtain administrator privileges temporarily, follow these steps:

1. Click the "Change settings" button. You will find this button in dialogs that allow the temporary assignment of administrator privileges.
   An operating system dialog box for entering an administrator password opens.
2. Enter an administrator password.

The settings can be temporarily changed. When you call the dialog again, the procedure must be repeated.

Note

This function is not supported by all operating systems. If no "Change settings" button is present or the button is grayed out, you will need to log on to the operating system with administrator privileges instead.

6.2 Help on the information system

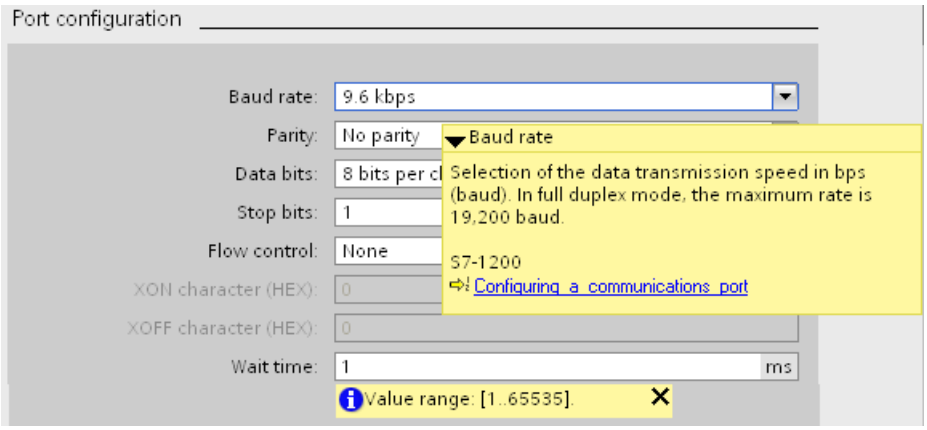
6.2.1 General remarks on the information system

Quick answers to your questions

A comprehensive help system is supplied with the TIA Portal for solving your tasks. It describes basic concepts, instructions and functions. While working with the program, you also receive the following support:

- Roll-out for correct inputs in dialog boxes
- Tooltips for information on elements of the user interface, for example text boxes, buttons and icons. Some of the tooltips are supplemented by cascades containing more precise information.
- Help on the current context, on menu commands for example when you click on the keys <F1> or <Shift+F1>.

The following figure shows an example of a cascading tooltip (top) and a roll-out (bottom):



Help

The supplied help system describes concepts, instructions and functions. It also contains reference information and examples. The help opens in a separate window.

A navigation pane appears on the left side of the help window. You can also hide the navigation pane to make room on the screen. The navigation pane provides you with the following functions:

- Table of contents
- Search in the index
- Full text search of the entire help
- Favorites

Identification of the topics in the help according to the type of information

The help topics are identified by different symbols depending on the type of information they contain.

Symbol	Information type	Explanation
	Operating instructions	Describes the steps to follow in order to carry out a particular task.
	Example	Contains a concrete example to explain the task.
	Factual information	Contains background information that you need to know to carry out a task.
	Reference	Contains comprehensive reference information to refer back to.

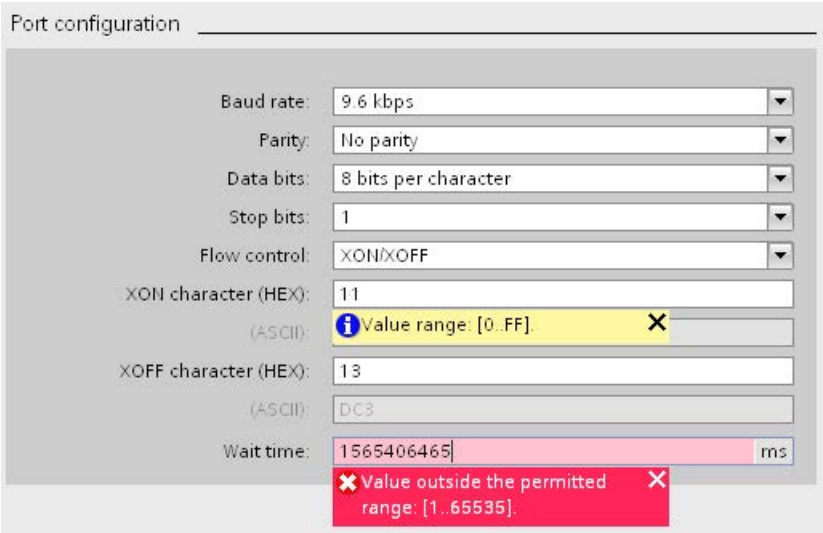
Identification of the topics in the help according to the target system

Depending on the products that are installed, the help system may contain sections that apply to specific devices only. To be able to recognize such sections at first glance, you will find a note in brackets in the table of contents. The search results in the full text search and in the index are marked in the same way if they only apply to a specific device.

Roll-out

Certain text boxes offer information that rolls out and helps you to enter valid parameters and values. The roll-out informs you about permitted value ranges and data types of the text boxes.

The following figure shows a roll-out (yellow) and a roll-out error message (red), which indicates an invalid value:



The roll-out is closed as soon as you exit the window or click on the "x" in the upper right-hand corner.

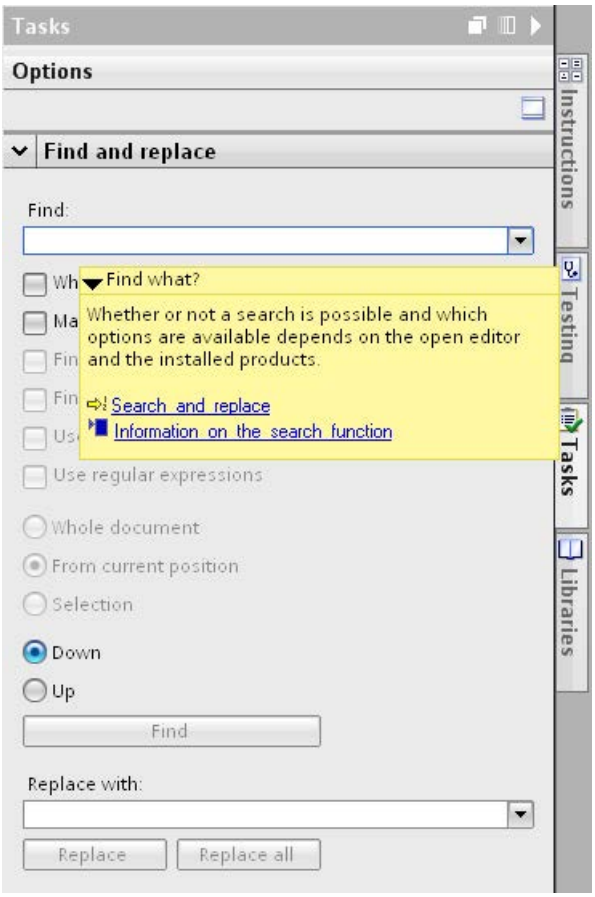
Tooltip

Interface elements offer you a tooltip for easier identification.

Tooltips, which have an arrow icon on the left, contain additional information in tooltip cascades. If you position the mouse pointer briefly over the tooltip or click the arrow icon, this information is displayed. The automatic display of tooltip cascades can be disabled.

If additional information is contained in the help system, a link to the corresponding help topic appears in the cascade. If you click on the link, the corresponding topic opens in help.

The following figure shows a tooltip with opened cascade:



See also

Configuring the display of tooltips and tooltip cascades (Page 327)

6.2.2 Opening the supplied Help system

You access the help via the menu, using links in the tooltip cascades, or with the <F1> key.

Procedure

To open the supplied help system, follow these steps:

1. In the "Help" menu, select the "Show help" command or press <F1> to display the help matching the current context.

Or:

1. Click on the link in a tooltip cascade to go directly to a point in the help system that contains more detailed information.

6.2.3 Searching the Help system for keywords

Searching for keywords in the help text

To search the help topics for predefined keywords, follow these steps:

1. Click the "Show/hide table of contents" button in the help toolbar to display the table of contents.
   The table of contents is displayed and the "Index", "Search" and "Favorites" tabs are visible.
2. Open the "Index" tab.
3. Enter the search term in the input box or select the search term from the list of key words.
4. Click "Display".

6.2.4 Full-text searches

Full-text searches

To search the entire text for specific words, follow these steps:

1. Click the "Show/hide table of contents" button in the help toolbar to display the table of contents.
   The table of contents is displayed and the "Index", "Search" and "Favorites" tabs are visible.
2. Open the "Search" tab.
3. Type in your search term in the text box.
4. Refine your search if necessary using additional criteria:
   - Select "Search previous results" to start an additional search operation of your last search results only.
   - Select "Search for similar words" to find words that differ only slightly from your search term.
   - Select "Search titles only" to obtain only results that contain your search term in the title. The contents of the Help topics are ignored during the search.

5. Click on the arrow button to the right of the search field to use logic operations. The following logic operations are available:
   - Combine two or more search terms using the "AND" operator to find only Help topics that contain all the search terms in the text.
   - Combine two or more search terms using the "OR" operator to find only Help topics that contain one or more of the search terms in the text.
   - Combine two or more search terms using the "NEAR" operator to find only Help topics that contain terms in close proximity to each other (eight words).
   - Precede a word with the "NOT" operator to exclude Help topics from the search that contain this word.
6. Click on "List topics" to start the search.
   The results are now listed with title, position and ranking. The "Position" column shows the section in which the Help topic found is located. Sorting according to ranking is based on the position of the Help topics found in the table of contents and based on the number of hits in the Help topics.

6.2.5 Using favorites

Using favorites

You can save individual help topics as favorites. This saves you searching for the help topic a second time.

Saving favorites:

To save a page as a favorite, follow these steps:

1. Open the help topic or the chapter you want to save as a favorite.
2. Click the "Show/hide table of contents" button in the help toolbar to display the table of contents.
   The table of contents is displayed and the "Index", "Search" and "Favorites" tabs are visible.
3. Open the "Favorites" tab.
4. Click the "Add" button.
   The help topic or chapter is saved as a favorite and is available the next time you open the help system.

Calling up favorites:

To call up a page from the favorites, follow these steps:

1. Click the "Show/hide table of contents" button in the help toolbar to display the table of contents.
   The table of contents is displayed and the "Index", "Search" and "Favorites" tabs are visible.
2. Open the "Favorites" tab.

3. Select the topic you want to open from the list.
4. Click the "Display" button.

Deleting favorites

To delete an entry from the favorites, proceed as follows:

1. Click the "Show/hide table of contents" button in the help toolbar to display the table of contents.
   The table of contents is displayed and the "Index", "Search" and "Favorites" tabs are visible.
2. Open the "Favorites" tab.
3. Select the topic you want to remove from the list.
4. Click the "Remove" button.

6.2.6 Printing help topics

Printing information

You can either print all the contents of the Help system or individual topics only.

Procedure

To select the topics you would like to print, follow these steps:

1. Click the "Display printing dialog" button.
   The table of contents opens in a separate window.
2. Select the check boxes for the folders and help topics to be printed in the "Print help topics" dialog.
3. Click the "Print" button to print the selected information.
   The "Print" dialog opens.
4. Select the printer on which you want print the help topics.
5. Click "Properties" if you want to make additional printer settings.
6. Confirm your entries with "OK".
   The help topics are printed out on the selected printer.

6.2.7 Configuring the display of tooltips and tooltip cascades

Configuration options for tooltips and tooltip cascade

You can customize the display of tooltips and tooltip cascades to suit your needs. You can make the following settings:

- Display or hide truncated text
  Sometimes texts may be too long for a text field. The texts are then fully displayed in a tooltip when you hover your mouse over the text field. You can enable or disable this function.
- Enable or disable tooltips
  Tooltips provide more detailed information about an element of the user interface. You can also have tooltips displayed in a cascade. If you disable the tooltips, the cascade with context-sensitive help is also no longer displayed. However, you have the option of manually displaying the tooltip for the currently active interface element by pressing <F1>.
- Enable or disable automatic opening of tooltip cascades
  By keeping the mouse pointer over a tooltip for a brief time, any available cascades are displayed automatically. You can enable or disable the automatic display of cascades. When automatic display is disabled, you must open the cascade manually if necessary. To do this, click on the arrow icon in the tooltip.

Procedure

To configure the display of tooltips and tooltip cascades, follow these steps:

1. Select the "Settings" command in the "Options" menu.
2. Select the "General" group in the area navigation.
3. Select or clear the individual check boxes in the "Tooltips" area to suit your needs. The "Open cascade automatically in tooltips" check box is only available if you have enabled the display of tooltips.

See also

General remarks on the information system (Page 320)

6.2.8 Safety Guidelines

Safety guidelines

This Help manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage

have no safety alert symbol. These notices shown below are graded according to the degree of danger.

DANGER

indicates that death or severe personal injury will result if proper precautions are not taken.

WARNING

indicates that death or severe personal injury may result if proper precautions are not taken.

CAUTION

with a safety alert symbol, indicates that minor personal injury can result if proper precautions are not taken.

NOTICE

without a safety alert symbol, indicates that property damage can result if proper precautions are not taken.

Note

indicates that an unintended result or situation can occur if the corresponding information is not taken into account.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The device/system may only be set up and used in conjunction with this documentation. Commissioning and operation of a device/system may only be performed by qualified personnel. Within the context of the safety notes in this documentation qualified persons are defined as persons who are authorized to commission, ground and label devices, systems and circuits in accordance with established safety practices and standards.

Prescribed Usage

Note the following:

WARNING

This device may only be used for the applications described in the catalog or the technical description and only in connection with devices or components from other manufacturers which have been approved or recommended by Siemens. Correct, reliable operation of the product requires proper transport, storage, positioning and assembly as well as careful operation and maintenance.

Trademarks

All names identified by ® are registered trademarks of the Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

6.2.9 Assembling customized documentation

Customized documentation

In the Siemens Industry Online Support, you can assemble customized documentation that is tailored to your needs. All configurable manuals and operating instructions of the Siemens Industry Online Support are available to you for this purpose. You can select the parts that are of interest to you and combine them into in a library to form personal documentation. You can organize the documentation using folders in the library. The folders will later become the individual chapters of your custom documentation.

You can open your personal library here (https://www.automation.siemens.com/mdm/?guiLanguage=en).

Requirements

- The manuals or operating instructions used must be configurable. You can recognize configurable manuals by the suffix "configurable" in their name.
- To use all the functions, you have to register in the Siemens Industry Online Support and log on.

Documentation in different languages

You can change the language of the assembled documentation to German, French, Spanish, Italian and Chinese. This gives you the possibility, for example, to gather relevant information for a specific project and make it available to colleagues who speak other languages.

Export function in the documentation

You can perform an export at any part of your library in various formats (PDF, XML, RTF).

Help on creating documentation

You can find more help on creating and using custom documentation in the Siemens Industry Online Support (https://www.automation.siemens.com/mdm/_help/en/mdm_reference_manual_de-DE.htm).

Editing projects 7

7.1 The basics of projects

Introduction

Projects are used to organize the storage of data and programs resulting from the creation of an automation solution. The data that makes up a project includes the following:

- Configuration data on the hardware structure and parameter assignment data for modules
- Project engineering data for communication over networks
- Project engineering data for the devices
- Logs for important events in the life cycle of the project

Project hierarchy

Data is stored in a project in the form of objects. Within the project, the objects are arranged in a tree structure (project hierarchy).

The project hierarchy is based on the devices and stations along with the configuration data and programs belonging to them.

Common data of the project and online access, for example, are also displayed in the project tree.

See also

Creating a new project (Page 332)

Opening projects (Page 334)

Saving projects (Page 338)

Deleting projects (Page 339)

Using logs (Page 331)

7.2 Using logs

For some operations within the TIA Portal, logs are created automatically in the background. These logs document changes in the project. Logs are created automatically, for example, when you migrate projects and programs or when you update instances from the library.

Logs are displayed in the "Common data" folder in the project tree. They are stored together with the project in the project folder and can therefore be read independently of the programming device/PC used as soon as you have opened the project.

In addition to displaying them in the TIA Portal, logs can also be printed.

Opening logs

To open a log, follow these steps:

1. Open the "Common data > Logs" folder in the project tree.
2. Double-click the desired log in the list.
   The contents of the log are displayed in the work area.

Deleting logs

To delete a log, follow these steps:

1. Select the log in the project tree.
2. Press <Del>.
   The selected log is deleted from the project directory and removed from the project tree.

7.3 Creating and managing projects

7.3.1 Creating a new project

Procedure

To create a new project, follow these steps:

1. Select the "New" command in the "Project" menu.
   The "Create a new project" dialog opens.
2. Enter your project name and path or accept the proposed settings.
3. Click the "Create" button.

Result

The new project is created and displayed in the project tree.

See also

The basics of projects (Page 331)

Opening projects (Page 334)

Saving projects (Page 338)

Deleting projects (Page 339)

7.3.2 Compatibility of projects

You can use the TIA Portal to open projects that were created with an older version of the TIA Portal or with a different installation package. In the following, you will learn what to consider in this case.

Opening projects from older product versions

The table below demonstrates the behavior of the TIA Portal when opening projects from older product versions:

Product version of the TIA Portal (file extension of the respective version)	Behavior when opening with TIA Portal V13
V10.5 (.ap10) V11.x (.ap11)	The project is automatically upgraded to the latest product version when opened, following your confirmation. The upgraded project is a copy of the original project. The original project is retained unchanged.
V12 (.ap12)	The project is upgraded when you open it. You can select if you want to upgrade the project to product version V12 SP1 or V13. The upgraded project is a copy of the original project in both cases. The original project is retained unchanged. When you upgrade the project to product version V12 SP1, the project is opened in compatibility mode.
V12 SP1 (.ap12)	When you open the project, you can decide if you want to open it in compatibility mode or upgrade it to the current product version. A copy of the original project is created when you upgrade the project. The original project is retained unchanged.

Compatibility mode

When you open a project in the V12 SP1 project format and do not upgrade it, the project is in compatibility mode. The range of functions of the TIA Portal is limited to the range of functions of V12 SP1. The project remains backward compatible and can still be opened and edited with TIA Portal V12 SP1.

To projects in compatibility mode, components can be added that were supplied at a later date for version V12 SP1 as part of a Hardware Support Package (HSP). The project is still backward compatible with TIA Portal V12 SP1. To continue working with the project in TIA Portal V12 SP1, you merely have to also install the Hardware Support Package.

Global libraries are always created in the most recent format and are not backwards compatible, even if the project is opened in compatibility mode.

You need to upgrade the project to use the full range of functions of the current version. By upgrading the project you convert it to the current project format. The full range of functions of the current version becomes available.

Backward compatibility with the current product version

Projects saved with the current version of the TIA Portal are not backward compatible with older versions due to their enhanced functionality. Projects saved with TIA Portal V13 can only be opened with TIA Portal V13 or later.

Opening projects created with add-on products

The project you want to open may include data that were created with an optional software. The following cases may occur if you did not install these software components:

- Software components which are not absolutely required are missing:
  A dialog appears listing the missing software components. After the project is opened, its properties are displayed. You now have the opportunity to install missing components. All the devices contained in the project are available even if you do not install the missing components. However, you can only work with the devices that are supported by the currently installed software.
  Devices which are not supported because software components are missing are identified by the following symbol in the project tree:

- Essential software packages are missing:
  A dialog appears listing the missing software components. The essential software components are marked. The project can only be opened if you install the missing software components.

See also

Opening projects (Page 334)

Upgrading projects (Page 335)

Compatibility of global libraries (Page 429)

7.3.3 Opening projects

You can open all projects from the current version and earlier versions in the TIA Portal.

Projects from earlier versions of the TIA Portal may first have to be upgraded to a more recent project format. You are prompted to upgrade the project when you open it.

You recognize projects of the TIA Portal by their file name extension ".ap[version number]". TIA Portal V13 projects have the file name extension ".ap13".

Procedure

To open an existing project, follow these steps:

1. Select the "Open" command in the "Project" menu.
   The "Open project" dialog box opens and the list of most recently used projects is displayed.
2. Select a project from the list and click "Open".
3. If the project you require is not included in the list, click the "Browse" button. Navigate to the desired project folder, and open the project file.
   Projects in the current project format are opened in the project view. The "Upgrade project" dialog box opens if you have selected a project from an older version of the TIA Portal. More information on upgrading the project can be found in chapter "Upgrading projects (Page 335)".

See also

Compatibility of projects (Page 333)
Upgrading projects (Page 335)
Compiling project data (Page 354)
Opening a global library (Page 430)
The basics of projects (Page 331)
Creating a new project (Page 332)
Saving projects (Page 338)
Deleting projects (Page 339)

7.3.4 Upgrading projects

Projects from older versions of the TIA Portal can be edited with the current version of the TIA Portal. Depending on the product version that was used to create the project, you either upgrade the project or open the project in compatibility mode. To find out which options you have depending on the product version of the project, go to chapter "Compatibility of projects (Page 333)".

When you open projects from older product versions, you are prompted to upgrade the project. You can manually upgrade projects that are already open in compatibility mode. After upgrading the project to the current product version, it can no longer be opened in older versions of the TIA Portal. But the original project is still available. The upgraded project is saved as a copy.

Upgrading projects from V11.x or lower

Proceed as follows to upgrade a project from TIA Portal V11.x or lower:

1. Open the project.
   The "Upgrade project" dialog box opens.
2. Click "OK".
   A copy of the project is created and upgraded. The project copy receives the name extension "_V13". The project is opened.
3. Compile the hardware and software of all devices in the project.

Upgrading projects from V12

Proceed as follows to upgrade a project from TIA Portal V12:

1. Open the project.
   The "Upgrade project" dialog box opens.
2. Select the target version for the upgrade:
   - Click "Yes" to upgrade the project to product version V13.
   - Click "No" to upgrade the project to product version V12 SP1 and to work in compatibility mode.

   A copy of the project is created and upgraded. The project copy receives the name extension "_V13" or "_V12SP1". The project is opened.
3. Compile the hardware and software of all devices in the project.

Upgrading V12 SP1 projects or using them in compatibility mode

Proceed as follows to upgrade a project from TIA Portal V12 SP1 or to use it in compatibility mode:

1. Open the project.
   The "Upgrade project" dialog box opens.
2. Select how you want to proceed with the project:
   - Click "Yes" to upgrade the project to product version V13.
   - Click "No" to use the project in compatibility mode.
3. Optional: If you have upgraded the project to the current product version, compile the hardware and software of all devices in the project.

Manually upgrading projects in compatibility mode to V13

To do this, the project must already be open in compatibility mode for the TIA Portal V12 SP1.

To manually upgrade a project in compatibility mode, follow these steps:

1. Select the "Upgrade" command in the "Project" menu.
   A security prompt appears.
2. Click "Yes" to confirm.
   A copy of the project is created and upgraded. The project copy receives the name extension "_V13". The project is opened.
3. Compile the hardware and software of all devices in the project.

Notes

Pay attention to the following notes after upgrading a project:

Note

Upgrading global libraries

Because global libraries are independent of projects, they are not upgraded automatically together with the project. If you want to continue using libraries from older versions of the TIA Portal, you need to upgrade these global libraries as well. More information on upgrading global libraries can be found in the chapter "Compatibility of global libraries (Page 429)".

Note

Upgrading know-how-protected blocks

The block is only upgraded and loaded after it has been opened once with the password. This means you should open know-how-protected blocks once after upgrading the project to also upgrade the blocks. If you have protected numerous know-how-protected blocks with the same password, you can select and open all of them at once.

See also

Compatibility of projects (Page 333)

Opening projects (Page 334)

Upgrading global libraries (Page 431)

Compiling project data (Page 354)

Compatibility of global libraries (Page 429)

7.3.5 Displaying properties of the project

You can display the properties of a project. Properties include the following:

- Metadata for the project
  This includes the following information: creation time, author, file path, project size, copyright, project languages, etc. Many of the properties can be changed.
- Project history
  The project history contains an overview with important events in the project life cycle. Here, for example, you can see the version of the TIA Portal used to create a project and whether it has been converted into another version in the meantime. If a project was created during a migration, for example, this is also indicated in the project history table with the date and time of the migration. If a log was created for an event, you can also call the log directly.

- Support packages in the project
  An overview of the add-on software needed to work with all devices in the project is displayed. In addition, installed GSD files are listed (device description files for other devices in the hardware catalog).
- Software products in the project
  You can display an overview of all installed software products needed for the project.

Procedure

To display the project properties, follow these steps:

1. Select the open project in the project tree.
2. Select "Properties" in the shortcut menu of the project.
   The dialog with the properties of the project opens.
3. Select the project properties in the area navigation that you want to have displayed.

7.3.6 Saving projects

You can save the project at any time either under the same or a different name. You can even save a project when it still contains elements with errors.

Saving a project

To save a project, follow these steps:

1. Select the "Save" command in the "Project" menu.
   All changes to the project are saved under the current project name. If you are editing a project from an earlier version of the TIA Portal, the file extension of the project is also retained and you can continue to edit the project in the earlier version of the TIA Portal.

Project Save as

To save a project under another name, follow these steps:

1. Select the "Save as" command in the "Project" menu.
   The "Save current project as" dialog opens.
2. Select the project folder in the "Save in" box.
3. Enter the new project name in the "File name" box.
4. Confirm your entry with "Save".
   The project is saved under the new name and opened.

See also

The basics of projects (Page 331)

Creating a new project (Page 332)

Opening projects (Page 334)

Deleting projects (Page 339)
Upgrading projects (Page 335)
Compatibility of projects (Page 333)

7.3.7 Closing projects

Procedure

To close a project, follow these steps:

1. Select the "Close" command in the "Project" menu.
   If you have made changes to the project since the last time you saved it, a message is displayed.
2. Decide whether or not you want to save the changes.

7.3.8 Deleting projects

Note

When you delete a project, the entire project data is removed from the storage medium.

Requirement

The project you want to delete is not open.

Procedure

Follow the steps below to delete an existing project:

1. Select the "Delete project" command in the "Project" menu.
   The "Delete project" dialog opens and includes the list of most recently used projects.
2. Select a project from the list.
   If the project you require is not included in the list, click the "Browse" button. Navigate to the desired project folder, and open the project file.
3. Click the "Delete" button.
4. Click "Yes" to confirm. This starts the deletion of the project.

Result

The entire project folder is deleted from the file system.

See also

The basics of projects (Page 331)

Creating a new project (Page 332)

Opening projects (Page 334)

Saving projects (Page 338)

7.3.9 Working with multi-language projects

7.3.9.1 Project text basics

Texts in different languages in the project

When you enter texts while working on a project, you would normally do this in your own language. If you then pass on the project to someone else who does not know this language, this person will require a translation of the relevant texts to a language they know. This is why all texts can be translated. In this way, you can ensure that anyone who is subsequently confronted with the texts sees the texts in his/her language of choice.

Project language

Project languages are all languages in which a project will later be used. Based on the editing language, all the texts can be translated to the various project languages. You specify the languages that will be available in the project tree under "Languages & Resources > Project languages".

Editing language

Every project has an editing language. When you enter texts, these are always created in the editing language. You should therefore make sure that the editing language set is the language in which you enter the texts. This avoids problems if you translate the texts later.

The editing language does not depend on the language of the user interface. You could, for example, set English as the user interface language, but use Italian as the editing language. If you enter texts, these will be created in this case in the project language "Italian", although the user interface of the TIA Portal displays English.

You set the editing language in the project tree under "Languages & Resources > Project languages > Editing language".

Reference language

The reference language is used as a template for translation. The text is displayed in the reference language for each text box in the "Tasks > Languages and resources" task card. You therefore know which text that belongs in a text box, even when no text is entered in the currently selected editing language.

User texts and system texts

For clarification purposes, a distinction is made between user texts and system texts:

- User texts are texts that the user created.
- System texts are texts that are created automatically according to the configuration in the project.

You manage the project texts in the project tree under "Languages & Resources > Project texts".

Examples of multilingual project texts

You can, for example, manage the following project texts in more than one language:

- Block titles and block comments
- Network titles and network comments
- Comments in tables
- Alarm texts
- Operator-relevant texts
- Text lists
- Labels of buttons
- Display names of recipes

Translating texts

The following procedures are available to translate texts.

- Translate all texts used in the project in tabular form
  You can enter the translations for the individual project languages directly in the "Project texts" table. You will find this in the project tree under "Languages & Resources > Project texts".
- Specify text assigned to individual objects in the Inspector window
  In the Inspector window, you can translate the texts that are assigned to the currently selected objects. Columns are displayed in a table for all available project languages. You can enter the translations for each text there.
- Translating texts using reference texts
  You can change the editing language for shorter texts. All the text cells are filled again with the default values and can be filled in the current language. As orientation, you can display what you last entered in the box in the reference language. To do this, select the "Tasks" task card and open the "Languages & resources".
- Exporting texts and translating them externally
  With larger volumes of text, you can export the texts to an Office Open XML file and translate them in a conventional table calculation program. You then import the translated texts again into the TIA Portal.

Note

Using Asian project languages

East Asian project languages are only displayed correctly in Windows XP, if the option "Install files for East Asian languages" is selected on the Languages tab of Regional and Language Options in the Control Panel of Windows XP Professional.

See also

Overview of the program settings (Page 273)

Changing the settings (Page 276)

Application examples for multilanguage projects (Page 347)

7.3.9.2 Select project languages

All the texts can be displayed within a project in the same language that you selected for your software user interface. This means that all project texts must exist in the corresponding language. You can select the available project languages yourself.

Requirement

- You are in the project view.
- A project is open.

Procedure

To select the project languages, follow these steps:

1. Click on the arrow symbol to the left of "Languages & Resources" in the project tree.
   The elements below this are displayed.
2. Double-click on "Project languages".
   In the work area, you will see a list of languages that you can select.
3. Select the required languages.

Result

All texts can be displayed in the activated languages if there is already a translation for these languages.

7.3.9.3 Setting the editing language

All the texts in the project are created in the editing language when they are entered. If you change the editing language, all future text input will be stored in the new editing language.

Requirement

- You are in the project view.
- A project is open.

Procedure

To change the editing language, follow these steps:

1. Click on the arrow symbol to the left of "Languages & Resources" in the project tree.
   The lower-level elements are displayed.
2. Double-click on "Project languages".
   The possible settings for the project languages are displayed in the work area.
3. Select the editing language in "General > Editing language".

7.3.9.4 Translating all project texts in tabular form

You can display and edit all project text used in the currently open project in a list. User and system texts are separated into two different lists for clarity. Both lists contain a separate column for each project language in which you can enter the translations for text.

Requirement

- You are in the project view.
- You have selected at least one further project language.

Procedure

To translate text in the project-wide list, follow these steps:

1. Click on the arrow symbol to the left of "Languages & Resources" in the project tree.
   The elements below this are displayed.
2. Double-click "Project texts".
   A list with the user texts in the project is displayed in the work area.
3. Click on "System texts" if you you want to edit the list of system texts rather than the user texts.
4. You can improve the clarity of the lists if you have a lot of texts.
   - To group identical texts and to translate them all at once, click the "Switch on/off grouping" button in the toolbar.
   - To hide texts that do not have a translation, click the "Filter for empty texts on/off" button in the toolbar.
   - To further limit the displayed project texts to certain devices, select the devices for which you want to display project texts in the drop-down list.
5. Enter the translation of the project texts in the relevant column.

7.3.9.5 Translating text associated with individual objects

If you want to edit the text of individual objects, it would be too difficult to locate the matching text in the table with all project texts. For this reason, there is a table in the Inspector window in which only the texts assigned to the currently selected objects are displayed. In the table, you can add missing translations for individual project languages or change existing texts.

Requirement

Text must be entered in at least one project language for the texts to be translated.

Procedure

To edit the text of the currently selected object, follow these steps:

1. Select the object whose text you want to edit.
2. Open the "Properties" tab in the Inspector window.
3. Open the lower-level "Texts" tab in the inspector window.
   A table with all the texts that belong to the selected objects is displayed. It contains one column for the currently selected editing language and the reference language, as well as additional columns for the other project languages.
4. Add or change the entries in the table for each project language.

See also

Application examples for multilanguage projects (Page 347)

7.3.9.6 Translating texts using reference texts

Introduction

After changing the editing language, all texts are shown in input boxes in the new editing language. If there is not yet a translation available for the newly set language, the input boxes are empty or filled with default values.

If you enter text in an input box, this is saved in the current editing language. Following this, the texts exist in two project languages for this input field, in the previous editing language and in the current editing language. This makes it possible to create texts in several project languages.

You can display existing translations for an input box in other project languages. These serve as a comparison for text input in the current editing language and they are known as the reference language.

Note

The display of reference texts depends on the installed products and is not supported by every editor.

Requirement

There is at least one translation into a different project language for an input field.

Procedure

To display the translation of an input cell in a reference language, follow these steps:

1. In the "Tasks" task card, select the "Languages & Resources" pane.
2. Select a reference language from the "Reference language" drop-down list.

Result

The reference language is preset. If you click in a text box, translations that already exist in other project languages are shown in the "Tasks > Languages & Resources" task card.

7.3.9.7 Exporting and importing project texts

You can export project texts for translation and then reimport them. The texts are exported to an Office Open XML file with the extension ".xlsx". This can be edited in Microsoft Excel or a number of other spreadsheet programs.

The following export options are available:

- Exporting individual project texts
- Exporting all user texts or system texts at once
  In this case, the export can be additionally limited by categories.

Note

Row limit in Microsoft Excel

Note that spreadsheet programs may be able to process only a certain number of rows. Microsoft Excel 2003 supports a maximum of 65536 rows, for example. Later versions of Microsoft Excel support significantly more rows.

Exporting individual project texts

To export individual project texts, follow these steps:

1. Open the "Languages & Resources" folder in the project tree.
   The lower-level elements are displayed.
2. Double-click "Project texts".
   The project texts editor opens.
3. Choose the "User texts" or "System texts" tab in the editor, depending on which texts you want to export.
4. Select the project texts you want to export.
5. Click "Export project texts" on the editor toolbar.
   The "Export" dialog box opens.

6. Choose the language you want to translate from in the "Source language" drop-down list.
7. Choose the language you want to translate to in the "Target language" drop-down list. The drop-down list contains the project languages you specified previously. If the required language is missing, you must first specify it in the project languages editor.
8. Specify a file path and a file name for the export file in the "Select file for export" input box.
9. Click "Export".

Exporting all system or user texts

To export all project texts, follow these steps:

1. Select the "Export project texts" command in the "Tools" menu.
   The "Export" dialog box opens.
2. Choose the language you want to translate from in the "Source language" drop-down list.
3. Choose the language you want to translate to in the "Target language" drop-down list. The drop-down list contains the project languages you specified previously. If the required language is missing, you must first specify it in the project languages editor.
4. In "Select content", select the check box "User texts" to export user texts. To export system texts, select "System texts". To export both user texts and system texts, select both check boxes.
5. In "Select content", select the required text categories for the user texts or the system texts.
6. In the "Export file" input field, specify a file name for the export file.
7. In the "Path" input field, select a path in the data system to which the export file is to be saved.
8. Click "Export".

Importing project texts

To import a file containing project texts, follow these steps:

1. Select the "Import project texts" command in the "Tools" menu.
   The "Import" dialog box opens.
2. Select the path and the file name of the import file from the "Select file for import" field.
3. Select the "Import base language" check box if you have made changes to the base language in the export file and you want to overwrite the entries in the project with the changes.
4. Click "Import".

See also

Application examples for multilanguage projects (Page 347)

7.3.9.8 Application examples for multilanguage projects

Introduction

Let us assume you are working in a team with colleagues some of whom speak English, some French and some German. You have created a project with the TIA Portal and have already created a functioning configuration.

To allow your other colleagues to be able to keep track of the project, you would like all devices being used to have comments in English and German. First, you would like to enter the comments in German. Following this, to save time and costs, you want to have the texts translated into English in a spreadsheet program by an external translation office.

In addition to this, you also want a single comment for a particular device in French so that your French-speaking colleague can continue working on this device.

The section below describes an example of how you can achieve this with the tools of the TIA Portal.

Translating the project into English

To enter the comments in German and to have them translated into English later, follow these steps:

1. Set the editing language to "German" and fill all the comment boxes with the relevant texts in German.
   On the device selected from the French-speaking colleague, enter, for example "Unser neues Gerät" in German first.
   All the comments are now stored in German.
2. Export all user texts to an Office Open XML file with the extension ".xlsx".
3. Have the user texts contained in the file translated into English in a spreadsheet program such as Microsoft Excel.
4. Import the file into the TIA Portal after it has been translated.
   All texts are now available in German and English.

Translating a single comment field to French

To translate an individual comment field to French, follow these steps:

1. Open the comment box for the device on which the French-speaking colleague will be working.
2. Open the "Languages & Resources" pane in the "Tasks" task card.
3. Set "French" as the editing language in the "Languages & Resources" pane. As the reference language, set, for example, "English".
   Since no translation has yet been installed in French, the comment box is empty. In the "Languages & Resources" pane, the English translation "Our new device" is displayed as a reference.
4. Orientating yourself on the English reference text enter "Notre nouvel appareil" in the comment box.
   The comment for this device is now available in the languages German, English and French.

See also

Project text basics (Page 340)

Exporting and importing project texts (Page 345)

Translating text associated with individual objects (Page 344)

7.3.10 Archiving and retrieving projects

7.3.10.1 Working with project archives

Archiving and transferring projects

If you work for a long time with a project, large files may result, especially with extensive hardware configurations. Therefore, you may want to reduce the size of the project, for example, when you archive it to an external hard drive, or when you send it via e-mail and require a smaller file size.

Options for reducing the size of the project

There are two ways to reduce the size of the project:

- Creating a project archive
  TIA Portal project archives are compressed files, each containing an entire project, including the entire folder structure of the project. Before the project directory is compressed into the archive file, all files are reduced to their essential components to further decrease the size of the project. Project archives are therefore well suited for sending via e-mail. Project archives have the file extension ".zap[version number of the TIA Portal]". Projects created with TIA Portal V13 have the file extension ". zap13".
  To open a project archive, retrieve the project archive. By retrieving it, the archive file with the included project files is extracted to a previously selected location in the original project directory structure.
- Minimizing a project
  You can skip additional compression in an archive file, and instead create a copy of the project directory. The included files are reduced to the essential elements of the project. This minimizes the required storage space. The full functionality of the project is maintained and you can open the project as usual.
  A minimized project is especially well suited for archiving, for example, on an external medium.

See also

Retrieving projects (Page 350)

Archiving projects (Page 349)

7.3.10.2 Archiving projects

You can reduce the storage space required for the currently open project by compressing the project into a file, or by reducing the project files to their essential components. You can do both of these with the archiving function of the TIA Portal.

Note

The most recently saved state of the project is used for archiving. Therefore, save the project before using the archiving function. This will ensure that your most recent changes are included in the archived project.

Procedure

To archive a project, follow these steps:

1. Select the "Archive" command in the "Project" menu.
   The "Archive current project as..." dialog opens.
2. Select the directory where you want to save the archive file or the new project directory.
3. Select the file type from the "File type" drop-down list:
   – TIA Portal project archive, if you want to create a compressed file of the project.
   – Minimized TIA Portal project, if you only want to create a copy of the project directory with minimal storage space.
4. Enter a file name in the "File name" field if you are creating an archive file. If you are creating a minimized project directory instead, enter the name of the new project directory to be created in the "File name" box.
5. Click "Save".

Result

When you have created a project archive, a compressed file with the extension ".zap13" is generated. Project archives of projects in compatibility mode for product version V12 SP1 receive the file extension ".zap12" and are backward compatible with TIA Portal V12 SP1. The archive file contains the complete project directory. The individual files of the project are also reduced to the essential components in order to save space.

If you have created a minimized TIA Portal project, only a copy of the original project directory is created at the desired location. The files contained within it are reduced to their essential components in order to save space.

See also

Working with project archives (Page 348)

Retrieving projects (Page 350)

7.3.10.3 Retrieving projects

You extract project archives of the TIA Portal with the "Restore from archive" function. This restores the project directory structure including all project files.

Requirement

No project should be open.

Procedure

To unpack a project archive, follow these steps:

1. Select the "Restore from archive" command in the "Project" menu.
   The "Retrieve archived project" dialog opens.
2. Select the project archive.
3. Click "Open".
4. The "Find folder" dialog opens.
5. Select the target directory to which the archived project should be extracted.
6. Click "OK".

Result

The project is extracted to the selected directory and opened immediately. If you extract a project archive that includes a project from a product version lower than V12 SP1, you may have to upgrade the project. You will automatically receive the corresponding prompt as soon as you open the project. The same rules apply that are described in the chapter "Compatibility of projects (Page 333)".

See also

Working with project archives (Page 348)

Opening projects (Page 334)

Compatibility of projects (Page 333)

Upgrading projects (Page 335)

7.4 Using reference projects

7.4.1 Basics of reference projects

Introduction

You can open other projects as a reference in addition to the current project. You can use these reference projects as follows:

- You can drag individual objects from a reference project into the current project and then edit them.
- You can open specific objects, for example, code blocks from a reference project as read-only. But this is not possible for all elements.
- You can use an offline/offline comparison to compare devices of the reference project to devices from the current project.

Note that reference projects are read-only. You cannot change the objects of a reference project.

Projects that were created with an older version of the TIA Portal or with a different installation package can also be opened as reference projects. The same compatibility rules as those for the normal opening of a project from an older version of the TIA Portal apply here.

See also: Compatibility of projects (Page 333)

See also

Comparing reference projects (Page 352)

Opening and closing a reference project (Page 351)

Reference projects (Page 297)

7.4.2 Opening and closing a reference project

Opening a reference project

To open a reference project, follow these steps:

1. In the "Reference projects" palette of the project tree, click on "Open reference project" in the toolbar.
   The "Open reference project" dialog box opens.
2. Navigate to the desired project folder, and open the project file. TIA Portal V13.x projects have the extension ".ap13". Older projects of the TIA Portal have the extension ".ap[version number]".
3. Click "Open".
   The selected project is opened as a read-only reference project.

Closing a reference project

To close a reference project, follow these steps:

1. In the "Reference projects" palette of the project tree, select the reference project you want to close.
2. Click on "Close reference project" in the toolbar.
   The selected reference project is closed.

See also

Basics of reference projects (Page 351)

Comparing reference projects (Page 352)

Reference projects (Page 297)

7.4.3 Comparing reference projects

Introduction

You can compare devices from reference projects with devices from both the current project as well as from the same or another reference project or from a library.

Note

Please note the following:

- You cannot specify actions for the comparison objects, since the reference projects are write-protected.
- You can perform a detailed comparison for the comparison objects, if the type of comparison object generally allows a detailed comparison.
- When comparing reference projects, you can always switch between automatic and manual comparison.

Procedure

To compare the objects of a reference project to the device data of the current project, follow these steps:

1. In the project tree, select the device whose data you want to compare to the data of a reference project and which allows offline/offline comparison.
2. Select "Compare > Offline/Offline" from the shortcut menu.
   The compare editor opens with the selected device displayed in the left area.
3. Open the "Reference projects" palette in the project tree.

4. Select the device of a reference project that you want to compare to the device data from the current project.
5. Drag the device from the reference project into the right drop area of the compare editor. You can identify the status of the objects based on the symbols in the status and action area. When you select an object, the object's properties and the corresponding object of the associated device is clearly shown in the properties comparison.
   You can drag a library or other devices from a reference project from the current project into drop areas at any time and thus start a new comparison. It does not matter which device you drag into the drop area.

See also

Basics of reference projects (Page 351)

Reference projects (Page 297)

Opening and closing a reference project (Page 351)

Carrying out offline/offline comparisons (Page 364)

Using the compare editor (Page 365)

7.5 Editing project data

7.5.1 Compiling and loading project data

7.5.1.1 Compiling project data

General information on compiling project data

Compiling project data

During compilation, project data is converted so that it can be read by the device. Hardware configuration data and program data can be compiled separately or together. You can compile the project data for one or more target systems at the same time.

The following project data must be compiled prior to loading:

- Hardware project data, for example, configuration data of the devices or networks and connections
- Software project data, for example, program blocks or process screens

Note

While a device is being compiled, no additional compiling process can be started. Note in this regard that you can not only perform a compiling process manually, but you can also trigger it automatically for HMI devices.

Scope of the compilation

When you compile project data, you have the following options depending on the device involved:

- Hardware and software (only changes)
- Hardware (only changes)
- Hardware (rebuild completely)
- Software (only changes)
- Software (rebuild all blocks)
- Software (reset memory reserve)

See also

Compiling project data (Page 354)

Compiling project data

The following section describes the general procedure for compiling project data in the project tree. You will find details of how certain objects are compiled and any special points to note in the online help of the product.

Procedure

To compile project data, follow these steps:

1. In the project tree, select the devices whose project data you want to compile.
2. Select the option you require in "Compile" submenu of the shortcut menu.

   Note

   Note that the options available to you depend on the selected device.

   The project data is compiled. You can check whether or not the compilation was successful in the Inspector window with "Info > Compile".

See also

General information on compiling project data (Page 353)

7.5.1.2 Loading project data

General information on loading

Introduction

In order to set up your automation system, you need to load the project data you generated offline on the connected devices. This project data is generated, for example when configuring hardware, networks, and connections or when programming the user program or when creating recipes.

The first time you download, the entire project data is downloaded. During later loading operations, only changes are downloaded.

You can download the project data to devices and memory cards.

Note

While a device is being compiled, no additional download process can be started. Note in this regard that you can not only perform a compiling process manually, but you can also trigger it automatically for HMI devices.

Possible options for downloading

Depending on the object you want to download, you have the following options:

- Hardware and software (only changes)
  Both the hardware configuration and software are downloaded to the destination if differences exist between the online and offline versions.
- Hardware configuration
  Only the hardware configuration is downloaded to the destination.
- Software (only changes)
  Only the objects that differ online and offline are downloaded to the destination.
- Load PLC program to device and reset
  All the blocks are loaded to the destination and all values are reset to their initial state. Be aware that this also applies to retentive values.

You can also upload project data already contained in a device back to your project. You have the following options:

- Upload entire device as new station
  The hardware configuration of the device and the software on the device is uploaded to the project.
  All relevant data of the device is uploaded to the project.
- Upload software of a device
  Only the blocks and parameters from the device are uploaded to an existing CPU in the project.

In both cases, during loading all instances of library types are connected again with the corresponding version of the type in the project library. If no suitable type is yet available for

a loaded instance or the correct version of the type does not exist in the project library, the type or the version is added to the project library.

Downloading with synchronization

In team engineering, it is possible for several users to work on one project with several engineering systems at the same time and to access one S7-1500 CPU. To ensure consistency within the shared project, it is necessary to synchronize the changed data prior to loading so that nothing gets overwritten unintentionally.

If differences are detected between the online and offline data management within the shared project that were caused by a different engineering system, automatic synchronization of the data to be loaded is offered when loading.

In this case, the "Synchronization" dialog displays the data to be synchronized with the current status (online-offline comparison) and the possible actions.

Use case	Recommendation	Synchronization
One or more blocks on the CPU (online) are more recent than in the engineering system (offline).	These blocks should be uploaded from the CPU to the engineering system before downloading.	Automatic synchronization is possible: The blocks in the engineering system are updated prior to loading.
One or more new blocks have been created and exist only on the CPU (online).	These blocks should be uploaded from the CPU to the engineering system before downloading.	Automatic synchronization is possible: The new blocks are added to the engineering system prior to the download.
One or more blocks on the CPU have been deleted.	The blocks should also be deleted prior to the download in the engineering system.	Automatic synchronization is not possible. The blocks deleted on the CPU should be manually deleted in the offline project in the engineering system.

Use case	Recommendation	Synchronization
One or more blocks on the CPU and in the engineering system are different. This is the case when a different user has changed blocks to which you have also made corrections and has already downloaded them to the CPU.	These blocks with competing changes must be adapted manually. You decide in this case which changes you are going to accept. If the blocks on the CPU are to be retained, you should adopt these blocks from the CPU in your engineering system before downloading to the CPU. If the blocks that you have changed are to be applied, you can continue with the download without synchronization.	Automatic synchronization is not possible: The affected blocks on the CPU or in the engineering system must be adapted manually. One of the existing block versions (online or offline) will be overwritten in the process.
There are differences in the hardware configuration on the CPU (online) and in the engineering system (offline).	Differences in the hardware configuration must be adapted manually. You decide in this case which hardware configuration you are going to accept. If the existing hardware configuration on the CPU is to be retained, you should adopt this in your engineering system prior to downloading. If you want to apply the hardware configuration you changed, you can continue downloading without synchronization.	Automatic synchronization is not possible: The hardware configuration must be adapted manually. One of the existing hardware configurations (online or offline) will be overwritten.

If required, you can use the "Force download to device" command to download blocks without synchronization.

See also

Downloading project data to a device (Page 357)

Downloading project data to a memory card (Page 359)

Uploading project data from a device (Page 360)

Downloading project data to a device

The following section describes the general procedure for downloading project data to a device. You will find details of how certain objects are downloaded and any special points to note in the online help of the product.

Requirement

- The project data is consistent.
- Each device to which you want to download is accessible via an online access.

Procedure

To download the project data to the selected devices, follow these steps:

1. Select one or more devices systems in the project tree.
2. Right-click on a selected element.
   The shortcut menu opens.
3. Select the option you require in the shortcut menu of the "Download to device" submenu.

   Note

   Note that the options available to you depend on the selected device.

   When necessary, the project data is compiled.

   – If you had previously established an online connection, the "Load preview" dialog opens. This dialog displays messages and proposes actions necessary for downloading.
   – If you had not previously established an online connection, the "Extended download to device" dialog opens, and you must first select the interfaces via which you want to establish the online connection to the device. You have the option of showing all compatible devices by selecting the corresponding option and clicking the "Start search" command.
     See also: Auto-Hotspot
4. Check the messages in the "Load preview" dialog, and select the actions in the "Action" column, if necessary.

   WARNING

   Preventing personal injury and material damage

   Performing the proposed actions while the plant is in operation can cause serious bodily injury and property damage in the event of malfunctions or program errors.

   Make sure that no dangerous situations can arise before you start the actions!

   As soon as loading becomes possible, the "Load" button is enabled.
5. Click the "Load" button.
   The loading operation is performed. If there is a need for synchronization, the system automatically displays the "Synchronization" dialog. This dialog displays messages and suggests actions that are required for the synchronization. You have the option of performing these actions or forcing the download without synchronization by clicking "Force download to device". If you have performed the suggested actions, you will be asked whether you want to continue with the download. The "Load results" dialog then opens. In this dialog, you can check whether or not the loading operation was successful and take any further action that may be necessary.
6. Click the "Finish" button.

Result

The selected project data was downloaded to the devices.

See also

General information on loading (Page 355)

Downloading project data to a memory card (Page 359)

Uploading project data from a device (Page 360)

Downloading project data to a memory card

You can drag-and-drop project data to load it to a memory card. For CPUs of the S7-300/400 series you can also explicitly download the user program to a memory card inserted in the CPU.

Requirement

A memory card is displayed.

See also: Accessing memory cards (Page 414)

Dragging project data to a memory card

To download project data to a memory card, follow these steps:

1. In the project tree, drag the project data you want to load to the memory card.
   The "Load preview" dialog opens. This dialog displays messages and proposes actions necessary for downloading.
2. Check the messages, and select the actions in the "Action" column, if necessary.
   As soon as loading becomes possible, the "Load" button is enabled.
3. Click the "Load" button.
   The loading operation is performed. The "Load results" dialog then opens. In this dialog, you can check whether or not the loading operation was successful and take any further action that may be necessary.
4. Click the "Finish" button.

Download the user program to a memory card in the CPU (S7-300/400 only)

To download the user program to a memory card in a CPU of the S7-300/400 series, follow the steps below:

1. Select a CPU of the S7-300/400 series in the project tree.
2. Select the "Download user program to memory card" command in the "Online" menu.
   The "Load preview" dialog opens. This dialog displays messages and proposes actions necessary for downloading.
3. Check the messages, and select the actions in the "Action" column, if necessary.
   As soon as loading becomes possible, the "Load" button is enabled.
4. Click the "Load" button.
   The download is made and the "Load results" dialog is displayed. This dialog displays messages and suggests possible actions.

5. Check the messages, and select the actions in the "Action" column, if necessary.
6. Click the "Finish" button.

See also

General information on loading (Page 355)

Downloading project data to a device (Page 357)

Uploading project data from a device (Page 360)

Uploading project data from a device

The following section describes the general procedure for uploading project data from a device. Which project data you can upload from a device depends on the products installed.

You have the following options for uploading project data from a device to your project:

- Upload as new station
  With this option you upload existing project data of a device to your project as a new station.
- Uploading project data of a device
  With this option you upload project data from the device to an existing CPU in the project. You will find the project data that can be loaded in the online help of the product.

In both cases, all instances of library types are connected again with the corresponding version of the type in the project library during loading. If no suitable type is yet available for a loaded instance or the correct version of the type does not exist in the project library, the type or the version is added to the project library.

Requirement

- A project is open.
- The hardware configuration and software to be downloaded have to be compatible with the TIA Portal. If the data on the device was created with a previous program version or with a different configuration software, please make sure they are compatible.

Upload as new station

To upload the complete device to your project, follow these steps:

1. Select the project name in the project tree.
2. Select "Upload device as new station (hardware and software)" in the "Online" menu.
   The "Upload device to PG/PC" dialog opens.
3. Select the type of interface you want to use for the load operation in the "Type of the PG/PC interface" drop-down list.
4. Select the interface to be used from the "PG/PC interface" drop-down list.
5. Click the "Configure interface" button to the right of the "PG/PC interface" drop-down list to adapt the settings for the selected interface.
   See also: Auto-Hotspot

6. Display all compatible devices by selecting the relevant option and clicking the "Start search" command. In the accessible devices table, select the device from which you want to upload project data.
7. Click on "Load".
   Depending on the selected device, a dialog appears in which you have to enter additional information, such as the position of the module rack.
   The project data of the device is uploaded to the project. You can edit it offline and then download it to the device again.

Uploading project data of a device

To upload only project data from one device to your project, follow these steps:

1. Establish an online connection to the device from which you want to download the project data.
   See also: Auto-Hotspot
2. Select the device in the project tree.
   The "Upload from device (software)" command in the "Online" menu is enabled.
3. Select the "Upload from device (software)" command in the "Online" menu.
   The "Upload preview" dialog box opens.
4. Check the messages in the "Upload preview" dialog, and select the necessary actions in the "Action" column.
   As soon as uploading becomes possible, the "Upload from device" button is enabled.
5. Click the "Upload from device" button.
   The loading operation is performed.

See also

General information on loading (Page 355)

Downloading project data to a device (Page 357)

Downloading project data to a memory card (Page 359)

Loading project data from a memory card

You have the following options for uploading project data from a memory card to your project:

- Upload project data of the memory card as a new station
  With this option you upload the project data of a memory card to your project as a new station.
- Upload project data of the memory card to an existing device
  With this option you upload project data of a memory card to an existing device in your project. You will find the project data that can be loaded in the online help of the product.

In both cases, all instances of library types are connected again with the corresponding version of the type in the project library during loading. If no suitable type is yet available for a loaded instance or the correct version of the type does not exist in the project library, the type or the version is added to the project library.

Requirement

- A project is open.
- The memory card is displayed.
  See also: Accessing memory cards (Page 414)
- The hardware configuration and software to be downloaded have to be compatible with the TIA Portal. If the data on the memory card was created with a previous program version or with different configuration software, please make sure they are compatible.

Upload project data as a new station

To upload project data from a memory card to your project, follow these steps:

1. In the project tree, select the project data you want to upload.
2. Select "Upload device as new station (hardware and software)" in the "Online" menu.

Or:

1. In the project tree, drag the memory card folder to the project.

Or:

1. Right-click on the memory card.
2. Select "Copy" in the shortcut menu.
3. Right-click on the project.
4. Select the "Paste" command in the shortcut menu.

Uploading project data to an existing device

To upload the project data of a memory card to an existing device, follow these steps:

1. In the project tree, drag the folder of the memory card to a device in the project or copy the memory card and paste the data into a device.
   The "Upload preview" dialog box opens.
2. Check the messages in the "Upload preview" dialog, and select the necessary actions in the "Action" column.
   As soon as uploading becomes possible, the "Upload from device" button is enabled.
3. Click the "Upload from device" button.
   The loading operation is performed.

7.5.2 Comparing project data

7.5.2.1 Basics of project data comparison

Function

You can compare project data of the same type in order to determine possible differences. In principle, the following comparison methods are available:

- Online/offline comparison
  With this comparison method you can compare the software of objects of a device with the objects of a project. This is only possible when you establish an online connection to the device.
- Offline/offline comparison
  With this comparison method you can either compare the software or the hardware. When you compare the software, you can compare objects from projects or libraries. The hardware comparison is available for devices from the currently open project or from reference projects. You can decide for the software as well as the hardware comparison whether the comparison should be performed automatically for all objects or whether you want to compare individual objects manually.
- Detailed comparison
  For some objects, for example, blocks, you can also perform a detailed comparison in addition to the online/offline and offline/offline comparison. This involves opening the blocks to be compared beside each other and highlighting the differences.

A simple online/offline comparison is performed as soon as you establish an online connection. During this process, comparable objects in the project tree are marked with icons that represent the result of the comparison.

The normal online/offline and offline/offline comparison is performed in the compare editor. With the software comparison, you can also select actions for non-identical objects in the compare editor.

Note

- Not all objects allow all types of comparison. Which comparison method you can use for which project data depends on the products installed.
- Compile your user program before you start a comparison or detailed comparison. After each change of the program during a comparison, repeat this step before you update the result of the comparison. This ensures that the comparison shows the current status.

See also

Carrying out an online/offline comparison (Page 364)

Carrying out offline/offline comparisons (Page 364)

Using the compare editor (Page 365)

Running a detailed comparison (Page 373)

7.5.2.2 Carrying out an online/offline comparison

The online/offline comparison lets you compare objects of a device with objects of a project.

Requirement

The project tree is open.

Procedure

To perform an online/offline comparison, follow these steps:

1. Select a device in the project tree that allows online/offline comparison.
2. Select the "Compare > Offline/online" command in the shortcut menu.
3. If you have not already established an online connection to this device, the "Go online" dialog opens. In this case, set all the necessary parameters for the connection and click "Connect".
   The online connection is established and compare editor opens.

Result

All objects that exist online and offline are displayed. The symbols in the compare editor and in the project tree show you the status of the objects. In the compare editor, you can now define certain actions for the objects, depending on their status.

See also

Basics of project data comparison (Page 363)

Carrying out offline/offline comparisons (Page 364)

Using the compare editor (Page 365)

Running a detailed comparison (Page 373)

7.5.2.3 Carrying out offline/offline comparisons

With offline/offline comparison you have the option to compare the project data of two devices. You can carry out a software as well as a hardware comparison. When you compare the software, you can compare objects from projects or libraries. The hardware comparison is available for devices from the currently open project or from reference projects. You can decide whether the comparison should be performed automatically for all objects or whether you want to compare individual objects manually.

You can drag any other device to the drop area at any time to perform further comparisons.

Requirement

The project tree is open.

Procedure

To perform an offline/offline comparison, follow these steps:

1. Select a device in the project tree that allows offline/offline comparison.
2. Select the "Compare > Offline/offline" command in the shortcut menu.
   The compare editor opens and the selected device is displayed in the left area.
3. Drag-and-drop an additional device to the drop area of the right pane.
   All existing objects of the selected devices are displayed depending on the settings of the compare editor in the "Software" tab and an automatic comparison is carried out. You can identify the status of the objects based on the symbols in the compare editor. You can define certain actions depending on the status of the objects.
4. If you want to carry out a manual comparison, click on the button for switching between automatic and manual comparison in the status and action area. Then select the objects that you want to compare.
   The properties comparison is displayed. You can identify the status of the objects based on the symbols. You can define certain actions depending on the status of the objects.
5. If you want to carry out a hardware comparison, open the "Hardware" tab. You can once again carry out a manual comparison, if necessary. But you cannot specify any action.

See also

Basics of project data comparison (Page 363)

Carrying out an online/offline comparison (Page 364)

Using the compare editor (Page 365)

Running a detailed comparison (Page 373)

7.5.2.4 Using the compare editor

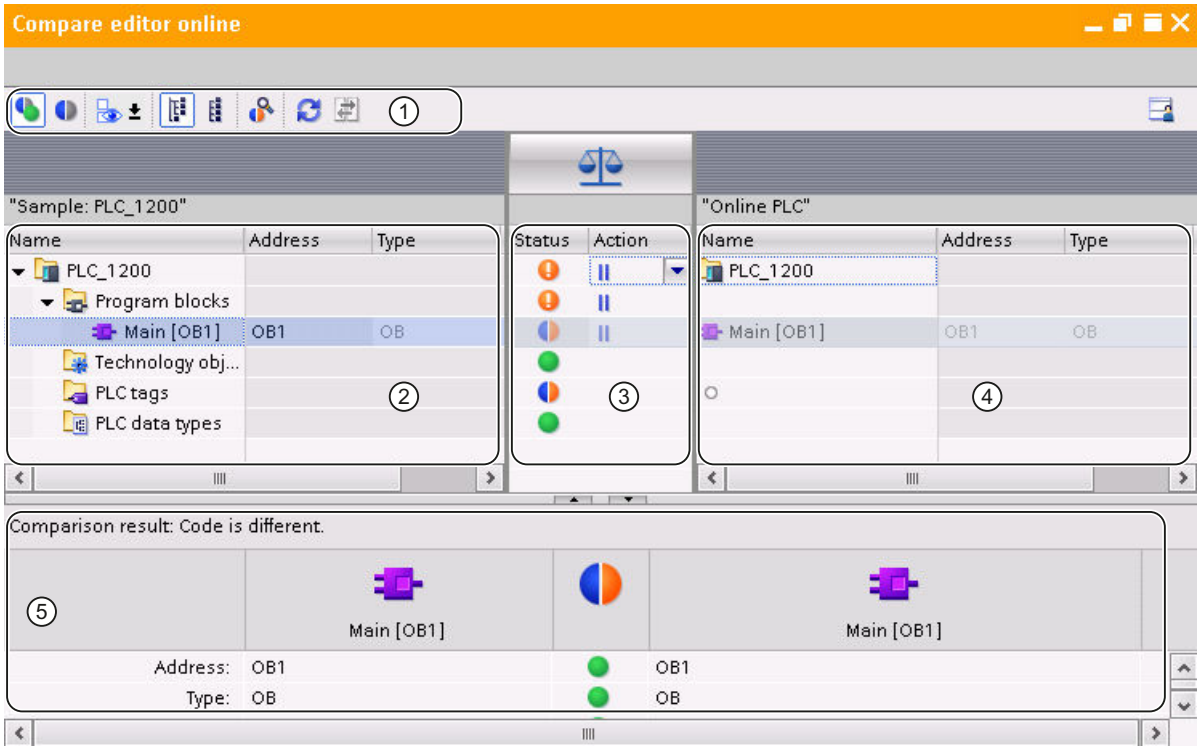
Overview of the compare editor

Function

The compare editor gives an overview of the results of a comparison in a table. The appearance varies slightly depending on whether it is an online/offline comparison or a hardware/software comparison.

Layout of the compare editor

The following figure shows the layout of the compare editor for an online/offline comparison:



① Compare editor toolbar

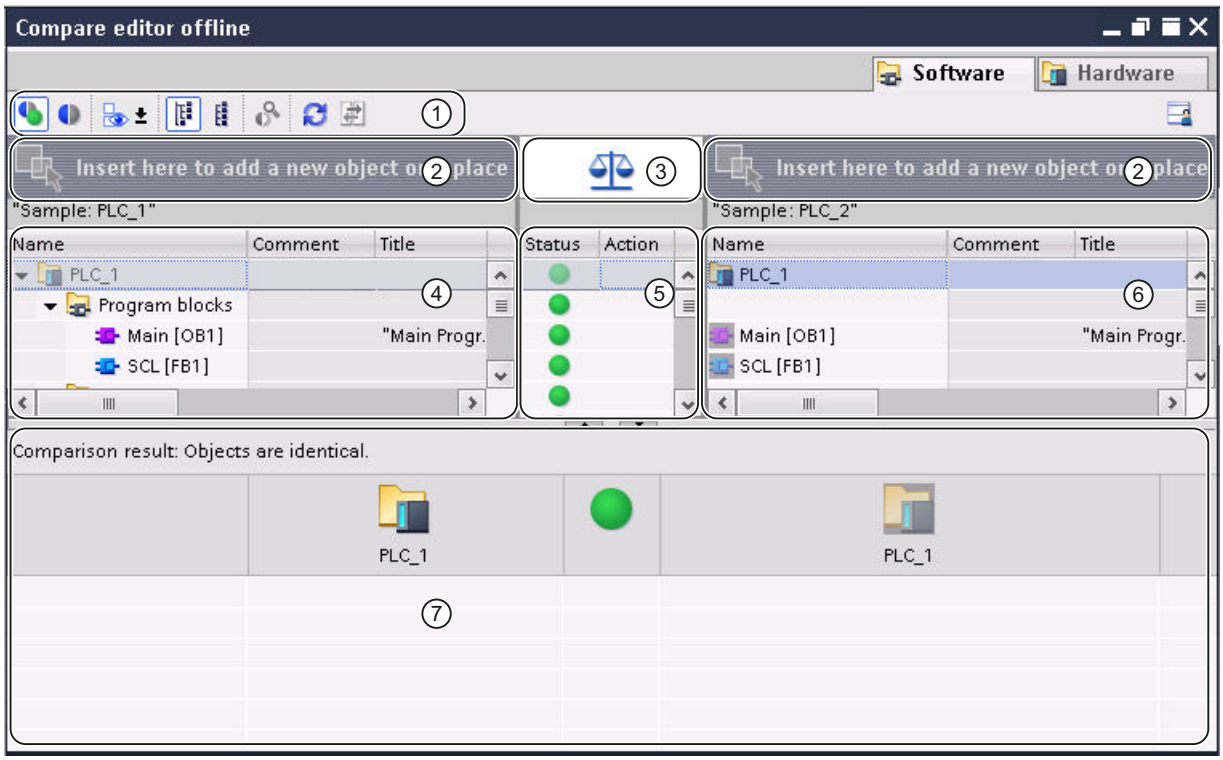
③ Left comparison table

③ Status and action area

④ Right comparison table

⑤ Properties comparison

The following figure shows the layout of the compare editor for an offline/offline comparison (software):



① Compare editor toolbar

② Drop areas

③ Button to toggle between automatic and manual comparison

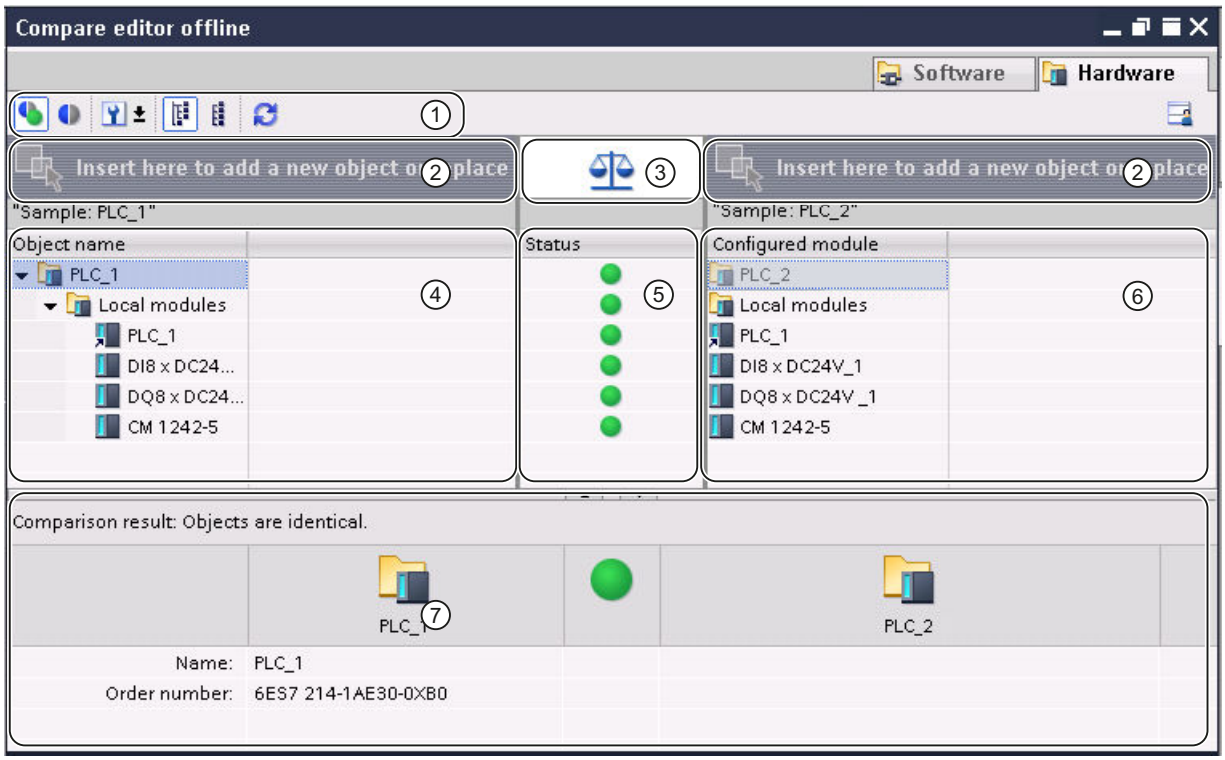
④ Left comparison table

⑤ Status and action area

④ Right comparison table

⑦ Properties comparison

The following figure shows the layout of the compare editor for an offline/offline comparison (hardware):



① Compare editor toolbar

② Drop areas

③ Button to toggle between automatic and manual comparison

④ Left comparison table

⑤ Status area

④ Right comparison table

⑦ Properties comparison

Compare editor toolbar

With the toolbar, you can access the following compare editor functions:

- Only show different objects
  You can hide identical objects to make the comparison easier to follow.
- Show identical and different objects
  You can show identical objects if you want to view the comparison in full.
- Scope of the comparison (only online/offline comparison and offline/offline comparison for software)
  You can define which objects are to be compared.
- Select assignment criterion (only offline/offline comparison for hardware)
  You can specify the criterion according to which the modules are to be assigned to each other for the comparison.

- Changing the view
  You can choose between a hierarchical and a flat view. In the hierarchical view, the devices are shown in their structure; in the flat view, the objects of the devices are listed without structure.
- Start detail comparison (only online/offline comparison and offline/offline comparison for software)
  You can start a detailed comparison for objects to show the individual differences. This function is, however, not available for every object.
- Updating comparison results
  After you have modified objects, you can update the comparison results using this function.
- Synchronize non-identical objects (only online/offline comparison and offline/offline comparison for software)
  You can synchronize non-identical objects using specific actions.

Drop areas

In the case of an offline/offline comparison, you can drag the devices you want to compare into the drop areas. In the case of a software comparison, the devices to be compared can originate from the opened project, from reference projects, from the project library or from global libraries. However, note that you can only drop complete libraries into the right drop area. In the case of a hardware comparison, you can compare devices from the opened project or from reference projects.

Button to toggle between automatic and manual comparison

With an offline/offline comparison, you can switch between automatic and manual comparison. In the case of automatic comparison, the objects to be compared are assigned automatically to each other. In the case of a manual comparison, you can select the objects that are to be compared.

Comparison tables

Comparison tables show the objects of the devices being compared to one another.

The following table shows the meaning of the columns of the comparison table:

Column	Description
Name	Name of the compare object
Comment	Comment on the compare object
Title	Title of the compare object
Address	Address of the compare object
Numbering	Type of numbering for the comparison object
Type	Type of compare object
Language	Programming language set for the compare object.
Time stamp interface	Time of the last modification to the block interface
Time stamp code	Time of the last modification to the source code
Author	Name of the author of the compare object

Column	Description
Version	Version of the compare object
Family	Name of the object family
Load memory	Memory usage of the load memory of the compare object
Work memory	Memory usage of the work memory of the compare object
Modified on	Time of last modification
Signature	Signature of the compare object (SIMATIC Safety)
Interface signature	Signature of the block interface of the compare object (SIMATIC Safety)

Not all columns are shown in the default setting. However, as in all table editors, you can show or hide the columns as required and sort according to individual columns. In the case of a hardware comparison, the comparison tables only contain the "Name" column.

Status and action area

The status and action area offers the following options:

- You can view the results of automatic comparison. The results are displayed with symbols.
- In an online/offline comparison and an offline/offline comparison for software, you can specify actions for non-identical objects.

Status and action symbol

The following table shows the comparison results symbols for an online/offline comparison:

Symbol	Description
	Folder contains objects whose online and offline versions differ
	Comparison results are not known
	Online and offline versions of the object are identical
	Online and offline versions of the object are different
	Object only exists offline
	Object only exists online

The following table shows the comparison results symbols for an offline/offline comparison:

Symbol	Description
	Reference program
	Version compared
	Folder contains objects of which the versions compared differ
	Results of the offline/offline comparison are not known

Symbol	Description
	The versions of the object compared are identical
	The versions of the object compared differ
	Object only exists in the reference program
	Object only exists in the version compared
	Hardware comparison only: Although the lower-level objects of the container are identical, there are differences between the containers themselves. Such a container may be a rack, for example.
	Hardware comparison only: The lower-level objects of the container are different. There are also differences between the containers. Such a container may be a rack, for example.

The following table shows the symbols for possible actions in a software comparison:

Symbol	Description
	No action
	Overwrite the object of the compared version with the object from the reference program
	Overwrite the object of the output program with the object from the compared version
	Different actions for the compare objects in the folder

Properties comparison

The properties comparison compares the properties of the selected compare objects. The result is displayed with symbols. Only the properties comparison is made with a manual comparison so that the status and action area remains empty. With automatic comparison, you can perform the property comparison in addition to the comparison in the comparison tables.

See also

Basics of project data comparison (Page 363)

Carrying out an online/offline comparison (Page 364)

Carrying out offline/offline comparisons (Page 364)

Changing the view (Page 377)

Show and hide table columns (Page 372)

Filtering the compare editor view (Page 372)

Updating the comparison results (Page 374)

Synchronizing non-identical objects (Page 375)

Show and hide table columns

In a software comparison, you can show or hide the columns of comparison tables as required.

Procedure

To show or hide table columns, follow these steps:

1. Click a column header.
2. Select the "Show/Hide" command in the shortcut menu.
   The selection of available columns is displayed.
3. To show a column, select the column's check box.
4. To hide a column, clear the column's check box.

The change affects both the left and the right comparison table.

Result

The columns in both the left and the right comparison table are shown or hidden.

See also

Basics of project data comparison (Page 363)

Carrying out an online/offline comparison (Page 364)

Carrying out offline/offline comparisons (Page 364)

Overview of the compare editor (Page 365)

Filtering the compare editor view (Page 372)

Running a detailed comparison (Page 373)

Updating the comparison results (Page 374)

Synchronizing non-identical objects (Page 375)

Changing the view (Page 377)

Filtering the compare editor view

You can improve the clarity of the compare editor using the following filters:

- Hiding identical comparison objects
  You can hide comparison objects which have identical online/offline or offline/offline versions. Any such comparison objects you have hidden can also be shown again at any time.
- Displayed objects
  In an online/offline comparison or an offline/offline comparison for software, you can specify the objects for which the comparison results are to be shown.

Requirement

The compare editor is open.

Hiding identical comparison objects

To hide identical objects, follow these steps:

1. Click on the "Show only objects with differences" button in the toolbar.
   Only the elements that differ online and offline are displayed.

Showing identical comparison objects

To show identical objects again, follow these steps:

1. Click on the "Show identical and different objects" button in the toolbar.
   All elements will be displayed.

Selecting displayed objects

To select the objects for which comparison results should be displayed, follow these steps:

1. Perform an online/offline or an offline/offline comparison for software.
2. Click the arrow of the button "Show additional available filters" in the toolbar.
3. Select the required filter.

See also

Basics of project data comparison (Page 363)

Carrying out an online/offline comparison (Page 364)

Carrying out offline/offline comparisons (Page 364)

Overview of the compare editor (Page 365)

Changing the view (Page 377)

Show and hide table columns (Page 372)

Running a detailed comparison (Page 373)

Updating the comparison results (Page 374)

Synchronizing non-identical objects (Page 375)

Running a detailed comparison

Note

Not all objects allow a detailed comparison. The project data for which you can perform a detailed comparison depends on the products installed. A detailed comparison of the hardware components is not available for hardware comparison.

Procedure

Proceed as follows to perform a detailed comparison:

1. First, perform an online/offline or an offline/offline comparison for software.
   The compare editor opens.

   Note

   You can only perform a detailed comparison for objects that are listed in the left as well as the right comparison table.

2. In the compare editor, select the object for which you want to perform a detailed comparison.
3. Click the "Start detailed comparison" button in the toolbar.

See also

Basics of project data comparison (Page 363)

Carrying out an online/offline comparison (Page 364)

Carrying out offline/offline comparisons (Page 364)

Overview of the compare editor (Page 365)

Show and hide table columns (Page 372)

Changing the view (Page 377)

Filtering the compare editor view (Page 372)

Updating the comparison results (Page 374)

Synchronizing non-identical objects (Page 375)

Updating the comparison results

As soon as you change an object, the comparison results are no longer valid and must be updated.

Note

For online/offline comparisons, you should note that changes in the device may result in the system automatically updating the comparison editor if objects in the comparison are affected by the change. This can have the following results:

- Some of the actions you have defined may become invalid, for example if the device no longer contains the object in question. Objects with such invalid actions will be highlighted so you can define new, valid actions.
- The selection you made before the automatic update may also be cancelled.

Requirement

The comparison editor is open.

Procedure

To update the comparison results, follow these steps:

1. Click the "Refresh view" button in the toolbar.
   The comparison results are updated.

Note

Please note that the "Refresh view" button will not be available while the comparison editor is loading or synchronizing content.

See also

Basics of project data comparison (Page 363)

Carrying out an online/offline comparison (Page 364)

Carrying out offline/offline comparisons (Page 364)

Overview of the compare editor (Page 365)

Show and hide table columns (Page 372)

Changing the view (Page 377)

Filtering the compare editor view (Page 372)

Running a detailed comparison (Page 373)

Synchronizing non-identical objects (Page 375)

Synchronizing non-identical objects

Specifying actions

If you have performed a comparison, you can specify the actions to be performed for non-identical objects in the compare editor. You cannot select any actions for identical objects. Note that you cannot execute any actions during hardware comparison.

In the case of an online/offline comparison, only synchronization actions in one direction are permitted, in order to retain program consistency. Thus, for example, you can load multiple blocks to a device or from a device, but you cannot perform a combination of loading actions in one synchronization action. In this case, the first action you set in the compare editor determines the synchronization direction. For example, if you specify for a block that the offline block is to be loaded to the device, then the other objects can also only be loaded to the device via a synchronization action. To load objects from the device again, first select the "No action"

option. You can then specify the action settings again as required. Or, you can perform a new comparison.

Note

Please note the following CPU-specific aspects when defining actions:

- S7-300/400:
  - You can define actions for the "Program blocks" folder, for folders you have created yourself or for individual blocks.
  - Neither SCL nor GRAPH blocks can be loaded from the device to the offline project.
- S7-1200/1500:
  - You can define actions for the "Program blocks" folder, for folders you have created yourself or for individual blocks. If you have performed an online/offline comparison and select download to the device as action, a consistent download is executed. If you upload the object from the device to the project, however, you can also upload individual blocks.
  - SCL blocks cannot be loaded from the device to the offline project.

Requirement

The compare editor is open.

Procedure

To select an action for a non-identical object, follow these steps:

1. In the status and action area, double-click in the "Action" column on the cell of the object for which you want to define an action.
   The cell changes to a drop-down list.
2. Click on the drop-down list.
3. Select the action you want.
   The action set will be carried out for the object in question the next time synchronization is performed.
   If you have accidentally changed the action you had selected, you can undo the change before the next synchronization.
4. To restore the previous action selected, right-click on the object or folder.
5. Select the "Restore previous selection" command in the shortcut menu.

See also

Basics of project data comparison (Page 363)

Carrying out an online/offline comparison (Page 364)

Carrying out offline/offline comparisons (Page 364)

Overview of the compare editor (Page 365)

Show and hide table columns (Page 372)

Filtering the compare editor view (Page 372)

Updating the comparison results (Page 374)

Synchronizing objects (Page 377)

Synchronizing objects

Synchronization executes the actions you have specified for non-identical objects. Note, however, that in the case of an online/offline comparison you can only perform actions in one direction in one synchronization action.

Requirement

- The compare editor is open.
- The desired actions have been selected.

Procedure

To synchronize objects, follow these steps:

1. Click the "Execute actions" button in the toolbar.

Result

The actions you specified for the objects are performed.

See also

Basics of project data comparison (Page 363)

Carrying out an online/offline comparison (Page 364)

Carrying out offline/offline comparisons (Page 364)

Overview of the compare editor (Page 365)

Show and hide table columns (Page 372)

Filtering the compare editor view (Page 372)

Updating the comparison results (Page 374)

Specifying actions (Page 375)

Changing the view

You can choose between a hierarchical and a flat view for the left comparison table. In the hierarchical view, the devices are shown in their structure; in the flat view, the objects of the devices are listed without structure. In the right comparison table, the objects are always displayed flat.

Setting the hierarchical view

To set the hierarchical view, follow these steps:

1. Click the "Display in hierarchical view" button in the toolbar of the compare editor.

Setting the flat view

To set the flat view, follow these steps:

1. Click the "Display in flat view" button in the toolbar of the compare editor.

See also

Basics of project data comparison (Page 363)
Carrying out an online/offline comparison (Page 364)
Carrying out offline/offline comparisons (Page 364)
Overview of the compare editor (Page 365)
Show and hide table columns (Page 372)
Filtering the compare editor view (Page 372)
Running a detailed comparison (Page 373)
Updating the comparison results (Page 374)
Synchronizing non-identical objects (Page 375)

7.5.3 Protecting project data

7.5.3.1 Protection concept for project data

Introduction

You can protect your project data from unauthorized access. These include, for example:

- Access protection for devices
- Copy and display protection of objects
- Restrictions for printouts of know-how-protected objects

For objects with know-how protection, this protection is also retained after the object is pasted into a library. Note that every protection mechanism is not available for all objects. How to protect specific objects is described in the online help of the product.

Revoking access rights for devices

If you want to execute a function that is password-protected by means of the device protection level, you are prompted to enter a password. When the password is entered correctly, you can

execute the required function. You continue to have access rights on the device until you close the TIA Portal.

If you want to reactivate password protection while the TIA Portal is open, you can explicitly revoke the access rights for a device. As a result, certain functions for the protected device cannot be executed until the correct password is entered again. You specify the functions for which a password must be entered when you assign the device protection level.

See also

Printing project data (Page 397)

7.5.3.2 Revoking access rights for devices

Requirement

- A protection level has been set for the device.
- A protected function for the device has been enabled by entering the password.

Procedure

To revoke the access rights for the device, follow these steps:

1. Select the device for which you want to revoke access rights in the project tree.
2. Select the "Delete access rights" command in the "Online" menu.

Result

The access rights are revoked, and starting from now the user will be prompted to enter the password again to execute a password-protected function on the device. The function can only be executed if the correct password is entered.

If the device has an online connection, it will be disconnected.

See also

Protection concept for project data (Page 378)

7.5.4 Printing project contents

7.5.4.1 Printing project documentation

Documentation settings

Introduction

Once a project is created, the contents can be printed in an easy-to-read format. You may print the entire project or individual objects within the project. A well-structured printout is helpful when editing the project or performing service work. The printout can also be used for your customer presentations or as full system documentation.

You can prepare the project in the form of standardized circuit manuals and print it in a uniform layout. You can limit the scope of the printout. You have the option to print to the entire project, individual objects along with their properties, or a compact overview of the project. In addition, you can print the contents of an open editor.

Improving the printout with frames and cover pages

You can design the appearance of the printed pages according to your own requirements, for example, to add your own company logo or the corporate design of your company in the project documentation. You can create any number of design variants as frames and cover pages. The frames and cover pages are stored in the project tree under the item "Documentation settings" and are part of the project. You can insert placeholders for data from previously entered document information within the frames and cover pages. These will be filled automatically with the appropriate metadata during printing.

If you want to avoid designing your own template, there are ready-made frames and covers pages available. These include templates complying with the ISO standard for technical documentation.

Modular structure of a printout

An printout generally consists of the following components:

- Cover page (only when printing from the project tree)
- Table of contents (only when printing from the project tree)
- Name and path of an object within the project tree
- Object data

Printout of the cover page or the table of contents can be deactivated in the "Print" dialog.

See also

Creating frames (Page 386)

Creating a cover page (Page 386)

Editing cover pages and frames (Page 388)
Entering document information (Page 384)
Print function for module labels (Page 399)

Printout of project contents

Availability of print function

The following contents can be printed:

- An entire project in the project tree
- One or more project-related objects in the project tree
- Contents of an editor
- Tables
- Libraries
- Diagnostics view of the Inspector window

It is not possible to print in the following areas:

- Portal view
- Detailed view
- Overview window
- Compare editor
- All tabs of the Inspector window, except the diagnostics view
- All task cards, except the libraries
- Most of the dialogs
- Properties and devices of the programming device/PC not related to the project, for example online portals and connected card readers.

Scope of printout

To be able to print, at least one printable element has to be selected.

If a selected object is printed, all subordinate objects are also printed. For example, if a device is selected in the project tree, all of its data is also printed. If you select the entire project in the project tree for printing, all project contents are printed with the exception of the graphical views. These have to be printed separately. Items in the project tree that are not part of the project cannot be printed. For example, this includes online portals and connected card readers and USB memory devices.

When table contents are printed, all lines in the table in which a cell is selected are printed. In order to print one or more table columns, the desired columns must be selected. If no individual cells or columns are selected, the entire table is printed.

Limitations when printing

In general, it is possible to print all objects that can be visualized on the user interface. Conversely, this means that you cannot print objects that you do not have access to. If a printout fails, possible reasons may include the following:

- A valid license does not exist for displaying an object.
- There is no device description for an object.
- A software component needed to display an object is not installed.

See also

Printing project data (Page 397)

Changing the print settings

Changing the print settings

You can specify general print settings that are retained even after the TIA Portal is closed and re-opened. Some settings are dependent on the products installed. The following settings are possible in every case:

Always print table data as pairs of values

If this option is selected, tables are not printed in tabular format but rather as a pairs of key and value.

Example:

Object name	Property 1	Property 2
Object A	Value A1	Value A2
Object B	Value B1	Value B2

In this case, the printout has the following appearance:

Object A

Property 1: Value A1

Property 2: Value A2

Object B

Property 1: Value B1

Property 2: Value B2

Printing mask editors

- Always print data in tables
  All parameters of technology objects are printed in tabular format.
- Print mask graphics if possible
  If the utilized editor supports this function, the contents of the editor are not printed as a table but rather as a complete graphic as it appears on the screen.

Procedure

To change the print settings, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "General" group.
3. Select the desired default settings in the "Print settings" area.
   The changes are applied immediately and are retained for all projects, even after the TIA Portal is closed.

See also

Overview of the print settings (Page 276)

Specifying the print layout

Specifying the print layout

If you do not want to rely on ready-made print templates, you can specify your own cover page or your own layout for the individual pages. Your designs are saved together with the respective project.

Your designs for the cover page and your templates for the page layout can be found in the project tree under the "Documentation information" group. You will also find metadata on the project there under the entry "Document information". For subsequent print operations, you can customize the appearance of the printout in the "Print" dialog using the saved cover pages and page layout templates and the available metadata.

Designing the cover page

The cover page can be customized. You can insert a background graphic and provide placeholders for text on the page. The placeholders are automatically filled with data from a documentation information during printing.

Cover pages are located in the project tree under the "Documentation information > Cover pages" group.

Designing the content page

The regular pages of a printout can contain the following elements:

- Frame with static content, such as a company logo
- Placeholders for text, such as the name of the project, the page number, and the time the printout was started
  Several different values for the individual placeholders can be specified in the document information. Other values, such as the project name, are preassigned and are inserted automatically during printing.
- Footnote
  The footnote is always output below the content area.
- Content area
  You can specify an area where the printed content is to be embedded.

The design of the content pages is saved in Frames. The individual frames are located in the project tree under the "Documentation information > Frames" group.

Entering document information

You can enter metadata in the document information for every project. In addition, a print frame and a cover page are specified in the document information. You can create different information, if required, to enable you to quickly switch between different document information containing different information, frames, cover pages, page sizes, and page orientations when printing. For example, this is useful if you want to generate printouts in different languages and different document information is provided for each language.

In the documentation editor, you can specify placeholders on the cover page or in the frame of the regular pages. These placeholders can be automatically replaced with metadata from the documentation information during printing.

The various document information are therefore part of the printing function and specify the print layout and print content.

Procedure

To add metadata, follow these steps:

1. To create new document information, double-click "Add new document information" under "Documentation information > Document information" in the project tree.
   The new document information is created and opened immediately.
2. Enter a name for the set in the "Name" field.
3. Fill in the individual fields with the metadata for the project.

Managing cover pages and frames

Using cover pages and frames

Uses for cover pages

You can give your plant documentation printouts a professional appearance by adding a cover page. You can design your own cover page or use ready-made cover pages. Ready-made cover pages can be adapted and stored again as a template.

Cover pages can be saved in global libraries where they are available for use across projects.

Cover pages are designed for use as a right printed page only.

Uses of frames

You can embed the regular pages of your plant documentation inside a consistently uniform page frame. The frame can contain placeholders for project metadata, which is stored in the document information. It can also contain graphic elements that you design yourself.

You can create your own frames or rely on ready-made page frames. You can adapt a ready-made page frame and then store it again as a new frame.

Like cover pages, frames can be saved in global libraries where they are available for use across projects.

Frames are designed for use on right printed pages only.

Cover pages and templates in the project tree

Cover pages and frames associated with the project are stored in the project tree under the entry "Documentation information". There are separate folders here for frames and cover pages.

The following actions are available in the project tree for cover pages and frames.

- Creating your own subfolders
- Copying and pasting
- Inserting cover pages and frames from the "Documentation templates" system library
- Copying cover pages and templates to a global library

Cover pages and templates in libraries

The "Documentation templates" system library contains a few cover pages and templates that are available in every project. The cover pages and templates can be moved from there to the project tree using a drag-and-drop operation. You can then adapt the cover pages and templates in the project tree according to the requirements of your project.

Cover pages and templates can be moved from the project tree to a global library. Afterwards, these are available in every project.

See also

Library basics (Page 415)

Overview of the "Libraries" task card (Page 417)

Designing cover pages and frames (Page 388)

Using ready-made frames and cover pages (Page 387)

Creating frames

You can create any number of frames for each project. The frames are stored in the project tree below the "Documentation information > Frames" group. You can assign a frame to all document information. When you select document information for printing, its associated frame is used.

Procedure

To create a new frame, follow these steps:

1. Double-click the entry "Add new frame" below the "Documentation information > Frames" group in the project tree.
   The "Creating frames" dialog opens.
2. Enter a name for the frame in the "Name" field.
3. Choose the paper size from the "Paper type" drop-down list.
4. Choose whether the page is to be created in portrait or landscape format in the "Orientation" drop-down list.

Click the "Add" button.

Result

A new frame is created. The frame is then opened automatically in the documentation editor where it can be edited.

See also

Editing cover pages and frames (Page 388)

Creating a cover page (Page 386)

Creating a cover page

You can create any number of cover pages for the printout for each project. The cover pages are stored in the project tree below the the "Documentation information > Cover pages" group. You can assign a cover page to all document information. When you select specific document information for printing, its associated cover page is used.

Procedure

To create a new cover page, follow these steps:

1. Double-click the entry "Add new cover page" below the "Documentation information > Cover pages" group in the project tree.
   The "Add new cover page" dialog box opens.
2. Enter a name for the cover page in the "Name" field.
3. Choose the paper size from the "Paper type" drop-down list.
4. Choose whether the page is to be created in portrait or landscape format in the "Orientation" drop-down list.

Click the "Add" button.

Result

A new cover page is created. The cover page is then opened automatically in the documentation editor where it can be edited.

See also

Editing cover pages and frames (Page 388)

Creating frames (Page 386)

Using ready-made frames and cover pages

The TIA Portal comes with some ready-made frames and cover pages. These can change according to your wishes.

Procedure

To create and edit the ready-made frames and cover pages, follow these steps:

1. Open the "Global libraries" pane in the "Libraries" task card.
2. In the "Templates" folder, open the "Cover Pages" or "Frames" folder.
3. Drag a cover page or a frame from one of the folders into the project tree and drop it into one of the following folders:
   - For frames: "Document information > Frames"
   - For cover pages: "Document information > Cover pages".

   The ready-made frame or cover page can now be used in the project.
4. Double-click on the new entry in the project tree click to edit the frame or the cover page.

See also

Using cover pages and frames (Page 385)

Editing cover pages and frames (Page 388)

Designing cover pages and frames

Editing cover pages and frames

The documentation editor is a graphical editor which allows you to design frames and cover pages for your plant documentation. You can place images or text elements on the frame and the cover pages in the document editor. The text elements are either static or they are automatically filled during printing with the data from the document information that you have selected in the print dialog.

Procedure

To edit a cover page or a frame in the documentation editor, follow these steps:

1. In the project tree, double-click on the entry for an existing cover page or frame under the "Documentation information > Frames " or "Documentation information > Cover pages" group.
   The documentation editor opens.
2. Design the cover page or frame as desired.
3. Close the documentation editor.
   The changes to the cover page or frame are applied automatically.

See also

Creating a cover page (Page 386)

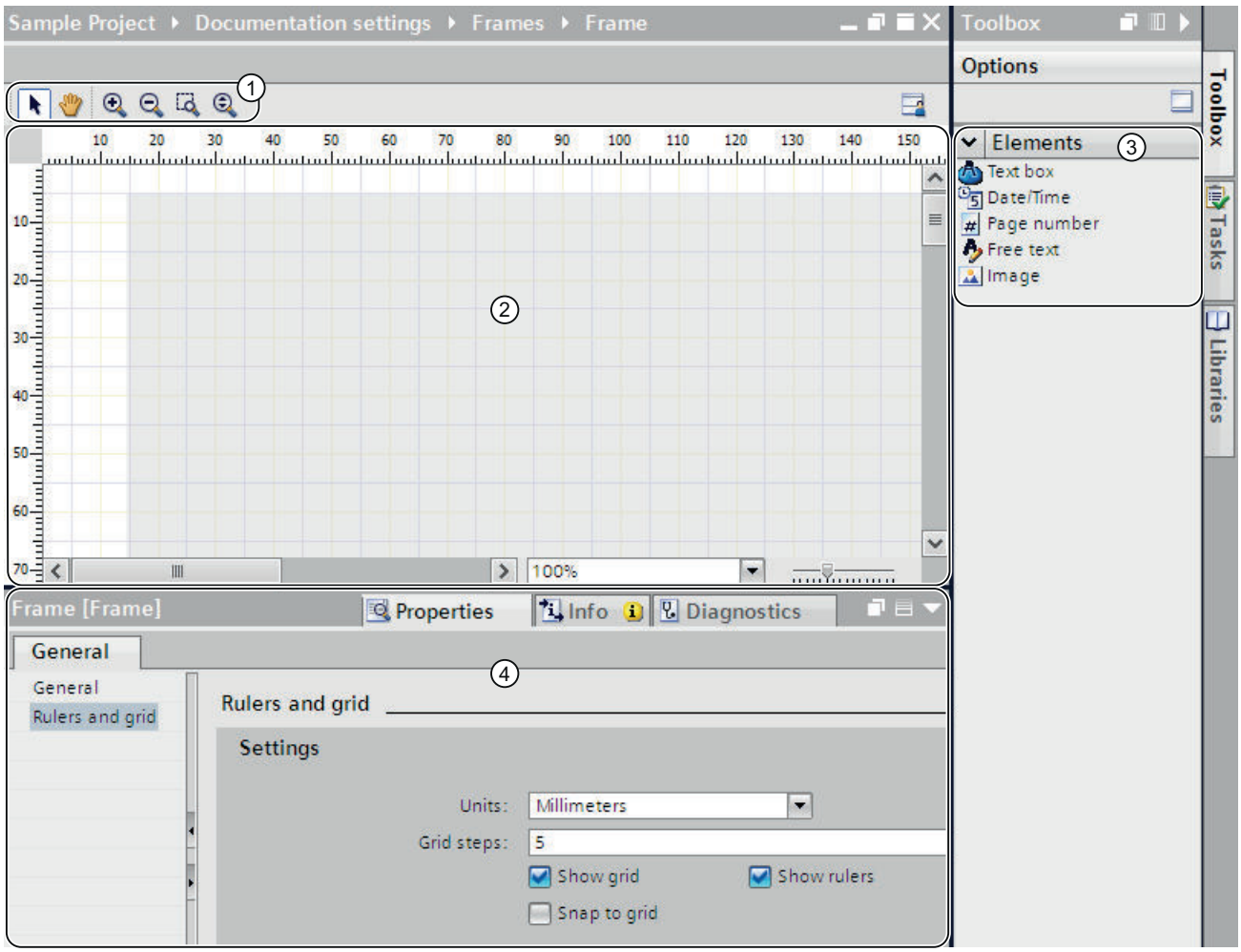
Creating frames (Page 386)

General operation of the documentation editor (Page 389)

General operation of the documentation editor

Components of the documentation editor

The following figure provides an overview of the components of the documentation editor:



① Toolbar

The toolbar provides the following tools (from left to right):

- Arrow tool
  Enables object selection.
- Navigation tool
  Allows shifting of the partial page.
- Zoom-in button
  Magnifies the page display incrementally.
- Zoom-out button
  Reduces the page display incrementally.
- Zoom with selection
  Adapts the page size to a selected work area.
- Dynamic zoom
  Adapts the page width to the work area.

② Work area

You can design the cover page or frame in the work area.

③ "Toolbox" task card

The "Toolbox" task card contains various types of placeholders that you can use on the cover sheet or frame. The placeholders can be placed in the work place using a drag-and-drop operation.

④ Properties in the Inspector window

You can display and modify the properties of the currently selected object in the "Properties" tab of the Inspector window. For example, you can modify the properties of the page, format text, specify the position of objects on the page, etc.

Operation in the documentation editor

The following basic functions are available in the documentation editor:

- Drag-and-drop functionality
  The documentation editor is a graphic editor. which means you can place objects anywhere with the mouse. An image of the page is displayed in the work area. This image corresponds to the ultimate print layout.
  If you want to select objects on the page in order to move them or modify their properties, the arrow tool must be activated in the toolbar.
- Zoom function
  You can use the zoom function to change the size of the page display. You have two options for changing the page size:
  - Via the buttons in the toolbar
    Select the "Zoom in" or "Zoom out" magnifying glass button in the toolbar of the documentation editor. Then click on the page in order to magnify (zoom in) or reduce (zoom out) the page incrementally.
    To zoom in on a particular area, select the "Zoom with selection" tool and use the mouse to drag an outline around the area you want to focus on.
    To continuously zoom in or zoom out of the work area, use the "Dynamic zoom" tool. To magnify the page display, click anywhere on the work area, and then hold down the mouse button while dragging the mouse toward the top of the page. To reduce the page display, drag the mouse toward the bottom of the page.
  - Via the zoom bar
    You can also use the zoom bar (located in the bottom right corner of the work area) to change the display size. Choose a percentage value from the drop-down list or enter a percentage value. Alternatively, you control the display size using the slider.
- Navigation over the page
  In addition to scrolling, you the option of changing the partial page with the navigation tool. To change the partial page with the navigation tool, select the Hand button in the toolbar. Then, click anywhere on the page and hold the mouse button down while moving the page to the desired position.

Using and adapting the positioning aids

You have various aids at your disposal to help you position elements on the page:

- Rulers
  Rulers are affixed to the page margins in the work area.
- Page grid
  A grid is placed underneath the page in the work area.

You can display, hide or adapt the positioning aids in the Inspector window under "Properties > Rulers and grid". You can make the following settings:

- Units:
  Specify the unit of measurement for the grid and the rulers.
- Grid steps:
  Specify the width of the grid.
- Show grid:
  Specify whether the grid is to be displayed or hidden.

- Snap to grid:
  Specify whether objects are to be aligned automatically to the grid. If the option is selected, the grid lines function like a "magnet".
- Show rulers:
  Specify whether the rulers are to be displayed.

See also

Editing cover pages and frames (Page 388)

Specifying the print area (Page 392)

Inserting placeholders for metadata (Page 392)

Specifying the print area

An area within the frame is provided for the actual printed contents. The project data is then always inserted inside this defined and uniformly consistent area within the frame. You can adjust the size of the print area.

Requirement

A frame is open in the documentation editor.

Procedure

To define an area for the printed contents, follow these steps:

1. Click on the slightly darker area within the page display in the documentation editor to select the area for the print content.
   This opens the properties of the area to be printed in the Inspector window.
2. Enter the position of the print area on the X and Y axes in the Inspector window.
3. Specify the width and height of the print area in cm in the Inspector window.

Alternatively, you can change the width and position of the print area in the graphic display of the page. To do so, use the mouse to drag the margins of the print area until the desired size and position are achieved.

See also

Creating frames (Page 386)

General operation of the documentation editor (Page 389)

Inserting placeholders for metadata

You can provide placeholders on the cover page and in a frame. The placeholders are automatically filled with metadata from documentation information during printing, if they are placeholders for text. Alternatively, you can add non-modifiable data, such as free text or an image.

All elements are arranged in numbered Z-Orders. If objects overlap, you can determine in which sequence these are to be arranged.

Types of placeholders

The following types of placeholders are available to you:

- Text field
  The text field stands as a placeholder for a text element from a document information. In the properties of the text field, you set which text from a document information should be automatically inserted during printing.
- Field for date and time
  A date and time is inserted instead of the placeholder when printing. This can be the date of creation or the point in time when the last change was made to the project. In the properties of the Inspector window, you specify which date or time is printed.
- Page number
  The correct page number is automatically applied when printing.
- Free text
  You can enter freely selectable text in the properties of the text field. The text is static and is not influenced by the document information selected at the time of printing.
- Image
  Select the image file in the properties of the placeholder in the Inspector window. Images in the formats BMP, JPEG, PNG, EMF or GIF are possible.

Requirement

An cover page or frame is open in the documentation editor.

Procedure

To insert placeholders for metadata on the cover sheet or in a frame, follow these steps:

1. Drag a field from the "Toolbox > Elements" task card to the work area of the documentation editor.
   The placeholder is inserted. The placeholder properties are shown in the Inspector window and can be edited there.
2. Select the metadata to be inserted during printing from the "Text" drop-down list in the Inspector window under "Properties > General > Text box". Alternatively, you have the option of entering free text or selecting an image depending on the type of placeholder.
3. In the Inspector window under "Properties > General > Position and size", specify the position of the placeholder on the X and Y axis and enter the width and height of the text box in cm. You specify the sequence of the objects in the "Z-Order" field, if these overlap. The smaller the value, the further down an object is located.
4. In the Inspector window, go to "Properties > View" and select the font formatting and the orientation of the text as well as the alignment of the text. You cannot make this setting for images.

See also

General operation of the documentation editor (Page 389)

Displaying print preview

Creating a print preview

Creating a print preview

You can create a preview of the printout. Document information can be chosen for this, in the same way as for the actual printout. In this way, you preview the selected frame and, if applicable, the cover sheet. The settings are retained for later printing.

Procedure

To create a print preview and to set the scope of the later printout, follow these steps:

1. Select the "Print preview" command in the "Project" menu.
   The "Print preview" dialog opens.
2. Select the frame layout you want to use for the printout.
   - In the "Document information" drop-down list, select the documentation information you want to use later for the printout.
   - Select the "Print cover page" check box to print the cover page, which is specified in the selected document information.
   - Select the "Print table of contents" check box to add a table of contents to the printout.

   The check boxes for printing the cover page and the table of contents can only be selected if you have started the printout in the project tree.
3. Under "Print objects/area", select what is to be printed. The selection is only possible if you have started the printout from an editor that supports this function.
   - Choose "All" to print out the entire content of the editor.
   - Choose "Selection" to print only the objects currently selected in the editor.

4. Select the print scope under "Properties".
   - Choose "All" to print all configuration data of the selected objects.
   - Choose "Visible" to print the information of an editor that is currently visible on the screen. This option can only be chosen if you have started the printout from an editor that supports this function.
   - Choose "Compact" to print out an abbreviated version of the project data.
5. Click "Preview" to generate the preview.
   A print preview is created in the work area.

Note

Wait time for extensive documents

It can take several minutes to generate the print preview in the case of very extensive projects. You can continue working normally in the meantime on systems with adequate resources. The progress of the print preview is shown in the status bar.

See also

Operation within the print preview (Page 396)

Operation within the print preview

Functions within the print preview

The print preview shows an exact image of the subsequent printout. You can use the buttons in the toolbar to modify the print preview display. The following functions are available (from left to right):

- Navigation mode
  Allows shifting of the partial page.
  To change the partial page with the navigation tool, select the arrow button in the toolbar. Then, click anywhere on the page and hold the mouse button down while moving the page to the desired position.
- Zoom function
  - "Zoom in" and "Zoom out"
    Magnifies or reduces the page display.
    To zoom in or zoom out the display incrementally, select the corresponding button. Then click on the page in order to magnify (zoom in) or reduce (zoom out) the page incrementally.
    To zoom in on a particular area, enable the "Zoom with selection" icon and use the mouse to drag an outline around the area you want to focus on.
    To zoom dynamically through the page, select the button "Zoom in / zoom out dynamically". With pressed mouse button, scroll down over the page to zoom in. Scroll up to zoom out.
  - Percentage value in the drop-down list
    Specifies the display size of the page in percent.
    Enter a percentage value or select a percentage value from the drop-down list. Alternatively, choose the "Fit to page" option from the drop-down list to adapt the page size to the work area. Or, choose "Fit to width" to adapt the page width to the work area.
- "Forward" and "Backward"
  Each change in the partial page, the page count, or the display size is saved in a history in the background. You can use the "Forward" or "Backward" button to return to the previous view or the next view.
- Page navigation
  - "First page"
    Jumps to the first page
  - "Previous page"
    Goes to the previous page.
  - "Page number" input field
    Shows the current page. To jump directly to a page, enter the page number of the page you want to view.
  - "Next page"
    Goes to the next page.
  - "Last page"
    Jumps to the last page.

See also

Creating a print preview (Page 394)

Printing project data

You have two options for printing out project data:

- Print immediately using default settings by means of the "Print" button in the toolbar.
  The button is only active if a printable object is selected.
- Printout with additional setting options with the "Project > Print" menu command.
  For example, you can choose a different printer or specific documentation information or you can specify whether a cover page and table of contents are to be printed. In addition, you can specify the print scope or display a print preview prior to printing.

Requirement

- At least one printer is configured.
- The objects to be printed are not protected.
  The print scope for protected objects is limited. Disable the know-how protection to print the objects in full.

Printing project data

To print out data from the current project or the entire project with additional setting options, follow these steps:

1. Select the entire project in the project tree in order to print out the entire project. To print only individual elements within a project, select them in the project tree.
2. Select the "Print" command in the "Project" menu.
   The "Print" dialog opens.
3. Select the printer in the "Name" box.
4. Click "Advanced" to modify the Windows printer settings.
5. Select the frame layout you want to use for the printout.
   - Select the documentation information in the "Document information" drop-down list.
     The frame stored in the document information is used for the printout. All placeholders within the chosen frame are filled with the metadata from the selected document information.
   - Select the "Print cover page" check box to print the cover page, which is stored in the selected document information.
   - Select the "Print table of contents" check box to add a table of contents to the printout.

   The check boxes for printing the cover page and the table of contents can only be selected if you have started the printout in the project tree.

6. Under "Print objects/area", select what is to be printed. The selection is only possible if you have started the printout from an editor that supports this function.
   - Choose "All" to print out the entire content of the editor.
   - Choose "Selection" to print only the objects currently selected in the editor.
7. Select the print scope under "Properties".
   - Choose "All" to print all configuration data of the selected objects.
   - Choose "Visible" to print the information of an editor that is currently visible on the screen. This option can only be chosen if you have started the printout from an editor.
   - Choose "Compact" to print out an abbreviated version of the project data.
8. Click "Preview" to generate a print preview in advance.
   A print preview is created in the work area.
9. Click "Print" to start the printout.

Note

Scope of the "Print" dialog

The options available in the "Print" dialog vary depending on the elements to be printed.

Result

The project data is prepared in the background for printing and then printed on the selected printer. The status bar shows the progress of the print operation. You can continue working normally while data is being prepared for printing.

The print results and any errors or warnings are listed in the Inspector window under "Info" at the conclusion of the print job.

Canceling a print job

To cancel an active print job, follow these steps:

1. Click the blue cross in the status bar next to the progress display for the printout.
   The printout is cancelled promptly.

See also

Protection concept for project data (Page 378)

Revoking access rights for devices (Page 379)

Printout of project contents (Page 381)

Designing cover pages and frames (Page 388)

7.5.4.2 Printing module labels

Print function for module labels

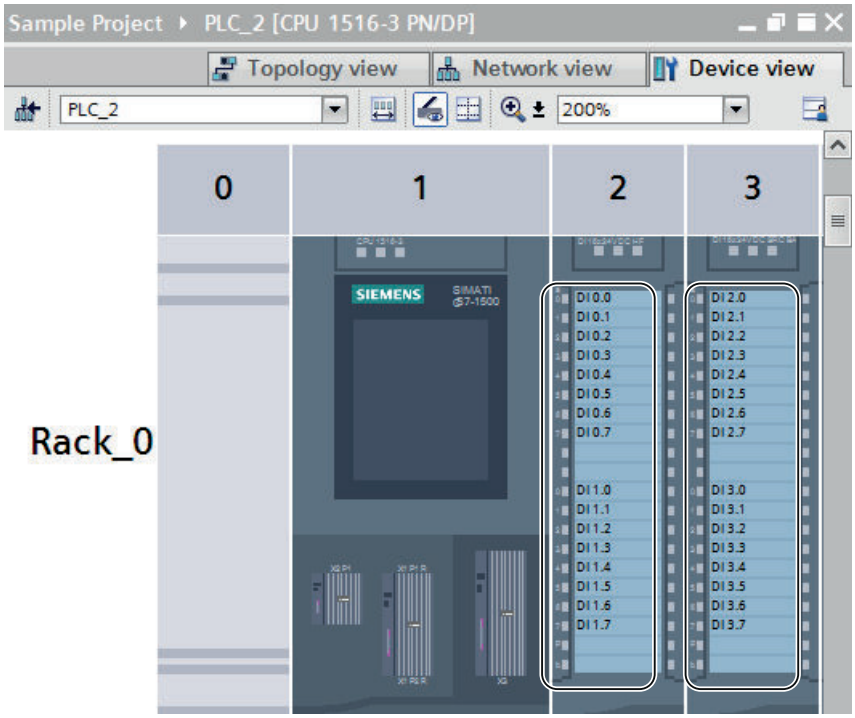
Printing of module labels for hardware modules

You can print labels for the modules in your project with the help of the TIA Portal. The labels are custom-fit to each module and can contain the following printed information:

- Symbolic name of the input or output
- Absolute address of the input or output
- Symbolic name and additionally the absolute address of the input or output. The order of the information can be specified.

The modules are displayed graphically in the device view. If you set the zoom level in the device view to at least 200%, the labels for the individual modules will be visible. The printout on the labels corresponds to the representation of the labeling in the device view.

The following figure shows an example of two modules in the device view on which the labeling of the inputs and outputs is visible:



Export and further editing as Microsoft Word file

Before you can print them, the labels are first exported as a Microsoft Word file (.docx). This file can be further edited with commonly available word processing programs such as Microsoft

Word 2010. The individual labels are represented as a table in the .docx file. You can format the text or the backgrounds of the individual cells as desired.

The character spacing of the text within the table is adapted by default, so that texts are not truncated. If you want to prevent this from stretching or compressing the text too much, change the character spacing of the text in the table cell properties.

Print media

You can print the labels either on ready-made print sheets or on standard DIN A4 paper. You can separate the individual labels from the ready-made print sheets and insert them in the designated labeling areas of your modules. If you print on standard paper, the individual labels must be cut out. Cut marks are automatically included on the printout and serve as aids.

Because the paper feeds of different printers differ slightly, the printout may be slightly offset on the paper. When the labels are printed on ready-made print sheets, printing that is accurate to the millimeter is important, because otherwise the text will not be custom-fit inside the stamped area. In addition, if the printing is imprecise, the labeling of an input or output may no longer be congruent with the channel status displays of the module. For this reason, you can enter an offset value for your printer in the TIA Portal to ensure precise printing. For information on how to determine the proper offset value for your printer, refer to Chapter "Determining the print area offset (Page 402)".

See also

Printing labels (Page 400)

Determining the print area offset (Page 402)

Documentation settings (Page 380)

Printing labels

You can print labels for the modules in your project if provision has been made for attaching labels to the utilized modules. The labels are first exported to a Microsoft Word file (.docx). A separate .docx file is created for each module family (for example, for all selected S7-1500 modules). You can then modify the labels according to your wishes. The labels are always printed from the word processing program.

If no provision has been made for printing labels, the corresponding shortcut menu command is inactive. If you have selected multiple modules for printing of labels and at least one of the modules does not support this, a message is shown on the "Info > General" tab of the Inspector window. The message lists all unsupported modules, and the export of the print file is continued for the supported modules.

Requirement

- The chosen modules must support the printing of labels.
- A word processing program that supports Microsoft Word .docx files must be installed, e.g., Microsoft Word 2010.
- You need the ready-made labels for your modules or commercially available DIN A4 paper.

Procedure

To print labels for hardware modules, follow these steps:

1. In the project tree, select the modules for which you want to print labels.
   - You can select one or more stations in order to print out labels for all modules plugged into these stations.
   - Or, select the desired modules below the stations in the "Local modules" folder.
2. Right-click on one of the devices, and select the "Export labels" command from the shortcut menu.
   The "Export labels" dialog box opens.
3. In the "Content of label" area, select which data are to be printed on the label:
   - Choose "Symbolic name" in order to print the symbolic name of the input or output (corresponds to the contents of the "Name" column in the IO tag table).
   - Choose "Absolute address" in order to print the absolute address of the input or output (corresponds to the contents of the "Address" column in the IO tag table).
   - Choose "Absolute and symbolic address" or "Symbolic and absolute address" in order to print both addresses. The print order corresponds to the indicated order.
4. Select which paper you plan to print on in the "Paper type" area.
   - Choose "Print on SIEMENS label sheet" if you want to print on a ready-made label sheet for your modules.
   - Choose "Print on standard paper" if you want to print on standard DIN A4 paper.
5. In the "Offset print area" select correction values for your printer, if required, for proper orientation of the print area. This is only necessary if you are printing on ready-made labels.
   - Enter a correction value, in millimeters, in the "Vertical offset" field. A negative value shifts the print area upward. A positive value shifts the print area downward.
   - Enter a correction value, in millimeters, in the "Horizontal offset" field. A negative value shifts the print area to the left. A positive value shifts the print area to the right.
6. In the "Path" field, select the path to which the exported .docx files should be stored.
7. Click the "Export" button to start the export to a .docx file.
   The .docx files are created.
8. Open the .docx files with a conventional word processing program, such as Microsoft Word 2010, and change the design of the labels if necessary.
9. Print out the labels from your word processing program. To do so, use the paper that you specified in the Export dialog box.
10. If you are using ready-made sheets, separate the labels at the stamped lines provided for that purpose. When standard DIN A4 paper is used, you must cut out the labels.

See also

Determining the print area offset (Page 402)

Determining the print area offset

If you are using a ready-made label sheet, the printing on it must be applied precisely so that the text is properly oriented on the prestamped labels and will have the proper fit relative to the channel status displays of the module. However, the paper feeds vary slightly from one printer to another. For this reason, you must enter a suitable correction value for your printer in the TIA Portal, if necessary. The print area is then shifted in the exported .docx file in such a way that the printing fits precisely on the ready-made label sheet.

The settings for shifting the print area are stored for the specific Windows user. If you log on to Windows using a different user name, you have to enter the correction values again.

The procedure for determining the correction value for your printer is described below.

Requirement

- You require a ready-made label sheet.
- You must have access to the actual printer that you will use subsequently for the printout. The printer must be made ready for printing on standard DIN A4 paper.

Procedure

To determine the correction value for your printer, follow these steps:

1. Print out a label sheet on standard DIN A4 paper, as described in Chapter "Printing labels (Page 400)".
2. Compare the printout on the DIN A4 paper with the ready-made label sheet.
3. If the print area is offset, you must use correction values.
   - Using a ruler, measure the horizontal offset relative to the ready-made label sheet. This will be entered later in the "Horizontal offset" field of the Export dialog box for the printing. If the print area is offset to the right, a negative correction value must be entered. If the print area is offset to the left , a positive correction value must be entered.
   - Using a ruler, measure the vertical offset relative to the ready-made label sheet. This will be entered later in the "Vertical offset" field of the Export dialog box for the printing. If the print area is offset downward, a negative correction value must be entered. If the print area is offset upward, a positive correction value must be entered.

7.6 Undoing and redoing actions

7.6.1 Basics of undoing and redoing actions

Function

You can undo performed actions at any time. For this purpose, every action you perform is saved in an action stack. When undoing actions, the stack is processed from top to bottom. In

other words, if you undo an action that lies further down in the stack, all actions located above it in the stack will also be undone automatically.

You can redo previously undone actions until you execute a new action. Once you execute a new action, it is no longer possible to redo previously undone actions.

Particularities for undoing

There are a few actions that empty the action stack. You cannot undo these actions or the actions performed before these actions. The following actions empty the action stack:

- Saving
- Project management (creating a new project, opening project, closing a project, deleting a project)
- Compiling
- Restoring blocks
- Establishing an online connection
- Loading
- Writing to memory cards

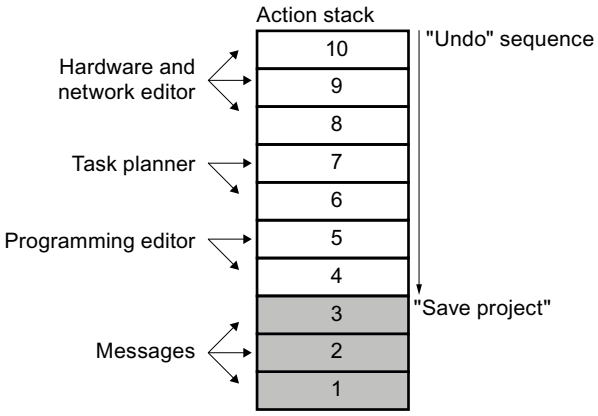
Displaying the action stack

The "Undo" button in the toolbar is enabled as soon as you perform an action that can be undone. This button is split; you can use the arrow down portion to open a drop-down list containing all actions of the action stack that you can undo. If you had performed actions in an editor other than the currently displayed editor, the corresponding editor is also displayed as a subheading. This allows you to always identify the point at which the undo operation will be applied. The subheadings are removed from the list when the editor responsible can no longer undo actions.

Actions you have undone are entered in the action stack from where they can be redone. Here, you can redo actions you have undone. The display of actions you can redo it is analogous to the display of the actions that you can undo.

Example of undoing actions

The figure below shows how actions performed in various editors and tables are undone:



In this example, you cannot undo actions 1 to 3 because the project was saved. You can undo actions 4 to 10 in the order indicated by the direction of the arrow. In other words, you must undo action 10 first. Once you have undone action 8, you cannot then undo action 5. You must first undo actions 7 and 6. As the final step in the sequence, you can then undo action 4. You also have the option of undoing several actions in a single step by undoing an action located further down in the action stack. All actions located above it in the stack will be undone automatically.

The same principle also applies to redoing of actions.

See also

Undoing an action (Page 404)

Redoing an action (Page 405)

7.6.2 Undoing an action

The following options are available for undoing actions:

- Undoing the last action only
  Only the last action performed is undone.
- Undoing as many actions as required
  Multiple actions in the action stack are undone in a single step.

Undoing the last action only

To undo the last action performed, follow the steps below:

1. Click the "Undo" button in the toolbar.
   - If the action was not performed in the currently displayed editor, a confirmation prompt appears.
   - If the undo operation requires an editor containing a protected object to be opened, you must enter the password for the object.
2. Click "Yes" to confirm.
3. Enter the password, if necessary.
   The editor in which the action was performed is displayed and the action is undone.

Undoing as many actions as required

To undo multiple actions in the action stack in a single step, follow these steps:

1. Click the Down arrow next to the "Undo" button in the toolbar.
   This opens a drop-down list containing all actions you can undo. Actions performed in other editors are identified by the editor name in the subheading.
2. Click the action you want to undo.
   The chosen action and all actions in the stack located above the chosen action are undone. If the undo operation requires an editor containing a protected object to be opened, you must enter the password for the object.
3. Enter the required passwords, if necessary.
   The editors in which the actions were performed are displayed and the actions are undone.

See also

Basics of undoing and redoing actions (Page 402)

Redoing an action (Page 405)

7.6.3 Redoing an action

You have the option of redoing an action that has been undone, so that you can return to the status present before "Undo" was performed. However, this is only possible until you perform a new action. The following options are available for redoing actions:

- Redoing the last undone action only
  Only the last undone action is redone.
- Redoing as many undone actions as required
  Multiple undone actions in the action stack are redone in a single step.

Redoing the last undone action only

To redo the last undone action, follow the steps below:

1. Click the "Redo" button in the toolbar.
   - If the action is not being redone in the currently displayed editor, a confirmation prompt appears.
   - If the redo operation requires an editor containing a protected object to be opened, you must enter the password for the object.
2. Click "Yes" to confirm.
3. Enter the password, if necessary.
   The editor in which the action was undone is displayed and the action is redone.

Redoing as many undone actions as required

To redo multiple undone actions in the action stack in a single step, follow these steps:

1. Click the Down arrow next to the "Redo" button in the toolbar.
   This opens a drop-down list containing all actions you can redo. Actions performed in other editors are identified by the editor name in the subheading.
2. Click the action you want to redo.
   The chosen action and all actions in the stack located above the chosen action are redone. If the redo operation requires an editor containing a protected object to be opened, you must enter the password for the object.
3. Enter the required passwords, if necessary.
   The editors in which the actions were undone are displayed and the actions are redone.

See also

Basics of undoing and redoing actions (Page 402)

Undoing an action (Page 404)

7.7 Finding and replacing in projects

7.7.1 Information on the search function

Find and replace

You can search for texts in the editors. The search function finds all texts containing the search key in the currently opened editor. The results are selected in sequence in the opened editor.

You also have the following options:

- Narrowing down the search with additional options
- Replacing found texts

The additional options and the type of texts for which you can search depend on the installed products and the currently open editor.

See also

Search and replace (Page 407)

7.7.2 Search and replace

Using Find

The "Find and replace" function enables you to search for or replace texts in an editor.

Additional options for searching

You can narrow down your search by selecting one of the following additional options:

- Whole words only
  Only whole words are found. Words that contain the search key as part of the word are ignored.
- Match case
  Upper- and lowercase letters are taken into account in the search.
- Find in substructures
  The search also includes texts contained in another object.
- Find in hidden texts
  Texts that are assigned to another text but that are currently hidden are also included in the search.
- Use wildcards
  Enter an asterisk as the wildcard for any number of characters. Example: You want to search for all words starting with "Device". Type in "Device*" in the search key box.
  Enter a question mark as the wildcard, however, if you only want to leave out a single character.
- Use regular expressions (for searching in scripts only)
  A regular expression is a character string used to describe sets of values and for filtering. This allows you to create complex search patterns.

The additional options available depend on the installed products and the editor opened.

Start search

Follow these steps to start the "Find and replace" function:

1. Select the "Find and replace" command in the "Edit" menu or open the "Find and replace" pane in the "Tasks" task card.
   The "Find and replace" pane opens.
2. Enter a term in the "Find" drop-down list.
   As an alternative, you can select the most recent search key from the drop-down list.
3. Select the options desired for the search.
4. Using the option buttons, select the starting point for the search and the search direction.
   - Select "Whole document" if you want to search through the entire editor regardless of the current selection,
   - Select "From current position" if you want to start the search at the current selection.
   - Select "Selection" if you only want to search within the current selection.
   - Select "Down" to search through the editor from top to bottom or from left to right.
   - Select "Up" to search through the editor from bottom to top or from right to left.
5. Click "Find".
   The first hit is marked in the editor.
6. Click "Find" again to display the next hit.
   The next hit is marked in the editor. Repeat this process, as necessary, until you reach the last hit.

Replacing the search key

You have the option of replacing hits individually or automatically replacing all the found texts, if the respective editor supports this function. Follow these steps to replace terms:

1. Enter a term in the "Find" drop-down list.
   As an alternative, you can select the most recent search key from the drop-down list.
2. Select the options desired for the search.
3. Click the "Find" button to start a search for the specified search key.
   The first hit is displayed in the editor.
4. In the "Replace" drop-down list, enter the text you wish to use to replace the search key.
   As an alternative, you can select the most recently text specified from the drop-down list.
5. Click the "Replace" button to replace the selected hit with the specified text.
   The found text is replaced and the next hit is marked in the editor.
   Repeat this process until you have replaced all the hits as wanted. To skip to the next hit without replacing the marked word, click the "Find" button instead of "Replace".
6. Click "Replace all" to automatically replace all hits at once.

See also

Information on the search function (Page 406)

7.8 Working with text lists

7.8.1 Text lists

Introduction

You can manage texts to be referenced in alarms centrally. All the texts are stored in text lists. Each text list has a unique name with which you can call up its content. A range of values is assigned to each text in a text list. If a value from a range of values occurs, the corresponding text is called up.

All the texts can be translated to all project languages. Here, you have two options available:

- You can enter the translation of the texts in a list. You will find the list in the project tree under "Languages & Resources > Project texts".
- You can export all texts to a file in Office Open XML format and enter the translation in a spreadsheet program. The translations can then be imported again.

The texts are translated into the other project languages within the framework of the project texts. In the text lists editor, you only have to manage the assignment of individual texts to a text list.

Each device in the project has its own text lists. For this reason, these lists are arranged under the devices in the project tree. In addition, there are text lists that apply to all devices. These can be found in the project tree under "Common data > Text lists".

User-defined and system-defined text lists

There are two types of text lists:

- User-defined text lists
  You can create user-defined text lists yourself and fill them with texts; in other words, you can specify value ranges and the corresponding texts yourself. With user-defined text lists, the name of the text list begins with "USER" as default. You can change this name to any suitable name.
- System-defined text lists
  System-defined text lists are created by the system. These always involve texts relating to devices. They are automatically created as soon as you insert a device in the project. With system alarms, the name of the text list begins with "SYSTEM". The name of the text list and the ranges of values it contains cannot be modified. You can only edit texts assigned to individual value ranges.

User-defined text lists	System-defined text lists
A user-defined text list can only be assigned to one device.	System-defined text lists can be assigned both to a device as well as to the entire project.
You can create new text lists and delete existing text lists.	You cannot create new text lists or delete text lists.

User-defined text lists	System-defined text lists
You can add and delete value ranges in the text lists.	You cannot add or delete value ranges in the text lists.
You can specify both the value ranges as well as the associated texts.	You can only edit the text associated with one value range.

Device-specific and cross-device text lists

Device-specific text lists relate to only one device in the project and are therefore only valid for this device. For this reason, they are arranged under a device in the project tree. Device-specific text lists can be used-defined or created by the system.

If system-defined text lists are generally valid for several devices or not intended uniquely for one device, these are grouped together in the project tree under "Common data". These text lists are available for all devices. Cross-device text lists are always created by the system and are used solely for system diagnostics alarms. For this reason, you cannot store any user-defined text lists under "Common data".

See also

Exporting and importing project texts (Page 345)

7.8.2 Creating user-defined text lists

Creating text lists

You can create user-defined text lists for individual devices.

Requirement

- You are in the project view.
- A project is open.
- The project includes a least one device.

Procedure

To create user-defined text lists, follow these steps:

1. Click on the arrow to the left of a device in the project tree.
   The elements arranged below the device are displayed.
2. Double-click on "Text lists".
   All the text lists assigned to the device are displayed in the work area listed in a table.
3. Double-click on the first free row in the table.
   A new user-defined text list is created.
4. Enter a name for your new text list in the "Name" column.

5. From the drop-down list in the "Selection" column, select whether you want to specify the value ranges in decimal, binary or in bits. Depending on the device, there may be further options available at this point.
6. Enter a comment in the "Comment" column.
   A new user-defined text list has been created and you can now enter the value ranges and texts.

7.8.3 Editing user-defined text lists

Editing user-defined text lists

You can enter value ranges and the corresponding texts in user-defined text lists. User-defined text lists are always located below a device in the project tree.

Requirement

- You are in the project view.
- A project is open.
- The project includes a least one device.

Adding to user-defined text lists with value ranges and texts

To add to user-defined text lists with value ranges and texts, follow these steps:

1. Click on the arrow to the left of a device in the project tree.
   The elements arranged below are displayed.
2. Double-click on "Text lists".
   All the text lists assigned to the device are displayed in the work area listed in a table.
3. Select a text list in the table.
   The contents of the selected text list are displayed in the work area. There, you can enter a value range and assign texts to the individual value ranges.
4. Enter the value ranges you require in the "Range from" and " Range to" columns. The entry must be made in the numeric format selected for the text list.
5. Enter a text for each value range in the "Entry" column.

7.8.4 Editing system-defined text lists

Editing system-defined text lists

In system-defined text lists, you can only modify the individual texts assigned to a value range.

System-defined text lists are located in the project tree either below a device or under "Common data".

Requirement

- You are in the project view.
- A project is open.
- The project includes a least one device.

Modifying texts in system-defined text lists

To edit texts in system-defined text lists that are assigned to a value range, follow these steps:

1. Click on the arrow to the left of a device in the project tree or the "Common data" element.
   The elements arranged below are displayed.
2. Double-click on "Text lists".
   All the text lists assigned to the device or used in common are displayed in the work area listed in a table.
3. Select a text list in the table.
   The contents of the selected text list are displayed in the work area. Here, you can add to or edit the texts assigned to a value range.
4. Enter a text for each value range in the "Entry" column.

7.9 Using memory cards

7.9.1 Basics about memory cards

Introduction

Memory cards are plug-in cards that come in a variety of types and can be used for a variety of purposes. Depending on the device type or device family, memory cards can be used for purposes, such as:

- Load memory of a CPU
- Storage medium for projects, firmware backups, or any other files
- Storage medium for performing a firmware update
- Storage medium for the PROFINET device name

For information regarding the technical variants of the respective memory cards and general information on their handling, refer to the respective documentation for the device. For

information on handling memory cards in the TIA Portal, refer to the online help under keyword "Memory Card".

NOTICE
Do not use memory cards for non-SIMATIC-related purposes, and do not use third-party devices or Windows tools to format them. This will irrevocably overwrite the internal structure of the memory card, rendering it unusable for SIMATIC devices!

See also

Adding a user-defined card reader (Page 413)

Accessing memory cards (Page 414)

Displaying properties of memory cards (Page 414)

7.9.2 Adding a user-defined card reader

Introduction

If your card reader is not detected automatically, you can add it manually.

Requirement

The project view is open.

Procedure

To add a card reader, follow these steps:

1. Open the project tree.
2. Select the "Card Reader / USB memory > Add user-defined Card Reader" command in the "Project" menu.
   The "Add user-defined Card Reader" dialog opens.
3. In the drop-down list box, select the path for the card reader.
4. Confirm your entries with "OK".

See also

Basics about memory cards (Page 412)

Accessing memory cards (Page 414)

Displaying properties of memory cards (Page 414)

7.9.3 Accessing memory cards

Requirement

- A memory card is inserted in the card reader.
- The project view is open.

Procedure

To access memory cards, follow these steps:

1. Open the project tree.
2. Select the "Card Reader / USB memory > Card Reader / Show USB memory" command in the "Project" menu.
   The "Card reader / USB memory" folder is displayed in the project tree.
3. Open the "Card Reader / USB memory" folder.
   You can now access the memory card.

Note

If data from a non-installed product is stored on the memory card, the folders that contain these data are shown in gray. You receive an error message when you attempt to access such a folder. Install the corresponding product if needed.

See also

Basics about memory cards (Page 412)

Adding a user-defined card reader (Page 413)

Displaying properties of memory cards (Page 414)

7.9.4 Displaying properties of memory cards

You can display the properties for the utilized memory cards. Note that different memory cards with different properties must be used, depending on the device.

Requirement

- A memory card is inserted in the card reader.
- The project view is open.

Procedure

To display the properties of a memory card, follow these steps:

1. Right-click on the memory card for which you want to display the properties.
2. Select the "Properties" command in the shortcut menu.
   The "Memory Card <name of the memory card>" dialog opens. The properties are displayed in this dialog.

See also

Basics about memory cards (Page 412)

Adding a user-defined card reader (Page 413)

Accessing memory cards (Page 414)

7.10 Using libraries

7.10.1 Library basics

Introduction

You can store objects you want to reuse in libraries. For each project there is a project library that is connected to the project. In addition to the project library, you can create any number of global libraries that can be used over several projects. Since the libraries are compatible with each other, library elements can be copied and moved from one library to another. Libraries are used, for example, to create templates for blocks that you first paste into the project library and then further develop there. Finally, you copy the blocks from the project library to a global library. You make the global libraries available to other employees working on your project. They use the blocks and further adapt them to their individual requirements, where necessary.

Both the project library and global libraries distinguish between two different types of objects:

- Master copies
  Almost every object can be saved as a master copy and pasted into the project again later. You can, for example, save entire devices with their contents or cover sheets for plant documentation as master copies.
- Types
  Elements that are required to run user programs, for example, blocks, PLC data types, user-defined data types or faceplates are suitable as types. Types can be versioned and therefore support professional further development. Projects using the types can be updated as soon as new versions of the types are available.

Project library

Each project has its own library, the project library. Here, you store the objects you want to use more than once in the project. This project library is always opened, saved, and closed together with the current project.

Global libraries

In addition to the project library, you use global libraries if you want to use libraries over several projects. Global libraries exist in three versions:

- System libraries
  Siemens supplies global libraries for its own software products. These include off-the-peg functions and function blocks that you can use within your project. The supplied libraries cannot be changed. The supplied libraries are loaded automatically matching the project. If you are working in V12 SP1 compatibility mode, the matching libraries for product version V12 SP1 are loaded. For all other projects the supplied libraries of the latest TIA Portal version are loaded.
- Corporate libraries
  Corporate libraries are made available centrally by your organization, for example, in a central folder on a network drive. The TIA Portal manages the corporate libraries automatically. As soon as a more recent version of an existing corporate library becomes available, you receive a prompt to update the corresponding corporate library to the more recent version.
- User libraries
  Global user libraries are independent of a specific project and can therefore be passed on to other users. Shared access to global user libraries is also possible, for example on a network drive, if all users open the global user library with write protection.
  Global user libraries you have created yourself from older versions of the TIA Portal can still be used. To continue using global user libraries from older versions of the TIA Portal, they must first be upgraded.

Comparing library objects

You can compare blocks and PLC data types with the objects of a device. This allows you to determine, for example, whether certain blocks or PLC data types have been used in a project and whether they have been modified.

See also

Overview of the "Libraries" task card (Page 417)

Overview of the library view (Page 420)

Overview of the library management (Page 423)

Basics on master copies (Page 441)

Basics on types (Page 444)

7.10.2 Using the "Libraries" task card

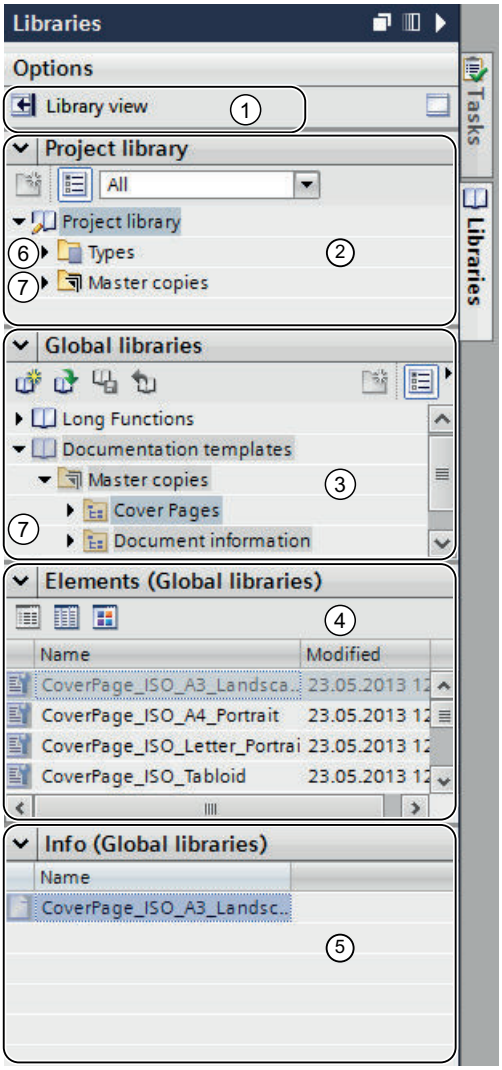
7.10.2.1 Overview of the "Libraries" task card

Function of the "Libraries" task card

The "Libraries" task card enables you to work efficiently with the project library and the global libraries.

Layout of the "Libraries" task card

The "Libraries" task card consists of the following components:



① "Library view" button
② "Project library" pane
③ "Global libraries" pane
④ "Elements" pane
⑤ "Info" pane
⑥ "Types" folder
⑦ "Master copies" folder

"Library view" button

The "Library view" button is used to change to the library view. The "Libraries" task card and the project tree are hidden with this action.

See also: Using the library view (Page 420)

"Project library" pane

In the "Project library" pane, you can store the objects that you want to use more than once in the project.

"Global libraries" pane

In the "Global libraries" pane, you manage the global libraries whose elements you want to reuse over several projects.

The "Global libraries" pane also lists libraries that were shipped together with the products you purchased. These libraries provide you with ready-made functions and function blocks, for example. The supplied global libraries cannot be edited.

"Elements" pane

In this pane, you can display the contents of folders in the library. The "Elements" pane is not displayed by default. If you want to display the "Elements" pane, you have to enable it first. Three view modes are available in the "Elements" palette:

- Details mode
  The properties of folders, master copies and types are shown in table form in details mode.
- List mode
  In list mode, the contents of folders are listed.
- Overview mode
  In overview mode, the contents of folders are displayed with large symbols.

See also: Using the element view (Page 419)

"Info" pane

You can display the contents of the library elements in the "Info" pane. The individual versions of types and the last revision date of the version are also displayed.

"Types" folder

In the "Types" directories, you can manage types and type versions of objects that you use as instances in the project.

See also: Using types (Page 444)

"Master copies" folder

In the "Master copies" directories, you can manage master copies of objects that you can use as copies in the project.

See also: Using master copies (Page 441)

See also

Library basics (Page 415)

Comparing library elements (Page 471)

7.10.2.2 Using the element view

Introduction

When you open the "Libraries" task card the first time, the "Project library" and "Global libraries" palettes are opened and the "Info" palette is closed. You can display the "Elements" palette if needed.

The elements view shows the elements of the selected library. Three view modes are available in the elements view:

- Details
  The properties of folders, master copies and types are shown in table form in details mode.
- List
  In list mode, the contents of folders are listed.
- Overview
  In overview mode, the contents of folders are displayed with large symbols.

The "Info" palette shows the contents of the selected library element. If you select a type in the elements view, for example, the type versions are displayed in the "Info" palette.

Requirement

The "Libraries" task card is displayed.

Procedure

To use the element view, follow these steps:

1. Click "Open or close the element view" in the "Project library" or "Global libraries" pane.
2. To change the view mode from the details view to the list mode or overview mode, click the corresponding icon on the toolbar.

See also

Library basics (Page 415)

Overview of the "Libraries" task card (Page 417)

Using global libraries (Page 428)

Comparing library elements (Page 471)

7.10.3 Using the library view

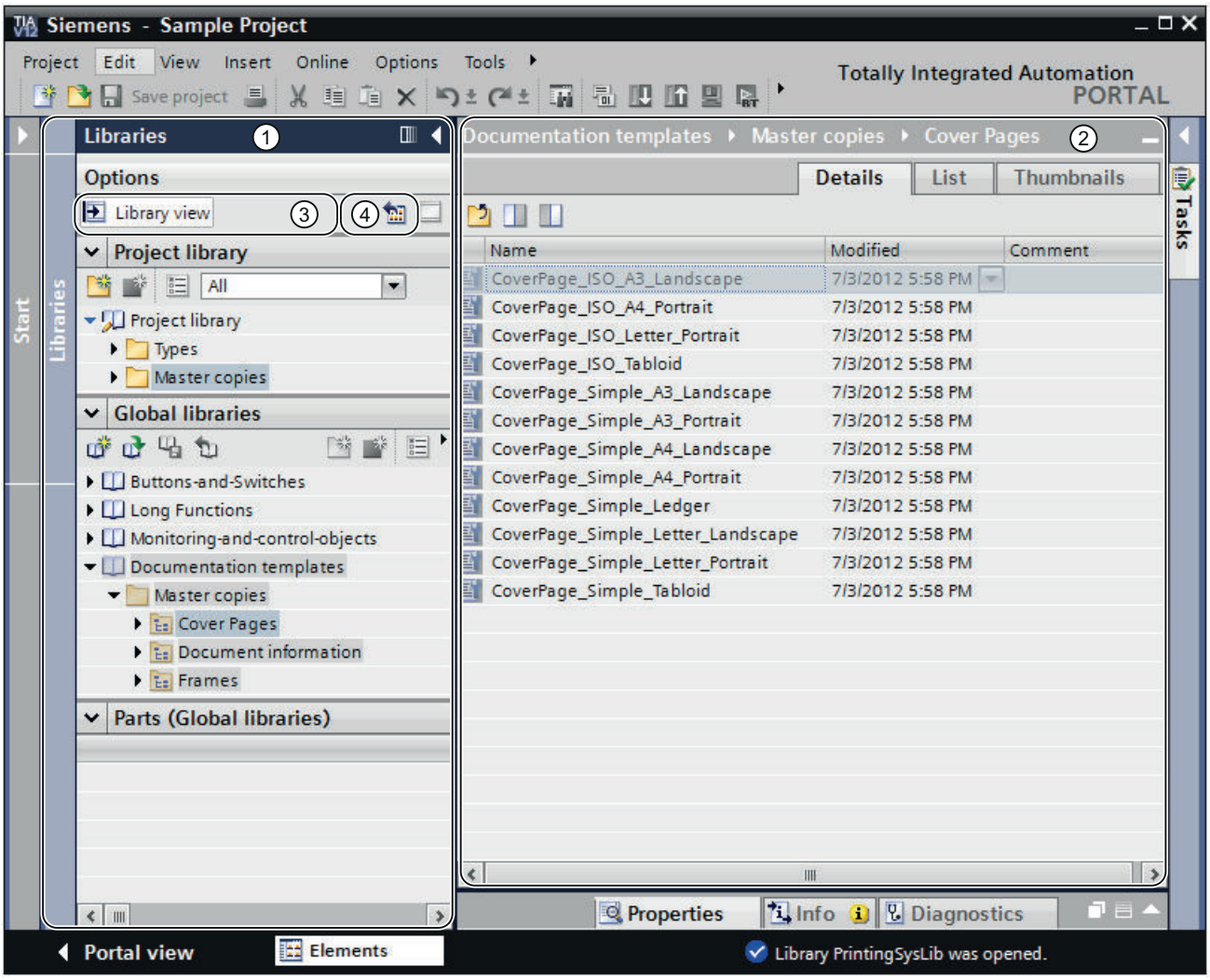
7.10.3.1 Overview of the library view

Function of the library view

The library view combines the functionality of the "Libraries" task card and the overview window. In the library view, the elements of a library are displayed in various views. In the details view, for example, you see additional properties of the individual elements. You can also edit and version the types in the library view.

Layout of the library view

The following figure shows the components of the library view:



① Library tree
② Library overview
③ "Library view" button
④ "Open or close library overview" button

Library tree

The library tree is similar to the "Libraries" task card, apart from a few minor differences. In contrast to the task card, there is no "Elements" palette, because the elements are displayed in the library overview. In addition, you can close the library view in the library tree, or open and close the library overview.

See also: "Libraries" task card (Page 417)

Library overview

The library overview corresponds to the overview window and displays the elements of the currently selected object in the library tree. You can display the elements in three different views:

- Details view
  The objects are displayed in a list with additional information, such as the date of the last change.
- List view
  The objects are displayed in a simple list.
- Icon view
  The objects are displayed as icons according to category.

In addition, you can perform the following actions in the library overview:

- Renaming elements
- Deleting elements
- Copying elements
- Moving elements
- Editing type instances
- Versioning types
- WinCC only: Editing faceplates and HMI user data types

See also: Overview window (Page 300)

See also

Basics on master copies (Page 441)

Basics on types (Page 444)

Opening and closing the library view (Page 422)

Library basics (Page 415)

Comparing library elements (Page 471)

7.10.3.2 Opening and closing the library view

The library view is opened automatically in some cases, for example, when you edit the test instance of a type or when you edit faceplates and HMI user data types. You can also open the library view manually.

Opening the library view

To open the library view manually, follow these steps:

1. Open the "Libraries" task card.
2. Click the "Open library view" button in the "Libraries" task card.
   The library tree opens. The "Library" task card and the project tree are closed.
3. If the library overview is not displayed, click "Open/close library overview" button in the library tree.
   The library overview opens.

Exit library view

To exit the library view, follow these steps:

1. Click the "Close library view" button in the library tree.
   The library tree closes. The "Libraries" task card and the project tree are opened.

See also

Overview of the library view (Page 420)

Library basics (Page 415)

Using the "Libraries" task card (Page 417)

Using global libraries (Page 428)

Comparing library elements (Page 471)

7.10.4 Using library management

7.10.4.1 Overview of the library management

Function of the library management

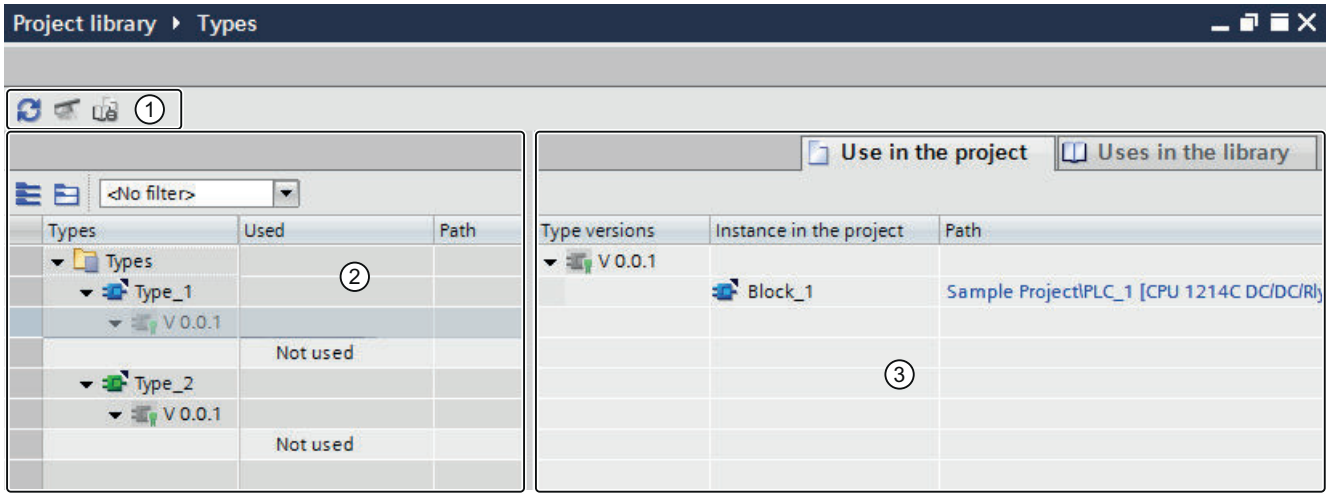
Master copies and types with dependencies to other library elements are subject to some functional restrictions. They can, for example, not be deleted as long as dependencies still exist. This prevents other library elements from becoming useless. The library management is used to identify the dependencies and to create an overview of the work progress.

The library management offers the following functions:

- Display of the correlations of types and master copies
  If a type is referenced in other types or master copies, the correlations are displayed in the library management. You will also be able to see which library elements reference a type or a master copy.
- Display of points of use of types in the project
- Use of filters to narrow down the displayed types

Layout of the library management

The following figure shows the components of the library management:



① Toolbar of the library management

② "Types" area

③ "Uses" area

Toolbar of the library management

You can perform the following tasks in the toolbar of the library management:

- Update view
  If the project was changed, you can update the view of the library management.
- Clean up library
  You can clean up the project library and global libraries. By cleaning up a library, you delete all types and type versions that are not linked to an instance in the project.
- Harmonize project
  By harmonizing a project, you adapt the names and the path structures of type uses in the project to the corresponding names and path structures of the types within a library.

"Types" area

The "Types" area displays the contents of the folder that you have selected in the library view. For each type, the types that it references are displayed. You can expand or collapse all type nodes by using the buttons in the toolbar of the "Types" area. You can also filter the view with the "Filter" drop-down list.

"Uses" area

The "Uses" area gives you an overview of the points of use of the selected types and master copies. The "Uses" area is divided into two tabs:

- "Uses in the project" tab
  The "Uses in the project" tab is used to show the instances of type versions and their respective point of use in the project. When you select an instance, you can show the cross references of the instance in the project in the Inspector window.
- "Uses in the library" tab
  The "Uses in the library" tab is used to show all points within the library at which a type or a master copy is used.

See also

Opening library management (Page 425)

Filtering the display of types (Page 426)

Displaying cross references of an instance (Page 427)

Displaying instances in the project (Page 426)

Displaying relations to other library objects (Page 427)

Library basics (Page 415)

Basics on master copies (Page 441)

Basics on types (Page 444)

7.10.4.2 Opening library management

Procedure

To open the library management, follow these steps:

1. Open the library view.
2. Select a type or any folder that contains types.
3. Select the "Library management" command from the shortcut menu.

Result

The library management opens and the types are displayed with their versions.

See also

Overview of the library management (Page 423)

7.10.4.3 Filtering the display of types

Use filters to obtain a better overview of types in extensive libraries. A filter provides you with the option of limiting the displayed types. The following filters are available:

- Display all types that include a version with the "In test" or "In progress" status
- Display all released types
- Display all types that have no instances in the project
- Display all types that have more than one version

Requirement

The library management is open.

Procedure

To filter the displayed types, follow these steps:

1. In the "Types" area, select the folder whose contents you want to filter.
2. Select the required filter in the "Filter" drop-down list.
   In the "Types" area, types are displayed that correspond to the selected filter criteria.

See also

Overview of the library management (Page 423)

State of type versions (Page 446)

Creating a test version of a type (Page 451)

Editing a test version of a type (Page 452)

Creating an editing version of a type (Page 453)

7.10.4.4 Displaying instances in the project

In the library management, you can have the instances of all versions of a type or an individual type version displayed. You can jump directly to each instance in the project.

Requirement

The library management is open.

Procedure

To display the instances of a type or its versions, follow these steps:

1. Select a type or one of its versions in the "Types" area.
2. Open the "Uses in the project" tab in the "Uses" area.
   The instances in the project are displayed for each type version. The "Path" column shows the path at which the respective instance is located in the project.
3. Optional: Click the path to jump directly to the respective instance in the project tree.
   The library management is hidden and the instance is selected in the project tree.

See also

Displaying cross references of an instance (Page 427)

Using types (Page 450)

Overview of the library management (Page 423)

Displaying relations to other library objects (Page 427)

7.10.4.5 Displaying cross references of an instance

You can display the cross references of an instance without exiting the library management.

Requirement

The library management is open.

Procedure

To display the cross references of an instance in the project, follow these steps:

1. In the "Types" area, select a type version whose instances you want to display.
2. Select the instance of the required type version in the "Uses > Uses in the project" area.
3. Open the "Info > Cross-references" tab in the inspector window.
   The cross reference of the instance are displayed in the project.

See also

Using cross-references (Page 472)

Overview of the library management (Page 423)

Displaying instances in the project (Page 426)

7.10.4.6 Displaying relations to other library objects

You can display relations between individual library objects in the library management. The references of the individual type versions to the other library objects are automatically

displayed in the "Types" area. In the "Uses" area, you can also view the other library objects in which the respective type version is referenced.

Requirement

The library management is open.

Procedure

To display the other library objects from which a type version is referenced, follow these steps:

1. Select a folder, an individual type or an individual version in the "Types" area.
2. Open the "Uses in the library" tab in the "Uses" area.
   In the "Uses" area, you can now see which other library objects are referenced by the individual type versions.
3. Optional: To jump to the referenced library object, click on the respective path in the "Path" column.

See also

Displaying instances in the project (Page 426)

Overview of the library management (Page 423)

7.10.5 Using global libraries

7.10.5.1 Creating a global library

Requirement

The "Libraries" task card is displayed or the library view is open.

Procedure

To create a new global library, follow these steps:

1. Click the "Create new global library" icon in the toolbar of the "Global libraries" palette or select the command "Global libraries > Create new library" in the "Options" menu.
   The "Create new global library" dialog opens.
2. Specify the name and the storage location for the new global library.
3. Confirm your entries with "Create".

Result

The new global library is generated and pasted into the "Global libraries" palette. A folder with the name of the global library is created in the file system at the storage location of the global library. This actual library file is given the file name extension ".al12".

See also

Library basics (Page 415)

Opening a global library (Page 430)

Displaying properties of global libraries (Page 433)

Saving a global library (Page 434)

Closing a global library (Page 435)

Deleting a global library (Page 436)

7.10.5.2 Compatibility of global libraries

You can use the TIA Portal to open global libraries that were created with an older version of the TIA Portal. However, the product version of global libraries must correspond to the product version of the opened project if you want to use objects from the global library in the project. Consequently, you may need to upgrade the global libraries.

Opening global libraries from older product versions

The table below illustrates the behavior of the TIA Portal when global libraries from older product versions are opened:

Product version of the TIA Portal (file extension of the respective global library)	Behavior when a global library is opened
V10.5 (.al10) V11.x (.al11)	The global library is automatically upgraded to the latest library version when opened, following your confirmation. The upgraded global library is a copy of the original library. The original global library is retained unchanged.
V12 (.al12)	The global library is upgraded when opened. You can choose whether you want to upgrade the global library to the library version V12 SP1 or V13. In both cases, the upgraded library is a copy of the original global library. The original global library is retained unchanged. When you upgrade the global library to library version V12 SP1, the library is opened in compatibility mode.
V12 SP1 (.al12)	When you open the global library, you can decide whether you want to open it in compatibility mode or upgrade it to the current library version. If you upgrade the global library, a copy of the original global library is created. The original global library is retained unchanged.

Compatibility mode

If a project is open in compatibility mode for product version V12 SP1 of the TIA Portal, the global library must also be in compatibility mode. You can use objects from the global library

in the project opened in compatibility mode. You can also save new objects in the global library if the objects originate from a project in compatibility mode. A global library in compatibility mode can still be opened and edited with TIA Portal V12 SP1.

Backward compatibility of the current library version

Global libraries saved with the library format of the TIA Portal V13 are not backward compatible with older versions due to their enhanced functionality. Global libraries in TIA Portal V13 library format can only be used in connection with TIA Portal V13 projects.

See also

Compatibility of projects (Page 333)

Opening a global library (Page 430)

Upgrading global libraries (Page 431)

Upgrading projects (Page 335)

7.10.5.3 Opening a global library

Global libraries can be further developed centrally and used over several projects. Several persons can open a global library simultaneously from a central storage location, provided all users open the global library in write-protected mode.

Requirement

The "Libraries" task card is displayed or the library view is open.

Procedure

To open a global library, follow these steps:

1. Click the "Open global library" icon in the toolbar of the "Global libraries" pane or select the command "Global libraries > Open library" in the "Options" menu.
   The "Open global library" dialog box is displayed.
2. Select the global library you want to open. Library files are identified by the file name extension ".al[version number]". This means global libraries that were saved with TIA Portal V13 have the file name extension ".al13".
3. Write protection is activated for the library. If you want to modify the global library, disable the "Open as read-only".
4. Click "Open".
   The global library is opened and pasted into the "Global libraries" pane if the library version matches the project version. The "Upgrade global library" dialog box opens if you have selected a global library from an older version of the TIA Portal. More information on upgrading the global library can be found in the chapter "Upgrading global libraries (Page 431)".

See also

Retrieving global libraries (Page 438)
Upgrading global libraries (Page 431)
Opening projects (Page 334)
Compatibility of global libraries (Page 429)
Library basics (Page 415)
Creating a global library (Page 428)
Displaying properties of global libraries (Page 433)
Saving a global library (Page 434)
Closing a global library (Page 435)
Deleting a global library (Page 436)

7.10.5.4 Upgrading global libraries

If you want to use objects from a global library in a project, the library version must match the product version of the project. Global libraries must be present in the library version of the TIA Portal V12 SP1 if you want to edit a project in V12 SP1 compatibility mode. Global libraries must be present in the library version of the TIA Portal V13 if you want to edit a project of the product version V13.

Depending on the library version, the following options are available:

- Upgrading the global library to the product version V13
  You can upgrade each global library from an older product version to the current product version. You are prompted accordingly when you open the global library. You can manually upgrade global libraries projects that are already open in compatibility mode to product version V13.
- Upgrading the global library to the product version V12 SP1 (compatibility mode)
  If you open a global library from version V12 of the TIA Portal, you also have the option of upgrading the global library to the library version V12 SP1. In this case, however, the global library can only be used in connection with projects from the TIA Portal V12 SP1.

Upgrading global libraries from V11.x or lower

Proceed as follows to upgrade a global library from TIA Portal V11.x or lower:

1. Open the global library.
   The "Upgrade global library" dialog box opens.
2. Click "OK".
   A copy of the global library is created and upgraded. The copy of the global library receives the name extension "_V13". The global library is opened.

Upgrading global libraries from V12

Proceed as follows to upgrade a global library from TIA Portal V12:

1. Open the global library.
   The "Upgrade global library" dialog box opens.
2. Select the target version for the upgrade:
   - Click "Yes" to upgrade the global library to product version V13.
   - Click "No" to upgrade the global library to product version V12 SP1 and to work in compatibility mode.

   A copy of the global library is created and upgraded. The copy of the global library receives the name extension "_V13" or "_V12SP1". The global library is opened.

Upgrading V12 SP1 libraries or using them in compatibility mode

Proceed as follows to upgrade a global library from TIA Portal V12 SP1 or to use it in compatibility mode:

1. Open the global library.
   The "Upgrade global library" dialog box opens.
2. Select how you want to proceed with the global library:
   - Click "Yes" to upgrade the global library to product version V13.
   - Click "No" to use the global library in compatibility mode.

   A copy of the global library is created and upgraded or the library is opened changed. The copy of the global library receives the name extension "_V13".

Manually upgrading libraries in compatibility mode to V13

To do this, a global library must already be open in compatibility mode for the TIA Portal V12 SP1.

Proceed as follows to manually upgrade global libraries to the library version V13:

1. Right-click on the global library you want to upgrade.
2. Select the "Upgrade library" command in the shortcut menu.
   The "Upgrade" dialog opens.
3. Confirm with "Yes".
   A copy of the global library is created and upgraded. The copy of the global library receives the name extension "_V13". The upgraded global library is opened. The original global library is closed.

See also

Opening a global library (Page 430)

Compatibility of global libraries (Page 429)

Upgrading projects (Page 335)

7.10.5.5 Displaying properties of global libraries

Global libraries contain properties for describing the respective library in more detail. Properties include the following:

- General information about the library
  This includes the following information: creation time, author, file path, file size, copyright, etc. Many of the attributes can be changed.
- Library history
  The library history contains an overview of the migrations performed. Here you can also call the log file for the migrations. The library history also contains information on updates of the global library.
- Support packages in the library
  You can display an overview of the additional software needed to work with all devices in the project.
- Software products in the library
  You can display an overview of all installed software products needed for the project.

Requirement

The "Libraries" task card is displayed or the library view is open.

Procedure

To display the properties of a global library, follow these steps:

1. Right-click the global library whose properties you want to display.
2. Select the "Properties" command in the shortcut menu.
   A dialog containing the properties of the global libraries opens.
3. Select the properties in the area navigation that you want to have displayed.

See also

Opening a global library (Page 430)

Library basics (Page 415)

Creating a global library (Page 428)

Saving a global library (Page 434)

Closing a global library (Page 435)

Deleting a global library (Page 436)

7.10.5.6 Displaying logs of global libraries

Logs are created when you update or clean up global libraries or assign a shared version to several types. The logs list all changes you have made to the global library. The logs are stored

together with the global library and are always available once you have opened the global library.

Procedure

To open the logs of a global library, follow these steps:

1. Open the global library in the "Libraries" task card or in the library view.
2. Open "Common data > Logs" in the lower-level folder.
3. Double-click the required log.
   The log opens in the work area.

See also

Updating a library with the contents of another library (Page 467)

7.10.5.7 Saving a global library

After you have changed a global library, you need to save it. You can save a global library under another name using the "Save library as" command.

Note

Backward compatibility to older versions of the TIA Portal

Note that global libraries can no longer be opened in older versions of the TIA Portal once they have been saved in the newer version.

Requirement

The "Libraries" task card is displayed or the library view is open.

Saving changes

To save a global library, follow these steps:

1. Right-click on the global library you want to save.
2. Select the "Save library" command in the shortcut menu.

Saving a global library under another name

To save a global library under another name, follow these steps:

1. Right-click the global library that you want to save under a different name.
2. Select the "Save library as" command in the shortcut menu.
   The "Save global library as" dialog opens.

3. Select the storage location and enter the file name.
4. Confirm your entries with "Save".
   The library is saved in the specified location under the new name. The original library is retained.

See also

Working with archives of global libraries (Page 436)

Archiving global libraries (Page 437)

Library basics (Page 415)

Creating a global library (Page 428)

Opening a global library (Page 430)

Displaying properties of global libraries (Page 433)

Closing a global library (Page 435)

Deleting a global library (Page 436)

7.10.5.8 Closing a global library

Global libraries are independent of projects. This means that global libraries are not closed together with your project. You must therefore close global libraries explicitly.

Requirement

The "Libraries" task card is displayed or the library view is open.

Procedure

To close a global library, follow these steps:

1. Right-click on the global library you want to close.
2. Select the "Close library" command in the shortcut menu.
   If you have made changes to the global library, select whether or not you want to save the changes.
   The global library is closed.

See also

Creating a global library (Page 428)

Opening a global library (Page 430)

Displaying properties of global libraries (Page 433)

Saving a global library (Page 434)

Library basics (Page 415)

Deleting a global library (Page 436)

7.10.5.9 Deleting a global library

If you no longer require a global library, you can delete it. Libraries supplied by Siemens cannot be deleted.

Requirement

The "Libraries" task card is displayed or the library view is open.

Procedure

To delete a global library, follow these steps:

1. Right-click the global library you want to delete.
2. Select the "Delete" command in the shortcut menu.
3. Click "Yes" to confirm.

Result

The global library is removed from the "Global libraries" palette. The entire library for the global library is deleted from the file system.

See also

Library basics (Page 415)

Creating a global library (Page 428)

Opening a global library (Page 430)

Displaying properties of global libraries (Page 433)

Saving a global library (Page 434)

Closing a global library (Page 435)

7.10.5.10 Archiving and disabling global libraries

Working with archives of global libraries

If you want to back up global libraries on an external hard drive or send them via e-mail, for example, use the archiving function to reduce the storage space of the library.

Options for reducing the size of the project

There are two ways to reduce the storage space of global libraries:

- Creating a compressed archive of global libraries
  Archives of global libraries are compressed files, each containing an entire global library, including the entire folder structure of the library. Before the directory with the global library is compressed into the archive file, all files are reduced to their essential components to further decrease the storage space. Compressed archives of global libraries are therefore well suited for sending via e-mail.
  Compressed archives of global libraries of TIA Portal V13 have the file extension ".zal13". Archives from TIA Portal V12.x have the file extension ".zal12".
  To open a compressed archive of a global library, retrieve the archive. The archive file is extracted to a location you have selected with the entire folder structure and all files.
- Minimizing global libraries
  You can skip additional compression in an archive file, and instead create a copy of the global library directory. The included files can be reduced to the essential elements. This minimizes the required storage space. The full functionality of the global library is retained and the global library can be loaded as usual.
  Minimized global libraries are especially well suited for archiving, for example, on an external medium.

See also

Archiving global libraries (Page 437)

Retrieving global libraries (Page 438)

Archiving global libraries

You reduce the storage space of a global library by packing it into a compressed file. You can also reduce the storage space by saving the global library reduced to the essential elements. You can do both of these with the archiving function for global libraries. When you archive a library, the original library version is retained. This means libraries in compatibility mode for TIA Portal V12 SP1 are still compatible with TIA Portal V12 SP1.

Requirement

The global library is loaded.

Procedure

To archive a global library, follow these steps:

1. Select the global library that you want to archive.
2. Right-click the global library and select the "Archive" command from the shortcut menu.
   The "Archive global library as..." dialog opens.
3. Select the directory where you want to save the archive file or the new directory of the global library.
   The directory may not be located in a project directory or within the directory of a global library.

4. Select the file type from the "File type" drop-down list:
   - Global libraries archive, if you want to create a compressed file of the library.
   - Minimized global library, if you only want to create a copy of the library directory with minimal storage requirement.
5. Enter a file name in the "File name" field if you are creating an archive file. If you are creating a minimized global library, enter the name of the new library directory to be created in the "File name" box.
6. Click "Save".

Result

A compressed file with the extension ".zal13" or ".zal12" is generated. The file extension depends on whether you have archived a library in V12 SP1 compatibility mode or in the library format of TIA Portal V13

The file includes the complete directory of the global library. The individual files of the global library are also reduced to the essential components in order to save space.

If you have minimized the global library, only a copy of the original directory of the global library is created at the required location. The files contained within it were reduced to their essential components in order to save space.

See also

Working with archives of global libraries (Page 436)

Retrieving global libraries (Page 438)

Retrieving global libraries

Before you can use an archived global library, you have to retrieve it. The global library is extracted and then opened in the TIA Portal.

Procedure

To extract the archive of a global library, follow these steps:

1. Select the "Global libraries > Retrieve library" command in the "Options" menu.
   The "Retrieve archived global library" dialog opens.
2. Select the archive file.
3. Select the check box "Open read-only" if you want to load the global library write-protected.
4. Click "Open".
5. The "Find folder" dialog opens.
6. Select the target directory to which the archived global library should be extracted.
7. Click "OK".

Result

The global library is extracted to the selected directory and opened immediately.

If the archive file contained a global library from TIA Portal V12 SP1, the global library is opened in compatibility mode. Manually upgrade the library to the library version of TIA Portal V13, if necessary.

See also

Working with archives of global libraries (Page 436)

Archiving global libraries (Page 437)

Opening a global library (Page 430)

Compatibility of global libraries (Page 429)

Upgrading global libraries (Page 431)

7.10.5.11 Using global corporate libraries

Basics of corporate libraries

Introduction

Corporate libraries are global libraries made available by an administrator and assigned to the TIA Portal. The administrator can assign new libraries or change libraries at any time. New libraries are automatically uploaded in the TIA Portal following your confirmation. As soon as more recent versions of corporate libraries are available, the existing corporate libraries are automatically updated to the more recent version, also following your confirmation.

The corporate libraries are located just like normal global libraries in the "Global libraries" palette of the "Libraries" task card.

Providing corporate libraries

Corporate libraries can be saved in any directory on the hard drive of the computer or on a network drive. Use an XML file to assign corporate libraries to the TIA Portal. The XML file includes the directories and names of the assigned corporate libraries. The XML file must be saved in the following directory on the hard drive of the computer:

`C:\ProgramData\Siemens\Automation\Portal V13\CorporateSettings\`

The XML file must be named "`CorporateSettings.xml`".

You either copy the configuration file yourself to the corresponding directory or the configuration file is assigned to you through the company network. The valid configuration is automatically applied when the TIA Portal is started. When the TIA Portal is started, it continuously monitors the directory for configuration files. If the configuration file has changed, you receive a prompt to apply the new configuration. You can reject this prompt twice. You will always receive the next prompt three hours later. You must apply the new configuration at the third prompt. You receive a changed configuration file from the project administrator, for example, if corporate libraries have been added or deleted.

Options as project administrator

You can automatically assign the configuration file or the corporate libraries to the computers of the team members or distribute updates to the team members. This function is not part of the TIA Portal and requires a corresponding IT infrastructure in your company. If you want to administer the configuration file centrally, discuss this approach with the IT managers at your company.

Creating a configuration file

Use a configuration file in XML format to make the corporate libraries available in the TIA Portal. The configuration file includes the directories and file names of the libraries to be loaded. Below you will learn how you create the XML configuration file and where to save it.

Procedure

To provide a configuration file for corporate libraries, follow these steps:

1. Create an XML configuration file with the content listed below. Use the coding "UTF-8".
2. Save the XML file under the name "CorporateSettings.xml".
3. Save the file in the following directory on your computer:
   C:\ProgramData\Siemens\Automation\Portal V13\CorporateSettings\

Content of the XML configuration file

The XML configuration file must have the following content:

XML

```
<?xml version="1.0" encoding="utf-8"?>
<Document>
 <Settings.Settings ID="0">
  <ObjectList>
   <Settings.General ID="1" AggregationName="General">
    <AttributeList>
     <CorporateLibraryPaths>
     <!-- Example of an entry -->
     <Item>D:\CorporateLibraries\Corporate_Library_1.al13</Item>
     <!-- Here you enter additional global libraries, if any. -->
     </CorporateLibraryPaths>
    </AttributeList>
   </Settings.General>
  </ObjectList>
 </Settings.Settings>
</Document>
```

Result

As soon as you have saved the XML configuration file in the respective directory, you will receive the prompt (in the TIA portal) to download the corporate libraries in the TIA Portal.

7.10.6 Creating folders in a library

The library elements are stored in the libraries according to their type in the "Types" and "Master copies" folders. Create additional folders below "Types" and "Master copies" to further organize master copies and types.

Requirements

- The "Libraries" task card is displayed or the library view is open.
- If you want to create new folders within a global library, you have to open the global library with write permission.

Procedure

To create a new folder, follow these steps:

1. Right-click any folder within the library.
2. Select "Add folder" from the shortcut menu.
   A new folder is created.
3. Enter a name for the new folder.

See also

Working with types in the project library (Page 449)

Filtering master copies (Page 443)

7.10.7 Using master copies

7.10.7.1 Basics on master copies

Master copies are used to create standardized copies of frequently used elements. You can create as many elements as needed and insert them into the project based on a master copy. The elements inherit the properties of the master copy.

You store master copies either in the project library or in a global library. Master copies in the project library can only be used within the project. When you create the master copy in a global library, it can be used in different projects.

The following elements can be created as master copies in the library, for example:

- Devices with their device configuration
- Tag tables
- Instruction profiles
- Monitoring tables
- Elements from the documentation settings, for example, cover sheets and frames

- Blocks and groups containing multiple blocks
- PLC data types and groups containing multiple PLC data types
- Text lists
- Alarm classes
- Technology objects

In many cases, the objects you add as master copy contain additional elements. A CPU, for example, can contain blocks. If the included elements are uses of a type version, the used versions of the types are automatically created in the library. The elements contained therein are then used as an instance and linked to the type.

See also

Adding master copies (Page 442)

Using master copies (Page 444)

Basics on types (Page 444)

Filtering master copies (Page 443)

7.10.7.2 Adding master copies

If you want to use objects multiple times, copy them as master copies in the project library or in a global library. You can choose from the following methods for creating master copies:

- You can select one or more elements and generate a master copy from it
- You can select multiple elements and generate a master copy from each element

Requirement

- The "Libraries" task card is displayed.
- If you add a device as a master copy, this device meets the following requirements:
  - The device is compiled and consistent.
  - The device contains no test instance of a type.
- If you add the master copy to a global library, the global library is opened with write permission.

Creating a master copy from one or more elements

To create a master copy for one or more elements, follow these steps:

1. Open the library in the "Libraries" task card.
2. Drag one or more elements and drop them into the "Master copies" folder or any subfolder of "Master copies".
   Alternative: Copy the elements to the clipboard and paste them at the required location.
   The element is pasted into the library as a single master copy.

Creating a master copy for each element among multiple elements

To create a master copy for each element among multiple elements, follow these steps:

1. Open the library in the "Libraries" task card.
2. Copy to the clipboard the elements that you want to create as master copies.
3. Right-click on the "Master copies" folder or any of its subfolders in the library.
4. In the shortcut menu, select "Paste as separate master copies".
   The elements are pasted into the library as separate master copies. In each case, a separate type is created automatically for any objects contained.

See also

Basics on master copies (Page 441)

Using master copies (Page 444)

Library basics (Page 415)

Adding types to the project library (Page 449)

7.10.7.3 Filtering master copies

To make it easier to keep track of a large number of master copies, you can filter the display according to the type of master copy.

Requirement

The "Libraries" task card is displayed or the library view is open.

Procedure

To filter the view, follow these steps:

1. Open the "Master copies" folder in the project library or a global library.
2. In the drop-down list of the toolbar, select the type of objects you want to display under "Master copies".

Result

Only the selected type of master copies is displayed. You can set the filter to "All" to return to an unfiltered view.

See also

Library basics (Page 415)

Creating folders in a library (Page 441)

Basics on master copies (Page 441)

Using master copies (Page 444)

Using the element view (Page 419)

7.10.7.4 Using master copies

Master copies are contained either in the project library or in a global library.

Requirement

The "Libraries" task card is displayed.

Procedure

To paste a master copy into the project, follow these steps:

1. Open the "Master copies" folder or any subfolder of "Master copies" in a library.
2. Drag-and-drop one or more master copies to the desired point of use.
   A copy of the master copy is inserted.

Or:

1. Open the element view.
2. Drag the master copy from the "Elements" palette and drop it at the location where you want to use it.
   A copy of the master copy is inserted.

See also

Basics on master copies (Page 441)

Adding master copies (Page 442)

Filtering master copies (Page 443)

Library basics (Page 415)

Using the element view (Page 419)

7.10.8 Using types and their versions

7.10.8.1 Basics on types

Using types

Types are elements that are required for the execution of user programs. Types can be versioned and further developed from a central location.

The following elements can be stored as type in the project library or the global library:

- Functions (FCs)
- Function blocks (FBs)
- PLC data types
- User data types
- Faceplates
- Figures
- User-defined functions

Any number of instances can be derived from the versions of types in the project. The instances are then linked to the version of the type. If you are using types from a global library, the type is also created in the project library. If the type already exists in the project library, any missing type versions are added if necessary. The instance is then linked only to the respective type version in the project library.

Types and their instances are marked with a black triangle. The following figure shows an instance, marked with a black triangle, and an ordinary program block:

Basics on versioning types

Type versioning provides you with the means to develop types centrally and then roll out the most recent version to the individual projects as an update. In this way, error corrections and added functions can be easily integrated into existing projects. If you have created a new version of a global library, you can update existing projects in an automatic process. This minimizes errors and reduces the amount of maintenance work for large automation solutions that contain numerous individual projects.

Versioning allows you to trace the development process of individual types. Before you release a version, you can try it out in a test environment to determine whether changes made to a type integrate smoothly into an existing project. You only release a version for productive use after you have made sure that everything operates without errors. You can view the history of individual instances in the project at any time and determine the version from which the instance was derived.

The TIA Portal also checks automatically if there are associated objects with individual versions of a type. Associated objects can be, for example, PLC data types or other blocks referenced in a block. All associated objects are already taken into consideration during the creation of a type or during copying between libraries. Before being released, versions of types are checked for their consistency to ensure that no inconsistencies arise in the project.

Versions of types

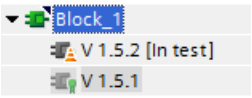
Versions are assigned to each type. The version number is displayed in both the "Libraries" task card as well as the library view next to the respective type. The version number is also displayed in the project tree next to instances of types. This allows you to see at all times which version of an instance is used in the project.

The version number consists of three numbers separated by periods. You can randomly assign the first two digits. Numbers from 1 to 999 are permitted as the first two numbers. The third digit is the build number. It is automatically incremented by one when you edit an instance related to the version. The build number is reset to 1 when you release the version.

The versions of types can have three states:

- In progress (faceplates and HMI user data types)
- In test (all sorts of types except faceplates and HMI user data types)
- Released

The following figure shows a type with two versions. One version is "in test" and one version is released:



See also

State of type versions (Page 446)

Basics on master copies (Page 441)

Adding types to the project library (Page 449)

Using types (Page 450)

7.10.8.2 State of type versions

The versions of types can be in three different states. The states can be determined from the instance or in the library.

"In progress" status

Only versions of faceplates and HMI user data types have the "in progress" state. If a version is in progress, "in progress" appears next to the version in the library.

When you create a new type or a new version of a released type, the type is assigned the status "in progress".

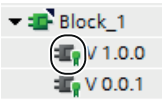
Types with the status "in progress" can be edited in the library view. A reference to an instance in the project does not have to exist. Upon release, the compatibility of the type is tested by a consistency check.

"In test" status

All types except for faceplates and HMI user data types can have the "in test" status. If a version is in test, "in test" appears next to the instance and in the library. A version in the test state is linked to a test instance in the project. This allows you try out the effects of your changes in a test environment including all online functions before you release a type for use in actual operations.

"Released" status

The "released" status is available for all types, regardless of the point of use. If a version has been released, the symbol of the version is marked with a seal in the library:



Released versions can be opened with write protection in their instance. If you want to edit a released version, you must first create a new "in progress" or "in test" version.

See also

Basics on types (Page 444)

Using types (Page 450)

Creating a test version of a type (Page 451)

Editing a test version of a type (Page 452)

Creating an editing version of a type (Page 453)

Performing a consistency check for a version (Page 453)

Discarding versions (Page 454)

Release versions (Page 455)

Assigning a version (Page 463)

Updating the project to the latest versions (Page 458)

Remove the link between instance and type (Page 459)

Filtering the display of types (Page 426)

7.10.8.3 Displaying a released type version

If you display a released version but do not want to edit it, open the instance with write protection. All types except faceplates and HMI user data types can be opened directly at the instance. Faceplates and HMI user data types can only be opened in the "Libraries" task card or in the library view.

Requirement

The released version has an instance in the project; except with types, this is a faceplate or an HMI user data type.

Opening a type version at an instance

To open a released version of a type with write protection starting from an instance, follow these steps:

1. Select the released version at the instance in the project tree.
2. To do this, right-click the instance and select the "Open" command from the shortcut menu. The instance is opened with write protection.

Opening a type version in the "Libraries" task card or library view

To open a released version of a type in the "Libraries" task card or in the library view, follow these steps:

1. Select the version.
2. To do this, right-click the version and select the "Open" command from the shortcut menu. If it is a faceplate or HMI user data type, it is opened directly in the library view. In this case, skip the remaining steps. If it is any other type, the "Open type" dialog opens.
3. Select the instance with the version that you want to display from the list of instances.
4. Click "OK" to confirm.
   The instance is opened with write protection.

7.10.8.4 Displaying properties of a type or version

The following properties can be viewed and changed in the properties of a type or its versions, if needed:

- Name of the type
- Version number (only visible in the case of a version)
- Data and time of the last change to the version (only visible in the case of a version)
- Author who created the version (only visible if there is a version)
- Comments on the type or the version of the type
  The comment on the type can be changed.
- Original library (only visible in the case of a version)
  Displayed is the project and the library from which the current version of the type was generated. This information is important, for example, for finding the original of the type after it has been copied from another library.
- ID of the type or the version of the type
  This ID is used to uniquely identify the type or the version of the type, even if, for example, there are types or versions with an identical name within the project library or the global library. The ID cannot be changed and is assigned automatically.

Procedure

To display the properties of a type or a version and to enter a comment, follow these steps:

1. Select a type or the version of a type in the "Libraries" task card or in the library view.
2. To do this, right-click on the type or one of its versions and select the "Properties" command from the shortcut menu.
   The "Properties" dialog box opens.
3. If needed, enter a comment on the type in the "Comment" field or edit an existing comment.

7.10.8.5 Working with types in the project library

Adding types to the project library

Types for reuse in the project can be created in the project library for various elements. You can create the following elements as types, for example:

- Program blocks
- Faceplates
- PLC data types
- HMI user data type

If you add an element as a type to the project library and this element has dependencies to other elements, the dependent elements are automatically created as a type as well.

After a type has been added to the project library, the type is linked to the added element from the project.

Requirement

- The "Libraries" task card is displayed.
- The elements that you want to add as type are compiled.
- The elements are consistent.

Procedure

To add an existing element to the project library as a type, follow these steps:

1. Open the project library in the "Libraries" task card.
2. Drag-and-drop one or more elements into the "Types" folder or any subfolder of "Types".
   Alternative: Copy the elements in the project tree to the clipboard and add the elements in the desired project library folder.
   The "Generate type" dialog opens.

3. Enter the properties of the new type:
   - Enter a name for the new type in the "Name of the type" field.
   - Enter a version number for the new type in the "Version" field.
   - Enter the name of the editor who is responsible for the type in the "Author" field.
   - Enter a comment on the type in the "Comment" field.
4. Click "OK" to confirm.
   The new type is generated with a released version. The version is linked to the element that has been added.

See also

Basics on types (Page 444)

Library basics (Page 415)

Adding master copies (Page 442)

Using types

To use the version of a type, create an instance of the version at a suitable location in the project tree. You can also position faceplates and HMI user data types directly in the editor. The instance is then linked to the version of the respective type. If the type contains dependent elements such as a PLC data types referenced in a block, these are also created as instance at a suitable location. You can only assign a version of the same type to a device.

If necessary, you can create instances of several types at the same time.

Requirement

- The "Libraries" task card is displayed.
- The desired versions have been released.
- A device which supports the sort of desired type is already available in the project
- The device is not yet assigned any further instance of the same type.

Procedure

To create instances of type versions, follow these steps:

1. In the project library, navigate to the versions of the types that you want to use.
2. Select the versions of which you want to create instances.
3. Drag the required type versions from the project library and drop them at the location where you want to use them.
   Alternative: To automatically use the most recent version, drag the types from the project library and drop them at the location where you want to use them.

Result

Instances of the types and their dependent elements are generated and pasted at the location where you want to use them. The instances are connected to the respective version of the type in the project library.

See also

Basics on types (Page 444)

State of type versions (Page 446)

Library basics (Page 415)

Using master copies (Page 444)

Creating a test version of a type

Before you release a type for productive use, you need to test the type within a project and on the automation system. The test is conducted in a specific test environment. This test environment can be a CPU, for example.

Create a version with "in testing" status for the test. The creation of a version "in testing" is suitable for all sorts of types except for faceplates and HMI user data types. On the other hand, versions "in progress" can be created from faceplates and HMI user data types.

There are two ways to create a test version of a type and define the test environment:

- In the "Libraries" task card or library view
  Generate the new version with "in testing" status in the "Libraries" task card or in the library view. You can generate the new version either directly from the type or from a specific version of the type.
- At an instance in the project tree
  You can also create the test version directly at the instance in the project tree. Since the instance is always used in a specific version in the project, a new version of the type is generated from the version used at the instance.

You can also create test versions from several types at the same time.

The following rules apply to a version "in testing":

- You can set only one version to "in testing" for each type at a given time.
- A version in testing may only be linked to a single instance in the project. Therefore, it is not possible to copy an instance to the clipboard, to duplicate it or to create an additional type from the instance as long as it has "in testing" status.

Requirement

- There is at least one instance of the type within the project in a given version.
- If you want to create the new version from a particular version of the type, the instance must be used in this version in the project.

Procedure

To generate a new test version of a type or the version of a type, follow these steps:

1. Select the type, a version of the type, or the instance.
   When you create the test version directly at the instance, you can select several elements or folders with multiple selection. You can skip steps 3 and 4 because the test environment is already defined by the selected instance.
2. Right-click the selected element and select the "Edit type" command from the shortcut menu.
   If you have started the editing in the "Libraries" task card or in the library view, the "Edit type" dialog opens. If you have started the editing at the instance in the project tree, the test instance is immediately opened for editing in the library view.
3. Select an instance of the type from the list in the project.
   If you have started editing at the type itself, the following applies:
   – The location at which the instance is used (for example, the CPU) is also used as test environment for the subsequent editing of the type.
   – Selecting the test instance also defines the version to be edited.

   The following applies to editing a specific version:
   If your starting point is a specific version, you can only select from the list instances that are used in the same version.
4. Click "OK" to confirm.

Result

A new version of the type is created. The new version is "in testing" and is identified as such in the user interface.

See also

Basics on types (Page 444)

State of type versions (Page 446)

Library basics (Page 415)

Editing a test version of a type

When you edit a version "in testing", a new version is not created. You can start the editing of the test version at the instance in the project tree, in the "Libraries" task card" or in the library view.

Note

Deleting and renaming interface parameters

You can add new parameters. However, if you rename or delete existing parameters, these parameters will not be supplied when the block is called.

Procedure

To edit the test version of a type, follow these steps:

1. Right-click the test version or the instance.
2. Select the "Edit type" command from the shortcut menu.
   The test instance is opened in the library view and can be edited.

See also

Basics on types (Page 444)

State of type versions (Page 446)

Library basics (Page 415)

Performing a consistency check for a version (Page 453)

Discarding versions (Page 454)

Release versions (Page 455)

Creating an editing version of a type

If you want to edit a type with faceplates or HMI user data types, create a new "in progress" version of the type. The new version is edited in the library view. To check the compatibility of the changes, a consistency check is automatically performed for the type prior to the release.

Requirement

The project library is opened in the "Libraries" task card or in the library view.

Procedure

To create a new version of a type in progress, follow these steps:

1. Right-click the type or the version of the type.
2. Select the "Edit type" command from the shortcut menu.
   A new "in progress" version is created and opened for editing in the library view.

See also

Basics on types (Page 444)

State of type versions (Page 446)

Library basics (Page 415)

Performing a consistency check for a version

A type version can unintentionally obtain an inconsistent state during editing. To notice errors in the development process in good time, you can regularly perform a consistency check.

However, the consistency check always takes place automatically as soon as you release a version.

Details of how to start the consistency check manually for the version of a type are provided below.

Requirement

- The project library is opened in the "Libraries" task card or in the library view.
- The version is "in progress" or "in test".

Procedure

To perform a consistency check for the version of a type, follow these steps:

1. Right-click on the version that you want to check for consistency.
2. Select the "Consistency check" command in the shortcut menu.
   The consistency check is carried out. You receive a message with the result of the consistency check.

See also

Release versions (Page 455)

Discarding versions (Page 454)

Editing a test version of a type (Page 452)

Basics on types (Page 444)

State of type versions (Page 446)

Library basics (Page 415)

Discarding versions

Discard versions of a type "in test" or "in progress" when you no longer need the version. You can also select several types or folders and discard all test or editing versions contained therein. All uses of the deleted versions are reset to the last released status.

Requirement

- The versions that you want to discard are in the "In test" or "In progress" status.
- Your are in the library view of the "Libraries" task card is open.

Procedure

To discard one version, follow these steps:

1. Right-click the version that you want to discard.
2. Select the "Discard changes and delete version" command from the shortcut menu.
   The version is deleted.

Alternative:

1. Click the "Discard changes and delete version" button in the toolbar while a version is opened for editing.
   The version is deleted.

See also

Basics on types (Page 444)

State of type versions (Page 446)

Library basics (Page 415)

Performing a consistency check for a version (Page 453)

Discarding all versions within a folder

You can discard all versions with the "In test" or "In progress" status at the same time within a folder. All uses of the deleted version are reset to the last released status.

Requirement

Your are in the library view of the "Libraries" task card is open.

Discarding a version of a single type

To discard all versions within a folder, follow these steps:

1. Right-click the folder.
2. Select the "Discard all" command from the shortcut menu.
   All "In test" or "In progress" versions are deleted.

Release versions

When you are finished editing a type version, release the version for productive use. Assign a version number for the release. You can also select to release several type versions at the same time.

Requirement

- The "Libraries" task card is open or you are in the library view.
- The versions that you want to release are "in test" or "in progress" status.
- The versions are consistent.
  A consistency check is performed as soon as you start the release. If errors that prevent a release occur during the consistency check, a message is displayed indicating how you can correct the errors.

Procedure

To release type versions, follow these steps:

1. Select the versions you want to release.
2. Right-click your selection.
3. Select the "Release version" command from the shortcut menu.
   The "Release type version" dialog box opens.
4. If necessary, change the properties of the version:
   – Enter a name for the type in the "Name" field. If you have selected several versions for release, the "Name" field cannot be changed.
   – In the "Version" field, define a main and an intermediate version number for the version to be released. If you have selected several versions for release, the "Version" field cannot be changed and the last version number is used for the release.
   – In the "Author" field, enter the editor of the version to be released.
   – In the "Comment" field, enter a comment on the version to be released.
5. Optional: Select the "Delete all unused type versions from the library" check box to delete all versions from the library that are not connected to any instance in the project. Versions with dependencies on other types or master copies are not deleted.
6. Click "OK" to confirm.

Alternative:

1. Click the "Release version" icon in the toolbar while a version is opened for editing.
2. Continue with steps 3 to 5 of the description above.

Result

The selected versions are released.

The properties are applied for the types themselves, the versions to be released, and for all future versions. Versions already released remain unaffected by the changes.

If needed, all instances with the same original version are updated to the most recent version and the unused versions of the type are deleted.

See also

Releasing all versions within a folder (Page 457)
Basics on types (Page 444)
State of type versions (Page 446)
Library basics (Page 415)
Performing a consistency check for a version (Page 453)
Assigning a version (Page 463)
Adding types to a global library (Page 460)
Updating the project to the latest versions (Page 458)

Releasing all versions within a folder

When you are finished editing all types within a folder, release all versions simultaneously.

Requirement

- Your are in the "Libraries" task card or in the library view.
- The folder includes versions "in test" or "in progress" status.
- All versions "in test" or "in progress" are consistent.
  A consistency check is performed as soon as you start the release. If errors that prevent release occur during the consistency check, a message is displayed indicating how to correct the errors.

Procedure

To release all type versions within a folder, follow these steps:

1. Right-click the required folder.
2. Select the "Release all" command from the shortcut menu.
   The "Release type version" dialog box opens.
3. If necessary, change the properties of the version:
   – In the "Author" field, enter the editor of the versions to be released.
   – In the "Comment" field, enter a comment on the versions to be released.
4. Select the "Delete all unused type versions from the library" check box to delete all versions from the library that are not connected to any instance in the project. Versions with dependencies on other types or master copies are not deleted.
5. Click "OK" to confirm.

Result

All type versions "in test" or "in progress" status within the selected folder are released.

The properties are applied for the versions to be released and for all future versions. Versions already released remain unaffected by the changes.

Versions of types not used in the project may be deleted.

See also

Release versions (Page 455)

Updating the project to the latest versions

After you have updated several types in the project library, update all instances in the project to the most recent version of the types from the project library. If you do not want to apply the changes to the entire project, restrict the update to individual devices in the project.

Each of the following elements can be selected as source for the update:

- The entire project library
- Individual folders within the project library
- Individual types
  You can select multiple types.

Requirement

Your are in the "Libraries" task card or in the library view.

Procedure

To update instances in a project with the contents from the project library, follow these steps:

1. Select the entire project library or elements from it.
2. Right-click the required elements and select the "Update > Project" command from the shortcut menu.
   The "Update project" dialog box opens.
3. Select either the entire project or individual devices for the update.
4. Select the options for the update process:
   – The "Update all instances of the affected types" check box is always selected during this process.
   – Select the "Delete all unused versions of affected types" check box to delete all older versions of the updated types from the project library.
5. Click "OK" to confirm.
   The update is performed.

Result

The following changes were made to the project:

- All older versions were deleted from the project library as needed.
- All instances within the selected devices were updated to the most recent version of the linked types.
- You can find a log of the update process in the projection tree under "Common data".

See also

Updating the project to the latest type versions (Page 462)

Using logs (Page 331)

Updating a library with the contents of another library (Page 467)

Basics on types (Page 444)

Library basics (Page 415)

State of type versions (Page 446)

Remove the link between instance and type

Instances of types are always connected to the version of the corresponding types. They cannot be edited like an ordinary object. If you edit the instance, a new version of the type is created automatically in the library and the changes therefore affect the entire project.

If you remove the link between the instance and its type, you then edit the object like an ordinary object in the project tree.

Requirement

The instance may not be "in test".

Procedure

To remove the link between instances and their type versions, follow these steps:

1. Select one or more instances in the project tree.
2. Right-click the selection and select the "Terminate connection to type" command from the shortcut menu.
3. The link to the corresponding type versions is removed.

See also

Basics on types (Page 444)

Library basics (Page 415)

State of type versions (Page 446)

7.10.8.6 Working with types in global libraries

Adding types to a global library

Global libraries are used as a central resource when working on multiple projects. Among the types, only the types in the project library can be edited directly. Therefore, use the project library if you want to work on types. When you are finished editing a type in the project library, you can add the type to a global library. Adding types from the project library corresponds to a normal copy process from the project library.

Requirement

- The "Libraries" task card is displayed or the library view is open.
- The global library to which you want to add types is opened and can be written to.

Procedure

To add types to a global library, follow these steps:

1. Open the required folder in the global library in the "Libraries" task card or in the library view.
2. Drag one or more types from the project library to the "Types" folder or any subfolder of the global library.

Alternative:

1. Copy the required types from the project library to the clipboard.
2. Open the required global library in the "Global library" palette of the "Libraries" task card.
3. Right-click the "Types" folder or any of its subfolders.
4. Select the "Paste" command in the shortcut menu.

Result

The types are inserted in the global library. Dependent types, such as types of HMI user data types or tags, are also copied to the global library, provided they do not already exist there. This ensures that all necessary elements for generating an instance are present in the global library.

If a type already exists in the global library, the described action corresponds to an update of the global library. In this case, the most recent released versions of the types are added to the global library.

See also

Basics on types (Page 444)

Release versions (Page 455)

Assigning a version (Page 463)

Updating the project to the latest type versions (Page 462)

Library basics (Page 415)

Using types

To use the version of a type from the global library, create an instance of the version at a suitable location in the project tree. You can also position faceplates and HMI user data types directly in the editor.

In the project, instances of types from a global library are not linked to the type in the global library. Instead, a copy of the type and its dependent elements is generated in the project library when an instance is pasted into the project. Dependent elements, for example, can be PLC data types that are referenced in a block. In each case, the copy of the type and the dependent elements in the project library contain the version that you link to the instance. If the type or a dependent element already exists in the project library, only the missing version is added when necessary to the type in the project library.

The instance is linked to the type in the project library. You can only assign a type to a device once irrespective of the version.

Requirement

- The "Libraries" task card is displayed.
- A device which supports the sort of desired type is available in the project.
- The device is not assigned to any further instances of the same type.

Procedure

To generate and use an instance from a type, follow these steps:

1. In the global library, navigate to the version of the type that you want to use.
2. Drag the required version of the type and drop it at the location where you want to use it.
   Alternative: To automatically use the most recent version, drag the type to the location where you want to use it.

Result

Missing elements are added to the project library. An instance is generated from the type and its dependent elements and pasted at the location where you want to use it. The instances are connected to the respective version of the type in the project library.

See also

Basics on types (Page 444)

Updating the project to the latest type versions (Page 462)

Library basics (Page 415)

Using the element view (Page 419)

Updating the project to the latest type versions

In large enterprises with numerous automation projects. the global libraries are frequently edited from a central location. The updated global libraries of the individual projects are made available after the completion of a new version. If you have received a more recent version of a global library, replace the outdated instances in your project with the most recent version. If you do not want to apply the changes to the entire project, restrict the update to individual devices in the project.

The project library is also updated with the new versions of the types in the global library during the updating of the project or individual devices.

The following elements can be selected as source for the updating:

- A global library
- Individual folders within a global library
- Individual types
  You can select multiple types.

Requirement

- Your are in the "Libraries" task card or in the library view.
- The updated global library is open.

Procedure

To update instances in a project with the contents from a global library, follow these steps:

1. Select the updated global library or the individual elements from it.
2. Right-click the global library or the required elements and select the "Update > Project" command from the shortcut menu.
   The "Update project" dialog box opens.
3. Select either the entire project or individual devices for the update.
4. Select the options for the update process:
   – The "Update all instances of the affected types" check box is always selected during this process.
   – Select the "Delete all unused versions of affected types" check box to delete all older versions of the updated types from the project library.
5. Click "OK" to confirm.
   The update is performed.

Result

The following changes were made to the project:

- The most recent version of the select types is present in the project library. All older versions were deleted if necessary.
- All instances within the selected devices were updated to the most recent version of the linked types.
- You can find a log of the update process in the projection tree under "Common data".

See also

Updating the project to the latest versions (Page 458)

Using logs (Page 331)

Updating a library with the contents of another library (Page 467)

Basics on types (Page 444)

Library basics (Page 415)

Adding types to a global library (Page 460)

7.10.8.7 Assigning a version

A library is more clearly structured if types related by content have the same version number. The identical version number reflects the work progress. When you have completed the work on multiple associated types, you can assign the same version number to these types.

You have the following options to assign a common version to types:

- The entire project library or a complete global library
- One or more folders in a library
- One or more types

Requirement

- The "Libraries" task card or the library view is open.
- Your selection must not contain types with "in test" or "in progress" status.

Procedure

To assign several types the same version, follow these steps:

1. Select the types to which you want to assign a common version.
2. Select the "Assign version" command from the shortcut menu.
   The "Assign version" dialog box opens.

3. If necessary, change the properties of the version:
   - In the "Version" field, determine the new version number. The version number must be higher than the highest version number of all selected types.
   - In the "Author" field, enter the person responsible for the version to be released.
   - In the "Comment" field, enter a comment on the version to be released.
4. If you have started the operation within the project library, select the options for the release. The following options cannot be selected in a global library.
   - Select the "Update all instances" check box to update all instances in the project to the new version. Types used in master copies are not updated.
   - Select the "Delete all unused versions from the library" check box to delete all older versions of the types from the project library. The versions are only deleted if they are not assigned to an instance in the project and if they have no dependencies to other types.
5. Click "OK" to confirm.

Result

The selected type versions are changed as follows:

- A new version of all selected types is created with the specified version number.
- The properties are applied to all selected types, the new version and to all future versions. Lower versions remain unaffected by the changes. If you make no changes to the properties, the properties of the last released version of each type are applied.
- If needed, all instances are updated to the most recent version and the unused versions of all types are deleted from the project library.
- The build number of dependent types is incremented to the next free build number as long as the dependent types were not in your selection. If you had selected a dependent type as well, the version number you specified will be assigned.

A log of the changes is created. If you have versioned the types in the project library, you find the log in the project tree under "Common data > Logs". If you have versioned types in a global library, you find the log in the "Common data > Logs" folder in the level below the global library.

See also

Basics on types (Page 444)

State of type versions (Page 446)

Library basics (Page 415)

Release versions (Page 455)

Adding types to a global library (Page 460)

Using logs (Page 331)

Displaying logs of global libraries (Page 433)

7.10.9 Editing library elements

Types, master copies and folder can be cut, copied, pasted, moved, renamed or deleted in the usual way within the "Libraries" task card or the library view. Global libraries must be opened with write permission for each of the above-described processes.

Note

User help for types and master copies

User help is not affected by any of the operations within the library. When you move a master copy or a type to a different location, you are also moving the associated user help in the file system to the corresponding location.

Additional help on using the user help can be found in the chapter "Conventions for creating user help".

Copying types

The following rules apply when you copy types to the clipboard:

- Types are always copied to the clipboard with all associated versions. However, only versions that have previously been released are copied.
- Types are always copied to the clipboard with all dependent elements.
- Master copies are always copied to the clipboard with all type versions used in them.

Copying and pasting type versions

When you copy type versions to another library, the types must already exist in the target library.

Cutting elements

You can only paste previously cut library elements into the same library. In so doing, you can only paste master copies into the "Master copies" folder or any of its subfolders. Likewise, you can only paste types into the "Types" folder or any of its subfolders.

Pasting types

Pasting types in a different library corresponds to updating the target library.

The following rules apply when you have pasted a type into a different library:

- A type is always pasted with all its versions.
- If the type already exists in the target library, all versions that are more recent than the existing versions are added to the corresponding types in the target library.
- If there is already a version with released status in the target library, this version is not pasted again.

- If the same version exists with in test or in progress status in the target library, it is replaced by the released version.
- If a type needs other types, these are also added at the respective location.

Pasting master copies

When you paste master copies, all type versions contained in these copies are also pasted. If the corresponding types already exist in the library, only the missing versions are added to the individual types. If one of the types used does not yet exist, it is pasted at the highest level in the library. The type includes the type version that was used in the master copy.

Moving elements

When you move an element from one library to another, the element is copied and not moved. The same rules apply as described under "Pasting types" and "Pasting master copies".

Deleting of types and type versions

Note the following when you delete types or type versions:

- A type or a type version can only be deleted if there are no dependencies to other types.
- When you delete a type, all versions of the type are deleted.
- If you delete all versions of a type, the type is also deleted.
- If you delete a version that has instances in the project, the instances are also deleted from the project.
- If you delete a type that is also stored at the same time as a master copy, the master copy is also deleted.

Deleting instances

If you delete an instance that has dependencies to other instances, this instance is restored during the next compilation. The instance is then linked again to the original type version. This restores the consistency of the project.

See also

Library basics (Page 415)

Remove the link between instance and type (Page 459)

Updating a library with the contents of another library (Page 467)

7.10.10 Updating a library with the contents of another library

An existing library can be updated with the contents of another library. The following options are available for updating libraries:

- Updating a global library with types from another global library or from the project library
- Updating the project library with types from a global library

Each of the following elements can be selected as source for the update:

- An entire library
- Individual folders within a library
- Individual types
  You can select multiple types.

During the update, new versions are added to the types that already exist in the target library. Types that do not yet exist in the target library are copied together with all their versions to the target library.

Note

User help for copying types

User help is not copied together with a type to another library. You need to copy the user help for types to the corresponding directory.

Additional help on using the user help can be found in the chapter "Conventions for creating user help".

Requirement

If you want to update a global library, you have to open it with write permission.

Procedure

To update a library with the contents of a different library, follow these steps:

1. Select a library or individual elements from a library as source for the update.
2. Right-click the source and select the "Update > Library" command from the shortcut menu. The "Update library" dialog box opens.
3. Select the type of library you want to update:
   - Select "Update the project library" to update the project library with types from a global library.
   - Select "Update a global library" if you want to update a global library.
4. Optional: In the drop-down list, select the global library that you want to update, if you want to update a global library.

5. Select the options for the update:
   - The "Update all instances of the affected types" option is always disabled during this process.
   - Select the "Delete all unused versions of the affected types" check box to delete all older versions of types from the project library if these are not assigned to any instance in the project and if there are no dependencies to other types. This option cannot be selected for the update of a global library, since types of a global library never have a point of use in the project.
6. Click "OK" to confirm.
   The update is performed.

Result

The following changes were made to the target library:

- Types not yet available in the target library were copied together with all their versions. More recent versions were added to the types that already exist in the target library. If a more recent version of a type already existed in the target library, the latest version was nevertheless copied and automatically assigned a newer version number.
- If needed, all versions of types were deleted from the project library if these were not used in any instance in the project.
- A log listing all performed changes to the target library was created for the update process. If you have updated the project library, you can find the log in the project tree under "Common data > Logs".
  If you have updated a global library, you can find the log in the "Common data > Logs" folder in the level below the global library.

See also

Using logs (Page 331)

Updating the project to the latest versions (Page 458)

Updating the project to the latest type versions (Page 462)

Displaying logs of global libraries (Page 433)

Library basics (Page 415)

7.10.11 Harmonizing names and path structure

You can harmonize the project with a library. This helps you correct the following items:

- Names of the instances:
  Instances can be created during the development phase of a library, the names of which are appended by "_1", "_2", etc. due to an automatic correction. This extension is used to prevent duplicate names in the project. During harmonization, the instances once again receive the names of their associated types.
- Path structure:
  The original path structure can be lost due to parallel development or copying of dependent instances. This affects the clarity of the project. During harmonization, the path structure within the project is adapted to the path structure of the library.

Procedure

To harmonize the names and the path structure, follow these steps:

1. Open the library management.
2. Click "Harmonize project" in the toolbar.
   The "Harmonize project" dialog box opens.
3. Select the device with which you wish to harmonize the library.
4. Select the "Harmonize paths between project and library" check box if you want to restore the path structure.
5. Select the "Harmonize names between project and library" check box if you want to have the names corrected.
6. Confirm your entries with "OK".

Result

Depending on your settings, the names and the path structure in the project are harmonized with the library.

The changes to the project are logged. The log is available under "Common data > Logs" in the project navigation.

See also

Library basics (Page 415)

Overview of the library view (Page 420)

Overview of the library management (Page 423)

Using logs (Page 331)

7.10.12 Clean up library

You can clean up the project library or global libraries to remove types or versions that are not connected to any instance in the project. This step provides more clarity within the libraries and decreases the size of the library.

Cleaning up the project library

To clean up the project library, follow these steps:

1. Open the library management.
2. Click on "Clean up library" in the toolbar.
   The "Clean up project library" dialog box opens.
3. Select the scope in which types or versions are to be deleted:
   – to retain the version with the highest version number, even if this has no instance, select the option "Delete old type versions and retain the newest type version".
   – Select the option "Delete complete types" to delete the complete type if no version is connected to an instance.
4. Confirm your entry with "OK".
   Depending on your selection, either unused type versions or types are removed from the project library.
   The changes are logged. The log is available under "Common data > Logs" in the project navigation.

Clean up global library

To clean up a global library, follow these steps:

1. Open the library management.
2. Click on "Clean up library" in the toolbar.
   The "Clean up global library" dialog box opens.
3. Click "Continue".
   Unused type versions are deleted. The latest version of a type is always retained.
   The changes are logged. The log is available in the "Common data > Logs" folder in the level below the global library.

See also

Library basics (Page 415)

Overview of the library view (Page 420)

Overview of the library management (Page 423)

Using logs (Page 331)

Displaying logs of global libraries (Page 433)

7.10.13 Comparing library elements

Introduction

You can compare devices from libraries with devices from both the current project as well as from the same or another libraries or reference projects. Note, however, that reference projects are write-protected. You can also compare instances in a device with their type version in a library. Not all actions are available for the comparison with types. You cannot, for example, overwrite an instance of a newer version with an older type version from the library.

When comparing library elements, you can always switch between automatic and manual comparison.

Procedure

To compare library elements with the device data of a project, follow these steps:

1. In the project tree, select the device whose data you want to compare to a library element and which allows offline/offline comparison.
2. Select the "Compare > Offline/offline" command in the shortcut menu.
   The compare editor opens and the selected device is displayed in the left area.
3. Open the "Libraries" task card.
4. Select the library element that you want to compare to the device data.
5. Drag the library element into the right drop area of the compare editor.
   You can identify the status of the objects based on the symbols in the status and action area. When you select an object, the object properties and the corresponding object of the assigned device are clearly shown in the property comparison.
   You can drag other devices into the drop areas from the current project, a library or from a reference point at any time to start a new comparison. It does not matter which device you drag into the drop area.

See also

Using the compare editor (Page 365)

Carrying out offline/offline comparisons (Page 364)

Using the library view (Page 420)

Library basics (Page 415)

Overview of the "Libraries" task card (Page 417)

Overview of the library view (Page 420)

Using the element view (Page 419)

Using global libraries (Page 428)

7.11 Using cross-references

7.11.1 Using cross-references

Introduction to cross-references

The cross-reference list provides an overview of the use of objects within the project. You can see which objects are interdependent and where the individual objects are located. Cross-references are therefore part of the project documentation.

You can also jump directly to the point of use of an object.

Which objects you can display and localize in the cross-reference list depends on the installed products.

See also

Displaying cross references of an instance (Page 427)

7.12 Simulating devices

7.12.1 Simulation of devices

Introduction

You can use the TIA Portal to run and test the hardware and software of the project in a simulated environment. The simulation is performed directly on the programming device or PC. No additional hardware is required.

The simulation software provides a graphical user interface for monitoring and changing the configuration. It differs according to the currently selected device.

Integration in the TIA Portal

The simulation software is fully integrated in the TIA Portal but is only supported by certain devices. Therefore, the button for calling the simulation software is only active if the selected device supports simulation.

The simulation software for some devices requires its own virtual interface to communicate with the simulated devices. The virtual interface can be found in the project tree under the "Online access" entry next to the physical interfaces of the programming device/PC.

Once you have opened the software, additional help on the simulation software is available via a separate link.

See also

Starting the simulation (Page 473)

7.12.2 Starting the simulation

Some devices can be simulated with additional software. You therefore do not have to have the actual devices to perform comprehensive testing of your project.

Procedure

To start the simulation software, follow these steps:

1. Select the device you want to simulate, for example, in the project tree.
2. Select the "Simulation > Start" command in the "Online" menu.
   This calls the simulation software.

See also

Simulation of devices (Page 472)

Editing devices and networks 8

8.1 Configuring devices and networks

8.1.1 Hardware and network editor

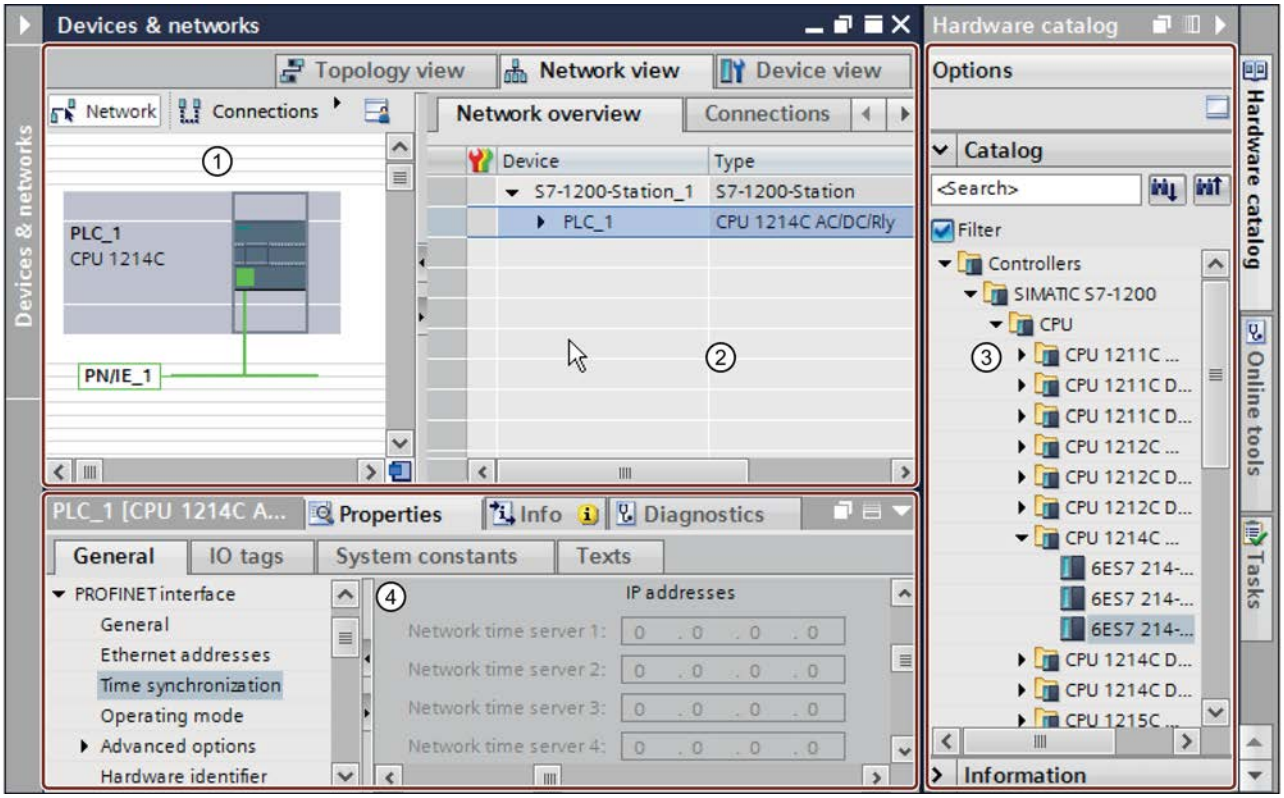
8.1.1.1 Overview of hardware and network editor

Function of the hardware and network editor

The hardware and network editor opens when you double-click on the "Devices and Networks" entry in the project tree. The hardware and network editor is the integrated development environment for configuring, networking and assigning parameters to devices and modules. It provides maximum support for the realization of the automation project.

Structure of the hardware and network editor

The hardware and network editor consists of the following components:



① Device view (Page 479), Network view (Page 477), Topology view (Page 482): Graphic area

② Device view (Page 479), Network view (Page 477), Topology view (Page 482): Table area

③ Hardware catalog (Page 488)

④ Inspector window (Page 487)

The hardware and network editor provides you with three views of your project. You can switch between these three views at any time depending on whether you want to produce and edit individual devices and modules, entire networks and device configurations or the topological structure of your project.

Drag the devices and modules you need for your automation system from the hardware catalog to the network, device ot topology view.

The inspector window contains information on the object currently marked. Here you can change the settings for the object marked.

8.1.1.2 Network view

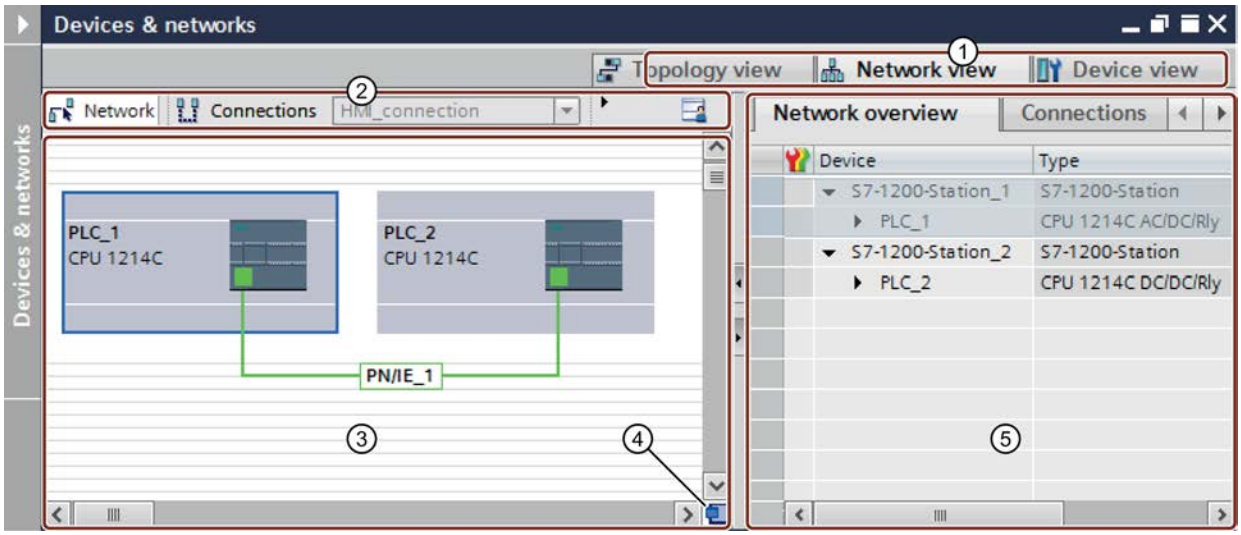
Introduction

The network view is one of three working areas of the hardware and network editor. You undertake the following tasks here:

- Configuring and assign device parameters
- Networking devices with one another

Structure

The following diagram shows the components of the network view:



① Changeover switch: device view / network view / topology view
② Toolbar of network view
③ Graphic area of network view
④ Overview navigation
⑤ Table area of network view

You can use your mouse to change the spacing between the graphic and table areas of the network view. To do this, click between the graphic and the table areas and change the spacing by moving the divider to the left or right while keeping the mouse button pressed. The Speedy Splitter (the two small arrow keys) allows you to use a single click to minimize the table view, maximize the table view or restore the last selected split.

Toolbar

The toolbar provides the following functions:

Icon	Meaning
	Mode to network devices.
	Mode to create connections. You can use the adjacent drop-down list to set the connection type.
	Mode to create relations.
	Show interface addresses.
	Adjust the zoom setting. You can use the adjacent drop-down list to select or directly enter the zoom setting. You can also zoom in or zoom out the view in steps using the zoom symbol or draw a frame around an area to be zoomed in.
	Show page breaks Enables page break preview. Dotted lines are displayed at the positions where the pages break when printed.
	Remember layout Saves the current table view. The layout, width and visibility of columns in the table view is stored.

Graphic area

The graphic area of the network view displays any network-related devices, networks, connections and relations. In this area, you add devices from the hardware catalog, connect them with each other via their interfaces and configure the communication settings.

Overview navigation

Click in the overview navigation to obtain an overview of the created objects in the graphic area. By holding down the mouse button, you can quickly navigate to the desired objects and display them in the graphic area.

Table area

The table area of the network view includes various tables for the devices, connections and communication settings present:

- Network overview
- Connections
- Relations
- I/O communication
- VPN

You can use the shortcut menu of the title bar of the table to adjust the tabular display.

See also

Adding a device to the hardware configuration (Page 504)
Layout of the user interface (Page 277)
Displaying diagnostics status and comparison status using icons (Page 2081)
Networking devices in the network view (Page 520)
Tabular network overview (Page 523)

8.1.1.3 Device view

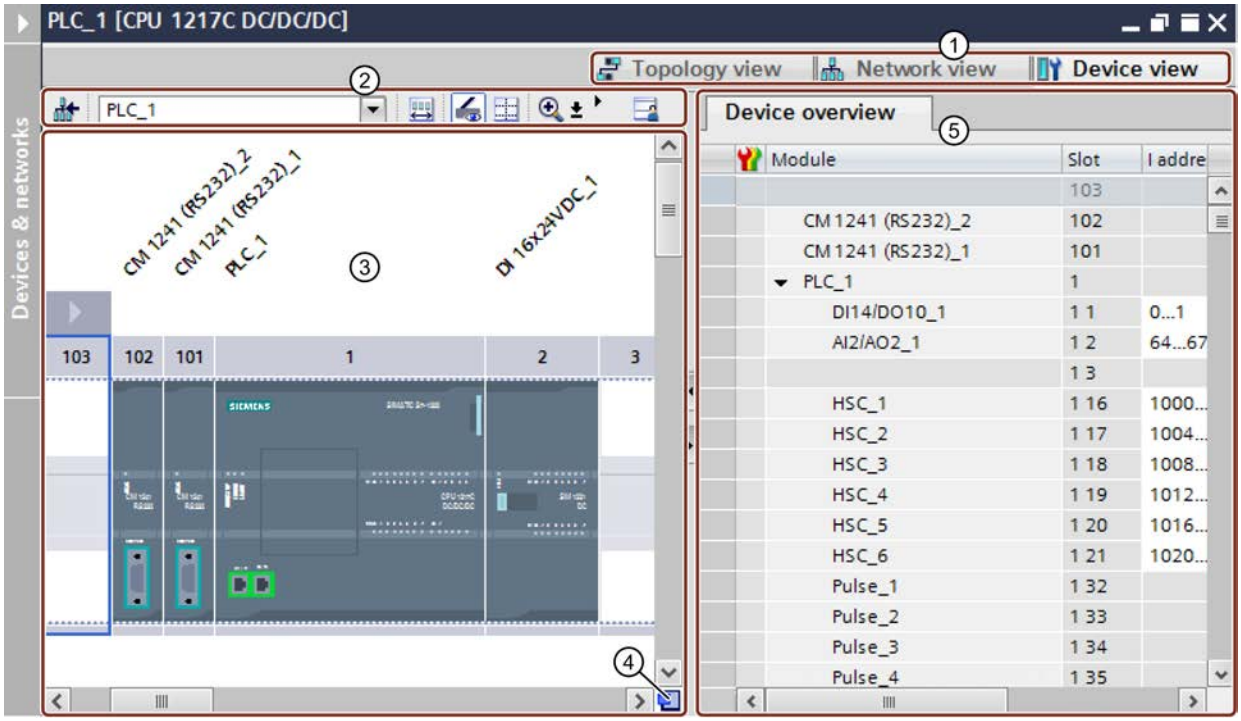
Introduction

The device view is one of three working areas of the hardware and network editor. You undertake the following tasks here:

- Configuring and assign device parameters
- Configuring and assign module parameters

Structure

The following diagram shows the components of the device view:



① Changeover switch: device view / network view / topology view
② Toolbar of device view
③ Graphic area of the device view
④ Overview navigation
⑤ Table area of device view

You can use your mouse to change the spacing between the graphic and table areas of the device view. To do this, click between the graphic and the table areas and change the spacing by moving the divider to the left or right while keeping the mouse button pressed. The Speedy Splitter (the two small arrow keys) allows you to use a single click to minimize the table view, maximize the table view or restore the last selected split.

Toolbar

The toolbar provides the following functions:

Icon	Meaning
	Switches to the network view. Note: The device view can switch between the existing devices using the drop-down list.
	Show the area of unplugged modules.
	Show module labels.

Icon	Meaning
	Adjust the zoom setting. You can use the adjacent drop-down list to select or directly enter the zoom setting. You can use the Zoom icon to zoom in or out incrementally or to drag a frame around an area to be enlarged. With signal modules, you can recognize the address labels from a zoom level of 200% or higher.
	Show page breaks Enables page break preview. Dotted lines are displayed at the positions where the pages break when printed.
	Remember layout Saves the current table view. The layout, width and visibility of columns in the table view is stored.

Graphic area

The graphic area of the device view displays hardware components and if necessary the associated modules that are assigned to each other via one or more racks. In the case of devices with racks, you have the option of installing additional hardware objects from the hardware catalog into the slots on the racks.

Overview navigation

Click in the overview navigation to obtain an overview of the created objects in the graphic area. By holding down the mouse button, you can quickly navigate to the desired objects and display them in the graphic area.

Table area

The table area of the device view gives you an overview of the modules used and the most important technical and organizational data.

You can use the shortcut menu of the title bar of the table to adjust the tabular display.

See also

Working with racks (Page 498)

Network view (Page 477)

Area for unplugged modules (Page 501)

Inserting a module into a rack (Page 507)

Objects in the device view (Page 500)

Layout of the user interface (Page 277)

Displaying diagnostics status and comparison status using icons (Page 2081)

8.1.1.4 Topology view

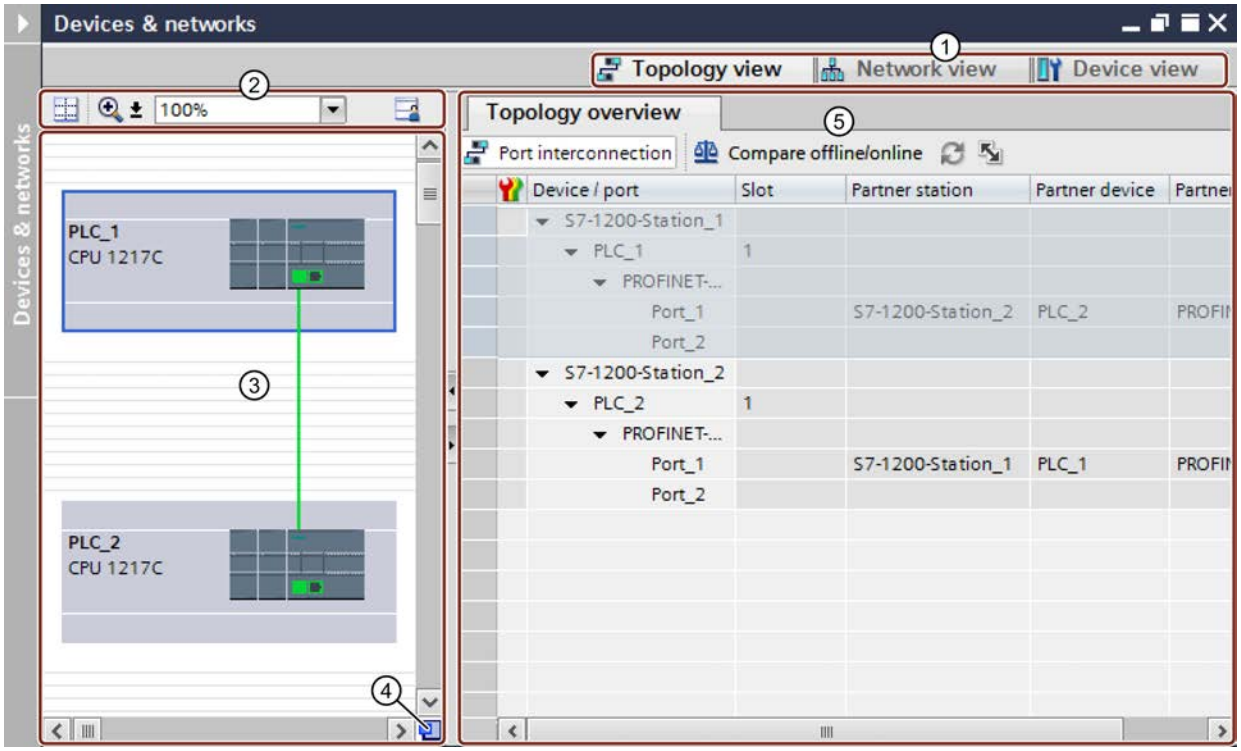
Introduction

The topology view is one of three working areas of the hardware and network editor. You undertake the following tasks here:

- Displaying the Ethernet topology
- Configuring the Ethernet topology
- Identifying and minimizing differences between the desired and actual topology

Structure

The following figure provides an overview of the topology view.



① Changeover switch: device view / network view / topology view
② Topology view toolbar
③ Graphic area of the topology view
④ Overview navigation
⑤ Table area of the topology view

You can use your mouse to change the spacing between the graphic and table areas of the topology view. To do this, click between the graphic and the table areas and change the spacing by moving the divider to the left or right while keeping the mouse button pressed. The Speedy Splitter (the two small arrow keys) allows you to use a single click to minimize the table view, maximize the table view or restore the last selected split.

Toolbar

The toolbar provides the following functions:

Icon	Meaning
	Adjusting the zoom setting. You can use the adjacent drop-down list to select or directly enter the zoom setting. You can also zoom in or zoom out the view in steps using the zoom symbol or draw a frame around an area to be zoomed in.
	Show page breaks Enables page break preview. Dotted lines are displayed at the positions where the pages break when printed.
	Remember layout Saves the current table view. The layout, width and visibility of columns in the table view is stored.

Graphic area

The graphic area of the topology view displays Ethernet modules with their appropriate ports and port interconnections. Here you can add additional hardware objects with Ethernet interfaces. See: Adding a device to the hardware configuration (Page 504)

Overview navigation

Click in the overview navigation to obtain an overview of the created objects in the graphic area. By holding down the mouse button, you can quickly navigate to the desired objects and display them in the graphic area.

Table area

This displays the Ethernet or PROFINET modules with their appropriate ports and port interconnections in a table. This table corresponds to the network overview table in the network view.

You can use the shortcut menu of the title bar of the table to adjust the tabular display.

See also

Layout of the user interface (Page 277)

Displaying diagnostics status and comparison status using icons (Page 2081)

8.1.1.5 Overview of settings for hardware configuration

You can make some settings for the hardware configuration in the settings of the TIA Portal.

Overview

The following table provides an overview of the settings for the hardware configuration:

Group	Setting	Description
Information on product support	Deactivated	Prevents access to the Siemens Industry Online Support
	Via Internet	Enables access to product information about individual devices in the hardware catalog via the Internet.
Topological overview	Temporarily assigning an IP address	Assigns a temporary IP address for topology discovery if a device does not have a valid IP address. Topology information (LLDP) cannot be read from a device without valid IP address.
	Show a warning if the option is activated	Shows a warning when a temporary IP address is assigned to a device during topology discovery.
Compiling	Download module comment	Transfers any existing comments in addition to the hardware configuration when loading the hardware configuration to the device. The comment is available after the device is loaded to a programming device.

See also

Changing the settings (Page 276)

8.1.1.6 Printing hardware and network configurations

Printout of hardware and network configurations

You can print out the following elements of the hardware and network view as part of the project documentation:

- Graphic network view
- Network overview table
- Graphic device view
- The device overview table
- The parameters of the object currently selected in the editor

Printout of editor content

If you start a printout within an opened editor and no module is selected, the content of the editor is always printed. This includes a graphic representation of the editor as well as the table for the editor. You can adapt the scope of the printout. You can specify whether only the graphic view, the table or both together are to be printed. Read section "Changing the print options (Page 486)" for more on this.

If the graphic is larger than the page layout you have selected, the printout is continued on the next page. No content is lost this way. Alternatively, you can change the zoom level of the graphic representation to fit the printout on one page. The printout is always made in the currently selected zoom setting.

To check that all content fits on one page, you can either use the print preview or activate the page break preview. When page break preview is activated, dashed lines are displayed within the graphic editor at the location where the page break is later made.

Printing very large tables

If a table is larger than the print area and therefore cannot be fully printed, the content of the table is not printed as a table, but instead as pairs between value and key.

Example:

Object name	Property 1	Property 2
Object A	Value A1	Value A2
Object B	Value B1	Value B2

In this case, the printout has the following appearance:

Object A

Property 1: Value A1

Property 2: Value A2

Object B

Property 1: Value B1

Property 2: Value B2

You can also preset this as a template so that tables are always printed as pairs between the key and the value. Read section "Changing the print settings (Page 382)" for more on this.

Printing module parameters

Parameters of selected modules are printed out along with the current value settings in text form. All parameters from corresponding modules are also printed. For example, if you have selected a CPU, the parameters of an inserted signal board, if present, are printed as well.

You can determine the scope of the module parameters to be printed. In the "Print" dialog, select whether all properties and parameters of a module are to be printed or whether to use the compact printout. If you select the compact form, only the entries in the "General" area of the module properties are printed. Comments on modules, as well as the author and module description, are excluded. In compact mode, the following module parameters are therefore printed, for example:

- Module specifications
  Name, module slot, short description, order number, firmware version
- Name of the PROFINET interface
- Subnet specifications
  Name of the subnet, ID of the S7 subnet

See also

Changing the print options (Page 486)
Documentation settings (Page 380)
Creating a print preview (Page 394)
Printing project data (Page 397)
Activating the page break preview for printout (Page 486)

8.1.1.7 Activating the page break preview for printout

You can activate the page break preview for the printout in the graphic editor. If this option is activated, dashed lines are shown within the graphic editor at the locations where page breaks are later made during printout.

Procedure

Proceed as follows to activate the page break preview:

1. Select the graphic area of the corresponding view.
2. Click on the "Show page break" symbol in the toolbar of the graphic editor.
   Dashed lines are displayed within the graphic editor at the location a page break is later made.
3. To modify the frame layout, select the "Print" command in the "Project" menu.
4. To disable page break preview, click again on the "Show page break" symbol in the toolbar of the graphic editor.

8.1.1.8 Changing the print options

Changing the scope of the printout

When printing from an editor, you can specify whether both graphics and tables are to be printed or just one of the two. Both are printed by default.

Procedure

To change the scope of the printout, proceed as follows:

1. In the "Options" menu, select the "Settings" command.
2. In the area navigation, open the "Print settings" parameter group under "General".
3. Scroll to the "Hardware configuration" group.
4. Select or clear the "Active graphic view" check box, depending on whether you want to print the graphics of the network and device view as well.
5. Select or clear the "Active table" check box, depending on whether you want to print the table for the editor as well.

See also

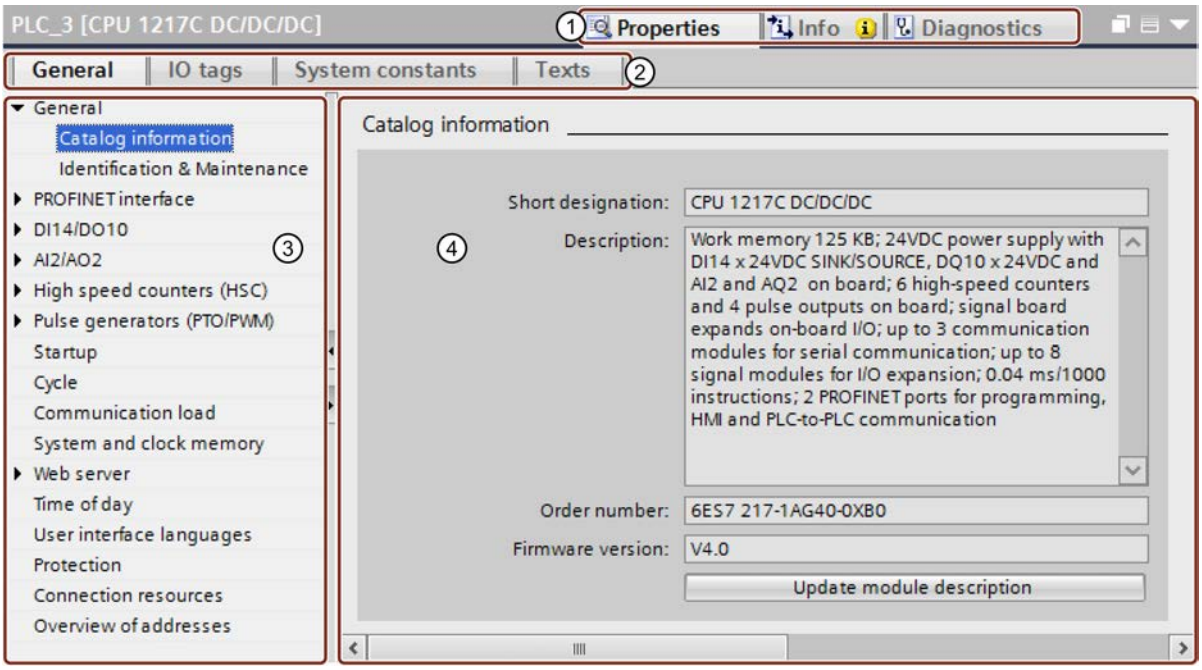
Printing hardware and network configurations (Page 484)

8.1.1.9 Inspector window

The properties and parameters shown for the object selected can be edited in the Inspector window.

Structure

The Inspector window consists of the following components:



① Switch between various information and work areas

② Switch between various tabs of the information and work areas

③ Navigation between various pieces of information and parameters

④ Display showing the selected information and parameters

Function

The information and parameters in the Inspector window are split into different types of information:

- Properties
- Info
- Diagnostics

To display the corresponding information and parameters, click in the relevant area. The "Properties" area is the most important one for configuring an automation system. This area is displayed by default and contains various tabs:

- General: Display the properties and settings of the device or module. Here you can edit the settings and parameters. The left pane of the Inspector window is used for area navigation. Information and parameters are arranged there in groups. If you click on the arrow symbol to the left of the group name, you can expand the group if sub-groups are available. If you select a group or sub-group, the corresponding information and parameters are displayed in the right pane of the Inspector window and can be edited there too.
- IO tags: Display the IO tags of the PLC. You can assign names for the tags, assign the tags to the user-defined tag tables via a drop-down list, and enter comments for the tags. The IO tags are also shown in the PLC tag table.
- System constants: Display the constants required by the system with the HW IDs of the modules. The system constants are also shown in the PLC tag table.
- Texts: Display the reference language and specify the text source for the project texts.

See also

Editing properties and parameters (Page 513)

Overview of hardware and network editor (Page 475)

Translating text associated with individual objects (Page 344)

Overview of PLC tag tables (Page 2473)

Project text basics (Page 340)

What you need to know about HW identifiers (Page 883)

Addressing modules (Page 937)

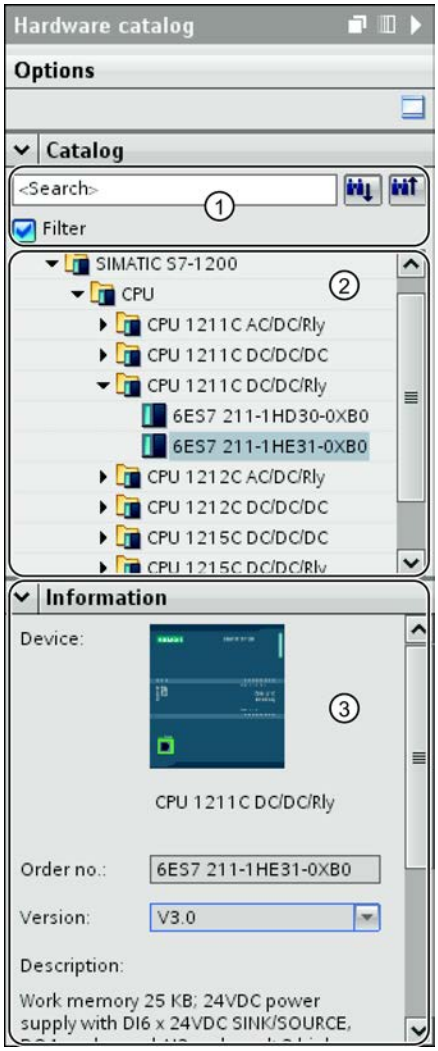
Addressing of modules (S7-1500) (Page 881)

8.1.1.10 Hardware catalog

The "Hardware catalog" task card gives you easy access to a wide range of hardware components.

Structure

The "Hardware catalog" task card consists of the following panes:



① "Catalog" pane, search and filter function

② "Catalog" pane, component selection

③ "Information" pane

Search and filter function

The search and filter functions of the "Catalog" pane make it easy to search for particular hardware components. You can limit the display of the hardware components to certain criteria using the filter function. For example, you can limit the display to objects that you can also place within the current context or which contain certain functions.

Objects that can be used in the current context include, for example, interconnectable objects in the network view or only modules compatible with the device in the device view.

Component selection

The component selection in the "Catalog" pane contains the installed hardware components in a tree structure. You can move the devices or modules you want from the catalog to the graphic work area of the device or network view.

Installed hardware components without a license are grayed out. You cannot use non-licensed hardware components.

Hardware components belonging to various components groups thematically are partially implemented as linked objects. When you click on such linked hardware components, a catalog tree opens in which you can find the appropriate hardware components.

Information

The "Information" pane contains detailed information on the object selected from the catalog:

- Schematic representation
- Name
- Order number
- Version number
- Description

See also

Browsing the hardware catalog (Page 497)

Overview of hardware and network editor (Page 475)

8.1.1.11 Enabling product support

For each device in the hardware catalog, you have the option of displaying additional information that is stored in the Siemens Industry Online Support. By default, the function is disabled. Instructions for enabling the function are given below.

Requirement

The TIA Portal has access to the Internet.

Procedure

To enable access to Siemens Industry Online Support, follow these steps:

1. In the "Options" menu, select the "Settings" command.
2. Open the "Hardware configuration" group in the area navigation.
3. Select the "Via Internet" check box.

Result

You can access product support, FAQs and manuals in the hardware catalog via the shortcut menu for the module.

See also

Displaying product support for hardware components (Page 491)

8.1.1.12 Displaying product support for hardware components

In the hardware catalog, you have direct access to information that is stored for each module in Siemens Industry Online Support. You can jump directly to the following pages in Siemens Industry Online Support:

- Information on product support
- FAQs
- Manuals

Requirement

- You have access to the Internet.
- Access to product support is enabled in the settings of the TIA Portal.
  For information on how to enable the function, refer to section "Enabling product support (Page 490)".

Procedure

To display the information for a particular module in Siemens Industry Online Support, follow these steps:

1. Navigate to the required module in the hardware catalog.
2. Right-click the module.
3. Select one of the following entries in the shortcut menu:
   - Information on product support
   - FAQs
   - Manuals

Result

The default browser set in the operating system is opened and the relevant page in the Siemens Industry Online Support is loaded.

See also

Enabling product support (Page 490)

8.1.1.13 Keyboard operation: Navigation in the editor

You can use shortcut keys in the network and device view to navigate between the components of the hardware and network editor and its objects.

Navigating between elements and functions

Function	Shortcut keys
Switch to the next lower selection level You can for example, use <Return> to switch from a selected rack to the lower selection level of the devices and modules that are snapped onto it. If a device is selected, you can use <Return> to switch to the lower selection level of the interfaces that are displayed on the device.	<Return>
Switch to the next higher selection level You can use <Esc>, for example, to switch from a selected interface to the higher selection level of the devices and modules. If a device is selected, you can use <Esc> to switch to the higher selection level of the rack.	<Esc>
Navigation between objects in the current selection level You can use the arrow keys to switch between the objects within a current selection level. To change the selection level, use the <Return> or <Esc> keys.	<Up arrow> <Down arrow> <Right arrow> <Left arrow>
Switches to the device view	<Ctrl+Shift+D>
Switches to the network view	<Ctrl+Shift+N>
Switches to the topology view	<Ctrl+Shift+T>
Switch between editor elements Use the <Tab> key to switch from one editor element to the next element. Use <Shift +Tab> to switch to the previous element. You can switch, for example, between the graphical view, Speedy Splitter, table view or underlying tabs.	<Tab> <Shift+Tab>
Switch between tabs Use the <Ctrl+Tab> keys to switch from one tab to the next tab on the right. Use <Ctrl +Shift+Tab> to switch to the next tab to the left. You can use these keys, for example, to switch between the device view, the network view and the topology view.	<Ctrl+Tab> <Ctrl+Shift+Tab>

Opening elements and functions

Function	Shortcut keys
Opening the online and diagnostics view When a device is selected, <Ctrl+D> opens the online and diagnostics view for the selected device.	<Ctrl+D>
Opening the download to device dialog When a device is selected, <Ctrl+L> opens the advanced download dialog.	<Ctrl+L>
Add new device <Ctrl+N> opens the dialog for adding a new device.	<Ctrl+N>
Opens the "Hardware catalog" task card	<Ctrl+Shift+C>
Opens "Online Tools" task card	<Ctrl+Shift+O>

See also

Keyboard operation in the TIA Portal (Page 308)

8.1.1.14 Keyboard operation: Editing objects

You can execute some of the functions of the network and device view directly with a combination of keyboard and mouse in the hardware and network editor. The keyboard operation in tables (Page 308) corresponds to standard characteristics. Here you find the keyboard operation for the graphic work area of the network and device view.

General keyboard operation

Function	Shortcut keys
Zoom in on view in frame Drag a frame in the graphical view in order to correspondingly change the size of the view.	<Ctrl+Space> + pressed mouse button
Move view Move the mouse pointer in order to move the view.	<Space> + pressed mouse button
Cancel current operation	<Esc>
Separate connection Use <Esc> or a double-click to exit connection mode when dragging a connection.	<Esc> or double-click
Zoom in graphic view The enlargement or reduction depends on the direction of rotation.	<Ctrl> + turn mouse wheel

Selected objects

Function	Shortcut keys
Select object	Mouse click
Cut an object The selected object is copied to the clipboard and deleted from the graphical view.	<Ctrl+X>
Copy object The selected object is copied to the clipboard.	<Ctrl+C>
Paste object The object from the clipboard is inserted into the selection.	<Ctrl+V>
Delete selected object	<Del>
Select several objects 1 You can add several objects to the selected objects by clicking on them individually. Alternatively, you can use <Shift> + pressed mouse key to drag a frame around the objects that are to be selected.	<Shift> + click
Select several objects 2 You can add several objects to the selected objects by clicking on them individually. Alternatively, you can use <Shift> + pressed mouse key to drag a frame around the objects that are to be selected. When holding down the <Ctrl> key, you can use a mouse click to deselect selected objects.	<Ctrl> + click

Function	Shortcut keys
Move selection When the mouse button is pressed, you can drag devices or modules to allowed slots on a rack.	Mouse button pressed
Copy selection Using <Ctrl> + pressed mouse button you can drag devices and modules to allowed slots on a rack. This copies the devices or modules.	<Ctrl> + pressed mouse button

8.1.2 Configuring devices

8.1.2.1 Basics

Introduction to configuring hardware

To set up an automation system, you will need to configure, assign parameters and interlink the individual hardware components. The work needed for this is undertaken in the device and network view.

Configuring

"Configuring" is understood to mean arranging, positioning, and networking devices and modules within the device or network view. Racks are represented symbolically. Just like "real" racks, they allow you to plug in a defined number of modules.

An address is automatically assigned to each module. The addresses can be subsequently modified.

When the automation system is started, the CPU compares the configuration that is preset by the software with the actual configuration of the system. Possible errors can be detected and reported straight away.

Assigning parameters

"Assigning parameters" is understood to mean the setting of properties of the components used. Parameter assignment is carried out for hardware components and for data exchange settings:

- Properties of modules with assignable parameters
- Settings for data exchange between components

The parameters are loaded to the CPU and transferred to the corresponding modules when the CPU starts up. Modules can be replaced with ease since the assigned parameters are automatically loaded into the new module during startup.

Adjusting the hardware to the project requirements

You need to configure hardware if you want to set up, expand or change an automation project. To do this, add hardware components to your configuration, connect them to existing components, and adapt the hardware properties to the tasks.

The properties of the automation systems and modules are preset in such a way that in many cases there is no need for additional parameter assignment. Parameter assignment is however needed in the following cases:

- You want to change the default parameter settings of a module.
- You want to use special functions.
- You want to configure communication connections.

See also

Changing properties of the modules (Page 1585)

Using existing configurations

Open existing projects

When opening existing projects, an automatic check is made to determine if the appropriate software is installed for all modules used within the project. If you try to open a project with modules that are not supported by the current scope of the installation of the TIA portal, a message appears on opening the project informing you of the missing software components. If the software components are not absolutely required to open the project, the project can still be opened.

Reaction to missing software components

Projects that contain modules not supported by the current scope of the installation react as follows:

- Display the modules on the GUI
  - The non-supported modules are displayed in the project navigation with all of their nested objects. However, the modules themselves cannot be processed in editors or in inspector windows. When possible, a replacement module is used that best matches the original module. Replacement modules are indicated by an exclamation mark.
  - Display of properties in tables is limited. This applies in particular to the display of network parameters, such as the IP address.
- Functional limitations
  - Non-supported modules cannot be printed out or compiled.
  - An online connection cannot be established to the module. It is therefore also impossible to download.
  - To change the device type, the device must first be deleted and then re-inserted. The "Change device type" function is not supported.
  - Copying and inserting nested objects, such as modules, is possible although the device itself cannot be copied and inserted.
  - The network configuration cannot be changed with replacement modules within the network view.
  - Cross-references can be displayed. However, the cross-references only reflect the state last saved within the project because on online comparison to the original module cannot be made.

See also

Opening projects (Page 334)

General slot rules

Introduction

Specific slot rules apply to each automation system and module.

If you select a module from the hardware catalog in the device view, all possible slots for the module selected are marked in the rack. You can only drag modules to marked slots.

If you insert, move or swap a module, the slot rules are also applied.

Consistency

Some slot rules depend on how the environment is configured. This means that you can sometimes plug modules into the rack although this would result in inconsistencies at the current time. If you change the configuration, for example by selecting different modules or module parameter settings, you can make the configuration consistent again.

In cases where inserting a module results in an inconsistency that can be corrected, this will be permitted. A consistency check is run when the configuration is compiled. Inconsistencies are displayed as alarms in the Inspector window under "Info". Based on the result of the consistency check, you can revise your configuration and make it consistent again.

Rules for arranging modules

The following rules apply generally to modules in racks:

- You can only plug modules into a rack.
- You can only plug interface modules into a module.
- You can only use modules of the same product or system family in one rack.

There are also other special rules for some modules:

- Can only be inserted in certain slots
- Insertion depends on other modules, CPUs or settings
- Limited number of uses in a rack

Browsing the hardware catalog

Introduction

Use the "Hardware catalog" task card to select the hardware components you want for a hardware configuration. Use the hardware catalog to select the interconnectable hardware components in the network and topology view and to select the modules you want in the device view.

Context filter

You can use the "Filter" option of the hardware catalog to restrict the number of displayed hardware components and the number of hardware components that can be found by searching.

If you select the filter, only those components are displayed that can be selected currently in the hardware catalog. If the do not select the filter, the entire hardware catalog is displayed.

If you switch between the various views, the view of the filter objects is adapted to the current context.

Search options

You can use the search function to search for specific entries in the hardware catalog. Note the following rules when entering search terms:

- No distinction is made between upper and lower case text.
- Dashes and blanks are ignored during the search.
- The search function considers parts of a search term.
- Several search terms must be separated by a space

You start the search from an object highlighted in the hardware catalog and either search upwards or downwards.

Symbol	Meaning
	Downwards search
	Upwards search

Browsing the hardware catalog

If you want to browse the hardware catalog, proceed as follows:

1. Click in the entry field of the search function
2. Enter a search term. The search includes the following elements:
   – Name of device or module
   – Order number (MLFB)
   – Description in "Information" pane
3. Click on either the "Downwards search" or "Upwards search" buttons.

Note

To ensure the right search direction, note which point you have marked in the hardware catalog. To browse the entire catalog, click on the topmost object of the hardware catalog and start the search once you have entered the search term by clicking "Downwards search".

The first match with the search term found is displayed as the result. For more search results, again click on the "Downwards search" or "Upwards search" button.

Observe the context filter of the hardware catalog. If this is selected, the search in the hardware catalog is restricted to the displayed inserted hardware components.

See also

Hardware catalog (Page 488)

Working with racks

Introduction

To assign modules to a device, you need a rack, for example a mounting rail. Secure the modules on the rack and connect these via the backplane bus with the CPU, a power supply or other modules.

Creating a rack

If you insert a device in the network view, a station and a rack suitable for the device selected are created automatically. The rack and slots available are displayed in the device view. The number of slots available again depends on the type of device used.

Rack structure

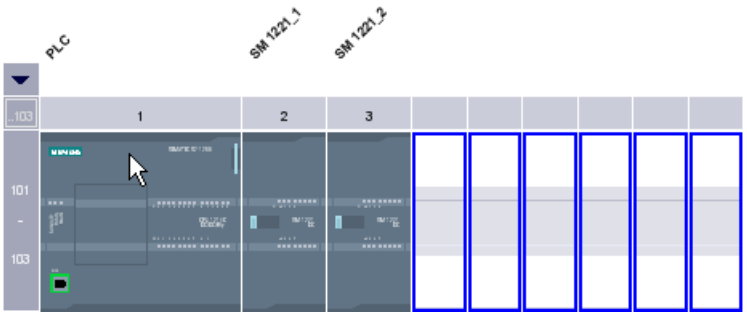
A rack always contains the device that has been inserted in the network view. The device is permanently assigned a slot which will depend on the type of device in question. There are additional slots on the right of the device and, if necessary, on left of the device; slot numbers are located above slots in which devices are plugged.

A corresponding short description is displayed above the plugged devices and modules. You show or hide this short description via the toolbar under "View" with the command "Display module titles" or the corresponding symbol in the toolbar of the device view (Page 479).

Symbol	Meaning
	Show module titles

When modules are selected in the hardware catalog, all the slots permitted for this module are marked. This allows you to see immediately the slot into which the selected module can be inserted.

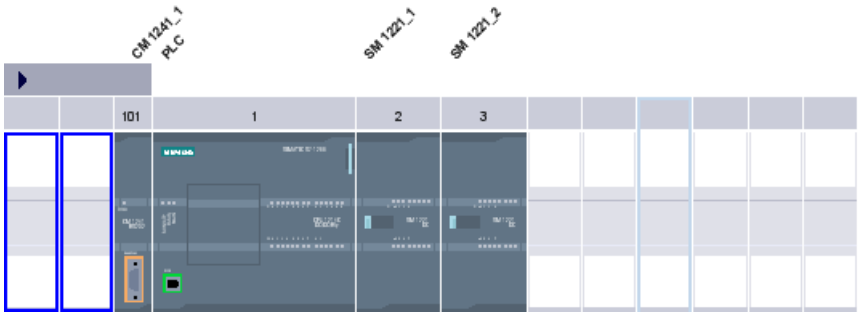
In the following screenshot, a signal module has been selected in the hardware catalog for a partially filled S7-1200 rack:



Since slots 101-103 are reserved for communications modules, only the other free slots are shown as available slots.

You can expand and collapse the front group of slots using an arrow symbol above the expandable slot. When the group of slots is collapsed, the first and last of the group's slot numbers are displayed.

The following figure shows the expanded slot group:



Groups of slots into which modules have already been plugged cannot be collapsed.

Multiple selection of modules and slots

There are various ways of selecting several modules or slots:

- By pressing <Shift> or <Ctrl>, you can select several modules or slots at the same time.
- Click outside the rack and then hold the mouse button and drag a frame to include the modules or slots you want to select.

Objects in the device view

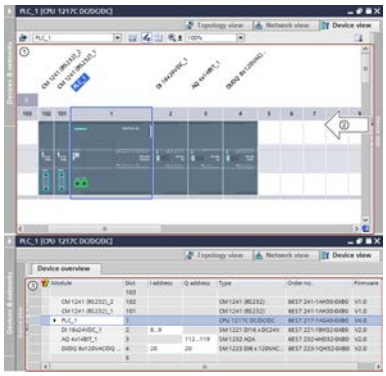
A graphic display of the rack and the devices plugged into it is shown in the left area of the device view. You can see the device overview in the right area of the device view. The device overview is a table showing you the most important information about the modules inserted in the rack. Both areas are displayed in a window. The size of both areas is adjusted using a divider. You can also use it to display one or the other area in full or to hide them.

Structure and content of device view

The offline configuration of the devices in the rack are displayed in the graphic device view. This is a symbolic representation of the configuration on the real rack.

The rack configuration is displayed as a table in the device view. Each line in the table contains the information for assigning a slot.

The following screenshot shows the device view with the configuration of a SIMATIC S7-1200 CPU.



① Graphic view of the allocation of the rack by the CPU and various modules in slots 1 to 4, as well as 101 and 102.

② You can use the divider to adjust the proportion of the device view between the left area (graphic view) and the right area (device overview). If you click on the arrows, you can quickly switch the division of the separate areas.

③ Device overview with a tabular representation of the slots of the rack and the plugged components.

Each line in the device overview represents one slot. The key information for each slot is displayed in the various columns:

Column	Meaning
Module	Name of module, can be edited in any way
Slot	Slot number
I address	Input address area, can be edited in any way
Q address	Output address area, can be edited in any way
Type	Catalog name of module
Order no.	Module order number
Firmware	Firmware version of module
Comments	Optional comments

See also

Device view (Page 479)

Area for unplugged modules

In some cases, the modules for a hardware configuration are not assigned a slot for short periods. Such unplugged modules are moved to the area of unplugged modules, a special area in the device view.

Adding modules to the storage area

The modules that, for example, should be assigned to a device using a copy action but for which the corresponding rack does not have a free compatible slot, are moved automatically into the area of unplugged modules.

Under the following conditions, modules are automatically added to the area of unplugged modules:

- In the network view, a module is moved to a device but the rack does not have a compatible free slot.
- In the device view, a module is moved or copied from the rack, the hardware catalog or the project tree straight into the storage area.

CPs and FMs that occupy a network resource can be moved into the area of unplugged modules but will lose the network resources they have been assigned.

You can add modules to the area of unplugged modules by means of drag-and-drop, for example. To do this, the area must be opened.

Using the area of unplugged modules

Use the corresponding button to open the area of unplugged modules.

You can find the area of unplugged modules in the device view.

You open the area of unplugged modules with the respective icon in the toolbar of the device view (Page 479).

Icon	Meaning
	Open area of unplugged modules

Note

To free up slots, move modules from your configuration into the storage area and plug the modules you want from the storage area into the freed up slots.

You can use this approach to temporarily move modules whose parameters have already been assigned out of the configuration without deleting them.

Treatment of modules in the storage area

The following rules apply to modules in the storage area:

- The modules appear in the project tree under the corresponding device in the "Local modules" folder.
- The modules retain all settings and parameters previously assigned.

- The modules are not taken into account during loading to a target system. This means that a consistency check is not undertaken for modules in the area of unplugged modules.
- Using the context menu, the modules can be copied, cut, or deleted, for example.

8.1.2.2 Configuring individual devices

Selecting a CPU

Introduction

Select a CPU from the hardware catalog and place it, together with a rack, in the network view. On this device drag the desired modules from the hardware catalog; they are arranged automatically on the rack.

Selecting the components in the hardware catalog

Each hardware component is displayed as a folder in the hardware catalog. When you open this folder you will see the different versions of the selected hardware component with its respective order numbers.

There will be an example of how to set up a CPU with a rack in network view.

Requirement

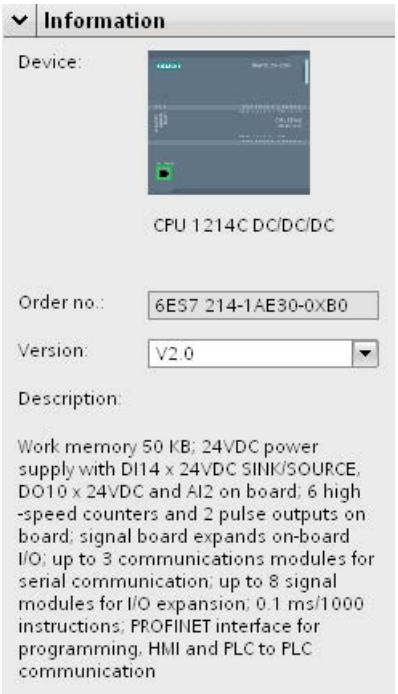
- The hardware catalog is open.
- You must be in the network view.

Procedure

To select a CPU from the hardware catalog, proceed as follows:

1. In the hardware catalog navigate to the folder with the desired CPUs.
2. Open the folder with the desired CPU type; you will see all order numbers for the selected CPU type.

3. Click on a CPU order number to get information about the selected CPU under "Information" pane.



4. Set up the CPU and a rack. You have the following options:
   - Use drag-and-drop to drag the CPU from the hardware catalog into network view.
   - Use Copy & Paste to copy the CPU to the network view.
   - Double-click the CPU entry in the hardware catalog.

See also

Browsing the hardware catalog (Page 497)

Adding a device to the hardware configuration (Page 504)

Inserting a module into a rack (Page 507)

Working with racks (Page 498)

Creating an unspecified CPU (Page 506)

Adding a device to the hardware configuration

Introduction

There are various ways of adding a connectable device from the hardware configuration in the network view and the topology view:

- Command "Add new device" in the project tree
- Double-click device in hardware catalog

- Drag-and-drop from the hardware catalog in network view or in topology view:
  - Text entry from the "Catalog" pane
  - Preview graphic from the "Information" pane
- "Add > Device" command from menu bar in network view or topology view
- Shortcut menu of a device in the hardware catalog for copying and pasting

A suitable rack is created along with the new device. The selected device is inserted at the first permitted slot of the rack.

Regardless of the method selected, the added device is visible in the project tree and the network view of the hardware and network editor.

Adding device using the project tree

To use the project tree to add a device to the hardware configuration, proceed as follows:

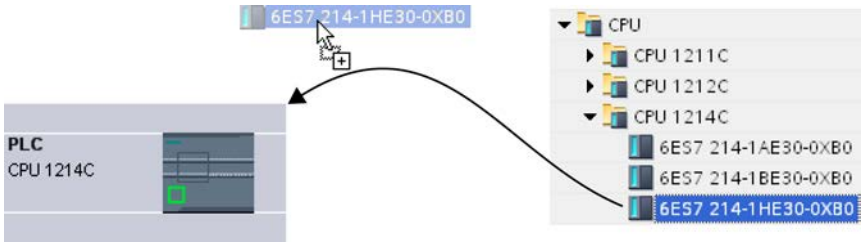
1. Click on the command "Add new device" in the project tree.
   The "Add new device" dialog box opens.
2. Display the required device in the tree structure:
   - Go to required device in the tree structure.
   - Enter a device name in the entry field.
3. Select the required device from the tree.
   More information about the device presently selected is displayed on the right-side of the dialog box.
4. If necessary, set the firmware version using the drop-down list in the dialog box.
5. Select the "Open device view" check box if you want to change to the device view immediately after adding the device.
   There you can immediately continue with device configuration and equipping the rack.
6. Click on "OK" to add the device selected.
   The dialog box closes.

Adding device from the hardware catalog

To add a device to the hardware configuration using the hardware catalog, proceed as follows:

1. Open the network view or the topology view.
2. Open the hardware catalog.
3. Go to the required device in the hardware catalog.
4. Click on the chosen device to select it.

5. If necessary, set the firmware version using the drop-down list in the hardware catalog.
6. Drag the device to the network view or the topology view.



You have now placed the device in the network view or in the topology view. The displayed rectangle (in other words "Station") symbolizes the plugged device together with its rack and any lower-level modules. Double-click on the device or station to open the device view and view the new rack and inserted device. In the next steps, you can configure the device in the device view and equip the rack with modules.

See also

Network view (Page 477)

Creating an unspecified CPU (Page 506)

Topology view (Page 482)

Creating an unspecified CPU

Introduction

If you have not yet selected a CPU but have already started programming or want to use an existing program, you have the option of using an unspecified CPU. You can also adjust some settings with unspecified CPUs. The setting options are restricted to parameters that all CPUs of the same CPU family have in common.

Creating an unspecified CPU in the portal view

To create an unspecified CPU in the portal view, follow these steps:

1. Now, click one of the following options:
   – "Devices & networks > Add new device"
   – "PLC programming" > "Device" button
2. For a device family, select an unspecified CPU from the tree structure of the "Add new device" dialog.
3. Click on "Add".

An unspecified CPU is created and the device view for this CPU appears.

Further options for creating unspecified CPUs

In the project view, you can create unspecified CPUs like specified CPUs:

- Using the "Add new device" button in the project tree
- In the "Hardware catalog" task card

You can also use these methods to create multiple unspecified CPUs.

Specifying unspecified CPUs

You have two options for specifying unspecified CPUs:

- Use drag-and-drop to assign an existing CPU from the hardware catalog to an unspecified CPU by means of module replacement (Page 513).
- Select a selected, unspecified CPU and then the menu command "Online > Hardware detection" and assign a CPU identified online. For this purpose, you assign an IP address using the "Add address for PG/PC" button.

Note

If you want to go online after the hardware detection, you have to first download the detected configuration to your project; otherwise, an error may occur due to inconsistent configurations. After hardware detection, the order numbers of the CPU in the project and the actually existing CPU are identical, but their parameters are not. The parameters of the CPU in the project have the default values; the parameters of the actually existing CPU have the values set by you.

See also

Selecting a CPU (Page 503)

Adding a device to the hardware configuration (Page 504)

Inserting a module into a rack

Introduction

Once you have added devices from the hardware catalog to your configuration in network view, you can add modules to the devices. There are various ways of adding a module to a rack in the device view:

- If there is an available valid slot, double-click a module in the hardware catalog.
- Use drag-and-drop to move the module from the hardware catalog to an available valid slot in the graphic or table area:
  - Text entry from the "Catalog" pane
  - Preview graphic from the "Information" pane
- Select "Copy" in the shortcut menu for a module in the hardware catalog and then select "Paste" in the shortcut menu on an available valid slot in the graphic or table area.

To access the device view from the network view, double-click a device or station in the network view or select the Device view tab. The device view contains an illustration of the device selected within a rack. The graphic illustration of the rack in the software corresponds to the real structure, i.e. you can see the same number of slots as exist in the real structure.

Note

You can also move a module to a rack in the network view. The filter function for the hardware catalog must be deactivated in this instance. The module is automatically plugged into a free and permitted slot. If there are no slots available, the module will be moved to the area of unplugged modules (Page 501).

Equipping a rack

Arrange the modules on a rack according to the applicable slot rules.

After a module has been inserted in a rack with an already inserted CPU, the address areas are checked automatically so that addresses are not assigned twice. After it has been inserted, each module then has one valid address range. To do so, DP slaves and IO devices must be networked with a CPU via the corresponding DP master or IO system.

Requirements

- You are in the device view.
- The hardware catalog is open.

Adding module from the hardware catalog

How to insert a module from the hardware catalog into a rack is illustrated based on the example of a signal module. To do this, follow these steps:

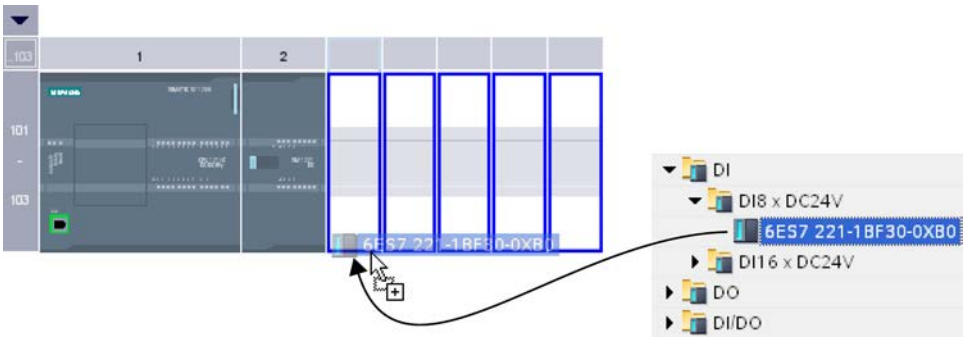
1. Go to the required module board in the hardware catalog.

   Note

   If you activate the filter function of the hardware catalog, only those modules which match the selected device type will be displayed.

2. Select the chosen module.

3. If necessary, set the firmware version using the drop-down list in the hardware catalog.
4. Drag the signal module to a free slot in the rack.



You have now inserted the digital signal module in a slot in the rack. Repeat these steps with the other modules.

The name of the module is displayed above the inserted modules. You can activate or deactivate module labeling in the menu bar with "View > Show module labels".

Inserting module

You can also drag modules and drop them between modules that have already been inserted. To do this, drag a module above and between the two existing modules while holding down the mouse button.

A mouse pointer appears. When you release the mouse button, all modules plugged to the right of the pointer are moved one slot to the right. Any redundant modules are moved to the area of unplugged modules. The new module is plugged at the point of the freed up slot.

See also

Device view (Page 479)

Area for unplugged modules (Page 501)

General slot rules (Page 496)

Rules for S7-300 modules (Page 864)

Rules for S7-400 modules (Page 865)

Deleting a hardware component

There are various ways of deleting hardware components. Deleted hardware components are removed from the system and assigned addresses made available again.

Rules

- CPUs or modules from the rack and from the area of unplugged modules can be deleted.
- When a rack is deleted in the device view, the plugged hardware components are moved to the area of unplugged modules.

Procedure

Proceed as follows to delete a hardware component:

1. Select the hardware components you want to delete.
   - Network view: Select devices or network relevant hardware components in the graphic view or in the network view.
   - Device view: In the graphic view or device overview, select racks or modules in racks or in the area of unplugged components.
   - Topology view: Select devices or hardware components with Ethernet interfaces in the graphic view or in the topology view.
   - Project tree: Select devices or individual hardware components from the tree structure.
2. Select "Delete" from the shortcut menu or press <Del>.
   If the "Delete" menu item is unavailable, your selection contains at least one component that cannot be deleted.

The selected hardware components are deleted.

Note

Deleting hardware components may result in inconsistencies in the project, such as violation of slot rules. Inconsistencies are reported during the consistency check. Correct the inconsistencies by taking appropriate action, for example, ensuring compliance with the slot rules.

See also

Keyboard operation: Editing objects (Page 493)

Copying a hardware component

You can copy hardware components in the device or network view. Copied hardware components are stored on a clipboard and can be pasted at another point from this clipboard. Copied stations are pasted as new stations in the network view. Copied devices and modules can be pasted into existing racks in the network and device view.

Rules

- Single objects as well as several objects can be copied at the same time.
- Modules inserted in the rack and in the area of unplugged modules can be copied.
- You can only copy devices and modules to free and valid slots in keeping with the slot rules.
- Racks with a CPU inserted cannot be copied individually, but only as complete units along with all inserted hardware components.

Procedure

Proceed as follows to copy a hardware component:

1. Select the hardware components you want to copy.
   - Device view: Select the module in a rack or put it in the area of unplugged modules.
   - Network view: Select the station or the relevant hardware component from the network view.
   - Project tree: Select the station or module.
2. Select "Copy" from the shortcut menu or press <Ctrl+C>.
   If the "Copy" menu item is unavailable, your selection contains at least one component that cannot be copied.
3. Select the location at which the content of the clipboard is to be pasted.
   - Device view: Select a free slot in the rack or area of unplugged modules.
   - Network view: Select a station where you want to insert devices or modules or move the mouse pointer to a free location in the network view to paste a copied station or a hardware component relevant to the network view.
4. Select "Paste" from the shortcut menu or press <Ctrl+V>.
   If the "Paste" menu item is unavailable, the clipboard is empty or contains at least one component that cannot be pasted at this point.

The selected object is pasted at the chosen point.

Once you have selected a station where you want to insert a module in the network view, the module is inserted in the first free and valid slot. If no free, valid slots are available, the object is inserted in the area of unplugged modules.

Note

You can also copy a module from one device to another:

To do so, copy a module in the hardware and network editor, select a different device in the network view or the drop down list of the device view and insert the module.

You can insert the copied object directly in a slot or place it in the area of unplugged modules in the device view. If you add the copied object in the network view of a device or a station, it will be inserted in the first available slot.

If there is no slot available for the object, it is automatically placed in the area of unplugged modules (Page 501).

Note

You can use <Ctrl> and drag-and-drop to directly copy a selected hardware component.

See also

Keyboard operation: Editing objects (Page 493)

Moving a hardware component

You can move hardware components in the device or network view.

Rules

- You can move devices and modules from the rack and the area for unplugged modules taking the slot rules into consideration.
- CPs can be moved in the network view. The CP is plugged in a free and valid slot in the target device. If there are no free slots available, the CP to be inserted is moved to the area for unplugged modules.
- In the network view, CPU and slave head modules can be moved between the devices; depending on CPU type also within the rack.

Note

Moved CPs are disconnected from their network but keep their network parameters and address. If you reconnect the CP to the network and its address has been assigned, use a dialog to assign a new unique address to the CP.

Procedure

Proceed as follows to move a hardware component:

1. Select the hardware component you want to move.
   - Device view: Select the module in a rack or put it in the area of unplugged modules.
   - Network view: Select the hardware component of relevance to the network view.
2. Select "Cut" from the shortcut menu or press <Ctrl+X>.
   If the "Cut" menu item is unavailable, your selection contains at least one component that cannot be cut.
3. Select the location to which the cut object is to be moved.
   - Device view: Select a free slot in the rack or area of unplugged modules.
   - Network view: Select a station where you want to insert devices or modules.
4. Select "Paste" from the shortcut menu or press <Ctrl+V>.
   If the "Paste" menu item is unavailable, the clipboard is empty or contains at least one component that cannot be pasted at this point.

The selected hardware component is moved to the target. If the hardware component being moved is a networked object, it is disconnected from the network.

Note

You can use drag-and-drop to directly move a selected hardware component.

See also

Keyboard operation: Editing objects (Page 493)

Replacing a hardware component

You can replace hardware components with others. This, for example, allows you to replace unspecified CPUs (Page 506) with available CPUs from the hardware catalog.

Rules

You can only replace hardware components if they support module replacement and if the two components are compatible.

Procedure

To replace one module with another, proceed as follows:

1. Select the module you want to replace.
2. Open the shortcut menu:
   - If the "Replace device" entry is enabled, the module can be replaced.
   - If the "Replace device" entry is disabled, a module cannot be replaced.
3. Click on "Replace device" in the shortcut menu. The "Replace device" dialog box appears.
4. Under "New device" in the tree structure, select the module with which you want to replace your current module.
5. Click "OK".

The existing module is replaced by the new one.

As an alternative, you can take a module by dragging it from the hardware catalog to the module you are replacing. If the module can be replaced by the selected module, this is indicated by the mouse pointer symbol.

Editing properties and parameters

Once you have inserted hardware components in your rack, you can edit their default properties, for example parameters or addresses in the network or device view.

Requirement

You are in the device view.

Note

You can also edit properties and parameters in the network view. In the graphic network view, you have access to the network-related hardware components and the station. You can access modules and hardware components not displayed in the graphic network view using the table network view.

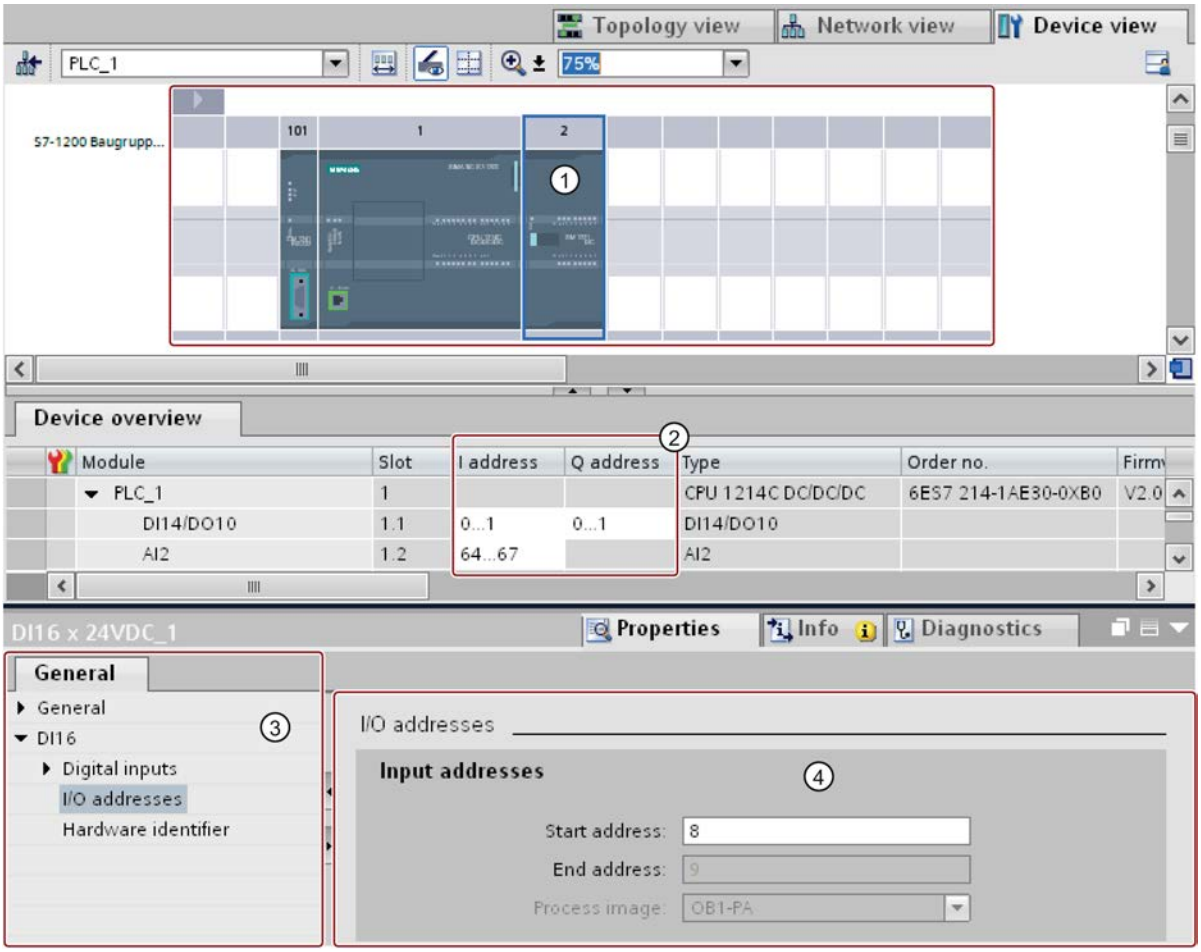
Procedure

To change the properties and parameters of the hardware components, proceed as follows:

1. In the graphic view, select the CPU, module, rack or interface you want to edit.
2. Edit the settings for the selected object:
   - Use the table view to edit addresses and names, for example.
   - In the Inspector window additional setting possibilities are available in "Properties".

Note that modules can only be fully parameterized if they are assigned to a CPU. Therefore, PROFIBUS or PROFINET interfaces modules must first be networked with the CPU or a centrally inserted communication module so that they form a master system or IO system. Only then is it possible, for example, to edit the addresses of the distributed components that are inserted.

Example of changing settings



① Selection of a module
② Editing option for addresses in the device overview
③ Selection options in the inspector window
④ Editing option for addresses in the inspector window

See also

Inspector window (Page 487)

Input and output addresses in the address overview

Introduction

The currently used input and output addresses can be displayed in the address overview in a table form. The address overview can be found in the Inspector window under "Properties" of the CPU.

Design of the address overview

With the different check boxes, you can set which objects should be displayed in the address overview:

- Inputs: Display of the input addresses
- Outputs: Display of the output addresses
- Address gaps: Display of open address spaces
- Slot: Display of the slot number

The following information is typically shown in the address overview:

Table header	Meaning
Type	Indicates whether the area is an input address area or an output address area.
Addr. from	Start address of the address range.
Addr. to	End address in the address range.
Module	Module using the address area.
PIP	Number of the process image partition. Shows whether the cyclic OB is within the process image partition.
OB	Organization block that is assigned to the process image. This column is not available for every CPU.
DP	Number of the master system. You can use the number to determine which slaves are assigned to a master. The value in brackets specifies the PROFIBUS address of the hardware component.
PN	Number of the IO system. The value in brackets stands for the device number of the hardware component.
Rack	Number of the rack on which the hardware component is inserted.
Slot	Number of the slot in the rack in which the hardware component is inserted.

See also

Specifying input and output addresses (Page 938)

Update module version

Explanation of terms

The terms "Module version" and "Firmware version" are explained in more detail in the following section.

- Module version: The specific version of the configuration software from which the module description stems.
  Example: V11.0.0.0
- Firmware version: The version of the firmware of the module whose parameters are assigned offline
  Example: V2.0

Requirement

- You have created a device configuration.
- You have installed an update or an optional package at a later date. As a result of this installation, the module version of at least one module type was updated in the hardware catalog, whereby the new version is incompatible with the previous version.
- You have used such modules in your device configuration and want to use the modified or added properties.

Procedure

Perform the following step for each affected module type.

1. Select the affected module in the device view.
2. In the Inspector window, go to "Properties > General > Catalog Information". Click the "Update module version" button there.
3. In the query that then appears, specify whether you want to update the module version only for the selected module or for all modules of this type in the current project.

Result

The selected modules are replaced by the same modules with updated module version in the the current project.

In which cases is it unnecessary to update the module version?

An updating of the module version is not necessary in the following cases:

- You do not want to use the modified or added properties of the modules.
- You open an existing project with a version of the configuration software that is more recent than the version with which you created the project, and the system automatically performs a project conversion, for example, from TIA Portal V12 to V13. In this case, all older module versions are automatically adapted.

8.1.2.3 Comparing devices

Basics of device comparison

Function

You have the option of comparing the hardware components of two devices allowing you to identify any differences. You can perform an offline/offline comparison for this purpose. The devices to be compared can be from one project or different projects.

You can compare the central as well as the distributed I/O. The devices to be compared can be assigned either automatically or manually. The automatic assignment of central I/O is based

on the slot number. With the distributed I/O, the automatic assignment can be made according to the following criteria:

- Assignment using the address/ID: The assignment is made using the addresses or IDs of the devices. This criterion is suitable for comparing devices within a project.
- Assignment using the name: The assignment is made based on the device names. This criterion is suitable for comparing devices in different projects.

You can either specify the assignment yourself or let the system decide. In the latter case, the system selects the assignment itself depending on the context.

See also

Basics of project data comparison (Page 363)

Overview of the compare editor (Page 365)

Making a device comparison (Page 518)

Making a device comparison

Procedure

Follow the steps below to compare devices:

1. Select a device in the project tree that allows offline/offline comparison.
2. Select the "Compare > Offline/offline" command in the shortcut menu.
   The comparison editor opens and the selected device is displayed in the left area.
3. Open the "Hardware" tab.
4. Drag an additional device to the drop area in the right-hand pane.
   All existing objects of the selected devices are displayed depending on the settings of the comparison editor in the "Hardware" tab and an automatic comparison is made. You can identify the status of the objects based on the symbols in the comparison editor.
5. If you want to change the matching criterion, click on the arrow of the "Available matching criteria" button in the toolbar. Then, select the matching criterion you want to use.
6. If you want to make a manual comparison, click the button for switching between automatic and manual comparison above the status area. Then select the objects you want to compare.
   The properties comparison is displayed. You can see the status of the objects based on the symbols.

See also

Basics of device comparison (Page 517)

Overview of the compare editor (Page 365)

8.1.3 Configure networks

8.1.3.1 Networking devices

Communication and networks

Communication between devices

The basis of all types of communication is always a previously configured network. The network configuration provides the necessary requirements for communication:

- All the devices in a network are provided with a unique address
- Communication of the devices with consistent transmission properties

Network configuration

The following steps are necessary when configuring networks:

- Connect devices to subnet
- Specify the properties/parameters for each subnet
- Specify the device properties for every networked module
- Download configuration data to the devices to supply interfaces with the settings resulting from the network configuration
- Document the network configuration

For Open User Communication, creating and configuring a subnet is supported by the assignment of connection parameters.

Relation between network configuration and project

Within a project, subnets and their properties are managed. Properties result mainly from adjustable network parameters and the quantity and communication properties of the connected devices.

The devices to be networked must be in the same project.

Subnet name and subnet ID

Within the project, subnets are clearly identified by a subnet name and ID. The subnet ID is saved in all components along with interconnectable interfaces. Components can then be clearly assigned to a subnet even after uploading into a project.

Networking options

In the project, you can create and network devices with components capable of communication. The following basic options are available for networking the devices:

- You link the interfaces of the components capable of communication with one another. A new subnet is created suitable for the type of interface.
- You connect the interface of the devices capable of communication with a new or existing subnet.
- You create an Open User Communication connection. When you assign parameters to the connection for Open User Communication, a subnet is created automatically between the communication partners.
- You use the graphic connection configuration to configure connections; missing networks are hereby recognized and are created either automatically or via dialog.

Due to the different tasks of the devices or the span of the plant, you may need to use several subnets. These subnets are managed in a project.

Networking devices in the network view

Options

In the graphic network view, you have an overview of the subnets of the entire system in the project. You can use the tabular network overview for additional support.

Depending on the starting situation, there are various ways of undertaking configuration to network the interface for a component capable of communication. The procedures are described in the following section:

- Creating an individual subnet
- Creating several subnets at one time
- Connecting two target devices via a new subnet
- Connecting devices to existing subnet
- Selecting an existing subnet from a list
- Automatic networking during the configuration of the connection;
  See also: Creating a new connection - overview (Page 600)

Possible starting situations are:

- A suitable subnet is not yet available.
- The subnet with which you want to connect the component already exists.

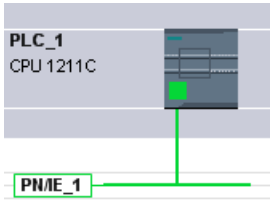
Procedure - creating a single subnet

To create a subnet and to connect it to an interface, proceed as follows:

1. Select the interface of a CPU / a CP.
2. Select the "Create subnet" command in the shortcut menu of the interface.

The selected interface is connected to a new subnet. Consistent address parameters are set automatically for the interface.

The following schematic shows an interface with outgoing line connecting to a subnet:



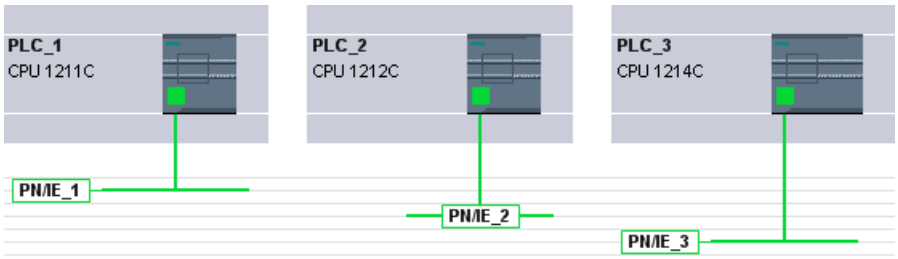
Procedure - creating several subnets at one time

To create several subnets at one time, proceed as follows:

1. Select several interfaces by clicking on them while pressing the <Ctrl> button.
2. Select the "Create subnet" command in the shortcut menu of the interface.

Each selected interface is connected to a new subnet. Consistent address parameters are set automatically for the interface.

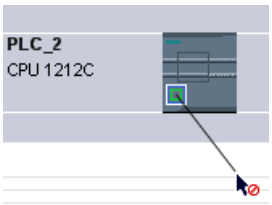
The following figure shows multiple subnets created by selecting multiple interfaces:



Procedure – Connecting two target devices via a new subnet

To connect an interface with another device via a subnet that does not yet exist, proceed as follows:

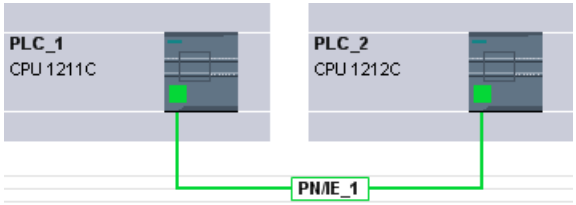
1. Position the mouse pointer over the interface for a component capable of communication requiring networking.
2. Click with the left mouse button and hold the button down.
3. Move the mouse pointer.
   The pointer now uses the networking symbol to indicate "Networking" mode. At the same time, the mouse pointer shows the lock symbol that will only disappear when the pointer is on a valid target.



4. Now move the pointer onto the interface of the target device. You can either keep the mouse button pressed or release it.
5. Now release the left mouse button or press it again (depending on previous action).

A new subnet is created. The interfaces are now connected via the new subnet. Consistent address parameters are set automatically for the interface.

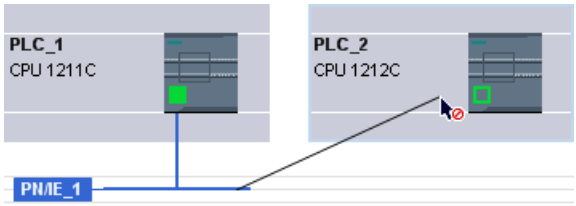
The following schematic shows two networked devices:



Procedure - Connecting devices to existing subnet

To connect an interface to an existing subnet, proceed as follows:

1. Position the mouse pointer on the interface of a communications-compliant component you want to network or on the existing subnet.
2. Click with the left mouse button and hold the button down.
3. Move the mouse pointer.
   The pointer now uses the networking symbol to indicate "Networking" mode. At the same time, the mouse pointer shows the lock symbol that will only disappear once the pointer is moved to a valid target.
4. Now move the mouse pointer to the existing subnet or to the interface to be networked. You can either keep the mouse button pressed or release it.



5. Now release the left mouse button or press it again (depending on previous action).

Result:

The interface and selected subnet are now connected. Consistent address parameters are set automatically for the interface.

Procedure - selecting an existing subnet from a list

To link an interface with a subnet that has already been created, proceed as follows:

1. Select the interface of a CPU.
2. Select the "Assign to new subnet" command in the shortcut menu of the interface.
   A list box containing the available subnets appears.
3. Select a subnet from the list.

The interface and selected subnet are now connected. Consistent address parameters are set automatically for the interface.

Tabular network overview

Meaning

The tabular network overview adds the following functions to the graphic network view:

- You obtain detailed information on the structure and parameter settings of the devices.
- Using the "Subnet" column, you can connect components capable of communication with created subnets.

Basic functions for tables

The network overview supports the following basic functions for editing a table:

- Displaying and hiding table columns
  Note: The columns of relevance to configuration cannot be hidden.
- Optimizing column width
- Sorting table
- Displaying the meaning of a column, a row or cell using tooltips.

Networking devices in the device view

Networking in the device view

In the device view, you can check and set all the parameters of the components belonging to a device and the interfaces in detail. Here you can also assign the interfaces to the subnets created in the project.

Requirements

- The subnet with which you want to connect an interface has already been created.
- If the subnet has not yet been created, change to the network view and make the settings required for networking.

Procedure - connecting to an existing subnet

To connect an interface to an existing subnet, proceed in the device view as follows:

1. Select the entire communications-compliant component or the interface to be networked. The properties of the selected interface or component are displayed in the Inspector window.
2. In the Inspector window, select the parameter group for the selected interface; for example, the "Ethernet addresses" parameter group for a PROFINET interface.
3. Select the subnet to be connected from the "Interface connected with" drop-down list.

The interface and selected subnet are now connected. Consistent address parameters are set automatically for the interface.

Procedure - creating a new subnet

To create a subnet and to connect it to the interface, proceed as follows in the device view:

1. Select the entire communications-compliant component or the interface to be networked. The properties of the selected interface or component are displayed in the Inspector window.
2. In the Inspector window, select the parameter group for the selected interface; for example, the "Ethernet addresses" parameter group for a PROFINET interface.
3. In "Interface connected with", click the "Add new subnet" button.

The interface is connected to a new subnet of the appropriate subnet type. Consistent address parameters are set automatically for the interface.

Checking or changing network parameters and interface parameters

Introduction

Communication between networked devices requires the following parameters to be configured:

- Network parameters
  Network parameters identify the network within the system configuration, for example, using a name.
- Interface parameters
  Interface parameters define specific properties of a component capable of communication. Addresses and transmission characteristics are set automatically and are consistent with the network parameters.

Note

Network parameters and interface parameters are usually set during networking such that communication can take place for numerous applications without the parameters having to be changed.

Procedure - checking or changing network parameters

Proceed as follows to check or change network parameters:

1. Enter the network view.
2. Select the subnet from the network view.
   You can see the network parameters in the "Properties" tab in the inspector window.
3. If necessary, check or modify the network parameters in the relevant group of parameters.

Procedure - checking or changing interface parameters

You can check and modify interface parameters in the network and device view.

Proceed as follows to check or change interface parameters:

1. Enter the network view or device view.
2. Select the interface.
   You can see the interface parameters in the "Properties" tab in the inspector window.
3. If necessary, check or modify the interface parameters in the relevant group of parameters.

Changing networkings

Introduction

You can cancel an interface's network connection or assign it to another subnet of the same subnet type.

Consequences

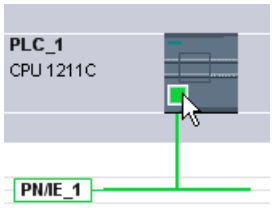
Depending on the version, a distinction should be made between:

- Canceling a network connection for an interface
  The configured parameters for the interface remain unchanged.
- Assigning a network connection to another subnet
  If the addresses in the assigned subnet are not unique, in other words, they already exist, they will be changed automatically to make them unique.

Procedure - disconnecting from a network

Proceed as follows to cancel the network connection for an interface:

1. Select the networked interface.



2. Select the "Disconnect from subnet" command in the shortcut menu of the interface.

The network connection is deleted, the interface addresses are, however, not changed.

Configured connections are retained; however these connections are marked red in the connection table because they are not networked. Specified connections remain specified.

See also

Networking devices in the network view (Page 520)

Copying, cutting or deleting subnets

Introduction

You can copy subnets as individual objects or copy them along with networked devices or other networks.

For example, you can create complex configurations to be used more than once in different variants within the project with no additional effort.

Effects on the copied subnet

Properties that have to be assigned explicitly within a project are re-assigned appropriately when the copied objects are copied.

For subnets, this means: The subnet ID and name are re-assigned to the copied subnet.

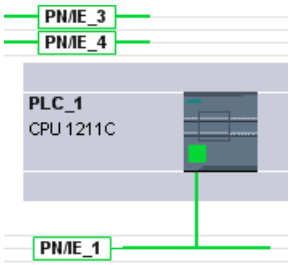
The configured properties are adopted in the copied subnet.

Procedure - copying a subnet

Proceed as follows to copy one or more subnets:

1. Select one or more subnets.
2. In the shortcut menu, select the "Copy" command.
3. Select the "Paste" command in the shortcut menu.

The copied subnets are shown as "orphaned" subnets in the top part of the network view.



Procedure - copying a subnet with connected devices

To copy one or more subnets with networked devices, proceed as follows:

1. Select one or more subnets with the connected devices, for example by drawing a lasso around them.
2. In the shortcut menu, select the "Copy" command.
3. Select the "Paste" command in the shortcut menu.

Complete copies of the subnets and connected devices are created.

Configured connections are adopted and remain within the copied devices. Connections to devices that have not been copied are interrupted and become unspecified.

MPI network configuration

Allocating MPI addresses

Note the following for devices with an MPI interface: All devices of a subnet must have different device addresses.

CPUs with MPI address ship with the default MPI address 2. Since you can only use this address once in the MPI subnet, you will have to change the default address in all other CPUs.

The following applies to devices with the order no. 6ES7 3xx-xxxxx-0AB0:

When planning the MPI addresses for several CPUs, you have to fill "MPI address gaps" for FMs and CPs with separate MPI addresses to prevent addresses being assigned twice.

Only when all the modules in a subnet have different addresses and your actual structure matches that of the network configuration produced, should you load the settings across the network.

Rules for assigning the MPI address

- Allocate the MPI addresses in ascending order.
- Reserve MPI address 0 for a programming device.
- You can link up to 126 (addressable) devices with one another in an MPI subnet; up to 8 devices at a transfer speed of 19.2 KB/s.
- All MPI addresses in an MPI subnet must be different.

You will find other rules relating to the structure of a network in the manuals for setting up automation systems.

PROFIBUS network configuration

PROFIBUS addresses

Rules for the network configuration

All the devices in a subnet must have a different PROFIBUS address.

Only when all the modules in a subnet have different addresses and your actual structure matches that of the network configuration produced, should you load the settings across the network.

You can connect devices to the PROFIBUS subnet that communicate via configured connections or that belong to a PROFIBUS DP master system.

You can find more information on configuring a DP master system in the following sections.

Requirements

The 121xC CPU is PROFIBUS compatible as of firmware version 2.0.

Rules for assigning PROFIBUS addresses

- Allocate the PROFIBUS addresses in ascending order.
- Reserve the PROFIBUS address "0" for a programming device.
- Allocate a unique PROFIBUS address between 0 and 126 for each device on the PROFIBUS network and/or for each DP master and each DP slave in the PROFIBUS network.
- There are modules with which the smallest address that can be set has to be greater than 1.
- All PROFIBUS addresses of a PROFIBUS subnet must be different.

You will find additional rules relating to the structure of a network in the manuals for setting up automation systems, for example SIMATIC S7-1200.

Note

PROFIBUS address "0"

Reserve PROFIBUS address "0" for a programming device that you will briefly connect up to the PROFIBUS network at a later date for servicing.

See also

What you need to know about PROFIBUS bus parameters (Page 529)

What you need to know about PROFIBUS bus parameters

Matching parameters to one another

The PROFIBUS subnet will only function without problem if the parameters for the bus profile are matched to one another. You should therefore only change the default values if you are familiar with how to configure the bus profile for PROFIBUS.

Note

It may be possible for the bus parameters to be adjusted depending on the bus profile. If the bus parameters cannot be adjusted, they are grayed out. The offline values of the bus parameters are always shown even if you are online and linked to the target system.

The parameters shown apply to the entire PROFIBUS subnet and are briefly explained below.

Activating cyclic distribution of the bus parameters

If the "Enable cyclic distribution of the bus parameters" check box is selected under "Bus parameters" while PROFIBUS subnet is selected in the Inspector window, the bus parameters are transferred cyclically during operation by the modules that support this function. This allows a programming device, for example, to be easily connected to the PROFIBUS in runtime.

You should deactivate this function:

- For a heterogeneous PROFIBUS subnet (or more accurately: when external devices are connected whose protocol uses the DSAP 63 for Multicast)
- When in constant bus cycle time mode (minimize bus cycle)

Bus parameters for the bus profile of PROFIBUS subnets

Bus parameter	Adjustable?	Limit values
Tslot_Init	Yes	Max. Tsdr + 15 <= Tslot_init <= 16.383 t_bit
Max. Tsdr	Yes	35 + 2*Tset + Tqui <= Max. Tsdr <= 1.023 t_bit
Min. Tsdr	Yes	11 t_bit <= Min. Tsdr <= MIN(255 t_bit, ... Max. Tsdr - 1, 34 + 2*Tset + Tqui)
Tset	Yes	1 t_bit <= Tset <= 494 t_bit
Tqui	Yes	0 t_bit <= Tqui <= MIN(31 t_bit, Min. Tsdr - 1)
GAP factor	Yes	1 <= GAP factor <= 100
Retry limit	Yes	1 <= Retry limit <= 15
Tslot	No	---
Tid2	No	Tid2 = Max. Tsdr
Trdy	No	Trdy = Min. Tsdr
Tid1	No	Tid1 = 35 + 2*Tset + Tqui
Ttr	Yes	256 t_bit <= Ttr <= 16.777.960 t_bit

Bus parameter	Adjustable?	Limit values
Ttr typical	No	This time is provided for information only and is not transferred to the nodes.
Response monitoring		10 ms <= response monitoring (watchdog) <= 650 s

If you want to create a customized bus profile, we recommend the following settings:

- Minimum target rotation time (Ttr) = 5000 x HSA (highest PROFIBUS address)
- Minimum response monitoring (watchdog) = 6250 x HSA

Recalculating

You can use the "Recalculate" button to recalculate the parameters.

See also

PROFIBUS addresses (Page 528)

Description of the bus parameters (Page 530)

Description of the bus parameters

Detailed description of PROFIBUS bus parameters

Bus parameter	Meaning
Tslot_Init	The wait-for-reception (slot time) defines the maximum time the sender will wait to receive a response from the addressed partner. If the influence of the line components on message frame run times is entered in the "Cable Configuration" parameter group, these components must also be taken into account. The component is added to the specified Tslot_Init and the total used as Tslot.
Max. Tsdr	The maximum protocol processing time defines the latest time by which the responding node should have replied.
Min. Tsdr	The minimum protocol processing time defines the earliest time by which the responding node may reply.
Tset	The trigger time is the time which may lapse between the reception of a data message frame and the response to it in the node.
Tqui	The modulator quiet time is the time which a sending node needs after the end of the message frame to switch from sending to receiving.
GAP factor	The GAP update factor defines the number of token rotations after which a newly added, active node can be added to the logical token ring.
Retry limit	This parameter defines the maximum number of attempts (message frame repeats) made to reach a node.
Tslot	The wait-for-reception time (slot time) defines the maximum time the sender will wait to receive a response from the addressed partner. If the influence of the bus design components on message frame run times is entered in the "Cable Configuration" parameter group, these components must also be taken into account. The component is added to the specified Tslot_Init and the total used as Tslot.

Bus parameter	Meaning
Tid2	Idle time 2 defines the earliest time by which a sending node may send the next message frame after sending a message frame that has not been acknowledged.
Trdy	The ready time specifies the earliest time by which a sending node may receive a response message frame.
Tid1	Idle time 1 defines the earliest time by which a sending node may send the next message frame after receiving a response.
Ttr	The target rotation time is the maximum time made available for a token rotation. During this time, all active nodes (DP masters etc.) have the right to send (token) once. The difference between the target rotation time and the actual token holding time of a node determines how much time is left over for the other active nodes (programming device, other DP masters etc.) to send message frames.
Ttr typical	The typical data cycle time is the average response time on the bus if all configured slaves are exchanging data with the DP master. None of the slaves report diagnostics and there is no extra message frame traffic with programming devices or other active nodes etc. on the bus.
Response monitoring	The response monitoring time is only needed for PROFIBUS-DP bus systems. It defines the latest time by which a DP slave has to be addressed by its DP master with a new data message frame. If this does not happen, the DP slave assumes that the DP master has failed and resets its outputs to a secure mode.

See also

What you need to know about PROFIBUS bus parameters (Page 529)

Bus profiles with PROFIBUS

Introduction

Depending on the device types connected and protocols used on the PROFIBUS, different profiles are available. The profiles differ in terms of the setting options and calculation of bus parameters. The profiles are explained below.

Devices with different profiles on the same PROFIBUS subnet

The PROFIBUS subnet only functions without problem if the bus parameters of all devices have the same values. For example, if both DP and FMS services are being used on a subnet, the "slower" sets of bus parameters must always be set for all devices, i.e. even the "Universal (DP/FMS)" profile for DP devices.

Profiles and transmission rates

Profiles	Supported transmission speeds in Kbits/s
DP	9.6 19.2 45.45 93.75 187.5 500 1500 3000 6000 12000
Standard	9.6 19.2 45.45 93.75 187.5 500 1500 3000 6000 12000
Universal (DP-FMS)	9.6 19.2 93.75 187.5 500 1500
Customized	9.6 19.2 45.45 93.75 187.5 500 1500 3000 6000 12000

Meaning of profiles

Profile	Meaning
DP	Select the "DP" bus profile when the only devices connected to the PROFIBUS subnet are those which satisfy the requirements of standard EN 50170 Volume 2/3, Part 8-2 PROFIBUS. The bus parameter setting is optimized on these devices. This includes devices with DP master and DP slave interfaces of the SIMATIC S7 and distributed I/Os of other manufacturers.
Standard	Compared to the "DP" profile, the "Standard" profile also offers scope for devices of another project or devices which have not been configured here to be taken into account when calculating the bus parameters. The bus parameters are then calculated following a simple, non-optimized algorithm.
Universal (DP/FMS)	Select the "Universal (DP/FMS)" bus profile when individual devices on the PROFIBUS subnet use the PROFIBUS-FMS service. This includes the following devices for example: • CP 343-5 (SIMATIC S7) • PROFIBUS-FMS devices of other manufacturers As with the "Standard" profile, this profile allows you to take other devices into account when calculating the bus parameters.
Customized	The PROFIBUS subnet will only function without problem if the parameters for the bus profile are matched to one another. Select the "Customized" profile when none of the usual profiles "match" a PROFIBUS device and you need to adapt the bus parameters to your special structure. Information on this can be found in the documentation for the PROFIBUS device. You should only change the default values if you are familiar with how to configure the bus profile for PROFIBUS. Not all combinations that can be theoretically set can be used even with this bus profile. The PROFIBUS standard specifies several parameter limits that depend on other parameters. For example, a responder must not respond (Min Tsdr) before the initiator can receive the message frame (Trdy). These standard specifications are also checked in the "Customized" profile. Tip: The bus parameters last valid on the PROFIBUS subnet are always automatically set as customized. For example, if the "DP" bus profile was valid for the subnet, then the bus parameters for "DP" are set in the "Customized" bus profile. The parameters can be modified on this basis. The monitoring times are not automatically recalculated so that you do not put at risk the consistency of set values, for example with configurations in other configuration tools without realizing that you have done so. You can also have the Ttr monitoring times and target rotation time calculated on the basis of parameters you have set: Click here on the "Recalculate" button.

Note

Both mono-master mode and multi-master mode are possible with all PROFIBUS profiles.

What you need to know about PROFIBUS line configuration

Cable configuration and bus parameters

Information regarding the cable configuration can be taken into consideration when calculating the bus parameters. For this purpose, you must select the "Consider cable configuration" check box in the properties for the PROFIBUS subnet.

The remaining information then depends on the type of cable used; the following settings are available:

- Copper cable
- Fiber-optic cable/optical ring

PROFIBUS line configuration, optical ring

The calculation depends on the OLM types used. The selection is made by activating the check box (multiple activation is possible, at least one OLM type must be selected):

- OLM/P12
- OLM/G12
- OLM/G12-EEC
- OLM/G12-1300

The following bus parameter adjustments are made:

- Configuration of a node not present

  Note

  The following restrictions apply to optical rings, even for passive nodes (DP slaves for example):

  A maximum of HSA-1 nodes may be connected to the PROFIBUS network. With an HSA of 126, addresses 126 and 125 must not be used. A maximum of 125 nodes are possible on the bus (No. 0 to 124).

  If an HSA is less than or equal to 125, the addresses HSA and greater may not be used. The address HSA-1 may not be used.

- Increase the retry value to 3
- Setting of minimum slot time needed for ring mode

  Note

  Short slot time values are needed for OLM/P12, average ones for OLM/G12 and OLM/G12-EEC and long ones for OLM/G12-1300. This results in high performance for small network lengths and average to low performance for average to large network lengths.

PROFIBUS communication load

Communication load - allowing for additional network stations

The bus parameters depend on the volume of communication between the active network nodes. There are differences between cyclic communication (DP) and connection-based, acyclic communication (S7 communication, Send/Receive (FDL), FMS). Unlike DP, the volume and size of communication tasks (communication load) depends on the user program. For this reason, the communication load cannot always be calculated automatically.

To calculate the bus times you can define a network configuration in the "Additional network stations" parameter group that differs from the network configuration.

Taking the profile into account

The network configuration can be defined for the "Standard", "Universal (DP/FMS)", and "User-defined" profiles. Parameters cannot be entered in the "Additional network stations" parameter group for the "DP" profile.

Quantification of the communication load

The following settings are available in order to make allowance for the communication load.

- Information regarding the number of unconfigured network stations;
- Information on the communication load resulting from the user programs for FDL or S7 communication. Here you can selected from the following settings:
  - Low
    Typically used for DP, no great data communication apart from DP.
  - Medium
    Typically used for mixed operations featuring DP and other communication services (such as for S7 communication), when DP has strict time requirements and for average acyclic volumes of communication.
  - High
    For mixed operations featuring DP and other communication services (such as for S7 communication), when DP has loose time requirements and for high acyclic volumes of communication.

Configuring Industrial Ethernet

Rules for the network configuration

The Ethernet interfaces of the devices have a default IP address that you can change.

IP address

The IP parameters are visible if the module capable of communication supports the TCP/IP protocol. This is usually the case for all Ethernet modules.

The IP address consists of 4 decimal figures in the range of 0 to 255. The decimal figures are separated from one another by a dot.

Example: 140.80.0.2

The IP address consists of:

- Address of IP subnet
- Address of the device (generally also called host or network node)

Subnet mask

The subnet mask splits these two addresses. It determines which part of the IP address addresses the network and which part of the IP address addresses the device.

The set bits of the subnet mask determine the network part of the IP address.

Example:

Subnet mask: 255.255.0.0 = 11111111.11111111.00000000.00000000

In the example given for the above IP address, the subnet mask shown here has the following meaning:

The first 2 bytes of the IP address identify the subnet - i.e. 140.80. The last two bytes address the device, i.e. 0.2.

It is generally true that:

- The network address results from AND logic operation of the IP address and subnet mask.
- The device address results from the AND NOT logic operation of the IP address and subnet mask.

Relation between IP address and default subnet mask

An agreement exists relating to the assignment of IP address ranges and so-called "Default subnet masks". The first decimal number (from the left) in the IP address determines the structure of the default subnet mask. It determines the number of "1" values (binary) as follows:

IP address (decimal)	IP address (binary)	Address class	Default subnet mask
0 to 126	0xxxxxxx.xxxxxxxx....	A	255.0.0.0
128 to 191	10xxxxxx.xxxxxxxx...	B	255.255.0.0
192 to 223	110xxxxx.xxxxxxxx...	C	255.255.255.0

Note

Range of values for the first decimal point

A value of between 224 and 255 is also possible for the first decimal number of the IP address (address class D, etc). This is not recommended, however, because there is no address check for these values.

Masking other subnets

You can use the subnet mask to add further structures and form "private" subnets for a subnet that is assigned one of the address classes A, B or C. This is done by setting other lower points of the subnet mask to "1". For each bit set to "1", the number of "private" networks doubles and the number of devices they contain is halved. Externally, the network functions like an individual network as it did previously.

Example:

You have a subnet of address class B (for example IP address 129.80.xxx.xxx) and change the default subnet mask as follows:

Masks	Decimal	Binary
Default subnet mask	255.255.0.0	11111111.11111111.00000000.00000000
Subnet mask	255.255.128.0	11111111.11111111.10000000.00000000

Result:

All devices with addresses between 129.80.001.xxx and 129.80.127.xxx are on one IP subnet, all devices with addresses between 129.80.128.xxx and 129.80.255.xxx are on another IP subnet.

Router

The task of the routers is to connect the IP subnets. If an IP datagram is to be sent to another network, it first has to be conveyed to a router. To make this possible, you have to enter the address of the router for each device in the IP subnet.

The IP address of a device in the subnet and the IP address of the router can only differ at the points at which there is a "0" in the subnet mask.

See also

Settings for interconnecting Ethernet devices (Page 536)

Settings for interconnecting Ethernet devices

The following paragraph describes the behavior of STEP 7 during interconnection of PROFINET devices and the effects of the port interconnection on the network view.

Relation between port interconnection and network view

Topology view:

In the topology view, you specify the physical interconnection of Ethernet ports.

You specifically determine which Ethernet port of a device is to be connected with a specific Ethernet port of another device by means of an Ethernet cable (preset topology).

Example:

You specify that port 1 of the PROFINET interface of the CPU is to be connected to port 2 of the PROFINET interface of device A by means of an Ethernet cable.

You also specify the interface for devices with several PROFINET interfaces.

Example:

You specify that port 1 of PROFINET interface X2 of the CPU is to be connected to port 2 of the PROFINET interface of device A by means of an Ethernet cable.

The Ethernet ports can be interconnected in a table or graphically.

Network view:

In the network view, you specify which devices are to be connected with each other via an Ethernet subnet. You do not specify the Ethernet ports by which the devices are interconnected with each other (that is the task of the port interconnection).

The port interconnection has effects on the network view:

When you interconnect Ethernet ports of devices with each other in the topology view, STEP 7 connects the interconnected PROFINET interfaces of the devices in the network view with an Ethernet subnet (green line).

However, the course of the green line does not reflect the actual cable routing. You specify the actual wiring in the topology view.

An Ethernet subnet always has a name and an S7 subnet ID. You can set these two values in the subnet properties.

Which Ethernet subnet is used to link interconnected devices?

STEP 7 distinguishes between the following cases:

- Ethernet subnet specified (default subnet).
- No Ethernet subnet specified (no default subnet).

Ethernet subnet specified (default subnet)

The option "Connect devices that are not linked with this subnet in case of port connection" is activated in the properties of an Ethernet subnet (default).

This option can be activated for exactly one Ethernet subnet.

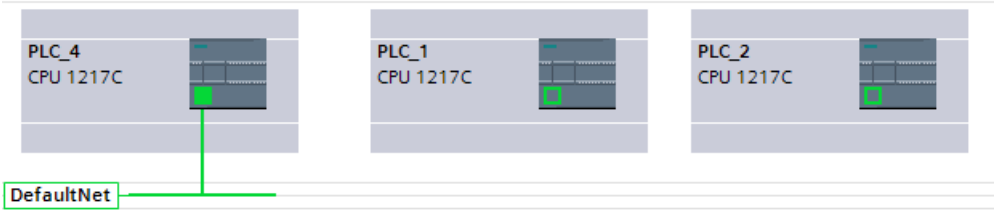
Activate the option, if necessary, for the Ethernet subnet that is to be continued when interconnecting devices that are not linked. This subnet is referred to as "default subnet" below.

Response of STEP 7:

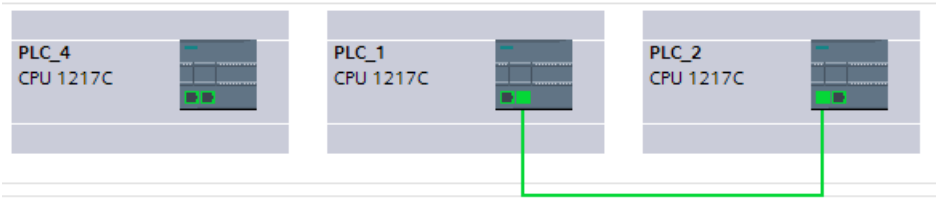
When you interconnect the ports of two devices that are not linked in the topology view, STEP 7 links these devices with the default subnet.

Example:

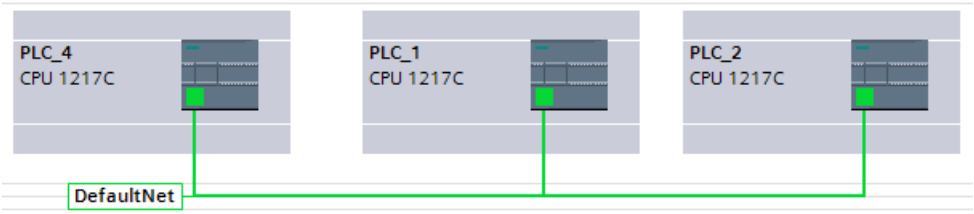
Step 1: Create a subnet in the network view at PLC_4, "Connect devices that are not linked with this subnet in case of port connection" option is **enabled**.



Step 2: Interconnect PLC_1 with PLC_2 (topology view).



Result: All PLCs are now connected to the same default subnet (network view).



No Ethernet subnet specified (no default subnet).

This is the case when the following conditions are fulfilled:

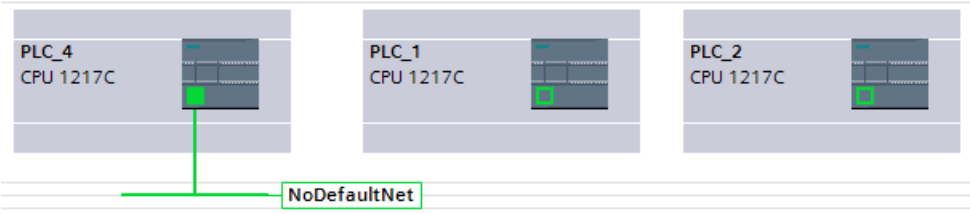
- A subnet has been added to a PROFINET interface (this interface is referred to as "Interface A" below).
- The option "Connect devices that are not linked with this subnet in case of port connection" is deactivated in the properties of this subnet (no default subnet).
- There is no other Ethernet subnet for which this option is activated.

Response of STEP 7:

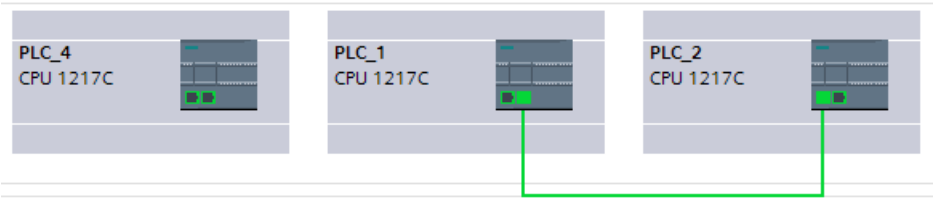
- The Ethernet subnet of interface A is only continued when you interconnect a port of interface A with a port of another device.
- When you interconnect ports of other devices that are not linked with each other, STEP 7 creates a new Ethernet subnet.

Example:

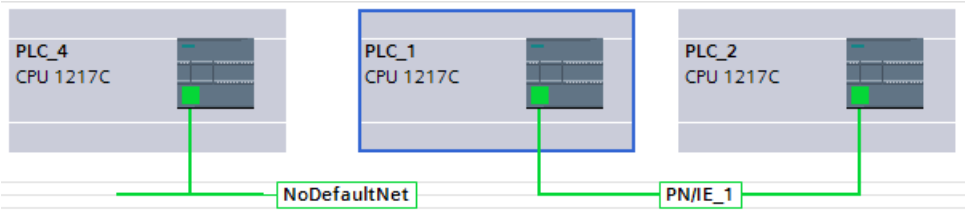
Step 1: Create a subnet in the network view at PLC_4, "Connect devices that are not linked with this subnet in case of port connection" option is **disabled**.



Step 2: Interconnect PLC_1 with PLC_2 (topology view).



Result: PLC_1 and PLC_2 are connected to a new subnet (network view).



Manual adaptation of the IP addresses

It can occur that STEP 7 does not adapt the IP addresses of the interconnected devices in such a way that the resulting network can be correctly compiled, for example, for devices with multiple PROFINET interfaces.

In these situations, you need to adapt the IP addresses of the devices manually.

The following rules apply:

- Devices that are to communicate with each other without a router cannot belong to different IP subnets.
- For devices with multiple PROFINET interfaces, the interfaces must be in different IP subnets.

To change the IP address of a PROFINET interface, perform the following steps:

1. Change to the network view (if not selected already).
2. Left-click on the icon of the PROFINET interface that is not to be part of the IP subnet.
3. Change the subnet part of the IP address in the properties of the PROFINET interface (in the "Ethernet addresses" area).

Example: Manual conversion of the IP subnet part for devices with multiple PROFINET interfaces in a subnet

The IP address is: "192.168.0.1".

The subnet mask is "255.255.255.0".

The first three numbers "192.168.0" form the IP subnet part of the IP address "192.168.0.1".

Change the IP subnet part, for example, to "192.168.1"

See also

Interconnecting ports in the graphic view (Page 718)

Interconnecting ports in the table view (Page 719)

Configuring Industrial Ethernet (Page 534)

Network configuration of AS interface

AS-Interface consists of an AS-i master and AS-i slaves connected to each other over an AS-i subnet.

Rules for AS interface network configuration

All the nodes in an AS-i subnet must have a different AS-i node address.

You should only load the settings over the network when all the modules in a subnet have different addresses and when the actual structure matches that of the network configuration you have created.

One AS-i master and up to 31 AS-i slaves can be operated on one AS-i subnet.

For more information on configuring an AS-Interface with an AS-i master and AS-i slaves, refer to the section on AS-Interface and the documentation of the AS-i master modules.

8.1.3.2 Communication via connections

Working with connections (S7-1200)

S7 connection

Introduction to configuring connections

Definition

A connection defines a logical assignment of two communication partners in order to undertake communication services. A connection defines the following:

- Communication partner involved
- Type of connection (for example, S7 connection)
- Special properties (e.g., whether a connection is established permanently or whether it is established and terminated dynamically in the user program, and whether status messages are to be sent)
- Connection path

What you need to know about connection configuration

During connection configuration, a local connection name is assigned for an S7 connection as a unique local identification.

In the network view, a "Connections" tab is displayed in addition to the "Network overview" tab. This tab contains the connection table. A row in this connection table represents a configured connection from the viewpoint of the local communication partner with its properties, for example, between two S7-1200 CPUs.

What you need to know about using connection resources

Introduction

Each connection requires connection resources for the end point and/or transition point on the devices involved. The number of connection resources is device-specific.

If all the connection resources of a communication partner are assigned, no new connections can be established. This situation is apparent when a newly created connection in the connection table has a red background. The configuration is then inconsistent and cannot be compiled.

S7 connections

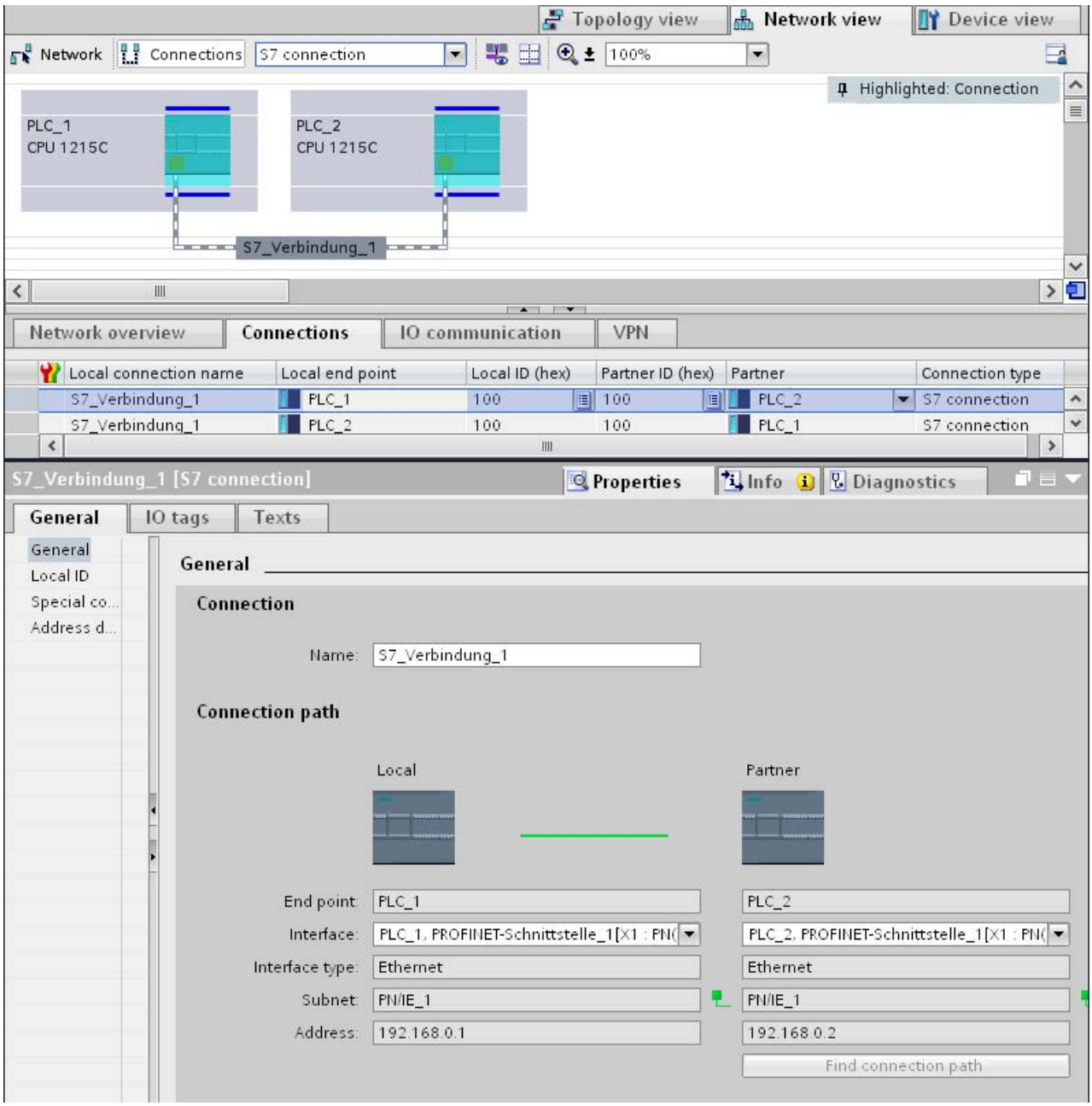
In the case of S7 connections via the PN interface, one connection resource per S7 connection is assigned for the endpoint for the S7-1200 CPU. One connection resource is also required for the connection partner.

You can find an overview of available and assigned connection resources for selected S7-1200 CPU in the Inspector window at "Properties > Connection Resources"

Views containing information about the configured connections

The views described below will provide you with comprehensive access to all the information and functions regarding configuring and checking communication connections.

- Connection display in the network view
- Connection table
- "Properties" tab for a connection in the inspector window



Benefits

The information shown in these views are always up-to-date in terms of the current user actions. This means:

- The connection table displays all connections created.
- If you have selected a connection in the connection table:
  - When connection mode is enabled, the connection path is highlighted in the network view.
  - The "Properties" tab in the Inspector window displays the parameters of this connection.

The connection table

The connection table offers the following functions:

- List of all connections in the project
- Selection of a connection and display of connections associated with it in the network view (when connection mode is enabled)
- Changing of connection partners
- Display showing status information

"Properties" tab for a connection in the inspector window

The properties dialog has the following meaning:

- Display for connection parameters
- Display of connection path
- Subsequent specification of connections using the "Find connection path" button

Creating a new connection

Creating a connection - alternatives

You have the following options for creating a connection in the network view:

- Graphic connection configuration
- Interactive connection configuration

You'll find the individual steps for this in the following chapters.

Requirement and result

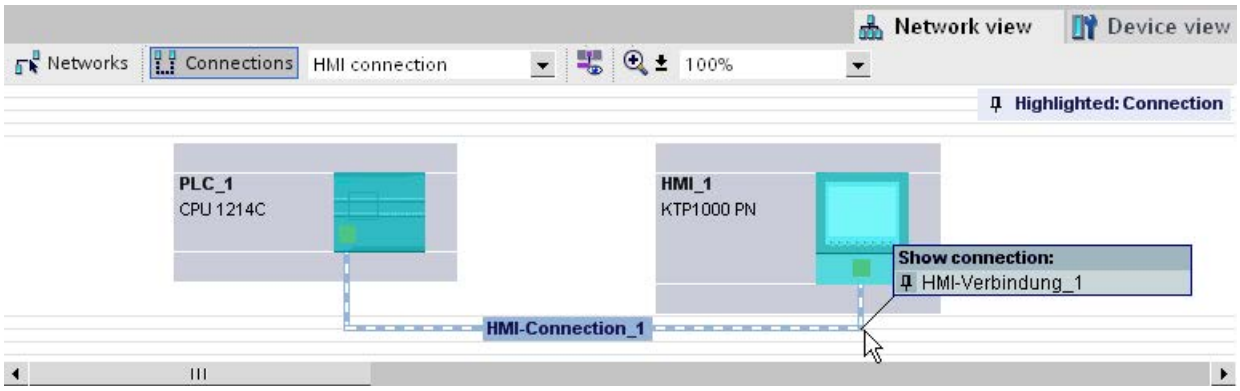
In the network view, you have add devices between which the connections should be configured.

Specifying a connection

If both partners for the connection type selected are networked on the same network, use the graphic or interactive selection of both communication partners to create a fully specified connection.

This connection is entered automatically in the connection table of the S7-1200 CPU. A local connection name is assigned for this connection.

The following schematic shows a configured connection with a networked device:



Creating a new connection graphically

Graphically configuring connections

In the case of graphic connection configuration, the connection path is automatically specified provided interfaces and resources are available. Select the devices to be connected in the current configuration.

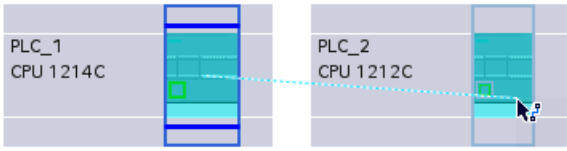
Automatically determining connection path

To create a connection graphically, follow these steps:

1. Click the "Connections" button.

   This step activates the connection mode: You can now select the connection type you want. You will see this from the following:
   The devices that can be used for the connection type selected in your project are color-highlighted in the network view.

2. Hold down the mouse button and drag the mouse pointer from the device from which the connection will originate to the device at which the connection ends.

3. Release the mouse button over the destination device to create the connection between the two devices.

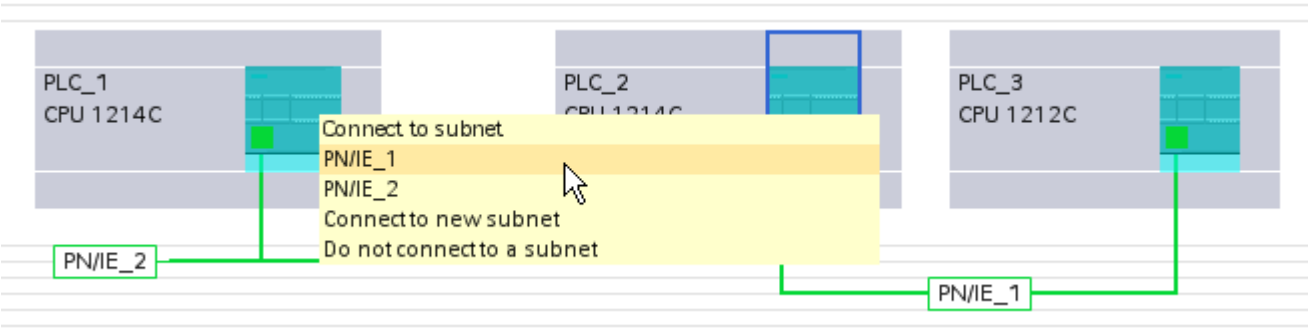
Result

- A specified connection is created.
- The connection path is highlighted.
- The connection is entered in the connection table.

Configuring a connection when there is no or no clear network assignment

Any networking not in place will if possible be created automatically when a connection is created. A query will be made upon completion of connection configuration if unique network assignment is not possible. You will be able to choose from the existing subnets.

Example below: A query is made upon creation of a connection between stations PLC_1 and PLC_2, which are not yet networked.



Interactively creating a new connection

Interactively configuring connections

Define the local device and its connection partners.

Procedure

Proceed as follows to interactively create a connection:

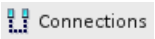
1. Select the "Add new connection" command in the shortcut menu of a connection partner for which you want to create a connection.
   The "Create new connection" dialog appears.
2. Select the partner endpoint.
   In the right pane of the dialog, a possible connection path fitting the selected endpoint is displayed, if available. Incomplete paths, for example, for a non-specified CPU, are marked by an exclamation mark on a red background.
3. To accept the configured connection and to configure additional connections to other endpoints, click "Add".
   To close the dialog, click "OK".

Working in the network view

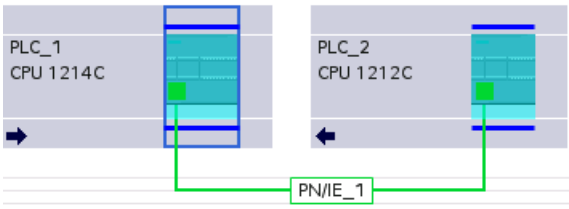
Highlighting connection path and partner in the network view

To display the connection partners for all or certain connection types in the network view, proceed as follows:

1. Click the "Connections" button.

2. Select the S7-CPU for which you want to display the connection partners in the network view and then select the "Highlight connection partners" command in the shortcut menu.
3. Select "All connection partners" in the following menu.
   The local device and the CPUs of the target devices are selected. The local connection partner shows an arrow pointing right and the remote connection partners show an arrow pointing left.
4. To open a list with information on the target devices, click the arrow of the local device.
   This additional function is useful in complex network configurations in which some devices are not visible.



Note

You can display one of the connection partners which cannot be seen in the current display range of the network view. Click on the communication partner in the list that appears. Result: The display is moved such that the connection partner becomes visible.

Working with the connection table

Basic functions for tables

The connection table supports the following basic functions for editing a table:

- Changing column width
- Displaying the meaning of a column, a row or cell using tooltips.

Changing column width

To adjust the width of a column to the content so that all texts in the lines are legible, follow these steps:

1. Position the cursor in the header of the connection table to the right of the column that you want to optimize until the cursor changes its shape to two parallel lines (as if you wanted to change the width of the column by dragging it with the cursor).
2. Double click on this point.

or

1. Open the shortcut menu on the header of the table.
2. Click on
   - "Optimize column width" or
   - "Optimize width of all columns".

For columns that are too narrow, the complete content of specific fields is shown when you pause with the cursor on the respective field.

Show / hide columns

You can use the shortcut menu of the header of the connection table to control the display of the various table columns. The shortcut menu entry "Show/hide columns" provides you with an overview of the available columns. Use the check box to control whether columns are shown or hidden.

If you want to store the layout, width and visibility of the table columns, click on the "Remember layout" function in the top right-hand of the network view.

Symbol	Meaning
	Remember layout Saves the current table view. The layout, width and visibility of columns in the table view is stored.

Using cursor keys to move within the connection table

You can use the UP and DOWN cursor keys to select a connection from the connection table; the selected connection is marked and is shown highlighted in the network view.

Changing properties of connection

You can directly edit some of the parameters displayed in the connection table. The name of the connection can, for example, only be changed in the connection table.

Changing connection partners

You can change the connection partner of a connection as follows:

1. Select the connection.
2. Select the new connection partner from the open drop-down list in the "Partner" column.

Deleting connections

You can delete configured connections using the network view or the connection table.

In the network view you can delete one highlighted connection per action. In the connection table you can delete one or several connections per action.

Procedure

To delete a connection, follow these steps:

1. Select the connection to be deleted:
   - In the network view: Select the connection to be deleted.
   - In the connection table: Select the rows of the connections to be deleted (multiple selection possible).
2. Open the shortcut menu with a right mouse click.
3. Select the "Delete" command.

Result

The selected connection is removed completely.

Copying connections

Introduction

Connections are not copied singly but always in context along with the project or the device.

You can copy:

- Entire projects
- One or more devices within a project or from one project to another

Copying a project

When you copy a project all configured connections will also be copied. No settings whatsoever are required for the copied connections because the connections remain consistent.

Copying devices

If you copy devices for which connections have been configured, the connections are copied as well. To complete the connection path, you must still finalize the networking.

An S7-1200 CPU with a V.10 firmware is merely a server for connections and has no connection configuration itself. Consequently, no connections are copied along with it when an S7-1200 CPU with a V1.0 firmware is copied.

Inconsistent connections - connections without assignment

With an inconsistent connection the structure of the connection data is destroyed or the connection is not functional in the project context.

Inconsistent connections cannot be compiled and loaded - operation is not possible with such a connection.

In the connection table inconsistent connections are marked in red.

Possible causes for inconsistent connections

- Deletion or change of the hardware configuration.
- Missing interface network links in the project, which are necessary for a connection.
- Connection resources are exceeded
- Connections to an unspecified connection partner without partner address information.

Detailed information regarding the reasons for the inconsistency can be found in the "Compile" tab following compilation (Edit > Compile).

Remedies

To assign a closed connection path to an existing open connection path, expand the device configuration in such a way that the interfaces required for the connection type are available

for both partners. At "Properties > General > Interface" in the Inspector window, you can use the "Find connection path" button to create a connection to an existing partner.

S7 connection - general settings

General connection parameters

General connection parameters are displayed in the "General" parameter group under the properties of the connection; these connection parameters identify the local connection end point.

Here, you can assign the connection path and specify all aspects of the connection partner.

Local ID

The local ID of the module from which the connection is viewed is displayed here (local partner). You can change the local ID. You may need to do this if you have already programmed communication function blocks, and you want to use the local ID specified in those function blocks for the connection.

Special connection properties

Display of connection properties (can be modified depending on the components used):

- One-way
  One-way means that the connection partner functions as a server on this connection and cannot send or receive actively.
- Active connection establishment
  In the case of one-way connection, for example with a S7-1200 CPU (firmware version V1.0), a connection partner can only be provided for the active connection establishment. In the case of a two-way connection you can set which connection partner will assume the active role.
- Sending operating mode messages
  Indicates whether or not the local partner sends operating mode messages to the connection partner.

Address details

Displaying address details of the S7 connection. With an unspecified partner, the values for the rack and slot can be changed. All other values are obtained from the current configuration and cannot be changed.

S7 connection - address details

Meaning

The address details show the end points of the connection and can localize these via the specification of rack and slot.

When a connection is established, the connection-specific resources of a module are assigned specifically to this connection. This assignment requires that the connection resource can be addressed. The TSAP (Transport Service Access Point) is, as it were, the address of the resource that is formed with the help of the connection resource or, in the case of S7-1200 CPUs (firmware V2.0 or higher) with the SIMATIC-ACC (SIMATIC Application Controlled Communication).

Configuration of TSAP for S7-1200

- For S7-1200 CPU (firmare V2.0 or higher):
  "SIMATIC-ACC"<nnn><mm>
  nnn = Local ID
  mm = any value
- For S7-1200 CPU (firmare V1.0):
  <xx>.<yz>
  xx = Number of the connection resource
  y = Rack number
  z = Slot number

TSAP structure, dependent on partner

The configuration of the TSAP for S7-1200 CPUs is dependent on the respective firmware and on the remote connection partner. When a S7-1200 CPU is connected with a S7-300/400 CPU, a S7-1200 CPU also uses a TSAP configuration that includes the connection resource.

See the following examples for TSAPs of various connection configurations

- Connection between two S7-1200 CPUs (both with firmware V2.0):
  - S7-1200 CPU "A" with firmware V2.0 and local ID 100:
    TSAP: SIMATIC-ACC10001
  - S7-1200 CPU "B" with firmware V2.0 and local ID 5AE:
    TSAP: SIMATIC-ACC5AE01
- Connection between two S7-1200 CPUs (firmware V2.0 and V1.0):
  - S7-1200 CPU with firmware V2.0 and local ID 1FF:
    TSAP: SIMATIC-ACC1FF01
  - S7-1200 CPU with firmware V1.0 (rack 0, slot 1, connection resource 03):
    TSAP: 03.01
- Connection between S7-1200 CPUs (firmware V2.0) and S7-300/400 CPU:
  - S7-1200 CPU with firmware V2.0 (rack 0, slot 1, connection resource 12):
    TSAP: 12.01
  - S7-300/400 CPU (rack 0, slot 2, connection resource 11):
    TSAP: 11.02

S7 connections via CM/CP

Introduction

S7-1200 CPUs with a firmware version of V2.0 or higher support one-way and two-way S7 connections via CM/CP interfaces. This increases the number of Ethernet ports and networks that can be used for S7 connections. Even though the connection is then guided via the CM/CP, the associated S7-1200 CPU is an end point of the connection. The other end point can be any other device in the case of two-way connections. This other device must also support S7 connections.

Data volumes and configuration limit

The number of communication connections which are supported by CM/CP can be found in the manual accompanying every CM/CP. The number of connections per device can be further increased by adding other CM/CPs.

If several CM/CPs are fitted in a device, an automatic switch is made to the next CM/CP when this limit is exceeded. Specifically assign the connections using the path selection as required.

Note

Data transfer > 240 byte transfer is supported by the current CPs.

CPs with an older product version support the transfer of data with a data length of up to 240 bytes.

For more information, refer to the details provided in the Ethernet CP equipment manual.

Tasks of the Ethernet CM/CP in online mode

During data transfer via a connection, the Ethernet CM/CP takes on the following tasks:

- Receiving
  Receiving data from the Ethernet and forwarding to the user data area in the CPU
- Sending
  Transferring data from the user data area of the CPU and sending data via the Ethernet

The connection is established automatically as soon as the partner can be reached.

HMI connection

Introduction to configuring connections

Definition

A connection defines a logical assignment of two communication partners in order to undertake communication services. A connection defines the following:

- Communication partner involved
- Type of connection (e.g., HMI connection)
- Special properties (e.g., whether a connection is established permanently or whether it is established and terminated dynamically in the user program, and whether status messages are to be sent)
- Connection path

What you need to know about connection configuration

During connection configuration, a local connection name is assigned for an HMI connection as a unique local identification.

In the network view, a "Connections" tab is displayed in addition to the "Network overview" tab. This tab contains the connection table. A line in this connection table represents a configured connection, e.g., between an HMI device and PLC, along with its properties.

What you need to know about using connection resources

Introduction

Each connection requires connection resources for the end point and/or transition point on the devices involved. The number of connection resources is device-specific.

If all the connection resources of a communication partner are assigned, no new connections can be established. This situation is apparent when a newly created connection in the connection table has a red background. The configuration is then inconsistent and cannot be compiled.

HMI connections

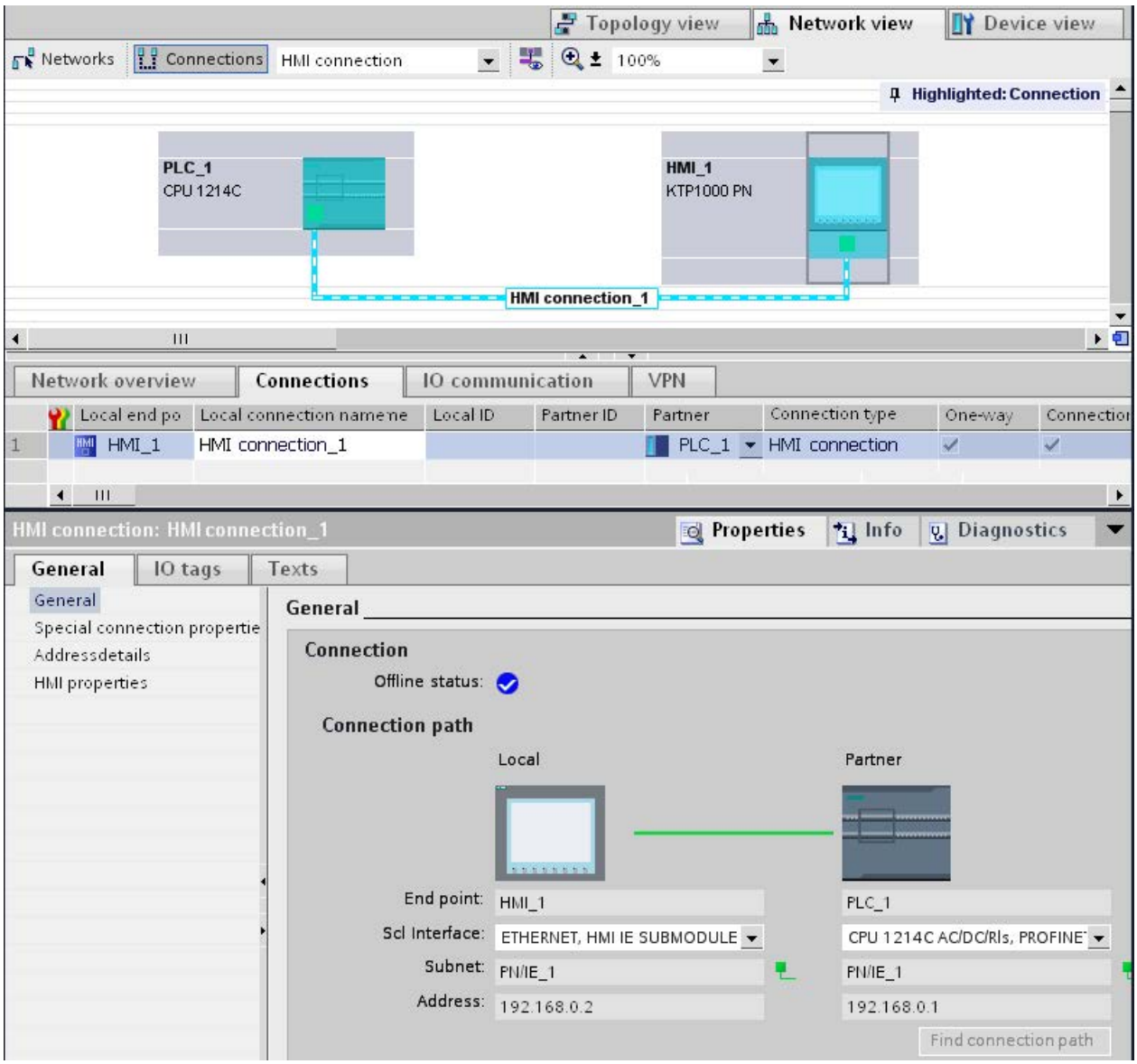
For HMI connections via the **integrated** PN interface, one connection resource for the endpoint per HMI connection is occupied for the HMI device.

One connection resource is also required for the connection partner (PLC).

Views containing information about the configured connections

The views described below will provide you with comprehensive access to all the information and functions regarding configuring and checking communication connections.

- Connection display in the network view
- Connection table
- "Properties" tab for a connection in the inspector window



Benefits

The information shown in these views are always up-to-date in terms of the current user actions. This means:

- The connection table displays all connections created.
- If you have selected a connection in the connection table:
  - You will graphically see the connection path in the network view.
  - The "Properties" tab in the Inspector window displays the parameters of this connection.

The connection table

The connection table offers the following functions:

- List of all connections in the project
- Selection of a connection and display of connections associated with it in the network view
- Changing of connection partners
- Display showing status information

"Properties" tab for a connection in the inspector window

The properties dialog has the following meaning:

- Display for connection parameters
- Display of connection path
- Subsequent specification of connections using the "Find connection path" button

Creating a new connection

Creating a connection - alternatives

You have the following options for creating a connection in the network view:

- Graphic connection configuration
- Interactive connection configuration

You'll find the individual steps for this in the following chapters.

Requirement and result

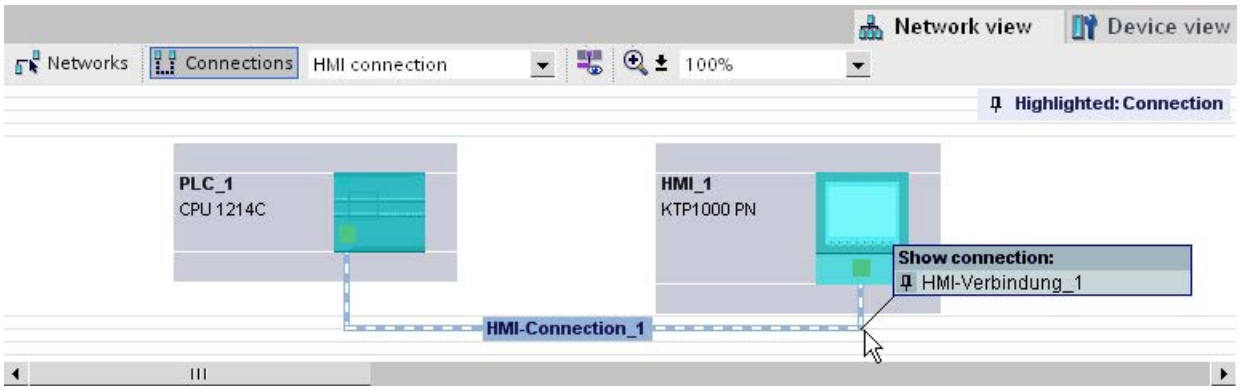
You have created the devices with CPUs and HMI devices between which you want to configure connections in the network view.

Specifying a connection

If both partners for the connection type selected are networked on the same network, use the graphic or interactive selection of both communication partners to create a fully specified connection.

This connection is entered automatically into the connection table of the HMI device. A local connection name is assigned for this connection.

The following schematic shows a configured connection with a networked device:



Creating a new connection graphically

Graphically configuring connections

When using the graphic connection configuration, if necessary the system asks you to define the connection path. Select the devices to be connected in the current configuration.

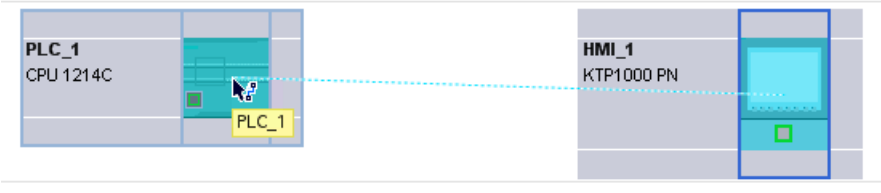
Automatically determining connection path

To create a connection graphically, follow these steps:

1. Click the "Connections" button.

   The connection mode for the connection type you have selected is then activated.
   You will see this from the following:
   The devices that can be used for the connection type selected in your project are color-highlighted in the network view.

2. Hold down the mouse button and drag the mouse pointer from the device from which the connection will originate to the device at which the connection ends.

3. Release the mouse button over the destination device to create the connection between the two devices.

Result

- A specified connection is created.
- The connection path is highlighted.
- The connection is entered in the connection table.

Interactively creating a new connection

Interactively configuring connections

Define the local device and its connection partners.

Procedure

Proceed as follows to interactively create a connection:

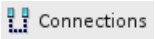
1. Select the "Create new connection" command in the shortcut menu of a connection partner for which you want to create a connection.
   The "Create new connection" dialog is opened.
2. Select the partner endpoint.
   In the right pane of the dialog, a possible connection path fitting the selected endpoint is displayed, if available. Incomplete paths, for example, for a non-specified CPU, are marked by an exclamation mark on a red background.
3. To close the dialog, click "OK".
   To accept the configured connection and to configure additional connections to other endpoints, click "Apply".

Working in the network view

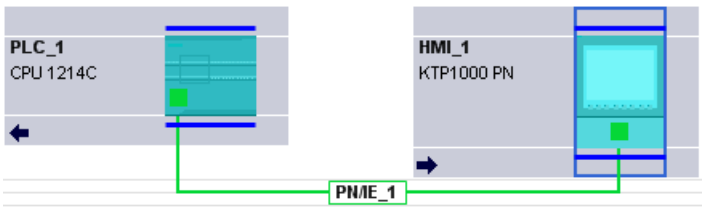
Highlighting connection path and partner in the network view

To display the connection partners for all or certain connection types in the network view, proceed as follows:

1. Click the "Connections" button.



2. Select the "Highlight connection partners" command in the shortcut menu for the HMI device whose connection partners you want to display in the network view.
3. Select "All connection partners" in the following menu.
   The local device and the CPUs of the target devices are selected. The local connection partner shows an arrow pointing right and the remote connection partners show an arrow pointing left.
4. To open a list with information on the target devices, click the arrow of the local device.
   This additional function is useful in complex network configurations in which some devices are not visible.



Note

You can display one of the connection partners which cannot be seen in the current display range of the network view. Click on the communication partner in the list that appears. Result: The display is moved such that the connection partner becomes visible.

See also

Creating a new connection graphically (Page 557)

Working with the connection table

Basic functions for tables

The connection table supports the following basic functions for editing a table:

- Changing column width
- Displaying the meaning of a column, a row or cell using tooltips.

Changing column width

To adjust the width of a column to the content so that all texts in the lines are legible, follow these steps:

1. Position the cursor in the header of the connection table to the right of the column that you want to optimize until the cursor changes its shape to two parallel lines (as if you wanted to change the width of the column by dragging it with the cursor).
2. Double click on this point.

or

1. Open the shortcut menu on the header of the table.
2. Click on
   - "Optimize column width" or
   - "Optimize width of all columns".

For columns that are too narrow, the complete content of specific fields is shown when you pause with the cursor on the respective field.

Show / hide columns

You can use the shortcut menu of the header of the connection table to control the display of the various table columns. The shortcut menu entry "Show/hide columns" provides you with an overview of the available columns. Use the check box to control whether columns are shown or hidden.

Using cursor keys to move within the connection table

You can use the UP and DOWN cursor keys to select a connection from the connection table; the selected connection is marked and is shown highlighted in the network view.

Changing properties of connection

You can directly edit the parameters displayed in the connection table in some cases. To change the name of a connection, you do not have to navigate to the Inspector window.

Changing connection partners

You can change the connection partner of a connection as follows:

1. Select the connection.
2. Select the new connection partner from the open drop-down list in the "Partner" column.

Deleting connections

You can delete configured connections using the network view or the connection table.

In the network view you can delete one highlighted connection per action. In the connection table you can delete one or several connections per action.

Procedure

To delete a connection, follow these steps:

1. Select the connection to be deleted:
   - In the network view: Select the connection to be deleted.
   - In the connection table: Select the rows of the connections to be deleted (multiple selection possible).
2. Open the shortcut menu with a right mouse click.
3. Select the "Delete" command.

Result

The selected connection is removed completely.

Copying connections

Introduction

Connections are not copied singly but always in context along with the project or the device.

You can copy:

- Entire projects
- One or more devices within a project or from one project to another

Copying a project

When you copy a project all configured connections will also be copied. No settings whatsoever are required for the copied connections because the connections remain consistent.

Copying devices

If you copy devices for which connections have been configured (HMI devices), the connections are copied as well. To complete the connection path, you must still finalize the networking.

An S7-1200 CPU with a V.10 firmware is only a server for connections and has no connection configuration itself. Consequently, no connections are copied along with it when an S7-1200 CPU with a V1.0 firmware is copied.

Inconsistent connections - connections without assignment

With an inconsistent connection the structure of the connection data is destroyed or the connection is not functional in the project context.

Inconsistent connections cannot be compiled and loaded - operation is not possible with such a connection.

In the connection table inconsistent connections are marked in red.

Possible causes for inconsistent connections

- Deletion or change of the hardware configuration.
- Missing interface network links in the project, which are necessary for a connection.
- Connection resources are exceeded
- Errors when backing up data due to insufficient memory
- Connections to an unspecified connection partner without partner address information.

Detailed information regarding the reasons for the inconsistency can be found in the "Compile" tab following compilation (Edit > Compile).

Remedies

If the connection cannot be repaired by opening the connection properties, changing them or undoing them in the configuration, then it may be necessary to delete the connection and re-create it.

HMI connection general settings

General connection parameters

General connection parameters are displayed in the "General" parameter group under the properties of the connection; these connection parameters identify the local connection end point.

Here, you can also assign the connection path and specify all aspects of the connection partner.

Special connection properties

Display of the connection properties (cannot be changed):

- Active connection establishment
  The connection establishment always starts from the HMI device. This option is selected by default if the address of the partner is specified.
- One-way
  One-way means that the connection partner functions as a server on this connection and cannot send or receive actively.
- Sending operating mode messages
  Not relevant for HMI devices.

Address details

Displaying address details of the HMI connection. With an unspecified partner, the values for the rack and slot can be changed. All other values are obtained from the current configuration and cannot be changed.

Miscellaneous

Display of the access points for the online connection between HMI device and connection partner.

Connections - Purpose and selection

Introduction to configuring connections

Why set up connections?

A communication connection, or connection, defines a logical assignment of two communication partners for performing communication services.

You can configure communication connections and under certain circumstances even set them up in a program-controlled manner.

This chapter shows you how to define connections, which special aspects you need to take into account, and which communication instructions you can use in the user program.

Definition

A connection defines the following:

- Communication partner involved
- Type of connection (for example S7, ISO-on-TCP, ISO-Transport, FDL, UDP, e-mail or TCP connection)

- Special properties, such as
  - Setup characteristics
    A connection can remain permanently enabled or, if necessary, can be enabled and disabled.
  - Sending operating mode messages
- Communication path
  - within the devices (CPU - CP)
  - in the network (subnets / router)

What you need to know about connection configuration

During connection configuration, one unique local ID is issued per connection. Only this local ID is needed when configuring the communication modules. The local ID is clearly assigned to the addresses of the local and remote connection partners.

If a connection endpoint refers to the application of a PC station, the connection name is used as "Local ID" for connection identification.

After the project is opened, the project-wide connection table is displayed. If you deselect the "Display all connections" check box in the shortcut menu of the connection table, only object-specific connections of selected objects are displayed. The created connections can be displayed separately for the following objects:

- Programmable modules that can be the endpoint of a connection
- Communications modules (CPs) used on the connection path
- Subnets

The addresses of the local and remote connection partners are set when the devices are networked and assigned to the connections during configuration. The following connections are an exception to this rule:

- Available UDP connection
  For the free UDP connection, the address at the communications interface is assigned in the user program.
- Programmed communication connection

Note

The term "connection" is used under SIMATIC communication also with UDP. This is because the communication partners are assigned to one another and therefore logically "connected" during configuration. With the UDP, there is no explicit establishing of a connection between the communication partners.

Networking and the path selection determine the configuration status

Depending on whether the connection partners are present in the project and networked, the address parameters of the connection endpoints can be fully or only partly specified.

- Fully specified
  - The addresses and network parameters of the local and remote communication partners are defined.
  - The communications connection is ready for operation after the connection parameters have been loaded to the device.
- Unspecified (partly specified)
  Only the local communication partner is defined. The partner for the selected connection type is either disconnected, is outside the project or is not defined because it requires special handling.
  Application case:
  - The complete networking of the devices is not yet possible, as the hardware configuration is not yet complete.
  - A ready-to-receive status has to be established for any (unspecified) communication partners.

  The communications connection is provisionally ready for operation after the connection parameters have been loaded to the device.

Programmed communication connections

In certain application areas it is not beneficial to establish the communication connections statically via the the configuration. Alternatively, you can set up certain connection types in a program-controlled manner via a specific application, and therefore dynamically if necessary.

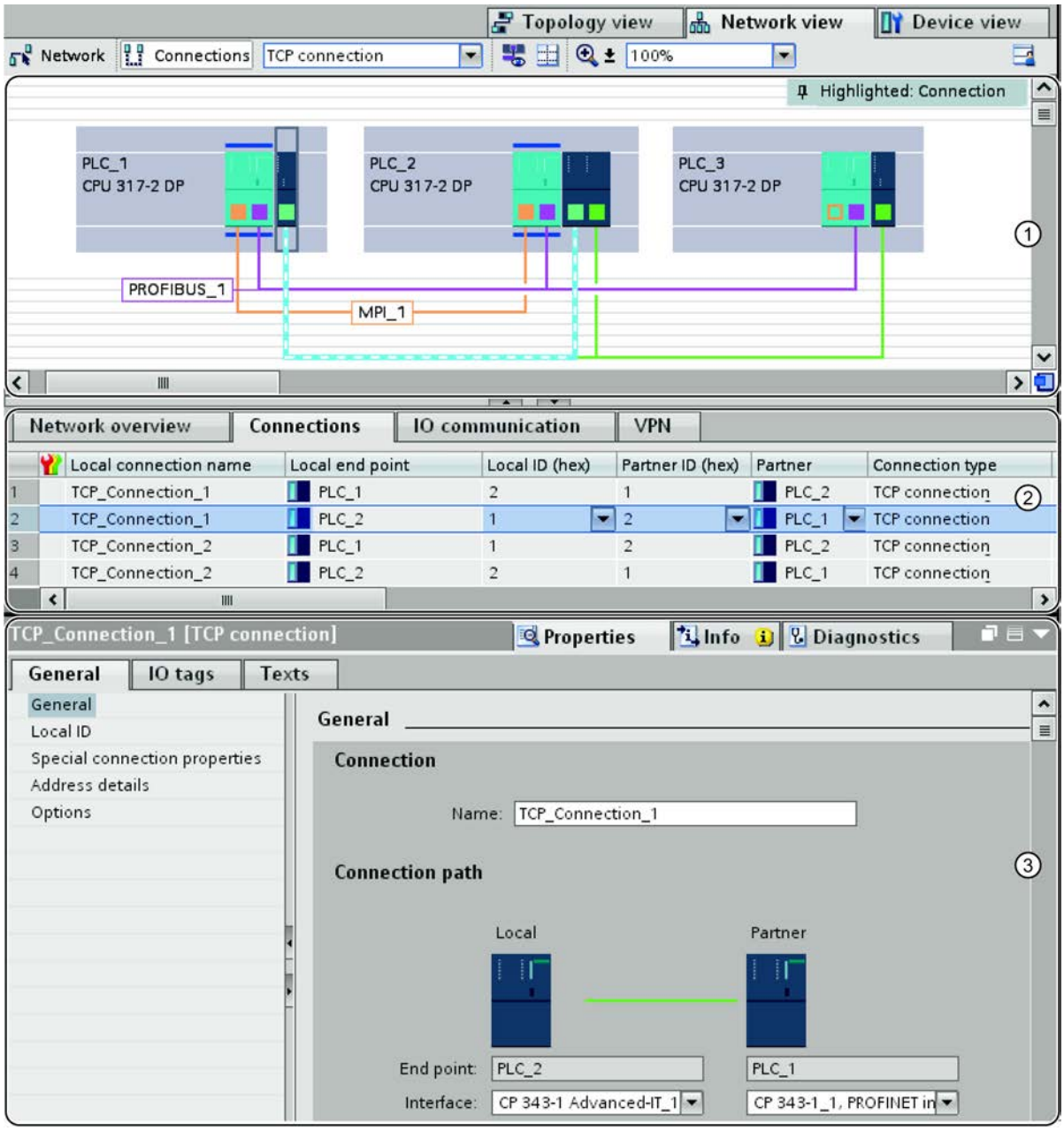
Points to note about configuring connections

The network view provides you with a clearly structured platform for configuring communication connections between configured devices.

Essential functions

- Graphic connection configuration in the network view
  You can create a connection graphically with only a few mouse clicks.
- Highlight possible connection partners
  You select a communication type - the possible communication partners are automatically highlighted.
- Automatic search for connection paths
  You arrange for the automatic assignment of the best possible connection path or the selection of alternative connection paths.
- Automatic transfer of configurations
  If you configure or network modules later on, connections that have already been configured and are supported by the module are automatically assigned.

- User-friendly selection of connection display
  If you select an object in the network view, for example a subnet or the interface for a module capable of communication, the associated configured connection link is displayed in the connection table.
- Taking into account available resources
  If connection resources are required for a connection type, any limitation specified during the creation of the connection will be taken into account. The maximum number of connection resources varies for both the CPU device types and for the CP device types.



① Connection display in the network view

② Connection table

③ "Properties" tab for a connection in the inspector window

Range of applications of connection types

Introduction

The following sections provide a brief overview of the connection types that you can use for communication to specific device types and areas of application.

Additional information

For more information on possible forms of communication in SIMATIC and the intended use of the connection types, refer the manual on communication with SIMATIC.

S7 connections

The features of S7 connections include:

- Connection type can be used in all S7 devices
- Can be used on the following subnet types: MPI, PROFIBUS, Industrial Ethernet
- When using BSEND/BRCV SFBs: secure transfer of data between SIMATIC S7-400/1500 stations; e.g., exchange of data block contents (up to 64 Kbytes)
- With CPU 317-2 PN/DP and with CPU 31x and a CP and when using BSEND/BRCV FBs: secure transfer of data to S7-300, S7-400, and S7-1500
- When using USEND/URCV SFBs: rapid, unsecured transfer of data regardless of the time at which the communication partner processes it; for example for operating and maintenance messages
- Acknowledgement of data transfer from communication partner at layer 7 of the ISO reference model

FDL connection

PROFIBUS-FDL (Fieldbus Data Link) is characterized by the following features:

- Transfer of data to a communication partner (for example SIMATIC S5 or PC), which supports sending and receiving according to the SDA function (Send Data with Acknowledge)
- Receipt of data is confirmed by the communication partner's FDL service using an acknowledgement
- Usable subnet type: PROFIBUS
- Corresponds to standard EN 50170 Vol.2 PROFIBUS
- Allows layer 2 of the ISO reference model to be assigned
- Interface in the S7 user program: SEND/RECEIVE;
- FDL services available as C functions on the PC

ISO transport connection

The ISO transport connection is characterized by the following features:

- "Data blocking" suited to large volumes of data
- Communication to a partner (for example SIMATIC S5 or PC) which supports sending and receiving data in accordance with ISO transport
- Usable subnet type: Industrial Ethernet
- Receipt of data is confirmed by the communication partner's ISO transport service using an acknowledgement
- Interface in the S7 user program: SEND/RECEIVE and FETCH/WRITE;
- ISO transport service (ISO 8073 class 4) corresponds to layer 4 of the ISO reference model
- ISO transport services available as C functions on the PC

ISO-on-TCP connection

The ISO-on-TCP transport connection is characterized by the following features:

- Corresponds to the standard TCP/IP (Transmission Control Protocol/Internet Protocol) with the RFC 1006 extension in accordance with layer 4 of the ISO reference model. RFC 1006 describes how the services from ISO layer 4 can be mapped on TCP
- Communication to a partner (for example PC or external system) which supports sending and receiving data in accordance with ISO-on-TCP
- Usable subnet type: Industrial Ethernet
- The receipt of data is confirmed using an acknowledgement
- Interface in the STEP 7 user program: SEND/RECEIVE and FETCH/WRITE;
- ISO-on-TCP transport services available as C functions on the PC

TCP connection

The TCP connection is characterized by the following features:

- Corresponds to the standard TCP/IP (Transmission Control Protocol/Internet Protocol):
- Communication with a partner (for example PC or third-party system) that supports sending and receipt of data in compliance with TCP/IP
- Usable subnet type: Industrial Ethernet
- Interface in the STEP 7 user program: SEND/RECEIVE and FETCH/WRITE;
- TCP/IP implementation of the PC operating system can usually be used

UDP connection

The UDP connection (User Datagram Protocol) is characterized by the following features:

- Usable subnet type: Industrial Ethernet (TCP/IP protocol)
- Unsecured transmission of related data fields between two nodes
- Interface in the S7 user program: SEND/RECEIVE.

E-mail connection

The e-mail connection is characterized by the following features:

- Usable subnet type: Industrial Ethernet (TCP/IP protocol)
- Enables for example the sending of for example process data from data blocks via e-mail using a CP with IT functionality (IT-CP)
- The mail server with which all e-mails sent by an IT-CP are served is defined by the e-mail connection.

FTP connection (specially set up TCP connection)

You use the FTP connection to process FTP order sequences.

When using the file transfer functions (FTP), CPs with IT functionality allow you to transfer files to and from the S7 devices.

HMI connection

Auto-Hotspot

See also

USEND: Send data uncoordinated (Page 7371)

URCV: Receive data uncoordinated (Page 7373)

BSEND: Send data in segments (Page 7376)

BRCV: Receive data in segments (Page 7379)

USEND: Send data uncoordinated (Page 5043)

URCV: Receive data uncoordinated (Page 5044)

BSEND: Send data in segments (Page 5046)

BRCV: Receive data in segments (Page 5048)

Selecting connection types

Selecting the connection type

Which type of connection you select depends on the required transmission properties, as well as on the subnet and transmission protocol by means of which the connection is established. In addition, the device type at each connection endpoint limits the connection types.

Which instructions or services you use at the interface (connection endpoints), depends on the connection type and the device type.

The following table shows these dependencies:

Connection type	Subnet type / protocol	Device type: Connection between SIMATIC ...	Interface
S7 connection	MPI, PROFIBUS, Industrial Ethernet	S7 - S7, S7 - PG/PC, S7 - PG/PC with WinCC, PG/PC - PG/PC S7 - partner in another project (S7, PG/PC with WinCC)	**Instructions for S7:** USEND, URCV, BSEND, BRCV, GET, PUT, START, STOP, RESUME, STATUS, USTATUS **PG/PC:** OPC services or programming interfaces
FDL connection	PROFIBUS (FDL protocol)	S7 - S7, S7 - S5, S7 - PC/PG, S7 - external device, PG/PC - PG/PC S7 - partner in another project (S7, S5, PG/PC, external device)	**Instructions for S7:** AG_SEND, AG_RECV, AG_LSEND, AG_LRECV **PG/PC:** OPC services
ISO transport connection	Industrial Ethernet (ISO transport protocol)	S7 - S7, S7 - S5, S7 - PC/PG, S7 - external device, PG/PC - PG/PC S7 - unspecified S7 - partner in another project (S7, S5, PG/PC, external device, unspecified)	**Instructions for S7:** AG_SEND, AG_RECV, AG_LSEND, AG_LRECV AG_LOCK AG_UNLOCK **PG/PC:**OPC services
ISO-on-TCP connection	Industrial Ethernet (TCP/IP protocol)	S7 - S7, S7 - S5, S7 - PC/PG, S7 - external devices, PG/PC - PG/PC, S7 - unspecified S7 - partner in another project (S7, S5, PG/PC, external device, unspecified)	**Instructions for S7:** AG_SEND, AG_RECV, AG_LSEND, AG_LRECV AG_LOCK AG_UNLOCK **PG/PC:**OPC services

Connection type	Subnet type / protocol	Device type: Connection between SIMATIC ...	Interface
TCP connection	Industrial Ethernet (TCP/IP protocol)	S7 - S7, S7 - S5, S7 - PC/PG, PG/PC - PG/PC S7 - external device, S7 - unspecified S7 - partner in another project (S7, S5, PG/PC, external device, unspecified)	**Instructions for S7:** AG_SEND*, AG_RECV*, AG_LSEND**, AG_LRECV** AG_LOCK AG_UNLOCK **PG/PC:**OPC services
UDP connection	Industrial Ethernet (TCP/IP protocol)	S7 - S7, S7 - S5, S7 - PC/PG, S7 - external device, S7 - unspecified S7 - partner in another project (S7, S5, PG/PC, external device, unspecified)	**Instructions for S7:** AG_SEND, AG_RECV, AG_LSEND, AG_LRECV
E-mail connection	Industrial Ethernet (TCP/IP protocol)	S7 - unspecified (S7 - mail server), S7 - PC/PG (S7 - mail server)	**Instructions for S7:** AG_SEND, AG_LSEND
FTP connection (specially set up TCP connection)	Industrial Ethernet (TCP/IP protocol)	S7 - unspecified	**Instructions for S7:** FTP_CONNECT, FTP_STORE, FTP_RETRIEVE, FTP_DELETE, FTP_QUIT

Connection to broadcast and multicast nodes

Note

For special connection types, there is the option of selecting not just one connection partner but several (broadcast and multicast nodes). You can select the connection partner "Broadcast to all..." or "... multicast nodes" in the dialog box for entering a new connection.

- You can set up a connection to "all broadcast nodes" (sending to all broadcast recipients at the same time) for connections of the FDL and UDP connection type.
- You can set up a connection to "all multicast nodes" (sending to several nodes at the same time) for connections of the FDL and UDP connection type.

Selection of connection types

Selection of connection type

You must decide on a communication service according to your automation task. The selection of communication service affects the following:

- The functionality that will be available
- The time at which the connection will be established
- The activities you will have to perform (e.g., configuring of connections, programming of instructions)

The table below provides an overview of the available communication services.

Communication service	Functionality	Via interface:		
		PN	DP	Serial
PG communication	For commissioning, test, diagnostics	X	X	-
HMI communication	For operator control and monitoring	X	X	-
I/O communication	Data exchange between e.g.: • IO controllers and IO devices • DP master and DP slave	X	X	-
S7 communication	Data exchange between client and server or between clients: Configuration of connection required Using PUT/GET, BSEND/BRCV, or USEND/URCV instructions	X	X	-
Open communication via TCP/IP	Data exchange via PROFINET with TCP/IP protocol With connection configuration: TSEND_C/TRCV_C or TSEND/TRCV instructions By means of user program only: TCON, TSEND, T_RCV, T_DISCON, or TSEND_C and TRCV_C instructions	X	-	-
Open communication via ISO-on-TCP	Data exchange via PROFINET with ISO-on-TCP protocol With connection configuration: TSEND_C/TRCV_C or TSEND/TRCV instructions By means of user program only: TCON, TSEND, T_RCV, T_DISCON, or TSEND_C and TRCV_C instructions	X	-	-
Open communication via UDP	Data exchange via PROFINET with UDP protocol With connection configuration: TSEND_C/TRCV_C or TSEND/TRCV instructions By means of user program only: TCON, TSEND, T_RCV, T_DISCON, or TSEND_C and TRCV_C instructions	X	-	-

Communication service	Functionality	Via interface:		
		PN	DP	Serial
Open communication via ISO (CPs with PN interface only)	Data exchange via PROFINET with ISO protocol With connection configuration: TSEND_C/TRCV_C or TSEND/TRCV instructions By means of user program only: TCON, TSEND, T_RCV, T_DISCON, or TSEND_C and TRCV_C instructions	X	-	-
Open communication via e-mail	Data exchange via e-mail By means of user program only: TMAIL_C instruction	X	-	-
Open communication via FTP (CPs with PN interface only)	Data exchange via FTP (File Transfer Protocol) By means of user program only: FTP_CMD instruction	X	-	-
Open communication via FDL (CMs with DP interface only)	Data exchange via PROFIBUS with FDL protocol With connection configuration: TSEND_C/TRCV_C or TSEND/TRCV instructions By means of user program only: TCON, TSEND, T_RCV, T_DISCON, or TSEND_C and TRCV_C instructions	-	X	-
Point-to-point connection	Data exchange via point-to-point with Freeport, 3964 (R), USS, or Modbus protocol Using SEND_P2P/RCV_P2P or MB_COMM_LOAD, MB_MASTER, MB_SLAVE instructions	-	-	X
S7 routing	Data exchange across network boundaries, e.g., for parameter assignment, test, and diagnostic purposes	X	X	-
Data record routing	Extension of S7 routing, e.g., for parameter assignment and diagnostics of field devices on PROFIBUS by an engineering system with a PROFINET interface.	X	X	-
Web server	Data exchange via the Internet, e.g., for diagnostics	X	-	-
SNMP (Simple Network Management Protocol)	Data exchange via SNMP standard protocol, for network diagnostics and parameter assignment	X	-	-
Time-of-day synchronization	Via NTP (Network Time Protocol); CPU is client	X	-	-
	CPU is time master or times slave	-	X	-

Connection to broadcast and multicast nodes

Note

For special connection types, there is the option of selecting not just one connection partner but several (broadcast and multicast devices). You can select the connection partner "To all broadcast..." or "... multicast devices" in the dialog box for entering a new connection.

- You can set up a connection to "all broadcast devices" (sending to all broadcast recipients at the same time) for FDL and UDP connection types.
- You can set up a connection to "all multicast devices" (sending to several devices at the same time) for FDL and UDP connection types.

Instructions for communication tasks

Interface to user program

Special instructions are available for your communication interface in the SIMATIC S7-300/400/1500 user program. Here is an overview of the instructions for the individual connection types.

The PUT/GET instructions are used for the S7-1500. See the following tables for the instructions of the S7-300/400.

Instructions for S7 connections

The system function blocks assigned to the instructions are integrated in the CPUs of the S7-400.

For S7-300 with more recent CPUs and CPs, you have the option of running the S7 communication actively (i.e. as client) via the CP's interface. The instructions have the same designation as the S7-400 SFBs, but must be called cyclically in the S7-300 CPU user program.

The CP must support the client function for S7 communication.

An S7-300 CPU with integrated PROFINET interface can also be configured as a client for S7 communication. The same blocks from the above scenario using S7-300 and CP are used. The client functionality is only available at the PROFINET interface.

Name	Brief description
USEND URCV	Uncoordinated exchange of data via a send and receive SFB Max. length SFB 8/9: 440 bytes split into 4x100 bytes Max. length FB 8/9: 160 bytes
BSEND BRCV	Exchange of data blocks of variable lengths between a send SFB and a receive SFB Max. length SFB 12/13: 64 KB Max. length FB 12/13: 32 KB
GET	Read data from a remote device Max. length SFB 14: 400 bytes split into 4x100 bytes Max. length FB 14: 160
PUT	Write data to a remote device Max. length SFB 15: 400 bytes split into 4x100 bytes Max. length FB 15: 160
START	Run a restart (warm start) in a remote device
STOP	Put a remote device in "STOP" mode
RESUME	Run a resume process in a remote device
STATUS	Specific querying of the status of a remote device
USTATUS	Receipt of status messages from remote device
CONTROL	Query the status of the connection which belongs to an SFB instance
C_CNTRL	Query the status of a connection (for S7-300 CPUs)

Instructions for S7 communication (Page 7356)

Instructions for point-to-point connections

You can use the BSEND, BRCV, GET, PUT and STATUS SFBs for connections of the point-to-point connection type (see above table).

The PRINT SFB can also be used:

Name	Brief description
PRINT	Send data to a printer

Instructions for FDL, ISO-on-TCP, UDP and TCP connections, as well as e-mail connection

Name	Brief description
AG_SEND	Sends data via a configured connection to the communication partner (with older CPs and with CPs for S7-400, the following limitation applies to this FC: <= 240 bytes)
AG_RECV	Receives data via a configured connection from the communication partner (with older CPs and with CPs for S7-400, the following limitation applies to this FC:<= 240 bytes, not e-mail)
AG_LSEND	Sends data via a configured connection to the communication partner
AG_LRECV	Receives data via a configured connection to the communication partner (not e-mail)
AG_LOCK	Lock external data access using FETCH/WRITE (not with UPD, e-mail)
AG_UNLOCK	Unlock external data access using FETCH/WRITE (not with UPD, e-mail)
AG_CNTRL	Diagnose connections via the user program

Instructions for open communications services (SEND/RECEIVE interface) (Page 7715)

Instructions for open communications services (SEND/RECEIVE interface) (Page 7853)

Instructions for FTP services via TCP connections (FTP connection)

Name	Brief description
FTP_CONNECT	Establish FTP connection
FTP_STORE	Transfer data block (file DB) from FTP client to FTP server
FTP_RETRIEVE	Transfer file from FTP server to FTP client
FTP_DELETE	Delete file from FTP server
FTP_QUIT	Quit the FTP connection stated by the ID

Instructions for FTP services (Page 7828)

Instructions for programmed open communication services with ISO-on-TCP, UDP and TCP connections, as well as for e-mail connection via CPs

Name	Brief description
IP_CONFIG	Set up program controlled communication connections and IP configuration of the CP.

Instructions for programmed connections (Page 7809)

Instructions for programmed open communication services (Open User Communication) with ISO-on-TCP, UDP and TCP connections via CPU or CP*)

*) Note: With these instructions, communication via CP is only possible using ISO-on-TCP.

Name	Brief description
TCON	Connection establishment
TDISCON	Disconnection
TSEND	Sending data
TRCV	Receiving data
TUSEND	Sending of data; connectionless protocol UDP in accordance with RFC 768
TURCV	Receiving of data; connectionless protocol UDP in accordance with RFC 768

Overview (Page 7410)

Connection resources

Configuration limits in the connection configuration

Connection resources

Some communication services require connections. During operation, communication connections utilize memory and program resources of the CPUs, CPs, and CMs involved in the communication path (e.g., memory areas in the CPU operating system). Within a controller (PLC), this applies, for example, to the CPU as connection end point and to the extensions with CPs as interfaces to the subnets. In most cases, one resource per CPU/CP/CM is assigned for a connection. In HMI communication, up to 3 connection resources are required for each HMI connection.

The number of connection resources available differs for the various automation systems:

- In S7-300 automation systems, the CPU defines the number of available resources for connections. The maximum number of resources for connections can be expanded by adding CPs.
- In S7-400 automation systems, the CPU defines the number of available resources for S7 connections. The resources can also be used for other connection types, in which case the maximum number of resources for other connection types can be expanded by adding CPs.
- In S7-1500 automation systems, the CPU that is used defines the applicable upper limit of connection resources for the whole automation system. The CPU itself accounts for a portion of these connection resources, and additional resources accrue via CPs and CMs. However, regardless of how many CPs and CMs are added, the total number of connection resources cannot exceed the upper limit of connection resources defined by the CPU type.

Available connection resources

The maximum configuration limit within the automation system is always defined by the CPU, either through the system upper limit defined by the CPU (S7-1500) or through the sum of

usable resources of the CPUs and CPs (S7-300/400). In most cases, the available connection resources of a CPU/CP/CM are automatically distributed to their interfaces, as required.

Each CPU brings along reserved connection resources for programming device, HMI, and Web server communication in its configuration. In addition there are available resources that can be used for any HMI communication, S7 communication, or open communication (S7-1500).

You can determine which resources are available to you based on the configuration of the devices as follows:

- Selection of the device type at connection end point (CPU)
- Number and type of additional communication modules (CP, CM)

When will connection resources be assigned?

Which resources are actually used in operation, depends on which communication functions are used at any specific time and which communication connections are enabled at any specific time.

Therefore, only limited checks of compliance with specific configuration limits and available communication resources can be performed at the time of configuration. The connection configuration offers the greatest possible support for this.

Which factors influence the configuration limit in the connection configuration?

You determine the communication resources with the device configuration:

- Number of available S7 connection resources in the CPU (S7-300/400)
- Number of connection resources for S7 communication and open communication in the station (S7-1500)
- Number of protocol-specific connection resources in the CPs/CMs
- Number of available CPs/CMs
- Automatic, programmed, or configured connection setup

What are the characteristics of the connection configuration?

During input, the connection configuration checks the following circumstances and behaves as follows:

- During the creation of S7 connections
  Is the maximum number of available S7 connections assigned? If yes, it is not possible to configure another S7 connection via the selected interface.
  For the S7-300, it is possible to increase the configuration limit by adding CPs.
- During the creation of any connection types
  Is the maximum number of protocol-specific connection resources in the interface (CP) assigned? If yes, it is not possible to configure another connection of the selected type via the selected interface.
  For the S7-300/400, it is possible to increase the configuration limit by adding CPs.
- During the creation of any connection types with OPC (PC stations)
  In the case of connections with OPC as connection end point, more connections can generally be configured than can be operated at any one time in the CP used in the PC station. That is because the connection properties differentiate between "Maintain permanently" and "Set up on demand".

When the connection configuration is compiled, a warning or, in the case of a non-assignable interface, an error is output, if appropriate.

Note

S7 connection resources are exhausted

If the maximum number of S7 connection resources is reserved, no S7 connection resources additionally required for data transmission between CPU and CP are in operation or possibly available for connections. The corresponding communication jobs are then refused.

You should therefore make sure that there is a sufficient number of non-assigned S7 connection resources, depending on the total assigned communication functions.

You can find the number of the reserved S7 connection resources for the selected S7-300 or S7-1500 CPU in the area navigation under "Connection resources". For the S7-300, only the connection resources of the CPU are shown. For the S7-1500, you also see the resources of the CPs and CMs.

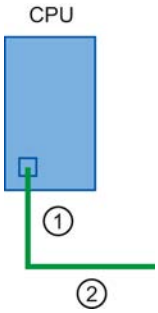
Consumption of connection resources

Every connection needs connection resources for the endpoint and transition point (e.g., CP, CM) on the devices involved. The number of available connection resources is CPU/CP/CM-specific.

If all the connection resources of a communication partner are assigned, no new connections can be established. Each type of communication is considered in detail below. Any number of combinations are possible however as long as the available connection resources are taken into account.

S7 connections

With S7 connections via the **integrated** MPI-/PROFIBUS-DP/PN interface, one connection resource is assigned for the endpoint on the CPU per S7 connection. This applies to all S7-300/400/1500 CPUs.



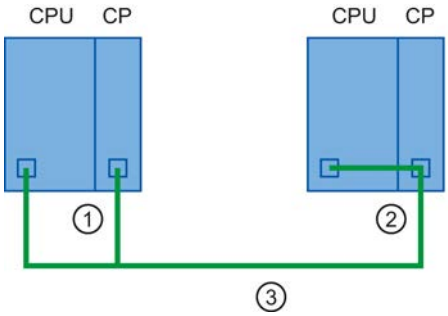
① Assigned S7 connection resource on the CPU

② Subnet (e.g., MPI, PROFIBUS-DP, or Industrial Ethernet/PROFINET)

- S7-300/400 CPU
  One S7 connection resource is assigned on the CPU.
- S7-1500 CPU
  One S7 connection resource is assigned on the CPU and thus also one S7 connection resource on the station.

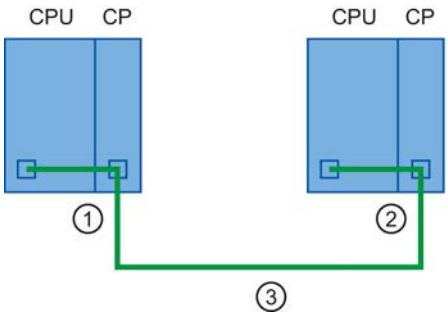
With S7 connections via an **external** CP interface, a distinction should be made between the following scenarios:

- S7-300 CPU
  - S7 connection configured at one end
    One S7 connection resource is assigned on the CPU (for the endpoint) and one S7 connection resource on the CP (transition point) for each S7 connection.



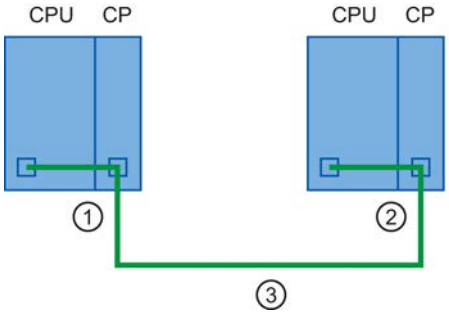
① One S7 connection resource in the CPU and one in the CP of the local end point
② Connection partner
③ Subnet, e.g., Industrial Ethernet

  - S7 connection configured at both ends
    One S7 connection resource is assigned on the CP for each S7 connection.



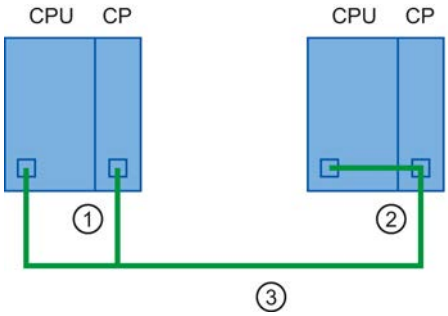
① One S7 connection resource in the CP of the local end point
② Connection partner
③ Subnet, e.g., Industrial Ethernet

- S7-400 CPU
  One S7 connection resource is assigned on the CPU (for the endpoint) and one S7 connection resource on the CP (transition point) for each S7 connection.



① One S7 connection resource in the CPU and one in the CP of the local end point
② Connection partner
③ Subnet, e.g., Industrial Ethernet

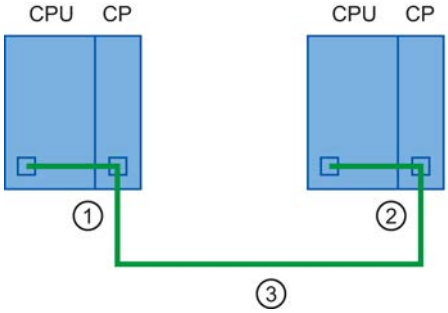
- S7-1500 CPU
  - S7 connection configured at one end
    One S7 connection resource is assigned on the CPU and thus also one S7 connection resource on the station for each S7 connection.



①
② Connection partner
③ Subnet, e.g., Industrial Ethernet

  - S7 connection configured at both ends

One S7 connection resource is assigned on the CPU and thus also one S7 connection resource on the station for each S7 connection.



① One S7 connection resource in the CP and, thus, also one in the station
② Connection partner
③ Subnet, e.g., Industrial Ethernet

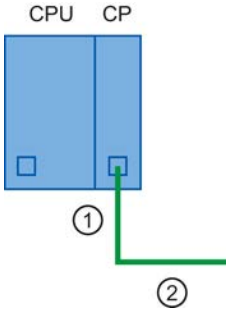
Note

The number of reserved S7 connection resources for the selected CPU for certain CPU types of the S7-300, and for the selected CPU and its connected communication modules for the S7-1500, can be seen in the area navigation under "Connection resources". S7 connection resources for OP/PG and S7 basic communication can be reserved here.

Connections for open communication services (SEND/RECEIVE interface)

Communication via the SEND/RECEIVE interface is solely via CPs. For this, one protocol-specific connection resource is allocated for the endpoint per connection (i.e., FDL, ISO transport ISO on TCP, UDP and TCP connection).

Here, no connection resources are required for the connection on the CPU.



① 1 assigned connection resource in the CP
② Subnet, e.g., Industrial Ethernet

Exception: on the S7-400 CPU, connections with the property "SPEED SEND/RECEIVE" require an additional S7 connection resource for each "SPEED SEND/RECEIVE" connection on the CPU.

Assignment of connection resources

Setting up the connection

Often, you can decide between a configured setup or a programmed setup. The programmed setup makes it easier to release connection resources following data transfer. With configured setup, you receive help in managing connection resources in STEP 7.

Table 8-1 Setting up the communication service

Communication service	Automatic setup	Programmed setup	Configured setup
PG communication	X	-	-
HMI communication	X	-	X
S7 communication	-	X	X
Open communication via TCP/IP	-	X	X
Open communication via ISO-on-TCP	-	X	X
Open communication via UDP	-	X	X
Open communication via ISO	-	X	X
Open communication via e-mail	-	X	-
Open communication via FTP	-	X	-
Open communication via FDL	-	X	X
S7 routing	-	-	X

Assignment of connection resources according to the connection setup

Connection resources are assigned in different ways, depending on the type of connection setup:

- Configured setup
  In the case of configured connections, the connection resource is assigned as soon as the hardware configuration has been downloaded to the CPU. For OPC servers, the resources are assigned, as required.
- Programmed setup
  The call of the appropriate instructions for establishing the connection (using TSEND_C or T_CON) triggers assignment of a connection resource.
- Automatic setup
  As soon as you have connected the programming device or HMI device to a CPU physically and online, the connection resources will be assigned.

Connection resources for configured setup

You set up the connection in the network view of the hardware and network editor in the context of a CPU.

Following the data transfer, the connection is not terminated, i.e., the connection resources remain permanently assigned. To terminate the connection you must delete the connection configuration and download the modified configuration to the CPU.

OPC connections are automatically established and terminated as needed.

If you have **configured** the connections, compliance with the maximum possible number of connection resources in the automation system will be monitored.

Connection resources for programmed setup

You set up the connection in the program editor in the context of a CPU by parameter assignment of instructions for communication, e.g., TSEND_C.

Once the connection is established by the TCON or TSEND_C instruction, the connection will be terminated through appropriate parameter assignment of the TSEND_C instruction or by calling the TDISCON instruction after the data transfer. If the connection is terminated, the connection resources in the CPU/CP/CM are available again. If the connection is retained and another data transfer is performed, the processing time of the instructions will be shorter because the connection does not have to be reestablished.

If the establishment and termination of connections are **programmed** in the user program, you are responsible for ensuring compliance with the automation system limits. If necessary, you will receive a communication error at the "STATUS" output parameter of the TSEND_C or T_CON instruction and can obtain more detailed information from the connection diagnostics.

Connection resources for automatic setup

The connection will be set up automatically (e.g., programming device or HMI connection) as soon as the PG/PC interface has been physically connected to an interface of the CPU and the assignment to the interface has been made in the "Go online" dialog.

For **automatically** set up connections, compliance with the connection resources reserved in the automation system for these connections is monitored.

See also

TCON: Establish communication connection (Page 5083)

TDISCON: Terminate communication connection (Page 5091)

TCON: Establish communication connection (Page 7431)

TDISCON: Terminate communication connection (Page 7434)

TSEND_C: Send data via Ethernet (Page 5050)

TSEND_C: Send data via Ethernet (Page 5054)

Display of connection resources

Connection resources in the inspector window

You will find an overview of the reserved and available connection resources of an S7-1500 automation system in the inspector window of the hardware and network editor under the CPU properties.

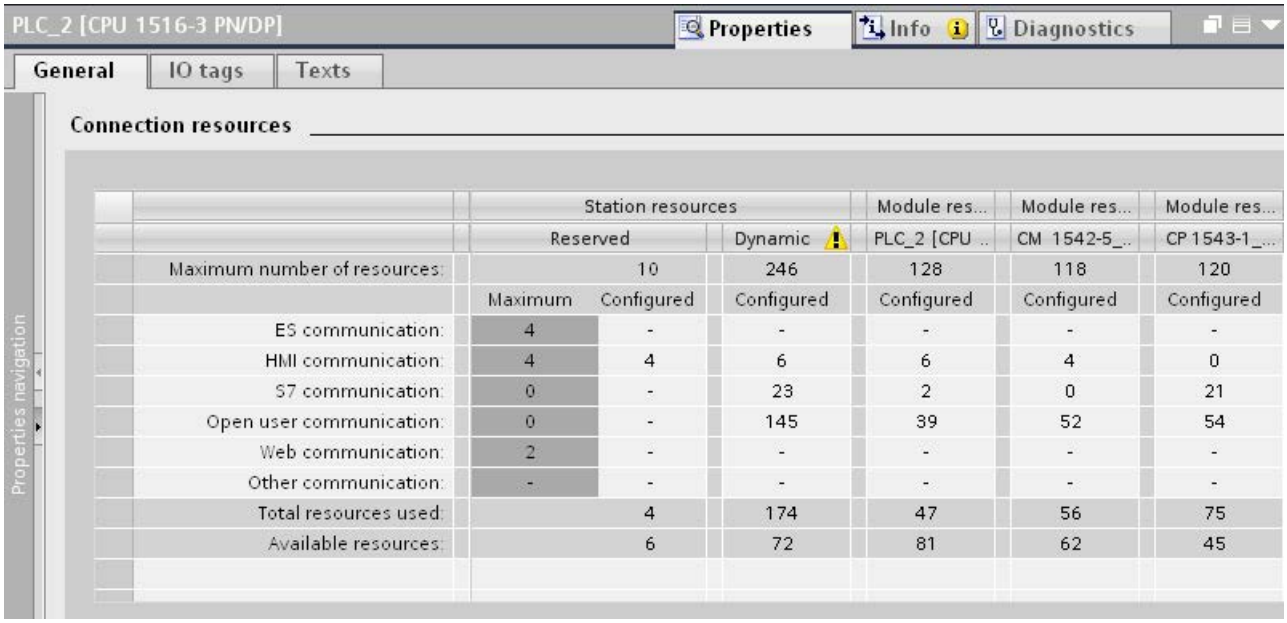
The display of connection resources distinguishes between offline and online views.

Display of connection resources in the offline view

The offline view displays the reserved and available connection resources of the configured connections.

The maximum available number of connection resources in the automation system is dependent on the CPU used. If the CPU-dependent system limit is reached, it makes no difference whether the CPU, CPs, and CMs have additional connection resources. The connection resources are exhausted for this device.

In the following figure, 246 dynamic connection resources and 10 reserved connection resources are available for the station.



	Station resources			Module res...	Module res...	Module res...
	Reserved		Dynamic	PLC_2 [CPU ...	CM 1542-5_...	CP 1543-1_...
Maximum number of resources:		10	246	128	118	120
	Maximum	Configured	Configured	Configured	Configured	Configured
ES communication:	4	-	-	-	-	-
HMI communication:	4	4	6	6	4	0
S7 communication:	0	-	23	2	0	21
Open user communication:	0	-	145	39	52	54
Web communication:	2	-	-	-	-	-
Other communication:	-	-	-	-	-	-
Total resources used:		4	174	47	56	75
Available resources:		6	72	81	62	45

The 10 connection resources are reserved for:

- 4 for programming device communication, e.g., for test and diagnostic functions or downloading to the CPU
- 4 for HMI communication, for assignment by the first configured HMI connections
- 2 for communication via Web server, which will be assigned online, if necessary

A maximum of 246 connection resources are available dynamically, i.e., for various communication services in the automation system. Of these, 174 connection resources have

already been configured for various communication services and modules. A total of 72 connection resources remain available for the automation system.

The columns of the module-specific connection resources contain the following for the CPU, CPs, and CMs:

- Number of maximum available connection resources for CPU/CP/CM
- Number of resources required for the various configured communication connections
- Number of all configured and thus "consumed" connection resources
- Number of connection resources still available

The CPU provides a maximum of 128 connection resources. The following have been configured for the CPU: 6 resources for HMI communication connections, 2 resources for S7 communication connections, and 39 open communication connections, which altogether utilize 47 resources in the CPU. 81 resources of the CPU remain available. For the CP 1543-1, 56 of the 118 available resources are assigned and 62 resources are still available. For the CM 1542-5, 75 of the 120 available resources are utilized and 45 resources are still available.

The warning triangle in the "Station resources" column in the "Dynamic station resources" field signals that the sum of the maximum available connection resources of CPU, CP, and CM (= 366 connection resources) exceeds the station limit of 256.

If the number of available resources is 0, which means all resources are already reserved for connections, a warning triangle is displayed in the line "Available resources" of the "Dynamic station resources" column. You cannot create any additional connections in this case.

Note

Exceeding of available connection resources

STEP 7 signals the exceeding of the station-specific connection resources with an error icon. When this occurs, you must either use a CPU with a greater maximum number of available connection resources or reduce the number of communication connections.

Reference

For a detailed example of the calculation of connection resources, see: Example for calculating the connection resources (Page 588)

Display of connection resources in the online view

If you have selected a CPU with an online connection in the network view of the hardware and network editor, you can display the current status of the assigned and unassigned connection

resources for the automation system in the inspector window of the "Diagnostics" area on the "Connection information" tab.

PLC_2 [CPU 1516-3 PN/DP] | Properties | Info | Diagnostics

Device information | Connection information | Alarm display

Connection resources

	Station resources					Module resources	
	Reserved			Dynamic		PLC_2 [CPU 1516-3 PN/DP]	
Maximum number of resources:		10	10	194	194	86	86
	Maximum	Configured	Used	Configured	Used	Configured	Used
ES communication:	4	-	4	-	0	-	0
HMI communication:	4	4	4	4	4	8	8
S7 communication:	0	0	0	72	68	34	34
Open user communication:	0	0	0	118	118	45	45
Web communication:	2	-	2	-	0	-	0
Other communication:	-	-	0	-	0	-	0
Total resources used:		4	10	194	190	82	82
Available resources:		6	0	0	4	4	4

The online view of the "Connection resources" table not only displays the reserved and dynamic connection resources of the configured connections but also includes columns containing the connection resources of the programmed and automatically set-up connections that are currently assigned in the station. Thus, the online view displays **all** assigned connection resources in the automation system, regardless of how the connection was set up. The "Other communication" row displays connection resources assigned for communication with external devices and communication via data record routing.

The table will be updated automatically. If there are inconsistencies between the configured hardware configuration and the hardware configuration available online, a corresponding message will be output.

See also

Resources for routing functions (Page 593)

Example for calculating the connection resources

Introduction

This example for the table displaying connection resources shows you how the individual table entries are put together.

Requirement

- 2x CPU 1516-3 PN/DP with one CP 1543-1 and one CM 1542-1 each as connection partners for purposes of extending the configuration limit.
- 1x TFT display MP 377 12" Touch with WinAC MP 377 as a connection partner for HMI connections.
- The devices are networked together.

Creating the connections

Once the connection partners (CPUs, CPs, CMs, MP 377) have been created and networked in the network view of the hardware and network editor, follow these steps:

1. Switch to connection mode in the network view.
2. Create the following connections:
   - S7 connections: 2x via CPU, 21x via CM
   - TCP connections: 2x via CPU, 4x via CM
   - ISO connection: 6x via CP
   - ISO-on-TCP: 3x via CPU, 4x via CM
   - UDP connections: 4x via CPU, 106x via CP
   - HMI connections: 1x via CPU, 2x via CP

Result

The display of connection resources shows you not only the available resources but also the resources required for the created connections.

The available resources of devices and modules are highlighted in the following figure.

	Station resources			Module res...	Module res...	Module res...
	Reserved		Dynamic ③	PLC_2 [CPU ..	CP 1543-1_...	CM 1542-5_..
Maximum number of resources:	① 10		246	② 128	118	118
	Maximum	Configured	Configured	Configured	Configured	Configured
ES communication:	4	-	-	-	-	-
HMI communication:	4	4	5	3	6	0
S7 communication:	0	-	23	2	0	21
Open user communication:	0	-	129	9	112	8
Web communication:	2	-	-	-	-	-
Other communication:	-	-	-	-	-	-
Total resources used:		4	157	14	118	29
Available resources:		6	89	114	0	89

① Depending on the CPU, at most 256 connections are available for this station. Ten (10) resources are reserved for particular connection types and 246 resources are dynamically available.

② The CPU provides 128 connection resources, and the CP and CM each provide another 118 resources.

③ CPU, CP, and CM altogether provide 128 + 118 + 118 = 364 resources. Because more connection resources are provided via the devices and modules than the station can operate with its maximum limit of 256 connection resources, a warning icon is displayed.

The following figure highlights the resources utilized by the created connections:

	Station resources			Module res...	Module res...	Module res...
	Reserved		Dynamic	PLC_2 [CPU ..	CP 1543-1_...	CM 1542-5_...
Maximum number of resources:		10	246	128	118	118
	Maximum	Configured	Configured	Configured	Configured	Configured
ES communication:	4	-	-	-	-	-
HMI communication:	4	4	5	3	6	0
S7 communication:	0	-	23	2	0	21
Open user communication:	0	-	129	9	112	8
Web communication:	2	-	-	-	-	-
Other communication:	-	-	-	-	-	-
Total resources used:		4	157	14	118	29
Available resources:		6	89	114	0	89

① These are the resources assigned via the HMI connections. In this example, one HMI connection utilizes 3 connection resources. For the whole station, 3 + 6 = 9 connection resources are required for the HMI communication.

② Four (4) resources are reserved for the HMI communication.

③ The 9 resources required for HMI communication are broken down as follows: For the first 4 resources, the resources already reserved for HMI communication will be used. The other 5 required resources will be taken from the dynamic resources.

④ These are the resources assigned via the S7 connections. An S7 connection occupies 1 connection resource. For the whole station, 2 + 21 = 23 connection resources are required.

⑤ For the S7 communication, there are no reserved resources. All 23 required resources come from the dynamically available resources.

⑥ These are the resources of the Open User Communication (for TCP, FDL, ISO, ISO-on-TCP, and UDP). For the whole station, 9 + 112 + 8 = 129 connection resources are required.

⑦ For the Open User Communication, there are no reserved resources. All 129 required resources come from the dynamically available resources.

Calculation of connection resources

Based on the available and utilized resources of individual devices and modules, the resources that are still free and the total resources used are calculated:

	Station resources			Module res...	Module res...	Module res...
	Reserved		Dynamic	PLC_2 [CPU ..	CP 1543-1_...	CM 1542-5_..
Maximum number of resources:		10	246	128	③ 118	118
	Maximum	Configured	Configured	Configured	Configured	Configured
ES communication:	4	-	-	-	-	-
HMI communication:	4	4	5	3	6	0
S7 communication:	0	-	23	① 2	0	21
Open user communication:	0	-	129	9	112	8
Web communication:	2	-	-	-	-	-
Other communication:	-	-	-	-	-	-
Total resources used:		4	157	② 14	④ 118	29
Available resources:		6	89	114	⑤ 0	89

① All required resources for the respective communication types of the CPU are displayed here. Three (3) resources are required for HMI communication, 2 for S7 communication, and 9 for the connection types of the Open User Communication.

② The total number of resources required is listed below the required resources for the communication types. For the CPU, therefore, 3 + 2 + 9 = 14 resources are required.

③ The CP 1543-1 has 118 available connection resources. If these are utilized, no more connections can be created for the CP.

④ For the CP 1543-1, 118 connection resources are already utilized by the 2 HMI connections (3 resources each = 6 resources) and 112 resources by connections of the open communication.

⑤ The CP 1543-1 has no more free resources. No more connections can be created for the CP 1543-1.

The number of resources still available and the total number of resources used is also displayed for the whole station:

	Station resources			Module res...	Module res...	Module res...
	Reserved		Dynamic	PLC_2 [CPU ..	CP 1543-1_...	CM 1542-5_..
Maximum number of resources:		10	246	128	118	118
	Maximum	Configured	Configured	Configured	Configured	Configured
ES communication:	4	-	-	-	-	-
HMI communication:	4	4	5	3	6	0
S7 communication:	0	① -	23	2	0	21
Open user communication:	0	-	129	9	112	8
Web communication:	2	-	-	-	-	-
Other communication:	-	-	-	-	-	-
Total resources used:		② 4	157	14	118	29
Available resources:		③ 6	89	114	0	89

① All connection resources used (configured), including both reserved and dynamic connection resources, for the individual communication types for the entire station. The values represent the sum of all resources utilized for devices and modules (CPU, CP, CM).

② All connection resources used (configured), including both reserved and dynamic connection resources, for all communication types for the entire station.

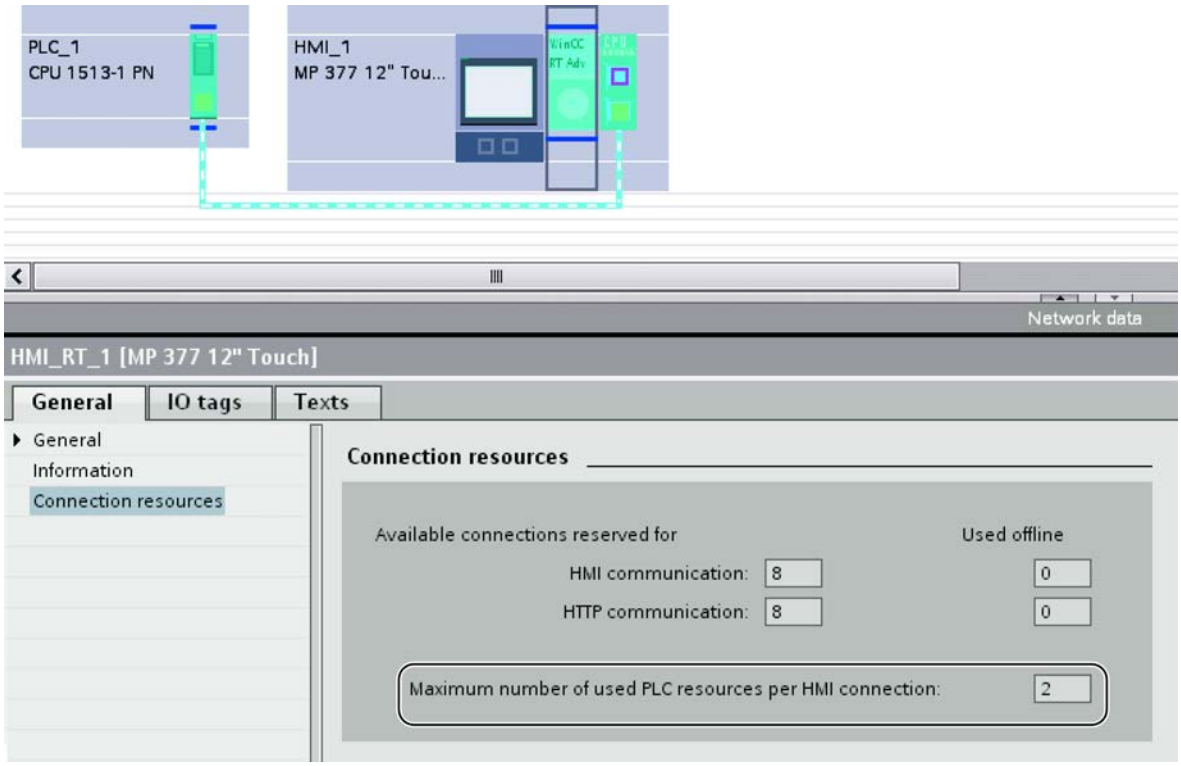
③ Total number of reserved and dynamic connection resources still available. The values are obtained by subtracting the number of resources used from the maximum number of resources that can be configured.

Resources for HMI communication

Connection resources for HMI communication

For HMI communication, connection resources in the CPU are assigned according to the HMI device used and the applications in which the HMI device is used. For example, a Basic Panel requires up to 2 connection resources in the CPU for an HMI connection, while a Comfort Panel requires up to 3 connection resources in the CPU.

For information regarding the availability and assignment of connection resources for HMI connections, refer to the "Connection resources" properties in the inspector window of the offline view (HMI context). There, you will also find out how many resources are required for an HMI connection:



If the maximum number of available connection resources for an HMI device is exceeded, a corresponding message will be output.

Additional information

To find out how many connection resources an HMI requires in the CPU, refer to the documentation for the HMI device.

Resources for routing functions

S7 connections beyond network boundaries (S7 routing)

For transferring data beyond network boundaries ("S7 routing"), an S7 connection is established between two CPUs extending across multiple subnets. The subnets are interconnected by routers. The router is the direct through-connection point of an S7 connection and is capable of establishing S7 connections. CPUs, CMs, and CPs in S7-1500 can be routers.

The connection resources for S7 routing are taken into consideration in the end points of the communication (CPUs) and displayed in the "Connection resources" table for S7 communication. For this reason, connection resources for the routing must be available at the routers.

NOTICE
Limited resources for S7 routing A maximum of 16 routed S7 connections are established in the CPU 1516. If you configure more routed S7 connections, the additional configured connections are not established.

Data record routing

Data record routing also enables transfer of data beyond network boundaries, from an engineering station connected to PROFINET to various field devices via PROFIBUS.

For data record routing, connection resources are assigned at the routers. Connection resources for data record routing in the end point of communication (CPU) are displayed in the online view of the "Connection resources" table in the "Other communication" row.

Note

Connection resources for S7 routing and data record routing

For S7 routing and data record routing, the appropriate connection resources are assigned in the routers. It is not possible to configure and monitor these connection resources in STEP 7. You are responsible for monitoring compliance with the limits.

Communication connections via CPs

Data volumes and configuration limit

The number of communication connections which are supported by the respective CP can be found in the equipment manual accompanying every CP. The number of connections per device can be increased by adding other CPs.

If several CPs are fitted in a device, an automatic switch is made to the next CP when this limit is exceeded. Specifically assign the connections using the path selection as required.

Ethernet CP

The Ethernet CP can transfer the following data volumes via one connection per order:

	ISO transport	ISO-on-TCP	TCP	UDP
Sending	8192 bytes	8192 bytes	8192 bytes	2048 bytes
Receiving	8192 bytes	8192 bytes	8192 bytes	2048 bytes

Note

Data transfer > 240 byte transfer is supported by the current CPs.

CPs with an older product version support the transfer of data with a data length of up to 240 bytes.

For more information, refer to the details provided in the Ethernet CP equipment manual.

Tasks of the Ethernet CP during online mode

During the transfer of data via a connection, the Ethernet CP takes on the following tasks:

- Receiving
  Receiving data from the Ethernet and forwarding to the user data area in the CPU
- Sending
  Transferring data from the user data area of the CPU and sending data via the Ethernet

The connection is established automatically as soon as the partner can be reached.

With a free UDP/FDL connection, the following also applies:

- When receiving
  Entering the sender who sent the message in the order header
- When sending
  Evaluating the order header and addressing the partner

PROFIBUS-CP

The PROFIBUS CP can transfer the following data volumes via one connection per order:

FDL connection type	Specified	Unspecified - free layer 2 access	Broadcast	Multicast
Sending	240 bytes	236 bytes [1)]	236 bytes [1)]	236 bytes [1)]
Receiving	240 bytes	236 bytes [1)]	236 bytes [1)]	236 bytes [1)]

1) The order header also assigns 4 bytes alongside the user data stated

Tasks of the PROFIBUS CP during online mode

When processing the transfer of data via an FDL connection, the PROFIBUS-CP takes on the following tasks:

- With specified connections
  - Receiving
    Receiving data from the PROFIBUS and forwarding to the user data area in the CPU
  - Sending
    Transferring data from the user data area of the CPU and sending data via the PROFIBUS
- With unspecified connections also
  - When receiving
    Entering the sender and FDL service in the order header
  - When sending
    Evaluating the order header and addressing the partner; performing the selected FDL service

Communication connections to PC stations

Connection endpoints of a PC station

On PC stations, you configure connections with one of the following components as connection endpoint:

- OPC server as central communication service provider
  Depending on the available CP, you can use the following connection types:
  - S7 connection
  - ISO connection
  - ISO-on-TCP connection
  - TCP connection
  - FDL connection
- Standard application
  With the S7 connection type, communication is possible via Ind. Ethernet or PROFIBUS, depending on the available CP.

Special features of the connection configuration with OPC server

In the case of connections with OPC as connection endpoint, more connections can generally be configured than can be operated at any time in the CP used in the PC stations.

That is because by the connection properties differentiate between "Maintain permanently" and "Set up on demand".

Accordingly, the number of connections with the property "Maintain permanently" cannot exceed the maximum number of CP connection resources. If it proves possible to establish additional connections on demand, you have to keep free the CP's corresponding connection

resources. Therefore, configure a smaller number connections with the property "Maintain permanently" than the maximum number of CP connection resources permits.

A warning may possibly be indicated during the compiling of the connection configuration.

Additional information

Industrial communication with the PG/PC Band 1 under the entry ID: 42783968 (http://support.automation.siemens.com/WW/view/en/42783968)

Industrial communication with the PG/PC Band 2 under the entry ID: 42783660 (http://support.automation.siemens.com/WW/view/en/42783660)

Commissioning PC stations - Manual under the entry ID: 13542666 (http://support.automation.siemens.com/WW/view/en/13542666)

Unspecified connections

Unspecified connections with Industrial Ethernet

Connections to an as yet unknown device (for example a diagnostics device) are configured as "unspecified" connections. The unspecified connection can be used in different ways as explained below based on the example of an ISO-on-TCP connection; on an ISO transport or TCP connection, the procedure is analogous.

- **Declaring ready for communication - passive connection establishment**
  Connection establishment must be set to "passive".
  The following then applies to the address setting for an ISO-on-TCP connection:
  the remote IP address and the remote TSAP are empty, in other words, they are not relevant for the CP. During connection establishment, every partner is then accepted (partner = connection name that addresses the CP with the correct IP address and TSAP).
  There is also the option of partial specification, in other words, the communication is permitted with any partner that matches the specified TSAP.
- **Connection to a specific station in any project.**
  The following then applies to the address setting for an ISO-on-TCP connection:
  You can specify the remote IP address and the port for any target station. The target station can be inside or outside of the current STEP 7 project.
  Use this variant if you have not created the partner station in the current project.
- **Connection without specifying a port**
  TCP connections are unspecified in the following situations:
  - The local port is not specified (active connection establishment).
  - The remote port is not specified (no active connection establishment).
- **IP addressing using DHCP**
  If you select the option of IP addressing using DHCP, it is not initially possible to create a fully specified connection in the project because the local IP address is unknown. You will therefore have to select "Unspecified" without active connection establishment as the connection type.

The following table lists all the options.

Table 8-2 Setting the "remote" address parameters

Significance of the connection establishment	IP address / MAC address (remote)	TSAP / port (remote)	Active connection establishment
By either partner	Empty	Empty	no
By either partner via a specific TSAP	Empty	Specified	no
To or from a specific partner	Specified	Specified	yes The local port can remain unspecified (but does not need to).
		Unspecified	no

The free UDP connection represents a further variant. With this type of connection, the address of the connection partner remains open in the configuration. The communication stations are determined by address information in the communication job of the user program.

Unspecified connections on PROFIBUS

An unspecified FDL connection with open layer 2 access enables program-controlled addressing of the communication partner and communication between two stations on PROFIBUS with the following properties:

The local station is specified by configuration. The remote node is entered by the user program in the job header of the job buffer with the AG_SEND instruction call. This means that each node on PROFIBUS (PROFIBUS addresses from 0 to 126) can be reached.

Additional information

You will find further information below in the sections describing specific connections.

Free UDP connection (Page 653)

FDL unspecified FDL connection (free layer 2 access) (Page 662)

Programmed communication connections via CP

Range of application

In certain applications it is advantageous not to set up the communication connections or the IP configuration of the CP via the configuration interface, but rather to set up these connections in a program-controlled manner using a specific application.

You have the option of freely configuring the communication connections in a program-controlled manner via industrial Ethernet CPs. Keep in mind the system performance of the CP type you are using.

Typical application cases are, for example, with manufacturers of standard machines. To be able to offer most comfortable interfaces to their clients, these manufacturers adapt the communication services to the respective operator inputs. The end users require no knowledge about how to configure connections.

The IP_CONFIG instruction is available for these applications, which permits the flexible transfer of data blocks with configuration data to an Ethernet CP.

Interaction of programming and configuration

Connections are either configured or configured by the user program as the S7 device is running. A mixed form of these variants is not possible within one CP!

Principle

Configuration data for communication connections are transferred to the CP via the IP_CONFIG instruction in the user program.

The configuration DB can be loaded in this manner in the CP at any time. The previous connections and configuration data (IP address, subnet mask, default router, NTP time server and other parameters) are then overwritten.

The Ethernet CP uses the configuration data to detect whether the communication connections are to be set up using the user program.

Scale

A maximum of 64 connections can be specified in the IP_CONFIG instruction. However, the maximum number of connections supported by the CP type you are using is decisive.

Special features/constraints

- Consistency check only with configured connection
  The connection configuration is linked to consistency checks which are not possible or are only partly possible with the programmed configuration.
- Connection configuration by partner needed
  When configuring specified connections, the connection for the partner is implicitly created during configuration. This is not possible with a programmed configuration. In this case, appropriate consistent connections must be configured for the partners.
- Configuration of IP access protection
  IP access protection can be used to limit communication via the CP of the local S7 device to partners with very specific IP addresses. This parameter assignment also applies to programmed communication connections.
- DHCP/DNS is supported
  IP addressing is also possible via DHCP/DNS for the programmed configuration.

See also

Overview (Page 7809)

Connections with Open User Communication via CPU and CP

Open User Communication via Industrial Ethernet - programmed communication connections

Open User Communication provides Industrial Ethernet services that you use via the instruction interface in the user program. Communication connections are set up via the instruction interface in the user program; it is not necessary to configure the connections.

Services via the CPU's interface

The following Open User Communication services are available with S7-1500 for data exchange via the integrated Ethernet interface of a CP:

- TCP as per RFC 793
- ISO-on-TCP as per RFC 1006
- UDP as per RFC 768 (broadcast or unicast, no support for multicast)

Services via the CPU's interface

The following Open User Communication services are available with S7-300/400/1500 for data exchange via the Ethernet interface of a CP:

- TCP as per RFC 793
- ISO as per RFC 905
- ISO-on-TCP as per RFC 1006
- UDP as per RFC 768 (broadcast or unicast, no support for multicast)

See also

Functional description of instructions for Open User Communication via Industrial Ethernet (Page 7411)

Creating and editing connections

Creating a new connection - overview

Creating a connection - alternatives

You have the following options for creating a connection in the network view:

- Graphic connection configuration
  You drag a connection line to the connection partner or assign a new connection to the previously selected connection partners. This is a simple method for creating individual connections in the network view.
  Missing networking is completed automatically or interactively.
- Interactive connection configuration
  Select the connecting partner you need from the list. This method is better than the graphic method if a large number of connections are to be configured.

You'll find the individual steps for this in the following chapters.

Assigning connection path

Depending on the application, a distinction should be made between the following procedures:

- Selecting the connection endpoint in the source and target device
  The software automatically identifies the best connection path to the target device. This connection can then be called the "standard connection path". The software considers the available connection resources when selecting the path.
  The endpoint can be a CPU or a PC application.
  Note: This variant is only possible with graphic connection configuration.
- Selecting connection paths in the source and target devices
  If intended for the purpose of splitting load between several components (CPs) and possibly also different subnets, you can also specifically configure the path of a connection. You specify a connection path via the connection endpoint and the interface to the subnet.

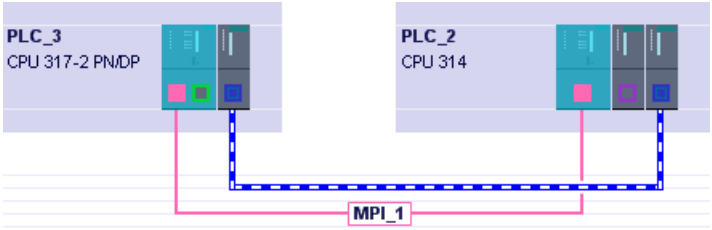
Requirement and result

You have created the devices with CPUs and PC applications and, if applicable, modules with communication capability (CPs) in the network view.

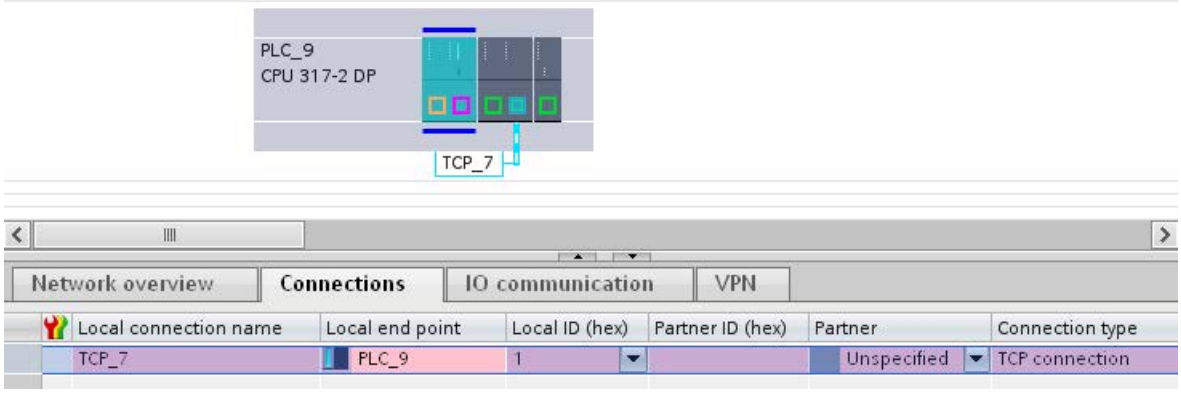
The result of configuration depends on the current configuration of the devices and networking in the project. You will therefore achieve the following results with the methods described in the following sections:

- Connection specified
  If both partners selected are networked in the same subnet, create a fully-specified connection using graphic or interactive selection.
  This fully-specified connection is automatically entered in the connection table of both the local device and that of the partner. The local ID and the partner ID are assigned for this connection. You will need these IDs when programming the communication instructions (value for the block parameter "ID") and the PC applications.
  Exceptions:
  - Unilateral S7 connections
    The connection partner acts as a server on an S7 connection configured at both ends. There is therefore no entry in the connection table for the partner. A connection ID is only issued for the local partner.

  The figure below shows a specified connection:



- Connection unspecified
  Create unspecified connections in the following cases by only selecting the local connection partner graphically or interactively:
  - If certain connection properties need an unspecified connection. Examples: E-mail connection, FTP connection, FDL connection with free layer 2 access.
  - If one of the two partners for the selected connection type is not (yet) available, but you want to reserve a connection resource.



Influence of the networking - response when networking is missing

Networking the connection partners it is necessary to ensure the uniqueness and consistency of the assigned connection parameters. If one or both partners is not networked on the same subnet, the following reactions in the graphic connection configuration must be distinguished.

Note

Dialog-guided connection configuration - no automatic networking

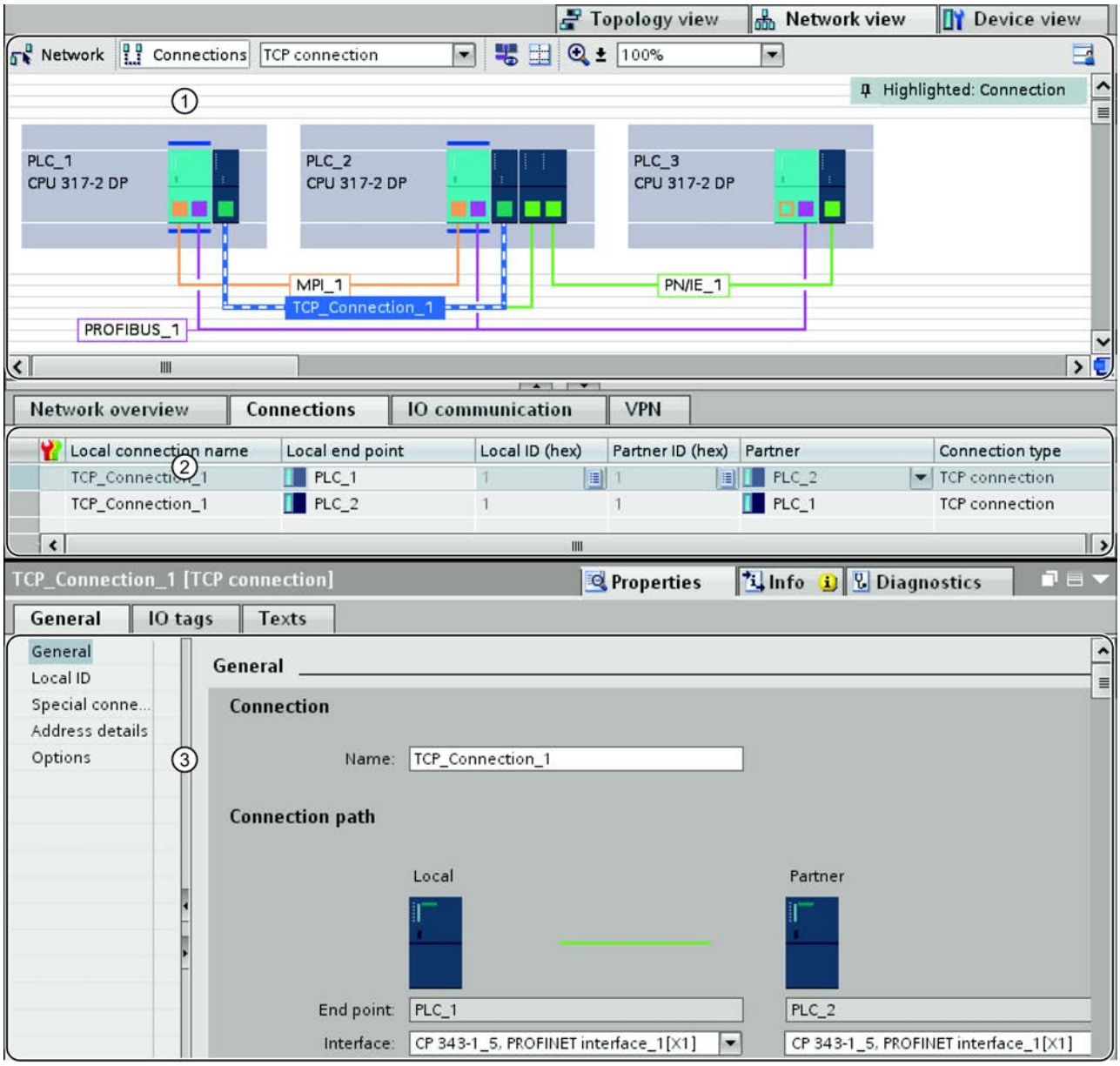
In dialog-guided connection configuration, there is no automatic networking.

- Case a: Interfaces of the partners you want to connect are not networked
  The interfaces of the connection partners are connected via a new subnet. The connection can be created consistently and can be specified.
- Case b: Interfaces of the partners you want to connect are initially not networked on the same subnet
  The "Connect to subnet" dialog that appears provides the following options for selection:
  - Assign existing subnet
    You select a subnet from the list of existing subnets. The connection can be created consistently and can be specified.
  - "Connect to new subnet"
    The interfaces of the connection partners are connected via a new subnet. The connection can be created consistently and can be specified.
  - "Do not connect to a subnet"
    The connection is created as a specified connection, due to the missing networking the consistency cannot be assured which is why the entry appears red in the connection table. Connections are shown as dashed lines in the network view;
- Case c: Multiple interfaces / CPs on the partner - at least one of the available interfaces on the partner is not networked
  The following applies depending on how you create the connection in this case:
  - If you specify the connection only by selecting the CPU / OPC server as the connection endpoint, the connection path is specified regardless of the networking. This can lead to the response described in Case b and the selection dialog is displayed.
  - If you deliberately select the free interface on the partner that is not yet networked as the connection endpoint, the response is as described in Case a.
    The interfaces of the connection partners are connected via a new subnet. The connection can be created consistently and can be specified.

Views containing information about the configured connections

The views described below will provide you with comprehensive access to all the information and functions regarding configuring and checking communication connections.

The view is accessed via the project navigation under "Devices & Networks".



① Connection display in the network view

② Connection table

③ "Properties" tab for a connection in the inspector window

Benefits

The information shown in these views are always up-to-date in terms of the current user actions. This means:

- If you have selected a CPU, for example in the network view, the connection table shows all the connections created for this CPU.
  Note: You can use the connection table shortcut menu to choose between displaying all connections in the project and displaying the connections for a selected component.
- If you have selected a connection in the connection table:
  - The connection path in the network view will be shown in a graphic display when connection mode is activated.
  - The properties view in the inspector window displays the parameters of this connection.

Connection display in the network view

The network view shows devices and their components as well as networks and communications connections.

In relation to connections, in the network view you see:

- The devices between which connections have been created.
- The devices that have unspecified connections.

The network view offers the following functions:

- Creating connections graphically
- Selecting connection paths when creating graphically

The connection table

The connection table offers the following functions:

- Listing of the connections to an object selected in the network view (network, CPU, CP)
- Listing of all connections in the project
- Selecting a connection and the associated display of the connection in the network view and the display of the properties of a selected connection in the Inspector window;
- Changing connection partners
- Option of entering central parameters
- Display of status information;
- Starting point for interactive creation of new connections (possible if the "Display all connections" option is disabled).

"Properties" tab for a connection in the inspector window

The properties dialog has the following meaning:

- Display and option of setting all connection parameters
- Setting the connection path automatically or manually
- Later specification of unspecified connections

Creating a new connection graphically

Graphically configuring connections

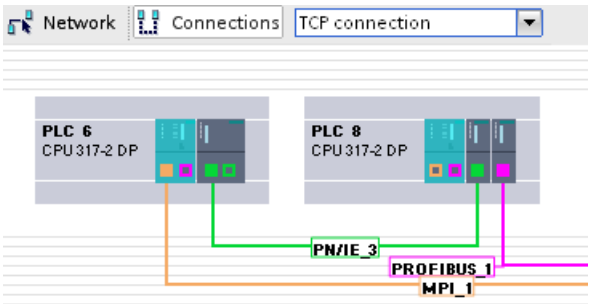
- In the simplest case, you only need to select the devices you want to connect. A possible connection path in the current configuration is selected and marked as "Standard connection path". The connection path is then selected such that the resources available are used as best as possible. The definition is based on the following behavior:
  - Non-networked components are networked when possible automatically to be able to create fully specified connections immediately.
  - The resources of one component are fully used up before switching to another component.
  - With connection types independent of networks (S7 connections), network types are subject to the following rule of preference: Industrial Ethernet before PROFIBUS.
- Alternatively, you can define the specific connection path - connection endpoint and interface to the subnet.

Automatically determining connection path

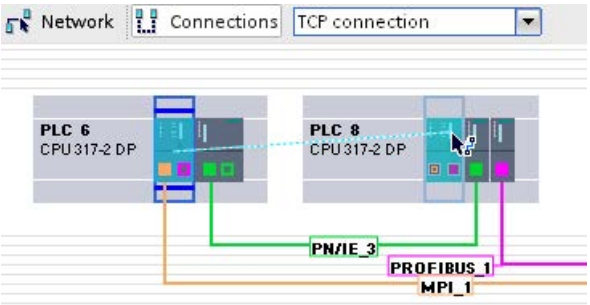
To graphically create a connection via a default connection path, proceed as follows:

1. On the toolbar, click the "Connections" icon. This activates connection mode.

2. Select the required connection type in the drop down list beside the icon.
   You will then see the following response:
   - The components which can be used in your project as connection endpoints in the selected connection type are highlighted in color in the network view. These components may be CPUs or PC applications.



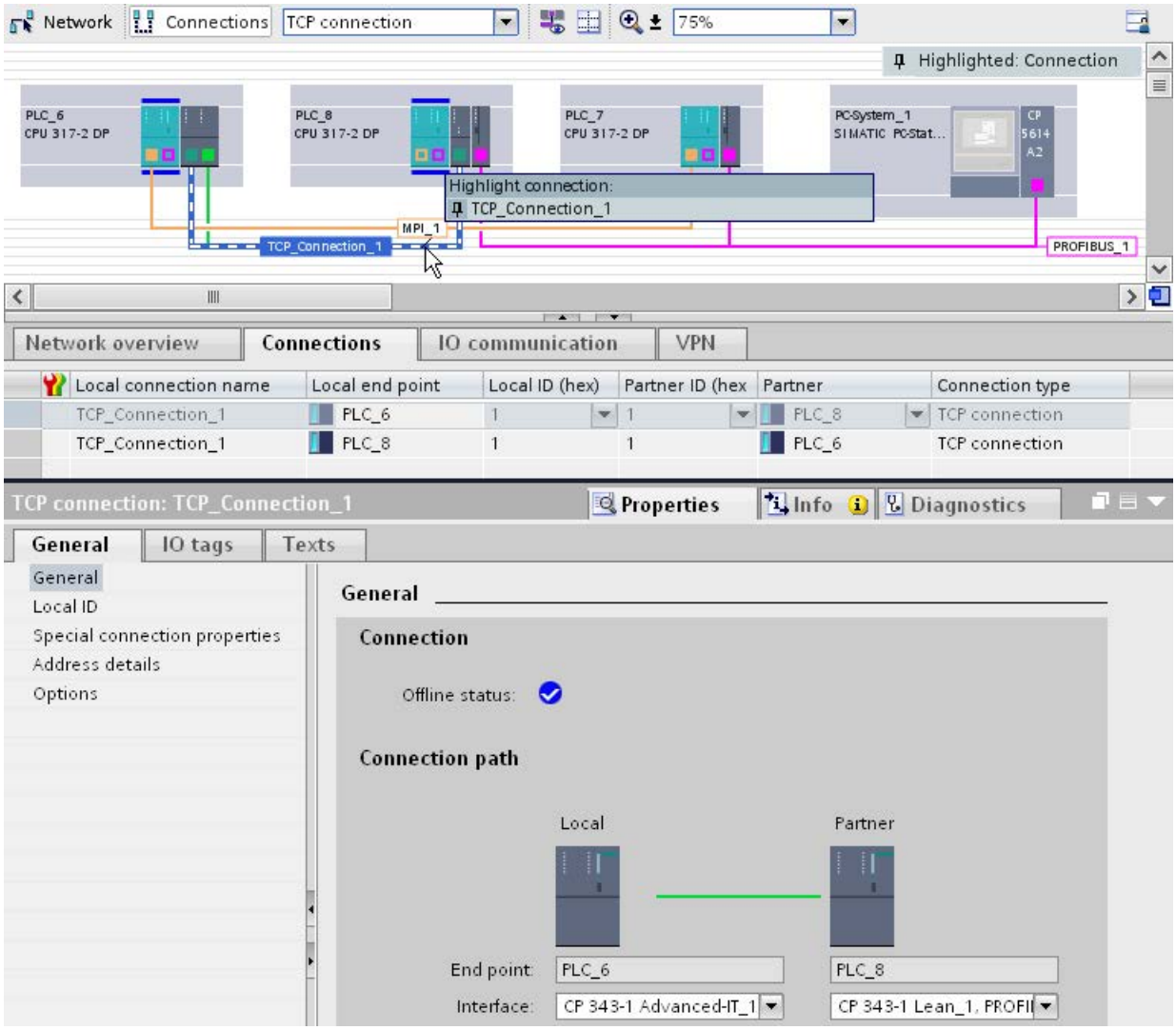
3. Select the component for the origin of the connection.
4. There are two options when creating the connection:
   – Click on the component from which the connection will originate and drag the mouse cursor to the target components. Confirm the connection endpoint with another mouse click.



   – While holding down the shift button, select the target components and with the right mouse button select the "Add new connection" command.

Result

- A specified connection is created.
- The connection path is highlighted. The connection path has been set automatically in line with the available connection resources and interfaces.
- The connection is entered in the connection tables for the components (CPU/PC application).



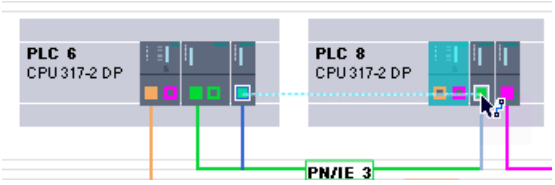
Selecting specific connection path

Proceed as follows to create a connection with an individually selected connection path:

1. On the toolbar, click the "Connections" icon. This activates connection mode.

2. Select the connection type you need from the drop-down list.

3. Click on the subnet interface in the device from which the connection is to be created.
4. Hold down the mouse button, drag the cursor to the relevant interface in the target device and then release the mouse button.



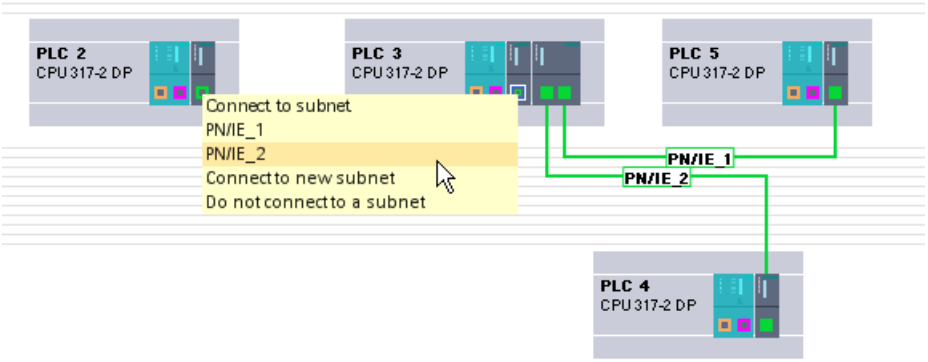
Result:

– A specified connection is created.
– The connection created between the endpoints will use the selected interfaces for its connection path.
– The connection path is highlighted.
– The connection is entered in the connection tables for the devices (CPU/PC application).

Configuring a connection when there is no or no clear network assignment

Any networking not in place will if possible be created automatically when a connection is created. A query will be made upon completion of connection configuration if unique network assignment is not possible. You will be able to choose from the existing subnets.

Example below: A query is made upon creation of a connection between stations PLC2 and PLC3, which are not yet networked.



Interactively creating a new connection

Interactively configuring connections

Interactive connection configuration is a useful way to create multiple connections in one process.

On the one hand, you have complete control over the selection of the connection path as the system displays all interface variants. At the same time, you have the option of leaving evaluation of available resources and thus selection of the connection path to the system.

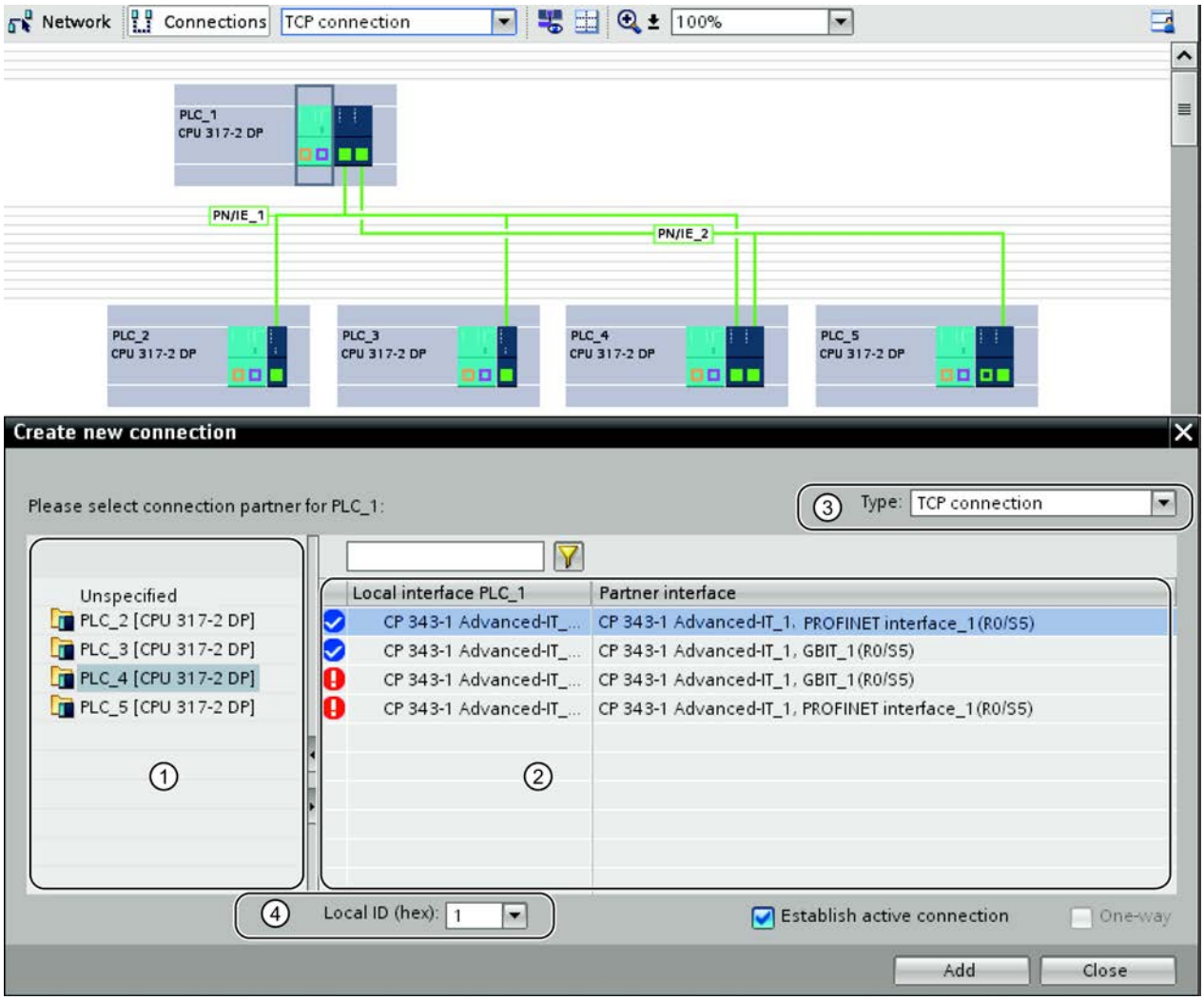
Note

Dialog-guided connection configuration - no automatic networking

In dialog-guided connection configuration, there is no automatic networking.

The "Create new connection" dialog

The structure of the "Create new connection" dialog is shown below. The example shows networking between several stations. Connection mode is active and the first station in the list has been selected.


Network
Connections
TCP connection
100%
PLC_1
CPU 317-2 DP
PN/IE_1
PN/IE_2
PLC_2
CPU 317-2 DP
PLC_3
CPU 317-2 DP
PLC_4
CPU 317-2 DP
PLC_5
CPU 317-2 DP
Create new connection
Please select connection partner for PLC_1:
3
Type:
TCP connection
Unspecified
PLC_2 [CPU 317-2 DP]
PLC_3 [CPU 317-2 DP]
PLC_4 [CPU 317-2 DP]
PLC_5 [CPU 317-2 DP]
1
2
Local interface PLC_1
Partner interface
CP 343-1 Advanced-IT_...
CP 343-1 Advanced-IT_1, PROFINET interface_1 (R0/S5)
CP 343-1 Advanced-IT_1, GBIT_1 (R0/S5)
4
Local ID (hex):
1
Establish active connection
One-way
Add
Close

① Shows the possible connection partners for the connection type selected.
Choose from the following options:
- A special connection configuration in line with the connection type:
  - "unspecified"
  - "Broadcast"
  - "Multicast"
- The connection partner

② Content area for selecting interfaces
You will see interface combinations suitable for the configuration.
Shows only those interfaces possible for the connection type selected.
The status display provides additional information on whether the connection can be set up as a function of networking and the available resources.
You can still create a connection even if the interfaces are not networked.
See also Configuration limits in the connection configuration (Page 577)
Example shown:
Between PLC1 and PLC4, the Gbit ports are networked, as are the PROFINET interfaces. Only these two combinations are therefore shown as completely assignable.

③ Drop down list for selecting the connection type

④ Default connection parameters

Status display

Icon	Meaning
	Interfaces are networked and connection resources are available for the selected partners.
	Interfaces are disconnected or not connected to the same subnet.

Procedure

Proceed as follows to interactively create a connection:

1. On the toolbar, click the "Connections" icon. This activates connection mode.

2. Select the command "Add connection" in the shortcut menu of the component which is to be the connection endpoint (CPU or PC application).
   or
   With the "Display all connections" option disabled and the CPU or application selected, select the "Add connection" command in the shortcut menu of the connection table.
   The "Create new connection" dialog opens.
3. Select the connection type you want, for example "FDL connection" from the "Type" drop-down list.

4. Check the default setting for the other connection parameters displayed in the footer and correct them in line with your requirements.

   Note

   You can also change these parameters later on in the connection table or in the properties for the connection.

5. Select the connection partner (device or CPU) in the left of the dialog.
   The interface combinations possible for the configuration together with status information will be displayed in the content area. The first interface combination to which a connection of the selected type can be assigned will be selected.

   Note

   Depending on the connection type selected, special connection types such as "unspecified", "broadcast" or "multicast" may be available in addition to the possible connection partners.

6. Leave the selection as it is or select the connection path you require.
7. Confirm the selection with the "Add" button.
   A connection is created and can be seen in the connection table.
8. If you have not yet closed the dialog, you have the alternative of continuing as follows:
   - You can create another connection of the same type to the same target device by clicking the "Add" button again.
   - Repeat the steps described above by selecting another connection type from the drop-down list or another connection partner.

Configuring connection without networking

Any networking not in place will if possible be created automatically when a connection is created. The relevant interface will be displayed as networked in the network view once the dialog is completed provided unique network assignment is possible.

Networking will not be displayed in the case of interactive connection configuration if unique network assignment is not possible. Connections created in this way are therefore not fully specified and are highlighted in red in the connection table.

Working in the network view

Meaning of the network view in connection mode

The network view shows devices and their components with interfaces to the subnets. In connection mode, you can make the communication paths of configured connections visible.

The following information about connections is then available:

- The devices between which connections have been created
- Which connection types have been created for a device
- The devices that have unspecified connections

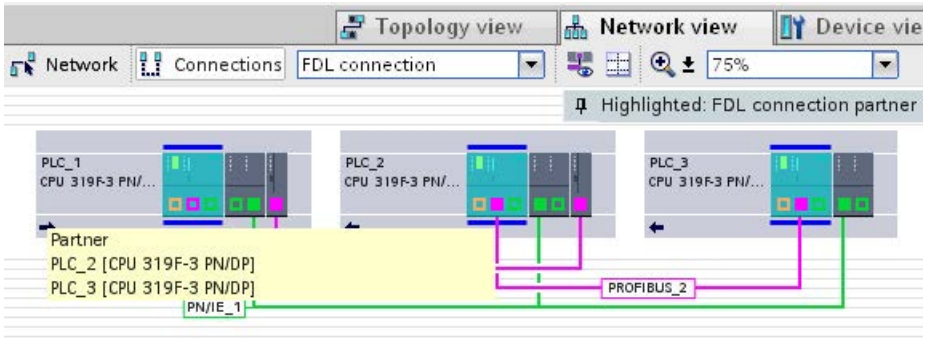
The network view offers the following functions in connetion mode:

- Graphically creating connections; see also Creating a new connection graphically (Page 605)
- Highlighting selected connections of a subnet;
- Highlighting connection partners.

Highlighting connection path and partner in the network view

To display the connection partners for all or certain connection types in the network view, proceed as follows:

1. Select the "Highlight connection partners" command in the shortcut menu of the connection endpoint (CPU or PC application) whose connection partners you want to display in the network view.
2. In the following menu, select either "All connection partners" or "Select connection type..." The local endpoint and the endpoints of the target devices relevant for all or the selected connection type are selected. In addition, the devices display an arrow.
3. You can open a list with information on the target devices by right-clicking on the arrow in the footer of the local device. This additional function is useful in complex network configurations in which some devices are not visible.



Note

You can display one of the connection partners which cannot be seen in the current display range of the network view. Click on the communication partner in the list that appears. Result: The display is moved such that the connection partner becomes visible.

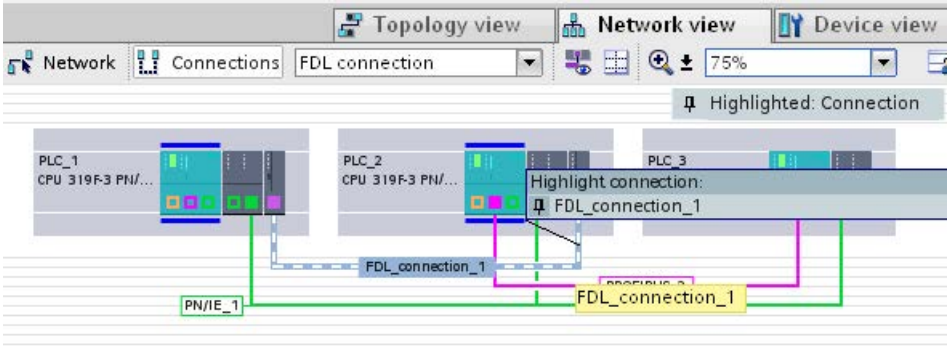
Highlighting/selecting the connection

You can select and highlight individual connections. This means:

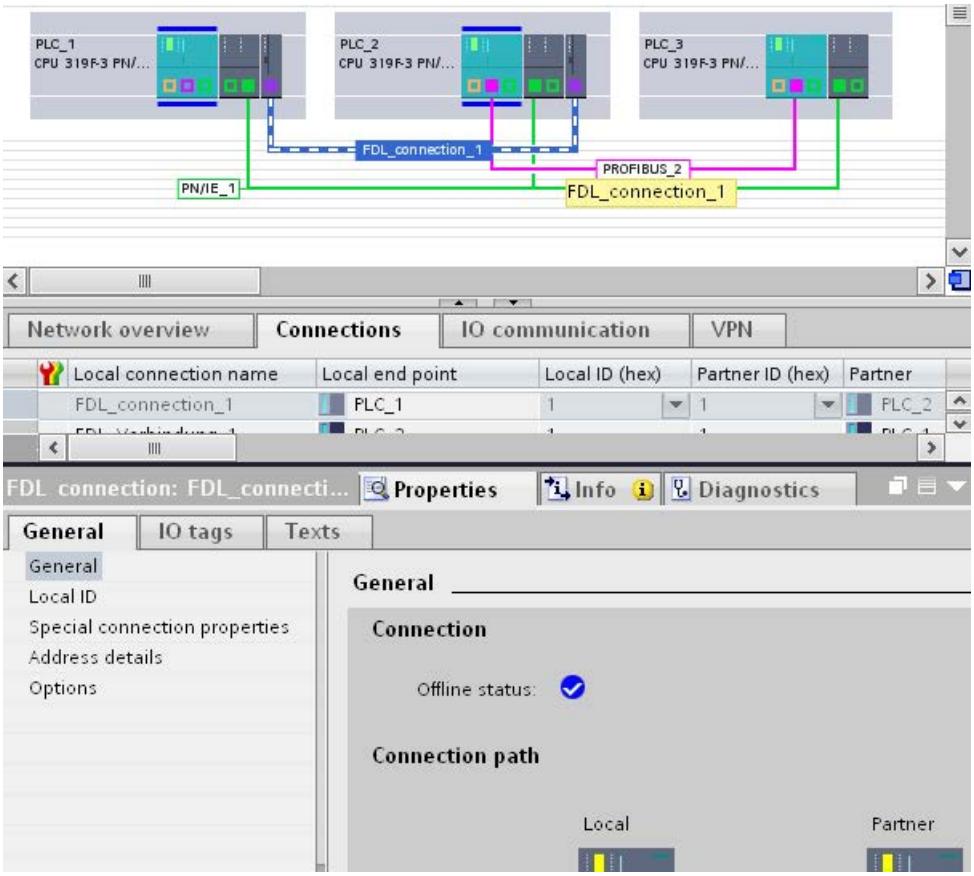
- The connection is highlighted in the connection table where you can set certain properties.
- The connection is displayed in the inspector window together with all parameter assignment options.

Follow these steps:

1. Place the cursor on the subnet.
2. Select and highlight the relevant connection in the "Highlight connection" dialog that appears.
   The connection path will be displayed as a light blue dotted line.



3. Click on the dotted line.
   The color of the line will change to dark blue and the connection will be displayed in the connection table and in the inspector window.



Working with the connection table

Functions

As well as allowing you to create new connections interactively, the connection table provides other functions:

- List of connections to an object selected in the network view (network, CPU, CP)
- List of all connections in the project
- Selection of a connection and display of connections associated with it in the network view
- Changing connection partners
- Option for entering central parameters
- Display showing status information

Creating new connections interactively

Interactively creating a new connection (Page 608)

Content of the connection table

You can use the connection table to display the details for all communication connections created in the project:

- Show all connections (default setting - can be selected via the shortcut menu)
  The connection table displays all connections created for the project.
- Show selected connections (recommended setting for complex projects)
  To access this display, clear the "Show all connections" option in the shortcut menu.
  The connection table shows the connections belonging to a selected subnet or selected CPU and/or CP. In other words, it's empty when none of these named objects are selected in the network view.
  Multiple objects of the same type (multiple subnets, multiple CPUs/CPs or multiple interfaces) can also be selected at the same time. The connection table then shows you all associated connections.

Basic functions for tables

The connection table supports the following basic functions for editing a table:

- Showing and hiding specific table columns
  Select the "Show/hide columns" command in the table header in the shortcut menu.
- Changing column width manually
  Move the dividing line in the header on the right of the column.

- Optimizing column width automatically
  Select the shortcut menu command "Optimize column width" in the table header.
- Showing column meaning using tooltips.

If you want to store the layout, width and visibility of the table columns, click on the "Remember layout" function in the top right-hand of the network view.

Icon	Meaning
	Remember layout Saves the current table view. The layout, width and visibility of columns in the table view is stored.

Using cursor keys to move within the connection table

You can use the UP and DOWN cursor keys to select a connection from the connection table; the selected connection will be marked and highlighted in the network view when connection mode is activated.

Changing properties of connection

You can directly edit some of the parameters displayed in the connection table. The name of the connection can, for example, only be changed in the connection table.

Changing connection partners

You can change the connection partner of a connection as follows:

1. Select the connection.
2. Select the new connection partner from the open drop-down list in the "Partner" column.

Changing the connection partner

Follow the procedure below if you need to change the connection partner for a connection already configured. The local ID and the connection type will be maintained.

Changes can only be made to connection endpoints in the connection table or in the inspector window.

Changing connections interactively

To reach a different connection partner on a default connection path proceed as follows:

1. In the network view, select the local endpoint to which you wish to assign a new connection partner.
   The endpoint connections will be displayed in the connection table.
2. Select the desired connection in the connection table.
3. Select the new target device for the connection from the drop-down list in the "Partner" table column.
   You can alternatively select the new target device under "Connection path" in the inspector window.

Note

Effect on previously configured properties

Note that when changing the connection partner the parameter assigned properties of the connection must be reset to the pre-select.

Use the entry possibilities in the connection table or in the Inspector window if you want to change the "Properties" of the connection.

Note

Effect on networking

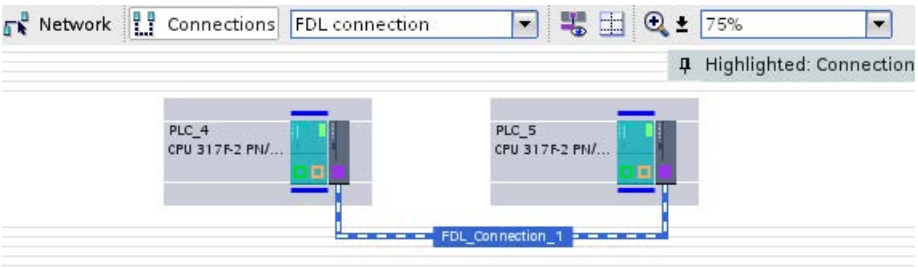
Subnets will not automatically be created in the absence of networking to the new connection endpoint in procedures for changing the connection partner.

Setting or changing the connection path

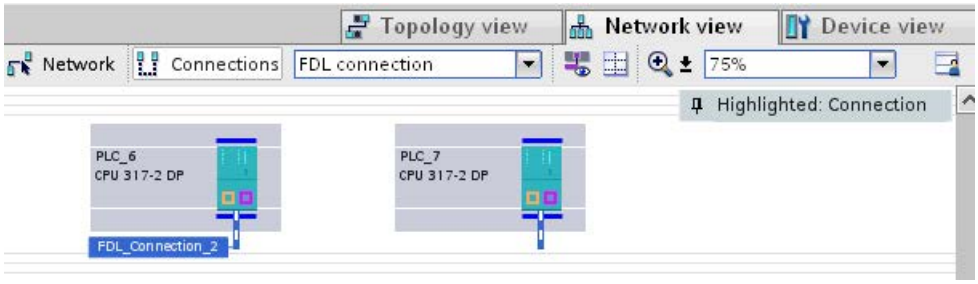
Depending on the components and interfaces available within a device, connections can take different paths within the device.

Connections in the configuration will have one of the following configuration statuses in line with the status of the device configuration and will be displayed accordingly:

- Connection path closed
  Both endpoints (CPU or PC application) and the interfaces required for communication (local interface in the CPU or additional CPU) are available.



- Connection path open
  - The connection path is not fully assignable as interfaces are missing.
  - The connection path is only specified at one endpoint because it is an unspecified connection.



The procedure for changing or reassigning a configured connection path in a local or partner device is described below.

You can set or change the connection path in the following views:

- Inspector window - "Properties > General"

Changing the connection path in the Inspector window

To assign a different connection path to an existing connection:

1. In the network view, select the endpoint (CPU or PC application) for which a connection is to be changed.
2. In the connection table select the connection for which you want to change the connection path.
   Hint: The current connection path is highlighted in the network view if you have set the "Highlight connection" view.
3. Select the new interface in the Inspector window under " Properties > General > Interface". The partner is offered the interfaces that can be considered in the context of the local interface (Note: S7 connections can be established via different subnet types).
   In the network view the new connection path is highlighted.

Assigning connection path

To assign a closed connection path to an existing open connection path:

1. Expand the device configuration to that the interfaces required for the connection type are available for both partners.
2. In the connection table, select the connection for which you want to change the connection path.
   Hint: The current connection path will be highlighted in the network view if you have set the "Highlight connection" view accordingly.
3. Activate the "Find connection path" button under "Properties > General > Interface" in the inspector window.
   In the network view the new connection path is highlighted.

Creating an unspecified connection

Create unspecified connections in the following cases by only selecting the local connection partner graphically or interactively:

- When certain service or connection properties require an unspecified connection (examples: E-mail connection, FTP connection, FDL connection with the "Free layer 2 access" property).
- If you want to set up a connection to be available for communication with any connection partner.
- If one of the two partners for the selected connection type is not (yet) available, but you want to reserve a connection resource.
- If the connection partner is located outside of the project (e.g. third-party stations).
- UDP connection with broadcast / multicast:
  - See UDP with broadcast and multicast (Page 650)
  - See Free UDP connection (Page 653)

Requirement

You must be in the network view.

Configuring an unspecified connection graphically

To create an unspecified connection graphically, proceed as follows:

1. In connection mode, select the required connection type in the drop-down list box, for example "TCP connection".

   The CPUs available for the selected connection type are selected and highlighted.
2. Click on the CPU from which a connection will be set up.
3. Move the mouse cursor within the CPU or on the interface of the CPU and confirm the connection endpoint by double-clicking the mouse.

Result

- An unspecified connection is created. Depending on the selected endpoint - CPU or CP - the following applies for path selection:
  - endpoint CPU: The best possible local connection path will be selected automatically. See also Creating a new connection graphically (Page 605)
  - endpoint CP interface: The connection path is determined by the entry.
- The unspecified connection is highlighted in the display by a short dotted line.

- The unspecified connection is entered in the connection table of the device and "Unknown" is entered in the "Partner" column.

Configuring the unspecified connection interactively

To interactively create an unspecified connection, proceed as follows:

1. Select the "Add connection" menu command in the context menu of the device (CPU) for which you want to create a connection.
   or
   With the "Display all connections" option disabled, select the "Add connection" command in the shortcut menu of the connection table.
   Result: The "Create new connection" dialog will open.
2. In the "Create new connection" dialog box select the desired connection type, e.g. "FDL connection".
3. In the top of the dialog box select the entry "Unspecified".
4. If several local interfaces are available, you can select the required interface from a list. The first row of the list displays the interface the system detected as the most suitable and this is selected.
5. Confirm the selection with the "Add" button.
   An unspecified connection will be created and will be visible in the connection table.
6. Repeat the process or exit the dialog via "OK".

Result

- One or more unspecified connections are created.
- The unspecified connections are entered in the connection table.

Note

Note that you can still assign parameters for special properties for each connection.

Creating UDP / FDL connections with broadcast / multicast

Meaning

Multicast and broadcast are special connection options that are supported by Industrial Ethernet CPs on UDP connections and by PROFIBUS on FDL connections and can be configured.

Creating a connection interactively - procedure

To create a multicast / broadcast connection interactively, follow these steps:

1. Select the "Add connection" command in the shortcut menu of the component that will be the connection endpoint.
   or
   With the "Display all connections" option disabled, select the "Add connection" command in the shortcut menu of the connection table.
   The "Create new connection" dialog opens.
2. Select the connection type UDP connection / FDL connection" from the "Type" drop-down list.
3. Select the connection type broadcast or multicast in the left-hand part of the dialog.
   The local interfaces possible for the configuration together with status information are displayed in the content area. The first interface to which a connection of the selected type can be assigned is selected.
4. Leave the selection as it is or select the connection path you require.
5. Confirm the selection with the "Add" button.
   A connection is created and can be seen in the connection table.
6. If you have not yet completed the dialog, you have the alternative of continuing as follows:
   - You can create another connection of the same type to the same target device by clicking the "Add" button again.
   - Repeat the steps described above by selecting another connection type from the drop-down list or another connection partner.

Creating a connection graphically - procedure

1. First, create an unspecified UDP connection / FDL connection using the graphic connection configuration tool; Creating a new connection graphically (Page 605)
2. Now make the address settings for the connection partner according to the requirements for multicast or broadcast. As an alternative, follow these steps:
   - In to the connection table, go to the "Partner" column and select the required connection type from the drop-down list; address parameters are assigned automatically.
     or
   - For UDP, enter the IP address or port directly in "General > Address details". For FDL, set the PROFIBUS station address 63.

See also

UDP with broadcast and multicast (Page 650)

FDL connection with broadcast (Page 662)

FDL connection with multicast (Page 664)

Specifying an unspecified connection

Characteristics of the unspecified connection

For an unspecified connection depending on the connection type either some or all of the parameters that are necessary for local access to the network and for addressing the partner, will be missing.

An unspecified connection is shown as follows:

- Network view:
  The unspecified connection will be displayed in the network view as a dotted line once it has been created or selected in the connection table.
- Connection table:
  - Display in the "Partner" column: "Unknown"

An unspecified connection occurs as described below: See also Creating an unspecified connection (Page 620)

Below are the main two different scenarios when specifying the connection later:

- The connection partner is in the same project
  You specify the connection by the subsequent selection of components available in the project.
- The connection partner is not in the project
  You specify the connection by entering the address parameters of the communication partner available outside the project.

Automatic supplementing of missing connection parameters

If the missing resources have been supplemented by adding components then connections will be assigned automatically if possible, and if necessary to an unspecified connection.

Requirement

- The components of both devices have free resources for the connection type.

Procedure in the connection table

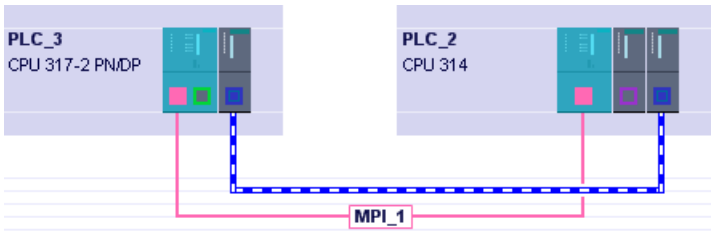
Proceed as follows to provide an unspecified connection with the address parameters of a target device already in the project:

1. Select the CPU for which you want to specify a connection.
   In the connection table, all unspecified connections configured for the CPU have the entry "Unknown" in the "Partner" column.
2. Select the required target device in the "Partner" column of the connection table.

Result

- The connection is specified with the relevant address details for the connection partner.
- The new connection path is highlighted in the network view as a dotted line.

The figure below shows a specified connection:



Deleting a connection

You can delete configured connections via the network view or the connection table.

In the network view you can delete a highlighted connection. In the connection table you can delete one or more connections.

Procedure in the network view

You can delete individual connections in the network view via the shortcut menu. Follow these steps:

1. Switch to connection mode.
2. Place the cursor on the subnet.
   The "Highlight connection" dialog will show the connections currently configured.
3. Select the connection to be deleted.
4. Place the cursor on the subnet again.

5. Open the shortcut menu with a right mouse click.
6. Select the "Delete" command.
   The selected connection will be deleted in full. If the partner for a specified connection is within the project, the connection will also be deleted from the connection table of the partner.

Note

Downloading connection data to a module

Download the connection table with the deleted connection to the programmable modules affected by it.

Procedure in the connection table / network view

You can delete one or more selected connections in the connection table via the shortcut menu. Follow these steps:

1. In the connection table, select the connections that you want to delete.
2. Open the shortcut menu with a right mouse click.
3. Select the "Delete" command.
   Each connection selected will be deleted in full. If the partner for a specified connection is within the project, the connection will also be deleted from the connection table of the partner.
4. Load the connection table with the deleted connections into the appropriate programmable module.

Note

Downloading connection data to a module

Download the connection table with the deleted connection to the modules affected by it.

Note

To delete all connections of the module you must load an empty connection table.

Copying a connection

Introduction

Connections are not copied singly but always in context along with the project or the device.

You can copy:

- Entire projects
- One or more devices within a project or from one project to another

Copying a project

When you copy a project all configured connections will also be copied. No new settings are required for the copied connections as the connections remain consistent.

Copying devices

- Copying a device within a project
  Copying individual devices produces unspecified connections and you will need to reassign the connection partners to the local node.
  If you copy a group of devices, the connections within the group will also be copied. Duplicating will however also produce duplicate connection names. Change the duplicate connection names after copying.
  You will need to network the copied devices again.
- Copying devices to another project
  You can copy one or more devices to another project. To continue editing the other project, restart the TIA Portal.
  Connections configured between the copied devices remain in the target project. You need only reassign networking in the target project.
  Copying the devices to the target project individually will produce temporarily unspecified connections. However, the communication partners will be assigned again if this is possible when the next device is added.

Additional information

See also Specifying an unspecified connection (Page 623)

Inconsistent connections - connections without assignment

An inconsistent connection is one in which the connection data is incomplete or inconsistent; the connection is not functional in the context with the project.

Inconsistent connections cannot be loaded - operation is not possible with such connections.

In the connection table inconsistent connections are marked in red.

Note

Unspecified connections

Unspecified connections are not necessarily inconsistent although they are always highlighted in red in the connection table.

Creating an unspecified connection (Page 620)

Possible causes for inconsistent connections

- Deletion or change of the hardware configuration.
- Missing interface network links in the project, which are necessary for a connection.

- Connection resources are exceeded
- Connections to an unspecified connection partner without partner address information.

Detailed information about the cause of the inconsistency is provided in the inspector window in the properties page of a connection; inconsistent parameters are shown in red. You can get supplemental information about these parameters by placing the cursor on the parameter.

Examples of and remedies for inconsistencies associated with typical configuration actions

Actions are explained below that can cause configured connections to lose their assignment or be deleted.

Note

Note that as opposed to the S7 connections, a CP-dependent ID is assigned to the connections of the SEND/RECEIVE interface. Consequently the actions described below may require customizing of the ID, which means that the interface must also be customized in the user program.

If a CP is replaced by a different CP then it must provide at least the same services and have at least the same version. This is the only way to ensure that the connections configured via the CP remain consistent and can be used.

Table 8-3 Actions for a CP interface that can cause changes in configured connections.

Action	Consequence for the connections	What you must do to restore the connection
Delete the CP (module).	The connections remain intact in the connection table without assignment to a CP. The assignment to the CPU / PC application remains.	After you have placed and networked a CP in the hardware configuration: 1. Assign the CP to the connection; 2. Check the module start address LADDR and customize if necessary in the user program. 3. Customize the connection IDs in the user program. 4. Reload the connection configuration into the CP.
Delete the device.	All the connections to this device are canceled in the project. • Connections remain as unspecified connections on the partner. • Connections without a connection partner are completely deleted.	Reconfigure the device and connections. Either reassign unspecified connections or delete them.
Change the subnet assignment of the CP.	The connections that were assigned via the CP remain but may be inconsistent.	Where necessary, reassign connections.

Properties of other connection types

S7 connection

S7 connection - general settings

General connection parameters

General connection parameters are displayed in the "General" parameter group under the properties of the connection; these connection parameters identify the local connection endpoint.

Here, you can assign the connection path and specify all aspects of the connection partner.

Local ID

The local ID of the module from which the connection is viewed is displayed here (local partner). You can change the local ID. You may need to do this if you have already programmed communication function blocks, and you want to use the local ID specified in those function blocks for the connection.

Special connection properties

Display of connection properties (can be modified depending on the components used):

- Configured at one end
  Configured at one end means that the connection partner functions as a server on this connection and cannot send or receive actively.
  Communications instructions used at both ends (with "pairs of blocks") cannot be used on connections configured at one end; for example USEND (Page 7371) and URCV (Page 7373), BSEND (Page 7376) and BRCV (Page 7379) or USTATUS (Page 7408).
- Active connection establishment
  Use this option to specify whether the connection is established from this device.
  If you selected "Unspecified" as the connection partner when you created the connection, the option is deselected by default. If you select this option, you must specify the address of the partner in the "Addresses" tab.
- Send operating mode messages
  Indicates whether or not the local partner sends operating mode messages to the connection partner.

Address details

Displaying address details of the S7 connection. With an unspecified partner, the values for the rack and slot can be changed. All other values are obtained from the current configuration and cannot be changed.

S7 connection - address details

Meaning

A connection resource such as shown under "Properties > Address details" is part of the TSAP (Transport Service Access Point) of the local module or partner.

The ability to use connection resources is subject to rules and restrictions. For example, not every connection resource can be used for every connection type.

Ranges of values

The following table provides information on the meaning of the values and the type of connection. Depending on the connection partner and connection type, the range of values is automatically limited to valid values or the value of the connection resource is assigned permanently.

Connection resource (1st byte of TSAP)	Meaning	Type	Meaning
0x01 ("PG")	Programming device connection	Free connection (not configured)	At least one resource per CPU is reserved for programming device connections. However, for certain S7-300 CPUs it is possible to reserve multiple resources in the CPU properties.
0x02 ("OP")	OP connection	Free connection (not configured)	At least one resource per CPU is reserved for OP connections. However, for certain S7-300 CPUs it is possible to reserve multiple resources in the CPU properties.
0x03	Other	Free connection (configured, unspecified connection)	This connection resource can operate multiple connections. Use: Connection configured at one end with unspecified connection partner! The connection partner does not have to be configured if the connection resource 0x03 is addressed.
0x10..0xDF	Connections with static or dynamic connection setup	Configured unspecified connection	One of these connection resources can operate one (and only one) connection. Use: Connection configured at both ends with unspecified connection partner!

Free S7 connections

Free S7 connections are set up dynamically during runtime; the following connections are free:

- Programming device connections (0x01)
  S7 connections that are typically set up from a programming device or from a PC (with ES functionality). This type of connection is used to configure and program the addressed station/module as well as to test and commission it; afterwards, the connection is typically cleared again. This connection resource allows both read and write access (e.g., monitoring and loading).
- OP connections (0x02)
  S7 connections that are typically set up from an OP or from a PC (with OS functionality). This type of connection is used to monitor the addressed station/module with regard to the process that is being controlled.
- Other (0x03)
  Use is not specified. For example, this resource is used automatically when an S7 connection configured at both ends is configured from an S7-400 to an S7-300.
- S7 basic communication (0xFD)
  Connections that are typically set up from a CPU to another module (CPU, FM, etc.) within a subnet. The connection setup is initiated by the application program, in which a connection configuration does not exist. This type of connection allows process data to be exchanged between the modules. For certain S7-300 CPUs, it is possible to reserve resources for S7 basic communication.

Recommendation for selection of the connection resource

STEP 7 always suggests a free connection resource for the local partner. We recommend that you accept it.

ISO transport connection

ISO transport – special connection properties

Relevance

You can specify additional properties of the local connection endpoint in the "Properties > General > Special connection properties" parameter group.

Establishing an active connection

Use this option to specify whether the connection is established from this device. This is the default option if the address of the partner is specified.

- ON: Connection is established actively.
- OFF: Connection is established by the partner.

If you selected "Unspecified" as the connection partner when you created the connection, the option is deselected by default. If you select the option you must specify the address of the partner in the "Addresses" tab.

Note

Effect on the mode

Note the effect on the mode.

- S7-300 / S7-400
  If you want to use FETCH or WRITE mode (see the "Options" parameter group), disable the "Active connection establishment" option.
- OPC server
  If you want to use FETCH or WRITE mode (see the "Options" parameter group), enable the "Active connection establishment" option.

Name

A connection endpoint name is proposed here when you create the connection. This name contains a connection number as suffix.

Use this field to identify the partner on unspecified connections.

ISO OPC establishment details

Relevance

Connection configuration for OPC in the parameter group "Properties > General > Establishment details".

Options for connection establishment

Entries in this area are possible only if you have selected active connection establishment in "Special connection properties".

Choose from the two following alternatives:

- Keeping the connection established permanently
  The connection is established permanently when you start up the PC station and is re-established following an interruption on the connection.
- Establishing the connection on request (access to tag)
  You can also specify how long the connection should stay up after access.

Monitoring connection establishment

- Timeout: Error wait time during the connection establishment
  If, during this wait time, the connection cannot be established, the connection is reset and a pending request acknowledged with an error.
  Range of values: 0 to 99999 ms
  Default: 15000 ms

Note

The connection establishment monitoring that can be set here, has the following relationship with the time monitoring that can be set in the module configuration: The lower value is valid.

Time monitoring can be configured in the module configuration in "Properties Ethernet interface".

ISO transport address details

Relevance

Connection configuration in the parameter group "Properties > General > Address details".

Address parameter

An ISO transport connection is specified by the local and remote connection endpoint.

- Local addresses:
  Local MAC address and local TSAP (Transport Service Access Point)
- Remote addresses:
  Remote MAC address and remote TSAP

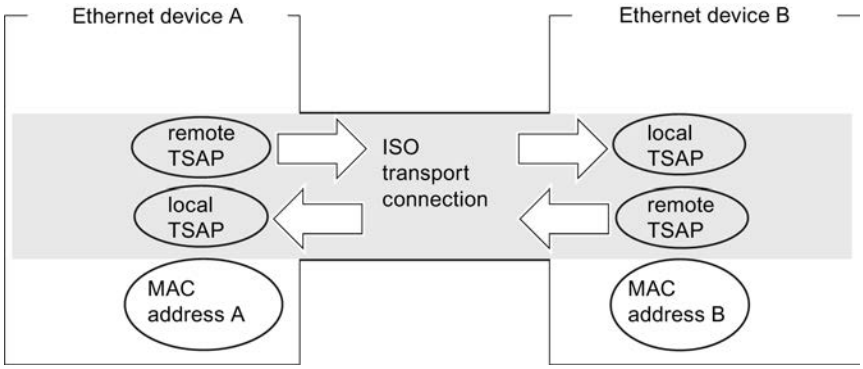


Figure 8-1 Sending and receiving via one ISO transport connection each

TSAP format

ISO transport connections have a TSAP length of 1 to 16 bytes. For the entry the actual length is displayed automatically (visible display: 16 ASCII characters). Local and remote TSAPs can be entered as a hexadecimal value or as an ASCII string. For ASCII input, the entered

characters are also displayed in hexadecimal. With hexadecimal input, printable characters are shown as an ASCII value (8 hexadecimal characters are visible). If non-printable characters are entered, the ASCII display is greyed out (ASCII input is no longer possible) and the non-printable characters are shown as periods.

Local and remote TSAPs

Remote and local TSAPs can be identical because the connection through the different MAC addresses is unique. If more than one connection is to be set up between two stations, the TSAPs must differ.

- Specified connection
  The TSAPs of an ISO transport connection must agree as follows:
  - Remote TSAP (on the Ethernet CP) = local TSAP (on the target station)
  - Local TSAP (on the Ethernet CP) = remote TSAP (on the target station)
- Unspecified connection

Some address parameters are not specified: Unspecified connections (Page 596)

Default TSAPs

Default values (changeable) are recommended for configuring local and remote TSAPs (for example ISO-1 for the first connection between two partners). If new connections are configured between the same partners, the default values will be incremented automatically (for example ISO-2, etc.). For a new connection to a new partner, the default value will start with ISO-1 again.

ISO transport options

Relevance

Connection configuration in the parameter group "Properties > General > Options".

Operating modes

- SEND/RECV:
  This is the standard mode with the AG_SEND/AG_LSEND, and AG_RECV/AG_LRECV FCs.
- FETCH/WRITE:
  System memory areas in the SIMATIC S7 can be accessed from the following devices if you select one of the operating modes below for the ISO transport connection:
  - SIMATIC S5
  - SIMATIC PC stations
  - Third-party devices

  The connection can then only be used for this mode. It will not be possible to send or receive via the AG_SEND/AG_LSEND or AG_RECV/AG_LRECV instructions.
  - FETCH
    Selecting the operating mode Fetch enables direct read access to data areas in the SIMATIC S7. Access can, for example, be from a SIMATIC S5 station or a third-party station.
  - WRITE
    Selecting the operating mode Write enables direct write access to data areas in the SIMATIC S7. Access can, for example, be from a SIMATIC S5 station or a third-party station.
- SPEED SEND/RECV:
  Selecting the SPEED SEND/RECV mode allows you to use the AG_SSEND and AG_SRECV instructions.
  AG_SSEND/AG_SRECV (SPEED SEND/RECEIVE) provide faster and therefore optimized transmission within a S7-400 station. Check the device documentation of the CP you are using to see whether this operating mode is supported by the CP and in conjunction with the selected CPU.

FETCH/WRITE requirements: Mode of connection establishment

Please select one of the following options in the parameter group "Properties > General > Special connection properties" in line with the station type for connection establishment.

- SIMATIC S7 station: PASSIVE connection establishment
- PC station: Active connection establishment

"S7 addressing mode" option (for FETCH/WRITE only)

You can select the addressing mode when you configure the FETCH ACTIVE/WRITE ACTIVE mode.

- S7 addressing mode: Byte address
- S5 addressing mode: Word address

This allows applications to access S5 or S7 stations without modifying the addresses. This is particularly useful for existing S5 applications, for example, that can now be used unchanged when accessing S7 stations.

The addressing mode for accessing SIMATIC S7 (option selected) is set by default.

See also

Connection configuration for FETCH/WRITE (Page 657)

Use of FETCH/WRITE (Page 655)

ISO transport - dynamics

Relevance

Connection configuration in the parameter group "Properties > General > Dynamics".

The relevant timers and counters for this connection are displayed. You can accept these default values.

Where necessary (for example when linking to third-party systems), you can also set the timers and counters individually to achieve the required dynamic response of the connection.

ISO OPC properties

Relevance

Connection configuration for OPC in the parameter group "Properties > General > OPC".

Here, you set connection-specific properties for an ISO transport connection used by the OPC server.

You can set the following parameters:

Timeout for job

If a pending job cannot be processed during the wait time set here, the connection is reset and the job acknowledged with an error.

Range of values: 0 to 99999 ms

- Default setting: 15000 ms

Send buffer size

Default size of the default send buffer. Additional send buffers of any length (up to 65535 bytes) may be used at runtime.

Range of values:

- Default setting: 256 bytes

- Range of values: 65535 bytes maximum

"Optimization of direct access for tag services" option

If you use this option, the OPC server attempts to combine direct accesses to several tags to minimize the number of jobs that need to be sent.

This option is enabled by default.

"Immediate response when interrupted connection is detected" option

If you select this option, you specify that the application is notified that a communication job (FETCH, or WRITE) cannot be executed when a connection is unavailable. The code is sent regardless of whether or not a current attempt to establish the connection succeeds.

If you do not select this option, the application is informed of the nonavailability of a connection only when the connection times out.

The option described here is relevant and available only when the two modes listed below are used:

- The connection is configured as a permanent connection.
  Connections can be configured as permanently established connections (in "Details of connection establishment"). This means that jobs can be forwarded to the partner with the shortest possible reaction time without needing to establish the connection first.
  If there is a disruption on the connection, the system automatically attempts to re-establish the connection. In this situation, there may be delays before a message can be forwarded to the application.
- You are using the FETCH or WRITE job types (can be selected in "Options").

Only with these job types is an immediate reaction necessary in certain situations if there is no connection.

ISO-on-TCP connection

ISO-on-TCP - special connection properties

Relevance

You can specify additional properties of the local connection endpoint in the "Properties > General > Special connection properties" parameter group.

Active connection establishment

Use this option to specify whether the connection is established from this device. This is the default option if the address of the partner is specified.

- ON: Connection is established actively.
- OFF: Connection is established by the partner.

If you selected "Unspecified" as the connection partner when you created the connection, the option is deselected by default. If you select the option you must specify the address of the partner in the "Addresses" tab.

Note

Effect on the mode

Note the effect on the mode.

- S7-300 / S7-400
  If you want to use FETCH or WRITE mode (see the "Options" parameter group), disable the "Active connection establishment" option.
- OPC server
  If you want to use FETCH or WRITE mode (see the "Options" parameter group), enable the "Active connection establishment" option.

Name

A connection endpoint name is proposed here when you create the connection. This name contains a connection number as suffix.

Use this field to identify the partner on unspecified connections.

ISO-on-TCP OPC establishment details

Relevance

Connection configuration for OPC in the parameter group "Properties > General > Establishment details".

Options for connection establishment

Entries in this area are possible only if you have selected active connection establishment in "Special connection properties".

Choose from the two following alternatives:

- The connection is established permanently
  The connection is established permanently when you start up the PC station and is re-established following an interruption on the connection.
- The connection is established when it is required (access to tag)
  You can also specify how long the connection should stay up after access.

Connection establishment monitoring

Timeout: Error wait time during the connection establishment

If, during this wait time, the connection cannot be established, the connection is reset and a pending request acknowledged with an error.

Range of values: 0 to 99999 ms

Default: 15000 ms

Note

The connection establishment monitoring that can be set here, has the following relationship with the time monitoring that can be set in the module configuration: The lower value is valid.

Time monitoring can be configured in the module configuration in "Properties Ethernet interface".

ISO-on-TCP address details

Relevance

Connection configuration in the parameter group "Properties > General > Address details".

Address parameter

A connection is specified through the local and remote connection endpoint.

- Local addresses:
  Local IP address and local TSAP (Transport Service Access Point)
- Remote addresses:
  Remote IP address and remote TSAP

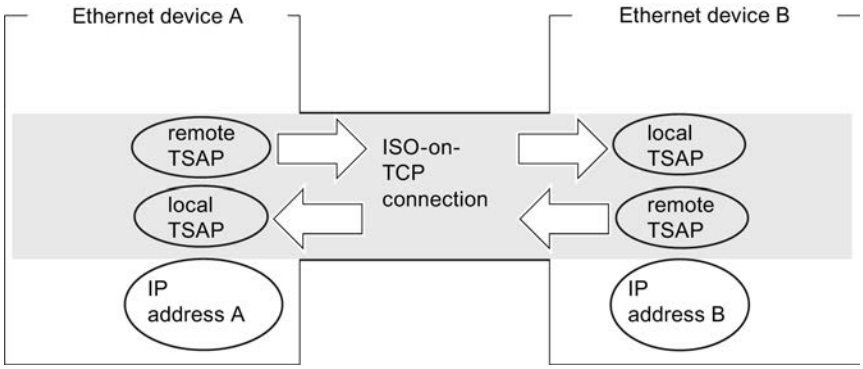


Figure 8-2 Sending and receiving via an ISO-on-TCP transport connection

Addresses tab

In the Addresses tab the relevant local and remote address information is displayed as recommended values. You have the possibility of setting the TSAPs individually.

TSAP format

ISO-on-TCP connections have a TSAP length of 1 to 16 bytes. For the entry the actual length is displayed automatically (visible display: 16 ASCII characters). Local and remote TSAPs can

be entered as a hexadecimal value or as an ASCII string. For ASCII input, the entered characters are also displayed in hexadecimal. With hexadecimal input, printable characters are shown as an ASCII value (8 hexadecimal characters are visible). If non-printable characters are entered, the ASCII display is greyed out (ASCII input is no longer possible) and the non-printable characters are shown as periods.

Local and remote TSAPs

Remote and local TSAPs can be identical because the connection through the different IP addresses is unique. If more than one connection will be set up between two stations then the TSAPs must also differ.

- Specified connection
  In the configuration of the Ethernet CP and the Ethernet target Station, the TSAPs of an ISO-on-TCP connection must be crossed over.
  - Remote TSAP (on the Ethernet CP) = local TSAP (on the target station)
  - Local TSAP (on the Ethernet CP) = remote TSAP (on the target station)
- Unspecified connection
  Some address parameters are not specified: Unspecified connections (Page 596)

Default TSAPs

For configuration of local and remote TSAPs there is a default value "TCP-1" for the first connection between both partners (changeable). For a new connection between the same partners the standard value "TCP-2" is suggested. For a new connection to a new partner TCP-1 is started again.

ISO-on-TCP general

Relevance

Connection configuration in the parameter group "Properties > General > Options".

Operating modes

- SEND/RECV:
  This is the standard mode with the AG_SEND/AG_LSEND, and AG_RECV/AG_LRECV FCs.
- FETCH/WRITE:
  System memory areas in the SIMATIC S7 can be accessed from the following devices if you select one of the operating modes below for the ISO transport connection:
  - SIMATIC S5
  - SIMATIC PC stations
  - Third-party devices

  The connection can then only be used for this mode. It will not be possible to send or receive via the AG_SEND/AG_LSEND or AG_RECV/AG_LRECV instructions.
  - FETCH
    Selecting the operating mode Fetch enables direct read access to data areas in the SIMATIC S7. Access can for example be from a SIMATIC S5 station or a third-party station.
  - WRITE
    Selecting the operating mode Write enables direct write access to data areas in the SIMATIC S7. Access can for example be from a SIMATIC S5 station or a third-party station.
- SPEED SEND/RECV:
  Selecting the operating mode SPEED SEND/RECV allows you to use the instructions AG_SSEND and AG_SRECV.
  AG_SSEND/AG_SRECV (SPEED SEND/RECEIVE) provide faster and therefore optimized transmission within a S7-400 station. Check the device documentation to see whether this operating mode is supported by the CP used.

FETCH/WRITE requirements: Mode of connection establishment

Please select one of the following options in the parameter group "Properties > General > Special connection properties" in line with the station type for connection establishment.

- SIMATIC S7 station: PASSIVE connection establishment
- PC station: Active connection setup

"S7 addressing mode" option (for FETCH/WRITE only)

You can select the addressing mode when you configure the FETCH ACTIVE/WRITE ACTIVE mode.

- S7 addressing mode: Byte address
- S5 addressing mode: Word address

This allows applications to access S5 or S7 stations without modifying the addresses. This is, for example, an interesting option for existing S5 applications that are now to be used unchanged to access S7 stations.

The addressing mode for accessing SIMATIC S7 (option selected) is set by default.

See also

Connection configuration for FETCH/WRITE (Page 657)

Use of FETCH/WRITE (Page 655)

ISO-on-TCP OPC properties

Relevance

Connection configuration for OPC in the parameter group "Properties > General > OPC".

Here, you set connection-specific properties for an ISO-on_TCP connection used by the OPC server.

You can set the following parameters:

Timeout for job

If a pending job cannot be processed during the wait time set here, the connection is reset and the job acknowledged with an error.

Range of values: 0 to 99999 ms

- Default setting: 15000 ms

Send buffer size

Default size of the default send buffer. Additional send buffers of any length (up to 65535 bytes) may be used at runtime.

Range of values:

- Default setting: 256 bytes

- Range of values: 65535 bytes maximum

"Optimization of direct access for tag services" option

If you use this option, the OPC server attempts to combine direct accesses to several tags to minimize the number of jobs that need to be sent.

This option is enabled by default.

"Immediate response when interrupted connection is detected" option

If you select this option, you specify that the application is notified that a communication job (FETCH, or WRITE) cannot be executed when a connection is unavailable. The code is sent regardless of whether or not a current attempt to establish the connection succeeds.

If you do not select this option, the application is informed of the nonavailability of a connection only when the connection times out.

The option described here is relevant and available only when the two modes listed below are used:

- The connection is configured as a permanent connection.
  Connections can be configured as permanently established connections (in "Details of connection establishment"). This means that jobs can be forwarded to the partner with the shortest possible reaction time without needing to establish the connection first.
  If there is a disruption on the connection, the system automatically attempts to re-establish the connection. In this situation, there may be delays before a message can be forwarded to the application.
- You are using the FETCH or WRITE job types (can be selected in "Options").
  Only with these job types is an immediate reaction necessary in certain situations if there is no connection.

TCP connection

TCP - special connection properties

Relevance

Connection configuration in the parameter group "Properties > General > Special connection properties".

Connection parameters specify additional properties for the local connection endpoint.

Establishing an active connection

Use this option to specify whether the connection is established from this device. This is the default option if the address of the partner is specified.

- ON: Connection is established actively.
- OFF: Connection is established by the partner.

If you selected "Unspecified" as the connection partner when you created the connection, the option is deselected by default. If you select the option you must specify the address of the partner in the "Addresses" tab.

Note

Effect on the mode

Note the effect on the mode.

- S7-300 / S7-400
  If you want to use FETCH or WRITE mode (see the "Options" parameter group), disable the "Active connection establishment" option.
- OPC server
  If you want to use FETCH or WRITE mode (see the "Options" parameter group), enable the "Active connection establishment" option.

FTP mode (with S7-300 / S7-400 - connection unspecified)

Selecting this option has the following results:

- The TCP connection is now used as an FTP connection.
- Parameter group "Address details": The addresses are specified automatically (port = 21; note: This setting is not visible under "Address details").
- Parameter group "Options": The mode is permanently set to FTP.

The "Active connection establishment" option is irrelevant and is not available.

Requirement: This option can only be selected for certain CPs with S7-300 / S7-400 when using an unspecified TCP connection.

TCP OPC establishment details

Relevance

Connection configuration for OPC in the parameter group "Properties > General > Establishment details".

Options for connection establishment

Entries in this area are possible only if you have selected active connection establishment in "Special connection properties".

Choose from the two following alternatives:

- The connection is established permanently
  The connection is established permanently when you start up the PC station and is re-established following an interruption on the connection.
- The connection is established when it is required (access to tag)
  You can also specify how long the connection should stay up after access.

Connection establishment monitoring

- Timeout: Error wait time during the connection establishment
  If, during this wait time, the connection cannot be established, the connection is reset and a pending request acknowledged with an error.
  Range of values: 0 to 99999 ms
  Default: 15000 ms

Note

The connection establishment monitoring that can be set here, has the following relationship with the time monitoring that can be set in the module configuration: The lower value is valid.

Time monitoring can be configured in the module configuration in "Properties Ethernet interface".

TCP address details

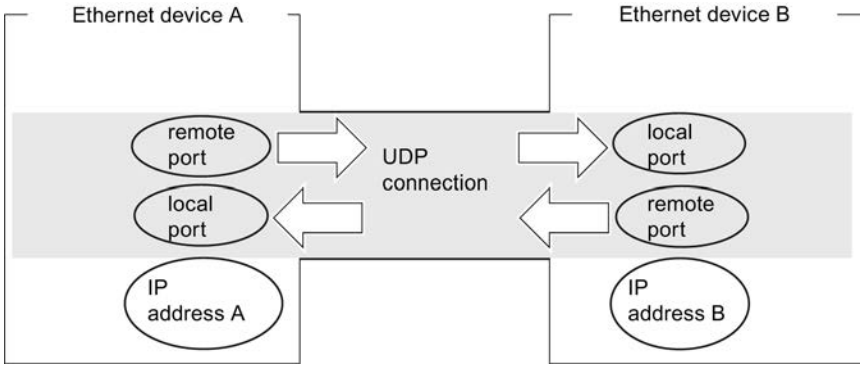
Relevance

Connection configuration in the parameter group "Properties > General > Address details".

Address parameters and connection types

With TCP, the communication partners are addressed through the local and remote endpoint as follows.

- Local addresses: Local IP address and local port
- Remote addresses: Remote IP address and remote port



Ports

The ports or the port addresses define the access point to the user program within the station/CPU. They must be unique within the station/CPU!

- Specified connection
  In the configuration of the Ethernet CP and the Ethernet target Station, the ports of a TCP connection must be crossed over.
  - Remote port (on the Ethernet CP) = local port (on the target station)
  - Local port (on the Ethernet CP) = remote port (on the target station)
- Unspecified connection
  Some address parameters are not specified: Unspecified connections (Page 596)

Port range of values - see also

Port settings (Page 647)

TCP options

Relevance

Connection configuration in the parameter group "Properties > General > Options".

Operating mode

- SEND/RECV:
  This is the standard mode with the AG_SEND/AG_LSEND, and AG_RECV/AG_LRECV FCs.
- FETCH/WRITE:
  System memory areas in the SIMATIC S7 can be accessed from the following devices if you select one of the operating modes below for the TCP connection:
  - SIMATIC S5
  - SIMATIC PC stations
  - Third-party devices

  The connection can then only be used for this mode. It is then not possible to send or receive using the AG_SEND/AG_LSEND or AG_RECV/AG_LRECV FCs.
  - FETCH
    Selecting this operating mode enables direct read access to data areas in the SIMATIC S7. Access can, for example, be from a SIMATIC S5 station or a third-party station.
  - WRITE
    Selecting this operating mode enables direct write access to data areas in the SIMATIC S7. Access can, for example, be from a SIMATIC S5 station or a third-party station.
- FTP
  If you have selected the "Use for FTP protocol" option in "Properties > General > Special connection properties", "FTP" will be displayed here and the TCP connection is then used as an "FTP connection"; no other options can be selected.
  Requirement: The option can only be selected for an unspecified TCP connection.
- SPEED SEND/RECV
  Selecting the SPEED SEND/RECV mode allows you to use the AG_SSEND and AG_SRECV instructions.
  The AG_SSEND/AG_SRECV (SPEED SEND/RECEIVE) instructions provide faster and therefore optimized transmission within an S7-400 station. Check the device documentation to see whether this operating mode is supported by the CP you are using.

FETCH/WRITE requirements: Mode of connection establishment

Select one of the following options for connection establishment in the "General" tab as suitable for the station type.

- SIMATIC S7 station: PASSIVE connection establishment
- PC station: Active connection establishment

"S7 addressing mode" option (for FETCH/WRITE only)

You can select the addressing mode here when you configure for FETCH/WRITE mode.

- S7 addressing mode: Byte address
- S5 addressing mode: Word address

This allows applications to access S5 or S7 stations without modifying the addresses. This is particularly useful for existing S5 applications, for example, that can now be used unchanged when accessing S7 stations.

The addressing mode for accessing SIMATIC S7 (option selected) is set by default.

See also

Connection configuration for FETCH/WRITE (Page 657)

Use of FETCH/WRITE (Page 655)

TCP OPC properties

Relevance

Connection configuration for OPC in the parameter group "Properties > General > OPC".

Here, you set connection-specific properties for a TCP connection used by the OPC server.

You can set the following parameters:

Job timeout

If a pending job cannot be processed during the wait time set here, the connection is reset and the job acknowledged with an error.

Range of values:

- Default setting: 15000 ms

Send buffer size

Default size of the default send buffer. Additional send buffers of any length (up to 65535 bytes) may be used at runtime.

Range of values:

- Default setting: 256 bytes
- Range of values: 65535 bytes maximum

Optimization of direct access for tag services

If you use this option, the OPC server attempts to combine direct accesses to several tags to minimize the number of jobs that need to be sent.

This option is enabled by default.

Use miniprotocol

With this option, you specify whether or not the data is transferred as individual fields with length information about the field size (2 bytes in the job buffer). The recipient then reads the data out according to the length information. This property must be configured identically for both communications partners.

As default, the option is disabled; the recipient then reads the data out according to the set receive buffer size.

- Receive buffer size
  If you do not use a miniprotocol, you can set the size of the receive buffer here.
  Range of values: 0 to 65535 bytes
  - Default setting: 256 bytes

Port settings

Port settings

The ports or the port addresses define the access point to the user program within the station/CPU. They must be unique within the station/CPU!

The following table provides information about the value range:

Port addresses		Use/note
0		Permanently assigned; must not be used!
1..1023		Assigned by default, should not be used (well-known ports)
1024..49151	Ports for application-specific protocols	
	from 2000 to 5000	Range in which the configuration tool searches for and allocates a free port address. You can set the port address in this range individually.
	5001..49151	The port addresses above 5000 are used by the system! Note: If you want to use these port addresses, please contact your system administrator to verify availability.
49152..65535		Dynamically assigned ports The use of these ports is not recommended.

The following local port numbers are reserved; you should not use them for any other purpose during connection configuration.

Port number	Protocol	Service
20, 21	TCP	FTP
25	TCP	SMTP
80	TCP	HTTP
102	TCP	RFC1006
135	TCP	RPC-DCOM

Port number	Protocol	Service
161	UDP	SNMP_REQUEST
34964	UDP	PN IO
65532	UDP	NTP
65533	UDP	NTP
65534	UDP	NTP
65535	UDP	NTP

UDP connection

UDP address details

Relevance

Connection configuration in the parameter group "Properties > General > Address details".

Address parameters and connection types

For UDP the communication partners are addressed through the local and remote endpoint as follows.

- Local addresses:
  Local IP address and local port
- Remote addresses:
  Remote IP address and remote port

Note

The term "connection" is also used here for UDP, because during configuration – as for TCP as well – the communication partners are assigned to each other and thus are logically "connected". In actuality, there is no explicit connection setup between communication partners during station operation in UDP.

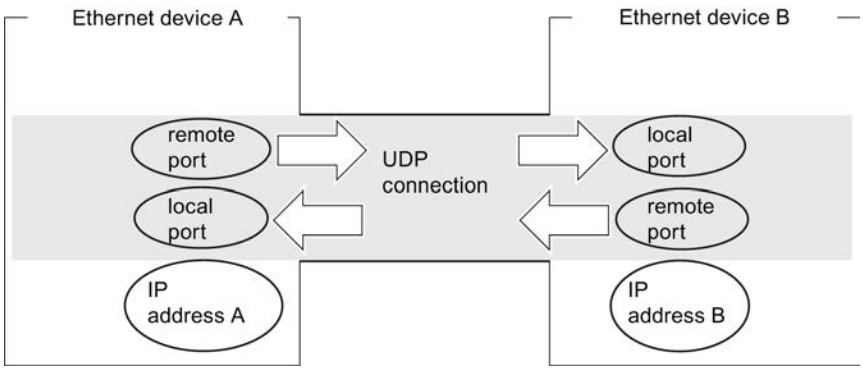


Figure 8-3 Sending and receiving over a UDP connection

Depending on the desired connection type, the remote address parameters are either specified or left open in the configuration.

- Specified UDP connection
  You have specified the connection of a target station when creating the new connection.
- UDP connection specified as broadcast
  You have specified "Broadcast" as the partner when creating the new connection (requires the connection to be created interactively or initially as an unspecified connection).
- UDP connection specified as multicast
  You have specified "Multicast" as the partner when creating the new connection (requires the connection to be created interactively or initially as an unspecified connection).
- Unspecified UDP connection
  You have selected "Unspecified" as the connection type when creating the connection.

Unspecified UDP connection

You can use the unspecified UDP connection in two ways:

- Available UDP connection
  To configure an available UDP connection, select the option "Address assignment in the block" option. In this case, entries can no longer be made in the input fields for the remote IP address and the remote port because the target addresses are now specified by the user program.
- Connection to a "third-party station" in a different project.
  You can specify the remote IP address and the port for any target station. The target station can be inside or outside of the current project.

Note

Partner addresses must be specified

Since there is no connection setup in UDP (datagram service), communication via the configured UDP connection is only possible if the partner addresses (IP address and port) are also specified.

See also

Unspecified connections (Page 596)

Connection to all broadcast devices

If you select "All broadcast devices" for the connection partner, you are specifying that UDP frames will be sent to all accessible broadcast devices.

In the "Address details" parameter group under the IP address (IP) for the partner, a valid broadcast address in the network is suggested for the partner.

Under PORT, you must enter a port address that is appropriate for all partners to be accessed.

Port settings - see also

Port settings (Page 654)

UDP with broadcast and multicast

Application

When selecting the connection partner for UDP connections, you also have the two options:

- Connection to all broadcast devices
  If you select "All broadcast devices" for the partner, you specify are specifying that UDP frames are sent to all accessible broadcast devices.

  Note

  With broadcast, S7-CPs can only be used to send; an S7-CP does not allow receipt of user data on broadcast connections, (see below).

- Connection to all multicast devices
  If you select "All multicast devices" for the partner, you are specifying that UDP frames are sent to all accessible broadcast devices.
  Multicast and broadcast are special connection options that are supported by Industrial Ethernet CPs on UDP connections and can be configured.

The frames are sent without acknowledgement because the UDP protocol does not provide for any acknowledgements. This is intended to prevent a data flood of acknowledgements. For example, if frames are sent to 100 partners, then 100 acknowledgements (1 per partner) would arrive at the same time. Such a flood of data could not be analyzed by the sender module.

See also

Creating UDP / FDL connections with broadcast / multicast (Page 622)

When do I use multicast instead of broadcast?

To enable simultaneous sending of a frame to a number of partners, the connection option "Multicast for UDP connections" was introduced.

As opposed to the broadcast connection option, this connection type can also receive frames that are sent to multiple devices in the multicast group.

By specifying a specific receiver group (multicast group), you can prevent extra load on recipients for which the data is not intended. For this reason, multicast is always a better solution than broadcast if frames are sent to groups of partner stations.

Why does an S7-CP not allow user data to be received on broadcast connections?

S7-CPs do not receive any broadcast frames for user data transfer. Only usable frames, such as an ARP request, are forwarded beyond the LAN controller and evaluated. However, S7 CPs can send broadcast frames to the network.

Reason:

Often there is a requirement to send frames to a number of partner devices from one device. In this regard, the important thing is that the frames are sent at the same time and arrive at the partner devices at virtually the same time. Consequently, sending and receiving of

broadcast frames is always required. With a broadcast message, the frame is also actually received by all devices in the network.

A typical application is a situation where broadcast frames are required to find a MAC address for an IP address (ARP request).

Therefore a communication module must always pick up broadcast frames and analyze them from the software side. A significant disadvantage in this regard is that if there are too many broadcast frames on the network, performance drops significantly. This is due to the fact each individual module must process all broadcast frames in order to determine whether the frame was destined for it.

To avoid the disadvantages cited above, the S7-CPs behave as follows relative to broadcast:

- After receipt, the broadcast frames are filtered out with high priority for all Ethernet CPs. In other words, all unusable frames are discarded immediately. Only usable frames, such as an ARP request, are forwarded beyond the LAN controller and evaluated. This prevents the possible negative influence of broadcast frames on the other connections.

Parameter group "Address details" - Connection to all multicast devices

By selecting the "All multicast nodes" for the connection partner, you are specifying the following:

- Sent UDP frames are delivered to all accessible multicast devices of the multicast group.
- The local device for multicast frames is ready to receive in the specified multicast group.

The multicast group is specified via the IP address and the port addresses.

Under IP address (IP) for the partner, an valid IP address in the network is suggested for the multicast group. With multicast, the partner is always a group of recipients (multicast group).

Enter a PORT address under PORT for all partners that you want to access. It must apply to all of these partners.

In principle, it is possible to address multiple multicast groups under one IP address. To do this you can create multiple UDP connections with the same IP address but different PORT addresses.

Note

You should assign identical port addresses for the local port and for the partner port within a multicast group. This is the only way that frames can be both sent and received by the CP within the one multicast group!

Note the following example for three devices in the multicast group:

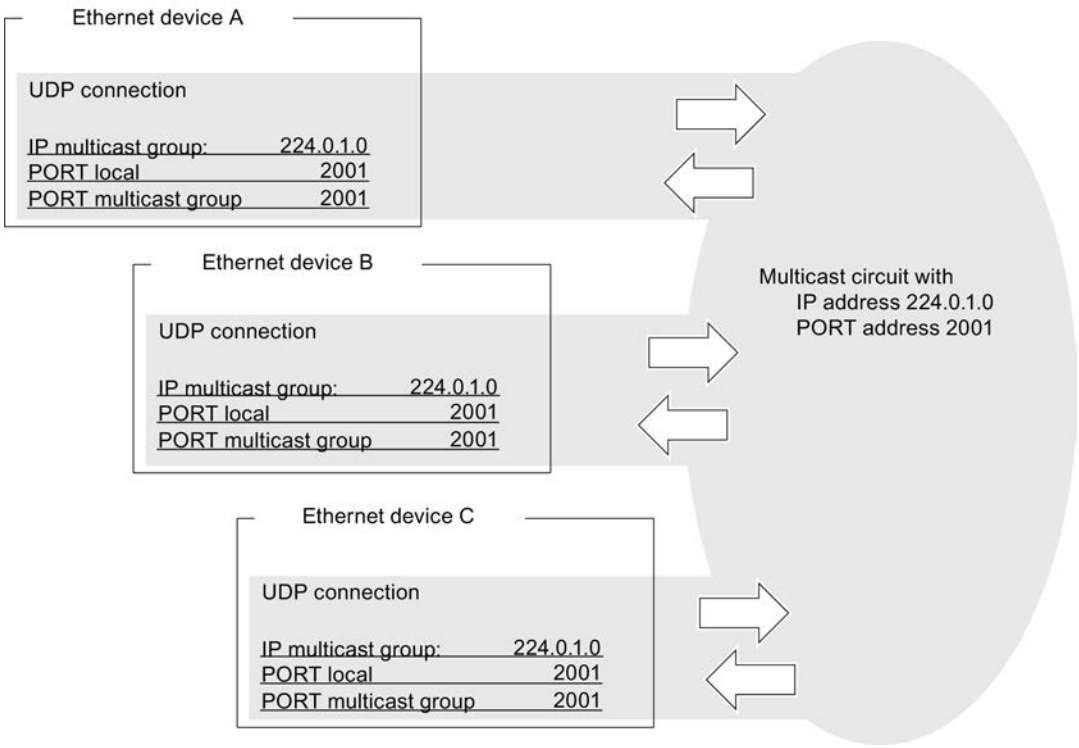
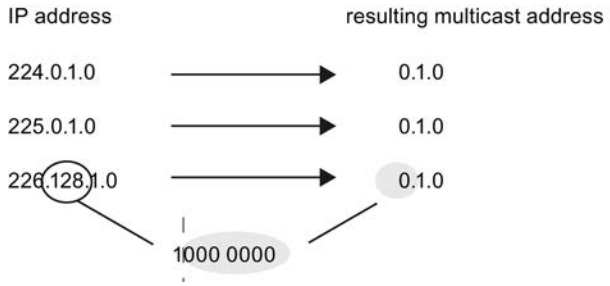


Figure 8-4 Sending and receiving in a multicast group via identical port addresses

IP addresses for IP multicast

- Value range
  For IP multicast, the IP addresses from 224.0.0.0 to 239.255.255.255 can be used. Because the IP addresses up to 224.0.0.255 are reserved for special purposes, we recommend using the IP addresses starting from 224.0.1.0 (default setting) for IP multicast.
- Identification of the multicast group
  A multicast group is not identified via the entire IP address; rather, the first address byte and the most significant bit of the second address byte are ignored. This is an important feature, because it means that apparently different IP addresses can address the same multicast group.
  Example:
  The following IP addresses each address the same multicast group.



Free UDP connection

Program-controlled addressing

A free UDP connection enables program-controlled addressing of the communication peer. Communication between two devices in Industrial Ethernet has the following characteristics:

- Data transfer is bidirectional, i.e. you can send and receive simultaneously on the UDP connection.
- The local device is specified by configuration. The remote device is entered by the user program in the request header of the request buffer with the AG_SEND call. Thus, any device can be accessed on the Ethernet/LAN/WAN.
- The sender's IP address and port can be read from the request header of the AG_RECV.

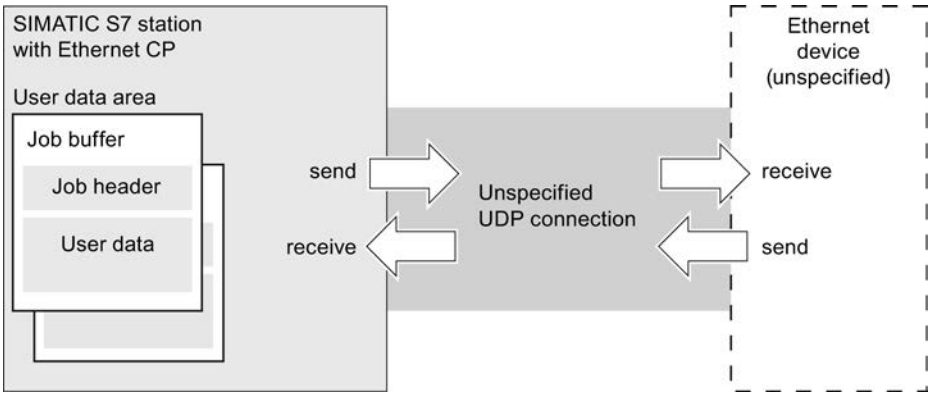


Figure 8-5 Sending and receiving via an unspecified UDP connection - Addressing via program

Data volumes and configuration limits

See the NO TAG manual provided with the Ethernet CP for the number of UDP connections that the respective Ethernet CP can support. By adding additional CPs, you can increase the number of connections per device.

Up to 2042 bytes of user data can be transferred per request buffer. The request header occupies an additional 6 bytes.

UDP options

Relevance

Connection configuration in the parameter group "Properties > General > Options".

Operating mode

- **SEND/RECV:**
  This is the standard mode with the AG_SEND/AG_LSEND, and AG_RECV/AG_LRECV FCs.
- **SPEED SEND/RECV**
  Selecting the SPEED SEND/RECV mode allows you to use the AG_SSEND and AG_SRECV instructions.
  The AG_SSEND/AG_SRECV (SPEED SEND/RECEIVE) instructions provide faster and therefore optimized transmission within an S7-400 station. Check the device documentation to see whether this operating mode is supported by the CP you are using.

Port settings

Port settings

The ports or the port addresses define the access point to the user program within the station/CPU. They must be unique within the station/CPU!

The following table provides information about the value range:

Port addresses		Use/note
0		Permanently assigned; must not be used!
1..1023		Assigned by default, should not be used (well-known ports)
1024..49151	Ports for application-specific protocols	
	from 2000 to 5000	Range in which the configuration tool searches for and allocates a free port address. You can set the port address in this range individually.
	5001..49151	The port addresses above 5000 are used by the system! Note: If you want to use these port addresses, please contact your system administrator to verify availability.
49152..65535		Dynamically assigned ports The use of these ports is not recommended.

The following local port numbers are reserved; you should not use them for any other purpose during connection configuration.

Port number	Protocol	Service
20, 21	TCP	FTP
25	TCP	SMTP
80	TCP	HTTP
102	TCP	RFC1006
135	TCP	RPC-DCOM
161	UDP	SNMP_REQUEST

Port number	Protocol	Service
34964	UDP	PN IO
65532	UDP	NTP
65533	UDP	NTP
65534	UDP	NTP
65535	UDP	NTP

FETCH/WRITE mode

Use of FETCH/WRITE

FETCH/WRITE

You can use the FETCH/WRITE services to access system memory areas of the S7 CPU.

- FETCH: Read data directly
- WRITE: Write data directly

Access is possible from the following devices:

- SIMATIC S5
- SIMATIC PC stations
- Non-SIMATIC devices

Connection types

The FETCH/WRITE services can be configured and used in SIMATIC S7 with the following connection types:

- ISO transport connections
- ISO-on-TCP connections
- TCP connections

System memory

You can use FETCH or WRITE to access the following operand ranges in the SIMATIC S7 system memory:

- Data blocks (DB)
  (Note the following limit for DB access: the highest DB no. is 255)
- Bit memory (M)
- Process image input (PII)
- Process image output (PIQ)
- I/O area inputs (PIW, PID, PIB)
- I/O area outputs (PQW, PQD, PQB)

- Counters (C)
- Timers (T)

Link to non-SIMATIC systems

In principle, FETCH and WRITE can be used to access S7 system memory areas from any non-SIEMENS device.

To implement this access for PC applications, for example, you must become acquainted with the PDU structure for the jobs. Refer to the device documentation for this purpose.

Alarms in the diagnostic buffer

FETCH/WRITE access may give rise to negative acknowledgements from the S7 CPU and corresponding connection-specific entries in the diagnostic buffer.

Table 8-4 Alarm coding in the diagnostic buffer at FETCH/WRITE

Coding	Meaning
01_H	Hardware error
03_H	Object access is not allowed.
05_H	Invalid address (syntax ID, range, type, bit number)
06_H	Data type is not supported.
07_H	Data type is inconsistent.
$0A_H$	The object does not exist, or the range limit was exceeded.
FF_H	Internal protocol error

Additional information

The PDU structure needed to access non-SIMATIC systems is described in the documentation for S7 CPs.

Connection configuration for FETCH/WRITE

Configuring the operating mode to suit the type of device

The following operating modes can be configured for the communication endpoint, according to the type of station:

- SIMATIC S7 station: FETCH PASSIVE/WRITE PASSIVE
  If you select either FETCH PASSIVE or WRITE PASSIVE mode for the connection, the system memory areas in SIMATIC S7 can be accessed directly from a SIMATIC S5 station or a non-SIMATIC station (unspecified connection).
  The connection can then only be used for this mode. It is then not possible to send or receive using the AG_SEND/AG_LSEND or AG_RECV/AG_LRECV FCs.
  The connection is established passively, which means that only the partner station (SIMATIC S5 station, a PC station, or a non-SIMATIC station) can establish the connection. You make the necessary settings under "Properties > Options". Once you have selected the operating mode, you will not be able to change the connection setup type.
- SIMATIC PC station: FETCH ACTIVE/WRITE ACTIVE
  If you select either FETCH ACTIVE or WRITE ACTIVE mode, the system memory areas in the SIMATIC S7 or SIMATIC S5 station can be accessed directly from the PC station. The connection is established actively, which means that the partner station must await the connection to be established (passive connection setup for the partner). You make the necessary settings under "Properties > Options". Once you have selected the operating mode, you will not be able to change the connection setup type.

"S7 addressing mode" option

You can select the addressing mode when you configure FETCH ACTIVE/WRITE ACTIVE mode. This defines how the address information in the FETCH/WRITE call in the SIMATIC S7 station will be interpreted when DBs are accessed:

- S7 addressing mode: Byte address
- S5 addressing mode: Word address

This allows applications to access S5 or S7 stations without modifying the addresses. This is particularly useful for existing S5 applications, for example, that can now be used unchanged when accessing S7 stations.

The addressing mode for accessing SIMATIC S7 (option selected) is set by default.

FDL connection

FDL connections - overview

Area of application

Data transmission over configured FDL connection is suitable for transmission of contiguous data blocks between two or more PROFIBUS stations.

There is a distinction between

- Specified FDL connection
  The communication stations are uniquely specified by the connection configuration.
  The connection partner can be inside or outside of the STEP7 project.
- Unspecified FDL connection (free layer 2 access)
  The address of the connection partner remains open in the configuration. The communication stations are determined by address information in the communication job of the user program. This means up to 126 stations can be reached via one configured unspecified FDL connection, if these stations support FDL connections.
  The connection partner can be inside or outside of the project.
- FDL connection with broadcast
  All stations ready to receive broadcasts on the PROFIBUS are reached.
- FDL connection with multicast
  All stations belonging to the multicast group on the PROFIBUS are reached.

Note

Configuration prerequisite

If FDL connections will be used, the CP mode of the PROFIBUS CP **should not be set** to **DP slave passive**.

Note

Priority of the telegrams

Note that the PROFIBUS-CPs for SIMATIC S7 send the telegrams in "LOW" priority.

Partner stations (SIMATIC S5, S7, or third-party stations) must also use "LOW" priority, otherwise a connection willl not be established.

FDL address details

Relevance

Connection configuration in the parameter group "Properties > General > Address details".

Address parameters for FDL connections

An FDL connection is specified by the local and remote connection endpoint. This includes:

- PROFIBUS address of the node that is to be reached.
- Local LSAP (Link Service Access Point):
  The local LSAP controls the readiness to receive of the PROFIBUS-CP. The receive resources for data receive in the PROFIBUS-CP are provided on the FDL connection for the LSAP.
- Remote LSAP (Link Service Access Point):
  The remote LSAP controls the send mode of the PROFIBUS-CP. The PROFIBUS-CP sends to the nodes on the FDL connection via the LSAP. The target node must be ready to receive for this SAP.

Specified FDL connection

The relevant local and remote address information is displayed as recommended values for a specified FDL connection. You have the option to set the LSAP addresses individually, if necessary.

Unspecified FDL connection

You can use the unspecified FDL connection in two ways:

- Connection to a "third-party station" in a different project.
  You can specify the remote PROFIBUS address and the LSAP for any target station. The target station can be inside or outside of the current STEP 7 project.
  If the remote address is unspecified, then communication via the FDL connection is not possible.
- Free layer-2 access
  Click the relevant check box to configure a free layer 2 access. The input fields for the remote PROFIBUS address and the remote LSAP can no longer be specified in this case because the target addresses are now specified by the user program.

FDL connection with broadcast

The remote address parameters for FDL connection with broadcast are permanently set. All broadcast nodes can be reached via the PROFIBUS address 127. Receive data is received via LSAP 63 from all broadcast nodes.

The local address parameters are entered in the job header of the message for sending and delivered to the recipient. The user program for the remote partner can thus determine the sender of the broadcast frame.

Note

CP receives only on a broadcast connection

If you use an FDL connection with broadcast, you cannot use any other broadcast connection on the CP.

CAUTION

"Activate cyclic distribution of bus parameters" function

If you select this option for the subnet in "General > Bus parameters", the bus parameters will be sent cyclically during operation as broadcast frames. To avoid frame handling conflicts in the user program that receives broadcast frames, you should take the following action:

- ignore all frames that are sent with an LSAP >56
  or
- disable the option under "General > Bus parameters".

FDL connection with multicast

The remote address parameters for FDL connection with multicast are permanently set. All multicast nodes can be reached via the PROFIBUS address 127. Sending and receiving data is handled by all nodes in the multicast group via the same LSAP (range 1 to 56). This means the value of the LSAP can only be selected locally and is automatically transferred to the remote LSAP.

The local address parameters are entered in the job header of the message for sending and delivered to the recipient. The user program for the remote partner can use this information to determine the sender of the multicast frame.

FDL options

Relevance

Connection configuration in the parameter group "Properties > General > Options".

Operating mode

- SEND/RECV:
  This is the standard mode with the AG_SEND/AG_LSEND, and AG_RECV/AG_LRECV FCs. This mode is the default on FDL connections and this cannot be modified.

FDL OPC properties

Relevance

Connection configuration for OPC in the parameter group "Properties > General > OPC".

Here, you set connection-specific properties for an FDL connection used by the OPC server.

You can set the following parameters:

Timeout

If a pending job cannot be processed during this wait time, the connection is reset and the job acknowledged with an error.

Max number of send retries

Here, specify how many retries there will be for a job if, for example, the connection partner cannot be reached due to a temporary lack of resources.

Send buffer size

The size of the default send buffer is set here.

This setting applies to both acknowledged and unacknowledged sending.

Additional send buffers of any length (maximum 244 or 246 bytes) can be used at runtime.

Default setting:

244 bytes or 246 bytes (with default SAPs, if the default SAP 255 is set for both the local and remote SAP)

Maximum number of parallel send jobs

Maximum number of simultaneous send jobs on the connection configured here.

Number of resources for received jobs

Specify how many receive jobs the OPC server will be able to receive simultaneously on the FDL connection configured here.

Select a value suitable for the expected amount of traffic; in other words,

- For a low total number of connections: Select a higher value to achieve a fast reaction time.
- For a high total number of connections: Select a lower value to avoid resource conflicts.
  - Default setting: 3

Send with higher priority

Specify whether the FDL frames on the connection configured here are sent via the LAN with priority (option enabled).

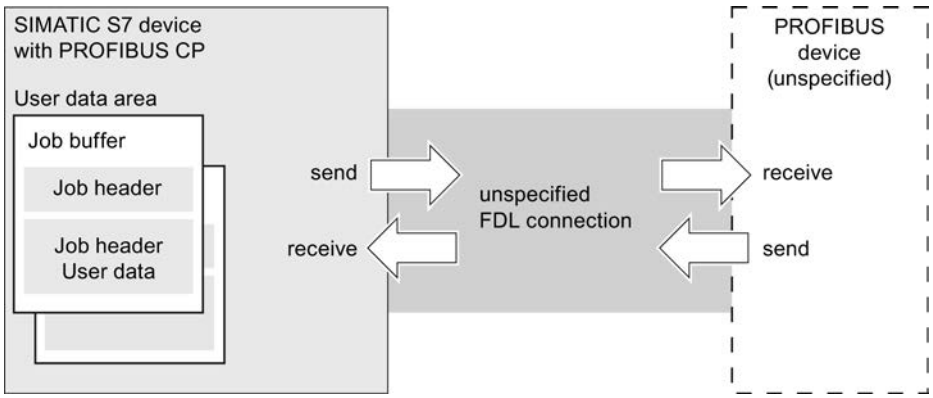
FDL unspecified FDL connection (free layer 2 access)

Properties

An unspecified FDL connection with open layer 2 access enables program-controlled addressing of the communication partner and communication between two stations on PROFIBUS with the following properties:

- Data transfer is bidirectional, which means you can send and receive data simultaneously on the FDL connection.
- The local station is specified by configuration. The remote node is entered by the user program in the job header of the job buffer with the AG_SEND instruction call. This means that each node on PROFIBUS (PROFIBUS addresses from 0 to 126) can be reached.
- The PB address, the LSAP, and the sender service can be read from the job header of the AG_RECV.

The figure below illustrates program-controlled addressing in the job buffer:



Data volume and configuration limits

Each job buffer can transfer up to 236 bytes of user data. The job header reserves an additional 4 bytes.

FDL connection with broadcast

Properties

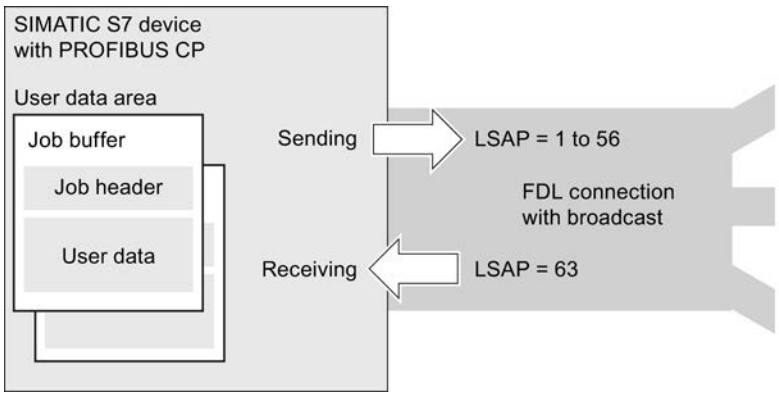
A broadcast connection sends a message to multiple recipients with **one** job. This means messages can be received on the same broadcast connection that are simultaneously received by other stations on the PROFIBUS.

The properties can be summarized as follows:

- Data transfer is bidirectional, which means you can send and receive data simultaneously on the broadcast connection.
- Sending and receiving takes place via the FDL service SDN (Send Data with No Acknowledge).

- The AG_SEND call needs a job buffer for sending. The job header of this buffer must be reserved.
- The PB address, the LSAP, and service of the broadcast sender can be read from the job header of the AG_RECV.
- The LSAP range from 1 to 56 is reserved for sending. LSAP 63 is reserved for all broadcast stations for receiving.

The figure below illustrates sending and receiving via an FDL connection with program-controlled broadcast addressing:



Configuring an FDL connection with broadcast

When creating the FDL connection select "All broadcast stations" as connection partner/station.

See also

Creating UDP / FDL connections with broadcast / multicast (Page 622)

Data volume and configuration limits

PROFIBUS-CP only supports **one** broadcast connection.

Each job buffer can transfer up to 236 bytes of user data. The job header reserves an additional 4 bytes.

Note

CP receives only on a broadcast connection

If you use an FDL connection with broadcast, you cannot receive any additional broadcast connection messages on the respective CP, neither on any FMS connection with broadcast.
Reason:
The receive LSAP for broadcast (63) is reserved with a broadcast connection.

FDL connection with multicast

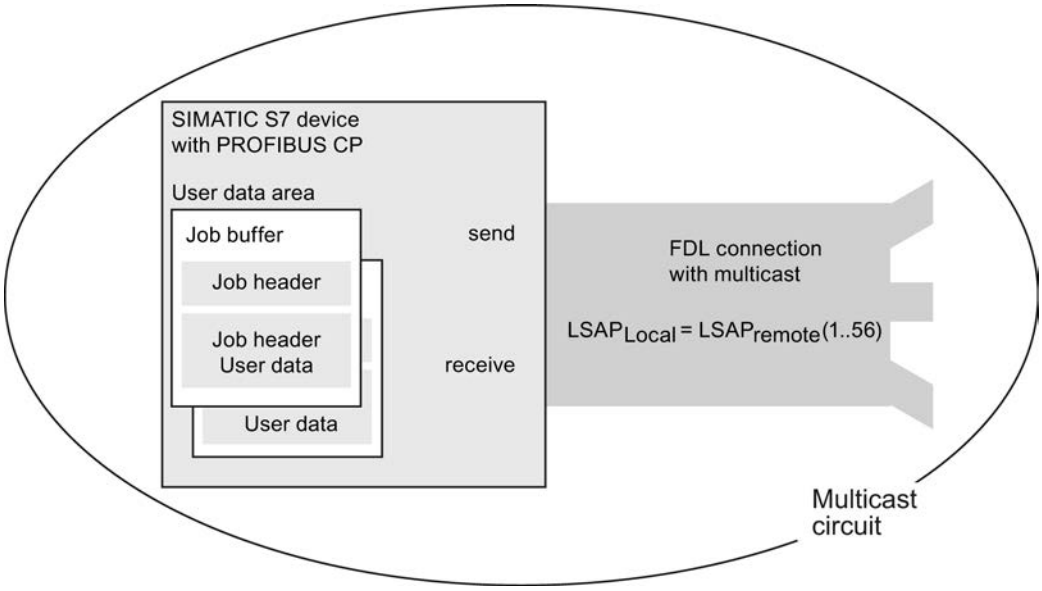
Properties

An FDL connection with multicast sends a message to multiple recipients of a multicast group with one job.

The properties can be summarized as follows:

- Data transfer is bidirectional, which means you can send and receive data simultaneously on the FDL connection with multicast.
- Sending and receiving takes place via the FDL service SDN (Send Data with No Acknowledge).
- The system sends via an LSAP (range from 1 to 56) that is uniform for the multicast group.
- The AG_SEND call needs a job buffer for sending. The job header of this buffer must be reserved.
- The PB address, the LSAP, and service of the multicast sender can be read from the job header of the AG_RECV.

The picture below illustrates sending and receiving via an FDL connection with program-controlled multicast addressing:



Configuring an FDL connection with multicast

When creating the FDL connection select "All multicast nodes" as connection partner/station.

See also

Creating UDP / FDL connections with broadcast / multicast (Page 622)

Data volume and configuration limits

See the device documentation for PROFIBUS-CP for the number of FDL connections that the respective PROFIBUS-CP supports. The number of connections per station can be increased by accepting additional CPs .

Each job buffer can transfer up to 236 bytes of user data. The job header reserves an additional 4 bytes.

E-mail connection

Setting up an email connection

Emails can be sent from the automation system, but they cannot be received.

To send email in the user program of the S7-CPU, use the send call of the SEND/RECEIVE interface (FC AG_SEND / AG_LSEND).

Configuration overview

One email connection per CP must always be set up to send emails. This email connection specifies the mail server that delivers all emails sent by the CP.

An email connection can be set up as follows:

- Connection configuration (standard application)
  This application is described below.
- User program with FB CP_CONFIG and configuration data block
  There are application ranges for which it is sensible to set up the communication connections with a program-controlled procedure via specific applications rather than via the configuration interface.

Procedure - example of graphic configuration

Configuration of an email connection is described below. The procedure is basically the same as when setting up an unspecified connection. Creating an unspecified connection (Page 620).

Select the connection type, email connection, in the drop-down list box.

This step activates the connection mode: The CPUs available for the selected connection type are selected and highlighted.

1. Click on the CPU from which a connection will be set up.
2. Move the cursor within the CPU and confirm the connection endpoint with another click of the mouse.
   An unspecified connection will be created.
3. In "Properties > Address details" in the Inspector window, enter the address parameters according to the information in the address details:
   E-mail address details (Page 666)

After loading the configuration data the user program can send emails via this email connection.

Using an OPC server as an SMTP server

Instead of an unspecified connection, you can select an OPC server as an SMTP server as the connection partner. This means that the IP address of the SMTP interface of the PC station it is entered in the partner's address details.

Note

The PC interface must support the SMTP protocol

The e-mail connection via the OPC server is only created consistently when an interface with SMTP protocol support is enabled on the PC station. Read the documentation of the module your using.

E-mail address details

Relevance

Connection configuration in the parameter group "Properties > General > Address details".

Parameter

Enter the parameters required for the delivery of the e-mail. You can set the following parameters:

Parameters	Description	Example
E-mail server (SMTP)	Address of the mail server via which the e-mails are sent. The IP address can be specified absolutely or symbolically. The following applies to symbolic names: A valid name has 1 to 64 characters of which at least one character must be a letter. If one or more "." characters are used, the following constraint also applies: A valid name has 1 to 3 characters behind the last "."; at least one of these characters must be a letter. A symbolic name can only be used if the Internet CP knows the address of the Domain Name System (DNS). Make an appropriate entry when configuring the Internet CP in the DNS configuration. A maximum of 64 characters can be entered.	absolute: 140.80.0.4 symbolic name: t-online.de
Default sender name	Specification of an address that is always inserted in the email as sender address, if the sender specification (FROM parameter) is empty in the header of the email. This information is normally irrelevant for delivery of the mail. It is simply information for the mail recipient. Since some mail servers do not forward jobs if there is no sender information, it is advisable to enter information here! A maximum of 126 characters can be entered. **Note:** Remember that a default sender name must be specified if you want a test mail to be initiated by special diagnostics.	Station2.CPU412@xy.factory2.de

Sending email

Procedure

Proceed as follows to send an e-mail:

1. Make the e-mail data available in a data block.
2. Use the AG_SEND or AG_LSEND instruction in the user program.

Requirement

You can send email if the email connection has been set up via the connection configuration. You must use the specified ID to call the FC AG_SEND/AG_LSEND for the connection configuration.

Data block

The entire email, meaning the address information and the message itself, will be set up in any data block. The example in STL notation below shows the appropriate information for the required DB structure.

Use the programming editor to create and enter the DB data.

The following table contains information about email data blocks in STL notation:

Address	Name	Type	Start value	Comment	Entry
0.0		STRUCT			
+0.0	TO[1)]	STRING[40]	"TO:name.name@t-online.de;"	Recipient	Mandatory
+42.0	CC[1)]	STRING[40]	"CC:name.name@t-online.de;"	CC recipient	Optional
+84.0	FROM	STRING[40]	"FROM:plant.werk2@xyz-online.de;"	Sender	Optional
+126.0	SUB	STRING[40]	'SUB:Status Station 7;'	Subject	Optional
+168.0	Text	STRING[100]	'TXT:Fault in plant section 2;'	Mail text	Mandatory
+270.0	Attachment	STRING[4]	'BNY:'	Here the attachment is initiated[3)]	Optional
+276.0	Value1	BYTE	**B#16#27**[2)]	Attachment/binary value[3)]	Optional
+277.0	Value2	BYTE	**B#16#03**[2)]	Attachment/binary value[3)]	Optional
=278.0		END_STRUCT			

1) Multiple recipients can be specified. The information must then be separated by a comma.

2) The information in bold will be delivered to the recipient as attachment

3) Data can also be supplied dynamically.

Comments on the table

- Structure and syntax of the data in the email DB
  The structure suggested here with multiple STRINGs is one of several variants. Crucial are the entries in the column "Initial value" with the IDs contained therein (TO:, SUB:, CC:, FROM:, TXT:, BNY:) which must be used in the DB in precisely this notation to identify the mail content! All entries must end with a semicolon; only the last entry does not need a semicolon.
  The string length is only shown in the table as an example; it can be customized to the actual character count (exception: The string length for identifying the attachment must be specified with [4]).
  For example, another variant would be using only one STRING in total, and assigning the entire text to this string with the identifiers.
- If you encounter a problem entering the "@" symbol, try the input ALT+64 instead.

- Attachments
  The user data entered in the email DB can also be delivered to the recipient entirely or partially as attachment. For this the sender must give the data the identifier "BNY:" .
  The data specified after this identifier will then be delivered to the recipient as attachment.
  In the table the attachment is 2 bytes; this is only an example however. You can enter attachments that are as complex as you like.
- Data length
  The data length specified in the call AG_SEND/AG_LSEND must at least include the length of the data in the DB; note the specifications in the address column of the programming editor (Note: The specification corresponds to the number of bytes).

Sending an email with AG_SEND/AG_LSEND [1)]

Use the FC AG_SEND instruction to send an e-mail or with data lengths >240 bytes, use AG_LSEND. A detailed description of call parameters is available in the online help for the instructions.

See also AG_SEND / AG_LSEND / AG_SSEND (Ind. Ethernet) (Page 7719)

Example:

STL	Explanation
call fc 50	//AG_LSEND block call
ACT := M 10.0	//Bit for job trigger
ID := MW 12	//Connection ID (connection configuration)
LADDR := W#16#0100	//Module address 256Dec. in hardware configuration
SEND := P#db99.dbx10.0 byte 278,	//Address of the data block; DB length
LEN := MW 14	//Length of the data range to be sent
DONE := M 10.6	//Address for return parameter DONE
ERROR := M 10.7	//Address for return parameter ERROR
STATUS := MW 16	//Address for return parameter STATUS

Point-to-point link

Point-to-point communication

Types of communication over PtP links

When communicating over a PtP connection, two different types of communication are distinguished:

- Homogeneous communication
- Heterogeneous communication

Support of PtP communication means that external devices such as printers can be accessed.

Homogeneous PtP communication

The PtP CP forwards data between modules that support the S7 protocol. Users have the full range of communications functions available if these are provided by the end devices connected via PtP (for example CPU with CPU or PG with CPU).

On these connections, the PtP CPs are not the endpoint of the connection but take over the role of a router.

With a homogeneous link, all the communications functions defined in S7 can be used if they are supported by both connection partners. Protocol handling for specific communication instruction calls is therefore not necessary on the CP.

Heterogeneous PtP communication

The PtP CP operates like a gateway using other protocols such as RK512, 3964, open drivers and special protocols. Selected blocks serve as a user interface synchronized with the user program. At the receiving end, the user program acting as server can be accessed via the heterogeneous PtP link once again using communications functions synchronized with the program.

From a communications engineering perspective, the PtP CP is the endpoint of the S7 connection in heterogeneous communication. Communication between a CPU and CP is therefore based on S7 protocol elements. The CP handles the conversion of these protocol elements to the PtP protocol and vice versa.

For heterogeneous communication, the following communication instructions can be used on the CPU:

- BSEND
- BRCV
- PUT
- GET

On the PtP CP, the suitable S7 protocol elements must be implemented for the BSEND, BRCV, PUT and GET instructions as client (CP is active) and PUT and GET instructions as server (CP is passive).

Instructions	Client	Server
BSEND (Page 7376)	Yes	no
BRCV (Page 7379)	Yes	no
PUT (Page 7368)	Yes	Yes
GET (Page 7366)	Yes	Yes

Communications functions

Heterogeneous communication can have two different qualities depending on the application:

- Controlled communications functions:
  With controlled communications functions, it is possible to control the transfer. This means that the transfer can not only be started at the active end but can be enabled at the passive end by the control and the end of the transfer can be detected.
  This functionality is implemented for the communications instructions in S7 with the parameters "EN_R" (ready to receive) and "NDR" (receipt of data complete).
- Non-controlled communications functions:
  With non-controlled communications functions, the transfer is handled by the active partner without the passive partner being able to take any action.

Assigning parameters for a point-to-point link

Hardware for configuring PtP

To establish a PtP connection, you require a CP 441. Communications modules of the type CP 441 have slots allowing the use of one (CP 441-1) or two (CP 441-2) PtP interface modules.

You will find PtP interface modules of the type IF963 in the hardware catalog of the SIMATIC S7-400 under the communications modules for PtP. The IF963 modules are available for the following interface types:

- RS-232C
- TTY
- RS-422/485

Depending on the interface type, the IF963 modules support at most the following communications protocols:

- ASCII
- 3964(R)
- RK512
- Printers
- Modbus master
- Modbus slave

If you use a PtP interface in a CP 441, a PtP subnet is automatically displayed and an unspecified PtP connection is added.

Once you have selected the CP 441 or selected the required interface directly, you can select and configure the protocol in the Inspector window.

PtP connection to an S7 device as the connection partner

You can enter a maximum of 8 connections in the connection table for the local partner. For the connection partner, you enter only **one** connection in the connection table. The connection partner can receive data from the 8 connections via this one connection.

Example: Settings for the local partner:

- 3964(R) protocol
- Communication direction 1: Local --> partner
- max. 8 connections, local IDs: 1000; 1001, ..., 1007

Settings for the connection partner:

- 3964(R) protocol
- Communication direction 2: Partner --> local
- max. 1 connection, local ID: 1000

Number of connections

The maximum number of configurable connections per device (CPU/FM) depends on the resources of the module you are using.

PtP communication direction

Communications direction

For PtP connections, you can specify different communications directions. Select the corresponding check box.

- Local --> partner:
  Connections via which you want to send frames.
- Partner → local:
  Connections via which you want to receive frames.
- Local ↔ partner:
  Connections via which you want to send and receive frames.

Note

You will find more detailed information on specifying the communications direction in the documentation of your module.

PtP link path

Setting the connection path

In "General > Connection path". you can set the path taken by the connection from the local endpoint to the connection partner in the settings relating to the communications direction. You can select the path depending on the configured PtP CPs. Example: The local partner contains two PtP CPs. In this case, you can select the CP via which the connection will run.

Special feature: In contrast to the S7 connections, when configuring PtP connections to an unspecified partner, networking of the local partner is not essential. The communications

partners only need to be networked in your real system before you put the connection into operation.

Selection options for the connection partners

- Via CP:
  For the local partner, you can select the PtP CP via which the connection will run (name of the CP along with the rack and slot).
- Interface:
  SIMATIC S7 CPs have several channels (interfaces) via which the PtP connections can be established. The channel and the protocol used for the channel are displayed (channel/protocol). You configured the protocol with the special configuration software of the CP.
- Type:
  The Interface type " PtP" is displayed here.

"Connection selected via RK512 CPU no.:" (local)

Automation systems capable of multicomputing (for example SIMATIC S7-400) can contain several CPUs. For this reason, you need to specify a CPU number with which the partner can access the connection.

If you have selected **Partner -> Local** or **Local <-> Partner** as the communications direction and want to receive frames from the partner using an SFB, enter the number of the CPU (1 to 4) of the **local partner** here.

The CPU number specified here must match the CPU number entered in the command frame header of the CP (for the structure of the command frame, refer to the documentation of the CP you are using). If they match, the partner accesses the connection using this CPU number.

Note

If you want to send and receive frames using SFBs, complete the two boxes below "Local" and "Partner". You will find the maximum number of connections over which you can send/receive frames in the documentation of the CP you are using.

"RK512 CPU no.:" (partner)

Automation systems capable of multicomputing (for example SIMATIC S7-400) can contain several CPUs. For this reason, you need to specify a CPU number to which the connection will go.

If you have selected **Local -> Partner** or **Local <-> Partner** as the communications direction and want to send frames to the partner using an SFB, enter the number of the CPU (1 to 4) of the **connection partner** here.

The CP enters the CPU no. you have specified in the command frame header of the frame to be sent. Using this CPU number, one of the 4 CPUs of the connection partner will be accessed.

When transferring to third-party devices, it is also possible to specify "0" as the CPU no. This makes broadcast access possible on the partner station. If a CP 441 is the partner, CPU 1 is then always accessed.

Note

If you want to send and receive frames using SFBs, complete the two boxes below "Local" and "Partner". You will find the maximum number of connections over which you can send/receive frames in the documentation of the CP you are using.

PtP - special connection properties

Relevance

You can specify additional properties of the local connection endpoint in the "Properties > General > Special connection properties" parameter group.

Configured at one end

Configured at one end means that the connection partner functions as a server on this connection and cannot send or receive actively.

Establishing an active connection

Use this option to specify whether the connection is established from this device. This is the default option if the address of the partner is specified.

- ON: Connection is established actively.
- OFF: Connection is established by the partner.

If you selected "Unspecified" as the connection partner when you created the connection, the option is deselected by default. If you select the option you must specify the address of the partner in the "Addresses" tab.

Note

Not every node is capable of handling the connection establishment. In this case, the partner must initiate the establishment; the check boxes are grayed out and cannot be changed.

Send operating mode messages

The connection partners can also exchange their operating mode messages via an existing connection.

Select the check box if you want the local device to send its operating mode messages to the partner. The mode messages can then be received on the partner using the USTATUS (Page 7408) instruction (SFB 23).

Using Open User communication (S7-1200, S7-1500)

Basics of Open User Communication

Introduction

Open User Communication is the name given to a program-controlled communication process for communicating via the integrated PN/IE interface of the S7-1200/1500 and S7-300/400 CPUs. Different connection types are available for this communication process.

The main feature of Open User Communication is its high degree of flexibility in terms of the data structures transferred. This allows open data exchange with any communicating devices providing they support the connection types available here. Since this communication is controlled solely by instructions in the user program, event-driven connection establishment and termination is possible. Connections can also be modified by the user program during runtime.

For CPUs with an integrated PN/IE interface, the TCP, UDP, and ISO-on-TCP connection types are available for Open User Communication. The communication partners can be two SIMATIC PLCs or a SIMATIC PLC and a suitable third-party device.

Instructions for Open User Communication

To create the connections, you have various instructions available after opening in the program editor in the "Instructions > Communication > Open User Communication" task card:

- Compact instructions for sending or receiving data via the integrated functions for establishing and terminating the connection (S7-1200/1500 only):
  - TSEND_C (Page 5050) (connection establishment/termination, sending)
  - TRCV_C (Page 5061) (connection establishment/termination, receiving)
- Individual instructions for sending and receiving data or for establishing or terminating connections:
  - TCON (Page 5083) (connection establishment)
  - TDISCON (Page 5091) (connection termination)
  - TSEND (Page 5096) (TCP or ISO-on-TCP: Sending)
  - TRCV (Page 5099) (TCP or ISO-on-TCP: Receiving)
  - TUSEND (Page 5109) (UDP: Sending)
  - TURCV (Page 5112) (UDP: Receiving)

Connection establishment

For Open User Communication, instructions for establishing and terminating the connection must exist for both communication partners. One communication partner sends its data using TSEND, TUSEND or TSEND_C while the other communication partner receives the data using TRCV, TURCV or TRCV_C.

One of the communication partners starts the connection establishment as the active partner. The other communication partner reacts by starting its connection establishment as the passive partner. If both communication partners have initiated their connection establishment, the communication connection is fully established.

Connection configuration

You can specify establishment of the connection via a connection description DB with the TCON_Param, TCON_IP_v4, or TCON_IP_RFC structure by means of parameter assignment as follows:

- Manually create, assign parameters and write directly to the instruction.
- Supported by connection configuration.

Connection configuration supports the establishment of the connection and should, therefore, be given preference over the other methods.

You specify the following in the connection configuration:

- Connection partner
- Connection type
- Connection ID
- Connection description DB
- Address details according to selected connection type

In addition, you specify here which communication partner activates the connection establishment and which partner establishes a connection passively in response to a request from its communication partner.

See also

Principle of operation of connection-oriented protocols (Page 686)

Connection configuration

Overview of connection configuration

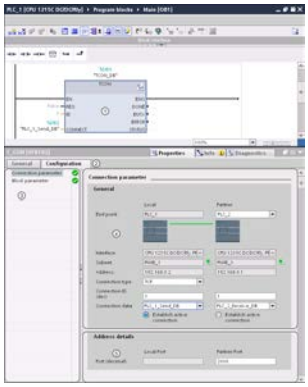
Introduction

You can find the connection configuration in the inspector window of the program editor if you want to program Open User Communication with the communication instructions TSEND_C, TRCV_C or TCON.

Connection configuration supports the flexible functionality of communication programming: The parameters entered for the connection configuration are stored in an automatically generated global DB derived from the TCON_Param, TCON_IP_v4 or TCON_IP_RFC structure. You can modify the connection parameters in this connection description DB.

Structure of the connection configuration

The connection configuration is made up of the following components:



① Communication instruction for TCON, TSEND_C or TRCV_C
② "Configuration" tab in the "Properties" tab
③ Area navigation of the "Configuration" tab
④ General properties of the connection parameters
⑤ Address details of the connection parameters (for selected connection DBs)

"Configuration" tab

Enter the desired connection parameters in the "Configuration" tab. The area navigation of the "Configuration" tab includes the "Connection parameters" group. This group contains the connection configuration. Here, you can enter the parameters for the connections and the address details with system support. Here, you also connect the CONNECT (TCON, TSEND_C, TRCV_C) or ID (TCON, TSEND, TRCV, TUSEND, TURCV) block parameters of the selected communication instructions.

When all the required parameters are assigned, a check mark is set in front of the "Connection parameters" group in the area navigation.

Note

The connection parameter assignment does not check whether the connection IDs and port numbers (TCP, UDP) or TSAPs (ISO-on-TCP, ISO) are unique. When you configure Open User Communication, you should, therefore, make sure that the parameter settings are unique within a device.

See also

Connection parameters with structure according to TCON_Param (Page 688)

Connection parameters with structure according to TCON_IP_v4 (Page 691)

Connection parameters with structure according to TCON_IP_RFC (Page 692)

Description of the connection parameters

Overview

The following table shows the general connection parameters:

Parameter	Description
End point	The names of the local end point and the partner end point are shown. The local end point is the CPU for which TCON, TSEND_C or TRCV_C is programmed. The local end point is, therefore, always known. The partner end point is selected from the drop-down list. The drop-down list shows all available possible connection partners including unspecified connection partners for devices whose data is unknown in the project. For S7-1500, broadcast can be selected as the partner end point (message to all subnet devices). For S7-1500 CPs/CMs, multicast can also be selected as the partner end point (message to a group within the subnet). The connection type is converted automatically to UDP in this case. As long as no connection partner is set, all other parameters in the mask are disabled.
Interface	The interface of the local end point is displayed. If multiple interfaces are available, e.g., by means of CPs or CMs, the interface can be selected from the drop-down list. To display or select the partner interface, a specified partner end point must first be selected.
Subnet	The subnet of the local end point is displayed, provided this exists. The partner subnet is displayed only after the partner end point has been selected. If at least one of the two connection partners is not connected with a subnet, the two connection partners are connected with each other. A connection between partners in different subnets is only possible with IP routing. The IP routing settings can be edited in the properties of the relevant Ethernet interfaces.
Address	The IP address of the local end point is displayed. The IP address of the partner is displayed only after the partner end point has been selected. If you have selected an unspecified connection partner, the input box is empty and has a red background. In this case, you need to specify a valid IP address. Broadcast (S7-1500 only): If "Broadcast" is set as the partner end point, a non-editable IP address with host address 255 is entered automatically for the connection partner. The network allocation corresponds to that of the sender. Example: Local IP address 192.168.0.1, partner IP address 192.168.0.255. Multicast (S7-1500 CPs/CMs only): If "Multicast" is set as the partner end point, the editable IP address 224.0.1.0 is entered automatically for the connection partner.
Connection type	Select the connection type you want to use from the "Connection type" drop-down list: • TCP • ISO-on-TCP • UDP With the S7-1500, you can also select the ISO connection type at the configuration type of the configured connections for TSEND_C and TRCV_C or TCON. The connection types can only be used for partners that support the corresponding protocol.

Parameter	Description
Connection type (for S7-1500 only)	With the S7-1500, two different configuration types can be set for TSEND_C and TRCV_C: • Programmed connections use program blocks for the connection description. • Configured connections are saved for the configuration and are only created after download to the device in runtime. You can also use the configured connection to select the connection type ISO. The specified configuration method depends on the selected connection type. If both configuration methods are possible, the programmed connection is preset. The same configuration method must be set for both connection partners.
Connection ID	Enter the connection ID in the input box. You can change the connection ID in the input boxes or enter it directly in TCON. Ensure that the connection ID assigned is unique within the device.
Connection data	The names of the connection description DBs for the connection description structured according to TCON_IP_v4, TCON_IP_RFC or TCON_Param are displayed in the drop-down lists. The drop-down list is still empty after selection of a connection partner. You can use the drop-down list to generate a new data block or to select an existing data block. This data block is filled automatically with the values from the connection configuration. The name of the selected data blocks is entered automatically in the CONNECT block parameter of the selected TSEND_C, TRCV_C or TCON instruction. From the drop-down list, you can also reference another valid data block. If a DB is referenced using the CONNECT input parameter of the TSEND_C, TRCV_C or TCON extended instructions and this does not correspond to the structure of a TCON_IP_v4, TCON_IP_RFC or TCON_Param, the drop-down list is shown with no content on a red background.
Connection name (for S7-1500 only)	If the connection type of the configured connections is set for TSEND_C and TRCV_C for the S7-1500, the "Connection data" parameter is replaced with the "Connection name" parameter. The name of the configured connection serves here as the connection data. The drop-down list is still empty after selection of a connection partner. You can use the drop-down list to generate a new connection or to select an existing connection. If needed, a data block is created and automatically filled with the values from the connection configuration. The name of the data block is entered automatically in the CONNECT block parameter of the TSEND_C or TRCV_C instruction. You can also reference an existing connection from the drop-down list.
Active connection establishment	Use the "Active connection establishment" check box to specify the active partner of the Open User Communication (only with TCP and ISO-on-TCP).
Port (only with TCP and UDP)	Address component for a TCP or UDP connection. The default after creating a new TCP connection is 2000. You can change the port numbers. The port numbers must be unique on the device!
TSAP (ISO-on-TCP only)	Address component for an ISO-on-TCP connection. The default value after creating a new ISO-on-TCP connection is E0.01.49.53.4F.6F.6E.54.43.50.2D.31 (S7-1200/1500) or E0.02.49.53.4F.6F.6E.54.43.50.2D.31 (S7-300/400). You can enter the TSAP-ID with an extension or as an ASCII TSAP. The TSAPs must be unique on the device!

Note

UDP connection for the "Broadcast" setting (S7-300/400/1200)

The parameters of the UDP connection for the "Broadcast" setting for the partner end point are stored in a connection description DB TCON_IP_v4 : With respect to UDP communication with TCON and TUSEND/TURCV , the TCON_IP_v4 is not filled with any partner parameters (value=0). However, the partner address and the partner port are necessary for sending the data and must be entered by the user in the TADDR_Param . The TADDR_Param for UDP communication is referenced by the TUSEND-/TURCV block parameter ADDR . The values for both parameters can be taken from the connection configuration.

The configuration must also be adapted for the other recipients of UDP communication. In order to receive broadcast frames, the partner port must be configured at the receiver end. For this purpose, the RemotePort parameter of the TADDR_Param must be filled at the ADDR block.

Note

Communication via TSEND_C and TRCV_C (S7-1500)

When TSEND_C and TRCV_C are used, a separate TSEND_C and TRCV_C block pair with a configured connection is required for each communication. Multiple TSEND_C and TRCV_C block pairs cannot simultaneously use the same configured connection for communication.

Additional connections for a TSEND_C or TRCV_C instruction can be created in the inspector window for the connection parameters using the appropriate button next to the connection data.

The connections configured using TSEND_C and TRCV_C are displayed in a connection table in the inspector window under "Properties > Configuration > Overview of configured connections" when the TSEND_C or TRCV_C block is selected.

See also

Assignment of port numbers (Page 693)

TSAP structure (Page 696)

Examples of TSAP assignment (Page 698)

Ability to read back connection description parameters (Page 695)

Creating and assigning parameters to connections (Page 682)

Connection parameters with structure according to TCON_Param (Page 688)

Connection parameters with structure according to TCON_IP_v4 (Page 691)

Connection parameters with structure according to TCON_IP_RFC (Page 692)

Starting connection parameter assignment

The connection configuration for Open User Communication is enabled as soon as a TCON, TSEND_C or TRCV_C instruction for communication is selected in a program block.

Requirement

- Your project must contain at least one S7-CPU.
- The program editor is open.
- A network is available.

Procedure

To insert the extended instructions for Open User Communication, proceed as follows:

1. Open the task card, pane and folder "Instructions > Communication > Open User Communication".
2. Drag one of the following instructions to a network:
   - TSEND_C
   - TRCV_C
   - TCON

   The "Call options" dialog opens.
3. Edit the properties of the instance DB in the "Call properties" dialog. You have the following options:
   - Change the default name.
   - Select the "Manual" check box to assign your own number.
   - You can also execute the DB as a multi-instance for function blocks.
4. Click "OK" to complete your entry.

Result

A corresponding instance DB is created at a single instance for the inserted instruction TSEND_C, TRCV_C or TCON. In the case of a multi-instance, the instance DB of the function block is used.

With TSEND_C, TRCV_C or TCON selected, you will see the "Configuration" tab under "Properties" in the Inspector window. The "Connection parameters" group in area navigation contains the connection parameter assignment that you can now make.

See also

Inserting FBD elements using the "Instructions" task card (Page 2623)

Inserting LAD elements using the "Instructions" task card (Page 2581)

Creating and assigning parameters to connections (Page 682)

Principles for single instances and multi-instances (Page 2182)

Creating and assigning parameters to connections

In the connection configuration for Open User Communication, you can create and configure connections of the TCP, UDP or ISO-on-TCP type.

Requirement

A CPU exists with a TCON, TSEND_C or TRCV_C communication instruction.

Procedure

To create a connection for Open User Communication, follow these steps:

1. Select a TCON, TSEND_C or TRCV_C block of Open User Communication in the program editor.
2. Open the "Properties > Configuration" tab in the inspector window.
3. Select the "Connection parameters" group. Until you select a connection partner, only the empty drop-down list for the partner end point is enabled. All other input options are disabled.
   The connection parameters already known are displayed:
   - Name of the local end point
   - Interface of the local end point
   - IP address of the local end point
4. In the drop-down list box of the partner end point, select a connection partner. You can select an unspecified device or a CPU in the project as the communication partner. Certain connection parameters are then entered automatically.
   The following parameters are set:
   - Name of the partner end point
   - Interface of the partner end point
   - IP address of the partner end point

   If the connection partners are networked, the name of the subnet is displayed.
5. With the S7-1500, in the "Configuration type" drop-down list, you choose between using program blocks or configured connections.

6. Select an existing connection description DB in the "Connection data" drop-down list or for configured connections select an existing connection under "Connection name". You can also create a new connection description DB or a new configured connection. Later, you can still select other connection description DBs or configured connections or change the names of the connection description DBs in order to create new data blocks:
   - You can also see the selected data block at the interconnection of the CONNECT input parameter of the selected TCON, TSEND_C or TRCV_C instruction.
   - If you have already specified a connection description DB for the connection partner using the CONNECT parameter of the TCON, TSEND_C or TRCV_C instruction, you can either use this DB or create a new DB.
   - If you edit the name of the displayed data block in the drop-down list, a new data block with the changed name but with the same structure and content is generated and used for the connection.
   - Changed names of a data block must be unique in the context of the communication partner.
   - A connection description DB must have the structure TCON_Param, TCON_IP_v4 or TCON_IP_RFC, depending on CPU type and connection.
   - A data block cannot be selected for an unspecified partner.

   Additional values are determined and entered after the selection or creation of the connection description DB or configured connection.
   The following is valid for specified connection partners:
   - ISO-on-TCP connection type
   - Connection ID with default of 1
   - Active connection establishment by local partner
   - TSAP ID
     for S7-1200/1500: E0.01.49.53.4F.6F.6E.54.43.50.2D.31
     for S7-300/400: E0.02.49.53.4F.6F.6E.54.43.50.2D.31

   The following is valid for unspecified connection partners:
   - TCP connection type
   - Partner port 2000

   The following applies for a configured connection with a specified connection partner:
   - TCP connection type
   - Connection ID with default of 257
   - Active connection establishment by local partner
   - Partner port 2000

   The following applies for a configured connection with an unspecified connection partner:
   - TCP connection type
   - Local port 2000

7. Enter a connection ID as needed for the connection partner. No connection ID can be assigned to an unspecified partner.

   Note

   You must enter a unique value for the connection ID at a known connection partner. The uniqueness of the connection ID is not be checked by the connection parameter settings and there is no default value entered for the connection ID when you create a new connection.

8. Select the desired connection type in the relevant drop-down list. Default values are set for the address details depending on the connection type. You can choose between the following:
   - TCP
   - ISO-on-TCP
   - UDP

   For configured connections with S7-1500, ISO applies in addition.
9. You can edit the input boxes in the address details. Depending on the selected protocol, you can edit the ports (for TCP and UDP) or the TSAPs (for ISO-on-TCP and ISO).
10. Use the "Active connection establishment" check box to set the connection establishment characteristics for TCP, ISO and ISO-on-TCP. You can decide which communication partner establishes the connection actively.

Changed values are checked immediately for input errors by the connection configuration and entered in the data block for the connection description.

Note

Open User Communication between two communication partners can only work when the program section for the partner end point has been downloaded to the hardware. To achieve fully functional communication, make sure that you load not only the connection description of the local CPU on the device but also that of the partner CPU as well.

See also

Description of the connection parameters (Page 678)

Starting connection parameter assignment (Page 681)

TSAP structure (Page 696)

Assignment of port numbers (Page 693)

Connection parameters with structure according to TCON_Param (Page 688)

Connection parameters with structure according to TCON_IP_v4 (Page 691)

Connection parameters with structure according to TCON_IP_RFC (Page 692)

Deleting connections

Introduction

The data of a created connection for Open User Communication is stored in a connection description DB. You can delete the connection by deleting the data block containing the connection description.

Requirement

You have created an Open User Communication connection.

Procedure

To delete a connection, follow these steps:

1. Select a communication partner for Open User Communication in the project tree.
2. Open the "Program blocks > System blocks > Program resources" folder below the selected communication partner.
3. Select the "Delete" command from the shortcut menu of the data block with the connection parameter assignment.

Note

If you are not certain which block to delete, open the extended instruction TCON, TSEND_C or TRCV_C. You will find the name of the data block as the CONNECT input parameter or in the connection parameter assignment as the "Connection data" parameter.

If you only delete the instance DBs of the extended instructions TCON, TSEND_C or TRCV_C, the assigned connections are not deleted as well.

Note

If the connection DB is still being used by other blocks of the extended instructions, then the corresponding calls, their instance DBs, and, if present, the combination blocks TSEND_C and TRCV_C must also be deleted from the block folder, provided they are not used elsewhere.

This action prevents the program from being inconsistent.

Result

You have deleted the connection.

Note

Insert an extended instruction TCON, TSEND_C, or TRCV_C again in order to reference an existing connection description with the TCON_Param, TCON_IP_v4, or TCON_IP_RFC structure again via the "Connection data" parameter.

How protocols work

Principle of operation of connection-oriented protocols

Introduction

Connection-oriented protocols establish a logical connection to the communication partner before data transmission is started. After the data transmission is complete, they then terminate the connection, if necessary. Connection-oriented protocols are used especially when reliable data transmission is important. Several logical connections can exist over one physical line.

Open User Communication supports the following connection types:

- TCP
- ISO-on-TCP
- ISO (S7-1500 only)
- UDP

Both communication partners must support the same connection type for a connection. If a communication partner does not support a connection of the type ISO-on-TCP, for example, use the connection type TCP instead, if it is supported.

For communication partners that cannot be configured in the TIA Portal, such as third-party devices or PCs, enter "unspecified" for the partner end point during connection parameter assignment. The required connection type for unspecified devices is listed in the respective documentation.

Note

Connections with ISO

For S7-1500 CPUs, configured connections of the type ISO can be created using the TSEND_C and TRCV_C instructions. For additional information on these connection types, refer to the general connection descriptions.

Characteristics of TCP

TCP is a streaming protocol in which the length of the data stream is transmitted to the receiver so that it can receive the data stream as individual TCP segments. This means no information about the start and end of a message is transmitted during data transmission via a TCP connection. The receiver cannot determine by the received segments of the data stream where one message in the data stream ends and the next one begins. It is therefore recommended that the number bytes to be received (parameter LEN, instruction TRCV/TRCV_C) be assigned the same value as the number of bytes to be sent (parameter LEN, instruction TSEND/TSEND_C).

If the length of the sent data and the length of the expected data do not match, the following occurs:

- Length of data to be received (parameter LEN, instruction TRCV/TRCV_C) greater than length of data to be sent (parameter LEN, instruction TSEND/TSEND_C):
  TRCV/TRCV_C copies the received data to the specified receive area (parameter DATA) only after the assigned length is reached. When the assigned length is reached, data of the subsequent job are already being received. As a result, the receive area contains data from two different send jobs. If you do not know the exact length of the first message, you are unable to recognize the end of the first message and the start of the second message.
- Length of data to be received (parameter LEN, instruction TRCV/TRCV_C) less than length of data to be sent (parameter LEN, instruction TSEND/TSEND_C):
  TRCV/TRCV_C copies the number of bytes you specified in the LEN parameter to the receive data area (parameter DATA). Then, it sets the NDR status parameter to TRUE (job completed successfully) and assigns RCVD_LEN (amount of data actually received) the value of LEN. With each subsequent call, you receive a further block of the sent data.

A receive area with fixed data length can be specified in the TRCV/TRCV_C instructions with the protocol version of the Ad-hoc mode.

Characteristics of ISO-on-TCP

ISO-on-TCP is a message-oriented protocol which detects the end of the message at the receiver end and indicates the data that belongs to the message to the user. This does not depend on the specified reception length of the message. This means that information regarding the length and the end of a message is included during data transmission via an ISO-on-TCP connection.

If the length of the sent data and the length of the expected data do not match, the following occurs:

- Length of data to be received (parameter LEN, instruction TRCV/TRCV_C) greater than length of data to be sent (parameter LEN, instruction TSEND/TSEND_C):
  TRCV/TRCV_C copies all the sent data to the receive data area (parameter DATA). Then, it sets the NDR status parameter to TRUE (job completed successfully) and assigns RCVD_LEN (amount of data actually received) the length of the data sent.
- Length of data to be received (parameter LEN, instruction TRCV/TRCV_C) less than length of data to be sent (parameter LEN, instruction TSEND/TSEND_C):
  TRCV/TRCV_C does not copy any data to the receive data area (parameter DATA), but instead supplies the following error information: ERROR=1, STATUS=W#16#8088 (destination buffer too small).

Characteristics of UDP

UDP is a message-oriented protocol which detects the end of the message at the receiver end and indicates the data that belongs to the message to the user. This does not depend on the specified reception length of the message. This means that information on the length and the end of a message is included during data transmission via a UDP connection.

If the length of the sent data and the length of the expected data do not match, the following occurs:

- Length of data to be received (parameter LEN, instruction TRCV/TRCV_C) greater than length of data to be sent (parameter LEN, instruction TUSEND/TSEND_C): TURCV/TRCV_C copies all the sent data to the receive data area (DATA parameter). Then, it sets the NDR status parameter to TRUE (job completed successfully) and assigns RCVD_LEN (amount of data actually received) the length of the data sent.
- Length of data to be received (parameter LEN, instruction TRCV/TRCV_C) less than length of data to be sent (parameter LEN, instruction TUSEND/TSEND_C): TRCV/TRCV_C copies as much data to the receive data area (parameter DATA) as the LEN parameter requests. No further error message is generated. In this case, the user has to call a T_URCV again in order to receive the remaining bytes.

See also

Basics of Open User Communication (Page 675)

TSEND_C: Send data via Ethernet (Page 5050)

TRCV_C: Receive data via Ethernet (Page 5061)

TSEND: Send data via communication connection (Page 5096)

TRCV: Receive data via communication connection (Page 5099)

TUSEND: Send data via Ethernet (UDP) (Page 5109)

TURCV: Receive data via Ethernet (UDP) (Page 5112)

Connection parameters with structure according to TCON_Param

Data block for connection description

A connection description DB with a structure according to TCON_Param is used for some S7-1200 CPUs when it comes to the assignment of parameters for TCP, UDP and ISO-on-TCP communication connections. The fixed data structure of the TCON_Param contains all the parameters that are needed to establish the connection. The connection description DB is automatically created for a new connection by the connection configuration for Open User Communication when the TSEND_C, TRCV_C or TCON instruction is used.

The CONNECT connection parameter of the instance DBs for TSEND_C, TRCV_C or TCON contains a reference to the data block used.

Structure of the connection description according to TCON_Param

Byte	Parameter	Data type	Start value	Description
0 … 1	block_length	UINT	64	Length: 64 bytes (fixed)
2 … 3	id	CONN_OUC	1	Reference to this connection (value range: 1 to 4095). You must specify the value of this parameter for the TSEND_C, TRCV_C or TCON instruction under ID.
4	connection_type	USINT	17	Connection type: • 17: TCP • 18: ISO-on-TCP • 19: UDP
5	active_est	BOOL	TRUE	Identifier for the type of connection establishment: FALSE always applies to UDP, since data can be sent and received via local ID. The following is valid for TCP and ISO-on-TCP: • FALSE: Passive connection establishment • TRUE: Active connection establishment
6	local_device_id	USINT	1	ID for the local PN/IE interface.
7	local_tsap_id_len	USINT	0	Length of parameter local_tsap_id used, in bytes; possible values: • 0 or 2, if connection type = 17 (TCP) Only the value 0 is permissible for the active side. • 2 to 16, if connection type = 18 (ISO-on-TCP) • 2, if connection type = 19 (UDP)
8	rem_subnet_id_len	USINT	0	This parameter is not used.
9	rem_staddr_len	USINT	4	Length of address of partner end point, in bytes: • 0: unspecified, in other words, parameter rem_staddr is irrelevant. • 4: valid IP address in the parameter rem_staddr (TCP and ISO-on-TCP only)
10	rem_tsap_id_len	USINT	2	Length of parameter rem_tsap_id used, in bytes; possible values: • 0 or 2, if connection type = 17 (TCP) Only the value 0 is permissible for the passive side. • 2 to 16, if connection type = 18 (ISO-on-TCP) • 0, if connection type = 19 (UDP)
11	next_staddr_len	USINT	0	This parameter is not used.

Byte	Parameter	Data type	Start value	Description
12 … 27	local_tsap_id	ARRAY [1..16] of BYTE	-	Local address component of connection: • TCP and UDP: local port no. (possible values: 1...49151; recommended values: 2000...5000); local_tsap_id[1] = high byte of port no. in hexadecimal notation; local_tsap_id[2] = low byte of port no. in hexadecimal notation; local_tsap_id[3-16] = irrelevant • ISO-on-TCP: local TSAP-ID: local_tsap_id[1] = B#16#E0; local_tsap_id[2] = rack and slot of local end points (bits 0 to 4: Slot number, bits 5 to 7: rack number); local_tsap_id[3-16] = TSAP extension, optional Note: Make sure that every value of local_tsap_id is unique within the CPU.
28 … 33	rem_subnet_id	ARRAY [1..6] of USINT	-	This parameter is not used.
34 … 39	rem_staddr	ARRAY [1..6] of USINT	-	TCP and ISO-on-TCP only: IP address of the partner end point, for example, for 192.168.002.003: • rem_staddr[1] = 192 • rem_staddr[2] = 168 • rem_staddr[3] = 002 • rem_staddr[4] = 003 • rem_staddr[5-6]= irrelevant
40 … 55	rem_tsap_id	ARRAY [1..16] of BYTE	-	Partner address component of connection • TCP: partner port no. (possible values: 1...49151; recommended values: 2000...5000); rem_tsap_id[1] = high byte of port no. in hexadecimal notation; rem_tsap_id[2] = low byte of port no. in hexadecimal notation; rem_tsap_id[3-16] = irrelevant • ISO-on-TCP: partner TSAP-ID: rem_tsap_id[1] = B#16#E0; rem_tsap_id[2] = rack and slot of partner end point (bits 0 to 4: Slot number, bits 5 to 7: rack number); rem_tsap_id[3-16] = TSAP extension, optional • UDP: This parameter is not used.
56 … 61	next_staddr	ARRAY [1..6] of BYTE	-	This parameter is not used.
62 … 63	spare	WORD	W#16#0000	Reserved.

Note

TCON_Param for S7-1500 CPU

The connection description DB with the structure according to TCON_Param is also supported by S7-1500 CPUs for migration reasons. However, we recommend that you use the new structures TCON_IP_v4 and TCON_IP_RFC.

See also

Principle of operation of connection-oriented protocols (Page 686)

Description of the connection parameters (Page 678)

Ability to read back connection description parameters (Page 695)

Basic principles for programming of data blocks (Page 2825)

Overview of connection configuration (Page 676)

TSAP structure (Page 696)

Assignment of port numbers (Page 693)

Connection parameters with structure according to TCON_IP_v4

Data block for connection description

A connection description DB with a structure according to TCON_IP_v4 is used for CPUs of S7-1200 V4.0 and higher and S7-1500 to assign parameters for TCP and UDP communication connections. The fixed data structure of the TCON_IP_v4 contains all parameters that are required to establish the connection. The connection description DB is automatically created for a new connection by the connection configuration for Open User Communication when the TSEND_C, TRCV_C or TCON instruction is used.

The CONNECT connection parameter of the instance DBs for TSEND_C, TRCV_C or TCON contains a reference to the data block used.

Structure of the connection description according to TCON_IP_v4

Byte	Parameter	Data type	Start value	Description
0 … 1	interface_id	HW_ANY	64	Hardware identifier of the local interface (value range: 0 to 65535).
2 … 3	id	CONN_OUC	1	Reference to this connection (value range: 1 to 4095). You must specify the value of this parameter for the TSEND_C, TRCV_C or TCON instruction under ID.

Byte	Parameter	Data type	Start value	Description
4	connection_type	BYTE	11	Connection type: • 11: TCP • 13: UDP
5	active_established	BOOL	TRUE	Identifier for the type of connection establishment: • FALSE: Passive connection establishment • TRUE: Active connection establishment
6 … 9	remote_address	ARRAY [1..4] of BYTE	-	IP address of the partner end point, for example, for 192.168.0.1: • addr[1] = 192 • addr[2] = 168 • addr[3] = 0 • addr[4] = 1
10 … 11	remote_port	UINT	2000	Port address of the remote connection partner (value range: 1 to 49151).
12 … 13	local_port	UINT	2000	Port address of the local connection partner (value range: 1 to 49151).

See also

Principle of operation of connection-oriented protocols (Page 686)

Description of the connection parameters (Page 678)

Ability to read back connection description parameters (Page 695)

Basic principles for programming of data blocks (Page 2825)

Overview of connection configuration (Page 676)

Assignment of port numbers (Page 693)

Connection parameters with structure according to TCON_IP_RFC

Data block for connection description

A connection description DB with a structure according to TCON_IP_RFC is used for CPUs of S7-1200 V4.0 and higher and S7-1500 to assign parameters to ISO-on-TCP communication connections. The fixed data structure of the TCON_IP_RFC contains all parameters that are required to establish the connection. The connection description DB is automatically created for a new connection by the connection configuration for Open User Communication when the TSEND_C, TRCV_C or TCON instruction is used.

The CONNECT connection parameter of the instance DBs for TSEND_C, TRCV_C or TCON contains a reference to the data block used.

Structure of the connection description according to TCON_IP_RFC

Byte	Parameter	Data type	Start value	Description
0 … 1	interface_id	HW_ANY	64	Hardware identifier of the local interface (value range: 0 to 65535).
2 … 3	id	CONN_OUC	1	Reference to this connection (value range: 1 to 4095). You must specify the value of this parameter for the TSEND_C, TRCV_C or TCON instruction under ID.
4	connection_type	BYTE	12	Connection type: ISO-on-TCP
5	active_established	BOOL	TRUE	Identifier for the type of connection establishment: • FALSE: Passive connection establishment • TRUE: Active connection establishment
8 … 11	remote_address	ARRAY [1..4] of BYTE	-	IP address of the partner end point, for example, for 192.168.0.1: • addr[1] = 192 • addr[2] = 168 • addr[3] = 0 • addr[4] = 1
12 … 45	remote_tselector	TSelector	-	TSelector of the remote connection partner: • TSelLength = Value range 0 to 32 as UINT • TSel[1-32] = Value range each 0 to 255 in bytes
46 … 79	local_tselector	TSelector	-	TSelector of the local connection partner: • TSelLength = Value range 0 to 32 as UINT • TSel[1-32] = Value range each 0 to 255 in bytes

See also

Principle of operation of connection-oriented protocols (Page 686)
Description of the connection parameters (Page 678)
Ability to read back connection description parameters (Page 695)
Basic principles for programming of data blocks (Page 2825)
Overview of connection configuration (Page 676)
TSAP structure (Page 696)

Assignment of port numbers

Introduction

When an Open User Communication is created, the value 2000 is automatically assigned as the port number.

Permissible values for port numbers are 1 to 49151. You can assign any port number within this range. However, because some ports may already be used depending on the system, port numbers within the range from 2000 to 5000 are recommended.

Note

Port numbers must be unique. The connection configuration or a corresponding block call is rejected with an error if the port numbers are assigned twice.

Overview of port numbers

The following table summarizes the system reactions to various port numbers.

Port no.	Description	System reaction
2000 … 5000	Recommended range	No warning, no error message on entry Port number is permitted and accepted
1 … 1999, 5001 … 49151	Can be used, but is outside the recommended range	Warning message on entry Port number is permitted and accepted
0, 20, 21, 25, 80, 102, 135, 161, 34962 … 34964	Can be used conditionally*	
53, 80, 102, 135, 161, 162, 443, 520, 9001, 34962 … 34964	Can be used conditionally**	

* Ports defined for specific functions:
0: ANY - Port number is automatically assigned by S7-1500 CPU as value (>49151)
20: FTP data transmission
21: FTP control
25: TMAIL_C (Simple Mail transfer protocol)
80: Web server
102: ISO-on-TCP (RFC1006)
135: DCE Endpoint Mapper for PROFINET
161: SNMP (Simple Network Management Protocol)
34962 ... 34964: PROFINET

Note

The user usually specifies the value 0 for the local port on the active connection end point for UDP/TCP. In this case, the CPU operating system selects the next available port above 49151. The partner port usually has the default 0 with the passive connection end point. The corresponding parameter is disabled in the connection configuration.

** These ports are disabled depending on the function scope of the CPU in use. The documentation of the respective CPUs provides the assignment of these ports.

See also

Description of the connection parameters (Page 678)

Creating and assigning parameters to connections (Page 682)

Ability to read back connection description parameters

Changing parameter values in the connection description

The connection description for exactly one connection of the Open User Communication is entered from the connection configuration in the connection description DB.

You can change the parameter values of the connection description DB outside of the connection configuration in the user program. Connection description DBs containing values you changed subsequently can be read back from the connection configuration. Under "Properties > Configuration > Connection parameters", the inspector window displays only the connection parameters stored in the connection description DB.

Note

You can only change the values in the running user program if the instructions TCON, TSEND_C or TRCV_C are not being processed and the referenced connection is not established.

The connection configuration does not support nested entries of connection descriptions in DB types that can only be found via offset referencing (for example, Global-DB).

The structure of the connection description cannot be changed.

Ability to read back individual connection parameters

For the "Address" parameter of the communication partner in a TCP or ISO-on-TCP connection, its IP address is displayed from the "rem_staddr" parameter of the connection description.

The following values can also be reloaded from the connection description:

- Connection type
- Local connection ID
- Active/passive connection establishment (only with UDP)
- Local TSAP (ISO-on-TCP only)
- Partner TSAP (ISO-on-TCP only)
- Local port (only with TCP and UDP)
- Partner port (only with TCP)

The values of the connection ID parameters of the communication partner, the connection data, as well as the connection establishment, are not included in the connection description in the local connection description DB. Consequently, these parameters cannot be displayed when the connection configuration is reopened. The connection establishment of the partner results from the local connection establishment and is therefore also displayed.

A new communication partner can be selected at any time in the "Partners" drop-down list box.

When a CPU recognized in the project is selected as a specified communication partner, the entry options for the connection ID and the connection data are shown again.

See also

Connection parameters with structure according to TCON_Param (Page 688)

Description of the connection parameters (Page 678)

TSAP structure

Introduction

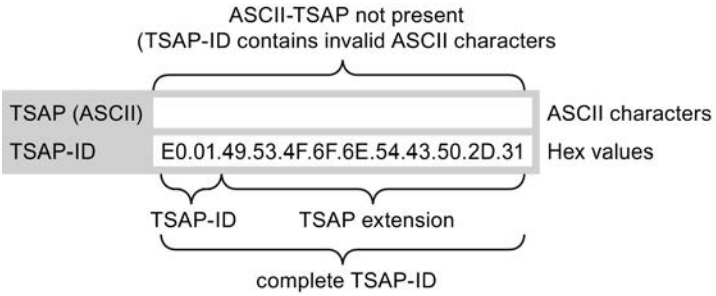
For an ISO-on-TCP connection, Transport Service Access Points (TSAPs) must be assigned for both communication partners. TSAP IDs are assigned automatically after an ISO-on-TCP connection is created. To ensure the uniqueness of TSAP IDs within a device, you can change the preassigned TSAPs in the connection parameter assignment.

Structure of TSAPs

You must comply with certain rules when assigning TSAPs. A TSAP must contain a certain number of bytes, which are able to be displayed and entered as hexadecimal values (TSAP-ID) or as ASCII characters (ASCII-TSAP):

Entries or changes of the TSAP-ID or the ASCII-TSAP in the corresponding entry fields always take effect in the other display format as well.

If a TSAP contains no valid ASCII characters, the TSAP is displayed only as TSAP-ID and not as ASCII-TSAP. This is the case after a connection is created. The first two hex characters as TSAP-ID identify the communication type and the rack/slot. Because these characters are not valid ASCII characters for a CPU, the ASCII-TSAP is not displayed in this case.



In addition to the rules for length and structure of TSAPs, you must also ensure the uniqueness of the TSAP-ID. The assigned TSAPs are not automatically unique.

Length and content of TSAPs

A TSAP is structured as follows:

- TSAP-ID with TSAP extension
  Length = 2 to 16 bytes
  x_tsap_id[0] = 0xE0 (Open User Communication)
  x_tsap_id[1] (bits 0 to 4) = slot number of CPU
  x_tsap_id[1] (bits 5 to 7) = rack number of CPU
  x_tsap_id[2...15] = any characters (TSAP extension, optional)
  (x = loc (local) or x = rem (partner))
- TSAP-ID as ASCII-TSAP
  Length = 3 to 16 bytes
  x_tsap_id[0 to 2] = 3 ASCII characters (0x20 to 0x7E)
  x_tsap_id[3...15] = any characters (optional)
  (x = loc (local) or x = rem (partner))

The following table shows the schematic structure of a TSAP-ID:

TSAP-ID	tsap_id_len	tsap_id[0]	tsap_id[1]	tsap_id[2..15]	tsap_id[3..15]
...with extension	2...16 bytes	0xE0	0x01 or 0x02 or 0x00*	Extension (optional)	Extension (optional)
...as ASCII-TSAP	3...16 bytes	0x20...0x7E	0x20...0x7E	0x20...0x7	Any (optional)

*An S7-1200/1500 CPU is normally inserted on rack 0 and slot 1, and an S7-300/400 CPU on rack 0 and slot 2. For this reason, hex value 01 or 02 is valid for the second position of the TSAP-ID with extension. If the connection partner is an unspecified CPU, for example, a third-party device, the hex value 00 is also permissible for the slot address.

Note

For unspecified communication partners, the local TSAP-ID and the partner TSAP-ID can have a length of 0 to 16 bytes, in which all hex values from 00 to FF are permitted.

ASCII code table for entry of ASCII TSAPs

For entry of an ASCII-TSAP in the connection parameter assignment, only hexadecimal values from 20 to 7E are permitted:

Code	..0	..1	..2	..3	..4	..5	..6	..7	..8	..9	..A	..B	..C	..D	..E	..F
2..		!	"	#	$	%	&	’	(	)	*	+	,	-	.	/
3..	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4..	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5..	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6..	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7..	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

See also

Examples of TSAP assignment (Page 698)

Description of the connection parameters (Page 678)

Creating and assigning parameters to connections (Page 682)

Examples of TSAP assignment

The following examples show the processing of the TSAPs for CPUs of the S7-1200/1500 (CPU on slot 1) under various points of view:

- Example 1: Creating a new connection for PLC-PLC communication
- Example 2: Entry of a local ASCII-TSAP
- Example 3: Entry of a TSAP extension in the TSAP-ID
- Example 4: Incorrect editing of the TSAP-ID
- Example 5: Entry of an ASCII-TSAP via the "TSAP-ID" entry field

Example 1: Creating a new connection for PLC-PLC communication

Once you have created a new connection with two PLCs for the Open User Communication, the TSAP extension "ISOonTCP-1" is assigned automatically.

This TSAP extension produces the TSAP-ID E0.01.49.53.4F.6F.6E.54.43.50.2D.31, which is entered automatically in the connection description DB and in the entry fields of the local and the partner TSAP. The entry fields of the ASCII-TSAPs remain empty:

	Local TSAP	Partner TSAP
TSAP (ASCII)		
TSAP-ID	E0.01.49.53.4F.6F.6E.54.43.50.2D.31	E0.01.49.53.4F.6F.6E.54.43.50.2D.31

You can change the values in the entry fields of the TSAP-ID and the ASCII-TSAP at any time.

The entry field of the TSAP-ID shows the complete TSAP stored in the data block of the connection description. The TSAP-ID with TSAP extension, which is limited to 16 characters, is not displayed in the "TSAP (ASCII)" entry field because the character E0 does not represent a valid character for the ASCII-TSAP.

If the displayed TSAP-ID is a valid ASCII-TSAP, it is displayed in the "TSAP (ASCII)" entry field.

Changes in the entry fields for TSAP-ID and ASCII-TSAP affect the other field.

Example 2: Entry of a local ASCII-TSAP

If you have created a new connection and assigned an ASCII value for the local TSAP in the "TSAP (ASCII)" entry field, for example, "ISOonTCP-1", the resulting TSAP-ID is created automatically.

When you exit the "TSAP (ASCII)" entry field, the number of ASCII characters is checked automatically for compliance with the limit (3 to 16 characters) and the resulting TSAP-ID is entered into the corresponding entry field:

	Local TSAP	Partner TSAP
TSAP (ASCII)	ISOonTCP-1	
TSAP-ID	49.53.4F.6F.6E.54.43.50.2D.31	E0.01.49.53.4F.6F.6E.54.43.50.2D.31

Example 3: Entry of a TSAP extension in the TSAP-ID

If, following creation of a connection and entry of an ASCII-TSAP (see examples 1 and 2) in the entry field of the local TSAP-ID, you add the prefix "E0.01" to the TSAP value, the ASCII-TSAP will no longer be displayed when the entry field is exited.

	Local TSAP	Partner TSAP
TSAP (ASCII)		
TSAP-ID	E0.01.49.53.4F.6F.6E.54.43.50.2D.31	E0.01.49.53.4F.6F.6E.54.43.50.2D.31

Once you have exited the entry field of the TSAP-ID, a check is performed automatically to determine whether the first character of the TSAP-ID is a valid ASCII character. Since the character "E0" now present in the TSAP-ID is not a valid character for the ASCII-TSAP, the "TSAP (ASCII)" entry field no longer displays an ASCII-TSAP.

If a valid ASCII character is used, the check for compliance with the length specification of 2 to 16 characters follows.

Example 4: Incorrect editing of the TSAP-ID

If you remove the hex value "E0" from a TSAP-ID beginning with "E0.01", the TSAP-ID now begins with "01" and therefore no longer complies with the rules and is invalid:

	Local TSAP	Partner TSAP
TSAP (ASCII)		
TSAP-ID	01.49.53.4F.6F.6E.54.43.50.2D.31	E0.01.49.53.4F.6F.6E.54.43.50.2D.31

After the entry field is exited, a message is output because the TSAP-ID is neither a valid ASCII-TSAP (this would have to have a hex value in the range from 20 to 7E as the first value) or a valid TSAP-ID (this would have to have the identifier "E0" as the first value).

Example 5: Entry of an ASCII-TSAP via the "TSAP-ID" entry field

If you remove the value "01" in addition to the value "E0" from the incorrect TSAP-ID in example 4, the TSAP-ID begins with the hex value 49. This value is within the permissible range for ASCII-TSAPs:

	Local TSAP	Partner TSAP
TSAP (ASCII)		
TSAP-ID	49.53.4F.6F.6E.54.43.50.2D.31	E0.01.49.53.4F.6F.6E.54.43.50.2D.31

When you exit the entry field, the TSAP-ID is recognized as a valid ASCII-TSAP and the resulting ASCII-TSAP "ISOonTCP-1" is written to the "TSAP (ASCII)" entry field.

See also

TSAP structure (Page 696)

Description of the connection parameters (Page 678)

Communication via PUT and GET instructions

Basic information on communication via the PUT/GET instruction

Basic information on PUT/GET instructions

Use PUT and GET instructions to exchange data between two CPUs via an S7 connection.

The GET instruction is used to read data from a partner CPU. The PUT instruction is used to control the writing of tags by the communication partner via the user program. Apart from the PUT and GET instructions, no additional communication functions are provided for reading and writing tags.

To simplify the use of the two instructions, specify all required parameters for the connection and all block parameters in the Inspector window of the program editor.

Requirement

To be able to use the PUT and GET instructions, the following requirements must be satisfied:

- At least one S7-1200/1500 CPU or S7-300/400 CPU must be created in the project. Firmware 2.0 or higher must be installed on an S7-1200 CPU. If you have not yet created a second CPU in the project, you can initially establish the connection to an unspecified partner.
- An S7 connection must exist between the two CPUs. If you have not yet established a connection between two CPUs, a connection is automatically established during the configuration of the instructions.
- For both instructions, an instance data block is required in which all data used by the instruction is stored. The instance data block is created automatically as soon as you drag a PUT or GET instruction to a network in the program editor. For the correct execution of the program, it is essential that the instance data blocks are not changed; consequently, these data blocks are know-how protected. You only have read access to the instance data blocks.

See also

Overview of connection configuration (Page 701)

Assigning parameters to start request (Page 706)

PUT: Set parameters for write and send area (Page 707)

GET: Set parameters for read and memory area (Page 708)

Connection configuration

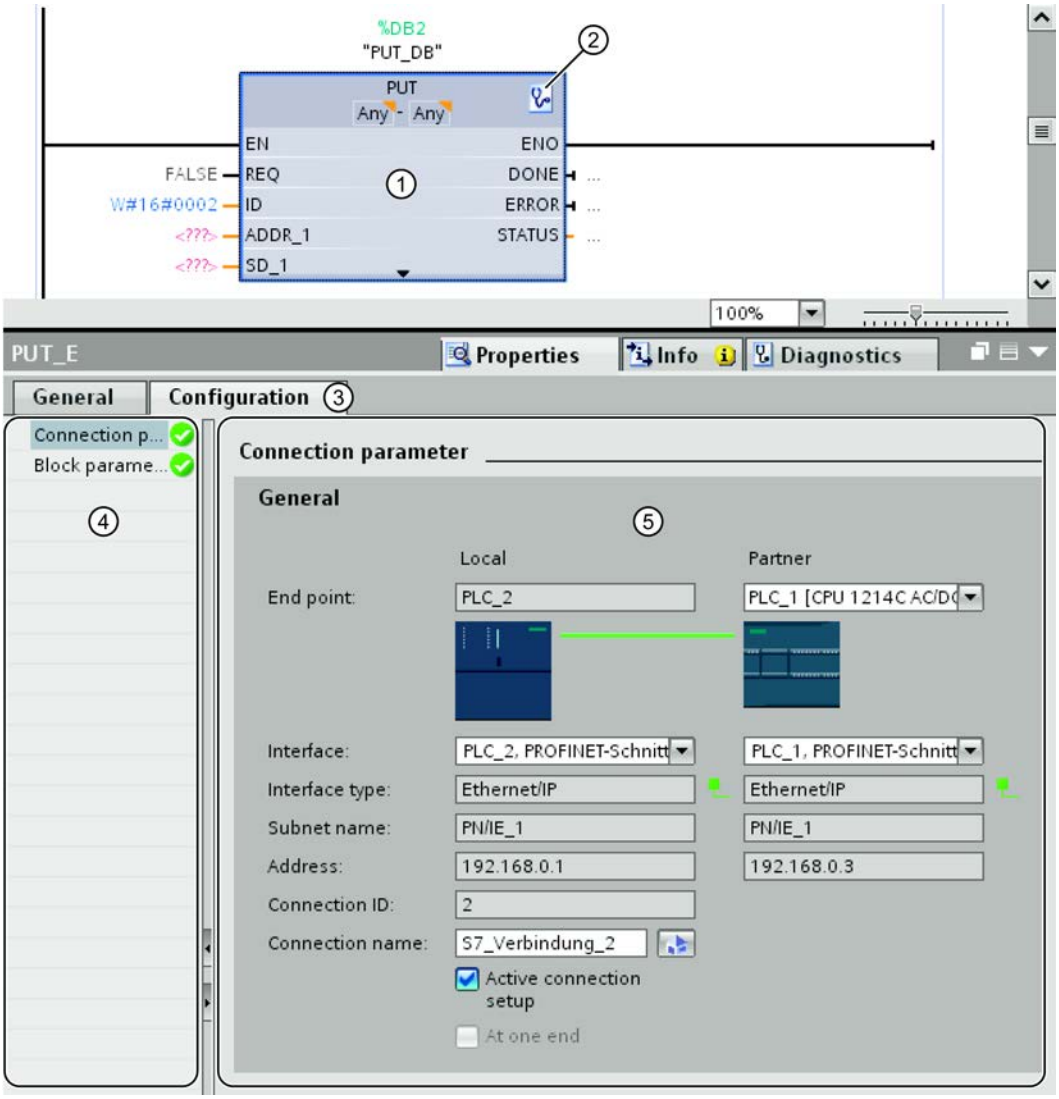
Overview of connection configuration

Introduction

The connection parameters for the PUT and GET instructions are assigned in the inspector window of the program editor. All parameters are saved in the corresponding instance data block.

Structure of the connection configuration

The connection configuration is made up of the following components:



① Communication instruction for PUT or GET
② Call of online and diagnostic functions
③ "Configuration" tab in the "Properties" tab
④ Area navigation of the "Configuration" tab
⑤ General properties of the connection parameters

Display of online and diagnostic functions

If you click the icon for starting the online and diagnostic functions, the associated CPU goes online automatically. The connection table in the network view is opened. In addition, the "Diagnostics" tab and the connection information are displayed in the inspector window.

Entering the connection parameters

Enter the desired connection parameters in the "Configuration" tab. The area navigation of the "Configuration" tab includes the "Connection parameters" group. This group contains the connection configuration. Here, you can enter the parameters for the connections using system functions. When all the required parameters are assigned, a check mark is set behind the "Connection parameters" group in the area navigation.

See also

Assigning parameters to start request (Page 706)

PUT: Set parameters for write and send area (Page 707)

GET: Set parameters for read and memory area (Page 708)

Description of the connection parameters

Overview

The following table shows the general connection parameters:

Parameter	Description
End point	The names of the local end point and the partner end point are shown. • Local end point The local end point is the CPU in which the PUT or GET instruction is programmed. • Partner end point The partner end point is selected from the drop-down list. The drop-down list shows all available possible connection partners including unspecified connection partners for devices whose data is unknown in the project. As long as no connection partner is set, all other parameters in the mask are disabled.
Interface	The interface of the partner CPU is displayed. The partner interface is not displayed until a specified partner CPU has been selected.
Interface type	The type of interface via which communication is handled is displayed.
Subnet name	The subnet of the local end point is displayed, provided this exists. The partner subnet is displayed only after the partner end point has been selected. If at least one of the two connection partners is not connected with a subnet, the two connection partners are automatically connected with each other. The partner which is not connected to a network is hereby connected to the same subnet via which the other partner is already connected to a network. A connection of connection partners to different subnets is only possible with IP or S7 routing. The IP routing settings can be edited in the properties of the relevant Ethernet interfaces.
Address	The IP address of the local end point is displayed. The IP address of the partner is displayed only after the partner end point has been selected. If you have selected an unspecified connection partner, the input box is empty and has a red background. In this case, you will need to specify a valid IP address for the connection partner.
Connection ID	The currently set connection ID is displayed. You can change the connection ID in the connection table in the network view. You can also directly access the connection table while you are setting the connection parameters. To do this, click the "Create new connection" icon.

Parameter	Description
Connection name	The name of the connection which was automatically created when the PUT/GET instruction was inserted is displayed. You can change the name of the connection by entered a different name in the field. You can also create a new connection or edit existing connections by clicking the "Create new connection" icon.
Active connection establishment	Use the "Active connection establishment" option button to specify which partner starts the communication. When the connection is created, the local partner is initially specified by default for the establishment of the connection. If a device does not support active connection establishment, you have to activate active connection establishment on the other partner.
Configured at one end	If this check box is selected, the connection partner is the server for this connection. It cannot actively send or receive. This corresponds to the behavior of the PUT/GET instructions. In this case, other instructions are not possible. If the check box is not selected, other instructions can also be used for the communication.

Starting connection parameter assignment

You can assign the connection parameters for PUT and GET in the inspector window as soon as you have inserted and selected a PUT or GET instruction in a program block.

Procedure

To insert PUT/GET instructions, follow these steps:

1. Open the "Instructions" task card in the "Communication > S7 Communication" folder.
2. Drag a PUT or GET instruction to a network.
   The "Call options" dialog opens.
3. Optional: Edit the properties of the instance DB in the "Call properties" dialog. You have the following options:
   – Change the default name.
   – Select the "Manual" check box to assign your own number.
4. Click "OK".

Result

A corresponding instance data block is created for the inserted PUT or GET instruction. For S7-300 CPUs, a function block is also created in the program resources.

When PUT or GET instruction is selected, you will see the "Configuration" tab under "Properties" in the inspector window. The "Connection parameters" group in area navigation contains the connection parameter assignment that you can now make.

See also

Creating and assigning parameters to connections (Page 705)

Deleting connections (Page 706)

Creating and assigning parameters to connections

You can create S7 connections and assign the parameters for these in the connection parameter assignment of the PUT/GET instructions. Changed values are checked immediately by the connection parameter assignment for input errors.

Requirement

A CPU exists with a PUT or GET communication instruction.

Procedure

To configure an S7 connection using PUT/GET instructions, follow these steps:

1. In the program editor, select the call of the PUT or GET instruction.
2. Open the "Properties > Configuration" tab in the inspector window.
3. Select the "Connection parameters" group. Until you select a connection partner, only the empty drop-down list for the partner end point is enabled. All other input options are disabled.
   The connection parameters already known are displayed:
   - Name of the local end point
   - Interface of the local end point
   - IP address of the local end point
4. In the drop-down list box of the partner end point, select a connection partner. You can select an unspecified device or a CPU in the project as the communications partner.
   The following parameters are automatically entered as soon as you have selected the connection partner:
   - Interface of the partner end point
   - Interface of the partner end point. If several interfaces are available, you can change the interface as required.
   - Interface type of the partner end point
   - Subnet name of both end points
   - IP address of the partner end point
   - Name of the connection which is used for the communication. If no connection exists yet, it is automatically established.
5. If required, change the connection name in the "Connection name" input box. If you want to create a new connection or edit an existing connection, click on the "Create new connection" icon.

Note

The PUT and GET instructions between two communication partners can only run if both the hardware configuration and the program part for the partner end point have been loaded into the hardware. To achieve fully functional communication, make sure that you load not only the connection description of the local CPU on the device but also that of the partner CPU as well.

Deleting connections

A connection which was automatically created during the insertion of a PUT or GET instruction appears in the connection table of the network view like every standard connection. As a result, it can be deleted in the connection table.

Procedure

To delete a connection, follow these steps:

1. Open the connection table in the network view.
2. In the connection table, select the connection that you want to delete.
3. To do this, right-click the connection and select the "Delete" command from the shortcut menu.

Result

The connection is deleted. The PUT or GET instruction and the associated instance data blocks are retained and must be manually deleted if necessary.

To continue using the PUT or GET instruction, you must configure the connection again in the inspector window of the program editor, since all connection parameters were also deleted when the connection was deleted. In this case, specify a new communication partner and a suitable connection.

Block parameter assignment

Assigning parameters to start request

To start communication via the PUT or GET instruction, you have to specify an event which activates the instruction. This event is referred to as control parameter (REQ). The communication job is activated as soon as there is a positive edge at the control parameter REQ.

Please note that the control parameter REQ is assigned the default FALSE at first call.

Requirement

- The program editor is open.
- You have already inserted a PUT or GET instruction.
- A connection has been established between two communication partners.

Procedure

To define the REQ control parameter, follow these steps:

1. Select the PUT or GET instruction in the program editor.
2. Open the "Configuration" tab in the inspector window.

3. Select the "Block parameter assignment" entry in the area navigation.
4. In the "REQ" field, select a tag of the "BOOL" data type to initialize the execution of the instruction. Alternatively, you can also interconnect a previous instruction in the program editor.

See also

Data consistency (Page 5033)

PUT: Set parameters for write and send area (Page 707)

GET: Set parameters for read and memory area (Page 708)

PUT: Set parameters for write and send area

For communication via the PUT instruction, you must specify the memory area of the partner CPU to which the data should be written. In addition, you must specify the memory area of the local CPU from which the data is to be read.

Requirement

- The program editor is open.
- You have already inserted a PUT instruction.
- A connection has been established between two communication partners.

Procedure

To specify the read and the memory area for the instruction, follow these steps:

1. Select the PUT instruction in the program editor.
2. Open the "Configuration" tab in the inspector window.
3. Select the "Block parameter assignment" entry in the area navigation.
4. In the "In/Outputs > Write area (ADDR_1) > Start" field, select a "REMOTE" data type pointer to the area of the partner CPU which is to be written.
   Only absolute addressing is permitted.
   Example: P#DB10.DBX5.0 Byte 10
5. In the "Length" field, enter the length of the write area and select the data type of the memory area from the drop-down list.
6. In the "In/Outputs > Send area (SD_1) > Start" field, select a pointer to the area in the local CPU which contains the data to be sent.
7. In the Length field, enter the length of the memory area to be read and select the data type from the drop-down list.
   Only the data types BOOL (for a bit array, "0" must be used as address and an integer multiple of byte must be used as length), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, COUNTER, TIMER are permitted.
   If the VARIANT pointer accesses a DB, the DB must always be specified (for example: P#DB10.DBX5.0 Byte 10).

See also

GET: Set parameters for read and memory area (Page 708)

GET: Set parameters for read and memory area

For communication via the GET instruction, you must specify the memory area of the local CPU to which the data should be written. In addition, you must specify the memory area of the partner CPU from which the data is to be read.

Requirement

- The program editor is open.
- You have already inserted a GET instruction.
- A connection has been established between two communication partners.

Procedure

To specify the read and the memory area for the instruction, follow these steps:

1. Select the GET instruction in the program editor.
2. Open the "Configuration" tab in the inspector window.
3. Select the "Block parameter assignment" entry in the area navigation.
4. In the "In/Outputs > Read area (ADDR_1) > Start" field, select a "REMOTE" data type pointer to the area of the partner CPU which is to be read.
   Only absolute addressing is permitted.
   Example: P#DB10.DBX5.0 Byte 10
5. In the "Length" field, enter the length of the read area and select the data type of the memory area from the drop-down list.
6. In the "In/Outputs > Memory area (RD_1) > Start" field, select a pointer to the area in the local CPU in which the read data is to be stored.
7. In the Length field, enter the length of the memory area and select the data type from the drop-down list.
   Only the data types BOOL (for a bit array, "0" must be used as address and an integer multiple of byte must be used as length), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, COUNTER, TIMER are permitted.

See also

PUT: Set parameters for write and send area (Page 707)

8.1.3.3 Displaying and configuring topology

Overview of the topology view

Functions of the topology view

The topology view is one of three working areas of the hardware and network editor. You undertake the following tasks here:

- Displaying the Ethernet topology
  - Displaying all the PROFINET devices and passive Ethernet components of the project along with their ports
  - Displaying interconnections between the ports
  - Displaying corresponding logical networks
  - Display diagnostic information of all ports
- Configuring the Ethernet topology
  - Creating, modifying and deleting interconnections of the ports
  - Renaming stations, devices, interfaces or ports
  - Adding PROFINET devices and passive Ethernet components to the project from the hardware catalog
- Identifying and minimizing differences between the desired and actual topology
  - Running an offline/online comparison of Ethernet modules, ports and port interconnections
  - Adopting existing online topology information in the offline project

---

Note

Devices without valid IP address

Topology information (LLDP) cannot be read from a device without valid IP address.

To prevent devices from having an invalid IP address, specify in the settings of the TIA Portal for hardware configuration that devices without valid IP address are temporarily assigned an IP address.

Differences between network view and topology view

- The network view shows all the logical subnets of the project.
  The topology view shows all Ethernet components of the project. These include passive components such as switches, media converters and cables.

  Note

  Stations with non-Ethernet components are also displayed if the station has a least one Ethernet component.

- The position of a device in the network view and its position in the topology view are not dependent on each other; in other words, the same device generally appears at different locations in the two views.
- If you open the hardware catalog in the topology view, you only see devices with an Ethernet interface.

Structure of the topology view

The topology view (Page 482) essentially consists of a graphic area (called the graphic view below) and a table area (called the table view below).

Which functions are there in the graphic view and which functions are there in the table view?

- Displaying the Ethernet topology

Function	Graphic view	Table view
Displaying all the PROFINET devices and passive Ethernet components of the project along with their ports	yes	yes
Display interconnections between the ports (including type of medium)	yes	yes
Displaying corresponding logical networks	no	yes
Displaying properties of the cables between the ports	no	yes
Display diagnostic information of all ports	yes	yes

- Configuring the Ethernet topology

Function	Graphic view	Table view
Creating, modifying and deleting interconnections of the ports	• Create: yes • Modify: no • Delete: yes	• Create: yes • Modify: yes • Delete: yes
Renaming stations, devices, interfaces or ports	no	yes
Adding PROFINET devices and passive Ethernet components to the project from the hardware catalog	yes	no

- Identifying and minimizing differences between the desired and actual topology

Function	Graphic view	Table view
Running an offline/online comparison of Ethernet modules, ports and port interconnections	no	yes
Adopting existing online topology information in the offline project	no	yes

See also

Overview of settings for hardware configuration (Page 483)

Starting the topology view

Requirement

The device or network view is open in the hardware and network editor.

Procedure

To start the topology view of your project, follow these steps:

1. Click on the "Topology view" tab.

Or:

1. Open the network view of the hardware editor.
2. Select a PROFINET device or a PROFINET module.
3. Select the "Go to topology view" command in the shortcut menu.

Result

The graphic view of the topology view is started. If you opened the topology view using the shortcut menu, the selected component remains selected after the change of view.

Displaying topology

Displaying the graphic view of the configured topology

What is shown?

The graphic view of the configured topology shows the following:

- Configured PROFINET devices and passive Ethernet components along with their ports
- Configured stations with non-Ethernet components if there is at least one Ethernet component in the station
- Configured interconnections between the ports

Type of display

The way in which the graphic view of the topology view and the network view are displayed is very similar:

- Compared with the device view, components are shown in a simplified form.
- The interconnections between ports are shown as horizontal and vertical lines. These are dashed when an interconnection between a tool changer port and its possible partner ports is involved.

Displaying the table view of the configured topology

What is shown?

The table view of the configured topology shows exactly the same content as the graphic view except for the logical PROFINET subnets.

- All the configured PROFINET devices and passive Ethernet components along with their ports
- All the configured stations with non-Ethernet components if there is at least one Ethernet component in the station
- Configured interconnections between the ports
  For each port with the "Alternating partner port" property, there are as many completed rows as there are potential partner ports plus one empty row.

Type of display

As the name implies, the table view of the topology view consists of a table, the topology overview table. It is structured like the network overview table. It consists of the following columns:

- Device / port
  This is the most important column of the table. The entries in this column have a hierarchical structure with the PROFINET ports being the last element in the hierarchy. You can expand and collapse the hierarchical entries. For a CPU, for example, an entry consists of the following elements:
  - Station name
  - Device name
  - Name of the PROFINET interface
  - Names of the ports

  Note: All the other columns only have entries in the rows containing the port names.
- Type (as default, this column is not displayed)
  Shows what type of station, device or interface the table row relates to or whether it belongs to a port.
- Order no. (as default, this column is not displayed)
  Order no. of device

- Subnet (as default, this column is not displayed)
  Configured subnet to which the interface belongs
- Master / IO system (as default, this column is not displayed)
  Shows whether or not the interface belongs to a PROFIBUS DP master system or a PROFINET IO system.
- Device address (as default, this column is not displayed)
  Configured address of the interface in the subnet
- Partner station
  Name of the station that contains the partner port
- Partner device
  Name of the device that contains the partner port
- Partner interface
  Interface to which the partner port belongs
- Partner port
- Cable data
  Contains the cable length and the signal delay of the cable connecting the ports

Basic functions for tables

The topology overview table supports the following basic functions for editing a table:

- Displaying and hiding table columns
  Note: The columns that define the configuration cannot be hidden.
- Optimizing column width
- Displaying the meaning of a column, a row or cell using tooltips.

Displaying the diagnostics status of ports and cables in the graphic view

Requirements

The graphic view of the topology view is open.

Procedure

To determine the diagnostics status of the port, follow these steps:

1. Go online with the required component or components.

Result

The following icons are displayed:

- The corresponding diagnostics icon is displayed for each device.
- If there is an error in at least one lower-level component, the diagnostics icon "Error in lower-level component" is also displayed in the left-hand lower corner of the diagnostics icon.
- The corresponding diagnostics icon is displayed for each port.
- Every cable between two ports that are online has the color that matches its diagnostics status.

You will find the possible diagnostics icons for ports and the color coding of Ethernet cables in the description of hardware diagnostics. See: Displaying diagnostics status and comparison status using icons (Page 2081)

Showing the diagnostics status of hardware components in the table view

Requirement

The table view of the topology view is open.

Procedure

To obtain the diagnostics status of hardware components of the topology overview table, follow these steps:

1. Go online with the required components.

Result

The following icons are displayed at the left-hand edge of the topology overview table in each row that belongs to the component involved:

- The diagnostics icon belonging to the hardware component is displayed.
- If the hardware component has lower-level components and if there is an error in at least one of the lower-level components, the diagnostics icon "Error in lower-level component" is also displayed in the left-hand lower corner of the diagnostics icon of the hardware component.

For the possible diagnostics icons for hardware components, refer to the description of hardware diagnostics. See: Displaying diagnostics status and comparison status using icons (Page 2081)

Note

The display of the diagnostics status of hardware components in the topology overview table and the network overview table is identical.

Running an offline/online comparison and displaying the results

Requirement

The topology view is open. There may be an online connection to one or more devices, but this is not actually necessary.

Procedure

To find the differences between the configured and the actual topology, follow these steps:

1. Click the "Offline/online comparison" button in the toolbar of the topology overview.

Result

The "Partner station", "Partner interface" and "Cable data" columns in the topology overview table are removed.

Two additional groups of columns are added to the right-hand side of the table and these are initially empty:

- On the far right, columns for the topology to be identified online are added.
- Between the columns for the offline and the online topology, the "Status", "Action" and "Description" columns are added to show the result of the offline/online comparison.

  Note

  As default, the "Description" column is not displayed.

The following buttons are enabled in the toolbar of the table:

Button	Name	Meaning
	Update	The detection of the existing online topology is started again.
	Synchronize	• Adopting the port interconnections identified online in the project (Page 724) • Adopt the devices identified online in the project (Page 725)

After the actual topology has been identified, the added columns are filled. These steps are described in more detail in the following section.

Note

A difference between offline and online view is displayed for that port connected with the PG/PC which is only available online. This is because the PG/PC cannot be configured offline.

Columns for the topology identified online

The following columns are displayed:

- "Device / port"
- "Type" (as default, this column is not displayed)
- "Order no." (as default, this column is not displayed)
- "IP address" (as default, this column is not displayed)
- "Partner device"
- "Partner port"
- "Cable data"

Columns for the result of the offline/online comparison

The following columns are displayed:

- "Status"
  The result of the offline/online comparison is shown here in the form of diagnostics icons. The following icons are possible:

Diagnostics icon	Meaning
	Differing topology information in at least one lower-level component
	Identical topology information
	Topology information only available offline or device is disabled.
	Topology information only exists online
	Differing topology information
	If a device does not support topology functions, the "Status" column remains empty.

- "Action"
  The possible actions are shown here in the form of icons. The following icons are possible:

Icon	Meaning
=	No action possible
←	Adopt the interconnection found online

- "Description"
  This column describes the selected action in words.

Configuring topology

Interconnecting ports

Overview

Interconnecting ports in the topology view

In the topology view, you have the following options for interconnecting ports:

- in the graphic view (Page 718)
- in the graphic view of a tool changer (Page 720)
- in the table view (Page 719)
- in the table view of a tool changer (Page 720)
- by adopting port interconnections identified online (Page 724)

What effects does the interconnection of ports have on the network view?

Note

In the properties of a subnet in the network view, you can specify that this subnet is used when a port interconnection is created between two devices that are not networked.

When you create an interconnection between two ports, the following effects are possible in the network view:

- If the corresponding interfaces are not networked: If you have specified a default subnet, this is used. Otherwise a new subnet is created to connect the two interfaces.
- If one (and only one) of the two interfaces involved is networked: The non-networked interface is connected to the same subnet as the already networked interface.
- In all other cases: The corresponding interfaces are not connected to a logical subnet.

See also

Interconnecting ports (Page 1455)

Settings for interconnecting Ethernet devices (Page 536)

Interconnecting ports in the graphic view

Requirement

You are in the graphic view of the topology view.

Procedure – Creating a new interconnection between two ports

To interconnect a port of a device with a port of another device, follow these steps:

1. Place the mouse cursor on the port you want to interconnect.
2. Click with the left mouse button and hold it down.
3. Move the mouse pointer.
   The pointer now shows the networking symbol to indicate "Interconnect" mode. At the same time, the mouse pointer shows the lock symbol that will only disappear when the pointer is on a valid target.
4. Now drag the mouse cursor to the target port. You can do this while holding down or after releasing the mouse button.
5. Now release the left mouse button or press it again (depending on your previous action).

Result: A new port interconnection is created.

Note

Creating a ring for S7-300, S7-400, and S7-1500 CPUs

If you create a ring using port interconnections for S7-300, S7-400, or S7-1500 CPUs, an MRP domain is created automatically.

Procedure – Changing an existing port interconnection without deleting it first

To do this, follow these steps:

1. Place the mouse cursor on the port of an existing interconnection that is to receive a new partner port.
2. Drag the port to the new partner port.

Result: The existing port interconnection is deleted. The new port interconnection is created.

Alternative procedure:

1. Place the mouse cursor on a port without an interconnection that is to be connected with an already interconnected port.
2. Drag the port to the already interconnected port.

Result: The existing port interconnection is deleted. The new port interconnection is created.

Procedure – Interconnecting two interconnected ports with each other without first deleting the two existing port interconnections

To do this, follow these steps:

1. Place the mouse cursor on the interconnected port that is to receive a new partner port.
2. Drag the port to the new partner port that is also interconnected already.

Result: The two existing port interconnections are deleted. The new port interconnection is created.

Interconnecting ports in the table view

Which actions are possible with port interconnections in the table view?

The following actions are possible with port interconnections in the table view:

- Creating a new port interconnection
- Changing an existing port interconnection
- Deleting an existing port interconnection

Requirement

The row with the port whose interconnection you want to create, modify or delete is visible in the topology overview.

Procedure

To create the interconnection of a port for the first time, to modify it or delete it, follow these steps:

1. Move the mouse pointer to the "Partner port" column in the row of the source port.
2. Click the drop-down list there.
3. Select the required partner port (when creating or changing a port interconnection) or the "Not interconnected" entry (when deleting a port interconnection).

Result

The required action is performed. The new partner port (after creating or modifying a port interconnection) or the "Select port" entry (after deleting a port interconnection) is displayed in the "Partner port" column.

Interconnecting a port with more than one partner port in the graphic view

Requirement

- You have configured a port of a PROFINET device with the "Alternative partners" property and have specified its possible partner ports.
- The graphic view of the topology view is open.

Procedure

1. Interconnect this port (referred to hereafter as source port) with one of the partner ports you have specified (referred to hereafter as target port).
2. Interconnect the source port with an additional target port.
   You can do this in several ways:
   - Drag the mouse pointer from a partner port that is already interconnected to a target port.
   - Drag the mouse pointer from an interconnection that has already been created to a target port.
   - Drag the mouse pointer from a target port to a partner port that is already interconnected.
   - Drag the mouse pointer from a target port to an already created interconnection.
3. If necessary, repeat the step above one or more times.

Result

An interconnection is created between the source port and the alternative partner ports. This is indicated by a dashed line.

Interconnecting a port with more than one partner port in the table view

Which actions are possible with port interconnections to several partner ports in the table view?

When working with a tool changer, the following actions can be performed with port interconnections to multiple partner ports in the table view:

- Creating a new port interconnection
- Changing an existing port interconnection
- Deleting an existing port interconnection

Requirement

- You have configured a port of a PROFINET device with the "Alternative partners" property and have specified its possible partner ports.
- The row with the port whose interconnection you want to create, modify or delete is visible in the topology overview.

Procedure

To create the interconnection of a port to one or more partner ports for the first time, to modify it, or to delete it, follow these steps:

1. Move the mouse pointer to the "Partner port" column in the row of the source port.
2. Click the drop-down list there.
3. Select the required partner port (when creating or changing a port interconnection) or the "Not interconnected" entry (when deleting a port interconnection).

Result

The required action is performed:

- If you are creating an interconnection, a new row is inserted in the topology overview table. The new partner port is displayed there in the "Partner port" column.
- If you change an interconnection, the new partner port is displayed in the "Partner port" column.
- If you delete an interconnection, the row with the previous port interconnection is deleted.

Note

With a tool changer, there are normally several rows for a port with port interconnections to more than one partner port. The last row is always an empty row. The first row can be edited, all other rows are read-only.

Renaming stations, devices, interfaces or ports

Rename a station, a device, an interface or a port

Requirement

The table view of the configured topology is open.

Procedure

To rename a station, a device, an interface or a port, proceed as follows:

1. Click twice in the relevant field of the topology overview table (the second click starts the editing mode).
2. Enter the new name and then press the ENTER key (this closes editing mode).

Result

The object is renamed.

Offline/online comparison

Automatic assignment of devices by offline/online comparison

Overview

During the offline/online comparison, the configured topology is compared with the actual existing topology. Devices identified online are automatically assigned to configured devices as far as this is possible.

Start of availability detection

You start the availability detection the first time by clicking the "Compare offline/online" button in the toolbar of the topology overview.

You restart availability detection by clicking the "Update" button.

Note

The availability detection can take several seconds. During this time, no user input is possible.

Automatic assignment

A device identified online is automatically assigned to a configured device if the following properties of the two devices match up:

- Device name
- Order number
- Number of ports

The following section describes the situations that can occur and what action you can take:

- Identical port interconnections
  This is the ideal situation. No action is necessary here.

"Action" column	Meaning
	No action

- There are interconnections for the identified and configured device, there are however differences.
  The following actions are possible:
  – If it is possible to adopt the online configuration

"Action" column	Meaning
	Adopt online interconnection (Page 724)
	No action

  – If it is not possible to adopt the configuration

"Action" column	Meaning
	No action

- The interconnection only exists online.
  The following actions are possible:
  – If it is possible to adopt the online configuration

"Action" column	Meaning
	Adopt online interconnection (Page 724)
	No action

  – If it is not possible to adopt the configuration

"Action" column	Meaning
	No action

- The interconnection only exists in the configuration.
  The following actions are possible:

"Action" column	Meaning
	Adopt the online interconnection (Page 724), in other words, the interconnection in the configuration will be deleted
	No action

No automatic assignment

In the following situations, no automatic assignment is possible:

- No device can be identified online to match a configured device. In this case the corresponding columns in the "Online topology" area of the topology overview table are empty.
  In this case, you should add the already configured device to your system or delete the configured device from the configuration.
- A device identified online cannot be assigned to any configured device. In this case the corresponding columns in the "Offline topology" area of the topology overview table are empty.
  In this case, you can adopt the device identified online in the project (Page 725).

Adopting the port interconnections identified online in the project

Requirement

You have run an offline/online comparison in the topology view. The result of this is that at least one device identified online was automatically assigned to a configured device, but that there are differences relating to the interconnection.

Procedure

To adopt one more port interconnections identified online in the project manually, follow these steps:

1. Select the value "Adopt" in the "Action" column for a port of a configured device to which a device identified online was assigned.
2. Repeat the step if necessary for other ports of the same configured device.
3. Repeat the steps up to now if necessary for other configured devices to which devices identified online were assigned and for which there are differences relating to the interconnection.
4. Click the "Synchronize" button.

Result

The port interconnections identified online and the cable information for the corresponding devices are adopted in the project. Successful adoption is indicated by the diagnostics icon "Identical topology information" for each port.

Note

If other port interconnections are recognized for a device identified online and these differ from those that exist in the project, adopting these in the project means that the port interconnections that were previously in the project are replaced by those identified online. If no port interconnections are detected for a device identified online, adopting in the project means that all the port interconnections of this device are deleted in the project.

Adopt the devices identified online in the project

Requirements

You have run an offline/online comparison in the topology view. The result of this is that at least one device identified online could not be assigned to any configured device.

Procedure

To adopt one more devices identified online in the project manually, follow these steps:

1. For a configured device without an online partner, move the mouse pointer to the "Device/port" column of the online topology.
2. Select the device you want to assign to the configured device from the drop-down list of this box.
3. Repeat the previous steps if necessary for other configured devices without an online partner.

Result

The selected device that was identified online is moved up from the end of the table. Following this, it is in the row of the configured device to which you have just assigned it.

8.1.3.4 Industrial Ethernet Security

Configuring security

General

Supported devices

Supported devices

Security functions can be configured for the following products:

- SCALANCE S:
  - S602 V2/V3/V4
  - S612 V2/V3/V4
  - S613 V2
  - S623 V3/V4
  - S627-2M V4
- SOFTNET Security Client:
  - SOFTNET Security Client V4

- S7-CPs: CP 343-1 GX31 Advanced, CP 443-1 GX30 Advanced, CP 1543-1
- PC CP: CP 1628
- Mobile wireless router: SCALANCE M875

General terminology "security module"

In this section of the information system, the following products are grouped together under the term "security module": SCALANCE S602 / SCALANCE S612 / SCALANCE S613 / SCALANCE S623 / SCALANCE S627-2M, CP 343-1 GX31 Advanced, CP 443-1 GX30 Advanced, CP 1543-1, CP 1628, SCALANCE M875.

Use of the terms "interface" and "port"

In this documentation, the following terms are used for the ports of SCALANCE S modules:

- "External interface": The external port of the SCALANCE S602 / S612 / S623 or an external port of the SCALANCE S627-2M (marked red)
- "Internal interface": The internal port of the SCALANCE S602 / S612 / S623 or an internal port of the SCALANCE S627-2M (marked green)
- "DMZ interface": The DMZ port of the SCALANCE S623 / S627-2M (marked yellow)

The term "port" itself is used when the focus of interest is a special port of an interface.

General use of the term "CP x43-1 Adv."

In this documentation, the term "CP x43-1 Adv." includes the following products: CP 343-1 GX31 Advanced / CP 443-1 GX30 Advanced.

Structure of this section of the help system

Topics that are relevant to all security modules can be found in the "General" section. Information that is only relevant to certain module types can be found in the sections relating specifically to these modules.

Overview - Scope of performance and method of operation

General use of the term "STEP 7"

Configuration of security functions is supported as of STEP 7 V12. For this reason, in this section of the information system, the name "STEP 7" will be used for all versions of STEP 7 V12 or higher.

Scope of performance

You can use the following security functions in STEP 7:

- Configuration of the security modules
- Creating VPN configuration data for SOFTNET Security Client V4
- Creating VPN configuration data for SCALANCE M875
- Test and diagnostics functions, status displays

Offline configuration view and online diagnostics view

The security functions are configured in two views:

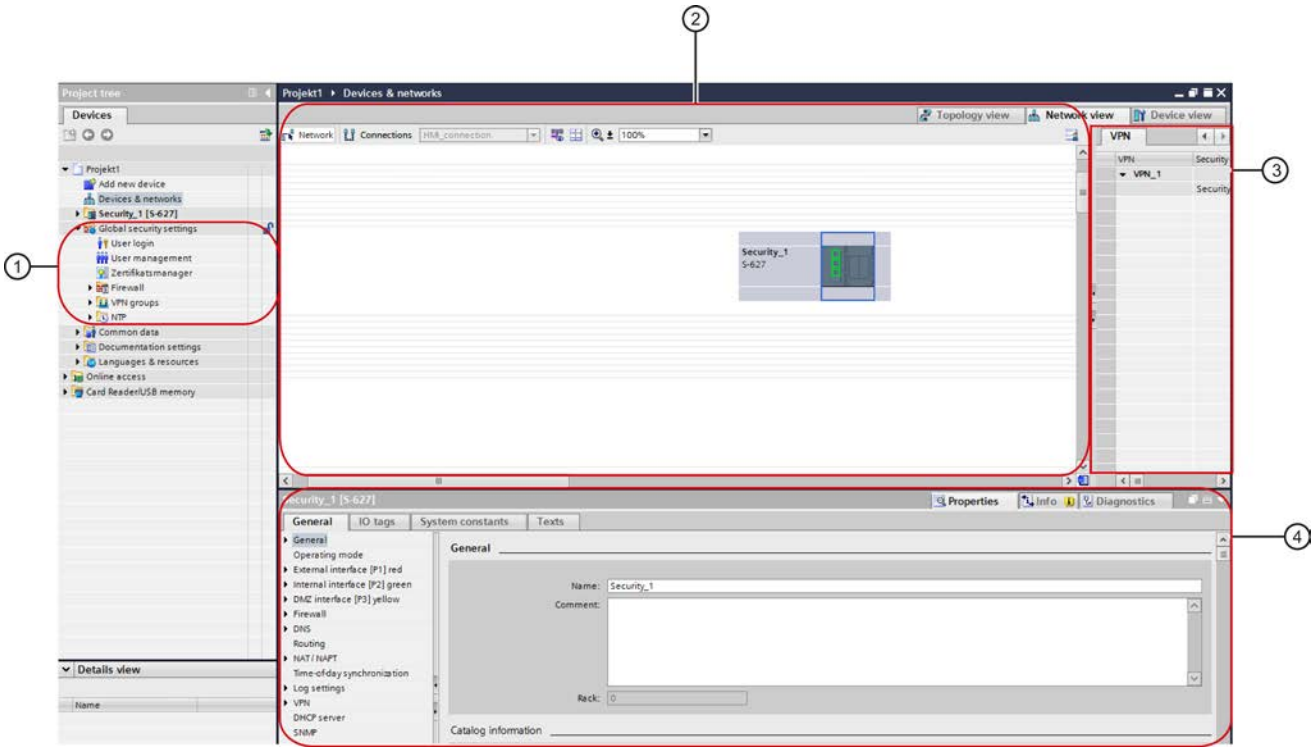
- Offline configuration view
  In the offline configuration view, you create the configuration data for the security modules and the SOFTNET Security Client. Prior to downloading, there must already be a connection to the security modules.
- Online diagnostics view
  The online diagnostics view is used for diagnostics of the security module and, among other things, allows you to run firmware updates.

How it works - security and consistency

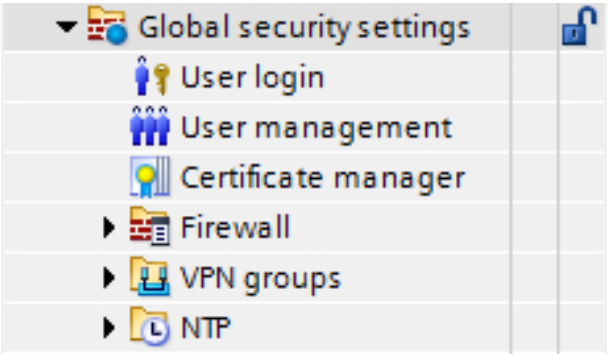
- Access only for authorized users
  The security functions of every project are protected from unauthorized access by assigning user names and passwords.
- Consistent project data
  Consistency checks are running even while you make the entries in the dialogs. In addition, cross-dialog, project-wide consistency checks are carried out.
  Only consistent project data can be downloaded to the security modules.
- Protecting project data by encryption
  The project and configuration data relevant to security are protected by encryption.
  Depending on the security module, data can be stored in the project and/or on the C-PLUG.

User interface - layout and menu commands

User interface for security functions in STEP 7



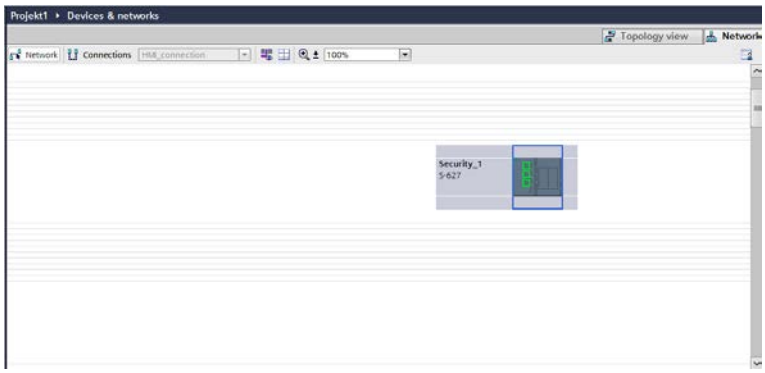
① Global security settings



The global security settings are located in the project navigation. These security settings can be configured independently of the module and subsequently assigned to individual security modules as required. If the first security module to be configured is a CP, the global security settings are only displayed when the security functions have been enabled in the local security settings of the CP. If the first security module to be configured is a SCALANCE S module, the global security settings are displayed after logging in to the security project. The following main folders and entries are available in the global security settings:

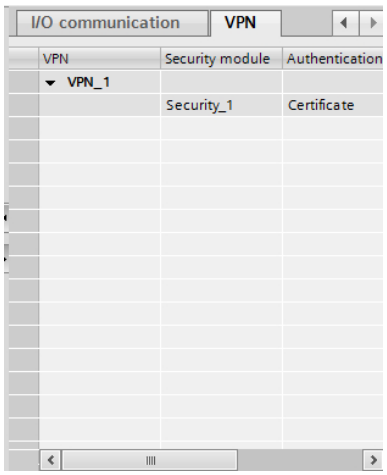
- User login
  For the security configuration within a project, there is a separate user management. Log in to the security configuration using the "User login" entry. The first time that there is a login to the security configuration, a user with the system-defined role "Administrator" is created automatically. You can create further users in the security configuration in the user management.
- User administration
  In user administration, you can create users, define rights for roles and assign these roles to users.
- Certificate manager
  In the certificate manager, you see an overview of all the certificates used in the project. You can, for example, import new certificates as well as export, modify or replace existing certificates.
- Firewall
  Under the "Firewall" entry, you can define global IP and MAC firewall rule sets and user-specific IP rule sets (SCALANCE S modules only) and assign security modules. IP and MAC service definitions are used to define the IP and MAC firewall rules compactly and clearly.
- VPN groups
  All created VPN groups are contained in this folder. You can create new VPN groups here and assign security modules to these VPN groups. You can also adapt VPN group properties of VPN groups that have already been created.
- NTP
  Here, you can create NTP servers and assign them to one or more security modules. This ensures that time synchronization is performed through the assigned NTP server. Unsecured NTP servers can only be configured in the local security settings.

② Working area with security module



Once you have selected a security module in the work area, you can configure its local security settings in "Properties" > "General". If the selected security module is in a VPN group, related information is displayed in the VPN tab.

③ VPN tab



This tab displays information about all the VPN groups to which the security module that was selected in the working area belongs. Information about the respective participants of a VPN group can be displayed and hidden.

④ Local security settings

Local security settings are configured for a specific security module. After a security module has been selected in the working area, its local security settings are available in the inspector window under "Properties" > "General".

Note for CPs:

Before security settings can be configured for CPs, the local security settings must be enabled. To do this, log in to your security project and then in the Inspector window, select the "Activate security features" check box under "Properties" > General" > "Security". The local security settings are then displayed below the "Security" entry. When the check box is selected, the following settings (assuming they were enabled) are migrated automatically to the local security settings.

CP x43-1 Advanced:

- SNMP
- FTP configuration
- Time-of-day synchronization
- Web server
- Entries of IP access lists

CP 1543-1:

- SNMP
- FTP configuration
- Time-of-day synchronization

CP 1628:

- SNMP
- Time-of-day synchronization

In addition, firewall rules that enable a connection to be established are created automatically for configured connections. Log settings are available to record blocked packets.

Secure and non-secure configuration areas

The user interface can be divided into secure and non-secure configuration areas.

The secure areas are areas in which configuration is possible only after logging in to the security configuration. These areas are encrypted and therefore only accessible to persons authorized in the user management even if the project is accessible to a wider circle of people.

Functions from the non-secure areas, on the other hand, can be configured without logging in to the security configuration. The correctness of the settings must be checked before downloading the project to the plant components if a wider circle of people can make modifications to the project.

Below, you will find a list of the configuration areas of the user interface showing which areas are secure and which are non-secure. To some extent, this depends on the security module for which the configuration is created.

- All settings from the global security settings are secure.
- Secure and non-secure configuration areas for SCALANCE S modules:
  - All the settings for the interfaces and ports, in particular IP addresses, are non-secure.
  - The settings under the entry "General" in the local security settings are non-secure.
  - Higher-level settings (e.g. MRP settings such as MRP manager etc.) that are not configured on the security module itself but may affect the security module are non-secure. This does not relate to the global security settings.
  - The other settings are protected.
- Secure and non-secure configuration areas for CP 343-1 GX31 Advanced, CP 443-1 GX30 Advanced, CP 1628, CP 1543-1:
  - All settings outside the "Security" entry are non-secure.
  - Higher-level settings (e.g. MRP settings such as MRP manager, PROFINET settings, connections etc.) that are not configured on the security module itself but may affect the security module are non-secure. This does not relate to the global security settings.
  - All the settings for the interfaces and ports, in particular IP addresses, are non-secure.
  - All settings below the "Security" entry are secure.

Running a consistency check

Overview

The following consistency checks are available:

- Local consistency checks
- Project-wide consistency checks

In the individual dialog descriptions in this help system, the rules you need to take into account for your entries are listed under the keyword "Consistency check".

Local consistency checks

A consistency check is local when it can be performed directly within a dialog. With the following actions, local consistency checks are made:

- After exiting a box
- After exiting a row in a table
- When you confirm a dialog with OK.

Project-wide consistency checks

Project-wide consistency checks provide you with information indicating whether or not project data is correctly configured. With the following actions, there is a consistency check through the entire project:

- When compiling a configuration
- When downloading a configuration

Note

You can only download configured data to a security module when the entire project is consistent.

Managing certificates

Overview of certificates

How do you manage certificates?

In the certificate manager, you have an overview of all the certificates, for example, CA certificates used in the project with information about the applicant, issuer, validity, use and the existence of a private key.

The CA certificate is issued by a certificate authority from which the device certificates are derived. These include the SSL certificates and if the security module is a member of a VPN group the VPN group certificates. The SSL certificates are required for authentication during secure communication between a network node and a security module. Certificate authorities can be:

- STEP 7 itself. If the "applicant" and "issuer" are the same, this is a self-signed certificate; in other words, issued by STEP 7.
- A higher ranking (commercial) certificate authority. These third-party certificates are external to the project and are imported and stored in the certificate store of STEP 7.

Certificates created by one of the two certificate authorities always have a private key so that the device certificates can be derived from them.

The following functions are also available in the certificate manager:

- Modification of existing certificates (for example, duration of validity).
- Import of new certificates and certificate authorities.
- Import of SSL certificates (CP x43-1 Adv. only), e.g. for FTP communication.
- Export of the certificates and certificate authorities used in the project.
- Renewal of expired certificates and certificate authorities.
- Replacement of existing certificate authorities with others.

- Addition of trusted certificates and certificate authorities.
- Deleting manually imported certificates.

Note

Downloading the configuration

After replacing or renewing certificates, the configuration must be downloaded to the relevant security modules.

After replacing or renewing CA certificates, the configuration must be downloaded to all security modules.

Note

Current date and current time of day on the security modules

When using secure communication (for example, HTTPS, VPN...), make sure that the security modules involved have the current time of day and the current date. Otherwise the certificates used are not evaluated as valid and the secure communication does not work.

How to access this function

Double-click on the "Certificate manager" entry in the global security settings.

In the individual tabs, you have the following commands available in the shortcut menu:

Command	Meaning
Import / Export	Import / export of device certificates or CA certificates. The certificates are transferred to the security module. The following formats are permitted: *.cer (certificate only) *.crt (certificate only) *.pem (certificate only) *.p12 (certificate and corresponding private key)
Displays	Opens the certificate dialog of Windows where you see an overview of all certificate data.
Renew (only in the "CA" and "Device certificates" tabs)	Opens the "Create new certificate" dialog in which you can import a certificate or have a new certificate created by STEP 7 when necessary, for example with compromised certificates.
Replace (only in the "CA" tab)	Opens the "Change certificate authority (CA)" dialog in which you can replace an existing certificate authority with a new one.
Delete (only with manually imported certificates)	Deletes a certificate in the "Trusted certificates and root certification authorities" tab.

Certificate authorities

"CA" tab

The certificates displayed here are created by a certificate authority.

- CA certificates of a project: When you create a new project, a CA certificate is generated for the project. The SSL certificates for the individual security modules are derived from this certificate.
- CA group certificates: When you create a new VPN group, a CA certificate is generated for the VPN group. The VPN group certificates of security modules located in the VPN group are derived from this certificate.

Device certificates

"Device certificates" tab

Display of the device-specific certificates generated by STEP 7 for a security module. These include:

- SSL certificate of a security module: An SSL certificate that is derived from a CA certificate of the project is generated for each security module that you create. SSL certificates are used for authentication during secure communication between PG/PC and security module when downloading the configuration.
- VPN group certificate of a security module: A VPN group certificate is also generated for each security module for each VPN group in which it is located.

Trusted certificates and root certification authorities

"Trusted root certification authorities" tab

Display of the third-party certificates imported into STEP 7. For example, server certificates can be imported from external FTPS servers or project certificates from other projects that were created with STEP 7.

With CPs, the imported third-party certificate is transferred to all the CPs managed in the project and these check the certificate. If you classify the certificate sent to the security modules as trusted, a connection can be established, for example, to an FTPS server. The imported certificate is not used at any other point in STEP 7.

With SCALANCE S modules, the certificate authorities required for verifying the security modules using external services such as dyn. DNS are displayed in this tab.

Renewing certificates

Meaning

In this dialog, you renew CA certificates and device certificates. If necessary, for example with compromised certificates, you can import a certificate or have a new certificate generated by STEP 7.

How to access this function

1. Right-click on a list entry in the certificate manager.
2. Select the "Renew" entry in the shortcut menu.
3. Decide whether or not the new certificate will be self-signed or signed by a certificate authority.
4. If the certificate is to be signed by a certificate authority, select the certificate authority to be used with the "Select" button. Only certificate authorities stored in the certificate store of the current project can be selected.
5. Depending on the certificate, enter the following values in the "Applicant" or "Alternative applicant name" input box:

Certificate to be renewed	Parameter	
	Applicant	Alternative applicant name
CA certificates of the project	Name of the CA certificate	-
CA group certificate	Name of the CA group certificate	-
SSL certificate for S7 CP	Name of the security module	Comma-separated IP addresses of the Gigabit and PROFINET interface
SSL certificate for PC CP	Name of the security module	IP address of the security module
SSL certificate for SCALANCE S, SCALANCE M, SOFTNET Security Client	Name of the security module	For SCALANCE S: External IP address, internal IP address and, if applicable, IP address of the DMZ interface For SCALANCE M: External IP address, internal IP address For SOFTNET Security Client: DNS name
VPN group certificate of the security module	Name of the VPN group certificate	Derived from the CA group certificate.

6. Select a period during which the certificate is valid. As default the current time is entered in the "Valid from:" box and the value of the current certificate is entered in the "Valid to:".

Replacing certificates

Meaning

Open the "Change certification authority (CA)" to replace the existing CA certificate of the project or the CA group certificate with a new one.

How to access this function

1. Right-click a list entry in the "CA" tab.
2. Select the "Replace" entry in the shortcut menu.
3. The "Change certification authority (CA)" dialog opens.

All certificates listed in the "Certificates involved" table are derived once again. This means that the CA group certificate of an already configured VPN group can be replaced in the project by the CA group certificate of a different project. The VPN group certificates for the VPN group members are therefore derived from the same CA group certificate in both projects.

After making changes in the certificate manager, download the configuration to all security modules affected.

Which format can the certificate have?

Other certificates are derived from the imported certificate authority in STEP 7. For this reason, you can only select certificates with a private key.

- *.p12

Managing users and roles

Rules for user names, roles and passwords

Which rules apply to user names, role names and passwords?

When creating or modifying a user, a role or a password, remember the following rules:

Table 8-5 Rules for user management

Permitted characters	The following characters from the US-ASCII character set are permitted: 0123456789 A...Z a...z !#$%&()*+,-./:;<=>?@ [\]_{|}~¡^
Characters not allowed	" '
Length of the user name	1 ... 32 characters
Length of the password	8 ... 32 characters
Length of the role name	1 ... 32 characters

Maximum number of users per project	128
Maximum number of users on one security module	32 + 1 administrator when creating the project
Maximum number of roles per project	124 (120 user-defined + 4 system-defined)
Maximum number of roles on one security module	33 (29 user-defined + 4 system-defined)

Note

User names and passwords

As vital measure under the aspect of maximizing security, always make sure that user names and passwords have a minimum length of 8 characters and that these contain special characters, uppercase/lowercase letters, number, etc..

Create users

Meaning

The security functions configured in STEP 7 are protected from unauthorized access by a separate user management. Before you can access the global and local security settings of security modules, you need to log in to the security configuration as a user.

Creating the first user in the project

After creating the first security module in the project, you first need to create a user. To do this, in the local security settings of the created security module, click the "User login" button under the "Security properties" entry and enter the login data of the user you want to create. You will then be logged in as the created user and the user is assigned the system-defined "Administrator" role. This role includes all configuration and module rights.

Creating users in the user management

When you are logged in as a user in the security configuration, you can create further users or delete users with the "User management" entry in the global security settings.

Note

Users with the "Administrator" role

There must always be at least one user with full configuration rights within a project. The administrator that is created automatically when you first enable the security functions in the project can only be deleted if at least one other user exists with the system-defined role "Administrator".

The following parameters are available in user management in the "User" tab:

Table 8-6 Information in the "User" tab

Parameter	Meaning
User name	Name of the user to be created. Click on the "Add new user" entry in the "User name" column to create a new user.
Password	Password with which a user authenticates him/herself when logging on.
Role	Selecting a system-defined or user-defined role.
Maximum time of the session (only for SCALANCE S V3 or higher)	Entry of the time after which a user logged on to the Web page for user-specific IP rule sets of SCALANCE S modules is automatically logged off. The time entered here starts after the logon and after renewing the session on the Web page of the security module. • Default setting: 30 minutes • Minimum value: 5 minutes • Maximum value: 480 minutes
Comment	Entry of optional comments.

Creating roles

Overview

You can assign a system-defined or a user-defined role to a user. Specify the module rights of a user-defined role for each security module.

System-defined roles

The system-defined roles listed below are predefined. Certain rights are assigned to the roles that are the same on all modules and that the administrator can neither change nor delete.

- Administrator
  Default role when creating a security configuration.
  Unlimited access rights to all configuration data and all security modules.
- Standard
  Role with restricted access rights.
- Diagnose
  Default role when creating new user.
  - Read access to configurations.
  - Read access to the security module in the "Online" mode for testing and diagnostics.
- Remote-Access
  Can be used as a template when creating user-defined roles.

You will find a detailed list of the configuration and module rights assigned to the system-defined roles "Administrator", "Standard" and "Diagnostics" in tables 1-3 to 1-7 of the section Managing rights (Page 740).

User-defined role

In addition to the system-defined roles, you can create user-defined roles. For a user-defined role, select the configuration or module rights and specify the appropriate module rights for every security module used in the project. You manually assign the user-defined roles to the relevant user.

How to access this function

1. Double-click on the "User management" entry in the global security settings.
2. Select the "Roles" tab in User management.

Table 8-7 Information in the "Roles" tab

Parameter	Meaning
Role	Freely selectable role name. Double-click on the "Add new role" entry to create a new user-defined role. You can then set the rights for the created role.
Description	Specifying the system-defined role With user-defined roles, the "User-defined role" character string is displayed.
Maximum time of the session (only for SCALANCE S V3 or higher)	Entry of the time after which a user with the assigned role is automatically logged off from the Web page for user-specific IP rule sets of SCALANCE S modules. The time entered here starts after the logon and after renewing the session on the Web page of the security module. • Default setting: 30 minutes • Minimum value: 5 minutes • Maximum value: 480 minutes
Comment	Entry of additional, optional comments.

Note

Deleting roles

A user-defined role can only be deleted when it is no longer assigned to any user. If necessary, assign the user a different role.

System-defined roles cannot be deleted.

Managing rights

How to access this function

1. Double-click on the "User management" entry in the global security settings.
2. Select the "Roles" tab in User management.

Creating and assigning a user-defined role

1. Double-click on the "Add new role" entry.
2. Enter a role name.
3. If necessary, select the system-defined role whose rights will be used as the template for the user-defined role from the drop-down list labeled "<Copy rights from>". User-defined roles cannot be selected from the drop-down list.
   Result: In the list of rights of the user roles, the rights are selected that are assigned to the selected system-defined role.
4. For each security module, enable or disable the rights to be assigned to the user-defined role.
5. Assign the role to a user in the "User" tab.

Configuration rights

Configuration rights are not module dependent and control the rights for configuration in STEP 7.

Depending on the user type, the following configuration rights are available for selection:

Table 8-8 Configuration rights

Configuration right	Administrator	Standard	Diagnose
Diagnose security	x	x	x
Configure security	x	x	-
Managing users and roles	x	-	-

Module rights

Module rights are configured per module. The "Service" column shows the service to which the particular right relates. With the "Copy rights" and "Paste rights" commands in the shortcut menu, you can transfer the rights from one module to another.

Depending on the user type, the following module rights are available for selection:

Table 8-9 Module rights CP x43-1 Advanced

Right within the service	Administrator	Standard	Diagnose	Service
Web: Format CP file system *	x	-	-	File system
FTP: Read files from the CP file system	x	x	x	
FTP: Write files to the CP file system	x	x	-	
FTP: Read files (DBs) from the S7 CPU **	x	x	x	PLC
FTP: Write files (DBs) to the S7 CPU ***	x	x	-	
Applet: Read tags using configured symbols *	x	x	x	
Applet: Write tags using configured symbols *	x	x	-	
Applet: Read tags using absolute addresses *	x	x	x	
Applet: Write tags using absolute addresses *	x	x	-	
Applet: Read status of the modules in the rack *	x	x	x	

Right within the service	Administrator	Standard	Diagnose	Service
Applet: Query order numbers of the modules in the rack *	x	x	x	
SNMP: Read MIB-II	x	x	x	SNMP
SNMP: Write MIB-II	x	x	-	
SNMP: Read automation MIB	x	x	x	
SNMP: Read LLDP-MIB	x	x	x	
SNMP: Read SNMPv2-MIB	x	x	x	
SNMP: Read MRP MIB	x	x	x	
SNMP: Write MRP MIB	x	x	-	
TIA Portal: Run diagnostics of the security module ****	x	x	x	Security
Web: Expand IP access control list *	x	-	-	Web
Web: Access Web diagnostics and CP file system	x	x	x	
Web: Send test e-mails *	x	x	x	
Web: Update firmware *	x	x	-	Maintenance
Web: Load diagnostics texts later *	x	x	-	

Table 8-10 Module rights CP 1628

Right within the service	Administrator	Standard	Diagnose	Service
SNMP: Read MIB-II	x	x	x	SNMP
SNMP: Write MIB-II	x	x	-	
SNMP: Read automation MIB	x	x	x	
SNMP: Read SNMPv2-MIB	x	x	x	
TIA Portal: Run diagnostics of the security module ****	x	x	x	Security

Table 8-11 Module rights SCALANCE S

Right within the service	Administrator	Standard	Diagnose	Service
Download the configuration files	x	x	-	Security
TIA Portal: Run diagnostics of the security module ****	x	x	x	
SNMP: Read automation MIB	x	x	x	SNMP
SNMP: Read MIB-II	x	x	x	
SNMP: Write MIB-II	x	x	-	
SNMP: Read MRP-MIB	x	x	x	
SNMP: Write MRP-MIB	x	x	-	
SNMP: Read SNMPv2-MIB	x	x	x	
Web: Update firmware	x	x	-	Maintenance

Table 8-12 Module rights CP 1543-1

Right within the service	Administrator	Standard	Diagnose	Service
FTP: Read files from the CP file system	x	x	x	File system
FTP: Write files to the CP file system	x	x	-	
TIA Portal: Run diagnostics of the security module ****	x	x	x	Security

Right within the service	Administrator	Standard	Diagnose	Service
SNMP: Read automation MIB	x	x	x	SNMP
SNMP: Read IPv6 MIB	x	x	x	
SNMP: Read LLDP-MIB	x	x	x	
SNMP: Read MIB-II	x	x	x	
SNMP: Write MIB-II	x	x	-	
SNMP: Read SNMPv2-MIB	x	x	x	
FTP: Read files (DBs) from the S7 CPU **	x	x	x	PLC
FTP: Write files (DBs) to the S7 CPU ***	x	x	-	

* To be able to use the function, the module right "Web: Access Web diagnostics and CP file system" must be enabled as well.

** To be able to use the function, the module right "FTP: Read files from CP file system" must be enabled as well.

*** To be able to use the function, the module right "FTP: Write files to CP file system" must be enabled as well.

**** To use the function, the configuration right "Security diagnostics" must also be enabled.

Setting module rights before and after creating the security modules

Within a user-defined role, the module rights for each security module are defined separately. If a security module was created before this role was added and module rights within a role need to be set, STEP 7 presets the module rights for the security module according to the selected rights template. The preset module rights can then be adapted when adding the role. If a security module was created after adding a role, no rights are preset for this security module. In this case, you will need to edit an existing role and set all the module rights yourself for the security module.

You can also transfer existing module rights to another module by copying and, if necessary, adapting them there. To do this, select a module in the shortcut menu in the module rights and select the "Copy rights" or "Paste rights" menu command.

Generating configuration data for SCALANCE M modules

Context

You can generate the VPN information for the assignment of parameters to a SCALANCE M using STEP 7. For this to be possible, the module must be in at least one VPN group with a security module or a SOFTNET Security Client. With the generated files, you can then configure the SCALANCE M using the Web Based Management of the device.

Generated files

The following file types are generated:

- Export file with the configuration data
  - File type: *.txt file in ASCII format
  - Contains the exported configuration information for the SCALANCE M including information on the additionally generated certificates.
- VPN group certificates of the module
  - File type of the private key: *.p12 file
  - The file contains the module certificate and the key material.
  - Access is password protected.
- CA certificates of VPN groups
  - File type: *.cer file

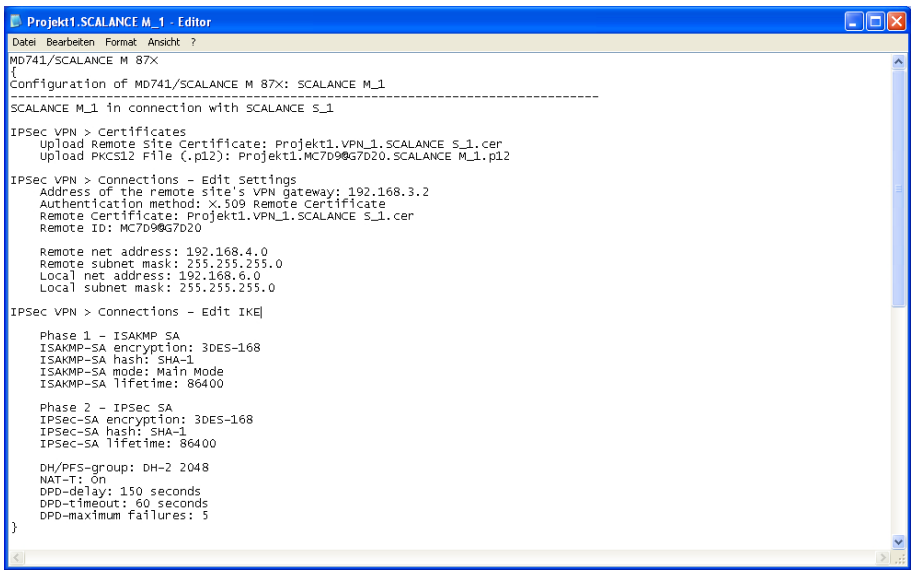


Figure 8-6 SCALANCE M configuration file

Note

No transfer to the security module

Configuration files are not transferred to the security module. An ASCII file is generated with which you can configure the SCALANCE M. To allow this, the SCALANCE M must be located in at least one VPN group with another security module.

Follow the steps below

1. Select the module of the type "SCALANCE M".
2. In the local security settings, select the entry "SCALANCE M configuration".

3. Select the "Generate SCALANCE M files" check box and select a storage location for the configuration files.
4. Compile the configuration of the SCALANCE M module.
5. In the dialog that follows, choose whether you want to create your own password for the created .p12 file.
   If you select "Cancel", the project name is assigned as the password, not the project password.

Result: The files (.txt file and certificates) are stored in the folder you specify.

Configuring interfaces for SCALANCE S modules

Overview

You will find information on configuration interfaces of SCALANCE S modules in Auto-Hotspot of the section "SCALANCE S". The information contained here deals with the following configuration options:

- Mode (bridge mode / routing mode): Setting the operating mode (Page 812)
- IP address parameters: Configuring IP address parameters (Page 812)
- Port settings (only for SCALANCE S ≥ V3): Configuring port mode (Page 814)
- Settings for the Internet Service Provider (ISP) if one of the interfaces is operated using PPPoE (only for SCALANCE S ≥ V3): Configuring an Internet connection (Page 814)
- Dynamic DNS (only for SCALANCE S ≥ V3): Configuring dynamic DNS (Page 815)
- LLDP (only for SCALANCE S ≥ V4 in routing mode): Configuring LLDP (Page 818)
- Media redundancy in ring topologies (MRP client), (only for SCALANCE S ≥ V4 in routing mode): Auto-Hotspot

The configuration of the interfaces of CPs is described in the sections dealing with the CPs.

Setting up a firewall

Overview of the firewall

Meaning

The firewall functionality of the security modules is intended to protect networks and stations from third-party influence and interference. This means that only certain, previously specified communications relations are permitted. The firewall discards invalid frames without sending a response.

To filter the data traffic, IP addresses, IP subnets, port numbers or MAC addresses can be used. You can also set a bandwidth limitation.

The firewall functionality can be configured for the following protocol levels:

- IP firewall with stateful packet inspection (layer 3 and 4)
- Firewall also for Ethernet "non-IP" frames according to IEEE 802.3 (layer 2)

With security modules capable of VPN, the firewall can also be used for the encrypted data traffic (IPsec tunnel). With the SCALANCE S602 Security module, the firewall can only be used for unencrypted data traffic.

Firewall rules

Firewall rules describe which packets in which direction are permitted or forbidden.

Types of firewall rules

- Global firewall rule sets: Global firewall rule sets can be assigned to several security modules at the same time. Global firewall rule sets are configured in the global security settings.
- Local firewall rules: Local firewall rules are configured in the local security settings of a security module.
- User-specific IP rule sets (only for SCALANCE S ≥ V3): User-specific IP rule sets can be assigned to individual or several security modules at the same time. User-specific IP rule sets are configured in the global security settings and assigned there to one or more users. You will find information on configuring user-specific IP rule sets for SCALANCE S V3 modules in the following section:
  Auto-Hotspot

Service definitions

With the aid of service definitions, you can also define firewall rules clearly in a compact form. Service definitions are configured in the global security settings and can be used in the global, local and user-specific firewall rules.

Adapting standard rules for IP services

For SCALANCE S modules V3 or higher, you can adapt service-specific firewall rules that are set as default for the interfaces of the security modules. You will find information on configuring these firewall rules in Adapting standard rules for IP services (Page 827) of the section "SCALANCE S".

Automatically generated firewall rules for CP connections

For connections that were configured using CPs, STEP 7 automatically creates firewall rules that allow communication with the partner of the CP in the specified direction (CP active/ passive). The connection establishment directions are taken into account. To display these firewall rules, if the advanced firewall mode is enabled, the "Update connection rules" button needs to be clicked. The firewall rules are then displayed in advanced firewall mode.

Which firewall rules are generated automatically is described in the following sections.

- For S7-300 CPs/S7-400 CPs/PC CPs: Connection-related automatic firewall rules (Page 851) in the section "Security for S7-300 / S7-400 / PC CPs".
- For S7-1500 CPs: Connection-related automatic firewall rules (Page 860) in the section "Security for S7-1500 CPs".

Enabling the firewall

Firewall functionality for a specific security module is controlled in the local security settings using the "Enable firewall" check box. If the check box is enabled, the firewall can be configured and is effective following the download. For a security module that is a member of a VPN group, the "Enable firewall" check box is activated by default and cannot be deactivated. After switching to advanced firewall mode, it is not possible to switch back to standard mode. For more information about the standard and advanced firewall modes, refer to the section:

Overview of local firewall rules (Page 755).

Global firewall rule sets

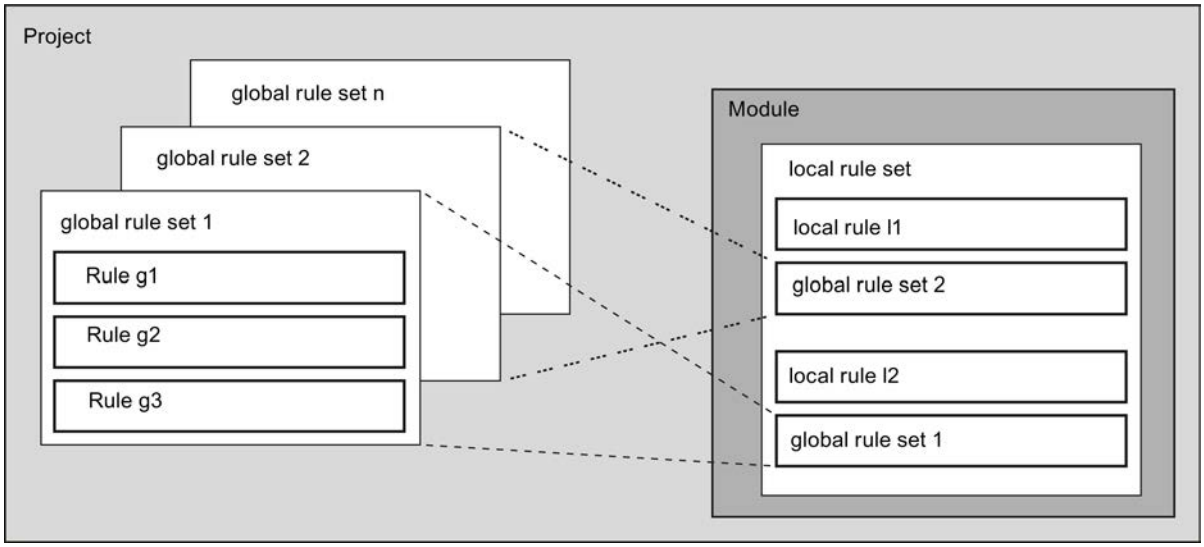
Application

Regardless of the module, global firewall rule sets are configured in the global security settings. A firewall rule set consists of one or more firewall rules and is assigned to the individual security modules.

In the global firewall rule sets, a distinction is made between the following:

- IP rule sets
- MAC rule sets

The following schematic illustrates the relationship between globally defined rule sets and locally used rule sets.



Configuring

When configuring global firewall rules, you can make detailed firewall settings. You can allow individual services for a single node or all services for the node for access to the station or network.

When are global IP and MAC firewall rules useful?

Global firewall rules are useful if you want to define identical filter criteria for communication with several security modules.

Note

Assigning firewall rule sets with incompatible firewall rules

For a security module, only the firewall rules from firewall rule sets are adopted correctly if the security module supports them. A firewall rule with the direction "From external" or "To any" in a global firewall rule set is not assigned to a CP 1628. The other firewall rules of the global firewall rule set are adopted if the CP supports them.

Global firewall rule sets - conventions

Global firewall rule sets are used locally

The following conventions apply when creating a global set of firewall rules and when assigning it to a module:

- Configuration view
  Global firewall rule sets are configured in the global security settings.
- Priority
  As default, locally defined rules have a higher priority than the global IP and MAC firewall rule sets. Newly assigned global IP and MAC firewall rule sets are therefore initially entered at the bottom of the local rule list.
  The priority can be changed by changing the position in the rule list.
- Entering, changing or deleting rule sets
  Global firewall rule sets cannot be edited in the local rule list of the firewall rules in the module properties. They can only be displayed there and positioned according to the required priority.
  It is not possible to delete a single rule from an assigned rule set in the local security settings. Only the entire rule set can be removed from the local list of rules. The firewall rule sets in the global security settings remain unaffected.

Creating global firewall rule sets

How to access this function

1. In the global security settings, select the entry "Firewall" > "Global firewall rule sets" > "IP rule sets" or "MAC rule sets".
   Result: The previously created IP rule sets or MAC rule sets are displayed under the selected entry.
2. Double-click on the entry "Add new IP rule set" or "Add new MAC rule set".
3. Enter the following data:
   – Name: Project-wide, unique name of the rule set. The name appears in the local rule list of the security module after the rule set is assigned.
   – Description (optional): Enter a description of the global rule set.
4. Enter the firewall rules one by one in the list.
   Note the parameter description in the following sections:
   For IP rule sets: Defining IP packet filter rules (Page 756)
   For MAC rule sets: Defining MAC packet filter rules (Page 759)

Result

You have created the global firewall rule set and can now assign this to the required security modules.
Note the descriptions in the following section:

Assigning global firewall rule sets (Page 749)

Assigning global firewall rule sets

Requirement

You have enabled the advanced firewall mode for the security modules you want to assign to a firewall rule set.

Procedure

1. In the global security settings, select the entry "Firewall" > "Global firewall rule sets" > "Assign module to a firewall rule set".
2. From the "Rule set" drop-down list, select the rule set to which you want to assign the security module.
   In the right-hand table, you will see the security modules that you can assign to the selected firewall rule set. In the left-hand table, you will see the security modules already assigned to the selected firewall rule set.
3. In the "Available modules" area, select the security modules you want to assign to the selected rule set.
4. Click the << button to assign the selected modules to the selected rule set.

Result

The global rule set is used by the assigned security modules as a local rule set and appears automatically at the end of the list of firewall rules in the local security settings.

IP services

Defining IP services

How to access this function

In the global security settings, select the entry "Firewall" > "Services" > "Define services for IP rules".

Procedure

Using the definition of IP services, you can define succinct and clear firewall rules for specific services. You select a name and assign the service parameters to it.

These services defined in this way can also be grouped together under a group name.

When you configure the packet filter rules, you then use this name.

Parameters for IP services

You define the IP services using the following parameters:

Table 8-13 IP services: Parameter

Parameter	Meaning/comment	Available options / ranges of values
Name	Name for the service that is used as identification in the rule definition or in the group. The names of predefined services cannot be changed.	• Name must start with a letter. • Name must not contain any special characters. • Name must not be redundant.
Protocol	Selection of the protocol type	• TCP • UDP • All
Source port	The filtering is based on the specified port number; this defines the service access at the frame sender.	Examples: *: Port is not checked 20 or 21: FTP service
Destination port	The filtering is based on the specified port number; this defines the service access at the frame recipient.	Examples: *: Port is not checked TCP 80: Web HTTP service TCP 102: S7 protocol

Defining ICMP services

How to access this function

1. In the global security settings, select the entry "Firewall" > "Services" > "Define services for IP rules".
2. Select the "ICMP" tab.

Procedure

Using the definition of ICMP services, you can define succinct and clear firewall rules for specific services. You select a name and assign the service parameters to it.

These services defined in this way can be grouped together under a group name.

When you configure the packet filter rules, you then use this name.

Parameters for ICMP services

Parameter	Meaning/comment	Available options / ranges of values
Name	User-definable name for the service that is used as identification in the rule definition or in the group. The names of predefined ICMPv6 services cannot be modified.	• Name must start with a letter. • Name must not contain any special characters. • Name must not contain more than 20 characters. • Name must not occur more than once
ICMPv6	If you enable this check box, the ICMP service is declared as an ICMPv6 service and you can select an ICMPv6-specific type and code for the service. An ICMPv6 service can only be used in the firewall rule of a security module that supports IPv6.	• Enabled • Disabled (default)
Type	Type of the ICMPv4 or ICMPv6 message.	If the "ICMPv6" check box is deselected, ICMPv4-specific types can be selected. If the check box is selected, ICMPv6-specific types can be selected.
Code	Code of the ICMP type.	Values depend on the selected type.

Creating service groups

How to access this function

1. In the global security settings, select the entry "Firewall" > "Services" > "Define services for IP rules".
2. Select the "Service groups" tab.

Forming service groups

You can put several services together by creating service groups. This allows you set up more complex services that you can then use in the packet filter rules simply by selecting the name. IPv4 and IPv6 services can be collected in the same service group.

Create groups in the open "Service groups" tab. You can then assign services to a group in the "Group management" tab.

Follow the steps below

1. First, create groups in this tab with names to suit your requirements and add a description if required.
2. Select the "Group management" tab. You can assign the previously specified IP services to the groups defined in this tab.

Managing service groups

How to access this function

1. In the global security settings, select the entry "Firewall" > "Services" > "Define services for IP rules".
2. Select the "Group management" tab.

Forming service groups

You can put several services together by creating service groups. This allows you set up more complex services that you can then use in the packet filter rules simply by selecting the name. IPv4 and IPv6 services can be collected in the same service group.

Use the "Group management" tab to assign services to a selected group you created previously in the "Service groups" tab.

Follow the steps below

1. Select a group created previously in the "Service groups" tab from the "Service groups" drop-down list in this tab.
2. Then assign the required services from the right-hand "Available services" list box to the group.

MAC services

Define MAC services

How to access this function

In the global security settings, select the entry "Firewall" > "Services" > "Define services for MAC rules".

Meaning

Using the definition of MAC services, you can define succinct and clear firewall rules for specific services. You select a name and assign the service parameters to it.

These services defined in this way can be grouped together under a group name.

When you configure the global or local packet filter rules, you use this name.

Parameters for MAC services

A MAC service definition is formed using protocol-specific MAC parameters:

Table 8-14 MAC services - parameters

Parameter	Meaning/comment	Available options / ranges of values
Name	User-definable name for the service that is used as identification in the rule definition or in the group.	• Name must start with a letter. • Name must not contain any special characters. • Name must not be redundant.
Protocol	Name of the protocol type: • ISO ISO identifies frames with the following properties: Lengthfield <= 05DC (hex), DSAP= userdefined SSAP= userdefined CTRL= userdefined • SNAP SNAP identifies frames with the following properties: Lengthfield <= 05DC (hex), DSAP=AA (hex), SSAP=AA (hex), CTRL=03 (hex), OUI=userdefined, OUI-Type=userdefined • PROFINET IO • As an alternative, a protocol number can be entered. The protocol entries 0800 (hex) and 0806 (hex) are not accepted since these values apply to IP or ARP frames.	• ISO • SNAP • PROFINET IO • 0x (code entry)
DSAP	Destination Service Access Point: LLC recipient address	
SSAP	Source Service Access Point: LLC sender address	

Parameter	Meaning/comment	Available options / ranges of values
CTRL	LLC control field	
OUI	Organizationally Unique Identifier (the first 3 bytes of the MAC address = vendor ID)	
OUI type	Protocol type/identification	

Note

Processing for S7-CPs

Only settings for ISO frames with DSAP=SSAP=FE (hex) are processed. Other frame types are not relevant for S7 CPs and are therefore discarded even before processing by the firewall.

Special settings for SIMATIC NET services

To filter special SIMATIC NET services, please use the following protocol settings:

- DCP (Primary Setup Tool):
  PROFINET
- SiClock :
  OUI= 08 00 06 (hex) , OUI-Type= 01 00 (hex)

Creating service groups

How to access this function

1. In the global security settings, select the entry "Firewall" > "Services" > "Define services for MAC rules".
2. Select the "Service groups" tab.

Forming service groups

You can put several services together by creating service groups. This allows you set up more complex services that you can then use in the packet filter rules simply by selecting the name.

Create groups in the open "Service groups" tab. You can then assign services to a group in the "Group management" tab.

Follow the steps below

1. First, create groups in this tab with names to suit your requirements and add a description if required.
2. Select the "Group management" tab. You can assign the previously specified MAC services to the groups defined in this tab.

Managing service groups

How to access this function

1. In the global security settings, select the entry "Firewall" > "Services" > "Define services for MAC rules".
2. Select the "Group management" tab.

Forming service groups

You can put several services together by creating service groups. This allows you set up more complex services that you can then use in the packet filter rules simply by selecting the name.

Use the "Group management" tab to assign services to a selected group you created previously in the "Service groups" tab.

Follow the steps below

1. Select a group created previously in the "Service groups" tab from the "Service groups" drop-down list in this tab.
2. Then assign the required services from the right-hand "Available services" list box to the group.

Overview of local firewall rules

Meaning

Local firewall rules are configured in the local security settings of a security module and apply only to this security module. After enabling the firewall functionality you can either use predefined firewall rules or define firewall rules in the advanced firewall mode.

Using predefined firewall rules

Here, you use simple, predefined rules. You can only enable service-specific rules. The enabled services are allowed for all nodes in the specified direction. You can find detailed information on the definition of firewall rules in this dialog in the following module-specific sections:

- For SCALANCE S: Auto-Hotspot
- For S7-300 CPs/S7-400 CPs/PC CPs: Auto-Hotspot
- For S7-1500 CPs: Auto-Hotspot

Defining firewall rules in advanced firewall mode

In advanced firewall mode, you can make detailed firewall settings. You can allow individual services for a single node or all services for the node for access to the station or network. You enable the advanced firewall mode using the "Activate firewall in advanced mode" check box.

In the local security settings, the firewall rules can then be configured in "Firewall" > "IP rules" or "MAC rules". The configuration options available here are described individually in the following section:

For IP packet filter rules: Defining IP packet filter rules (Page 756)

For MAC packet filter rules: Defining MAC packet filter rules (Page 759)

Note

Deactivating the advanced firewall mode not possible

Once you have activated the advanced firewall mode, you can no longer deactivate it.

Quantity structure

Number of firewall rules (advanced firewall mode)	
SCALANCE S	Maximum of 256
S7 CPs	Maximum of 256
CP 1628	Maximum of 256

Defining IP packet filter rules

Meaning

With IP packet filter rules, you filter according to IP frames, such as TCP, UDP and ICMP.

Within a packet filter rule, you can also fall back on the definitions of the IP services.

Entering IP packet filter rules

Enter the firewall rules one by one in the list. Note the following parameter description.

Table 8-15 IP rules: Parameter

Parameter	Meaning/comment	Available options / ranges of values
Action	Allow/disallow (enable/block)	• Allow Allow frames according to definition. • Drop Block frames according to definition. For firewall rules generated automatically as result of configuring a connection and subsequently adapted manually: • Allow* • Drop* If you change automatically created connection rules, when the "*" option is selected they are not recreated and overwritten by STEP 7.
From / To	Selection of the communications directions for which the rule will apply.	Described in separate sections. • For SCALANCE S modules: IP packet filter directions SCALANCE S (Page 826) • For S7-300 CPs/S7-400 CPs/PC CPs: IP packet filter directions S7-300-/S7-400-/PC-CPs (Page 849) • For S7-1500 CPs: IP packet filter directions S7-1500 CPs (Page 859)
IPv6 (for the CP 1543-1 only)	If this check box is selected, you can use a previously defined ICMPv6 service in the firewall rule. For the CP 1543-1 V1.1, after selecting the check box you can enter additional IPv6 addresses in the "Source IP address" and "Destination IP address" input boxes. For the CP 1543-1 V1.0, after enabling the check box you cannot enter any source IP address or any destination IP address for the firewall rule.	The check box can only be selected and deselected if there are no entries in the "Source IP address" and "Destination IP address" input boxes. If IPv6 is disabled in the local settings of the CP 1543-1, you cannot select the "IPv6" check box in the local security settings of the CP and therefore cannot use ICMPv6 services or IPv6 addresses in firewall rules. Existing firewall rules that use IPv6 are shown grayed out if IPv6 is disabled.
Source IP address	The firewall rule is applied to the frames whose sender has the IP address specified here. If you do not specify an IP address, the firewall rule applies to all nodes in the communications direction you selected in the "From" column.	You can find additional information about IP addresses in the IP addresses in IP packet filter rules (Page 760) section.
Destination IP address	The firewall rule is applied to the frames whose recipient has the IP address specified here. If you do not specify an IP address, the firewall rules applies to all nodes in the communications direction you selected in the "To" column.	

Parameter	Meaning/comment	Available options / ranges of values
Service	Name of the IP/ICMP service or service group used. Here, you select one of the services you defined in the "IP services" dialog: • IP services or • ICMP services Before you select an ICMPv6 service, you need to select the "IPv6" check box.	The drop-down list box displays the services and service groups configured in the global security settings and you can select them.
Bandwidth (Mbps)	Option for setting a bandwidth limitation Can only be entered if the "Allow" action is selected. A packet passes through the firewall if the allow rule matches and the permitted bandwidth for this rule has not yet been exceeded.	CP x43-1, CP 1543-1 and SCALANCE S < V3.0: 0.001 ... 100 Mbps CP 1628 and SCALANCE S ≥ V3.0: 0.001 ... 1000 Mbps For global and user-specific rules: 0.001 ... 100 Mbps Note: If the direction "From tunnel to station" is configured in a firewall rule for the CP 1543-1, no bandwidth limitation can be specified.
Logging	Enable and disable logging for this rule. If logging is enabled, the settings for the packet filter logging configured in the local security settings apply.	• Enabled • Disabled (default)
Number	Automatically assigned number for the rule	
Comment	Space for your own explanation of the rule	If a comment is marked with "AUTO", it was created for an automatic connection rule. For rules you have created, entry of a comment is optional.

Table 8-16 Meaning of the entries in the shortcut menu

Entry in the shortcut menu	Meaning
Delete	This deletes the selected rule or the selected rule set. Notes on removing a globally defined and locally assigned rule set: if you delete a rule set here, this only cancels the assignment to the security module.
Save as global rule set (only for local firewall rules)	Copies the selected firewall rule(s) and inserts it as a global firewall rule set in the global security settings. The firewall configuration currently configured for the security module remains unaffected by this procedure.
Move up	Use this button to move the selected rule or selected rule set up one position in the list. The rule / rule set you moved is therefore handled with higher priority.
Move down	Use this button to move the selected rule or selected rule set down one position in the list. The rule / rule set you moved is therefore handled with lower priority.
Define service for IP rules	This opens the dialog in which you can manage the IP services and service groups.

Defining MAC packet filter rules

Meaning

With the MAC packet filter rules, you filter according to MAC frames.

Within a packet filter rule, you can also fall back on the definitions of the MAC services.

Entering MAC packet filter rules

Enter the firewall rules one by one in the list. Note the following parameter description.

Table 8-17 MAC rules: Parameter

Parameter	Meaning/comment	Available options / ranges of values
Action	Allow/disallow (enable/block)	• Allow Allow frames according to definition. • Drop Block frames according to definition.
From / To	Selection of the communications directions for which the rule will apply.	Described in separate sections. • For SCALANCE S modules: MAC packet filter directions SCALANCE S (Page 827) • For S7-300 CPs/S7-400 CPs/PC CPs: MAC packet filter directions S7-300-/S7-400-/PC-CPs (Page 850) • For S7-1500 CPs: MAC packet filter directions S7-1500 CPs (Page 860)
Source MAC address	The firewall rule is applied to the frames whose sender has the MAC address specified here. If you do not specify a MAC address, the firewall rule applies to all nodes in the communications direction you selected in the "From" column.	MAC address in the correct format
Destination MAC address	The firewall rule is applied to the frames whose recipient has the MAC address specified here. If you do not specify a MAC address, the firewall rule applies to all nodes in the communications direction you selected in the "To" column.	
Service	Name of the MAC service or service group used	The drop-down list box displays the configured services and service groups you can select.
Bandwidth (Mbps)	Option for setting a bandwidth limitation. Can only be entered if the "Allow" action is selected. A packet passes through the firewall if the allow rule matches and the permitted bandwidth for this rule has not yet been exceeded.	CP x43-1, CP 1543-1 and SCALANCE S < V3.0: 0.001 ... 100 Mbps CP 1628 and SCALANCE S ≥ V3.0: 0.001 ... 1000 Mbps For global rules: 0.001 ... 100 Mbps Note: If the direction "From tunnel to station" is configured in a firewall rule for the CP 1543-1, no bandwidth limitation can be specified.

Parameter	Meaning/comment	Available options / ranges of values
Logging	Enable and disable logging for this rule. If logging is enabled, the settings for the packet filter logging configured in the local security settings apply.	• Enabled • Disabled (default)
Number	Automatically assigned number for the rule	
Comment	Space for your own explanation of the rule	If a comment is marked with "AUTO", it was created for an automatic connection rule. For rules you have created, entry of a comment is optional.

Table 8-18 Meaning of the menu commands

Button	Meaning
Delete	This deletes the selected rule or the selected global rule set. Notes on removing a globally defined and locally assigned rule set: if you delete a rule set here, this only cancels the assignment to the security module.
Save as global rule set (only for local firewall rules)	Copies the selected firewall rule(s) and inserts it as a global firewall rule set in the global security settings. The firewall configuration currently configured for the security module remains unaffected by this procedure.
Move up	Use this button to move the selected rule or selected global rule set up one position in the list. The rule / rule set you moved is therefore handled with higher priority.
Move down	Use this button to move the selected rule or selected global rule set down one position in the list. The rule / rule set you moved is therefore handled with lower priority.
Define service for MAC rules	This opens the dialog in which you can manage the MAC services and service groups.

IP addresses in IP packet filter rules

Entering IP addresses in IP packet filter rules

In IP packet filter rules, you have the following options for entering IP addresses:

- Nothing specified
  The rule applies to all IP addresses.
- An IP address
  The rule applies specifically to the specified address.

- Address range
  The rule applies to all the IP addresses covered by the address range.
  An address range is defined by specifying the number of valid bit places in the IP address in the format: [IP address]/[number of bits to be included]
  - [IP address]/24 therefore means that only the most significant 24 bits of the IP address are included in the filter rule. These are the first three octets of the IP address.
  - [IP address]/25 means that only the first three octets and the highest bit of the fourth octet of the IP address are included in the filter rule.
- Address area
  For the source IP address, an address range can be entered in the following format:
  [Start IP address]-[End IP address]

IPv4 addresses

An IPv4 address consists of 4 decimal numbers in the range from 0 to 255 separated from each other by a dot.

Table 8-19 Examples of address ranges in IPv4 addresses

Source IP address or destination IP address	Address range		Number of addresses*)
	from	to	
192.168.0.0/**16**	192.168.0.0	192.168.255.255	65.536
192.168.10.0/**24**	192.168.10.0	192.168.10.255	256
192.168.10.0/**25**	192.168.10.0	192.168.10.127	128
192.168.10.0/**26**	192.168.10.0	192.168.10.63	64
192.168.10.0/**27**	192.168.10.0	192.168.10.31	32
192.168.10.0/**28**	192.168.10.0	192.168.10.15	16
192.168.10.0/**29**	192.168.10.0	192.168.10.7	8
192.168.10.0/**30**	192.168.10.0	192.168.10.3	4
*) Note: Note that the network address and the broadcast address are not available as IP addresses of network nodes in an address range.			

IPv6 addresses

IPv6 addresses consist of 8 fields each with four hexadecimal numbers (128 bits in total). The fields are separated by a colon. IPv6 addresses can only be entered in IP packet filter rules for the CP 1543-1 V1.1.

Example: fd00:ffff:ffff:ffff:ffff:ffff:2f33:8f21

Rules / simplifications:

- Leading zeros within a field can be omitted.
  Example: Instead of 2001:0db8:2426:08d3:1457:8a2e:0070:7344 it is also possible to use the notation 2001:db8:2426:8d3:1457:8a2e:70:7344.
- If one or more fields have the value 0 (or 0000), a shortened notation is possible.
  Example: Instead of 2001:0db8:0:0:0:0:1428:57ab it is also possible to use the notation 2001:db8::1428:57ab.
  To ensure uniqueness, this shortened form can only be used once within the entire address.
- Decimal notation with periods
  The last 2 fields or 4 bytes can be written in the normal decimal notation with periods.
  Example: The IPv6 address fd00::ffff.125.1.0.1 is equivalent to fd00::ffff:7d01:1.
- Address range notation in IP packet filter rules: In the same way as IPv4 addresses, IPv6 addresses can also be noted in the form of address ranges.
  Example: The entry "2001:0db8:85a3:08d3:1319:8a2e:0:0/96" covers all IPv6 addresses from 2001:0db8:85a3:08d3:1319:8a2e:0:0 to 2001:0db8:85a3:08d3:1319:8a2e:ffff:ffff.

Carrying out module-specific log settings

Log settings - overview

Log settings in the configuration

The log settings made here are loaded on the module with the configuration and take effect when the security module starts up.

You can restrict the configured packet filter log settings in the online functions if necessary. For example, you can use the online functions to specify that merely IP logging is displayed if you have configured IP and MAC logging.

Logging procedures and event classes

Here, you can specify which data should be logged. As a result, you enable logging as soon as you download the configuration to the security module.

During configuration, you also select one or both of the possible logging procedures:

- Local logging
- Network Syslog

In both logging procedures, the security module recognizes the three following types of events:

- Packet filter events
- Audit events
- System events

Configuring local logging

How to access this function

1. Select the module to be edited.
2. Select the "Log settings" > "Local log memory" entry in the local security settings.

Configuring local logging

Table 8-20 Local logging - settings for log events

Log event	Meaning	Comments
Packet filter log (firewall)	The packet filter log records certain packets of the data traffic. Only those data packets for which a configured packet filter rule (firewall) applies or to which the basic protection reacts (corrupt or invalid packets) are logged. This is only possible when logging is enabled for the packet filter rule. In the "Packets to be logged" drop-down list, you can specify the data packets to be logged: • "All packets": The data packets that are logged are those to which a configured firewall rule applies. In addition to this, all the response packets to such packets are recorded that have passed the firewall according to a configured allow rule. • "Status generating packets": The only data packets that are logged are those to which a configured firewall rule (standard mode or advanced mode) applies and that initially generate a status in the firewall. Data packets that pass through the firewall by using this firewall status are not logged.	Packet filter log data is not retentive The data is stored in volatile memory on the security module and is therefore no longer available after the power supply has been turned off. For retentive storage, you can also save the log data displayed in the "Online & diagnostics" dialog in a file.
Audit log	The logging of audit events is always enabled. The logged information is always stored in the ring buffer. The audit log automatically records security-relevant events, for example user actions such as activating or deactivating packet logging or downloading configurations to the security module.	Audit log data is retentive The data is stored in a retentive memory of the security module and is therefore still available after turning off the power supply. Note for CPs: The audit log files are not retentive on CPs. You should therefore use a Syslog server to backup this data.
System log	The system log automatically logs successive system events, for example the start of a process or the failed login attempt of a user. Select the "Log settings" > "Configure system events" entry to configure the event filter and cable diagnostics functions.	System log data is not retentive The data is stored in volatile memory on the security module and is therefore no longer available after the power supply has been turned off. For retentive storage, you can also save the log data displayed in the "Online & diagnostics" dialog in a file.

Table 8-21 Local logging - storage of recorded data

Storage	Meaning
Ring buffer	At the end of the buffer, the recording continues at the start of the buffer and overwrites the oldest entries.
One-shot buffer	Recording stops when the buffer is full.

Configuring system events

How to access this function

1. Select the module to be edited.
2. Select the "Log settings" > "Configure system events" entry in the local security settings.

Filtering of the system events

In this dialog, you set a filter level for the system events. As default, the following values are set:

- SCALANCE S: Level 3 (Error)
- CP: Level 3 (Error)

The priority of the selected filter level must be the same or lower than the severity set for the cable diagnostics; see the "Setting parameters for line diagnostics" table (not for CPs).

Recommendation: Select "Error" as the filter level or a higher value to exclude logging of general, uncritical events.

Note for CP

For CPs, select only level 3 or level 6 since only events of these levels are generated for CPs.

- Level 0 to level 3 error messages are output if level 3 is selected.
- If you select level 6, the error messages of levels 0 to 6 are output.

Properties of the system events - line diagnostics (not for CPs)

Line diagnostics generates a special system event. A system event is generated when a selectable percentage of bad frames is reached. This system event is assigned the severity and facility set in this dialog.

Table 8-22 Setting parameters for line diagnostics

Function / option / parameter	Meaning
Enable	Enabling and disabling logging
Limit	Selectable percentage of faulty frames representing the limit at which a system event is triggered.

Function / option / parameter	Meaning
Facility	Select a facility from the drop-down list that identifies the system event to be logged.
Severity	Using the severity, you weight the system events of line diagnostics relative to the severity of the other system events.

Note

Severity of the system events of line diagnostics

Make sure that you do not assign a lower severity to the system events of line diagnostics than for the filter. At a lower severity, these events will not pass through the filter and will not be logged.

Configuring the network Syslog

How to access this function

1. Select the module to be edited.
2. Select the "Log settings" > "Network Syslog" entry in the local security settings.

Configuring the network Syslog

Table 8-23 Network Syslog - basic settings

Option / parameter	
Enable network Syslog	Enables and disables the transfer of logging events to the Syslog server.
Syslog server	Here, enter the IP address of the Syslog server. As an alternative, an FQDN can be entered for SCALANCE S V4 modules. The Syslog server must be accessible from the security module at the specified address and, if applicable, using the router configuration under the "Routing" entry of the local security settings. If the Syslog server cannot be accessed no Syslog messages are sent. You can recognize this operating state on the basis of corresponding system alarms. To activate the sending of the Syslog information again, you may have to update the routing information and initiate a restart of the security module.
Module name	The module name is displayed and cannot be changed at this point.

Table 8-24 Network Syslog - settings for log events

Log event	Configuring	Comments
Packet filter events (firewall)	The packet filter log records certain packets of the data traffic. Only those data packets for which a configured packet filter rule (firewall) applies or to which the basic protection reacts (corrupt or invalid packets) are logged. This is only possible if logging is activated for the packet filter rule. Syslog alarms can be classified according to their origin and their severity level by setting Facility and Severity. This assignment is carried out with drop-down lists. The severity and facility that you set here is assigned to every event.	The value you select here depends on the evaluation in the Syslog server. If you retain the "default" value setting, the security module specifies the combination of facility and severity with which the event is displayed.
Audit events	The audit log automatically records security-relevant events, for example user actions such as activating or deactivating packet logging or downloading configurations to the security module. Assignment of the severity and facility is carried out with drop-down lists. The severity and facility that you set here is assigned to every event.	The value you select here for the severity and facility depends on the evaluation in the Syslog server. If you retain the "default" value setting, the security module specifies the combination of facility and severity with which the event is displayed.
System events	The system log automatically logs successive system events, for example the start of a process or the failed login attempt of a user.	Select the "Log settings" > "Configure system events" entry in the local security settings to configure the event filter and cable diagnostics functions.

Security module as router

Overview of the routing settings

Meaning

If you operate the security module in routing mode, the networks connected to the internal and external interface are transformed into separate subnets. The DMZ interface (SCALANCE S623/S627-2M only) is connected in routing mode regardless of the mode. In routing mode, the frames intended for an existing IP address in the subnet are forwarded. The firewall rules for the direction of transmission also apply.

You have the following additional options:

- Setting specific routes - can be set in the local security settings under "Routing" (SCALANCE S only), see Specifying routes (Page 829) in the section "SCALANCE S".
- Use standard router - can be set in the local security settings with "External interface [P1] red", "Internal interface [P2] green" or "DMZ interface [P3] yellow" (SCALANCE S623/ S627-2M only) see Configuring IP address parameters (Page 812) in the section "SCALANCE S".
  Only one standard router can be used per interface.
- NAT/NAPT routing - can be set with "NAT / NAPT" (not for CP 1628 and CP 1543-1) in the local security settings. To be able to use NAT/NAPT, the security module must be in routing mode.

Enabling routing mode (required for SCALANCE S modules only)

For this mode, you configure an internal IP address and an internal subnet mask for addressing the router in the internal subnet in the local security settings. All network requests that do not belong to a subnet are forwarded by the security module to a different subnet.

Note: In contrast to the bridge mode of the security module, VLAN tags are lost in routing mode. VLAN is not supported by SCALANCE S modules. Frames with a VLAN tag are therefore discarded by SCALANCE S modules.

1. Select the "Routing mode" option under "Mode" in the local security settings.
2. In the local security settings, enter an internal IP address and an internal subnet mask for addressing the router on the internal subnet in the input boxes under "Internal interface [P2] green" > Ethernet addresses".

Overview of NAT/NAPT

Module-specific function

This function is not available for the CP 1628 and CP 1543-1.

Requirement

The security module is in routing mode or the DMZ interface is activated (SCALANCE S623 / S627-2M only).

How to access this function

1. Select the module to be edited.
2. In the local security settings, select the entry "NAT / NAPT".
3. When required, enable address translation according to NAT(Network Address Translation) or NAPT (Network Address Port Translation).

Address translation with NAT (Network Address Translation)

NAT is a procedure for address translation between two address spaces.
The main task is to translate private addresses into public addresses; in other words into IP addresses that are used and even routed on the Internet. As a result, the IP addresses of the internal network are not known to the outside in the external network. The internal nodes are only visible to the external network by means of external IP address that is specified in the address translation list (NAT table). If the external IP address is not the address of the security module and if the internal IP address is unique, this is known as 1:1 NAT. With 1:1 NAT, the internal address is translated to this external address without port translation. Otherwise, n:1 NAT is being used.

Address translation with NAPT (Network Address Port Translation)

The address translation with NAPT changes the target address and the target port to a communication relation (port forwarding).

Frames coming from the external network or DMZ network and intended for the IP address of the security module are translated. If the destination port of the frame is identical to one of the values specified in the "Source port" column, the security module replaces the destination IP address and the destination port as specified in the corresponding row of the NAPT table. With the reply, the security module uses the values for the destination IP address and destination port as contained in the initial frame as the source IP address and the source port.

The difference to NAT is that with this protocol ports can also be translated. There is no 1:1 translation of the IP address. There is now only a public IP address that is translated to a series of private IP addresses with the addition of port numbers.

Consistency check - these rules must be adhered to

When assigning addresses, remember the following rules to obtain consistent entries:

Check / rule	Check made	
	locally	project-wide
The internal IP addresses must not be identical to the IP addresses of the module.		x
Use the part specified by the subnet mask for the IP addresses: • IP addresses that you specify for the direction "external" must be in the same subnet range as the external IP address of the security module under the entry "External interface [P1] red". • IP addresses that you specify for the direction "internal" must be in the same subnet range as the internal IP address of the security module under the entry "Internal interface [P2] green". • IP addresses that you specify for the direction "DMZ" must be in the same subnet range as the DMZ IP address of the security module under the entry "DMZ interface [P3] yellow".		x
An IP address used in the NAT/NAPT address conversion list must not be a multicast or broadcast address.		x

Check / rule	Check made	
	locally	project-wide
The external ports assigned for the NAPT translation are in the range > 0 and ≤ 65535. Port 123 (NTP), 443 (HTTPS), 514 (Syslog), 161 (SNMP), 67+68 (DHCP), 500+4500 (IPsec) and 161 (SNMP) are excluded if the relevant services are activated on the security module.	x	
The external IP address of the security module or the IP address of the DMZ interface may only be used in the NAT table for the action "Source NAT".	x	
Checking for duplicates in the NAT table An external IP address or an IP address in the DMZ network used in the direction "Destination NAT", "Source NAT +Destination NAT" or "Double NAT" may only be used once in each specified direction.	x	
Checking for duplicates in the NAPT table: A source port number may only be entered once for each interface.	x	
Internal NAPT ports can be in the range > 0 and ≤ 65535.	x	

See also

Overview of the routing settings (Page 767)

NAT/NAPT routing

Enabling NAT

The input boxes for NAT are enabled. NAT address translations only take effect with the entries in the address translation list described below. After creating NAT rules, the corresponding firewall rules are generated and displayed in advanced firewall mode, see section:

Relationship between NAT/NAPT router and firewall (Page 775)

If PPPoE is activated for the external interface or the DMZ interface, the action "Destination NAT" cannot be configured. When configuring the action "Source NAT", the IP address cannot be entered in the "Source translation" input box because this is obtained dynamically during runtime.

Possible address translations for NAT

The following tables show the input options for address translation with NAT.

Action "Destination NAT" - "Redirect"

The action "Destination NAT" can be performed in the following direction:

- External to internal

If the DMZ interface of the security module (SCALANCE S623/S627-2M only) is activated, the action "Destination NAT" can also be performed in the following directions:

- External to DMZ
- DMZ to internal
- DMZ to external

The following applies, for example for the direction "external to internal": The destination IP address of a frame coming from the external network is checked to see whether it matches the IP address specified in the "Destination IP address" input box. If it matches, the frame is forwarded into the internal network by replacing the destination IP address of the frame with the IP address specified in the "Destination translation" input box. Access from external to internal using the external address is possible.

The following table shows the input required for the action "Destination NAT".

Box	Possible entries	Meaning
Source IP address	Not relevant for this action.	-
Source translation	Not relevant for this action.	-
Destination IP address	IP address in the source network	Destination IP address in the source network with which an IP address in the destination network will be accessed. If the destination IP address in a frame matches the address entered, the address is replaced by the corresponding IP address in the destination network. If the IP address entered here is not the IP address of the security module, this becomes an alias address. This means that the specified IP address is also registered as an IP address on the selected interface. Make sure that the alias address does not cause an IP address conflict in the network.
Destination translation	IP address in the destination network	The destination IP address is replaced by the IP address specified here.
No.	-	Consecutive number assigned by STEP 7 used to reference the firewall rule generated by STEP 7 for the NAT rule.

Action "Source NAT" - "Masquerading"

The action "Source NAT" can be performed in the following direction:

- Internal to external

If the DMZ interface of the security module (SCALANCE S623/S627-2M only) is activated, the action "Source NAT" can also be performed in the following directions:

- Internal to DMZ
- External to DMZ
- DMZ to external

The following applies, for example for the direction "internal to external": The source IP address of a frame coming from the internal network is checked to see whether it matches the IP address specified in the "Source IP address" input box. If it matches, the frame with the external IP address specified in the "Source translation" input box is forwarded to the external network as a new source IP address. In the external network, the external IP address is effective.

The following table shows the input required for the action "Source NAT".

Box	Possible entries	Meaning
Source IP address	IP address in the source network	The source IP address of the specified node is replaced by the IP address specified in the "Source translation" input box.
	IP address range in the source network	The IP address range is replaced by the IP address specified in the "Source translation" input box.
	*	The IP addresses of all nodes in the source network are replaced by the IP address specified in the "Source translation" input box.
Source translation	IP address in the destination network	Entry of the IP address that will be used as the new source IP address. If the IP address entered here is not the IP address of the security module, this becomes an alias address. This means that the specified address is also registered as an IP address on the selected interface. Make sure that the alias address does not cause an IP address conflict in the network.
Destination IP address	Not relevant for this action.	Not relevant for this action.
Destination translation	Not relevant for this action.	Not relevant for this action.
No.	-	Consecutive number assigned by STEP 7 used to reference the firewall rule generated by STEP 7 for the NAT rule.

You can configure an address translation to the module IP address in the destination network for all frames going from a source network to a destination network. The security module also assigns a port number for each frame. This is an n:1 NAT address translation in which multiple IP addresses of the source network are translated to one IP address of the destination network.

Enter, for example, the following parameters for the direction "internal to external":

- Action: "Source NAT"
- From: "Internal"
- To "External"
- Source IP address: "*"
- Source translation: External IP address of the security module

Action "Source NAT + Destination NAT" - "1:1-NAT"

The action "Source NAT + Destination NAT" can be performed in the following direction:

- Internal to external

If the DMZ interface of the security module (SCALANCE S623/S627-2M only) is activated, the action "Source NAT + Destination" can also be performed in the following directions:

- Internal to DMZ
- External to DMZ
- DMZ to external

The following applies, for example for the direction "internal to external": When accessing from internal to external, the action "Source NAT" is performed. When accessing from external to internal, the action "Destination NAT" is performed.

The following table shows the input required for the action "Source NAT + Destination NAT":

Box	Possible entries	Meaning
Source IP address	IP address in the source network	The configuration is always specified in the source NAT direction. The IP addresses of the destination NAT direction are then inserted automatically by STEP 7.
Source translation	IP address in the destination network	
Destination IP address	Not relevant for this action.	
Destination translation	Not relevant for this action.	
No.	-	Consecutive number assigned by STEP 7 used to reference the firewall rules generated by STEP 7 for the NAT rule.

Action "Double NAT"

The action "Double NAT" can be performed in the following directions:

- Internal to external
- External to internal

If the DMZ interface of the security module (SCALANCE S623/S627-2M only) is activated, the action "Double NAT" can also be performed in the following directions:

- Internal to DMZ
- External to DMZ
- DMZ to internal
- DMZ to external

In every direction, Source and Destination NAT always take place at the same time.
The following applies, for example for the direction "external to internal": When accessing from external to internal, the source IP address of the external node is replaced (Source NAT). Access to the internal network also uses the external IP address specified in the "Destination IP address" input box (Destination NAT).

You can, for example, use this action if a standard router other than the security module is entered for a device to be accessed using Destination NAT. Response frames from this device are then not sent to the entered standard router but to the corresponding interface of the security module.

The following table shows the input required for the action "Double NAT":

Box	Possible entries	Meaning
Source IP address	IP address in the source network	IP address of the node in the source network
Source translation	-	The Source NAT address translation is always to the IP address of the security module in the destination network. For this reason, the "Source translation" input box cannot be configured.
Destination IP address	IP address in the source network	Destination IP address in the source network with which an IP address in the destination network will be accessed. If the destination IP address in a frame matches the IP address entered, the IP address is replaced by the IP address specified in the "Destination translation" input box. If the IP address entered here is not the IP address of the security module, this becomes an alias address. This means that the specified address is also registered as an IP address on the selected interface. Make sure that the alias address does not cause an IP address conflict in the network.
Destination translation	IP address in the destination network	The destination IP address is replaced by the IP address specified here.
No.	-	Consecutive number assigned by STEP 7 used to reference the firewall rule generated by STEP 7 for the NAT rule.

Enabling NAPT

The input boxes for NAPT are enabled. NAPT translations only take effect with the entries in the list described below. After creating NAPT rules, the corresponding firewall rules are generated and displayed in advanced firewall mode, see section:
Relationship between NAT/NAPT router and firewall (Page 775)

The IP address translation with NAPT can be performed in the following direction:

- External to internal

If the DMZ interface of the security module (SCALANCE S623/S627-2M only) is activated, the IP address translation with NAPT can also be performed in the following directions:

- External to DMZ
- DMZ to internal
- DMZ to external

The following applies, for example for the direction "external to internal": Frames intended for the external IP address of the security module and for the port entered in the "Source port"

column are forwarded to the specified destination IP address in the internal network and to the specified destination port.

The following table shows the input required for address translation with NAPT:

Box	Possible entries	Meaning
Source port	TCP/UDP port or port range Example of entering a port range: 78:99	A node in the source network can send a frame to a node in the destination network by using this port number.
Destination IP address	IP address in the destination network	Frames intended for the IP address of the security module in the source network and the TCP/UDP port specified in the "Source port" box are forwarded to the IP address specified here.
Destination port	TCP/UDP port	Port number to which the frames coming from the source network are forwarded.
Protocol	• TCP+UDP • TCP • UDP	Selection of the protocol family for the specified port numbers
No.	-	Consecutive number assigned by STEP 7 used to reference the firewall rule generated by STEP 7 for the NAPT rule.

Relationship between NAT/NAPT router and firewall

Meaning

After creating NAT/NAPT rules, STEP 7 automatically generates firewall rules that enable communication in the configured address translation direction. The generated firewall rules are visible in advanced firewall mode and can, if necessary, be moved and expanded (additional IP address, services, bandwidth). In addition to this, the generated firewall rules should be checked in terms of their priority and their position. If necessary, the generated firewall rules can be moved. Firewall parameters generated by STEP 7 cannot be adapted. After deactivating NAT/NAPT, the firewall rules generated by 7 are disabled.

To clarify the relationship between the NAT/NAPT rules and the corresponding firewall rules, they are identified by corresponding, consecutive numbers.

The following table shows the system behind the firewall rules generated for NAT rules.

Table 8-25 NAT address translation and corresponding firewall rules

NAT action	Created firewall rule				
	Action	From	To	Source IP address	Destination IP address
Source NAT	Allow	Source network	Destination network	IP address of the node specified in the "Source IP address" input box	-
Destination NAT	Allow	Source network	Destination network	-	IP address of the node specified in the "Destination translation" input box

NAT action	Created firewall rule				
	Action	From	To	Source IP address	Destination IP address
Source NAT + Destination NAT	Allow	Source network	Destination network	IP address of the node specified in the "Source IP address" input box	-
	Allow	Destination network	Source network	-	IP address of the node that was inserted in the "Destination translation" input box by STEP 7
Double NAT	Allow	Source network	Destination network	IP address of the node specified in the "Source IP address" input box	IP address of the node specified in the "Destination translation" input box

The following table shows the system behind the firewall rules generated for NAT rules.

Table 8-26 NAPT translations and created firewall rules

Created firewall rule					
Action	From	To	Source IP address	Destination IP address	Service
Allow	Source network	Destination network	-	IP address of the node specified in the "Destination IP address" input box	[Portnumber_protocol]

Configuring time-of-day synchronization

Overview of time-of-day synchronization

Meaning

The date and time are maintained on the security module to verify the validity (time) of a certificate and for time stamping log entries. For CPs, time-of-day synchronization is active as default.

The following alternatives can be configured:

- Set time of day when loading (only for SCALANCE S): The module time is set automatically to the PC time when a configuration is downloaded.
- SIMATIC (for CPs only): If the security module receives MMS time-of-day messages, its local time is synchronized provided the NTP procedure was not configured (MMS = Manufacturing Message Specification).
- NTP: Automatic setting and cyclic synchronization of the time-of-day by means of Network Time Protocol server.

Note

Time-of-day synchronization relates solely to the security module and cannot be used to synchronize devices in the internal network of the security module. S7 CPs can forward the time of day to other modules of the station.

Note

Configuring the firewall for communication with NTP servers

If the NTP server cannot be reached by the security module, you will need to allow the frames from the NTP server explicitly in the firewall (UDP, port 123).

Defining an NTP server

Creating NTP servers in the global security settings

In the global security settings only NTP servers of the type "NTP (secure)" can be created and assigned CPs or SCALANCE S V4 modules. Non-secure NTP servers for SCALANCE S modules < V4 and CPs therefore need to be created in the local security settings.

1. Double-click the "NTP" entry in the global security settings.
2. Double-click on the "Add new NTP server" entry.
3. Enter a name for the NTP server (secure).
4. Enter the IP address of the NTP server (secure). If the NTP server will only be assigned SCALANCE S V4 modules, an FQDN can be specified as an alternative.
5. Specify the encryption parameters for the NTP server (secure).

Property	Meaning
Key ID	Numeric value between 1 and 65534.
Authentication	Select the authentication algorithm.
Hex/ASCII	Select the format for the NTP key.
Key	Enter the NTP key with the following lengths: Hex: 22 ... 40 characters ASCII: 11 ... 20 characters

6. Assign a the required security module to the NTP server (secure) you have created, see section:
   Assigning the security module to an NTP server (secure) (Page 781).

Creating NTP servers in the local security settings

1. Select the module to be edited.
2. Select the "Time-of-day synchronization" entry in the local security settings.
3. Select the required synchronization mode.
4. Enter a name and the IP address of the NTP server. If you have selected the "NTP (secure)" synchronization mode, you can select an NTP server (secure) that you created in the global security settings in the "Name" column.

Configuration limits for NTP servers

The following rules apply when creating NTP servers:

- You can create 32 NTP servers throughout the project.
- You can assign a maximum of 4 NTP servers to one security module.

Importing/exporting NTP servers (secure)

Using the "Import" or "Export" commands in the shortcut menu, you can export the key list of the currently selected NTP server (secure) and import the file into an NTP server (secure) or vice versa.

Configuring time-of-day synchronization for a security module

How to access this function

1. Select the module to be edited.
2. Select the "Time-of-day synchronization" entry in the local security settings.
3. Select the "Activate time-of-day synchronization" check box.

Alternatives for time-of-day synchronization

The following alternatives can be configured:

Table 8-27 Time synchronization for CP and SCALANCE S

Possible selection	Meaning / effect
Set time of day when loading (only for SCALANCE S)	The module time is set automatically to the PC time when a configuration is downloaded.
SIMATIC (not for SCALANCE S)	If the security module receives MMS time-of-day messages, its local time is synchronized provided the NTP procedure was not configured (MMS = Manufacturing Message Specification).
NTP (not for SCALANCE S ≥ V3)	Automatic setting and periodic synchronization of the time using an NTP server.
NTP (secure) (not for SCALANCE S < V4)	Automatic setting and periodic synchronization of the time using an NTP server (secure).

Selecting the mode of time-of-day synchronization

Follow these steps:

1. Select the synchronization mode.
2. The following configuration options are available to you depending on the selected mode:

- **Set time of day when loading:** When a configuration is downloaded to a SCALANCE S module, the module time of day is set with the PC time of day.
- **SIMATIC:** For the CP x43-1 select whether the CP is to adopt or forward the time of day. The CP 1628 always forwards the time of day. For CPs you also the set direction for forwarding the time.
  Available directions:
  - Automatic (only for CP x43-1): The CP receives the time from the station or from the LAN and forwards it to the station or to the LAN. If multiple CPs are being operated in the station, this automatic setting can cause collisions. To avoid this, you can specifically define the forwarding direction.
  - From station (for CP x43-1 only).
  - From LAN.

  If forwarding of the time of day is enabled, you can use the "Use corrected time" check box to specify whether or not a correction factor included in the time-of-day frame is used. For the CP 1628, this option is enabled as default and cannot be disabled.
- **NTP:**
  - Time zone: In NTP mode, generally UTC (Universal Time Coordinated) is transmitted. This corresponds to GMT (Greenwich Mean Time). The time offset from UTC can be set by configuring the local time zone.
  - Update interval in seconds: Defines the interval between the time queries in seconds. For SCALANCE S ≥ V3, the time interval for querying the NTP server is specified automatically.

    Note
    Setting the update interval for CPs

    If the "Enable security functions" check box is enabled in the local security settings of a CP, the setting for the update interval is transferred from the CP's local settings into the CP's local security settings.

  - Time-of-day synchronization on the full minute (for CP x43-1/CP 1628 only): With this option, you can decide whether or not the time of day is forwarded to the communications bus on the full minute. This option is required for certain special applications.
  - Accept time of non-synchronized NTP servers (only for CP x43-1/CP 1628/CP 1543-1): Here you can specify whether the security module also accepts the time-of-day from non-synchronized NTP servers.
  - Forward time of day to station (only for CP x43-1/CP 1628): Disable this option if the CPU requests the time separately from an NTP server. This prevents the time on the CPU obtained directly from the NTP server from being overwritten by the time detected in the CP. The accuracy may be reduced slightly due to forwarding via the CP.
  - NTP server: Creating NTP servers in the local security settings is described in the section Defining an NTP server (Page 777).

Assigning the security module to an NTP server (secure)

Requirement

- You have defined an NTP server (secure) in the global security settings.
- "NTP" or "NTP (secure)" is selected as the time-of-day synchronization mode in the local security settings of the security module that you want to assign to an NTP server (secure).

Procedure

1. Double-click the "NTP" entry in the global security settings.
2. Double-click the entry "Assign module to an NTP server".
3. From the "NTP Server" drop-down list, select the NTP server (secure) to which you want to assign a security module.
4. In the "Available modules" section, select the security module that you want to assign to the selected NTP server (secure).
5. Click the << button to assign the selected security module to the selected NTP server (secure).

Result

You have assigned the security module to the NTP server (secure). The NTP server (secure) is displayed automatically in the local security settings in the list of NTP servers.

Security module as DHCP server

Module-specific function

The use of the security module as a DHCP server is only possible with SCALANCE S modules, see subsection:
Auto-Hotspot in the section "SCALANCE S".

Configuring SNMP

Overview of SNMP

What is SNMP?

The security module supports the transfer of management information using the Simple Network Management Protocol (SNMP). For this purpose, an SNMP agent that receives and responds to SNMP requests is installed on the security module. The information on the properties of SNMPcompliant devices is entered in MIB files (MIB = Management Information Base) for which the user must have the required rights.

In SNMPv1, the "community string" is also sent. The "community string" is like a password that is transmitted along with the SNMP request. If the community string is correct, the security module replies with the required information. If the string is incorrect, the security module discards the query and does not reply. The community string is transmitted via SNMPv1 without encryption.

SNMPv3 lets you transmit encrypted data.

Configuring SNMP - "SNMP" entry

Module-specific function

The configuration of SNMP is possible only for SCALANCE S V3 or higher, S7 CPs and the CP 1628.

How to access this function

1. Select the module to be edited.
2. In the local security settings, select the entry "SNMP".
3. Activate the "Activate SNMP" check box.
4. Select one of the following SNMP protocol versions:

   Note
   Encrypted data transmission with SNMPv3
   You should use SNMPv3 to transmit data in encrypted form in order to enhance security.

   – SNMPv1
     The security module uses the following default values for the community strings to control the access rights in the SNMP agent. These default values should be adapted to increase security.
     For read access: public
     For read and write access: private
     To enable write access using SNMP, select the "Allow write access" option.
   – SNMPv3
     Select either only an authentication algorithm or an authentication algorithm and an encryption algorithm.
     Authentication algorithm: none, MD5, SHA-1
     Encryption algorithm: none, AES-128, DES

     Note
     Preventing the use of DES
     DES is an insecure encryption algorithm. Therefore, it should only be used for reasons of down compatibility.

5. If SNMPv3 is to be used, assign a user a role with corresponding activated SNMP rights to enable access to the module via SNMP. An overview of SNMP rights is available in the section
Managing rights (Page 740).
6. In the "Advanced settings" area, configure module-specific information about the author, location and e-mail address overwriting the information from the project properties.
The following applies to values written to the security module by an SNMP tool using an SNMP-SET command:
If you select the "Keep values written by SNMP set" check box, the values are not overwritten when there is another download of an SCT configuration to the security module.

Configuring proxy ARP

Module-specific function

This function is available only for SCALANCE S V3 modules or higher, see section Configuring proxy ARP (Page 833)

Activate Web server on security module

Module-specific function

This function is only available for CP x43-1 Advanced, see subsection:
Activating the Web server on CP x43-1 Advanced (Page 854) in the section "Security for S7-300 / S7-400 / PC CPs".

IPsec tunnel: Creating and assigning VPN groups

How to create an IPsec tunnel with VPN groups

Module-specific function

This function is not available for SCALANCE S602 and CP 1543-1 V1.0. CP 1543-1 supports this function only as of firmware V1.1.

Requirement

Note

Current date and current time of day on the security modules

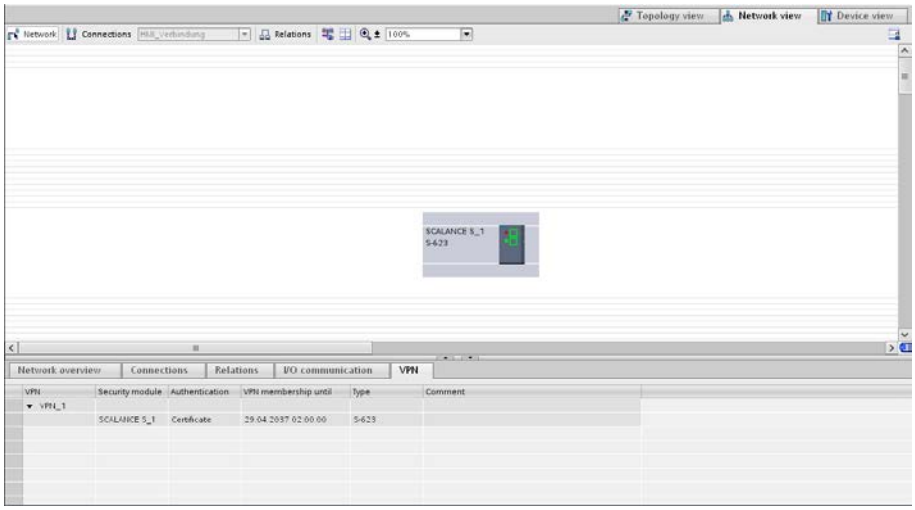
When using secure communication (for example, HTTPS, VPN...), make sure that the security modules involved have the current time of day and the current date. Otherwise the certificates used are not evaluated as valid and the secure communication does not work.

How to access this function

1. Double-click on the entry "VPN groups" > "Add new VPN group" in the global security settings to create a VPN group.
2. Double-click on the entry "VPN groups" > "Assign module to a VPN group" in the global security settings and assign the security modules and SOFTNET security client modules to the VPN group between which VPN tunnels will be established. Here, note the rules for forming VPN groups. You will also find these rules in the section:
   Modes of VPN groups (Page 788).

Display of the VPN groups with their properties

If you select a security module that is in one or more VPN group(s), the properties of the VPN group(s) of which the security module is a member are shown in the "Network data" area.



The following properties of the VPN groups are displayed in columns in the "VPN" tab of the "Network data" area:

Property/column	Meaning
VPN	Names of the VPN groups of which the selected security module is member
Security module	Names of the assigned security modules
Authentication	Type of authentication: Preshared key or certificate
Group membership until	Date and time up to which the VPN group certificate of the security module is valid
Type	Model numbers of the assigned security modules
Comment	Comment

Setting the life of certificates

Open the dialog in which you can set the expiry date of the certificate as follows:

1. Select the VPN group you want to edit in the "VPN" tab.
2. In the "Properties" > "General" tab of the Inspector window, select the entry "Authentication".

Note

Expiry of a certificate

Communication through the VPN tunnel continues after the certificate has expired until the tunnel is terminated or the SA lifetime expires. For more information on certificates, refer to section:
Auto-Hotspot.

Quantity structure

Number of IPSec tunnels	
SCALANCE S612 V2	Maximum of 64
SCALANCE S612 V3/V4	Maximum of 128
SCALANCE S613	Maximum of 128
SCALANCE S623 V3/V4	Maximum of 128
SCALANCE S627-2M V4	Maximum of 128
CP x43-1 Advanced	Maximum of 32
CP 1628	Maximum of 64
CP 1543-1 V1.1	Maximum of 16

Authentication methods

The following methods are available:

The authentication method is specified per VPN group and decides the type of authentication used.

The following key-based or certificate-based authentication methods are supported:

- Preshared key
  Authentication is achieved using a previously specified character string that is distributed to all modules in the group.
  Enter a preshared key in the "Key" field under "Authentication" > "General" in the VPN group properties dialog.
- Certificate
  Certificate-based authentication "Certificate" is the default setting. The procedure is as follows:
  – When a group is created, a CA certificate is generated as a root certificate.
  – Each security module that is a member of the VPN group also receives a group certificate signed with the key of the CA certificate.

  All certificates are based on the ITU standard X.509v3 (ITU, International Telecommunications Union).
  The certificates are generated by a certificate authority contained in STEP 7.

Note

Restriction in VLAN operation

No VLAN tagging is transmitted in IP frames via the VPN tunnel of the security module. The VLAN tags included in IP frames are lost when they pass through the security modules because IPsec is used to transfer the IP frames.

As default, no IP broadcast or IP multicast frames can be transferred with IPsec through a layer 3 VPN tunnel. Through a layer 2 VPN tunnel of the security module, IP broadcast or IP multicast frames are packaged just like MAC packets including the Ethernet header in UDP and transferred. With these packets, the VLAN tagging is therefore retained.

Group properties for selected VPN group

VPN group properties

Note

Knowledge of IPsec necessary

To be able to set these parameters, you require IPsec experience. If you do not make or modify any settings, the defaults apply.

The following settings can be configured in the properties of a VPN group:

- Authentication method (entry: "General")
- IKE settings (entry: "Advanced settings phase 1")
- IPsec settings (entry: "Advanced settings phase 2")

How to access this function

1. In the global security settings select the VPN group whose properties you want to configure under the entry "VPN groups".
2. Select "Open" in the shortcut menu of this entry.
   Result: The VPN group properties appear in the area of the local security settings.
3. In the "Authentication" entry, select whether to use a preshared key or certificate for authentication. For more detailed information, refer to section:
   Authentication methods (Page 785).

Parameters for advanced settings phase 1 - IKE settings

Phase 1: Key exchange (IKE = Internet Key Exchange):

Here, you set the parameters for the protocol of the IPsec key management. Key exchange is handled by means of the standardized IKEv1 method for which you can set the following protocol parameters:

Parameter	Description
IKE mode	Key exchange method: • Main mode • Aggressive mode The difference between the main and aggressive mode is the "identity protection" used in the main mode. The identity is transferred encrypted in main mode but not in aggressive mode.
DH group phase 1	Diffie-Hellman key agreement: • Group 1 • Group 2 • Group 5 • Group 14 Diffie-Hellman groups (selectable cryptographic algorithms in the Oakley key exchange protocol).
SA lifetime type	Phase 1 Security Association (SA): • Time: Time limit in minutes The lifetime of the current key material is limited in time. When the time expires, the key material is renegotiated.
SA lifetime	Numeric value: Range of values for time: 1440 ... 2500000 minutes (default: 2500000)
Encryption phase 1	Encryption algorithm: • DES*: Data Encryption Standard (56 bit key length, mode CBC) • 3DES-168: Triple DES (168-bit key length, mode CBC) • AES-128, 192, 256: Advanced Encryption Standard (128 bit, 192 bit or 256 bit key length, mode CBC)
Authentication phase 1	Authentication algorithm: • MD5: Message Digest Algorithm 5 • SHA1: Secure Hash Algorithm 1

*DES is an insecure encryption algorithm. It should only be used for reasons of down compatibility. DES is not supported by the CP 1543-1 V1.1.

Parameters for advanced settings phase 2 - IPsec settings

Phase 2: Data exchange (ESP = Encapsulating Security Payload)

Here, you set the parameters for the protocol of the IPsec data exchange. The data exchange is in "quick mode". The entire communication during this phase is encrypted using the standardized security protocol ESP for which you can set the following protocol parameters:

Parameter	Description
SA lifetime type	Phase 2 Security Association (SA): • Time: Time limit in minutes. The lifetime of the current key material is limited in time. When the time expires, the key material is renegotiated. • Limit: Limitation of the data volume in MB
SA lifetime	Numeric value: • Range of values for time: 60 ... 16666666 minutes (default: 2880) • Range of values for limit: 2000 ... 500000 MB (default: 4000)
Encryption phase 2	Encryption algorithm: • DES*: Data Encryption Standard (56 bit key length, mode CBC) • 3DES-168: Triple DES (168-bit key length, mode CBC) • AES-128: Advanced Encryption Standard (128-bit key length, mode CBC)
Authentication phase 2	Authentication algorithm: • MD5: Message Digest Algorithm 5 • SHA1: Secure Hash Algorithm 1
Perfect Forward Secrecy	Select whether or not before each time an IPsec-SA is renegotiated, the key is negotiated again using the Diffie-Hellman method. Perfect Forward Secrecy safely prevents the new key from being derived from keys generated previously.

*DES is an insecure encryption algorithm. It should only be used for reasons of down compatibility. DES is not supported by the CP 1543-1 V1.1.

Modes of VPN groups

VPN modes

Depending on the mode of the security modules that were added to a VPN group, different modes of VPN groups are distinguished. The mode of a VPN group provides information about the security modules and their modes that can be added to the VPN group.

Rules for forming groups

Remember the following rules if you want to create VPN groups:

- For SCALANCE S612 / S613 / S623 / S627-2M / SCALANCE M
  The first module assigned in a VPN group decides which other modules can be added to it.
  If the first SCALANCE S module to be added is in routing mode or if the first security module is a SCALANCE M module, only additional SCALANCE S modules with routing activated or SCALANCE M modules can be added because SCALANCE M modules always operate in routing mode. If the first SCALANCE S module added is in bridge mode, you can only add other SCALANCE S modules in bridge mode. If you want to change the mode of a VPN group, you have to remove all the modules contained in the group and add them again. A CP and an SSC can be added to a group with a SCALANCE S in bridge or routing mode.
- For CP / SSC
  If a CP / SSC is the first module in a VPN group, security modules can be added in any mode. The next security module that decides the mode then also decides the mode of the VPN group. A CP / SSC can be assigned to several VPN groups at the same time and use different modes. The CP / SSC is then operated in mixed mode.
- It is not possible to add a SCALANCE M module to a VPN group that contains a module in bridge mode.

Refer to the following table to see which modules can grouped together in a VPN group:

Table 8-28 Security modules and VPN modes

Module	Can be included in a VPN group in...	
	Bridge mode	Routing mode
SCALANCE S612 / S613 / S623 / S627-2M in bridge mode	x	-*
SCALANCE S612 / S613 / S623 / S627-2M in routing mode	-	x
CP x43-1 Adv.	x	x
CP 1543-1 V1.1	x	x
CP 1628	x	x
SOFTNET Security Client V4.0	x	x
SCALANCE M875	-	x

* SCALANCE S623/S627-2M modules in bridge mode can be inserted in a VPN group in routing mode if their DMZ interface is activated (not at the same time).

Including security module in configured VPN group

The configured VPN group properties are adopted for security modules added to in an existing VPN group.

Procedure after including a security module in a configured VPN group

Depending on whether or not you have changed the VPN group properties since the last download, you must make a distinction between the following:

- Case a: If you have not changed the VPN group properties and the module to be added actively establishes the connection to the configured modules:

1. Add the new security module to the VPN group.
2. Download the configuration to the new module.

- Case b:If you have changed the VPN group properties or the module being added does not actively establish the connection to the already configured modules:

1. Add the new security module to the VPN group.
2. Download the configuration to all modules that belong to the VPN group.

In case a, it is not necessary to reconfigure and load the already commissioned security modules. Active communication is not influenced or interrupted.

Settings for nodes with an unknown IP address

Nodes whose IP address is unknown at the time of configuration (unknown peers) can be inserted in an existing VPN group. Since the nodes are usually mobile and obtain their IP addresses dynamically (for example a SOFTNET security client or SCALANCE M), the VPN tunnel can only be established if you set the parameters for Phase 1 according to one of the following tables ("Encryption parameter 1 to Encryption parameter 4). If you use other settings, you cannot establish a VPN tunnel to the end device.

Table 8-29 Encryption parameter 1

Parameter	Setting
Authentication method	Certificate
DH group phase 1	Group 2
SA lifetime	1440 … 2500000 minutes
Encryption phase 1	AES-256
Authentication phase 1	SHA-1

Table 8-30 Encryption parameter 2

Parameter	Setting
Authentication method	Certificate
DH group phase 1	Group 2
SA lifetime	1440 … 2500000 minutes
Encryption phase 1	3DES-168
Authentication phase 1	SHA-1

Table 8-31 Encryption parameter 3

Parameter	Setting
Authentication method	Certificate
DH group phase 1	Group 2
SA lifetime	1440 … 2500000 minutes
Encryption phase 1	DES
Authentication phase 1	MD5

Table 8-32 Encryption parameter 4

Parameter	Setting
Authentication method	Preshared key
DH group phase 1	Group 2
SA lifetime	1440 … 2500000 minutes
Encryption phase 1	3DES-168
Authentication phase 1	SHA1

Additional restrictions for the SOFTNET Security Client

For the SOFTNET security client, the following restrictions also apply:

Table 8-33 Encryption parameters for the SOFTNET Security Client

Parameter	Setting / special feature
Encryption phase 1	AES-256 possible only with Windows 7
SA lifetime phase 1	1440 to 2879 minutes
SA lifetime type	Must be selected identical for both phases
Encryption phase 2	No AES 128 possible
SA lifetime phase 2	60 to 2879 minutes
Authentication phase 2	No MD5 possible

Procedure after removing an active member from a VPN group

If you remove an active node from an existing VPN group, this can still establish a connection to the group members even if you have downloaded the project to all members of the VPN group again.

If you do not want the removed active node to be able to establish the connection any longer, renew the CA group certificate and download the project again to the members of the VPN group. The certificate can be renewed in the group properties of the VPN group or in the "CA" tab of Certificate Manager.

Configuring internal network nodes

Overview of configuring internal network nodes

Configuring internal network nodes

Each security module must know the network nodes in the entire internal network to be able to recognize the authenticity of a frame.

The security module must know both its own internal nodes as well as the internal nodes of the security modules in the same VPN group. This information is used on a security module to decide which data packet will be transferred in which tunnel.

By adding a security module to a VPN group, the local internal nodes/subnets of the security module are automatically made known to the security module. To allow communication via the VPN tunnel with other subnets or nodes of another subnet, these subnets or nodes need to be enabled for VPN tunnel communication in the configuration.

SCALANCE S modules allow network nodes to be learned automatically or configured statically. The mode of the security module also decides the options available for learning internal network nodes.

SCALANCE S in bridge mode

In bridge mode, you can configure the internal IP/MAC nodes and the internal subnets or alternatively can allow automatic learning of internal nodes by the SCALANCE S.

SCALANCE S in routing mode

In routing mode there is no automatic learning mode available. Instead, enter entire subnets here that need to be released for tunnel communication.

CP x43-1 Advanced and CP 1628

- CP x43-1 Adv.
  Decide whether or not tunnel communication to the CP (Gbit interface) and/or to the internal subnet (PROFINET subnet) is permitted for VPN connection partners in routing mode (SCALANCE S / M).
- CP 1628
  Enter the NDIS nodes you want to be reachable through the tunnel of VPN connection partners in routing mode (SCALANCE S / M).

Automatic learning of internal network nodes

Module-specific function

SCALANCE S modules in bridge mode provide a learning mode with which the internal network nodes can be learned automatically during operation. For more detailed information, refer to

subsection:
Using the learning mode to learn internal nodes (Page 833) in the section "SCALANCE S".

Configuring IP network nodes manually for SCALANCE S

Module-specific function

How to configure IP network nodes for SCALANCE S modules manually is explained in:
Configuring IP network nodes manually (Page 835) in the section "SCALANCE S".

Configuring MAC network nodes manually for SCALANCE S

Module-specific function

How to configure MAC network nodes for SCALANCE S modules manually is explained in
Configuring MAC network nodes manually (Page 835) in the section "SCALANCE S".

Configuring internal subnets for SCALANCE S manually

Module-specific function

How to configure the internal subnets for SCALANCE S modules is explained in
Configuring internal subnets manually (Page 836) in the section "SCALANCE S".

Allow access to S7-300 / S7-400 CPs for VPN connection partners

Module-specific function

How to allow access to S7-300 / S7-400 CPs for VPN connection partners is explained in
Allow access to S7-300 / S7-400 CPs for VPN connection partners (Page 854) in the section "Security for S7-300 /S7-400 /PC CPs".

Configuring NDIS nodes for PC CPs that can be reached through the tunnel

Module-specific function

How you configure NDIS nodes for PC CPs that can be reached through the tunnel is explained in
Configuring NDIS nodes manually for PC CPs that can be reached through the tunnel (Page 854) in the section "Security for S7-300 / S7-400 / PC CPs".

Configuring module- and connection-specific VPN settings

Requirement

The module is a member of a VPN group.

Module- and connection-specific settings

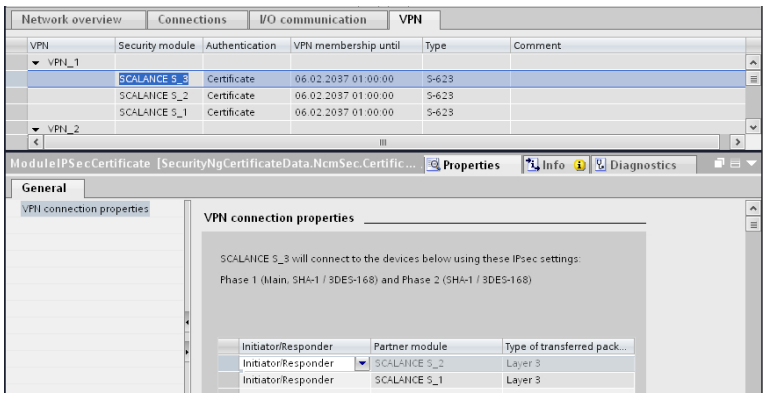
With module- and connection-specific settings, you can configure specific VPN settings. Module-specific settings are configured specifically for a security module while connection-specific settings are configured specifically for a security module in a certain VPN group.

You can configure the following **module-specific** properties in the local security settings with the "VPN" entry:

- Dead peer detection
- Permission to initiate connection establishment
- Public address (IP address / FQDN) for communication via Internet gateways
- Nodes that need to be enabled for tunnel communication

If you select a security module in the list of VPN groups in the "Network data" area, the following **connection-specific** VPN settings can be seen and can be configured:

- Permission to initiate connection establishment
- Partner modules to which tunnel connections exist
- Type of transferred packets
- Selection of the local interface of the selected security module that will serve as the tunnel endpoint
- Selection of the partner interface that will serve as the tunnel endpoint



Dead peer detection (DPD)

As default, DPD is enabled.

When DPD is enabled, the modules exchange additional messages at selectable intervals if no communication occurs at these points in time. This means that it is possible to recognize

whether the IPsec connection is still valid or possibly needs to be re-established. If there is no longer a connection, the security associations (SA) of phase 2 are terminated prematurely. If DPD is disabled, the SA is ended only after the SA lifetime has expired. For information on setting the SA lifetime, see section
Group properties for selected VPN group (Page 786).

Permission to initiate connection establishment

You can restrict the permission for initiating the VPN connection establishment to certain modules in the VPN.

The decisive factor the setting of the parameter described is the assignment of the IP address for the gateway of the module you are configuring. If a static IP address is assigned, the module can be found by the partner. If the IP address is assigned dynamically and therefore changes constantly, the partner cannot establish a connection as things stand.

Mode	Meaning
Start connection to partner (initiator/ responder) (default)	If this option is selected, the module is "active", in other words, it attempts to establish a connection to the partner with a fixed IP address. This option is recommended when you obtain a dynamic IP address from the provider for the gateway of the security module you are configuring. The partner is addressed over its external module IP address or its configured WAP IP address.
Wait for partner (responder)	If this option is selected, the module is "passive", in other words, it waits for the partner to initiate the connection. This option is recommended when you have been assigned a static IP address by the provider for the gateway of the security module you are configuring. It means attempts to establish a connection can only be initiated by the partner. This partner can, for example, have a dynamic WAN IP address.

Note

Make sure that you do not set all the modules in a VPN group to "Wait for connection from remote VPN gateway" otherwise no connection is established.

WAN IP address / FQDN - IP addresses of the modules and gateways in a VPN over Internet

When operating a VPN with IPsec tunnel over the Internet, additional IP addresses are generally required for the Internet gateways such as DSL routers. The individual security or

SCALANCE M modules must know the external IP addresses of the partner modules in the VPN.

Note

To use a WAN as an external public network, enter the IP address that you received from the provider as the external IP address, through which the security module is then reachable in the WAN (Internet). To allow the security module to send packets via the WAN, you need to enter your DSL router as "Standard router".

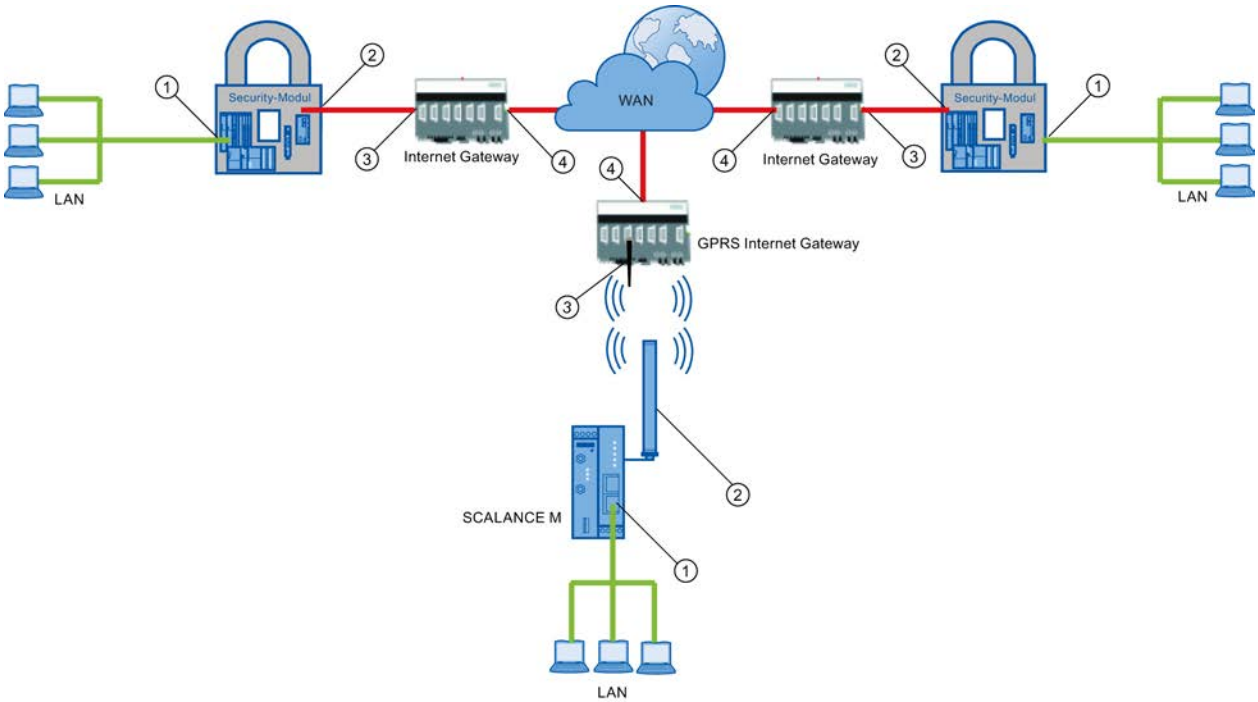
If you use a DSL router as Internet gateway, the following ports (at least) must be opened on it:

- Port 500 (ISAKMP)
- Port 4500 (NAT-T)

To allow this, when you configure the module, you have the option of assigning an external IP address as a "WAN IP address". When you download the module configuration, the group members are then informed of the WAN IP addresses of the partner modules. As an alternative to a WAN IP address, you can also enter an FQDN. If you have configured dynamic DNS on the security module at the same time, this FQDN must match the FQDN from the dyn. DNS configuration tab that is registered with a provider for dynamic DNS.

Whether the external IP address, the IP address of the DMZ interface (only SCALANCE S623 / S627-2M) or the WAN IP address / the FQDN will be used can be specified in the module-specific VPN settings of the security module. In these settings, you can also specify the interface to be used for communication by the nodes of a VPN group and the security module that is authorized to set up connections.

If you do not enter an access point here, the external IP address or the IP address of the DMZ interface (SCALANCE S623/S627-2M only) will be used as the VPN endpoint.



① Internal IP address - of a module
② External IP address - of a module
③ IP address of an Internet gateway (for example, GPRS gateway)
④ IP address (WAN IP address) of an Internet gateway (for example, DSL router)

Configuring internal network nodes

The configuration of internal network nodes is described in the following section: Auto-Hotspot

See also

How to create an IPsec tunnel with VPN groups (Page 783)

Online functions - test / diagnostics and logging

Overview of diagnostics and logging in

For test and monitoring purposes, the security module has diagnostic and logging functions.

- Diagnostic functions
  These include various system and status functions that you can use for an existing network connection to the security module.
- Logging functions
  This involves the logging of system and security events and data packets.

Recording events with logging functions

You select the events to be logged in the log settings for the relevant security module.

You can configure the following variants for logging:

- Local logging
  In this variant, you log events in the local buffer of the security module. You can then access these logs, display them and archive them on the service station in the "Online & diagnostics" dialog. The evaluation of the buffer areas of the security module is only possible if there is a network connection to the selected security module.
- Network Syslog
  With Network Syslog, you use a Syslog server that exists in the network. This logs the events according to the configuration in the log settings for the relevant security module.

Archiving log data and reading in from a file

You can save the logged events for archiving in a log file and open this later even without a network connection to the security module. For more detailed information, refer to section Auto-Hotspot.

Function overview online dialog

Functions of the "Online & diagnostics" dialog

In STEP 7, the security module provides the following functions in the "Online & diagnostics" dialog:

Entry in the online dialog	Function
Status	Display of status information about the selected security module, such as the current IP addresses of the interfaces and the current time and date.
Interface settings (only for SCALANCE S V3 or higher)	Overview of the settings of the individual interfaces.
Dyn DNS (only for SCALANCE S V3 or higher)	Overview of the settings for dynamic DNS.
System log	Display of logged system events, starting and stopping logging (only if there is an online connection to SCALANCE S modules) and starting and stopping the reading out of log data from the local buffer of the security module.
Audit log	Display of logged security events, starting and stopping reading out of log data from the local buffer of the security module.
Packet filter log	Display of logged data packets, starting and stopping logging (only if there is an online connection to SCALANCE S modules) and starting and stopping the reading out of log data from the local buffer of the security module.
ARP table (only for SCALANCE S V3 or higher)	Display of the ARP table of the security module.
Logged in users (only for SCALANCE S V3 or higher)	Shows the users logged in to the Internet page for user-specific IP rule sets.
Communication status (not for SCALANCE S602 and CP 1543-1 V1.0)	Display of status information about VPN tunnel connections and members of VPN groups to which the selected security module belongs.

Entry in the online dialog	Function
Internal nodes (not for SCALANCE S602, CP 1543-1 V1.0 and PC CPs)	Display of the learned or configured internal network nodes of the security module.
Log files (offline view)	
• System log	Display of logged system events as well as starting and stopping the display.
• Audit log	Display of logged security events as well as starting and stopping the display.
• Packet filter log	Display of logged data packets as well as starting and stopping the display.
Dynamically updated firewall rules (only for S7 CPs)	Display of the IP addresses that were released dynamically over HTTP or HTTPS, or loaded by a user.
Date and time (only for SCALANCE S)	Date and time setting.

Requirements for access

The following requirements must be met to use the online functions of a security module:

- There is a network connection to the selected module
- The project with which the module was configured is open
- a user with the required rights must be logged in to the project

How to access this function

1. Right-click on the module to process.
2. Select the "Online & diagnostics" command from the shortcut menu.
3. If there is no online connection established to the security module, click the "Connect online" button in the "Diagnostics" entry.

Online settings are not saved in the configuration

Settings that you make in online mode (for example settings for the logging memory) are not stored in the configuration on the security module. This is why the configuration settings are always applied at the restart of the module.

Functions of the online diagnostics

Status information of the security module - "Status" entry

Meaning

Display of the status of the security module selected in the project.

Table 8-34 Online & diagnostics: - "Status" entry

System and status functions	Meaning
Overview	
Hardware type	The type of the security module.

System and status functions	Meaning
External IP address	The external IP address of the security module. For CP 1628: The IP address of the Industrial Ethernet interface. For S7 CPs: The IP address of the Gbit interface.
Internal IP address	The internal IP address of the security module. For CP 1628: The IP address of the NDIS interface. If there is more than one NDIS address, only one is displayed. For S7 CPs: The IP address of the PROFINET interface.
DMZ IP address (SCALANCE S623 / S627-2M only)	The DMZ IP address of the security module.
Serial number	The serial number of the security module.
Order number	The MLFB identifier of the security module that is used when ordering.
Firmware version	The firmware version of the security module.
Operating mode	Mode of the security module (bridge mode / routing mode)
External MAC address	The external MAC address of the security module. For CP 1628: The MAC address of the Industrial Ethernet interface. For S7 CPs: The MAC address of the Gbit interface.
Internal MAC address	The internal MAC address of the security module. For CP 1628: The MAC address of the NDIS interface. For S7 CPs: The MAC address of the PROFINET interface.
DMZ MAC address (SCALANCE S623 / S627-2M only)	The DMZ MAC address of the security module
Hardware release	The hardware product version of the security module.
C-PLUG	Shows whether or not a C-PLUG is inserted.
Local time	

System and status functions	Meaning
Current time	Date and time that is displayed on the security module. Format for "German" user interface language: dd.mm.yyyy (date) hh:mm:ss (time) Format for "English" user interface language: mm/dd/yyyy (date) hh:mm:ss AM/PM (time) Format for "French" user interface language: dd/mm/yyyy (date) hh:mm:ss (time) Format for "Italian" user interface language: dd/mm/yyyy (date) hh:mm:ss (time) Format for "Spanish" user interface language: dd/mm/yyyy (date) hh:mm:ss (time) **Note (not for CPs)** You set the local time on the SCALANCE S module in "Functions" > "Date and time".
Operating time	Time since the last restart of the security module. Format for "German" user interface language: dddd.hh:mm:ss Format for "English" user interface language: dddd.hh:mm:ss Format for "French" user interface language: dddd.hh:mm:ss Format for "Italian" user interface language: dddd.hh.mm.ss Format for "Spanish" user interface language: dddd.hh:mm:ss
Time-of-day source	The source from which the date and time are obtained.
Configuration	
Created	Date and time when the project was first created. Format for "German" user interface language: dddd.hh:mm:ss hh:mm:ss (time) Format for "English" user interface language: dddd.hh:mm:ss hh:mm:ss AM/PM (time) Format for "French" user interface language: dddd.hh:mm:ss hh:mm:ss (time) Format for "Italian" user interface language: dddd.hh.mm.ss hh.mm.ss (time) Format for "Spanish" user interface language: dddd.hh:mm:ss hh:mm:ss (time)
Name	File name under which the project was last saved. The name is set by default by STEP 7 and is changed when you rename the project.
Author	Name of the user who created the project. Is adopted from the project properties.

System and status functions	Meaning
Loaded	Date and time when the project was last loaded on the security module. Format for "German" user interface language: dddd.hh:mm:ss hh:mm:ss (time) Format for "English" user interface language: dddd.hh:mm:ss hh:mm:ss AM/PM (time) Format for "French" user interface language: dddd.hh:mm:ss hh:mm:ss (time) Format for "Italian" user interface language: dddd.hh.mm.ss hh.mm.ss (time) Format for "Spanish" user interface language: dddd.hh:mm:ss hh:mm:ss (time)
Storage location	Specifies the location (e.g. town) to be set using an SNMP tool. For SCALANCE S modules, the storage location can be configured as an alternative in the advanced SNMP settings.
File system (not for CPs)	
RAM	Indicates how much RAM and flash is occupied in the file system.
Flash	

Overview of the individual interfaces - "Interface settings" entry

Module-specific function

This function is available only for SCALANCE S V3 modules or higher, see section Overview of the individual interfaces - "Interface settings" entry (Page 837)

Overview of the DDNS settings - "Dynamic DNS" entry

Module-specific function

This function is available only for SCALANCE S V3 modules or higher, see section Overview of the DDNS settings - "Dynamic DNS" entry (Page 838)

Display of the ARP table - "ARP table" entry

Module-specific function

This function is available only for SCALANCE S V3 modules or higher, see section Display of the ARP table - "ARP table" entry (Page 840)

Users logged in to the Web page - "Logged in users" entry

Module-specific function

This function is available only for SCALANCE S V3 modules or higher, see section Users logged in to the Web page - "Logged in users" entry (Page 841)

VPN connections of the security module - "Communication status" entry

Module-specific function

This function is not available for SCALANCE S 602 modules.

Meaning

Display of the communication status of the following network components:

- Other security modules of the VPN group to which the selected security module belongs
- Internal network nodes of these security modules

Table 8-35 Online & diagnostics: "Communication Status" entry

System and status functions	Meaning
Known security devices or modules	Display of the nodes with which the selected security module is in a VPN group. This also shows whether the tunnel status is active or passive. To obtain additional information on one of the nodes, select this in the list. **Note:** Configured, inactive tunnels are indicated for CPs only.
Endpoints	Display of information on the internal network nodes of the security module that you selected in the "Known security devices or modules" table. For each internal network node, this shows whether it was learned or configured. It also shows the subnet in which the internal network node is located. With SCALANCE S modules, the subnet of network nodes is only displayed in bridge mode.
Tunnel properties	Display of properties of the VPN tunnel established to the security module that you selected in the "Known security devices or modules" table.

Found internal network nodes - "Internal nodes" entry

Module-specific function

This function is not available for SCALANCE S602 modules and the CP 1543-1.

Meaning

Display of all learned and configured network nodes. This also displays whether or not the learning mode of the security module is enabled.

Updated firewall rules - "Dynamically updated firewall rules" entry

Module-specific function

This function is only available for CP x43-1, see subsection:
Updated firewall rules - "Dynamically updated firewall rules" entry (Page 855) in the section "Security for S7-300 / S7-400 / PC CPs".

Display of the firewall blacklist - "Firewall blacklist" entry

Module-specific function

This function is available only for SCALANCE S V4 modules or higher, see section
Display of the firewall blacklist - "Firewall blacklist" entry (Page 841).

Setting the date and time - "Date and Time" entry

Module-specific function

This function is available only for SCALANCE S modules, see section
Setting the date and time - "Date and time" entry (Page 842).

Logging functions

Logging system events - "System log" entry

Meaning

Display of logged system events and starting and stopping reading system events from the local memory of the security module.

The system log automatically logs successive system events, for example the start of a process. The logging can be scaled based on event classes.

System and status functions	Meaning
Start/stop logging (not for CPs)	Starts/stops recording of system events. The method and the event classes that are logged are configured in the local security settings.
Start/stop reading	Starts/stops reading of system events from the local memory of the security module. If you select the "Save log file" check box, the recorded log data is also saved as file. Select the storage location and enter a file name. Note If you select the "Save log file" check box after starting reading, the data read out up to then can no longer be saved in a log file.
Clear display	Deletes the log data shown in the table.

For more information on opening saved system events in log files, refer to section Evaluating system events in offline mode - "System log" entry (offline view) (Page 806).

Logging security events - "Audit log" entry

Meaning

Display of logged security events and starting and stopping reading of security events from the local memory of the security module.

The audit log automatically logs successive security-relevant events. This includes, for example, user actions such as enabling and disabling packet logging.

System and status functions	Meaning
Start/stop reading	Starts/stops reading of security events from the local memory of the security module. If you select the "Save log file" check box, the recorded log data is also saved as file. Select the storage location and enter a file name. Note If you select the "Save log file" check box after starting reading, the data read out up to then can no longer be saved in a log file.
Clear display	Deletes the log data shown in the table.

For more information on opening security events stored in log files, refer to section Evaluating security events in offline mode - "Audit log" entry (offline view) (Page 807).

Logging data packets - "Packet filter log" entry

Meaning

Display of logged data packets and starting and stopping reading of packet filter events.

The packet filter log records certain packets of the data traffic. Data packets are only logged if they match a configured packet filter rule (firewall) or to which the basic protection reacts (corrupt or invalid packets). This is only possible when logging is enabled for the packet filter rule.

For information about activation of the logging, refer to section Auto-Hotspot.

As well as reading the log data from the buffer and transferring it to the display, it can also be saved in a file for archiving.

System and status functions	Meaning
Start/stop logging (not for CPs)	Starts/stops logging of data packets. The method with which the data is logged is configured in the local security settings.
Start/stop reading	Starts/stops reading out of logged data packets from the local memory of the security module. If you select the "Save log file" check box, the recorded log data is also saved as file. Select the storage location and enter a file name. **Note** If you select the "Save log file" check box after starting reading, the data read out up to then can no longer be saved in a log file.
Clear display	Deletes the log data shown in the table.
Log category	Select the data packets for which the logging will be displayed. The selection depends on the settings you configured offline in the local security settings. Only the data packets for which logging was enabled are logged. If you select a category for which logging was not enabled, no data will be logged for this category.

For information on opening the stored packet filter log data, refer to section Evaluating packet filter events in offline mode - "Packet filter log" entry (offline view) (Page 807).

Evaluating log files in offline mode

Evaluating system events in offline mode - "System log" entry (offline view)

How to access this function

1. Right-click on the module to process.
2. Select the "Online & diagnostics" command from the shortcut menu.
3. Select "Diagnostics" > "Log files (offline view)" > "System log".

Meaning

Opens logged system events that you saved as a file in the online view.

For more information, refer to chapter Logging system events - "System log" entry (Page 804).

Evaluating security events in offline mode - "Audit log" entry (offline view)

How to access this function

1. Right-click on the module to process.
2. Select the "Online & diagnostics" command from the shortcut menu.
3. Select "Diagnostics" > "Log files (offline view)" > "Audit log".

Meaning

Opens logged security events that you saved as a file in the online view.

For more detailed information, refer to section Logging security events - "Audit log" entry (Page 805).

Evaluating packet filter events in offline mode - "Packet filter log" entry (offline view)

How to access this function

1. Right-click on the module to process.
2. Select the "Online & diagnostics" command from the shortcut menu.
3. Select "Diagnostics" > "Log files (offline view)" > "Packet filter log".

Meaning

Opens logged data packets that you saved as a file in the online view.

For more detailed information, refer to section Logging data packets - "Packet filter log" entry (Page 805).

Download functions

Special features when downloading security configurations

Security configurations can influence the reachability of the security module for the configuration PC. This, for example, is the case if there is a tunnel connection configured for one security module to another security module in a configuration and this configuration is downloaded from the configuration PC to the security module. After the download by the configuration PC, the security module can no longer be reached and the reachability test performed as default by STEP 7 following the download of the configuration fails. The error message output by STEP 7 relates solely to the reachability test; the actual download of the

configuration is ensured if the project data is consistent and the IP address relationship between the security module and the configuration PC is correct.

Special features when downloading configurations and firmware to SCALANCE S modules are described in the following subsection of the section "SCALANCE S":
Auto-Hotspot

Uploading configurations to the engineering station

Uploading a configuration from a SCALANCE S module or from the CP 1628 to an engineering station is not possible.

Uploading a configuration from an S7 CP that supports security to an engineering station is also possible if security features were included in the configuration. Configured security functions are not, however, transferred to the engineering station. In the transferred configuration on the engineering station, the "Activate security features" check box is also deselected.

SOFTNET Security Client

Using the SOFTNET Security Client

Area of application - access over VPN

With the SOFTNET Security Client (SSC) PC software, secure remote access is possible from PGs/PCs to automation systems protected by a security module via public networks. You require SOFTNET Security Client V4.0 HF1 for S7-300/S7-400 CPs and for the PC CP 1628. These CPs are not approved for operation with SOFTNET Security Client ≤ V4.0. The SOFTNET Security Client has not been released for the CP 1543-1.

With the SOFTNET Security Client, a PG/PC is configured automatically so that it can establish a secure IPsec tunnel communication in the VPN (Virtual Private Network) with one or more security modules.

This IPsec tunnel communication makes it possible for PG/PC applications such as NCM diagnostics to securely access devices or networks that are located in an internal network protected by SCALANCE S.

How does the SOFTNET Security Client work?

The SOFTNET Security Client reads the configuration that was created by STEP 7 and determines from the file which certificates are to be imported.

The CA certificate and, when applicable, the private key are imported and stored on the local PG/PC.

Following this, security settings are made based on the data from the configuration so that applications can access IP addresses downstream from the security modules.

If the learning mode is enabled for the internal nodes or programmable controllers, a security policy is first set for secure access to security modules. Then, the SOFTNET Security Client

addresses the security modules in order to obtain the IP addresses of the relevant internal nodes.

The SOFTNET Security Client enters these IP addresses in special filter lists belonging to this security policy. Applications can then communicate with the programmable controllers via VPN.

Creating a configuration file in STEP 7

Configuring a SOFTNET Security Client module in the project

The SOFTNET security client is created as a module in the project. In contrast to the other security modules, you do not need to configure any further properties.

Assign the SSC module to the VPN group or groups at which an IPsec tunnel to the PG/PC is to be set up.

Note

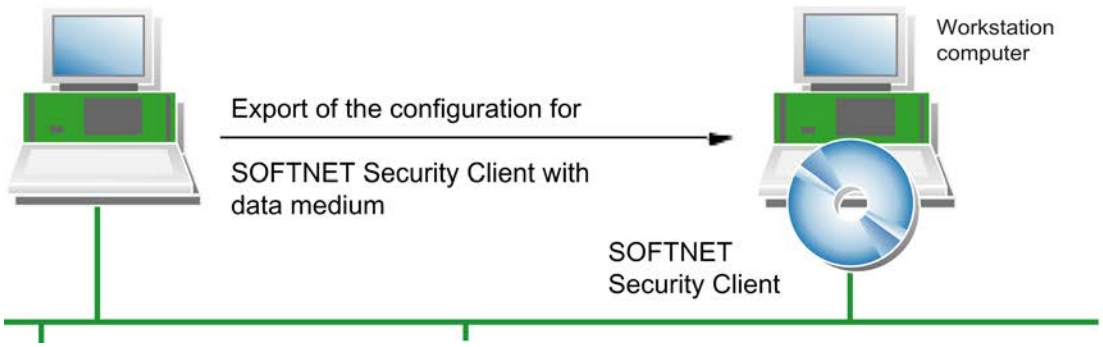
Refer to the information on parameters in section
Including security module in configured VPN group (Page 789).

Note

If you create several SOFTNET Security Clients within a group, no tunnels are set up between these clients but only from the relevant client to the security modules.

Configuration files for the SOFTNET Security Client

The interface between STEP 7 as configuration tool and the SOFTNET Security Client is controlled by configuration files.



The configuration is stored in the following file types:

- *.dat
- *.p12
- *.cer

Procedure

To generate the configuration files, perform the following steps in STEP 7:

1. In the "Devices & networks" view, select the "Topology view" or the "Network view" tab.
2. Insert a PC system of the type "SOFTNET Security Client" in the selected tab from the hardware catalog.
3. Assign the SOFTNET Security Client to the VPN groups in which the PG/PC will communicate over IPsec tunnels.
4. Make sure that the "Generate SSC files" check box is selected under the "Configuration of the SOFTNET Security Client" entry in the local security settings of the SOFTNET Security Client.
5. Select the storage location for the configuration files.
6. Compile the configuration of the SOFTNET Security client to export the configuration file.
7. If you selected certificate as the authentication method, specify a password for the certificate of the VPN configuration. If you do not assign a password, the project name (not the project password) is used as the password.
   Result: Export of the configuration files is completed.
8. Adopt the files of the type *.dat, *.p12, *.cer on the PG/PC on which you want to operate the SOFTNET Security Client.

SCALANCE S

Configuring interfaces for SCALANCE S modules

Overview

Configuring the mode

With the mode you specify how interface routing is handled (external/internal). The DMZ interface of the security module (SCALANCE S623/S627-2M only) is always connected in routing mode. For more detailed information, refer to the section
Configuring IP address parameters (Page 812)

Configuring the interfaces

If the external interface or the DMZ interface (SCALANCE S623/S627-2M only) of a security module needs to be configured, it must be activated using the "Activate interface" check box. Set the IP address information for each interface and settings for the individual ports. The

following options are available with which you can assign an IP address for the external interface and for the DMZ interface in the "General" entry (SCALANCE S623/S627-2M only):

- Static IP address with subnet mask. For more detailed information, refer to the section Configuring IP address parameters (Page 812)
- Address assignment using PPPoE. For more detailed information, refer to the section Configuring an Internet connection (Page 814)
  The internal interface can only be configured using a static IP address.

Note

External interface and DMZ interface as Internet access

The simultaneous operation of PPPoE on the external interface and on the DMZ interface (dual ISP) is not possible.

Point to Point Protocol over Ethernet (PPPoE)

To allow Internet/WAN access directly via a DSL modem, the IP address on the external interface or on the DMZ interface is assigned using PPPoE. PPPoE is a dial-in protocol for obtaining IP addresses from an Internet service provider (ISP). SCALANCE S is operated here in routing mode.

To use this IP address assignment mode, specify the ISP in the "PPPoE" entry. The IP address, the subnet mask, the standard router and the DNS server of the interface are specified by the ISP.

Note

A configured standard router is not taken into account when using PPPoE. This is assigned dynamically to the module by the ISP.

Note

No network components between SCALANCE S and DSL modem

If the interface of a SCALANCE S module is operated using PPPoE, there must be no other network components between this interface and the connected DSL modem, otherwise the dial-in data of the Internet Service Provider will be transferred unencrypted over this link due to the protocol being used.

Configuring media modules

In addition to the functions of the SCALANCE S623, the S627-2M has two media module slots in which an electrical or optical media module with two ports can be inserted. This expands both the external and internal interface by up to two ports. In routing mode, the additional ports of the security module can be used to link the external and internal interface to ring topologies.

To integrate media modules in the SCALANCE S627-2M, select the security module and change to the device view. Then select the required media modules from the hardware catalog.

For ports with the port type "Copper", you can set the transmission speed and the duplex method manually using the port mode. For ports with the port type "Optical", the port mode is fixed by the media module used or by the SFP transceiver used and cannot be adapted.

You will find information on connecting the media module ports to MRP rings in the following section:
Auto-Hotspot

See also

Setting the operating mode (Page 812)

Setting the operating mode

How to access this function

1. Select the module to be edited.
2. In the local security settings, select the entry "Mode".

Operating mode - possible selections

You can change the operating mode in this dialog if the security module is not included in a VPN group. If the security module is in a VPN group, the mode cannot be changed.

The selection applies to interface routing between the external and internal interface. The DMZ interface (SCALANCE S623 and S627-2M only) is always connected in routing mode.

Bridge mode	For operation in flat networks. External and internal interface are in the same IP subnet. For S623 / S627-2M: External and internal interface are in the same IP subnet, the DMZ interface is in a different IP subnet or is deactivated.
Routing mode	All interfaces are in different IP subnets. If you have activated the routing mode, you must configure an internal IP address and subnet mask for the internal interface of the security module. Note If you have enabled the routing mode for the SCALANCE S module, no MAC firewall rules can be defined.

Configuring IP address parameters

Meaning

Specify network parameters such as the IP address and subnet mask for the interface(s) of the security module.

How to access this function

1. Select the module to be edited.
2. Select "External interface [P1] red" or "DMZ interface [P3] yellow" in the local security settings.

   Note
   Configuration of the internal interface in routing mode

   If you have selected the "Routing" mode for the security module, you must also configure an internal IP address and subnet mask for the internal interface of the security module. You can access this function in the local security settings under "Internal interface [P2] green" > Ethernet addresses".

3. If applicable, enable the interface using the "Activate interface" check box.
4. Select the "Ethernet addresses" entry.
5. Complete the settings specified in the following table.

Parameter	Meaning
IP address	IP address for the external interface. The IP address consists of four decimal numbers from 0 to 255, with each number being separated by a period, for example, 141.80.0.16.
Subnet mask	The subnet mask consists of four decimal numbers that are separated by period, for example, 255.255.0.0
Use router	Select this check box if you want to use a standard router and enter its address in the "Router address" input box.

Note
Networking the interfaces

Network the individual interfaces of the security module with suitable subnets to avoid IP address conflicts.

Special features with a standard router

- If the IP assignment configured is via "PPPoE", no standard router needs to be configured because the standard route is always automatically via the PPPoE interface.
- If the address assignment configured is with "Static address" and if the security module is connected to the Internet via a DSL (NAPT) router, the DSL router must be entered as the standard router.

Configuring port mode

Meaning

The port mode specifies the transmission speed and the duplex method. For SCALANCE S V2 modules, the port mode is set to "Autonegotiation" as default. This means the transmission speed and the duplex setting are selected automatically. Moreover, the auto-crossing function is supported.

Configurable port modes

For SCALANCE S V3 and higher, the following port modes can be configured:

Port mode	Meaning
Autonegotiation	The transmission speed and the duplex setting are selected automatically. Note A transmission speed of 1000 Mbps and the autocrossing function are supported only if autonegotiation is selected.
10 Mbps, half and full duplex	Transmission speed of 10 Mbps
100 Mbps, half and full duplex	Transmission speed of 100 Mbps

Configuring an Internet connection

Module-specific function

This function is available only for SCALANCE S V3 modules or higher.

Requirement

The "PPPoE" entry is only displayed in the local security settings if the IP assignment method "PPPoE" is configured for one of the interfaces.

How to access this function

1. Select the security module to be edited
2. In the local security settings, select the entry "PPPoE".

Meaning

In this entry, you make settings related to the Internet Service Provider (ISP) if a connection using PPPoE is set for one of the interfaces of the security module.

Table 8-36 Settings for the ISP account

Function	Description
Authentication	Select none or one of the following authentication protocols: • PAP (Password Authentication Protocol) • CHAP (Challenge Handshake Authentication Protocol) **Note** Both communications partners have to use the same authentication method otherwise no connection can be established.
User name	Enter the name for logging in with the ISP account.
Password	Enter the password for logging in with the ISP account.
Repeat password	Enter the password for logging in with the ISP account again.

Table 8-37 Settings for the connection

Function	Description
Permanent connection	Permanent Internet connection. After the connection has been terminated by the provider, the connection is automatically restored even if there are currently no packets to be sent.
On-demand connection	The Internet connection is established automatically if packets need to be sent to the Internet. In this setting, delays in the sending of packets are possible.
Maximum idle time (only with the setting "on-demand connection")	If no packets are sent during a certain time, the Internet connection is automatically terminated. In the "Maximum idle time" box, enter the time in seconds after which the connection will be terminated. Permitted values: 10 … 3600.
Forced disconnection (only with the "Permanent connection" setting)	Select the check box to be able to adapt the time of the forced disconnection by the security module.
Forced disconnection time (only with the "Permanent connection" setting)	The provider terminates the Internet connection automatically after a certain period. If you enter a time of day in this box, the security module terminates the Internet connection itself at this time. This allows disconnection of the Internet connection by the provider to be delayed under certain circumstances. A self-initiated forced disconnection is only possible with an existing permanent connection. Permitted entries: 00:00 ... 23:59

Configuring dynamic DNS

Module-specific function

FQDNs can be configured for SCALANCE S V3 or higher. The resolution of FQDNs by SCALANCE S modules is possible as of SCALANCE S V4.

Meaning

With dynamic DNS, you can access a constantly changing IP address with a permanently defined name (FQDN). This is necessary, for example, if you want to access a server that can be reached via a public IP address that changes.

How it works

The security module signals the current WAN IP address via which the security module can be reached to a provider for dynamic DNS (for example DynDNS.org, no-ip.com). The provider makes sure that DNS queries sent to the FQDN of the security module are replied to with the current WAN IP address of the security module.

Dynamic DNS is permitted on the following interfaces:

- External interface
- DMZ interface

Setting up dynamic DNS - Requirements

Requirement:

- An account has been created with a provider of dynamic DNS and an FQDN has been registered.

Setting up dynamic DNS - Follow the steps below:

1. Select the security module to be edited.
2. In the local security settings, select the entry "DNS".
3. If the security module is downstream from a DSL router or DSL modem, you specify a valid DNS server address. To do this, two options are available:

Option	Meaning
Obtain DNS server address automatically	The address of the DNS server can be obtained automatically using PPPoE if the security module is connected to the Internet via a DSL modem. Can only be set for the external interface and the DMZ interface.
Use the following DNS server address:	Enter the address of the preferred and of the alternative DNS server manually.

4. Activate the "Activate service" check box in the "Primary dynamic DNS service" area and make the following settings:

Setting	Meaning
Provider	Choose the provider with which you have set up an account for dynamic DNS.
User account with the provider	Enter the user name that you specified when you created the account.
Password with the provider	Enter the password that you specified when you created the account.
FQDN	Enter the host name (e.g. mysecuritymodule) and the domain name (e.g. dyndns.org) that is registered with the provider separated by a period. If an FQDN is also entered in the "VPN" entry, both must match.
Monitor IP address change on DSL router	If the security module is connected to the Internet via a DSL router, enabling this function activates the function of the Check IP service. The security module periodically sends queries to determine the current IP address of the DSL router and to detect an IP address change on the DSL router. The IP address specified in this way is sent to the provider with each change ID.
Period	Specify the interval at which the Check IP service is called. Permitted values: 10 to 1440 minutes

5. In case the primary provider fails, create a second provider in the "Secondary dynamic DNS service" area (optional setting).

Setting up a user-defined provider - follow the steps below:

Select the "User-defined" entry in the "Provider" drop-down list and make the following entries:

Setting	Meaning
Ignore errors when checking the server certificate	To ensure that the authentication data is protected, the certificate of the update server is checked as default. If the certificate check fails, the HTTPS connection is terminated and the account data is not transferred. If you select the check box, the function is disabled, for example if the server certificate of the dynamic DNS service is invalid (for example expired). It is advisable not to ignore the check and not to select the check box.
Provider update URL	With the predefined providers (DynDNS.org and No-IP.com), the URL is already populated.
Check IP service URL	With the predefined providers (DynDNS.org and No-IP.com), the URL is already populated.

Configuring LLDP

Module-specific function

This function is available only for SCALANCE S V4 modules or higher.

Requirement

The security module is in routing mode.

Meaning

LLDP (Link Layer Discovery Protocol) is a protocol used to discover network topologies. A device capable of LLDP can send information about itself to neighboring devices at regular intervals and at the same time receive information from neighboring devices. The received information is stored on every device with LLDP capability in an LLDP MIB file. Network management systems can access these LLDP MIB files using SNMP and therefore recreate the existing network topology.

Configurable parameters

The degree of activity of the security module in terms of LLDP can be configured under the "LLDP mode" entry of the relevant interface.

Parameter	Description
Name	Name of the port for which the setting is configured.
LLDP mode	Configured LLDP mode: • RxTx: LLDP frames can be sent and received • Off: No LLDP frames can be sent or received

Media redundancy in ring topologies

Media redundancy with MRP

Module-specific function

This function is available only for SCALANCE S627-2M modules.

Meaning

The term "media redundancy" groups together various methods for increasing availability in Industrial Ethernet networks in which devices can be reached over different paths. This might be achieved by meshing networks, arranging parallel transmission paths or by closing a linear bus topology to form a ring.

Media redundancy method MRP

Media redundancy within a ring topology is available with SIMATIC NET products among other things with the MRP method (Media Redundancy Protocol).

With this method, one of the nodes is configured as the redundancy manager. The other nodes are redundancy clients. SCALANCE S627-2M modules can only adopt the role of an MRP client. Using test frames, the redundancy manager checks the ring to make sure it is not interrupted. The redundancy clients forward the test frames. If the test frames of the redundancy manager no longer arrive at the other ring port of the redundancy manager due to an interruption, the redundancy manager switches through its two ring ports and informs the redundancy clients of the change immediately.

The time the SCALANCE X switches need to switch through their ring ports as redundancy manager is 200 ms.

Note on the use of MRP

- MRP is supported in ring topologies with up to 50 devices. Exceeding this number of devices can lead to a loss of data traffic.
- It is recommended that you set the ring ports involved to full duplex and 100 Mbps. Otherwise there may be a loss of data traffic.

Possible uses of MRP on media module ports

MRP is supported only on the media module ports of the SCALANCE S627-2M. The following table shows the possible uses of MRP on the media module ports of a SCALANCE S627-2M:

Ring ports	Media module 1		Media module 2	
	P4	P5	P6	P7
MRP Client	-	-	-	-
	Ring 1	Ring 1	-	-
	-	-	Ring 2	Ring 2
	Ring 1	Ring 1	Ring 2	Ring 2

With two lower-layer rings per SCALANCE S module, layer 3 communication is possible between the rings.

Configuring MRP for the security module

Requirement

- The security module is in routing mode.
- The interfaces of the security module to be connected to MRP rings are networked with the relevant ring managers.

How to access this function

1. Select the security module to be edited.
2. In the settings of the required interface, select the "Media redundancy" entry.

Configurable parameters

Parameter	Meaning	Possible selections
MRP domain (only if the "MRP client" media redundancy role is selected)	The members of an MRP ring are specified with the help of MRP domains. The same MRP domain must be selected for the interfaces of all modules to be connected to the same MRP ring.	Display of the MRP domain used for the interface.
Media redundancy role	Selection of the media redundancy protocol or disabling of media redundancy for the interface.	• Not a node in the ring • MRP Client
Ring port 1 (only when the media redundancy role "MRP client" is selected)	Name of the first ring port of the selected interface if the media redundancy role "MRP client" was selected for it.	-
Ring port 2 (only when the media redundancy role "MRP client" is selected)	Name of the second ring port of the selected interface if the media redundancy role "MRP client" was selected for it.	-
Domain settings	Using the domain settings, you can add new MRP domains, edit the names of existing MRP domains and delete existing MRP domains.	-
Alternative media redundancy protocol	Select this check box to enable the interface of the security module for other media redundancy protocols.	• Enable interface for other media redundancy protocols • Disable interface for other media redundancy protocols (default setting)
Passive listening	Enable this check box if the selected interface will be connected to third-party networks in which STP/RSTP (Spanning Tree Protocol/ Rapid Spanning Tree Protocol) is used.	• Activate passive listening • Deactivate passive listening (default)

Setting up a firewall

Local firewall rules for SCALANCE S modules

Configuring a firewall with predefined firewall rules

Configuring a firewall using predefined firewall rules

How to access this function

1. Select the module to be edited.
2. Select the "Firewall" entry in the local security settings.

Firewall enabled as default

The "Enable firewall" check box is enabled by default. The firewall is therefore activated automatically and all access from external to the security module is blocked. By clicking the relevant check box, enable the firewall rules for the specific direction.

Note

Detailed firewall settings in advanced firewall mode

In advanced firewall mode, you can restrict firewall rules to individual nodes. To change to advanced firewall mode, select the "Enable firewall in advanced mode" check box. For more detailed information about the advanced firewall mode, refer to the section Overview of local firewall rules (Page 755)

Firewall configuration with VPN

If the security module is in a VPN group, the "Tunnel communication only" check box is enabled as default. This means that only encrypted IPsec data transfer is permitted via the external interface or via the DMZ interface.

If you deselect the check box, tunneled communication and also the types of communication selected in the other boxes are permitted.

Table 8-38 Available firewall rules and directions

Service	From internal to external	From external to internal	From internal to DMZ (only for S623 / S627-2M)	From DMZ to internal (only for S623 / S627-2M)	Enabled ports	Meaning
Allow IP communication	x	x	x	x	-	IP communication for the selected communication directions is allowed.
Allow S7 protocol	x	x	x	x	TCP port 102	Communication of the nodes using the S7 protocol is allowed.
Allow FTP/ FTPS (explicit mode)	x	x	x	x	TCP port 20 TCP port 21	For file management and file access between server and client.
Allow HTTP	x	x	x	x	TCP port 80	For communication with a Web server.
Allow HTTPS	x	x	x	x	TCP port 443	For secure communication with a Web server, for example, for Web diagnostics.
Allow DNS	x	x	x	x	TCP port 53 UDP port 53	Communication connection to a DNS server is allowed.
Allow SNMP	x	x	x	x	TCP port 161/162 UDP port 161/162	For monitoring nodes capable of SNMP.
Allow SMTP	x	x	x	x	TCP port 25	For sending e-mails via an SMTP server.
Allow NTP	x	x	x	x	UDP port 123	For synchronization of the time of day.
Allow DHCP	x	x	x	x	UDP port 67 UDP port 68	Only in bridge mode Communication connection with a DHCP server is permitted.

Table 8-39 Logging

Option	Action when activated
Log tunneled packets	Only active if the security module is a member of a VPN group. All IP packets transferred via the tunnel are logged.
Log blocked incoming packets	All incoming IP packets that are discarded are logged.
Log blocked outgoing packets	All outgoing IP packets that are discarded are logged.

You can view the logged packets at the "Packet filter log" entry in the "Online & Diagnostics" dialog. For more detailed information, refer to the section Logging data packets - "Packet filter log" entry (Page 805).

Configuring a firewall with predefined MAC rules

How to access this function

1. Select the module to be edited.
2. Select the "Firewall" entry in the local security settings.

Firewall enabled as default

The "Enable firewall" check box is enabled by default. The firewall is therefore activated automatically and all access from external to the security module is blocked. By clicking the relevant check box, enable the firewall rules for the specific direction.

Note

Detailed firewall settings in advanced firewall mode

In advanced firewall mode, you can restrict firewall rules to individual nodes. To change to advanced firewall mode, select the "Enable firewall in advanced mode" check box. For more detailed information about the advanced firewall mode, refer to the section Overview of local firewall rules (Page 755).

Firewall configuration with VPN

If the security module is in a VPN group, the "Tunnel communication only" check box is enabled as default. This means that only encrypted IPsec data transfer is permitted via the external interface or via the DMZ interface.

If you deselect the check box, tunneled communication and also the types of communication selected in the other boxes are permitted.

Available MAC rules and directions

Service	From internal to external	From external to internal	Meaning
Allow MAC communication	x	x	The MAC traffic from internal to external and vice versa is allowed.
Allow ISO protocol	x	x	The ISO traffic from internal to external and vice versa is allowed.
Allow SiClock	x	x	SiClock time frames from internal to external nodes and vice versa are permitted.
Allow DCP	x	x	Internal to external or external to internal DCP traffic for IP address assignment is permitted.

Table 8-40 Logging

Option	Action when activated
Log tunneled packets	Only active if the security module is a member of a VPN group. All MAC packets transferred via the tunnel are logged.
Log blocked incoming packets	All incoming MAC packets that are discarded are logged.
Log blocked outgoing packets	All outgoing MAC packets that are discarded are logged.

User-specific IP rule sets

Overview

Module-specific function

This function is available only for SCALANCE S V3 modules or higher.

Meaning

Initially, individual or multiple users are assigned to user-specific IP rule sets. The user-specific IP rule sets are then assigned to individual or multiple security modules. This makes it possible, to allow user-specific access. If, for example all access to the networks downstream from a security module is blocked, certain nodes can be allowed for a user based on their IP addresses. This means that access is allowed for this user but access remains blocked for other users.

User logon via the Internet

The user can log in to the external interface or the DMZ interface of the security module via a Web page. If authentication is successful, the predefined IP rule set for this user is enabled. The connection of the security module is via HTTPS using the IP address of the connected port and taking into account the valid routing rules:

Example:

External interface: 192.168.10.1

Call up of the login page with: https://192.168.10.1/

Users can log on with every role as long as the user is assigned to a user-specific firewall rule set.

User-specific IP rule sets are used locally - conventions

The same conventions as described in the following section apply:
Global firewall rule sets - conventions (Page 748)

Creating and assigning user-specific IP rule sets

Creating user-specific IP rule sets

1. In the global security settings, select the entry "Firewall" > "User-specific IP rule sets" > "IP rule sets".
   Result: The previously created user-specific IP rule sets are displayed under the selected entry.
2. Double-click on the "Add new IP rule set" entry.
3. Enter the following data:
   – Name: Project-wide, unique name of the rule set. The name appears in the local rule list of the security module after the rule set is assigned.
   – Description (optional): Enter a description of the user-specific IP rule set.
4. Enter the firewall rules one by one in the list.
   Note the parameter description in the following sections:
   Defining IP packet filter rules (Page 756)

Assigning user-specific IP rule sets

1. Within the created user-specific IP rule set, change to the "User" tab.
2. In the "Available users" area, select the users you want to assign to the user-specific rule set.
3. Click the << button to assign the selected users to the user-specific rule set.

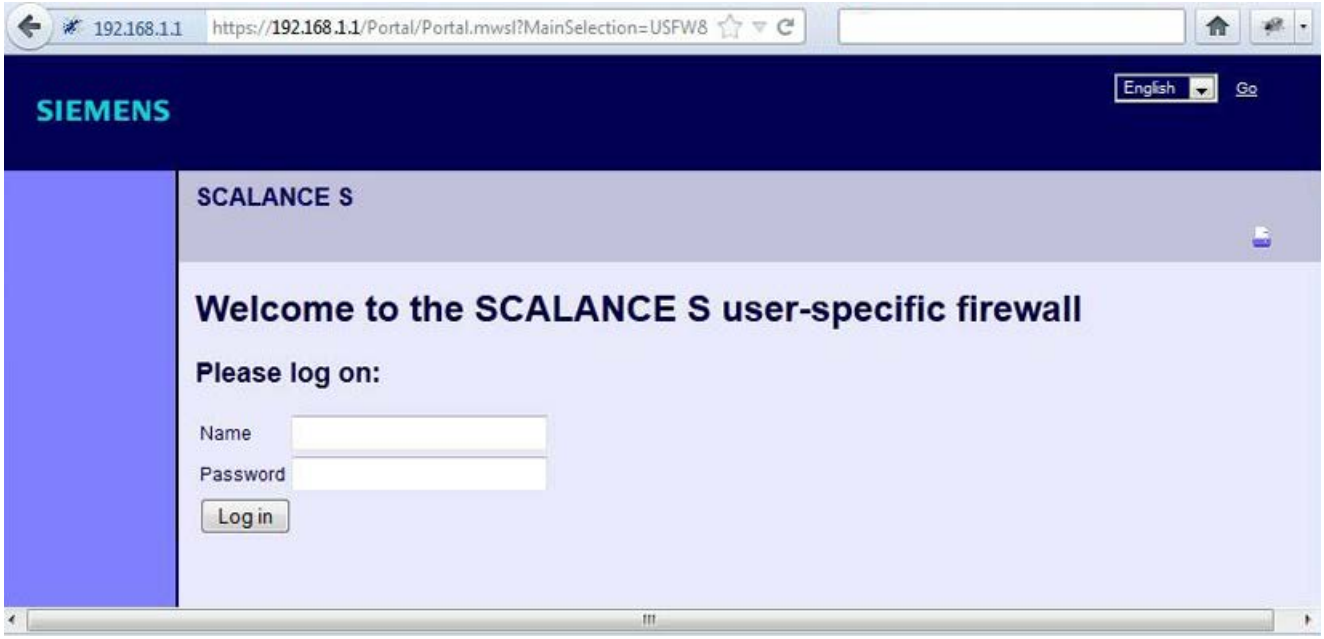
Note

Assignment of user-specific IP rule sets

- A security module can only be assigned one user-specific rule set per user.
- Due to the assignment, the right to log in to the security module is activated implicitly for all users assigned to the IP rule set.

Result

- The user-specific rule set is used by the assigned security modules as a local rule set and automatically appears in the local list of firewall rules.
- The user can log on with the security module.



IP packet filter directions SCALANCE S

Meaning

Possible selections for the communication directions "From" and "To" in the IP rules of the advanced firewall mode.

The following directions are available

Available options / ranges of values		Security module		
From	To	S602	S61x	S623 / S627-2M
Internal	External	x	x	x
	Tunnel	-	x	x
	Any	-	x	x
	DMZ	-	-	x
	Internal	x	x	x
External	Internal	x	x	x
	Any	-	-	x
	Tunnel	-	-	x
	DMZ	-	-	x
Tunnel	Internal	-	x	x

Available options / ranges of values		Security module		
	External	-	x	x
	DMZ	-	-	x
Any	Internal	-	x	x
	External	-	-	x
	DMZ	-	-	x
DMZ	Internal	-	-	x
	External	-	-	x
	Any	-	-	x
	Tunnel	-	-	x

x = communication direction is configurable
- = communication direction is not configurable

MAC packet filter directions SCALANCE S

Meaning

Possible selections for the communication directions "From" and "To" in the MAC rules of the advanced firewall mode.

The following directions are available

Available options / ranges of values		Security module		
From	To	S602	S61x	S623 / S627-2M
Internal	External	x	x	x
	Tunnel	-	x	x
	Any	-	x	x
External	Internal	x	x	x
	Any	-	-	x
	Tunnel	-	-	x
Tunnel	Internal	-	x	x
	External	-	x	x
Any	Internal	-	x	x
	External	-	-	x

x = communication direction is configurable
- = communication direction is not configurable

Adapting standard rules for IP services

Module-specific function

This function is available only for SCALANCE S V3 modules or higher.

Requirement

This function is available only in advanced firewall mode.

How to access this function

1. Select the security module to be edited.
2. In the local security settings, select the entry "Firewall" > "Standard rules for IP services".

Meaning of the advanced settings

Parameter	Meaning when activated
Use advanced status options	The permitted number of connections and firewall statuses per time unit is limited. If a network node exceeds this limit, its IP address is entered in the IP blacklist of the security module. The IP blacklist of the security module can be viewed in online mode.
Log all activated rules	Packets that are allowed according to the default rules for IP services are logged.
Enable ICMP test for interfaces	Ping queries coming in on an interface of the security module can be forwarded to other interfaces. This means that, for example, ping queries can be made from the external network to the internal interface of the security module.

Default firewall rules for SCALANCE S

The following table lists the default firewall rules for SCALANCE S modules. In some cases, the firewall rules are only set if the relevant service is used by the security module (e.g. SNMP).

Service	Direction	X1 interface (red)	Interface X2 (green)	Interface X3 (yellow) (only for S623, S627-2M)	Tunnel interface (not for S602)
Interface rerouting	outgoing	-	x	-	-
HTTPS		x	x*	x	x*
ICMP	incoming	-	x	-	x
SNMP	incoming	x	x	x	x
Syslog	outgoing	x	x	x	x
NTP	outgoing	x	x	x	x
DNS	outgoing	x	x	x	x
HTTP	outgoing	x	-	x	-
VPN (IKE)		x	-	x	-
VPN (NAT Traversal)		x	-	x	-
BootP Server	incoming	-	x	x	-
BootP Client	outgoing	-	x	x	-

x enabled as default

- disabled as default

* cannot be adapted

Specifying routes

Meaning

Specify routes for addressing subnets that cannot be reached directly via the security module.

How to access this function

1. Select the module to be edited.
2. In the local security settings, select the "Routing" entry.
3. Double-click "Add new" in the table to add a route.
4. Enter the following values:

Parameter	Function	Example of a value
Network ID	Requests to nodes of the subnet with the network ID and the subnet mask specified here are forwarded to the subnet via the specified router IP address. Based on the network ID, the router recognizes whether a destination address is inside or outside the subnet. The specified network ID must not be located in the same subnet as the IP address of the security module.	192.168.11.0
Subnet mask	The subnet mask determines the network structure. Based on the network ID, the router recognizes whether a destination address is inside or outside the subnet.	255.255.255.0
Router IP address	IP address of the router that connects to the subnet. As an alternative, an FQDN can be entered for SCALANCE S V4 modules. The specified router IP address must be in the same subnet as the IP address of the security module.	192.168.10.2
Activate rerouting (only for SCALANCE S V3/V4 modules)	Select this check box if the frames of the entered route will enter and leave on the same interface of the security module (rerouting). Rerouting is only supported on the internal interface of the security module.	

Security module as DHCP server

Overview DHCP server

Overview

You can operate the SCALANCE S module on the internal network and on the DMZ network as DHCP server (DHCP = Dynamic Host Configuration Protocol). This allows IP addresses to be assigned automatically to the devices connected to the internal network.

Simultaneous DHCP server operation on both interfaces is possible (SCALANCE S623/ S627-2M only).

The IP addresses are either distributed dynamically from an address range you have specified or you can select a specific IP address and assign it to a particular device. If devices on the internal interface or on the DMZ interface should always be assigned the same IP address for firewall configuration, the address assignment must only be static based on the MAC address or based on the client ID.

Requirements

You configure the devices on the internal network or on the DMZ network so that they obtain the IP address from a DHCP server.

Depending on the mode, the security module sends an IP address of the standard router to the nodes in the relevant subnet or you need to inform the nodes in the subnet of a router IP address.

- Router IP address will be transferred
  In the following situations, the DHCP protocol of the security module will inform the nodes of the router IP address:
  - The node is connected to the DMZ interface (SCALANCE S623/S627-2M only)
    In this case, the security module sends its own IP address as the router IP address.
  - The node is connected to the internal interface and the security module is configured for router mode
    In this case, the security module sends its own IP address as the router IP address.
  - The node is connected to the internal interface and the security module is not configured for router mode, there is, however, a standard router specified in the configuration of the security module.
    In this case, the security module transfers the IP address of the standard router as the router IP address.
- Router IP address will not be transferred
  In the following situations, enter the router IP address manually on the node:
  - The node is connected to the internal interface and the security module is not configured for router mode. There is also no standard router specified in the configuration of the security module.

See also

Configuring a DHCP server (Page 831)

Configuring a DHCP server

How to access this function

1. Select the module to be edited.
2. In the local security settings, select the entry "DHCP server".
3. Select the interface for which you want to make the DHCP settings.
4. Make the address assignment. You have the following configuration options:
   - Static address assignment
     Devices with a specific MAC address or client ID are assigned the specified IP addresses. You specify these addresses by entering the devices in the address list in the "Static address assignment" group box.
   - Dynamic address assignment
     Devices whose MAC address or whose client ID was not specified specifically, are assigned a random IP address from a specified address range. For this purpose, activate the "Dynamic IP address pool" check box. Set the address range in the "Dynamic IP address pool" input area.

Note

Dynamic address assignment - reaction after interrupting the power supply

Please note that dynamically assigned IP addresses are not saved if the power supply is interrupted. On return of the power, you must therefore make sure that the nodes request an IP address again.

You should therefore only use dynamic address assignment for the following nodes:

- Nodes that are used temporarily in the subnet (such as service devices).
- Nodes that have been assigned an IP address and send this as the "preferred address" the next time they request an address from the DHCP server (for example PC stations).

For permanent nodes you should preferably use static address assignment by specifying a client ID or the MAC address.

Consistency check - these rules must be adhered to

Remember the following rules when making the entries.

Check / rule	Check made[1)]	
	locally	Project-wide/ module-wide
The IP addresses assigned in the address list in the "Static address assignments" group box must not be in the range of the dynamic IP addresses.		x
IP addresses, MAC addresses and client IDs may only occur once in the "Static address assignment" table (related to the security module).		x
For the statically assigned IP addresses, you must specify either the MAC address or the client ID (computer name).	x	
The client ID is a string with a maximum of 63 characters. Only the following characters may be used: a-z, A-Z, 0-9 and - (dash). Note In SIMATIC S7, a client ID can be assigned to the devices on the Ethernet interface to allow them to obtain an IP address using DHCP. With PCs, the procedure depends on the operating system being used; it is advisable to use the MAC address here for the assignment.	x	
For the statically assigned IP addresses, you must specify the IP address.	x	
The following IP addresses must not be in the range of the free IP address range (dynamic IP addresses): • All router IP addresses in the "Routing" entry • Syslog server • Standard router • Security module address(es)		x
DHCP is supported by the security module on the interface to the internal subnet and on the interface to the DMZ network. The following additional requirements for IP addresses in the range of the dynamic address assignments result from this operational behavior of the security module: • Bridge mode The range must be within the network defined by the security module. • Routing mode The range must be within the internal subnet defined by the security module. Note The DMZ network always represents a separate subnet. When using DHCP on the DMZ interface, make sure that the free IP address range (dynamic IP addresses) is within the DMZ subnet.		x
The free IP address range must be fully specified by entering the start address and the end address. This end address must be higher than the start address.	x	
The IP addresses you enter in the address list in the "Static address assignments" input area must be in the address range of the internal subnet or in the DMZ subnet of the security module.		x

Legend:

[1] Note the explanations in the section
Running a consistency check (Page 732).

Configuring proxy ARP

Module-specific function

This function is available only for SCALANCE S V3 modules or higher.

Overview

Proxy ARP allows routers to respond to ARP queries for hosts. The hosts are in networks separated by routers but use the same IP address range.

If PC1 sends an ARP request to PC2, it receives an ARP response and the hardware address of the interface (MAC address of the port of the security module) on which the query was received from the security module located in between and not from PC2. The querying PC1 then sends its data to the security module that then forwards it to PC2.

How to access this function

This function is only available for the internal interface of a security module that is a member of a VPN group and is in bridge mode.

1. Select the security module to be edited.
2. In the local security settings, select the entry "Proxy ARP".
3. If the security module is to respond to an ARP query from its own LAN as a substitute for the specific connection partner, enter the corresponding IP address.

IPsec tunnel: Creating and assigning groups

Configuring internal network nodes

Using the learning mode to learn internal nodes

Finding nodes for tunnel communication automatically

One great advantage when configuring and operating tunnel communication is that SCALANCE S modules in bridge mode can find the nodes on the internal interface automatically. New nodes are detected by the security module during operation. The detected nodes are signaled to the security modules belonging to the same group. This allows data exchange within the tunnels of a group in both directions at any time.

Detectable nodes

The following nodes are detected:

- Network nodes with IP capability
  Network nodes with IP capability are found when they transmit an ICMP response to the ICMP subnet broadcast.
  IP nodes downstream from routers can be found if the routers pass on ICMP broadcasts.
- ISO network nodes
  You can also teach-in network nodes without IP capability that can be addressed by means of ISO protocols.
  This is only possible if they reply to XID or TEST packets. TEST and XID (Exchange Identification) are auxiliary protocols for exchanging information on layer 2. By sending these packets with a broadcast address, these network nodes can be located.
- PROFINET nodes
  Using DCP (Discovery and basic Configuration Protocol), it is possible to find PROFINET nodes.

Network nodes that do not meet these conditions must be configured manually.

Subnets located on the other side of internal routers also need to be configured manually.

How to access the function

1. Select the module.
2. In the local security settings, select the entry "VPN" > "Nodes".

Enabling/disabling the learning mode

The learning function is enabled in the configuration as default for every security module.

Learning can also be disabled completely for SCALANCE S. In this case, you will need to configure all internal network nodes participating in the tunnel communication manually.

When is it useful to disable the automatic learning mode?

The default settings for the security module assume that internal networks are always secure; in other words, in a normal situation, no network node is connected to the internal network if it is not trustworthy.

Disabling the learning mode can be useful if the internal network is static; in other words, when the number of internal nodes and their addresses do not change.

If the learning mode is disabled, this reduces the load on the medium and the nodes in the internal network resulting from the learning packets. The performance of the security module is also slightly improved since it does not need to process the learning packets.

Note: In the learning mode, all nodes in the internal network are detected. The information relating to VPN configuration limits relates only to nodes that communicate over VPN in the internal network.

Note

If more than 128 internal nodes are being operated, the permitted configuration limits are exceeded and an illegal operating status results. Due to the dynamics in the network traffic, this causes internal nodes that have already been learned to be replaced by new previously unknown internal nodes.

See also

Configuring internal subnets manually (Page 836)

Configuring IP network nodes manually

Meaning

As an alternative to the learning mode that you enable using the "Enable learning of internal nodes" check box and that allows the security module to learn the internal network nodes dynamically, you can enter the network nodes to be learned manually in the "Internal IP nodes" entry and in doing so enable them for VPN tunnel communication. The MAC address of a network node can be specified as an option.

Requirement

- The security module is in bridge mode.
- The security module is a member of a VPN group.

How to access this function

1. Select the module to be edited.
2. In the local security settings, select the entry "VPN" > "Nodes" > "Internal IP nodes".

Configuring MAC network nodes manually

Meaning

As an alternative to the learning mode that you enable using the "Enable learning of internal nodes" check box and that allows the security module to learn the internal network nodes dynamically, you can enter the network nodes to be learned manually in the "Internal MAC nodes" entry and in doing so enable them for VPN tunnel communication.

Requirement

- The security module is in bridge mode.
- The security module is a member of a VPN group.

How to access this function

1. Select the module to be edited.
2. In the local security settings, select the entry "VPN" > "Nodes" > "Internal MAC nodes".

Configuring internal subnets manually

Requirement

- The security module is a member of a VPN group.

How to access this function

1. Select the module to be edited.
2. In the local security settings, select the entry "VPN" > "Nodes" > "Internal subnets".

Security module in bridge mode - "Internal subnets" entry

To be able to enable internal subnets for VPN tunnel communication manually, you need to enter the following address parameters:

Parameter	Function	Example of a value
Network ID	Network ID of the subnet to be enabled for VPN tunnel communication. Based on the network ID, the router recognizes whether a destination address is inside or outside the subnet. Must not be located in the same subnet as the IP address of the security module.	192.168.11.0
Subnet mask	The subnet mask determines the network structure. Based on the network ID, the router recognizes whether a destination address is inside or outside the subnet.	255.255.255.0
Router IP address	IP address of the router via which the subnet you are allowing is reached. Must be located in the same subnet as the IP address of the security module.	192.168.10.2

Security module in routing mode - "Subnets reachable through tunnel" entry

In routing mode, entire subnets are always tunneled. To be able to enable internal subnets for VPN tunnel communication manually, you need to enter the following address parameters:

Parameter	Function	Example of a value
Network ID	Network ID of the subnet to be enabled for VPN tunnel communication. Based on the network ID, the router recognizes whether a destination address is inside or outside the subnet. Must not be located in the same subnet as the IP address of the security module.	192.168.11.0
Subnet mask	The subnet mask determines the network structure. Based on the network ID, the router recognizes whether a destination address is inside or outside the subnet.	255.255.255.0
Comment	Entry of additional, optional comments.	

Online functions - testing / diagnostics and logging

Overview of the individual interfaces - "Interface settings" entry

Module-specific function

This function is available only for SCALANCE S V3 modules or higher.

Meaning

Table 8-41 Online diagnostics: "Interface settings" entry

System and status functions	Meaning
Interfaces	Table above: General overview of the interfaces of the security module. Table below: Information on the interface operated via PPPoE. • Status: Indicates whether or not a connection was established to the Internet Service Provider (ISP). • Current IP address: Current IP address of the interface • Gateway: IP address of the gateway • Primary dynamic DNS service: Primary service for dynamic DNS • Secondary dynamic DNS service: Secondary service for dynamic DNS • Error code (numeric): Error information if no connection could be established to the ISP.
Media redundancy (SCALANCE S627-2M only)	• Interface: Interface connected to the MRP ring • Protocol: Protocol used (MRP) • Ring port 1: Name of the first media module port of the interface connected to the MRP ring • Ring port 2: Name of the second media module port of the interface connected to the MRP ring • Domain name: Name of the MRP domain • Discrepancy: Displays whether or not there is an error in the MRP configuration • Domain UUID: UUID of the MRP domain
Media modules (SCALANCE S627-2M only)	• Port: Port ID(s) of the media module port(s) • Name: Name of the media module • MLFB: MLFB of the media module • Revision: Version of the media module • Discrepancy: Displays whether or not there is an error in the configuration of the media module

Overview of the DDNS settings - "Dynamic DNS" entry

Module-specific function

This function is available only for SCALANCE S V3 modules or higher.

Meaning

Table 8-42 Online diagnostics: "Dynamic DNS"

System and status functions	Meaning
Client status	Indicates whether or not a connection was established to a dyn. DNS server.
Current IP address	Own (in other words logged on to dyn. DNS provider) IP address.
Defined IP address	IP address assigned via PPPoE.
Current time	The current time-of-day.
Force update	The security module obtains the current IP address of its Internet access and sends an update query to the configured update server(s). This publishes the current IP address on the Internet. The status is displayed in the primary or secondary dynamic DNS service fields. This means, for example that it is possible to check whether configured data such as the user name and password of the dynamic DNS account are correct.
Cancel suspension	Cancels the suspension (blocking of IP address updates from the dynamic DNS provider on the security module), for example after the dynamic DNS provider password has been corrected or an error has been eliminated.
Primary and secondary dynamic DNS service	
FQDN	Fully Qualified Domain Name registered with the provider.
Server IP Address	IP address of the update server used.
Successful update	Successful update in the dynamic DNS update service.
Last update attempt	Last update attempt for DynDNS update service.
Last failed update	Last update error in the dynamic DNS update service.
Error code	Error status of the last unsuccessful update attempt during a dynamic DNS update.

What is the meaning of the messages?

The messages of the last DDNS update attempt have the following meaning:

Message	Meaning
Success	
`DDNS_OK`	The update query was successful.
Connection-related status messages	
`DDNS_E_CON_UPD_SRV_RESOLV_ERR`	DNS name of the update server unknown, FQDN cannot be resolved using the known DNS server.
`DDNS_E_CONUPD_SRV_UNREACHABLE`	Update server unreachable ("Timeout").
Security-related status messages (errors)	
`DDNS_E_CERT_SUBJECT_INVALID`	The common name of the subject in the certificate does not match the domain name of the update server or its IP address.
`DDNS_E_CERT_UNABLE_TO_GET_ISSUER_CERT`	Issuer certificate not found. The certificate chain could not be followed back to the root CA because an issuer certificate was not found. The trust chain is incomplete.
`DDNS_E_CERT_SIGNATURE_INVALID`	The signature of a certificate could not be read or is invalid.

Message	Meaning
DDNS_E_CERT_NO_TRUST	A certificate in the trust chain is invalid, in other words: • Not yet valid or ready expired • V3 extensions invalid • Critical V3 extension is not supported
DDNS_E_CERT_DEPTH_ZERO_SELFSIGNED_CERT	The update server has supplied a self-signed certificate and the certificate is not in the certificate store for trustworthy root CA certificates.
DDNS_E_CERT_SELF_SIGNED_CERT_IN_CHAIN	The certificate chain could be established using non trustworthy certificates but no suitable root CA certificate was found in the certificate store for trustworthy certificates.
DDNS_E_CERT_CHAIN_TOO_LONG	The certificate chain exceeds the maximum supported verification depth.
DDNS_E_CERT_INVALID_CA	A CA certificate is invalid, in other words expired, not yet valid or the V3 extensions are not suitable for the intended purpose (for example CA not set to TRUE for CA certificates).
DDNS_E_CERT_KEYUSAGE_UNSUITED	The V3 extensions key usage or extended key usage set in a certificate of the trust chain are not suitable for the use of the certificate.
DDNS_E_CERT_EXTENSION_UNSUPPORTED	A certificate in the trust chain used an extension marked as critical that is not supported.
Agent-related status messages (errors)	
DDNS_E_AGT_BAD_AGENT	• The update request does not correspond to the structure required by cRSP, for example URL parameters missing. • The update request was sent to an illegal URL on the update server. • The configured update string contains errors.

Display of the ARP table - "ARP table" entry

Module-specific function

This function is available only for SCALANCE S V3 modules or higher.

Meaning

Display of the ARP table of the security module.

Table 8-43 Online diagnostics: "ARP table" tab

System and status functions	Meaning
ARP table	Display of the static (proxy ARP) and dynamic entries of the ARP table on the security module. The "Publication type" tab specifies whether the entry is configured statically or learnt.

Users logged in to the Web page - "Logged in users" entry

Module-specific function

This function is available only for SCALANCE S V3 modules or higher.

Meaning

Shows the users logged in to the Internet page for user-specific IP rule sets.

System and status functions	Meaning
User name	Name of the logged on user.
Source IP address	IP address with which the user logged on.
Time remaining	Remaining time before the user is automatically logged off.
Maximum time of the session	Configured total time of the session.
Log off	If you click the button, the selected user is logged off.

Display of the firewall blacklist - "Firewall blacklist" entry

Meaning

Displays the IP addresses of nodes that have exceeded the permitted number of connections and firewall statuses per time unit. These nodes are entered in the IP blacklist of the firewall.

The number of connections and firewall statuses per time unit is only limited if the "Use extended status options" check box is selected in the "Standard rules for IP services" entry in the local security settings.

If you click the "Delete all" button, the displayed IP addresses are removed from the firewall blacklist of the security module. The IP addresses are also no longer displayed.

Setting the date and time - "Date and time" entry

How to access this function

1. Select the security module whose time and date you want to check or set.
2. Select the "Online & diagnostics" command from the shortcut menu.
3. In online diagnostics, select the "Functions" > "Date and time" entry.

Setting the local time on the security module

In this area, you can read out and set the time and date of the security module. When you click the "Apply" button, the security module is assigned the time and date currently entered in the "Date" and "Time" input boxes.

Setting the local time on PC

This area shows the current time and current date of the PC on which STEP 7 is installed. If you click the "Adopt for module" button, the security module is assigned the current time and current date of the PC.

Download functions

Downloading a configuration

This needs to be taken into consideration before downloading a configuration

- IP address used for downloading configurations
  Configurations are always downloaded to the security module using the IP address configured in the local security settings for the internal or external port. This IP address must match that of the security module.
- Ports
  You can download the configuration data to the security module both via the external and the internal interface.
  Ideally, you should configure the modules of a VPN group via the common external network of these modules (device port 1). If the configuration computer is located on an internal network, you must enable the IP addresses of the other modules of the group explicitly in the firewall of this SCALANCE S module and first load this module.

  Note
  Selecting a network adapter

  If you are using more than one network adapter in your PC/PG, first select the network adapter via which you can reach the SCALANCE S module in the "PG/PC Interface" drop-down list of the "Extended download to device" dialog.

- Operating mode
  Configurations can be downloaded while the SCALANCE S devices are operating. Restart the SCALANCE S module to activate your configuration changes.

  Note
  Special characteristics
  - As long as a module has not yet set IP parameters (in other words, prior to the first configuration), there must be no router between the module and the configuration computer.
  - If you swap a PC from the internal to the external interface of the SCALANCE S, access from this PC to the SCALANCE S is blocked for approximately 20 minutes.

Transferring firmware

This needs to be taken into consideration before transferring new firmware

To transfer new firmware to a security module, the following conditions must be met:

- You have the rights required to transfer firmware; refer to section Auto-Hotspot.
- The security module is configured with an IP address.

The transfer is secure

The firmware is transferred over a secure connection and can therefore also be transferred from the unprotected network.

The firmware itself is signed and encrypted. This ensures that only authentic firmware can be downloaded to the SCALANCE S module.

The transfer can take place during operation

The firmware can be transferred while a SCALANCE S module is in operation. Newly downloaded firmware only becomes active after the SCALANCE S module has been restarted. If the transfer is disturbed and aborted, the module starts up again with the old firmware version.

Security for S7-300 /S7-400 / PC CPs

Setting up a firewall

Local firewall rules for S7-300 /S7-400 / PC CPs

Overview S7-300 CPs / S7-400 CPs /PC CPs

Enabling packet filter rules

If you enable the security function for the CPs in the local security settings, initially all access to and via the CP is permitted. To enable individual packet filter rules, select the "Activate firewall" check box. Then enable the required services. Firewall rules created automatically due to a connection configuration have priority over rules set manually.

Note

Detailed firewall settings in advanced firewall mode

In advanced firewall mode, you can restrict firewall rules to individual nodes. To change to advanced firewall mode, select the "Activate firewall in advanced mode" check box.

Firewall configuration with VPN

If the security module is added to a VPN group, the firewall is enabled by default. In addition, the "Tunnel communication only" check box is enabled. This means that only encrypted IPsec data transfer is permitted via the external interface. External data traffic is blocked.

If you deselect the check box, tunneled communication and also the types of communication selected in the other boxes are permitted.

Updating connection rules

Changes to the connection configuration of CPs also change the connection-related firewall rules. To display the modified firewall rules, click the "Update connection rules" button. The modified firewall rules are then displayed in advanced firewall mode.

Configuring a firewall with predefined firewall rules - CP x43-1 Advanced

Configuring a firewall with predefined IP rules - CP x43-1 Advanced

How to access this function

1. Select the module to be edited.
2. In the local security settings, select the entry "Firewall" > "Predefined IP rules".

Table 8-44 Available services and directions

Service	From station/ internal to external	External to internal	From external to station	Enabled ports	Meaning
Allow IP communication	x	x	x	All	IP traffic for the selected communication directions is allowed.
Allow S7 protocol	x	x	x	TCP port 102	Communication of the nodes using the S7 protocol is allowed.
Allow FTP/ FTPS (explicit mode)	x	x	x	TCP port 20 TCP port 21	For file management and file access between server and client.
Allow HTTP	x	x	x	TCP port 80	For communication with a Web server.
Allow HTTPS	x	x	x	TCP port 443	For secure communication with a Web server, for example, for Web diagnostics.
Allow DNS	x	x	-	TCP port 53 UDP port 53	Communication connection to a DNS server is allowed.
Allow SNMP	x	x	x	TCP port 161/162 UDP port 161/162	For monitoring nodes capable of SNMP.
Allow SMTP	x	x	-	TCP port 25	For sending e-mails via an SMTP server.
Allow NTP	x	x	-	UDP port 123	For synchronization of the time of day.

Table 8-45 Logging

Option	Action when activated	Relevant firewall rule		
IP log settings		Action	From	To
Log tunneled packets	Only active if the security module is a member of a VPN group. All IP packets transferred via the tunnel are logged.	Allow	Station	Tunnel
		Allow	Tunnel	Station
Log blocked incoming packets	All incoming IP packets that are discarded are logged.	Drop	External	Station

Note

Relationship between log settings in default mode and firewall rules

Log settings that are made in "Predefined IP rules" and "Predefined MAC rules" have no effect on firewall rules that were automatically created as a result of configuring a connection. This means, for example, that tunneled frames belonging to a configured connection cannot be logged. In advanced firewall mode, logging can be extended to the automatically generated firewall rules of connections.

Configuring a firewall with predefined MAC rules - CP x43-1 Advanced

How to access this function

1. Select the module to be edited.
2. Select the entry "Firewall" > "Predefined MAC rules".

Table 8-46 Available services and directions

Service	From station to external	From external to station	Meaning
Allow MAC communication	x	x	The MAC traffic from station to external and vice versa is allowed.
Allow ISO protocol	x	x	The ISO traffic from station to external and vice versa is allowed.

Table 8-47 Logging

Option	Action when activated	Relevant firewall rule		
MAC log settings		Action	From	To
Log tunneled packets	Only active if the security module is a member of a VPN group. All MAC packets transferred via the tunnel are logged.	Allow	Station	Tunnel
		Allow	Tunnel	Station

Option	Action when activated	Relevant firewall rule		
MAC log settings		Action	From	To
Log blocked incoming packets	All incoming MAC packets that are discarded are logged.	Drop	External	Station
Log blocked outgoing packets	All outgoing MAC packets that are discarded are logged.	Drop	Station	External

Note

Relationship between log settings in default mode and firewall rules

Log settings that are made in "Predefined IP rules" and "Predefined MAC rules" have no effect on firewall rules that were automatically created as a result of configuring a connection. This means, for example, that tunneled frames belonging to a configured connection cannot be logged. In advanced firewall mode, logging can be extended to the automatically generated firewall rules of connections.

Configuring a firewall with predefined firewall rules - CP 1628

Configuring a firewall with predefined IP rules - CP 1628

How to access this function

1. Select the module to be edited.
2. Select the "Security" > "Firewall" > "Predefined IP rules" entry.

Table 8-48 Available services and directions

Service	From external to station	Enabled ports	Meaning
Allow IP communication	x	All	IP traffic from external to station is allowed.
Allow S7 protocol	x	TCP port 102	Communication of the nodes using the S7 protocol is allowed.
Allow FTP/FTPS (explicit mode)	x	TCP port 20 TCP port 21	For file management and file access between server and client.
Allow HTTP	x	TCP port 80	For communication with a Web server.
Allow HTTPS	x	TCP port 443	For secure communication with a Web server, for example, for Web diagnostics.
Allow DNS	x	TCP port 53 UDP port 53	Communication connection to a DNS server is allowed.

Service	From external to station	Enabled ports	Meaning
Allow SNMP	x	TCP port 161/162 UDP port 161/162	For monitoring nodes capable of SNMP.
Allow SMTP	x	TCP port 25	For sending e-mails via an SMTP server.
Allow NTP	x	UDP port 123	For synchronization of the time of day.

Table 8-49 Logging

Option	Action when activated	Relevant firewall rule		
IP log settings		Action	From	To
Log tunneled packets	Only active if the security module is a member of a VPN group. All IP packets transferred via the tunnel are logged.	Allow	Station	Tunnel
		Allow	Tunnel	Station
Log blocked incoming packets	All incoming IP packets that are discarded are logged.	Drop	External	Station

Note

Relationship between log settings in default mode and firewall rules

Log settings that are made in "Predefined IP rules" and "Predefined MAC rules" have no effect on firewall rules that were automatically created as a result of configuring a connection. This means, for example, that tunneled frames belonging to a configured connection cannot be logged. In advanced firewall mode, logging can be extended to the automatically generated firewall rules of connections.

Configuring a firewall with predefined MAC rules - CP 1628

How to access this function

1. Select the module to be edited.
2. Select the entry "Security" > "Firewall" > "MAC rules".

Table 8-50 Available services and directions

Service	From station to external	From external to station	Meaning
Allow MAC level communication	x	x	The MAC traffic from external to the station and vice versa is allowed.
Allow ISO communication	x	x	ISO traffic from external to the station and vice versa is allowed.
Allow SiClock	x	x	SiClock time-of-day frames from external to the station and vice versa are allowed.

Table 8-51 Logging

Option	Action when activated	Relevant firewall rule		
MAC log settings		Action	From	To
Log tunneled packets	Only active if the security module is a member of a VPN group. All MAC packets transferred via the tunnel are logged.	Allow	Station	Tunnel
		Allow	Tunnel	Station
Log blocked incoming packets	All incoming MAC packets that are discarded are logged.	Drop	External	Station
Log blocked outgoing packets	All outgoing MAC packets that are discarded are logged.	Drop	Station	External

Note

Relationship between log settings in default mode and firewall rules

Log settings that are made in "Predefined IP rules" and "Predefined MAC rules" have no effect on firewall rules that were automatically created as a result of configuring a connection. This means, for example, that tunneled frames belonging to a configured connection cannot be logged. In advanced firewall mode, logging can be extended to the automatically generated firewall rules of connections.

IP packet filter directions S7-300-/S7-400-/PC-CPs

Meaning

Possible selections for the communication directions "From" and "To" in the IP rules of the advanced firewall mode.

The following directions are available

Available options / ranges of values		Security module		Meaning
From	To	CP x43-1 Adv.	CP 1628	
Internal	Station	x	-	Access from the internal network to the station.
	Any	x	-	Access from internal to the external network, VPN tunnel partner and the station.
External	Station	x	x	Access from the external network to the station.
	Any	x	-	Access from external to the internal network and the station.
Station	Internal	x	-	Access from the station to the internal network.
	External	x	x	Access from the station to the external network.
	Tunnel	x	x	Access from the station to the VPN tunnel partner.
Tunnel	Station	x	x	Access via the VPN tunnel partner to the station.

Available options / ranges of values		Security module		Meaning
	Any	x	-	Access from VPN tunnel partners to the internal network and the station.
Any	External	x	-	Access from the internal network and the station to the external network.

MAC packet filter directions S7-300-/S7-400-/PC-CPs

Context

Possible selections for the communication directions "From" and "To" in the MAC rules of the advanced firewall mode.

The following directions are available

Available options / ranges of values		Security module		Meaning
From	To	CP x43-1 Adv.	CP 1628	
External	Station	x	x	Access from the external network to the station.
Station	External	x	x	Access from the station to the external network.
	Tunnel	x	x	Access from the station to the VPN tunnel partner.
Tunnel	Station	x	x	Access via the VPN tunnel partner to the station.

Configuring the access list

Module-specific function

This function is not available for the CP 1628.

Meaning

You set access protection for certain IP addresses using the IP access lists. List entries that have already been created and the corresponding rights are displayed in the local settings of the CP in the entry "Firewall" > "IP rules" (advanced firewall mode).

Note

Changed behavior after activation of security

- Once you have activated the security function for a CP, access protection will only apply to the external interface. You can apply access protection to the internal interface as well, by configuring suitable firewall rules in advanced firewall mode.
- The CP also responds to ARP requests from IP addresses that have not been released (layer 2).
- If the IP access list of a CP contains no entries and you activate security for the CP, the firewall will be activated and prevent access to the CP from external locations. Configure the corresponding firewall rules in advanced firewall mode so that the CP can be reached.

Effect of IP access list entries at activation of security

If security is enabled in the local settings of a CP, the corresponding rules are created in the advanced firewall mode. A firewall rule "Allow" > "External" > "Station" is created for an IP address you specified in the address list. The IP address from the IP access list is used accordingly as source IP address. IP addresses from a defined IP address range are also integrated into corresponding firewall rules.

Requirements for editing

Before you can edit created firewall rules, the following condition must be met:

- for editing using STEP 7: "Configure security" configuration right
- for editing using a Web server: Module right "Web: Expand IP access control list"

The requirements for editing the IP access lists outside the local security settings are described in the sections on the specific CPs.

Connection-related automatic firewall rules

Meaning

For connections that were configured using CPs, STEP 7 automatically creates firewall rules that allow communication with the partner of the CP in the specified direction (CP active/ passive). The connection establishment directions are taken into account. To display these

firewall rules, if the advanced firewall mode is enabled, the "Update connection rules" button needs to be clicked. The firewall rules are then displayed in advanced firewall mode.

Note

Enabling UDP multicast and UDP broadcast connections manually

No automatic firewall rules are created for UDP multicast and UDP broadcast connections. To enable the connections, add the relevant firewall rules manually in advanced firewall mode.

Depending on how the connection establishment is configured, the following level 3 firewall rules are created. If the security module is in a VPN group, the direction "External" changes to "Tunnel". This applies only to CPs that support VPN.

The IP address of the connection partner is entered in the "Source IP address" or "Destination IP address" column of these firewall rules.

CP->external	Action	From	To
active	Allow	Station	External
	Drop	External	Station
passive	Drop	Station	External
	Allow	External	Station
active and passive	Allow	External	Station
	Allow	Station	External

CP->internal	Action	From	To
active	Allow	Station	Internal
	Drop	Internal	Station
passive	Drop	Station	Internal
	Allow	Internal	Station
active and passive	Allow	Internal	Station
	Allow	Station	Internal

For level 2 connections, "Allow" rules are created for both directions. If the security module is in a VPN group, the direction "External" changes to "Tunnel". This applies only to CPs that support VPN.

The MAC address of the connection partner is entered in the "Source MAC address" or "Destination MAC address" column of these firewall rules.

CP->external	Action	From	To
active, passive, active and passive	Allow	Station	External
	Allow	External	Station

Note

Changing the connection configuration

Changes to the connection configuration of CPs also change the connection-related firewall rules. To display the modified firewall rules, click the "Update connection rules" button.

Conventions for automatically created firewall rules

- Priority
  The rules have highest priority and are therefore inserted at the top in the local rule list.
- Deleting rules
  The rules cannot be deleted. Logging can be enabled and services can be assigned. Moreover, you may insert a bandwidth and a comment.
- Changing the action
  If you set the action from "Allow" to "Drop" or vice versa, this is overwritten again during renewed system synchronization. Select "Allow*" or "Drop*" to retain your changes. In this case, only the IP address is synchronized and the action and direction remain as set. Settings for logging, service, bandwidth and comment are also retained after a renewed system synchronization even without changing the action to "Allow*" or "Drop*". If the configured connection is deleted, the corresponding rules are removed from the list.

Security module in VPN group

As default, the "Tunnel communication only" check box is enabled. If you deselect the check box, in addition to tunnel communication between tunnel partners, communication is also possible with network nodes to which there is no tunnel.

- Communication is untunneled if the partner address belongs to a station known in STEP 7 for which no VPN tunnel is configured.
- Communication is through the tunnel if the partner address is a VPN endpoint.
- If it is not clear whether connection should bypass or run through the VPN tunnel, the connection is assigned to the VPN tunnel and a message to this effect is displayed. The assignment can be adapted in advanced firewall mode, for example, by changing the "From" direction "Tunnel" to "External". To avoid this adaptation being overwritten by the next system synchronization, the "Allow*" or "Drop*" action must be selected.

Note

If you want to ensure that only communication through the tunnel is possible, you will need to create suitable firewall rules in advanced firewall mode, for example, for internal nodes or NDIS addresses.

To allow only tunneled communication for a CP, add a "Drop" > "Any" > "External" rule. For the CP 1628, add a "Drop" > "Station" > "External" rule. In addition to this, you need to remove existing firewall rules that allow untunneled communication.

Activating the Web server on CP x43-1 Advanced

Module-specific function

This function is only available for CP x43-1.

Meaning

After activating the Web server, you have access to the Web pages of the module. In the local security settings, you can restrict access to these Web pages using the HTTPS protocol. This access is controlled using the "Allow access only using HTTPS" check box. In addition, you must configure the firewall accordingly.

IPsec tunnel: Creating and assigning groups

Configuring internal network nodes - "Nodes" entry

Allow access to S7-300 / S7-400 CPs for VPN connection partners

Possible selections

Decide whether or not the VPN connection partner can have access to the CP and/or the internal subnet of the CP in routing mode (SCALANCE S / M).

How to access this function

1. Select the module to be edited.
2. In the local security settings, select the entry "Node".
3. Configure access for the VPN connection partner of the CP in routing mode (SCALANCE S / M):
   - Allow connection to the CP (Gbit interface)
   - Allow connection to the internal subnet (PROFINET subnet)

Configuring NDIS nodes manually for PC CPs that can be reached through the tunnel

Configuring NDIS nodes that can be reached through the tunnel

The internal nodes are learned and assigned to the routes dynamically. This concerns the NDIS IP addresses of the Windows PC.

Follow the steps below

1. Select the module to be edited.
2. In the local security settings, select the entry "Nodes" > "NDIS nodes reachable via tunnel".
3. Enter the NDIS IP addresses.

Online functions - Debug / Diagnostics and Logging

Updated firewall rules - "Dynamically updated firewall rules" entry

Meaning

Display of the IP addresses or IP address ranges that were enabled dynamically using HTTP or HTTPS, or loaded by a user. The rights assigned for accessing the S7 CP are displayed for the enabled IP addresses. An update of the IP addresses in this tab can only be triggered by the following events:

- Extension/modification of the IP access control list
- Update of firewall rules
- Dynamic extensions transmitted to the CP at runtime, for example, PROFINET IO devices

Since only the dynamically updated firewall rules are displayed here, you also need to take into account the firewall rules that were configured offline and downloaded to the station for a full picture of the current firewall status of the module.

Security for S7-1500 CPs

Setting up a firewall

Local firewall rules for S7-1500 CPs

Overview of local firewall rules

Enabling packet filter rules

If you enable the security function for the CPs in the local security settings, initially all access to and via the CP is permitted. To enable individual packet filter rules, select the "Activate

firewall" check box. Then enable the required services. Firewall rules created automatically due to a connection configuration have priority over rules set manually.

Note

Detailed firewall settings in advanced firewall mode

In advanced firewall mode, you can restrict firewall rules to individual nodes. To change to advanced firewall mode, select the "Activate firewall in advanced mode" check box.

Updating connection rules

Changes to the connection configuration of CPs also change the connection-related firewall rules. To display the modified firewall rules, click the "Update connection rules" button. The modified firewall rules are then displayed in advanced firewall mode.

Configuring a firewall with predefined firewall rules - CP 1543-1

Configuring a firewall with predefined IP rules - CP 1543-1

How to access this function

1. Select the module to be edited.
2. In the local security settings, select the entry "Firewall" > "Predefined IP rules".

Table 8-52 Available services and directions

Service	From external to station	Enabled ports	Meaning
Allow IP communication	x	All	IP traffic from external to station is allowed.
Allow S7 protocol	x	TCP port 102	Communication of the nodes using the S7 protocol is allowed.
Allow FTP/FTPS (explicit mode)	x	TCP port 20 TCP port 21	For file management and file access between server and client.
Allow HTTP	x	TCP port 80	For communication with a Web server.
Allow HTTPS	x	TCP port 443	For secure communication with a Web server, for example, for Web diagnostics.
Allow SNMP	x	TCP port 161/162 UDP port 161/162	For monitoring nodes capable of SNMP.
Allow SMTP	x	TCP port 25	For sending e-mails via an SMTP server.
Allow security diagnostics	x	TCP port 8448	Allow/do not allow security diagnostics.

Table 8-53 Logging

Option	Action when activated	Relevant firewall rule		
IP log settings		Action	From	To
Log tunneled packets	Only active if the security module is a member of a VPN group. All IP packets transferred via the tunnel are logged.	Allow	Station	Tunnel
		Allow	Tunnel	Station
Log blocked incoming packets	All incoming IP packets that are discarded are logged.	Drop	External	Station

Note

Relationship between log settings in default mode and firewall rules

Log settings that are made in "Predefined IP rules" and "Predefined MAC rules" have no effect on firewall rules that were automatically created as a result of configuring a connection. This means, for example, that tunneled frames belonging to a configured connection cannot be logged. In advanced firewall mode, logging can be extended to the automatically generated firewall rules of connections.

Configuring a firewall with predefined IPv6 rules - CP 1543-1

Meaning

With the predefined IPv6 rules, you have the option of configuring the firewall for services in which IPv6 is used. By enabling a predefined IPv6 rule in the local security settings of the CP 1543-1 V1.1, the system-defined ICMPv6 services that can be seen in the global security settings in the "ICMP" tab in "Firewall" > "Services" > "Define services for IP rules"are also enabled in the firewall. The firewall of the CP 1543-1 V1.0 allows ICMPv6 packets through even without enabling a predefined IPv6 rule.

How to access this function

1. Select the module to be edited.
2. Select the "Firewall" > "Pre-defined IPv6 rules" item in the local security settings.

Table 8-54 Available services and directions

Service	From external to station	Enabled ports	Meaning
Allow IP traffic	x	All	IP traffic from external to station is allowed.
Allow S7 protocol	x	TCP port 102	Communication of the nodes using the S7 protocol is allowed.
Allow FTP/FTPS (explicit mode)	x	TCP port 20 TCP port 21	For file management and file access between server and client.
Allow HTTP	x	TCP port 80	For communication with a Web server.

Service	From external to station	Enabled ports	Meaning
Allow HTTPS	x	TCP port 443	For secure communication with a Web server, for example, for Web diagnostics.
Allow SNMP	x	TCP port 161/162 UDP port 161/162	For monitoring nodes capable of SNMP.
Allow SMTP	x	TCP port 25	For sending e-mails via an SMTP server.
Allow security diagnostics	x	TCP port 8448	Allow/do not allow security diagnostics via IPv6.

Table 8-55 Logging

Option	Action when activated	Relevant firewall rule		
IP log settings		Action	From	To
Log tunneled packets	Only active if the security module is a member of a VPN group. All IP packets transferred via the tunnel are logged.	Allow	Station	Tunnel
		Allow	Tunnel	Station
Log blocked incoming packets	All incoming IP packets that are discarded are logged.	Drop	External	Station

Note

Relationship between log settings in default mode and firewall rules

Log settings made in "Predefined IPv6 rules" have no effect on firewall rules that were automatically generated as a result of configuring a connection. This means, for example, that tunneled frames belonging to a configured connection cannot be logged. In advanced firewall mode, logging can be extended to the automatically generated firewall rules of connections.

Configuring a firewall with predefined MAC rules - CP 1543-1

How to access this function

1. Select the module to be edited.
2. Select the entry "Firewall" > "Predefined MAC rules".

Table 8-56 Available services and directions

Service	From station to external	From external to station	Enabled ports	Meaning
Allow MAC communication	x	x	-	The MAC traffic from external to the station and vice versa is allowed.
Allow ISO protocol	x	x	-	ISO traffic from external to the station and vice versa is allowed.
Allow SiCLOCK	x	x	-	SiCLOCK traffic from external to the station and vice versa is allowed.

Table 8-57 Logging

Option	Action when activated	Relevant firewall rule		
MAC log settings		Action	From	To
Log blocked incoming packets	All incoming MAC packets that are discarded are logged.	Drop	External	Station
Log blocked outgoing packets	All outgoing MAC packets that are discarded are logged.	Drop	Station	External

Note

Relationship between log settings in default mode and firewall rules

Log settings that are made in "Predefined IP rules" and "Predefined MAC rules" have no effect on firewall rules that were automatically created as a result of configuring a connection. This means, for example, that tunneled frames belonging to a configured connection cannot be logged. In advanced firewall mode, logging can be extended to the automatically generated firewall rules of connections.

IP packet filter directions S7-1500 CPs

Meaning

Possible selections for the communication directions "From" and "To" in the IP rules of the advanced firewall mode.

The following directions are available

Available options / ranges of values		Meaning
From	To	
External	Station	Access from the external network to the station.
Station	External	Access from the station to the external network.
	Tunnel	Access from the station to VPN tunnel partners.*
Tunnel	Station	Access by VPN tunnel partners to the station.*

* Not for the CP 1543-1 V1.0.

MAC packet filter directions S7-1500 CPs

Meaning

Possible selections for the communication directions "From" and "To" in the MAC rules of the advanced firewall mode.

The following directions are available

Available options / ranges of values		Meaning
From	To	
External	Station	Access from the external network to the station.
Station	External	Access from the station to the external network.
	Tunnel	Access from the station to VPN tunnel partners.*
Tunnel	Station	Access by VPN tunnel partners to the station.*

* Not for the CP 1543-1 V1.0.

Connection-related automatic firewall rules

Meaning

For connections that were configured using CPs, STEP 7 automatically creates firewall rules that allow communication with the partner of the CP in the specified direction (CP active/ passive). The connection establishment directions are taken into account. To display these firewall rules, if the advanced firewall mode is enabled, the "Update connection rules" button needs to be clicked. The firewall rules are then displayed in advanced firewall mode.

Note

Enabling UDP multicast and UDP broadcast connections manually

No automatic firewall rules are created for UDP multicast and UDP broadcast connections. To enable the connections, add the relevant firewall rules manually in advanced firewall mode.

Depending on how the connection establishment is configured, the following level 3 firewall rules are created. If the CP 1543-1 V1.1 is in a VPN group, the direction "External" changes to "Tunnel".

The IP address of the connection partner is entered in the "Source IP address" or "Destination IP address" column of these firewall rules.

CP->external	Action	From	To
active	Drop	External	Station
	Allow	Station	External
passive	Drop	Station	External
	Allow	External	Station
active and passive	Allow	External	Station
	Allow	Station	External

For level 2 connections, "Allow" rules are created for both directions. If the CP 1543-1 V1.1 is in a VPN group, the direction "External" changes to "Tunnel".

The MAC address of the connection partner is entered in the "Source MAC address" or "Destination MAC address" column of these firewall rules.

CP->external	Action	From	To
active, passive, active and passive	Allow	Station	External
	Allow	External	Station

Note

Changing the connection configuration

Changes to the connection configuration of CPs also change the connection-related firewall rules. To display the modified firewall rules, click the "Update connection rules" button.

Conventions for automatically created firewall rules

- Priority
  The rules have highest priority and are therefore inserted at the top in the local rule list.
- Deleting rules
  The rules cannot be deleted. Logging can be enabled and services can be assigned. Moreover, you may insert a bandwidth and a comment.
- Changing the action
  If you set the action from "Allow" to "Drop" or vice versa, this is overwritten again during renewed system synchronization. Select "Allow*" or "Drop*" to retain your changes. In this case, only the IP address is synchronized and the action and direction remain as set. Settings for logging, service, bandwidth and comment are also retained after a renewed system synchronization even without changing the action to "Allow*" or "Drop*". If the configured connection is deleted, the corresponding rules are removed from the list.

Security module in VPN group

As default, the "Tunnel communication only" check box is enabled. If you deselect the check box, in addition to tunnel communication between tunnel partners, communication is also possible with devices to which there is no tunnel.

- Communication is untunneled if the partner address belongs to a station known in STEP 7 for which no VPN tunnel is configured.
- Communication is through the tunnel if the partner address is a VPN endpoint.
- If it is not clear whether connection should bypass or run through the VPN tunnel, the connection is assigned to the VPN tunnel and a message to this effect is displayed. The assignment can be adapted in advanced firewall mode, for example, by changing the "From" direction "Tunnel" to "External". To avoid this adaptation being overwritten by the next system synchronization, the "Allow*" or "Drop*" action must be selected.

Note

If you want to ensure that only communication through the tunnel is possible, you will need to create suitable firewall rules in advanced firewall mode.

To allow only tunneled communication for a CP, add a "Drop" > "Any" > "External" rule. In addition to this, you need to remove existing firewall rules that allow untunneled communication.

8.1.4 Creating configurations

8.1.4.1 Information about the web server

Introduction

The web server lets you monitor and administer the CPU through authorized users by means of a network. This permits evaluation and diagnostics over long distances. All you need is a web browser.

Alarms and status information are visualized on HTML pages.

Web browser

You need a web browser to access the HTML pages of the CPU.

The following web browsers have been tested for communication with the CPU:

- Internet Explorer (Version 8)
- Mozilla Firefox (Version 21)
- mobileSafari (iOS5)

Web access to the CPU via PG/PC

Proceed as follows to access the web server:

1. Connect the client (PG/PC) to the CPU via the PROFINET interface.
2. Open the web browser.
   Enter the IP address of the CPU in the "Address" field of the web browser in the format http://ww.xx.yy.zz (example: http://192.168.3.141).
   The start page of the CPU opens. From the start page, you can navigate to further information.

Additional information

Additional information about the web server of the various CPU families is available under the key word "Web server" in the information system.

Information on creating your own web pages for access to the CPU is available under the keyword "User-defined web pages" in the information system.

See also

S7-1500 web server (http://support.automation.siemens.com/WW/view/en/59193560)

Documentation S7-300 (http://support.automation.siemens.com/WW/view/en/12996906)

Documentation S7-400 (http://support.automation.siemens.com/WW/view/en/44444467)

S7-1200 web server (http://support.automation.siemens.com/WW/view/en/36932465)

8.1.4.2 Things you should know about PROFIBUS DP operating modes

Introduction

DP master systems that consist of a DP master and DP slaves which are connected via a bus and communicate with one another via the PROFIBUS DP protocol are referred to as distributed I/O.

Below, we refer to communication-capable modules with DP interface that can take on the role of DP master or DP slave.

"DP master" and "DP slave" option

Communication-capable modules, such as CPUs with DP interface and CPs or CMs with DP interface, have the area "Mode" in their module properties.

For S7-300 CPUs with integrated DP interface, for example, you can set the mode "DP master" and "DP slave". A CPU or a CP that is configured as DP slave is also referred to as intelligent DP slave (I-slave).

For S7-1500 CPUs with integrated DP interface, only the "DP master" mode is possible. To operate S7-1500 CPUs as I-slave, you have to insert the communication module CM 1542-5 and configure it as DP slave.

S7-1200 CPUs do not have integrated DP interfaces. To operate an S7-1200 as DP master or DP slave, you must insert a communication module CM 1243-5 (DP master only) or a communication module CM 1242-5 (DP slave only; I-slave).

Additional information

Additional information on distributed I/O is available under the key word "Distributed I/O" and "I-slave" in the information system.

8.1.4.3 Configuring automation systems (S7-300, S7-400, S7-1500)

Configuring central racks

In the chapters below you will learn how to configure central racks. For this you will also find information on addressing modules.

Overview for configuring modules

Step	Description
1	Select a central module (Page 872).
2	Specify the properties of the modules (Page 875).
3	Specify the input and output addresses (Page 882).

See also

Addressing of modules (S7-1500) (Page 881)

Addressing of modules (S7-300, S7-400) (Page 880)

Rules for S7-300 modules (Page 864)

Rules for S7-400 modules (Page 865)

Introduction to configuring hardware (Page 494)

Rules for S7-300 modules

Introduction

Specific slot rules apply for S7-300 modules.

When you select a module in the device view of the hardware catalog, all possible slots for the selected module are marked in the rack. You can only drag modules to marked slots.

If you insert, move or swap a module, the slot rules are also applied.

Slot rules for S7-300

In addition to the general slot rules, the following rules also apply for S7-300:

- You can use up to 4 racks.
  Rack 0:
  - Slot 1: Only power supply (e.g. 6ES7 307-...) or empty
  - Slot 2: Only central rack (e.g. 6ES7 314-...)
  - Slot 3: Interface module (e.g. 6ES7 360-.../365-...) or empty
  - Slot 4 to 11: Signal modules or function modules, communication processors, or empty

  Racks 1 to 3:
  - Slot 1: Only power supply module (e.g. 6ES7 307-...) or empty
  - Slot 2: Empty
  - Slot 3: Interface module (e.g. 6ES7 361-.../365-...)
  - Slot 4 to 11: Signal module or function module, communication processors, (depending on the interface module used) or empty
- In rack 3 when using a CPU with integrated I/O you cannot use slot 11.
- You must mount modules side-by-side without gaps.

Note

Slot 3 in S7-300 is reserved for the interface module. If you use a single rack, this slot will not be assigned because you do not have to plug an interface module. You do not have a gap in the real structure because otherwise the backplane bus would be interrupted!

An ET200m with active backplane bus can have gaps in the allocation.

Special rules for DM 370 dummy module

You can insert the DM 370 dummy module (6ES7 370-0AA01-0AA0) in place of a module that will be used later.

The DM 370 dummy module reserves address space of 1 byte for a module in the modular DP slave (e.g. for a digital I/O module).

See also

General slot rules (Page 496)

Rules for S7-400 modules

Introduction

Specific slot rules apply for S7-400 modules. The rules for the assignment of components in an S7-400 rack depend on the type of rack used.

When you select a module in the device view of the hardware catalog, all possible slots for the selected module are marked in the rack. You can only drag modules to marked slots.

If you insert, move or swap a module, the slot rules are also applied.

Slot rules of the S7-400 for central racks

In addition to the general slot rules, the following rules for components on a central rack apply for S7-400:

- Only insert power supply modules in slot 1 (exception: redundant power supply modules)
- insert a max. of 6 interface modules (transmission IMs); max of 2 with power transmission
- connect a max. of 21 expansion racks to the central racks through interface modules
- connect a max. of 1 expansion rack **with power transmission** to an interface of the transmission IM (IM 460-1 with IM 461-1)
- max. of 4 expansion racks **without power transmission** (IM 460-0 with IM 461-0 or IM 460-3 with 461-3)

Slot rules of the S7-400 for expansion racks

In addition to the general slot rules, the following rules apply for components on extension racks for S7-400:

- Only insert power supply modules in slot 1 (exception: redundant power supply modules)
- insert the interface module (recipient IM) only on the outer right slot (slot 9 and slot 18)
- Insert K-bus modules only in expansion racks with a number no larger than 6 (otherwise it cannot be accessed)

Special rules for redundant power supply modules

Redundant power supply modules can be inserted into a rack twice. You can find out if the power transmission module is redundant by looking in the "Information" palette on the "Hardware Catalog" task card.

The following rules must be observed:

- You can only insert redundant power supply modules into the designated racks (recognizable from the informational text in the "Hardware Catalog" task card).
- The redundant power transmission modules must be inserted into slot 1 and the connecting slot (no gaps allowed!).
- Redundant and non-redundant power transmission modules may not be inserted in the same rack (this means that no "mixed operation" is possible).

See also

General slot rules (Page 496)

Rules for S7-1500 modules

Use of S7-1500 modules

Introduction

The CPUs of the S7-1500 automation system are modular controllers for medium- and high-performance automation system solutions for discrete and continuous processes.

Application area

The S7-1500 automation system is designed for system solutions in process and manufacturing automation. As a universal automation system, the S7-1500 automation system represents the optimal solution for centralized and decentralized design applications.

The scalable structure allows you to exactly align your control system's design to the specific needs of your location.

The S7-1500 automation system complies with IP 20 degree of protection and is intended for installation in a control cabinet.

Structure

The S7-1500 automation system is installed on a mounting rail and can comprise up to 32 modules. Bus connectors are used to interconnect the modules. Power is supplied via the backplane bus; the CPU itself can also supply power to several modules. If the power consumed by the modules exceeds the amount supplied by the CPU, you can insert up to three additional power supply modules.

See also

Slot rules for S7-1500 modules (Page 867)

Configuration of an S7-1500 station (Page 868)

Power budget calculation (Page 870)

Checking the power budget (Page 871)

Slot rules for S7-1500 modules

Specific slot rules apply to S7-1500 modules. When you select a module in the device view of the hardware catalog, all possible slots for the selected module are marked in the rack. You can only drag modules to marked slots. The slot rules apply whenever you insert, move or replace a module.

Slot rules for S7-1500

The following slot rules apply to S7-1500:

- In addition to the CPU, you can also insert up to 31 modules on one rack.
- You can insert a maximum of three power supply modules. The power consumption of the individual modules determines how many power supply modules are required, at a minimum.
  - One power supply module can be inserted to the left of the CPU (slot 0). Together with the CPU, this module supplies power to all downstream devices on the rack. The power supply module and modules supplied by it form a power segment.
  - Up to two additional power supply modules can be inserted on the slots to the right of the CPU. If a power supply module is not installed on slot 0, up to three power supply modules can be inserted to the right of the CPU.

Note

Power segments

A power supply module and modules supplied by it form a power segment. A power supply module always supplies power to the modules installed on its right. The power segment ends at the next power supply module and a new power segment begins for the next power supply module and the modules installed on its right. The modules located on the left of the power supply module are either supplied from the upstream power supply module or the CPU.

See also

Power budget calculation (Page 870)

Configuration of an S7-1500 station (Page 868)

Configuration of an S7-1500 station

The individual S7-1500 CPUs can be expanded with a variety of I/O modules. Besides analog and digital input and output modules, you can use technology modules, communication modules and special modules. In spite of their high performance, S7-1500 CPUs are subject to limits on the number of I/O modules and configured IO systems in situations where the hardware configuration is very large. These limits vary according to the CPU used. When you configure your hardware you should therefore observe the limits described in the technical specifications sheet for the respective CPU.

Maximum configuration of a station

When the hardware configuration is compiled, a check is made to determine whether the station configuration exceeds maximum limits. However, you are advised to make sure that any quantity limitations are met when you are configuring the S7-1500 station. In particular,

when creating your hardware configuration you should check the technical specifications for your devices for the following criteria:

- Maximum number of communication modules for PROFINET, PROFIBUS and Industrial ETHERNET as well as communication processors
  In the case of an S7-1513 (6ES7 513-1AL00-0AB0), for example, the maximum number allowed is six.
- Maximum number of distributed PROFINET IO systems and distributed PROFIBUS DP systems.
  IO systems that are connected via an I-device do not have to be counted toward the maximum number. In this case, the rules for the maximum configuration of the I-device apply.
- Maximum number of distributed I/O devices (IO devices and DP slaves)
  The number includes all distributed I/O devices that are assigned to the CPU, regardless of the interface to which they are connected (integrated interface or CP/CM interface). The CPU is only included in the IO devices if it is configured as an I-device.
  If I-devices are connected to the PROFINET IO system, the IO devices on an I-device's lower-level PROFINET IO system are not counted.
- Maximum number of modules/submodules
  The maximum number of pluggable modules is very large and is only rarely reached in conventional hardware configurations of an S7-1500 automation system.
  If modules with submodules are used in very large installations, the CPU's available resources for modules/submodules can be exceeded.
  Simple I/O modules, which have no submodules, require a single resource.
  Complex modules can have multiple submodules and require one resource for each submodule. A module with submodules also requires an additional resource (which represents the whole module).
  Example:
  An IM 155-5PNHF interface module requires five resources based on the following method of counting:
  Two resources for the PROFINET interface's two ports, one resource for the interface of the PROFINET interface, one resource for the submodule for the interface module's parameters. As explained above, an additional resource is added because the module has submodules.

Example

The following figures show an example of a hardware configuration and illustrate the counting method described above:

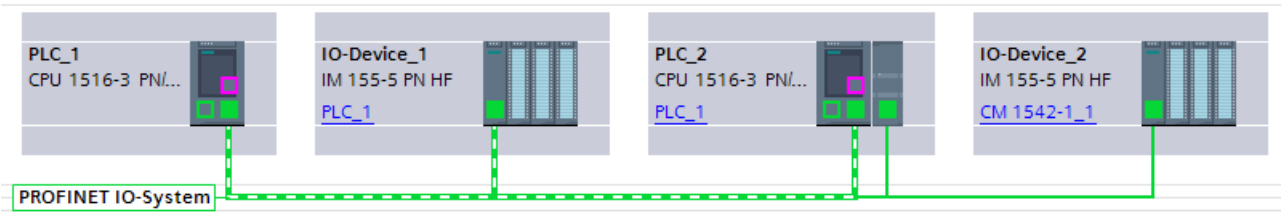


Figure 8-7 In this figure, PLC_1 is configured as the IO controller. IO-Device_1 and PLC_2 are devices on the IO system of PLC_1. Although IO-Device_2 is connected to the same network, it is not a part of the IO system. The IO system of PLC_1 therefore has two devices.

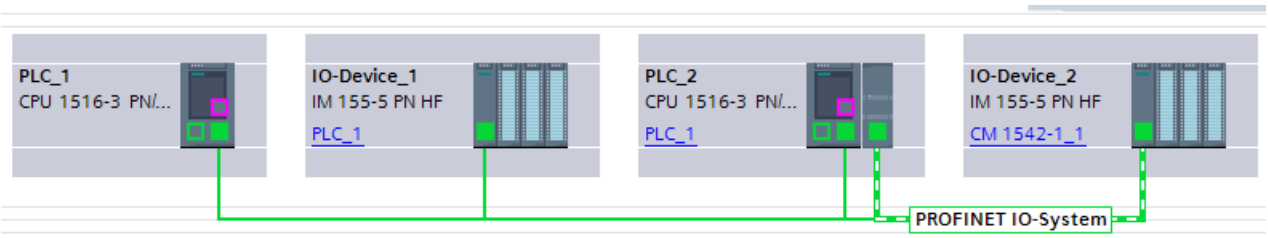


Figure 8-8 The second figure shows that while PLC_2 is an I-device in the IO system of PLC_1, it is simultaneously also an IO controller. A lower-level IO system is spanned via the communication module and this IO system is connected to IO device_2. In total, two devices are counted for PLC_2.

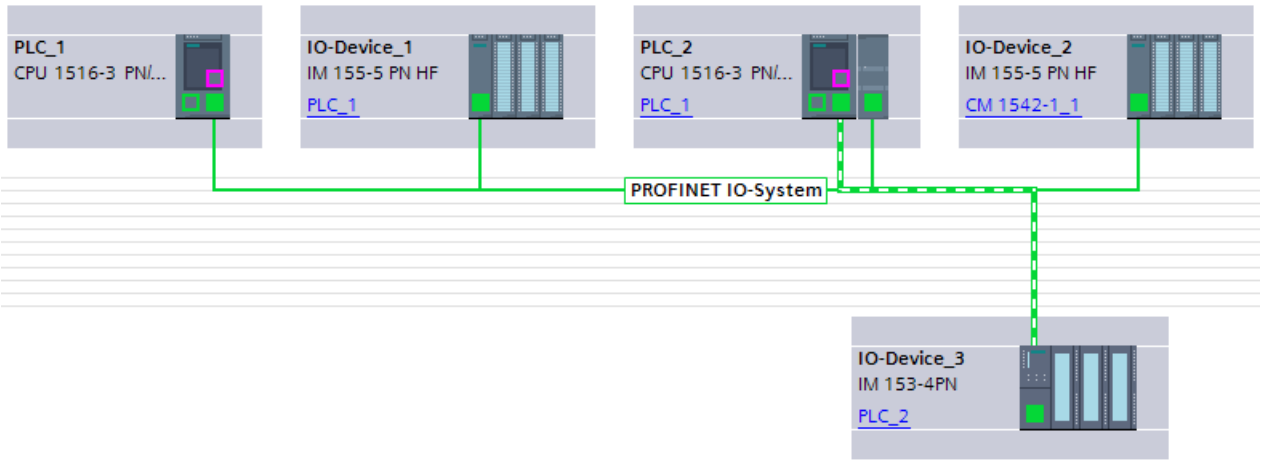


Figure 8-9 In the third figure, an additional IO system is assigned to PLC_2. PLC_2 now has two IO systems, one connected to the internal interface and one connected to the communication module. Another device must be counted for the new IO system. In total, three devices are counted for this hardware configuration for PLC_2 - one I-device (PLC_2 is assigned to PLC_1) and two I/O devices (assigned to PLC_2).

See also

Slot rules for S7-1500 modules (Page 867)

Power budget calculation (Page 870)

Power budget calculation

Introduction

All I/O modules draw the power needed to control the electronics from the backplane bus. This power is fed from the CPU and, if required, also by additional system power supplies to the backplane bus.

Power supply/consumption ratio check while configuring

STEP 7 helps you to check the power supply/consumption ratio. All the inserted I/O modules and their maximum power consumption are listed in a table. The power consumption is offset

by the plugged system power supply modules. Thus, you see at a glance whether sufficient power is available for all modules.

You can find the table in the "System power supply" section of the Inspector window after you have selected the CPU and then selected the "Properties" tab.

In addition, a check is made during the configuration to determine if sufficient power is available. If the I/O modules consume more power from the backplane bus than is supplied, the corresponding table cells are colored red and you receive a corresponding warning after the hardware configuration is compiled.

Power supply/consumption ratio check by the CPU

During operation, the CPU monitors compliance with the power supply/consumption ratio:

- At every Power ON
- At every change of the installed hardware

If the CPU identifies a backplane bus overload, it switches off all I/O modules that are supplied by the CPU or the system power supply and enters an alarm in the diagnostic buffer. The CPU does not start up while the supply/consumption ratio is negative.

Causes of overload

An overload may be caused by a mismatch between the installed and configured hardware. For example, the system power supply units required for operation are not inserted or are not switched on or there are more modules inserted than configured.

See also

Checking the power budget (Page 871)

Slot rules for S7-1500 modules (Page 867)

Configuration of an S7-1500 station (Page 868)

Checking the power budget

You can see whether sufficient power is available for all I/O modules as you are configuring a station. To do so, you must first specify whether or not the CPU is supplied with voltage.

Procedure

To activate the power supply of the CPU and evaluate the power supply/consumption ratio, follow these steps:

1. Select the S7-1500 CPU, for example, in the network view.
2. Open the "Properties" tab in the inspector window.
   The properties of the CPU are displayed.
3. Select the "System power supply" item in the area navigation.

4. Select the "Connection to supply voltage L+" option if the CPU is supplied with voltage. The option is activated by default. As a result, the power consumption of the CPU is included in the power supply/consumption ratio.
5. Check the "Power segment overview" table to find out if the power budget is positive. If the power supply/consumption ratio is negative, the underpowered modules is marked in red.

See also

Power budget calculation (Page 870)

Selecting modules

Introduction

Select a CPU from the hardware catalog and place it, together with a rack, in the network view. On this device drag the desired modules from the hardware catalog; they are arranged automatically on the rack.

Selecting the components in the hardware catalog

Each hardware component is displayed as a folder in the hardware catalog. When you open this folder you will see the different versions of the selected hardware component with its respective order numbers.

There will be an example of how to set up a CPU with a rack in network view.

Requirement

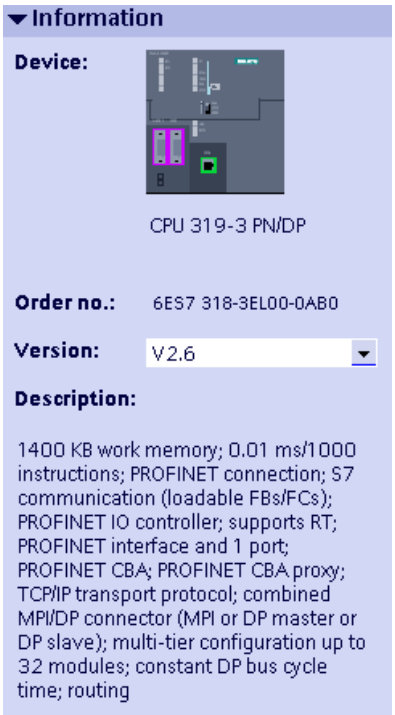
- The hardware catalog is open.
- You must be in the network view.

Procedure

To select a CPU from the hardware catalog, proceed as follows:

1. In the hardware catalog navigate to the folder with the desired CPUs.
2. Open the folder with the desired CPU type; you will see all order numbers for the selected CPU type.

3. Click on a CPU order number to get information about the selected CPU under "Information" pane.



4. Set the matching firmware version of your CPU in the drop-down list box "Version".
5. Set up the CPU and a rack. You have the following options:
   - Use drag-and-drop to drag the CPU from the hardware catalog into network view.
   - Double-click the CPU entry in the hardware catalog.

See also

Adding a device to the hardware configuration (Page 504)

Inserting a module into a rack (Page 507)

Working with racks (Page 498)

Browsing the hardware catalog (Page 497)

Selecting fail-safe modules

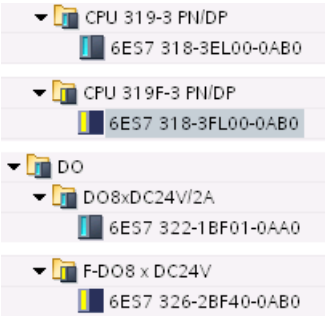
Introduction

Fail-safe automation systems (F systems) are used to control processes with a safe state that can be achieved immediately on shutdown. F systems control processes that can be shut down immediately without endangering human beings or the environment.

Selecting the F components in the hardware catalog

Each hardware component is displayed as a folder in the hardware catalog. Fail-safe hardware components are displayed with yellow symbols in the hardware catalog in contrast to the blue hardware components that are not fail-safe.

The following figure shows excerpts from the hardware catalog with two CPUs and two digital output modules, normal in blue and F in yellow:

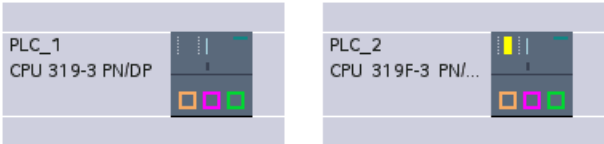


Based on this color coding, you can identify F components in the hardware catalog simply and quickly.

Display of the F components in the hardware and network editor

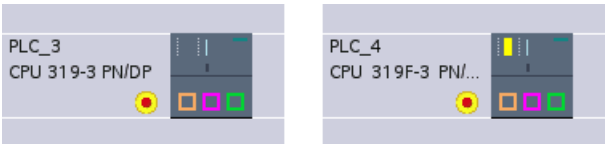
If you use fail-safe devices and "normal" devices in the graphic views of the hardware and network editor, you can recognize them simply by the yellow color-coding of the devices.

The following figure shows two CPUs. The F CPU is shown in yellow:



Fail-safe modules in racks or devices can also be recognized in the device view due to their yellow color coding. Since modules that are not capable of networking are not displayed in the network or topology view, devices with F components without networking capability are represented by a special symbol in the network and topology view.

The following figure shows a "normal" CPU and an F-CPU in the network view that both contain F components:



Based on the color coding of the fail-safe devices and modules, you can also identify the F components quickly in the graphic views of the hardware and network editor.

Changing the properties of CPUs

Each hardware component (module, interface, or interface module) has pre-selected properties, e.g. for analog entry modules pre-selected measurement types and measurement ranges.

Properties of CPUs

The properties of the CPUs have special significance for system behavior. For example for a CPU you can set:

- Startup characteristics
- Local data storage areas
- Priorities for interrupts
- Memory areas
- Magnetic latching
- Clock memory
- Protection level
- Password

The entry possibilities specify what is adjustable and in which value ranges. Fields that cannot be edited are disabled or are not shown in the properties window.

You can assign parameters for an integrated MPI interface or PROFIBUS DP interface in the properties of an interface, for example, and connect them with an existing subnet.

Requirement

You have already arranged the hardware components for which you want to change properties, on a rack.

Procedure

To change the properties and parameters of the hardware components, proceed as follows:

1. In the device or network view, select the hardware components that you want to edit. You can also select individual parts of a hardware component, e.g.:
   - Interface
   - Input
   - Output
2. Edit the settings for the selected object:
   - For example in the device view you can edit addresses and names.
   - In the Inspector window additional setting possibilities are available.

You do not need to confirm your entries, the changed values will be applied immediately.

Additional possibilities for parameter assignment

For the S7-300/400 automation systems, for example, you have the possibility of setting parameters for analog modules in the user program. To do this, open the system functions WR_PARM (Page 6600) (SFC 55), DP_DPARM (Page 6599) (SFC 56) and PARM_MOD (Page 6596) (SFC 57) in the user program. However these settings will be lost at start-up or restart (warm restart).

Additional information

For additional information about system functions and standard functions for S7-300/400, refer to the system software reference manual for S7-300/400.

See also

Editing properties and parameters (Page 513)

Linking racks (SIMATIC S7-300)

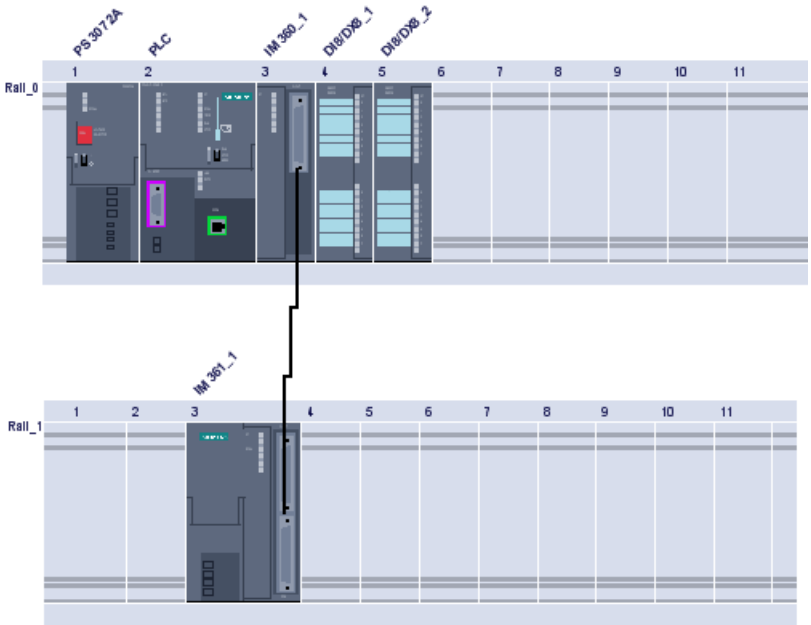
Introduction

Depending on the device used, the option exists to link racks using interface modules for some S7-300 CPUs.

Linking racks

To link racks through an interface module, another rack must be moved from the hardware catalog into the device view. Then a sender module is plugged into the central rack and a

receiver module into the new rack. A link is automatically established between sender and receiver modules.



With S7-300, for example, you can link up to four racks. The interface modules are always plugged in slot 3.

Note

If your configuration does not include an expansion, slot 3 will always be identified as empty even if the module in slot 4 is actually connected directly to the CPU and there is no gap.

Use the following interface modules:

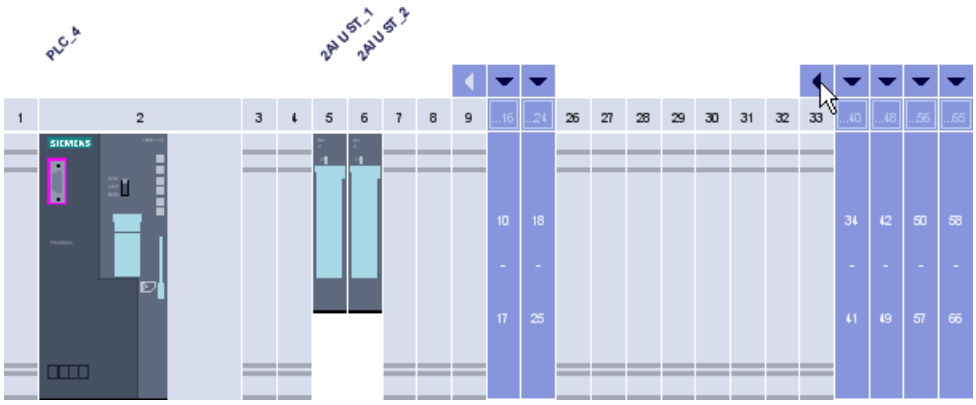
- To expand by one rack precisely:
  Racks 0 and 1: IM 365
- To expand by up to 3 racks:
  Rack 0: IM 360
  Racks 1 to 3: IM 361

If you prefer a manual link, first delete the connection line representing the link. Then press and hold down the mouse button and draw a new connection line between the interfaces of the sender and receiver modules.

Expandable racks

Depending on the device used, several CPUs (e.g. IM 151-1) have expandable racks whose groups of slots can be expanded and collapsed using the arrow symbols above the expandable slot. When the groups of slots are collapsed, the first and last of the group's slot numbers are displayed.

The following diagram shows a section of an expanded rack:



Groups of slots into which modules have already been plugged cannot be collapsed. The first group of slots and device cannot be collapsed. You can tell from the grayed out arrow symbol if a group of slots cannot be collapsed.

See also

Working with racks (Page 498)

Linking racks (SIMATIC S7-400)

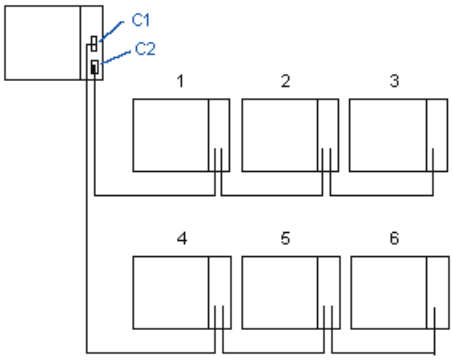
Expanding a central rack by adding an expansion rack

Configuring expansion racks in SIMATIC 400

In SIMATIC 400, the expansion options are more complex due to be various racks and interface modules.

All expansion racks connected to an interface of the IM send module of the central rack form a chain.

The following figure shows two groups of three expansion racks, with each group connected to an interface of the IM send module.



Rules for linking expansion racks

Linking expansion racks

If you link expansion racks (SIMATIC 400) to an interface of the interface module (IM send module) of the central rack, the following properties of the IM send module and IM receive module must match:

- Power transfer (with/without)
- Type of link (central/distributed)
- K bus transfer (with/without interrupt transfer)

Slot rules of the S7-400 for expansion racks

In addition to the general slot rules, the following rules apply for components on extension racks for S7-400:

- Insert power supply modules in slot 1
- Insert the interface module (receive IM) only in the outer right-hand slot (slot 9 or slot 18)
- Insert K-bus modules only in expansion racks with a number no larger than 6 (otherwise it cannot be accessed)

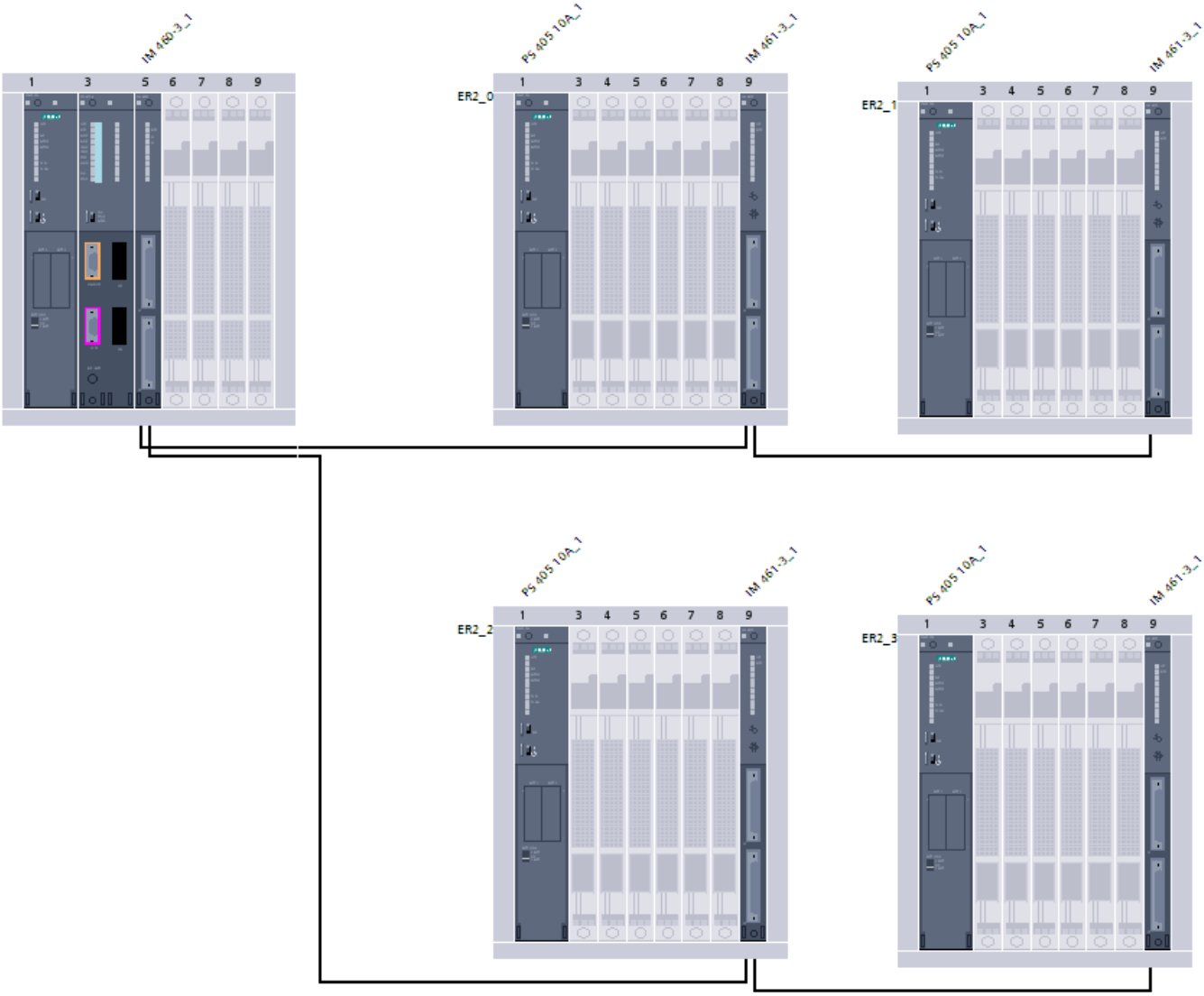
Arrangement of the expansion rack

Basic procedure

To arrange expansion racks and drag connections between the interface modules in the racks, follow these steps:

1. Select a CPU rack (for example CPU 417) from the "hardware catalog".
2. Drag the controller to the device view.
3. Assign the controller modules (for example, power supply and IM 460) to the rack.
4. Select suitable expansion racks (for example ER1) from the "hardware catalog".
5. Drag the expansion racks one after the other to the device view.
6. Assign modules (for example, power supply or IM 461) to each rack.
   Important: The interface modules must be inserted in all racks to allow you to link racks together.
7. Connect the interface of the IM send module with the iterface of the IM receive module of the required rack.
   Connection lines then show the assignment of the racks among themselves.

The following figure shows groups of two expansion racks, with each group connected via a receive IM to an interface of the send IM.



Addressing modules

Addressing of modules (S7-300, S7-400)

Introduction

In the device overview, you see the addresses or address ranges of the modules in the I address and Q address columns.

Input and output addresses are assigned automatically when modules are inserted in the rack. The address of the first channel is the start address of a module. The addresses of the other channels are derived from this start address. The end address is derived from the module-specific address length.

Special feature: Input addresses with asterisk (*)

In the device view certain addresses from the input address range are marked with an asterisk (*). In this case these are diagnostics addresses, e.g. interface modules that do not address user data, but rather that are used for system diagnostics.

Difference between device addresses and I/O addresses

When assigning addresses a distinction is made between device addresses and I/O addresses (input/output addresses):

- Device addresses are addresses of programmable modules (MPI addresses, PROFIBUS addresses, Industrial Ethernet addresses). They are required to address the different stations of a subnet, for example, to download a user program to a CPU.
- I/O addresses are used to read inputs or set outputs in the user program.

Special feature: MPI addresses of FMs and CPs (S7-300)

For CPs and FMs with own MPI address, the default MPI address is set by STEP 7.

The MPI address can be changed subsequently (in the inspector window via "Properties > MPI interface").

For CPs and FMs without own MPI address, the CPU addresses these modules via the K bus using the module rack and slot number.

See also

Specifying input and output addresses (Page 882)

Assigning addresses to a location in the program (Page 893)

Addressing of modules (S7-1500)

Introduction

In the device overview, you see the addresses or address ranges of the modules in the I address and Q address columns. There are other addresses as well, which are explained below.

I/O address

I/O addresses (input/output addresses) are required to read inputs and set outputs in the user program.

Input and output addresses are assigned automatically when modules are inserted in the rack. The address of the first channel is the start address of a module. The addresses of the other channels are derived from this start address. The address end is obtained from the module-specific address length.

Device address (e.g., Ethernet address)

Device addresses are addresses of programmable modules (Industrial Ethernet addresses). They are required to address the different stations of a subnet, for example, to download a user program to a CPU.

Hardware identifier for identifying modules and submodules

In addition to the I addresses and Q addresses, a hardware identifier (HW ID) is assigned automatically and is used to address and identify the module. Submodules (units of a module), such as integrated counters, also receive a hardware identifier. The hardware identifier consists of an integer and is output by the system with diagnostics alarms to allow the faulty module or the faulty submodule to be localized. In addition, the hardware identifier is used for a number of instructions to address the corresponding module. The hardware identifier cannot be changed.

See also

What you need to know about HW identifiers (Page 883)

Inspector window (Page 487)

Specifying input and output addresses

The input and output addresses are indeed preset automatically, however you can change the addressing retroactively.

Requirement

- You are in the device view.
- The module is a compact DP slave or is inserted in one of the following objects:
  - Central rack
  - Expansion rack
  - DP slave
- The CPU allows free address assignment.

Procedure

To change the preset address range proceed as follows:

1. In the device view, click on the module for which you want to set the start address.
2. Click on "I/O Addresses" under "Properties" in the Inspector window.
3. Under "Start address" enter the required start address.
4. Press <Return> or click on any object to accept a modified value.

If you have entered an invalid address a message will be shown in which you can accept a suggested address or enter a different address.

Note

You can also change the addresses directly in the device overview. If address specification is not enabled by the CPU, the address will be disabled in the display.

For modules within a local bus segment, formed by an FM (S7-300) or for special FMs (S7-400), additionally assign another start address. In addition to the start address for the CPU, the module will then also have a start address for the FM. In the device overview in this case the start address is always displayed from the perspective of the FM.

See also

Editing properties and parameters (Page 513)

What you need to know about HW identifiers

Introduction

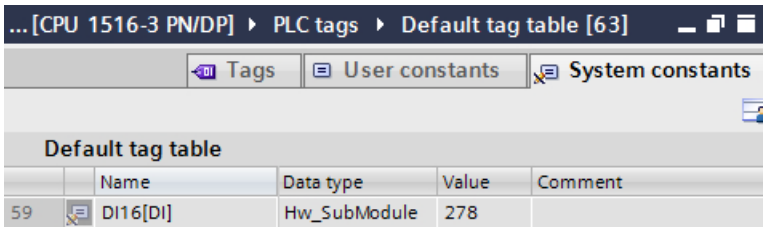
Every hardware object, e.g., a module, interface or port of a module, or even transfer areas of I-devices, has a HW identifier that STEP 7 assigns automatically when a hardware component is configured.

In S7-1500 and S7-1200 systems, these HW identifiers have a similar function as the diagnostic addresses of HW components without user data or as the start addresses of HW objects with user data in S7-300/400 systems. The HW identifiers of the modules remain the same even if the I/O addresses are changed. When I/O addresses are changed, the instructions that use a hardware identifier do not have to be adapted.

HW identifiers in the component properties

For the HW objects indicated above, an integer value is displayed in the Inspector window, "General" tab (HW identifier area).

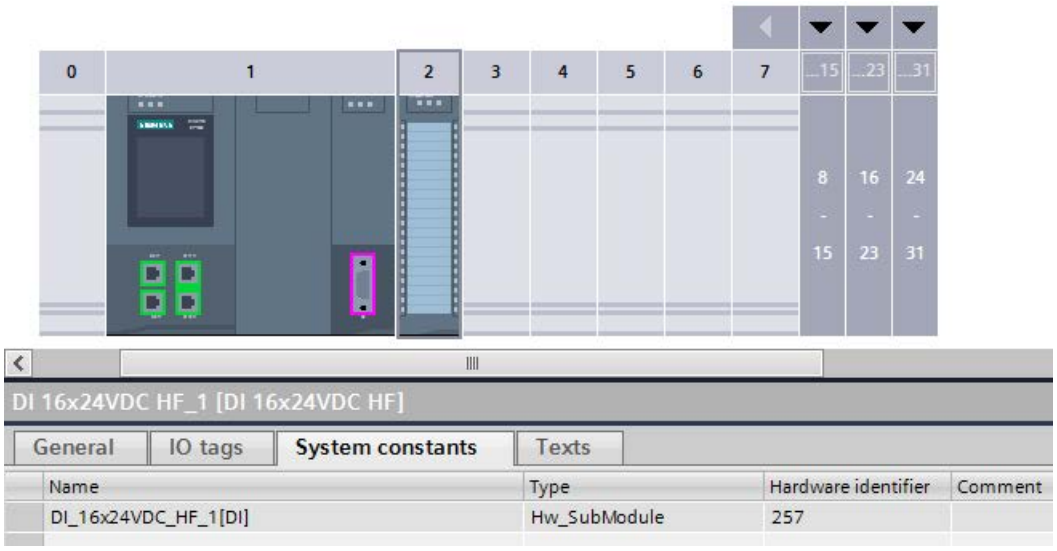
Example:



HW identifiers in the "System constants" tab in the Inspector window

When you select a hardware object in the device view and select the "System constants" tab in the Inspector window, all HW identifiers of the hardware object are displayed.

Example:



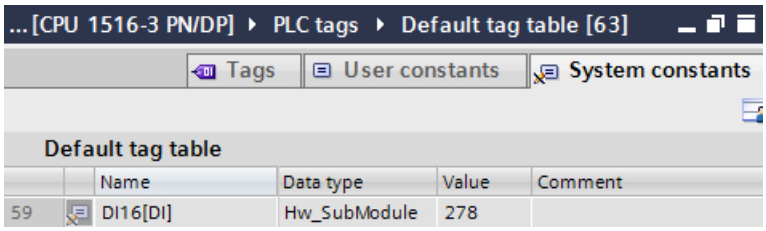
HW identifiers in the system constants of the PLC tag table

In addition, STEP 7 adds entries to the PLC tags ("System constants" tab) for each configured hardware component. These entries are read-only. The name, hardware data type and, in the comment, the path of the HW components are assigned in addition to the integer.

The name matches the editable name in the properties of the component (inspector window).

Recommendation: To facilitate identification of HW objects, assign the most descriptive names possible for HW objects when configuring the hardware. The names should include the association of the HW object to a plant unit or device.

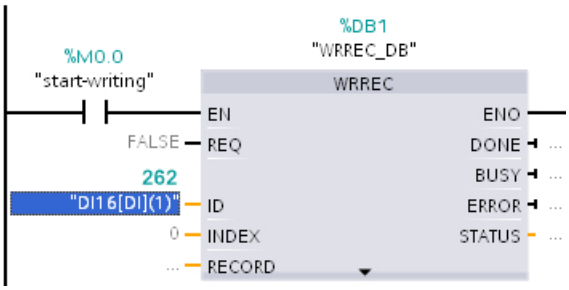
Example:



Hardware identifiers in programming

If you address a HW component in an instruction with the "ID" or "LADDR" input parameter, for example, you can double-click this input parameter to display a list of all relevant HW components for selection.

In the start information of a hardware interrupt OB, for example, you can identify the interrupt-triggering module through the hardware identifier.



See also

Hardware interrupt OBs (Page 1902)

Example of a hardware interrupt event (Page 1856)

Example for the use of hardware identifiers (Page 885)

Inspector window (Page 487)

Example for the use of hardware identifiers

You use hardware identifiers to address hardware objects such as modules, submodules or devices. The hardware identifiers have different hardware data types, depending on the instruction. The permitted hardware data type is listed for all configured components in the default tag table, "System constants" tab.

Below, you will find an explanation of the hardware identifiers that have to be used for selected instructions.

Note

Information on the respective hardware identifier

For each extended instruction that addresses hardware objects, the hardware object that has to be used is documented in the help. You access the help for the selected instruction with the <F1> key.

Recommendation

You can enter the name of the hardware object or the numerical hardware identifier of the hardware object in the Program editor.

Recommendation: Assign the most descriptive names possible for hardware objects, for example, for the modules. Read out the name and HW identifier from the "System constants" tab in the device view and enter it in the Program editor at the corresponding instruction. If you have assigned a descriptive name, you will will the name more easily in the drop-down list to select the HW identifier at the block parameter "ID".

Additional options for input of the HW identifier

You can drag the HW identifier from the tag table to the block parameter "ID" with drag-and-drop.

Advantage: The hardware data type is checked by the Program editor and may be rejected if an invalid hardware object was selected.

No check is possible with a numerical input.

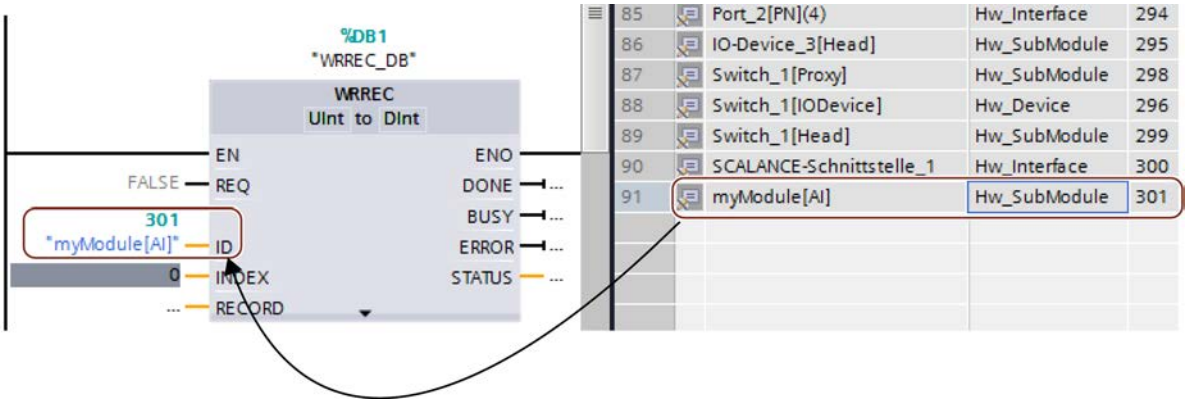
If the permitted hardware object is an IO device, for example, select the entire IO device in the network view and read out the hardware identifier under "Properties" ("General" tab) in the inspector window. If you do not select the IO device in this case but rather the interface module in the IO device, the hardware identifier visible in the inspector window would be invalid for this instruction. You will not notice this error during programming with numerical input of the HW identifier.

Example of entering a hardware identifier

The following name of a module may be listed in the default tag table, "System constants" tab, for example:
`myModule[AI]`

1. Open the programming editor
2. Open the default tag table ("PLC tags" folder)
3. Place the editor window of the programming editor and the default tag table next to each other.
   To do so, click the "Split editor space vertically" button in the toolbar.

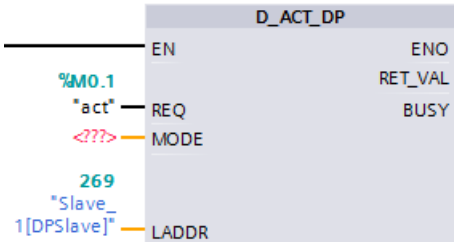
4. Drag the "myModule[AI]" from the "System constants" tab and drop it in the "ID" or "LADDR" box of the respective instruction with drag-and-drop as seen in the figure below.



Example for addressing with hardware identifier

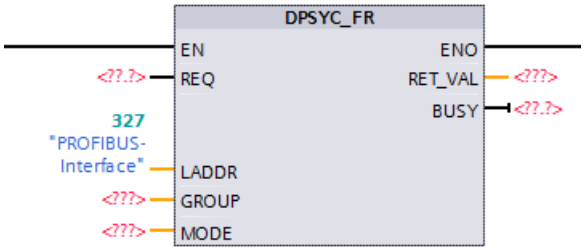
Extended instruction D_ACT_DP: Deactivate/activate the DP slave or PROFINET IO device.

The hardware data type is Hw_DpSlave or Hw_Device. DP slaves or IO devices are permitted hardware objects for the input parameter "LADDR".



Extended instruction DPSYC_FR: Synchronize/freeze DP slaves.

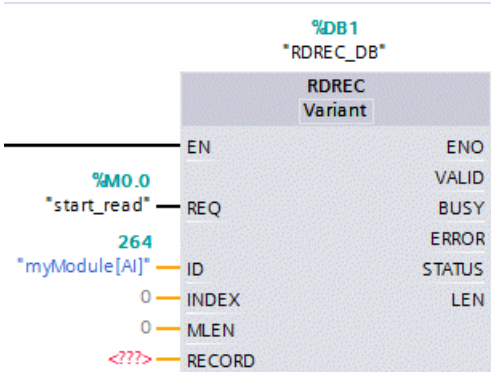
The hardware data type is Hw_Interface. The DP master interface is the permitted hardware object for the input parameter "LADDR".



Extended instruction RDREC: Read data record.

The hardware data type is Hw_SubModule. Modules or submodules from which the data record is to be read are permitted hardware objects for the input parameter "ID".

You also use this HW data type for the extended instruction DPRD_DAT.



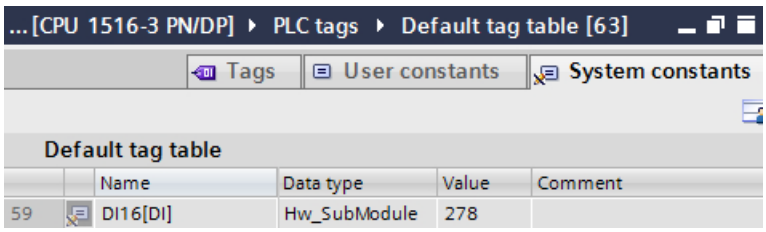
Extended instruction WRREC: Write data record. The instruction is used, for example, to write data records to a module or submodule.

The hardware data type is also Hw_SubModule. Permitted hardware objects for the input parameter "ID" are modules or submodules to which the data record is to be transferred.

You also use this HW data type for the extended instruction DPWR_DAT.

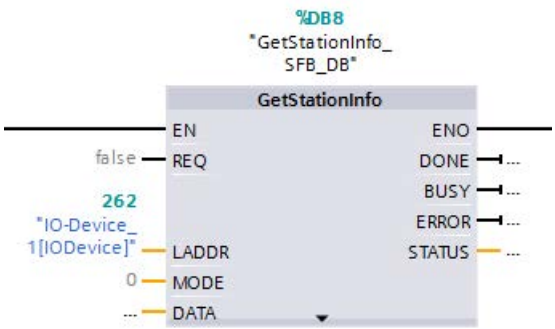
Notes:

- When dividing an I/O module into several submodules (e.g., 1 x 8 AI => 8 x 1AI), use the hardware identifier of a submodule. Do not use the hardware identifier of the module in this case.
- Use the hardware identifier of the head for interface modules of a distributed I/O system to which a data record is to be transferred for configuration control (option handling), see figure.



Extended instruction Station_Info: Reading information of an IO device.

The hardware data type is Hw_Device. An IO device is the permitted hardware object for the input parameter "ID".



See also

What you need to know about HW identifiers (Page 883)

Input and output addresses in the address overview

Introduction

The currently used input and output addresses can be displayed in the address overview in a table form. The address overview can be found in the inspector window under "Properties" of the CPU.

Design of the address overview

With the different check boxes, you can set which objects should be displayed in the address overview:

- Inputs: Display of the input addresses
- Outputs: Display of the output addresses
- Address gaps: Display of open address spaces
- Slot: Display of the slot number

The following information is typically shown in the address overview:

Table header	Meaning
Type	Indicates whether the area is an input address area or an output address area.
Address from	Start address in the address range.
Address to	End address in the address range.
Module	Modules using the address area.
PIP	Number of the process image partition. Shows if the cyclical OB in the process image partition.
OB	Assigned organization block
DP	Number of the master system. You can use the number to determine which slaves are assigned to a master. The value in brackets specifies the PROFIBUS address of the hardware component.
PN	Number of the IO system. The value in brackets stands for the device number of the hardware component.
Racks	Rack number on which the hardware component is inserted.
Slot	Slot number on the rack in which the hardware component is inserted.

Assigning process image / process image partition (S7-300, S7-400)

Introduction

For S7-300/400, if input addresses and output addresses are contained in a process image, they can either be in the OB1 process image (OB1-PI) or in a process image partition (PIP).

A process image offers the advantage that you can access a consistent image of process signals during cyclic program execution. For S7-300/400, you can choose between "OB1-PI" and "PIP 1" to max. "PIP 15".

Each input/output address that you have assigned to a process image partition with STEP 7 no longer belongs to the OB 1 process image input/output! Input/output addresses can be assigned only once across the OB 1 process image and all process image partitions.

You define the process image partition during address assignment (which input/output addresses of modules are managed in which process image partition). The process image partition is updated either by the user with the help of instructions or automatically by the system by connection to an OB. Exception: process image partitions of isochronous mode interrupt OBs are not updated by the system, even though they are connected to an OB (OB 61 to OB 64).

You can update each process image partition in the user program with special instructions. Use "UPDAT_PI (Page 6452)" (SFC 26) for the process image partition of inputs and "UPDAT_PO (Page 6454)" (SFC 27) for the process image partition of outputs.

You assign the connection of a process image partition to an OB in the properties of the CPU ("Interrupts" area).

Note

The option can only be selected for modules with inputs or outputs. In addition, the CPU must support process image partitions. The number of available process image partitions also depends on the CPU.

Requirement

- You are in the device view.
- The module is a compact DP slave or is inserted in one of the following objects:
  - Central rack
  - Expansion rack
  - DP slave
- The CPU allows free address assignment.

Procedure

To change the preset process image range, proceed as follows:

1. In the device view, select the module whose start address you want to set.
2. Click on "I/O Addresses" under "Properties" in the Inspector window.
3. Under "Process image," select the desired process image.

Note

If the entry "- - -" is visible in the "Process image" field (not changeable), this means that a process image is not available for the specified address range.

- For digital input or output modules, select a lower start address.
- If an analog module is to be addressed in a process image, then also select a lower start address. Shifting of the start address to the area of the address range for process images must be supported by the CPU.
- If the size of the process image can be specified for a CPU (via parameter assignment), then you can enlarge the process image (cycle properties of the CPU), so that the entered address is again in the process image. The parameter is called "Size of the process image...".

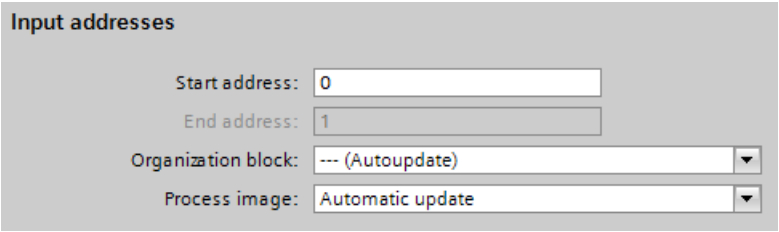
Assigning process image / process image partition (S7-1500)

Introduction

A process image offers the advantage that you can access a consistent image of process signals during cyclic program execution. For S7-1500, you can choose between "Update automatically" and "PIP 1" to max. "PIP 31".

Assigning addresses to the automatically updating process image area

You define the process image during address assignment (inspector window, "I/O addresses" area). The following setting should be selected if the addresses of a module are to lie in the process image area that is updated automatically by the system in each program cycle:



For comparison: For S7-300/400 modules, this setting is called "OB1 process image".

If the addresses should not lie in the process image area, select "None" as the process image.

Assigning an address to a process image partition

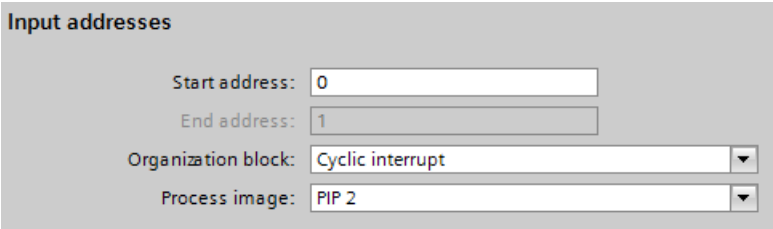
You define the process image partition during address assignment (inspector window, "I/O addresses" area). The process image partition is updated either by the user with the help of instructions or automatically by the system by connection to an OB.

Connecting a process image partition to an OB

You also connect a process image partition to an OB in the I/O addresses area ("Organization block" parameter).

For OBs of the type "Cyclic Interrupt", for example, the next free process image partition is assigned automatically; this can be changed, however. After the OB has been started, the assigned process image partition of inputs is updated by the system. At the end of the OB, the outputs of the assigned process image partition are written to the peripheral outputs by the system. The process image partitions are excluded from the automatic update.

Example for the coupling of the process image partition 2 (PIP 2) to a cyclic interrupt:



Exception: process image partitions of isochronous mode interrupt OBs are not updated by the system, even though they are connected to an OB (OB 61 to OB 64).

The assigned process image or process image partition is displayed in the properties of the assigned OB (cannot be changed).

Updating a process image partition in the user program

You can update each process image partition in the user program with special instructions. Use "UPDAT_PI" for the process image partition input and "UPDAT_PO" for the process image partition output.

Requirement

- You are in the device view.
- The module or submodule has an I/O address.

Procedure

To change the preset process image range, proceed as follows:

1. In the device view, select the module whose start address you want to set.
2. In the inspector window, click on the "I/O Addresses" area under "Properties".
3. Under "Process image," select the desired process image.

Using process image partitions in instructions

Introduction

Instructions such as "UPDAT_PI" for updating process image partition inputs and "UPDAT_PO" for outputting process image partition outputs to I/O outputs use the "PART" input parameter for inputting the process image partition.

In the following section we show you how to define a user constant for a process image partition and how to use it for inputting the process image partition in the user program. If you use different process image partitions for your I/O configuration, the symbols makes programming easier.

Requirement

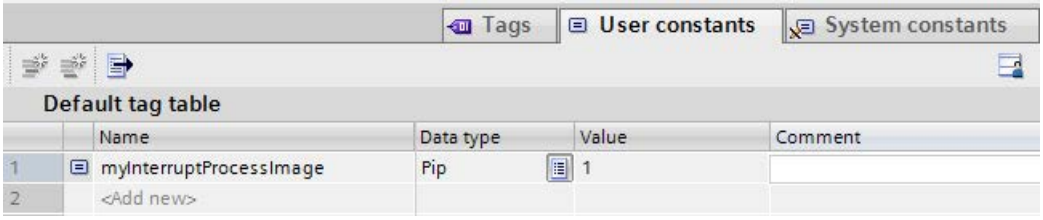
You have configured the I/O, assigned addresses and process image partitions, and created the required blocks.

The programming language is set to LAD for this example.

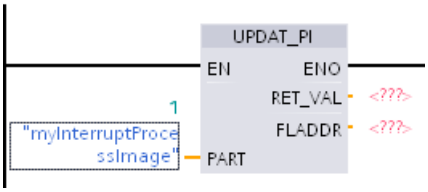
Procedure

Follow these steps:

1. Open the tag table and enter constants of data type "Pip" (process image partition) in the "User constants" tab. For each process image partition used, assign a name and, if necessary, a comment.



2. Open the program editor (e.g., by double-clicking an OB in the project tree).
3. Use a drag-and-drop operation to move the UPDAT_PI instruction to the network.
4. Double-click the PART input parameter to open the tag selection.
   Only process image partition constants will be listed in the tag selection.
5. Select the desired process image partition.



Assigning addresses to a location in the program

You can assign addresses of the I/O channels of modules directly to the points of use in the program or to a tag table.

Requirement

- The device view of the hardware and network editor is open.
- The zoom level in the device view must be set to at least 200% to allow you to see the individual I/O channels.
- The instruction window of the programming editor or a tag table is open.

Procedure

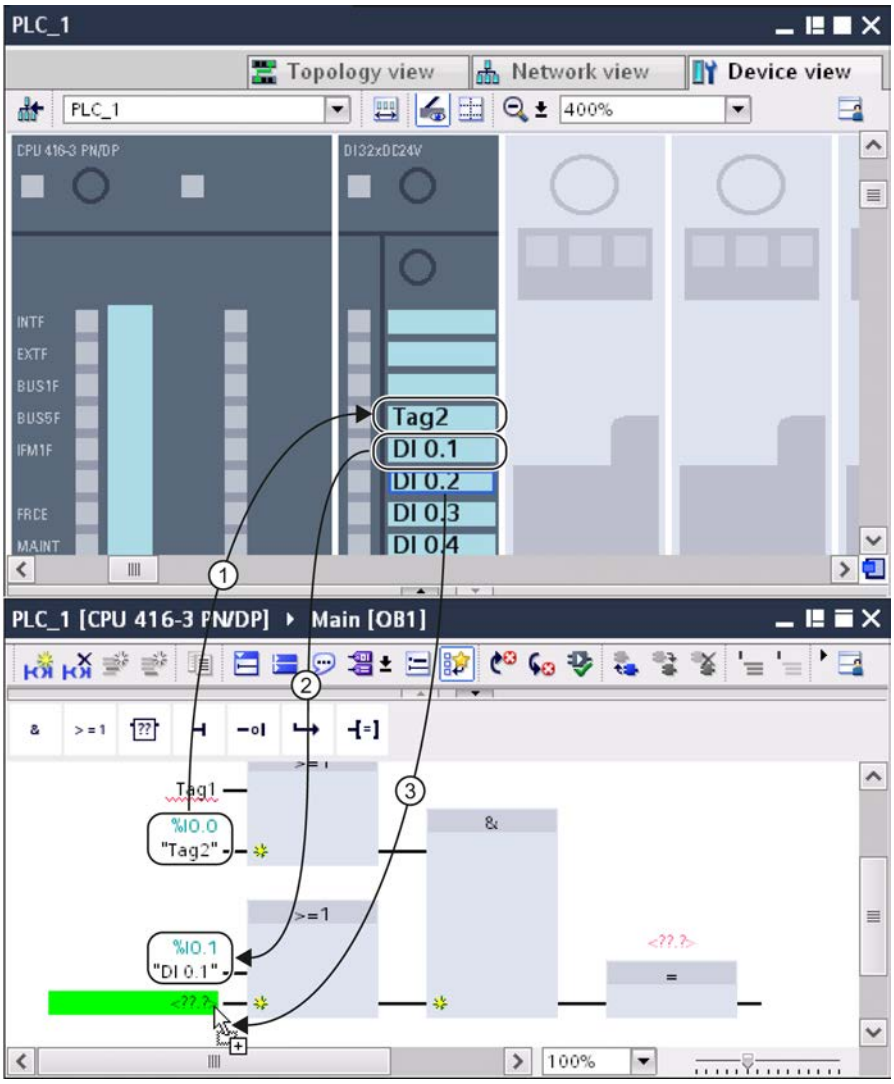
To assign I/O channels of modules to the points of use in the program or to a tag table, follow these steps:

1. In the device view, navigate to the module with the desired I/O channel.
2. Click and hold down the mouse button to drag the desired I/O address to the corresponding point of use of the block or to the tag table.

The address of the module is assigned to the point of use in the program or entered as a tag in the tag table.

Note

The tag for an input or output of a block can also be dragged to the input or output of a module in order to link the tag to the I/O channel of the module.



① The location of a block was dragged to an I/O channel.

② The name of an I/O channel was dragged to a point of use in the program.

③ Hold down the mouse button and drag from the name of an I/O channel to the input of a block.

See also

Overview of PLC tag tables (Page 2473)

Configuration control

Important information regarding configuration control

Operating principle

You can use the configuration control to set the configuration of a controller (here: central configuration of an S7-1500) and to operate variants (options) that deviate from this configuration.

A control data record you have programmed in the startup program notifies the CPU as to which modules are missing in the real installation as compared to the configuration or which modules are located on a different slot as compared to the configuration. The configuration control does not have an impact on the parameter assignment of the modules.

Configuration control gives you the flexibility to vary the central installation as long as the real configuration can be derived from the set maximum configuration.

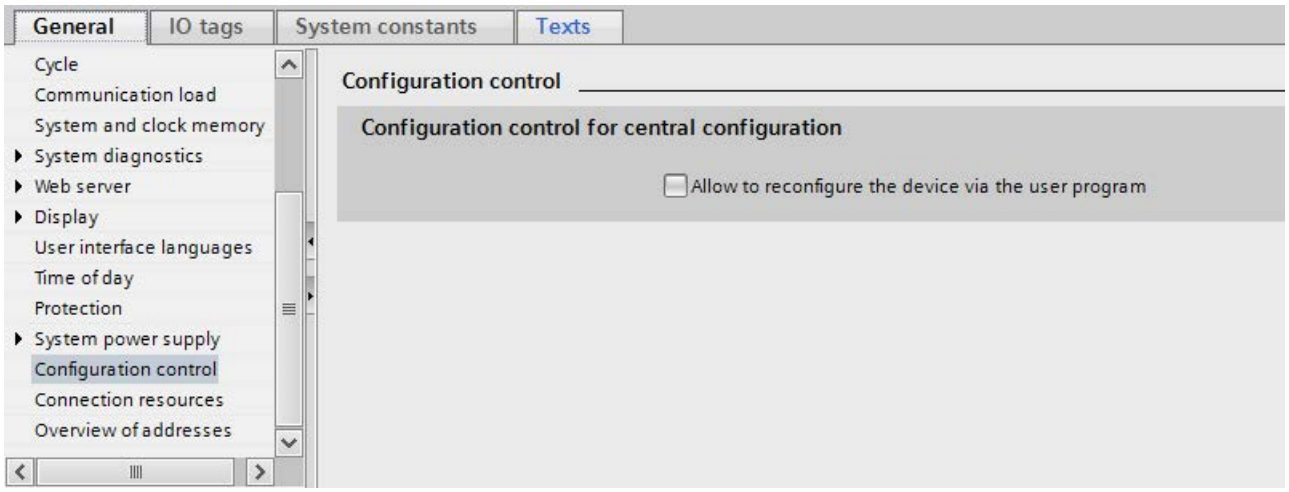
Below you will find a description of how to activate configuration control (CPU parameter assignment) and how to structure the required data record.

Requirement

- STEP 7 Professional as of version V13
- CPU S7-15XX as of firmware version V1.5
- Recommendation: Before you load a new program with a modified control data record, perform a memory reset. This action will prevent errors that occur if any incompatible control data record is present.

Required steps

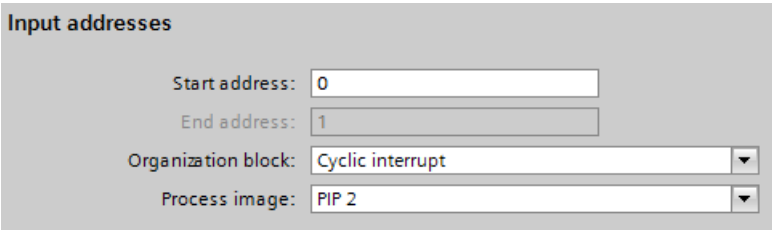
- Enable the "Allow to reconfigure the device via the user program" parameter when configuring the CPU ("Configuration control" area).



- Create a control data record (e.g., in a data block) according to the current configuration based on the sample described below for the control data record. The control data record has the number 196.

		Name	Data t...	Start v..	Retain	Accessi...	Visibl...	Setpoint	Comment
1		Static							
2		Block_length	Byte	9	☑	☑	☑	☐	4+number of slots
3		Block_ID	Byte	196	☑	☑	☑	☐	Data record number
4		Version	Byte	4	☑	☑	☑	☐	
5		Subversion	Byte	0	☑	☑	☑	☐	
6		slot_0	Byte	16#0	☑	☑	☑	☐	configured slot 0 < real slot 0
7		slot_1	Byte	16#1	☑	☑	☑	☐	configured slot 1 < real slot 1
8		slot_2	Byte	16#2	☑	☑	☑	☐	configured slot 2 < real slot 2
9		slot_3	Byte	16#FF	☑	☑	☑	☐	configured slot 3 < not used
10		slot_4	Byte	16#3	☑	☑	☑	☐	configured slot 4 < real slot 3

- In the startup OB, call the extended WRREC (Write data record) instruction and transfer the created control data record.



The configuration control is not ready for operation if a valid control data record is not transferred in the startup OB. The CPU returns from startup to STOP in this case.

Arrangement of the modules

The following table shows the slot number assignment:

Slot	Modules	Comment
0	Power supply module (optional)	Before the CPU
1	CPU	Slot 1 is always the CPU
2 - 31	I/O modules/system power supply modules, depending on the configuration variant	After the CPU

Control data record

A control data record 196 containing a slot assignment is defined for the configuration control.

Byte	Element	Code	Explanation
0	Block length	4 + number of slots	Header
1	Block ID	196	
2	Version	4 (for central I/O)	
3	Version	0	
4	Assignment of configured slot 0	Real slot 0	Control element Describes in each element which real slot in the device is assigned to the configured slot.
5	Assignment of configured slot 1	Real slot 1 (always 1, because the CPU is always in slot 1)	
:	:	:	
4 + (max. slot number)	Assignment of configured maximum slot	Real slot	

Control element

Each element of the control data record must contain the following information about the slot:

- Assignment of configured slot ⇔ real slot

Byte	Bit	Meaning
Assignment of configured slot (e.g. in byte 5)	0 to 7	11111111: Module not present 00000000 up to maximum slot: Real slot

Rules

Observe the following rules:

- The central configuration must not contain a CP/CM, e.g., CM 1542-5 (DP master or DP slave). You cannot load the configuration if you have configured a module of this type and configuration control is enabled.
- If you have enabled configuration control, the CPU is not ready for operation without a control data record. The CPU returns from startup to STOP if a valid control data record is not transferred in the startup OB. The central I/O is not initialized in this case. The cause for the STOP operating mode is entered in the diagnostic buffer.
- For addressing the WRREC instruction, use the HW identifier 33 (decimal, for the LADDR block parameter) to write the control data record.
  The HW identifiers can be found, for example, under **PLC tags > Default tag table > System constants** or in the "System constants" tab in the Inspector window of the CPU.
- The control data record is saved retentively in the CPU, which means that it is not necessary to write the control data record 196 again at a restart if the configuration is unchanged. Prior to commissioning, we recommend that a memory reset be performed for the CPU in order to delete any control data record that is present.
- Slot entries in the control data record outside the configured preset configuration are ignored.
- The control data record can be shortened. It must contain the entries up to the last slot of the current preset configuration.
- Each real slot may only be present once in the control data record.
- A real slot may only be assigned to one configured slot.
- System power supply modules (PS) can also be subject to configuration control. Note the information on this topic in the S7-1500 system manual (http://support.automation.siemens.com/WW/view/en/59191792). In particular for a system power supply module (PS) on slot 0, we recommended that you avoid reconfiguration.

Note

Modified configuration

The writing of a control data record with a modified configuration triggers the following automatic reaction by the CPU: Memory reset with subsequent startup with this modified configuration.

As a result of this reaction, the retentively saved original data record 196 is deleted and the new data record 196 is saved retentively.

Behavior during operation

Effect of the discrepancy between the configured configuration and real configuration:

- For the online display and for the display in the diagnostic buffer (module OK or module faulty), the hardware configuration is always used and not the differing real configuration. Example: A module outputs a diagnostic. This module is configured in slot 4, but is really inserted in slot 3 (missing module; see example in the next section). In the online view, a configured slot 4 is indicated as faulty. In the real configuration, the module at slot 3 indicates an error via an LED display.

If modules are entered as missing in the control data record, the automation system behaves as follows:

- Modules designated as not present in the control data record do not supply diagnostics and their status is always OK. The value status is OK.
- Direct write access to the outputs or write access to the process image of outputs that are not present: Remains without effect; no access error is signaled.
- Direct read access to the inputs or read access to the process image of inputs that are not present: Value "0" is supplied; no access error is signaled.
- Write data record to module that is not present: Remains without effect; no error is signaled.
- Read data record from module that is not present: An error is signaled because a valid data record cannot be returned.

Error messages

The following error messages are returned if an error occurs during writing of the control data record:

Table 8-58 Error messages

Error code	Meaning
$80B1_H$	Invalid length; the length information in data record 196 is not correct.
$80B5_H$	Configuration control parameters not assigned.
$80E2_H$	Data record was transferred in the wrong OB context. The data record must be transferred in the startup program.
$80B8_H$	Parameter error; module signals invalid parameters.

See also

S7-1500 system manual (http://support.automation.siemens.com/WW/view/en/59191792)

Example of a configuration control

A configuration consisting of a system power supply, CPU, and 3 I/O modules is configured in the following section.

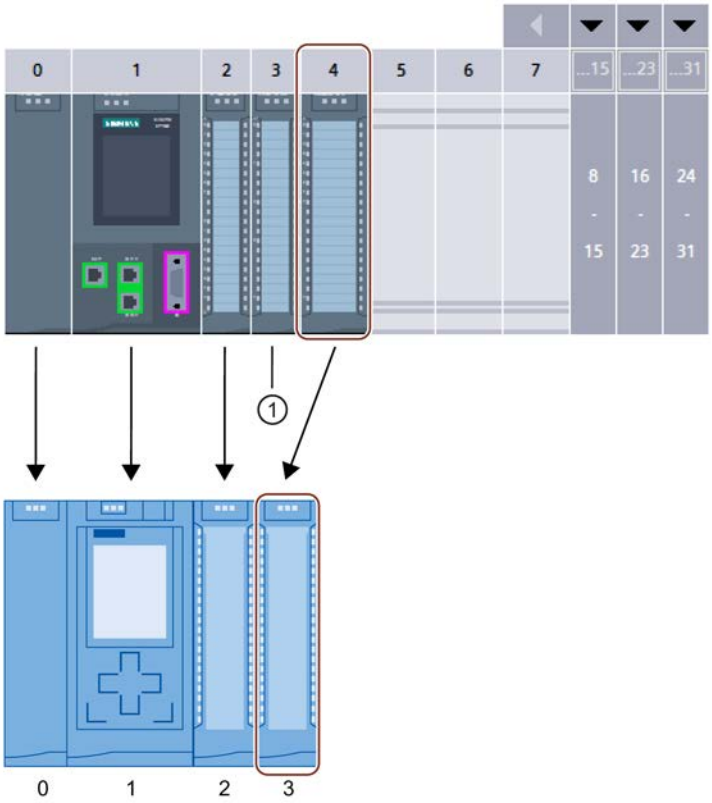
The module at slot 3 is not present in the first configuration expansion and is "hidden" by the configuration control.

In the second configuration expansion, the module that was initially hidden is located in the last slot. The added slot is made known to the CPU by a modified control data record.

Actual configuration with missing module

The specified configuration contains all modules that can be present in a final expansion stage.

The module that is inserted in slot 3 in the specified configuration is missing in the real expanded configuration. For this reason, slot 3 must be coded in the control data record accordingly with "FF $_H$" (= not present).



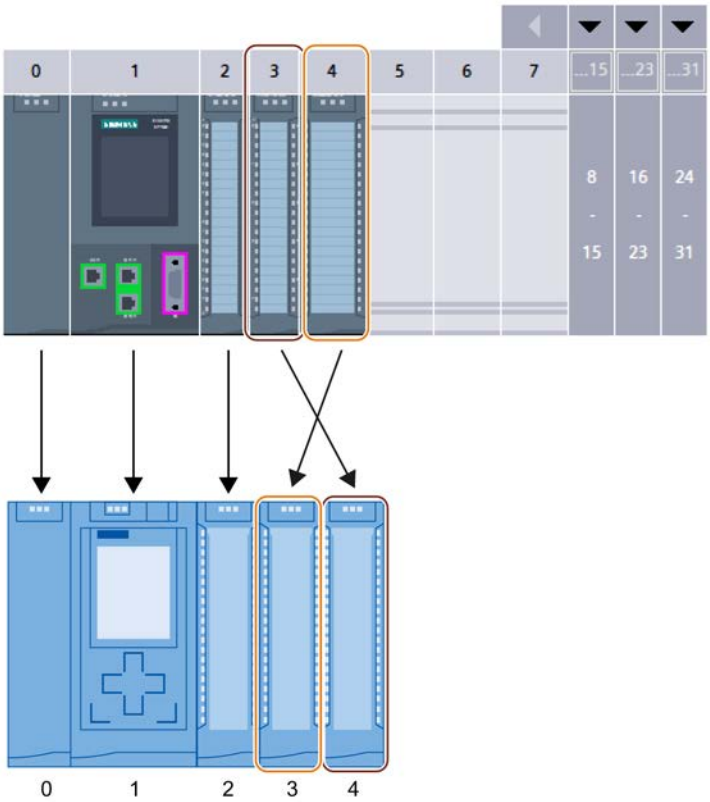
		Name	Data t...	Start v...	Retain	Accessi...	Visibl...	Setpoint	Comment
1		Static							
2		Block_length	Byte	9	☑	☑	☑	☐	4+number of slots
3		Block_ID	Byte	196	☑	☑	☑	☐	Data record number
4		Version	Byte	4	☑	☑	☑	☐	
5		Subversion	Byte	0	☑	☑	☑	☐	
6		slot_0	Byte	16#0	☑	☑	☑	☐	configured slot 0 < real slot 0
7		slot_1	Byte	16#1	☑	☑	☑	☐	configured slot 1 < real slot 1
8		slot_2	Byte	16#2	☑	☑	☑	☐	configured slot 2 < real slot 2
9		slot_3	Byte	16#FF	☑	☑	☑	☐	configured slot 3 < not used
10		slot_4	Byte	16#3	☑	☑	☑	☐	configured slot 4 < real slot 3

① Module is missing in the real configuration

Actual configuration with subsequently added module

The module present in slot 3 in the specified configuration is added to the back of the real configuration by inserting it as the last module in slot 4.

The control data record is adapted accordingly.



		Name	Data type	Start value	Retain	Acc...	Visi...	Setpo...	Comment
1		Static							
2		Block_length	Byte	9	☑	☑	☑	☐	4+number of slots
3		Block_ID	Byte	196	☑	☑	☑	☐	Data record number
4		Version	Byte	4	☑	☑	☑	☐	
5		Subversion	Byte	0	☑	☑	☑	☐	
6		slot_0	Byte	16#0	☑	☑	☑	☐	configured slot 0 < real slot 0
7		slot_1	Byte	16#1	☑	☑	☑	☐	configured slot 1 < real slot 1
8		slot_2	Byte	16#2	☑	☑	☑	☐	configured slot 2 < real slot 2
9		slot_3	Byte	16#4	☑	☑	☑	☐	configured slot 3 < real slot 4
10		slot_4	Byte	16#3	☑	☑	☑	☐	configured slot 4 < real slot 3

Configurations for Web server

Information about the web server

Introduction

The web server lets you monitor and administer the CPU through authorized users by means of a network. This permits evaluation and diagnostics over long distances. All you need is a web browser.

Alarms and status information are visualized on HTML pages.

Web browser

You need a web browser to access the HTML pages of the CPU.

The following web browsers have been tested for communication with the CPU:

- Internet Explorer (Version 8)
- Mozilla Firefox (Version 21)
- mobileSafari (iOS5)

Reading information via the web server

The following information can be read from the CPU. The availability of the respective web pages depends on the CPU and its firmware version.

Detailed information on the individual web pages and on the exact scope of the information can also be found in the following manuals:

Documentation for S7-300 (http://support.automation.siemens.com/WW/view/en/12996906)

Documentation for S7-400 (http://support.automation.siemens.com/WW/view/en/44444467)

Documentation for S7-1500 (http://support.automation.siemens.com/WW/view/en/59193560)

Information	Description
Start page with general CPU information	The start page provides an overview of general information on the CPU, the name of the CPU, the type of CPU and basic information on the current operating state.
Identification information	Displays the static identification information such as serial number, order number and version numbers.
Content of the diagnostic buffer	Displays the content of the diagnostics buffer with the most recent entries first.
Status of the modules	Displays whether the components of a station are in proper order, if, for example, there are maintenance requirements or components cannot be reached.

Information	Description
Alarms (without acknowledgment option)	Displays project-oriented alarm text with filters and sorting options.
Information about communication	Display of the communication connections, display of the resources as well as interface parameters and port statistics.
Topology	Display of the topology with the graphic, table views and status overview for networked PROFINET participants.
Status of the tags	Display of the status of user program operands for observing the values.
Tag tables	Display of the tag tables created with STEP 7 with the "web server" option.
User pages (if user-defined web pages have been configured and loaded)	The user web pages deliver a list of web pages with customer-specific web applications.
File browser	S7-1500: The file browser lists files and directories that are located on the SIMATIC Memory Card. The files can be downloaded, deleted, renamed or uploaded regardless of the file type. Directories can only be created, deleted or renamed. Exception: System files are not displayed in the file browser and can therefore not be changed or deleted. The system files are the order file and all the special directories including their content which is referenced from the order file.

Web access to the CPU via PG/PC

Proceed as follows to access the web server:

1. Connect the client (PG/PC) to the CPU via the PROFINET interface.
   If you do not access the web via the PROFINET interface of the CPU but via the interface of a CP (e.g., CP 1543-1), the web pages load much more slowly than with the connection via the interface of the CPU. Make sure that the cache settings (temporary Internet files) of the browser are set to "Automatic" in this case.
2. Open the web browser.
   Enter the IP address of the CPU in the "Address" field of the web browser in the format http://ww.xx.yy.zz (example: http://192.168.3.141).
   The start page of the CPU opens. You can navigate from the start page to additional information.
   If the start page does not appear, make sure that the web server was activated in the configuration and that the configuration was loaded.

Web access to an S7-300/400 CPU via HMI devices and mobile devices

The web server also supports the Windows Terminal Service so that in addition to the operation of programming devices and PCs, thin client solutions with mobile devices (e.g. PDA, MOBIC T8) and rugged local stations (e.g. SIMATIC MP370 with ThinClient/MP option) with Windows CE are possible.

Proceed as follows to access the web server:

1. Connect the client (HMI device, PDA) with the CPU via the PROFINET interface.
2. Open the web browser.
   Enter the IP address of the CPU in the "Address" field of the web browser in the format http://ww.xx.yy.zz /basic (example: http://192.168.3.141/basic).
   The start page of the CPU opens. From the start page, you can navigate to further information.

HMI devices operating with the Windows CE operating system V 5.x or earlier process CPU information in a browser specially developed for Windows CE. The information appears in a simplified format in this browser.

Note

Using SIMATIC Micro Memory Card with web server (S7-300)

The configuration data for the web server is stored on the SIMATIC Micro Memory Card. That is why we recommend using a SIMATIC Micro Memory Card with at least 512 KB.

You can also use the web server without SIMATIC Micro Memory Card. The CPU must have been assigned an IP address to permit operation.

- The content of the diagnostic buffer is displayed in hexadecimal code.
- Start page, identification and communication information and tag status are displayed as plain text.
- Following displays remain empty:
  - Topology
  - Messages
  - Status of the modules
- The automatic page refresh is also activated without SIMATIC Micro Memory Card and without configuration.

Web access to an S7-1500 CPU via HMI devices and mobile devices

HMI devices operating with the preinstalled Windows CE operating system (V 5.x or earlier) process CPU information in a browser specially developed for Windows CE. The information appears in a simplified format in this browser.

The web server offers web pages with reduced complexity which have been optimized for devices with small screens and low computing power.

Requirement for access with mobile devices:

- WLAN connection for the CPU (WLAN Access Point, for example, SCALANCE W788-1RR or SCALANCE W784-1)
- Activated web server on the CPU
- WLAN has to be activated for the mobile device and a connection to the access point has to be established.

The procedure for a mobile device is identical to that for other devices (programming device/ PC). In the browser, you enter the IP address of the CPU interface, download the certificate and install it subsequently.

Security

Make sure that you protect the CPU from being compromised through the use of different methods, for example, by limiting network access and using firewalls.

The web server offers the following security functions:

- Access via the secure transmission protocol https
- Configurable user authorization via user list

See also

Working with multi-language projects (Page 340)

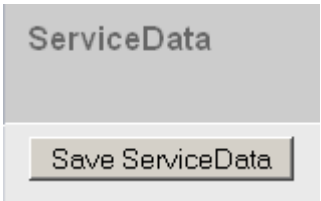
Requirements for web access (Page 906)

Reading out service data

The web server of an S7-1500 CPU gives you the option of saving service data. In addition to the contents of the diagnostic buffer, service data includes additional information on the internal status of the CPU. If you should encounter a problem with the CPU that you cannot resolve, you have the option of submitting the service data to the Service&Support team.

Procedure

1. Enter the following address in the address bar of your web browser:
   "http://<CPU IP address>/save_service_data.html", e.g.,
   "http://192.168.3.141/save_service_data.html"
2. Your screen displays the service data page with a button for saving the service data.



3. Save the service data locally on your display device by clicking "Save ServiceData".

Result

The data is saved into a .dmp file with the following naming convention: "<MLFB><serial number><time stamp>.dmp". The file name can be changed subsequently.

Standard websites

Requirements for web access

In the following, the requirements for accessing standard CPU websites are explained, and the effects of missing or existing configuration information are illustrated.

Requirement

The web server must be started.

The web server does not start until you have enabled it in the "Web server" area of the CPU properties.

Note the following:

- The websites are transmitted by default through a non-secure connection and are not protected from third party access. If you want to transmit the websites in an encrypted fashion, then enable the "Only permit access through HTTPS" option in the CPU characteristics. Make sure that the URL of the CPU begins with https:// in this case.
- Access through HTTPS only works if the time was set in the CPU.
- For further access protection, create a user in the "Web server" area of the CPU properties.
- The full functionality of the web pages in relation to module state, topology and alarms requires system diagnostic data to be generated and loaded (Report System Error) for S7-300/400 CPUs. After the web server has been activated, a query appears asking if the system diagnostics is to be enabled. You should acknowledge this query with "Yes".

Logon

To access the standard websites, no logon is required. A user must be logged on to execute certain actions like changing the operating mode of the CPU or for write access. The input fields for logging on are located in the upper left corner on each standard website.

If you logon as a user, you have to enter the user name and password here. The user name and password are set in the "Web server" area of the CPU properties.

Web access without project information

Without configuration, only basic information is available on the standard websites. This information is available in different languages:

- Diagnostic buffer entries are not displayed as clear text without configuration information, but rather only as a hexadecimal code.
- If the web server supports the CPU tag tables (watch tables):
  The tag tables page does not display any tags without configuration. You can always open up a tag status.

Web access with configuration information

Texts are contained in the selected languages in the configuration information:

- Text for diagnostic buffer entries as short/long text
- For S7-300/400 CPUs: Alarm texts; for example, alarm component AlarmS/SQ, AlarmD/DQ
- If tag tables (watch tables) are supported: Tag tables with the symbols and comments contained in them

The texts for diagnostic buffer entries and alarms are shown during operation in the language that you set in your web browser.

See also

Settings for operation (Page 908)

Settings for operation

Settings for operation

To use a CPU web server and set the behavior of the web server, you have to highlight the CPU in the network view, device view or topology view.

Execute the following settings in the Inspector window under "Properties > General > Web server":

- Enable the web server
- Limiting web access to secure Hypertext transmission protocol (HTTPS)
- Activate automatic update
- Set the regional web language
- Expanding user lists
- For S7-300/400 CPUs: Selecting the display classes of the alarms

Every PROFINET interface of an S7-1500 CPU or of a PROFINET CP or a PROFINET-CM can also be enabled or disabled for "external" access to the web server. It is simplest to manage this access centrally in the properties of the CPU. To do so, click the "Web server > Overview of interfaces" area in the "Properties" tab of the inspector window and activate the interfaces through which access is allowed.

See also

Project text basics (Page 340)

Requirements for web access (Page 906)

Enable the web server

Enable the web server

The web server is disabled by default. The corresponding option box must be enabled to have the CPU show websites.

This setting can be found in the Inspector window under "Properties > General > Web server"; optional box "Enable web server for this module".

If the web server was activated, a prompt is displayed for S7-300/400 CPUs asking whether the system diagnostics (Report System Errors) should be enabled. Click on "Yes" if the required modules and texts should be generated with "Report System Errors" so that the alarm texts are available in the web browser during operation. For S7-1500 CPUs, the system diagnostics is always activated (cannot be deactivated).

In the case of S7-1500 CPUs with several PROFINET interfaces or when CPs with PROFINET interfaces are configured, you additionally have to activate the interfaces through which web server access is to be allowed. This setting is located under "Properties > General > Web server > Overview of interfaces".

Activate automatic update

Activate automatic update

With the "Automatic updating" check box activated, the web sites are automatically updated in the time intervals set under the "Update interval". The "Identification" website is excluded from the updating.

Proceed as follows to activate the automatic updating:

1. Select the check box "Activate automatic update" under "Properties > General > Web server > Automatic update".
2. Enter the updating interval.

Note

Update time

The set activation interval is the shortest updating time. Larger amounts of data or several HTTP/HTTPS connections increase the update time.

Set the regional web language

Set the regional web language

To select the regional languages for language-dependent texts which will be downloaded to the CPU, proceed as follows:

1. Navigate in the Inspector window to the "Languages" area under "Properties > General > Web server > Languages".
2. Activate the required languages for display in the web browser in the first column; the number of selectable languages is specific to the CPU.
3. Assign a project language to the activated web server languages.
   The project languages that are assigned must be activated and the corresponding texts (translations) must be in the project. The project language selection can be found in the project navigation under "Languages & Resources".
   Through these assignments, for example, display texts not available in the CPU will be downloaded in the selected language (with language and country code) when loading the CPU.

	Enable	Web server language	Assign project language
	☐	German	None
	☐	English	None
	☐	French	None
	☐	Spanish	None
	☐	Italian	None

Note

If you activate the web server and do not assign a language, messages and diagnostic information are displayed in a hexadecimal code.

Set the regional web language

Set the user interface language for the web

To select the regional languages for language-dependent texts which will be downloaded to the CPU, proceed as follows:

1. Navigate to the "User interface languages" area ("Properties > General > User interface languages") with selected CPU in the Inspector window.
2. Assign a project language to each user interface language.
   The project languages that are assigned must be activated and the corresponding texts (translations) must be in the project. The project language selection can be found in the project tree under "Languages & Resources".
   Through these assignments, for example, display texts not available in the CPU will be downloaded in the selected language (with language and country code) when loading the CPU.

Note

The number of loadable languages for the web server is limited for specific CPUs. If, for example, two languages are possible, assign the corresponding project language to your native language in the "Web server language" column. In the remaining lines, you assign the same substitute language to the "Web server language".

User lists

User Management

The User Management offers the following possibilities:

- Create users
- Set execution rights
- Assign passwords

With this assignment, users are exclusively entitled to options that are assigned to the execution rights.

- If a user is configured, this user can access the web pages according to his access authorizations after logging on with his password.
- Depending on the CPU and firmware used, you can assign different user rights. Rights that your CPU does not support cannot be selected.
- A user called "Everybody" with minimal access rights is created by default in the user list: Read access to the intro and start page. The user "Everybody" is defined without an assigned password and cannot be changed.

Access for HTTPS

Access only through HTTPS

HTTPS serves for the encryption and authentication of the communication between the browser and web server.

To limit the data transmission between the browser and CPU exclusively to the HTTPS protocol, activate the "Access only through HTTPS" option under "Properties > General > Web server".

Requirements

The requirements for an error-free HTTPS access to the CPU are as follows:

- The current clock time must be set in the CPU
- A valid installed certificate.

If no certificate is installed, a warning will be shown recommending not to use the site. To see the site, you have to explicitly "add an exception".

You will obtain a valid certificate (Certification Authority) as a download from the "Intro" website under "Download certificate". With the help of your respective web browser, you can find out how to install the certificate.

Example of access to web pages with HTTPS protocol

To access web pages of the CPU with HTTPS protocol, enter the URL in the following format: https://ww.xx.yy.zz, where ww.xx.yy.zz is the IP address of the CPU.

Example entry: https://192.168.3.141

You can recognize an encrypted connection by a lock symbol in the status bar of the web page.

Display classes of the messages

Display classes of the messages

All display classes of the messages are activated in the basic configuration. The messages for the selected display classes are displayed during operation on the "Messages" website. Messages for display classes that are not selected are shown as hexadecimal code and not as plain text.

How to configure the display classes:

- for "Report System Errors" in the Inspector window under "Properties > General > System diagnostics"
- for module-related messages in the message editor

Note

Reducing memory requirements

You can reduce the memory requirements for the web server configuration by selecting only the display classes of the messages that should be used.

Create and download user-defined websites

What you need to know about user-defined web pages

Concept

The concept of user-defined web pages allows you to access freely-designed web pages of the CPU from a web browser. The web server of the CPU provides this function.

You are not dependent on special tools for the design and functionality of the user-defined web pages. You can adapt the pages in the layout with CSS, provide dynamic content with JavaScript or use any framework to produce web pages.

The totality of files processed by the web server is also referred to as the "web application".

Web application and user program

Using HTML code in user-defined web pages, you can also transmit data via a web browser to the user program of the CPU for further processing and can display data from the operand area of the CPU in the web browser.

You can use script instructions (such as Javascript) to optimize your web pages, for example to dynamically change contents or validate user entries.

To synchronize between the user program and the web server, but also to initialize, you must call the WWW (SFC 99) instruction in the user program.

- If no interaction is required between the web application and the user program, for example, if a web page only provides static information, only initialization in the user program is required.
- If a simple data exchange is necessary between PLC tags and tags in web applications, to display the contents of PLC tags or write a value in a PLC tag for example, the syntax for reading and writing tags has to be observed. In this case only an initialization is required in the user program, for example in the startup OB.
- If a further interaction is required between the web application and the user program, you must handle status and control information from the Web Control DB in addition to the synchronization between Web server and user program. This is the case, for example, when user entries are transmitted via the web browser to the web server for evaluation by the CPU. Unlike simple data exchange, the user program directly influences the time at which the requested web page is relayed back to the web browser. In this case, you must be acquainted with the concept of manual fragments and the structure of the Web Control DB.

Initialization

User-defined web pages are "packaged" in data blocks for processing by the CPU. You must generate appropriate data blocks from the source data (HTML files, images, JavaScript files, etc.) during configuration to be able to download the web application into the CPU. The Web Control DB takes on a special role (default: DB 333). It contains status and control information as well as links to additional data blocks with coded web pages. Data blocks that contain coded web pages are termed "Fragment DBs".

When the data block is downloaded into the CPU, the CPU does not "know" that user-defined web pages are coded inside it. The "WWW" (SFC 99) instruction, for example, in the Startup OB informs the CPU which DB is the Web Control DB. The user-defined web pages can be accessed via a web browser after this initialization.

Synchronization

If the user program is to exchange data with the user-defined web pages, the WWW (SFC 99) instruction must be used in the cyclic program section.

Examples of interaction between user program and web page:

- Check received data
- Compiling and returning data for web browser making request

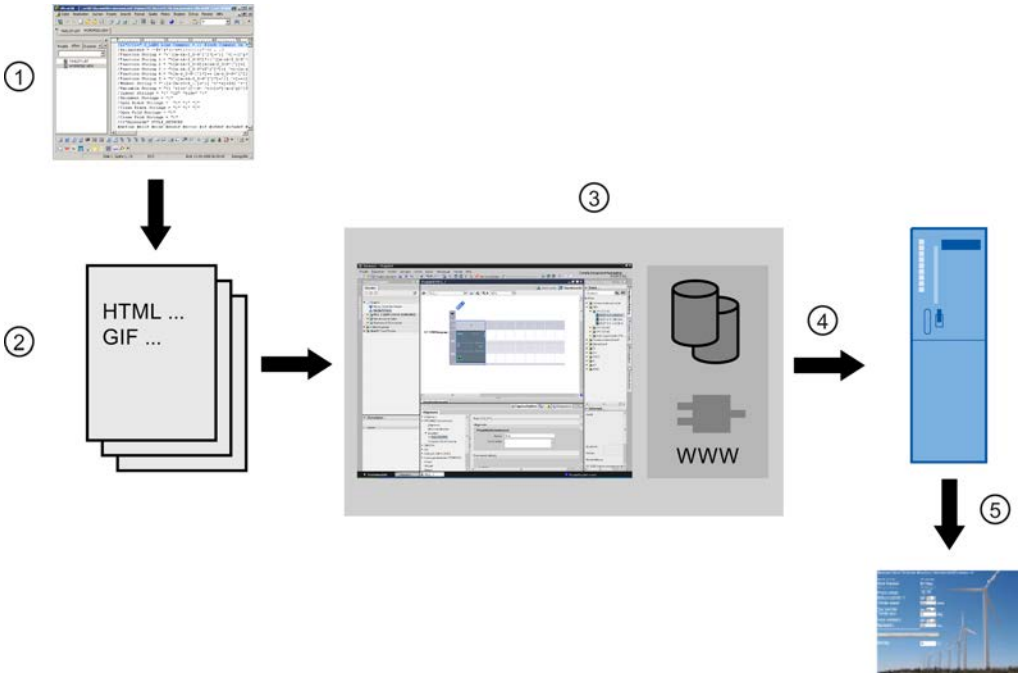
In this case, the status information must be able to be evaluated and control information must be transmitted to the web server, for example, to release a requested web page.

Procedure overview

Basic

In this chapter, the basic procedure is explained step-by-step to create, load and use user-defined websites in the operating phase.

The following image displays the procedure for the creation and display of user-defined websites:



① Programming a web application (maybe with the help of suitable tools, if applicable, with AWP commands for dynamic sites).

② Web application consists of individual source files, for example, *.html, *.gif, *.js, ...

③ With STEP 7:

- Generating the data blocks (web-control-DB and fragment-DBs) from source files. The DBs contain meta information and the complete web application including the images and the dynamic and static parts of the web application. The DBs are stored in the project navigation under "System blocks".
- Open the "WWW" (SFC 99) instruction in the user program. The instruction initializes the web server of the CPU for a web application.
- If needed, program out interaction between web server and user program

④ Load the modules in the CPU.

⑤ Open website in the browser. The opening of websites from the CPU occurs through the Entry of the IP address of the CPU.

Creating web pages

Web design tools from various companies can be used to create user-defined web pages. As a rule, the web pages should be programmed and designed compliant to the conventions of the W3C (World Wide Web Consortium). No check is made for compliance to W3C criteria in the web server of the CPU.

Rules

- The tool must be able to directly edit the HTML code so that the AWP command can be inserted into the HTML page.
  Only the AWP commands are parsed in the CPU and, for example, replaced by values from the user program/process image of the CPU.
- Files containing AWP commands must be coded in UTF-8. In the metadata of the HTML page, therefore, set the attribute charset to UTF-8 and save the file UTF-8 coded.
- Files containing AWP commands must not contain the following sequence:]]
- Files containing AWP commands must not contain the following sequence outside of the "Tag read ranges" (:=<Tag name>:): :=
  Tip: Replace the first character of a prohibited sequence with its character coding; for the colon, for example, :.

A small example for a custom web page should make clear the basic design.

Requirement

- The CPU must have a web server and the web server of the CPU must be activated.
- To be able to access PLC tags with write access as a user, you must be logged on as "admin".
- For the example below, PLC tags must be defined for those PLC tags that are to be shown on the web page. This is shown here for the first tab used, "Tank_below_max".

		Name	Data type	Address
1		Tank_below_max	Bool	%I0.0

Creating user-defined web pages

The following code for an example web page reads values from the process image and provides them in a table.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/
TR/html4/strict.dtd">
<html>
  <head>
   <meta http-equiv="Content-Type" content="text/html;
charset=utf-8">
   <title>Mix</title>
  </head>
   <body>
    <h1>Mix</h1>
    <h2> Actual State </h2>
    <table border="1">
     <tr>
      <th>Variable</th>
      <th>State</th>
     </tr>
     <tr>
      <td>Tank below max</td>
      <td>:="Tank_below_max":</td>
```

```
      </tr>
      <tr>
       <td>Tank above min</td>
       <td>:="Tank_above_min":</td>
      </tr>
     </table>
    </body>
</html>
```

AWP commands

The interface between a freely-programmable web application for a CPU that has a web server and the CPU data is declared by the AWP command (Automation Web Programming).

To develop web applications you are only subject to the restrictions of the web browser. In one of the programming languages of STEP 7, control with the user program which CPU data is displayed at what time in the web browser of the viewer. Use AWP commands, which you comment within the HTML files, to declare data to be used for intentional interaction between the web application and the user program.

AWP commands are inserted as HTML comments with a special syntax into HTML files; they declare the following features:

- Read PLC tags
- Write PLC tag
- Read special tags
- Write special tags
- Define enum types
- Assign tags to enum types
- Defining fragments
- Import fragments

Syntax of AWP commands

An AWP command begins with "`<!--AWP_`" and ends with "`-->`". In JavaScript files, the commands should also be enclosed by JavaScript comments ("`/*...*/`").

Notation rules for PLC tag names within an AWP command

The AWP commands "AWP_In_Variable" and "AWP_Out_Variable" contain a name attribute and optionally a use attribute. A PLC tag name is assigned to these attributes, by means of which the PLC tags in the browser are written or read. The following rules apply to handling PLC tag names in HTML code:

- PLC tags must be enclosed in quotation marks (" ... ").
- PLC tags used in AWP commands must also be enclosed by single quotation marks ('" ... ") or with quotation marks masked by a backslash ("\" ... \"").

- If the PLC tag name contains the character \ (backslash), this character must be designated with the escape sequence \\ as standard character of the PLC tag name.
- If the PLC tag name in the AWP command is also enclosed by single quotation marks and the single quotation mark (') occurs within the name, it must also be designed as normal character by the escape sequence \'.
- If an absolute address (input, output, bit memory) is used in AWP command, it is enclosed by single quotation marks.

PLC tag	PLC tag in HTML code
"Velocity"	<!-- AWP_In_Variable Name='"Velocity"' -->
	<!-- AWP_In_Variable Name="\"Velocity\"" -->
"abc\de"	<!-- AWP_In_Variable Name='"abc\\de"' -->
"abc'de"	<!-- AWP_In_Variable Name='"abc\'de"' -->
"abc'de"	<!-- AWP_In_Variable Name="abcde" Use'"abc\'de"' -->
"DB name".tag	<!-- AWP_In_Variable Name='"DB name".tag' -->
"DB name"."ta.g"	<!-- AWP_In_Variable Name='"DB name"."ta.g"' -->
-	<!-- AWP_Out_Variable Name=’flag1’ Use='M0.0' -->

Reading tags

User-defined web pages can read PLC tags.

The PLC tag must be specified by a PLC tag name.

These OUT variables (direction of output as viewed from the controller) are inserted at any location within the HTML text with the syntax described in the following.

Syntax

```
:=<varname>:
```

These references are replaced when the web server is in operation by the current values of the PLC tag in each case.

`<varname>` can be a simple, global PLC tag but also a complete tag path to a structural element.

Notation rules for PLC tag names

- PLC tags in HTML code are enclosed by quotation marks ("), if they are defined in the tag table. In the case of data block tags, the name of the data block is enclosed by quotation marks. If special characters are used in the structure elements of the data block, for example the dot (.) or blank, this part must also be enclosed by quotation marks.
- Quotation marks are not used for absolute addresses of inputs, outputs or bit memories.

PLC tag	PLC tag in HTML code
"DB_name".var_name	:="DB_name".var_name:
"DB_name"."var.name"	:="DB_name"."var.name":
"memory"	:="memory":

PLC tag	PLC tag in HTML code
-	:=I0.0:
	:=Q0.0:
	:=MW100:
	:=%MW100:
"My_Data_Block".flag1	<!-- AWP_Out_Variable Name='flag1' Use='"My_Data_Block".flag1' -->
	...
	:=flag1:

- If the PLC tag name contains the character : (colon) or \ (backslash), this character must be designated with the escape sequence \: or \\ as standard character of the PLC tag name.

PLC tag	PLC tag in HTML code
"abc:de"	:="abc\:de":
"abc\de"	:="abc\\de":

- Special characters "<, &, >"
  Display problems can occur if these characters are contained in the tag name (for example, "a<b").
  Avoid expressions such as :="a<b": in the HTML page.
  To prevent display problems from occurring, use e.g. an AWP command with a use expression according to the pattern depicted below. The use attribute defines the PLC tag with the problematic character, the name attribute defines the name without problematic character, as it is used in the HTML page.

PLC tag	PLC tag in HTML code
"a<b"	<!—AWP_Out_Variable Name='simplename' Use='"a<b"' -->
	...
	:=simplename:

Writing tags

Custom web pages can write data into the CPU.

This requires an AWP command that identifies the PLC tag to be written.

The PLC tag must also be specified by a PLC tag name.

The IN tags (direction of input as viewed from the controller) are placed on the browser page. This can be done, for example, in a form.

The tags are either set in the HTTP header (by cookie or POST method) or in the URL (GET method) by the browser and are then written by the web server into the respective PLC tag.

Syntax

To allow the IN tags to be written to the CPU, the tags must first be defined by an explicit AWP instruction:

```
<!-- AWP_In_Variable Name='"<Varname1>"' Name='"<Varname2>"'
Name='"<Varname3>"' -->
```

Several tags can be defined in an instruction - such as that shown above.

In cases where the name of the tag that you use for the web application is not identical to the name of the PLC tag, the "Use" parameter can be used to assign to a PLC tag.

```
<!-- AWP_In_Variable Name='<Varname_Webapp>' Use='<PLC_Varname>'
```

Example

The "AWP_In_Variable" AWP command is indispensable when handling forms.

```
<form method="post" action="/awp/appl/x.html">
 <p>
  <input name='"var1"' type="text">
  <input value="set" name='"Button1"' type="submit">
 </p>
</form>
```

In the form defined above, tags "var1" and "Button1" are transferred to the web server by "post". The user places the "var1" tag in the form field. The "Button1" tag has the fixed value "set". To allow the "var1" and "Button1" tags to be written to the CPU, the following instruction must be included in the same fragment:

```
<!-- AWP_In_Variable Name='"var1"' Name='"Button1"' -->
```

Because global tags are enclosed in quotation marks, the name must be enclosed in the AWP command by apostrophes or by masked quotation marks (\").

```
<!-- AWP_In_Variable Name='"Info".par1' -->
```

Conditions for write access during operation

The following requirement has to be met in order for a user to be able to write to PLC tags from a user-defined web page.

- A user was set up and is logged on.

This rules applies to all writing access to web pages on a CPU.

See also

User lists (Page 911)

Special tags

Special tags are mainly HTTP tags set in the definition of the World Wide Web Consortium (W3C) . Special tags are also used for cookies and server tags.

The AWP command to read and write special tags differ only in that they have additional parameters than the AWP command used to read and write normal tags.

Reading a special tag

The Web server can read PLC tags and transfer these to special tags in the HTTP Response Header. You can, for example, read a URL for a diversion to another web page and transfer to the special tag HEADER:Location using the special tag HEADER:Location.

The following special tags can be read:

Name	Description
COOKIE_VALUE:name	Value of cookie with name: "name"
COOKIE_EXPIRES:name	Execution time of cookie with name: "name" in seconds (must be set beforehand).
HEADER:Status	HTTP status code (if no other value has been set, status code 302 is returned).
HEADER:Location	Path for forwarding to another page. Status code 302 must be set.
HEADER:Retry-After	Anticipated time in which the service is not available. Status code 503 must be set.
HEADER: …	All other header tags can also be forwarded in this way.

Use the AWP command "AWP_Out_Variable" to specify which PLC tags are to be transferred in the HTTP header to the web browser.

Basic structure:

```
<!-- AWP_Out_Variable Name="<Typ>:<Name>" [Use="<Varname>"] -->
```

Parameter description

- Name: Type and name of special tag
- Use (optional parameter): In cases where the name of the special tag is not identical to the name of the PLC tag, parameter "Use" can be used to assign to a PLC tag.

Example:

```
<!-- AWP_Out_Variable Name="COOKIE_VALUE:siemens" Use='"info".language' -->
```

Writing a special tag

In principle, all HTTP tags written in the HTTP header by the web browser can be evaluated by the user program of the CPU. Examples of tag types:

Name	Description
HEADER:Accept-Language	Accepted or preferred language
HEADER:Authorization	Proof of authorization for a requested resource
HEADER:Host	Host and port of the requested resource

Name	Description
HEADER:User-Agent	Information on the browser
HEADER: …	All other header tags can also be forwarded in this way
SERVER:current_user_id	Indicates whether a user is logged on (current_user_id=0: no user logged on)
SERVER:current_user_name	User name of the user logged on
SERVER:GET	Request method is GET
SERVER:POST	Request method is POST
COOKIE_VALUE:name	Value of cookie with name: "name"

The AWP command "AWP_In_Variable" is used to define which special tags are to be evaluated in the user program of the CPU.

Basic structure:

```
<!-- AWP_In_Variable Name="<Typ>:<Name>" [Use="<Varname>"] -->
```

Parameter description:

Name: Type and name of special tag

Use (optional parameter): In cases where the name of the special tag is not identical to the name of the PLC tag, the parameter Use can be used to assign to a PLC tag.

Examples:

```
<!-- AWP_In_Variable Name="COOKIE_VALUE:siemens" Use='"info".language' -->
```

The tag name in the HTTP header is replaced by the PLC tag name specified by Use . The cookie is written to the PLC tag "info".language .

```
<!-- AWP_In_Variable Name='COOKIE_VALUE:siemens' Use='"info".language' -->
```

The tag name in the HTTP header is replaced by the PLC tag name specified by Use. The cookie is written to the PLC tag "info".language .

```
<!-- AWP_In_Variable Name='"COOKIE_VALUE:siemens"' -->
```

The HTTP-header variable is written in the same-name PLC variable.

Enumeration types

Enumeration types (enums)

Numerical values from the PLC program can be converted into text and vice versa using enums. The numerical values can also be assigned for several languages.

Creating enums

Enter an AWP command using the following syntax at the start of the HTML file:

```
<!-- AWP_Enum_Def Name="<Name of the enum type>"
Values='0:"<Text_1>", 1:"<Text_2>", ... , x:"<Text_x>"' -->
```

For example, for German values to be saved as an HTML file in the "de" folder of the HTML directory:

```
<!-- AWP_Enum_Def Name="Enum1" Values='0:"an", 1:"aus", 2:"Störung"' -->
```

For example, for English values, to be saved as an HTML file in the "en" folder of the HTML directory:

```
<!-- AWP_Enum_Def Name="Enum1" Values='0:"on", 1:"off", 2:"error"' -->
```

Assigning enums

Tags are assigned from the user program to the individual enum texts using a special AWP command:

```
<!-- AWP_Enum_Ref Name="<VarName>" Enum="<EnumTypeName>" -->
```

<VarName> is thereby the symbolic name from the user program and <EnumTypeName> is the previously set name of the enum type.

Note

In each fragment in which enum texts are referenced by a PLC tag, this PLC tag must be assigned by the appropriate AWP command of the enum type name.

Ensure that no AWP command for importing fragments is positioned between an enum assignment and enum usage because this import can result in the enum assignment lying in a different fragment than the enum usage.

Example

Enum type "state" is defined with values "0" and "1". "0" means "off", "1" means "on":

```
<!-- AWP_Enum_Def Name="state" Values='0:"off", 1:"on"' -->
```

The following code is contained in the HTML code of the web page to be output:

```
<!-- AWP_Enum_Ref Name="operating state" Enum="state" -->
:=operating state:
```

Depending on the value of the "operating state" tag, the ´result displayed is no longer "0" or "1", but "off" or "on".

Simplified use of enumeration types

At S7-1200 CPUs as of firmware version 4 it is possible to use enumerations directly in AWP commands to read and write PLC variables.

You create enums as described in the previous section, and you can then utilize the values with user program read and write commands.

Creating enums

```
<!-- AWP_Enum_Def Name="<Name des Enum Typs>" Values='0:"<Text_1>",
1:"<Text_2>", ... , x:"<Text_x>"' -->
```

Utilizing enums in the user program read and write commands

```
<!-- AWP_In_Variable Name='<Varname>' Enum="<EnumType>" -->
<!-- AWP_Out_Variable Name='<Varname>' Enum="<EnumType>" -->
```

Example of reading PLC tags

```
<!-- AWP_Enum_Def Name='AlarmEnum' Values='0:"No alarms", 1:"Tank is
full", 2:"Tank is empty"' --><!-- AWP_Out_Variable Name='"Alarm"'
Enum="AlarmEnum" -->...<p>The current value of "Alarm"
is :="Alarm":</p>
```

If the value of "Alarm" in the CPU is "2", the following text will be displayed on the HTML page:

'The current value of "Alarm" is Tank is empty' because the enum definition assigns the string "Tank is empty" to the numerical value 2.

Example of writing PLC tags

```
<!-- AWP_Enum_Def Name='AlarmEnum' Values='0:"No alarms", 1:"Tank is
full", 2:"Tank is empty"' --><!-- AWP_In_Variable Name='"Alarm"'
Enum='AlarmEnum' -->...
```

```
<form method="POST">
<p><input type="hidden" name='"Alarm"' value="Tank is full" /></p>
<p><input type="submit" value='Set Tank is full' /><p>
</form>
```

Because the enum definition assigns the string "Tank is full" to the numerical value "1", the value "1" is written to the PLC tag "Alarm".

Definition of fragments

Fragments

Fragments are "logical sections" of a web page to be processed by the CPU individually.

Fragments are usually complete pages but can also be individual elements such as files (for example, images) or complete documents.

Defining fragments

```
<!-- AWP_Start_Fragment Name="<Name>" [Type="<Type>"] [ID="<Id>"]
[Mode=<Mode>]-->
```

The start of a fragment is specified by this command. A fragment runs to the start of the next fragment or to the end of the file.

- <Name> Indicates the name of the fragment.
  The name must start with a letter [a-zA-Z] or an underscore (_). Letters, underscores or numbers [0-9] can follow after this first character.
- <Type> Indicates the type of the fragment.
  - "manual" The user program is informed of the request for a fragment; the web page to be returned can be changed by the user program.
  - "automatic" The page is automatically processed (default).

- <id> A numeric ID can be stipulated for the fragment. If no ID is assigned, the fragment is automatically assigned an ID. For manual pages (<Type>=manual) , the fragment can be addressed in the user program of the CPU by this ID.

  Note

  Keep the ID low because the highest ID influences the size of the Web Control DB.

- <Mode> Fragments support the visible and hidden modes.
  - "visible" The fragment is a part of the web page. This mode is preset and can also be omitted.
  - "hidden" The fragment is not part of the web page. However, the fragment will be saved in the Web DB and is available to the user program for inserting in a requested web page. You use an exchange of the fragment ID (Web-Control-DB.fragment_index tag) to insert a "hidden" fragment in the requested web page.

The input document is completely divided into fragments by the "AWP_Start_Fragment" command. "AWP_End_Fragment" is therefore unnecessary.

Without a start fragment command, a file is mapped as a fragment; the fragment name is derived from the file name. If a file is divided into several fragments (by "AWP_Start_Fragment"), the file must begin with the "AWP_Start_Fragment" command.

Importing fragments

You can declare a fragment in an HTML page and import this fragment into other web pages.

Example

A company logo is to be displayed on all web pages of a web application.

There is only one instance of the HTML code for the fragment that displays the company logo. You can import the fragment as often and into as many HTML files as required.

Syntax

```
<!-- AWP_Import_Fragment Name = "<name>"-->
```

- <name> is the name of the fragment to be imported.

Example

HTML code within a web page that declares a fragment:

```
<!-- AWP_Start_Fragment Name = "My_Company_Logo"-->
<p><img src = "compay_logo.jpg"></p>
```

Example

HTML code within another web page that imports the declared fragment:

```
<!-- AWP_Import_Fragment Name = "My_Company_Logo"-->
```

Creating and loading a data block

Requirement

- All source files required for the web application (*.html, *.js, *.png, ...) have been created.
- The source files are located in one folder, but only those source files that are required for the web application. No other files may be located in this folder.

Note

Length of file names and tag names

If you have a comprehensive web application with many files and directories, the generation of the web data blocks may possibly fail. If this happens, the generation is aborted with the message "Text list overflow...". The cause is system-internal size limitations for management information saved in the web data block.

Remedy: Use short file names and short tag names.

Procedure

To create data blocks from the source files for user-defined web pages in STEP 7, proceed as follows:

1. Select the CPU, for example, in the device configuration.
2. Select the properties for user-defined web pages in the inspector window under "Properties > General > Web server".
3. As "HTML source", select the folder that contains the source files for the web application.
4. Enter the HTMP page to be opened on starting the web application as the start HTML page.
5. Enter a name for the application if required.
6. You can supplement a range of file name extensions as "Files with dynamic content" if necessary. Only enter those file name extensions that also contain AWP commands.
7. The number for the Web Control DB and for the fragment DB start number can be kept as long as they are not already being used by your user program.

8. Click on the "Generate" button to create DBs from the source files.
   The generated data blocks are saved in the project navigation in the "System block" folder (in the "Web server" subfolder).
9. In the CPU, select the network view to be loaded and then select the "Download to device" command in the "Online" menu to download the blocks. Compilation of the blocks is implicitly initiated before downloading.
   If errors are reported during this process, you must correct these errors before you can download the configuration.

Structure of the PLC program

Your user program must call the " WWW" instruction to even allow the web application, for example, the user-defined web pages, to be available to the CPU on the standard web pages and to allow them to be called up there.

The Web Control DB you have created from the source files is the input parameter (CTRL_DB) for the "WWW" instruction. The Web Control DB references the content of the user-defined web pages coded in the fragment DB and then receives status and control information.

Calling the "WWW" instruction in the startup program

If you do not want the user program to influence requested web pages, it is sufficient to only call the "WWW" instruction once in a startup OB. This instruction initializes communication between the web server and the CPU.

Calling the "WWW" instruction in the cyclic program

The "WWW" instruction can also be called in an OB processed in cycles (for example, OB 1). This has the advantage of being able to respond to web server requests from within the user program. Manual fragments must be used for this.

In this case, you must evaluate information from the Web Control DB in order to identify the requested web page or the requested fragment. On the other hand, you must set a bit in the user program in order to explicitly release the web page to be returned by the web server after processing the web page request.

The structure of the Web Control DB is described in the following section.

Web Control DB

The Web Control DB (DB 333 by default) is created by STEP 7 and contains information on the structure of user pages, the status of communication and any errors that occur.

Additional fragment DBs are also created as well as the Web Control DB. These fragment DBs (there may also only be one fragment DB) are referenced in the Web Control DB. The fragment DBs contain the web pages and media data coded in fragments, for example, images. The content of the fragment DB cannot be changed by the user program. It is created automatically and is only for data management.

The status and control tags of the Web Control DB are accessed via symbols.

The following lists the tags of the Web Control DB required for status evaluation and to control interaction.

The Web Control DB provides two types of information:

- Global status information: Not bound to a concrete web page request.
- Request status and control information: Information about queued requests.

Global status information

"WEB-Control_DB".commandstate.init	Activates and initializes the web application.
"WEB-Control_DB".commandstate.deactivate	Deactivates the web application.
"WEB-Control_DB".commandstate.inititializing	The web application is initialized (read Web Control DB, etc.).
"WEB-Control_DB".commandstate.error	Web application could not be initialized. The reason is coded in "WEB-Control_DB".commandstate.last_error .
"WEB-Control_DB".commandstate.deactivating	The web application is closed.
"WEB-Control_DB".commandstate.initialized	The web application has been initialized and is ready.
"WEB-Control_DB".commandstate.last_error	Refer to the next table for a value table of possible errors.

Last_error	Description
1	Fragment DB is inconsistent (does not match the Web Control DB).
2	A web application already exists with this name.
3	Memory problem initializing in the web server.
4	Inconsistent data in the Web Control DB.
5	A fragment DB is not available (not loaded).
6	No AWP ID for a fragment DB.
7	The enum fragment is not available (contains the texts and information on the enum types).
8	An action requested via the command flag in the Web Control DB is prohibited in the current state.
9	Web application is not initialized (if there is no reinitializing after disabling).
10	Web server is disabled.
...	Last_error is reset once the web application has been successfully initialized.

Request status information

Request status information is bound to one of four possible requests, x = [1 … 4].

"WEB-Control_DB".requesttab[x].idle	Nothing need be done.
"WEB-Control_DB".requesttab[x].waiting	The user program must react to a request from a manual fragment and explicitly initiate further processing in the web browser.

"WEB-Control_DB".requesttab[x].sending	The web server is occupied with processing the request/fragment.
"WEB-Control_DB".requesttab[x].aborting	The TCP connection is closed by the web server.

Request control information

Request control information is bound to one of four possible requests, x = [1 … 4].

"WEB-Control_DB".requesttab[x].continue	Releases the fragment being processed for transmission. Processing of the next fragment is initiated.
"WEB-Control_DB".requesttab[x].repeat	Releases the fragment being processed for transmission. The fragment is then processed again.
"WEB-Control_DB".requesttab[x].abort	Closes the TCP connection.
"WEB-Control_DB".requesttab[x].finish	Releases the fragment being processed for transmission. Stops further processing of requests (terminates the request).

Example:

The tag for the DB is: "WEB-Control_DB". Whether errors have occurred during initialization of the web application can be determined by requesting bit "WEB-Control_DB".commandstate.error in the user program.

If an error has occurred you can analyze it using the "WEB-Control_DB".commandstate.last_error value.

Interaction with the user program

With the help of manual fragments, you can make sure that the user program reacts synchronously to browser entries so that the returned website can be prepared by the user program.

Fragment type

To react directly to the received data in the user program, the "manual" fragment type (for "manual sites") must be used for the fragment that the data writes.

```
<!-- AWP_Start_Fragment Name="testfrag" ID="1" Type="manual" -->
```

The transmission of values to the CPU web server occurs in the same way for automatic and manual sites.

Example:

```
<form method="POST" action="">
<p>
<input type="submit" value="Set new value">
<input type="text" name='"Velocity"' size="20">
```

```
</p>
</form>
```

User program for manual fragments

In the CPU user program, the "WWW" (SFC 99) instruction must be cyclically opened with the usage of manual sites.

To react to the values entered in the browser, it is required to evaluate the request that the manual site places for the web server in the user program. The web-control-DB (for example, DB 333) must investigate pending requests to do this. In the section "requesttab" from the web-control-DBs you will find the array that manages four requests. Each element of the array contains information about the respective request in a structure.

A simple AWL program example shows how pending requests are checked provided the variables from the web-control-DB.

```
      U     "WEB-Control_DB".requesttab[1].waiting
      SPBN  R2
      L     "WEB-Control_DB".requesttab[1].fragment_index
      L     1
      SPA   PdRq                        // request pending

R2:   U     "WEB-Control_DB".requesttab[2].waiting
      SPBN  R3
      L     "WEB-Control_DB".requesttab[2].fragment_index
      L     2
      SPA   PdRq                        // request pending

R3:   U     "WEB-Control_DB".requesttab[3].waiting
      SPBN  R4
      L     "WEB-Control_DB".requesttab[3].fragment_index
      L     3
      SPA   PdRq                        // request pending

R4:   U     "WEB-Control_DB".requesttab[4].waiting
      SPBN  NoRq
      L     "WEB-Control_DB".requesttab[4].fragment_index
      L     4
      SPA   PdRq                        // request pending

NoRq: L     0
      T     #frag_index
      T     #req_index
      SPA   Exit

PdRq: T     #req_index
      POP
      T     #frag_index

Exit: BE
```

If there is a request present, this AWL program section writes the fragment ID in the variable #frag_index and the request number (value range 1-4) in the variable #req_index.

With this information, the information transferred in the request can now be processed in the AWL program for each fragment ID separately (for example, plausibility check).

If the processing of the request by the AWL program is complete, the request must be answered and the corresponding entry under "requesttab" from the web-control DB (for example, DB 333) must be reset.

A simple AWL program example for answering requests:

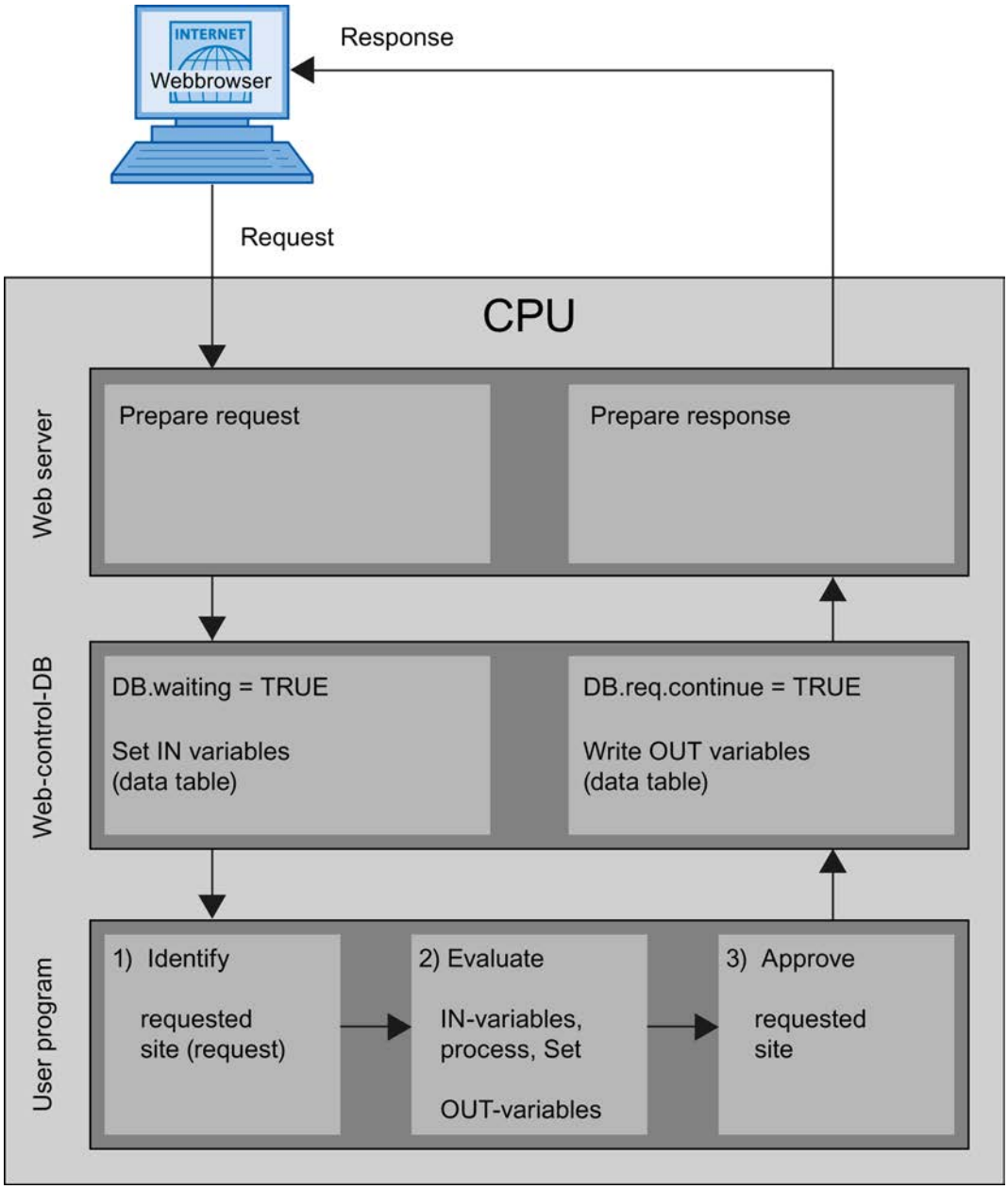
```
SET
      L     #req_index
      SPL   Con0
      SPA   Con0
      SPA   Con1
      SPA   Con2
      SPA   Con3
      SPA   Con4
Con0: SPA   Exit

Con1: =     "WEB-Control_DB".requesttab[1].continue
      SPA   Exit
Con2: =     "WEB-Control_DB".requesttab[2].continue
      SPA   Exit
Con3: =     "WEB-Control_DB".requesttab[3].continue
      SPA   Exit
Con4: =     "WEB-Control_DB".requesttab[4].continue
      SPA   Exit

Exit: BE
```

Basic procedure of a browser request with the interaction of the user program

The following image shows the simplified, basic procedure from the web browser request to the effects on the web-control-DB contents and the required actions of the user program to the return of the prepared website (response).



Displaying custom web pages in the browser

Display web pages in browser

Web pages are called from the standard web pages of the web browser.

In addition to the other links in the navigation bar, the standard web pages also have a link to "user pages".

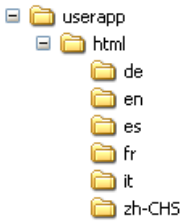
Click on the "user pages" link to open the web browser you have configured as the default HTML page.

Create multilingual user defined websites

You have the possibility to provide your user-defined websites in the different languages.

Requirements

The language-dependent HTML sites will be filed in a folder structure that contains the following language abbreviations:



Defined language abbreviations

The language abbreviations "de", "en", "fr", "es", "it" and "zh-CHS" are defined. Other language folders or other language folders labeled differently are not supported.

If needed, you can create more folders in the same folder hierarchy for other files; for example, an "img" folder for images and a "script" folder for Java script files.

Language switching for custom web pages

Requirements

The HTML pages are contained in the predefined language folders, for example, HTML pages with German text are in the "de" folder, HTML pages with English text are in the "en" folder.

Language switching concept

Language switching is based on a predefined cookie named "siemens_automation_language". If the cookie is set to value "de", at the next web page request or web page update, the web server switches to the web page from the "de" folder.

Similarly, the web server switches to the web page from the "en" folder when the cookie is set to "en".

Example of a language switch

The example is designed as follows:

- In both language folders "de" and "en", you will find the language-dependent HTML files with the same names, for example, "langswitch.html". The text displayed within the two files is in German or English corresponding with the folder name.
- Additionally, there is a folder "script" in the folder structure in which the Java script file "lang.js" is located. Functions are stored in this file that are required for the language switch.

Design of the "langswitch.html" file ("de" folder)

The meta data "Content-Language", charset and path to the Java script file are set in the header of the file.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Language" content="de">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Language Switch German Site</title>
<script type="text/javascript" src="script/lang.js" ></script>
</head>
```

In the body of the file, the language selection occurs with the help of the HTML element "select". The select element activates a selection list and contains the options "de", labeled with "German" and "en", labeled with "English"; "de" is preselected.

With the help of the event handler "onchange", the function "DoLocalLanguageChange(this)" is opened. The "this" parameter transfers the select-object with the selected option to this function. "onchange" executes the option to open the function upon each change.

```
<!-- Language Selection -->
<table>
    <tr>
        <td align="right" valign="top" nowrap>
         <!-- change language immediately on change of the selection
-->
          <select name="Language"
onchange="DoLocalLanguageChange(this)" size="1">
                <option value="de" selected >German</option>
                <option value="en" >English</option>
            </select>
        </td>
     </tr>
</table>
<!-- Language Selection End-->
```

Design of the file "langswitch.html" (folder "en")

The header of the HTML file with English text is designed analog to the HTML file with German text.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
```

```
<meta http-equiv="Content-Language" content="en">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Language switching english page</title>
<script type="text/javascript" src="script/lang.js" ></script>
```

In the body of the file, the language selection occurs with the help of the HTML element "select". In contrast to the German HTML file, the English option is preselected and the text and labels are in English.

```
<!-- Language Selection -->
<table>
    <tr>
        <td align="right" valign="top" nowrap>
         <!-- change language immediately on change of the selection
-->
          <select name="Language"
onchange="DoLocalLanguageChange(this)" size="1">
                <option value="de" >German</option>
                <option value="en" selected >English</option>
            </select>
        </td>
     </tr>
</table>
<!-- Language Selection End-->
```

Design of the "lang.js" file (in the "script" folder)

The "DoLocalLanguageChange" function is defined in the Java script file, which opens the "SetLangCookie" function with the language selection value. SetLangCookie combines the cookie name and cookie value and sets the cookie through the corresponding document.cookie property. To make the web server react to the setting of cookies with the display of the desired language, the website must be reloaded (top.window.location.reload).

```
function DoLocalLanguageChange(oSelect) {
        SetLangCookie(oSelect.value);
        top.window.location.reload();
    }

 function SetLangCookie(value) {
        var strval = "siemens_automation_language=";
            // this is the cookie by which the web server
            // detects the desired language
            // this name is required by the web server
        strval = strval + value;
        strval = strval + "; path=/ ;";
            // set path to the application, since otherwise
           // path would be set to the requesting page would not get
the cookie
            /*
            use expiration if this cookie should live longer
            than the current browser session
            var now      = new Date();
            var endttime = new Date(now.getTime() + expiration);
```

```
                strval = strval + "; expires=" + endttime.toGMTString()
+ ";";
                */
            document.cookie = strval;
        }
```

Additional information

Information and examples for user-defined pages

Examples and further explanations are available in the Service & Support portal.

- An example for accessing the web server of an S7-1200 using an iPad and for integrating the jQuery library is available here (http://support.automation.siemens.com/WW/view/en/62046437?Datakey=47071380).
- An application example for the creation of a user-defined web page is available here (http://support.automation.siemens.com/WW/view/en/58862931).
- An overview of the topic of industrial security is available here (http://support.automation.siemens.com/WW/view/en/50203404).

8.1.4.4 Configuring automation systems (S7-1200)

Addressing modules

Addressing modules

Introduction

In the device overview, you see the addresses or address ranges of the modules in the I address and Q address columns. There are other addresses as well, which are explained below.

I/O address

I/O addresses (input/output addresses) are required to read inputs and set outputs in the user program.

Input and output addresses are assigned automatically when modules are inserted in the rack. The address of the first channel is the start address of a module. The addresses of the other channels are derived from this start address. The address end is obtained from the module-specific address length.

Device address (e.g., Ethernet address)

Device addresses are addresses of programmable modules (Industrial Ethernet addresses). They are required to address the different stations of a subnet, for example, to download a user program to a CPU.

Hardware identifier for identifying modules and submodules

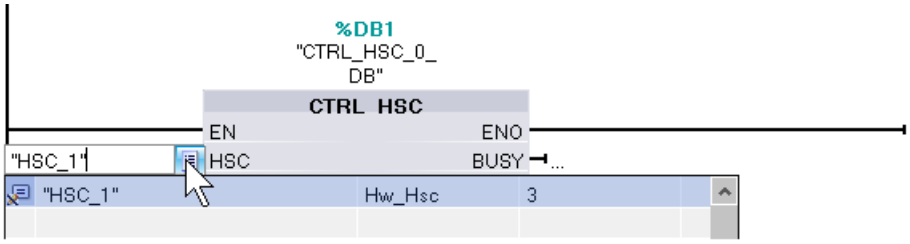
In addition to the I addresses and Q addresses, a hardware identifier (HW ID) is assigned automatically and is used to address and identify the module. Submodules (units of a module), such as an integrated counter, also receive such a hardware identifier.

The hardware identifier consists of an integer and is output by the system with diagnostics alarms to allow the faulty module or the faulty submodule to be localized.

In addition, the hardware identifier is used for a number of instructions to address the corresponding module.

The hardware identifier cannot be changed.

Example: Identifying high-speed counters of the S7-1200 CPU



The hardware identifier is assigned automatically when components are inserted in the device or network view and in the PLC tags. A name is also assigned automatically for the hardware identifier. The system constants of the PLC tags cannot be changed either.

See also

Specifying input and output addresses (Page 938)

Assigning addresses to a location in the program (Page 939)

Introduction to loading a configuration (Page 1553)

Inspector window (Page 487)

Specifying input and output addresses

Default input and output addresses are set automatically. You can, however, change the address assignment later.

All addresses of modules are located in the process image area. The process image is automatically updated cyclically.

Requirement

You are in the device view.

Procedure

To change the preset address range proceed as follows:

1. In the device view, click on the module for which you want to set the start address.
2. Go to "I/O addresses" in "Properties" in the inspector window.
3. Under "Start address" enter the required start address.
4. Press <Return> or click on any object to accept a modified value.

If you have entered an invalid address, a message indicating the next available address is displayed.

Note

You can also change the addresses directly in the device overview.

See also

Editing properties and parameters (Page 513)

Input and output addresses in the address overview (Page 515)

Assigning addresses to a location in the program

You can assign addresses of the I/O channels of modules directly to the points of use in the program or to a tag table.

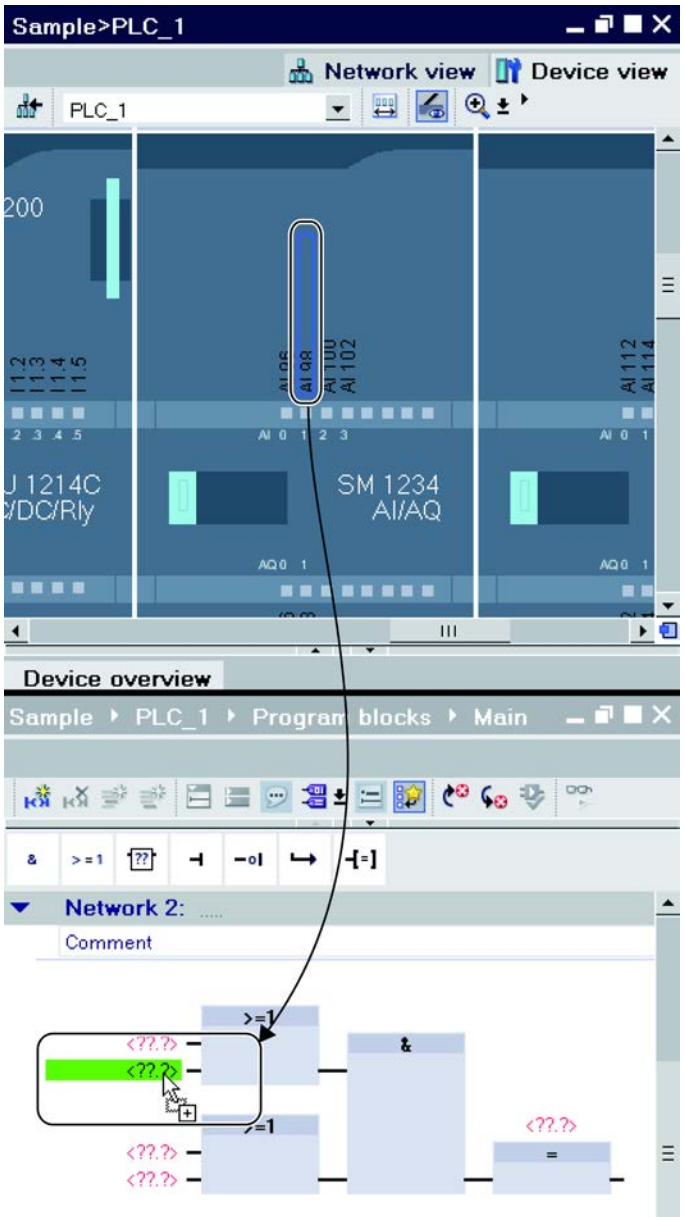
Requirement

- The device view of the hardware and network editor is open.
- The zoom level in the device view must be set to at least 200% to allow you to see the individual I/O channels.
- The instruction window of the programming editor or a tag table is open.

Procedure

To assign I/O channels of modules to the points of use in the program or to a tag table, follow these steps:

1. In the device view, navigate to the module with the desired I/O channel.
2. Click and hold down the mouse button to drag the desired I/O address to the corresponding point of use of the block or to the tag table.



The address of the module is assigned to the point of use in the program or entered as a tag in the tag table.

Note

The tag for an input or output of a block can also be dragged to the input or output of a module in order to link the tag to the I/O channel of the module.

Signal board

Inserting a signal board in a CPU

Introduction

Signal boards allows you to increase the number of the S7-1200 CPU's own inputs and outputs. Just like all other hardware components, you will find signal boards in the hardware catalog. However, you do not insert signal boards in the rack like other modules but directly in a slot of the CPU itself.

Note the following points when using a signal board:

- Each CPU can have only one signal board inserted in it.
- A signal board can only be inserted when the slot in the CPU is free.

There are various ways of inserting a signal board in a CPU:

- Double click on a signal board in the hardware catalog when there is a free slot in the CPU
- Drag from the hardware catalog to a free slot in the CPU
- Shortcut menu of a signal board in the hardware catalog for copying and pasting

Requirement

- The hardware catalog is open.
- The S7-1200 CPU has a free slot for the signal board.

Inserting a signal board in a CPU

To insert a signal board in a CPU, proceed as follows:

1. Go to the required signal board in the hardware catalog.
2. Select the signal board you want to use.
3. Drag the signal board to the free slot in the CPU.

You have now inserted a signal board in the slot of the CPU.

If you are in the network view, you can also drag a signal board to a device. If the CPU has a an empty slot for a signal board, the signal board is inserted automatically into this slot.

Configurations for Web server

Information about the web server

Introduction

The web server allows you to monitor the CPU via the Internet or the intranet of your company. This permits evaluation and diagnostics over long distances.

Alarms and status information are visualized on HTML pages.

Web browser

You need a web browser to access the HTML pages of the CPU.

The following PC web browsers, for example, are suitable for communication with the CPU:

- Internet Explorer (Version 8.0, 9.0)
- Mozilla Firefox (Version 17.0.1 and higher)
- Google Chrome 23.0
- Apple Safari 5.1.7 (Windows)
- Apple Safari 6.0.2 (Mac)

The following Web browsers of mobile devices are also suitable:

- Internet Explorer 6.0 for HMI Panels
- Mobile Safari (iOS 5.0.1)

- Mobile Android Browser 2.3.4
- Mobile Google Chrome 23.0

Reading information via the web server

The following information can be read from the CPU. The availability of the respective web pages depends on the CPU and its firmware version.

As of firmware version 4, the web pages are also available in several languages.

Page/information	Description
Intro/Introduction	Entry page for the standard web pages
Start Page Start page with general CPU information	The start page provides an overview of general information on the CPU, the name of the CPU, the type of CPU and basic information on the current operating state.
Identification Identification information	Displays the static identification information such as serial number, order number and version numbers.
Diagnostic Buffer Diagnostic information	Displays the content of the diagnostics buffer with the most recent entries first.
Module Information Module information	Displays whether the centrally inserted components of a station are OK, whether there are maintenance requirements or components cannot be reached, for example. As of firmware version 4 a firmware update is possible via this Web page.
Communication Communication	Displays the communication connections during open communication (OUC); displays resources and address parameters.
Varable Status Tags	Displays the status of operands of the user program to monitor and change the values.
Data Logs (File Browser as of firmware version 4)	Data logs in CSV format to transfer to the hard disk of the programming device. The data logs are created with data log instructions in the user program and filled with data. As of firmware version 4 you have access to files of the internal load memory and of the external load memory (Memory Card), for example to the content of the directory "DataLogs" and "Recipes" via the Web page "File Browser".
User Pages User pages (if user-defined web pages have been configured and loaded)	The user web pages deliver a list of web pages with customer-specific web applications.

Web access to the CPU via PG/PC

Proceed as follows to access the web server:

1. Connect the client (PG/PC) to the CPU via the PROFINET interface.
2. Open the web browser.
   Enter the IP address of the CPU in the "Address" field of the web browser in the format http://ww.xx.yy.zz (example: http://192.168.3.141).
   The start page of the CPU opens. From the start page, you can navigate to further information.

See also

S7-1200 web server (http://support.automation.siemens.com/WW/view/en/36932465)

Standard web pages

Requirements for web access

The requirements for access to standard CPU web pages are explained in the following, as well as the effects of missing or existing configuration information.

Requirement

The web server must be started.

The web server only starts when it has been activated in the properties of the CPU in the "Web server" section.

Note the following:

The web pages are normally transmitted via an non-secure connection and are not secured against hacker attacks. If you want to transfer the web pages in encrypted form to the browser, use the URL https://, followed by the IP address of the CPU.

Logon

No logon is required to access the standard web pages read-only. A user must be logged on to execute certain actions like changing the operating mode of the CPU or for write access.

For S7-1200 CPUs up to FW version V3:

You must be logged on as "admin" for the actions listed above. The logon input boxes are on the top left of each standard web page.

If you log on as "admin", you must enter the user name and password there.

Name: admin.

Password: configured CPU password (for password-protected CPU).

For S7-1200 CPUs as of FW version 4:

You can freely select the names of users and the passwords (CPU parameter "Web server", area "User management").

You assign rights to the users, for example the right to query diagnostics or to update the firmware.

JavaScript and cookies

The standard web pages use JavaScript and cookies. You must enable both in your web browser.

If JavaScript is not enabled, the following limitations apply:

- Data from standard web pages is not automatically updated.
- You cannot log on as user.
- Fields cannot be sorted (module information)

If cookies are not enabled, you cannot log on.

See also

Access for HTTPS (Page 946)

Settings for operation

Settings for operation

To be able to use the web server of an S7-1200-CPU, you must select the CPU in the network view or the device view and make the following settings in the inspector window under "Properties > General > Web server":

- Enable the web server
- Restricting access to the CPU to HTTPS transmission protocol (encrypted transmission) Access via port 80 is then blocked. Communication is only possible via port 443.

- Enabling automatic update of web pages
  The update interval is set by default and cannot be changed. The CPU updates web pages with changing content ((for example, status information or diagnostics information) at regular intervals.
- Creating and managing users
  Users are exclusively entitled to options that are assigned to the access rights. Depending on the CPU and firmware used, you can assign different user rights. Rights that your CPU does not support cannot be activated. A user called "Everybody" with minimal access rights, that you can, however, extend, is set by default in the user list. A user who uses the Web server without entering a password has "Everybody" user rights.
  You have the possibility of configuring further users with different access rights. These users have to log on with the configured user name and password.

WARNING

Unauthorized access to the CPU by means of the Web server

Unauthorized access to the CPU or the changing of PLC variables to invalid values can result in interruptions of the processes controlled by the CPU and cause death, serious bodily harm or material damage!

Since the activation of the Web server allows authorized persons for example to change operating modes, to write-access CPU data or to update the firmware, we recommend that the following security measures be taken:

- If possible limit access to the HTTPS protocol.
- Create users with secure passwords. An example of a secure password is one which is only used for a single application, is more than 8 characters long, and consists of lower- and upper-case letters as well as symbols and numbers (?!+%$1234...). In addition, passwords based on common keyboard sequences or words from the dictionary should be avoided.
  Change the password at regular intervals.
- Do not extend the rights of the "Everybody" user.
- Check the PLC variables in the user program and limit the range of values to permissible ranges since users can set invalid values via the Web server.

Access for HTTPS

Access via HTTPS

HTTPS is used for encrypting and authentication of communication between the browser and web server.

To transfer data between the browser and the CPU using the HTTPS protocol, enter the URL as https://ww.xx.yy.zz in the address line of your browser, whereby ww.xx.yy.zz stands for the IP address of the CPU.

You require a valid, installed certificate for error-free HTTPS access to the CPU.

If no certificate is installed a warning is displayed with a recommendation not to use this page. To view the page you must explicitly "Add exception".

You can receive a valid certificate (Certification Authority) "SIMATIC CONTROLLER" as a download from the "Intro" web page under "Download certificate". The help function for your respective web browser provides information on how to install a certificate.

Accessing data of the CPU memory

You can access data in the internal or external CPU load memory by means of a standard web page.

- Use the web page "Data logs" for S7-1200 CPUs up to and including FW version 3. From this web page, you transfer the data logs from the CPU to a drive on your PC.
- Use the web page "File Browser" for S7-1200 CPUs as of FW version 4. From this web page, you transfer data from the folders "Data logs" or "Recipes", for example, to a drive on your PC.

Depending on the file type and the access permissions you have configured for the web server user, you can download, delete, rename or upload the files. The actual directories can only be created, deleted or renamed.

Example: Data logs

To open a data log, click on the link of the desired data log. You can then open the file (.csv), for example, in Microsoft Excel or in another program you choose or you can save the file.

Special note: Data logs are saved in U.S. American CSV format. You can only open the file directly using the U.S. version of Microsoft Excel. If you are using another national version of Microsoft Excel, you must import the file, selecting "comma" in the import assistant as the delimiter.

Downloading a data log

To download a data log, click on the download icon of the desired data log. You can then open the file (.csv), for example, in Microsoft Excel or in another program you choose or you can save the file.

Downloading and clearing or deleting a data log

For a CPU with FW version up to V3.0:

To download and delete the current entries of the data log, you must be logged on. To do this, click on the "Download and delete" icon of the required data log. You can then open the file (.csv), for example, in Microsoft Excel or in another program you choose or you can save the file.

For a CPU as of FW version V4.0:

To reset the data log, follow these steps:

1. Open the CSV file, for example with Excel.
2. Delete the rows between the headline and the row with the entry "//END", if this row exists.

3. Save the file to a drive on your PC.
4. On the web page "File Browser", delete the data log (which means the CSV file) and load the prepared CSV file to the CPU with the "Upload file" button of the "File Browser" web page.

Additional information is available in the S7-1500 CPU system manual.

See also

Data logging - Overview (Page 4774)

Create and download user-defined websites

What you need to know about user-defined web pages

Concept

The concept of user-defined web pages allows you to access freely-designed web pages of the CPU from a web browser. The web server of the CPU provides this function.

You are not dependent on special tools for the design and functionality of the user-defined web pages. You can adapt the pages in the layout with CSS, provide dynamic content with JavaScript or use any framework to produce web pages.

The totality of files processed by the web server is also referred to as the "web application".

Web application and user program

Using HTML code in user-defined web pages, you can also transmit data via a web browser to the user program of the CPU for further processing and can display data from the operand area of the CPU in the web browser.

You can use script instructions (such as Javascript) to optimize your web pages, for example to dynamically change contents or validate user entries.

To synchronize between the user program and the web server, but also to initialize, you must call the WWW (SFC 99) instruction in the user program.

- If no interaction is required between the web application and the user program, for example, if a web page only provides static information, only initialization in the user program is required.
- If a simple data exchange is necessary between PLC tags and tags in web applications, to display the contents of PLC tags or write a value in a PLC tag for example, the syntax for reading and writing tags has to be observed. In this case only an initialization is required in the user program, for example in the startup OB.
- If a further interaction is required between the web application and the user program, you must handle status and control information from the Web Control DB in addition to the synchronization between Web server and user program. This is the case, for example, when user entries are transmitted via the web browser to the web server for evaluation by the CPU. Unlike simple data exchange, the user program directly influences the time at which the requested web page is relayed back to the web browser. In this case, you must be acquainted with the concept of manual fragments and the structure of the Web Control DB.

Initialization

User-defined web pages are "packaged" in data blocks for processing by the CPU. You must generate appropriate data blocks from the source data (HTML files, images, JavaScript files, etc.) during configuration to be able to download the web application into the CPU. The Web Control DB takes on a special role (default: DB 333). It contains status and control information as well as links to additional data blocks with coded web pages. Data blocks that contain coded web pages are termed "Fragment DBs".

When the data block is downloaded into the CPU, the CPU does not "know" that user-defined web pages are coded inside it. The "WWW" (SFC 99) instruction, for example, in the Startup OB informs the CPU which DB is the Web Control DB. The user-defined web pages can be accessed via a web browser after this initialization.

Synchronization

If the user program is to exchange data with the user-defined web pages, the WWW (SFC 99) instruction must be used in the cyclic program section.

Examples of interaction between user program and web page:

- Check received data
- Compiling and returning data for web browser making request

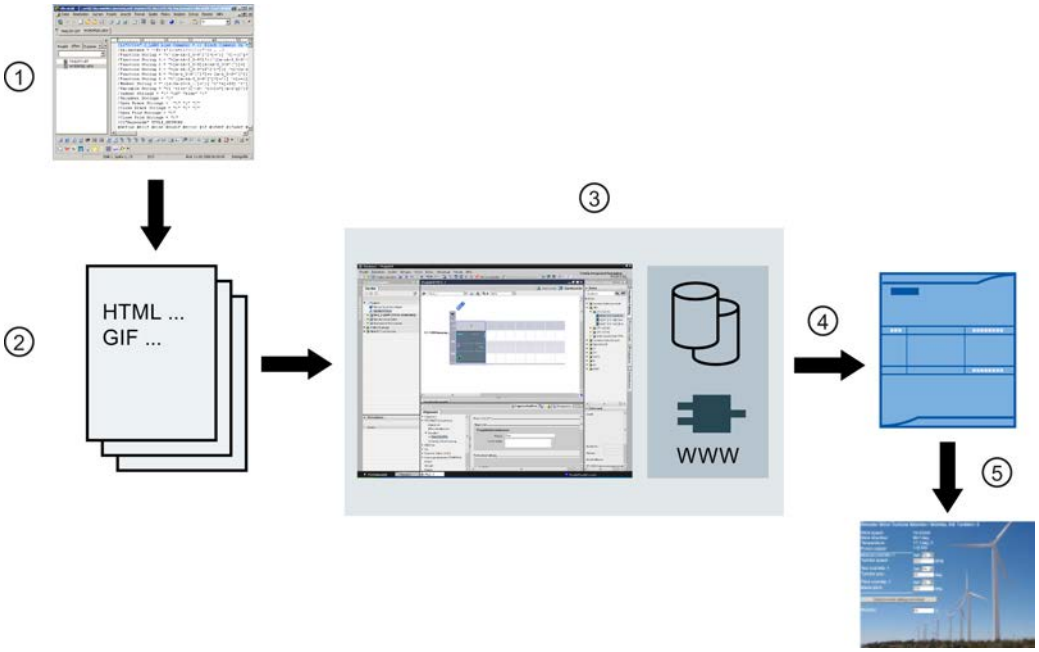
In this case, the status information must be able to be evaluated and control information must be transmitted to the web server, for example, to release a requested web page.

Procedural overview

Basic information

This section provides a step-by-step explanation of the basic procedure used to create and download custom web pages and to use them in the operating phase.

The following graphic provides a simplified representation of the process used in creating and displaying custom web pages:



① Programming a web application (using suitable tools when required and AWP commands for dynamic pages when applicable).

② The web application is comprised of single source files, for example, *.html, *.gif, *.js, etc.

③ Using STEP 7:

- Generate the data blocks (Web Control DB and fragment DBs) from source files. The DBs contain meta information and the complete web application, including the images and the dynamic and static parts of the web application. The DBs are stored under "System blocks" in the project tree.
- Call the "WWW" instruction in the user program. This instruction initializes the web server of the CPU for a web application.
- If required, complete final programming for interaction between the web server and user program

④ Downloading the blocks to the CPU.

⑤ Call the web page in the browser. The web pages of the CPU are called by entering the IP address of the CPU.

Additional information

You can find additional information and examples relating to the S7-1200 web server on the Internet (http://support.automation.siemens.com/WW/view/en/36932465).

Creating web pages

Web design tools from various companies can be used to create user-defined web pages. As a rule, the web pages should be programmed and designed compliant to the conventions of the W3C (World Wide Web Consortium). No check is made for compliance to W3C criteria in the web server of the CPU.

Rules

- The tool must be able to directly edit the HTML code so that the AWP command can be inserted into the HTML page.
  Only the AWP commands are parsed in the CPU and, for example, replaced by values from the user program/process image of the CPU.
- Files containing AWP commands must be coded in UTF-8. In the metadata of the HTML page, therefore, set the attribute charset to UTF-8 and save the file UTF-8 coded.
- Files containing AWP commands must not contain the following sequence:]]
- Files containing AWP commands must not contain the following sequence outside of the "Tag read ranges" (:=<Tag name>:): :=
  Tip: Replace the first character of a prohibited sequence with its character coding; for the colon, for example, :.

A small example for a custom web page should make clear the basic design.

Requirement

- The CPU must have a web server and the web server of the CPU must be activated.
- To be able to access PLC tags with write access as a user, you must be logged on as "admin".
- For the example below, PLC tags must be defined for those PLC tags that are to be shown on the web page. This is shown here for the first tab used, "Tank_below_max".

		Name	Data type	Address
1		Tank_below_max	Bool	%I0.0

Creating user-defined web pages

The following code for an example web page reads values from the process image and provides them in a table.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/
TR/html4/strict.dtd">
<html>
  <head>
   <meta http-equiv="Content-Type" content="text/html;
charset=utf-8">
   <title>Mix</title>
  </head>
   <body>
    <h1>Mix</h1>
    <h2> Actual State </h2>
    <table border="1">
     <tr>
      <th>Variable</th>
      <th>State</th>
     </tr>
     <tr>
      <td>Tank below max</td>
      <td>:="Tank_below_max":</td>
```

```
      </tr>
      <tr>
       <td>Tank above min</td>
       <td>:="Tank_above_min":</td>
      </tr>
     </table>
    </body>
</html>
```

AWP commands

The interface between a freely-programmable web application for a CPU that has a web server and the CPU data is declared by the AWP command (Automation Web Programming).

To develop web applications you are only subject to the restrictions of the web browser. In one of the programming languages of STEP 7, control with the user program which CPU data is displayed at what time in the web browser of the viewer. Use AWP commands, which you comment within the HTML files, to declare data to be used for intentional interaction between the web application and the user program.

AWP commands are inserted as HTML comments with a special syntax into HTML files; they declare the following features:

- Read PLC tags
- Write PLC tag
- Read special tags
- Write special tags
- Define enum types
- Assign tags to enum types
- Defining fragments
- Import fragments

Syntax of AWP commands

An AWP command begins with "`<!--AWP_`" and ends with "`-->`". In JavaScript files, the commands should also be enclosed by JavaScript comments ("`/*...*/`").

Notation rules for PLC tag names within an AWP command

The AWP commands "AWP_In_Variable" and "AWP_Out_Variable" contain a name attribute and optionally a use attribute. A PLC tag name is assigned to these attributes, by means of which the PLC tags in the browser are written or read. The following rules apply to handling PLC tag names in HTML code:

- PLC tags must be enclosed in quotation marks (" ... ").
- PLC tags used in AWP commands must also be enclosed by single quotation marks ('" ... "') or with quotation marks masked by a backslash ("\" ... \"").

- If the PLC tag name contains the character \ (backslash), this character must be designated with the escape sequence \\ as standard character of the PLC tag name.
- If the PLC tag name in the AWP command is also enclosed by single quotation marks and the single quotation mark (') occurs within the name, it must also be designed as normal character by the escape sequence \'.
- If an absolute address (input, output, bit memory) is used in AWP command, it is enclosed by single quotation marks.

PLC tag	PLC tag in HTML code
"Velocity"	<!-- AWP_In_Variable Name='"Velocity"' -->
	<!-- AWP_In_Variable Name="\"Velocity\"" -->
"abc\de"	<!-- AWP_In_Variable Name='"abc\\de"' -->
"abc'de"	<!-- AWP_In_Variable Name='"abc\'de"' -->
"abc'de"	<!-- AWP_In_Variable Name="abcde" Use'"abc\'de"' -->
"DB name".tag	<!-- AWP_In_Variable Name='"DB name".tag' -->
"DB name"."ta.g"	<!-- AWP_In_Variable Name='"DB name"."ta.g"' -->
-	<!-- AWP_Out_Variable Name=’flag1’ Use='M0.0' -->

See also

Reading tags (Page 953)

Writing tags (Page 955)

Special tags (Page 956)

Reading tags

User-defined web pages can read PLC tags.

The PLC tag must be specified by a PLC tag name.

These OUT variables (direction of output as viewed from the controller) are inserted at any location within the HTML text with the syntax described in the following.

Syntax

```
:=<varname>:
```

These references are replaced when the web server is in operation by the current values of the PLC tag in each case.

`<varname>` can be a simple, global PLC tag but also a complete tag path to a structural element.

Notation rules for PLC tag names

- PLC tags in HTML code are enclosed by quotation marks ("), if they are defined in the tag table. In the case of data block tags, the name of the data block is enclosed by quotation marks. If special characters are used in the structure elements of the data block, for example the dot (.) or blank, this part must also be enclosed by quotation marks.
- Quotation marks are not used for absolute addresses of inputs, outputs or bit memories.

PLC tag	PLC tag in HTML code
"DB_name".var_name	:="DB_name".var_name:
"DB_name"."var.name"	:="DB_name"."var.name":
"memory"	:="memory":
-	:=I0.0:
	:=Q0.0:
	:=MW100:
	:=%MW100:
"My_Data_Block".flag1	<!-- AWP_Out_Variable Name='flag1' Use='"My_Data_Block".flag1' -->
	...
	:=flag1:

- If the PLC tag name contains the character : (colon) or \ (backslash), this character must be designated with the escape sequence \: or \\ as standard character of the PLC tag name.

PLC tag	PLC tag in HTML code
"abc:de"	:="abc\:de":
"abc\de"	:="abc\\de":

- Special characters "<, &, >"
  Display problems can occur if these characters are contained in the tag name (for example, "a<b").
  Avoid expressions such as :="a<b": in the HTML page.
  To prevent display problems from occurring, use e.g. an AWP command with a use expression according to the pattern depicted below. The use attribute defines the PLC tag with the problematic character, the name attribute defines the name without problematic character, as it is used in the HTML page.

PLC tag	PLC tag in HTML code
"a<b"	<!—AWP_Out_Variable Name='simplename' Use='"a<b"' -->
	...
	:=simplename:

See also

AWP commands (Page 952)

Writing tags

Custom web pages can write data into the CPU.

This requires an AWP command that identifies the PLC tag to be written.

The PLC tag must also be specified by a PLC tag name.

The IN tags (direction of input as viewed from the controller) are placed on the browser page. This can be done, for example, in a form.

The tags are either set in the HTTP header (by cookie or POST method) or in the URL (GET method) by the browser and are then written by the web server into the respective PLC tag.

Syntax

To allow the IN tags to be written to the CPU, the tags must first be defined by an explicit AWP instruction:

```
<!-- AWP_In_Variable Name='<PLC_Varname1>' Name='<PLC_Varname2>'
Name='<PLC_Varname3>' -->
```

Several tags can be defined in an instruction - such as that shown above.

The specific PLC tag name is hereby written in double quotation marks; for example <PLC_Varname1> = "myVar".

In cases where the name of the tag that you use for the web application is not identical to the name of the PLC tag, the "Use" parameter can be used to assign to a PLC tag.

```
<!-- AWP_In_Variable Name=’<Webapp_Varname>’ Use=’<PLC_Varname>’
```

Example

The "AWP_In_Variable" AWP command is indispensable when handling forms.

```
<form method='post' action='/awp/appl/x.html'>
 <p>
  <input name='"var1"' type='text'>
  <input value='set' name='Button1' type='submit'>
 </p>
</form>
```

In the form defined above, the HTTP request method "post" is used to transfer the tag "var1" to the web server. The user places the "var1" tag in the form field. The tag 'Button1' has the value 'set', but is not required for the CPU. To allow the "var1" tag to be written to the CPU, the following instruction must be included in the same fragment:

```
<!-- AWP_In_Variable Name='"var1"' -->
```

Since PLC tags are enclosed in double quotation marks ("), the name in the AWP command must be enclosed in single quotation marks (') or in masked quotation marks (\"). To avoid the numerous escape sequences, we recommend the use of single quotation marks.

```
<!-- AWP_In_Variable Name=’"Info".par1’ -->
<!-- AWP_In_Variable Name="\"Info".par1\"" -->
```

Conditions for write access during operation

The following requirements have to be met in order for a user to be able to write to PLC tags from a user-defined web page.

- The CPU is is password protected.
- The user is logged in as "admin".

This rules applies to all writing access to web pages on a CPU.

See also

Requirements for web access (Page 944)

AWP commands (Page 952)

Special tags

Special tags are mainly HTTP tags set in the definition of the World Wide Web Consortium (W3C) . Special tags are also used for cookies and server tags.

The AWP command to read and write special tags differ only in that they have additional parameters than the AWP command used to read and write normal tags.

Reading a special tag

The Web server can read PLC tags and transfer these to special tags in the HTTP Response Header. You can, for example, read a URL for a diversion to another web page and transfer to the special tag HEADER:Location using the special tag HEADER:Location.

The following special tags can be read:

Name	Description
COOKIE_VALUE:name	Value of cookie with name: "name"
COOKIE_EXPIRES:name	Execution time of cookie with name: "name" in seconds (must be set beforehand).
HEADER:Status	HTTP status code (if no other value has been set, status code 302 is returned).
HEADER:Location	Path for forwarding to another page. Status code 302 must be set.
HEADER:Retry-After	Anticipated time in which the service is not available. Status code 503 must be set.
HEADER: …	All other header tags can also be forwarded in this way.

Use the AWP command "AWP_Out_Variable" to specify which PLC tags are to be transferred in the HTTP header to the web browser.

Basic structure:

```
<!-- AWP_Out_Variable Name="<Typ>:<Name>" [Use="<Varname>"] -->
```

Parameter description

- Name: Type and name of special tag
- Use (optional parameter): In cases where the name of the special tag is not identical to the name of the PLC tag, parameter "Use" can be used to assign to a PLC tag.

Example:

```
<!-- AWP_Out_Variable Name="COOKIE_VALUE:siemens" Use='"info".language' -->
```

Writing a special tag

In principle, all HTTP tags written in the HTTP header by the web browser can be evaluated by the user program of the CPU. Examples of tag types:

Name	Description
HEADER:Accept-Language	Accepted or preferred language
HEADER:Authorization	Proof of authorization for a requested resource
HEADER:Host	Host and port of the requested resource
HEADER:User-Agent	Information on the browser
HEADER: …	All other header tags can also be forwarded in this way
SERVER:current_user_id	Indicates whether a user is logged on (current_user_id=0: no user logged on)
SERVER:current_user_name	User name of the user logged on
SERVER:GET	Request method is GET
SERVER:POST	Request method is POST
COOKIE_VALUE:name	Value of cookie with name: "name"

The AWP command "AWP_In_Variable" is used to define which special tags are to be evaluated in the user program of the CPU.

Basic structure:

```
<!-- AWP_In_Variable Name="<Typ>:<Name>" [Use="<Varname>"] -->
```

Parameter description:

Name: Type and name of special tag

Use (optional parameter): In cases where the name of the special tag is not identical to the name of the PLC tag, the parameter Use can be used to assign to a PLC tag.

Examples:

```
<!-- AWP_In_Variable Name="COOKIE_VALUE:siemens" Use='"info".language' -->
```

The tag name in the HTTP header is replaced by the PLC tag name specified by Use . The cookie is written to the PLC tag "info".language .

```
<!-- AWP_In_Variable Name='COOKIE_VALUE:siemens' Use='"info".language' -->
```

The tag name in the HTTP header is replaced by the PLC tag name specified by Use. The cookie is written to the PLC tag "info".language .

```
<!-- AWP_In_Variable Name='"COOKIE_VALUE:siemens"' -->
```

The HTTP-header variable is written in the same-name PLC variable.

See also

AWP commands (Page 952)

Enumeration types

Enumeration types (enums)

Numerical values from the PLC program can be converted into text and vice versa using enums. The numerical values can also be assigned for several languages.

Creating enums

Enter an AWP command using the following syntax at the start of the HTML file:

```
<!-- AWP_Enum_Def Name="<Name of the enum type>"
Values='0:"<Text_1>", 1:"<Text_2>", ... , x:"<Text_x>"' -->
```

For example, for German values to be saved as an HTML file in the "de" folder of the HTML directory:

```
<!-- AWP_Enum_Def Name="Enum1" Values='0:"an", 1:"aus", 2:"Störung"' -->
```

For example, for English values, to be saved as an HTML file in the "en" folder of the HTML directory:

```
<!-- AWP_Enum_Def Name="Enum1" Values='0:"on", 1:"off", 2:"error"' -->
```

Assigning enums

Tags are assigned from the user program to the individual enum texts using a special AWP command:

```
<!-- AWP_Enum_Ref Name="<VarName>" Enum="<EnumTypeName>" -->
```

<VarName> is thereby the symbolic name from the user program and <EnumTypeName> is the previously set name of the enum type.

Note

In each fragment in which enum texts are referenced by a PLC tag, this PLC tag must be assigned by the appropriate AWP command of the enum type name.

Ensure that no AWP command for importing fragments is positioned between an enum assignment and enum usage because this import can result in the enum assignment lying in a different fragment than the enum usage.

Example

Enum type "state" is defined with values "0" and "1". "0" means "off", "1" means "on":

```
<!-- AWP_Enum_Def Name="state" Values='0:"off", 1:"on"' -->
```

The following code is contained in the HTML code of the web page to be output:

```
<!-- AWP_Enum_Ref Name="operating state" Enum="state" -->
:=operating state:
```

Depending on the value of the "operating state" tag, the ´result displayed is no longer "0" or "1", but "off" or "on".

Simplified use of enumeration types

At S7-1200 CPUs as of firmware version 4 it is possible to use enumerations directly in AWP commands to read and write PLC variables.

You create enums as described in the previous section, and you can then utilize the values with user program read and write commands.

Creating enums

```
<!-- AWP_Enum_Def Name="<Name des Enum Typs>" Values='0:"<Text_1>",
1:"<Text_2>", ... , x:"<Text_x>"' -->
```

Utilizing enums in the user program read and write commands

```
<!-- AWP_In_Variable Name='<Varname>' Enum="<EnumType>" -->
<!-- AWP_Out_Variable Name='<Varname>' Enum="<EnumType>" -->
```

Example of reading PLC tags

```
<!-- AWP_Enum_Def Name='AlarmEnum' Values='0:"No alarms", 1:"Tank is
full", 2:"Tank is empty"' --><!-- AWP_Out_Variable Name='"Alarm"'
Enum="AlarmEnum" -->...<p>The current value of "Alarm"
is :="Alarm":</p>
```

If the value of "Alarm" in the CPU is "2", the following text will be displayed on the HTML page:

'The current value of "Alarm" is Tank is empty' because the enum definition assigns the string "Tank is empty" to the numerical value 2.

Example of writing PLC tags

```
<!-- AWP_Enum_Def Name='AlarmEnum' Values='0:"No alarms", 1:"Tank is
full", 2:"Tank is empty"' --><!-- AWP_In_Variable Name='"Alarm"'
Enum='AlarmEnum' -->...
<form method="POST">
<p><input type="hidden" name='"Alarm"' value="Tank is full" /></p>
<p><input type="submit" value='Set Tank is full' /><p>
</form>
```

Because the enum definition assigns the string "Tank is full" to the numerical value "1", the value "1" is written to the PLC tag "Alarm".

Definition of fragments

Fragments

Fragments are "logical sections" of a web page to be processed by the CPU individually.

Fragments are usually complete pages but can also be individual elements such as files (for example, images) or complete documents.

Defining fragments

```
<!-- AWP_Start_Fragment Name="<Name>" [Type="<Type>"] [ID="<Id>"]
[Mode=<Mode>]-->
```

The start of a fragment is specified by this command. A fragment runs to the start of the next fragment or to the end of the file.

- <Name> Indicates the name of the fragment.
  The name must start with a letter [a-zA-Z] or an underscore (_). Letters, underscores or numbers [0-9] can follow after this first character.
- <Type> Indicates the type of the fragment.
  - "manual" The user program is informed of the request for a fragment; the web page to be returned can be changed by the user program.
  - "automatic" The page is automatically processed (default).
- <id> A numeric ID can be stipulated for the fragment. If no ID is assigned, the fragment is automatically assigned an ID. For manual pages (<Type>=manual) , the fragment can be addressed in the user program of the CPU by this ID.

  Note

  Keep the ID low because the highest ID influences the size of the Web Control DB.

- <Mode> Fragments support the visible and hidden modes.
  - "visible" The fragment is a part of the web page. This mode is preset and can also be omitted.
  - "hidden" The fragment is not part of the web page. However, the fragment will be saved in the Web DB and is available to the user program for inserting in a requested web page. You use an exchange of the fragment ID (Web-Control-DB.fragment_index tag) to insert a "hidden" fragment in the requested web page.

The input document is completely divided into fragments by the "AWP_Start_Fragment" command. "AWP_End_Fragment" is therefore unnecessary.

Without a start fragment command, a file is mapped as a fragment; the fragment name is derived from the file name. If a file is divided into several fragments (by "AWP_Start_Fragment"), the file must begin with the "AWP_Start_Fragment" command.

Importing fragments

You can declare a fragment in an HTML page and import this fragment into other web pages.

Example

A company logo is to be displayed on all web pages of a web application.

There is only one instance of the HTML code for the fragment that displays the company logo. You can import the fragment as often and into as many HTML files as required.

Syntax

```
<!-- AWP_Import_Fragment Name = "<name>"-->
```

- <name> is the name of the fragment to be imported.

Example

HTML code within a web page that declares a fragment:

```
<!-- AWP_Start_Fragment Name = "My_Company_Logo"-->
<p><img src = "compay_logo.jpg"></p>
```

Example

HTML code within another web page that imports the declared fragment:

```
<!-- AWP_Import_Fragment Name = "My_Company_Logo"-->
```

Creating and loading a data block

Requirement

- All source files required for the web application (*.html, *.js, *.png, ...) have been created.
- The source files are located in one folder, but only those source files that are required for the web application. No other files may be located in this folder.

Note

Length of file names and tag names

If you have a comprehensive web application with many files and directories, the generation of the web data blocks may possibly fail. If this happens, the generation is aborted with the message "Text list overflow...". The cause is system-internal size limitations for management information saved in the web data block.

Remedy: Use short file names and short tag names.

Procedure

To create data blocks from the source files for user-defined web pages in STEP 7, proceed as follows:

1. Select the CPU, for example, in the device configuration.
2. Select the properties for user-defined web pages in the inspector window under "Properties > General > Web server".
3. As "HTML source", select the folder that contains the source files for the web application.
4. Enter the HTMP page to be opened on starting the web application as the start HTML page.
5. Enter a name for the application if required.
6. You can supplement a range of file name extensions as "Files with dynamic content" if necessary. Only enter those file name extensions that also contain AWP commands.
7. The number for the Web Control DB and for the fragment DB start number can be kept as long as they are not already being used by your user program.
8. Click on the "Generate" button to create DBs from the source files.
   The generated data blocks are saved in the project navigation in the "System block" folder (in the "Web server" subfolder).
9. In the CPU, select the network view to be loaded and then select the "Download to device" command in the "Online" menu to download the blocks. Compilation of the blocks is implicitly initiated before downloading.
   If errors are reported during this process, you must correct these errors before you can download the configuration.

Structure of the PLC program

Your user program must call the " WWW" instruction to even allow the web application, for example, the user-defined web pages, to be available to the CPU on the standard web pages and to allow them to be called up there.

The Web Control DB you have created from the source files is the input parameter (CTRL_DB) for the "WWW" instruction. The Web Control DB references the content of the user-defined web pages coded in the fragment DB and then receives status and control information.

Calling the "WWW" instruction in the startup program

If you do not want the user program to influence requested web pages, it is sufficient to only call the "WWW" instruction once in a startup OB. This instruction initializes communication between the web server and the CPU.

Calling the "WWW" instruction in the cyclic program

The "WWW" instruction can also be called in an OB processed in cycles (for example, OB 1). This has the advantage of being able to respond to web server requests from within the user program. Manual fragments must be used for this.

In this case, you must evaluate information from the Web Control DB in order to identify the requested web page or the requested fragment. On the other hand, you must set a bit in the

user program in order to explicitly release the web page to be returned by the web server after processing the web page request.

The structure of the Web Control DB is described in the following section.

Web Control DB

The Web Control DB (DB 333 by default) is created by STEP 7 and contains information on the structure of user pages, the status of communication and any errors that occur.

Additional fragment DBs are also created as well as the Web Control DB. These fragment DBs (there may also only be one fragment DB) are referenced in the Web Control DB. The fragment DBs contain the web pages and media data coded in fragments, for example, images. The content of the fragment DB cannot be changed by the user program. It is created automatically and is only for data management.

The status and control tags of the Web Control DB are accessed via symbols.

The following lists the tags of the Web Control DB required for status evaluation and to control interaction.

The Web Control DB provides two types of information:

- Global status information: Not bound to a concrete web page request.
- Request status and control information: Information about queued requests.

Global status information

"WEB-Control_DB".commandstate.init	Activates and initializes the web application.
"WEB-Control_DB".commandstate.deactivate	Deactivates the web application.
"WEB-Control_DB".commandstate.inititializing	The web application is initialized (read Web Control DB, etc.).
"WEB-Control_DB".commandstate.error	Web application could not be initialized. The reason is coded in "WEB-Control_DB".commandstate.last_error .
"WEB-Control_DB".commandstate.deactivating	The web application is closed.
"WEB-Control_DB".commandstate.initialized	The web application has been initialized and is ready.
"WEB-Control_DB".commandstate.last_error	Refer to the next table for a value table of possible errors.

Last_error	Description
1	Fragment DB is inconsistent (does not match the Web Control DB).
2	A web application already exists with this name.
3	Memory problem initializing in the web server.
4	Inconsistent data in the Web Control DB.
5	A fragment DB is not available (not loaded).
6	No AWP ID for a fragment DB.

Last_error	Description
7	The enum fragment is not available (contains the texts and information on the enum types).
8	An action requested via the command flag in the Web Control DB is prohibited in the current state.
9	Web application is not initialized (if there is no reinitializing after disabling).
10	Web server is disabled.
...	Last_error is reset once the web application has been successfully initialized.

Request status information

Request status information is bound to one of four possible requests, x = [1 … 4].

"WEB-Control_DB".requesttab[x].idle	Nothing need be done.
"WEB-Control_DB".requesttab[x].waiting	The user program must react to a request from a manual fragment and explicitly initiate further processing in the web browser.
"WEB-Control_DB".requesttab[x].sending	The web server is occupied with processing the request/fragment.
"WEB-Control_DB".requesttab[x].aborting	The TCP connection is closed by the web server.

Request control information

Request control information is bound to one of four possible requests, x = [1 … 4].

"WEB-Control_DB".requesttab[x].continue	Releases the fragment being processed for transmission. Processing of the next fragment is initiated.
"WEB-Control_DB".requesttab[x].repeat	Releases the fragment being processed for transmission. The fragment is then processed again.
"WEB-Control_DB".requesttab[x].abort	Closes the TCP connection.
"WEB-Control_DB".requesttab[x].finish	Releases the fragment being processed for transmission. Stops further processing of requests (terminates the request).

Example:

The tag for the DB is: "WEB-Control_DB". Whether errors have occurred during initialization of the web application can be determined by requesting bit "WEB-Control_DB".commandstate.error in the user program.

If an error has occurred you can analyze it using the "WEB-Control_DB".commandstate.last_error value.

Interaction with the user program

Using manual fragments ensures that the user program reacts synchronously to the user program, thereby allowing the responding web page to be processed by the user program.

Fragment type

To react to the received data in the user program the "manual" fragment type must be used for the fragment writing the data (for "manual pages"):

```
<!-- AWP_Start_Fragment Name="testfrag" ID="1" Type="manual" -->
```

The values are always transferred to the web server of the CPU for automatic and manual pages in the same way:

Example:

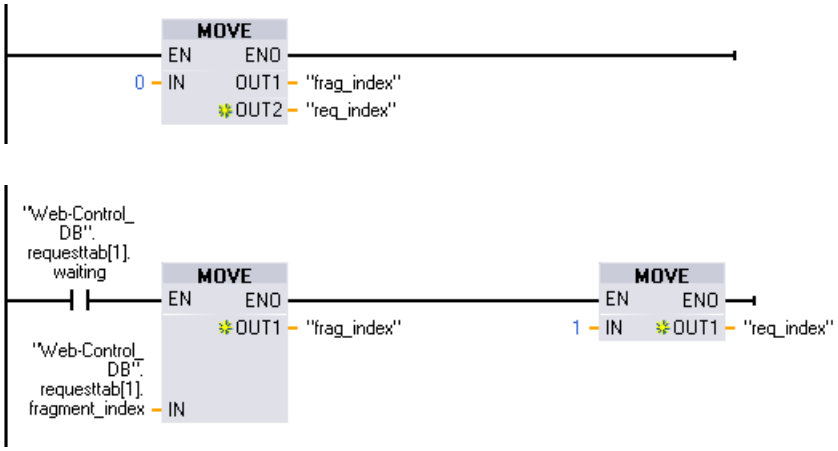
```
<form method="POST" action="">
<p>
<input type="submit" value="Set new value">
<input type="text" name='"Velocity"' size="20">
</p>
</form>
```

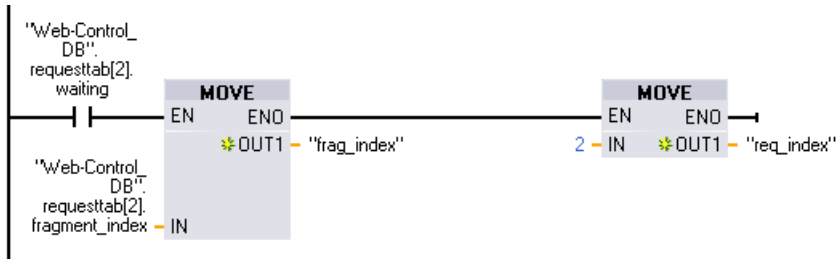
User program for manual fragments

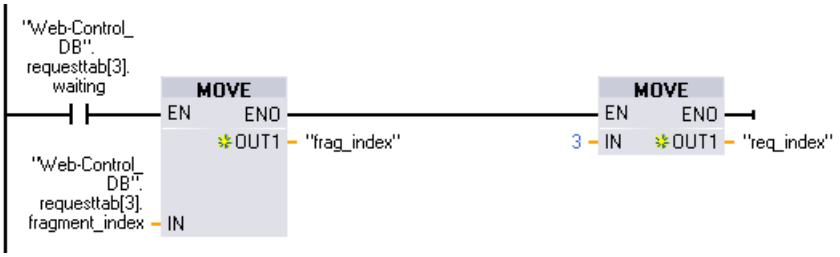
When using manual pages, the "WWW" instruction must be called in cycles in the user program of the CPU.

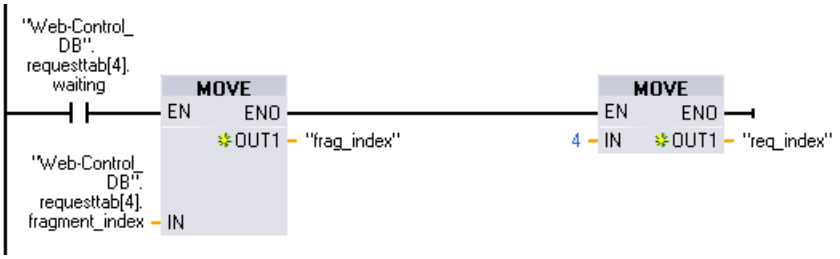
To react to values entered in the browser, the request - which is made by the manual page to the web server - must first be evaluated in the user program. To do this, the Web Control DB (for example, DB 333) must be examined in cycles for queued requests. The array that manages four requests is contained in the "requesttab" section of the Web Control DB. Each element of the array thereby contains information on the respective request within a structure.

A simple programming example shows how queued requests are checked based on the tags of the Web Control DB.







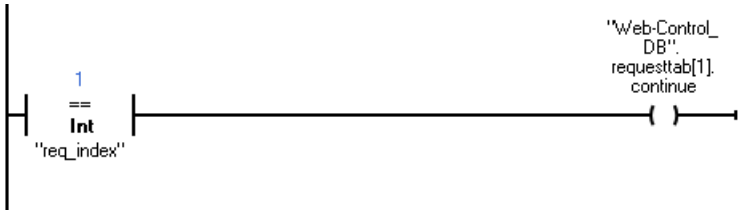


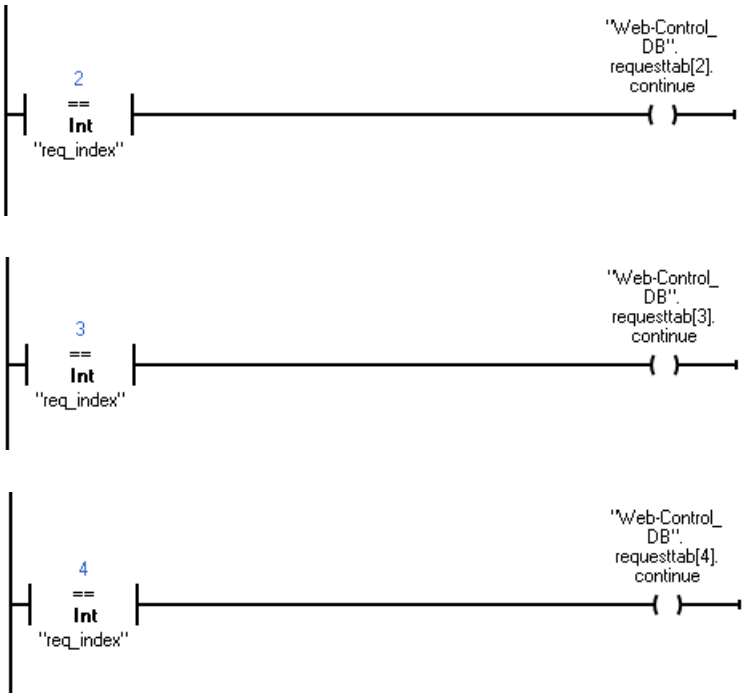
In cases where a request has been made, this program section writes the fragment ID in the #frag_index tag and the request no. (value range 1-4) in the #req_index tag.

Using the information from this, the information transferred in the request can now be processed separately for each fragment ID in the program (for example, plausibility check).

Once processing of the request has been completed by the program, the request must be answered and the appropriate entry is once more reset under"requesttab" of the Web Control DB (for example, DB 333).

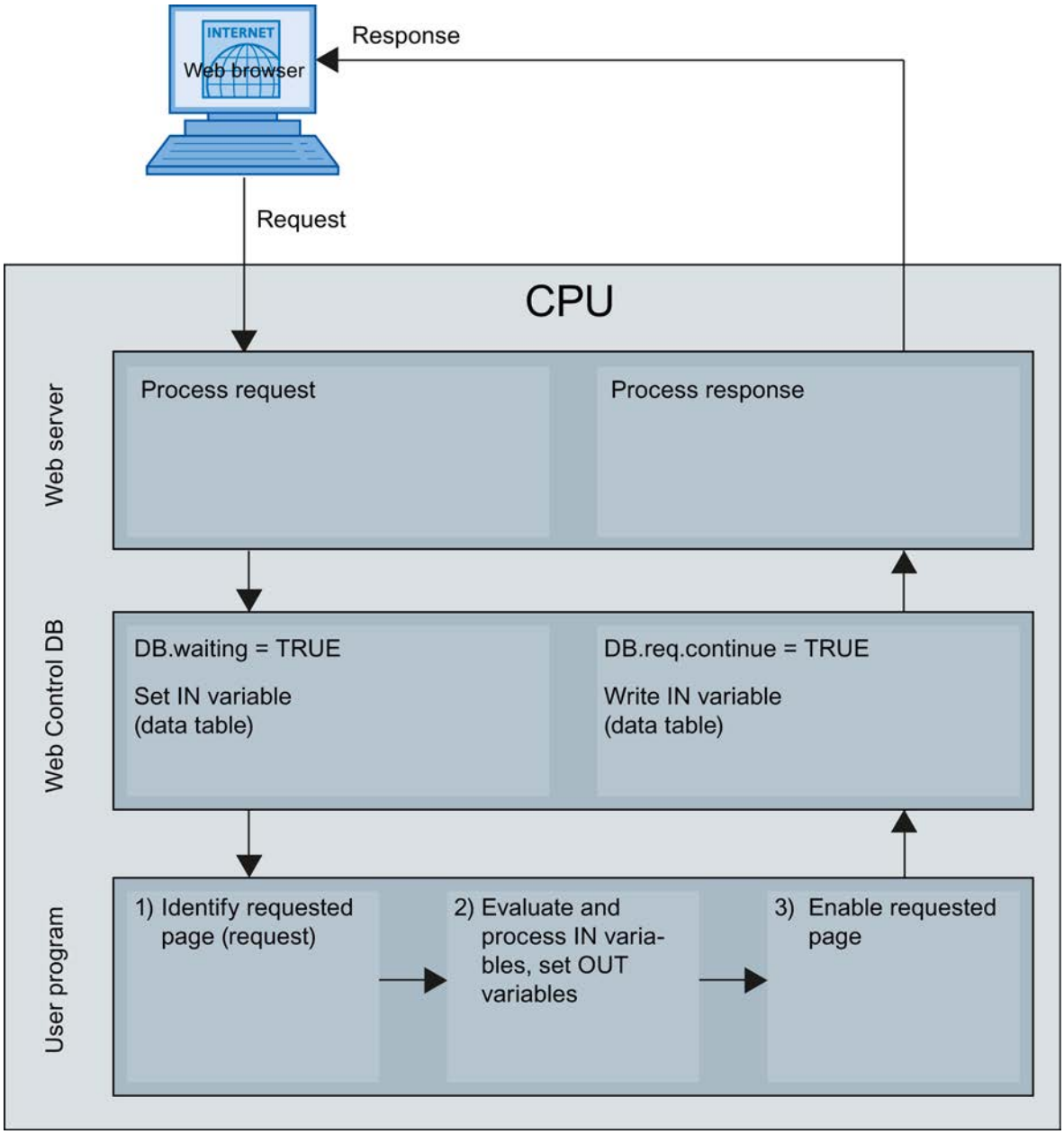
A simple programming example for replying to requests:




2
==
Int
"req_index"
"Web-Control_
DB".
requesttab[2].
continue
3
==
Int
"req_index"
"Web-Control_
DB".
requesttab[3].
continue
4
==
Int
"req_index"
"Web-Control_
DB".
requesttab[4].
continue

Principle sequence of a browser request with interaction from the user program

The following figure shows the simplified, principle sequence of the web browser request on the effects of Web Control DB content and the actions required from the user program until the processed web page is returned (response).



Displaying custom web pages in the browser

Display web pages in browser

Web pages are called from the standard web pages of the web browser.

In addition to the other links in the navigation bar, the standard web pages also have a link to "user pages".

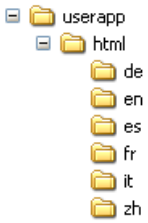
Click on the "user pages" link to open the web browser you have configured as the default HTML page.

Creating custom web pages in several languages

You can make each of your custom web pages available in various languages.

Requirements

The language-dependent HTML; pages must be stored in a folder structure containing folders with the respective language abbreviations:



Specified language abbreviations

Language abbreviations "de", "en", "fr", "es", "it" and "zh" are fixed. Additional language folders or other designated language folders are not supported.

Additional folders within the same folder hierarchy for other files can be created as required; for example, an "img" folder for images and a "script" folder for JavaScript files.

Language switching for custom web pages

Requirements

The HTML pages are contained in the predefined language folders, for example, HTML pages with German text are in the "de" folder, HTML pages with English text are in the "en" folder.

Language switching concept

Language switching is based on a predefined cookie named "siemens_automation_language". If the cookie is set to value "de", at the next web page request or web page update, the web server switches to the web page from the "de" folder.

Similarly, the web server switches to the web page from the "en" folder when the cookie is set to "en".

Example of language switching

The example is structured as follows:

- The language-dependent HTML files with the same name, for example, "langswitch.html" are located in both language folders "de" and "en". The text to be displayed within the two files are German or English, corresponding to the name of the folder.
- There is an additional "script" folder in the folder structure containing the JavaScript file "lang.js". Functions required for language switching are stored in this file .

Structure of the "langswitch.html" file ("de" folder)

Meta data "content language", charset and path to JavaScript file are set in the file header.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Language" content="de">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Switch language to German page</title>
<script type="text/javascript" src="script/lang.js" ></script>
<head>
```

Language selection is implemented in the body of the file by the "select" HTML element. The select element initiates a list box and contains the "de" option, labeled as "German" and "en", labeled as "English"; "de" is the default.

The "DoLocalLanguageChange(this)" function is called using the "onchange" event handler. The "this" parameter transmits the select object with the selected option to this function. "onchange" calls the function each time the option is changed.

```
<!-- Language Selection -->
<table>
    <tr>
        <td align="right" valign="top" nowrap>
         <!-- change language immediately on change of the selection
-->
          <select name="Language"
onchange="DoLocalLanguageChange(this)" size="1">
                <option value="de" selected >Deutsch</option>
                <option value="en" >English</option>
           </select>
        </td>
     </tr>
</table>
<!-- Language Selection End-->
```

Structure of the "langswitch.html" file ("en" folder)

The header of the HTML file with English text is structured similarly to the HTML file with German text.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Language" content="en">
```

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Language switching english page</title>
<script type="text/javascript" src="script/lang.js" ></script>
```

Language selection is also implemented in the body of the file by the "select" HTML element. In contrast to the German HTML file, the English option is already selected as a default and the test or the labels are in English.

```
<!-- Language Selection -->
<table>
    <tr>
        <td align="right" valign="top" nowrap>
         <!-- change language immediately on change of the selection
-->
          <select name="Language"
onchange="DoLocalLanguageChange(this)" size="1">
                <option value="de" >German</option>
                <option value="en" selected >English</option>
            </select>
        </td>
     </tr>
</table>
<!-- Language Selection End-->
```

Structure of "lang.js" file (in the "script" folder)

The " DoLocalLanguageChange" function is defined in the Java script file and calls the "SetLangCookie" function with the language selection value. SetLangCookie combines the cookie name and cookie value and then sets the cookie by means of the corresponding document.cookie property. The web page must then be reloaded (top.window.location.reload) to allow the web server to react to the setting of the cookie by displaying the required language.

```
function DoLocalLanguageChange(oSelect) {
        SetLangCookie(oSelect.value);
        top.window.location.reload();
    }

 function SetLangCookie(value) {
        var strval = "siemens_automation_language=";
            // this is the cookie by which the web server
            // detects the desired language
            // this name is required by the web server
        strval = strval + value;
        strval = strval + "; path=/ ;";
            // set path to the application, since otherwise
            // path would be set to the requesting page
            // would not get the cookie.
            // The path for user defined applications follows this
sample:
            // path=/awp/<application name>/<pagename>
            // example: path=/awp/myapp/myappstartpage.htm
            //(where myapp is the name of the web application
            // entered in the web server properties of the cpu)
```

```
                /*
                use expiration if this cookie should live longer
                than the current browser session
                var now      = new Date();
                var endttime = new Date(now.getTime() + expiration);
                strval = strval + "; expires=" + endttime.toGMTString()
+ ";";
                */
            document.cookie = strval;
    }
```

Additional configurations

Configuring additional functions

The S7-1200 automation system has numerous additional functions that are useful as integrated CPU functions or available via plug-in modules (for example, communication modules). You can find the description via the following links.

See also

Overview of point-to-point communication (Page 1598)

General information on high-speed counters (Page 1594)

Configuring PID_Compact V1 (Page 9575)

Configuring PID_3Step V1 (Page 9607)

Motion functionality of the CPU S7-1200 (Page 10005)

8.1.4.5 Configurations for point-to-point connections (S7-1500)

Configuration for point-to-point links

Introduction

Overview of the communication modules

Automation systems encompass a wide range of components. These also include communication modules. A simple possibility of data exchange is provided by serial communication via point-to-point connections.

Customizing to a wide range of communication partners is possible by setting the communication parameters at a lower layer of the OSI layer model (see chapter Transmission security (Page 979)).

Communication through point-to-point connection with S7-1500, ET 200MP and ET 200SP takes place exclusively by means of communication modules (CM) with serial interfaces.

SIMATIC S7 offers a number of modules that provide the physical interface and fundamental protocol mechanisms for this purpose.

- RS232: An interface that can coordinate the communication between the partners through additional accompanying signals.
- RS422/RS485: An interface that allows longer lines through the use of differential voltages as transmission technology and also enables structures with more than 2 devices through a bus structure (RS485).

Instructions that carry out the coordination between the CPU and CM (Communication Module) are available to transfer data from the CPU to the respective modules. They inform the user program about a successful transfer or the receipt of new data (in systems without a SIMATIC CPU users have to program the function of these instructions themselves).

The function and use of the PtP communication modules is described in this function manual.

Overview of components and order numbers

Tabular overview of communication modules and their application suitability

Communication module	S7-1500	ET 200MP	ET 200SP	Order number
CM PtP RS232 BA [1]	X	X	-	6ES7540-1AD00-0AA0
CM PtP RS422/485 BA	X	X	-	6ES7540-1AB00-0AA0
CM PtP RS232 HF [2]	X	X	-	6ES7541-1AD00-0AB0
CM PtP RS422/485 HF	X	X	-	6ES7541-1AB00-0AB0
CM PtP (ET 200SP)	-	-	X	6ES7137-6AA00-0BA0

[1] BA = Basic

[2] HF = High Feature

Overview of components and interfaces

Tabular overview of communication modules and their functions.

Communication module	Interface	Protocols					Connection technology	
		Freeport	3964(R)	Modbus Master	Modbus Slave	USS-Master	D-Sub 9-pin	D-Sub 15-pin
CM PtP RS232 BA	RS232	X	X	-	-	X	X	-
CM PtP RS422/485 BA	RS422	X	X	-	-	X	-	X
	RS485	X	-	-	-	X	-	X
CM PtP RS232 HF	RS232	X	X	X	X	X	X	-
CM PtP RS422/485 HF	RS422	X	X	X	X	X	-	X
	RS485	X	-	X	X	X	-	X
CM PtP (ET 200SP)	RS232	X	X	X	X	X	ET 200SP BaseUnit [1]	

Communication module	Interface	Protocols					Connection technology
	RS422 2)	X	X	X	X	X	
	RS485	X	-	X	X	X	

1) BaseUnit with terminals instead of D-Sub; assignment depending on physical transmission properties

2) The CM PtP communication module can also be used for multipoint connection in RS422 mode

Overview of components and data transmission rates

The communication modules can send and receive data with different data transmission rates. The table below shows the assignment to the individual communication modules.

Communication module	Data transmission rate in bps										
	300	600	1200	2400	4800	9600	19200	38400	57600	76800	115200
CM PtP RS232 BA	X	X	X	X	X	X	X	-	-	-	-
CM PtP RS422/485 BA	X	X	X	X	X	X	X	-	-	-	-
CM PtP RS232 HF	X	X	X	X	X	X	X	X	X	X	X
CM PtP RS422/485 HF	X	X	X	X	X	X	X	X	X	X	X
CM PtP (ET 200SP)	X	X	X	X	X	X	X	X	X	X	X

Overview of components and receive buffer size

Each communication module has a buffer to temporarily store received frames. The table below shows the assignment of the maximum size of an individual frame as well as the size of the memory for the individual communication modules.

Module	Receive buffer size KB	Max. frame length KB	Bufferable frames
CM PtP RS232 BA	2	1	255
CM PtP RS422/485 BA	2	1	255
CM PtP RS232 HF	8	4	255
CM PtP RS422/485 HF	8	4	255
CM PtP (ET 200SP)	4	2	255

Accompanying signals and data flow control

- Software data flow control with XON/XOFF
  The Freeport protocol supports data flow control with XON/XOFF via the RS232 and RS422 interfaces.
- Hardware data flow control with RTS/CTS
  The Freeport protocol supports data flow control with RTS/CTS via the RS232 interface.
- Automatic operation of accompanying signals
  The RS232 accompanying signals can be controlled with the Freeport, Modbus master and Modbus slave protocols by means of the RS232 interface. (Only available if hardware data flow control is not activated.)

Protocols of the communication modules

You may set up a communication connection with different protocols, depending on the communication modules used:

- Freeport: Transmission of ASCII character strings without specified protocol format
- 3964(R): Communication between programmable logic controllers (master/master communication)
- USS: Communication between a programmable logic controller and a drive (master/slave communication) Communication is tailored to the drive technology requirements. The communication module can only be master.
- Modbus RTU: Communication between programmable logic controllers (master/slave communication) The communication module can be the master as well as the slave.

Overview of the processing steps

Point-to-point connection

The system provides various networking options for the exchange of data between two or more communication partners. The simplest form of data interchange is via a point-to-point connection between two communication partners.

The communication module (CM) forms the interface between a programmable logic controller and a communication partner. Data is sent in serial mode via point-to-point connection with the communication module.

Configuring / parameter assignment

Configuring the communication module includes the arrangement of the communication module in the device configuration of STEP 7 V12 as well as the settings of the specific protocol parameters in the properties dialog of the communication module (static configuration).

Programming

Programming includes the program-specific connection of the communication module to the corresponding CPU by means of the user program. You program the communication module with STEP 7 V12.

Communication between CPU, communication module and a communication partner takes place through instructions. A number of instructions are available for the S7-1500 automation system that you can use to initiate and control communication in the user program as well as influence the configuration for runtime (dynamic configuration).

For more information, please refer to Overview of instructions (Page 977) and the STEP 7 V12 online help.

Overview of instructions

Note

CPU quantity structures

The point-to-point instructions communicate with the communication modules by reading or writing data records.
When using the instructions, you should therefore check the quantity structure of the CPU for reading and writing data records.
If multiple instructions need to read or write data records simultaneously on a CPU, there may need to be a gap between the calls of each instruction by the user program.

Overview of the instructions

The communication protocols are implemented on the communication module. The protocol is used to adapt the interface of the communication module to the interface of the communication partner.

Communication between the CPU, the communication module and a communication partner takes place by means of special instructions and protocols that support the corresponding communication modules.

The instructions form the software interface between the CPU and the communication module. They must be called cyclically from the user program. Data transmission takes place across several cycles.

The instructions are part of STEP 7 V12. The instructions are available in the "Instructions" task card under Communication > Communication processor.

The corresponding instruction descriptions are available in the information system. They apply to all listed communication modules if they support the required function.

Table 8-59 Instructions for PtP

Instruction	Meaning
Port_Config (Page 5184)	The instruction Port_Config allows you to dynamically assign basic interface parameters.
Send_Config (Page 5187)	The instruction Send_Config (send configuration) allows you to dynamically assign serial send parameters of a port.
Receive_Config (Page 5188)	The instruction Receive_Config (receive configuration) allows you to dynamically assign serial receive parameters of a port.
P3964_Config (Page 5193)	The instruction P3964_Config (protocol configuration) allows you to dynamically configure the parameters of the 3964(R) procedure.
Send_P2P (Page 5194)	The instruction Send_P2P allows you to send data to a communication partner.
Receive_P2P (Page 5197)	The instruction Recveive_P2P allows you to receive data from a communication partner.
Receive_Reset (Page 5199)	The instruction Receive_Reset allows you to clear the receive buffer of the communication module.
Signal_Get (Page 5199)	The instruction Signal_Get allows you to read the RS232 accompanying signals.

Instruction	Meaning
Signal_Set (Page 5200)	The instruction Signal_Set allows you to set the RS232 accompanying signals.
Get_Features (Page 5201)	The instruction Get_Features allows you to read the extended functions supported by the communication module.
Set_Features (Page 5202)	The instruction Set_Features allows you to activate the extended functions supported by the communication module.

Table 8-60 Instructions for USS

Instruction	Meaning
USS_Port_Scan (Page 5220)	The instruction USS_Port_Scan allows you to communicate by means of the USS network.
USS_Drive_Control (Page 5222)	The instruction USS_Drive_Control allows you to swap data with the drive.
USS_Read_Param (Page 5225)	The instruction USS_Read_Param allows you to read parameters from the drive.
USS_Write_Param (Page 5226)	The instruction USS_Write_Param allows you to change parameters in the drive.

Table 8-61 Instructions for Modbus

Instruction	Meaning
Modbus_Comm_Load (Page 5246)	The instruction Modbus_Comm_Load allows you to configure the port of the communication module for Modbus RTU. .
Modbus_Master (Page 5249)	The instruction Modbus_Master allows you to communicate as Modbus master by means of the PtP port.
Modbus_Slave (Page 5254)	The instruction Modbus_Slave allows you to communicate as Modbus slave by means of the PtP port.

Basics of serial communication

Serial data transmission

During serial data transmission, the individual bits of a character of information to be transmitted are sent successively in a defined sequence.

Bidirectional data traffic - operating mode

In the context of bidirectional data traffic, we distinguish between two operating modes for the communication module:

- Half-duplex operation
  The data is exchanged between the communication partners in both directions alternately. In half-duplex operation one communication partner is sending and the other communication partner is receiving at the same time. In the process one line is alternately used for sending or receiving.
- Full-duplex operation
  The data is exchanged between one or more communication partners in both directions simultaneously, which means you can send and receive data at the same time. This process requires one line for sending and one line for receiving.

Asynchronous data transmission

Serial data transmission is performed asynchronously with communication modules. The so-called time base synchronism (a fixed timing code used in the transmission of a fixed character string) is only upheld during transmission of a character. Each character to be sent is preceded by a synchronization impulse, also called start bit. The length of the start-bit transmission determines the clock pulse. The end of character transmission is formed by one or two stop bits.

Declarations

In addition to start and stop bits, additional declarations must be made between the sending and receiving partners before serial transmission can take place. These include:

- Data transmission rate
- Frame start and end criteria (e.g., character delay time)
- Parity
- Number of data bits (7 or 8 bits/characters)
- Number of stop bits (1 or 2)

Transmission security

Transmission security plays an important role in the transmission of data and in the selection of the transmission procedure. Generally speaking, the more layers of the reference model are applied, the higher the transmission security.

Classification of existing protocols

The figure below illustrates how the protocols of the communication module fit into the reference model.

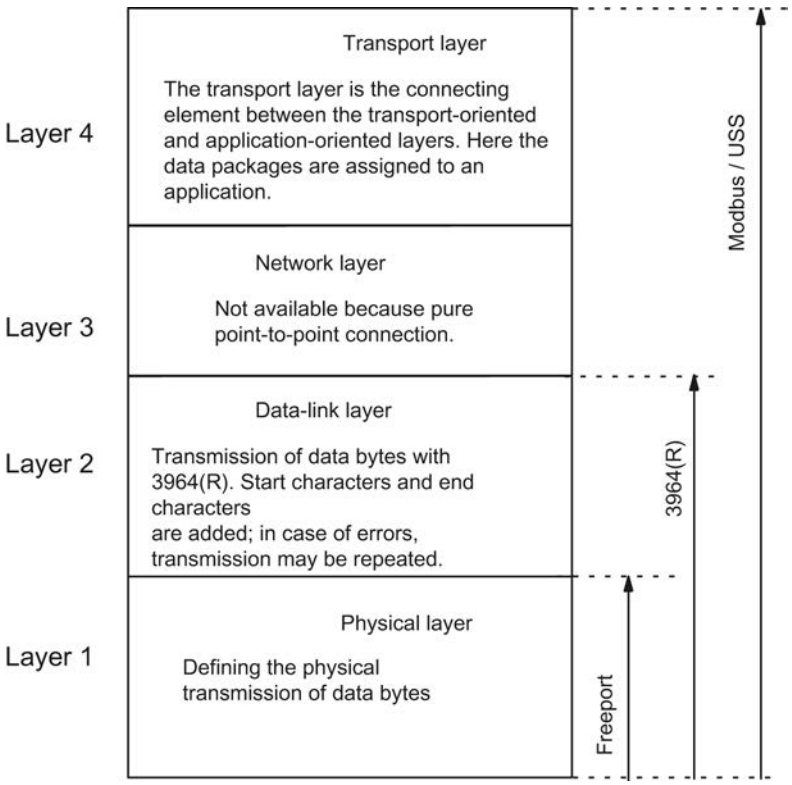


Figure 8-10 Classification of the existing protocols of the communication module in the reference model

Transmission security with Freeport

Transmission security when using Freeport:

- When data is sent with Freeport, there are no data security measures other than the use of a parity bit. This means data transmission with Freeport is very efficient, but data security is not guaranteed. A certain degree of data security can be achieved through parameter assignment of the frame start and frame end conditions.
- Using the parity bit ensures that the inversion of a bit in a character to be sent can be recognized. If two or more bits of a character are inverted, however, there is no guarantee that these errors are still detected.
- To increase transmission security, you can, for example, implement a checksum, a frame length specification, or configurable end conditions. These measures must be implemented by the user.
- A further increase in data security can be achieved by means of acknowledgment frames in response to send or receive frames. This is the case with high-grade protocols for data communication (ISO 7-layer reference model).

Transmission security with 3964(R)

The parity bit is used to increase data security; depending on the configuration, it completes the number of data bits to be transmitted to form an even or odd number.

Using the parity bit ensures that the inversion of a bit in a character to be sent can be recognized. If two or more bits of a character are inverted, however, these errors can no longer be reliably detected.

If parity is set to "none", no parity bit is transmitted. This setup reduces transmission security.

Two different procedures for data transmission can be used, either with or without a block check character:

- Data transmission without block check character: 3964
  Transmission security is achieved by means of a specified frame structure, frame breakdown, and frame repetitions.
- Data transmission with block check character: 3964R
  The high degree of transmission security is achieved by means of a specified frame structure and breakdown, frame repetitions, as well as inclusion of a block check character (BCC).

In this manual, the term 3964(R) is used when descriptions and notes refer to both data transmission modes.

Transmission integrity for Modbus and USS

The parity bit is used to increase transmission security; depending on the configuration, it completes the number of data bits to be transmitted to form an even or odd number.

Using the parity bit ensures that the inversion of a bit in a character to be sent can be recognized. If two or more bits of a character are inverted, however, this error can no longer be clearly detected.

If parity is set to "none", no parity bit is transmitted. This setup reduces transmission security.

The CRC (cyclic redundancy check) is additionally used with Modbus. With this method additional redundancy in the form of a so-called CRC value is added for each data block of the user data before data transmission. This is a check value calculated by using a specific procedure that can be used to detect any errors that may occur during transmission.

A BCC (block check character) is additionally used with USS. The block check character is formed during the receipt and is compared with the received BCC after the entire frame has been read in. If these two characters do not match, the frame is not evaluated. (If a character is incorrectly transmitted, an error is reliably detected. If an even number of characters are incorrectly transmitted, an error can no longer be reliably detected.)

RS232 mode

The following communication modules support RS232 mode:

- CM PtP RS232 BA
- CM PtP RS232 HF
- CM PtP (ET 200SP)

In RS232 mode, data is sent via two lines. A separate line is available for the send direction and the receive direction. Simultaneous sending and receiving is possible (full duplex).

RS232 signals

In addition to the TXD (interface transmits; Transmitted Data), RXD (interface receives; Received Data) and GND (Ground; isolated) signals, the communication module provides the following RS232 signals when using RS232 hardware:

DCD	(input)	Data Carrier detect; Carrier signal when connecting a modem. The communication partner signals that it recognizes incoming data.
DTR	(output)	Data terminal ready; DTR set to "ON": Communication module switched on, ready for operation DTR set to "OFF": Communication module not switched on, not ready for operation
DSR	(input)	Data set ready; DSR set to "ON": Communication partner signals readiness for operation DSR set to "OFF": Communication partner not switched on, not ready for operation
RTS	(output)	Request to send; RTS set to "ON": Communication module ready to send; signals to the communication partner that there is data ready to send RTS set to "OFF": Communication module not ready to send
CTS	(input)	Clear to send; Communication partner can receive data from the communication module (response to RTS = ON of the communication module) CTS set to "ON": Signals readiness to receive to the communication partner CTS set to "OFF": Signals "Not ready to receive" to the communication partner
RI	(input)	Ring Indicator; Incoming call when connecting a modem

After power on of the communication module, the output signals are in the OFF state (inactive).

You configure the operation of the DTR/DSR and RTS/CTS control signals in the user interface of the communication module.

The RS232 signals cannot be influenced in case of:

- configured data flow control "Hardware RTS always switched"
  (corresponds to automatic operation of the accompanying signals)
- configured data flow control "Hardware RTS always ON"
  (corresponds to hardware flow control with RTS/CTS)
- configured data flow control "Hardware RTS always ON, ignore DTR/DSR"

For more information on this topic, refer to chapter Handshake procedure (Page 984).

RS422 mode

The following communication modules support RS422 mode:

- CM PtP RS422/485 BA
- CM PtP RS422/485 HF
- CM PtP (ET 200SP)

In RS422 mode, data is transmitted via two line pairs (four-wire operation). A separate line pair is available for the send direction and the receive direction. Simultaneous sending and receiving is possible (full duplex).

All communication partners must be capable of simultaneous operation of a sender and receiver unit.

The data can be exchanged simultaneously between one or several communication partners. In RS422 multipoint mode, only one slave may send data at any given time.

Interface operating modes

The following table is a summary of the interface operating modes for the various communication modules and protocols.

The communication module can be used in the following topologies in RS422 mode:

- Link between two devices: Point-to-point connection
- Link between several devices: Multipoint connection
  (only available with CM PtP (ET 200SP))

Operating mode	Description
Full duplex (RS422) four-wire operation (point-to-point connection)	Both devices have the same priority in this operating mode.
Full duplex (RS422) four-wire operation (multipoint master)	The communication module can be used as multipoint master.
Full duplex (RS422) four-wire operation (multipoint slave)	The communication module can be used as multipoint slave.

The following applies for a multipoint master/slave topology in RS422 mode:

- The master's sender is interconnected with the receivers of all slaves.
- The slaves' senders are interconnected with the master's receiver.
- Only the receiver of the master and the receiver of one slave have a default setting. All other slaves operate without default settings.

RS485 mode

The following communication modules support RS485 mode:

- CM PtP RS422/485 BA
- CM PtP RS422/485 HF
- CM PtP (ET 200SP)

In RS485 mode, data is transmitted via one line pair (two-wire operation). The line pair is available alternately for the send and receive directions. It is possible to either send or receive (half duplex). On completion of a send operation, operation is immediately switched to receive mode (ready to receive). Send mode is reset again as soon as a new send job is received.

RS485 mode

The following table is a summary of the interface operating modes for the various communication modules and protocols.

Operating mode	Description
Half duplex (RS485) two-wire operation	Operating mode for point-to-point connection or multipoint connection (multipoint) in two-wire operation. The communication module can be the master as well as the slave.

If you operate the Freeport in RS485 mode (half duplex, two-wire operation), you must make provisions in the user program to ensure that only one device sends data at any given time. If more than one device sends data at the same time, the frames are corrupted.

Modbus automatically ensures that only one device is sending.

Changeover times for RS485 communication module in half duplex mode

A maximum time of 0.1 ms is set for the changeover between sending and receiving.

Handshake procedure

Introduction

Handshaking controls the data flow between two communication partners. The use of the handshaking method prevents data loss during transmission if the devices are operating at different speeds.

We can basically distinguish between the following methods:

Table 8-62 Overview of methods and interfaces

Method	RS232	RS422	RS485
Software data flow control XON/XOFF	X	X	-
Hardware data flow control (RTS/CTS)	X	-	-
Automatic operation of accompanying signals	X	-	-

Software data flow control

Software data flow control is implemented as follows on the communication module:

- XON/XOFF
  - As soon as the communication module has been set to the "XON/XOFF" operating mode by means of parameter assignment, it sends the XON character, thereby allowing the communication partner to send data.
  - On reaching the configured maximum number of frames, or 16 characters ahead of receive buffer overflow, the communication module sends the XOFF character, thereby requesting that the communication partners stop sending. If the communication partner nonetheless continues to send data, an error message is generated if the receive buffer overflows. Data received in the last frame is discarded.
  - As soon as a frame has been fetched by the CPU and the receive buffer is ready to receive data again, the communication module sends the XON character.
  - If the communication module receives the XOFF character during sending, it cancels the current send operation until it receives a XON again from its communication partner. If no XON is received within a specific configurable time, send operation is canceled and a corresponding error message is output.

Note

You can configure the characters for XON and XOFF (any ASCII character).

During parameter assignment of the XON/XOFF software data flow control, user data may not contain any of the configured XON or XOFF characters.

Hardware data flow control

Note

The DTR/DSR signals do not have to be wired for "Hardware RTS always ON, ignore DTR/DSR" parameter assignment.

If "Hardware RTS always ON" is configured, it is imperative that you fully wire the interface signals used. Make sure that the local RTS (out) is connected with the CTS (in) of the communication partner and the local CTS is connected with the RTS of the communication partner. Accordingly, the local DTR must be connected with the DSR of the communication partner and the local DSR with the DTR of the communication partner.

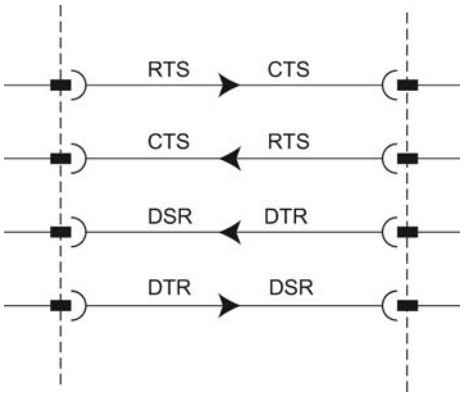


Figure 8-11 Wiring of the interface signals

- Hardware RTS always ON, ignore DTR/DSR
  - As soon as the communication module has been set to an operating mode with "Hardware RTS always ON" through parameter assignment, it outputs the RTS = ON signal to the communication partner to indicate its ready state.
  - RTS is set to OFF as soon as the configured maximum number of frames or 16 characters before buffer overflow is reached.
    If the communication partner nonetheless continues to send data, an error message is generated on overflow of the receive buffer. Data received in the last frame is discarded.
  - RTS is reset to ON as soon as the frame has been fetched by the CPU and the receive buffer is ready to receive data again.
  - If CTS switches to OFF during the send operation, the communication module interrupts the send operation until CTS is reset to ON. If CTS is not reset to ON within a specific configurable time, the send operation is canceled and a corresponding error message is output.
- Hardware RTS always ON
  The "Hardware RTS always ON" mode corresponds to the "Hardware RTS always ON, ignore DTR/DSR" mode. However, you also need to wire DTR and DSR.
  - As soon as the communication module has been set set to an operating mode with "Hardware RTS always ON" through parameter assignment, it sets DTR = ON and RTS = ON to signal its general ready state to the communication partner.
  - RTS is set to OFF as soon as the configured maximum number of frames or 16 characters before buffer overflow is reached.
    If the communication partner nonetheless continues to send data, an error message is generated on overflow of the receive buffer. Data received in the last frame is discarded.
  - RTS is reset to ON as soon as the frame has been fetched by the CPU and the receive buffer is ready to receive data again.
  - If CTS switches to OFF during the send operation, the communication module interrupts the send operation until CTS is reset to ON. If CTS is not reset to ON within a specific configurable time, the send operation is canceled and a corresponding error message is output.
  - A switch from DSR = ON to DSR = OFF cancels an active send job and triggers an error message.

Automatic operation of accompanying signals

- **Hardware RTS always switched**
  "Hardware RTS always switched" is implemented as follows on the communication module:
  - As soon as the communication module is set to the operating mode with "Hardware RTS always switched" through parameter assignment, it sets the line RTS to OFF and DTR to ON (communication module ready for operation).
    It is not possible to send frames until the DSR line is set to ON. No data is sent via the RS232C interface as long as DSR is set to OFF. A send job is canceled and a corresponding error message is generated.
  - When a send job is pending, RTS is set to ON and the configured RTS ON delay starts. On expiration of the data output time, the system checks whether the communication partner has set CTS to ON. If so, the data is sent via the RS232 interface.
  - If the CTS line is not set to ON within the RTS ON delay, or if CTS changes to OFF during transmission, the send job is aborted and an error message is generated.
  - Once the data has been sent and the configured clear RTS OFF delay has elapsed, the RTS line is set to OFF. The system does not wait for CTS to change to OFF.
  - It is always possible to receive data via the RS232 interface. There will be no reaction if there is a danger of the receive buffer of the communication module overflowing.
  - A switch from DSR = ON to DSR = OFF cancels an active send job and triggers an error message.

Note

Set the "RTS ON delay" in such a way that the communication partner is able to enter the ready to receive state before the time elapses.

Set the "RTS OFF delay" in such a way that the communication partner is able to receive the last characters of the frame completely before RTS is set to OFF and the send request is canceled.

Note

When automatic operation of the RS232 signals is configured, RTS and DTR cannot be controlled by means of the corresponding instruction!

Time diagram

The figure below shows the chronological sequence of a send job with configured data flow control "Hardware RTS always switched":

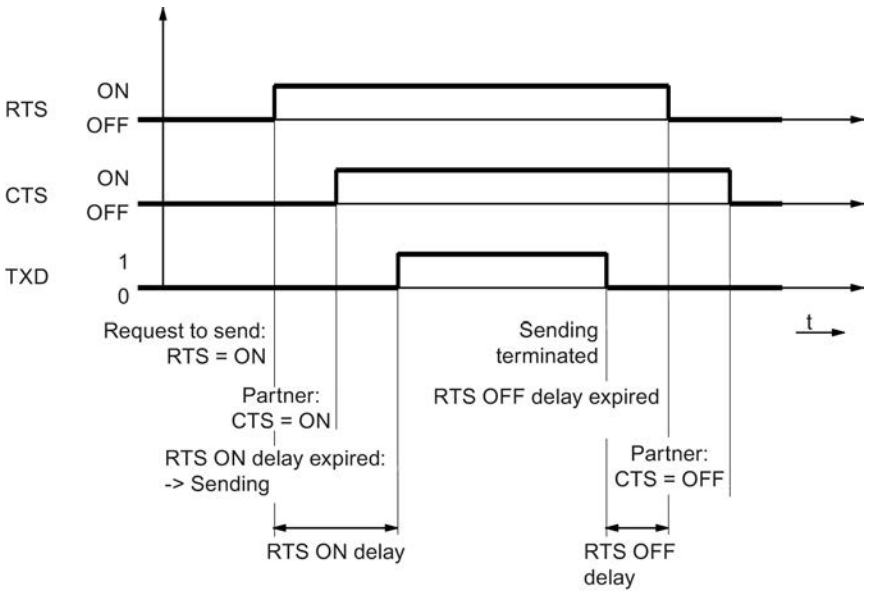


Figure 8-12 Time diagram for Hardware RTS always switched

Additional information

Note

Operation of DTR/DSR or RTS/CTS is accepted by the communication module with the following settings:

- Hardware RTS always ON, ignore DTR/DSR
- Hardware RTS always ON
- Hardware RTS always switched

Configuring / parameter assignment

Configuring / parameter assignment of a communication module

The following sections contain explanations on the following protocols and their parameters:

- Communication using Freeport (Page 989)
- Communication using 3964(R) (Page 995)
- Communication using USS (Page 999)
- Communication through Modbus RTU (Page 1001)

This information is required to carry out the parameter assignment and subsequently programming of the communication in accordance with the used protocol.

Configuration and parameter assignment are carried out in the device view of STEP 7 and in the properties dialog of the communication module. Some configurations can also be changed

during runtime by means of the corresponding "Config" instructions (Port_Config, Send_Config, Receive_Config, P3964_Config).

Procedure for setting up point-to-point communication

The procedure does not depend on the communication module used.

1. Configure an S7-1500 design with CPU and communication module in the device view of the STEP 7 V12 hardware editor.
2. You assign the parameters of the communication module interface (protocol, protocol parameters, addresses) in the "General" area of the "Properties" tab.

Additional information

Information about programming the communication is available in the section Programming - communication using instructions (Page 1006).

Communication using Freeport

Data transmission with Freeport

Introduction

Freeport is a freely programmable frame-based protocol that is also known as ASCII protocol.

The Freeport protocol controls data transmission by means of a point-to-point connection between the communication module and a communication partner. The Freeport protocol contains the physical layer (Layer 1).

The Freeport protocol supports sending and receiving of messages with any structure (all characters from 00H through FFH (for character frames with 8 data bits) or from 00H through 7FH (for character frames with 7 data bits)).

The frame start and end criteria must be configured both for the send and the receive direction. The start and end criteria can be configured differently.

Instructions are available for communication with a communication partner (see Programming - communication using instructions (Page 1006)).

Sending data with Freeport

Specifying settings for sending

To send a message, the partner must be informed of the start and end of the message. These settings can be set fixed in the hardware configuration or be adjusted during runtime by using the instruction Send_CFG. You can select one of the following options or also combine them:

- Send break before frame start
  You can specify that an additional Break is sent at the beginning of each message transmission on expiration of the RTS ON delay time.
  The duration of the "Break" is specified in bit times.
  Compliance with the send break can be deactivated if other mechanisms are used for synchronization.
- Send idle line
  You can specify that an additional "Idle Line" signal is output at the start of each message transmission.
  The duration of the "Idle Line" is specified in bit times.
  Compliance with the send break can be deactivated if other mechanisms are used for synchronization.
- RTS ON delay
  You can configure the time that has to expire after the RTS (Request to send) before the actual data transmission starts (RS232 only).
- RTS OFF delay
  You can configure the time that has to expire after transmission has been completed before the RTS signal is deactivated (RS232 only).
- Send up to and including the end delimiter
  You can configure the number of end delimiters (1 or 2) and their value.
  All data up to the end delimiter(s) is sent, independent of the selected frame length. The end delimiter must be included in the data to be sent. Data is sent only up to and including the delimiter, even if the specified data length is longer.
- Number of appended characters
  Input of the number of appended characters. Sending takes place up to the configured length. The end delimiter(s) is/are appended automatically. Depending on the number of end delimiters, one to five characters more than the number specified at the instruction are sent to the partner.

Note

If you combine "Send break before frame start", "Send idle line" and "RTS ON delay", these are processed in the order "RTS ON delay", "Send break before frame start" and "Send idle line".

Receiving data with Freeport

Specifying the message start

For data transmission with Freeport, you can choose between several different start criteria. The start criterion defines when a frame starts. Once a criterion that indicates the start of the message is met, the data stream is scanned for message end criteria. Here you select the settings that correspond to the properties of the sending communication partner.

Two different methods are available for detecting the message start:

- Start on any character
  Any character can be used to define the start of the message (default).
  This means that the first character sent at the start of communication, or after the frame end has been detected, will be identified as the first character of a message.
- Start on special condition
  The start of the message is detected based on the following specified conditions.
  - After detection of a line break
    The frame start is not accepted unless a break has been received beforehand, in other words, it is compulsory for the partner to send a break before sending a frame.
  - After detection of an idle line
    The frame start is not accepted until the configured idle line duration has expired. This procedure requires a minimum interval between two frames.
  - After receipt of a start character
    The frame start is detected when the configured start character is identified.
  - After detection of one or several start sequences
    The frame start is detected when the configured string with a length of up to five characters is identified. You can configure up to 4 start sequences. The start sequences that are up to 5 characters long can also contain "don't care characters".

Example:

Table 8-63 Configured start conditions

Start condition	1. character	2. character	3. character	4. character	5. character
1	0x68	xx	xx	0x68	xx
2	0x10	0xaa	xx	xx	xx
3	0xdc	0xaa	xx	xx	xx
4	0xe5	xx	xx	xx	xx
:					

The following message has been received: 68 10 aa 68 bb 10 aa 16

The evaluation of the start criteria begins with the receipt of the first character 0x68.
The 2nd and 3rd characters are free.
When the 4th character (second 0x68) is received, the first start condition is met and further evaluation of the message begins.

Specifying the message end

You can choose from several different end criteria for data transmission using the Freeport protocol. The end criterion defines the point at which a frame has been received completely.

Configurable end criteria are:

- Recognize message end by message timeout
- Recognize message end by response timeout
- After character delay time elapses (default)
- After receipt of a fixed frame length
- After receipt of a maximum number of characters
- Read message length from message
- After receipt of an end sequence

Message timeout

When data is received, the end of frame is detected on expiration of the configured time for transferring a frame. Time measurement starts after the start criterion has been met.

Response timeout

The response time is used to monitor the response behavior of the communication partner. If a valid frame start is not recognized after the completion of a send job, the send job is acknowledged with a corresponding message.

The actual end criterion has to be configured additionally.

Expiration of character delay time

When data is received, the frame end is detected when the configured maximum time between successive characters is exceeded (character delay time). The value is specified in bit times.

In this case, the character delay time must be set in such a way as to ensure that it expires between two consecutive frames. However, it should be of sufficient length to exclude incorrect identification of the end of the frame whenever the communication partner performs a transmission pause within a frame.

Fixed frame length

When data is received, the end of the frame is identified after the configured frame length has been reached.

An error message is output and the frame is discarded if the character delay time expires (if activated) before the fixed frame length has been reached.

Please note the following if the frame length of the received characters does not match the fixed configured frame length:

- All characters received after the fixed configured frame length has been reached will be discarded until a new start criterion is detected.
- An error message is output and the frame is discarded if another (activated) end criterion is met before the fixed frame length has been reached.

Maximum number of characters

When receiving data, the end of the frame is recognized after the declared number of characters have arrived.

This setting can be combined with the "Character delay time" settings. The frame received is also assessed as free of error if another end condition occurs, regardless of whether the maximum number of characters has been reached.

Please note the following if the frame length of the received characters does not match the configured maximum frame length:

- All characters received after the configured maximum number of characters has been reached will be discarded until a new start criterion (e.g. "Idle Line") is detected.
- If a different (activated) end criterion is met before the configured maximum number of characters has been reached, this "frame part" is assessed as a valid frame and the partner waits for a new start criterion. All characters received prior to fulfillment of a new start criterion are discarded.

Note

If no further end criterion is activated, the fixed frame length and maximum number of characters will respond in the same way.

Message length in the message

When data is received, the frame end is detected when the frame length sent with the received frame has been reached.

The following parameters define the characters to be used for evaluation of the message length:

- **Offset of length field in message**
  In the message, the value defines the position of the character that is to be used to determine the message length.
  You can set values from 0 to 4095 characters, depending on the buffer size.
- **Size of length field**
  This value specifies the number of characters as of the first evaluation position to be used to determine the message length.
  You can set values of 1, 2 and 4 characters.
- **Number of characters not counted in length specification**
  Number of characters appended to the frame without counting towards the frame length. This value defines the number of bytes at the end of the frame which should not be included in the evaluation of the message length.
  You can set values from 0 to 255 characters.

Example:

Parameter assignments for "Message length in the message"

Offset of length field in message:	3. byte ("2" has to be configured as offset)
Size of length field:	1 byte
Number of characters not counted in length specification:	3 bytes

Message					Number of characters not counted in length specification		
Start character	Address	Length field			Checksum	End delimiter	
Byte 1	Byte 2	Byte 3	Byte ...	Byte X	Byte X+1	Byte X+2	Byte X+3

End sequence

When data is received, the end of the frame is identified when the configured end sequence (max. 5 characters) is received. The end sequence which is up to 5 characters long can also contain "don't care characters". The received data is applied by the CPU, including the end sequence.

If you are working with the end sequence, transmission is not code-transparent and you must exclude the presence of end code in the user data.

Note

Frame end sequence

If there is only one end delimiter, the entry **must** take place in the 5th line.

If there are two end delimiters, the entries **must** take place in the 4th and 5th line (no gaps).

The same applies to the use of additional characters.

Code transparency

Code transparency

Code-transparent means that any character combinations can occur in the user data without the end criterion being recognized.

The code transparency of the procedure depends on the selection of the configured end criterion and flow control:

- With specified end sequence or using XON/XOFF flow control
  - Not code-transparent
- End criterion character delay time, fixed frame length, maximum frame length, message timeout, or response timeout and message length in the message
  - Code-transparent

Receive buffer

Receive buffer of the module

The communication modules have a receive buffer that stores the received frames temporarily until they are transmitted to the CPU. The receive buffer is implemented as a ring buffer, which means the frames are transmitted to the CPU in the order in which they were received until the receive buffer is full. If additional frames are received once the buffer is full, the oldest frame is overwritten. If "Prevent overwriting" was configured, a corresponding message is generated when the receive buffer is full. All further frames are rejected until the receive buffer is ready to receive new ones.

During the parameter assignment, you can specify whether the receive buffer should be deleted during startup. You can also specify the range of values (1 to 255) for the number of buffered receive frames.

The receive buffer of the module may have a size of up to 8 KB, depending on the communication module used (see chapter Introduction (Page 973)). The frame has a maximum length of 4 KB. This means that each communication module is capable of buffering at least two frames.

If you always want to transfer the last frame received to the CPU, you must set the value "1" for the number of buffered frames and deactivate overwrite protection.

Note

If continuous reading of the received data in the user program is interrupted for a certain time, you may find when the receive data is requested again, that the communication module first sends older frames before the CPU receives the most recent one. At the time of interruption, the old frame had already been transmitted from the receive buffer of the communication module and prepared for transmission to the CPU.

Communication using 3964(R)

Data transmission with 3964(R) procedure

Introduction

The 3964(R) procedure controls point-to-point data exchange between the communication module and a communication partner. and contains both the physical layer (layer 1) and the link layer (layer 2).

Instructions are available for communication with a communication partner (see Programming - communication using instructions (Page 1006)).

Control characters

Introduction

During data transmission, the 3964(R) procedure adds control characters to the information data (link layer). The communication partner can use these control characters to check whether it has received all data completely and without errors.

Control characters of the 3964(R) procedure

The 3964(R) procedure evaluates the following control characters:

STX	Start of Text	Beginning of the character string to be transmitted	02H
DLE	Data Link Escape	Data transmission changeover	10H
ETX	End of Text	End of the character string to be transmitted	03H
NAK	Negative Acknowledge	Negative acknowledgment	15H
BCC	Block Check Character (only with 3964R)	Block check character	

BCC is formed and monitored automatically in the communication module. The block check character is not transmitted as frame content to the CPU.

Note

If the DLE character is transmitted as an information character within a frame, it is sent twice (DLE duplication) to distinguish it from the DLE control character during connection establishment and termination. The receiver reverses the DLE duplication.

Priority

With the 3964(R) procedure, one communication partner must be assigned a higher and the other a lower priority. If both partners start to establish a connection at the same time, the partner having lower priority will cancel its send job.

Block check character

Block check character

With the 3964R transfer protocol, data security is enhanced by sending an additional block check character (BCC = Block Check Character).

The block check character is the even longitudinal parity (EXOR logic operation of all data bytes) of a sent or received block. Its calculation begins with the first byte of user data (first

byte of the frame) after the connection establishment, and ends after the DLE ETX character at connection termination.

Note

With DLE duplication, the DLE character is included twice in the BCC calculation.

Sending data with 3964(R)

Connection establishment for sending

The 3964(R) procedure sends the STX control character to set up the connection. If the communication partner responds with the DLE character before the acknowledgment delay time expires, the procedure switches to send mode.

If the communication partner answers with NAK or any other character (except for DLE or STX), or the acknowledgment delay time expires without a response, the procedure tries to set up the connection again. After the configured number of unsuccessful setup attempts, the procedure cancels the connection setup and sends the NAK character to the communication partner. The communication module outputs a corresponding error message.

Sending data

If the connection is successfully established, the user data contained in the output buffer of the communication module is sent to the communication partner with the selected transmission parameters (a DLE recognized in the user data is doubled during the send job). The communication partner monitors the time intervals between the incoming characters. The interval between two characters must not exceed the character delay time. Monitoring of the character delay time starts immediately after the connection has been established.

If the communication partner sends the NAK character during an active send operation, the procedure cancels the block and repeats it as described above, beginning with connection establishment. If a different character is sent, the procedure first waits for the character delay time to expire and then sends the NAK character to set the communication partner to idle state. Then, the procedure restarts sending with the connection setup STX.

Connection termination during sending

Once the contents of the buffer have been sent, the procedure appends the DLE and ETX characters and (only with 3964R) the block checksum BCC as the end identifier, and then waits for an acknowledgment character. If the communication partner sends the DLE characters within the acknowledgment delay time, the data block has been received without errors. If the communication partner responds with NAK, any other character (except DLE), or with a corrupted character, or if the acknowledgment delay time expires without a response, the procedure restarts sending with the connection setup STX.

After the configured number of attempts to send, the procedure stops the process and sends an NAK to the communication partner. The communication module outputs a corresponding error message.

Receiving data with 3964(R)

Connection setup for receiving

In idle state, when there is no send job to be processed, the procedure waits for the communication partner to set up the connection.

A wait time is started (wait time = acknowledgment delay time - 10 ms, however, maximum of 400 ms) if no free receive buffer is available during the connection setup with STX. An error message is generated if no free receive buffer is available on expiration of this time. The procedure sends the NAK character and returns to the idle state. Otherwise, the procedure sends a DLE and receives the data as described above.

The acknowledgment delay time should be set to the same value at both communication partners.

If the procedure receives any character (except for STX or NAK) while in idle state, it waits for the character delay time (CDT) to expire and then sends the NAK character. The communication module outputs a corresponding error message.

Receiving data

After a successful connection establishment, the incoming receive characters are saved to the receive buffer. If two consecutive DLE characters are received, only one of these is saved to the receive buffer.

After connection has been established and after each receive character, the procedure waits for the next character during the character delay time. If this period expires before another character is received, an NAK is sent to the communication partner. The communication module outputs a corresponding error message. A retry is then expected.

If transmission errors occur during receiving (frame errors, parity errors, etc.), the procedure continues to receive data until the connection is terminated and then sends an NAK to the communication partner. A retry is then expected. If the block still cannot be received without errors after the specified number of transfer attempts, or if the communication partner does not start the retry within a block wait time of 4 seconds, the procedure cancels the receive operation. The communication module reports the first corrupted transfer and the final cancelation.

Connection setup for receiving

If the 3964 procedure detects a DLE ETX string, it terminates the receive operation and confirms a successfully received block by sending a DLE to the communication partner. In the case of a receive error, an NAK is sent to the communication partner. A retry is then expected.

The 3964R procedure terminates the receive operation after having detected the DLE ETX BCC string. It compares the received block check character BCC with the internally calculated longitudinal parity. If the BCC is correct and no other receive errors have occurred, the 3964R procedure sends a DLE and returns to the idle state. The communication module informs the control system that new receive data is available.

If the BCC is faulty or a different receive error occurs, an NAK is sent to the communication partner. A retry is then expected.

Communication using USS

Overview of USS communication

Position in the system environment

The following USS description refers to the use of the corresponding communication modules.

- CM PtP RS232 BA
- CM PtP RS422/485 BA
- CM PtP RS232 HF
- CM PtP RS422/485 HF
- CM PtP (ET 200SP)

Introduction

The USS® protocol (Universal Serial Interface Protocol) is a basic serial data transmission protocol designed to meet the requirements of drive technology.

The USS protocol defines an access method based on the master-slave principle for communication via a serial bus. One master and up to 16 drives (slaves) can be connected to the bus. The individual drives are selected by the master using an address character in the frame. A drive can never send anything without first being initiated by the master. Therefore, direct data transmission between individual drives is not possible. Communication functions in half-duplex mode. The master function cannot be transferred.

Drive technology requires specific response times for the control tasks and therefore strict cyclical frame traffic:

The master continuously sends frames (job frames) to the drives and expects a response frame from each addressed drive.

A drive must send a response frame if

- it has a received a frame without errors and
- it was addressed in this frame.

A drive may not send if these conditions are not met or the drive was addressed in the broadcast.

The connection with the respective drives exists for the master once it receives a response frame from the drive after a specified processing time (response delay time).

frame structure

Each frame begins with a start character (STX), followed by the length specification (LGE) and the address byte (ADR). The data field comes after that. The frame ends with the block check character (BCC). The frame length includes the user data (quantity n), the address byte (ADR) and the data verification character (BCC).

STX	LGE	ADR	1	2	...	N	BCC

For single-word (16-bit) data, the high byte is sent first followed by the low byte. Correspondingly, with double-word data the high word is sent first, followed by the low word. The length of a frame is specified in bytes.

Data encryption

The data is encrypted as follows:

- STX: 1 byte, start of text, 02H
- LGE: 1 byte, contains the frame length as a binary number
- ADR: 1 byte, contains the slave address and frame type in binary code
- Data fields: One byte each, content depending on job
- BCC: 1 byte, block check character

Data transmission procedure

The master ensures cyclic data transmission in frames. The master addresses all slave devices one after another with a job frame. The nodes addressed respond with a response frame. In accordance with the master-slave procedure, the slave must send the response frame to the master after it has received the job frame. Only then can the master address the next slave.

Data field in the frame

The data field is divided into two areas: the parameter area (PKW) and the process data area (PZD).

STX	LGE	ADR	Parameter (PKW)	Process data (PZD)	BCC

- **Parameter area** (PKW)
  The PKW area handles parameter transmission between two communication partners (e.g., controller and drive). This involves, for example, reading and writing parameter values and reading parameter descriptions and the associated text. The PKW interface generally contains jobs for operation and display, maintenance and diagnostics.
- **Process data area** (PZD)
  The PZD area consists of signals that are required for automation:
  - Control words and setpoints from the master to the slave
  - Status words and actual values from the slave to the master

  The contents of the parameter area and process data area are defined by the slave drives. For additional information on this, refer to the drive documentation.

Overview of functions

Transmission sequence

The instructions process the data transmission cyclically with up to 16 drive slaves. Only one job is active for each drive at any one time.

Performance features:

- Creation of data storage areas for communication, depending on the bus configuration
- Execution and monitoring of PKW jobs
- Monitoring of the complete system and troubleshooting
- Communication with the CPU
- Access to the drive functions
- Reading the drive parameters
- Writing the drive parameters

Communication through Modbus RTU

Overview of modbus communication

Modbus RTU communication

Modbus RTU (Remote Terminal Unit) is a standard protocol for communication in the network and uses the electrical RS232 or RS422/485 connection for serial data transmission between Modbus devices in the network.

Modbus RTU uses a master/slave network in which the entire communication is triggered by only one master device while the slaves can only respond to the request of the master. The master sends a request to a slave address and only this slave address responds to the command (exception: broadcast frames to slave address 0 which are not acknowledged by the slaves).

The procedure used is a code-transparent, asynchronous half-duplex procedure. Data transmission is carried out without handshake.

Position in the system environment

The following Modbus description refers to the use of the corresponding communication modules.

- CM PtP RS232 HF
- CM PtP RS422/485 HF
- CM PtP (ET 200SP)

Function of the coupling

With the corresponding communication modules and the related instructions, you can establish a communication connection between a remote Modbus control system and a SIMATIC S7.

The GOULD-MODBUS protocol in RTU format is used for transmission.

Function codes 01, 02, 03, 04, 05, 06, 08, 15 and 16 are used for communication between a communication module operated as a Modbus slave and a master system (see Function Codes (Page 1004)).

If a SIMATIC S7 communication module is operated as a Modbus master, function codes 11 and 12 are also available.

SIMATIC S7 as a Modbus slave

The master has the initiative for transmission, the communication module works as a slave.

frame traffic from slave to slave is not possible.

The instruction Modbus_Slave makes the data available on a SIMATIC data area in accordance with the mapping specification or stores them.

SIMATIC S7 as a Modbus master

As master, the communication module initiates transmission and, after outputting a request frame, it waits for the configured response monitoring time for a response frame from the slave. If the slave does not respond, the master repeats the request in accordance with the configuration before it outputs an error message.

frame structure

The data exchange "Master-Slave" and/or "Slave-Master" begins with the **slave address**, followed by the **function code**. Then the data are transferred. The structure of the data field depends on the function code used. The CRC check is transmitted at the end of the frame.

ADDRESS	FUNCTION	DATA	CRC-CHECK
Byte/Word	Byte	n byte	2 byte

ADDRESS	Modbus slave address • Standard address: 1 to 247 (bytes) • Extended station address: 1 to 65535 (word)
FUNCTION	Modbus Function Codes (Page 1004)
DATA	frame data: Management and net data depending on the function code
CRC-CHECK	frame checksum

Slave address

The slave address can be range from 1 to 247 (byte) or 1 to 65535 (word). The address is used to address a defined slave on the bus.

Broadcast Message

The master uses slave address 0 to address all slaves on the bus.

Broadcast messages are only permitted in conjunction with writing Function codes 05, 06, 15 and 16.

A broadcast message is not followed by a response frame from the slave.

Data Field DATA

The data field DATA is used to transfer the function code-specific data such as:

- Bytecount, Coil_Startaddress, Register_Startaddress; Number_of_Coils, Number_of_Registers, ...
  For details, see "Function Codes (Page 1004)".

CRC-Check

The end of the frame is identified by means of the CRC 16 checksum consisting of 2 bytes. It is calculated by the following polynominal: $x^{16} + x^{15} + x^2 + 1$.

The low byte is transmitted first, followed by the high byte.

End of frame

The end of frame is recognized when no transmission takes place during the time period required for the transmission of three and a half characters (3.5 times character delay time) (see Modbus Protocol Reference Guide).

This end of frame TIME_OUT therefore depends on the data transmission rate and is indicated in bit times (35 bit times are fix coded internally; further bit times can be configured in addition at the instruction).

The Modbus frame received from the connection partner is evaluated and formally checked after the end of frame TIME_OUT is received.

Exception responses

On recognition of an error in the request frame from the master, for example, register address illegal, the slave sets the highest value bit in the function code of the response frame.

This step is followed by transmission of a byte exception code that describes the cause of the error.

A detailed description of the meaning of the above-mentioned parameters is available in the "GOULD MODICON Modbus Protocol" (not part of this documentation).

Exception code frame

The exception code frame from the slave has the following structure:

- for example, slave address 5, function code 5, exception code 2

Response frame from the slave EXCEPTION_CODE_xx:

05H	Slave address
85H	Function code
02H	Exception code (1...7)
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

On receipt of an exception code frame by the driver, the current job is completed with error.

The following error codes are defined in accordance with the Modbus specification:

Error code	Meaning in accordance with Modbus specification	Cause - Short Description *
1	Illegal function	Illegal function code
2	Illegal data address	Slave has illegal data address
3	Illegal data value	Slave has illegal data value
4	Failure in Associated Device	Slave has internal error
5	Acknowledge	Function is carried out
6	Busy, Rejected message	Slave is not ready to receive
7	Negative acknowledgement	The function cannot be carried out.
* Check slave for further details.		

RS232 mode

The following communication modules support RS232 mode:

- CM PtP RS232 HF
- CM PtP (ET 200SP)

For more information on RS232 mode, see the chapter RS232 mode (Page 981).

For information on hardware data flow control and on automatic operation of the accompanying signals, refer to the Handshake procedure (Page 984) chapter.

RS422/485 mode

The following communication modules support RS422/485 mode:

- CM PtP RS422/485 HF
- CM PtP (ET 200SP)

For more information on RS422/485 mode, see the chapters RS422 mode (Page 983) and RS485 mode (Page 983).

Function Codes

Used function codes

The function code defines the meaning of the frame. It also defines the structure of a frame.

The following function codes are supported by the communication module:

Function code	Function in accordance with MODBUS specification	Range
01	Read Coil Status	1 to 2000 bit/request
02	Read Input Status	1 to 2000 bit/request
03	Read Holding Registers	1 to 124/125 bit/request (124 at extended station address)
04	Read Input Registers	1 to 124/125 bit/request (124 at extended station address)
05	Force Single Coil	1 bit/request
06	Preset Single Register	1 word/request
08 *	Loop Back Test	Read slave status or reset event counter in the slave
11 *	Fetch Communications Event Counter (only master)	-
15	Force Multiple Coils	1 to 1968 bits/request
16	Preset Multiple Registers	1 to 123 bit/request

* Diagnostic information for slave communication

Modbus function code 00 sends a broadcast message to all slaves (without slave response).

Assignment of the Modbus addresses to the SIMATIC addresses

The table below shows the assignment of the Modbus addresses to the SIMATIC addresses.

Modbus				S7-1500	
FC 1)	Function	Declaration	Address area	Declaration	CPU address
01	Read bits	Output	1 - 9999	Process image of outputs	Q0.0 to Q1248.6
02	Read bits	Input	10001 - 19999	Process image of inputs	I0.0 - 1248.6
03 2)	Read words	Holding Register	40001 - 49999 or 400001 - 465535	DW0 - DW19998 or DW0 - DW131068	The M address area depends on the CPU
04	Read words	Input	30001 - 39999	Process image of inputs	IW0 to IW19996
05 2)	Write bits	Output	1 - 9999	Process image of outputs	Q0.0 to Q1248.7
06	Write words	Holding Register	40001 - 49999 or 400001 - 465535	DW0 - DW19998 or DW0 - DW131068	The M address area depends on the CPU
15	Write bits	Output	1 - 9999	Process image of outputs	Q0.0 to Q1248.6
16 2)	Write words	Holding Register	40001 - 49999 or 400001 - 465535	DW0 - DW19998 or DW0 - DW131068	The M address area depends on the CPU

1) FC = function code

2) The value of the HR_Start_Offset determines whether data areas or bit memory address areas can be addressed with the FCs 03, 05 and 16 in the SIMATIC CPU.

Programming - communication using instructions

Overview of point-to-point programming

Data exchange using Freeport or 3964(R) communication

You must make the send data available in data blocks or in the bit memory address area in the user program of the corresponding CPU. A receive buffer is available in the communication module for the receive data. A corresponding data block is set up in the data block.

In the user program of the CPU, the instructions Send_P2P and Receive_P2P carry out the data transfer between the CPU and the communication module. The receive buffer can be deleted with the instruction Receive_Reset.

Dynamic configuration by means of the user program

As an alternative to or in addition to the parameter assignment of the communication module interface described in chapter Configuring / parameter assignment of a communication module (Page 988), it may be advisable in certain application areas to set up the communication dynamically, i.e., program-controlled by a specific application.

All configurations carried out in the properties dialog of the communication module can also be changed during runtime by means of the corresponding "Config" instructions (Port_Config, Send_Config, Receive_Config, P3964_Config).

Program calls for point-to-point communication - sequence

The figure below shows the function of the point-to-point instructions for communication between the user program and communication partner.

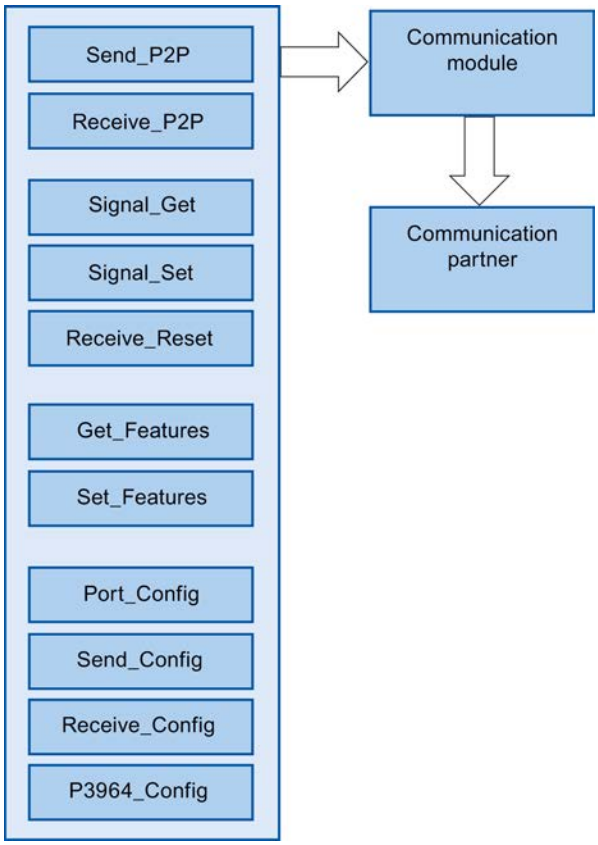


Figure 8-13 Program sequence

PtP instructions

- Instructions for dynamic parameter assignment
  - Port_Config
    You can use the Port_Config instruction (port configuration) to configure basic interface parameters, such as the data transmission rate, parity and data flow control dynamically through your user program.
  - Send_Config
    With the instruction Send_Config (send configuration) you can configure serial send parameters, such as RTS ON-/RTS OFF delay, dynamically for a point-to-point communication interface.
  - Receive_Config
    The instruction Receive_Config (receive parameter assignment) allows you to dynamically assign serial receive parameters to a communication module.
    This instruction configures the conditions that specify the start and the end of a received message.
  - P3964_Config
    The instruction P3964_Config (configure protocol) can be used to dynamically configure protocol parameters of the procedure 3964(R), such as character delay time, priority and block check using your program.
- Instructions for data exchange (communication)
  - Send_P2P
    The instruction Send_P2P (send point-to-point data) can be used to send data to the communication partner.
    Call up the instruction Send_P2P to send data using the Freeport protocol. You have to call the instruction cyclically until you receive a corresponding acknowledgement at the output parameters of the instruction.
    Note: During parameter assignment of the XON/XOFF data flow control, user data may not contain any of the configured XON or XOFF characters. Default settings are DC1 = 11H for XON and DC3 = 13H for XOFF.
  - Receive_P2P
    The instruction Receive_P2P (receive point-to-point data) can be used to pick up the messages received in the communication module from a communication partner.
    Call the instruction Receive_P2P cyclically to receive data using the Freeport protocol. The instruction indicates at the NDR parameter if new received data is available.
    To signal the start and end of a message transmission, you need to define criteria in the Freeport protocol which identify the start and end of the message.
  - Receive_Reset
    The instruction Receive_Reset (delete receive buffer) allows you to clear the receive buffer of the communication module.

- Instructions for operation of the RS232 signals
  - Signal_Get
    With the Signal_Get instruction (get RS232 signals) you can read the current states of the RS232 signals.
  - Signal_Set
    With the instruction Signal_Set (get RS232 signals), you can set the states of the RS232 signals DTR and RTS.
- Instructions for advanced functions
  - Get_Features
    You can use the instruction Get_Features (get extended functions) to get information on the Modbus support and on generating diagnostic alarms.
  - Set_Features
    If supported by the module, you can use the instruction Set_Features (set extended functions) to activate the generation of diagnostic alarms.

Procedure for setting up Freeport or 3964(R) communication

Requirement: The configuration and parameter assignment of a CPU and a communication module in the device view and in the properties dialog of the communication module are complete.

1. In the project navigation for the CPU select the folder "Program blocks" and open the Main (OB1) in the folder by double-clicking it. The program editor opens.
2. From the "Instructions" task card, "Communication" area select the instructions Send_P2P and Receive_P2P and drag-and-drop them into a network of the Main (OB1).
3. Configure the instructions in accordance with your specifications.
4. Download the hardware configuration and the user program to the CPU.

Additional information

You can find a list of available instructions in chapter Overview of instructions (Page 977).

Information about the instructions is available in the info system of STEP 7 V12.

Overview of USS programming

Program calls for USS communication - sequence

The figure below shows the function of the USS instructions for communication between user program and USS drive. (The instructions Send_P2P, Receive_P2P and the Config instruction are required downstream).

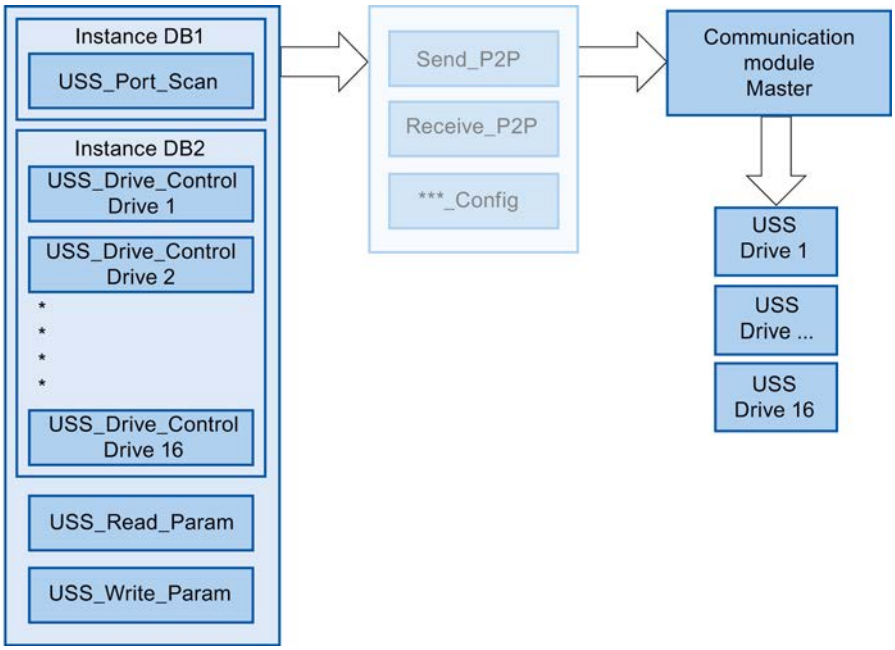


Figure 8-14 Program sequence

USS instructions

- USS_Port_Scan (Page 5220):
  The instruction USS_Port_Scan allows you to communicate via a communication module with up to 16 drives using a USS network (must be called cyclically).
  The instruction USS_Port_Scan controls the communication between CPU and the drives by means of the PtP communication port. A communication with the drive is processed every time you call this function. The instruction USS_Port_Scan is required once:
  Since most drives features a configurable internal function that monitors the integrity of the communication based on a timeout, the instruction USS_Port_Scan should be called from a time-controlled OB.
- USS_Drive_Control (Page 5222):
  The instruction USS_Drive_Control allows you to prepare the send data for a drive and to display the receive data.
  The inputs and outputs of the instruction correspond to the states and operating functions of the drive. The instruction USS_Drive_Control must be called once for each drive. Only one common instance DB is required for all calls of the instruction USS_Drive_Control for a USS network. Interconnect all calls of the instructions USS_Drive_Control for a USS network with the same instance DB.
  The instruction USS_Drive_Control should be called from the cyclic Main (OB1) of the main program.
- USS_Read_Param (Page 5225):
  The instruction USS_Read_Param allows you to read parameters from the drive.
  You use the USS_Read_Param instruction to read the operating parameters of the drive that control the internal drive functions.
  The instruction USS_Read_Param should be called from the cyclic Main (OB1) of the main program.
- USS_Write_Param (Page 5226):
  The instruction USS_Write_Param allows you to change parameters in the drive.
  The instruction USS_Write_Param should be called from the cyclic Main (OB1) of the main program.

Procedure for setting up USS communication

Requirement: The configuration and parameter assignment of a CPU and a communication module in the device view and in the properties dialog of the communication module are complete.

1. In the project tree for the CPU, select the "Program blocks" folder and open the desired time-controlled OB by double-clicking it. The program editor opens.
2. From the "Instructions" task card, "Communication" area select the instruction USS_Port_Scan and drag-and-drop it into a network of a time-controlled OB.
   The instruction USS_Port_Scan allows you to communicate by means of the USS network.
3. In the project navigation for the CPU select the folder "Program blocks" and open the Main (OB1) in the folder by double-clicking it. The program editor opens.

4. From the "Instructions" task card, "Communication" area select the instructions for USS communication in accordance with your task and drag-and-drop them into a network of the Main (OB1):
   - The instruction USS_Drive_Control is used for data exchange with the drive.
   - The instruction USS_Read_Param is used for reading parameters from the drive.
   - The instruction USS_Write_Param is used for changing parameters in the drive.
5. Configure the instructions in accordance with your specifications.
6. Download the hardware configuration and the user program to the CPU.

Additional information

You can find a list of available instructions in chapter Overview of instructions (Page 977).

Information about the instructions is available in the info system of STEP 7 V12.

Overview of Modbus programming

Program calls for Modbus communication - sequence

The figure below shows the function of the Modbus instructions for communication between user program and Modbus device. (The instructions Send_P2P, Receive_P2P and the Config instruction are required downstream).

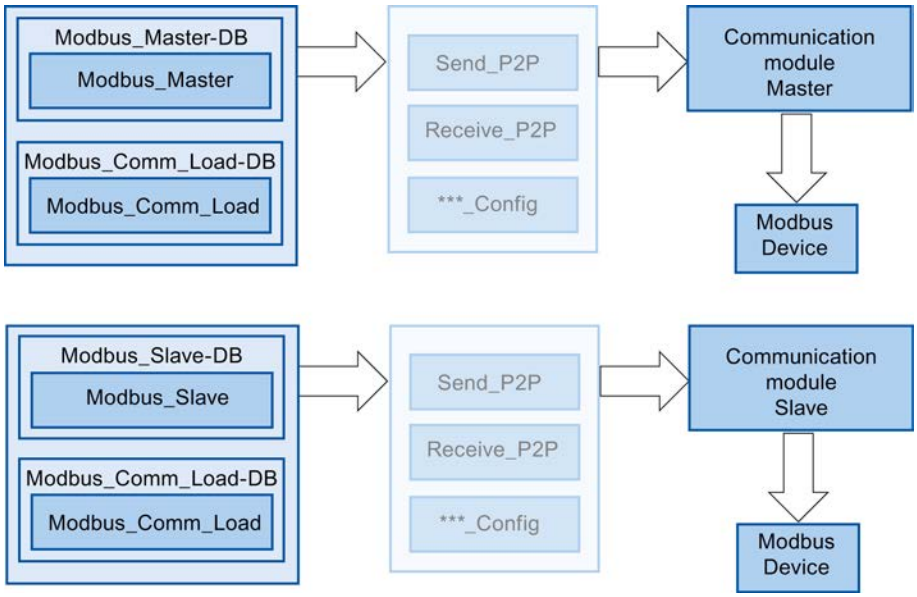


Figure 8-15 Program sequence

Modbus instructions

- Modbus_Comm_Load (Page 5246):
  The instruction Modbus_Comm_Load allows you to configure the port of the communication module for Modbus RTU.
  You have to run Modbus_Comm_Load to set up PtP port parameters, such as data transmission rate, parity and flow control. Once you have configured the interface for the Modbus RTU protocol, it can only be used by the instruction Modbus_Master or the instruction Modbus_Slave .
- Modbus_Master (Page 5249):
  The instruction Modbus_Master allows you to communicate as Modbus master by means of the PtP port.
  The CPU can be used as Modbus RTU master device with the Modbus_Master instruction for communication with one or several Modbus slave devices.
- Modbus_Slave (Page 5254):
  The instruction Modbus_Slave allows you to communicate as Modbus slave by means of the PtP port.
  The CPU can be used as Modbus RTU slave device with the Modbus_Slave instruction for communication with one Modbus master device.

Note

Alternative use of Modbus_Slave and Modbus_Master

A communication module can be operated either as master or as slave.

Procedure for setting up Modbus communication

Requirement: The configuration and parameter assignment of a CPU and a communication module in the device view and in the properties dialog of the communication module are complete.

1. In the project navigation for the CPU select the folder "Program blocks" and open the Main (OB1) in the folder by double-clicking it. The program editor opens.
2. From the "Instructions" task card, "Communication" area select the instructions for Modbus communication in accordance with your task and drag-and-drop them into a network of the Main (OB1):
   - The instruction Modbus_Comm_Load configures the port of the communication module for Modbus communication.
     The Modbus_Comm_Load must be called in Main (OB1) until DONE (or ERROR) is reported.
   - The instruction Modbus_Master is used for the Modbus master functionality.
   - The instruction Modbus_Slave is used for the Modbus slave functionality.
3. Configure the instructions in accordance with your specifications.
4. Download the hardware configuration and the user program to the CPU.

Additional information

You can find a list of available instructions in chapter Overview of instructions (Page 977).

Information about the instructions is available in the info system of STEP 7 V12.

Startup and Diagnostics

Startup characteristics

Operating mode transitions

After the communication module starts up, all data between the CPU and the communication module is exchanged by means of instructions.

CPU STOP	During a running data transmission communication module - CPU, both a send and a receive job is aborted.
CPU RUN	Send and receive operation is ensured in the RUN state of the CPU. With a corresponding configuration in the properties dialog of the communication module, you can automatically clear the receive buffer on the communication module during CPU startup.

From the view of the communication module, there are no further operating states/operating state transitions.

Diagnostic functions

Introduction

The diagnostic functions of the communication module allow errors that have occurred to be located quickly. The following diagnostic options are available to you:

Diagnostics by means of the display elements of the communication module	The indicators provide information on the operating mode or the possible error states of the communication module. The indicators provide an initial overview of any internal or external errors and interface-specific errors. For more information refer to the device manual of the corresponding communication module.
Diagnostics via the STATUS output of the instructions	Instructions have a STATUS output for error diagnostics; it provides information about communication errors between the communication module and the CPU. You can evaluate the STATUS parameter in the user program (it is active for exactly one cycle).
Diagnostic error interrupt	The communication module can trigger a diagnostic error interrupt on the CPU assigned to it. The communication module makes diagnostic information available. The analysis of this information is made via the user program or by reading the CPU diagnostics buffer.

Error messages

Overview of error messages - PtP

The error messages are provided at the STATUS output of an instruction and can be evaluated there or processed in the user program.

Error code	Description	Remedy
16#0000	No error	-
RECEIVE status and error codes		
16#0094	frame end identified based on the "Receipt of fixed/ maximum frame length"	-
16#0095	frame end identified based on "Message timeout"	-
16#0096	frame end identified based on expiration of the "Character delay time"	-
16#0097	The frame was aborted because the maximum response time was reached.	-
16#0098	frame end identified based on the fulfillment of the "Read message length from message" conditions	-
16#0099	frame end identified based on the receipt of the "End sequence"	-
SEND status and error codes		
16#7000	Block idle	-
16#7001	Initial call for a new frame: Data transmission initiated	-
16#7002	Interim call: Data transmission running	-
16#8085	Invalid length	Select a suitable frame length. The following are valid (depending on the module): 1-1024/2048/4096 (Byte)
16#8088	The specified length exceeds the range set in the receive buffer. Note: If the data type STRING has been specified at the BUFFER parameter, this error code also appears if the current string is shorter than the length specified at the LENGTH parameter.	Change the range in the receive buffer or select a frame length which corresponds to the range set in the receive buffer. The following are valid (depending on the module): 1-1024/2048/4096 (Byte)
16#8090	Configuration error: Odd number of bytes for WString	Select an even number of bytes.
RECEIVE status and error codes		
16#8088	The number of characters received exceeds the number specified at the BUFFER parameter.	Select a suitable frame length. The following are valid (depending on the module): 1-1024/2048/4096 (Byte)
16#8090	Configuration error: Odd number of bytes for WString	Select an even number of bytes.
Error message codes of the special functions		
16#818F	Incorrect parameter number setting (with USS only)	Select a suitable parameter number (PARAM). The following numbers are valid: 0-2047
16#8190	Incorrect setting of the CRC calculation	Select a suitable value for the CRC calculation. The following are valid: deactivated or activated. Check whether the module addressed supports CRC calculation.

Error code	Description	Remedy
16#8191	Incorrect setting of the diagnostic error interrupt	Select a suitable value for "Diagnostic error interrupt". The following are valid: Diagnostic error interrupt deactivated or diagnostic error interrupt activated. Check whether the module addressed supports the generation of diagnostic interrupts.
Error message codes of the "Port configuration"		
16#81A0	The module does not support this protocol.	Select a valid protocol for the module (PROTOCOL).
16#81A1	The module does not support this data transmission rate.	Select a valid data transmission rate for the module (BAUD).
16#81A2	The module does not support this parity setting.	Select a suitable value for "Parity" (PARITY). The following are valid: • None (1) • Even (2) • Odd (3) • Mark (4) • Space (5) • Any (6)
16#81A3	The module does not support this number of data bits.	Select a suitable value for "Number of data bits" (DATABITS). The following are valid: • 7 (2) • 8 (1)
16#81A4	The module does not support this number of stop bits.	Select a suitable value for "Number of stop bits" (STOPBITS). The following are valid: • 1 (1) • 2 (2)
16#81A5	The module does not support this type of data flow control.	Select a valid data flow control for the module (FLOWCTRL).
16#81A7	Invalid value for XON or XOFF	Select suitable values for XON (XONCHAR) and XOFF (XOFFCHAR). Valid range of values: 0...255
16#81AA	Invalid operating mode	Valid operating modes are: • Full duplex (RS232) (0) • Full duplex (RS422) four-wire mode (point-to-point) (1) / (CM PtP (ET 200SP)) • Full duplex (RS422) four-wire mode (multipoint master) (2)/ (CM PtP (ET 200SP)) • Full duplex (RS 422) four-wire mode (multipoint slave) (3) • Half duplex (RS485) two-wire mode. (4)

Error code	Description	Remedy
16#81AB	Invalid receive line initial state	Valid initial states are: • "No" default setting (0) • Signal R(A)=5 V, signal R(B)=0 V (break detection) (1): Can only be selected with: "Full duplex (RS422) four-wire operation (point-to-point connection)" and "Full duplex (RS422) four-wire operation (multipoint slave)". • Signal R(A)=0 V, signal R(B)=5 V (2): This default setting corresponds to the idle state (no active send operation).
16#81AC	Invalid value for "Break detection"	Select a suitable value for "Break detection". The following are valid: • Break detection deactivated (0) • Break detection activated (1).
16#81AF	The module does not support this protocol.	Select a valid protocol for the module.
Error codes of the "Send configuration"		
16#81B5	More than two end delimiters or end sequence > 5 characters	Select suitable values for "End delimiter" and "End sequence". The following are valid: • deactivated (0), • 1 (1) or 2 (2) end delimiters or • deactivated (0), • 1 (1) up to 5 (5) characters for the end sequence.
16#81B6	Send configuration rejected because the 3964(R) protocol was selected	Make sure that no send configuration is transmitted if the 3964(R) protocol is set.
Error codes of the "Receive configuration"		
16#81C0	Invalid start condition	Select a suitable start condition. The following are valid: • Send break before frame start • Send Idle Line.
16#81C1	Invalid end condition or no end condition selected	Select a suitable end condition (see Receiving data with Freeport (Page 991)).
16#81C3	Invalid value for "Maximum message length"	Select a suitable value for "Maximum message length" (MAXLEN). Valid range of values (depending on the module): 1-1024/2048/4096 (Byte)
16#81C4	Invalid value for "Offset of the length specification in the message"	Select a suitable value for "Offset of the length specification in the message". Valid range of values (depending on the module): 1-1024/2048/4096 (Byte)

Error code	Description	Remedy
16#81C5	Invalid value for "Size of length field"	Select a suitable value for "Size of length field" (LENGTHSIZE). Valid range of values in bytes: • 1 (1) • 2 (2) • 4 (4)
16#81C6	Invalid value for "Number of characters not counted in length specification"	Select a suitable value for "Number of characters not counted in length specification" (LENGTHM). Valid range of values: 0 to 255 (bytes)
16#81C7	The total of "Offset in the message + size of length field + number of characters not counted" is greater than the maximum frame length	Select a suitable value for "Offset in message", "Size of length field" and "Number of characters not counted". Valid range of values: • Offset in the message (depending on the module): 0-1024/2048/4096 (Byte) • Size of length field: 1, 2, or 4 (bytes) • Number of characters not counted: 0 to 255 (bytes)
16#81C8	Invalid value for "Response timeout"	Select a suitable value for "Response timeout". Valid range of values: 1-65535 (ms)
16#81C9	Invalid value for "Character delay time"	Select a suitable value for "Character delay time". Valid range of values: 1 to 65535 (bit times)
16#81CB	frame end sequence is activated, but no character is activated for the check	Activate one or several characters for the check.
16#81CC	frame start sequence is activated, but no character is activated for the check	Activate one or several characters for the check.
16#81CD	Invalid value for "Prevent overwriting"	Select a suitable value for "Prevent overwriting". The following are valid: • Prevent overwriting is deactivated (0) or • Prevent overwriting is activated (1)
16#81CE	Invalid value for "Clear receive buffer on startup"	Select a suitable value for "Clear receive buffer on startup". The following are valid: • Clear receive buffer at startup is deactivated (0) • Clear receive buffer at startup is activated (1)
SEND status and error codes		
16#81D0	Receiving send request during runtime of a send command	Make sure that you do not receive an additional send request during runtime of a send command.
16#81D1	The waiting time for XON or CTS = ON has expired.	The communication partner has a fault, is too slow or is offline. Check the communication partner or change the parameters, if necessary.
16#81D2	"Hardware RTS always ON": Send job canceled due to change from DSR = ON to OFF	Check the communication partner. Make sure that DSR is ON for the entire duration of transmission.
16#81D3	Send buffer overflow / send frame too long	Select a shorter frame length. The following are valid (depending on the module): 1 to 1024/2048/4096 (bytes)

Error code	Description	Remedy
16#81D5	Transmission canceled due to parameter changes, detected wire break, or CPU in STOP	Check the parameter assignment, wire break, and CPU status.
16#81D6	Transmission canceled because end identifier was not received	Check the parameter assignment of the end characters and the frame of the communication partner.
16#81D7	Communication error between the user program and module	Check the communication (e.g., matching the sequence number).
16#81D8	Transmission attempt rejected because module is not configured	Configure the module.
Error codes of the receive configuration		
16#81E0	Frame aborted: Send buffer overflow / send frame too long	Increase the call rate for the receive function in the user program or configure communication with data flow control.
16#81E1	Frame aborted: Parity error	Check the connection line of the communication partners, or verify that both devices are configured for the same data transmission rate, parity and stop bit number.
16#81E2	Frame aborted: Character frame error	Check the settings for start bit, data bits, parity bit, data transmission rate, and stop bit(s).
16#81E3	Frame aborted: Character overflow error	Firmware error: Please contact Customer Support.
16#81E4	Frame aborted: The total length of "Offset in the message + size of the length field + number of characters not counted" is greater than the receive buffer	Select a suitable value for offset in message, size of the length field, and number of characters not counted.
16#81E5	Frame aborted: Break	Receive line to partner is interrupted. Reconnect or switch on partner.
16#81E6	Maximum number of "Buffered received frames" exceeded	In the user program call the instruction more often or configure a communication with data flow control or increase the number of buffered frames.
16#81E7	Synchronization error module and Receive_P2P	Make sure that different instances of the Receive_P2P do not access the same module.
16#81E8	Frame aborted: The character delay time has expired before the message end criterion was detected	Partner device faulty or too slow. Check this, if required, using an interface tester that is interconnected in the transmission line.
16#81E9	Modbus CRC error (only communication modules which support Modbus)	Checksum error of the Modbus frame. Check the communication partner.
16#81EA	Modbus frame too short (only communication modules which support Modbus)	Minimum length of Modbus frame not met. Check the communication partner.
16#81EB	Frame aborted: Maximum frame length reached	Select a shorter frame length at the communication partner. The following are valid (depending on the module): 1-1024/2048/4096 (Byte) Check the parameters for end of frame detection.
Error codes V24 accompanying signals		
16#81F0	The module does not support V24 accompanying signals	You have tried to set accompanying signals for a module that does not support V24 accompanying signals. Make sure that this is an RS232 module or that RS232 mode (ET 200SP) is set.

Error code	Description	Remedy
16#81F1	No operation of the V24 accompanying signals	The V24 accompanying signals cannot be operated manually if hardware data flow control is active.
Error codes of the receive configuration		
16#8201 [1)]	Receive_Conditions is a pointer to an invalid data type	Enter a pointer to one of the following data types: DB, BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TIME_OF_DAY, TIME, S5TIME, DATE_AND_TIME, STRING
16#8225	Receive_Conditionspoints to an optimized memory area greater than 1 kB or Receive_Conditions points to an optimized memory area and the receive length is greater than the area addressed by Receive_Conditions.	Enter a pointer to an area with a maximum length of: • Optimized memory area: 1 kByte • Non-optimized area: 4 kByte Note: If the pointer points to an optimized memory area, do not send more than 1 kByte.
16#8229 [1)]	Receive_Conditions is a pointer to BOOL with a number of bits not equal to n * 8	If you are using a pointer to BOOL, the number of bits must be a multiple of 8.
Error codes, general		
16#8280	Negative acknowledgment when reading module	You can find more detailed information on error causes in theRDREC.STATUS static parameters and in the description of the SFB RDREC. • Check the input at the PORT parameter • Set the COM_RST parameter prior to the initial call.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter You can find more detailed information on error causes in the WRREC.STATUS static parameters and in the description of the SFB WRREC.
16#8282	Module not available	Check the input at the PORT parameter and ensure that the module can be reached.
Error codes of the receive configuration		
16#82C1	Invalid value for "Buffered received frames".	Select a suitable value for "Buffered received frames". Valid range of values: 1-255
16#82C2	Receive configuration rejected because the 3964(R) protocol was selected	Make sure that no receive configuration is sent if the 3964(R) protocol is set.
16#8301 [1)]	Receive_Conditions is a pointer to an invalid data type	Select a valid data type. The following are valid: DB, BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TIME_OF_DAY, TIME, S5TIME, DATE_AND_TIME, STRING
16#8322	Range length error when reading a parameter	Check the input at the Receive_Conditions parameter
16#8324	Range error when reading a parameter	Check the input at the Receive_Conditions parameter
16#8328	Setting error when reading a parameter	Check the input at the Receive_Conditions parameter
SEND status and error codes		
16#8328 [1)]	BUFFER is a pointer to BOOL with a number of bits not equal to n * 8	If you are using a pointer to BOOL, the number of bits must be a multiple of 8.
Error codes of the receive configuration		
16#8332	Invalid data block at the Receive_Conditions parameter	Check the input at the Receive_Conditions parameter

Error code	Description	Remedy
16#833A	The designation of the data block at the Receive_Conditions parameter refers to a data block which is not loaded.	Check the input at the Receive_Conditions parameter
16#8351	Invalid data type	Check the input at the Receive_Conditions parameter
16#8352 [1)]	Receive_Conditions does not point to a data block	Check the pointer to Receive_Conditions
16#8353 [1)]	Receive_Conditions does not point to a structure of the type Receive_Conditions	Check the pointer to Receive_Conditions
Error codes 3964(R) protocol		
16#8380	Parameter assignment error: Invalid value for "Character delay time".	Select a suitable value for "Character delay time" (CharacterDelayTime). Valid range of values: 1 to 65535 (ms)
16#8381	Parameter assignment error: Invalid value for "Response timeout".	Select a suitable value for "Response timeout" (AcknDelayTime). Valid range of values: 1 to 65535 (ms)
16#8382	Parameter assignment error: Invalid value for "Priority".	Select a suitable value for "Priority" (Priority). The following are valid: • High (1) • Low (0)
16#8383	Parameter assignment error: Invalid value for "Block check"	Select a suitable value for "Block check" (BCC). The following are valid: • With block check (1) • Without block check (0)
16#8384	Parameter assignment error: Invalid value for "Connection attempts".	Select a suitable value for "Connection attempts" (BuildupAttempts). Valid range of values: 1-255
16#8385	Parameter assignment error: Invalid value for "Transmission attempts".	Select a suitable value for "Transmission attempts" (RepetitionAttempts). Valid range of values: 1-255
16#8386	Runtime error: Number of connection attempts exceeded	Check the interface cable and the transmission parameters. Also check whether the receive function is configured correctly at the partner device.
16#8387	Runtime error: Number of transmission attempts exceeded	Check the interface cable, the transmission parameters and the configuration of the communication partner.
16#8388	Runtime error: Error at the "Block check character" The internally calculated value of the block check character does not correspond to the block check character received by the partner at the connection end.	Check whether the connection is seriously disrupted; in this case you may also occasionally see error codes. Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#8389	Runtime error: Invalid character received while waiting for free receive buffer	The send request of the communication partner (STX, 02H) is only answered with DLE when the receive buffer is empty. No additional character may be received before (except STX again). Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.

Error code	Description	Remedy
16#838A	Runtime error: Logical error during receiving. After DLE was received, a further random character (other than DLE or ETX) was received.	Check if partner DLE in frame header and in data string is always in duplicate or the connection is released with DLE ETX. Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#838B	Runtime error: Character delay time exceeded	Partner device too slow or faulty. Verify by using an interface test device that is switched into the transmission line, if necessary.
16#838C	Runtime error: Wait time for free receive buffer has started	In the user program call the instruction more often or configure a communication with data flow control.
16#838D	Runtime error: frame repetition does not start within 4 s after NAK	Check the communication partner. A received frame that is possibly corrupted must be repeated by the partner within 4 seconds.
16#838E	Runtime error: In idle mode, one or several characters (other than NAK or STX) were received.	Check for proper function of the partner device, possibly using an interface test device that is switched into the transmission line.
16#838F	Runtime error: Initialization conflict - Both partners have set high priority	Set the "Low" priority at one of the partners
16#8391	Parameter assignment error: 3964 configuration data rejected because Freeport is set	If the Freeport protocol is set, make sure that no 3964 parameter assignment data is sent.

[1] Only with instructions for S7-300/400 CPUs

Overview of error messages - Modbus

Error code	Description	Remedy
16#0000	No error	-
Configuration error of the interface - Modbus_Comm_Load		
16#8181	The module does not support this data transmission rate.	Select a valid data transmission rate for the module at the BAUD parameter.
16#8182	The module does not support this parity setting.	Select a suitable value for "Parity" at the PARITY parameter. The following are valid: • None (1) • Even (2) • Odd (3) • Mark (4) • Space (5) • Any (6)
16#8183	The module does not support this type of data flow control.	Select a valid data flow control for the module at the FLOW_CTRL parameter.
16#8184	Invalid value for "Response timeout".	Select a suitable value for "Response timeout" at the RESP_TO parameter. Valid range of values: 1 to 65535 (ms)

Error code	Description	Remedy
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the Send_Config.RDREC.STATUS or Receive_Config.RDREC.STATUS static parameters or RDREC.STATUS and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the Send_Config.WRREC.STATUS or Receive_Config.WRREC.STATUS static parameters or WRREC.STATUS and in the description of the SFB WRREC.
16#8282	Module not available	Check the input at the PORT parameter and ensure that the module can be reached.
Configuration error - Modbus_Slave		
16#8186	Invalid slave address	Select a suitable slave address at the MB_ADDR parameter. The following are valid: 1-247 at standard address area; 1-65535 at extended address area (0 is reserved for Broadcast)
16#8187	Invalid value at MB_HOLD_REG parameter	Select a suitable value for the hold register at the MB_HOLD_REG parameter.
16#8188	Invalid operating mode or broadcast (MB_ADDR = 0) and MODE parameter ≠ 1	Select the value 1 for MODE in Broadcast mode or select a different operating mode.
16#818C	The pointer to a MB_HOLD_REG area must be a data block or a bit memory address area.	Select a suitable value for the pointer to the MB_HOLD_REG area.
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.WRREC.STATUS or Receive_P2P.WRREC.STATUS, and in the description of the SFB WRREC.
16#8452 [1)]	MB_HOLD_REG is not a pointer to a DB or a bit memory area	Check the MB_HOLD_REG pointer
16#8453 [1)]	MB_HOLD_REG is not a pointer of type BOOL or WORD	Check the MB_HOLD_REG pointer
16#8454 [1)]	The area addressed by MB_HOLD_REG is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the MB_HOLD_REG pointer
16#8455 [1)]	MB_HOLD_REG points to a write-protected DB	Check the MB_HOLD_REG pointer
16#8456 [1)]	Error during instruction execution. The cause of the error is shown in the STATUS parameter.	Determine the value of the SFCSTATUS parameter. Check what this means in the description for SFC51, STATUS parameter.

Error code	Description	Remedy
Configuration error - Modbus_Master		
16#8180	Invalid value for MB_DB parameter	The value configured for MB_DB (instance data DB) at the Modbus_Comm_Load instruction is not valid. Check the interconnection of the Modbus_Comm_Load instruction and its error messages.
16#8186	Invalid station address	Select a suitable station address at the MB_ADDR parameter. The following are valid: 1-247 at standard address area; 1-65535 at extended address area (0 is reserved for Broadcast)
16#8188	Invalid operating mode or broadcast (MB_ADDR = 0) and MODE parameter ≠ 1	Select the value 1 for MODE in Broadcast mode or select a different operating mode.
16#8189	Invalid data address	Select a suitable value for the data address at the DATA_ADDR parameter. See description Modbus_Master (Page 5249) in the Info system
16#818A	Invalid length	Select a suitable data length at the DATA_LEN parameter. See description Modbus_Master (Page 5249) in the Info system
16#818B	Invalid value for DATA_PTR	Select a suitable value for the data pointer at the DATA_PTR parameter (M or DB address). See description Modbus_Master (Page 5249) in the Info system
16#818C	Interconnection error of the DATA_PTR parameter	Check the interconnection of the instruction.
16#818D	The area addressed by DATA_PTR is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the DATA_PTR pointer
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.WRREC.STATUS or Receive_P2P.WRREC.STATUS or Receive_Reset and in the description of the SFB WRREC.
Communication errors - Modbus_Master and Modbus_Slave		
16#80 D1	The waiting time for XON or CTS = ON has expired.	The communication partner has a fault, is too slow or is offline. Check the communication partner or change the parameters, if necessary.
16#80D2	"Hardware RTS always ON": Send job canceled due to change from DSR = ON to OFF	Check the communication partner. Make sure that DSR is ON for the entire duration of transmission.

Error code	Description	Remedy
16#80E0	Frame aborted: Send buffer overflow / send frame too long	In the user program call the instruction more often or configure a communication with data flow control.
16#80E1	Frame aborted: Parity error	Check the connection line of the communication partners, or verify that both devices are configured for the same data transmission rate, parity and stop bit number.
16#80E2	Frame aborted: Character frame error	Check the settings for start bit, data bits, parity bit, data transmission rate, and stop bit(s).
16#80E3	Frame aborted: Character overflow error	Check the number of data in the frame of the communication partner.
16#80E4	Frame aborted: Maximum frame length reached	Select a shorter frame length at the communication partner. The following are valid (depending on the module): 1 to 1024/2048/4096 (bytes)
Communication error - Modbus_Master		
16#80C8	The slave does not respond within the set time	Check the data transmission rate, parity and wiring of the slave.
16#8200	The interface is busy with an ongoing request.	Repeat the command later. Make sure that there are no commands still running before you start a new one.
Protocol error - Modbus_Slave (only communication modules that support Modbus)		
16#8380	CRC error	Checksum error of the Modbus frame. Check the communication partner.
16#8381	The function code is not supported or is not supported for broadcast.	Check the communication partner and make sure that a valid function code is sent.
16#8382	Invalid length information in the request frame	Select a suitable data length at the DATA_LEN parameter.
16#8383	Invalid data address in the request frame	Select a suitable value for the data address at the DATA_ADDR parameter.
16#8384	Invalid data value error in the request frame	Check the data value in the request frame of the Modbus master
16#8385	The diagnostic value is not supported by the Modbus slave (function code 08)	The Modbus slave only supports the diagnostic values 16#0000 and 16#000A.
Protocol error - Modbus_Master (only communication modules that support Modbus)		
16#8380	CRC error	Checksum error of the Modbus frame. Check the communication partner.
16#8381	Response frame from Modbus Slave with the following error message: The function code is not supported.	Check the communication partner and make sure that a valid function code is sent.
16#8382	Response frame from Modbus Slave with the following error message: Invalid length	Select a suitable data length.
16#8383	Response frame from Modbus Slave with the following error message: Invalid data address in the request frame	Select a suitable value for the data address at the DATA_ADDR parameter.
16#8384	Response frame from Modbus Slave with the following error message: Data value error	Check the request frame to the Modbus slave.
16#8385	Response frame from Modbus Slave with the following error message: The diagnostic value is not supported by the Modbus slave	The Modbus slave only supports the diagnostic values 16#0000 and 16#000A.

Error code	Description	Remedy
16#8386	The returned function code does not match the requested function code.	Check the response frame and the addressing of the slave.
16#8387	A slave that was not requested answers	Check the response frame of the slave. Check the address settings of the slave.
16#8388	Error in the response of the slave to a write request.	Check the response frame of the slave.
16#8828 [1]	DATA_PTR points to a bit address that is not equal to n * 8	Check the DATA_PTR pointer
16#8852 [1]	DATA_PTR is not a pointer to a DB or a bit memory area	Check the DATA_PTR pointer
16#8853 [1]	DATA_PTR is not a pointer of type BOOL or WORD	Check the DATA_PTR pointer
16#8855 [1]	DATA_PTR points to a write-protected DB	Check the DATA_PTR pointer
16#8856 [1]	Error during call of SFC51	Call the Modbus_Master instruction again
Error - Modbus_Slave (only communication modules that support Modbus)		
16#8428 [1]	MB_HOLD_REG points to a bit address that is not equal to n * 8	Check the MB_HOLD_REG pointer
16#8452 [1]	MB_HOLD_REG is not a pointer to a DB or a bit memory area	Check the MB_HOLD_REG pointer
16#8453 [1]	MB_HOLD_REG is not a pointer of type BOOL or WORD	Check the MB_HOLD_REG pointer
16#8454 [1]	The area addressed by MB_HOLD_REG is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the MB_HOLD_REG pointer
16#8455 [1]	MB_HOLD_REG points to a write-protected DB	Check the MB_HOLD_REG pointer
16#8456 [1]	Error during call of SFC51	Call the Modbus_Slave instruction again

[1] Only with instructions for S7-300/400 CPUs

Overview of error messages - USS

Error code	Description	Remedy
16#0000	No error	-
16#8180	Length error in response of drive	Check the response frame of the drive.
16#8181	Data type error	Select a suitable data type. The following are valid: • Real • Word • Double word
16#8182	Data type error: "Double word" or "Real" may not be returned for the "Word" request.	Check the response frame of the drive.
16#8183	Data type error: "Word" may not be returned for the "Double word" or "Real" request.	Check the response frame of the drive.
16#8184	Checksum error in response of drive	Check the drive and the communication connection.
16#8185	Addressing error	Valid drive address range: 1 to 16
16#8186	Setpoint error	Valid setpoint range: -200 % to +200 %
16#8187	Incorrect drive number returned	Check the response frame of the drive.

Error code	Description	Remedy
16#8188	Invalid PZD length	Permitted PZD lengths: 2, 4, 6, 8 words
16#8189	The module does not support this data transmission rate.	Select a valid data transmission rate for the module.
16#818A	A different request for this drive is currently active.	Repeat the parameter read or write command later.
16#818B	The drive does not respond.	Check the drive.
16#818C	The drive responds with an error message to a parameter request.	Check the response frame of the drive. Check the parameter request.
16#818D	The drive responds with an access error message to a parameter request.	Check the response frame of the drive. Check the parameter request.
16#818E	The drive was not initialized.	Check the user program and make sure that the USS_Drive_Control instruction is called for this drive.
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Port_Config.RDREC.STATUS, Send_Config.RDREC.STATUS , Receive_Config.RDREC.STATUS , Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Port_Config.WRREC.STATUS, Send_Config.WRREC.STATUS , Receive_Config.WRREC.STATUS , Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB WRREC.

[1] Only with instructions for S7-300/400 CPUs

8.1.4.6 Configurations for point-to-point connections (S7-300, S7-400)

Configuration for point-to-point links

Introduction

Overview of the components and interfaces

Contents

Table overview of the module variations and their functions.

For CP 340/341/440/441 and ET 200S 1SI

Module	Interface	Protocols						
		ASCII	3964(R)	RK512	Printer driver	Modbus master	Modbus slave	USS master
CP 340	RS232 C	X	X	-	X	-	-	-
CP 340	20 mA TTY	X	X	-	X	-	-	-
CP 340 [1]	RS422	X	X	-	X	-	-	-
	RS485	X	-	-	-	-	-	-
CP 341	RS232 C	X	X	X	X	X [2]	X [2]	-
CP 341	20 mA TTY	X	X	X	X	X [2]	X [2]	-
CP 341 [1]	RS422	X	X	X	X	X [2]	X [2]	-
	RS485	X	-	-	X	X [2]	X [2]	-
CP 440	RS422	X	X	-	-	-	-	-
	RS485	X	-	-	-	-	-	-
CP 441-1	RS232 C	X	X	-	X	-	-	-
CP 441-1	20 mA TTY	X	X	-	X	-	-	-
CP 441-1 [1]	RS422	X	X	-	X	-	-	-
	RS485	X	-	-	X	-	-	-
CP 441-2	RS232 C	X	X	X	X	X [2]	X [2]	-
CP 441-2	20 mA TTY	X	X	X	X	X [2]	X [2]	-
CP 441-2 [1]	RS422	X	X	X	X	X [2]	X [2]	-
	RS485	X	-	-	X	-	-	-
ET 200S 1SI for ASCII and 3964(R)	RS232 C	X	X	-	-	-	-	-
	RS422	X	X	-	-	-	-	-
	RS485	X	-	-	-	-	-	-
ET 200S 1SI for modbus and USS	RS232 C	-	-	-	-	X	X	X
	RS422	-	-	-	-	X	X	-
	RS485	-	-	-	-	X	X	X
CPU 313C-2 PtP	RS422	X	X	-	-	-	-	-
	RS485	X	-	-	-	-	-	-
CPU 314C-2 PtP	RS422	X	X	X	-	-	-	-
	RS485	X	-	-	-	-	-	-

[1] The difference between RS422 and RS485 is set in the configuration dialog.

[2] A loadable driver with a dongle is needed for the Modbus protocol.

Accompanying signals and data flow control

The RS232C accompanying signals can be controlled with the ASCII, Modbus master and Modbus slave protocols via the RS232C interface.

Data flow control with RTS/CTS is possible with the ASCII and printer driver protocols via the RS 232C interface.

Data flow control with XON/XOFF is possible with the ASCII and printer driver protocols via the RS232C, 20mA TTY and RS422 interfaces.

Overview of the instructions

Overview of the instructions

The communication between the CPU, communication module and a communication partner is performed via the instructions and the protocols of the communication module.

The instructions form the software interface between the CPU and the communication module. They must be called cyclically from the user program.

The communication protocols are implemented on the communication module. The protocol is used to adapt the interface of the communication module to the interface of the communication partner.

Table 8-64 Instructions for CP 340

Instruction	Meaning
P_SEND (Page 7564)	The instruction P_SEND allows you to send an entire range or a partial range of a data component to a communication partner.
P_RCV (Page 7567)	The instruction P_RCV allows you to receive data from a communication partner and to store them in a data block.
P_PRINT (Page 7570)	The P_PRINT instruction allows you to print a message text with up to 4 variables on a printer.
P_RESET (Page 7570)	The P_RESET instruction allows you to reset the internal receive buffer of the CP 340.
V24_STAT_340 (Page 7575)	The V24_STAT instruction allows you to read the signal states at the RS232C interface of the CP 340-RS232C.
V24_SET_340 (Page 7577)	The V24_SET instruction allows you to set/reset the outputs on the RS232C interface of the CP 340 RS232C.

Table 8-65 Instructions for CP 341

Instruction	Meaning
P_SND_RK (Page 7584)	You can use the instruction P_SND_RK to send the entire range or a partial range of a data block to a communication partner, or you can fetch data from the communication partner (using RK512).
P_RCV_RK (Page 7595)	You can use the P_RCV_RK instruction to receive data from a communication partner and store it in a data block or make data available to the communication partner (using RK512).
P_PRINT341 (Page 7606)	The instruction P_PRINT341 allows you to print a message text with up to 4 variables on a printer.
V24_STAT (Page 7575)	The V24_STAT instruction allows you to read the signal states at the RS232C interface of the CP 341-RS232C.
V24_SET (Page 7577)	The V24_SET instruction allows you to set/reset the outputs on the RS232C interface of the CP 341 RS232C.

Table 8-66 Instructions for CPU 313C-2 PTP

Instruction	Meaning
SEND_PTP (Page 7894)	With the instruction SEND_PTP, you can send the entire range or a partial range of a data block to a communication partner.
RCV_PTP (Page 7896)	With the instruction RCV_PTP, you can receive the data from a communication partner and store it in a data block.
RES_RCVB (Page 7896)	You can use the RES_RCVB instruction to reset the receive buffer of the module.

Table 8-67 Instructions for CPU 314C-2 PTP

Instruction	Meaning
SEND_PTP (Page 7894)	With the instruction SEND_PTP, you can send the entire range or a partial range of a data block to a communication partner.
RCV_PTP (Page 7896)	With the instruction RCV_PTP, you can receive the data from a communication partner and store it in a data block.
RES_RCVB (Page 7899)	You can use the RES_RCVB instruction to reset the receive buffer of the module.
SEND_RK (Page 7900)	You can use the SEND_RK instruction to send the entire range or a partial range of a data block to a communication partner using the RK512 protocol.
FETCH_RK	You can use the FETCH_RK instruction to request data from the communication partner using the RK512 protocol.
SERVE_RK (Page 7909)	You can use the SERVE_RK instruction to receive RK512 protocol data from a communication partner and store it in a data block or make data available to the communication partner.

Table 8-68 Instructions for CP 440

Instruction	Meaning
SEND_440 (Page 7623)	With the instruction SEND_440 , you can send the entire range or a partial range of a data block to a communication partner.
RECV_440 (Page 7625)	With the instruction RECV_440 , you can receive the data from a communication partner and store it in a data block.
RES_RECV (Page 7628)	With the instruction RES_RECV , you can reset the receive buffer of the CP 440.

Table 8-69 Instructions for CP 441

Instruction	Meaning
V24_STAT_441 (Page 7638)	The V24_STAT_441 instruction allows you to read the signal states on the RS232C interface of the CP 441.
V24_SET_441 (Page 7639)	The V24_SET_441 instruction allows you to set/reset the outputs on the RS232C interface of the CP 441 .
The automation system S7-400 provides a row of instructions that activate and control the communication in the user program between CPU and the communication processor CP 441. These instructions are stored in the memory of the CPU (see below in the section "Instructions from the automation system S7-400").	

Table 8-70 Instructions for ET 200S 1SI

Instruction	Meaning
S_SEND (Page 7667)	The instruction S_SEND allows you to send an entire range or a partial range of a data component to a communication partner.
S_RCV (Page 7663)	The instruction S_RCV allows you to receive data from a communication partner and to store them in a data block.
S_VSTAT (Page 7670)	The instruction S_VSTAT allows you to read the signal states on the RS 232C-interface of the component ET 200S 1SI.
S_VSET (Page 7671)	The instruction S_VSET allows you to set/reset the outputs on the RS 232C-interface of the module ET 200S 1SI.
S_XON (Page 7673)	With the instruction S_XON , you can set additional settings if the module was set for the flow control XON/XOFF.
S_RTS (Page 7674)	With the instruction S_RTS , you can set additional settings if the module was set for the flow control RTS/CTS.
S_V24 (Page 7676)	With the instruction S_V24 , you can set additional settings if the module was set for the automatic operation of the V.24 signals.
S_MODB (Page 7689)	The S_MODB instruction allows you to establish a communication connection to a Modbus master control system and implement the Modbus addresses in the SIMATIC memory areas.
S_USST (Page 7698)	The instruction S_USST processes the transfer of the network data (PZD and PKW data) to the slaves depending on the network data structure being used.
S_USSR (Page 7701)	The instruction S_USSR processes the reception of the network data (PZD and PKW data) from the slaves depending on the network data structure being used.
S_USSI (Page 7703)	If this instruction S_USSI is called upon starting up the S7-system, communication processor DB, network DB and parameter DB are created, which are required for communication.

Table 8-71 Instructions for Modbus

Instruction	Meaning
MODB_341 (Page 7653)	With the instruction MODB_341 and the affiliated driver, a communication connection between a modbus master control system and the communication module CP 341 "modbus-capable" slave system is allowed for.
MODB_441 (Page 7655)	With the instruction MODB_441 and the affiliated driver, a communication connection between a modbus master control system and the communication module CP 441-2 "modbus-capable" slave system is allowed for.

Instructions from the automation system S7-400

In the following table, you will find the instructions from the automation system S7-400, which are available for communication between CPU and CP 441.

Table 8-72 Instructions from the automation system S7-400

Instruction	Meaning
BSEND	With the instruction BSEND, you can send data from a S7-data range to a communication partner with a destination.
BRCV	With the instruction BRCV, you can receive the data from a communication partner and transfer it to a S7-data range.

Instruction	Meaning
GET	RK512 only: With the instruction GET, you can fetch the data from a communication partner.
PUT	RK512 only: With the instruction PUT, you can send the data to the communication partner with dynamic destination addresses..
PRINT	With the instruction PRINT, you can print a message text on a printer with up to four variables.
STATUS	With the instruction STATUS, you can call the status of a communication connection.

Further Information

A detailed description of the instructions from the automation system S7-400 and their settings can be found under: Overview of the instructions for S7 communication (Page 7356)

See also

P_RESET: Delete receive buffer (Page 7574)

S_RCV: receiving data (Page 7683)

S_SEND: Sending Data (Page 7686)

FETCH_RK: Fetch data (RK 512) (Page 7904)

Basic information

Serial transfer of a character

Introduction

To exchange data between two or more communication partners, there are different networking possibilities available. The point-to-point connection between two communication partners is the simplest case of information exchange.

Point-to-point connection

For the point-to-point connection, the communication module forms the interface between a programmable logic controller and a communication partner. Data is transferred serially with the communication module for the point-to-point connection.

The addressing mechanisms of the selected transmission procedure areimplemented on the communication module. Therefore, the point-to-point connection is closed on the communication module and not on the communication partner, as it is with other connections.

Serial data transmission

During serial data transmission, the individual bits from a byte, which must be transferred, are transmitted sequentially in a determined sequence.

Unidirectional/bidirectional data traffic

Data transmission with the communication partner is processed independently by the communication module via the serial interface. The communication module is equipped with various drivers to accomplish this.

- unidirectional data traffic:
  - Printer driver
- bidirectional data traffic:
  - ASCII driver
  - 3964(R) procedure
  - Computer connection RK512
  - Modbus/USS

Data transmission via the serial interface is processed by the communication module depending on the physical interface and the selected driver.

Unidirectional data traffic - printout

During the printout (printer driver), n bytes of user data are output to a printer. A reception of characters does not occur. Individual control characters for data flow control (for example, XON/XOFF) are exceptions to this.

Bidirectional data traffic - operating modes

With bi-directional data traffic, there are two different operating modes for the communication module:

- Half-duplex mode (3964 (R procedure), ASCII driver, RK512, Modbus/USS)
  The data is transferred alternating in both directions between one or multiple communication partners. Half-duplex mode means that everything is sent or received at one point in time. The exceptions to this can be individual control characters for data flow control (for example, XON/XOFF), which can also be received/sent during transmission/reception operation.
- Full-duplex mode (ASCII-driver)
  The data is exchanged simultaneously between one or more communication partners; it may be sent and received at one point in time. Each communication partner must be able to simultaneously operate a transmission and reception set-up.

Only half-duplex mode can be used with an X27 interface module (RS422/485) set to RS485 (2-wire).

Asynchronous data transmission

Serial data transmission is performed asynchronously with the communication module. The so-called time grid simultaneous operation (fixed time grid for the transfer of a fixed character string) must only be maintained during the transmission of a character. Each transferred character precedes a synchronization impulse, also referred to as a start bit. The length of the start bit transmission is determined by the clock. The end of the character transfer is formed by the stop bit.

Agreements

In addition the start and stop bits, there are other agreements between the two communication partners required for serial data transmission. This includes:

- Transmission speed (baud rate)
- Character and acknowledgement delay time
- Parity
- Number of data bits
- Number of stop bits

Character frame

The data between communication module and a communication partner is transmitted via the serial interface in a 10, 11 or 12 bit character frame. Several data formats are available for each character frame. Configure the desired format of the data transmission in the properties dialog of the module. Depending on the particular communication module, not all the theoretically possible character frames (combinations) are available.

10-Bit Character Frame

In the following illustration, three data formats of the 10-bit character frame are shown as an example.

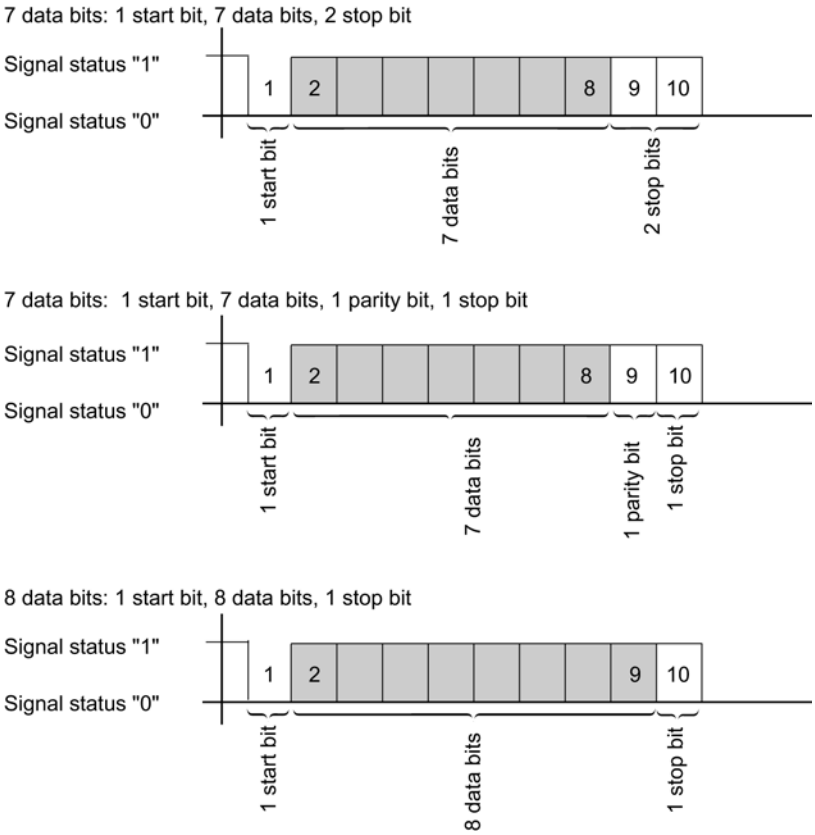


Figure 8-16 10-Bit Character Frame

11-Bit Character Frame

In the following illustration, three data formats of the 11-bit character frame are shown as an example.

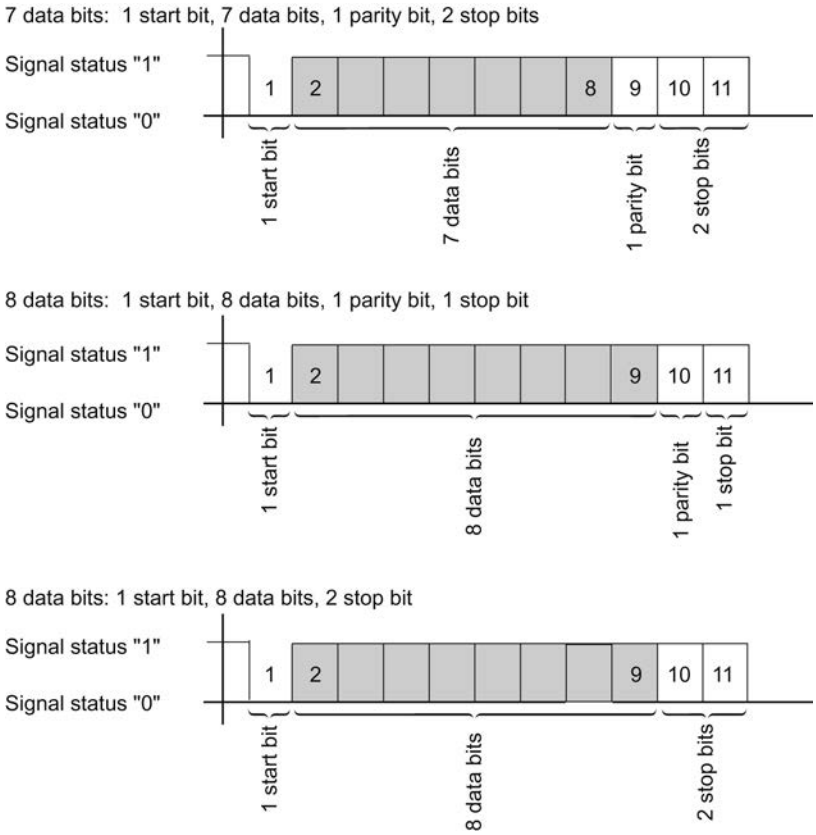


Figure 8-17 11-Bit Character Frame

Character Delay Time

The following illustration shows the maximum permissible interval between two received characters within a message frame (character delay time).

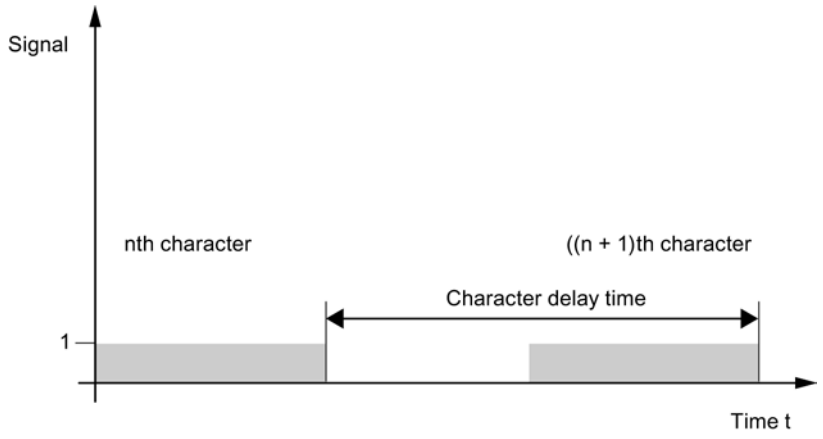


Figure 8-18 Character Delay Time

Transmission integrity

Transmission integrity plays an important role in the transmission of data and in selection of the transmission procedure. Generally speaking, the more layers of the reference model are applied, the greater the transmission integrity.

Classifying the supplied protocols

The figure below illustrates how these protocols of the communication module fit into the reference model:

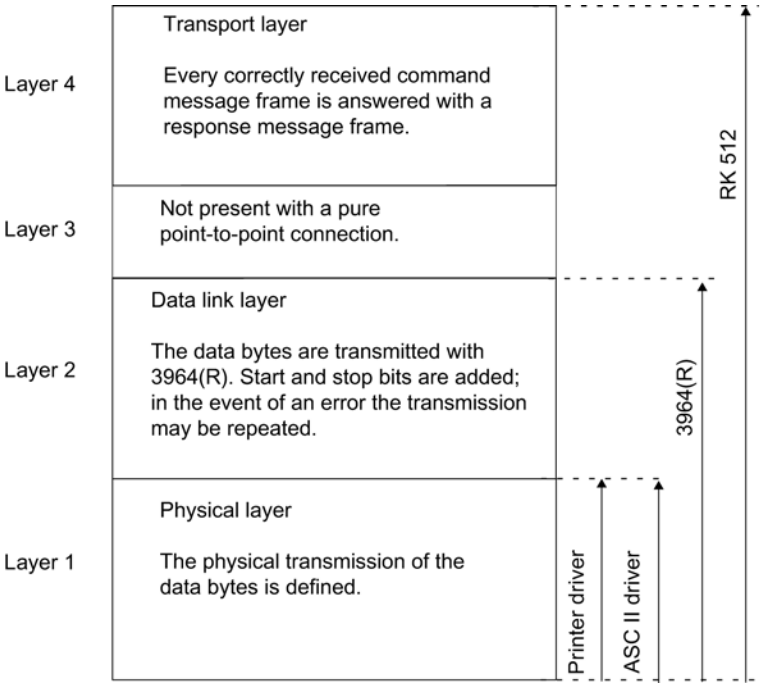


Figure 8-19 Classification of the existing protocols of the communication module in the reference model

Mapping of protocols to the communication modules

Protocol	CP 340	CP 341	CPU 313 C-2 PtP	CPU 314C-2 PtP	CP 440	CP 441	ET 200S 1SI
Printer driver	●	●				●	
ASCII driver	●	●	●	●	●	●	●
3964(R) procedure	●	●	●	●	●	●	●
Computer connection RK512		●		●		● 2)	
Modbus master		● 1)				● 1) 2)	●
Modbus slave		● 1)				● 1) 2)	●
USS-master							●

1) A loadable driver with a dongle is needed for the Modbus protocol.

2) only CP 441-2

Transmission Integrity with the printer driver

Data integrity when using the printer driver:

- No data integrity precautions are taken for data transport with the printer driver.
- To prevent data from being lost in the event of the printer receive buffer overflowing, you can work with data flow control (XON/XOFF, RTS/CTS).
- When data is output to the printer, the printer's BUSY signal is evaluated. The communication module receives the BUSY signal as a CTS signal and evaluates it in the same way (see ASCII driver). Note that, when using CTS/RTS flow control, you must set the polarity of the BUSY signal to CTS = "OFF" on the printer.

Transmission Integrity with the ASCII driver

Data integrity when using the ASCII driver:

- When data is transmitted via the ASCII driver, there are no data integrity precautions other than the use of a parity bit (can also be canceled, depending on how the character frame is set). This means that, although this type of data transport has a very efficient throughput rate, security is not guaranteed.
- Using the parity bit makes it possible to detect an inverted bit in a character that is to be transmitted. If two or more bits of a character are inverted, this error can no longer be detected.
- To increase transmission integrity, a checksum and length specification for a message frame can be employed. These measures must be implemented by the user.
- A further increase in data integrity can be achieved by means of acknowledgment message frames in response to send or receive message frames. This is the case with high-level protocols for data communication (ISO 7-layer reference model).

Transmission Integrity with 3964(R)

Enhanced data integrity through use of the 3964(R) procedure:

- The Hamming distance with the 3964(R) is 3. This measures the integrity of a data transmission.
- The 3964(R) procedure ensures high transmission integrity on the transmission line. This high transmission integrity is achieved by means of a specified message-frame setup and release as well as the use of a block check character (BCC).

Two different procedures for data transmission can be used, either with or without a block check character:

- Data transmission without a block check character: **3964**
- Data transmission with a block check character: **3964(R)**

In this manual, the designation 3964(R) is used when descriptions and notes refer to both data transmission procedures.

Transmission integrity with RK512

Very high data security using RK512:

- The Hamming distance with the RK512 and 3964R is 4. The Hamming distance is a measure of the integrity for a data transmission.
- Using the RK512 computer connection ensures high transmission integrity on the transmission line (because RK512 uses the 3964R procedure for data transport).
- Further processing at the communication partner is ensured (because the RK512 interpreter checks the additional length specification in the header and, after storing the data in the destination data area of the communication partner, generates a message frame acknowledging the success or failure of the data transport).
- The RK512 computer connection independently guarantees the correct use of the 3964R procedure and the analysis/addition of the length specification as well as the generation of the response telegram. There is no user handling! All you need to do is evaluate the positive/negative final acknowledgment.

Transmission integrity for Modbus/USS

Data security when using the Modbus/USS driver.

- The Hamming distance for Modbus/USS is 4. The Hamming distance is a measure of the integrity for a data transmission.
- Using Modbus/USS guarantees high transmission integrity on the transmission line (through a CRC16 block check for Modbus and an XOR block check for USS).
- Further processing in the communication partner is assured because both Modbus and USS evaluate additional length information in the header and generate a (time-monitored) acknowledgment message frame relating to the successful/unsuccessful data transmission.

Data transmission with ASCII driver

Data Transmission with the ASCII Driver

Assignment of the protocols to the communication modules, see: Overview of the components and interfaces (Page 1027)

Introduction

The ASCII driver controls data transmission via a point-to-point connection between the communication processor and a communication partner. This driver contains the physical layer (layer 1).

The structure of the message frames is left open through the S7 user passing on the complete send message frame to the communication processor. For the receive direction, the end criterion of a message must be configured. The structure of the send message frames may differ from that of the receive message frames.

The ASCII driver allows data of any structure (all printable ASCII characters as well as all other characters from 00 through FFH (with 8 data bit character frames) or from 00 through 7FH (with 7 data bit character frames) to be sent and received.

Sending Data with the ASCII Driver

Sending Data

For sending, you specify the number of user data bytes to be transmitted in the length parameter when you call the respective send instruction. For more information, see the section "Overview of the instructions (Page 1029)".

When you work with the end criterion "Character Delay Time" when receiving data, the ASCII driver pauses between two message frames when sending. You can call the send instruction at any time, but the ASCII driver only begins output when the time since the last message frame was sent is longer than configured character delay time.

Note

When XON/XOFF flow control is configured, the user data must not contain the configured XON or XOFF characters. The default settings are DC1 = 11H for XON and DC3 = 13H for XOFF.

If you work with the "end delimiter" criterion, you have a choice of three options:

- Send up to and including the end delimiter
  The end delimiter must be included in the data to be sent. Data is sent only up to and including the end delimiter, even if the data length specified in the instruction is longer.
- Send up to the length configured at the instruction
  Data is sent up to the length that is configured at the instruction. The last character must be the end delimiter.
- Send up to the length configured at the instruction and automatically append the end delimiter(s)
  Data is sent up to the length that is configured at the instruction. The end delimiter is automatically appended, in other words the end delimiters must not be included in the data to be sent. 1 or 2 characters more than the number specified at the instruction are sent to the partner, depending on the number of end delimiters.

When you work with the "fixed message frame length" end criterion, the amount of data transferred in the send direction is what you have specified for the length parameter of the instruction. The amount of data transferred in the receive direction, in other words, into the receive DB, is specified at the receiver using the "fixed message frame length" parameter in the properties dialog of the module. The two parameter settings must be identical, in order to ensure correct data traffic. A pause equal to the length of the character delay time (CDT) is inserted between two message frames when sending, to allow the partner to synchronize (recognize start of message frame).

If some other method of synchronization is used, the pause in sending can be deactivated in the properties dialog of the module.

Send operation

The figure below illustrates a send operation.

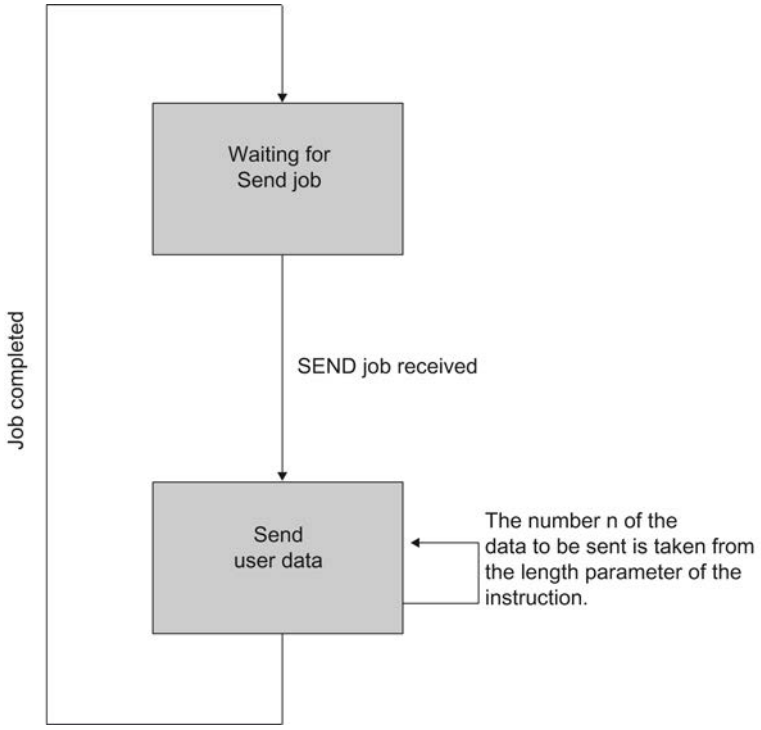


Figure 8-20 ASCII driver, sequence of sending data

Receiving Data with the ASCII Driver

Selectable end criteria

For data transmission using the ASCII driver you can choose between three different end criteria. The end criterion defines when a complete message frame is received. The possible end criteria are as follows:

- After character delay time elapses
  The message frame has neither a fixed length nor a defined end delimiter; the end of the message is defined by a pause on the line (After character delay time elapses).
- On receipt of the end delimiter(s)
  The end of the message frame is marked by one or two defined end delimiters.
- On receipt of fixed number of characters
  The length of the receive message frames is always identical.

Code transparency

The code transparency of the procedure depends on the choice of configured end criterion and flow control:

- With one or two end delimiters
  - not code-transparent
- When end criterion is character delay time or fixed message frame length
  - code-transparent
- Code-transparent operation is not possible when the flow control XON/XOFF is used.

Code-transparent means that any character combinations can occur in the user data without the end criterion being recognized.

End criterion "After character delay time elapses"

When data is received, the end of the message frame is recognized when the character delay time expires. The received data is accepted from the CPU.

In this case, the character delay time must be set so that it reliably expires between two consecutive message frames. But it should be long enough so that the end of the message frame is not falsely identified whenever the partner in the connection takes a send pause within a message frame.

The figure below illustrates a receive operation with the end criterion "After character delay time elapses".

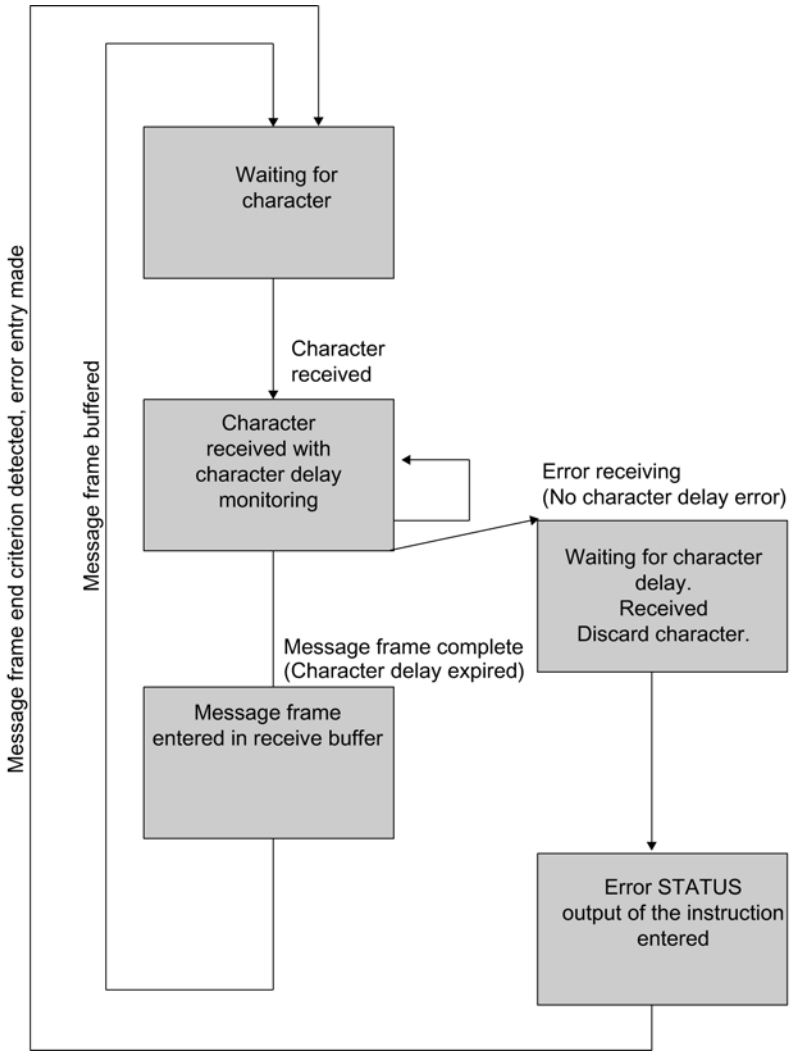


Figure 8-21 Sequence of Receive Operation with End Criterion "Expiration of Character Delay Time"

End criterion end delimiter

When data is received, the end of the message frame is recognized when the configured end delimiter(s) arrive. The received data including the end delimiter(s) is accepted from the CPU.

If the character delay time expires while the message frame is being received, the receive operation is terminated. An error message is issued and the message frame fragment is discarded.

If you are working with end delimiters, transmission is not code-transparent, and you must make sure that the end code(s) are not in the user data of the user.

Note the following when the last character in the received message frame is not the end delimiter.

- End delimiter elsewhere in the message frame:
  All characters including the end delimiter are entered in the receive DB. The characters following the end delimiter
  - is discarded if the character delay time (CDT) expires at the end of the message frame.
  - is merged with the next message frame if a new message frame is received before the character delay time expires.
- End delimiter not included in message frame:
  The message frame
  - is discarded if the character delay time (CDT) expires at the end of the message frame.
  - is merged with the next message frame if a new message frame is received before the character delay time expires.

The figure below illustrates a receive operation with the end criterion "end delimiter".

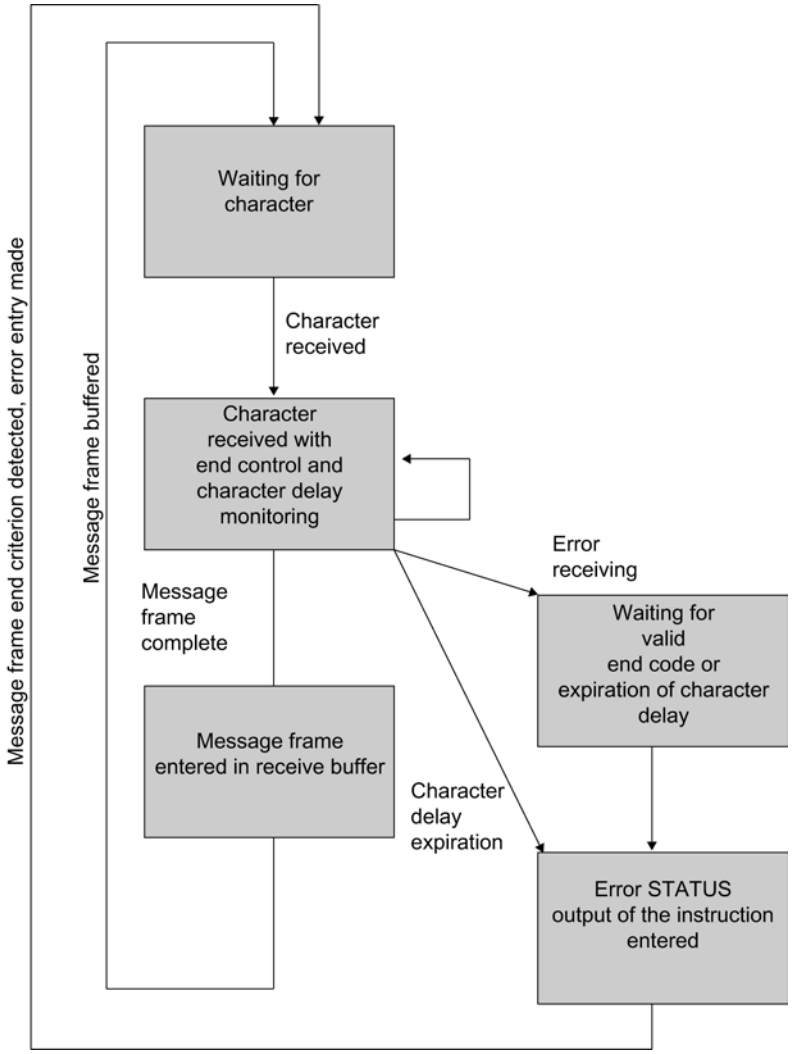


Figure 8-22 Sequence of receive operation with "End-of-Text Character" end criterion

End criterion fixed message frame length

When data is received, the end of the message frame is recognized when the configured number of characters has arrived. The received data is accepted from the CPU.

If the character delay time expires before the configured number of characters has been reached, the receive operation is terminated. An error message is issued and the message frame fragment is discarded.

Note the following if the message frame length of the received characters does not match the configured fixed message frame length:

- Message frame length of received characters greater than configured fixed message frame length:
  All characters received after the configured fixed message frame length is reached
  - is discarded if the character delay time (CDT) expires at the end of the message frame.
  - is merged with the next message frame if a new message frame is received before the character delay time expires.
- Message frame length of received characters less than configured fixed message frame length:
  The message frame
  - is discarded if the character delay time (CDT) expires at the end of the message frame.
  - is merged with the next message frame if a new message frame is received before the character delay time expires.

The figure below illustrates a receive operation with the end criterion "fixed message frame length".

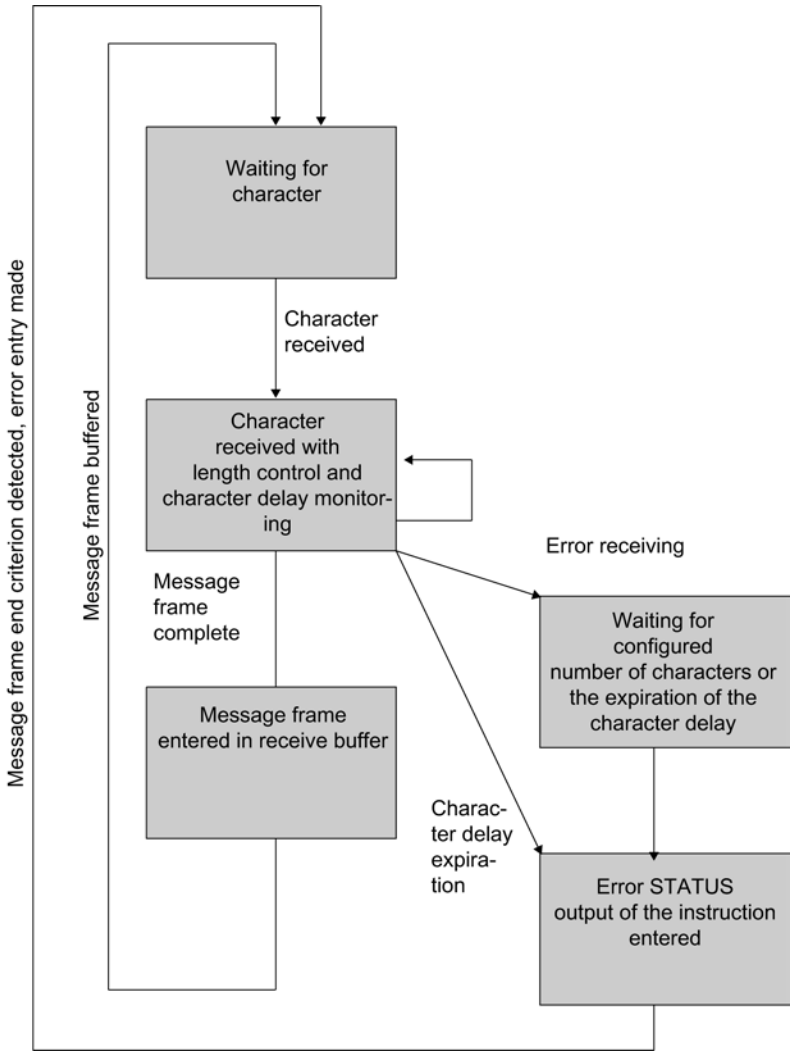


Figure 8-23 Sequence of receive operation with "Fixed Message Frame Length" end criterion

Receive buffer of the module

The receive buffer of the module can be up to up to 4096 bytes, depending on the communication module.

During configuration, you can specify if:

- CP and CPU 31xC-2 PtP
  the receive buffer should be deleted at startup or overwriting of data in the receive buffer should be prevented. You can also specify the value range (1 to 250) for the number of buffered receive message frames.
- ET 200S 1SI
  the receive buffer should be deleted on startup and if an overwriting of the data in the receive buffer should be avoided. Additionally, you can activate or block the buffering of received message frames.

The receive buffer on the module is a ring buffer:

- If several message frames are entered in the receive buffer of the module, the following applies: it is always the oldest message frame that is sent from the module to the CPU.
- If you only want to transfer the newest message frame, you must:
  - CP and CPU 31xC-2 PtP
    set "1" as the number of buffered message frames and disable the overwrite protection.
  - ET 200S 1SI
    block the dynamic message frame and disable the overwrite protection.

Note

If constant reading of the received data from the user program is interrupted for a time, you may find that when the received data is requested again, the CPU first receives old message frames from the module before it receives the most recent one.

During this interruption, the old message frames were either on the way between the module and CPU or had already been received by the instruction.

RS485 mode (CP 34x, 44x, CPU 31xC-2 PtP + ET 200S 1SI)

RS485 mode

When you run the ASCII driver in RS485 mode (half-duplex, two-wire mode), you must take steps in the user program to ensure that only one user sends data at any one time. If two users send data simultaneously, the message frame is corrupted.

Switch-over Times for RS485 Module in Half-Duplex Mode

The maximum switch-over time between sending and receiving is 1 ms.

RS232 mode (CP 34x, 44x, CPU 31xC-2 PtP + ET 200S 1SI)

RS232C accompanying signals

The following RS232C accompanying signals are available on the modules with the RS232C interface:

DCD	(input)	Data carrier detect
DTR	(Output)	Data terminal ready; module ready for operation
DSR	(input)	Data set ready; communication partner ready for operation
RTS	(Output)	Request to send, module ready to send

CTS	(input)	Clear to send; communication partner can receive data from the module (response to RTS = ON of the communication module)
RI	(input)	Ring indicator,

When the module is switched on, the output signals are in the OFF state (inactive).

You can configure the way in which the DTR/DSR and RTS/CTS control signals are used in the properties dialog of the module or control them via instructions in the user program.

Operate RS232C accompanying signals

The RS232C accompanying signals can be used as follows:

- When automatic use of all RS232C accompanying signals is configured
- When data flow control (RTS/CTS) is configured
- via the instructions V24_STAT and V24_SET

Note

When automatic operation of the R232C accompanying signals is configured, neither RTS/CTS data flow control nor RTS and DTR control by means of the V24_SET instruction are possible! When RTS/CTS data flow control is configured, RTS control by means of the V24_SET is not possible! On the other hand, it is always possible to read all RS232C accompanying signals using the V24_STAT instruction.

The sections that follow describe how the control and evaluation of the RS232C accompanying signals is handled.

Automatic use of accompanying signals

Automatic operation of the RS232C accompanying signals on the communication module is realized as follows:

- As soon as the module is switched by configuration to an operating mode with automatic operation of the RS232C accompanying signals, it sets RTS line to OFF and DTR line to ON (communication module ready for use).
  This prevents sending and receiving of message frames until the DTR line is set to ON. As long as DTR remains set to OFF, no data is received via the RS232C interface. If a send job is made, it is aborted with a corresponding error message.
- When a send job is made, RTS is set to ON and the configured data output waiting time starts. When the data output time elapses and CTS = ON, the data is sent via the RS232C interface.
- If the CTS line is not set to ON within the data output time so that data can be sent, or if CTS changes to OFF during transmission, the send job is aborted and an error message generated.
- Once the data has been sent and the configured clear RTS time has elapsed, the RTS line is set to OFF. The CP does not wait for CTS to change to OFF.

- Data can be received via the RS232C interface as soon as the DSR line is set to ON. If the receive buffer of the communication module threatens to overflow, there is no response from the communication module.
- If DSR changes from ON to OFF, an active send job as well as the receipt of data will be canceled with an error message. The message "DSR = OFF (automatic use of V24 signals)" is entered in the diagnostic buffer of the module .

Note

When automatic operation of the R232C accompanying signals is configured, neither RTS/CTS data flow control nor RTS and DTR control by means of the V24_SET instruction are possible!

The "Clear RTS time" must be set in the properties dialog of the module so that the communication partner can receive the last characters of the message frame in their entirety before RTS, and thus the send job, is revoked. The "data output waiting time" must be set so that the communication partner can be ready to receive before the time elapses.

Time Diagram

The figure illustrates the chronological sequence of a send job.

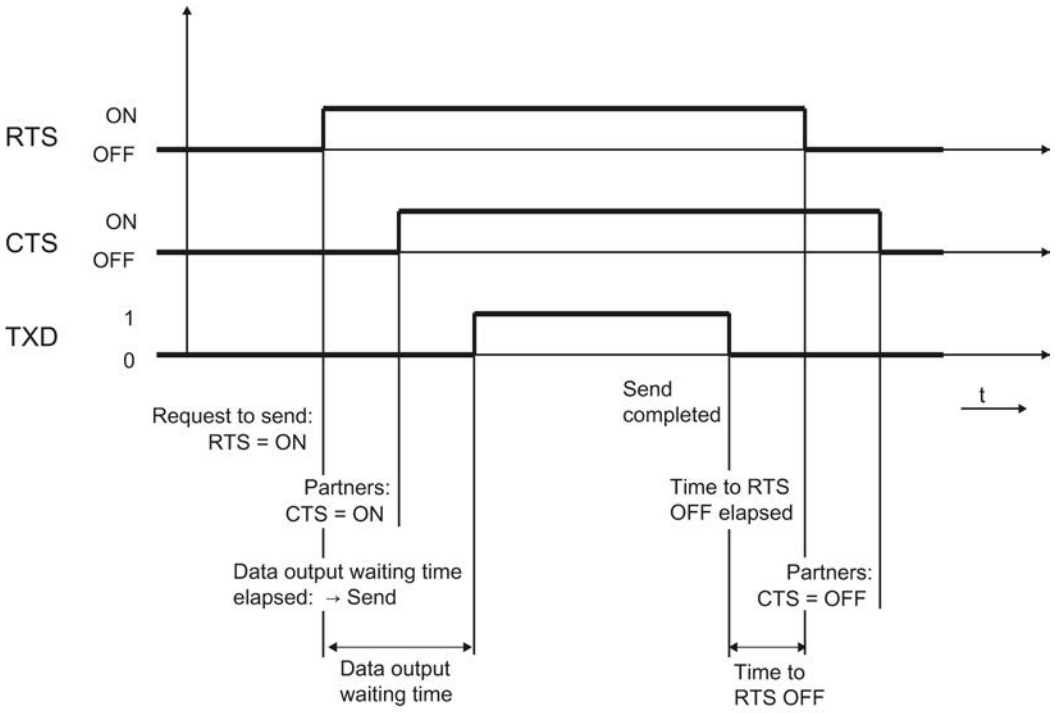


Figure 8-24 Time diagram of automatic use of the RS232C accompanying signals

Tasks of the instruction command V24_STAT/SET

The V24_STAT instruction command allows the status of each RS232C accompanying signal to be determined. The V24_SET allows the DTR and RTS output signals to be controlled.

Software/hardware handshake

Introduction

Handshake protocol to control the flow of data between two communication partners. The use of the handshake procedure prevents data from getting lost during transmission when the two devices are operating at different rates.

Handshake procedure

There are essentially two types of handshaking here:

- Software handshaking (e.g. XON/XOFF)
- Hardware handshaking (e.g. RTS/CTS)

Data flow control is implemented as follows on the communication module:

- As soon as the module is switched by configuration to an operating mode with flow control, it sends the XON character or it sets the RTS line to ON.
- When the configured number of message frames is reached, or alternatively 50 characters before the receive buffer overflows (size of the receive buffer: 4096 bytes), the module sends XOFF character or sets the RTS line to OFF. If the communication partner continues to send data regardless of this, the receive buffer overflows and an error message is generated. The data received in the last message frame is discarded.
- As soon as a message frame is retrieved by the S7-CPU and the receive buffer is ready to receive, the module sends the XON character or sets the RTS line to ON.
- If the module receives the XOFF character, or the CTS control signal is set to OFF, the module interrupts the transmission. If neither an XON character is received nor CTS is set to ON after a configured time has elapsed, the transmission is aborted and an appropriate error message (0708H) is entered in the error message area of the module.

Note

When RTS/CTS data flow control is configured, you must fully wire the interface signals in the plug connection. When RTS/CTS data flow control is configured, RTS control by means of the V24_SET is not possible!

Data transmission with 3964(R)

Data Transmission with the 3964(R) Procedure

Assignment of the protocols to the communication modules, see: Overview of the components and interfaces (Page 1027)

Introduction

The 3964(R) procedure controls point-to-point data exchange between the communication processor and a communication partner. As well as the physical layer (layer 1), the 3964(R) procedure also incorporates the data-link layer (layer 2).

Startup

The figure below illustrates the start-up of the 3964(R) procedure.

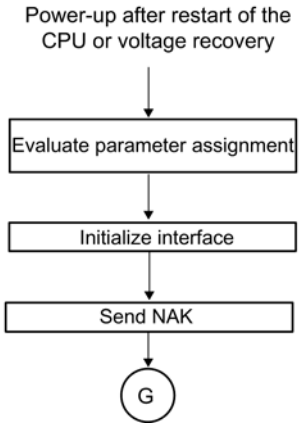


Figure 8-25 Flow diagram of the start-up of the 3964(R) procedure

Control characters

Introduction

The RK 512 computer connection provides a very high degree of data integrity. During data transmission, the 3964(R) procedure adds control characters to the information data (data-connection layer). These control characters allow the communication partner to check whether the data has arrived complete and without errors.

The Control Characters of the 3964(R) Procedure

The 3964(R) procedure analyzes the following control codes:

- **STX**Start of Text; start of character string for transfer
- **DLE**Data Link Escape; data connection escape
- **ETX**End of Text; end of character string for transfer

- **BCC**Block Check Character (3964(R) only)
- **NAK**Negative Acknowledge

Note

If DLE is transmitted as an information string, it is sent twice so that it can be distinguished from the control code DLE during connection setup and release on the send line (DLE duplication). The receiver then reverses the DLE duplication.

Priority

With the 3964(R) procedure, one communication partner must be assigned a higher priority and the other partner a lower priority. If both partners begin connection setup at the same time, the partner with the lower priority will defer its send request.

Block Checksum

Block Checksum

With the 3964(R) transmission protocol, data integrity is increased by the additional sending of a block check character (BCC).

Message frame:

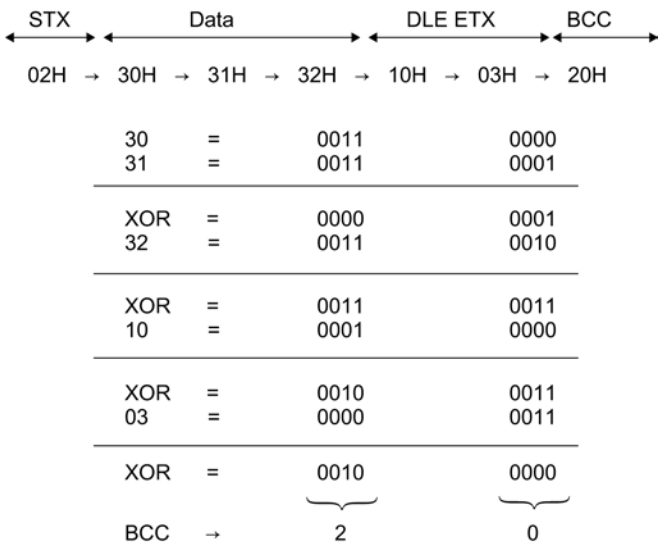


Figure 8-26 Block Checksum

The block checksum is the even longitudinal parity (EXOR operation on all data bytes) of a sent or received block. Its calculation begins with the first byte of user data (first byte of the

message frame) after the connection setup, and ends after the DLE ETX code on connection release.

Note

If DLE duplication occurs, the DLE code is accounted for twice in the BCC calculation.

Sending data with 3964(R)

Process of Data Transmission when Sending

The figure below illustrates the transmission sequence when data is sent with the 3964(R) procedure.

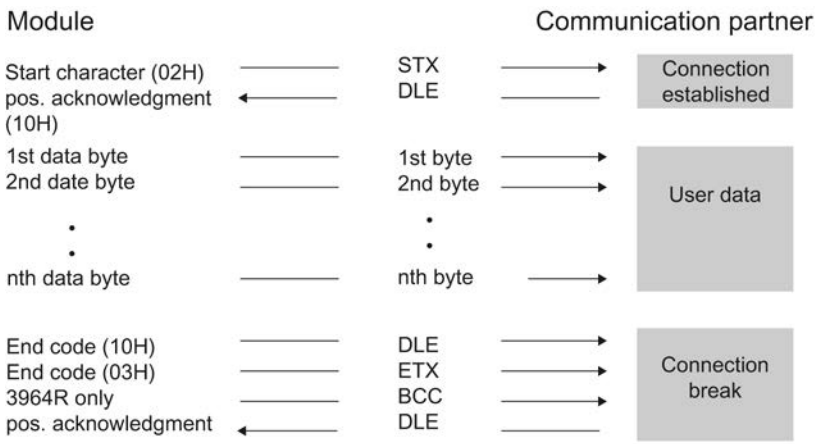


Figure 8-27 Data traffic when sending with the 3964(R) procedure

Establishing a Send Connection

To establish the connection, the 3964(R) procedure sends the control code STX. If the communication partner responds with the DLE code before the acknowledgment delay time expires, the procedure switches to send mode.

If the communication partner answers with NAK or with any other control code (except for DLE or STX), or the acknowledgment delay time expires without a response, the procedure repeats the connection setup. After the defined number of unsuccessful setup attempts, the procedure aborts the connection setup and sends the NAK code to the communication partner. The communication module enters a corresponding error number in the error message area.

Sending Data

If a connection is successfully established, the user data contained in the output buffer of the communication processor is sent to the communication partner with the chosen transmission parameters. The partner monitors the times between incoming characters. The interval between two characters must not exceed the character delay time.

If the communication partner sends the NAK control code during an active send operation, the procedure aborts its transmission of the block and tries again as described above, beginning with connection setup. If a different code is sent, the procedure first waits for the character delay time to expire and then sends the NAK code to change the mode of the communication partner to idle. Then the procedure starts to send the data again with the connection setup STX.

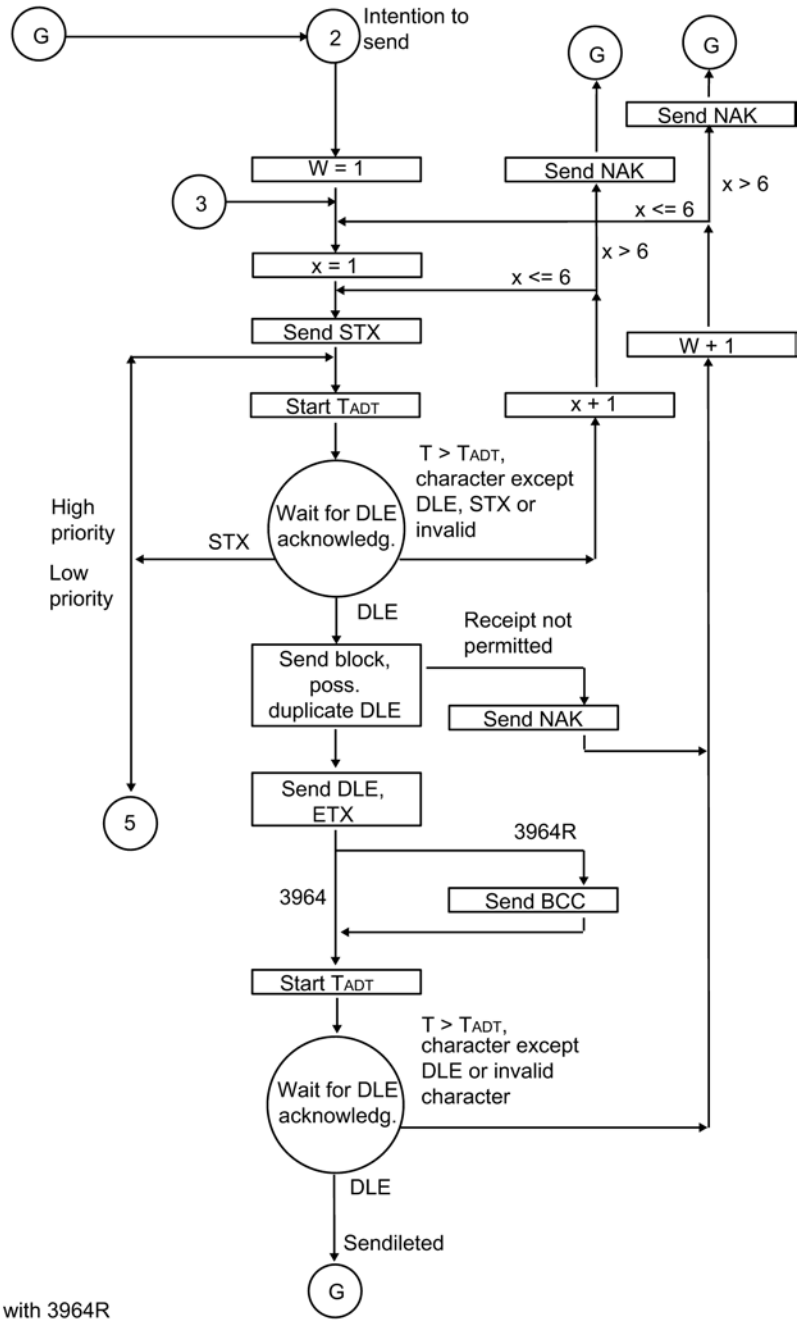
Releasing a Send Connection

Once the contents of the buffer have been sent, the procedure adds the codes DLE, ETX and with the 3964R only the block checksum BCC as the end identifier, and waits for an acknowledgment code. If the communication partner sends the DLE code within the acknowledgment delay time, the data block has been received without errors. If the communication partner responds with NAK, any other code (except DLE), or a damaged code, or if the acknowledgment delay time expires without a response, the procedure starts to send the data again with the connection setup STX.

After the defined number of attempts to send the data block, the procedure stops trying and sends an NAK to the communication partner. The communication module reports the error in the error message area.

Sending with the 3964(R) Procedure

The figure below illustrates sending with the 3964(R) procedure.



BCC only with 3964R
x = setup attempt count
T_{ADT} = 500 ms (3964R T_{ADT}=2s)
W = transmission attempt count
Immediate return to initial state at line break (BREAK)

Figure 8-28 Flow diagram of sending with the 3964(R) procedure

Receiving data with 3964(R)

Process of Data Transmission when Receiving

The figure below illustrates the transmission sequence when data is received with the 3964(R) procedure.

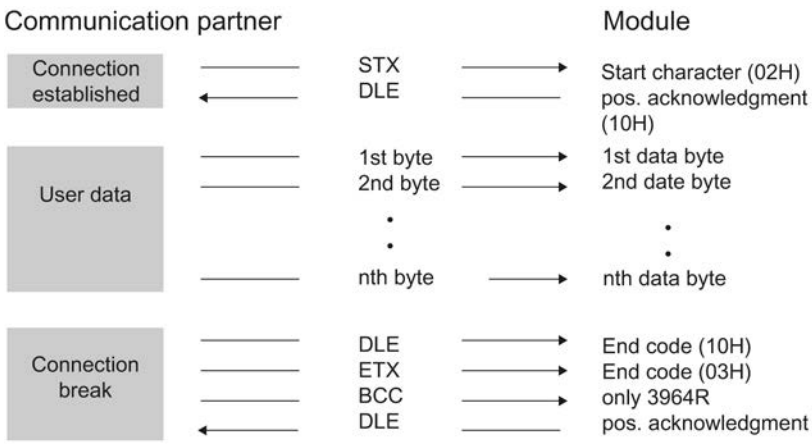


Figure 8-29 Data traffic when receiving with the 3964(R) procedure

Note

As soon as it is ready, the 3964(R) procedure sends a single NAK to the communication partner to set the latter to idle.

Establishing a Receive Connection

In idle mode, when there is no send job to be processed, the procedure waits for the communication partner to establish the connection.

If no empty receive buffer is available during a connection setup with STX, a wait time of 400 ms is started. If, after this time, there is no empty receive buffer, the system program reports the error in the error message area of the module. and the procedure sends a NAK and returns to idle mode. Otherwise, the procedure sends a DLE and receives the data as described above.

If the idle procedure receives any control code except for STX or NAK, it waits for the character delay time to expire, then sends the code NAK. The communication module reports the error in the error message area of the module.

receiving data

After a successful connection setup, the receive characters that are arrive are stored in the receive buffer. If two consecutive DLE codes are received, only one of these is stored in the receive buffer.

After each receive character, the procedure waits out the character delay time for the next character. If this period expires before another character is received, an NAK is sent to the communication partner. The communication module reports the error in the error message area of the module. The 3964(R) procedure does not initiate a repetition.

If transmission errors occur during receiving (lost character, frame error, parity error, etc.), the procedure continues to receive until the connection is closed. NAK is then sent to the communication partner. A repetition is then expected. If the block still cannot be received after the number of transmission attempts defined in the static parameter set, or if the communication partner does not start the repetition within a block wait time of 4 seconds, the procedure aborts the receive operation. The communication module reports the first transmission error and the final abort in the error message area of the module.

Releasing a Receive Connection

When the 3964 procedure detects a DLE ETX character string, it ends the receiving operation and confirms the successfully received block by sending a DLE signal to the communication partner. When errors are found in the received data, it outputs a NAK signal to the communication partner. A repetition is then expected.

If the 3964R procedure recognizes the string DLE ETX BCC, it stops receiving and compares the received block check character with the longitudinal parity calculated internally. If the BCC is correct and no other receive errors have occurred, the CP sends the code DLE to the communication partner. If the BCC is correct and no other receive errors have occurred, the 3964R procedure sends a DLE and returns to idle mode. If the BCC is faulty or a different receiving error occurs, an NAK is sent to the communication partner. A repetition is then expected.

Receiving with the 3964(R) Procedure

The figure below illustrates receiving with the 3964(R) procedure.

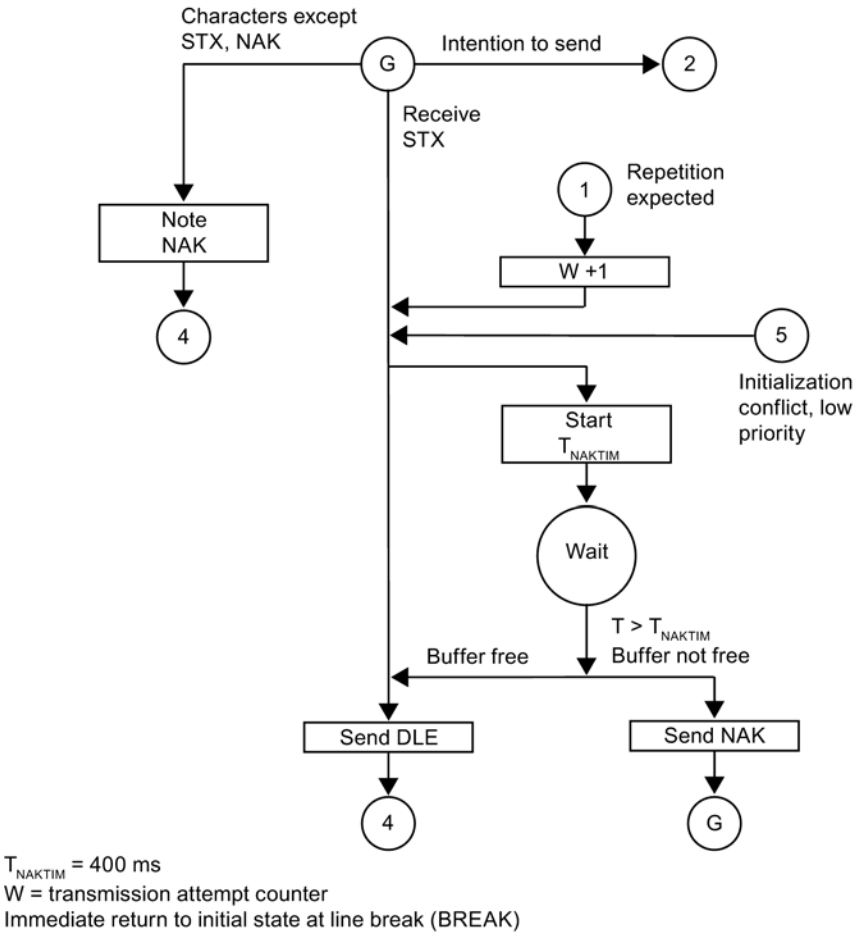


Figure 8-30 Flow diagram of receiving with the 3964(R) procedure (part 1)

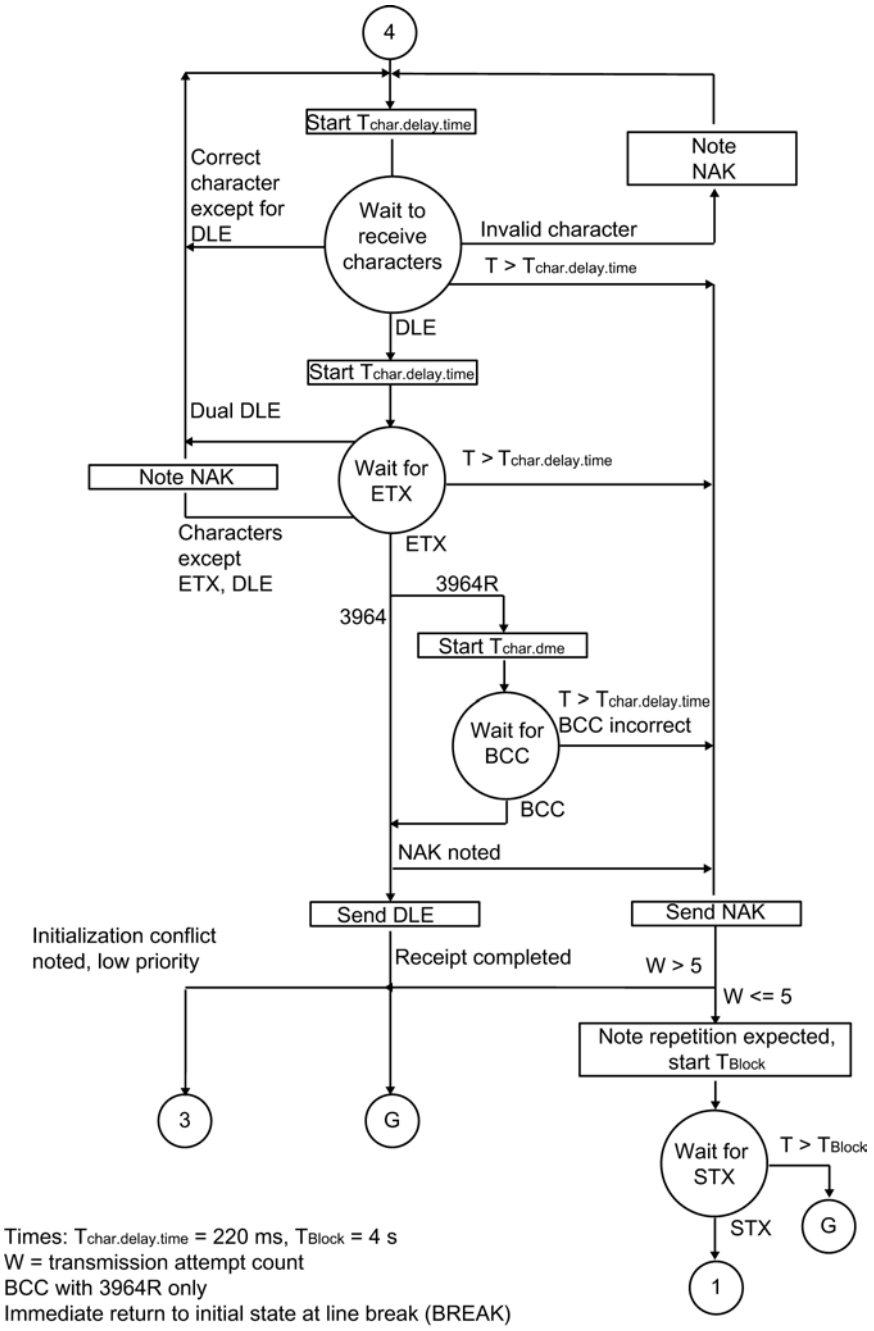


Figure 8-31 Flow diagram of receiving with the 3964(R) procedure (part 2)

Data transmission with Rk512

Data transmission with the RK512 computer connection

Assignment of the protocols to the communication modules, see: Overview of the components and interfaces (Page 1027)

Introduction

The RK512 computer connection controls data transmission in a point-to-point connection between the communication module and a communication partner.

Unlike the 3964(R) procedure, the RK512 computer connection includes not only the physical layer (layer 1), and the data link layer (layer 2), but also the transport layer (layer 4) of the ISO reference model. The RK512 computer connection also offers greater data security and enhanced addressing options.

Response message frame

The RK512 computer connection answers each correctly received command message frame with a response message frame to the client (transport layer). This allows senders to check whether their data has arrived undamaged at the CPU of the communication partner or whether the data they require is available on the CPU of the communication partner.

Command message frame

Command message frames are either SEND frames or FETCH frames.

SEND frame

A SEND frame is created when the communication module sends a command message frame with user data, and the communication partner replies with a response message frame without user data.

FETCH frame

A FETCH frame is created when the communication module sends a command message frame without user data, and the communication partner replies with a response message frame with user data.

Continuation message frame

If the volume of data exceeds 128 bytes, SEND and FETCH frames are automatically sent with continuation message frames.

Message frame header

With RK512, each message frame begins with a message frame header. It can contain message frame IDs, information on the data destination and source and an error number.

Structure of the message frame header

The table below indicates the structure of the header of the command message frame.

Table 8-73 Format of the message frame header for the command message frame (RK512)

Byte	Meaning
1	Message frame ID in command message frames (00H), in continuation command message frames (FFH)
2	Message frame ID (00H)
3	'A' (41H) for SEND job with destination DB or 'O' (4FH) for SEND job with destination DX or 'E' (45H) for FETCH job
4	Data to be transmitted consist of (only 'D' possible for sending): 'D' (44H)=data block 'I' (45H) = Input bytes 'O' (41H) = output bytes 'M' (4DH)=flag bytes 'C' (5AH)= counter cells 'T' (54H) = time cells
5 and 6	Data destination of SEND/PUT job or data source of GET job e.g. byte 5 = DB no., byte 6 = DW no. (RK512 addressing describes data source and destination with word limits. Conversion to byte addresses in SIMATIC S7 is automatic.)
7 and 8	Length of high byte Length of data to be transmitted according to type in bytes or Length of low byte words
9	Byte number of the communication flag; FFH is shown here if you have not specified an interprocessor communication flag.
10	**Bit 0 to 3:** Bit number of the communication flag, The protocol enters FH here if you have not specified an interprocessor communication flag. **Bit 4 to 7:** CPU number (number from 1 to 4); If you have not specified a CPU number but you have specified an interprocessor communication flag, OH is displayed here; if you specified neither a CPU number nor an interprocessor communication flag, FH is shown here.

The letters in bytes 3 and 4 are ASCII characters.

The header of the continuation command ^message frame consists of bytes 1 to 4 only.

Response message frame

Once the command message frame has been transmitted, the RK512 expects a response message frame from the communication partner within the monitoring time. The length of the monitoring time is 20 seconds irrespective of the transmission speed (baud rate).

Depending on the PtP module used, this monitoring time can be reduced by user configuration in the properties dialog of the module. By selecting the "transmission-rate dependent" option, monitoring is performed with the following maximum waiting times:

Table 8-74 Transmission-rate dependent monitoring time for response telegram

Transmission rate	Monitoring time
• 300 baud	10 s
• 600 baud	7 s
• 1200 baud	5 s
• from 38400 baud	3 s

The "grayed" "Maximum wait time" field is used only for showing the configured monitoring time and cannot be edited!

Structure and contents of the response message frame

The response message frame consists of 4 bytes and contains information on the progress of the job.

Byte	Meaning
1	Message frame ID in response message frames (00H), in continuation response message frames (FFH)
2	Message frame ID (00H)
3	Displays 00H
4	Error number of the communication partner in the response message frame: • **00H** if transmission was error-free • **> 00H** error number The error number in the response message frame automatically causes an error number to be entered in the error message area of the module.

Sending data with RK512

Process of Data Transmission when Sending

The figure below shows the transmission sequence when sending data with a response message frame using the RK512 computer connection.

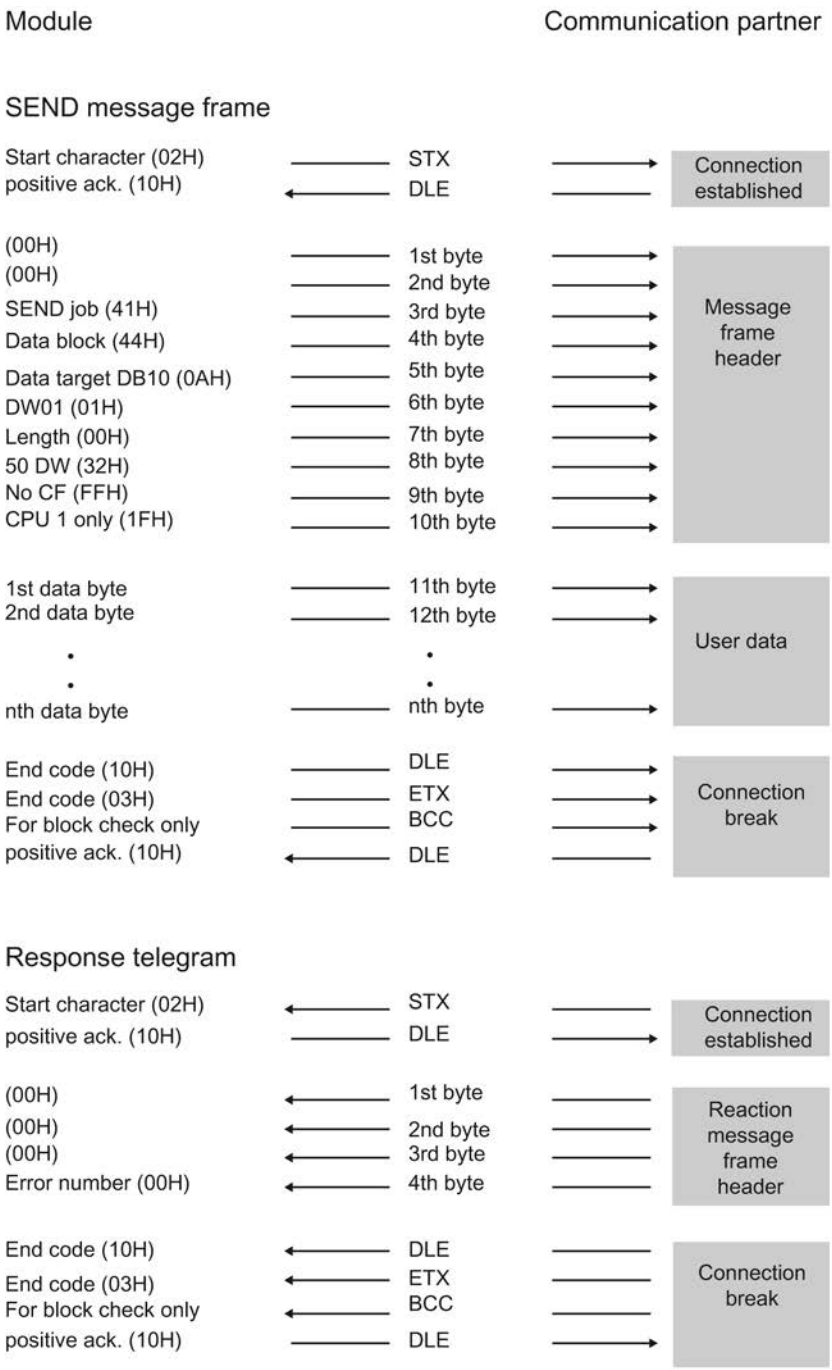


Figure 8-32 Data traffic when sending with a response frame

Sending Data

The SEND job is executed in the following sequence:

- Active partner
  Sends a SEND message frame. This contains a message frame header and data.
- Passive partner
  Receives the message frame, checks the header and the data, and acknowledges it with a response frame after passing the data on to the CPU.
- Active partner
  Receives the response message frame and terminates the SEND job.

Note

If the user data of the SEND job is not received correctly by the CPU or an error occurs in the message frame header, the communication partner enters an error number in the 4th byte of the response message frame. This does not apply when protocol errors occur.

Continuation SEND message frame

A continuation SEND job is started when the amount of data exceeds 128 bytes. The procedure is similar to the SEND job.

If more than 128 bytes are sent, the extra bytes are automatically transmitted in one or more continuation message frames. Each continuation SEND frame is acknowledged with a continuation response message frame.

The figure below shows the data transmission sequence when sending a continuation SEND message frame with a continuation response message frame.

The continuation response message frame for the last continuation SEND frame is only sent after successful transfer of the entire user data to the CPU.

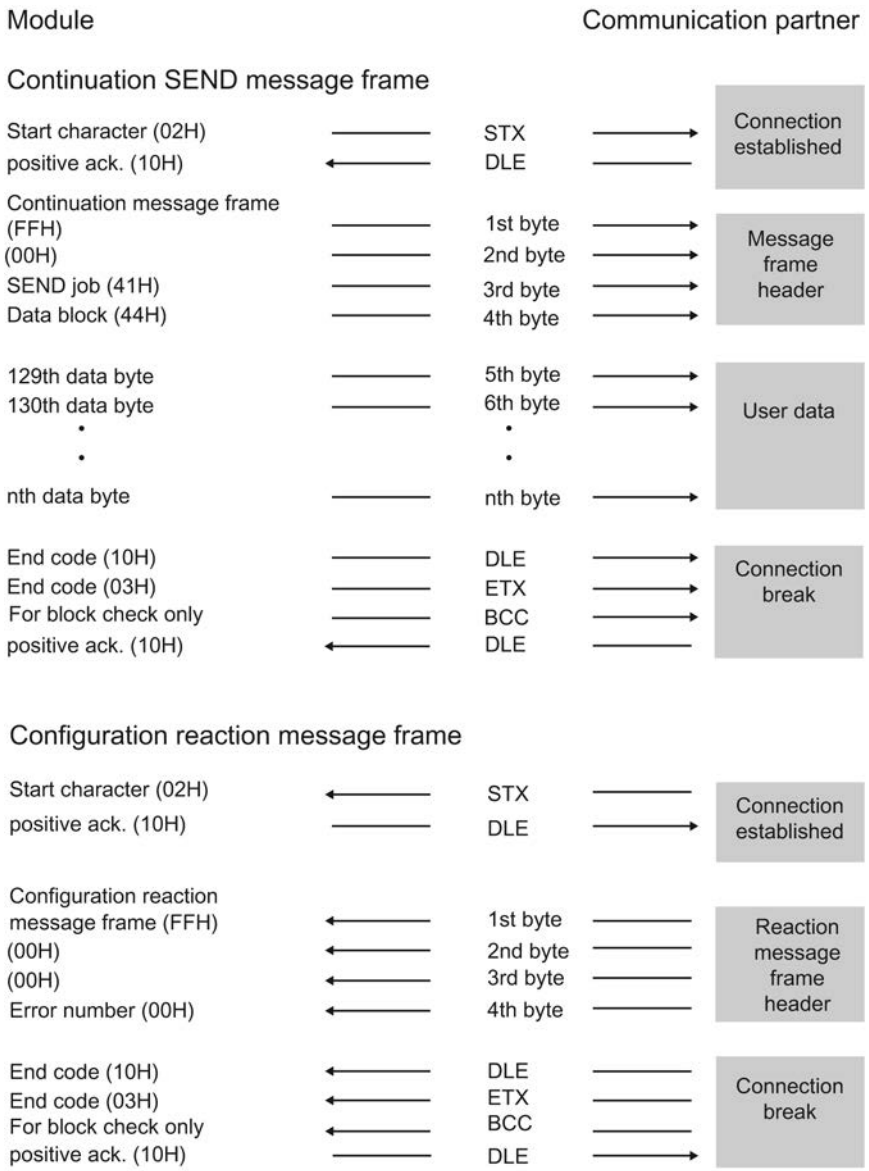


Figure 8-33 Sequence of a continuation SEND message frame with a continuation response message frame

Fetching data with RK512

Procedure for fetching data with RK512

The figure below shows the transmission sequence when fetching data with a response message frame using the RK512 computer connection.

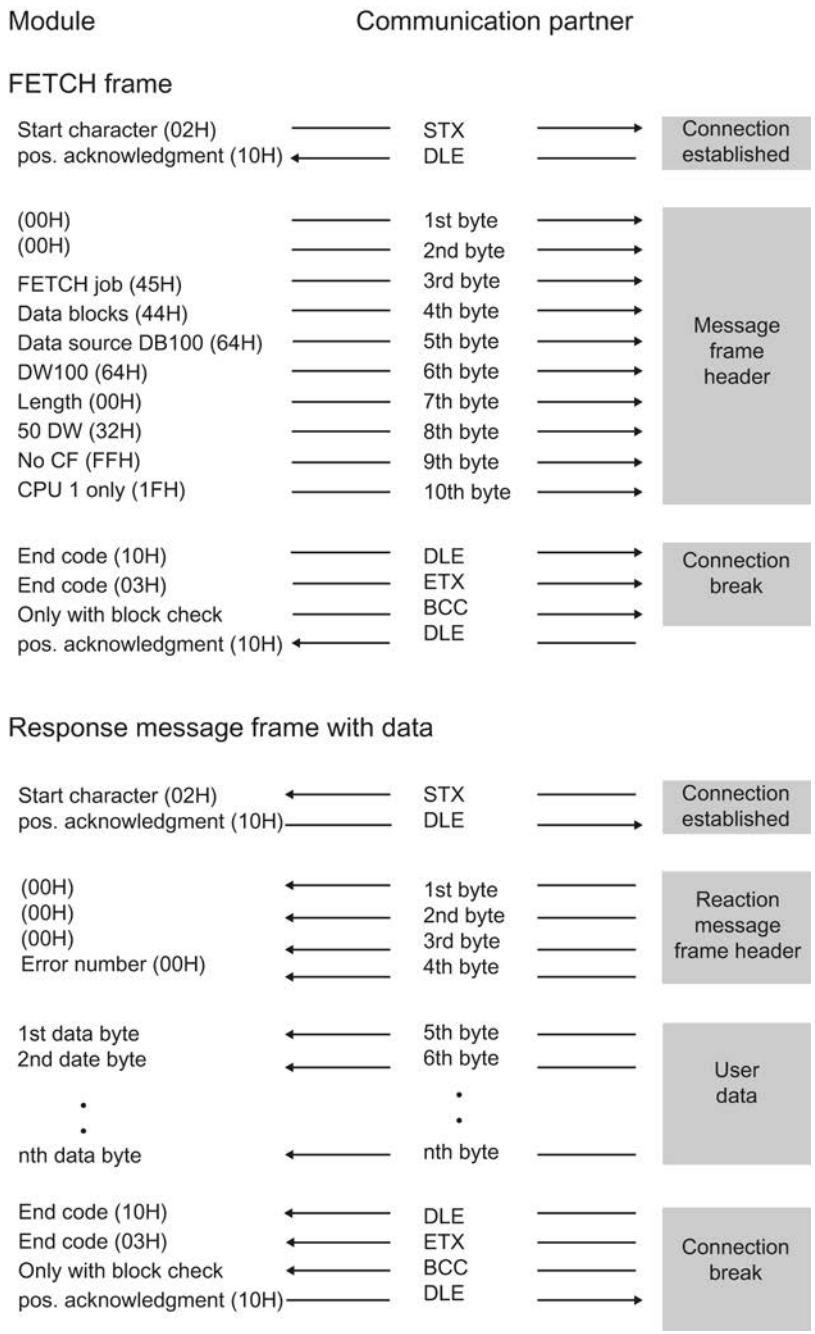


Figure 8-34 Data traffic when fetching with a response message frame

Fetching data

The FETCH job is executed in the following sequence:

- Active partner
  Sends a FETCH frame. This contains a message frame header.
- Passive partner
  Receives the message frame, checks the header, fetches the data from the CPU, and acknowledges this with a response frame containing the data. This contains the data.
- Active partner
  Receives the response message frame, passes user data received in the response message frame to the CPU, and terminates the FETCH job.

If there is an error number (not equal to 0) in the 4th byte, the response frame does not contain any data.

If more than 128 bytes are requested, the extra bytes are automatically fetched in one or more continuation message frames. Each continuation FETCH frame is acknowledged with a continuation response message frame (with more user data).

After receipt of the last continuation response message frame (with the last user data block), the entire user data are passed to the CPU and the FETCH job is completed.

Note

If the FETCH message frame is not correctly processed by the CPU or if an error occurs in the message frame header, the communication partner enters an error number in the 4th byte of the response message frame. This does not apply when protocol errors occur.

Continuation FETCH frame

The figure below shows the transmission sequence when fetching data with a continuation response message frame.

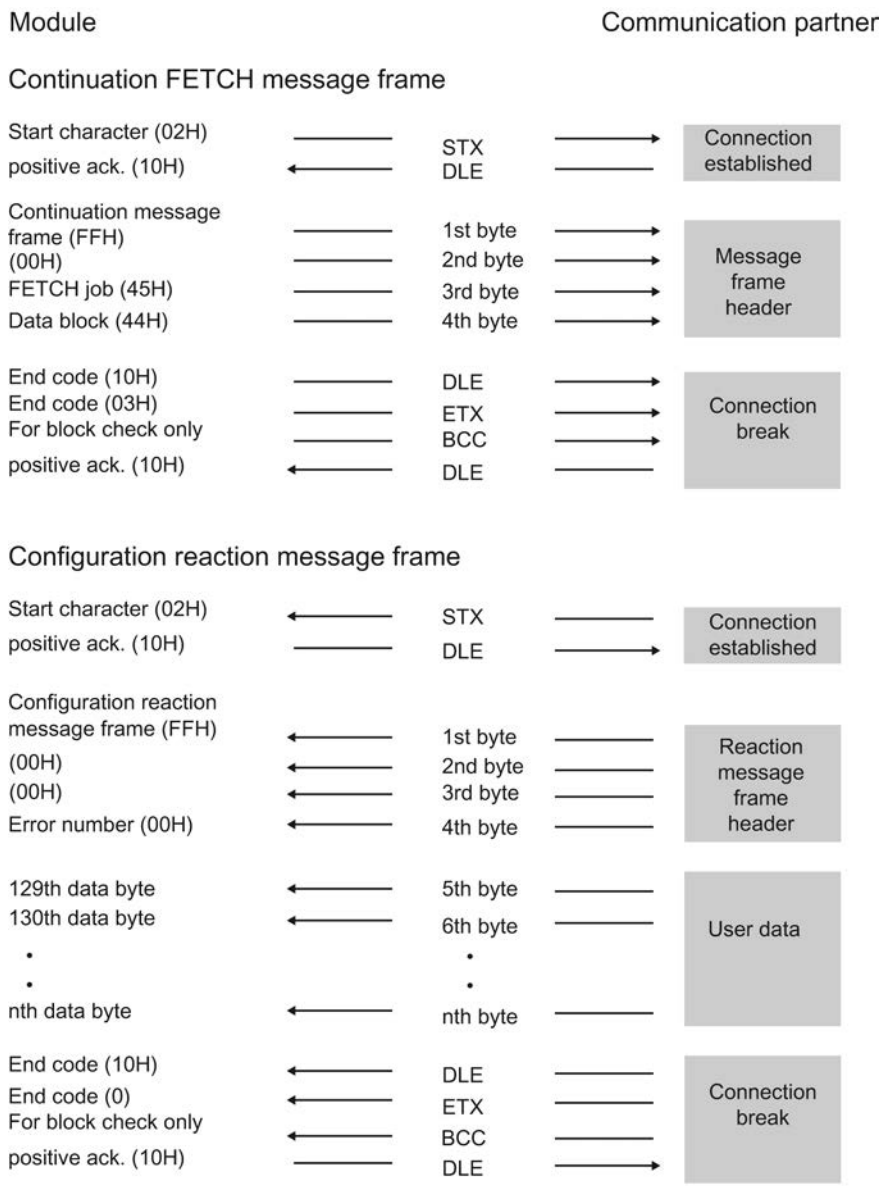


Figure 8-35 Sequence of a FETCH frame with a continuation response message frame

Quasi-full-duplex operation

Quasi-full-duplex mode means: The partners can send command and response frames at any time as long as the other partner is not sending. The maximum nesting depth for command and response frames is "1". The next command message frame, therefore, cannot be processed until the previous one has been answered with a response frame.

It is possible under certain circumstances - when both partners want to send - to transmit a SEND or FETCH message frame from the partner before the response message frame. This can occur, for example, when a SEND or FETCH message frame from the partner is entered in the output buffer of the communication module before the response message frame.

In the following figure, the continuation response message frame for the first SEND frame is only sent after the SEND frame of the partner.

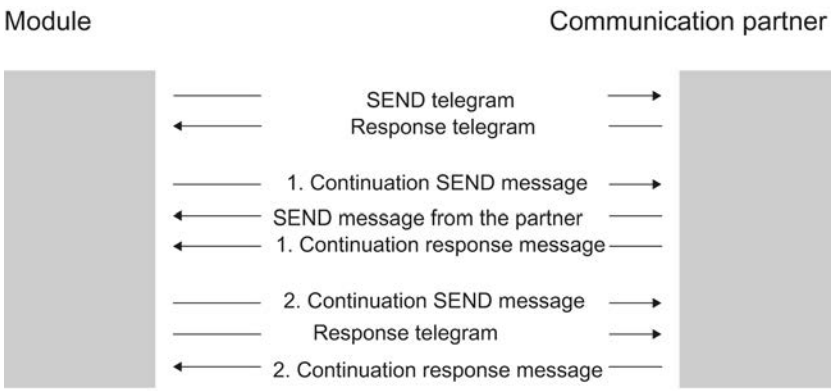


Figure 8-36 Quasi-full-duplex operation

RK512 CPU jobs

The figure below shows the sequences involved in the RK512 computer connection when CPU jobs are made.

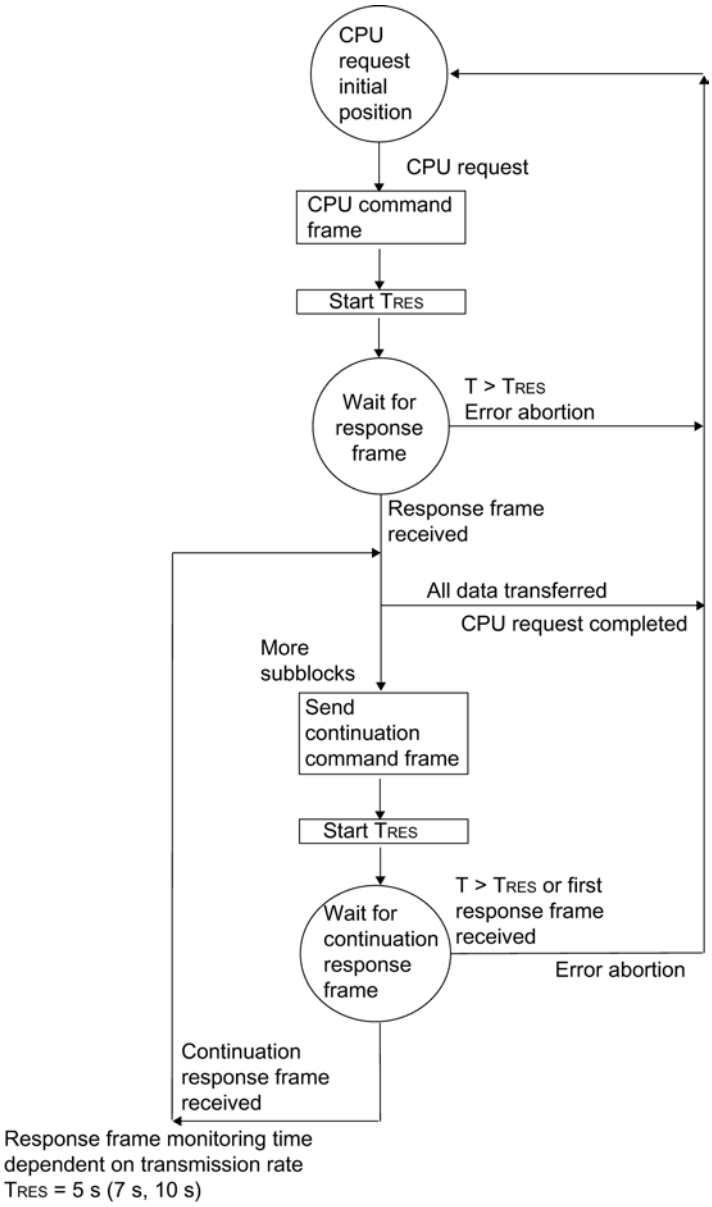
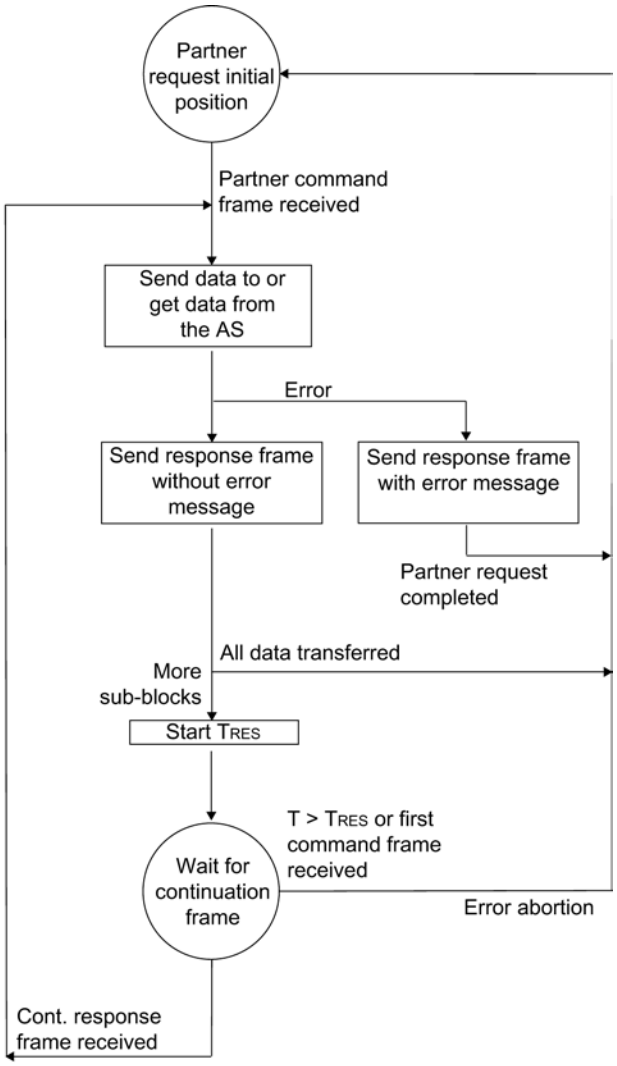


Figure 8-37 Sequence for data transmission from CPU jobs with RK512

RK512 partner jobs

The figure below shows the sequences involved in the RK512 computer connection when partner jobs are made.



Response frame monitoring time dependent on transmission rate
TRES = 5 s (7 s, 10 s)
AS = automation system

Figure 8-38 Flow chart for data transmission from partner jobs with RK512

Data transmission with printer driver

Data transmission with the printer driver

Assignment of the protocols to the communication modules, see: Overview of the components and interfaces (Page 1027)

Introduction

The printer driver allows you to output message texts with the date and time to a printer. This enables you to monitor simple processes, print error or fault messages or issue instructions to the operating personnel, for example.

The printer driver contains the physical layer (layer 1).

Message texts and parameters for printout

In the properties dialog of the module, you configure message text and set the parameters (page layout, character set, control characters) for the printout. Message texts and printout parameters are transmitted to the communication module together with the module parameters when it starts up.

- Message texts:
  You can configure message texts with variables and control instructions (e.g. for bold, narrow, wide or italic type and underlining). Each message text is assigned a number during parameter assignment. The printout of a specific message text with CP 441 modules is made by specifying a reference (to the memory cell containing the message text no.) at the send parameters SD_1 to SD_4 of the instruction. With CP 34x modules, this is done by specifying the message text number in a format string when calling the instruction.
  You can also save configured message texts to an external text file (SDB name, max. 8 ASCII characters), and edit them with a suitable editor. In this way, the texts can be translated and then read back, for example.
  Structure of the message text file:
  <Versions number>
  <Number of the message>:<Content the message>
- Page layout:
  You can configure the margins, optional line breaks and headers and footers of the page layout.
- Character set:
  STEP 7 converts the ANSI character set to the printer character set using a character conversion table. You can change a character conversion table suggested for a printer type in order to include special characters required for a particular language, for example.
- Control characters:
  Using a control character table, you can change the control instructions in the message text for printer emulation to switch on and off bold, narrow, wide or italic type and underline, and to add control characters.

Variables

Up to 4 variables (3 + text message number) are displayed in a message text. Variable values can be transmitted from the CPU to the communication module. The following variables can be displayed:

- Calculated values of the user program (e.g. fill levels)
- Date and time
- Strings (string variable)
- Other message texts

A conversion instruction must be specified for each variable in the configured message text or in the format string, in which the meaning and the output format of the variable value are encrypted.

Format string

You can use the format string to define the appearance and the composition of a message text. The format string can consist of:

- Text (any printable characters, e.g. Fill level l reached at ...!)
- Conversion instructions for variables (such as %N = pointer to the message text number x, where x is a variable value (see Example 2 below))
  For each variable, exactly one conversion instruction must be present in the format string or configured message text. The conversion instructions are applied to the variables according to their order.
- Control instructions with control characters for bold, narrow, wide, italics, underline (e.g. \B = bold on) or with additional control characters you have defined

You can use additional control characters when you enter these characters into the control character table in the properties dialog of the module and configure the new communication module.

Other functions

In addition to the output of message texts, you can use the following functions for the printout. You run these functions from a format string.

- Set page number (format string = %P)
- Start new page (format string = \F)
- Print with/without line break (\x at the end of the format string)

Note that a line feed is performed by default for each printout.

Examples

Example 1: Fill level "200" l reach at "5:30 pm"!

Format string = Fill level %i l reach at %Z!
Variable 1 = time
Variable 2 = level

Example 2: The chamber pressure "drops"

Format string = %N %S
Variable 1 = 17 (message text no. 17: The chamber pressure ...)
Variable 2 = Reference to string (string variable: ... drops)

Example 3: (Set page number to 10)

Format string = %P
Variable 1 = 10 (page number: 10)

Printout

To output n bytes of user data to a printer, the format string and the variables of the message text must be specified as parameters when the instruction is called.

During output the data is edited for printing. Preparation for printing is carried out according to the parameters in the properties dialog of the module (page layout, character set, control characters, etc.).

Characters are not received during printout. The exception to this are any flow control characters that have been configured. Any characters received are not adopted.

Message text output

The figure below illustrates the sequence of operations at printout.

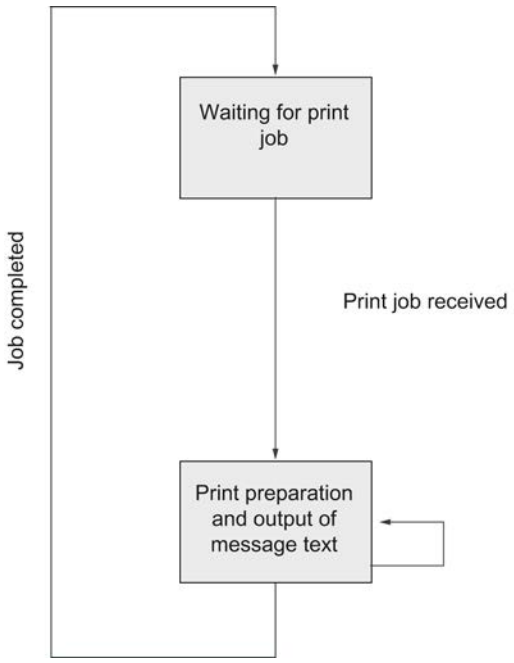


Figure 8-39 Flow chart of printout

Software/hardware handshake

Handshaking controls the data flow between two communication partners. Handshaking ensures that data is not lost in transmissions between devices that work at different speeds.

You can also send printout data with flow control. There are essentially two types of handshaking here:

- Software handshaking (e.g. XON/XOFF)
- Hardware handshaking (e.g. RTS/CTS)

Implementation of flow control on the communication module for printout is performed as follows:

- As soon as the communication module is switched by configuration to an operating mode with flow control, it sends the XON character or sets the RTS line to ON.
- If the communication module receives the XOFF character or the CTS control signal is set to OFF, the communication module interrupts the output of characters. If no XON is received or CTS is not set to ON after a configurable time, the printout is interrupted and a corresponding error message (0708H) is generated at the STATUS output of the instruction.

Note

If you configure RTS/CTS flow control, be sure the interface signals used are fully wired in making the plug connection.

BUSY signal

The communication module evaluates the "BUSY" control signal of the printer. The printer reports to CP 34x1 its readiness to receive

- With CP 34x 20mA-TTY: With power on RxD line.
- With CP 34x RS232C and CP 34x RS422/485: With CTS = "ON" signal.

Note

If you configure RTS/CTS flow control, you must set the polarity of the BUSY signal on the printer as follows:

- BUSY signal: CTS = "OFF"

Note that some printers show the BUSY signal using the DTR signal. In this case, you need to rewire the cable to the communication module correspondingly.

Indicator for end of a message freme

End criteria

You can choose between three different end criteria for data transmission using the ASCII driver:

- **After character delay time elapses**
- **On receipt of the end delimiter(s)**
- **On receipt of fixed number of characters**

Protocol parameters

The following table provides a description of the protocol parameters.

Table 8-75 Protocol parameters (ASCII driver)

Parameters	Description	Value range	Default value
End detection indicator of a receive message frame	Specifying the criteria that determine the end of message frames.	• After character delay time elapses • On receipt of the end delimiter(s) • On receipt of fixed number of characters	After character delay time elapses
End delimiter 1 [1]	Code of the first end detection.	• With 7 data bits: 0 to 7FH (Hex) [2] • With 8 data bits: 0 to FFH (Hex) [2]	3 (03H = ETX)
End delimiter 2 [1]	Code of the second end detection, if selected.	• With 7 data bits: 0 to 7FH (Hex) [2] • With 8 data bits: 0 to FFH (Hex) [2]	0
[1] Only for the end delimiter end criterion. [2] Depending on whether you configure 7 or 8 data bits for the character frame.			

Further Information

A detailed description of data reception with the ASCII driver can be found in section Receiving Data with the ASCII Driver (Page 1042).

Data flow control and secondary signals

Introduction

The communication module enables point-to-point communication with SIMATIC modules and with third-party products.

Supported interface functions

Different driver functions can be used depending on the interface used:

Table 8-76 Functions of the communication module based on the interface employed

	RS232C	20mA TTY	X27 (RS422/485)	
			RS422*	RS485*
Function CP 441				
3964(R) procedure	●	●	●	-
Computer connection RK512	●	●	●	-
ASCII driver:	●	●	●	●

	RS232C	20mA TTY	X27 (RS422/485)	
			RS422*	RS485*
• Operation of the RS232C accompanying signals	●	-	-	-
• Controlling/reading the RS232C accompanying signals with instructions	●	-	-	-
• RTS/CTS data flow control	●	-	-	-
• XON/XOFF data flow control	●	●	●	-
Printer driver:	●	●	●	●
• RTS/CTS data flow control	●	-	-	-
• XON/XOFF data flow control	●	●	●	-
Function CP 440				
3964(R) procedure			●	-
ASCII driver:			●	●
• XON/XOFF data flow control			●	-
Function CP 341				
3964(R) procedure	●	●	●	-
Computer connection RK512	●	●	●	-
ASCII driver:	●	●	●	●
• Operation of the RS232C accompanying signals	●	-	-	-
• Controlling/reading the RS232C accompanying signals with instructions	●	-	-	-
• RTS/CTS data flow control	●	-	-	-
• XON/XOFF data flow control	●	●	●	-
Printer driver:	●	●	●	●
• RTS/CTS data flow control	●	-	-	-
• XON/XOFF data flow control	●	●	●	-
Function CP 340				
3964(R) procedure	●	●	●	-
ASCII driver:	●	●	●	●
• Operation of the RS232C accompanying signals	●	-	-	-
• Controlling/reading the RS232C accompanying signals with instructions	●	-	-	-
• RTS/CTS data flow control	●	-	-	-
• XON/XOFF data flow control	●	●	●	-
Printer driver:	●	●	●	●
• RTS/CTS data flow control	●	-	-	-
• XON/XOFF data flow control	●	●	●	-

Function ET 200S 1SI				
3964(R) procedure	●		●	-
ASCII driver:	●		●	●
• Operation of the RS232C accompanying signals	●		-	-
• Controlling/reading the RS232C accompanying signals with instructions	●		-	-

• RTS/CTS data flow control	•		-	-
• XON/XOFF data flow control	•		•	-
* The difference between RS422 and RS485 is defined by configuration.				

Application options for the communication modules

Depending on the communication module used, you can establish a point-to-point connection with various Siemens modules and with third-party products:

- SIMATIC S5 through 3964(R) driver or RK512 with corresponding interface module at S5 end
- Siemens BDE-terminals ES 2-family through 3964(R) driver
- MOBY I (ASM 420/421, SIM), MOBY L (ASM 520) and capture station ES 030K through 3964R-driver
- PCs via procedure 3964(R) (development tools exist for programming on the PC: PRODAVE DOS 64R (6ES5 897-2UD11) for MS-DOS, PRODAVE WIN 64R (6ES5 897-2VD01) for Windows or the ASCII driver)
- Bar code reader via 3964(R) driver or ASCII driver
- PLC from other manufacturers via 3964(R), ASCII driver or RK512
- other devices with simple protocol structures via corresponding protocol adjustment with the ASCII driver
- Additional devices that also feature 3964(R) drivers or RK512
- Printer (HP-Deskjet, HP-Laserjet, post script, Epson, IBM)

Interface operating modes

Possible interface operating modes

The following table is a summary of the interface operating modes from the various communication modules and protocols.

Operating mode	Description
Half-duplex (RS485) two-wire mode.	Operating mode for point-to-point connection or multipoint connection (multipoint) in two-wire mode. The communication module can be the master as well as the slave. During RS485-mode, the data transmission occurs through two lines (two-wire mode). The two lines are available alternatively for send and receive directions. They can either send or receive (half-duplex). After a send process, it is automatically switched to receive (maximum switch-over time of 1 ms).
Full-duplex (RS422), four-wire mode	During RS422-mode, the data transmission occurs through four lines (four-wire mode). There are always two lines available for send and receive directions. They can send and receive simultaneously (full-duplex). The data is exchanged simultaneously between one or more communication partners, i.e., it can be sent and received at the same time. Each communication partner must be able to simultaneously operate a transmission and reception set-up.
Full-duplex (RS 422) four-wire mode (point-to-point connection) [1)]	The communication module can be used in the following topologies in RS422 mode: • Link between two participants: Point-to-point connection • Connection between numerous stations: Multipoint connection The communication module can be used as both master and slave.

Operating mode	Description
Full-duplex (RS422) four-wire mode (multipoint master) [1]	The communication module can be used in the following topologies in RS422 mode: • Link between two participants: Point-to-point connection • Connection between numerous stations: Multipoint connection The communication module can be used as a multipoint master as well as a multipoint slave. In the case of a master/slave topology in the RS422 mode: • The master's sender is interconnected with the receivers of all the slaves. • The slaves' senders are interconnected with the master's receiver. • Only the receiver of the master and the receiver of one slave have a default setting. All the other slaves function without default settings.
Full-duplex (RS422) four-wire mode (multipoint slave) [1]	The communication module can be used in the following topologies in RS422 mode: • Link between two participants: Point-to-point connection • Connection between numerous stations: Multipoint connection The communication module can be used as a multipoint master as well as a multipoint slave. In the case of a master/slave topology in the RS422 mode: • The master's sender is interconnected with the receivers of all the slaves. • The slaves' senders are interconnected with the master's receiver. • Only the receiver of the master and the receiver of one slave have a default setting. All the other slaves function without default settings.

[1] CP 440 only

Initial state of receive line

X27 (RS422/485) interface

The table below contains descriptions of the parameters for the X27 (RS422/485) interface submodule. RS485-mode is not possible with the 3964(R) and RK512 procedures and the printer.

Table 8-77 X27 (RS422/485) interface module

Parameters	Description	Value range	Default value
Initial state of receive line	**none**: This setting only makes sense with bus-capable special drivers.	None	Signal R(A) 5V, signal R(B) 0V (break detection) [2]
	Signal R(A) 5V, signal R(B) 0V (break detection): Break detection is possible in "Full-duplex (RS 422) four-wire mode" with this setting.	Signal R(A) 5V, signal R(B) 0V (break detection) [1]	

Parameters	Description	Value range	Default value
	Signal R(A) 0V, signal R(B) 5V: This initial state corresponds to idle (no senders active) in "Half Duplex (RS 485) Two-Wire Mode". No break detection is possible with this setting.	Signal R(A) 0V, signal R(B) 5V	
(1) Only with "Full-duplex (RS422) four-wire mode" (2) Only with "Full-duplex (RS422) four-wire mode", with "Half-duplex (RS485) two-wire mode" the default setting is signal R (A) 0V, signal R (B) 5V			

Initial state of receive line

The figure illustrates the wiring of the receiver at the X27 (RS422/485) interface:

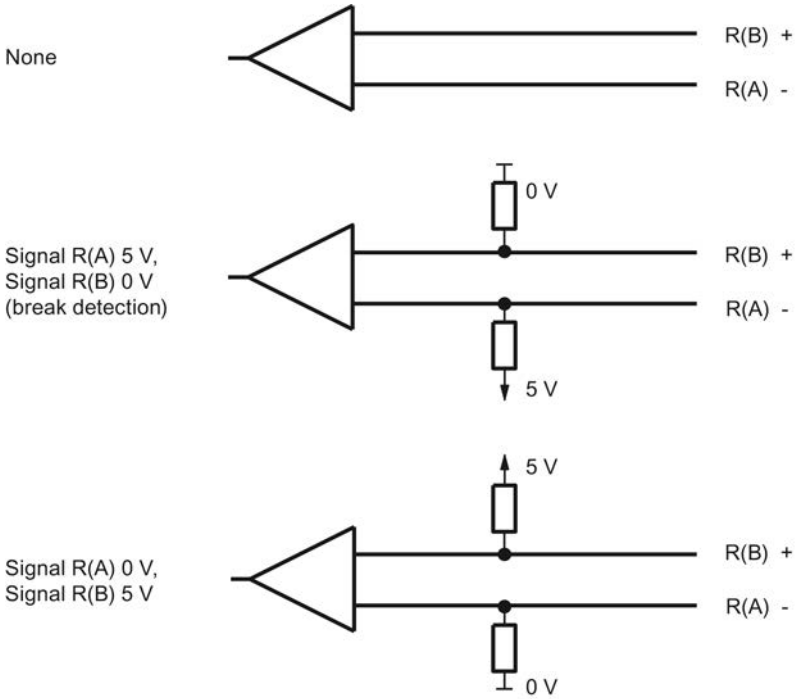


Figure 8-40 Wiring of the receiver at the X27 (RS422/485) interface

CP 440 in a multipoint topology with the RS422/485 protocols

Application options

The CP 440 can be used in different topologies in the RS422 and RS485 operating modes.

Distinctions are drawn between connections with:

- Two nodes (**point-to-point**) and
- Several nodes (**multipoint**)

In these cases the CP 440 can be used as:

- Master or
- Slave.

RS422 or RS485 mode for point-to-point connections

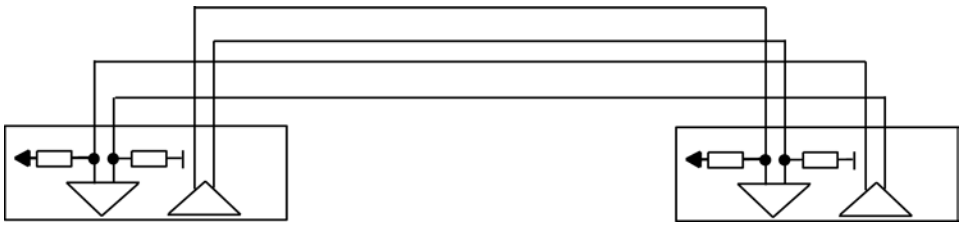


Figure 8-41 RS422 point-to-point

Figure 8-42 RS485 point-to-point

In the case of a master/slave topology, there must be an appropriate message frame in the user program. Example: The master sends all the slaves a message frame with address information. All the slaves listen in and compare the address with their own. If the address is the same, the addressed slave sends its answer.

The senders of all slaves must be able to switch to low impedance.

RS422 mode in multipoint connections

In the case of a master/slave topology in RS422 mode:

- The master's sender is interconnected with the receivers of all the slaves.
- The slaves' senders are interconnected with the master's receiver.
- Only the receiver of the master and the receiver of one slave have a default setting. All the other slaves function without default settings.

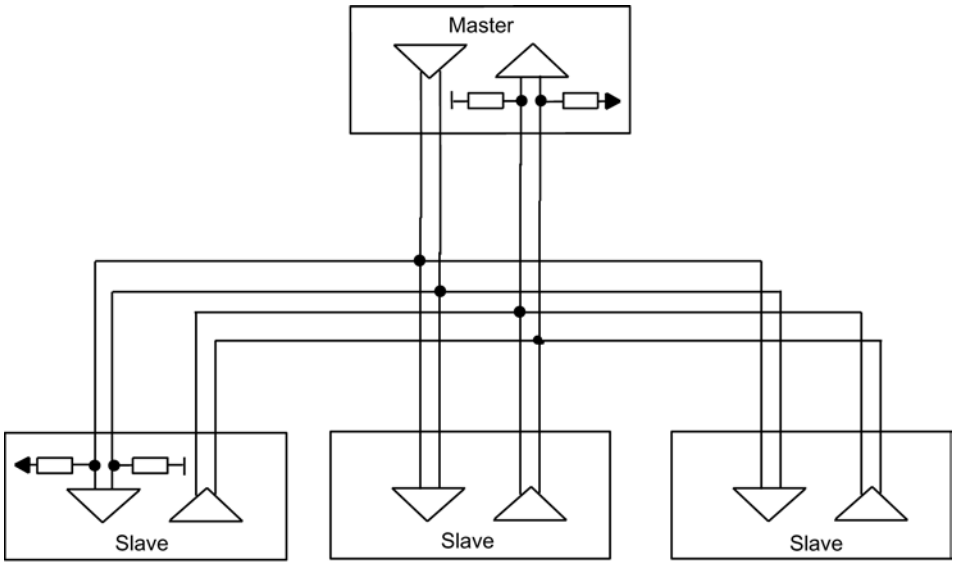


Figure 8-43 RS422 multipoint

RS485 mode in multipoint connections

In the case of a topology in RS485 operation:

- The cable pair is interconnected for the send/receive line of all the nodes.
- Only the receiver of a node has a default setting. All the other modules function without default settings.

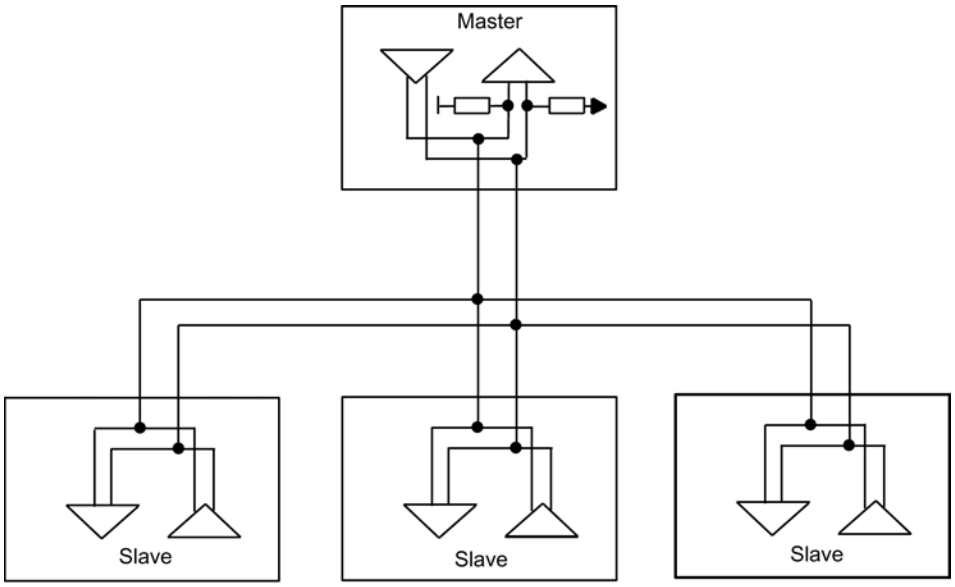


Figure 8-44 RS485 multipoint

The necessary settings are made for the different topologies in the "Interface" section of the properties dialog for the module.

Note

When you run the ASCII driver in RS422 multipoint or RS485 mode, you must take steps in the user program to ensure that only one node sends data at any one time. If two users send data simultaneously, the message frame is corrupted.

Receive buffer

Receive buffer on the module

The following table describes the parameters for the module receive buffer.

Table 8-78 Receive buffer from the module - procedure 3964(R) / RK512 / ASCII driver

Parameters	Description	Value range	Default value
Clear module receive buffer on startup	The module receive buffer is not deleted at CPU start-up (STOP RUN transition).	No (fixed)	No
Use CPU receive mailbox (CP 441 only)	Here you can specify whether a receive mailbox is to be set up on the CPU. You must set up a receive mailbox if you have not programmed a BRCV instruction for the module in the user program of the CPU. If you have programmed a BRCV, you must deactivate this parameter, otherwise data will be stored in the receive mailbox defined here instead of being processed by the BRCV.	• Yes • No	No
DB number (1)	Number of the data block for the receive mailbox on the CPU.	1 to 65535 (depending on the CPU)	1
(1) Only when "Use CPU Receive Mailbox" = "Yes" (2) Only when "Buffered receive message frames" = "1"			

Table 8-79 Receive buffer from the module - additional for ASCII drivers

Parameters	Description	Value range	Default value
Buffered receive message frames	Here you can specify the number of receive message frames to be buffered in the receive buffer of the communication module. If you specify "1" here and deactivate the following parameter "prevent overwrite" and cyclically read the received data from the user program, a current message frame will always be sent to the CPU.	1 to 250	250
Dynamic message frame ET 200S 1SI	For the reception of messages, you can specify if a message should just be buffered or if the messages should be dynamically buffered. By activating dynamic message frames, the module can buffer multiple messages with different lengths. The buffer is a ring buffer. If the buffer is full, the oldest message is overwritten, unless the "Prevent overwriting buffers" parameter is enabled. In this case, the newest message is discarded. In both cases, a diagnosis alarm will display saying that data was lost.	• Activated • Blocked	Activated
Prevent overwrite	You can deactivate this parameter if the parameter "buffered receive message frames" is set to "1". This authorizes the buffered receive message frame to be overwritten.	• Yes • no (2)	Yes
(1) Only when "Use CPU Receive Mailbox" = "Yes" (2) Only when "Buffered receive message frames" = "1"			

Modbus communication

Overview of modbus communication

Position in the system environment

The following Modbus description refers to the use of the reloadable driver on the CP 341 (S7 300) and CP 441-2 (S7 400) communication processors, and the operation of the ET 200S 1SI module in the Modbus/USS model.

Function of the coupling

With the appropriately equipped communication modules and the related instructions, you can establish a communication connection between a remote Modbus control system (such as Modicon PLCs or Honeywell TDC3000) and a SIMATIC S7.

The GOULD-MODBUS protocol in the RTU format is used for transmission.

Function codes 01, 02, 03, 04, 05, 06, 08, 15 and 16 are used for communication between a communication module operated as a Modbus slave and a master system.

If a SIMATIC S7 communication module is operated as a Modbus master, function codes 07, 11 and 12 are additionally available.

Modbus protocol

The modbus protocol is a communication protocol based on a master/slave and client/server architecture.

The procedure used is a code-transparent, asynchronous half-duplex procedure. Data transmission is carried out without handshake.

SIMATIC S7 as a Modbus slave

The master has the initiative for transmission, the CP / S7 CPU works as a slave.

Frame traffic from slave to slave is not possible.

SIMATIC S7 as a Modbus master

As master, the communication module initiates transmission and, after outputting a request message frame, it waits for the configured response monitoring time for a response message frame from the slave.

Message frame structure

The data exchange "Master-Slave" and/or "Slave-Master" begins with the **slave address**, followed by the **function code**. Then the data are transferred. The structure of the data field depends on the function code used. The CRC check is transmitted at the end of the message frame.

ADDRESS	FUNCTION	DATA	CRC-CHECK
Byte	Byte	n byte	2 byte

ADDRESS	MODBUS slave address
FUNCTION	MODBUS function code
DATA	Message frame data: Byte_Count, Coil_Number, Data
CRC-CHECK	Message frame checksum

Slave address

The slave address can be within the range of 1 to 255. The address is used to address a defined slave on the bus.

Broadcast Message

The master uses slave address 0 to address all slaves on the bus.

Broadcast messages are only permitted in conjunction with writing **Function codes 05, 06, 15 and 16**.

A broadcast message is not followed by a response message frame from the slave.

Function code

The function code defines the meaning of the message frame. It also defines the structure of a message frame. The following function codes are supported by the communication module:

Function Code	Function in accordance with MODBUS specification
01	Read Coil Status
02	Read Input Status
03	Read Holding Registers
04	Read Input Registers
05	Force Single Coil
06	Preset Single Register
07	Read Exception Status (only master)
08	Loop Back Test
11	Fetch Communications Event Counter (only master)
12	Fetch Communications Event Log (only master)
15	Force Multiple Coils
16	Preset Multiple Registers

Data Field DATA

The data field DATA is used to transfer the function code-specific data such as:

- Bytecount, Coil_Startaddress, Register_Startaddress; Number_of_Coils, Number_of_Registers, ...
  See section "Function codes (Page 1101)".

CRC-Check

The end of the message frame is identified by means of the CRC 16 checksum consisting of 2 bytes. It is calculated by the following polynominal: $x^{16} + x^{15} + x^2 + 1$.

The low byte is transferred first, followed by the high byte.

End of Message Frame

The loadable driver recognizes the end of the message frame when no transmission takes place during the time period required for the transmission of three and a half characters (3.5 times character delay time) (see MODBUS Protocol Reference Guide).

This message frame end TIME_OUT is thus dependent on the data transmission rate.

Data transmission rate	TIME_OUT
115200 bps	0,3 ms
76800 bps	0,5 ms
57600 bps	0.7 ms
38400 bps	1 ms
19200 bps	2 ms

Data transmission rate	TIME_OUT
9600 bps	4 ms
4800 bps	8 ms
2400 bps	16 ms
1200 bps	32 ms
600 bps	64 ms
300 bps	128 ms

During "Normal operation" the Modbus message frame received by the connection partner is evaluated and checked after the end of frame for TIME_OUTs is received.

During "Interference suppression" the end of frame is recognized by correctly formatted receive frame with a correct CRC code.

Exception responses

On recognition of an error in the request message frame from the master, for example, register address illegal, the slave sets the highest value bit in the function code of the response message frame.

This is followed by transmission of one byte of error code, exception code, which describes the reason for the error.

A detailed description of the meaning of the above-mentioned parameters can be found in the “GOULD MODICON Modbus Protocol.”

Exception code message frame

The error code response message frame from the slave has the following structure:

- for example, slave address 5, function code 5, exception code 2

Response frame from the slave EXCEPTION_CODE_xx:

05H	Slave address
85H	Function code
02H	Exception code (1...7)
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

On receipt of an error code response message frame by the driver, the current job is completed with error.

An error number corresponding to the received error code (exception code 1-7) is also entered in the error message area.

There is no entry in the target data block.

The following error codes are defined in accordance with the MODBUS Specification:

Error code	Meaning in accordance with MODBUS Specification	Cause - Short Description *
1	Illegal function	Illegal function code
2	Illegal data address	Slave has illegal data address
3	Illegal data value	Slave has illegal data value
4	Failure in Associated Device	Slave has internal error
5	Acknowledge	Function is carried out
6	Busy, Rejected message	Slave is not ready to receive
7	Negative acknowledgement	The function cannot be carried out.
* Check slave for further details.		

RS232C accompanying signals

The following RS232C accompanying signals are available on the communication modules that use the RS232C interface module:

DCD	(input)	Data carrier detect	Data carrier detected
DTR	(Output)	Data terminal ready	Communication module ready
DSR	(input)	Data set ready	Communication partner ready for operation
RTS	(Output)	Request to send	Communication module ready to send
CTS	(input)	Clear to send	Communication partner can receive data from the communication module (response to RTS = ON of the CP)
RI	(input)	Ring indicator	Ring indicator

When the communication module is switched on, the output signals are in the OFF state (inactive).

You can configure the operation of the DTR/DSR and RTS/CTS control signals in the properties dialog of the module or via instructions in the user program.

Operating RS232C accompanying signals

The RS232C accompanying signals can be used as follows:

- When the automatic use of all RS232C accompanying signals is configured.
- Via the corresponding instructions. For more information, see the section "Overview of the instructions (Page 1029)".

Note

When automatic operation of the RS232C accompanying signals is configured, RTS and DTR cannot be controlled via the corresponding instruction (see section "Overview of the instructions (Page 1029)")!

On the other hand, it is always possible to read all RS232C accompanying signals using the corresponding instruction.

The following sections describe the basic procedures for control and evaluation of the RS232C accompanying signals.

Automatic use of accompanying signals

Automatic operation of the RS232C accompanying signals on the communication module is realized as follows:

- As soon as the communication module is switched by configuration to an operating mode with automatic operation of the RS232C accompanying signals, it sets the RTS line to OFF and the DTR line to ON (communication module ready for use).
- This prevents sending and receiving of message frames until the DTR line is set to ON. As long as DTR remains set to OFF, no data is received via the RS232C interface. If a send job is made, it is aborted with a corresponding error message.
- When a send job is made, RTS is set to ON and the parameterized data output waiting time starts. When the data output time elapses and CTS = ON, the data is sent via the RS232C interface.
- If the CTS line is not set to ON within the data output time so that data can be sent, or if CTS changes to OFF during transmission, the send job is aborted and an error message generated.
- Once the data has been sent and the configured clear RTS time has elapsed, the RTS line is set to OFF. The CP does not wait for CTS to change to OFF.
- Data can be received via the RS232C interface as soon as the DSR line is set to ON. If the receive buffer of the communication module threatens to overflow, there is no response from the communication module.
- If DSR changes from ON to OFF, an active send job as well as the receipt of data will be canceled with an error message. The message "DSR = OFF (automatic operation of V24 signals)" is entered in the diagnostics buffer of the communication module.

Note

When automatic operation of the RS232C accompanying signals is configured, RTS and DTR cannot be controlled via the corresponding instruction! For more information, see the section "Overview of the instructions (Page 1029)".

Note

Set "Clear RTS time" in the properties dialog of the module so that the communication partner can receive the last characters of the message frame in their entirety before RTS, and thus the send job, is revoked. The "data output waiting time" must be set such that the communication partner can be ready to receive before the time elapses.

Time Diagram

The following figure illustrates the chronological sequence of a send job.

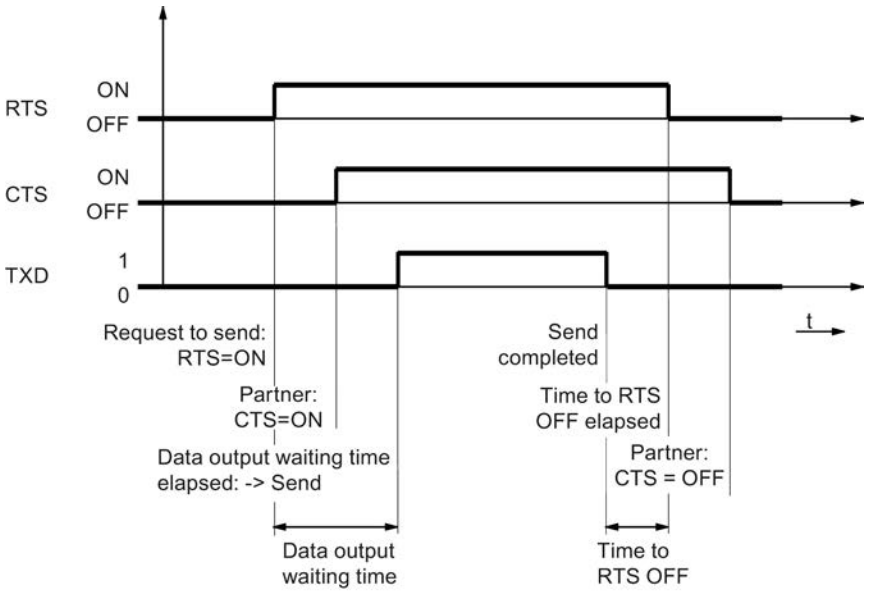


Figure 8-45 Time diagram for automatic operation of the RS232C accompanying signals

Modbus master

CP 341

Data transmission from CPU to communication module with P_SND_RK (CP 341)

Activation

Execution of a MODBUS function code is activated by means of the instruction **P_SND_RK** with an **edge** at the **REQ** input.

Enter ‘S’ for SEND at the SF parameter.

The logical module address is entered at LADDR.

You must enter ‘X’ for expanded data block as the area type of the partner CPU. No values must be specified for the other parameters of the partner CPU (R_...).

This ensures transfer to the driver of the parameters required for the execution of the function code.

Data source

When P_SND_RK is activated, the **source data area** specified with the **DB_NO** and **DBB_NO** parameters is transferred to the communication module with the length **LEN**.

Length Indication

The length **LEN** depends on the function code used.

Function code	Length LEN in bytes
01	6
02	6
03	6
04	6
05	6
06	6
07	2
08	6
11	2
12	2
15	>6
16	>6

If the transferred data quantities differ from those listed above for the individual function codes, the job is not carried out and P_SND_RK rejects it with an edge at output ERROR.

SEND Source DB

The parameters required for the execution of a function code must be entered as user data in the source data area.

A detailed description of each P_SND_RK source DB is available in the description of the individual function codes in the section "Function codes (Page 1101)".

Generation of Message Frames

The request message frames to the slave are generated in accordance with the transferred P_SND_RK source data and sent by the communication module.

First of all the driver checks if the length LEN specified at P_SND_RK corresponds to the length for this function code.

If not, the job is not carried out and it is completed with an edge at output ERROR of the P_SND_RK.

When using function codes other than those listed, the activated job is not carried out either and is completed with ERROR at P_SND_RK.

The “byte counter” and “CRC check” elements in the request message frame are generated by the communication module; an entry in the P_SND_RK source DB is not required.

Job Completion for Writing Functions

For writing function codes, the activated P_SND_RK is completed after a response message frame is received without error. This is communicated to the SIMATIC user program by means of an edge at output **DONE** of the P_SND_RK.

If **errors** are detected in the message frame traffic, or if the slave sends an **error code** response message frame, this is reported by an edge at output **ERROR**.

Job Completion for Reading Functions

For reading functions, the activated P_SND_RK is completed after the response message frame is received without error **and** complete transfer of the received data to the CPU.

This is communicated to the SIMATIC user program by means of an edge at output **DONE** of the P_SND_RK.

At this time the received data are already available in the CPU.

If **errors** are detected in the message frame traffic, or if the slave sends an **error code** response message frame, this is reported by an edge at output **ERROR**.

In this case no receive data are transferred to the CPU.

STATUS Entry on Job Completion

For those instances when a job is completed with **ERROR** at P_SND_RK, an additional error code is entered in the **STATUS** parameter.

The exact cause for the error can be determined with this error code

Data transmission from the communication module to the CPU with P_RCV_RK (CP 341)

Prerequisite

All **reading** function codes require a P_RCV_RK.

Data destination

When the instruction P_RCV_RK is ready to receive, it accepts the received data from the communication module and enters it into the data destination specified in the parameters **DB_N0** and **DBB_N0**.

How Receipt of Data is Displayed

The user is informed of the receipt of data in the CPU by means of an edge at output **NDR**.

At this point the length of the received data block is displayed in the parameter **LEN**.

Completion of the entire modbus job can be recognized at output DONE of the instruction P_SND_RK.

How to Handle an Error

In the event of receive or transfer errors, no data is transferred to the CPU. In this case, the instruction P_SND_RK is completed with an edge at the output ERROR.

P_RCV_RK Destination DB

The user data received with a reading function code are entered into the P_RCV_RK destination area.

A detailed description of each P_RCV_RK destination DB can be found in section " Function codes (Page 1101)".

The length of data entered is displayed an the parameter LEN of P_RCV_RK.

CP 441-2

Data transmission from CPU to communication module with BSEND (CP 441-2)

Communication connection

The **Parameter ID** describes the unique communication connection to a communication partner. You must specify the local ID from the data link configuration here

Block Relationship

The **Parameter R_ID** describes the unique block relationship within a communication connection.

With this driver, **any** values from **0..255** may be entered for R_ID on BSEND.

In the event of reading jobs, parameter assignment of the associated BRCV must have the same R_ID as BSEND.

Activation

Execution of a MODBUS function code is activated by means of the instruction **BSEND** with an **edge** at the **REQ** input.

This ensures transfer to the driver of the parameters required for the execution of the function code.

Data source

When BSEND is activated, the **source data area** specified with parameter **SD_1** is transferred to the communication module with the length **LEN**.

Length Indication

The length **LEN** depends on the function code used.

Function code	Length LEN in bytes
01	6
02	6
03	6
04	6
05	6
06	6
07	2
08	6
11	2
12	2
15	>6
16	>6

If the transferred data quantities differ from the ones listed above for the individual function codes, the job is not carried out and BSEND rejects it with an edge at output ERROR.

BSEND Source DB

The parameters required for the execution of a function code must be entered as user data in the source data area.

A detailed description of each BSEND source DB is available in the description of the individual function codes in section "Function codes (Page 1101)".

Generation of Message Frames

The request message frames to the slave are generated in accordance with the transferred BSEND source data and sent by the communication module.

First of all the driver checks if the length LEN specified at BSEND corresponds to the length for this function code.

If not, the job is not carried out and it is completed with an edge at output ERROR of BSEND.

When using function codes other than those listed, the activated job is not carried out either and is completed with ERROR at BSEND.

The “byte counter” and “CRC check” elements in the request message frame are generated by the communication module; an entry in the BSEND source DB is not required.

Job Completion for Writing Functions

For writing function codes, the activated BSEND is completed after a response message frame is received without error.

This is communicated to the SIMATIC user program by means of an edge at output **DONE** of the BSEND.

If **errors** are detected in the message frame traffic, or if the slave sends an **error code** response message frame, this is reported by an edge at output **ERROR**.

Job Completion for Reading Functions

For reading functions, the activated BSEND is completed after the response message frame is received without error **and** complete transfer of the received data to the CPU.

This is communicated to the SIMATIC user program by means of an edge at output **DONE** of the BSEND.

At this time the received data are already available in the CPU.

If **errors** are recognized in the data frame traffic, or if the slave sends an **error code** response message frame, this is reported by an edge at output **ERROR**.

In this case no receive data are transferred to the CPU.

Entry in the error message area at job completion

For those instances when a job is completed with **ERROR** at BSEND, an additional error code is entered in the error message area.

The exact cause for the error can be determined with this error code.

Data transmission from the communication module to the CPU with BRCV (CP 441-2)

Communication connection

The **Parameter ID** describes the unique communication connection to a communication partner. You must specify the local ID from the data link configuration here.

Block Relationship

The **Parameter R_ID** describes the unique block relationship within a communication connection.

All **reading** function codes require a BRCV.
Parameter assignment of **R_ID** on **BRCV** must be the **same R_ID** as for the corresponding BSEND, which was used to activate this job (any values 0 to 255).

In this way you can program several BSEND / BRCV pairs in the SIMATIC user program.

The response message frames received from the Modbus slave are then stored in different destination areas, depending on the R_ID used for this job.

Data destination

When the BRCV instruction is ready to receive, it accepts the received data from the communication module and enters them in the data destination specified in the parameter **RD_1**. This means that the data destination is variable.

How Receipt of Data is Displayed

The user is informed of the receipt of data in the CPU by means of an edge at output **NDR**.

At this point the length of the received data block is displayed in the parameter **LEN**.

You can recognized that the entire Modbus job is completed at the DONE output of the BSEND instruction.

How to Handle an Error

In the event of receive or transfer errors, no data is transferred to the CPU. In this instance BSEND is completed with an edge at the output ERROR.

BRCV Destination DB

The user data received with a reading function code are entered into the BRCV destination area.

The detailed structure of the respective BRCV destination DB is provided in the description of the individual function codes in section "Function codes (Page 1101)".

The length of data entered is displayed at the parameter LEN of BRCV.

ET 200S 1SI

Data transmission with the ET 200S 1SI Modbus master

Introduction

Data transmission between the module and CPU using the S_SEND and S_RCV instructions.

S_SEND instruction: Send data to a communication partner

The S_SEND and S_RCV instructions must be activated in order to execute a Modbus master job. The S_SEND instruction is activated by an edge at the REQ input when data is to be output to the module. The S_RCV instruction is made ready to receive data from the module with

EN_R=1. An S_RCV is required for all reading function codes. The figure below shows the overall behavior of the S_SEND and S_RCV parameters when a Modbus job is executed.

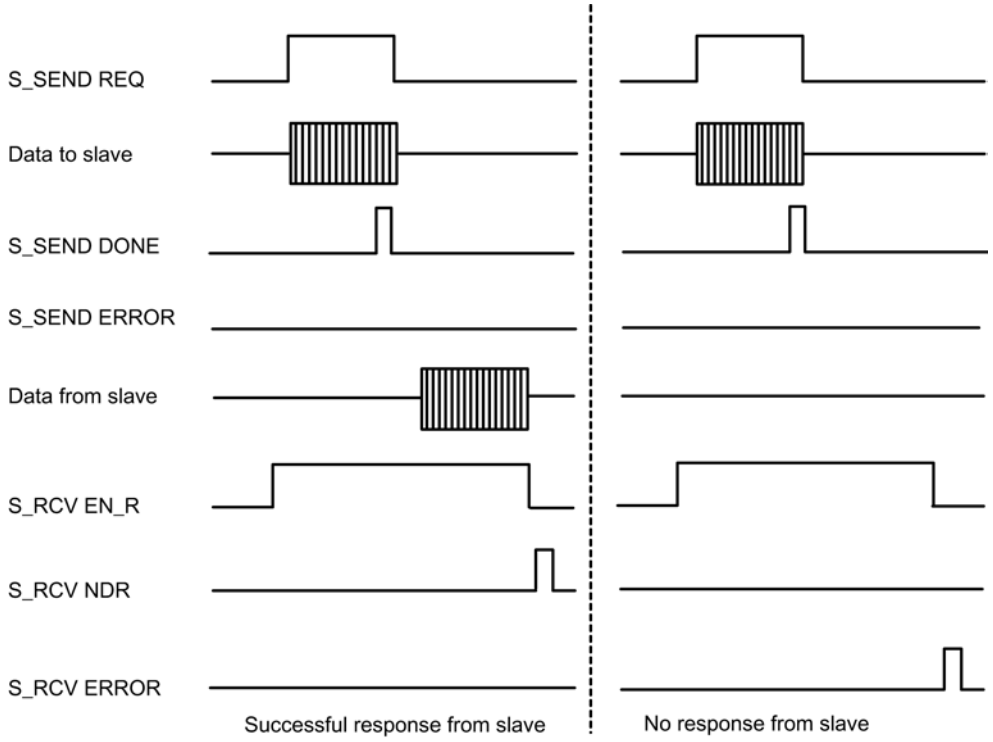


Figure 8-46 Time Sequence Chart for Modbus Request

Data transmission is initiated by a positive edge at the REQ input. Data transmission can take several calls (program cycles), depending on the data volume.

The S_SEND instruction can be called cyclically by setting the signal state at parameter input R to "1". This cancels transmission to the module and resets the S_SEND instruction to its initial state. Data that has already been received by the module is still sent to the communication partner. If the signal state remains static at "1" at the input R, sending is disabled.

The address of the ET 200S Modbus/USS serial interface module to be addressed is specified at the LADDR parameter.

The DONE output indicates "Job completed without errors". ERROR indicates error events. If an error has occurred, the corresponding event number is displayed in STATUS. If no error occurs, STATUS has the value 0. DONE and ERROR/STATUS are also output at RESET of S_SEND. The binary result is reset if an error has occurred. If the block is completed without errors, the binary result has the status "1".

Modbus master read job

As the interface between the user program and the interface module operates in half-duplex mode you must observe the following items:

After positive acknowledgment of the Modbus master read job, you first have to fetch the receive data from the interface module by calling the S_RCV instruction before starting a new Modbus master send job.

Modbus slave

Initialization of the communication module

Introduction

The Modbus communication instructions ,(MODB_341, MODB_441 and S_MODB) for the loadable Modbus slave driver must be called in the cyclic part of the SIMATIC S7 CPU.

The Modbus communication instruction initializes the CP and executes the Modbus functions that the driver cannot perform independently. The Modbus communication instruction must then also be called in the user program if these function codes are not used by the Modbus master system.

Communication between the CP and the instruction is performed through the CPU operating system functions and instructions BSEND (CP 441-2) or P_SND_RK and P_RCV_RK (CP 341), which are called by the instruction.

Startup, initialization

The Modbus communication instruction must be initialized after a warm restart or hot restart of the CPU.

Initialization is triggered by a rising edge at the CP_START input.

The instruction first deletes the instance DB, then it reads the address areas I, O, M, T, C from the CPU and saves them in the instance DB. This allows the write requests from the Modbus master system to be tested to determine if they exceed the range.

With a send request, the CP is informed of the number of the instance DB and the previously successful initialization.

If the initialization ends with errors, no Modbus communication is possible. All requests from the Modbus master system are answered with an exception code message frame.

Instance DB

All data relating to the Modbus communication instruction are in an instance data block. This data block is both an instance DB (multi-instances) for the various instructions used, and a work area for the Modbus communication instruction. An additional data area is not required.

The Modbus communication instruction only works with the instance DB and with local data.

The instance DB is read-only.

Monitoring time

After power on, the CP needs several seconds for the hardware and memory test until it is ready. Initialization attempts made by the Modbus communication instruction during this time end with an error. Therefore, the initialization job is repeated several times during the monitoring period.

Read time interval SYSTAT (CP 441-2 only)

Since a regular SYSTAT read each cycle or every second cycle put an unnecessarily load on the CP and K bus and thereby reduce the data throughput, a time interval can be set for reading the SYSTAT (only relevant for CP 441-2 (MODB_441)).

Function codes

Modbus master (RTU)

Function Code 01 – Read Output Status

Function

This function enables individual bits to be read from the slave.

Start address

The **Bit start address** parameter is not checked by the driver and is sent unchanged.

Bit number

For **Bit number**, number of coils, any value between **1** and **2040** is permitted (for ET 200S 1SI Modbus up to 2008).

SEND source DB

Structure of SEND source area:

Address	Name	Type	Initial value	Comment
+0.0	address	BYTE	B#16#5	Slave address
+1.0	Function	BYTE	B#16#1	Function code
+2.0	bit_startadr	WORD	W#16#0040	Bit start address
+4.0	bit_number	INT	16	Bit number

Example

Request message frame FUNCTION 01:

05H	Slave address
01H	Function code
00H	Bit start address "High"
40H	Bit start address "Low"
00H	Bit number "High"
10H	Bit number "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Response message frame from slave FUNCTION 01:

05H	Slave address
01H	Function code
02H	Byte counter
01H	<Data>
17H	<Data>
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

RCV destination DB

Content of the RCV destination area:

Address	Name	Type	Current value	Comment
+0.0	data[1]	WORD	W#16#1701	Data

The driver enters the data of the response message frame into the destination DB **word by word**.

The 1st received byte is entered as the Low Byte of the 1st word “data[1],” the 3rd received byte as the Low Byte of the 2nd word “data[2],” etc.

If fewer than 9 bits or if only one Low Byte was read, the value **00H** is entered into the remaining High Byte of the last word.

Function code 02 - Read Input Status

Function

This function enables individual bits to be read from the slave.

Start address

The **Bit start address** parameter is not checked by the driver and is sent unchanged.

Bit number

For **Bit number**, number of coils, any value between **1** and **2040** is permitted (for ET 200S 1SI Modbus up to 2008).

SEND source DB

Structure of SEND source area:

Address	Name	Type	Initial value	Comment
+0.0	address	BYTE	B#16#5	Slave address
+1.0	Function	BYTE	B#16#2	Function code
+2.0	bit_startadr	WORD	W#16#0120	Bit start address
+4.0	bit_number	INT	24	Bit number

Example

Request message frame FUNCTION 02:

05H	Slave address
02H	Function code
01H	Bit start address "High"
20H	Bit start address "Low"
00H	Bit number "High"
18H	Bit number "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Response message frame from slave FUNCTION 02:

05H	Slave address
02H	Function code
03H	Byte counter
04H	<Data>
26H	<Data>
48H	<Data>
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

RCV destination DB

Content of the RCV destination area:

Address	Name	Type	Current value	Comment
+0.0	data[1]	WORD	W#16#2604	Data
+2.0	data[2]	WORD	W#16#0048	Data

The driver enters the data of the response message frame into the destination DB **word by word**.

The 1st received byte is entered as the Low Byte of the 1st word “data[1],” the 3rd received byte as the Low Byte of the 2nd word “data[2],” etc.

If fewer than 9 bits or if only one Low Byte was read, the value **00H** is entered into the remaining High Byte of the last word.

Function Code 03 - Read Output Registers

Function

This function enables you to read individual registers from the slave.

Start address

The **Register start address** parameter is not checked by the driver and is sent unchanged.

Register number

1 Up to a **maximum of 127 registers** (for ET 200S 1SI Modbus max. 125 registers) can be read (1 register = two bytes).

SEND source DB

Structure of SEND source area:

Address	Name	Type	Initial value	Comment
+0.0	address	BYTE	B#16#5	Slave address
+1.0	Function	BYTE	B#16#3	Function code
+2.0	reg_startadr	WORD	W#16#0040	Register start address
+4.0	reg_number	INT	2	Register number

Example

Request message frame FUNCTION 03:

05H	Slave address
03H	Function code
00H	Register start address "High"
40H	Register start address "Low"
00H	Register number "High"
02H	Register number "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Response message frame from slave FUNCTION 03:

05H	Slave address
03H	Function code
04H	Byte counter
21H	Register address 40H data "High"
23H	Register address 40H data "Low"
25H	Register address 41H data "High"
27H	Register address 41H data "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

RCV destination DB

Content of the RCV destination area:

Address	Name	Type	Current value	Comment
+0.0	data[1]	WORD	W#16#2123	Data
+2.0	data[2]	WORD	W#16#2527	Data

Function code 04 - Read Input Registers

Function

This function enables you to read individual registers from the slave.

Start address

The **Register start address** parameter is not checked by the driver and is sent unchanged.

Register number

1 Up to a **maximum of 127 registers** (for ET 200S 1SI Modbus max. 125 registers) can be read (1 register = two bytes).

SEND source DB

Structure of SEND source area:

Address	Name	Type	Initial value	Comment
+0.0	address	BYTE	B#16#5	Slave address
+1.0	Function	BYTE	B#16#4	Function code
+2.0	reg_startadr	WORD	W#16#0050	Register start address
+4.0	reg_number	INT	3	Register number

Example

Request message frame FUNCTION 04:

05H	Slave address
04H	Function code
00H	Register start address "High"
50H	Register start address "Low"
00H	Register number "High"
03H	Register number "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Response message frame from slave FUNCTION 04:

05H	Slave address
04H	Function code
04H	Byte counter
31H	Register address 50H data "High"
32H	Register address 50H data "Low"
33H	Register address 51H data "High"
34H	Register address 51H data "Low"
35H	Register address 52H data "High"
36H	Register address 52H data "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

RCV destination DB

Content of the RCV destination area:

Address	Name	Type	Current value	Comment
+0.0	data[1]	WORD	W#16#3132	Data
+2.0	data[2]	WORD	W#16#3334	Data
+4.0	data[3]	WORD	W#16#3536	Data

Function Code 05 - Force Single Coil

Function

This function serves to set or delete individual bits in the slave.

Bit address

The **bit address** parameter is not checked by the driver and is sent unchanged.

Bit Status

The following two values are permitted as the **bit status**:

- FF00H = Set bit
- 0000H = Delete bit

SEND source DB

Structure of SEND source area:

Address	Name	Type	Initial value	Comment
+0.0	address	BYTE	B#16#5	Slave address
+1.0	Function	BYTE	B#16#5	Function code
+2.0	bit_address	WORD	W#16#0019	Bit address
+4.0	bit_state	WORD	W#16#FF00	Bit Status

Example

Request message frame FUNCTION 05:

05H	Slave address
05H	Function code
00H	Bit address "High"
19H	Bit address "Low"
FFH	Set bit

00H	
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Response message frame from slave FUNCTION 05:

05H	Slave address
05H	Function code
00H	Bit address "High"
19H	Bit address "Low"
FFH	Bit status "High"
00H	Bit status "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

The response message frame is not entered in the receive DB.

Function code 06 - Preset Single Register

Function

This command enables a slave register to be overwritten with a new value.

Register address

The **Register address** parameter is not checked by the driver and is sent unchanged.

Register value

Any value can be used as the **Register value**.

SEND source DB

Structure of SEND source area:

Address	Name	Type	Initial value	Comment
+0.0	address	BYTE	B#16#5	Slave address
+1.0	Function	BYTE	B#16#6	Function code
+2.0	reg_address	WORD	W#16#0180	Register address
+4.0	reg_value	WORD	W#16#3E7F	Register value

Example

Request message frame FUNCTION 06:

05H	Slave address
06H	Function code
01H	Register address "High"
80H	Register address "Low"
3EH	Register value "High"
7FH	Register value "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Response message frame from slave FUNCTION 06:

05H	Slave address
06H	Function code
01H	Register address "High"
80H	Register address "Low"
3EH	Register value "High"
7FH	Register value "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

The response message frame is not entered in the receive DB.

Function Code 07 - Read Exception Status

Function

This function code enables 8 event bits to be read from the connected slave.

The start bit number of the event bit is determined by the connected device and therefore does not have to be specified by the SIMATIC user program

SEND source DB

Structure of SEND source area:

Address	Name	Type	Initial value	Comment
+0.0	address	BYTE	B#16#5	Slave address
+1.0	Function	BYTE	B#16#7	Function code

Example

Request message frame FUNCTION 07:

05H	Slave address
07H	Function code
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Response message frame from slave FUNCTION 07:

05H	Slave address
07H	Function code
3EH	<Data>
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

RCV destination DB

Content of the RCV destination area:

Address	Name	Type	Current value	Comment
+0.0	data[1]	WORD	W#16#3Exx	Data

The driver enters the individual bytes of the response message frame in the **High Byte** in the destination DB data[1].

The low byte of data[1] remains unchanged.

A value of 1 is displayed as the length in the LEN parameter of the receive instruction.

Function code 08 - Loop back Diagnostic Test

Function

This function is used to check the communications connection.

Only **Diagnostic code 0000** is supported with this function code!

Diagnostic Code

The only permissible value for the parameter **Diagnostic code** is 0000.

Test Value

Any value can be used as the **Test Value**.

SEND source DB

Structure of SEND source area:

Address	Name	Type	Initial value	Comment
+0.0	address	BYTE	B#16#5	Slave address
+1.0	Function	BYTE	B#16#8	Function code
+2.0	diag_code	WORD	W#16#0000	Diagnostic Code
+4.0	test_value	WORD	W#16#A5C3	Test Value

Example

Request message frame FUNCTION 08:

05H	Slave address
08H	Function code
00H	Diagnostic code "High"
00H	Diagnostic code "Low"
A5H	Test value "High"
C3H	Test value "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Response message frame from slave FUNCTION 08:

05H	Slave address
08H	Function code
00H	Diagnostic code "High"
00H	Diagnostic code "Low"
A5H	Test value "High"
C3H	Test value "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

The slave must return the request message frame received from the master unchanged as an echo.

The response message frame is not entered into an RCV DB.

Function code 11 - Fetch Communications Event Counter

Function

This function code is used to read a **"Status Word"** (2 bytes long) and an **"Event Counter"** (2 bytes long) from the slave.

The meaning of the above parameters is described in detail in the “GOULD MODICON Modbus Protocol.

SEND source DB

Structure of SEND source area:

Address	Name	Type	Initial value	Comment
+0.0	address	BYTE	B#16#5	Slave address
+1.0	Function	BYTE	B#16#0B	Function code

Example

Request message frame FUNCTION 11:

05H	Slave address
0BH	Function code
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Response message frame from slave FUNCTION 11:

05H	Slave address
0BH	Function code
FEH	Status Word "High"
DCH	Status Word "Low"
01H	Event Counter "High"
08H	Event Counter "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

RCV destination DB

Content of the RCV destination area:

Address	Name	Type	Current value	Comment
+0.0	data[1]	WORD	W#16#FEDC	Status Word
+2.0	data[2]	WORD	W#16#0108	Event Counter

Function Code 12 - Fetch Communications Event Log

Function

This function code enables the following to be read from the slave.

- 2 Byte **"Status Word"**
- 2 Byte **"Event Counter",**
- 2 Byte **"Message Counter"** and
- 64 Byte **"Event Bytes"**

The meaning of the above parameters is described in detail in the “GOULD MODICON Modbus Protocol.”

SEND source DB

Structure of SEND source area:

Address	Name	Type	Initial value	Comment
+0.0	address	BYTE	B#16#5	Slave address
+1.0	Function	BYTE	B#16#0C	Function code

Example

Request message frame FUNCTION 12:

05H	Slave address
0CH	Function code
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Response message frame from slave FUNCTION 12:

05H	Slave address
0CH	Function code
46H	Byte counter

87H	Status Word "High"
65H	Status Word "Low"
01H	Event Counter "High"
08H	Event Counter "Low"
02H	Message Counter "High"
20H	Message Counter "Low"
01H	Event Byte 1
12H	Event Byte 2
:	:
C2H	Event Byte 63
D3H	Event Byte 64
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

RCV destination DB

Content of the RCV destination area:

Address	Name	Type	Current value	Comment
+0.0	data[1]	WORD	W#16#8765	Status Word
+2.0	data[2]	WORD	W#16#0108	Event Counter
+4.0	data[3]	WORD	W#16#0220	Message Counter
+6.0	bytedata[1]	BYTE	B#16#01	Event Byte 1
+7.0	bytedata[2]	BYTE	B#16#12	Event Byte 2
:	:			:
+68.0	bytedata[63]	BYTE	B#16#C2	Event Byte 63
+69.0	bytedata[64]	BYTE	B#16#D3	Event Byte 64

Function code 15 - Force Multiple Coils

Function

With this function code, up to 2040 bits (for ET 200S 1SI Modbus max. 1744 bits) can be changed in the slave

Start address

The **Bit start address** parameter is not checked by the driver and is sent unchanged.

Bit number

For **Bit number**, number of coils, any value between **1** and **2040** is permitted (for ET 200S 1SI Modbus up to 1744).

This specifies how many bits in the slave are to be overwritten.

The "Byte counter" parameter in the request message frame is generated by the driver on the basis of the transferred parameter "Bit number".

SEND source DB

Structure of SEND source area:

Address	Name	Type	Initial value	Comment
+0.0	address	BYTE	B#16#5	Slave address
+1.0	Function	BYTE	B#16#0F	Function code
+2.0	bit_startadr	WORD	W#16#0058	Bit start address
+4.0	bit_number	INT	10	Bit number
+6.0	coil_state[1]	WORD	W#16#EFCD	Status Coil 5FH..58H/57H..50H

Example

Request message frame FUNCTION 15:

05H	Slave address
0FH	Function code
00H	Bit address "High"
50H	Bit address "Low"
00H	Bit number "High"
0AH	Bit number "Low"
02H	Byte counter
CDH	Status Coil 50H..57H
EFH	Status Coil 58H..59H
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Response message frame from slave FUNCTION 15:

05H	Slave address
0FH	Function code
00H	Bit address "High"
50H	Bit address "Low"
00H	Bit number "High"
0AH	Bit number "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

The response message frame is not entered in the receive DB.

Function code 16 - Preset Multiple Registers

Function

Function code 16 enables you to overwrite up to **127 registers** in the slave (for ET 200S Modbus 1SI max. 109 registers) with a request message frame.

Start address

The **Register start address** parameter is not checked by the driver and is sent unchanged.

Register number

1 Up to a **maximum of 127 registers** (for ET 200S 1SI Modbus max. 109 registers) can be read (1 register = two bytes).

The "Byte counter" parameter in the request message frame is generated by the driver on the basis of the transferred parameter "number of registers".

SEND source DB

Structure of SEND source area:

Address	Name	Type	Initial value	Comment
+0.0	address	BYTE	B#16#5	Slave address
+1.0	Function	BYTE	B#16#10	Function code
+2.0	reg_startadr	WORD	W#16#0060	Register start address
+4.0	reg_number	INT	3	Register number
+6.0	reg_data[1]	WORD	W#16#41A1	Register data
+8.0	reg_data[2]	WORD	W#16#42A2	Register data
+10.0	reg_data[3]	WORD	W#16#43A3	Register data

Example

Request message frame FUNCTION 16:

05H	Slave address
10H	Function code
00H	Register address "High"
60H	Register address "Low"
00H	Register number "High"
03H	Register number "Low"
06H	Byte counter
41H	<reg_data[1]> "High"
A1H	<reg_data[1]> "Low"
42H	<reg_data[2]> "High"

A2H	<reg_data[2]> "Low"
43H	<reg_data[3]> "High"
A3H	<reg_data[3]> "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Response message frame from slave FUNCTION 16:

05H	Slave address
10H	Function code
00H	Register address "High"
60H	Register address "Low"
00H	Register number "High"
03H	Register number "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

The response message frame is not entered in the receive DB.

Modbus slave (RTU)

Function Code 01 – Read Coil (Output) Status

Function

This function enables the MODBUS master system to read individual bits from the SIMATIC memory areas listed below.

Request message frame

ADDR	FUNC	start_address	bit_number	CRC

Response message frame

ADDR	FUNC	Byte_count n	n byte DATA	CRC

start_address

The MODBUS bit address "**start_address**" is interpreted by the driver as follows:

The driver checks that “start_address” is located within one of the areas which were specified during parameter assignment in the dialog box "**Conversion of MODBUS addressing for FC 01, 05, 15**" (from/to: memory bits, outputs, timers, counters).

If the MODBUS bit address **start_address** is located in one of these areas,	access is made to this **SIMATIC memory area**	
from aaaaa to bbbbb	commence at memory bit	M uuuuu.0
from ccccc to ddddd	commence at output	Q ooooo.0
From eeeee to fffff	commence at timer	T ttttt
From ggggg to hhhhh	commence at counter	C zzzzz

The address calculation for access, address conversion, is carried out as follows:

Access beginning with SIMATIC		Conversion formula			
Memory byte	=	((start_address	– aaaaa)	/ 8)	+ uuuuu
Output byte	=	((start_address	– ccccc)	/ 8)	+ ooooo
Timer	=	((start_address	– eeeee)	/ 16)	+ ttttt
Counters	=	((start_address	– ggggg)	/ 16)	+ zzzzz

Access to “memory bits” and “outputs”

When accessing SIMATIC areas “Memory bits” and “Outputs,” the remaining Rest Bit_Number is calculated and used to address the relevant bit within the first / last memory or output byte.

Access to "timers" and "counters"

With the address calculation, it must be possible to divide the result by 16 without having a left over value:

- (start_address - eeeee)
- (start_address - ggggg)

Only access word-by-word starting from word limit is possible.

bit_number

For **bit_number**, number of coils, values between **1** and **2040** (with ET 200S 1SI Modbus max. 1768 bits) are allowed. This number of bits is read.

Access to "timers" and "counters"

When accessing SIMATIC areas “Timers” and “Counters,” it must be possible to divide the "bit_number" by 16 (access word-by-word only).

Please note that a maximum of 16 “timers” and/or “counters” can be read if you are using the CP 341.

Note

Note the CPU-specific restrictions.

Application example

Example of parameter assignment:

Table 8-80 Conversion of MODBUS addressing for function codes FC 01, 05 and 15

MODBUS address in transmission message frame	SIMATIC memory area	
From *0* to *2047*	commence at memory bit	M *1000.0*
From *2048* to *2559*	commence at output	Q *256.0*
From *4096* to *4607*	commence at timer	T *100*
From *4608* to *5119*	commence at counter	C *200*

Request message frame FUNCTION 01:

05H	Slave address ADDR
01H	Function code FUNC
00H	start_address "High"
40H	start_address "Low"
00H	bit_number "High"
20H	bit_number "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Response message frame FUNCTION 01:

05H	Slave address ADDR
01H	Function code FUNC
04H	Byte_count
01H	<DATA 1> M 1008.0 - M 1008.7
17H	<DATA 2> M 1009.0 - M 1009.7
02H	<DATA 3> M 1010.0 - M 1010.7
18H	<DATA 4> M 1011.0 - M 1011.7
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Address calculation:

The MODBUS address “start_address” 0040 Hex (64 decimal) is located in the “memory bit” area:

Memory byte	=	((start_address	- aaaaa)	/ 8)	+ uuuuu
	=	((64	- 0)	/ 8)	+ 1000
	=	1008 ;			

The remaining rest bit_number has the following result:

Rest bit_no.	=	((start_address	- aaaaa)	% 8)	[Modulo 8]
	=	((64	- 0)	% 8)	
	=	0 ;			

Access is made starting from bit M 1008.0 up to and including M 1011.7.

Bit number:

The number of MODBUS bits "bit_number" 0020 Hex (32 decimal) means that 32 bits = 4 bytes are to be read.

Further examples

Some other access examples are listed in the table below.

All examples are based on the above area specification.

start_address		Access in SIMATIC beginning					->	with
Hex	Decimal	(Decimal)						
0000	0	Bit memory	((0	- 0)	/ 8)	+ 1000	->	M 1000.0
0021	33	Bit memory	((33	- 0)	/ 8)	+ 1000	->	M 1004.1
0400	1024	Bit memory	((1024	- 0)	/ 8)	+ 1000	->	M 1128.0
0606	1542	Bit memory	((1542	- 0)	/ 8)	+ 1000	->	M 1192.6
0840	2112	Output	((2112	- 2048)	/ 8)	+ 256	->	Q 264.0
09E4	2532	Output	((2532	- 2048)	/ 8)	+ 256	->	Q 316.4
1010	4112	Timers	((4112	- 4096)	/ 16)	+ 100	->	T 101
10C0	4288	Timers	((4288	- 4096)	/ 16)	+ 100	->	T 112
1200	4608	Counters	((4608	- 4608)	/ 16)	+ 200	->	C 200
13E0	5088	Counters	((5088	- 4608)	/ 16)	+ 200	->	C 230

Function code 02 - Read Input Status

Function

This function enables the MODBUS master system to read individual bits from the SIMATIC memory areas listed below.

Request message frame

ADDR	FUNC	start_address	bit_number	CRC

Response message frame

ADDR	FUNC	Byte_count n	n byte DATA	CRC

start_address

The MODBUS bit address "**start_address**" is interpreted by the driver as follows:

The driver checks that "start_address" is located within one of the areas which were entered during parameter assignment in the dialog box "**Conversion of MODBUS addressing for FC 02**" (from/to: memory bits, inputs).

If the MODBUS bit address is located in area **start_address** ,	**SIMATIC memory area** is accessed	
from kkkkk to lllll	commence at memory bit	M vvvvv.0
from nnnnn to rrrrr	from input	I sssss.0

The address calculation for access, address conversion, is carried out as follows:

Access beginning with Conversion formula
SIMATIC

Memory byte = ((start_address – *kkkkk*) / 8) + *vvvvv*

Input byte = ((start_address – *nnnnn*) / 8) + *sssss*

Access to “memory bits” and “inputs”

When accessing SIMATIC areas “Memory bits” and “Inputs,” the remaining rest bit_number is calculated and used to address the relevant bit within the first / last memory or input byte.

bit_number

For **bit_number**, number of coils, values between **1** and **2040** (with ET 200S 1SI Modbus max. 1768 bits) are allowed. This number of bits is read.

Note

Please note the CPU-specific restrictions.

Application example

Example of parameter assignment:

Table 8-81 **Conversion of MODBUS addressing for function code FC 02**

MODBUS address in transmission message frame	SIMATIC memory area	
from 0 to 4095	commence at memory bit	M 2000.0
from 4096 to 5119	commence at input	I 128.0

Request message frame FUNCTION 02:

05H	Slave address ADDR
02H	Function code FUNC
10H	start_address "High"
30H	start_address "Low"
00H	bit_number "High"
18H	bit_number "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Response message frame FUNCTION 02:

05H	Slave address ADDR
02H	Function code FUNC
03H	Byte_count
12H	<DATA 1> I 134.0 - I 134.7
34H	<DATA 2> I 135.0 - I 135.7
56H	<DATA 3> I 136.0 - I 136.7
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Address calculation:

The MODBUS address "start_address" 1030 Hex (4144 decimal) is located in the "Inputs" area:

Input byte	=	((start_address	- nnnnn)	/ 8)	+ sssss
	=	((4144	- *4096*)	/ 8)	+ *128*
	=	134 ;			

The remaining rest bit_number has the following result:

Rest bit_no.	=	((start_address	- nnnnn)	% 8)	[Modulo 8]
	=	((4144	- 4096)	% 8)	
	=	0 ;			

Inputs I 134.0 up to and including I 136.7 are accessed.

Bit number:

The number of MODBUS bits "bit_number" 0018 Hex (24 decimal) means that 24 bits = 3 bytes are to be read.

Further examples

Some other access examples are listed in the table below.

All examples are based on the above area specification.

start_address		Access in SIMATIC beginning					->	with
Hex	Decimal	(Decimal)						
0000	0	Bit memory	((0	- 0)	/ 8)	+ 2000	->	M 2000.0
0071	113	Bit memory	((113	- 0)	/ 8)	+ 2000	->	M 2014.1
0800	2048	Bit memory	((2048	- 0)	/ 8)	+ 2000	->	M 2256.0
0D05	3333	Bit memory	((3333	- 0)	/ 8)	+ 2000	->	M 2416.5
1000	4096	Input	((4096	- 4096)	/ 8)	+ 128	->	I 128.0
10A4	4260	Input	((4260	- 4096)	/ 8)	+ 128	->	I 148.4

Function Code 03 - Read Output Registers

Function

This function enables the MODBUS master system to read data words from a data block.

Request message frame

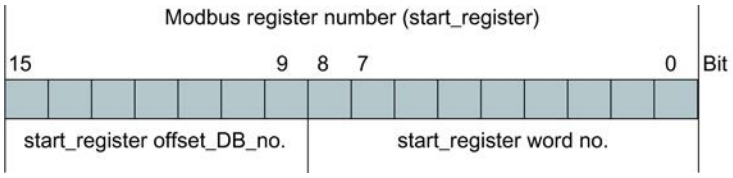
ADDR	FUNC	start_register	register_number	CRC

Response message frame

ADDR	FUNC	Byte_count n	n/2 register DATA (High, Low)	CRC

start_register

The MODBUS register address "**start_register**" is interpreted by the driver as follows:



For further address generation, the driver uses the “base DB number” (commencing at DB xxxxx) entered in the dialog box "**Conversion of MODBUS addressing for FC 03, 06, 16**" during parameter assignment .

The address calculation for access, address conversion, is carried out as follows:

Access to SIMATIC		Conversion formula	
Data block DB (resulting DB)	=	(base DB number *xxxxx*	+ start_register offset_DB_No.)
Data word DBW	=	(start_register word_No.	* 2)

Calculation formula for start_register

Providing the resulting DB to be read is known, the MODBUS address **start_register** required in the master system can be calculated in accordance with the following formula:

start_register = ((resulting DB – base DB number) * 512) + (data word_DBW / 2)

This is based on even numbered data word numbers only.

register_number

For **bit_number**, number of registers, values between **1** and **127** (with ET 200S 1SI Modbus max. 110 bits) are allowed. This number of registers is read. Please observe the following rule:

$(\text{register_number})_{max}$ = 512 - start_register

Note

Note the CPU-specific restrictions.

Application example

Example of parameter assignment:

Table 8-82 Conversion of MODBUS addressing for function codes FC 03, 06 and 16

MODBUS address in transmission message frame	SIMATIC memory area	
0	commence at data block (base DB number)	DB 800

Request message frame FUNCTION 03:

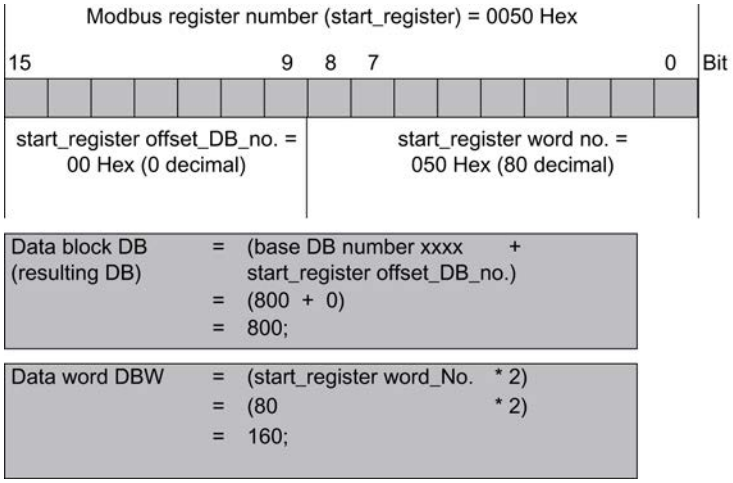
05H	Slave address ADDR
03H	Function code FUNC
00H	start_register "High"
50H	start_register "Low"
00H	register_number "High"
02H	register_number "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Response message frame FUNCTION 03:

05H	Slave address ADDR
03H	Function code FUNC
04H	Byte_count
87H	<DATA 1> DBW 160 "High"
65H	<DATA 2> DBW 160 "Low"
43H	<DATA 3> DBW 161 "High"
21H	<DATA 4> DBW 161 "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Address calculation:

The MODBUS address "start_register" 0050 Hex (80 decimal) is interpreted as follows:



DB 800, data word DBW 160 is accessed.

Register number:

The number of MODBUS registers "register_number" 0002 Hex (2 decimal) means that 2 registers = 2 data words are read.

Further examples

Some other access examples are listed in the table below.

			start_register				
start_register		Base DB no.	Offset DB_no.	Word number		Resulting DB	DBW
Hex	Decimal	Decimal	Decimal	Hex	Decimal	Decimal	Decimal
0000	0	800	0	000	0	800	0
01F4	500	800	0	1F4	500	800	1000
0200	512	800	1	000	0	801	0
02FF	767	800	1	0FF	255	801	510
0300	768	800	1	100	256	801	512
03FF	1023	800	1	1FF	511	801	1022
0400	1024	800	2	000	0	802	0

Function code 04 - Read Input Registers

Function

This function enables the MODBUS master system to read data words from a data block.

Request message frame

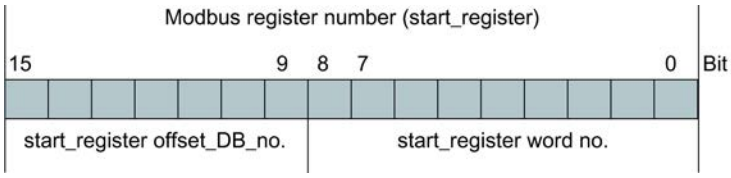
ADDR	FUNC	start_register	register_number	CRC

Response message frame

ADDR	FUNC	Byte_count n	n/2 register DATA (High, Low)	CRC

start_register

The MODBUS register address "**start_register**" is interpreted by the driver as follows:



For further address generation, the driver uses the “base DB number” (commencing at DB xxxxx) specified in the dialog box "**Conversion of MODBUS addressing for FC4**" .

The address calculation for access, address conversion, is carried out in two steps as follows:

Access to SIMATIC		Conversion formula
Data block DB (resulting DB)	=	(base DB number *xxxxx* + start_register-offset_DB_no.)
Data word DBW	=	(start_register word_No. * 2)

Calculation formula for start_register

Providing the resulting DB to be read is known, the MODBUS address **start_register** required in the master system can be calculated in accordance with the following formula:

start_register = ((resulting DB – base DB number) * 512) + (data word_DBW / 2)

Only even numbered data word numbers are permissible.

register_number

For **bit_number**, number of registers, values between **1** and **127** (with ET 200S 1SI Modbus max. 110 bits) are allowed. This number of registers is read. Please observe the following rule:

$(\text{register_number})_{max} = 512 - \text{start_register}$

Note

Note the CPU-specific restrictions.

Application example

Example of parameter assignment:

Table 8-83 Conversion of MODBUS addressing for function code FC 04

MODBUS address in transmission message frame	SIMATIC memory area	
0	commence at data block (base DB number)	DB 900

Request message frame FUNCTION 04:

05H	Slave address ADDR
04H	Function code FUNC
02H	start_register "High"
C0H	start_register "Low"
00H	register_number "High"
03H	register_number "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

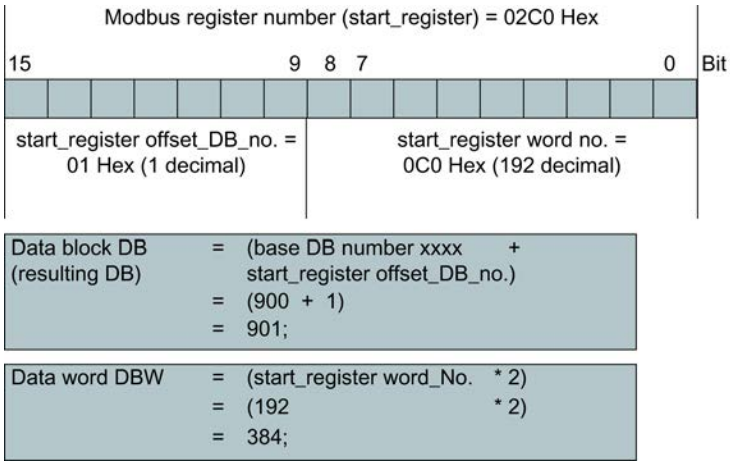
Response message frame FUNCTION 04:

05H	Slave address ADDR
04H	Function code FUNC
06H	Byte_count
A1H	<DATA 1> DBW 384 "High"
A2H	<DATA 2> DBW 384 "Low"
A3H	<DATA 3> DBW 385 "High"

A4H	<DATA 4> DBW 385 "Low"
A5H	<DATA 5> DBW 386 "High"
A6H	<DATA 6> DBW 386 "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Address calculation:

The MODBUS address "start_register" 02C0 Hex (704 decimal) is interpreted as follows:



DB 901, data word DBW 384 is accessed.

Register number:

The number of MODBUS registers "register_number" 0003 Hex (3 decimal) means that 3 registers = 3 data words are read.

Further examples

Some other access examples are listed in the table below.

			start_register				
start_register		Base DB no.	Offset DB_no.	Word number		Resulting DB	DBW
Hex	Decimal	Decimal	Decimal	Hex	Decimal	Decimal	Decimal
0000	0	900	0	000	0	900	0
0064	100	900	0	064	100	900	200
00C8	200	900	0	0C8	200	900	400
0190	400	900	0	190	400	900	800
1400	5120	900	10	000	0	910	0

			start_register				
start_register		Base DB no.	Offset DB_no.	Word number		Resulting DB	DBW
Hex	Decimal	Decimal	Decimal	Hex	Decimal	Decimal	Decimal
1464	5220	900	10	064	100	910	200
14C8	5320	900	10	0C8	200	910	400

Function Code 05 - Force Single Coil

Function

This function enables the MODBUS master system to write a bit into the SIMATIC memory of the CPU as listed below.

Request message frame

ADDR	FUNC	coil_address	DATA on/off	CRC

Response message frame

ADDR	FUNC	coil_address	DATA on/off	CRC

coil_address

The MODBUS bit address "**coil_address**" is interpreted by the driver as follows:

The driver checks whether "coil_address" is located within one of the areas which were entered during parameter assignment in the dialog box "**Conversion of MODBUS Addressing for FC 01, 05, 15**" (from/to: memory bits, outputs, timers, counters).

If the MODBUS bit address **coil_address** is located in the area	the **SIMATIC memory area** is accessed	
from aaaaa to bbbbb	commence at memory bit	M uuuuu.0
from ccccc to ddddd	commence at output	Q ooooo.0

The address calculation for access (address conversion) is carried out as follows:

Access beginning with SIMATIC		Conversion formula			
Memory byte	=	((coil_address	- *aaaaa*)	/ 8)	+ *uuuuu*
Output byte	=	((coil_address	- *ccccc*)	/ 8)	+ *ooooo*

Access to “memory bits” and “outputs”

When accessing the SIMATIC areas "memory bits" and "outputs", the remaining rest bit_number is calculated and used to address the relevant bit within the memory or output byte.

Access to "timers" and "counters"

Access to the SIMATIC timers and counters areas is not permitted with function code FC 05 and is rejected by the driver with an error message frame.

DATA on/off

The following two values are permitted as **DATA-on/off**:

- FF00H -> Set bit.
- 0000H -> Delete bit.

Application example

Example of parameter assignment:

Table 8-84 Conversion of MODBUS addressing for function codes FC 01, 05, 15

MODBUS address in transmission message frame	SIMATIC memory area	
from 0 to 2047	commence at memory bit	M 1000.0
from 2048 to 2559	commence at output	Q 256.0

Request message frame FUNCTION 05:

05H	Slave address ADDR
05H	Function code FUNC
08H	coil_address "High"
09H	coil_address "Low" A257.1
FFH	DATA on/off "High"
00H	DATA on/off "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Reply message frame FUNCTION 05:

05H	Slave address ADDR
05H	Function code FUNC
08H	coil_address "High"
09H	coil_address "Low" A257.1
FFH	DATA on/off "High"
00H	DATA on/off "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Address calculation:

The MODBUS address "coil_address" 0809 Hex (2057 decimal) is located in the "outputs" area:

```
Output byte    =  ((coil_address   - ccccc)  / 8)   + ooooo
               =  ((2057           - 2048)   / 8)   + 256
               =   257 ;
```

The remaining rest bit_number has the following result:

```
Rest bit_no.   =  ((coil_address   - ccccc)  % 8)   [Modulo 8]
               =  ((2057           - 2048)   % 8)
               =   1 ;
```

Output Q 257.1 is accessed.

Further examples

For further examples of access to memory bits and outputs, refer to FC 01.

Function code 06 - Preset Single Register

Function

This function enables the MODBUS master system to write a data word in a data block of the CPU.

Request message frame

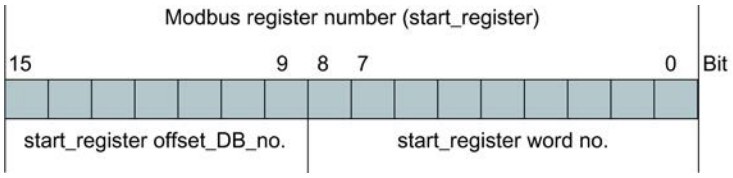
ADDR	FUNC	start_register	DATA value (High, Low)	CRC

Response message frame

ADDR	FUNC	start_register	DATA value (High, Low)	CRC

start_register

The MODBUS register address "**start_register**" is interpreted by the driver as follows:



For further address generation, the driver uses the "base DB number" commencing at DB xxxxx entered in the dialog box "**Conversion of MODBUS addressing for FC 03, 06, 16**" during parameter assignment .

The address calculation for access, address conversion, is carried out in two steps as follows:

Access to SIMATIC		Conversion formula
Data block DB (resulting DB)	=	(base DB number xxxxx + start_register-offset_DB_no.)
Data word DBW	=	(start_register word_No. * 2)

Calculation formula for start_register

Providing the resulting DB to be written is known, the MODBUS address **start_register** required in the master system can be calculated in accordance with the following formula:

start_register = ((resulting DB – base DB number) * 512) + (data word_DBW / 2)

Only even numbered data word numbers are permissible.

DATA Value

Any value can be used as the DATA **value**, register value.

Application example

Example of parameter assignment:

Table 8-85 Conversion of MODBUS addressing for function codes FC 03, 06 and 16

MODBUS address in transmission message frame	SIMATIC memory area	
0	commence at data block (base DB number)	DB 800

Request message frame FUNCTION 06:

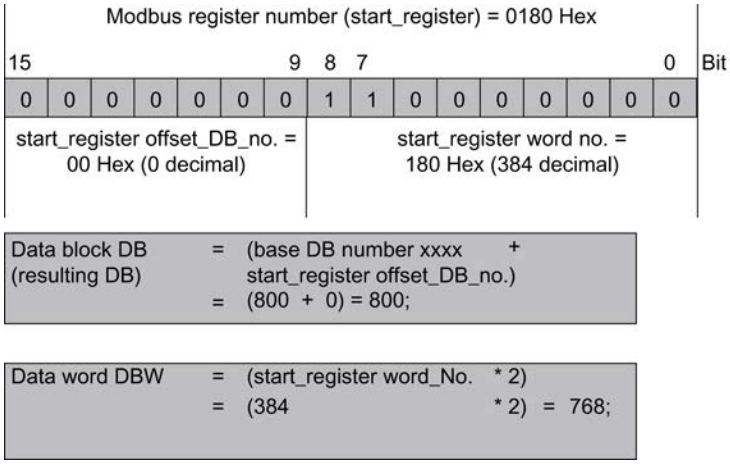
05H	Slave address ADDR
06H	Function code FUNC
01H	start_register "High"
80H	start_register "Low" DBW 768
2BH	DATA value "High"
1AH	DATA value "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Reply message frame FUNCTION 06:

05H	Slave address ADDR
06H	Function code FUNC
01H	start_register "High"
80H	start_register "Low" DBW 768
2BH	DATA value "High"
1AH	DATA value "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Address calculation:

The MODBUS address "start_register" 0180 Hex (384 decimal) is interpreted as follows:



DB 800, data word DBW 768 is accessed.

Further examples

For further access examples, please refer to FC 03.

Function code 08 - Loop back Diagnostic Test

Function

This function is used to check the communications connection.

It has no effect on the S7 CPU, the user programs or user data. The received message frame is returned to the master system by the driver independently.

Request message frame

ADDR	FUNC	Diagnostic code (High,Low)	Test Data	CRC

Response message frame

ADDR	FUNC	Diagnostic code (High,Low)	Test Data	CRC

Diagnostic Code

Only **Diagnostic Code 0000** is supported!

Test Data

Any16-bit value.

Application example

Request message frame FUNCTION 08:

05H	Slave address ADDR
08H	Function code FUNC
00H	Diagnostic code "High"
00H	Diagnostic code "Low"
A5H	Test value "High"
C3H	Test value "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Reply message frame FUNCTION 08:

05H	Slave address ADDR
08H	Function code FUNC
00H	Diagnostic code "High"
00H	Diagnostic code "Low"
A5H	Test value "High"
C3H	Test value "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Function code 15 - Force Multiple Coils

Function

This function enables the MODBUS master system to write several bits to the SIMATIC memory areas listed below.

Request message frame

ADDR	FUNC	start_address	quantity	byte_count n	n DATA	CRC

Response message frame

ADDR	FUNC	start_address	quantity	CRC

start_address

The MODBUS bit address "**start_address**" is interpreted by the driver as follows:

The driver checks whether “start_address” is located within one of the areas which were entered during parameter assignment in the dialog box "**Conversion of MODBUS addressing for FC 01, 05, 15**" (from/to: memory bits, outputs, timers, counters).

If the MODBUS bit address is located in area **start_address** ,	**SIMATIC memory area** is accessed	
from aaaaa to bbbbb	commence at memory bit	M uuuuu.0
from ccccc to ddddd	commence at output	Q ooooo.0

The address calculation for access, address conversion, is carried out as follows:

Access beginning with SIMATIC		Conversion formula
Memory byte	=	((start_address - aaaaa) / 8) + uuuuu
Output byte	=	((start_address - ccccc) / 8) + ooooo

Access to “memory bits” and “outputs”

When accessing the SIMATIC areas "memory bits" and "outputs", the remaining rest bit_number is calculated and used to address the relevant bit within the memory or output byte.

Access to "timers" and "counters"

Access to the SIMATIC timers and counters areas is not permitted with function code FC 15 and is rejected by the driver with an error message frame.

Quantity

For **quantity**, bit number, any value between **1** and **2040** (with ET 200S 1SI Modbus max. 1976 bits) is allowed.

Note

Please note the CPU-specific restrictions.

DATA

Any values are contained in the **DATA field** as bit statuses.

Application example

Example of parameter assignment:

Table 8-86 Conversion of MODBUS addressing for function codes FC 01, 05 and 15

MODBUS address in transmission message frame	SIMATIC memory area	
from 0 to 2047	commence at memory bit	M 1000.0
from 2048 to 2559	commence at output	Q 256.0

Action:

The MODBUS master system wants to write the following bit statuses on to memory bits M 1144.0 ... M 1144.7 and M 1145.0 ... M 1145.3:

Memory bit M 1144	7	6	5	4	3	2	1	0	Bit
	ON	ON	OFF	OFF	ON	ON	OFF	ON	
Memory bit M 1145	7	6	5	4	3	2	1	0	Bit
	-	-	-	-	ON	OFF	OFF	ON	

Request message frame FUNCTION 15:

05H	Slave address ADDR
0FH	Function code FUNC
04H	start_address "High"
80H	start_address "Low" (M 1144.0 ...)
00H	quantity "High"
0CH	quantity "Low" (12 bits)
02H	bytecount
CDH	status coil (M 1144.0 ... M 1144.7)
09H	status coil (M 1145.0 ... M 1145.3)
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Response message frame FUNCTION 15:

05H	Slave address ADDR
0FH	Function code FUNC

04H	start_address "High"
80H	start_address "Low"
00H	quantity "High"
0CH	quantity "Low"
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Address calculation:

The MODBUS address "coil_address" 0480 Hex (1152 decimal) is located in the "memory bit" area:

```
Memory byte   =  ((start_address  - aaaaa)  / 8)   + uuuuu
              =  ((1152           - 0)      / 8)   + 1000
              =   1144 ;
```

The remaining rest bit_number has the following result:

```
Rest bit_no.  =  ((start_address  - aaaaa)  % 8)  [Modulo 8]
              =  ((1152           - 0)      % 8)
              =   0 ;
```

Access is made to memory bits commencing at M 1144.0.

Further examples

For further examples of access to memory bits and outputs, refer to FC 01.

Function code 16 - Preset Multiple Registers

Function

The function code enables the MODBUS master system to write several data words in a data block of the SIMATIC CPU.

Request message frame

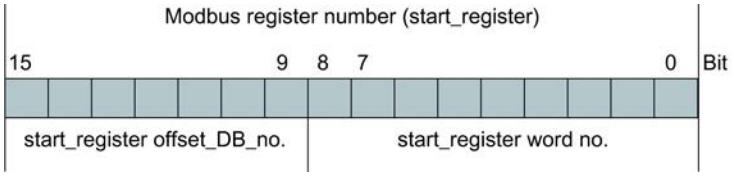
ADDR	FUNC	start_register	quantity	byte_count n	n-DATA (High,Low)	CRC

Response message frame

ADDR	FUNC	start_register	quantity	CRC

start_register

The MODBUS register address "**start_register**" is interpreted by the driver as follows:



For further address generation, the driver uses the "base DB number" commencing at DB xxxxx entered in the dialog box "**Conversion of MODBUS addressing for FC 03, 06, 16**" during parameter assignment .

The address calculation for access, address conversion, is carried out in two steps as follows:

Access to SIMATIC	Conversion formula
Data block DB (resulting DB)	= (base DB number *xxxxx* + start_register-offset_DB_no.)
Data word DBW	= (start_register word_No. * 2)

Calculation formula for start_register

Providing the resulting DB to be written is known, the MODBUS address **start_register** required in the master system can be calculated in accordance with the following formula:

start_register = ((resulting DB – base DB number) * 512) + (data word_DBW / 2)

Only even numbered data word numbers are permissible.

Quantity

For **quantity**, register number, any value between **1** and **127** (with ET 200S 1SI Modbus max. 123 bits) is allowed. Please observe the following rule:

$(\text{Quantity})_{max}$ = 512 - start_register

Note

Please note the CPU-specific restrictions.

DATA (High,Low)

Any value can be used as DATA **DATA (High,Low)** (register value).

Application example

Example of parameter assignment:

Table 8-87 Conversion of MODBUS addressing for function codes FC 03, 06, 16

MODBUS address in transmission message frame	SIMATIC memory area	
0	commence at data block (base DB number)	DB 800

Action:

The MODBUS master system wants to write the values CD09 Hex, DE1A Hex and EF2B Hex to data words DBW 100, DBW 102 and DBW 104 of DB 800.

Request message frame FUNCTION 16:

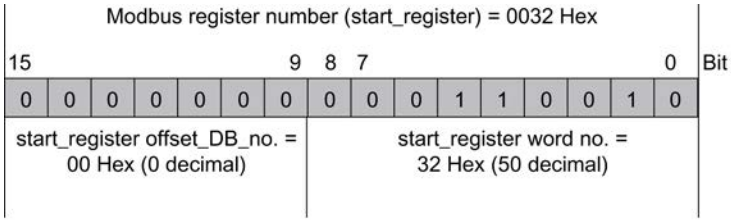
05H	Slave address ADDR
10H	Function code FUNC
00H	start_register "High"
32H	start_register "Low" DBW 100
00H	quantity "High"
03H	quantity "Low" (3 registers)
06H	bytecount
CDH	register value -High (DBW100)
09H	register value -Low
DEH	register value -High (DBW102)
1AH	register value -Low
EFH	register value -High (DBW104)
2BH	register value -Low
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Response message frame FUNCTION 16:

05H	Slave address ADDR
10H	Function code FUNC
00H	start_register "High"
32H	start_register "Low"
00H	quantity "High"
03H	quantity "Low" (3 registers)
xxH	CRC checksum "Low"
xxH	CRC checksum "High"

Address calculation:

The MODBUS address "start_register" 0032 Hex (50 decimal) is interpreted as follows:



Data block DB (resulting DB)	=	(base DB number xxxx + start_register offset_DB_no.)
	=	(800 + 0) = 800;

Data word DBW	=	(start_register word_No. * 2)
	=	(50 * 2) = 100;

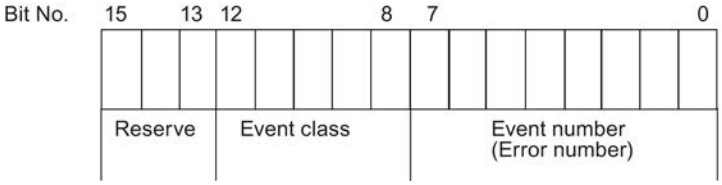
DB 800, data word DBW 100 is accessed.

Further examples

For further access examples, please refer to FC 03.

Error message area

Numbering scheme event class / event number



Event classes

Event class	Meaning
2	Error initializing
5	Error executing a CPU job
8	Receive errors
14 ($0E_H$)	General processing error
30 ($1E_H$)	Error during communication between serial port and the CPU.

Event class / event number

Error code(W#16 #...)	Description	Remedy
0201	No (valid) configuration available.	Provide the module with correct parameters. If necessary, ensure the system is correctly installed.
0502	Order inadmissible in this operating mode of the serial interface module ET 200S Modbus/USS (example: device interface is not configured).	Evaluate the diagnostic interrupt and rectify the error accordingly.
050E	Invalid telegram length	The send message frame is longer than 224 bytes. The send job was aborted by the ET 200S Modbus/USS module. Select a shorter telegram length.
0518	• Transmission length during sending is too large (> 4 Kbyte) • The transmission length during SEND is too small.	Check LEN parameter on SEND.
0530	Modbus master send job rejected because the response of the communication partner to a previous reading Modbus master send job has not yet been retrieved.	After a successful read Modbus master send job, you must first read the response from the communication partner from the module before you start a new Modbus master send job.
0551	Frame sequence errors in the communication between the serial interface module ET 200S Modbus/USS and the CPU. The error occurred while transmitting a received message frame of the ET 200S SI serial interface module to the CPU.	The module and the CPU have canceled the transmission. Repeat the receive job. The ET 200S Modbus/USS serial interface module sends the received message again.
0806	Character delay time ZVZ exceeded	Error by the partner device or repair disruptions on the transmission line.
080A	Overflow of the receive buffer at the master during the reception of the response message frame.	Check the protocol settings of the slave.
080C	A transmission error was recognized in a character (parity error, overrun error, stop bit error (frame)).	Check to see if the disruptions have an effect on the transmission line. If applicable, change the system structure and the cable routing. Check to see if the protocol parameters for data transmission rate, data bit number, parity, stop bit number are identically set for the communication module and connection partner.
080D	BREAK The receive line to the partner device is disrupted.	Establish the connection between the devices or switch on the partner device. During TTY mode, check to see if the line current is flowing in idle state. If applicable, check and change the default setting of the 2-wire receive line R(A), R(B) with an RS422/485 (X27) connection.
0810	Parity error: There is a break in the connection (line break) between the two communication partners.	Check the connection line of the communication partners or verify that both devices are configured with the same data transmission rate, parity and stop bit number. Change your system setup or cable wiring.

Error code(W#16 #...)	Description	Remedy
0811	Character frame error: There is a break in the connection (line break) between the two communication partners.	Check the connection line of the communication partners or verify that both devices are configured with the same data transmission rate, parity and stop bit number. Change your system setup or cable wiring.
0812	Additional characters were received after the CTS serial port was set to OFF.	Reconfigure the communications partner or fetch from the serial interface faster.
0830	The response monitoring time after sending a request message frame expired without the beginning of the response message frame being recognized.	Check if transmission line is interrupted (interface analyzer may be required). Check to see if the protocol parameters for data transmission rate, data bit number, parity, stop bit number are identically set for the communication module and connection partner. In the properties dialog of the module, check if the configured response monitoring time is set high enough. Check to see if the specified slave address is available.
0831	The first character of the response message frame from the slave is different from the slave address sent in the request message frame (with "normal" operating mode).	The wrong slave has replied. Check if transmission line is interrupted (interface analyzer may be required).
0832	Overflow of the receive buffer in the communication module upon receipt of the response message frame.	Check the protocol settings of the slave.
0833	Bit or register number for function code FC 15/16 and message frame element byte_count do not match.	Correct the bit/register number or byte_count.
0834	Illegal bit coding for "Set bit / Reset bit" recognized.	Use only the 0000Hex or FF00Hex codes for FC05.
0835	Illegal diagnostic subcode (not 0000Hex) at function code FC 08 "Loop back test" recognized.	Use only the subcode 0000Hex for FC08.
0836	The internally formed value of the CRC 16 checksum does not match the received CRC checksum.	Check the formation of the CRC checksum on the Modbus master system.
0837	Message frame sequence error: The Modbus master system sent a new request message frame before the last response message frame was transmitted by the driver.	Increase the timeout on the slave response message frame for Modbus master system.
0850	Length of the receive message frame greater than 224 bytes or greater than the configured message frame length.	Adjust the message frame length of the partner.
0E01	Error during initialization of the driver-specific SCC process.	Reassign parameters of driver and reload.
0E02	Error during startup of driver: Wrong SCC process active (SCC driver). The driver cannot function with this SCC driver.	Reassign parameters of driver and reload.
0E03	Error during startup of driver: Wrong data transmission process active (interface to instructions). The driver cannot work with this data transmission process.	Reassign parameters of driver and reload.
0E04	Error during startup of driver: Illegal interface submodule. The driver cannot run with the configured interface submodule.	Check and correct the parameter assignment.

Error code(W#16 #...)	Description	Remedy
0E05	Error with driver dongle: No dongle plugged in, or inserted dongle is faulty. The driver is not ready to run.	Check if a driver dongle is plugged into the communication module. If the inserted dongle is faulty, replace it with a functioning dongle.
0E06	Error with driver dongle: The dongle has no valid content. The driver is not ready to run.	Obtain a correct dongle from the SIEMENS office which supplied you with the driver.
0E10	Internal error procedure: Default branch in procedure automatic device.	Restart the communication module (POWER_ON)
0E11	Internal error procedure: default branch for procedure status Send / Receive.	Restart the communication module (POWER_ON)
0E12	Internal error active automatic device: Default branch	Restart the communication module (POWER_ON)
0E13	Internal error passive automatic device: Default branch	Restart the communication module (POWER_ON)
0E20	The number of data bits for this connection must be 8. The driver is not ready to run.	Correct the parameter assignment of the driver.
0E21	The multiplication factor set for the character delay time is not within the value range of 1 to 10. The driver is operating with a default setting of 1.	Correct the parameter assignment of the driver.
0E22	The operating mode set for the driver is illegal. “Normal operation” or “Interference Suppression” must be specified. The driver is not ready to run.	Correct the parameter assignment of the driver.
0E23	An illegal value for the response monitoring time has been set: Valid values are 5 to 65500ms. The driver is not ready to run.	Correct the parameter assignment of the driver.
0E24	Invalid limits were configured for write access. The driver is not ready.	Correct the parameter assignment of the driver.
0E25	An invalid "from/to" combination was configured in the specification of the "Conversion of the Modbus addresses with function code FC 01, 05, 15" area. (Memory bits, outputs, timers, counters area). The driver is not ready.	Correct the parameter assignment of the driver.
0E26	An invalid "from/to" combination was configured in the specification of the "Conversion of the Modbus addresses with function code FC 02" area. (Memory bits, inputs area). The driver is not ready.	Correct the parameter assignment of the driver.
0E27	An overlap was configured in the specification of the "Conversion of the Modbus addresses with function code FC 01, 05, 15" area. (Memory bits, outputs, timers, counters area). The driver is not ready.	Correct the parameter assignment of the driver.
0E28	An overlaps was configured in the specification of the "Conversion of the Modbus addresses with function code FC 02" area. (Memory bits, inputs area). The driver is not ready.	Correct the parameter assignment of the driver.

Error code(W#16 #...)	Description	Remedy
0E2E	An error occurred when reading the interface parameter file. The driver is not ready to run.	Restart the communication module (POWER_ON).
0E30	Internal error during data transmission to CPU: Unexpected acknowledgment Passive.	Can be ignored if it happens intermittently.
0E31	Timeout during data transmission to CPU.	Check CP-CPU interface.
0E32	Error occurred during data transmission to CPU with RCV: Exact failure reason (detailed error) is in SYSTAT before this entry.	Check CP-CPU interface.
0E33	Internal error during data transmission to CPU: Illegal status of automatic device	Check CP-CPU interface.
:		
0E38	An error has occurred accessing SIMATIC areas "Memory bit, output, timers, counters, input" with the function code FC 01 or FC 02: For example, input not available or attempt to read beyond the end of the area.	Check whether the addressed SIMATIC area exists and that no attempt was made to access beyond the end of the area.
0E39	An error has occurred accessing the SIMATIC "Data block" area with the function code FC 03, 04, 06, 16: Data block not available or too short.	Check if the addressed data block exists and is sufficiently long.
0E3A	An error has occurred executing a write job with the function code FC 05, 15: Instance data of the Modbus instruction is not available or too short.	Check if the configured instance DB for the Modbus communication instruction is available and is sufficiently long.
0E3B	Timeout (waiting for acknowledgement) in the execution of a write job by the Modbus communication instruction.	Check the connection configuration as well as CP-CPU interface (BSEND), you may need to reload the Modbus communication instruction.
0E3C	Illegal job with this driver.	Only SEND, RCV, STATUS (CP 441-2 only) permitted
0E40	Value specified for LEN parameter in the SEND instruction is too low.	Minimum length is 2 bytes.
0E41	Value specified for LEN parameter in the SEND instruction is too low. A greater length is required for the transferred function code.	The minimum length for this function code is 6 bytes.
0E42	Transferred function code is illegal.	The only function codes permitted are those listed in the section " Function codes (Page 1101) ".
0E43	Slave address 0 (= Broadcast) not permitted with this function code.	Only use slave address 0 for the suitable function codes.
0E44	The value of the transferred “Bit number” parameter is not within the range 1 to 2040.	The “Bit number” parameter must be within the range 1 to 2040.
0E45	The value of the transferred “Register number” parameter is not within the range 1 to 127.	The “Register number” parameter must be within the range 1 to 127.
0E46	Function codes 15 or 16: The value of the transferred “Bit number” or “Register number” parameter is not within the range 1 to 2040 and/or 1 to 127.	The “Bit number” or “Register number” parameter must be within the range 1 to 2040 and/or 1 to 127.

Error code(W#16 #...)	Description	Remedy
0E47	Function codes 15 or 16: The LEN parameter for the SEND instruction does not correspond to the transferred “Bit number” or “Register number” parameter, LEN parameter is too small.	Increase the LEN parameter for SEND until a sufficient amount of user data is transferred to the communication module. A larger amount of user data must be transferred to the communication module because of the “Bit number” or “Register number”.
0E48	Function code 05: The code specified in SEND source DB for “Set Bit” (FF00H) or “Delete Bit” (0000H) is wrong.	The only permitted codes are FF00H and 0000H.
0E49	Function code 08: The code specified in SEND source DB for “Diagnostic Code” is wrong.	The only permitted code is “Diagnostic Code” 0000H.
0E50	In the word-based SIMATIC timers/counters areas, the resulting residual bit number from the Modbus address is ≠ 0.	Only use Modbus addresses that result in valid bit numbers.
0E51	The received Modbus address is outside the configured "from/to" areas.	Only use addresses as address information in the request message frame that have previously been defined in the configuration.
0E52	• A SIMATIC area limit was exceeded during an access attempt by the Modbus master system: Resulting DB number < 1 • Write access to an unreleased area (configuration) • Write access to the instance DB of the communication instruction	Access area limited to valid SIMATIC memory areas.
0E53	A SIMATIC area limit was exceeded during an access attempt by the Modbus master system: For example, overflow from the formation of the resulting DB no. (> 65535)	Access area limited to valid SIMATIC memory areas.
0E54	Access beyond the configured area end or access beyond the SIMATIC area end.	Access area limited to valid SIMATIC memory areas.
0E55	Write access to this SIMATIC memory area is not allowed.	Write only to the SIMATIC memory bits, outputs data areas.
0E56	The coupling cannot be operated because the communication instruction is not running.	Call the Modbus communication instruction in the STEP 7 user program cyclically. You may need to repeat initialization of the communication instruction.
0E57	An error has occurred executing the Modbus function code in the communication instruction.	Analyze the exact cause.
0E4A	The length of this function code is greater than the maximum length.	The manual contains the maximum lengths for each function code.
0E4F	CP 441: The R_ID specified in the SEND instruction is invalid for this driver.	Only use R_IDs 0 to 255 (00 00 00 00 ... 00 00 00 FFH)
	CP 341: The R_TYPE specified at the SEND_RK instruction is invalid for this driver.	"X" has to be entered as R_TYP.

Error code(W#16 #...)	Description	Remedy
0E50	Slave address incorrect: The received slave address is different from the sent slave address.	The wrong slave has replied. Check if transmission line is interrupted (interface analyzer may be required).
0E51	Function code incorrect: The function code received in the response message frame is different from the sent function code.	Check the slave device.
0E52	Byte Underflow: Amount of characters received is less than should have resulted from the byte counter of the response message frame, or is less than expected with this function code.	Check the slave device.
0E53	Byte Overflow: Amount of characters received is more than should have resulted from the byte counter of the response message frame, or is more than expected with this function code.	Check the slave device.
0E54	Byte counter too small: The byte counter received in the response message frame is too small.	Check the slave device.
0E55	Byte counter wrong: The byte counter received in the response message frame is wrong.	Check the slave device.
0E56	Echo wrong: The data of the response message frame (amount of bits, ...) echoed from the slave are different from the data sent in the request message frame.	Check the slave device.
0E57	CRC Check incorrect: An error has occurred on checking the CRC 16 checksum of the response message frame from the slave.	Check the slave device.
0E61	Response message frame with exception code 01: Illegal function	See “Manual of Slave Device”
0E62	Response message frame with exception code 02: Illegal data address	See “Manual of Slave Device”
0E63	Response message frame with exception code 03: Illegal data value	See “Manual of Slave Device”
0E64	Response message frame with exception code 04: Failure in associated device	See “Manual of Slave Device”
0E65	Response message frame with exception code 05: Acknowledge	See “Manual of Slave Device”
0E66	Response message frame with exception code 06: Busy, Rejected message	See “Manual of Slave Device”
0E67	Response message frame with exception code 07: Negative Acknowledgement	See “Manual of Slave Device”
1E0D	"Job aborted due to warm restart, hot restart or reset"	
1E0E	Static error calling instruction DP_RDDAT. The RET_VAL return value of the instruction is provided to you in the SFCERR tag in the instance DB for evaluation.	Load the SFCERR tag from the instance DB.
1E0F	Static error calling instruction DP_WRDAT. The RET_VAL return value of the instruction is provided to you in the SFCERR tag in the instance DB for evaluation.	Load the SFCERR tag from the instance DB.

Error code(W#16 #...)	Description	Remedy
1E10	Static error calling instruction RD_LGADR. The RET_VAL return value of the instruction is provided to you in the SFCERR tag in the instance DB for evaluation.	Load the SFCERR tag from the instance DB.
1E11	Static error calling instruction RDSYSST. The RET_VAL return value of the instruction is provided to you in the SFCERR tag in the instance DB for evaluation.	Load the SFCERR tag from the instance DB.
1E20	Parameter out of range.	Enter a parameter within the range valid for the function module.
1E41	Number of bytes specified in the LEN parameter of the instruction invalid.	You must stay within a range of values of 1 to 256 bytes.

USS communication

Overview of USS communication

Position in the system environment

The following USS description refers to the operation of the ET 200S 1SI module in the Modbus/ USS model.

Introduction

The USS protocol is a straightforward serial data transmission protocol designed to meet the requirements of drive technology.

The USS protocol defines an access method based on the master-slave principle for communication via a serial bus. One master and up to 31 slaves can be connected to the bus. Master and slaves form a USS network. The data exchanged between master and slave are referred to as network data. The network data block contains the data for the master. The individual slaves are selected by the master using an address character in the message frame. A slave can never send anything without first being initiated by the master. Therefore, direct data transmission between individual slaves is not possible. Communication functions in half-duplex mode. The master function cannot be transferred. The USS system has only one master.

Message frame structure

Each message frame begins with a start character (STX), followed by the length specification (LGE) and the address byte (ADR). The data field comes after that. The message frame ends with the block check character (BCC).

STX	LGE	ADR	1	2	...	N	BCC

For single-word (16-bit) data in the network data block, the high byte is sent first, followed by the low byte. Correspondingly, with double-word data the high word is sent first, followed by the low word.

The protocol does not identify any tasks in the data fields.

Data encryption

The data is encrypted as follows:

- STX: 1 byte, start of text, 02H
- LGE: 1 byte, contains the message frame length as a binary number
- ADR: 1 byte, contains the slave address and message frame type in binary code
- Data fields: One byte each, contents are task-dependent
- BCC: 1 byte, block check character

Data transmission procedure

The master ensures cyclic data transmission in message frames. The master addresses all slave nodes one after another with a task message frame. The nodes addressed respond with a response frame. In accordance with the master-slave procedure, the slave must send the response frame to the master after it has received the task message frame. Only then can the master address the next slave.

General structure of the network data block

The network data block is split into two areas: the parameter area (PKW) and the process data area (PZD).

STX	LGE	ADR	Parameter (PKW)	Process data (PZD)	BCC

- **Parameter area** (PKW)
  The PKW area handles parameter transmission between two communication partners (e.g., the controller and drive). This involves, for example, reading and writing parameter values and reading parameter descriptions and the associated text. The PKW interface generally contains tasks for operation and display, maintenance and diagnostics.
- **Process data area** (PZD)
  The PZD area consists of signals that are required for automation:
  - Control words and setpoints from the master to the slave
  - Status words and actual values from the slave to the master

  The contents of the parameter area and process data area are defined by the slave drives. For additional information on this, refer to the drive documentation.

USS-master

Overview of functions

Network data transmission sequence

The instructions handle network data transmission in cycles with up to 31 drive slaves, in accordance with the sequence specified in the polling list (configuration DB). Only one job is active for each slave at any one time. The user stores the network data for each slave in a data block (network data block) and calls it from there. As specified in the program definition in the polling list, the data is transmitted to and called from the communications processor via another data storage area (communications processor DB).

Two function calls are required for this procedure (one send and one receive instruction). Another function supports the generation and presetting of the data blocks required for communication.

Performance features:

- Creation of data storage areas for communication, depending on the bus configuration
- Presetting of the polling list
- Message frame structure in accordance with the USS specification
- Network data exchange can be parameterized in accordance with the required network data structure
- Execution and monitoring of PKW jobs
- Handling of parameter change reports
- Monitoring of the complete system and error elimination

Various network data structures can be used to send network data. In other words, the content of network data can be selected according to a given application.

Depending on the structure selected, the network data has a PZD area for process data and a PKW area for parameter processing.

In the PKW area, the master can read and write parameter values, and the slave can display parameter changes by means of parameter change reports.

The PZD area contains signals required for process control, such as control words and setpoints from the master to the slave, and status words and actual values from the slave to the master.

The correct sequence for instruction calls is: S_USST, S_SEND, S_RCV, S_USSR. This is important because the outputs of the S_SEND and S_RCV instructions are only valid in the current cycle of the automation system.

The figure below shows the data traffic between the user program and the USS slave.

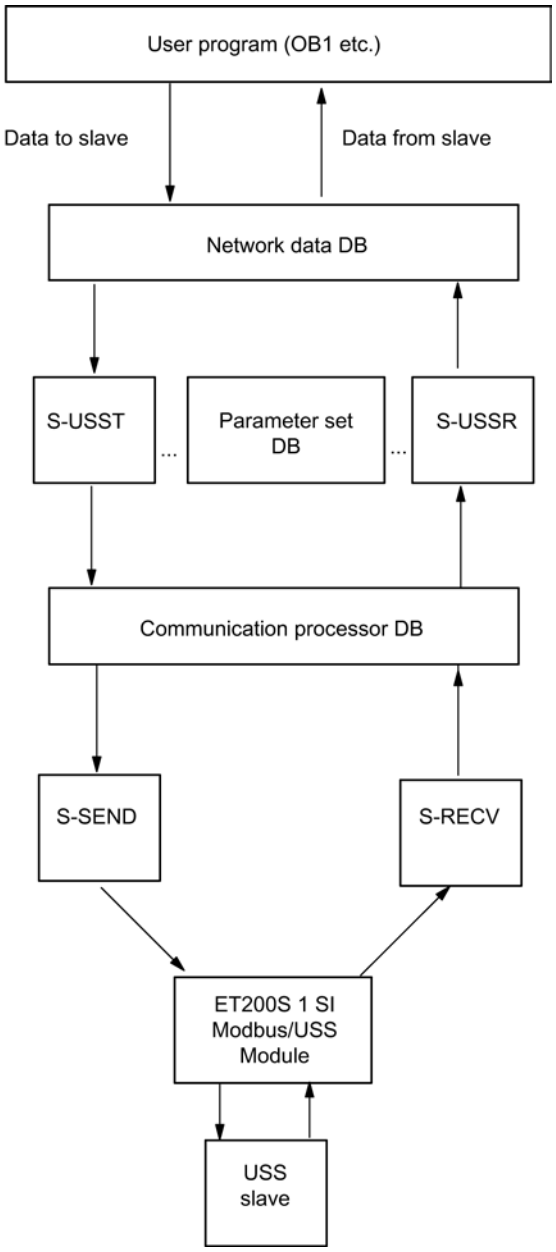


Figure 8-47 Data traffic between the user program and the USS slave

Further Information

Information about the data blocks used can be found here:

- Network data DB (Page 7704)
- Network data DB (Page 7709)
- Communication processor DB (Page 7710)

Startup and Diagnostics

Startup characteristics

Startup characteristics

The start of the communication module is divided into the following phases:

- Initialization (POWER ON of the communication module)
- Parameter assignment

Initialization

Once the communication module is connected to power and after the completion of a hardware test program, the firmware is prepared for the operation on the communication module.

Parameter assignment

During the parameter assignment phase, the communication module receives the module parameters assigned to the current slot. The communication module is now operational.

Operating mode transitions

After the communication module starts up, all data between the CPU and the communication module is exchanged via instructions. The operating state transition characteristics of the communication module depends on the operating mode of the CPU.

CPU STOP	Active data transmission from the communication module to the CPU and both sending and receiving jobs are stopped and a warm restart of the connection is initiated.
CPU RUN	Unlimited send and receive operation is possible in the RUN state of the CPU. The communication module and the given instruction are synchronized in the first instruction cycle following CPU warm restart,.
CPU warm restart	The jobs on the CPU are reset during a warm restart of the CPU; in other words, all active jobs between the CPU and the communication module are automatically cancelled.
Hot restart of the CPU	With a hot restart of the CPU, jobs continue to be processed. With a corresponding configuration in the properties dialog of the module, you can automatically clear the receive buffer on the communication module during CPU startup.

Diagnostic functions

Introduction

The diagnostic functions of the communication module allow errors that have occurred to be located quickly. The following diagnostic options are available to you:

Diagnostics via the display elements of the communication module	The indicators provide information on the operating state or the possible error states of the communication module. The indicators provide an initial overview of any internal or external errors and interface-specific errors.
Diagnostics via the STATUS output of the instructions	Instructions have a STATUS output for error diagnostics. This output provides information about communication errors between the communication module and the CPU. You can evaluate the STATUS parameter in the user program.
Diagnostics on the SYSTAT error message area [1]	You can query the status of an interface by programming the STATUS instruction in the user program. By reading the SYSTAT, you obtain detailed information about errors/events that have occurred in communication between the communication module and the associated CPU, as well as the connected communication partner on this interface.
Diagnostics via the error numbers in the response message frame	If you are using the RK512 computer connection and an error occurs with a SEND or FETCH message frame at the communication partner, the communication partner sends a response message frame with an error number in the 4th byte.
Diagnostics via the diagnostic buffer of the communication module	All errors/events are also entered in the diagnostic buffer [2] of the communication module. Similar to the CPU diagnostic buffer, you can also display user-specific information from the CP diagnostic buffer in plain text.
Diagnostic interrupt	The communication module can trigger a diagnostic interrupt on the CPU assigned to it. The communication module provides 4 bytes of diagnostic information. The analysis of this information is made via the user program or by reading the CPU diagnostic buffer.

[1] CP 441-1 and CP 441-2 only

[2] CP 340, ET 200S 1SI modules and CPU31xC-2 PtP have no diagnostic buffer

8.1.4.7 Configuring PC stations

What you need to know about PC stations

Introduction

PC-based controllers provide the same functionality as the SIMATIC S7-CPUs S7-300 or S7-400 (hardware controllers).

The SIMATIC PC station is a PC or an OS station that can contain SIMATIC components such as applications (e.g. WinCC) for automation tasks.

Comparison: SIMATIC S7 with a SIMATIC PC station

Due to the functional differences between the SIMATIC PC station and SIMATIC S7, additional terms are used for the PC stations:

- The PC station in the device and network editor is a device that can host hardware components such as CPs as well as software components, for example, applications or OPC servers.
- Analogous to SIMATIC S7, SIMATIC PC stations are in the form of a module rack and show the contained slots, where the modules and applications are arranged. In contrast to the SIMATIC S7 configuration, the SIMATIC PC station has a virtual array and does not correspond to a real slot allocation.
- Only the device (CPU, CP, etc.) is displayed in the network view of an SIMATIC S7-300. The modules or components that you insert in the device view are not displayed in the network view, nor can they be edited in the network view.
  With the PC station in the network view, you will also see all the components you have added in the device view of the PC station. You can select individual components in the network or device view and edit them in the Inspector window.
- With the virtual slot allocation, an "index" is used instead of an "address" or slot number to differentiate the terminology. Similar to a slot, the index can accommodate exactly one object from the hardware catalog.

Required operating mode

There are different areas of application for commissioning and operating the PC station. Select the appropriate operating mode depending on the area of application:

- Programming device mode: Default operating mode for programming device (PG/PC) and HMI station. In this operating mode, the PC station is used to program and test other devices.
- Configuration mode: Operating mode in which the PC station is configured for an automation task and with which other automation systems can communicate.

Overview: Planning configurations and connections for a SIMATIC PC-station

With the "SIMATIC PC station" station type, the complete configuration of a PC station is done. The components configured here are then also available for configuring connections. The following components can be configured in a SIMATIC PC station:

- Software PLC WinLC (starting with version V4.5)
- Slot PLC CPU 41x-2 PCI (WinAC Slot 412 and WinAC Slot 416)
- Communication modules for the PC
- SIMATIC HMI: WinCC and Protool Pro
- Applications (for example on SAPI based user applications)
- OPC server

Additional information

For additional information about the concept for PC stations, refer to the configuration manual, "SIMATIC NET - Commissioning PC Stations".

Software components of a PC station

A PC application contains a range of configurable properties, which are adopted in the PC station with the download. There are two different usage scenarios, which have different effects on the generation and handling of the project engineering data:

- OPC server as central communication service provider
- Standard application

You configure specific properties for this component to fulfill this role and provide the services, which are used in a variety of ways.

OPC server as central communication service provider

OPC clients use the interface to an OPC server to communicate with an automation device such as a SIMATIC S7-300.

Configure its properties to adapt the OPC server. The parameters set or changed here take effect after the project engineering data is downloaded to the PC station and the OPC server is booted.

You can use the following communication services depending on the module in use:

- All connection types
- DP services (DPV0)

Standard application

With standard applications, you communicate with other applications and devices via your PC module.

You can configure the following communication services depending on the module in use:

- S7 connections
- DP services

Additional information

For additional information about the meaning of PC applications and the OPC server and how to handle them, refer to the manuals "SIMATIC NET - Commissioning PC Stations" and "SIMATIC NET - Industrial Communication".

See also

Communication connections to PC stations (Page 595)

Slot rules for a PC station

Introduction

Components such as CPs, PC-based user programs for connection configuration and OPC servers can be plugged into slots (index) 1 to 32. Certain slot rules apply for PC stations.

Inserting an OPC server

In PC stations, OPC servers can be inserted for OPC clients that access remote automation systems via configured connections:

- A maximum of one OPC server can be inserted in a PC station for OPC clients.
- If an OPC server is plugged into a PC station, a CP can be assigned to it.

Other standard applications can be used instead of the OPC server or in addition to an OPC server.

Plugging components

The following rules apply to index numbering and plugging components into the PC station:

- The index numbers 1 to 32 can be assigned.
- The first plugged component always gets index 1.
- Additional plugged components get the next free index number.
- If a component is deleted, the index numbers of the other modules remain unchanged. There may be gaps in index numbering for this reason.
- Inserting components between existing components is permissible.
- The following applies for inserting components when components are already present:
  - If free index numbers are available between two components, the first free index will be assigned.
  - If no free index numbers are available between two components, the index of all subsequent components will be incremented by one number until the first free index. A component with index 32 can in this way be shifted to the area of unplugged modules. You can therefore expand the PC station without losing components with important settings in the event of over-allocation.

Inserting submodules

WinLC RTX supports as a sub-module

- a maximum of one CP 5611/21 card or an integrated CP 5611/21 interface
- a maximum of one PROFINET interface (CP 1616, CP 1604, among others)
- a maximum of four sub-modules altogether. From the four submodules, there may be any number of CP 5613 communication interfaces.

The following rules apply when a component has submodules:

- A controller contains a fixed number of free slots.
- Free slots are shown in the graphic and tabular device view.
- Submodules can only be inserted on free sub slots.
- A subslot is freed when a submodule is deleted.
- Subslots do not reserve an index number.

See also

Area for unplugged modules (Page 501)

Configuring PC stations online

Introduction

If you want to commission PC modules for the first time, you need to perform an initial configuration. Following the initial configuration of the PC modules, the PC station is set up for receiving configuration data.

You can perform an initial configuration for a PC station available online and then load the project engineering data in the configured PC station.

Note

If the PC station is not available online, the initial configuration can be made with an XDB file (Page 1159).

Requirement

You must be in the network view.

Procedure

To configure a PC station online, proceed as follows:

1. Drag a device for a PC station into the network view from the hardware catalog under "PC systems".
2. Select the PC station.
3. Change the name of the PC station to suit your needs under "Properties > General" in the Inspector window.

4. Drag-and-drop the additional components required into the PC station from the hardware catalog. An added component will be inserted on a new index.

   Note

   You can change the index of a selected component in the Inspector window under "Properties > General > Position number". There may be gaps in index numbering for this reason. These gaps are not displayed in the graphic presentation of the device view. However, gaps are displayed in the tabular area of the device view.

5. Select the PC station.
6. Perform the initial configuration and/or download the project engineering data:
   - To perform the initial configuration of the PC station, select the shortcut menu command, "Configure PC station online".
     In the "Configure" dialog box you can make various settings for your local network, the available computers and the target PC station.
   - To download the project engineering data, select the shortcut menu command, "Load to device".
     The PC station must already have an initial configuration to download the project engineering data.

See also

Device view (Page 479)

Configuring PC stations offline

Introduction

If you want to commission PC modules for the first time, you need to perform an initial configuration. Following the initial configuration of the PC modules, the PC station is set up for receiving configuration data.

You can perform the initial configuration of a PC station offline. The project engineering data for the initial configuration and operation are saved in this case in an XDB file as the basic data for the PC station.

Note

If the PC station is available online, the initial configuration and transfer of the project engineering data can be done directly online (Page 1158).

Requirement

You must be in the network view.

Procedure

Proceed as follows to configure a PC station using an XDB file:

1. Drag a device for a PC station into the network view from the hardware catalog under "PC systems".
2. Select the PC station.
3. Change the name of the PC station to suit your needs under "Properties > General" in the Inspector window.
4. Drag-and-drop the additional components required into the PC station from the hardware catalog. An added component will be inserted on a new index.

   Note

   You can change the index of a selected component in the Inspector window under "Properties > General > Position number". There may be gaps in index numbering for this reason. These gaps are not displayed in the graphic presentation of the device view. However, gaps are displayed in the tabular area of the device view.

5. Select the PC station.
6. Under "Properties > XDB Configuration" in the Inspector window, activate the "Generate XDB file" check box.
   The connection data and addresses for CPs and applications are saved in the XDB file.
7. Specify the path for saving the XDB file.
8. Select the PC station.
9. Select the shortcut menu command "Compile > ...".
   When saving and compiling the configuration of a PC station system data and the XDB configuration file are generated that you can then load into the target system, or that must be installed.

   Note

   The location of the configuration file is set on the target PC station via the program "Set PG-PC interface" (in the "STEP 7 Configuration" tab).

See also

Device view (Page 479)

Create memory card file

With this menu command, you can create a new memory card file.

Procedure

Proceed as follows to create a memory card file:

1. Select the "New" command in the "Project > Memory Card-File" menu.
   A dialog opens.
2. Select the desired directory.
3. Enter the file name.
4. Confirm with "Create".

Result

The memory card file is shown in the project navigation.

You can copy blocks to the memory card file by copying or moving them from different block folders, libraries or an S7-memory card.

Open existing file

To display an existing memory card file in the project navigation, select the "Open" command in the "Project > Memory Card-File" menu.

Special features of the S7-modular Embedded Controller

Introduction and basic procedure

Introduction

S7-modular Embedded Controllers (S7-mEC) are PCs in S7-300 design with preinstalled and preconfigured software:

- Windows Embedded Standard operating system
- WinAC RTX (F)
- SIMATIC NET
- WinCC Runtime (only with EC31-HMI/RTX)

Ethernet interfaces of the S7-modular Embedded Controller

An S7-modular Embedded Controller has two integrated Industrial Ethernet interfaces:

- The X1 PN LAN P1 / X1 PN LAN P2 interface can be configured as a PROFINET interface.
- The X2 P1 PN/IE LAN interface is assigned to the PC station in index 3 at IE General and is preconfigured for Industrial Ethernet communication.

Note

If accessing S7-mEC via Remote Desktop Protocol, ensure that S7-mEC and the Engineering PC/programming device are connected to the same Ethernet subnet.

To access S7-mEC via Remote Desktop protocol, you have to use the IE General network connection (usually X2 P1 PN/IE LAN interface).

Basic configuration and programming of the S7-mEC

Since an S7-mEC is a preconfigured system, follow these steps:

1. Create a project.
2. Insert a PC station.
3. Configure the hardware.
4. Create an S7 program.
5. Configure visualization in WinCC for the EC31-HMI/RTX.
6. Download the configuration data to the WinLC RTX EC.

EC31-HMI/RTX: Configuring visualization

For EC31-HMI/RTX only

Create the visualization project for the EC31-HMI/RTX on the engineering PC, using SIMATIC WinCC.

Requirement

- WinCC is installed on the engineering PC.
- A WinCC RT object had been inserted in the PC station.

Procedure

In order to create a visualization project for EC31-HMI/RTX, follow these steps:

1. Open the device view of the PC station of the EC31-HMI/RTX.
2. Select WinCC RT object.
3. Select the "Open object" command.
   WinCC is started.

4. Create and configure the necessary elements: Pictures, alarms, tags, etc., using the WinCC editors.
5. Save and test the visualization.
6. Transfer the project to the EC31-HMI/RTX target system.

Downloading the configuration data to the target system

Download options

You can download the following configuration data from STEP 7 to the embedded controller:

- The entire PC station (including, for example, IE General)
- Only the configuration data of the WinLC RTX / WinLC RTX F
- Only the S7 program of the WinLC RTX / WinLC RTX F

For an EC31-HMI/RTX, you will need to transfer the configuration data from WinCC.

Requirement

- The enhanced write filter (EWF) must be deactivated on the Embedded Controller if the entire PC station is to be downloaded.
- The embedded controller is connected to the programming device / PC via Industrial Ethernet. The PG/PC interface must be set to TCP/IP.
- **EC31-HMI/RTX only:** The visualization project has been created in WinCC.
- Before the following functions become available, the embedded controller must be connected to the engineering system via a port of the X1 PN LAN interface:
  - Assign PROFINET IO device names
  - PROFINET CBA: Interconnection download and online functions

Procedure

1. Deactivate EWF on the EC31-RTX / EC31-RTX F / EC31-HMI/RTX (if not already deactivated). After EWF has been deactivated, turn off and restart the embedded controller.
2. Start WinLC.
3. Activate the PC station in the Station Configuration Editor ("Enable station" button).
4. Change the embedded controller to STOP mode.
5. Download the hardware configuration, the S7 program and the configuration data to the target system.

Making ready for operation

To make the EC31-RTX / EC31-HMI/RTX ready for operation, follow these steps:

1. Set the enhanced write filter for drive C:\ to the "activated" state.

NOTICE
Flash memory failure If you leave the enhanced write filter in "deactivated" state, you risk an early failure of your flash memory due to the persistent write access of the operating system. When the enhanced write filter is activated, the number of write accesses to the operating system partition is minimized, thereby increasing the life of the flash memory.

2. Record all special settings (such as changes to the IP addresses) made in the course of application development. Store them along with your project.
3. Create a backup copy of your settings.

8.1.4.8 S7-1200 CM/CP

S7-1200 CM/CP

Telecontrol S7-1200

CP 1242-7

Copying the CP 1242-7 into another project

If you copy a CP 1242-7 from one project into another project, the following parameters in the parameter group "CP identification" are changed on the target station:

- Project number of the CP
- Station number of the CP

Download to device

Only use the "Download to device" function with the CP 1242-7 via a TeleService connection as follows:

1. Select the CP in STEP 7.
2. Select the "Online" > "Download to device" menu.
3. In the "Extended download" dialog that appears, select the TeleService interface.
4. Download the project data from the "Extended download" dialog.

Upload from device

The function "Upload from device" is not supported by the CP 1242-7.

CP 1242-7 configuration

Modes of the GPRS-CP

Modes of the CP

The CP 1242-7 allows an S7-1200 to communicate as a GPRS station with a central station or other remote networks via the GSM network. For communication using GPRS, the CP is set to one of the following operating modes:

- Telecontrol
  This operating mode of the CP allows the GPRS station to exchange data with the following partners:
  - Communication with the Telecontrol server
    This CP mode allows the GPRS station to exchange data with a telecontrol server. The Telecontrol server is a PC connected to the Internet with the "TELECONTROL SERVER BASIC" application. It is generally located in the master station and serves to monitor and control the remote GPRS stations. Data can be exchanged with the OPC client of a central control system via the integrated OPC interface.
    The Telecontrol server PC is not configured in STEP 7. The "TELECONTROL SERVER BASIC" application has its own configuration user interface.
  - Communication with another remote GPRS station
    The message frames are transmitted via the Telecontrol server.
  - Communication with an engineering station (for TeleService)

  Communication with the Telecontrol server is performed via the GSM network and the Internet.
  This operating mode requires a SIM card with GPRS service enabled and a Telecontrol server that can be reached by the CP.
- GPRS direct
  This operating mode of the CP is used for direct communication between remote stations via the GSM network. No Telecontrol server is required.
  To allow network nodes in public wireless networks to be directly accessible, these need to be addressed using a fixed address. Here, SIM cards with a fixed IP address are used that allow the stations to address each other directly.
  The possible communications services and security functions (for example VPN) depend on what is offered by the network provider.
  Possible communications partners of the GPRS station with a CP 1242--7 in "GPRS direct" mode are:
  - A subscriber that can be reached by the CP via an IP address (GPRS station with CP 1242-7)
  - An engineering station (for TeleService)

Establishing a connection using GPRS communication

Connection modes

- "GPRS direct" mode
  There are no different connection modes in the "GPRS direct" mode.
- "Telecontrol" mode
  The CP can be configured for the following connection modes.
  - "Permanent" connection mode
    There is a permanent TCP connection to the telecontrol server. Following connection establishment, there is a permanent TCP connection to the telecontrol server even if data is not transferred permanently.
  - "Temporary" connection mode
    A connection is only established to the telecontrol server when required.

If a TCP connection is established, process data is sent as soon as the telecontrol instructions are called on the CPU.

A connection is always established by the CP. If a connection established by the CP is interrupted, the CP automatically attempts to re-establish the connection.

Triggering connection establishment for permanent stations ("Telecontrol" mode)

In the "Telecontrol" mode, the permanent connection to the telecontrol server is established when the station starts up. If there is an interruption on the connection, the establishment of the connection can be triggered by a wake-up SMS (see below).

Triggering connection establishment for temporary stations ("Telecontrol" mode)

With "temporary" stations, connection establishment can be triggered by the following events:

- Event on the local CPU that needs to be evaluated by the program.
  In terms of the program, two situations need to be distinguished:
  - Events that lead to a single connection establishment (for example alarms or commands from the operator).
  - Expiry of an interval that leads to cyclic connection establishment (for example once daily for data transmission)
- Request by a communications partner (OPC client or S7 station)
  The request from the communications partner leads to the connection being established.
- Request for TeleService by an engineering station
  The request switched by the telecontrol server or TeleService gateway does not need to be evaluated in the program.
- Wake-up SMS of the telecontrol server
  The wake-up SMS can be triggered spontaneously on the telecontrol server. It is also possible to configure cyclic sending on the telecontrol server.

- Telephone wake-up call
  The wake-up call can be sent from a telephone that has a phone number authorized in the STEP 7 project. The telephone must support the CLIP function (transfer of its own call number).
  The connection establishment with the (main) telecontrol server is triggered.
- Telephone wake-up SMS
  The wake-up SMS can be sent from a telephone that has a phone number authorized in the STEP 7 project. The telephone must support the CLIP function (transmission of its own number) and sending of SMS.
  The connection establishment with the telecontrol server specified in the SMS is triggered.

When a temporary station is woken up, all the data is transferred if this has changed since the last data transfer.

Triggering connection establishment in "GPRS direct" mode

In "GPRS direct" mode, a connection establishment is triggered by the following events:

- Event on the local CPU that is evaluated by the program.
- Request by a communications partner (not an engineering station)
  The request in the frame received from the communications partner is evaluated in the program by calling the telecontrol instructions.
- Request for TeleService by an engineering station
  The request switched by the telecontrol server or TeleService gateway does not need to be evaluated in the program.

Right to wake-up by "authorized phone numbers"

The CP only accepts an SMS if the sending communication partner is authorized based on its phone number. These numbers are in configured for the CP in STEP 7 in the "authorized phone numbers" list.

Note

"Authorized phone numbers" in the STEP 7 project

- A phone number entered here gives the sender who transfers this phone number the right to trigger connection establishment.
- If an asterisk (*) is entered in the list, the CP accepts SMS messages from all senders.
- If the list is empty, the CP cannot be woken up for connection establishment.

Wake-up SMS

Depending on the connection type and the triggering server or intermediary TeleService server, the following text must be transferred in the wake-up SMS:

- For telecontrol connections:
  - Text for the wake-up SMS message for establishing a connection to the telecontrol server:
    `TELECONTROL`
  - Text for the wake-up SMS message for establishing a connection to the main telecontrol server:
    `TELECONTROL MAIN`
  - Text for the wake-up SMS message for establishing a connection to the substitute telecontrol server:
    `TELECONTROL BACKUP`

  The configuration of the telecontrol server for the GPRS CP is set in STEP 7 in "Telecontrol interface > Operating mode > main or substitute telecontrol server".

  Note

  Wake-up with a mobile phone

  - One of the texts listed above can be used in a wake-up SMS message.
  - With a wake-up call, the station always connects to the main telecontrol server.

- For TeleService connections:
  - Text for the wake-up SMS message for establishing a connection to the first configured TeleService server:
    `TELESERVICE`
    or
    `TELESERVICE 1`
  - Text for the wake-up SMS message for establishing a connection to the second configured TeleService server:
    `TELESERVICE 2`

  The configuration of the TeleService server for the GPRS CP is set in STEP 7 in "Telecontrol interface > TeleService authorization > 1st or 2nd TeleService server".

Preferred GSM networks

Selection of preferred GSM networks

The following options are provided to select the GSM networks that the CP 1242-7 should preferably dial up:

- Automatic dialup
  The CP dials with highest priority into the GSM network of the contracted network operator set on the SIM card. If a connection cannot be made within the contracted network, the CP dials up another GSM network with which the network operator has a roaming contract and whose access data is stored on the SIM card.
- Contract network only
  The CP only dials up the GSM network of the contracted network operations whose SIM card is plugged into the CP. No roaming.
- Contracted network and alternative network
  The CP dials up the contracted network with the highest priority. If dialup to the contracted network is unsuccessful, the CP dials up an alternative GSM network, in decreasing priority, as entered in the list of preferred network operators.
  The alternative networks are entered in the list as "Public Land Mobile Network" (PLMN). PLMN is a construct of Mobile Country Code (MCC) and Mobile Network Code (MNC).
  Example: 26276
  This is the PLMN for the test network of Siemens AG with MCC = 262 and MNC = 76.

Time-of-day synchronization in NTP mode - parameter assignment

NTP mode (NTP: Network Time Protocol)

In NTP mode, the CP sends time-of-day queries at regular intervals to one or more NTP servers. From the responses of the servers, the CP selects the most accurate time of day.

The advantage of this mode is that it allows the time to be synchronized across subnets.

In NTP mode, it is generally UTC (Universal Time Coordinated) that is transferred. This corresponds to GMT (Greenwich Mean Time).

CP 1242-7 TeleService

TeleService via GPRS

Communication path for TeleService via GPRS

With TeleService for remote S7 stations with the CP 1242-7 via GPRS, the connection is always established via an intermediary between the engineering station and remote S7 station.

This intermediary can be either:

- A telecontrol server (and possibly a substitute telecontrol server)
  The telecontrol server can be a separate PC or installed on the engineering station in the form of the "TELECONTROL SERVER BASIC" application.
- A TeleService server
  A TeleService server is used when no telecontrol server is available; in other words, when the CP operates in "GPRS direct" mode.

The telecontrol server or TeleService server can be connected to the engineering station via LAN or Internet and the TeleService function can be called from there.

Requirements for TeleService via GPRS

- A telecontrol server or a TeleService server
- The STEP 7 project with the required data for GPRS communication

Note on project engineering

The telecontrol server and TeleService server are not configured in STEP 7.

See also

Establishing a connection using GPRS communication (Page 1166)

Modes of the GPRS-CP (Page 1165)

Establishing a TeleService connection via GPRS

Establishing a connection for TeleService via GPRS

The request to establish a connection is triggered by the engineering station and transmitted by a wake-up SMS to the station. The CP 1242-7 in the S7-1200 station establishes a connection to the engineering station via the GSM network and the Internet.

Starting the TeleService via GPRS

Start the TeleService via GPRS as follows:

1. In the project on the authorized engineering station, select the remote S7 station with which you want to establish a TeleService connection via GPRS.
2. Alternatively, open the "Connect online" dialog box as follows:
   - "Connect online" button
   - "Connect online" shortcut menu (right mouse button)
   - "Online" > "Connect online" menu

   The "Connect online" dialog box opens.
3. Choose interface type "TeleService GPRS" in the "Type of PG/PC interface" drop-down list.

4. In the "PG/PC interface" drop-down list, select the "GPRS TeleService board" option if it is not automatically displayed.
5. Click on the "Connect" icon next to the "PG/PC interface" drop-down list.
   The "Establish remote connection" dialog box opens.
6. Make the necessary settings in the "Establish remote connection" dialog.
   Details of this are contained in the tooltip cascade of STEP 7.

The following details are required to successfully establish a connection:

Details required to establish a connection with the GPRS station

The following information is necessary in the Establish remote connection" dialog:

- IP address or DNS name of the telecontrol server
- TCP port number of the telecontrol server or the DSL router via which the connection between the engineering station and the remote GPRS station is running.
- Server password of the ES to authenticate the engineering station at the telecontrol server
  Only required if a group-specific password has been configured in the "TELECONTROL SERVER BASIC" application.
- TeleService user name
  See CP configuration in STEP 7.
- TeleService password
  See CP configuration in STEP 7.
- Access ID of the CP
  Only required when there are several CP 1242-7s in the station. See CP configuration in STEP 7.

Status

Connection statuses of TeleService via GPRS

The following connection statuses can be displayed in the "Establish remote connection" dialog box:

- Not connected
  There is no connection to the remote GPRS station. Connection establishment has not yet started.

If connection establishment was started by clicking the "Connect" button, the following statuses are displayed one after the other after the connection has been established:

- Connect to telecontrol server
  The engineering station connects to the telecontrol server.
- Wait for S7 station
  The wake-up SMS has been sent to the remote station. Wait for reply from station.

- Authentication at S7 station
  The S7 station has established an IP connection with the engineering station via GPRS and Internet and is checking the received logon and authentication data.
- Connected
  The station has successfully established the connection to the engineering station.

The following states may be displayed if the connection cannot be successfully established:

- Telecontrol server not accessible
  Possible causes:
  – The connection between the engineering station and the telecontrol server has been interrupted.
  – The telecontrol server is switched off.
- Wrong server password
  Cause: The wrong server password for logging on and authentication was entered at the telecontrol server.
- S7 station not responding
  Possible causes:
  – Faulty GSM communication between the telecontrol server and the station.
  – Faulty connection between the GSM network and the Internet.
  – Faulty Internet connection.
  – The telecontrol server could not send a wake-up SMS.
  – The CP has not received a wake-up SMS.
  – The sender of the SMS was not configured in the list of authorized wake-up call numbers.
- Wrong TeleService user name or TeleService password
  Possible causes:
  – An incorrect TeleService user name or incorrect TeleService password was entered in the authentication dialog for the CP 1242-7.
  – The TeleService user name or TeleService password was not configured in STEP 7.
- All TeleService access points are in use.
- The CP is not known to the telecontrol server.
  Cause: The CP originates from a STEP 7 project that does not match the project of the telecontrol server.
- No resources available for TeleService on the CP: Please contact the hotline.
- Protocol error
  Cause: Wrong frame or frame from wrong subscriber. Please contact the hotline.

CP 1243-1

CPU scan cycle

Structure of the CPU scan cycle

The cycle (including the pause) with which the CP scans the memory area of the CPU is made up of the following phases:

- High-priority read jobs
  For data points of the type "Input", which are configured with the "High priority" setting in the data point configuration in "General > Priority in the scan cycle", the PLC tags are all read in one scan cycle.
- Write jobs
  In every cycle, the values of a certain number of unsolicited write jobs are written to the CPU. The number of tags written per cycle is specified for the CP in the "Communication with the CPU" parameter group with the "Max. number of write jobs" parameter. The tags whose number exceeds this value are then written in the next or one of the following cycles.
- Low-priority read jobs - proportion
  For data points of the type "Input", which are configured with the "Low priority" setting in the data point configuration in "General > Priority in the scan cycle", the values of a part of the PLC tags are read in every scan cycle.
  The number of tags read per cycle is specified for the CP in the "Communication with the CPU" parameter group with the "Max. number of read jobs" parameter. The tags that exceed this value and can therefore not be read in one cycle are then read in the next or one of the following cycles.
- Cycle pause time
  This is the waiting time between two scan cycles. It is used to reserve adequate time for other processes that access the CPU via the backplane bus of the station.

Duration of the CPU scan cycle

Since no fixed time can be configured for the cycle and since the individual phases cannot be assigned a fixed number of objects, the duration of the scan cycle is variable and can change dynamically.

Datapoint types

During the configuration of the user data to be transferred by the CP, each data point is assigned a protocol-specific data point type. The data point types supported by the CP along with the compatible S7 data types are listed below. They are grouped according to format (memory requirements).

The direction relates to the direction of transfer (monitoring direction = "in", control direction = "out").

CP 1243-1 DNP3: Supported data point types

Table 8-88 Supported data point types, DNP3 object groups, variants and compatible S7 data types

Format (memory requirements)	Data point type	DNP3 object group [variations]	Direction	S7 data types	Address area
Bit	Binary Input	1 [1, 2]	in	BOOL	I, Q, M
	Binary Input Event	2 [1, 2]	in	BOOL	I, Q, M
	Binary Output [1]	10 [2]	out		
	Binary Output Event [1]	11 [1, 2]	out		
	Binary Command	12 [1]	out	BOOL	I, Q, M
Integer (16 bits)	Counter Static	20 [2]	in	UINT, WORD	I, Q, M
	Frozen Counter [2]	21 [2, 6]	in		
	Counter Event	22 [2, 6]	in	UINT, WORD	I, Q, M
	Frozen Counter Event [3]	23 [2, 6]	in		
	Analog Input	30 [1]	in	INT	I, Q, M
	Analog Input Event	32 [2, 4]	in	INT	I, Q, M
	Analog Output Status [4]	40 [2]	out		
	Analog Output	41 [2]	out	INT	I, Q, M
	Analog Output Event [4]	42 [2, 4]	out		
Integer (32 bits)	Counter Static	20 [1]	in	UDINT, DWORD	I, Q, M
	Frozen Counter [2]	21 [1, 5]	in		
	Counter Event	22 [1, 5]	in	UDINT, DWORD	I, Q, M
	Frozen Counter Event [3]	23 [1, 5]	in		
	Analog Input	30 [2]	in	DINT	I, Q, M
	Analog Input Event	32 [1, 3]	in	DINT	I, Q, M
	Analog Output Status [4]	40 [1, 3]	out		
	Analog Output	41 [1]	out	DINT	I, Q, M
	Analog Output Event [4]	42 [1]	out		
Floating-point number (32 bits)	Analog Input	30 [5]	in	REAL	Q, M
	Analog Input Event	32 [5, 7]	in	REAL	Q, M
	Analog Output Status [4]	40 [3]	out		
	Analog Output	41 [3]	out	REAL	Q, M
	Analog Output Event [4]	42 [5, 7]	out		
Floating-point number (64 bits)	Analog Input	30 [6]	in	LREAL	Q, M
	Analog Input Event	32 [6, 8]	in	LREAL	Q, M
	Analog Output	41 [4]	out	LREAL	Q, M
	Analog Output Event [4]	42 [6, 8]	out		
Block of data (1...64 bytes) [5]	Octet String	110 [-]	in	[5]	I, Q, M
	Octet String Event [5]	111 [-]	in	[5]	I, Q, M

[1] This object group can be configured in the Data point editor of STEP 7 using object group 12.

[2] This object group can be configured in the Data point editor of STEP 7 using object group 20.

[3] This object group can be configured in the Data point editor of STEP 7 using object group 22.

[4] This object group can be configured in the Data point editor of STEP 7 using object group 41.

[5] With these data point types, contiguous memory areas up to a size of 64 bytes can be transferred. All S7 data types with a size between 1 and 64 bytes are compatible. If the array is modified later, the data point must be recreated.

Configuration and feedback of binary and analog values

- **Binary values**
  You configure binary values (inputs) using object groups 1 and 2.
  The object groups 10 and 11 are used to feed back the current values in the process image of the CPU to the control system.
- **Analog values**
  You configure analog values using the object groups 30, 32 and 41 depending on the transfer direction.
  The object groups 40 and 42 are used to feed back current values in the process image of the CPU to the control system.

CP 1243-1 IEC: Supported data point types

Table 8-89 Supported data point types, IEC types and compatible S7 data types

Format (memory requirements)	Data point type	IEC type	Direction	S7 data types	Address area
Bit	Single point information	<1>	in	BOOL	I, Q, M
	Single point information with time tag	<30>	in	BOOL	I, Q, M
	Single command	<45>	out	BOOL	I, Q, M
Byte	Step position information	<5>	in	BYTE	I, Q, M
	Step position information with time tag	<32>	in	BYTE	I, Q, M
	Regulating step command	<47>	out	BYTE	I, Q, M
Integer (16 bits)	Measured value, normalized value	<9>	in	INT	I, Q, M
	Measured value, normalized value with time tag	<34>	in	INT	I, Q, M
	Measured value, scaled value	<11>	in	INT	I, Q, M
	Measured value, scaled value with time tag	<35>	in	INT	I, Q, M
	Set point command, normalised value	<48>	out	INT	I, Q, M
	Set point command, scaled value	<49>	out	INT	I, Q, M
Integer (32 bits)	Bitstring of 32 bits	<7>	in	DWORD, UDINT	I, Q, M
	Bitstring of 32 bits with time tag	<33>	in	DWORD, UDINT	I, Q, M
	Integrated totals	<15>	in	DWORD, UDINT	I, Q, M
	Integrated totals with time tag	<37>	in	DWORD, UDINT	I, Q, M
Floating-point number (32 bits)	Measured value, short floating point number	<13>	in	REAL	Q, M
	Measured value, short floating point number with time tag	<36>	in	REAL	Q, M

Format (memory requirements)	Data point type	IEC type	Direction	S7 data types	Address area
	Set point command, short floating point number	<50>	out	REAL	Q, M
Block of data (1...32 Bit) [1]	Bitstring of 32 bits [1]	<7>	in	[1]	I, Q, M
	Bitstring of 32 bits with time tag [1]	<33>	in	[1]	I, Q, M
	Bitstring of 32 bits [1]	<51>	out	[1]	I, Q, M

[1] With these data point types, contiguous memory areas up to a size of 32 bits can be transferred. All S7 data types with a size between 1 and 32 bits are compatible. If the array is modified later, the data point must be recreated.

Types of transmission and event classes

Storing and transferring values

The values of data points can be assigned to different memory areas on the CP and therefore to different types of transfer:

- Static values
  Static values are entered in the process image of the CP.
  The value of a static data point is sent after being requested by the communications partner.
- Events
  The values of data points configured as an event are also entered in the process image of the CP.
  The values of events are also entered in the send buffer of the CP.
  The value of a data point that was configured as an event is sent unsolicited to the communications partner if this function is enabled by the master.
  The send buffer has the following size:
  - With DNP3: Max. 64000 events
  - With IEC: Max. 65535 events

  The capacity of the send buffer is divided up equally for all enabled partners.

Types of transmission

Depending on your CP type, you have the following transmission types available:

- Transfer after call
  The current value of the data point is entered in the image memory of the CP. A new value overwrites the older value in the image memory.
  After being called by the communications partner, the current value at the time is transferred.
- Every value triggered
  Whenever the trigger condition is met, the individual value of the data point is entered separately in the send buffer and transferred to the communications partner.

- Event class ...
  The value is transferred to the send buffer triggered as a class 1, 2 or 3 event. If unsolicited transfer is enabled by the master, the value is transferred unsolicited.
  You will find details of the event classes in the following sections.
- Current value triggered
  Only the current value of a data point at the time of the trigger is entered in the send buffer and transferred to the communications partner.

Event classes for the DNP3 protocol

The process data of the various event classes is handled as follows:

- Event class 1 / event class 2
  Classes according to the DNP3 protocol: Event, class 1 / class 2
  Each value change is entered in the send buffer in chronological order.
  The evaluation of the classification (1 or 2) must be handled by the master.
- Event class 3
  Class according to the DNP3 protocol: Static event, class 3
  Only the current value at the time the trigger condition was met is entered in the send buffer and overwrites the last value stored there.

Event classes for the IEC protocol

The process data of the various event classes is handled as follows:

- Event class 1
  Classes according to the IEC protocol: Event, class 1
  Each value change is entered in the send buffer in chronological order.
- Event class 3
  Class according to the IEC protocol: Static event, class 3
  Only the current value at the time the trigger condition was met is entered in the send buffer and overwrites the last value stored there.

Analog value preprocessing

Some CPs have analog value preprocessing that, among other things, supports the functions described below.

Smoothing factor

Analog values that fluctuate quickly can be evened out using the smoothing function.

The smoothing factors are calculated according to the following formula as with S7 analog input modules.

$$y_n = \frac{x_n + (k-1)y_{n-1}}{k}$$

where
y_n = smoothed value in the current cycle

x_n = value acquired in the current cycle n
k = smoothing factor

The following values can be configured for the module as the smoothing factor.

- 1 = No smoothing
- 4 = Weak smoothing
- 32 = Medium smoothing
- 64 = Strong smoothing

Fault suppression time

An analog value in the overflow range (32767 / $7FFF_h$) or underflow range (-32768 / 8000_h) is not transferred for the duration of the fault suppression time. This also applies to live zero inputs. The value in the overflow/underflow range is only sent after the fault suppression time has elapsed, if it is still pending.

If the value returns to the measuring range before the fault suppression time elapses, the current value is transferred immediately.

A typical use case for this parameter is the suppression of peak current values when starting up powerful motors that would otherwise be signaled to the control center as a disruption.

The suppression is adjusted to analog values that are acquired by the S7 analog input modules as raw values. These modules return the specified values for the overflow or underflow range for all input ranges (also for live zero inputs).

If the CPU makes preprocessed finished values available in the bit memory area or in a data block, fault suppression is only possible or useful if these values have the values listed above (32767 / $7FFF_h$) or (-32768 / 8000_h) in the overflow or underflow range. If this is not the case, the parameter should not be enabled for preprocessed values.

Mean value generation

With this parameter, acquired analog values are transferred as mean values.

The current values of an analog datapoint are acquired cyclically and totaled. The number of acquired values per time unit depends on the read cycle of the CPU and the CPU scan cycle of the CP. The mean value is calculated from the accumulated values as soon as the transfer is triggered by a time trigger. Following this, the accumulation starts again so that the next mean value can be calculated.

The mean value can also be calculated if the transmission of the analog value message is triggered by a request from the communications partner. The duration of the mean value calculation period is then the time from the last transmission (for example triggered by the trigger) to the time of the request. Once again, the accumulation restarts so that the next mean value can be calculated.

If the acquired analog value in the overflow range (32767 / $7FFF_h$) or underflow range (-32768 / 8000_h), this value can either be taken into account immediately in the calculation of the mean value or it can be suppressed for a specific period for the calculation of the mean value. The required response can be decided with the Fault suppression time parameter:

- Fault suppression time = 0
  Acquisition of a value in the overflow or underflow range results in the mean calculation being stopped immediately. The value 32767 / $7FFF_h$ or -32768 / 8000_h is saved as an invalid mean value for the current mean value calculation period and sent when the next analog value message is triggered. The calculation of a new mean value is then started. If the analog value remains in the overflow or underflow range, this new value is again saved immediately as an invalid mean value and sent when the next frame is triggered.
- Fault suppression time > 0
  If the acquired analog value is in the overflow or underflow range, the bad values are excluded from the calculation of the mean value for a maximum time as defined by the Fault suppression time. If this time is exceeded, the value 32767 / $7FFF_h$ or -32768 / 8000_h is saved as an invalid mean value and sent when the next analog value frame is triggered. The procedure is identical in each new averaging period; in other words, bad values are again suppressed for the duration of the fault suppression time.

The duration of the fault suppression time also indirectly decides the proportion of invalid values per averaging period. If the mean value is formed, for example in a cycle of 15 minutes, and the fault suppression time is set to 5 minutes, this means that the mean value is only transferred as invalid if more than 33% of the acquired analog values are in the overflow or underflow range in the current averaging period.

8.1.4.9 SCALANCE X, W and M

SCALANCE X

Useful information

VLAN

Network definition regardless of the spatial location of the nodes

VLAN (Virtual Local Area Network) divides a physical network into several logical networks that are shielded from each other. Here, devices are grouped together to form logical groups. Only nodes of the same VLAN can address each other. Since multicast and broadcast frames are only forwarded within the particular VLAN, they are also known as broadcast domains.

The particular advantage of VLANs is the reduced network load for the nodes and network segments of other VLANs.

To identify which packet belongs to which VLAN, the frame is expanded by 4 bytes (VLAN tagging (Page 1181)). This expansion includes not only the VLAN ID but also priority information.

Options for the VLAN assignment

There are various options for the assignment to VLANs:

- Port-based VLAN
  Each port of a device is assigned a VLAN ID. You configure port-based VLAN in "Layer 2 > VLAN > Port-based VLAN".
- Protocol-based VLAN
  Each port of a device is assigned a protocol group. You can configure protocol-based VLAN in "Layer 2 > VLAN > Protocol Based VLAN Port"
- Subnet-based VLAN
  The IP address of the device is assigned a VLAN ID. You configure subnet-based VLAN in "Layer 2 > VLAN > Ipv4 Subnet Based VLAN".

processing the VLAN assignment

If more than one VLAN assignment is created on the device, the assignments are processed in the following order:

1. Subnet-based VLAN
2. Protocol-based VLAN
3. Port-based VLAN

The frame is first examined for the IP address. If a rule on the "Ipv4 Subnet Based VLAN" tab applies, the frame is sent to the corresponding VLAN. If no rule applies, the protocol type of the frame is examined. If a rule on the "Protocol Based VLAN Port" tab applies, the frame is sent to the corresponding VLAN. If no rule applies, the frame is sent via the port-based VLAN. The rules for the port-based VLAN are specified on the "Port Based VLAN" tab.

See also

General (Page 1265)

GVRP (Page 1267)

Port-based VLAN (Page 1268)

Protocol-based VLAN group (Page 1269)

Protocol-based VLAN port (Page 1270)

IPv4 subnet-based VLAN (Page 1270)

VLAN tagging

Expansion of the Ethernet frames by four bytes

For CoS (Class of Service, frame priority) and VLAN (virtual network), the IEEE 802.1 Q standard defined the expansion of Ethernet frames by adding the VLAN tag.

Note

The VLAN tag increases the permitted total length of the frame from 1518 to 1522 bytes. With the IE switches, the standard MTU size is 1536 bytes. The MTU size can be changed to values from 64 to 9216 bytes.
The end devices on the networks must be checked to find out whether they can process this length / this frame type. If this is not the case, only frames of the standard length may be sent to these nodes.

The additional 4 bytes are located in the header of the Ethernet frame between the source address and the Ethernet type / length field:

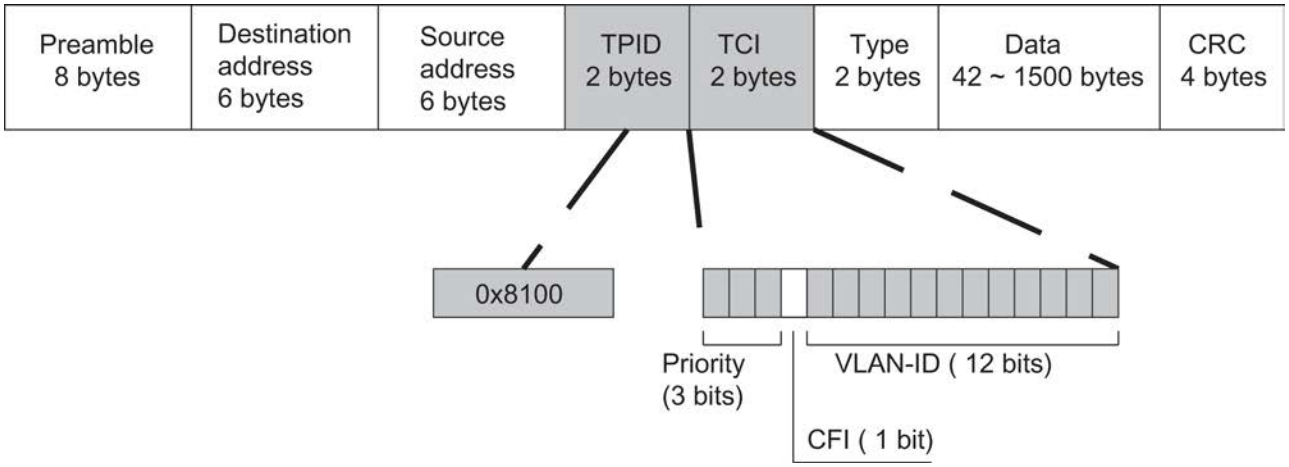


Figure 8-48 Structure of the expanded Ethernet frame

The additional bytes contain the tag protocol identifier (TPID) and the tag control information (TCI).

Tag protocol identifier (TPID)

The first 2 bytes form the Tag Protocol Identifier (TPID) and always have the value 0x8100. This value specifies that the data packet contains VLAN information or priority information.

Tag Control Information (TCI)

The 2 bytes of the Tag Control Information (TCI) contain the following information:

CoS prioritization

The tagged frame has 3 bits for the priority that is also known as Class of Service (CoS). The priority according to IEEE 802.1p is as follows:

CoS bits	Type of data
000	Non time-critical data traffic (less then best effort [basic setting])
001	Normal data traffic (best effort [background])
010	Reserved (standard)
011	Reserved (excellent effort)
100	Data transfer with max. 100 ms delay
101	Guaranteed service, interactive multimedia
110	Guaranteed service, interactive voice transmission
111	Reserved

The prioritization of the data packets is possible only if there is a queue in the components in which they can buffer data packets with lower priority.

The device has eight parallel queues in which the frames with different priorities can be processed. First, the frames with the highest priority ("Strict Priority" method) are processed. This method ensures that the frames with the highest priority are sent even if there is heavy data traffic.

Canonical Format Identifier (CFI)

The CFI is responsible for compatibility between Ethernet and the token Ring.
The values have the following meaning:

Value	Meaning
0	The format of the MAC address is canonical. In the canonical representation of the MAC address, the least significant bit is transferred first. Standard-setting for Ethernet switches.
1	The format of the MAC address is not canonical.

VLAN ID

In the 12-bit data field, up to 4095 VLAN IDs can be formed. The following conventions apply:

VLAN ID	Meaning
0	The frame contains only priority information (priority tagged frames) and no valid VLAN identifier.
1 - 4094	Valid VLAN identifier, the frame is assigned to a VLAN and can also include priority information.
4095	Reserved

SNMP

Introduction

With the aid of the Simple Network Management Protocol (SNMP), you monitor and control network elements from a central station, for example routers, switches. SNMP controls the communication between the monitored devices and the monitoring station.

Tasks of SNMP:

- Monitoring of network components
- Remote control and remote parameter assignment of network components
- Error detection and error notification

In versions v1 and v2c, SNMP has no security mechanisms. Each user in the network can access data and also change parameter assignments using suitable software.

For the simple control of access rights without security aspects, community strings are used.

The community string is transferred along with the query. If the community string is correct, the SNMP agent responds and sends the requested data. If the community string is not correct, the SNMP agent discards the query. Define different community strings for read and write permissions. The community strings are transferred in plain text.

Standard values of the community strings:

- public
  has only read permissions
- private
  has read and write permissions

Note

For security reasons, do not use the standard values "public" or "private". Change the values following the initial installation.

Further simple protection mechanisms at the device level:

- Allowed Host
  The IP addresses of the monitoring systems are known to the monitored system.
- Read Only
  If you assign "Read Only" to a monitored device, monitoring stations can only read out data but cannot modify it.

SNMP data packets are not encrypted and can easily be read by others.

The central station is also known as the management station. An SNMP agent is installed on the devices to be monitored with which the management station exchanges data.

The management station sends data packets of the following type:

- GET
  Request for a data record from the agent
- GETNEXT
  Calls up the next data record.
- GETBULK (available as of SNMPv2)
  Requests multiple data records at one time, for example several rows of a table.
- SET
  Contains parameter assignment data for the relevant device.

The SNMP agent sends data packets of the following type:

- RESPONSE
  The agent returns the data requested by the manager.
- TRAP
  If a certain event occurs, the SNMP agent sends traps.

SNMPv1 and SNMPv2 and SNMPv3 use UDP (User Datagram Protocol). The data is described in a Management Information Base (MIB).

SNMP v3

Compared with the previous versions SNMP v1 and SNMP v2, SNMP v3 introduces a comprehensive security concept.

SNMP v3 supports:

- Fully encrypted user authentication
- Encryption of the entire data traffic
- Access control of the MIB objects at the user/group level

Spanning Tree

Avoiding loops on redundant connections

The spanning tree algorithm allows network structures to be created in which there are several connections between two stations. Spanning tree prevents loops being formed in the network by allowing only one path and disabling the other (redundant) ports for data traffic. If there is an interruption, the data can be sent over an alternative path. The functionality of the spanning tree algorithm is based on the exchange of configuration and topology change frames.

Definition of the network topology using the configuration frames

The devices exchange configuration frames known as BPDUs (Bridge Protocol Data Unit) with each other to calculate the topology. The root bridge is selected and the network topology created using these frames. BPDUs also bring about the status change of the root ports.

The root bridge is the bridge that controls the spanning tree algorithm for all involved components.

Once the root bridge has been specified, each device sets a root port. The root port is the port with the lowest path costs to the root bridge.

Response to changes in the network topology

If nodes are added to a network or drop out of the network, this can affect the optimum path selection for data packets. To be able to respond to such changes, the root bridge sends configuration messages at regular intervals. The interval between two configuration messages can be set with the "Hello Time" parameter.

Keeping configuration information up to date

With the "Max Age" parameter, you set the maximum age of configuration information. If a bridge has information that is older than the time set in Max Age, it discards the message and initiates recalculation of the paths.

New configuration data is not used immediately by a bridge but only after the period specified in the "Forward Delay" parameter. This ensures that operation is only started with the new topology after all the bridges have the required information.

RSTP, MSTP, CIST

Rapid Spanning Tree Protocol (RSTP)

One disadvantage of STP is that if there is a disruption or a device fails, the network needs to reconfigure itself: The devices start to negotiate new paths only when the interruption occurs. This can take up to 30 seconds. Fur this reason, STP was expanded to create the "Rapid Spanning Tree Protocol" (RSTP, IEEE 802.1w). This differs from STP essentially in that the devices are already collecting information about alternative routes during normal operation and do not need to gather this information after a disruption has occurred. This means that the reconfiguration time for an RSTP controlled network can be reduced to a few seconds.
This is achieved by using the following functions:

- Edge ports (end node port)
  Edge ports are ports connected to an end device.
  A port that is defined as an edge port is activated immediately after connection establishment. If a spanning tree BPDU is received at an edge port, the port loses its role as edge port and it takes part in (R)STP again. If no further BPDU is received after a certain time has elapsed (3 x hello time), the port returns to the edge port status.
- Point-to-point (direct communication between two neighboring devices)

  By directly linking the devices, a status change (reconfiguration of the ports) can be made without any delays.
- Alternate port (substitute for the root port)

  A substitute for the root port is configured. If the connection to the root bridge is lost, the device can establish a connection over the alternate port without any delay due to reconfiguration.
- Reaction to events

  Rapid spanning tree reacts to events, for example an aborted connection, without delay. There is no waiting for timers as in spanning tree.
- Counter for the maximum bridge hops
  The number of bridge hops a package is allowed to make before it automatically becomes invalid.
  In principle, therefore with rapid spanning tree, alternatives for many parameters are preconfigured and certain properties of the network structure taken into account to reduce the reconfiguration time.

Multiple Spanning Tree Protocol (MSTP)

The Multiple Spanning Tree Protocol (MSTP) is a further development of the Rapid Spanning Tree Protocol. Among other things, it provides the option of operating several RSTP instances within different VLANs or VLAN groups and, for example, making paths available within the individual VLANs that the single Rapid Spanning Tree Protocol would globally block.

Common and internal Spanning Tree (CIST)

CIST is a term from the multiple spanning tree. CIST identifies the internal instance used by the switch that is comparable in principle with an internal RSTP instance.

Routing function

Introduction

The term routing describes the specification of routes for communication between different networks; in other words, how does a data packet from subnet A get to subnet B.

SCALANCE X supports the following routing functions:

- Static routing
  With static routing, the routes are entered manually in the routing table.
- Router redundancy
  With standardized VRRP (Virtual Router Redundancy Protocol), the availability of important gateways is increased by redundant routers.
- Dynamic routing
  The entries in the routing table are dynamic and are updated continuously. The entries are created by the routing protocol.
  - OSPF v2 (Open Shortest Path First)

Static routing

The route is entered manually in the routing table. Enter the route in the routing table on the WBM page "Routes".

Router redundancy with VRRP

With the Virtual Router Redundancy Protocol (VRRP), the failure of a router in a network can be countered.

Several VRRP routers in a network segment are put together as a logical group that forms a virtual router (VR). The group is defined using the virtual ID (VRID). Within the group, the VRID must be the same. The VRID can no longer be used for other groups.

The virtual router is assigned a virtual IP address and a virtual MAC address. One of the VRRP routers within the group is specified as the master router. The master router has priority 255. The other VRRP routers are backup routers. The master router assigns the virtual IP address and the virtual MAC address to its network interface. The master router sends VRRP packets

(advertisements) to the backup routers at specific intervals. With the VRRP packets, the master router signals that it is still alive. The master router also replies to the ARP queries.

If the virtual master router fails, a backup router takes over the role of the master router. The backup router with the highest priority becomes the master router. If the priority of the backup routers is the same, the higher MAC address decides. The backup router becomes the new virtual master router.

The new virtual master router adopts the virtual MAC and IP address. This means that no routing tables or ARP tables need to be updated. The consequences of a device failure are therefore minimized.

You configure VRRP in "Layer 3 > VRRP".

OSPFv2

Dynamic routing with OSPF v2

OSPF (Open Shortest Path First) is a cost-based routing protocol. To calculate the shortest and most cost-effective route, the Short Path First algorithm by Dijkstra is used. OSPF was developed by the IETF (Internet Engineering Task Force).
You configure OSPFv2 in "Layer 3 > OSPFv2".

OSPF v2 divides an autonomous system (AS) into different areas.

Areas in OSPF

The following areas exist:

- Backbone
  The backbone area is area 0.0.0.0. All other areas are connected to this area. The backbone area is connected either directly or via virtual connections with other areas.
  All routing information is available in the backbone area. As a result, the backbone area is responsible for forwarding information between different areas.
- Stub Area
  This area contains the routes within its area within the autonomous system and the standard route out of the autonomous system. The destinations outside this autonomous system are assigned to the standard route.
- Totally Stubby Area
  This area knows only the routes within its area and the standard route out of the area.
- Not So Stubby Area (NSSA)
  This area can forward (redistribute) packets from other autonomous systems into the areas of its own autonomous system. The packets are further distributed by the NSSA router.

Routers of OSPF

OSPF distinguishes the following router types:

- Internal router (IR)
  All OSPF interfaces of the router are assigned to the same area.
- Area Border Router (ABR)
  The OSPF interfaces of the router are assigned to different areas. One OSPF interface is assigned to the backbone area. Where possible, routes are grouped together.
- Backbone Router (BR)
  At least one of the OSPF interfaces is assigned to the backbone area.
- Autonomous System Area Border Router (ASBR)
  One interface of the router is connected to a different AS, for example an AS that uses the routing protocol RIP.

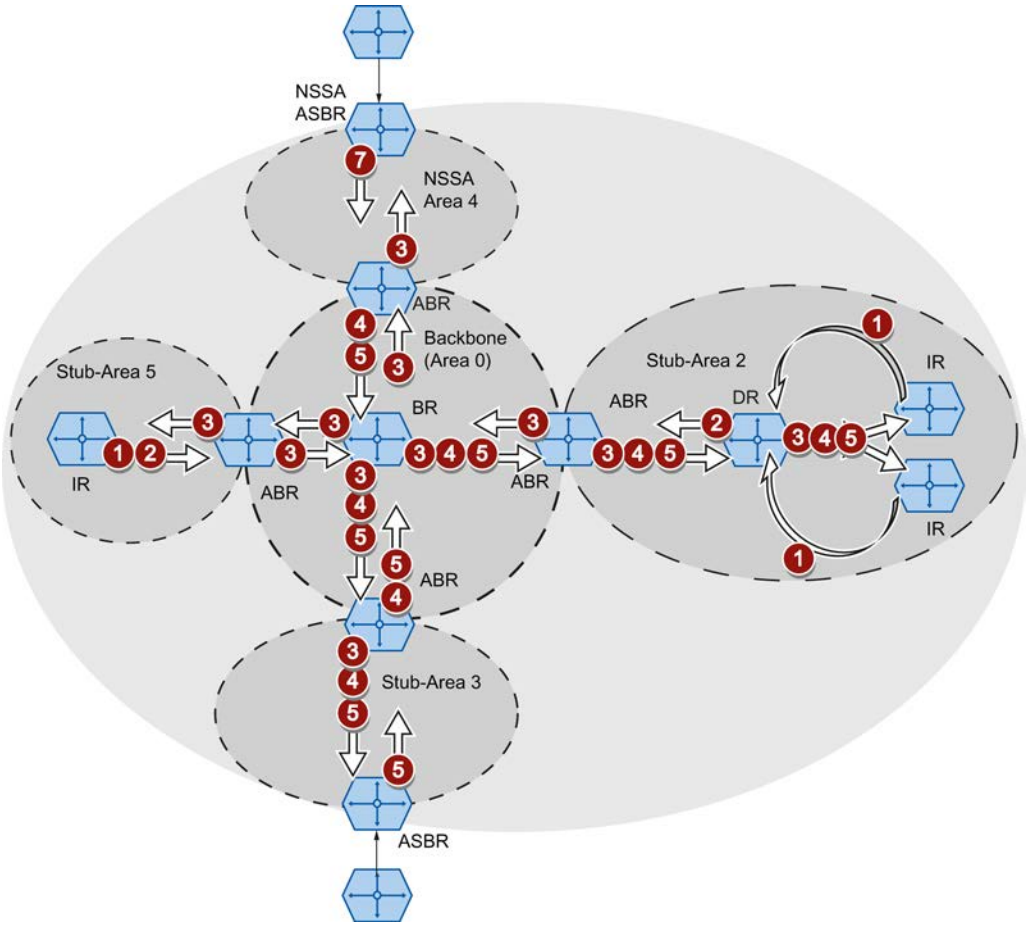
Virtual connection

Each area must be physically connected to the backbone area. In some situations a physical connection is not possible. The remote router is connected to a backbone router via a virtual connection. In the area through which the virtual connection runs, there is no designated router or designated backup router.

LSA types

Within the autonomous system, packets are exchanged that contain information about the connections of a router and the connection status message. The packets are also known as LSAs (Link State Advertisements). The LSAs are always sent from the router to the neighbor router.

If there are changes in the network, LSAs are sent to all routers in the network. The information depends on the LSA type.



1. **Router LSA (LSA Type 1)**
   The LSA Type 1 is only sent within an area. For each active connection of the router that belongs to the area in consideration, an LSA Type 1 is generated. The LSA Type 1 contains information about the status and the costs of the connection, for example IP address, network mask, network type
2. **Network LSA (LSA Type 2)**
   The LSA Type 2 is sent only within an area. For each network that belongs to the relevant area, the router generates an LSA Type 2. If several routers are interconnected in a network, the LSA Type 2 is sent by the designated router (DR). The LSA Type 2 includes the network address, the network mask and a list of routers that are connected to the network

3 4 **Summary LSA (LSA Type 3 / LSA Type 4)**
The Summary LSA is generated by the area border router and sent into the area. The Summary LSA contains information about routes outside the area but inside the AS. Where possible, the routes are grouped together.

- Summary LSA (LSA Type 3)
  The LSA Type 3 describes the routes to the networks and advertises the standard route to the areas.
- AS Summary LSA (LSA Type 4)
  The LSA Type 4 describes the routes to the ASBR.

5 **External LSA (LSA Type 5 / LSA Type 7)**
The External LSA is generated by the ASBR. The LSA type depends on the area.

7
- AS External LSA (LSA Type 5)
  The LSA Type 5 is sent by the AS border router into the areas of the autonomous system except the Stub and NSSA areas. The LSA contains information about routes to a network in another AS. The routes are either created manually or learned externally. The ASBR uses LSA Type 5 to distribute standard routes to the backbone area.
- NSSA External LSA (LSA Type 7)
  The LSA Type 7 is generated by the AS border router of an NSSA. The router is also known as the NSSA ASBR. The LSA Type 7 is sent only within the NSSA. If the P bit in LSA Type 7 = 1, these LSAs are converted to LSA Type 5 by the ABR and sent to the backbone area.

Establishing the neighborhood

The router runs through the following statuses to establish a connection to the neighbor router.

1. Attempt state / Init state
   The router activates OSPF and begins to send and receive Hello packets. Based on the received Hello packets, the router learns which OSPF routers are in its vicinity. The router checks the content of the Hello packet. The Hello packet also contains the list of the neighbor routers (neighbor table) of the "sender".
2. Two way state
   If, for example, the ID of the area, the area type and the settings for the times match, a connection (adjacency) can be established to the neighbor. In a point-to-point network, the connection is established directly. If several neighbor routers can be reached in a network, the designated router (DR) and the designated backup router (DBR) are identified based on Hello packets. The router with the highest router priority becomes the designated router. If two routers have the same router priority, the router with the lower router ID becomes the designated router. The router establishes a connection to the designated router.
3. Exchangestart state
   The neighbor routers decide which router starts communication. The router with the higher router ID becomes master.
4. Exchange state
   The neighbor routers send packets that describe the content of their neighborhood database. The neighborhood database (link state database - LSDB) contains information on the topology of the network.

5. Loading state
   The router completes the received information. If the router still has questions relating to the status of a specific connection, it sends a link state request. The neighbor router sends a response (link state update). The response contains a suitable LSA. The router confirms receipt of the response (link state acknowledge).
6. Full State
   The information exchange with the neighbor router is completed. The neighborhood database of the neighbor router is the same. Based on the Short Path First algorithm, the router calculates a route to every destination. The route is entered in the routing table.

Check the neighborhood

The Hello packets are only used to establish the neighborhood relations. Hello packets are used to check the connection to the neighbor router by sending them cyclically. If no Hello packet is received within a certain interval (dead interval), the connection to the neighbor is marked as "down". The relevant entries are deleted.

Updating the neighborhood database

Once the neighborhood database is established, LSAs are sent to all routers in the network if there are changes in the topology.

RIPv2

Dynamic routing with RIPv2

The Routing Information Protocol (RIPv2) is used to create routing tables automatically. RIPv2 is used in autonomous systems (AS) with a maximum of 15 routers. It is based on the Distance-Vector algorithm.

RIPv2 was developed by the IETF (Internet Engineering Task Force) and is described in RFC 2453.
You configure RIPv2 in "Layer 3 > RIPv2".

Setting up a routing table

Since a router initially only knows its directly connected networks, it sends a request to its direct neighbor routers. As the reply, it receives the routing tables of the neighbor routers. Based on the information it receives, the router set up its own routing table.

The routing table contains entries for all possible destinations. Each entry includes the distance to the destination and the first router on the route.

The distance is also known as the metric. This indicates the number of routers to be passed through on the route to the destination (hop count). The maximum distance is 15 routers (hops).

Updating the routing table

Once the routing table is set up, the router sends its routing table to each direct neighbor router at intervals of 30 seconds.

The router compares new routing information with its existing routing table. If the new information includes shorter routes, the existing routes are overwritten. The router only keeps the shortest route to a destination.

Checking neighbor routers

If a router does not receive messages from a neighbor router for longer than 180 seconds, it marks the router as being invalid. The router assigns the metric 16 for the neighbor router.

Link aggregation

Link aggregation

With link aggregation, several parallel physical connections with the same transmission speed are grouped together to form a logical connection with a higher transmission speed. This method based on IEEE 802.3ad is also known as port trunking or channel bundling.

Link aggregation works only with full duplex connections with the same transmission speed in point-to-point mode. This achieves multiplication of the bandwidth or transmission speed. If part of the connection fails, the data traffic is handled via the remaining parts of the connection.

To control and monitor, the Link Aggregation Control Layer (LACL) and the Link Aggregation Control Protocol (LACP) are used.

Authentication

Authentication method

You can configure the authentication methods "802,1x" and "MAC Authentication" as well as the "Guest VLAN" option separately for each port.

The functions have a hierarchical order. If all three functions are enabled, an attempt is made initially to authenticate the end device using "802.1x". If this authentication is unsuccessful, "MAC Authentication" is started. If this authentication is also unsuccessful, the end device is enabled for communication in the "Guest VLAN". ""Guest VLAN" can only be used if at least one authentication method is active.

The two authentication methods depend on the end device. If the end device supports EAP (Extensible Authentication Protocol), it can be authenticated using the "802.1x" method. If the end device does not support EAP, it can be authenticated using "MAC Authentication". In this case, the IE switch adopts the role of the end device and uses the MAC address of the device as the authentication parameter.

802.1x

The "802.1x" authentication method works as follows:
An end device that supports EAP sends authentication information to the IE switch. The IE switch forwards the information to the authentication server. The authentication server checks the information and allows or denies the end device access to the network.

MAC Authentication

The "MAC Authentication" authentication method works as follows:
As soon as the IE switch receives a frame from the end device, the IE switch sends a query to the RADIUS server to allow or deny the end device access to the network.

Editing properties and parameters

Editing options

You have the following options for editing properties and parameters:

- Hardware and network editor
  Once you have inserted the network component, you can edit the properties and parameters, for example the device name. You will find more detailed information in "Hardware and network editor".
- Web Based Management (WBM)
  The parameters and properties are accessible using the supplied HTML pages (WBM pages). Each WBM page has its own help page that describes the properties and parameters. For further information, refer to the configuration manual "SCALANCE XM400/X-500 Web Based Management". You will find the MIB at Siemens Industry Automation and Drives Service & Support on the Internet under the entry ID 67428305 (http://support.automation.siemens.com/WW/view/en/67428305).
- Command Line Interface
  All the configuration settings for the device can be made using the CLI. The CLI provides the same options as Web Based Management (WBM). For further information, refer to the configuration manual "SCALANCE XM400/X-500 Command Line Interface". You will find the MIB at Siemens Industry Automation and Drives Service & Support on the Internet under the entry ID 67430663 (http://support.automation.siemens.com/WW/view/en/67430663).

Availability

The availability of the settings depends

- on the port type
- on the configuration mode

SCALANCE X distinguishes the following port types:

- Switch port
- Router port

The following configuration modes are available:

- Offline configuration
  You can configure a PC station the first time offline. In this mode only the settings that require no connection to the device are available.
- Online configuration mode
  If there is a connection to the device, there are additional pages available in the Inspector window. These pages include the note "This page is only available if there is an online connection to the device". With some settings, the additional information "only available online" is included.

Creating and deleting an entry

As an example, an entry will be created and deleted on the Syslog client. The procedure is valid on all pages.

Creating a new entry

1. Select the device in the network or device view.
2. Open the properties of the device in the Inspector window.
3. In the Inspector window go to System > Syslog client.
4. Enter the IP address for the Syslog server.
5. Click in the table.
6. Select the entry "New entry" in the shortcut menu.
   A new entry is created in the table.

Deleting an entry

1. Select the required entry in the table.
2. Select the "Delete" entry in the shortcut menu.

Buttons you require often

- **Refresh the display with "Refresh"**
  Pages that display current parameters have a "Refresh" button at the lower edge of the page. Click this button to request up-to-date information from the device for the current page.
- **Save settings for all ports with "Transfer to table"**
  Pages on which you can configure several ports have 2 tables. In the first table, you can make the settings for all ports and transfer these to the second table. In the last column of the first table, there is the "Transfer to table" button. Click this button to save the settings you have made for all ports.

IP address assignment

Configuration options

When shipped and after restoring to the factory settings, the device does not have an IP address.

The following options are available to assign an IP address to the device:

- DHCP (default setting)
- Primary Setup Tool
  For further information, refer to the configuration manual "Primary Setup Tool". You will find the MIB at Siemens Industry Automation and Drives Service & Support on the Internet under the entry ID 19440762 (http://support.automation.siemens.com/WW/view/en/19440762).
- STEP 7
- CLI via the serial interface
  For further information, refer to the configuration manual "SCALANCE X500 Command Line Interface". You will find the MIB at Siemens Industry Automation and Drives Service & Support on the Internet under the entry ID 67430663 (http://support.automation.siemens.com/WW/view/en/67430663).

Requirement

- "Set IP address on device" is enabled in the properties of the device. You will find more detailed information in "Addressing PROFINET devices".

SCALANCE XR500 as IO device: "Compile" and "Download to device"

With a SCALANCE XR500 configured as a PROFINET IO device and assigned to an IO controller, the functions "Compile" and "Download to device" only download the data to the switch that can also be configured in Web Based Management (WBM) (layer 2, layer 3, system, security).

If you want to use the "Compile" or "Download to device" functions for the PROFINET IO device data of the XR500, first select the assigned IO controller.

Information

Versions

Note

The page is only available if there is an online connection to the device.

This page shows the versions of the hardware and software of the device.

Description

Table 1 has the following columns:

- **Hardware**
  Shows the device being used.
- **Name**
  Shows the name of the device or module.

- Revision
  Shows the hardware version of the device.
- Order ID
  Shows the order number of the device or module.

Table 2 has the following columns:

- Software
  - Firmware
    Shows the current firmware version. If a new firmware file was downloaded and the device has not yet restarted, the firmware version of the downloaded firmware file is displayed here. After the next restart, the downloaded firmware is activated and used.
  - Bootloader
    Shows the version of the boot software stored on the device.
- Description
  Shows the short description of the software.
- Version
  Shows the version number of the software version.
- Date
  Shows the date on which the software version was created.

I&M

Note

The page is only available if there is an online connection to the device.

This page contains information about device-specific vendor and maintenance data such as the order number, serial number, version number etc. You cannot configure anything on this page.

Description of the displayed values

The table has the following rows:

- Manufacturer ID
  Shows the manufacturer ID.
- Order number
  Shows the order number.
- Serial number
  Shows the serial number.
- Hardware revision
  Shows the hardware version.
- Software version
  Shows the software version.

- **Revision counter**
  Shows the revision counter: Counter for revisions since the initial commissioning
- **Revision Date**
  Date and time of the last revision
- **Function tag**
  Shows the function tag (plant designation) of the device. The plant designation (HID) is created during configuration of the device with HW Config of STEP 7.
- **Location tag**
  Shows the location tag of the device. The location identifier (LID) is created during configuration of the device with HW Config of STEP 7.

ARP table

Note

The page is only available if there is an online connection to the device.

Assignment of MAC address and IP address

With the Address Resolution Protocol (ARP), there is a unique assignment of MAC address to IPv4 addresses. This assignment is kept by each network node in its own separate ARP table. The page shows the ARP table of the device.

Displayed values

The table has the following columns:

- **Interface**
  Shows the interface via which the row entry was learnt.
- **MAC address**
  Shows the MAC address of the target device.
- **IP address**
  Shows the IP address of the target device.
- **Media Type**
  Shows the type of connection.
  - Dynamic
    The device recognized the address data automatically.
  - Static
    The addresses were entered as static addresses

Log table

Logging events

Note

The page is only available if there is an online connection to the device.

The device allows you to log events that occur, some of which you can specify in " System > Events". This, for example, allows you to record when an authentication attempt failed or when the connection status of a port has changed. The content of the events log table is retained even when the device is turned off.

Settings

- **Severity Filters**
  You can filter the entries in the table according to severity. To display all the entries, enable or disable all settings.
  - Info
    Information (6)
  - Warning
    Warnings (4)
  - Critical
    Critical (2)

Displayed values

The table has the following columns:

- **Restart**
  Counts the number of restarts since you last reset to factory settings and shows the device restart after which the corresponding event occurred.
- **System up time**
  Shows the time the device has been running since the last restart when the described event occurred.
  If the system time is set, the time is also displayed at which the event occurred.
- **System Time**
  Shows the date and time of the device.
- **Severity**
  Shows the severity.
- **Log Message**
  Displays a brief description of the event that has occurred.

Buttons

- **Delete**
  Click this button to delete the content of the event log file. The display is also cleared. The restart counter is only reset after you have restored the device to the factory settings and restarted the device.

  Note

  The number of entries in this table is restricted to 400. When this number is reached, the oldest entries are overwritten. The table remains permanently in memory.

Fault

Error status

Note

The page is only available if there is an online connection to the device.

This page displays any errors that occur. If there are no more unanswered error/fault messages, the fault LED goes off.

The time calculation always begins after the last system start. When the system is restarted, a new entry with the type of restart is created in the fault memory.

Description of the displayed values

- **No. of Signaled Faults**
  Shows the number of reported faults.

The table contains the following columns:

- **Fault Time**
  Shows the time the device has been running since the last restart when the described error/fault occurred.
- **Fault description**
  Display of the fault state for the device.
- **Clear Fault State**
  When you can delete faults, the "Clear Fault State" button is enabled.

Redundancy

Spanning Tree

Note

The page is only available if there is an online connection to the device.

The page shows the current information about the Spanning Tree and the settings of the root bridge.

- If Spanning Tree is turned off, only the basic information about this device is displayed.
- If Spanning Tree is activated, the following is displayed:
  - Parameters of the selected instance
  - Parameters for the ports of the selected instance

  The information depends on the selected Spanning Tree mode.

Displayed values

- **Spanning Tree Mode**
  Shows the set mode. You specify the mode in "Layer 2 > MSTP > General".
  The following values are possible:
  - '-'
  - STP
  - RSTP
  - MSTP
- **Instance ID**
  Shows the number of the instance. The parameter depends on the configured mode.
- **Bridge Priority / Root Priority**
  Which device becomes the root bridge is decided by the bridge priority. The bridge with the highest priority (in other words, with the lowest value for this parameter) becomes the root bridge. If several devices in a network have the same priority, the device whose MAC address has the lowest numeric value will become the root bridge. Both parameters, bridge priority and MAC address together form the bridge identifier. Since the root bridge manages all path changes, it should be located as centrally as possible due to the delay of the frames. The value for the bridge priority is a whole multiple of 4096.
- **Bridge address / root address**
  The bridge address shows the MAC address of the device and the root address shows the MAC address of the root bridge.
- **Root Cost**
  Shows the path costs from the device to the root bridge.
  In MSTP mode, the path costs as far as the root bridge of the CIST are displayed.
- **Bridge Status**
  Shows the status of the bridge, e.g. whether or not the device is the root bridge.

- **Regional root priority** (available only with MSTP)
  For a description, see Bridge priority / Root priority.
- **Regional root address** (available only with MSTP)
  Shows the MAC address of the device.
- **Regional root costs** (only available with MSTP)
  The path costs from this device to the regional root bridge.

The table has the following columns:

- **Port**
  Shows the port via which the device communicates.
- **Role**
  Shows the status of the port. The following values are possible:
  - Disabled
    The port was removed manually from the spanning tree and will no longer be taken into account by the spanning tree.
  - Designated
    The ports leading away from the root bridge.
  - Alternate
    The port with an alternative route to a network segment
  - Backup
    If a switch has several ports to the same network segment, the "poorer" port becomes the backup port.
  - Root
    The port that provides the best route to the root bridge.
  - Master
    This port points to a root bridge located outside the MST region.
- **Status**
  Displays the current status of the port. The values are only displayed. The parameter depends on the configured protocol. The following statuses are possible:
  - Discarding
    The port receives BPDU frames. Other incoming or outgoing frames are discarded.
  - Listening
    The port receives and sends BPDU frames. The port is involved in the spanning tree algorithm. Other outgoing and incoming frames are discarded.
  - Learning
    The port actively learns the topology; in other words, the node addresses. Other outgoing and incoming frames are discarded.
  - Forwarding
    Following the reconfiguration time, the port is active in the network. The port receives and sends data frames.
- **Oper. Version**
  Describes the type of spanning tree in which the port operates

- **Priority**
  If the path calculated by the spanning tree is possible over several ports of a device, the port with the highest priority (in other words the lowest value for this parameter) is selected. A value between 0 and 240 can be entered for the priority in steps of 16. If you enter a value that cannot be divided by 16, the value is automatically adapted.
- **Path costs**
  This parameter is used to calculate the path that will be selected. The path with the lowest value is selected. If several ports of a device have the same value, the port with the lowest port number will be selected.
  If the value "Cost Calc." is "0", the automatically calculated value is displayed. Otherwise, the value of "Cost Calc." is displayed.
  The calculation of the path costs is largely based on the transmission speed. The higher the achievable transmission speed is, the lower the value of the path costs.
  Typical values for path costs with rapid spanning tree:
  - 10,000 Mbps = 2,000
  - 1000 Mbps = 20,000
  - 100 Mbps = 200,000
  - 10 Mbps = 2,000,000.
- **Edge type**
  Shows the type of the connection. The following values are possible:
  - Edge Port

    An edge port is connected to this port.
  - No Edge Port
    There is a spanning tree or rapid spanning tree device at this port.
- **P.t.P. Type**
  Shows the type of point-to-point link. The following values are possible:
  - P.t.P.
    With half duplex, a point-to-point link is assumed.
  - Shared Media

    With a full duplex connection, a point-to-point link is not assumed.

VRRP statistics

Introduction

Note

The page is only available if there is an online connection to the device.

This page shows the statistics of the VRRP protocol and all configured virtual routers.

Description of the displayed values

The following fields are displayed:

- VRID errors
  Shows how many VRRP frames containing an unsupported VRID were received.
- Version errors
  Shows how many VRRP frames containing an invalid version number were received.
- Checksum errors
  Shows how many VRRP frames containing an invalid checksum were received.

The table has the following columns:

- Interfaces
  Interface to which the settings relate.
- VRID
  Shows the ID of the virtual router.
  Valid values are 1 to 255.
- Change to master status
  Shows how often this virtual router changed to the "Master" role.
- Advertisements recd.
  Shows how often a VRRP frame was received that contained a bad address list.
- Advertisement Interval Errors
  Shows how many bad VRRP packets were received whose interval does not match the value set locally.
- IP TTL errors
  Shows how many bad VRRP packets were received whose TTL (Time to live) value in the IP header is incorrect.
- Prio 0 received
  Shows how many VRRP packets with priority 0 were received. VRRP packets with priority 0 are sent when a master router is shut down. These packets allow a fast handover to the relevant backup router.
- Prio 0 sent
  Shows how many VRRP packets with priority 0 were sent. Packets with priority 0 are sent when a master router is shut down. These packets allow a fast handover to the relevant backup router.
- Invalid Type
  Shows how often a VRRP packet was received that had the wrong type.
- Address List Errors
  Shows the number of errors in the address list.
- Invalid Auth. Type
  Shows how many bad VRRP packets were received whose authentication type was not type 0. Type 0 means "no authentication".

- **Auth. Type Mismatch**
  Shows how many bad VRRP frames were received whose authentication type does not match.
- **Frame length error**
  Shows how many bad VRRP frames were received whose length is not correct.

Ring Redundancy

Note

The page is only available if there is an online connection to the device.

Automatic Redundancy Detection (ARD) is the default when the device ships. If you want to use the previous High Speed Redundancy Protocol (HRP), HRP must be configured.

- Reconfiguration time of the frame traffic following a failover in MRP: 200 ms
- Reconfiguration time of the frame traffic following a failover in HRP: 300 ms

This page informs you about the values currently set for the device.

Description

The following fields are displayed:

- Redundancy Function
  The following values are shown:
  - Disabled
    Ring redundancy is disabled on the device.
  - MRP Manager
    The device is operating as MRP manager.
  - MRP Client
    The device is operating as MRP client.
  - HRP Client
    The device is operating as HRP client.
  - HRP Manager
    The device is operating as HRP manager.
- RM Status
  The "RM Status" column shows whether or not the IE switch is operating as redundancy manager and whether it has opened or closed the ring in this role.
  - Passive
    The IE switch is operating as redundancy manager and has opened the ring; in other words, the line of switches connected to the ring ports is operating problem free. The passive status is also displayed if the IE switch is not operating as the redundancy manager (RM Function Disabled).
  - Active
    The IE switch is operating as redundancy manager and has closed the ring; in other words, the line of switches connected to the ring ports is interrupted (problem). The redundancy manager connects its ring ports through and restores an uninterrupted linear topology.
  - If media redundancy in ring topologies is completely disabled, ring ports configured last are displayed and the text "Ring Redundancy disabled" is displayed.
- Observer Status
  Shows the current status of the observer.
- Ring ports
  Ring ports 1 and ring port 2 show the ports used for redundancy.
- No. of Changes to RM Active State
  Shows how often the device as redundancy manager switched to the active status, i.e. closed the ring.
  If the redundancy function is disabled or the device is not the HRP/MRP manager, the text "Redundancy manager disabled" appears.
- Max. delay of the RM test frames [ms]
  Shows the maximum delay for test frames of the redundancy manager.
  If the redundancy function is disabled or the device is not the HRP/MRP manager, the text "Redundancy manager disabled" appears.

Standby

Note

The page is only available if there is an online connection to the device.

This page shows the standby status of the device.

For more information on Ethernet cabling and the topological location of master and slave, refer to the "Industrial Ethernet Network Manual".

Description

The following fields are displayed:

- **Standby ports**
  Shows the ports being used.
- **Standby name**
  Shows the name for the standby connection. The master/slave device pair is defined by this name (both must be in the same ring). This is achieved by entering the same name on two devices in the ring. You can select any name to suit your purposes, however, you can only use the name for one pair of devices in the entire network.
- **Standby Function**
  Shows whether the standby connection is activated or deactivated.
  - Master
    The device has a connection to the partner device and is operating as master. In normal operation, the standby port of this device is active.
  - Slave
    The device has a connection to the partner device and is operating as slave. In normal operation, the standby port of this device is inactive.
  - Disabled
    The standby link is disabled. The device is operating neither as master nor slave. The standby port is working as a normal port without standby function.
  - Waiting for Connection...
    No connection has yet been established to the partner device. The standby port is inactive. In this case, either the configuration on the partner device is inconsistent (for example incorrect connection name, standby link disabled) or there is a physical fault (for example device failure, link down).
  - Connection Lost
    The existing connection to the partner device has been lost. In this case there is either a physical fault (for example, device failure, link down) or the configuration on the partner device was modified (for example different connection name, standby link disabled).

- **Standby status**
  Shows the status of the standby port:
  - Active
    The standby port of this device is active; in other words is enabled for frame traffic.
  - Passive
    The standby port of this device is inactive; in other words is blocked for frame traffic.
- **No. of Changes to Standby Active State**
  Shows the number of changes from the inactive to the active status.

Ethernet Statistics

Interface statistics

Interface statistics

The page shows the statistics from the interface table of the Management Information Base (MIB).

Note

The page is only available if there is an online connection to the device.

Displayed values

The table has the following columns:

- **Received bytes**
  Shows the number of received bytes.
- **Sent bytes**
  Shows the number of sent bytes.
- **Received unicast frames**
  Shows the number of received unicast frames.
- **Received non-unicast frames**
  Shows the number of received frames that are not of the type unicast.
- **Sent unicast frames**
  Shows the number of sent unicast frames.
- **Sent non-unicast frames**
  Shows the number of sent frames that are not of the type unicast.

Frame length

Frames sorted by length

This page displays how many frames of which size were sent and received at each port.

Note

The page is only available if there is an online connection to the device.

Displayed values

The table has the following columns:

- **Port**
  Shows the available ports and link aggregations.

  Note
  Display of frame statistics
  In the statistics relating to frame lengths, note that both incoming and outgoing frames are counted.

- **Frame lengths**
  The other columns after the port number contain the absolute numbers of incoming frames according to their frame length.
  The following frame lengths are distinguished:
  - 64 bytes
  - 65 - 127 bytes
  - 128 - 255 bytes
  - 256 - 511 bytes
  - 512 - 1023 bytes
  - 1024 - max.

Button

Reset Counter

Click "Reset Counter" to reset all counters. The counters are also reset by a restart.

Frame type

Received frames sorted by type

This page displays how many frames of the type "unicast", "multicast", and "broadcast" were received at each port.

Note

The page is only available if there is an online connection to the device.

Displayed values

The table has the following columns:

- **Port**
  Shows the available ports and link aggregations.
- **Unicast/Multicast /Broadcast**
  The other columns after the port number contain the absolute numbers of the incoming frames according to their frame type "unicast", "multicast" and "broadcast".

Button

Reset Counter

Click "Reset Counter" to reset all counters. The counters are also reset by a restart.

Packet Errors

Received bad frames

This page shows how many bad frames were received per port.

Note

The page is only available if there is an online connection to the device.

Displayed values

The table has the following columns:

- **Port**
  Shows the available ports and link aggregations.
- **Error types**
  The other columns after the port number contain the absolute numbers of the incoming frames according to their error type.
  In the columns of the table, a distinction is made according to the following error types:
  - CRC
    Frames whose content does not match the CRC checksum.
  - Undersize
    Frames with a length less than 64 bytes.
  - Oversize
    Frames discarded because they were too long.
  - Fragments
    Frames with a length less than 64 bytes and a bad CRC checksum.
  - Jabbers
    VLAN-tagged frames with an incorrect CRC checksum that were discarded because they were too long.
  - Collisions
    Collisions that were detected.

Button

Reset Counter

Click "Reset Counter" to reset all counters. The counters are also reset by a restart.

History

Samples of the statistics

The page shows samples from each port with information from the statistics.

Note

The page is only available if there is an online connection to the device.

Settings

- Port
  Select the port for which the history will be displayed.

Displayed values

- Entries
  Maximum number of samples that can be saved at the same time.
- Interval [s]
  Interval after which the current status of the statistics will be saved as a sample.

The table has the following columns:

- Sample
  Number of the sample
- Time for the Sample
  System up time at which the sample was taken.
- Unicast
  Number of received unicast frames.
- Multicast
  Number of received multicast frames.
- Broadcast
  Number of received broadcast frames.
- CRC
  Number of frames with a bad CRC checksum.
- Undersize
  Number of frames that are shorter than 64 bytes.
- Oversize
  Number of frames discarded because they were too long.
- Fragments
  Number of frames that are shorter than 64 bytes and have a bad CRC checksum.
- Jabbers
  Number of frames with a VLAN tag that have a bad CRC checksum and will be discarded because they are too long.
- Collisions
  Number of collisions of received frames.
- Utilization
  Utilization of the port during a sample.

Unicast

This page shows the learnt and static unicast MAC addresses.

Description

- VLAN ID
  Shows the VLAN-ID assigned to this MAC address.
- MAC Address
  Shows the MAC address of the node that the device has learned or the user has configured.
- Status
  Shows the status of each address entry:
  - Learnt
    The specified address was learned by receiving a frame from this node and will be deleted when the aging time expires if no further packets are received from this node.
  - Static
    Configured by the user. Static addresses are stored permanently; in other words, they are not deleted when the aging time expires or when the switch is restarted.
- Port
  Shows the port via which the node with the specified address can be reached. Frames received by the device whose destination address matches this address will be forwarded to this port.

Multicast

This page shows the learnt and static multicast MAC addresses.

Description

- VLAN ID
  Shows VLAN ID of the VLAN to which the MAC multicast address is assigned.
- MAC Address
  Shows the MAC multicast address that the device has learned or the user has configured.
- Status
  Shows the status of each address entry. The following information is possible:
  - static
    The address was entered statically by the user. Static addresses are stored permanently; in other words, they are not deleted when the aging time expires or when the device is restarted. These must be deleted by the user.
  - IGMP
    The destination port for this address was obtained by IGMP configuration.
  - GMRP
    The destination port for this address was registered by a received GMRP frame.

Routing

Routing table

This page shows the routing table of the device.

Description

The table has the following columns:

- **Destination network**
  Shows the destination address of this route.
- **Subnet mask**
  Shows the subnet mask of this route.
- **Gateway**
  Shows the gateway for this route.
- **Interface**
  Shows the interface for this route.
- **Metric**
  Shows the metric of the route. The higher value, the longer the packets require to their destination.
- **Routing protocol**
  Shows the routing protocol from which the entry in the routing table originates. The following entries are possible:
  - Connected: Connected routes
  - Static: Static routes
  - RIP: Routes using RIP
  - OSPF: Routes using OSPF
  - other: Other routes

OSPFv2 interfaces

Overview

Note

The page is only available if there is an online connection to the device.

This page shows the configuration of the OSPF interface.

Description of the displayed values

The table has the following columns:

- **IP address**
  Shows the IP address of the OSPF interface
- **Area ID**
  Shows the Area ID to which the OSPF interface belongs.

- Interface Status
  Shows the status of the WLAN interface:
  - Down
    The interface is not available.
  - Loopback
    Loopback interface
  - Waiting
    Startup and negotiation of the interface.
  - Point to Point
    Point-to-point connection
  - Designated Router
    The router is a designated router and generates a network LSA.
  - Backup D. Router
    The router is backup for the designated router.
  - Other D. Router
    The interface is started up. The router is neither a designated router nor a backup designated router.
- OSPF status
  Shows the status of OSPF.
  - Enabled: OSPF is enabled on the interface.
  - Disabled: OSPF is disabled on the interface.
- Designated Router
  Shows the IP address of the designated router for this OSPF interface.
- Backup Designated Router
  Shows the IP address of the designated backup router for this OSPF interface.
- Events
  Shows the number of status changes of OSPF.

OSPFv2 neighbors

Note

The page is only available if there is an online connection to the device.

This page shows the dynamically detected neighbor routers in the relevant networks.

Description of the displayed values

The table has the following columns:

- IP address
  Shows the IP address of the neighbor router in this network.
- Router ID
  Shows the ID of the neighbor router. The two addresses can match.

- **Status**
  Shows the status of the neighbor router. The status can adopt the following values:
  - unknown
    Status of the neighbor router is unknown.
  - down
    The neighbor router cannot be reached.
  - attempt and init
    Short-lived statuses during initialization
  - two-way
    Two-way receipt of Hello packets. Specification of the designated router and the designated backup router.
  - exchangestart, exchange and loading
    Status during exchange of the LSAs
  - full
    The database is complete and synchronized within the area. The routes can now be detected.

  Note
  Normal status
  If the partner router is a designated router or a designated backup router, the status is "full". Otherwise the status is "two-way".

- **Assoc. Area Type**
  Shows the area type via which the neighbor-neighbor relation is maintained. The following area types exist:
  - Normal
  - Stub
  - NSSA
  - Backbone
- **Priority**
  Shows the priority of the neighbor router. This is only significant when selecting the designated router on a network. For virtual neighbor routers, this information is irrelevant.
- **Hello Suppr.**
  Shows whether there are suppressed Hello packets to the virtual neighbor router.
  - no: There are no suppressed Hello packets (default).
  - yes: There are suppressed Hello packets.
- **Hello Queue**
  Shows the length of the queue with Hello packets still to be transmitted.
- **Events**
  Shows the number of status changes.

OSPFv2 virtual neighbors

Note

The page is only available if there is an online connection to the device.

This page shows the configured virtual neighbor routers.

Overview

The table has the following columns:

- **IP address**
  Shows the IP address of the virtual neighbor router in this network.
- **Router ID**
  Shows the router ID of the virtual neighbor router.
- **Status**
  Shows the status of the neighbor router. The status can adopt the following values:
  - unknown
    Status of the neighbor router is unknown.
  - down
    The neighbor router cannot be reached.
  - attempt and init
    Short-lived statuses during initialization
  - two-way
    Two-way receipt of Hello packets. Specification of the designated router and the designated backup router.
  - exchangestart, exchange and loading
    Status during exchange of the LSAs
  - full
    The database is complete and synchronized within the area. The routes can now be detected.

    Note

    Normal status

    If the partner router is a designated router or a designated backup router, the status is "full". Otherwise the status is "two-way".
- **Trans. Area ID**
  Shows the ID of the area via which the virtual neighborhood relation exists.
- **Hello Suppr.**
  Shows whether there are suppressed Hello packets to the virtual neighbor router.
  - no: There are no suppressed Hello packets (default)
  - yes: There are suppressed Hello packets.

- **Hello Queue**
  Shows the length of the queue with Hello packets still to be transmitted.
- **Events**
  Shows the number of status changes.

OSPFv2 LSDB

Overview

Note

The page is only available if there is an online connection to the device.

The link state database is the central database for managing all links within an area. It consists of the link state advertisements (LSAs). The most important data of these LSAs is shown on the this WBM page.

Description of the displayed boxes

The table has the following columns:

- **Area ID**
  Shows the ID of the area to which the LSA belongs. If the LSA is an external connection, '-' is displayed.
- **LSA type**
  Shows the LSA type. The following values are possible:
  - unknown
    LSA type is unknown.
  - Router
    The router LSA (Type 1) is sent by the OSPF router within an area. The LSA contains information about the status of all router interfaces.
  - Network
    The network LSA (Type 2) is sent by the designated router within an area. The LSA contains a list of routers connected to the network.
  - NSSA External
    The NSSA external LSA (Type 7) is sent by the NSSA-ASBR within an NSSA. The NSSA-ASBR receives LSAs of Type 5 and converts the information to LSAs of Type 7. The NSSA router can forward these LSAs within an NSSA.
  - Summary
    The summary LSA (Type 3) is sent by the ABR within an area. The LSA contains information about routes to other networks.
  - AS Summary
    The AS summary LSA (Type 4) is sent by the area border router within an area. The LSA contains information about routes to other autonomous systems.
  - AS External
    The AS external LSA (Type 5) is sent by the AS border router within an autonomous system. The LSA contains information about routes from one network to another.
- **Link State ID**
  Shows the ID of the LSA.
- **Router ID**
  Shows the ID of the router that sent this LSA.
- **Sequence number**
  Shows the sequence number of the LSA. Each time an LSA is renewed, this sequence number is incremented by one.

RIPv2 statistics

Overview

Note

The page is only available if there is an online connection to the device.

This page shows the statistics of the RIP interface.

Description of the displayed values

The table has the following columns:

- **IP address**
  Shows the IP address of the RIPv2 interface
- **Bad packets**
  Number of received RIP packets that were deleted and therefore ignored.
- **Bad Routes**
  Number of routes of valid RIP packets that could not be taken into consideration.
- **Updates Sent**
  Shows how often the router has sent its routing table to its neighbor routers.

System

Configuration

On this page, you specify the services with which the device can be accessed. With some services, there are further configuration pages on which more detailed settings can be made.

Settings

- **Telnet Server**
  Enable or disable the "Telnet Server" service for unencrypted access to the CLI.
- **SSH server**
  Enable or disable the "SSH Server" service for encrypted access to the CLI.
- **HTTPS server only**
  Enable or disable access using HTTP.
- **DNS client**
  Enable or disable depending on whether the IE switch should operate as a DNS client. You can configure other settings in "System > DNS Client".
- **SMTP Client**
  Enable or disable the SMTP client. You can configure other settings in "System > SMTP Client".
- **Syslog Client**
  Enable or disable the system event client. You can configure other settings in "System > Syslog client".

- DCP Server
  Specify whether or not the device can be accessed with DCP (Discovery and Configuration Protocol):
  - "-" (disabled)
    DCP is disabled. Device parameters can neither be read nor modified.
  - Read/write access
    With DCP, device parameters can be both read and modified.
  - Read-Only
    With DCP, device parameters can be read but cannot be modified.
- Time
  Select the required setting. The following settings are possible:
  - Manual
    The system time is set manually.
  - SNTP Client
    The system time is set via an SNTP server. You can configure other settings in "System > System time > SNTP client".
  - NTP Client
    The system time is set via an NTP server. You can configure other settings in "System > System time > NTP client".
  - SIMATIC Time
    The system time is set using a SIMATIC time transmitter. You can configure other settings in "System > System Time SIMATIC Time Client".
  - PTP Client
    The system time is set via a PTP server. You can configure other settings in "System > System Time > PTP Client".
    This function is only available for SCALANCE X500.
- SNMP
  Select the required protocol. The following settings are possible:
  - "-" (SNMP disabled)
    Access to device parameters using SNMP is not possible.
  - SNMPv1/v2c/v3
    Access to device parameters is possible with SNMP versions 1, 2c or 3. You can configure other settings in "System > SNMP > General".
  - SNMPv3
    Access to device parameters is possible with SNMP version 3. You can configure other settings in "System > SNMP > General".
- SNMPv1/v2 Read-Only
  Enable or disable write access to SNMP variables with SNMPv1/v2c.
- SNMPv1 traps
  Enable or disable the sending of SNMP traps (alarm frames). You can configure other settings in "System > SNMP > Traps".

- NFC (Near Field Communikation)
  Enable or disable the "NFC" function.
  This function is only available for SCALANCE XM400.
- Configuration mode
  Select the required mode. The following modes are possible:
  - Automatic save
    Automatic save mode. Approximately 1 minute after the last parameter change or when you restart the device, the configuration is automatically saved.
  - Trial
    In the "Trial" configuration mode, although changes are adopted, they are not saved in the configuration file (startup configuration).
    To save changes in the configuration file, use the "Write startup config" button. The button is displayed when you set this configuration mode. The display area also shows the message "Trial mode active - Press "Write Startup Config" button to make your settings persistent" as soon as there are unsaved modifications. This message can be seen on every page until the changes made have either been saved or the device has been restarted.

General

Device

This page contains the general device information.

Settings

- Current System Time(Only available online)
  Shows the current system time. The system time is either set by the user or by a time-of-day frame: either SINEC H1 time-of-day frame, NTP or SNTP. (readonly)
- System Up Time(only available online)
  Shows the operating time of the device since the last restart. (readonly)
- Device Type(only available online)
  Shows the type designation of the device. (readonly)
- System Name
  You can enter the name of the device. The entered name is displayed in the selection area. A maximum of 255 characters are possible. The system name is also displayed in the CLI input prompt. The number of characters in the CLI input prompt is limited. The system name is truncated after 16 characters.

- System Contact
  You can enter the name of a contact person responsible for managing the device.
- Location
  You can enter the location of the device. The entered installation location is displayed in the selection area.

Note

The ASCII characters 0x20 to 0x7e are used in the input boxes.

At the start and end of the input boxes"System Name", "System Contact" and "System Location", the characters "<", ">" and "space" are not permitted.

Coordinates

On this page, you configure the geographic coordinates (latitude, longitude and the height above the ellipsoid according to WGS84). These boxes are purely for information with a maximum length of 32 characters.

Getting the coordinates

Use suitable maps for obtaining the geographic coordinates of the device.

The geographic coordinates can also be obtained using a GPS receiver. The geographic coordinates of these devices are normally displayed directly and only need to be entered in the input boxes of this page.

Settings

- Latitude
  Enter the north or south latitude for the location of the device.
  For example, +49° 1´ 31.67" means that the device is located at 49 degrees, 1 arc minute and 31.67 arc seconds north latitude.
  A south latitude is shown by a preceding minus character.
  You can also append the letters N (north latitude) or S (south latitude) to the numeric information (49° 1´ 31.67" N).
- Longitude
  Enter the value of the eastern or western longitude of the location of the device.
  For example, +8° 20´ 58.73" means that the device is located at 8 degrees, 20 minutes and 58.73 seconds east.
  A western longitude is indicated by a preceding minus sign.
  You can also add the letter E (eastern longitude) or W (western longitude) to the numeric information (8° 20´ 58.73" E).
- Height
  Enter the value of the height above sea level in meters.
  For example, 158 m means that the device is located at a height of 158 m above sea level.
  Heights below sea level (for example the Dead Sea) are indicated by a preceding minus sign.

Agent IP

Configuration of the IP addresses

Note

The page is only available if there is an online connection to the device.

On this page, you set the IP configuration for the device.

With devices with more than one IP interface, this call references the "Subnets > Configuration" menu item in the "Layer 3" menu and the configuration of the TIA interface there.

Note

The IP address of the in-band port and the out-band port must belong to different subnets.

Settings

- **IP Address**
  In the input boxes under "In band", enter the IP address, subnet mask and the default gateway. In the input boxes under "Out band", enter the IP address, subnet mask and the default gateway.
  If you change the IP address, the Web browser should automatically set itself to the new address. If this does not happen, enter the new address in the Web browser manually.
- **Subnet mask**
  Here, in "In-band", you enter the subnet mask of the in-band port and in "Out-band" the subnet mask of the out-of-band port.
- **Default Gateway**
  If the device is required to communicate with devices (diagnostics stations, e-mail servers etc.) in another subnet, enter the IP address of the default gateway here. The out-of-band port it is not accessible from a different subnet.
- **Agent VLAN ID**
  From the drop-down list, select the VLAN ID for the in-band management. You can only select VLANs that have already been configured.

  Note
  Changing the agent VLAN ID

  If the configuration PC is connected directly to the device via Ethernet and you change the agent VLAN ID, the device is no longer reachable via Ethernet following the change.

- **MAC Address**
  Shows the MAC address of the device. The MAC address is linked to the hardware and cannot be modified.

DNS client

The DNS () server (Domain Name System) assigns a domain name to an IP address so that a device can be uniquely identified.

If this function is enabled, the IE switch can communicate with a DNS server as a DNS client.

Note

The DNS client function can only be used if there is a DNS server in the network.

Description

The page contains the following boxes:

- **DNS Client**
  Enable or disable depending on whether the IE switch should operate as a DNS client.
- **DNS Server IP Address**
  Enter the IP address of the DNS server.

Restart

On this page, there is a button with which you can restart the device and various options for resetting to the device defaults.

Note

The page is only available if there is an online connection to the device.

Note

Note the following points about restarting a device:

- You can only restart the device with administrator privileges.
- A device should only be restarted with the buttons of this page or with the appropriate CLI commands and not by a power cycle on the device.
- Any modifications you have made only become active on the device after clicking the "Set Values" button on the relevant page. If the device is in "Trial Mode", configuration modifications must be saved manually before a restart. In "Automatic Save" mode, the last changes are saved automatically before a restart.

Settings

The following options are available for restarting:

- Restart
  Click this button to restart the system. You must confirm the restart in a dialog box. During a restart, the device is reinitialized, the internal firmware is reloaded, and the device runs a self-test. The learned entries in the address table are deleted. You can leave the browser window open while the device restarts. You then need to log in again:
- Restore Factory Defaults and Restart
  Click this button to restart the system. You must confirm the restart in a dialog box. During a restart, the device is reinitialized, the internal firmware is reloaded, and the device runs a self-test. The learned entries in the address table are deleted. You can leave the browser window open while the device restarts. You then need to log in again.

Note

By resetting all the defaults to the factory settings, the IP address and the passwords are also lost. Following this, the device can only be accessed using the Primary Setup Tool or using DHCP.

With the appropriate attachment, a previously correctly configured device can cause circulating frames and therefore the failure of the data traffic.

Load & save

Uploading and saving using HTTP

Loading and saving data via HTTP

On this page, you store device data in an external file on your client PC or load such data from an external file from the PC to the devices. This means, for example, that you can also load new firmware from a file located on your client PC.

Note

Incompatibility with predecessor versions

During the installation of a previous version, the configuration data can be lost. In this case, the device starts up with the factory settings after the firmware has been installed.

Incompatibility with previous versions with PLUG inserted

During the installation of a previous version, the configuration data can be lost. In this case, the device starts up with the factory settings after the firmware has been installed. In this situation, if a PLUG is inserted in the device, following the restart, this has the status "Not Accepted" since the PLUG still has the configuration data of the previous more up-to-date firmware. This allows you to return to the previous, more up-to-date firmware without any loss of configuration data. If the original configuration on the PLUG is no longer required, the PLUG can be deleted or rewritten manually using System > PLUG.

Note

Configuration files and Trial mode/Automatic Save

In "Automatic Save" mode, the data is saved automatically before the configuration files (ConfigPack and Config) are transferred.
In Trial mode, although the changes are adopted, they are not saved in the configuration files (ConfigPack and Config). Use the "Write Startup Config" button on the "System > Configuration" WBM page to save changes in the configuration file.

Settings

The table has the following columns:

- **Type**
  Shows the file type.
- **Description**
  Shows the short description of the file type.
- **Load**
  With this button, you can upload files to the device. The button can be enabled, if this function is supported by the file type.
- **Save**
  With this button, you can save files from the device. The button can only be enabled if this function is supported by the file type and the file exists on the device.
- **Delete**
  With this button, you can delete files from the device. The button can only be enabled if this function is supported by the file type and the file exists on the device.

Loading new firmware

After successfully downloading the firmware, you will be requested to restart the device.

Restart the device and continue to configure the newly started firmware.

Procedure

Loading files using HTTP

1. Start the upload function by clicking the one of the "Load" buttons.
   The dialog for uploading a file opens.
2. Go to the file you want to upload.
3. Click the "Open" button in the dialog.
   The file is now uploaded.
4. When the message "Restart required" appears, click the "Yes" button to trigger the restart.
   If you click the "No" button, there is no device restart. The changes only take effect after a restart.

Saving files using HTTP

1. Start the save function by clicking the one of the "Save" buttons.
2. You will be prompted to select a storage location and a name for the file. Or you accept the proposed file name. To make the selection, use the dialog in your browser. After making your selection, click the "Save" button.

Deleting files using HTTP

1. Start the delete function by clicking the one of the "Delete" buttons.
   The file will be deleted.

Reusing configuration data

If several devices are to receive the same configuration and the IP addresses are assigned using DHCP, the effort for configuration can be reduced by saving and reading in the configuration data.

Follow the steps below to reuse configuration data:

1. Save the configuration data of a configured device on your PC.
2. Download this configuration file to all other devices you want to configure in this way.
3. If individual settings are necessary for specific devices, these must be made online on the relevant device.

Note that the configuration data is coded when it is saved. This means that you cannot edit the files with a text editor.

Uploading and saving using TFTP

Loading and saving data via a TFTP server

On this page, you can configure the TFTP server and the file names. You can also store device data in an external file on your client PC or load such data from an external file from the PC to the devices. This means, for example, that you can also load new firmware from a file located on your client PC.

Note

Incompatibility with predecessor versions

During the installation of a previous version, the configuration data can be lost. In this case, the device starts up with the factory settings after the firmware has been installed.
Incompatibility with predecessor versions with inserted PLUG
During the installation of a previous version, the configuration data can be lost. In this case, the device starts up with the factory settings after the firmware has been installed. In this situation, if a PLUG is inserted in the device, following the restart, this has the status "Not Accepted" since the PLUG still has the configuration data of the previous more up-to-date firmware. This allows you to return to the previous, more up-to-date firmware without any loss of configuration data. If the original configuration on the PLUG is no longer required, the PLUG can be deleted or rewritten manually on the "System > PLUG page.

Note

Configuration files and Trial mode/Automatic Save

In "Automatic Save" configuration mode, the data is saved automatically before the configuration files (ConfigPack and Config) are transferred.
In the "Trial" configuration mode, although the changes are adopted, they are not saved in the configuration files (ConfigPack and Config). Use the "Write Startup Config" button on the "System > Configuration" WBM page to save changes in the configuration files.

Settings

- **TFTP Server IP Address**
  Here, enter the IP address of the TFTP server with which you exchange data.
- **TFTP Server Port**
  Here, enter the port of the TFTP server via which data exchange will be handled. If necessary, you can change the default value 69 to your own requirements.

The table has the following columns:

- **File type**
  Shows the file type.
- **Description**
  Shows the short description of the file type.
- **File name**
  Enter a file name.
- **Select action** (only available online)
  Select the action from the drop-down list. The selection depends on the selected file type, for example the log file can only be saved.
  The following actions are possible:
  - Save file
    With this selection, you save a file on the TFTP server.
  - Load file
    With this selection, you load a file from the TFTP server.

Events

Configuration

On this page, you specify how the device reacts to events.

Settings

- **Signaling Contact Response**
  Select the behavior of the signaling contact. The following reactions are possible:
  - conventional
    Default setting for the signaling contact. An error/fault is displayed by the fault LED and the signaling contact is opened. When the error/fault state no longer exists, the fault LED goes off and the signaling contact is closed.
  - User-defined
    The way the signaling contact works does not depend on the error/fault that has occurred. The signaling contact can be opened or closed as required by user actions.
- **Signaling Contact Status**
  Select the status of the signaling contact. The following statuses are possible:
  - close
    Signaling contact is closed.
  - open
    Signaling contact is opened.

Table 1 has the following columns:

- **1st column**
  Shows that the settings are valid for all ports of table 2.
- **E-mail/Trap/Log table/Syslog/Errors**
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Event**
  The "Event" column contains the following values:
  - Cold/Warm Start
    The device was turned on or restarted by the user.
  - Link Change
    This event occurs only when the port status is monitored and has changed, see "System > Fault Monitoring > Link Change".
  - Authentication Failure
    This event occurs when attempting access with a bad password.
  - RMON Alarm
    An alarm or event has occurred relating to the remote monitoring of the system.
  - Power Supply Switchover
    This event occurs only when power supply lines 1 and 2 are monitored. It indicates that there was a change to line 1 or line 2, see System > Fault monitoring > Power supply".
  - RM status change
    The status of the redundancy manager has changed
  - Spanning Tree Topology Change
    The STP or RSTP or MSTP topology has changed.
  - Fault State Change
    The fault state has changed. The fault state can relate to the activated port monitoring, the response of the signaling contact or the power supply monitoring.
  - VRRP State Change (only with routing using VRRP)
    The status of the virtual router has changed
  - Loop Detection
    A loop was detected in the network segment.
  - OSPF State Change (only with routing using OSFP)
    The status of OSPF has changed
- **E-mail**
  The device sends an e-mail. This is only possible if the SMTP server is set up and the "SMTP client" function is enabled.
- **Trap**
  The device sends an SNMP trap. This is only possible if "SNMPv1 traps" is enabled in "System > Configuration".
- **Log table**
  The device writes an entry in the log table.
- **Syslog**
  The device writes an entry to the system log server. This is only possible if the system log server is set up and the "Syslog client" function is enabled.
- **Fault**
  The device triggers a fault. The error LED lights up

Severity filter

On this page, set the threshold levels for sending system event notifications.

Settings

The table has the following columns:

- **Client Type**
  Select the client type for which you want to make settings:
  - E-mail
    Sending system event messages by e-mail
  - Log table
    Entry of system events in the log table.
  - Syslog
    Entering system events in the Syslog file
- **Severity**
  Select the required level. The following settings are possible:
  - Info
    System events are processed as of the severity level "Info".
  - Warning
    System events are processed as of the severity level "Warning".
  - Critical
    System events are processed as of the severity level "Critical".

SMTP client

The device provides the option of automatically sending an e-mail if an alarm event occurs (for example to the network administrator). The e-mail contains the identification of the sending device, a description of the cause of the alarm in plain language, and a time stamp. This allows centralized network monitoring to be set up for networks with few nodes based on an E-mail system. When an e-mail event message is received, the WBM can be started by the browser using the identification of the sender to read out further diagnostics information.

On this page, you can configure up to three SMTP servers and the corresponding e-mail addresses.

Note

Depending on the properties and configuration of the SMTP server, it may be necessary to adapt the "From” input box for the e-mails. Check with the administrator of the SMTP server.

Settings

- **SMTP Client**
  Enable or disable the SMTP client.
- **Email address of the sender**
  Enter the name of the sender that will be included in the e-mail, for example the device name.

- **SMTP Port**
  Change the port if the SMTP server is not obtainable via port 25.
- **SMTP Server IP Address**
  Enter the IP address of the SMTP server.

This table contains the following columns:

- **SMTP Server IP Address**
  Shows the SMTP server IP address.
- **Email address of the recipient**
  Enter the e-mail address to which the device sends an e-mail if a fault occurs. The e-mail address can be the address of a single person or a distribution list.

Syntax of email addresses

The following conditions apply to e-mail addresses:

- Alphanumeric characters are permitted.
- The following special characters are permitted:
  - @
  - _ (underscore)
  - - (hyphen)
  - . (period)
- A @ character must be included.
- Only one @ may be used.
- The characters "@" and the "." must not be the first or last character.

DHCP client

If the DHCP mode is activated, the DHCP client starts a DHCP request to a configured DHCP server and is assigned an IP address as the response. The server manages an address range from which it assigns IP addresses. It is also possible to configure the server so that the client always receives the same IP address in response to its request.

Settings

- **DHCP client configuration request (opt. 66, 67)**
  Select this option if you want the DHCP client to use options 66 and 67 to download and then enable a configuration file.
- **DHCP Mode**
  Select the DHCP mode. The following modes are possible:
  - via MAC Address
    Identification is based on the MAC address.
  - via DHCP Client ID
    Identification is based on a freely defined DHCP client ID.
  - via System Name
    Identification is based on the device name. If the device name is 255 characters long, the last character is not used for identification.

This table contains the following columns:

- **Interface**
  Shows the available interfaces.
- **DHCP**
  Enable or disable the DHCP client.

SNMP

General

On this page, you make the basic settings for SNMP. Enable the functions you want to use.

Settings

- **SNMP**
  Select the SNMP protocol. The following settings are possible:
  - "-" (disabled)
    SNMP is disabled.
  - SNMPv1/v2c/v3
    SNMPv1/v2c/v3 is supported.
  - SNMPv3
    Only SNMPv3 is supported.
- **SNMPv1/v2 Read-Only**
  If you enable this option, SNMPv1/v2c can only read the SNMP variables.

> **Note**
>
> **Community String**
>
> For security reasons, do not use the standard values "public" or "private". Change the community strings following the initial installation.

- SNMPv1/v2c Read Community String
  Enter the community string for access of the SNMP protocol.
- SNMPv1/v2c Read/Write Community String
  Enter the community string for read and write access of the SNMP protocol.
- SNMPv1 Traps
  Enable or disable the sending of SNMP traps (alarm frames). On the "Trap" tab, specify the IP addresses of the devices to which SNMP traps will be sent.
- SNMPv1/v2c Trap Community String
  Enter the community string for sending SNMPv1/v2 messages.

Traps

If an alarm event occurs, a device can send SNMP traps (alarm frames) to up to ten different management stations at the same time. Traps are only sent if events specified in "Events > Configuration" occur.

Note

SNMP traps are sent only when the "SNMPv1 Traps" setting was selected in "SNMP > General".

Settings

- IP Address
  Enter the IP address of the stations to which the device sends SNMP traps

The table has the following columns:

- IP Address
  Shows the IP addresses of the stations to which the device sends SNMP traps.
- Trap
  Enable or disable the sending of SNMP traps. Stations that are entered but not selected do not receive SNMP traps.

v3 Groups

Security settings and assigning permissions

SNMP version 3 allows permissions to be assigned, authentication, and encryption at protocol level. The security levels and read/write permissions are assigned according to groups. The settings automatically apply to every member of a group.

Settings

- **Group Name**
  Enter the name of the group. This name must have at least two characters, the maximum length is 32 characters.
- **Security Level**
  Select the security level (authentication, encryption) valid for the selected group. You have the following options for the security levels:
  - no Auth/no Priv
    No authentication enabled / no encryption enabled.
  - Auth/no Priv
    Authentication enabled / no encryption enabled.
  - Auth/Priv
    Authentication enabled / encryption enabled.

The table has the following columns:

- **Group Name**
  Shows the defined group names.

  Note

  Once a group name and the security level have been specified, they can no longer be modified after the group is created. If you want to change the group name or the security level , you will need to delete the group and recreate it and reconfigure it with the new name.

- **Security Level**
  Shows the configured security level.
- **Read**
  Enable or disable read access.
- **Write**
  Enable or disable write access.
- **Persistence**
  Shows whether or not the group is assigned to an SNMPv3 user. If the group is not assigned to an SNMPv3 user, no automatic saving is triggered and the configured group disappears again after restarting the device.
  - Yes
    The group is assigned to an SNMPV3 user.
  - No
    The group is not assigned to an SNMPV3 user.

v3 users

User-specific security settings

On the WBM page, you can create new SNMPv3 users and modify or delete existing users. The user-based security model works with the concept of the user name; in other words, a

user ID is added to every frame. This user name and the applicable security settings are checked by both the sender and recipient.

Settings

- User name
  Enter a freely selectable user name. After you have entered the data, you can no longer modify the name.

The table has the following columns:

- User name
  Shows the created users.
- Group Name
  Select the group to which the user will be assigned.
- Authentication Protocol
  Select the authentication protocol. Can only be enabled, if this group supports the function. The following settings are available:
  - None
  - MD5
  - SHA
- Encryption Protocol
  Specify whether or not the user uses the DES algorithm. Can only be enabled, if the group supports this function.
- Authentication Password
  Enter the authentication password in the first input box.
- Confirm authentication password
  Confirm the password by repeating the entry.
- Encryption Password
  Enter your encryption password.
- Confirm Encryption Password
  Confirm the encryption password by repeating the entry.
- Persistence
  Shows whether or not the user is assigned to an SNMPv3 group. If the user is not assigned to an SNMPv3 group, no automatic saving is triggered and the configured user disappears again after restarting the device.
  - Yes
    The user is assigned to an SNMPv3 group.
  - No
    The user is not assigned to an SNMPv3 group.

System time

Manual time setting

On this page, you set the date and time of the system. For this setting to be used, enable "Manual time setting".

Note

The page is only available if there is an online connection to the device.

Settings

- **Time Manually**
  Enable or disable the manual time setting.
- **System Time**
  Enter the date and time in the format "MM/DD/YYYY HH:MM:SS". Can only be edited, if manual time setting is enabled.
  After a restart, the time of day begins at 01/01/2000 00:00:00
- **Use PC Time**
  Click the button to use the time setting of the PC.
- **Last Synchronization Time**
  This box is read-only and shows when the last time-of-day synchronization took place. If no time-of-day synchronization was possible, the box displays "Date/time not set".
- **Last Synchronization Mechanism**
  This box displays how the last time-of-day synchronization was performed.
  - Not set
    The time was not set.
  - Manual
    Manual time setting
  - SNTP
    Automatic time-of-day synchronization using SNTP
  - NTP
    Automatic time-of-day synchronization using NTP
  - SIMATIC
    Automatic time-of-day synchronization using the SIMATIC time frame.
  - PTP
    Automatic time-of-day synchronization with PTP

DST overview

On this page, you can create new entries for the daylight saving time changeover.

The table provides an overview of the existing entries.

Settings

- DST No.
  Shows the number of the entry.
  If you create a new entry, a new line with a unique number is created.
- Name
  Shows the name of the entry.
- Year
  Shows the year for which the entry was created.
- Start Date
  Shows the month, day and time for the start of daylight saving time.
- End Date
  Shows the month, day and time for the end of daylight saving time.
- Type
  Shows how the daylight saving time changeover is made:
  - Date
    A fixed date is entered for the daylight saving time changeover.
  - Recurring
    A rule was defined for the daylight saving time changeover.

DST configuration

On this page, you can configure the entries for the daylight saving time changeover. As result of the changeover to daylight saving or standard time, the system time for the local time zone is correctly set.

You can define a rule for the daylight saving time changeover or specify a fixed date.

Settings

Note

The content of this page depends on the selection in the "Type" box.

The boxes "DST No.", "Type" and "Name" are always shown.

- **DST No.**
  Select the type of the entry.
- **Type**
  Select how the daylight saving time changeover is made:
  - Date
    You can set a fixed date for the daylight saving time changeover.
    This setting is suitable for regions in which the daylight saving time changeover is not governed by rules.
  - Recurring
    You can define a rule for the daylight saving time changeover.
    This setting is suitable for regions in which the daylight saving time always begins or ends on a certain weekday.
- **Name**
  Enter a name for the entry.

Settings with "Date" selected

You can set a fixed date for the start and end of daylight saving time.

- **Year**
  Enter the year for the daylight saving time changeover.
- **Start Date**
  Enter the following values for the start of daylight saving time:
  - Day
  - Hour
  - Month
- **End Date**
  Enter the following values for the end of daylight saving time:
  - Day
  - Hour
  - Month

Settings with "Recurring" selected

You can create a rule for the daylight saving time changeover.

- **Start Date**
  Enter the following values for the start of daylight saving time:
  - Hour
  - Month
  - Week
    You can select the 1st to 5th or the last week of the month.
  - Weekday
- **End Date**
  Enter the following values for the end of daylight saving time:
  - Hour
  - Month
  - Week
    You can select the 1st to 5th or the last week of the month.
  - Weekday

SNTP client

SNTP (Simple Network Time Protocol) is used for synchronizing the time in the network. The appropriate frames are sent by an SNTP server in the network.

Settings

- **SNTP Client**
  Enable or disable automatic time-of-day synchronization using SNTP.
- **Current System Time** (only available online)
  Shows the values currently set in the system for date and time.
- **Last Synchronization Time** (only available online)
  This box is read-only and shows when the last time-of-day synchronization took place. If no time-of-day synchronization was possible, the box displays "Date/time not set".

- **Last Synchronization Mechanism** (only available online)
  This box displays how the last time-of-day synchronization was performed.
  - Not set
    The time was not set.
  - Manual
    Manual time setting
  - SNTP
    Automatic time-of-day synchronization using SNTP
  - NTP
    Automatic time-of-day synchronization using NTP
  - SIMATIC
    Automatic time-of-day synchronization using the SIMATIC time frame
  - PTP
    Automatic time-of-day synchronization with PTP
- **Time Zone**
  Enter the time zone you are using in the format "+/- HH:MM". The time zone relates to UTC standard world time. Settings for daylight-saving and standard time are taken into account in this box by specifying the time offset.
- **SNTP Mode**
  Select the synchronization mode. The following types of synchronization are possible:
  - Poll
    If you select this protocol type, the input boxes "SNTP Server IP Address", "SNTP Server Port" and "Poll Interval" are displayed for further configuration. With this type of synchronization, the device is active and sends a time query to the SNTP server.
  - Listen
    With this type of synchronization, the device is passive and "listens" for SNTP frames that deliver the time of day.
- **SNTP Server IP Address**
  Enter the IP address of the SNTP server.
- **SNTP Server Port**
  Enter the port of the SNTP server.
- **Poll Interval [s]**
  Enter the interval in seconds between two time polls.

NTP client

If you require time-of-day synchronization using NTP, you can make the relevant settings here.

Settings

- **NTP Client**
  Enable or disable automatic time-of-day synchronization using NTP.
- **Current System Time** (only available online)
  Shows the values currently set in the system for date and time.

- **Last Synchronization Time** (only available online)
  This box is read-only and shows when the last time-of-day synchronization took place. If no time-of-day synchronization was possible, the box displays "Date/time not set".
- **Last Synchronization Mechanism** (only available online)
  This box displays how the last time-of-day synchronization was performed.
  - Not set
    The time was not set.
  - Manual
    Manual time setting
  - SNTP
    Automatic time-of-day synchronization using SNTP
  - NTP
    Automatic time-of-day synchronization using NTP
  - SIMATIC
    Automatic time-of-day synchronization using the SIMATIC time frame
  - PTP
    Automatic time-of-day synchronization with PTP
- **Time Zone**
  Enter the time zone you are using in the format "+/- HH:MM". The time zone relates to UTC standard world time. Settings for daylight-saving and standard time are taken into account in this box by specifying the time offset.
- **NTP Server IP Address**
  Enter the IP address of the NTP server.
- **NTP Server Port**
  Enter the port of the NTP server.
- **Poll Interval [s]**
  Enter the interval in seconds between two time polls.

SIMATIC Time Client

On this page, you configure time synchronization with the SIMATIC Time Client.

Settings

- **SIMATIC Time Client**
  Enable or disable loop the SIMATIC Time Client.
- **Current System Time** (only available online)
  Shows the values currently set in the system for date and time.

- **Last Synchronization Time** (only available online)
  This box is read-only and shows when the last time-of-day synchronization took place. If no time-of-day synchronization was possible, the box displays "Date/time not set".
- **Last Synchronization Mechanism** (only available online)
  This box displays how the last time-of-day synchronization was performed.
  - Not set
    The time was not set.
  - Manual
    Manual time setting
  - SNTP
    Automatic time-of-day synchronization using SNTP
  - NTP
    Automatic time-of-day synchronization using NTP
  - SIMATIC
    Automatic time-of-day synchronization using the SIMATIC time frame
  - PTP
    Automatic time-of-day synchronization with PTP

PTP Client

On this page, you configure time synchronization with PTP (Precision Time Protocol).

This function is only available for SCALANCE X500.

Settings

- **PTP Client**
  Enable or disable time synchronization using PTP. You can configure other settings in "Layer 2 > PTP".
- **Current System Time** (only available online)
  This box is read-only and displays the current system time.
- **Last Synchronization Time** (only available online)
  This box is read-only and shows when the last time-of-day synchronization took place. If no time-of-day synchronization was possible, the box displays "Date/time not set".

- **Last Synchronization Mechanism** (only available online)
  This box displays how the last time-of-day synchronization was performed.
  - Not set
    The time was not set.
  - Manual
    Manual time setting
  - SNTP
    Automatic time-of-day synchronization using SNTP
  - NTP
    Automatic time-of-day synchronization using NTP
  - SIMATIC
    Automatic time-of-day synchronization using the SIMATIC time frame
  - PTP
    Automatic time-of-day synchronization with PTP
- **Time Zone**
  Enter the time zone you are using in the format "+/- HH:MM". The time zone relates to UTC standard world time.

Auto logout

On this page, set the times after which there is an automatic logout from WBM or the CLI following user in activity.

Note

No automatic logout from the CLI

If the connection is not terminated after the set time, check the setting of the "keepalive" mechanism on the Telnet client.

If the interval is shorter than the configured time, the connection is kept alive although no user data is transferred. Example: you selected 300 seconds for the automatic logout and 120 seconds is set for the keepalive function. In this case, a packet is sent every 120 seconds that keeps the connection up.

- Disable the keepalive mechanism (interval time = 0)
  or
- Set the interval high enough so that the underlying connection is terminated when there is inactivity.

Settings

- **Web Based Management [s]**
  Enter the time in seconds for the automatic logout from WBM. If you enter the value 0, the automatic logout is disabled.
- **CLI (TELNET, SSH, Serial) [s]**
  Enter the time in seconds for the automatic logout from the CLI. If you enter the value 0, the automatic logout is disabled.

Button

The "Select/Set" button has the following functions:

- Change display mode
- Enable redundancy manager
- Restore to factory defaults
- Fault mask and LED display

You will find a detailed description of the individual functions available with the buttons in the operating instructions of the specific device.

Settings

- Restore Factory Defaults
  Enables or disables the "Restore Factory Defaults" function for the Select/Set button.

  CAUTION

  Button function "Restore Factory Defaults" active during startup

  If you have disabled this function in your configuration, disabling is only valid during operation. When restarting, for example after power down, the function is active until the configuration is loaded so that the device can inadvertently be reset to the factory settings. This may cause unwanted disruption in network operation since the device needs to be reconfigured if this occurs. An inserted PLUG is also deleted and returned to the status as shipped

- Redundancy Manager
  Enables or disables the redundancy manager function for the Select/Set button.
- Set Fault Mask
  Enable or disable the function "Define fault mask via the LED display" with the Select/set button. This function only works in display mode D.

Syslog client

Syslog according to RFC 3164 is used for transferring short, unencrypted text messages over UDP in the IP network. This requires a system log server.

Requirements for sending log entries:

- The system log function is enabled on the device.
- The system log function is enabled for the relevant event.
- There is a system log server in your network that receives the log entries. Since this is a UDP connection, there is no acknowledgment to the sender.
- The IP address of the system log server is entered on the device.

Settings

- Syslog Client
  Enable or disable the system log function.
- Syslog Server IP Address
  Enter the IP address of the system log server.

This table contains the following columns

- Syslog Server IP Address
  Shows the IP address of the system log server.
- Server Port
  Enter the port of the Syslog server being used.

Ports

Port overview

The page shows the configuration for the data transfer for all ports of the device.

Settings

- Port
  Shows the configurable ports.
- Port name
  Shows the name of the port.
- MAC Address (only available online)
  Shows the MAC address of the port.
- Mode (only available online)
  Shows the transfer parameters of the port
- Negotiation
  Shows whether the automatic configuration is enabled or disabled.
- Flow Ctrl. Type
  Shows whether flow control is enabled or disabled for the port.
- Flow Ctrl.
  Shows whether or not flow control is working on this port.
- MTU
  Shows the maximum packet size.
- Port Type (only with routing)
  Shows the type of the port. The following types are possible:
  - Router Port
  - Switch Port VLAN Hybrid
  - Switch-Port VLAN Trunk
- Status
  Shows whether the port is on or off. Data traffic is possible only over an enabled port.

- **Combo Port Media Type**(SCALANCE XM400 only)
  Shows the mode of the combo port:
  - auto
  - rj45
  - sfp
- **Link** (only available online)
  Shows the connection status to the network. With the connection status, the following is possible:
  - Up
    The port has a valid link to the network, a link integrity signal is being received.
  - Down
    The link is down, for example because the connected device is turned off.

Configuration

On this page, you configure the ports of the device

Settings

- **Port**
  Select the port to be configured. The port is made up of the port number and the slot number, for example port 0.1 is slot 0, port 1.
- **Status**
  Specify whether the port is enabled or disabled.
  - enabled
    The port is enabled. Data traffic is possible only over an enabled port.
  - disabled
    The port is disabled but the connection remains.
  - Link down
    The port is disabled and the connection to the partner device is terminated.
- **Port name**
  Enter a name for the port.
- **MAC Address** (only available online)
  Shows the MAC address of the port.
- **Mode Type**
  Shows the transmission speed and the transmission method of the port. You make the settings for "Autonegotiation" and "Transmission rate" in the port options.

  Note

  Before the port and partner port can communicate with each other, the settings must match at both ends.

- **Mode** (only available online)
  Shows which transmission speed and which transmission mode is currently being used.

- **Negotiation**
  Shows whether the automatic configuration of the connection to the partner port is enabled or disabled.
- **Flow Ctrl. Type**
  Shows whether or not flow control is working on this port.

  Note
  Turning flow control on/off with autonegotiation
  Flow control can only be enabled or disabled if the "autonegotiation" function is turned off. The function cannot enabled again afterwards.

- **Flow Ctrl.**
  Shows whether or not flow control is working on this port.
- **MTU**
  Enter the packet size.
- **Port Type** (only with routing)
  Select the type of the port:
  - Router Port
    The port is an IP interface. It does not support layer 2 functions.
  - Switch Port VLAN Hybrid
    The port sends tagged and untagged frames. It is not automatically a member of a VLAN.
  - Switch-Port VLAN Trunk
    The port only sends tagged frames and is automatically a member of all VLANs.
- **Combo Port Media Type**(SCALANCE XM400 only)
  Specify the mode for the combo port:
  - auto
    If you select this mode, the SFP transceiver port has priority. As soon as an SFP transceiver is plugged in, an existing connection at the fixed RJ-45 port is terminated. If no SFC transceiver is plugged in, a connection can be established via the fixed RJ-45 port.
  - rj45
    If you select this mode, the fixed RJ-45 port is used regardless of the SFP transceiver port.
  - sfp
    If you select this mode, the SFP transceiver port is used regardless of the built-in RJ-45 port.

  The factory setting is auto mode.
- **Link** (only available online)
  Shows the connection status to the network. With the connection status, the following is possible:
  - Up
    The port has a valid link to the network, a link integrity signal is being received.
  - Down
    The link is down, for example because the connected device is turned off.

Changing the port configuration

Note

Optical ports only work with the full duplex mode and at maximum transmission rate. As a result, the following settings cannot be made for optical ports:

- Automatic configuration
- Transmission speed
- Transmission technique

Note

With various automatic functions, the device prevents or reduces the effect on other ports and priority classes (Class of Service) if a port is overloaded. This can mean that frames are discarded even when flow control is enabled.

Port overload occurs when the device receives more frames than it can send, for example as the result of different transmission speeds.

Fault monitoring

Power supply

Configure whether or not the power supply should be monitored by the messaging system. Depending on the hardware variant, there are one or two power connectors (Supply 1 / Supply 2). With a redundant power supply, configure the monitoring separately for each individual feed-in line.

If there is no power on one of the monitored lines (Supply 1 or Supply 2) or when the voltage is too low, a fault is signaled by the signaling system.

Note

You will find the permitted operating voltage limits in the compact operating instructions of the device.

An error causes the fault LED to light up on the device. Depending on the configuration, the error may trigger a trap, an e-mail or an entry in the event log table.

Settings

- Line 1
  Enable or disable the monitoring of power connector 1.
- Line 2
  Enable or disable the monitoring of power connector 2.

Link Change

On this page, you configure whether or not an error message is triggered if there is a status change on a network connection.

If connection monitoring is enabled, an error is signaled

- when there should be a link on a port and this is missing.
- or when there should not be a link on a port and a link is detected.

An error causes the signaling contact to trigger and the fault LED to light up on the device. Depending on the configuration, the error may trigger a trap, an e-mail or an entry in the event log table.

Settings

Table 1 has the following columns:

- **1st column**
  Shows that the settings are valid for all ports.
- **Setting**
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns

- **Port**
  Shows the available ports.
- **Setting**
  Select the setting. You have the following options:
  - Up
    Error handling is triggered when the port changes to the active status.
    (From "Link down" to "Link up")
  - Down
    Error handling is triggered when the port changes to the inactive status.
    (From "Link up" to "Link down")
  - "-" (disabled)
    The error handling is not triggered.

Redundancy

On this page, you configure whether or not an error message is triggered if there is a status change on a network connection.

Setting

- **Redundancy loss (HRP only)**
  Enable or disable connection monitoring. If the redundancy of the connection is lost, an error is signaled.

PNIO

On this page, you configure the device response to PROFINET input and output.

Description

The page contains the following boxes:

- **PNIO runtime mode**
  Shows the status of the PNIO operation.
- **PNIO runtime mode for next startup**
  Select the runtime mode that will be active the next time the device starts up.
- **PNIO AR status**
  This box shows the PROFINET IO application relation status; in other words, whether or not the IE switch is connected "online" or "offline" with a PROFINET controller.
  In this context, online means that a connection to a PROFINET IO controller exists, that the controller has downloaded its configuration data to the IE switch and that the device can send status data to the PROFINET IO controller. In this status known as "in data exchange", the parameters set with the PROFINET IO controller cannot be configured on the IE switch.
- **PNIO Device Name**
  Enter the PROFINET IO device name.
- **Allow PNIO Data Exchange**
  Enable or disable PNIO data exchange
- **Simulate PNIO Data Exchange**
  Enable or disable the simulation of PNIO data exchange.

PLUG

PLUG configuration

Note

The page is only available if there is an online connection to the device.

NOTICE
Do not remove or insert a C-PLUG / KEY-PLUG during operation! A PLUG may only be removed or inserted when the device is turned off. The device checks whether or not a PLUG is present at one second intervals. If it is detected that the PLUG was removed, there is a restart. If a valid KEY-PLUG was inserted in the device, the device changes to a defined error state following the restart.

Information about the configuration of the C-PLUG / KEY-PLUG

This page provides detailed information about the configuration stored on the C-PLUG or KEY-PLUG. It is also possible to reset the PLUG to "factory defaults" or to load it with new contents.

Note

The action is only executed after you click the "Set Values" button.

The action cannot be undone.

If you decide against executing the function after making your selection, click the "Refresh" button. As a result the data of this page is read from the device again and the selection is canceled.

Note

Incompatibility with previous versions with PLUG inserted

During the installation of a previous version, the configuration data can be lost. In this case, the device starts up with the factory settings after the firmware has been installed. In this situation, if a PLUG is inserted in the device, following the restart, this has the status "NOT ACCEPTED" since the PLUG still has the configuration data of the previous more up-to-date firmware. This allows you to return to the previous, more up-to-date firmware without any loss of configuration data.

If the original configuration on the PLUG is no longer required, the PLUG can be deleted or rewritten manually using "System > PLUG".

Settings

The table has the following rows:

- **Status**
  Shows the status of the PLUG. The following are possible:
  - ACCEPTED
    There is a PLUG with a valid and suitable configuration in the device.
  - NOT ACCEPTED
    Invalid or incompatible configuration on the inserted PLUG.
  - NOT PRESENT
    There is no C-PLUG or KEY-PLUG inserted in the device.
  - FACTORY
    PLUG is inserted and does not contain a configuration. This status is also displayed when the PLUG was formatted during operation.
  - MISSING
    There is no PLUG inserted. Functions are configured on the device for which a license is required.
- **Device Family**
  Shows the SIMATIC NET product line that used the C-PLUG or KEY-PLUG previously.
- **Device Type**
  Shows the device type within the product line that used the C-PLUG or KEY-PLUG previously.
- **Configuration Revision**
  The version of the configuration structure. This information relates to the configuration options supported by the device and has nothing to do with the concrete hardware configuration. This revision information does not therefore change if you add or remove additional components (modules or extenders), it can, however, change if you update the firmware.
- **File System**
  Displays the type of file system on the PLUG.

NOTICE
New file system UBI As of firmware version 3.0, UBI is the standard file system for the C-PLUG or KEY-PLUG. If a C-PLUG with the previous file system IECP is detected in such a device, this C-PLUG will be formatted for the UBI file system and the data will be rewritten to the C-PLUG. The file system is also changed following a firmware update to V3.0. A downgrade to the previous version of the corresponding software is then a problem. The firmware can neither read nor write the C-PLUG or KEY-PLUG and it is not even possible to "Erase PLUG to factory default".

- **File System Size [bytes]**
  Shows the maximum storage capacity of the file system on the C-PLUG.
- **File System Usage**
  Displays the storage space in use in the file system of the C-PLUG.

- Info String
  Shows additional information about the device that used the PLUG previously, for example, order number, type designation, and the versions of the hardware and software. The displayed software version corresponds to the version in which the configuration was last changed. With the "NOT ACCEPTED" status, further information on the cause of the problem is displayed.
- Modify PLUG
  Select the required setting.
  - Write current configuration to PLUG
    This option is available only if the status of the PLUG is "NOT ACCEPTED" or FACTORY. The configuration in the internal flash memory of the device is copied to the PLUG.
  - Erase PLUG to factory default
    Deletes all data from the C-PLUG and triggers low-level formatting.

PLUG license

Note

The page is only available if there is an online connection to the device.

NOTICE

Do not remove or insert a C-PLUG / KEY-PLUG during operation!

A PLUG may only be removed or inserted when the device is turned off.
The device checks whether or not a PLUG is present at one second intervals. If it is detected that the PLUG was removed, there is a restart. If a valid KEY-PLUG was inserted in the device, the device changes to a defined error state following the restart.

If the device was configured at some time with a PLUG, the device can no longer be used without this PLUG. To be able to use the device again, reset the device to the factory settings.

Note

Incompatibility with previous versions with PLUG inserted

During the installation of a previous version, the configuration data can be lost. In this case, the device starts up with the factory settings after the firmware has been installed. In this situation, if a PLUG is inserted in the device, following the restart, this has the status "NOT ACCEPTED" since the PLUG still has the configuration data of the previous more up-to-date firmware. This allows you to return to the previous, more up-to-date firmware without any loss of configuration data.

If the original configuration on the PLUG is no longer required, the PLUG can be deleted or rewritten manually using "System > PLUG".

Information about the license of the KEY-PLUG

A C-PLUG can only store the configuration of a device. In addition to the configuration, a KEY-PLUG also contains a license that enables certain functions of your SIMATIC NET device.

This page provides detailed information about the license on the KEY-PLUG. In this example, the KEY-PLUG contains the data for enabling the layer 3 functions of the device.

Displayed values

- **Status**
  Shows the status of the KEY-PLUG. The following are possible:
  - ACCEPTED
    The KEY-PLUG in the device contains a suitable and valid license.
  - NOT ACCEPTED
    The license of the inserted KEY-PLUG is not valid.
  - NOT PRESENT
    No KEY-PLUG is inserted in the device.
  - MISSING
    There is no KEY-PLUG or a C-PLUG with the status "FACTORY" inserted in the device. Functions are configured on the device for which a license is required.
  - WRONG
    The inserted KEY-PLUG is not suitable for the device.
  - UNKNOWN
    Unknown content of the KEY-PLUG.
  - DEFECTIVE
    The content of the KEY-PLUG contains errors.
- **Order ID**
  Shows the order number of the KEY-PLUG. The KEY-PLUG is available for various functional enhancements and for various target systems.
- **Serial Number**
  Shows the serial number of the KEY-PLUG.
- **Info String**
  Shows additional information about the device that used the KEY-PLUG previously, for example, order number, type designation, and the versions of the hardware and software. The displayed software version corresponds to the version in which the configuration was last changed. With the "NOT ACCEPTED" status, further information on the cause of the problem is displayed.

Note

When you save the configuration, the information about whether or not a KEY-PLUG was inserted in the device at the time is also saved. This configuration can then only work if a KEY-PLUG with the same order number / license is inserted.

Ping

Reachability of an address in an IP network

Note

The page is only available if there is an online connection to the device.

With the ping function, you can check whether a certain IP address is reachable in the network.

Settings

The table has the following columns:

- **IP Address**
  Enter the IP address of the device.
- **Repeat**
  Enter the number of ping requests.
- **Ping**
  Click this button to start the ping function.
- **Ping Output**
  This box shows the output of the ping function.

PoE

General

On this page, you specify the maximum power for the power sourcing equipment (PSE).

PSE (power sourcing equipment)

The SCALANCE X-500 represents a PSE (Power Sourcing Equipment).

With a SCALANCE XM400, you can use the "Power over Ethernet" function via the port extender PE408PoE. Each group of four ports with PoE capability is known as a PSE. The numbering of the PSEs is fixed and does not change with the number of plugged-in PoE port extenders or the slot. A maximum of 4 PSEs are possible.

Setting

- **PSE**
  Shows the number of the PoE power supply.
- **Maximum Power**
  Maximum power that a PSE provides to supply PoE devices.
- **Allocated Power**
  Sum of the power reserved by the PoE devices according to the "Classification".

- Used power
  Sum of the power used by the end devices.
- Usage Threshold [%]
  When the power being used by the end devices exceeds the percentage shown here, an event is triggered.

Port

Settings for the ports

For each individual PoE port, you can specify whether or not the power will be supplied via Ethernet. You can also set a priority for each connected powered device (PD). Devices for which a high priority was set, take preference over other devices for the power supply.

Settings

The table has the following columns:

- Port
  Shows the configurable PoE ports.
  The port is made up of the port number and the slot number, for example port 0.1 is slot 0, port 1.
- Setting
  Enable the PoE power supply for this port or interrupt it.
- Priority
  Select which priority this port will have for the power supply.
  The following settings are possible, in ascending order of relevance:
  - low
    low priority
  - high
    medium priority
  - critical
    high priority

  If the same priority is set for two ports, the port with the lower port number will be preferred when necessary.
- Type
  Here, you can enter a string to describe the connected device in greater detail.
- Classification
  The classification specifies the class of the device. From this, it is possible to recognize the maximum power of the device.

- **Status**
  Shows the current status of the port.
  The following states are possible:
  - disabled
    The PoE power supply is deactivated for this port.
  - delivering
    The PoE power supply is activated for this port and a device is connected.
  - searching
    The PoE power supply is activated for this port but there is no device connected.

  Note

  If a device is connected to a port with PoE capability, a check is made to determine whether the power of the port is adequate for the connected device. If the power of the port is inadequate, although PoE is enabled in "Setting", the port nevertheless has the status "disabled". This means that the port was disabled by the PoE power management.

- **Power [mW]**
  Shows the power that the SCALANCE provides at this port.
- **Voltage [V]**
  Shows the voltage applied to this port.
- **Current [mA]**
  Shows the current with which a device connected to this port is supplied.

Port diagnostics

Cable tester

Note

The page is only available if there is an online connection to the device.

With this page, each individual Ethernet port can run independent fault diagnostics on the cable. This test is performed without needing to remove the cable, connect a cable tester and install a loopback module at the other end. Short-circuits and cable breaks can be localized to within a few meters.

Note

Please note that this test is permitted only when no data connection is established on the port to be tested.

Settings

- **Port**
  Select the required port from the drop-down list.
- **Run Test**
  Activates error diagnostics. The result is shown in the table.

This table contains the following columns:

- **Pair**
  Shows the wire pair in the cable.

  Note
  Wire pairs

  Wire pairs 4-5 and 7-8 of 10/100 Mbps network cables are not used.

  In 1000 Mbps or gigabit Ethernet, all 4 wire pairs are used.

  The wire pair assignment - pin assignment is as follows (DIN 50173):

  Pair 1 = pin 4-5

  Pair 2 = pin 1-2

  Pair 3 = pin 3-6

  Pair 4 = pin 7-8

- **Status**
  Displays the status of the cable.
- **Distance [m]**
  Displays the distance to the cable end, cable break, or short-circuit.

SFP diagnostics

On this page, you run independent error diagnostics for each individual SFP port. This test is performed without needing to remove the cable, connect a cable tester or install a loopback module at the other end.

Note

Please note that this test is permitted only when no data connection is established on the port to be tested.

If, however, there is a data connection to the port to be tested, this is briefly interrupted.

Automatic re-establishment of the connection can fail and then needs to be done manually.

Description

The page contains the following boxes:

- Port
  Select the required port from the drop-down list.

The values are shown in the following boxes:

- Name
  Shows the name of the interface.
- Model
  Shows the type of interface.
- Revision
  Shows the hardware version of the SFP.
- Serial
  Shows the serial number of the SFP.
- Nominal Bit Rate [Mbps]
  Shows the nominal bit rate of the interface.
- Max. Link (50.0/125um) [m]
  Shows the maximum distance in meters that is possible with this medium.
- Max. Link (62.5/125um) [m]
  Shows the maximum distance in meters that is possible with this medium.

The following table shows the values of the SFP transceiver used in this port:

- Temperature [°C]
  Shows the temperature of the interface.
- Voltage [V]
  Shows the voltage applied to the interface [V].
- Current [mA]
  Shows the current consumption of the interface [mA].
- RX Power [mW]
  Shows the receive power of the interface [mW].
- Tx Power [mW]
  Shows the transmit power of the interface [mW].
- Current
  Shows the current value.
- Low
  Shows the lowest value.
- High
  Shows the highest value.

Layer 2

Configuration

The functions of layer 2 are configured on this page. With some functions, there are further configuration pages on which more detailed settings can be made. You can also check the settings on the configuration pages.

Settings

- Protocol Based VLAN
  Enable or disable protocol-based VLAN. You can configure other settings in "Layer 2 > VLAN".
- Subnet Based VLAN
  Enable or disable subnet-based VLAN. You can configure other settings in "Layer 2 > VLAN".
- Dynamic MAC Aging
  Enable or disable the "Aging" mechanism. You can configure other settings in "Layer 2 > Dynamic MAC Aging".
- Redundancy Type
  The following settings are available:
  - "-" (disabled)
    The redundancy function is disabled.
  - Ring
    Enables ring redundancy. You can configure other settings in "Layer 2 > Ring Redundancy > Ring".
  - Spanning Tree
    If you select this option, you specify the required redundancy mode in "Redundancy Mode".

- **Redundancy Mode**
  If you select "Spanning Tree" for the "Redundancy Type", the following options are then available:
  - **STP**
    Enables the Spanning Tree Protocol (STP). Typical reconfiguration times with spanning tree are between 20 and 30 seconds. You can configure other settings in "Layer 2 > Spanning Tree".
  - **RSTP**
    Enables the Rapid Spanning Tree Protocol (RSTP). If a spanning tree frame is detected at a port, this port reverts from RSTP to spanning tree. You can configure other settings in "Layer 2 > Spanning Tree".

    Note

    When using RSTP (Rapid Spanning Tree Protocol), loops involving duplication of frames or frames being overtaken may occur briefly. If this is not acceptable in your particular application, use the slower standard spanning tree mechanism.

  - **MSTP**
    Enables the Multiple Spanning Tree Protocol (MSTP). You can configure other settings in "Layer 2 > Spanning Tree".

  If you select "Ring" for the "Redundancy Type", the following options are then available:
  - **Automatic Redundancy Detection**
    Select this setting to create an automatic configuration of the redundancy mode. In this mode, the device automatically detects whether or not there is a device with the "HRP Manager" role in the ring. If this is the case, the device adopts the role of "HRP Client". If no HRP manager is found, all devices with the "Automatic Redundancy Detection" or "MRP Auto Manager" setting negotiate among themselves to establish which device adopts the role of "MRP Manager". The device with the lowest MAC address will always become "MRP Manager". The other devices automatically set themselves to redundancy type "MRP Client".
  - **MRP Auto-Manager**
    Automatic media redundancy manager
  - **MRP Client**
    Media redundancy client
  - **HRP Client**
    High Speed Redundancy Protocol client
  - **HRP Manager**
    High Speed Redundancy Protocol manager
- **Standby**
  Enable or disable the "Standby redundancy" function. You can configure other settings in "Layer 2 > Ring Redundancy > Standby".
- **Passive Listening**
  Enable or disable the "passive listening" function.

- RMON
  If you select this check box, Remote Monitoring (RMON) allows diagnostics data to be collected on the device, prepared and read out using SNMP by a network management station that also supports RMON. This diagnostic data, for example port-related load trends, allow problems in the network to be detected early and eliminated.
- Dynamic Multicast
  The following settings are possible:
  - "-" (disabled)
  - IGMP Snooping
    Enables IGMP (Internet Group Management Protocol). You can configure other settings in "Layer 2 > Multicast > IGMP".
  - GMRP
    Enables GMRP (GARP Multicast Registration Protocol). You can configure other settings in "Layer 2 > Multicast > GMRP".

    Note

    GMRP and IGMP cannot operate at the same time.
- GVRP
  Enable or disable "GVRP" (GARP VLAN Registration Protocol). You can configure other settings in "Layer 2 > VLAN > GVRP".
- Mirroring
  Enable or disable port mirroring. You can configure other settings in "Layer 2 > Mirroring > Port".
- Loop Detection
  Enable or disable loop detection. You can configure other settings in "Layer 2 > Loop Detection".
- PTP
  Specify how the device will process PTP messages. You can configure other settings in "Layer 2 > PTP".
  - Off
    The device does not process any PTP messages. PTP messages are, however, forwarded according to the rules of the switch.
  - transparent
    The device adopts the function of a transparent clock and forwards PTP messages to other nodes while at the same time making entries in the correction field of the PTP message.

QoS

CoS map

On this page, CoS priorities are assigned to certain queues (traffic queues).

Settings

- CoS
  Shows the CoS priority of the incoming packets.
- Queue
  Select the forwarding queue (send priority) that is assigned the CoS priority.
  The higher the number of the queue, the higher the send priority.

DSCP map

On this page, DSCP settings are assigned to various queues (traffic queues).

Settings

- DSCP
  Shows the DSCP priority of the incoming packets.
- Queue
  Select the forwarding queue (send priority) that is assigned the DSCP value. The higher the queue number, the higher the send priority.

Load limitation

Limiting the transfer rate of incoming and outgoing data

On this page, you configure the load limitation (maximum number of data packets per second) for the individual ports. You can specify the category of frame for which these limit values will apply.

Settings

Table 1 has the following columns:

- 1st column
  Shows that the settings are valid for all ports.
- Limit Ingress Unicast (DLF) / Limit Ingress Broadcast / Limit Ingress Multicast
  Select the required setting.
  - enabled: Enables the function.
  - disabled: Disables the function
  - No Change: The setting in table 2 remains unchanged
- Total Ingress Rate [pkts/s]
  Specify the maximum number of incoming packets processed by the device. If "No Change" is entered, the entry in table 2 remains unchanged.

- **Egress Rate [Kb/s]**
  Specify the data rate for all outgoing frames. If "No Change" is entered, the entry in table 2 remains unchanged
- **Transfer to table**
  If you click the button, the settings are adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows the port to which the settings relate.
- **Limit Ingress Unicast (DLF)**
  Enable or disable the data rate for limiting incoming unicast frames with an unresolvable address (Destination lookup failure).
- **Limit Ingress Broadcast**
  Enable or disable the data rate for limiting incoming broadcast frames.
- **Limit Ingress Multicast**
  Enable or disable the data rate for limiting incoming multicast frames.
- **Total Ingress Rate [pkts/s]**
  Specify the maximum number of incoming packets processed by the device.
- **Egress Rate [Kb/s]**
- Specify the data rate for all outgoing frames.

Note

Rounding of the values, deviation from desired value

When you enter the values for transmission rates, note that the WBM rounds to correct values.

If values are configured for total ingress transmission rate and egress transmission rate, the actual values in operation can exceed or fall below the set values by 10%.

VLAN

General

On this page, you define the VLAN and specify the use of the ports.

Note

Changing the "Agent VLAN ID"

If the configuration PC is connected directly to the device via Ethernet and you change the "agent VLAN ID", the device is no longer reachable via Ethernet following the change.

Important rules for VLANs

Make sure you keep to the following rules when configuring and operating your VLANs:

- Frames with the VLAN ID "0" are handled as untagged frames but retain their priority value.
- As default, all ports on the device send frames without a VLAN tag to ensure that the end node can receive these frames.
- With SCALANCE X devices, the VLAN ID "1" is the default on all ports.
- If an end node is connected to a port, outgoing frames should be sent without a tag (static access port). If, however, there is a further switch at this port, the frame should have a tag added (trunk port).
- With a trunk port, the VLAN assignment is dynamic. Static configurations can only be created if, in addition to the trunk port property, the port is also entered statically as a member in the VLANs involved. An example of a static configuration is the assignment of the multicast groups in certain VLANs.

Settings

- VLAN ID
  Enter the VLAN ID.

The table has the following columns:

- VLAN ID
  Shows the VLAN ID. The VLAN ID is assigned once when you create a new data record and can then no longer be changed. To make a change, the entire data record must be deleted and created again.
- Name
  Enter a name for the VLAN. The name only provides information and has no effect on the configuration.
  The VLAN name can be a maximum of 32 characters long.

- Status
  Shows the status of the entry. Here, static means that the address was entered as a static address by the user. The entry GVRP means that the configuration was registered by a GVRP frame. This is, however, only possible if GVRP was enabled for the device.
- List of ports
  Specify the use of the port. The following options are available:
  - "-"
  - The port is not a member of the specified VLAN.
    With a new definition, all ports have the identifier "-".
  - M
    The port is a member of the VLAN. Frames sent in this VLAN are forwarded with the corresponding VLAN tag.
  - R
    The port is a member of the VLAN. A GVRP frame is used for the registration.
  - U (upper case)
    The port is an untagged member of the VLAN. Frames sent in this VLAN are forwarded without the VLAN tag. Frames without a VLAN tag are sent from this port.
  - u (lower case)
    The port is an untagged member of the VLAN, but the VLAN is not configured as a port VLAN. Frames sent in this VLAN are forwarded without the VLAN tag.
  - F
    The port is not a member of the specified VLAN and it is not possible for the VLAN to be registered dynamically at this port using GVRP. You can configure other settings in "Layer 2 > VLAN > Port-based VLAN".

GVRP

Configuration of GVRP functionality

Using GVRP frame, a different device can register at the port of the device for a specific VID. A different device, can, for example be an end device or a switch. The device can also send GVRP frames via this port.

Settings

- GVRP
  Enable or disable the "GVRP" function.

Table 1 has the following columns:

- **1st column**
  Shows that the settings are valid for all ports of table 2.
- **Setting**
  Select the setting. You have the following setting options:
  - enabled
    Enables the sending of GVRP frames.
  - disabled
    Disables the sending of GVRP frames.
  - No Change
    No change in table 2.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2. If you have selected "No Change", the content of table 2 will not be changed.

Table 2 has the following columns:

- **Port**
  Shows the available ports.
- **Setting**
  Enable or disable the sending GVRP frames.

Port-based VLAN

Processing received frames

On this page, you specify the configuration of the port properties for receiving frames.

Settings

Table 1 has the following columns:

- **1st column**
  Shows that the settings are valid for all ports.
- **Priority / Port VID / Acceptable Frames / Ingress Filtering**
  Select the setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- Port
  Shows the available ports and link aggregations.
- Priority
  Select the priority assigned to untagged frames.
  The CoS priority (Class of Service) used in the VLAN tag. If a frame is received without a tag, it will be assigned this priority. This priority specifies how the frame is further processed compared with other frames.
  There are a total of eight priorities with values 0 to 7, where 7 represents the highest priority (IEEE 802.1p Port Priority).
- Port VID
  Select the VLAN ID. Only VLAN IDs defined on the "VLAN > General" page can be selected. If a received frame does not have a VLAN tag, it has a tag with the VLAN ID specified here added to it and is sent according to the rules at the port.
- Acceptable Frames
  Specify which types of frames will be accepted. The following alternatives are possible:
  - Tagged Frames Only
    The device discards all untagged frames. Otherwise, the forwarding rules apply according to the configuration.
  - All
    The device forwards all frames
- Ingress Filtering
  Specify whether the VID of received frames is evaluated.
  The following options are available:
  - enabled
    The VLAN ID of received frames decides whether they are forwarded: To forward a VLAN tagged frame, the receiving port must be a member in the same VLAN. Frames from unknown VLANs are discarded at the receiving port.
  - disabled
    All frames are forwarded.

Protocol-based VLAN group

On this page, you specify groups to which a protocol will be assigned.

Settings

- Protocol Based VLAN
  Enable or disable the protocol-based VLAN assignment.
- Protocol Value
  Enter the hexadecimal protocol value.
  A few examples are shown below:
  - PROFINET: 88:92
  - IP: 08:00
  - Novell: 81:37
  - netbios: f0:f0
  - appletalk: 80:9b
- Group ID
  Enter the ID of the group.

The table has the following columns:

- Protocol Value
  Shows the file value.
- Group ID
  Shows the group ID.

Protocol-based VLAN port

On this page, you specify which protocol and which VLAN is assigned to the individual port.

Settings

- Port
  Select the required port. All available ports and the link aggregations can be selected.
- Group ID
  Select the group ID from the drop-down list. Specify the ID in "Layer 2 > VLAN > Protocol Based VLAN Group".

The table has the following columns:

- Port
  All available ports and the link aggregations are shown.
- Group ID
  Shows the group ID assigned to the port.
- VLAN ID
  Select the required VLAN ID to be assigned to the port.

IPv4 subnet-based VLAN

On this page, you specify which VLAN ID is assigned to the subnet.

Settings

- Subnet Based VLAN
  Enable or disable the subnet-based VLAN assignment
- Port
  Select the port. All available ports and the link aggregations can be selected.
- Subnet Address
  Enter the IP address of the subnet.
  Example: 192.168.10.0 for the network 192.168.10.x with nodes 192.168.10.1 to 192.168.10.254.
- Subnet Mask
  Enter the subnet mask.

The table has the following columns:

- Port
  All available ports and the link aggregations are shown.
- Subnet Address
  Shows the subnet assigned to the port.
- Subnet Mask
  Shows the subnet mask.
- VLAN ID
  Select the VLAN ID you want to assign to the port or the subnet.

Mirroring

Basics

Mirroring

The device provides the option of simultaneously channeling incoming or outgoing data streams via other interfaces for analysis or monitoring. This has no effect on the monitored data streams. This procedure is known as mirroring. In this menu section, you enable or disable mirroring and set the parameters.

Mirroring ports

Mirroring a port means that the data traffic at a port (mirrored port) of the IE switch is copied to another port (monitor port). You can mirror one or more ports to a monitor port.

If a protocol analyzer is connected to the monitor port, the data traffic at the mirrored port can be recorded without interrupting the connection. This means that the data traffic can be investigated without being affected. This is possible only if a free port is available on the device as the monitor port.

General

On this page, you can enable or disable the mirroring function and make the basic settings.

Note

If the maximum data rate of the mirrored port is higher than that of the monitor port, data may be lost and the monitor port no longer reflects the data traffic at the mirrored port. Several ports can be mirrored to one monitor port at the same time.

Mirroring a port does not work beyond switch core boundaries.

Disable port mirroring if you want to connect a normal end device to the monitor port.

Settings

- **Mirroring**
  Enable or disable mirroring of the data traffic.
- **Monitor Barrier**
  Enable or disable the option to restrict communication via the monitor port.

  Note
  Monitor Barrier

  If you enable the monitor barrier, the data traffic on the destination port is automatically blocked (broadcast, multicast, unicast, DCP forwarding, LLDP) so that only the mirrored traffic is present. To be able to allow other data traffic again, you need to configure this. The previous statuses of these options are not restored after stopping the monitor barrier and must be reconfigured.

  – enabled
    The monitor port is taken out of normal frame switching.
  – disabled
    Communication via the monitor port is unrestricted.

This table has the following columns:

- Session ID
  Enable or disable listening in on incoming packets at the required port.
- Session Type
  Specify which data traffic is mirrored. The following options are available:
  - '-'
    None
  - Port Based
    Port-based mirroring You can configure other settings in "Layer 2 > Mirroring > Port".
  - VLAN
    VLAN-based mirroring. You can configure other settings in "Layer 2 > Mirroring > VLAN".
  - MAC ACL
    Mirroring of the MAC Access Control List. You can configure other settings in "Layer 2 > Mirroring > MAC Flow".
  - IP ACL
    Mirroring of the IP Access Control List. You can configure other settings in "Layer 2 > Mirroring > IP Flow".
- Status
  Shows whether or not mirroring is active.
- Dest. Port
  Select the destination port to be mirrored to in this session.

Port

Mirroring ports

You can only configure the settings on this page if you have already generated a session ID with the session type "Port Based" on the "General" tab.

Settings

- Session ID
  Select the session ID. Only session IDs of the session type "Port Based" are available.

The table has the following columns:

- Port
  Shows the port to be monitored.
- Ingress Mirroring
  Enable or disable mirroring of incoming packets at the required port.
- Egress Mirroring
  Enable or disable mirroring of outgoing packets at the required port.

VLAN

VLAN sources of the port mirroring function

You can only configure the settings on this page if you have already generated a session ID with the session type "VLAN" on the "General" tab.

On this page, you specify the VLAN whose incoming data traffic will be mirrored to the monitor port.

Settings

- Session ID
  Select the session ID. Only session IDs of the session type "VLAN" are available.
- VLAN ID
  Enter a VLAN ID. The VLAN ID can only be assigned once when you create a new data record and can then no longer be changed. To make a change, the entire data record must be deleted and created again.

The table has the following columns:

- VLAN ID
  Shows the VLAN ID.

MAC Flow

You can only configure the settings on this page if you have already generated a session ID with the session type "MAC ACL" on the "General" tab.

The MAC ACL filter decides which data is available at the monitor port.

Settings

- Session ID
  Select the session ID. Only session IDs of the session type "MAC ACL" are available.

The table has the following columns:

- ACL Filter Number
  Shows the number of the ACL filter. You configure the MAC ACL filter in "SecurityPort ACL MAC".
- Ingress Mirroring
  Shows whether incoming packets are mirrored.

Note

Rules

A rule selected for ingress mirroring only becomes active if it was configured as a port ingress rule on at least one port. You configure the port ingress rules in "Security> Port MAC IP > Port Ingress Rules".

- Source MAC
  Shows the MAC address of the sender.
- Dest. MAC
  Shows the MAC address of the recipient.
- Ingress Port
  Shows a list of all ports to which this rule applies.
- Egress Port
  Shows a list of all ports to which this rule applies.

IP Flow

You can only configure the settings on this page if you have already generated a session ID with the session type "IP ACL" on the "General" tab.

The IP ACL filter decides which data is mirrored to the monitor port.

Settings

- Session ID
  Select the session ID. Only session IDs of the session type "IP ACL" are available.

The table has the following columns:

- ACL Filter Number
  Shows the number of the ACL filter. You configure the IP ACL filter in "Security > Port ACL IP".
- Ingress Mirroring
  Shows whether incoming packets are mirrored.

> **Note**
>
> **Rules**
>
> A rule selected for ingress mirroring only becomes active if it was configured as a port ingress rule on at least one port. You configure the port ingress rules in "Security> Port ACL IP > Port Ingress Rules".

- Source IP
  Shows the IP address of the sender.
- Source Subnet Mask
  Shows the subnet mask of the sender.
- Dest. IP
  Shows the IP address of the recipient.
- Dest. Subnet Mask
  Shows the subnet mask of the recipient.
- Ingress Port
  Shows a list of all ingress ports to which this rule applies.
- Egress Port
  Shows a list of all egress ports to which this rule applies.

Dynamic MAC Aging

The device automatically learns the source addresses of the connected nodes.

This information is used to forward data frames to the nodes specifically involved. The network load for the other nodes is reduced.
If a device does not receive a frame whose source address matches a learnt address within a certain time, the learnt address is deleted. This mechanism is known as aging. Aging prevents frames being forwarded incorrectly, for example when an end device (for example a programming device) is connected to a different port. If the check box is not enabled, a device does not delete learned addresses automatically.

Settings

- Dynamic MAC Aging
  Enable or disable the function for automatic aging of learned MAC addresses:
- Aging Time [s]
  Enter the time in seconds. After this time, a learned address is deleted if the device does not receive any further frames from this sender address.

Ring Redundancy

Ring

On this page, you can select the required mode for fast ring redundancy. The "Automatic Redundancy Detection" ring redundancy mode is the default when the device ships.

Note

The ring redundancy cannot be enabled if Spanning Tree is still enabled on the device.

Settings

- **Ring Redundancy**
  Enable or disable ring redundancy.
- **Ring redundancy mode**
  Specify the ring redundancy mode. The following options are available:
  - **"-" (disabled)**
    The redundancy function is disabled.
  - **Automatic Redundancy Detection**
    Select this setting to create an automatic configuration of the redundancy mode. In this mode, the device automatically detects whether or not there is a device with the "HRP Manager" role in the ring. If this is the case, the device adopts the role of "HRP Client". If no HRP manager is found, all devices with the "Automatic Redundancy Detection" or "MRP Auto Manager" setting negotiate among themselves to establish which device adopts the role of "MRP Manager". The device with the lowest MAC address will always become "MRP Manager". The other devices automatically set themselves to redundancy mode "MRP Client".
  - **MRP Auto Manager**
    Devices with the setting "Automatic Redundancy Detection" or "MRP Auto Manager" negotiate among themselves which device will adopt the "MRP Manager" role. The device with the lowest MAC address will always become "MRP Manager". In contrast to the setting "Automatic Redundancy Detection", the devices are not capable of detecting whether or not an HRP manager is in the ring. This means that they never adopt the role of HRP client.
  - **MRP Client**
    In a ring whose devices are configured with MRP, at least one device must be set to the "Automatic Redundancy Detection" or "MRP Auto Manager" mode. You can set the "MRP Client" role for all other devices. If all devices except one are configured as "MRP Client", this one device automatically adopts the role of "MRP Manager".
    Select "MRP Client" mode if you want to operate the device along with components that do not originate from Siemens in the ring.
  - **HRP Client**
    Here, you can select the role "HRP Client".
  - **HRP Manager**
    When you configure an HRP ring, one device must be set as HRP manager. All other devices must be configured as HRP clients.
- **Ring ports**
  Specify the ports to be used as ring ports in media redundancy in ring topologies.
  The ring port you select in the upper drop-down menu is the "Isolated Port" in HRP.

Note

If you restore the factory defaults, the default redundancy mode "Automatic Redundancy Detection" becomes active.

The ring port configuration is also reset to the factory default ports. If other ports were used previously as ring ports, with the appropriate attachment, a previously correctly configured device can cause circulating frames and therefore the failure of the data traffic.

- Observer
  Enable or disable the observer. The "Observer" function is only available in HRP rings. The ring port selected in the upper drop-down menu is connected to the "Isolated Port" of an HRP manager.
  The observer monitors malfunctions of the redundancy manager or incorrect configurations of an HRP ring.
  If the observer is enabled, it can interrupt the connected ring if errors are detected. To do this, the observer switches a ring port to the "blocking" status. When the error is resolved, the observer enables the port again.
- Restart Observer (only available online)
  If numerous errors occur in quick succession, the observer no longer enables its port automatically. The ring port remains permanently in the "blocking" status. This is signaled by the error LED and a message text.
  After the errors have been eliminated, you can enable the port again using the "Restart Observer" button.

Standby

Standby manager

The standby manager allows the redundant linking of two HRP rings. To do this, two neighboring devices within a ring must be configured as standby partners. Enable the standby manager for both standby partners and select the port via which the device is connected to the ring you want to link to. For the "Standby Connection Name", a name unique within the ring must be assigned for both partners. This identifies the two modules as standby partners that belong together.

Note

To be able to use the function, HRP must be activated.

Settings

- **Standby**
  Enable or disable the standby manager.

  Note

  If two devices are linked by the standby function, the "Standby" function must be enabled on both devices.

  Note

  The standby manager always requires an activated HRP client.

- **Standby Connection Name**
  Enter the name for the standby connection. This name defines the master/slave device pair. Both must be located in the same ring. This is achieved by entering the same name on two devices in the ring.

  Note

  Make sure that the standby name (for one pair of devices) is used only once in the network.

- **Force device to standby master**
  When selected, the device is configured as standby master regardless of its MAC address.
  - If the setting is enabled for neither of the two devices for which the standby function is enabled, then assuming that no error has occurred, the device with the higher MAC address adopts the role of standby master.
  - If the setting is enabled for both devices or if the setting is only supported by one device, the standby master is also selected based on the MAC address.

  This type of assignment is important in particular when a device is replaced. Depending on the MAC addresses, the previous device with the slave function can take over the role of the standby master.

This table has the following columns:

- **Port**
  Shows the port to which the setting relates.
- **Setting**
  Here, you specify which ports are standby ports. The standby ports are involved in the redirection of data traffic.
  If there are no problems, only the standby ports of the master are enabled and handle to the data traffic into the connected HRP rings (buses). If the master or the Ethernet connection (link) of one of the standby ports of the master fails, all standby ports of the master will be disabled and the standby ports of the slave enabled. As a result, a functioning Ethernet connection to the connected network segments (HRP rings/linear buses) is restored.

Spanning Tree

General

General settings of MSTP

On this page, you configure the settings for MSTP. As default, Rapid Spanning Tree is enabled that can be set to the MSTP, RSTP or STP compatible mode with a switch.

On the configuration pages of these functions, you can make detailed settings.

Depending on the compatibility mode, you can configure the corresponding function on the relevant configuration page.

Settings

- Spanning Tree
  Enable or disable Spanning Tree.
- Protocol Compatibility
  Select the compatibility mode of Spanning Tree. If, for example, you select RSTP, Spanning Tree behaves like RSTP.
  The following settings are available:
  - STP
  - RSTP
  - MSTP

Using a link aggregation in an MSTP instance

If you want to use a link aggregation in an MSTP instance, follow the steps below during configuration:

1. Create a link aggregation in "Layer 2" > "Link aggregation (Page 1290)".
2. Create an MSTP instance in "Layer 2" > "Spanning Tree" > "MST General (Page 1285)".
3. Configure the link aggregation in "Layer 2" > "Spanning Tree" > "MST port (Page 1286)".

Automatic activation of MRP in redundant topologies

If you connect SCALANCE X switches with redundant network structures in the topology view, MRP is activated automatically on the switches involved.

If there is an existing configuration with other redundancy mechanisms, such as MSTP, this is automatically deactivated.

CIST general

On this page, you configure CIST.

Settings

- **Bridge Priority / Root Priority** (only available online)
  Which device becomes the root bridge is decided based on the bridge priority. The bridge with the highest priority becomes the root bridge. The lower the value, the higher the priority. If several devices in a network have the same priority, the device whose MAC address has the lowest numeric value will become the root bridge. Both parameters, bridge priority and MAC address together form the bridge identifier. Since the root bridge manages all path changes, it should be located as centrally as possible due to the delay of the frames.
- **Bridge Address / Root Address**(only available online)
  The bridge address shows the MAC address of the device and the root address shows the MAC address of the root bridge.
- **Root Port** (only available online)
  Shows the port via which the switch communicates with the root bridge.
- **Root Cost** (only available online)
  The path costs from this device to the root bridge.
  In MSTP mode, path costs up to the root bridge are shown.
- **Topology Changes** / **Last Topology Change** (only available online)
  The entry for the device shows the number of reconfigurations due to the spanning tree mechanism since the last startup. For the root bridge, the time since the last reconfiguration is displayed as follows:
  - Seconds: sec unit after the number
  - Minutes: min unit after the number
  - Hour: hr unit after the number
- **Bridge Hello Time / Root Hello Time** (only available online)
  Each bridge regularly sends configuration frames (BPDUs). The interval between two such frames is the Hello time. The default for this parameter is 2 seconds.
- **Bridge Forward Delay / Root Forward Delay**
  New configuration data is not used immediately by a bridge but only after the period specified in the parameter. This ensures that operation is only started with the new topology after all the bridges have the required information.
- **Bridge Max Age / Root Max Age** (only available online)
  Bridge Max Age defines the maximum "age" of a received BPDU for it to be accepted as valid by the switch.
- **Bridge Max Hop Count**
  This parameter specifies how many MSTP nodes a BPDU may pass through. If an MSTP BPDU is received and has a "Bridge Max Hop Count" that exceeds the value configured here, it is discarded.
- **Regional root priority** (only available online)
  For a description, see Bridge Priority / Root Priority
- **Regional root address** (only available online)
  The MAC address of the device.
- **Regional root costs** (only available online)
  Shows the path costs from this device to the regional root bridge.

- **Region Name**
  Enter the name of the MSTP region to which this device belongs. As default, the MAC address of the device is entered here. This value must be the same on all devices that belong to the same MSTP region.
- **Region Version**
  Enter the version number of the MSTP region in which the device is located. This value must be the same on all devices that belong to the same MSTP region.
- **Reset Counter** (only available online)
  Click this button to reset the counters on this page to zero.

CIST port

MSTP-CIST port configuration

When the page is called, the table displays the current status of the configuration of the port parameters.

To configure them, click the relevant cells in the port table.

Settings

Table 1 has the following columns:

- **1st column**
  Shows that the setting is valid for all ports of table 2.
- **MSTP Status**
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows the available ports.
- **MSTP Status**
  Specify whether or not the port is integrated in the spanning tree.

  Note

  If you disable the "MSTP Status" option for a port, this may cause the formation of loops. The topology must be kept in mind.

- **Priority**
  Enter the priority of the port. The priority is only evaluated when the path costs are the same. The value must be divisible by 16. If the value that cannot be divided by 16, the value is automatically adapted.
- **Cost Calc.**
  Enter the path cost calculation. If you enter the value "0" here, the automatically calculated value is displayed in the "Path Costs" box.

- **Path Costs** (only available online)
  This parameter is used to calculate the path that will be selected. The path with the lowest value is selected as the path. If several ports of a device have the same value for the path costs, the port with the lowest port number is selected.
  If the value for "Cost Calc." is "0", the automatically calculated value is displayed. Otherwise, the value of the "Cost Calc." box is displayed. The calculation of the path costs is largely based on the transmission speed. The higher the achievable transmission speed is, the lower the value of the path costs.
  Typical values for path costs with rapid spanning tree:
  - 10,000 Mbps = 2,000
  - 1000 Mbps = 20,000
  - 100 Mbps = 200,000
  - 10 Mbps = 2,000,000

  The values can, however, also be set individually.
- **Status** (only available online)
  Displays the current status of the port. The values are only displayed and cannot be configured. The "Status" parameter depends on the configured protocol. The following is possible for status:
  - Disabled

    The port only receives and is not involved in STP, MSTP and RSTP.
  - Discarding
    In the "Discarding" mode, BPDU frames are received. Other incoming or outgoing frames are discarded.
  - Listening

    In this status, BPDUs are both received and sent. The port is involved in the spanning tree algorithm.
  - Learning

    Stage prior to the forwarding status, the port is actively learning the topology (in other words, the node addresses).
  - Forwarding

    Following the reconfiguration time, the port is active in the network; it receives and forwards data frames.

- **Fwd. Trans** (only available online)
  Specifies the number of changes from the "Discarding" status to the "Forwarding" status
- **Edge type**
  Specify the type of edge port. You have the following options:
  - "-"
    Edge port is disabled. The port is treated as a "no edge port".
  - Admin
    Select this option when there is always an end device on this port. Otherwise a reconfiguration of the network will be triggered each time a connection is changed.
  - Auto
    Select this option if you want a connected end device to be detected automatically at this port. When the connection is established the first time, the port is treated as a "no Edge Port".
  - Admin/Auto
    Select these options if you operate a combination of both on this port. When the connection is established the first time, the port is treated as an Edge Port.
- **Edge** (only available online)
  Shows the status of the port.
  - Enabled

    An edge port is connected to this port.
  - Disabled
    There is a spanning tree or rapid spanning tree device at this port.

  With an end device, a switch can change over the port faster without taking into account spanning tree frames. If a spanning tree frame is received despite this setting, the port automatically changes to the "Disabled" setting for switches.

- **P.t.P. Type**
  Select the required option. The selection depends on the port that is set.
  - "-"
    Point to point is calculated automatically. If the port is set to half duplex, a point-to-point link is not assumed.
  - P.t.P.

    Even with half duplex, a point-to-point link is assumed.
  - Shared Media

    Even with a full duplex connection, a point-to-point link is not assumed.

    Note

    Point-to-point connection means a direct connection between two devices. A shared media connection is, for example, a connection to a hub.

- **Hello Time**
  Enter the interval after which the bridge sends configuration BPDUs

  Note

  The port-specific setting of the Hello time is only possible in MSTP compatible mode.

MST General

Multiple Spanning Tree configuration

With MSTP, in addition to RSTP, several VLANs can be managed in a LAN with separate RSTP trees.

Settings

- **MSTP Instance ID**
  Enter the number of the MSTP instance.

The table has the following columns:

- **MSTP Instance ID**
  Shows the number of the MSTP instance.
- **Root Address**
  Shows the MAC address of the root bridge
- **Root Priority**
  Shows the priority of the root bridge.

- **Bridge Priority**
  Enter the bridge priority. The value for the bridge priority is a whole multiple of 4096.
- **VLAN ID**
  Enter the VLAN ID. Here, you can also specify ranges with Start ID, "-", End ID. Several ranges or IDs are separated by ",".

MST port

Configuration of the Multiple Spanning Tree port parameters

On this page, you set the parameters for the ports of the configured multiple spanning tree instances.

Settings

- **MSTP Instance ID**
  Select the ID of the MSTP instance.

Table 1 has the following columns

- **1st column**
  Shows that the setting is valid for all ports.
- **MSTP Status**
  Select the setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2

The table has the following columns:

- **Port**
  Shows all available ports and link aggregations.
- **MSTP Instance ID**
  ID of the MSTP instance.
- **MSTP Status**
  Enable or disable MSTP for this port.
- **Priority**
  Enter the priority of the port. The priority is only evaluated when the path costs are the same. The value must be divisible by 16. If the value that cannot be divided by 16, the value is automatically adapted.
- **Cost Calc.**
  Enter the path cost calculation in the input box. If you enter the value "0" here, the automatically calculated value is displayed in the next box "Path Costs".

- **Path Costs** (only available online)
  This parameter is used to calculate the path that will be selected. The path with the lowest value is selected as the path. If several ports of a device have the same value for the path costs, the port with the lowest port number is selected.
  If the value for "Cost Calc." is "0", the automatically calculated value is displayed. Otherwise, the value of the "Cost Calc." box is displayed. The calculation of the path costs is largely based on the transmission speed. The higher the achievable transmission speed is, the lower the value of the path costs.
  Typical values for path costs with rapid spanning tree:
  - 10,000 Mbps = 2,000
  - 1000 Mbps = 20,000
  - 100 Mbps = 200,000
  - 10 Mbps = 2,000,000

  The values can, however, also be set individually.
- **Status** (only available online)
  Displays the current status of the port. The values are only displayed and cannot be configured. The "Status" parameter depends on the configured protocol. The following is possible for status:
  - Disabled

    The port only receives and is not involved in STP, MSTP and RSTP.
  - Discarding
    In the "Discarding" mode, BPDU frames are received. Other incoming or outgoing frames are discarded.
  - Listening

    In this status, BPDUs are both received and sent. The port is involved in the spanning tree algorithm.
  - Learning

    Stage prior to the forwarding status, the port is actively learning the topology (in other words, the node addresses).
  - Forwarding

    Following the reconfiguration time, the port is active in the network; it receives and forwards data frames.
- **Fwd. Trans** (only available online)
  Specifies the number of changes from the "Discarding" status to the "Forwarding" status

Enhanced Passive Listening Compatibility

Enabling the function

On this page, you enable the expanded compatibility for passive listening.

Settings

- **Enhanced Passive Listening Compatibility**
  Enable or disable this function for the entire device.

Table 1 has the following columns:

- **1st column**
  Shows that the settings are valid for all ports of table 2.
- **Setting**
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Displays the port of the device.
- **Setting**
  - enabled
    Enables the function for all ports of the device.
  - disabled
    Disables the function for all ports of the device.

Loop detection

With the "Loop Detection" function, you specify the ports for which loop detection will be activated. The ports involved send special test frames - the loop detection frames. If these frames are sent back to the device, there is a loop.

A "local loop" involving this device means that the frames are received again at a different port of the same device. If the sent frames are received again at the same port, there is a "remote loop" involving other network components.

Note

A loop is an error in the network structure that needs to be eliminated. The loop detection can help to find the errors more quickly but does not eliminate them. The loop detection is not suitable for increasing network availability by deliberately including loops.

Note

Note that loop detection is only possible at ports that were not configured as ring ports or standby ports.

Settings

- **Loop Detection**
  Enable or disable loop detection.
- **VLAN Loop Detection**
  Enable or disable loop detection in a VLAN.

Table 1 has the following columns:

- **1st column**
  Shows that the settings are valid for all ports of table 2
- **Threshold / Remote Reaction / Local Reaction**
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows the available ports.
- **Setting**
  Specify how the port handles loop detection frames.

  Note

  Test frames create additional network load. We recommend that you only configure individual switches, for example at branch points of the ring, as "sender" and the others as "forwarder".

  – sender
    Loop detection frames are sent out and forwarded.
  – forwarder
    Loop detection frames from other devices are forwarded.
  – blocked
    The forwarding of loop detection frames is blocked.
- **Threshold**
  Specify the number of received loop detection frames as of which a loop is assumed.
- **Remote Reaction**
  Specify how the port will react if a remote loop occurs. Select one of the two options from the drop-down list:
  – no action
    A loop has no effect on the port.
  – disable
    The port is blocked.

- Local Reaction
  Specify how the port will react if a local loop occurs. Select one of the two options from the drop-down list:
  - no action
    A loop has no effect on the port.
  - disable
    The port is blocked
- Status (only available online)
  This box shows whether loop detection is enabled or disabled for this port.
- Source Port (only available online)
  Shows the output port of the loop detection frame that triggered the last reaction
- Source VLAN (only available online)
  Shows the VLAN ID of the loop detection frame that triggered the last reaction.
  This is only possible if "VLAN Support Enabled" was selected earlier on the "Loop Detection Configuration" page.
- Reset (only available online)
  After a loop in the network has been eliminated, click this button to reset the port again

Link aggregation

Bundling network connections for redundancy and higher bandwidth

A link aggregation according to IEEE 802.3ad allows several connections between neighboring devices to be bundled to achieve higher bandwidths and to protect against failure.

Ports on both partner devices are included in link aggregations and the devices are then connected via these ports. To assign ports (in other words links) correctly to a partner device, the Link Aggregation Control Protocol (LACP) from the IEEE 802.3ad standard is used.

Up to 8 link aggregations can be defined. A maximum of 8 ports can be assigned to each link aggregation.

You can also use link aggregations within an MSTP instance, see "Layer 2" > "Spanning Tree" > "General (Page 1280)".

Settings

The table has the following columns:

- Port
  Shows the virtual port number of this link aggregation. This identifier is assigned internally by the firmware.
- Link Aggregation Name
  Enter the name for the link aggregation. This name can be specified by the user during configuration. The name is not absolutely necessary but can be useful to distinguish between the various link aggregations.
- MAC Address
  Shows the MAC address.

- Status
  Enable or disable link aggregation.
- MTU
  Specify the packet size.
- LACP
  - on
    Enables the sending of LACP frames.
  - off
    Disables the sending of LACP frames.
- Frame Distribution
  Set the type of distribution of packets on the individual links of an aggregation.
  - Destination&Source Mac
    The distribution is based on a combination of the destination and source MAC address.
  - Destination&Source IP-MAC
    The distribution is based on a combination of the destination and source IP address and MAC address.
- VLAN Mode
  Specify how the link aggregation is entered in a VLAN:
  - Hybrid
    The link aggregation sends tagged and untagged frames. It is not automatically a member of a VLAN.
  - Trunk
    The link aggregation only sends tagged frames and is automatically a member of all VLANs.
- Port
  Shows the ports that belong to this link aggregation. The following values can be selected from the drop-down list:
  - "-" (disabled)
    Link aggregation is disabled.
  - "a" (active)
    The port sends LACP frames and is only involved in the link aggregation when LACP frames are received.
  - "p" (passive)
    The port is only involved in the link aggregation when LACP frames are received.
  - "o" (on)
    The port is involved in the link aggregation and does not send any LACP frames.

  Note

  Within a "link aggregation", only ports with the following configuration are possible:
  - all ports with "o"
  - all ports with "a" or "p".

DCP forwarding

Applications

The DCP protocol is used by STEP 7 and the PST Tool for configuration and diagnostics. When shipped, DCP is enabled on all ports; in other words, DCP frames are forwarded at all ports. With this option, you can disable the sending of these frames for individual ports, for example to prevent individual parts of the network from being configured with the PST Tool or to divide the full network into smaller parts for configuration and diagnostics.

Settings

Table 1 has the following columns:

- 1st column
  Shows that the setting is valid for all ports of table 2.
- Setting
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- Transfer to table
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- Port
  Shows the available ports.
- Setting
  Specify whether the port should block or forward outgoing DCP frames. The following options are available:
  - forward
    DCP frames are forwarded via this port.
  - block
    No outgoing DCP frames are forwarded via this port. It is nevertheless still possible to receive via this port.

LLDP

Applications

PROFINET uses the LLDP protocol for topology diagnostics. In the default setting, LLDP is enabled for all ports; in other words, LLDP frames are sent and received on all ports. With this function, you have the option of enabling or disabling sending and/or receiving per port.

Settings

Table 1 has the following columns:

- **1st column**
  Shows that the setting is valid for all ports of table 2.
- **Setting**
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows the port.
- **Setting**
  Specify the LLDP functionality. The following options are available:
  - Rx
    This port can only receive LLDP frames.
  - Tx
    This port can only send LLDP frames.
  - Rx & Tx
    This port can receive and send LLDP frames.
  - "-" (disabled)
    This port can neither receive nor send LLDP frames.

Unicast

Filter

Address filtering

This page shows the current content of the unicast filter table. This table lists the source addresses of unicast address frames. The displayed entries were made statically, in other words, the user set them.

On this page, you also define the static unicast filters.

Settings

- **VLAN ID**
  Select the VLAN ID in which you configure a new static MAC address. If nothing is set, "VLAN1" is set as the basic setting.
- **MAC Address**
  Enter the MAC address here.

This table contains the following columns:

- VLAN ID
  Shows the VLAN-ID assigned to this MAC address.
- MAC Address
  Shows the MAC address of the node that the device has learned or the user has configured.
- Status
  Shows the status of each address entry:
  – static
    Configured by the user. Static addresses are stored permanently; in other words, they are not deleted when the aging time expires or when the switch is restarted.
- Port
  Shows the port via which the node with the specified address can be reached. Frames received by the device whose destination address matches this address will be forwarded to this port.

Note

You can only specify **one** port for unicast addresses.

Learning

Starting/stopping learning

Note

The page is only available if there is an online connection to the device.

With the automatic learning function, all connected devices are automatically entered in the unicast filter table. As long as the "Start learning" function is enabled, all learned unicast addresses are created immediately as static unicast entries.
The learning process is ended only after clicking the "Stop learning" button. With this method, learning can take a few minutes or several hours in larger networks before all nodes have really been learned. Only nodes that send packets during the learning phase are found.
By subsequently enabling the Port Lock function, only packets from the nodes known after the end of the learning phase (static unicast entries) will be accepted at the relevant ports.

Note

If the Port Lock function was already active on individual ports prior to the automatic learning phase, no addresses will be learned on these ports. This makes it possible to restrict learning to certain ports. To do this, first enable the Port Lock function of the ports that are not intended to learn addresses.

Settings

- **Start learning**
  Click the "Start learning" button to start the learning phase.
  The device now enters the addresses of connected devices until you stop the function.
- **Stop learning**
  Click the "Stop Learning" button to stop the learning phase.
  The learned entries are stored.
- **Deleting all static unicast addresses**
  Click the "Clear all static unicast addresses" button to delete all static entries.
  In large networks with numerous nodes, automatic learning may lead to a lot of undesired static entries. To avoid having to delete these individually, this button can be used to delete all static entries. This function is disabled during automatic learning.

Note

Depending on the number of entries involved, deleting may take some time.

Locked ports

Activating the access control

On this page, you can block individual ports for unknown nodes.

If the Port Lock function is enabled, packets arriving at this port from unknown MAC addresses are discarded immediately. Packets from known nodes are accepted by the port.
Since ports with the port lock function enabled cannot learn any MAC addresses, learned addresses on these ports are automatically deleted after the port lock function is enabled.

Settings

Table 1 has the following columns:

- **1st column**
  Shows that the setting is valid for all ports of table 2.
- **Setting**
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows the available ports.
- **Setting**
  Enable or disable access the port lock function for the port.

Blocking

Blocking the forwarding of unknown unicast frames

On this page, you can block the forwarding of unknown unicast frames for individual ports.

Settings

Table 1 has the following columns:

- **1st column**
  Shows that the setting is valid for all ports of table 2.
- **Setting**
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows the available ports.
- **Setting**
  Enable or disable the blocking of unicast frames

Note

Ring redundancy / standby

If ring redundancy or standby is enabled, the ports configured for this are not included in the blocking.

Multicast

Groups

Multicast applications

In the majority of cases, a frame is sent with a unicast address to a particular recipient. If an application sends the same data to several recipients, the amount of data can be reduced by sending the data using one multicast address. For some applications, there are fixed multicast addresses (NTP, IETF1 Audio, IETF1 Video etc.).

Reducing network load

In contrast to unicast frames, multicast frames represent a higher load for the device. Generally, multicast frames are sent to all ports. There are three ways of reducing the load caused by multicast frames:

- Static entry of the addresses in the multicast filter table.
- Dynamic entry of the addresses by listening in on IGMP parameter assignment frames (IGMP Configuration).
- Active dynamic assignment of addresses by GMRP frames.

The result of all these methods is that multicast frames are sent only to ports for which an appropriate address is entered.

"Multicast" shows the multicast frames currently entered in the filter table and their destination ports. The displayed entries were made statically, in other words, the user set them.

Settings

- VLAN ID
  Select the VLAN ID to be assigned to the MAC multicast address.
- MAC Address
  Here you enter a new MAC multicast address you want to configure.

The table has the following columns:

- VLAN ID
  Shows VLAN ID of the VLAN to which the MAC multicast address is assigned.
- MAC Address
  Shows the MAC multicast address that the device has learned or the user has configured.
- Status
  Shows the status of each address entry. The following information is possible:
  - static
    The address was entered statically by the user. Static addresses are stored permanently; in other words, they are not deleted when the aging time expires or when the device is restarted. These must be deleted by the user.

IGMP

Function

IE switches support "IGMP snooping" and the IGMP querier function. If "IGMP snooping" is enabled, IGMP frames are evaluated and the multicast filter table is updated with this information. If "IGMP Querier is also enabled, IE switches also send IGMP queries that trigger responses from IGMP-compliant nodes.

IGMP Snooping Aging Time

In this menu, you can configure the aging time for IGMP Configuration. When the time elapses, entries created by IGMP are deleted from the address table if they are not updated by a new IGMP frame.

This applies to all ports; a port-specific configuration is not possible.

IGMP Snooping Aging Time depending on the querier

SCALANCE XR500 as IGMP querier:

If a SCALANCE XR500 is used as an IGMP querier, the query interval is 125 seconds. For the "IGMP Snooping Aging Time", set at least 250 seconds.

Other IGMP queriers

If a different IGMP querier is used, the value of the "IGMP Snooping Aging Time" should be at least twice as long as the query interval.

Settings

- **IGMP Snooping**
  Enable or disable IGMP (Internet Group Management Protocol). The function allows the assignment of IP addresses to multicast groups. If the check box is selected, IGMP entries are included in the table and IGMP frames are forwarded.
- **IGMP Snooping Aging Time**
  Enter the value for the aging time in seconds in this box.
- **IGMP Querier**
  Enable or disable "IGMP Querier". The device sends IGMP queries.

GMRP

Enabling GMRP

By selecting the check box, you specify whether or not GMRP is used for each individual port. If "GMRP" is disabled for a port, no registrations are made for it and it cannot send GMRP frames.

Settings

- **GMRP**
  Enable or disable the GMRP function.

Table 1 has the following columns:

- **1st column**
  Shows that the setting is valid for all ports of table 2.
- **Setting**
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the settings are adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows the available ports and the link aggregations.
- **Setting**
- Enable or disable GMRP for the port or for the link aggregation.

Blocking

On this page, you can block the forwarding of unknown multicast frames for individual ports.

Settings

Table 1 has the following columns:

- **1st column**
  Shows that the setting is valid for all ports of table 2.
- **Setting**
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows the available ports.
- **Setting**
  Enable or disable the blocking of multicast frames.

Broadcast

Blocking the forwarding of broadcast frames

On this page, you can block the forwarding of broadcast frames for individual ports.

Note

Some communication protocols work only with the support of broadcast. In these cases, blocking can lead to loss of data communication. Block broadcast only when you are sure that you do not need it.

Settings

Table 1 has the following columns:

- **1st column**
  Shows that the setting is valid for all ports of table 2.
- **Setting**
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- **Transfer to table**
  If you click the button, the setting is adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  All available ports and the link aggregations are shown.
- **Setting**
  Enable or disable the blocking of broadcast frames.

PTP

General

Note

PTP is only available with SCALANCE X500.

The Precision Time Protocol (PTP) complying with IEEE 1588v2 allows the time-of-day synchronization of devices connected to the ports of a device. These devices forward the synchronization frames through the network using the "Transparent Clock" (TC) mechanism. The connection mechanisms "end-to-end" and "peer-to-peer" are supported.

To use IEEE 1588v2, enable this function and configure every port that is on the synchronization path as well as ports that are blocked due to redundancy mechanisms. IEEE 1588v2 can also be used with redundancy mechanisms in the ring such as HRP, standby linking of rings, MRP and RSTP.

Setting

- 1588 Mode
  You can make the following settings:
  - off
    The device does not process any PTP messages. PTP messages are, however, forwarded according to the rules of the device.
  - transparent
    The device adopts the function of a transparent clock and forwards PTP messages to other nodes while at the same time making entries in the correction field of the PTP message.

TC general

1588 Transparent Clock

On this page, you specify the general settings for PTP.

Settings

- Delay Mechanism
  Specify the delay mechanism the device will work with:
  - End-to-end(delay request response mechanism will be used)
  - Peer-to-peer (peer delay mechanism will be used)
- Domain Number
  Enter the identification number for the time domain. Synchronization is only for the devices within the domain. The device ignores PTP messages with a different domain number. A SCALANCE device can only be assigned to one synchronization domain.

TC port

On this page, you specify the ports that can process PTP messages.

Settings

Table 1 has the following columns:

- 1st column
  Shows that the settings are valid for all ports of table 2.
- Setting / Transport Mechanism
  Select the setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- Transfer to table
  If you click the button, the settings are adopted for all ports of table 2.

Table 2 has the following columns:

- **Port**
  Shows the available ports.
- **Setting**
  Enable or disable PTP. When enabled, the port processes PTP messages.
- **Faulty Flag**
  Shows error status relating to PTP.
  - **true**
    An error occurred.
  - **false**
    No error has occurred on this port.
- **Transport Mechanism**
  Specifies the protocol for transferring the PTP messages. This protocol must also be supported by the communications partner of the port
  - Ethernet
  - UDP IPv4

RMON

Statistics

Statistics

On this page you can specify the ports for which statistics are displayed.

Settings

- **RMON**
  If you select this check box, Remote Monitoring (RMON) allows diagnostics data to be collected on the device, prepared and read out using SNMP by a network management station that also supports RMON. This diagnostic data, for example port-related load trends, allow problems in the network to be detected early and eliminated.
- **Port**
  Select the ports for which statistics will be displayed.

The table has the following columns:

- **1st column**
  Select the check box of the ports for which no more statistics will be displayed.
- **Port**
  Shows the ports for which statistics will be displayed.

History

Samples of the statistics

On this page, you can specify whether or not samples of the statistics are saved for a port. You can specify how many entries should be saved and at which intervals samples should be taken.

Settings

Table 1 has the following columns:

- 1st column
  Shows that the settings are valid for all ports.
- Setting
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- Entries
  Enter the maximum number of samples to be stored at the same time. If "No Change" is entered, the entry in table 2 remains unchanged.
- Interval [s]
  Enter the interval after which the current status of the statistics will be saved as a sample. If "No Change" is entered, the entry in table 2 remains unchanged.
- Transfer to table
  If you click the button, the settings are adopted for all ports of table 2.

Table 2 has the following columns:

- Port
  Shows the port to which the settings relate.
- Setting
  Enable the port for which the history will be displayed.
- Entries
  Enter the maximum number of samples to be stored at the same time.
- Interval [s]
  Enter the interval after which the current status of the statistics will be saved as a sample.

Layer 3

Configuration

The page contains the overview of the layer 3 functions of the device. To use the "Routing", "VRRP", "OSPF" and "RIP" functions, the device requires a KEY-PLUG.

Settings

- Routing (only available on layer 3)
  Enable or disable the "Routing" function.
- DHCP Relay Agent
  Enable or disable the DHCP relay agent. You can configure other settings in "Layer 3 > DHCP Relay Agent".
- VRRP (only available on layer 3)
  Enable or disable routing using VRRP. To use VRRP, first enable the "Routing" function. You can configure other settings in "Layer 3 > VRRP".
- OSPF (only available on layer 3)
  Enable or disable routing using OSPF. To use OSPF , first enable the "Routing" function. You can configure other settings in "Layer 3 > OSPFv2".
- RIP (only available on layer 3)
  Enable or disable routing using RIP. To use RIO , first enable the "Routing" function. You can configure other settings in "Layer 3 > RIPv2".

Subnets

Overview

Subnet

The page shows the subnets for the selected interface. If more than one subnet is available on an interface, the first entry of this interface is of the address type "Primary". All other subnets have the address type "Secondary".

On this page, you can create further subnets "Layer 3 > Subnets > Overview". In "Layer 3 > Subnets > Configuration", you configure the settings of the created subnets.

Settings

The following settings are possible for subnets of the "Secondary" address type:

- Interface
  Select the interface on which you want to configure another subnet.

The table has the following columns:

- Interface
  Shows the interface.
- TIA Interface
  Shows whether or not the entry is a TIA interface.
- Interface Name
  Shows the name of the interface.
- MAC Address
  Shows the MAC address.

- **IP Address**
  Shows the IP address of the subnet.
- **Subnet Mask**
  Shows the subnet mask.
- **Address Type**
  Shows the address type. The following values are possible:
  - Primary
    The first IP address that was configured on an IP interface.
  - Secondary
    All other IP addresses that were configured on an interface.
- **IP Assignment Method**
  Shows how the IP address is assigned.
  - Static
    The IP address is static. You enter the IP settings in "IP Address" and "Subnet Mask".
  - Automatic (DHCP)
    The device obtains a dynamic IP address from a DHCP server.
- **Address Collision Detection Status**
  Shows the status of the "Detection of address collisions" function.
  - Idle
    The interface is not enabled and does not have an IP address.
  - Starting
    This status indicates the start-up phase. In this phase, the device initially sends a query as to whether the planned IP address already exists. If the address is not yet been assigned, the device sends the message that it is using this IP address as of now.
  - Conflict
    The interface is not enabled. The interface is attempting to use an IP address that is already been assigned.
  - Defending
    The interface uses a unique IP address. Another interface is attempting to use the same IP address.
  - Active
    The interface uses a unique IP address. There are no collisions.
  - Not supported
    The function for detection of address collisions is not supported.

Configuration

On this page, you configure the subnet. You create the subnet in "Layer 3 > Subnets > Overview".

Settings

- Interface (Name)
  Select the required interface.
- Interface Name
  Enter the name of the interface, e.g. eth0; P3.
- MAC Address (only available online)
  Shows the MAC address of the selected interface
- DHCP
  Enable or disable the DHCP client for this interface.
- IP Address
  Enter the IP address of the subnet. The IP address must not be used more than once.
- Subnet Mask
  Enter the subnet mask of the subnet you are creating. Subnets on different interfaces must not overlap.
- Address Type
  Shows the address type. The following values are possible:
  - Primary
    The first subnet of the interface.
  - Secondary
    All other subnets of the interface.
- Address Collision Detection
  Enable or disable the function.
  If new IP addresses become active in the network, this function checks whether this can result in address collisions.
  With this function, IP addresses that will be assigned twice can be detected.

  Note

  The function does not run a cyclic check.

- TIA Interface
  Enable or disable the setting.
  The following conditions apply to the TIA interface:
  - Only interfaces with the address type "Primary" can be enabled as the TIA interface.
  - There must only ever be one TIA interface.
  - There can only be one TIA interface.
  - A TIA interface is always a VLAN interface.

TIA interface and PROFINET interface

The IP address of the TIA interface is linked to the IP address located in the parameter group "Properties > General > PROFINET interface > Ethernet addresses".

Both IP addresses always have the same value. If you change the value of one IP address, the other IP address changes accordingly.

TIA Interface

The TIA interface is the interface via which all PROFINET functions of the IE switch are handled.

TIA interface is the name of the IP address for PROFINET from the perspective of the IE switch.

PROFINET interface

You make the IP setting of the PROFINET device in TIA under PROFINET interface.

PROFINET interface is the name of the IP address for PROFINET from the perspective of the TIA Portal.

Routes

Static route

On this page, you create the static routes. During automatic adaptations, static routes are not taken into account and need to be adapted manually.

Settings

- Destination Network
  Enter the network address of the destination.
- Subnet Mask
  Enter the corresponding subnet mask.
- Gateway
  Enter the IP address of the next gateway.
- Metric
  Enter the metric for the route. The metric corresponds to the quality of a connection, for example speed, costs.

The table has the following columns:

- Destination Network
  Shows the network address of the destination.
- Subnet mask
  Shows the corresponding subnet mask.
- Gateway
  Shows the IP address of the next gateway.
- Interface (only available online)
  Shows the interface of the route.
- Metric
  Enter the metric for the route. When creating the route, "not used" is entered automatically. The metric corresponds to the quality of a connection, for example speed, costs. If there are several equal routes, the route with the lowest metric value is used.
- Status (only available online)
  Shows the status of the route.

Route maps

General

Route maps

With route maps, you control how routing information is further processed. You can filter routing information and specify whether the information is further processed, modified or discarded.

Route maps operate according to the following principle:

- Routing information is compared with the filters of the route maps.
- The comparison is continued until the filters of a route map match the properties of an item of information.
- The information is then processed according to the route map settings:
  - The routing information is discarded.
  - The properties of the routing information are changed.

Settings

- Name
  Enter the name for the route map.
- Sequence No
  Enter a number for the route map.
  The numbers specify the order in which the route maps are processed.

The table has the following columns:

- Name
  Shows the name of the route map.
- Sequence No
  Shows the sequence number of the route map.
- Action
  Specify what happens to the routing information that matches the settings of the route map:
  - permit
    The routing information is further processed according to the settings you make in the "Settings" tab.
  - deny
    The routing information is discarded.

Interface and metric match

On this page, you specify whether or not the routing information for a route map is filtered according to interfaces or metric.

Settings

- **Route Map (name/seq. no.)**
  Select a route map.
  The created route maps are available to you.
- **Type**
  Select the basis for the filtering:
  - Interface
  - Metric
- **Interface**
  Select an interface.
  This box is active only if you have selected the "Interface" entry in the "Type" drop-down list.
- **Metric**
  Enter a value for the metric.
  This box is active only if you have selected the "Metric" entry in the "Type" drop-down list.

The table has the following columns:

- **Type**
  Shows the selected type:
  - Interface
  - Metric
- **Value**
  Shows the selected interface or the value of the metric.

Source match

On this page, you specify whether or not the routing information for a route map is filtered based on the source IP address.

Settings

- **Route Map (name/seq. no.)**
  Select a route map.
- **IP Address**
  Enter the IP address of the source on which the filtering is based.
- **Subnet Mask**
  Enter the subnet mask of the source on which the filtering is based.

The table has the following columns:

- **IP Address**
  Shows the IP address of the source.
- **Subnet Mask**
  Shows the subnet mask of the source.

Destination match

On this page, you specify whether or not the routing information for a route map is filtered based on the destination IP address.

Settings

- Route Map (name/seq. no.)
  Select a route map.
- IP Address
  Enter the IP address of the destination on which the filtering is based.
- Subnet Mask
  Enter the subnet mask of the destination on which the filtering is based.

The table has the following columns:

- IP Address
  Shows the IP address of the destination.
- Subnet Mask
  Shows the subnet mask of the destination.

Next hop match

On this page, you specify whether or not the filtering for a route map will be based on the router to which the routing information is sent next.

Settings

- Route Map (name/seq. no.)
  Select a route map.
- IP Address
  Enter the IP address of the router to which the routing information will be sent next.

The table has the following columns:

- IP Address
  Shows the IP address of the next router.

Settings

On this page, you specify whether or not the routing information for a route map will be changed.

If, for example, you have filtered based on a certain metric, you can change the value of the metric here. The routing information is then forwarded with the new value.

Settings

- Route Map (name/seq. no.)
  Select a route map.

The table has the following columns:

- Name
  Shows the name of the route map.
- Sequence No
  Shows the sequence number of the route map.
- Interface
  Enter the new interface via which the routing information will be forwarded.
- Metric
  Enter the new value for the metric with which the routing information will be forwarded.
- Next Hop IP Address
  Enter the IP address of the router to which the routing information will be sent next.

DHCP Relay Agent

General

If the DHCP server is in a different network, the device cannot reach the DHCP server. The DHCP relay agent intercedes between a DHCP server and the device. The DHCP relay agent forwards the port number of the device with the DHCP query to the DHCP server. If a DHCP server is unreachable, the device can switch to a different DHCP server.

Settings

- DHCP Relay Agent (Opt. 82)
  Enable or disable the DHCP relay agent.
- Server IP Address
  Enter the IP address of the DHCP server.

The table has the following columns:

- Server IP address
  Shows the IP address of the DHCP server.

Option

Parameters of the DHCP relay agent

On this page, you can specify parameters for the DHCP server, for example the circuit ID. The circuit ID describes the origin of the DHCP query, for example which port received the DHCP query.

You specify the DHCP server on the "General" tab.

Settings

Global configuration

- **Circuit ID router index**
  When enabled, the router index is added to the generated circuit ID.
- **Circuit ID recipient VLAN ID**
  When enabled, the VLAN ID is added to the generated circuit ID.
- **Circuit ID receiving port**
  When enabled, the receiving port is added to the generated circuit ID.

> **Note**
>
> You need to select a least one option.

- **Remote ID**
  Shows the device ID.

Interface-specific configuration

- **Interface**
  Select the required interface.

The table has the following columns:

- **Interface**
  Shows the selected interface.
- **Remote ID type**
  Select the type of the device ID. You have the following options:
  - IP Address
    The current IP address of the interface will be used as the device identifier and copied to the "Remote ID" box.

    > **Note**
    >
    > **No automatic updating**
    >
    > There is no link between the "Remote ID" box and the currently set IP address.
    >
    > If the IP address is changed, the new IP address is not entered in the "Remote ID" box automatically. Only the value of the "Remote ID Type" box changes to "Free Text". To use the current IP address again, select "IP Address" again in the "Remote ID Type" box.

  - MAC Address
    The MAC address of the device is used as the device ID.
  - Free Text
    If you use "Free Text", you can enter the device name as the device identifier in the "Remote ID" box.
- **Remote ID**
  The box can only be edited if you select the entry "Free Text" for "Remote ID Type".
  The remote IP must be unique.

- Circuit ID Type
  Select the type of circuit ID from the drop-down list. You have the following options:
  - Predefined
    The circuit ID is created automatically based on the router index, VLAN ID or port.
  - Free Number
    If you use "Free Number", you can enter the ID for "Circuit ID".
- Circuit ID
  Enter the circuit ID.
  The box can only be edited if you select the "Free Number" entry for the "Circuit ID Type".

VRRP

Router

On this page, you create virtual routers. You can configure other parameters in "Layer 3 > VRRP > Configuration".

Note

- VRRP is available only on layer 3.
- VRRP can only be used in conjunction with VLAN interfaces. Router ports are not supported.
- Select "VRRP" to configure VRRP

Settings

The following settings are available:

- VRRP
  Enable or disable routing using VRRP.
- React to pings on virtual interfaces
  When enabled, the virtual IP addresses also reply to the ping.
- Interface
  Select the interface operating as virtual router.
- VRID
  Enter the ID of the virtual router in the input box. This ID defines the group of routers that form a virtual router (VR). In the group, this is the same. it can no longer be used for other groups.

The table has the following columns:

- Interface
  Shows the Interface that functions as the virtual router.
- VRID
  Shows the ID of the virtual router.

- **Virtual MAC Address**
  Shows the virtual MAC address of the virtual router.
- **Primary IP Address**
  Shows the primary IP address in this VLAN. The entry 0.0.0.0 means that the "Primary" address on this VLAN is used. Otherwise all IP addresses configured in this VLAN in "Layer 3 > Subnets" are valid values.
- **Router Status**
  Shows the current status of the virtual router. Possible values are as follows:
  - Master
    The router is the master router and handles the routing functionality for all assigned IP addresses.
  - Backup
    The router is the backup router. If the master router fails, the backup router takes over the tasks of the master router.
  - Initialize
    The virtual router has just been turned on. It will soon change to the "Master" or "Backup" status.
- **Master IP Address** (only available online)
  Shows the IP address of the master router.
- **Priority**
  Shows the priority of the virtual router.
  The current master router is automatically given 255. All other priorities can be distributed freely among the VRRP routers. The higher the priority, the earlier the VRRP router becomes "Master".
- **Advert. Internal**
  Shows the interval at which the master router sends VRRP packets.
- **Preempt**
  Shows the precedence of a router when changing roles between backup and master.
  - yes
    This router has precedence when changing roles.
  - no
    This router does not have precedence when changing roles.

Configuration

Introduction

On this page, you configure the virtual router.

Note

VRRP is available only on layer 3.

Description of the displayed values

The page contains the following boxes:

- **Interface / VRID**
  Select the ID of the virtual router to be configured.
- **Primary IP Address**
  Select the primary IP address. If the router becomes master router, the router uses this IP address.

  Note

  If you only configure one subnet on this VLAN, no entry is necessary. The entry is 0.0.0.0. If you configure more than one subnet on the VLAN and you want a specific IP address to be used as the source address for VRRP packets, select the IP address from the drop-down list. Otherwise, the IP address with priority will be used.

- **Master**
  If this option is enabled, the primary IP address is entered for "Associated IP Address". This means that the highest priority IP address of the VRRP router is used as the virtual IP address of the virtual master router. The option must be disabled for the backup routers in this group and the IP address of the router in "Assigned IP address" must be used.
- **Priority**
  Enter the priority of this virtual router.
  The current master router is always given 255. All other priorities can be distributed freely among the redundant routers. The higher the priority, the earlier the router becomes "Master".
- **Advertisement Interval [s]**
  Enter the interval in seconds after which a master router sends a VRRP packet again.
- **Preempt lower priority master**
  Allow the precedence when changing roles between Backup and Master based on the selection process.

Address overview

Overview

This page shows which IP addresses the virtual router monitors.

Note

VRRP is available only on layer 3.

Settings

The table has the following columns:

- **Interface**
  Shows the Interface that functions as the virtual router.
- **VRID**
  Shows the ID of this virtual router.
- **Number of Addresses**
  Shows the number of IP addresses.
- **Associated IP Address (1) ... Associated IP Address (4)**
  Shows the router IP addresses monitored by this virtual router. If a router takes over the role of master, the routing function is taken over by this router for all these IP addresses.

Address configuration

Note

VRRP is available only on layer 3.

Settings

The following settings are possible:

- **Interface / VRID**
  Select the required virtual router.
- **Associated IP Address**
  Enter the IP address that the virtual router will monitor.

The table has the following columns:

- **Associated IP Address**
  Shows the IP addresses that the virtual router monitors.

OSPFv2

Configuration

Introduction

On this page, you configure routing with OSPF.

Note

OSPF is available only on layer 3.

Settings

- OSPFv2
  Enable or disable routing using OSPF.
- Router ID
  Enter the name of one of the OSPF interfaces. The name is entered in the IP address format and does not need to match the local IP address.
- OSPFv2 RFC1583 Compatibility
  Enable the option if you still have old OSPF routers in operation that are not compatible with RFC 2328.
- Border Router
  Shows the status of the OSPF router. If the local system is an active member in at least 2 areas, this is an area border router.
- AS Border Router
  Specify whether the router is an AS border router. An AS border router intercedes between multiple autonomous systems, for example if you have an additional RIP network. An AS border router is also necessary to add and to distribute static routes.
- New LSA Received (only available online)
  Shows the number of received LSAs.
  Updates and local LSAs are not counted.
- New LSA Configured (only available online)
  Number of different LSAs sent by this local system.
- External LSA Maximum
  To limit the number of entries of external LSAs in the database, enter the maximum number of external LSAs.
- Exit Interval [s]
  Enter the interval after which the OSPF router once again attempts to come out of the overflow status. A 0 means that the OSPF router attempts to exit the overflow status only following a restart.
- Inbound Filter
  Select a route map that filters inbound routes.

- **Redistribute Routes**
  Specify which known routes are distributed using OSPF.
  The following types of route exist:
  - Default
  - Connected
  - Static
  - RIPv2

  Note

  The settings can only be enabled on an AS border router. Enabling "Default" and "Static", in particular, can cause problems if they are enabled at too many points in the network, for example, forwarding loops.

- **Route Map**
  Select a route map that filters which routes are forwarded using OSPF.

Areas

Overview

An autonomous system can be divided into smaller areas.

On this page, you can view, create, modify or delete the areas of the router.

Note

OSPF is available only on layer 3.

Settings

- **Area ID**
  Enter the ID of the area. The database is synchronized for all routers of an area.
  Input format: x.x.x.x
  x = 0 ... 255

This table contains the following columns:

- **Area ID**
  Shows the ID of the area.
- **Area Type**
  Select the area type from the drop-down list.
  - Normal
  - Stub
  - NSSA
  - Backbone
- **Summary**
  Specify whether summary LSAs are generated for this area.
  - Summary: Summary LSAs are sent to the area.
  - No Summary: Summary LSAs are not sent to the area.
- **Metric**
  Displays the costs for the OSPF interface.
- **Updates**
  Shows the number of recalculations of the routing tables.
- **LSA Count**
  Shows the number of LSAs in the database.
- **Area BR**
  Shows the number of reachable area border routers (ABR) within this area.
- **AS BR**
  Shows the number of reachable autonomous system border routers (ASBR) in this area.

Area range

Creating a new OSPFv2 area range

On this page, networks can be grouped together under one area ID. The method is used only with area border routers. This means that an area border router only propagates one route for each address area to the outside.

Note

OSPF is available only on layer 3.

Settings

- **Area ID**
  Select the area ID. You configure the ID in "Layer 3 > OSPFv2 > Areas".
- **Subnet Address**
  Enter the address of the network that will be grouped.
- **Subnet Mask**
  Enter the subnet mask of the network that will be grouped.

This table contains the following columns:

- **Area ID**
  Shows the ID of the area.
- **Subnet Address**
  Shows the address of the network that will be grouped with other networks.
- **Subnet Mask**
  Shows the subnet mask of the network that will be grouped with other networks.
- **Advertise**
  Enable this option to advertise the grouped network.

Interfaces

Overview

On this page, you can configure OSPF interfaces.

Note

OSPF is available only on layer 3.

Settings

- **IP Address**
  Select the IP address of the OSPF interface.
- **Area ID**
  Select the area ID of the area with which the OSPF interface is connected.

  Note

  For secondary address types, select the same Area ID as for the corresponding primary address type.
  The information whether an address type is primary or secondary can be found in the "Address Type" column on the "Subnets > Overview" page.

This table contains the following columns:

- IP Address
  Displays the IP address of the OSPF interface.
- Area ID
  Select the area ID with which the OSPF interface is connected.
- OSPF Status
  Specify whether or not OSPF is active on the interface.
  - Enabled: OSPF is enabled on the interface.
  - Disabled: OSPF is disabled on the interface
- Metric
  Enter the costs for the OSPF interface.
- Priority
  Enter the router priority. The priority is only relevant for selecting the designated router. This parameter can be selected differently on routers within the same subnet.
- Trans. Delay
  Enter the required delay when sending a connection update.
- Retrans. Delay
  Enter the time after which an OSPF packet is transferred again if no confirmation was received.
- Hello Interval
  Enter the interval between two Hello packets.
- Dead Interval
  Enter the interval after which the neighbor router is marked as "failed" if no more Hello packets are received from it during this time.

Interface authentication

Configuring the interface login

On this page, you define the authentication of the interface.

Note

OSPF is available only on layer 3.

Settings

- **OSPF Interface**
  Select the OSPF interface for which you want to configure authentication.
- **Authentication type**
  Select the authentication method of the OSPF interface. You have the following options:
  - None: No authentication
  - simple: Authentication using an unencrypted password
  - MD5: Authentication using MD5

Simple authentication

- **Password**
  For "Simple authentication, enter the password if you have selected this type of authentication.
- **Confirmation**
  Confirm the entered password.

MD5 authentication

- **Authentication Key ID**
  If you have selected this type of authentication, enter the ID for the MD5 authentication under which the password is used as the key. Since the key ID is transferred with the protocol, the same key must be stored under the same key ID on all neighboring routers.

The table has the following columns:

- **Authentication Key ID**
  Can only be edited if you set the MD5 authentication method. It is only possible to use several keys there.
- **MD5 Key**
  Enter the MD5 key.
- **Confirm MD5 Key**
  Confirm the entered key.
- **Youngest Key ID** (only available online)
  Shows whether or not the MD5 key is the latest key ID.

Virtual links

Overview

Due to the protocol, each area border router must have access to the backbone area. If a router is not connected directly to the backbone area, a virtual link to it is created.

Note

OSPF is available only on layer 3.

Note

Note that when creating a virtual link both the transit area and the backbone area must already be configured.

the virtual link must be configured identically at both ends.

Note

- **Virtual links are not effective unless the device is an ABR.** (only available online)
  This is displayed when at least one virtual link entry is configured and the device is not an area border router.

Settings

- **Neighbor Router ID**
  Enter the ID of the neighbor router at the other end of the virtual connection.
- **Transit Area ID**
  Select the ID of the area that connects the two routers.

This table contains the following columns:

- **Transit Area ID**
  Shows the ID via which the two routers are connected.
- **Neighbor Router ID**
  Shows the ID of the neighbor router at the other end of the virtual connection.
- **Virt. Link Status**
  Specify the status of the virtual link. The following statuses are possible:
  - down: The virtual link is inactive.
  - point-to-point: The virtual link is active.
- **Trans. Delay**
  Enter the required delay when sending a link update packet (in seconds).
- **Retrans. Delay**
  Enter the time after which a packet is transferred again if no confirmation was received (in seconds).
- **Hello Interval**
  Enter the interval between two Hello packets (in seconds).
- **Dead Interval**
  Enter the interval after which the neighbor router counts as "failed" if no more Hello packets are received from it during this time (in seconds).

Virtual Link Authentication

Configuring the virtual link login

On this page, you define the authentication for the virtual link.

Note

OSPF is available only on layer 3.

Settings

- **Virtual link (area/neighbor)**
  Select the virtual link for which you want to configure authentication.
- **Authentication type**
  Select the authentication method of the OSPF interface. You have the following options:
  - None: No authentication
  - simple: Authentication using an unencrypted password.
  - MD5: Authentication using MD5

Simple authentication

- **Password**
  Enter the password for "simple authentication".
- **Confirmation**
  Confirm the entered password.

MD5 authentication

- **Authentication Key ID**
  Enter the ID for MD5 authentication with which the password will be used as a key. Since the key ID is transferred with the protocol, the same key must be stored under the same key ID on all neighboring routers.

The table has the following columns:

- **Authentication Key ID**
  Can only be edited if you set the MD5 authentication method. It is only possible to use several keys there.
- **MD5 Key**
  Enter the MD5 key.
- **Confirm MD5 Key**
  Confirm the entered key.
- **Youngest Key ID**
  Shows whether or not the MD5 key is the latest key ID.

RIPv2

Configuration

On this page, you configure routing with RIP.

Note

RIPv2 is available only on layer 3.

Settings

- **RIPv2**
  Enable or disable routing using RIPv2.
- **Inbound Filter**
  Select a route map that filters inbound routes.
- **Redistribute Routes**
  Specify which known routes are distributed using RIPv2.
  The following types of route exist:
  - Static Default
  - Connected
  - Static
  - OSPF
- **Route Map**
  Select a route map that filters which routes are forwarded using RIPv2.

Interfaces

Overview

On this page, you can configure RIPv2 interfaces.

Note

RIPv2 is available only on layer 3.

Settings

- **IP Address**
  Select the IP address of the RIPv2 interface.

This table contains the following columns:

- **IP Address**
  Displays the IP address of the RIPv2 interface.
- **Send Updates**
  Select the way in which updates are sent:
  - no send
    No updates are sent.
  - RIPv1
    Updates for RIPv1 are sent.
  - RIPv1-compat.
    RIPv2 updates are sent as broadcasts according to the rules of RIPv1.
  - RIPv2
    Updates for RIPv2 are sent as multicasts.
  - RIPv1 demand/RIPv2 demand
    RIP packets are sent only as a response to an explicit query.
- **Receive Updates**
  Select the form in which received updates are accepted:
  - no receive
    No updates are received.
  - RIPv1
    Only updates of RIPv1 are received.
  - RIPv2
    Only updates of RIPv2 are received.
  - RIPv1/v2
    Updates of RIPv1 and RIPv2 are received.
- **Default Metric**
  Enter the costs for the RIPv2 interface.

Security

Passwords

Note

The page is only available if there is an online connection to the device.

Only the administrator can change the device passwords for administrator and users.

The factory settings for the passwords when the devices ship are as follows:

- Administrator: admin
- User: user

Note

If you log on the first time or log on after a "Reset to factory defaults and restart", you will be prompted to change the password.

Note

Changing the password in "Trial" configuration mode

Even if you change the password in "Trial" configuration mode, this change is saved immediately.

Settings

- **Current Admin Password**
  Enter the valid administrator password.
- **User name**
  Select the required user. The following options are available:
  - Administrator: admin
  - User: user
- **New Password**
  Enter the new password.
- **Password Confirmation**
  Confirm the new password.

AAA

General

The term "AAA" used in the menu stands for "Authentication, Authorization, Accounting" and is used to identify and allow network nodes, to provide them with the appropriate services and to determine the amount of use.

On this page, you configure the login.

Settings

- Login Authentication
  Specify how the login is made:
  - Local
    Login with local user name and password.
  - RADIUS
    Login using a RADIUS server.

RADIUS client

The term "AAA" used in the menu stands for "Authentication, Authorization, Accounting" and is used to identify and allow network nodes, to provide them with the appropriate services and to determine the amount of use.

Authentication over an external server

The concept of RADIUS is based on an external authentication server. An end device can only access the network after the device has verified the logon data of the device with the authentication server. Both the end device and the authentication server must support the EAP protocol (Extensive Authentication Protocol).

Each column of the table contains access data for one server. In the search order, the primary server is queried first. If the primary server cannot be reached, secondary servers are queried in the order in which they are entered.

If no server responds, there is no authentication. The client has no access to the network although a link is indicated at the port.

Settings

The table has the following columns:

- Server IP Address
  Enter the IP address of the RADIUS server.
- Server Port
  Here, enter the input port on the RADIUS server. As default, input port 1812 is set.
- Shared Secret
  Enter your access ID.
- Confirm Shared Secret
  Enter your access ID again as confirmation.
- Max. Retrans
  Enter the maximum number of query attempts before another configured RADIUS server is queried or the login counts as having failed. As default, 3 is set.

- Primary Server
  Specify whether or not this server is the primary server. You can select one of the options "yes" or "no".
- Status
  Enable or disable the RADIUS server.

Note

You can configure a maximum of two servers on this page.

802.1x authenticator

The term "AAA" used in the menu stands for "Authentication, Authorization, Accounting" and is used to identify and allow network nodes, to provide them with the appropriate services and to determine the amount of use.

Enabling authentication for individual ports

By selecting the check box, you specify whether or not network access protection according to IEEE 802.1x is enabled on this port.

Settings

- MAC Authentication
  Enable or disable MAC authentication for the device.
- Guest VLAN
  Enable or disable the "Guest VLAN" function for the device.

Table 1 has the following columns:

- 1st column
  Shows that the settings are valid for all ports of table 2.
- 802.1x Auth. Control
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- 802.1x Re-authentication
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- MAC Authentication
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- Adopt RADIUS VLAN Assignment
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- MAC Auth. Max Permitted Addresses
  Enter how many end devices are allowed to be connected to the port at the same time. If "No Change" is entered, the entry in table 2 remains unchanged.

- Guest VLAN
  Select the required setting. If "No Change" is selected, the entry in table 2 remains unchanged.
- Guest VLAN ID
  Enter the VLAN ID of the port. If "No Change" is entered, the entry in table 2 remains unchanged.
- Guest VLAN max. Permitted Addresses
  Enter how many end devices are allowed in the guest VLAN at the same time.
- Transfer to table
  If you click the button, the settings are adopted for all ports of table 2.

Table 2 has the following columns:

- Port
  Shows the port to which the setting relates.
  If a configuration is not possible for a port, it is displayed grayed out and you cannot modify the settings.
- 802.1x Auth. Control
  Specify the authentication of the port:
  - Force Unauthorized
    Data traffic via the port is blocked.
  - Force Authorized
    Data traffic via the port is allowed without any restrictions.
    Default setting
  - Auto
    End devices are authenticated on the port with the "802.1x" method.
    The data traffic via the port is permitted or blocked depending on the authentication result.
- 802.1x Re-authentication
  Enable this option if you want reauthentication of an already authenticated end device to be repeated cyclically.
- MAC Authentication
  Enable this option if you want end devices to be authenticated with the "MAC Authentication" method.
- Adopt RADIUS VLAN Assignment
  The RADIUS server informs the IE switch of the VLAN to which the port belongs.
  Enable this option if you want the information of the server to be taken into account. The port then belongs to the corresponding VLAN.
  If the option is disabled, the VLAN information is discarded.
- MAC Auth. Max Permitted Addresses
  Enter how many end devices are allowed to be connected to the port at the same time.
- Guest VLAN
  Enable this option if you want the end device to be permitted in the guest VLAN if authentication fails.
- Guest VLAN ID
  Enter the VLAN ID of the port.

- Guest VLAN max. Permitted Addresses
  Enter how many end devices are allowed in the guest VLAN at the same time.
- MAC Auth. Currently Permitted Addresses
  Shows the number of currently connected end devices.
- MAC Auth. Currently Blocked Addresses
  Shows the number of currently blocked end devices.
- Guest VLAN Currently Permitted Addresses
  Shows how many end devices are currently allowed in the guest VLAN.

Port ACL MAC

Rule configuration

On this page, you specify the ACL rules for the MAC-based ACL.

Settings

The table has the following columns:

- Rule Number
  Shows the number of the ACL rule. When you create a rule, a new row with a unique number is created.
- Source MAC
  Enter the unicast MAC address of the source.
  Dest. MAC
  Enter the unicast MAC address of the destination.
- Action
  Select the action. The following is possible:
  - Forward
    If the frame complies with the ACL rule, the frame is forwarded.
  - Discard
    If the frame complies with the ACL rule, the frame is not forwarded.
- Ingress Port
  Shows a list of all ingress ports to which this rule applies.
- Egress Port
  Shows a list of all egress ports to which this rule applies.

Note

If you enter the address "00:00:00:00:00:00" for the source and/or destination MAC address, the rule created in this way applies to all source or destination MAC addresses.

Port ingress rules

On this page, you specify the ACL rule according to which incoming frames are handled by the port.

Settings

- Ports
  Select the required port.
- Add Rule
  Select the ACL rule to be assigned to the port. You specify the ACL rule on the "Rules Configuration" page.
- Add
  To permanently assign the ACL rule to the port, click the "Add" button. The configuration is shown in the table.
- Remove Rule
  Select the ACL rule to be deleted.
- Remove
  To remove the ACL rule from the port, click the "Remove" button.

The table has the following columns:

- Rule Order
  Shows the order of the ACL rules.
- Rule Number
  Shows the number of the ACL rule. When you create a rule, a new row with a unique number is created.
- Source MAC
  Shows the unicast MAC address of the source.
- Dest. MAC
  Shows the unicast MAC address of the destination.
- Action
  Select the action. The following is possible:
  - Forward
    If the frame complies with the ACL rule, the frame is forwarded.
  - Discard
    If the frame complies with the ACL rule, the frame is not forwarded.

Port egress rules

On this page, you specify the ACL rule according to which outgoing frames are handled by the port.

Description

- **Ports**
  Select the required port.
- **Add Rule**
  Select the ACL rule to be assigned to the port. You specify the ACL rule on the "Rules Configuration" page.
- **Add**
  To permanently assign the ACL rule to the port, click the "Add" button. The configuration is shown in the table.
- **Remove Rule**
  Select the ACL rule to be deleted.
- **Remove**
  To remove the ACL rule from the port, click the "Remove" button.

The table has the following columns:

- **Rule Order**
  Shows the order of the ACL rules.
- **Rule Number**
  Shows the number of the ACL rule. When you create a rule, a new row with a unique number is created.
- **Source MAC**
  Shows the unicast MAC address of the source.
- **Dest. MAC**
  Shows the unicast MAC address of the destination.
- **Action**
  Select the action. The following is possible:
  - Forward
    If the frame complies with the ACL rule, the frame is forwarded.
  - Discard
    If the frame complies with the ACL rule, the frame is not forwarded.

Port ACL IP

Rule configuration

On this page, you specify the rules for the IP-based ACL.

Settings

The table has the following columns:

- Rule Number
  Shows the number of the ACL rule. When you create a rule, a new row with a unique number is created.
- Source IP
  Enter the IP address of the source.
- Source Subnet Mask
  Enter the subnet mask in which the source is located.
- Dest. IP
  Enter the IP address of the destination.
- Dest. Subnet Mask
  Enter the subnet mask in which the destination is located.
- Action
  Select the action from the drop-down list. The following is possible:
  - Forward
    If the frame complies with the ACL rule, the frame is forwarded.
  - Discard
    If the frame complies with the ACL rule, the frame is not forwarded.
- Ingress Port
  Shows a list of all ingress ports to which this rule applies.
- Egress Port
  Shows a list of all egress ports to which this rule applies.

Protocol configuration

On this page, you specify the rules for protocols.

Settings

The table has the following columns:

- Rule Number
  Shows the number of the protocol rule. When you create a rule, a new row with a unique number is created.
- Protocol
  Select the protocol for which this rule is valid.
- Protocol Number
  Enter a protocol number to define further protocols.
  This box can only be edited if you have set "Other Protocol" for the protocol.
- Source Port Min.
  Enter the lowest possible port number of the source port.
  This box can only be edited if you have set "TCP" or "UDP" for the protocol.

- Source Port Max.
  Enter the highest possible port number of the source port.
  This box can only be edited if you have set "TCP" or "UDP" for the protocol.
- Dest. Port Min.
  Enter the lowest possible port number of the destination port.
  This box can only be edited if you have set "TCP" or "UDP" for the protocol.
- Dest. Port Max.
  Enter the highest possible port number of the destination port.
  This box can only be edited if you have set "TCP" or "UDP" for the protocol.
- Message Type
  Enter a message type to decide the format of the message.
  This box can only be edited if you have set "ICMP" for the protocol.
- Message Code
  Enter a message code to specify the function of the message.
  This box can only be edited if you have set "ICMP" for the protocol.
- DSCP
  Enter a value for classifying the priority.
  This box cannot be edited if you have set "ICMP" for the protocol.

Port ingress rules

On this page, you specify the ACL rule according to which incoming frames are handled by the port.

Settings

- Ports
  Select the required port.
- Add Rule
  Select the ACL rule to be assigned to the port. You specify the ACL rule on the "Rules Configuration" page.
- Add
  To permanently assign the ACL rule to the port, click the "Add" button. The configuration is shown in the table.
- Remove Rule
  Select the ACL rule to be deleted.
- Remove
  To remove the ACL rule from the port, click the "Remove" button.

The table has the following columns:

- Rule Order
  Shows the order of the ACL rules.
- Rule Number
  Shows the number of the ACL rule. When you create a rule, a new row with a unique number is created.

- Protocol
  Shows the protocol for which this rule is valid.
- Protocol Number
  Shows the protocol number.
- Source IP
  Shows the IP address of the source.
- Source Subnet Mask
  Shows the subnet mask in which the source is located.
- Dest. IP
  Enter the IP address of the destination.
- Dest. Subnet Mask
  Enter the subnet mask in which the destination is located.
- Action
  Select the action. The following is possible:
  - Forward
    If the frame complies with the ACL rule, the frame is forwarded.
  - Discard
    If the frame complies with the ACL rule, the frame is not forwarded.
- Source Port Min.
  Shows the lowest possible port number of the source port.
- Source Port Max.
  Shows the highest possible port number of the source port.
- Dest. Port Min.
  Shows the lowest possible port number of the destination port.
- Dest. Port Max.
  Shows the highest possible port number of the destination port.
- Message Type
  Shows the message type.
- Message Code
  Shows the message code.
- DSCP
  Shows the value for classifying the priority.

Port egress rules

On this page, you specify the ACL rule according to which outgoing frames are handled by the port.

Settings

- Ports
  Select the required port.
- Add Rule
  Select the ACL rule to be assigned to the port. You specify the ACL rule on the "Rules Configuration" page.
- Add
  To permanently assign the ACL rule to the port, click the "Add" button. The configuration is shown in the table.
- Remove Rule
  Select the ACL rule to be deleted.
- Remove
  To remove the ACL rule from the port, click the "Remove" button.

The table has the following columns:

- Rule Order
  Shows the order of the ACL rules.
- Rule Number
  Shows the number of the ACL rule. When you create a rule, a new row with a unique number is created.
- Protocol
  Shows the protocol for which this rule is valid.
- Protocol Number
  Shows the protocol number.
- Source IP
  Shows the IP address of the source.
- Source Subnet Mask
  Shows the subnet mask in which the source is located.
- Dest. IP
  Enter the IP address of the destination.
- Dest. Subnet Mask
  Enter the subnet mask in which the destination is located.
- Action
  Select the action from the drop-down list. The following is possible:
  - Forward
    If the frame complies with the ACL rule, the frame is forwarded.
  - Discard
    If the frame complies with the ACL rule, the frame is not forwarded.
- Source Port Min.
  Shows the lowest possible port number of the source port.
- Source Port Max.
  Shows the highest possible port number of the source port.

- Dest. Port Min.
  Shows the lowest possible port number of the destination port.
- Dest. Port Max.
  Shows the highest possible port number of the destination port.
- Message Type
  Shows the message type.
- Message Code
  Shows the message code.
- DSCP
  Shows the value for classifying the priority.

Management ACL

On this page, you can increase the security of your device. To specify which station with which IP address is allowed to access your device, configure the IP address or an entire address range.

You can select the protocols and the ports of the station with which it is allowed to access the device. You define the VLAN in which the station may be located. This ensures that only certain stations within a VLAN have access to the device.

Note

Note that a bad configuration may mean that you can no longer access the device.

Settings

- Management ACL
  Enable or disable the function.
- IP Address
  Enter the IP address or the network address to which the rule will apply. If you use the IP address 0.0.0.0, the settings apply to all IP addresses.
- Subnet Mask
  Enter the subnet mask. The subnet mask 255.255.255.255 is for a specific IP address. If you want to allow a subnet, for example a C subnet, enter 255.255.255.0. The subnet mask 0.0.0.0 applies to all subnets.

The table has the following columns:

- Rule Order
  Shows the number of the ACL rule. When you create a rule, a new row with a unique number is created.
- IP Address
  Shows the IP address.
- Subnet Mask
  Shows the subnet mask.

- VLANs Allowed
  Enter the number of the VLAN from which access is allowed.
  Only stations located in these VLANs can access the device. If this input box remains empty, there is no restriction.
  You can enter several individual VLANs and VLAN ranges separated by commas, for example 1,5,10-12.
- SNMP
  Specify whether the station or the IP address can access the device using the SNMP protocol.
- TELNET
  Specify whether the station or the IP address can access the device using the TELNET protocol.
- HTTP
  Specify whether the station or the IP address can access the device using the HTTP protocol.
- HTTPS
  Specify whether the station or the IP address can access the device using the HTTPS protocol.
- SSH
  Specify whether the station or the IP address can access the device using the SSH protocol.
- Px.y
  Specify whether the station or the IP address can access this device via this port.

8.1.4.10 Configuring PROFIBUS DP

Configuring distributed I/O

In the chapter below you will learn how to configure distributed I/O. This chapter will also provide you with basic information on configuring a DP master system.

Basic information

- Basics of configuring a DP master system (Page 1340)
- DP slaves in the hardware catalog (Page 1341)
- Structure of a configuration with simple DP slaves (mono master system) (Page 1343)
- Installation of a configuration with intelligent slaves (I-slaves) (Page 1343)
- Installation of a configuration with two master systems (Page 1345)

Overview for configuring distributed I/O

Step	Description
1	Create a DP master system (Page 1346).
2	If needed configure the interfaces of the DP master system (Page 1349).

Step	Description
3	Add a DP slave and configure it (Page 1351).
4	If necessary, assign DP slaves to a SYNC/FREEZE group (Page 1354).

See also

Configuring DPV1 devices (Page 1385)

Configuring direct data exchange (Page 1376)

The basics of configuring a DP master system

Distributed I/O

DP master systems that consist of a DP master and DP slaves which are connected via a bus and communicate with one another via the PROFIBUS DP protocol are referred to as distributed I/O.

Configuring distributed I/O

The fact that the type of device may vary as far as DP masters/DP slaves are concerned means that these instructions deal only with the basic configuration procedure. However, the process for configuring distributed I/O is practically identical to that of non-distributed configuration.

Creating the DP master system in the network view

After you have used dragged a DP master and a DP slave (for example, a CPU 315--2 and an IM 153--1) from the hardware catalog to the network view, connect them both to a PROFIBUS subnet.

As the network view only allows you to configure devices (not modules), you will need to switch to the device view in order to configure the DP master and slaves.

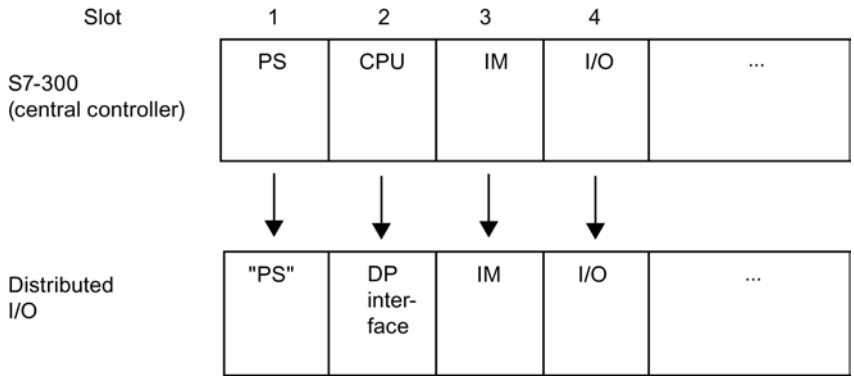
Configuring a DP slave

Switch to the device view to configure the DP slave. In the device view, populate the DP slave rack with modules and parameterize DP identifiers and I/O addresses. You can check and edit the settings in the device overview and in the inspector window.

Slot numbering for distributed I/O

Depending on the type of DP slave used, you should start the DP slave slot numbers at either "0" or "4".

The numbering of DP slave slots, on an ET 200M for example, is based on the structure for an S7-300:



DP slave slot rules in accordance with the S7-300 structure

The following slot rules apply to DP slaves that are structured in accordance with the S7-300 scheme:

- The actual I/O (inputs/outputs) always start at slot 4.
- Slot 1 is always reserved for a power supply module (PS) even if the actual configuration does not happen to include a PS.
- Slot 2 is always reserved for the DP interface.
- Slot 3 is always reserved for an extension interface module (IM) regardless of whether or not the actual I/O is capable of being expanded.

This scheme applies to all modular and compact DP slave types that are structured in accordance with the S7-300 model. The manner in which the slots are assigned is significant from the point of view of evaluating diagnostics messages ("Slot triggering diagnostics").

Additional information

Please note the additional information relating to the scope of available functions, access methods, etc. that can be found in the manuals for the relevant devices or in the online help for specific FCs (for example, DP-SEND and DP-RECEIVE for CP 342-5).

DP slaves within the hardware catalog

DP slaves in the hardware catalog

You will find the DP slaves in the "Distributed I/O" folder of the hardware catalog. Compact and modular slaves are located there:

- Compact DP slaves
  Modules with integrated digital/analog inputs and outputs, for example, ET 200L
- Modular DP slaves
  (Interface modules with S7 modules assigned, for example, ET 200M

The available DP master and the desired functionality will determine which DP slaves can be used.

I-slaves within the hardware catalog

You will find devices that you can configure as intelligent DP slaves in the "PLC" folder of the hardware catalog. The following are examples of devices that you can configure as I-slaves:

- CP 342-5 DP
  in the hardware catalog under "PLC > SIMATIC S7-300 > CP > PROFIBUS".
- CPU 315-2 DP, CPU 317-2 DP, CPU 319-3 PN/DP
  in the hardware catalog under "PLC > SIMATIC S7-300 > CPU".
- IM 151-7 CPU
  in the hardware catalog under "PLC > SIMATIC ET200 PLC > ET 200S > CPU".

See also

Browsing the hardware catalog (Page 497)

DP/DP coupler in the hardware catalog

Introduction

A DP/DP coupler connects two PROFIBUS DP networks as a gateway so that the DP master from one network can transfer data to the DP master of the other network.

The maximum amount of data that can be transferred is 244 bytes input data and 244 bytes output data.

DP/DP coupler in the hardware catalog

Details of a DP/DP coupler as gateway between two DM master systems are contained in the hardware catalog in the folder "Other field devices > PROFIBUS-DP > Gateways".

Configuring the DP/DP coupler

DP/DP couplers are configured in both PROFIBUS networks, each in their own master systems.

The input and output areas of both networks must thereby be matched to one another. The output data from one end of the DP/DP coupler are accepted as input data at the other respective end and vice versa.

See also

Installing the GSD file (Page 1382)

Creating a DP master system (Page 1346)

Configurations involving PROFIBUS DP

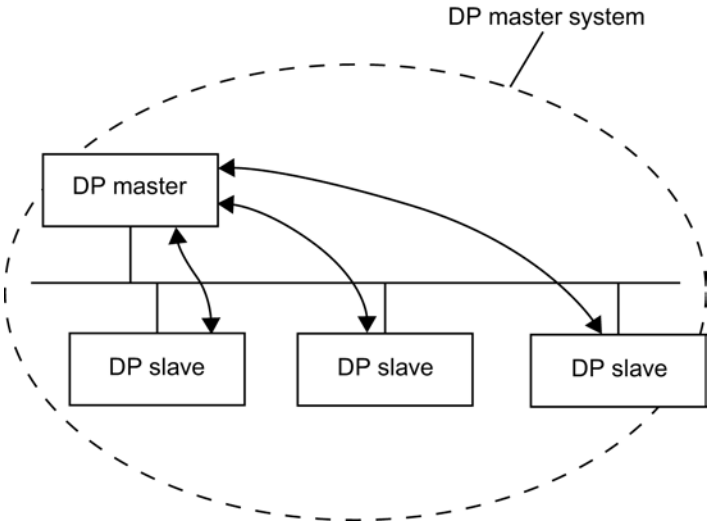
Configurations involving "simple" DP slaves

Communication between DP master and DP slave

In the case of a configuration involving simple DP slaves, data is exchanged between the DP master and simple DP slaves, i.e. with I/O modules via the DP master. The DP master polls each of the configured DP slaves in its polling list within the DP master system in succession and sends the output data/receives the input values from the slaves.

Mono-master system

The configuration with only one DP master is also described as mono-master system. A single DP master with its associated DP slaves is connected to a physical PROFIBUS DP subnet.



Configurations involving several DP master systems

Configuration methods involving two or more DP master systems

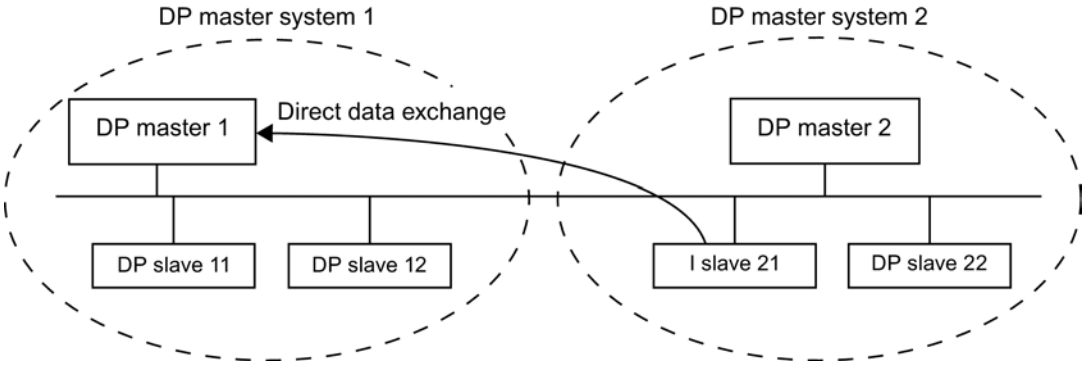
When more than one DP master system is connected to a physical PROFIBUS DP subnet it is sometimes called a multi-master system.

You can use various methods to connect these DP master systems via PROFIBUS DP:

- Without logical connection of the DP master systems to each other
- Direct DP slave > DP master data exchange
- Direct DP slave > I-slave data exchange

Direct DP slave > DP master data exchange

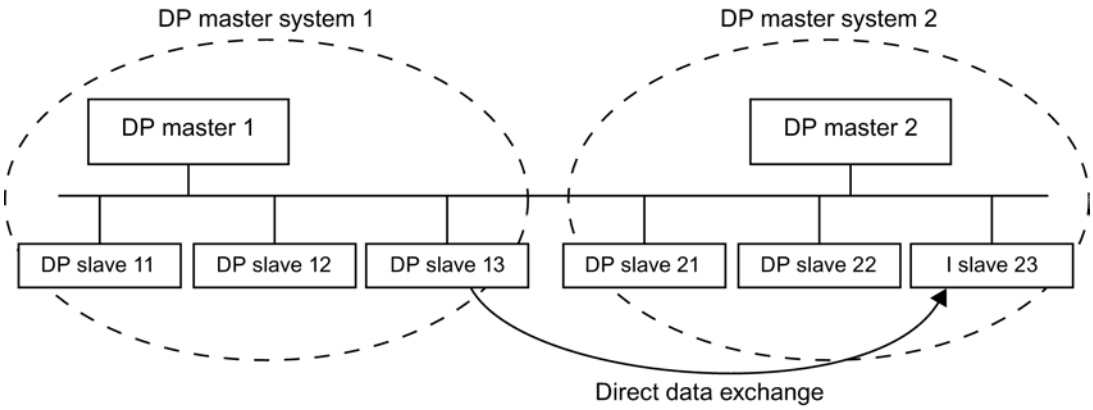
For the configuration of multi-master systems, input data of DP slaves of DP masters can be read on the same physical PROFIBUS DP subnet.



Direct DP slave > I-slave data exchange

This means that an intelligent DP slave (I-slave), such as a CPU 315-2 DP, can support the direct transfer of input data from DP slaves with different DP master systems to its input data area.

In principle, DP slaves of a certain version or higher can make selected input data available for the purpose of direct data exchange. However, only intelligent DP slaves are able to make further use of this input data.



See also

What you need to know about direct data exchange (Page 1375)

Configuring direct data exchange (Page 1376)

Configurations involving intelligent DP slaves

Definition

DP slaves that feature their own preprocessing program are referred to as intelligent DP slaves (I-slaves). Examples of intelligent DP slaves include:

- CPU 315-2 DP
- CPU 317-2 DP
- CPU 319-3 PN/DP

Configurations involving intelligent DP slaves

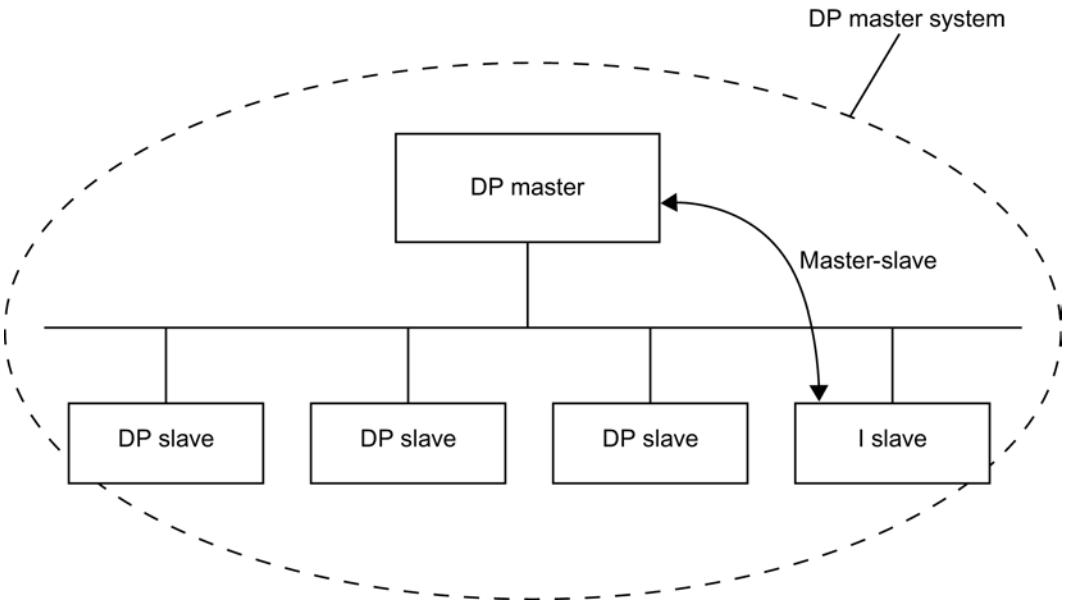
There are two methods for configuring intelligent DP slaves using PROFIBUS DP:

- I-slave <> DP master data exchange
- Direct DP slave > I-slave data exchange

I-slave <> DP master data exchange

A higher-level automation system processes the automation task, which is broken down into sub-tasks. The necessary control tasks are processed separately and efficiently in a CPU as preprocessing programs. This CPU can be implemented in the form of an intelligent DP slave.

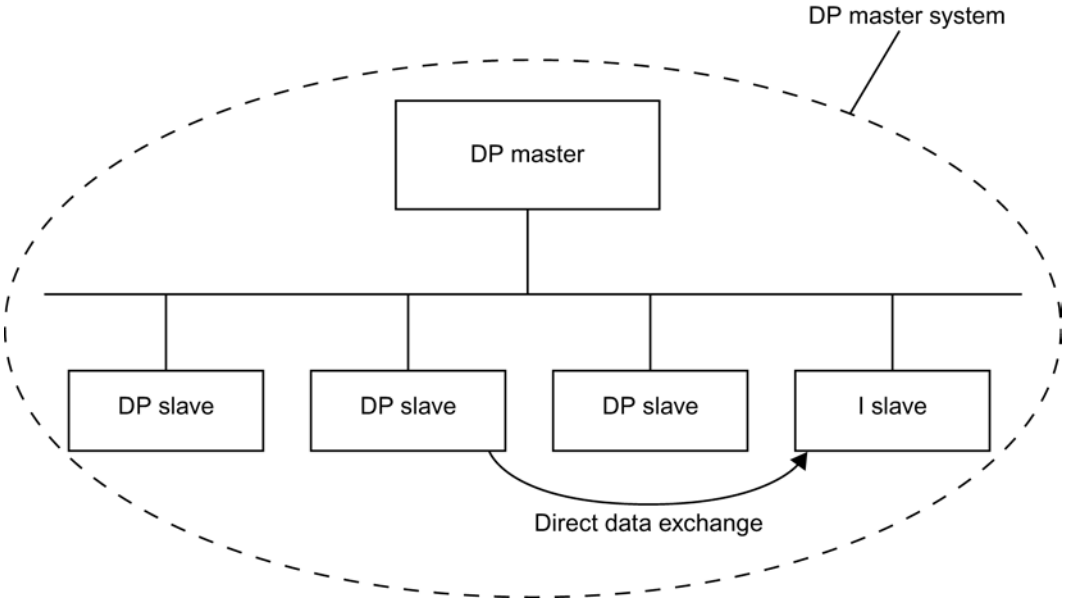
In the case of configurations involving intelligent DP slaves, for example CPU 315-2 DP, the DP master merely accesses the address area of the I-slave CPU, rather than the I/O modules of the intelligent DP slave. The address area must not be assigned to any actual I/O modules on the I-slave. The assignment must be made during I-slave configuration.



Direct DP slave > I-slave data exchange

With this configuration, input data from simple DP slaves can be transferred to intelligent DP slaves on the PROFIBUS DP subnet at high speed.

In principle, all simple DP slaves (of a certain version or higher) and intelligent DP slaves can make their input data available for the purpose of direct data exchange. Intelligent DP slaves can be used as recipients of this data.



See also

Adding an I-slave to a DP master system (Page 1367)

Example of how to configure a DP slave as an I slave (Page 1374)

What you need to know about direct data exchange (Page 1375)

Configuring distributed I/O systems

Creating a DP master system

Introduction

To create a DP master system, you need to have one DP master and at least one DP slave. As soon as you connect a DP master to a DP slave, a master-slave link is established.

DP master

You can use any of the following devices as a DP master:

- CPU with a permanently integrated or plug-in DP master interface
- CP in conjunction with a CPU
- Interface module assigned to a CPU/FM
- Interface module with a DP master interface

The DP master must have a DP interface. With some devices, you can switch between modes (such as "master" and "slave") under "Mode" in the inspector window of the DP interface. and then setting the required mode by selecting it from the drop-down list box.

Requirement

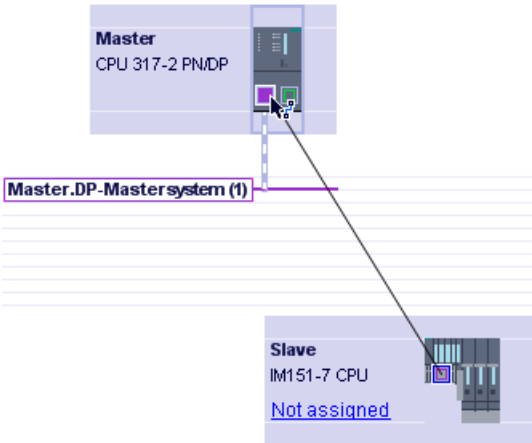
- You must be in the network view.
- The hardware catalog is open.

Procedure

To create a DP master system, follow these steps:

1. Select a DP master from the hardware catalog.
2. Pull the DP master onto the free area within the network view.
3. Right-click on the DP master's DP interface.
4. Select "Create master system" from the shortcut menu.

A DP master system with one DP master will be created as a single node. If you connect a DP slave's DP interface to that of the DP master, the DP slave will be added to the master system.



Assuming that you have already placed both a DP master and a DP slave within the network view, you can drag-and-drop to connect the two and thereby create a DP master system. To do this, follow these steps:

1. Click the DP interface of either the DP master or DP slave.
2. Hold down the mouse button and draw a connecting line between this DP interface and that of the desired communications partner.

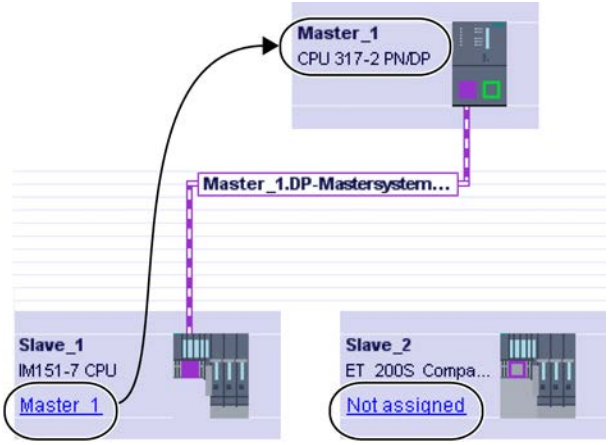
This will create a subnet with one DP master system between the DP master and DP slave.

Note

If you use the mouse to draw a connecting line between the MPI interface of a DP master and the PROFIBUS interface of a DP slave, the MPI interface of the DP master will automatically be configured as a PROFIBUS interface and a DP master system will be created.

DP master display on the DP slave

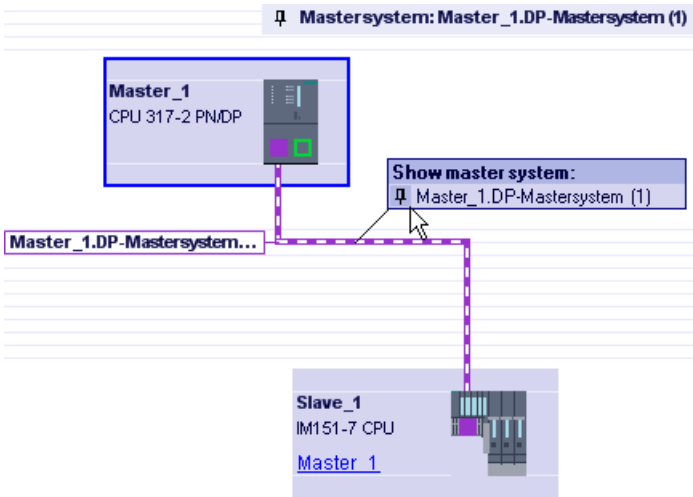
When you connect a DP slave head module (for example an ET 200S head module) to a DP master, the name of the DP master will be displayed on the slave as a hyperlink. Click the hyperlink to select the associated DP master.



Highlighting applied to the DP master system

When you create a new DP master system, highlighting will be applied to it. This enables you to identify quickly which devices belong to the DP master system. You can also highlight a DP master system yourself by moving the mouse pointer over a subnet. This will result in the

names of the available DP master systems being displayed. Click the required DP master system to highlight it.



There are various ways of removing the highlighting from a DP master system:

- Highlight a different master system.
- Click the drawing pin with the name of the master system in the top right-hand corner of the network view.

See also

DP/DP coupler in the hardware catalog (Page 1342)

The basics of configuring a DP master system (Page 1340)

Editing DP master systems and interfaces

Introduction

Once you have created a DP master system, you also have the option of disconnecting the DP master system and its components. This can result in subnets with DP slaves but no DP master.

Generally, there is no need to edit the interfaces of a DP master.

You can change the name and number on the DP master system.

Disconnection from DP master systems

If you have configured either a CPU with an integrated PROFIBUS DP interface or a PROFIBUS CP (that can be configured as an intelligent DP slave) as a DP master with master system, then you will be able to disconnect the DP master from the DP master system. After this, the device will no longer be connected to the DP master system.

Disconnecting the subnet for a DP master effectively eliminates the master system in the sense that it is no longer assigned to a DP master. However, the individual DP slaves remain

interconnected via the subnet and any direct data exchange that has been configured remains unaffected.

If you delete the DP slaves or detach them from the master system, the master system is retained on the DP master.

Requirement

- You must be in the network view.
- There has to be a DP master system with one DP master and at least one DP slave.

Disconnecting the DP master from the DP master system

To disconnect the DP master system, proceed as follows:

1. Right-click on the DP master's DP interface.
2. Select "Disconnect from master system" from the shortcut menu.

The selected DP master will be disconnected from the DP master system. A subnet with the DP slaves will be retained.

Adding a DP master to the DP master system

To reassign a DP master to a subnet, proceed as follows:

1. Right-click on the DP master's DP interface.
2. Select "Create master system" from the shortcut menu.
3. Draw connecting lines between the new DP master system and the DP interfaces of the DP slaves.

The DP master combines with the DP slaves to recreate a DP master system.

Ensuring the online capability of PROFIBUS DP interfaces

To enable a CPU's integrated MPI/DP interface to actively participate in PROFIBUS DP communication, proceed as follows:

1. Select the CPU interface symbol in the device or network views.
2. In the inspector window, select the parameter "Interface type: PROFIBUS" under "MPI address"..
3. If necessary change the suggested PROFIBUS address.
4. If required and available, select an existing subnet under "Subnet".

Note

If you use the mouse to draw a connecting line from the PROFIBUS interface of a DP slave to the MPI interface of a DP master, the MPI interface of the DP master will automatically be configured as a PROFIBUS interface and a DP master system will be created.

Editing the properties of a DP master system

To edit the properties of a DP master system, proceed as follows:

1. Move the mouse pointer over a subnet with a DP master system.
2. A message will appear displaying the available DP master systems. Click the one you want to edit. The DP master system will now be color-highlighted.
3. Now click the DP master system that has been color-highlighted.
4. In the inspector window, edit the DP master system attributes under "Properties > General".

Note

If you click a subnet when no DP master system is highlighted, you will be able to edit the properties of the entire subnet under "Properties" in the inspector window.

Adding DP slaves to the master system and configuring them

In the network view, you can add various DP slaves from the hardware catalog directly by using the drag-and-drop function or by double-clicking.

Types of DP slave

For configuration purposes, DP slaves are broken down into the following categories:

- Compact DP slaves
  (Modules with integrated digital/analog inputs and outputs, for example, ET 200L)
- Modular DP slaves
  (interface modules with S5 or S7 modules assigned, for example ET 200M)
- Intelligent DP slaves (I-slaves)
  (SIMATIC S7-300 with, for example, CP 342-5, CPU 315-2 DP, or ET 200S with IM 151-7 CPU)

Slot rules

- Your DP master system should only contain one DP master, but it may contain one or or more DP slaves.
- Your DP master system must not exceed the number of DP slaves specified on the DP master.

Note

When configuring the DP master system, remember to observe the DP master technical data (max. number of nodes, max. number of slots, max. quantity of user data). Slot or user data restrictions may prevent you from being able to use the maximum number of nodes that is theoretically possible.

Requirement

- You must be in the network view.
- A DP master system must have been created.

Adding a DP slave to the DP master system

To add a DP slave from the hardware catalog to the DP master system, follow these steps:

1. Select a modular DP slave (for example ET 200M) from the hardware catalog.
2. Drag-and-drop the DP slave from the hardware catalog into the network view.
3. Draw a connecting line between the DP master's DP interface or a highlighted DP master system and the DP interface of the new DP slave.

A DP master system will be generated and the DP slave will be connected to the DP master automatically.

Note

When a DP master system is highlighted, you can double-click the required DP slave in the hardware catalog. This will result in the DP slave being added to the highlighted DP master system automatically.

Disconnecting a DP slave from the DP master system

To disconnect a DP slave from the DP master system, follow these steps:

1. In the network view, right-click on the DP slave's DP interface.
2. From the shortcut menu, select the method for disconnecting from the DP master system:
   - "Disconnect from subnet": The PROFIBUS connection is broken and the device is no longer connected to the DP master system or a subnet.
   - "Disconnect from master system": The DP slave remains connected to the subnet, but is no longer assigned to the DP master system as a DP slave.

The selected DP slave will be disconnected from the DP master system.

Assigning a DP slave to a new DP master system

To assign an existing DP slave to a new DP master system, follow these steps:

1. Right-click on the DP slave's DP interface.
2. From the shortcut menu, select "Assign to new master". It does not matter whether the DP slave concerned is already assigned to another DP master system.
3. You will be presented with a list of existing DP masters in the form of a selection window. Select the DP master whose DP master system is to have the DP slave connected to it.

The selected DP slave has now been assigned to a new DP master system. The "Assign to new subnet" function works in a similar way in that it allows you to connect a DP slave to a

new subnet. However, in this case, the DP slave will not be connected to an existing DP master system.

Configuring a DP slave

To configure a DP slave, follow these steps:

1. Switch to the DP slave's device view.
2. Select the desired modules from the hardware catalog. In the case of modular DP slaves, the possible modules can be found within the hardware catalog underneath the relevant DP slave "family". You should, therefore, use the hardware catalog's filter function to facilitate the process of locating them.
3. Drag-and-drop the required modules from the hardware catalog onto the rack of the DP slave.
4. Configuring the DP slave and inserted modules in the inspector window.

See also

Creating a DP master system (Page 1346)

DP slaves within the hardware catalog (Page 1341)

Browsing the hardware catalog (Page 497)

Working with racks (Page 498)

Inserting a module into a rack (Page 507)

Hint: Quick configuration of master systems

If the DP master system has several DP slaves, use drag-and-drop operation to assign to the master in one step all DP slaves that were placed.

Requirement

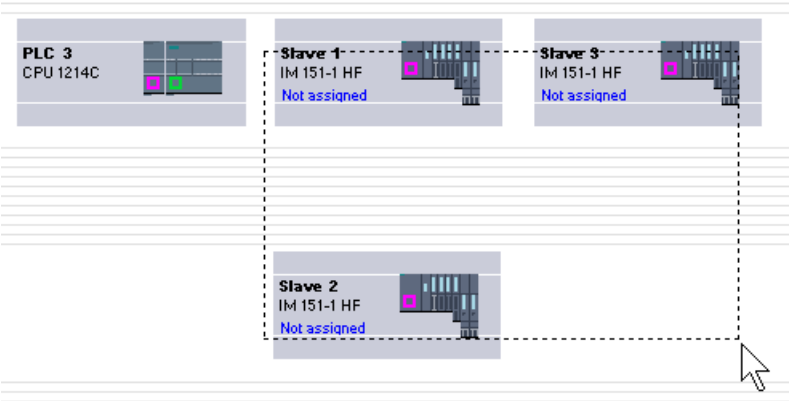
DP master and DP slaves are placed in the network view.

Assigning DP slaves to a DP master system

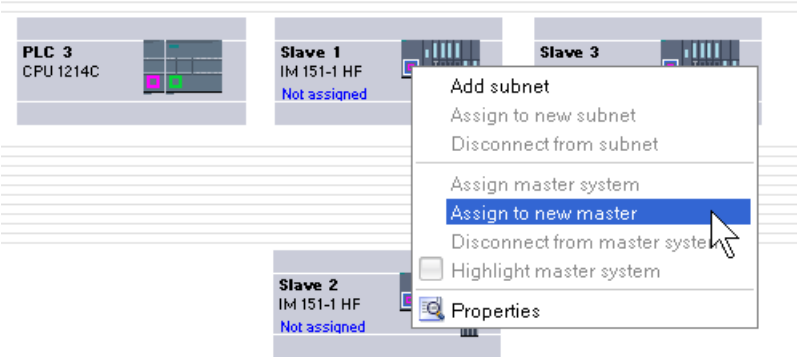
To do this, follow these steps:

1. Select an appropriate zoom factor so that you can see as many DP slaves as possible in the network view.
2. Arrange the DP slaves in a maximum of two rows.

3. Select all DP interfaces with the mouse cursor (not all devices!). This only works if you begin to drag the mouse cursor outside of the first DP slave and release the mouse button at the last DP slave (selection with the lasso).



4. Select the shortcut menu "Assign to new master" and select the corresponding DP interface for the DP master in the subsequent dialog.



5. The DP slaves are automatically networked with the DP master and combine with it to form a DP master system.

Note

When a DP master system is highlighted, you can double-click on a DP slave in the hardware catalog and thereby quickly add additional DP slaves. This will result in the DP slave being added to the highlighted DP master system automatically.

Assigning a DP slave to a SYNC/FREEZE group

Introduction

Assuming that it supports this function, a DP master can send the SYNC and/or FREEZE control commands to a group of DP slaves simultaneously in order to synchronize them. If you require this, you will need to assign the DP slaves to SYNC/FREEZE groups.

Requirement

A DP master system must have been created.

Procedure

To assign a DP slave to a SYNC/FREEZE group, follow these steps:

1. In the device or network views, select the DP slave interface that you want to assign to a group.
2. In the inspector window under "Sync/Freeze" group, select the check box for the Sync/Freeze group you want.

Note

You can assign a DP slave to a maximum of one SYNC/FREEZE group.

Exception: If you are using a CP 342-5 as the DP master, you can assign up to eight SYNC/FREEZE groups to each assigned DP slave.

Please also note the documentation relating to the CP 342-5.

What you need to know about the SYNC and FREEZE control commands

Using the SYNC and FREEZE control commands on the DP master, you can synchronize the DP slaves on an event-driven basis. The DP master simultaneously sends the control commands to all the DP slaves that make up a group within its master system. Any DP slaves that have failed or currently have diagnostics pending will be ignored.

Before DP slaves can be synchronized using control commands, you must first assign the DP slaves to SYNC/FREEZE groups.

For an S7 CPU, use the instruction DPSYC_FR (SFC 11) to synchronize the DP slaves.

When you select the DP master, a list of the eight SYNC/FREEZE groups is displayed in the Inspector window under "Properties > DP interface > SYNC/FREEZE" or "Properties> MPI/DP interface > SYNC/FREEZE" (when operating as a PROFIBUS interface).

Note

There are DP slaves which are only SYNC-capable or only FREEZE-capable. Refer to the "SYNC/FREEZE" area in the DP interface properties to find out which of these functions a DP slave supports.

In order to be able to assign to such a DP slave of a SYNC or FREEZE group, you need to restrict the affected group to the function that the DP slave supports in the DP master interface properties.

Example: For a slave that is only FREEZE-capable, you need to clear the "SYNC" checkbox in the SYNC/FREEZE group of the DP master.

The SYNC control command

The DP master sends the SYNC command to freeze the output states on a group of DP slaves so that they retain the current value.

When subsequent message frames arrive, the DP slaves will store the output data from the DP master. However, the DP slave output states will remain unchanged.

Then, when a new SYNC control command is sent, each DP slave will align its outputs with the values of the output data from the DP master.

Cyclic updating of the outputs will only resume when the DP master sends the UNSYNC control command.

The FREEZE control command

When they receive the FREEZE control command from the DP master, the DP slaves within the relevant group will freeze the current state of their inputs The DP slaves send this frozen input data to the DP master in the following cyclic frames.

Whenever they receive a new FREEZE control command, the DP slaves will freeze the current state of their inputs again.

Cyclic transfer of the state of the DP slave inputs from the DP slave to the DP master will only resume when the DP master sends the UNFREEZE control command.

See also

Creating a DP master system (Page 1346)

Setting PROFIBUS properties (Page 1406)

Example: Configuring ET 200S as DP slave

Configuring an ET 200S

Introduction

DP slaves and I/O devices from the ET 200S family are configured in the same way as other modular DP slaves and I/O devices. The modules for the ET 200S are available in the hardware catalog under "Distributed I/O".

Slot rules for configuring an ET 200S

The following rules apply when configuring an ET 200S:

- Do not leave any gaps when inserting the ET 200S modules.
- Slot 1: only for PM-E or PM-D Power Modules.
- To the left of an Electronics Module (EM): an EM or a Power Module (PM-E or PM-D) only.
- To the left of Motor Starter (MS): an MS, a PM-D, PM-D Fx (1..x..4) Power Module or a PM-X Power Module only.

- To the left of a PM-X: a motor starter or a PM-D only
- Up to 63 modules and one IM Interface Module are permitted

Note

Remember to ensure that the correct PM-E and EM voltage ranges are assigned.

Operation when expected and actual configurations do not match

This parameter controls the response during startup, and when modules are removed and inserted during operation (cyclical data exchange).

Check box is selected (operation is permitted if preset configuration is not the same as actual configuration):

- The module will start up even if the preset configuration is not equal to the actual configuration.
- The module remains in cyclical data exchange following removal and insertion (not a CPU stop).

Check box is cleared (operation is not permitted if preset configuration is not the same as actual configuration):

- The module will not start up (does not switch over to cyclical data exchange) if the preset configuration is different from the actual configuration.
- If modules are removed and inserted during operation, this causes a CPU stop, and the module no longer participates in the cyclical data exchange.

Refer to the manual for the respective module for information on which constraints cause a CPU stop, or prevent the data exchange starting, even if operation is enabled when preset configuration is not the same as the actual configuration.

Interference frequency suppression

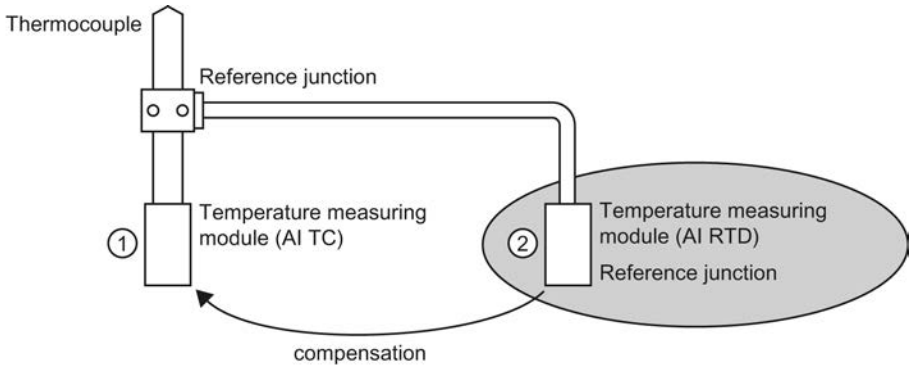
The frequency of your alternating current network can interfere with the measured value in the following cases in particular: measurement in small voltage ranges and measurement with thermocouples. Set the frequency of the power for the entire ET 200S to that which predominates in your plant.

The analog electronic modules AI set their integration time or conversion time in relation to the selected interference frequency suppression.

Configuring reference junctions

A reference junction is the connection of a thermocouple to a supply line (generally in the terminal box). The voltage that occurs due to the effects of temperature falsifies the temperature value measured by the module.

On the ET 200S, one channel of the AI RTD module can be programmed as a reference junction. Other AI TC modules can correct their measured values using the temperature at the reference junction as measured by this module.



① Configuring the AI TC:
→ Selection of the reference junction used

② Configuring of the AI RTD:
→ Activation of the reference junction
→ Specifying the slot and channel of the AI RTD

Special characteristics to be noted when assigning parameters for reference junctions

The process of assigning parameters for reference junctions will be explained based on the example of a resistance thermometer with a Pt 100 climatic range that is used for measuring the reference junction temperature.

To assign parameters for the reference junction, follow these steps:

1. In the ET 200S device view, insert an analog RTD electronics module, for example a 2AI RTD HF. The module must support the "thermal resistor" measuring type (linear, 4-wire) and the "Pt 100 climatic range" measuring range.
2. Select the module on the rack.
3. Under "Properties > General > Inputs" in the inspector window, set a channel for the reference junctions function.
   Measuring type: Thermal resistor (linear, 4-wire)
   Measuring range: Pt 100 climatic range
4. Select the ET 200S (that is, the interface module).
5. Under "Properties > General > Module parameters >Reference junctions" in the inspector window, select the "Reference junction" check box and specify the slot and the channel number of the relevant RTD module.
6. Insert the analog electronics module for measuring the temperature using a thermocouple (TC module) and select the option "RTD" as a reference junction under "Properties > General > Inputs > Channel x" in the inspector window.

Additional information

For additional information on the various types and uses of ET 200S modules, please refer to the operating instructions and the manual titled "ET 200S Distributed I/O System".

For additional information on analog value processing, please see the documentation for the ET 200S distributed I/O system.

See also

Configuring the ET 200S in DPV1 mode (Page 1365)

Documentation on ET 200S (http://support.automation.siemens.com/WW/view/en/1144348)

Packing addresses

Introduction

DP slaves and I/O devices from the ET 200S family are configured in the same way as other modular DP slaves and I/O devices. As well as supporting all the standard modular DP slave and I/O device functions, the ET 200S also offers the "Pack addresses" function:

When digital electronics modules requiring an address space of 2 or 4 bits are inserted into the device view, they will initially be spread over a total area of 1 byte. However, the address area actually occupied can be compressed after configuration using the "Pack addresses" function.

	Initial state	After "Pack addresses"
Module	I address	I address
2DI (2-bit)	Byte 10	10.0 to 10.1
4DI (4-bit)	Byte 11	10.2 to 10.5

Requirement

- You are in the device view.
- An ET 200S, for example an IM 151-1, must be present.
- A pair of digital electronics modules, for example 2DI AC120V ST, must be inserted into the slots.

Packing addresses

To pack addressed, follow these steps:

1. Select the electronics modules whose addresses are to be packed. The following options are available for selecting multiple electronics modules:
   - Press and hold down <Shift> or <Ctrl> while clicking the relevant electronics modules.
   - Click off the rack and select the required electronics modules by drawing round them with the mouse.
2. Click "Pack addresses" in the shortcut menu for the selected electronics modules.

The address areas for inputs, outputs and motor starters are packed separately. The packed addresses will be displayed in the I address and Q address columns of the device overview.

How packed addresses are generated and structured

If you make use of the "Pack addresses" function, the addresses of the selected electronics modules will be packed in accordance with the following rules:

- The start of the address area is determined by the lowest address of the selected electronics modules: X.0
- If the bit address is not "0", then the next (free) byte address as of which the selected area can be compacted will be selected automatically: (X+n).0
- If no coherent area exists, the addresses will be automatically packed into any available address gaps.

Electronics modules with packed addresses and the same byte address form a packing group.

Unpacking addresses

To unpack addressed, follow these steps:

1. Select one or more electronics modules with packed addresses.
2. Click "Unpack addresses" in the shortcut menu for the selected electronics modules.

The packing groups of the selected electronics modules will be disbanded and the packed addresses for the relevant electronics modules unpacked.

The packing group will also be disbanded and the packed addresses unpacked in the following cases: if you delete electronics modules from a packing group, move electronics modules out of a packing group or insert electronics modules on a free slot within a packing group.

The start addresses of the unpacked electronics modules will be assigned to the next available byte addresses in each case.

Special characteristics of electronics modules with packed addresses

The following special characteristics apply to electronics modules with packed addresses:

- As far as the CPU is concerned, there is no way of assigning a slot for the electronics module. Consequently, the instruction GADR_LGC (SFC 5) outputs error information W#16#8099 "Slot not configured" for the actual slot of the electronic module.
- The instruction LGC_GADR (SFC 49) and SZL-ID W#16#xy91 "module status information" cannot be evaluated for an electronics module.
- The electronics module receives an additional diagnostics address via the DPV1 function, because otherwise the packed addresses would prevent interrupts from being assigned as far as the CPU is concerned.
- The "Insert/remove interrupt" is not possible. This is because the "Pack addresses" and "Insert/remove interrupt" functions are mutually exclusive.

Configuring option handling with standby modules

You can use the option handling function to prepare the ET 200S with PROFIBUS interface for future expansions (options). This section describes option handling with standby modules.

You do this by assembling, wiring, configuring, and programming the maximum configuration envisaged for the ET 200S and by using cost-effective standby modules (138-4AA00 or 138-4AA10) during assembly until it becomes time to replace them with the necessary electronics modules.

Note

The ET 200S can be completely factory-wired with the master cabling, as no connection exists between a standby module and the terminals of the terminal module (and, in turn, to the process).

Requirement

- ET 200S interface module
  - IM 151-1 STANDARD (6ES7 151-1AA03-0AB0 or higher)
  - IM 151-1 FO STANDARD (6ES7 151-1AB02-0AB0 or higher)
- Power module with option handling
  - PM-E DC24..48V
  - PM-E DC24..48V/AC24..230V

Procedure

To activate option handling, follow these steps:

1. Select the IM 151-1 in the device view and enable it in "Option handling" check box under "Properties > General > Option handling" in the inspector window.
2. Select the numbered check boxes for the slots that are initially to accommodate the standby modules prior to the future electronics modules.
3. Select the power module in the device view and enable it in the "Option handling" check box under "Properties > Addresses" in the inspector window. Reserve the necessary address space for the control and check-back interface in the process image output (PIQ) and process image input (PII).

The assembled standby modules can be replaced with the configured modules at a later date without having to modify the configuration.

Note

The addresses for these interfaces are reserved as soon as you activate option handling on the power module. The "Option handling" function must also be activated on the DP slave (IM 151-1 STANDARD Interface Module). If it is not activated, the addresses reserved for the control and check-back interface will be released again.

Note that activating and deactivating the option handling function repeatedly can change the address of the control and check-back interface.

Option handling may be activated for one PM-E DC24..48V or one PM-E DC24..48V/AC24..230V Power Module only.

Additional information

For additional information on the assignment and significance of bytes within the process image, option handling with PROFIBUS and the use of standby modules, please refer to the documentation for the ET 200S distributed I/O system.

How option handling works during startup

If "Startup when expected/actual config. differ" is not available, the ET 200S will still start up if a standby module is inserted instead of the configured electronics module and option handling has been activated for the slot concerned.

How option handling works during operation

During operation, the option handling mode varies in accordance with the following:

- Option handling enabled for a slot:
  Either the standby module (option) or the configured electronics module can be plugged into this slot. If any other kind of module is present on this slot, diagnostics will be signaled (no module/incorrect module).
- Option handling disabled for a slot:
  Only the configured electronics module can be plugged into this slot. If any other kind of module is present, diagnostics will be signaled (no module/incorrect module).

Standby module substitute values

- Substitute value for digital inputs: 0
- Substitute value for analog inputs: 0x7FFF

Control and evaluation in the user program

The ET 200S is equipped with a control and feedback interface for the "Option handling" function.

The control interface is located in the process image output (PIQ). Each bit in this address area controls one of the slots from 2 to 63:

- Bit value = 0: Option handling parameterized. Standby modules are permitted.
- Bit value = 1: Option handling canceled. Standby modules are not permitted on this slot.

The check-back interface is located in the process image input (PII). Each bit in this address area provides information about the modules that are actually plugged into slots 1 to 63:

- Bit value = 0: This slot has the standby module, an incorrect module or no module plugged into it.
- Bit value = 1: The configured module is plugged into this slot.

See also

Which modules support option handling? (http://support.automation.siemens.com/WW/view/en/22564754)

Configuring option handling without standby modules

You can use the option handling function to prepare the ET 200S for future expansions (options) without installing standby modules. This section describes option handling without standby modules.

Note

ET 200S with PROFINET interface

This description refers to the ET 200S with PROFIBUS interface. In principle, option handling for ET 200S with PROFINET interface functions as described here without standby modules. PN interface modules are to be used instead of the DP interface modules listed here. You can find additional information about option handling for ET 200S with PROFINET interface in the corresponding manuals.

Requirement

- ET 200S interface module
  - IM 151-1 HIGH FEATURE (6ES7151-1BA02 or higher)
  - IM 151-1 STANDARD (6ES7 151-1AA05-0AB0 or higher)
- Power module with option handling
  - PM-E DC24..48V
  - PM-E DC24..48V/AC24..230V

Procedure

To activate option handling, follow these steps:

1. Select the IM 151-1 in the device view and enable it in "Option handling" check box under "Properties > General > Option handling" in the inspector window.
2. Select the power module in the device view and enable it in the "Option handling" check box under "Properties > Addresses" in the inspector window. Reserve the necessary address space for the control and check-back interface in the process image output (PIQ) and process image input (PII).
3. Configure the slave's maximum configuration. The selection/clearing of options is controlled via the user program.

Note

The addresses for these interfaces are reserved as soon as you activate option handling on the power module. The "Option handling" function must also be activated on the DP slave (IM 151-1 interface module). If it is not activated, the addresses reserved for the control and check-back interface will be released again.

Note that activating and deactivating the option handling function repeatedly can change the address of the control and check-back interface.

Option handling may be activated for one PM-E DC24..48V or one PM-E DC24..48V/AC24..230V Power Module only.

Additional information

For additional information on the assignment and significance of bytes within the process image, option handling with PROFIBUS and the use of standby modules, please refer to the documentation for the ET 200S distributed I/O system.

Control and evaluation in the user program

The ET 200S is equipped with a control and feedback interface for the "Option handling" function.

The control interface is located in the process image output (PIQ). Each bit in this address area controls one of the slots from 1 to 63:

- Bit value = 0: Slot is not available in the actual configuration.
- Bit value = 1: Slot is available in the actual configuration.

The check-back interface is located in the process image input (PII). Each bit in this address area provides information about the modules that are actually plugged into slots 1 to 63:

- Bit value = 0: Slot belongs to an option that is not available or module status is faulty.
- Bit value = 1: The configured module is plugged into this slot.

See also

Sample applications for ET 200S, option handling without standby module (http://support.automation.siemens.com/WW/view/en/29430270)

Configuring the ET 200S in DPV1 mode

PROFIBUS DPV1 enables you to access extended PROFIBUS functions.

Requirement

- You must be in the network view.
- An DP master with DPV1 functionality must be available.
- A master-slave connection must be established with PROFIBUS.

Procedure

To switch the DP slave over to DPV1, follow these steps:

1. Select the DP slave.
2. Under "Properties > Module parameters" in the inspector window, select "DPV1" mode from the "DP interrupt mode" drop-down list box.

or

1. Select the DP master.
2. In the I/O communications table select the row with the connection between the DP master and the desired DP slave.
3. Under "Properties > Module parameters" in the inspector window, select "DPV1" mode from the "DP interrupt mode" drop-down list box.

Special characteristics

The parameters are subject to interdependencies, which are outlined below:

Parameter	DPV0 mode	DPV1 mode
Operation when target configuration does not match actual configuration	Fully available	Fully available
Diagnostics interrupt (OB 82)	Not available, not set	Fully available
Hardware interrupt (OB 40 to 47)	Not available, not set	Fully available
Insert/remove interrupt (OB 83)	Not available, not set	Only available when addresses are not packed. "Startup when target configuration does not match actual configuration" is activated automatically along with an insert/remove interrupt.

Interrupts in the case of modules with packed addresses

If the module is capable of triggering interrupts and the bit address is not equal to 0 because of packed addresses, you will need to assign a diagnostics address in the ET 200S address dialog.

The diagnostics address is required for the purpose of assigning a DPV1 interrupt to the module as an interrupt trigger. The CPU will only be able to assign an interrupt correctly and store information on the interrupt in the interrupt OB start information/in the diagnostics buffer if the module concerned has this "unpacked" address. In this context, the CPU cannot make use of "packed" addresses.

From the point of view of interrupt processing (interrupt OB), this means the module will have the assigned diagnostics address, but from the point of view of processing the input and output data in the user program, the module will have the packed addresses.

Note

When the module addresses are packed, the insert/remove interrupt for the ET 200S is unavailable.

See also

Introduction to PROFIBUS DPV1 (Page 1384)

Configuring DPV1 devices (Page 1385)

Configuring an ET 200S (Page 1356)

Additional configurations

Configuring additional functions

The ET 200S distributed I/O system has numerous additional functions that are available via plug-in modules (e.g. communication module). You can find the descriptions for these in the sections "Using technology functions" and "Configuration for point-to-point connections".

Additional distributed I/O systems

Information on additional distributed I/O systems is available by following further links under "See also".

See also

Configuring and assigning parameters to 1Count24V (Page 9221)

Configuring and assigning parameters to 1Count5V (Page 9282)

Configuring and assigning parameters to 1SSI (Page 10430)

Configuring and assigning parameters to 1PosU (Page 10690)

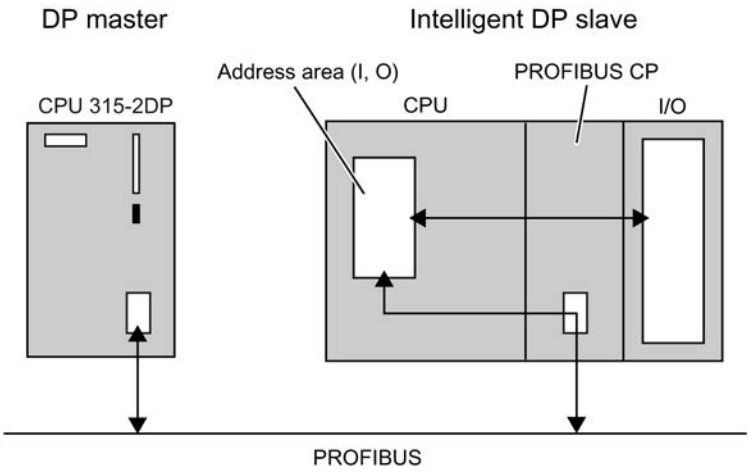
1STEP 5V configuration and parameter assignment (Page 10737)

Configuring intelligent DP slaves

Adding an I-slave to a DP master system

Introduction

One of the key characteristics of intelligent DP slaves (I-slaves) is that the DP master is not provided with I/O data directly by a real I/O, but by a preprocessing CPU. Together with the CP, this preprocessing CPU forms the I-slave.



Difference: DP slave v. intelligent DP slave

In the case of a DP slave, the DP master accesses the distributed I/O directly.

In the case of an intelligent DP slave, the DP master actually accesses a transfer area in the I/O address space of the preprocessing CPU instead of accessing the connected I/O of the intelligent DP slave. The user program running on the preprocessing CPU is responsible for ensuring data exchange between the address area and I/O.

Note

The I/O areas configured for data exchange between the DP master and DP slaves must not be used by I/O modules.

Applications

Configurations involving intelligent DP slaves:

- I-slave <> DP master data exchange
- Direct DP slave > I-slave data exchange

Procedure

To add an I-slave to a DP master system, proceed as follows:

1. From the hardware catalog, drag two devices with PROFIBUS DP interface for configuration as DP master and I-slave (e.g. CPU 317-2 DP) into the network view.
2. Switch the device that is to be used as the I-slave over to "Slave" mode under "Properties > DP interface" in the Inspector window. The device will now operate as an intelligent DP slave.
3. Draw a connecting line between the DP interfaces of both devices. This way you connect the I-slave with a DP master in a DP master system.

Result

You have now set up a DP master system with one DP master and one I-slave.

Please note the following I-slave configuration examples:

- Configuring a CP as an I-slave (Page 1372)
- Configuring a CPU as an I-slave (Page 1373)
- Configuring a DP slave as an I-slave (Page 1374)

See also

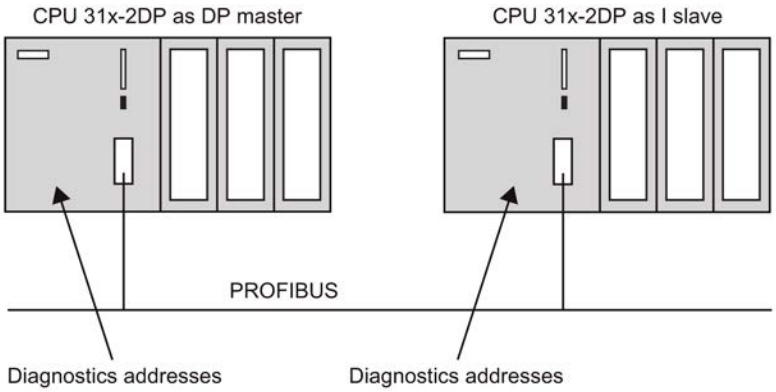
Interrupts on the I-slave (Page 1370)

Configuring diagnostics addresses (Page 1368)

Configuring diagnostics addresses

Introduction

When assigning diagnostic addresses for PROFIBUS DP, you must ensure that DP diagnostic addresses are assigned to both the DP master and the I-slave.



Diagnostic addresses

A distinction is made between diagnostic addresses on the DP master and those on the I-slave:

- Diagnostic addresses on the DP master
  The DP master receives information about the status of the DP slave or a bus interruption via the diagnostic addresses assigned to the DP master.
- Diagnostic addresses on the I-slave
  The I-slave receives information about the status of the DP master or a bus interruption via the diagnostic addresses assigned to the I-slave.

STEP 7 automatically assigns diagnostic addresses for the respective CPUs when the master-I-slave configuration is specified. The following describes how to change the specified diagnostic addresses.

Tip: In order to obtain an overview of all addresses assigned for the CPU, including diagnostic addresses, select the "Address overview" area in the inspector window of the selected CPU.

Requirement

- You must be in the network view.
- You must have set up a DP master system with a DP master and an I-slave.

Procedure

To edit the diagnostic addresses, proceed as follows:

1. Select the DP master or the I-slave.
2. In the I/O communication table, select the line showing "I-slave" mode.
3. Edit the diagnostic addresses under "Properties > I-Slave Communication" in the inspector window.

Or:

1. Select the I-slave.
2. Edit the diagnostic addresses under "Properties > DP interface > Mode > I-slave communication" in the inspector window.

You can select and edit the fields containing the values of the diagnostic addresses both for the slave and the master.

The diagnostic address section contains the addresses for the operating mode transitions and for device diagnostics:

- Operating mode transitions: transitions from RUN > STOP or from STOP > RUN
- Device diagnostics: for example, a bus interruption caused by a short circuit or unplugged connector ("station failure").

Additional information

For further information on diagnostic addresses, refer to the manuals for the individual S7 CPUs.

See also

Addressing of modules (S7-300, S7-400) (Page 880)

Interrupts on the I-slave

Types of interrupt according to DPV1 and DPV0 modes

I-slaves with the advanced instruction "SALRM" (SFB 75) can trigger interrupts on the associated DP master. The following table lists the possible types of interrupt according to the PROFIBUS DPV1 and DPV0 modes:

Type of interrupt	DPV0	DPV1
Diagnostic interrupt (OB 82)	Yes	Yes
Hardware interrupt (OB 40 to 47)	Yes	Yes
Pull/plug interrupt (OB 83)	Yes (if I-slave supports this interrupt)	Yes
Status interrupt (OB 55)	No	Yes
Update interrupt (OB 56)	No	Yes
Manufacturer-specific interrupt (OB 57)	No	Yes

Addresses for triggering interrupts

Use the addresses configured in the I-slave's inspector window under "Properties > DP interface > Mode > I-slave communication" to trigger interrupts with advanced instruction "SALRM". Rather than being assigned to physical modules, these addresses correspond to virtual "slots".

You cannot use the addresses relating to operating mode transitions and device diagnostics for the purpose of triggering interrupts.

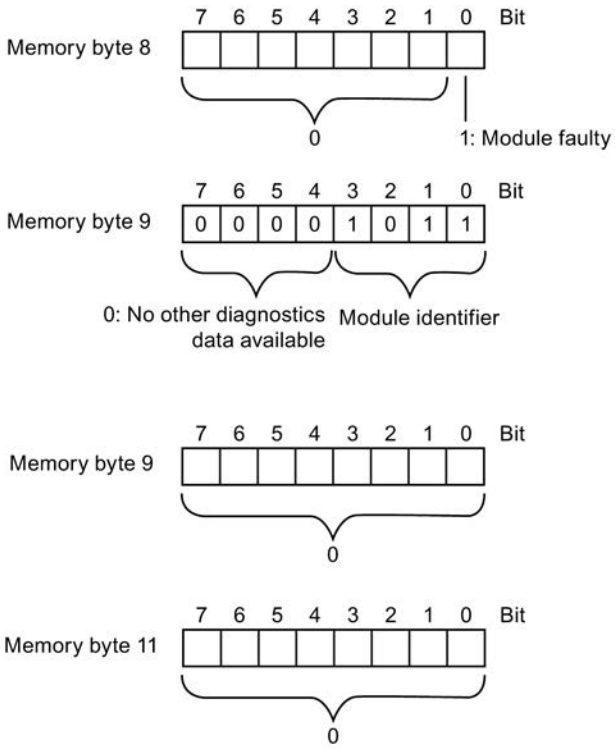
How interrupt generation works

The principle of interrupt generation can be explained using the example of a diagnostics interrupt.

- On the I-slave, the output address 0 is assigned to a virtual slot.
- In the example, the output address 0 is used to trigger a diagnostics interrupt (OB 82) on the DP master.

Diagnostics interrupts are accompanied by data relating to individual user programs (AINFO). This data must correspond to the basic structure for supplementary interrupt information.

For example, the following simplified structure is possible:



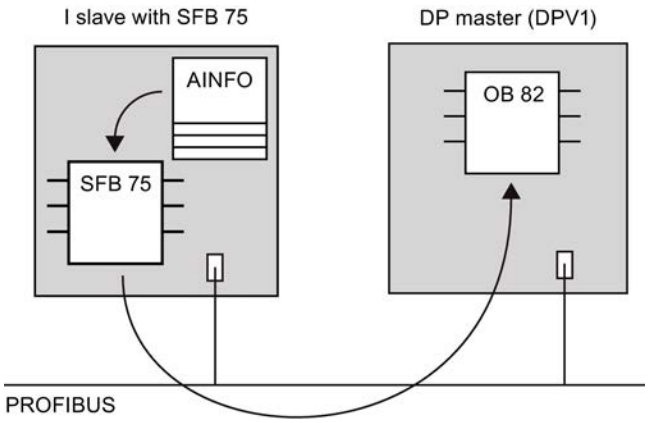
Additional information

Note the additional information on the structure presented above in the reference manual for the S7-300/400 system and standard functions, Volume 2, "Diagnostic data" section, and in the manuals for the system and diagnostic functions of the S7-1500.

Note

The supplementary interrupt information affects the module status data and the group error LED of the I-slave (S7-300/400). The interrupt also affects the module status data and error LEDs of the associated DP master. Therefore, you will need to take account of what the diagnostic data records mean (data record 0 and data record 1) when compiling data for the supplementary interrupt information.

The figure below illustrates the interrupt generation process:



The advanced instruction "SALRM" can be called as follows:

```
STL
CALL   FB     SALRM , DB75
       REQ    :=M0.0
       ID     :=DW#16#8000    //Output address 0
       ATYPE  :=1             //Diagnostics interrupt
       ASPEC  :=1             //Incoming interrupt, EG faulty
       LEN    :=4             //Interrupt info is 4 bytes in length
       DONE   :=M1.3
       BUSY   :=M1.4
       ERROR  :=M1.5
       STATUS:=MD50
       AINFO  :=P#M 8.0 BYTE 4 //Area for interrupt info
```

See also

Introduction to PROFIBUS DPV1 (Page 1384)

Configuring DPV1 devices (Page 1385)

Example of how to configure a CP as an I slave

In this example, you are going to configure a station with a CP 342-5 as an I-slave. In order to do this, you will need to turn the CP 342-5 into an intelligent DP slave by means of the "DP slave" mode.

Requirements

- You must be in the network view.
- There must be a CP 342-5 available in the network view.
- The CP 342-5 must have been populated with I/O modules in the device view.
- A DP master (CPU with integrated PROFIBUS DP interface or CP with PROFIBUS DP interface) is available in the network view.

Procedure

To configure a CP as an I-slave, proceed as follows:

1. Select the CP 342-5.
2. In the Inspector window, switch the CP 342-5 to "DP slave" mode under "Properties > Mode".
3. Draw a connecting line between the CP 342-5's DP interface and the DP interface of the DP master.
4. Select the CP 342-5.
5. Click on "I-slave communication" under "Properties > Mode" in the Inspector window.
6. Edit the address assignment between the DP master and I-slave.

Alternatively, after selecting the CP 342-5 in the I/O communications table, you can call the line with the "I-slave" mode in order to edit the address assignment under "Properties" and "I-slave communication" in the Inspector window.

Additional information

For additional information on data exchange between the preprocessing CPU and the DP slave's internal CP 342-5 DP, please refer to the manual that deals with NCM S7 Industrial Ethernet.

Example of how to configure a CPU as an I slave

In this example, you are going to configure a station with a CPU 31X-2 DP featuring an integrated DP interface (a CPU 315-2 DP, for example) as an I-slave. The CPU 31X-2 DP must have an integrated DP interface, and you will need to turn it into an intelligent DP slave by activating "DP slave" mode.

Requirements

- You must be in the network view.
- There must be a CPU 315-2 DP available in the network view.
- The CPU 315-2 DP must have been populated with I/O modules in the device view.
- A DP master (CPU with integrated PROFIBUS DP interface or CP with PROFIBUS DP interface) is available in the network view.

Procedure

To configure a CPU as an I-slave, proceed as follows:

1. Select the CPU 315-2 DP.
2. In the Inspector window, switch the CPU 315-2 DP to "DP slave" mode under "Properties > DP interface > Mode".
3. Draw a connecting line between the DP interface of the CPU 315-2 DP and the DP interface of the DP master.

4. Select the CPU 315-2 DP.
5. Click on "I-slave communication" under "Properties > DP interface > Mode" in the Inspector window.
6. Edit the address assignment between the DP master and I-slave.

Alternatively, after selecting the CPU 315-2 DP in the I/O communications table, you can call the line with the "I-slave" mode in order to edit the address assignment under "Properties > I-slave communication" in the Inspector window.

Additional information

For additional information on configuring CPUs as I-slaves, please see the reference manuals for the relevant CPUs.

Example of how to configure a DP slave as an I slave

In this example, you are going to configure an ET 200S with Interface Module IM 151-7 CPU as an I-slave.

Note

You can find the interface module of the ET 200S in the hardware catalog under "PLC > SIMATIC ET 200 PLC > ET 200S > CPU".

Requirements

- You must be in the network view.
- An ET 200S with IM 151-7 CPU is available in the network view.
- The ET 200S has been fitted with I/O expansion modules in the device view.
- A DP master (CPU with integrated PROFIBUS DP interface or CP with PROFIBUS DP interface) is available in the network view.

Procedure

To configure a DP slave as an I-slave, proceed as follows:

1. Select the IM 151-7 CPU.
2. Draw a connecting line between the DP interface of the IM 151-7 CPU and the DP interface of the DP master.

   Note

   Some stations have permanently specified interfaces for master or slave. Under "Properties > MPI/DP interface" and/or "Properties > DP interface" you will find the settings for the respective mode.

3. Select the IM 151-7 CPU.

4. In the I/O communications table, select the line showing "I-slave" mode.
5. Edit the address assignment between the DP master and I-slave under "Properties > I-slave communication" in the Inspector window.

Additional information

For additional information on configuring DP slaves as I-slaves, please refer to the manuals and operating instructions for the ET 200 distributed I/O systems.

Configuring direct data exchange

What you need to know about direct data exchange

Introduction

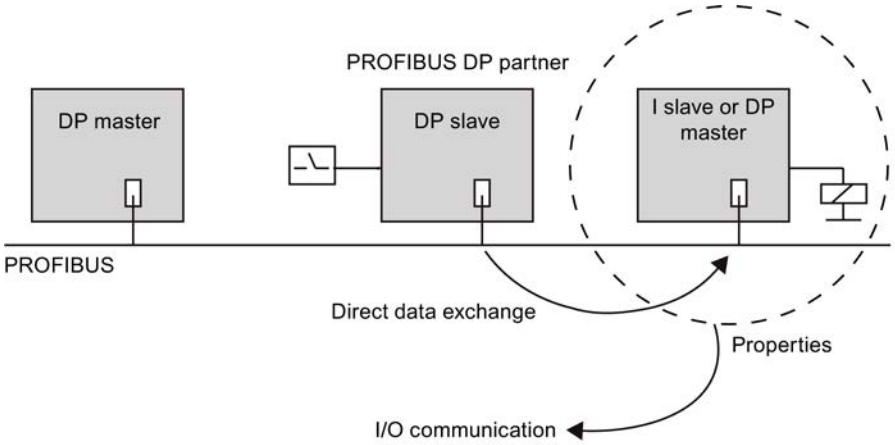
Direct data exchange (data exchange broadcast) is another form of I/O communication, which is not restricted to the context of a DP master system. With direct data exchange, the receiver has read-only access (in other words it cannot write data).

In a DP master system, a DP master only controls the DP slaves that have been assigned to it. With direct data exchange, a DP master or a DP slave configured as a receiver on the PROFIBUS subnet can "listen in" to find out what input data another DP slave is sending to its DP master in a sender capacity. This method of transferring data between DP slaves is referred to as direct data exchange.

Principle of operation

In a direct data exchange configuration, local input address areas of an intelligent DP slave or of a DP master are assigned to the input address areas of a PROFIBUS DP partner.

The intelligent DP slave or DP master can then use these input address area assignments to receive data from its PROFIBUS DP partner.



Configuration options

Possible applications of direct data exchange include:

- A configuration involving intelligent DP slaves: DP slave > I slave
- A configuration involving two DP master systems: DP slave > DP master

Number of PROFIBUS DP partners that can be connected

The total number of PROFIBUS DP partners that can be connected directly to a DP interface or that can be addressed via this interface using direct data exchange depends on the specific interface concerned. A maximum of 32 PROFIBUS DP partners can be addressed on the MPI/DP interface.

See also

Configurations involving several DP master systems (Page 1343)

Configurations involving intelligent DP slaves (Page 1345)

Configuring direct data exchange

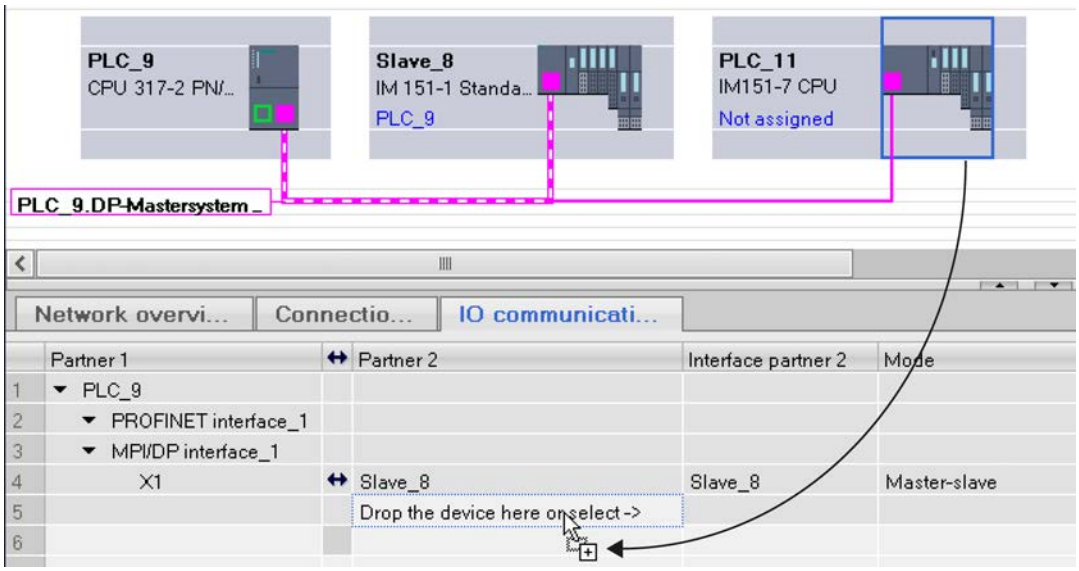
Requirement

- You must be in the network view.
- A DP master system with input modules is available on the DP slave.
- A DP master or DP slave is linked with the DP master system as a "receiver".

Setting up direct data exchange

To set up direct data exchange, proceed as follows:

1. Click on the required communications partner (sender or receiver) to select it.
2. Press and hold down the left mouse button and then drag the second communications partner into the "Place device here" field, or select the "Partner 2" column of the I/O communications table.



Alternatively, click directly on the "Place device here..." and select the connection partner you want from the drop-down list box.

Using drag-and-drop, you have now managed to set up a relationship in the table between two nodes for the purpose of communication. In a new line of the I/O communications table, "Direct data exchange" will now be displayed.

Note

The sender is always selected first in the case of a direct data exchange between two I-slaves. Subsequently, the receiver is dragged to the "Place device here or select" field or select from the drop-down list box. The communication direction of the direct data exchange determines the order in this case.

Configuring direct data exchange

To configure direct data exchange, proceed as follows:

1. Select a direct data exchange partner.
2. In the I/O communications table, select the line showing "Direct data exchange" mode.
3. Select "Properties > Direct data exchange" in the Inspector window.

4. Create a new transfer area under "Direct data exchange" in the transfer area table:
   In the "Transfer area" column, click on "<Add new>" and select the "DX" communication type for direct data exchange in the "Type" drop-down list box. You can also create the new transfer area by setting the communication type in a new column only under "Type". The transfer area is then created automatically.
5. Under "Direct data exchange", click on the newly created transfer area.
   The detail view of the transfer area is opened.
6. Edit the properties for the sender and receiver and the general parameters of the direct data exchange.

You have now configured direct data exchange.

Deleting direct data exchange

To delete the direct data exchange, proceed as follows:

1. In the I/O communications table, select the line showing "Direct data exchange" mode.
2. Press "Del" or select the "Delete" command from the shortcut menu.

Example of how to configure direct data exchange

Introduction

In this example, you are going to configure three CPUs in a master-slave configuration for direct data exchange. You are then going to assign meaningful names and configure the address areas.

Use the CPUs below, which should be configured as follows:

CPU	Task	Address	Data
317-2 PN/DP	DP master	I 200	8 words from sender
315-2 DP	Sender	Q 100	8 words to master
317-2 DP	Receiver	I 120	First 2 bytes of the 8 words from the sender

Requirement

- You must be in the network view.
- The hardware catalog is open.

Setting up a PROFIBUS DP master system

To set up a PROFIBUS DP master system, proceed as follows:

1. Drag and drop the following CPUs from the hardware catalog into the network view: 315-2 DP, 317-2 DP and 317-2 PN/DP.
2. Give the CPUs meaningful names:
   - Select the CPU 317-2 PN/DP.
   - Assign the name "DP master" to the CPU under "Properties > General" in the Inspector window.

   Now repeat the naming process for the other CPUs, calling them "Sender" and "Receiver" respectively.
3. Select the CPU 317-2 PN/DP and then, if necessary, change the interface type from "MPI" to "PROFIBUS" under "Properties > MPI/DP interface > MPI address" in the Inspector window.
   Note: If the MPI/DP interface is set to "PROFIBUS" type, the entry is no longer "MPI address" but rather "PROFIBUS address".
4. Select CPUs 315-2 DP and 317-2 DP one after the other and in both cases select "Slave" mode under "Properties > DP interface > Mode" in the Inspector window.
   The two CPUs 315-2 DP "Sender" and 317-2 "Receiver" are now being operated as I-slaves.
5. While holding down the mouse button, draw the PROFIBUS connections between the master CPU and the two slave CPUs.

You have now successfully set up a PROFIBUS master system with the CPU 317-2 PN/DP as DP master and the 315-2 DP "Sender" and 317-2 DP "Receiver" as I-slaves. New lines for the communications partners will have been created in the I/O communications table relating to the CPUs. "I-slave" will be displayed in the "Mode" column.

Setting up direct data exchange

To set up direct data exchange within the PROFIBUS master system, proceed as follows:

1. Select the "Sender".
2. Press and hold down the left mouse button and then drag the "Receiver" to the "Place device here" field, which appears in the "Partner 2" column of the I/O communications table.
   Note: You can also open the drop-down list box in the "Place device here" field of the "Partner 2" column, and select the "Receiver" from this list.

You have now successfully set up a direct data exchange between the two I-slaves. A new line will have been created in the I/O communications table relating to the two I-slaves. "Direct data exchange" will be displayed in the "Mode" column.

Note

The arrow symbols in the I/O communication table between the "Peer 1" and "Peer 2" columns indicate the directions of the communication relations. The arrows point in both directions for master-slave and slave-I-slave relations. The arrows point from the sender to the receiver in the case of communication relations between I-slaves or during direct data exchange.

Configuring master-slave communication

To select master-slave communication, proceed as follows:

1. Select one of the two communication partners of master-slave communication (DP master CPU 317-2 PN/DP or I-slave CPU 315-2 DP "sender")
2. In the I/O communications table, click on the line with the respective second partner and "I-slave" mode.
3. Select "Properties > I-slave communication" in the Inspector window.

Or:

1. Select the CPU 315-2 DP "sender" I-slave.
2. Select "Properties > DP interface > Mode > I-slave communication" in the Inspector window.

To configure master-slave communication, proceed as follows:

1. Create a new transfer area under "I-slave communication" in the transfer area table:
   In the "Transfer area" column, click on "<Add new>" and select the "MS" communication type for master-slave communication in the "Type" drop-down list box. You can also create the new transfer area by setting the communication type in a new column only under "Type". The transfer area is then created automatically.
2. Under "I-slave communication", click on the newly created transfer area.
   The detail view of the transfer area is opened.
3. Make the following entries:
   for "DP master":
   - Address type I
   - Start address 200
   - Length 8
   - Unit WORD
   - Consistency total length

   for "Sender":
   - Address type Q
   - Start address 100

Once the data length has been specified, the slave and master addresses for the required data space are completed. If you click on "I-slave communication" in the Inspector window, you will see an overview of the updated configuration of master-slave communication.

Configuring direct data exchange

To configure direct data exchange, proceed as follows:

1. Select one of the two communication partners of direct data exchange (I-slave CPU 315-2 DP "Sender" or I-slave CPU 317-2 DP "Receiver")
2. In the I/O communications table, click on the line showing "Direct data exchange" mode.
3. Select "Properties > Direct data exchange" in the Inspector window.

To configure direct data exchange, proceed as follows:

1. Create a new transfer area under "Direct data exchange" in the transfer area table:
   In the "Transfer area" column, click on "<Add new>" and select the "DX" communication type for direct data exchange in the "Type" drop-down list box. You can also create the new transfer area by setting the communication type in a new column only under "Type". The transfer area is then created automatically.
2. Under "Direct data exchange", click on the newly created transfer area.
   The detail view of the transfer area is opened.
3. Make the following entries:
   for "Sender":
   - Address 100
   - Data length 1
   - Unit WORD

   for "Receiver":
   - Address 120

Note

If the sender in the direct data exchange is a normal DP slave, no transfer areas will be configured. Instead, an input module has to be added to the sender.

Particularity

In theory, you can also set an start address higher than 200 (202, for example). The length of consistent data will be adapted automatically. Furthermore, you can set a shorter length than that specified by the sender (1 byte, for example).

Note

If a consistent length of 3 bytes or more than 4 bytes is set for the sender and the data is to be transferred with the extended instruction of system function "DPWR_DAT" (SFC 15), the receiver will always need to use extended instruction "DPRD_DAT" (SFC 14) even if only 1 byte is actually being read, for example!

If you are using a loading operation (L IB..) instead of the DPRD_DAT to read 1 byte, a "0" is read in (incorrect value).

DPWR_DAT call on sender (CPU 315-2 DP)

```
STL
CALL  "DPWR_DAT"
       LADDR  :=W#16#64                //Start address Q 100
       RECORD :=P#M 10.0 BYTE 16       //User data source area
       RET_VAL:=MW100                  //Return value
```

DPRD_DAT call on receiver (CPU 317-2 DP)

```
STL
CALL  "DPRD_DAT"
       LADDR  :=W#16#78                //Start address I 120
       RET_VAL:=MW100                  //Return value
       RECORD :=P#M 10.0 BYTE 2        //User data target area
```

Using GSD files

GSD revisions

What you need to know about GSD revisions

The properties of DP slaves are made available to configuration tools by means of GSD files.

Functional enhancements in the area of the distributed I/O will have an effect on the GSD specification, for example, they will require the definition of new keywords.

This results in the versioning of the specification. In the case of GSD files, the version of the specification on which a GSD file is based is called a "GSD revision".

From GSD revision 1, the GSD revision must be included as a keyword "GSD_revision" in GSD files. GSD files without this keyword will therefore be interpreted by configuration tools as GSD revision "0".

GSD files can be interpreted up to GSD revision 5. This means that DP slaves that support the following functions, for example, will be supported:

- Diagnostic alarms for interrupt blocks
- Isochronous mode and constant bus cycle time
- SYNC/FREEZE
- Clock synchronization for DP slaves

Installing the GSD file

Introduction

A GSD file (device data file) contains all the DP slave properties. If you want to configure a DP slave that does not appear in the hardware catalog, you must install the GSD file provided by the manufacturer. DP slaves installed via GSD files are displayed in the hardware catalog and can then be selected and configured.

Requirement

- The hardware and network editor is closed.
- You have access to the required GSD files in a directory on the hard disk.

Procedure

To install a GSD file, proceed as follows:

1. In the "Options" menu, select the "Install device master data files" command.
2. In the "Install device master data files" dialog box, choose the folder in which you want to save the GSD files.
3. Choose one or more files from the list of displayed GSD files.
4. Click on the "Install" button.
5. To create a log file for the installation, click on the "Save log file" button.
   Any problems during the installation can be tracked down using the log file.

You will find the new DP slave installed by means of the GSD file in a new folder in the hardware catalog.

Note

Installation of GSD file cannot be undone.

See also

Overview of hardware and network editor (Page 475)

Configuring GSD-based DP slave

DP slaves that you have inserted through installation of a GSD file can be selected as usual via the hardware catalog and inserted in the network view. If you want to insert the modules of the GSD-based DP slaves, you must take into account some particular details.

Requirement

- You have installed a DP slave using a GSD file.
- You have inserted the head module in the network view in the usual manner.
- The device overview opens in the device view.
- The hardware catalog is open.

Procedure

To add the modules of a GSD-based DP slave, proceed as follows:

1. In the hardware catalog, navigate to the modules of the GSD-based DP slave.
   GSD-based DP slaves, also referred to as DP standard slaves, can be found in the "Other field devices" folder of the hardware catalog.
2. Select the desired module.
3. Use drag-and-drop to move the module to a free space in the device overview.
4. Select the module in the device overview to edit parameters.

You have now inserted the module in a free slot of the GSD-based DP slave and can edit its parameters.

Note

You can see only the GSD-based DP slave in the graphic area of the device view. The added modules of GSD-based DP slaves are only found in the device overview.

Preset configuration

For modules with an adjustable preset configuration, you can change this configuration in the inspector window under "Properties > Preset configuration".

See also

Objects in the device view (Page 500)

Inserting a module into a rack (Page 507)

Adding a device to the hardware configuration (Page 504)

Using PROFIBUS DPV1

Introduction to PROFIBUS DPV1

General

The DPV1 functionality is an extension of the PROFIBUS standard for distributed I/O. Relative to the functionality in the DPV0 predecessor, the DPV1 functionality has several extensions and simplifications:

- PROFIBUS DPV0 handles the cyclic exchange of data and the diagnostics.
- PROFIBUS DPV1 handles acyclic and cyclic data exchange plus interrupt handling.

Additional functionality for DPV1 devices (DP masters/DP slaves)

DP masters and DP slaves that support DPV1 have the following additional functions compared to devices that only support DPV0:

- The acyclic data exchange between master and slave is supported (read/write data record, e.g. to reassign parameters to a slave in operation). The data records of a module and the structure of these data records is provided in the documentation of the respective module.
- Interrupts can be set by a DPV1 slave that ensure handling of the interrupt triggering event in the master CPU. Even in "STOP" mode the interrupt data will be analyzed in the CPU (update of the diagnostic buffer and module status). However, OB processing does not take place in STOP.
  In addition to the interrupts that are known from SIMATIC, such as diagnostic interrupt for ET 200M, now the interrupt, status interrupt, update interrupt and vendor-specific interrupt are also supported.

See also

S7 integration of DPV1 slaves (http://support.automation.siemens.com/WW/view/en/10259221)

Configuring DPV1 devices

Some devices, such as the DP interface IM 151-1 HF, can toggle between DPV0 and DPV1.

Requirement

- You must be in network view.
- A DP master with DPV1 functionality must be available.
- A master-slave connection must be established with PROFIBUS.

Procedure

To switch the DP slave mode from DPV1 to DPV0 functionality, proceed as follows:

1. Select the DP slave.
2. Under "Properties > General > Module parameters" in the Inspector window, select "DPV0" or "DPV1" mode from the "DP interrupt mode" drop-down list.

or

1. Select the DP master.
2. In the I/O communications table, select the row with the connection between the DP master and the desired DP slave.
3. Under "Properties > General > Module parameters" in the Inspector window, select "DPV0" or "DPV1" mode from the "DP interrupt mode" drop-down list.

Using devices with different PROFIBUS functionality

The following rules apply to the mixed use of devices with DPV0 and DPV1 functionality:

- In principle, DPV0 slaves can also be operated on a DP interface with DPV1 functionality.
- In principle, DPV1 slaves can also be operated on a DP interface with DPV0 functionality. In this case, DPV1 functionality will be switched off. However, vendor-specific configuration rules for certain DP slaves can force DPV1 operation, so that these DP slaves cannot operate with the DP master system.

Assigning the diagnostics address

In the case of DPV1 slaves, the diagnostics address is automatically assigned to the virtual slot 0 as station proxy.

The following assignment basically applies:

- Diagnostics and interrupts that only refer to the entire DP slave are assigned to the virtual slot 0 and its diagnostics address: For example interrupts for modules in non-configured slots, station failure/station recovery (OB86 (Page 1796)).
- Diagnostics and interrupts that originate from this module (for example, from a DP connection IM 153-2 in slot 2) are assigned to the remaining slots with their respective start addresses.

See also

Interrupts on the I-slave (Page 1370)

Programming DPV1 devices

For DPV1 functionality, there are OBs and SFBs/SFCs that cannot be used for DPV0.

Interrupt OBs for DPV1 events

DPV1 slaves can trigger interrupts. For diagnostics interrupts, hardware interrupts and remove/insert interrupts, you can use the appropriate OBs made available by the operating system of the S7-CPUs.

The interrupt OBs for DPV1 are presented in the table below. Detailed information is provided in the description of the OBs.

DPV1 interrupt	OB	Explanation
Status interrupt	OB55 (Page 1764)	The status interrupt can be triggered if the operating state of a device or a module changes, for example, from RUN to STOP. A precise description of the events for which a status interrupt is triggered is provided in the documentation provided by the manufacturer of the respective DPV1 slave.
Update interrupt	OB56 (Page 1768)	An update interrupt can be triggered when new parameters are assigned to a slot. This may have been caused, for example, by local access or partner access to the parameters. A precise description of the events for which an update interrupt is triggered is provided in the documentation provided by the manufacturer of the respective DPV1 slave.
Vendor-specific interrupt	OB57 (Page 1772)	The event that triggers the vendor-specific interrupt can be specified by the manufacturer of a DPV1 slave.

Instructions for access to DPV1 slaves

The DPV1 interfaces with their functions are presented in the table below. Detailed information is provided in the description of the extended instructions.

Function	Interface (DPV1)	Comments
Read data record	RDREC (Page 6459)	-
Write data record	WRREC (Page 6461)	-
Receive interrupt from a DP slave	RALRM (Page 6468)	The instruction must be called in the OB that triggers the interrupt.

Note

If a DPV1 slave is used and the DP interface of the DP master is set to "DPV0", no data records may be read from or written to the I/O modules using "WR_REC"/"RD_REC" or "WRREC"/"RDREC" instructions in the user program. In this case the DP master addresses the wrong slot (configured slot+3).

Solution: Change the interface of the DP master to "DPV1".

Check list for changing from DPV0 to DPV1

The following sections of the existing user program must be observed when you have edited the configuration and changed the interface from "DPV0" to "DPV1".

Function	What do I need to observe?
Address conversions	If you have used address conversions in the user program, for example, with extended instructions of system functions "GADR_LGC (Page 6788)" (SFC 5), "LGC_GADR (Page 6790)" (SFC 49), "RD_LGADR (Page 6787)" (SFC 50), you must check the slot <-> logical start address assignment for the DPV1 slaves. In addition, slot 0 has an address. DPV1 slave: Whereas previously the first I/O module of the DP slave was assigned to slot 4 by the extended instructions, now the first I/O module is assigned to slot 1 (as shown in the hardware configuration). DPV1 slave integrated in the hardware catalog (for example, ET 200M): Interface module (slot 2) has its own address.
Reading diagnostics information using the system function "DPNRM_DG (Page 6523)"	The originally assigned diagnostics address will also continue to function. Internally this address is assigned to slot 0. The diagnostic data record of the DPV1 slave is structured differently than it is with DPV0 (see description of the DP slave for ET 200M, for example, also under the keyword "Extended diagnostics").
Reading/writing data records	If you transfer data records to a DPV1 slave using extended instruction "WR_REC (Page 6496)" (SFC 58) or read data records from a DPV1 slave using extended instruction "RD_REC (Page 6491)" (SFC 59) and this slave is working in DPV1 mode, the DP master analyzes the error information received from the slave as follows: If the error information is in the areas W#16#8000 to W#16#80FF or W#16#F000 to W#16#FFFF, the DP master will forward the error information to the SFC. If it is outside of these areas it gives the value W#16#80A2 to the SFC and suspends the slave. For the description of the error information originating from the DPV1 slave, see the description of the "RALRM" (SFB 54) instruction. Additional information is available in the descriptions on jumps in language descriptions, module help and system attributes.
Reading out system status lists	If you use the extended instruction "RDSYSST (Page 6681)", for example to read module status information or rack/station status information, you must also take the changed meaning of the slots and the additional slot 0 (see above) into consideration.

Example for analyzing interrupt information

Introduction

A distributed S7 digital input module (start address 288) triggers a hardware interrupt. In OB40 (Page 1762), the additional interrupt information of this module can be read using the advanced instruction "RALRM (Page 6468)". The system checks whether the first channel has triggered a hardware interrupt.

In the case of S7 modules, additional interrupt information could also be directly read from the start information of the OB40. However, the DPV1 functionality allows up to 59 bytes of additional interrupt information, which is far too much for the start information of the OB40.

Analyzing interrupt information from OB40 with "RALRM"

STL

```
// ...
// ...
//Switch for interrupt triggering address (288)
    L    DW#16#120
    T    "MD10"
    CALL  "RALRM" , "DB54"
      MODE  :=1                   //Function mode: 1=set all output parameters (i.e. F_ID
has no effect)
      F_ID  :="MD10"              //Start address of the slot from which an interrupt is
permitted
      MLEN  :=8                   //Max. length of the additional interrupt info in bytes
(for example, for channel status of the module)
      NEW   :="Alarm_neu"         //Interrupt received? (Yes = 1)
      STATUS:="DP_RALRM_STATUS"   //Return value with function result/error message
      ID    :="Slotaddress_Interrupt" //Start address of the slot from which an interrupt
was received
      LEN   :="Length_Alarminfo"  //Length of the additional interrupt info (4-byte header
info + for example, 4 bytes for S7 I/O modules)
      TINFO :=P#M 100.0 BYTE 28   //Pointer for OB start info +management info: 28 bytes
from MB 100
     AINFO :=P#M 130.0 BYTE 8    //Pointer for target area of the header info + supplemental
interrupt info (max. 59 bytes)
    A    M    124.0               //Is input 1 (bit 0) the interrupt triggerer?
    JC    Alrm
    BEU
Alrm:    S    Q   0.0             // interrupt processing
// ...
```

Example for analyzing diagnostic data

Introduction

The target area for diagnostic data should suffice for standard diagnostics (6 bytes), for identifier-specific diagnostics (3 bytes for 12 slots) and for analyzing the device-specific

diagnostics (module status only, which is an additional 7 bytes). Extended instruction "RALRM (Page 6468)" in OB82 (Page 1783) is used for this.

For more extensive analysis (channel-specific diagnostics), additional bytes would be reserved if the DP slave supports this function.

Analyzing diagnostic data in OB82 with "RALRM" (SFB 54)

STL

```
// ...
// ...
      L     120                          //Specify start address for device/module,
      T     "Slotadress_Diag"           //from which diagnostics must be fetched


      CALL  "RALRM" , "DB54"
       MODE  :="Alle_Params"             // 1 = All output parameters are set
       F_ID  :="Slotadresse_Diag"        //Start address of the slot from which the
diagnostics will be fetched
       MLEN  :=20                       //Max. length of the diagnostic data in bytes
       NEW   :="new"                     //irrelevant
       STATUS:="RET_VAL"                 //Function result, error message
       ID    :="Slotadresse_Alarm"      //Start address of the slot from which an interrupt
was received
      LEN   :="Laenge_Alarminfo"       //Length of additional interrupt info (4-byte header
info +16 bytes diagnostic data)
       TINFO :=P#M 100.0 BYTE 28   //Pointer for OB start info +management info: 28 bytes
from MB 100
       AINFO :=P#M 130.0 BYTE 20         //Pointer on target area in which the diagnostic
data will be stored

// ...
//Structure of the stored diagnostic data:
//   MB 130 to MB 133: Header info (length, identifier, slot)
//   MB 134 to MB 139: Standard diagnostics (6 bytes)
//   MB 140 to MB 142: Identifier-specific diagnostics (3 bytes)
//   MB 143 to MB 149: Module status (7 bytes)
// ...
      A     M    141.0                  //Slot 1 with error?
      JC      stp1
      BE

stp1: L     MB   147                    //Module status slot fetch 1 to 4
      AW    W#16#3                      //Filter out slot 1
      L     W#16#2                      //2-bit status "wrong module" inserted
      ==I
      S     Q      0.1                  //Response to wrong module

      L     MB   147                    //Module status slot fetch 1 to 4
      AW    W#16#3                      //Filter out slot 1
      L     W#16#1                      //2-bit status "invalid data"
      ==I
      S     Q      0.2                  //Response to invalid user data
// ...
```

Slot model of the DPV1 slaves for I slaves

Introduction

This section describes how to make the assignments of I/O addresses and diagnostics addresses to the slots visible for DPV1. In particular, it deals with addresses that do not transport user data and their configuration.

The slot model for DPV1

With DPV1, a DP slave is formed by slots in precisely the same manner as DPV0. The numbers of the slots are 0, 1, ...n. Slot 0 applies only for DPV1 and has prominent significance because it is the proxy of the entire DP slave.

For example, proxy means that interrupts that have been triggered from slot 0 originate from the entire DP slave and not from a specific slot within the DP slave. Diagnostics that come from this slot are assigned to the entire DP slave and not to a specific slot or individual module.

Addresses for DP interfaces

From the CPU perspective, there is a separate logical address for each of its interfaces. The address for the respective DP interface is displayed in various ways:

- When you select a device or a DP interface in the Inspector window under "Properties > PROFIBUS address (Xn)" or "Properties > MPI/DP interface > PROFIBUS address (Xn)".
- With an enabled "Display addresses" in the toolbar in the network view: You will then see the name of any connected subnets for each interface, followed by the assigned PROFIBUS address.

These addresses have nothing to do with the slot model of DP slaves; they serve merely as internal CPU identification, for example, in the event of a failure of the interface. This address is of minor significance for the user program.

Slots and addresses for user data

Every manufacturer of a DP slave can freely determine which data will come from which slot.

As opposed to DPV1 slaves with user data starting with slot 1, DPV0 slaves are subject to a restriction in that the first I/O module must always be located in slot 4. DP slaves that are installed via a GSD file, can have user data starting with slot 1. Because distributed I/O data is usually accessed via its addresses, in the same way as central I/O data, user data of the DPV0 slaves must be addressed from the start address of slot 4. This also applies to intelligent DP slaves.

In the case of intelligent DP slaves, you can allocate I/O memory areas of the I-slaves to I/O memory areas of the DP master by means of the "I-slave communication" transfer areas. During operation (cyclic data exchange), the data that you transfer into the I/O memory areas in the user program of the intelligent DP slave are transferred via the transfer area into the I/O memory area of the DP master. You will find the table for the assignment of the I/O memory areas to the transfer areas by selecting the I-slave connection in the I/O communications table under "Properties > I-slave communication" in the Inspector window. After creating a transfer area, you can carry out the rest of the configuration under "Details of the transfer area".

When configuring the addresses under "Details of the transfer area", slot numbers are also displayed. The slots are not generated by real modules, as is the case with ET 200M, for example, but rather by freely definable lengths of the respective I/O areas. In this case they are also referred to as virtual slots.

The following points are important for an understanding of the address assignments:

- In addition to its real slots, the intelligent DP slave has virtual slots that are located in the I/O memory area.
- Like the real slots, the virtual slots are accessed via logical addresses. In the case of a DP slave, such as ET 200M, the virtual slots are accessed via the start address of a module; in the case of I-slaves via the configured address of the "I-slave communication".
- The addresses of the virtual slots are different from the perspective of the DP master and that of the DP slave. The assignment can be configured. Thus, the same DP slave will usually be accessed by the DP master under a different address from the one with which it is addressed by the DP slave.

Example of an address assignment for user data

The following table shows the assignment of the virtual slots in the transfer area:

Sample address from the perspective of the DP slave	Significance for the DP slave	Slot	Significance for the DP master	Sample address from the perspective of the DP master
		0		
		1		
		2		
		3		
I 2	Via input byte 2, you can read ...	4	... what the master has written in output byte 4.	Q 4
Q 5	what was written to output byte 5 in the slave...	5	... can be read in the master as input byte 6	I 6
I 8	...	6	...	Q 8
		...		
		35		

Note

In the network view, you will find slot assignment for the selected I-slave connection in the I/O communications table under "I-slave communication > Transfer areas" in the Inspector window.

Addresses for system information

Addresses for system information are used to handle diagnostics information, for example, or information about operating state transitions.

Addresses of the DP slave

The system information of a DP slave is assigned to modifiable addresses. In this context, the following addresses are relevant for DPV1:

- Addresses for device diagnostics:
  - The DP master diagnoses a failure or recovery of the I-slave.
  - The I-slave diagnoses a failure or recovery of the DP master.
- Addresses for operating state transitions:
  - The DP master can determine a change in operating state of the DP slave.
  - The intelligent DP slave can determine a change in operating state of the DP master.

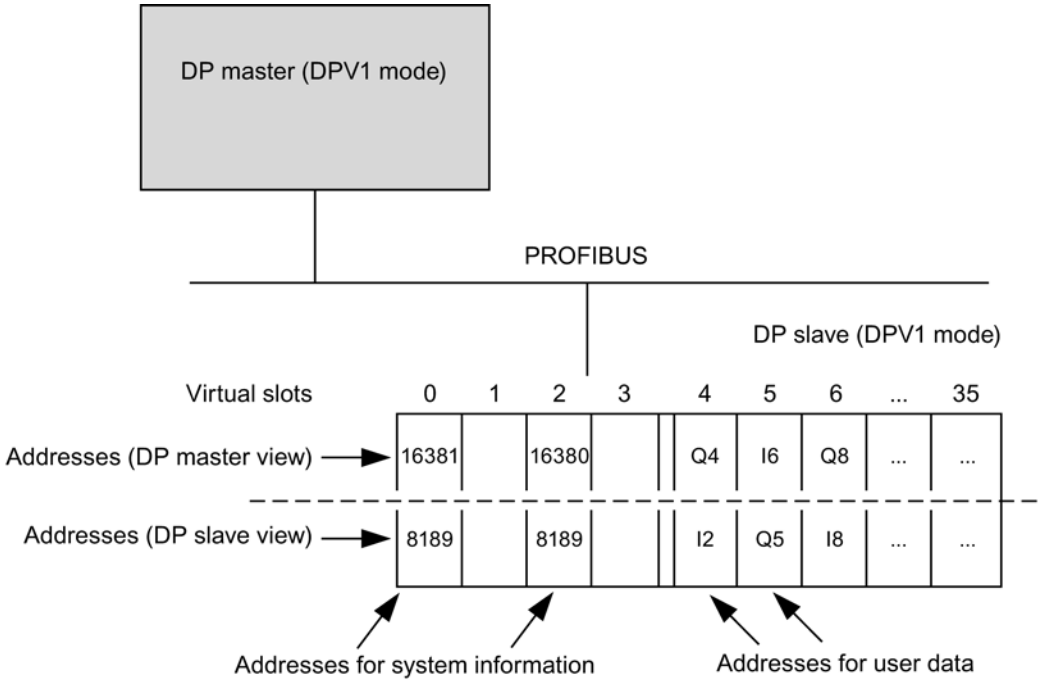
The addresses are automatically assigned from top to bottom so that a conflict with user data does not arise. The recommended addresses should be accepted even if they can be changed. If the user program is to run on different CPUs, check whether the addresses are also in the address area of the "smallest" CPU.

The following table shows the assignment of the addresses in I-slave communication:

Sample address from the perspective of the DP slave	Significance for the DP slave	Sample address from the perspective of the DP master	Significance for the DP master
8189	Station failure/station recovery of the DP master	16381	Station failure/station recovery of the DP slave
8188	Operating state transition of the DP master	16380	Operating state transition of the DP slave

Summary

With virtual slots in the transfer area, the I-slave configuration looks like this:



Triggering a hardware interrupt with the DP_PRAL instruction

Using the system instruction in the extended instruction "DP_PRAL (Page 6515)", you can trigger a hardware interrupt for each configured address from the user program of the I-slave CPU. This applies both to user data addresses from the I/O area and to the address of virtual slot 2.

In the user program of the I-slave, for example, use the I/O addresses configured in the "Local..." column for instruction "DP_PRAL".

A hardware interrupt will then be triggered in the user program of the master. In the start information of the hardware interrupt OB (for example, OB40 (Page 1762)), the address that you have configured in the "PROFIBUS DP partner" column will also be provided as the interrupt triggering address.

Constant and isochronous modes

Information about constant bus cycle time and isochronous modes

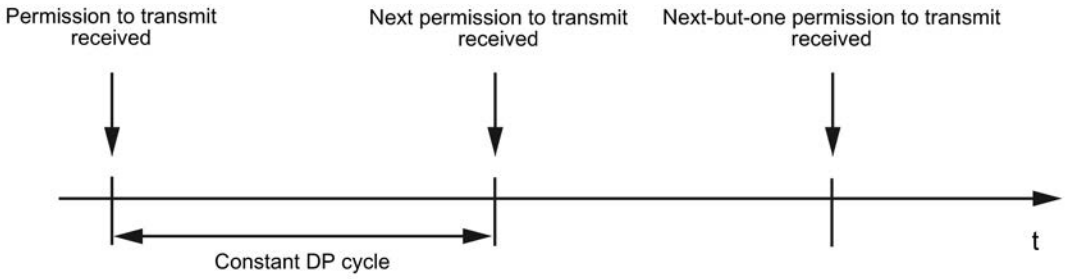
Same length bus cycles for PROFIBUS subnets

Introduction

The PROFIBUS properties "Isochronous mode" and "Constant bus cycle time" are the basis for synchronized processing cycles.

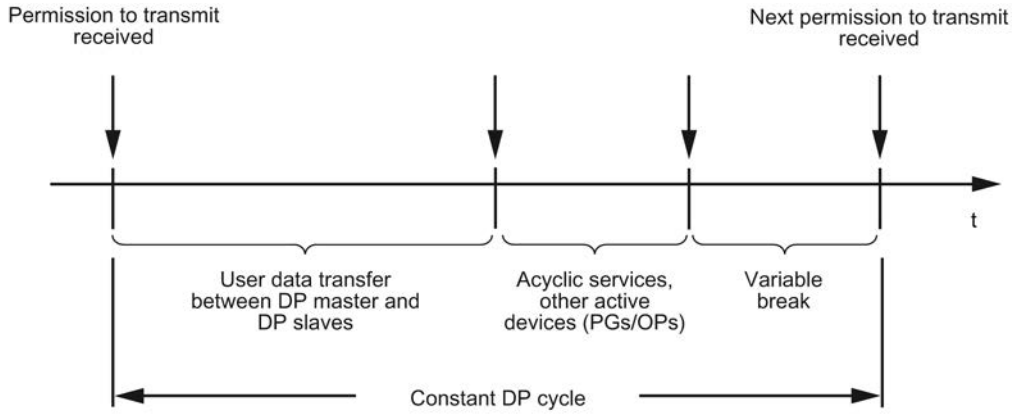
Constant bus cycle time ensures a precise same length time span for bus cycles. "Same length bus cycles" mean that the PROFIBUS-DP master always starts the DP bus cycle after the same time span. Thus the connected slaves also get their data from the master in time intervals that stay precisely the same. This is also referred to as a "Clocking of the bus cycle".

Constant bus cycle time is the prerequisite for isochronous mode.



Composition of bus cycle time

The following figure shows how time is composed for a bus cycle.



The "variable pause" shown in the figure is always minimal, if communication jobs are still pending, e.g. for additional active stations. The master, or to more precise the constant bus

cycle time master, controls the communication proportions in such a manner that the same duration for a bus cycle is always achieved.

Requirements

Constant bus cycle times have the following requirements:

- The DP master must support the "Constant bus cycle time" function.
- The constant bus cycle time master must be a class 1 DP master. A programming device/PC is not suitable as a constant bus cycle time master.
- The constant bus cycle time master must be the only active station on the PROFIBUS DP. Only one DP master system should be present on the PROFIBUS subnet. Programming devices or PCs may be connected in addition.
- No SYNC/FREEZE groups should be configured.
- Constant bus cycle time is only possible for the "DP" and "user-defined" bus profiles.
- Do not configure CiR (Configuration in Run).
- A H-CPU cannot be connected on the PROFIBUS subnet.

Time for constant bus cycle time DP cycle

The time for the constant bus cycle time for DP cycle is calculated automatically:

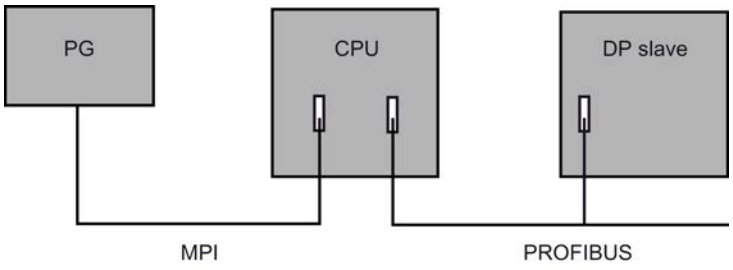
- From the PROFIBUS configuration (number of configured stations).
- From additional information for the calculation that is optionally entered: For example programming devices that must be taken into consideration, but that are outside of the project.

The calculated time for the constant bus cycle time for DP cycle takes the user data traffic of the PROFIBUS DP master into account. If errors occur then constant bus cycle time violations can occur.

You can change the calculated time within the displayed interval.

Influence of connected active stations (programming devices/PCs and I slaves)

A programming device/PC must only be taken into consideration if it is directly connected on the PROFIBUS with its PROFIBUS interface. If the programming device is connected via the MPI interface of the CPU, then it does not need to be taken into account as is shown in the figure below.



If intelligent DP slaves, e.g. CPU 315-2 DP, are connected then the time for constant bus cycle time for DP cycle must be more generously measured.

Synchronized execution cycles

PROFIBUS supports isochronous mode

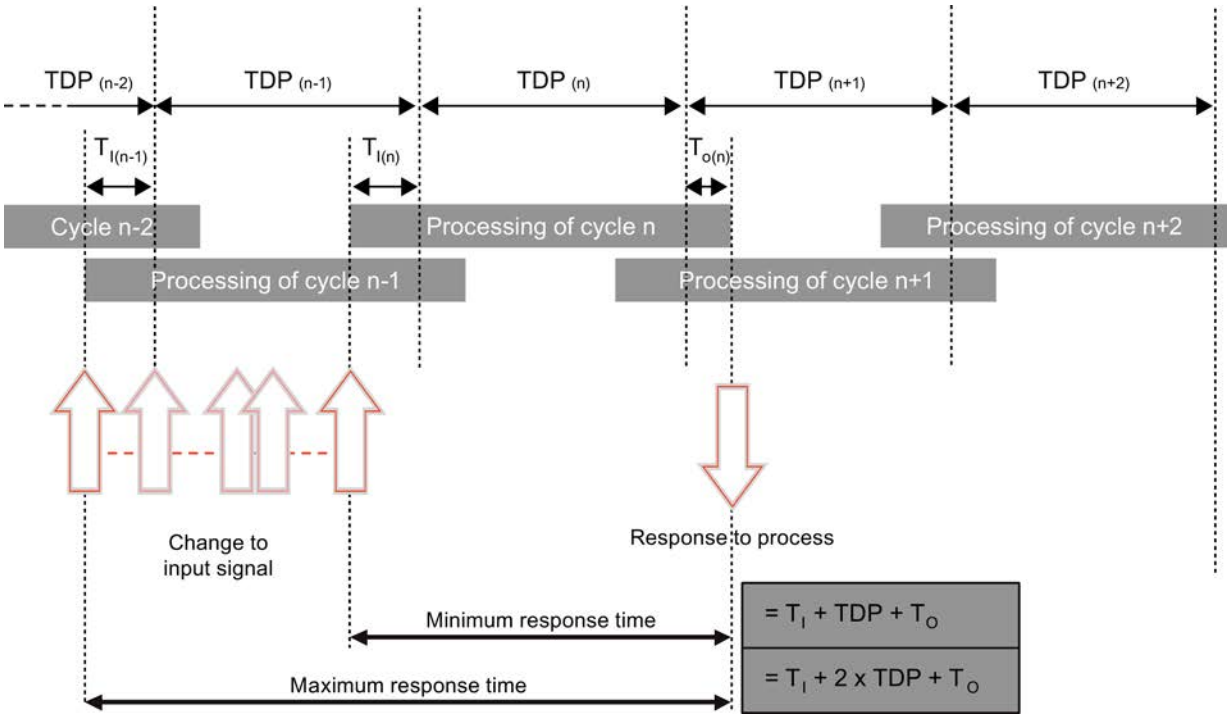
Isochronous PROFIBUS is the essential basis for synchronized process cycles. The PROFIBUS system provides a reliable basic clock. The "isochronous mode" system property couples an automation solution with the constant bus cycle time PROFIBUS.

That is:

- Input data is read in synchronism with the DP cycle; all input data is read simultaneously.
- The user program which processes the I/O data is synchronized with the DP cycle by means of isochronous mode interrupt OBs OB 61 to OB 64.
- Data output is synchronized with the DP cycle; all output data take effect simultaneously.
- All input and output data is transferred consistently. That is, all data of the process image are associated logically and based on the same timing.

Synchronization of the single cycles

The following figure shows the bandwidth of possible response times for a processing cycle from a series of isochronous single cycles that occur due to synchronization of the input data read-in.



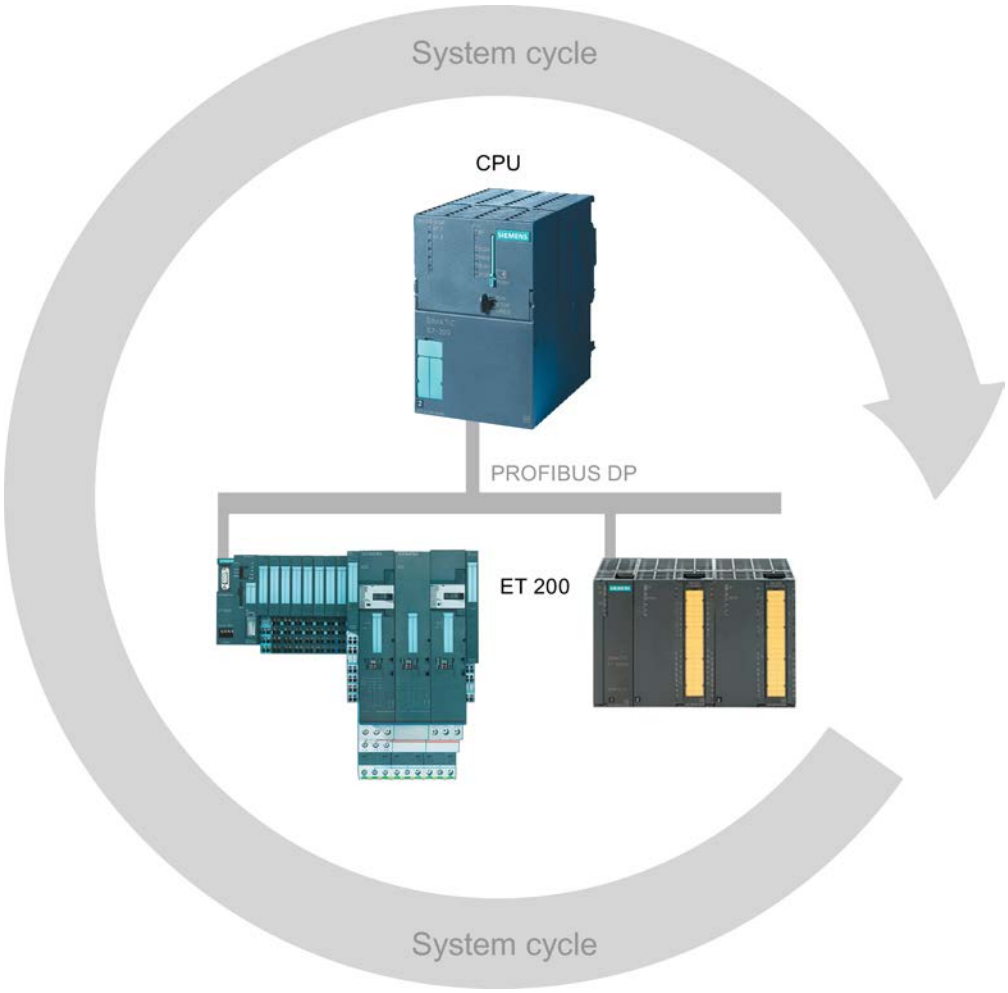
TDP	System cycle
Ti	Time of data input
To	Time of data output

Synchronization of single cycles makes it possible to read input data within cycle "n-1", to process and transfer the data within cycle "n", and to transfer the calculated output data to the "terminals" at the start of cycle "n+1". Resultant process reaction time: from "Ti + TDP + To" to "Ti + (2 x TDP) + To".

The "isochronous mode" system property enables constant cycle times within the SIMATIC system; the SIMATIC system is strictly deterministic on the bus system.

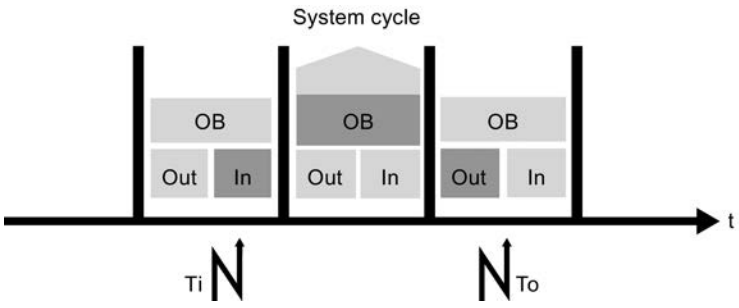
Just in time

The fast and reliable response time of a system operating in isochronous mode is based on the fact that all data is provided just-in-time. The constant bus cycle time (isochronous) DP cycle specifies the cycle for this.



Bias time Ti

The start of the I/O read cycle is advanced by the amount of a bias time Ti in order to make all input data available for transfer on the DP subnet at the start of the next DP cycle. The bias time Ti is configured by the user or it is specified automatically.



Processing the input data

PROFIBUS transfers the input data to the DP master via the DP subnet. The isochronous mode interrupt OB (OB 61, OB 62, OB 63 or OB 64) is called. The user program in the isochronous mode interrupt OB determines the process reaction and provides the output data in time for the start of the next DP cycle. You can have the length of the DP cycle calculated automatically or you can configure it yourself.

Making the output data available

The following applies for the output data:

- It will be made available promptly at the start of the next DP cycle.
- Is transmitted via the DP subnet to the DP slaves.
- It is isochronously, i.e. simultaneously forwarded to the process at the time To.

The result is a total reproducible reaction time of "Ti + (2 x TDP) + To" for the transfer from the input to the output terminal.

Characteristics of isochronous mode

Isochronous mode is based on three essential characteristics:

- The user program is synchronized with I/O processing, i.e. all operations are coordinated on a time basis. All input data is logged at a defined time. The output data also take effect at a defined time. The I/O data is synchronized with the system cycle up to the terminals. The data of one cycle are always processed within the next cycle, and take effect at the terminals within the subsequent cycle.
- I/O data is processed in isochronous mode, that is, input data is always read at constant intervals, and are always output at constant intervals.
- All I/O data is transferred consistently, that is, all data of a process image is associated logically and based on the same timing.

Example for isochronous capture of multiple measuring points

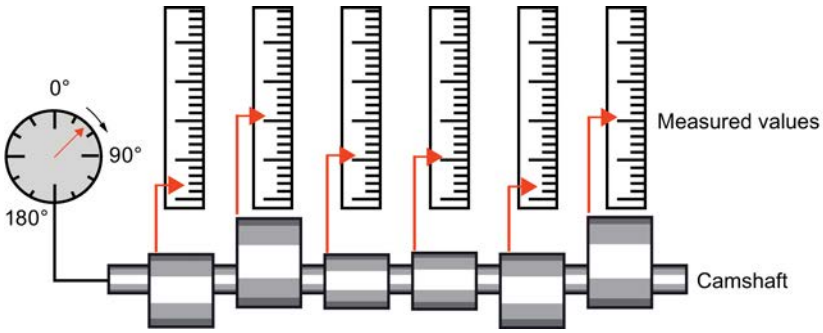
Introduction

With the "Isochronous mode" system property, measured values and process data can be captured in a fixed system clock. The signal processing until switching on the output terminal occurs in the same system clock. Isochronous mode thus contributes to high quality of the controller and thus to greater manufacturing precision. With isochronous mode the possible fluctuations of process response times are drastically reduced. Time-assured processing can be used for a higher machine clock/cycle.

Basically isochronous mode is an advantage wherever measured values must be captured synchronously, movements coordinated, and process responses must be defined and simultaneously executed as in the following example.

Task

QC requires precise measurement of dimensions within a camshaft production process.



Conventional workflow

1. Machining of the cam shaft on the lathe
2. Stop
3. Measurement of the angular position
4. Measurement of all cam gradients
5. Continue machining, stop, measurement...

Isochronous mode workflow

Implementation of the "isochronous mode" system property and the resultant synchronization of measured value acquisition forms the basis of continuous and accelerated measuring processes. Resultant workflow:

1. Continuous machining of the camshaft
2. Synchronous measurement of the camshaft positions and gradients within a continuous rotation.
3. Machining of the next camshaft

All camshaft positions and the corresponding measured values (red in the figure) are measured synchronously within a single rotation of the camshaft. This increases machine output and maintains or enhances precision of the measurement.

Rules, requirements, and supplemental conditions

What you need to be aware of for configuring isochronous mode is

- Isochronous mode cannot be used on optical PROFIBUS networks.
- Constant bus cycle time and isochronous mode are only possible for the "DP" and "user-defined" bus profiles.
- Isochronous mode is only possible with the DP interfaces integrated in the CPU. Isochronous operation with CPs is not possible for PROFIBUS.
- Only the constant bus cycle time master is allowed as active station on isochronous PROFIBUS-DP. OPs and programming devices (or PCs with programming device functionality) influence the dynamic response of the constant bus cycle time for the DP cycle and thus are not recommended.
- Isochronous mode across subnets is not possible.
- The addresses of the isochronous I/O have to be located in the address area of the process image partitions since the isochronous I/O can only be processed in process image partitions. Without the use of process image partitions isochronous consistent data transfer is not possible. Compliance with configuration limits is monitored because the number of slaves and the number of bytes on the DP master system are limited per process image partition.
- In addition, non-isochronous I/O may be located along with isochronous I/O in the same process image partition. An isochronous module/submodule has to be located in the process image partition.
- A full isochronous mode from "terminal" to "terminal" is only possible if all components participating in the chain support the "isochronous mode" system property. When selecting in the catalog, make sure you check the "isochronous mode" or "processing in isochronous mode" entry in the information field of the module.
- When you configure isochronous mode, you cannot assign a SYNC/FREEZE group to the slave.

See also

Assigning a DP slave to a SYNC/FREEZE group (Page 1354)

Isochronous mode manual (http://support.automation.siemens.com/WW/view/en/15218045)

Configuring and assigning parameters to isochronous mode

Optimizing dynamic response by selecting a suitable topology

Influence of the DP topology

The dynamic response of isochronous mode can be enhanced by selecting a suitable DP topology.

Separation of isochronous and non-isochronous I/O

Isochronous and non-isochronous I/O can be combined in a DP master system.

It is advisable to distribute isochronous and non-isochronous I/O to different DP master systems for good performance.

If TDs, OPs or programming devices are used then you should not operate them on the isochronous DP master system.

Practical distribution of the I/O submodules

Reduce Ti and To times to minimum by setting the focus on a practical distribution of I/O submodules to the interface modules (ET 200M, ET 200S) when you design the DP topology.

Note

Vary the different configurations and calculate the temporal behavior. Select the most favorable configuration.

Additional information

Additional instructions for optimization of the constant bus cycle time are available in the operating instructions of the ET 200M and ET 200S DP slave families.

Assigning constant bus cycle time and isochronous mode parameters

Configuring constant bus cycle time and isochronous mode parameters

Configuration for isochronous mode

A configuration for isochronous mode consists of the following isochronous components:

- CPU with integrated DP interface (for example, CPU 319-3 PN/DP)
- DP interface modules (for example ET 200S interface IM 151-1 High Feature)
- Distributed input/output modules (for example DI 2xDC24V, High Feature [131-4BB00], DO 2xDC24V/2A, High Feature [132-4BB30])

A current list of isochronous components is available in the Service & Support Portal (http://support.automation.siemens.com/WW/view/en/14747677).

Procedure

To configure a constant bus cycle time and isochronous mode, follow these basic steps:

1. Configure a CPU capable of isochronous mode and set the properties for isochronous mode:
   - Isochronous mode interrupt
   - Assigned DP master system
   - Process image partition
   - Delay time
2. Configure isochronous DP slaves with isochronous modules and set the properties for isochronous mode centrally in the PROFIBUS properties or for the specific DP slaves (slave properties). Here, these are essentially the Ti/To values and how they are calculated.

   Note

   You can set all properties for the isochronous I/O centrally in the PROFIBUS properties. Here, you also have an overview of all slaves and their settings on PROFIBUS at the same time.

3. Write the user program with accesses to the isochronous I/O.

You will find a detailed description of the procedure in the next sections.

See also

Setting CPU properties (Page 1404)

Setting PROFIBUS properties (Page 1406)

Setting the isochronous mode of DP slaves (Page 1408)

Creating a user program (Page 1408)

Setting CPU properties

The CPU settings for isochronous mode interrupts link the user program into isochronous mode processing using the relevant organization blocks (for example, isochronous mode interrupt OBs).

Requirement

The CPU must support isochronous mode and the constraints must not exclude isochronous mode.

Procedure

To set the CPU properties, proceed as follows:

1. Select the CPU in the device view or the network view.
2. Select the "Isochronous mode interrupts" CPU properties.
3. Make the following settings for each isochronous mode interrupt OB:
   - DP master system used
   - Required process image partition or partitions
   - Delay time

Additional information about the process image partitions

The assigned process image partitions must be updated in the isochronous mode interrupt OB by calling extended instructions "SYNC_PI" (SFC 126) and "SYNC_PO" (SFC 127). The "SYNC_PI" call updates the process image input partition and the "SYNC_PO" call updates the process image output partition.

If the distributed I/O addresses have been assigned to different process image partitions, enter the numbers of the process image partitions separated by commas (for example, 1,2,3).

Additional information on the delay time

- The delay time is the time between the global control frame and the start of OB 6x. During this time, the DP master performs the cyclic data exchange with the DP slaves.

  Note

  Recommendation: There is a check box "Automatic minimum" next to the input box for the delay time. If you have selected this check box, you do not need to set the delay time again after configuration changes!

- The delay time can then only be set when the "Automatic minimum" check box is deselected.

  Note

  If the delay time is 0, OB 6x starts with the global control frame. In this case, access to isochronous I/O by means of the advanced instruction "SYNC PI" (SFC 26) will very probably fail because the I/O data is not yet available at the start time of the OB.

  If a too large delay time were selected, access to the isochronous I/O would have been possible earlier. A delay time that is too large leads to an unnecessarily long OB runtime.

See also

Configuring constant bus cycle time and isochronous mode parameters (Page 1403)

Rules, requirements, and supplemental conditions (Page 1402)

Setting PROFIBUS properties

Introduction

To edit the properties of isochronous mode on PROFIBUS, you will need to change the settings on the subnet/DP master system.

The constant bus cycle time for isochronous mode in a PROFIBUS subnet with a constant bus cycle time is activated automatically if the following conditions are met:

- Assignment of a DP master system to the isochronous mode interrupt OB in the CPU properties for isochronous mode interrupts.
- Isochronous mode setting for a DP slave on PROFIBUS.

You only need to edit the PROFIBUS properties if you want to change the times manually. Otherwise all times will be calculated automatically.

Requirement

PROFIBUS subnet with constant bus cycle time.

Procedure

To change the properties of isochronous mode in PROFIBUS, follow these steps:

1. In the network view, select the PROFIBUS subnet, and in the Inspector window select "Constant bus cycle time" under "Properties".

   Note

   Make sure that you do not select a highlighted master system. The PROFIBUS properties and the isochronous mode properties can only be changed if the PROFIBUS subnet is selected.

2. Set the properties for the isochronous mode. You will find explanations of the specific settings in the sections below.

Properties for the isochronous mode are not selectable

If the properties for isochronous mode cannot be selected, check the following:

- Attachment of only one DP master on the PROFIBUS subnet. Isochronous mode is not possible if multiple DP masters are attached.
- Support of isochronous mode by the DP master (for example a CPU with an integrated DP interface).
- Configurability of the SYNC/FREEZE groups. SYNC/FREEZE groups and isochronous mode are mutually exclusive.

Setting DP cycle time and Ti/To values manually

If you want to set the DP cycle time manually, first clear the "Automatic minimum DP cycle time" check box. Now you can enter a higher value than the automatically calculated minimum DP cycle time.

If you want to set Ti and To values manually, first clear the "Automatic Ti/To values" check box.

Isochronous mode and Ti/To settings in the detailed overview

The detailed overview allows you to set the following properties for each suitable slave and/or for each suitable module in the slave:

- Isochronous mode
- Ti/To values

The detailed overview is a table. You can use the shortcut menu of the title bar of the table to adapt the tabular display, e.g. to display the values for TDPMin.

It is not possible to manually select or clear the check box "Isochronous mode" for a CPU (PLC). The check box is selected automatically if an isochronous mode interrupt is assigned to the DP master system in the "Interrupts" section of the CPU properties.

Possible settings for calculating the Ti/To values in the detailed overview

You can make the following settings for calculating the Ti/To values:

- From subnet
  If you activate isochronous mode, then the option Ti/TO values "From subnet" is automatically the default. With this option, all slaves that obtain their Ti/To values from the network have identical Ti/To times (those of the PROFIBUS subnet).
- Automatic minimum
  If, for example, you change the DP cycle time manually or change Ti/To of other slaves values manually and this change affects the Ti/To times of the slave, then all necessary adjustments will be made automatically. With this option, the Ti/To values of the slaves can deviate from each other.
- Manual
  With this option you can enter Ti/To values for the slave manually. Errors can occur with this setting:
  - Times selected too short
  - Ti/To of other DP slaves can no longer be calculated automatically due to the settings you made

Display "-" for time Ti (To)

If no value is displayed for Ti or To for the PROFIBUS subnet (display "-") then this is due to the following cause: The common parts of all Ti/To value ranges of all DP slaves is used for automatic calculation of the Ti/To time for the PROFIBUS subnet. The smallest value from these common ranges is selected. If no common range can be formed then "-" is displayed. In this case, a DP slave cannot obtain a Ti/To value from the network!

See also

Configuring constant bus cycle time and isochronous mode parameters (Page 1403)

Ti/To overlap (Page 1409)

Setting the isochronous mode of DP slaves (Page 1408)

Assigning a DP slave to a SYNC/FREEZE group (Page 1354)

Setting the isochronous mode of DP slaves

Settings on the DP slave

Except for the constant bus cycle time for the DP cycle and the Ti/To times for the PROFIBUS subnet, you can make the same settings in the DP slave properties that you make in the PROFIBUS properties.

Procedure

To set the isochronous mode for modules, follow these steps:

1. In network view, select the DP slave whose properties you want to edit.
2. Under "Properties > Isochronous mode", adapt the parameters to suit your requirements.

Setting Ti/To times the same for all slaves

The default behavior after adding DP slaves in network view is: Ti/To values are obtained from the network and are therefore automatically the same.

You can either set the same Ti/To times manually in an edited configuration or set all slaves so that they obtain their Ti/To values from the network.

See also

Configuring constant bus cycle time and isochronous mode parameters (Page 1403)

Setting PROFIBUS properties (Page 1406)

Creating a user program

Procedure

To create the user program for the isochronous mode, proceed as follows:

1. Create the necessary isochronous mode interrupt OBs (for example, OB 61).
2. In the isochronous mode interrupt OB, call the following extended instructions to update the process image partition:
   - "SYNC_PI" (SFC 126) to update the process image input partition
   - "SYNC_PO" (SFC 127) to update the process image output partition

Use the process image partition that is set on the CPU (parameter area "Isochronous mode interrupts").

Note

The following situation can occur, particularly with very short DP cycle times: The execution time of the user program (OB 6x with SYNC_PI/SYNC_PO called) is greater than the shortest cycle (see technical specifications of the CPU, section "Isochronous mode"). In this case, you will need to increase the automatically calculated DP cycle time manually.

See also

Configuring constant bus cycle time and isochronous mode parameters (Page 1403)

Ti/To overlap

Ti/To overlap

Reduction of the system cycle and of the process reaction time

The TDP system cycle and process response time are reduced by overlapping Ti and To, that is, overlap reduces the accumulated time that includes generation and detection of an event, plus the processing time, plus the time required to return a response signal at the outputs.

Requirement of Ti /To overlap

Condition of overlapping Ti/To:

ET 200M: Interface module IM 153-2; order no. 6ES7 153-2BAx2-0XB0 or higher.

ET 200S: Interface module IM 151-1; order no. 6ES7 151-2BAx2-0XB0 or higher.

Note

Modules which handle input conversion times (TCI) and output conversion times (TCO) (depending on the operating mode of the I/O submodule) cannot be operated in isochronous mode with Ti/To overlap. Overlap of Ti and To is only possible if these modules **are not** activated for isochronous mode.

See also

Setting PROFIBUS properties (Page 1406)

Operating principle of Ti/To overlapping (Page 1410)

Setting isochronous mode with overlap of Ti and To (Page 1411)

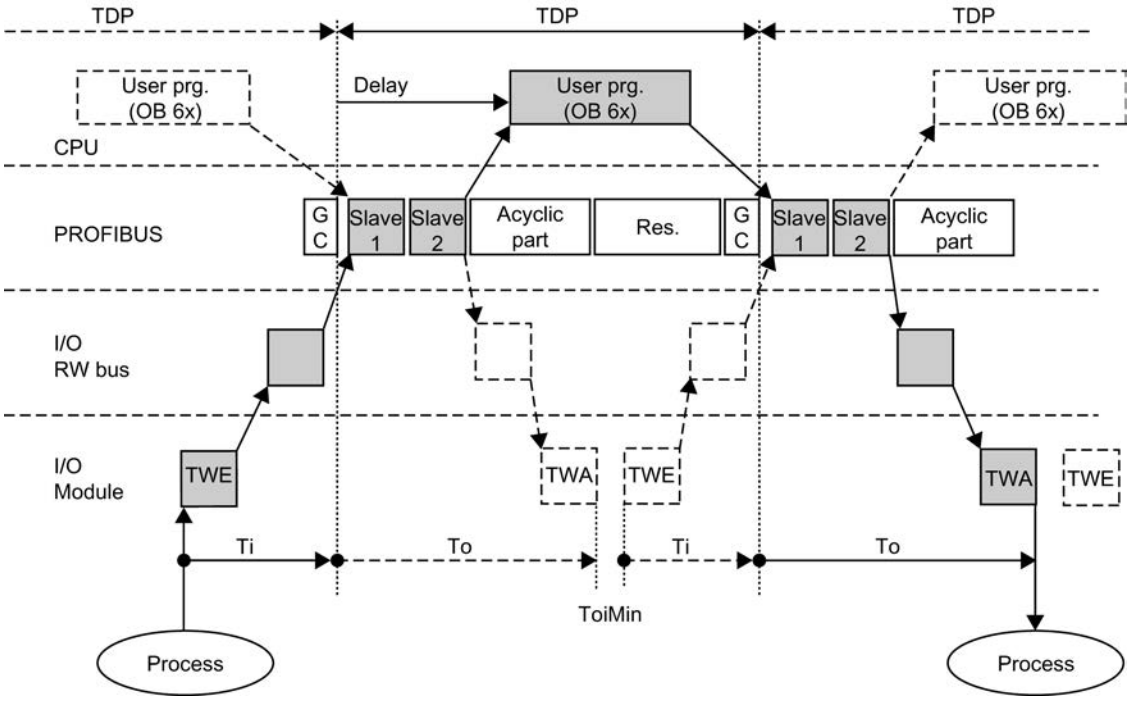
Operating principle of Ti/To overlapping

Overlapping processes

The function principle of Ti/To overlap is based on the fact that the peripheral input module already reads its inputs while the peripheral output module is still busy transferring the process reactions of the user program to its outputs. This functionality reduces the TDP system cycle and process reaction times.

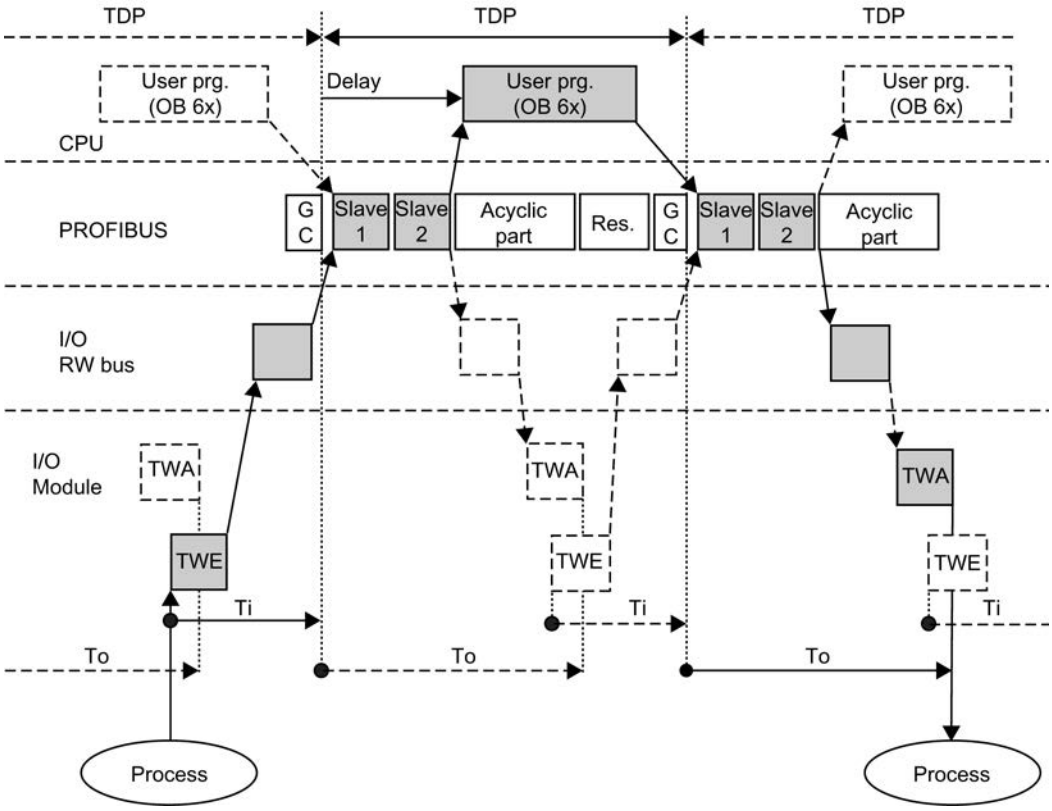
DP bus cycle without Ti/To overlap

The diagram below shows a constant DP bus cycle time in isochronous mode without Ti/To overlap.



DP bus cycle with Ti/To overlap

The diagram below shows a constant bus cycle time in isochronous mode with Ti/To overlap.



Rules for Ti, To and TDP

Rules for Ti, To and TDP:

- Ti + To ≤ TDP without overlap, that is: TDP ≥ Ti + To
- The conditions Ti + To > TDP or TDP < Ti + To may develop with overlap

See also

Ti/To overlap (Page 1409)

Setting isochronous mode with overlap of Ti and To

Configuring isochronous mode with Ti/To overlap

For isochronous mode with Ti/To overlap you must be aware of the following:

The DP slave for which Ti/To overlap is possible should not get its Ti/To values from the PROFIBUS subnet. For this DP slave select the option "Ti/To values automatic".

The best-possible values for Ti and To are set automatically, depending on the current set constant bus cycle time for DP cycle.

Rules for Ti/To overlap

Rules for operation in isochronous mode with Ti/To overlap:

- Ti ≤ TDP
- To ≤ TDP

Finding system reserves

Visualize the sequences using a calculation table (http://support.automation.siemens.com/WW/view/en/23876584) to determine system reserves and use the result to optimize these sequences in STEP 7.

Note

Disclaimer of liability

Siemens AG shall not be liable for material or immaterial damage caused by the use of the calculation table. Any liability of Siemens AG in particular for direct or consequential injury, damage to assets and financial losses sustained as a result of the use of the calculation table shall be ruled out.

Siemens does not provide support for the calculation table.

Note

Remember the rules governing the optimization of the constant bus cycle times and Ti/To overlap.

Additional information

Additional information for optimizing the constant bus cycle time are provided in the ET 200M operating instructions and in the ET 200S operating instructions.

See also

Ti/To overlap (Page 1409)

8.1.4.11 Configuring PROFIBUS DP (S7-1200)

The basics of configuring a DP master system

Distributed I/O

DP master systems that consist of a DP master and DP slaves which are connected via a bus and communicate with one another via the PROFIBUS DP protocol are referred to as distributed I/O.

Firmware version of the S7-1200 CPU

Use of the PROFIBUS functions with the S7-1200, requires CPUs with firmware version 2.0 or higher.

Configuring distributed I/O

Since DP masters and DP slaves are different devices, these instructions only provide a basic configuration procedure. However, the process for configuring distributed I/O is practically identical to that of non-distributed configuration.

Creating the DP master system in the network view

After you have used dragged a DP master and a DP slave (for example, a CM 1243-5 and a CM 1243-5) from the hardware catalog to the network view, connect them both to a PROFIBUS subnet.

Additional information

Observe additional information on the scope of functions in the manuals of the respective device.

DP slaves within the hardware catalog

DP slaves within the hardware catalog

You will find the DP slaves in the "Distributed I/O" folder of the hardware catalog. Compact and modular DP slaves are located there:

- Compact DP slaves
  Modules with integrated digital/analog inputs and outputs, for example, ET 200L
- Modular DP slaves
  (Interface modules with S7 modules assigned, for example, ET 200M

The available DP master and the desired functionality will determine which DP slaves can be used.

I slaves within the hardware catalog

The CM 1242-5 is, for example, an DP slave that can be configured as intelligent DP slave. You can find it in the hardware catalog at:

- CM 1242-5
  "PLC > SIMATIC S7 1200 > Communication module > PROFIBUS"

DP/DP coupler in the hardware catalog

Introduction

A DP/DP coupler connects two PROFIBUS DP networks as a gateway so that the DP master from one network can transfer data to the DP master of the other network.

The maximum amount of data that can be transferred is 244 bytes input data and 244 bytes output data.

DP/DP coupler in the hardware catalog

Details of a DP/DP coupler as gateway between two DM master systems are contained in the hardware catalog in the folder "Other field devices > PROFIBUS-DP > Gateways".

Configuring the DP/DP coupler

DP/DP couplers are configured in both PROFIBUS networks, each in their own master systems.

The input and output areas of both networks must thereby be matched to one another. The output data from one end of the DP/DP coupler are accepted as input data at the other respective end and vice versa.

Configurations involving PROFIBUS DP

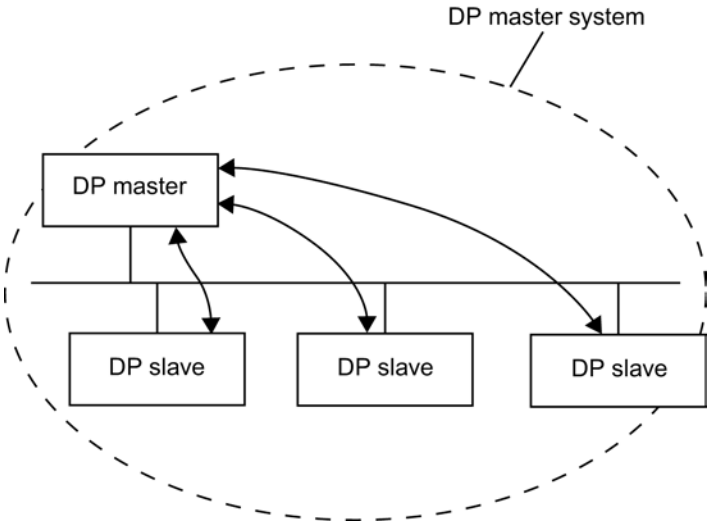
Configurations involving basic DP slaves

Communication between DP master and DP slave

In the case of a configuration involving simple DP slaves, data is exchanged between the DP master and simple DP slaves, i.e. with I/O modules via the DP master. The DP master sequentially polls each DP slave configured within the DP master system and contained in its polling list. In doing so, it transfers the output data to the slaves and receives their input data in return.

Mono-master system

The configuration with only one DP master is also described as mono-master system. A single DP master with its associated DP slaves is connected to a physical PROFIBUS DP subnet.



Configurations involving intelligent DP slaves

Definition

DP slaves that feature their own preprocessing program are referred to as intelligent DP slaves (I-slaves).

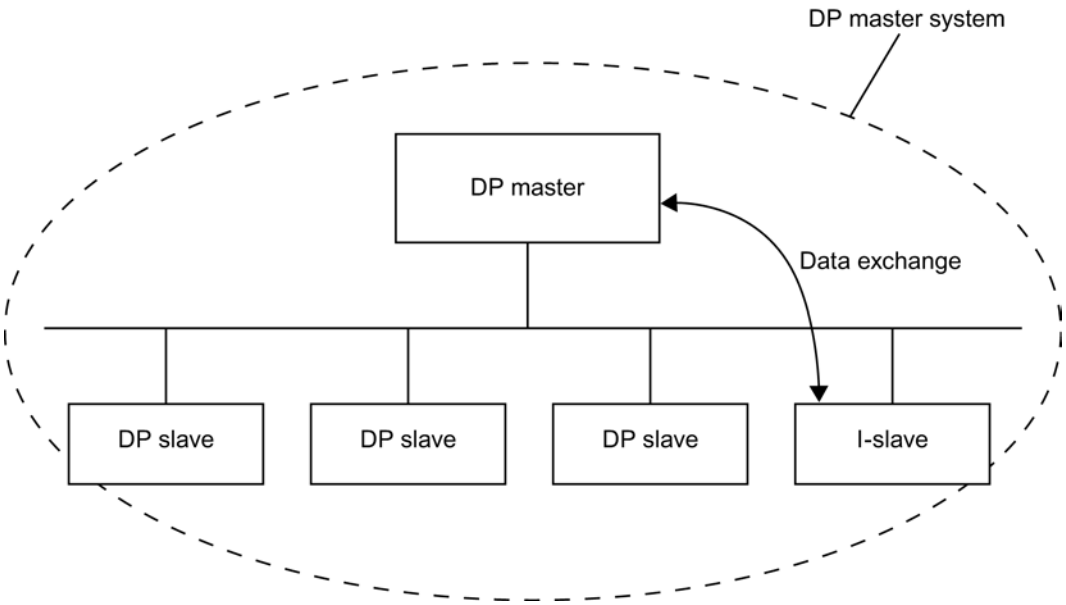
CM 1242-5 is an intelligent DP slave.

I-slave <> DP master data exchange

A higher-level automation system processes the automation task, which is broken down into sub-tasks. The sub-tasks are processed in the lower-level automation systems. The necessary control tasks are processed separately and efficiently in the CPUs as preprocessing programs.

In the case of configurations involving intelligent DP slaves, the DP master only accesses the operand area of the I-slave's CPU, and not the I/O modules of the intelligent DP slave. The operand area must not be assigned to any actual I/O modules in the I-slave. The assignment must be made during I-slave configuration.

The addresses of the data to be exchanged between the master and slave are configured in the transfer area of the I-slave.



Configuring distributed I/O systems

Hint: Quick configuration of master systems

If the DP master system has several DP slaves, use drag-and-drop operation to assign to the master in one step all DP slaves that were placed.

Requirements

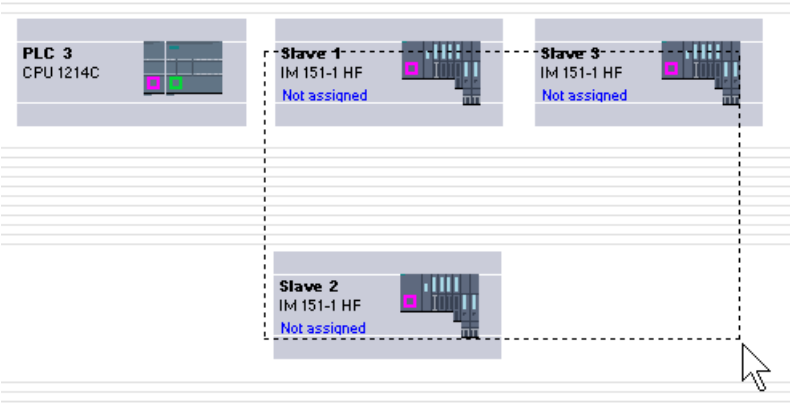
DP master and DP slaves are placed in the network view.

Assigning DP slaves to a DP master system

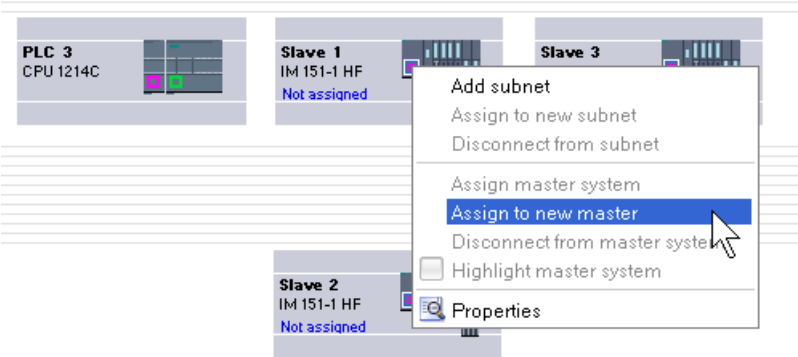
To do this, follow these steps:

1. Select an appropriate zoom factor so that you can see as many DP slaves as possible in the network view.
2. Arrange the DP slaves in a maximum of two rows.

3. Select all DP interfaces with the mouse cursor (not all devices!). This only works if you begin to drag the mouse cursor outside of the first DP slave and release the mouse button at the last DP slave (selection with the lasso).



4. Select the shortcut menu "Assign to new master" and select the corresponding DP interface for the DP master in the subsequent dialog.



5. The DP slaves are automatically networked with the DP master and combine with it to form a DP master system.

Note

When a DP master system is highlighted, you can double-click on a DP slave in the hardware catalog and thereby quickly add additional DP slaves. This will result in the DP slave being added to the highlighted DP master system automatically.

Creating a DP master system

Introduction

To create a DP master system, you need to have one DP master and at least one DP slave. As soon as you connect a DP master to a DP slave, a master-slave link is established.

DP master

You can use any of the following devices as a DP master:

- CM 1243-5

Requirement

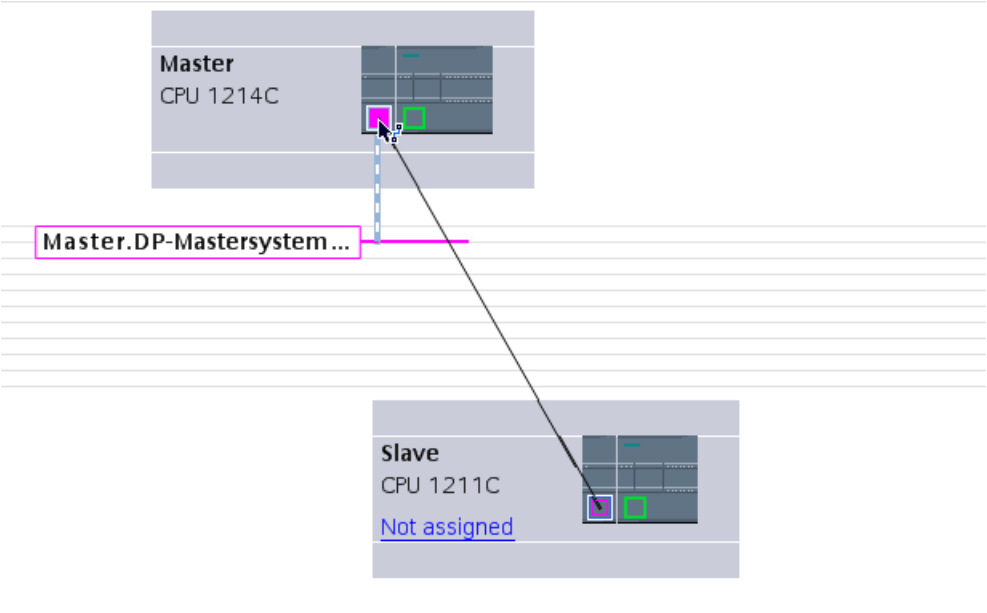
- You must be in the network view.
- The hardware catalog is open.

Procedure

To create a DP master system, follow these steps:

1. Select a DP master from the hardware catalog.
2. Pull the DP master onto the free area within the network view.
3. Right-click on the DP master's DP interface.
4. Select "Create master system" from the shortcut menu.
   A DP master system with one DP master will be created as a single node.

If you connect a DP slave's DP interface to that of the DP master, the DP slave will be added to the master system.



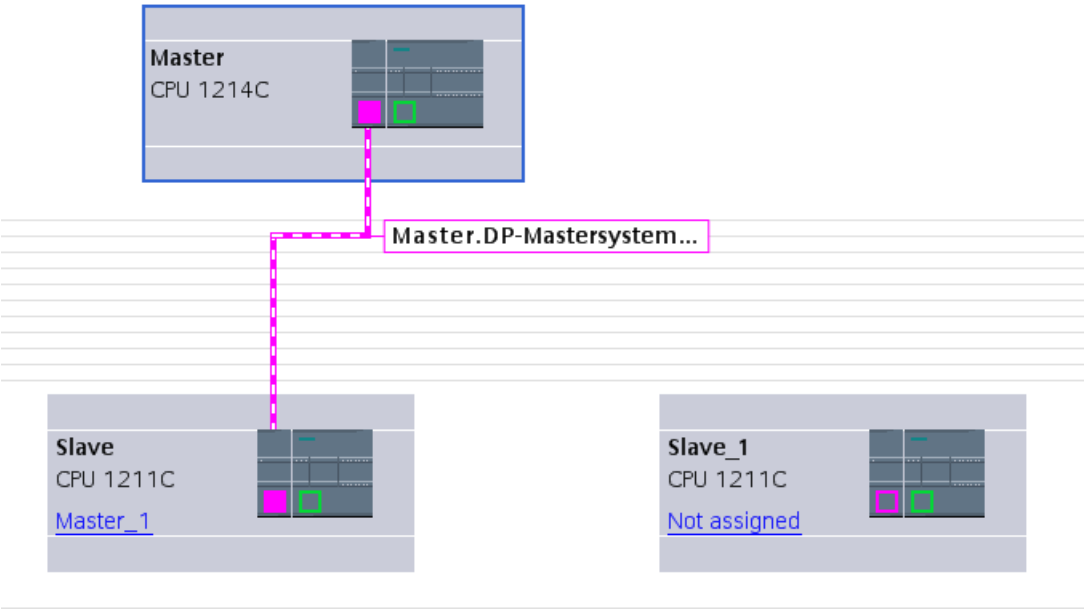
Assuming that you have already placed both a DP master and a DP slave within the network view, you can drag-and-drop to connect the two and thereby create a DP master system. To do so, follow these steps:

1. Click on the DP interface of either the DP master or DP slave.
2. Hold down the mouse button and draw a connecting line between this DP interface and that of the desired communication partner.

This will create a subnet with one DP master system between the DP master and DP slave.

DP master display on the DP slave

When you connect a DP slave to a DP master, the name of the DP master is displayed on the DP slave as a hyperlink. Click the hyperlink to select the associated DP master.

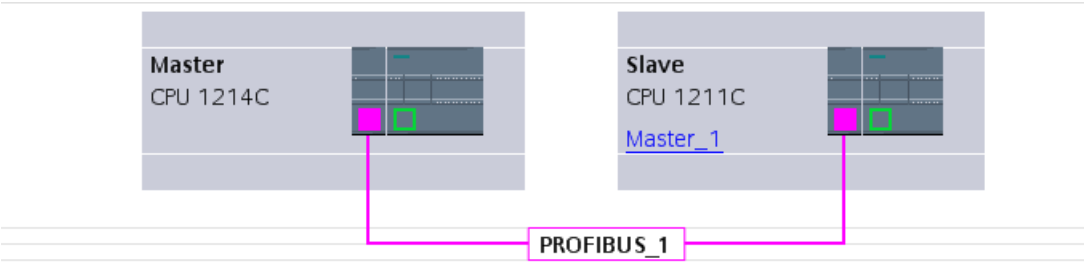


Highlighting applied to the DP master system

When you create a new DP master system, highlighting will be applied to it. This enables you to identify quickly which devices belong to the DP master system. You can also highlight a DP master system yourself by moving the mouse pointer over a subnet. This will result in the names of the available DP master systems being displayed. Click the required DP master system to highlight it.

There are various ways of removing the highlighting from a DP master system:

- Highlight a different master system.
- Click on the drawing pin with the name of the master system in the top right-hand corner of the network view.



Editing DP master systems and interfaces

Introduction

Once you have created a DP master system, you also have the option of disconnecting the DP master system from its components. This can result in subnets with DP slaves but without DP master.

Generally, there is no need to edit the interfaces of a DP master.

You can change the name and number on the DP master system.

Disconnection of master or slaves from the DP master system

If you have configured a PROFIBUS CP as a DP master with master system, you can then disconnect the DP master system from the DP master. After this, the device will no longer be connected to the DP master system.

Disconnecting the subnet from a DP master effectively eliminates the master system in the sense that it is no longer assigned to a DP master. However, the individual DP slaves are still interconnected via the subnet.

If you delete the DP slaves or disconnect them from the master system, the master system is then retained on the DP master.

Requirement

- You must be in the network view.
- There has to be a DP master system with one DP master and at least one DP slave.

Disconnecting the DP master from the DP master system

To disconnect the DP master system, proceed as follows:

1. Right-click on the DP master's DP interface.
2. Select "Disconnect from master system" from the shortcut menu.

The selected DP master will be disconnected from the DP master system. A subnet with the DP slaves will be retained.

Adding a DP master to the DP master system

To reassign a DP master to a subnet, proceed as follows:

1. Right-click on the DP master's DP interface.
2. Select "Create master system" from the shortcut menu.
3. Draw connecting lines between the new DP master system and the DP interfaces of the DP slaves.

The DP master combines with the DP slaves to recreate a DP master system.

Editing the properties of a DP master system

To edit the properties of a DP master system, proceed as follows:

1. Move the mouse pointer over a subnet with a DP master system.
2. A message will appear displaying the available DP master systems. Click the one you want to edit. The DP master system will now be color-highlighted.
3. Click on the highlighted DP master system.
4. In the inspector window, edit the DP master system attributes under "Properties > General".

Note

If you click a subnet when no DP master system is highlighted, you will be able to edit the properties of the entire subnet under "Properties" in the inspector window.

Adding DP slaves to the master system and configuring them

In the network view, you can add various DP slaves from the hardware catalog directly by using the drag-and-drop function or by double-clicking.

DP slaves

For configuration purposes, DP slaves are broken down into the following categories:

- Compact DP slaves
  (Modules with integrated digital/analog inputs and outputs, for example, ET 200L)
- Modular DP slaves
  (interface modules with S5 or S7 modules assigned, for example ET 200M)
- Intelligent DP slaves (I slaves)
  (CM 1242-5 or ET 200S with IM 151-7 CPU)

Rules

- Your DP master system should only contain one DP master, but it may contain one or or more DP slaves.
- You may only configure as many DP slaves in a DP master system as are permitted for the specific DP master.

Note

When configuring the DP master system, remember to observe the DP master technical data (max. number of nodes, max. number of slots, max. quantity of user data). User data restrictions may possibly prevent you from being able to use the maximum number of nodes that is theoretically possible.

Requirements

- You must be in the network view.
- A DP master system must have been created.

Adding a DP slave to the DP master system

To add a DP slave from the hardware catalog to the DP master system, follow these steps:

1. Select a DP slave from the hardware catalog.
2. Drag-and-drop the DP slave from the hardware catalog into the network view.
3. Draw a connecting line between the DP master's DP interface or a highlighted DP master system and the DP interface of the new DP slave.

A DP master system will be created and the DP slave will be connected to the DP master automatically.

Note

When a DP master system is highlighted, you can double-click the required DP slave in the hardware catalog. This will result in the DP slave being added to the highlighted DP master system automatically.

Disconnecting a DP slave from the DP master system

To disconnect a DP slave from the DP master system, follow these steps:

1. In the network view, right-click on the DP slave's DP interface.
2. From the shortcut menu, select the method for disconnecting from the DP master system:
   - "Disconnect from subnet": The PROFIBUS connection is broken and the device is no longer connected to the DP master system or a subnet.
   - "Disconnect from master system": The DP slave remains connected to the subnet, but is no longer assigned to the DP master system as a DP slave.

The selected DP slave will be disconnected from the DP master system.

Assigning a DP slave to a new DP master system

To assign an existing DP slave to a new DP master system, follow these steps:

1. Right-click on the DP slave's DP interface.
2. From the shortcut menu, select "Assign to new master".
   It does not matter whether the DP slave concerned is already assigned to another DP master system.
3. From the selection list, select the DP master whose DP master system is to have the DP slave connected to it.
   The selected DP slave is now assigned to a new DP master system.

The "Assign to new subnet" function works in a similar way in that it allows you to connect a DP slave to a new subnet. However, in this case, the DP slave will not be connected to an existing DP master system.

Configuring a DP slave

To configure a DP slave, follow these steps:

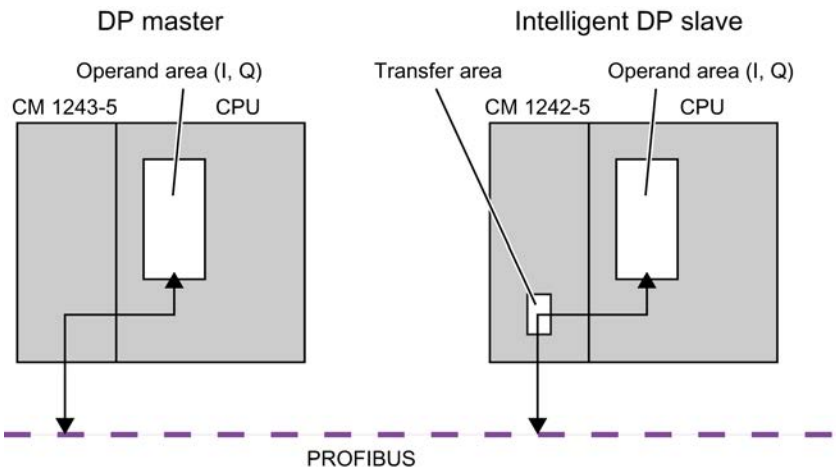
1. Switch to the DP slave's device view.
2. Select the module you want.
3. Configure the DP slave in the Inspector window.

Configuring intelligent DP slaves

Adding an I-slave to a DP master system

Introduction

One of the characteristics of an intelligent DP slave (I-slave) is that the DP master is not provided with I/O data directly by a real I/O, but by a preprocessing CPU. Together with the CP, this preprocessing CPU forms the I-slave.



Difference: DP slave v. intelligent DP slave

In the case of a DP slave, the DP master accesses the distributed I/O directly.

In the case of an intelligent DP slave, the DP master actually accesses a transfer area in the I/O address space of the preprocessing CPU instead of accessing the connected I/O of the

intelligent DP slave. The user program running on the preprocessing CPU is responsible for ensuring data exchange between the address area and I/O.

Note

The I/O areas configured for data exchange between the DP master and DP slaves must not be used by I/O modules.

Applications

Configurations involving intelligent DP slaves: I-slave <> DP master data exchange

Procedure

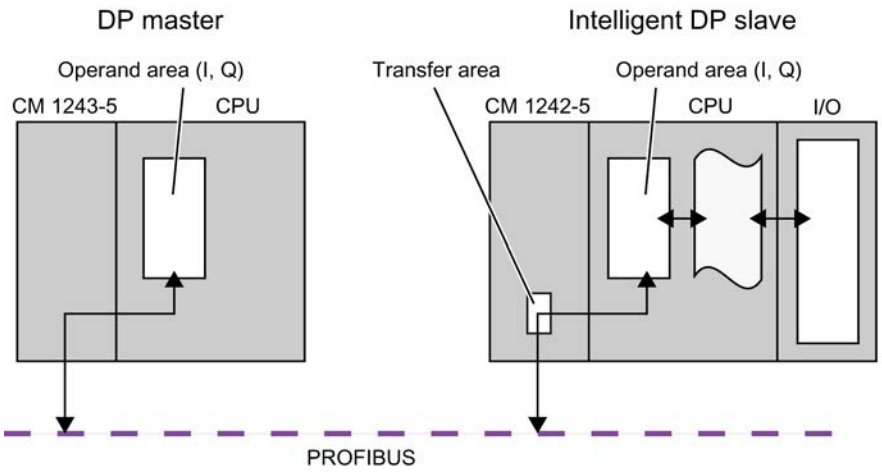
To add an I-slave to a DP master system, follow these steps:

1. In the network view, drag from the hardware catalog to a station one CM 1242-5 as an I-slave and one CM 1243-5 as a DP master.
2. Draw a connecting line between the DP interfaces of both devices.
   This way you connect the I-slave with a DP master in a DP master system.
   Result: You have now set up a DP master system with one DP master and one I-slave.

Configuring access to I slave data

Data access

The following applies to the CP 1242-5 in its function as I-slave: The addresses for the data transfer area and the address for the I/O modules in the I-slave differ. This means that the start address occupied by an I/O module can no longer be used for the transfer memory. If the higher-level DP master is to access the data of an I/O module in the I-slave, you must configure this data exchange between the I/O module and transfer area in the I-slave user program.



Configuration of the transfer area for the CM 1242-5 (transfer area)

With CM 1242-5, the transfer area for the cyclic PROFIBUS data exchange is configured as transfer area in the parameter group "PROFIBUS interface > Mode > I Slave Communication".

Direct data access from CPU to CPU

Direct data access from CPU to CPU via PROFIBUS is supported by the S7-1200 PROFIBUS CMs only via the PUT/GET services.

Configuring DP slaves as distributed I/O devices

Configuring an ET 200S

Slot rules for configuring an ET 200S

The following rules apply when configuring an ET 200S:

- Do not leave any gaps when inserting the ET 200S modules.
- Slot 1: only for PM-E or PM-D Power Modules.
- To the left of an Electronics Module (EM): an EM or a Power Module (PM-E or PM-D) only.
- To the left of Motor Starter (MS): an MS, a PM-D, PM-D Fx (1..x..4) Power Module or a PM-X Power Module only.
- To the left of a PM-X: a motor starter or a PM-D only
- Up to 63 modules and one IM Interface Module are permitted

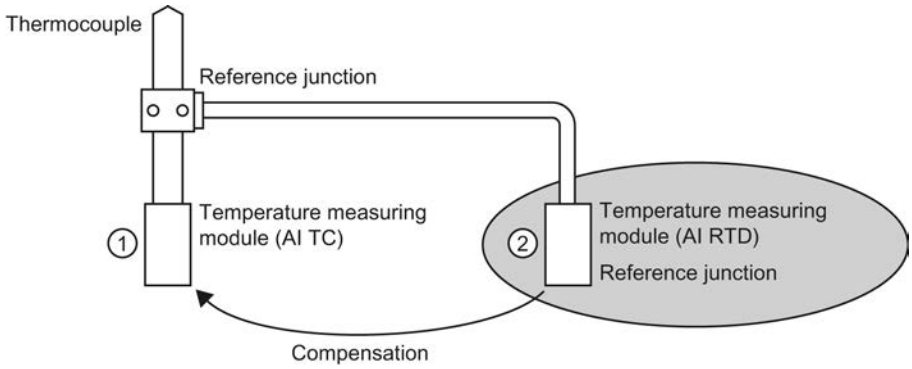
Note

Remember to ensure that the correct PM-E and EM voltage ranges are assigned.

Configuring reference junctions

A reference junction is the connection of a thermocouple to a supply line (generally in the terminal box). The voltage that occurs due to the effects of temperature falsifies the temperature value measured by the module.

On the ET 200S, one channel of the AI RTD module can be programmed as a reference junction. Other AI TC modules can correct their measured values using the temperature at the reference junction as measured by this module.



① Configuring the AI TC:
→ Selection of the reference junction used

② Configuring of the AI RTD:
→ Activation of the reference junction
→ Specifying the slot and channel of the AI RTD

Special characteristics to be noted when assigning parameters for reference junctions

The process of assigning parameters for reference junctions will be explained based on the example of a resistance thermometer with a Pt 100 climatic range that is used for measuring the reference junction temperature.

To assign parameters for the reference junction, follow these steps:

1. In the ET 200S device view, insert an analog electronics module, for example a 2AI RTD HF.
2. Select the module on the rack.
3. Under "Properties > Inputs" in the inspector window, set a channel for the reference junctions function to the "RTD-4L Pt 100 climatic range" measuring range.
4. Select the ET 200S.
5. Under "Properties > Module parameters > Reference junctions" in the inspector window, select the "Reference junction" check box and specify the slot and channel number of the relevant RTD module.
6. Insert the analog electronics module for measuring the temperature using a thermocouple (TC module) and parameterize it with the reference junction number of the RTD module.

Additional information

For additional information on the various types and uses of ET 200S modules, please refer to the operating instructions and the manual titled "ET 200S Distributed I/O System".

For additional information on analog value processing, please see the documentation for the ET 200S distributed I/O system.

Packing addresses

Introduction

DP slaves and I/O devices from the ET 200S family are configured in the same way as other modular DP slaves and I/O devices. As well as supporting all the standard modular DP slave and I/O device functions, the ET 200S also offers the "Pack addresses" function:

When digital electronics modules requiring an address space of 2 or 4 bits are inserted into the device view, they will initially be spread over a total area of 1 byte. However, the address area actually occupied can be compressed after configuration using the "Pack addresses" function.

	Initial state	After "Pack addresses"
Module	I address	I address
2DI (2-bit)	Byte 10	10.0...10.1
4DI (4-bit)	Byte 11	10.2...10.5

Requirements

- You are in the device view.
- An ET 200S, for example an IM 151-1, must be present.
- A pair of digital electronics modules, for example 2DI AC120V ST, must be inserted into the slots.

Packing addresses

To pack addressed, follow these steps:

1. Select the electronics modules whose addresses are to be packed. The following options are available for selecting multiple electronics modules:
   – Press and hold down <Shift> or <Ctrl> while clicking the relevant electronics modules.
   – Click off the rack and select the required electronics modules by drawing round them with the mouse.
2. Click "Pack addresses" in the shortcut menu for the selected electronics modules.

The address areas for inputs, outputs and motor starters are packed separately. The packed addresses will be displayed in the I address and Q address columns of the device overview.

How packed addresses are generated and structured

If you make use of the "Pack addresses" function, the addresses of the selected electronics modules will be packed in accordance with the following rules:

- The start of the address area is determined by the lowest address of the selected electronics modules: X.0
- If the bit address is not "0", then the next (free) byte address as of which the selected area can be compacted will be selected automatically: (X+n).0
- If no coherent area exists, the addresses will be automatically packed into any available address gaps.

Electronics modules with packed addresses and the same byte address form a packing group.

Unpacking addresses

To unpack addressed, follow these steps:

1. Select one or more electronics modules with packed addresses.
2. Click "Unpack addresses" in the shortcut menu for the selected electronics modules.

The packing groups of the selected electronics modules will be disbanded and the packed addresses for the relevant electronics modules unpacked.

The packing group will also be disbanded and the packed addresses unpacked in the following cases: if you delete electronics modules from a packing group, move electronics modules out of a packing group or insert electronics modules on a free slot within a packing group.

The start addresses of the unpacked electronics modules will be assigned to the next available byte addresses in each case.

Special characteristics of electronics modules with packed addresses

The following special characteristics apply to electronics modules with packed addresses:

- As far as the CPU is concerned, there is no way of assigning a slot for the electronics module. Consequently, the instruction GADR_LGC (SFC 5) outputs error information W#16#8099 "Slot not configured" for the actual slot of the electronic module.
- The instruction LGC_GADR (SFC 49) and SZL-ID W#16#xy91 "module status information" cannot be evaluated for an electronics module.
- The electronics module receives an additional diagnostics address via the DPV1 function, because otherwise the packed addresses would prevent interrupts from being assigned as far as the CPU is concerned.
- The "Insert/remove interrupt" is not possible. This is because the "Pack addresses" and "Insert/remove interrupt" functions are mutually exclusive.

Configuring option handling with standby modules

You can use the option handling function to prepare the ET 200S with PROFIBUS interface for future expansions (options). This section describes option handling with standby modules.

You do this by assembling, wiring, configuring, and programming the maximum configuration envisaged for the ET 200S and by using cost-effective standby modules (138-4AA00 or 138-4AA10) during assembly until it becomes time to replace them with the necessary electronics modules.

Note

The ET 200S can be completely factory-wired with the master cabling, as no connection exists between a standby module and the terminals of the terminal module (and, in turn, to the process).

Requirement

- ET 200S interface module
  - IM 151-1 STANDARD (6ES7 151-1AA03-0AB0 or higher)
  - IM 151-1 FO STANDARD (6ES7 151-1AB02-0AB0 or higher)
- Power module with option handling
  - PM-E DC24..48V
  - PM-E DC24..48V/AC24..230V

Procedure

To activate option handling, follow these steps:

1. Select the IM 151-1 in the device view and enable it in "Option handling" check box under "Properties > General > Option handling" in the inspector window.
2. Select the numbered check boxes for the slots that are initially to accommodate the standby modules prior to the future electronics modules.
3. Select the power module in the device view and enable it in the "Option handling" check box under "Properties > Addresses" in the inspector window. Reserve the necessary address space for the control and check-back interface in the process image output (PIQ) and process image input (PII).

The assembled standby modules can be replaced with the configured modules at a later date without having to modify the configuration.

Note

The addresses for these interfaces are reserved as soon as you activate option handling on the power module. The "Option handling" function must also be activated on the DP slave (IM 151-1 STANDARD Interface Module). If it is not activated, the addresses reserved for the control and check-back interface will be released again.

Note that activating and deactivating the option handling function repeatedly can change the address of the control and check-back interface.

Option handling may be activated for one PM-E DC24..48V or one PM-E DC24..48V/AC24..230V Power Module only.

Additional information

For additional information on the assignment and significance of bytes within the process image, option handling with PROFIBUS and the use of standby modules, please refer to the documentation for the ET 200S distributed I/O system.

How option handling works during startup

If "Startup when expected/actual config. differ" is not available, the ET 200S will still start up if a standby module is inserted instead of the configured electronics module and option handling has been activated for the slot concerned.

How option handling works during operation

During operation, the option handling mode varies in accordance with the following:

- Option handling enabled for a slot:
  Either the standby module (option) or the configured electronics module can be plugged into this slot. If any other kind of module is present on this slot, diagnostics will be signaled (no module/incorrect module).
- Option handling disabled for a slot:
  Only the configured electronics module can be plugged into this slot. If any other kind of module is present, diagnostics will be signaled (no module/incorrect module).

Standby module substitute values

- Substitute value for digital inputs: 0
- Substitute value for analog inputs: 0x7FFF

Control and evaluation in the user program

The ET 200S is equipped with a control and feedback interface for the "Option handling" function.

The control interface is located in the process image output (PIQ). Each bit in this address area controls one of the slots from 2 to 63:

- Bit value = 0: Option handling parameterized. Standby modules are permitted.
- Bit value = 1: Option handling canceled. Standby modules are not permitted on this slot.

The check-back interface is located in the process image input (PII). Each bit in this address area provides information about the modules that are actually plugged into slots 1 to 63:

- Bit value = 0: This slot has the standby module, an incorrect module or no module plugged into it.
- Bit value = 1: The configured module is plugged into this slot.

See also

Which modules support option handling? (http://support.automation.siemens.com/WW/view/en/22564754)

Configuring option handling without standby modules

You can use the option handling function to prepare the ET 200S for future expansions (options) without installing standby modules. This section describes option handling without standby modules.

Note

ET 200S with PROFINET interface

This description refers to the ET 200S with PROFIBUS interface. In principle, option handling for ET 200S with PROFINET interface functions as described here without standby modules. PN interface modules are to be used instead of the DP interface modules listed here. You can find additional information about option handling for ET 200S with PROFINET interface in the corresponding manuals.

Requirement

- ET 200S interface module
  - IM 151-1 HIGH FEATURE (6ES7151-1BA02 or higher)
  - IM 151-1 STANDARD (6ES7 151-1AA05-0AB0 or higher)
- Power module with option handling
  - PM-E DC24..48V
  - PM-E DC24..48V/AC24..230V

Procedure

To activate option handling, follow these steps:

1. Select the IM 151-1 in the device view and enable it in "Option handling" check box under "Properties > General > Option handling" in the inspector window.
2. Select the power module in the device view and enable it in the "Option handling" check box under "Properties > Addresses" in the inspector window. Reserve the necessary address space for the control and check-back interface in the process image output (PIQ) and process image input (PII).
3. Configure the slave's maximum configuration. The selection/clearing of options is controlled via the user program.

Note

The addresses for these interfaces are reserved as soon as you activate option handling on the power module. The "Option handling" function must also be activated on the DP slave (IM 151-1 interface module). If it is not activated, the addresses reserved for the control and check-back interface will be released again.

Note that activating and deactivating the option handling function repeatedly can change the address of the control and check-back interface.

Option handling may be activated for one PM-E DC24..48V or one PM-E DC24..48V/AC24..230V Power Module only.

Additional information

For additional information on the assignment and significance of bytes within the process image, option handling with PROFIBUS and the use of standby modules, please refer to the documentation for the ET 200S distributed I/O system.

Control and evaluation in the user program

The ET 200S is equipped with a control and feedback interface for the "Option handling" function.

The control interface is located in the process image output (PIQ). Each bit in this address area controls one of the slots from 1 to 63:

- Bit value = 0: Slot is not available in the actual configuration.
- Bit value = 1: Slot is available in the actual configuration.

The check-back interface is located in the process image input (PII). Each bit in this address area provides information about the modules that are actually plugged into slots 1 to 63:

- Bit value = 0: Slot belongs to an option that is not available or module status is faulty.
- Bit value = 1: The configured module is plugged into this slot.

See also

Sample applications for ET 200S, option handling without standby module (http://support.automation.siemens.com/WW/view/en/29430270)

Configuring the ET 200S in DPV1 mode

PROFIBUS DPV1 enables you to access extended PROFIBUS functions.

Requirement

- You must be in network view.
- A DP master with DPV1 functionality must be available.
- A master-slave connection must be established with PROFIBUS.

Procedure

To switch the DP slave over to DPV1, follow these steps:

1. Select the DP slave.
2. Under "Properties > Module parameters" in the Inspector window, select "DPV1" mode from the "DP interrupt mode" drop-down list.

or

1. Select the DP master.
2. In the I/O communications table, select the row with the connection between the DP master and the desired DP slave.
3. Under "Properties > Module parameters" in the Inspector window, select "DPV1" mode from the "DP interrupt mode" drop-down list.

Special characteristics

The parameters are subject to interdependencies, which are outlined below:

Parameter	DPV0 mode	DPV1 mode
Operation when target configuration does not match actual configuration	Fully available	Fully available
Diagnostics interrupt (OB 82)	Not available, not set	Fully available
Hardware interrupt (OB 40 to 47)	Not available, not set	Fully available
Insert/remove interrupt (OB 83)	Not available, not set	Only available when addresses are not packed. "Startup when target configuration does not match actual configuration" is activated automatically along with an insert/remove interrupt.

Interrupts in the case of modules with packed addresses

If the module is capable of triggering interrupts and the bit address is not equal to 0 because of packed addresses, you will need to assign a diagnostics address in the ET 200S address dialog.

The diagnostics address is required for the purpose of assigning a DPV1 interrupt to the module as an interrupt trigger. The CPU will only be able to assign an interrupt correctly and store information on the interrupt in the interrupt OB start information/in the diagnostics buffer if the module concerned has this "unpacked" address. In this context, the CPU cannot make use of "packed" addresses.

From the point of view of interrupt processing (interrupt OB), this means the module will have the assigned diagnostics address, but from the point of view of processing the input and output data in the user program, the module will have the packed addresses.

Note

When the module addresses are packed, the insert/remove interrupt for the ET 200S is unavailable.

Using GSD files

GSD revisions

What you need to know about GSD revisions

The properties of DP slaves are made available to configuration tools by means of GSD files.

Functional enhancements in the area of the distributed I/O will have an effect on the GSD specification, for example, they will require the definition of new keywords.

This results in the versioning of the specification. In the case of GSD files, the version of the specification on which a GSD file is based is called a "GSD revision".

From GSD revision 1, the GSD revision must be included as a keyword "GSD_revision" in GSD files. GSD files without this keyword will therefore be interpreted by configuration tools as GSD revision "0".

GSD files can be interpreted up to GSD revision 5. This means that DP slaves that support the following functions, for example, will be supported:

- Diagnostic alarms for interrupt blocks
- Isochronous mode and constant bus cycle time
- SYNC/FREEZE
- Clock synchronization for DP slaves

Installing the GSD file

Introduction

A GSD file (device data file) contains all the DP slave properties. If you want to configure a DP slave that does not appear in the hardware catalog, you must install the GSD file provided by the manufacturer. DP slaves installed via GSD files are displayed in the hardware catalog and can then be selected and configured.

Requirement

- The hardware and network editor is closed.
- You have access to the required GSD files in a directory on the hard disk.

Procedure

To install a GSD file, proceed as follows:

1. In the "Options" menu, select the "Install device master data files" command.
2. In the "Install device master data files" dialog box, choose the folder in which you want to save the GSD files.
3. Choose one or more files from the list of displayed GSD files.
4. Click on the "Install" button.
5. To create a log file for the installation, click on the "Save log file" button.
   Any problems during the installation can be tracked down using the log file.

You will find the new DP slave installed by means of the GSD file in a new folder in the hardware catalog.

Note

Installation of GSD file cannot be undone.

Configuring GSD-based DP slave

DP slaves that you have inserted through installation of a GSD file can be selected as usual via the hardware catalog and inserted in the network view. If you want to insert the modules of the GSD-based DP slaves, you must take into account some particular details.

Requirement

- You have installed a DP slave using a GSD file.
- You have inserted the head module in the network view in the usual manner.
- The device overview opens in the device view.
- The hardware catalog is open.

Procedure

To add the modules of a GSD-based DP slave, proceed as follows:

1. In the hardware catalog, navigate to the modules of the GSD-based DP slave.
   GSD-based DP slaves, also referred to as DP standard slaves, can be found in the "Other field devices" folder of the hardware catalog.
2. Select the desired module.

3. Use drag-and-drop to move the module to a free space in the device overview.
4. Select the module in the device overview to edit parameters.

You have now inserted the module in a free slot of the GSD-based DP slave and can edit its parameters.

Note

You can see only the GSD-based DP slave in the graphic area of the device view. The added modules of GSD-based DP slaves are only found in the device overview.

Preset configuration

For modules with an adjustable preset configuration, you can change this configuration in the inspector window under "Properties > Preset configuration".

8.1.4.12 Configurations for PROFINET IO

What you need to know about PROFINET IO

What is PROFINET IO?

PROFINET IO

PROFINET is an Ethernet-based automation standard of PROFIBUS Nutzerorganisation e.V. (PNO) which defines a manufacturer-neutral communication, automation and engineering model.

Objective

The objective of PROFINET is:

- Integrated communication via field bus and Ethernet
- Open, distributed automation
- Use of open standards

Architecture

The PROFIBUS User Organisation e.V. (PNO) has designated the following aspects for PROFINET architecture:

- Communication between controllers as components within distributed systems.
- Communication between field devices, such as I/O devices and drives.

Implementation by Siemens

The demand for "Communication between controllers as components within distributed systems" is implemented by "Component Based Automation" (CBA). Component Based Automation is used to create a distributed automation solution based on prefabricated components and partial solutions.

The demand for "Communication between field devices" is implemented by Siemens with "PROFINET IO". Just as with PROFIBUS DP, the complete configuration and programming of the components involved is possible using the Totally Integrated Automation Portal.

The following sections deal with the configuration of communication between field devices using PROFINET IO.

Overview of RT classes

RT classes in PROFINET IO

PROFINET IO is a scalable, real-time communication system based on Ethernet technology. The scalable approach is reflected in several real-time classes:

- **RT:** Transmission of data in prioritized, non-isochronous Ethernet frames. The required bandwidth is within the free bandwidth area for TCP/IP communication.
- **IRT:** Isochronous transmission of data with high stability for time-critical applications (for example, motion control). The required bandwidth is from the area of bandwidth reserved for cyclic data.

Depending on the device, not all real-time classes are supported.

Connecting existing bus systems

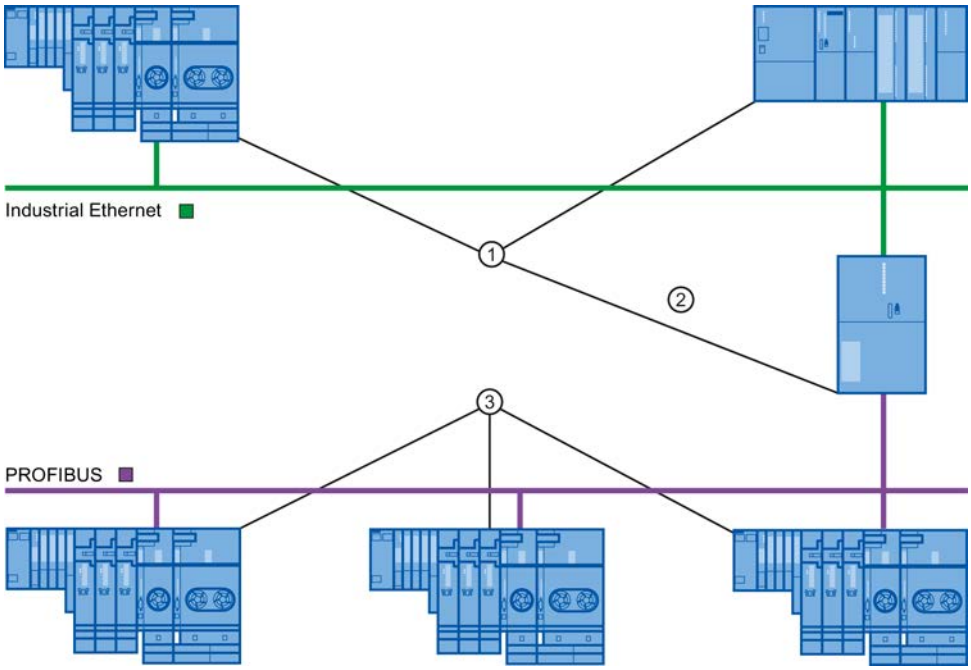
Connection of PROFINET and PROFIBUS

PROFINET IO and PROFIBUS DP can be connected with each other as follows:

- via Industrial Ethernet:
  To connect the two network types, Industrial Ethernet (plant control level) and PROFIBUS (cell level/field level), use the IE/PB link, for example.
- via Industrial Wireless LAN:
  PROFIBUS devices, for example, can be connected to PROFINET IO via a wireless LAN/PB link. This allows existing PROFIBUS configurations to be integrated into PROFINET.

AS interface devices can be connected by an IE/AS-i link PN IO to the interface of a PROFINET device. This allows the existing AS-i network to be integrated into PROFINET.

The following figure shows the connection of a PROFIBUS subnet via a PROFINET device with proxy functions.



① PROFINET devices
② PROFINET device with proxy functions (for example, IE/PB link)
③ PROFIBUS devices

PROFINET device with proxy functions used as proxy for a PROFIBUS device

The PROFINET device with proxy functions is the proxy for a PROFIBUS device on the Ethernet. The proxy functionality enables a PROFIBUS device to communicate with all devices on the PROFINET and not just with its master.

Existing PROFIBUS systems can easily be integrated into PROFINET communication using the proxy functions.

If, for instance, you connect a PROFIBUS device via an IE/PB link to PROFINET, the IE/PB link acts as a proxy for the PROFIBUS components for communicating via PROFINET.

Configuration using IE/PB link PN IO

Configuration using IE/PB Link PN IO

You can use the IE/PB Link PN IO to connect PROFIBUS DP configurations to PROFINET IO.

From the CPU perspective, the PROFIBUS DP slaves are connected to the same network as the IE/PB Link PN IO. These slaves have the same device name and IP address as the IE/PB Link PN IO, but different device numbers. Furthermore, each also has a specific PROFIBUS address.

In the properties of the IE/PB link, the PROFIBUS addresses of the connected DP slaves are displayed in addition to the PROFINET device numbers, as this device has two addressing schemes.

Handling device numbers and PROFIBUS addresses on the master system

During placement, the same number is assigned for the PROFINET device number and the PROFIBUS address.

In the Inspector window under "General Properties > PROFINET device number", you can find an overview of the device numbers used and the PROFIBUS addresses of an IE/PB link. The device number can also be changed here. You can also set that the device number and the PROFIBUS address should or should not always be identical. If the "PROFINET device number=PROFIBUS address" option is activated, you do not have to update the device number when the PROFIBUS address changes.

The PROFIBUS address can be changed in the properties of the PROFIBUS device.

Restrictions

The following restrictions apply to DP slaves configured as described above on the PROFIBUS subnet of an IE/PB link:

- No pluggable IE/PB link
- No pluggable DP/PA link
- No pluggable Y link
- Not CiR-compliant
- No pluggable redundant slaves
- No isochronous transmission / constant bus cycle time can be configured
- SYNC/FREEZE instructions ("DPSYC_FR") of a CPU on the the Ethernet subnet for DP slaves behind the IE/PB-Link are not supported.

See also

Modes of the IE/PB Link PN IO (Page 2003)

Configuration using IWLAN/PN link

Maximum number of devices in a IWLAN segment

If an Ethernet subnet is set up as wireless network (IWLAN = Industrial Wireless LAN), cyclic data exchange between IO controllers and IO devices is possible via a wireless route.

On one side of the wireless route there are fixed installed access points (for example, SCALANCE W 788) and on the other side mobile stations (with, for example IWLAN/PB links with PROFIBUS devices).

If the action radius of the mobile stations is large, it may be necessary to install several access points (SCALANCE W 788). Since each access point forms a segment with its wireless range, the IWLAN is made up of a series of segments.

The mobile devices "on the one side" of the wireless link with their IWLAN/PB links can move along the segments.

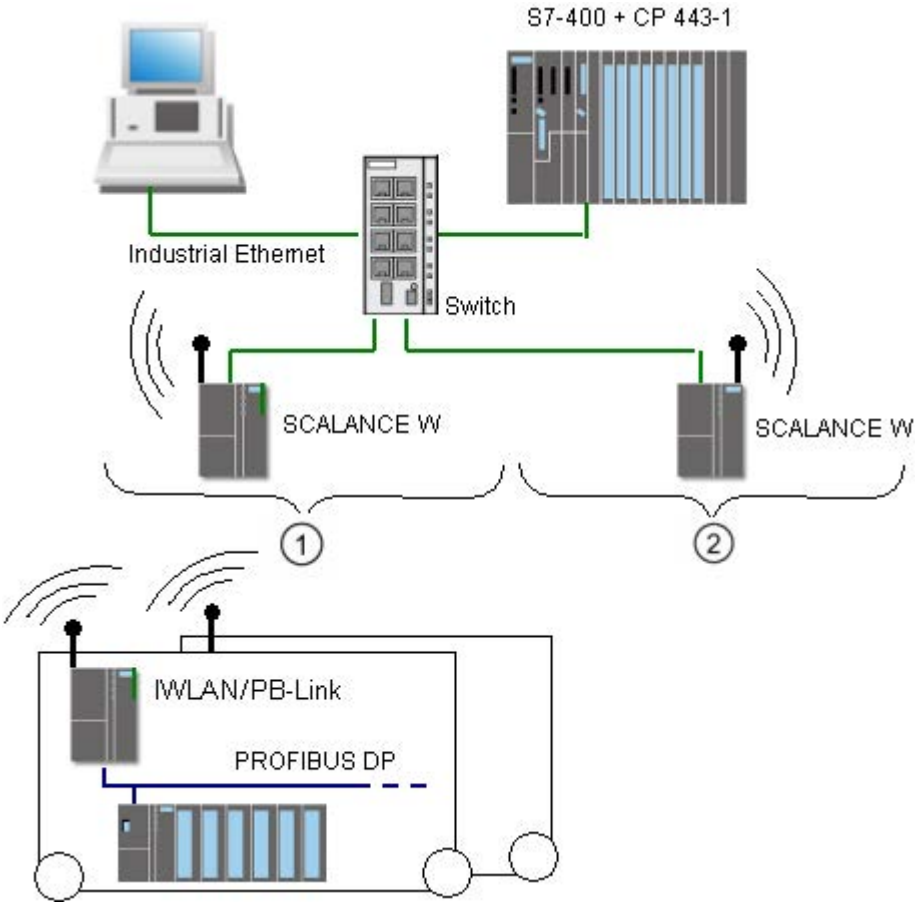
Special feature

If several IWLAN/PB links are located within a segment, they have to share the bandwidth that is available for wireless transmission. This leads to a lengthening of the update time for these devices.

Example

In the following example there are two IO devices (IWLAN/PB link) with a segment.

If no more than a maximum of two IWLAN/PB links are present in a IWLAN segment at the same time, enter a "2".



① Segment 1
② Segment 2

Configure PROFINET IO

Addressing PROFINET devices

Assigning addresses and names to PROFINET devices

In this chapter you will learn which address and naming conventions are valid for the PROFINET devices.

IP addresses

All PROFINET devices work with the TCP/IP protocol and therefore require an IP address for Ethernet operation.

You can set the IP addresses in the module properties. If the network is part of an existing company Ethernet network, ask your network administrator for this data.

The IP addresses of the IO devices are assigned automatically, usually at CPU startup. The IP addresses of the IO devices always have the same subnet mask as the IO controller and are assigned in ascending order, starting at the IP address of the IO controller.

Device names

Before an IO device can be addressed by an IO controller, it must have a device name. This procedure was chosen for PROFINET because names are easier to administer than complex IP addresses.

Both the IO controller as well as IO devices have a device name. When the "Generate PROFINET device name automatically" option is activated, the device name is automatically derived from the name configured for the device (CPU, CP or IM):

- The PROFINET device name is made up of the name of the device (for example, the CPU), the name of the interface (only with multiple PROFINET interfaces) and optionally the name of the IO system:
  <CPU name>.<Name of the interface>.<IO system name>
  You cannot change this name directly. You change the PROFINET device name indirectly, by changing the name of the affected CPU, CP or IM in the general properties of the module. This PROFINET device name is also displayed, for example, in the list of accessible devices. If you want to set the PROFINET device name independently of the module name, you have to deactivate the "Generate PROFINET device name automatically" option.
- A "converted name" is generated from the PROFINET device name. This is the device name that is actually loaded into the device.
  The PROFINET device name is only converted if it does not comply to the rules of IEC 61158-6-10. You cannot change this name directly either.

Rules for the converted name

The rules for the converted name are listed in the following section. If the converted name is **not** different from the name of the module, the name of the module must comply with this rule.

- The name consists of one or more labels , which are separated by a dot [.].
- Restricted to a total of 240 characters (lower case letters, numbers, dash, or dot)
- A name component within the device name, which means a character string between two dots, must not exceed 63 characters.
- A name component consists of the characters [a-z, 0-9].
- The device name must not begin or end with the "-" character.
- The device name must not begin with a number.
- The device name form n.n.n.n (n = 0, ... 999) is not permitted.
- The device name must not begin with the string "port-xyz" or "port-xyz-abcde" (a, b, c, d, e, x, y, z = 0, ... 9).

Example of device names

```
device-1.machine-1.plant-1.vendor
```

If you assign this name to a CPU, for example, STEP 7 will not convert it since it conforms to the rules described above.

Device number

In addition to the device name, a device number is also automatically assigned when an IO device is plugged in. You can change this number.

Devices in the PROFINET subnet

In a PROFINET subnet the maximum allowable number of devices is monitored during configuration.

See also

Assigning the device name and IP address (Page 1442)

Retentivity of IP address parameters and device names (Page 1450)

Assigning the device name and IP address

Assigning an IP address and subnet mask for an IO controller the first time

There are various options for this.

During parameter assignment of the PROFINET interface you must specify if the IP address is set in the project (which means in the hardware configuration) or if the IP address is to be set on the device.

Assignment of an IP address	Comments
Option "IP address is set in the project": The IO controller receives the IP address by loading the hardware configuration, for example, via one of the PROFINET interfaces, by means of the PROFIBUS interface or via the MPI interface.	When the hardware configuration is loaded to the IO controller (e.g. CPU), the IP address and the device name, if set, are also loaded. Example PROFINET interface: 1. Connect your programming device/PC to the same network as the relevant PROFINET device. The interface of the PG/PC must be set to TCP/IP (Auto) mode. 2. Have a list of accessible devices displayed. 3. Select the target device by using its MAC address and load the hardware configuration including the configured IP address (IP address is then saved retentively). If your PROFINET device has an MPI or PROFIBUS DP interface, connect your programming device/PC directly to the PROFINET device via the MPI or PROFIBUS DP interface. The configured IP address is applied when the hardware configuration is loaded.
Option "IP address is set directly at the device": • Assign online • Assignment by user program (instruction IP_CONFIG for S7-300/400, T_CONFIG for S7-1200/1500) • Assign via CPU display (S7-1500) • Higher-level IO controller makes assignment (only with I-devices)	If you have selected this option in the properties of the PROFINET interface, the IP address can be assigned by the online and diagnostics editor, by the primary setup tool, or by the user program ("IP_CONFIG" instruction). This option is set automatically if the option "Multiple use IO system" has been enabled in the properties of the PROFINET IO system (standard machine project). In case of an S7-1200-CPU, make sure that access to the CPU is not protected by a password. If the CPU is write-protected, no IP address and no device name can be assigned directly on the device.

Commissioning a PROFINET interface

Further details of how to commission a PROFINET interface can also be found in the operating instructions for the PROFINET devices of the SIMATIC family.

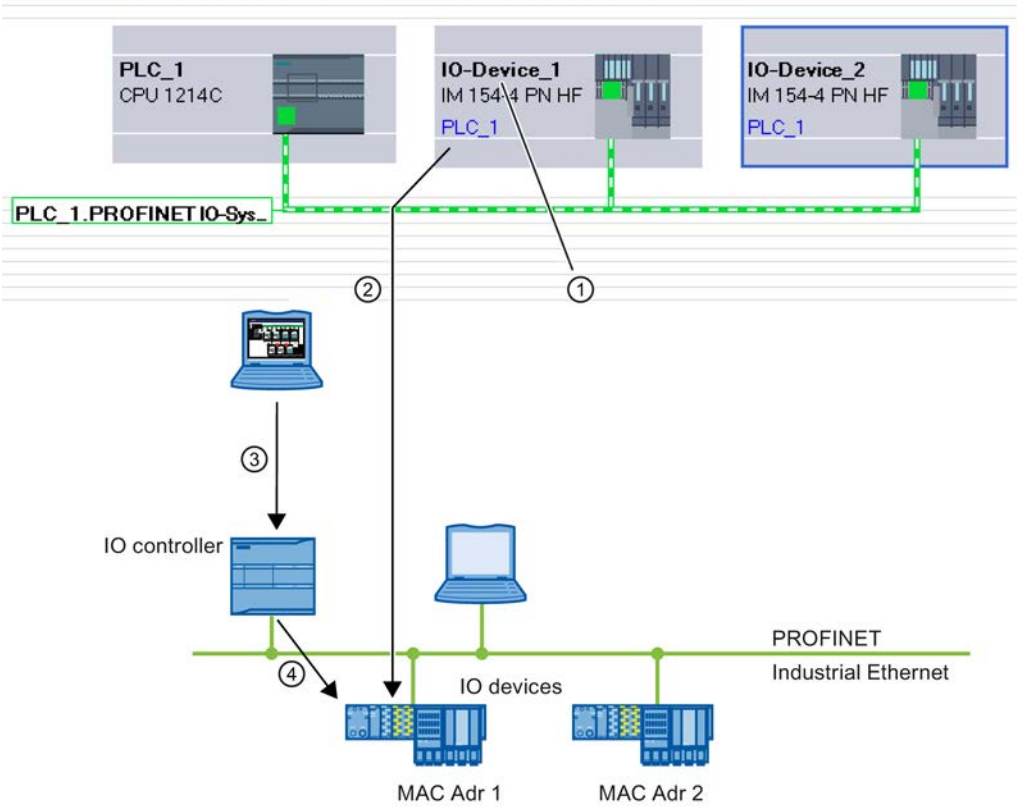
Assignment of device name for IO devices when the "Support device replacement without exchangeable medium" option is selected

For IO controllers with the "Support device replacement without exchangeable medium" option selected, you do not have to assign device names to the IO devices locally, for example in the event of device replacement. Another application is automatic commissioning, in which the CPU automatically assigns the device name and IP address parameters to the IO devices during startup.

Requirement: The ports of the devices are interconnected, and the devices involved support LLDP. The devices have been put into delivery state or - for S7-1500 CPUs version V1.5 and higher - the "Permit overwriting device names of all assigned IO devices" option is selected for the IO controller ("Ethernet addresses" area, "PROFINET" section of the properties of the PROFINET interface).

Assignment of device name and address for an IO device

The following graphic illustrates the process for assigning the device name and address. This process does not apply when the "Support device replacement without exchangeable medium" option is selected.



① Each device receives a name; STEP 7 automatically assigns an IP address.

② From the name, STEP 7 generates a PROFINET device name that you assign to an IO device online (MAC address) and that is written to the device.

③ You load the configuration to the IO controller.

④ The IO controller assigns the appropriate IP address to the IO device with the assigned PROFINET device name during startup.

Changing the device name and IP address

You can change the name and IP address manually. The device name must first be changed in the configuration in order to subsequently assign it to the IO device via the memory card or online with programming device/PC.

Offline with memory card:

1. Place the configured data (device name: for example, turbo-3) for the IO device on the MMC in the PG/PC. Use the command "SIMATIC Card Reader > Save Device Name to Memory Card" in the "Project" menu for this.
2. Then insert the MMC into the IO device. The IO device automatically adopts the configured device name.

Online with programming device/PC:

1. Connect the programming device/PC directly to the Ethernet subnetwork via the PROFINET interface.
2. Select the subnet in the network view and then select the "Assign device name" shortcut command.
3. In the "Assign PROFINET device name" dialog box, select the suitable PG/PC interface to connect to the Ethernet subnet. All configured PROFINET device names are in the top drop-down list. Select a PROFINET device name from it and select the IO device to receive this device name from the table at the bottom. You can filter the display of devices in the table according to various criteria.
4. You can easily identify the device using the "Flash LED" button.
5. Click "Assign name".

The IO controller recognizes the IO device by its device name and automatically assigns the configured IP address to it.

IP address assignment for special IO devices

Special IO devices, for example, SCALANCE X, S7-300 CPs, support the option of assigning the IP addresses not from the IO controller during startup. In this case, the IP address is assigned in a different way. The option is called "IP address is set directly at the device". For additional information, refer to the manual of the respective PROFINET device of the SIMATIC device family.

Another special case is the option "IP address is set by the IO controller during runtime" in the "IP protocol" area of the Ethernet address properties of an IO device. This option is set automatically when the option "Multiple use IO system" is selected for a standard machine project in the associated PROFINET IO system. In this case, an adapted IP address is not assigned by the IO controller until the IO controller itself has received a local IP address.

Requirement for additional procedures when assigning IP address and device name

If the IO device, as described above, should not obtain the IP address or device name from the IO controller, proceed as follows:

1. Select device or network view.
2. Open the properties of the respective PROFINET device and select the area "PROFINET interface [X1]" > "Ethernet addresses".
3. Select the option "IP address is set directly at the device" under "IP parameters" or the option "Permit setting of PROFINET device name directly on the device" under "PROFINET".

Rules

If the "IP address/device name is set directly at the device" option is used for a PROFINET device, note the following:

- The subnet part of the IP address of the IO device must match the subnet part of the IP address of the IO controller.
- The corresponding PROFINET device cannot be used as a gateway.

See also

Assigning a name in the online and diagnostics view opened via "Accessible devices" (Page 2117)

Enabling device replacement without exchangeable medium (Page 1461)

Example of the assignment of the device name

In this example you assign device names to a PROFINET IO controller and a PROFINET IO device. To make assignment easier, the device names should also contain the names of the PROFINET IO system.

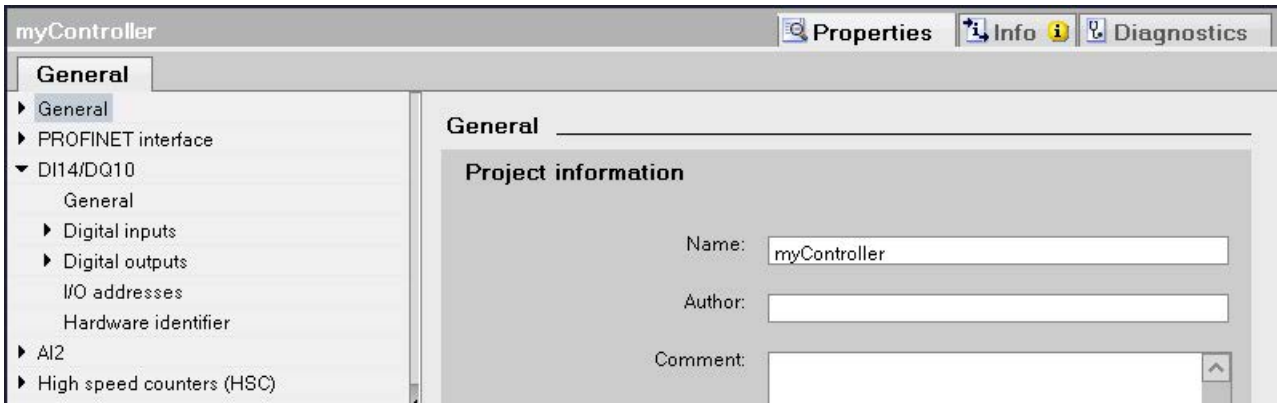
Requirement

- You must be in the network view.
- A CPU 1214C (V2.0 or higher) must be available in the network view.
- An interface module IM 151-3PN exists.
- The PROFINET interfaces of both modules are networked.

Procedure

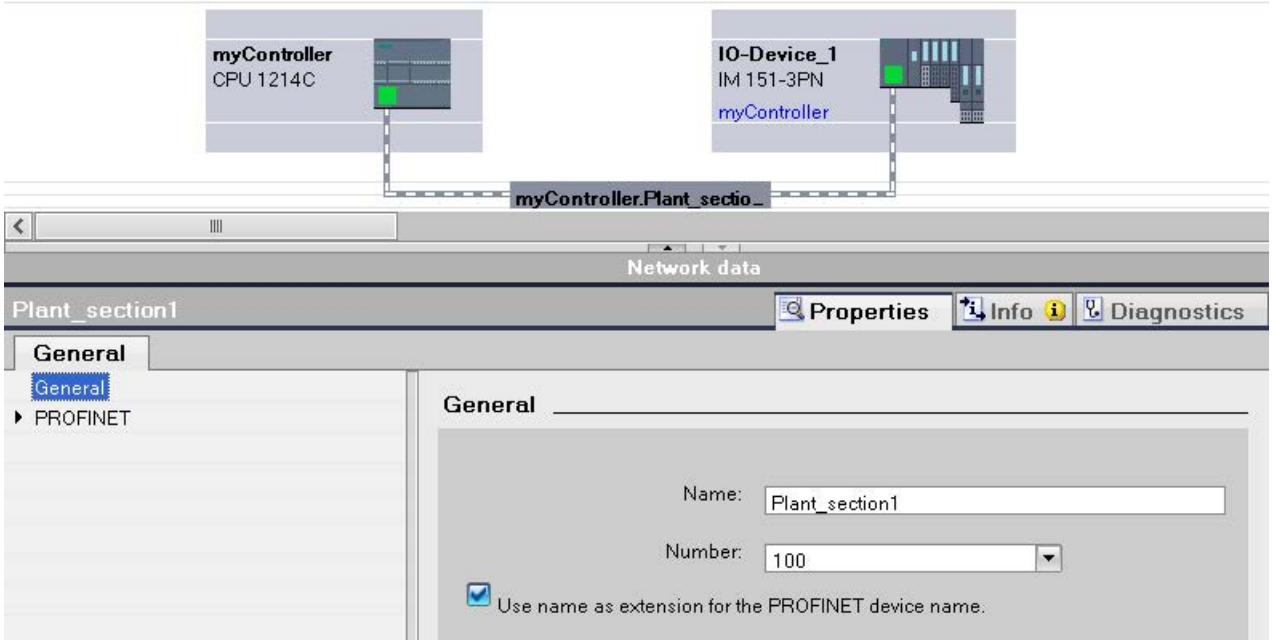
To assign the names, follow these steps:

1. Select the CPU.
   Make sure that you have selected only the CPU and not the complete device!
2. Assign the name "myController" in the Inspector window, under "General".

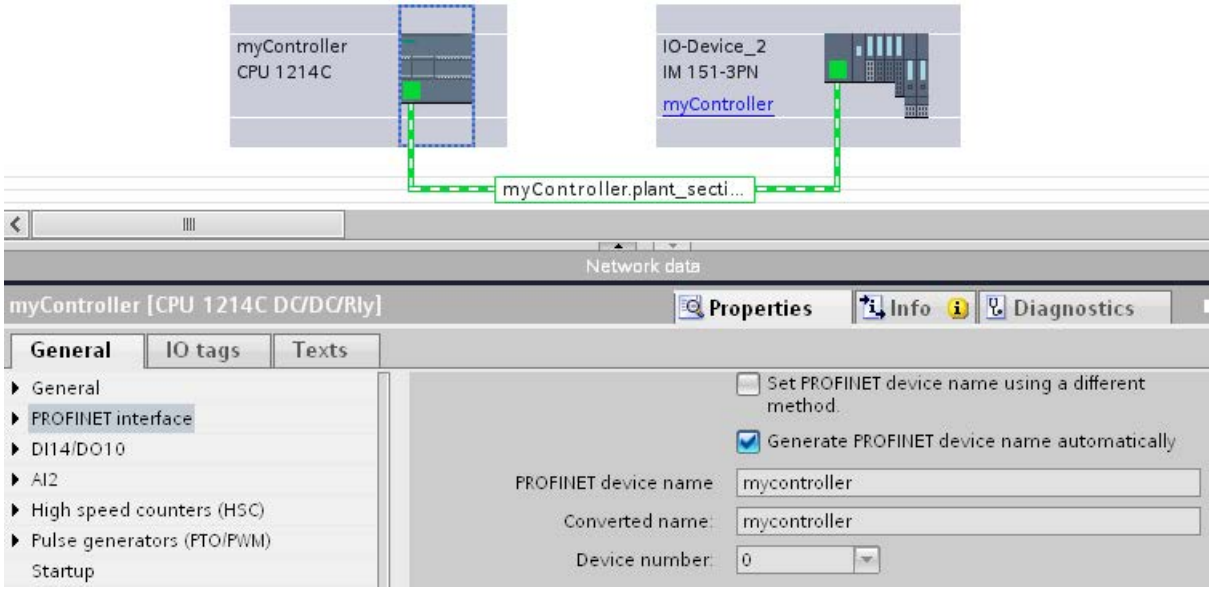


3. Select the interface module.
   Ensure that you have selected only the interface module and not the complete ET 200S device.
4. Assign the name "Device_1" in the Inspector window, under "General".
5. Right-click on the PROFINET IO system and select the "Properties" command.

6. Assign the name "Plant_section1" to the IO system and select the check box "Use name as extension for PROFINET device names".



7. You can find the automatically generated PROFINET device names at the selected device in the Inspector window, at "PROFINET interface".



The PROFINET device name corresponds to the name of the module (with the name of the IO system as extension) with the difference that only lower case text is used. Background: No distinction is made between upper and lower case ("case insensitive") for the storing of the name.
If you want to specify the device name independently of the module name, you have to deactivate the "Generate PROFINET device name automatically" option. The PROFINET device name can be edited in this case.

The converted name is displayed below. This is the name that is automatically generated from the PROFINET device name and satisfies the DNS conventions. If you work with STEP 7, you do not require this name. This name is displayed here as a check and corresponds to the name that is stored in the device. If you work with other tools that are able to record the data exchange and read the actual device names, then you find the converted names.

Other special features

For PROFINET devices with multiple PROFINET interfaces, the name of the interface is attached to the name of the module, separated by a dot.

Example:

- Name of the module: myController
- Name of the interface: Interface_1
- PROFINET device name: mycontroller.interface_1

Assign device name via memory card

Introduction

You can configure the device names of PROFINET IO devices offline. To do this, store a configured device name on a memory card and then insert the card into the appropriate IO device.

If an IO device has to be completely replaced due to a device defect, the IO controller automatically reconfigures the new device. Using the memory card, a device can be replaced without a programming device.

Requirements

- The programming device has a card reader for memory cards.
- The IO device must support the assignment of the device name via memory card.
- The station and its PROFINET IO system is configured.

Procedure

To store a device name on a memory card, follow these steps:

1. Insert the memory card into the card reader.
2. Select the IO device whose device name is to be assigned by the memory card.
3. Select the "Card reader > Save Device Name to Memory Card" command in the "Project" menu.
   If the memory card is not empty, a message will be issued informing you of this and you will have the option to delete the card.

Retentivity of IP address parameters and device names

The retentivity of IP address parameters (IP address, subnet mask, router setting) and device name depends on how the address is assigned.

The non-retentive, temporary assignment means:

- IP address parameters and device name remain valid for the following time period:
  - Until the next POWER OFF
  - Until the next memory reset
  - Until termination of the online connection (for example, after loading the program) After POWER OFF / POWER ON or a memory reset, the CPU can only be accessed via the MAC address.

If the IP address parameters are not retentive, communication based on the IP protocol can no longer take place after the above described events (for example, after POWER OFF/ POWER ON).

The assignment of a temporary IP address also deletes retentively saved IP address parameters.

Assigning IP address parameters and device name non-retentively

IP address parameters and device name are not retentive in the following cases:

- A temporary IP address that is not retentive is implicitly assigned with the "Accessible devices" function, if the device (e.g. CPU) does not yet have an IP address.
- The device is a "normal" IO controller (i.e., not an I-device), and it is specified in the user program ("IP_Conf" instruction) that the IP address parameters/device name are not to be retentive.

Assigning IP address parameters and device name retentively

IP address parameters and device name are retentive in the following cases:

- In the properties of the PROFINET interface, it is specified that the IP address parameters are set in the project ("Set IP address in the project" option).
- In the properties of the PROFINET interface, it is specified that the IP address is to be set on the device.
  - Once the configuration is loaded, the IP address parameters and the device name are assigned via STEP 7 or a setup tool such as PST (STEP 7: online and diagnostic function "Assign IP address"). The assigned IP address parameters are retentive.
  - The device is a "normal" IO controller (i.e., not an I-device), and it is specified in the user program ("IP_Conf" instruction) that the IP address parameters/device name are to be retentive.

Special features of the I-device

In the properties of the PROFINET interface of the I-device, it is specified that the IP address parameters are to be set on the device. The IP address parameters for the I-device are assigned by the higher-level IO controller.

- If prioritized startup is set, the IP address parameters are retentive.
- If no prioritized startup is set, the IP address parameters are not retentive.

Recommendation

If possible, use the "Set IP address in project" option and specify an appropriate IP address. In this case, the IP address is assigned retentively.

Resetting of retentive IP address parameters and device names

Retentive IP address parameters and device names are reset via the online and diagnostic function "Reset to factory settings".

Note

Consequences of reassignment of IP address parameters on top of existing IP parameters

- The temporary assignment of IP address parameters/device names results in a reset of any retentively saved IP address parameters/device names.
- With a fixed assignment of IP address parameters/device names, previously retentively-saved parameters are replaced by the newly assigned parameters.

Note

Reuse of devices

Execute the "Reset to factory settings" before you install a device with retentive IP address parameters and device name in another subnet or system or before you place it in storage.

Creating a PROFINET IO system

A PROFINET IO system is comprised of a PROFINET IO controller and its assigned PROFINET IO devices.

To create a PROFINET IO system you require an IO controller (for example, CPU 1214C) and one or more IO devices (for example, a head module from the distributed I/O family ET 200S).

As soon as you connect an IO controller to an IO device, a controller-device link is established.

Procedure

To create a PROFINET IO system, proceed as follows:

1. Use drag-and-drop to pull an IO controller from the hardware catalog (for example, CPU 1214C) into the free area of the network view.
   The IO controller is created in the project.
2. Use drag-and-drop to move an IO device from the hardware catalog (for example, ET 200S) into the free area of the network view.
3. Click on the PROFINET interface of the IO controller or the IO device.
4. Hold down the mouse button and draw a connecting line between this selected interface and that of the partner device.
   A subnet with an IO system between the IO controller and the IO device is created.
5. If required, adapt the properties of the Ethernet subnet or the IO controller (for example, IP address) under "Properties" in the inspector window.

Handling PROFINET IO systems

Using shortcut menu commands, you can delete PROFINET IO systems, create new ones or even connect the interface to another subnet from within the network view.

An existing PROFINET configuration can thereby be corrected in the network view.

Create new PROFINET IO system for IO controller

To create a new PROFINET IO for an IO controller, proceed as follows:

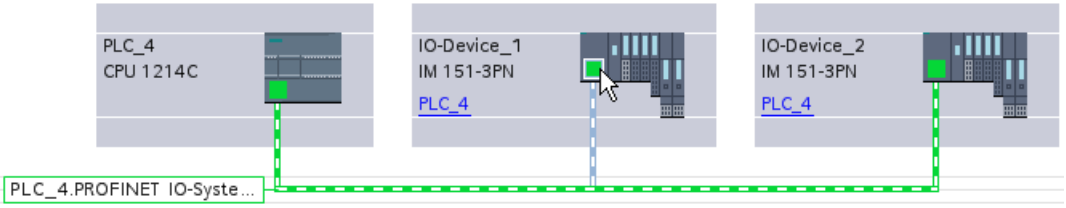
1. Make sure that no IO system is already assigned to the IO controller. If an IO system is already assigned to the IO controller, the "Assign IO system" shortcut menu command is disabled.
2. Select the PROFINET interface and then select the "Assign IO system" shortcut menu command.

A new PROFINET IO system is created at the IO controller and you can assign IO devices to this IO system.

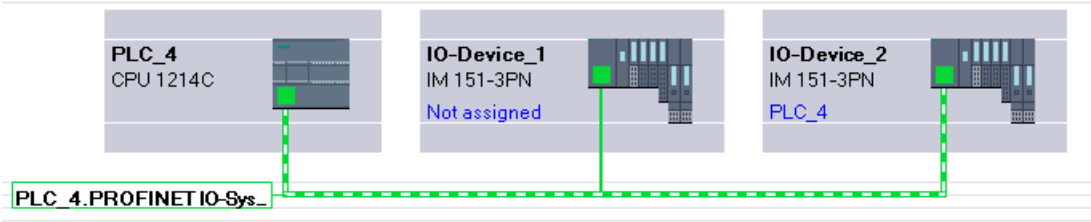
Disconnecting PROFINET IO devices from PROFINET IO system

To disconnect an already networked PROFINET IO device from its PROFINET IO system, follow these steps:

1. Click on the PROFINET interface of an IO device.



2. Select the "Disconnect from IO system" shortcut menu command.
   The IO device that was assigned to this IO system is then no longer assigned to it.



You can create a new IO systems and can assign each of the non-assigned IO devices to an IO controller.

Assign PROFINET IO devices to other IO controllers

Existing PROFINET IO systems can be easily reconfigured in the network view:

1. Select the interface of an IO device and then select the shortcut menu. You have the following options here:
   - Assign a new subnet to the IO device or disconnect it from the existing subnet
   - Assign a new IO controller to the IO device
   - Assign a new IO system to the IO device or disconnect it from the existing subnet
2. To assign another IO controller to the IO device, select the "Assign to new IO controller" shortcut menu command.
   If there is no connection, a subnet is automatically created and the IO device is assigned to the IO system of the new IO controller.

Tip: Quick configuration of IO systems

If the IO system has a lot of IO devices, assign all IO devices placed by drag-and-drop operation to an IO controller on one step.

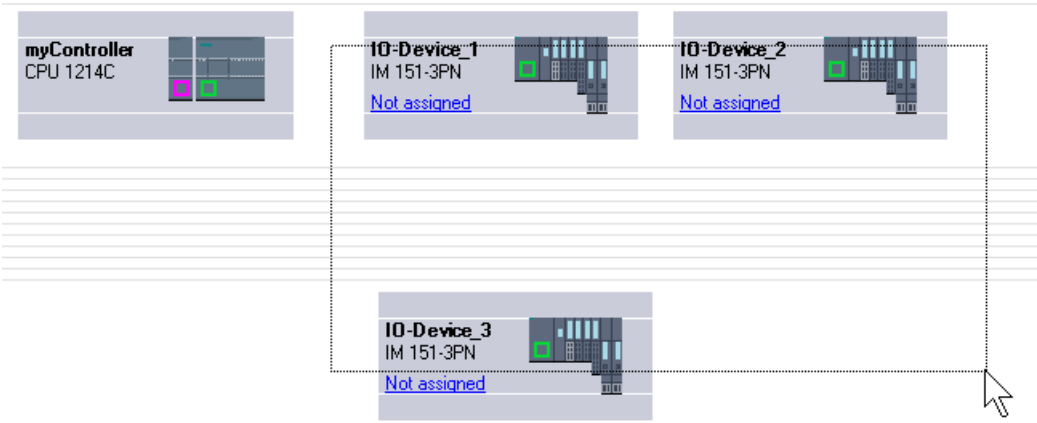
Requirements

IO controller and IO devices are placed in the network view.

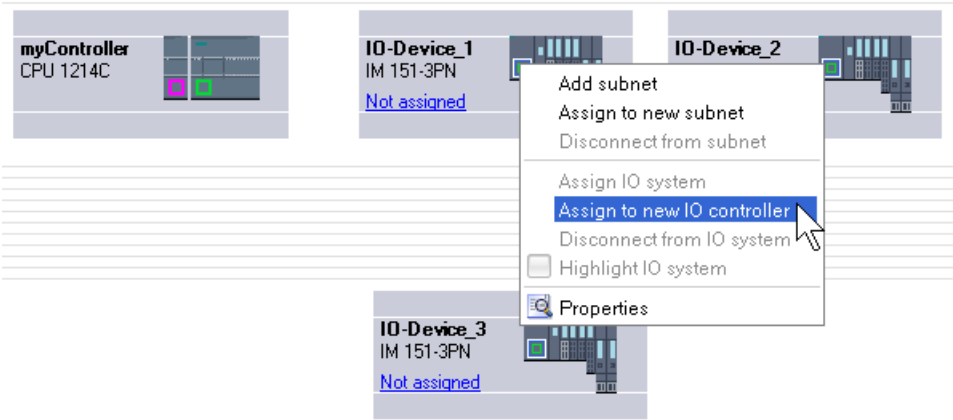
Assign IO devices to an IO system

To do this, follow these steps:

1. Select an appropriate zoom factor so that you can see as many IO devices as possible in the network view.
2. Arrange the IO devices in not more than of two rows.
3. Select all IO interfaces (not all devices) with the mouse cursor. This only works if you begin to drag the mouse cursor outside of the first IO device and release the mouse button at the last IO device (selection with the lasso).



4. Select the shortcut menu "Assign new IO controller" and select the corresponding IO interface of the IO controller in the subsequent dialog.



5. The IO devices are automatically networked with the IO controller and combine with it to form an IO system.

Note

When an IO system is highlighted, you can double-click on an IO device in the hardware catalog and thereby quickly add additional IO devices. Result: The IO device is automatically added to the highlighted IO system.

Interconnecting ports

If an IO device is assigned to an IO controller, this does not yet specify how the ports are connected to each other.

Although a port interconnection is not required to make use of the Ethernet/PROFINET functions, it does offer the following advantages:

- A target topology is specified with the port interconnection. Based on an online-offline comparison, it is possible to conduct a target-actual comparison with all devices that support this function.
- Only with IRT communication: If a port interconnection is configured, STEP 7 can determine the required bandwidth more precisely. As a rule, this leads to a higher performance.

Make sure that no invalid ring structures occur through the interconnection of ports.

Port interconnection is only advisable for devices that support the topology configuration.

Interconnecting ports in the Inspector window

To interconnect ports, follow these steps:

1. Select the Ethernet/PROFINET device or the Ethernet/PROFINET interface.
2. Navigate to the port property "Port interconnection".
   When the Ethernet/PROFINET interface is selected, you can find this setting in the Inspector window as follows: Properties > General > Advanced Options > Port [...] > Port Interconnection.
3. In the "Local port" section, you can find the settings at the local port. In the case of fiber-optic cable you can, for example, set the cable names here.
   In the "Partner port" section, click on the black triangle in the "Partner port" box to display and select the available partner ports.
4. If the port interconnection is a port interconnection with copper as medium and the devices support IRT communication, you can also set cable length and signal transit time.

If the Ethernet/PROFINET interface was not networked, it is automatically networked by this action. In the properties of the subnet you can set whether this subnet should or should not be used for the networking.

See also

Overview (Page 717)

Overview of the CPU properties (Page 1860)

Setting the send clock

Requirements to change the send clock at the PROFINET device

No IRT (Isochronous Realtime) should be configured. In detail, this means:

- No device must be configured at the IO system as a sync slave or sync master.
- All devices at the IO system must be unsynchronized.

If IRT is configured (in other words, if the IO controller is configured as sync master), the send clock can only be configured in the sync domain.

Procedure

To set the send clock on the PROFINET device, follow these steps:

1. Select the PROFINET IO controller in the device or network view.
2. Change the value for the shortest possible update interval in the properties of the PROFINET interface under "PROFINET Interface > Advanced options > Real-time settings > IO communication > Send clock".

The send clock is valid for all PROFINET devices at the IO system. If the synchronization role is set to a value other than "Unsynchronized", you can only set the send clock in the sync domain, in other words, centrally at the PROFINET IO system.

Setting the update time

Update time

An IO device / IO controller in the PROFINET IO system is supplied with new data from the IO controller / IO device within this time period. The update time can be separately configured for each IO device and determines the time interval in which data is transmitted from the IO controller to the IO device (outputs) as well as data from the IO device to the IO controller (inputs).

STEP 7 calculates the update time automatically in the default setting for each IO device of the PROFINET IO system, taking into account the volume of data to be exchanged as well as the set send clock.

Setting the update time

If you do not want to have the update time calculated automatically, you can change the setting.

To change the update time, proceed as follows:

1. Select the PROFINET interface of the IO device in the network or device view.
2. Change the update time in the interface properties under "Advanced options > Realtime settings > IO cycle".
   - To have a suitable update time calculated automatically, select "Automatic".
   - To set the update yourself, select "Can be set" and enter the required update time in ms.
3. To ensure consistency between the send clock and the update time, activate the "Adapt update time when send clock changes" option.
   This option ensures that the update time is not set to less than the send clock.

Non-automatic setting of the send clock may result in errors if the available bandwidth is insufficient or if other limits are exceeded (for example, too many devices are configured).

Setting the watchdog time

Watchdog time

You can configure the watchdog time for PROFINET IO devices.

If the IO device is not supplied with input or output data (IO data) by the IO controller within the watchdog time, it switches to the safe state.

Do not enter the watchdog time directly, but as "Accepted number of update cycles when IO data is missing". This makes setting easier because the update time can be shorter or longer, depending on the power of the IO device or the setting.

The resulting watchdog time is automatically calculated from the "Accepted number of update cycles when IO data is missing".

Configuring the watchdog time

To specify the watchdog time, follow these steps:

1. Select the PROFINET interface of the IO device in the network or device view.
2. In the properties of the interface, navigate to "Advanced options > Realtime settings > IO cycle".
3. Select the required number of cycles from the drop-down list "Trigger watchdog after # cycles with missing IO data".

The watchdog time is subsequently calculated automatically based on the preset factor. It must not be more than 1.92 seconds.

Note

The default setting should only be changed in exceptional cases, for example, during the commissioning phase.

Calculated bandwidth for cyclic IO data

Calculated bandwidth for cyclic IO data

Adherence to the maximum available bandwidth for cyclic IO data is monitored by the system. The maximum bandwidth depends on the send clock cycle. If the send clock cycle is greater than or equal to 1 ms, the maximum bandwidth is 0.5 ms. If the send clock cycle is shorter, the maximum available bandwidth is also reduced.

The bandwidth actually required for cyclic IO data is determined by the system based on the number of configured IO devices and IO modules. Furthermore, the required bandwidth depends on the update time that is used.

In general, the calculated bandwidth increases in the following cases:

- There is a greater number of IO devices
- There is a greater number of IO modules
- The update times are shorter.

Maximum bandwidth for cyclic IO data depending on the send clock

The following table shows how the maximum available bandwidth for cyclic IO data reacts based on the send clock:

Send clock cycle	Maximum bandwidth for cyclic IO data
250 µs – 468.75 µs	<< 125 µs
500 µs – 968.75 µs	= send clock / 2
1 – 4 ms	= 500 µs

Setting port options

Setting the port options

Changing connection settings for the PROFINET IO port

You can change the network settings for the PROFINET IO port as required. By default, the settings are made automatically. In normal situations, this guarantees problem-free communication.

Possible settings for transmission medium/duplex

Depending on the selected device, you can make the following settings for "Transmission medium/duplex":

- Automatic setting
  Recommended default setting of the port. The transmission settings are automatically "negotiated" with the partner port. The "Enable autonegotiation" option is automatically selected by default.
- TP/ITP at x Mbps full duplex (half duplex)
  Setting of the transmission rate and the full duplex/half duplex mode. The effectiveness depends on the "Enable autonegotiation" setting:
  - Autonegotiation enabled
    You can use both cross cable and patch cable.
  - Autonegotiation disabled
    Make sure that you use the correct cable (patch cable or cross cable)! The port is also monitored with this setting.
- Deactivated
  Depending on the module type, the drop down list box can contain the "- Disabled -" option. This option, for example, allows you to prevent access to an unused port for security reasons. With this setting, diagnostic events are not generated.

"Monitor" option

This option is used to activate or deactivate the port diagnostics. Examples of port diagnostics: The link status is monitored, in other words, the diagnostics are generated during link-down and the system reserve is monitored in the case of fiber optic ports.

Option "Enable autonegotiation "

The autonegotiation setting can only be changed if a concrete medium (for example, TP 100 Mbps full duplex) is selected. Whether or not a concrete medium can be set depends on the properties of the module.

If autonegotiation is disabled, this causes the port to be permanently specified, as for example, is necessary for a prioritized startup of the IO device.

You must make sure the partner port has the same settings because, with this option, the operating parameters of the connected network are not detected and the data transmission rate and transmission mode can therefore not be optimally set.

Note

When a local port is interconnected, STEP 7 makes the setting for the partner port if the partner port supports the setting. If the partner port does not support the setting, an error message is generated.

See also

Wiring rules for disabled autonegotiation (Page 1459)

Boundaries at the port (Page 1460)

Wiring rules for disabled autonegotiation

Requirement

You have made the following settings for the port in question, for example, to accelerate the startup time of the IO device:

- Fixed transmission speed
- Autonegotiation incl. autocrossing disabled

This saves you the time required to negotiate the transmission rate during startup.

If you have disabled autonegotiation, you must observe the wiring rules.

Wiring rules for disabled autonegotiation

PROFINET devices have the following two types of ports:

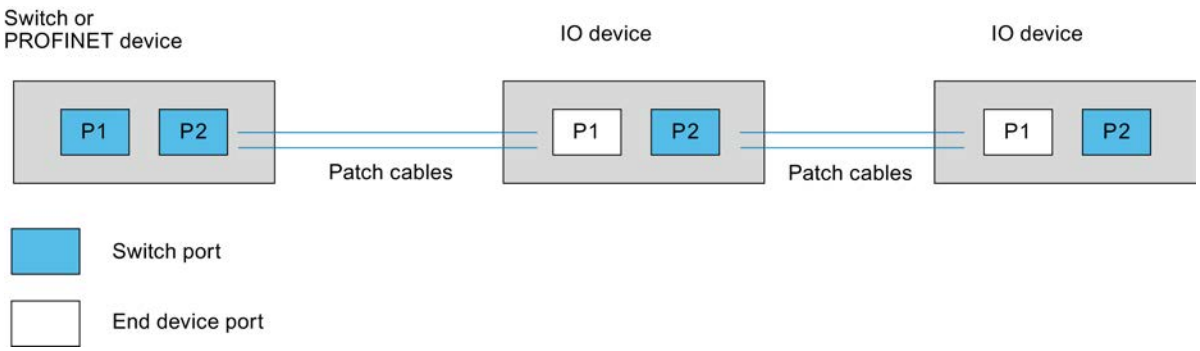
Type of port	PROFINET devices	Note
Switch port with crossed pin assignment	For IO devices: Port 2 For S7 CPUs with 2 ports: Ports 1 and 2	Crossed pin assignment means that the pin assignment for the ports for sending and receiving between the respective PROFINET devices is exchanged internally.
End device port with uncrossed pin assignment	For IO devices: Port 1 For S7 CPUs with one port: Port 1	-

Validity of the wiring rules

The cabling rules described in the following paragraph apply exclusively for the situation in which you have specified a fixed port setting.

Rules for cabling

You can connect several IO devices in a line using a patch cable (one-to-one wiring of both connectors). To do this, you connect port 2 (P2) of the IO device to port 1 (P1) of the next IO device. The following graphic gives an example with two IO devices.



Boundaries at the port

Requirement

To use boundaries, the respective device must have more than one port. If the device for PROFINET does not support boundary settings, the corresponding parameters are disabled.

Enable boundaries

"Boundaries" are limits for transmission of certain Ethernet frames. The following boundaries can be set at a port:

- "End of detection of accessible devices"
  DCP frames for detection of accessible devices are not forwarded. Devices located behind this port are no longer displayed in the project tree under "Accessible devices". Devices located behind this port can no longer be reached by the CPU.
- "End of topology discovery"
  LLDP frames (Link Layer Discovery Protocol) for topology detection are not forwarded.
- "End of sync domain"
  Sync frames transmitted for synchronization of devices within a sync domain are not forwarded.
  For example, if you operate a PROFINET device with more than two ports in a ring, you should prevent the sync frames from being fed into the ring by setting a sync boundary (at the ports not inside the ring).
  Additional example: If you want to use several sync domains, configure a sync domain boundary for the port connected to a PROFINET device of the other sync domain.

Restrictions

The following restrictions must be observed:

- The individual check boxes can only be used if the port supports the function in question.
- If a partner port has been defined for the port, the following check boxes cannot be used:
  - "End of detection of accessible devices"
  - "End of topology discovery"
- If autonegotiation is disabled, none of the check boxes can be used.

Enabling device replacement without exchangeable medium

Replacing an IO device without exchangeable medium

It is often necessary to replace IO devices in automation systems. The IO devices are generally assigned a device name by either inserting an exchangeable medium or via the programming device. The IO controller uses this device name to identify the IO device.

Subject to certain conditions, IO devices can also receive their device names without the insertion of an exchangeable medium (e.g., memory card) or without a programming device. For this purpose, the IO controller uses Ethernet mechanisms (LLDP protocol; Link Layer Discovery Protocol) to analyze the relationships between the individual IO devices and the IO controller. From these relationships, the IO controller detects which IO device was replaced and assigns the configured device name to it.

Requirements

- A port interconnection is already configured.
- The affected IO devices in the automation system must support device replacement without exchangeable medium (LLDP protocol).
  If the individual IO devices in the automation system do not support device replacement without exchangeable medium, a corresponding alarm is output for the IO device.

Note

Use only new IO devices as replacement devices, or restore IO devices with an existing parameter assignment to delivery state prior to commissioning.

For S7-1500 CPUs with firmware version V1.5 or higher, it is not necessary to reset IO devices with an existing parameter assignment to delivery state. The condition for this is that the "Permit overwriting of device name" option is enabled for the IO controller ("Ethernet addresses" area, "PROFINET" section of the properties of the PROFINET interface).

Procedure

In order to enable the replacement of an IO device without exchangeable medium, proceed as follows:

1. In the device or network view, select the PROFINET interface of the corresponding IO controller.
2. In the interface properties under "Advanced settings > Interface options", select the "Support device replacement without exchangeable medium" check box.

The option "Support device replacement without exchangeable medium" also permits automatic commissioning, which means you can commission the IO system with the IO devices without assigning their device names in advance.

See also

Assigning the device name and IP address (Page 1442)

Permit overwriting of PROFINET device name (Page 1547)

Components with the the device replacement without exchangeable medium function (http://support.automation.siemens.com/WW/view/en/36752540)

Using GSD files

GSD files for IO devices

Basic information on GSD files of IO devices

The properties of PROFINET IO devices are not stored in a keyword-based text file (as for PROFIBUS DP slaves), but in an XML file whose structure and rules are determined by a GSDML schema.

The language used to describe the GSD files is GSDML (Generic Station Description Markup Language). It is defined by the GSDML schema.

A GSDML schema contains validation rules that allow it, for example, to check the syntax of a GSD file. GSDML schemas (as schema files) are acquired by IO device manufacturers from PROFIBUS International.

Functional enhancements in the area of PROFINET IO will have an effect on the GSDML specification and the corresponding schema. A new version of the specification and of the schema is created by the functional enhancement.

Names of GSD files for IO devices

One possible example of a GSD file name for IO devices is:

"GSDML-V1.0-Siemens-ET200S-20030616.xml"

Name component	Explanation
GSDML	String at the start of each GSD file for IO devices
V1.0	Version of the GSDML schema
Siemens	Manufacturer
ET200S	Name of the device
20030616	Version code (date)
.xml	File extension

Versioning of GSD files for IO devices

The version information of GSD files is two-fold:

First, the version of the GSDML schema is indicated. This determines the language scope used by a GSD file.

This is followed by the version, listed as an issue date. The version number of GSD files is incremented, for example, after elimination of an error or introduction of a functional enhancement.

Functional enhancements may result in a new version of the GSDML schema. A new version of a GSDML schema might only be supported with restrictions.

Installing the GSD file

Introduction

A GSD file (generic station description file) contains all properties of an IO device. If you want to configure an IO device that is not available in the hardware catalog, you must install the GSD file provided by the manufacturer. IO devices installed via GSD files are displayed in the hardware catalog and can then be selected and configured.

Requirement

- The hardware and network editor is closed.
- You have access to the required GSD files in a directory on the hard disk.

Procedure

To install a GSD file, follow these steps:

1. In the "Options" menu, select the "Install generic station description files" command.
2. In the "Install generic station description files" dialog box, select the folder in which the GSD files are stored.
3. Choose one or more files from the list of displayed GSD files.
4. Click on the "Install" button.
5. To create a log file for the installation, click on the "Save log file" button.
   Any problems during the installation can be tracked down using the log file.

You will find the new IO devices installed by means of GSD files in the hardware catalog under "Additional field devices > PROFINET".

Note

Installation of a GSD file cannot be undone.

Changing the revision of a GSD file

Changing the revision of a GSD file

You can change the revision of a GSD file for an IO device:

- Only for the current IO device
- All suitable IO devices within the IO system
- All suitable IO devices within the complete project

First, all existing GSD files for the current IO device are shown. The only difference between the GSD files shown is their revision status. The currently used GSD file is highlighted.

Requirement

- The I/O data is the same for all IO devices whose revision is to be changed.
- The order number has not changed.
- The number of submodules is identical.
- The configuration data has not changed.
- There must be no module or submodule in a slot that is invalid after the new GSD file has been created.

Procedure

To change the revision of one or more IO devices, proceed as follows:

1. Select the IO device whose GSD file revision is to be changed.
2. Click on the "Change revision" button under "General> Catalog information" in the properties of the IO device.
   The "Change revision" dialog box opens.
3. Select the GSD revision you want to use in the "Available revisions" table.
4. Under "Use selected revision for", select the devices whose version are to be changed:
   – Only for the current IO device
   – For all suitable IO devices in the IO system
   – For all suitable IO devices in the project
5. Click the "Apply" button.

Special PROFINET configurations

Special features for S7-300/400/1500 and PC stations

Address and name assignments for PROFINET devices

In this chapter you will learn which address and naming conventions are valid for the PROFINET devices.

IP addresses

All PROFINET devices work with the TCP/IP protocol and therefore require an IP address for Ethernet operation.

You can set the IP addresses in the module properties. If the network is part of an existing company Ethernet network, ask your network administrator for this data.

The IP addresses of the IO devices are assigned automatically, usually at CPU startup. The IP addresses of the IO devices always have the same subnet mask as the IO controller and are assigned in ascending order, starting at the IP address of the IO controller.

Device names

Before an IO device can be addressed by an IO controller, it must have a device name. This procedure was chosen for PROFINET because names are easier to administer than complex IP addresses.

The device name is automatically derived from the name configured for the device (CPU, CP or IM):

- The PROFINET device name is made up of the name of the device (for example, the CPU), the name of the interface (only with multiple PROFINET interfaces) and optionally the name of the IO system:
  <CPU name>.<Name of the interface>.<IO system name>
  You cannot change this name directly. You change the PROFINET device name indirectly, by changing the name of the affected CPU, CP or IM in the general properties of the module. This PROFINET device name is also displayed, for example, in the list of accessible devices. If you want to set the PROFINET device name independently of the module name, you have to deactivate the "Generate PROFINET device name automatically" option.
- A "converted name" is generated from the PROFINET device name. This is the device name that is actually loaded into the device.
  The PROFINET device name is only converted if it does not comply to the rules of IEC 61158-6-10. You cannot change this name directly either.

Device number

In addition to the device name, a device number is also automatically assigned when an IO device is plugged in. You can change this number.

An IO device can be identified in the user program with this device number, for example, with the instruction "LOG_GEO (Page 6784)". The device name, unlike the device number, is not visible in the user program.

Devices in the PROFINET subnet

In a PROFINET subnet the maximum allowable number of devices is monitored during configuration.

What you should know about the send clock

Send clock - the smallest possible update interval

The send clock shows the time between two successive communication cycles. It is the shortest possible transmission interval for data exchange. The update times are calculated as multiples of the send clock.

Requirements for changing the send clock

For unsynchronized devices, the send clock is configured on the device; for synchronized devices (sync master, sync slave) it is configured in the sync domain.

The send clock cannot be changed on the device if there is a sync slave or sync master device on the IO system. In this case, only the send clock which was selected in the properties of the respective synchronization domain is displayed. To change this you must call the properties of the respective sync domain ("Domain Settings" button).

The send clock can be changed on the device if all devices on the IO system are unsynchronized. You can see the resulting update times for the individual IO devices in the IO communication table or in the properties of the individual IO devices.

Effect of the send clock on the maximum possible bandwidth for cyclical IO data

The following table shows how the send clock affects the maximum available bandwidth for cyclical IO data:

Send clock	Maximum bandwidth for cyclic IO data
250 µs – 468.75 µs	<< 125 µs
500 µs – 968.75 µs	= send clock / 2
1 – 4 ms	= 500 µs

The send clock influences the update time of IO devices for which the "Automatic" and "Can be set" option is selected (with "Adapt update time when send clock changes" option selected).

When the "Can be set" option is selected for IO devices, this can lead to inconsistencies in the configuration. On these IO devices you must subsequently adjust the update time, if necessary.

See also

Setting the send clock (Page 1455)

Setting the send clock

Requirements to change the send clock at the PROFINET device

No IRT (Isochronous Realtime) should be configured. In detail, this means:

- No device must be configured at the IO system as a sync slave or sync master.
- All devices at the IO system must be unsynchronized.

If IRT is configured (in other words, if the IO controller is configured as sync master), the send clock can only be configured in the sync domain.

Procedure

To set the send clock on the PROFINET device, follow these steps:

1. Select the PROFINET IO controller in the device or network view.
2. Change the value for the shortest possible update interval in the properties of the PROFINET interface under "PROFINET Interface > Advanced options > Real-time settings > IO communication > Send clock".

The send clock is valid for all PROFINET devices at the IO system. If the synchronization role is set to a value other than "Unsynchronized", you can only set the send clock in the sync domain, in other words, centrally at the PROFINET IO system.

See also

Configuring PROFINET IO with IRT (Page 1474)

Adjusting odd send clocks

S7-1500 CPUs support both even and odd send clocks.

Even send clocks: 0.25 ms, 0.5 ms, 1 ms, 2 ms, etc.

Odd send clocks: 0.375 ms, 0.75 ms, 1.125 ms, etc.

For IO devices that do not support the fast send clock of the CPU (i.e., of the IO controller) or that do not support odd send clocks, the CPU "adapts" the send clock of the IO device. Thus, for example, it is possible that the CPU (IO controller) and a connected IO device both operate with an 875 µs send clock, while another IO device operates with a 2 ms send clock.

Rules

- The send clock adaptation only works for unsynchronized IO devices (RT class "RT").
- For synchronized IO devices (RT class "IRT"), a uniform send clock that is set in the sync domain applies. The update time corresponds to the send clock in this case.
- For IO devices whose update time option is set to "Automatic," an update time of 2 ms or greater is set by STEP 7, i.e., update times of less than 2 ms are not possible with this option.

Example

A PROFINET IO master system consists of the following components:

- IO controller: CPU 1516-3 PN/DP (6ES7 516-3AN00-0AB0)
- IO device 1: ET 200MP (6ES7 155-5AA00-0AB0) with one DI module
- IO device 2: ET 200S (6ES7 151-3BA23-0AB0 V6.0) with one DI module

IO device 2 does not support the odd send clock (here: 0.875 ms).

1. Assign the IO devices to the IO controller.
2. Set the "Sync slave" synchronization role for IO device 1 (inspector window, PROFINET interface area > Advanced options > Realtime settings > Synchronization).
3. For the IO controller, you set the synchronization role "sync master" in the same way.
4. Interconnect the ports of the IO controller and IO device 1 (e.g., in the topology view).
5. IO device 2 remains unsynchronized; the update option is set to "Automatic".
6. Specify the 0.875 ms send clock in the sync domain (subnet properties). All devices in the sync domain then have this send clock or update time.

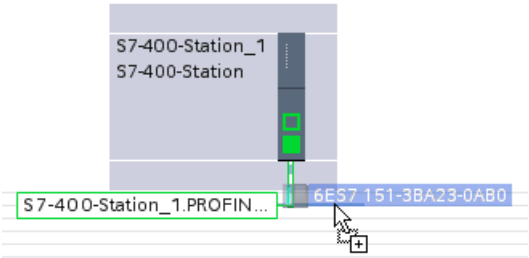
The unsynchronized IO device 2 automatically receives the update time of 2 ms.

Configuration with external IO controller

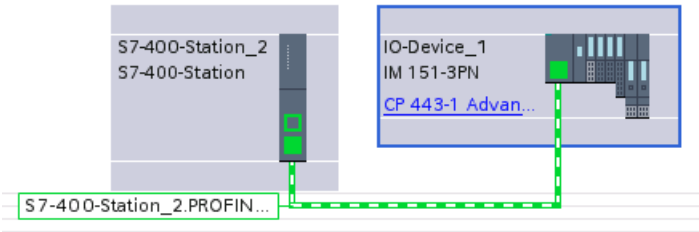
Configuration with external IO controller

CPs which can be used as an external IO controller have a number of communications options and are therefore not designed for use as an IO controller.

If you use an external IO controller (such as CP 443-1 Advanced), you must integrate an IO system after inserting the IO controller.



You can then drag the desired IO devices from the hardware catalog to the PROFINET IO system.



Configuration with SIMATIC PC Station

Configuration with SIMATIC PC Station

A "PC Station" is a PC with communications modules and software components within a SIMATIC automation solution.

By using corresponding communications modules and software components you can operate a PC station as a PROFINET IO controller.

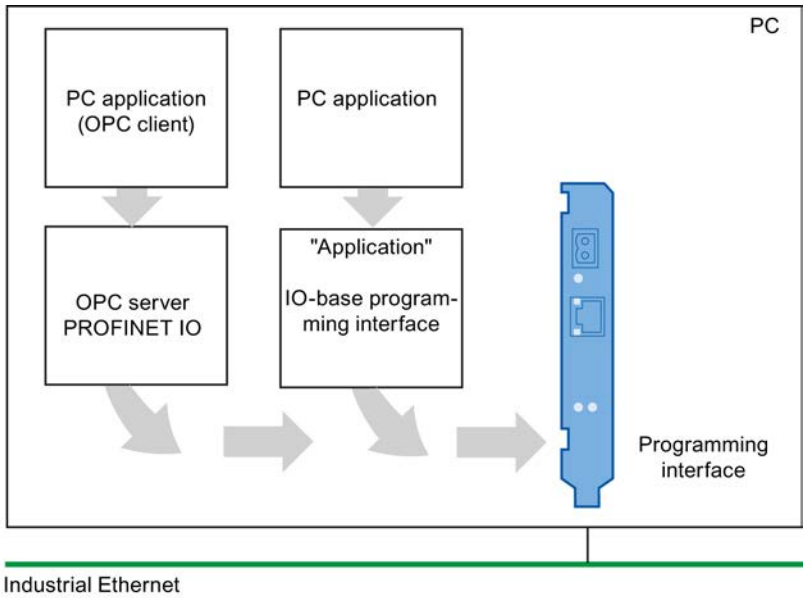
Your PC applications on the PC Station have the following access options to the PROFINET IO controller:

- As OPC client via the OPC server PROFINET IO
- Directly via the PROFINET IO user interface (IO base programming interface)

PC applications can only ever use one of these access possibilities at a time (open/close sequence)

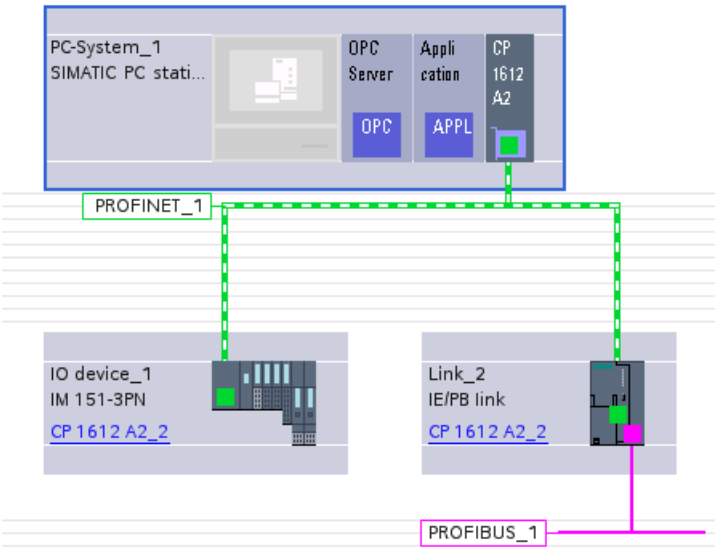
Functions	OPC Server	RTE-base programming interface
Reading and writing IO data	Yes	Yes
Reading and writing data records	Yes	Yes
Receiving and acknowledging interrupts	No	Yes

The following figure shows a PC Station with the components described.



Configuration in the network view

The following figure shows a PC Station with an OPC server and a PC application. The CP 1612 Ethernet interface added converts the PC station into an IO controller with connected IO devices. IE/PB link has also been used as a gateway from PROFINET to PROFIBUS.



IO routing with CP 1616 / CP 1616 onboard / CP 1604

CP as PROFINET IO router - what this means in practice

You can operate a PC station with the CP 1616/CP 1604 as PROFINET IO router by employing the relevant CP as PROFINET IO device and PROFINET IO controller simultaneously. This IO routing allows process data to be exchanged between two PROFINET IO systems. If the configuration is set up correctly the CP can, for example, allow certain process level inputs or outputs to be accessed at control level.

In a typical configuration the CP then simultaneously satisfies the following functions:

- PROFINET IO Device for a higher-level PROFINET IO controller of the control level
- PROFINET IO controller with connected IO devices for the subordinate process level

Assign configuration - transfer area "TM" (transfer module mapping)

In the configuration, select the "IO device" option under "Operating mode" .

The additional "I-device communication" configuration area appears, in which you can make the settings for the IO routing. In the "Transfer area" table there, assign the data areas for data exchange with the higher-level IO controller.

In this case, you have the following options in the "Type" column:

- CD: Controller Device
  Select this option to specify a transfer area between the IO controller and IO device. Refer to additional information on this in the online help under "Configuration for PROFINET IO".
- TM: Transfer module mapping
  Select this option for the IO router application. An additional configuration area with the name specified appears in the Inspector window under "Operating mode / I-device communication". In this configuration area make the assignment for the transfer module mapping in the mapping table.

Transfer module mapping

Transfer modules represent the data routed from the lower-level IO system into the high-level IO system.

Enter the IO devices from the lower-level IO system in the mapping table.

The table contains the following columns:

Parameter	Meaning
TM offset	Offset in relation to the first bit in the transfer module. The specification has the form <Byte>.<Bit>. Example: 0.5
Device/module (slot)	Specification of the IO device The displayed dialog shows the IO devices configured on the IO system.
IO type	Select the IO type from the drop-down list.

Parameter	Meaning
Start	Start address (offset) in the data area of the IO device. The specification has the form <Byte>.<Bit>. Example: 0.0
End	End address (offset) in the data area of the IO device. The specification has the form <Byte>.<Bit>. Example: 8.0

PROFINET device diagnostics

PROFINET device diagnostics

The diagnostic methods available for PROFIBUS DP are the same as those for PROFINET IO. The procedure is identical. The evaluation of diagnostic information is somewhat different with the extended instructions for the distributed I/O in the S7-300/400 CPU user program.

In PROFINET IO the structure for data records and diagnostic information is manufacturer-independent. Diagnostic information is provided only for defective channels.

By using the system status lists (SSLs) and the extended instructions "RALRM (Page 6468)" and "RDREC (Page 6459)" you can make the PROFINET IO system status and the S7 user program diagnostic information available:

- In order to get a complete overview of the PROFINET IO system status, read SSL 0x0X91 with "RDSYSST (Page 6681)".
- To read diagnostic data records directly from a defective module, use the extended instruction "RDREC" (Read data record). You will receive status-related, detailed error information.
- To read event-related diagnostic data records (triggered by error OB), use the extended instruction "RALRM" (read additional alarm info) in the corresponding error OB.

The extended instruction "RALRM" and "RDREC" can also be used for PROFIBUS DP.

You can find information as to which SSLs and which diagnostic data records are defined for PROFINET IO and how the diagnostic data records are structured here (http://support.automation.siemens.com/WW/view/en/19289930).

Configuring IRT communication

Introduction: Isochronous Realtime Ethernet

Introduction: Isochronous Realtime Ethernet

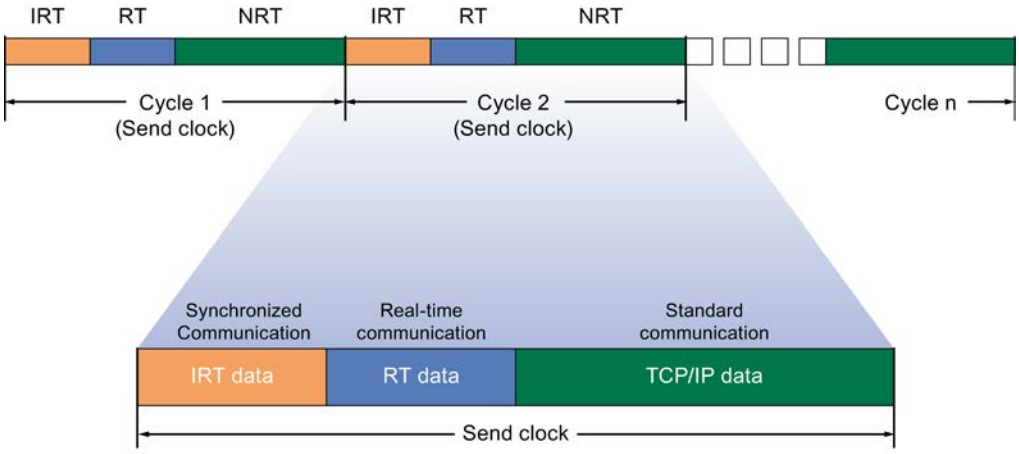
IRT is a transmission method by which PROFINET devices are synchronized with very high accuracy.

A sync master specifies the clock to which sync slaves are synchronized. An IO controller or an IO device can have the role of sync master.

Sync master and sync slaves are always devices in a sync domain. Bandwidth is reserved within the sync domain for IRT communication. Real-time and non-real-time communication (TCP/IP communication) is possible outside of the reserved bandwidth.

Time ranges of communication cycles

The communication cycle is divided into three time ranges, which are represented in the following chart:



- IRT data (synchronized communication)
  This area can be reserved in specific steps, depending on the send clock. Within this time range the IRT data is transmitted.
- RT data (real-time communication)
  In this time range the cyclic RT date is transmitted. RT data has a higher priority than "normal" TCP/IP data. TCP/IP data or Ethernet frames can have a priority between 1 and 7. RT data has a priority of 6.
- TCP/IP data (standard communication)
  Standard communication (TCP/IP, etc.) is transmitted in the remaining interval of the communication cycle.

Applications of IRT

PROFINET with IRT is especially suited for:

- High performance and deterministics for large quantity structures with regard to user data communications (productive data)
- High performance even with many devices in the line topology with regard to user data communications (productive data).
- Parallel transmission of production and TCP/IP data over wire, and securing the forwarding of production data when data volume is high by reserving transmission bandwidth.

See also

Basic procedures for configuring IRT (Page 1474)

Configuring PROFINET IO with IRT (Page 1474)

Sync domain (Page 1476)

Basic procedures for configuring IRT

Basic procedures for configuring IRT

If you want to upgrade equipment with PROFINET IO to IRT, follow these three steps:

1. Configure the PROFINET IO device. The PROFINET device must support IRT.
2. Set which device acts as the sync master and synchronizes the other devices. For this you must configure a sync domain with a sync master and at least one sync slave.
3. Download the configuration to all involved devices.

Creating PROFINET IO configuration for IRT

As a prerequisite for IRT configuration, there must be an existing PROFINET IO configuration, i.e., one or more stations must be configured with an IO controller and IO devices.

You can read which components are suitable for IRT communication in the task card "Information", once you have selected the corresponding components in the hardware catalog.

The requirements for functioning IRT communication are met simply by configuring a PROFINET IO system with components that support IRT.

Configuring PROFINET IO with IRT

Introduction

If you want to increase the precision of cyclic data exchange via PROFINET IO with IRT, you must first configure the PROFINET IO devices. These PROFINET devices must support IRT. Now set which device is to act as sync master and synchronize the other devices. This is done by configuring a sync domain with a sync master and at least one sync slave.

Requirement

- There is an IO system with an IO controller and at least one IO device.
- All devices involved support IRT.

Procedure

Proceed as follows to configure IRT for an existing IO system:

1. Select the PROFINET interface of the **IO controller**.
2. Navigate to "Advanced options > Realtime settings > Synchronization" in the inspector window.
3. Assign the IO controller the role of sync master under "Synchronization role".
4. Now select the PROFINET interfaces of an associated **IO device**.
5. Navigate to "Advanced options > Realtime settings > Synchronization" in the inspector window.
6. Activate the RT class "IRT". The IO device is then automatically assigned the synchronization role "sync slave".
7. You can check and correct your settings at any time using the "Sync domain" button.
8. Interconnect the ports between the sync slaves and sync master.

Or

1. Highlight the PROFINET IO system in the network view.
2. Click on the PROFINET IO system.
3. Navigate to the node of the required sync domain in the inspector window.
4. Enter all necessary settings in the tables:
   - Select an IO system.
   - Set the synchronization role "sync master" for the IO controller.
   - Set the RT class for the IO devices to "IRT". The IO devices is then automatically assigned the synchronization role "sync slave".
5. Interconnect the ports between the sync slaves and sync master.

You can now load the configuration to the relevant devices with PROFINET IRT.

Note

Automatic assignment of a sync master

You can also simply configure an I-device or IO device as sync slave by switching the RT class from RT to IRT. The IO controller that has been assigned to the I-device or IO device is automatically configured as sync master.

If you have configured the IO controller as a sync slave, this setting is not overwritten. The IO controller is thus not switched automatically from sync slave to sync master.

If you assign more than one IO controller to the IO device as a Shared Device, the IO controller automatically selected as the sync master is the one that is assigned to the interface of the IO device.

Sync domain

Sync domain

A sync domain is required for synchronization of PROFINET IO devices. The sync domain assures that all devices belonging to it communicate synchronously.

Prerequisite for IRT communication is a synchronization cycle to distribute a common time base for all PROFINET devices in a sync domain. With this base synchronization, a synchronous transmission cycle of PROFINET devices within a sync domain is achieved. The sync master (usually an IO controller) generates the common synchronization clock and specifies the time basis on which all other sync slaves (usually IO devices) are synchronized.

If the sync master fails, the communication between the IRT devices reverts to RT communication.

Background processes during configuration of an IO system

If you configure an IO controller and network it with an Ethernet subnet, it will automatically be added to the default sync domain of the Ethernet subnet. The default sync domain is always available. The IO controller operates unsynchronized.

If you assign an additional IO device to the IO system of the IO controller, the IO device automatically assigns the IO controller's sync domain.

See also

Working with sync domains (Page 1476)

Working with sync domains

Working with sync domains

All components that take part in IRT communication must belong to a sync domain.

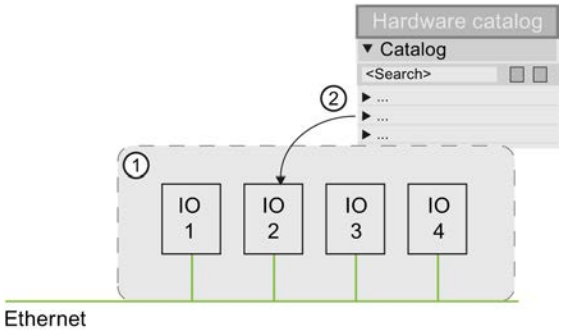
A sync domain is a group of PROFINET devices that are synchronized to a common clock. Exactly one device has the role of sync master (clock); all other devices have the role of sync slaves.

Default sync domain

If you have applied an Ethernet subnet, a special sync domain is automatically applied: the default sync domain. All PROFINET devices configured for the Ethernet subnet automatically belong to this sync domain.

The following figures illustrate the interaction:

① Default domain; always present



①	Default domain; always present
②	Newly configured PROFINET IO systems are first located in the default domain.

Editing properties of sync domains

Editing properties of sync domains

Details of how to define the name and the send clock in a sync domain are given below. In addition, you will learn how to set a sync domain as default domain.

Requirement

- At least one IO device is created in the project.
- At least one sync domain already exists.

Editing a sync domain

To change the name of a sync domain, adjust the send clock, or set up a default domain, follow these steps:

1. Select the Ethernet subnet in the network view in which the sync domain is set up.
2. Open the Ethernet subnet properties.

3. In the local navigation, choose the entry "Domain management > Sync domains > <Name of sync domain> ".
4. Enter the changes.
   - Enter the desired name of the sync domain in the field "Sync domain name".
   - Choose the send clock from the drop-down menu "Send clock".
   - Select the "Default domain" check box if you wish to set the sync domain as the default domain.

Bandwidths for the various RT classes

Available and calculated bandwidths for the various RT classes within a sync domain are displayed in addition to the name and send clock settings.

Select the "Details" section in the area navigation under the sync domain name.

Defining devices of a sync domain

Defining devices of a sync domain

All components that take part in IRT communication must belong to a sync domain. Details on how to define the devices of a sync domain are given below.

Requirement

PROFINET IO systems are already configured and are in the default domain.

Adding a new device to the sync domain

To assign a sync domain to a device, follow these steps:

1. Select the Ethernet subnet in the network view in which the sync domain is set up.
2. Open the Ethernet subnet properties.
3. In the local navigation, select the entry "Domain management > Sync domains > <Name of sync domain> > Devices".
4. Click "Add devices" in the "Station / IO System" table.
   The "Select IO system" dialog opens.
5. Select all devices you want to assign to the sync domain and click "OK".

The selected IO system will be removed from the original sync domain and assigned to the current sync domain.

Removing devices from the sync domain

To remove a device from a sync domain, follow these steps:

1. Select the Ethernet subnet in the network view in which the sync domain is set up.
2. Open the Ethernet subnet properties.
3. In the local navigation, select the entry "Domain management > Sync domains > <Name of sync domain> > Devices".
4. Select the devices in the table "Station / IO-System" which you would like to remove and press <DEL>.

The devices will be deleted and reassigned to the default domain.

Configuring a redundant sync master

You can configure a redundant sync master for PROFINET IO devices in IRT operation.

Example: An S7-1500 CPU can be configured as a sync master, a Scalance 6GK5 201-3BH00-2BA3 V5.0 can be configured as a redundant sync master.

Principle of operation

A redundant sync master belongs to the same sync domain as the "primary" sync master. It prevents a communication failure in the sync domain in the event the primary sync master fails.

Both the sync master and the redundant sync master send sync frames. A sync slave is synchronized to the sync frame of the sync master. If the sync master fails, a sync slave is synchronized automatically to the sync frames of the redundant sync master.

Rules

- The redundant sync master role is assigned to only one device in the sync domain.
- The sync domain contains only IRT devices with "IRT with high performance" option.
- There are no more than 2 devices between the sync master and the redundant sync master.

Procedure

To configure a redundant sync master, follow these steps:

1. In the device view or network view, select the device that you want to configure as a redundant master.
2. Under "PROFINET interface > Advanced options > Real time settings > Synchronization > Synchronization role", select the "Redundant sync master" option.

Regulations and information for IRT configuration

The following rules and information apply to PROFINET configurations. Additional system recommendations for optimizing your PROFINET system can be found here (Page 1437).

Setting up PROFINET with IRT

Keep in mind the following rules for the set up and operation of a PROFINET IO system in IRT operation. They serve to insure an optimal operation of your PROFINET IO system.

- When using IRT, you must configure the topology. This will enable exact calculation of the update time, bandwidth, and additional parameters.
- If you would like to use multiple sync domains, configure a sync domain boundary for the port which is currently connected to the PROFINET device of another sync domain.
- In a sync domain you can only configure one sync master at a time.
- A PROFINET IO system may only be part of a single sync domain.
- If you configure a PROFINET device in a sync domain and want to synchronize with IRT, the PROFINET devices concerned must support IRT communication.
- Wherever possible, use the same PROFINET device as the PROFINET IO controller and sync master.
- If only some of the PROFINET devices in a PROFINET IO system are synchronized, please keep in mind the following: PROFINET devices which are not part of IRT communication are placed outside of the sync domain.

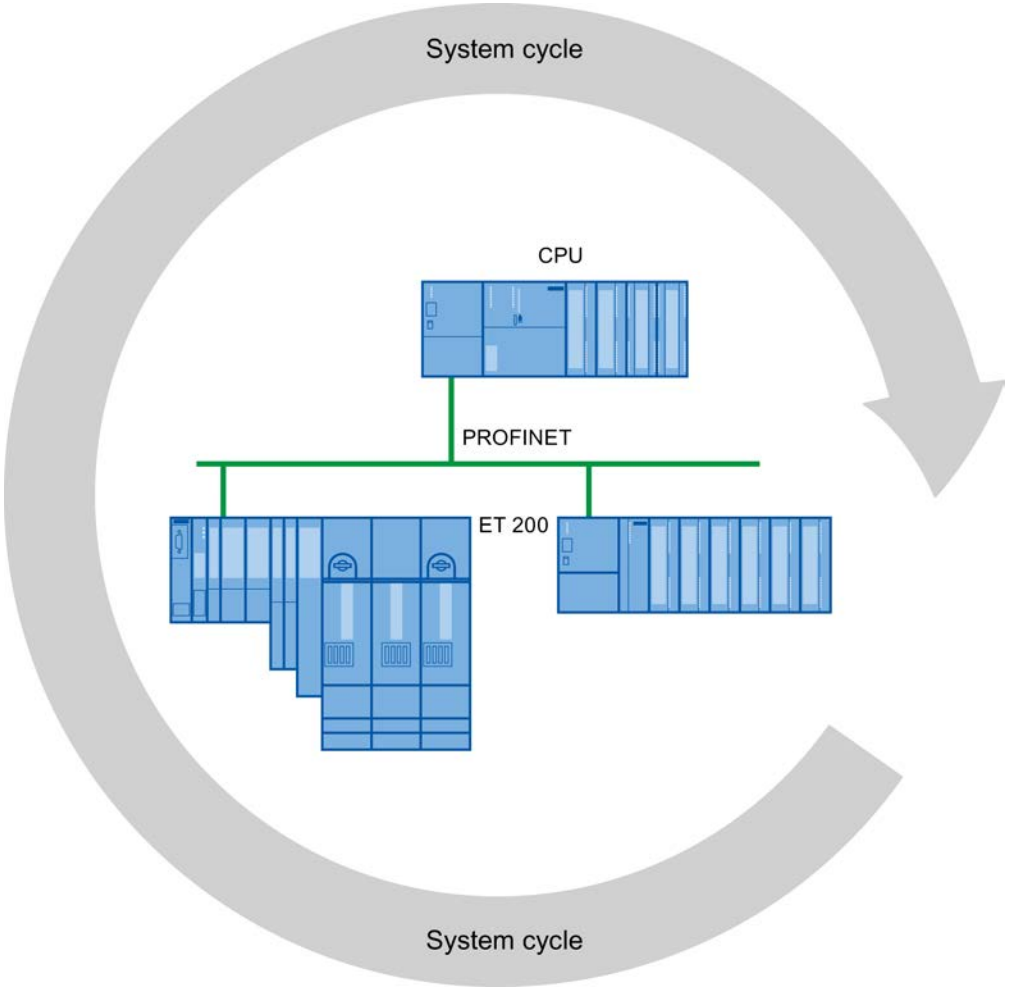
Configuring isochronous mode

What is isochronous mode?

Objectives of isochronous operation

Assuming public transport were to operate at maximum speed while reducing stop times at the passenger terminals to absolute minimum, the last thing many potential passengers would notice of the departing contraption are its red tail lights. The overall travel time is, however, decided by the train, bus or underground clock, because well adjusted timing is essential to a good service. This also applies to distributed automation engineering. Not only fast cycles but also the adaptation and synchronization of the individual cycles result in optimum throughput.

Just-In-Time



The high speed and reliable reaction time of a system operating in isochronous mode is based on the fact that all data are provided just-in-time. The constant PROFINET IO cycle beats the time here.

Use of isochronous mode

Use of isochronous mode

With the "Isochronous mode" system property, measured values and process data can be acquired in a fixed system clock. The signal processing up to the switching-through to the "output terminal" occurs within the same system clock. Isochronous mode thus contributes to high-quality control and hence to greater manufacturing precision. With isochronous mode, the possible fluctuations of process response times are drastically reduced. The time-assured processing can be utilized to improve machine cycle times.

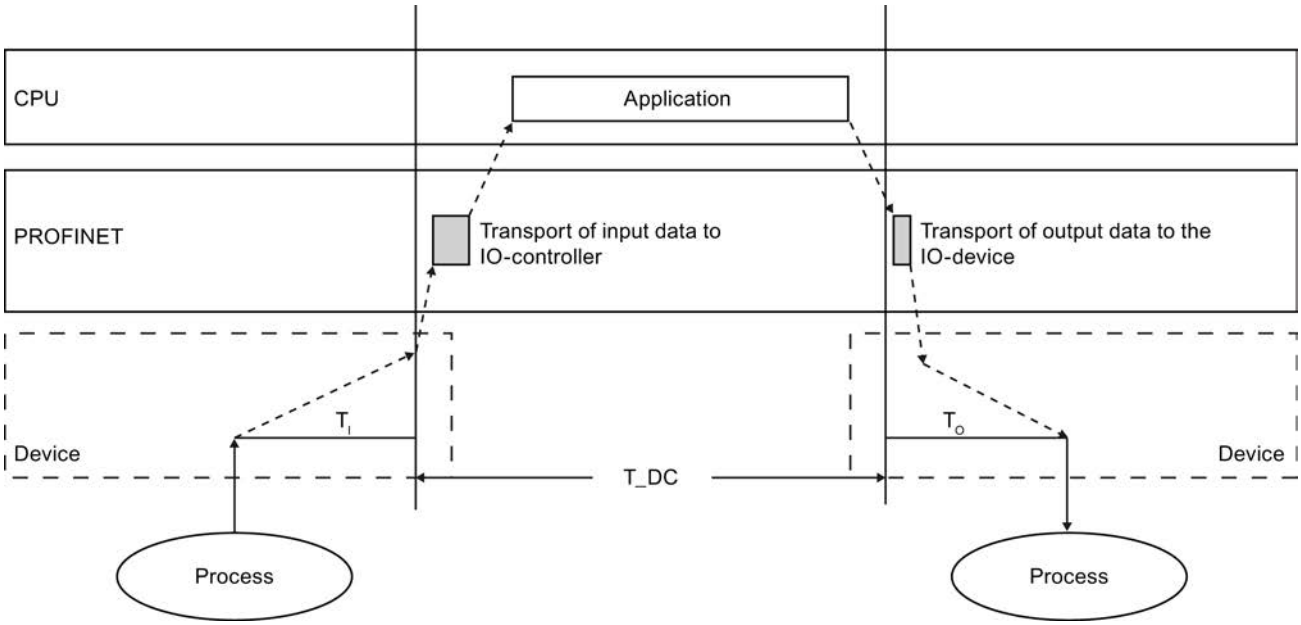
In principle, isochronous mode is worthwhile whenever measured values must be acquired synchronously, movements must be coordinated, and process responses must be defined and simultaneously executed, as in the following example.

Time sequence of synchronization

From reading-in or input data to outputting of output data

The basic time sequence of all components involved in synchronization is explained in the following:

- Reading-in of input data in isochronous mode
- Transport of input data to the IO controller (CPU) via the PROFINET subnet
- Further processing in the isochronous application of the CPU
- Transport of output data to the outputting IO device via the PROFINET subnet
- Outputting of output data in isochronous mode



T_DC	Data cycle (Time_DataCycle)
T_I	Time for reading in the input data
T_O	Time for outputting the output data

To ensure that all input data is ready for transportation via the PROFINET IO line when the next PROFINET IO cycle begins, the IO read cycle has a lead time T_I so that it starts earlier. T_I is the "flashbulb" for the inputs; at this instant, all synchronized inputs are read in. T_I is necessary in order to compensate for analog-to-digital conversion, backplane bus times, and the like. The bias time T_I can be configured by STEP 7 or by the user. We recommend automatically assigning the bias time T_I of STEP 7.

The PROFINET IO line transports the input data to the IO controller. The isochronous mode interrupt OB (OB 61, OB 62, OB 63, or OB 64) is called after a parameterizable delay time. The user program in the synchronous cycle interrupt OB decides the process reaction and provides the output data in time for the start of the next data cycle. The length of the data cycle is always configured by the user.

T_O is the compensation arising from the backplane bus and the digital-to-analog conversion within the device. T_O is the "flashbulb" for the outputs; at this instant, the synchronized outputs

are output. The time T_{IO}can be configured by STEP 7 or by the user. We recommend automatically assigning the bias time T_O of STEP 7.

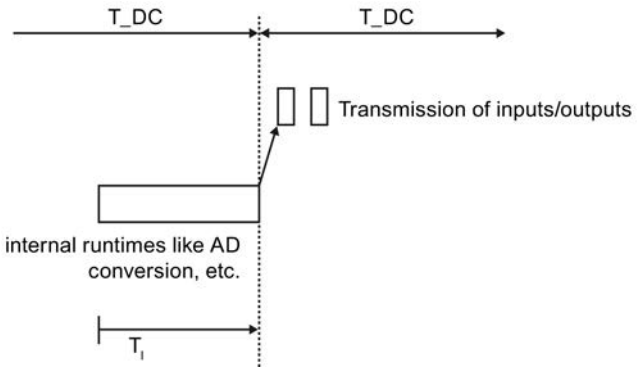
Isochronous mode and non isochronous mode distributed I/O

It is possible to combine isochronous mode distributed I/O with non isochronous mode distributed I/O on one IO controller.

The Ti Value

Effect of T_I

The effect of T_I is illustrated by the following figure:



T_DC	Data cycle
T_I	Time for reading in the input data

Sequence

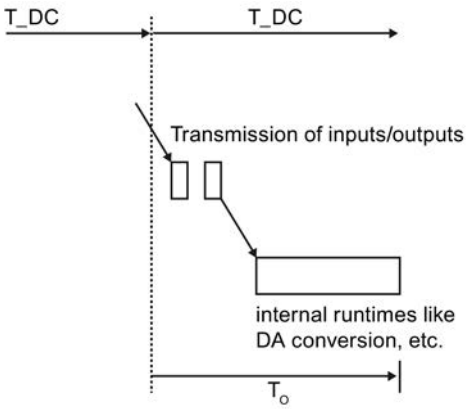
So that a consistent state of the inputs can be transferred to the IO controller at the start of the new system clock cycle, the import must be brought forward by the time T_I. The time T_I for a particular input module includes at least the signal conditioning and conversion time on the electronics modules and the time for transfer to the interface module on the IO device backplane bus.

In the plant, the values are imported at the same time because the T_I of the various input modules is set to the same value and this value is greater than or equal to the highest minimum T_I of all input modules. With the default setting, Step 7 makes sure that the smallest possible mutual T_I is set.

The To Value

Effect of T_O

The effect is illustrated by the following figure:



T_DC	Data cycle
T_O	Time of data output

Sequence

To ensure that a consistent status of the outputs can be transferred to the process at the start of the new system clock cycle, the data is output at the time T_o following the last clock beat. The time T_O records the time of the transfer from the IO-controller to the IO device (through PROFINET IO) for a certain output module and in the IO device, the transfer of the outputs from the interface module to the electronic module (backplane bus)(if applicable, plus time for the digital-analog conversion).

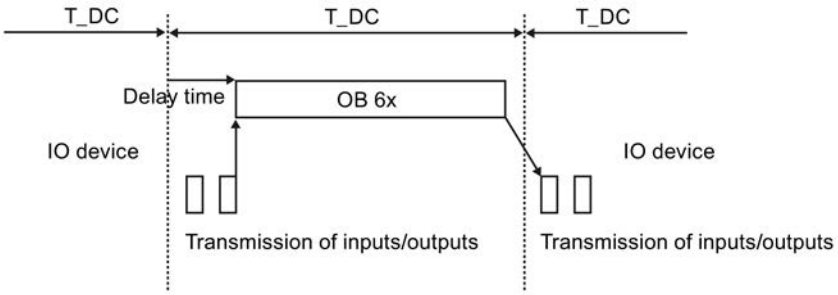
The system makes sure that the values are written simultaneously by setting the T_O of all isochronous output modules to the same value. This value must be greater than or equal to the largest minimal T_O of all isochronous output modules. STEP 7 automatically calculates the shared smallest possible T_O.

Isochronous interrupt OB (OB 6x)

You use the isochronous mode interrupt OB (e.g., OB 61) for the isochronous connection of the user program.

Call of isochronous mode interrupt OB

The isochronous interrupt OB is triggered by the system clock, or more precisely by an adjustable delay time after the system clock. The effect is illustrated by the following figure:



T_DC Data cycle

Delay time

STEP 7 initially automatically calculates a suitable value for the delay time. The delay time compensates the transfer time of the inputs of the isochronous IO devices to the IO controller through the PROFINET IO-network. The execution of OB 6x is linked to the system clock and the specified delay time. The delay time can also be manually corrected, if necessary. For example, you can shorten the delay time in order to process program sections that do not access the isochronous I/O at the start of OB 6x. This gains you runtime for the isochronous user program.

Connection of isochronous I/O to the isochronous mode interrupt OB

You access the isochronous I/O via a process image partition, i.e., the addresses of isochronous modules must be in a process image partition.

Any access to isochronous I/O must be programmed by the SYNC_PI and SYNC_PO instructions within the isochronous mode interrupt OB.

You call SYNC-PI at the start of OB 6x, provided you have selected the automatic setting for the delay time. You call SYNC_PO at the end of OB 6x.

TD_C, application cycle, and send clock

Time TC_C designates the time for one data cycle, i.e., this time is the shortest possible time for reading in, processing, and outputting the data (IPO model). TC_C is always a multiple of the send clock.

If IRT is activated for an IO device, TC_C is then equal to the send clock. The update time corresponds to the send clock.

The application cycle is a multiple of the data cycle. It can be identical to the data cycle if the runtime of the isochronous mode interrupt OB is correspondingly short.

Program processing according to the IPO model with short time

The IPO model with application cycle factor = 1 is characterized by the fact that the I/O data are completely processed within one system clock T_DC. This model is used to achieve the shortest response times.

Program Execution According to the IPO Model with Long Time

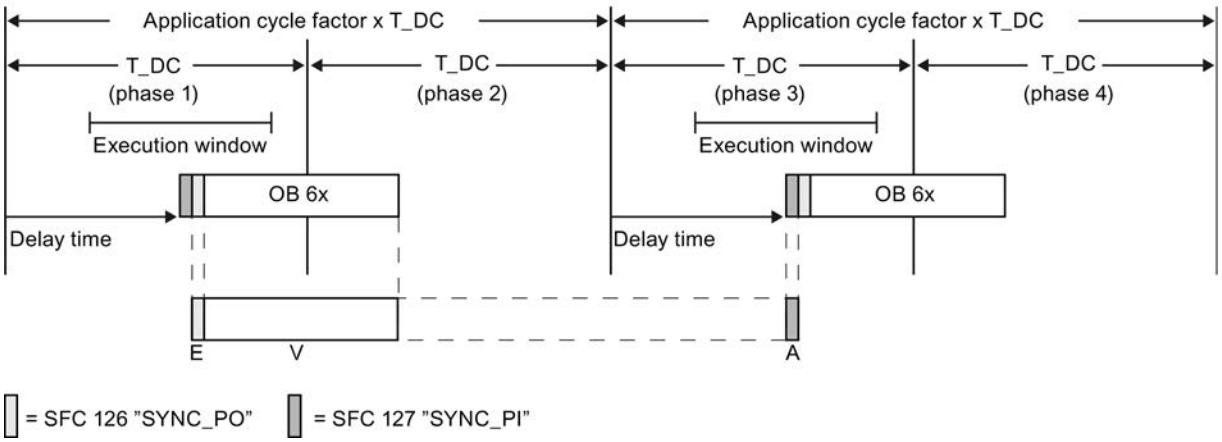
Label of the IPO-model with application cycle factor > 1

If the runtime of the isochronous mode OB is longer than one data cycle T_DC, the output must be delayed by one application cycle (OB 6x). For this reason, data output occurs before data input in this model.

The SYNC_PI and SYNC_PO instructions must then be called during the execution window of the first data cycle (only available there when application cycle factor > 1).

The execution window refers to the time period during which a call of the SYNC_PI and SYNC_PO instructions is possible, i.e., after transfer of cyclic data via PROFINET up to the time shortly before the end of data cycle T_DC.

The other user program processing in OB 6x then occurs in the next data cycle or over the following data cycles.



The figure shows the signal sequence with application cycle factor = 2 (application cycle = 2x data cycle) from the time of acquisition (E) through the processing (V) in the IO controller until output (A) of the process values. STEP 7 calculates the delay time. In this time, the read input data is on its way on PROFINET IO.

This model is especially well-suited for large I/O configurations with one extensive user program in OB 6x. With this model, longer calculation times are possible when processing the input data and obtaining the corresponding output data.

The constant processing time from the "input terminal" to the "output terminal" is T_I + (application cycle factor + 1) x T_DC + T_O. T_I + (2 x application cycle factor + 1) x T_DC + T_O can be guaranteed as a process reaction time.

Configuring isochronous mode

Introduction

The configuring of isochronous mode of a module is described as IO device in the following based on the new ET 200MP distributed I/O series. The procedure described also applies to other distributed I/O systems (e.g., ET 200S or ET 200SP).

The IO controller is an S7-1500 CPU.

Requirement

- The network view is open.
- An S7-1500 CPU is placed (e.g., CPU 1516-3 PN/DP).
- A distributed I/O device is placed (e.g., IM 155-5 PN HF) and networked with the CPU.
- All requirements for an IRT configuration are met, namely:
  - The ports of the CPU and the IM are interconnected.
  - The RT class of the IM is set to "IRT" (area "Advanced options > Real-time settings > Synchronization").

Procedure

To create an isochronous connection between the I/O and user program, follow these steps:

1. Select the distributed I/O device (the IM 155-5 PN HF) in the network view, and switch to the device view.
2. Insert an I/O module with isochronous mode capability (module with name extension "HF", e.g., DI 16x24VDC HF - 6ES7 521-1BH00).
3. Go to "I/O addresses" area in the inspector window of the selected I/O module.
4. Make the following settings in the I/O addresses area:
   - Select the "Isochronous mode" option.
   - Select a process image partition, e.g., process image partition 1.
   - Click on the "Assigned organization block" drop-down list and click the "Add object" button.
     A dialog box for selecting organization blocks opens.
   - Select the "Synchronous Cycle" OB and confirm the selection with "OK".
     In the case of automatic number assignment, OB 61 will be generated and opened. The "Isochronous mode" area is selected in the inspector window, and you can continue directly with the setting of the application cycle and delay time and start the programming of the OB in the instruction section.
5. Add other IO devices as required, network and interconnect these devices, and adapt the configuration and the isochronous mode settings.
6. To retrieve information about calculated bandwidths or to adapt the send clock, select the subnet and navigate to the corresponding area of the sync domain management in the inspector window.

Configuring isochronous mode

Introduction

The configuring of isochronous mode of a module is described as IO device in the following based on the new ET 200MP distributed I/O series. The procedure described also applies to other distributed I/O systems (e.g., ET 200S or ET 200SP).

The IO controller is an S7-300/400 CPU.

Requirement

- The network view is open.
- An S7-300/400 CPU is placed (e.g., CPU 319-3 PN/DP).
- A distributed I/O device is placed (e.g., IM 155-5 PN HF) and networked with the CPU.
- All requirements for an IRT configuration are met, namely:
  - The ports of the CPU and the IM are interconnected.
  - The synchronization role of the PROFINET interface is set to "sync master" (properties of the CPU, area "PROFINET interface> Advanced options > Real-time settings > Synchronization").
  - The RT class of the IM is set to "IRT" (area "PROFINET interface > Advanced options > Real-time settings > Synchronization").

Procedure

To create an isochronous connection between the I/O and user program, follow these steps:

1. Select the CPU, and navigate to the "Interrupts > Isochronous mode interrupts" area in the inspector window.
2. Select the distributed I/O device (the IM 155-5 PN HF) in the network view, and switch to the device view.
3. Insert at least one I/O module with isochronous mode capability (module with name extension "HF", e.g., DI 16x24VDC HF - 6ES7 521-1BH00).
4. Assign a process image partition to the I/O module (inspector window, "I/O addresses" area).

5. Select the IM in the device view or network view and navigate in the inspector window to the "PROFINET interface > Advanced options > Isochronous mode" area.
   - Select the "Isochronous mode" option.
   - If you want to change the send clock or the application cycle, click the green link next to the appropriate fields. This takes you to the sync domain where these settings can be centrally changed.
   - For the Ti/To times, select whether their minimum times are to be automatically calculated or whether you want to set them manually. The Ti/To values can only be changed with manual setting.
   - In the detail overview of modules: Select the "Isochronous mode" option for modules that are to be operated in isochronous mode.
6. Add other IO devices as required, network and interconnect these devices, and adapt the configuration and the isochronous mode settings.
7. To retrieve information about calculated bandwidths or to adapt the send clock, select the subnet and navigate to the corresponding area of the sync domain management in the inspector window.

Setting isochronous mode on the subnet

Introduction

The subnet properties include not only the sync domain properties and MRP properties but also provide an overview of the isochronous mode settings.

Requirement

A PROFINET IO system with IO controller and IO devices is configured.

Procedure

To display the isochronous mode properties of the subnet, follow these steps:

1. Select the subnet.
2. Select the "Overview of isochronous mode" area in the properties.

Settings in the overview of isochronous mode

CPU	Selection of a CPU (IO controller) of the subnet
Isochronous application	Selection of an OB of the selected CPU that is assigned to an IO system.
Details of the isochronous application	

Application cycle	Indicates the application cycle (multiple of data cycle). For S7-300/400, you specify the time in the CPU properties, while for S7-1500, you specify the time in the isochronous mode OB properties.
Delay time	Indicates the time between the system clock and start of the isochronous mode interrupt OB (e.g., OB 61).
Send clock	Indicates the time period between two consecutive intervals for IRT or RT communication.
IO system no.	Indicates the no. of the PROFINET IO system assigned to the OB.
Process image partition	Indicates all process image partition numbers that are assigned to the PROFINET IO system.
Ti/To values for OB	
Ti/To values automatic	If this option is selected, STEP 7 then calculates one Ti value and one To value suitable for all modules that will be operated in isochronous mode and are assigned to the OB. You can reference these common Ti/To values in the IO devices/ Modules table (Ti/To values = from the OB).
Time Ti	Time for reading-in the isochronous input data.
Time To	Time for outputting the isochronous output data.
IO devices/Modules	
Table with Ti/To values:	Isochronous mode can be activated and deactivated for each IO device/module in the IO device. Ti/To values: Three settings are possible: • Automatic: Ti/To will be calculated automatically. • Manual: Ti/To can be edited. • From the OB: Ti/To values will be taken from the OB. Ti, To: Can only be edited when "manual" setting is selected. TiMin/TiMax: Minimum or maximum time for Ti. ToMin/ToMax: Minimum or maximum time for To. Interval: Specified interval for Ti/To values.

Rules, requirements, and supplemental conditions

What you need to be aware of for configuring isochronous mode is

Take into consideration the following notes on configuring isochronous mode.

Components and PROFINET interfaces

- Isochronous mode is only possible with the interfaces integrated in the CPU. For S7-1500 with two PROFINET interfaces, only the X1 interface can be configured for PROFINET IO and isochronous mode. Isochronous operation with CPs is not possible.
- A full isochronous mode from "terminal" to "terminal" is only possible if all components participating in the chain support the "isochronous mode" system property. When selecting in the catalog, make sure you check the "isochronous mode" or "processing in isochronous mode" entry in the information field of the module.
- If you operate an IO device in isochronous mode (i.e., you have assigned it the sync slave role, for example), the IO device must contain at least one module or submodule operated in isochronous mode.

Subnet-overlapping isochronous mode

- Subnet-overlapping isochronous mode is not currently possible.

Process image partitions

- Isochronous I/O can only be processed in process image partitions. Without the use of process image partitions isochronous consistent data transfer is not possible. Compliance with configuration limits is monitored because there is a limit on the number of IO devices and the number of bytes on the DP System per process image partition.
- The addresses of the isochronous I/O have to lie in a process image partition.
- The process image partition for isochronous I/O may also contain non-isochronous I/O.
- S7-300/400: More than one process image partition is possible per isochronous mode interrupt OB or IO system.
- S7-1500: Exactly one process image partition is possible per isochronous mode interrupt OB or IO system. Only modules or submodules of the same IO system or the same isochronous mode interrupt OB may be contained in the same process image partition.

Times

- If isochronous mode is active, the update time in the IO devices calculated by STEP 7 must always be identical to the send clock. It is not possible to manually change the update time.
- The time for the application cycle is limited to 32 ms.
- The delay time for the isochronous mode interrupt OB must be between 0 and (data cycle - 1 µs).

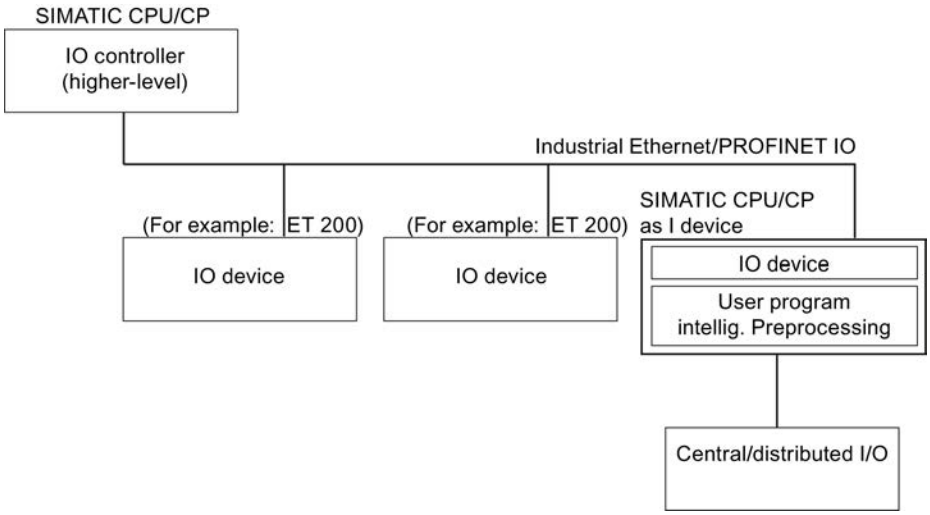
Configuring I-devices

I-Device Functionality

I-Device Functionality

The "I-device" functionality (intelligent IO device) of a CPU allows data to be exchanged with an IO controller and therefore to use the CPU, for example, as an intelligent preprocessing unit of subprocesses. The I-device is linked as an IO device to a "higher-level" IO controller.

The preprocessing is handled by the user program on the CPU. The process values acquired in the central or distributed (PROFINET IO or PROFIBUS DP) I/O are preprocessed by the user program and made available via a PROFINET IO device interface to the CPU or the CP of a higher-level station.



"I-device" naming conventions

In the remainder of this description, a CPU or a CP with the I-device functionality is simply called an "I-device".

Properties and Advantages of the I-Device

Fields of application

Areas of application of the I-device:

- Distributed processing
  A complex automation task can be divided into smaller units/subprocesses. This makes the process easier to handle because the subtasks are simpler.
- Separating subprocesses
  Complicated, widely distributed and extensive processes can be subdivided into several subprocesses with manageable interfaces by using I-devices. These subprocesses can be stored in individual projects if necessary, which can later be merged to form one master project.
- Know-how protection
  Components can only be delivered with a GSD file for the I-device interface description instead of with a project. The know-how of the user program must no longer be published.

Properties

Properties of the I-device:

- Unlinking projects
  Creators and users of an I-device can have completely separated automation projects. The interface between the projects forms the GSD file. This allows a link to standard IO controllers via a standardized interface.
- Real-time communication
  The I device is provided with a deterministic PROFINET IO system via a PROFINET IO interface and therefore supports RT (real-time communication).

Advantages

The I-device offers the following advantages:

- Simple linking of IO controllers without additional software tools
- Real-time communication between SIMATIC CPUs and to standard IO controllers.
- By distributing the computing power among several I devices, the required computing power of the individual CPUs and the IO controller can be reduced.
- Lower communications load by processing process data locally.
- Clarity by processing the subtasks in separate CPUs or projects.

Data Exchange between Higher- and Lower-level IO System

Introduction

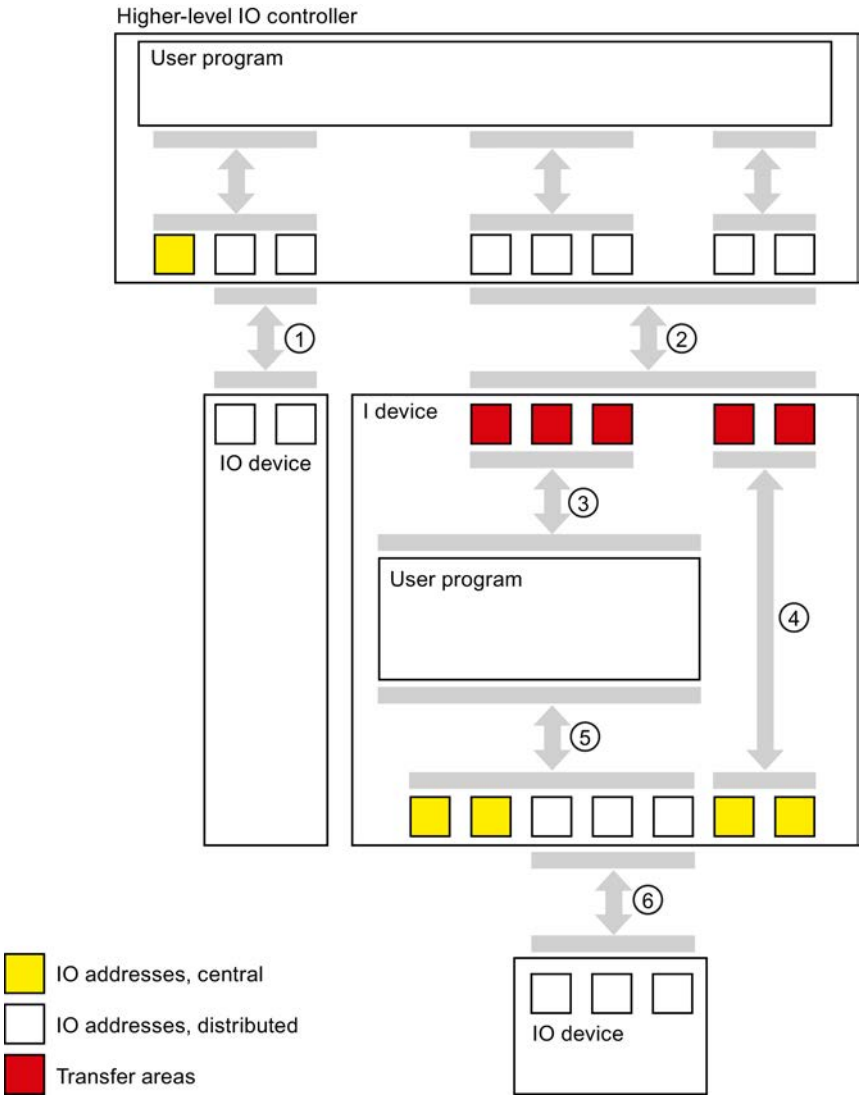
The next section shows the data exchange between a higher- and lower-level IO system.

Transfer areas

The data for communication between IO controller and I device is made available in the transfer areas. A transfer area contains an information unit that is exchanged consistently between IO controller and I device.

Application transfer areas are an interface to the user program of the I-device CPU. Inputs are processed in the user program and outputs are the result of the processing in the user program.

The following figure shows the data exchange between a higher- and lower-level IO system. The individual communication relations are explained below based on the numbers.



① **Data exchange between higher-level IO-controller and normal IO-device**

In this way, the IO controller and IO device exchange data through PROFINET.

② **Data exchange between the higher-level IO controller and I-device**

In this way, the IO controller and the I-device exchange data through PROFINET.

The data exchange between a higher-level IO controller and an I-device is based on the conventional IO controller / IO device relationship.

For the higher-level IO controller, the transfer areas of the I-devices represent submodules of a preconfigured station.

The output data of the IO controller is the input data of the I-device. Analogously, the input data of the IO controller is the output data of the I-device.

③ **Transfer relationship between the user program and the transfer area**

In this way, the user program and the application transfer area exchange input and output data.

④ **Transfer relationship between the transfer area and the I/O of the I-device**

This is not currently supported in the configuration.

⑤ **Data exchange between the user program and the I/O of the I-device**

In this way, the user program and the central / distributed I/O exchange input and output data.

⑥ **Data exchange between the I-device and a lower-level IO device**

In this way, the I-device and its IO devices exchange data. The data transfer is via PROFINET.

Configuring the I device

Introduction

There are basically two possibilities for configuration:

- Configuration of an I-device within a project
- Configuration of an I-device that is used in another project or engineering system.

STEP 7 allows to you configure an I-device for another project or for another engineering system by exporting a configured I-device to a GSD file. You import the GSD file in other projects or engineering systems as with other GSD files. The transfer areas for the data exchange, among other data, is stored in this GSD file.

Configuration of an I-device within a project

1. Drag and drop a PROFINET IO controller from the hardware catalog into the network view.
2. Drag and drop a PROFINET IO controller which can also be configured as IO device from the hardware catalog into the network view. This device is configured as an I-device (for example, CPU 319-3 PN/DP, FW 3.2).
3. Select the PROFINET interface for the I-device.
4. In the area navigation of the Inspector window, select the "Operating mode" entry and select the "IO device" check box.

5. You now have the option of selecting the IO controller in the "Assigned IO controller" drop-down list.
   Once you have chosen the IO controller, the networking and the IO system between both devices are displayed in the network view.
6. With the "Parameter assignment of PN interface by higher-level IO controller" check box, you can specify whether the interface and its ports are set by the I-device itself or a higher-level IO controller.
   If you operate the I-device with a lower-level IO system, then the parameters of the I-device PROFINET interface (for example, port parameter) cannot be set with the higher-level IO controller.
7. Configure the transfer areas. Those are the I/O areas over which the I-device exchanges data with the higher-level IO controller. The transfer areas are found in the area navigation section "I-device communication".
   – Click in the first field of the "Transfer areas" column. STEP 7 assigns a default name which you can change.
   – Select the type of communication relationship: you can currently only select CD or F-CD for "Controller-device communication relation".
   – Addresses are automatically preset; you can correct addresses if necessary and determine the length of the transfer area which is to be consistently transferred.
8. A separate entry is created in the area navigation for each transfer area. If you select one of these entries, you can adjust the details of the transfer area, or correct them and comment on them.
   If the CPU supports it, you can also activate the "Bi-directional address mapping" function. In this case the automatic address areas will be arranged "in the opposite direction", so that CPU input and output areas are occupied symmetrically.

Configuring an I-device with a GSD file

If you use an I-device in another project, or if the I-device is used in another engineering system, then configure the higher-level IO controller and the I-device as described above.

However, click on the "Export" button after configuring the transfer areas so a new GSD file is created from the I-device. This GSD file represents the configured I-device in other projects.

The "Export" button is found in the "I-device communication" section of the Inspector window.

The hardware configuration is compiled and the export dialog opened.

Assign a name for the I-device proxy as well as a description in the fields provided. Click the "Export" button to complete your process.

Finally, import the GSD file, for example, in another project.

See also

Configuring an I-device as a shared device (Page 1528)

Constraints on the Use of I-Devices

Boundary conditions

When using I-devices, certain constraints must be taken into account in certain situations.

Bandwidth

The number of addresses of the configured transfer areas is reflected in the usable bandwidth of the I-device:

Bandwidth of the transfer areas + bandwidth of the lower-level IO system = total bandwidth used on the I-device.

If the address space of the transfer areas is too large, there is not enough bandwidth for the lower-level IO system to achieve fast update times.

Tip: Keep the address space of the transfer area as small as possible.

Rules for RT and IRT communication

IO systems with I-devices are also suitable for setting up real-time applications with RT communication. In the following, a hierarchical system is required, consisting of a higher-level IO controller to which an I-device is assigned (higher-level IO system). IO devices are, in turn, assigned to the I-device (lower-level IO system). The following rules apply:

Both the higher-level and the lower-level IO system support RT communication. You can use RT communication for both IO systems at the same time.

IRT communication can be combined with RT communication. However, IRT communication cannot take place in both IO systems at the same time.

Example:

- If the option "Parameter assignment for the PN interface and its ports on the higher-level IO controller" is activated on the I-device, the I-device can be operated on the higher-level IO controller with the RT class "IRT". IRT communication is not possible for the lower-level IO system of the I-device in this case.
- If the option "Parameter assignment for the PN interface and its ports on the higher-level IO controller" is deactivated on the I-device, the I-device can be operated on the higher-level IO controller with the RT class "RT". IRT communication is possible for the lower-level IO system.

If you configure the I-device on an IO system with the RT class "RT", you cannot set odd send clocks.

Rules for data access

- The higher-level IO controller can access transfer areas when the I-device is in RUN.
- Only the application transfer areas may be accessed in the I-device CPU:
  - Access to input-application transfer areas possible if the higher-level IO-controller is in RUN.
  - Access to output application transfer areas possible independent of the operating state of the higher-level IO controller.

IP address parameters and device name

Just like every other IO device, an I-device also requires IP address parameters / device name to be able to communicate via PROFINET. IP address parameters consist of three parts, the IP address itself, the subnet mask and the address of the router (gateway).

The IP address parameters / device name for an I-device can be assigned in two ways - via the project or directly on the device.

IP address/device name is set in the project:

- The IP address parameters / device name are permanently assigned during configuration (in the I-device project) in STEP 7. This is the standard method.

Set IP address/device name on the device:

- IP address parameter / device name via DCP: The IP-address parameters / device name are assigned via DCP (Discovery and Configuration Protocol). This can happen in two ways:
  - Through a setup tool such as PST or STEP 7 (via accessible devices).
  - Through the higher-level I/O controller
- IP address parameter / device name via user program: The assignment of the IP address parameters occurs in the user program of the I-device CPU (via the instruction "IP_CONFIG").

Note

Topology illustration

If the device name and the IP address parameters are adapted for an I-device directly on the device, the I-device can be displayed multiple times in the topology editor.

Diagnostics and interrupt characteristics

Diagnostics and interrupt characteristics

S7 CPUs have numerous diagnostics and interrupt functions that can, for example, report errors/faults or failures of underlying IO systems. These diagnostics alarms reduce down times and simplify localization and elimination of problems.

Diagnostics of the I-device on the higher-level IO controller and in the I-device

The following mechanisms are available in the higher-level IO controller and the I-device CPU for diagnosing the state of the respective partner:

- Pull/plug interrupt OB (OB 83) for "Return of submodule" interrupt
- Program execution error OB (OB 85) for, for example, process image transfer errors (only S7-300/400)
- Rack failure OB (OB 86) for the failure of expansion units, IO systems or distributed I/O devices (station failure, station return)
- I/O access error OB (OB 122), optional for S7-1500
- Local troubleshooting (only S7-1500)

Note

The diagnostics alarms of the I/O can be processed in the user program of the I-device CPU and passed on from there to the higher-level IO controller via transfer areas.

Operating state changes and station failures / station returns

In the following tables, you can see the effects which the operating state change or the failure of individual CPUs (I device, IO controller) have on each other.

Please note that an S7-1500 CPU calls the OB 122, if it exists, in the event of errors due to direct I/O access to the transfer areas (for example, because the other CPU goes into STOP). However, in contrast to an S7-300/400 CPU, the CPU does not go into STOP if the OB 122 does not exist. With an S7-1500 CPU, local troubleshooting within the block in which the error occurred is also possible. This situation is indicated in the table with the abbreviated wording "Call of OB 122".

Events

Initial status	Event	I device response	Response of higher-level controller
Both CPUs are in RUN operating state (higher-level IO controller and I-device).	The I device CPU changes to STOP.	/	**S7-300/400/1500:** Error code in RET_VAL at updating of the process image with the instruction UPDAT_PI or UPDAT_PO. OB 122 is called in the case of direct access to the transfer areas to the I-device (optional for S7-1500). **S7-300/400**: With a configured OB85 call if I/O access error occurs, OB 85 is called at a process image transfer to the I-device by the system.
I-device CPU is in STOP mode, higher-level IO controller is in RUN mode.	The I device CPU is starting up.	**S7-300/400/1500:** Call of OB 100 (startup). Call of OB 83 for input transfer areas to the higher-level IO controller. Until OB 83 is called, access errors occur in the event of direct access to the input transfer areas to the higher-level IO controller: Call of OB 122. **S7-300/400:** With a configured OB85 call if I/O access error occurs, OB 85 continues to be called at a process image transfer to the higher-level IO controller by the system until OB 83 is called.	**S7-300/400/1500:** Call of OB 83: For all transfer areas to the I-device. Until OB 83 is called, OB 122 continues to be called in the event of direct access to the transfer areas to the I-device. **S7-300/400:** With a configured OB85 call if I/O access error occurs, OB 85 continues to be called at a process image transfer to the I-device by the system until OB 83 is called.

Initial status	Event	I device response	Response of higher-level controller
Both CPUs are in RUN mode (higher-level IO controller and I-device).	The higher-level IO controller changes to STOP	**S7-300/400/1500:** Error code in RET_VAL at updating of the process image with the instruction UPDAT_PI or UPDAT_PO. OB 122 is called in the event of direct access to the input transfer areas to the higher-level IO controller. Note: Output transfer areas can still be accessed. **S7-300/400**: With a configured OB85 call if I/O access error occurs, OB 85 is called at a process image transfer of the input transfer areas to the higher-level IO controller by the system.	/
Higher-level IO controller is in STOP mode, I-device CPU is in RUN mode.	The higher-level IO controller starts up.	**S7-300/400/1500**: Call of OB 83 for input transfer areas to the higher-level IO controller. Until OB 83 is called, OB 122 continues to be called in the event of direct access to the input transfer areas to the higher-level IO controller. **S7-300/400**: With a configured OB85 call if I/O access error occurs, OB 85 is called at a process image transfer of the input transfer areas to the higher-level IO controller by the system.	Call of OB 100 (startup).

Initial status	Event	I device response	Response of higher-level controller
Both CPUs are in RUN (higher-level IO controller and I-device).	Station failure I device, for example, through bus disruption	S7-300/400/1500: If the I device continues to run without a bus connection: Call of OB86 (station failure). Error code in RET_VAL at updating of the process image with the instruction UPDAT_PI or UPDAT_PO. OB 122 is called in the event of direct access to the transfer areas to the higher-level IO controller S7-300/400: With a configured OB85 call if I/O access error occurs, OB 85 is called at a process image transfer of the transfer areas to the higher-level IO controller by the system.	S7-300/400/1500: Call of OB86 (station failure). Error code in RET_VAL at updating of the process image with the instruction UPDAT_PI or UPDAT_PO. OB 122 is called in the case of direct access to the transfer areas to the I-device. S7-300/400: With a configured OB85 call if I/O access error occurs, OB 85 is called at a process image transfer of the transfer areas to the I-device by the system.
Both CPUs are in RUN (higher-level IO controller and I-device). The communication connection between the IO-controller and I-device is disrupted (bus disruption).	The bus connection between the IO-controller and I device is reestablished and the I device is recorded in the usage data traffic again.	S7-300/400/1500: Please note the information under this table about the special considerations when the I-device is powered up. Call of OB 86 (station return). Call of OB 83 for input transfer areas to the higher-level IO controller. Until OB 83 is called, OB 122 continues to be called in the event of direct access to the input transfer areas to the higher-level IO controller. S7-300/400: With a configured OB85 call if I/O access error occurs, OB 85 is called at a process image transfer of the input transfer areas to the higher-level IO controller by the system.	S7-300/400/1500: Call of OB86 (station return) Until OB 86 is called, OB 122 continues to be called in the event of direct access to the transfer areas to the I-device. S7-300/400: With a configured OB85 call if I/O access error occurs, OB 85 is called at a process image transfer of the transfer areas to the I-device by the system.

Note

Special feature when starting up the I device

In contrast to the station return alarm from IO devices in the IO controller, which is covered by calling OB86, the station return alarm of a higher-level IO controller in the I device is divided into 2 parts:

1. Call of OB 86: The initial values for the outputs of the I device are set. The input values, however, are not yet valid; they will first be valid when opening the OB86 in the higher-level IO controller.
2. Call of OB 83 for each input transfer area. The validity of an input transfer area is shown with this call. I-device startup is not complete until OB 83 is called for the input transfer areas. This step can be delayed or not occur at all in the following situations:
   - Higher level IO controller is in STOP: OB 83 is not called until the STOP → RUN transition of the higher-level IO controller.
   - The IRT communication is faulty, for example, because the sync master has failed due to topology errors, etc.: The OB 83 is not started until the IRT communication has been established again.

See also

Basics of error handling (Page 2821)

Insert/remove module interrupt organization block (OB 83) (Page 1786)

Pull/plug interrupt OB (Page 1909)

Rack failure organization block (OB 86) (Page 1796)

Rack error OB (Page 1910)

I/O access error organization block (OB 122) (Page 1814)

I/O access error OB (Page 1914)

Priority class error organization block (OB 85) (Page 1791)

Online diagnostics for I devices

I device online diagnostics is basically similar to online diagnostics of "normal" IO devices.

Displaying diagnostic status by going online

You will receive a selection of the following information depending on which device you selected when you went online:

- If you have selected a higher-level IO controller:
  The diagnostics status of the I device is shown in the project navigation below the IO controller in the "Distributed I/O" folder. No diagnostics status will be shown in the project navigation for the I device itself.
- You have selected the I device that this IO controller is assigned to.
  The diagnostics status of the I device is shown in project navigation. No diagnostics status of the I device is shown in the project navigation below the IO controller in the "Distributed I/O" folder.

Online and diagnostics view

You will receive a selection of the following information depending on which device you selected in the online and diagnostics view:

- You have selected a higher-level IO controller and are connected online with this device:
  The online and diagnostic view displays the I device diagnostics data from the higher-level IO controller. The display corresponds to the display of a "normal" IO controller.
- You have selected the I-device that this IO controller is assigned to and are connected online with this device.
  The online and diagnostics view displays I device diagnostics data from the local view, that is, the view of the I device. The display corresponds to the display of a "normal" IO device.

Display in the list of accessible devices

In the list of accessible devices the I device will be displayed as a "normal" CPU.

Changing IO devices during operation

Docking station - changing IO devices during operation

Using changing IO devices during operation ("changing partner ports") in a docking station

The following Figure shows an automation cell with a docking system and several docking units.

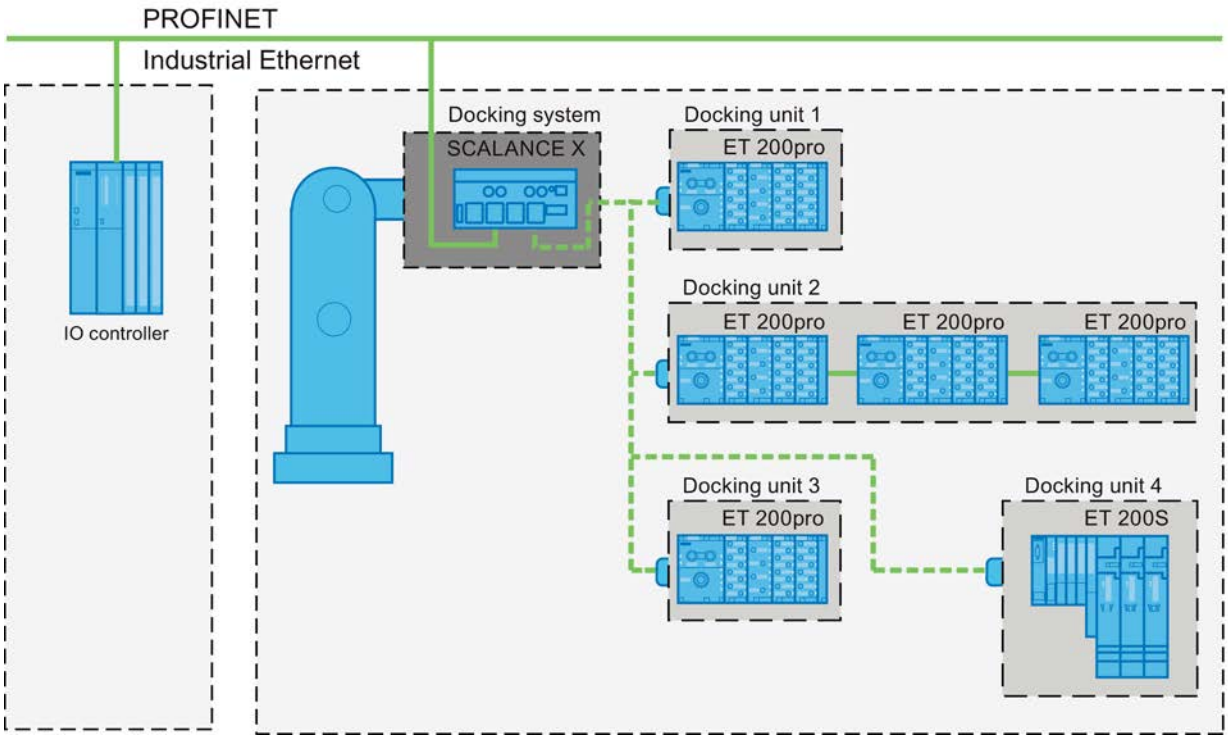


Figure 8-49 Changing IO devices (partner ports) in a docking station

Applicative conditions

The following points should be observed when realizing a docking system with changing IO devices during operation:

- The IO devices of all docking units must be deactivated as default in the configuration.
- Only one docking unit may be active at any given time, in other words only the IO devices of one docking unit can be activated. All IO devices of other docking units must be deactivated or become deactivated before the IO devices of a docking unit can be activated. This is done with the instruction "D_ACT_DP".
- A physical connection to this docking unit and its IO devices must be created in order to activate a docking unit. The IO devices are then switched on (power on). All the IO devices of this docking unit must at the same time be activated in the user program with "D_ACT_DP".

- After the message "IO device activated" you can access the IO device by using the command "Direct I/O access".
- Call the instruction "D_ACT_DP" to activate and deactivate the IO device as close as possible to the start of the OB 1 cycle.

Area of application for changing IO devices during operation

You can use the PROFINET functionality "Changing IO devices during operation" ("changing partner ports"), e.g. for tool change for robots. Typical tools are, for example:

- welding guns and
- holding tools for production parts.

Note

Number of changing IO devices during operation ("changing partner ports") - number of docking units

If you wish to achieve the shortest possible tool-change times, you must observe the following points that are dependent on the CPU or the CP that is being used:

- Only those IO devices that have been configured with the PROFINET function "Prioritized startup" can run in an optimized fashion. The number of IO devices with configuration for this PROFINET function is restricted.
- Only a specific number of IO devices can be enabled at the same time (number depends on the available resources for instruction "D_ACT_DP"). A docking unit should therefore not include more than this number of IO devices. If more IO devices are operated in a docking unit, the IO devices must be enabled one after the other, which takes correspondingly longer.

Example: An S7 CPU 319-3 PN/DP can operate a maximum of 32 IO devices with prioritized startup and can simultaneously activate 8 IO devices with the instruction "D_ACT_DP".

Therefore, for a scheduled optimum use, a docking unit should include no more than 8 IO devices and no more than 32 IO devices should be used in all the changing docking units.

Restriction in the interconnection

The interconnection with a partner port is not possible in the following cases:

- The partner port does not have a suitable type of cable. In this case, a media convertor from the catalog must be inserted.
- The partner port is locked (disabled).
- The two ports that are to be interconnected belong to the same interface (it is possible only to interconnect the ports from different interfaces in a station).
- You are trying to create a ring connection with a module that is not redundant-capable.
- The two ports that are to be interconnected belong to different Ethernet subnets.
- The port of a PROFINET interface of an IO controller cannot be directly configured with the function "Changing IO devices during operation" ("changing partner port").

See also

D_ACT_DP: Enable/disable DP slaves (Page 6486)

Configuring docking units in a docking system

Configuring docking units in a docking system

To configure more docking units in a docking system, follow these steps:

1. Configure your hardware configuration as usual; first, consider the IO devices which correspond to the changing partners to be normal IO devices.
2. In the inspector window, open the properties of the ports on which you would like to operate the changing IO devices (changing partner ports).
3. Under "General > PROFINET interface > Advanced settings > RJ45 100 Mbit Port [Xn Pm] > Port interconnection", select the check box "Alternative partner".
4. Insert the alternative partner in the table, as described in the "Interconnecting ports" chapter.

See also

Interconnecting ports by changing partner ports (Page 1507)

Interconnecting ports by changing partner ports

Requirement

- All devices involved support topology configuration
- The IO controller, the changing IO devices (docking unit) and the switch (docking system) to which the changing IO devices are to be operated, must support this feature.
- The docking unit must be connected with a switch that supports the PROFINET function "prioritized startup" (e.g., from the SCALANCE X200IRT family).

Example

- IO controller: CPU 416-3 PN/DP
- Docking system: Scalance X-200IRT
- Docking units (changing IO devices): for example, 2 x ET 200pro (IM 154-4)

Procedure

In the description of the procedure, it is assumed that the configuration matches that of the above-described example. To interconnect ports, follow these steps:

1. Connect all PROFINET interfaces of the devices to each other in the network view.
2. In the network view or device view, select the docking system (Scalance X200 IRT) whose port you want to interconnect.
3. Open the device properties in the Inspector window.
4. Under "Properties > PROFINET interface > Advanced settings > RJ45 100 Mbit port [Xn Pm] > Port interconnection" choose the partner port for the CPU (IO controller) from the "Partner port" drop-down list.
   The ports of the CPU and the docking system (Scalance) are interconnected.
5. The docking system is still selected: For any other unconnected port, now select the partner ports for the IO devices:
   - Select the "Alternative partner" check box.
     A table for the partner port is displayed.
   - Within the table, click the black triangle of the first row to open a drop-down list.
     The drop-down list shows all devices with their unconnected ports for selection.
   - Select the port of the first docking unit.
   - Click the next line in the table and select another alternative partner port.
   - Repeat the steps outlined above until all required partner ports are recorded in the table.

Note

Use of a CP 1243-1 (WAN-CP)

When a CPU S7-1200 with CP 1243-1 is used, the procedure is the same as above, except the Ethernet interface of the CP is used instead of the PROFINET interface of the IO controller.

Interconnecting several IO devices with changing IO devices during operation

Configure IO devices on a docking unit which are interconnected in line with IO devices changing during operation ("Alternative partner") with the port of the changing IO device as usual; this means the IO device interconnected in line then has no changing partner port.

Deleting the partner port

To remove the interconnection of the partner port that changes during operation, follow these steps:

1. In the table of alternative partners, click the row that contains the alternative partner port that is to be deleted and open the drop-down list for the partner port (black triangle).
2. Select "Any partner" from the list.
   The alternative partner port is removed from the table.

See also

Enabling device replacement without exchangeable medium (Page 1461)

What is prioritized startup? (Page 1509)

Optimizing startup time

What is prioritized startup?

Definition

Prioritized startup denotes the PROFINET function for accelerating the startup of IO devices in a PROFINET IO system with RT and IRT communication. It shortens the time that the respective configured IO devices require in order to reach the cyclic user data exchange in the following cases:

- After the supply voltage has returned
- After a station has returned
- After IO devices have been activated

Benefits

The PROFINET functionality "prioritized startup" enables PROFINET IO applications in which machine parts or tools and their IO devices have been permanently replaced. Waiting times of several seconds between the scheduled processes of the restart are reduced to a minimum by this optimization. This accelerates the production process with removable IO devices, e.g. in tool-changer applications and enables a greater throughput in production.

The PROFINET functionality "prioritized startup" also offers a considerable increase in performance for applications where a quick startup time of the IO devices after "power on" or after station failure / station return is required or when activating IO devices.

Properties

You can realize the following properties with the PROFINET function "Prioritized startup":

- Communication readiness of the IO devices (distributed I/O) up to minimum 500 ms.
- You can use the prioritized startup for IO devices (distributed I/O) both with RT as well as with IRT communication.

Startup times

The length of the startup time of an IO device (distributed I/O) with the PROFINET function "Prioritized startup" is dependent on the following points:

- IO devices used (distributed I/O)
- IO structure of the IO devices (distributed I/O)
- Used modules of the IO devices (distributed I/O)

- Used IO controller
- Used switch
- Port setting
- Cabling
- Configured RT class of the IO device

Note

Prioritized startup after a startup for the first time

A prioritized startup of the IO device is always available to you after the first configuration of this IO device in the very first startup of the PROFINET IO system. Even in the case of spare parts or a reset to factory settings, the first startup is a standard startup for the respective configured IO devices.

Note

In the following situations, despite prioritized startup, there may be startup times of up to 8 s:

- An IO/device is disconnected and reconnected within 8 seconds.
- On a docking point, you can dock multiple physical IO devices as an IO device with a certain device name and a certain IP configuration (for example, docking point for automatic transport system).

If you wish to achieve a shortest possible startup time of 500 ms, you must perform the following measures:

- Port settings on an IO device
- Wiring dependant on the interconnected PROFINET device
- Measures in the user program

Setting prioritized startup

Requirement

You can enable the PROFINET function "Prioritized startup" for the IO devices (distributed I/O) only in the following cases:

- The IO controller used can prioritize selected IO devices during startup.
- The IO device used supports prioritization.

Note

Prioritized startup

In the case of an accelerated startup (prioritized startup), you must observe special conditions when setting up the PROFINET interface and when wiring if you wish to achieve the shortest possible startup times.

Procedure

1. Select the IO device in the network view or device view for which you wish to accelerate startup.
2. Open the IO device properties in the inspector window.
3. Select "PROFINET interface > Advanced options > Interface options".
4. Select the "Enable prioritized startup" check box.
5. Download the configuration to the IO controller.

Note:

For devices whose DP mode can be selected, the previously described check box can be found under "PROFINET interface > Operating mode". The CP 1616 for PC stations is one example of this type of device.

Note

Number of IO devices (distributed I/O) with prioritized startup

You can only start up a set maximum number of IO devices with PROFINET functionality "Prioritized startup" within one IO system. This maximum number depends on the IO controller used.

See also

Optimizing port settings on the IO device and IO controller (Page 1511)

Optimizing port settings on the IO device and IO controller

Optimizing port settings on the IO device and IO controller

The transfer medium and the duplex option are checked during startup of the IO device for control unit wiring.

These checks require time, but with specific presets of these options you can save time. Make certain that the settings made correspond to the actual conditions (using the correct wiring).

Synchronizing the properties of local port and partner port

To synchronize the properties for the local port and partner port, follow these steps:

1. First, select the relevant local port and then the partner port and perform the following steps for both ports.
2. Synchronize the "Transmission medium/duplex" setting of the respective ports in the properties under "Port options". Use the setting "TP 100 Mbps full duplex".
3. Under "Port options", clear the check box "Enable autonegotiation" for both ports.

See also

Wiring rules for disabled autonegotiation (Page 1459)

Configuring media redundancy

What you should know about media redundancy

Media redundancy options

To increase the network availability of an industrial Ethernet network with optical or electrical line topology, the following options are available:

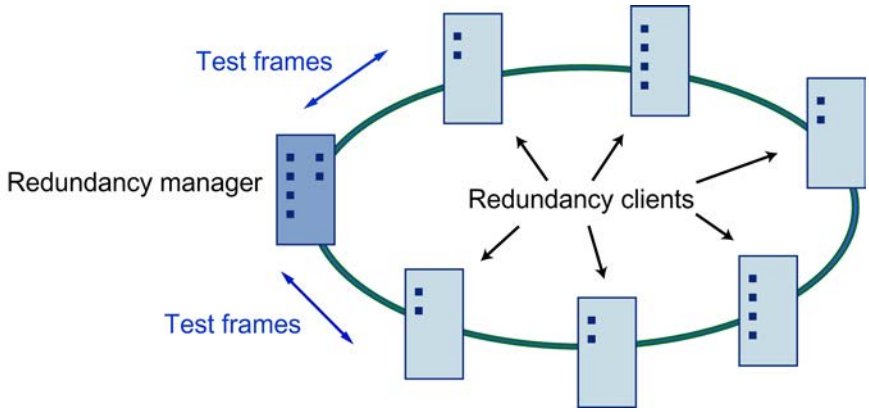
- Meshing networks
- Parallel switching of transmission modes
- Consolidation of a line topology to a ring topology

Media redundancy in ring topologies

Devices in ring topologies can be the external switches and/or integrated switches of communication modules.

To set up a ring topology with media redundancy you must join both free ends of a line-shaped network topology to one device. Consolidation of a line topology to a ring takes place over 2 ports (ring ports) in a device in the ring. This device is the redundancy manager. All other devices in the ring are the redundancy clients.

The following figure shows the ring-shaped interconnection of the IO devices:



The two ring ports in a device are the ports which create the connection to both of its neighboring devices in a ring topology. The selection and set up of the ring ports takes place during configuration of the respective device.

Download the configuration in the individual devices before physically connecting the ring.

Functions of media redundancy in a ring topology

If a part of the ring is disconnected, then the data paths between the individual devices is automatically reconfigured. After the reconfiguration the devices are once again accessible in the newly created topology.

In the redundancy manager, 2 ring ports will be separated from each other during interruption-free network operation, so that no data frames circulate. The ring topology is a line when viewed from the standpoint of data transfer. The redundancy manager monitors the ring topology. To do so it sends a test frame not only from ring port 1 but also from ring port 2. The test frame passes through the ring in both directions until it arrives at the other ring port on the redundancy manager.

An interruption in the ring can occur from connection failure between two devices or device failure in the ring.

If the redundancy manager's test telegram does not reach the other ring port because of interruption in the ring, the redundancy manager connects via both ports. With this alternate route a functioning connection is again created among all remaining devices in a line-shaped topology.

The time between ring interruption and re-establishment of a functional line topology is called the reconfiguration time.

As soon as the interruption is eliminated, the original transmission route is again created, both ring ports in the redundancy manager are separated from each other, and the redundancy clients are notified of the change. The redundancy clients then use the new route to the other devices.

If the redundancy manager fails, then the ring is changed to a functional line.

Media redundancy processes

In STEP 7 you can use the media redundancy process MRP (Media Redundancy Protocol) for ring topologies; up to 50 devices per ring can be included.

Media Redundancy Protocol (MRP)

Media Redundancy Protocol (MRP)

The MRP process works in conformity with Media Redundancy Protocol (MRP), which is specified in IEC 61158 Type 10 "PROFINET".

The reconfiguration time following an interruption of the ring amounts to a maximum of 0.2 seconds.

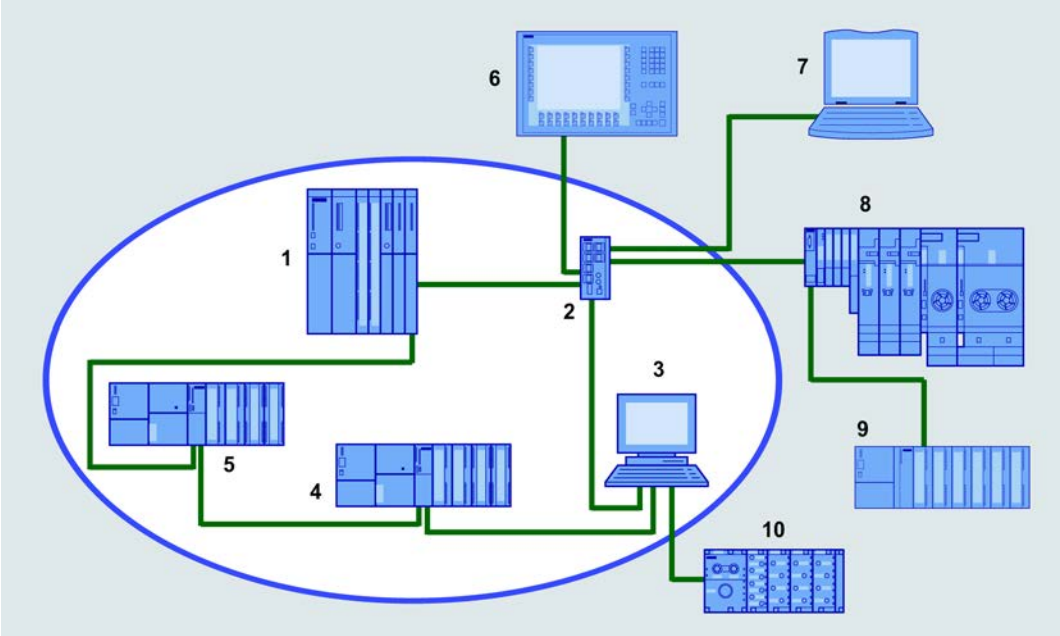
Requirements for trouble-free operation

Requirements for trouble-free operation with media redundancy process MRP are:

- MRP is supported in ring topologies of up to 50 devices. Exceeding the device limit can lead to data traffic failure.
- All devices must be connected to each other via their ring ports.
- "MRP" has to be activated in all devices in the ring.
- The connection settings (transfer medium / duplex) must be set to full duplex and for at least 100 Mbit/s for all ring ports. Otherwise data traffic failure can occur.
- In STEP 7 configuration in the properties dialog for all ports in the ring, set the connection in the register "Options" to "Automatic settings".
- The ring in which you would like to install MRP can only consist of devices which support this function. Following are example of such devices:
  - Industrial Ethernet switches like SCALANCE X 200 as of firmware version V4.0 and SCALANCE X 200 IRT as of firmware version V4.0.
  - Communications processors like CP 443 1 Advanced (6GK7 443 1GX20 0XE0) as of firmware-version V2.0, CP 343 1 Advanced (6GK7 343 1GX30 0XE0) as of firmware version V1.0, CP 1616 (6GK1 161 6AA00) as of firmware version V2.2, CP 1604 (6GK1 160 4AA00) as of firmware version V2.2
  - Other IEC 61158 Type 10 "PROFINET" conforming MRP devices.

Port interconnection

The following figure shows possible topology for devices in a ring with MRP. The devices within the white oval are found in the redundancy domain.



①	S7-400 with CP 443-1 Advanced
②	Switch SCALANCE X206-1
③	PC with CP 1616
④	S7-300 with CP 343-1 Advanced
⑤	S7-300 with CP 343-1 Advanced
⑥	HMI station
⑦	Programming device/PC
⑧	ET 200S
⑨	ET 200M
⑩	ET 200pro

For ring topology with media redundancy per MRP process the following rules apply:

- All devices within the ring topology are members of the same redundancy domain.
- A device within the ring is the redundancy manager.
- All other devices in the ring are the redundancy clients.

Devices not capable of MRP can be connected to the ring with a SCALANCE X switch or a PC with CP 1616.

Prioritized startup

If you configure MRP in a ring, then you can not use the "prioritized startup" function in involved devices in PROFINET applications.

If you want to use the "prioritized startup" function, then you must deactivate MRP in the configuration.

See also

Configuring media redundancy (Page 1516)

Configuring media redundancy

Requirements

- The components involved must support the media redundancy protocol (MRP).
  These include the following components:
  - ET 200S as of V7.0
  - ET 200SP
  - ET 200MP
  - S7-300 CPUs as of V3.2
  - S7-400 CPUs as of V6.0
  - S7-1500 CPUs
- IRT communication is not configured, i.e., no PROFINET interface is set to "sync master" or "sync slave" (inspector window "Advanced options > Real-time settings > Synchronization").

Procedure

Media redundancy must be configured or all devices that should be operated as media redundant. Follow these steps:

1. Configure the PROFINET IO device. The PROFINET device must support MRP.
2. Establish a ring by means of port interconnections (e.g. in the topology view).
   If no rules are violated, an MRP domain will be created automatically.
3. Check the media redundancy roles, e.g., who has the role of redundancy manager.
   In the properties of a PROFINET interface of a CPU 319-3 PN/DP (FW 3.2), for example, select the media redundancy role "Manager (auto)" under "Advanced settings". There can be multiple devices with this role in a ring; the actual manager is negotiated among the Managers (auto).
   The default ring ports are automatically displayed in the subjacent fields "Ring port 1" and "Ring port 2".
   If the ports are set at the factory, then the fields are unavailable. Otherwise, select one at a time those ports which you want to configure as ring port 1 and ring port 2.
4. Select the media redundancy role "client" for the rest of the devices at the same spot.
   The default ring ports will automatically be displayed in the subjacent fields.
5. For PROFINET devices which you do not want to operate within ring topology with MRP, assign the option "Not member of the ring".

6. Mark the option "Diagnostics interrupts" if diagnostics interrupts should be issued regarding the MRP status in the local CPU.
7. If you operate a PROFINET device with more than two ports in a ring, then you should prevent sync frame feed in the ring by setting a sync boundary (on the ports that are not in the ring). This setting is made under "Advanced settings > Port [Xn Pm] > Port options > Boundaries".
8. Download the configuration to all involved devices.
   The MRP settings are effective after the device is restarted, or after a power failure and restart.

Rules

- All devices in the ring must support MRP.
- Only one PROFINET interface in the ring with the media redundancy role "Manager" is allowed. No other PROFINET interfaces in the ring with the "Manager" or "Manager (auto)" role are allowed.
- Multiple PROFINET interfaces with the "Manager (auto)" media redundancy role are allowed.
- At least one PROFINET interface in the ring must have the "Manager (auto)" media redundancy role.
- All PROFINET interfaces in the ring must belong to the same MRP domain. A PROFINET interface can only belong to one MRP domain.

Diagnostic interrupts

The following diagnostics interrupts can be configured:

- Wiring or port error
  Diagnostics interrupts will be generated for the following errors in the ring ports:
  - A neighbor of the ring port does not support MRP
  - A ring port is connected to a non-ring port
- Interruption / return (only redundancy manager)
  If the ring is interrupted and the original configuration is returned, a diagnostics interrupt will be generated.
- Additional redundancy manager interrupts
  There are additional diagnostics interrupts, for example when multiple devices of the "manager" type are available in the ring.

Note

To assure trouble-free operation when a third party device is set as redundancy manager in a ring, you must assign the role "Redundancy client" to all other devices in the ring before you close the ring. Otherwise circulating data frames and network failure could occur.

See also

Media Redundancy Protocol (MRP) (Page 1514)

Managing MRP domains

You can monitor and adjust the media redundancy settings centrally.

Requirements

- The components involved must support the media redundancy protocol (MRP).
- The devices are configured and networked.

Procedure

To open the MRP domain, follow these steps:

1. Select any PROFINET interface, and select the "Advanced options > Media redundancy" area in the inspector window.
2. Click the "Domain settings" button.

or

1. Select the subnet or the IO system.
2. Navigate to the "PROFINET > Domain management > MRP domains" area in the inspector window.

Here, you can specify centrally which MRP domain is to be the default domain. You can change the names of all domains and check redundancy roles and ring ports.

For each MRP domain, you receive an overview of how many PROFINET interfaces are interconnected in the ring, how many PROFINET interfaces are located outside the ring, and the number of redundancy managers and redundancy clients.

You can have the devices in the ring, including device name, associated MRP domain, media redundancy role, and utilized ring ports, displayed in groups:

- MRP domains
  The list shows you the existing MRP domains of the selected IO system.
- <Name of MRP domain>
  The list shows you the number of PROFINET interfaces and devices with the respective redundancy roles for the selected MRP domain.
- Devices
  The list shows you all configured devices of the selected IO system.

Configuring PROFIenergy

Overview of PROFIenergy

Saving energy with PROFIenergy

PROFIenergy is a PROFINET-based data interface for switching off consumers centrally and with full coordination during pause times regardless of the manufacturer or device type. Through this, the process should only be provided with the energy that is absolutely required. The majority of the energy is saved by the process; the PROFINET device itself only contributes a few watts of savings potential.

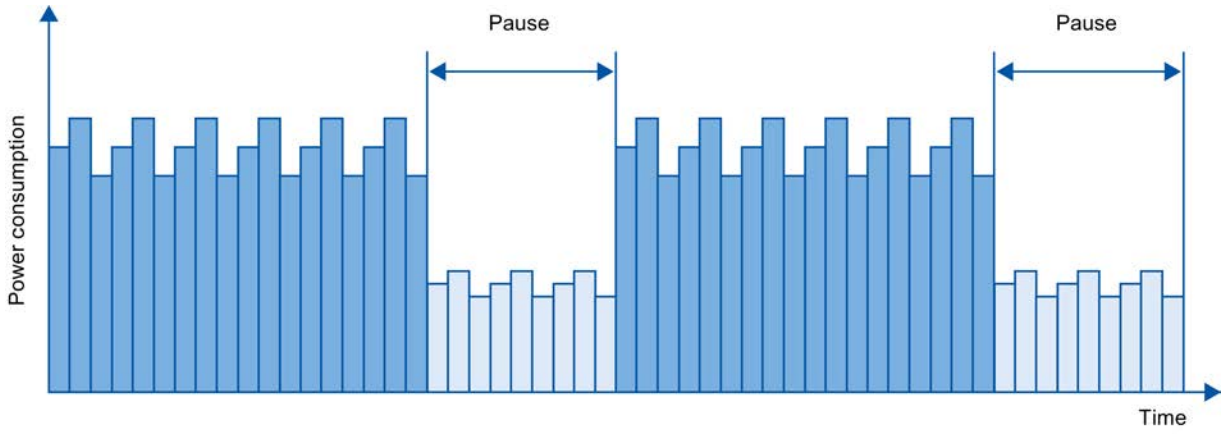


Figure 8-50 Energy savings during pauses with PROFIenergy

Basic information

Turning off the PROFINET devices or the power modules occurs through special commands in the user program of the PROFINET IO Controller. No additional hardware is needed, the PROFIenergy commands are directly interpreted by the PROFINET devices.

Principle of operation

At the beginning and end of pauses, the system manager enables or disables the pause function of the system; then the IO controller sends the PROFIenergy command "Start_Pause" / "End_Pause" to the PROFINET device. The device interprets the contents of the PROFIenergy command and turns on / off.

Through other PROFIenergy functions, device information can be accessed during pauses. The user can use this information in order to punctually transfer the "Start_Pause" / "End_Pause" command.

PROFIenergy instructions for IO controller

Two instructions are needed for controlling and monitoring the PROFIenergy functions.

The PE_START_END instruction allows you to easily activate and deactivate the idle state of PROFINET devices. This occurs by means of an incoming edge or outgoing edge. The PE_START_END instructions provides a simple interface for realizing PROFIenergy commands Start_Pause and End_Pause.

The PE_CMD instruction allows you to transmit all PROFIenergy commands, including Start_Pause and End_Pause. With the other commands, for example, the current status of the PROFINET device or the behavior during the pauses can be inquired. The PE_CMD instruction is a convenient means for handling all PROFIenergy functions.

PROFIenergy instruction for I-devices

The PE_I_DEV instruction allows you to realize PROFIenergy on I-devices, as well. The instruction receives PROFIenergy commands on the I-device and forwards these to the user program for processing. After processing the command, the user program calls the PE_I_DEV instruction again in order to send the acknowledgement to the IO controller. For these replies, each command offers you a helper instruction that supplies the reply data to the instruction.

The instructions can be found in the "Instructions" task card of the STEP 7 program editor.

An application example can be found on the Internet (http://support.automation.siemens.com/WW/view/en/41986454) in the Service and Support portal.

Configuration and programming

The functions can be comfortably integrated into existing systems. No configuration is required for the use of PROFIenergy. However, amendments to the user program are required:

- Before the "Start_Pause" command, you must ensure that your system is brought into a condition that is suitable for a pause.
- A sequential control system for the beginning of the pause of the devices and for the punctual restarting of the device on break must be programmed (depending on the required startup times that the respective PROFINET device demands).
- The error messages of the PE_CMD instruction must be evaluated, and the required reaction must be programmed (for example, cancellation or continuation of further commands on lower-level PROFINET devices).

See also

PROFIenergy (Page 4628)

Configuring PROFIenergy for ET 200S

The ET 200S distributed I/O system offers a special feature: for certain interface modules you can specify which potential groups or power modules of the IO device are activated for PROFIenergy.

Special features for ET 200S HF V7.0 and ET 200S FO V7.0 interface modules

You can configure the initial behavior of the power modules (e.g., the switchable power module (PM-E DC24V/8A RO)) for the above-named IO devices so that you do not have to bring the

device to the energy saving state in the startup program with the PROFIenergy PE_Start_End_Pause instructions.

You activate the energy saving function in the properties of the power module (module parameters area, section "PROFIenergy").

Note that provisions are necessary in the user program for the use of PROFIenergy functions.

Rules

The configuration option described above is not available for ET 200S devices that were incorporated in the hardware catalog using GSDML files.

Configuring Shared Devices

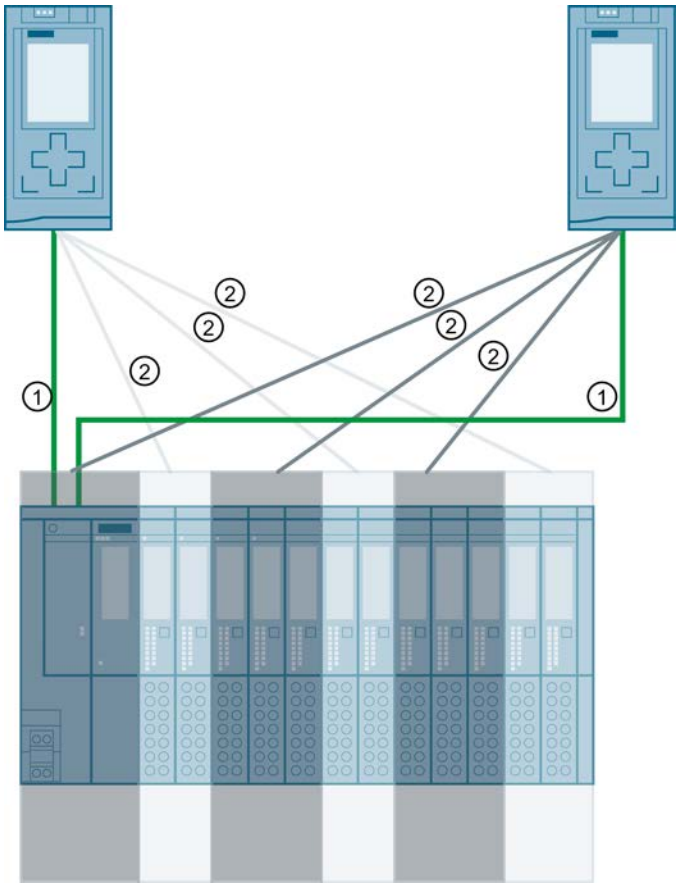
Useful information on Shared Devices

Shared Device functionality

Numerous IO controllers are often used in larger or widely distributed systems.

Without the "Shared Device" function, each I/O module of an IO device is assigned to the same IO controller. If sensors that are physically close to each other must provide data to different IO controllers, several IO devices are required.

The "Shared Device" function allows the modules or submodules of an IO device to be divided up among different IO controllers. Thus allowing flexible automation concepts. You have, for example, the possibility of combining I/O modules lying near other into an IO device.



① PROFINET
② logical assignment

Principle

Access to the submodules of the Shared Device is then divided up among the individual IO controllers. Each submodule of the Shared Device is assigned exclusively to one IO controller.

Requirement (GSD configuration)

- STEP 7 as of V12 Service Pack 1
- CPU 15XX as of FW 1.1 as IO controller
- IO device supports the Shared Device functionality, e.g. interface module IM 155-5 PN ST
- GSD file for configuring the IO device is installed

Comments:

- An S7-1500 CPU configured as an I-device can be used as a Shared Device as of firmware version 1.5. The I-device must be integrated by GSD import into the IO system of the IO controller.
- An S7-300 CPU or S7-400 CPU configured as an I-device can be used as a Shared Device under the following condition: You must export the PROFINET GSD file for the I-device from STEP 7 (as of V5.5) and then import it into STEP 7 (TIA Portal).

Configuring the access

The IO device must be present in several projects so that the modules or submodules of an IO device can be assigned to different IO controllers. A separate project is required for each IO controller.

You use the "Shared Device" parameter of the interface module to determine the modules or submodules to which the IO controller has access:

- If the local IO controller has access to the configured module, select the name of the IO controller from the list.
- If the IO controller from a different project and not the local IO controller is to have access to the configured module, select the entry "---".

The configuration is consistent regarding access if each module or submodule in exactly one project is assigned to an IO controller.

Module or submodule is assigned to another IO controller

The paragraph below describes the consequences of the "---" setting of the "Shared Device" parameter from the point of view of the local IO controller.

In this case, the local IO controller does not have access to the module configured in this way. Specifically, this means:

- No data exchange with the module or submodule
- No reception of alarms or diagnostics, which means no display of the diagnostics status in the online view
- No parameter assignment of the module or submodule

Setting of the real-time properties

STEP 7 calculates the communication load and thus the resulting update times. You must enter the number of project-external IO controllers in the project in which the PROFINET interface of the Shared Device is assigned to the IO controller so that a calculation is possible with Shared Device configurations.

The maximum possible number of IO controllers for the Shared Device depends on the device. This number is stored in the GSD file of the Shared Device.

You can set a very short send clock with an S7-1500 CPU as IO controller. The send clock can be shorter than the shortest send clock supported by the Shared Device. In this case, the Shared Device is operated by the IO controller with a send clock that it supports (send clock adaptation).

Example:

An S7-1500 CPU supports send clocks starting from 0.25 ms. A configured IO device also supports send clocks starting at 0.25 ms; another IO device supports send clocks starting at 1 ms. In this case, you have the option of setting the short send clock of 0.25 ms for the CPU. The CPU operates the "slow" IO device with the send clock of 1 ms, for example.

This function is not available if you do not configure one of the IO controllers with access to the Shared Device with STEP 7. In this case, you must read out the shared send clock in the properties of the Shared Device in STEP 7 and enter it in the engineering tool for the IO controller.

Rules for the configuration

- IO controllers that use the Shared Device are created in different projects. In each project, care must be taken that the Shared Device is configured identically in each station. Only one IO controller may ever have full access to a submodule. Inconsistencies in the configuration result in a failure of the Shared Device.
- I/O addresses of a module or submodule can only be edited if a module or submodule is assigned to the IO controller in the same project.
- The Shared Device must have the same IP parameters and the same device name in each project.
- The send clock must be identical for all IO controllers that have access to the Shared Device.
- The S7 subnet ID of the subnet to which the Shared Device is connected must be identical in all projects.
- The following functions are only available if the PROFINET interface of the Shared Device is assigned to the local IO controller:
  - IRT operation
  - Prioritized startup
  - Parameter assignment of the port properties

Boundary conditions

The following boundary conditions result because a Shared Device configuration is distributed across several projects:

- The addresses of modules or submodules that are not assigned to this IO controller are missing in the address overview of each IO controller that has access to a Shared Device.
- The modules or submodules that are not assigned are not taken into consideration in the configuration limit calculation for the Shared Device during the consistency check. This means that you have to check yourself that the maximum number of submodules or the maximum amount of cyclic IO data is not exceeded for the Shared Device.

- Configuration errors such as the assignment of a module or submodule to several IO controllers are not detected in STEP 7.
- CPUs that are loaded with a Shared Device configuration do not have any information on whether the IO device is a Shared Device. Modules or submodules that are assigned to other IO controllers and therefore other CPUs are missing in the loaded configuration. These modules or submodules are therefore displayed neither in the CPU web server nor in the CPU display.

See also

Displaying IO controllers that access modules of a Shared Device (Page 2094)

Configuring an I-device as a shared device (Page 1528)

Configuring Shared Device (GSD configuration)

Below, you will find a description of how to configure a distributed I/O system as a shared device with STEP 7 V12, Service Pack 1 or higher.

A "distributed" configuration with different engineering tools for different IO controller families is always possible. However, the description of the procedure is exclusively based on STEP 7 V12, Service Pack 1. The description is limited to two IO controllers of the S7-1500 family that share one shared device.

Two projects are created (Shared-Device-1 and Shared-Device-2), each with one IO controller (PLC1 and PLC2). You must create the shared device in both projects, even though it is physically one and the same IO device.

Requirement

- STEP 7 as of V12 Service Pack 1
- In the context of STEP 7 V12, Service Pack 1: Only CPU 15XX as of FW 1.1 can be configured as IO controller (no CPU 31X, CPU 41X or CPU 12XX).
- In the context of other IO controllers that are to have access to a shared device: Corresponding engineering tool, for example, STEP 7 V5.5, Service Pack 3 for CPU 31x-3 PN/DP.
- IO device supports shared device functionality, e.g., interface module IM 155-5 PN ST V2.0.
- GSD file for configuring the IO device as shared device is installed.

Procedure - Creating project 1

To create the first project with a shared device, follow these steps:

1. Start STEP 7.
2. Create a new project with the name "Shared-Device-1".
3. Insert a CPU 1513-1 PN from the hardware catalog in the network view. Name it "PLC1".

4. Insert an IO device with the function "shared device" from the hardware catalog (hardware catalog: Other field devices > PROFINET IO > I/O).
5. Assign the IO controller "PLC1" to the IO device.
6. Double-click the IO device and insert all required modules and submodules from the hardware catalog in the device overview table.
7. Assign the module parameters.
8. Save the project.

Procedure - Creating project 2

To create the second project with a shared device, follow these steps:

1. Start STEP 7 once again.
   A new instance of STEP 7 opens.
2. In the new instance, create a new project with the name "Shared-Device-2".
3. Insert a CPU 1513-1 PN in the network view. Name it "PLC2".
4. Copy the IO device from the project "Shared-Device-1" and insert it in the network view of project "Shared-Device-2".
5. Assign the IO controller "PLC2" to the IO device.
6. Save the project.

Both projects now have an identically structured IO device that must be configured in the next step for the different types of IO controller access.

Procedure - Configuring access to the shared device

The modules and submodules you insert in the shared device are automatically assigned to the local CPU. To change the assignment, follow these steps:

1. Select the interface module in the network or device view of project "Shared-Device-1".
2. Select the "Shared device" area in the Inspector window.
   A table shows which CPU has access to the respective module or submodule for all configured modules. The default setting is that the local CPU has access to all modules and submodules.
3. Keep the "PLC1" setting for all modules and submodules that are to remain in the address range of the local CPU
   Select the setting "---" for all modules and submodules that are to be located in the address range of the CPU from the "Shared-Device-2" project (PLC2). This means that an IO controller outside the project is to have access to the module or submodule.
4. Select the interface module in the network or device view of project "Shared-Device-2".
5. Select the "Shared device" area in the Inspector window.
   A table shows which CPU has access to the respective module or submodule for all configured modules.
6. Select the setting "---" for all modules and submodules that are to be located in the address range of the CPU from the "Shared-Device-1" project (PLC1).

7. Finally, check whether the settings for access are "complementary" for each module or submodule in both projects. This means that if the local CPU has access in one project, the option "---" must be set in the other project and vice versa.
   Special note: The option "---" for the PROFINET interface and therefore for the ports makes the associated parameters read-only and not changeable. Parameters of the PROFINET interface and port parameters can only be edited in the project in which the PROFINET interface is assigned to the local CPU. The ports can be interconnected in both projects regardless of this.
8. Check whether the same IP address parameters and device name are set for the shared device in all projects.
   Check whether the same S7 subnet ID is set in all projects for the subnet to which the shared device is connected (subnet properties, "General" area in the Inspector window).

Note

If you make changes to the shared device: Make the same changes in each project on the shared device. Make sure that only one IO controller has access to a module or submodule.

Procedure - Adjusting the real-time settings

To ensure that all IO controllers and shared devices are operated with the appropriate send clock and that the update times are calculated correctly based on the communication load, you must adjust and check the following settings:

1. Select the project whose IO controllers have access to the PROFINET interface and the ports of the shared device.
2. Select the interface module of the shared device in the network view.
3. In the Inspector window, navigate to the "PROFINET interface > Advanced options > Real time settings > IO cycle" area.
4. In the "Shared device" area, set the number of project-external IO controllers. The maximum number depends on the IO device (default in GSD file).
5. You must set the same send clock for each IO controller that has access to modules and submodules of the shared device:
   - If you configure the IO controller with STEP 7 (TIA Portal):
     Open the corresponding project.
     Select the PROFINET interface of the IO controller.
     Select the "Advanced options > Real time settings > IO communication" area in the Inspector window and set the shared send clock.
   - If you configure the IO controller with a different engineering tool:
     Select the PROFINET interface of the shared device in STEP 7 (TIA Portal) and read out the send clock on the shared device ("Advanced options > Real time settings" area)
     Enter the read send clock in the engineering tool.
   - Special note: If you configure **all** IO controllers that have access to the shared device in STEP 7 (TIA Portal or V5.5), you can set shorter send clocks on the IO controller than supported by the shared device (send clock adaptation).

Compiling and loading

You must compile the configurations for the different IO controllers and load them to the CPUs one after the other.

Due to the distributed configuration with separate projects, STEP 7 does not output consistency errors in the case of incorrect access parameter assignment. Example for incorrect access parameter assignment:

- Several IO controllers have access to the same module
- IP address parameters or send clocks are not identical

These errors do not show up until operation and are output as configuration errors, for example.

Configuring an I-device as a shared device

Below, you will find a description of how you configure an S7-1500 as an I-device with STEP 7 Version V13 or higher and then use it in two projects as a Shared Device.

A "distributed" configuration with different engineering tools for different IO controller families is generally also possible here. The procedure described below is based on STEP 7 V13 and is limited to a configuration with two IO controllers of the S7-1500 family that share the transfer areas of an I-device as a Shared Device. The I-device itself is also an S7-1500 CPU.

Three projects are created with one IO controller each (PLC-I-Device, PLC_1, and PLC_2).

PLC-I-Device is used to configure the I-device. The PROFINET GSD variant of PLC-I-Device is used in the PLC_1 and PLC_2 projects in order to assign the transfer areas in the respective higher-level IO controller.

Shared I-device concept

For the introduction of the shared I-device concept, two roles are distinguished:

- The role of manufacturer (e.g., machine manufacturer): The manufacturer configures and programs an I-device that performs a particular automation task. Transfer areas are defined as the I/O interface to the operator of the machine. These transfer areas can be assigned to different IO controllers. For the connection to higher-level IO controllers, the manufacturer provides a PROFINET GSD file and discloses the transfer areas via which the I-device can be accessed.
- The role of the operator: The operator uses the I-device as a PROFINET GSD variant during configuration of the PROFINET IO system and, in this process, specifies the I/O addresses under which the IO controllers access the transfer areas.

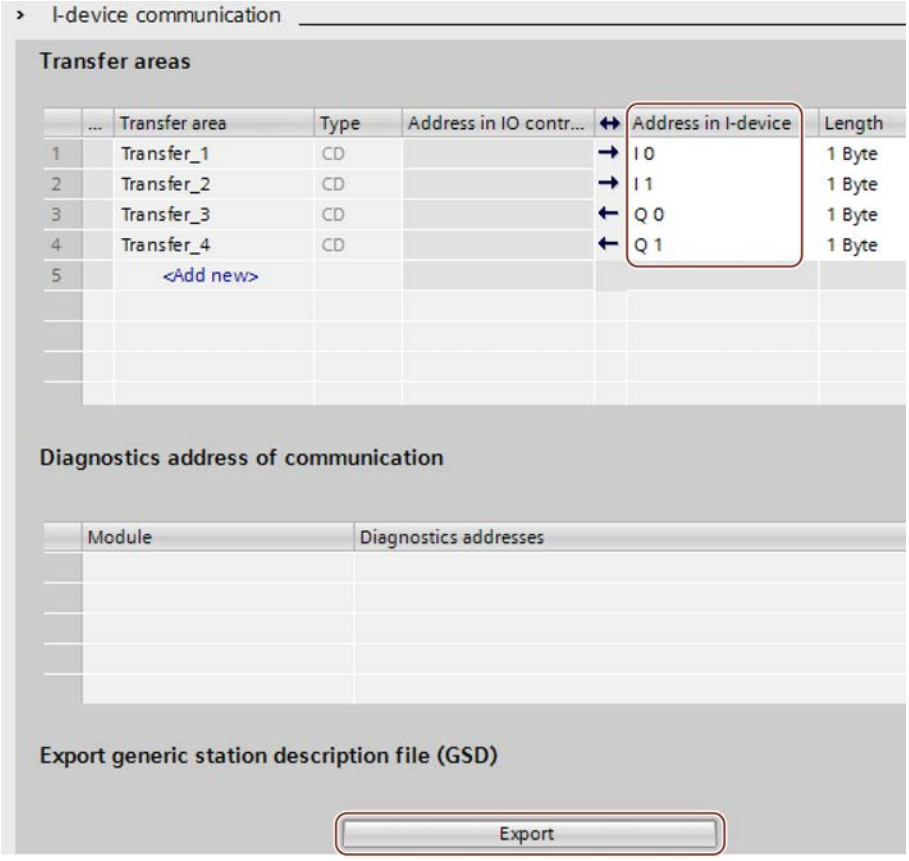
Manufacturer view

You configure an S7-1500 CPU as an I-device with centralized and distributed I/O with the desired transfer areas and with the number of IO controllers that can access this I-device (always greater than 1 for a Shared Device).

Special note: The I-device is configured without a higher-level IO controller. As a result, only the local I/O addresses of the transfer area are available (= "Address in the I-device") in order to create the user program for editing the addresses from the transfer area. The I-device that

has been completely configured except for the connection to the higher-level IO controller is loaded to the S7-1500 CPU.

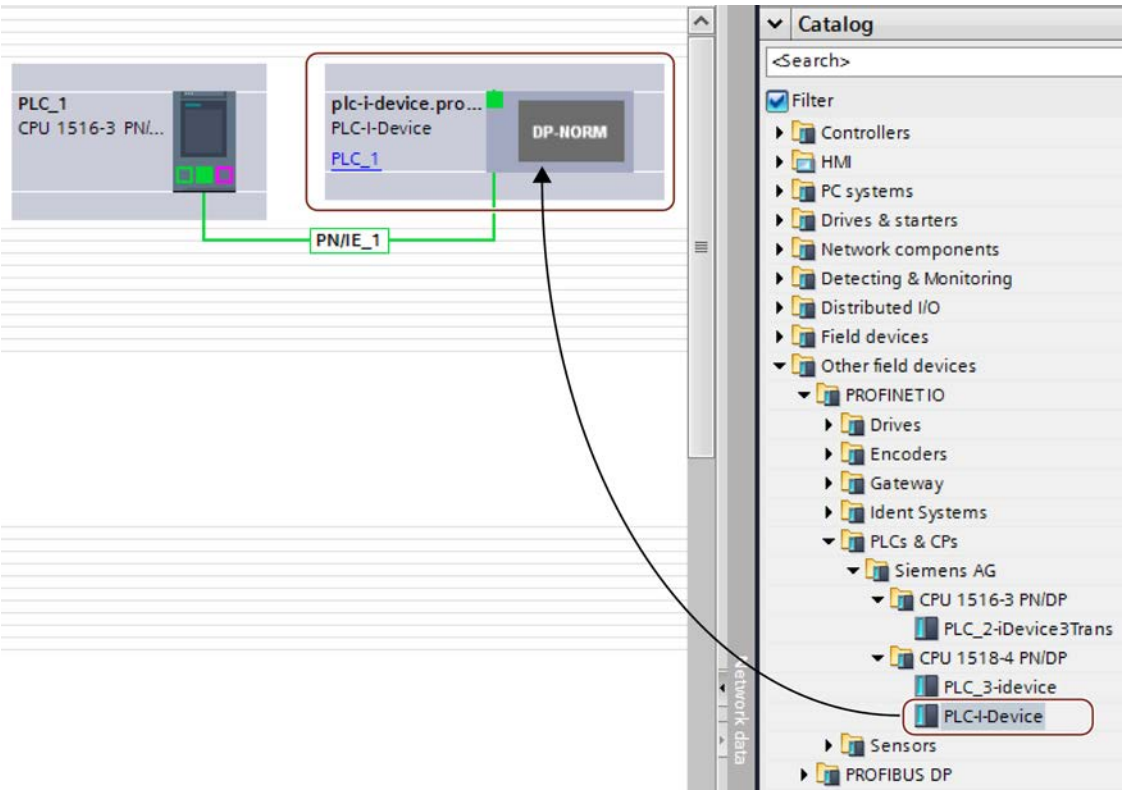
You export a PROFINET GSD file from the I-device configuration.



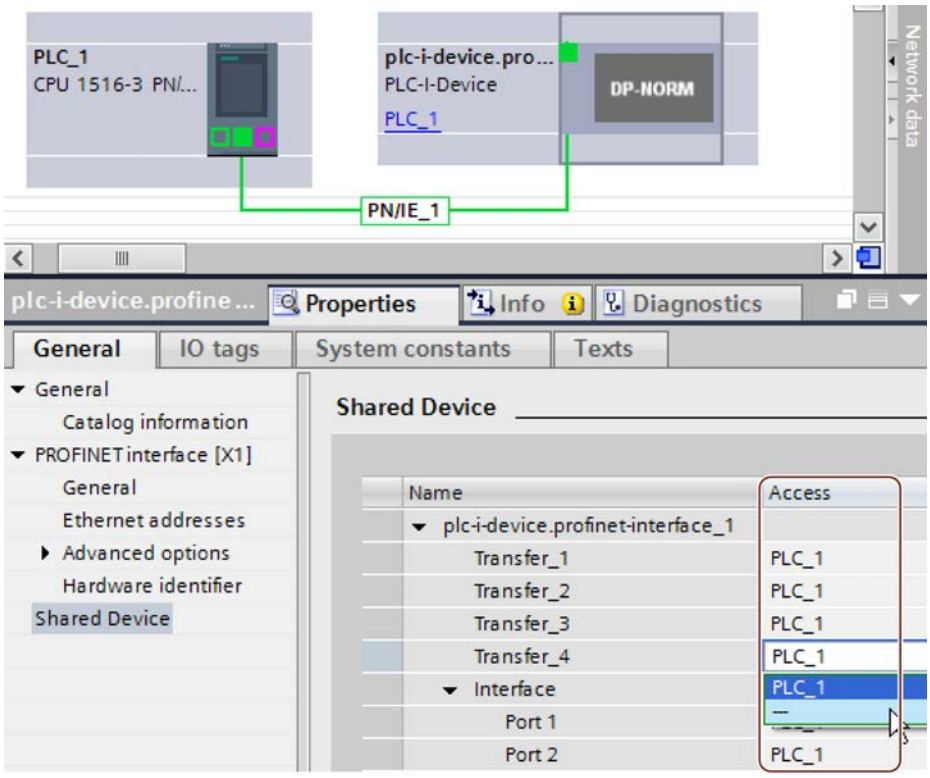
Operator view

You must install the PROFINET GSD file created from the I-device configuration in all engineering systems that are involved in configuring a PROFINET IO system with this shared I-device. If all uses of this I-device will be configured with STEP 7 V13, it is sufficient to install the GSD file in STEP 7.

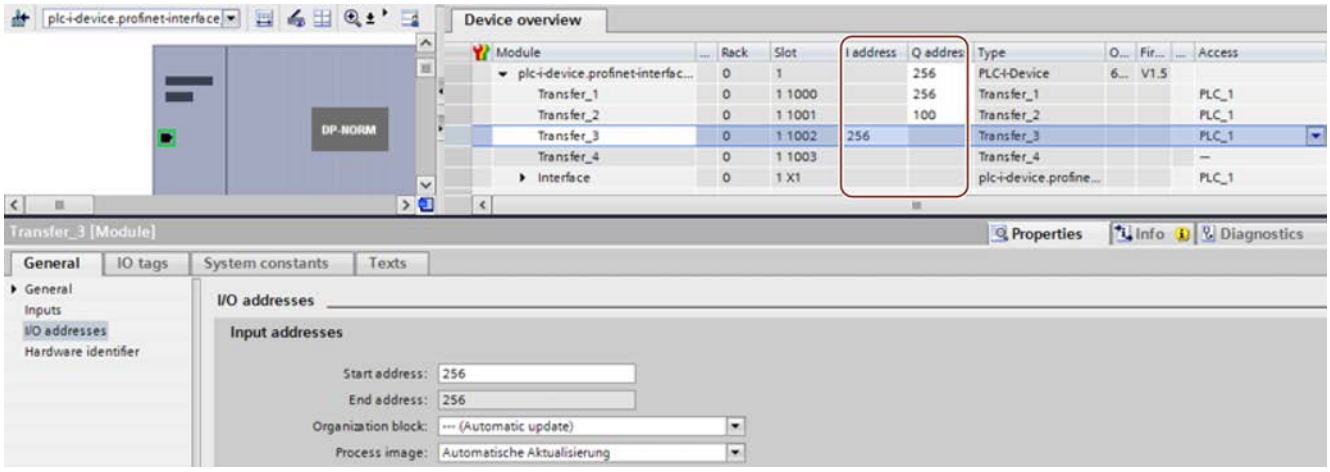
You configure the I-device as a GSD variant on the PROFINET IO system in the projects involved. In STEP 7 V13, this I-device can be found under "Other field devices > PROFINET IO > PLCs & CPs" following installation.



In each of the projects involved, you define which transfer areas are assigned exclusively to the higher-level IO controller (default setting: all). You set the other transfer areas to "---" (not assigned). As a result of this setting, the local IO controller has no access to this transfer area and it can therefore be assigned to another IO controller in another project.



You adapt the addresses from the view of the IO controller in the device overview. To open the device overview, double-click the I-device.



Requirement

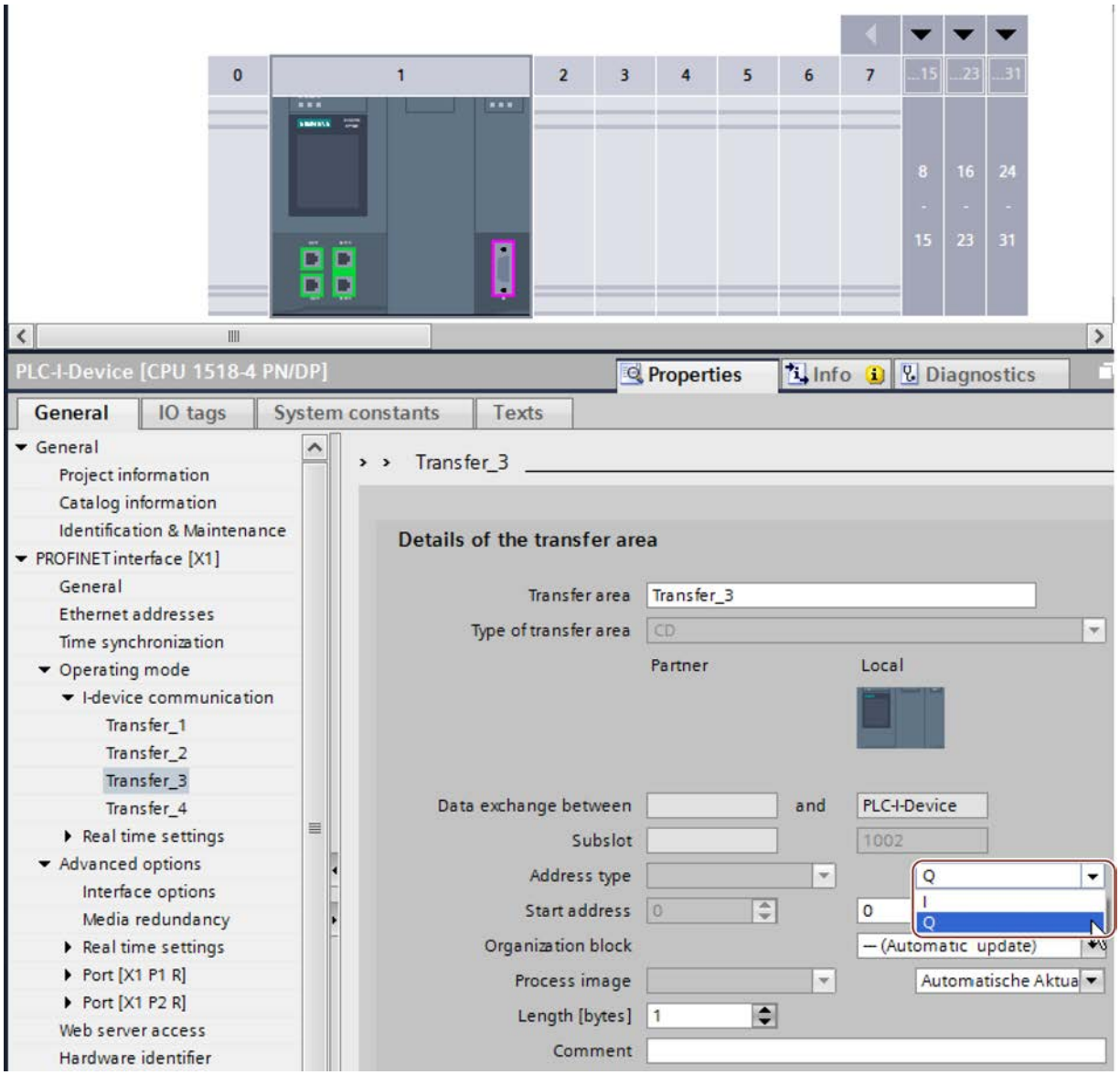
- STEP 7 V13 or higher
- In the context of STEP 7 V13: Only CPU 15XX FW 1.5 or higher can be configured as a shared I-device (no CPU 31X, CPU 41X, or CPU 12XX).
- In the context of other IO controllers that are to have access to a shared I-device: Corresponding engineering tool, for example, STEP 7 V5.5, Service Pack 3 for CPU 31x-3 PN/DP.

Procedure - Creating the PLC-I-device project

To create the project with a shared I-device, follow these steps:

1. Start STEP 7.
2. Create a new project with the name "PLC-I-device".
3. Insert a CPU 1518-4 PN/DP from the hardware catalog in the network view. Assign the name "PLC-I-device".
4. Double-click the IO device and configure all required modules and submodules.

5. Assign the module parameters.
   In particular, the following settings for the CPU are necessary in the area of the PROFINET interface [X1]:
   - Enable the "IO device" option in the "Operating mode" area.
   - Configure the transfer areas in the "Operating mode" > "I-device configuration" area.
     The "Address in IO controller" column remains empty because no IO controller is assigned.
     Note: To change an input area to an output area, and vice versa, you must navigate to the area of the corresponding transfer area.



   - Select the number of IO controllers that will access the shared I-device during operation ("Mode" > "Real time settings" area, "Shared Device" section).

6. Save the project.
7. Click the "Export" button ("Mode" > "I-device configuration" area, "Export generic station description file (GSD)" section).
   If you do not change the name in the Export dialog, the GSD file has a name in the form "GSDML-V2.31-#Siemens-PreConf_PLC-I-Device-20130925-123456", for example.

Procedure - Creating the PLC_1 project

To create the first project with a shared I-device, follow these steps:

1. Start STEP 7.
2. Install the PROFINET GSD file from the export of the I-device CPU (PLC-I-Device).
3. Create a new project with the name "PLC_1".
4. Insert a CPU 1516-3 PN/DP in the network view. The name of the CPU should be "PLC_1".
5. Insert the I-device from the hardware catalog (Hardware catalog: Other field devices > PROFINET IO > PLCs & CPs).
6. Assign the IO controller "PLC_1" to the I-device.
7. Select the "Shared Device" area in the properties of the I-device.
   In the table, all transfer areas and the PROFINET interface are assigned to the local IO controller (PLC_1).
8. Define the transfer areas to which the PLC_1 CPU should **not** have access. Select the "---" entry for these areas. These transfer areas are provided for PLC_2.
9. Save the project.

Procedure - Creating the PLC_2 project

To create the second project with a Shared Device, follow these steps:

1. Start STEP 7 once again.
   A new instance of STEP 7 opens.
2. In the new instance, create a new project with the name "PLC_2".
3. Insert a CPU 1516-3 PN/DP in the network view. Assign the name "PLC_2".
4. Insert the I-device from the hardware catalog (Hardware catalog: Other field devices > PROFINET IO > PLCs & CPs).
5. Assign the IO controller "PLC_2" to the I-device.
6. Adapt the access to the transfer areas as in the PLC_1 project. Ensure that no duplicate assignments result.
7. Adapt the parameters of the subnet and PROFINET interface. Because the shared I-device involves the same device in different projects, these data must match.
8. Save the project.

Both projects now have an identically configured shared I-device. The IO controller access and the parameters of the PROFINET interface should still be checked in the different projects during the next step.

Summary - Assigning parameters for access to the sShared Device

The transfer areas are automatically assigned to the local IO controller. To change the assignment, follow these steps:

1. Click the "PLC_I-Device" device in the network view of the "PLC_1" project, and select the "Shared Device" area.
2. A table shows which CPU has access to each of the configured transfer areas. The default setting is that the local CPU has access to all modules and submodules.
3. Keep the setting "PLC_1" for all transfer areas that are to remain in the address range of the local CPU
   Select the setting "---" for all transfer areas that are to be located in the address range of the "PLC_2" CPU from the "PLC_2" project. This means that an IO controller outside the project is to have access to the transfer area.
4. Follow the same procedure for the remaining projects.
5. Finally, check whether the settings for access are "complementary" for each module or submodule in both projects. This means that if the local CPU has access in one project, the option "---" must be set in the other project and vice versa.
   Special note: The option "---" for the PROFINET interface and therefore for the ports makes the associated parameters read-only and not changeable. Parameters of the PROFINET interface and port parameters can only be edited in the project in which the PROFINET interface is assigned to the local CPU. The ports can be interconnected in both projects regardless of this.
6. Check whether the same IP address parameters and device name are set for the Shared Device in all projects.
   Check whether the same S7 subnet ID is set in all projects for the subnet to which the Shared Device is connected (subnet properties, "General" area in the Inspector window).

Note

If you make changes to the I-device (e.g., change the number or length of the transfer areas):

Export the I-device as a GSD file again. Re-install the GSD file in each project that uses the I-device as a Shared Device. Make sure that only one IO controller has access to a transfer area.

Procedure - Adjusting the real-time settings

To ensure that all IO controllers and Shared Devices are operated with the appropriate send clock and that the update times are calculated correctly based on the communication load, you must adjust and check the following settings:

1. You must set the same send clock for each IO controller that has access to modules and submodules of the Shared Device:
   - If you configure the IO controller with STEP 7 (TIA Portal):
     Open the corresponding project.
     Select the PROFINET interface of the IO controller.
     Select the "Advanced options > Real time settings > IO communication" area in the Inspector window and set the shared send clock.
   - If you configure the IO controller with a different engineering tool:
     Select the PROFINET interface of the Shared Device in STEP 7 (TIA Portal) and read out the send clock on the Shared Device ("Advanced options > Real time settings" area)
     Enter the read send clock in the engineering tool.
   - Special note: If you configure **all** IO controllers that have access to the shared I-device in STEP 7 (TIA Portal or V5.5), you can set shorter send clocks on the IO controller than supported by the Shared Device (send clock adaptation).

Compiling and loading

You must compile the configurations for the different IO controllers and load them to the CPUs one after the other.

Due to the distributed configuration with separate projects, STEP 7 does not output consistency errors in the case of incorrect access parameter assignment. Example for incorrect access parameter assignment:

- Several IO controllers have access to the same module
- IP address parameters or send clocks are not identical

These errors do not show up until operation and are output as configuration errors, for example.

See also

Configuring the I device (Page 1495)

Useful information on Shared Devices (Page 1521)

Module-internal shared input/shared output (MSI/MSO)

Introduction

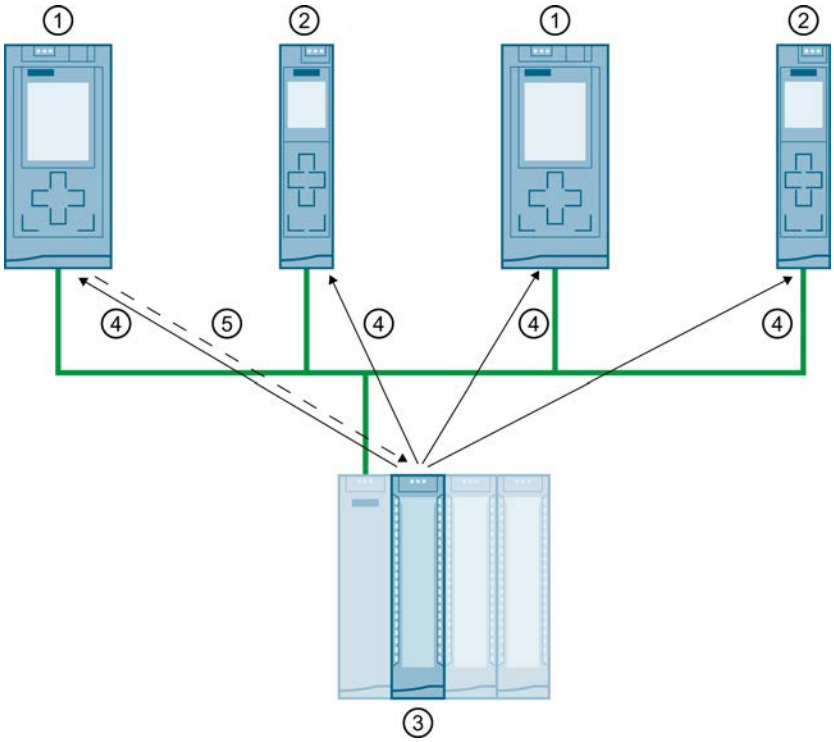
This section describes the module-internal shared input/shared output (MSI/MSO) functionality for the S7-1500 I/O modules that are operated in an ET 200M on PROFINET.

Module-internal shared input/shared output functionality

The module-internal shared input (MSI) function allows an input module to make its input data available to up to four IO controllers. Each controller has read access to the same channels.

The module-internal shared output (MSO) function allows an output module to make its output data available to up to four IO controllers. One IO controller has write access. The other IO controllers have read access to the same channels.

The following figure shows the MSI/MSO functionality.



① CPU 1516-3 PN/DP as IO controller
② CPU 1511-1 PN as IO controller
③ I/O module with MSI/MSO
④ Read access to the channels of the I/O module
⑤ Write access to the channels of the I/O module (only with MSO)

Advantages of MSI/MSO

Module-internal shared input/shared output (MSI/MSO) offers the following advantages:

- Real-time acquisition in multiple CPUs
- Lower costs due to saving on additional IO devices and modules
- Lower space requirements due to saving on additional IO devices and modules
- Reduced communication load because no CPU-CPU communication is needed
- No additional programming effort is needed for CPU-CPU communication

Requirements for the use of MSI/MSO

Observe the following requirements:

- MSI/MSO can only be used with PROFINET IO
- Configuration software: STEP 7 (TIA Portal) V12 SP1 or higher with GSD file.
- The IM 155-5 PN ST interface module and the modules support MSI/MSO firmware version V2.0.0 or higher.

Boundary conditions for the use of MSI/MSO

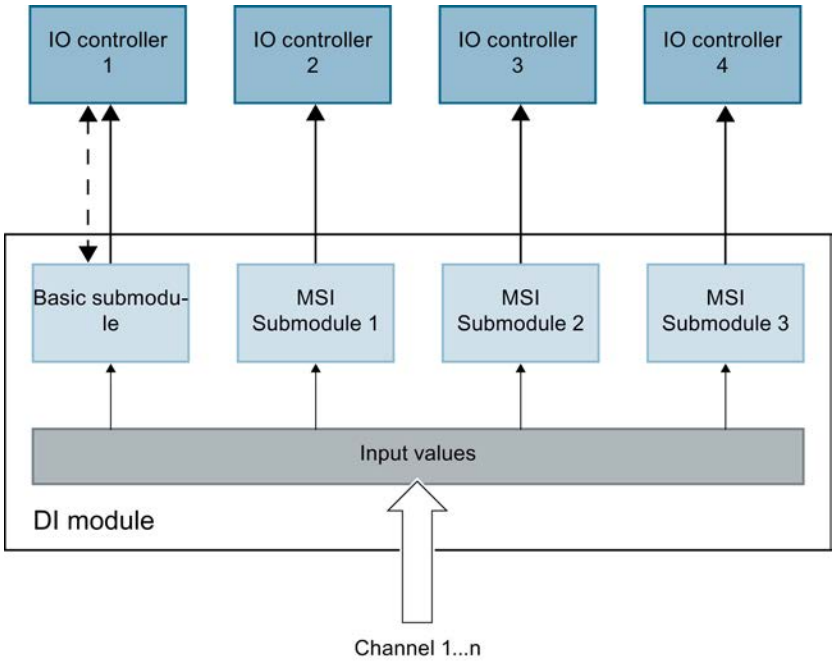
Note the following boundary conditions:

- MSI/MSO cannot be used for modules with channel groups.
- Modules with MSI/MSO cannot be operated in isochronous mode.

MSI submodules

The input values of all channels are copied to a basic submodule and up to three other MSI submodules during MSI configuration of an input module. The channels of the module are then available with identical input values in the basic submodule and the MSI submodules. The submodules can be assigned to up to four IO controllers when the module is used in a Shared Device. Each IO controller has read access to the same channels.

The following figure shows a digital input module with the basic submodule and three MSI submodules. Each submodule is assigned to an IO controller. Diagnostics and parameter assignment of the digital input module can be performed from the IO controller 1 via the basic submodule.



Value status of the MSI submodule (Quality Information, QI)

The meaning of the value status depends on the submodule to which it pertains.

For the basic submodule (first submodule), the value status 0 indicates that the value is incorrect.

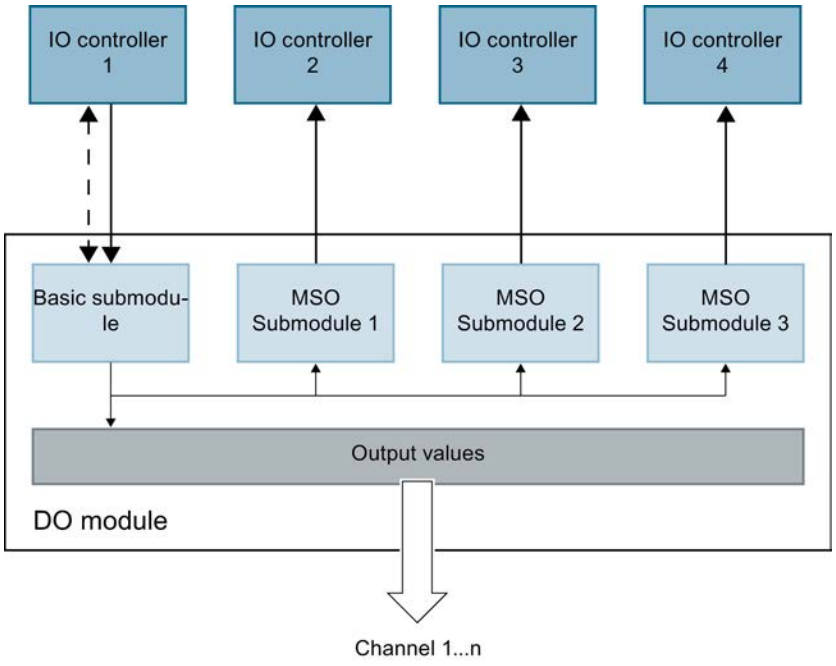
For an MSI submodule (submodules 2 to 4), the status value 0 indicates that the value is incorrect or the basic submodule parameters have not yet been assigned (not ready for operation).

MSO submodules

The output values of all channels of the module are copied from a basic submodule to up to three other MSO submodules during MSO configuration of an output module. The channels of the module are then available with identical values in the basic submodule and the MSO submodules. The MSO submodules can be assigned to up to four IO controllers when the module is used in a Shared Device:

- The IO controller to which the basic submodule is assigned has write access to the outputs of the module. The basic submodule therefore occupies output addresses in the process image of the IO controller.
- The IO controllers to which the MSO submodules are assigned have read access to the outputs of the module. MSO submodules therefore occupy input addresses in the process image of the IO controller.

The following figure shows a digital output module with the basic submodule and three MSO submodules. Each submodule is assigned to an IO controller. Diagnostics and parameter assignment of the digital output module can be performed from IO controller 1 via the basic submodule.



Value status of the MSO submodule (Quality Information, QI)

The meaning of the value status depends on the submodule to which it pertains.

For the basic submodule (first submodule), the value status 0 indicates that the value is incorrect.

For an MSO submodule (submodules 2 to 4), the status value 0 indicates that the value is incorrect or one of the following errors has occurred:

- The basic submodule parameters have not yet been assigned (not ready for operation).
- The connection between the IO controller and the basic submodule has been interrupted.
- The IO controller of the basic submodule is in STOP or POWER OFF state.

Configuring I/O modules with MSI/MSO submodules

Requirement

- Configuration software STEP 7 V12 Service Pack 1 or higher
- GSD file for I/O modules with MSI/MSO.
- IO device supports MSI/MSO (e.g., IM 155-5 PN ST firmware version V2.0.0 or higher)

Procedure

1. In the network view of STEP 7, insert an interface module as of IM 155-5 PN ST V2.0.
2. Double-click the IO device.
   You are now in the device view.
3. Place the modules with MSI/MSO from the hardware catalog in a suitable slot.
4. Configure the modules as described in the product information for the module.

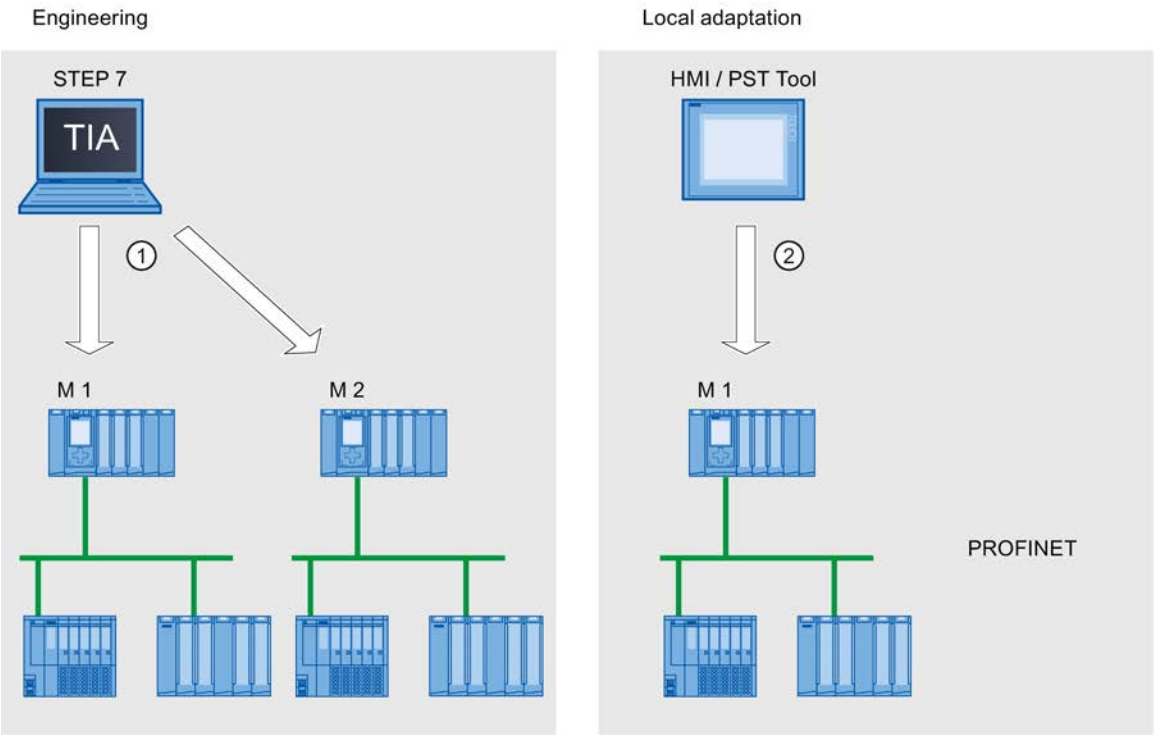
Creating a standard machine project

Important information on standard machine projects

Multiple use automation solutions

The following requirements apply to a flexibly usable automation solution such as for standard machines:

- The automation solution (machine) is used multiple times by a customer.
- The machine is used in different plants by various customers.
- A centrally maintained project (configuration and program) can be loaded to various machines of the same type without changes.
- Only a few easy adaptations are needed locally to integrate the machine into an existing network infrastructure.



① Load configuration with multiple use IO system

② Set IP address and device name locally on the IO controller

Principle

The automation components for a machine include a PROFINET IO system, consisting of an IO controller (PROFINET interface of a CPU) and the IO devices assigned to it.

With the "Multiple use IO system" setting for the IO system, you turn a STEP 7 project into a "Standard machine project".

The "Multiple use IO system" setting triggers various settings and checks of the configuration by STEP 7. The settings ensure that the IO system is self-contained and there are no dependencies on components outside the IO system.

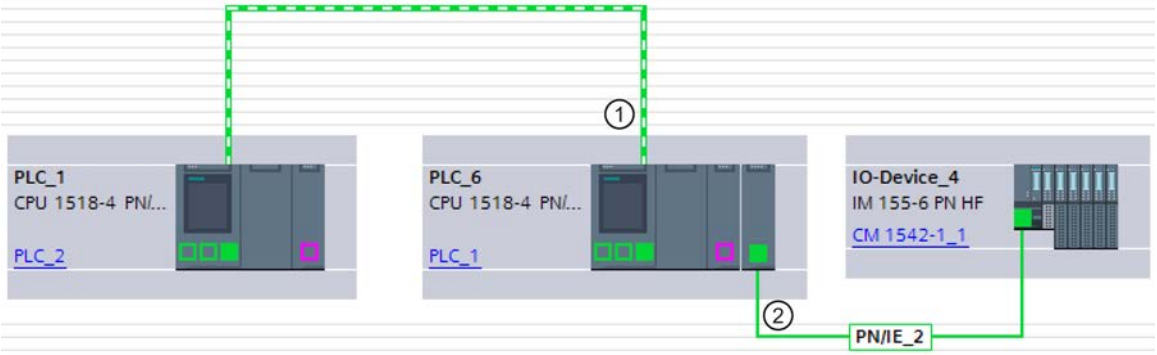
Requirement

- STEP 7 V13 or higher
- CPU 15XX firmware version V1.5 or higher as IO controller

Rules

The following rules apply to a multiple use IO system:

- No IO device can be configured as Shared Device.
- The ports of the devices must be interconnected.
  Devices for which no port interconnection is configured, for example, interface module IM 154-6 IWLAN (ET200pro PN), cannot be operated with STEP 7 V13 as IO devices on a multiple use IO system.
- If an IO device in a multiple use IO system is an I-device (CPU as IO device):
  - If the I-device has a lower-level IO system, this I-device cannot be connected to the same PROFINET interface as the higher-level IO controller. A PROFINET interface on the I-device can therefore be operated either as "IO controller" only or as "IO device" only.
    Note: If the I-device is configured using a PROFINET GSD, STEP 7 cannot check for compliance with this rule. In this case, you are responsible for ensuring compliance with the rule.



① I-device in multiple use IO system. The PROFINET interface is operated as IO device. No other IO system is connected here.

② A lower-level IO system on the I-device is connected to another PROFINET interface.

  - The PROFINET interface of the I-device must be set to “Parameter assignment of PN interface by higher-level controller”.
- If MRP (Media Redundancy Protocol) is configured:
  - All IO devices on multiple use IO systems must belong to the same MRP domain.
- If IRT (Isochronous Real Time) is configured:
  - All IO devices in multiple use IO systems must belong to the same sync domain.
  - The sync domain must not include any other IO devices.
- IE/PB links cannot be operated as IO device with STEP 7 V13 in a multiple use IO system.

Configuration

You specify whether or not a configuration can be used multiple times in the properties of the IO system.

All other parameter settings for the configured devices are then set automatically by STEP 7 and checked during compilation.

Boundary conditions

To prevent a standard machine project from having dependencies on other devices outside of the machine, observe the following:

- Configure only one CPU as IO controller and the corresponding IO devices in the standard machine project.
- Do not use two-ended connections for the communication. Instead, use only one-ended connections or unspecified connections if necessary.
- Use Open User Communication for the communication and make sure that the address parameters can be easily adapted on site.

See also

Unspecified connections (Page 596)

Basic information on communication via the PUT/GET instruction (Page 700)

Specifying an unspecified connection (Page 623)

Creating a new connection - overview (Page 600)

Configuring a standard machine project

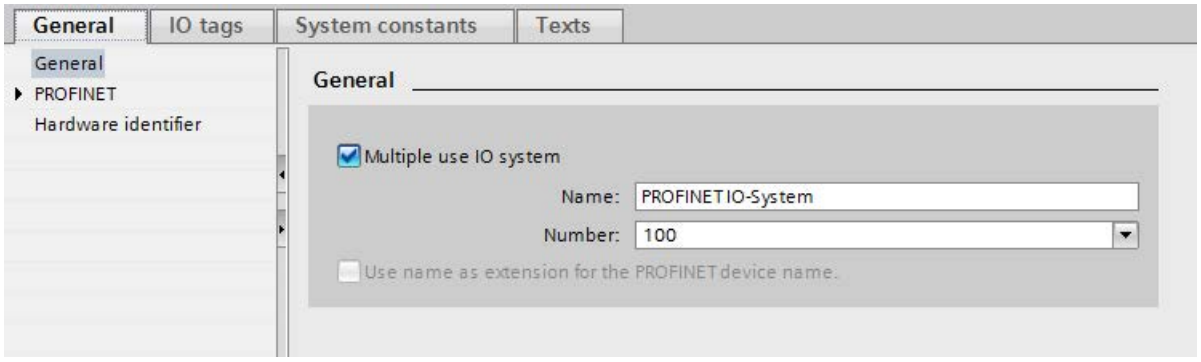
Requirement

- CPU S7-1500 firmware version V1.5 or higher as IO controller

Procedure

To create a standard machine project, follow these steps:

1. Create a project.
2. Configure an S7-1500 CPU firmware version V1.5 or higher as IO controller.
3. Configure the required IO devices and assign the IO devices to the IO controller.
4. Configure the port interconnection between the devices.
5. Select the IO system so that you can edit the properties in the inspector window.
6. Select the "Multiple use IO system" check box in the "General" area of the inspector window.



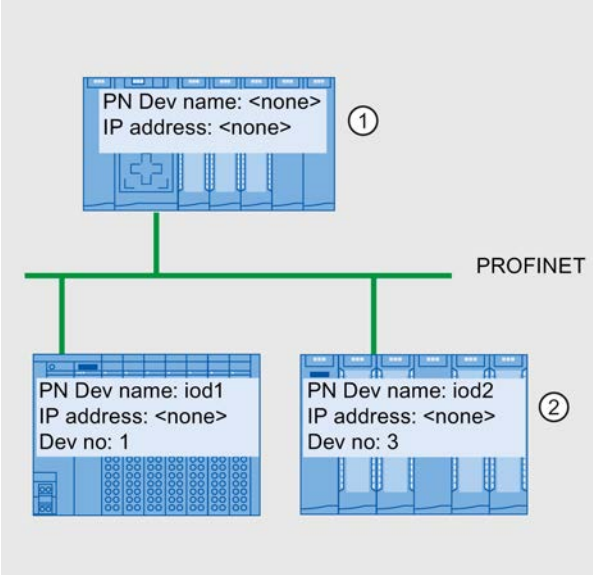
Result: The following settings are made by STEP 7:

- The device name of the IO controller (CPU) in the standard machine project is set to "Permit setting of PROFINET device name directly on the device". The IO controller (CPU) has no PROFINET device name initially.
- The IP protocol of the IO controller (CPU) is permanently set to "IP address is set directly at the device". The CPU has no IP address initially.
- The "Support device replacement without exchangeable medium" option is selected automatically. This option enables automatic commissioning. A commissioning engineer does not have to assign device names to the IO devices locally. The IO controller assigns the device name and IP address to the IO devices based on the preset topology and other settings during startup.
- The device name of the IO devices is set to "Generate PROFINET device name automatically" (from the configured name of the IO device).
- The IP protocol of the IO devices is set to "IP address is set by the IO controller during runtime". The IO devices have no IP address initially. If an IO device is not a typical distributed I/O system (e.g., ET 200 systems), but rather another device such as an HMI device, change the option to "IP address is set directly at the device"; see below.
- The device number for the IO devices is automatically assigned and is used locally for making the IP address unique.

You must enable the "Permit overwriting of PROFINET device name" option (CPU parameters, properties of the PROFINET interface, Ethernet addresses area) so that the IO controller can adapt the device name later at the operator.

This option is disabled by default.

The following figure shows the above-described settings for the IP address and PROFINET device name.



① After the configuration is loaded from the standard machine project, the IO controller has no device name and no IP address.

② Following loading, the IO devices have a device name and a device number but no IP address.

How an IO device obtains an IP address locally

Below, you will find an explanation of the "IP address is set by the IO controller during runtime" and "IP address is set directly at the device" options, which can generally be configured for a multiple use IO system.

If you have set the "Multiple use IO system" option for the IO system, STEP 7 automatically sets the "IP address is set by the IO controller during runtime" option for the IO devices.

In this case, the IO controller assigns the IO device an IP address that results from the locally assigned IP address of the IO controller (see next section). This option is appropriate if the IO device is a field device, e.g., ET 200MP, ET 200SP, or another distributed I/O system.

If the IO device is not a "standard" field device, for example, an HMI device for a Windows operating system, the "IP address is set by the IO controller during runtime" option described above does not work. In this case, choose the "IP address is set directly at the device" option. You must then assign the IP address to the device locally and take steps to ensure that this address is suitable for the IP addresses of the other IO devices and the IP address of the IO controller.

See also

Adapting a standard machine locally (Page 1545)

Adapting a standard machine locally

A few steps are needed to adapt the machine that was loaded with the standard machine project.

Only the device name and IP address of the IO controller must be adapted locally. The device names and IP addresses of the IO devices result from these adaptations. In this example, the effects of local settings are described for two specific machine modules.

The local settings are possible, for example, with the CPU display and commissioning tools such as Primary Setup Tool (PST) or PRONETA. You do not need a programming device with STEP 7 to make these settings, even though it is possible to do so.

Requirement

- The machine was loaded with a standard machine project (see Configuring a standard machine project (Page 1543)).
- The display is ready for operation and the desired tool for assigning the IP address and device name is available (e.g., PST Tool, STEP 7).
- The ports of the IO controller and IO devices are interconnected according to the configuration.

Procedure

Observe the boundary conditions and instructions for commissioning an S7-1500. Refer to "See also" for a reference to the S7-1500 system manual in which commissioning is described.

To adapt a standard machine locally, follow these steps:

1. Integrate the machine into the network.
2. Connect the device for assigning the IP address and device name to the CPU.
3. Assign the desired device name and IP address to the IO controller.

The IO controller (CPU) then assigns the adapted PROFINET device name and a unique IP address to the IO devices.

The following rules apply to the assignment:

- The device names of the IO devices are formed by chaining together the following name components, separated by a period:
  <configured name of the IO device from the standard machine project>.<name of the associated IO controller set on the device>
- The IP addresses of the IO devices result from the locally configured IP address of the associated IO controller and the device number (sum).

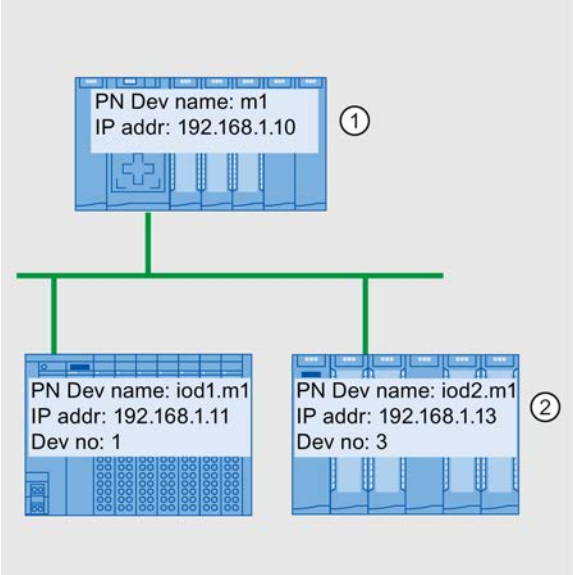
Note

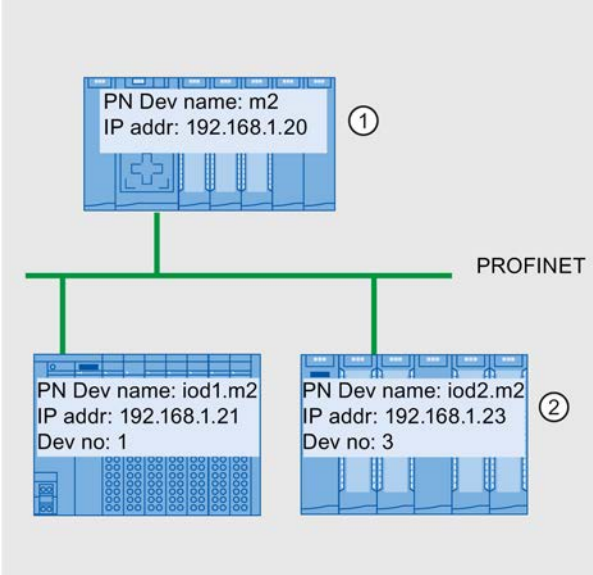
Make sure that duplicate IP addresses cannot be created on the subnet during the assignment. The IO controller does not assign a new IP address in this case.

In the following figure, the device name "m1" and the IP address 192.168.1.10 have been assigned to the IO controller of the first machine.

The device name "m2" and the IP address 192.168.1.20 has been assigned for the second machine.

Refer to the figure for the resulting device names and IP addresses.





① Set device name and IP address on the IO controller

② After startup, the IO devices have an updated device name (<configured device name>.<device name of IO controller>) and an adapted IP address (= <IP address of IO controller> + <device number>)

See also

Configuring a standard machine project (Page 1543)

S7-1500 system manual (http://support.automation.siemens.com/WW/view/en/59191792)

Permit overwriting of PROFINET device name

With S7-1500 CPUs firmware version V1.5 and higher you can overwrite PROFINET device names of IO devices during startup of the CPU. This option reduces the effort during automatic commissioning, for example, when replacing a device.

Effect of "Permit overwriting of PROFINET device name" option

The IO controller (CPU) can overwrite the PROFINET device names of IO devices in the IO system when the "Permit overwriting of PROFINET device name" option is enabled.

Multiple use IO systems can only be operated when this option is enabled. The IO controller checks prior to overwriting if the type of the IO device matches the configured type.

WARNING

When the "Permit overwriting of PROFINET device name" option is selected, incorrectly connected devices can receive an incorrect PROFINET device name from the configuration.

Depending on the connected I/O, there is risk of death, severe injuries or damage caused by malfunctions.

To rule out any danger, check whether the suitable replacement device has been connected in case of a device replacement and whether the port interconnection matches the configured preset topology.

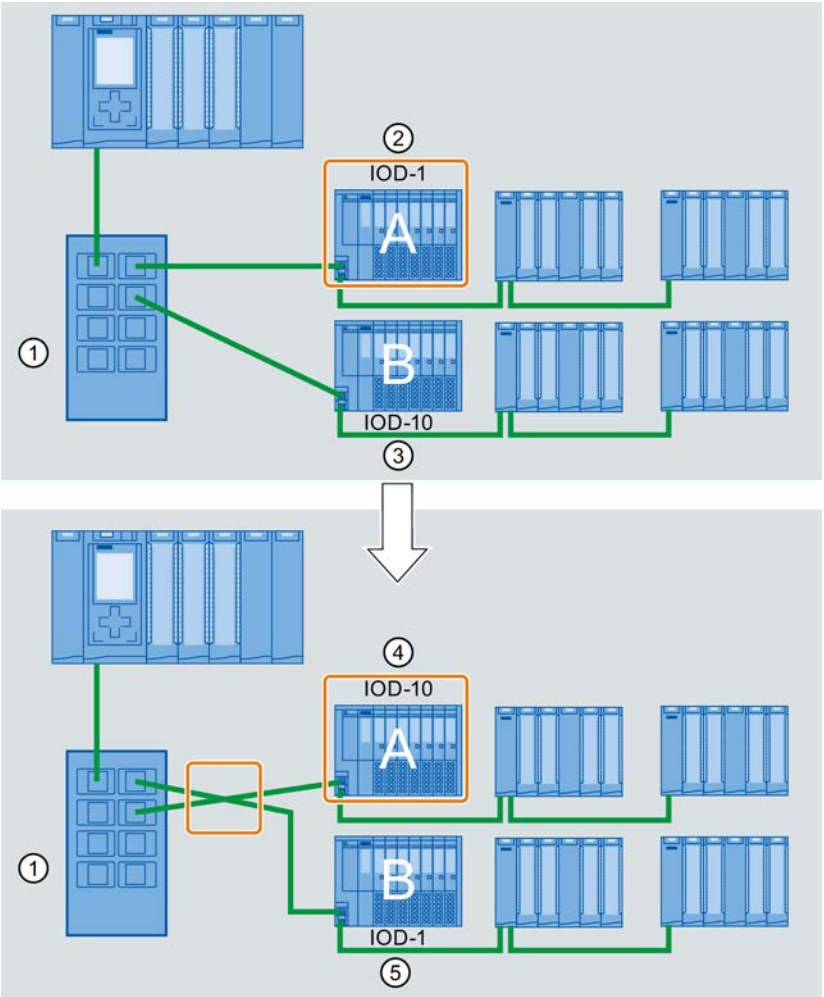
If the option is not selected, the IO controller cannot overwrite the device names of the IO devices. In this case, you must manually assign the PROFINET device name on the IO device if the PROFINET device name changes in the configuration or when a device is replaced, or delete the device names of the IO devices prior to an automatic commissioning.

Typical source of danger

When replacing an IO device ("standard case"), it is almost guaranteed that the replaced device will be connected according to the configured port interconnection.

The figure below shows a scenario whereby the connections of two identical PROFINET cables are swapped at two switch ports. Because the IO controller assigns the device names according to the preset topology, the incorrect connection of the devices has serious effects on the naming.

Due to the control of different actuators, the plant could become hazardous in this case.



① Switch with connected PROFINET cables
② Device A, device name "IOD-1": controls motor 1
③ Device B, device name "IOD-10": controls motor 10
④ Device A controls motor 10!
⑤ Device B controls motor 1!

Procedure

To change the "Permit overwriting of PROFINET device name" option, follow these steps:

1. Select the PROFINET interface of the CPU for which you want to change the option in the network view or in the device view.
2. Select the area "Advanced options", section "Interface options".
3. Change the option.

See also

Enabling device replacement without exchangeable medium (Page 1461)

8.1.4.13 Bus coupling with PN/PN coupler

Application and function

Application

The PN/PN coupler is used to link two Ethernet subnets with one another and to exchange data. That way use data about input or output address areas or datasets can be used. The maximum size of the transferable input and output data is 1024 bytes. The division into input and output data is preferable, so that e.g. 800 byte input data and 200 byte output data can be configured.

As a device, the PN/PN coupler has two PROFINET interfaces, each of which is linked to one subnet.

In the configuration, two IO Devices are produced from this one PN/PN coupler which means that there is one IO Device for each station with its own subnet. The other part of PN/PN coupler in each case is known as the bus node. Once configuring is complete, the two parts are joined.

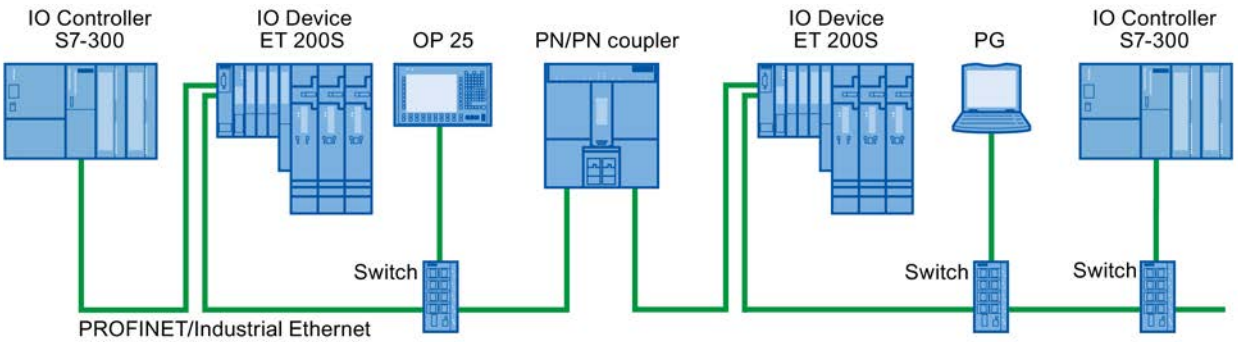


Figure 8-51 Coupling two PROFINET IO subnets with one PN/PN coupler

Additional information

For additional information on "PN/PN couplers", refer to Service & Support on the Internet (http://support.automation.siemens.com/WW/view/en/44319532).

Linking Ethernet subnets

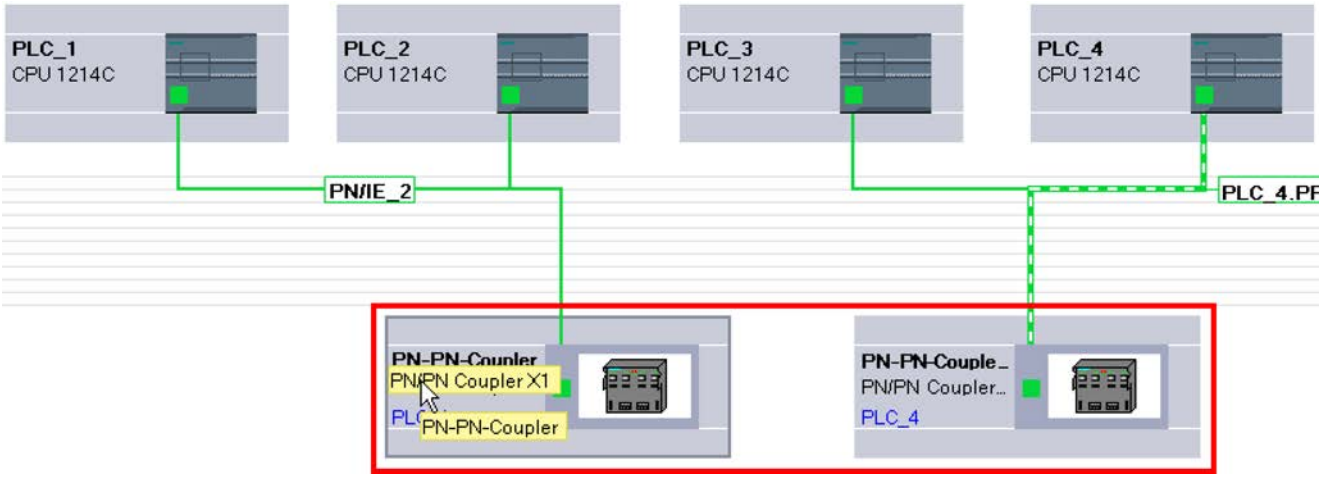
Linking Ethernet subnets with a PN/PN coupler

You can link Ethernet subnets with the standard device PN/PN coupler.

To link Ethernet subnets, follow these steps:

1. Create your Ethernet subnets.
2. Select the standard field devices in the hardware catalog. Find the PN/PN coupler as head module in the "PROFINET IO" folder.

3. In the network view, drag the two components X1 and X2 to the required version of the PN/PN coupler per drag-and-drop operation. The components form a device, but are shown separately to make handling easier.
4. Connect the Ethernet interface of the PN/PN coupler X1 to the first Ethernet subnet.
5. Connect the Ethernet interface of the PN/PN coupler X2 to the second Ethernet subnet. The Ethernet subnets are now linked through the two components of the PN/PN coupler.



8.1.4.14 Integrating external tools

Integrating S7-external tools

Introduction

Tools external to STEP 7 ("Device Tools") with a special call interface (Tool Calling Interface) can be used to configure distributed devices. Such devices are also referred to as "TCI capable".

The performance range of these tools exceeds the possibilities provided within GSD configuration, for example, they can provide expanded graphical input options.

Distributed devices can be as follows:

- PROFIBUS DP slaves
- Modules within a DP slave
- PROFINET IO devices
- Modules within an IO device

Note

Warranty and liability

Siemens accepts no liability for third-party software (device tools) called with the TCI (Tool Calling Interface) or for proper interaction with the associated devices.

Requirement

The call interface of the tool complies with the TCI specification. Parameters and commands are forwarded to the distributed device via this call interface.

Such tools have to be installed using a setup provided by the manufacturer. The "S7-PCT" (Port Configuration Tool) device tool for IO-Link master modules and IO-Link devices is an exception; this is supplied with STEP 7. Special note: After the installation, the tool is not shown in the list of installed software or in the list of software products in the project.

The GSD file of the distributed device that is to be configured with the Device Tool must be installed.

Starting the device tool

The command for starting the device tool is available in the shortcut menu of the TCI-capable device in the shortcut menu of the graphical and tabular device view: "Start device tool".

See also

Starting the SIMATIC S7-PCT (Page 1552)

Starting the SIMATIC S7-PCT

Introduction

The "S7-PCT" (Port Configuration Tool) device tool is installed with STEP 7.

The tool is used to assign parameters to the ports of IO-Link modules such as 4SI IO-Link (S7-1200, ET 200S) or 4IOL+8DI+4DO (ET 200eco PN).

Requirement

You have configured the corresponding CPU, the DP slave or the IO device with an IO-Link module.

Procedure

To start via the graphical device view, follow these steps:

1. Select the IO-Link module in the device view.
2. Select "Start device tool" from the shortcut menu.

OR, to start with the tabular device view, follow these steps:

1. Select the IO-Link module in the device view.
2. Arrange the areas in the work area in such a way that the tabular device overview is visible (it is located between the device view and the inspector window).
3. Select the row with the IO-Link module in the device overview.
4. Select "Start device tool" from the shortcut menu.

Result

The tool starts and you can configure the ports.

See also

Integrating S7-external tools (Page 1551)

8.1.4.15 Loading a configuration

Introduction to loading a configuration

In order to commission a device, identical configurations must be stored on the programming device/PC and on the connected devices. To synchronize the configurations on the programming device/PC and the connected devices, you load a configuration. Configuration data can be loaded in two directions:

- From the programming device/PC to a device
- From the device to a PG/PC

See also

Uploading project data from a device (Page 360)

General information on loading (Page 355)

Downloading a configuration to a device (Page 1553)

Downloading project data to a device (Page 357)

General information on loading to PG/PC (Page 1554)

Special features during startup (Page 1568)

Downloading a configuration to a device

Loading the hardware configuration

After you have inserted a new device in the project and configured it or if you have modified an existing hardware configuration, the next step is to load the current configuration to the device. This ensures that the same configuration is set on the programming device/PC and on the module that is physically present. Use the "Online > Download to device (Page 357)" menu command for this.

In the first download, the complete hardware configuration is downloaded. In subsequent downloads, only changes to the hardware configuration are downloaded.

You have the following options for loading the hardware configuration:

- Loading in the device or network view
- Loading in the project tree
- Loading to an accessible device

> **WARNING**
>
> **Perform load operation only in STOP mode**
>
> Following loading, the machine or process may behave unexpectedly if the parameter assignment is incorrect. A CPU must be set to STOP mode for the load operation to rule out possible damage to equipment or personal injury.

Special considerations for loading isochronous applications

Isochronous applications consist of a hardware configuration part and a software part.

Example: If you change the number of an IO system, the delay time, or the assignment of a process image partition of the isochronous I/O in the hardware configuration, this affects the parameters of the isochronous mode interrupt OB and thus also the software part.

With isochronous applications, you should always load the complete project (hardware and software). With partial loading (loading hardware and software separately at different times), inconsistencies can arise that, for example, can prevent CPU startup or isochronous operation of the application.

See also

General information on loading (Page 355)

Loading a configuration to PG/PC

General information on loading to PG/PC

Introduction

If you bring your PG/PC to a plant and the STEP 7 project used to create the configuration of this plant is not available, load the configuration to a new project on your PG/PC, for example. Use the "Online > Loading the device as new station (hardware and software) (Page 360)" menu command for this.

The list of accessible devices in the project tree is always used when loading a device to your programming device. You can select multiple devices and load them to the project at the same time.

Requirements

- The hardware configuration in the device must be created in TIA Portal V12 or higher. A hardware configuration present in the device that was created with an older version cannot be loaded and must be upgraded (Page 335).
- Modules present in the device from GSD (ML), HSPs, or service packs must be installed in the TIA Portal on the PG/PC.

- A project must be open. This project can be a new (empty) project or an existing project.
- The opened project is in offline mode.

Scope of load operation

The following list shows an overview of the loadable components of a configuration:

- The device (e.g., a CPU) with all I/O modules and all parameter settings
- PROFIBUS master systems and all PROFIBUS-relevant settings
- PROFINET IO systems and all PROFINET-relevant settings
- I devices and I slaves
- Settings for direct data exchange

After loading a CPU, all other modules within the address area of the CPU are also loaded.

The following connections are also loaded when the configuration is loaded:

- S7 connections (including routed connections) in combined PB/IE networks, including via IE-CP or PB-CM interfaces. S7 connections are automatically accepted as configured at one end when a device configuration is loaded, even if the S7 connection was configured at both ends in the original project. When both connection partners are loaded, the connection is joined together again during the next compilation.
- TCP connections via CPU-internal Ethernet interface, UDP/ISOonTCP connections, and TCP, UDP, ISO, and ISOonTCP connections via IE-CP interface
- Connections via the OUC connection parameter assignment for projects from STEP 7 V13 and higher

Note

The hardware configuration loaded to PG/PC is not completely identical to the configuration originally loaded to the device. Note the additional information on loading, in particular with regard to partially loaded configuration data in the case of cross-device communication.

Loading PC systems, such as WinAC or PC-based automation, is not possible.

See also

General information on loading (Page 355)

Loading specific device configurations

Information on loading

When device configurations are loaded to the PG/PC, all assigned parameters are transferred from the device to the project. If the CPU is connected to a subnet, all parameters of the device are loaded and the CPU is displayed in the network view as networked.

Note

CPUs reset to their factory settings do not have any hardware configuration. Therefore, nothing is loaded in this case after "Load to PG/PC" is selected in the "Online" menu.

Loading S7-300/400 configurations

To avoid conflicts when loading a device to an existing project, the following rules must be followed:

- Device names for CPUs, PROFIBUS slaves (DP slaves, I-slaves) and PROFINET devices (IO devices, I devices) must be unique
- The combination of network name, subnet ID, and IP/DP address for modules must be unique

If conflicts occur, the load operation is canceled and an alarm provides information about the problems that occurred. You can then adapt the project correspondingly or install missing components and then repeat the load operation.

Alarm configurations are not loaded to PG/PC.

Loading S7-1200/1500 configurations

Take the following into consideration when loading:

- CPUs of the S7-1200 series with firmware version V1.0 are not supported when loading.
- Device-specific diagnostic interrupts of the S7-1200 are not supported when loading. For device-specific diagnostic interrupts of the S7-1200 to be generated on the PG/PC once again, the hardware configuration must be compiled once again.
- Module comments of the S7-1200/1500 are loaded from the device to the PG/PC if the same project language is set that was used during loading to the device. You can deselect the loading of comments, if desired.

Note

All types of PC systems, such as WinAC, Embedded Controller, CP 1616 or PC CPs, do not support loading to PG/PC.

Loading distributed I/O

The following functionalities and settings of the distributed I/O are loaded:

- DP master systems/IO systems with associated DP masters/IO controllers (CPUs and CPs), DP slaves/IO devices, the utilized modules and their parameter assignment and properties, such as options handling, status bytes, or SYNC/FREEZE
- Connection of process image partitions (PIP) to organization blocks (OB). Applies to the module and OB properties
- Configured hardware interrupts with the associated properties
- DP master systems with I-slave
- CPs as PROFIBUS I-slave or PROFINET I-device
- Direct data exchange

Master-slave relationships between the I-slave/I-device and assigned DP master/IO controller are only established in the project if both the master as well as the I-slave are loaded to the PG. It does not matter whether you load the DP master/IO controller or the I-slave/I-device first. As soon as both devices are loaded, the master-slave/controller-device relationships will also be re-established.

Loading subnets and devices with MPI, PROFIBUS, Ethernet, and PtP

The following special considerations apply to loading subnets and end points of connections for MPI, PROFIBUS, Ethernet, and PtP with their respective connection properties:

- If a device with PROFIBUS interface is loaded, the bus parameters of the device will initially be different from the settings in the original project. The bus parameters will only match those of the original project after all devices involved are loaded and no additional devices are on the same bus.
- Passive communication devices that are not connected as DP slaves or IO devices to a corresponding master system or IO system do not participate in the data exchange. They are therefore not loaded.
- All devices involved should therefore be loaded for cross-device configurations. A warning is output during compiling of the project for missing network stations. Missing routing information due to communication stations that are not loaded is displayed as warning during compilation. If the configuration is loaded from the PG/PC back to the device, different routing information results.

When you compile the project after loading devices to the PG/PC, STEP 7 checks if all devices to which communication relationships have been configured are available. If devices are missing, you receive an alarm with the number of missing communication stations.

NOTICE
Cross-device communication When you load a configuration with cross-device communication to the PG/PC, you must also load all corresponding network stations to the PG/PC. If required network stations are missing and the configuration is loaded back to the device, there is no guarantee that cross-device communication is working again.

See also

Upgrading projects (Page 335)

Loading configurations with web server

Information on loading

The hardware configuration of a CPU also contains the web server settings. A number of restrictions apply to loading a web server configuration to the PG/PC:

- The assignment of the web server language and project language is not loaded for S7-300/400. The project texts are not loaded and an alarm is output that no project languages are assigned. The languages assigned in STEP 7 are loaded without restriction for S7-1200/1500 CPU.
- User administration data of the S7-1200/1500 can be loaded but not edited. You can use a check box to select whether you want to use the existing data as read-only data or discard these data and enter new data.
- Watch tables of the web server are not loaded.

The source files of the user-defined web pages (HTML pages, Java scripts, etc.) are not loaded. The program blocks generated during loading can only be edited if you enter the properties and the HTML page yourself.

See also

Information about the web server (Page 942)

Information about the web server (Page 903)

Loading configurations with PROFIBUS

Information on loading

A DP master is loaded to PG/PC. The DP master system and all connected DP slaves are inserted into the project. The respective settings remain unchanged. If a suitable PROFIBUS subnet has already been created, the loaded devices with PROFIBUS interface are connected to the existing subnet.

As a prerequisite for loading DP master systems with standard slaves, the corresponding GSD files must be installed in the TIA Portal and available in the hardware catalog. If a required

GSD file is not available in the same version as in the device, differences are identified during the consistency check.

Note

Direct data exchange in a configuration is only loaded if all communication partners involved in the direct data exchange are loaded to the PG/PC. Missing communication partners are represented by a placeholder device. This means the configuration with direct data exchange can still be compiled.

Isochronous mode

Note the following when loading DP master systems with activated PROFIBUS functionality "Isochronous mode":

- Bus parameters and the settings for isochronous mode are only identical after the device is loaded to the PG/PC if all devices relevant for calculation of isochronous mode are loaded.
- Only mono-master systems with isochronous mode are supported. Therefore, only configurations with just one DP master on the PROFIBUS subnet are loaded.

Loading configurations with PROFINET

Information on loading

If you have selected a CPU in the list of accessible devices and perform an load operation to PG/PC, all IO controllers and IO devices associated with this device along with their IO systems will be loaded. Settings for the topology are also transferred. If there is already a suitable Ethernet network in the project, the loaded devices are integrated into the existing network.

Relationships between the IO controllers and IO devices are only established within the project if both the IO controller as well as the I-device are loaded to the PG. It does not matter whether you load the IO controller or the I-devices first.

Supported functions

The following functionalities and settings are loaded:

- PROFINET configurations (RT and IRT) in IO systems with the associated IO controllers (CPUs and CPs), IO devices, and the modules used
- Logical addresses and interface properties
- Port interconnections
- Isochronous mode
- Sync domains/MRP domains
- Redundancy role "Client" or "Manager" for MRP configurations

Note

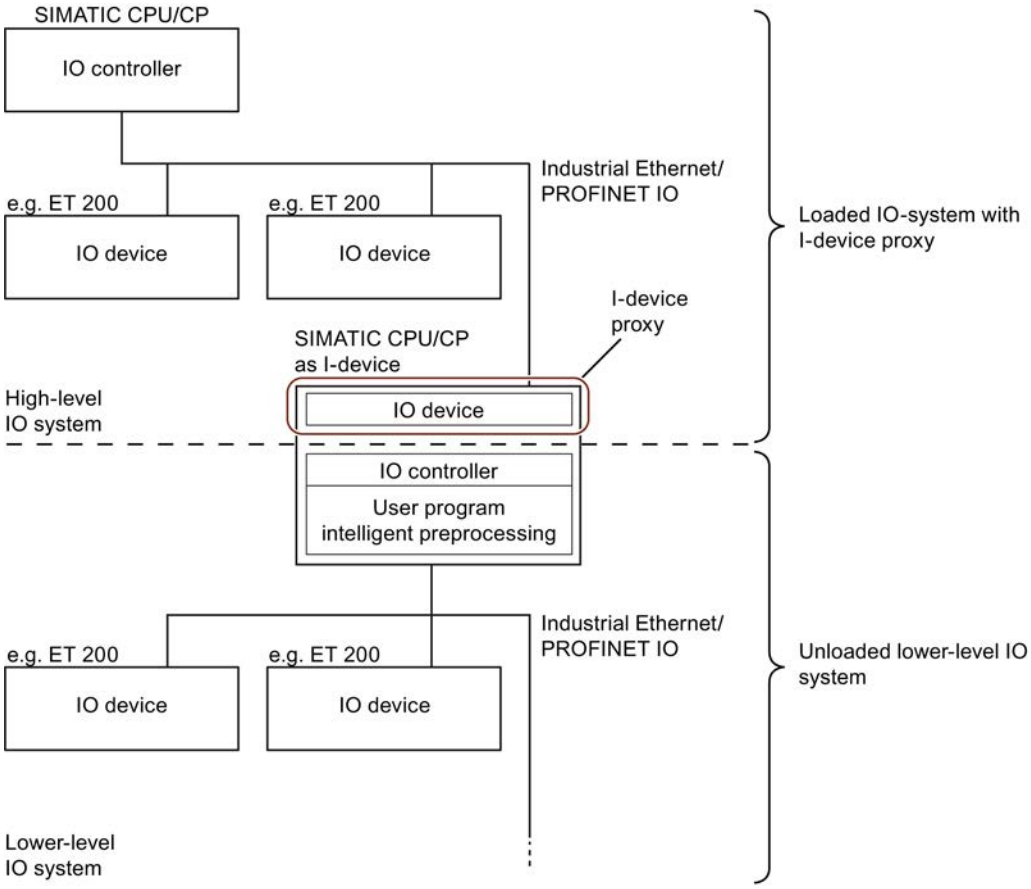
Empty Sync domains and MRP domains are not included in the loading.

GSD-based IO devices

As a prerequisite for loading GSD based IO devices, the corresponding GSD files must be installed in the TIA Portal and available in the hardware catalog. If a required GSD file is not available in the same version as in the device, differences are identified during the consistency check.

I-device

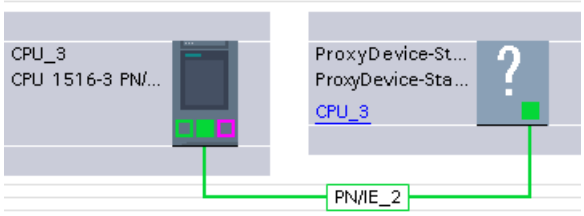
If you load the IO controller from an IO system with connected I-device, the IO controller and its IO devices will be loaded. An involved I-device is loaded only as an I-device proxy. The CPU parameter assignment, including the parameter assignment of the CPU's "own" (subordinate) IO system, is missing in the I-device proxy. Only the interface to the higher-level IO controller is loaded.



For I-device proxy devices:

- IO controller can be compiled and loaded
- Properties are displayed but cannot be changed
- Diagnostics in the project tree is not carried out

In the network view, placeholder devices are represented with a question mark:



In order to edit the I-device in the project, you must load the I-device from the device to the PG/PC. This replaces the I-device proxy with the complete I-device along with its subordinate IO devices.

It is possible to load an IO controller with connected I-device proxy from the PG/PC to the device.

Note

The replacement of I-device placeholders is only possible if the required I-device is available in the hardware catalog.

An I-device proxy representing a SIMOTION I-device cannot be loaded and replaced.

Configurations with IE/PB link

If one of the following configurations with IE/PB Link PN IO as PROFINET IO device is present, the complete configuration with all subordinate PROFIBUS devices is loaded:

- CPU/CP of S7-300/400
- PC station and connected PROFIBUS master system

The complete configuration consists of the following:

- CPU
- CP configuration
- PROFINET IO system with connected IE/PB link
- PROFIBUS master system of the IE/PB link with connected DP slaves

An example configuration consists of an S7-300 CPU with a CP as PROFINET IO controller. An IE/PB link as IO device is connected to the IO controller. As the PROFIBUS DP master,

the IE/PB link polls a PROFIBUS DP slave, e.g., ET 200L. If you load the CPU from the device to the PG/PC, the complete configuration is loaded.

Note

If the IE/PB link is not operated as a PROFINET IO proxy but rather as a gateway in standard mode, the IE/PB link functions as a CPU and can be loaded separately.

Loading Shared Devices

The following applies to loading comments: If an input/output module with the Module-internal Shared Input (MSI) or Module-internal Shared Output (MSO) function consists of only one submodule, the submodule does not have its own comment. Instead, it uses the comment of the input/output module. The module must be subdivided into several submodules for the input/output module and all submodules to have their own comment fields.

Loading HMI devices

Information on loading

We distinguish between the following cases when loading HMI devices to PG/PC:

- HMI devices connected to a DP master as DP slave or to an IO controller as IO device are loaded as DP slave or IO device respectively (for example, PP 17-I PROFIsafe).
- HMI devices in a master system as I-slave or in an IO system as I-device are loaded as I-slave proxies or I-device proxies (Page 1559) (for example, SIMATIC Comfort Panels). The settings of the device proxy are read-only.
- HMI devices are not loaded if they are connected to a subnet (PROFIBUS or PROFINET) but not to a master system or IO system (for example, KP600 Basic color DP).

Going online with loaded configurations

You can go online with the loaded project or with loaded portions of the project.

Requirement

The hardware configuration loaded from the device to the PG/PC has been compiled. The statuses of the central and distributed modules are only correctly displayed after compilation.

Note

If you go online before compiling, the diagnostics icon “?” is displayed (diagnostics not possible). A corresponding alarm is displayed under "Info > General" in the Inspector window.

Dependencies

Depending on how completely the hardware configuration was loaded to the PG, restrictions apply to going online and to diagnostics:

- Completely loaded device with all associated central and distributed modules, such as DP slaves or IO devices:
  Going online and diagnostics are possible.
- Loaded device with connected I-devices/I-slaves:
  - I device/I slave is not loaded: Going online for the device and its modules is possible. For the dependent components of the configuration that are not loaded, device proxies are handled with only minimum diagnostics support. The online status is represented as an icon. The standard diagnostics is shown in the online and diagnostics view. I&M data are not loaded.
  - I device/I slave is also loaded: Going online is possible for all devices; diagnostics is fully supported.

8.1.5 Additional information on configurations

8.1.5.1 Functional description of S7-1200 CPUs

Operating modes

Principles of the operating modes of S7-CPUs

Introduction

Operating modes describe the behavior of the CPU. The following operating modes are possible:

- STARTUP
- RUN
- STOP

In these operating modes, the CPU can communicate via the PN/IE interface, for example.

Other operating modes

If the CPU is not ready for operation, it is in one of following two operating modes:

- Deenergized, i.e. the supply voltage is switched off.
- Defective, which means an internal error has occurred.
  If the "Defective" status is caused by a firmware error, this state is indicated by the status LEDs of the CPU (refer to the description of the CPU). To find out the cause, follow these steps:
  - Turn the power supply switch off and on again.
  - Read out the diagnostics buffer when the CPU starts up and send the data for analysis to Customer Support.

  If the CPU does not start up, replace it.

See also

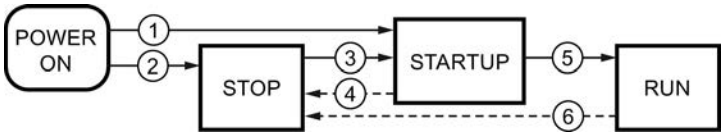
STOP mode (Page 1569)

RUN mode (Page 1569)

Operating mode transitions

Overview

The following figure shows the operating modes and the operating mode transitions of S7-1200 CPUs:



The following table shows the conditions under which the operating modes will change:

No.	Operating mode transition	Conditions
①	POWER ON → STARTUP	After switching on, the CPU goes to "STARTUP" mode if: • "Warm restart" startup type is set, and • the hardware configuration and the program blocks are consistent. Non-retentive memory is cleared and the contents of non-retentive DBs are reset to the initial values of the load memory. Retentive memory and retentive DB contents are retained.
②	POWER ON → STOP	When startup type "No startup" is set, the CPU goes to "STOP" mode after the supply voltage is switched on. Non-retentive memory is cleared and the contents of non-retentive DBs are reset to the initial values of the load memory. Retentive memory and retentive DB contents are retained.

No.	Operating mode transition	Conditions
③	STOP → STARTUP	The CPU switches to "STARTUP" mode if: • CPU is set to "RUN" from the programming device, and • the hardware configuration and the program blocks are consistent.
④	STARTUP → STOP	The CPU returns to the "STOP" mode in the following situations: • Error detected during startup. • The CPU is set to "STOP" from the programming device. • A STOP command is processed in the STARTUP OB.
⑤	STARTUP → RUN	If the STARTUP is successful, the CPU switches to "RUN".
⑥	RUN → STOP	The CPU returns to the "STOP" mode in the following situations: • An error is detected that prevents continued processing. • The CPU is set to "STOP" from the programming device. • A STOP command is processed in the user program.

"STARTUP" operating mode

Principles of the STARTUP mode

Function

After turning on the CPU, it executes a startup program before starting to execute the cyclic user program.

By suitably programming startup OBs, you can specify certain initialization variables for your cyclic program in the startup program. There is no rule in terms of the number of startup OBs. That is, you can set up one or several startup OBs in your program, or none at all.

Parameter settings for startup characteristics

You can specify whether the CPU remains in STOP mode or whether a warm restart is run. Over and above this, you can set the response during startup (RUN or previous mode) in the "Startup" group of the CPU properties.

Special characteristics

Note the following points regarding the "STARTUP" mode:

- The startup OBs are executed. All startup OBs you have programmed are executed, regardless of the selected startup mode.
- No time-based program execution can be performed.
- Interrupt controlled program execution limited to:
  - OB 82 (diagnostics interrupt)
- The outputs on the modules are disabled.
- The process image is not updated; direct I/O access to inputs is possible.

See also

Editing properties and parameters (Page 513)

Principles of the operating modes of S7-CPUs (Page 1563)

Organization blocks for startup (Page 1614)

Warm restart (Page 1566)

Warm restart

Function

During a warm restart, all non-retentive bit memory is deleted and non-retentive DB contents are reset to the initial values from load memory. Retentive bit memory and retentive DB contents are retained.

Program execution begins at the call of the first startup OB.

Triggering a warm restart

You can trigger a "Warm restart" using a corresponding menu command on your programming device in the following situations:

- The CPU must be in "STOP" mode.
- After a memory reset
- After downloading a consistent program and a consistent hardware configuration in the "STOP" mode of the CPU.

"POWER ON" triggers a "warm restart" if you have set the following parameters for the startup response:

- Startup type "warm restart - RUN" (regardless of the CPU operating mode prior to POWER OFF).
- "Warm restart - mode prior to POWER OFF" (depending on the CPU operating mode prior to POWER OFF. The CPU must have been in RUN mode prior to this.)

See also

Retentive memory areas (Page 1574)

Startup activities

Overview

The following table shows which activities the CPU performs at STARTUP:

Activities in execution sequence	At warm restart
Clear non-retentive bit memories	Yes
Clear all bit memories	No
Clear the process image output	Yes
Processing startup OBs	Yes
Update the process image input	Yes
Enable outputs after changing to "RUN" mode	Yes

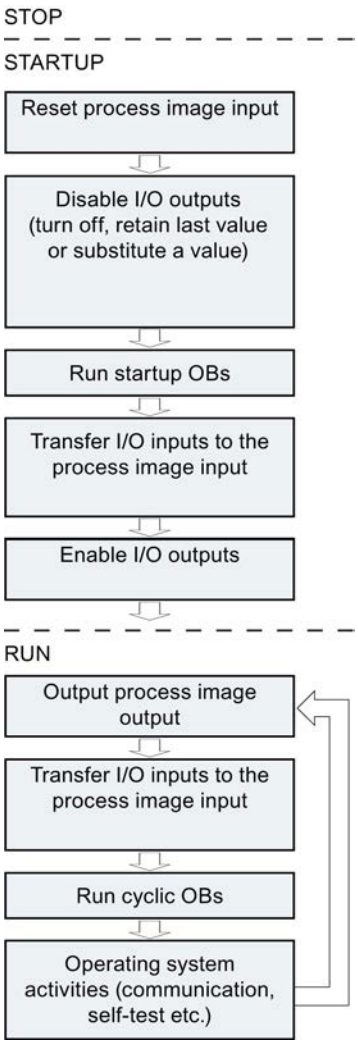
Sequence

The following figure shows the activities of the CPU in "STOP", "STARTUP", and "RUN" modes.

You can use the following measures to specify the state of the I/O outputs in the first cycle of the user program:

- Use assignable output modules to be able to output substitute values or to retain the last value.
- Set default values for outputs in startup OBs.

During the startup, all interrupt events are entered in a queue so that they can be processed later during RUN mode. In RUN mode, hardware interrupts can be processed at any time.



Special features during startup

Response when expected and actual configurations do not match

The expected configuration is represented by the engineering configuration loaded on the CPU. The actual configuration is the actual configuration of the automation system.

If the expected configuration and actual configuration differ, the CPU nevertheless initially changes to RUN.

Canceling a STARTUP

If errors occur during startup, the startup is canceled and the CPU remains in "STOP" mode.

Under the following conditions, a startup will not be performed or will be canceled:

- If an invalid SD card is inserted.
- If no hardware configuration has been downloaded.

See also

Overview of the CPU properties (Page 1586)

RUN mode

Function

In "RUN" mode the cyclic, time-driven, and interrupt-driven program sections execute:

- The process image output is read out.
- The process image input table is read.
- The user program is executed.

Active data exchange between S7-1200 CPUs by means of Open User Communication is only possible in "RUN" mode.

Running the user program

Once the CPU has read the inputs, the cyclic program runs from the first to the last instruction.

If you have configured a minimum cycle time, the CPU will not end the cycle until this minimum cycle time is up even if the user program is completed sooner.

A maximum cycle time is set which you can adjust according to your requirements. This ensures that the cyclic program is completed within a specified time. The system will respond with a time error if the cyclic program is not completed within this time.

Other events such as hardware and diagnostic interrupts can interrupt the cyclic program flow and prolong the cycle time.

See also

Principles of the operating modes of S7-CPUs (Page 1563)

Events and OBs (Page 1578)

STOP mode

Function

In "STOP" mode, the user program is not executed. All outputs are disabled or react according to the parameter settings: They provide a substitute value as set in the parameters or retain the last value output and bring the controlled process to a safe status.

The CPU checks the following points:

- Hardware, for example whether are all modules are available
- Whether the default settings for the CPU are applicable or parameter sets are present
- Whether the general conditions for the programmed startup behavior are correct

See also

Principles of the operating modes of S7-CPUs (Page 1563)

Basics of a memory reset

Function

A memory reset on the CPU is possible only in STOP mode.

When memory is reset, the CPU is changed to an "initial status". This means:

- An existing online connection between your programming device/PC and the CPU is terminated.
- The content of the work memory and the retentive and non-retentive data are deleted.
- The diagnostic buffer, the time, the IP address, the hardware configuration, and active force jobs are retained.
- The load memory (code and data blocks) is then copied to work memory. As a result, the data blocks no longer have current values but their configured initial values.

Memory areas

Useful information on memory cards

How the memory card functions

The SIMATIC Memory Card for a S7-1200 is an SD memory card preformatted by Siemens for the CPU user program.

You may only delete files and folders. If you format the memory card with Windows, for example with a commercially available card reader, you make the memory card unusable as a storage medium for an S7 CPU.

Setting the card type

You can use the memory card as a transfer card, a program card or a firmware update card.

To set the card type, insert the memory card into the card reader of the programming device and select the "Card reader/USB memory" folder from the project tree. In the properties of the selected memory card, designate the card type:

- Program
  If it is used as a program card, you can load the user program on the memory card. In this case, the internal load memory of the device is replaced by the memory card and the internal load memory is erased. The user program is then fully executable from the memory card. If the memory card with the user program is removed, there is no longer a program available.
- Transfer
  If it is used as a transfer card, you can transfer the user program from the memory card to the internal load memory of the CPU. You can then remove the memory card again.
- Firmware card
  Firmware for the S7-1200 modules can be stored on a memory card. Therefore it is possible to perform a firmware update with the help of a specifically prepared memory card. Likewise, a backup copy of firmware for a module can be stored on a memory card.

Transferring objects from the project to a memory card

When the memory card is inserted in the programming device or in an external card reader, you can transfer the following objects from the project tree to the memory card:

- Individual blocks (multiple selection possible)
  In this case a consistent transfer is available, as the dependencies of the blocks to each other is taken into account with block selection.
- PLC
  In this case, all objects relevant to processing are transferred, such as blocks and the hardware configuration on the memory card, just as with downloading.

To perform the transfer, you can move the objects with drag-and-drop or use the command "Card reader/USB memory > Write to memory card" in the "Project" menu.

Transferring objects from the memory card to the project

You can transfer Individual blocks (multiple selection is possible) by dragging them to the project. A hardware configuration cannot be transferred from the memory card to the project.

Updating firmware with a memory card

You can get the latest firmware data on the Internet from the Service & Support pages:

http://support.automation.siemens.com (http://support.automation.siemens.com/WW/view/en/34143537)

Save the firmware files on the hard disk and plug the SIMATIC Memory Card into the card reader of your programming device.

To store the file on the memory card, select the memory card in the "Card Reader/USB memory" folder in the project tree. Select the shortcut menu "Card Reader/USB memory > Create firmware update memory card".

Then follow the instructions in the Service & Support portal for performing a firmware update with your CPU.

Updating the firmware changes the CPU firmware status. If you have used the CPU in the project, you will have to update the CPU already configured to the CPU with the new firmware status by changing devices offline, and adapt and then load the program or configuration.

See also

Replacing a hardware component (Page 513)

Useful information on CPU firmware versions and STEP 7 versions (Page 1637)

Load memory

Function

Each CPU has an internal load memory. The size of this internal load memory depends on the CPU used.

This internal load memory can be replaced by using external memory cards. If there is no memory card inserted, the CPU uses the internal load memory; if a memory card is inserted, the CPU uses the memory card as load memory.

The size of the usable external load memory cannot, however, be greater than the internal load memory even if the inserted SD card has more free space.

See also

Using memory cards (Page 412)

Work memory

Function

Work memory is a non-retentive memory area for storing elements of the user program that are relevant for program execution. The user program is executed exclusively in work memory and system memory.

System memory

System memory areas

Function

System memory contains the memory elements that each CPU makes available to the user program, such as the process image and bit memory.

By using appropriate operations in your user program, you address the data directly in the relevant operand area.

The following table shows the operand areas of the system memory:

Operand area	Description	Access via units of the following size:	S7 notation
Process image output	The CPU writes the values from the process image output table to the output modules at the start of the cycle.	Output (bit)	Q
		Output byte	QB
		Output word	QW
		Output double word	QD
Process image input	The CPU reads the inputs from the input modules and saves the values to the process image input table at the start of the cycle.	Input (bit)	I
		Input byte	IB
		Input word	IW
		Input double word	ID
Bit memory	This area provides storage for intermediate results calculated in the program.	Bit memory (bit)	M
		Memory byte	MB
		Memory word	MW
		Memory double word	MD
Data block	Data blocks store information for the program. They can either be defined so that all code blocks can access them (global DBs) or assigned to a specific FB or SFB (instance DB). Requirement: The block attribute "Optimized block access" is not enabled.	Data bit	DBX
		Data byte	DBB
		Data word	DBW
		Data double word	DBD
Local data	This area contains the temporary data of a block while the block is being processed. Requirement: The block attribute "Optimized block access" is not enabled. Recommendation: Access local data (temp) symbolically.	Local data bit	L
		Local data byte	LB
		Local data word	LW
		Local data double word	LD
I/O input area	The I/O input and output areas permit direct access to central and distributed input and output modules.	I/O input bit	<tag>:P
		I/O input byte	
		I/O input word	
		I/O input double word	
I/O output area		I/O output bit	
		I/O output byte	
		I/O output word	
		I/O output double word	

See also

Diagnostics buffer (Page 1576)

Basic principles of process images (Page 1575)

Basic principles for programming of data blocks (Page 2825)

Declaring local tags and constants in the block interface (Page 2550)

Layout of the block interface (Page 2544)

Access to the I/O addresses (Page 1577)

Retentive memory areas

Retentive memory areas

Data loss after power failure can be avoided by marking certain data as retentive. This data is stored in a retentive memory area. A retentive memory area is an area that retains its content following a warm restart, in other words, after cycling the power when the CPU changes from STOP to RUN.

The following data can be assigned retentivity:

- Bit memory: The precise width of the memory can be defined for bit memory in the PLC tag table or in the assignment list.
- Tags of a function block (FB): You can define individual tags as retentive in the interface of an FB if you have enabled optimized block access. Retentivity settings can be defined only in the assigned instance data block if optimized block access has not been activated for the FB.
- Tags of a global data block: You can define retentivity either for individual or for all tags of a global data block depending on the settings for access.
  - Block with optimized access: retentivity can be set for each individual tag.
  - Block with standard access: The retentivity setting applies to all tags of the DB; either all tags are retentive or no tag is retentive.

See also

Setting the retentivity of PLC tags (Page 2483)

Warm restart (Page 1566)

Basics of block access (Page 2175)

process image input/output

Basic principles of process images

Function

When the user program addresses the input (I) and output (O) operand areas, it does not query or change the signal states on the digital signal modules. Instead, it accesses a memory area in the system memory of the CPU. This memory area is referred to as the process image.

Advantages of the process image

Compared with direct access to input and output modules, the main advantage of accessing the process image is that the CPU has a consistent image of the process signals for the duration of one program cycle. If a signal state on an input module changes during program execution, the signal state in the process image is retained until the process image is updated again in the next cycle. The process of repeatedly scanning an input signal within a user program ensures that consistent input information is always available.

Access to the process image also requires far less time than direct access to the signal modules since the process image is located in the internal memory of the CPU.

Updating the process images

Sequence

The operating system updates the process images at cyclic intervals unless defined otherwise in your configuration. The process image input/output is updated in the following order:

1. The internal tasks of the operating system are performed.
2. The process image output (PIQ) table is written to the outputs of the module.
3. The status of inputs is read to the process image input (PII) table.
4. The user program is executed with all the blocks that are called in it.

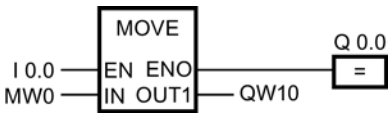
The operating system automatically controls the writing of the process image output to the outputs of the modules and the reading of the process image input.

Special characteristics

You have the option of accessing inputs or outputs directly using direct I/O access.

- If an instruction accesses an output directly and the output address is located in the process image output, the process image of the relevant output is updated.
- If an instruction accesses an output directly and the output address is **not** located in the process image output, the process image of the relevant output is **not** updated.

Example of normal I/O access by way of the process image



Update QW10 in the I/O output area with the value from MW0.

I/O access error during process image updating

If an error occurs during process image updating (I/O access error), the CPU reacts with the default system reaction "Ignore".

See also

Basics of error handling (Page 2821)

Start address of a module (Page 1577)

Access to the I/O addresses (Page 1577)

Startup activities (Page 1567)

Diagnostics buffer

Function

The diagnostics buffer is part of the system memory of the CPU. It contains the errors detected by the CPU or modules with diagnostics capability. It includes the following events:

- Every mode change of the CPU (for example, POWER UP, change to STOP mode, change to RUN mode)
- Every diagnostics interrupt

The diagnostics buffer of the S7-1200-CPU has a capacity of 50 entries of which the last (most recent) 10 entries are retained following power cycling.

Those entries can only be cleared by restoring the CPU to factory defaults.

You can read the content of the diagnostics buffer with the help of the Online and Diagnostics view.

See also

Basic information on the diagnostics buffer (Page 2123)

I/O data area

Start address of a module

Definition

The start address is the lowest byte address of a module. It acts as the initial address of the module user data area.

Configuring module start addresses

The addresses used in the user program and the module start addresses are coordinated when the modules are configured.

In the module properties ("I/O addresses" group), you can change the start addresses that were assigned automatically after the modules were inserted.

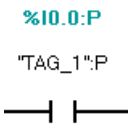
You can also make a setting that decides whether or not the addresses are located in the process image.

Access to the I/O addresses

I/O addresses

If you insert a module in the device view, its user data is located in the process image of the S7-1200 CPU (default). The CPU handles the data exchange between the module and the process image area automatically during the update of the process images.

Append the suffix ":P" to the I/O address if you want the program to access the module directly instead of using the process image.



This could be necessary, for example, during execution of a time-sensitive program which also has to control the outputs within the same cycle.

Basics of program execution

Events and OBs

Events and OBs

The operating system of S7-1200-CPUs is based on events. There are two types of events:

- Events which can start an OB
- Events which cannot start an OB

An event which can start an OB triggers the following reaction:

- It calls the OB you possibly assigned to this event. The event is entered in a queue according to its priority if it is currently not possible to call this OB.
- The default system reaction is triggered if you did not assign an OB to this event.

An event which cannot start an OB triggers the default system reaction for the associated event class.

The user program cycle is therefore based on events, the assignment of OBs to those events, and on the code which is either contained in the OB, or called in the OB.

The following table provides an overview of the events which can start an OB, including the associated event classes and OBs. The table is sorted based on the default OB priority. Priority class 1 is the lowest.

Event class	OB no.	Number of OBs	Start event	OB priority (default)
Cyclic program	1, >= 123	>= 1	Starting or end of the last program cycle OB	1
Startup	100, >= 123	>=0	STOP to RUN transition	1
Time-of-day interrupt	>= 10	Max. 2	Start time has been reached	2
Time-delay interrupt	>= 20	Max. 4	Delay time expired	3
Cyclic interrupt	>= 30		Constant bus cycle time expired	8
Hardware interrupt	>= 40	Max. 50 (more can be used with DETACH and ATTACH)	• Positive edge (max. 16) • Negative edge (max. 16)	18
			• HSC: Count value = reference value (max. 6) • HSC: Count direction changed (max. 6) • HSC: External reset (max. 6)	18
Status interrupt	55	0 or 1	CPU has received status interrupt	4
Update interrupt	56	0 or 1	CPU has received update interrupt	4
Manufacturer- or profile-specific interrupt	57	0 or 1	CPU has received manufacturer-specific or profile-specific interrupt	4
Diagnostic error interrupt	82	0 or 1	Module has detected an error	5
Pull/plug interrupt	83	0 or 1	Removal/insertion of modules of distributed I/O	6

Event class	OB no.	Number of OBs	Start event	OB priority (default)
Rack error	86	0 or 1	Error in the I/O system of the distributed I/O	6
Time error	80	0 or 1	• Maximum cycle time exceeded • Called OB is still being executed • Time-of-day interrupt missed • Time-of-day interrupt missed during STOP • Queue overflow • Interrupt loss due to high interrupt load	22

The following table describes events which do not trigger an OB start, including the corresponding reaction of the operating system. The table is sorted based on event priority.

Event class	Event	Event priority	System reaction
Insert/remove central modules	Insert/remove a module	21	STOP
I/O access error during process image update	I/O access error during process image update	22	Ignore
Programming error	Programming error in a block for which you use system reactions provided by the operating system (note: the error handling routine in the block program is executed if you activated local error handling).	23	RUN
I/O access error	I/O access error in a block for which you use system reactions provided by the operating system (note: the error handling routine in the block program is executed if you activated local error handling).	24	RUN
Maximum cycle time exceeded twice	Maximum cycle time exceeded twice	27	STOP

Assignment between OBs and events

With the exception of the cyclic program and startup program and event can only be assigned to one OB. However, in certain event classes such as hardware interrupts one and the same OB can be assigned to several events.

The assignment between OBs and events is defined in the hardware configuration. Defined assignments can be changed at runtime by means of ATTACH and DETACH instructions.

OB priority and runtime behavior

S7-1200-CPUs support the priority classes 1 (lowest) to 27 (highest). An OB is assigned the priority of its start event.

OBs are always executed on a priority basis: The OBs with the highest priority are executed first. Events of the same priority are processed in order of occurrence.

As of firmware version V4.0 of the S7-1200 CPUs you can specify in the device configuration, under properties of the CPU, if the OBs are interruptible or not. This parameter assignment has an effect on all OBs with exception of the cycle OBs which are always interruptible.

The following applies to S7-1200 CPUs with firmware version < V4.0:

- Any OB with priority >= 2 will interrupt cyclic program execution.
- An OB of priority 2 to 25 cannot be interrupted by any event of priority group 2 to 25. This rule also applies to events of a priority higher than that of the currently active OB. Such events are processed later.
- A time error (priority 26) will interrupt any other OB.

The following applies to S7-1200 CPUs as of firmware version V4.0:

If you do not configure the OBs as interruptible, an OB is always processed completely even if an event of a higher priority occurs during its runtime. Specifically, this means:

- Any OB with priority >= 2 will interrupt cyclic program execution.
- An OB of priority 2 to 25 cannot be interrupted by any event. This rule also applies to events of a priority higher than that of the currently active OB, which also includes a time error. Such events are processed later.

If you do configure the OBs as interruptible and an event of a higher priority occurs during the runtime of an OB, the running OB is interrupted and the OB associated with the occurring event is processed. Once this OB has been completed, processing of the interrupted OB continues. Specifically, this means:

- Any OB with priority >= 2 will interrupt cyclic program execution.
- An OB of priority 2 to 25 can be interrupted by any event whose priority is higher than that of the running OB. This is also true for time errors: A time error (priority 26) will interrupt any OB.

OB start information

Certain OBs have start information, while others do not. This is explained in greater detail in the description of the relevant OB.

See also

Event-based program execution (Page 1581)

Principles of local error handling (Page 2823)

Organization blocks (OB) (Page 2170)

Basics of error handling (Page 2821)

ATTACH: Attach an OB to an interrupt event (Page 4691)

DETACH: Detach an OB from an interrupt event (Page 4693)

Event-based program execution

OB priority and runtime behavior

S7-1200-CPUs support the priority classes 1 (lowest) to 27 (highest). An OB is assigned the priority of its start event.

Interrupt OBs can only be interrupted by time error interrupts. This rule also applies to events of a priority higher than that of the currently active OB. That is, only one interrupt OB can be active, with exception of the time error interrupt OB.

Any further event of generated while an interrupt OB is being executed is added to a queue in accordance with its priority. Start events within a queue are processed later based on the chronological order of their occurrence.

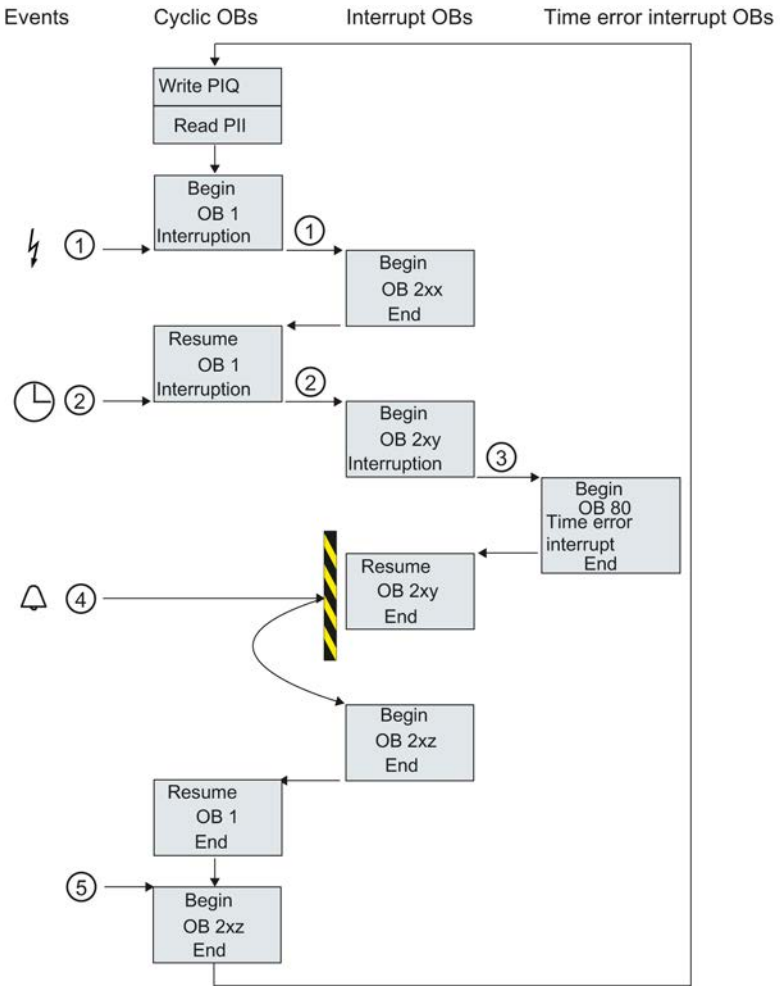
Program execution on the CPU

Cyclic OBs are interrupted by interrupt OBs.

Reactions to events which start an interrupt OB:

- For CPUs up to firmware version V3: Interrupt OBs can only be interrupted by time error interrupt OBs.
- For CPUs as of firmware version V4: Interrupt OBs can be interrupted by interrupt OBs of a higher priority.

The figure below shows the basic procedure in case interrupt OBs cannot be interrupted (behavior up to firmware version V3):



① and ② An event (e.g. a hardware interrupt) calls its associated OB.

A called OB is executed without interruption, including all of its nested blocks. Execution of the cyclic OB is resumed on completion of interrupt processing, provided the queue does not contain any events which trigger an OB start.

③ An interrupt OB can only be interrupted by a time error interrupt OB (OB 80).

④ An new alarm-triggering event occurs during interrupt processing.

Reaction for CPUs up to and including firmware version V3:

This new event is added to a queue. The queued events successively call their corresponding OBs only after execution of the current interrupt OBs was completed and according to the following rules:

- Events are processed in the order of their priority (starting at the highest priority).
- Events of the same priority are processed in chronological order.

Reaction for CPUs as of firmware version V4:

You use a CPU parameter to set the interruptibility for CPUs as of firmware version V4. Default behavior: OBs are interruptible. In this case: If the new event has a higher priority than the currently running OB, the OB started by the new event interrupts the currently running OB.

If you disable the option, the interrupt OBs cannot be interrupted.

⑤ The cyclic OBs are processed one after the other.

Notes on queues

- Every priority class (OBs of the same priority to be called) is assigned a separate queue. The size of those queues is set by default.
- Any new event leading to the overflow of a queue is discarded and therefore lost. A "time error interrupt event" is generated simultaneously. Information identifying the OB that caused the error is included in the start information of the time error interrupt OB (OB 80). A corresponding reaction such as an alarm trigger can be programmed in the time error interrupt OB.

Example of a hardware interrupt event

The function principle of event-oriented program execution in the S7-1200 CPU is described based on the example of a hardware interrupt-triggering module.

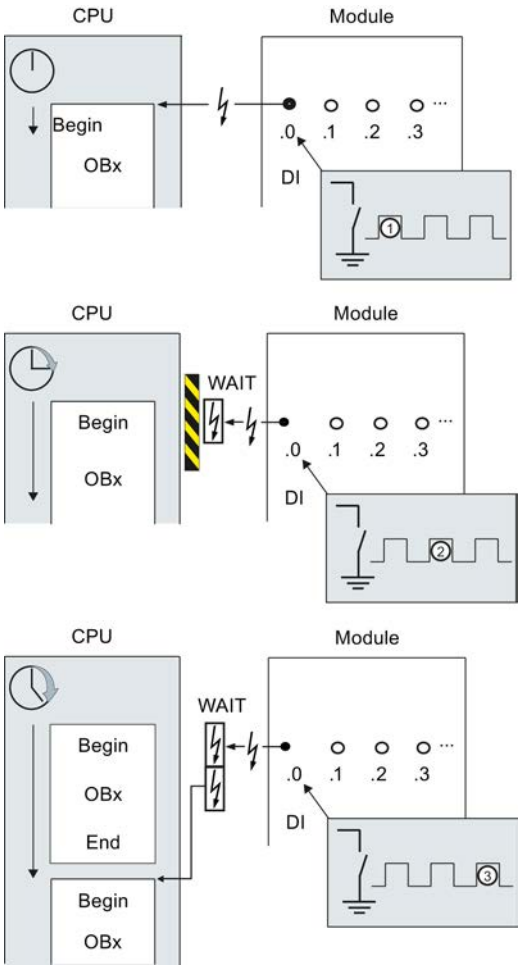
Process events and their priority

Process events are triggered by the I/O (e.g. at a digital input) and initiate a call of the assigned OB in the S7-1200 CPU. OBs assigned to a process event are called hardware interrupt OBs.

Examples of process events and their priority:

- Process events "rising edge" or "falling edge" at an interrupt-triggering module: The hardware interrupt OB started by such an event is always assigned priority 5.
- Process events from a high-speed counter
  - Count value corresponds to the reference value
  - Change count direction
  - External reset of the high-speed counter
    The hardware interrupt OB started by this event is always assigned priority 6.

The figure below shows the chronological sequence of hardware interrupt execution: In the case of two hardware interrupt events in immediate succession, the second hardware interrupt triggering event is held back in the queue until the first OBx has been processed. The next hardware interrupt triggering event can only start the associated OBx when the OBx has been processed. Additional hardware interrupt triggering events are lined up in the queue according to this principle.



Hardware interrupt execution

① A hardware interrupt-triggering event such as a rising edge at the input calls the OB to which it is assigned.

② If a new event occurs that triggers a hardware interrupt while the OB is executing, this event is entered in a queue.

③ The new event that triggers a hardware interrupt starts the hardware interrupt OB assigned to the event.

Assigning the interrupt-triggering event

The interrupt-triggering event is assigned to an OB in the input properties of the device view.

- An interrupt-triggering event can only be assigned to a single OB.
- OBs, however, can be assigned to several interrupt-triggering events.
  This means, for example, that you can assign both the rising and the falling edge to the same interrupt OB in order to trigger the same reaction to any change of the input signal.

- The started OB can interrupt a cycle OB at every instruction. Consistent data access is secured up to dword size.
- You can parameterize module-specific interrupt-triggering events such as a rising and the falling edge at the input.
- Assign the interrupt-triggering event and the OB to be started in the configuration of the interrupt-triggering module. However, within the started hardware interrupt OB you can override this assignment using the DETACH instruction, or assign the same event to a different OB using the ATTACH instruction. This functionality allows a flexible reaction to external process signals.

Setting the operating behavior

Changing properties of the modules

Default settings

When they leave the factory, all hardware components with parameters have default settings suitable for standard applications. These default values allow the hardware components to be used immediately without making any additional settings.

You can, however, modify the behavior and the properties of the hardware components to suit the requirements and circumstances of your application. Hardware components with settable parameters include, for example, communications modules and several analog and digital modules.

Setting and loading parameters

When you have selected a hardware component in the device or network view, you can set the properties in the Inspector window. When you save a device configuration with its parameters, data is generated that needs to be loaded on the CPU. This data is transferred to the relevant modules during startup.

Properties of the CPUs

The properties of the CPUs have special significance for system behavior. For example for a CPU you can set:

- Interfaces
- Inputs and outputs
- High-speed counters
- Pulse generators
- Startup behavior
- Time-of-day
- Protection level
- Bit memory for system and clock

- Cycle time
- Communications load

The entry possibilities specify what is adjustable and in which value ranges. Fields that cannot be edited are disabled or are not shown in the properties window.

Requirement

You have already arranged the hardware components for which you want to change properties on a rack.

Procedure

To change the properties and parameters of the hardware components, follow these steps:

1. In the device or network view, select the hardware component or interface that you want to edit.
2. Edit the settings for the selected object:
   – For example in the device view you can edit addresses and names.
   – In the Inspector window additional setting possibilities are available.

You do not need to confirm your entries, the changed values will be applied immediately.

See also

Editing properties and parameters (Page 513)

Introduction to loading a configuration (Page 1553)

CPU properties

Overview of the CPU properties

Overview

The following table provides you with an overview of the CPU properties:

Group	Properties	Description
General	Project information	General information to describe the inserted CPU. Except for the slot number, you can change this information.
	Catalog information	Read-only information from the hardware catalog for this CPU.
PROFINET interface	General	Name and comment for this PROFINET interface. The name is limited to 110 characters.

Group	Properties	Description
	Ethernet addresses	Select whether the PROFINET interface is networked. If subnets have already been created in the project, they are available for selection in the drop-down list. If not, you can create a new subnet with the "Add new subnet" button. Information on the IP address, subnet mask and IP router usage in the subnet is available in the IP protocol. If an IP router is used, the information about the IP address of the IP router is necessary.
	Advanced options	Name, comment and additional setting options of the Ethernet interface port.
	Time synchronization	Settings for time synchronization in the NTP time format. The NTP (network time protocol) is a general mechanism for synchronizing system clocks in local and global area networks. In NTP mode, the interface of the CPU sends time queries (in client mode) at regular intervals to NTP servers on the subnet (LAN) and the addresses must be set in the parameters here. Based on the replies from the server, the most reliable and most accurate time is calculated and synchronized. The advantage of this mode is that it allows the time to be synchronized across subnets. The accuracy depends on the quality of the NTP server being used.
DI#/DO#	General	Name of and comment on the integrated digital inputs of the CPU.
	Digital inputs	Input delays can be set for digital inputs. The input delays can be set in groups (in each case for 4 inputs). The detection of a positive and a negative edge can be enabled for each digital input. A name and a hardware interrupt can be assigned to this event. Depending on the CPU, pulse catches can be activated at various inputs. When the pulse catch is activated, even pulse edges that are shorter than the cycle time of the program are detected.
	Digital outputs	The reaction to a mode change from RUN to STOP can be set for all digital outputs: The state can either be frozen (corresponds to retain last value) or you set a substitute value ("0" or "1")
	I/O addresses	The address space of the input and output addresses is specified as is the process image.
	Hardware identifier	The hardware identifier of the device is displayed.
AI#	General	Name of and comment on the integrated analog inputs of the CPU.

Group	Properties	Description
	Analog inputs	During noise reduction, the specified integration time suppresses interference frequencies at the specified frequency (in Hz). The channel address, measurement type, voltage range, smoothing and overflow diagnostics must be specified in the "Channel #" group. The measurement type and voltage range are set permanently to voltage, 0 to 10 V. Smoothing analog values provides a stable analog signal for further processing. Smoothing analog values can be useful with slow measured value changes, for example, in temperature measurement. The measured values are smoothed with digital filtering. Smoothing is achieved by the module forming mean values from a specified number of converted (digitalized) analog values. The selected level (slight, medium, strong) decides the number of analog signals used to create the mean value. If overflow diagnostics is enabled, a diagnostics event is generated if an overflow occurs.
	I/O addresses	The address space of the input addresses is specified as is the process image.
	Hardware identifier	The hardware identifier of the device is displayed.
High-speed counter (HSC)	High-speed counter (HSC)#	High-speed counters are typically used to drive counting mechanisms. See: Configuring high-speed counters (Page 1597)
Pulse generators (PTO/PWM)	PTO#/PWM#	A pulse generator is activated and can be initialized with project information. For the configuration of an activated pulse generator, specify the usage as PWM (Pulse Width Modulation) or as PTO (Pulse Train Output). Specify the output source, time base, pulse width format, cycle time and initial pulse width for PWM. A pulse output is specified as the hardware output. The PWM output is controlled by the CTRL_PWM instruction, see CTRL_PWM (Page 4766). Specify the output source for PTO. A pulse output and a direction output are specified as the hardware outputs. A PTO is operated together with a high-speed counter in the "axis of motion" count mode and controlled by the Motion Control technology object (see keyword "Motion Control S7-1200") . The hardware ID is displayed in the I/O-diagnostics addresses and, if the PWM function is selected, the address space of the output addresses and the process image can be selected.
Startup	Startup after POWER ON	Setting the startup characteristics after cycling power. See: Principles of the STARTUP mode (Page 1565)
	Comparison of preset configuration and actual configuration	Specifies whether modules (SM, SB, CM, CP or even the CPU) can be replaced: • Startup of the CPU only if compatible • Startup of the CPU even if there are differences Example: A signal module with 16 digital inputs and 16 digital outputs (DI16/DQ16) can be a compatible replacement for a signal module with 8 digital outputs (DQ8) or 4 digital inputs (DI4).

Group	Properties	Description
	Parameter assignment time for distributed I/O	Specifies a maximum period (standard: 60000 ms) in which the distributed I/O must start up. (The CMs and CPs are supplied with voltage and communication parameters during the CPU startup. This configuration time provides a time period during which I/O modules connected to the CM or CP must start up.) The CPU switches to RUN as soon as the distributed I/O has started and is ready for operation, regardless of the "Parameter assignment time for distributed I/O" parameter. If the distributed I/O has not started up during this period, the CPU switches to RUN without the distributed I/O.
Cycle	Maximum cycle time and minimum cycle time.	Specification of a maximum cycle time or a fixed minimum cycle time. If the cycle time exceeds the maximum cycle time, the CPU goes to STOP mode. See: Cycle time and maximum cycle time (Page 1590)
Communication load	Maximum allocation of the cycle for communication (as a percentage)	Controls the duration of communication processes that always also extend the cycle time, within certain limits. Examples of communication processes include: Transferring data to another CPU or loading blocks (initiated via the PC). See: Cycle loading by communications (Page 1591)
System and clock memory	System memory bits and clock memory bits	You use system memory bits for the following scans: • Is the current cycle the first since cycling power? • Have there been any diagnostics state changes compared with the previous cycle? • Scan for "1" (high) • Scan for "0" (low) Clock memory bits change their values at specified periodic intervals. See: Enabling system memory (Page 1607) See: Using clock memory (Page 1608)
Web server	Automatic update	Sends the requested web page with current CPU data periodically to the web browser. Enter the period duration under "Update interval". Automatic update can only be activated if the web server is enabled. See: Activate automatic update (Page 909)
	User-defined web pages	Allows access to freely-designed web pages of the CPU via a web browser. See: What you need to know about user-defined web pages (Page 913)
Time	Local time and daylight saving time	Setting of the time zone in which the CPU is operated and setting of the daylight-saving/standard time changeover.
Protection	Protection and password for read/write access	Setting the read/write protection and the password for access to the CPU. See: Setting options for the protection level (FW V1 to V3) (Page 1609) See: Setting options for the protection (FW as of V4) (Page 1610)

Group	Properties	Description
Connection resources	-	Display of available, reserved and previously configured connection resources of the CPU.
Address overview	-	Tabular representation of all addresses used by the CPU for integrated inputs/outputs as well as for the inserted modules. Addresses that are not used by any module are represented as gaps. The view can be filtered according to • Input addresses • Output addresses • Address gaps

See also

Specifying input and output addresses (Page 938)

Assigning parameters to hardware interrupt OBs (Page 1631)

Access to the I/O addresses (Page 1577)

Addressing modules (Page 937)

Special features during startup (Page 1568)

Cycle time and maximum cycle time

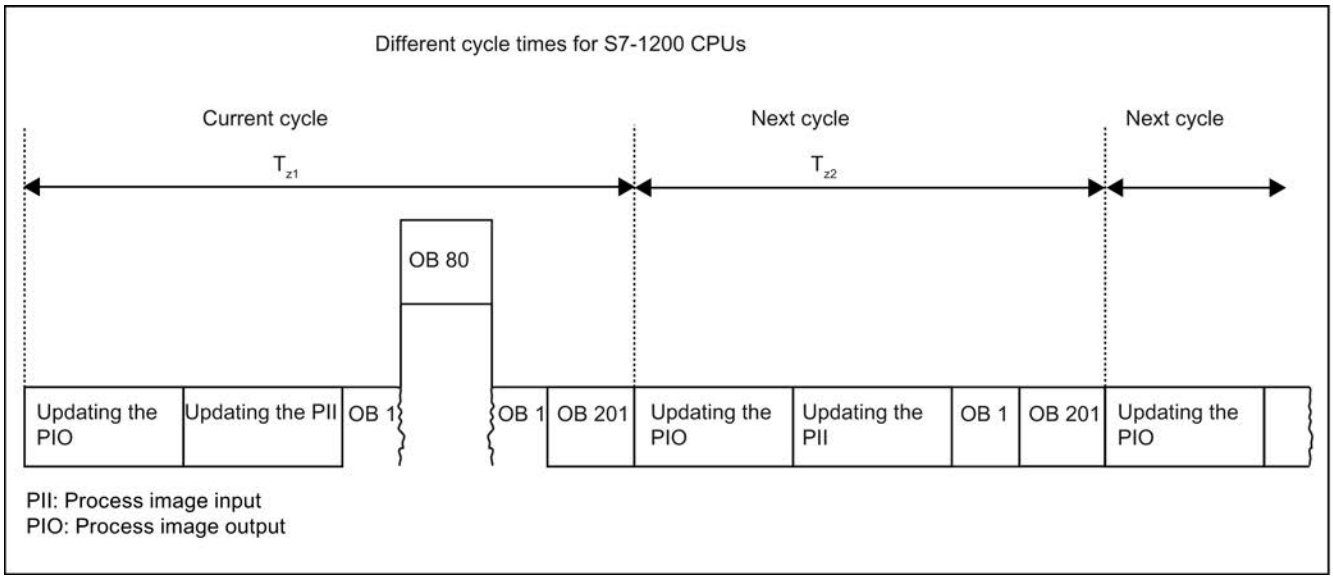
Function

The cycle time is the time that the operating system requires to execute the cyclic program and all the program sections that interrupt this cycle. The program execution can be interrupted by:

- Time errors and 2xMaxCycleTime errors
- System activities, e.g., process image updating

The cycle time (Tcyc) is therefore not the same for every cycle.

The following schematic shows an example of different cycle times (TZ1 ≠ TZ2) for S7-1200 CPUs:



In the current cycle, the cyclic OB used here (e.g. OB 1) will be interrupted by a time error (e.g. OB 80). Following the cyclic OB, the next cycle OB 201 is processed.

Maximum cycle time

The operating system monitors the execution time of the cyclic program for a configurable upper limit known as the maximum cycle time. You can restart this time monitoring at any point in your program by calling the RE_TRIGR instruction.

If the cyclic program exceeds the maximum cycle time, the operating system will attempt to start the time error OB (OB 80). If the OB is not available, the CPU ignores the overshoot of the maximum cycle time.

In addition to monitoring the runtime for overshooting of the maximum cycle time, adherence to a minimum cycle time is guaranteed. To do this, the operating system delays the start of the new cycle until the minimum cycle time has been reached. During this waiting time, new events and operating system services are processed.

If the maximum cycle time is exceeded a second time, for example while the time error OB is being processed (2xMaxCycleTime error), the CPU changes to STOP mode.

Cycle loading by communications

Function

The cycle time of the CPU can be extended due to communications processes. These communications processes include for example:

- Transferring data to another CPU
- Loading of blocks initiated by a programming device

You can control the duration of these communications processes to some extent using the CPU parameter "Cycle load due to communication".

In addition to communications processes, test functions also extend the cycle time. The "Cycle load due to communication" parameter can be used to influence the duration.

How the parameter works

You use the "Cycle load due to communication" parameter to enter the percentage of the overall CPU processing capacity that can be available for communications processes. This CPU processing capacity is now available at all times for communication. This processing capacity can be used for program execution when not required for communication.

Effect on the actual cycle time

The "Cycle load due to communication" parameter can be used to extend the cycle time of the cyclic organization block (e.g., OB 1) by a factor calculated according to the following formula:

$$\frac{100}{100 - \text{"Cycle load due to communication"}}$$

The formula does not take into account the effect of asynchronous events such as hardware interrupts or cyclic interrupts on the cycle time.

If the cycle time is extended due to communication processes, more asynchronous events may occur within the cycle time of the cyclic organization block. This extends the cycle still further. The extension depends on how many events occur and how long it takes to process them.

Example 1 – no additional asynchronous events:

If the "Cycle load due to communication" parameter is set to 50%, this can cause the cycle time of the cyclic organization block to increase by up to a factor of 2.

Example 2 – additional asynchronous events:

For a pure cycle time of 500 ms, a communication load of 50% can result in an actual cycle time of up to 1000 ms, provided that the CPU always has enough communications jobs to process. If, parallel to this, a cyclic interrupt with 20 ms processing time is executed every 100 ms, this cyclic interrupt would extend the cycle by a total of 5*20 ms = 100 ms without communication load. That is, the actual cycle time would be 600 ms. Because a cyclic interrupt also interrupts communications, it affects the cycle time by adding 10 * 20 ms at 50 % communication load. That is, in this case, the actual cycle time amounts to 1200 ms instead of 1000 ms.

Note

Observe the following:

- Check the effects of changing the value of the "Cycle load due to communication" parameter while the system is running.
- You must always consider the communication load when setting the minimum cycle time as time errors will otherwise occur.

Recommendations

- Increase this value only if the CPU is used primarily for communication purposes and the user program is not time critical.
- In all other situations you should only reduce this value.

Time-of-day functions

Basic principles of time of day functions

All S7-1200 CPUs are equipped with an internal clock. The backup supports the display of the correct time for up to 10 hours if the power supply is interrupted.

Time-of-day format

The clock always shows the time of day with a resolution of 1 millisecond and the date including the day of the week. The time adjustment for daylight-saving time is also taken into account.

Setting and reading the time of day

Setting and reading the time with instructions

You can set, start and read the time-of-day and date on the CPU clock with the following instructions in the user program:

- Set the time-of-day: "WR_SYS_T"
- Read time of day "RD_SYS_T"
- Read local time "RD_LOC_T"
- Set time zone "SET_TIMEZONE"

Manual setting

You can also read and set the time-of-day manually in the online and diagnostics view under "Functions > Set time-of-day".

See also

WR_SYS_T: Set time-of-day (Page 4508)

RD_SYS_T: Read time-of-day (Page 4509)

RD_LOC_T: Read local time (Page 4510)

SET_TIMEZONE: Set time zone (Page 4513)

Assigning the clock parameters

Clock parameters

The clock parameters allow you to make the following settings:

- Enable time synchronization via NTP server
  Select this check box if you want the internal clock to be synchronized using the NTP synchronization mode.
- Network time server
  The IP addresses of up to four NTP servers need to be configured.
- Update interval
  The update interval defines the interval between time queries.

See also

Time-of-day synchronization in NTP mode - parameter assignment (Page 1928)

High-speed counters

General information on high-speed counters

Introduction

High-speed counters are typically used to drive counting mechanisms in which a shaft turning at a constant speed is equipped with an incremental step encoder. The incremental step encoder ensures a certain number of count values per rotation and a reset pulse once per rotation. The clock memory bit(s) and the reset pulse of the incremental step encoder supply the inputs for the high-speed counter.

The various S7-1200 CPUs have differing numbers of high-speed counters available:

S7-1200 CPU	Number of HSCs	HSC designation
CPU 1211C	3 (with digital signal board 4)*	HSC1…3 (and HSC5)*
CPU 1212C	4 (with digital signal board 5)*	HSC1…4 (and HSC5)*
CPU 1214C CPU 1215C CPU 1217C	6	HSC1…6

* with DI2/DO2 signal board

How it works

The first of several default values is loaded on the high-speed counter. The required outputs are enabled for the time during which the current value of the counter is lower than the default value. The counter is set up so that an interrupt occurs if the current value of the counter is equal to the default value or when the counter is reset.

If the current value is equal to the default value and an interrupt event results, a new default value is loaded and the next signal state is set for the outputs. If an interrupt event occurs because the counter is reset, the first default value and the first signal states of the outputs are set and the cycle repeated.

Since the interrupts occur much less frequently than the high-speed counter counts, a precise control of the fast operations can be implemented with only a slight influence on the overall cycle of the automation system. Since you can assign specific interrupt programs to interrupts, each new default can be loaded in a separate interrupt program allowing simple control of the state.

Note

You can also process all interrupt events in a single interrupt program.

Count algorithms of the various counters

All counters work in the same way, however some high-speed counters do not support all count algorithms. There are four basic count algorithms:

- Single-phase counter with internal direction control
- Single-phase counter with external direction control
- 2-phase counter with 2 clock inputs
- A/B counter

Each high-speed counter can be used with or without a reset input. If the reset input is activated, this resets the current value. The current value remains reset until the reset input is deactivated.

See also

Configuring high-speed counters (Page 1597)

Interdependency of the counter mode and counter inputs (Page 1595)

Interdependency of the counter mode and counter inputs

General information on counter mode and counter inputs

You can assign not only the counter modes and counter inputs to the high-speed counters but also functions such as clock pulse generator, direction control, and reset. The following rules apply:

- An input cannot be used for two different functions.
- If an input is not required by the current counter mode of the defined high-speed counter, it can be used other purposes.

If, for example, you set HSC1 to counter mode 1, in which inputs I0.0 and I0.3 are required, you can use I0.1 for edge interrupts or for HSC2.

If, for example, you set HSC1 and HSC5, inputs I0.0 (HSC1) and I1.0 (HSC5) are always used with the counting and frequency counter modes. As a result, these two inputs are not available for any other functions when counters are operated.

You have additional inputs available if you use a digital signal board.

Overview of the interdependency of counter mode and counter inputs

Counter mode	Description	Inputs		
	HSC1	I0.0 (CPU) I4.0 (signal board)	I0.1 (CPU) I4.1 (signal board)	I0.3 (CPU) I4.3 (signal board)
	HSC2	I0.2 (CPU) I4.2 (signal board)	I0.3 (CPU) I4.3 (signal board)	I0.1 (CPU) I4.1 (signal board)
	HSC3*	I0.4 (CPU)	I0.5 (CPU)	I0.7 (CPU)
	HSC4 (CPU 1212/14/15/17C only)	I0.6 (CPU)	I0.7 (CPU)	I0.5 (CPU)
	HSC5 (CPU 1214/15/17C only)**	I1.0 (CPU) I4.0 (signal board)	I1.1 (CPU) I4.1 (signal board)	I1.2 (CPU) I4.3 (signal board)
	HSC6 (CPU 1214/15/17C only)**	I1.3 (CPU)	I1.4 (CPU)	I1.5 (CPU)
Counting / frequency	Single-phase counter with internal direction control	Clock pulse generator	-	-
Counting				Resetting
Counting / frequency	Single-phase counter with external direction control	Clock pulse generator	Direction	-
Counting				Resetting
Counting / frequency	2-phase counter with 2 clock inputs	Clock pulse generator forwards	Clock pulse generator backwards	-
Counting				Resetting
Counting / frequency	A/B counter	Clock pulse generator A	Clock pulse generator B	-
Counting				Resetting
Motion axis	Pulse generators PWM/PTO	HSC1 and HSC2 support the motion axis count mode for the PTO1 and PTO2 pulse generators: • For PTO1, HSC1 evaluates the Q0.0 outputs for the number of pulses. • For PTO2, HSC2 evaluates the Q0.2 outputs for the number of pulses. Q0.1 is used as the output for the motion direction.		

* HSC3 can only be used for CPU 1211 without a reset input

** HSC5 can be also be used for CPU 1211/12 if a DI2/DO2 signal board is used

See also

General information on high-speed counters (Page 1594)

Configuring high-speed counters (Page 1597)

Configuring high-speed counters

Requirement

An S7-1200 CPU has been inserted in the hardware configuration.

Procedure

To configure a high-speed counter, follow these steps:

1. Select an S7-1200 CPU in the device or network view.
2. Click on the required high-speed counter under "Properties > High-speed counter" in the inspector window:
   - CPU 1211C: HSC1 to HSC3 (also HSC5 with a DI2/DO2 signal board)
   - CPU 1212C: HSC1 to HSC4 (also HSC5 with a DI2/DO2 signal board)
   - CPU 1214C / 1215C / 1217C: HSC1 to HSC6
3. Enable the high-speed counter in the "General" parameter group using the relevant check box.

   Note

   If you use a CPU 1211C or CPU 1212C with a DI2/DO2 signal board, you can also enable the high-speed counter HSC5.

   Note

   If you activate the pulse generators and operate them as PTO1 or PTO2, they use the associated high-speed counter HSC1 or HSC2 with "Motion axis" counting mode to evaluate the hardware outputs. If you configure high-speed counter HSC1 or HSC2 for other counting tasks, these cannot be used by pulse generator PTO1 or PTO2, respectively.

   If required, you can enter a name and a comment for the high-speed counter here.
4. Define the functionality of the high-speed counter in the "Function" parameter group:
   - Count mode: Select what you want to be counted from the drop-down list.
   - Operating phase: Select the count algorithm from the drop-down list.
   - Input source: Select the on-board CPU inputs or the inputs of an optional digital signal board as the input source for the count pulses from the drop-down list.
   - Count direction is specified by: If you have selected a single-phase operating phase, open the drop-down list and select whether the count direction is set internally by an SFB parameter of the user program or externally via a digital input.
   - Initial count direction: If the user program is set as the internal direction control for the count direction, you can select the count direction at the start of counting from the drop-down list.
   - Frequency meter period: If frequency is set as the count mode, you can select the duration of the frequency meter periods in the drop-down list.

5. Specify the initial values and reset condition of the high-speed counter in the "Reset to initial values" parameter group:
   - Initial counter value: Enter a start value for the high-speed counter.
   - Initial reference value: Enter a maximum value for the high-speed counter.

   Here, you can also specify whether the high-speed counter will use a reset input and the set the corresponding signal level for the reset input from the drop-down list.
6. Configure the reaction of the high-speed counter to certain events in the "Event configuration" parameter group. The following events can trigger an interrupt:
   - The counter value matches the reference value.
   - An external reset event was generated.
   - A change of direction was triggered.

   Enable an interrupt reaction using the check box, enter a name and select a hardware interrupt for the interrupt from the drop-down list.
7. Assign the start address for the high-speed counter in the "I/O/Diagnostics addresses" parameter group.

Note

In the "Hardware inputs" parameter group, you can only see which hardware inputs and values are being used for the clock, direction determination, reset pulse, and maximum count speed.

Result

You have now adapted the parameters of the high-speed counter to the requirements of your project.

See also

General information on high-speed counters (Page 1594)

Interdependency of the counter mode and counter inputs (Page 1595)

Point-to-point communication

Overview of point-to-point communication

PtP communication is communication via a serial interface that uses standardized UART data transmission (Universal Asynchronous Receiver/Transmitter). The S7-1200 uses communications modules with an RS-232 or RS-485 interface to establish PtP communication.

Functions of point-to-point communication

Point-to-point communication (PtP) allows numerous applications:

- Direct transmission of information to an external device, for example a printer or a barcode reader
- Reception of information from external devices such as barcode readers, RFID readers, cameras and third-party optical systems as well as many other devices.
- Exchange of information with third-party devices, for example GPS devices, radio modems and many others

The Freeport protocol

The S7-1200 supports the Freeport protocol for character-based serial communication. Using Freeport communication, the data transmission protocol can be configured entirely by the user program.

Siemens provides libraries with Freeport communication functions that you can use in your user program:

- USS Drive protocol
- Modbus RTU Master protocol
- Modbus RTU Slave protocol

See also

Configuring a communications port (Page 1600)

Using RS-232 and RS-485 communications modules

Communications modules with RS-232 and RS-485 interfaces

In an S7-1200 CPU, you can use two different communications modules:

- RS-232 communications module
- RS-485 communications module

The communications modules can be connected to the S7-1200 CPU via the I/O channel on the left. You can plug in up to three different modules.

Properties of the communications modules

The communications modules have the following features:

- Support of the Freeport protocol
- Configuration by the user program with the aid of expanded instructions and library functions

Configuring a communications port

Configuring a communications port

After you have inserted a communications module with an RS-232 or RS-485 interface, you then set the interface parameters. You set the parameters for the interface either in the properties of the interface or you control the interface parameters from the user program using the PORT_CFG instruction. The following description relates to the graphic configuration.

Note

If you use the user program to change the port setting, the settings of the graphic configuration are overwritten.

You should also keep in mind that the settings made by the user program are not retained if there is a power down.

Requirement

- A communications module is already plugged in.
- You are in the device view.

Procedure

To configure the communications port, proceed as follows:

1. Select the interface in the graphic representation in the device view.
   The properties of the interface are displayed in the Inspector window.
2. Select the "Port configuration" group in the area navigation of the Inspector window.
   The settings of the port are displayed.
3. From the "Transmission speed" drop-down list, select the speed for the data transmission.
   With user-programmed communication, remember the influence of the transmission speed on the changeover time.
4. From the "Parity" drop-down list, select the type of detection of bad information words.
5. Using the "Data bits" drop-down list, decide whether a character consists of eight or seven bits.
6. From the "Stop bit" drop-down list, select how many bits will identify the end of a transmitted word.

7. From the "Flow control" drop-down list, select the method for ensuring a trouble-free data stream between sender and receiver. This parameter can only be set for the RS-232 interface.
   - Enter a HEX value in the "XON character" box that will cause the transmission of the message to be continued when it is detected. This parameter can only be set for software-controlled data flow control.
   - Enter a HEX value in the "XOFF character" box that will cause the transmission of the message to be suspended for the set wait time. This parameter can only be set for software-controlled data flow control.
8. In the "Wait time" box, enter a wait time in ms that must be kept to after the end of the message before the next transmission can start.

Note

You can configure the interface in the network view as well. To do so, you must first select the communication module in the tabular network view and then select the interface in the Inspector window. Then you can continue as described above.

See also

Setting data flow control (Page 1601)

Setting data flow control

Data flow control

Data flow control is a method that ensures a balanced send and receive behavior. In the ideal situation, the intelligent control does not allow data to be lost. It ensures that a device does not send more information than the receiving partner can process.

There are two methods of data flow control:

- Hardware-controlled data flow control
- Software-controlled data flow control

With both methods, the DSR signals of the communications partners must be active at the beginning of transmission. If the DSR signals are inactive, the transmission is not started.

The RS-232 communications module can handle both methods. The RS-485 communications module does not support data flow control.

Hardware-controlled data flow control

Hardware-controlled data flow control uses the request to send (RTS) and clear to send (CTS) signals. With the RS-232 communications module, the RTS signal is transmitted via the output by pin 7. The CTS signal is received via pin 8.

If hardware-controlled data flow control is enabled, the RTS signal is then always set to activated when data is sent. At the same time, the CTS signal is monitored to check whether

the receiving device can accept data. If the CTS signal is active, the module can transfer data until the CTS signal becomes inactive. If the CTS signal is inactive, the data transfer must be suspended for the set wait time. If the CTS signal is still inactive after the set wait time, the data transfer is aborted and an error is signaled back to the user program.

Data flow control using hardware handshaking

If data flow control is controlled by hardware handshaking, the sending device sets the RTS signal to active as default. A device such as a modem can then transfer data at any time. It does not wait for the CTS signal of the receiver. The sending device itself monitors it own transmission by sending only a limited number of frames (characters), for example to prevent overflow of the receive buffer. If there is nevertheless an overflow, the transferring device must hold back the message and signal an error back to the user program.

Software-controlled data flow control

Software-controlled data flow control uses certain characters within the messages and these control the transfer. These characters are ASCII characters selected for XON and XOFF.

XOFF indicates when a transmission must be suspended. XON indicates when a transmission can be continued.

If the sending device receives the XOFF character, it must suspend sending for the selected wait time. If the XON character is sent after the selected wait time, the transfer is continued. If no XON character is received after the wait time, an error is signaled back to the user program.

Software data flow control requires full duplex communication because the receiving partner needs to send the XON character during the ongoing transfer.

See also

Configuring a communications port (Page 1600)

Configuration of message transfer

User-programmed communication

You can control the data traffic between a communications module and a device connected externally via the serial interface using your own mechanisms. If you want to do this, you will need to define a communications protocol yourself. In freely programmable communication, ASCII and binary protocols are supported for message transfer.

Within the communications protocol, you will need to specify the criteria by which the start and end of a transferred message can be recognized in the data stream.

User-programmed communication can only be activated in RUN mode. If there is a change to STOP mode, the user-programmed communication is stopped.

Specifying the communications protocol

You can specify the communications protocol as follows:

- With the user program
  - The behavior when sending data is controlled by the SEND_CFG instruction.
  - The behavior when receiving data is controlled by the RCV_CFG instruction.
- Using parameter settings set graphically in the Inspector window

Note

If you change the communications protocol from the user program, the settings of the graphic configuration are overwritten.

You should keep in mind that the settings made by the user program are not retained if there is a power down.

See also

User-programmed communication with RS-232 devices (Page 1603)

Making the settings for sending (Page 1605)

Specifying the start of the message (Page 1605)

Specifying the end of the message (Page 1606)

User-programmed communication with RS-232 devices

RS-232/PIP multi-master cable and user-programmed communication with RS-232 devices

Using the RS-232/PIP multi-master cable and user-programmed communication, you can connect a wide variety of RS-232-compliant devices to the communications modules of the S7-1200. The cable must, however, be set to the "PIP/user-programmed communication" mode.

Settings on the cable

The switches on the cable must be set as follows:

- Switch 5 must be set to 0
- Switch 6 sets either the local mode (DCE) or the remote mode (DTE):
  - Switch set to 0 for the local mode
  - Switch set to 1 for the remote mode

Changing over between send and receive mode

The RS-232/PIP multi-master cable is in send mode when data is sent from the RS-232 interface to the RS-485 interface. The cable is in receive mode when it is idle or when data is sent from the RS-485 interface to the RS-232 interface. The cable changes from receive to send mode immediately when it detects characters on the RS-232 send line.

Supported transmission speeds

The RS-232/PIP multi-master cable supports transmission rates between 1200 baud and 115.2 kbaud. The RS-232/PIP multi-master cable can be set to the required transmission rate using the DIP switch on the PC/PIP cable.

The following table shows the switch settings for the various transmission speeds:

Transmission speed	Switchover time	Settings (1 = up)
115200 bps	0.15 ms	110
57600 bps	0.3 ms	111
38400 bps	0.5 ms	000
19200 bps	1.0 ms	001
9600 bps	2.0 ms	010
4800 bps	4.0 ms	011
2400 bps	7.0 ms	100
1200 bps	14.0 ms	101

The cable returns to receive mode when the RS-232 send line is idle for a certain time that is defined as the changeover time of the cable. The set transmission speed influences the changeover time as shown in the table.

Influence of the changeover time

When working with an RS-232/PIP multi-master cable in a system in which user-programmed communication is used, the program must take into account the changeover time for the following reasons:

- The communications module reacts to messages sent by the RS-232 device.
  Once the communications module has received a request from the RS-232 device, it must delay the reaction message for a period that is equal to or longer than the changeover time of the cable.
- The RS-232 device reacts to messages sent by the communications module.
  Once the communications module has received a reaction message from the RS-232 device, it must delay the next request message for a period that is equal to or longer than the changeover time of the cable.

In both situations, the RS-232-PIP multi-master cable has enough time to change from send to receive mode so that the data can be sent from the RS-485 interface to the RS-232 interface.

See also

Configuration of message transfer (Page 1602)
Making the settings for sending (Page 1605)
Specifying the start of the message (Page 1605)
Specifying the end of the message (Page 1606)

Making the settings for sending

Sending messages

You can program pauses between individual messages.

The following table shows which pauses can be set:

Parameter	Definition
RTS ON delay	You can set the time that must elapse after the send request RTS (request to send) before the actual data transfer starts.
RTS OFF delay	You can set the time that must elapse after the complete transfer before RTS signal is deactivated.
Send pause at the start of the message	You can specify that a pause is sent at the start of every message transfer when the RTS ON delay has elapsed. The pause is specified in bit times.
Send Idle Line after a pause	You can make a setting so that following a selected pause at the start of the message, the "Idle Line" signal is output to signal that the line is not in use. To enable the parameter, "Send pause at message start" must be set. The duration of the "Idle Line" signal is specified in bit times.

See also

Specifying the start of the message (Page 1605)
Specifying the end of the message (Page 1606)
User-programmed communication with RS-232 devices (Page 1603)

Specifying the start of the message

Recognizing the start of the message

To signal to the receiver when the transfer of a message is completed and when the next message transfer starts, criteria must be specified in the transmission protocol to identify the end and start of a message.

If a criterion is met that indicates the start of a message, the receiver starts searching the data stream for criteria that mean the end of the message.

There are two different methods for identifying the start of a message:

- Starting with any character:
  Any character can defined the start of a message. This is the default method.
- Starting with a specific condition:
  The start of a message is identified based on selected conditions.

Conditions for detecting the start of a message

The following table shows the various options for defining the start of a message:

Parameter	Definition
Recognize start of message by line break	The receiver recognizes a line break when the received data stream is interrupted for longer than one character. If this is the case, the start of the message is identified by the line break.
Recognize start of message by idle line	The start of a message is recognized when the send transmission line is in the idle state for a certain time (specified in bit times) followed by an event such as reception of a character.
Recognize start of message with individual characters	The start of a message is recognized when a certain character occurs. You can enter the character as a HEX value.
Recognize start of message by a character string	The start of a message is detected when one of the specified character sequences arrives in the data stream. You can specify up to four character sequences each with up to five characters.

The individual conditions can be logically linked in any way.

See also

Making the settings for sending (Page 1605)

User-programmed communication with RS-232 devices (Page 1603)

Specifying the end of the message

Recognizing the end of the message

To signal to the receiver when the transfer of a message is completed and when the next message transfer starts, criteria must be specified in the transmission protocol to identify the end and start of a message.

In total, there are six different methods of recognizing the end of a message and these can all be logically linked in any way. The following table shows the various possible setting options:

Parameter	Definition
Recognize end of message by message timeout	The end of a message is recognized automatically when a selected maximum duration for a message is exceeded. Values from 0 to 65535 ms can be set.
Recognize end of message by reply timeout	The end of a message is recognized when there is no reply within a set time after transferring data. Values from 0 to 65535 ms can be set.

Parameter	Definition
Recognize end of message by timeout between characters	The end of a message is detected when the time between two characters specified in bit times is exceeded. Values from 0 to 2500 bit times can be set. The S7-1200 CPU only accepts a maximum time of eight seconds even if the value that is set results in a duration of more than eight seconds.
Recognize end of message by maximum length	The end of a message is recognized when the maximum length of a message is exceeded. Values from 1 to 1023 characters can be set.
Read message length from message	The message itself contains information about the length of the message. The end of a message is reached when the value taken from the message is reached. Which characters are used for the evaluation of the message length is specified with the following parameters: • Offset of the length field in the message The value decides the position of the character in the message that will be used to indicate the message length. Values from 1 to 1022 characters can be set. • Size of the length field This value specifies how many characters starting at the first evaluation position will be used to indicate the message length. Values of 0, 1, 2 and 4 characters can be set. • The data following the length field (does not belong to the message length) The value specifies the number of bytes after the length field that must be ignored in the evaluation of the message length. Values from 0 to 255 characters can be set.
Recognize message end with a character sequence	The end of a message is detected when the specified character sequence arrives in the data stream. You can define up to five characters in the character string that are to be checked. If the specified characters appear at the correct location in the message, the message end is recognized. To recognize the message end when character 1 and character 3 have a certain value, for example, you have to activate the check box for character 1 and character 3 and enter a character value.

See also

Making the settings for sending (Page 1605)

User-programmed communication with RS-232 devices (Page 1603)

Enabling system memory

System memory

A system memory is a bit memory with defined values.

You decide which memory byte of the CPU will become the system memory byte when assigning the system memory parameters.

Benefits

You can use system memory in the user program, for example to run program segments in only the first program cycle after start-up. Two system memory bits are constant 1 or constant 0.

Bits of the system memory bytes

The following table shows the meaning of the system memory:

Bit of the system memory bytes	7	6	5	4	3	2	1	0
Meaning	Reserved (=0)	Reserved (=0)	Reserved (=0)	Reserved (=0)	=0	=1	=1 with change to the diagnosis status	=1 in first program cycle after startup, otherwise 0

Note

The selected memory byte cannot be used for intermediate storage of data.

Using clock memory

Clock memory

A clock memory is a bit memory that changes its binary status periodically in the pulse-no-pulse ratio of 1:1.

You decide which memory byte of the CPU will become the clock memory byte when assigning the clock memory parameters.

Benefits

You can use clock memory, for example, to activate flashing indicator lamps or to initiate periodically recurring operations such as recording of actual values.

Available frequencies

Each bit of the clock bit memory byte is assigned a frequency. The following table shows the assignment:

Bit of the clock memory byte	7	6	5	4	3	2	1	0
Period (s)	2.0	1.6	1.0	0.8	0.5	0.4	0.2	0.1
Frequency (Hz)	0.5	0.625	1	1.25	2	2.5	5	10

Note

Clock memory runs asynchronously to the CPU cycle, i.e. the status of the clock memory can change several times during a long cycle.

The selected memory byte cannot be used for intermediate storage of data.

Protection

Setting options for the protection level (FW V1 to V3)

Protection level

The following section describes how to use the various protection levels of the S7-1200 CPUs V1 to V3.

Effects of the protection level setting

You can choose between the following protection levels:

- No protection: This corresponds to the default behavior. You cannot enter a password. Read and write access is always permitted.
- Write protection: Only read-only access is possible. You cannot change any data on the CPU and cannot load any blocks or a configuration. HMI access and communication between CPUs are excluded from the write protection. Assignment of a password is required to select this protection level.
- Write/read protection: No write or read access is possible in the "Accessible devices" area or in the project for devices that are switched online. Only the CPU type and the identification data can be displayed in the project tree under "Accessible devices". Display of online information or blocks under "Accessible devices", or in the project for devices interconnected online, is possible.
  HMI access and communication between CPUs are excluded from the write protection. Assignment of a password is required to select this protection level.

Behavior of a password-protected CPU during operation

The CPU protection takes effect after the settings are downloaded to the CPU.

Validity is checked before the online function is executed. If password protection is in place, you are prompted to enter a password.

Example: The module was assigned write protection and you want to execute the "Modify tags" function. This requires write access; therefore, the assigned password must be entered to execute the function.

The functions protected by a password can only be executed by one programming device/PC at any one time. Another programming device/PC cannot log on with a password.

Access authorization to the protected data is in effect for the duration of the online connection or until the access authorization is manually rescinded with "Online > Delete access rights". Access authorization will also expire when the project is closed.

Note

You can not restrict functions for process control, monitoring, and communications.

Some functions are still protected due to their use as online data. RUN/STOP in the "Online Tools" task card or "Set the time" in the diagnostics and online editor is therefore write-protected.

Setting options for the protection (FW as of V4)

Protection level

The following section describes how to use the various access levels of the S7-1200 CPUs as of V4.

S7-1200 CPUs provide various access levels to limit the access to specific functions.

The parameters for the access levels are assigned in a table. The green checkmarks in the columns to the right of the respective access level specify which operations are possible without knowing the password of this access level. If you want to use the functions of check boxes that are not selected, a password has to be entered.

NOTICE

Configuring an access level does not replace know-how protection

Configuring access levels prevents unauthorized changes to the CPU by restricting download privileges. However, blocks on the memory card are not write- or read-protected. Use know-how protection to protect the code of blocks on the memory card.

Default characteristics

The default access level is "Full access (no protection)". Every user can read and change the hardware configuration and the blocks. A password is not set and is also not required for online access.

The access levels in detail

With an S7-1200 CPU, you can configure the following access levels:

- Full access (no protection): The hardware configuration and the blocks can be read and changed by all users.
- Read access: With this access level, only read access to the hardware configuration and the blocks is possible without entering a password - meaning that you can load the hardware configuration and blocks into the programming device. In addition, HMI access and access to diagnostics data is possible.
  You cannot load blocks or a hardware configuration into the CPU without entering the password. Moreover, writing test functions and firmware updates are **not** possible without a password.
- HMI access: With this access level, only HMI access and access to diagnostics data is possible without entering the password.
  Without entering the password, you can neither load blocks and hardware configuration into the CPU, nor load blocks and hardware configuration from the CPU into the programming device. In addition, the following is **not** possible without a password: Writing test functions, changing the operating state (RUN/STOP) and firmware updates.
- No access (complete protection): When the CPU is completely protected, no read or write access to the hardware configuration and the blocks is possible. HMI access is also not possible. The server function for PUT/GET communication is disabled in this access level (cannot be changed).
  Authorization with the password again provides you full access to the CPU.

Behavior of a password-protected module during operation

The CPU protection takes effect after the settings are downloaded to the CPU.

Validity is checked before the online function is executed. If password protection is in place, you are prompted to enter a password.

Example: The module was configured with read access and you want to execute the "Modify tags" function. This requires write access; therefore, the assigned password must be entered to execute the function.

The functions protected by a password can only be executed by one programming device/PC at any one time. Another programming device/PC cannot log on.

Access authorization to the protected data is in effect for the duration of the online connection or until the access authorization is manually rescinded with "Online > Delete access rights".

Each access level allows unrestricted access to certain functions without entering a password, for example, identification using the "Accessible devices" function.

Configuring access levels

The following section describes how to configure an access level and enter passwords for an S7-1200 CPU as of V4.

For an S7-1200 CPU, you can enter multiple passwords and thereby set up different access rights for individual user groups.

The passwords are entered in a table in such a way that exactly one access level is assigned to each password.

The effect of the password is visualized in the "Access" column and explained in the text below the table.

Example

You select the "No access (complete protection)" access level for a standard CPU (in other words, not an F-CPU) and enter a separate password for each of the access levels that lie above it in the table.

For users who do not know any of the passwords, the CPU is completely protected. Not even HMI access is possible.

For users who know one of the set passwords, the effect depends on the table row in which the password occurs:

- The password in row 1 (Full access (no protection)) allows access as if the CPU were completely unprotected. Users who know this password have unrestricted access to the CPU.
- The password in row 2 (read access) allows access as if the CPU were write-protected. Users who know this password have read-only access to the CPU.
- The password in row 3 (HMI access) allows access as if the CPU were write-protected and read-protected so that only HMI access is possible for users who know this password.

Procedure

To configure the access levels of an S7-1200 CPU, follow these steps:

1. Open the properties of the module in the inspector window.
2. Open the "Protection" entry in the area navigation.
   A table with the possible access levels appears in the inspector window.

Protection level	Access			Access permission	
Protection	HMI	Read	Write	Password	Confirmation
Full access (no protection)	✓	✓	✓		
Read access	✓	✓			
HMI access	✓				
No access (complete protection)					

Figure 8-52 Access protection

3. Activate the required access level in the first column of the table. The green checkmarks in the columns to the right of the respective protection level show you which operations are still available without entering the password.

4. In the "Password" column, specify a password for full access in the first row. In the "Confirmation" column, enter the selected password again to protect against incorrect entries.
   Ensure that the password is sufficiently secure, in other words, that is does not follow a machine-recognizable pattern!
   You must enter a password in the first row "Full access (no protection)". This enables unrestricted access to the CPU for those who know the password, regardless of the selected protection level.
5. Assign additional passwords as needed to other access levels if the selected protection level allows you to do so.
6. Download the hardware configuration so that the access level takes effect.

Result

The hardware configuration and the blocks are protected against unauthorized access according to the set access level. If an operation cannot be executed without a password due to the set access level, a dialog for entering a password is displayed.

Restriction of communication services

Introduction

The CPU can be the server for a number of communication services. This means that other communication participants can access CPU data even if you have not configured and programmed connections for the CPU.

The local CPU as a server thus does not have the possibility to control communication to the clients.

The parameter "Connection mechanisms" in the "Protection" area of the CPU parameters is used to specify whether this type of communication is permitted or not for the local CPU during operation.

Permit access with PUT/GET communication from remote partners

By default, the "Permit access with PUT/GET communication from remote partners (...)" option is disabled. In this case, read and write access to CPU data is only possible for communication connections that require configuration or programming both for the local CPU and for the communication partner. Access through BSEND/BRCV instructions is possible, for example.

Connections for which the local CPU is only a server (meaning that no configuration/ programming of the communication with the communication partner exists at the local CPU), are therefore not possible during operation of the CPU, for example,

- For PUT/GET, FETCH/WRITE or FTP access via communication modules
- For PUT/GET access from other S7 CPUs
- For HMI access that is realized via PUT/GET communication

If you want to allow access to CPU data from the client side, meaning that you do not want to restrict the communication services of the CPU, activate the "Permit access with PUT/GET communication from remote partners" option.

Organization blocks

Organization blocks for startup

Description

You can determine the boundary conditions for the startup characteristics of your CPU, for example, the initialization values for "RUN". To do this, write a startup program. The startup program consists of one or more startup OBs (OB numbers 100 or >= 123).

The startup program is executed once during the transition from "STOP" mode to "RUN" mode. Current values from the process image of the inputs are not available for startup program, nor can these values be set.

After the complete execution of the startup OBs, the process image of the inputs is read in and the cyclic program is started.

There is no time limit for executing the startup routine. Therefore the scan cycle monitoring time is not active. Time-driven or interrupt-driven organization blocks cannot be used.

Start information

A startup OB has the following start information:

Tag	Data type	Description
LostRetentive	BOOL	= 1, if retentive data storage areas have been lost
LostRTC	BOOL	= 1, if realtime clock has been lost

See also

Organization blocks (OB) (Page 2170)

Events and OBs (Page 1578)

Organization blocks for cyclic program execution

Introduction

For the program execution to start, at least one program cycle OB must be present in the project. The operating system calls this program cycle OB once in each cycle and thereby starts the execution of the user program. You can use multiple OBs (OB numbers >= 123). When multiple program cycle OBs are used, these are called in one after the other in the order of their OB numbers. The program cycle OB with the lowest OB number is called first.

The program cycle OBs have the priority class 1. This corresponds to the lowest priority of all OBs. The cyclic program can be interrupted by events of any other event class.

Programming cyclic program execution

You program cyclic program execution by writing your user program in the cycle OBs and the blocks that they call.

The first cyclic program execution begins as soon as the startup program has ended without errors. The cycle restarts after the end of each cyclic program execution.

Sequence of cyclic program execution

One cycle of the program execution encompasses the following steps:

1. The operating system starts the maximum cycle time.
2. The operating system writes the values from the process image output to the output modules.
3. The operating system reads out the state of the inputs of the input modules and updates the process image input.
4. The operating system processes the user program and executes the operations contained in the program.
5. At the end of a cycle, the operating system executes any tasks that are pending, for example, loading and deleting blocks or calling other cycle OBs.
6. Finally, the CPU returns to the start of the cycle and restarts the scan cycle monitoring time.

See also: Auto-Hotspot

Options for interrupting

Cyclic program execution can be interrupted by the following events:

- Interrupt
- A STOP command, triggered by
  - Operation of the programming device
  - "STP" instruction
- Supply voltage failure
- Occurrence of a device fault or program error

Start information

- None
- Optimized start information:

Name	Data type	Meaning
first_scan	BOOL	= TRUE in the first call of this OB: • Transition from STOP or HOLD to RUN • After reloading
retentivity	BOOL	= TRUE, if retentive data are available

See also

Organization blocks (OB) (Page 2170)

Events and OBs (Page 1578)

Organization blocks for interrupt-driven program execution

Organization blocks for time-of-day interrupt

Function

Organization blocks for time-of-day interrupt (OB number >=123) can be processed as follows:

- One time to a preset time (date with time of day)
- Periodically with preset start time and the following intervals:
  - Every minute
  - Hourly
  - Daily
  - Weekly
  - Monthly
  - Yearly
  - End of month

Time-of-day interrupt-OBs are therefore used to run parts of the user program on a time-controlled basis.

Status for time-of-day interrupt

The following table contains the possible status of a time-of-day interrupt as well as it's meaning.

Status	Meaning
Cancelled	The one-time processing has already taken place, or the start event of an not yet processed time-of-day interrupt has been deleted with the extended instruction CAN_TINT.
Set	You have scheduled the time or start time for processing.
Activated	You have scheduled whether processing takes place one time or periodically, and in the case of periodic processing, you have scheduled the interval.

Rules for time-of-day interrupts

The following rules apply to the use of time-of-day interrupts:

- A time-of-day interrupt can only be processed if it is set and activated, and a corresponding organization block exists in the user program.
- The start times of periodic time-of-day interrupts must correspond to a real date. For example, it is not possible to repeat an organization block monthly which first occurs on January 31st. In this case, on OB will only be started in the months that have 31 days.
- A time-of-day interrupt activated during startup by extended instruction call ACT_TINTwill not be executed until the startup is complete.
- After each CPU startup, you must reactivate previously set time-of-day interrupts

Setting and activating a time-of-day interrupt OB

Before a time-of-day interrupt can be deleted and the corresponding time-of-day interrupt OB can be processed from the operating system, you must set and activate the interrupt. You have the following options:

Set time-of-day interrupt	Activate time-of-day interrupt
Means of configuring	Means of configuring
Means of configuring	By extended instruction call ACT_TINT
By extended instruction call SET_TINTL	By extended instruction call ACT_TINT

Note

If you configure a time-of-day interrupt in such a way that the corresponding OB is to be processed once, the start time must not be in the past (relative to the real-time clock of the CPU).

If you configure a time-of-day interrupt in such a way that the corresponding OB is to be processed periodically, but the start time is in the past, then the time-of-day interrupt will be processed the next time it is due.

Time-of-day interrupt status query

In order to query the status of the time-of-day interrupt, call the extended instruction QRY_TINT.

Cancelling a time-of-day interrupt

You can cancel a time-of-day interrupt which has not yet been processed with the extended instruction CAN_TINT.

You can reset cancelled time-of-day interrupts with the extended instruction SET_TINTL and activate them with the extended instruction ACT_TINT.

Conditions that effect the time-of-day interrupt OBs

Since a time-of-day interrupt occurs only at specific intervals, certain conditions can affect the function of the associated OBs during execution of your program. The following table shows some of these conditions and describes how these affect the processing of a time-of-day interrupt OB.

Condition	Result
The extended instruction CAN_TINT is called in the user program.	The operating system deletes the start event (date and time) of the time-of-day interrupt. You must reset and reactivate the time-of-day interrupt if you are going to call the corresponding time-of-day interrupt OB again.
By synchronizing or correcting the CPU system clock, the time of day will be set ahead. With this, the start time for a time-of-day interrupt OB will be bypassed.	The operating system calls the time error interrupt OB (OB 80) and records the start event, the number, and the priority of the first bypassed time-of-day interrupt OB in its start information. After completion of the OB 80, the operating system processes the bypassed time-of-day interrupt OB only once.
By synchronizing or correcting the CPU system clock, the time of day will be set back. The corrected clock time is before the start time of an already processed time-of-day interrupt OB.	The time-of-day interrupt OB is repeated.
A time-of-day interrupt OB is still being processed when the start event for its next execution occurs.	The operating system then calls time error interrupt OB 80. The requested OB is processed only after the processing and further execution of the current time-of-day interrupt OB has completed.

Start information

A time-of-day interrupt OB has the following start information:

Tag	Data type	Description
CaughtUp	BOOL	=1, if the OB call is executed because clock time is turned ahead.
SecondTime	BOOL	=1 if the OB is called a second time because clock time is turned back (more exactly, if the planned time of the current OB processing is earlier than or the same time as the planned time for the previous OB processing). Note: SecondTime is only set once.

Organization block for status interrupts

Description

When it receives a status interrupt, the operating system of the S7-1200 CPU calls the status interrupt OB from a DP master or IO controller. This may be the case if a module of a slave changes its operating mode, for example, from "RUN" to "STOP". For more detailed information

on events that trigger a status interrupt, refer to the documentation of the slave or device manufacturer.

Structure of the start information

The status-interrupt OB has the following start information:

Name	Data type	Meaning
LADDR	HW_IO	Hardware address of the component triggering the interrupt
Slot	UINT	Slot number of the component triggering the interrupt
Specifier	WORD	Interrupt specifier from the interrupt frame

See also

Events and OBs (Page 1578)

Organization block for update interrupts

Description

When it receives a status interrupt, the operating system of the S7-1200 CPU calls the update interrupt OB from a DP master or IO controller. This may be the case if you changed a parameter on a slot of a slave or device. For more detailed information on events that trigger an update interrupt, refer to the documentation of the slave or device manufacturer.

Structure of the start information

The update interrupt OB has the following start information:

Name	Data type	Meaning
LADDR	HW_IO	Hardware address of the component triggering the interrupt
Slot	UINT	Slot number of the component triggering the interrupt
Specifier	WORD	Interrupt specifier from the interrupt frame

See also

Events and OBs (Page 1578)

Organization block for manufacturer-specific or profile-specific interrupts

Description

When it receives a manufacturer-specific or profile-specific interrupt from a DP master or IO controller, the operating system of the S7-1200 CPU calls OB 57. For more detailed information

on events that trigger this type of interrupt, refer to the documentation of the slave or device manufacturer.

Structure of the start information

The OB for manufacturer-specific and profile-specific interrupts has the following start information:

Name	Data type	Meaning
LADDR	HW_IO	Hardware address of the component triggering the interrupt
Slot	UINT	Slot number of the component triggering the interrupt
Specifier	WORD	Interrupt specifier from the interrupt frame

See also

Events and OBs (Page 1578)

Organization blocks for time-delay interrupts

Description

A time-delay interrupt OB is started after a configurable time delay of the operating system. The delay time starts after the SRT_DINT instruction is called.

You can use up to four time-delay interrupt OBs or cyclic OBs (OB numbers >= 123) in your program. If, for example, you are already using two cyclic interrupt OBs, you can insert a maximum of two further time-delay interrupt OBs in your program.

You can use the CAN_DINT instruction to prevent the execution of a time-delay interrupt that has not yet started.

Function of time-delay interrupt OBs

The operating system starts the corresponding OB after the delay time, which you have transferred with an OB number and an identifier to the SRT_DINT instruction.

To use a time-delay interrupt OB, you must execute the following tasks:

- You must call the instruction SRT_DINT.
- You must download the time-delay interrupt OB to the CPU as part of your program.

The delay time is measured with a precision of 1 ms. A delay time can immediately start again after it expires.

Time delay interrupt OBs are executed only when the CPU is in the "RUN" mode. A warm restart clears all start events of time-delay interrupt OBs.

The operating system calls the time-delay interrupt OB if one of the following events occurs:

- If the operating system attempts to start an OB that is not loaded and you specified its number when calling the SRT_DINT instruction.
- If the next start event for a time-delay interrupt occurs before the time delay OB has completely executed.

You can disable and re-enable time-delay interrupts using the DIS_AIRT and EN_AIRT instructions.

Note

If you disable an interrupt with DIS_AIRT after executing SRT_DINT, this interrupt executes only after it has been enabled with EN_AIRT. The delay time is extended accordingly.

Start information

- None
- Optimized start information:

Name	Data type	Meaning
sign	WORD	User ID: Input parameter SIGN from the call of the "SRT_DINT" instruction

See also

SRT_DINT: Start time-delay interrupt (Page 4706)

CAN_DINT: Cancel time-delay interrupt (Page 4708)

Organization blocks (OB) (Page 2170)

Events and OBs (Page 1578)

Organization blocks for cyclic interrupts

Description

Cyclic interrupt OBs serve to start program in periodic time intervals independently of the cyclic program execution. The start times of a cyclic interrupt OB are specified using the time base and the phase offset.

The time base defines the intervals at which the cyclic interrupt OB is started and is an integer multiple of the basic clock cycle of 1 ms. The phase offset is the time by which the start time is offset compared with the basic clock cycle. If several cyclic interrupt OBs are being used, you can use this offset to prevent a simultaneous start time if the time bases of the cyclic interrupt OBs have common multiples.

You can specify a time period between 1 ms and 60000 ms as the time base.

You can use up to four cyclic interrupt OBs or time-delay OBs (OB numbers >= 123) in your program. If, for example, you are already using two time-delay interrupt OBs, you can insert a maximum of two further cyclic interrupt OBs in your program.

Note

The execution time of each cyclic interrupt OB must be noticeably smaller than its time base. If a cyclic interrupt OB has not been completely executed but execution is again pending because the cycle clock has expired, the time error OB is started. The cyclic interrupt that caused the error is executed later or discarded.

Example of the use of phase offset

You have inserted two cyclic interrupt OBs in your program:

- Cyclic interrupt OB1
- Cyclic interrupt OB2

For cyclic interrupt OB1, you have set a time base of 20 ms and for cyclic interrupt OB2 a time base of 100 ms. After expiration of the time base of 100 ms, the cyclic interrupt OB1 reaches the start time for the fifth time, cyclic interrupt OB2 for the first time. To nevertheless execute the cyclic interrupt OBs offset, enter a phase offset for one of the two cyclic interrupt OBs.

Start information

- None
- Optimized start information:

Name	Data type	Meaning
first_scan	BOOL	= TRUE in the first call of this OB • At the transition from STOP or HOLD to RUN • After reloading
event_count	INT	Number of lost start events since the last start of this OB

See also

Organization blocks (OB) (Page 2170)

Assigning parameters to cyclic interrupt OBs (Page 1630)

Events and OBs (Page 1578)

Organization blocks for hardware interrupts

Description

You can use hardware interrupt OBs to react to specific events. You can assign an event that triggers an alarm to precisely one hardware interrupt OB. A hardware interrupt OB in contrast can be assigned to several events.

Hardware interrupts can be triggered by high-speed counters and input channels. For each high-speed counter and input channel that should trigger a hardware interrupt, the following properties need to be configured:

- The process event that should trigger the hardware interrupt (for example, the change of a count direction of a high-speed counter)
- The number of the hardware interrupt OB which is assigned to this process event

You can use up to 50 hardware interrupt OBs (OB numbers >= 123) that are independent of each other in your program.

Functionality of a hardware interrupt OB

After triggering a hardware interrupt, the operating system identifies the channel of the input or the high-speed counter and determines the assigned hardware interrupt OB.

If no other interrupt OB is active, the determined hardware interrupt OB is called. If a different interrupt OB is already being executed, the hardware interrupt is placed in the queue of its priority class. The hardware interrupt is acknowledged after the completion of the assigned hardware interrupt OB.

If another event that triggers a hardware interrupt occurs on the same module during the time between identification and acknowledgement of a hardware interrupt, the following applies:

- If the event occurs on the channel that previously triggered the hardware interrupt, then no additional hardware interrupt is triggered. Another hardware interrupt can only be triggered if the current hardware interrupt is acknowledged.
- If the event occurs on a different channel, a hardware interrupt is triggered.

Hardware interrupt OBs are called only in the CPU's "RUN" mode.

Start information

- None
- Optimized start information:

Name	Data type	Meaning
Laddr	HW_IO	Hardware identifier of the module that triggers the hardware interrupt
USI	WORD	Identifier for future extensions (not user-relevant)
IChannel	USINT	Number of the channel that triggered the hardware interrupt
EventType	BYTE	Identifier for the event type associated with the event triggering the interrupt (e.g., positive edge) This identifier can be found in the description of the respective module.

See also

Organization blocks (OB) (Page 2170)

Assigning parameters to hardware interrupt OBs (Page 1631)

Events and OBs (Page 1578)

Organization block for time error

Description

The operating system calls the time error OB (OB 80) if one of the following events occurs:

- The cyclic program exceeds the maximum cycle time.
- The OB called is currently still being executed (possible for time-delay interrupt OBs and cyclic interrupt OBs).
- A time-of-day interrupt was missed because the clock time was set forward by more than 20 seconds.
- A time-of-day interrupt was missed during a STOP.
- An overflow has occurred in an interrupt OB queue.
- An interrupt was lost due to the excessive interrupt load.

If you have programmed no time error OB, the S7-1200 CPU reacts as follows:

- CPUs with firmware version V1.0: The CPU remains in RUN mode.
- CPUs with firmware version V2.0:
  - The CPUs goes to STOP mode when the maximum cycle time is exceeded.
  - With all other start events of the time error OB, the CPU remains in RUN mode.

With CPUs with firmware version V1.0, the two-time overshooting of the maximum cycle time does not lead to the calling of an OB, but to the STOP of the CPU. You can avoid the second violation by restarting the cycle monitoring of the CPU with the RE_TRIGR instruction.

You can use only one time error OB in your program.

Start information

The time error OB has the following start information:

Tag	Data type	Description
fault_id	BYTE	• 0x01: Maximum cycle time exceeded • 0x02: Called OB is still being executed • 0x05: Expired time-of-day interrupt due to time jump • 0x06: Expired time-of-day interrupt on return to RUN mode • 0x07: Queue overflow • 0x09: Interrupt loss due to high interrupt load
csg_OBnr	OB_ANY	Number of the OB being executed at the time of the error
csg_prio	UINT	Priority of the OB being executed at the time of the error

See also

Organization blocks (OB) (Page 2170)

Events and OBs (Page 1578)

Organization block for diagnostic interrupts

Description

You can enable the diagnostic error interrupt for diagnostics-capable modules so that the module detects changes in the I/O status. As a result, the module triggers a diagnostic error interrupt in the following cases:

- A fault is present (incoming event)
- A fault is no longer present (outgoing event)

If no other interrupt OB is active, then the diagnostic interrupt OB (OB 82) is called. If another interrupt OB is already being executed, the diagnostic error interrupt is placed in the queue of its priority group.

You can use only one diagnostic interrupt OB in your program.

Start information

The diagnostic interrupt OB has the following start information:

Tag	Data type	Description
IO_state	WORD	Contains the I/O status of the diagnostics-capable module.
laddr	HW_ANY	HW-ID
Channel	UINT	Channel number
multi_error	BOOL	= 1, if there is more than one error

IO_state tag

The following table shows the possible I/O states that the IO_state tag can contain:

IO_state	Description
Bit 0	Configuration correct: = 1, if the configuration is correct = 0, if the configuration is no longer correct
Bit 4	Error: = 1, if an error is present, e.g., a wire break = 0, if the error is no longer present
Bit 5	Configuration not correct: = 1, if the configuration is not correct = 0, if the configuration is correct again
Bit 6	I/O cannot be accessed: = 1, if an I/O access error has occurred In this case, laddr contains the hardware identifier of the I/O with the access error. = 0, if the I/O can be accessed again

See also

Organization blocks (OB) (Page 2170)

Events and OBs (Page 1578)

Organization block for pulling and plugging

Description

The S7-1200 CPU operating system calls the pull/plug interrupt OB (OB 83) if a configured and non-disabled distributed I/O module or submodule (PROFIBUS, PROFINET, AS-i) is removed or inserted.

Note

The removal or insertion of a central module leads to the STOP of the CPU.

Start information

The pull/plug interrupt OB has the following start information:

Name	Data type	Meaning
LADDR	HW_IO	Hardware identifier of the affected module or submodule
Event_Class	BYTE	• B#16#38: (Sub)module plugged • B#16#39: (Sub)module pulled or not responding
Fault_ID	BYTE	Error code (possible values: B#16#51, B#16#54, B#16#55, B#16#56, B#16#57, B#16#58)

The following table shows which event caused the start of the pull/plug interrupt OB.

ev_class (B#16# ...)	fault_id (B#16# ...)	Meaning
39	51	Module pulled
39	54	Submodule pulled
38	54	Submodule plugged, which conforms to the parameterized submodule
38	55	Submodule inserted, which does not conform to the submodule parameter assignment
38	56	Submodule inserted, but error in module parameter assignment
38	57	Submodule or module inserted, but with a fault or maintenance
38	58	Submodule access error remedied

See also

Events and OBs (Page 1578)

Organization block for rack fault

Description

The S7-1200 CPU operating system calls the OB 86 in the following cases:

- The failure of a DP master system or of a PROFINET IO system is detected (in the case of either an incoming or an outgoing event).
- The failure of a DP slave or of an IO device is detected (in the case of either an incoming or an outgoing event).
- Failure of some of the submodules of a PROFINET I-device is detected.

Structure of the start information

The rack fault OB has the following start information:

Name	Data type	Meaning
LADDR	HW_IO	Hardware identifier of the defective hardware object
Event_Class	BYTE	• B#16#32: Activation of a DP slaves or an IO device • B#16#33: Deactivation of a DP slaves or an IO device • B#16#38: outgoing event • B#16#39: incoming event
Fault_ID	BYTE	Error code (possible values: B#16#C3, B#16#C4, B#16#C5, B#16#C6, B#16#C7, B#16#C8, B#16#C9, B#16#CA, B#16#CB, B#16#CC, B#16#CD, B#16#CE, B#16#CF, B#16#F8, B#16#F9)

The following table shows which event caused the start of OB 86.

Ev_class B#16# ...	Fault_id B#16# ...	Meaning
39	C3	Failure of a DP master system
39/38	C4	Failure/return of a DP slave
38	C5	Return of a DP slave; however, slave is faulty
38	C6	Expansion unit return, but error in module parameter assignment
38	C7	Return of a DP device, but there is error in module configuration
38	C8	Return of a DP device, but discrepancy between preset/actual configuration
32/33	C9	Activation/deactivation of a DP slave with the "D_ACT_DP" instruction
39	CA	Failure of a PROFINET IO system
39/38	CB	Failure/return of a PROFINET IO device
38	CC	Return of a PROFINET IO device with fault or maintenance
38	CD	Return of a PROFINET IO device, deviation between preset/actual configuration
38	CE	Return of a PROFINET IO device, error in module configuration
32/33	CF	Activation/deactivation of an IO device with the "D_ACT_DP" instruction
39/38	F8	Failure/return of some of the submodules of a PROFINET I-device
38	F9	Return of some of the submodules of a PROFINET I-device with a device configuration difference

See also

Events and OBs (Page 1578)

Block parameters of organization blocks

Basics of block parameters

Introduction

Several organization blocks (OBs) have properties with which you can control their behavior or their assignment to specific events. You can influence these properties by assigning parameters.

Overview

You can assign parameters to the properties for the following organization blocks:

- Time-of-day interrupt OBs
- Cyclic interrupt OBs
- Hardware interrupt OBs

See also

Assigning parameters to hardware interrupt OBs (Page 1631)

Assigning parameters to cyclic interrupt OBs (Page 1630)

Parameter assignment for time-of-day interrupt OBs

Procedure for setting the parameters

To set the parameters of a time-of-day interrupt OB, proceed as follows:

1. Open the "Properties" dialog belonging to the time-of-day interrupt OB in question.
2. Click the "Time-of-day interrupt" group in the area navigation.

Overview of the parameters that can be set

You can set the following parameters:

- Execution
- Start date and time-of-day
- Option buttons "Local time" and "System time"

"Execution" parameter

Use the drop-down list "Execution" to define the periods at which the time-of-day interrupt OB is to be executed. The time intervals relate to the settings for "Start date" and "Time-of-day".

The following values are possible for "Execution":

- Never
- Once
- Every minute
- Hourly
- Daily
- Weekly
- Monthly
- Yearly
- End of month

Note

With the value "End of month", the value specified under "Start date" is irrelevant.

"Start date" and "Time-of-day" parameter

Here, you specify the time at which the time-of-day interrupt is to be executed for the first time.

Example: Start date = 07/05/2013, time =11:16

Depending on the value of the "Execution" parameter, the CPU generates additional time-of-day interrupts periodically. Depending on the setting, the start time relates either to the local time or to the coordinated universal timeUTC.

Note

If you set the "Execution" parameter to "Monthly", the start date cannot be set to the 29th, the 30th, or the 31st. If you want the time-of-day interrupt OB to start at the end of the month, you should set the parameter "Execution" to "End of month".

"Local time" or "System time"

Here, you decide which time the start time of the time-of-day interrupt OB relates to:

- "Local time": The start time relates to the time zone set for the CPU.
- "System time": The start time relates to the coordinated universal time UTC (Universal Time Coordinated).

Assigning parameters to cyclic interrupt OBs

Introduction

You can use cyclic interrupt OBs to start programs at regular time intervals. To do so you must enter a scan time and a phase shift for each cyclic interrupt OB used.

You can use up to four cyclic interrupt OBs or time-delay OBs (OB numbers >= 200) in your program. If, for example, you are already using two time-delay interrupt OBs, you can insert a maximum of two further cyclic interrupt OBs in your program.

Note

If you assign multiple cyclic OBs, make sure that you assign a different cycle time or phase offset to each cyclic interrupt OB to avoid them executing at the same time or having to queue. When you create a cyclic interrupt OB, the cycle time 100 and the phase offset 0 are entered as start value.

Procedure

To enter a scan time and a phase shift for a cyclic interrupt OB, proceed as follows:

1. Open the "Program blocks" folder in the project tree.
2. Right-click on an existing cyclic interrupt OB.
3. Select the "Properties" command in the shortcut menu.
   The "<Name of the cyclic interrupt OB>" dialog box opens.
4. Click the "Cyclic interrupt" group in the area navigation.
   The text boxes for the scan time and the phase shift are displayed.
5. Enter the scan time and the phase shift.
6. Confirm your entries with "OK".

See also

Creating organization blocks (Page 2501)

Basics of block parameters (Page 1629)

Organization blocks for cyclic interrupts (Page 1621)

Assigning parameters to hardware interrupt OBs

Introduction

You must select the corresponding event and assign the following parameters for every input channel and high-speed counter that should trigger a hardware interrupt:

- Event name
- Number of the hardware interrupt OB that is assigned to this process event

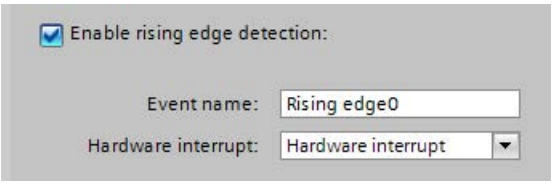
The parameters of the hardware interrupt are assigned in the properties of the corresponding device. You can assign parameters for up to 50 hardware interrupt OBs.

You can create the hardware interrupt OB to be assigned parameters either before or during activation of an event.

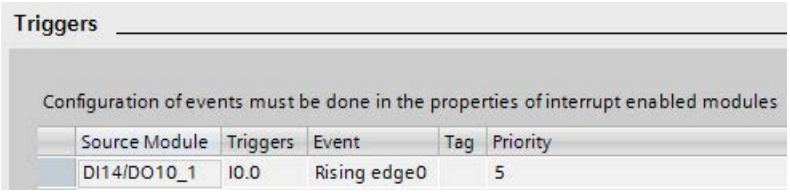
Procedure

To configure a hardware interrupt event, follow these steps:

1. Double-click the "Devices & Networks" command in the project tree.
   The hardware and network editor opens in the network view.
2. Change to the device view.
3. If the Inspector window closed in the device view, select the "Inspector window" check box in the "View" menu.
   The Inspector window opens.
4. Click the "Properties" tab.
5. In the device view, select the module for which you want to a assign a hardware interrupt.
6. Select the corresponding event that will trigger a hardware interrupt, e.g., a positive edge.



7. Enter an event name.
8. Select an existing hardware interrupt OB from the "Hardware interrupt" drop-down list or create a new hardware interrupt OB. If you have not previously created any hardware interrupt OBs, you can click the "Add new block" button in the drop-down list.
   The start information of the corresponding hardware interrupt OB, including all specifications for the interrupt-triggering event, is updated.



9. If you want to assign further hardware interrupts, repeat steps 5 to 8.

A system constant of data type Event_HwInt is created automatically for the event identified by the explicit event name. The system constants are displayed in the standard tag table.

See also

Creating organization blocks (Page 2501)

Basics of block parameters (Page 1629)

Organization blocks for hardware interrupts (Page 1623)

Events and OBs (Page 1578)

Symbolic and numerical names of instructions

Description

The instructions from the task card are comprised of functions (FC), function blocks (FB), system functions (SFC) and system function blocks (SFB) that are identified internally by numbers.

The following tables show the assignment of numerical and symbolic names.

Function blocks (FBs)

Numerical name	Symbolic name
FB 105	TC_CONFIG
FB 110	Port_Config
FB 111	Send_Config
FB 112	Receive_Config
FB 113	Send_P2P
FB 114	Receive_P2P
FB 115	Receive_Reset
FB 116	Signal_Get
FB 117	Get_Features
FB 118	Set_Features
FB 163	TC_SEND
FB 164	TC_RECV
FB 165	TC_CON
FB 166	TC_DISCON
FB 804	SET_TIMEZONE
FB 1030	TSEND_C
FB 1031	TRCV_S
FB 1071	USS_DRIVE
FB 1080	MB_COMM_LOAD
FB 1081	MB_MASTER
FB 1082	MB_SLAVE
FB 1084	MB_CLIENT
FB 1085	MB_SERVER
FB 1100	MB_Halt
FB 1101	MC_Home
FB 1102	MC_MoveAbsolute
FB 1103	MC_MoveJog
FB 1104	MC_MoveRelative
FB 1105	MC_MoveVelocity
FB 1107	MC_Power
FB 1108	MC_Reset

Numerical name	Symbolic name
FB 1110	MC_MoveInterrupt
FB 1111	MC_ChangeDynamik
FB 1112	MC_CommandTable
FB 1113	MC_MoveLinearAbs_2D
FB 1114	MC_MoveLinearRel_2D
FB 1115	MC_MoveCircular_2D
FB 1130	PID_Compact
FB 1134	PID_3Step
FB 1140	HSC
FB 2040	RecipeCreate
FB 2041	RecipeOpen
FB 2042	RecipeRead
FB 2043	RecipeWrite
FB 2044	RecipeAppend
FB 2045	RecipeClose

Functions (FCs)

Numerical name	Symbolic name
FC 2 (1)	CONCAT
FC 4 (1)	DELETE
FC 11 (1)	FIND
FC 17 (1)	INSERT
FC 20 (1)	LEFT
FC 21 (1)	LEN
FC 22 (1)	LIMIT
FC 25 (1)	MAX
FC 26 (1)	MID
FC 27 (1)	MIN
FC 31 (1)	REPLACE
FC 32 (1)	RIGHT
FC 36 (1)	ENCO
FC 36 (1)	SEL
FC 37	DECO
FC 800	LED
FC 801	IM_DATA
FC 802	DeviceStates
FC 803	ModuleStates
FC 1070	USS_PORT
FC 1072	USS_RPM
FC 1073	USS_WPM
(1) MC7+ instruction	

System data types (SDTs)

Numerical name	Symbolic name
SDT 99	WWW_CDB
SDT 513	CONDITIONS
SDT 581	Send_Conditions
SDT 582	Receive_Conditions

System function blocks (SFBs)

Numerical name	Symbolic name
SFB 0 (1)	CTU
SFB 1 (1)	CTD
SFB 2 (1)	CTUD
SFB 3 (1)	TP
SFB 4 (1)	TON
SFB 5 (1)	TOF
SFB 27	START_OB
SFB 52	RDREC
SFB 53	WRREC
SFB 54	RALRM
SFB 105	T_CONFIG
SFB 106	TDIAG
SFB 107	TRESET
SFB 110	PORT_CFG
SFB 111	SEND_CFG
SFB 112	RCV_CFG
SFB 113	SEND_PTP
SFB 114	RCV_PTP
SFB 115	SGN_GET
SFB 116	SGN_SET
SFB 117	RCV_RST
SFB 120	CTRL_HSC
SFB 122	CTRL_PWM
SFB 140	DataLogCreate
SFB 141	DataLogOpen
SFB 142	DateLogWrite
SFB 143	DataLogClear
SFB 144	DataLogClose
SFB 145	DataLogDelete
SFB 146	DataLogNewFile

System functions (SFCs)

Numerical name	Symbolic name
SFC 7	DP_PRAL
SFC 11	DPSYC_FR
SFC 13	DPNRM_DG
SFC 14	DPRD_DAT
SFC 16	RD_OBINF
SFC 23	DEL_DB
SFC 28	SET_TINT
SFC 29	CAN_TINT
SFC 30	ACT_TINT
SFC 31	QRY_TINT
SFC 32	SRT_DINT
SFC 33	CAN_DINT
SFC 34	QRY_DINT
SFC 41	DIS_AIRT
SFC 42	EN_AIRT
SFC 43	RE_TRIGR
SFC 45	DE_ACT
SFC 46	STP
SFC 82	CREA_DBL
SFC 83	READ_DBL
SFC 84	WRIT_DBL
SFC 86	CREATE_DB
SFC 89	RST_EVOV
SFC 99	WWW
SFC 101	RTM
SFC 117	GET_DIAG
SFC 124	ATTR_DB
SFC 140	IO2MOD
SFC 143	RD_ADDR
SFC 154	RD_LOC_T
SFC 154	DPWR_DAT
SFC 161	WR_LOC_T
SFC 180	ID2LOG
SFC 181	LOG2ID
SFC 182	ID2GEO
SFC 190	SET_CINT
SFC 191	QRY_CINT
SFC 192	ATTACH
SFC 193	DETACH

Numerical name	Symbolic name
MC7+ Anweisung	GET_ERROR
MC7+ Anweisung	GET_ERR_ID

Useful information on CPU firmware versions and STEP 7 versions

CPUs and engineering software for configuring the CPUs is constantly developed further to improve performance and security. New versions that have some special features with respect to the interaction of the components are created in this way. The sections below describe the special features of the S7-1200 CPUs with firmware version V4 as compared to firmware versions V1 to V3.

For a detailed comparison of the range of functions, read the S7-1200 System Manual (description of new instructions, new organization blocks and advanced configuration options).

Required engineering software

S7-1200 CPUs V4 can be configured with STEP 7 as of V13.

Compatibility between memory card content and firmware version of the CPU

Memory cards (transfer cards or program cards) with configuration and program for an S7-1200 CPU V1, V2 or V3 do not work with an S7-1200 CPU V4.
Memory cards with configuration and program for an S7-1200 CPU V4 do not work with an S7-1200 CPU V1, V2 or V3.

You must change an S7-1200 CPU V1 to V3 configuration to an S7-1200 CPU V4 configuration (device replacement) and then download it to the CPU. A gradual device replacement is required for an S7-1200 CPU V1-V2 (see below).

When you insert the memory card into a CPU with an incompatible firmware version, the CPU does not start up. When you insert a memory card for a CPU V1, V2 or V3 into an S7-1200 CPU V4, this CPU outputs a version error.

Going online and loading

When you have configured an S7-1200 CPU with firmware version V1, V2 or V3 with STEP 7, the CPU to which you want to go online or which you want to download must have one of these firmware versions. You cannot go online to an S7-1200 CPU V4 with a configured S7-1200 CPU V1, V2 or V3.

On the other hand, you cannot go online to an S7-1200 CPU V1, V2 or V3 with a configured S7-1200 CPU V4 or download this CPU.

Replacing an existing CPU

You can replace a configured S7-1200-CPU V1, V2 or V3 with a new CPU with a firmware version greater than or equal to 4. In case of an existing S7-1200 CPU V1 or V2, you must

start by replacing the device with an S7-1200 CPU V3: It is not possible to directly replace it with an S7-1200 CPU V4.

1. S7-1200 CPU V1 (V2) > S7-1200 CPU V3
2. S7-1200 CPU V3 > S7-1200 CPU V4

As long as you do not download the configuration, you can undo the device replacement ("Undo" command in the "Edit" menu). When you download the configuration for the new CPU firmware version ("V4 configuration") to the CPU, you can no longer return to version V3.

This means you should save the existing project, for example, with a V3 configuration as project archive so that you can access it later.

Special notes for device replacement (V3 > V4):

- The interrupt behavior of interrupt OBs remains the same; they are not configured as interruptible. This is also the default behavior of S7-1200 CPUs V1-V3. The interrupt behavior of the interrupt OB can be configured for S7-1200 V4 CPUs. If you drag an S7-1200 CPU V4 directly from the hardware catalog into the network view, this option is activated (interrupt OBs are interruptible).
- The behavior for PUT/GET access of remote partners remains the same; access is permitted. This is also the default behavior of S7-1200 CPUs V1-V3. The access via PUT/GET communication by means of remote partners can be configured for S7-1200 V4 CPUs ("Protection" area of CPU parameters). If you drag an S7-1200 CPU V4 directly from the hardware catalog into the network view, access is not permitted and must be explicitly enabled.
- The wording for the protection levels changes but the effect of the settings remains the same. You can also select the access level "No access (complete protection)".
- The web server settings for activation of the web server and HTTP/HTTPS settings are applied. You also have the option to create users and assign them specific rights (Web server area > User management of CPU parameters). Web server users only have access to standard websites when you do not configure users. An S7-1200 CPU V4 no longer supports the user "admin" and his/her password.

Communication with HMI devices

When you connect an HMI device to an S7-1200 CPU V4, make sure that you use the appropriate Runtime software version of the HMI device.

You may have to transfer the latest HMI Runtime version by means of the WinCC engineering software.

You must compile the HMI configuration once again and download it to the HMI device for the CPU-HMI communication to work.

See also

Useful information on memory cards (Page 1570)

8.1.5.2 Functional description of S7-300/400 CPUs

Operating modes

Principles of the operating modes of S7-CPUs

Operating modes describe the behavior of the CPU. You can query and evaluate the operating modes when programming the STARTUP, when testing the controller and when performing error diagnostics at any time.

S7 CPUs from the various different CPU families such as S7-300, S7-400, ET 200S, ET 200Pro or WinLC (PC-based controller) or embedded controller can have the following operating modes:

- STOP
- STARTUP
- RUN
- HOLD

The CPU can communicate via its interfaces (e.g. MPI interface) in these operating modes.

"STOP" operating mode

In "STOP" operating mode the CPU checks whether all configured modules and all modules used by the standard addressing are available. Provided the result is good, the CPU then sets the I/O to a pre-defined default condition. The user program is not executed when the CPU is in "STOP" operating mode.

See also: STOP mode (Page 1641)

"STARTUP" operating mode

A distinction is made in "STARTUP" operating mode between the "warm restart", "hot restart" and "cold restart" start-up modes.

See also: Auto-Hotspot

"RUN" operating mode

In "RUN" operating mode the CPU executes the user program, updates the inputs and outputs and processes interrupts and error messages.

See also: RUN mode (Page 1650)

"HOLD" operating mode

In "HOLD" operating mode, execution of the user program is halted. The "HOLD" operating mode is used only when using the programming device for testing.

See also: HOLD mode (Page 1650)

Other operating modes

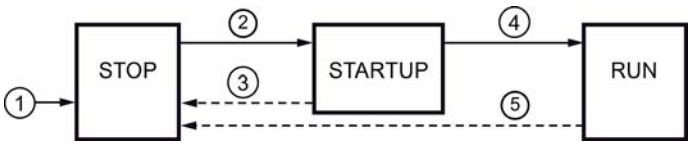
If the CPU is not ready for operation, it is in one of two operating modes:

- Deenergized, i.e. the supply voltage is switched off.
- Defective, i.e. a non-recoverable error has occurred.
  Check that the CPU is truly defective. To do this, set the CPU to "STOP" and switch the power switch off and on again. To analyze the error, read the diagnostics buffer when the CPU starts up. If the CPU does not start up, replace it.

Operating mode transitions

Overview

The following figure shows the operating modes and the operating mode transitions for the S7-300 CPUs:



The following table shows the conditions under which the operating modes will change:

No.	Operating mode transition	Conditions
①	STOP	After you turn on the power supply, the CPU is in "STOP" operating mode. It then determines the required type of startup and changes to the next operating mode.
②	STOP → STARTUP	In the following cases the CPU switches to "STARTUP" operating mode: • The CPU is set to "RUN" either by the operating mode selector or from the programming device. • After automatic triggering of a STARTUP operating mode by "POWER-ON". • The communications function "RESUME" or "START" is performed. In the latter two cases the operating mode selector must be set to "RUN".
③	STARTUP → STOP	In the following cases the CPU switches back to STOP operating mode: • An error is detected during the STARTUP. • The CPU is set to "STOP" either by the operating mode selector or from the programming device. • A STOP command is executed in the STARTUP OB. • The communications function "STOP" is performed.
④	STARTUP → RUN	If the STARTUP is successful, the CPU switches to "RUN".
⑤	RUN → STOP	In the following cases the CPU switches back to STOP operating mode: • An error is detected in RUN operating mode and the associated OB is not loaded. • The CPU is set to "STOP" either by the operating mode selector or from the programming device. • A STOP command is executed in the user program. • The communications function "STOP" is performed.

Priority of the operating modes

Function

If several operating mode changes are requested at the same time, the CPU switches to the operating mode with the highest priority. If, for example, the operating mode selector is set to "RUN" and an attempt is made via the programming device to switch the CPU to "STOP", the CPU will switch to "STOP", because this operating mode has the highest priority.

Priority	Operating mode
Highest	STOP (Page 1641)
	HOLD (Page 1650)
	STARTUP
Lowest	RUN (Page 1650)

STOP mode

Function

In "STOP" operating mode the user program is not executed. All outputs are set to substitute values and the process being controlled is thereby brought to a safe operating mode. The CPU checks the following points:

- Hardware, for example whether are all modules are available.
- Whether the default settings for the CPU are applicable or parameter sets are present.
- Whether the general conditions for the programmed STARTUP behavior are correct.

In STOP mode you can receive global data. In addition passive one-way communications via the communications system function blocks for configured connections and via communications system functions for unconfigured connections can be performed.

See also

Principles of the operating modes of S7-CPUs (Page 1639)

"STARTUP" operating mode

Principles of the STARTUP mode

Function

After the CPU has been switched on, it executes a STARTUP program before starting to execute the user program. By suitably programming the STARTUP OB, you can use the startup program to specify certain default settings for your cyclic program.

The following STARTUP operating modes are available:

- Warm restart
- Hot restart
- Cold restart

Special characteristics

Note the following points regarding the "STARTUP" operating mode:

- The program in the STARTUP OB will be executed. This means OB 100 for a "warm restart", OB 101 for a "hot restart", and OB 102 for a "cold restart".
- No time-based or interrupt-based program execution can be performed.
- The timers are updated.
- The runtime meter is running.
- The outputs on the modules are disabled.

See also

Principles of the operating modes of S7-CPUs (Page 1639)

Hot restart (Page 1643)

Cold restart (Page 1642)

Startup organization blocks (OB 100, OB 101 and OB 102) (Page 1807)

Complete overview of the startup (Page 1648)

Cold restart

Cold restart

During a cold restart, all data (process image, bit memory, timers, counters and data blocks) are reset to the start values stored in the program (load memory), irrespective of whether they were configured as retentive or non-retentive.

Program execution is restarted at the beginning (startup OB or OB 1).

Warm restart

Permitted warm restart

You can perform a warm restart in the following cases:

- After a memory reset.
- After loading the user program in CPU STOP operating mode.

You cannot run a warm restart if the system has requested a memory reset.

Manual restart (warm restart)

A manual restart ("warm restart") is triggered by:

- The operating mode selector.
- A menu command from the programming device or via a communications function, if the operating mode selector is set to RUN.

Automatic restart (warm restart)

An automatic restart ("warm restart") can by triggered by "POWER ON" in the following cases:

- When the "POWER OFF" occurred, the CPU was not in "STOP" operating mode.
- The operating mode selector is set to "RUN".
- The CPU was interrupted by a power failure during a warm restart. This is irrespective of the assigned STARTUP operating mode.

Automatic restart (warm restart) without battery backup

If your CPU is running without a backup battery, e.g. if maintenance-free operation is required, after switching on or on restoration of power after "POWER OFF" the CPU is automatically subjected to a memory reset and then a "warm restart". The user program must exist on flash EPROM (memory card) so that the user program can be copied to work memory again.

Hot restart

Hot restart

This start-up mode is only available with S7-400. The features of hot restart are as follows:

- When a hot restart is performed, all data and the process image retain their last valid value.
- Program execution is resumed at the breakpoint.
- The outputs do not change their status until the current cycle is completed.
- After a power supply interruption, the hot restart function is only available in backup mode.

S7-400 CPUs execute an initialization routine and then automatically a hot restart. During a hot restart, the user program is continued at the location at which the processing was interrupted. The part of the user program that was no longer processed before the power failure is referred to as remaining cycle. The remaining cycle can also contain time and interrupt controlled program parts.

As a rule a hot restart is only permissible if the user program was not modified in STOP state (for example, by reloading a modified block), or if a warm restart is not required for other reasons. It is necessary to distinguish between a manual and an automatic restart.

Manual hot restart

A manual hot restart is only possible if the appropriate parameters are set for the CPU and after the following STOP causes:

- The mode selector was changed from RUN to STOP
- By user programmed STOPs, STOPs after calling of OBs that are not loaded
- STOP state was caused by the programming device or by communication functions

A manual restart can be triggered via menu command from programming device or via communication functions (if the mode selector is on RUN or RUN-P). A requirement for this is that the restart is not disabled by operation (startup parameter).

- If the parameter set of the CPU's manual restart was configured

Automatic restart

An automatic restart was triggered by POWER ON if the CPU was not in STOP or HOLD at POWER OFF. A requirement for this is that the restart is not disabled by operation (startup parameter).

Retentive behavior after loss of power

Retentive data areas after power failure

To avoid loss of data in the event of a power failure, you can specify the retentivity of bit memory, timers, counters and areas in data blocks. Providing that the STARTUP operating mode is "warm restart" (default), an "automatic restart with memory" will be performed when power is restored.

S7-300 and S7-400 CPUs react differently to power return after power failure.

S7-300 CPUs know only the "warm restart" restart mode To avoid loss of data in the event of a power failure, you can specify with STEP 7 the retentivity of bit memory, timers, counters and areas in data blocks. An "Automatic warm restart with memory" is executed when power returns.

Depending on the set parameters, S7-400 CPUs react to power return with either a warm restart (after buffered on unbuffered POWER ON) or with hot restart (only possible after buffered POWER ON).

Retentive behavior for CPUs with battery back-up

The following table shows the retentive behavior in the work memory of the S7-300 and S7-400 CPUs with buffering at "warm restart" and "cold restart":

Data	Warm restart	Cold restart	Hot restart
Blocks in the load memory	X	X	X
DB in the work memory	X	0	X
Bit memory, timers, counters configured as retentive	X	0	X

Data	Warm restart	Cold restart	Hot restart
Bit memory, timers, counters, configured as non-retentive	0	0	X
X: Data is retained 0: Data is reset or cleared (content of DBs)			

Retentive behavior for CPUs without battery back-up

No hot restart is permitted with S7-400 CPUs!

The following table shows the retentive behavior in the work memory of the S7-300 CPUs without battery backup at "warm restart" and "cold restart":

Data	Warm restart		Cold restart	
	S7-300	S7-400	S7-300	S7-400
Blocks in the load memory	VC	VC	VC	VC
DB in the work memory configured as retentive	VX	not possible	V	not possible
DB in the work memory configured as non-retentive	V	V	V	V
Bit memory, timers, counters configured as retentive	X	0	0	0
Bit memory, timers, counters, configured as non-retentive	0	0	0	0
X: Data is retained 0: Data is reset or cleared (content of DBs) V: Data is set to the default value from the EPROM VC: Code block in the EPROM is retained, any overloaded code blocks are lost VX: A data block is only retained if it is available in the load EPROM; retentive data is transferred the from NV-RAM (data blocks loaded or generated in RAM are lost)				

Startup activities

Overview

The following table shows which activities the CPU performs at STARTUP:

Activities in execution sequence	At warm restart	At cold restart	At hot restart
Clear the interrupt and block stack	X	X	0
Clear non-retentive bit memory, timers, counters	X	0	0
Clear all bit memory, timers, counters	0	X	0
Clear the process image output	X	X	Can be set

Activities in execution sequence	At warm restart	At cold restart	At hot restart
Reset the signal module outputs	X	X	Can be set
Discard the process interrupts	X	X	0
Discard the time-delay interrupts	X	X	0
Discard the diagnostics interrupts	X	X	X
Update the system status list (SSL)	X	X	X
Evaluate the module parameters and transfer them to modules or transfer default values	X	X	X
Execute the respective STARTUP OB	X	X	X
Execute the outstanding cycle (part of the user program that because of the "POWER OFF" was not executed	0	0	X
Update the process image input	X	X	X
Enable digital outputs (cancel the OD (Output Disabled) signal) after operating mode transition to RUN	X	X	X
X: Activity is performed 0: Activity is not performed			

See also

Complete overview of the startup (Page 1648)

Cancellation of a startup

Reasons for cancellation

If errors occur during startup, the startup is canceled and the CPU switches to or remains in "STOP" operating mode. This has the following effects on the "warm restart" operating mode:

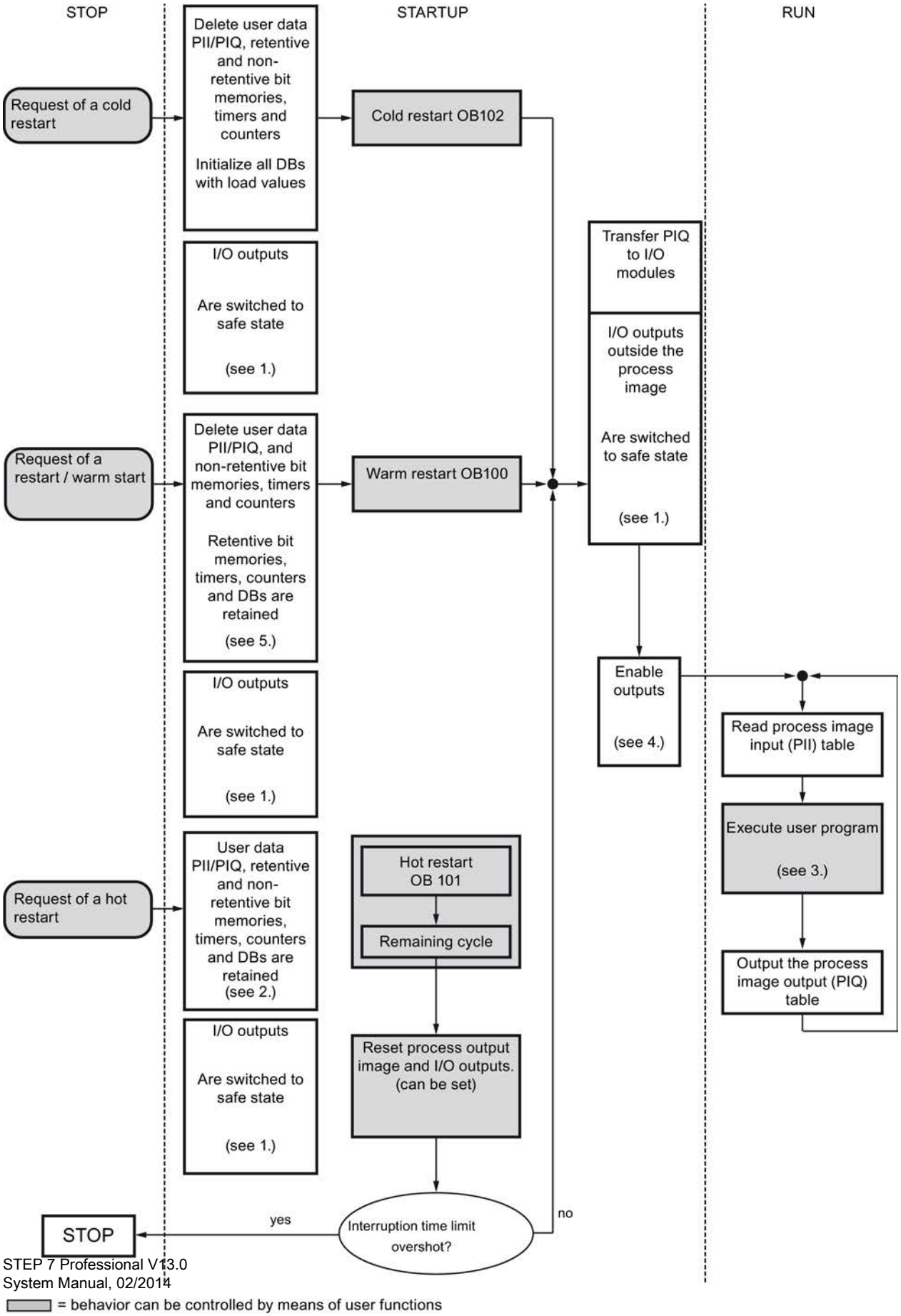
- A "warm restart" that is canceled must be repeated.
- Both a warm restart and a hot restart are possible in the event of a canceled hot restart.

- Under the following conditions a warm restart or hot restart will not be executed or will be canceled:
  - If the CPU operating mode selector is set to the "STOP" operating mode.
  - If a memory reset is requested.
  - If a memory card whose application code is not permissible is plugged in.
  - If more than one CPU is plugged in in single processor operation.
  - If the user program contains an organization block that the CPU does not recognize or has been disabled.
  - If, after the power is switched on, the CPU detects that not all modules that are listed in the configuration created with STEP 7 are actually plugged in (target-actual difference in configuration is not permitted).
  - If errors occur during the assignment of the module parameters.
- Under the following conditions, a hot restart will not be performed or will be canceled:
  - If the CPU memory was previously reset (after the memory reset, only a warm restart is permitted).
  - If the interruption limit is exceeded (the interruption time is the time that expires after leaving RUN mode until the startup OB incl. remaining cycle time was processed).
  - If the module configuration was changed (e.g. module exchange).
  - If the only warm restart configuration is permitted.
  - If blocks were loaded, deleted or changed in STOP mode.

Complete overview of the startup

Sequence

The following figure shows the activities of the CPU in "STOP", "STARTUP", and "RUN" operating modes.


STOP
STARTUP
RUN
Request of a cold restart
Delete user data PII/PIQ, retentive and non-retentive bit memories, timers and counters
Initialize all DBs with load values
Cold restart OB102
I/O outputs
Are switched to safe state
(see 1.)
Transfer PIQ to I/O modules
I/O outputs outside the process image
Are switched to safe state
(see 1.)
Request of a restart / warm start
Delete user data PII/PIQ, and non-retentive bit memories, timers and counters
Retentive bit memories, timers, counters and DBs are retained
(see 5.)
Warm restart OB100
I/O outputs
Are switched to safe state
(see 1.)
Enable outputs
(see 4.)
Read process image input (PII) table
Execute user program
(see 3.)
Output the process image output (PIQ) table
Request of a hot restart
User data PII/PIQ, retentive and non-retentive bit memories, timers, counters and DBs are retained
(see 2.)
Hot restart OB 101
Remaining cycle
I/O outputs
Are switched to safe state
(see 1.)
Reset process output image and I/O outputs. (can be set)
STOP
yes
Interruption time limit overshot?
no
= behavior can be controlled by means of user functions

Legend for figure:

1. All peripheral outputs are switched by the peripheral modules (hardware side) to safe state (default value = "0"). This is regardless of whether it is used in the user program inside or outside the process image area.
   If signal modules with substitute value capability are used, the behavior of the outputs can be assigned, e.g. retain last value.
2. Necessary for the processing of the remaining cycle.
3. The interrupt OBs are also available during first calling of a current process image of the inputs.
4. You can use the following measures to specify the state of the local and distributed peripheral outputs in the first cycle of the user program:
   – Use configurable output modules to be able to output substitute values or to retain the last value.
   – On restart: Enable CPU startup parameter "Reset outputs on restart" in order to output a "0" (corresponds to default setting).
   – Pre-assign outputs in the startup OB (OB 100, OB 101, OB 102).
5. In S7-300 systems only the DB areas that are configured as retentive are retained.

RUN mode

Function

In "RUN" operating mode the cyclic, time-driven, and interrupt-driven program execution is performed:

- The process image input is read in.
- The user program is executed.
- The process image output is read out.

The active exchange of data between CPUs via global data communication (global data table), via communication instructions for configured connections and via communication instructions for non-configured connections is only possible in RUN mode.

See also

Principles of the operating modes of S7-CPUs (Page 1639)

HOLD mode

Function

"HOLD" mode is a special case, since it is used only for tests in "STARTUP" or "RUN" operating mode.

"HOLD" mode:

- All timers are frozen. Timers and runtime meters are not processed, monitoring timers and the basic clock cycles of the time-driven levels are halted.
- The real-time clock still runs.
- Outputs are not enabled, they can however be enabled for test purposes.
- Inputs and outputs can be controlled.
- When power is restored after a power failure, CPUs with battery back-ups switch from "HOLD" to "STOP" operating mode and do not perform a "warm restart". On restoration of power, CPUs without battery back-ups automatically perform a "warm restart".
- They can receive global data. In addition passive one-way communication via the communications SFBs for configured connections and via communications SFCs for unconfigured connections can be performed.

See also

Principles of the operating modes of S7-CPUs (Page 1639)

Basics of a memory reset

Function

A CPU memory reset can be performed in STOP mode. A memory reset can be performed manually using the mode selector (MRES) or from the programming device, e.g. before loading a user program.

A memory reset resets the CPU to the "initial condition". This means that:

- The entire user program in the work memory and in the RAM load memory and all operand areas are cleared.
- The system parameters together with the CPU and module parameters are reset to the default setting. The MPI parameters set before the memory reset remain unchanged.
- When a memory card (flash EPROM) or micro memory card is plugged in, the CPU will copy the user program from the memory card to the work memory. If the respective configuration data is also present on the memory card, the CPU and module parameters are also copied.
- The diagnostics buffer, the MPI parameters, the time of day, and the runtime meter are not reset.

Memory areas

What you should know about micro memory cards for S7-300

How the memory card functions

The SIMATIC micro memory card is a memory module for a S7-300-CPU. You can configure the micro memory card as a load memory or as removable media.

STEP 7 V11 supports use as load memory.

To use it you have to insert the SIMATIC micro memory card in the CPU.

If you format the micro memory card with Windows, for example with a commercially available card reader, you make the micro memory card unusable as a storage medium for a S7-CPU.

What can be saved on a SIMATIC Micro Memory Card

You can save the user program (blocks and system data) on a micro memory card.

Alternatively, you can use the micro memory card for firmware updates or for backup of the CPU firmware.

Transferring objects from the project to a micro memory card

When the micro memory card is inserted in the programming device or an external prommer, you can transfer the following objects from project navigation to the micro memory card:

- Individual blocks (multiple selection possible)
  In this case a consistent transfer is available, as the dependencies of the blocks to each other is taken into account with block selection.
- PLC
  In this case, all objects relevant to processing are transferred, such as blocks and the hardware configuration on the micro memory card, just as with downloading.

To perform a transfer, you can move the objects by dragging and dropping, or use the command "Write to memory card" in the "Project" menu.

Transferring objects from the micro memory card to the project

You can transfer Individual blocks (multiple selection is possible) by dragging them to the project. A hardware configuration cannot be transferred from the memory card to the project.

Backing up firmware on the micro memory card

We recommend that your create a backup copy of the firmware for emergencies.

To do so, you must insert a blank micro memory card in the CPU and adhere to a specific sequence of steps. You will find them described in the CPU documentation.

Documentation for S7-300 (http://support.automation.siemens.com/WW/view/de/13008499/0/en)

Updating firmware with a micro memory card

You can get the latest firmware on the internet as *.UPD files from the Service & Support pages:

http://support.automation.siemens.com (http://support.automation.siemens.com)

Save the firmware files on the hard drive.

In order to store the files on the micro memory card, select the command "Create firmware update memory card" in the "Project" menu.

Then follow the instructions in the Service & Support portal for performing a firmware update with your CPU.

See also

Memory areas of the S7-300 CPUs (Page 1655)

What you should know about memory cards for S7-400

How the memory card functions

You can insert a memory card in all S7-400 CPUs. They act as an expansion of the load memory. Depending on the type of memory card used, the user program remains saved on the memory card, even if the module has no voltage.

Types of memory cards

- RAM card: only for expanding the integrated load memory.
- FLASH card: for saving the user program, even if there is no voltage, without buffering or external to the CPU.

Transferring objects from the project to a memory card

When the memory card is inserted in the programming device or an external prommer, you can transfer the following objects from project navigation to the memory card:

- Individual blocks (multiple selection possible)
  In this case a consistent transfer is available, as the dependencies of the blocks to each other is taken into account with block selection.
- PLC
  In this case, all objects relevant to processing are transferred, such as blocks and the hardware configuration on the memory card, just as with downloading.

To perform a transfer, you can move the objects by dragging and dropping, or use the command "Write to memory card" in the "Project" menu, and if applicable, follow the instructions in the dialog.

Transferring objects from the memory card to the project

You can transfer Individual blocks (multiple selection is possible) by dragging them to the project. A hardware configuration cannot be transferred from the memory card to the project.

Downloading the user program to the memory card

The command "Download user program to memory card" in the menu "Online" saves the entire user program to the FLASH card. The FLASH card is in the CPU. Before the download, all existing data on the FLASH card is deleted.

- The CPU must be in STOP mode.
- You can only download to FLASH cards with these menu commands

Updating firmware with a memory card

You can get the latest firmware on the internet as *.UPD files from the Service & Support pages:

http://support.automation.siemens.com (http://support.automation.siemens.com)

Save the firmware files on the hard drive.

To store the file on the memory card, select the command "Create firmware update memory card" in the "Project" menu.

Then follow the instructions in the Service & Support portal for performing a firmware update with your CPU.

See also

Memory areas of the S7-400-CPUs (Page 1657)

Organization of Memory Areas

Distribution

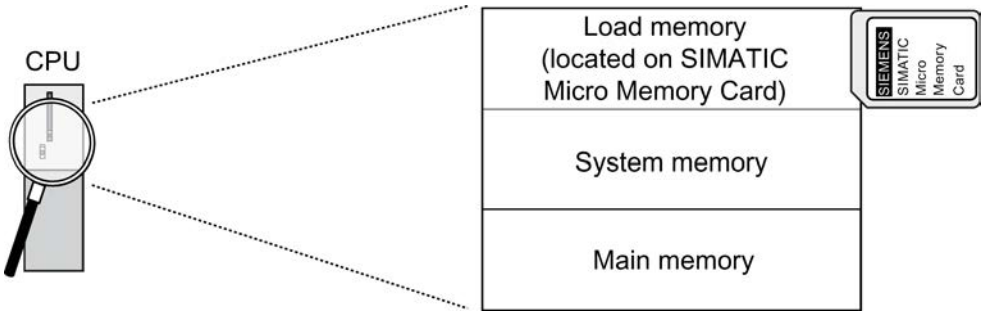
The memory of the S7 CPUs can be divided into three areas:

- Load memory (depending on the CPU and memory card type executed as RAM, EPROM or flash memory)
- Work memory (integrated RAM)
- System memory (integrated RAM)

Memory areas of the S7-300

Memory areas of the S7-300 CPUs

The three memory areas of your CPU



Load memory

The load memory is located on the SIMATIC Micro Memory Card. The size of the load memory corresponds exactly to the size of the SIMATIC Micro Memory Card. It is used to store code blocks, data blocks, and system data (configuration, connections, module parameters, etc.). Blocks that are identified as not relevant for execution are stored exclusively in the load memory. You can also store all the configuration data for your project on the SIMATIC Micro Memory Card.

Note

You must insert a SIMATIC Micro Memory Card into the CPU to enable loading of user programs and operation of the CPU.

System memory

The system memory is integrated in the CPU and cannot be expanded.

It contains

- The address areas for bit memories, timers, and counters
- The process images of the inputs and outputs
- Local data

Work memory

The work memory is integrated in the CPU and cannot be extended. It is used to execute the code and process user program data. Programs only run in the work memory and system memory.

See also

What you should know about micro memory cards for S7-300 (Page 1651)

Retentivity of load memory, system memory, and main memory

Your CPU is equipped with a maintenance-free retentive memory, i.e. no back-up battery is required for its operation. Due to the retentivity, the content of the retentive memory is retained even during a POWER OFF and restart (warm restart).

Retentive data in the load memory

Your program in the load memory is always retentive: It is stored on the SIMATIC Micro Memory Card, where it is protected against power failures or memory resets

Retentive data in the system memory

For bit memory, timers and counters, you decide via configuration (CPU properties) which part should be retentive and which should be initialized to "0" at warm restart.

The diagnostic buffer, MPI address and transmission speed (baud rate), as well as the runtime meter, are generally stored in the retentive memory area on the CPU. Retentivity of the MPI address and transmission speed ensures that your CPU can continue to communicate, even after a power failure, memory reset or loss of communication parameters (e.g. due to removal of the SIMATIC Micro Memory Card or deletion of communication parameters).

Retentive data in the work memory

The contents of retentive DBs are always retentive at restart and POWER ON/OFF. Retentive data blocks can be uploaded to the work memory in accordance with the maximum limit allowed by the work memory.

Non-retentive DBs are initialized from the load memory with their initial values at restart or POWER ON/OFF. Non-retentive data blocks and code blocks can be loaded in accordance with the maximum work memory limit.

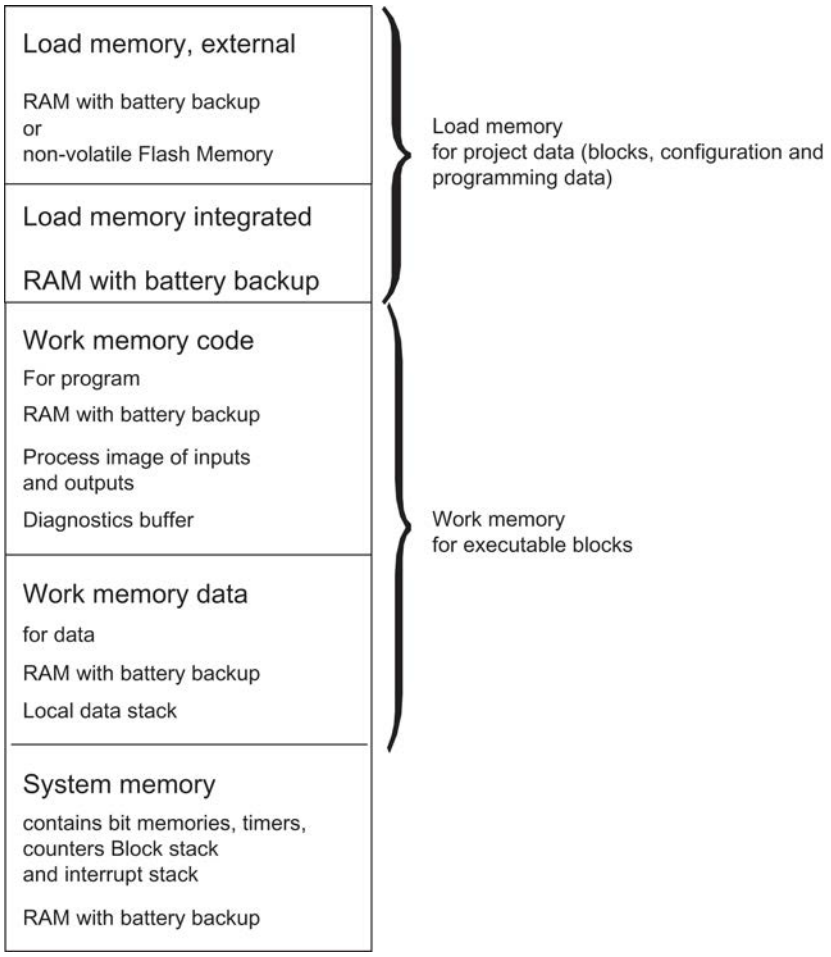
You can find the size of the CPU retentive work memory (for retentive data blocks) in the technical specifications of the respective CPU.

Memory areas of the S7-400

Memory areas of the S7-400-CPUs

Organization of Memory Areas

The memory of the S7-400-CPU can be divided into the following areas:



Memory Types in S7-400 CPUs

- Load memory for the project data, e.g. blocks, configuration and parameter settings.
- Work memory for the runtime-relevant blocks (logic blocks and data blocks).

- System memory (RAM) contains the memory elements that each CPU makes available to the user program, such as bit memory, timers and counters. System memory also contains the interrupt stack
- System memory of the CPU also makes temporary memory available (local data stack, diagnostic buffer and communication resources) that is assigned to the program for the temporary data of a called block. These data is only valid as long as the block is active. By changing the default values for the process image, local data, diagnostic buffer and communications resources (CPU properties for which you can set parameters) you can influence the work memory available for the process-relevant blocks.

Note

Please note the following if you expand the process image of a CPU. Reconfigure modules whose addresses have to be over the highest address of the process image so that the new addresses are still over the highest address of the expanded process image.

Important note for CPUs after the parameter settings for the allocation of RAM have been changed

If you change the work memory allocation by modifying parameters, this work memory is reorganized when you load system data into the CPU. The result of this is that data blocks that were created with SFC are deleted, and the remaining data blocks are assigned initial values from the load memory.

The usable size of the working memory for logic or data blocks is changed when loading the system data if you change the following parameters:

- Size of the process image (byte-oriented; in the "Cycle/Clock Memory" tab)
- Communication resources (S7-400 only; "Memory" tab)
- Size of the diagnostic buffer ("Diagnostics/Clock" tab)
- Number of local data for all priority classes ("Memory" tab)

Basis for Calculating the Required Working Memory

To ensure that you do not exceed the available amount of working memory on the CPU, you must take into consideration the following memory requirements when assigning parameters:

Table 8-90 Memory requirements

Parameters	Required working memory	In code/data memory
Size of the process image (inputs)	12 bytes per byte in the process image input as of V6.0: 20 bytes per byte in the process image input	Code memory
Size of the process image (outputs)	12 bytes per byte in the process image input as of V6.0: 20 bytes per byte in the process image input	Code memory
Communication resources (communication jobs)	72 bytes per communication job	Code memory
Size of the diagnostic buffer	32 bytes per entry in the diagnostic buffer	Code memory
Quantity of local data	1 byte per byte of local data	Data memory

Flexible memory capacity

- Work memory:
  The capacity of the work memory is determined by selecting the appropriate CPU from the graded range of CPUs.
- Load memory:
  The integrated load memory is sufficient for small and medium-sized programs.
  The load memory can be increased for larger programs by inserting the RAM memory card. Flash memory cards are also available to ensure that programs are retained in the event of a power failure even without a backup battery. Flash memory cards (8 MB or more) are also suitable for sending and carrying out operating system updates.

Backup

- The backup battery provides backup power for the integrated and external part of the load memory, the data section of the working memory and the code section.

See also

What you should know about memory cards for S7-400 (Page 1653)

Contents of work and system memory

Address areas of the CPUs

Overview

Memory elements that each CPU makes available to the user program, such as the process image input and output, bit memory, timers and counters. The work or system memory of the S7-CPUs is divided into operand areas. Through use of appropriate operations in your user program, you address the data directly in the respective operand area.

The following table shows the operand areas:

Operand area	Access via units of the following size:	S7 notation	Description
Process image input	Input (bit)	I	At the start of every cycle, the CPU reads the inputs from the input modules and saves the values to the process image input.
	Input byte	IB	
	Input word	IW	
	Input double word	ID	

Operand area	Access via units of the following size:	S7 notation	Description
Process image output	Output (bit)	Q	During its cycle, the program calculates the values for the outputs and places them in the process image output. At the end of the cycle, the CPU writes the calculated output values to the output modules.
	Output byte	QB	
	Output word	QW	
	Output double word	QD	
Bit memory	Bit memory (bit)	M	This area provides storage for intermediate results calculated in the program.
	Memory byte	MB	
	Memory word	MW	
	Memory double word	MD	
Timer	Timer (T)	T	This area provides storage for timers.
Counter	Counter (C)	C	This area provides storage for counters.
Data block	Data block, opened with "OPN DB":	DB	Data blocks store information for the program. They can either be defined so that all code blocks can access them (global DBs) or assigned to a specific FB or SFB (instance DB).
	Data bit	DBX	
	Data byte	DBB	
	Data word	DBW	
	Data double word	DBD	
	Data block, opened with "OPN DI":	DI	
	Data bit	DIX	
	Data byte	DIB	
	Data word	DIW	
	Data double word	DID	
Local data	Local data bit	L	This area contains the temporary local data of a block while the block is being processed. The L stack also provides memory for transferring block parameters and for saving intermediate results from LAD networks.
	Local data byte	LB	
	Local data word	LW	

Operand area	Access via units of the following size:	S7 notation	Description
	Local data double word	LD	
I/O area: Inputs	Peripheral input byte	PIB	The peripheral input and output areas permit direct access to central and distributed input and output modules.
	Peripheral input word	PIW	
	Peripheral input double word	PID	
I/O area: Outputs	Peripheral output byte	PQB	
	Peripheral output word	PQW	
	Peripheral output double word	PQD	

process image input/output

Basic principles of process images

Function

When the user program addresses the input (I) and output (O) operand areas, it does not query or change the signal states on the digital signal modules. Instead, it accesses a memory area in the system memory of the CPU. This memory area is referred to as the process image.

Advantages of the process image

Compared with direct access to input and output modules, the main advantage of accessing the process image is that the CPU has a consistent image of the process signals for the duration of one program cycle. If a signal state on an input module changes during program execution, the signal state in the process image is retained until the process image is updated again in the next cycle. The process of repeatedly scanning an input signal within a user program ensures that consistent input information is always available.

Access to the process image also requires far less time than direct access to the signal modules since the process image is located in the internal memory of the CPU.

Updating the process images

Sequence

The process image inputs/outputs (OB 1 process image) are updated in the following order:

1. The internal tasks of the operating system are performed.
2. The process image output (PIQ) table is written to the outputs of the module.

3. The status of the inputs is read into the process image input (PII) table.
4. The user program is executed with all the blocks that are called in it.

The operating system automatically controls the writing of the process image output to the outputs of the modules and the reading of the process image input.

I/O access error during process image updating

S7-300 CPUs respond to an error during the update of the process image as follows:

- No diagnostics buffer entry, no OB call. The respective input bytes are set to "0" and remain at "0" until the error is resolved.

S7-400 CPUs respond to an error during process image update as follows:

- Diagnostics buffer entry and start of OB 85 with every instance of I/O access each time the corresponding process image is updated. The defective input bytes are set to "0" each time the process image is accessed.

You can set the reaction to I/O access errors by selecting one of the following alternatives:

- The CPU creates a diagnostics buffer entry and starts OB 85 each time the I/O is accessed.
- The CPU creates a diagnostics buffer entry and starts OB 85 only for incoming and outgoing I/O access errors. Before OB 85 is called, the faulty input bytes are set to "0" and are no longer overwritten by the operating system until the outgoing I/O access error.
- The CPU exhibits the default behavior of the S7-300. OB 85 is not called, the corresponding input bytes are set to "0" and are no longer overwritten by the operating system until the error is resolved.

Frequency of OB 85 starts

In addition to the assigned reaction to I/O access errors, the address space of a module also influences how often OB 85 starts:

- For a module with an address space up to a double word, OB 85 starts once, for example, for a digital module with up to 32 inputs or outputs, or for an analog module with two channels.
- For modules with a larger address space, OB 85 starts as often as access has to be made to it with double word commands, for example, twice for an analog module with four channels.

Local data stack

Function

When you create organization blocks, you can declare temporary tags (TEMP) that are only available when the block is executed and are then overwritten again. Each organization block also requires 20 bytes of local data for its start information.

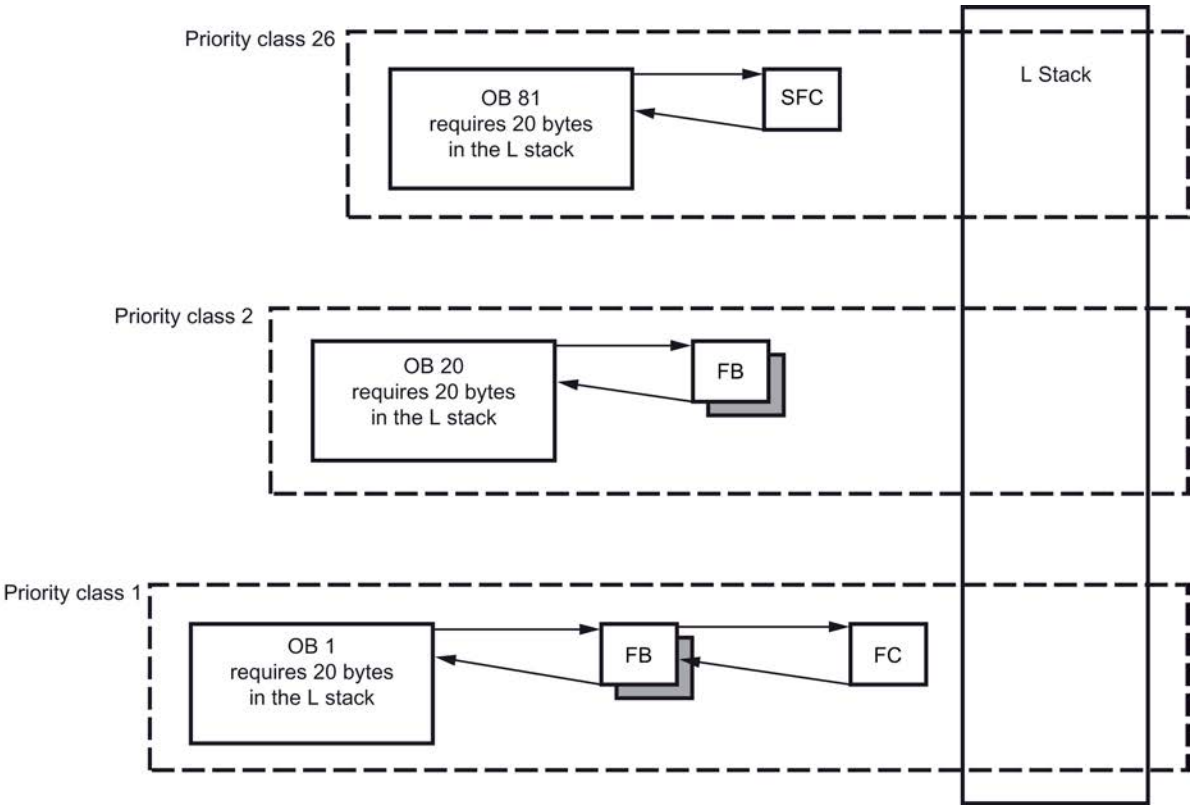
The CPU has a limited amount of memory for the temporary tags of the local data of blocks currently being executed. The size of this local memory, called the local data stack or L stack, depends on the particular CPU. By default, the local data stack is divided equally among the priority classes. This means that each priority class has its own area in the local data stack. This ensures that high-priority classes and their OBs have space available for their local data.

Before the local data stack is accessed for the first time, the local data must be initialized.

The local data stack stores the following data:

- Temporary tags of the local data of blocks
- Start information of the organization blocks
- Information regarding the transfer of parameters
- Intermediate results of the logic in ladder logic programs

The following figure shows the assignment of local data to the priority classes:



In the local data stack, OB 1 is first interrupted by OB 20, which in turn is interrupted by OB 81.

CAUTION

All temporary tags (TEMP) of an OB and its associated blocks are stored in the local data stack. If you use a large number of nesting levels when executing your blocks, the local data stack can overflow.

If you exceed the permissible local data stack size for a program, the S7-CPU changes to STOP operating mode.

You can get an overview of the required local data with the function "Show program information".

Don't forget the local data requirements of the synchronous error OBs.

Assigning local data to priority classes

For S7-300: The maximum amount of local data per priority class and per block is limited for the S7-300 (see the technical data of the CPU).

For S7-400: The maximum amount of local data in total is limited (see the technical data of the CPU). You distribute the available local data freely to the various priority classes within the framework of the CPU parameter assignment. The maximum amount of local data must not be exceeded in the process.

Interrupt stack

Function

If the program execution is interrupted by an organization block with higher priority, the operating system saves the following data in the interrupt stack (I stack):

- The current contents of the accumulators and address registers
- The number and size of the open data blocks

Once the new organization block has been executed, the operating system loads the information from the interrupt stack. The execution of the interrupted block then resumes at the place where it was interrupted.

Block stack

Function

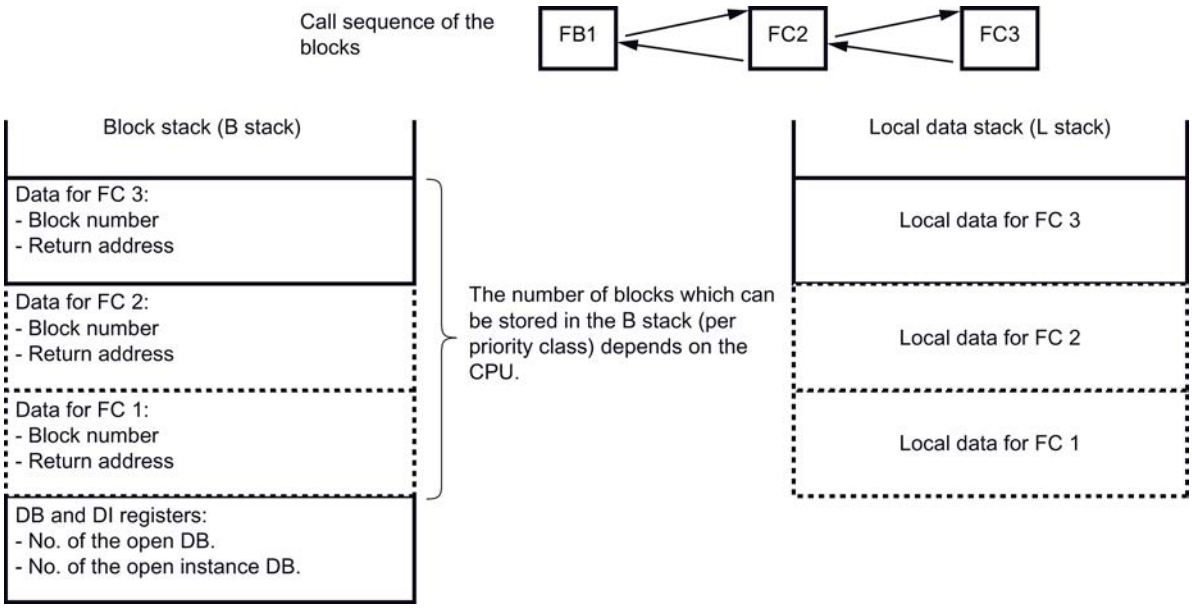
If processing of a block is interrupted by the call of another block or by a higher priority class, such as an interrupt or error handling, the operating system saves the following data in the block stack (B stack):

- Number, type (OB, FB, FC, SFB, SFC) and return address of the block that was interrupted.
- Numbers of the data blocks from the DB and DI registers that were open when the block was interrupted.

Using this data, the user program can be resumed after the interruption.

You can read the block stack in "STOP" mode on the programming device with the program editor open and using the test function "Call hierarchy". The block stack lists all the blocks that had not been completely executed when the CPU switched to STOP operating mode.

The following figure shows the order in which the blocks are called:



Data block register

Two data block registers are available. They contain the numbers of the data blocks that have been opened:

- The DB register contains the number of the opened global data block.
- The DI register contains the number of the opened instance data block.

Diagnostics buffer

Function

The diagnostics buffer contains system- and user-defined diagnostics events, entered in the order in which they occur. The first entry contains the latest event. The number of entries in the diagnostics buffer depends on the respective module and its current operating mode.

Possible diagnostics events:

- Faults on a module
- Faults in the process wiring
- System errors in the CPU
- Operating mode transitions of the CPU
- Errors in the user program
- User-defined diagnostics events using system function SFC 52

The information that is entered in the diagnostics buffer when a system diagnostics event occurs is identical to the start information that is transferred to the respective organization block.

The entries in the diagnostics buffer cannot be deleted; the content of the diagnostics buffer is retained even after a memory reset.

Advantages of the diagnostics buffer

The diagnostics buffer offers the following advantages:

- If the CPU changes to "STOP" mode, you can evaluate the last events leading up to the STOP and thus determine the cause of the STOP.
- You can recognize the causes of errors more quickly and thus increase the availability of the system.
- You can evaluate and optimize the dynamic system response.

Organization of the diagnostics buffer

The diagnostics buffer is designed as a ring buffer. The maximum number of entries depends on the module; for example, the S7-300 CPU 314 can have 100 entries. When this maximum number of entries is reached, the next diagnostics buffer event causes the oldest entry to be deleted. All entries then move back on place. This means that the newest entry is always the first entry in the diagnostics buffer.

The number of entries displayed in the diagnostics buffer depends on the module and its current operating mode. With some CPUs, it is possible to assign the length of the diagnostics buffer.

Evaluation of the diagnostics buffer

The contents of the diagnostics buffer can be accessed as follows:

- Online via the diagnostics view
- In the user program via SFC 51 "RDSYSST".

By default, the CPU automatically sends the last diagnostics buffer entry to a monitoring device (programming device, OP, or TD) before it changes from "RUN" to "STOP" mode. This allows you to quickly find out why the operating mode changed and rectify the cause. You can change this default.

Note

To make the best use of the time stamp information on the diagnostics buffer entries, set the date and time of day on the module and check the settings occasionally.

Timers and counters

Definition

Timers and counters are memory areas of the system memory. You specify the function of a timer or counter in the user program, for example the on-delay or the initial count value. The number of timers and counters available depends on the CPU.

Note

Note the following information on timers:

- If you use more timers and counters in your user program than the CPU permits, a synchronous error is signaled and organization block "OB 121" is started.

Access to the I/O data area

I/O data area

Structure of the distributed I/O data area

I/O data area

The I/O data area can be divided into:

- User data
- Diagnostics and parameter data

Both areas have an input area with read-only access and an output area with write-only access.

User data

Definition

User data can be:

- Digital and analog input and output signals from signal modules
- Control and status information from function modules
- Information for point-to-point and bus connections from communications modules

Access to user data

The user data in digital signal modules are addressed via the byte address of the input or output area.

For analog signal modules, the word address of the input or output area is used for addressing.

You can access user data via load and transfer commands or communications functions (operator control and monitoring) or via the process image transfer.

With load and transfer commands, it is possible to achieve a data consistency of 1, 2 and 4 bytes. Data from the process image are always consistent.

If you want to achieve data consistency of 3 or more than 4 bytes, you have to use the instruction "DPRD_DAT" (SFC 14) or "DPWR_DAT" (SFC 15).

To transfer 4 contiguous and unmodified bytes, use the instruction "Transfer double word". If you use four separate "transfer input byte" instructions, the consistency of the bytes cannot be maintained and unwanted results may occur.

See also

DPRD_DAT: Read consistent data of a DP standard slave (Page 6498)

DPWR_DAT: Write consistent data of a DP standard slave (Page 6500)

Diagnostic and parameter data

Data records of the diagnostics and parameter data

The diagnostics and parameter data for a module cannot be addressed individually. Rather, the data is always transferred in the form of complete data records. This means that consistent diagnostics and parameter data is always transferred.

Diagnostics and parameter data is addressed via the start address of the respective module and the data record number. Data records are divided into input and output data records. Input data records can only be read and output data records can only be written. You can access data records using system functions or communications functions.

The following table shows the assignment of data records to diagnostics and parameter data:

Data	Description
Diagnostics data	For modules with diagnostics capability, you obtain the diagnostics data of the module by reading data records 0 and 1.
Parameter data	For modules with parameters, you transfer the parameters to the module by writing data records.

Accessing the data records of a module

Instructions for accessing data records

You can use the information in the data records to reassign parameters for modules with parameters and to read out diagnostics information for modules with diagnostics capability.

The following table shows which functions you can use to access data records:

Instruction	Application
Assigning module parameters	
WR_PARM	Transfers the modifiable parameters (data record 1) to the addressed signal module
WR_DPARM	Transfers the individual parameter data records from SDBs 100 to 129 to the addressed signal module
PARM_MOD	Transfers all parameters from SDBs 100 to 129 to the addressed signal module
WR_REC	Transfers any data record to the addressed signal module
Reading out diagnostics information	
RD_REC	Reads out the diagnostics data

Note

Data records must not be read or written by the I/O modules in the user program with instruction "WR_REC/RD_REC" or instructions "WRREC/RDREC" if a DPV1 slave is configured using a GSD file (GSD from rev. 3) and the DP interface of the DP master (S7-400 CPU) is set to "S7-compatible". In this case the DP master would address the wrong slot, i.e. the configured slot +3. To avoid this problem, change the DP master interface to "DPV1".

Module start address

The module start address is the lowest byte address of a module. It represents the start address of the user data area in the module and is used in many cases to represent the entire module.

The module start address is entered in the start information of the appropriate organization block when, for example, process interrupts, diagnostics interrupts, insert/remove module interrupts, and power supply errors occur and thus identifies the interrupt-generating module.

Configuring module start addresses

The addresses used in the user program and the module start addresses are coordinated when the modules are configured.

You can change the start addresses automatically assigned when a module is inserted in the rack in the module properties ("Addresses" group).

User data area

Access to the I/O addresses

User data in the I/O operand area

The user data represent the I/O operand area. The following areas exist:

- Peripheral inputs (PI)
- Peripheral outputs (PQ)

If the user data are in the process image, the CPU automatically undertakes the data exchange between module and process image area when updating the process images.

If the user data are not in the process image, you must access them from the user program by direct I/O access to the data (e.g. L PIB 500).

The addresses of analog input and analog output modules are not normally in the process image area. You can place the addresses of analog modules in this area.

See also

Process images (input and output memory area) (Page 1742)

Direct access to the I/O (PI and PQ memory areas) (Page 1743)

Address areas

I/O address areas

There is a separate address area for inputs and outputs. This means that the address of an I/O area must not only include the byte or word information but also the "I" identifier for inputs and the "Q" identifier for outputs.

The following table lists the available I/O address areas:

Operand area	Access via units of the following size:	S7 notation
I/O area: Inputs	Peripheral input byte Peripheral input word Peripheral input double word	PIB PIW PID
I/O area: Outputs	Peripheral output byte Peripheral output word Peripheral output double word	PQB PQW PQD

See also

Direct access to the I/O (PI and PQ memory areas) (Page 1743)

Setting the operating behavior

Introduction to tuning the operational performance

Influencing the properties of automation systems

Properties of automation systems that are not preset can be influenced as follows:

- By setting the system parameters
- By using system functions (SFCs) in extended instructions

Changing module parameters

Default settings

In their as-delivered state, all modules with parameters have default settings suitable for standard applications. These default values allow the modules to be used immediately without making any additional settings.

You can, however, modify the behavior and the properties of the modules to suit the requirements and circumstances of your application. Modules for which you can set parameters include CPUs, FMs, CPs and some analog input and analog output modules and digital input and output modules.

Module parameters

Module parameters are grouped to improve clarity; for example, the following groups exist for CPUs:

- Startup
- Cycle
- PROFINET interface or other interface
- Diagnostics

- Retentive address area
- Clock memory
- Interrupts
- Protection
- Clock
- Connection resources

Settings and loading parameters

You can set the module parameters. When you save a device configuration with its parameters, system data is created and loaded to the CPU with the user program and then transferred to the modules during startup.

Assigning parameters with instructions

In addition to assigning parameters, you can also use advanced instructions in the program to modify module parameters.

The following table shows which instructions can change which module parameters:

System function	Application
WR_PARM (Page 6600)	Transfers the modifiable parameters (data record 1) to the addressed signal module.
WR_DPARM (Page 6601)	Transfers individual parameter data records from the corresponding SDBs to the addressed signal module.
PARM_MOD (Page 6596)	Transfers all parameters from the corresponding SDBs to the addressed signal module.
WR_REC (Page 6496)	Transfers any data record to the addressed signal module

General CPU parameters

General properties of modules

The "General" parameter group specifies the type and location of the module as well as information relevant to the network, e.g. the node address, in the case of modules with interfaces.

Short designation, order number/firmware, name, higher level item designation, and location designation

The short designation, the information below it, and the order number/firmware are identical to the information in the hardware catalog.

The "Name" field contains the name of the module. You can change this as required. If you change the name, the changed name appears both in the device view and in the network view.

CPUs allow you to specify a specific plant designation. The higher-level item designation is used to identify parts of the plant uniquely according to their function. It is hierarchically

structured in accordance with IEC 1346-1. You can also enter a location identifier of up to 32 characters. This is part of the item designation that identifies the exact location of, for example, a process tag or process tag group within a process (similar to a street on a city map).

The identifier entered in the "higher level item designation" or "location designation" field can be evaluated in the user program of the CPU. Use the option of entering a higher level item designation or location designation if you want to run the user program as a function of the higher level item designation of location designation.

Comment

Each module has a comments field. You can specify the purpose of the module in this field, for example.

Startup

Startup parameters

Startup behavior of the CPU

The parameters that determine the startup behavior of the S7-CPU are described below.

Startup when expected and actual configurations do not match

The preset configuration is represented by the configuration loaded to the CPU (system data blocks). The actual configuration is the actual configuration of the automation system.

If the "Startup when expected and actual configurations do not match" check box is cleared, the CPU changes to "STOP" in the following situations:

- If one or more modules are not inserted in their configured slot.
- If a module type that differs from the configured module type is inserted, e.g. during commissioning.
- If the distributed I/O is missing.

If additional modules (central or distributed) beyond those configured are present, they are disregarded when comparing the preset and actual configurations.

If the "Startup when expected and actual configurations do not match" check box is selected, the CPU will startup even in the cases listed above.

The I/O structure (central and distributed modules, central rack and DP slaves) is not checked by the CPU. PROFIBUS-DP interface modules are an exception. Because of their close connection to the CPU, configured modules of this type must always be inserted in order for the CPU to startup.

The "Startup when expected and actual configurations do not match" option is selected by default for the CPU. This means that the CPU will startup even if the preset and actual configurations are different.

Monitoring time for the "ready message" from the modules

Maximum wait time for the "ready" message of all configured modules after POWER ON. If after expiration of this monitoring time any module has not sent a "ready" message to the CPU, the actual configuration is different from the preset one. The CPU reacts to this based on the setting of the "Startup when expected and actual configurations do not match" parameter.

Within this monitoring time, for example, the various power supplies for a multi-tier configuration or for devices connected to the CPU via the DP master interface can be switched on in any order.

Monitoring time for the transfer of parameters to the modules

Maximum wait time for parameters to be "distributed" to modules with parameters (monitoring time starts after "Ready message from modules"). For CPUs with a DP master interface, you also use this parameter to specify the power-up time monitoring for the DP slaves. In other words,within this time the DP slaves must power-up and be assigned parameters by the CPU (as DP master).

If this monitoring time expires before all of the module/DP slave parameters have been assigned, the actual configuration is different from the preset one. The CPU reacts to this based on the setting of the "Startup when expected and actual configurations do not match" parameter.

Startup parameters only for S7-400 CPUs

Reset outputs at hot restart

Reset outputs at hot restart

S7-400 CPUs reset the outputs (and therefore also the process image of the outputs PIO) upon hot restart .

See also

Hot restart (Page 1643)

Disable hot restart at startup by means of operator control

Disable hot restart at startup by means of operator control (for example, programming device) or communication job (for example, from MPI stations).

You have the option of disabling "hot restart" startup mode for the S7-400 if it would otherwise be triggered by programming device operation or communication job. Hot restart can then only be executed locally.

- Select this checkbox if the startup mode triggered by operator control or communication job is to be limited to warm restart (=manual restart) and cold restart.
- All startup modes are possible if the checkbox is not selected.

Hot restart upon operator control or communication job (=manual hot restart) will be triggered under the following conditions:

- The "Disable hot restart ..." checkbox has not been selected and this configuration has been transferred to the CPU
- There is nothing to prevent hot restart (battery ok, no user program change in STOP mode ...)
- Hot restart operating mode is selected in the programming device or another device triggers hot restart with a communication job.

See also

Hot restart (Page 1643)

Cyclic behavior

Settings for the cycle behavior

Update OB1 process image cyclically

If this option is selected, the OB1 process image is updated cyclically (default). For S7-400-CPUs the OB1 updating can be switched off with these parameters.

Scan cycle monitoring time

The scan cycle monitoring time is set in ms.

If the cycle time exceeds the scan cycle monitoring time, the CPU switches to "STOP" mode. You can prevent "STOP" by programming organization block OB 80.

Possible causes for exceeding the scan cycle monitoring time include:

- Communications processes
- Accumulation of interrupt events
- CPU program error

Control of cycle loading by communications processes

The "Cycle load due to communication" parameter allows you to restrict the time taken by communications processes, which inevitably also extend the cycle time. Examples of communications processes include:

- Transferring data to another CPU
- Loading blocks (initiated via a programming device)

Test functions performed with the programming device are hardly influenced by this parameter, they can however significantly extend the cycle time. The time available for test functions can be restricted in process operation.

How the parameter works

The operating system of the CPU constantly provides the configured percentage of the overall CPU processing capacity for communications (time sharing). If the allocated processing capacity is not required for communications, it is made available to other processes.

Effect on the actual cycle time

In the absence of additional asynchronous events, the OB 1 cycle time is extended by a factor calculated by the following formula:

$$\frac{100}{100 - \text{"Cycle load due to communication"}}$$

Setting the size of the process image

For CPUs with variable process image size, the size of the process image (in bytes) can be set. The process image area always starts at input or output byte 0.

For S7-300-CPUs with variable process image size, there are special characteristics that must be observed during programming when accessing the process image area. These special characteristics are described in the documentation for the CPU.

Example: The variable setting of the process image currently takes effect only when the process image is updated at the cycle control point. The PII is updated only up to the set PII size, and only the set PIQ area is output to the output modules. User program accesses to the area between the assigned process image size and maximum process image size do not trigger a synchronous access error.

See also

Examples of cycle loading by communications (Page 1676)

Examples of cycle loading by communications

Below are two examples of how communications can affect the cycle loading.

Example of cycle load due to communications without additional asynchronous events

When you set the cycle load due to communication to 50%, the OB 1 cycle time can be doubled.

At the same time, the OB 1 cycle time is also influenced by asynchronous events (such as hardware interrupts or cyclic interrupts). From a statistical point of view, even more asynchronous events occur within the OB1 cycle because of the extension of the cycle time by the communication portion. This further extends the OB 1 cycle. This extension depends on how many events occur per OB 1 cycle and the duration of the event processing.

Example of cycle load due to communications with additional asynchronous events

For a pure OB 1 cycle time of 500 ms, a communications load of 50% can result in an actual cycle time of up to 1000 ms, assuming that the CPU has always enough communications jobs

to process. If, parallel to this, a cyclic interrupt with 20 ms processing time is executed every 100 ms, this cyclic interrupt would extend the cycle by a total of 5*20 ms = 100 ms without communication load. That is, the actual cycle time would be 600 ms. Because a cyclic interrupt also interrupts communications, it affects the cycle time by adding 10 * 20 ms at 50% communication load. That is, in this case, the actual cycle time amounts to 1200 ms instead of 1000 ms.

Note

Check the effects of changing the value of the "Cycle load due to communication" parameter while the system is running.

The communications load must always be taken into account when setting the minimum cycle time; otherwise, time errors will occur.

OB 85 call on a distributed I/O access error (DAE)

Settings for the OB 85 call

You can change the default reaction of the CPU to I/O access errors that occur while the system updates the process image.

Recommendation: If you want to activate the OB 85 call for I/O access errors, select the "only for incoming and outgoing errors" option, so that the CPU cycle time is not extended by repeated OB 85 calls.

In their default settings, S7-300-CPUs do not call OB 85 nor do they make an entry in the diagnostics buffer in reaction to I/O access errors.

Retentive behavior

Retentive settings for bit memory, timers, and counters

You use the retentivity parameters to specify the memory areas that are to be retained after a power failure or after a change from "STOP" to "RUN".

Number of memories bytes starting at MB 0

Enter the number of retentive memory bytes starting at memory byte 0.

Example

If you enter "2", memory bytes MB 0 and MB 1 are retentive.

Number of S7 timers starting at T0

Enter the number of retentive S7 timers (T) starting at T 0.

Each S7 timer occupies 2 bytes.

Example

If you enter "2", S7 timers T 0 and T 1 are retentive.

Number of S7 counters starting at C0

Enter the number of retentive S7 counters (C) starting at C 0.

Example

If you enter "2", S7 counters C 0 and C 1 are retentive.

Retention settings for data blocks

Function

In the retentivity properties, you can specify which memory areas of the CPU should be retained after a power failure and after a transition from "STOP" to "RUN", even without a battery backup in the backup memory of the CPU. Whether retentive areas can be configured and, if so, how many depends on the CPU.

If a backup battery is installed, data blocks are always retentive.

Retentive areas (relevant only for S7-300 without battery backup or micro memory card)

Retentive areas are data areas of data blocks.

If you data areas in the work memory of the CPU are to be retentive, enter the following:

- DB no.
  Numbers of the data blocks that are to be retentive.
- Byte address (start address with the length of the data bytes)
  Byte addresses of the data blocks that are to be retentive.
- Number of bytes
  Number of the retentive data bytes starting with the byte address. For example, if you enter "1", then data byte "0" is retentive.

Possible settings for interrupts

Settings for hardware interrupts

Hardware interrupts

The higher the priority selected, the higher the priority will be for processing the associated hardware interrupt. Priority "0" deselects the interrupt OB. The default priority cannot be changed in S7-300 CPUs.

Process image partitions can be assigned to a hardware interrupt, provided the CPU supports process image partitions. At the start of the hardware interrupt OB, the assigned process image partition is updated so that the OB uses current process values. If the setting is "---", a process image is not assigned.

See also

Hardware interrupt organization blocks (OB 40 to OB 47) (Page 1762)

Settings for time-delay interrupts

Time-delay interrupts

Time-delay interrupt OBs allows parts of the user program to be delayed in their execution.

The higher the priority selected, the higher the priority with which the associated time-delay interrupt will be executed. Priority "0" deselects the interrupt OB. The default priority cannot be changed in S7-300 CPUs.

Process image partitions can be assigned to a time-delay interrupt, provided the CPU supports process image partitions. At the start of the hardware interrupt OB, the assigned process image partition is updated so that the OB uses current process values. If the setting is "---", a process image is not assigned.

Time-delay interrupts are handled using the following instructions:

- SRT_DINT Start time-delay interrupt
- CAN_DINT Cancel time-delay interrupt
- QRY_DINT Query time-delay interrupt status

See also

Time-delay interrupt organization blocks (OB 20 to OB 23) (Page 1759)

Settings for asynchronous error interrupts

Asynchronous error interrupts

The default priority cannot be changed in S7-300 CPUs.

The priority of asynchronous error OBs can be changed so that the CPU can remains in "RUN" mode even under the following circumstances:

- Asynchronous errors accumulate, all triggering the same interrupt OB (e.g. diagnostics interrupt, call OB 82).
- In addition another asynchronous event occurs, triggering another interrupt OB (e.g. program execution error, call OB 85).
- Only one alarm should be issued (e.g. in the case of station failure, which station has failed). No process response is necessary. In this case. OB 86 can be assigned a low priority, so as not to increase the reaction time to other interrupts.

Example of why a priority change is necessary

Severing a string a signal cables could trigger a diagnostics interrupt (call OB 82) for every signal module that is affected - which would immediately fill the start information buffer for OB 82.

So that an event that calls another asynchronous error OB (such as a program execution error, OB 85) does not cause the CPU to go the STOP mode, you should assign, for example, OB 85 a higher priority than OB 82.

See also

Power supply error organization block (OB 81) (Page 1780)

Diagnostic interrupt OB (OB 82) (Page 1783)

Insert/remove module interrupt organization block (OB 83) (Page 1786)

CPU hardware fault organization block (OB 84) (Page 1790)

Priority class error organization block (OB 85) (Page 1791)

Rack failure organization block (OB 86) (Page 1796)

Communication error organization block (OB 87) (Page 1802)

Settings for time-of-day interrupts

Priority

The priority of time-of-day interrupts determines the interrupt behavior of the interrupt OBs. For S7-300 the priority cannot at present be changed.

Active

Select the check box of the time-of-day interrupt OB if this OB should be started automatically when a restart occurs. You can alternatively activate the time-of-day interrupt in your user program using the OB_NR parameter in the ACT_TINT instruction.

Execution

Select how often the interrupts should be executed.

The intervals **every minute** to **every year** are available.

The intervals relate to the settings for start date and time of day.

Note

For **execution** = **end of month** the day of the start date is irrelevant (always the last day of the month).

Start time

Enter the date and time at which the interrupt is to be executed for the first time.

Example

Start date	Time-of-day
30.03.2006	10:15

Depending on the "Execution" parameter, the CPU will create further time-of-day interrupts at the specified intervals. The start time always relates to the base time of the CPU.

Note

If you selected "monthly" execution, the start date cannot be the 29th, 30th, or 31st, because then no time-of-day interrupt would occur in the month of February.

Instead, select "End of month" execution! In this case the day of the start date is not relevant (always the last day of the month!).

Process image partition

A process image partition can be assigned to a time-of-day interrupt, provided the CPU supports process image partitions. At the start of the interrupt OB, the assigned process image partition is updated so that the OB uses current process values. If the setting is "---", a process image partition is not assigned.

Special case: the one-time start time configured is in the past

The following configuration exists:

- Execution: once
- Active: Yes
- Start date/Time-of-day: is in the past (in relation to the CPU real-time clock)

Behavior of the CPU: After a cold restart or warm restart, the corresponding time-of-day interrupt OB is **called once** by the operating system.

See also

Time-of-day interrupt organization blocks (OB 10 to OB 17) (Page 1755)

Setting time interrupts

A cyclic interrupt is a periodic signal that leads to the automatic calling of a cyclic interrupt OB. The available settings are detailed below.

Priority

The priority determines the interrupt behavior of the interrupt OB. In the S7-300 CPUs it cannot be changed.

Execution

Enter the time intervals in the units shown (e.g. milliseconds) for execution of cyclic interrupt OBs.

The time cycle starts when the operating mode changes from "STOP" to "RUN".

Phase offset

If the phase offset is adjustable: Enter the time in milliseconds (ms) by which the actual execution time of the cyclic interrupt should be delayed.

Note

A phase offset is sensible if several cyclic interrupts are active. Without phase offset, cyclic interrupts would be called simultaneously and executed in succession depending on their priority.

The phase offset allows the execution time of the cyclic interrupts to be distributed across the cycle.

Unit

Refers to the "Execution" and "Phase offset" parameters. This conversion is available only on special CPUs that support cyclic interrupt settings in microseconds.

Process image partition

A process image partition can be assigned to a cyclic interrupt, provided the CPU supports process image partitions. At the start of the cyclic interrupt OB, the assigned process image partition is updated so that the OB works with current process values. If the setting is "---", a process image is not assigned.

See also

Cyclic interrupt organization blocks (OB 30 to OB 38) (Page 1760)

Settings for the DPV1 interrupts

Interrupts for DPV1

Interrupts can be originated by a DPV1 slave to ensure that the master CPU handles the interrupt-triggering event. The interrupt data are evaluated in the CPU even in STOP mode (update of diagnostics buffer and module status). However an OB is not executed in "STOP" mode.

In addition to the normal SIMATIC interrupts (such as diagnostics interrupt for ET 200M), interrupts such as status interrupts, update interrupts, and manufacturer-specific interrupts are supported.

See also

Status interrupt OB (OB 55) (Page 1764)

Update interrupt OB (OB 56) (Page 1768)

OB for manufacturer-specific alarms (OB 57) (Page 1772)

Setting the isochrone mode interrupts

Properties of isochronous mode interrupts

Isochronous mode interrupt OBs are executed synchronously with the equidistant clock cycle.

DP master system no.

Select the DP master system whose DP slaves should be assigned to the interrupt OB. The inputs and outputs of these DP slaves can then be synchronized with the equidistant clock cycle and the user program in the interrupt OB.

Process image partition(s)

Enter here the numbers of the process image partitions that the interrupt OB should access. The assigned distributed I/O are updated synchronously with the equidistant clock cycle via the process image partitions.

The assigned process image partitions must be updated in the synchronous cycle interrupt OB using the instructions SYNC_PI" and "SYNC_PO". The instruction "SYNC_PI" updates the input process image partition and the instruction "SYNC_PO" the output process image partition.

If the distributed I/O addresses are assigned to different process image partitions, the numbers of the process image partitions used must be entered separated by commas (e.g. 1,2,3).

Delay time

The delay time is the time between the global control telegram and the start of OB 61 (or OB 62 to OB 64). During this time, the DP master performs the cyclic data exchange with the DP slaves.

If the delay time = 0 is, OB 6x starts with the global control telegram. It is in this case extremely likely that access to the isochronous I/O via "SYNC_PI" at the start of the OB will fail as the I/O data is not yet available.

If the delay time is larger than the default value, then the OB runtime is "given away", since the clock-synchronized I/O could have been accessed earlier.

See also

Synchronous cycle interrupt OBs (OB 61 to OB 64) (Page 1776)

Time-of-day functions

Basic principles of time of day functions

All S7-300/400 CPUs are equipped with either a real-time clock or a software clock. The clock can act in the automation system as both a time-of-day master and as a slave with external synchronization. This allows the use of time-of-day interrupts and run-time meters.

Time-of-day format

The clock always shows the time of day with a minimum resolution of 1 second and the date including the day of the week. Some CPUs also permit the display of milliseconds.

Setting and reading the time of day

Setting the time of day

You can set and start the time of day and date of the CPU clock using the following instructions:

- "SET_CLK"
- "SET_CLKS" if this is available in the CPU

Note

To avoid time-of-day display differences in HMI systems, set the CPU to standard time or use central time-of-day synchronization.

Reading the time of day

You read the current date and time of day of the CPU with the instruction "READ_CLK" or via a menu command from the programming device.

Time synchronization

Reference

Parameter assignment of the interface properties in the following parameter groups:

- For the CPU: "Properties > General > Time"
- For the DP interface: "Properties > General > DP interface > Time synchronization".

Clock parameters

The clock parameters allow you to make the following settings:

- Synchronization type
  You can set the synchronization properties for the CPU internally (PLC) and externally (MPI).
  If you have set the synchronization mode to "As master", you can then select the time interval for automatic synchronization.
  If more than one module with a clock is present within a network, you need to specify which CPU will act as the master and which as the slave for time synchronization purposes. There can only be one time master.
- Update interval
  The update interval defines the interval between time queries.
- Correction factor
  The correction factor is used for the CPU to correct any variation of the clock within a 24 hour period.
  You can enter positive or negative value in ms.
  Example: If in 24 hours the clock has lost 4 seconds, you must enter a correction factor of "+4000 ms".

Instructions for synchronizing the time

To ensure that the time is consistent in all modules in the network, the time slaves are updated by the system program at regular configurable intervals. Instruction "SNC_RTCB" allows the date and time to be transferred from the time master to the time slave independently of the synchronization interval configured.

Using an operating hours counter

Definition

An run-time meter counts the operating hours of connected equipment or the total runtime of the CPU. A CPU can have up to 8 run-time meters. The numbering starts at 0.

In "STOP" mode, the run-time meter is stopped. Its count value is retained even after a memory reset. During a "warm restart", the run-time meter must be restarted by the user program.

Programming run-time meters

You can program the runtime meters using the following instructions:

- Use instruction "SET_RTM" to set the runtime meter to an initial value.
- Use instruction "CTRL_RTM" to start or stop the runtime meter.
- Use the instruction "READ_RTM" to read out the current number of operating hours and the status of the counter - either "stopped" or "counting".

Use of clock bit memories

Clock memory

A clock memory is a bit memory that changes its binary status periodically in the pulse-no-pulse ratio of 1:1.

You decide which memory byte of the CPU will become the clock memory byte when assigning the clock memory parameters.

Benefits

You can use clock memory, for example, to activate flashing indicator lamps or to initiate periodically recurring operations such as recording of actual values.

Available frequencies

Each bit of the clock bit memory byte is assigned a frequency. The following table shows the assignment:

Bit of the clock memory byte	7	6	5	4	3	2	1	0
Period (s)	2.0	1.6	1.0	0,8	0.5	0,4	0.2	0.1
Frequency (Hz)	0.5	0.625	1	1.25	2	2.5	5	10

Note

Clock memory runs asynchronously to the CPU cycle, i.e. the status of the clock memory can change several times during a long cycle.

The selected memory byte cannot be used for intermediate storage of data.

Web settings

In this parameter group you can select:

- CPU Webserver start
- The languages in which texts are to be loaded (language-dependent texts are for example, diagnostics buffer entries and alarms).

Activating the Web server on this module

If the check box is selected, after the configuration data has been loaded, the Web server of the CPU is started and information can be read from the CPU via a Web browser.

Further settings relate to the update of web pages displayed. You can if required set the update time.

Only allow access via HTTPS

Enable this radio button for the encrypted transfer of web pages from the web server to the web browser.

Note

Access via HTTPS will only work if the time has been set in the CPU!

Selecting the languages

Select the national languages for language-dependent texts to be loaded to the CPU. The number of languages available is CPU-dependent.

Display and selection of CPU-specific languages

The display shows the languages which can be set for web access to the CPU. You can select a number of project languages for this "web server language" selection for a given CPU. Texts of diagnostics buffer entries and event texts are loaded into the CPU for the selected languages. For these languages, the information is also displayed on the Web browser in the corresponding language during operation.

If, during operation, you select a language in the web browser that has not yet been loaded, the diagnostics buffer entries and event texts will be shown in hexadecimal.

Note that each language requires memory space on the CPU.

User administration

You can define users for access to the CPU via the web browser and assign these users rights and passwords.

Rights can also be defined for the default user who does not log on to the web browser with a password ("everybody").

Monitoring tables

You can give users access via the web browser to the monitoring tables you define. Select the existing tables from the drop-down list and specify in the drop-down in the second column whether read-only or write access is to be possible.

Display classes of the messages

The selected message display classes are subsequently displayed on the CPU web pages.

You can reduce the memory occupied by the web SDBs by selecting only those display classes which are actually required.

The display class is set in the CPU properties "Message attributes" under "System diagnostics" in the inspector window.

User-defined web pages

The section "User-defined web pages" allows you to load your own web pages to the CPU and therefore make your own web applications available via the web browser.

See also

Information about the web server (Page 903)

Create and download user-defined websites (Page 913)

Reservation of communications resources

Assigning connection resources

Each communications connection requires a connection resource on the S7 CPU as a management element for the duration of the communications connection. As listed in the technical specifications, each S7 CPU has a certain number of connection resources available, which are assigned to the various communications services (programming device/OP communication, S7 communication or S7 basic communication).

Assignment of connection resources according to chronological order

When communications services log on, the connection resources are assigned according to the order in which they logged on.

Manual assignment of communications resources

In order to avoid allocation of the connection resources being dependent only on the chronological order in which various communication services are requested, connection resources can be reserved for the following services:

- Programming device communication and OP communication
- S7 basic communication

At least one connection resource is reserved by default for programming device communication and for OP communication. Smaller values are not possible.

Other communications services such as S7 communication with the extended instructions "PUT (Page 7368)" and "GET (Page 7366)" cannot occupy this connection resource, even if they establish their connection ahead of time. Instead, they are assigned other available connection resources, i.e. those that have not been specially reserved for a service.

Information on the maximum number and already occupied connection resources

The CPU properties display the following additional information regarding reservation of connection resources (read-only):

- Number of resources already occupied by S7 communication. These are resources such as for one-way S7 connections from another CPU to this CPU or for two-way S7 connections. These connections have already been configured in the network view.
- Maximum number of connection resources for this CPU.

Test, commissioning and routing

Test, commissioning, routing

Select the check box if, for example, you want to use PG functions regularly over this interface for commissioning and testing.

If you select the "Test, commissioning, routing" option, the interface becomes the active device on PROFIBUS (in other words, the interface is included in the token rotation of PROFIBUS). The following functions are then possible:

- Programming (for example loading)
- Testing (status/modify)
- S7 routing (I-slave as gateway)

This can extend the token rotation time. In time-critical applications and when S7 routing and client functionality are not required for communication, this option should therefore not be enabled.

If the "Test, commissioning, routing" check box is not selected, the interface functions only as a server for communication services.

Server functionality means that only the partner (for example PG, OP or automation system) can initiate communication. Note the following features when the "Test, commissioning, routing" option is not enabled:

- S7 routing is not possible
- Programming and testing are possible but slow (no extension of the bus token rotation time)

Settings for system diagnostics

The diagnostics system is the recording, evaluation and reporting of errors within the automation system.

Examples of errors

- CPU program error
- Failures of modules
- Wire break for sensors and actuators

Report cause of STOP

Select the check box to display the reason why a CPU changed to "STOP" mode on the display device (programming device or OP).

The alarm is also entered in the diagnostics buffer.

Alarms triggered by acknowledgment of the ALARM extended instruction

This parameter is relevant to CPUs that are used for process control systems. When this function is enabled, the extended instructions "ALARM (Page 6649)", "ALARM_8 (Page 6641)" and "ALARM_8P (Page 6643)" (SFBs 33 to 35) only report a signal change again if the previous signal change (i.e., the previous incoming alarm) was acknowledged. This prevents the accumulation of alarms that cannot be processed.

Setting options for the level of protection

Protection level

This section explains how to use the individual protection levels of the S7 CPUs and what influence the type of S7 CPU has on the parameter assignment and use of the respective protection level.

Default behavior

There is no password assignment in the default protection level behavior. This level of protection means "No protection".

A password can only be given if you have selected the option "Removable with password". This option gives you write access to the CPU in RUN mode if the CPU has been write-protected with the "PROTECT" instruction in the user program.

Note

If the default protection level of your CPU is active and your CPU supports the instruction "PROTECT (Page 5712)", you can use this instruction to switch between the default protection level "No protection" and the protection level "Write protection".

Additional password-protected protection levels

A password is required for the following protection levels:

- Write protection: With this protection level, only read-only access is available without entering the correct password, regardless of the keyswitch setting. To enable this protection level a password is required.
- Write/read protection: With this protection level, neither read nor write access is available without entering the correct password, regardless of the keyswitch setting. To enable this protection level a password is required.

Write and read access is possible if the password is known regardless of the level of protection set.

Behavior of a password-protected module during operation

The CPU protection takes effect after the settings are downloaded in the CPU

Before an online function is executed, the necessary permission is checked and, if necessary, the user is prompted to enter a password.

Example: The module was assigned a protection level and you want to execute the "Modify tags" function. This requires write access; therefore, the assigned password must be entered to execute the function.

The functions protected by a password can only be executed by one programming device/PC at any one time. Another programming device/PC cannot log on with a password.

Access authorization to the protected data is in effect for the duration of the online connection, or until the access authorization is manually rescinded with "Online > Delete access rights".

Note

You can not restrict functions for process control, monitoring, and communications. Thus, for example, the "Set time of day/date" function cannot be locked with a password.

Test mode and process mode

Introduction

For some CPU models you can set the mode for the program test.

In process operation, test functions such as program status or monitor/modify tags are restricted so that the set permissible cycle time extension is not exceeded. Testing using breakpoints and step-by-step program execution cannot be performed.

In test operation, all test functions via a programming device/PC – even if they increase the cycle time significantly – can be used.

Possible settings for operating behavior

You can set the following modes for testing with the program status in the inspector window under "Properties > Mode":

- Test mode
  All test functions can be used without restrictions. The CPU cycle time can be extended considerably since, for example, the status of instructions in programmed loops is checked each time they are run through.
- Process mode
  The program status test function is restricted to ensure that the cycle time load is kept as low as possible:
  - For example, no call conditions are permitted.
  - The status display of a programmed loop is stopped at the return point.
  - The "HOLD" and step-by-step program execution test functions cannot be used.

Note

If the CPU is in test operation you must ensure that the CPU and the process can handle significant cycle time increases.

The "process operation" option allows you to prevent changes to the call environment being made in the program editor.

Parameters of the compact CPU

Counting function

Properties of Counting

Description

If you select the CPU with integrated count and measurement function in the network view or the device view, you can configure the counting function for each of the channels listed in the inspector window under "Properties > General > Count".

If you decide to use channel 0 or channel 1 as the counting channel you can no longer use "Positioning" technology.

Note

The number of channels you can utilize as counting channels depends on the type of CPU you are using.

In the "Operating Mode" selection box, you can determine how you want the CPU to count for each channel. There are various options for this. Default is "not configured".

See also

Documentation on technological functions of the CPU (http://support.automation.siemens.com/WW/view/de/12429336/0/en)

Interrupt selection

You can select several interrupts for the "Counting" function.

Requirement

- The CPU is selected.
- Interrupt selection is located at "Properties > General > Counting > Interrupt selection".

Interrupt selection (basic parameters)

In the "Select interrupt" drop-down list, select which interrupts the CPU should trigger. You have the following options:

- None
- Diagnostics
- Process
- Diagnostics and hardware

Default selection is "None".

Select the events you want to use for hardware interrupt triggering, dependent on the set operating mode.

Counting

Events that can trigger a hardware interrupt:

- HW gate opening (with opened SW gate)
- HW gate closing (with opened SW gate)
- On reaching (activating) the comparator
- On a counting pulse
- Overflow (when the high count limit is exceeded)
- Underflow (when the low count limit is fallen below)

Frequency measurement

Events that can trigger a hardware interrupt:

- HW gate opening (with opened SW gate)
- HW gate closing (with opened SW gate)
- End of measurement
- High limit violated
- Low limit violated

Pulse-width modulation

Events that can trigger a hardware interrupt:

- HW gate opening (with opened SW gate)

Channel-specific parameters

General

All channel-specific parameters are set under "Properties > General > Counting > Channel 0 (... Channel n)". There are six options available for this operating mode: "Not configured", "Count continuously", "Count once", "Count periodically", "Measure frequency" and "Pulse-width modulation".

"Not configured" is set as the default.

Count

Operating parameters for counting

Main counting direction

In "Count once" and "Count periodically" modes, you can specify from the "Main count direction" drop-down list whether or not you want the counter to have a main count direction and, if so, whether the main count direction is up or down. Default selection is "None".

The "Main counting direction" drop-down list is not shown in the "Count continuously" operating mode.

Gate Function

In the "Gate function" drop-down list, you can select whether the count operation is to be canceled or interrupted when the gate is closed.

Here, if you select "Cancel count", the count is resumed with the load value after the gate is closed and opened again. Here, if you select "Stop count", the count is resumed with the last actual counter value after the gate is closed and opened again.

Default selection is "Cancel count".

Initial value / End value

In the Initial value / End value input field, enter a value for the respective counting limit, dependent on which main count direction you have selected in the respective drop-down list.

The main count direction and this entry are related as follows:

- If the main count direction is up, enter an end value here.
- If the main count direction is down, enter an initial value here.

This input field is not shown if you have not selected a main count direction.

The default setting always has the value "2147483647" (only visible if you have selected a main counting direction).

Compare Value

In the "Comparison value" input field, enter a value that is assigned to the digital output, to the comparator status bit STS_CMP and to a hardware interrupt. The count value is compared with the comparison value. Dependent on the result of this comparison, the output and the comparator status bit STS_CMP can be set and the hardware interrupt for "Reached comparison value" can be generated.

In the output group, customize how the output is actually set dependent on the counter value.

Default comparison value is "0".

Hysteresis

In the input field "Hysteresis" enter a hysteresis range of 0 to 255 (0 and 1 means: hysteresis is switched off).

An encoder might stop at a certain position and then "swing" up / down in this position. As a result, the counter value fluctuates around a certain value. Thus, the associated output would be switched on / off, dictated by the rhythm of those fluctuations if, for example, a comparison value lies within this fluctuation range. In order to prevent a switching action as a result of small fluctuations, the CPU offers a configurable hysteresis.

You can modify this hysteresis in a current operation via the user program.

The hysteresis influences the output and the comparator status bits STS_CMP, the overflow STS_OFLW, underflow STS_UFLW and the zero crossing STS_ZP.

Default is "0".

Time base

You can specify whether the period is to be measured in units of 125 ns or 1 µs at a maximum counting frequency of 1 kHz. If the maximum counting frequency is greater than 1 kHz, the period is not measured. You can measure the period of the counting signal up to a maximum counting frequency of 1 kHz.

At a maximum counting frequency of 1 kHz, the time between two consecutive counting edges is always measured.

With a time base of 1 µs, periods of up to 4294 s (71 min, 34 s) can be measured and with a time base of 125 ns, periods of up to 178 ms (2 min, 58 s) can be measured.

If the counting edges are further apart in time, the measured period is incorrect because a overflow has not been taken into account.

Input 0 (count)

Signal evaluation

In the "Signal evaluation" drop-down list, select which signals the connected encoder is to deliver. You can choose from the following options:

- Pulse and direction
- Rotary encoder single
- Rotary encoder double
- Rotary encoder quadruple

The default setting is "Pulse and direction".

HW gate

The "HW gate" check box allows you to select whether or not a hardware gate is to be used.

This check box is not selected by default.

Count Direction Inverted

The "Count direction inverted" check box allows you to select whether or not the count direction is to be inverted.

This check box is not selected by default.

Output (count)

Output function course

The "Output function course" drop-down list offers you the following alternative options:

- No comparison
  The output will be switched in the same way as a normal output.
  The SFB input parameters CTRL_DO and SET_DO are not active.
  The status bit STS_DO and STS_CMP (status comparator in the IDB) remain reset.
- Set when count ≥ comparison value
  The output is set when the counter value reaches the compare value. It stays set as long as the counter value is greater than / equal to the compare value.
- Set when count ≤ comparison value
  The output is set when the counter value reaches the compare value. It stays set as long as the counter value is less than / equal to the compare value.
- Pulse when comparison value is reached
  You can specify a pulse period for the adaptation to the actuators you are using. The output is set for the specified pulse period when the counter value reaches the compare value. The output is only set when it reaches the compare value of the main counting direction, if you have configured a main counting direction.

Default is "No comparison".

Pulse Period

If you have selected the "Pulse when comparison value is reached" alternative in the "Output function course" drop-down list, you must enter the pulse period for the output signal in ms in the "Pulse period" input field. The maximum length is 510 ms. You can only enter even values.

The pulse period starts when the respective digital output is set. The pulse period inaccuracy is lower than 2 ms.

The output stays set until the compare conditions do not apply anymore if you have entered a 0-ms pulse period.

Default is "0 ms".

Note

The bit STS_CMP "Comparator Status" is set together with the output when the compare value is reached.

Max. frequency

Counting signals / HW gate

In this drop-down list you can set the maximum counting frequency for the track A / pulse, track B / direction and hardware gate signals in fixed intervals. The maximum value depends on the CPU in use:

CPU	Range of values	Default selection
CPU 312C	10, 5, 2, 1 kHz	10 kHz
CPU 313C CPU 313C-2 DP CPU 313C-2 PtP	30, 10, 5, 2, 1 kHz	30 kHz
CPU 314C-2 DP CPU 314C-2 PN/DP CPU 314C-2 PtP	60, 30, 10, 5, 2, 1 kHz	60 kHz

If your CPU does not support this function, the drop-down list in unavailable and grayed out.

Latch

In this drop-down list you can set the maximum counting frequency for the latch signal in fixed intervals. The maximum value depends on the CPU in use:

CPU	Range of values	Default selection
CPU 312C	10, 5, 2, 1 kHz	10 kHz
CPU 313C CPU 313C-2 DP CPU 313C-2 PtP	30, 10, 5, 2, 1 kHz	10 kHz
CPU 314C-2 DP CPU 314C-2 PN/DP CPU 314C-2 PtP	60, 30, 10, 5, 2, 1 kHz	10 kHz

If your CPU does not support this function, the drop-down list in unavailable and grayed out.

Hardware interrupt at (count)

Description

The corresponding event triggers a hardware interrupt when you select the respective button. The following events can trigger a hardware interrupt:

- HW gate opening (with opened SW gate)
- HW gate closing (with opened SW gate)
- On reaching (activating) the comparator
- On a counting pulse

- Overflow (when the high count limit is exceeded)
- Underflow (when the low count limit is fallen below)

You must enable this in the Basic Parameters tab to ensure that the CPU generates a hardware interrupt.

No option is selected by default.

Triggering a hardware interrupt on each counting edge results in high CPU utilization at higher counting frequencies. If the hardware interrupts in the "Counting" submodule occur faster than they can be processed in the hardware interrupt OB (OB 40), this produces the diagnostic "Hardware interrupt lost), provided the diagnostic interrupt is enabled.

Assigning input data

Assigning input data

You can select whether the count value or the period can be read at a maximum counting frequency of 1 kHz in the input data (I data) of the "Counting" submodule. If the maximum counting frequency is greater than 1 kHz, only "Count value" is possible.

You can measure the period of the counting signal up to a maximum counting frequency of 1 kHz.

At a maximum counting frequency of 1 kHz, the time between two consecutive counting edges is always measured. If the maximum counting frequency is greater than 1 kHz, the period is not measured and the value is 0.

Frequency measurement

Measure Frequency

Description

In this operating mode the CPU counts the pulses it receives with default time intervals (integration time) and calculates the frequency via the number of pulses and integration time.

The calculated frequency value is supplied in "mHz" units.

Operating parameters for frequency measurement

Description

In this operating mode the CPU counts the pulses it receives within a configured integration time.

The measurement is carried out within this integration time. The measured value is updated when the integration time has expired.

If at least one rising edge does not occur during the integration time, a value of 0 or the averaged value is returned as the measured value, depending on the parameter assignment.

Integration time

In the "Integration time" input field, enter an integration time between 10 ms and 10000 ms.

Default is "100 ms".

Low limit and high limit

In the "Low limit" and "High limit" input fields, enter the values that the determined frequency is compared with, respectively, after the integration time has expired. The following are permissible ranges for limit value monitoring, dependent on the type of CPU:

- Low limit
  - CPU 312C: 0 to 9 999 999 mHz
  - CPU 313C: 0 to 29 999 999 mHz
  - CPU 314C: 0 to 59 999 999 mHz
- High limit
  - CPU 312C: 1 mHz to 10 000 000 mHz
  - CPU 313C: 1 mHz to 30 000 000 mHz
  - CPU 314C: 1 mHz to 60 000 000 mHz

Note

At the time of input it is verified whether the high limit > low limit.

Default for the low limits is 0 mHz respectively. CPU-dependent default high limits are:

- CPU 312C: 10 000 000 mHz
- CPU 313C: 30 000 000 mHz
- CPU 314C: 60 000 000 mHz

Max. counting frequency / HW gate

In this drop-down list you can set the maximum counting frequency for the track A / pulse, track B / direction and hardware gate signals in fixed intervals. The maximum value depends on the CPU in use:

CPU	Range of values	Default selection
CPU 312C	10, 5, 2, 1 kHz	10 kHz
CPU 313C CPU 313C-2 DP CPU 313C-2 PtP	30, 10, 5, 2, 1 kHz	30 kHz
CPU 314C-2 DP CPU 314C-2 PN/DP CPU 314C-2 PtP	60, 30, 10, 5, 2, 1 kHz	60 kHz

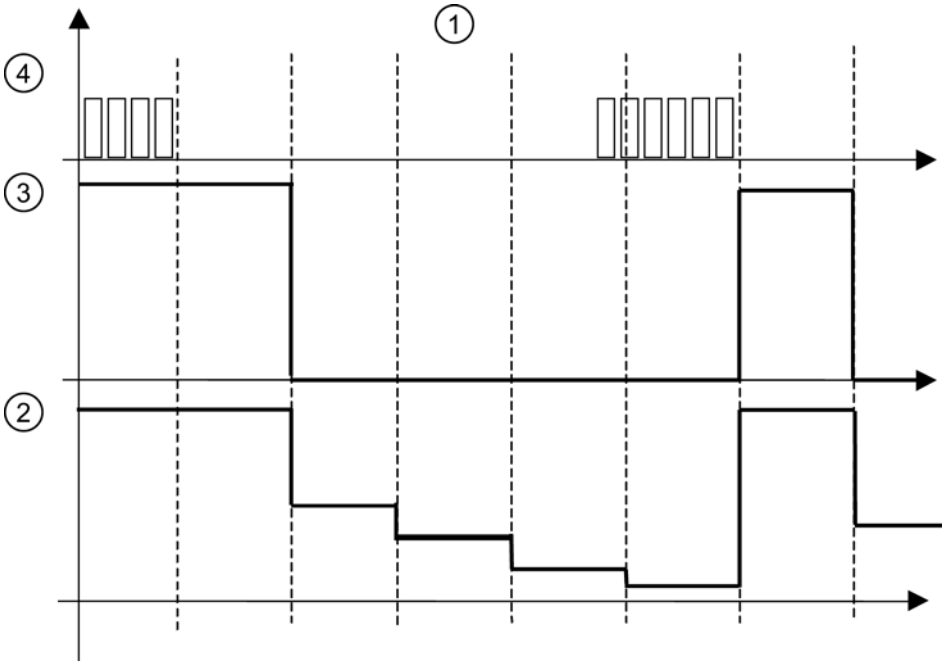
Output measured value

You can select whether the measured value is to be output directly or as an averaged value. To do this, select the "Directly" or "Averaged" option in the "Output measured value" drop-down list.

The measured frequency is displayed at the end of an integration time within which at least one positive edge has occurred.

If at least one positive edge does not occur during the assigned integration time:

- for direct frequency value, "0" is returned at the end of the integration time.
- for averaged frequency value, the last value is exported to subsequent measurement intervals without positive edge (f ≥ 1 mHz). This operation represents an extension of the integration time. Here, the last measured value is divided by the number of measurement intervals without positive edge.
  Example: If the last measured value was 12 000 mHz, the value 4000 mHz is output after three measurement intervals.



① Time interval
② Frequency averaged
③ Frequency direct
④ Pulses

The "Direct" option is selected as the default in the drop-down list.

Input 0 (frequency measurement)

Signal evaluation

In the "Count direction inverted" drop-down list, select which signals the connected encoder is to supply. You can choose from the following options:

- Pulse and direction
- Rotary encoder single

The default setting is "Pulse and direction".

HW gate

The "HW gate" check box allows you to select whether or not a hardware gate is to be used.

This check box is not selected by default.

Count Direction Inverted

The "Count direction inverted" check box allows you to select whether or not the count direction is to be inverted.

This check box is not selected by default.

Output (frequency measurement)

Output function course

The "Output function course" drop-down list offers you the following alternative options:

- No comparison
  The output will be switched in the same way as a normal output.
  The SFB input parameters MAN_DO and SET_DO are not active.
  The status bit STS_DO remains reset.
- Set when outside the limits
  The output is set when the measured frequency is higher than the upper limit or lower than the lower limit.
- Set when lower limit is violated
  The output is set when the measured frequency drops below the lower limit.
- Set when upper limit is violated
  The output is set when the measured frequency exceeds the upper limit.

Default is "No comparison".

Hardware interrupt at (frequency measurement)

Description

The corresponding event triggers a hardware interrupt when you select the respective check box. The following events can trigger a hardware interrupt:

- HW gate opening (with opened SW gate)
- HW gate closing (with opened SW gate)
- End of measurement
- High limit exceeded
- Low limit violated

You must enable this in the Basic Parameters tab to ensure that the CPU generates a hardware interrupt.

No option is selected by default.

Pulse-width modulation

Pulse-Width Modulation

Description

The output value you have specified is converted by the CPU into a pulse sequence of the respective pulse/pulse ratio (pulse-width modulation).

This pulse sequence is output on digital output DO (output sequence) after the parameterized on-delay has expired.

Operating parameters for pulse width modulation

Output format

In the"Output format" drop-down list, select the value range of the output value by selecting the "ppm" or "S7 analog value" option.

If your output value lies between 0 and 1,000 check the "ppm" option.

If your output value is a SIMATIC S7 analog value (between 0 and 27,648), check the "S7 analog value" option.

The "ppm" button is selected by default.

Time base

In the "Time base" drop-down list, select the time base valid for the resolution, value range of the period, minimum pulse width and for the on-delay.

If you select the "1 ms" option, you can set times with a resolution of 1 ms.

Select "0.1 ms" option to configure times with a resolution of 0.1 ms.

The "0.1 ms" option is selected by default.

On delay

In the "On delay" input field, enter a value for the delay time between the start of an output sequence and the output of the pulse.

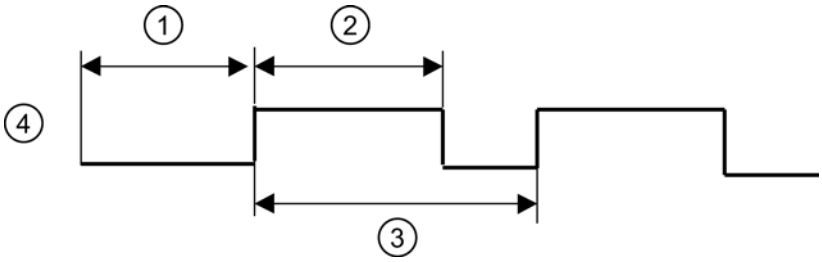
Enter a value as in the resolution configured above.

If you select the "1 ms" option in the "Time base" drop-down list, you can set times from 0 ms to 65535 ms with a resolution of 1 ms.

If you select the "0.1 ms" option in the "Time base" drop-down list, you can set times from 0 ms to 6553.5 ms with a resolution of 0.1 ms.

Default is "0".

This pulse sequence is output on digital output DO (output sequence) after the on-delay has expired.



① On delay
② Pulse Period
③ Period
④ Output DO

Default setting is "0".

Period

The pulse / pause period defines the length of the output sequence.

In the "Period" input field, enter a value as in the resolution configuration above.

If you select the "1 ms" option in the "Time base" drop-down list, you can set times from 1 ms to 65535 ms at a resolution of 1 ms.

If you select the "0.1 ms" option in the "Time base" drop-down list, you can set times from 0.4 ms to 6553.5 ms with a resolution of 0.1 ms.

Default setting is "20000".

Minimum pulse time

You can set a minimum pulse time for the attenuation of short output pulses/pauses. This suppresses all pulses or pauses shorter than the minimum pulse time.

In the "Minimum pulse time" input field, enter a value as in the resolution configuration above.

If you select the "1 ms" option in the "Time base" drop-down list, enter period values from 0 to /2.

If you select the "0.1 ms" option in the "Time base" drop-down list, enter period values from 2 to /2.

The minimum pulse width is internally set to 0.2 ms if you have selected 1 ms for the time base and a minimum pulse time of 0.

Default setting is "2".

Input 0 (pulse-width modulation)

HW gate

The "HW gate" check box allows you to select whether or not a hardware gate is to be used.

This check box is not selected by default.

Start of pulse-width modulation with a hardware gate is only possible if you first open the software gate and then generate a positive edge on digital input "hardware gate".

Pulse width modulation can only be stopped with a negative edge at the software gate. The hardware gate status has no effect.

Filter frequency

HW gate

This drop-down list is only available if the "HW gate" option is selected under "Input". Here you can set the maximum frequency of the hardware gate signal in fixed intervals. The maximum value depends on the CPU in use:

CPU	Range of values	Default selection
CPU 312C	10, 5, 2, 1 kHz	10 kHz
CPU 313C CPU 313C-2 DP CPU 313C-2 PtP	30, 10, 5, 2, 1 kHz	30 kHz
CPU 314C-2 DP CPU 314C-2 PN/DP CPU 314C-2 PtP	60, 30, 10, 5, 2, 1 kHz	60 kHz

Hardware interrupt at (pulse-width modulation)

Description

The corresponding event triggers a hardware interrupt when you select the respective check box. The following events can trigger a hardware interrupt:

- HW gate opening (with opened SW gate)

You must enable this in the Basic Parameters tab to ensure that the CPU generates a hardware interrupt.

No option is selected by default.

Continuous Counting

Description

The "Count continuously" operating mode is defined as follows:

After the counter reaches its upper limit it jumps to the lowest counting limit at the next count pulse and resumes the count from there.

After the counter reaches its lower limit it jumps to the upper counting limit at the next count pulse and resumes the count from there.

Those counting limits are set fixed to the maximum counting range.

In this operating mode, counting is started and stopped via the gate function.

Count once

Description

When the counter reaches the end value 1, in the up-counting main counting direction mode, the counter jumps to the counter range start at the next pulse and stops, even if there are additional count pulses coming in.

When the counter reaches the end value 1, in the down-counting main counting direction mode, the counter jumps to the counter range start at the next pulse and stops, even if there are additional count pulses coming in.

The counting limits are set to the maximum counting range if there is no master counting direction configuration.

In this operating mode, counting is started and stopped via the gate function.

When the gate opens, the counter can receive counting impulses and the count starts. When the gate is closed, the counter can no longer receive count pulses and the count stops.

Count periodically

Description

If the main count direction of the counter is up and the counter has reached the end value - 1, the arrival of an additional count pulse causes the counter to jump to the load value from where it continues counting.

If the main count direction of the counter is down and the counter has reached the count value = 1, the arrival of an additional count pulse causes the counter to jump to the start value (load value) from where it continues counting.

The count limits are set to the maximum counting range if a main count direction is not configured.

In this operating mode, counting is started and stopped via the gate function.

When the gate opens, the counter can receive count pulses and the count starts. When the gate is closed, the counter can no longer receive count pulses and the count stops.

Main counting direction

Description

In your parameters you can declare "up" or "down". This determines the counting limit for the start value or end value for the operating modes Single Counting and Periodic Counting.

For main counting direction "up", the counter starts at "0" or at the load value and counts up to the limit value you have declared in your parameters.

For the main count direction "down", the counter starts at the start value you have declared in your parameters and then counts down to "0".

Even though you may have declared "down" counts in your parameters for the main counting direction, you must still set a respective direction signal for reverse counting or you must declare "Invert direction of count" in your parameters.

Gate Function

Software gate and Hardware gate

The CPU has two logical AND linked gates:

A **software gate** (SW gate), controlled via the user program.

The software gate can only be opened on a 0-1 edge transition of the input parameters SW_GATE on the SFB. A reset of this parameter closes the gate.

A **hardware gate** (HW gate), controlled via the digital input "Hardware gate". In the "Count" tab you can specify whether or not you want to use the hardware gate. This gate opens at a 0-1 transition and closes at a 1-0 transition on the digital input.

Positioning function

Interrupt selection

You can select interrupts for the "Positioning" function.

Requirement

- The CPU is selected.
- Interrupt selection is located at "Properties > General > Positioning > Interrupt selection".

Interrupt selection

In the "Interrupt Selection" drop-down list, select whether or not the CPU should trigger diagnostic interrupts. You have the following options:

- None
- Diagnostics

Default selection is "None".

You must carry out supplementary configurations to ensure that the corresponding events are really going to trigger diagnostic interrupts.

Enable the required monitoring under "Properties > Positioning > Channel 0" with operating mode "Positioning with analog output" selected:

- Under "Drive"
  Actual value, target approach and target range can be monitored.
- Under "Axis"
  For a linear axis, you can specify here whether or not to monitor the working range. The traversing range is always monitored.
- Under "Encoder"
  For encoders with zero mark, it is verified whether the encoder always supplies the same number of pulses between two consecutive zero marks.

Select which events are to trigger a diagnostic interrupt under "Channel 0 - Diagnostics".

For a diagnostic interrupt, you must configure the parameters in the following settings:

- Interrupt selection under "Properties > General > Positioning > Channel 0"
- The corresponding monitoring in the appropriate "Drive", "Axis" or "Encoder" section.
- The respective enable signal in the "Diagnostics" section.

Operating mode

Operating mode

Use the "Operating mode" parameter to determine how you want to position. You have the following options:

- Analog outputs: Positioning with analog output
- Digital outputs: Positioning with digital outputs

Default selection is "not configured".

The operating mode is selected under "Properties > General > Positioning > Channel 0".

Note

You cannot configure the "Positioning" circuit if you have configured channel 0 or 1 for the counting circuit.

You can only assign an analog output to positioning if you have disabled output 0 in section AI5/AO2.

I/O channels configured for positioning can no longer be addressed by the user program.

See also

Documentation on technological functions of the CPU (http://support.automation.siemens.com/WW/view/de/12429336/0/en)

Positioning with analog output

Drive

Controlling the drive and control mode

Controlling the drive and control mode

In "Properties > Position > Channel 0" in the "Channel 0 - Drive" section, specify the parameters for the drive and axis for positioning using an analog output.

Drive description

The drive is controlled via two fixed assigned analog outputs, with either a voltage of ±10 V (Pin 16) or a current of ±20 mA (Pin 17).

Note

You can only assign an analog output to positioning if you have disabled output 0 in section AI5/AO2 of the area navigation.

Description of control mode

In the drop-down list you have the option of 2 steer modes. The control mode describes how your connected power unit is controlled.

- Voltage ±10 V or current ±20 mA:

A positive voltage or current is generated when traveling in the plus direction (forward). A negative voltage or current is generated when traveling in the minus direction (back).

- Voltage 0 to 10 V or current from 0 to 20 mA and direction signal:

A positive voltage or current is generated when traveling in the plus direction (forwards) and the CONV_DIR digital output is switched off.

A positive voltage or current is generated when traveling in the minus direction (back) and the CONV_DIR digital output is switched on.

The default setting is "±10".

Max. frequency

Position feedback

In this drop-down list you can set the maximum counting frequency for the position feedback signals (encoder signals A, B, N) in fixed levels.

If your CPU does not support this function, the drop-down list in unavailable and grayed out.

Default setting here is "60 kHz".

Accompanying signals

In this drop-down list you can set the maximum counting frequency for the length measurement and reference point switch signals in fixed levels.

If your CPU does not support this function, the drop-down list in unavailable and grayed out.

Default setting here is "10 kHz".

Target approach (analog)

Target range

In the "Target range" input field, enter an even-numbered value between 0 and 2 x 10 pulses. An odd value is rounded off internally. The target range lies symmetrically around the target. Here, if you enter the value 0, the target must be reached with the accuracy of a pulse. The target range monitoring is switched off and the relevant check box in the "Monitoring" area is unavailable.

Default is "50" pulses.

Monitoring time

In the "Monitoring time" input field, enter a time between 0 and 100000 ms. The CPU rounds this value to a 4 ms execution cycle.

The CPU uses the monitoring time to monitor the actual value and the target approach.

If you enter the value 0, the actual value monitoring and target approach monitoring are switched off and the relevant check boxes in the "Monitoring" area are then unavailable.

Default selection is "2000 ms".

Maximum speed

In the "Maximum speed" input field, enter a value between 10 and 1 000 000 pulses/s. This corresponds to a level of 10 V or 20 mA at the analog output.

Default is "1000 pulses/s".

Creep speed / reference speed

In the "Creep-/referencing speed" input field, enter a value between 10 pulses/s and the assigned maximum speed.

The drive is decelerated to creep speed after it reaches the braking point.

In the reference point approach mode, the drive is decelerated to reference speed when it reaches the reference point switch.

Default is "100" pulses/s.

Enable power section

Off delay

In the "Off delay" input field, enter a time in milliseconds. The CPU rounds this value to a 4 ms execution cycle. Output CONV_EN is reset after a delay of this amount of time when a run is canceled (power section enable is removed).

If you actuate a brake via the digital output, the delay time ensures that the axis, at the time the output is reset, is slow enough for the brake to handle the kinetic energy.

Default is "1000 ms".

Monitoring (analog)

Description

Here you can switch process value, target approach and target range monitoring on and off.

Target range

Once you have selected the "Target range" button to monitor whether the drive remains in the approached target position or drifts off after it has reached this target position.

You cannot switch on target range monitoring if you have customized a 0 ms monitoring time for "Target approach".

An external error message is returned when the monitoring unit is triggered. You must acknowledge this external error with ERR_A. Monitoring is switched off. Monitoring is not switched on until a new run starts.

This monitoring function is disabled by default.

Target approach

Once you have selected the "Target approach" button, it is monitored whether the axis has approached the target range within the monitoring time after it has reached the switch-off point.

You cannot switch on process value monitoring if you have customized a 0 ms monitoring time for "Target approach".

This monitoring function is disabled by default.

Process value

Axis movement is monitored for the run start to the switch-off point if you have selected the "Process value" button. The run is interrupted if the axis does not move by at least one pulse into the specified direction within the monitoring time.

You cannot switch on process value monitoring if you have customized a 0 ms monitoring time for "Target approach".

The CPU does not recognize the failure of a digital input. Process value monitoring lets you detect missing encoder signals. This error can have the following causes:

- Digital input failure
- Wire break
- Defective encoder
- Faulty power section

This monitoring feature is switched on by default.

Axis

Axis type (analog)

Description

Specify the parameters for the axis under "Properties > Positioning > Channel 0" in the "Channel 0 - Axis " section.

Axis type

In "Axis type" select whether to control a linear axis or a rotary axis.

Default selection is "Linear axis".

Start and end of software limit switch

The working area of a linear axis is between the start of the software limit switch and the end of the software limit switch. The software limit switches belong to the working area. Enter values between -5 x 10^8 and +5 x 10^8 pulses in the relevant boxes. The start of the software limit switch must always be less than the end of the software limit switch.

The software limit switches are monitored, provided the axis is synchronized and working range monitoring is enabled.

Default is "-100000000 pulses" for the start of the software limit switch and "100000000 pulses" for the end of the software limit switch.

End of rotary axis

In the "End of rotary axis" box, enter a value between 1 and 10^9 pulses.

The "End of rotary axis" value is the theoretically largest value that the actual value can reach. Physically, it has the same position as the start of the rotary axis (0).

Default is "100000" pulses.

Length measurement (analog)

Description

The "Length measurement" function allows you to measure a distance during a run operation. The start and end of the length measurement are edge-triggered via the digital input "Length measurement". Select the type of edge in the "Detection" drop-down list.

You have the following options:

- Off
- Start/end on rising edge
  A rising edge on the digital input starts the length measurement, the next rising edge ends it.
- Start/end on falling edge
  A falling edge on the digital input starts the length measurement, the next falling edge ends it.
- Start on rising edge, end on falling edge
  A rising edge on the digital input starts the length measurement, the next falling edge ends it.
- Start with falling edge, end with rising edge
  A falling edge on the digital input starts the length measurement, the next rising edge ends it.

Default is "off".

Reference point (analog)

Coordinate

After the CPU performs a STOP-RUN transition, the axis is set to the value of the reference point coordinate while the axis has not yet synchronized.

After a reference point run the value of the reference point coordinate is assigned to the reference point.

For a linear axis, the value of the reference point coordinate must lie within the working range (including the start / end of the software limit switches).

For a rotary axis, the value of the reference point coordinate must lie within the range from 0 to "End of rotary axis - 1".

Default is "0" pulses.

Reference Point Location for Reference Point Switch

In the "Reference point location for reference point switch" drop-down list, select whether the reference point lies in the plus direction (increasing actual values) or minus direction (decreasing actual values) relative to the reference point switch.

Default is "in plus direction".

Axis monitoring (analog)

Working range

For a linear axis, you can specify here whether or not to monitor the working range. If you select the "Working range" button, a check is made to determine whether the actual position value is out of range of the software limit switches. Monitoring is only active on a synchronized axis.

The coordinates of the software limit switches as such belong to the working range.

If monitoring is triggered the run is aborted.

This monitoring function is enabled by default.

Traversing Range

You use traversing range monitoring to check whether the value is outside the permitted traversing range of - 5 x 10^8 to + 5 x 10^8. The traversing range is always monitored. The monitoring cannot be disabled (always enabled in the "Monitoring" parameter).

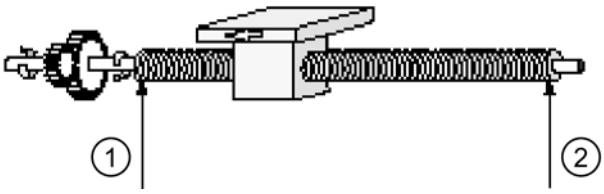
If monitoring is triggered the run is aborted.

Linear axis

Description

The **linear axis** is an axis whose physical traversing range is restricted.

The working area on the linear axis is limited by the software limit switches.



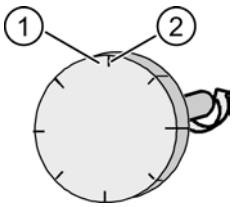
① Physical start

② Physical end

Rotary axis

Description

The **rotary axis** is an axis the traversing range of which is not restricted by mechanical stops.



① Largest display value = end of rotary axis - 1

② Start of rotary axis (coordinate 0) = end of rotary axis

A rotary movement of the rotary axis starts at the coordinate "Zero" and ends at the coordinate "End of rotary axis". Physically, the "zero" coordinate is identical with "End of rotary axis" (= 0). The actual position value is toggled at this point. The display of the actual position is always positive.

Encoder

Increments per encoder revolution

In the "Increments per encoder revolution" input field, enter the number of increments your encoder should output per 360 degrees. You can find this value in the encoder description.

The CPU evaluates the increments 4 times, in other words, an increment corresponds to four pulses.

Default setting is "1000" increments.

Count direction

In "Count direction", adapt the direction of the position feedback to the axis motion direction.

In the "count direction" drop-down list, select the "Normal" option if rising count pulses correspond to increasing actual position values.

In the "count direction" drop-down list, select the "Inverted" option if rising count pulses correspond to decreasing actual position values.

You should consider all the directions of rotation of the transmission elements, such as clutches and gears.

The "Normal" option is selected by default.

Monitoring

In "Monitoring" you can determine whether the encoder is to be monitored for missing pulses. This is only possible for encoders where the value of "Increments per encoder revolution" is divisible by 10 or 16.

If you select the "Missing pulse" check box, a check is performed to determine whether the encoder always supplies the same number of pulses between two consecutive zero marks.

If you use an encoder without zero mark, you must disable the monitoring for missing pulses.

This monitoring function is disabled by default.

Diagnostics

Description

In "Properties > Position > Channel 0" in section "Channel 0 - Diagnostics", select the events for which the diagnostics interrupt is to be generated. No check box is selected by default.

Missing pulse

If you select the "Missing pulse" check box, a diagnostic interrupt is generated as soon as the encoder does not supply the same number of pulses between two consecutive zero marks. You must enable the corresponding monitoring function in the "Channel 0 - Encoder" section.

Traversing range

If you select the "Traversing range" check box, a diagnostic interrupt is generated immediately when the axis exits the traversing range. The corresponding monitoring function is always enabled.

Working range

If you select the "Working range" check box, a diagnostic interrupt is generated if the actual position value is out of range of the software limit switches. You must enable the corresponding monitoring function in the "Channel 0 - Axis" section.

You cannot use this diagnostic interrupt for positioning on a rotary axis. The respective options box is grayed out.

Actual value

If you selected the "Actual value" check box, a diagnostic interrupt is generated if during a traversing motion, the axis does not move by at least one pulse in the specified direction within the monitoring time. The run is cancelled.

The actual value is monitored from the start of travel until the switch-off point is reached.

You must enable the corresponding monitoring function in the "Channel 0 - Drive" section.

The CPU does not recognize the failure of a digital input. Enabling the actual value monitoring allows you to detect missing encoder signals.

This check box is selected by default.

Target approach

If you select the "Target approach" check box, a diagnostic interrupt is generated if the axis does not reach the target range within the monitoring time after it has reached the switch-off point.

You must enable the corresponding monitoring function in the "Channel 0 - Drive" section.

Target range

If you selected the "Target range" check box, a diagnostic interrupt is generated if the drive does not stop on an approached target position after the target range is reached or if the drive drifts away from the approached target position.

You must enable the corresponding monitoring function in the "Channel 0 - Drive" section.

Positioning with digital outputs

Drive

Controlling the drive

In "Properties > Position > Channel 0" in the "Channel 0 - Drive" section, specify the parameters for the drive and axis for positioning with digital inputs.

Description

The drive is controlled via four permanently allocated 24 V digital outputs.

In the drop-down list you have the option of 4 steer modes. The control mode describes how the 4 digital outputs (DO+1.0 to DO+1.3) control the power section.

Below lists the individual output functions under the various control modes:

Output	Control mode 1	Control mode 2	Control mode 3	Control mode 4
DO + 1.0	Rapid traverse	Rapid / creep traverse	Rapid traverse	Rapid traverse plus
DO + 1.1	Creep traverse	Position reached	Creep traverse	Creep traverse plus
DO + 1.2	Drive plus	Drive plus	Drive plus	Rapid traverse minus
DO + 1.3	Drive minus	Drive minus	Drive minus	Creep traverse minus

Max. frequency

Position feedback

In this drop-down list you can set the maximum counting frequency for the position feedback signals (encoder signals A, B, N) in fixed levels.

If your CPU does not support this function, the drop-down list in unavailable and grayed out.

Default setting here is "60 kHz".

Accompanying signals

In this drop-down list you can set the maximum counting frequency for the length measurement and reference point switch signals in fixed levels.

If your CPU does not support this function, the drop-down list in unavailable and grayed out.

Default setting here is "10 kHz".

Target approach (digital)

Target range

Enter an even-numbered value between 0 and 2×10^{8} pulses in the "Target range" input field. An odd value is rounded off internally. The target range lies symmetrically around the target. Here, if you enter the value 0, the target must be reached with the accuracy of a pulse. In this case, the target range monitoring is switched off and the relevant check box in the "Monitoring" area is unavailable.

Default is "50" pulses.

Monitoring time

Enter a time between 0 and 100000 ms in the "Monitoring time" input field. The CPU rounds this value to a 4 ms execution cycle.

The CPU uses the monitoring time to monitor the actual value and the target approach.

If you enter the value 0, the actual value monitoring and target approach monitoring are switched off and the relevant check boxes in the "Monitoring" area are then unavailable.

The default setting is "2000 ms".

Monitoring (digital)

Description

Here you can switch actual value, target approach and target range monitoring on and off.

Target approach

If you select the "Target approach" check box, the axis is monitored to determine whether it reaches the target range within the monitoring time after the switch-off point is reached.

You cannot switch on actual value and target approach monitoring if you have set a 0 ms monitoring time for "Target approach". The respective fields are grayed out.

This monitoring function is disabled by default.

Target range

If you select "Target range" check box, the drive is monitored to determine whether it stops on an approached target position after the target range is reached or if the drive drifts away from the target range.

An external error is returned when the monitoring is triggered. You must acknowledge this external error with ERR_A. Monitoring is switched off. Monitoring is not switched on again until a new run starts.

This monitoring function is disabled by default.

Actual value

If you select the "Axis value" check box, the axis movement is monitored from the start of the run up until the switch-off point is reached. The run is canceled if the axis does not move by at least one pulse into the specified direction within the monitoring time.

You cannot switch on actual value and target approach monitoring if you have set a 0 ms monitoring time for "Target approach". The respective fields are grayed out.

The CPU does not recognize the failure of a digital input. Actual value monitoring lets you detect missing encoder signals. This error can have the following causes:

- Digital input failure
- Wire break
- Defective encoder
- Faulty power section

This monitoring function is enabled by default.

Axis

Axis type (digital)

Description

In "Axis type" select whether to control a linear axis or a rotary axis.

Default selection is "Linear axis".

Start and end of software limit switch

The working area of a linear axis is between the start of the software limit switch and the end of the software limit switch. The software limit switches belong to the working area. Enter values between -5 x 10^8 and +5 x 10^8 pulses in the relevant boxes. The start of the software limit switch must always be less than the end of the software limit switch.

The software limit switches are monitored, provided the axis is synchronized and working range monitoring is enabled.

Default is "-100000000 pulses" for the start of the software limit switch and "100000000 pulses" for the end of the software limit switch.

End of rotary axis

In the "End of rotary axis" box, enter a value between 1 and 10^9 pulses.

The "End of rotary axis" value is the theoretically largest value that the actual value can reach. Physically, it has the same position as the start of the rotary axis (0).

Default is "100000" pulses.

Length measurement (digital)

Description

The "Length measurement" function allows you to measure a distance during a run operation. The start and end of the length measurement are edge-triggered via the digital input "Length measurement". Select the type of edge in the "Detection" drop-down list.

You have the following options:

- Off
- Start/end on rising edge
  A rising edge on the digital input starts the length measurement, the next rising edge ends it.
- Start/end on falling edge
  A falling edge on the digital input starts the length measurement, the next falling edge ends it.
- Start on rising edge, end on falling edge
  A rising edge on the digital input starts the length measurement, the next falling edge ends it.
- Start with falling edge, end with rising edge
  A falling edge on the digital input starts the length measurement, the next rising edge ends it.

Default is "off".

Reference point (digital)

Coordinate

After the CPU performs a STOP-RUN transition, the axis is set to the value of the reference point coordinate while the axis has not yet synchronized.

After a reference point run the value of the reference point coordinate is assigned to the reference point.

For a linear axis, the value of the reference point coordinate must lie within the working range (including the start / end of the software limit switches).

For a rotary axis, the value of the reference point coordinate must lie within the range from 0 to "End of rotary axis - 1".

Default is "0" pulses.

Reference Point Location for Reference Point Switch

In the "Reference point location for reference point switch" drop-down list, select whether the reference point lies in the plus direction (increasing actual values) or minus direction (decreasing actual values) relative to the reference point switch.

Default is "in plus direction".

Axis monitoring (digital)

Working range

For a linear axis, you can specify here whether or not to monitor the working range. If you select the "Working range" button, a check is made to determine whether the actual position value is out of range of the software limit switches. Monitoring is only active on a synchronized axis.

The coordinates of the software limit switches as such belong to the working range.

If monitoring is triggered the run is aborted.

This monitoring function is enabled by default.

Traversing range

You use traversing range monitoring to check whether the value is outside the permitted traversing range of $- 5 \times 10^8$ to $+ 5 \times 10^8$. The traversing range is always monitored. The monitoring cannot be disabled (always enabled in the "Monitoring" parameter).

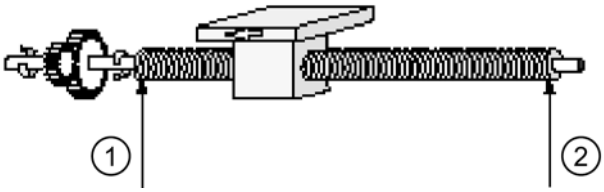
If monitoring is triggered the run is aborted.

Linear axis

Description

The **linear axis** is an axis whose physical traversing range is restricted.

The working area on the linear axis is limited by the software limit switches.

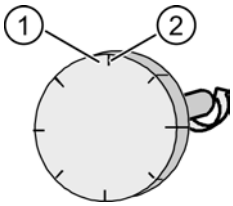


① Physical start
② Physical end

Rotary axis

Description

The **rotary axis** is an axis the traversing range of which is not restricted by mechanical stops.



① Largest display value = end of rotary axis - 1
② Start of rotary axis (coordinate 0) = end of rotary axis

A rotary movement of the rotary axis starts at the coordinate "Zero" and ends at the coordinate "End of rotary axis". Physically, the "zero" coordinate is identical with "End of rotary axis" (= 0). The actual position value is toggled at this point. The display of the actual position is always positive.

Encoder

Increments per encoder revolution

In the "Increments per encoder revolution" input field, enter the number of increments your encoder should output per 360 degrees. You can find this value in the encoder description.

The CPU evaluates the increments 4 times, in other words, an increment corresponds to four pulses.

Default setting is "1000" increments.

Count direction

In "Count direction", adapt the direction of the position feedback to the axis motion direction.

In the "count direction" drop-down list, select the "Normal" option if rising count pulses correspond to increasing actual position values.

In the "count direction" drop-down list, select the "Inverted" option if rising count pulses correspond to decreasing actual position values.

You should consider all the directions of rotation of the transmission elements, such as clutches and gears.

The "Normal" option is selected by default.

Monitoring

In "Monitoring" you can determine whether the encoder is to be monitored for missing pulses. This is only possible for encoders where the value of "Increments per encoder revolution" is divisible by 10 or 16.

If you select the "Missing pulse" check box, a check is performed to determine whether the encoder always supplies the same number of pulses between two consecutive zero marks.

If you use an encoder without zero mark, you must disable the monitoring for missing pulses.

This monitoring function is disabled by default.

Diagnostics

Description

In "Properties > Position > Channel 0" in section "Channel 0 - Diagnostics", select the events for which the diagnostics interrupt is to be generated. No check box is selected by default.

Missing pulse

If you select the "Missing pulse" check box, a diagnostic interrupt is generated as soon as the encoder does not supply the same number of pulses between two consecutive zero marks. You must enable the corresponding monitoring function in the "Channel 0 - Encoder" section.

Traversing range

If you select the "Traversing range" check box, a diagnostic interrupt is generated immediately when the axis exits the traversing range. The corresponding monitoring function is always enabled.

Working range

If you select the "Working range" check box, a diagnostic interrupt is generated if the actual position value is out of range of the software limit switches. You must enable the corresponding monitoring function in the "Channel 0 - Axis" section.

You cannot use this diagnostic interrupt for positioning on a rotary axis. The respective options box is grayed out.

Actual value

If you selected the "Actual value" check box, a diagnostic interrupt is generated if during a traversing motion, the axis does not move by at least one pulse in the specified direction within the monitoring time. The run is cancelled.

The actual value is monitored from the start of travel until the switch-off point is reached.

You must enable the corresponding monitoring function in the "Channel 0 - Drive" section.

The CPU does not recognize the failure of a digital input. Enabling the actual value monitoring allows you to detect missing encoder signals.

This check box is selected by default.

Target approach

If you select the "Target approach" check box, a diagnostic interrupt is generated if the axis does not reach the target range within the monitoring time after it has reached the switch-off point.

You must enable the corresponding monitoring function in the "Channel 0 - Drive" section.

Target range

If you selected the "Target range" check box, a diagnostic interrupt is generated if the drive does not stop on an approached target position after the target range is reached or if the drive drifts away from the approached target position.

You must enable the corresponding monitoring function in the "Channel 0 - Drive" section.

Point-to-point connection function

ASCII protocol

PtP transmission (ASCII)

Description

The interrupts, characteristics of a CPU STOP, and protocol are set in "Properties > General > PtP > Interrupt selection". Select "ASCII" as the protocol.

Depending on the type of protocol selected, the further subdivision changes in the area navigation.

For "ASCII" there are 4 parameter groups: transmission, end delimiter, data reception and signal assignment.

This section deals with PtP transmission and is divided into 3 sub-sections.

Transmission speed

Select the data transmission rate in bps (baud) from the "Transmission speed" drop-down list. You can select a value between 300 and 38400 baud. In full duplex mode, the maximum transmission speed is 19200 baud.

The default setting is "9600 baud".

Character frame

The data are transmitted via the serial interface in a character frame. Two data formats are available for each character frame.

- Data bits
  In the "Data bits" drop-down list, select the number of bits onto which a character is to be mapped.
  The default setting is "8 bits".
- Stop bits
  In the "Stop bits" drop-down list, select the number of bits appended to every transmitted character. The stop bit identifies the end of a character.
  The default setting is "1 bit".
- Parity
  In the "Parity" drop-down list, select whether the data transmission reliability is to be increased by using a parity bit. A parity of "None" means that no parity bit is transmitted.
  The setting "None" is not possible in a 7-data-bit configuration.
  A sequence of information bits can be extended to include another bit, the parity bit. The addition of its value ("0" or "1") brings the value of all the bits up to a defined state – either even or odd.
  Default selection is "Even".

Data flow control transmission

- Data flow control
  Here you can only specify parameters if you have selected "XON/XOFF" data flow control.
  Default selection is "None".
- XON character
  Enter a code in the "XON character" input field for the XON character. Enter a hex format code here.
  Default selection is "11".

- XOFF character
  Enter a code in the "XOFF character" input field for the XOFF character. Enter a hex format code here.
  Which values you can enter depends on whether you transmit your data with 7 or 8 data bits:
  - 7 data bits: 0 to 7FH
  - 8 data bits: 0 to FFH
    Default selection is "13".
- Wait for XON after XOFF
  In the "Wait for XON after XOFF" input field, specify a time the module has to wait after sending until it receives the XON character from the communication partner.
  You can enter a value in steps of 10 ms, from 20 ms to 65530 ms.
  Default selection is "20000 ms".

PtP end delimiter (ASCII)

Description

The interrupts, characteristics of a CPU STOP, and protocol are set in "Properties > General > PtP > Interrupt selection". Select "ASCII" as the protocol.

Depending on the type of protocol selected, the further subdivision changes in the area navigation.

For "ASCII" there are 4 PtP areas: transmission, end delimiter, data reception and signal assignment.

This section deals with PtP end delimiter and is divided into 2 sub-sections.

Detection of received frame

Select the required option from the "Detection of end of received frame" drop-down list. Three options are available: "After character delay time elapses", "Using end delimiter" and "After receiving a fixed number of characters".

- After character delay time elapses
  Select "After character delay time elapses" from the drop-down list to specify a configured time interval after which the end of a frame is recognized. The character delay time defines the permissible maximum interval between 2 consecutive received characters.
  This option is selected by default.
- Character delay time
  In the "Character delay time" input field you must enter this time interval in milliseconds between 1 and 65,535.
  The shortest character delay that you can select depends on the transmission speed.

Transmission speed	Shortest character delay time
300 bps	130 ms
600 bps	65 ms
1200 bps	32 ms

Transmission speed	Shortest character delay time
2400 bps	16 ms
4800 bps	8 ms
9600 bps	4 ms
19200 bps	2 ms
38400 bps	1 ms

The default setting is "4 ms".

- Using end delimiter
  Select "Using end delimiter" from the drop-down list to specify that the end of a message frame is recognized by its end characters.
- Monitoring time for missing end delimiter
  In the "Monitoring time for missing end delimiter" input field you can set the character delay time that can be used as monitoring time for missing end delimiter. This applies to the end delimiter for the following settings:
  - after receiving a fixed number of characters
  - after receiving the end delimiter(s)
    You can also modify the monitoring time here. The same marginal conditions as for the character delay time are applied.
    The calculation when sending and evaluating the block check character must be carried out in the user program itself.
    The default selection is "4 ms".
- End delimiter
  Select the "Using end delimiter" option of the "Detection of end of received frame" parameter to specify the use of one or two defined end delimiters for designating the end of a frame. You can work with one or two end delimiters. Optionally, one or two additional characters can be received after the end delimiter detection. You can use these characters, for example, to include a block check character (BCC) in your transmission.
  Select one of the following options in the corresponding drop-down list:
  - 1st character
  - 1st character with 1 BCC
  - 1st character with 2 BCC
  - 1st AND 2nd character
  - 1st AND 2nd character with 1 BCC
  - 1st AND 2nd character with 2 BCC
    If you have selected this option, the default setting is "1st character".
- Receiver end delimiter
  In the "Receiver 1st end delimiter" input field, and, if required, in the "Receiver 2nd end delimiter" input field, enter a code for the respective end delimiter.
  Which values you can enter depends on whether you transmit your data with 7 or 8 data bits:
  - 7 data bits: 0 to 7FH
  - 8 data bits: 0 to FFH
    The default setting is "3" for the first end delimiter and "0" for the second end delimiter.

- After receiving a fixed number of characters
  Select "After receiving a fixed number of characters" from the drop-down list to specify a fixed number of characters after which the end of a frame is recognized. The length of the received frames is always identical.
  This option is selected by default.
- Pause in transmission between frames (same length as monitoring time)
  If you have selected the "Pause in transmission between frames (same length as monitoring time)" option, a pause equal to the length of the monitoring time (for missing end delimiter) is observed so that the partner detects the receipt of the frame and can synchronize itself accordingly.
  This check box is not selected by default.
- Frame length
  In the "Frame length" input field, you can customize the number of bytes a frame should consist of. You can set a value between 1 and 1024 bytes.
  The default setting is "200 bytes".

Send with end delimiter

You can transmit with an end delimiter after you have specified an end delimiter as the end delimiter of a received frame.

Default selection is "Send up to and including end delimiter".

- Send up to and including end delimiter
  Select the "End criteria > Send up to and including end delimiter" drop-down list to specify whether the end delimiter should be included in the data to be transmitted. Only data up to and including the end-of-message character is transmitted, even if a greater data length is specified in the SFB.
- Send up to length set in the block
  Select the "End criteria > Send up to length set in the block" drop-down list to determine that data are transmitted up to the length declared in the SFB parameters. The last character must be the end delimiter.
- Send with automatically append the end delimiter
  Select the "End criteria > Automatically append the end delimiter" drop-down list to determine that data are transmitted up to the length declared in the SFB parameters. The end delimiter(s) are automatically appended, that is, the end delimiters must not be contained in the data to be transmitted. Depending on the number of end identifiers, either one or two characters more than specified in the SFB (max. 1024 bytes) are transmitted to the partner.
- Sender end delimiter
  If you are using an end delimiter you can now specify its hex code. You can also select whether you want to use a second end delimiter and specify its hex code also. In the "Sender 1st end delimiter" input field, and, if required, in the "Sender 2nd end delimiter" input field, enter a code for the respective end delimiter.
  The default setting is "3" for the first end delimiter and "0" for the second end delimiter.

PtP - data reception (ASCII)

Description

The interrupts, characteristics of a CPU STOP, and protocol are set in "Properties > General > PtP > Interrupt selection". Select "ASCII" as the protocol.

Depending on the type of protocol selected, the further subdivision changes in the area navigation.

For "ASCII" there are 4 PtP areas: transmission, end delimiter, data reception and signal assignment.

This section deals with PtP data reception.

Reception buffering on the CPU

- Clear CPU receive buffer during startup
  Select the "Clear CPU receive buffer during startup" check box to clear the receive buffer of the technological function after power up and after the CPU transition from STOP to RUN.
  This check box is not selected by default.
- Prevent overwriting
  Select the "Prevent overwriting" check box to prevent overwriting data in the receive buffer when the buffer is full.
  This check box is selected by default.
- Use entire buffer
  Select the "Use entire buffer" check box to utilize the complete input buffer of 2048 bytes. When you use the complete buffer, the number of received frames is only dependent on the frame length.
  This check box is selected by default.
- Maximum number of buffered received frames
  In the "Maximum number of buffered received frames" input field, you can enter the number of received frames you want to have buffered in the receive buffer of the technological function. To do this, deactivate the "Use entire buffer" check box. You can select a value between 1 and 10.
  Default setting is "10".

Note

If you specify "1" here and deactivate the "Prevent overwrite" parameter **and** cyclically read the received data from the user program, a current message frame will always be sent to the CPU.

PtP - signal assignment (ASCII)

Description

The interrupts, characteristics of a CPU STOP, and protocol are set in "Properties > General > PtP > Interrupt selection". Select "ASCII" as the protocol.

Depending on the type of protocol selected, the further subdivision changes in the area navigation.

For "ASCII" there are 4 PtP areas: transmission, end delimiter, data reception and signal assignment.

This section deals with PtP signal assignment and is divided into 2 sub-sections.

Operating mode

Select one of the three operating modes.

Default is "Full duplex (RS 422) four-wire point-to-point mode".

Full duplex (RS 422) four-wire mode point-to-point

Check the "Full duplex (RS422) four-wire PtP mode" option button to select the suitable operating mode for a PtP connection in four-wire mode. Here, you have the following options to preset the receive line:

- None
- Signal R(A) 5 volts, Signal R(B) 0 volts
- Signal R(A) 0 volts, Signal R(B) 5 volts

Full duplex (RS 422) four-wire multipoint master mode

If the CPU is the master, check the "Full duplex (RS422) four-wire multipoint master mode" option button to select a suitable operating mode for a four-wire connection that is capable of multipoint master operation. Here, you have the following options to preset the receive line:

- Signal R(A) 0 volts, Signal R(B) 5 volts

Half duplex (RS 485) two-wire mode

Check the "Half duplex (RS485) two-wire mode" option button to select a suitable operating mode for a PtP or multipoint connection in two-wire mode. The CPU can be master or slave. Here, you have the following options to preset the receive line:

- None
- Signal R(A) 0 volts, Signal R(B) 5 volts

Receive line initial state

- None
  Check the "None" option button to select a configuration that is significant for bus-compatible special drives only.
- Signal R(A) 5 volts, Signal R(B) 0 volts
  Select the "Signal R(A) 5 volts, Signal R(B) 0 volts" option button to select a configuration for break recognition. "Not available for full duplex (RS422) four-wire multipoint master and half duplex (RS485) two-wire operation)
- Signal R(A) 0 volts, Signal R(B) 5 volts
- Select the "Signal R(A) 0 volts, Signal R(B) 5 volts" option button to select a configuration that represents the idle state (no sending active). Break recognition is not possible with this configuration.
- Default is "Signal R(A) 5 volts, Signal R(B) 0 volts".

3964(R) protocol

PtP - transmission (3964(R))

Description

The interrupts, characteristics of a CPU STOP, and protocol are set in "Properties > General > PtP > Interrupt selection". Select "3964(R)" as the protocol.

Depending on the type of protocol selected, the further subdivision changes in the area navigation.

For "3964(R)" there are 3 PtP areas: transmission, data reception and signal assignment.

This section deals with PtP transmission and is divided into 3 sub-sections.

Transmission speed

Select the data transmission rate in bps (baud) from the "Transmission speed" drop-down list. You can select a value between 300 and 38400 baud.

Default selection is "9600 baud".

Character frame

The data are transmitted via the serial interface in a character frame. Two data formats are available for each character frame.

- Data bits
  In the "Data bits" drop-down list, select the number of bits onto which a character is to be mapped.
  The default setting is "8 bits".
- Stop bits
  In the "Stop bits" drop-down list, select the number of bits appended to every transmitted character. The stop bit identifies the end of a character.
  The default setting is "1 bit".
- Parity
  In the "Parity" drop-down list, select whether the data transmission reliability is to be increased by using a parity bit. A parity of "None" means that no parity bit is transmitted.
  The setting "None" is not possible in a 7-data-bit configuration.
  A sequence of information bits can be extended to include another bit, the parity bit. The addition of its value ("0" or "1") brings the value of all the bits up to a defined state – either even or odd.
  Default selection is "Even".
- Priority
  In the "Priority" drop-down list, select whether the CPU should have high or low priority. A partner has high priority if its send request takes precedence over the send request of the other partner. A partner has low priority if its send request must wait until the send request of the other partner has been dealt with. For the 3964(R) procedure, both communication partners must be assigned different priorities, in other words, one partner is assigned high and the other low priority.
  Default selection is "High".

Frame parameters

- With block check
  Click the "With block check" check box to specify whether to transmit frames with or without block check.
  - Without block check
    The protocol parameters can be assigned by the user.
    If the CPU recognizes the string DLE ETX, it stops receiving and sends a DLE to the communication partner if the block was received without errors or an NAK if the block was received but with errors.
  - With block check
    The protocol parameters are set to default values.
    If the CPU recognizes the string DLE ETX BCC, it stops receiving. It compares the received block check character BCC with the internally generated longitudinal parity. If the BCC is correct and no other receive error has occurred, it returns the character string DLE (the character string NAK is returned to the communication partner if an error has occurred).

  The "With block check" check box is selected by default.
- Use defaults
  Select the "Use defaults" check box if you want to assign the default values to the protocol parameters.
  The module terminates receiving when it detects the character string DLE ETX and returns a DLE (or NAK) string to the communication partner to confirm that it has received an error-free (of faulty) block.
  You can specify the following parameters individually should you decide not to use defaults.
  The "Use defaults" check box is selected by default.
- Character delay time
  In the "Character delay time" input field, you specify the maximum permissible time interval between two received characters within a frame. You can enter a value in 10 ms increments between "20 ms" and "65530 ms".
  The shortest character delay that you can select depends on the transmission speed.

Transmission speed	Shortest character delay time
300 bps	60 ms
600 bps	40 ms
1200 bps	30 ms
2400 bps	20 ms
4800 bps	20 ms
9600 bps	20 ms
19200 bps	20 ms
38400 bps	20 ms

The default setting is "220 ms".

- Acknowledgment delay time
  In the "Acknowledgment delay time" input field, specify the maximum permissible time period until the acknowledgment of the partner when a connection is established (time between STX and DLE acknowledgment of the partner) or closed (time between DLE ETX and DLE acknowledgment of the partner). You can enter a value in 10 ms increments between "20 ms" and "65530 ms".
  The shortest acknowledgment delay time that you can select depends on the transmission speed.

Transmission speed	Shortest acknowledgment delay time
300 bps	60 ms
600 bps	40 ms
1200 bps	30 ms
2400 bps	20 ms
4800 bps	20 ms
9600 bps	20 ms
19200 bps	20 ms
38400 bps	20 ms

Default is "2000 ms".

- Connection retries
  In the "Connection retries" list, specify the maximum number of tries the module has to establish a connection. You can select a value between "1" and "255".
  Default selection is "6".
- Transmission retries
  In the "Transmission retries" list, specify the maximum number of tries the module is allowed to transmit a frame (including the first frame) in case of error. You can select a value between "1" and "255".
  Default selection is "6".

PtP - data reception (3964(R))

Description

The interrupts, characteristics of a CPU STOP, and protocol are set in "Properties > General > PtP > Interrupt selection". Select "3964(R)" as the protocol.

Depending on the type of protocol selected, the further subdivision changes in the area navigation.

For "3964(R)" there are 3 PtP areas: transmission, data reception and signal assignment.

This topic deals with PtP data reception.

Reception buffer on CPU

- Clear CPU receive buffer during startup
  Select the "Clear CPU receive buffer during startup" check box to clear the receive buffer after power up and after the CPU transition from STOP to RUN.
  This check box is not selected by default.
- Prevent overwriting
  Select the "Prevent overwriting" check box to prevent overwriting data in the receive buffer when the buffer is full.
  This check box is selected by default.
- Use entire buffer
  Select the "Use entire buffer" check box if you want to use the entire 2048 bytes of the CPU receive buffer. When you use the complete buffer, the number of received frames is only dependent on the frame length.
  This check box is selected by default.
- Maximum number of buffered received frames
  In the "Maximum number of buffered received frames" input field, enter the number of received frames to be buffered in the CPU receive buffer. Enter a value between 1 and 10. Default setting is "10".

Note

If you specify "1" here and deactivate the "Prevent overwrite" parameter **and** cyclically read the received data from the user program, a current message frame will always be sent to the CPU.

PtP - signal assignment (3964(R))

Description

The interrupts, characteristics of a CPU STOP, and protocol are set in "Properties > General > PtP > Interrupt selection". Select "3964(R)" as the protocol.

Depending on the type of protocol selected, the further subdivision changes in the area navigation.

For "3964(R)" there are 3 PtP areas: transmission, data reception and signal assignment.

This section deals with PtP signal assignment.

Receive line initial state

With the 3964(R) protocol the X27 (RS 422/485) interface is operated in full duplex (RS 422) four-wire mode. This operating mode is suitable for four-wire PtP connections.

The following options are available as defaults for the receive line:

- None
  Select the "None" option to select a configuration that is only advisable for bus-compatible special drives.
- Signal R(A) 5 V, signal R(B) 0 V (break detection):
  With this option you select a setting that allows break detection.
- Signal R(A) 0 V, signal R(B) 5 V
  With this option you select a setting that corresponds to idle state (no sender active). No break detection is possible with this configuration.

"Signal R(A) 5 V, signal R(B) 0 V (break detection)" is selected as default.

RK512 protocol

PtP - transmission (RK512)

Description

The interrupts, characteristics of a CPU STOP, and protocol are set in "Properties > General > PtP > Interrupt selection". Select "RK512" as the protocol.

Depending on the type of protocol selected, the further subdivision changes in the area navigation.

For "RK512" there are 2 PtP areas: transmission and signal assignment.

This section deals with PtP transmission and is divided into 3 sub-sections.

Transmission speed

Select the data transmission rate in bps (baud) from the "Transmission speed" drop-down list. You can select a value between 300 and 38400 baud.

Default selection is "9600 baud".

Character frame

The data are transmitted via the serial interface in a character frame. Two data formats are available for each character frame.

- Data bits
  The "Data bits" drop-down list is grayed out as one character is always mapped onto 8 bits in RK512 transmission.
- Stop bits
  In the "Stop bits" drop-down list, select the number of bits appended to every transmitted character. The stop bit identifies the end of a character.
  The default setting is "1 bit".

- Parity
  In the "Parity" drop-down list, select whether the data transmission reliability is to be increased by using a parity bit. A parity of "None" means that no parity bit is transmitted. A sequence of information bits can be extended to include another bit, the parity bit. The addition of its value ("0" or "1") brings the value of all the bits up to a defined state – either even or odd.
  Default selection is "Even".
- Priority
  In the "Priority" drop-down list, select whether the CPU should have high or low priority. A partner has high priority if its send request takes precedence over the send request of the other partner. A partner has low priority if its send request must wait until the send request of the other partner has been dealt with. With RK512, both communication partners must be assigned different priorities, that is, one partner is assigned high and the other is assigned low priority.
  Default selection is "High".

Frame parameters

- With block check
  Click the "With block check" check box to specify whether to transmit frames with or without block check.
  - Without block check
    The protocol parameters can be assigned by the user.
    If the CPU recognizes the string DLE ETX, it stops receiving and sends a DLE to the communication partner if the block was received without errors or an NAK if the block was received but with errors.
  - With block check
    The protocol parameters are set to default values.
    If the CPU recognizes the string DLE ETX BCC, it stops receiving. It compares the received block check character BCC with the internally generated longitudinal parity. If the BCC is correct and no other receive error has occurred, it returns the character string DLE (the character string NAK is returned to the communication partner if an error has occurred).

  The "With block check" check box is selected by default.
- Use defaults
  Select the "Use defaults" check box if you want to assign default values to the protocol parameters.
  If the CPU recognizes the string DLE ETX, it stops receiving and sends a DLE to the communication partner if the block was received without errors or an NAK if the block was received but with errors.
  You can specify the following parameters individually should you decide not to use defaults.
  The "Use defaults" check box is selected by default.
- Character delay time
  In the "Character delay time" input field, you specify the maximum permissible time interval between two received characters within a frame. You can enter a value in 10 ms increments between "20 ms" and "65530 ms".
  The shortest character delay that you can select depends on the transmission speed.

Transmission speed	Shortest character delay time
300 bps	60 ms
600 bps	40 ms
1200 bps	30 ms
2400 bps	20 ms
4800 bps	20 ms
9600 bps	20 ms
19200 bps	20 ms
38400 bps	20 ms

The default setting is "220 ms".

- Acknowledgment delay time
  In the "Acknowledgment delay time" input field, specify the maximum permissible time period until the acknowledgment of the partner when a connection is established (time between STX and DLE acknowledgment of the partner) or closed (time between DLE ETX and DLE acknowledgment of the partner). You can enter a value in 10 ms increments between "20 ms" and "65530 ms".
  The shortest acknowledgment delay time that you can select depends on the transmission speed.

Transmission speed	Shortest acknowledgment delay time
300 bps	60 ms
600 bps	40 ms
1200 bps	30 ms
2400 bps	20 ms
4800 bps	20 ms
9600 bps	20 ms
19200 bps	20 ms
38400 bps	20 ms

Default selection is "2000 ms".

- Connection retries
  In the "Connection retries" list, specify the maximum number of tries the module has to establish a connection. You can select a value between 1 and 255.
  Default selection is "6".
- Transmission retries
  In the "Transmission retries" list, specify the maximum number of attempts the module is allowed to transmit a frame. You can select a value between "1" and "255".
  Default selection is "6".

PtP - signal assignment (RK512)

Description

The interrupts, characteristics of a CPU STOP, and protocol are set in "Properties > General > PtP > Interrupt selection". Select "RK512" as the protocol.

Depending on the type of protocol selected, the further subdivision changes in the area navigation.

There are 2 PtP areas for "RK512": transmission and signal assignment.

This section deals with PtP transmission and is divided into 3 sub-sections.

Receive line initial state

The X27 (RS 422/485) interface is operated in full duplex (RS 422) four-wire operating mode. This operating mode is suitable for four-wire PtP connections.

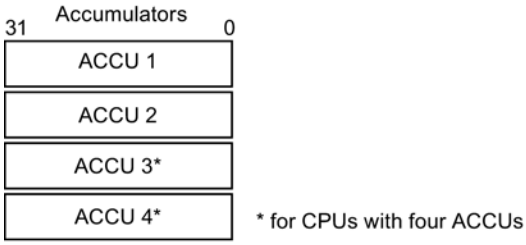
The following options are available as defaults for the receive line:

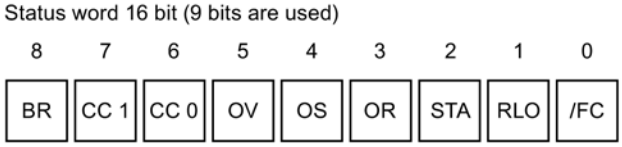
- None
  Select the "None" option to select a configuration that is only advisable for bus-compatible special drives.
- Signal R(A) 5 V, signal R(B) 0 V (break detection):
  With this option you select a setting that allows break detection.
- Signal R(A) 0 V, signal R(B) 5 V
  With this option you select a setting that corresponds to idle state (no sender active). No break detection is possible with this configuration.
- "Signal R(A) 5 V, signal R(B) 0 V (break detection)" is selected as default.

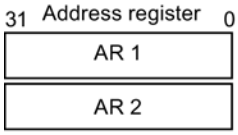
CPU information

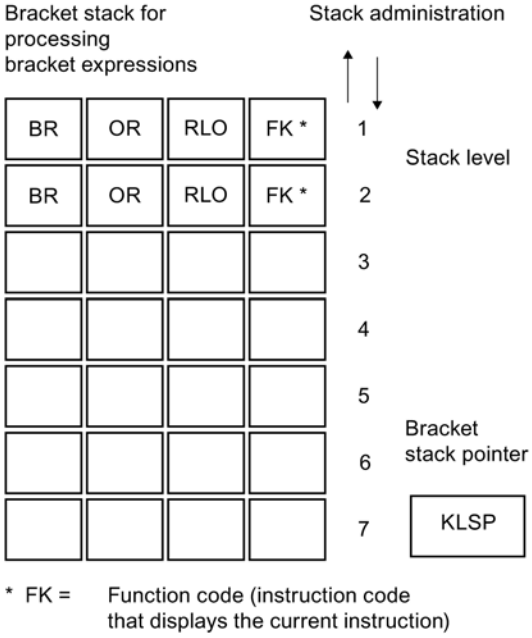
CPU registers

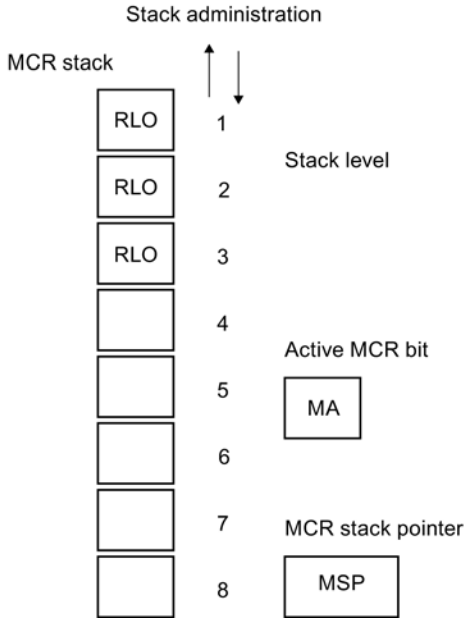
Overview











Accumulators

Description

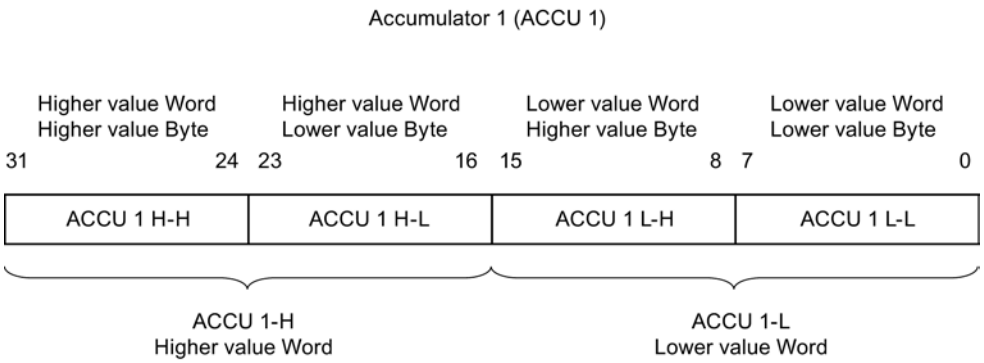
The 32-bit accumulators are general-purpose registers for processing bytes, words, and double words. For this purpose, the operands are loaded into the ACCUs and linked there.

The result of the operation is always located in ACCU 1 from where it can be transferred to a memory location.

The stack mechanism for managing the accumulators works as follows:

- A load operation only affects ACCU 1 and stores the old content in ACCU 2.
- A transfer operation (copy operation) does not change the contents of the accumulators.
- The TAK operation swaps the contents of ACCU 1 and ACCU 2.
- The result of logic operations between ACCU 1 and ACCU 2 (math operations, comparison operations, AND, OR, etc.) is always stored in ACCU 1.

Definition of bytes and words in a 32-bit register (based on the example of ACCU 1)



Addressing

Process images (input and output memory area)

Memory area of the process images

The OB-process image (OB1 PI) is cyclically updated by the operating system.

This process image is saved by the I/O hardware during each cycle of the instruction list. This means that the bits of the memory area I in the process image do not change during the access to the instructions. Even though the bits of the memory area Q in the process image can have different states during the program cycle, only the last signal state is sent to the output module.

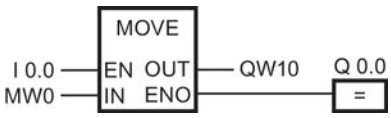
See also: User data (Page 1668) and Updating the process images (Page 1661)

Exceptions

The following cases do not use the process image to access the I/O:

- Direct I/O addressing of the inputs and outputs peripheral memory area (P area).
- Using functions that process the I/O data immediately.

Example of a normal I/O access via the process image



Update QW10 in the peripheral output with the value from MW0.

See also

Access to the I/O addresses (Page 1670)

Direct access to the I/O (PI and PQ memory areas)

Direct access to the I/O

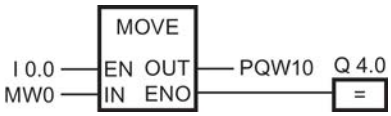
If your project needs to update the data of the I/O more quickly than the process image makes this available, you can access the signal modules directly in the peripheral inputs and outputs (PI, PQ). The current status of the input modules can be read from the PI memory area area. Output modules can be set by writing to the PQ memory area.

CAUTION

Single bits (byte. bit) cannot be addressed in the I/O area area. To set single bits, you address bytes, words or double words in logic operations. When you write an updated bit (within a byte, word, or double word) to the PQ area, all bits at the address of this I/O area will be updated.

Refer to the **Process image** entry, for more information on the connection between the I/O memory area (PI, PQ memory area) and the process image (I and Q memory area). The entry also explains how the process image saves (stabilizes) what has been written from the signal modules or sent to them.

Example of direct access to the I/O via the I/O memory area



Update PQW10 in the peripheral output with the value from MW0.

See also

Access to the I/O addresses (Page 1670)

Address areas (Page 1670)

Full syntax for entering constants

Overview

Short form	Long form
B#	BYTE#
W#	WORD#
DW#	DWORD#
D#	DATE#
T#	TIME#
TOD#	TIME_OF_DAY#
S5T#	S5TIME#
DT#	DATE_AND_TIME#
P#	POINTER#

Summary of bit, byte, word, and double word identifiers

Overview

CPU memory area	Operand ID			
	Bit	Byte	Word	Double word
Process image input	I	IB	IW	ID
Process image output	Q	QB	QW	QD
Bit memory	M	MB	MW	MD
Local data	L	LB	LW	LD
Direct peripheral input area		PIB	PIW	PID
Direct peripheral output area		PQB	PQW	PQD
Data block [OPN] DB [OPN] DI	 DBX DIX	 DBB DIB	 DBW DIW	 DBD DID
Timer (T)	T		T	
Counter (C)	C		C	

Note

The two operands **IB0** and **IB1** (memory area I, format: byte, address bytes 0, 1) address the same two bytes as the operand **IW0** (memory area I, format: word, 2 bytes, incrementing of byte 0 to address bytes 0, 1).

Identifiers of elements and blocks

Overview

T	Timer
DB	Global data block
DI	Instance data block
FC	Function
FB	Function block
OB	Organization block
C (or Z)	Counter (depending on the mnemonics)
SDB	System data block
SFC	System function
SFB	System function block

Symbolic and numerical names of instructions

Description

The instructions from the task card are comprised of functions (FC), function blocks (FB), system functions (SFC) and system function blocks (SFB) that are identified internally by numbers.

The following tables show the assignment of numerical and symbolic names.

Function blocks (FBs)

Numerical name	Symbolic name
FB 2	S_RCV
FB 3	S_SEND
FB 4	S_VSTAT
FB 5	S_VSET
FB 6	S_XON
FB 7	S_RTS
FB 8	S_V24
FB 8 (1)	USEND_S
FB 9 (1)	URCV_S
FB 14 (1)	GET_S
FB 15 (1)	PUT_S
FB 40 (1)	FTP_CMD
FB 46	PG_DIAL
FB 47	AS_DIAL
FB 48	SMS_SEND

Numerical name	Symbolic name
FB 49	AS_MAIL
FB 52 (1)	PNIO_RW_REC
FB 53	PE_DS3_Write_ET200S
FB 54 (1)	PNIO_ALARM
FB 55	IP_CONFIG
FB 56 (1)	LOGICAL_TRIGGER
FB 60	SET_SW
FB 61	SET_SW_S
FB 62	TIMESTMP
FB 80	LEAD_LAG
FB 81	DCAT
FB 81	S_MODB
FB 82	MCAT
FB 83	IMC
FB 84	SMC
FB 85	DRUM
FB 86	PACK
FB 210	FW_TCP
FB 220	FW_IOT
FB 450 (2)	RED_IN_MGP
FB 451 (2)	RED_OUT_MGP
FB 452 (2)	RED_DIAG_MGP
FB 453 (2)	RED_STATUS_MGP
FB 815	PE_Start_End_Pause
FB 816	PE_CMD
FB 817 (1)	PE_I_DEV
(1) for S7-300 (2) for S7-400 (3) not for WinAC (4) for S7-400, not for WinAC (5) for WinAC	

Functions (FCs)

Numerical name	Symbolic name
FC 0 (1)	PE_Error_RSP
FC 1 (1)	PE_Start_RSP
FC 1 (1)	DP_SEND
FC 2	CONCAT
FC 2 (1)	PE_End_RSP
FC 2 (1)	DP_RECV
FC 3 (1)	PE_List_Modes_RSP

Numerical name	Symbolic name
FC 3 (1)	DP_DIAG
FC 4	DELETE
FC 4 (1)	PE_Get_Mode_RSP
FC 4 (1)	DP_CTRL
FC 5 (1)	PE_PEM_Status_RSP
FC 5 (1)	AG_SEND
FC 6 (1)	PE_Identify_RSP
FC 6 (1)	AG_RECV
FC 7	Asi_3422
FC 7 (1)	PE_Measurement_List_RSP
FC 7 (1)	AG_LOCK
FC 8 (1)	PE_Measurement_Value_RSP
FC 8 (1)	AG_UNLOCK
FC 10 (1)	AG_CNTRL
FC 11	FIND
FC 11 (1)	PNIO_SEND
FC 12 (1)	PNIO_RECV
FC 17	INSERT
FC 17	S_USST
FC 18	S_USSR
FC 19	S_USSI
FC 20	LEFT
FC 21	LEN
FC 22	LIMIT
FC 25	MAX
FC 26	MID
FC 27	MIN
FC 31	REPLACE
FC 32	RIGHT
FC 36	ENCO
FC 36	SEL
FC 37	DECO
FC 60	LOC_TIME
FC 61	BT_LT
FC 62	LT_BT
FC 62 (1)	C_CNTRL
FC 63	S_LTINT
FC 74 (3)	I_ABORT
FC 80	TONR
FC 82	RSET
FC 83	SET
FC 84	ATT
FC 85	FIFO

Numerical name	Symbolic name
FC 86	TBL_FIND
FC 87	LIFO
FC 88	TBL
FC 89	TBL_WRD
FC 90	WSR
FC 91	WRD_TBL
FC 92	SHRB
FC 93	SEG
FC 98	BCDPL
FC 99	BITSUM
FC 100	RSETI
FC 101	SETI
FC 102	DEV
FC 103	CDT
FC 104	TBL_TBL
FC 105	SCALE
FC 106	UNSCALE
FC 450 (2)	RED_INIT_MGP
FC 451 (2)	RED_DEPA_MGP
(1) for S7-300 (2) for S7-400 (3) not for WinAC (4) for S7-400, not for WinAC (5) for WinAC	

System function blocks (SFBs)

Numerical name	Symbolic name
SFB 0	CTU
SFB 1	CTD
SFB 2	CTUD
SFB 3	TP
SFB 4	TON
SFB 5	TOF
SFB 16 (4)	PRINT
SFB 19 (2)	START
SFB 20 (2)	STOP
SFB 21 (4)	RESUME
SFB 22 (2)	STATUS
SFB 23 (2)	USTATUS
SFB 29 (1)	HS_COUNT_300C
SFB 30 (1)	FREQ_MES_300C

Numerical name	Symbolic name
SFB 31 (2)	NOTIFY_8P
SFB 32	DRUM_X
SFB 33 (2)	ALARM
SFB 34 (2)	ALARM_8
SFB 35 (2)	ALARM_8P
SFB 36 (2)	NOTIFY_8P
SFB 37 (2)	AR_SEND
SFB 38 (1)	HSC_A_B_300C
SFB 39 (1)	POS_300C
SFB 44 (1)	ANALOG_300C
SFB 46 (1)	DIGITAL_300C
SFB 47 (1)	COUNT_300C
SFB 48 (1)	FREQUENC_300C
SFB 49 (1)	PULSE_300C
SFB 52	RDREC
SFB 53	WRREC
SFB 54	RALRM
SFB 60 (1)	SEND_PTP_300C
SFB 61 (1)	RCV_PTP_300C
SFB 62 (1)	RES_RCVB_300C
SFB 63 (1)	SEND_RK_300C
SFB 64 (1)	FETCH_RK_300C
SFB 65 (1)	SERVE_RK_300C
SFB 73	RCVREC
SFB 74	PRVREC
SFB 75	SALRM
SFB 81	RD_DPAR
SFB 104	IP_CONF
SFB 65001 (5)	CREA_COM
SFB 65002 (5)	EXEC_COM
SFB 65002 (5)	ASYN_COM
(1) for S7-300 (2) for S7-400 (3) not for WinAC (4) for S7-400, not for WinAC (5) for WinAC	

System functions (SFCs)

Numerical name	Symbolic name
SFC 2	SET_RTM
SFC 3	CTRL_RTM

Numerical name	Symbolic name
SFC 4	READ_RTM
SFC 5	GADR_LGC
SFC 7 (3)	DP_PRAL
SFC 9 (2)	EN_MSG
SFC 10 (2)	DIS_MSG
SFC 11	DPSYC_FR
SFC 12	D_ACT_DP
SFC 13	DPNRM_DG
SFC 14	DPRD_DAT
SFC 17	ALARM_SQ
SFC 18	ALARM_S
SFC 19	ALARM_SC
SFC 20	BLKMOV
SFC 21	FILL
SFC 22	CREAT_DB
SFC 23	DEL_DB
SFC 24	TEST_DB
SFC 26 (2)	UPDAT_PI
SFC 27 (2)	UPDAT_PO
SFC 28	SET_TINT
SFC 29	CAN_TINT
SFC 30	ACT_TINT
SFC 31	QRY_TINT
SFC 32	SRT_DINT
SFC 33	CAN_DINT
SFC 34	QRY_DINT
SFC 35 (4)	MP_ALM
SFC 36	MSK_FLT
SFC 37	DMSK_FLT
SFC 38	READ_ERR
SFC 39	DIS_IRT
SFC 40	EN_IRT
SFC 41	DIS_AIRT
SFC 42	EN_AIRT
SFC 43	RE_TRIGR
SFC 44	REPL_VAL
SFC 46	STP
SFC 47	WAIT
SFC 48 (4)	SNC_RTCB
SFC 49	LGC_GADR
SFC 50	RD_LGADR
SFC 51	RDSYSST
SFC 52	WR_USMSG

Numerical name	Symbolic name
SFC 54 (2)	RD_DPARM
SFC 55	WR_PARM
SFC 56	WR_DPARM
SFC 57	PARM_MOD
SFC 58	WR_REC
SFC 59	RD_REC
SFC 60 (4)	GD_SND
SFC 61 (4)	GD_RCV
SFC 62 (2)	CONTROL
SFC 64	TIME_TCK
SFC 64	RD_SINFO
SFC 65 (3)	X_SEND
SFC 66 (3)	X_RCV
SFC 67 (3)	X_GET
SFC 68 (3)	X_PUT
SFC 69 (3)	X_ABORT
SFC 70	GEO_LOG
SFC 71	LOG_GEO
SFC 72 (3)	I_GET
SFC 73 (3)	I_PUT
SFC 75	SET_ADDR
SFC 78 (2)	OB_RT
SFC 79	SETP
SFC 80	RESETP
SFC 81	UBLKMOV
SFC 82 (1) (6)	CREA_DBL
SFC 83 (1) (6)	READ_DBL
SFC 84 (1) (6)	WRIT_DBL
SFC 85 (6)	CREA_DB
SFC 87 (2)	C_DIAG
SFC 90 (4)	H_CTRL
SFC 99	WWW
SFC 100 (2)	SET_CLKS
SFC 101	RTM
SFC 102 (1)	RD_DPARA
SFC 103 (3)	DP_TOPOL
SFC 104 (2)	CIR
SFC 105	READ_SI
SFC 106	DEL_SI
SFC 107	ALARM_DQ
SFC 108	ALARM_D
SFC 109	PROTECT
SFC 112	PN_IN

Numerical name	Symbolic name
SFC 113	PN_OUT
SFC 114	PN_DP
SFC 126	SYNC_PI
SFC 127	SYNC_PO
SFC 154	DPWR_DAT
(1) for S7-300 (2) for S7-400 (3) not for WinAC (4) for S7-400, not for WinAC (5) for WinAC	

Organization blocks

Overview of organization blocks

Organization blocks

Organization blocks (OBs) are the interface between the CPU operating system and the user program. OBs are used to execute specific program sections:

- At CPU startup
- In cyclic or clocked execution
- Whenever errors occur
- Whenever hardware interrupts occur.

Organization blocks are executed according to the priority they have been allocated.

Available OBs

Not all CPUs can process all OBs available in S7. See the operation list of your CPU to determine which OBs are available to you.

Start events and default priority class

The following table contains the start event belonging to each OB as well as the default priority class.

OB	Start event	Default priority class	Explanation
OB 1	End of startup or end of OB 1	1	Free cycle
OB 10	Time-of-day interrupt 0	2	No default time specified
OB 11	Time-of-day interrupt 1	2	
OB 12	Time-of-day interrupt 2	2	

OB	Start event	Default priority class	Explanation
OB 13	Time-of-day interrupt 3	2	
OB 14	Time-of-day interrupt 4	2	
OB 15	Time-of-day interrupt 5	2	
OB 16	Time-of-day interrupt 6	2	
OB 17	Time-of-day interrupt 7	2	
OB 20	Time-delay interrupt 0	3	No default time specified
OB 21	Time-delay interrupt 1	4	
OB 22	Time-delay interrupt 2	5	
OB 23	Time-delay interrupt 3	6	
OB 30	Cyclic interrupt 0 (default: 5 s sampling time)	7	Cyclic interrupts
OB 31	Cyclic interrupt 1 (default: 2 s sampling time)	8	
OB 32	Cyclic interrupt 2 (default: 1 s sampling time)	9	
OB 33	Cyclic interrupt 3 (default: 500 ms sampling time)	10	
OB 34	Cyclic interrupt 4 (default: 200 ms sampling time)	11	
OB 35	Cyclic interrupt 5 (default: 100 ms sampling time)	12	
OB 36	Cyclic interrupt 6 (default: 50 ms sampling time)	13	
OB 37	Cyclic interrupt 7 (default: 20 ms sampling time)	14	
OB 38	Cyclic interrupt 8 (default: 10 ms sampling time)	15	
OB 40	Hardware interrupt 0	16	Hardware interrupts
OB 41	Hardware interrupt 1	17	
OB 42	Hardware interrupt 2	18	
OB 43	Hardware interrupt 3	19	
OB 44	Hardware interrupt 4	20	
OB 45	Hardware interrupt 5	21	
OB 46	Hardware interrupt 6	22	
OB 47	Hardware interrupt 7	23	
OB 55	Status interrupt	2	DPV1 interrupts
OB 56	Update interrupt	2	
OB 57	Manufacturer-specific interrupt	2	
OB 61	Isochronous mode interrupt 1	25	Isochronous mode interrupts
OB 62	Isochronous mode interrupt 2	25	
OB 63	Isochronous mode interrupt 3	25	
OB 64	Isochronous mode interrupt 4	25	
OB 80	Time error	26, 28 [1]	Asynchronous error interrupts
OB 81	Power supply error	26, 28 [1] for S7-300, 25, 28 [1] for S7-400	
OB 82	Diagnostic interrupt	26, 28 [1] for S7-300, 25, 28 [1] for S7-400	
OB 83	Pull/plug interrupt	26, 28 [1] for S7-300, 25, 28 [1] for S7-400	

OB	Start event	Default priority class	Explanation
OB 84	CPU hardware fault	26, 28 [1] for S7-300, 25, 28 [1] for S7-400	
OB 85	Priority class error	26, 28 [1] for S7-300, 25, 28 [1] for S7-400	
OB 86	Failure of an expansion device, of a DP master system or a device for distributed I/O	26, 28 [1] for S7-300, 25, 28 [1] for S7-400	
OB 87	Communication error	26, 28 [1] for S7-300, 25, 28 [1] for S7-400	
OB 88	Processing interrupt	28	
OB 90	Warm restart or cold restart or deleting of a block being executed in OB 90, or loading of an OB 90 into the CPU or end of OB 90	29 [2)]	Background cycle
OB 100	Warm restart	27 [1]	Startup
OB 101	Hot restart	27 [1]	
OB 102	Cold restart	27 [1]	
OB 121	Programming error	Priority class of the OB that has caused the error	Synchronous error interrupts
OB 122	I/O access error	Priority class of the OB that has caused the error	
[1]Priority classes 27 and 28 are valid in the priority class model of the startup. [2]Priority class 29 corresponds to priority 0.29. This means that the background cycle has lower priority than does the free cycle.			

Cyclic program (OB 1)

Description

The operating system of the S7 CPU executes OB 1 cyclically. When OB 1 has been executed, the operating system starts executing OB 1 again. Cyclic execution of OB 1 is started after the startup has been completed. You can call instructions in OB 1.

Function of OB 1

OB 1 has the lowest priority of all OBs that are monitored for runtime. With the exception of OB 90 all other OBs can interrupt execution of OB 1. The following events cause the operating system to call OB 1:

- End of startup processing
- The execution of OB 1 (the previous cycle) has finished.

When OB 1 has been executed, the operating system sends global data. Before restarting OB 1, the operating system writes the process image output to the output modules, updates the process image input, and receives any global data for the CPU.

S7 monitors the maximum cycle time, ensuring a maximum response time. The value for the maximum cycle time is preset to 150 ms. You can set a new value or you can restart the time monitoring anywhere within your program with the "RE_TRIGR" instruction. If your program exceeds the maximum cycle time for OB 1, then the operating system calls OB 80 (time error OB). If OB 80 has not been programmed, the CPU changes to STOP mode.

Apart from monitoring the maximum cycle time, it is also possible to guarantee a minimum cycle time. The operating system will delay the start of a new cycle (writing of the process image output to the output modules) until the minimum cycle time has been reached.

The configuration is used to change the maximum and minimum cycle time parameters.

Local data for OB 1

The following table describes the temporary (TEMP) tags for OB 1. The tag names are the default names of OB 1.

Tag	Data type	Description
OB1_EV_CLASS	BYTE	Event class and identifiers: B#16#11: OB 1 is active
OB1_SCAN_1	BYTE	• B#16#01: Completion of the warm restart • B#16#02: Completion of the hot restart • B#16#03: Completion of the free cycle • B#16#04: Completion of the cold restart • B#16#05: First OB 1 cycle of the new master CPU after master-reserve switchover and STOP of the previous master
OB1_PRIORITY	BYTE	Priority class: 1
OB1_OB_NUMBR	BYTE	OB number (01)
OB1_RESERVED_1	BYTE	Reserved
OB1_RESERVED_2	BYTE	Reserved
OB1_PREV_CYCLE	INT	Runtime of last scan cycle (ms).
OB1_MIN_CYCLE	INT	Minimum cycle time (ms) since the last startup
OB1_MAX_CYCLE	INT	Maximum cycle time (ms) since the last startup
OB1_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

Time-of-day interrupt organization blocks (OB 10 to OB 17)

Description

S7 provides up to eight OBs (OB 10 to OB 17) which can be started once or periodically. You can assign parameters for your CPU in such a manner that these OBs are processed in the following intervals:

- Once
- Every minute
- Hourly

- Daily
- Weekly
- Monthly
- Yearly
- End of month

Note

For monthly execution of a time-of-day interrupt OB, only the days 1, 2, to 28 can be used as a starting date.

Function of time-of-day interrupt OBs

To start a time-of-day interrupt, you must first set and then activate the interrupt. The three following start possibilities exist:

- Automatic start of the time-of-day interrupt. This occurs if you have both set and activated the time-of-day interrupt per configuration. The following table shows the principle possibilities when activating a time-of-day interrupt per configuration.
- You set the time-of-day interrupt per configuration and then activate it by calling the "ACT_TINT (Page 6607)" instruction in your program.
- You set the time-of-day interrupt by calling the "SET_TINT (Page 6605)" instruction and then activate it by the "ACT_TINT (Page 6607)" instruction.

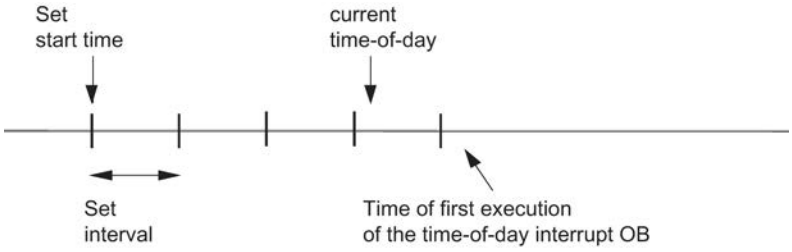
Interval	Description
Not activated	The time-of-day interrupt OB is not executed, even when it is loaded in the CPU. You can activate the time-of-day interrupt by calling the "ACT_TINT" instruction.
Activated once only	The time-of-day interrupt OB is canceled automatically after it runs once as specified. Your program can use the "SET_TINT" instruction to reset the time-of-day interrupt and the "ACT_TINT" instruction to reactivate it.
Activated periodically	When the time-of-day interrupt occurs, the CPU calculates the next start time for the time-of-day interrupt based on the current time of day and the period.

Note

If you configure a time-of-day interrupt in such a way that the corresponding OB is to be processed once, the date and time must not be in the past (relative to the real-time clock of the CPU).

If you configure a time-of-day interrupt in such a way that the corresponding OB is to be processed periodically, but the start date and time are in the past, then the time-of-day interrupt will be processed the next time it is due. This is illustrated in the following figure which shows the initial processing of a time-of-day interrupt OB if the start time is in the past, and if periodic activation has been set.

You can use the "DIS_IRT (Page 6624)" instruction to disable the call of time-of-day interrupt OBs, the "EN_IRT (Page 6626)" instruction to re-enable it, and the "DIS_AIRT (Page 6628)" and "EN_AIRT (Page 6628)" instructions to delay it.



Conditions that effect the time-of-day interrupt OBs

As a time-of-day interrupt occurs only at specific intervals, certain conditions can affect the function of the associated OBs during execution of your program. The following table shows some of these conditions and describes how these affect the processing of the time-of-day interrupt OB.

Condition	Result
Your program calls the "CAN_TINT (Page 6607)" instruction and cancels a time-of-day interrupt.	The operating system deletes the start event (date and time) of the time-of-day interrupt. You have to reset and activate the start event if the OB is to be called again.
Your program tried to activate a time-of-day interrupt OB that was not loaded in the CPU at the time of activation.	The operating system then calls OB 85. If OB 85 has not been programmed (loaded to the CPU), then the CPU switches to STOP mode.
By synchronizing or correcting the system clock of the CPU you preset the time, and the start event, the date or the time for a time-of-day interrupt OB is skipped.	The operating system calls OB 80 and codes the number of the time-of-day interrupt OB and the start event information in OB 80. The operating system then processes the time-of-day interrupt OB once, regardless of how often this OB actually should have been processed. The start event information of OB 80 shows the date and the time at which the time-of-day interrupt OB was initially skipped.
By synchronizing or correcting the system clock of the CPU you reset the time, and the start event, the date or the time for a time-of-day interrupt OB is repeated.	S7-400-CPUs: If the time-of-day interrupt OB was already activated before the time was reset, it is not called again for the times that have already passed. S7-300-CPUs: The time-of-day interrupt OB is carried out.

The CPU carries out a warm restart or a cold restart.	Each time-of-day interrupt OB that was configured by an instruction then adopts the configuration again that you have specified. If you have configured a time-of-day interrupt in such a way that the associated OB is started once and have set and activated it by means of configuration, then the OB will be called once after a warm restart or cold restart of the operating system as long as the configured starting time lies in the past (referring to the real time clock of the CPU).
A time-of-day interrupt OB is still being processed while the start event for the next interval is already occurring.	The operating system then calls OB 80. If OB 80 has not been programmed, the CPU changes to STOP mode. Otherwise, the requested OB execution takes place after processing of OB 80 and the time-of-day interrupt OB is complete.

Local data of the time-of-day interrupt OB

The following table includes the temporary (TEMP) tags of a time-of-day interrupt OB. The default names of OB 10 are selected as tag names.

Tag	Data type	Description
OB10_EV_CLASS	BYTE	Event class and identifiers: B#16#11: Interrupt is active
OB10_STRT_INFO	BYTE	B#16#11: Start request for OB 10 (B#16#12: Start request for OB 11) : : (B#16#18: Start request for OB 17)
OB10_PRIORITY	BYTE	Assigned priority class; default value: 2
OB10_OB_NUMBR	BYTE	OB number (10 to 17)
OB10_RESERVED_1	BYTE	Reserved
OB10_RESERVED_2	BYTE	Reserved
OB10_PERIOD_EXE	WORD	The OB is processed at the specified interval: W#16#0000: Once W#16#0201: Every minute W#16#0401: Hourly W#16#1001: Daily W#16#1201: Weekly W#16#1401: Monthly W#16#1801: Yearly W#16#2001: End of month
OB10_RESERVED_3	INT	Reserved
OB10_RESERVED_4	INT	Reserved
OB10_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

Time-delay interrupt organization blocks (OB 20 to OB 23)

Description

S7 provides up to four OBs (OB 20 to OB 23) which are executed after a specified delay. Every time-delay interrupt OB is started by calling the "SRT_DINT (Page 6611)" instruction. The delay time is an input parameter of the instruction.

When your program calls the "SRT_DINT (Page 6611)" instruction, you provide the OB number, the delay time, and a user-specific identifier. After the specified delay, the associated OB starts. You can also cancel execution of a time-delay interrupt that has not yet started.

Function of time-delay interrupt OBs

After the delay time has expired, whose value in milliseconds you have provided to the "SRT_DINT (Page 6611)" instruction together with an OB number, the operating system will start the corresponding OB.

To use the time-delay interrupts, you must perform the following tasks:

- You must call the "SRT_DINT (Page 6611)" instruction.
- You must download the time-delay interrupt OB to the CPU as part of your program.

Time-delay interrupt OBs are executed only when the CPU is in RUN mode. A warm restart or a cold restart clears any start events for a time-delay interrupt OB. If a time-delay interrupt has not yet been activated, you can use the "CAN_DINT (Page 6612)" instruction to cancel its execution.

The delay time is measured with a precision of 1 ms. A delay time can be immediately started after your sequence. You can query the status of a time-delay interrupt using the "QRY_DINT (Page 6613)" instruction.

The operating system calls an asynchronous error OB if one of the following events occur:

- If the operating system attempts to start an OB that is not loaded and you have specified its number when calling the "SRT_DINT (Page 6611)" instruction.
- If the next start event for a time-delay interrupt occurs before the corresponding time-delay interrupt OB has been completely executed.

You can use the "DIS_IRT (Page 6624)" instruction to disable the call of time-delay interrupt OBs, the "EN_IRT (Page 6626)" instruction to re-enable it, and the "DIS_AIRT (Page 6628)" and "EN_AIRT (Page 6628)" instructions to delay it.

Local data for time-delay interrupt OBs

The following table shows the temporary (TEMP) tags of a time-delay interrupt OB. The tag names are the default names of OB 20.

Tag	Data type	Declaration	Description
OB20_EV_CLASS	BYTE	TEMP	Event class and identifiers: B#16#11: Interrupt is active
OB20_STRT_INF	BYTE	TEMP	• B#16#21: Start request for OB 20 • (B#16#22: Start request for OB 21) • (B#16#23: Start request for OB 22) • (B#16#24: Start request for OB 23)
OB20_PRIORITY	BYTE	TEMP	Assigned priority class: Default values 3 (OB 20) to 6 (OB 23) Default value for S7-1500 CPUs: 3
OB20_OB_NUMBR	BYTE	TEMP	OB number (20 to 23)
OB20_RESERVED_1	BYTE	TEMP	Reserved
OB20_RESERVED_2	BYTE	TEMP	Reserved
OB20_SIGN	WORD	TEMP	User ID: Input parameter SIGN from the call of the "SRT_DINT (Page 6611)" instruction
OB20_DTIME	TIME	TEMP	Assigned delay time in ms
OB20_DATE_TIME	DATE_AND_TIME	TEMP	Date and time when the OB was called

Cyclic interrupt organization blocks (OB 30 to OB 38)

Description

S7 provides up to nine cyclic interrupt OBs (OB 30 to OB 38). You can use it to start programs after equidistant time phases. The following table shows the default values for the time frame and priority classes for the cyclic interrupt OBs.

Cyclic interrupt OB	Default value for time frame	Default value for the priority class
OB 30	5 s	7
OB 31	2 s	8
OB 32	1 s	9
OB 33	500 ms	10
OB 34	200 ms	11
OB 35	100 ms	12
OB 36	50 ms	13
OB 37	20 ms	14
OB 38	10 ms	15

Function of the cyclic interrupt OBs

The equidistant start times of the cyclic interrupt OBs are determined by the cycle clock and the phase offset.

Note

You must make sure that the run time of each cyclic interrupt OB is significantly shorter than its interval. If a cyclic interrupt OB has not been completely executed before it is due for execution again because the interval has expired, the time error OB (OB 80) is started. The cyclic interrupt that caused the error is executed later.

You can use the "DIS_IRT (Page 6624)" instruction to disable the call of the cyclic interrupt OBs, the "EN_IRT (Page 6626)" instruction to re-enable it, and the "DIS_AIRT (Page 6628)" and "EN_AIRT (Page 6628)" instructions to delay it.

Refer to the specifications of your specific CPU for the range of the parameters cycle clock, priority class, and phase offset. You can change parameter settings through configuration.

Local data for cyclic interrupt OBs

The following table shows the temporary (TEMP) tags of a cyclic interrupt OB.

Tag	Data type	Description
OB3x_EV_CLASS	BYTE	Event class and identifiers: B#16#11: Interrupt is active
OB3x_STRT_INF	BYTE	• B#16#30: Special start request for a cyclic interrupt OB in the H-system (assigned special handling on changeover to the "Redundant" system state) • B#16#31: Start request for OB 30 … • B#16#36: Start request for OB 35 … • B#16#39: Start request for OB 38 • B#16#3A: Start request for cyclic interrupt OBs (OB 30 to OB 38) with cyclic interrupt cycle less than one millisecond
OB3x_PRIORITY	BYTE	Assigned priority class; default values: 7 (OB 30) to 15 (OB 38) Default values for S7-1500 CPUs: 8 to 14; 16 to 17
OB3x_OB_NUMBR	BYTE	OB number (30 to 38)
OB3x_RESERVED_1	BYTE	Reserved
OB3x_RESERVED_2	BYTE	Reserved
OB3x_PHS_OFFSET	INT	for OBs 30 to 34 and 36 to 38: • If OB3x_STRT_INF=B#16#3A: phase offset in µs • In all other cases: phase offset in ms

Tag	Data type	Description
OB35_PHASE_OFFSET	WORD	for OB 35: • If OB35_STRT_INF=B#16#3A: phase offset in µs • In all other cases: phase offset in ms
OB3x_RESERVED_3	INT	Reserved
OB3x_EXC_FREQ	INT	• If OB3x_STRT_INF=B#16#3A: Cycle clock in µs • In all other cases: Cycle clock in milliseconds
OB3x_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called.

Hardware interrupt organization blocks (OB 40 to OB 47)

Description

S7 provides up to eight independent hardware interrupts each with its own OB.

Per configuration you specify which channels will trip a hardware interrupt,

- under which supplementary condition for each signal module.
- Which hardware interrupt OB is assigned to the individual groups of channels (as default, all hardware interrupts are processed by OB 40).

With CPs and FMs, you must use the appropriate software for the module for this.

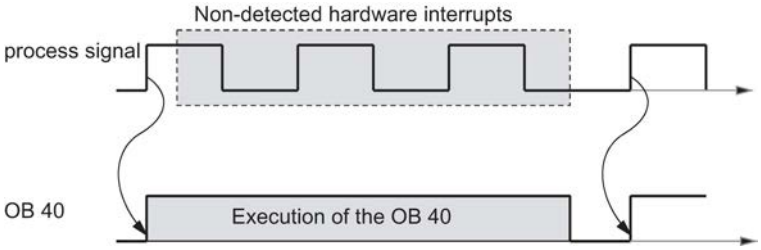
You select the priority classes for the individual hardware interrupt OBs per configuration.

Function of the hardware interrupt OBs

After the module triggers a hardware interrupt the operating system identifies the slot and determines the associated hardware interrupt OB. If this has a higher priority than the priority class active at the moment then it will be started. The channel-specific acknowledgement is sent after this hardware interrupt OB has been executed.

If another event that triggers a hardware interrupt occurs on the same module during the time between identification and acknowledgement of a hardware interrupt, the following applies:

- If the event occurs on the channel that previously triggered the hardware interrupt, then the associated interrupt is lost. The following figure illustrates the connection between a process signal and the execution of the associated hardware interrupt OB based on the example of a channel of a digital input module. The triggering event is the rising edge. The associated hardware interrupt OB is OB 40.



- If the event occurs on a different channel of the same module, then no hardware interrupt can be triggered at that moment. However, it is not lost, it is then triggered after the currently active hardware interrupt has been acknowledged.

If a hardware interrupt is triggered the OB of which is currently active on account of a hardware interrupt of another module, then the new request is registered and the OB is worked off at the specified time.

You can use the "DIS_IRT (Page 6624)" instruction to disable the call of the hardware interrupt OBs, the "EN_IRT (Page 6626)" instruction to re-enable it, and the "DIS_AIRT (Page 6628)" and "EN_AIRT (Page 6628)" instructions to delay it.

You can use the "WR_PARM (Page 6600)", " WR_DPARM (Page 6601)", and "PARM_MOD (Page 6596)" instructions to assign the hardware interrupt parameters for a module.

Local data of the hardware interrupt OBs

The following table shows the temporary (TEMP) tags of a hardware interrupt OB. The default names of OB 40 have been selected as tag names.

Tag	Data type	Description
OB40_EV_CLASS	BYTE	Event class and identifiers: B#16#11: Interrupt is active
OB40_STRT_INF	BYTE	• B#16#41: Interrupt via interrupt line 1 • B#16#42: Interrupt via interrupt line 2 (for S7-400 only) • B#16#43: Interrupt via interrupt line 3 (for S7-400 only) • B#16#44: Interrupt via interrupt line 4 (for S7-400 only) • B#16#45: WinAC: Interrupt triggered by PC Note: The interrupt lines 1, ... 4 are assigned in the multicomputing mode of the CPUs 1, ... 4.
OB40_PRIORITY	BYTE	Assigned priority class: Default values: 16 (OB 40) to 23 (OB 47) Default value for S7-1500 CPUs: 18
OB40_OB_NUMBR	BYTE	OB number (40 to 47)

Tag	Data type	Description
OB40_RESERVED_1	BYTE	Reserved
OB40_IO_FLAG	BYTE	Input module: B#16#54 Output module: B#16#55
OB40_MDL_ADDR	WORD	Logical start address of the module that has triggered the interrupt
OB40_POINT_ADDR	DWORD	• In the case of digital modules: Bit array containing the inputs on the module that have triggered the hardware interrupt Refer to the respective module description for information on the assignment of bits of OB40_POINT_ADDR to the module channels. • In the case of analog modules:bit field with the information as to which channel has exceeded which thresholds (refer to the module description for the precise structure). • In the case of CPs or IMs: Interrupt status of the module (not user-relevant)
OB40_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

Note

If you are using a DPV1-capable CPU, you can use the "RALRM (Page 6468)" instruction to obtain additional information about the interrupt that goes beyond the start information of the OB. This also applies if the DP master is operated in S7 compatible mode.

Status interrupt OB (OB 55)

Description

The CPU operating system calls OB 55 if a status interrupt was triggered by the slot of a DPV1 slave. This might be the case if a component (module or rack) of a DPV1 slave changes its operating mode, for example from RUN to STOP. For precise information on events that trigger a status interrupt, refer to the documentation provided by the DPV1 slave's manufacturer.

Note

If S7-400 CPUs or S7-400 PROFIBUS CPs are set to the "S7 compatible" DP mode, no status interrupt OB (OB 55) can be used.

Local data of the status interrupt OB

The following table shows the temporary (TEMP) tags of the status interrupt OB. The tag names are the default names of OB 55.

Tag	Data type	Description
OB55_EV_CLASS	BYTE	Event class and identifiers: B#16#11 (incoming event)
OB55_STRT_INF	BYTE	• B#16#55: Status interrupt for DP • B#16#58: Status interrupt for PROFINET IO
OB55_PRIORITY	BYTE	Assigned priority class, default value: 2 or 4 (for S7-1500 CPUs)
OB55_OB_NUMBR	BYTE	OB number (55)
OB55_RESERVED_1	BYTE	Reserved
OB55_IO_FLAG	BYTE	Input module: B#16#54 Output module: B#16#55
OB55_MDL_ADDR	WORD	Logical start address of the interrupt triggering component (module)
OB55_LEN	BYTE	Length of the data block supplied by the interrupt
OB55_TYPE	BYTE	ID for interrupt type "status interrupt"
OB55_SLOT	BYTE	Slot number of the component triggering the interrupt (module)
OB55_SPEC	BYTE	Specifier: • Bit 0 to 1: Interrupt specifier • Bit 2: Add_Ack • Bit 3 to 7: Seq. no.
OB55_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

Note

The meaning of OB55_LEN, OB55_TYPE, OB55_SLOT, OB55_SPEC indicated in the table above applies only to a status interrupt for DP. If the status interrupt is for PROFINET IO, you must organize the local tags as indicated in the next table.

Note

You can obtain complete supplemental information on the interrupt from the DP frame by calling the "RALRM (Page 6468)" instruction within OB 55.

If you want to program OB55 dependent on start events, we recommend that you organize the local tags as follows:

Tag	Data type
OB55_EV_CLASS	BYTE
OB55_STRT_INF	BYTE
OB55_PRIORITY	BYTE
OB55_OB_NUMBR	BYTE
OB55_RESERVED_1	BYTE
OB55_IO_FLAG	BYTE
OB55_MDL_ADR	WORD
OB55_Z2	WORD
OB55_Z3	WORD
OB55_DATE_TIME	DATE_AND_TIME

Depending on the start event, the tags OB55_Z2 and OB55_Z3 contain different information. This is explained in greater detail below.

Meaning of OB55_Z2

OB55_STRT_INF	Meaning of OB55_Z2
B#16#55	• Low byte: ID for interrupt type "status interrupt" • High byte: Length of the data block supplied by the interrupt
B#16#58	ID for interrupt type: • W#16#0000: Reserved • W#16#0001: Diagnostic interrupt (incoming) • W#16#0002: Hardware interrupt • W#16#0003: Pull interrupt • W#16#0004: Insert module interrupt • W#16#0005: Status interrupt • W#16#0006: Update interrupt • W#16#0007: Redundancy interrupt • W#16#0008: Controlled by supervisor • W#16#0009: Released • W#16#000A: Wrong submodule inserted • W#16#000B: Return of submodule • W#16#000C: Diagnostic interrupt (outgoing) • W#16#000D: Direct data exchange connection alarm • W#16#000E: Neighborhood change alarm • W#16#000F: Isochronous mode alarm (on bus side) • W#16#0010: Isochronous mode alarm (on device side) • W#16#0011: Network component alarm • W#16#0012: Time synchronization alarm (on bus side) • W#16#0013 to 001E: Reserved • W#16#001F: Pull module interrupt • W#16#0020 to 007F: Manufacturer-specific interrupt • W#16#0080 to FFFF: Reserved

Meaning of OB55_Z3

OB55_STRT_INF	Meaning of OB55_Z3
B#16#55	• Low byte: Specifier – Bits 0 to 1: Interrupt specifier – Bit 2: Add_Ack – Bits 3 to 7: Sequence number • High byte: Slot of component triggering the interrupt (module)
B#16#58	Interrupt specifier • Bits 0 to 10: Sequence number (value range 0 to 2047) • Bit 11: Channel diagnostics – 0: No channel diagnostics available – 1: Channel diagnostics available • Bit 12: Status of manufacturer-specific diagnostics: – 0: No manufacturer-specific status information available – 1: Manufacturer-specific status information available • Bit 13: Status of diagnostics for submodule: – 0: No status information available, all errors eliminated – 1: Either channel diagnostics or status information (or both) available • Bit 14: Reserved • Bit 15: Application relationship diagnostics state: – 0: None of the modules configured within this AR are reporting diagnostic information – 1: At least one of the modules configured in this AR is reporting diagnostic information

Update interrupt OB (OB 56)

Description

The CPU operating system calls OB 56 if an update interrupt was triggered via the slot of a DPV1 slave. This can be the case if you have changed the parameters for the slot of a DPV1 slave (via local or remote access). For precise information on events that trigger an update interrupt, refer to the documentation of the DPV1 slave‘s manufacturer.

Note

If S7-400 CPUs or S7-400 PROFIBUS CPs are set to the "S7 compatible" DP mode, no update interrupt OB (OB 56) can be used.

Local data of the update interrupt OB

The following table shows the temporary (TEMP) tags of the update interrupt OB. The tag names are the default names of OB 56.

Tag	Data type	Description
OB56_EV_CLASS	BYTE	Event class and identifiers: B#16#11 (incoming event)
OB56_STRT_INF	BYTE	• B#16#56: Update interrupt for DP • B#16#59: Update interrupt for PROFINET IO
OB56_PRIORITY	BYTE	Assigned priority class, default value: 2 or 4 (for S7-1500 CPUs)
OB56_OB_NUMBR	BYTE	OB number (56)
OB56_RESERVED_1	BYTE	Reserved
OB56_IO_FLAG	BYTE	Input module: B#16#54 Output module: B#16#55
OB56_MDL_ADDR	WORD	Logical start address of the interrupt triggering component (module)
OB56_LEN	BYTE	Length of the data block supplied by the interrupt
OB56_TYPE	BYTE	ID for the interrupt type “Update interrupt”
OB56_SLOT	BYTE	Slot number of the component triggering the interrupt (module)
OB56_SPEC	BYTE	Specifier: • Bit 0 to 1: Interrupt specifier • Bit 2: Add_Ack • Bit 3 to 7: Seq. no.
OB56_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

Note

The meaning of OB56_LEN, OB56_TYPE, OB56_SLOT, OB56_SPEC indicated in the table above applies only to an update interrupt for DP. If the update interrupt is for PROFINET IO, you must organize the local tags as indicated in the next table.

Note

You can obtain complete supplemental information on the interrupt from the DP frame by calling the "RALRM (Page 6468)" instruction within OB 56.

If you want to program OB 56 dependent on start events, we recommend that you organize the local tags as follows:

Tag	Data type
OB56_EV_CLASS	BYTE
OB56_STRT_INF	BYTE
OB56_PRIORITY	BYTE
OB56_OB_NUMBR	BYTE
OB56_RESERVED_1	BYTE
OB56_IO_FLAG	BYTE
OB56_MDL_ADR	WORD
OB56_Z2	WORD
OB56_Z3	WORD
OB56_DATE_TIME	DATE_AND_TIME

Depending on the start event, the tags OB56_Z2 and OB56_Z3 contain different information. This is explained in greater detail below.

Meaning of OB56_Z2

OB56_STRT_INF	Meaning of OB56_Z2
B#16#56	• Low byte: ID for interrupt type "status interrupt" • High byte: Length of the data block supplied by the interrupt
B#16#59	ID for the interrupt type • W#16#0000: Reserved • W#16#0001: Diagnostic interrupt (incoming) • W#16#0002: Hardware interrupt • W#16#0003: Pull interrupt • W#16#0004: Insert module interrupt • W#16#0005: Status interrupt • W#16#0006: Update interrupt • W#16#0007: Redundancy interrupt • W#16#0008: Controlled by supervisor • W#16#0009: Released • W#16#000A: Wrong submodule inserted • W#16#000B: Return of submodule • W#16#000C: Diagnostic interrupt (outgoing) • W#16#000D: Direct data exchange connection alarm • W#16#000E: Neighborhood change alarm • W#16#000F: Isochronous mode alarm (on bus side) • W#16#0010: Isochronous mode alarm (on device side) • W#16#0011: Network component alarm • W#16#0012: Time synchronization alarm (on bus side) • W#16#0013 to 001E: Reserved • W#16#001F: Pull module interrupt • W#16#0020 to 007F: Manufacturer-specific interrupt • W#16#0080 to FFFF: Reserved

Meaning of OB56_Z3

OB56_STRT_INF	Meaning of OB56_Z3
B#16#56	• Low byte: Specifier – Bits 0 to 1: Interrupt specifier – Bit 2: Add_Ack – Bits 3 to 7: Sequence number • High byte: Slot of component triggering the interrupt (module)
B#16#59	Interrupt specifier: • Bits 0 to 10: Sequence number (value range 0 to 2047) • Bit 11: Channel diagnostics – 0: No channel diagnostics available – 1: Channel diagnostics available • Bit 12: Status of manufacturer-specific diagnostics: – 0: No manufacturer-specific status information available – 1: Manufacturer-specific status information available • Bit 13: Status of diagnostics for submodule: – 0: No status information available, all errors eliminated – 1: Either channel diagnostics or status information (or both) available • Bit 14: Reserved • Bit 15: Application relationship diagnostics state: – 0: None of the modules configured within this AR are reporting diagnostic information – 1: At least one of the modules configured in this AR is reporting diagnostic information

OB for manufacturer-specific alarms (OB 57)

Description

The CPU operating system calls OB 57 if a manufacturer-specific interrupt was triggered via the slot of a DPV1 slave.

Note

If S7-400 CPUs or S7-400 PROFIBUS CPs are set to the "S7 compatible" DP mode, no manufacturer-specific interrupt OB (OB 57) can be used.

Local data of the manufacturer-specific interrupt OB

The table below contains the temporary (TEMP) tags of the manufacturer-specific interrupt OB. Selected tag names are the default names of OB 57.

Tag	Data type	Description
OB57_EV_CLASS	BYTE	Event class and identifiers: B#16#11 (incoming event)
OB57_STRT_INF	BYTE	• B#16#57: Manufacturer interrupt for DP • B#16#5A: Manufacturer interrupt for PROFINET IO • B#16#5B: IO: Profile-specific interrupt
OB57_PRIORITY	BYTE	Assigned priority class, default value: 2 or 4 (for S7-1500 CPUs)
OB57_OB_NUMBR	BYTE	OB number (57)
OB57_RESERVED_1	BYTE	Reserved
OB57_IO_FLAG	BYTE	Input module: B#16#54 Output module: B#16#55
OB57_MDL_ADDR	WORD	Logical start address of the interrupt triggering component (module)
OB57_LEN	BYTE	Length of the data block supplied by the interrupt
OB57_TYPE	BYTE	ID for the interrupt type “manufacturer-specific interrupt”
OB57_SLOT	BYTE	Slot number of the component triggering the interrupt (module)
OB57_SPEC	BYTE	Specifier: • Bit 0 to 1: Interrupt specifier • Bit 2: Add_Ack • Bit 3 to 7: Seq. no.
OB57_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

Note

The meaning of OB57_LEN, OB57_TYPE, OB57_SLOT, OB57_SPEC indicated in the table above applies only to a manufacturer-specific interrupt for DP. If the manufacturer-specific interrupt is for PROFINET IO, you must organize the local tags as indicated in the next table.

Note

You can obtain complete supplemental information on the interrupt from the DP message frame by calling the "RALRM (Page 6468)" instruction within OB 57.

If you want to program OB 57 dependent on start events, we recommend that you organize the local tags as follows:

Tag	Data type
OB57_EV_CLASS	BYTE
OB57_STRT_INF	BYTE
OB57_PRIORITY	BYTE
OB57_OB_NUMBR	BYTE
OB57_RESERVED_1	BYTE
OB57_IO_FLAG	BYTE
OB57_MDL_ADR	WORD
OB57_Z2	WORD
OB57_Z3	WORD
OB57_DATE_TIME	DATE_AND_TIME

Depending on the start event, the tags OB57_Z2 and OB57_Z3 contain different information. This is explained in greater detail below.

Meaning of OB57_Z2

OB57_STRT_INF	Meaning of OB57_Z2
B#16#57	• Low byte: ID for interrupt type "status interrupt" • High byte: Length of the data block supplied by the interrupt
B#16#5A	ID for interrupt type: • W#16#0000: Reserved • W#16#0001: Diagnostic interrupt (incoming) • W#16#0002: Hardware interrupt • W#16#0003: Pull interrupt • W#16#0004: Insert module interrupt • W#16#0005: Status interrupt • W#16#0006: Update interrupt • W#16#0007: Redundancy interrupt • W#16#0008: Controlled by supervisor • W#16#0009: Released • W#16#000A: Wrong submodule inserted • W#16#000B: Return of submodule • W#16#000C: Diagnostic interrupt (outgoing) • W#16#000D: Direct data exchange connection alarm • W#16#000E: Neighborhood change alarm • W#16#000F: Isochronous mode alarm (on bus side) • W#16#0010: Isochronous mode alarm (on device side) • W#16#0011: Network component alarm • W#16#0012: Time synchronization alarm (on bus side) • W#16#0013 to 001E: Reserved • W#16#001F: Pull module interrupt • W#16#0020 to 007F: Manufacturer-specific interrupt • W#16#0080 to FFFF: Reserved

Meaning of OB57_Z3

OB57_STRT_INF	Meaning of OB57_Z3
B#16#57	• Low byte: Specifier – Bits 0 to 1: Interrupt specifier – Bit 2: Add_Ack – Bits 3 to 7: Sequence number • High byte: Slot of component triggering the interrupt (module)
B#16#5A	Interrupt specifier • Bits 0 to 10: Sequence number (value range 0 to 2047) • Bit 11: Channel diagnostics – 0: No channel diagnostics available – 1: Channel diagnostics available • Bit 12: Status of manufacturer-specific diagnostics: – 0: No manufacturer-specific status information available – 1: Manufacturer-specific status information available • Bit 13: Status of diagnostics for submodule: – 0: No status information available, all errors eliminated – 1: Either channel diagnostics or status information (or both) available • Bit 14: Reserved • Bit 15: Application relationship diagnostics state: – 0: None of the modules configured within this AR are reporting diagnostic information – 1: At least one of the modules configured in this AR is reporting diagnostic information

Synchronous cycle interrupt OBs (OB 61 to OB 64)

Description

Isochronous mode interrupts give you the option of starting programs isochronously with the DP cycle or PN send clock. OB 6y, 1 <= y <= 4, serves as an interface OB to the isochronous mode interrupt TSAL y. You can set the priority for OB 61 to 64 between 0 (OB deselected) and from 2 to 26.

CAUTION

Direct access

In the case of direct access with L or T commands (e.g., L PIB, T PQB) or with use of the "DPRD_DAT (Page 6498)" and "DPWR_DAT (Page 6500)" instructions, avoid accessing I/O areas to which process image partitions with OB6x connection (isochronous mode interrupts) are assigned.

Local data for the isochronous mode interrupt OB

The following table describes the temporary (TEMP) tags of the isochronous mode interrupt OB. The default names of OB 61 have been selected as tag names.

Tag	Data type	Description
OB61_EV_CLASS	BYTE	Event class and identifiers: B#16#11: Interrupt is active
OB61_STRT_INF	BYTE	B#16#64: Start request for OB 61 : B#16#67: Start request for OB 64
OB61_PRIORITY	BYTE	Assigned priority class; default value: 25 or 21 (for S7-1500 CPUs)
OB61_OB_NUMBR	BYTE	OB number: 61 ... 64
OB61_RESERVED_1	BYTE	Reserved
OB61_RESERVED_2	BYTE	Reserved
OB61_GC_VIOL	BOOL	GC violation with PROFIBUS DP
OB61_FIRST	BOOL	First use after startup or stop status
OB61_MISSED_EXEC	BYTE	Number of failed starts of OB 61 since last execution of OB 61
OB61_DP_ID	BYTE	DP master system ID of isochronous DP master system (1 to 32) or PROFINET IO System ID of isochronous PNIO system (100 to 115)
OB61_RESERVED_3	BYTE	Reserved
OB61_RESERVED_4	WORD	Reserved
OB61_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

Time error OB (OB 80)

Description

The CPU operating system calls OB 80 whenever an error occurs while executing an OB, such errors include:

- Exceeding the cycle time
- Acknowledgement error in the execution of an OB
- Setting time forward (time skip) to start an OB

If for example a start event occurs for a cyclic interrupt OB, before the previous execution of the same OB has been completed, then the operating system calls OB 80.

If OB 80 has not been programmed, the CPU changes to STOP mode.

You can use the "DIS_IRT (Page 6624)" instruction to disable the call of the time error OB, the "EN_IRT (Page 6626)" instruction to re-enable it, and the "DIS_AIRT (Page 6628)" and "EN_AIRT (Page 6628)" instructions to delay it.

Note

If OB 80 is called twice during the same cycle due to the cycle time being exceeded, then the CPU changes to STOP mode. You can prevent this by calling the "RE_TRIGR" instruction at a suitable point.

Note on use of "Report System Errors"

"Report System Errors" delays the processing of interrupts that occur during its execution time.

If the **Execution time of "Report System Errors" > 50% of cycle time** condition is met, this effect is greater (here, the execution time of "Report System Errors" is understood to mean the sum of all processing times of "Report System Errors" within the cycle time). This can lead to the call of the time error OB and possibly to CPU STOP (if the time error OB is not loaded).

The following measures are available to prevent this:

- If you are using isochronous mode interrupt OBs (OB 61 to OB 64), you must ensure that these are called less frequently per time unit than before. You achieve this by increasing the associated "Application cycle factor" parameter for each isochronous mode interrupt OB.
- Increase the ratio (cycle time - execution time of "Report System Errors")/(execution time of "Report System Errors"). The following two options are available for this:
  - If you call "Report System Errors" in cyclic interrupt OBs (OB 30 to OB 38), then increase the cycle clocks for the associated OBs.
  - In OB 1, increase the execution time of the program section independent of "Report System Errors". You achieve this, for example, by using the "WAIT" instruction.

Local data for the time error OB

The following table lists the temporary (TEMP) tags of the time error OB. The default names of OB 80 have been selected as tag names.

Tag	Data type	Description
OB80_EV_CLASS	BYTE	Event class and identifiers: B#16#35
OB80_FLT_ID	BYTE	Error code (possible values: B#16#01, B#16#02, B#16#05, B#16#06, B#16#07, B#16#08, B#16#09, B#16#0A, B#16#0B)
OB80_PRIORITY	BYTE	Priority class: OB 80 runs with priority class 26 in RUN mode, and in the event of an overflow of the OB request buffer with priority class 28 For S7-1500 CPUs: Priority 22
OB80_OB_NUMBR	BYTE	OB number (80)
OB80_RESERVED_1	BYTE	Reserved

Tag	Data type	Description
OB80_RESERVED_2	BYTE	Reserved
OB80_ERROR_INFO	WORD	Error information: Depends on the error code
OB80_ERR_EV_CLASS	BYTE	Class of the event that caused the error
OB80_ERR_EV_NUM	BYTE	Number of the event that caused the error
OB80_OB_PRIORITY	BYTE	Error information: Depends on the error code
OB80_OB_NUM	BYTE	Error information: Depends on the error code
OB80_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

The error code-dependent tags have the following meaning:

Error code	Tag	Bit	Meaning
B#16#01			Cycle time exceeded
	OB80_ERROR_INFO		Runtime of last scan cycle (ms)
	OB80_ERR_EV_CLASS		Class of the event that triggered the interrupt
	OB80_ERR_EV_NUM		Number of the event that triggered the interrupt
	OB80_OB_PRIORITY		Priority class of the OB that was being processed when the error occurred
	OB80_OB_NUM		Number of the OB that was active when the error occurred
B#16#02			The requested OB is still being processed.
	OB80_ERROR_INFO		The corresponding temporary tag of the requested OB. This tag is determined by OB80_ERR_EV_CLASS and OB80_ERR_EV_NUM.
	OB80_ERR_EV_CLASS		Class of the event that triggered the interrupt
	OB80_ERR_EV_NUM		Number of the event that triggered the interrupt
	OB80_OB_PRIORITY		Priority class of the OB that caused the error (e.g.: "7" for OB 30/priority class 7, which should have been started but was not able to be started)
	OB80_OB_NUM		Number of the OB that caused the error (e.g.: "30" for OB 30, which should have been started but was not able to be started)
B#16#05			Expired time interrupt due to time jump
	OB80_ERROR_INFO	Bit 0 set	For the time interrupt 0, the starting time lies in the past
		...	
		Bit 7 set	For the time interrupt 7, the starting time lies in the past
		Bit 8 to 15	Not used
	OB80_ERR_EV_CLASS		Not used
	OB80_ERR_EV_NUM		Not used
	OB80_OB_PRIORITY		Not used
	OB80_OB_NUM		Not used
B#16#06	Same as for error code B#16#05		Expired time interrupt upon re-entry into RUN after HOLD

Error code	Tag	Bit	Meaning
B#16#07	Same as for error code B#16#02		Overflow of the OB request buffer for the current priority class (each OB start request for a priority class is entered in the associated OB request buffer; the entry is deleted again after the end of the OB. If a priority class has more OB start requests than the maximum possible number of entries in the associated OB request buffer, OB 80 is called with error code B#16#07.)
B#16#08	Same as for error code B#16#02		Isochronous mode interrupt - time error
B#16#09	Same as for error code B#16#02		Interrupt loss due to high interrupt load
B#16#0A			Resume RUN after CiR
	OB80_ERROR_INFO		CiR synchronization time in ms
B#16#0B			Technology synchronization interrupt - time error
	OB80_ERR_EV_NUM		Number of the event that triggered the interrupt: W#16#116A
	OB80_OB_PRIORITY		Priority class of the OB that was being processed when the error occurred
	OB80_OB_NUM		Number of the OB that was active when the error occurred: 65

Power supply error organization block (OB 81)

Description

The CPU operating system calls OB 81 whenever an event occurs that is triggered by an error or fault related to the power supply (only on an S7-400) or the back-up battery (for either an incoming or outgoing event).

In S7-400, OB 81 is only called in the event of a battery fault if the battery test function has been activated with the BATT.INDIC switch.

The CPU does not change to STOP mode if OB 81 is not programmed.

You can use the "DIS_IRT (Page 6624)" instruction to disable the call of the power supply error OB, the "EN_IRT (Page 6626)" instruction to re-enable it, and the "DIS_AIRT (Page 6628)" and "EN_AIRT (Page 6628)" instructions to delay it.

Local data for the power supply error OB

The following table lists the temporary (TEMP) tags of the power supply error OB. The default names of OB 81 have been selected as tag names.

Tag	Data type	Description
OB81_EV_CLASS	BYTE	Event class and identifiers: • B#16#38: Outgoing event • B#16#39: Incoming event
OB81_FLT_ID	BYTE	Error code (possible values: B#16#21, B#16#22, B#16#23, B#16#25, B#16#26, B#16#27, B#16#31, B#16#32, B#16#33)
OB81_PRIORITY	BYTE	Priority class: Can be set during configuration e.g., values 2 to 26 can be set for RUN mode.
OB81_OB_NUMBR	BYTE	OB number (81)
OB81_RESERVED_1	BYTE	Reserved
OB81_RESERVED_2	BYTE	Reserved
OB81_RACK_CPU	WORD	• Bits 0 to 7: B#16#00 • Bits 8 to 15: – For a standard CPU: B#16#00 – For a H-CPU: Bits 8 to 10: Rack no., bit 11: 0=Reserve CPU, 1=Master CPU, bits 12 to 15: 1111
OB81_RESERVED_3	BYTE	Relevant only for error codes B#16#31, B#16#32 and B#16#33
OB81_RESERVED_4	BYTE	Relevant only for error codes B#16#31, B#16#32 and B#16#33
OB81_RESERVED_5	BYTE	Relevant only for error codes B#16#31, B#16#32 and B#16#33
OB81_RESERVED_6	BYTE	Relevant only for error codes B#16#31, B#16#32 and B#16#33
OB81_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

The tags OB81_RESERVED_i, 3 ≤ i ≤ 6 indicate the expansion racks on which the battery backup (error code B#16#31), the back-up voltage (error code B#16#32) or the 24-V power supply (error code B#16#33) has failed or returned. The following table shows which bit is assigned to which expansion rack in the tags OB81_RESERVED_i, 3 ≤ i ≤ 6.

	OB81_RESERVED_6	OB81_RESERVED_5	OB81_RESERVED_4	OB81_RESERVED_3
Bit 0	Reserved	8. expansion rack	16. expansion rack	Reserved
Bit 1	1. expansion rack	9. expansion rack	17. expansion rack	Reserved
Bit 2	2. expansion rack	10. expansion rack	18. expansion rack	Reserved
Bit 3	3. expansion rack	11. expansion rack	19. expansion rack	Reserved
Bit 4	4. expansion rack	12. expansion rack	20. expansion rack	Reserved
Bit 5	5. expansion rack	13. expansion rack	21. expansion rack	Reserved

	OB81_RESERVED_6	OB81_RESERVED_5	OB81_RESERVED_4	OB81_RESERVED_3
Bit 6	6. expansion rack	14. expansion rack	Reserved	Reserved
Bit 7	7. expansion rack	15. expansion rack	Reserved	Reserved

The bits in the tags OB81_RESERVED_i have the following meaning (for the expansion rack concerned):

When the event occurs, the expansion racks are marked (the corresponding bits are set) on which at least one battery or back-up voltage or the 24 V power supply has failed. Expansion racks on which at least one battery or back-up voltage or the 24 V power supply failed earlier are no longer indicated. When the event is eliminated and the backup is restored on at least one expansion rack, this is signaled (the corresponding bits are set).

The following table shows the event that started OB81:

OB81_EV_CLASS	OB81_FLT_ID	Meaning
B#16#39/B#16#38	B#16#21	At least one back-up battery of the central rack is exhausted/problem eliminated (BATTF) **Note:** The incoming event occurs only if one of the two batteries (in the case of redundant buffer batteries) fails. If the other battery then fails, the event does not occur again.
B#16#39/B#16#38	B#16#22	Buffer voltage in the central device is missing / remedied (BAF).
B#16#39/B#16#38	B#16#23	Failure of the 24-V supply in the central device / remedied.
B#16#39/B#16#38	B#16#25	At least one buffer battery in at least one redundant central device is empty / remedied (BATTF).
B#16#39/B#16#38	B#16#26	Buffer voltage in at least one redundant central device is missing / remedied (BAF).
B#16#39/B#16#38	B#16#27	Failure of the 24-V supply in at least one redundant central device / remedied.
B#16#39/B#16#38	B#16#31	At least one buffer battery in at least one expansion unit is empty / remedied (BATTF).
B#16#39/B#16#38	B#16#32	Buffer voltage in at least one expansion unit is missing / remedied (BAF).
B#16#39/B#16#38	B#16#33	Failure of the 24-V supply in at least one expansion unit / remedied.

Sample program for OB 81

A sample STL program will show you how you can read the error code in OB 81.

The program is structured as follows:

The error code in OB 81 (OB81_FLT_ID) is read and compared with the identifier for the "At least one back-up battery of the central rack is exhausted" (B#16#21) and "Back-up voltage in the central rack is not present" (B#16#22) events.

If the error code corresponds to one of the indicated events, the program jumps to the BF label. Otherwise, the block is terminated.

Starting at the BF label, the program sets the *Battery error* tag if the event is an incoming event. If the event is an outgoing event, the program resets this tag.

```
L    B#16#21                                  //Identifier for the "At least one back-up battery
                                              of the central rack is exhausted" event
L    #OB81_FLT_ID                             //Error code in OB 81
==I                                           //If the same,
JC    BF                                      //then jump to BF
L    B#16#22                                  //Identifier for the "Back-up voltage in central
                                              rack is not present" event
==I                                           //If the same as the error code in OB 81,
JC    BF                                      //then jump to BF
BEU                                           //No alarm about battery error

BF:    L    B#16#39                           //Identifier for incoming event
L    #OB81_EV_CLASS                           //Event class and identifiers for OB 81 call
==I                                           //If the same,
S    Battery error                            //then set the battery error (tag from the tag
                                              table)
L    B#16#38                                  //Identifier for outgoing event
==I                                           //If the same as the event class and identifiers
                                              for the OB 81 call,
R    Battery error                            //then reset the battery error (tag from the tag
                                              table)
```

Diagnostic interrupt OB (OB 82)

Description

If a module with diagnostics capability for which you have enabled the diagnostic interrupt detects a change in its diagnostic state, it sends a diagnostic interrupt request to the CPU:

- There is a problem or a component requires maintenance or both (incoming event).
- There is no problem and no further components require maintenance (outgoing event).

The operating system then calls OB 82.

The local tags of OB 82 contain the logical start address as well as four bytes of diagnostics data of the defective module (see the following table).

If OB 82 has not been programmed, the CPU changes to STOP mode.

You can use the "DIS_IRT (Page 6624)" instruction to disable the call of the diagnostic interrupt OB, the "EN_IRT (Page 6626)" instruction to re-enable it, and the "DIS_AIRT (Page 6628)" and "EN_AIRT (Page 6628)" instructions to delay it.

Note

For PROFINET IO controllers, you can specify during configuration whether or not communication interrupts are to trigger a call of the diagnostic interrupt OB. These interrupts involve diagnostic events of the PROFINET interface.

Default setting: these events do not trigger an OB 82 call.

Local data for diagnostic interrupt OB

The following table lists the temporary (TEMP) tags of the diagnostic interrupt OB. The default names of OB 82 have been selected as the tag names.

Tag	Data type	Description
OB82_EV_CLASS	BYTE	Event class and identifiers: • B#16#38: outgoing event: module error-free (all errors have been corrected.) • B#16#39: incoming event: module will be or remain faulty (with change in the number of its pending errors)
OB82_FLT_ID	BYTE	Error code (B#16#42)
OB82_PRIORITY	BYTE	• Priority class; can be set via configuration • For S7-1500 CPUs: priority, default value: 5
OB82_OB_NUMBR	BYTE	OB number (82)
OB82_RESERVED_1	BYTE	Reserved
OB82_IO_FLAG	BYTE	• For S7-300/S7-400: – Input module: B#16#54 – Output module: B#16#55 • For S7-1500: zero
OB82_MDL_ADDR	WORD	• For S7-300/S7-400: Logical start address of the module where the fault occurred • For S7-1500: Hardware identifier of the hardware object that triggered the diagnostic interrupt
OB82_MDL_DEFECT	BOOL	Module is defective
OB82_INT_FAULT	BOOL	Internal fault
OB82_EXT_FAULT	BOOL	External fault
OB82_PNT_INFO	BOOL	Channel fault
OB82_EXT_VOLTAGE	BOOL	External auxiliary voltage missing
OB82_FLD_CONNCTR	BOOL	Front panel connector not plugged in
OB82_NO_CONFIG	BOOL	Module parameters not assigned
OB82_CONFIG_ERR	BOOL	Incorrect parameters on module

Tag	Data type	Description
OB82_MDL_TYPE	BYTE	• Bit 0 to 3: Module class • Bit 4: Channel information exists • Bit 5: User information exists • Bit 6: Diagnostic interrupt from substitute • Bit 7: Maintenance required
OB82_SUB_MDL_ERR	BOOL	Application module is missing or has an error
OB82_COMM_FAULT	BOOL	Communication problem
OB82_MDL_STOP	BOOL	Operating mode (0: RUN, 1: STOP)
OB82_WTCH_DOG_FLT	BOOL	Watchdog timer responded
OB82_INT_PS_FLT	BOOL	Internal power supply failed
OB82_PRIM_BATT_FLT	BOOL	Battery exhausted
OB82_BCKUP_BATT_FLT	BOOL	Entire backup failed
OB82_RESERVED_2	BOOL	Maintenance demanded
OB82_RACK_FLT	BOOL	Expansion rack failure
OB82_PROC_FLT	BOOL	Processor failure
OB82_EPROM_FLT	BOOL	EPROM fault
OB82_RAM_FLT	BOOL	RAM fault
OB82_ADU_FLT	BOOL	ADC/DAC error
OB82_FUSE_FLT	BOOL	Fuse tripped
OB82_HW_INTR_FLT	BOOL	Hardware interrupt lost
OB82_RESERVED_3	BOOL	Reserved
OB82_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

Note

If you are using a DPV1-capable CPU, you can use the "RALRM (Page 6468)" instruction to obtain additional information about the interrupt that goes beyond the start information of the OB. This also applies if the DP master is operated in S7 compatible mode.

Programming OB 82

You must add OB 82 to the blocks of your CPU. Write the program that is to be executed in OB 82 to the generated block, and load the block to the CPU as part of your user program.

You can use OB 82, for example, as follows:

- You evaluate the start information in OB 82.
- You carry out detailed diagnostics of the error that occurred.

If a diagnostic interrupt is triggered, the faulty module automatically enters 4 bytes of diagnostics data and their start address in the start information of the diagnostic interrupt OB

and in the diagnostics buffer. This provides you with information about when an error occurred and on which module.

With a suitable program in OB 82, you can evaluate additional diagnostics data for the faulty module (which channel the error occurred on, which error has occurred). You can use the "RDSYSST (Page 6681)" instruction to read out module diagnostics data and the "WR_USRMSG (Page 6665)" instruction to enter this information in the diagnostics buffer. You can also send a user-defined diagnostics alarm to a logged-on monitoring device.

Insert/remove module interrupt organization block (OB 83)

Description

The CPU operating system calls OB 83 after a configured module has been pulled or plugged in.

If you have not programmed OB 83, the CPU changes to STOP mode.

You can use the "DIS_IRT (Page 6624)" instruction to disable the pull/plug interrupt OB call, the "EN_IRT (Page 6626)" instruction to re-enable it, and the "DIS_AIRT (Page 6628)" and "EN_AIRT (Page 6628)" instructions to delay it.

Removing and inserting modules

Every removal and insertion of a configured module (not allowed: power supply modules, CPUs, adapter casings and IMs) in RUN, STOP, or STARTUP mode results in a pull/plug interrupt. This interrupt causes an entry in the diagnostics buffer and in the system status list for the CPU involved. The pull/plug OB is also started if the CPU is in RUN mode. If this OB has not been programmed, the CPU changes to STOP mode.

The removal and insertion of S7-400 modules is monitored within the system every second. To enable the CPU to detect the removal and insertion of an S7-400 module, a minimum time interval of two seconds must elapse between removal and insertion. This minimum time is slightly higher for other modules.

If you remove a configured module in RUN mode, OB 83 is started. An access error may be detected first if the module is accessed directly or when the process image is updated.

If you insert a module in a configured slot in RUN mode, the operating system checks whether the type of the module inserted matches the configuration. OB 83 is then started and parameters are assigned if the module types match.

Points to note about S7-300, ET 200S and ET 200Pro

- The removal and insertion of central IO devices is not permitted in S7-300.
- With S7-300 CPUs, there is a pull/plug interrupt only for 31x PN/DP CPUs, and then only for PROFINET IO components.

- On an ET 200Pro, for example with an IM 154-4-8 CPU, there is a pull/plug interrupt only for central IO devices.
- On an ET 200S, for example with an IM151-8 PN/DP CPU, there is a pull/plug interrupt only for central IO devices.

Local data for pull/plug OB

The following table lists the temporary (TEMP) tags of the pull/plug OB. The default names of OB 83 have been selected as the tag names.

Tag	Data type	Description
OB83_EV_CLASS	BYTE	Event class and identifiers: • B#16#32: End of module parameter reassignment • B#16#33: Start of module parameter reassignment • B#16#38: Module inserted • B#16#39: Module removed or not responding, or end of parameter reassignment
OB83_FLT_ID	BYTE	Error code (possible values: B#16#51, B#16#54, B#16#55, B#16#56, B#16#57, B#16#58, B#16#61, B#16#63, B#16#64, B#16#65, B#16#66, B#16#67, B#16#68)
OB83_PRIORITY	BYTE	Priority class; can be set via configuration For S71500 CPUs: priority, default value: 6
OB83_OB_NUMBR	BYTE	OB number (83)
OB83_RESERVED_1	BYTE	Identification of module or submodule/interface module
OB83_MDL_TD	BYTE	Range: • B#16#54: Peripheral range of inputs (PI) • B#16#55: Peripheral range of outputs (PQ)
OB83_MDL_ADDR	WORD	• Central or distributed PROFIBUS DP: Logical start address of the module affected. If it is a mixed module, it is the smallest logical address used in the module. If the I and O addresses in the mixed block are equal, the logical start address is the one that receives the event identifier. • Distributed PROFINET IO: Logical start address of the module/submodule
OB83_RACK_NUM	INT	• If OB83_RESERVED_1 =B#16#A0: number of the submodule/interface submodule (low byte) • If OB83_RESERVED_1 = B#16#C4: – central: rack number – distributed PROFIBUS DP: number of the DP device (low byte) and DP master system ID (high byte) – Distributed PROFINET IO: physical address: identifier bit (bit 15, 1 = PROFINET IO), IO system ID (bits 11 to 14) and device number (bits 0 to 10)

Tag	Data type	Description
OB83_MDL_TYPE	WORD	• Central or distributed PROFIBUS DP: Module type of the module affected (X: not relevant for user). Module types not listed here are documented in the manual of the relevant module. – W#16#X5XX: analog module – W#16#X8XX: function module – W#16#XCXX: CP – W#16#XFXX: digital module – W#16#8340: substitution type ID for input module – W#16#9340: substitution type ID for output module – W#16#A340: substitution type ID for combination module (I/O) – W#16#F340: substitution type ID for empty module or module that cannot be uniquely identified (e.g. with packed addresses) – Module types not listed here are documented in the manual of the relevant module. • Distributed PROFINET IO: – W#16#8101: module type of the inserted module is the same as the module type of the removed module – W#16#8102: module type of the inserted module is not the same as the module type of the removed module
OB83_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

The following table shows which event caused the start of OB 83:

OB83_EV_CLASS	OB83_FLT_ID	Meaning
B#16#39	B#16#51	PROFINET IO module removed
B#16#39	B#16#54	PROFINET IO submodule removed
B#16#38	B#16#54	PROFINET IO submodule inserted and matches configured submodule
B#16#38	B#16#55	PROFINET IO submodule inserted, but does not match configured submodule
B#16#38	B#16#56	PROFINET IO submodule inserted; but error in module parameter assignment
B#16#38	B#16#57	PROFINET IO submodule or module inserted, but with a problem or maintenance
B#16#38	B#16#58	PROFINET IO submodule, access error corrected
B#16#39	B#16#61	Module removed or not responding OB83_MDL_TYPE: Actual module type
B#16#38	B#16#61	Module inserted, module type OK OB83_MDL_TYPE: Actual module type

OB83_EV_CLASS	OB83_FLT_ID	Meaning
B#16#38	B#16#63	Module inserted but module type is incorrect OB83_MDL_TYPE: Actual module type
B#16#38	B#16#64	Module inserted but faulty (module ID cannot be read) OB83_MDL_TYPE: Configured module type
B#16#38	B#16#65	Module inserted but there is an error in module parameter assignment OB83_MDL_TYPE: Actual module type
B#16#39	B#16#66	Module not responding, load voltage error
B#16#38	B#16#66	Module responds again, load voltage error corrected
B#16#33	B#16#67	Start module parameter reassignment
B#16#32	B#16#67	End of module parameter reassignment
B#16#39	B#16#68	Module parameter reassignment terminated with error

Note

If you are using a DPV1 or PROFINET capable CPU, you can use the "RALRM (Page 6468)" instruction to obtain additional information about the interrupt that goes beyond the start information of the OB. This also applies if the DP master is operated in S7 compatible mode.

Programming OB 83

You must add OB 83 to the blocks of your CPU. Write the program that is to be executed in OB 83 to the generated block, and load the block to the CPU as part of your user program.

You can use OB 83, for example, as follows:

- You evaluate the start information in OB 83.
- You then assign parameters for the newly inserted module using the "WR_PARM (Page 6600)", "WR_DPARM (Page 6601)", "PARM_MOD (Page 6596)", "WR_REC (Page 6496)", and "RD_REC (Page 6491)" instructions.

CPU hardware fault organization block (OB 84)

Description

The CPU operating system calls OB 84 in the following cases:

- After memory errors have been detected and corrected
- For WinAC RTX: Errors in the operating system (e.g., "blue screen")

If you have not programmed OB 84, the CPU does not change to STOP mode

You can use the "DIS_IRT (Page 6624)" instruction to disable the CPU hardware error OB, the "EN_IRT (Page 6626)" instruction to re-enable it, and the "DIS_AIRT (Page 6628)" and "EN_AIRT (Page 6628)" instructions to delay it.

Local data for the CPU hardware fault OB

The following table includes the temporary (TEMP) tags of the CPU hardware fault. The default names of OB 84 have been selected as the tag names.

Tag	Data type	Description
OB84_EV_CLASS	BYTE	Event class and identifiers: • B#16#38: outgoing event • B#16#35, B#16#39: incoming event
OB84_FLT_ID	BYTE	Error code (B#16#82, B#16#83, B#16#85, B#16#86, B#16#87)
OB84_PRIORITY	BYTE	Priority class; can be set via configuration (hardware configuration)
OB84_OB_NUMBR	BYTE	OB number (84)
OB84_RESERVED_1	BYTE	Reserved
OB84_RESERVED_2	BYTE	Reserved
OB84_RESERVED_3	WORD	Reserved
OB84_RESERVED_4	DWORD	Reserved
OB84_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

The following table shows which event caused the start of OB 84:

OB84_EV_CLASS	OB84_FLT_ID	Start event of OB 84
B#16#35	B#16#82	Memory error in operating system detected and corrected
B#16#35	B#16#83	Accumulation of detected and corrected memory errors
B#16#35	B#16#85	Error in PC operating system
B#16#39	B#16#86	Performance of an H-Sync link affected
B#16#35	B#16#87	Multi-bit memory error detected and corrected

Programming OB 84

You must add OB 84 to the blocks of your CPU. Write the program that is to be executed in OB 84 to the generated block, and load the block to the CPU as part of your user program.

You can use OB 84, for example, as follows:

- You evaluate the start information in OB 84.
- You send an alarm to the diagnostics buffer using the "WR_USMSG (Page 6665)" instruction.

Priority class error organization block (OB 85)

Description

The CPU operating system calls OB 85 whenever one of the following events occurs:

- Start event for a non-loaded OB (except OB 80, OB 81, OB 82, OB 83, and OB 86)
- Error when the operating system accesses a block
- I/O access error during update of the process image by the system (if the OB 85 call was not suppressed due to the configuration).

Note

If OB 85 has not been programmed, the CPU changes to STOP mode when one of these events is detected.

You can use the "DIS_IRT (Page 6624)" instruction to disable the priority class error OB, the "EN_IRT (Page 6626)" instruction to re-enable it, and the "DIS_AIRT (Page 6628)" and "EN_AIRT (Page 6628)" instructions to delay it.

Local data for the priority class error OB

The following table lists the temporary (TEMP) tags of the priority class error OB. The default names of OB 85 have been selected as the tag names.

Tag	Data type	Description
OB85_EV_CLASS	BYTE	Event class and identifiers: B#16#35, B#16#38 (only for error code B#16#B3 and B#16#B4), B#16#39 (only for error code B#16#B1, B#16#B2, B#16#B3 and B#16#B4)
OB85_FLT_ID	BYTE	Error code (possible values: B#16#A1, B#16#A2, B#16#A3, B#16#A4, B#16#B1, B#16#B2, B#16#B3, B#16#B4)
OB85_PRIORITY	BYTE	Priority class; can be set via configuration
OB85_OB_NUMBR	BYTE	OB number (85)
OB85_RESERVED_1	BYTE	Reserved
OB85_RESERVED_2	BYTE	Reserved

Tag	Data type	Description
OB85_RESERVED_3	INT	Reserved
OB85_ERR_EV_CLASS	BYTE	Class of the event that caused the error
OB85_ERR_EV_NUM	BYTE	Number of the event that caused the error
OB85_OB_PRIOR	BYTE	Priority class of the OB that was being processed when the error occurred (only for some error codes; for more details, see below).
OB85_OB_NUM	BYTE	Number of the OB that was being processed when the error occurred (only for some error codes; for more details, see below).
OB85_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

If you want to program OB 85 dependent on the possible error codes, we recommend that you organize the local tags as follows:

Tag	Data type
OB85_EV_CLASS	BYTE
OB85_FLT_ID	BYTE
OB85_PRIORITY	BYTE
OB85_OB_NUMBR	BYTE
OB85_DKZ23	BYTE
OB85_RESERVED_2	BYTE
OB85_Z1	WORD
OB85_Z23	DWORD
OB85_DATE_TIME	DATE_AND_TIME

The following table shows which event caused OB 85 to start and how the error code-dependent tags are assigned.

OB85_EV_CLASS	OB85_FLT_ID	Meaning
B#16#35	B#16#A1	Your program or the operating system (based on your configuration) creates a start event for an OB that is not loaded on the CPU. • OB85_Z1: The corresponding temporary tag of the requested OB. This is determined by OB85_Z23. • OB85_Z23: – High word: Class and number of the event causing the OB call – Low word, high byte: Program level active at the time of the error Low word, low byte: Active OB
B#16#35	B#16#A2	Your program or the operating system (based on your configuration) creates a start event for an OB that is not loaded on the CPU. OB85_Z1 and OB85_Z23 as for OB85_FLT_ID=B#16#A1
B#16#35	B#16#A3	Error when the operating system accesses a block • OB85_Z1: Error ID of the operating system – High byte: 1=integrated function, 2=IEC timer – Low byte: 0=no error resolution, 1=block not loaded, 2=area length error, 3=write-protect error • OB85_Z23: – High word: Block number – Low word: Relative address of the MC7 command causing the error. The block type must be taken from OB 85_DKZ23 (B#16#88: OB, B#16#8C: FC, B#16#8E: FB, B#16#8A: DB).
B#16#35	B#16#A4	PROFINET interface DB cannot be addressed
B#16#34	B#16#A4	PROFINET interface DB can be addressed again

OB85_EV_CLASS	OB85_FLT_ID	Meaning
B#16#39	B#16#B1	I/O access error when updating the process image of the inputs • OB85_DKZ23: ID of the type of process image transfer during which the I/O access error occurred – B#16#10: Byte access – B#16#20: Word access – B#16#30: DWord access – B#16#56 or B#16#57: Transmitting a configured consistency range • OB85_Z1: Reserved for internal use by the CPU: Logical start address of the module If OB85_RESERVED_2 has the value B#16#76, then OB85_Z1 contains the return value of the instruction involved ("DPRD_DAT (Page 6498)", "DPWR_DAT (Page 6500)", "UPDAT_PI (Page 6452)", or "UPDAT_PO (Page 6454)"). • OB85_Z23: – Byte 0: If OB85_DKZ23=B#16#57: Process image partition no. – Byte 1: Irrelevant, if OB85_DKZ23=B#16#10, 20, or 30; length of the consistency range in bytes, if OB85_DKZ23=B#16#57 – Byte 0 and byte 1, if OB85_DKZ23=B#16#56: Length of the consistency range in bytes – Bytes 2 and 3: The I/O address causing the I/O access error, if OB85_DKZ23=B#16#10, 20, or 30; Logical start address of the consistency range, if OB85_DKZ23=B#16#57
B#16#39	B#16#B2	I/O access error when transferring the process image output to the output modules OB85_DKZ23, OB85_Z1 and OB85_Z23 as for OB85_FLT_ID=B#16#B1
You obtain the error codes B#16#B1 and B#16#B2, if you have configured the repeated OB 85 call for I/O access errors for the system-side process image update.		

OB85_EV_CLASS	OB85_FLT_ID	Meaning
B#16#39/B#16#38	B#16#B3	I/O access error when updating the process image input, incoming/outgoing • OB85_DKZ23: ID of the type of process image transfer during which the I/O access error occurred – B#16#10: Byte access – B#16#20: Word access – B#16#30: DWord access – B#16#56 or B#16#57: Transmitting a configured consistency range • OB85_Z1: Reserved for internal use by the CPU: Logical start address of the module If OB85_RESERVED_2 has the value B#16#76, then OB85_Z1 contains the return value of the instruction involved ("DPRD_DAT (Page 6498)", "DPWR_DAT (Page 6500)", "UPDAT_PI (Page 6452)", or "UPDAT_PO (Page 6454)"). • OB85_Z23: – Byte 0: If OB85_DKZ23=B#16#57: Process image partition no. – Byte 1: Irrelevant, if OB85_DKZ23=B#16#10, 20, or 30; length of the consistency range in bytes, if OB85_DKZ23=B#16#57 – Byte 0 and byte 1, if OB85_DKZ23=B#16#56: Length of the consistency range in bytes – Bytes 2 and 3: The I/O address causing the I/O access error, if OB85_DKZ23=B#16#10, 20, or 30; Logical start address of the consistency range, if OB85_DKZ23=B#16#57
B#16#39/B#16#38	B#16#B4	I/O access error when updating the process image of the outputs incoming/outgoing OB85_DKZ23, OB85_Z1, OB85_Z23 as for OB85_FLT_ID=B#16#B3
You obtain the error codes B#16#B3 and B#16#B4 if you only configured the OB 85 call for incoming and outgoing I/O access errors for system side process image table updating. After a cold or warm restart, all access to non-existing inputs and outputs will be reported as incoming I/O access errors during the next process image table update.		

Programming OB 85

You must add OB 85 to the blocks of your CPU. Write the program that is to be executed in OB 85 in the generated block, and load the block to the CPU as part of your user program.

You can use OB 85, for example, as follows:

- To evaluate the start information of OB 85 and to determine which module is missing or defective (information regarding the module start address).
- You determine the slot of the module involved using the "LGC_GADR (Page 6790)" instruction.

Rack failure organization block (OB 86)

Description

The CPU operating system calls OB 86 in the following cases:

- Failure of a central expansion unit (not for S7-300) is detected (both if incoming or outgoing event).
- Failure of a DP master system is detected (both if incoming or outgoing event).
- Failure of a distributed I/O device (PROFIBUS DP or PROFINET IO) is detected (both if incoming or outgoing event).
- You have deactivated a distributed I/O device (PROFIBUS DP or PROFINET IO) with the "D_ACT_DP (Page 6486)" instruction using MODE=4.
- You have activated a distributed I/O device (PROFIBUS DP or PROFINET IO) with the "D_ACT_DP (Page 6486)" instruction using MODE=3.
- The failure of a PROFINET IO system or a PROFINET IO station or some of the submodules of a PROFINET I-device has been detected.

If OB 86 has not been programmed, the CPU changes to STOP mode when this type of error is detected.

You can use the "DIS_IRT (Page 6624)" instruction to disable the rack failure OB, the "EN_IRT (Page 6626)" instruction to re-enable it, and the "DIS_AIRT (Page 6628)" and "EN_AIRT (Page 6628)" instructions to delay it.

Local data for the rack failure OB

The following table lists the temporary (TEMP) tags of the rack failure OB. The default names of OB 86 have been selected as the tag names.

Tag	Data type	Description
OB86_EV_CLASS	BYTE	Event class and identifiers: • B#16#32: Activation of a device with the "D_ACT_DP" instruction using MODE=3 • B#16#33: Deactivation of a device with the "D_ACT_DP" instruction using MODE=4 • B#16#38: outgoing event • B#16#39: incoming event
OB86_FLT_ID	BYTE	Error code (possible values: B#16#C1, B#16#C2, B#16#C3, B#16#C4, B#16#C5, B#16#C6, B#16#C7, B#16#C8, B#16#C9, B#16#CA, B#16#CB, B#16#CC, B#16#CD, B#16#CE, B#16#CF, B#16#F8, B#16#F9)
OB86_PRIORITY	BYTE	Priority class; can be set via configuration For S7-1500 CPUs: priority, default value: 6
OB86_OB_NUMBR	BYTE	OB number (86)
OB86_RESERVED_1	BYTE	Reserved
OB86_RESERVED_2	BYTE	Reserved
OB86_MDL_ADDR	WORD	Depends on the error code

Tag	Data type	Description
OB86_RACKS_FLTD	ARRAY [0 ..31] OF BOOL	Depends on the error code
OB86_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

If you want to program OB 86 depending on the possible error codes, we recommend that you organize the local tags as follows:

Tag	Data type
OB86_EV_CLASS	BYTE
OB86_FLT_ID	BYTE
OB86_PRIORITY	BYTE
OB86_OB_NUMBR	BYTE
OB86_RESERVED_1	BYTE
OB86_RESERVED_2	BYTE
OB86_MDL_ADDR	WORD
OB86_Z23	DWORD
OB86_DATE_TIME	DATE_AND_TIME

The following table shows which event caused the start of OB 86.

OB86_EV_CLASS	OB86_FLT_ID	Meaning
B#16#39	B#16#C1	Expansion rack failure • OB86_MDL_ADDR: Logical start address of the IM • OB86_Z23: contains a bit for each possible expansion rack: Each expansion rack that caused a call of OB 86 is reported as having failed (the respective bits are set). Expansion racks that previously failed are no longer shown. – Bit 0: always 0 – Bit 1: 1. expansion rack – : – Bit 21: 21. expansion rack – Bit 22 to 29: always 0 – Bit 30: Failure of at least one expansion rack in the SIMATIC S5 area – Bit 31: always 0
B#16#38	B#16#C1	Expansion rack operational again OB86_MDL_ADDR as for OB86_FLT_ID=B#16#C1. The expansion racks that are operational again are reported in OB86_Z23 (the respective bits are set).

OB86_EV_CLASS	OB86_FLT_ID	Meaning
B#16#38	B#16#C2	Expansion rack operational again (expansion rack failure with discrepancy between expected and actual configuration) • OB86_MDL_ADDR: Logical start address of the IM • OB86_Z23: Contains one bit for every possible expansion rack, see OB86_FLT_ID B#16#C1. Meaning of a set bit: In the affected expansion rack: – modules with incorrect module IDs exist. – Configured modules missing. – At least one module is defective.
B#16#39	B#16#C3	Distributed I/O: Failure of a DP master system Only an incoming event causes OB 86 start with error code B#16#C3. An outgoing event causes OB 86 start with error code B#16#C4 and event class B#16#38: The return of each lower-level DP device has an OB 86 start as consequence.) • OB86_MDL_ADDR: Logical start address of the DP master • OB86_Z23: DP master system ID: – Bit 0 to 7: Reserved – Bit 8 to 15: DP master system ID – Bit 16 to 31: Reserved
B#16#39/B#16#38	B#16#C4	Failure of a DP device • OB86_MDL_ADDR: Logical start address of the DP master • OB86_Z23: Address of the affected DP slave: – Bit 0 to 7: No. of the DP device – Bit 8 to 15: DP master system ID – Bit 16 to 30: Logical start address, if an S7 slave is used, or diagnostics address if a DP standard slave is used. – Bit 31: I/O identifier
B#16#39/B#16#38	B#16#C5	Return of a DP device, but device faulty OB86_MDL_ADDR and OB86_Z23 as with FLT_ID=B#16#C4

OB86_EV_CLASS	OB86_FLT_ID	Meaning
B#16#38	B#16#C6	Expansion unit return, but error in module parameter assignment • OB86_MDL_ADDR: Logical start address of the IM • OB86_Z23: contains a bit for each possible expansion rack: – Bit 0: always 0 – Bit 1: 1. expansion rack – … – Bit 21: 21. expansion rack – Bit 22 to 30: Reserved – Bit 31: always 0 • Meaning of the set bit: In the affected expansion rack, – modules with incorrect module IDs exist. – Modules with missing or incorrect parameters exist.
B#16#38	B#16#C7	Return of a DP device, but error in module parameter assignment • OB86_MDL_ADDR: Logical start address of the DP master • Address of the affected DP slave: – Bit 0 to 7: No. of the DP device – Bit 8 to 15: DP master system ID – Bit 16 to 30: Logical start address of the DP slave – Bit 31: I/O identifier
B#16#38	B#16#C8	Return of a DP device, however discrepancy in configured and actual configuration • OB86_MDL_ADDR: Logical start address of the DP master • OB86_Z23: Address of the affected DP slave: – Bit 0 to 7: No. of the DP device – Bit 8 to 15: DP master system ID – Bit 16 to 30: Logical start address of the DP slave – Bit 31: I/O identifier
B#16#32/B#16#33	B#16#C9	Activation/deactivation of a DP slave with the "D_ACT_DP" instruction using MODE=3/MODE=4 • OB86_MDL_ADDR: Logical start address of the DP master • OB86_Z23: Address of the affected DP slave – Bit 0 to 7: No. of the DP device – Bit 8 to 15: DP master system ID – Bit 16 to 30: Logical start address, if an S7 slave is used, or diagnostics address if a DP standard slave is used. – Bit 31: I/O identifier

OB86_EV_CLASS	OB86_FLT_ID	Meaning
B#16#39	B#16#CA	PROFINET IO system failure • OB86_MDL_ADDR: Logical start address of the IO controller • OB86_Z23: – Bit 0 to 10: 0 (station number) – Bit 11 to 14: IO system ID – Bit 15: 1 – Bit 16 to 31: 0
B#16#39/38	B#16#CB	PROFINET IO device failure/device return • OB86_RESERVED_1: B#16#C4 • OB86_MDL_ADDR: Logical start address of the IO controller • OB86_Z23: – Bit 0 to 10: Station number – Bit 11 to 14: IO system ID – Bit 15: 1 – Bit 16 to 30: Logical start address of the station – Bit 31: I/O identifier Note: Before you check in your program whether a PROFINET IO station failure is the cause for the OB 86 start, you should check whether a PROFINET IO system failure occurred.
B#16#/38	B#16#CC	PROFINET IO station return with problem or maintenance • OB86_RESERVED_1: B#16#C4 • OB86_MDL_ADDR: Logical start address of the IO controller • OB86_Z23: – Bit 0 to 10: Station number – Bit 11 to 14: IO system ID – Bit 15: 1 – Bit 16 to 30: Logical start address of the station – Bit 31: I/O identifier
B#16#38	B#16#CD	PROFINET IO device return, expected and actual configuration differ • OB86_MDL_ADDR: Logical start address of the IO controller • OB86_Z23: – Bit 0 to 10: Station number – Bit 11 to 14: IO system ID – Bit 15: 1 – Bit 16 to 30: Logical start address of the station – Bit 31: I/O identifier

OB86_EV_CLASS	OB86_FLT_ID	Meaning
B#16#38	B#16#CE	PROFINET IO station return, error in module parameter assignment • OB86_MDL_ADDR: Logical start address of the IO controller • OB86_Z23: – Bit 0 to 10: Station number – Bit 11 to 14: IO system ID – Bit 15: 1 – Bit 16 to 30: Logical start address of the station – Bit 31: I/O identifier
B#16#32/B#16#33	B#16#CF	Activation/deactivation of a PROFINET IO device with the "D_ACT_DP" instruction using MODE=3/MODE=4 • OB86_MDL_ADDR: Logical start address of the IO controller • OB86_Z23: – Bit 0 to 10: Station number – Bit 11 to 14: IO system ID – Bit 15: 1 – Bit 16 to 30: Logical start address of the station – Bit 31: I/O identifier
B#16#39/B#16#38	B#16#F8	Failure/return of some of the submodules of a PROFINET I-device • OB86_MDL_ADDR: Logical start address of the IO controller • OB86_Z23: – Bit 0 to 10: Station number – Bit 11 to 14: IO system ID – Bit 15: 1 – Bit 16 to 30: Logical start address of the station – Bit 31: I/O identifier
B#16#38	B#16#F9	Return of some of the submodules of a PROFINET I-device with a device configuration difference • OB86_MDL_ADDR: Logical start address of the IO controller • OB86_Z23: – Bit 0 to 10: Station number – Bit 11 to 14: IO system ID – Bit 15: 1 – Bit 16 to 30: Logical start address of the station – Bit 31: I/O identifier

Note

If you are using a DPV1 or PROFINET capable CPU, you can use the "RALRM (Page 6468)" instruction to obtain additional information about the interrupt that goes beyond the start information of the OB. This also applies if the DP master is operated in S7 compatible mode.

Programming OB 86

You must add OB 86 to the blocks of your CPU. Write the program that is to be executed in OB 86 to the generated block, and load the block to the CPU as part of your user program.

You can use OB 86, for example, as follows:

- To evaluate the start information of OB 86 and to determine which rack is defective or missing
- You enter an alarm in the diagnostic buffer using the "WR_USMSG (Page 6665)" instruction and send the alarm to a monitoring device.

Communication error organization block (OB 87)

Description

The CPU operating system calls OB 87 when an event occurs that was triggered by a communication error.

If you have not programmed OB 87 and a start event for OB 87 occurs, your CPU will react as follows:

- An S7-300 CPU changes to STOP mode.
- An S7-400 CPU does not change to STOP mode.

You can use the "DIS_IRT (Page 6624)" instruction to disable the communication error OB, the "EN_IRT (Page 6626)" instruction to re-enable it, and the "DIS_AIRT (Page 6628)" and "EN_AIRT (Page 6628)" instructions to delay it.

Local data for communication error OB

The following table lists the temporary (TEMP) tags of the communication error OB. The default names of OB 87 have been selected as the tag names.

Tag	Data type	Description
OB87_EV_CLASS	BYTE	Event class and identifiers: B#16#39
OB87_FLT_ID	BYTE	Error code (possible values: B#16#D2, B#16#D3, B#16#D4, B#16#D5, B#16#E1, B#16#E2, B#16#E3, B#16#E4, B#16#E5, B#16#E6)
OB87_PRIORITY	BYTE	Priority class; can be set via configuration
OB87_OB_NUMBR	BYTE	OB number (87)
OB87_RESERVED_1	BYTE	Reserved
OB87_RESERVED_2	BYTE	Reserved
OB87_RESERVED_3	WORD	Depends on the error code
OB87_RESERVED_4	DWORD	Depends on the error code
OB87_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

The error code-dependent tags have the following meaning:

Error code	Tag	Meaning
B#16#D2		It is not possible to send the diagnostics entries at present.
B#16#D3		Synchronization message frames cannot be sent (master).
B#16#D4		Illegal time jump through time synchronization
B#16#D5		Error in adopting the synchronization time (slave)
	OB87_RESERVED_3	Includes no extra information
	OB87_RESERVED_4	Includes no extra information
B#16#E1		Incorrect message frame identifier in global data communication
B#16#E2		GD package status cannot be entered in a DB
	OB87_RESERVED_3	DB number
	OB87_RESERVED_4	• High word: Includes no extra information
		• Low word: GD circle number (high byte), GD packet number (low byte)
B#16#E3		Message frame length error in global data communication
B#16#E4		Illegal GD package number received
	OB87_RESERVED_3	Interface identifier (0: K bus, 1: MPI)
	OB87_RESERVED_4	• High byte: GD loop number • Low byte: Includes no extra information
B#16#E5		Error accessing DB when exchanging data via communication function blocks
	OB87_RESERVED_1	Block type: B#16#88: OB, B#16#8A: DB, B#16#8C: FC, B#16#8E: FB
	OB87_RESERVED_3	Reserved for internal use by the CPU
	OB87_RESERVED_4	• High word: Number of the block with the MC7 instruction that has caused the error • Low word: Relative address of the MC7 instruction that has caused the error

Error code	Tag	Meaning
B#16#E6		GD package status cannot be entered in a DB
	OB87_RESERVED_3	DB number
	OB87_RESERVED_4	Includes no extra information

Programming OB 87

You must add OB 87 to the blocks of your CPU. Write the program that is to be executed in OB 87 to the generated block, and load the block to the CPU as part of your user program.

You can use OB 87, for example, as follows:

- You evaluate the start information in OB 87.
- You create a data block when the data block for the status information of the shared data communication is missing.

Processing interrupt OB (OB 88)

Description

The CPU operating system calls OB 88 after a program block execution is been aborted. Possible causes of the abort are:

- Nesting depth exceeded for synchronous errors
- Nesting depth exceeded for block calls (U stack)
- Error during allocation of local data

If you have not programmed OB 88 and processing is interrupted, the CPU goes to STOP mode (event W#16#4570).

If program block execution is aborted in priority class 28, the CPU goes into STOP mode.

You can use the "DIS_IRT (Page 6624)" instruction to disable the processing interrupt OB, the "EN_IRT (Page 6626)" instruction to re-enable it, and the "DIS_AIRT (Page 6628)" and "EN_AIRT (Page 6628)" instructions to delay it.

Local data of the processing interrupt OB

The following table lists the temporary (TEMP) tags of the processing interrupt OB. The default names of OB 88 have been selected as the tag names.

Tag	Data type	Description
OB88_EV_CLASS	BYTE	Event class and identifiers: B#16#35
OB88_SW_FLT	BYTE	Error code Possible values: • B#16#71: Nesting depth of nesting levels exceeded • B#16#72: Nesting depth of Master Control Relay exceeded • B#16#73: Nesting depth exceeded for synchronous errors • B#16#74: Nesting depth of block calls (U stack) exceeded • B#16#75: Nesting depth of block calls (B stack) exceeded • B#16#76: Error in local data allocation • B#16#78: Unknown instruction • B#16#7A: Jump instruction with destination outside of the block Please refer to the operation list for your CPU to determine which error codes apply to your CPU.
OB88_PRIORITY	BYTE	Priority class: 28
OB88_OB_NUMBR	BYTE	OB number (88)
OB88_BLK_TYPE	BYTE	Type of block in which the error occurred: • B#16#88: OB • B#16#8C: FC • B#16#8E: FB • B#16##00: Could not determine interrupt source
OB88_RESERVED_1	BYTE	Reserved
OB88_FLT_PRIORITY	BYTE	Priority class of the OB that has caused the error
OB88_FLT_OB_NUMBR	BYTE	Number of the OB that has caused the error
OB88_BLK_NUM	WORD	Number of the block with the MC7 instruction that has caused the error
OB88_PRG_ADDR	WORD	Relative address of the MC7 instruction that has caused the error
OB88_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

Programming OB 88

You must add OB 88 to the blocks of your CPU. Write the program that is to be executed in OB 88 to the generated block, and load the block to the CPU as part of your user program.

You can use OB 88, for example, as follows:

- You evaluate the start information in OB 88.
- You enter the cause of the error in an alarm data block.

Background organization block (OB 90)

Description

With S7 you can monitor a maximum cycle time and guarantee a minimum cycle time. If the execution time of OB 1 including all the nested interrupt processing and system activities is less than the minimum cycle time that you have specified, the operating system reacts as follows:

- It calls the background OB (providing it exists on the CPU).
- It delays the next OB 1 start (if OB 90 does not exist on the CPU).

Function of the background OB

OB 90 has the lowest priority of all OBs. It is interrupted by any system activity and any interrupt (even by OB 1 after the minimum cycle time has elapsed) and is only resumed if the selected minimum cycle time has not yet been reached. The only exception is the execution of instructions that are started in OB 90. These are executed with the priority of OB 1 and are therefore not interrupted by OB 1. There is no time monitoring of OB 90.

The user program in OB 90 is processed starting with the first instruction in the following situations:

- Following a warm, cold, or hot restart
- After loading or deleting a block
- After downloading OB 90 to the CPU in RUN mode
- After terminating the background cycle

Note

In configurations in which the minimum cycle time and the maximum cycle time are almost the same, the cycle time may be exceeded unexpectedly when instructions are called in the background OB.

Local data for the background OB

The following table describes the temporary (TEMP) tags of OB 90. The tag names are the default names of OB 90.

Tag	Data type	Description
OB90_EV_CLASS	BYTE	Event class and identifiers: B#16#11: Active
OB90_STRT_INF	BYTE	• B#16#91: Warm restart/cold restart/hot restart • B#16#92: Deletion of a block • B#16#93: Downloading of OB 90 to the CPU in RUN mode • B#16#95: Termination of the background cycle
OB90_PRIORITY	BYTE	Priority class: 29 (corresponds to priority 0.29)
OB90_OB_NUMBR	BYTE	OB number (90)
OB90_RESERVED_1	BYTE	Reserved
OB90_RESERVED_2	BYTE	Reserved
OB90_RESERVED_3	INT	Reserved
OB90_RESERVED_4	INT	Reserved
OB90_RESERVED_5	INT	Reserved
OB90_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

Startup organization blocks (OB 100, OB 101 and OB 102)

Startup types

A distinction is made between the following types of startup

- Hot restart (S7-400)
- Warm restart
- Cold restart

In the following table, you can see which OB is called by the operating system during startup.

Startup type	Corresponding OB
Hot restart	OB 101
Warm restart	OB 100
Cold restart	OB 102

Startup events

The CPU executes a startup as follows:

- After POWER ON
- After a request by a communication function (via a menu command from the programming device or by calling the "START (Page 7400)" or "RESUME (Page 7404)" instruction on a different CPU)

Depending on the start event, the particular CPU, and its parameters, the appropriate startup OB (OB 100, OB 101, or OB 102) is called. With suitable programming, you can make certain default settings for your cyclic program in these blocks.

Local data for startup OBs

The following table shows the temporary (TEMP) tags of a startup OB. The default names have been selected as tag names.

Tag	Data type	Description
OB10x_EV_CLASS	BYTE	Event class and identifiers: B#16#13: Active
OB10x_STRTUP	BYTE	Startup request: • B#16#81: Manual warm restart request • B#16#82: Automatic warm restart request • B#16#83: Request for manual hot restart • B#16#84: Request for automatic hot restart • B#16#85: Request for manual cold restart • B#16#86: Request for automatic cold restart • B#16#87: Master: Request for manual cold restart • B#16#88: Master: Request for automatic cold restart • B#16#8A: Master: Request for manual warm restart • B#16#8B: Master: Request for automatic warm restart
OB10x_PRIORITY	BYTE	Priority class: 27 or 1 (for S7-1500 CPUs)
OB10x_OB_NUMBR	BYTE	OB number (100, 101, or 102)
OB10x_RESERVED_1	BYTE	Reserved
OB10x_RESERVED_2	BYTE	Reserved
OB10x_STOP	WORD	Number of the event that caused the CPU to stop
OB10x_STRT_INFO	DWORD	Supplementary information about the current startup
OB10x_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

The following table shows the assignment of the tags OB10x_STR_INFO.

Bit no.	Meaning	Possible binary values	Explanation
31-24	Startup information	0000 xxxx	Rack number 0 (H CPUs only)
		0100 xxxx	Rack number 1 (H-CPUs only)
		1000 xxxx	Rack number 2 (H-CPUs only)
		0001 xxxx	Multicomputing (S7-400 only)
		0010 xxxx	Operation of more than one CPU in the segmented rack (S7-400 only)

Bit no.	Meaning	Possible binary values	Explanation
		xxxx xxx0	No difference between expected and actual configuration (S7-300 only)
		xxxx xxx1	Difference between expected and actual configuration (S7-300 only)
		xxxx xx0x	No difference between expected and actual configuration
		xxxx xx1x	Difference between expected and actual configuration
		xxxx x0xx	Not an H CPU
		xxxx x1xx	H CPU
		xxxx 0xxx	Clock for time stamp not battery-backed at last POWER ON
		xxxx 1xxx	Clock for time stamp battery-backed at last POWER ON
23-16	Startup type just performed	0000 0001	Warm restart for multicomputing without operator input on the CPU according to parameter assignment (S7-400 only)
		0000 0011	Warm restart triggered by mode selector
		0000 0100	Warm restart triggered by MPI operator input
		0000 0101	Cold restart in multicomputing without operator input on the CPU according to parameter assignment (S7-400 only)
		0000 0111	Cold restart triggered with mode selector
		0000 1000	Cold restart triggered by MPI operator input
		0000 1010	Hot restart in multicomputing without operator input on the CPU according to parameter assignment (S7-400 only)
		0000 1011	Hot restart triggered with mode selector (S7-400 only)
		0000 1100	Hot restart triggered by MPI operator input (S7-400 only)
		0001 0000	Automatic warm restart after battery-backed POWER ON
		0001 0001	Cold restart after battery-backed POWER ON according to parameter assignment
		0001 0011	Warm restart triggered with mode selector; last POWER ON battery-backed
		0001 0100	Warm restart triggered by MPI operator input; last POWER ON battery-backed
		0010 0000	Automatic warm restart after non-battery-backed POWER ON (with memory reset by system)
		0010 0001	Cold restart after non-battery-backed POWER ON (with memory reset by system)
		0010 0011	Warm restart triggered with mode selector; last POWER ON not battery-backed

Bit no.	Meaning	Possible binary values	Explanation
		0010 0100	Warm restart triggered by MPI operator input; last POWER ON not battery-backed
		1010 0000	Automatic hot restart after battery-backed POWER ON according to parameter assignment (S7-400 only)
15-12	Permissibility of automatic startup	0000	Automatic startup illegal, memory reset requested
		0001	Automatic startup illegal, parameter modifications, etc. necessary
		0111	Automatic warm restart permitted
		1111	Automatic restart (warm/hot) permitted (S7-400 only)
11-8	Permissibility of manual startup	0000	Startup illegal, memory reset requested
		0001	Startup illegal, parameter modifications, etc. necessary
		0111	Warm restart permitted
		1111	Warm restart and hot restart permitted (S7-400 only)
7-0	Last valid intervention or setting of the automatic startup at POWER ON	0000 0000	No startup
		0000 0001	Warm restart in multicomputing without operator input on the CPU according to parameter assignment (S7-400 only)
		0000 0011	Warm restart triggered by mode selector
		0000 0100	Warm restart triggered by MPI operator input
		0000 0101	Cold restart in multicomputing without operator input on the CPU according to parameter assignment (S7-400 only)
		0000 0111	Cold restart triggered with mode selector
		0000 1000	Cold restart triggered by MPI operator input
		0000 1010	Hot restart in multicomputing without operator input on the CPU according to parameter assignment (S7-400 only)
		0000 1011	Hot restart triggered with mode selector (S7-400 only)
		0000 1100	Hot restart triggered by MPI operator input (S7-400 only)
		0001 0000	Automatic warm restart after battery-backed POWER ON
		0001 0001	Cold restart after battery-backed POWER ON according to parameter assignment
		0001 0011	Warm restart triggered with mode selector; last POWER ON battery-backed
		0001 0100	Warm restart triggered by MPI operator input; last POWER ON battery-backed
		0010 0000	Automatic warm restart after non-battery-backed POWER ON (with memory reset by system)
		0010 0001	Cold restart after non-battery-backed POWER ON (with memory reset by system)

Bit no.	Meaning	Possible binary values	Explanation
		0010 0011	Warm restart triggered with mode selector; last POWER ON not battery-backed
		0010 0100	Warm restart triggered by MPI operator input; last POWER ON not battery-backed
		1010 0000	Automatic hot restart after battery-backed POWER ON according to parameter assignment (S7-400 only)

See also

Principles of the STARTUP mode (Page 1641)

Programming error organization block (OB 121)

Description

The CPU operating system calls OB 121 whenever an event occurs that is caused by an error related to the processing of the program. For example, if your program calls a block that has not been loaded on the CPU, OB 121 is called.

Function of the programming error OB

OB 121 is executed in the same priority class as the interrupted block.

If OB 121 is not programmed, the CPU changes the operating mode from RUN to STOP.

S7 provides the following instructions for use in masking and unmasking start events of OB 121 during the execution of your program:

- The "MSK_FLT (Page 6620)" instruction masks certain error codes.
- The "DMSK_FLT (Page 6621)" instruction unmasks the error codes that were masked by the "MSK_FLT (Page 6620)" instruction.
- The "READ_ERR (Page 6622)" instruction reads the event status register.

Local data for the programming error OB

The following table lists the temporary (TEMP) tags of the programming error OB. The default names of OB 121 have been selected as the tag names.

Tag	Data type	Description
OB121_EV_CLASS	BYTE	Event class and identifiers: B#16#25
OB121_SW_FLT	BYTE	Error code (possible values: B#16#21, B#16#22, B#16#23, B#16#24, B#16#25, B#16#26, B#16#27, B#16#28, B#16#29, B#16#30, B#16#31, B#16#32, B#16#33, B#16#34, B#16#35, B#16#3A, B#16#3C, B#16#3D, B#16#3E, B#16#3F)

Tag	Data type	Description
OB121_PRIORITY	BYTE	Priority class of the OB where the error occurred For S7-1500 CPUs: priority, default value: 7
OB121_OB_NUMBR	BYTE	OB number (121)
OB121_BLK_TYPE	BYTE	Type of block where the error occurred (no valid value is entered here in case of S7-300): B#16#88: OB, B#16#8A: DB, B#16#8C: FC, B#16#8E: FB
OB121_RESERVED_1	BYTE	Reserved
OB121_FLT_REG	WORD	Source of the error (depends on error code); for example: • Register in which the conversion error occurred • Incorrect address (read/write error) • Incorrect timer/counter/block number • Incorrect memory area
OB121_BLK_NUM	WORD	Number of the block with the MC7 command that caused the error (no valid number is entered here for an S7-300)
OB121_PRG_ADDR	WORD	Relative address of the MC7 instruction that has caused the error FB) (no valid number is entered here for an S7-300)
OB121_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

The error code-dependent tags have the following meaning:

Error code	Tag	Meaning
B#16#21		BCD conversion error
	OB121_FLT_REG	Identifier for the affected tab (W#16#0000: battery 1)
B#16#22		Area length error when reading
	OB121_RESERVED_1	• Bit 7 to 4 type of access: – 0: Bit access, – 1: Byte access, – 2: Word access, – 3: DWord access • Bit 3 to 0 memory area: – 0: I/O area – 1: Process image input – 2: Process image output – 3: Bit memory – 4: Global DB – 5: Instance DB – 6: own local data – 7: Local data of the caller
B#16#23		Area length error when writing
	OB121_RESERVED_1	Same as for error code B#16#22

Error code	Tag	Meaning
B#16#28		Read access to a byte, word or double word with a pointer, the bit address of which is not 0
	OB121_RESERVED_1	Same as for error code B#16#22
B#16#29		Write access to a byte, word or double word with a pointer, the bit address of which is not 0 Faulty byte address. Refer to OB121_RESERVED_1 for the data area and the type of access
	OB121_RESERVED_1	Same as for error code B#16#22
B#16#24		Area error when reading
	OB121_FLT_REG	The low byte contains the identifier for the unauthorized area (B#16#86 own local data area)
B#16#25		Area error when writing
	OB121_FLT_REG	The low byte contains the identifier for the unauthorized area (B#16#86 own local data area)
B#16#26		Error in timer number
	OB121_FLT_REG	Unauthorized number
B#16#27		Error in counter number
	OB121_FLT_REG	Unauthorized number
B#16#30		Write access to a write-protected global DB
	OB121_FLT_REG	Unauthorized DB number
B#16#31		Write access to a write-protected instance DB
	OB121_FLT_REG	Unauthorized DB number
B#16#32		DB number error when accessing a global DB
	OB121_FLT_REG	Unauthorized DB number
B#16#33		DB number error when accessing an instance DB
	OB121_FLT_REG	Unauthorized DB number
B#16#34		Number error during FC call
	OB121_FLT_REG	FC number
B#16#35		FB number error during FC call
	OB121_FLT_REG	FB number
B#16#3A		Access to a DB that is not loaded; the DB number is located in the permitted area
	OB121_FLT_REG	DB number
B#16#3C		Access to an FC that is not loaded; the FC number lies in the permissible area
	OB121_FLT_REG	FC number
B#16#3D		Access to an instruction that is not available; the SFC number is located in the permitted area
	OB121_FLT_REG	SFC number
B#16#3E		Access to an FB that is not loaded; the FB number lies in the permissible area
	OB121_FLT_REG	FB number
B#16#3F		Access to an SFB that is not available; the SFB number lies in the permissible area
	OB121_FLT_REG	SFB number

Programming OB 121

You must add OB 121 to the blocks of your CPU. Write the program that is to be executed in OB 121 to the generated block, and load the block to the CPU as part of your user program.

You can use OB 121, for example, as follows:

- You evaluate the start information in OB 121.
- You enter the cause of the error in an alarm data block.

I/O access error organization block (OB 122)

Description

The CPU operating system calls OB 122 whenever an error occurs while accessing data on a module. For example, if the CPU detects a read error when accessing data on a signal module, the operating system calls OB 122.

Function of the I/O access error OB

OB 122 is executed in the same priority class as the interrupted block. If OB 122 is not programmed, the CPU changes the operating mode from RUN to STOP.

S7 provides the following instructions for use in masking and unmasking start events of OB 122 during the execution of your program:

- The "MSK_FLT (Page 6620)" instruction masks certain error codes.
- The "DMSK_FLT (Page 6621)" instruction unmasks the error codes that were masked by the "MSK_FLT (Page 6620)" instruction.
- The "READ_ERR (Page 6622)" instruction reads the event status register.

Local data for the I/O access error OB

The following table lists the temporary (TEMP) tags of the I/O access error OB. The default names of OB 122 have been selected as the tag names.

Tag	Data type	Description
OB122_EV_CLASS	BYTE	Event class and identifiers: B#16#29
OB122_SW_FLT	BYTE	Error code: • B#16#42: I/O access error, reading. • B#16#43: I/O access error, writing.
OB122_PRIORITY	BYTE	Priority class of the OB where the error occurred For S7-1500 CPUs: priority, default value: 7
OB122_OB_NUMBR	BYTE	OB number (122)
OB122_BLK_TYPE	BYTE	Type of block where the error occurred (B#16#88: OB, B#16#8C: FC, B#16#8E: FB) (no valid number is entered here for an S7-300)

Tag	Data type	Description
OB122_MEM_AREA	BYTE	Memory area and access type: • Bit 7 to 4: Access type – 0: Bit access – 1: Byte access – 2: Word access – 3: DWord access • Bit 3 to 0: memory area – 0: I/O area – 1: Process image input – 2: Process image output
OB122_MEM_ADDR	WORD	Memory address where the error occurred
OB122_BLK_NUM	WORD	Number of the block with the MC7 command that caused the error (no valid number is entered here for an S7-300)
OB122_PRG_ADDR	WORD	Relative address of the MC7 command that caused the error (no valid value is entered here for an S7-300)
OB122_DATE_TIME	DATE_AND_TIME	Date and time when the OB was called

Sample program for OB 122

The following sample program will provide a substitute value in the "REPL_VAL" instruction. If an input module fails, the processing of the L PEB0 instruction generates a synchronous error and starts OB 122. The load command reads in the value 0 by default. However, you can use the "REPL_VAL (Page 5858)" instruction to specify any substitute value suitable for the process. This instruction replaces the ACCU content with the specified substitute value.

```
STL                                     Description
L    B#16#42                            //Identifier for "I/O access error reading" event
L    #OB122_SW_FLT                      //Error code of OB 122
==I                                     //If the same,
SPB    Qfeh                             //then jump to Qfeh
L    B#16#43                            //Identifier for "I/O access error writing" event
<>I                                     //If not the same as the error code in OB 122,
SPB    STOP                             //then jump to STOP
Qfeh:
CALL    "REPL_VAL"                      //Call the "REPL_VAL" instruction
    VAL:= DW#16#2912                    //Substitute value that is loaded in ACCU 1
    RET_VAL:= #error                    //Save the return value in #error
L    #error                             //Return value of the "REPL_VAL" instruction
L    0
==I                                     //If the same as 0 (no error has occurred),
BEC                                     //then end the block
STOP:
CALL    "STP"                           //Call the "STP" instruction: The CPU switches to STOP mode.
```

Events

Events and event ID

Event

All events are numbered within the SIMATIC S7. This allows you to relate a message text to an event.

Event ID

An event ID is assigned to every event. The event ID is structured as follows:

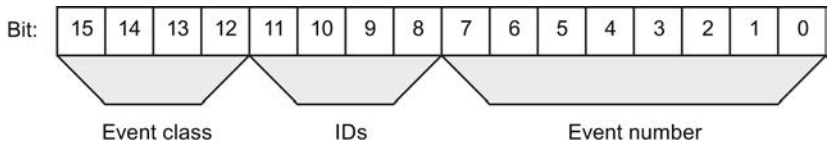


Figure 8-53 Structure of the event ID.

Event class

The event classes are as follows:

Number	Event class
1	Standard OB events
2	Synchronous errors
3	Asynchronous errors
4	Mode transitions
5	Runtime events
6	Communication events
7	Events for fail-safe and fault-tolerant systems
8	Standardized diagnostics data on modules
9	Predefined user events
A, B	Freely definable events
C , D, E	Reserved
F	Reserved for modules not in central rack (for example, CPs or FMs)

Identifier

The identifier is used to distinguish the type of events. The four bits have the following significance:

Bit no. in the event ID	Meaning
8	= 0: outgoing event = 1 incoming event
9	= 1: Entry in diagnostics buffer
10	= 1: internal error
11	= 1: external error

Event class 1 - standard OB events

Description of the event IDs

Event ID	Event
W#16#1381	Manual warm restart request
W#16#1382	Automatic warm restart request
W#16#1383	Request for manual hot restart
W#16#1384	Request for automatic hot restart
W#16#1385	Request for manual cold restart
W#16#1386	Request for automatic cold restart
W#16#1387	Master CPU: Request for manual cold restart
W#16#1388	Master CPU: Request for automatic cold restart
W#16#138A	Master CPU: Manual warm restart request
W#16#138B	Master CPU: Automatic warm restart request
W#16#138C	Standby CPU: Request for manual restart
W#16#138D	Standby CPU: Request for automatic restart

Event class 2 - synchronous errors

Description of the event IDs

Event ID	Event	OB
W#16#2521	BCD conversion error	OB 121
W#16#2522	Area length error when reading	OB 121
W#16#2523	Area length error when writing	OB 121
W#16#2524	Area error when reading	OB 121
W#16#2525	Area error when writing	OB 121
W#16#2526	Timer number error	OB 121

Event ID	Event	OB
W#16#2527	Counter number error	OB 121
W#16#2528	Alignment error when reading	OB 121
W#16#2529	Alignment error when writing	OB 121
W#16#2530	Write error when accessing the DB	OB 121
W#16#2531	Write error when accessing the DI	OB 121
W#16#2532	Block number error when opening a DB	OB 121
W#16#2533	Block number error when opening a DI	OB 121
W#16#2534	Block number error when calling an FC	OB 121
W#16#2535	Block number error when calling an FB	OB 121
W#16#253A	DB not loaded	OB 121
W#16#253C	FC not loaded	OB 121
W#16#253D	SFC not loaded	OB 121
W#16#253E	FB not loaded	OB 121
W#16#253F	SFB not loaded	OB 121
W#16#2942	I/O access error, reading	OB 122
W#16#2943	I/O access error, writing	OB 122

Event class 3 - synchronous errors

Description of the event IDs

Event ID	Event	OB
W#16#3501	Cycle time exceeded	OB 80
W#16#3502	User interface (OB or FRB) request error	OB 80
W#16#3505	Time-of-day interrupt(s) skipped due to new clock setting	OB 80
W#16#3506	Time-of-day interrupt(s) skipped when changing to RUN after HOLD	OB 80
W#16#3507	Multiple OB request errors caused internal buffer overflow	OB 80
W#16#3508	Isochronous mode interrupt - time error	OB 80
W#16#3509	Loss of interrupt due to high interrupt load	OB 80
W#16#350A	Resume RUN after CiR	OB 80
W#16#350B	Technology synchronization interrupt - time error	OB 80
W#16#3921/3821	BATTF: failure on at least one backup battery of the central rack/ problem eliminated Note: the incoming event only occurs if one of the backup battery fails (if there are redundant backup batteries). If the other battery then fails, the event does not occur again.	OB 81
W#16#3922/3822	BAF: failure of backup voltage on central rack/ problem eliminated	OB 81

Event ID	Event	OB
W#16#3923/3823	24 volt supply failure on central rack / problem eliminated	OB 81
W#16#3925/3825	BATTF: failure on at least one backup battery in a redundant central rack/ problem eliminated	OB 81
W#16#3926/3826	BAF: failure of backup voltage on redundant central rack/ problem eliminated	OB 81
W#16#3927/3827	24 volt supply failure on redundant central rack / problem eliminated	OB 81
W#16#3931/3831	BATTF: failure of at least one backup battery of the expansion rack/ problem eliminated	OB 81
W#16#3932/3832	BAF: failure of backup voltage on expansion rack/ problem eliminated	OB 81
W#16#3933/3833	24 volt supply failure on at least one expansion rack/ problem eliminated	OB 81
W#16#3942	Diagnostic error interrupt (module/sub-module fault or maintenance required or both)	OB 82
W#16#3842	Diagnostic error interrupt (module/sub-module OK)	OB 82
W#16#3951	PROFINET IO module removed	OB 83
W#16#3954	PROFINET IO interface sub-module/sub-module removed	OB 83
W#16#3854	PROFINET IO interface sub-module/sub-module and matches the configured sub-module/sub-module	OB 83
W#16#3855	PROFINET IO interface sub-module/sub-module inserted, but does not match the configured sub-module/ sub-module	OB 83
W#16#3856	PROFINET IO interface sub-module/sub-module inserted, but error in module parameter assignment	OB 83
W#16#3857	PROFINET IO submodule/module inserted, but with problem or maintenance	OB 83
W#16#3858	PROFINET IO submodule, access error corrected	OB 83
W#16#3861	Module/interface module inserted, module type OK	OB 83
W#16#3961	Module/interface module removed, cannot be addressed	OB 83
W#16#3863	Module/interface module inserted, but wrong module type	OB 83
W#16#3864	Module/interface module inserted, but causing problem (type ID unreadable)	OB 83
W#16#3865	Module inserted, but error in module parameter assignment	OB 83
W#16#3866	Module responds again, load voltage error corrected	OB 83
W#16#3966	Module not responding, load voltage error	OB 83
W#16#3367	Start module parameter reassignment	OB 83
W#16#3267	End of module parameter reassignment	OB 83

Event ID	Event	OB
W#16#3968	Module parameter reassignment terminated with error	OB 83
W#16#3571	Nesting depth of nesting levels exceeded	OB 88
W#16#3572	Nesting depth of Master Control Relay exceeded	OB 88
W#16#3573	Nesting depth exceeded for synchronous errors	OB 88
W#16#3574	Nesting depth of block calls (U stack) exceeded	OB 88
W#16#3575	Nesting depth of block calls (B stack) exceeded	OB 88
W#16#3576	Error during allocation of local data	OB 88
W#16#3578	Unknown instruction	OB 88
W#16#357A	Call statement with destination outside of the block	OB 88
W#16#3981	Interface error, incoming	-
W#16#3881	Interface error, outgoing	-
W#16#3582	Memory error in operating system detected and corrected	OB 84
W#16#3583	Accumulation of detected and corrected memory errors	OB 84
W#16#3585	Error in the PC operating system (only with Win AC controllers)	OB 84
W#16#3986	Performance of an H-Sync link affected	OB 84
W#16#3587	Multi-bit memory error detected and corrected	OB 84
W#16#35A1	User interface (OB or FRB) not found	OB 85
W#16#35A2	OB not loaded (started by SFC or operating system due to configuration)	OB 85
W#16#35A3	Error when operating system accesses a block	OB 85
W#16#35A4	PROFINET Interface DB cannot be addressed	OB 85
W#16#34A4	PROFINET Interface DB can be addressed again	OB 85
W#16#39B1	I/O access error when updating the process image of the inputs	OB 85
W#16#39B2	I/O access error when transferring the process image to the output modules	OB 85
W#16#39B3/38B3	I/O access error when updating the process image of the inputs	OB 85
W#16#39B4/38B4	I/O access error when updating the process image of the outputs	OB 85
W#16#38C1	Expansion rack operational again (1 to 21), outgoing	OB 86
W#16#39C1	Expansion rack failure (1 to 21), incoming	OB 86
W#16#38C2	Expansion rack operational again but mismatch between setpoint and actual configuration	OB 86
W#16#39C3	Distributed I/O: master system failure incoming	OB 86
W#16#39C4	Distributed I/O: Station failure, incoming	OB 86
W#16#38C4	Distributed I/O: Station failure, outgoing	OB 86
W#16#38C5	Distributed I/O: Station return with problem	OB 86
W#16#38C6	Expansion unit return, but error in module parameter assignment	OB 86

Event ID	Event	OB
W#16#38C7	DP: Station return, but error in module parameter assignment	OB 86
W#16#38C8	DP: Station return, but difference between configured and actual configuration	OB 86
W#16#39CA	PROFINET IO system failure	OB 86
W#16#39CB	PROFINET IO station failure	OB 86
W#16#38CB	PROFINET IO station return	OB 86
W#16#38CC	PROFINET IO station return with problem or maintenance	OB 86
W#16#39CD	PROFINET IO station return, but difference between configured and actual configuration	OB 86
W#16#39CE	PROFINET IO station return, error in module parameter assignment	OB 86
W#16#33CF/32CF	Failure/return of a PROFINET IO device resulting from deactivation/activation with the D_ACT_DP instruction	OB 86
W#16#39F8/38F8	Failure/return of some of the submodules of a PROFINET I-device	OB 86
W#16#38F9	Return of some of the submodules of a PROFINET I-device with a device configuration difference	OB 86
W#16#35D2	Diagnostics entries cannot be sent at present	OB 87
W#16#35D3	Synchronization frames cannot be sent	OB 87
W#16#35D4	Illegal time jump resulting from synchronization	OB 87
W#16#35D5	Error adopting the synchronization time	OB 87
W#16#35E1	Incorrect message frame ID in GD	OB 87
W#16#35E2	GD packet status cannot be entered in DB	OB 87
W#16#35E3	Frame length error in GD	OB 87
W#16#35E4	Illegal GD packet number received.	OB 87
W#16#35E5	Error while accessing DB in communication instructions for configured S7 connections	OB 87
W#16#35E6	GD total status cannot be entered in DB	OB 87

Event class 4 - stop events and other mode changes

Description of the event IDs

Event ID	Event
W#16#4300	Backed-up power on
W#16#4301	Mode transition from STOP to STARTUP
W#16#4302	Mode transition from STARTUP to RUN
W#16#4303	STOP caused by stop switch being activated
W#16#4304	STOP caused by STOP operation on programming device or because of STOP instruction
W#16#4305	HOLD: breakpoint reached

Event ID	Event
W#16#4306	HOLD: breakpoint exited
W#16#4307	Memory reset started by PG operation
W#16#4308	Memory reset started by switch setting
W#16#4309	Memory reset started automatically (power on not backed up)
W#16#430A	HOLD exited, transition to STOP
W#16#430D	STOP caused by other CPU in multicomputing
W#16#430E	Memory reset executed
W#16#430F	STOP on the module due to STOP on a CPU
W#16#4510	STOP violation of the CPU's data range
W#16#4318	Start of CiR
W#16#4319	CiR completed
W#16#4520	DEFECT: STOP not possible
W#16#4521	DEFECT: failure of instruction processing processor
W#16#4522	DEFECT: failure of clock chip
W#16#4523	DEFECT: failure of clock pulse generator
W#16#4524	DEFECT: failure of timer update function
W#16#4525	DEFECT: failure of multicomputing synchronization
W#16#4926	DEFECT: failure of the watchdog for I/O access
W#16#4527	DEFECT: failure of I/O access monitoring
W#16#4528	DEFECT: failure of scan time monitoring
W#16#4530	DEFECT: memory test error in internal memory
W#16#4931	STOP or DEFECT: memory test error in memory sub-module
W#16#4532	DEFECT: failure of core resources
W#16#4933	Checksum error
W#16#4934	DEFECT: memory not available
W#16#4935	DEFECT: cancelled by watchdog/processor exceptions
W#16#4536	DEFECT: mode switch defective
W#16#4540	STOP: memory expansion of the internal work memory is not seamless. First memory expansion too small or missing.
W#16#4541	STOP caused by priority class sequence control system
W#16#4542	STOP caused by object management system
W#16#4543	STOP caused by test functions
W#16#4544	STOP caused by diagnostics system
W#16#4545	STOP caused by communication system
W#16#4546	STOP caused by CPU memory management
W#16#4547	STOP caused by process image management
W#16#4548	STOP caused by I/O management
W#16#4949	STOP caused by continuous hardware interrupt
W#16#454A	STOP caused by configuration: An OB deselected per configuration was loaded into the CPU during STARTUP
W#16#494D	STOP caused by I/O error
W#16#494E	STOP caused by power failure
W#16#494F	STOP caused by configuration error

Event ID	Event
W#16#4550	DEFECT: internal system error
W#16#4555	No hot restart possible, monitoring time elapsed
W#16#4556	STOP: memory reset request from communication system / due to data inconsistency
W#16#4357	Module watchdog started
W#16#4358	All modules are ready for operation
W#16#4959	One or more modules not ready for operation
W#16#4562	STOP caused by programming error (OB not loaded or not possible)
W#16#4563	STOP caused by I/O access error (OB not loaded or not possible)
W#16#4567	STOP caused by H-event
W#16#4568	STOP caused by time error (OB not loaded or not possible)
W#16#456A	STOP caused by diagnostic error interrupt (OB not loaded or not possible)
W#16#456B	STOP caused by removing/inserting module (OB not loaded or not possible)
W#16#456C	STOP caused by CPU hardware error (OB not loaded or not possible, or no FRB)
W#16#456D	STOP caused by program sequence error (OB not loaded or not possible)
W#16#456E	STOP caused by communication error (OB not loaded or not possible)
W#16#456F	STOP caused by rack failure OB (OB not loaded or not possible)
W#16#4570	STOP caused by hardware interrupt (OB not loaded or not possible)
W#16#4571	STOP caused by nesting stack error
W#16#4572	STOP caused by master control relay stack error
W#16#4573	STOP caused by exceeding the nesting depth for synchronous errors
W#16#4574	STOP caused by exceeding interrupt stack nesting depth in the priority class stack
W#16#4575	STOP caused by exceeding block stack nesting depth in the priority class stack
W#16#4576	STOP caused by error when allocating the local data
W#16#4578	STOP caused by unknown opcode
W#16#457A	STOP caused by code length error
W#16#457B	STOP caused by DB not being loaded on on-board I/O
W#16#497C	STOP caused by integrated technology
W#16#457D	Reset/clear request because the version of the internal interface to the integrated technology was changed.
W#16#457F	STOP caused by STOP command
W#16#4580	STOP: back-up buffer contents inconsistent (no transition to RUN)
W#16#4590	STOP caused by overloading the internal functions
W#16#49A0	STOP caused by parameter assignment error or non-permitted difference between configured and actual configuration: Start-up disabled.
W#16#49A1	STOP caused by parameter assignment error: memory reset request
W#16#49A2	STOP caused by error in parameter modification: Start-up disabled.
W#16#49A3	STOP caused by error in parameter modification: memory reset request

Event ID	Event
W#16#49A4	STOP: inconsistency in configuration data
W#16#49A5	STOP: Distributed I/O: inconsistency in the loaded configuration information
W#16#49A6	STOP: Distributed I/O: invalid configuration information
W#16#49A7	STOP: Distributed I/O: no configuration information
W#16#49A8	STOP: error indicated by the interface module for the distributed I/O
W#16#43B0	Firmware update / backup was successful
W#16#49B1	Firmware update data incorrect
W#16#49B2	Firmware update: hardware version does not match firmware
W#16#49B3	Firmware update: module type does not match firmware
W#16#43B4	Error in firmware backup
W#16#43B6	Firmware updates canceled by redundant modules
W#16#43D0	LINK-UP aborted due to violation of coordination rules
W#16#43D1	LINK-UP/UPDATE sequence aborted
W#16#49D2	Standby CPU changed to STOP due to STOP on the master CPU during link-up
W#16#43D3	STOP on standby CPU
W#16#49D4	STOP on a master, since partner CPU is also a master (link-up error)
W#16#43D5	LINK-UP rejected due to mismatched CPU memory configuration of the sub-PLC
W#16#43D6	LINK-UP rejected due to mismatched system program of the sub-PLC
W#16#43D7	LINK-UP rejected due to change in user program or in configuration
W#16#49D8	STOP/trouble shooting mode/DEFECT: hardware fault detected due to other error
W#16#49D9	STOP due to SYNC module error
W#16#49DA	STOP due to synchronization error between H-CPUs
W#16#43DC	Abort during link-up with switchover
W#16#43DD	LINK-UP rejected due to running test or other online functions
W#16#43DE	Updating aborted due to monitoring time being exceeded during the n-th attempt, new update attempt initiated
W#16#43DF	Updating aborted for the last time due to monitoring time being exceeded after completing the maximum amount of attempts. User intervention required.
W#16#43E0	Change from solo mode after link-up
W#16#43E1	Change from link-up after updating
W#16#43E2	Change from updating to redundant mode
W#16#43E3	Master CPU: change from redundant mode to solo mode
W#16#43E4	Standby CPU: change from redundant mode after error-search mode
W#16#43E5	Standby CPU: change from error-search mode after link-up or STOP
W#16#43E6	Link-up aborted on the standby CPU
W#16#43E7	Updating aborted on the standby CPU
W#16#43E8	Standby CPU: change from link-up after startup
W#16#43E9	Standby CPU: change from startup after updating
W#16#43F1	Reserve-master switchover
W#16#43F2	Coupling of incompatible H-CPUs blocked by system program
W#16#42F3	Checksum error detected and corrected by the operating system
W#16#43F4	Standby CPU: Connecting/updating disabled by means of the H_CTRL instruction in the master CPU

Event class 5 - status runtime events

Description of the event IDs

Event ID	Event
W#16#530D	New startup information in STOP mode
W#16#510F	A problem has occurred with WinLC. This problem caused the CPU to go into STOP mode or has caused a fault in the CPU.
W#16#5311	Startup despite missing completed message from module(s)
W#16#5545	Start of parameter reassignment as part of a plant modification during operation
W#16#5445	End of parameter reassignment as part of a plant modification during operation
W#16#5961	Parameter assignment error
W#16#5962	Parameter assignment error with startup blocked
W#16#5963	Parameter assignment error with memory reset request
W#16#5966	Parameter assignment error when switching
W#16#5967	Parameter assignment error: Unknown configuration data
W#16#5968/5868	Emergency address of an Ethernet interface activated/deactivated
W#16#5969	Parameter assignment error with startup blocked
W#16#596A	PROFINET IO: IP address of an IO device already present
W#16#596B	IP address of an existing Ethernet interface
W#16#596C	Name of an existing Ethernet interface
W#16#596D	The existing network configuration does not mach the system requirements or configuration.
W#16#5371	Distributed I/O: end of the synchronization with a DP master
W#16#5979/5879	Diagnostics alarm from DP interface: EXTF LED on/off
W#16#597C	DP Global Control command failed or moved
W#16#5380	Diagnostics buffer entries from interrupt and asynchronous error events disabled
W#16#5581	One or more licenses for runtime software are missing.
W#16#5481	All licenses for runtime software are complete again.
W#16#558A	Difference between the Machine-Readable Product Code of the configured and inserted CPU
W#16#558B	Difference between the firmware version of the configured and inserted CPU
W#16#5966	Parameter assignment error when switching
W#16#597C	DP Global Control command failed or moved
W#16#5395	Distributed I/O: Reset of a DP master
W#16#5598	Start of potential inconsistency with DP master systems due to CiR
W#16#5498	End of potential inconsistency with DP master systems due to CiR
W#16#59A0	Interrupt in the CPU cannot be assigned
W#16#59A1	Configuration error in the integrated technology
W#16#53A2	Download of technology firmware successful
W#16#59A3	Error when downloading the integrated technology
W#16#53A4	Download of technology DB not successful

Event ID	Event
W#16#55A5	Version conflict: internal interface with integrated technology
W#16#55A6	The maximum number of technology objects has been exceeded.
W#16#55A7	A technology DB of this type is already present.
W#16#53FF	Reset to factory setting

Event class 6 - communication events

Description of the event IDs

Event ID	Event
W#16#6500	Connection ID exists twice on module
W#16#6501	Connection resources inadequate
W#16#6502	Error in the connection description
W#16#6905/6805	Resource problem with permanently configured connections/corrected
W#16#6510	CFB structure error detected in instance DB when evaluating EPROM
W#16#6514	GD packet number exists twice on the module
W#16#6515	Inconsistent length specifications in GD configuration information
W#16#6316	Interface error at power up of the AS
W#16#6521	No memory sub-module and no internal memory available
W#16#6522	Illegal module: replace sub-module and reset memory
W#16#6523	Memory reset request due to error accessing module
W#16#6524	Memory reset request due to error in block header
W#16#6526	Memory reset request due to memory replacement
W#16#6527	Memory replaced, therefore hot restart not possible
W#16#6528	Object handling function in STOP/HOLD mode, no hot restart possible
W#16#6529	No startup possible during the "load user program" function
W#16#652A	No startup because block exists twice in user memory
W#16#652B	No startup because block is too long for sub-module - replace sub-module Module replacement necessary
W#16#652C	No startup due to illegal OB on sub-module
W#16#6532	No startup because illegal configuration information on sub-module
W#16#6533	Memory reset request because of invalid sub-module content
W#16#6534	No startup: block exists more than once on module
W#16#6535	No startup: not enough memory to transfer block from module
W#16#6536	No startup: module contains an illegal block number
W#16#6537	No startup: module contains a block with an illegal length
W#16#6538	Local data or write-protection ID (for DB) of a block illegal for CPU
W#16#6539	Illegal command in block (detected by compiler)
W#16#653A	Memory reset request because local OB data on module too short
W#16#6543	No startup: illegal block type
W#16#6544	No startup: attribute "relevant for processing" illegal

Event ID	Event
W#16#6545	Source language illegal
W#16#6546	Maximum number of blocks of one block type reached
W#16#6547	Parameter assignment error when assigning parameters to modules (not on I/O bus, cancel download)
W#16#6548	Plausibility error during block check
W#16#6549	Structure error in block
W#16#6550	A block has an error in the CRC
W#16#6551	A block has no CRC
W#16#6353	Firmware update: Start of firmware download over the network
W#16#6253	Firmware update: End of firmware download over the network
W#16#6560	SCAN overflow
W#16#6981	Interface error, incoming
W#16#6881	Interface error, outgoing
W#16#6390	Formatting of a Micro Memory Card executed

Event class 8 - diagnostic events for modules

Description of the event IDs

Event ID	Event	Module type
W#16#8x00	Module fault/OK	Any
W#16#8x01	Internal error	
W#16#8x02	External error	
W#16#8x03	Channel fault	
W#16#8x04	External auxiliary voltage missing	
W#16#8x05	Front panel connector not plugged in	
W#16#8x06	No parameter assignment	
W#16#8x07	Incorrect parameters on module	
W#16#8x30	User module is missing or has an error	
W#16#8x31	Communication problem	
W#16#8x32	RUN/STOP mode (STOP: incoming, RUN: outgoing)	
W#16#8x33	Time monitoring responded (watchdog)	
W#16#8x34	Internal power supply failed	
W#16#8x35	BATTF: battery exhausted	
W#16#8x36	Entire backup failed	
W#16#8x40	Expansion rack failure	
W#16#8x41	Processor failure	
W#16#8x42	EPROM fault	
W#16#8x43	RAM fault	
W#16#8x44	ADC/DAC error	
W#16#8x45	Fuse tripped	

Event ID	Event	Module type
W#16#8x46	Hardware interrupt lost	
W#16#8x50	Configuration/parameter assignment error	Analog input
W#16#8x51	Common mode error	
W#16#8x52	P short circuit	
W#16#8x53	M short circuit	
W#16#8x54	Wire break	
W#16#8x55	Reference channel error	
W#16#8x56	Measuring range undershoot	
W#16#8x57	Measuring range exceeded	
W#16#8x60	Configuration/parameter assignment error	Analog output
W#16#8x61	Common mode error	
W#16#8x62	P short circuit	
W#16#8x63	M short circuit	
W#16#8x64	Wire break	
W#16#8x66	No load voltage	
W#16#8x70	Configuration/parameter assignment error	Digital input
W#16#8x71	Ground fault	
W#16#8x72	P short circuit (sensor)	
W#16#8x73	M short circuit	
W#16#8x74	Wire break	
W#16#8x75	No sensor power supply	
W#16#8x80	Configuration/parameter assignment error	Digital output
W#16#8x81	Ground fault	
W#16#8x82	P short circuit	
W#16#8x83	M short circuit	
W#16#8x84	Wire break	
W#16#8x85	Fuse tripped	
W#16#8x86	No load voltage	
W#16#8x87	excessive temperature	
W#16#8xB0	FM-count: Signal A defective	FM
W#16#8xB1	FM-count: Signal B defective	
W#16#8xB2	FM-count: Signal N defective	
W#16#8xB3	FM-count: Transfer defective value between the channels	
W#16#8xB4	FM-count: Sensor supply 5.2 V defective	
W#16#8xB5	FM-count: Sensor supply 24 V defective	

Event class 9 - standard user events

Description of the event IDs

Event ID	Event
W#16#9001	Automatic mode
W#16#9101	Manual mode
W#16#9x02	OPEN/CLOSED, ON/OFF
W#16#9x03	Manual command enable
W#16#9x04	Unit protective command (OPEN/CLOSED)
W#16#9x05	Process enable
W#16#9x06	System protection command
W#16#9x07	Process value monitoring responded
W#16#9x08	Manipulated variable monitoring responded
W#16#9x09	System deviation greater than permitted
W#16#9x0A	Limit position error
W#16#9x0B	Runtime error
W#16#9x0C	Command execution error (sequence control)
W#16#9x0D	Operating status running > OPEN
W#16#9x0E	Operating status running > CLOSED
W#16#9x0F	Command blocking
W#16#9x11	Process status OPEN/ON
W#16#9x12	Process status CLOSED/OFF
W#16#9x13	Process status intermediate position
W#16#9x14	Process status ON via AUTO
W#16#9x15	Process status ON via manual
W#16#9x16	Process status ON via protective command
W#16#9x17	Process status OFF via AUTO
W#16#9x18	Process status OFF via manual
W#16#9x19	Process status OFF via protective command
W#16#9x21	Function error on approach
W#16#9x22	Function error on return
W#16#9x31	Actuator (DE/WE) limit position OPEN
W#16#9x32	Actuator (DE/WE) limit position not OPEN
W#16#9x33	Actuator (DE/WE) limit position CLOSED
W#16#9x34	Actuator (DE/WE) limit position not CLOSED
W#16#9x41	Illegal status, tolerance time elapsed
W#16#9x42	Illegal status, tolerance time not elapsed
W#16#9x43	Interlock error, tolerance time = 0
W#16#9x44	Interlock error, tolerance time > 0
W#16#9x45	No reaction
W#16#9x46	Final status exited illegally, tolerance time = 0
W#16#9x47	Final status exited illegally, tolerance time > 0

Event ID	Event
W#16#9x50	Upper limit of signal range USR
W#16#9x51	Upper limit of measuring range UMR
W#16#9x52	Lower limit of signal range LSR
W#16#9x53	Lower limit of measuring range LMR
W#16#9x54	Upper alarm limit UAL
W#16#9x55	Upper warning limit UWL
W#16#9x56	Upper tolerance limit UTL
W#16#9x57	Lower tolerance limit LTL
W#16#9x58	Lower warning limit LWL
W#16#9x59	Lower alarm limit LAL
W#16#9x60	GRAPH 7 step entering/leaving
W#16#9x61	GRAPH 7 interlock error
W#16#9x62	GRAPH 7 execution error
W#16#9x63	GRAPH 7 error noted
W#16#9x64	GRAPH 7 error acknowledged
W#16#9x70	Trend exceeded in positive direction
W#16#9x71	Trend exceeded in negative direction
W#16#9x72	No reaction
W#16#9x73	Final state exited illegally
W#16#9x80	Limit value exceeded, tolerance time = 0
W#16#9x81	Limit value exceeded, tolerance time > 0
W#16#9x82	Below limit value, tolerance time = 0
W#16#9x83	Below limit value, tolerance time > 0
W#16#9x84	Gradient exceeded, tolerance time = 0
W#16#9x85	Gradient exceeded, tolerance time > 0
W#16#9x86	Below gradient, tolerance time = 0
W#16#9x87	Below gradient, tolerance time > 0
W#16#9190/9090	User parameter assignment error incoming/outgoing
W#16#91F0	Overflow
W#16#91F1	Underflow
W#16#91F2	Division by 0
W#16#91F3	Illegal calculation operation

Event classes A and B - free user events

Description of the event IDs

Event ID	Event
W#16#Axyz	Events available for user
W#16#Bxyz	Events available for user

Reserved event classes

Reserved

The following event classes are reserved for later expansions:

- C
- D
- E
- F reserved for modules not in central rack (e.g., CPs or FMs)

8.1.5.3 Functional description of S7-1500 CPUs

Operating modes

Basics of the operating modes of S7-CPUs

Introduction

Operating modes describe the states of the CPU. The following operating modes are possible:

- STARTUP
- RUN
- STOP

In these operating modes, the CPU can communicate via the PN/IE interface, for example.

Other operating modes

If the CPU is not ready for operation, it is in one of following two operating modes:

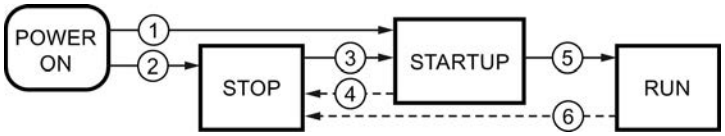
- Off-voltage, in other words, line power is switched off.
- Defective, which means that an internal error has occurred.
  If the "Defective" status is caused by a firmware error, this state is indicated by the status LEDs of the CPU (refer to the description of the CPU in the manuals). Proceed as follows to identify the cause:
  - Turn the power supply switch off and on again.
  - Read out the diagnostics buffer when the CPU starts up and send the data for analysis to Customer Support.

  If the CPU does not start up, replace it.

Operating mode transitions

Overview

The following figure shows the operating modes and the operating mode transitions of S7-1500 CPUs:



The following table shows the conditions under which the operating modes will change:

No.	Operating mode transition	Conditions
①	POWER ON → STARTUP	After switching on, the CPU goes to "STARTUP" mode if: • "Warm restart" startup type is set, and • the hardware configuration and the program blocks are consistent. Non-retentive memory is cleared and the contents of non-retentive DBs are reset to the initial values of the load memory. Retentive memory and retentive DB contents are retained.
②	POWER ON → STOP	When startup type "No startup" is set, the CPU goes to "STOP" mode after the supply voltage is switched on. Non-retentive memory is cleared and the contents of non-retentive DBs are reset to the initial values of the load memory. Retentive memory and retentive DB contents are retained.
③	STOP → STARTUP	The CPU switches to "STARTUP" mode if: • CPU is set to "RUN" from the programming device, and • the hardware configuration and the program blocks are consistent.
④	STARTUP → STOP	The CPU returns to the "STOP" mode in the following situations: • Error detected during startup. • The CPU is set to "STOP" from the programming device. • A STOP command is processed in the STARTUP OB.
⑤	STARTUP → RUN	If the STARTUP is successful, the CPU switches to "RUN".
⑥	RUN → STOP	The CPU returns to the "STOP" mode in the following situations: • An error is detected that prevents continued processing. • The CPU is set to "STOP" from the programming device. • A STOP command is processed in the user program.

"STARTUP" operating mode

Basics of "STARTUP" operating mode

Function

After turning on the CPU, it executes a startup program before starting to execute the cyclic user program.

By suitably programming startup OBs, you can specify certain initialization variables for your cyclic program in the startup program. There is no rule in terms of the number of startup OBs. That is, you can set up one or several startup OBs in your program or none at all.

Parameter settings for startup characteristics

You can specify whether the CPU remains in "STOP" mode or whether a warm restart is performed. Over and above this, you can set the response during startup ("RUN" or previous mode) in the "Startup" group of the CPU properties.

Special characteristics

Note the following points regarding the "STARTUP" mode:

- The outputs on the modules are disabled.
- The process image is initialized.
- The process image is not updated.
  - In order to read the current state from inputs during "STARTUP", you can access inputs via direct I/O access.
  - In order to initialize outputs during STARTUP, values can be written via the process image or via direct I/O access. The values are output at the outputs during the transition to the "RUN" mode.
- The non-retentive bit memories, timers and counters are initialized.
- The non-retentive tags in data blocks are initialized.
- During startup, no cycle time monitoring is running yet.
- The startup OBs are executed. Regardless of the selected type of startup, all programmed startup OBs are executed.
  The following OBs can be started during startup, if a corresponding event occurs:
  - OB 82: Diagnostic interrupt
  - OB 83: Removal/insertion of modules
  - OB 86: Rack error
  - OB 121: Programming error (only for global error handling)
  - OB 122: I/O access error (only for global error handling)

  All other OBs cannot be started until the transition into the "RUN" operating mode.

Warm restart

Function

During a warm restart, all non-retentive bit memory is deleted and non-retentive DB contents are reset to the initial values from load memory. Retentive bit memory and retentive DB contents are retained.

Program execution begins at the call of the first startup OB.

Triggering a warm restart

You can trigger a "Warm restart" using a corresponding menu command on your programming device in the following situations:

- The CPU must be in "STOP" mode.
- After a memory reset
- After downloading a consistent program and a consistent hardware configuration in the "STOP" mode of the CPU.

"POWER ON" triggers a "warm restart" if you have set the following parameters for the startup response:

- Startup type "warm restart - RUN" (regardless of the CPU operating mode prior to POWER OFF).
- "Warm restart - mode prior to POWER OFF" (depending on the CPU operating mode prior to POWER OFF. The CPU must have been in RUN mode prior to this.)

STARTUP activities

Overview

The following table shows which activities the CPU performs at STARTUP:

Activities in execution sequence	At warm restart
Clear non-retentive bit memories	yes
Clear all bit memories	no
Clear the process image output	yes
Processing startup OBs	yes
Update the process image input	yes
Enable outputs after changing to "RUN" mode	yes

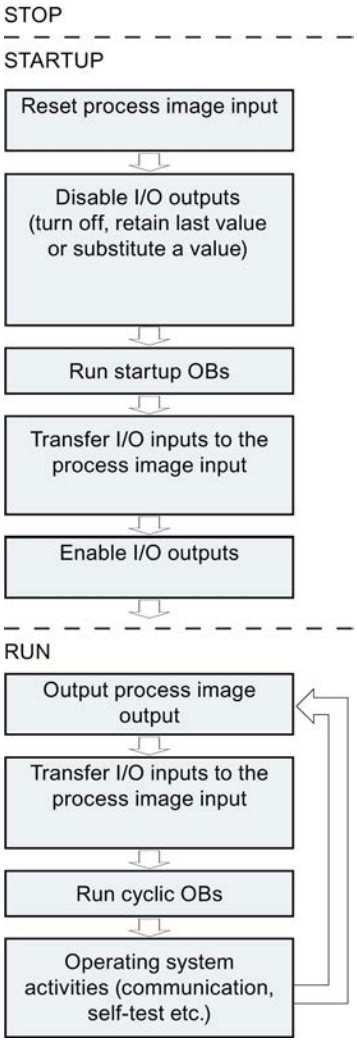
Sequence

The following figure shows the activities of the CPU in "STOP", "STARTUP", and "RUN" operating modes.

You can use the following measures to specify the state of the I/O outputs in the first cycle of the user program:

- Use assignable output modules to be able to output substitute values or to retain the last value.
- Set default values for outputs in startup OBs.

During the startup, all interrupt events are entered in a queue so that they can be processed later during RUN mode. In RUN mode, hardware interrupts can be processed at any time.



Special features during startup

Response when expected and actual configurations do not match

The expected configuration is represented by the engineering configuration loaded on the CPU. The actual configuration is the actual configuration of the automation system.

If the preset configuration and actual configuration deviate from one another, the CPU's characteristics are specified by the "Supported hardware compatibility" parameter.

Canceling a STARTUP

If errors occur during startup, the startup is canceled and the CPU remains in "STOP" mode.

Under the following conditions, a startup is not performed or canceled:

- If no SIMATIC memory card is inserted or an invalid SIMATIC memory card is inserted.
- If no hardware configuration has been downloaded.

"RUN" operating mode

Function

In "RUN" mode the cyclic, time-driven and interrupt-driven program sections execute. Addresses that are in the "Automatic Update" process image are automatically updated in each program cycle.

- The process image output is read out.
- The process image input table is read.
- The user program is executed.

Active data exchange between S7-1200 CPUs by means of Open User Communication is only possible in "RUN" mode.

Running the user program

Once the CPU has read the inputs, the cyclic program runs from the first to the last instruction.

If you have configured a minimum cycle time, the CPU does not end the cycle until this minimum cycle time is up even if the user program is completed sooner.

A maximum cycle time is set which you can adjust according to your requirements. This ensures that the cyclic program is completed within a specified time. The system responds with a time error if the cyclic program is not completed within this time.

Other events such as hardware and diagnostic interrupts can interrupt the cyclic program flow and prolong the cycle time.

"STOP" operating mode

Function

In "STOP" mode, the user program is not executed. All outputs are disabled or react according to the parameter settings: They provide a substitute value as set in the parameters or retain the last value output and bring the controlled process to a safe status.

The CPU checks the following points:

- Whether the default settings for the CPU are applicable or parameter sets are present
- Whether the general conditions for the programmed startup characteristics are correct

Basics of a memory reset

Function

A memory reset on the CPU is possible only in "STOP" mode.

When memory is reset, the CPU is changed to an "initial status". This means:

- An existing online connection between the PG/PC and the CPU is disconnected.
- The content of the work memory and the retentive and non-retentive data are deleted.
- The diagnostic buffer, time of day, and IP address are retained.
- The CPU is then initialized with the loaded project data (hardware configuration, code and data blocks, force jobs). These data are copied from the load memory to the work memory. Result:
  - If an IP address was set in the hardware configuration ("Set IP address in the project" option), this IP address is valid after the memory reset.
  - Data blocks no longer have current values but rather their configured start values.
  - Force jobs remain active

Memory areas

Things you should know about memory cards

How the memory card functions

The SIMATIC Memory Card for an S7-1500 is an SD memory card preformatted by Siemens for the CPU user program.

You may only delete files and folders. If you format the memory card with Windows, for example with a commercially available card reader, you make the memory card unusable as a storage medium for an S7 CPU.

Setting the card type

You can use the memory card as a program card or as a firmware update card.

To set the card type, insert the memory card into the card reader of the programming device and select the "Card reader/USB memory" folder from the project navigation. In the properties of the selected memory card, designate the card type:

- Program
  You use a program card as an external load memory for the CPU. It contains the entire user program for the CPU. The user program is transferred to the work memory and runs there. If the SIMATIC memory card with the user program is removed, the program is no longer available.
  Directory: SIMATIC.S7S
- Firmware card
  Firmware for the S7-1500 modules can be stored on a memory card. Therefore it is possible to perform a firmware update with the help of a specifically prepared memory card.
  Directory: FWUPDATE.S7S

Transferring objects from the project to a memory card

When the memory card is inserted in the programming device or in an external card reader, you can transfer the following objects from the project tree to the memory card:

- Individual blocks (multiple selection possible)
  In this case a consistent transfer is available, as the dependencies of the blocks to each other is taken into account with block selection.
- PLC
  In this case, all objects relevant to processing are transferred, such as blocks and the hardware configuration on the memory card, just as with downloading.

To perform the transfer, you can move the objects with drag-and-drop or use the command "Card reader/USB memory > Write to memory card" in the "Project" menu.

Transferring objects from the memory card to the project

You transfer objects by selecting the SIMATIC memory card symbol in the project tree and then dragging it to a CPU in the project. The contents of the SIMATIC memory card must correspond to the CPU type in the project tree. If blocks existed beforehand, these are deleted prior to the transfer.

Updating firmware with a memory card

You can get the latest firmware data on the Internet from the Service & Support pages:

http://support.automation.siemens.com

Save the firmware files on the hard disk and plug the SIMATIC Memory Card into the card reader of your programming device.

To store the file on the memory card, select the memory card in the "Card Reader/USB memory" folder in the project navigation. Select the shortcut menu "Card reader/USB memory > Create firmware update memory card".

Then follow the instructions for performing the firmware update in the Service & Support portal or in the S7-1500 system manual.

Updating the firmware provides the module, for example, CPU or I/O module, with a new firmware version. If you have used the module in the project, you must replace the already configured module with the module with the new firmware version offline by means of module replacement and adapt and then load the program or configuration.

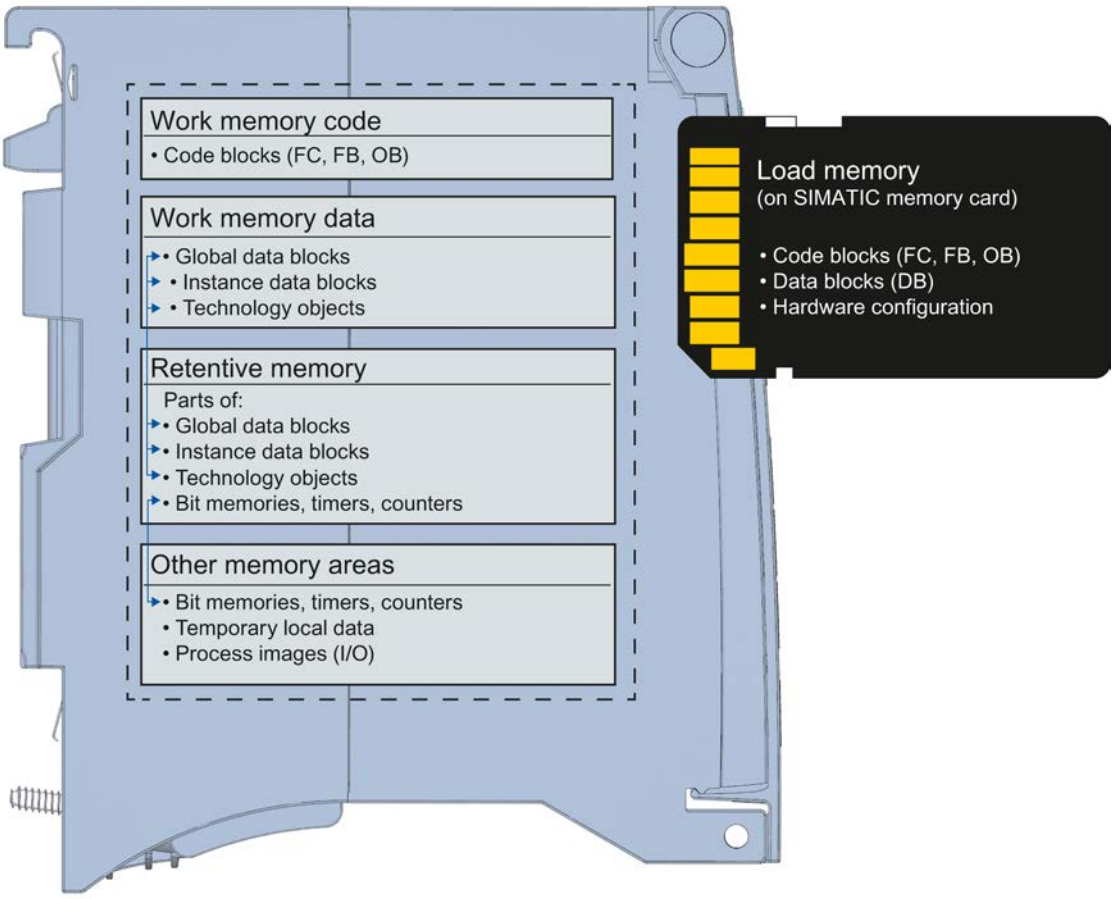
Force jobs on the memory card

Please take into account that a force job stored on the memory card is executed on every CPU into which this memory card is inserted. If this is not intended, terminate the force job before you insert the memory card into a different CPU.

CPU memory areas

Memory areas of the S7-1500 CPUs

The following figure shows the CPU memory areas and the load memory on the SIMATIC memory card. In addition to the load memory, the SIMATIC memory card may contain other data, such as recipes, Data Logs, HMI backups.



Load memory

The load memory is a non-volatile memory for program code, data blocks and hardware configuration. When these objects are loaded to the CPU, they are first stored in the load memory. This memory is located on the SIMATIC memory card. A SIMATIC memory card must be inserted in order for the CPU to operate.

Work memory

The work memory is volatile memory that contains the code and data blocks. The work memory is integrated in the CPU and cannot be extended.

The work memory in the S7-1500 is divided into two areas:

- Code work memory for runtime-relevant parts of the program code.
- Data work memory for runtime-relevant parts of data blocks and technology objects. At the operating mode transitions POWER ON → startup and STOP → startup, tags from global data blocks, instance data blocks, and technology objects are initialized with their start values. Retentive tags retain their actual values that were saved in the retentive memory.

Retentive memory

Retentive memory is a non-volatile memory for storing a limited amount of data in the event of a power failure. The tags and operand areas that have been defined as retentive are saved in the retentive memory.

This data is retained after a power-off or power failure. All other tags are lost and are set to their start values during the operating mode transitions startup after POWER ON and startup after STOP.

The contents of the retentive memory are deleted by the following actions:

- Memory reset
- Reset to factory settings

Specific tags of technology objects are also stored in the retentive memory. These tags are not deleted during memory reset.

Other memory areas

The CPU has other memory areas in addition to those described for the user program and data.

These areas include the following runtime-relevant areas:

- Bit memory, timers, counters
- Temporary local data
- Process images

The relevant CPU-specific variables can be found in the CPU's technical specifications.

Information on memory areas

You can view information offline and online via the memory areas of your S7-1500 CPU.

Offline:

During the creation or modification of a program, you can determine whether or not it is too large for a specific CPU. You can find this information, for example, under "Program information" in the project tree. In the "Resources" tab, you will find information on the overall size of the memory areas and on the memory space already allocated. At this location, you can also find information on the inputs and outputs that are already assigned.

For the S7-1500, you can specify the total size of the load memory (program information; the Load memory - total box is a drop down list for selecting the load memory size).

Online:

Start "Online & Diagnostics" in the project navigation. The "Memory" function provides information on the fill level of all the CPU's memories.

Additional information on your CPU's resources can be found in the "Assignment list" and "Call structure" tabs of the program information as well as in the PLC tag table.

See also

Overview of available program information (Page 2936)

Showing fill levels of all types of memory on a CPU (Page 2098)

Retentive memory areas

Introduction

The S7-1500 CPUs have a retentive memory for storing retentive data in the event of a POWER OFF. The size of the retentive memory is documented in the CPU's technical specifications.

The utilization of the retentive memory of the configured CPU can be found offline in the project tree under "Program information > Resources" or online in the project tree under "Online & Diagnostics > Diagnostics > Memory".

If you define data as retentive, their content is retained after a power failure, during CPU startup, and when a modified program is loaded.

The following data and objects can be defined as retentive:

- Bit memory, timers, counters
- Tags of global data blocks
- Tags of instance data blocks of a function block

Certain tags of technology objects are always retentive, for example, calibration values of absolute encoders.

Bit memory, timers, counters

In the PLC tag table, you use the "Retain" button to define the number of retentive bit memories, timers and counters.

Tags of a global data block

Depending on the access settings, you can define either individual tags or all tags of a global data block as retentive in a global data block:

- Optimized block access enabled: retentivity can be set for each individual tag.
- Optimized block access disabled: The retain setting applies to all tags of the DB; either all or none of the tags are retentive.

Instance data block (function block) tags

You can define the tags of a function block's instance data block as retentive. Depending on the access to the instance data block, you can define the following for individual tags or for all tags of the block jointly:

- Optimized block access enabled: You can define individual tags as retentive in both the interface of the function block and in the assigned instance data block.
- Optimized block access disabled: You can define retentivity for all tags together in the instance data block.

Creating a data block in the user program

The instruction "Create_DB" is used to create a data block. Depending on the selection for the ATTRIB parameter, the generated data block either has the property "retentive" or "non-retentive". In this case, it is not possible to set the retentivity for individual tags.

Technology object tags

Certain tags of technology objects are also retentive, for example, calibration values of absolute encoders. STEP 7 manages the retentivity of the technology objects automatically. Hence, it is not necessary to configure retentivity for these. These tags are not affected by a memory reset and can only be deleted by resetting to factory settings.

Summary of retentive behavior

Retentive characteristics of the memory objects

In addition to the retentive memory areas described, there are other objects with retentive characteristics, for example, the diagnostic buffer. These objects do not occupy retentive memory.

The following table shows the retentive characteristics of the memory objects:

- in the operating mode transitions STOP → Startup and POWER OFF → Startup
- in the memory-influencing functions Memory reset and Reset to factory settings

Table 8-91 Retentive characteristics of memory objects ("x" = content is retained; "-" = content is initialized)

Memory object	Operating mode transitions		Memory reset	Reset to factory settings
	STOP → STARTUP	POWER OFF → STARTUP		
Actual values of the data blocks, instance data blocks	Can be set in the properties of the DB.		-	-
Bit memories, timers and counters - configured as retentive	x	x	-	-
Bit memories, timers and counters - configured as non-retentive	-	-	-	-
Certain retentive tags from technology objects (for example, adjustment values of absolute encoders)	x		x	-
Diagnostic buffer (retentive area)	x	x	x	-
Diagnostic buffer entries (non-retentive area)	x	-	-	-
Runtime meters	x	x	x	-
Time-of-day	x	x	x	-

Diagnostics buffer

Part of the diagnostic buffer of S7-1500 CPUs is retentive. The most recent entries in the diagnostic buffer are retained after a power failure or memory reset. The retentive part of the diagnostic buffer can only be deleted by resetting to factory settings. The number of retentive diagnostic buffer entries depends on the CPU. The entries in the diagnostic buffer do not require retentive memory.

Runtime meters

The runtime meter of S7-1500 CPUs is retentive and not affected by a memory reset. After a reset to factory settings, it is set to zero.

Time-of-day

The time-of-day of S7-1500 CPUs is retentive and not affected by memory reset. After a reset to factory settings, the time-of-day is set to 01/01/2012 00:00:00.

Memory behavior when loading software changes

Introduction

In the STOP and RUN modes of the CPU, the S7-1500 CPU offers you the option of loading changes to the software without affecting the actual values of already loaded tags during the download operation.

In STEP 7, you load changes to the software under "Download to device > Software (changes only)".

Impact of software changes on PLC tags

You can load the following software changes, **without** affecting the actual values of tags that have already been loaded:

- Changing names
- Changing comments
- Adding new tags
- Deleting tags
- Changing retentivity settings for bit memories, times, and counters.

The actual values are affected by loading the following software changes:

- Changing data types
- Changing addresses

Impacts of software changes on data blocks with memory reserve

If you are using memory reserve for data blocks ("Optimized block access" attribute and "Download without reinitialization" button activated), you can load the following software changes **without** the actual values of already loaded tags being reinitialized.

- Changing start values
- Changing comments
- Adding new tags

To load the following software changes, you must deactivate the "Download without reinitialization" button. During the next load, all actual values of the data block will be reinitialized:

- Changing names
- Changing data types
- Changing retentivity settings
- Deleting tags
- Changes to the memory reserve settings

Impacts of software changes on data blocks without memory reserve

If you are not using memory reserve, you can load the following software changes, without affecting the actual values of tags that have already been loaded:

- Changing start values
- Changing comments

When loading the following software changes, all actual values of the data block will be reinitialized:

- Changing names
- Changing data types

- Changing retentivity settings
- Adding new tags
- Deleting tags

Memory reserve of data blocks

Each function block or data block with the activated attribute "Optimized block access" contains a memory reserve by default which you may use for subsequent interface changes. The memory reserve is initially not used. When you have compiled and loaded the block, and then observe that you want to reload interface changes, activate the memory reserve. All tags that you subsequently declare will be placed in the memory reserve. During the next load the new tags will be initialized to their start values; the previously loaded tags will not be reinitialized.

The setting for memory reserve can be found in the data block properties under "Download without reinitialization".

See also

Basic information on loading block extensions without reinitialization (Page 2914)

Initializing setting values in the online program (Page 2852)

General information on loading (Page 355)

I/O data areas

Access to the I/O addresses

I/O addresses

If you insert a module in the device view, its user data is located in the process image of the S7-1500-CPU (default). The CPU handles the data exchange between the module and the process image area automatically during the update of the process images.

Append the suffix ":P" to the I/O address if you want the program to access the module directly instead of using the process image.

Direct I/O access does not improve performance for S7-1500 CPUs, however. Where possible, you should therefore use process images or process image partitions to address the I/O.

Start address of a module

Definition

The start address is the lowest byte address of a module. It acts as the initial address of the module user data area.

Configuring module start addresses

The addresses used in the user program and the module start addresses are coordinated when the modules are configured.

In the module properties ("I/O addresses" group), you can change the start addresses that were assigned automatically when the modules were inserted.

You can also set whether the addresses are located in the process image or in which process image partition they are located.

In addition to the process image partition, you can also select the organization block to which the process image partition is assigned. With the exception of the isochronous interrupt OB, the associated process image partition is updated when the assigned OB is called. Immediately before the beginning of the OB start, the process image partition of the inputs is updated. Immediately after the end of the OB, the process image partition of the outputs is transferred to the modules.

Select the "Automatic update" process image if the process image in which the addresses of the module are located is to be updated automatically at the cycle control point.

Evaluating the value status

Introduction

Value status (QI, quality information) refers to diagnostic information of I/O channels that is made available to the user program via the process image inputs (PII). The value status is transferred synchronously with the user data.

Each bit of the value status is assigned to a channel and provides information on the validity of the value (0 = value is incorrect).

For example, if there is a wire break at a digital input, the user data signal is logically "0". The module also sets the associated bit in the value status to 0 due to the wire break diagnostics, which means you have the option to determine that the value is invalid by querying the value status.

The assignment of the value status bytes in the process image inputs depends on the module used. Detailed information can be found in the manual for the relevant I/O module.

Activating the value status

If you want to evaluate the value status of the channel, select the "Value status" check box in the properties of the I/O module. By activating the value status, additional addresses are assigned for the value status in the process image inputs (PIIs). For input modules, STEP 7

assigns the input addresses directly following the user data; for output modules, the next available input addresses are assigned.

Special features for modules with MSI/MSO

Requirement:

- GSD file for I/O modules with MSI/MSO
- IO device supports MSI/MSO (e.g., IM 155-5 PN ST firmware version V2.0.0 or higher)

The value status has the following meaning for modules with diagnostic capability for which you have configured a module-internal Shared Input (MSI) or module-internal Shared Output (MSO):

MSI:

For the first submodule (=basic submodule), the value status 0 indicates that the value is incorrect.

For the 2nd to 4th submodule (=MSI submodule), the value status 0 indicates that the value is incorrect or the basic submodule has not yet been configured (not ready).

MSO:

For the first submodule (=basic submodule), the value status 1 indicates that the output value specified by the user program is actually output at the module terminal.

Possible causes for value status = 0:

- Value is incorrect, for example, because the supply voltage is missing.
- IO controller of basic submodule is in STOP.

For the 2nd to 4th submodule (=MSO submodule), the value status 1 indicates that the output value specified by the user program is actually output at the module terminal.

Possible causes for value status = 0:

- Value is incorrect, for example, because the supply voltage is missing.
- IO controller of basic submodule is in STOP.
- The basic submodule parameters have not yet been assigned.

See also

Cyclic OBs (Page 1891)

Example for evaluating the value status (Page 1848)

System diagnostics (http://support.automation.siemens.com/WW/view/en/68052815)

Module-internal shared input/shared output (MSI/MSO) (http://support.automation.siemens.com/WW/view/en/68052815)

Example for evaluating the value status

The DI 32x24VDC HF module of the S7-1500 can be configured in STEP 7 (device view) for various applications. Depending on the configuration, additional/different addresses are assigned in the process image input.

This example shows various configurations with their respective assigned value status and describes the effect on the address assignment.

Configuration options of DI 32x24VDC HF

The following configurations are possible:

- 1 x 32-channel without value status
- 1 x 32-channel with value status
- 4 x 8-channel without value status
- 4 x 8-channel with value status
- 1 x 32-channel with value status for module-internal shared input (MSI) with up to 4 submodules

An additional bit for the value status is assigned to each channel in the configurations with value status. The value status bit indicates if the read in digital value is valid. (0 = value is incorrect).

Address space for configuration as 32-channel DI 32x24VDC HF with value status

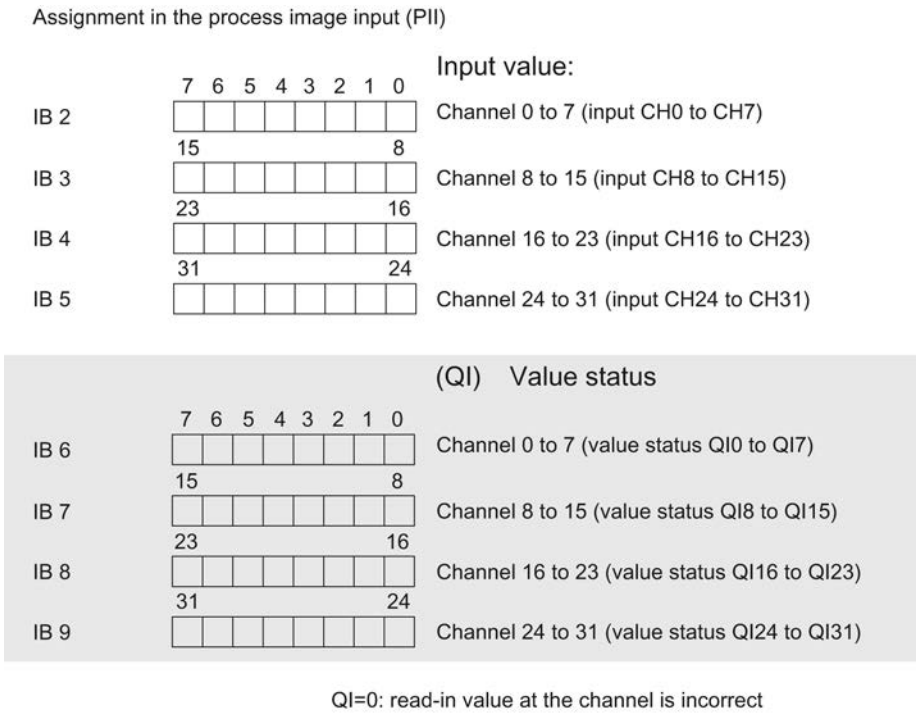
The preset DI configuration is retained initially (32 channels, neither division into submodules nor module-internal shared input (MSI) configuration); however, the value status is to be enabled.

- Select the "Enable value status" check box in the "DI configuration" area in the module properties (Inspector window).

The following figure shows the assignment of the address space for the configuration as a 32-channel module with value status. You can freely assign the start address for the module. The addresses of the channels are derived from the start address.

Example: Start address 2 (input byte 2), end address 9 (input byte 9).

STEP 7 automatically reserves the addresses for the input values (4 bytes) and the subsequent addresses for the value status (also 4 bytes) so that the following address assignment results:



Address space for configuration as 4 x 8-channel DI 32x24VDC HF with value status

The DI configuration is to be adapted so that the 32 channels of the DI 32x24VDC HF module of the S7-1500 are divided among four submodules with 8 channels each. The submodules can be assigned to different IO controllers when the module is used in a Shared Device. The value status is also to be enabled in this case.

- Select the submodule configuration "4 submodules with 8 digital inputs" in the "DI configuration" area in the module properties (Inspector window).
- Select the "Enable value status" check box.

The 4 submodules appear with their own name (e.g., DI 32x24VDC BA_1) in the area navigation of the Inspector window.

Contrary to the 1 x 32-channel module configuration, each of the four submodules has a freely assignable start address.

Example: The start addresses are assigned as follows:

- 1. Submodule: 10
- 2. Submodule: 20
- 3. Submodule 30
- 4. Submodule 40

STEP 7 automatically reserves a byte for the value status for each submodule. The byte for the value status has the address "Start address+1" so that the following address assignment results:

Assignment in the process image input (PII)

		Input value:	
	7 6 5 4 3 2 1 0		
IB 10		Channel 0 to 7 (input CH0 to CH7)	Submodule 1
	15 8		
IB 20		Channel 8 to 15 (input CH8 to CH15)	Submodule 2
	23 16		
IB 30		Channel 16 to 23 (input CH16 to CH23)	Submodule 3
	31 24		
IB 40		Channel 24 to 31 (input CH24 to CH31)	Submodule 4

		(QI) Value status	
	7 6 5 4 3 2 1 0		
IB 11		Channel 0 to 7 (value status QI0 to QI7)	Submodule 1
	15 8		
IB 21		Channel 8 to 16 (value status QI8 to QI15)	Submodule 2
	23 16		
IB 31		Channel 16 to 23 (value status QI16 to QI23)	Submodule 3
	31 24		
IB 41		Channel 24 to 31 (value status QI24 to QI31)	Submodule 4

QI = 0: read-in value at the channel is incorrect

Address space for configuration as 1 x 32-channel DI 32x24VDC HF with MSI and value status

The configuration shown here uses the "Module-internal shared input (MSI)" function. This function allows simultaneous reading of the input values by various IO controllers (CPUs) when the module is used in a Shared Device (ET 200MP).

The channels 0 to 31 of the module are copied in up to 4 submodules with configuration 1 x 32-channel module with module-internal shared input (MSI). Channels 0 to 31 are then available with identical input values in different submodules. These submodules can be assigned to up to four IO controllers when the module is used in a sShared Device. Each IO controller has read access to the same channels ("listening in").

The number of usable submodules (= copies of the inputs) depends on the interface module used. Observe the information regarding this in the device manual of the particular interface module.

- Select the following options in the "DI configuration" area in the module properties (Inspector window):
  - For "Subdivision of module": None
  - For "Copies of module": One additional copy of the inputs
    This setting enables one additional IO controller to read the inputs of the module.

The value status is enabled automatically for MSI configurations (cannot be changed). The reason for this setting is that an IO controller of the Shared Device that is "listening in" can determine from the value status whether or not the module is ready for operation and providing valid values.

Below, the submodules that result from the above-described settings are named as follows:

- Basic submodule: That is the submodule assigned to the parameter-assigning IO controller. Only the parameter-assigning IO controller has access to module diagnostics and receives hardware interrupts.
- MSI submodule: That is the submodule assigned to an IO controller that is "listening in". The IO controller that is listening in does not have access to module diagnostics and only receives information on the validity of the input values by the value status.

Value status (Quality Information, QI)

The meaning of the value status depends on the submodule to which it pertains.

For the first submodule (=basic submodule), the value status 0 indicates that the value is incorrect.

For the 2nd to 4th submodule (=MSI submodule or copy of inputs), the value status 0 indicates that the value is incorrect or that the basic submodule parameters have not yet been assigned (not ready for operation).

Example: The DI 32x24VDC HF module is configured as described above (an additional copy of the inputs). A basic submodule and one MSI submodule result.

The basic submodule receives the start address 10, and the MSI submodule receives the start address 100 in this example. The address of the MSI submodule is not relevant for the IO controller because this submodule will be assigned to another IO controller later in the Shared Device configuration process and will disappear from the address space of the parameter-assigning IO controller as a result.

The following figure shows the assignment of the address space with submodules 1 and 2 and the value status. All submodules are initially assigned to the IO controller.

Assignment in the process image input (PII) for submodules 1 and 2

Input value of submodule 1 (basic submodule):

IB 10	7 6 5 4 3 2 1 0	Channel 0 to 7 (input CH0 to CH7)
IB 11	15 … 8	Channel 8 to 15 (input CH8 to CH15)
IB 12	23 … 16	Channel 16 to 23 (input CH16 to CH23)
IB 13	31 … 24	Channel 24 to 31 (input CH24 to CH31)
IB 14	7 6 5 4 3 2 1 0	Channel 0 to 7 (value status QI0 to QI7)
IB 15	15 … 8	Channel 8 to 15 (value status QI8 to QI15)
IB 16	23 … 16	Channel 16 to 23 (value status QI16 to QI23)
IB 17	31 … 24	Channel 24 to 31 (value status QI24 to QI31)

Input value of submodule 2 (MSI submodule):

IB 100	7 6 5 4 3 2 1 0	Channel 0 to 7 (input CH0 to CH7)
IB 101	15 … 8	Channel 8 to 15 (input CH8 to CH15)
IB 102	23 … 16	Channel 16 to 23 (input CH16 to CH23)
IB 103	31 … 24	Channel 24 to 31 (input CH24 to CH31)
IB 104	7 6 5 4 3 2 1 0	Channel 0 to 7 (value status QI0 to QI7)
IB 105	15 … 8	Channel 8 to 15 (value status QI8 to QI15)
IB 106	23 … 16	Channel 16 to 23 (value status QI16 to QI23)
IB 107	31 … 24	Channel 24 to 31 (value status QI24 to QI31)

QI = 0: read-in value at the channel is incorrect (basic submodule) or module is not ready for operation (MSI submodule)

See also

Evaluating the value status (Page 1846)

Basics of program execution

Events and OBs

OB start events

An OB start event causes the following reaction when it occurs:

- If the event comes from an event source to which you have assigned an OB, it triggers the execution of the assigned OB. This means that the event is positioned in a queue according to its priority.
- If the event comes from an event source to which you have not assigned an OB, the default system reaction is executed.

Note

Some event sources, such as startup, pull/plug, exist even if you do not configure them.

The following table provides an overview of the OB start events, including the possible values for OB priority, possible OB numbers, default system reaction and number of OBs. The table is sorted based on the default OB priority. Priority class 1 is the lowest.

Types of event sources	Possible priorities (default priority)	Possible OB numbers	Default system reaction	Number of supported OBs
Startup	1	100, >= 123	Ignore	100
Cyclic program	1	1, >= 123	Ignore	100
Time interrupt	2 to 24 (2)	10 to 17, >= 123	not applicable	20
Status interrupt	2 to 24 (4)	55	Ignore	1
Update interrupt	2 to 24 (4)	56	Ignore	1
Manufacturer- or profile-specific interrupt	2 to 24 (4)	57	Ignore	1
Time-delay interrupt	2 to 24 (3)	20 to 23, >= 123	not applicable	20
Cyclic interrupt	2 to 24 (8 to 17, depending on the cycle time)	30 to 38, >= 123	not applicable	20
Hardware interrupt	2 to 26 (18)	40 to 47, >= 123	Ignore	50
Isochronous mode interrupt	16 to 26 (21)	61 to 64, >= 123	Ignore	20 (one per isochronous interface)
MC servo interrupt	17 to 31 (25)	91	not applicable	1
MC interpolator interrupt	16 to 30 (24)	92	not applicable	1
Time error	22	80	Ignore	1
Cycle monitoring time violated once			STOP	
Diagnostic interrupt	2 to 26 (5)	82	Ignore	1
Removal/insertion of modules	2 to 26 (6)	83	Ignore	1
Rack error	2 to 26 (6)	86	Ignore	1

Types of event sources	Possible priorities (default priority)	Possible OB numbers	Default system reaction	Number of supported OBs
Programming error (only for global error handling)	2 to 26 (7)	121	STOP	1
I/O access error (only for global error handling)	2 to 26 (7)	122	Ignore	1

Assignment between event source and OBs

The type of OB determines where you make the assignment between OB and event source:

- With hardware interrupts and isochronous mode interrupts, the assignment is made during the configuration of the hardware or when the OB is created.
- STEP 7 automatically assigns OBs 91/92 to the MC Servo interrupt and MC-Interpolator interrupt, as soon as a technology object is added to "S7-1500 Motion Control".
- For all other types of OB, the assignment is made when the OB is created, possibly after you have configured the event source.

For hardware interrupts, you can change an assignment which has already been made during runtime with the instructions ATTACH and DETACH. In this case, only the actually effective assignment is changed, not the configured assignment. The configured assignment is effective after loading and for every startup.

Hardware interrupts to which no OB is assigned (because they were configured in this way or after the DETACH instruction is called) are ignored. The check as to whether an OB is assigned to an event does not take place when the associated event occurs, but only when the hardware interrupt actually has to be executed.

OB priority and runtime characteristics

S7-1500 CPUs support the priority classes 1 (lowest) to 26 (highest). An OB is assigned the priority of its start event.

The OBs are executed exclusively on a priority basis. This means that the OB with the highest priority is executed first when multiple OB requests occur at the same time. If an event with higher priority than the currently active OB occurs, this OB is interrupted. Events of the same priority are processed in order of occurrence.

See also

Organization blocks (OB) (Page 2170)

Basics of error handling (Page 2821)

Principles of local error handling (Page 2823)

ATTACH: Attach an OB to an interrupt event (Page 4691)

DETACH: Detach an OB from an interrupt event (Page 4693)

CPU overload behavior

Principle of CPU overload behavior

The sequence of the user program for S7-1500 CPUs is based on events of different priorities. An event triggers the execution of an OB. The OBs are executed exclusively on a priority basis. This means that the OB with the highest priority is executed first when multiple OB requests occur at the same time. If an event with higher priority than the currently active OB occurs, this OB is interrupted. Events of the same priority are processed in order of occurrence.

For the event scenarios considered in the following section, it is assumed that you have assigned an OB to each event source and that these OBs have the same priority. The second condition, in particular, is only for the sake of a simplified representation.

When an event occurs, the execution of the associated OB is triggered. This means that the event is positioned in a queue according to its priority. Depending on the OB priority and the current processor load, there is usually a time delay before the OB is executed. The same event can therefore occur once or several times before the OB belonging to the preceding event has been processed. Such a situation is handled by the CPU as follows: In the order of their occurrence, the operating system positions the events into the queue according to their priority.

To control temporary overload situations, you can limit the number of pending events that are linked from the same source. The next event is discarded as soon as the maximum number of pending start events of a specific cyclic interrupt OB, for example, is reached.

An overload occurs when events which originate from the same source occur faster than they can be processed.

More detailed information is provided in the following sections.

Discarding similar events or fetching them later

Below, the term "similar events" refers to events from a single source, such as start events for a specific cyclic interrupt OB.

The OB parameter "Events to be queued" is used to specify how many similar events the operating system places in the associated queue and therefore post-processes. If this parameter has the value 1, for example, exactly one event is stored temporarily.

Note

Post-processing of cyclic events is often not desirable, as this can lead to an overload with OBs of the same or lower priority. Therefore, it is generally advantageous to discard similar events and to react to the overload situation during the next scheduled OB processing. If the value of the "Events to be queued" parameter is low, this ensures that an overload situation is mitigated rather than aggravated.

If, for example, the maximum number of start events is reached in the queue for a cyclic interrupt OB, each additional start event is only counted and subsequently discarded. During the next scheduled execution of the OB, the number of discarded start events is made available (in optimized start information) in the "event_count" input parameter. You can then react appropriately to the overload situation. The CPU then sets the counter for lost events to zero.

If the CPU, for example, first discards a start event for a cyclic interrupt OB, its further behavior depends on the OB parameter "Report event overrun into diagnostic buffer": If the check box is selected, the CPU enters the event DW#16#0002:3507 once in the diagnostic buffer for the overload situation at this event source. Additional diagnostic buffer entries of the event DW# 16 #0002:3507 that refer to this event source are suppressed until all events from this source have been post-processed.

Threshold mechanism for time error OB request

The OB parameter "Enable time error" is used to specify whether the time error OB should be called when a specific overload level is reached for similar events.

If yes (check box selected), you use the OB parameter "Event threshold for time error" to specify the number of similar events in the queue at which the time error OB should be called. If this parameter has the value 1, for example, the event DW#16#0002:3502 is entered once in the diagnostic buffer and the time error OB is requested when the second event occurs. Additional diagnostic buffer entries of the event DW# 16 #0002:3502 that refer to this event source are suppressed until all events from this source have been post-processed.

In the event of an overload, you therefore have the option of programming a reaction well before the limit is reached for similar events and thus before the events are discarded.

The following value range applies to the "Event threshold for time error" parameter: 1 <= "Event threshold for time error" <= "Events to be queued"

Example of a hardware interrupt event

The functional principle of event-oriented program execution in the S7-1500 CPU is described based on the example of a module triggering a hardware interrupt.

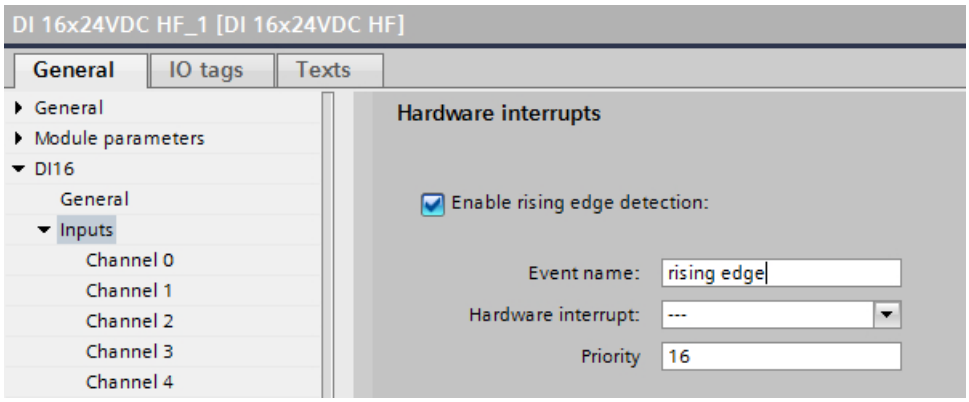
Process events are events which are triggered by the I/O (e.g. a digital input) and initiate a call of the assigned OB in the S7-1500 CPU. OBs assigned to a process event are called hardware interrupt OBs.

Procedure

To configure a hardware interrupt, proceed as follows:

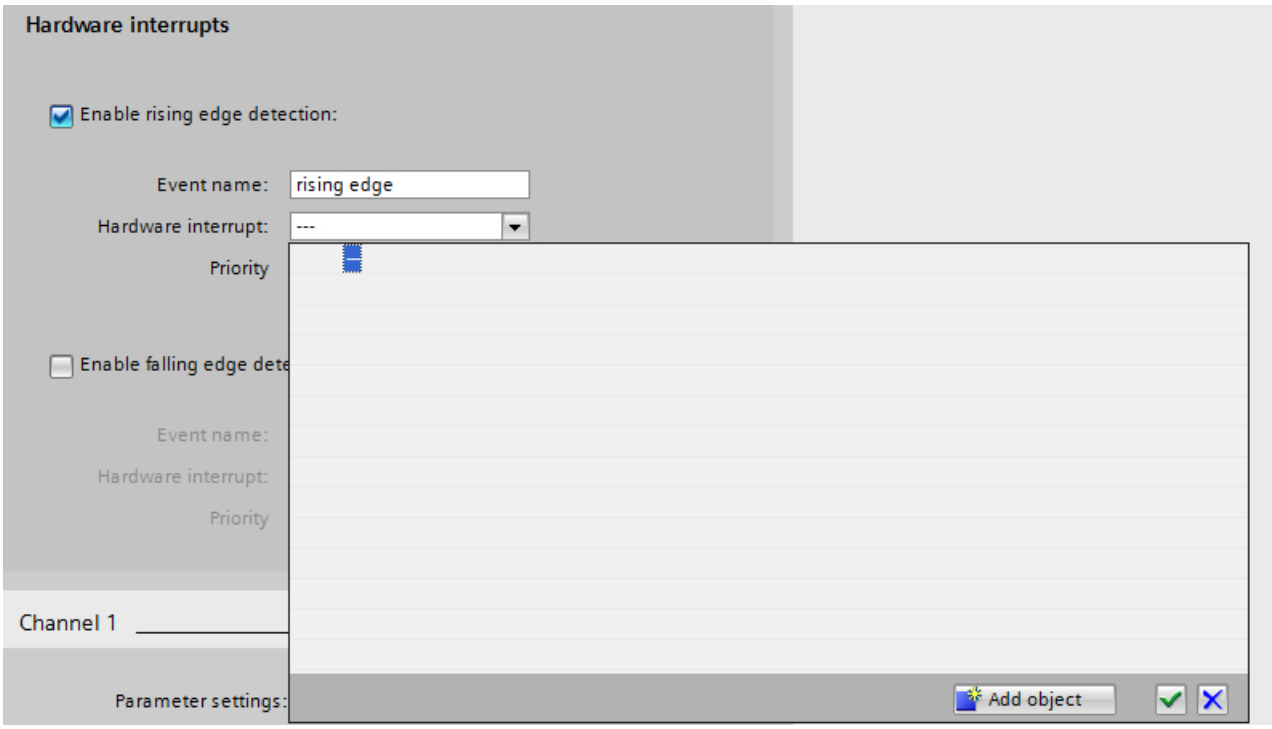
1. Configure an S7-1500 with CPU and a hardware interrupt-compatible module, e.g. DI 16x24VDC HF.
2. Select the hardware interrupt-compatible module and navigate in the area navigation to the area "Inputs > Channel 0".
3. Activate the option "Enable rising edge detection".
4. Assign a meaningful name to the event.

5. If necessary, change the priority of this event.
   In the PLC tags ("System constants" tab), a system constant of the data type Event_HwInt is created for the event.

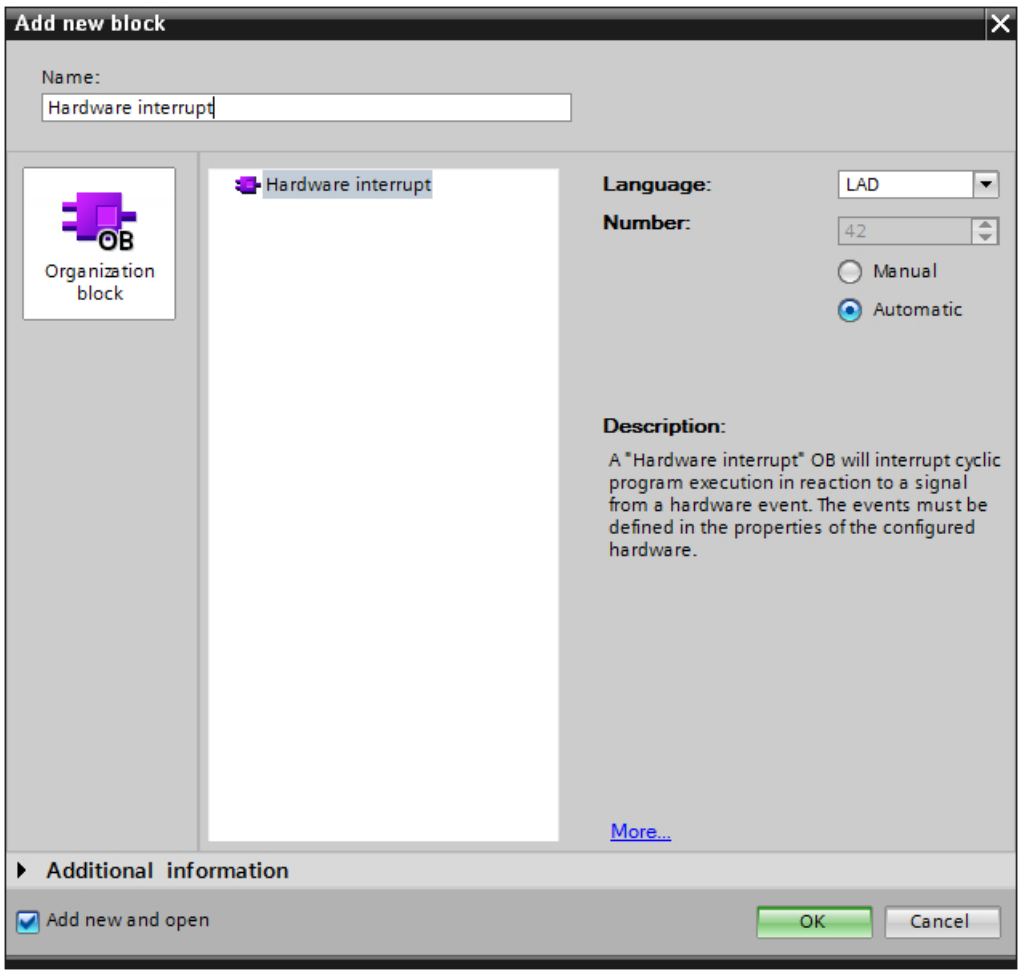


6. Click the "Hardware interrupt" drop down list to create a new hardware interrupt OB. If a hardware interrupt OB already exists, it can be selected.

7. Click the "Add object" button to create a new hardware interrupt OB.

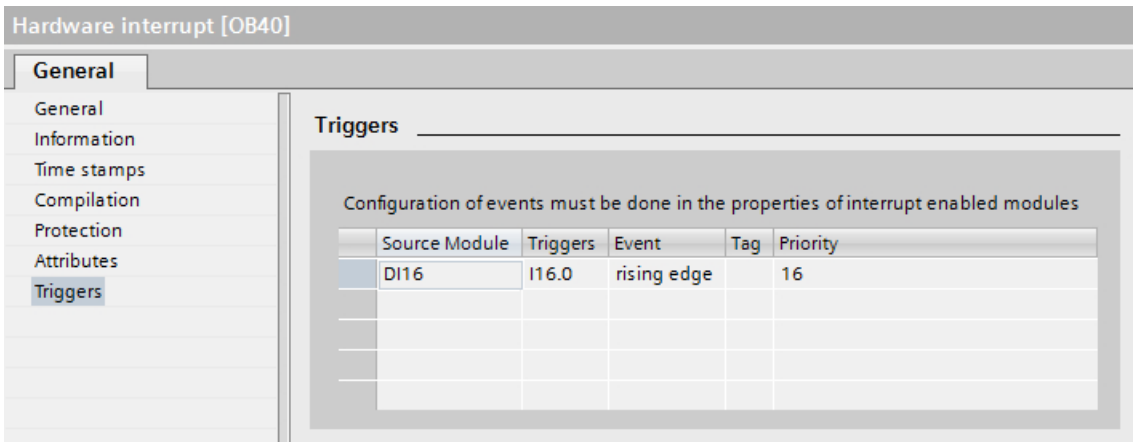


8. You can specify in the selection dialog whether the programming editor should start immediately with the opened hardware interrupt OB ("Add new and open" option).



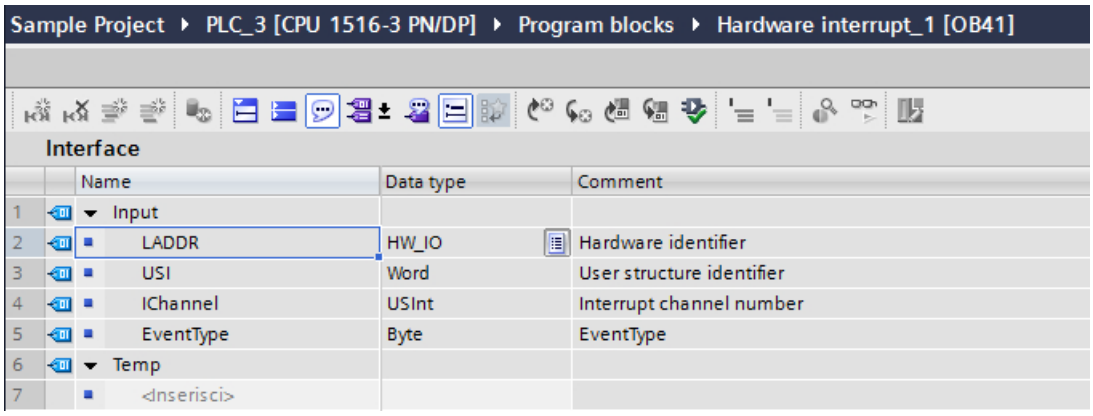
Hardware interrupt OB properties

The properties of the hardware interrupt OB contain the corresponding information about the start event for each interrupt-triggering event:



Hardware interrupt OB program

In the user program, you can use the start information of the hardware interrupt OB to identify and evaluate the hardware interrupt event.



See also

Assigning parameters to hardware interrupt OBs (Page 1918)

Hardware interrupt OBs (Page 1902)

What you need to know about HW identifiers (Page 883)

Setting the operating behavior

CPU properties

Overview of the CPU properties

Overview

The following table provides you with an overview of the CPU properties:

Group	Properties	Description
General	Project information	General information to describe the inserted CPU. Except for the slot number, you can change this information.
	Catalog information	Read-only information from the hardware catalog for this CPU.
	Identification & Maintenance	For saving application-specific information such as the name of the plant and the installation location.
PROFINET interface	General	Name and comment for this PROFINET interface. The name is limited to 110 characters.

Group	Properties	Description
	Ethernet addresses	Select whether the PROFINET interface is networked. If subnets have already been created in the project, they are available for selection in the drop-down list. If not, you can create a new subnet with the "Add new subnet" button. Information on the IP address, subnet mask and IP router usage in the subnet is available in the IP protocol. If an IP router is used, the information about the IP address of the IP router is necessary. See: Assigning addresses and names to PROFINET devices (Page 1441)
	Time-of-day synchronization	Settings for time synchronization in the NTP time format. The NTP (network time protocol) is a general mechanism for synchronizing system clocks in local and global area networks. In NTP mode, the interface of the CPU sends time queries (in client mode) at regular intervals to NTP servers on the subnet (LAN) and the addresses must be set in the parameters here. Based on the replies from the server, the most reliable and most accurate time is calculated and synchronized. The advantage of this mode is that it allows the time to be synchronized across subnets. The accuracy depends on the quality of the NTP server being used.
	Operating mode	Selection of operation of CPU as IO controller or IO device including the necessary settings. As IO device (I-device), you must assign an IO controller and specify if the PROFINET interface is configured by the higher-level IO controller or if the local IO controller configures the PROFINET interface. The prioritized startup can also be enabled for the role IO device if the higher-level IO controller supports this function. If you have configured the CPU as IO device, the table for configuration of the transfer areas is displayed. See: Configuring the I-device (Page 1495)

Group	Properties	Description
	Advanced options	Name, comment and additional setting options of the Ethernet interface and the ports. This includes the following settings: • Interface options such as the reaction of the user program in case of communication errors, if device replacement is possible without exchangeable medium or automatic commissioning, setting regarding IEEE-compliant LLDP mode (PROFINET V2.3), time lag between sending of Keep Alives for connections. • Media redundancy settings, such as the media redundancy role (client, manager (Auto) or devices not participating in the media redundancy ring), display of the ring ports and if diagnostic interrupts are generated in case of redundancy errors. A "Domain settings" interface is the link to the domain settings of the connected subnet (MRP domain and Sync domain). • Real-time settings such as send clock and synchronization properties with a "Domain settings" interface which is the link to the domain settings of the connected subnet (MRP domain and Sync domain). In the Sync domain you set the send clock for isochronous IO systems and specify the RT classes (RT and IRT) as well as the synchronization roles for IRT. • Ports: For ports you set the interconnection with neighboring devices (alternative: use the topology view), if necessary. Port interconnections are required for isochronous applications (IRT) as well as for automatic commissioning ("Device replacement without exchangeable medium" option is selected). You also set the port options, for example, if the port is used (you can block unused ports). The port options include the connection settings (transmission rate/duplex) with the options for monitoring and autonegotiation. You also set the boundaries for the end of detecting accessible devices, for the end of topology detection and for the end of the sync domain. The hardware identifiers are displayed for ports. See: Setting the send clock (Page 1455) Interconnecting ports (Page 1455) Setting the port options (Page 1458) What you should know about media redundancy (Page 1512) Introduction: Isochronous Realtime Ethernet (Page 1472) Enabling device replacement without exchangeable medium (Page 1461)
	Hardware identifier	The hardware identifier of the interface is displayed. See: What you need to know about HW identifiers (Page 883)
	Web server access	Allows access to the web server function.
PROFIBUS interface	General	Name and comment for this PROFIBUS interface. The name is limited to 110 characters.

Group	Properties	Description
	PROFIBUS address	Selection of whether the PROFIBUS interface is networked. If subnets have already been created in the project, they are available for selection in the drop-down list. If not, you can create a new subnet with the "Add new subnet" button. The PROFIBUS address is assigned and the highest address and the transmission speed are displayed.
	Operating mode	Selection of CPU operating mode as DP master or DP slave. A DP master can be assigned to a DP slave and the I-slave communication parameters can be set here.
	Clock	The time can be synchronized. A default can be set for the time source and synchronization frequency.
	SYNC/FREEZE	The DP master can send the SYNC and/or FREEZE control command to a group of DP slaves to synchronize these. For this purpose, the DP slaves are assigned to SYNC/FREEZE groups.
	Hardware identifier	The hardware identifier of the interface is displayed.
Startup	Startup after POWER ON	Setting the startup characteristics after a POWER OFF/POWER ON transition. See: Basics of "STARTUP" operating mode (Page 1833)

Group	Properties	Description
	Comparison of preset configuration and actual configuration	Specifies the startup characteristics for situations in which the actual configuration of the S7-1500 station does not correspond to the preset configuration: • Startup of the CPU only if compatible • Startup of the CPU even if there are differences With the "Startup of the CPU only if compatible" setting, a module on a configured slot must be compatible with the configured module. Compatible means that the module that is present matches the number of inputs and output and must match with respect to its electrical and functional properties. A compatible module must be fully able to replace a configured module; it may be more capable, but not less capable. Example for the "Startup of the CPU only if compatible" setting: A CPU can be a compatible replacement for a CPU of the same type with a lower firmware version. An input module with 32 digital inputs can be a compatible replacement for a signal module with 16 digital inputs. The CPU starts when the configured module or a compatible module is plugged. The CPU does not start up if an incompatible module is inserted. Example for the "Startup of the CPU even if there are differences" setting: Instead of a configured digital input module an analog output module is plugged or no module is present in this slot. Although the configured inputs cannot be accessed, the CPU starts up. Note in this case that the user program cannot function correctly, so take the appropriate measures! Note: For S7-1500 modules, you can configure the comparison between preset module and actual module individually for each slot. The default is "From CPU" (in other words, the CPU's setting applies to the slot). For I/O modules from other I/O series, generally the CPU's setting applies (cannot be changed).
	Parameter assignment time for central and distributed I/O	Specifies a maximum time period (default: 60000 ms) in which the central and distributed I/O must be ready for operation. The CMs and CPs are supplied with voltage and communication parameters during the CPU startup. The configuration time limits the period during which I/O modules connected to the CM or CP must be ready for operation. If the central and distributed I/O is not ready for operation within the configuration time, the startup characteristics of the CPU depends on the setting of the "Supported hardware compatibility" parameter.
Time	Local time and daylight saving time	Selection of the time zone in which the CPU operates and setting of standard/daylight saving time.

Group	Properties	Description
Clock	-	Definition for synchronizing the clock via the backplane bus. The setting options depend on the CPU. • As slave: The clock is synchronized by another clock. • As master: The clock synchronizes other clocks. • None: No synchronization takes place. The time intervals for the synchronization can be selected. See: Auto-Hotspot
Protection	Protection and password for read/ write access	Setting the read/write protection and the password for access to the CPU. See: Setting options for the protection (Page 1884) See: Configuring access levels (Page 1886)
System and clock memory	System memory bits and clock memory bits	Setting a byte for the system memory functions and setting a byte for the clock memory functions (switches each bit on and off in a specified frequency). You use system memory bits for the following scans: • Is the current cycle the first since cycling power? • Have there been any diagnostics state changes compared with the previous cycle? • Scan for "1" (high) • Scan for "0" (low) Clock memory bits change their values at specified periodic intervals. See: Enabling system memory (Page 1883) See: Using clock memory (Page 1884)
Cycle	Maximum cycle time and minimum cycle time	Specification of a maximum cycle time or a fixed minimum cycle time. If the cycle time exceeds the maximum cycle time, the CPU goes to STOP mode. Causes of a violation: • Communication processes • Accumulation of interrupt events • CPU program error. If the cycle time is shorter than the minimum cycle time you have entered, the CPU waits until the minimum cycle time has been reached. The CPU processes the OB 90 (background OB) in this additional program processing time, if it is loaded. See: Cycle time and maximum cycle time (Page 1869)
Communication load	Cycle load due to communication	Controls the duration of communication processes that always also extend the cycle time, within certain limits. Examples of communication processes include: Transferring data to another CPU or loading blocks (initiated via the PC). See: Cycle load due to communication (Page 1870)

Group	Properties	Description
System diagnostics	General	System diagnostics is the recording, evaluation and reporting of errors within the automation system. Examples of errors: • CPU program error • Failure of modules • Wire break to sensors and actuators Default alarm texts are available for determining the system diagnostics. If necessary, existing alarm texts can be changed and new ones added. The settings are saved with the project and are effective only after the compilation and loading of the hardware configuration to the relevant components. For S7-1500, the system diagnostics is automatically activated (cannot be deactivated).
	Alarm settings	You set for each of the alarm categories, such as errors, maintenance demanded, maintenance required and information, if the alarms of this category are displayed and if they have to be acknowledged.
Address overview	-	Tabular representation of all addresses used by the CPU for integrated inputs/outputs as well as for the inserted modules. Addresses that are not used by any module are represented as gaps. The view can be filtered according to • Input addresses • Output addresses • Address gaps
Connection resources	-	Provides an overview of the assigned reserved and dynamic resources for the CPU connections. The currently used resources are also displayed in the online view. See: Consumption of connection resources (Page 579)
Display	General	The display remains dark in standby mode and is re-activated as soon as one of the display's keys is pressed. In the display menu of the display, you can also change the standby mode. In energy-saving mode, the display shows the information with reduced brightness. Energy-saving mode is switched off as soon as any display key is pressed. You can also change the energy-saving mode in the display menu of the display. Display language: The language is changed immediately after the hardware configuration is loaded with the set standard language. You can also change the language in the display menu of the display.
	Automatic update	Input of the time interval at which the display is updated.
	Password	Option to input a password for protection from unauthorized access. You can set the time until automatic logoff after input of the password on the display.
	Watch tables	Configured watch tables are offered for selection. You can use the selected watch tables on the display during operation.
	User-defined logo	You can select a user-defined logo and load it with the hardware configuration into the CPU.

Group	Properties	Description
Web server	-	Activates and configures the web server function. See: Enable the web server (Page 909)
	Automatic update	Sends the requested web page with current CPU data periodically to the web browser. Enter the period duration under "Update interval". Automatic update can only be activated if the web server is enabled. See: Activate automatic update (Page 909)
	User Management	Here you enter the user names with their access rights. • Everyone: Pre-defined user whose access rights you assign using a drop-down list. The "Everyone" user is a user who does not log on with a password for web access. • User#: Editable name for a user for whom you assign access rights and a password. As soon as you click in a line below "Everyone", a preset name is displayed that you can change. Each user can be assigned different access rights via a drop-down list. There are access rights that are connected to each other. Example: When you activate the access right "Perform firmware update", the access rights "Change operating mode" and "... query diagnostics" are activated automatically. These two rights are required to perform a firmware update from the web server. See: User lists (Page 911)
	User-defined web pages	Allows access to freely-designed web pages of the CPU via a web browser. See: Create and download user-defined websites (Page 913)
	Interface overview	Tabular representation of all modules with their Ethernet interfaces providing web server functionality for this device. Here you can permit or deny access to the web server by means of the respective interface for each Ethernet interface of the device (CPU, CP, CM).
User interface languages	-	The web server and the CPU display support different user interface languages. A project language can be assigned to each language of the web server to display the project texts (up to 2 are supported). Here you specify the languages that are to be displayed in the CPU display for selection during web access. A project language is assigned to the web server language, for example, the project language "English (Ireland)" to the web server language English, by selecting it from the drop-down list. Activate the project language at "Project languages" in the project tree to make it available for selection in the drop-down list.
Configuration control	-	Enables configuration control. Enables a configuration change in the user program within certain limits. See: Configuration control

Group	Properties	Description
Power supply	Power supply	Specifies that the CPU is connected to the power grid. As a result, this CPU performance has a positive impact on the power supply/ consumption ratio.
	Power budget	In the power segment overview, the power provided by the CPU and the power supply modules is compared with the power required by the signal modules. If the power supply/consumption ratio is negative, in other words, the power requirements are higher than the power that is provided by the CPU and the power supply modules, this is indicated in the table.

See also

Set the regional web language (Page 910)

Setting prioritized startup (Page 1510)

Assigning the device name and IP address (Page 1442)

CPU settings on the display (Page 1868)

Rules for CPUs with several PROFINET interfaces (Page 1871)

IEEE-compliant LLDP (Page 1873)

CPU settings on the display

The S7-1500 CPUs have a display with operator control buttons. The display can be used to read out status information about the configured modules, for diagnostic purposes, and also to adapt some CPU settings.

Example: Setting the IP address and the PROFINET device name on the display

If you want to assign the IP address parameters (IP address, subnet mask and router settings) on the display, you must select the following option in the relevant project for the PROFINET interface of the CPU:

- Set IP address on the device

The same applies accordingly to the PROFINET device name: If you want to assign the PROFINET device name on the display, then you must select the following option in the relevant project for the PROFINET interface of the CPU:

- Set PROFINET device name on the device

Cycle time and maximum cycle time

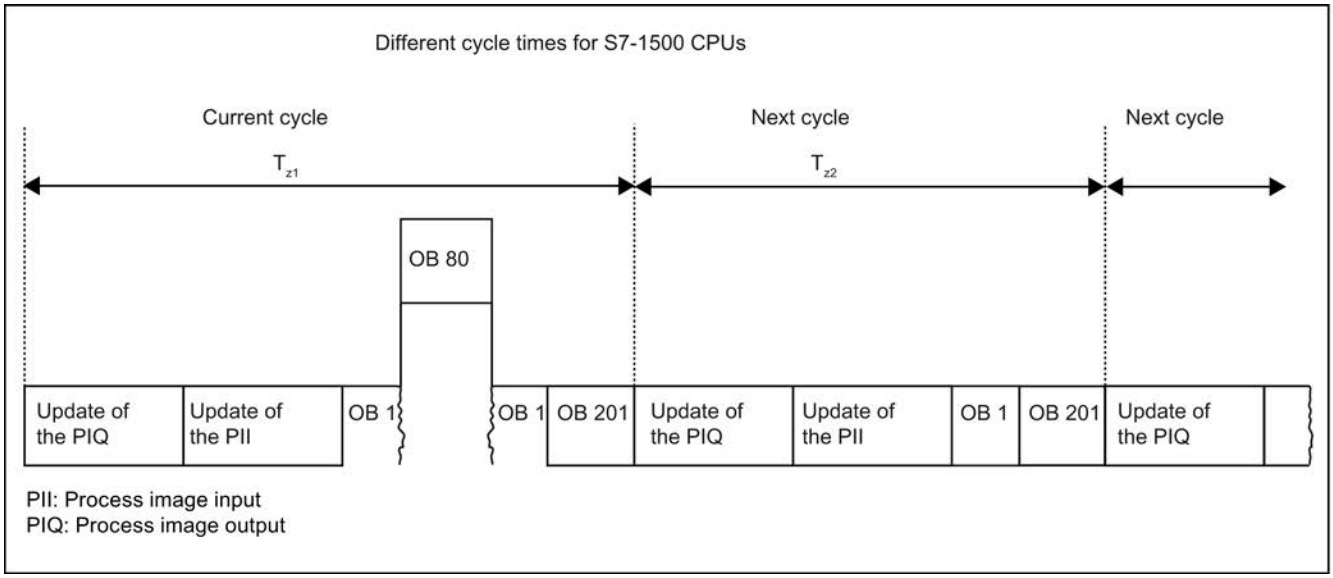
Function

The cycle time is the time that the operating system requires to execute the cyclic program and all the program sections that interrupt this cycle. The program execution can be interrupted by:

- Time errors and 2xMaxCycleTime errors
- System activities, e.g., process image updating

The cycle time (Tcyc) is therefore not the same for every cycle.

The following schematic shows an example of different cycle times (TZ1 ≠ TZ2) for S7-1500 CPUs:



In the current cycle, the cyclic OB used here (e.g. OB 1) will be interrupted by a time error (e.g. OB 80). Following the cyclic OB, the next cycle OB 201 is processed.

Maximum cycle time

The operating system monitors the execution time of the cyclic program for a configurable upper limit known as the maximum cycle time. You can restart this time monitoring at any point in your program by calling the RE_TRIGR instruction.

If the cyclic program exceeds the maximum cycle time, the operating system will attempt to start the time error OB (OB 80). If the OB is not available, the CPU ignores the overshoot of the maximum cycle time.

In addition to monitoring the runtime for overshooting of the maximum cycle time, adherence to a minimum cycle time is guaranteed. To do this, the operating system delays the start of the new cycle until the minimum cycle time has been reached. During this waiting time, new events and operating system services are processed.

If the maximum cycle time is exceeded a second time, for example while the time error OB is being processed (2xMaxCycleTime error), the CPU changes to STOP mode.

Cycle load due to communication

Function

The cycle time of the CPU can be extended due to communication processes. These communication processes include for example:

- Transferring data to another CPU
- Loading of blocks initiated by a programming device

You can control the duration of these communication processes to some extent using the CPU parameter "Cycle load due to communication".

In addition to communication processes, test functions also extend the cycle time. The "Cycle load due to communication" parameter can be used to influence the duration.

How the parameter works

You use the "Cycle load due to communication" parameter to enter the percentage of the overall CPU processing capacity that can be available for communication processes.

This processing capacity can be used for program execution when not required for communication.

For the priority classes **lower than or equal to 15**, the processing power for communication is continuously assigned by the CPU.

Priority classes **greater than 15** are not by interrupted by communication.

By assigning the priority of an event or an OB, you thus influence the interruptibility of the OB and the blocks called by the OB. If program sections are made uninterruptible in this manner in order to minimize cycle times, the online functions of STEP 7 are delayed!

Effect on the actual cycle time

The "Cycle load due to communication" parameter can be used to extend the cycle time of the cyclic organization block (e.g., OB 1) by a factor calculated according to the following formula:

$$\frac{100}{100 - \text{"Cycle load due to communication"}}$$

The formula does not take into account the effect of asynchronous events such as hardware interrupts or cyclic interrupts on the cycle time.

If the cycle time is extended due to communication processes, more asynchronous events may occur within the cycle time of the cyclic organization block. This extends the cycle still further. The extension depends on how many events occur and how long it takes to process them.

Example 1 – no additional asynchronous events:

If the "Cycle load due to communication" parameter is set to 50%, this can cause the cycle time of the cyclic organization block to increase by up to a factor of 2.

Example 2 – additional asynchronous events:

For a pure cycle time of 500 ms, a communication load of 50% can result in an actual cycle time of up to 1000 ms, provided that the CPU always has enough communication jobs to process.

If, parallel to this, a cyclic interrupt (priority class lower than or equal to 15) with 20 ms processing time is executed every 100 ms, this cyclic interrupt would extend the cycle by a total of 5*20 ms = 100 ms without communication load. In other words, the actual cycle time would be 600 ms.

Since communication at 50% communication load extends the time required for both the cyclic program and the cyclic interrupt, the actual cycle time amounts to 1200 ms instead of 1000 ms.

If the cyclic interrupt has a priority greater than 15, the processing time of the cyclic interrupt OB is not extended by communication, but the actual cycle time remains the same (1200 ms).

Note

Observe the following:

- Check the effects of changing the value of the "Cycle load due to communication" parameter while the system is running.
- You must always consider the communication load when setting the minimum cycle time otherwise time errors occur.

Recommendations

- Increase this value only if the CPU is used primarily for communication purposes and the user program is not time critical.
- In all other situations you should only reduce this value.
- If you do not want interrupt OBs to be interrupted by communication, you must set their priority to a value greater than 15.

CPUs with several PROFINET interfaces

Rules for CPUs with several PROFINET interfaces

The S7-1500 family of CPUs includes CPUs with two or three PROFINET interfaces, for example, CPU 1516-3 PN/DP.

One PROFINET interface can be used for PROFINET IO communication, while the other is being used for communication with a higher-level network (backbone, router, Internet) or for communication with a different machine or automation cell, for example.

Please note that some settings can be changed only at interface X1 and are read-only at the other interfaces (for example, X2).

Rules

The following applies to an S7-1500 CPU with several PROFINET interfaces:

- Only the first PROFINET interface (X1) supports PROFINET IO and isochronous real-time (IRT). Real-time settings can be configured only at this interface.
- The second or all other PROFINET interfaces (e.g., X2) supports PROFINET basic functionality (no IO controller/IO device role).
  - The second or the other interfaces can, for example, send topology via LLDP protocol, manage LLDP MIBs, reply to and manage SNMP requests, support the "Accessible devices" function (DCP protocol), etc.
- The IP subnets of both interfaces must be different. This means that the subnet part of the IP address of both interfaces must be different from each other. Advantages regarding communication across several PROFINET interfaces only apply when the interfaces are in different IP subnets.
  Example:
  - Subnet mask of both interfaces: 255.255.255.0
  - IP address of the first interface: 192.168.0.10
  - IP address of the second interface: 192.168.1.10
- The interfaces can be connected to the same Ethernet subnet.
- Only one router may be effective at any time. The IP address of the router must be located in the IP subnet of the PROFINET interface on which the router is configured.
- The name of the interface and the name of the device (CPU) are used in the automatic assignment of the device name. Example: plc_1.profinet interface_1
- Time-of-day synchronization: The activation of the synchronization and the settings for the NTP servers can only be made on the PROFINET interface X1. The settings are applied for the interface X2 and additional interfaces (read-only there).
- Keep-alive option: For the activation of the advanced option "Send keepalives for connections ... ", the same applies as for the time-of-day synchronization: Can be activated only at interface X1, read-only at interface X2 and additional interfaces.

Go online

For CPUs with multiple PROFINET interfaces, it is not sufficient when a PG which is to be connected online with the CPU and the PROFINET interfaces are located in the same S7 subnet. The IP subnet addresses of the PG and the PROFINET interfaces must be taken into account.

If you want to use a PG/PC with STEP 7 to access a CPU with two or three PROFINET interfaces online, the following activities then run in the background:

- While establishing the online connection, STEP 7 checks whether one of the PROFINET interfaces connected to the S7 subnet is located in the same IP subnet as the IP address (= IP address of the online access of the PG/PC) and then uses this IP address/PROFINET interface to go online.
- If none of the PROFINET interfaces is located in the same IP subnet, STEP 7 goes online via the PROFINET interfaces for which a router is configured.
- If these requirements are not met, STEP 7 prompts the user to change the IP subnet address of the PG/PC interface so that PG/PC is in the same IP subnet.

IEEE-compliant LLDP

LLDP stands for "Link Layer Discovery Protocol"; a manufacturer-independent protocol that is defined in the standard IEEE-802.1AB.

Ethernet devices use LLDP to send information about themselves to their neighboring devices in regular intervals. The neighboring devices save this information.

SNMP tools read this data and use it to create the network topology (SNMP stands for "Simple Network Management Protocol").

LLDP and IEC 61158

The fieldbus standard IEC 61158 V2.2, or "IEC V2.2" for short, implements LLDP for devices with PROFINET interface. SNMP tools use it to recognize the topology of Ethernet networks to which devices with PROFINET interface are connected.

Devices with several PROFINET interfaces require a new version of this standard, IEC V2.3. Siemens tools and third-party tools use it to detect all interfaces of a device with several PROFINET interfaces.

All networked PROFINET interfaces must be set to the same mode (either IEC V2.3 or IEC V2.2).

When you configure all devices in the PROFINET subnet in one project, STEP 7 automatically sets the correct mode and you do not have to worry about the settings. Below you will find out about the rules STEP 7 uses to automatically make the settings and the cases in which you have to set the mode manually.

How do you determine which mode is supported by a PROFINET interface?

You can see which mode is supported by a PROFINET interface in the properties of the PROFINET interface for a selected device or selected interface in the device view or network view. You can find the option "Use IEC V2.2 LLDP mode" in the area "PROFINET interface > Advanced options".

- If the option "Use IEC V2.2 LLDP mode" is selected and cannot be changed, the PROFINET interface only supports V2.2 mode.
- If the option "Use IEC V2.2 LLDP mode" is disabled and can be changed, the PROFINET interface supports V2.2 mode as well as V2.3 mode.

Automatic setting of the PROFINET interfaces

STEP 7 sets the mode of the PROFINET interfaces according to the following rules:

- If the devices support IEC V2.3 mode, STEP 7 sets all PROFINET interfaces on the subnet consistently to mode IEC V2.3.
- If only one device supports IEC V2.2 mode, STEP 7 sets all PROFINET interfaces on the subnet to IEC V2.2 mode.

Example 1: Automatic setting

In this example, an I-device was created in project 1 and then inserted in project 2 as GSD file (two projects).

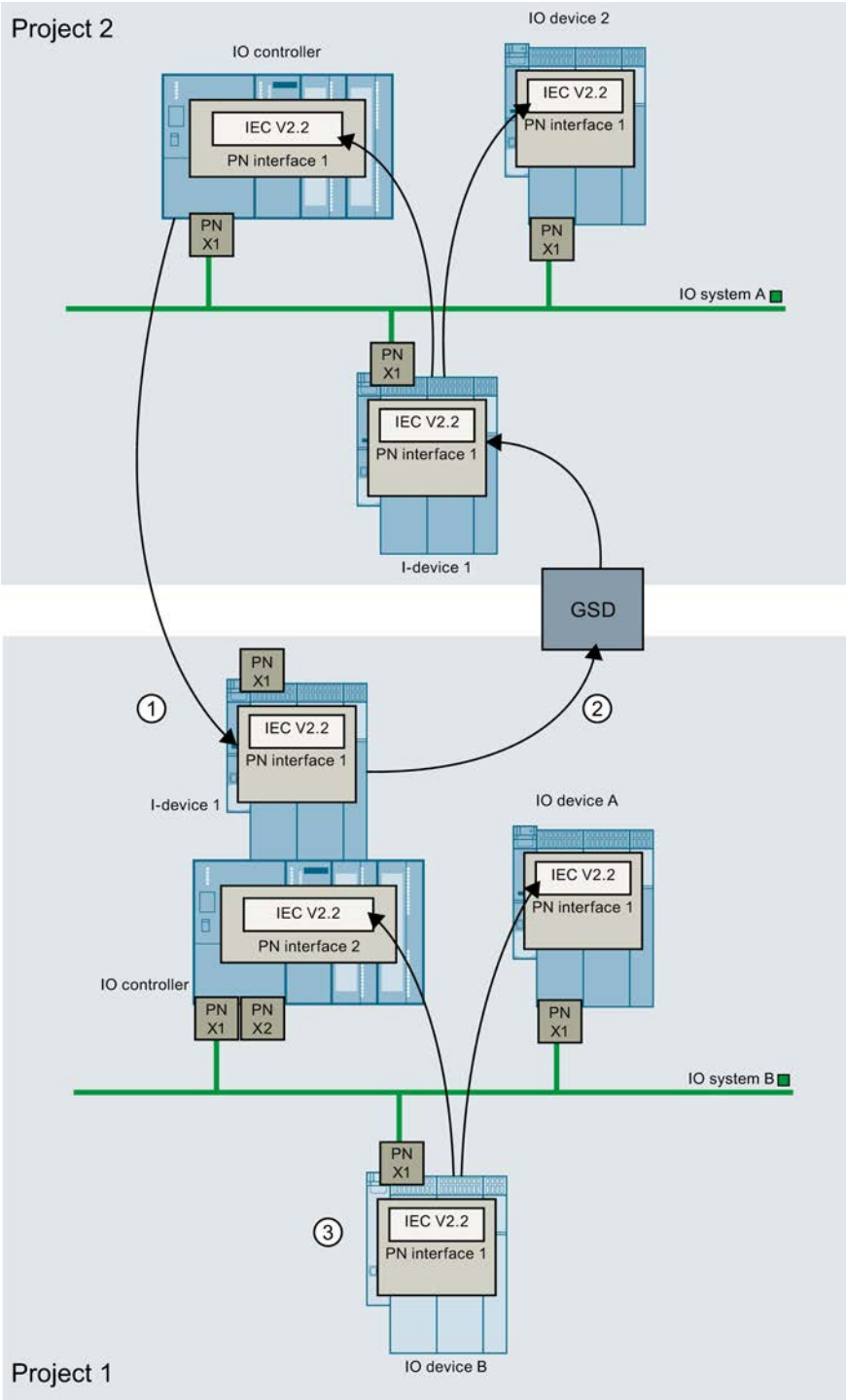
Only one device (3) in project 1 supports mode V2.2. STEP 7 therefore set all PROFINET interfaces in this project to mode V2.2 and noted this setting in the GSD file.

The I-device project was then exported in a GSD file (2) and inserted into project 2.

STEP 7 automatically set all PROFINET interfaces in project 2 also to mode V2.2 due to the entry in the GSD file.

In this example, the higher-level IO controller configures the lower-level IO system (1).

SNMP tools detect only one interface for each device in this example. Tools from Siemens detect all PROFINET interfaces.



① Higher-level IO controller configures the PROFINET interface of the I-device in the IO system A.

② PROFINET GSD file has entry that only IEC V2.2 mode is supported.

③ Devices with IEC V2.2 mode force IEC V2.2 mode for all PROFINET interfaces in project 1.

Manual setting of the PROFINET interfaces

The manual setting of a PROFINET interface is required in the following cases:

- You configure a system in two or more projects. A device that only supports V2.2 mode is used in one of the projects (project 1). Only devices in V2.3 mode are used in another project (project 2). In this case, you must set one of the devices in project 2 manually to V2.2 mode.
- You are using two projects. The first project includes an I-device. The second project includes a higher-level IO system in which the I-device is integrated. This case is explained in greater detail below:

Setting the IEC mode manually

If the following conditions apply, you must manually set a PROFINET interface to IEC V2.2 mode:

- You configure an I-device and export the project as GSD file.
- You use the GSD file in another project.
- You insert the I-device into a higher-level IO system.
- The project with the I-device only uses devices which support V2.3 mode. This mode is set automatically.
- The higher-level IO system includes a device which only supports the standard IEC V2.2.

You force a PROFINET interface of a device in the lower-level IO system into V2.2 mode by executing the following steps.

1. Open the I-device project in STEP 7.
2. Select any device.
3. Enable the option "Use IEC V2.2 LLDP mode" in the area "Interface options".

Then, generate the GSD file once again. Use the new GSD file in the higher-level IO system.

Download the new I-device configuration to the I-device CPU.

Example 2: Setting the IEC mode manually

Project 1: You have created an I-device in the project.

All PROFINET interfaces support V2.3 mode.

This means STEP 7 did not change the mode of the PROFINET interfaces and noted mode V2.3 in the GSD file (2).

Project 2: I-device from project 1 is inserted (GSD file).

One device only supports V2.2 mode.

This means STEP 7 sets all PROFINET interfaces in this project to V2.2 mode.

The PROFINET interfaces in project 1 and 2 have different settings.

This means you have to open the I-device project (project 1) in STEP 7 again and manually set a PROFINET interface to V2.2 mode.

You set a PROFINET interface of a device in the lower-level IO system to V2.2 mode by executing the following steps.

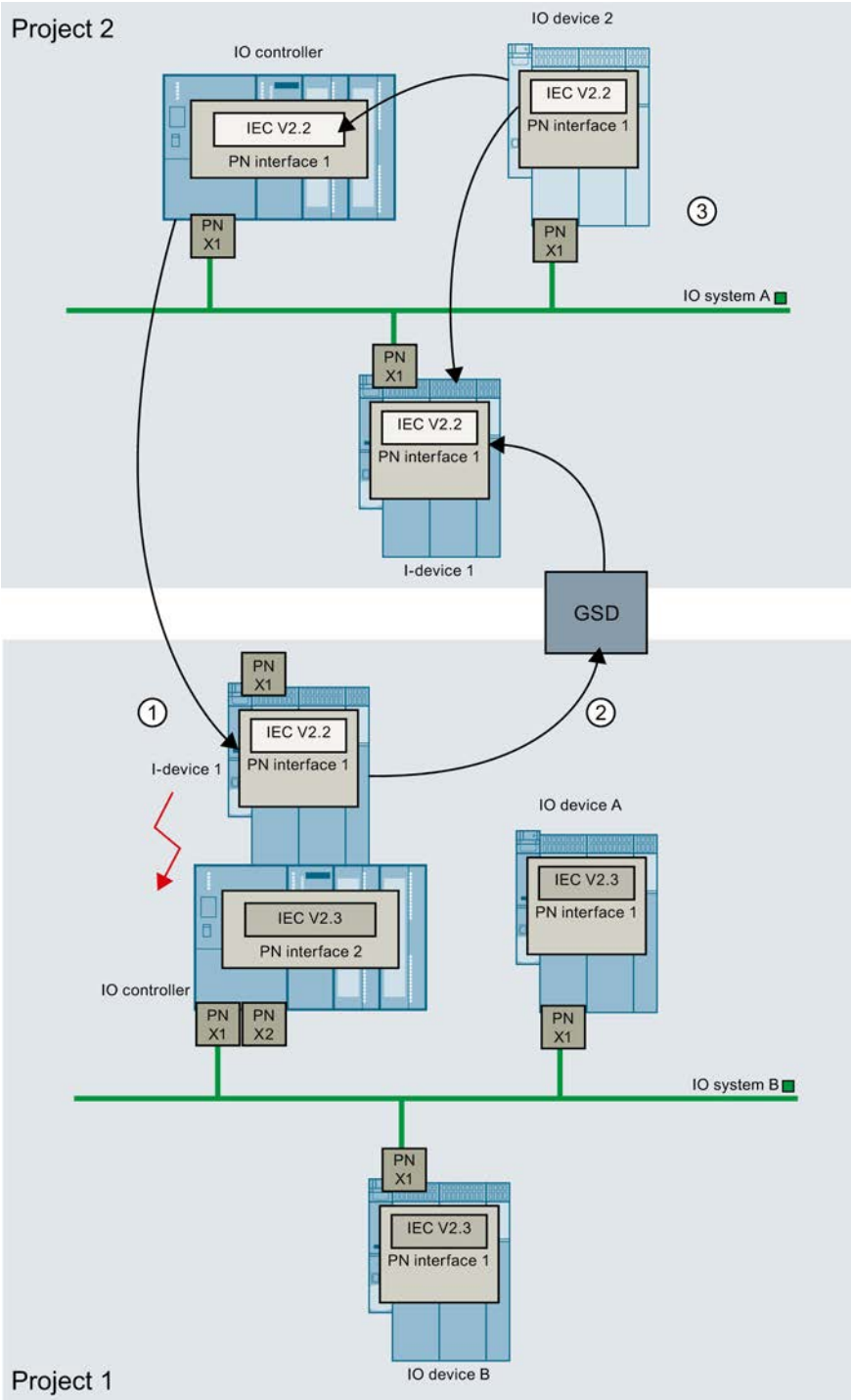
1. Open the I-device project in STEP 7.
2. Select any device.
3. Enable the option "Use IEC V2.2 LLDP mode" in the area "Interface options".

Then, generate a new GSD file. Insert the new GSD file in the higher-level IO system (project 2).

Download the project to the I-device CPU once again.

SNMP tools detect only one interface for each device in this example. Tools from Siemens detect all PROFINET interfaces.

If you do not change the setting of the PROFINET interface, each interface sends a diagnostics alarm during runtime (during operation).



① The PROFINET interfaces of the I-device have different settings (the PROFINET interface in IO system A is set to V2.2 mode; the PROFINET interface in the IO system B is set to V2.3 mode).

② GSD file has entry that the I-device only supports IEC V2.3 mode.

③ IO device 2 only supports V2.2 mode. This means STEP 7 sets all other PROFINET interfaces in project 2 to this mode (including the PROFINET interface of the I-device in IO system A).

Example 3: Setting the IEC mode manually

This example is largely the same as example 2.

Only difference: The settings of the PROFINET interfaces are stored in the I-device (no parameter assignment by the higher-level IO controller as in example 2).

The PROFINET interfaces have different settings in IO system A (I-device set to V2.3, IO controller set to V2.2, IO device 2 only supports V2.2).

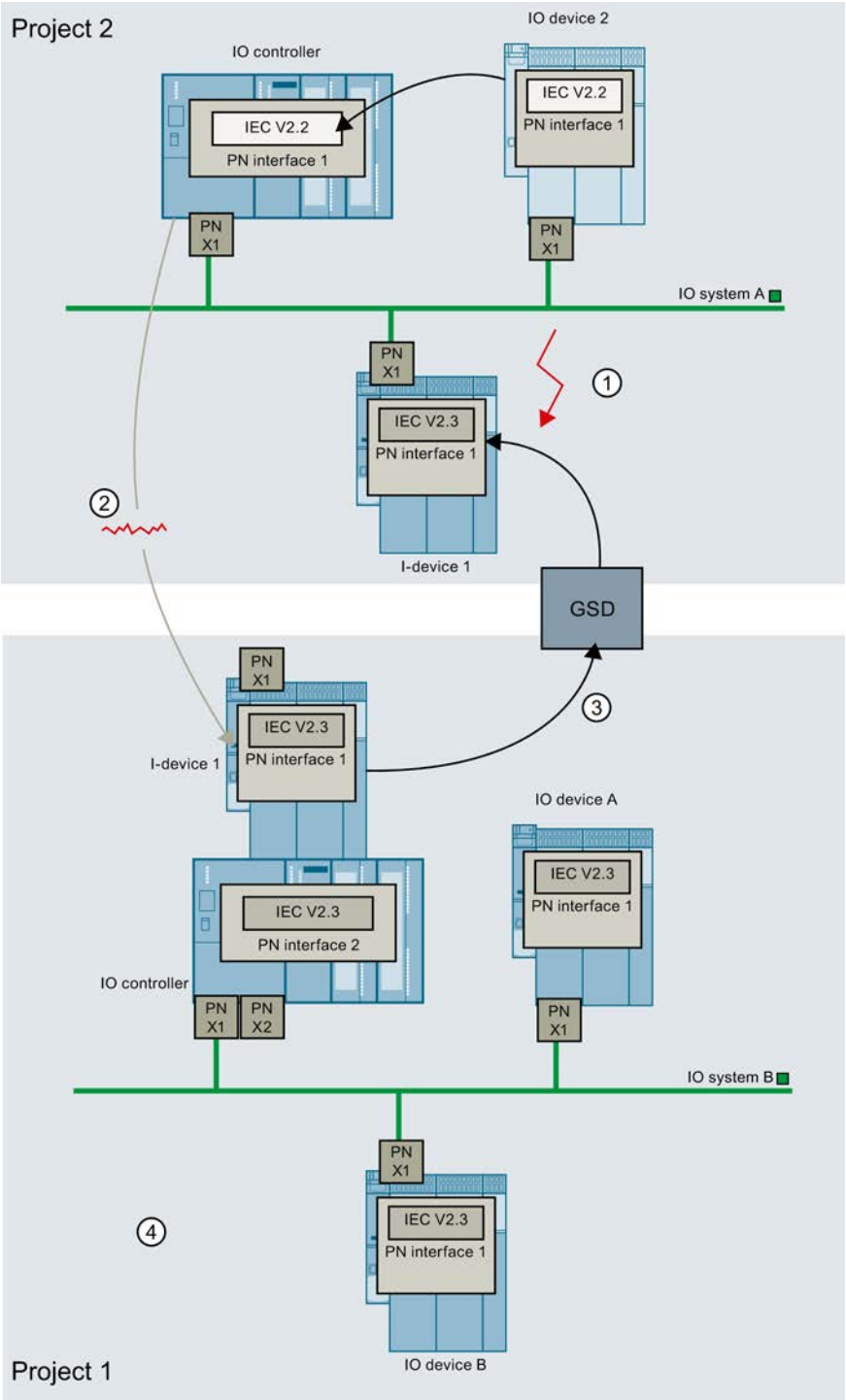
This means you must set a PROFINET interface in IO system B to V2.2 mode.

You set a PROFINET interface of a device in the lower-level IO system to V2.2 mode by executing the following steps.

1. Open the I-device project in STEP 7.
2. Select any device.
3. Enable the option "Use IEC V2.2 LLDP mode" in the area "Interface options".

Then generate a new GSD file and use the new GSD file.

Download the new I-device configuration to the I-device CPU.


Project 2
IO device 2
IO controller
IEC V2.2
PN interface 1
IEC V2.2
PN interface 1
PN X1
PN X1
IO system A
PN X1
IEC V2.3
PN interface 1
I-device 1
1
2
GSD
PN X1
IEC V2.3
PN interface 1
I-device 1
3
IO device A
IEC V2.3
PN interface 1
IEC V2.3
PN interface 2
IO controller
PN X1
PN X2
PN X1
IO system B
PN X1
IEC V2.3
PN interface 1
4
IO device B
Project 1

① The PROFINET interfaces have different settings in IO system A (I-device set to V2.3, IO controller set to V2.2, IO device 2 only supports V2.2).

② The settings of the PROFINET interfaces are stored in the I-device (no parameter assignment by the higher-level IO controller).

③ GSD file has entry that IEC V2.3 mode is set.

④ All devices are automatically set to V2.3 mode: This means you must set a PROFINET interface in project 1 manually to V2.2 mode.

Time-of-day functions

Basic principles of time of day functions

All S7-1500 CPUs are equipped with an internal clock. The backup supports the display of the correct time for up to 10 hours if the power supply is interrupted.

Time-of-day format

The clock always shows the time-of-day with a resolution of 1 millisecond and the date including the day of the week. The time adjustment for daylight saving time is also taken into account.

Setting and reading the time of day

Setting and reading the time with instructions

You can set, start and read the time and date on the CPU clock with the following instructions in the user program:

- Set the time: "WR_SYS_T"
- Read time "RD_SYS_T"
- Read local time "RD_LOC_T"
- Write local time "WR_LOC_T"

Manual setting

You can also read and set the time manually in the online and diagnostics view under "Functions > Set time".

See also

WR_SYS_T: Set time-of-day (Page 4508)

RD_SYS_T: Read time-of-day (Page 4509)

RD_LOC_T: Read local time (Page 4510)

WR_LOC_T: Write local time (Page 4511)

SET_TIMEZONE: Set time zone (Page 4513)

Time synchronization

Reference

Assigning parameters for the interface properties in the "Properties > General > DP interface > Time synchronization" parameter group.

Clock parameters

The clock parameters allow you to make the following settings:

- Synchronization type
  If you have set the synchronization mode to "As master", you can then select the time interval for automatic synchronization.
  If more than one module with a clock is present within a network, you need to specify which CPU will act as the master and which as the slave for time synchronization purposes. There can only be one time master.
- Update interval
  The update interval defines the interval between time queries.

Instructions for synchronizing the time

To ensure that the time is consistent in all modules in the network, the time slaves are updated by the system program at regular configurable intervals. Instruction "SNC_RTCB" allows the date and time to be transferred from the time master to the time slave independently of the synchronization interval configured.

Time-of-day synchronization in NTP mode

Reference

Assigning parameters for the interface properties in the "Properties > General > PROFINET interface > Time synchronization" parameter group. The "Enable time synchronization via NTP server" option is selected.

NTP mode (NTP: Network Time Protocol)

In NTP mode, the device sends time queries at regular intervals (in client mode) to the NTP server in the subnet (LAN). Based on the replies from the server, the most reliable and most accurate time is calculated and the time on the station is synchronized.

The advantage of this mode is that it allows the time to be synchronized across subnets.

The IP addresses of up to four NTP servers need to be configured. The update interval defines the interval between the time queries (in seconds). The value of the interval ranges between 10 seconds and one day.

In NTP mode, it is generally UTC (Universal Time Coordinated) that is transferred; this corresponds to GMT (Greenwich Mean Time).

Time synchronization parameters in NTP mode

The clock parameters allow you to make the following settings:

- Enable time synchronization via NTP server
  Select this check box if you want the internal clock to be synchronized using the NTP synchronization mode.
- Server 1-4
  The IP addresses of up to four NTP servers need to be configured.
- Update interval
  The update interval defines the interval between time queries.

Enabling system memory

System memory

A system memory is a bit memory with defined values.

You decide which memory byte of the CPU will become the system memory byte when assigning the system memory parameters.

Benefits

You can use system memory in the user program, for example to run program segments in only the first program cycle after start-up. Two system memory bits are constant 1 or constant 0.

Bits of the system memory bytes

The following table shows the meaning of the system memory:

Bit of the system memory bytes	7	6	5	4	3	2	1	0
Meaning	Reserved (=0)	Reserved (=0)	Reserved (=0)	Reserved (=0)	=0	=1	=1 with change to the diagnosis status	=1 in first program cycle after startup, otherwise 0

Note

The selected memory byte cannot be used for intermediate storage of data.

Using clock memory

Clock memory

A clock memory is a bit memory that changes its binary status periodically in the pulse-no-pulse ratio of 1:1.

You decide which memory byte of the CPU will become the clock memory byte when assigning the clock memory parameters.

Benefits

You can use clock memory, for example, to activate flashing indicator lamps or to initiate periodically recurring operations such as recording of actual values.

Available frequencies

Each bit of the clock bit memory byte is assigned a frequency. The following table shows the assignment:

Bit of the clock memory byte	7	6	5	4	3	2	1	0
Period (s)	2.0	1.6	1.0	0.8	0.5	0.4	0.2	0.1
Frequency (Hz)	0.5	0.625	1	1.25	2	2.5	5	10

Note

Clock memory runs asynchronously to the CPU cycle, i.e. the status of the clock memory can change several times during a long cycle.

The selected memory byte cannot be used for intermediate storage of data.

Protection

Setting options for the protection

Protection level

The following section describes how to use the various access levels of the S7-1500 CPUs.

S7-1500 CPUs provide various access levels to limit the access to specific functions.

The parameters for the access levels are assigned in a table. The green checkmarks in the columns to the right of the respective access level specify which operations are possible

without knowing the password of this access level. If you want to use the functions of check boxes that are not selected, a password has to be entered.

NOTICE
Configuring an access level does not replace know-how protection Configuring access levels offers a high degree of protection against unauthorized changes to the CPU by restricting download privileges to the CPU. However, blocks on the memory card are not write- or read-protected. Use know-how protection to protect the code of blocks on the memory card.

Default characteristics

The default access level is "Full access (no protection)". Every user can read and change the hardware configuration and the blocks. A password is not set and is also not required for online access.

The access levels in detail

With an S7-1500 CPU, you can configure the following access levels:

- Full access (no protection): The hardware configuration and the blocks can be read and changed by all users.
- Read access for F-blocks (F-CPUs only): F-blocks of the safety program cannot be changed without authorization by the password of this access level or a higher level.
  Further information is available in the programming and operating manual *SIMATIC Safety - Configuring and Programming.*
- Read access: With this access level, only read access to the hardware configuration and the blocks is possible without entering a password - meaning that you can load the hardware configuration and blocks into the programming device. In addition, HMI access and access to diagnostics data is possible.
  You cannot load blocks or a hardware configuration into the CPU without entering the password. Moreover, writing test functions and firmware updates are **not** possible without a password.

- HMI access: With this access level, only HMI access and access to diagnostics data is possible without entering the password.
  You can neither load blocks and the hardware configuration into the CPU nor load blocks and the hardware configuration from the CPU into the programming device without entering the password. In addition, the following is **not** possible without a password: Writing test functions, changing the operating state (RUN/STOP) and firmware updates.
- No access (complete protection): When the CPU is completely protected, no read or write access to the hardware configuration and the blocks is possible. HMI access is also not possible. The server function for PUT/GET communication is disabled in this access level (cannot be changed).
  Authorization with the password again provides you full access to the CPU.

Behavior of a password-protected module during operation

The CPU protection takes effect after the settings are downloaded to the CPU.

Before an online function is executed, the necessary permission is checked and, if necessary, the user is prompted to enter a password.

Example: The module was configured with read access and you want to execute the "Modify tags" function. This requires write access; therefore, the assigned password must be entered to execute the function.

The functions protected by a password can only be executed by one programming device/PC at any one time. Another programming device/PC cannot log on.

Access authorization to the protected data is in effect for the duration of the online connection or until the access authorization is manually rescinded with "Online > Delete access rights".

Each access level allows unrestricted access to certain functions without entering a password, for example, identification using the "Accessible devices" function.

Access to a password-protected S7-1500 CPU can be restricted locally on the display. The restriction is only in effect when the operating mode switch is set to RUN.

See also

Disabling online access to the CPU via the display (Page 1889)

Restriction of communication services (Page 1888)

Protection concept for project data (Page 378)

Configuring access levels (Page 1886)

Protecting blocks (Page 2929)

Printing project data (Page 397)

Configuring access levels

The following section describes how to configure an access level and enter passwords for an S7-1500 CPU.

For an S7-1500 CPU, you can enter multiple passwords and thereby set up different access rights for various user groups.

The passwords are entered in a table in such a way that exactly one access level is assigned to each password.

The effect of the password is given in the "Access level" column.

Example

You select the "No access (complete protection)" access level for a standard CPU (in other words, not an F-CPU) and enter a separate password for each of the access levels that lie above it in the table.

For users who do not know any of the passwords, the CPU is completely protected. Not even HMI access is possible.

For users who know one of the set passwords, the effect depends on the table row in which the password occurs:

- The password in row 1 (Full access (no protection)) allows access as if the CPU were completely unprotected. Users who know this password have unrestricted access to the CPU.
- The password in row 2 (read access) allows access as if the CPU were write-protected. Despite knowing the password, users who know this password only have read access to the CPU.
- The password in row 3 (HMI access) allows access as if the CPU were write-protected and read-protected so that only HMI access is possible for users who know this password.

Procedure

To configure the access levels of an S7-1500 CPU, follow these steps:

1. Open the properties of the module in the inspector window.
2. Open the "Protection" entry in the area navigation.
   A table with the possible access levels appears in the inspector window.

Protection level	Access			Access permission	
Protection	HMI	Read	Write	Password	Confirmation
Full access (no protection)	✓	✓	✓		
Read access	✓	✓			
HMI access	✓				
No access (complete protection)					

Figure 8-54 Access protection

3. Activate the required access level in the first column of the table. The green checkmarks in the columns to the right of the respective protection level show you which operations are still available without entering the password.

4. In the "Password" column, specify a password for the selected access level. In the "Confirmation" column, enter the selected password again to protect against incorrect entries.
   Ensure that the password is sufficiently secure, in other words, that is does not follow a machine-recognizable pattern!
   You must enter a password in the first row "Full access (no protection)". This enables unrestricted access to the CPU for those who know the password, regardless of the selected protection level.
5. Assign additional passwords as needed to other access levels if the selected protection level allows you to do so.
6. Download the hardware configuration so that the access level takes effect.

Result

The hardware configuration and the blocks are protected against unauthorized access according to the set access level. If an operation cannot be executed without a password due to the set access level, a dialog for entering a password is displayed.

See also

Setting options for the protection (Page 1884)

Restriction of communication services

Introduction

The CPU can be the server for a number of communication services. This means that other communication participants can access CPU data even if you have not configured and programmed connections for the CPU.

The local CPU as a server thus does not have the possibility to control communication to the clients.

The parameter "Connection mechanisms" in the "Protection" area of the CPU parameters is used to specify whether this type of communication is permitted or not for the local CPU during operation.

Permit access with PUT/GET communication from remote partners

By default, the "Permit access with PUT/GET communication from remote partners (...)" option is disabled. In this case, read and write access to CPU data is only possible for communication connections that require configuration or programming both for the local CPU and for the communication partner. Access through BSEND/BRCV instructions is possible, for example.

Connections for which the local CPU is only a server (meaning that no configuration/ programming of the communication with the communication partner exists at the local CPU), are therefore not possible during operation of the CPU, for example,

- For PUT/GET, FETCH/WRITE or FTP access via communication modules
- For PUT/GET access from other S7 CPUs
- For HMI access that is realized via PUT/GET communication

If you want to allow access to CPU data from the client side, meaning that you do not want to restrict the communication services of the CPU, activate the "Permit access with PUT/GET communication from remote partners" option.

Disabling online access to the CPU via the display

You can block access to a password-protected CPU on the display of an S7-1500 CPU (local block).

The access block only functions if the operating mode switch is set to RUN.

The access block is effective irrespective of the password protection. This means that even if someone accesses the CPU via a connected programming device and has entered the correct password, access to the CPU remains disabled.

The access block can be set separately for each protection level on the display, so that, for example, read access is allowed locally, but write access is not allowed locally.

In addition, a password can be configured for the display in the properties of the CPU so that the local access protection is protected by a local password.

Procedure

Proceed as follows to set the local access protection for an S7-1500 CPU on the display:

1. Select the menu Settings > Protection on the display.
2. Confirm with "OK" and set for each protection level whether access in RUN mode is permitted:
   Allow: Access to the CPU is possible with password input.
   Block in RUN: If the operating mode switch is set to RUN, no user can log on to the CPU with the rights of this protection level, even when the relevant password is known. In STOP mode, access is possible with password input.

See also

Setting options for the protection (Page 1884)

Test, commissioning and routing

Test, commissioning, routing

Select the check box if, for example, you want to use PG functions regularly over this interface for commissioning and testing.

If you select the "Test, commissioning, routing" option, the interface becomes the active device on PROFIBUS (in other words, the interface is included in the token rotation of PROFIBUS). The following functions are then possible:

- Programming (for example loading)
- Testing (status/modify)
- S7 routing (I-slave as gateway)

This can extend the token rotation time. In time-critical applications and when S7 routing and client functionality are not required for communication, this option should therefore not be enabled.

If the "Test, commissioning, routing" check box is not selected, the interface functions only as a server for communication services.

Server functionality means that only the partner (for example PG, OP or automation system) can initiate communication. Note the following features when the "Test, commissioning, routing" option is not enabled:

- S7 routing is not possible
- Programming and testing are possible but slow (no extension of the bus token rotation time)

Organization blocks

OB start information

Structure of the OB start information

When you configure the properties of an OB for S7-1500 CPUs, you also specify the structure of the OB start information:

- As for S7-300 and S7-400 CPUs
- Optimized start information

OB start information as for S7-300 and S7-400 CPUs

The OB start information is part of the local data and has the same structure as for S7-300 and S7-400 CPUs.

Optimized start information

The optimized start information is a set of input tags.

It only provides the information that is necessary to identify the OB start event.

Semantically identical information is displayed for various OBs under the same tag name within the optimized start information.

Description of all types of organization blocks

Startup OBs

Description

The operating system calls each startup OB once at the transition from "STOP" to "RUN" mode. If several startup OBs are available, they are called in the order of their OB numbers, starting with the OB that has the lowest OB number.

The entirety of all startup OBs is referred to as the startup routine.

You can define default settings for your cyclic program in the startup routine.

All values of the process image input have the value zero here. There is no time limit for executing the startup routine. Time-driven or interrupt-driven organization blocks cannot be used.

After the startup routine has been completed, the operating system reads in the process image input and starts the cyclic program.

Structure of the start information

A startup OB has the following start information:

- As for S7-300 and S7-400 CPUs
  See Startup organization blocks (OB 100, OB 101 and OB 102) (Page 1807)
  In contrast to the assignment of start information with S7-300 and S7-400-CPUs:
  - OB_NUMBR=B#16#FF, if the OB number > 254
- Optimized start information:

Name	Data type	Meaning
LostRetentive	BOOL	=TRUE, if the contents of retentive data areas have been lost
LostRTC	BOOL	=TRUE, if the time of the real-time clock has been lost

See also

Events and OBs (Page 1853)

OB start information (Page 1890)

Cyclic OBs

Description

The essential basis of the user program is the cyclic program. Its associated program parts are processed cyclically: When the processing has been completed, the operating system starts processing it again.

One or more cycle OBs belong to the cyclic program. These have priority 1. This is the lowest priority of all OBs. The cyclic program can be interrupted by events of any other event class.

When you have created several program cycle OBs, these are called one after the other in the order of their OB numbers. The program cycle OB with the lowest OB number is called first.

The following events start the cyclic program:

- End of startup processing
- End of the processing of the previous run of the cyclic program

After the cyclic program is complete, the operating system updates the process images as follows:

1. It writes the values from the process image output to the output modules.
2. It reads the inputs at the input modules and transfers these to the process image input.

Cycle time monitoring

Cycle time refers to the runtime of the cyclic program, including the runtime of all nested program parts of higher-priority OBs. If you have created several program cycle OBs, each program cycle OB contributes to the cycle time.

The operating system monitors whether the cycle time remains smaller than the configured maximum cycle time. If the cycle time exceeds the maximum cycle time, OB 80 is called (start event W#16#0002:3501). You can prevent this by restarting time monitoring at an appropriate point in program by calling the RE_TRIGR instruction. If the maximum cycle time is exceeded a second time within a cycle (without restarting time monitoring), the CPU switches to STOP mode (event W#16#0002:3500).

Minimum cycle time

Apart from monitoring the maximum cycle time, it is also possible to guarantee a minimum cycle time. To do this, the operating system delays the start of a new cycle until the minimum cycle time has been reached.

Structure of the start information

- As for S7-300 and S7-400 CPUs
  See Cyclic program (OB 1) (Page 1754)
  In contrast to the assignment of start information with S7-300 and S7-400-CPUs:
  - SCAN_1=B#16#01 at the first call and B#16#03 at each further call
  - OB_NUMBR=B#16#FF, if the OB number > 254

- Optimized start information:

Name	Data type	Meaning
Initial_Call	BOOL	=TRUE in the first call of this OB at: • Transition from STOP or HOLD to RUN • After reloading
Retentivity	BOOL	=TRUE, if retentive data is available

See also

Events and OBs (Page 1853)

Evaluating the value status (Page 1846)

OB start information (Page 1890)

Time-of-day interrupt OBs

Description

The S7-1500 CPUs provide time interrupt OBs, which can be started as follows:

- Once at a specific time (date with time)
- Periodically (the following time intervals are possible in this case: every minute, hourly, daily, weekly,
monthly, end of month, yearly)

Starting a time interrupt OB

The following requirements must be met in order for the CPU to start a time interrupt OB:

- You must have previously set and activated the associated time interrupt. (Setting a time interrupt means specifying its start time and its duration.)
- You must have loaded the time interrupt OB to your CPU.

To set and activate a time interrupt, you have the following three options:

- You set and activate the time interrupt with STEP 7.
- You set the time interrupt with STEP 7 and activate it by calling the ACT_TINT instruction from your program.
- You set the time interrupt by calling the SET_TINTL instruction and activate it by calling the ACT_TINT instruction from within your program.

Rules for time interrupts

- If you set a time interrupt in such a way that the corresponding OB is to be processed once, the start time must not be in the past (relative to the real-time clock of the CPU).
- If you set a time interrupt in such a way that the corresponding OB is to be processed periodically, but the start time is in the past, then the time interrupt OB is processed the next time it is due according to the current time.

- The date of periodic time interrupts must correspond to a real date. For example, the monthly repetition of a time interrupt OB with start date 1/31 is therefore not possible. In this case, an OB is only started in the months that have 31 days.
- A time interrupt activated during startup is not executed until the startup has been completed.
- A startup deletes all time interrupts that were set and activated by an instruction in the user program.
- In the event of a transition from HOLD to RUN, the operating system checks whether there are elapsed time interrupts. If so, it calls the time error OB. If you have selected the same start information for it as for S7-300 and S7-400 CPUs, it enters whether time interrupts have been missed for the OBs 10 to 17 (and only for these). After the processing of OB 80 has been completed, the first missed OB is executed subsequently.

Behavior of time interrupts when time is adjusted ahead or back

The characteristics of the time interrupts depend on the time difference by which the time is set ahead or back. This is explained in more detail in the following table.

Changing the time	Effect on time interrupt
< 20 s	• When the time is set ahead: Every skipped time interrupt is executed subsequently. The start information tag "CaughtUp" is set. OB 80 is not called. • When the time is adjusted back: A time interrupt that was already executed in the reset time range or is currently active is not repeated.
> > = 20 s	• When the time is set ahead: If one or more time interrupts were skipped, OB 80 is called. For each priority, the start information provides information on whether at least one time interrupt was skipped. If you do not delete the time interrupt in OB 80, the first skipped time interrupt is executed subsequently and the start information tag "CaughtUp" is set. • When the time is adjusted back: The time interrupts in the range of the reset time are repeated. If the time correction is greater than the duration of a time interrupt, the operating system calculates the time of the first time interrupt to be repeated. In the case of a repeated time interrupt, the start information tag "SecondTime" is set.

Changing the time	Effect on time interrupt
Changing from standard to daylight saving time and vice versa	This change is only effective if you use local time. • When there is a change from standard to daylight saving time, a skipped time interrupt is executed subsequently and the start information tag "CaughtUp" is set. OB 80 is not called. • When there is a change from daylight saving time to standard time: the time interrupts in the range of the reset time are repeated. In the case of a repeated time interrupt, the start information tag "SecondTime" is set.
Changing the time zone (results in change of the local time)	The start time of the active time interrupts is recalculated. • When the local time is adjusted ahead, a skipped time interrupt is executed subsequently and the start information tag "CaughtUp" is set. OB 80 is not called. • When the local time is adjusted back, time interrupts that have already been executed may be repeated. In the case of repeated time interrupts, the start information tag "SecondTime" is set.

Time monitoring

When working with time interrupts, the following events can occur due to time monitoring by the operating system:

- W#16#0002:3504
  The time was adjusted ahead by the change from standard to daylight saving time. The start time of time interrupts was skipped during this process. OB 80 is called.
- W#16#0002:3505
  The time was adjusted ahead. The start time of time interrupts was skipped during this process. OB 80 is called.
- W#16#0002:3507
  Buffer overflow for OB start events. An entry is made in the diagnostic buffer.
- W#16#0002:3502
  Number of pending OB start events violates a preset limit. OB 80 is called.

Structure of the start information

- As for S7-300 and S7-400 CPUs
  See Time-of-day interrupt organization blocks (OB 10 to OB 17) (Page 1755)
  In contrast to the assignment of start information with S7-300 and S7-400-CPUs:
  - STRT_INF=B#16#1F, if the OB number > 122
  - OB_NUMBR=B#16#FF, if the OB number > 254
- Optimized start information:

Name	Data type	Meaning
CaughtUp	BOOL	=TRUE if an OB call was executed subsequently because time was adjusted ahead
SecondTime	BOOL	=TRUE if an OB was called a second time because time was adjusted back. Note: SecondTime is set only once in the cases where time is adjusted back.

See also

Parameter assignment for time-of-day interrupt OBs (Page 1916)

Events and OBs (Page 1853)

OB start information (Page 1890)

Status interrupt OB

Description

When it receives a status interrupt, the operating system of the S7-1500 CPU calls the status interrupt OB. This may be the case if a module of a slave changes its operating state, for example, from "RUN" to "STOP". For more detailed information on events that trigger a status interrupt, refer to the documentation of the slave or device manufacturer.

Structure of the start information

- As for S7-300 and S7-400 CPUs
  See Status interrupt OB (OB 55) (Page 1764)
- Optimized start information:

Name	Data type	Meaning
LADDR	HW_IO	Hardware identifier of the component triggering the interrupt
Slot	UINT	Slot number of the component triggering the interrupt
Specifier	WORD	Interrupt specifier from the interrupt frame: • PROFIBUS: – Bit 0 to 1: Interrupt specifier – Bit 2: Add_Ack – Bit 3 to 7: Seq. no. – Bit 8 to 15: Reserved • Central and PNIO: – Bits 0 to 10: Sequence number (value range 0 to 2047) – Bit 11: Channel diagnostics: 0: No channel diagnostics available 1: Channel diagnostics available – Bit 12: Status of manufacturer-specific diagnostics: 0: No manufacturer-specific status information available 1: Manufacturer-specific status information available – Bit 13: Status of diagnostics for submodule: 0: No status information available, all errors eliminated 1: Either channel diagnostics or status information (or both) available – Bit 14: Reserved – Bit 15: Application relationship diagnostics state: 0: None of the modules configured within this AR are reporting diagnostic information 1: At least one of the modules configured in this AR is reporting diagnostic information

See also

Events and OBs (Page 1853)

OB start information (Page 1890)

Update interrupt OB

Description

When it receives an update interrupt, the operating system of the S7-1500 CPU calls the update interrupt OB. This may be the case if you changed a parameter on a slot of a slave or device. For more detailed information on events that trigger an update interrupt, refer to the documentation of the slave or device manufacturer.

Structure of the start information

- As for S7-300 and S7-400 CPUs
  See Update interrupt OB (OB 56) (Page 1768)
- Optimized start information:

Name	Data type	Meaning
LADDR	HW_IO	Hardware identifier of the component triggering the interrupt
Slot	UINT	Slot number of the component triggering the interrupt
Specifier	WORD	Interrupt specifier from the interrupt frame: • PROFIBUS: – Bit 0 to 1: Interrupt specifier – Bit 2: Add_Ack – Bit 3 to 7: Seq. no. – Bit 8 to 15: Reserved • Central and PNIO: – Bits 0 to 10: Sequence number (value range 0 to 2047) – Bit 11: Channel diagnostics: 0: No channel diagnostics available 1: Channel diagnostics available – Bit 12: Status of manufacturer-specific diagnostics: 0: No manufacturer-specific status information available 1: Manufacturer-specific status information available – Bit 13: Status of diagnostics for submodule: 0: No status information available, all errors eliminated 1: Either channel diagnostics or status information (or both) available – Bit 14: Reserved – Bit 15: Application relationship diagnostics state: 0: None of the modules configured within this AR are reporting diagnostic information 1: At least one of the modules configured in this AR is reporting diagnostic information

See also

Events and OBs (Page 1853)

OB start information (Page 1890)

OB for manufacturer- or profile-specific interrupt

Description

When it receives a manufacturer- or profile-specific interrupt, the operating system of the S7-1500 CPU calls OB 57. For more detailed information on events that trigger this type of interrupt, refer to the documentation of the slave or device manufacturer.

Structure of the start information

- As for S7-300 and S7-400 CPUs
  See OB for manufacturer-specific alarms (OB 57) (Page 1772)
  In contrast to the assignment of start information with S7-300 and S7-400-CPUs:
  - STRT_INF=B#16#67
- Optimized start information:

Name	Data type	Meaning
LADDR	HW_IO	Hardware identifier of the component triggering the interrupt
Slot	UINT	Slot number of the component triggering the interrupt
Specifier	WORD	Interrupt specifier from the interrupt frame: • PROFIBUS: – Bit 0 to 1: Interrupt specifier – Bit 2: Add_Ack – Bit 3 to 7: Seq. no. – Bit 8 to 15: Reserved • Central and PNIO: – Bits 0 to 10: Sequence number (value range 0 to 2047) – Bit 11: Channel diagnostics: 0: No channel diagnostics available 1: Channel diagnostics available – Bit 12: Status of manufacturer-specific diagnostics: 0: No manufacturer-specific status information available 1: Manufacturer-specific status information available – Bit 13: Status of diagnostics for submodule: 0: No status information available, all errors eliminated 1: Either channel diagnostics or status information (or both) available – Bit 14: Reserved – Bit 15: Application relationship diagnostics state: 0: None of the modules configured within this AR are reporting diagnostic information 1: At least one of the modules configured in this AR is reporting diagnostic information

See also

Events and OBs (Page 1853)

OB start information (Page 1890)

Time-delay interrupt OBs

Description

A time-delay interrupt OB is started after a configurable time delay of the operating system. The time delay begins to elapse when the "SRT_DINT" instruction is called (this is the start of the associated time-delay interrupt).

Before a time-delay interrupt OB can be called by the operating system, the following conditions must be met:

- The time-delay interrupt must be started by calling the "SRT_DINT" instruction.
- The time-delay interrupt OB must not be deselected during configuration.

The time-delay interrupt OB must exist in the CPU.

Influencing time-delay interrupts with instructions

You can use the "CAN_DINT" instruction to cancel a time-delay interrupt which has already been started and whose associated time-delay interrupt OB has not yet been called by the operating system and use the "SRT_DINT" instruction to restart it (and thereby assign a different delay time to it). In addition, you can use the "QRY_DINT" instruction to query the current status of a time-delay interrupt.

You can disable and re-enable time-delay interrupts using the "DIS_AIRT" and "EN_AIRT" instructions.

Note

If you block this interrupt with "DIS_AIRT" after the execution of "SRT_DINT", processing is only continued after enabling with "EN_AIRT". This means that the delay time is extended accordingly.

Calling error OBs in connection with time-delay interrupt OBs

If the delay time of a time-delay interrupt expired before the processing of the associated time-delay interrupt OB was completed, the operating system generates a time error (calls OB 80).

Processing of time-delay interrupts depending on operating mode

"SRT_DINT" can be called and the delay time can elapse in both startup and in RUN mode. Time-delay interrupt OBs, however, are only processed in RUN mode. If a time-delay interrupt starts in a startup OB and the delay time has expired before the CPU reaches RUN mode, the call of the corresponding time-delay interrupt OB is delayed until the CPU is in RUN mode.

A warm or cold restart clears all start events for a time-delay interrupt OB.

Structure of the start information

- As for S7-300 and S7-400 CPUs
  See Time-delay interrupt organization blocks (OB 20 to OB 23) (Page 1759)
  In contrast to the assignment of start information with S7-300 and S7-400-CPUs:
  - STRT_INF=B#16#2F, if the OB number > 122
  - OB_NUMBR=B#16#FF, if the OB number > 254

- Optimized start information:

Name	Data type	Meaning
Sign	WORD	User ID: Input parameter SIGN from the call of the "SRT_DINT" instruction

See also

SRT_DINT: Start time-delay interrupt (Page 4706)

Events and OBs (Page 1853)

OB start information (Page 1890)

Cyclic interrupt OBs

Description

You can start program parts at equidistant time intervals with the help of a cyclic interrupt OB.

The start times of a cyclic interrupt OB result as follows from its time interval and its phase offset:

Start times = n * time interval + phase offset (n = 0, 1, 2 ...)

- The time interval is the period of time between two calls. It is an integer multiple of the basic clock cycle of 1 µs.
- The phase offset is the time interval by which the start times are offset compared to the multiple of the time interval.
  A phase offset may be advisable when you are using multiple cyclic interrupt OBs. When their time intervals have a common multiple, for example, you can use a phase offset to prevent simultaneous start times.

Example of the use of a phase offset

You are using two cyclic interrupt OBs in your program: one OB with a time interval of 20 ms and one with a time interval of 100 ms. Use a phase offset to ensure that these are not called simultaneously at integer multiples of 100 ms.

Call of error OBs in connection with cyclic interrupt OBs

The runtime of a cyclic interrupt OB must be significantly smaller than its time interval. Otherwise, it could still happen that the time interval of a cyclic interrupt has expired but the corresponding cyclic interrupt OB is still being processed. In this case, the operating system generates a time error (calls OB 80). The cyclic interrupt that caused the error is executed later or discarded. However, only one cyclic interrupt OB for each priority can be executed later.

Structure of the start information

- As for S7-300 and S7-400 CPUs
  See Cyclic interrupt organization blocks (OB 30 to OB 38) (Page 1760)
  In contrast to the assignment of start information with S7-300 and S7-400-CPUs:
  - STRT_INF=B#16#3F, if the OB number > 122 and the phase offset and cycle time are specified in ms
  - OB_NUMBR=B#16#FF, if the OB number > 254
- Optimized start information

Name	Data type	Meaning
Initial_Call	BOOL	=TRUE in the first call of this OB • At the transition from STOP or HOLD to RUN • After reloading
Event_count	INT	Number of discarded start events since the last start of this OB

See also

Assigning parameters to cyclic interrupt OBs (Page 1917)

Events and OBs (Page 1853)

OB start information (Page 1890)

Hardware interrupt OBs

Description

You can use hardware interrupt OBs to react to those events of modules with hardware interrupt capability which trigger a hardware interrupt. You define which events these are by means of configuration.

You can assign only one hardware interrupt OB to each event that triggers a hardware interrupt. Multiple events, however, can be assigned to one hardware interrupt OB (in extreme cases, all events of a module triggering a hardware interrupt).

For S7-1500 modules, individual channels can trigger hardware interrupts (e.g., a binary input on a rising edge).

Functionality of a hardware interrupt OB

After a module or submodule has triggered a hardware interrupt, the operating system determines the associated event and the associated hardware interrupt OB. If the currently active OB has a lower priority than this hardware interrupt OB, the hardware interrupt OB is started. Otherwise, the hardware interrupt OB is positioned in the queue that corresponds to its priority. The hardware interrupt is acknowledged after the completion of the corresponding hardware interrupt OB.

If another process event occurs on the same module or submodule during the time between identification and acknowledgement of a hardware interrupt, the following applies:

- If the event occurs on the channel that triggered the current hardware interrupt, the relevant hardware interrupt is lost. Another hardware interrupt can only be triggered by this channel when the current hardware interrupt is acknowledged.
- If the event occurs on a different channel, a hardware interrupt is triggered.

There may be additional restrictions for distributed I/O: for example, that a module or submodule can trigger only one interrupt of an interrupt type until the current interrupt has been acknowledged.

Hardware interrupt OBs are called only in RUN mode of the CPU.

Structure of the start information

- As for S7-300 and S7-400 CPUs: The USI, IChannel and EventType tags (explanation can be found in the optimized start information) are copied to "OB40_POINT_ADDR".
  See Hardware interrupt organization blocks (OB 40 to OB 47) (Page 1762)
  In contrast to the assignment of start information with S7-300 and S7-400-CPUs:
  - STRT_INF=B#16#41
  - OB_NUMBR=B#16#FF, if the OB number > 254

- Optimized start information:

Name	Data type	Meaning
LADDR	HW_IO	Hardware identifier of the module that triggers the hardware interrupt
USI	WORD	Identifier for future extensions (not user-relevant)
IChannel	USINT	Number of the channel that triggered the hardware interrupt
EventType	BYTE	Identifier for the event type to which the event that triggered the interrupt belongs (e.g., rising edge) This identifier can be found in the description of the respective module.

See also

Assigning parameters to hardware interrupt OBs (Page 1918)

Events and OBs (Page 1853)

Example of a hardware interrupt event (Page 1856)

What you need to know about HW identifiers (Page 883)

OB start information (Page 1890)

Isochronous mode interrupt OBs

Description

Isochronous mode interrupt OBs are used to start subprograms isochronously to the DP cycle or PN send clock.

You update the associated process image partition of the inputs and the associated process image partition of the outputs by calling the instructions SYNC_PI and SYNC_PO in the isochronous mode interrupt OB.

CAUTION

Direct access

In cases of direct access, avoid access to I/O areas to which process image partitions with isochronous mode interrupt OB connection are assigned.

Note

Effectiveness of the OB parameter "Report event overflow into diagnostic buffer"

When you select the check box "Report event overflow into diagnostic buffer" for the OB parameters, this does not become effective until you have loaded the OB properties to the CPU and the CPU has then performed a STOP-RUN transition.

Structure of the start information

- As for S7-300 and S7-400 CPUs
  See Synchronous cycle interrupt OBs (OB 61 to OB 64) (Page 1776)
  In contrast to the assignment of start information with S7-300 and S7-400-CPUs:
  - STRT_INF=B#16#6F, if the OB number > 122
  - OB_NUMBR=B#16#FF, if the OB number > 254

- Optimized start information:

Name	Data type	Meaning
Initial_Call	BOOL	=TRUE in the first call of this OB during transition from STOP to RUN
PIP_Input	BOOL	TRUE: The associated process image input is up-to-date
PIP_Output	BOOL	TRUE: The associated process image output was transferred to the outputs in good time after the last cycle
IO_System	USINT	Number of the distributed I/O system triggering the interrupt
Event_Count	INT	• = n: Number of lost cycles • = -1: An unknown number of cycles has been lost (e.g., because cycle has changed).
SyncCycleTime	LTIME	Calculated cycle time

See also

Assigning parameters to isochronous mode interrupt OBs (Page 1919)

Events and OBs (Page 1853)

OB start information (Page 1890)

MC-servo OB

Description

When you create a technology object for S7-1500 Motion Control, the organization block "MC-Servo [OB91]" for processing the technology objects is created automatically. The motion control functionality of the technology objects creates its own priority class and is called in the SIMATIC S7-1500 execution system according to the application cycle.

The MC-Servo OB is write-protected. The contents cannot be changed.

The position control algorithms of all technology objects configured for motion control on the CPU are calculated within the MC-Servo OB.

You can set the application cycle and the priority of the organization block in accordance with your requirements for control quality and system load.

Application cycle

You can set the application cycle in which the MC-Servo OB is called in the properties of the organization block:

- Synchronous to the BUS
  The MC-Servo OB is called synchronously with a bus system. You set the send clock in the properties of the selected bus system. You can select the following bus systems:
  - Isochronous PROFIBUS DP
  - Isochronous PROFINET IO

  You cannot call the MC-Servo OB synchronously with a bus system that is connected to the CPU via a communication processor/communication module (CP / CM).
- Cyclical
  The MC-Servo OB is called cyclically with the specified application cycle.

The selected application cycle must be long enough to allow all the technology objects for motion control to be processed in one cycle. If the application cycle is not observed, overflows occur. The CPU switches to STOP mode in the case of an overflow of the MC-Servo OB.

Clock reduction (CPU V1.5 and higher)

You can reduce the application cycle of the MC-Servo OB relative to the send clock of an isochronous PROFINET IO system. You can set an integer multiple of the send clock as the factor. Values up to 14 times the send clock (maximum 32 ms) are possible for the application cycle.

If you call a synchronous mode interrupt OB and the MC-Servo OB synchronously with the same PROFINET IO system, you must set the same application cycle for both organization blocks.

See also

Assigning the MC-Servo OB parameters (Page 1921)

Events and OBs (Page 1853)

MC-interpolator OB

Description

When you create a technology object for S7-1500 Motion Control, the organization block "MC-Interpolator [OB91]" for processing the technology objects is created automatically. The motion control functionality of the technology objects creates its own priority class and is called in the SIMATIC S7-1500 execution system according to the application cycle of the MC-Servo OB.

The MC-Interpolator OB is write-protected. The contents cannot be changed.

The evaluation of the motion control instructions as well as the monitoring and the setpoint generation for all technology objects configured on the CPU (motion control) is performed within the MC-Interpolator OB.

The MC-Interpolator OB is called at the end of processing of the MC-Servo OB. The clock ratio between MC-Interpolator OB and MC-Servo OB is always 1:1.

Process response

You must select the application cycle of the MC-Servo OB long enough to allow all the technology objects for motion control to be processed in one application cycle. If the application cycle is not observed, overflows occur. Note the following response to overflows:

- The CPU tolerates a maximum of three consecutive overflows of the MC-Interpolator OB.
- The processing of an MC-Interpolator OB can be interrupted by the calling of the MC Servo OB, at most.

If more overflows or interruptions occur, the CPU switches to STOP mode.

See also

Assigning the MC-Servo OB parameters (Page 1921)

Events and OBs (Page 1853)

Time error OB

Description

The S7-1500 CPU operating system calls the time error OB if one of the following errors occurs:

- First violation of the cycle time within a cycle

  Note

  If the cycle time is violated twice in the same cycle, the CPU goes to STOP mode. You can prevent this by calling the "RE_TRIGR" instruction at a suitable point.

- The number of requested but not completely processed OB requests has reached the configured warning limit. (For migrated programs, this warning limit set to 1.)
  If you have set the warning limit to 1, this means: If a start event, e.g., for a cyclic interrupt OB, occurs before the previous processing of the same OB has been completed, the operating system calls OB 80.
  This applies to all events whose associated OB can cause a time error.
- Expired time interrupt on re-entry into RUN mode after the CPU was, for example, in HOLD
- Skipped time interrupt due to the time being adjusted more than 20 s ahead (also in the case of changeover from standard time to daylight saving time)

Structure of the start information

- As for S7-300 and S7-400 CPUs
  See Time error OB (OB 80) (Page 1777)
- Optimized start information

Name	Data type	Meaning
Fault_ID	BYTE	Error code. Possible values: • B#16#01: Cycle time exceeded • B#16#02: The requested OB is still being processed. • B#16#05: Expired time interrupt due to time jump • B#16#06: Expired time interrupt upon re-entry into RUN after HOLD • B#16#07: Overflow of the OB request buffer for the current priority class • B#16#08: Isochronous mode interrupt - time error • B#16#09: Interrupt loss due to high interrupt load • B#16#0B: Technology synchronization interrupt - time error
Csg_OBnr	OB_ANY	Number of the OB that was being processed when the time error occurred
Csg_Prio	UINT	Priority of the OB that was being processed when the time error occurred

See also

Events and OBs (Page 1853)

OB start information (Page 1890)

Diagnostic interrupt OB

Description

The operating system of the S7-1500 CPU calls the diagnostic interrupt OB in the following cases:

- A diagnostics-capable S7-1500 module detects a change in its diagnostic status and sends a diagnostic interrupt request to the CPU.
- A diagnostics-capable S7-300 or S7-400 module for which you have enabled the diagnostic interrupt detects a change in its diagnostic status and sends a diagnostic interrupt request to the CPU.
- An event occurs that was triggered by an error in the power supply or the battery backup.

  Note

  With the S7-300 and S7-400 CPUs, such an event causes the power supply error OB to be called.

- The operating system has detected memory errors.

  Note

  With the S7-300 and S7-400 CPUs, such an event causes the CPU hardware error OB to be called.

Structure of the start information

- As for S7-300 and S7-400 CPUs
  See Diagnostic interrupt OB (OB 82) (Page 1783)
- Optimized start information:

Name	Data type	Meaning
IOstate	WORD	Status of the hardware object: • Bit 0: Good • Bit 1: Deactivated • Bit 2: Maintenance required • Bit 3: Maintenance demanded • Bit 4: Error • Bit 5: Not available • Bit 6: Qualified • Bit 7: Not available
LADDR	HW_ANY	Hardware identifier of the hardware object that triggered the diagnostic interrupt
Channel	UINT	Channel number
MultiError	BOOL	=TRUE, if there is more than one error

See also

Events and OBs (Page 1853)

OB start information (Page 1890)

Pull/plug interrupt OB

Description

The S7-1500 CPU operating system calls the pull/plug interrupt OB if a configured and non-disabled distributed I/O module or submodule is removed or inserted.

Note

The removal or insertion of a central module leads to the STOP of the CPU.

Structure of the start information

- As for S7-300 and S7-400 CPUs
  See Insert/remove module interrupt organization block (OB 83) (Page 1786)
- Optimized start information:

Name	Data type	Meaning
LADDR	HW_IO	Hardware identifier of the affected module or submodule
Event_Class	BTYE	• B#16#38: (Sub)module plugged • B#16#39: (Sub)module pulled or not responding
Fault_ID	BYTE	Error code (possible values: B#16#51, B#16#54, B#16#55, B#16#56, B#16#57, B#16#58, B#16#61, B#16#63, B#16#64, B#16#65, B#16#66)

The following table shows which event caused the start of the pull/plug interrupt OB.

ev_class B#16# ...	fault_id B#16# ...	Meaning
39	51	PROFINET IO module removed
39	54	PROFINET IO submodule removed
38	54	PROFINET IO submodule inserted and matches configured submodule
38	55	PROFINET IO submodule inserted, but does not match configured submodule
38	56	PROFINET IO submodule inserted; but error in module parameter assignment
38	57	PROFINET IO submodule or module inserted, but with a problem or maintenance
38	58	PROFINET IO submodule, access error corrected
39	61	Module removed or not responding
38	61	Module inserted, module type OK
38	63	Module inserted but module type is incorrect
38	64	Module inserted but faulty (module ID cannot be read)
38	65	Module inserted, but error in module parameter assignment
39	66	Module not responding, load voltage error
38	66	Module responds again, load voltage error corrected

See also

Events and OBs (Page 1853)

OB start information (Page 1890)

Rack error OB

Description

The S7-1500 CPU operating system calls the OB 86 in the following cases:

- The failure of a DP master system or of a PROFINET IO system is detected (in the case of either an incoming or an outgoing event).
- The failure of a DP slave or of an IO device is detected (in the case of either an incoming or an outgoing event).
- You have disabled or enabled a DP slave or an IO device with the "D_ACT_DP" instruction.
- Failure of some of the submodules of a PROFINET I-device is detected.

Structure of the start information

- As for S7-300 and S7-400 CPUs
  See Rack failure organization block (OB 86) (Page 1796)
- Optimized start information:

Name	Data type	Meaning
LADDR	HW_Device	Hardware identifier of the defective hardware object
Event_Class	BYTE	• B#16#32: Activation of a DP slaves or an IO device • B#16#33: Deactivation of a DP slaves or an IO device • B#16#38: outgoing event • B#16#39: incoming event
Fault_ID	BYTE	Error code (possible values: B#16#C3, B#16#C4, B#16#C5, B#16#C6, B#16#C7, B#16#C8, B#16#C9, B#16#CA, B#16#CB, B#16#CC, B#16#CD, B#16#CE, B#16#CF, B#16#F8, B#16#F9)

The following table shows which event caused the start of OB 86.

ev_class B#16# ...	fault_id B#16# ...	Meaning
39	C3	Failure of a DP master system
39/38	C4	Failure/return of a DP slave
38	C5	Return of a DP slave; however, slave is faulty
38	C6	Expansion unit return, but error in module parameter assignment
38	C7	Return of a DP device, but there is error in module configuration
38	C8	Return of a DP device, but discrepancy between preset/actual configuration
32/33	C9	Activation/deactivation of a DP slave with the "D_ACT_DP" instruction
39	CA	Failure of a PROFINET IO system
39/38	CB	Failure/return of a PROFINET IO device
38	CC	Return of a PROFINET IO device with fault or maintenance
38	CD	Return of a PROFINET IO device, deviation between preset/actual configuration
38	CE	Return of a PROFINET IO device, error in module configuration
32/33	CF	Activation/deactivation of an IO device with the "D_ACT_DP" instruction
39/38	F8	Failure/return of some of the submodules of a PROFINET I-device
38	F9	Return of some of the submodules of a PROFINET I-device with a device configuration difference

See also

Events and OBs (Page 1853)

OB start information (Page 1890)

Programming error OB

Requirement for calling the programming error OB

Global error handling must be set if you want to enable the S7-1500 CPU to call the programming error OB.

Description

The S7-1500 CPU operating system calls the programming error OB if a programming error occurs during the processing of an instruction of the user program. The programming error OB is processed with the priority set for it.

Note

If you have set local error handling, the programmed error handling in the corresponding block becomes effective when a programming error occurs.

Structure of the start information

- As for S7-300 and S7-400 CPUs
  See Programming error organization block (OB 121) (Page 1811)
- Optimized start information:

Name	Data type	Meaning
BlockNr	UINT	Number of the block in which the programming error occurred
Reaction	USINT	• 0: Ignore errors • 1: Replace incorrect value • 2: Skip command • 3: Programmed error handling, triggered by an array access with invalid index, for example, or by an error during parameter supply of an FC or FB
Fault_ID	BYTE	Error code (possible values: B#16#00, B#16#03, B#16#04, B#16#05, B#16#20, B#16#21, B#16#22, B#16#23, B#16#24, B#16#25, B#16#26, B#16#27, B#16#28, B#16#29, B#16#2C, B#16#30, B#16#31, B#16#32, B#16#33, B#16#34, B#16#35, B#16#38, B#16#39, B#16#3A, B#16#3B, B#16#3C, B#16#3D, B#16#3E, B#16#3F, B#16#50, B#16#51, B#16#75, B#16#76, B#16#A1, B#16#A2)
BlockType	USINT	Type of block in which the error occurred: • OB: 1 • FC: 2 • FB: 3 • SFC: 4 • SFB: 5 • DB: 6

Name	Data type	Meaning
Area	USINT	Area in which the incorrect access occurred: • Local data: B#16#40 to 4E, 86, 87, 8E, 8F, C0 to CE • Process image input: B#16#01 • Process image output: B#16#02 • Technology DB: B#16#04 • I: B#16#81 • Q: B#16#82 • M: B#16#83 • DB: B#16#84, 85, 8A, 8B
DBNr	Block_DB	DB no. if AREA = DB or DI
Csg_OBNr	OB_ANY	OB number (121)
Csg_Prio	USINT	OB priority
Width	USINT	Type of access during which the error occurred: • Bit: B#16#00 • Byte: B#16#01 • Word: B#16#02 • DWord: B#16#03 • LWord: B#16#04

The following table shows the meaning of the error codes:

fault_ID B#16# ...	Meaning
00	Maximum nesting depth of block calls exceeded
03	You have used a NULL pointer to address an operand.
04	Unknown instruction
05	Stop instruction
20	The addressed string has incorrect length information.
21	BCD conversion error
22	Area length error when reading
23	Area length error when writing
24	Area error when reading
25	Area error when writing
26	Error in timer no.
27	Error in counter no.
28	Read access to a byte, word or double word with a pointer whose bit address is not zero
29	Read access to a byte, word or double word with a pointer, whose bit address is not zero
2C	You have used a NULL pointer to address an operand.
30	Write access to a write-protected global DB
31	Write access to a write-protected instance DB
32	DB number error when accessing a global DB
33	DB number error when accessing an instance DB

fault_ID B#16# ...	Meaning
34	Number error during FC call
35	Number error during FB call
38	The type-safe data block is not loaded.
39	The data block is not defined as type-safe.
3A	Access to a DB that is not loaded; the DB number is located in the permitted area
3B	DB does not exist
3C	Access to an FC that is not loaded; the FC number lies in the permissible area
3D	Access to an SFC that is not loaded; the SFC number lies in the permissible area
3E	Access to an FB that is not loaded; the FB number lies in the permissible area
3F	Access to an SFB that is not available; the SFB number lies in the permissible area
50	The type-safe data block is not loaded.
51	The data block is not defined as type-safe.
75	The maximum nesting depth for block calls was exceeded.
76	The maximum of available local data was exceeded.
A1	Write access to a write-protected tag
A2	Access to a tag that contains an invalid numerical value

See also

Basics of error handling (Page 2821)

Events and OBs (Page 1853)

Principles of local error handling (Page 2823)

OB start information (Page 1890)

I/O access error OB

Requirement for call of the I/O access error OB

Global error handling must be set to enable the S7-1500 CPU to call the I/O access error OB.

Description

The S7-1500-CPU operating system calls the I/O access error OB if an error occurs during direct access to the I/O data while an instruction of the user program is being executed. This, for example, is the case if a read error occurs during direct access to data of an input module. The I/O access error OB is processed with the priority assigned to it.

Note

If local error handling is set, the error handling characteristics programmed in the associated block take effect if an I/O access error occurs.

Structure of the start information

- As for S7-300 and S7-400 CPUs
  See I/O access error organization block (OB 122) (Page 1814)
- Optimized start information:

Name	Data type	Meaning
BlockNr	UINT	Number of the block in which the I/O access error occurred
Reaction	USINT	0: Ignore error, 1: Replace incorrect value, 2: Skip command
Fault_ID	BYTE	Error code: • B#16#42: I/O access error, reading • B#16#43: I/O access error, writing
BlockType	USINT	Type of block in which the error has occurred: • OB: 1 • FC: 2 • FB: 3 • SFC: 4 • SFB: 5 • DB: 6
Area	USINT	Identifier for the range in which the incorrect access occurred: • B#16#01: Direct access to input • B#16#02: Direct access to output • B#16#81: Access to process image input • B#16#82: Access to process image output
DBNr	Block_DB	not user-relevant
Csg_OBNr	OB_ANY	OB number (122)
Csg_Prio	USINT	OB priority (priority of the OB)
Width	USINT	Type of access during which the error occurred: • Bit: B#16#00 • Byte: B#16#01 • Word: B#16#02 • DWord: B#16#03 • LWord: B#16#04

See also

Basics of error handling (Page 2821)

Events and OBs (Page 1853)

Principles of local error handling (Page 2823)

OB start information (Page 1890)

Setting parameters of organization blocks

Parameter assignment for time-of-day interrupt OBs

Procedure for setting the parameters

To set the parameters of a time-of-day interrupt OB, proceed as follows:

1. Open the "Properties" dialog belonging to the time-of-day interrupt OB in question.
2. Click the "Time-of-day interrupt" group in the area navigation.

Overview of the parameters that can be set

You can set the following parameters:

- Active
- Execution
- Start date and time-of-day
- Option buttons "Local time" and "System time"

"Active" parameter

If the check box is selected, the time-of-day interrupt is automatically activated after startup.

"Execution" parameter

Use the drop-down list "Execution" to define the periods at which the time-of-day interrupt OB is to be executed. The time intervals relate to the settings for "Start date" and "Time-of-day".

The following values are possible for "Execution":

- Never
- Once
- Every minute
- Hourly
- Daily
- Weekly
- Monthly
- Yearly
- End of month

Note

With the value "End of month", the value specified under "Start date" is irrelevant.

"Start date" and "Time-of-day" parameter

Here, you specify the time at which the time-of-day interrupt is to be executed for the first time.

Example: Start date = 01/05/2013, time =11:16

Depending on the value of the "Execution" parameter, the CPU generates additional time-of-day interrupts periodically. Depending on the setting, the start time relates either to the local time or to the coordinated universal timeUTC.

Note

If you set the "Execution" parameter to "Monthly", the start date cannot be set to the 29th, the 30th, or the 31st. If you want the time-of-day interrupt OB to start at the end of the month, you should set the parameter "Execution" to "End of month".

"Local time" or "System time"

Here, you decide which time the start time of the time-of-day interrupt OB relates to:

- "Local time": The start time relates to the time zone set for the CPU.
- "System time": The start time relates to the coordinated universal time UTC (Universal Time Coordinated).

See also

Time-of-day interrupt OBs (Page 1893)

Assigning parameters to cyclic interrupt OBs

Procedure for setting the parameters

To set the parameters of a cyclic interrupt OB, proceed as follows:

1. Open the "Properties" dialog belonging to the cyclic interrupt OB in question.
2. Click the "Cyclic interrupt" group in the area navigation.

Overview of the parameters that can be set in the "Cyclic interrupt" group

In the "Cyclic interrupt" group, you can set the following parameters:

- Cycle time
- Phase offset

"Cycle time" parameter

Use the parameter "Cycle time" to set the period of time between two calls of the cyclic interrupt OB. It is an integer multiple of 1 µs.

"Phase offset" parameter

Here, you set the time period by which the start times are shifted as compared to the multiple of the cycle time.

A phase offset may be advisable when you are using multiple cyclic interrupt OBs. If their cycle times have common multiples, you can use a phase offset to prevent simultaneous start times.

Priority of the cyclic interrupt OB

When you add a new cyclic interrupt OB to your project, you enter its cycle time. The priority of the cyclic interrupt OB is set in accordance with the following table.

Cycle time	Default priority
0.25 ms <= cycle time <= 2 ms	17
2 ms < cycle time <= 5 ms	16
5 ms < cycle time <= 10 ms	14
10 ms < cycle time <= 100 ms	13
100 ms < cycle time <= 200 ms	12
200 ms < cycle time <= 500 ms	11
500 ms < cycle time <= 1000 ms	10
1000 ms < cycle time <= 2000 ms	9
Cycle time > 2000 ms	8

You can change the default priority as follows:

1. Open the "Properties" dialog belonging to the cyclic interrupt OB in question.
2. Click the "Attributes" group in the area navigation.
3. In the "Priority" area, enter the desired value in the input field "Priority".

See also

Cyclic interrupt OBs (Page 1901)

Assigning parameters to hardware interrupt OBs

Procedure for configuration of the start events of a hardware interrupt OB

Each start event for a hardware interrupt OB is configured in the properties of the corresponding module. To do this, follow these steps:

1. Open the "Properties" dialog related to the corresponding module.
2. In the area navigation, click the channel that is to trigger a hardware interrupt.
3. Select those check boxes whose associated events are to trigger a hardware interrupt. Enter associated limit values, if any.
4. Enter an event name, for example, "Rising edge 0".

5. Select an existing hardware interrupt OB from the drop-down list "Hardware interrupt" or create a new hardware interrupt OB.
6. If necessary, enter the priority of the hardware interrupt OB.

A system constant of the data type "Event_HwInt" is created automatically for the event identified by its unique event name. This is displayed in the standard tag table in the "System constants" tab.

Overview of the start events of a hardware interrupt OB

To display the currently configured start events of a hardware interrupt OB, proceed as follows:

1. Open the "Properties" dialog associated with the hardware interrupt OB in question.
2. Click the "Start events" group in the area navigation.

See also

Hardware interrupt OBs (Page 1902)

Example of a hardware interrupt event (Page 1856)

Assigning parameters to isochronous mode interrupt OBs

Procedure for setting the parameters

To set the parameters of an isochronous mode interrupt OB, proceed as follows:

1. Open the "Properties" dialog associated with the isochronous mode interrupt OB in question.
2. In the area navigation, click the "Isochronous mode" group .

Overview of the parameters that can be set

You can set the following parameters:

- Application cycle
- Automatic setting
- Delay time

"Application cycle" parameter

- For PROFINET: Integer multiple (possible values: 1, ... 14) of the send clock
- For PROFIBUS DP: DP cycle

Check box "Automatic setting"

- If the check box is cleared, you specify the delay time manually.
- If the check box is selected, the delay time is calculated automatically. The associated input box is then grayed out.

"Delay time" parameter

Note

You can only set the parameter "Delay time" if the check box "Automatic setting" is cleared.

The delay time is the time interval between the send clock or the global control frame and the start of the isochronous mode interrupt OB. In this period of time, the PROFINET IO controller or the DP master handles the cyclic data exchange with the IO devices or the DP slaves.

If the delay time is zero, the isochronous mode interrupt OB starts with the send cycle clock or the global control frame. In this case, it is extremely likely that access to the isochronous I/O via the extended instruction "SYNC_PI" at the start of the OB will fail as the I/O data is not yet available.

If the delay time is greater than the default value, then the OB runtime is "given away", since the clock-synchronized I/O could have been accessed earlier.

Overview of the displayed parameters

The following parameters are displayed:

- Distributed I/O
- Process image partition

"Distributed I/O"

Here, the PROFINET IO system or DP Master system containing the IO devices or DP slaves assigned to the isochronous mode interrupt OB are displayed.

The definition of a PROFINET IO system or DP master system takes place during the assignment of the process image partition to the isochronous mode interrupt OB.

"Process image partition"

Here, the process image partition assigned to the isochronous mode interrupt OB is displayed. This assignment has been made during hardware configuration.

See also

Isochronous mode interrupt OBs (Page 1903)

Assigning the MC-Servo OB parameters

Procedure for setting the parameters

To assign the parameters of an MC-Servo OB, follow these steps:

1. Open the "Properties" dialog associated with the MC-Servo OB in question.
2. Click on the "Cycle time " group in the area navigation.

Overview of the parameters that can be set

In the "Cycle time" group, select how the MC-Servo OB will be called in the execution system of the SIMATIC S7-1500:

- Cyclical
- Synchronous to the bus

You can set the following parameters according to the option selected:

- Application cycle
- Distributed I/O
- Factor

"Cyclical" option

The MC-Servo OB is called cyclically with the specified application cycle.

You set the cycle in which the MC-Servo OB is called in the "Application cycle" field.

"Synchronous to the bus" option

The MC-Servo OB is called synchronously with a bus system. You set the send clock in the properties of the selected bus system.

The following parameters are available for the "Synchronous to the bus" option:

- Distributed I/O
  In the drop-down list, select the bus system for which the MC-Servo OB will be called isochronously. You can select the following bus systems:
  - Isochronous PROFINET IO
  - Isochronous PROFIBUS DP

  You cannot call the MC-Servo OB synchronously with a bus system that is connected to the CPU via a communication processor/communication module (CP / CM).
- Send clock (ms)
  The send clock or DP cycle that is set in the selected PROFINET IO system or DP master system is displayed in this field.

- Factor
  You can reduce the application cycle of the MC-Servo OB relative to the send clock of an isochronous PROFINET IO system. You set an integer multiple of the send clock in the "Factor" field. Values up to 14 times the send clock (maximum 32 ms) are possible for the application cycle.
- Application cycle (ms)
  An application cycle that results from the send clock and the specified factor is displayed in this field.

8.1.5.4 CPs, PC stations and network components

CP Industrial Ethernet / PROFINET

Settings

Module access protection / UDP buffering / file system

Reference

Making special settings in the "Properties > General > Settings" parameter group

Depending on the CP type, you will find settings relating to the following areas and these are described below:

- Module protection
- UDP buffering
- File system

Module access protection - "Protection level" option

With this option, you can protect the CP from accidental or unauthorized access. The following options can be selected in the drop-down list:

- Not locked
- Status-dependent
  With this setting, it is only possible to access the CP when the CPU is in STOP mode.

Switch off UDP frame buffering

UDP frames are buffered by the CP depending on the communication load between the CP and the CPU.

You can select between the following using the UDP buffering (switch off UDP frame buffering) option:

- Disabled (default setting)
  Buffering of UDP frames is enabled. UDP frames received by the CP are buffered until they have been transferred to the CPU.
- Enabled
  The CP always transfers the last received, in other words, the current frame to the CPU. As long as no new UDP frame can be transferred between the CP and the CPU due to the current communication load, only the last frame received is buffered in the CP.

Turning off the UDP frame buffering achieves the shortest possible reaction time between the arrival of the UDP frame and its evaluation on the CPU. The characteristics associated with enabling the option can be critical in certain applications with high frame traffic. Buffering of a lot of frames may result in an undesired time offset between the frames accumulated in the CPU and the current frames at the Ethernet interface.

File system - Match case (case sensitive) - option for CPs with IT functions

Enabled: When files are placed in RAM, the CP distinguishes between upper- and lower-case characters in the file names.

Data transfer > 240 bytes

Reference

Making special settings in the "Properties > General > Settings" parameter group

Depending on the CP type, you will find setting "Data transfer > 240 bytes"

Notes on data transfer > 240 bytes

- Data transfer > 240 bytes is supported by the newer versions of the CPs. Refer to the information in the documentation of your Ethernet CP.
- With the S7-300 series, remember that this configuration uses one connection resource (free connection for S7 functions) on the S7-300 CPU. CPU connection resources are also used, for example, by S7-300 CPs in FMS mode or by programming devices or OPs.

Ethernet nodes / PROFINET interface

Advanced options

Send keepalives for connections (TCP and ISO-on-TCP connections only)

Reference

Making special settings for the PROFINET interface in the "Properties > General >PROFINET interface > Advanced options > Interface options" parameter group

Send keepalives for connections (TCP and ISO-on-TCP connections only)

Here, you can set the interval at which keepalives are sent to the partner of a communications connection.

The Ethernet CP is configured for all connection-oriented services so that keepalives are sent. This ensures that connections are terminated if one of the communication partners fails and the connection resources are released again. The setting made here applies to all TCP or ISO-on-TCP connections operated via the CP; a connection-oriented setting is not possible.

Range of values

- Default setting: 30 seconds
- Maximum value: see information displayed in the input box
- Keepalive disabled: 0 seconds

Note

Note that the keepalive mechanism can cause subordinate connections (for example, an ISDN telephone connection) to be maintained even though user data are not actually being transferred. If you want to avoid this, you will have to select one of the following alternatives:

- Disable the keepalive mechanism (interval time = 0)
  or
- set the interval so high that the underlying connection is terminated before a keepalive is sent when there is no data traffic.

IP configuration

Reference

Setting parameters for the IP protocol in the "Properties > General > PROFINET interface > Ethernet addresses" parameter group

Meaning

You can decide the route and the method with which the IP address of the local interface is obtained and assigned.

With the options available here, it is possible to assign IP addresses "dynamically" outside the configuration.

The selection you make also decides whether communication connections are set up by the project engineering or via the interface in the user program (IP_CONFIG instruction).

The following options are available:

- Set IP address in the project
  This is the default setting for PLCs. You specify the IP address when the device is networked. The IP address CP is therefore fixed.
  With this option, you must configure communications connections.
- Obtain an IP address from a DHCP server
  If you select this option, the IP address is obtained from a DHCP server when the device starts up.
  The DHCP server is informed of the MAC address of the interface or the client ID that can be entered in the configuration.
  The client ID is a string with a maximum of 63 characters.
  Requirements / restrictions: If you select this option, it is initially not possible to create a fully specified connection in the project because the local IP address is not known. You therefore select "unspecified" with passive connection establishment as the connection type.
- Set IP address in the user program
  With this option, you specify that the IP address is set over the user program interface (function block IP_CONFIG). This allows the IP address to be supplied dynamically during operation.
  In this situation, communications connections are set up solely via the interface of the user program, connection configuration is not possible (relates to connections via: TCP, ISO-on-TCP, UDP, ISO transport).
- Set IP address on the device
  With this option, you specify that the IP address is set by other services outside the configuration.
  In this case, connection configuration is no longer possible (relates to connections via: TCP, ISO-on-TCP, UDP, ISO transport).

Note

If you have already configured communications connections via the interface configured here and you select a setting other than "Set IP address in the project", a message will be displayed indicating that the configured connections no longer function.

Time-of-day synchronization

Time-of-day synchronization - mode

Reference

Making settings for the interface properties in the "Properties > General > PROFINET interface > Advanced options > Time-of-day synchronization" parameter group.

How the time synchronization modes work

- SIMATIC mode
  When the CP receives MMS time messages, its local time is synchronized provided that the NTP mode was not configured (MMS = Manufacturing Message Specification).
  The advantage of this mode is that it is generally more accurate than the NTP mode.
  The time messages can be received either from an S7-41x CPU or from the LAN.
  The two following situations are possible depending on where the time-of-day master is located:
  - Time messages can come from the subnet (LAN) and be forwarded to the station. In this case, the CPU's time synchronization must be configured as a slave.
  - Time messages can come from the station and be forwarded to the subnet (LAN). In this case, a time-of-day synchronization of the CPU must be configured as the master or another CP forwards time messages to the K bus.

  If you have several CPs in your station, take into account the flow of time-of-day messages depending on the time-of-day master since you can forward time-of-day messages from one network to another network using the function described here. There must be only one time-of-day master in your station.
  If there are several CPs that are connected to the same network in a station, only one of these CPs may forward the time messages.
- NTP mode (NTP: Network Time Protocol)
  In NTP mode, the CP sends time queries (in client mode) at regular intervals to NTP servers on the subnet (LAN). Based on the replies from the server, the most reliable and most accurate time is calculated and the time of day on the station is synchronized.
  The advantage of this mode is that it allows the time to be synchronized across subnets.
  In this case, any MMS time messages that are received will be ignored.
  The IP addresses of up to four NTP servers need to be configured. The update interval defines the interval between the time queries (in seconds). The value of the interval ranges between 10 seconds and one day.
  In NTP mode, it is generally UTC (Universal Time Coordinated) that is transferred; this corresponds to GMT (Greenwich Mean Time). The time offset from UTC can be set by configuring the local time zone.

Time-of-day synchronization in SIMATIC mode - parameter assignment

Reference

Making settings for the interface properties in the "Properties > General > PROFINET interface > Advanced options > Time-of-day synchronization" parameter group

"Activate time-of-day synchronization" and "SIMATIC mode" options are selected.

Time-of-day synchronization in SIMATIC mode

In "SIMATIC" mode, the CP forwards MMS time-of-day messages when it receives them from a SIMATIC time transmitter. The time-of-day messages can be received either from the communication bus (from an S7-41x CPU) or from the LAN.

The two following situations are possible depending on where the time-of-day master is located:

- Time messages can come from the subnet (LAN) and be forwarded to the station.

  Note

  Time-of-day synchronization of the CPU (exception: CPU 318) must be configured as the slave.

- Time messages can come from the station and be forwarded to the subnet (LAN). In this case, a time-of-day synchronization of the CPU must be configured as the master or another CP forwards time messages to the K bus.

If you have several CPs in your station, take into account the flow of time-of-day messages depending on the time-of-day master since you can forward time-of-day messages from one network to another network using the function described here. There must be only one time-of-day master in your station.

If there are several CPs that are connected to the same subnet and if forwarding of the time of day is activated for these, the following applies: The CP with the lowest slot number handles the forwarding of the messages. All other CPs configured for forwarding time-of-day messages, automatically stop forwarding.

Forwarding the time of day - possible settings for the direction:

- Automatic
  The CP receives the timeofday message from the LAN or from the station and forwards it to the station or to the LAN.
  If several CPs are being operated in the station, this automatic setting can lead to collisions. To avoid this, you can specify the direction of forwarding with the following options:
- From station
- From LAN

"Use corrected time" option

With the "Use corrected time" option, you can decide whether or not a correction factor in the time-of-day frame is taken into account. The correction factor specifies a time offset (in units of ½ hours) resulting either from the time zone setting or the daylight-saving/standard time setting of the time transmitter.

If you select this option, the CP includes the correction factor in the time of day so that systems to which the time is forwarded do not need to use the factor.

Example:

The time 9:21 is transferred in the time-of-day frame. The frame also contains the correction factor +2x0.5. The corrected time forwarded by the CP is therefore 10:21.

Time-of-day synchronization in NTP mode - parameter assignment

Reference

Making settings for the interface properties in the "Properties > General > PROFINET interface > Advanced options > Time-of-day synchronization" parameter group

"Activate time-of-day synchronization" and "NTP mode" options are selected.

NTP mode (NTP: Network Time Protocol)

In NTP mode, the CP sends time-of-day queries at regular intervals (in the client mode) to the NTP server in the subnet (LAN). Based on the replies from the server, the most reliable and most accurate time is calculated and the time of day on the station is synchronized.

The advantage of this mode is that it allows the time to be synchronized across subnets.

The IP addresses of up to four NTP servers need to be configured. The update interval defines the interval between the time queries (in seconds). The value of the interval ranges between 10 seconds and one day.

In NTP mode, it is generally UTC (Universal Time Coordinated) that is transferred; this corresponds to GMT (Greenwich Mean Time).

"Time-of-day synchronization on the full minute" option

With the "Time-of-day synchronization on the full minute" option, you can decide whether or not the time of day is forwarded to the communication bus on the full minute.

This option is required for certain special applications.

"Forward time of day to station" option

As default, the "Forward time of day to station" option is enabled.

Some CPUs provide the option of requesting the time of day automatically from an NTP server. If this option is enabled on the CPU, you should disable the option described here on the CP. This avoids the time of day acquired by the CPU from the NTP server being overwritten again by the time of day received by the CP. Forwarding over the CP might result in a lower degree of accuracy.

Additional information

Individual network settings

Individual network settings

Here you can make fixed network settings as required. The "Automatic setting" is selected by default. In normal situations, this guarantees problem-free communication.

If problems do occur in communication (such as when connections cannot be established or frequent network faults occur), this may be caused by an unsuitable selected or automatic network setting. In this case, you should select a network setting that is suitable for your network configuration.

Range of values and example of the IP address

Range of values for the IP address

The IP address consists of four decimal numbers from the range 0 to 255 separated by a period.

Notation: xxx.yyy.zzz.www

Example:

141.80.0.16

- The following addresses have special meanings (*stands for any number between numeric 0 and 255):
  - 127.*.*.* Loopback
  - 0.0.0.0 unspecified network
  - 255.255.255.255 broadcast to all devices (permitted only when "All broadcast devices" was selected as the connection peer)
  - Host address 255 broadcast address for all devices of a network:
    For class A: xxx.255.255.255
    For class B: xxx.yyy.255.255
    For class C: xxx.yyy.zzz.255
- The following are not permitted:
  - Host address 0
    For class A: xxx.0.0.0
    For class B: xxx.yyy.0.0
    For class C: xxx.yyy.zzz.0

Range of values and example of the subnet mask

Range of values for subnet mask

To separate the network ID from the host address in the IP address, the subnet mask was introduced.

The subnet mask consists of four decimal numbers from the range 0 to 255 separated by a period, for example 255.255.0.0.

In their binary representation, the four decimal numbers of the subnet mask must include a continuous series of "1" values without any gaps from the left and a series of "0" values without any gaps from the right.

The "1" values decide the network number within the IP address. The "0" values decide the host address within the IP address.

Example:

Correct values: 255.255.0.0 D = 1111 1111.1111 1111.0000 0000.0000 0000 B

255.255.128.0 D = 1111 1111.1111 1111.1000 0000.0000 0000 B

255.254.0.0 D = 1111 1111.1111 1110.0000 0000.0000.0000 B

Incorrect value: 255.255.1.0 D = 1111 1111.1111 1111.0000 0001.0000 0000 B

Relationship between the IP address and subnet mask

The first decimal number of the IP address (from the left) decides the structure of the subnet mask in terms of the number of "1" values (binary) as follows ("x" stands for the host address or "0" in the subnet mask):

- First decimal number of the IP address subnet mask
  Class A: 0 to 126 255.x.x.x
  Class B: 128 to 191 255.255.x.x
  Class C: 192 to 223 255.255.255.x

Note

You can also enter a value between 224 and 255 for the first decimal number of the IP address. This is, however, not advisable since STEP 7 does not check these addresses.

Relationship between the IP address and address of the gateway

Relationship between the IP address and address of the gateway

The IP address and the address of the gateway may only differ at the positions at which there is a "0" in the subnet mask.

Example:

You have entered the following: 255.255.128.0 for the subnet mask; 141.30.0.5 for the IP address and 141.30.128.1 for the address of the gateway. The IP address and the address of the gateway may only have a different value in the fourth decimal number. In the example, however, the third position is different.

In the example, you will therefore need to make one of the following changes:

The subnet mask to: 255.255.**0**.0 or

The IP address to: 141.30.**128**.5 or

The address of the gateway to: 141.30.**0**.1

Port parameters

Individual network settings for each port

When necessary, you can make fixed network settings for the transmission characteristics of each available interface (port). The "Automatic setting" is selected by default. In normal situations, this guarantees problem-free communication.

If problems do occur in communication (such as when connections cannot be established or frequent network disruptions occur), this may be the result of unsuitable network settings, either selected or automatic. In this case, you should select a network setting that is suitable for your network configuration.

Functional description of the automatic switchover

All ports of the CP provide a 10/110 Mbps full duplex connection with autosensing and autonegotiation of the network settings. These functions run as follows after turning on the CP:

- The CP attempts to detect the transmission speed used by the partner.
- If detection is possible, the CP attempts to negotiate the optimum duplex mode with the partner.
- If no negotiation is possible, the CP uses the previously detected transmission speed and half duplex.

This takes approximately 2 seconds.

The "Automatic setting" also includes an autocrossing mechanism; this means that you can use a crossover or straight-through cable for the connection regardless of the properties of the partner device.

Changing to individual network settings

If you create any configuration manually, automatic switchover is no longer effective. This also applies to the autocrossing mechanism integrated in the CP. If the connected partner device does not support the autocrossing mechanism, you will need to use a crossover cable (for a switch) or a straight-through cable (for an end device).

"Disable" port option:

Depending on the module type, the drop-down list includes the "- Disable -" option. This option, for example, allows you to prevent access to an unused port for security reasons.

See also

Further information

Please read the manual on the relevant CP for further information.

Maximum frame size / volume of data for PROFINET IO devices

Size of the user data with the CP 1616

The size of the user data (process data) is calculated from the maximum frame size (gross volume of data) minus the values for the modules plugged into this IO device. This results in the net volume of data.

Table 8-92 Example of the calculation of the user data (bytes) of the CP 1616 as an IO device with homogeneous modules (for example only DI modules)

Bytes	Explanation
1434	Gross size of the frame
-16	for the maximum number of pluggable modules
= 1418	Net size of the user data

Table 8-93 Example of the calculation of the user data (bytes) of the CP 1616 as an IO device with heterogeneous modules (for example DI/DO)

Bytes	Explanation
1434	Gross size of the frame
-32	for twice the maximum number of pluggable modules
= 1370	Net size of the user data

Size of the user data with the CP 343-1 (Lean/Standard/Advanced)

The size of the user data (process data) is calculated from the maximum frame size (gross volume of data) minus the values for the CP and the interface itself, for the number of ports and the modules plugged into this IO device. This results in the net volume of data.

Table 8-94 Example of calculating the user data (bytes) of the CP 343-1 as an IO device

Bytes	Explanation
548	Gross size of the frame
-1	for the CP itself
-1	for the Interface
-2	for 2 ports
-32	for the maximum number of pluggable modules
= 512	Net size of the user data *

* By using fewer but larger modules, the volume of user data can be increased.

DNS configuration

Reference

Setting the DNS server address in the "Properties > General > DNS configuration" parameter group

Meaning

When configuring an e-mail connection, the address of the e-mail server via which the e-mails are sent must be specified. This address can be specified in absolute or symbolic form. If you use a symbolic address, the absolute address is obtained by querying the DNS server specified here.

See also

Setting up an email connection (Page 665)

E-mail address details (Page 666)

IP access protection

IP access protection

Reference

Making settings for the interface properties in the "Properties > General > IP access protection" parameter group.

Overview

The following options can be assigned for IP access protection according to the CP type:

- "Enable web server" option
  The CP provides you with the functionality of a web server for access by means of a web browser. Certain HTML pages with CP information and diagnostic functions are stored in a memory area of the CP for this.
  Enable this option in order to be granted access to these HTML pages. Port 80 of the CP is thereby enabled.
  Web server access is enabled by default.
- "Enable FTP server" option
  Select this option if you want to allow FTP access to the S7 station via Port 20/21 of the CP.
  If you configure FTP access to file DBs in the CPU in the "FTP" tab, access is only possible if you select the option here.
  FTP server access via Port 20/21 is enabled by default.
- "Enable access protection for IP communication" option
  The Ethernet interface allows you to restrict access to the local device to partners with specified IP addresses. Partners you have not authorized then have no access to data on the local device via this interface.
  Here, you can enable or disable IP access protection and can also enter certain IP addresses in an IP address control list (IP-ACL).
  As default, the IP access protection is **dis**abled.
  Blocked access attempts are registered on the CP and can be viewed with special diagnostics in the "IP access protection" diagnostic object. If the CP has IT functionality, a LOG file is also created in the file system of the CP that you can view with a WEB browser. Please refer to the instructions on handling the LOG file and the notes on loading the configuration data below.

IP access protection for configured communication partners

If you want to restrict access so that only the communication partners you specified in the configuration have access to the local device, you simply need to enable access protection. In this case, you do not need to enter any IP addresses in the list.

These communication partners include:

- Stations to which communication connections are configured;
  With the exception of S7 connections, this also applies to connections on which the connection partner is located in a different subnet.
  You should, however, remember that where you have unspecified connections whose partner addresses are unknown; all IP addresses that are not entered are unauthorized and will be rejected. Even connections in PROFINET CBA will be treated as unspecified connections. You will need to enter the IP addresses of such connections explicitly in the IP access control list.
- PROFINET IO devices when the Ethernet CP is used as a PROFINET IO controller;
  The IP addresses of PROFINET IO devices are entered dynamically in the IP access control list when the CPU is in RUN mode.
  If the CPU changes to STOP mode, the IP addresses of the PROFINET IO devices are deleted from the access list.
- NTP servers, SMTP servers, DNS servers and DHCP servers.
  The IP addresses of NTP servers, SMTP servers, DNS servers and DHCP servers are also entered and removed dynamically when there are requests to these servers.

IP access protection relates to all connection types that use the IP protocol (TCP, ISO-on-TCP, UDP, S7)

Note on dynamically assigned IP addresses:
Since each service manages its dynamic entry in the ACL itself, it is perfectly possible for the same IP address to appear several times in module diagnostics.

IP access protection for partners with specific IP addresses

To allow access only by certain IP addresses, enter these IP addresses in the IP access control list. These can, for example, be the IP addresses of connection partners that remained unspecified in the connection configuration, of individual programming devices or of connection partners on PROFINET CBA.

The IP addresses you specified in the connection configuration always belong to the permitted IP addresses and do not therefore need to be entered explicitly in the IP-ACL.

Viewing the LOG File with a Web browser

On CPs with IT functionality, a LOG file is created in the file system of the CP and this can be viewed with a Web browser. Compared with the recording made in the special diagnostics, the LOG file provides space for up to 512 entries.

The LOG file is available only after activating IP access protection the first time.

You will find the LOG file as an HTML file in the file system of the CP in the following directory:

·ram/security/IPLogFile.htm

Further properties:

The LOG file is created as a ring buffer. When more than 512 entries have been recorded, the oldest entries are then overwritten.

Entries are made chronologically and there are no other criteria for sorting.

Note that this function is supported only by the newer firmware versions of the CPs; you will find more detailed information in the CP device documentation.

With Advanced CPs as of CP 343-1 Advanced (GX30) and CP 443-1 Advanced (GX20) the LOG file is not created. On these CPs, you can view the blocked access attempts directly with Web diagnostics.

Entering access permissions in the IP access control list (IP-ACL)

Reference

Making settings for the interface properties in the "Properties > General > IP access protection" parameter group.

Rules for entering the IP addresses:

Keep to the following rules when entering IP addresses:

- You can enter the IP address singly or as a range.
- You can also enter comments.
- The maximum number of addresses you can enter is: 32
- The maximum number of address ranges you can enter is: 10
- The system does not check whether addresses that have already been entered singly are also included in an address range.
- The system does not check whether a range includes invalid addresses (for example, broadcast addresses can be specified here although they cannot occur as the IP address of a sender).
- Comments begin with the "#" character

Examples:

- Individual IP address with comment: 141.80.0.16 #address of partner x
- Range of IP addresses with comment: 141.80.0.16 - 141.80.0.25 #address range for control level

Rights

The access permissions described here can be selected in the IP access control list with their short designations.

Table 8-95 Access permissions and short designations

Access right	Short designation	Meaning / remark
Access to station	A	Access Communications partners with addresses in the specified range have access to the station (CP / CPU) belonging to the CP. This access permission is set implicitly for IP addresses you have specified in the connection configuration.
Modifying the IP access control list	M	Modify Communications partners with addresses in the specified range, have the right to access the IP access control list. These communications partners can send entries for the IP access control list to the CP using HTTP. If you select this permission, the permission "A" is also set. Notes on the entries sent using HTTP: • With each list transferred using HTTP, a previous list sent using HTTP becomes invalid. • The access permissions sent using HTTP can be added to entries configured in STEP 7, but the entries cannot be deleted. • A list transferred using HTTP is deleted if there is a power down on the CP (power OFF). You will find information on the syntax of a list transferred using HTTP in the documentation of the CP.
IP routing access to the another IP subnet	R	Routing Communications partners with addresses in the specified range have access to other subnets connected to the CP. This access permission is not set automatically for IP addresses you have specified in the connection configuration. Where necessary, this access permission must be set here explicitly.

Note

- Loading configuration data via TCP/IP
  When downloading via TCP/IP with access protection enabled, note that you will need to include the IP address of the programming device / PC from which the configuration is downloaded in the IP-ACL. The programming device / PC would otherwise be detected as unauthorized during the loading and the loading routine would be aborted
- Loading S7 connections later
  If you want to load individual S7 connections to the S7 station later, these are not included automatically in the IP-ACL. In this case, you will need to load the entire station configuration again.
- Adopting and editing a configuration by loading to the programming device / PC
  If you want to edit the connection configuration and have enabled IP access protection, you will require the full connection information on the programming device / PC. This means that you will need to upload the current connection data. When you then load the data to the S7 station again, you will need to load the entire connection data and station data.

Note

Using special diagnostics, you can view the current IP-ACL on the S7 station and, if required, compare it with the configured IP-ACL.

Editing access permissions in the user management

Reference

Setting user permissions in the "Properties > General > User management" parameter group

Meaning of user management

In user management, you specify which users have which permissions when accessing the S7 station via a Web browser or as an FTP client.

In the alphabetical list, you will find the users that have already been entered under user name for which passwords have been stored.

The “Everybody" entry is a default entry and is always present. This cannot be modified. It is also not possible to assign a password to this entry.

User entries are always linked to a password. Exception: The "Everybody" entry is not password protected.

Note

With the "Everybody" setting, no rights whatsoever are assigned. For service purposes, however, it is possible to assign permissions.

Remember, however, that if rights are assigned to the "Everybody" user, they are available to every other user. If you do assign permissions for service purposes, remember to cancel these again afterwards! Otherwise, you allow services to be executed without authorization with every access.

If an access right is set for the "Everybody" user, the corresponding check boxes are selected and visible when you assign rights to other users.

Entering new users

1. Select the first free row in the user list.
   A new user entry is created with the access level "Minimum".
2. Adapt the proposed user name in the "Name" column.
3. Double-click in the "Access level" column of the newly created row.
   The icon for the drop-down list becomes visible.
4. Click on the icon of the drop-down list.
5. In the dialog that then opens, set specific user permissions.
6. Confirm your entries with "OK".
7. Assign a new password in the password dialog that opens.

Changing user permissions or a password:

1. Double-click in the "Access level" column of the row of the required user entry.
   The icon for the drop-down list becomes visible.
2. Click on the icon of the drop-down list.
3. In the next dialog, modify the specific user permissions or click on the "Password" button to assign a new password.
4. Confirm your entries with "OK".
5. Enter the password in the password dialog that opens.

Deleting user entries:

1. Select "Delete" in the context menu of the row of the required user entry.

Tag declaration - Edit tag list

Reference

Setting the tags configured on the CPU in the "Properties > General > Tag declaration" parameter group.

Meaning of the tag declaration

To access tags of the CPU using a Web browser and Java applet, the CP must be aware of the names, addresses, and access rights of these tags.

In the symbol list in the tag declaration, you can see the currently known tags with their symbolic names, the corresponding operands and the assigned access rights.

Adding a new tag entry

1. Double click on the first free row in the table.
   The symbol becomes visible for selection from the drop-down list.
2. Select the symbol to open the structure display of the tags on the current device.
3. Select the symbols of the tags or the structure elements to which access will be possible using a Web browser.
   Note: You must first store the symbols in the Symbol Editor!

Deleting a tag entry

1. Select "Delete" in the shortcut menu of the row of the tag to be deleted.
   Caution:

   Note

   There is no prompt for confirmation, the entry is deleted immediately. Multiple selections are not possible.

Setting access permissions - "Writable" option

The default permissions for the tags are read-only.

To assign write/read rights for the tags entered here, follow the steps below:

1. Select one or more tags in the symbol list.
2. Set the "Only reading allowed" option as required for the selected tags.

Tag declaration - Edit tag list (ERPC)

Reference

Configuring the tags on the CPU in the "Properties > General > Tag declaration" parameter group.

You select the tags you want to make accessible to the ERPC application.

Information displayed

The list shows the ERPC tags stored in the STEP 7 project. Any ERPC tags that may have been downloaded to the station previously are not displayed.

The following information is displayed in the list of tags configured for ERPC:

- Info
  Additional information from the check of changes made
  ERPC tags that you have changed in the STEP 7 project since the last time you saved have additional information displayed in the "Info" column by the check changes function (see below).
- Tag
  Name of the tag
- Writable
  The permissions configured for ERPC communication (reading from the CPU, writing to the CPU), as default only read permissions are assigned.
- Comment
  Comments on the ERPC tags specifically for ERPC communication.

Below the table:

- Advantage
  If you select the "Read only" option, you assign the "read only from the CPU data area" permission to the selected ERPC tags.
- Number of tags:
  The total number of tags used for ERPC

Check for changes

When you open the tag declaration, a check is made for any changes. The tags selected for the ERPC application are compared with the those in the tag table of the assigned CPU in the STEP 7 project (offline tag table).

Note

There is no comparison of ERPC tags you may have downloaded previously from the STEP 7 project to the station.

The "Info" column displays the relevant information on differences between the ERPC tags and those of the tag table of the CPU.

Display in the "Info" column	Status / information
	Tag unchanged
ACC	The tag cannot currently be edited. Reason: The data block in which the tag is used is currently open.
DEL	Tag deleted in the tag table of the CPU
ADR	Address changed in the tag table of the CPU
TYP	Data type changed in the tag table of the CPU Note: If the address and the data type of a tag are changed in the tag table, the "ADR" information is overwritten by the "TYP" info.
DEF	Tag type not supported
UPL	Tag uploaded from the station (after using the STEP 7 function "Upload to PG")

If entries of the type "DEL", "ADR" and "TYP" exist, you will need to make suitable corrections before you save the STEP 7 project.

Multiple selection of tags

You can select more than one tag using the <Shift> and <Ctrl> keys.

FTP configuration for FTP server mode

Ethernet CP as FTP server for the CPU data

Reference

Data acquisition for automatic creation of a file allocation table in the "Properties > General > FTP configuration" parameter group.

With the information in the file allocation table, it is possible to address data blocks in one or more CPUs (up to 4) in an S7 station.

"Activate FTP server" option (only with certain CP types)

Select this option to allow FTP access to the S7 station via port 20/21 of the CP.

This option needs to be enabled to enable FTP access to file DBs on the CPU with the option described below.

FTP server access via Port 20/21 is enabled by default.

"Use FTP server for S7 CPU data" option

If you select this option, the file allocation table configured here is created on the CP when you download the project engineering data and store it in the /config folder of the file system of the CP.

An existing file_db.txt file is overwritten.

Note

With security and the "Allow access only via FTPS" enabled, the following applies:

- The files are transferred encrypted.
- For the user, the rights "FTP: Read files (DBs) from the S7 CPU" or "FTP: Write files (DBs) from the S7 CPU" must be activated.
- If the firewall is activated, the FTP/FTPS protocols must be allowed.

Ethernet CP as FTP server for the S7 CPU data

To transfer data using FTP, you create data blocks on the CPU of your S7 station; due to their special structure, these are known here as file DBs.

When operating as an FTP server, if the Ethernet CP receives an FTP command, it checks a file allocation table (file file_db.txt) to find out how the data blocks used in the S7 station for file transfer are mapped to files.

You can create the file allocation table and transfer it to the CP as follows:

- With an entry in the parameter group "Properties > FTP configuration" described here
  The file allocation table is then created in the FTP configuration based on this information and downloaded automatically to the CP along with the configuration data.
- By creating a file_db.txt file directly
  If you create the file allocation table in this way, you will need to load it to the CP using an FTP command.

The file allocation table file_db.txt is stored in the file system of the Ethernet CP in the /config directory.

See also

Layout and structure of the FTP configuration list (Page 1944)

Structure of the data blocks (file DBs) for FTP services - FTP server mode (Page 1945)

Layout and structure of the FTP configuration list

Layout and structure of the FTP configuration list for generating the file allocation table

The fields of the table in the FTP configuration have the following meaning and syntax:

Titles in the table for FTP configuration	"CPU"	"DB"	File name	Comment
Meaning	Rack/slot assignment of the CPU Can be selected from a drop-down list; cannot be typed in.	Number of the data block Entry can be typed in	File name assigned to the file DB Automatic name proposal; entry can be typed in / modifiable.	Informal comment
Example 1	cpu1 [PLC_1]	20	cpu1_db20.dat	Measured values in the Plant1 area
Example 2	cpu2 [PLC_2]	35	Cpu2_db35.dat	Measured values in the Plant2 area

Notes on the syntax

- Valid delimiters for the entries are "spaces".
- The following applies to the file name of a file DB:
  - The file name begins with "cpuX" (where X=1, 2, 3 or 4);
  - "cpuX" must first be defined in the rack/slot assignment;
  - Length: Maximum of 64 characters (including "cpuX");
- Maximum of 100 entries;
- Permitted characters: Letters "A-Z, a-z"; numbers "0-9", "_", "."

Note

Keep to the notation (lower case for "cpu" and no leading spaces at the start of the row). Otherwise, the files will not be recognized.

The tabulator is **not** permitted as the delimiter.

See also

Ethernet CP as FTP server for the CPU data (Page 1942)

Structure of the data blocks (file DBs) for FTP services - FTP server mode

How it works

To transfer data with FTP, create data blocks (file DBs) on the CPU of your S7 station. These data blocks must have certain structure to allow them to be handled as transferable files by the FTP services. They consist of the following sections:

- Section 1: File DB header (has a fixed length (20 bytes) and structure)
- Section 2: User data (has a variable length and structure) data type "FILE_DB_HEADER

Data type "FILE_DB_HEADER"

Using the "Add new block" function, you assign the type "FILE_DB_HEADER" to a data block directly.

File DB header for FTP server mode

Note: The file DB header described here is largely identical to the file DB header for client mode. The differences relate to the following parameters:

- WRITE_ACCESS
- FTP_REPLY_CODE

Parameter	Type	Value / meaning	Supply
EXIST	BOOL	The EXIST bit indicates whether the user data area contains valid data. The retrieve FTP command executes the job only when EXIST=1. • 0: The file DB does not contain valid user data ("file does not exist"). • 1: The file DB contains valid user data ("file exists").	The dele FTP command sets EXIST=0; The store FTP command sets EXIST=1;
LOCKED	BOOL	The LOCKED bit is used to restrict access to the file DB. • 0: The file DB can be accessed. • 1: The file DB is locked.	The stor and retr FTP commands set LOCKED=1 when they are executed. The following function is also possible when writing from the user program: The user program on the S7 CPU can set or reset LOCKED during write access to achieve data consistency. Recommended sequence in the user program: 1. Check LOCKED bit; if = 0 2. Set WRITEACCESS bit = 0 3. Check LOCKED bit; if = 0 4. Set LOCKED bit = 1 5. Write data 6. Set LOCKED bit = 0
NEW	BOOL	The NEW bit indicates whether data has been modified since the last read access. • 0: The content of the file DB is unchanged since the last write access. The user program of the S7 CPU has registered the last modification. • 1: The user program of the S7 CPU has not yet registered the last write access.	After execution, the stor FTP command sets NEW=1 After reading the data, the user program on the S7-CPU must set NEW=0 to allow store to be used again or to be able to delete the file with the dele FTP command.

Parameter	Type	Value / meaning	Supply
WRITE_ACCESS	BOOL	0: The FTP client on the PG/PC has no write access rights for the file DBs on the S7 CPU. 1: The FTP client on the PG/PC has write access rights for the file DBs on the S7 CPU.	During the configuration of the DB, the bit is set to an initialization value. Recommendation: Whenever possible, the bit should remain unchanged! In special situations, adaptation during operation is possible.
ACT_LENGTH	DINT	Current length of the user data area. The content of this field is only valid when EXIST = 1.	The current length is updated following write access.
MAX_LENGTH	DINT	Maximum length of the user data area (length of the entire DB less 20 bytes header).	The maximum length should be specified during configuration of the DB. The value can also be modified by the user program during operation.
FTP_REPLY_CODE	INT	This parameter is irrelevant in FTP server mode.	Is set to "0" by the FTP server.
DATE_TIME	DATE_AND_TIME	Date and time of the last modification to the file. The content of this field is only valid when EXIST = 1.	The current date is updated following a write access. If the function for forwarding the time of day is used, the entry corresponds to the time that was passed on. If the function for forwarding the time of day is not used, a relative time is entered. This time relates to the startup of the IT-CP (the initialization value is 1.1.1994 0.0 (midnight)).

See also

Ethernet CP as FTP server for the CPU data (Page 1942)

I/O addresses

Reference

Setting the input/output addresses in the "Properties > General > I/O addresses" parameter group

Meaning

- Inputs, outputs
  Assign a start address to the module. (Outputs only if the option "Address setting for LOCK/UNLOCK with FETCH/WRITE" is selected)
- "Address setting for LOCK/UNLOCK with FETCH/WRITE" option
  Select this option if you want to use the access coordination function with the FCs LOCK/UNLOCK in FETCH/WRITE mode.
  This function uses process output via the backplane/P bus. As a result, the output addresses can be set as soon as this option is selected.

Web

Web server on the CP

Reference

Parameter assignment for access to the Web server on the CP in the "Properties > General > Web server" parameter group

"Activate Web server" option

The CP provides you with the functionality of a web server for access by means of a web browser. Certain HTML pages with CP information and diagnostic functions are stored in a memory area of the CP for this.

Enable this option in order to be granted access to these HTML pages. Port 80 of the CP is thereby enabled.

Web server access is enabled by default.

Web diagnostic options (availability depends on the CP type)

- "Display topology" option
  The option allows the display of topology information for the networked PROFINET IO interfaces of the CP in the Web server.
  To collect and store the topology information, additional memory is taken up on the CP. With complex project configurations making high demands on the memory resources, it may therefore be advisable to activate the option only temporarily for service purposes.
- "Download firmware via Web" option
  By enabling the option, the function for downloading the firmware of the CP from the download center is enabled in the Web server.
  This option is not restricted by the user administration if security is disabled. It is therefore recommended that the option is enabled only when required.
- "Reload of language files for the diagnostics displays via Web" option
  Diagnostics displays of the CPs are shown in plain language in the Web diagnostics buffer. These displays are language-specific.
  If you enable the option, the function for reloading missing language files from the download center is enabled in the Web server.

Automatic update / "Update interval"

If you select the "Enable" option, the CP updates the displayed Web pages at the selected intervals.

Range of values for the update interval 1 to 999 s

Web diagnostics on the CP

Reference

Start diagnostics in the Web browser in the "Properties > General > Web diagnostics" parameter group.

Requirement:

Establish a physical connection between the programming device and the SIMATIC S7 station and assign the parameters for your PG/PC interface so that the CP is accessible. Further help is available in the "Set PG/PC Interface" dialog box.

"Start of Web diagnostics"

Click the "Web diagnostics" button to display the result of the module diagnostics in the Web browser. The content is supplied by the integrated HTTP server of the CP.

- "Access via" option
  From the "Access via" list, select the interface via which the CP is addressed.
  - Available interfaces can be selected in the drop-down list.
  - You can also enter an IP address.

Note on activated Security

If the option "Allow access only via HTTPS" is enabled, note the following:

- The data is transferred encrypted.
- The "Access to Web diagnostics" right must be activated for the user.
- If the firewall is activated, the HTTP/HTTPS protocols must be allowed.

SNMP

Reference

You enable and configure SNMP in the "Properties > General > SNMP" parameter group

"Activate SNMP" option

- If the option is not enabled, no SNMP service (neither v1 nor v3) may reply.
- If the option is enabled, the module acts as an SNMPv1 agent by default. The module acts as an SNMPv3 agent only when security is enabled and when SNMPv3 is selected for the SNMP configuration.

CP/CM Industrial Ethernet

FTP configuration

CP as FTP server for the S7 CPU data

Ethernet CP as FTP server for the CPU data

Reference

Parameter assignment of the FTP server mode in the parameter group of the Ethernet CP

- with security disabled in "Properties > General > FTP configuration"
- With security enabled in "Properties > Security > General > FTP configuration"

S7-1500 CP as FTP server

The functionality described here allows you to transfer data in the form of files to data blocks or from data blocks of an S7 station using FTP commands. At the same time, the conventional FTP commands for reading, writing and managing files can also be used.

To transfer data with FTP, create data blocks on the CPU of your S7 station. Due to their special structure, these are known as file DBs.

When it receives an FTP command, the CP 1543-1 acting as FTP server queries a file allocation table to find out how the data blocks used for file transfer in the S7 station will be mapped to files.

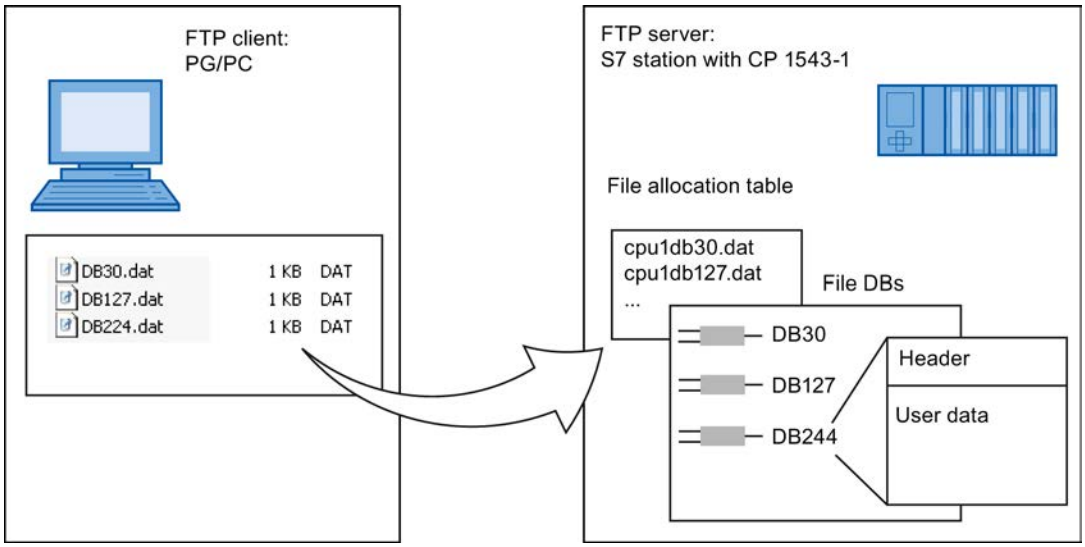


Figure 8-55 S7 CPU with CP 1543-1 as FTP server for the S7 CPU data

File allocation table

The CP 1543-1 as FTP server requires information on how the data blocks used in the S7 station for file transfer are mapped to files.

The file allocation table contains the following entries:

- Assignment of the CPU
- DB assignment
- File name

Requirements in the CPU configuration

Use the following settings to allow FTP access:

- In the configuration data of the CPU in "Properties > General > Protection": Disable the "Disable PUT/GET communication" option.
- For all data blocks being used as file DBs, disable the "Optimized block access" attribute

See also

Layout and structure of the FTP configuration list (Page 1952)

FTP commands on the FTP client (Page 1953)

Structure of the data blocks (file DBs) for FTP services - FTP server mode (Page 1955)

Layout and structure of the FTP configuration list

Layout and structure of the FTP configuration list for generating the file allocation table

The fields of the table in the FTP configuration have the following meaning and syntax:

Titles in the table for FTP configuration	"CPU"	"DB"	File name	Comment
Meaning	Assignment of the CPU Can be selected from a drop-down list; cannot be typed in.	Number of the data block Entry can be typed in	File name assigned to the file DB Automatic name proposal; entry can be typed in / modifiable.	Informal comment
Example	cpu1 [PLC_1]	20	cpu1_db20.dat	Measured values in the Plant1 area

Notes on the syntax

- The following applies to the file name of a file DB:
  - The file name begins with "cpuX" (where X=1 for S7-1500);
  - Length: Maximum of 64 characters (including "cpuX");

Note

Keep to the notation (lower case for "cpu" and no leading spaces at the start of the row). Otherwise, the files will not be recognized.

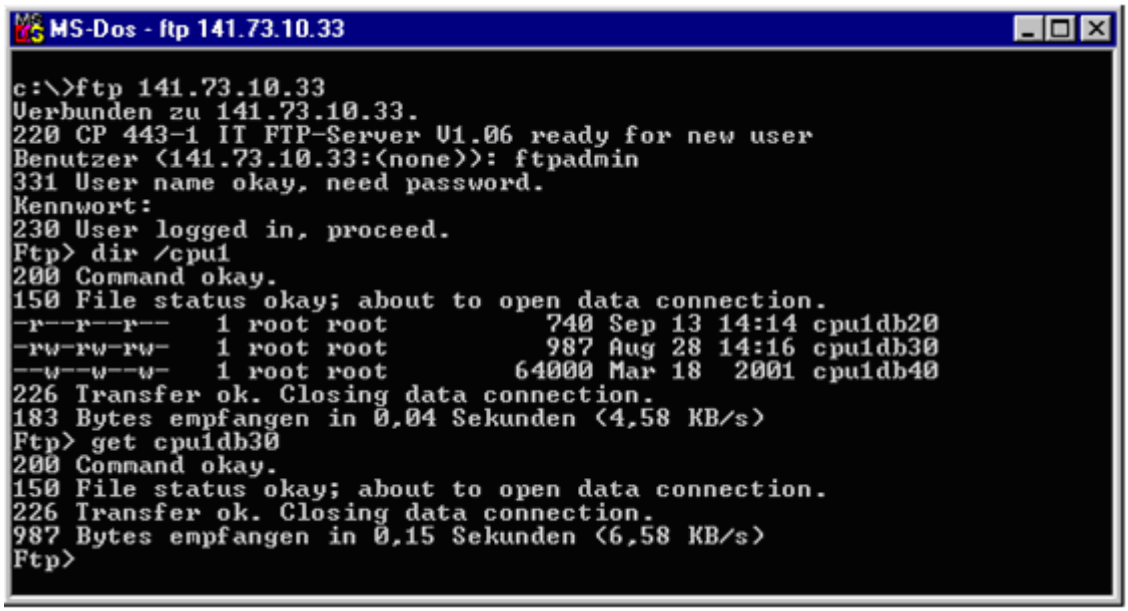
See also

Ethernet CP as FTP server for the CPU data (Page 1950)

FTP commands on the FTP client

Example of access

The following figure shows an example of a typical access sequence in the MS-DOS window.



How typical FTP commands use the permitted FTP functions

The following table shows which FTP commands can be executed to access file DBs on the CPU. The table also shows which FTP functions are used for these FTP commands in typical input consoles such as the MS-DOS prompt.

Table 8-96 FTP commands

Typical FTP functions						FTP command	Meaning
open	dir	put	get	close	del		
x						User	Logging on
x						Pass	Authorization by password
	x	x	x			Port	Specifies the port via which the client wants to transfer.
	x					List	Lists the file DBs in the addressed CPU.
					x	Dele	Deletes a file DB by setting the EXIST bit in the file DB header to "0".

Typical FTP functions						FTP command	Meaning
			x			Retr	Reads the user data in the specified file DB into the specified file on the FTP client.
		x				Stor	Transfers the specified file from the FTP client to the user data area in the specified file DB.
				x		quit	Terminates the current FTP connection.

Note

You cannot use the FTP commands "rename", "append", "rnfr" and "rnto" with the file DBs.

How FTP commands are executed on the CP

To illustrate how the FTP interface to the file DBs on the CPU works, the execution sequence is explained below based on the example of the stor command.

The FTP server on the CP executes the sequence shown below:

1. Identification of the addressed file DB based on the entry in the file allocation table.
2. Check of the bits in the file DB header.
   The write is executed only when the following is recognized:
   LOCKED bit = 0
   NEW bit = 0
   WRITEACCESS bit = 1
3. Writes the file content to the user data area of the file DB on the CPU. At the beginning of the write function, the LOCKED bit is set and reset when writing is completed.
4. When the write function is completed, the NEW bit is also set in the file DB header and the current date entered in the DATE_TIME field.
5. The FTP server sends a message about the file transfer event to the FTP client.

Transfer mode for file transfer

File transfer only uses the binary mode. To change to this mode, enter the command "binary" in the input console after logging on.

See also

Ethernet CP as FTP server for the CPU data (Page 1950)

Structure of the data blocks (file DBs) for FTP services - FTP server mode

How it works

To transfer data with FTP, create data blocks (file DBs) on the CPU of your S7 station. These data blocks must have certain structure to allow them to be handled as transferable files by the FTP services. They consist of the following sections:

- Section 1: File DB header (has a fixed length (20 bytes) and structure)
- Section 2: User data (has a variable length and structure)

Data type "FILE_DB_HEADER"

Using the "Add new block" function, you assign the type "FILE_DB_HEADER" to a data block directly.

Requirements in the CPU configuration

Use the following settings to allow FTP access:

- For all data blocks being used as file DBs, disable the "Optimized block access" attribute

File DB header for FTP server mode

Note: The file DB header described here is largely identical to the file DB header for client mode. The differences relate to the following parameters:

- WRITE_ACCESS
- FTP_REPLY_CODE

Parameter	Type	Value / meaning	Supply
EXIST	BOOL	The EXIST bit indicates whether the user data area contains valid data. The retrieve FTP command executes the job only when EXIST=1. • 0: The file DB does not contain valid user data ("file does not exist"). • 1: The file DB contains valid user data ("file exists").	The dele FTP command sets EXIST=0; The store FTP command sets EXIST=1;
LOCKED	BOOL	The LOCKED bit is used to restrict access to the file DB. • 0: The file DB can be accessed. • 1: The file DB is locked.	The stor and retr FTP commands set LOCKED=1 when they are executed. The following function is also possible when writing from the user program: The user program on the S7 CPU can set or reset LOCKED during write access to achieve data consistency. Recommended sequence in the user program: 1. Check LOCKED bit; if = 0 2. Set WRITEACCESS bit = 0 3. Check LOCKED bit; if = 0 4. Set LOCKED bit = 1 5. Write data 6. Set LOCKED bit = 0
NEW	BOOL	The NEW bit indicates whether data has been modified since the last read access. • 0: The content of the file DB is unchanged since the last write access. The user program of the S7 CPU has registered the last modification. • 1: The user program of the S7 CPU has not yet registered the last write access.	After execution, the stor FTP command sets NEW=1 After reading the data, the user program on the S7-CPU must set NEW=0 to allow store to be used again or to be able to delete the file with the dele FTP command.

Parameter	Type	Value / meaning	Supply
WRITE_ACCESS	BOOL	0: The FTP client on the PG/PC has no write access rights for the file DBs on the S7 CPU. 1: The FTP client on the PG/PC has write access rights for the file DBs on the S7 CPU.	During the configuration of the DB, the bit is set to an initialization value. Recommendation: Whenever possible, the bit should remain unchanged! In special situations, adaptation during operation is possible.
ACT_LENGTH	DINT	Current length of the user data area. The content of this field is only valid when EXIST = 1.	The current length is updated following write access.
MAX_LENGTH	DINT	Maximum length of the user data area (length of the entire DB less 20 bytes header).	The maximum length should be specified during configuration of the DB. The value can also be modified by the user program during operation.
FTP_REPLY_CODE	INT	This parameter is irrelevant in FTP server mode.	Is set to "0" by the FTP server.
DATE_TIME	DATE_AND_TIME	Date and time of the last modification to the file. The content of this field is only valid when EXIST = 1.	The current date is updated following a write access. If the function for forwarding the time of day is used, the entry corresponds to the time that was passed on. If the function for forwarding the time of day is not used, a relative time is entered. This time relates to the startup of the IT-CP (the initialization value is 1.1.1994 0.0 (midnight)).

See also

Ethernet CP as FTP server for the CPU data (Page 1950)

CP as FTP client for the S7 CPU data

How the FTP client works

Principle of the FTP client

To transfer data with FTP, create data blocks (file DBs) on the CPU of your S7 station.

The user program sends FTP jobs that are executed by the CP 1543-1 as an FTP client. In the user program, use the FTP_CMD instruction for connection establishment and data transfer.

Among other things, you specify the following in the job:

- IP address of the FTP server,
- storage location of the file on the FTP server

- File names
- Access information

The following schematics illustrate how the function works when using the FTP_CMD instruction.

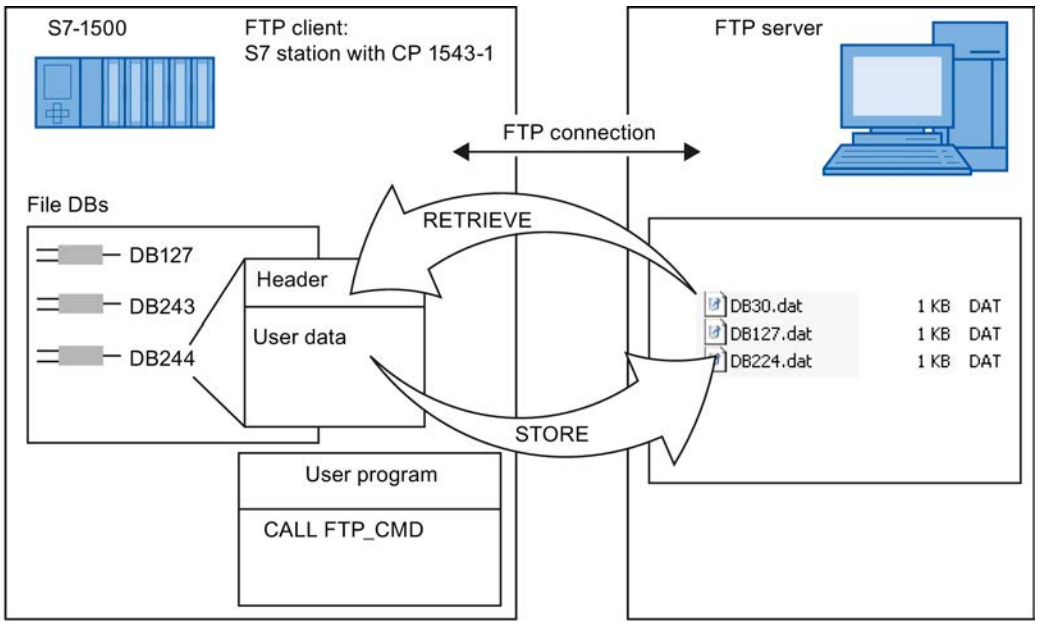


Figure 8-56 CPU with CP 1543-1 as FTP client for the S7 CPU data

Requirements in the CPU configuration

Use the following settings to allow FTP access:

- In the configuration data of the CPU in "Properties > General > Protection": Disable the "Disable PUT/GET communication" option.
- For all data blocks being used as file DBs, disable the "Optimized block access" attribute

Further information

See also section Auto-Hotspot.

FETCH/WRITE mode

Use of FETCH/WRITE

FETCH/WRITE

You can use the FETCH/WRITE services to access system memory areas of the S7-CPU.

- FETCH: Read data directly
- WRITE: Write data directly

Access is possible from the following devices:

- SIMATIC S5
- SIMATIC PC stations
- Non-SIMATIC devices

System memory

You can use FETCH or WRITE to access the following operand ranges in the SIMATIC S7 system memory:

- Data blocks (DB)
  (Note the following limit for DB access: the highest DB no. is 255)
- Bit memory (M)
- Process image input (PII)
- Process image output (PIQ)
- I/O area inputs (PIW, PID, PIB)
- I/O area outputs (PQW, PQD, PQB)
- Counters (C)
- Timers (T)

Requirements in the CPU configuration

Use the following settings to allow FETCH/WRITE access:

- In the configuration data of the CPU in "Properties > General > Protection": Disable the "Disable PUT/GET communication" option.

Link to non-SIMATIC systems

The FETCH and WRITE mode that is supported for ISO transport connections and ISO-on-TCP or TCP connections can in principle be used to access S7 system memory areas from any non-SIMATIC devices.

To be able to use this type of access, for example for PC applications, you need to know the PDU structure of the jobs. Refer to the following section.

See also

Configuring FETCH/WRITE access (Page 1960)

Linking to other systems with FETCH/WRITE (Page 1961)

Configuring FETCH/WRITE access

Reference

Parameter assignment for the FETCH/WRITE mode in the "Properties > General > Options" parameter group

FETCH/WRITE configuration

To enable FETCH/WRITE access, the option first needs to be selected.

Following this, enter the access identification in the display lists. Access identification is possible with the following entries:

- TSAP (Transportation Service Access Point)
  For access over ISO transport or ISOonTCP connections, TSAPs need to be specified. FETCH/WRITE accesses whose TSAP information matches an entry in the list are permitted.
- TCP port
  For access via TCP connections, TCP ports need to be specified. FETCH/WRITE accesses whose port information matches with an entry in the list are permitted.

"S7 addressing mode" option

You can select the addressing mode when you configure FETCH/WRITE mode. This defines how the address information in the FETCH/WRITE call in the SIMATIC S7 station will be interpreted when DBs are accessed:

- S7 addressing mode: Byte address
- S5 addressing mode: Word address

This allows applications to access S5 or S7 stations without modifying the addresses. This is useful for existing S5 applications, for example, that can now be used unchanged when accessing S7 stations.

The addressing mode for accessing SIMATIC S7 (option selected) is set by default.

See also

Use of FETCH/WRITE (Page 1959)

Port settings (Page 1961)

Port settings

Port settings

The following table provides information about the values and uses of the port addresses:

Port addresses		Use/note
0		Permanently assigned; must not be used!
1..1023		Assigned by default, should not be used (well-known ports)
1024..49151	Ports for application-specific protocols	
	from 2000 to 5000	Range in which the configuration tool searches for and allocates a free port address. You can set the port address in this range individually.
	5001..49151	The port addresses above 5000 are used by the system! Note: If you want to use these port addresses, please contact your system administrator to verify availability.
49152..65535		Dynamically assigned ports The use of these ports is not recommended.

Linking to other systems with FETCH/WRITE

In principle, the mode for FETCH and WRITE can be used by any third-party devices to access S7 system memory areas.

To be able to use this type of access, for example for PC applications, you need to know the PDU structure of the jobs. The required S7 or S5 headers for request and response frames are 16 bytes long and their structure is described in this chapter.

a) Structure of WRITE frames

The meaning and values of parameters shown without values in the following table can be found in the section "Parameter values".

WRITE request frame			WRITE acknowledgment frame		
0	System ID	="S"	0	System ID	="S"
1		="5"	1		="5"
2	Length of header in bytes	=16d.	2	Length of header	=16d.
3	ID OP code	=01	3	ID OP code	=01
4	Length OP code	=03	4	Length OP code	=03
5	OP code	=03	5	OP code	=04
6	ORG field	=03	6	Ack field	=0Fh
7	Length ORG field	=08	7	Length ack field	=03
8	**ORG ID**		**8**	**Error field**	**=No**
9	**DBNR**		**9**	**Empty field**	**=FFh**
A	**Start address**	**High byte**	**A**	**Length empty field**	**=07**

B		Low byte	B		
C	Length	High byte	C		
D		Low byte	D	free	
I	Empty field	=FFh.	I		
F	Length empty field	=02	F		
	Data field up to 64 K				

a) Structure of FETCH frames

The meaning and values of parameters shown without values in the following table can be found in the section "Parameter values".

FETCH request frame			FETCH response frame		
0	System ID	="S"	0	System ID	="S"
1		="5"	1		="5"
2	Length of header	=0x10	2	Length of header	=0x10
3	ID OP code	=0x01	3	ID OP code	=0x01
4	Length OP code	=0x03	4	Length OP code	=0x03
5	OP code	=0x05	5	OP code	=0x06
6	ORG field	=0x03	6	Ack field	=0x0F
7	Length ORG field	=0x08	7	Length ack field	=0x03
8	ORG ID		8	Error field	=No
9	DBNR		9	Empty field	=0xFF
A	Start address	High byte	A	Length empty field	=0x07
B		Low byte	B	free	
C	Length	High byte	C		
D		Low byte	D		
I	Empty field	=0xFF	I		
F	Length empty field	=0x02	F		
				Data up to 64 K but only if Error no. =0	

Parameter values

S7 address area	DB	M	I	A
ORG ID	01_H	02_H	03_H	04_H
	Source/dest. data from/to data block in main memory	Source/dest. data from/to bit memory area	Source/dest. data from/to process image of the inputs (PII)	Source/dest. data from/to process image of the outputs (PIQ)

DBNR	DB, from which the source data is taken or to which the dest data is transferred	irrelevant	irrelevant	irrelevant
permitted range	1...255			
Start address	DW number from which the data is taken or written to	Memory byte no., from which the data is taken or written to	Input byte no. from which the data is taken or written to	Output byte no., from which the data is taken or written to
permitted range	0...2047	All memory bytes made available by a CPU.	0...127	0...127
Length	Length of the source/dest. data field in words	Length of the source/dest. data field in bytes	Length of the source/dest. data field in bytes	Length of the source/dest. data field in bytes
permitted range	Up to 8192 bytes	Up to 8192 bytes	1...128	1...128

S7 address area	PI/PQ	C	T
ORG ID	05_H	06_H	07_H
	Source/dest. data from/to I/O modules. With source data input modules, with dest data output modules	Source/dest data from/to counter cells	Source/dest data from/to timer cells
DBNR	irrelevant	irrelevant	irrelevant
Start address	I/O byte no., from which the data is taken or written to	Number of the counter cell from which the data is taken or written to	Number of the timer cell from which the data is taken or written to
permitted range	0...127 digital I/O 128...255 analog I/O	0...255	0...255
Quantity	Length of the source/ dest. data field in bytes	Length of the source/ dest. data field in words (counter cell = 1 word)	Length of the source/ dest. data field in words (counter cell = 1 word)
permitted range	1...256	1	1

See also

Use of FETCH/WRITE (Page 1959)

CP PROFIBUS

Time-of-day synchronization

Reference

Setting the properties for PROFIBUS CPs in the "Properties > Settings > Time-of-day synchronization" parameter group.

Time-of-day synchronization

Decide whether or not the CP will forward time-of-day frames.

The two following situations are possible depending on where the time-of-day master is located:

- Time-of-day frames come from the subnet and are forwarded to the station.
- Time-of-day frames come from the station and are forwarded to the subnet (LAN).

With some of the newer modules, you explicitly specify the direction in which the time-of-day frames are forwarded. With modules for which forwarding can simply be enabled or disabled, you can assume that the CP can forward a time-of-day frame in either direction.

If you have several CPs in your station, take into account the flow of time-of-day frames depending on the time-of-day master because you can forward time-of-day frames from one network to another network using the function described here making conflict situations possible.

If there is more than one CP in a station connected to the same network, only **one** of these CPs is allowed to forward timeofday synchronization messages.

Field device parameter settings (data record routing)

Reference

Setting the properties for PROFIBUS CPs in the "Properties > Settings > Field device parameter settings" parameter group.

Field device parameter settings (data record routing)

if you select this option, you can use the CP as a router for data records intended for field devices (DP slaves). Data records transferred by devices that are not connected directly to PROFIBUS and that therefore have no direct access to the field devices (DP slaves) can then be forwarded by the CP to the field devices.

SIMATIC PDM (Process Device Manager) is a tool that creates data records of this type for assigning parameters to field devices.

As default, this function is enabled. Since the function requires additional memory resources, you can turn off the option if you are putting a high load on the memory resources of the CP as a whole (connections etc.) and do not require the "Data record routing" function.

CPs with DP functionality - Selecting the mode

Reference

Setting the CP mode for PROFIBUS DP in the "Properties > General > Operating mode" parameter group.

Relationship between CP mode and CP functionality

Refer to this table for the configurable CP modes and resulting CP functionality:

CP mode	Possible CP functions				
	PG/OCM via PROFIBUS	FDL via PROFIBUS	DP master	DP slave	Configured S7 connections
No DP mode	X	X	-	-	X
DP master mode	X	X	X	-	X
DP slave active (see below)	X	X	-	X	-
DP slave passive (see below)	-	-	-	X	-

Key: "X" = Function supported; "-" = Function not supported

DP master mode / "DP delay time" parameter

Here, you can set an additional delay for the updating of the DP I/O data. This delay is used by the PROFIBUS CP for processing other protocols, for example FDL connections.

The displayed DP reaction times are only updated when you take one of the following actions:

- Click the "Recalculate..." button or
- Select a different parameter group

If a lot of DP slaves are configured, this recalculation can take several seconds.

DP master class 2 (diagnostic functions - can only be selected with certain CP types)

In this mode, the device can read input and output data from any DP slaves not assigned to it via the PROFIBUS CP.

This means, for example, that a process signal can be acquired by several DP masters and can help to save sensors in the field (shared input / shared output). Programming, diagnostic or management devices typically operate as DP class 2 masters.

DP slave mode / "Test, Commissioning, Routing" option (the name of the function can vary depending on the CP type)

- Option enabled: DP slave active
  Select the check box if, for example, you want to use PG functions regularly over this interface for commissioning and testing. This selects the "DP slave active" CP mode. (In other words, the interface is involved in the PROFIBUS token rotation).
  The following functions are then possible (the name of the function can vary depending on the CP type):
  - Programming (for example loading)
  - Testing (status/modify)
  - S7 routing (I slave as gateway)
  - S5-compatible communication (SEND/RECEIVE interface)
  - S7 communication (only in server mode on connections configured at one end without communication blocks, configuration on partner)

  Note

  Special effects of the "DP slave active" mode
  - S7 connections for data exchange using communication blocks cannot be configured in this mode
  - This can extend the token rotation time. In time-critical applications and when S7 routing and client functionality are not required for communication, this option should therefore not be enabled.

- Option disabled: DP slave passive
  If the "Test, commissioning, routing" box is not checked, the "DP slave passive" CP mode is selected and the interface only operates as a server for communication services and is not involved in the token rotation on PROFIBUS.
  Server functionality means that only the partner (for example PG, OP or automation system) can initiate communication. Note the following features when the option is not enabled:
  - S7 routing is not possible.
  - Programming and testing are possible but slow (no extension of the bus token rotation time).

CP supports several DP slave types

Reference

Setting the DP slave type for PROFIBUS DP in the "Properties > General > Operating mode" parameter group.

DP mode

On CPs that support different DP slave types, you can select the following DP modes depending on the CP type:

- DPV1 functionality (default mode)
  This includes DP slaves complying with the PROFIBUS DPV0 and DPV1 standard as well as Siemens DP slaves.
  This mode is only possible with certain CPU types. Refer to the information in the S7-CP manual for your PROFIBUS CP.
- DPV0 compatible
  This includes DP slaves complying with the PROFIBUS DPV0 standard; DP slaves complying with the DPV1 standard can only be used with restricted functionality.
- S7-compatible
  This includes DP slaves complying with the PROFIBUS DPV0 standard and Siemens DP slaves; DP slaves complying with the DPV1 standard can only be used with restricted functionality.

CP/CM PROFIBUS

Setting parameters for the clock / time-of-day synchronization

Time-of-day synchronization using the CM

With the time-of-day synchronization parameters, you specify the role of the CM in the forwarding of time-of-day frames.

Depending on the mode of the device on PROFIBUS, the following types of synchronization can be set:

- CM is passive node on PROFIBUS (for example DP slave): Slave, none
- CM is active node on PROFIBUS: Master, slave, none

Meaning of the type of synchronization of the time-of-day

- Slave:
  The CM clock is synchronized with another clock. The synchronization frames come from PROFIBUS. With this type of synchronization, the CM forwards the time-of-day frames to the station at a fixed interval.
  Within the S7 station, there may be only one time-of-day slave.
- Master:
  The CM clock sends synchronization frames to PROFIBUS.
  If the synchronization type "Master" is set, the interval for outputting synchronization frames to PROFIBUS can be selected.
- None:
  There is no synchronization.

If more than one module with a clock is present within a network, you need to specify which module will act as the master and which as the slave for time-of-day synchronization purposes. There can only be one time-of-day master.

- Update interval
  With the synchronization type "Master", the update interval defines the interval at which the time-of-day frames are output.

CPs settings that do not depend on the protocol or the network

Replacing a module without a PG

Reference

Making special settings in the "Properties > General > Settings" parameter group

Depending on the CP type, you will see the option "Module replacement without PG"

Effect - nonvolatile, long-term storage on the CPU

If you have selected this option, your configuration data is stored long-term in nonvolatile memory on the CPU instead of on the EEPROM of the CP. Remember, however, that even on the CPU such data is only retained long-term in the event of a failure if the CPU has a backup battery or an S7 memory card is used.

Resource requirements on the CPU

If you select this option, you occupy additional resources on your CPU. When you load the user programs and the configuration data, you will be informed if there is not enough storage space available. Resource bottlenecks can be avoided by using an S7 memory card.

Note

If you are faced with a lack of resources and do not want to use an S7 memory card, you can also deselect this option and store the configuration data on the CP. Later, you can then create an S7 memory card with the configuration data so that the "Module replacement without PG" option is enabled. If you then insert the S7 memory card in the CPU, you can replace the CP at any time. The configuration data is then loaded from the CPU or from the S7 memory card automatically when the CP starts up.

Multiplexing OP connections / occupying internal CPU connection resources

Reference

Making special settings in the "Properties > General > Settings" parameter group

Depending on the CP type, you will see the option "Multiplex OP connections"

Multiplex mode

To connect TD/OPs and/or HMI devices, you can optimize the connection resources in the S7-300 CPU by allowing up to 16 such devices to communicate on a single CPU connection resource (multiplex mode).

If you do not use multiplex mode, the number of TD/OPs and/or HMI devices that can be used will depend on the number of available connection resources of the CPU used.

Use of internal CPU connection resources

Configured S7 connections via the CP use the same multiplex channel that you use in multiplex mode for HMI connections. This means that when you configure S7 connections, you already occupy one CPU connection resource.

Note

PG connections are not operated via the multiplexer; one connection resource is always occupied when you operate a PG.

Note

Addressing in multiplex mode

In multiplex mode, the rack/slot assignment of the CP must be specified instead of the rack/slot assignment of the CPU when assigning addresses for TD/OP/HMI connections.

Applications (for example ProAgent) that require block-related alarms (Alarm_S: SFC 17 , SFC 18 , SFC 19) are not supported in multiplex mode.

PC station

SIMATIC NET PC software for PC communications processors

SIMATIC NET PC software for PC communications processors

The use of the SIMATIC NET PC software versions listed below is recommended for the following communications processors depending on the operating system:

- CP 1613, CP 1613 A2, CP 1623, CP 1628
- CP 5613, CP 5613 A2, CP 5614, CP 5614 A2, CP 5623, CP 5624

You should use the following versions of the SIMATIC NET PC depending on the operating systems:

- Windows 8 (Professional, Enterprise), 32- and 64-bit and Windows Server 2012
  DVD "SIMATIC NET PC Software" V12
- Windows 7 (Professional, Enterprise, Ultimate), 32- and 64-bit and Windows Server 2008 R2
  DVD "SIMATIC NET PC Software" V12
- Windows XP, Windows Server 2003 and Windows Server 2008, 32-bit
  CD "SIMATIC NET, PC Software, Edition 2008 + SP6", version number V7.1.6

PC station - user applications OPC server / application

OPC server

General protocol-specific settings

General

Cycle time

Effects of the selectable cycle time

The cycle time influences how often the OPC server updates the values of the OPC items in its cache using a new communications job and how often it checks for changes. The cycle time therefore influences the performance, the communication load, and the processor load.

If the value for the cycle time is set too low, this increases processor load unnecessarily!

Relationship between the cycle time and update interval (UpdateRate)

The update interval (UpdateRate) parameter that you can set in the user program specifies the shortest interval for checking the values of the OPC items of an active OPC group. The server checks whether or not the values of the OPC items have changed at the earliest when this interval has elapsed. This limits the communication load.

The update intervals used by the OPC server for SIMATIC NET are multiples of the cycle time specified here during configuration. The shortest possible update interval is the cycle time.

Relationship between the cycle times set for specific protocols

Since the SIMATIC NET OPC Server can use variables of different protocols at the same time,

the minimum update interval of the OPC Server is the highest value set for the cycle time for the active protocols (protocols for which connections are configured).

Example:

A cycle time of 100 ms is configured for S7. No other protocol is being used. 100 ms is used by the OPC server as the minimum update time. The DP protocol is now added and the cycle

time configured for this protocol is 50 ms. Result: The OPC server now uses the value 50 ms as the minimum update time for all protocols.

Note: The response described results from the requirements of the OPC standard. The OPC standard specifies that data changes must not be signaled more quickly than specified by the minimum update rate.

Actual update interval

The actual update interval (UpdateRate) that can be achieved during operation depends on various factors including:

- The underlying communication protocol
- The communication partner
- Number and type of items

As a result, the actual update interval that can be achieved maybe longer than the minimum value configured here.

If the process value changes several times within the actual update interval, some value changes will not be forwarded to the application.

Note:

The data throughput with asynchronous and synchronous jobs does not depend on the update interval parameter (UpdateRate) and the cycle time set in the configuration.

Access protection

Enabling access protection

Reference

Settings for the OPC server / "Properties > General" parameter group; a protocol-specific node is selected.

Meaning of access permissions for OPC variables

With the OPC server, access permissions to individual variables or groups of variables can be specified for a specific protocol. By specifying access permissions, you can, for example, prevent an internal controller calculation variable from being overwritten.

- Default rights
  For OPC items that are not specifically named, you can specify a default value here:
  - RW: Write and read access
  - R: Read-only access
  - W: Write-only access
  - NONE: Neither write nor read access
- Rights specific to OPC items

If you select this option, you can assign access permissions explicitly for certain OPC items in the input box.

See also

Syntax - Assigning access rights to specific OPC items (Page 1972)

Syntax - Assigning access rights to specific OPC items

You can enter access rights for one or more OPC items.

The default rights defined for the other OPC items are not affected by these values.

The syntax for OPC item-specific access rights is defined as follows:

<OPCItem>=<rights>

where:

- <OPCItem>
  You specify one or more OPC items according to the syntax defined in the OPC documentation. It is also possible to use aliases.
  The following placeholders can be used:
  * Any number of irrelevant characters
  ? Exactly one character is irrelevant
- <rights>

RW	Write and read access
R	Read-only access
W	Write-only access
NONE	Neither write nor read access

Rules for evaluation

- The specific rights assigned here have priority over the default rights you assigned under Access rights.
- The specific rights assigned here are evaluated in the order shown here. If there is a multiple assignment to an OPC item, the last assignment is always valid.
  Example:
  DP:[CP 5611]Slave040_QB*=RW
  DP:[CP 5611]Slave040_QB1=R
  DP:[CP 5611]Slave040_QB2=W
  DP:[CP 5611]Slave040_QB1=W
  DP:[CP 5611]Slave040_QB1*=R
  Due to this entry, the following access rights are effective:
  DP:[CP 5611]Slave040_QB2=W
  DP:[CP 5611]Slave040_QB1=R
  Tip: Please read the detailed description of the syntax in the OPC documentation listed below. There, you will find a series of examples - also for applications such as PROFINET IO.
- Leading zeros in items and in the definitions made here for the access rights are always ignored.
  The following example illustrates the advantage of this:
  You configure the following access right: Slave004_QB0=R
  You have therefore implicitly assigned the access right to the following items:
  - Item 1: Slave00004_QB000
  - Item 2: Slave4_QB00,1 (note: the syntax ",1" is demanded by the OPC Scout)

  When using the wildcards "*" and "?", this means the following:
  Do not use these wildcards at positions at which a leading zero can precede an item!
  Example: Leading zeros in items - example with wildcards:

Further information

Read the detailed description of the syntax in the OPC documentation. There, you will find a series of examples - also for applications such as PROFINET IO.

Leading zeros in items - example with wildcards:

You are using the following item:

- Slave00004_QB000

An access right definition that would fail in the comparison might be:

- Slave0?4_QB0=R

Reason:

In the comparison, a leading "0" is found in the item, the item is therefore reduced to: Slave4_QB

Due to the wildcard "?", the leading "0" in the definition is no longer detected as a leading "0". The definition for the access right is therefore not reduced and the comparison fails.

Remedy:

To achieve the required rights for this example, the following two definitions would be necessary:

- Slave?4_AB0=R
- Slave4_AB0=R

S7-specific settings

OPC symbols

Using symbols

Reference

Settings for OPC server / "Properties > S7 > OPC symbols" parameter group.

Using symbols (as of OPC server V6.1)

Here, you can specify whether symbols from the symbol tables configured for a CPU will be included in the system data for the OPC server. This is necessary before user applications (OPC client) can access variables symbolically via the OPC server.

The symbol tables of the CPUs to which S7 connections are configured for the OPC server are used. Only symbols that relate to data blocks (DBs) are taken into account.

The available alternatives have the following meaning:

- None
  No symbols from STEP 7 are included in the system data for the OPC Server. The access rights configured for the OPC items under Access protection apply.
- All
  All relevant symbols from STEP 7 are included in the system data for the OPC server. Making this selection sets the access rights for all symbolic access to the assigned variables to Read/Write (RW). These rights have priority over the access rights for the OPC items set under Access rights.

- Configuring
  All selected and specifically configured symbols from STEP 7 are adopted in the system data for the OPC server. With the "Configure" button, you open the "Symbol configuration" dialog in which you can select and configure the symbols.
- Array elements visible at runtime (selectable as of OPC server V8.0)
  Here, you can decide whether you want the elements of arrays to be visible on the OPC server during runtime. This relates to the arrays configured as symbols.
  As default, the option is disabled because making array elements visible causes a high load on the OPC server. With older OPC server versions (< V8.0), array elements are always visible and this setting cannot be disabled.

Note

Consistency

If you select the "All" option, the current CPU symbol tables are taken into account when you save and compile.

If you use the "Configured" option, remember that subsequent changes in the symbol tables have the following effects in the system data for the OPCserver:

- Newly added symbols are ignored
- Selected symbols that were removed later in the symbol table cannot be saved.

Note

Bit arrays

Symbols that include data of the "bit array" type are not included in the system data of the OPC server!

Note

Note on the procedure

While the symbols are being adopted as described here, no FB in which symbols are used may be open in LAD / STL / FBD. Otherwise some symbols will not be read in.

Symbol configuration

Reference

Settings for OPC server / "Properties > S7 > OPC symbols" parameter group.

Calling the symbol configuration with the "Configure symbols" button.

Overview

In this dialog, you can select and configure the symbols to be adopted in the system data of the OPC server.

The CPUs relevant for the selection of symbols are those to which S7 connections are configured for the OPC server.

In the navigation area on the left, you will see the relevant stations and CPUs along with any structure elements. In the content area on the right, you will see the symbols defined for the selected CPU or the selected structure element. The assigned data type and the currently configured properties are shown for each symbol. These properties can be configured in the lower part of the dialog.

Selecting and configuring symbols

1. First select the symbol group of a CPU or a structure element in the navigation area.
2. Then select one or more symbols with the mouse cursor in the content area.
3. Configure the properties of the selected symbols in the lower part of the dialog as described below.

"Elements in active branch" display

The meaning of the display is as follows:

- Symbols: Total number of elements in the selected branch or in the selected structure element (a selected structure element is included in this number).
- Array elements: Number of elements in the array
  - Value shown on the left: Number of elements selected as visible
  - Value shown on the right: Total number of elements

Configurable properties

- "Visible" option
  If you disable this option, the selected symbol is excluded from the download to the OPC server.
- "Download only visible symbols" option
  If you select the option, only the symbols marked as "visible" are adopted in the system data of the OPC server and are therefore visible to the OPC client. The option reduces the time for the download to the OPC server considerably if large numbers of symbols are marked as "not visible".
- Access permissions
  From the drop-down list, you can assign the following access permissions to the selected symbols:
  - RW: Read and write
  - R: Read
  - W: Write
  - None: No access
    The access permissions selected here have priority over the access permissions configured for the OPC items in access protection.

- Connection
  From the drop-down list of the configured connections, select the S7 connection via which the symbolic variable access will be made.
- "Edit EU Lo / Hi" input boxes - Specify tolerance range for measured values (EU Lo and EU Hi limit values)
  Here, you specify limit values for calculating a deadband for a selected OPC item. The input boxes are enabled under the following condition:
  - You have only selected a single symbol;
  - An elementary data type is assigned to the symbol that permits a numeric representation (INT, DINT, REAL, BYTE, WORD, DWORD).
    Note:
    This entry is possible only for OPC servers as of version 6.4.
    See also Entering limit values for EU Lo and EU Hi (Page 1977)

"Access path" display

The storage path of the selected object in the current STEP 7 project is displayed in the navigation area.

Special features

- Bit arrays are not adopted
  Symbols that include data of the "bit array" type are not adopted in the system data of the OPC server. This restriction applies although this symbol is displayed for selection. This restriction applies regardless of whether you set the "visible" option (see above) for these symbols.
- The data types "Array of UDT" and "Array of Struct" are displayed differently
  With symbols of the complex data type "Array of UDT" or "Array of Struct", the structure of the "UDT" or "Struct" data type is only displayed once in the left-hand area of the dialog. The "Array" entry in the "Data type" box on the right indicates that this involves an array. In contrast to this, other tools, such as the Symbol File Configurator display multiple array elements.
  The data structures involved are nevertheless identical.
  Note that the setting for the "Visible" option described above applies to all elements of an array including array elements that are not displayed.

Entering limit values for EU Lo and EU Hi

Entering the tolerance range for measured values (limit values EU Lo and EU Hi)

Here, you specify limit values for calculating a deadband for a selected OPC item (for the meaning, see below).

You can specify numeric values as integers, floating points or in exponential notation (scientific notation)

Note: As separator use the "." character instead of the "," character.

- EU Lo (Engineering Unit Low)
  Low limit
- EU Lo (Engineering Unit Low)
  High limit

the following must apply: EU Lo <= EU Hi

If an input box is left empty it is automatically completed with "0".

Make sure that you keep to the value ranges depending on the data type. Range of values of the data types (Page 1979)

- Examples:
  - EU Lo = 10 EU Hi = 25
  - EU Lo = 4.5 EU Hi = 14.5
  - EU Lo = 1.2e-2 EU Hi = 2.5e-2
- Notes:
  - The tolerance range ("Percent Deadband") applies to a group of OPC items.
  - This entry is possible only for OPC servers as of version 6.4.

Meaning of EU Lo and EU Hi

Fo rmeasured values, you can specify deadbands for measured value acquisition. Within such a percent deadband, a measured value can change without the change being signaled.

To achieve this, a relative value is calculated from the specified parameters according to the following formula. This value specifies the maximum number of units by which a physical value can change up or down without a value change being signaled.

A = PD / 100 x (EU HI – EU Lo)
---

where:

A = amount of the deviation (up or down)

PD = Percent Deadband / %

EU Lo = Engineering Unit Low

EU Hi = Engineering Unit High

Example

With a temperature measurement, you assume that there is a typical deadband for the measured value in the range between 30°C and 40°C. As the percent deadband, set 10%.

You would therefore select the following values:

EU Lo = 30

EU Hi = 40

PD = 10%

This results in: A = 10 / 100 x (40 – 30) = 1

A temperature change is therefore signaled when the temperature is more than 1 °C above or below the previous measured value.

See also

Symbol configuration (Page 1975)

Range of values of the data types

The following ranges of values apply to the data types used:

BYTE= 0 .. 256

INT= -32768 ... 32767

WORD= 0 ... 65535

DWORD= 0 ... 4294967295.0

DINT= -2147483648 ... 2147483647.0

REAL POS UPPER = 3.402823e+38

REAL POS UNDER = 1.175495e-38

REAL NEG UPPER = -1.175495e-38(negative high limit)

REAL NEG UNDER = -3.402823e+38 (negative low limit)

See also

Entering limit values for EU Lo and EU Hi (Page 1977)

OPC alarms:

Configuring OPC alarms

Reference

Settings for OPC server / "Properties > S7 > OPC alarms" parameter group.

Language / message text language setting

For OPC alarms, you specify which language versions of the text library for alarm texts (block- and symbol-related alarms) will be downloaded to the OPC server.

Using the two check boxes, you can set the following:

- The default language (specified as the default language for display devices):
  When the default language for alarms is one of the following STEP 7 default languages: German (Germany), English (USA), French (France), Spanish (Traditional Sort), Italian (Italy). Otherwise the check box is disabled.
- English (USA)
  (if English (USA) is installed as the default language)
  If the default language for display devices is already English (USA), the check box is disabled.

You can select one or both options. If you select both options, a check is made as to whether multiple language text storage was selected in "Language for display devices" in the project. If this is the case, the relevant text libraries are downloaded when you download to the PC station.

Time stamps for alarms

In this parameter group, you decide which entity will set the time stamp of an alarm. If you leave time stamping to the CPU, you obtain alarms with the exact time of their occurrence. With the option of specifying a time offset, you can also register alarms occurring, for example, in plants with different time zones with a common reference time in the acquisition system.

- CPU time stamp
  Alarms are given the time stamp of the signaling CPU. Exception: StatepathEvents are locally generated alarms and are given the PC time (UTC).
- CPU time + offset
  Alarms are given the time stamp of the CPU sending the alarm; a selectable time offset accurate to a second is added to the time stamp. Exception: StatepathEvents are locally generated alarms and are given the PC time (UTC).
  Range of the time offset: -23:59:59 to + 23:59:59
- PC time (UTC)
  Alarms are given the time stamp (UTC) at the time of arrival at the receiving PC station (OPC server).

Assignment table for OPC alarms

Assignment table for OPC alarms

Reference

Settings for OPC server / "Properties > S7 > Assignment table OPC alarms" parameter group.

Meaning

The alarm classes configured in the project in "Common data > Alarm classes" are displayed in a table. The OPC alarm category "OffNormal" is assigned to the message classes as default. If necessary, assign a different OPC alarm category.

Optimized S7 access to PLC data for OPC servers

To allow optimized access by the OPC server V12 to data blocks, bit memory, inputs, outputs, counters and timers of a SIMATIC S7-1500 and SIMATIC S7-1200 that are not type-correct, you can configure an optimized S7 connection to the CPU.

S7 connections for optimized data access to an S7-1500 or S7-1200 are available only using Industrial Ethernet.

Requirements

- STEP 7 Professional V12 SP1
- OPC server V12
- S7-1500 or S7-1200 V4.0 or higher

Connection names for OPC server V12 and S7 connections

When you create an S7 connection for an OPC server, STEP 7 proposes a valid S7 connection name. You can change the S7 connection name at any time. Some characters and connection names are used by the system and cannot be used when assigning the connection name. STEP 7 checks for invalid characters in the connection name automatically.

Connection table

In the table network view in the "Connections" tab, the connections you have created and their parameters are displayed in the connection table. In the "Partner" column, you can change the communications partner to the OPC server V12.

Parameters of special connection properties

In the Inspector window of the selected S7 connection configured at one end (one-way), go to "Properties > Configuration > Special connection properties", the "One-way" and "Active connection establishment" check boxes are selected. You cannot make any changes in the "Special connection properties" parameter group.

Address details

In the Inspector window of the selected S7 connection configured at one end in "Properties > Configuration > Address details" you will see the TSAPs of the communications partners in the connection resources. The local TSAP is structured as follows:

SNOPCCxxxxyyyyyy

- SNOPCC = SIMATIC NET OPC Configured connection
- xxxx = index of the OPC server V12
- yyyyyy = connection ID of the S7 connection

The partner TSAP is "SIMATIC-ROOT-OTH".

You cannot make any changes in the "Address details" parameter group.

OPC parameters

In the Inspector window of the selected S7 connection configured at one end in "Properties > Configuration > OPC" you have the following configuration options:

- Enabling optimized access
- Configuration of the buffer size for optimized access to tags
- Administration of the optimized access to tags by entering the access password

Buffer size:

It may be useful to change the default buffer size in specific applications with particularly large numbers of data tags on a CPU (S7-1500 or S7-1200 V4.0). Access to contiguous data tags within a data area is optimized to improve performance. Optimum results are achieved with buffer sizes that include at least the majority of the contiguous data tags that are accessed regularly and often.

Example: Many data tags between bytes 1000 and 1999 are read regularly from the data block "db10", more seldom also data tags in byte 30000. In this case, a buffer size of 1000 bytes is adequate.

Connection resources

With configured S7 connections for an OPC server V12 or higher, to allow optimized access to a PLC S7-1500 or S7-1200 V4.0 or higher, the connection resources required for this are displayed on the PLC page. You will find the display of the required connection resources in the properties of the relevant CPU in the Inspector window in "Properties > General > Connection resources".

For S7 connections for optimized access by an OPC server V12 or higher, the resources of such an S7 connection are not assigned to the "S7 communication" row on the PLC but to the rows "Other communication" (S7-1500) or "Free available connections" (S7-1200 V4.0 or higher). For a configured S7 connection for an OPC server V12 to an S7-1500 or S7-1200 V4.0 or higher, more than one connection resource is required.

Replacing a OPC server

With the "Change device" shortcut menu of a selected OPC server you have the option of replacing an OPC server V12 with optimized access to S7-1500 or S7-1200 V4.0 with an OPC server < V12 and vice versa.

If S7 connections had already been configured for the OPC server you are replacing, during the replacement of the server you will receive a message that the existing S7 connections will be deleted. Following replacement, these need to be recreated.

Display of an S7 connection configured at one end (one-way) between an OPC server V12 and an S7-1500 or S7-1200 V4.0 or higher

When an OPC server V12 has sent a successful job for optimized data access to an S7-1500 or S7-1200 V4.0 or higher, this connection is displayed in the online connection table of the corresponding PLCs in STEP 7.

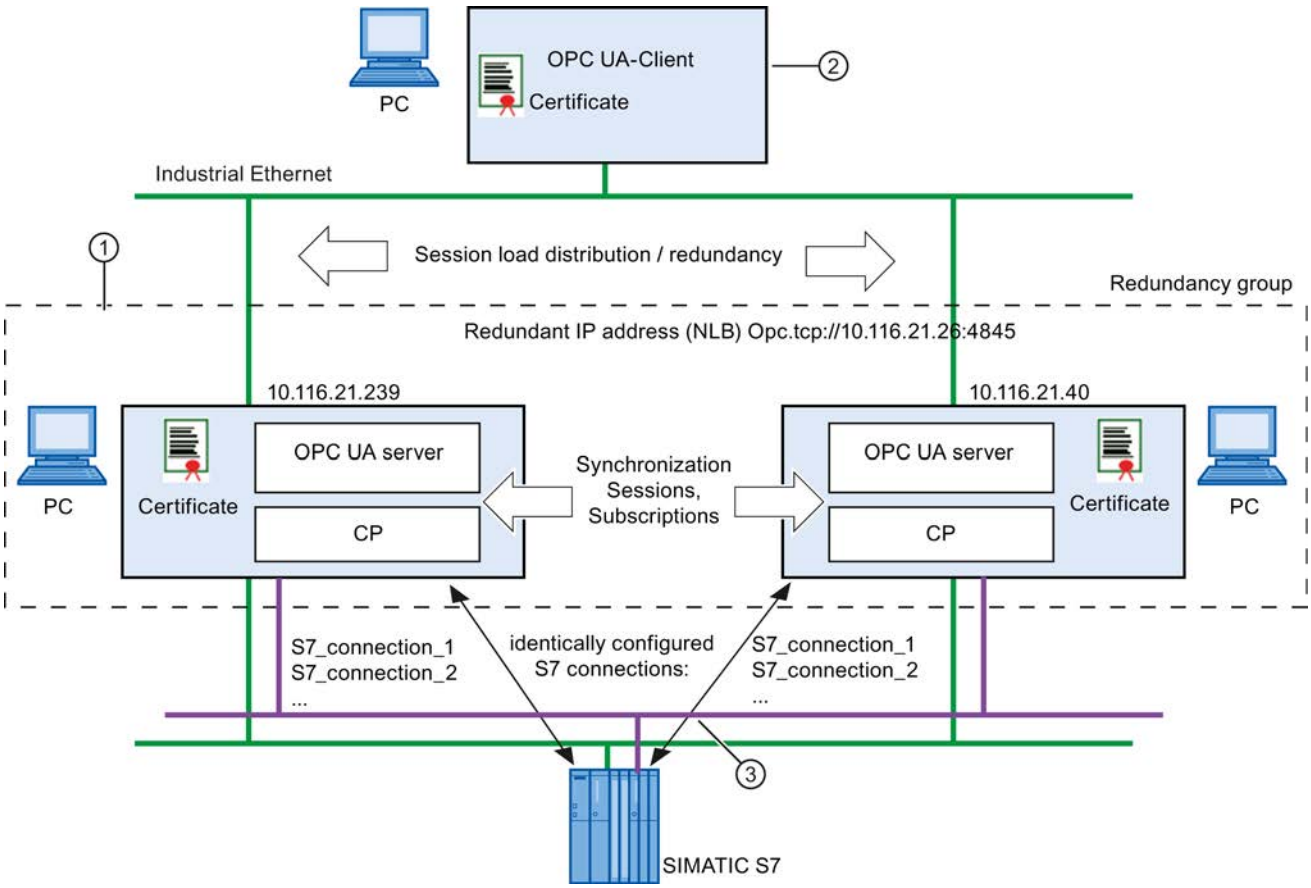
Such connections are also shown in the Inspector window "Connection information" of the selected online CPU: the online connection resources used on the PLC are displayed here.

Redundant OPC UA server

Redundant OPC UA server - basics

Transparent OPC UA redundancy servers

The following figure shows how the OPC UA servers are embedded in a redundancy group and configured for the transparent OPC UA redundancy solution.



① You configure a PC station for each redundant OPC UA server. The redundant PC stations assigned to an OPC UA client form a redundancy group. This assignment is made by the uniform configuration of the "IP address of the redundancy group" for the OPC UA servers.

② The OPC UA client that is using the redundancy is located on a higher-level PC station.
The higher-level PC station does not need to be configured for use of OPC UA redundancy in the STEP 7 project.

③ The communication between the OPC UA servers and the programmable controllers is via multiple, identically configured S7 connections.
Depending on the configuration, S7 connections can be used via Ind. Ethernet or via PROFIBUS.

Setting a SIMATIC NET OPC server as S7 OPC UA server for transparent OPC UA redundancy

You can achieve transparent redundancy to be used by the higher-level OPC UA client by following the steps outlined below during configuration:

1. To operate PC stations with redundant SIMATIC NET OPC UA servers, create a component of the type OPC server >= V8.1 for each station. Enable S7 UA redundancy for these OPC servers.
2. They adopt the virtual IP address. You specified this virtual IP address previously in the network load balancing manager (NLB) for both PC stations. The virtual IP address is mapped to the real IP addresses of the OPC UA server.
3. You specify a port via which the two OPC UA servers can communicate so that they can synchronize their data;
4. You configure identical S7 connections for both OPC UA servers for communication with the connected automation systems;
5. Authentication certificates are generated automatically and assigned to the PC stations. If necessary, assign the two redundant PC stations newly generated authentication certificates.

Consistency check

When you save and compile the configuration, a consistency check is run. In relation to OPC UA redundancy, the following checks are made:

- The server-to-server addresses within a redundancy group must be unique. A common IP address of a redundancy group may also only be used once within a project.
- Each member of a redundancy group must have the same S7 connections with the same connection names and the same communications partners.
- The hardware configuration of the PC stations in the redundancy groups must be identical. In other words, the modules used for the redundancy function must be identical in the redundant PC stations.
- The parameters of the S7 connections must be identical; these include connection establishment, alarms, optimizations, PDU size, number of parallel jobs, monitoring times.
- The S7 settings of the OPC UA servers must be identical; these include cycle time, access rights, S7 alarms and S7 symbols.

Note

Recommendation: Permanently established S7 connections

We strongly recommend that you configure active, permanently established S7 connections on the OPC UA servers. This achieves shorter failover times if there is a server failure.

Procedure / recommendation

The parameter assignments of the redundant OPC UA servers must match each other. When you save and compile the configuration, a consistency check is run.

By using the STEP 7 options for copying, you reduce the risk of inconsistent information. Either copy the entire PC station or only the OPC server. If you have already configured S7 connections, these are also copied. The connections in the copied PC station must still be assigned the relevant connection partner.

See also

Configuring S7 UA redundancy tab (Page 1985)

Configuring S7 UA redundancy tab

Reference

Setting a redundant OPC UA server in the "Properties > General > S7 UA Redundancy" parameter group

Transparent OPC UA redundancy

In this parameter group, you configure the PC station as a redundant OPC UA server for operation with transparent OPC UA redundancy. Here, use the SIMATIC NET OPC server version 8.1

The redundant PC stations assigned to an OPC UA client belong to a redundancy group.

In this parameter group, follow the steps outlined below:

1. Select the "Enable redundancy" option.
2. Enter the IP address of the redundancy group. Here, you have the following options:
   - You can either enter the IP address directly or accept or modify the proposed IP address. Select this option if you want to create a new redundancy group.
   - Click the "Assign" button
     Select this option to assign the OPC server to an existing redundancy group.

     Note

     After entering or assigning, note the "IP addresses of the other members of the redundancy group" table at the end of the parameter group. You can see which OPC servers are assigned to the same redundancy group.
3. Specify the redundancy port for communication with the OPC UA client; if appropriate, accept the default setting.
4. Set the parameters for the server-to server communication.
5. If necessary check the authentication certificate assigned to the OPC server using the "Display..." button.

Parameters / input areas / buttons

- "Enable redundancy" option
  By selecting the option, you enable the parameter assignment option for the OPC UA server functionality.
- "Assign..." button
  You open the dialog in which the redundancy groups that have already been created can be selected.
- "IP address of redundancy group" parameter
  Virtual IP address via which the OPC UA client reaches the redundant OPC UA server (the redundancy group).
  You specify this virtual IP address in the network load balancing manager (NLB) for both PC stations. The virtual IP address is mapped to the real IP addresses of the OPC UA server.
- "Redundancy port" parameter
  Port via which communication with the OPC UA client is handled.
  Default: 4845
- "Server-to-server port" parameter
  Port via which the communication for synchronization with the OPC UA servers in the redundancy group is handled.
  Default: 4846
- "Server-to-server interface" parameter
  Here, select the interface to the subnet via which the synchronization with the OPC UA servers in the redundancy group is handled.
- "Certificate" input area / "Display..." button
  STEP 7 automatically generates a certificate that it is transferred to the OPC UA client during connection establishment. The OPC UA client uses this for authentication during the subsequent communication.
  The certificate is also used to transfer the key for data encryption.
  The private key remains on the server and is used for encryption. With the corresponding public key it received from the server with the certificate, the client can then encrypt messages of the server.
  You can display the currently valid certificate here and check it.
  Note:
  The certificate display does not recognize whether or not a certificate is already being displayed. Each time you click the "Display..." button, a new certificate display is opened.
- "Certificate" input area / "Update" button
  You instruct STEP 7 to create a new certificate.

IP addresses of the other members of the redundancy group

Display of the other members of the redundancy group. A redundancy group is made up of the PC stations that are assigned the same "IP address of the redundancy group".

Note

Maximum number of members in a redundancy group

In the current version of STEP 7 V12, **2** OPC UA servers are supported within a redundancy group.

See also

Redundant OPC UA server - basics (Page 1983)

Assigning an S7-UA redundancy group

Reference

Setting a redundant OPC UA server in the "Properties > General > S7 UA Redundancy > S7 UA Redundancy Master" parameter group

Assigning an existing S7-UA redundancy group

The table shows the redundancy groups available and selectable in the STEP 7 project along with their IP addresses.

If you have assigned a new IP address to the current OPC server, the following entry appears instead of this assignment:

- "(own redundancy group <IP address>)"

Follow the steps outlined below:

1. Select the required redundancy group.
2. Confirm the dialog with the "OK" button.

The IP address of the selected redundancy group is then adopted in the "S7-UA Redundancy" parameter group.

See also

Redundant OPC UA server - basics (Page 1983)

CPU messages for OPC server

SIMATIC NET OPC server - Recommendations on the configuration of CPU messages

If you configure CPU messages for a SIMATIC NET OPC server of a PC station, their consistent configuration will not be verified by STEP 7. To ensure error-free functioning of the messages, note the following recommendations:

- Only ever configure one S7 connection with alarm capability per CPU for receiving messages.
- Only use one language for message texts in a given project. The use of multi-language message texts is not supported.
- For all messages, avoid inputting user-specific sources (additional text 1). The uniqueness of the messages is automatically ensured if the default sources are used.

PC station CP Ethernet

Pulling and plugging Ethernet PC CPs

Pulling and plugging Ethernet PC CPs

Ethernet modules, for example USB network adapters, must already be plugged and activated when the system starts. If Ethernet modules are deactivated or do not exist when the system starts or they are pulled and plugged again during operation, the "Available devices" functionality does not display all the nodes in STEP 7. This also applies to the change of docking stations when the Ethernet module is not plugged into the notebook but rather into the docking stations.

Activate the deactivated Ethernet modules by using "Control Panel > Network Connection" and reboot the system. Also reboot the system if Ethernet modules were plugged during operation or if the docking station with the Ethernet module was changed during operation.

OPC configuration (Ethernet CP)

Reference{"OPC configuration"}

Setting the OPC configuration for the PC Ethernet CP in the "Properties > General > OPC configuration" parameter group.

The parameter group exists if an OPC server is configured in the PC station.

"Access via the OPC server" option

Enable the module for operation via an OPC server configured on the device.

"Automatic mode of the PROFINET IO controller" option

If you select this option, the operating mode of the PROFINET IO controller is set automatically to OPERATE.

In the OPERATE mode, the PROFINET IO controller has access to the connected PN IO system.

The option can be selected if the option "Access via the OPC server" was selected.

- Option enabled (default setting)
  If you select this option, the PROFINET IO controller is changed to the OPERATE mode as soon as there is access to an IO device. This access can either be over the OPC Item interface or the IO Base user programming interface.
  No prior calls are required to initialize the operating state.
- Option disabled
  Prior to data access, the PROFINET IO controller needs to be set to the OPERATE mode. To do this, use suitable initialization calls for an interface in the user program.
  The interface in the user program can be an OPC item interface or an IO-Base user programming interface.

Ethernet port

Time synchronization

Time synchronization

Here, you decide whether or not the CP processes time frames.

The following two options are available depending on where the time master is located

- Master (time transmitter)
  Time messages can come from the PC station and be forwarded to the subnet (LAN). A PC application that performs the time master function transfers the time information to the CP in a job.
- Slave (time recipient)
  Time messages can come from the subnet (LAN) and be forwarded to the station. A PC application can read the transferred time using a suitable job.

If you are running several CPs in your station, you should consider the flow of time frames in relation to the time-of-day master since you can pass time frames from one network to another network using the function described here.

If there are several CPs that are connected to the same network in a station, only **one** of these CPs may forward the time synchronization messages.

Advanced options

Sending keepalives for connections

Reference

Making special settings for the Ethernet interface in the "Properties > General >Ethernet interface > Advanced options > Interface options" parameter group

Sending keepalives for connections

If the keepalive mechanism is activated, keepalives are sent for connections via TCP/IP. This ensures that connections are terminated if one of the communication partners fails and that connection resources are released again.

Depending on the module type, there are two options for the keepalive setting:

- Enable/disable keepalives with a fixed interval
  You can enable and disable the mechanism for connection monitoring. An interval time of 30 seconds is specified for the module; this is the interval at which the keepalives will be sent to the partner of a communication connection.
  - Keepalive disabled: 0 seconds
  - Keepalive enabled: Interval time permanently set to 30 seconds
- Keepalives with adjustable interval time
  Here, you can set the interval at which keepalives are sent to the partner on a communication connection.
  Range of values:
  - Default setting: 30 seconds
  - Maximum value: 999 999 seconds
  - Keepalive disabled: 0 seconds

Notes / recommendations:

Note that the keepalive mechanism can cause subordinate connections (for example, an ISDN telephone connection) to be maintained even though user data are not actually being transferred. If you want to avoid this, you will have to either

- Disable the keepalive mechanism (interval time = 0)

or

- set the interval so high that the underlying connection is terminated before a keepalive is sent when there is no data traffic.

Note
Modules of other manufacturers

When using modules of other manufacturers, please refer to their information regarding keepalive settings.

Notes on connection release monitoring

Relationship to connection configuration

The possible time monitoring setting in this case relates to the time monitoring that can be set in the connection configuration as follows: The lower value is valid.

Time monitoring can be configured in the connection configuration in "Properties > Details on connection setup".

Web

Web server on the PC CP

Reference

Configuration of access to the Web server on the PC CP in the "Web server" parameter group

"Enable Web server on this module" option

The CP provides you with the functionality of a Web server for access via port 80 using a Web browser. Certain HTML pages with CP information and diagnostic functions are stored in a memory area of the CP for this.

Enable this option in order to be granted access to these HTML pages.

- Option enabled
  Enable this check box so that the Web server of the CP is started after downloading the configuration data. You can then read out all the information available on the CP Web pages using a Web browser.
- Option disabled
  If the check box is disabled, access to information provided by the CP Web pages is restricted. Disabling may be suitable in certain operating situations to keep additional resources free on the CP.
  The following pages are not displayed:
  - Diagnostics buffer
  - Module information
  - Topology

Automatic update

If the check box is selected, the Web pages are updated automatically at the interval set in "Update interval".

The Identification Web page is excluded from the automatic update.

Selecting languages

Select the languages for language-dependent texts to be downloaded to the CP. The number of selectable languages is CP dependent. The texts in the diagnostics buffer, for example, are language dependent.

You should also remember that the languages you select here must be installed in the project. You will find the selection of project languages in the project tree in "Languages & Resources"

If the languages you select were not previously installed, the Web server can only display texts in your default language.

PC station CP PROFIBUS

Operating PROFIBUS CPs in computers with STEP 7 V5.5

Operating PROFIBUS CPs in computers with STEP 7 V5.5

If both STEP 7 V5.5 and STEP 7 Professional are installed on a computer and STEP 7 V5.5 is removed, the setup repair function of STEP 7 Professional must be run before the following communications processors can be used as network adapters for downloading firmware:

- CP 561x, CP 5611 A2, CP 5612, CP 5613 A2, CP 5614 A2
- CP 5621, CP 5622, CP 5623, CP 5624
- CP 5711

Firmware download for S7-300/400 PROFIBUS CPs

Firmware download for S7-300/400 PROFIBUS CPs via CP 5612 or CP 5622

If the required interface parameter assignment "FWL" or "FWL_FAST_LOAD" is not available for the firmware download, follow these steps:

1. Open the "Set PG/PC interface" program. You can find the program in the Control Panel of your Windows operating system.
2. Select the interface parameter assignment depending on the operating system and then copy this:
   - For 32-bit operating systems:
     "CP5612(PROFIBUS)" / "CP5622(PROFIBUS)"
   - For 64-bit operating systems:
     "CP5612.PROFIBUS.1" / "CP5622.PROFIBUS.1"

3. Depending on your requirements, give the copy one of the following names:
   - For 32-bit operating systems:
     CP5612(FWL) / CP5622(FWL)
     CP5612(FWL_FAST_LOAD) / CP5622(FWL_FAST_LOAD)
   - For 64-bit operating systems:
     "CP5612.FWL.1" / "CP5622.FWL.1"
     "CP5612.FWL_FAST_LOAD.1" / "CP5622.FWL_FAST_LOAD.1"
4. Open the properties of the entry and make the following settings in the "PROFIBUS" tab:
   - Select the check box for "PG/PC is the only master on the bus".
   - Assign the value "0" to the address.
   - Set the value 187.5 Kbps (or the value 1.5 Mbps for FWL_FAST_LOAD) as the transmission speed.
   - Change the "Highest device address" to the value 126.
   - Change the profile to "Universal (DP/FMS)".
5. Exit the "Set PG/PC interface" program.

Result:

The configured interface parameter assignment is created and is then available in the firmware loader.

LSAP configuration

Reference

Setting the LSAP configuration for the PC PROFIBUS CP in the "Properties > General > LSAP configuration" parameter group.

Reserving LSAPs

Reserve LSAPs (Local Service Access Point) for other protocols used on the CP.

Example FDL connections: The FDL protocol uses LSAPs that must not be used by other protocols at the same time.

To reserve an LSAP, select the number belonging to the LSAP in the displayed list.

OPC configuration (PROFIBUS CP)

Reference

Setting the OPC configuration for the PC PROFIBUS CP in the "Properties > General > OPC configuration" parameter group.

The parameter group exists if an OPC server is configured in the PC station.

"Access via the OPC server" option

Enable the module for operation via an OPC server configured on the device.

"Set DP master to OPERATE automatically" option

If you select this option, DP mode starts automatically when the OPC server starts up.

Activity monitoring of the CP enabled - "DP watchdog" option

If the OPC server no longer accesses the DP interface (due to an error), the module reacts as follows if the watchdog is activated: Data with the value 0 (substitute values) is sent to all DP slaves assigned to the OPC server.

Selectable parameter:

- Timeout of the DP watchdog
  If the watchdog is activated, the substitute values are transferred if there is no further access to the DP interface within the time set here.
  Default: 6000 ms

"Diagnostics monitoring" option enabled

If diagnostics monitoring is enabled, the OPC server itself monitors the diagnostics flag when inputs and outputs are accessed and evaluates the diagnostics data of DP standard slaves. If errors or problems are detected in the diagnostics data, access to the inputs and outputs of the DP slaves is prevented for the lock time set here; read and write jobs are therefore acknowledged with an error.

Selectable parameter:

- Lock time for Read / Write
  Default: 500 ms

Error wait time for DP C1 jobs

Here, you specify an error wait time for the DPC1 services (acyclic transfer as DP master class 1).

Default: 15000 ms

OPC server functions for DP master class 2

Here, you can enable the OPC server functions for DP master class 2. The settings here relate to the data record functions according to DP master class 2.

A 2-level option is available:

- Enable data record functions
  With the data record functions, device-specific information can be read out or parameters can be written. With this option, you enable the OPC server that supports this function.
- Operate in PROFIdrive mode
  If you select this extra option, the PROFIdrive bus server is enabled. As a result the data record functions can be used (only) in the special PROFIdrive mode.

Selectable parameter:

- Connection establishment timeout
  If the request to establish a connection is not responded to by the communications partner within the time set here, OPC jobs are aborted with an error message.
  Since due to the structure of the network some time may be necessary before a connection can be established successfully, the connection establishment timeout should be longer than the job timeout (refer to the error wait time for asynchronous jobs).
  Default: 15000 ms
- Error wait time for asynchronous jobs
  If a read or write job sent to a partner device is not responded to within the time set here, the pending OPC jobs are aborted with an error message.
  Default: 15000 ms

Configuring the LDB file

Reference

Setting the LDB configuration for the PC PROFIBUS CP in the "Properties > General > LDB configuration" parameter group.

Note

Version of the SIMATIC NET PC software

The LDB file can only be used for CPs with the SIMATIC NET PC software up to the version May 2000.

Generating LDB file (applies only to CP is with firmware < V6.0 SP5 or up to SIMATIC NET PC software May 2000)

The configuration of the DP master system must be exported to an LDB file (loadable database) when saving and compiling. This file contains the DP project information. It is required by the DP system attachments for PG/PC such as SOFTNET or CP 5614 in the PC stations.

Note

LDB file and identifying the input/output addresses of the DP modules

The addresses of the input/output areas of the DP slaves are acquired automatically in ascending order based on the PROFIBUS address of the DP slaves and on the basis of the inserted DP modules and entered in the LDB file.

Address information or address changes to the input/output addresses of the DP slaves in the DP master system configured here therefore have no effect on the address information stored in the LDB file.

Networked PC configurations

Configuration dialog box

Reference

You select the " Online > Configure PC station online" menu command for a selected PC station.

You transfer the selected configuration for a PC station or display its current configuration with the highlighted dialog.

Requirements

The "SIMATIC NET PC Software" DVD must be installed on your target computer and the configuration computer.

If more than one communication module is present on the configuring computer or the target computer, you must first specify the particular module for the configuration operation. The SIMATIC Shell program is available for this purpose.

If you have specified the module for the configuring computer or if only one communication module is present, the local network connection is already displayed and cannot be changed.

Provided you have selected the target computer in your STEP 7 project and there is an explicit assignment to the accessible computers, the "Accessible target computer" and "Target computer" fields are also preassigned accordingly and do not have to be changed.

Note

SIMATIC Shell

The "SIMATIC Shell" program is installed with the SIMATIC NET PC software products. Note the documentation on the SIMATIC NET PC software.

Follow these steps:

1. From the drop down list box, select the network connection to be used to address the target computer.
2. Click the "Update" button to determine the accessible computers again.
3. In the "Accessible computers" list, select the desired target computer or enter the name of the target computer directly in the "Target computer" field. The designation of the target computer is the same as the computer name in Windows.
4. Now select one of the following procedures:
   – Display the current configuration of the selected target computer: Click the "Display..." button for this purpose.
   – Download the configured configuration to the target computer: Click the "Configure..." button for this purpose.
5. Note any messages that appear following configuration.

Assigning the network connection and target computer

If you do not want to use any preassignments or there is no preassignment, start by following these steps:

1. If necessary, disable the "Use configured target computer" option.
2. From the drop down list box, select the network connection to be used to address the target computer.
3. Click the "Update" button to determine the accessible computers again.
4. In the "Reachable computers" list, select the required target computer. As an alternative, enter the name of the target computer directly in the "Target computer" box. The designation of the target computer is the same as the computer name in Windows.

Configuring the target computer

Continue as follows:

1. Choose between the following procedures:
   – Display the current configuration of the selected target computer: Click the "Display..." button for this purpose.
   – Download the configured configuration to the target computer: Click the "Configure..." button for this purpose.
2. Note any messages that appear following configuration.

Tip:

You can also access the functions described for the buttons by right-clicking.

General notes

Note that after the start of the configuration dialog, changes will not be detected on the local computer. The settings or configuration data detected during the startup are valid. This affects the following actions:

- Enabling or disabling the network adapters.
- Changes in the STEP 7 configuration of the PC station.

"Target computer configuration" dialog box

This dialog box shows the current configuration including modules and applications.

Here, you can compare a current configuration with a configured one.

Station

The station name is shown here.

The station name can be the same as the computer name of the target computer if the target station has not yet been configured with a new name after installation of the SIMATIC NET PC software. Upon booting, the computer name is initially transferred as the station name.

See also

Status symbols during configuring (Page 1999)

Configuration dialog box (Page 1996)

"Configure: selected target computer" dialog box

This dialog box shows the configuration, as currently configured, for the selected target computer including modules and applications.

You can initiate the transfer for the initial configuration or configuration change from here.

When the dialog box is opened, a check is made to determine whether the configuration, as configured, is valid for transfer to the target computer and, thus, can be transferred.

Station

The station name is shown here.

The station name can be the same as the computer name of the target computer if the target station has not yet been configured with a new name after booting. Upon booting, the computer name is initially transferred as the station name.

Checking and adopting the configuration, as configured

Here, you can check the configuration, as configured, before carrying out the final data transfer.

The information bar below the configuration list indicates whether or not the configuration can be performed. For this purpose, the existing physical configuration must match the configuration, as configured.

Note the symbols that indicate special circumstances.

After you have checked the displays and initiated the transfer (OK button), you will be prompted again to confirm the transfer.

When the configuration is rejected

The configuration cannot be transferred in the following cases:

- Configured modules are not present in the PC station.
- Configured modules cannot be explicitly assigned (see assignment rules below).
- The configuration, as configured, exceeds the permissible quantity framework for the PC station.
- A connection to the PC station cannot be established.
  This can be due to the fact that a firewall configured in the PC station blocks access to Port 102, which is used by STEP 7. Enable Port 102 in the PC station for TCP.

Review the notes or messages in the "Error" and/or "Status" fields! For example, you receive warnings if the assignment would cause a change in the previous configuration (example: an already configured module is moved to a different index).

If you double-click the message text, a window with further information regarding the message is displayed.

Assignment rules

During the import, a check is made to determine whether the modules can be assigned and, thus, transferred.

- The specified modules must be present.
- If multiple modules (e.g., PROFIBUS CPs) are found, an attempt is made to make an assignment:
  - Based on the parameter assignment string assigned to the module (assigned in STEP 7)
  - Based on the address (PB address, MAC/IP address)
  - Based on compatibility (range of functions)

  Note
  Softnet-CPs

  Note that only 1 Softnet-CP (PROFIBUS or Industrial Ethernet) at a time may be present in the configuration!

See also

Status symbols during configuring (Page 1999)

Configuration dialog box (Page 1996)

Status symbols during configuring

"Status" column

Note the additional information in the "Causes" column.

Error: Component cannot be assigned/transferred or is not installed

Warning: Assignment was adapted or is not unique.

Information bar below the configuration list

A group message containing information about the import operation is output in the information bar:

Configuration cannot be performed

Configuration can be performed, but with warnings: Assignment of individual components was adapted or is not unique.

Configuration cannot be performed

See also

"Configure: selected target computer" dialog box (Page 1998)

PC station settings for the PC CP not dependent on the protocol or network

XDB configuration

Reference

Setting the XDB configuration for the a PC station in the "Properties > General > XDB configuration" parameter group.

XDB configuration

- Compatibility - "S7RTM is installed" option
  When you configure a new PC station, the check box is enabled. In this case, a runtime station manager (S7RTM) is required on the target PC station. The target PC station must be configured with the Station Configuration Editor (SIMATIC NET PC software as of approximately 7/2001).
  If the check box is disabled, configuration files are generated that can be interpreted by "older" PC stations.
- Storage location of the configuration file
  The configuration with the connection descriptions for the SIMATIC PC station is stored in a configuration file (*.XDB) when you save and compile. You then install this file on the relevant PC station.
  With the "Browse" button, you can change the default path for the configuration file.
  The relative path information (.\...) relates to the folder in which the current project is stored (<Install dir>\S7proj\<project>\XDB). If you enter an absolute path (for example C:\...), remember that the access only works as long as the project remains in the current environment.

Setting the interface parameter assignment

Checking the interface parameter assignment

When you take the PC communications card from the hardware catalog and place it in a rack, it is assigned a suitable interface parameter assignment.

The valid parameter assignment can be recognized based on the information on the module type in the module catalog (name of the module and number of configurable modules of the same type) and the existing already configured modules (assignment of the board number).

If several CPs of the same type are configured in the station, the module names have an additional consecutive number added to them – the board number. From the available drop-down list, you can see the permitted entries and, if necessary, select them.

In special cases, for example when you change the module arrangement of identical modules in the PC station, it may be necessary to adapt the interface assignment here; changing the arrangement means the actual hardware and not the "virtual" slot in HW Config.

Note

Interface parameter assignment on the target system (PC system)

Make sure that the same interface parameter assignment is used in the target system (PC station) as you specify here. Otherwise, you will have an interface conflict in the station after loading the configuration file.

Note

Checking on the target system

Use the Setting the PG/PC Interface function to check the settings on the target system (PC station). You will find this in the Windows Control Panel. In the "Access path" tab, you will see which interfaces are configured and whether conflicts exist.

IE/PB Link PN IO

PROFINET device numbers

Reference

Display and setting of the properties for PROFINET IO devices on the DP master system of an IE/PB Link in the "Properties > General > PROFINET device number" parameter group.

Meaning and parameter assignment

To allow you to recognize the assignment better, STEP 7 attempts to assign the same digits for the device number and the PROFIBUS address when the DP slave is placed in the DP master system. As default, the "Use PROFIBUS address as device number" option is therefore enabled as long as no device number has yet been assigned that corresponds to the current PROFIBUS address.

- "Prioritized startup" option
  Enabling the check box means that the PROFINET IO device (here of the DP slave as PROFINET IO device) is given preference by the IO controller so that the startup time is reduced.
  You can select this option for up to 8 DP slaves connected to the DP master system.
  The option can only be selected if the PROFINET IO controller supports prioritized startup and as long as the previously mentioned maximum number has not yet been reached.
- "Device number = PROFIBUS address" option
  If you deselect the option, the input box for the PROFINET device number becomes editable (drop-down list becomes visible).
- PROFINET device number
  Range of values: Depending on the IO controller that you are using, if you exceed the permitted range, you will receive a message relating to the permitted range of values on completion of the dialog.
  If you use the "Use PROFIBUS address as device number" option again after entering a value, the value will be reset (this is only visible after exiting the dialog).

Note

Unique device numbers

Remember that the device numbers assigned in a PROFINET IO system must be unique. The IO devices (DP slaves) connected to the DP master system of the IE/PB Link must also be included.

Modes of the IE/PB Link PN IO

Reference

Configuration of the mode of the gateway and display of the resulting mode for PROFIBUS DP.

- Setting the mode for the IE/PB Link PN IO gateway in the "Properties > General > Operating mode" parameter group.
- Display of the mode for PROFIBUS DP in the "Properties > General > DP/MPI interface" parameter group.

Meaning and parameter assignment

You can use the IE/PB Link PN IO gateway in the following modes:

- Gateway as PROFINET IO proxy
  From the perspective of the PROFINET IO controller on Industrial Ethernet, there is no difference in the access to PROFINET IO devices connected to Industrial Ethernet and to PROFIBUS DP slaves connected to PROFIBUS DP.
  The IE/PB Link PN IO adopts the role of a proxy for the DP slaves connected to PROFIBUS DP (DP mode "DP master" is active).
- Gateway in standard operation
  In this mode, you have the following connection options
  - PG/OP communication
    PG/OP communication is used to download programs and configuration data, to execute test and diagnostics functions and for operator control and monitoring (HMI systems) of a plant.
  - Field device parameter settings (data record routing)
    You can use the IE/PB Link PN IO as a router for data records that are intended for field devices (DP slaves). This means that devices not connected directly to PROFIBUS and therefore without direct access to the field devices (DP slaves) can transfer data records to the field devices via the IE/PB Link PN IO.
    One example of a tool that creates data records of this type for assigning parameters to field devices is SIMATIC PDM (Process Device Manager).
  - Gateway to a DP master system with constant bus cycle time
    The IE/PB Link PN IO serves as a gateway between Ethernet and the field devices of a DP master system. The IE/PB Link PN IO is operated as an active node along with a DP master on PROFIBUS with the parameter setting for a constant bus cycle time.
  - Cross-subnet S7 connections for HMI operation
    The IE/PB Link PN IO forwards communication via S7 connections.
    This service is used, for example, by HMI applications (PC stations).

See also

Configuration using IE/PB link PN IO (Page 1438)

Background on connections

Window Time and Inactivity Time - relation

Relevance

ISO transport connections configuration in the parameter group "Properties > General > Dynamics".

Avoid connection abort with appropriate update

The partner station replies to keepalives with a frame. These are sent to the partner station at the intervals specified by the Window Time.

To avoid unwanted connection aborts, the Inactivity Time should be at least three times longer than the Window Time.

"Immediate response when interrupted connection is detected" option

Meaning

If you set the "Immediate response when interrupted connection is detected" option, this means that if the connection is not available, the application immediately receives a code indicating that a communication job (FETCH or WRITE) cannot be executed. The code is sent regardless of whether or not a current attempt to establish the connections succeeds.

If you do not select this option, the application is informed of the nonavailability of a connection only when the connection times out.

The option described here is relevant and available only when the two modes listed below are used:

- The connection is configured as a permanent connection.
  Connections can be configured as permanently established connections (option in the "Details of connection establishment" tab). This means that jobs can be forwarded to the partner with the shortest possible reaction time without needing to establish the connection first.
  If there is a disruption on the connection, the system automatically attempts to re-establish the connection. In this situation, there may be delays before a message can be forwarded to the application.
- You are using the FETCH or WRITE job types (can be selected in the "Options" tab).

An immediate reaction is necessary only in certain situations with these job types if there is no connection.

Address information - E-mail connection

Parameters for E-mail delivery

In "Properties > Address details", you enter the parameters required for the delivery of the E-mail. You can set the following parameters:

Parameter	Description	Example
E-mail server (SMTP)	Address of the mail server via which the E-mails are sent. The IP address can be entered as an absolute address or symbolic name. The following applies to symbolic names: A valid name has 1 to 64 characters of which at least one character must be a letter. If one or more "." characters are used, the following constraint also applies: A valid name must have 1 to 63 further characters after the last "."; here only the characters [a-z], [A-Z], [0-9] or [-] are permitted and at least one of these characters must be a letter. A symbolic name can only be used if the Internet CP knows the address of the Domain Name System (DNS). A suitable entry is required in the configuration of the Internet CP in HW Config. For more detailed information, refer to the HW Config online help. A maximum of 64 characters can be entered.	absolute: 140.80.0.4 symbolic name: t-online.de
Default sender name	Specifies an address that is always included in the E-mail as the sender address if the sender information (FROM parameter) is empty in the header of the E-mail. This information is normally irrelevant for delivery of the mail. It is simply information for the mail recipient. Since some mail servers do not forward jobs if there is no sender information, **it is advisable to enter information here!** A maximum of 126 characters can be entered. **Note:** Remember that a default sender name must be specified if you want a test mail to be initiated by diagnostics.	Station2.CPU412@xy.factory2.de

Symbols in offline status

Symbols in offline status

The offline status of a connection is represented as follows:

Symbol	Meaning/connection status
	Consistent – connection configuration can be loaded
	Inconsistent – connection configuration cannot be loaded

ISO-on-TCP connection: TSAP

TSAP format

ISO-on-TCP connections have a TSAP length of 1 to 16 bytes. When you are entering values, the current length is displayed automatically. You can enter local TSAPs and partner TSAPs as a hexadecimal value or as an ASCII string. If you opt for ASCII, the characters you enter are also displayed in hexadecimal format. If you enter them in hexadecimal, any printable characters are displayed as ASCII values (8 hexadecimal characters are visible). If you enter non-printing characters, the ASCII display is disabled (ASCII entry is no longer possible), and the non-printing characters appear as dots.

Note

TSAP length

Do not use length 1 or 2 TSAPs, as these TSAPs are already used for internal purposes.

Local TSAPs and partner TSAPs

Partner and local TSAPs may be the same, since the connection is unique due to the different IP addresses. If you have to set up more than one connection between two devices, then the TSAPs must also be different.

Default TSAPs

For configuring the local and partner TSAPs, there is an "ISO-on-TCP-1" default value for the first connection between two partners (can be changed). The default value "ISO-on-TCP-2" is then suggested for a new connection between the same partners. ISO-on-TCP-1 is then suggested for a new connection with a new partner.

Connection setup for FETCH/WRITE

Relevance

ISO transport connections / ISO-on-TCP connections / TCP connections configuration in the parameter group "Properties > General > Options".

How connection establishment affects FETCH/WRITE

Please note the following points for FETCH/WRITE mode:

- For S7 stations:
  If you want to use FETCH/WRITE mode (see the "Options" parameter group), clear the "Active connection establishment" option.
- For PC stations / OPC servers:
  If you want to use FETCH/WRITE mode (see the "Options" parameter group), enable the "Active connection establishment" option.

S7 connection

OPC connection parameters for time monitoring

Connection timeout

If there is no reply to a request for a connection from a communication partner within the time set here, OPC jobs are cancelled with an error message. Values from 2,000 to 100,000 ms can be set.

Since successful establishment of a connection can take some time depending on the network structure, the connection establishment timeout should be longer than the job timeout.

The connection timeout starts with the start of the request for the connection. This time depends on the configuration of the connection characteristics (see above). Two cases are possible:

- **Maintain connection permanently:**In this case, the connection request starts with the start of the OPC server for the S7 protocol. This start is triggered by connecting an OPC client to the OPC server and searching the S7 namespace and/or by adding and actively monitoring an S7 item.
- **Connect on request:**The request for a connection starts with first access to variables of this S7 connection.

If the S7 connection is interrupted in running operation, an attempt is made to reestablish the connection. This resets and restarts the connection timeout.

Job timeout

Network jobs directed to the partner device are monitored by the job timeout. If no response is issued to a network job within the job timeout, the corresponding OPC jobs are cancelled with an error message. Values from 1 to 100,000 ms can be set.

The job timeout starts at the time a network job is sent in the OPC server. In general, there is only an indirect relationship between write/read/monitor jobs at the OPC interface. Several OPC items can be handled for each network job. Additionally, several OPC clients can request the same OPC item for each network job.

Furthermore, the "Optimize write access" and "Optimize read access" parameters play a role because several OPC jobs can be assembled there on the network level.

OPC connection parameters for the connection state

Immediate response when interrupted connection is detected

Can only be enabled if the "Maintain connection permanently" option is selected. The immediate reporting of S7 connections that have not been established with OPC jobs, means that you avoid repeated job timeouts.

Functional description: An interrupted connection is detected and the connection stored. If the connection has not yet been established or it is currently being attempted to re-establish an uninterrupted connection, OPC jobs are replied to immediately with an error message to this effect. The error message is sent without waiting for the job or connection timeout.

Optimizing write access, optimizing read access

Principle

When accessing variables over S7 connections, the OPC Server for SIMATIC NET provides an optimization algorithm to increase the data transmission speed.

Several simultaneous jobs to access individual variables are converted internally to create a single access to an array on the communication partner. This reduces the number of data packets transported over the network. Utilization of single packets is improved and the payload of a packet is increased.

This optimization is used for both read and write access and is enabled as the default setting. For the OPC client, for example an OPC-compliant HMI device, this optimization algorithm is invisible.

Further information

Additional details are contained in the documentation for the OPC server:

- Industrial communication with the PG/PC Volume 1 in contribution ID: 42783968 (http://support.automation.siemens.com/WW/view/en/42783968)
- Industrial communication with the PG/PC Volume 2 in contribution ID: 42783660 (http://support.automation.siemens.com/WW/view/en/42783660)

One-way S7 connection option

Meaning

If the "One-way" check box is selected, only one-way communication blocks can be used on this connection ("PUT" and "GET" instructions) in the user program of an S7 CPU. One-way means that the connection partner functions as a server on this connection and cannot send or receive actively.

Communications instructions used at both ends (with "pairs of blocks") cannot be used on connections configured at one end; for example USEND (Page 7371) and URCV (Page 7373), BSEND (Page 7376) and BRCV (Page 7379) or USTATUS (Page 7408).

Since communication options are generally clearly defined based on the connection end points, the check box is preset and it may not be possible to change it.

Example 1:

The check box is grayed out (cannot be changed) and enabled in the following configuration:

- The local connection end point is an S7-400 CPU or an S7-300 CPU with loadable S7 communication (via CP) and the partner connection end point is an S7-300 CPU without loadable S7 communication.

Example 2:

The check box is grayed out (cannot be changed) and not enabled in the following configuration:

- The local connection end point and the partner connection end point are S7-400 CPUs. In this case, STEP 7 automatically creates a two-way connection. An ID (local ID and partner ID) is entered for both the local end point and the end point of the connection partner.

See also

GET: Read data from a remote CPU (Page 5038)

PUT: Write data to a remote CPU (Page 5040)

GET: Read data from a remote CPU (Page 7366)

PUT: Write data to a remote CPU (Page 7368)

OPC server - alarms

Block-related alarms and diagnostic alarms

The options act as filters so that you can set the specific type of alarm that can arrive via the S7 connection.

The communication partner generates block-related alarms with the Alarm_8, Alarm_S or Alarm_SQ blocks.

Block-related alarms can be signaled

- As conditional events
- As simple events (simple events for compatibility with older applications that do not generally expect conditional events).

If you select "Block-related alarms", the OPC server then registers for alarm messages on this connection. The received alarms are transferred via the Alarms&Events OPC interface.

Diagnostic alarms are derived from the system status list (SSL) of the S7 device. Diagnostic events that have not yet been reported to the OPC client are reported if it has registered for diagnostic alarms. Diagnostic events have no status and do not therefore need to be acknowledged.

Functions to set the priority for single alarm messages and diagnostics alarms can be reached via the area navigation.

Further information

Additional details are contained in the documentation for the OPC server:

- Industrial communication with the PG/PC Volume 1 in contribution ID: 42783968 (http://support.automation.siemens.com/WW/view/en/42783968)
- Industrial communication with the PG/PC Volume 2 in contribution ID: 42783660 (http://support.automation.siemens.com/WW/view/en/42783660)

AS-i

I/O configuration for AS-i slaves

I/O configuration

The I/O configuration is assigned as follows on the basis of the I/O code standard:

Table 8-97 AS-i standard slave

I/O code	I/O configuration			
	D0	D1	D2	D3
0	I	I	I	I
1	I	I	I	O
2	I	I	I	B
3	I	I	O	O
4	I	I	B	B
5	I	O	O	O
6	B	B	B	B
7	B	B	B	B
8	O	O	O	O
9	O	O	O	I
A	O	O	O	B
B	O	O	I	I

I/O code	I/O configuration			
	D0	D1	D2	D3
C	O	O	B	B
D	O	I	I	I
E	O	B	B	B
F	F	F	F	F

Key: I=Input; O=Output; B=Bidirectional (input/output); F=TriState

Table 8-98 AS-i A/B slave

I/O code	I/O configuration			
	D0	D1	D2	D3
0	I	I	I	I
1	I	I	I	-
2	I	I	I	I
3	I	I	O	-
4	I	I	B	I
5	I	O	O	-
6	B	B	B	I
7	B	B	B	I
8	O	O	O	-
9	O	O	O	I
A	O	O	O	I
B	O	O	I	I
C	O	O	B	I
D	O	I	I	I
E	O	B	B	I
F	F	F	F	F

Key: I=Input; O=Output; B=Bidirectional (input/output); F=TriState

ASM 475

What you should know about the ASM 475

Introduction

Up to eight ASM 475 interface modules can be plugged in and operated in a rack of the SIMATIC S7-300. If you have a configuration with several racks (max. four racks), the ASM 475 can be plugged in and operated in each rack. This means that with the maximum configuration of a SIMATIC S7-300, 32 ASM 475 modules can be operated.

A maximum of two SLGs (write/read devices) can be connected to the ASM 475. The processing of the connected SLGs is parallel. FC45 allows simple programming using the SIMATIC S7 tools.

FC45 can be used both in the S7-300 and S7-400.

With S7-400, the ASM 475 is connected via an ET 200M.

Note that the

- IM 153-1 must have at least order number 6ES7 153-1AA03-0XB0 or 6ES7 153-1AA83-0XB0 and
- IM 153-2 at least order number 6ES7 153-2AA02-0XB0 or 6ES7 153-2AB01-0XB0.

Access to the MDS data is with normal addressing.

Physical addressing of the MDS (mobile data storage)

The physical addressing of an MDS is also known as **normal addressing**. Users set up the structure of the MDS themselves. They know the physical MDS address to which data is written. The addressing of the MDS storage generally begins at 0000 hex and ends at an end address corresponding to the size of the MDS storage.

File handler addressing of the MDS

For file handler addressing, the user specifies a file name for access to the data. The file name consists of eight ASCII characters. The file handler manages the user data independently on the MDS. The user does not need to configure data structures on the MDS.

Prior to use, the MDS needs to be formatted.

8.1.5.5 Identification systems

RFID systems

Communications module RF120C

Reader parameter group

The "Reader" parameter group contains the following:

- Diagnostics: Setting to decide whether or not hardware diagnostics messages are output.
- Reader System: Selection of the connected RFID system. Depending on the selection you make, the "Reader System" parameter group is adapted.

"Diagnostics" parameter

Parameter assignment options:

- None
  Apart from standard diagnostics messages, no other diagnostics messages are generated.
- Hard Errors
  Extended diagnostics messages are generated if the following events occur.
  - Hardware error (memory test)
  - Firmware error (checksum)
  - Connection to reader lost
  - Short-circuit fault/interruption if supported by the hardware

Further information

For further information on diagnostics, refer the documentation for the RF120C communications module with application blocks for S7-1200 and S7-1500.

Reader System parameter group

The parameters for the selected identification system are set in the "Reader system" parameter group.

The table shows the parameters that exist for all identification systems (RFID and code reader systems).

Table 8-99 Standard parameters for all identification systems

Parameter	Parameter value	Default value	Description
Baud rate reader	19.2 kBd 57.6 kBd 115.2 kBd	115.2 kBd	After changing the baud rate, the reader must be turned off and on again.
Presence Check	On Off (RF field off) Off (RF field on)	On	On = presence is reported as soon as there is a transponder in the antenna field of the reader Off (RF field on) = the presence check in the FB is suppressed. The antenna on the reader is nevertheless turned on as long as it has not been turned off by a command. Off (RF field off) = the antenna is turned on only when a command is sent and it then turns itself off again (RF300 only)
Reset ERR-LED	On Off	Off	On = the flashing of the error LED on the RF120C is reset by each FB reset. Off = the error LED always indicates the last error. The display can only be reset by turning off the RF120C.

The following parameters are system specific according to the selection you made in the "Reader" parameter group.

RF200

Table 8-100 RF200 general

Parameter	Parameter value	Default value	Description
-	-	-	Only the standard parameters are available.

Table 8-101 RF290R

Parameter	Parameter value	Default value	Description
RF power	0.50 - 5.00 W	1.00 W	Setting for the output power of the reader.

RF300

Table 8-102 RF300 general

Parameter	Parameter value	Default value	Description
Transponder type	RF300 ISO 15693	RF300	Selection of the transponders used.

Table 8-103 RF380R

Parameter	Parameter value	Default value	Description
RF power	0.50 - 2.00 W	1.25 W	Setting for the output power of the reader.
Transponder type	RF300 ISO 15693	RF300	Selection of the transponders used.

RF600

Table 8-104 Reader System: RF600

Parameter	Parameter value	Default value	Description
Max. no. of transponders	1 - 80	1	Number of transponders expected in the antenna field. Permitted values: • 1 - 40 for RF620R • 1 - 80 for RF630R with 2 antennas • 1 - 40 for RF630R with 1 antenna The value stored in "multitag" defines the expected number of transponders to be read (EPC-ID) in the inventory. The value does not restrict the number of transponders to be processed in the antenna field. To ensure an efficient inventory of transponders in the antenna field, make sure that the values specified here do not deviate by more than approximately 10% from the maximum number of transponders to be expected in the antenna field.
Reader type	RF620R ETSI RF620R FCC RF620R CMIIT RF630R ETSI RF630R FCC RF630R CMIIT	RF620R ETSI	Selection of the reader used. By selecting a reader, you open the "Reader type" parameter group. The parameters are described in the following table.

Reader type (RF600 only)

In the "Reader type" parameter group further specific parameters are set for the reader type selected in the "Reader system" parameter group.

Table 8-105 RF600 menu: Reader type

Parameter	Parameter value	Default value	Description
Wireless profile	-	-	Selection of the relevant wireless profile for ETSI, FCC or CMIIT.
Multitag mode	UID = EPC-ID (8 bytes) UID = Handle ID (4 bytes)	UID = EPC-ID (8 bytes)	• UID = EPC-ID (8 bytes) 8 byte UID of the bytes 5-12 of the 12 byte long EPC-ID • UID = handle ID (4 bytes) 4 byte UID as handle ID for access to transponders with an EPC-ID of any length
Intelligent Single Tag Mode (ISTM)	On Off	Off	Enable/disable the "Intelligent Single Tag Mode ISTM" algorithm [1)]

Parameter	Parameter value	Default value	Description
Black List	On Off	Off	Enable/disable the "Black list" [1]
Radiated power internal antenna (RF620R)	0 - B	4	Setting the radiated power for the internal antenna [1] [2]
Internal antenna (RF620R)	-	-	Enable/disable the internal antenna. With the RF620R, either only the internal or only the external antenna can be set.
Transmit power (external antenna RF620R)	0 - 9	4	Setting the transmit power for the external antenna [1] [2]
External antenna (SetAnt block required) (RF620R)	-	-	Enable/disable the external antenna. With the RF620R, either only the internal or only the external antenna can be set.
Transmit power ANT 1 (RF630R)	0 - 9	4	Setting the transmit power for antenna 1[1] [2]
Transmit power ANT 2 (RF630R)	0 - 9	4	Setting the transmit power for the antenna 2[1] [2]
Communications speed	Reliable detection Fast detection	Reliable detection	[1]
Tag hold	On Off	Off	Enable/disable "Tag hold" [1]
Scanning mode	On Off	Off	Enable/disable the "Scanning mode" [1]
Channel assignment (only with wireless profile ETSI)	-	-	Selection of the wireless channels to be used [1]

[1] You will find further information in the "Configuration manual RF620R/RF630R". *

[2] The values for the transmit/radiated power of the antennas can be found in the following table.

* You will find the configuration manual RF620R/RF630R here (http://support.automation.siemens.com/WW/view/en/10805817).

Table 8-106 Transmit / radiated power of the antennas

Hex value	RF630R transmit power	RF620R radiated power (internal antenna)			RF620R transmit power
	dBm / (mW)	ETSI dBm / (mW) ERP	FCC dBm / (mW) EIRP	CMIIT dBm / (mW) ERP	dBm / (mW)
0	18 / (63)	18 / (63)	20 / (100)	18 / (65)	18 / (63)
1	19 / (79)	19 / (79)	21 / (126)	19 / (79)	19 / (79)
...	...	...	...	...	...
4	22 / (158)	22 / (158)	24 / (251)	22 / (158)	22 / (158)
...	...	...	...	...	...
9	27 / (501)	27 / (501)	29 / (794)	27 / (501)	27 / (501)
A	27 / (501)	28 / (631)	30 / (1000)	28 / (631)	27 / (501)
B (...F)	27 / (501)	29 / (794)	31 / (1259)	29 / (794)	27 / (501)

SLG D10S

Table 8-107 SLG D10S

Parameter	Parameter value	Default value	Description
RF power	0.50 - 10.00 W	1.00 W	Setting for the output power of the reader.
Transponder type	ISO I-Code (e.g. MDS D139)	ISO	Selection of the transponders used.

SLG D11S/D12S

Table 8-108 SLG D11S/D12S

Parameter	Parameter value	Default value	Description
Transponder type	ISO I-Code (e.g. MDS D139)	ISO	Selection of the transponders used.

MOBY U

Table 8-109 MOBY U

Parameter	Parameter value	Default value	Description
Standby time	0 - 1400 ms	0 ms	Standby time (scanning_time) for the transponder. If the transponder receives a further command before the standby time has elapsed, this command can be executed immediately. If the transponder receives a command after standby time has elapsed, command execution is delayed by the "sleep_time" of the transponder.
Range limitation	0.2 m 0.5 m 1.0 m 1.5 m 2.0 m 2.5 m 3.0 m 3.5 m	1.5 m	-
Max. no. of transponders	1 - 12	1	Maximum number of transponders that can be processed at the same time in the antenna field.

Parameter	Parameter value	Default value	Description
BERO mode	Without BEROs 1 or 2 BEROs 1st BERO on, 2nd BERO off Synchronization by cable connection	Without BEROs	• Without BEROs No reader synchronization • 1 or 2 BEROs The BEROs are logically ORed. The antenna field is turned on during the actuation of a BERO. • 1st BERO on, 2nd BERO off 1st BERO turns on the antenna field and the 2nd BERO turns the antenna field off. If there are two BEROs present and "BERO time in s" is set, the antenna field is turned off automatically if the 2nd BERO does not switch within this BERO time. If no "BERO time in s" is set, the antenna field remains turned on until the 2nd BERO is activated. • Synchronization by cable connection Enable reader synchronization via cable connection (see manual on configuration, installation and service for MOBY U).
BERO time in s	0 - 255 s	0 s	Can only be set if the BERO mode is set to "1st BERO on, 2nd BERO off". • 0 The time monitoring is disabled. To turn off the field, the 2nd BERO is required. • 1 ... 255 s Operating time for the reader field

General Reader

Table 8-110 General Reader

Parameter	Parameter value	Default value	Description
Input box for byte sequence (hexadecimal)	00000000	00000000	Expert mode With this function, you can specify the reset parameters directly in hexadecimal notation. This setting may only be selected if you have previously received the hexadecimal string for the setting from a member of the SIEMENS staff.

Parameters via FB / code readers

Table 8-111 Parameters via FB / code readers

Parameter	Parameter value	Default value	Description
MOBY mode	RF200, RF300, RF600, MOBY D/U, MV	RF200, RF300, RF600, MOBY D/U, MV	Only the mode "RF200, RF300, RF600, MOBY D/U, MV" is supported. For further information on resetting a function block, refer to the documentation for the RF120C communications module with application blocks for S7-1200 and S7-1500. *

* You will find information on the documentation here (http://support.automation.siemens.com/WW/view/en/10805817).

8.1.5.6 Distributed I/O

Distributed I/O systems

SIMATIC ET 200 - The right solution for all applications

SIMATIC ET 200 provides the most varied range of distributed I/O systems.

- Solutions for use in the control cabinet
- Solutions without control cabinet directly at the machine

Additionally, there are also components that can be used in explosive areas. SIMATIC ET 200 systems for construction without a control cabinet are contained in robust, glass-fibre reinforced plastic casing and are therefore shock-resistant, resistant to dirt and watertight.

Their modular design allows the ET 200 systems to be easily scaled and expanded in small steps. Fully-integrated auxiliary modules lower costs and also provide a wide range of possible applications. There are several combination possibilities available:

- Digital and analog I/OS
- Intelligent modules with CPU functions,
- Safety technology,
- Pneumatics,
- Frequency converters
- Various technology modules.

Communication via PROFIBUS and PROFINET, uniform engineering, clear diagnostic possibilities as well as optimal connection to SIMATIC controller and HMI devices vouch for the unique consistency provided by Totally Integrated Automation.

The following table provides an overview of I/O devices for use in the control cabinet:

I/O device	Properties
ET 200S	• Highly modular design with multiple conductor connections • Multifunctional due to a wide range of modules • Use in explosive areas (Zone 2)
ET 200S COMPACT	• Highly modular design with multiple conductor connections • Multifunctional due to a wide range of modules • Use in explosive areas (Zone 2) • Integrated DE/DA
ET 200L	• Cost-effective digital block I/OS • Digital electronic blocks up to 32 channels
ET 200M	• Modular design with standard modules from SIMATIC-S7-300 • Failsafe I/O modules • Use in explosive areas up to Zone 2, sensors and actuators up to Zone 1 • High level of plant availability, for example by plugging and unplugging when in operation
ET 200iSP	• Modular design, also possible with redundancy • Robust and intrinsically safe design • Use in explosive areas up to Zone 1/21; sensors and actuators even up to Zone 0/20 • High level of plant availability, for example by plugging and unplugging when in operation

The following table provides an overview of I/O devices for use without a control cabinet:

I/O device	Properties
ET 200pro	• Modular design with compact housing • Easy assembly • Multifunctional due to a wide range of modules • High level of availability due to plugging and unplugging in operation and permanent wiring • Comprehensive diagnostics
ET 200eco PN	• Cost-efficient, space-saving block I/OS • Digital modules up to 16 channels (also configurable) • Analog modules, IO-link master and load voltage distributor • PROFINET connection with 2-port switch in each module • Can be flexibly distributed via PROFINET in line or star shape directly within the plant

I/O device	Properties
ET 200eco	• Cost-effective digital block I/OS • Flexible connection possibilities • Failsafe modules • High level of plant availability
ET 200R	• Specially for use on robots • Assembled directly on the chassis • Resistant to weld spatter due to robust metal housing

See also

Documentation on ET 200L (http://support.automation.siemens.com/WW/view/de/1142908/0/en)

Documentation on ET 200S (http://support.automation.siemens.com/WW/view/en/1144348)

Documentation on ET 200M (http://support.automation.siemens.com/WW/view/de/1142798/0/en)

Documentation on ET 200pro (http://support.automation.siemens.com/WW/view/de/21210852/0/en)

Documentation on ET 200iSP (http://support.automation.siemens.com/WW/view/de/28930789/0/en)

Documentation on ET 200R (http://support.automation.siemens.com/WW/view/de/11966255/0/en)

Documentation on ET 200eco PN (http://support.automation.siemens.com/WW/view/de/29999018/0/en)

Documentation on ET 200eco (http://support.automation.siemens.com/WW/view/de/12403834/0/en)

ET 200iSP

ET 200iSP Distributed I/O Station

Definition

The ET 200iSP distributed I/O station is a highly modular and intrinsically safe DP slave with degree of protection IP 30.

Area of application

The ET 200iSP distributed I/O station can be operated in potentially explosive atmospheres characterized by gas and dust:

Approval	ET 200iSP Station*	Inputs and outputs
ATEX	Zone 1, Zone 21	up to Zone 0, Zone 20 **
IECEx	Zone 2, Zone 22	up to Zone 0, Zone 20 **
* In combination with an appropriate enclosure ** for electronic module 2 DO Relay UC60V/2A: up to Zone 1, Zone 21		

The ET 200iSP distributed I/O station can, of course, also be used in the safety area.

You can insert almost any combination of ET 200iSP I/O modules directly next to the interface module that transfers the data to the DP master. This means you can adapt the configuration to suit your on-site requirements.

Every ET 200iSP consists of a power supply module, an interface module and a maximum of 32 electronic modules (for example digital electronics modules). Remember not to exceed the maximum current consumption.

Terminal modules and electronic modules

In principle, the ET 200iSP distributed I/O station consists of various passive terminal modules onto which you plug the power supply and the electronic modules.

The ET 200iSP is connected to PROFIBUS RS 485-IS by means of a connector on terminal module TM-IM/EM. Every ET 200iSP is a DP slave on the PROFIBUS RS 485-IS.

DP master

All ET 200iSP modules support communication with DP masters that are compliant with *IEC 61784-1:2002 Ed1 CP 3/1* and operate with "DP" transmission protocol (DP stands for distributed peripherals or distributed I/O).

See also

Documentation on ET 200iSP (http://support.automation.siemens.com/WW/view/de/28930789/0/en)

Assigning the channel and IEEE tag

Properties

Analog electronic modules 4 AI I 2WIRE/HART, 4 AI I 4WIRE/ HART and 4 AO I HART support up to four IEEE tags.

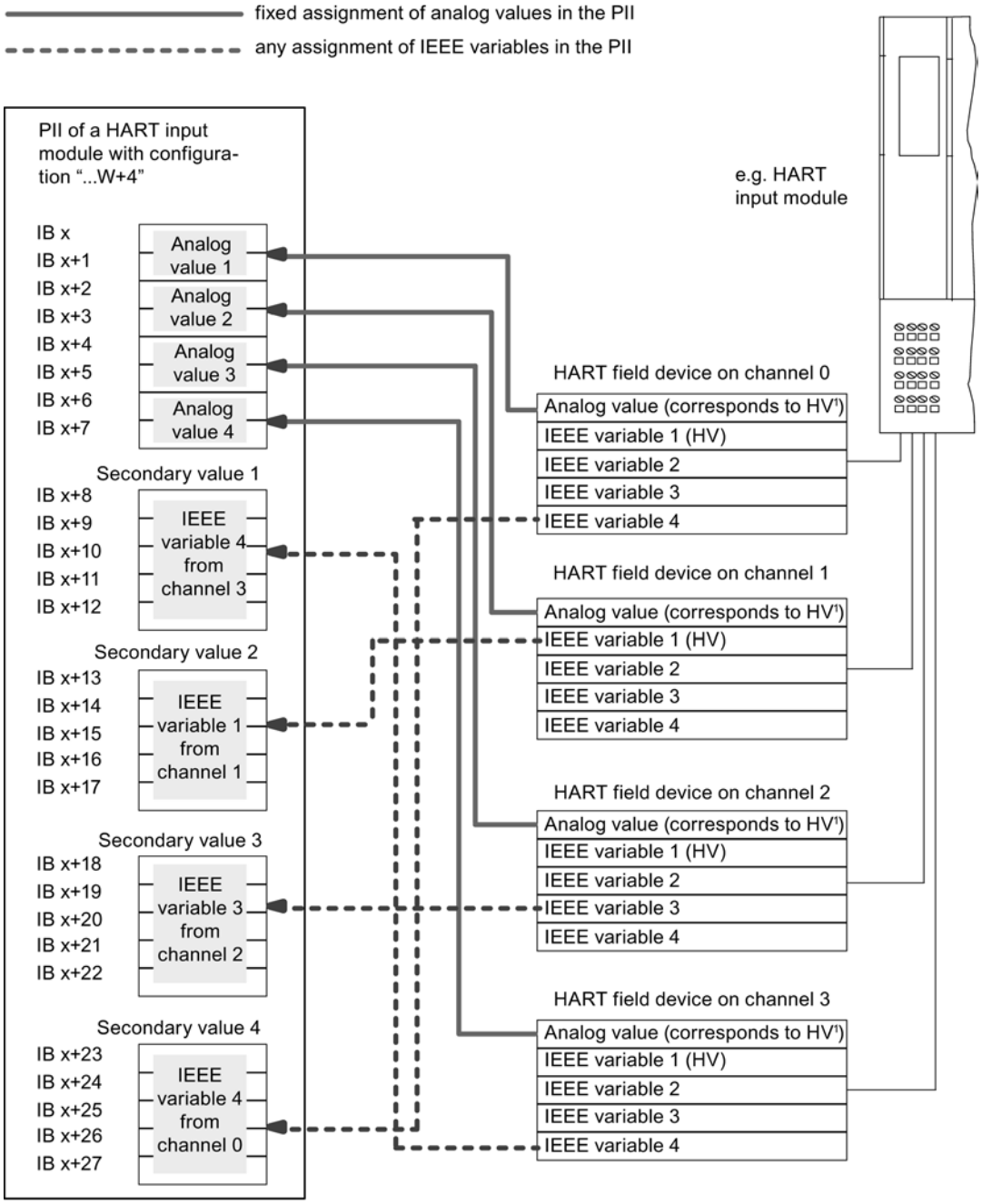
The process input image (PII) provides up to 20 bytes per module for the IEEE tags. Thus, four blocks of 5 bytes each are available for the four IEEE tags within the PII.

Requirements

The HART field device must support the assigned number of IEEE tags.

Assigning IEEE tags

You assign the IEEE tags of the field devices to any one of the four blocks in the PII.



[1] HV = main variable

See also

Documentation on ET 200iSP (http://support.automation.siemens.com/WW/view/de/28930789/0/en)

Assigning parameters to reference junctions for thermocouples

Compensation of the reference junction temperature

There are various ways of obtaining the reference junction temperature in order to get an absolute temperature value from the temperature difference between the reference junction and the measuring point.

Table 8-112 Compensation of the reference junction temperature

Option	Explanation	Reference junction parameters
No compensation	You record not only the temperature of the measurement point. The temperature of the reference junction (transition from Cu line to thermocouple line) also affects the thermo-electromotive force. The measured value then includes an error.	None
Use of a Pt100 Climatic Range resistance thermometer to record the reference junction temperature (best method)	You can record the reference junction temperature using a resistance thermometer (Pt100 Climatic Range). If parameterized accordingly, this temperature value is distributed to the 4 AI TC modules in the ET 200iSP where it is offset against the temperature value obtained at the measuring location. Number of reference junctions: 2	The parameter assignment of the IM 152 and the 4 AI TC must be coordinated: • 4 AI RTD assigned parameters for Pt100 climatic range in correct slot; • 4 AI TC: Reference junction : "yes"; select reference junction number "1" or "2" • IM 152-1:Assignment of the reference junction to a slot with 4 AI RTD; channel selection;
Internal compensation 4 AI TC	The TC sensor module (temperature sensor) is mounted onto the terminals of terminal module EM 4 AI TC. The temperature sensor reports the temperature of the terminals to the 4 AI TC. This value is then calculated together with the measured value from the channel of the electronic module.	• 4 AI TC: Reference junction number "internal"

Compensation by means of a resistance thermometer at the 4 AI RTD

If thermocouples that are connected to the inputs of the 4 AI RTD have the same reference junction, compensate by means of a 4 AI RTD.

For both channels of the 4 AI TC module, you can select "1", "2" or "internal" as the reference junction number. If you select "1" or "2", the same reference junction (RTD channel) is always used for all four channels.

Setting parameters for the reference junction

You set the reference junctions for the 4 AI TC electronic modules by means of the following parameters:

Table 8-113 Reference junction parameters

Parameter	Module	Range of values	Explanation
Slot reference junction 1 to slot 2	IM 152	none, 4 to 35	With this parameter, you can assign up to 2 slots (none, 4 to 35), on which the channels for reference temperature measurement (calculating the compensation value) are located.
Input reference junction 1 to 4 input reference junction	IM 152	RTD on channel 0 RTD on channel 1 RTD on channel 2 RTD on channel 3	This parameter allows you to set the channel (0/1/2/3) for measuring the reference temperature (calculation of the compensation value) for the assigned slot.
Reference junction E0 to reference junction E3	4 AI TC	None yes	This parameter allows you to enable the use of the reference junction.
Reference junction number	4 AI TC	1 2 Internal	This parameter allows you to assign the reference junction (1, 2) that contains the reference temperature (compensation value).

See also

Documentation on ET 200iSP (http://support.automation.siemens.com/WW/view/de/28930789/0/en)

Fundamentals of Time Stamping

Properties

Time stamping is possible with the IM 152 in customer applications using FB 62 (FB TIMESTMP).

Principle of operation

A modified input signal is assigned a time stamp and stored in a buffer (data record). If time stamped signals exists or a data record is full, a hardware interrupt is generated to the DP master. The buffer is evaluated with "Read data record". Special messages are generated for events that influence the time stamping (communication with the DP master interrupted, frame failure of time master, ...).

Parameter Assignment

With the parameter assignment you define which IM 152 user data will be monitored. For the time stamping these are digital inputs that are monitoring for signal changes.

Parameter	Setting	Description
Time stamping	• disabled • enabled	Activate the time staming for the channels of the electronics module 8 DI NAMUR.
Edge evaluation incoming event	• rising edge • falling edge	Determine the type of signal change that will be time-stamped.

Counting

Count properties

Counting functions

The 8 DI NAMUR electronics module has configurable counting functions:

- 2 x 16-bit up counters (standard counting function) or
- 2 x 16-bit down counters (standard counting function) or
- 1 x 32-bit down counter (cascading counter function)
- Setting a setpoint with the PIQ
- GATE function
- You can configure the control signals of the counters:
  - Configuration channel 0..1: "Counter", channel 2..7: "DI": Two counters are configured. The control signals of the counters are stored in the PIQ (process image output).
  - Configuration channel 0..1: "Counter", channel 2..7: "Control": Two counters are configured. The control signals of the counters are stored in the PIQ (process image output). They are also controlled by the digital inputs of the 8 DI NAMUR.

See also

Principle of operation (Page 2026)

Configuring counters (Page 2029)

Assigning parameters to counters (Page 2032)

Principle of operation

16-bit up counters (standard counting function)

The counting range is 0 to 65,535.

With each count pulse at the digital input, the count is incremented by 1. Once the count limit is reached, the counter is reset to 0 and it counts up again from this value.

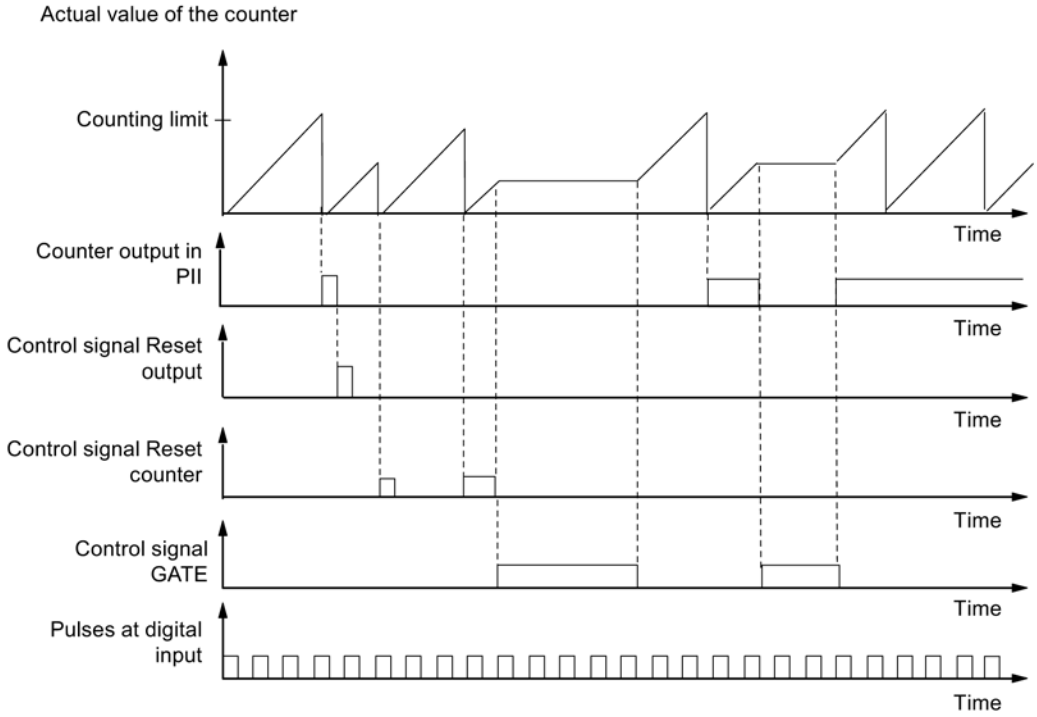
If there is counter overflow, the corresponding output is set in the PII.

A positive edge of the *Reset output* control signal resets the output in the PII. This does not affect the current count value.

In 16-bit up counting operations, the system does not set any outputs in the PIQ. These are always reset.

The positive edge of the *Reset counter* control signal sets the counter to 0 and resets the set counter output.

The *GATE* control signal pauses the counting on a positive edge. Count pulses are processed at the digital input again, but only at the negative edge. The *Reset counter* control signal is also effective when *GATE* is active.



16-bit down counters (periodic counting function)

The maximum counting range is always 65,535 to 0.

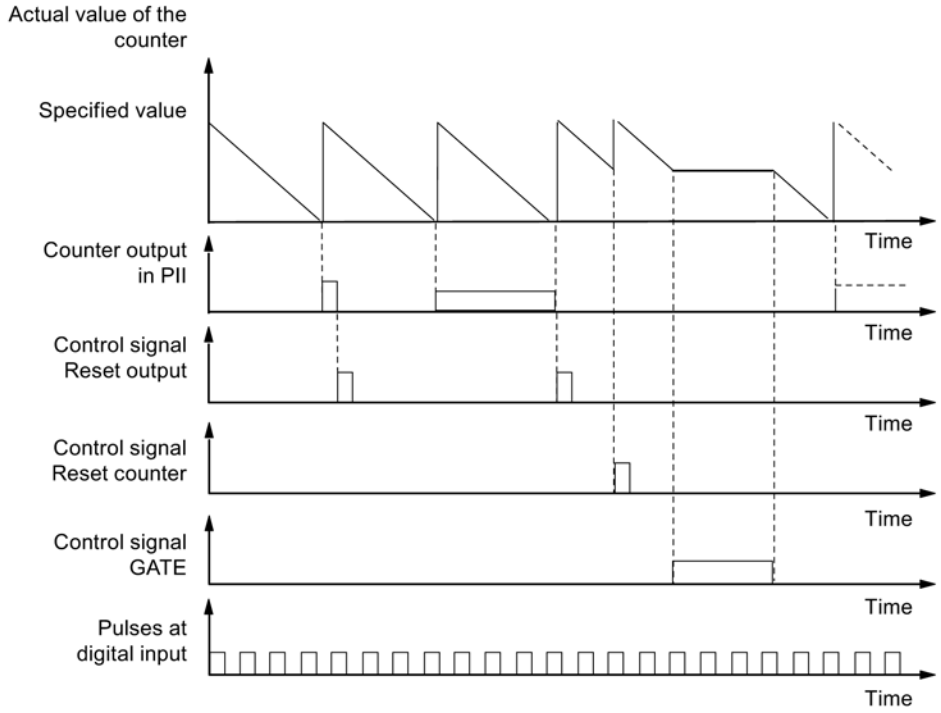
When the counter is started, the actual value is set to the selected setpoint. Each counted pulse reduced the actual value by 1. Once the actual value reaches 0, the corresponding output in the PII is turned on and the actual value is set to the selected setpoint. The counter then counts down from this value.

The positive edge of the *Reset counter* control signal resets the selected setpoint and the corresponding output in the PII.

A positive edge of the *Reset output* control signal resets the output in the PII. This does not affect the current count value.

The *GATE* control signal pauses the counting on a positive edge. At the same time, the assigned output in the PII is reset. Count pulses are processed at the digital input again, but only at the negative edge. The *Reset output* and *Reset counter* control signals are also effective when *GATE* is active.

The setpoint of the counter is set and changed using the PIQ. The setpoint is adopted on a positive edge of the *Reset counter* control signal or when the counter has reached zero.



32-bit down counter (cascading counter function)

The maximum counting range is always 4294967295 to 0.

The principle of operation is identical to that of the 16-bit down counter. Channel 1 has no function.

See also

Count properties (Page 2026)

Configuring counters

Procedure

1. Using the mouse, pull module 8 DI Namur from the hardware catalog into distributed I/O station ET 200iSP.
2. Select the required configuration (channel 0..1: "Counter", channel 2..7: "DI" or "Control"). In the module properties (inspector window), you can find this setting under "Parameters > Inputs > Configuration".

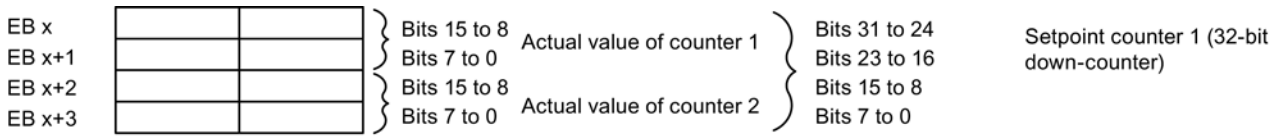
Configuration channel 0..1: "Counter", channel 2..7: "DI"

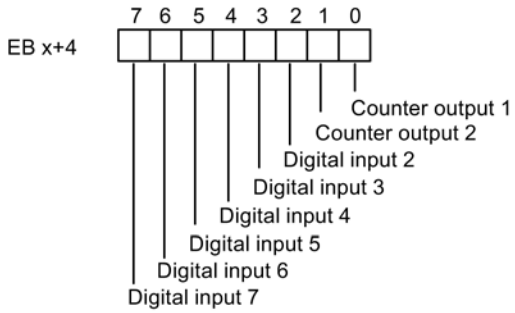
- Assignment of the digital inputs on the electronic module 8 DI NAMUR

Table 8-114 Assignment of digital inputs for channel 0..1: "Counter", channel 2..7: "DI":

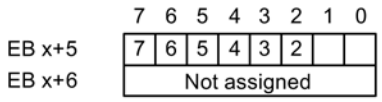
Digital input	Terminal	Assignment
Channel 0	1, 2	Counter 1
Channel 1	5, 6	Counter 2 (does not apply to 32-bit down counters)
Channel 2	9, 10	Digital input 2
Channel 3	13, 14	Digital input 3
Channel 4	3, 4	Digital input 4
Channel 5	7, 8	Digital input 5
Channel 6	11, 12	Digital input 6
Channel 7	15, 16	Digital input 7

- Assignment of the process image input (PII)



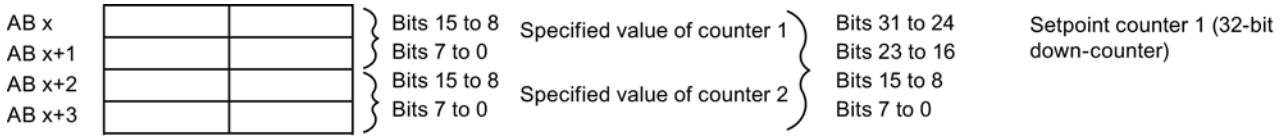


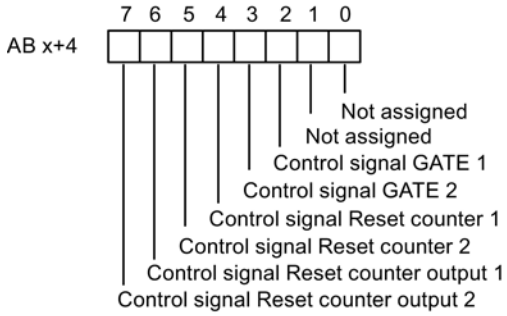
S7 format



Value status for channels 2 to 7:
1_B: Input signal is valid
0_B: Input signal is invalid

- Assignment of the process image output (PIQ)





Configuration channel 0..1: "Counter", channel 2..7: "CONTROL"

With this configuration, you can also control the counters over the digital inputs.

- Assignment of the digital inputs on electronic module 8 DI NAMUR
  For further information on input assignments, refer to the technical data for electronic module 8 DI NAMUR.

Table 8-115 Assignment of the digital inputs for 2 Count/ 6 Control

Digital input	Terminal	Assignment
Channel 0	1, 2	Counter 1
Channel 1	5, 6	Counter 2 (does not apply to 32-bit down counters)
Channel 2	9, 10	control signal *GATE 1*
Channel 3	13, 14	control signal *GATE 2*
Channel 4	3, 4	control signal *Reset counter 1*
Channel 5	7, 8	control signal *Reset counter 2*
Channel 6	11, 12	control signal *Reset counter output 1*
Channel 7	15, 16	control signal *Reset counter output 2*

- Assignment of the process image input (PII)
  Assignment is identical to configuration 0..1: "Counter", channel 2..7: "DI".
- Assignment of the process image output (PIQ)
  Assignment is identical to configuration 0..1: "Counter", channel 2..7: "DI".

See also

Count properties (Page 2026)

Assigning parameters to counters

Parameters for the counting function

Only those parameters that are relevant for the counters are explained below. These belong to the parameters of electronic module 8 DI NAMUR and depend on the selected configuration:

Table 8-116 Parameters for the counters

Parameter	Setting	Description
Sensor type counter inputs	• Channel disabled • NAMUR sensor • Single contact, no load resistance	Select the sensor for the respective counter of channels 0 or 1.
Mode for counter 1	• Standard counting function • Periodic counting function • Cascaded counting function	Select the mode for counter 1.
Mode for counter 2	• Standard counting function • Periodic counting function • Cascaded counting function	Select the mode for counter 2. This parameter is not relevant if you have set the "Mode for counter 1" parameter to "Cascaded counter function".

See also

Count properties (Page 2026)

Frequency measurement

Frequency measurement properties

Properties

The electronic module 8 DI NAMUR allows the frequencies to be measured on channel 0 and 1:

- 2 frequency meters from 1 Hz to 5 kHz
- Configurable metering window (GATE)
- The signals of the frequency meter are read in by means of the digital inputs of the electronic module.

See also

Principle of operation (Page 2033)

Configuring frequency meters (Page 2033)

Assigning parameters for the frequency meters (Page 2035)

Principle of operation

Frequency measurement

The signal frequencies are identified from the input signals of channel 0 or 1 of the electronic module. To calculate the frequency the signals are measured within a configurable gate.

The frequency is displayed as 16-bit value in fixed-point format and transferred to the PII.

The frequency meter calculates the frequency according to the follow formula:

$$\text{Frequency [Hz]} = \frac{\text{Number of rising edges at digital input}}{\text{Measuring window [s]}}$$

Exceeding the input frequency

If the input frequency exceeds 5kHz, $7FFF_H$ is reported as actual value. If the input frequency is above approx. 8 kHz it is no longer possible to display correct actual values.

See also

Frequency measurement properties (Page 2032)

Configuring frequency meters

Procedure

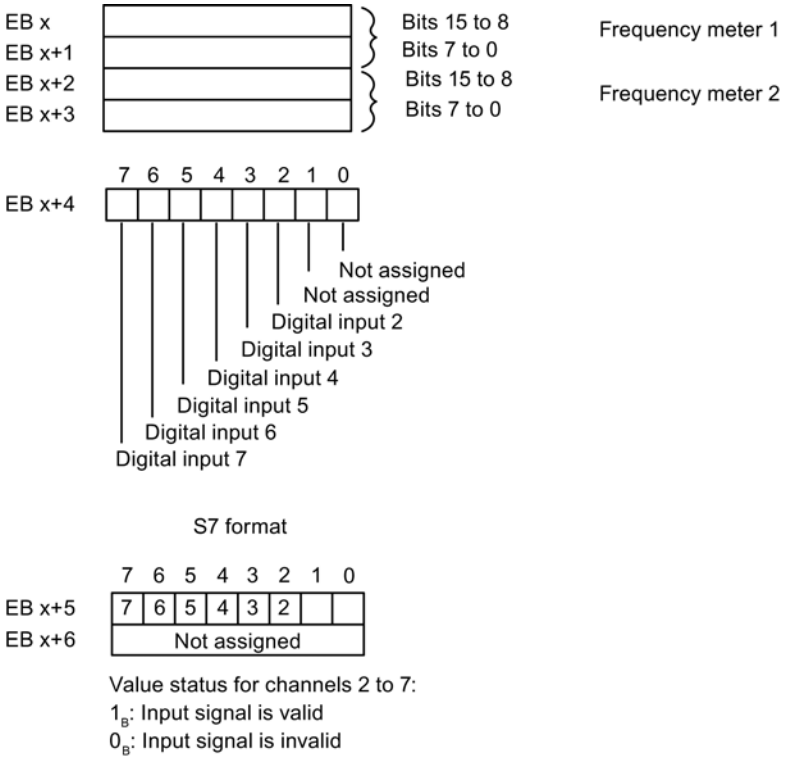
1. Using the mouse, pull module 8 DI Namur from the hardware catalog into distributed I/O station ET 200iSP.
2. Select the required configuration (channel 0..1: "Trace", channel 2..7: "DI"). In the module properties (inspector window), you can find this setting under "Parameters > Inputs > Configuration".

Configuration 0..1: "Trace", channel 2..7: "DI"

Assignment of the digital inputs on electronic module 8 DI NAMUR

Digital input	Terminal	Assignment
Channel 0	1, 2	Frequency counter 1
Channel 1	5, 6	Frequency counter 2
Channel 2	9, 10	Digital input 2
Channel 3	13, 14	Digital input 3
Channel 4	3, 4	Digital input 4
Channel 5	7, 8	Digital input 5
Channel 6	11, 12	Digital input 6
Channel 7	15, 16	Digital input 7

Assignment of process image input (PII) for configuration of channel 0..1: "Trace", channel 2..7: "DI"



Assignment of the process image output (PIQ): The PIQ is not assigned.

See also

Frequency measurement properties (Page 2032)

Assigning parameters for the frequency meters

Parameters for frequency meter

Only those parameters that are relevant for the frequency meters are explained below. These are part of the parameters of electronic module 8 DI NAMUR.

Table 8-117 Parameters for the frequency meters

Parameter	Setting	Description
Sensor type frequency inputs	• Channel disabled • NAMUR sensor • Single contact, no load resistance	Select the sensor for the relevant frequency meter for channel 0 or 1.
Measuring window (GATE)	• 50 ms • 200 ms • 1 s	Select the required measuring window for channel 0 or 1. To achieve the highest possible accuracy when metering frequencies, remember the following rules: • High frequencies (> 4 kHz): Set a low measuring window (50 ms) • Variable/medium frequencies: set medium measuring window (200 ms) • Low frequencies (< 1 kHz): Set a high measuring window (1 s)

See also

Frequency measurement properties (Page 2032)

ET 200eco PN

ET 200eco PN Distributed I/O Device

Definition

The ET 200eco PN distributed I/O device is a compact PROFINET IO device in degree of protection IP 65/66 or IP 67 and UL Enclosure Type 4x, Indoor use only.

Field of application

The fields of application of the ET 200eco PN are derived from its special properties.

- A robust design and degree of protection IP 65/66 or IP 67 make the ET 200eco PN distributed I/O device suitable in particular for use in rugged industrial environments.
- The compact design of the ET 200eco PN is particularly favorable for applications in confined areas.
- The easy handling of ET 200eco PN facilitates efficient commissioning and maintenance.

Properties

The ET 200eco PN has the following properties:

- Integrated switch with 2 ports
- Supported Ethernet services:
  - ping
  - arp
  - Network diagnostics (SNMP)
  - LLDP
- Interrupts
  - Diagnostics interrupts
  - Maintenance interrupts
- Port diagnostics
- Isochronous real-time communication
- Prioritized startup
- Device replacement without programming device
- Media redundancy
- Connection to intelligent sensors/actuators via IO link master interface module.

IO Controller

The ET 200eco PN can communicate with all IO Controllers that conform to IEC 61158.

ET 200eco PN can be configured on a CPU with advanced diagnostics.

See also

Documentation on ET 200eco PN (http://support.automation.siemens.com/WW/view/en/29999018)

Parameter description analog input

Group diagnostics

You can generally enable and disable the diagnostics function of the device with this parameter.

The "Fault" and "Parameter assignment error" diagnostics functions are always independent of the group diagnostics.

Diagnostics, missing 1L+

If you enable this parameter, the check for missing supply voltage is enabled.

Diagnostics, sensor supply short circuit

If you enable this parameter, a diagnostics event is generated if a short-circuit of the sensor supply to ground is detected and the channel is enabled. The sensor supply is monitored for connectors X1, X3, X5 and X7. It is not possible to differentiate which connector has experienced the sensor short circuit.

Interference frequency suppression

With this parameter, you set the integration time of the device, based on the selected interference frequency. Select the frequency of the supply voltage used. Interference frequency suppression **Off** means 500 Hz, which corresponds to an integration time of 2 ms for a measurement channel.

Temperature unit

Specify the unit of the temperature measurement here.

Measurement type (channel-wise)

With this parameter, you set the measurement type, for example, voltage. For any unused channels, you must select the **disabled** setting. For a disabled channel, the conversion time and integration time of the channel = 0 s and the cycle time is optimized.

Measuring range

With this parameter, you set the measuring range of the selected measurement type.

Temperature coefficient (for RTD, thermoresistor)

The correction factor for the temperature coefficients (α-value) indicates by what extent the resistance of specific material changes relatively if the temperature increases by 1 °C.

The α-values conform to EN 60751, GOST 6651, JIS C 1604 and ASTM E-1137.

The temperature coefficient depends on the chemical composition of the material.

Smoothing

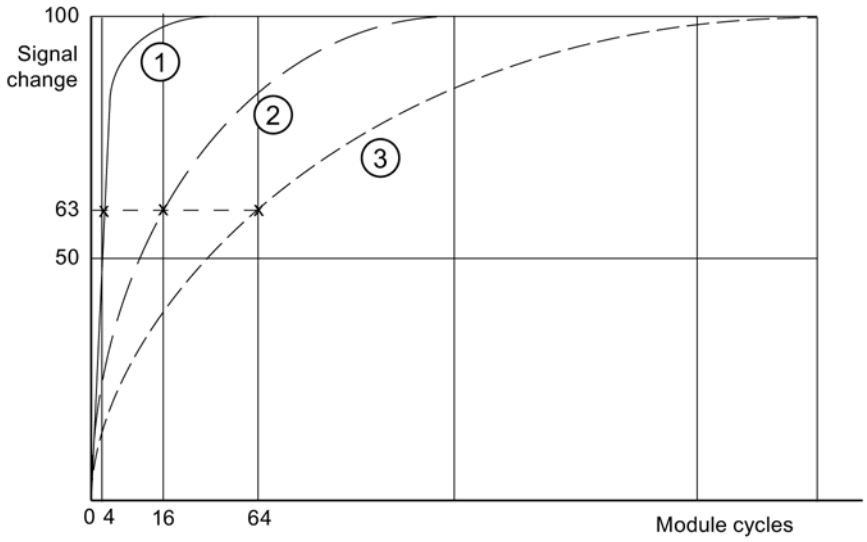
Smoothing of the analog values produces a stable analog signal for further processing. The smoothing of analog values is useful when handling wanted signals (measured values) with a slow rate of change, for example, temperature measurements.

The measured values are smoothed by digital filtering. To achieve smoothing, the device generates a mean value from a specified number of converted (digitized) analog values.

You assign a maximum of four levels for the smoothing (none, weak, medium, strong). The level determines the number of module cycles, from which the mean value is generated.

The stronger the smoothing, the more stable the smoothed analog value and the longer it takes until the smoothed analog value is applied following a signal change (see the example below).

The figure below shows the number of cycles a module requires to apply the smoothed analog value at almost 100% after a step response, based on the smoothing function settings. The figure applies to all signal changes at the analog input. The smoothing value defines the number of cycles a module requires to reach 63% of the end value of the changed signal.



① Smoothing, weak
② Smoothing, medium
③ Smoothing, strong

Diagnostics, wire break

When this parameter is enabled, the **Wire break** diagnostics event is generated when a wire break is detected.

Observe the rules outlined below to handle a wire break in the 1 to 5 V and 4 to 20 mA measuring ranges:

Parameter	Event	Measured value	Explanation
Enable wire break[1]	Wire break	$7FFF_H$	Diagnostics, **wire break**
Wire break disabled[1] Underflow enabled	Wire break	8000_H	Measured value after leaving the underrange Diagnostic message **Lower limit value undershot**
Wire break disabled[1] Underflow disabled	Wire break	8000_H	Measured value after leaving the underrange
[1] Measuring range limits for wire break detection and measuring range undershoot detection: • 1 to 5 V: At 0.296 V • 4 to 20 mA: At 1.185 mA			

Diagnostics, underflow

If you enable this parameter, the **Underflow** diagnostics event is generated when the measured value reaches the underflow range.

Diagnostics, overflow

If you enable this parameter, the **Overflow** diagnostics event is generated when the measured value reaches the overflow range.

Reference junction for thermoresistor (TC)

A difference in temperature between the measuring point and the free ends of the thermocouple (terminal point) generates a voltage between the free ends, namely the thermoelectric voltage. The value of this thermoelectric voltage is determined by the temperature difference between the measuring point and the free ends and by the type of material combination of the thermocouple. Since a thermocouple always measures a temperature difference, the free ends at the reference junction must be maintained at a known temperature in order to determine the temperature of the measuring point.

If you specify **Internal compensation**, the temperature of the measuring point in the housing of the I/O device is measured. With the **External compensation** setting, you can connect a compensation box in series in order to increase the accuracy of the temperature measurement.

Parameter description analog output

Group diagnostics

You can generally enable and disable the diagnostics function of the device with this parameter.

The "Fault" and "Parameter assignment error" diagnostics functions are always independent of the group diagnostics.

Diagnostics, missing 1L+

If you enable this parameter, the check for missing supply voltage is enabled.

Diagnostics, sensor supply short circuit

When this parameter is enabled, the system generates a diagnostics event if it detects a short-circuit of the sensor supply to ground. This diagnostics function is activated when the group diagnostics function is enabled.

Response to CPU/Master STOP

Select how the module's outputs will respond to a CPU STOP:

- Shut down
  The I/O device goes to the safe state. The process image output is deleted (=0).
- Keep last value
  The I/O device retains the last value to be output before STOP.
- Substitute value
  The I/O device outputs the value for the channel set beforehand.

Note

Make sure that the plant is always in a safe state if "Keep last value" is selected.

Type of output

With this parameter, you set the output type, for example, voltage. For any unused channels, select the **disabled** setting. For a disabled channel, the conversion time and integration time of the channel = 0 s, and the cycle time is optimized.

Output range

With this parameter, you set the output range of the selected output type.

Diagnostics, wire break (in current mode)

When this parameter is enabled, the **Wire break** diagnostics event is generated when a wire break is detected. This diagnostics event cannot be detected in the zero range.

Diagnostics, short circuit (in voltage mode)

If you enable this parameter, a diagnostics event is generated in the event of a short circuit in the output line. This diagnostics event cannot be detected in the zero range.

Diagnostics, overload

If you enable this parameter, the diagnostics event is generated in the event of an overload.

Substitute values

With this parameter, you enter a substitute value that the module is to output in CPU-STOP mode. The substitute value must be in the nominal range, overrange, or underrange.

ET 200SP

ET 200SP distributed I/O system

Definition

The ET 200SP distributed I/O system is a scalable, highly flexible distributed I/O system for connection of process signals to a central controller via a field bus.

Application area

The ET 200SP is a multi-functional distributed I/O system for various fields of application. The scalable design allows you to configure the system exactly to the specific requirements on location.

The ET 200SP is approved for degree of protection IP 20 and for installation in a control cabinet.

Structure

The ET 200SP is mounted on a mounting rail and comprises:

- An interface module which can communicate with all IO controllers that conform to the PROFINET standard IEC 61158
- Up to 32 I/O modules which can be inserted on passive BaseUnits in any combination
- A server module that completes the design of the ET 200SP.

Interface module parameters

Status bytes

Status bytes

If you enable the "Status bytes" option, 4 bytes of input data are reserved for the status of the supply voltage of each I/O module.

	7	6	5	4	3	2	1	0
Byte 0	8	7	6	5	4	3	2	1
Byte 1	16	15	14	13	12	11	10	9
Byte 2	24	23	22	21	20	19	18	17
Byte 3	32	31	30	29	28	27	26	25

I/O module slots

Bit = 0: Load voltage missing or I/O module not present

Bit = 1: Load voltage and I/O module present

Note

An inserted or missing server module always reports for the slot bit = 0.

Group diagnostics, missing supply voltage L+

Group diagnostics, missing supply voltage L+

This diagnostics is a group diagnostics that covers the supply voltage status of all I/O modules of a potential group which are defined by BaseUnits with incoming power supply (light-colored BaseUnit BU...D).

The group diagnostics is formed from the states of the supply voltage of the inserted I/O modules within the potential group.

The group diagnostics does not depend on the "Missing supply voltage L+" parameter of the I/O modules being enabled.

The server module does not influence the missing supply voltage L+ group diagnostics.

Requirements for the correct operation of the group diagnostics for missing supply voltage L+:

- I/O modules or BU covers must be inserted on the light-colored and dark-colored BaseUnits. If no I/O module is inserted on a light-colored BaseUnit, the start of this potential group will not be detected by the interface module; the I/O modules of this potential group will thus belong to the previous potential group. A supply voltage L+ group error will then be assigned to the wrong potential group.
  When an I/O module is inserted on the light-colored BaseUnit, the interface module detects the new potential group, re-evaluates the status, and reports a new group diagnostics in the case of an error.
- The server module must be inserted.
  The server module itself does not influence the missing supply voltage L+ group diagnostics.

Configuration control

Operating principle

Configuration control allows you to operate various real configurations (options) with a single configuration of the distributed I/O device ET 200SP.

Configuration control provides you with the option of configuring the ET 200SP distributed I/O device with its maximum configuration and nevertheless operating it with missing modules. If missing modules are retrofitted later, no new configuration is required and the hardware configuration does not have to be reloaded either.

Using control data records, which are transferred to the interface module in the user program, you define a current preset configuration.

- The configured module is not present on a slot.
  - A base unit cover may be inserted on this slot instead of the configured I/O module. As there is no configured module on the slot, the term "Configuration control with empty slots" is also used.
  - The module that is configured to the right of the missing module can be inserted on this slot instead of the configured module. The missing module makes the actual configuration appear pushed together. As the configured module is missing but no gap arises in the configuration, this is also referred to as a "Configuration without empty slots".
- The configuration is extended by an already configured module.
  - In the case of configuration control with empty spaces, you extend the configuration by inserting the configured module on the corresponding empty slot.
  - In the case of configuration control without empty spaces, insert the configured module on the right-hand side next to the last module of the ET 200SP.

Addressing the interface module using the hardware identifier

To transfer data record 196 with the instruction WRREC, you must enter the hardware identifier of the interface module as input parameter for the instruction. The system constant of this hardware identifier is, for example, "IO-Device_2 [Head]". The system constants of a selected device are, for example, displayed in the network view in the "System constants" tab. Use the associated value for addressing.

More information and examples

For information on the rules for configuration control, the structure of the control data record, and the behavior during operation as well as examples of the structure of the control data record for various configurations, refer to the IM 155-6 PN interface module manual (http://support.automation.siemens.com/WW/view/en/59768173).

Output module parameters

Substitute value reaction

Substitute value reaction

In the ET 200SP, the substitute value reaction is executed by the IO controller per slot.

The respective output reacts according to its set substitute value reaction:

- "Turn off"
- "Output substitute value"
- "Keep last value"

This substitute value reaction is triggered in the following cases:

- IO controller in STOP
- Controller failure (connection interruption)
- Firmware update
- Reset to factory settings
- More than one I/O module withdrawn simultaneously
- Disable the IO device
- Station stop
  - Missing server module
  - More than one I/O module withdrawn simultaneously
  - At least one I/O module is inserted on the wrong BaseUnit

Note

Reducing a configuration

If you reduce the configuration of the ET 200SP and download the configuration to the CPU, the modules which are no longer configured but still present retain their original substitute value reaction. This applies until the supply voltage on the BaseUnit BU...D or on the interface module is turned off.

Input module parameters

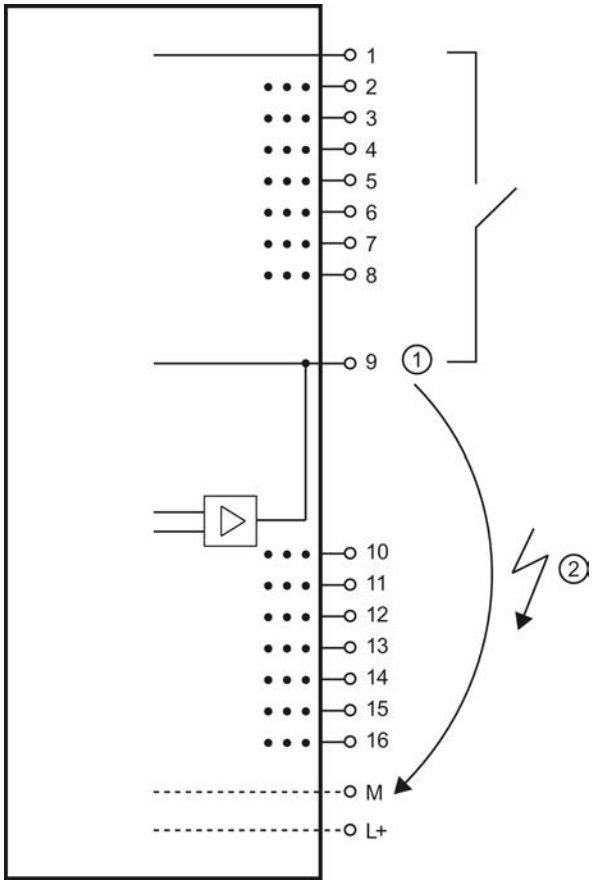
Parameters of the digital input modules

Diagnostics missing supply voltage L+

Enabling of the diagnostics for missing or insufficient supply voltage L+.

Diagnostics short-circuit to ground

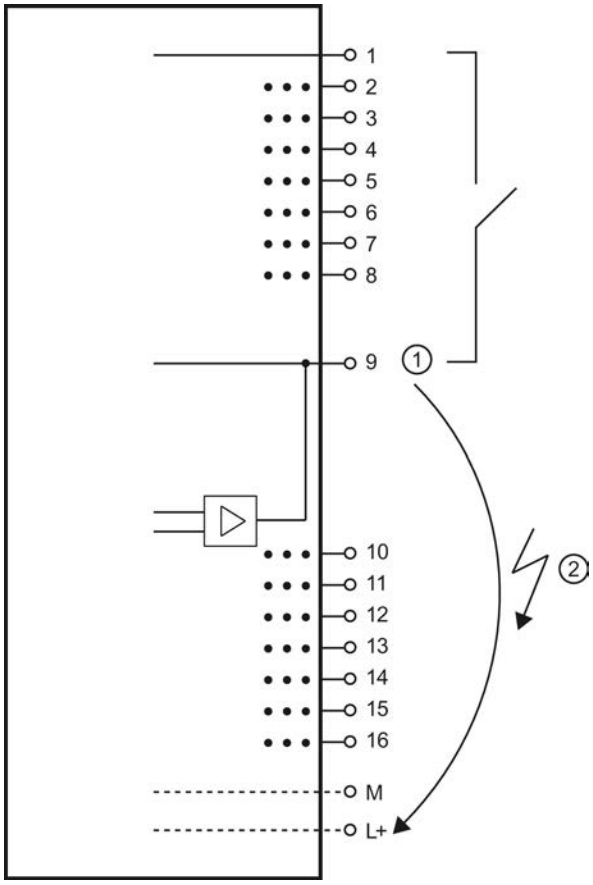
Enabling of the diagnostics if a short-circuit of the actuator supply to ground occurs.



① Encoder supply
② Short-circuit

Diagnostics short-circuit to L+

Enabling of the diagnostics if a short-circuit of the encoder supply to L+ occurs.



① Encoder supply
② Short-circuit

Diagnostics wire break

Enabling diagnostics if the line to the encoder is interrupted.

Operating mode

Determines whether a channel is enabled or disabled.

Pulse extension (only High Feature modules)

Pulse extension is a function for changing a digital input signal. A pulse at a digital input is extended to at least the configured length. If the input pulse is already longer than the configured length, the pulse is not changed.

Pulse extension is started whenever the state of the input signal changes and no pulse extension is active for this channel.

Potential group of the left module / new potential group

Specifies whether the I/O module is located on a base unit with supply voltage feed (new potential group) or whether it is it located on a base unit without supply voltage feed (in which case it belongs to the potential group of the left module).

Parameters of the analog input modules

Missing supply voltage L+

Enabling of the diagnostics, with missing or too little supply voltage L+.

Reference junction (AI 4xRTD/TC 2-/3-/4-wire HF)

A BaseUnit with internal temperature sensor (BU..T) or the channel 0 of the I/O module can be used as reference junction, provided this has been configured as "Thermal resistance Pt100 climatic range Celsius".

A possible configuration is shown below:

Table 8-118 RTD channel

Setting	Description
No reference channel operation	Temperature value at channel 0 can be used as module-wide reference value if the parameters of the other channels are assigned accordingly.
Reference channel of Group 0	The channel acts as sender for the reference junction temperature of Group 0. Distribution is performed via the interface module.

Table 8-119 TC channel

Setting	Description
Reference channel of the module	The corresponding TC channel uses the channel 0 of the same module as reference junction temperature. This must be set as "Thermal resistance Pt 100 climatic range Celsius" and "No reference channel operation"; otherwise, reference junction diagnostics is triggered.
Internal reference junction	The reference junction temperature is read by an internal temperature sensor on the BaseUnit. Reference junction diagnostics is triggered if there is a wrong BaseUnit type.
Reference channel of Group 0	With the setting "TC" (thermocouple...), the channel acts as receiver for the reference junction temperature of Group 0
Fixed reference temperature	No temperature compensation occurs. The linearization is executed with an assumed reference junction temperature of 0 °C.

Overflow

Enabling of the diagnostics if the measured value exceeds the overflow range.

Underflow

Enabling of the diagnostics if the measured value falls below the underflow range.

Wire break

Enabling of the diagnostics if the module has no current flow or too low a current flow for the measurement on the corresponding configured input.

Smoothing

The individual measurements are smoothed using digital filtering. The smoothing can be set in 4 stages, whereby the smoothing factor k multiplied by the cycle time of the I/O module corresponds to the time constant of the smoothing filter. The larger the smoothing, the larger the time constant of the filter.

The following figure shows the step response for the various smoothing factors depending on the number of module cycles.

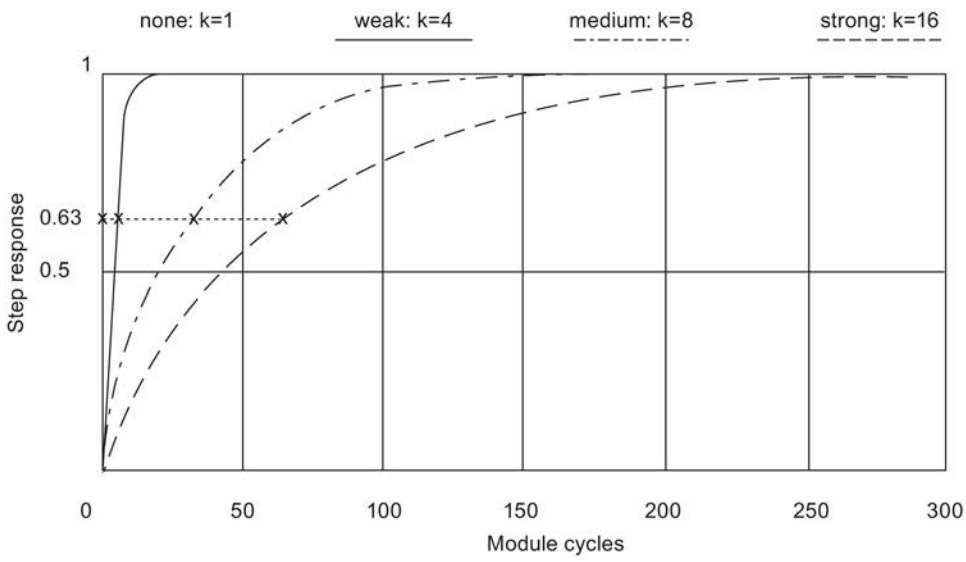


Figure 8-57 Smoothing with AI 4×RTD/TC 2-/3-/4-wire HF

Interference frequency suppression

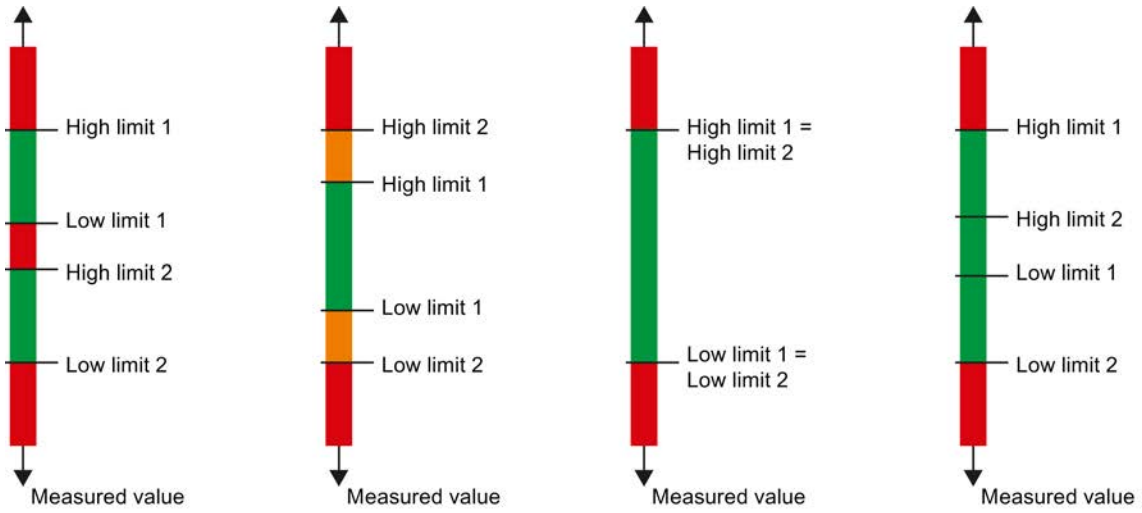
With analog input modules, suppresses the disturbance caused by the frequency of the AC network used.

The frequency of the AC network may interfere with measured values, particularly for measurements within low voltage ranges and when thermocouples are being used. This parameter is used to define the predominant power frequency of the system.

Hardware interrupt limits

If the high limit 1/2 or the low limit 1/2 is violated, the module triggers a hardware interrupt.

Below are some examples for the selection of the limits 1 and 2.



Low limit 1/2

Specify a threshold whose undershoot triggers a hardware interrupt.

High limit 1/2

Specify a threshold whose overrange triggers a hardware interrupt.

Potential group of the left module / new potential group

Specifies whether the I/O module is located on a base unit with supply voltage feed (new potential group) or whether it is it located on a base unit without supply voltage feed (in which case it belongs to the potential group of the left module).

Temperature coefficient (measuring type thermoresistor)

The correction factor for the temperature coefficient (α value) defines the relative rate of change of the resistance of a specific material at a temperature rise of 1 °C.

The temperature coefficient depends on the chemical composition of the material. In Europe, only one value is used per sensor type (default value).

The further values facilitate a sensor-specific setting of the temperature coefficient and enhance accuracy.

See also

Special features of AI 4xRTD/TC 2-/3-/4-wire HF (Page 2050)

Special features of AI 4xRTD/TC 2-/3-/4-wire HF

Use of Cu10 sensors

- Select "3-wire thermal resistor" and "Cu10" in the parameter assignment.
- Wire the Cu10 sensor using 3-wire connection technology.
- An automatic, internal compensation of the line resistance of the missing measuring line takes place during operation.

Note

To ensure optimum line compensation for Cu10, please note the following:

- The sum of cable resistance and measuring resistance must not exceed 31 Ω.
- Cable resistance should not exceed 8 Ω if you want to use the temperature range up to over 312 °C.
  Example: A 200 m long copper cable with 0.5 mm^2 core cross-section has approx. 7 Ω. A lower cross-section reduces the permissible cable length accordingly.

Use of PTC resistors

PTCs are suitable for temperature monitoring of or as thermal protective equipment for complex drives or transformer windings.

- Select "2-wire resistor" and "PTC" in the parameter assignment.
- Connect the PTC using 2-wire technology.
- Use type A PTC resistors (PTC thermistors) in accordance with DIN/VDE 0660, Part 302.
- If the "Over-/underflow" diagnostics is enabled, a "low limit violation" diagnostics which shows a short circuit is generated for resistance values < 18 Ω.
- Sensor data on PTC resistance:

Table 8-120 Use of PTC resistors

Property	Technical specifications	Note
Switching points	**Reaction to rising temperature**	
	< 550 Ω	**Normal range:** • SIMATIC S7: bit 0 = "0", bit 2 = "0" (in the PII)
	550 Ω to 1650 Ω	**Prewarning range:** • SIMATIC S7: bit 0 = "0", bit 2 = "1" (in the PII)
	< 1650 Ω	**Response range:** • SIMATIC S7: bit 0 = "1", bit 2 = "0" (in the PII)
	Reaction to falling temperature	
	< 750 Ω	**Response range:** • SIMATIC S7: bit 0 = "1", bit 2 = "0" (in the PII)
	750 Ω to 540 Ω	**Prewarning range:** • SIMATIC S7: bit 0 = "0", bit 2 = "1" (in the PII)

Property	Technical specifications	Note
	< 540 Ω	**Normal range:** • SIMATIC S7: bit 0 = "0", bit 2 = "0" (in the PII)
(TNF-5) °C (TNF+5) °C (TNF+15) °C Measuring voltage Voltage on the PTC	max. 550 Ω min. 1330 Ω min. 4000 Ω max. 7.5 V	RRT= rated response temperature

- Assignment in the process image inputs (PII) with SIMATIC S7

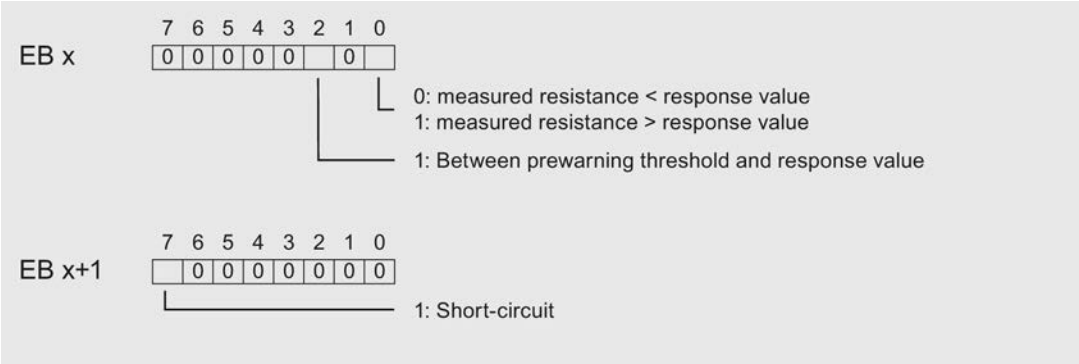


Figure 8-58 Assignment in the process image inputs (PII)

- Notes on programming

Note

Only the bits 0+2 are relevant for the evaluation in the process image inputs. You can use the bits 0+2 to monitor the temperature, for example, of a motor.

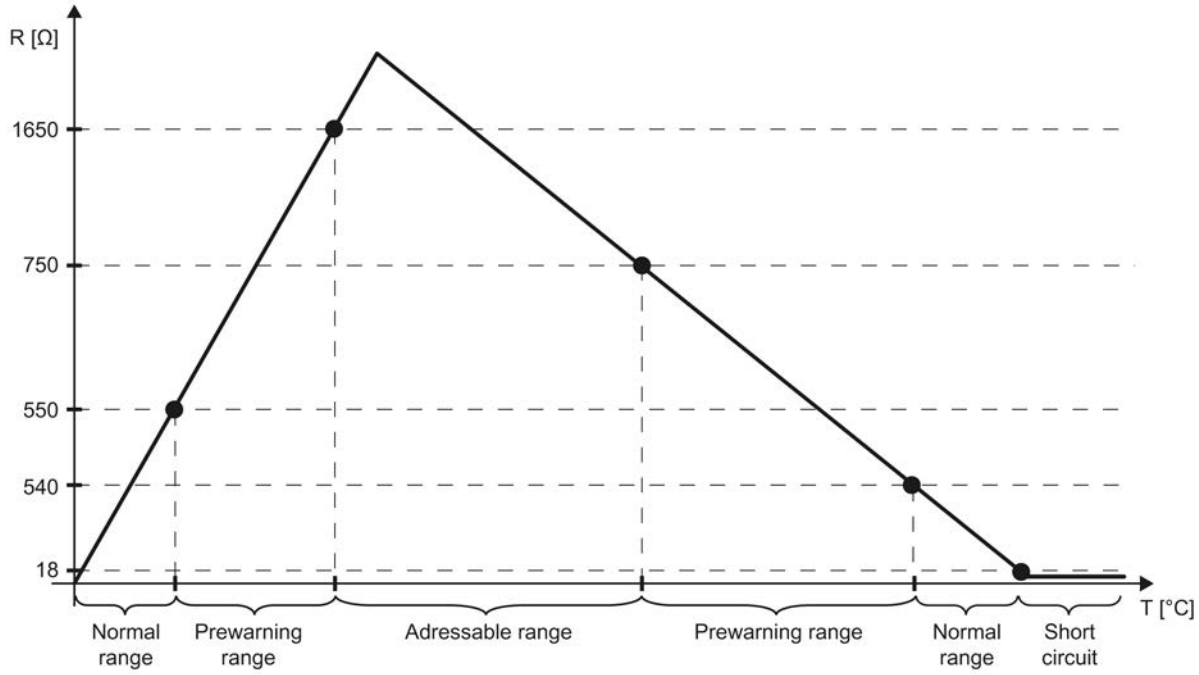
The bits 0+2 in the process image inputs have no latching function. When you are assigning parameters, take into consideration that a motor, for example, starts up in a controlled manner (via an acknowledgment).

Bits 0+2 can never be set simultaneously, but are instead set consecutively.

For safety reasons, always evaluate the diagnostic entries of the AI 4×RTD/TC 2-/3-/4-wire HF, as no measurement is possible if I/O modules are unplugged, if the supply voltage of the I/O module has failed, or if there is a wire break or short circuit of the measuring lines.

Example

The following diagram shows the temperature variation and the associated switching points.



See also

Parameters of the analog input modules (Page 2047)

ET 200MP

ET 200MP distributed I/O system

Definition

The ET 200MP distributed I/O system is a scalable and flexible distributed I/O system for connection of process signals to a central controller via a field bus.

Application area

The ET 200MP is a multi-functional distributed I/O system for various fields of application. The scalable design allows you to configure the system exactly to the specific requirements on location.

The ET 200MP complies with IP 20 degree of protection and is intended for installation in a control cabinet.

Structure

The ET 200MP is installed on a mounting rail and comprises:

- An interface module that communicates with all IO controllers conforming to the PROFINET standard IEC 61158
- Up to 30 modules (power supply modules and I/O modules from the S7-1500 I/O range) can be inserted to the right of the interface module.
- If you insert a power supply module to the left of the interface module, this yields a possible maximum configuration of 32 modules in total.
- The number of insertable I/O modules is limited by their power requirements.

Slot rules

- Slot 0: Power supply module (optional)
- Slot 1: Interface module
- Slot 2 to 31: I/O modules or power supply modules

Interface module parameters

Supply voltage L+ connected

Parameter "Supply voltage L+ connected"

This parameter influences the diagnostics and the checking of the power budget.

- Diagnostics of the ET 200MP:
  If the actual configuration does not conform to the setpoint configuration with regard to the supply voltage of the interface module, the interface module generates a diagnostic alarm. Example: You have deactivated the "Supply voltage L+ connected" option, but you have connected the supply voltage in the actual configuration.
- Power budget check during configuration:
  The power budget changes in accordance with the parameter setting: Either the interface module feeds power into the backplane bus or it draws power from the backplane bus.

The default ("Supply voltage L+ connected" option is **activated**) means that the front of the interface module is supplied with 24 V DC and the power is stored in the backplane bus.

If the "Supply voltage L+ connected" option is **deactivated**, the interface module may not be supplied with 24 V DC on the front.

In this case, insert a power supply unit (PS) on the left next to the interface module that supplies the interface module and the modules to the right of the interface module.

Note

We recommend that you always supply the interface module on the front side with 24 V DC. If a system power supply unit (PS) is inserted and connected additionally **before** or on the left next to the interface module, both the power from the system power supply unit (PS) as well as the power from the integrated power supply of the interface module are then available to the configuration.

In this case, you do not have to change the default setting of the parameter.

Configuration control

Operating principle

Configuration control allows you to operate various real configurations (options) with a single configuration of the ET 200MP distributed I/O device.

Configuration control provides you with the option of configuring the ET 200MP distributed I/O device with its maximum configuration and nevertheless operating it with missing modules. If missing modules are retrofitted later, no new configuration is required and the hardware configuration does not have to be reloaded either.

Using control data records, which are transferred to the interface module in the user program, you define a current preset configuration.

We distinguish between the following procedures:

- Configuring configuration control without empty slots (option handling)
- Expanding configuration (step-by-step commissioning)

Requirement

Enable the Configuration control" option of the interface module ("Configuration control" area). Control takes place with control data record 196 to specify the required configuration. Configuration control is not ready for operation without a control data record. You transfer the control data record with the instruction WRREC in the user program.

Addressing the interface module using the hardware identifier

To transfer data record 196 with the instruction WRREC, you must enter the hardware identifier of the interface module as input parameter for the instruction. The system constant of this hardware identifier is, for example, "IO-Device._V2 [Head]". The system constants of a selected device are, for example, displayed in the network view in the "System constants" tab. Use the associated value for addressing.

More information and examples

For information on the rules for configuration control, the structure of the control data record and the behavior during operation, as well as examples of the structure of the control data record for various configurations, refer to the IM 155-5 PN ST interface module manual (http://support.automation.siemens.com/WW/view/en/67295970).

Input module parameters

Parameters of the analog input modules

Missing supply voltage L+

Enabling of the diagnostics for missing or insufficient supply voltage L+.

Wire break

Enabling of the diagnostics if the module has no current flow or the current is too weak for the measurement at the corresponding configured input or the applied voltage is too low.

Current limit for wire break diagnostics

Threshold at which a wire break is reported. The value can be set to 1.185 mA or 3.6 mA, depending on the sensor used.

Overflow

Enabling of the diagnostics if the measured value exceeds the overrange.

Underflow

Enabling of the diagnostics if the measured value undershoots the underrange.

Common mode error

Enable diagnostics if the valid common mode voltage is exceeded.

Reference channel error (only for AI 8xU/I/RTD/TC ST)

- Enable diagnostics on error at the temperature compensation channel, e.g. wire break.
- Dynamic reference temperature compensation type is configured and no reference temperature has been transferred to the module yet.

Temperature coefficient

The temperature coefficient depends on the chemical composition of the material. In Europe, only one value is used per sensor type (default value).

The correction factor for the temperature coefficient (α value) specifies how much the resistance of a certain material changes when the temperature is raised by 1 °C.

The further values facilitate a sensor-specific setting of the temperature coefficient and enhance accuracy.

Interference frequency suppression

At analog input modules, this suppresses interference caused by the frequency of AC mains.

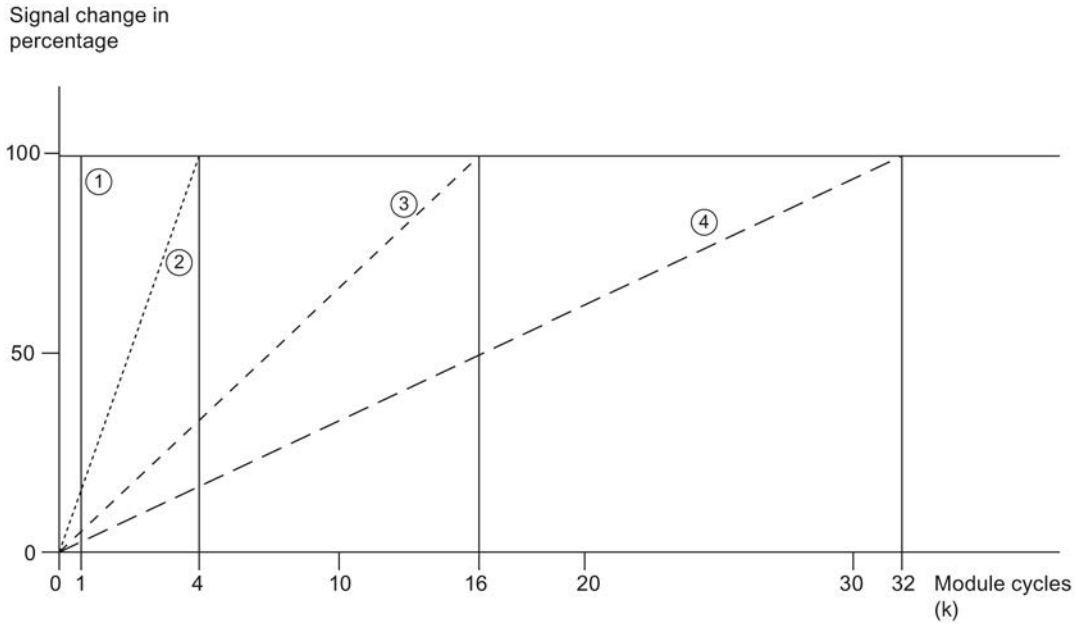
The frequency of the AC network may interfere with measured values, particularly for measurements within low voltage ranges and when thermocouples are being used. With this parameter, you define the mains frequency in your system.

Smoothing

The individual measured values are smoothed using filtering. Smoothing can be set in 4 stages for the analog input modules AI 8xU/I/RTD/TC ST and AI 8xU/I HS.

Smoothing time = number of module cycles (k) x cycle time of the module.

The figure below shows the number of module cycles after which the smoothed analog value is almost 100%, depending on the set smoothing. Is valid for each signal change at the analog input.



① None (k = 1)
② Weak (k = 4)
③ Medium (k = 16)
④ Strong (k = 32)

Reference junction (only for AI 8xU/I/RTD/TC ST)

The following settings can be configured for the reference junction parameter:

Table 8-121 Possible parameter assignments for the reference junction parameter

Setting	Description
Fixed reference temperature	The reference junction temperature is configured and stored in the module as a fixed value.
Dynamic reference temperature	The reference junction temperature is transferred in the user program from the CPU to the module by data records 192 to 199 using the WRREC (SFB 53) instruction.
Internal reference junction	The reference junction temperature is determined using an integrated sensor of the module.
Reference channel of the module	The reference junction temperature is determined using an external resistance thermometer (RTD) at the reference channel (COMP) of the module.

Note

Fixed reference temperature

During parameter assignment of a thermocouple Type B, only the setting "Fixed reference temperature" with a temperature of 0 °C is possible.

Enable hardware interrupt 1 or 2

Enable a hardware interrupt if high limit 1 or 2 is exceeded, or low limit 1 or 2 is violated.

Low limit 1 or 2

Specifies the low limit threshold that triggers hardware interrupt 1 or 2.

High limit 1 or 2

Specifies the high limit threshold that triggers hardware interrupt 1 or 2.

Temperature compensation for thermocouples

Introduction

You have several options of measuring the reference junction temperature in order to obtain an absolute temperature value as a function of the temperature difference between the reference junction and the measuring point.

You can use various compensation options depending on the required location of the reference junction.

Note

During parameter assignment of a thermocouple Type B, only the setting "Fixed reference temperature" with a temperature of 0 °C is possible.

Options of compensating for the reference junction temperature

Compensation options	Explanation	Application case
Internal reference junction	With this compensation, the reference junction temperature is determined using an integrated sensor of the module. **Procedure** Connect the thermocouple to the I/O module directly or with compensating lines.	• For the connection, you use compensating lines matching the thermocouple material. • If the reference junction temperature and the module temperature are identical in your system, you may also use lines made from a different material.
Reference channel of the module	The reference junction temperature is determined using an external resistance thermometer (RTD). **Procedure** Connect the thermocouple to the supply lines at the reference junction, either directly or with compensating lines. You connect the supply lines to the appropriate terminals of the module. Connect the resistance thermometer (RTD) to the reference channel of the module. The resistance thermometer (RTD) must be placed in the area of the reference junction.	• You want to measure the temperature directly at the reference junction. • The measured temperatures of all channels that you have configured for this compensation type is corrected automatically by the temperature value of the reference junction. • You can use inexpensive lines, e.g., copper lines, from the reference junction to the module.

Compensation options	Explanation	Application case
Dynamic reference temperature	The temperature of the reference junction is determined via a module. This temperature value is transferred to other modules via a data record in the user program. **Procedure** Connect the resistance thermometer (RTD) for the reference junction to any channel. The reference junction temperature is communicated from the CPU to the module by data records 192 to 199 using the WRREC instruction.	• You use multiple modules at the reference junction and can therefore compensate all channels using a common temperature value. • You require only one resistance thermometer (RTD) to acquire the temperature value. • You can use inexpensive lines, e.g., copper lines, from the reference junction to the module.
Fixed reference temperature	The reference junction temperature is stored in the module as a fixed value. **Procedure** Connect the thermocouple to the supply lines at the reference junction, either directly or with compensating lines. You connect the supply lines to the appropriate terminals of the module. When configuring the module, specify a fixed temperature value for the reference junction (e.g. 20 °C).	• You keep the reference junction temperature constant and know the temperature value. • You can use inexpensive lines, e.g., copper lines, from the reference junction to the module.

Output module parameters

Parameters of the analog output modules

Missing supply voltage L+

Enabling of the diagnostics, with missing or too little supply voltage L+.

Short-circuit to ground

Enabling of the diagnostics if a short-circuit of the actuator supply to ground occurs.

Wire break

Enabling diagnostics if the line to the encoder is interrupted.

Overflow

Enabling of the diagnostics if the measured value exceeds the overflow range.

Underflow

Enabling of the diagnostics if the measured value falls below the underflow range.

Reaction to CPU STOP

Determines the reaction of the output to the CPU going into STOP state.

Substitute value

The substitute values are values that the outputs (the output) issue in the event of a CPU STOP.

ET 200M

Configuring an ET 200M

Introduction

For the ET 200M series, you can find a wide range of modules in the hardware catalog under "Distributed I/O".

Configuration and parameter assignment

Information on configuration and parameter assignment can be found in the following sections.

ET 200M configuration

Definition

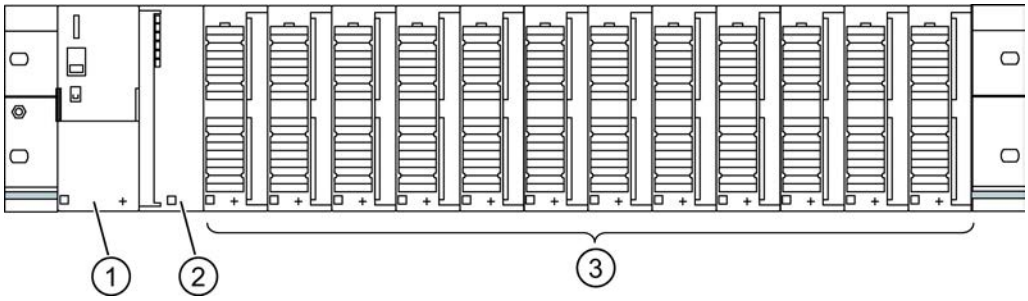
The distributed IO device, ET 200M, is a modular DP slave with an IP 20 degree of protection.

The ET 200M has the configuration technology of the S7-300 automation system and consists of an IM 153-x and I/O modules of the S7-300.

ET 200M supports communication with:

- all DP masters compliant with IEC 61784-1:2002 Ed1 CP 3/1
- all IO controllers compliant with IEC 61158

Configuration of the ET 200M (example)



① Power supply module PS 307
② Interface module IM 153-x
③ up to 12 I/O modules (SM / FM / CP)

Configuration of the 'Module replacement during operation' function

Introduction

The ET 200M supports the "Replace modules during operation" function and the associated pull/plug interrupt.

The "Replace modules during operation" function makes it possible for you to pull modules from or plug modules into the ET 200M rack during operation.

Requirement

You have configured an interface module that supports replacing modules during operation. (as of IM 153-1, order no. 153-1AA02-0XB0).

In addition, the configured CPU must also support the function, e.g. for PROFIBUS an S7-400 with DP interface.

You must use the active backplane bus (bus rail with slots) for the hardware configuration. The conventional profile rail with bus connectors between the modules does not support this function.

Configuring

If the configuration requirements have been met, the "Replace modules during operation" parameter is available for selection in the inspector window's "Module parameters" area. Below this parameter, a table for the configured modules is displayed, which shows the required active bus modules for the hardware configuration.

For a PROFIBUS configuration, the "Startup if preset configuration does not match actual configuration" option is displayed. This option is automatically enabled if "Replace modules during operation" is enabled.

Configuring HART variables

Introduction

Numerous HART field devices make available additional measured quantities (e.g. sensor temperature). These can be read if they are set accordingly in the field device configuration. Using the HART variables. it is possible to apply the set measured values directly from the field device into the I/O area of your automation system.

Regardless of the number of configured channels, a maximum of 8 HART variables can be assigned for HART modules and no more than 4 HART variables per channel. You assign the HART variables to a channel in the properties for the module ("HART variable settings" area).

Requirement

The HART module is plugged into an ET 200 M (as of IM 153-2, 6ES7 153-2BA02-0AB0).

Address assignment

The HART module occupies 16 input/output bytes. If you configure HART variables, the module occupies an additional 5 bytes for each HART variable.

If you use all 8 HART variables, the HART input module occupies a total of 56 input/output bytes (16 bytes + 8 x 5 bytes = 56 bytes).

The "None" configuration occupies no additional input bytes.

Configuration of HART variables

You can configure up to 4 HART variables for a channel

- PV (Primary Variable)
- SV (Secondary Variable)
- TV (Tertiary Variable)
- QV (Quaternary)

CiR is a placeholder that reserves the address space for a HART variable. You must configure the HART variables you are not using with the "None" parameter.

Configuration of HART variables

The HART variables are structured as follows:

4 bytes of HART data	1 byte QC

Structure of the "Quality-Code" byte

The Quality-Code (QC) can assume the following values:

Quality-Code (QC)	Meaning
0x4C or 0	Initialization: 0 value of IM and 4C of module
0x18	Communication cancelled / no communication
0x0C	Fault in HART device
0x47	HART device is busy
0x84	OK "Configuration changed"
0x80	OK

See also

Documentation for HART analog modules (http://support.automation.siemens.com/WW/view/en/22063748)

Signal modules for process automation

Fundamentals

Introduction

Signal modules for the process automation are S7-300 models, such as SM 321; DI 16xNAMUR or SM 322; DO 16x24VDC/0.5A.

They are being operated in a DP slave (IM 153-2).

Unlike standard modules, they offer the following additional technical functions, such as pulse extension and chatter monitoring.

See also

Changeover contact (Page 2063)

Technological parameters (Page 2064)

Changeover contact

"Changeover contact" sensor type

If the digital inputs of a channel group are configured as "changeover contacts", the module runs diagnostics for the changeover contact sensor type for this channel group.

Changeover contact

A changeover contact is an auxiliary switch with only one moving switch element with one close setting each for closed and open switching device.

Remember the following rule:

- Always connect a normally open contact to the "even" channel
- Always connect a normally closed contact to the "odd" channel.

The tolerated switchover time between the two channels is fixed at 300 ms.

If the result of the check is negative, then

- the module identifies the value status of the normally open channel as "invalid"
- the module generates a diagnostic entry for the normally open channel
- triggers a diagnostic interrupt (if diagnostic interrupts have been enabled)

The digital input signal and the value status are updated only for the normally open channel. For the normally closed channel, the digital input signal is set permanently to "zero" and the value status to "invalid" since this channel is used only to check the sensor.

Diagnostics depends on the "Selection" parameter (of the sensor). You should also note the special features of diagnostics with the changeover contact sensor type in the "Signal Modules for Process Automation" manual.

See also

Documentation on modules for process automation (http://support.automation.siemens.com/WW/view/de/7215812/0/en)

Technological parameters

Pulse extension and flutter monitoring

Pulse extension is a function for changing a digital input signal. A pulse at a digital input is extended to at least the length set in the parameters. If the input pulse is already longer than the specified length, it is not changed.

If you want the pulse to be extended, click in the box to select the time. If you do not want the pulses to be extended, select the "---" entry.

Flutter monitoring is a process control function for digital input signals. It detects and reports signal changes that are unexpected in process control, for example when an input signal fluctuates too often between "0" and "1".

Flutter monitoring is possible only when group diagnostics has also been enabled for this input.

Monitoring window and number of signal changes

Flutter monitoring works with aid of the two parameters Monitoring window and Number of signal changes.

The first time the signal changes, the time set as the monitoring window is started. If the signal changes more often during this time than allowed by the number of signal changes parameter, this is signaled as a flutter error. If no flutter error is detected during the monitoring window time, the monitoring window can be restarted at the next signal change.

Note

If you set pulse extension for an input channel, this also affects the flutter monitoring enabled for this channel. The "extended pulse" signal is the input signal for the flutter monitoring. You should therefore make sure that the values set for pulse extension and flutter monitoring are compatible with each other.

See also

Documentation on modules for process automation (http://support.automation.siemens.com/WW/view/de/7215812/0/en)

IQ Sense module

Properties of 8 IQ-SENSE

Properties

The 8 IQ-SENSE module has the following properties:

- Connection of sensors with IQ-SENSE®, photoelectric proximity switches: for example, reflex sensors, diffuse sensors, and laser sensors.
- It can be used centrally in an S7-300 or distributed in an ET 200M.
- You can connect up to 8 sensors to every module. Each sensor requires a two-wire cable.
- Function reserve that can be assigned parameters.
- Time functions, switching hysteresis, synchronous mode that can be assigned parameters
- Sensitivity and distance values can be specified (*IntelliTeach* using the "IQ-SENSE Opto" FB)
- Teach-in
- Sensors can be removed and inserted during operation (automatic reassignment of parameters)

Anti-interference group

Only for optical IQ Sense devices (IQ profile ID 1).

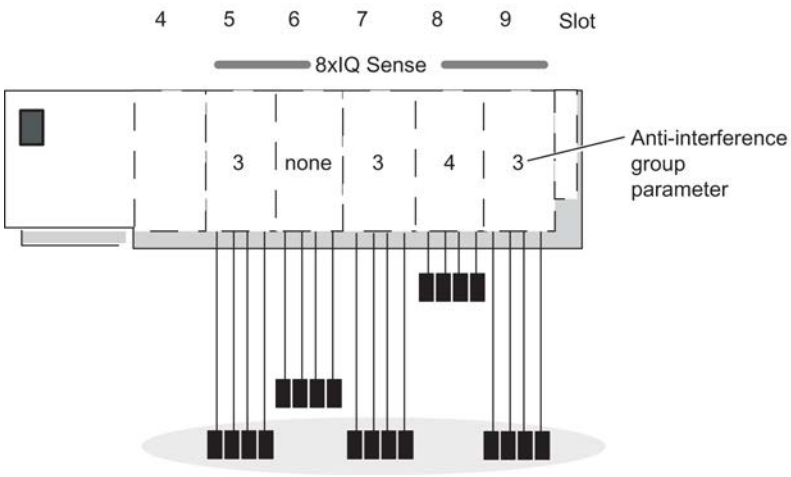
For IQ Sense devices with IQ profile ID 128 (ultrasound), see "Multiplex/synchronous mode" under the channel-specific parameters.

Prevention of interference (e.g., scattered light) by assigning an anti-interference group. This means:

- Anti-interference group: None (= default)
  Optical sensors on one or more modules can mutually influence each other when unfavorably arranged.
- Anti-interference group: 3 or 4
  Optical sensors on the same module with anti-interference group 3 or 4 cannot mutually influence each other. Similarly, optical sensors on different modules with anti-interference group 3 or 4 cannot mutually influence each other. You need not maintain minimum clearance between the IQ Sense devices and can, for example, align two retroreflective sensors on a single reflector.

Operating principle

The diagram below explains the functioning of the anti-interference group parameter:



Mutual interference is only possible between the optical sensors of the modules in slot 5, 6, 7 and 9 because they are in the same anti-interference group 3 or "None" is set.

Note

Sensors in the same anti-interference group must be installed to maintain the minimum clearance (see sensor package insert) and to prevent mutual interference.

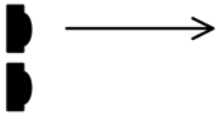
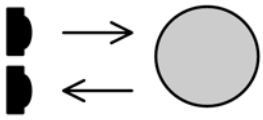
Encoder type

This parameter is used to set the sensor type per channel:

- Reflex sensor or
- Diffuse sensor or
- Disabled

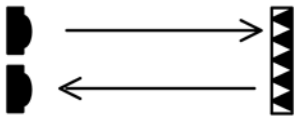
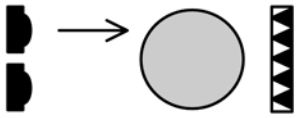
Diffuse sensor

Table 8-122 Diffuse sensor

Diffuse sensor	Object	
Transmitter Receiver		Circuit state 0: No object detected, which means the object is not in the beam. The receiver does not see any light.
Transmitter Receiver		Circuit state 1: Object detected, which means the object is in the beam. The receiver does not see any light.

Reflex sensor

Table 8-123 Reflex sensor

Reflex sensor	Object	
Transmitter Receiver		Circuit state 0: No object detected, which means the object is not in the beam. The receiver sees light.
Transmitter Receiver		Circuit state 1: Object detected, which means the object is in the beam. The receiver does not see any light.

Switching hysteresis

Faults with the diffuse sensor or in the production process can result in signal wobbles. The measured value then changes the switching threshold by 100 % (object detected - object not detected). You can prevent this switching threshold wobble using the switching hysteresis parameter. This will ensure a stable output signal on the sensor.

You can assigned parameters to 5 %/10 %/20 %/50 % for switching hysteresis.

Requirements

You can only set the switching hysteresis parameter for diffuse sensors with background fadeout.

Operating principle

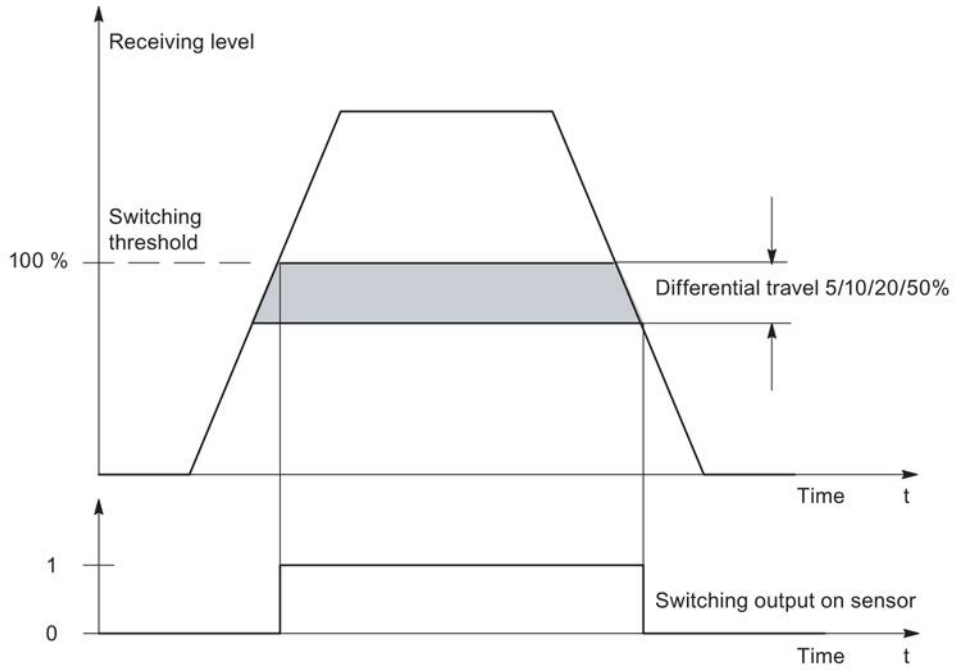


Figure 8-59 Switching hysteresis parameter

Time function,time value

These parameters can be used to set the electronic module for its specific application.

Operating principle

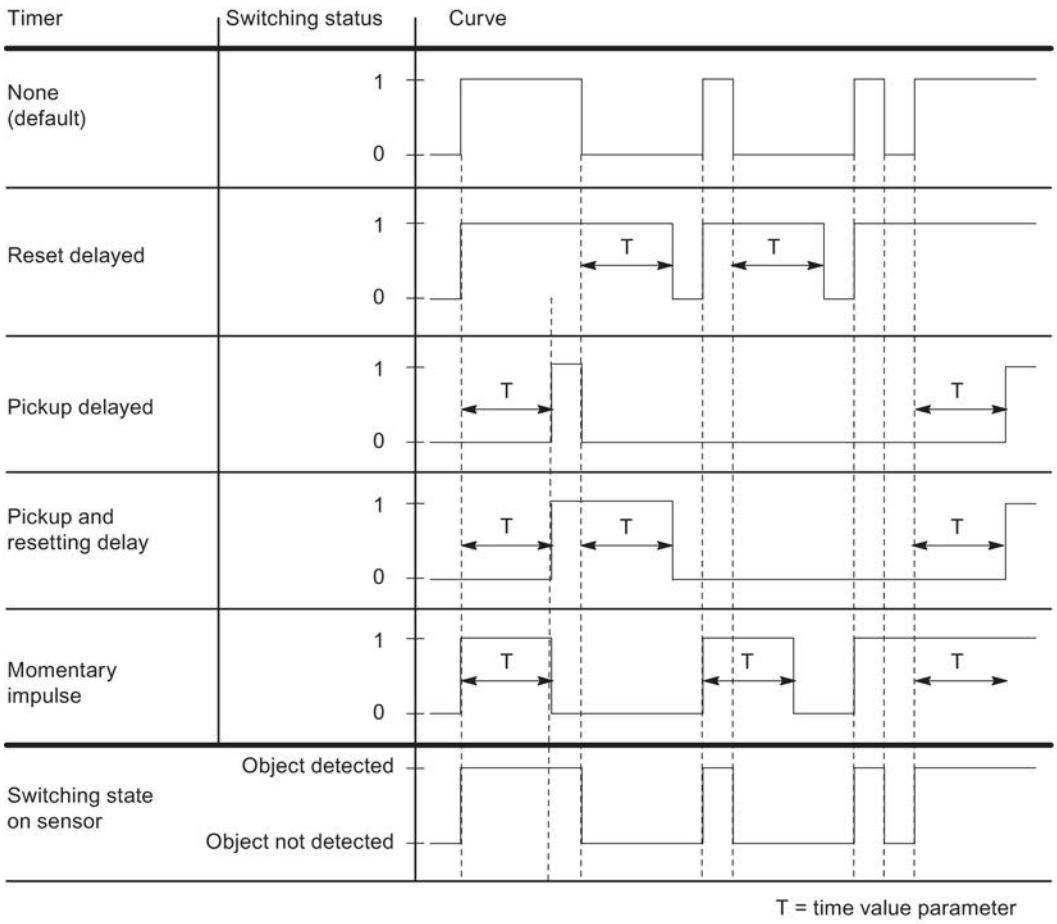


Figure 8-60 Time functions, time values parameters

Multiplex/synchronous mode

For the prevention of mutual influence between IQ Sense ultrasound devices in spatial proximity (devices with IQ profile ID 128), use the "Multiplex/synchronous operation" parameter.

Settings for the multiplex/synchronous mode parameter

Disabled: Mutual influence between IQ Sense ultrasound sensors in spatial proximity is possible (default). The cycle time is determined by the IQ Sense ultrasound sensor.

Multiplex: The IQ Sense ultrasound sensors determine the process value (distance) one after another, preventing them from affecting one another. The cycle time here is the sum of the configured synchronous cycle times of the IQ Sense ultrasound sensors that are to be multiplexed.

Synchronization: The IQ Sense ultrasound sensors determine the process value (distance) at exactly the same time, preventing them from affecting one another. The cycle time here

corresponds to the greatest configured synchronous cycle time from among the IQ Sense ultrasound sensors that are to be synchronized.

You can, for example, use synchronous operation for a curtain function in which several IQ Sense ultrasound sensors aligned in parallel share a single extended detection area. The sensors simultaneously emit an ultrasound impulse. When an object enters the detection area, the sensor nearest to the object receives the echo most quickly. The object can therefore not only be detected, it can be located as well.

AFI value

Using the AFI value (application series identifier, as defined in the ISO 15693-3 international standard), transponders can be selected for different applications. Only transponders whose AFI value coincides with the value set on the sensor are processed. If a transponder has the AFI value "0", it can be identified and processed regardless of the AFI value of the sensor.

This parameter is only important if it is supported by the ident system, otherwise any value (normally "0") may be assigned.

Transponder type

Depending on the type of the transponder, you must configure whether it is an ISO transponder or a vendor-specific type.

For transponders in accordance with international standard ISO 15693, the value "1" should be selected; for all other types "0" is set. Based on this setting, one of the two possible air interface drivers is selected in the sensor.

This parameter is only important if it is supported by the ID system, otherwise any value (normally "0") may be assigned.

ET 200S

Configuring an ET 200S

Introduction

For the ET 200S series, you can find a wide range of modules in the hardware catalog under "Distributed I/O".

Assigning parameters

For information on configuration and parameter assignment, refer to "See also".

See also

Configuring an ET 200S (Page 1356)

Frequency converters

Use of the frequency converter

Frequency converters

The frequency converter ICU24 and ICU24F (as fail-safe version) are modular design frequency converters that are completely embedded in the distributed I/O system ET 200S. For parameterization of both modules, please see the following.

Message frame

The message frame number and the operating mode of the module are only displayed and cannot be modified.

Application ID

You indicate the saved parameters in the frequency converter as a whole with the application ID. Enter an application ID from the value range 0 to 65535. During startup (or pull/plug), this ID is compared with the application ID stored on the converter.

Converters that work with identical applications are usually also identically parameterized and should be identified with the same application ID. Converters with the same application ID may be exchanged between each other. Copying of the complete parameterization of a converter to another converter, for example, via an MMC, is only accepted, if both have the same application ID.

Converters that work with different applications and are parameterized differently must be identified by different application IDs. This prevents a converter with unsuitable parameterization from starting on an incorrect slot, i.e. on the wrong application. This also prevents the parameterization that is saved in the converter from being accidently overwritten with any parameteriation that is stored on an MMC.

Enable diagnostic interrupt

You can enable the diagnostic interrupt for the frequency converter. If diagnostic interrupt is enabled, an OB 82 must be available in a CPU to process the diagnostic events.

See also

Documentation for the frequency converter (http://support.automation.siemens.com/WW/view/en/26291825/0/en)

ET 200pro

Use of the frequency converter

Frequency converters

The frequency converters ET 200pro FC and ET 200pro F-FC (as fail-safe version) are modularly design frequency converters that are completely embedded in the distributed I/O system ET 200pro. The following section describes how to configure the two modules.

Message frame

The message frame number and the operating mode of the module are only displayed and cannot be modified.

Application ID

You indicate the saved parameters in the frequency converter as a whole with the application ID. Enter an application ID from the value range 0 to 65535. During startup (or pull/plug), this ID is compared with the application ID stored on the converter.

Converters that work with identical applications are usually also identically configured and should be identified with the same application ID. Converters with the same application ID may be exchanged between each other. Copying of the complete configuration of a converter to another converter, for example, via an MMC, is only applied, if both have the same application ID.

Converters that work with different applications and are configured differently must be identified by different application IDs. This prevents a converter with unsuitable configuration from starting on an incorrect slot, in other words on the wrong application. This also prevents the configuration that is saved in the converter from being accidently overwritten with any configuration that is stored on an MMC.

Enable diagnostic interrupt

You can enable the diagnostic interrupt for the frequency converter. If diagnostic interrupt is enabled, an OB 82 must be available in a CPU to process the diagnostic events.

8.1.5.7 IPv6 configuration

IPv6 protocol

The Internet Protocol version 6 - called IPv6 below - extends the Internet Protocol version 4 (IPv4) that is used predominantly at the current time.

Address format IPv6: Notation

IPv6 addresses consist of 8 fields each with four-character hexadecimal numbers (128 bits in total). The fields are separated by a colon.

Example:

fd00:ffff:ffff:ffff:ffff:ffff:2f33:8f21

Rules / simplifications:

- Leading zeros within a field can be omitted.
  Example: fd01:0:ffff::2d12:7d23
- If one or more fields have the value 0, a shortened notation is possible.
  The address fd00:0:0:0:0:0:0:8f21 can also be shortened and written as follows:
  fd00::8f21
  To ensure uniqueness, this shortened form can only be used once within the entire address.
- Decimal notation with periods
  The last 2 fields or 4 bytes can be written in the normal decimal notation with periods.
  Example: The IPv6 address fd00::ffff.125.1.0.1 is equivalent to fd00::ffff:7d01:1

Entry and appearance

The entry of IPv6 addresses is possible in the notations described above. IPv6 addresses are always shown in the same notation in which they were entered.

See also

IPv6 with the CP 1543-1 (Page 2073)

IPv6 with the CP 1543-1

Use of IPv6 with the CP 1543-1

The CP supports the Internet Protocol version 4 (IPv4) for all IP services

The additional specification of addresses in IPv6 format can be used with the CP for the following services and applications:

- FETCH/WRITE
  Direct write/read access by PC stations, SIMATIC S5 or third-party devices
- FTP client
  FTP access from the S7-1500 CPU to an FTP server with the program block FTP_CMD
- FTP server
  FTP access from an FTP client to data areas of the S7-1500 CPU
- SNMP
  Data query using MIB objects according to SNMP
- E-mail
  Data transfer from the S7-1500 CPU using the program block T_Mail

8.1.6 Displaying alarms

8.1.6.1 Overview of the alarm display

The "Alarm display" function can be used to output asynchronous alarms of diagnostics events and user-defined diagnostics alarms as well a alarms from ALARM instructions.

From the alarm display, you can also start the alarm editor with the "Edit alarm" shortcut menu command and then create user diagnostic alarms.

Icons

The following table shows the icons and their functions:

Icon	Function
Archive view	Shows the alarms located in the archive.
Active alarms	Shows the currently active (pending) alarms. Alarms that must be acknowledged are shown in blue lettering.
Ignore	Ignores the arrival of alarms, These alarms are neither shown in the window nor stored in the archive.
Acknowledge	Confirms the selected alarm as read. Alarms requiring acknowledgment are shown in blue lettering.
Clear archive	Deletes all alarms in the archive.
Export archive	Exports the current alarm archive to a file in xml format.

8.1.6.2 Archive view

In the archive view, alarms are displayed and archived according to the time they appear. You can set the size of the archive (between 200 and 3000 alarms) with the menu command "Options > Settings > Online & Diagnostics". If the selected archive size is exceeded, the oldest alarm it contains is deleted.

Alarms that must be acknowledged are displayed in blue lettering and can be acknowledged with the shortcut menu command "Acknowledge alarm(s)".

The archive is constantly updated and does not need to be saved explicitly.

8.1.6.3 Layout of the alarms in the archive view

In the archive view, all events occurring on the selected CPUs are logged. A new entry is created for each individual event and shown as a further row in the table.

Table structure

All attributes of the alarms can be shown as columns. You can show or hide individual columns as well as modify the width and order of the columns. These settings are saved when the project is closed.

The alarms can be displayed in one or more rows. In the single row display, only the first row of the multiple-row alarm data is displayed.

The alarms either require acknowledgment or do not require acknowledgment. The alarms requiring acknowledgment that have not yet been acknowledged are highlighted in blue lettering and can be acknowledged either with the button in the toolbar for the particular context or with the shortcut menu command "Acknowledge alarm(s)".

8.1.6.4 Receiving alarms

To allow alarms to be displayed, you must first set the receipt of alarms for each CPU.

Procedure

Toe receive alarms, follow these steps:

1. Double-click on the "Online & Diagnostics" folder of the relevant CPU in project navigation.
2. Click the "Online access" group in the area navigation.
3. Select the option "Receive alarms".

Note

If you select this procedure, alarms are only received after you have re-established an online connection to the device.

Or:

1. Select the relevant CPU in the device, network, or topology view.
2. Select the command "Receive alarms" in the "Online" menu or in the shortcut menu.

Or:

1. Select the CPU in project navigation.
2. Select the command "Receive alarms" in the "Online" menu or in the shortcut menu.

Note

If you select one of the two above-named procedures, you must have first established an online connection to the device.

8.1.6.5 Export archive

To archive alarms, you can export the archive. Follow these steps:

1. Go to the archive view.
2. Click the "Export archive" button.
3. In the dialog that opens, select the path to export the archive.

Result

The archive is saved as an xml file at the location you selected.

8.1.6.6 Clear archive

The archive is organized as a ring buffer, in other words, when it is full, the oldest alarms are deleted from the archive. With the "Clear archive" button, you can delete the entire archive.

Procedure

To clear the archive, follow these steps:

1. Click the "Clear archive" button in the toolbar of the alarm display.

8.1.6.7 "Active alarms" view

The "Active alarms" view is an image of the alarm acknowledgement memory of the selected CPU(s).

8.1.6.8 Layout of the alarms in the "Active alarms" view

The "Active alarms" view represents an image of the alarm acknowledgment memory of the selected CPUs. One entry is shown in the table per active alarm. Events of an alarm ("incoming", "outgoing" and "acknowledged") are displayed in one row.

Table structure

All attributes of the alarms can be shown as columns. You can show or hide individual columns as well as modify the width and order of the columns. These settings are saved when the project is closed.

The alarms can be displayed in one or more rows. In the single row display, only the first row of the multiple-row alarm data is displayed.

The alarms either require acknowledgment or do not require acknowledgment. The alarms requiring acknowledgment that have not yet been acknowledged are highlighted in bold print and can be acknowledged either with the button in the toolbar for the specific context or with the shortcut menu command "Acknowledge alarm(s)".

8.1.6.9 Status of the alarms

Depending on whether you are in the "Active alarms" view or the archive view, the displayed alarms may have a different status.

Status of the alarms in the "Active alarms" view

- I: Alarm came
- IA: Alarm came and was acknowledged
- IO: Alarm has gone

If more signal changes occur than can be sent (signal overflow), OV is displayed as the status and the status is shown in red.

Status of the alarms in the archive view

- No information: only with alarms generated by the PG/PC and displayed in the "Archive" tab, for example logon status, connection abort, mode changes
- I: Alarm came
- A: Alarm came and was acknowledged
- O Alarm has gone
- D: The alarm was deleted.

If more signal changes occur than can be sent (signal overflow), OV is displayed as the status and the status is shown in red.

8.1.6.10 Acknowledging alarms

Alarms that must be acknowledged are shown in blue lettering.

Procedure

To acknowledge an alarm, follow these steps:

1. Select the required alarm or alarms from the table.
2. Click the "Acknowledge" button.

Note

You can select more than one alarm to acknowledge at the same time. To do this, hold down the <Ctrl> key and then select the alarms you want to acknowledge.

Result

The selected alarm was acknowledged and is then shown in normal characters.

Note

In the "Active alarms" view, acknowledged alarms that have already gone are no longer displayed.

8.1.6.11 Ignoring alarms

Ignoring alarms

To ignore alarms, follow these steps:

1. Click the "Ignore" button.
   The icon is shown on a gray background.

Result

From this point onwards, all alarms will be ignored. A message is created in the archive view indicating that the display of alarms and events is disabled.

Canceling the ignoring of alarms

To cancel the ignoring of alarms, follow these steps:

1. Click the "Ignore" button.
   The icon is shown on a white background.

Result

All alarms, in other words, even the alarms currently pending on the CPU while the "Ignore alarms" function was active, are displayed again from this point onwards. A message is created in the archive view indicating that the display of alarms and events is enabled again.

8.1.6.12 Keyboard commands in the alarm display

Alarm display

Function	Shortcut keys
Select all alarms	Ctrl+A
Acknowledge all selected alarms	Ctrl+Q

8.2 Device and network diagnostics

8.2.1 Hardware diagnostics

8.2.1.1 Overview of hardware diagnostics

Principal methods of hardware diagnostics

Principal methods of hardware diagnostics

Hardware diagnostics can be performed as follows:

- Using the Online and Diagnostics view
- Using the "Online Tools" task card
- Using the "Diagnostics > Device Info" area of the Inspector window
- Using diagnostics icons, for example, in the device view and the project tree

Structure of the Online and Diagnostics view

The Online and Diagnostics view consists of two windows alongside each other:

- The left window shows a tree structure with folders and - when you open the folder - groups.
- The right window contains detailed information on the selected folder or selected group.

The "Online access" group and the "Diagnostics" and "Functions" folders are located here:

- "Online access" group: Displays whether or not there is currently an online connection with the associated target. In addition, you can establish or disconnect the online connection.
- "Diagnostics": Contains several diagnostics groups for the selected module.
- "Functions": Contains several groups, in which you can make settings for the selected module or issue commands to the module.

Function and structure of the "Online Tools" task card

For modules with their own operating mode (such as CPUs), the "Online tools" task card allows you to read current diagnostics information and commands to the module.

If you selected a module without its own operating mode or if you selected several modules before activation of the "Online Tools" task card, the task card relates to the relevant CPU.

The "Online Tools" task card consists of the following panes:

- CPU control panel
- Cycle time
- Memory

Note

A pane is filled with content only if the module controls the associated functions and an online connection exists.

If there is no online connection to the respective module, the display "No online connection" appears in blue. If an existing online connection was disconnected, then "This target is not available" will be displayed.

Structure or the "Diagnostics" tab of the Inspector window

The "Diagnostics" tab of the Inspector window itself consists of several tabs: Of these tabs, the following is relevant for the hardware diagnostics.

- Device information
  This tab relates to all CPUs of the project to which the online connection is established. Alarms are reported here if one or more CPUs are defective or are not in RUN mode.

See also

Basics on task cards (Page 295)

Inspector window (Page 293)

Determination of which of the devices that are connected online are defective

Overview of the defective devices

In the "Diagnostics > Device Info" area of the Inspector window you will obtain an overview of the defective devices that are or were connected online.

The "Diagnostics> Device Info" area of the Inspector window consists of the following elements:

- Header line with the number of defective devices
- Table with detailed information on each defective device

If you originate the establishment of an online connection to a device which is not reachable or reports one or more faults or is not in RUN mode, it will rank as defective.

Structure of the table with detailed information on the defective devices

The table consists of the following columns:

- Online status: Contains the online status as a diagnostic symbol and in words
- Operating mode: Contains the operating mode as a symbol and in words
- Device / module: Name of the affected device or the affected module

- Message: explains the entry of the previous column
- Details: The link opens the online and diagnostics view for the device, and places it in the foreground. If an online connection does not exist any longer, the link will open the connection establishment dialog.
- Help: The link supplies further information on the defect that has occurred.

See also

Displaying diagnostics status and comparison status using icons (Page 2081)

Displaying diagnostics status and comparison status using icons

Determining diagnostics status online and displaying using icons

When the online connection to a device is established, the diagnostics status of the device and, if applicable, its lower-level components is determined. The operating state of the device is also determined, where applicable.

The following is a description of which icons are displayed in specific views.

- Device view
  - The associated diagnostic icon is displayed for every hardware component (except the signal board on the CPU).
  - For a hardware component with lower-level components, if there is a hardware error in at least one lower-level component, the diagnostic icon appears as follows: The hardware component's diagnostic icon has a pale appearance and the diagnostic icon "Hardware error in lower-level component" is also shown in the lower right corner.
  - For hardware components with their own operating state, the operating state icon is also displayed to the left of or above the diagnostic icon.
  - The following applies to modules or submodules of a shared device with an S7-1500 CPU: No diagnostic icons are displayed (due to the configuration as GSDML device).
- Device overview
  - The associated diagnostic icon is displayed for every hardware component.
  - For a hardware component with lower-level components, if there is a hardware error in at least one lower-level component, the diagnostic icon appears as follows: The hardware component's diagnostic icon has a pale appearance and the diagnostic icon "Hardware error in lower-level component" is also shown in the lower right corner.
  - The following applies to modules or submodules of a shared device with an S7-1500 CPU: For modules assigned to the CPU, the associated diagnostic icon is displayed (modules that are not assigned do not receive a diagnostic icon). The associated diagnostic icon is displayed for plug-in submodules of an assigned module (submodules that are not pluggable are not visible and therefore do not receive a diagnostic icon).
- Network view
  - The associated diagnostic icon is displayed for every device.
  - For a hardware component with lower-level components, if there is a hardware error in at least one lower-level component, the diagnostic icon appears as follows: The hardware component's diagnostic icon has a pale appearance and the diagnostic icon "Hardware error in lower-level component" is also shown in the lower right corner.
  - The following applies to modules or submodules of a shared device with an S7-1500 CPU: A diagnostic icon is displayed. It belongs to that part of the station that is assigned to the CPU.

- Network overview
  - The associated diagnostic icon is displayed for every hardware component.
  - For a hardware component with lower-level components, if there is a hardware error in at least one lower-level component, the diagnostic icon appears as follows: The hardware component's diagnostic icon has a pale appearance and the diagnostic icon "Hardware error in lower-level component" is also shown in the lower right corner.
  - The following applies to modules or submodules of a shared device with an S7-1500 CPU: A diagnostic icon is displayed. It belongs to that part of the station that is assigned to the CPU.
- Topology view
  - The associated diagnostic icon is displayed for every device.
  - For a hardware component with lower-level components, if there is a hardware error in at least one lower-level component, the diagnostic icon appears as follows: The hardware component's diagnostic icon has a pale appearance and the diagnostic icon "Hardware error in lower-level component" is also shown in the lower right corner.
  - The associated diagnostic icon is displayed for every port. The meaning of the individual colors is described further below.
  - Each cable between two online ports is assigned the color associated with its diagnostics status.
    The color of the cable between two ports depends on the status of the individual ports:

Color of the first port	Color of the second port	Color of the connecting cable
green	green	green
green	gray	gray
green	red	red
gray	red	red

  - The following applies to modules or submodules of a shared device with an S7-1500 CPU: A diagnostic icon is displayed. It belongs to that part of the station that is assigned to the CPU.
- Topological overview
  - The associated diagnostic icon is displayed for every hardware component.
  - For a hardware component with lower-level components, if there is a hardware error in at least one lower-level component, the diagnostic icon appears as follows: The hardware component's diagnostic icon has a pale appearance and the diagnostic icon "Hardware error in lower-level component" is also shown in the lower right corner.
  - The following applies to modules or submodules of a shared device with an S7-1500 CPU: A diagnostic icon is displayed. It belongs to that part of the station that is assigned to the CPU.

- Project tree
  - The associated diagnostic icon is displayed behind every hardware component.
  - For a hardware component with lower-level components (e.g., distributed I/O, Slave_1), if there is a hardware error in at least one lower-level component, the diagnostic icon appears as follows: The hardware component's diagnostic icon has a pale appearance and the diagnostic icon "Hardware error in lower-level component" is also shown in the lower right corner.
  - For hardware components with their own operating state, the operating state icon is also displayed in the top right corner of the diagnostic icon.
  - If forcing is active on a CPU, a red F is displayed at the left margin of the diagnostic icon.
  - The diagnostic icon "Hardware error in lower-level component" is displayed behind the "Local modules" folder when there is a hardware error in at least one of the associated modules.
  - The diagnostic icon "Hardware error in lower-level component" is displayed behind the "Distributed I/O" folder when there is a hardware error in at least one of the associated modules.
  - The diagnostic icon "Hardware error in lower-level component" is displayed behind the project folder when the "Hardware error in lower-level component" diagnostic icon is displayed behind at least one of the "Local modules" or "Distributed I/O" folders.
  - The following applies to modules or submodules of a shared device with an S7-1500 CPU: The associated diagnostic icon is displayed for modules assigned to the CPU (modules that are not assigned are grayed out and do not receive a diagnostic icon). The associated diagnostic icon is displayed for plug-in submodules of an assigned module (submodules that are not pluggable are not visible and therefore do not receive a diagnostic icon).

Note

If the diagnostic for a hardware component is "not reachable from the CPU", the diagnostic icon "Hardware error in lower-level component" is not additionally shown.

Diagnostic icons for modules and devices

The following table shows the available icons and their respective meaning.

Icon	Meaning
	The connection with a CPU is currently being established.
	The CPU is not reachable at the set address.
	The configured CPU and the CPU actually present are of incompatible types.
	On establishment of the online connection to a protected CPU, the password dialog was terminated without specification of the correct password.
	No fault

Icon	Meaning
	Maintenance required
	Maintenance demanded
	Error
	The module or device is deactivated.
	The module or the device cannot be reached from the CPU (valid for modules and devices below a CPU).
	Diagnostics data are not available because the current online configuration data differ from the offline configuration data.
	The configured module or device and the module or device actually present are incompatible (valid for modules or devices under a CPU).
	The configured module does not support display of the diagnostics status (valid for modules under a CPU).
	The connection is established, but the module status has not yet been determined or is unknown.
	The configured module does not support display of the diagnostics status.
	Hardware error in lower-level component: A hardware error is present in at least one lower-level hardware component. (occurs as a separate icon only in the project tree)

Note

Some modules, for example, the FM 450-1, are only indicated as having a problem in the case of an error if you have enabled the diagnostic interrupt when assigning the module property parameters.

Icons for the comparison status

The diagnostic icons can be combined at the bottom right with additional smaller icons that indicate the result of the online/offline comparison. The following table shows the available comparison icons and their meaning.

Icon	Meaning
	Hardware error in lower-level component: The online and offline versions differ (only in the project tree) in at least one lower-level hardware component.
	Software error in lower-level component: The online and offline versions differ (only in the project tree) in at least one lower-level software component.
	Online and offline versions of the object are different
	Object only exists online
	Object only exists offline
	Online and offline versions of the object are the same

Note

If both a comparison icon and the "Error in lower-level component" diagnostic icon are to be displayed at the bottom right in the device view, the following rule applies: The diagnostic icon for the lower-level hardware component has a higher priority than the comparison icon. This means that a comparison icon is only displayed if the lower-level hardware components have no errors.

Display of software errors in the project tree

- The associated comparison icon is shown behind each block.
- Behind each folder, under which exclusively blocks are contained, the diagnostic icon "Software error in lower-level component" is displayed when there is a software error in at least one of the associated blocks.
- For a hardware component with lower-level software components, if there is no hardware error and there is an error in at least one lower-level software component, the diagnostic icon appears as follows: The hardware component's diagnostic icon has a pale appearance and the diagnostic icon "Software error in lower-level component" is also shown in the lower right corner.

Combined diagnostics and comparison icons

The following table shows examples of icons that are displayed in the diagnostics icon.

Icon	Meaning
	Folder contains objects whose online and offline versions differ (only in the project tree)
	Object only exists online

Operating state icons for CPUs and CPs

The following table shows the available icons and their respective operating states.

Icon	Operating state
	RUN
	STOP
	STARTUP
	HOLD
	DEFECTIVE

Icon	Operating state
	Unknown operating state
	The configured module does not support display of the operating state.

Note

If forcing is active on a CPU, a red F is displayed on a pink background at the bottom right of the operating state icon.

Color marking of ports and Ethernet cables

The following table shows the available colors and their respective meaning.

Color	Meaning
	No fault or maintenance required
	Maintenance demanded
	Communication error or topological error
	no diagnostic capability

Start online and diagnostics view

Overview of possible ways of starting the Online and Diagnostics view

You can start the Online and Diagnostics view of a module to be diagnosed at the following locations:

- Overview
- Project tree
- Device view
- Device overview
- Network view
- Network overview
- Topology view

In the following, examples are used to show how to proceed.

Requirement

The project with the module to be diagnosed is open.

Note

This requirement does not apply if you call the online and diagnostics view from the project tree after you have identified the accessible devices.

Procedure

To start the online and diagnostics view of a module, follow these steps:

1. In the project tree, open the respective device folder.
2. Double click on "Online & Diagnostics".

Or:

1. In the project tree, select the respective device folder.
2. Select the "Online & Diagnostics" command in the shortcut menu or the "Online" main menu.

Or:

1. In the project tree, open the "Online access" folder.
2. Open the folder for the interface with which you want to establish the online connection.
3. Double click on "Show/Update accessible devices".
4. Select the module to be diagnosed.
5. Select the "Online & Diagnostics" command in the shortcut menu or the "Online" main menu.

Or:

1. In the project tree, open the "Local modules" folder.
2. Select the respective device or the module that is to be diagnosed.
3. Select the "Online & Diagnostics" command in the shortcut menu or the main menu.

Or:

1. Open the device view in the device configuration.
2. Select the module to be diagnosed.
3. Select the "Online & Diagnostics" command in the shortcut menu or the "Online" main menu.

Or:

1. Open the device view in the device configuration.
2. Establish an online connection to the module to be diagnosed.
3. Double-click on the diagnostics icon above the module.

Or:

1. Open the network view in the device configuration.
2. Select the station with the module to be diagnosed.
3. Select the "Online & Diagnostics" command in the shortcut menu or the "Online" main menu.

Or:

Result

The online and diagnostics view of the module to be diagnosed will be started. If an online connection to the associated CPU had previously been created, the header bar of the Online and Diagnostics view will now have an orange background.

Note

If no online connection exists when the online and diagnostics view is started, no online information is displayed and the display fields remain empty.

Activation of the "Online Tools" task card

Activation of the "Online Tools" task card

You can activate this task card as follows:

1. Start the online and diagnostics view.
2. Click on the "Online Tools" task card.

Or:

1. Start the device view.
2. Click on the "Online Tools" task card.

Or:

1. Start the network view.
2. Click on the "Online Tools" task card.

8.2.1.2 Showing non-editable and current values of configurable module properties

Showing general properties and system-relevant information for a module

Where do I find the information I need?

The general properties and system-relevant information for a module can be found in the "General" group in the "Diagnostics" folder in the online and diagnostics view of the module to be diagnosed.

Structure of the "General" group

The "General" group consists of the following areas:

- Module
- Module information
- Vendor information

"Module" area

This area shows the following data of the module:

- Short designation, for example, CPU 1214C DC/DC/DC
- Order no.
- Hardware
- Firmware
- Racks
- Slot

"Module information" area

This area shows the following data of the module that you configured during hardware configuration:

- Module name
- Installation date (not displayed for all modules)
- Additional information (not displayed for all modules)

"Manufacturer information" area

This area shows the following data of the module:

- Manufacturer
- Serial number
- Profile: Profile ID as hexadecimal number

  Note

  You will find the corresponding profile name in the profile ID table for PROFIBUS International (see "www.profibus.com").

- Profile details: Profile-specific type as hexadecimal number

  Note

  You will find the corresponding profile-specific type name in the profile-specific type table for PROFIBUS International (see "www.profibus.com").

Display configured cycle times

Where do I find the information I need?

The required information can be found in the following places:

- In the "Cycle time" group of the "Diagnostics" folder in the Online and Diagnostics view of the module to be diagnosed.
- In the "Cycle time" pane of the "Online Tools" task card

Structure of the "Cycle time" group in the "Diagnostics" folder of the Online and Diagnostics view

The "Cycle time" group consists of the following areas:

- Cycle time diagram (graphical display of the assigned and measured cycle times)
- Cycle time configured (display of the assigned cycle times as absolute values)
- Cycle times measured (display of the measured cycle times as absolute values)

Structure of the "Cycle time" pane of the "Online Tools" task card

The "Cycle time" pane displays the cycle time diagram and below it the measured cycle times as absolute values.

Assigned cycle times

The following assigned cycle times are displayed in the cycle time diagram and in the "Cycle time configured" area.

- Minimum cycle time
- Maximum cycle time

In the cycle time diagram, the minimum cycle time and the maximum cycle time correspond to the two markings on the time axis.

In the "Cycle time configured" area, the assigned cycle times are displayed as absolute values.

Show interfaces and interface properties of a module

Where do I find the information I need?

The interfaces and interface properties of a module can be found in the "Diagnostics" folder in the online and diagnostics view of the module to be diagnosed in the following group:

- PROFINET interface

"PROFINET interface" group

This group is divided into the following areas:

- "Ethernet address" with the "Network connection" and "IP Parameters" subareas
- "Ports"

"Network connection" subarea of the "Ethernet address" area

This subarea shows the following data for the module:

- MAC Address:
  MAC address of the interface.
  The MAC address consists of two parts. The first part ("Basic MAC address") identifies the manufacturer (Siemens, 3COM, ...). The second part of the MAC address differentiates between the various Ethernet devices. Each Ethernet module is assigned a unique MAC address.

"IP Parameters" subarea of the "Ethernet address" area

This subarea shows the following data for the module:

- IP address:
  Internet protocol address of the device on the bus (TCP/IP)
- Subnet mask:
  The subnet mask shows which part of the IP address determines the membership of a particular sub-network.
- Default router:
  If the subnet is connected via a router to other subnets, the IP address of the default router must be known. This is the only way a datagram can be forwarded with a non-matching subnet address.
- IP settings:
  Identifier for the path by which the device has obtained its IP settings (IP address, subnet mask, default router).

Identifier	Meaning
0	IP address is not initialized
1	By configuration (i.e., by the configuration loaded to the device from the device or network view)
2	Via the "Assign IP address" group of the online and diagnostics view

Identifier	Meaning
3	Via the DHCP server (i.e., the IP parameters are obtained by a special service from a DHCP server (Dynamic Host Configuration Protocol) and assigned for a limited time)
4	IP address is set by a user program
5	Source of IP address unknown

- IP setting time:
  Time stamp of the last change to the IP address directly through the Ethernet connection of the module

"Ports" area

This area shows the following data for the module:

- Ethernet ports
  Physical properties of the PROFINET interface

Properties of the PROFINET interface	Meaning
Port no.	Port number The short description of interface (X + interface no.) and port (P + port no.) is specified in parentheses. An "R" in the short description of a port means that it is a ring port.
Status	Displays the status of the port LINK LED. • Status "OK" means another device (such as a switch) is connected to the port and the physical connection is available. • Status "disconnected" means no other device is connected to the port. • Status "deactivated" means that access to the port is blocked.
Settings	• "Automatic" for automatic network settings of the device • Network settings for speed and transmission method for manual network settings of the device
Operating mode	Network settings for speed and transmission method

If you select a line in the port table, additional help information will be provided for the corresponding port.

Displaying IO controllers that access modules of a Shared Device

Where do I find the information I need?

The display of those IO controllers that access the modules of a Shared Device can be found in the Online and Diagnostics view of the interface module of the Shared Device in the "Diagnostics" folder in the following area of the "PROFINET interface" group:

- IO controller

See also

Useful information on Shared Devices (Page 1521)

Displaying sync domain properties of a PROFINET device

Where do I find the information I need?

The sync domain properties of a PROFINET device can be found in the following area of the "PROFINET interface" group in the "Diagnostics" folder of the Online and Diagnostics view of the device to be diagnosed:

- Domain

"Domain" area

This area is divided into the following subareas:

- Sync domain
- MRP domain

What is a sync domain?

A sync domain is a group of PROFINET devices that are synchronized to a common clock. Exactly one device has the role of the sync master (clock generator); all other devices assume the role of a sync slave. The sync master is usually an IO controller or a switch.

Non-synchronized PROFINET devices are not part of a sync domain.

"Sync domain" subarea of the "Domain" area

This subarea shows the following properties of the sync domain:

- Name:
  Name of sync domain
- Role:
  Role of the PROFINET device in the sync domain. The following roles are possible:
  - Sync master
  - Sync slave

- Synchronization interval:
  Interval at which the synchronization is performed
- Send clock
  Smallest possible send interval for the data exchange
- Jitter accuracy of the send clock
- Reserved bandwidth for cyclic communication

Displaying MRP domain properties of a PROFINET device

Where do I find the information I need?

The MRP domain properties of a PROFINET device can be found in the following area of the "PROFINET interface" group in the "Diagnostics" folder of the Online and Diagnostics view of the device to be diagnosed:

- Domain

"Domain" area

This area is divided into the following subareas:

- Sync domain
- MRP domain

What is an MRP domain?

The Media Redundancy Protocol (MRP) enables redundant networks to be structured. Redundant transmission paths (ring topology) ensure that, if one transmission path fails, an alternative communication path is available. The PROFINET devices that are part of this redundant network form an MRP domain.

"MRP domain" subarea of the "Domain" area

This subarea shows the following properties of the MRP domain:

- Name:
  Name of MRP domain
- Role:
  Role of the PROFINET device in the MRP domain. The following roles are possible:
  - Manager
  - Manager (Auto)
  - Client
  - Not a device of the ring
- Ring port 1:
  The port of the PROFINET device that has the "Ring port 1" property

- Ring port 2:
  The port of the PROFINET device that has the "Ring port 2" property
- Status of the MRP ring:
  Indicates whether the ring is interrupted ("open" status) or not ("closed" status).

Displaying current firmware of a module

Displaying firmware

You can display the currently installed firmware of a module.

Requirements

- The module supports a firmware update.
- The module is connected online.

Procedure

To display the current firmware, follow these steps:

1. Open the module in the Online and Diagnostics view.
2. Select the "Firmware update" group in the "Functions" folder.
3. You read off the current firmware in the "Online data" area under "Firmware".

8.2.1.3 Showing the current values of dynamic modules properties

Display measured cycle times

Where do I find the information I need?

The measured cycle times can be found at each of the following places:

- In the "Cycle time" group of the "Diagnostics" folder in the Online and Diagnostics view of the module to be diagnosed.
- In the "Cycle time" pane of the "Online Tools" task card

Structure of the "Cycle time" group in the "Diagnostics" folder of the Online and Diagnostics view

The "Cycle time" group consists of the following areas:

- Cycle time diagram (graphical display of the assigned and measured cycle times)
- Cycle time configured (display of the assigned cycle times as absolute values)
- Cycle times measured (display of the measured cycle times as absolute values)

Structure of the "Cycle time" pane of the "Online Tools" task card

The "Cycle time" pane displays the cycle time diagram and below it the measured cycle times as absolute values.

Graphical display of the measured cycle times

The following measured cycle times are displayed in the cycle time diagram:

- Shortest cycle time: Duration of the shortest cycle since the last transition from STOP to RUN
  This corresponds to the dashed gray arrow on the left in the diagram.
- Current / last cycle time: Duration of the last cycle
  This corresponds to the green arrow in the diagram. If the current / last cycle time exceeds the maximum cycle time, the arrow will turn red.

  Note

  If the duration of the last cycle comes close to the maximum cycle time, it may be possible that it will be exceeded. Depending on the CPU type, parameter assignment and your user program, the CPU can switch to STOP mode. If for instance you are monitoring the tags in your program, this will increase the cycle time.

  If the cycle lasts longer than double the maximum cycle time, and you do not restart the maximum cycle time in the user program (by calling the extended RE_TRIGR) instruction, the CPU will switch to STOP mode.

- Longest cycle time: Duration of the longest cycle since the last transition from STOP to RUN.
  This corresponds to the dashed blue arrow on the right in the diagram.

A blue band extends between the two dashed lines; this band corresponds to the entire range of the measured cycle times. If a measured cycle time is greater than the maximum cycle time, the portion of the band that lies outside the assigned limits will be colored red.

Display of the measured cycle times as absolute values

The following measured times are displayed in the "Cycle times measured" area and in the "Cycle time" pane.

- Shortest cycle time since the last transition from STOP to RUN.
- Current/last cycle time:
- Longest cycle time since the last transition from STOP to RUN.

Showing the current status of the LEDs of a CPU

Where do I find the information I need?

The current status of the LEDs of a CPU can be found in the display area of the "CPU control panel" pane of the "Online tools" task card.

Display area of the "CPU control panel" pane of the "Online Tools" task card

This area contains the following displays:

- Station name and CPU type (short designation)
- RUN / STOP (corresponds to the "RUN / STOP" LED of the CPU)
- ERROR (corresponds to the "ERROR" LED on the CPU)
- MAINT (corresponds to the "MAINT" LED on the CPU)

Showing fill levels of all types of memory on a CPU

Where do I find the information I need?

The fill levels of all types of memory on a CPU can be found on the following two pages:

- In the display area of the "Memory" group in the "Diagnostics" folder in the online and diagnostics view of the module to be diagnosed
- In the display area of the "Memory" pane on the "Online Tools" task card

Display area of the "Memory" group in the "Diagnostics" folder of the online and diagnostics view

This area contains the current memory utilization of the associated module and details of the individual memory areas.

The memory utilization is shown both as a bar diagram and as a numerical value (percentage).

The following memory utilizations are shown:

- Load memory
  If no memory card is inserted, the internal load memory is displayed.
  If a memory card is inserted, the operating system only uses the inserted load memory as the load memory. This is displayed here.
- Work memory
- Retentive memory

Display area of the "Memory" pane of the "Online Tools" task card

This area contains the current memory utilization of the associated module. The available memory is shown both as a bar diagram and as a numerical value (percentage). The numerical value is rounded to an integer value.

Note

If less than 1% of a memory area is utilized, the available portion of this memory area is shown as "99%".

The following memory utilizations are shown:

- Load memory
  If no memory card is inserted, the internal load memory is displayed.
  If a memory card is inserted, the operating system only uses the inserted load memory as the load memory. This is displayed here.
- Work memory
- Retentive memory

See also

Load memory (Page 1572)

Work memory (Page 1572)

Retentive memory areas (Page 1574)

Displaying fill level of all types of memory of an S7-1500 CPU

Where do I find the information I need?

The fill levels of all types of memory of an S7-1500 CPU can be found at the following two places:

- In the display area of the "Memory" group in the "Diagnostics" folder in the online and diagnostics view of the module to be diagnosed
- In the display area of the "Memory" pane on the "Online Tools" task card

Display area of the "Memory" group in the "Diagnostics" folder of the online and diagnostics view

This area contains the current memory utilization of the associated module and details of the individual memory areas.

The memory utilization is shown both as a bar diagram and as a numerical value (percentage).

The following memory utilizations are shown:

- Load memory

  Note

  The load memory is located on the SIMATIC memory card.

- Code work memory: work memory for program code
- Data work memory: work memory for data blocks
- Retentive memory

Display area of the "Memory" pane of the "Online Tools" task card

This area contains the current memory utilization of the associated module. The available memory is shown both as a bar diagram and as a numerical value (percentage). The numerical value is rounded to an integer value.

Note

If less than 1% of a memory area is utilized, the available portion of this memory area is shown as "99%".

The following memory utilizations are shown:

- Load memory

  Note

  The load memory is located on the SIMATIC memory card.

- Code work memory: work memory for program code
- Data work memory: work memory for data blocks
- Retentive memory

See also

CPU memory areas (Page 1839)

8.2.1.4 Checking a module for defects

Determining the diagnostic status of a module

Where is the diagnostics status of a module displayed?

The diagnostic status of a module is displayed in the "Diagnostic status" group in the "Diagnostics" folder in the online and diagnostics view of the module to be diagnosed.

The "Diagnostics status" group consists of the following areas:

- Status
- Standard diagnostics (for S7-300 and S7-400 only for non-CPU modules)

"Status" area

The following status information is displayed in this area:

- Status of the module as viewed by the CPU, for example:
  - Module available and OK.
  - Module defective.
    If the module experiences a fault and you have enabled the diagnostic error interrupt during configuration, the "Module defective" status is displayed.
  - Module configured, but not available.
    Example: Diagnostics data is not available because the current online configuration differs from the offline configuration.
- Detected differences between the configured and the inserted module. Provided it can be ascertained, the order number will be displayed for the set and actual type.

The scope of the displayed information depends on the selected module.

"Standard diagnostics" area

The following diagnostics information for non-CPU modules is displayed in this area:

- Internal and external faults that relate to the overall module
- Associated diagnostics events

Examples of such diagnostics information are:

- Entire backup failed
- Module defective

Note

Diagnostic interrupts

A diagnostic interrupt can be reported to the CPU only if the module has diagnostic interrupt capability and the diagnostic interrupt has been enabled.

The display of the diagnostic interrupt is a snapshot. Sporadic module defects can be identified in the diagnostics buffer of the respective CPU.

Reading out the diagnostics buffer of a CPU

Where do you read out the diagnostics buffer of a CPU?

You read out the diagnostics buffer of a CPU in the "Diagnostics buffer" group in the "Diagnostics" folder in the Online and Diagnostics view.

Structure of the "Diagnostics buffer" group

The "Diagnostics buffer" group consists of the following areas:

- "Events"
- "Settings"

Diagnostics buffer

The diagnostics buffer is used as a log file for the diagnostics events that occurred on the CPU and the modules assigned to it. These are entered in the order of their occurrence, with the latest event shown at the top.

"Events" area

The "Events" area consists of the following elements:

- Check box "CPU time stamp takes into account local PG/PC time"
- Event table
- "Freeze display" or "Cancel freeze" button
- Details of the event: Event no., event ID, description, time stamp, incoming/outgoing information
- "Help on event", "Open in editor", "Save as ..." buttons

Check box "CPU time stamp takes into account local PG/PC time"

If you have not activated the check box, the diagnostics buffer entries are shown with the module time.

If you have activated the check box, the diagnostics buffer entries are shown with the time given by the following formula:

Displayed time = module time + time zone offset on your programming device / PC

This requires the module time to be identical to UTC.

You should use this setting if you wish to see the times of the diagnostics buffer entries for the module expressed in the local time of your programming device / PC.

Selecting or clearing the check box immediately changes the times displayed for the diagnostics buffer entries.

Note

If you use the "WR_SYS_T" instruction in your program or if you set the real-time clock of the CPU using an HMI device instead of using UTC, we recommend that you clear the "CPU time stamp takes into account local PG/PC time" check box. In this case, the module time is the sole time of concern.

Event table

The following information is displayed in the table for each diagnostics event:

- Sequential number of the entry
  The first entry contains the latest event.
- Date and time of the diagnostics event
  If no date and time are shown, the module has no integral clock.
- Short name of the event and, if applicable, the reaction of the CPU

Note

If an individual parameter of a text cannot be determined, the character string "###" is shown in its place.

If no display text is yet available for new modules or new events, the numbers of the events and the individual parameters are stated as hexadecimal values.

- Icon for information related to incoming/outgoing status
  The following table shows the available icons and their respective meaning.

Icon	Meaning
	Incoming event
	Outgoing event
	Incoming event to which there is no independent outgoing event
	User-defined diagnostics event

- Only for S7-1200 and S7-1500 CPUs: Icon for the severity of the event
  The following table shows the available icons and their respective meaning.

Icon	Meaning
	No maintenance and/or no fault
	Maintenance required
	Maintenance demanded
	Error

You can change the order of the columns, adjust the column widths and remove and add individual columns in the event table. In addition, you can sort as follows: by sequential number, by "Date and time" and by "Event".

"Freeze display" or "Cancel freeze" button

The "Freeze display" or "Cancel freeze" button is only enabled when there is an online connection to the CPU.

The default setting is "Freeze display".

The following happens when you click the "Freeze display" button:

- The current display of the diagnostics buffer entries is frozen.
- The labeling of the button changes to "Cancel freeze".

If an error has occurred in your system, diagnostics events can occur very quickly in succession. This produces a high update rate on the display. Freezing the display allows you to calmly examine the situation in more detail.

If the display is frozen and you click the "Cancel freeze" button, the following happens:

- The display of the diagnostics buffer entries is updated again.
- The labeling of the button changes to "Freeze display".

Note

If you freeze the diagnostics buffer display, the CPU continues to enter events in the diagnostics buffer.

Details of the event

If you select a line in the list of events, you obtain detailed information on the respective event:

- Sequential number of the event in the diagnostics buffer
- Event ID
- Description of the event with event-dependent additional information. Examples of this additional information:
  - Command that caused the event
  - Operating mode switch caused by the diagnostics event
- Time stamp
- Only for S7-1200 and S7-1500 CPUs: Associated I&M data (module, rack/slot, plant designation, location designation)
- Priority of the event
- Information on whether the event is an incoming or outgoing event

"Help on event" button

If you click on this button, the selected event is explained in more detail and any remedies given.

Note

For a small number of events, the "Help on event" button is grayed out.

"Open in editor" button

The following table shows if the "Open block" button is active and which function it conceals.

When is the "Open in editor" button enabled?	What happens when you click this button?
If the diagnostics event references the relative address of a block. This is the address of the command that caused the event.	The "Open in editor" function opens the referenced block in the offline view at the programming instruction that causes the error. This allows you to check and, if necessary, change the source code of the block at the specified place and then download it again to the CPU.
If the diagnostics event was triggered by a module.	The "Open in editor" function opens the Device view of the module involved.

"Save as ..." button

If you click this button, the content of the diagnostics buffer is saved in a text file. "Diagnostics", depending on the language, with the extension ".txt" is suggested as the file name. You can however change this name.

"Settings" area

The "Settings" area consists of the following elements:

- "Display events" list
- "Apply settings as default" button
- "Output event information in hexadecimal format" check box

"Display events:" list

There is an check box in this list for every event class (default setting: all check boxes are selected). If you clear a check box, the events of that event class is no longer displayed in the "Events" area. Reselecting the check box displays the associated events once again.

"Apply settings as default" button

If you click this button, the settings are also applied to future occasions when the "Events" tab is opened.

"Output event information in hexadecimal format" check box

If you select the check box, the event IDs in the Events list of the "Events" area is displayed in hexadecimal format. If you clear the check box, the event information is given in text form.

See also

Basic information on the diagnostics buffer (Page 2123)

Saving service data

Purpose

In the event of servicing it may be possible that the SIEMENS Customer Support requires very special information about the state of a module of your system for diagnostic purposes.

If such a case occurs in your system, you will be asked by Customer Support to save the service data of the module and send the resulting file to them.

Where do you carry out the saving of service data of a module?

You carry out the saving of service data of a module in its online and diagnostics view at the following points: In the "Functions" folder in the "Save service data" group"

The "Save service data" group consists of the following areas:

- Online data
- Saving service data

"Online data" area

This area shows the following data for the module:

- Order number
- Firmware version
- Module name (you configured this while configuring the hardware.)
- Rack
- Slot

"Save service data" area

Proceed as follows to create and save a file with special service data:

1. Select the point in the file system at which you want to save the file:
   - You use the path preset in the "Path" field.
   - Click the three-dot (browse) button. In the dialog that opens specify the desired path and enter the file name.
2. Click the "Save data" button.

8.2.1.5 Changing the properties of a module or the programming device/PC

Changing the mode of a CPU

Requirement

There is an online connection to the CPU whose mode you want to change.

Procedure

To change the mode of the CPU, follow these steps:

1. Enable the "Online tools" task card of the CPU.
2. Click the "RUN" button in the "CPU control panel" pane if you want to change the CPU to RUN mode or the "STOP" button if you want to change the CPU to STOP mode.

   Note

   The only button active is the one that can be selected in the current operating mode of the CPU.

3. Acknowledge the confirmation prompt with "OK".

Or:

1. Open the "Online" menu.
2. Choose the "Start CPU" menu command if you want to set the CPU to RUN mode and "Stop CPU" if you want to set the CPU to STOP mode.

   Note

   The only button that is active is the one that can be chosen in the current operating mode of the CPU.

3. Acknowledge the confirmation prompt with "OK".

Or:

1. Click the "Start CPU" button in the toolbar if you want to set the CPU to RUN mode and the "Stop CPU" button if you want to set the CPU to STOP mode.

   Note

   The only button that is active is the one that can be chosen in the current operating mode of the CPU.

2. Acknowledge the confirmation prompt with "OK".

Result

The CPU will be switched to the required operating mode.

Performing a memory reset

Requirement

- There is an online connection to the CPU on which the memory reset is to be performed.
- The CPU is in STOP mode.

Note

If the CPU is still in RUN mode and you start the memory reset, you can place it in STOP mode after acknowledging a confirmation prompt.

Procedure

To perform a memory reset on a CPU, follow these steps:

1. Enable the "Online Tools" task card of the CPU.
2. Click the "MRES" button in the "CPU control panel" pane.
3. Acknowledge the confirmation prompt with "OK".

Result

The CPU is switched to STOP mode, if necessary, and the memory reset is performed on the CPU.

See also

Basics of a memory reset (Page 1570)

Basics of a memory reset (Page 1651)

Basics of a memory reset (Page 1837)

Determining and setting the time of day on a CPU

Where do I find the functions I need?

You determine and change the time of day on a CPU in the "Set time of day" group in the "Functions" folder of the Online and Diagnostics view. This requires an online connection.

Structure of the "Set time of day" group

The "Set time of day" group consists of the following areas:

- Area for reading out and setting the time of day
- Time system (This area does not exist for S7-1200 and will not be examined here.)

Structure of the area for reading out and setting the time of day

This area consists of the following parts:

- Programming device / PC time
  Here the time zone setting, the current date and the current time setting of your programming device / PC are displayed.
- Module time
  Here the date and time values currently read from the module (for example the CPU), are converted to local time and date and displayed.
  If the "Take from PG/PC" check box is selected, when you click the "Apply" button, the date and the PG/PC time converted to UTC are transferred to the module.
  If the "Take from PG/PC" check box is not selected, you can assign the date and time for the integrated clock of the module. After clicking the "Apply" button, the date and the time recalculated to UTC time are transferred to the module.

Updating the firmware of a module

Performing a firmware update

Using firmware files, you can update the firmware of a module.

Requirements

- The module is connected online.
- The module supports a firmware update.
- For those modules that require a supply voltage to perform the firmware update correctly: The supply voltage of the module is secured. For details, see the documentation of the module.

Procedure

To perform a firmware update, follow these steps:

1. Open the module in the Online and Diagnostics view.
2. Select the "Firmware update" group in the "Functions" folder.

   Note

   For S7-1500-CPUs, this group is subdivided into "PLC" and "Display".

3. Click the "Browse" button in the "Firmware update" area in order to select the path to the firmware update files.
4. Select one of these files. The table then lists all modules for which an update is possible with the selected firmware file.

5. Optional: Select the "Run firmware after update" check box to reset the module after the load operation and to start the new firmware.
6. Click the "Start update" button. If the selected file can be interpreted by the module, it is downloaded to the module. If the mode of the CPU needs to be changed, you will be prompted to do this in dialogs.

WARNING

Invalid plant states possible

An S7-1500 CPU immediately goes to STOP mode when you start the firmware update, which can have an effect on the operation of an online process or a machine.
Unexpected operation of a process or a machine can result in severe or fatal injuries and/or damage to property.

Note

After you have run a firmware update, you will need to replace the module involved with the same module with the current firmware version in the hardware configuration of your project. The engineering configuration then matches the actual physical configuration again.

"Run firmware after update" check box

If you have not selected the "Run firmware after update" check box, the previous firmware remains active until the module is reset (for example by cycling power). The new firmware only becomes active after the module has been reset.

If you have selected the check box, the module is automatically reset after the firmware has been downloaded and it then continues with the new firmware.

Activating the firmware following the update has the following consequences:

- A station executes a restart. This means that all modules in the station become unavailable.
- If the corresponding CPU is in RUN, activating the firmware can lead to access errors or other problems in the user program and might even mean that the CPU remains permanently in STOP.

Note

For some CPUs, the "Run firmware after update" check box is grayed out and deactivated. In this case, you must restart the CPU manually.

For S7-1500 CPUs, the "Run firmware after update" check box is grayed out and selected. In this case, the new firmware is activated immediately after the download operation.

See also

Replacing a hardware component (Page 513)

Resetting an S7-1200 CPU to factory settings

Requirement

- There is no memory card inserted in the CPU.
- There is an online connection to the CPU that you want to reset to the factory settings.
- The CPU is in STOP mode.

Note

If the CPU is still in RUN mode and you start the reset operation, you can place it in STOP mode by answering the security prompt with yes.

Procedure

To reset an S7-1200 CPU to factory settings, follow these steps:

1. Open the Online and Diagnostics view of the CPU.
2. Select the "Reset to factory settings" group in the "Functions" folder.
3. Select the "Keep IP address" check box if you want to keep the IP address or the "Delete IP address" check box if you want to delete the IP address.

   Note

   The two check boxes mentioned are only available if the module to be reset is able to choose whether to retain or delete the IP address.

4. Click the "Reset" button.
5. Acknowledge the confirmation prompt with "OK".

Result

The module is switched to STOP mode if necessary and the settings are then reset to factory settings. This means:

- The work memory and the internal load memory and all operand areas are cleared.
- All parameters are reset to their defaults.
- The diagnostic buffer is cleared.
- The time is reset.
- The IP address is kept or deleted depending on which setting you made.

Resetting an S7-1500 CPU to factory settings

Requirement

- If you start a reset to factory settings from the project context, an online connection to the relevant CPU must exist.
- The relevant CPU is in STOP mode.

Note

If the CPU is still in RUN mode and you start the reset operation, you can place it in STOP mode after acknowledging a confirmation prompt.

Procedure

To reset an S7-1500 CPU to factory settings, follow these steps:

1. Open the Online and Diagnostics view of the CPU (either from the project context or via "Accessible devices").
2. Select the "Reset to factory settings" group in the "Functions" folder.
3. Select the "Keep IP address" check box if you want to keep the IP address or the "Delete IP address" check box if you want to delete the IP address.

Note

With "Delete IP address", all IP addresses are deleted. This applies regardless of how you created the online connection.

If a memory card is inserted, selecting the "Delete IP address" option causes the following: The IP addresses are deleted and the CPU is reset to factory settings. Then, the configuration (including IP addresses) that is stored on the memory card is transferred into the CPU (see below). If the memory card was formatted before resetting to factory settings or if it is empty, no IP address is transferred into the CPU.

4. Click the "Reset" button.
5. Acknowledge the confirmation prompt with "OK".

Result

The module is switched to STOP mode if necessary and the settings are then reset to factory settings. This means:

- The work memory and the internal retentive system memory and all operand areas are cleared.
- All parameters are reset to their defaults.
- The diagnostic buffer is cleared.
- The time of day is reset.
- The I&M data are deleted except for I&M0 data.

- The runtime meters are reset.
- The IP address is kept or deleted depending on which setting you made.
- If a memory card was inserted prior to the reset to factory settings, the configuration contained on the memory card (hardware and software) is downloaded to the CPU.

Formatting an S7-1500 memory card

Requirement

- If you start the formatting of the memory card from the project context, an online connection to the relevant CPU must exist.
- The relevant CPU is in STOP mode.

Note

If the CPU is still in RUN mode and you start a formatting operation, you can place it in STOP mode after acknowledging a confirmation prompt.

Procedure

To format an S7-1500 memory card, follow these steps:

1. Open the Online and Diagnostics view of the CPU (either from the project context or via "Accessible devices").
2. Select the "Format memory card" group in the "Functions" folder.
3. Click the "Format" button.
4. Answer the safety prompt with "Yes".

Result

- The memory card is formatted.
- The CPU is temporarily unavailable.
- The project data on the CPU are deleted with the exception of the IP address.
- If you start the formatting of the memory card from the project context, the Online and Diagnostics view remains open. If formatting is started via "Accessible devices", the Online and Diagnostics view will close.

Assigning an IP address to a PROFINET IO device

Basic information on assigning an IP address to a PROFINET IO device

Overview

All PROFINET IO devices work with the TCP/IP protocol and therefore require an IP address for operation on Industrial Ethernet. Once an IP address has been assigned to an IO device, it can be accessed via this address. You can then download configuration data or perform diagnostics, for example.

Requirement

- The Ethernet LAN connection must already be established.
- The Ethernet interface of your programming device or PC must be accessible.
- The IO device that is to be assigned an IP address must be in the same IP band as the programming device or PC.

Starting the address assignment via "Accessible devices"

Requirement

- You have opened the Online and Diagnostics view of the PROFINET IO device using "Update accessible devices" (in the project tree) or "Accessible devices..." ("Online" menu).

Procedure

1. Open the "Functions" folder and the "Assign IP address" group inside this folder. The "MAC address" field displays the MAC address of the PROFINET IO device. The "Accessible devices" button is grayed out.
2. Enter the desired IP address.
3. Enter the subnet mask.
4. If a router is to be used, select the "Use router" check box and enter its IP address.
5. Click the "Assign IP address" button.

Result

The IP address is permanently assigned to the IO device or to the relevant PROFINET interface of the IO device. It is retained even through a startup or a power failure.

Note

For an S7-1500 CPU, you can also use the above-described method to change the IP address of a PROFINET interface, if a project has already been downloaded to the CPU via this interface. This overwrites the IP address downloaded via the project.

See also

Retentivity of IP address parameters and device names (Page 1450)

Starting the address assignment from the project context

Requirement

- An online connection to the PROFINET IO device exists.
- You have opened the Online and Diagnostics view of the PROFINET IO device from the project context.
- The PROFINET IO device is not assigned to any IO controller.

Procedure

1. Open the "Functions" folder and the "Assign IP address" group inside this folder.
2. Click the "Accessible devices" button in order to identify the devices that can be accessed. Note: For an S7-1500 CPU, there are two entries here because it has two PROFINET interfaces.
3. Select the IO device. The "IP address" field, "Subnet mask" field, "Use router" check box and "Router address" field are grayed out and contain the node properties you used to establish the current online access.
4. Click the "Assign IP address" button.

Result

The IP address is permanently assigned to the IO device or to the relevant PROFINET interface of the IO device. It is retained even through a startup or a power failure.

See also

Retentivity of IP address parameters and device names (Page 1450)

Assigning a PROFINET device name

Basic information on assigning a name to a PROFINET IO device

Device name

Before an IO device can be addressed by an IO controller, it must have a device name. This procedure was chosen for PROFINET because names are easier to handle than complex IP addresses.

Assigning a device name to a PROFINET IO device is comparable to setting the PROFIBUS address for a DP slave.

An IO device has no device name in its delivery state. For an IO controller to address an IO device, it must first be assigned a device name using the programming device or PC. It is now ready to transfer the configuration information including the IP address during startup or exchange user data in cyclic operation.

Rules for the device name

The device name is subject to the following limitations:

- Restricted to a total of 240 characters (lower case letters, numbers, dash, or dot)
- A name component within the device name, which is a character string between two dots, must not exceed 63 characters.
- No special characters such as umlauts, brackets, underscore, slash, blank space, etc. The only special character permitted is the dash.
- The device name must not begin or end with the "-" character.
- The device name must not begin with a number.
- The device name form n.n.n.n (n = 0, ... 999) is not permitted.
- The device name must not begin with the string "port-xyz" or "port-xyz-abcde" (a, b, c, d, e, x, y, z = 0, ... 9).

Where do I find the function I am seeking?

You can assign the name of a PROFINET IO device at the following places:

- In the online and diagnostics view of the device in the "Assign name" group of the "Functions" folder. The user interface for this group differs depending on how you open the online and diagnostics view:
  - Open via "Accessible devices"
  - Open from the project context
- In the "Assign PROFINET device name" dialog

See also

Assigning a name in the online and diagnostics view opened via "Accessible devices" (Page 2117)

Assigning a name in the online and diagnostics view opened from the project context (Page 2117)

Assigning a name in the "Assign PROFINET device name" dialog (Page 2118)

Assigning a name in the online and diagnostics view opened via "Accessible devices"

Requirement

- You have opened the online and diagnostics view of the PROFINET IO device using "Update accessible devices" (in the project tree) or "Accessible devices..." ("Online" menu).

Procedure

1. Open the "Functions" folder and the "Assign name" group inside this folder. The "Type" field displays the module type of the PROFINET IO device.
2. Enter the required device name in the "PROFINET device name" input box.
3. Optional: Select the "Flash LED" check box in order to run an LED flash test on the PROFINET IO device. In this way, you verify that you are naming the desired IO device.

   Note

   The LED flash test is not supported by all PROFINET IO devices.

   The LED flash test runs until you cancel it. This is done, for example, by clearing the "Flash LED" check box or by closing the online and diagnostics view.
4. Click "Assign name".

Result

The entered name is assigned to the PROFINET IO device.

Assigning a name in the online and diagnostics view opened from the project context

Requirement

- You have opened the online and diagnostics view of the PROFINET IO device from the project context.
- The PROFINET IO device can be accessed using at least one PG/PC interface.

Procedure

1. Open the "Functions" folder and the "Assign name" group inside this folder. The "PROFINET device name" drop-down list displays the current name in the offline project, and the "Type" box shows the module type of the PROFINET IO device.

   Note

   For CPUs with several PROFINET interfaces, the names of all existing PROFINET interfaces are displayed in the offline project.

2. Choose a different name from the drop-down list, if necessary.

   Note

   In steps 3 to 5, you determine the IO devices that are present in the PROFINET subnet.

3. In the "PG/PC interface for assignment" drop-down list, select the PG/PC interface you want to use to establish the online connection.
4. Optional: Use the three check boxes to make a selection from all IO devices available online.
5. Click the icon for determining the IO devices present in the PROFINET subnet. The table is then updated.
6. Select the desired IO device in the table.
7. Optional: Select the "Flash LED" check box in order to run an LED flash test on the PROFINET IO device. In this way, you verify that you are naming the desired IO device.

   Note

   The LED flash test is not supported by all PROFINET IO devices.

   The LED flash test runs until you cancel it. This is done, for example, by clearing the "Flash LED" check box, by selecting another IO device in the table, or by closing the online and diagnostics view.

8. Click "Assign name".

Result

The selected name is assigned to the PROFINET IO device or one of its PROFINET interfaces.

Assigning a name in the "Assign PROFINET device name" dialog

Requirement

- You have opened the "Assign PROFINET device name" dialog from the network view (from the shortcut menu of a PN/IE connection).
- The PROFINET IO device can be accessed using at least one PG/PC interface.

Procedure

1. The following is displayed in the "PROFINET device name" drop-down list:
   - the name of the interface that was used to open the project in the current offline project
   - the names of those IO devices that are connected by means of this interface

   The "Type" field displays the module type of the PROFINET IO device.
   Choose a different name from the drop-down list, if necessary.

   Note

   In steps 2 to 4, you determine the IO devices that are present in the PROFINET subnet.

2. In the "PG/PC interface for assignment" drop-down list, select the PG/PC interface you want to use to establish the online connection.
3. Optional: Use the three check boxes to make a selection from all IO devices available online.
4. Click the icon for determining the IO devices present in the PROFINET subnet. The table is then updated.
5. Select the desired IO device in the table.
6. Optional: Select the "Flash LED" check box in order to run an LED flash test on the PROFINET IO device. In this way, you verify that you are naming the desired IO device.

   Note

   The LED flash test is not supported by all PROFINET IO devices.

   The LED flash test runs until you cancel it. This is done, for example, by clearing the "Flash LED" check box or by selecting another IO device in the table.
7. Click "Assign name".

Result

The selected name is assigned to the PROFINET IO device or the interface that was used to open the dialog.

Calibrating an S7-1500 analog module

Calibrating an S7-1500 analog module - Overview

Where do you calibrate an S7-1500 analog module?

You calibrate an S7-1500 analog module in its Online and Diagnostics view in the "Calibrate" group of the "Functions" folder.

Overview of the function scope of the calibrating function

You can perform the following functions for an S7-1500 analog module in the "Calibrate" group:

- Specifying the current calibration of all channels
- Calibrating a channel
- Canceling a running calibration
- Resetting the calibration of a channel to the factory settings

Requirement for the calibrating function described below

The following is required for the calibrating function described below:

- You have opened the Online and Diagnostics view from the project context (thus not from the project tree or via the "Online" menu).
- There is an online connection to the analog module that is to be calibrated.
- Offline and online configuration are identical.

Calibrating an S7-1500 analog module

Overview of the calibration of a channel of an S7-1500 analog module

The calibration of a channel of an S7-1500 analog module consists of the following steps:

1. Start calibration
2. Perform the second step up to the next to last step of the calibration
3. Complete calibration

These steps are described in more detail in the following section.

Requirement

- You have opened the Online and Diagnostic view of the S7-1500 analog module from the project context and are in the "Calibrate" group of the "Functions" folder.
- The associated CPU is online.
- No calibration is currently running on the analog module (if you want to start the calibration) or the last step initiated has been performed successfully (if you want to resume or complete the calibration).

Procedure for starting the calibration

To start the calibration, follow these steps:

1. In the overview table, select the line that belongs to the channel to be calibrated.
2. Click the "Start manual calibration" button.

The user interface then changes as follows:

- The overview table and the "Start manual calibration" button and "Set to factory settings" buttons become inactive.
- The step display is activated and displays the numbers of the current and last steps.
- The "Command" field becomes active and indicates what the user must do in the next calibration step.
- The "Status" field becomes active and shows the current status of the calibration, e.g., "Calibration successfully started".
- The "Measured value" field becomes active. For an input module a value is displayed here; you must enter a value here for an output module.
- The "Cancel" button becomes active.
- The "Next" button becomes active. This button can be used to advance to the next step of the calibration.

Procedure for the second to the next to last step of the calibration

Follow these steps:

1. Click the "Next" button.

The fields of the user interface described above are then updated.

Procedure for the last step of the calibration

Follow these steps:

1. Click the "Next" button.

The user interface then changes as follows:

- The overview table becomes active.
- The calibration display of the calibrated channel is updated.
- The "Start manual calibration" button and "Set to factory settings" buttons become active.
- The step display is deactivated and the numbers of the current step and last steps are empty.
- The "Command" field becomes inactive and is empty.
- The "Status" field becomes inactive and shows the last status of the calibration, e.g., "Calibration successfully finished".
- The "Measured value" field becomes inactive and is empty.
- The "Cancel" button becomes inactive.
- The "Next" button becomes inactive.

Error occurrence

If an error occurs during the calibration, the module cancels the calibration. Afterwards, the channel that was to be calibrated has the same settings as before the start of the calibration.

Except for the "Status" field, the user interface appears the same after the occurrence of an error as before the calibration. The "Status" field displays the error that the module detected during the calibration.

Canceling a running calibration of an S7-1500 analog module

Requirement

- You have opened the Online and Diagnostic view of the S7-1500 analog module from the project context and are in the "Calibrate" group of the "Functions" folder.
- The associated CPU is online.
- A calibration is currently running on the analog module.

Procedure

To cancel a running calibration, follow these steps:

1. Click the "Cancel" button.

Result

The running calibration is canceled, and afterwards the channel to be calibrated has the same settings as before the calibration.

All operator controls in the user interface are deactivated until the cancelation is complete. Except for the "Status" field, the user interface appears the same afterwards as before the calibration. The "Status" field displays the result of the cancelation.

Resetting an S7-1500 analog module to factory settings

Requirement

- You have opened the Online and Diagnostic view of the S7-1500 analog module from the project context and are in the "Calibrate" group of the "Functions" folder.
- The associated CPU is online.

Procedure

To reset a channel of an S7-1500 analog module to factory settings, follow these steps:

1. Select the line associated with the channel to be reset in the overview table.
2. Click the "Set to factory settings" button.

Result

All operator controls in the user interface are deactivated until the reset operation is complete. Except for the "Status" field, the user interface appears the same afterwards as before the reset operation. The "Status" field displays the result of the reset operation.

8.2.1.6 Diagnostics in STOP mode

Basic information on the diagnostics buffer

Function

The operating system of the CPU enters the errors detected by the CPU and the diagnostics-capable modules into the diagnostics buffer in the order in which they occurred. This includes the following events:

- Every mode change of the CPU (POWER UP, change to STOP mode, change to RUN mode)
- Every hardware and diagnostic error interrupt

The top entry contains the most recent event. The entries in the diagnostics buffer are stored permanently. They are retained even if the power supply fails and can only be deleted by resetting the CPU to factory settings.

A diagnostics buffer entry contains the following elements:

- Time stamp
- Error ID
- Additional information specific to the error ID

Advantages of the diagnostics buffer

The diagnostics buffer offers the following advantages:

- After the CPU has changed to STOP mode, you can evaluate the last events prior to the STOP so that you can locate and identify the cause of the STOP.
- You can detect and eliminate the causes of errors more quickly and thus increase the availability of the system.
- You can evaluate and optimize the dynamic system response.

Organization of the diagnostics buffer

The diagnostics buffer is a ring buffer. The maximum number of entries for the S7-1200 CPUs is 50. When the diagnostics buffer is full and a further entry needs to be made, all existing entries are shifted by one position (which means that the oldest entry is deleted) and the new entry is made at the top position that is now free (FIFO principle: first in, first out).

Evaluation of the diagnostics buffer

The contents of the diagnostics buffer can be accessed as follows:

- Using the Online and Diagnostics view

The evaluation of events occurring prior to the error event (e.g., transition to STOP mode) allows you to obtain a picture of the possible causes or to zero in more closely or specify in more detail the possible causes (depending on the error type).

Read the detailed information about the events carefully and use the "Help on event" button to obtain additional information and possible causes of individual entries.

Note

To make the best use of the time stamp information on the diagnostics buffer entries in time-critical systems, it is advisable to check and correct the date and time of day on the CPU occasionally.

Alternatively, it is possible to perform a time-of-day synchronization using an NTP time server.

See also

Resetting an S7-1200 CPU to factory settings (Page 2111)

Determining the cause of a STOP of a CPU (Page 2124)

Determining and setting the time of day on a CPU (Page 2108)

Assigning the clock parameters (Page 1594)

Determining the cause of a STOP of a CPU

Requirement

The CPU you want to analyze is in STOP mode.

Procedure

To find out the reason why a CPU changed to STOP, follow these steps:

1. Open the online and diagnostics view of the CPU.
2. Select the "Diagnostics buffer" group from the "Diagnostics" folder.
3. Evaluate the events occurring prior to the transition to STOP mode. Use this to obtain a picture of the possible causes or to zero in on or specify in more detail the possible causes (depending on the error type).
   Read the detailed information about the events carefully and use the "Help on event" button to obtain additional information and possible causes of individual entries.

Result

You were able to zero in on or determine in more detail the cause of the CPU STOP.

Note

If the analysis does not enable you to overcome the problem, contact Customer Support. In this case, use the "Save as" button to back up the content of the diagnostics data to a text file and submit it to Customer Support.

See also

Reading out the diagnostics buffer of a CPU (Page 2101)

8.2.1.7 Online accesses in the Online and Diagnostics view

Displaying status of the online connection

Requirement

- The associated device can be accessed using at least one PG/PC interface.

Procedure

1. Open the online and diagnostics view for the device whose online connection status you want to display.
2. Select the "Online access" group.

Note

The "Online access" group exists only for CPUs and some CPs. However, if you have opened the online and diagnostics view using the "Show/update accessible devices" function, it is not displayed.

Result

The status of the online connection is displayed in the "Status" area both graphically and in text form.

Specifying a PG/PC interface, going online

Requirement

- The associated device can be accessed using at least one PG/PC interface.
- There is currently no online connection to the relevant device.

Procedure

1. Open the Online and Diagnostics view of the device to which you want to establish an online connection.
2. Choose the "Online access" group and the "Online access" area within this group.

   Note

   The "Online access" group exists only for CPUs and some CPs. If you have opened the Online and Diagnostics view using the "Show/update accessible devices" function, it is not displayed.

3. If an online connection was established previously for the device, the associated data for this online connection is preset in the drop-down lists. In this case, you can immediately continue with the last step of this operation, provided you have not changed the IP address in the meantime using the Online and Diagnostics view.
4. Choose the interface type in the "Type of PG/PC interface" drop-down list.
   The "PG/PC interface for online access" drop-down list then shows only the interfaces of the programming device or PC that match the selected interface type.
5. In the "PG/PC interface for online access" drop-down list, select the programming device or PC interface via which you want to establish the online connection.
6. Optional: Click the "Properties" button to change the properties of the associated CP.
7. In the "Connection to subnet" drop-down list, select the subnet via which the device is connected to the PG/PC interface.

   Note

   The PG/PC interface is connected to an interface of a device.

   If you only want to access this device, select the setting "Directly at slot <interface name>" in the drop-down list.

   If you want to access another device by means of routing, however, create a subnet at this interface in the hardware configuration and then select this subnet in the drop-down list.

8. If the device is accessible via a gateway, select the gateway that connects the two subnets involved in the "1st gateway" drop-down list.
9. In the "Device address" entry field, enter the IP address of the device to which you want to establish an online connection, if necessary.

   Note

   For CPUs with multiple IP addresses, select the IP address of the PROFINET interface you want to use to establish an online connection from the "Device address" drop-down list.

10. Alternatively: Click the "Show accessible devices" button and choose the device from the list of accessible devices to which you want to establish an online connection.
11. Click the "Go online" button.

Result

The online connection to the desired device is established.

Going offline

Requirement

- There is currently an online connection to the relevant device.

Procedure

1. Open the online and diagnostics view of the device for which you want to disconnect the online connection.
2. Choose the "Online access" group and the "Online access" area within this group.

   Note

   The "Online access" group exists only for CPUs and some CPs. However, if you have opened the online and diagnostics view using the "Show/update accessible devices" function, it is not displayed.

3. Click the "Go offline" button.

Result

The online connection to the desired device will be disconnected.

Performing the flash test for a device with an online connection

Requirement

- There is currently an online connection to the relevant device.
- The FORCE function is not active.

Procedure

1. Open the online and diagnostics view of the device for which you want to perform a flash test.
2. Choose the "Online access" group and the "Status" area within this group.

   Note

   The "Online access" group exists only for CPUs and some CPs. However, if you have opened the online and diagnostics view using the "Show/update accessible devices" function, it is not displayed.

3. Select the "LED flash test" check box.

Result

- On an S7-1200 CPU, the RUN/STOP, ERROR and MAINT LEDs flash.
- On an S7-1500 CPU, the RUN/STOP, ERROR, and MAINT LEDs flash.
- On an S7-300 or S7-400 CPU, the FRCE LED flashes.

The LEDs flash until you cancel the flash test. This is done, for example, by clearing the "LED flash test" check box, by changing to another group of the online and diagnostics view, or by changing settings in the "Online access" area.

8.2.1.8 Checking PROFIBUS DP subnets for faults

Basic information on the diagnostic repeater

What is the diagnostic repeater?

The diagnostic repeater is a repeater that can monitor a segment of an RS 485 PROFIBUS subnet (copper cable) during operation and signal line errors to the DP master via a diagnostics message frame.

The diagnostic repeater detects, localizes and visualizes line errors during operation at an early stage. As a result, problems in the system are identified early and production downtimes will be minimized.

Function of the diagnostic repeater

The diagnostic repeater can perform line diagnostics on the DP2 and DP3 segments because it has a measuring circuit for these segments.

The line diagnostics run in two steps:

- Step 1: Topology determination
  You start the topology determination by calling the "DP_TOPOL" instruction in your program. The diagnostic repeater then determines the PROFIBUS addresses and the distance of the devices and creates a topology table.
- Step 2: Error localization
  The diagnostic repeater checks the lines during operation. It determines the distance to the point of the error and the reason for the error; it then issues a diagnostics alarm with relative information on the error location.

Display of detailed information on the determined error location

You receive detailed information on the determined error location in the Online and Diagnostics view of the diagnostic repeater.

- By means of icons
- By means of a display with graphics and text

See also

Displaying the status of the segment diagnostics using icons (Page 2129)

Displaying the status of the segment diagnostics using graphics and text (Page 2130)

Displaying the status of the segment diagnostics using icons

Where do I find the information I need?

The icons for the status of the segment diagnostics are available:

- In the expanded "Segment diagnostics" folder in the navigation pane of the Online and Diagnostics view of the relevant diagnostic repeater.

The diagnostics icon associated with the segment will be displayed behind the segment designation. It must be noted here that line errors will be displayed for the DP2 and DP3 segments only. The DP1 and programming device segments do not display errors in the form of a diagnostics icon; rather, they signal only a few bus errors.

Diagnostics icons

The following table shows the available icons and their meaning.

Icon	Meaning
	Segment is error-free
	Segment contains errors
	Segment is deactivated

Displaying the status of the segment diagnostics using graphics and text

Where is the status of the segment diagnostics displayed with graphics and text?

The status of the segment diagnostics will be displayed using graphics and text in the "DP1", "DP2", "DP3", and "PG" groups of the "Segment diagnostics" folder in the Online and Diagnostics view of the relevant diagnostic repeater.

Structure of the "DP1", "DP2" "DP3", and "PG" groups

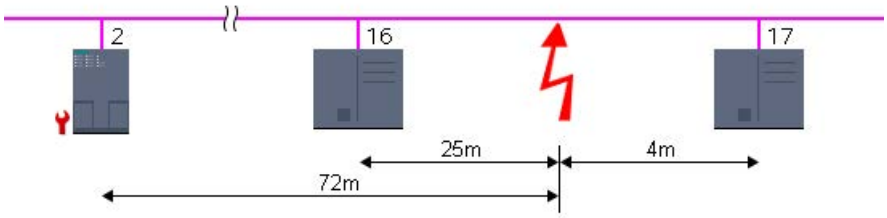
The "DP1", "DP2", "DP3", and "PG" groups consist of the following elements:

- "Error location" field
- "Error" field
- "Resolution" field
- "Help on event" button
- "Freeze display" or "Cancel freeze" button

"Error location" field

This field displays the error location graphically, provided the diagnostic repeater can determine the location.

The following picture shows an example for a line error occurring in the DP2 segment.



In this example, the diagnostic repeater has the PROFIBUS address 2, and a line error has occurred between the devices with PROFIBUS addresses 16 and 17. This line error is located 25 m from device 16, 4 m from device 17, and 72 m from the diagnostic repeater.

"Error" field

The error is explained in plain text in this field.

"Resolution" field

Here, you will find actions for resolving the error.

"Help on event" button

Click this button to obtain a more detailed explanation of the error and additional details on resolving the error, if applicable.

"Freeze display" or "Cancel freeze" button

The "Freeze display" or "Cancel freeze" button is only enabled when there is an online connection to the diagnostic repeater.

The default setting is "Freeze display".

The following happens when you click the "Freeze display" button:

- The current display of the segment diagnostics is frozen.
- The labeling of the button changes to "Cancel freeze".

If the display is frozen and you click the "Cancel freeze" button, the following happens:

- The display of the segment diagnostics is updated again.
- The labeling of the button changes to "Freeze display".

8.2.2 Connection diagnostics

8.2.2.1 Overview of connection diagnostics

Basics

Connection diagnostics, as described below, refers to the diagnostics of communication connections.

The connection diagnostics is started each time an online connection is established to a module (CPU or CP) that participates in one or more communication services. The connection status is updated automatically in the background.

In the case of one-way connections, an online connection must exist to the communication partner that has established the communication connection.

On connections configured at both ends, a distinction between the following two situations must be made:

- If there is an online connection to only one connection endpoint, only the part of the connection belonging to this connection endpoint can be diagnosed.
- If there is an online connection to both connection endpoints, both parts of the connection (and therefore the entire connection) can be diagnosed.

Basic connections diagnostics options

Connection diagnostics can be performed as follows:

- Using icons on the connection status display
  This display is generated in the connection table.
- Through detailed connection diagnostics
  This step is available in the "Diagnostics > Connection information" area of the Inspector window.

Requirement for the connection diagnostics described below

You can display the details of either all the communication connections created in the project (default) or selected communication connections in the connection table.

The connection diagnostics described in the following assume that you display the details of selected communication connections. To do this, clear the "Show all connections" option in the shortcut menu.

See also

Working with the connection table (Page 616)

8.2.2.2 Displaying the connection status using icons

Content of connection table without an online connection

- For a CPU or CP, the connection table lists the communication connections (including properties) configured offline, if an online connection is not established.

Content of connection table with an online connection

After the online connection has been established, the properties of the communication connections listed offline will be expanded to include diagnostics icons for the connection status ("Online status" column).

In addition, entries for all communication connections that exist online only (e.g., connections for the instructions for Open User Communication, programming device and OP connections, connections for web server access) will now be added to the connection table.

For connections that exist online or offline only, the diagnostics icon at the bottom right is combined with a smaller additional comparison status icon.

Diagnostics icons for communication connections

The following table shows the diagnostics icons for communication connections.

Icon	Meaning
	Connection setup
	Connection not setup / is being setup
	Connection not available

Diagnostics icons for the comparison status

The diagnostic icons for communication connections can be combined at the bottom right with additional smaller icons that indicate the result of the online/offline comparison. The following table shows the available comparison icons and their meaning.

Icon	Meaning
	Connection exists online only
	Connection exists offline only

8.2.2.3 Detailed connection diagnostics

Detailed connection diagnostics - overview

Where do I perform detailed connection diagnostics?

To perform detailed connection diagnostics, go to the "Diagnostics > Connection" information of the Inspector window.

How do I open the "Diagnostics > Connection information" area of the Inspector window?

The following options are available for opening the "Connection information" tab of the Inspector window.

- Select the line of the relevant connection in the connection table. Click the "Diagnostics" and "Connection information" tabs one after the other in the Inspector window.
- Double-click the diagnostics icon of the relevant connection in the connection table.
- This step takes you to the programming editor for a S7 communication instruction or open user communication instruction. Double-click the diagnostic icon of the instruction (stethoscope).

Structure or the "Diagnostics > Connection information" area of the Inspector window

Requirements: the content of the "Connection information" tab has been filled, and an online connection to at least one end point of the relevant connection has been established.

If a module has been selected (network view), the tab will contain the following group:

- Connection resources (for S7-1200 and S7-1500)

If a connection has been selected (connection table), it will contain the following groups:

- Connection details
- Address details of the connection (for S7-1200 and S7-1500)

Determining online connection resources for S7-1200

Where do you determine the online connection resources?

The online connection resources are obtained in the "Connection resources" group. This group is located in the "Diagnostics > Connection information" area of the Inspector window. It is displayed only if you have selected a module in the network view to which an online connection exists.

Number of connection resources

- Maximum number: Specifies the maximum number of available connection resources of the module.
- Not assigned: Indicates how many connection resources are not yet assigned. If connection resources are already reserved for certain types of communication, then the unreserved connection resources cannot always be used for the various connection types.

Reserved and currently assigned connection resources

For the communication types indicated below, the connection resources that are reserved and currently assigned by the module will be displayed.

Communication type	Meaning
PG communication	Resources for connections between the module and programming devices (for example, for the establishment of a connection from the project tree, for online diagnostics, etc.)
HMI communication	Resources for connections between the module and HMI devices
Open User Communication	Resources for connection of open user communication instructions
S7 communication	Resources for configured S7 connections, through which data can be exchanged by calling instructions in the user program.
Other communication	Specifies other assigned connection resources for which connection resources are not reserved.

Determining online connection resources for S7-1500

Where do you determine the online connection resources?

The online connection resources are obtained in the "Connection resources" group. This group is located in the "Diagnostics > Connection information" area of the Inspector window. It is displayed only if you have selected a module in the network view to which an online connection exists.

Description of the detailed display of the connection resources

The detailed display of the connection resources includes:

- Number of available connection resources
- Number of configured connection resources
- Number of connection resources still available

For a description of these, go to here (Page 586) .

Determining connection details

Where do I determine the connection details?

The connection details are obtained in the "Connection details" group. This group is located in the "Diagnostics > Connection information" area of the Inspector window.

When is the "Connection details" group filled in?

The following requirements must be met to fill in the "Connection details" group on the "Connection information" tab:

- An online connection to the end point of the relevant connection must exist.
- You have selected a line in the connection table.

Structure of the "Connection details" group

The "Connection details" group consists of the following elements:

- Local ID (hex)
- Connection type (for S7-1200 and S7-1500)
- Protocol
- Connection status: icon and description
- Details
- Last status change (for S7-300 and S7-400 only)

Determining the address details of a connection

Where do I determine the address details of a connection?

The address details of a connection are obtained in the "Connection address details" group. This group is located in the "Diagnostics > Connection information" area of the Inspector window.

For which CPUs is the "Connection address details" group available?

The "Connection address details" group of the "Connection information" tab is available for S7-1200 and S7-1500 CPUs.

When is the "Connection address details" group filled in?

The following requirements must be met to fill in the "Connection address details" group on the "Connection information" tab:

- An online connection to the end points of the relevant connection must exist.
- You have selected a line in the connection table.

Structure of the "Connection address details" group

The address details relevant to the connection type are specified for the two communication partners.

8.2.3 System diagnostics with 'Report System Errors'

8.2.3.1 Introduction to system diagnostics with 'Report System Errors'

Introduction

When a system error occurs, hardware components and devices of other manufacturers (slaves whose properties are set by the GSD files) can trigger organization block calls.

Example: If a wire breaks, a module with diagnostics capability can call OB 82.

The hardware components provide information on the system error that has occurred. The start event information, in other words, the local data of the assigned OB (including data record 0), provides general information on the location (for example, logical address of the module) and type (for example channel error or module fault) of the error.

The problem can also be specified in greater detail using additional diagnostics information (reading data record 1 with the "RDSYSST" instruction or reading the diagnostics frame from standard DP slaves with the "DPNRM_DG" instruction): For example, channel 0 or 1, broken wire or measuring range exceeded.

System diagnostics with "Report System Errors" (RSE) provides you with a convenient way of evaluating this diagnostics information for S7-300/400 PLCs, ET200S, ET200Pro and software PLCs and to display it in the form of messages. The required blocks and alarm texts are created in the properties of the particular PLC. You only need to download the generated blocks to the CPU and, if required, transfer the texts to connected HMI devices.

To display diagnostics events graphically, for example on an HMI device or via a Web server, you can generate one or more status DBs. These status DBs are updated by the system diagnostics blocks and then contain information on the current state of the system.

8.2.3.2 Basics of system diagnostics

Using system diagnostics with "Report System Errors", you can generate blocks that analyze errors in the system and generate alarms with a textual error description and the error location.

These alarms are defined per component with alarm capabilities (for example channel errors or rack errors) and are limited to 255 alarms component with alarm capability. You will receive a message if this limit is exceeded.

General settings that you make or change in system diagnostics are stored along with the project. The influence on the system diagnostics only takes effect after generating the blocks, compiling the hardware configuration and downloading the configuration to the components involved.

Recommended procedure

Make the settings for reporting system errors and the structure of the alarms. Specify which diagnostics blocks will be created. Configure the OBs, Status DBs, as well as the PLC in STOP. Generate the blocks and download the configuration to apply the changes.

You will find detailed information in the sections below.

Note

If you use system diagnostics, the system response of the plant may change if an error occurs. For example, the CPU may not change to "STOP" mode as it would without system diagnostics.

Make sure that all protective mechanisms of the plant are working properly.

8.2.3.3 Modules with limited diagnostic possibilities

Introduction

The components of the S7-300 stations, S7-400 stations, PROFINET IO devices, DP slaves, and WinAC are supported by "Report System Errors", provided that they support functions such as diagnostic interrupts, pull/plug interrupts, and channel-specific diagnostics.

Components not supported

The following components are not supported by "Report System Errors":

- M7 configurations
- PROFIBUS DP configurations using a DP master interface module (CP 342-5 DP) in S7-300 stations
- PROFINET IO devices using an external controller (CP 343-1 Advanced) in S7-300 stations

Please note that a hot restart can also result in missing interrupt alarms. This behavior is due to the fact that the alarm acknowledgment memory of the CPU is not cleared during a hot restart, but "Report System Errors" resets the internal data. Not all of the module or channel faults that occur before startup or during a failure will be reported.

A maximum of 8 channel faults are reported per module.

Note

If you are using a CP 443-5 and it is in STOP mode, a master system failure is not reported at startup.

PROFIBUS DP

For all PROFIBUS DP slaves, station faults (failure/return) are indicated with a plain text alarm.

For all PROFIBUS DP slaves, the manufacturer-specific diagnostics are supported with the following limitations:

- Only V1 slaves are supported. These are slaves in whose GSD file the entry "DPV1_Slave=1" is set.
- The DP interrupt mode for this slave must be set to "DPV0".

Diagnostics repeater

The alarms of the diagnostics repeater are output as plain text in DPV0 mode. The texts are read out from the GSD file.

PROFINET IO

Below you will find the diagnostics of the various PROFINET IO devices that are supported by "Report System Errors".

Errors that occur during operation (CPU in RUN):

- Device faults (failure, return) are supported
- Module faults and submodule faults (Module/submodule removed, Wrong module/submodule, Compatible module/submodule) are supported
- Channel faults are supported and extended channel fault information is evaluated

AS-Interface

For AS-Interface slaves, an alarm is sent if the actual configuration does not match the expected configuration.

Shared devices

"Report System Errors" evaluates the information as to whether a (sub)module has been configured as a shared device. Because "Report System Errors" is always looking at a CPU, only the (sub)modules that have the full access setting for this CPU are taken into consideration for the diagnostics. The (sub)modules configured as "not assigned" are not evaluated.

8.2.3.4 Diagnostics blocks for reporting system errors

Once you have made the settings for reporting system errors, the required blocks (FB with assigned instance DB and one or more global DBs, an FC and, depending on the setting, also OBs that do not yet exist) and status DBs are created the next time you compile the hardware configuration.

The following blocks are created:

- Diagnostics FB (default: FB 49),
- Instance DB for the diagnostics FB (default: FB 49),
- Global DB (default: FB 50),
- Diagnostics FC (default: FC 49),
- Error OBs (are automatically created if the CPU supports this function),
- Cyclic OBs (if you have selected this option in the "OB configuration" group box),
- Optional status DBs

The created blocks (except OBs) are stored in the project tree under "Program blocks > System blocks > Report System Errors".

Note

The block numbers set for the created system diagnostics blocks must not overlap existing block numbers. Moreover, the numbers must not overlap the numbers of blocks located in libraries.

If problems occur in the generation of blocks, check whether the preset numbers (FB 49, DB 49, FC 49, DB 50) have been changed. If this is the case, try to repeat the generation with the preset block numbers or use numbers which you are certain are not in use elsewhere.

See also

Properties of the blocks (Page 2140)

8.2.3.5 Properties of the blocks

Diagnostics blocks

The created diagnostics blocks (FB with assigned instance DB and one or more global DBs and an FC) evaluate the local data of the error OB and read any additional diagnostics information from the hardware component that triggered the error.

They have the following properties:

- Created in RSE language (report system errors) (also applies to the blocks listed above)
- Know-how protected (also applies to the blocks listed above)
- Delay incoming interrupts during runtime

Status DBs

The status DBs act as the interface for the diagnostics blocks and allow diagnostics events to be displayed graphically on an HMI device or a Web server.

See also

Settings for system diagnostics blocks (Page 2147)

8.2.3.6 Supported error OBs

The following error OBs are created automatically if they are supported by the configured CPU:

OB	Diagnostics FB call	Description
OB 70 (IO redundancy error)	possible	This OB exists only with H-CPUs.
OB 72 (CPU redundancy error)	possible	This OB exists only with H-CPUs.
OB 73 (communications redundancy error)	possible	This OB exists only with a few H-CPUs.
OB 81 (power supply error)	possible	
OB 82 (diagnostics interrupt OB)	possible	
OB 83 (pull/plug interrupt)	possible	

OB	Diagnostics FB call	Description
OB 85 (program execution error)	not possible	If 'RSE' creates this OB when it is generating the diagnostics blocks, additional networks are inserted to implement the following program sequence: • If errors occur when updating the process image (for example removing the module), the CPU is prevented from changing to STOP so that the diagnostic FB can be run in OB 83. If there is a setting for "CPU STOP" after an alarm from "RSE", this takes effect in OB 83. • With all other error events of OB 85, the CPU changes to STOP.
OB 86 (failure of an expansion rack, a DP master system or a distributed I/O device)	possible	

If the OBs already exist...

Existing error OBs on not overwritten. If appropriate, the call for the diagnostics FB is appended in the existing OB.

If the configuration includes distributed I/O devices...

To evaluate errors in the distributed I/O, the created FB automatically calls the "DPNRM_DG" instruction (read diagnostics data of the DP slaves). For this function to operate correctly, the created FB must be called either only in OB 1 or in a cyclic interrupt OB with a short call interval and in the startup OBs.

Note

Please note the following:

- Due to the OB 85 created by the system diagnostics, the CPU does not change to STOP if the "error updating the process image" error event occurs.
- OB 85 is also called by the CPU in the following error situations.
  - "Error event for an OB that is not loaded"
  - "Error calling or accessing a block that is not loaded"
    When these errors occur, the OB 85 generated by the system diagnostics still changes the CPU to STOP, just like before the use of system diagnostics.

8.2.3.7 Diagnostics status DB

Within system diagnostics, you have the option of creating a status DB. This status DB is updated by the system diagnostics blocks and then contains information on the current system status.

Interface for the diagnostics status DB

The created data block (default: DB 127) allows you to query the system state of a configured component and, if required, all its lower-level components.

This data block is required to support system diagnostics using the Web server or diagnostics viewer. As default, it is enabled.

Note

Please note that system diagnostics must be activated both on the Web server or diagnostics viewer and in the Inspector window of the PLC to be able to display diagnostics data of the status DBs.

After restarting a CPU with Web server, the module state is displayed following a delay. To shorten the waiting time, you can call the RSE diagnostics block in a cyclic interrupt OB.

Structure of the diagnostics status DB

Address	Name	Data type	Description
+0	Directory		
0	D_Version	WORD	Version supported by system diagnostics
2	D_pGlobalState	WORD	Byte offset to the start of the "GlobalState" section
4	D_pQuery	WORD	Byte offset to the start of the "Query" section
6	D_pComponent	WORD	Byte offset to the start of the "Component" section
8	D_pError	WORD	Byte offset to the start of the "Error" section
10	D_pState	WORD	Byte offset to the start of the "State" section
12	D_pAlarm	WORD	Byte offset to the start of the "Alarm" section
14	D_pSubComponent	WORD	Byte offset to the start of the "Subcomponent" section
+16	GlobalState		
0	G_EventCount	WORD	ID of the last event (counter)
2.0	G_StartReporting	BOOL	Startup evaluation active
+20	Query		
0	Q_ClientID_User	DWORD	ID of the client; here, please use a value between 1 and 255. Make sure that different clients use different IDs.
4	Q_ClientID_Intern	DWORD	ID of the client (internal)

Address	Name	Data type	Description
8.0	Q_WithSubComponent	BOOL	With/without status of the lower-level components (slower)
8.1	Q_SubComponentAlarm	BOOL	AS-i master returns AS-i slave interrupts
8.2	Q_Reserved2	BOOL	Reserved
8.3	Q_Reserved3	BOOL	Reserved
8.4	Q_Reserved4	BOOL	Reserved
8.5	Q_Reserved5	BOOL	Reserved
8.6	Q_Reserved6	BOOL	Reserved
8.7	Q_Reserved7	BOOL	Reserved
9.0	Q_Start	BOOL	Start query
10	Q_Error	BYTE	Internal error in query
11	Q_Reserved8	BYTE	Reserved
+32	Component		
0	C_AddressMode	BYTE	Addressing mode of the module
1	C_Reserved1	BYTE	Reserved
2	C_ComponentID	WORD	Hardware ID of the component (internal)
+36	Error		
0	E_ErrorNo	WORD	Index of the requested/actual error
2.0	E_LastError	BOOL	Is set when E_ErrorNo does not equal 0. Value TRUE if E_ErrorNo is the index of the last error otherwise FALSE
2.1	E_Reserved	ARRAY [1 to 15] BOOL	Reserved
+40	State		
0	S_Hierarchy	BYTE	Reserved
1	S_Periphery	BYTE	Reserved
2.0	S_SupFault	BOOL	The component cannot be reached
2.1	S_NotAvailable	BOOL	The component does not exist
2.2	S_Faulty	BOOL	The component is faulty, the "Alarm" section is not empty
2.3	S_MoreErrors	BOOL	There are more errors than system diagnostics can store
2.4	S_Maintenance1	BOOL	A maintenance request is pending
2.5	S_Maintenance2	BOOL	A maintenance demand is pending
2.6	S_Deactivated	BOOL	The component was deactivated *)
2.7	S_Reserved2	BOOL	Reserved
3.0	S_SubFault	BOOL	A subcomponent is faulty
3.1	S_SubMaintenance1	BOOL	A maintenance request is pending for a subcomponent
3.2	S_SubMaintenance2	BOOL	A maintenance demand is pending for a subcomponent
3.3	S_SubDeactivated	BOOL	A subcomponent is deactivated

Address	Name	Data type	Description
3.4	S_Reserved4	BOOL	Reserved
3.5	S_Reserved5	BOOL	Reserved
3.6	S_Reserved6	BOOL	Reserved
3.7	S_Reserved7	BOOL	Reserved
4	S_TIAMS	DWORD	Maintenance state of the component
8	S_TIAMSChannelExist	DWORD	Maintenance state: Configured channels
12	S_TIAMSChannelOK	DWORD	Maintenance state: Faulty channels
16	S_ChannelCount	WORD	Number of channels; valid only when Q_WithSubComponent is set
18.0	S_ChannelVector	ARRAY [0 to 255] BOOL	Number of channels affected; valid only when Q_WithSubComponent is set
+90	Alarm		
0	A_ComponentID	WORD	Hardware ID of the component (internal)
2	A_TextID1	WORD	ID of the first error text
4	A_TextLexikonID1	WORD	ID of the first error text lexicon
6	A_HelpTextLexikonID1	WORD	ID of the first help text lexicon
8	A_MapTextID	WORD	ID of the first error text (HMI)
10	A_MapHelpTextID	WORD	ID of the first help text (HMI)
12	A_TextID2	WORD	ID of the second error text
14	A_TextLexikonID2	WORD	ID of the second error text lexicon
16	A_HelpTextLexikonID2	WORD	ID of the second help text lexicon
18	A_MapTextID2	WORD	ID of the second error text (HMI)
20	A_MapHelpTextID2	WORD	ID of the second help text (HMI)
22	A_AlarmID	DWORD	Alarm number
26	A_ValueCount	WORD	Number of further bytes occupied (12)
28	A_AssociatedValue	ARRAY [1 to n] WORD	Associated values of the alarm n = A_ValueCount / 2 (= 6)
+130	SubComponent		
0	U_SubComponentCount	WORD	Number of subcomponents
2	U_SubComponentFault	ARRAY [1 to n] BYTE	List of subcomponents "n" depends on the configuration **)

*) If the component was deactivated, the index of the requested/actual error is not changed and "E_LastError" is set to "true". The tag area of the alarm is also not filled in.

**) The list of subcomponents is valid only when Q_WithSubComponent is set. The ARRAY has a status byte for each configured component. For a master, the ARRAY contains the status of the configured stations sorted in ascending order according to the station ID. For a station, the ARRAY contains the status of the configured slots sorted in ascending order according to slot number. This field can contain a maximum of 4096 entries (for an IO system): only the actual maximum size is displayed.
The status byte for each lower-level component is defined as follows:

Bit 0 = SubFault: the component cannot be reached
Bit 1 = Fault: the component is not available or is faulty
Bit 2 = Maintenance1: the component has reported a need for maintenance
Bit 3 = Maintenance2: the component has reported a need for maintenance
Bit 4 = Deactivated: the component was deactivated
Bit 5 = SubFault: a subcomponent is faulty
Bit 6 = SubMaintenance1: a subcomponent has reported a need for maintenance
Bit 7 = SubMaintenance2: a subcomponent has reported a need for maintenance

8.2.3.8 Displaying settings of system diagnostics

Requirement

You are in the "Properties" area of the Inspector window of the required PLC.

Procedure

To display and edit the settings for system diagnostics, follow these steps:

1. Click "System diagnostics" in the area navigation of the Inspector window. In the right-hand part of the Inspector window, you can display and edit the various settings.

8.2.3.9 Settings

General settings

Activate system diagnostics for this PLC

This option is deactivated by default.

Note

When this option is cleared, diagnostics data is also not read out in the Web server and diagnostics viewer, even if the option is selected there.

If you clear this option on a PLC on which it was previously selected, all the diagnostics data is deleted the next time you compile the hardware configuration.

Alarm settings

For the alarm categories listed below, you can define whether an alarm is output and whether or not this alarm requires acknowledgment:

- Error
- Maintenance demanded
- Maintenance required
- Info

To do this, select the relevant options in the "Alarm" and "Acknowledgment" columns.

The set alarm class is displayed in the "Alarm class" column.

Settings for diagnostics support

With the help of diagnostics support, diagnostics data is made available in special status DBs by the diagnostics FB. This can then be displayed graphically, for example on an HMI device.

Status DB

Enable the status DB in order to read out the current system status of configured components using this data block.

The following status DB is preset by default:

Block	Name	Number
Diagnostics status DB	RSE_DIAGNOSTIC_STATUS_DB	127

You can, however, assign a different name and/or number for this block, as long as it is not already being used.

If the "Enable Web server on this module" functionality is enabled in the properties of the CPU, the diagnostics status DB must be enabled. As default, DB 127 is generated if the CPU supports the Web server and if this is enabled in the properties of the CPU.

Extended diagnostics settings

This query is only possible if the PLC includes the "D_ACT_DP" instruction.

Enable the option "Query for status "activated/deactivated" after PLC startup" if you want the status of slaves to be queried.

Enable the "Send alarm if status changes from/to activated or deactivated" if you want an alarm to be output when there is a status change. To use this option, execute the "D_ACT_DP" instruction in mode 3/4 and select the diagnostics DB call in OB 86 in the "OB configuration" group.

See also

D_ACT_DP: Enable/disable DP slaves (Page 6486)

Settings for system diagnostics blocks

As default, the following values are set for the system diagnostics blocks:

Block	Name	Number
Diagnostics FB	RSE_FB	49
Diagnostics DB	RSE_DB	49
first global DB	RSE_GLOBAL_DB	50
Diagnostics FC	RSE_FC	49

You can, however, assign different numbers and/or names for these blocks as long as they are not already being used.

8.2.4 System diagnostics for S7-1500 PLCs

8.2.4.1 Introduction

Introduction

When a system error occurs, hardware components and devices of other manufacturers (slaves whose properties are set by their GSD file) can trigger a system reaction. The hardware components provide information on the system error that has occurred.

For S7-1500 PLCs, the system diagnostics provides a convenient way to evaluate this diagnostic information and display it in the form of alarms.

In addition, additional diagnostic information allows the error to be specified in more detail, e.g., channel "switch_top", wire break, or measuring range exceeded.

8.2.4.2 Basics of system diagnostics

You can use the system diagnostics to analyze errors in the system and generate alarms with a written error description and an indication of the error location.

These alarms are defined for each component with alarm capability (e.g., channel fault, rack error).

Recommended procedure

Make the settings for the system diagnostics and the structure of the alarms.

You will find detailed information in the sections below.

8.2.4.3 Displaying settings of system diagnostics

Requirement

You are in the "Properties" area of the Inspector window of the required PLC.

Procedure

To display and edit the settings for system diagnostics, follow these steps:

1. Click "System diagnostics" in the area navigation of the Inspector window. In the right-hand part of the Inspector window, you can display and edit the various settings.

8.2.4.4 Settings

General settings

Activate system diagnostics for this PLC

This option is selected by default and cannot be cleared.

Alarm settings

For the alarm categories listed below, you can define whether an alarm is output and whether or not this alarm requires acknowledgment:

- Error
- Maintenance demanded
- Maintenance required
- Info

To do this, select the relevant options in the "Alarm" and "Acknowledgment" columns.

The set alarm class is displayed in the "Alarm class" column.

8.2.5 Using additional diagnostics options

8.2.5.1 Hardware diagnostics

Displaying non-editable and current values of assignable module properties

Showing general properties and system-relevant information for a module

Where do I find the information I need?

The general properties and system-relevant information for a module can be found in the "General" group in the "Diagnostics" folder in the Online and Diagnostics view of the module to be diagnosed.

Structure of the "General" group

The "General" group consists of the following areas:

- Module
- Module information
- Manufacturer information

"Module" area

This area displays the following additional module information for S7-300 and S7-400:

- Version of bootloader

"Module information" area

This area shows the following additional module information for S7-300 and S7-400 that you have specified during the hardware configuration:

- Device name (you specified this in the network view).
- Plant designation
  If a CPU, a slave, an I/O device, or a module in the device supports the assignment of a plant-wide unique plant designation, then this is shown here. (The plant designation clearly identifies parts of the plant according to functional aspects. It is hierarchically structured in accordance with IEC 1346-1). This allows, for instance, the CPU to be identified from the user program. You can read out the plant designation with the "RDSYSST" instruction.
- Location designation
  If a CPU, a slave, an IO device, or a module in the device supports the assignment of a location designation, then this is shown here (the location designation is part of the equipment identifier that identifies the precise location, e.g., a process tag within a process manufacturing plant, similar to a road in a road map).

"Manufacturer information" area

This area displays the following additional module information for S7-300 and S7-400:

- Copyright law entry

Showing the communications properties of a module

Where do I find the information I need?

The communications properties of a module can be found in the "Communication" group of the "Diagnostics" folder in the Online and Diagnostics view of the module to be diagnosed.

This shows the following properties:

- Cycle load due to communication, configured:
  Set value of the "Cycle loading by communications" parameter, i.e., the proportion of the CPU processing power available for communication in each cycle.
- Connection resources, maximum number:
  Specifies the maximum possible number of available connection resources of the module. A certain number of the available connection resources are reserved for programming device and OP communications and on the S7-300 (with the exception of the 318-2 DP CPU) for S7 basic communications. You can however establish other programming device and OP connections in addition; the number of unassigned connection resources is then reduced accordingly.

Show interfaces and interface properties of a module

Where do I find the information I need?

The interfaces and interface properties of a module can be found in the following groups of the "Diagnostics" folder in the Online and Diagnostics view of the module to be diagnosed:

- MPI interface
- MPI/DP interface
- DP interface
- PROFINET interface

"MPI interface" group

This group displays the following data:

- Transmission speed:
  Current transmission rate setting at the MPI interface for the automation system

"MPI/DP interface" group

This group displays the following data:

- Transmission speed:
  Current transmission rate setting for the MPI/DP interface of the automation system
- DP mode:
  Compatibility mode of the MPI/DP interface whose interface type is set to "PROFIBUS". In the case of modules that support the "DPV1" mode, these can assume the values "S7 compatible" and "DPV1".

Note

If you have set the interface type of the MPI/DP interface to "MPI", then "DP mode" does not contain a valid value.

"DP interface" group

This group displays the following data:

- DP mode:
  Compatibility mode of the DP interface. In the case of modules that support the "DPV1" mode, these can assume the values "S7 compatible" and "DPV1".

"PROFINET interface" group

This group is divided into the following areas:

- "Ethernet address" with the "Network connection" and "IP Parameters" subareas
- "Ports"
- "Statistics" (this area is not considered here, because it does not deal with the current values of configurable module properties).

Note

The "Ethernet address" and "Ports" areas have already been described outside the chapter "Using additional diagnostics options" as they apply for S7-1200 and S7-300/S7-400.

Showing the operand areas, organization blocks and system blocks of a CPU

Where do I find the information I need?

The operand areas, organization blocks, and system blocks of a CPU can be found in the "Performance data" group in the "Diagnostics" folder of the Online and Diagnostics view of the module to be diagnosed.

"Performance data" group

The following information for the respective CPU is shown in the "Performance data" group:

- Operand areas:
  The table gives information regarding
  - The operand type, for example process image input, counters
  - The number of operands or size of their areas
  - The available address range of the operands, for example I0.0 to I127.7
- Organization blocks:
  All organization blocks (OBs) including their function
  You can sort the table rows as follows: Click a column heading to arrange the table rows in such a way that the contents of this column are ordered alphabetically starting from the top.
- System blocks:
  All system functions (SFCs) and system function blocks (SFBs)
  You can sort the table rows as follows: Click a column heading to arrange the table rows in such a way that the contents of this column are ordered alphabetically starting from the top.

Displaying properties of the time system of a module

Where do I find the functions I need?

You can find the properties of the time system of a module in the "Set time of day" group in the "Functions" folder of the Online and Diagnostics view.

Structure of the "Set time of day" group

The "Set time of day" group consists of the following areas:

- Area for reading out and setting the time of day (this area is not examined here)
- Time system

Structure of the "Time system" area

The "Time system" area consists of the following subareas:

- "Clock"
- "Synchronization"

"Clock" subarea

The "Clock" subarea consists of the following elements:

- Resolution:
  States the precision with which the calculation and output of the time is performed
- Real-time clock:
  Displays whether the CPU has a real-time clock: "present" if yes; "not present" if no
- Correction factor:
  Assigned correction factor for the realtime clock of the module. The daily deviation of the clock is compensated with the correction factor.

"Synchronization" subarea

Synchronization mechanisms allow creation of synchronicity between the clocks in different sub-systems. The settings for time-of-day synchronization are made when assigning the module parameters.

The integral real-time clock - where present - can thus synchronize the clocks of other modules (master), or it can itself be synchronized by the clocks of other devices (slave), or it can take no part in the synchronization. Synchronization can be performed in the AS or on the MPI.

Displaying current values of dynamic module properties

Displaying current status of the LEDs and the mode selector of a CPU

Where do I find the information I need?

The current status of the LEDs of the mode selector of a CPU can be found in the display area of the "CPU control panel" pane of the "Online tools" task card.

Display area of the "CPU control panel" pane of the "Online Tools" task card

This area contains the following displays:

- Device name and CPU type
- Error (corresponds to the "SF" LED on S7-300 CPUs)
- RUN (corresponds to the "RUN" CPU operating mode)
- STOP (corresponds to the "STOP" CPU operating mode)

- FORCE (corresponds to the "FRCE" LED on the CPU)
- Current position of the mode selector

Note

"RUN-P" may be displayed erroneously for S7-300 CPUs with a mode selector. In this case, the mode selector is set to "RUN".

Showing connection resources and occupied and reserved connections for a CPU

Where do I find the information I need?

The connection resources not yet assigned and the assigned and reserved connections for a CPU can be found in the "Communications" group in the "Diagnostics" folder in the Online and Diagnostics view.

The "Connection resources" area contains the current values of the following dynamic module properties.

- Number of connection resources not yet assigned.
  These are shown under "Not assigned".

  Note

  If connection resources are already reserved for certain types of communication, the unassigned resources cannot always be used for the various connection types.

- PG communication "reserved" and "assigned":
  Number of connection resources reserved/assigned for connections between the module and programming devices.

  Note

  The number of connection resources assigned can be larger than the number of connection resources reserved, since - within the overall maximum number - other free connection resources can be used for programming device communication. On S7-300-CPUs with assignable connection resource parameters, you can increase the number of connection resources for programming device communication at the expense of the S7 basic communication.

- OP communication "reserved" and "assigned":
  Number of connection resources reserved/assigned for connections between the module and HMI devices (TDs, OPs).

  Note

  The number of connection resources assigned can be larger than the number of connection resources reserved, since - within the overall maximum number - other free connection resources can be used for OP communications. On S7-300-CPUs with assignable connection resource parameters, you can increase the number of connection resources for OP communication at the expense of the S7 basic communication.

- S7 communication "reserved" and "assigned":
  This is displayed on S7-400 CPUs, on 318-2 CPUs and on 317-2 PN/DP CPUs. Reserved/assigned connection resources for configured S7 connections intended for exchanging data by calling communications function blocks (SFBs) in the user program. These can be regarded as hard wired connections (for instance CPU-CPU connections). They can be automatically established when starting communications or they can be established and broken off under program control. S7 communication can be performed via MPI, PROFIBUS, and Industrial Ethernet.
- S7 basic communication "reserved" and "assigned":
  This is displayed on S7-300 CPUs (but not on 318-2 CPUs and not on 317-2 PN/DP CPUs). Reserved/assigned connection resources for non-configured connections intended for exchanging data by calling communications functions (SFCs) in the user program. S7 basic communication can only be performed via MPI or within the automation system.
- Other communication "assigned":
  Additional connection resources for which no connection resources are reserved. Examples:
  - Communication by a S7-300 CPU using a CP (can be performed only on CPUs with assignable connection resource parameters)
  - Communication between networks (S7 routing) on S7-400 CPUs

Note

By means of a server function, in the case of assigned connection resources for S7 communication (i.e., if an S7-400 CPU is communicating with an S7-300 CPU via the SFCs "PUT" or "GET") the following resources are displayed for S7-300 CPUs:

- If the CPU has no assignable connection resource parameters: Under "S7 basic communication"
- If the CPU has assignable connection resource parameters: In the case of CPUs 317-2 PN / DP and 318-2 under "S7 communication", for all others under "Other communication"

Displaying statistical data of data transmissions made using the Ethernet interface

Where do I find the information I need?

Statistical data for the data transmissions that took place via the Ethernet interface can be found in the "Statistics" area under the "PROFINET interface" group in the "Diagnostics" folder of the Online and Diagnostics view.

"Statistics" area

- Display
  Displays statistical data for sent and received data packets. This data allows the data transmission quality to be evaluated.
  The start of the recordings of the statistical data is shown in the first line ("Data packets since:"). This time is reset every time the statistical data is reset ("Reset" button).
  The statistical data display is updated continuously.
- "Reset" button
  The "Data packets since" time is reset and the statistical data are reset to "0".

Displaying information about the runtime meter of a CPU

Where do I find the information I need?

Information about the runtime meter can be found in the "Runtime meter" group in the "Diagnostics" folder of the Online and Diagnostics view.

"Runtime meter" group

The "Runtime meter" group displays the following information for the active CPU runtime meter:

- No. of the runtime meter including information about whether it is a 16-bit or 32-bit runtime meter.
- Elapsed operating hours
- Status
- Information on whether or not an overflow has occurred

If the CPU has one or more 32-bit runtime meters, the respective symbol is given a red border when the metered value exceeds 32767 (maximum value for a 16-bit runtime meter). The "Overrun" column shows "no" if the runtime meter has not yet exceeded the maximum value for a 32-bit runtime meter.

The 32-bit runtime meter is set, started, stopped, and read out by the "RTM (Page 6419)" instruction.

Checking a module for defects

Performing channel-specific diagnostics of non-CPU modules

Where are channel-specific diagnostic events of non-CPU modules displayed?

The channel-specific diagnostic events of non-CPU modules are displayed in the "Channel diagnostics" group in the Online and Diagnostics view.

"Channel diagnostics" group

The display consists of the following elements for each event:

- Slot of faulty module
- Number of faulty channel
  Note that the sequence of channel numbers in this display starts at 0.
- Fault type

> **Note**
> **Diagnostic interrupts**
>
> A diagnostic interrupt can be reported to the CPU only if the module has diagnostic interrupt capability and the diagnostic interrupt has been enabled.
>
> The display of the diagnostic interrupt is a snapshot. Sporadic module defects can be identified in the diagnostics buffer of the respective CPU.

If you select a line in the table containing the channel errors, additional information will be provided for this error (if present).

Performing communication diagnostics of PROFINET ports

Where are port-specific errors of PROFINET ports displayed?

The port-specific errors of PROFINET ports are displayed in the "Communication diagnostics" group of the "Diagnostics" folder in the Online and Diagnostics view of the module to be diagnosed.

Structure of the "Communication diagnostics" group

The "Communication diagnostics" group consists of the following areas:

- Table containing the port-specific errors
- "Details" text field
- "Help on selected diagnostic line" text field

Structure of the table containing the port-specific errors

Each line of the error table corresponds to one error.

The table consists of the following columns:

- Name: port no. and port designation, as well as diagnostics icon
- Error: description of the error that occurred on this port

"Details" text field

If you select a line in the error table, the "Detail" text field will contain detailed information (if available) on the corresponding error.

"Help on selected diagnostic line" text field

If you select a line in the error table, the "Help on selected diagnostic line" text field will contain help information (if available) on the corresponding error.

PROFINET IO component diagnostics

Where do I perform PROFINET IO component diagnostics?

You perform the PROFINET IO component diagnostics in the "PROFINET IO diagnostics" group under "PROFINET interface" in the "Diagnostics" folder of the Online and Diagnostics view.

"PROFINET IO diagnostics" group

Manufacturer-related diagnostics texts for the IO device or the selected module of the IO device are displayed in this area.

If the diagnostics can be read by the IO controller, then communication errors and configuration errors (online/offline discrepancies) are displayed here.

Starting NCM S7 diagnostics

Requirements

- An online connection to the module is not necessary.

Procedure

To start the NCM S7 diagnostics function, follow these steps:

1. Open the module in the Online and Diagnostics view.
2. Select the "Special diagnostics" group in the "Functions" folder.
3. Click the "Start special diagnostics" button in the "Special diagnostics" area.

Result

The NCM S7 diagnostics will be started.

Changing the properties of a module or the programming device/PC

Special feature for S7-400 CPU operating mode switchover

Specifying the startup type of an S7-400 CPU for switchover to "RUN" mode

If you want to switch an S7-400 CPU to "RUN" mode, choose one of the following startup types:

- Warm restart
- Cold restart
- Hot restart

Loging onto the CPU for alarms and system diagnostics events

Requirement

- The PG/PC interface allows access to one or more devices in a subnet.

Procedure

1. Open the Online and Diagnostics view of the respective CPU.
2. Select the "Online access" group.
3. Select the "Receive alarms" check box in the "Alarms" area.

Result

During the next online connection, your programming device or PC will be logged on for receiving alarms and system diagnostics events from the relevant CPU.

Note

The Alarm view requires a continuous online connection for every CPU from which alarms are to be received.

Logging on when an online connection already exists

If the online connection to a CPU has already been established, you can still log on or off as follows for receipt of alarms:

- In the "Online" menu via the "Receive alarms" command
- In project tree using the CPU shortcut menu
- In the device view of the hardware configuration using the CPU shortcut menu
- In the network view of the hardware configuration using the CPU shortcut menu

See also

Receiving alarms (Page 2075)

Compressing the memory and copying from RAM to ROM

Where do I find the information I need?

The reorganization of the memory and copying from RAM to ROM can be found in the command area of the "Memory" pane in the "Online Tools" task card.

The reorganization of the memory can also be found in the command area of the "Memory" group in the "Diagnostics" folder in the Online and Diagnostics view of the module to be diagnosed.

Command area of the "Memory" pane on the "Online Tools" task card

This area can contain the following buttons:

- Compress
  Pressing this button reorganizes the memory, to increase the size of the largest continuous free memory area.
- Copy RAM to ROM
  Pressing this button copies the content of the RAM load memory (user program) to a FEPROM:
  - To the memory card inserted into the CPU.
  - To the integral FEPROM.

Note

A button is specifically displayed when the module commands the respective function.

Command area of the "Memory" group in the "Diagnostics" folder of the Online and Diagnostics view

This area contains the following buttons:

- Compress
  Pressing this button reorganizes the memory, to increase the size of the largest continuous free memory area.

Resetting the PROFINET interface parameters of an S7-300 or S7-400 CPU

Requirement

- You are using an S7-300 CPU with any firmware version or an S7-400 CPU with firmware version >= V6.0.
- There is no memory card inserted in the CPU.
- There is an online connection to the CPU whose PROFINET interface parameters you want to reset.
- The CPU is in STOP mode.

Note

If the CPU is still in RUN mode and you start the reset operation, you can place it in STOP mode by answering the security prompt with yes.

Procedure

To reset the PROFINET interface parameters of an S7-300 or S7-400 CPU, follow these steps:

1. Open the Online and Diagnostics view of the CPU.
2. Select the "Reset of PROFINET interface parameters" group in the "Functions" folder.

   Note

   The user interface of the "Reset of PROFINET interface parameters" group of the Online and Diagnostics view depends on how you have opened the Online and Diagnostics view:
   - If you opened it from the project tree with "Update accessible devices", the MAC address of the associated module is known and grayed out.
   - If you opened it from within the project context, you must enter the relevant MAC address. Otherwise, the "Reset" button remains grayed out.

3. Click the "Reset" button.
4. Acknowledge the confirmation prompt with "OK".

Result

The module is switched to STOP mode, if necessary, and the PROFINET interface parameters are reset. This means:

- The Ethernet interface settings are reset.
- The IP address is deleted.

Starting the firmware loader for an S7-300 or S7-400 CP

Requirements

- The CP to be updated supports a firmware update.
- An online connection to the CP is not necessary.

Procedure

To start the firmware loader for an S7-300 or S7-400 CP, follow these steps:

1. Open the CP in the Online and Diagnostics view.
2. Select the "Firmware update" group in the "Functions" folder.
3. Click the "Start firmware loader" button in the "Firmware update" area.

Result

The firmware loader will be started.

Note

Updating the firmware with S7-300 and S7-400 CPs using the firmware loader differs from the standard mechanisms for updating firmware.

Assigning an IP address to a PROFINET IO device

Basic information on assigning an IP address to a PROFINET IO device

Overview

You have two basic options for choosing an IP address:

- You assign the IP address yourself.
- You obtain the IP address from a DHCP server. (This option is not available for all IO devices.) In this case, the client ID, the MAC address, or the device name of the IO device is communicated to the DHCP server, depending on the option selected.

Requirement

The following assumes that the IP address of the IO device can also be obtained from a DHCP server. This is the case, for example, for some S7-300 CPs.

Starting the address assignment via "Accessible devices"

Requirement

- You have opened the Online and Diagnostics view of the PROFINET IO device using "Update accessible devices" (in the project tree) or "Accessible devices..." ("Online" menu).

Procedure for specifying the IP address yourself

To assign the IP address you have specified yourself to the IO device, follow these steps:

1. Open the Online and Diagnostics view of the IO device.
2. Select the "Assign IP address" group in the "Functions" folder.
3. Select the "Use IP parameters" option.
4. Enter the desired IP address.
5. Enter the subnet mask.

6. If a router is to be used, select the "Use router" check box and enter its IP address.
7. Click the "Assign IP address" button.

Procedure for obtaining the IP address from a DHCP server

To assign an IP address specified by a DHCP server to the IO device, follow these steps:

1. Open the Online and Diagnostics view of the IO device.
2. Select the "Assign IP address" group in the "Functions" folder.
3. Select the "Obtain IP address from a DHCP server" option.
4. If the DHCP server is to identify the target device using its client ID, choose the "Client ID" option. Enter the client ID in the entry field of the same name.

   Note

   The client ID is a string with a maximum of 63 characters. Only the following characters may be used: a-z, A-Z, 0-9 and - (hyphen)

5. Alternatively: If the DHCP server is to identify the target device using its MAC address, choose the "MAC address" option.
6. Alternatively: If the DHCP server is to identify the target device using its device name, choose the "Device name" option.

   Note

   If you specify that the DHCP server is to identify the target device using its device name, you must have assigned a device name to the target device beforehand.

7. Click the "Assign IP address" button.

Result

The IP address is permanently assigned to the IO device. It is retained even through a startup or a power failure.

See also

Retentivity of IP address parameters and device names (Page 1450)

Starting the address assignment from the project context

Requirement

- An online connection to the PROFINET IO device exists.
- You have opened the Online and Diagnostics view of the PROFINET IO device from the project context.

Procedure

1. Open the "Functions" folder and the "Assign IP address" group inside this folder. The "Use IP parameters" option is selected.
2. Click the "Accessible devices" button in order to identify the devices that can be accessed via MAC addresses. After the search is complete, select the IO device with the MAC address known to you.
3. Click the "Assign IP address" button.

Result

The IP address is permanently assigned to the IO device. It is retained even through a startup or a power failure.

See also

Retentivity of IP address parameters and device names (Page 1450)

8.2.5.2 Connection diagnostics

Detailed connection diagnostics

Determining the connection details of an NCM connection

Where do I determine the connection details of an NCM connection?

The connection details of an NCM connection can be found in the "Connection details" group. This group is located in the "Diagnostics > Connection information" area of the Inspector window.

When is the "Connection details" group filled in?

The following requirements must be met to fill in the "Connection details" group on the "Connection information" tab:

- There is an online connection to the relevant module.
- You have selected a line in the connection table.

Structure of the "Connection details" group for an NCM connection

In case of an NCM connection, the "Connection details" group consists of the following elements:

- Local connection ID (hex)
- Connection name
- Connection status

- Send status
- Receive status
- Number of bytes sent
- Number of bytes received
- "Special diagnostics" button
  Click this button to start the NCM connection diagnostics.

See also

Starting NCM S7 diagnostics (Page 2159)

Programming the PLC 9

9.1 Creating a user program

9.1.1 Programming basics

9.1.1.1 Operating system and user program

Operating system

Function

The operating system is contained in every CPU and organizes all CPU functions and sequences that are not associated with a specific control task.

The tasks of the operating system, for example, include the following:

- Processing a warm restart
- Updating the process image of the inputs and outputs
- Calling the user program
- Detecting interrupts and calling interrupt OBs
- Detecting and handling errors
- Managing memory areas

The operating system is a component of the CPU and is already installed there upon delivery.

See also

User program (Page 2167)

User program

Function

The user program contains all functions that are necessary for processing your specific automation task.

The tasks of the user program include:

- Checking the requirements for a (warm) restart using startup OBs, for example, limit switch in correct position or safety relay active.
- Processing process data, e.g. linking binary signals, reading in and evaluating analog values, defining binary signals for output, and outputting analog values
- Reaction to interrupts, for example, diagnostic error interrupt if the limit value of an analog expansion module is overshot.
- Error handling in normal program execution

You write the user program and load it into the CPU.

See also

Operating system (Page 2167)

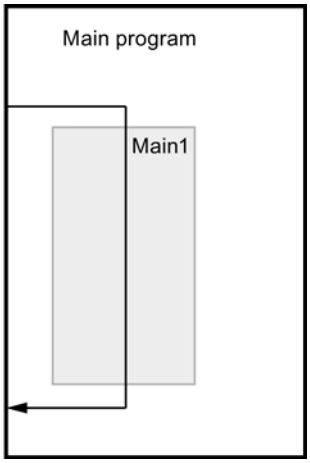
9.1.1.2 Blocks in the user program

Linear and structured programming

Linear programming

Solutions for small automation tasks can be programmed linearly in a program cycle OB. This is only recommended for simple programs.

The following figure shows a linear program schematically: The "Main1" program cycle OB contains the complete user program.



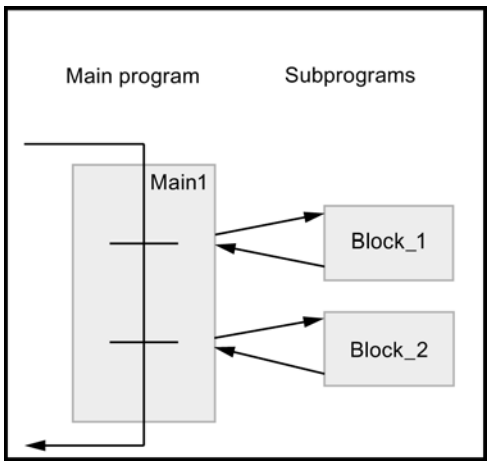
Structured programming

Complex automation tasks can be more easily handled and managed by dividing them into smaller sub-tasks that correspond to the technological functions of the process or that can be reused. These sub-tasks are represented in the user program by blocks. Each block is then an independent section of the user program.

Structuring the program offers the following advantages:

- Extensive programs are easier to program through the structure.
- Individual program sections can be standardized and used repeatedly with changing parameters.
- Program organization is simplified.
- Changes to the program can be made more easily.
- Debugging is simplified since separate sections can be tested.
- Commissioning is simplified.

The following figure shows a structured program schematically: The "Main1" program cycle OB calls subprograms one after the other that execute defined subtasks.



Overview of the block types

Block types

Different BLOCK types are available to perform tasks within an automation system. The following table shows the available block types:

Block type	Brief description
Organization blocks (Page 2170) (OB)	Organization blocks define the structure of the user program.
Functions (Page 2170) (FC)	Functions contain program routines for recurring tasks. They have no "memory".
Function blocks (Page 2171) (FB)	Function blocks are code blocks that store their values permanently in instance data blocks, so that they remain available even after the block has been executed.
Instance data blocks (Page 2174)	Instance data blocks are assigned to a function block when it is is called for the purpose of storing program data.
Global data blocks (Page 2172)	Global data blocks are data areas for storing data that can be used by any blocks.

Organization blocks (OB)

Definition

Organization blocks (OBs) form the interface between the operating system and the user program. They are called by the operating system and control, for example, the following operations:

- Startup characteristics of the automation system
- Cyclic program processing
- Interrupt-driven program execution
- Error handling

You can program the organization blocks and at the same time determine the behavior of the CPU. Various organization blocks are available to you depending on the CPU used.

For more information on organization blocks, refer to the descriptions of the modes of operation of CPUs in the "Additional information on configurations" chapter in "Configuring Hardware and Networks".

Start information of organization blocks

When certain organization blocks are started, the operating system provides information that can be evaluated in the user program. Refer to the descriptions of the organization blocks to find out which information is provided, if any.

See also

Creating organization blocks (Page 2501)

Functions (FCs)

Definition

Functions (FCs) are code blocks without memory. You have no data memory in which values of block parameters can be stored. Therefore, when a function is called, all formal parameters must be assigned actual parameters.
Functions can use global data blocks to store data permanently.

Application

A function contains a program that is executed when the function is called by another code block. Functions can be used, for example, for the following purposes:

- To return function values to the calling block, e.g. for mathematical functions
- To execute technological functions, e.g. individual controls using bit logic operations

A function can also be called several times at different points in a program. As a result, they simplify programming of frequently recurring functions.

Note

Parameter transfer when calling functions

To avoid errors when working with functions, observe the information in chapter "Parameter transfer at block call (Page 2185)".

See also

Creating functions and function blocks (Page 2502)

Function blocks (FB)

Definition

Function blocks are code blocks that store their input, output and in-out parameters permanently in instance data blocks, so that they remain available even after the block has been executed. Therefore they are also referred to as blocks "with memory".

Function blocks can also operate with temporary tags. Temporary tags are will not be stored in the instance DB, but are available for one cycle only.

Application

Function blocks contain subroutines that are always executed when a function block is called by another code block. A function block can also be called several times at different points in a program. As a result, they simplify programming of frequently recurring functions.

Instances of function blocks

A call of a function block is referred to as an instance. An instance data block is required for each instance of a function block; it contains instance-specific values for the formal parameters declared in the function block.

The function block can store its instance-specific data in its own instance data block or in the instance data block of the calling block.

Access modes

S7-1200 and S7-1500 offer two different access options for the instance data blocks, which can be assigned to a function block when this is called:

- Data blocks with optimized access
  Data blocks with optimized access have no firmly defined memory structure. The data elements contain only a symbolic name in the declaration, no fixed address within the block.
- Data blocks with standard access (compatible with S7-300/400)
  Data blocks with standard access have a fixed memory structure. The declaration elements contain both a symbolic name in the declaration and a fixed address within the block.

Note

To avoid errors when working with function blocks, refer to the section "Parameter transfer at block call (Page 2185)".

See also

Creating functions and function blocks (Page 2502)

Multi-instances (Page 2183)

Instance data blocks (Page 2174)

Basics of block access (Page 2175)

Global data blocks (DB)

Definition

Data blocks are used to store program data. Data blocks thus contain variable data that is used by the user program. Global data blocks store data that can be used by all other blocks.

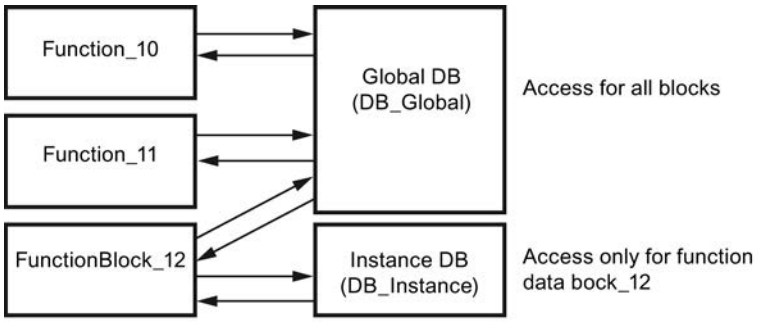
The maximum size of data blocks varies depending on the CPU. You can define the structure of global data blocks anyway you please.

You also have the option of using PLC data types (UDT) as a template for creating global data blocks.

Global data blocks in the user program

Every function block, function, or organization block can read the data from a global data block or can itself write data to a global data block. This data remains in the data block even after the data block is exited. A global data block and an instance data block can be open at the same time.

The following figure shows the different accesses to data blocks:



Access modes

S7-1200 and S7-1500 offer two different access options for global data blocks:

- Data blocks with optimized access
  Data blocks with optimized access have no fixed defined structure. In the declaration, the data elements are assigned only a symbolic name and no fixed address within the block.
- Data blocks with standard access (compatible with S7-300/400)
  Data blocks with standard access have a fixed structure. In the declaration, the data elements are assigned both a symbolic name and a fixed address within the block.

ARRAY data blocks (S7-1500)

ARRAY data blocks are a particular type of global data block. These consist of an ARRAY of any data type. For example, an ARRAY of a PLC data type (UDT) is possible. The DB contains no other elements besides the ARRAY. Because of their flat structure, ARRAY data blocks facilitate access to the ARRAY elements and their transfer to called blocks.

The "Optimized block access" attribute is always enabled for ARRAY data blocks. ARRAY data blocks with standard access are not possible.

The "Move operations" section of the "Instructions" task card offers options for addressing of ARRAY DBs.

See also

Creating data blocks (Page 2503)

Programming PLC data types (Page 2854)

Basics of block access (Page 2175)

Programming data blocks (Page 2825)

Instance data blocks

Definition

The call of a function block is referred to as an instance. The data with which the instance works is stored in an instance data block.

The maximum size of instance data blocks varies depending on the CPU. The tags declared in the function block determine the structure of the instance data block.

Access modes

S7-1200 and S7-1500 offer two different access options for the instance data blocks, which can be assigned to a function block when this is called:

- Data blocks with optimized access
  Data blocks with optimized access have no firmly defined structure. The declaration elements contain only one symbolic name in the declaration, and no fixed address within the block.
- Data blocks with standard access (compatible with S7-300/400)
  Data blocks with standard access have a fixed structure. The declaration elements contain both a symbolic name in the declaration and a fixed address within the block.

See also: Call function blocks as single or multi-instances (Page 2182)

See also

Creating data blocks (Page 2503)

Insert block calls in LAD (Page 2587)

Inserting block calls in FBD (Page 2629)

Basics of block access (Page 2175)

Programming data blocks (Page 2825)

CPU data blocks

Definition

CPU data blocks are generated by the CPU at runtime. To this purpose, insert the "CREATE_DB" instruction into your user program. You can use the data block that is generated at runtime to save your data.

CPU data blocks are indicated by means of a small CPU icon in the "Program blocks" folder of an available node. You can monitor the values of the variables of a CPU data block in online mode, similar to those of a different data block type.

You cannot create CPU data blocks in your offline project.

Loading CPU data blocks

The CPU data block that the user program has generated by means of the "CREATE_DB" instruction is initially only available on the device in online mode. All CPU data blocks will be included with the other blocks the next time you perform a complete download from the device to the project. The CPU data blocks are marked with a small CPU icon in the process. However, you cannot upload these CPU data blocks to your device again.

Restrictions on CPU data blocks in the project

Once the CPU data blocks have been loaded into your offline project, you can open and view their content. However, note that the CPU data blocks in the project are write-protected. The CPU data blocks in the project are therefore subject to the following restrictions:

- You cannot edit CPU data blocks, or convert these into a different data block type.
- CPU data blocks cannot be assigned a know-how protection.
- You cannot change the programming language of a CPU data block.
- CPU data blocks cannot be compiled or downloaded to a device.

Comparing CPU data blocks

Once the CPU data blocks have been loaded into your offline project, you can run an online/offline comparison for the CPU DBs loaded. The comparison editor provides you with a corresponding overview of the differences. It is possible to synchronize the online and off-line version of CPU data blocks if differences are found, but not by downloading the offline version to the device.

Deleting CPU data blocks

You can delete CPU data blocks both from the project and from the CPU.

See also

Deleting CPU data blocks (Page 2522)

Blocks with optimized access

Basics of block access

Introduction

STEP 7 offers data blocks with different access options:

- Data blocks with optimized access (S7-1200/S7-1500)
- Data blocks with standard access (S7-300 / S7-400 / S7-1200 / S7-1500)

Within one program you can combine the two types of blocks.

Data blocks with optimized access

Data blocks with optimized access have no fixed defined structure. In the declaration, the data elements are assigned only a symbolic name and no fixed address within the block. The elements are saved automatically in the available memory area of the block so that there are no gaps in the memory. This makes for optimal use of the memory capacity.

Tags are identified by their symbolic names in these data blocks. To address the tag, enter its symbolic name. For example, you access the "Fill Level" tag in the "Data" DB as follows:
`"Data".Fill Level`

Blocks with optimized access offers the following advantages:

- You can create data blocks with any structure without paying attention to the physical arrangement of the individual data elements.
- Quick access to the optimized data is always available because the data storage is optimized and managed by the system.
- Access errors, as with indirect addressing or from the HMI, for example, are not possible.
- You can define specific individual tags as retentive.
- Optimized blocks are equipped with a memory reserve by default which lets you expand the interfaces of function blocks or data blocks during operation. You can download the modified blocks without setting the CPU to STOP and without affecting the values of already loaded tags.

Note

The "Optimized block access" attribute is always enabled for the following blocks and cannot be deselected.

- GRAPH blocks
- ARRAY data blocks

Data blocks with standard access

Data blocks with standard access have a fixed structure. In the declaration, the data elements are assigned both a symbolic name and a fixed address within the block. The address is shown in the "Offset" column.

Tags in these data blocks can be addressed in both symbolic and absolute form.
`"Data".Fill Level`
`DB1.DBW2`

Setting Retentivity for Optimized Access or Standard Access

If you define data as retentive, its values are retained even after a power failure or a network off. A retentive tag is not initialized after the hot restart but retains the value it had prior to the power failure. If a DB tag is defined as retentive, it is stored in the retentive memory area of the data block.

The options for setting the retentivity depend on the access type of the block.

- In data blocks with standard access, you cannot set the retentive behavior of individual tags. The retentivity setting is valid for all tags of the data block.
- In data blocks with optimized access you can define the retentive behavior of individual tags.
  For structured data type tags, the retentivity setting always applies to the entire structure. You cannot make any individual retentivity setting for separate elements within the data type.

Setting Addressing Options for Optimized Access or Standard Access

Blocks with optimized access permit only "type-safe" access. Type-safe access addresses tags by their symbolic name only. This means even changes to the block or the block interface will not result in inconsistencies in the program or access errors.

The following table shows the permitted addressing options for optimized data:

Addressing	Block with standard access	Block with optimized access
Symbolic addressing	x	x
Indexed addressing of ARRAYs		x
Slice access	x	x
Overlapping with AT	x	-
Absolute addressing	x	-
Indirect addressing via ANY	x	-
Indirect addressing via POINTER and VARIANT	x	with symbolic notation only

See also

Setting up block access (Page 2177)

Addressing variables in data blocks (Page 2211)

Creating blocks (Page 2500)

Setting up block access

Introduction

Block access is set up automatically when you create a block:

- Blocks created on CPUs of the S7-1200/1500 product range provide optimized access by means of a default setting.
- New blocks created on CPUs of the S7-300/S7-400 product range provide standard access by means of a default setting.

Access to a block that you copy or migrate to a CPU of a different product range is not converted automatically. However, in certain situations it may be useful to change block access in manual mode, e.g., in order to utilize the full functional scope of the CPU.

In most cases, you will have to recompile and load the program after block access has been converted.

NOTICE
Optimized block access for GRAPH blocks The "Optimized block access" attribute is always enabled for GRAPH blocks in S7-1500 and cannot be deselected.

Procedure

To set the block access, proceed as follows:

1. Open the "Program blocks" folder in the project tree.
2. Right-click on the block whose block access you want to change.
3. Select the "Properties" command in the shortcut menu.
   The properties dialog box of the block opens.
4. Click "Attributes" in the area navigation.
5. Enable or disable the "Optimized block access" option.
6. Confirm your entries with "OK".

Restrictions and special features

As a matter of principle, you can only convert block access on CPUs of the S7-1200/1500 product range, as only these support the "optimized" access mode.

The following restrictions or special features apply in this context:

- Instance data blocks
  The block access of instance data blocks is always determined by the assigned function block and cannot be changed in manual mode. If you change the access mode on a function block, you also need to update the assigned instance data blocks. This update procedure adapts the access mode of the instance data block.
- System blocks and know-how protected blocks
  You cannot manually edit the block access for system blocks and know-how protected blocks.
- Organization blocks
  The start information of an OB with standard access is always stored in the first 20 bytes of the "Temp" section in the block interface. By contrast, the start information of an OB with optimized access is always written to the "Input" section. For this reason, the block interface of OBs will also change whenever you convert block access. Additional information is provided in the following sections.

Converting block access from "standard" to "optimized".

A block copied from the CPU of the S7-300/400 product range to a CPU of the S7-1200/1500 product range will initially retain the "standard" access mode. However, you can significantly increase the performance of program execution by using blocks with optimized access, which is why it may be useful to modify the access mode manually.

The blocks are adapted as follows in the course of conversion:

- Function blocks
  All interface parameters are assigned the "Non-retain" retentivity setting.
- Global data blocks
  The retentivity setting that was assigned centrally to the entire data block is transferred to the individual interface parameters. It is now possible to manipulate the retentivity setting of the various parameters.
  However, the following rule will still apply: For structured data type tags, the retentivity setting always applies to the entire structure. You cannot assign separate retentivity settings to the various elements within a structured data type. It therefore follows that you cannot assign individual retentivity settings to the tags of data blocks that are based on PLC data types.
- Organization blocks
  All interface parameters that are stored in the first 20 bytes of the "Temp" section will be deleted. New CPU-specific start information is entered in the "Input" section. Naming conflicts with user-defined interface parameters occurring in the process are resolved by renaming the user-defined interface parameters.

CAUTION

The conversion of the block access has the following consequences:

- Absolute addressing of the interface parameters of the block is no longer possible after conversion of block access to the "optimized mode.
  Example: `#L0.1` is no longer valid.
- Since conversion to the "optimized" block access mode of organization blocks also modifies the OB interface,

you may possibly have to adapt, recompile and load the program again due to these changes.

See also: Compiling and downloading blocks (Page 2898)

Converting block access from "optimized" to "standard".

If you want to copy or move a block from the CPU of the S7-300/400 product range to a CPU of the S7-1200/1500 product range, you first need to set the "standard" access mode.

The blocks are adapted as follows in the course of conversion:

- Function blocks and global data blocks.
  You can no longer set a retentivity in the function block. The corresponding setting is made in the instance data block.
  All interface parameters in the instance DB or global DB are assigned the same retentivity setting. The conversion is subject to the following rule:
  - If all interface parameters in the original block were retentive, the entire block will be retentive after conversion.
  - If all interface parameters in the original block were non-retentive, the entire block will be non-retentive after conversion.
  - If the interface parameters in the original block had different retentivity settings, the entire block will be non-retentive after conversion.
- Organization blocks
  All interface parameters stored in the "Input" section will be deleted. New CPU-specific start information is entered in the "Temp" section. This data is written to the first 20 bytes. Naming conflicts with user-defined interface parameters occurring in the process are resolved by renaming the user-defined interface parameters.

CAUTION

The conversion of the block access has the following consequences:

Since a conversion to "standard" block access mode might change the retentivity settings of the interface parameters, you may possibly have to adapt, recompile and load the program again due to these changes.

See also: Compiling and downloading blocks (Page 2898)

See also

Specifying block properties (Page 2512)

Basics of block access (Page 2175)

9.1.1.3 Block calls

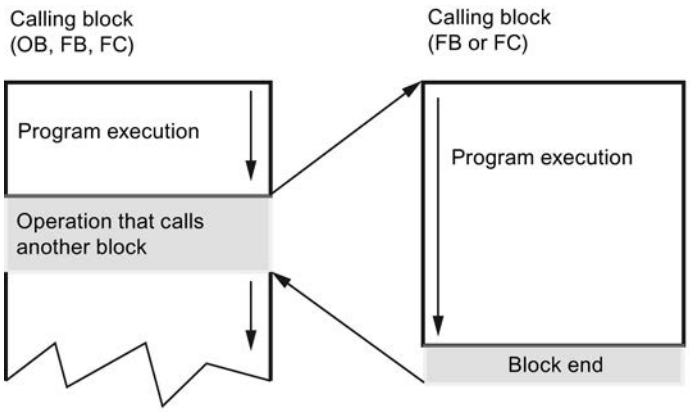
Principles of block calls

Function of block calls

For your blocks to be executed in the user program, they need to be called from another block.

When one block calls another block, the instructions of the called block are executed. Only when execution of the called block has been completed does execution of the calling block resume. The execution is continued with the instruction that follows on the block call.

The following figure shows the sequence of a block call within a user program:



Parameter transfer

When a block is called, you must assign values to the parameters in the block interface. By providing input parameters you specify the data with which the block is executed. By providing the output parameters you specify where the execution results are saved.

See also

Call hierarchy (Page 2181)

Principles for single instances and multi-instances (Page 2182)

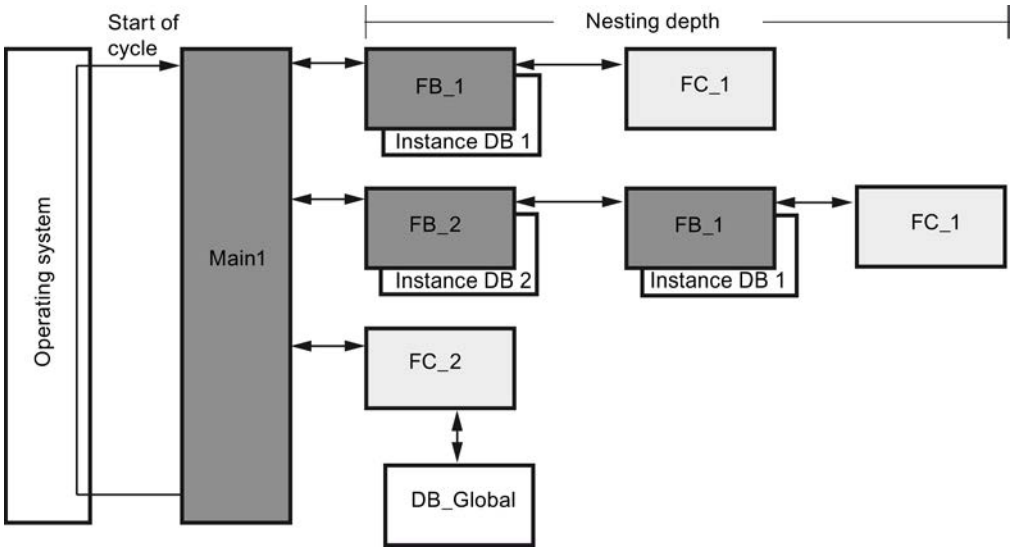
Parameter transfer at block call (Page 2185)

Call hierarchy

Definition

The order and nesting of block calls is referred to as the call hierarchy. The permissible nesting depth depends on the CPU.

The following figure shows an example of the order and nesting of block calls within an execution cycle:



See also

Principles for single instances and multi-instances (Page 2182)

Principles of block calls (Page 2180)

Call function blocks as single or multi-instances

Principles for single instances and multi-instances

Use of single instances and multiple instances

Function blocks (FBs) store their data in instance data blocks. Instance data blocks store the values of the block parameters and the static local data of the function blocks.

You can assign instance data blocks as follows:

- Single instance:
  One instance data block for each instance of a function block
- Multiple instance:
  An instance data block for the instance of a function block and all instances of function blocks called in it.

See also

Addressing variables in data blocks (Page 2211)

Principles of block calls (Page 2180)

Multi-instances (Page 2183)

Single instances (Page 2183)

Call hierarchy (Page 2181)

Single instances

Definition

The call of a function block, which is assigned its own instance data block, is called a single instance data block.

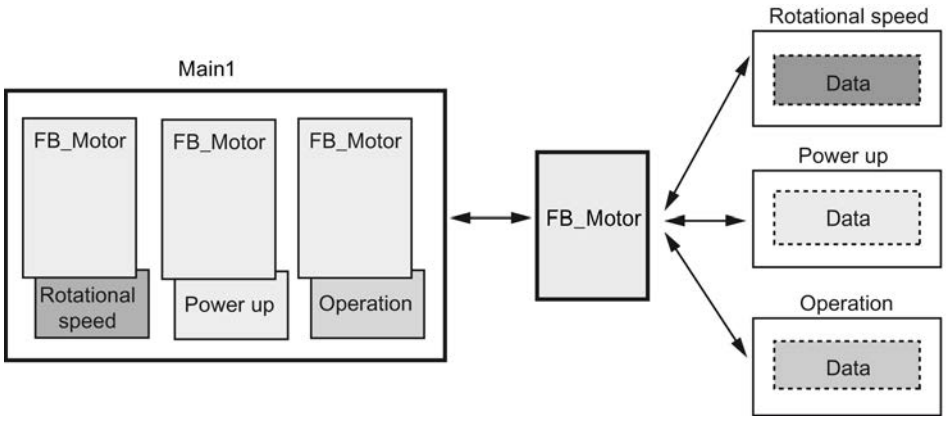
By assignment of the instance data block, you specify where the instance data of the FB is to be stored. By assigning a different instance data block to each call, you can use the same FBs several times with different instance data in each case.

Example of a single instance

You can control several motors using one function block. For this purpose, you assign a different instance data block for each function block call for motor control.

The different data for the various motors, such as speed, ramp-up time, total operating time, are saved in the different instance data blocks. A different motor will be controlled, depending on the instance data block assigned.

The following figure shows the control of three motors using one function block and three different data blocks:



See also

Principles for single instances and multi-instances (Page 2182)

Multi-instances (Page 2183)

Multi-instances

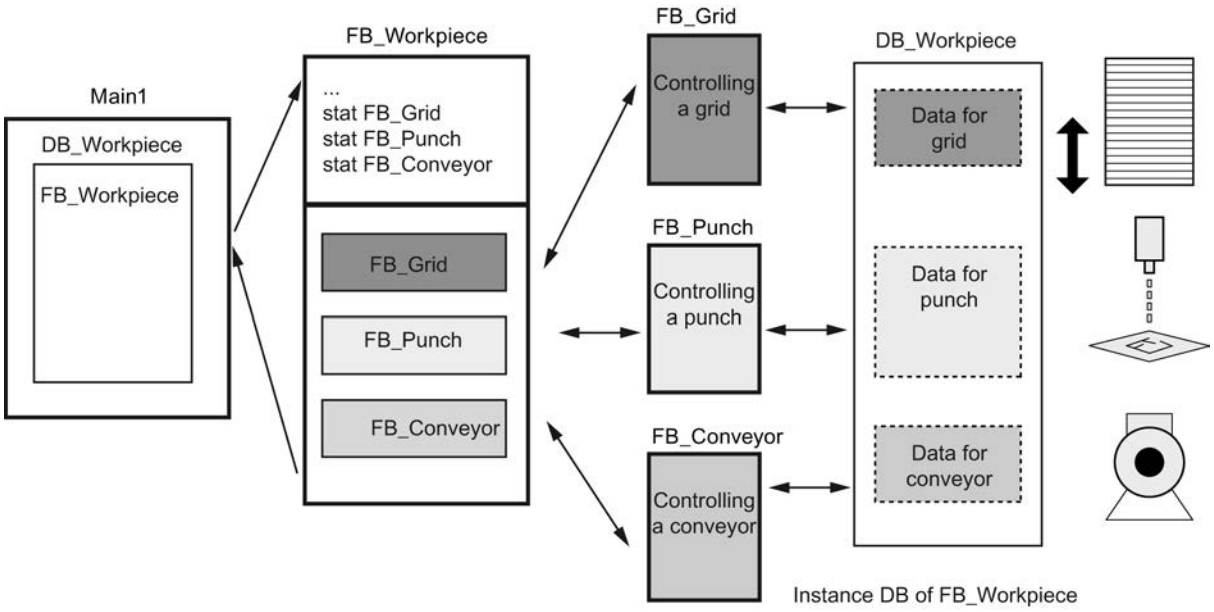
Definition

Multi-instances enable a called function block to store its data in the instance data block of the calling function block.

This allows you to concentrate the instance data in one instance data block and thus make better use of the number of instance data blocks available.

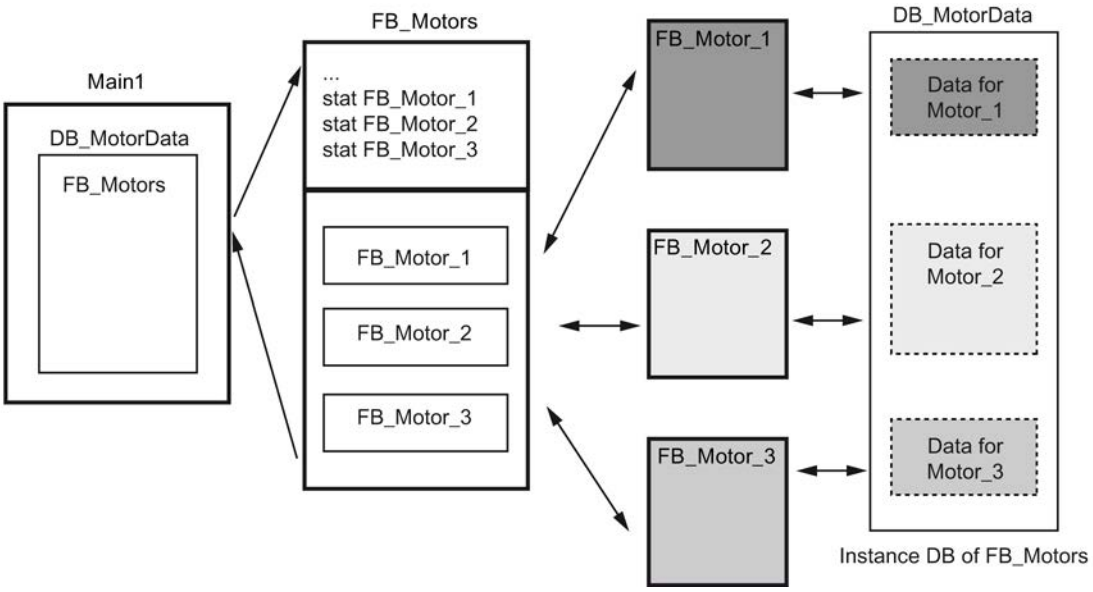
One instance data block for the instances of different function blocks

The following figure shows how multiple different function blocks store their data in a calling block. The FB_Workpiece calls the following on after the other: FB_Grid, FB_Punch and FB_Conveyor. The called blocks store their data in the DB_Workpiece, the instance data block of the calling block.



One instance data block for multi-instances of a function block

The following figure shows how a function block that is called in multi-instances stores the data for all the instances in one instance data block.



The function block FB_Motors calls three instances of the FB_Motor. The instances are "Motor_1", "Motor_2" and "Motor_3". Each call uses different instance data. However, all instance data are located in a single instance data block, DB_MotorData.

See also

Principles for single instances and multi-instances (Page 2182)

Single instances (Page 2183)

Parameter transfer at block call

Basics of block parameters

Introduction

The calling block gives the called block the values with which it is to work. These values are referred to as block parameters. The input parameters provide the called block with the values that it has to process. The block returns the results via the output parameters.

Block parameters are therefore the interface between the calling and the call block.

You use input parameters when you want to only query or read values, and output parameters when you want to set or write these values. If block parameters are read and written you have to create these as in-out parameters.

Formal and actual parameters

The block parameters are defined in the interface of the called block. These parameters are referred to as formal parameters. They are placeholders for the parameters that are transferred to the block when it is called. The parameters transferred to the block when it is called are referred to as actual parameters.

Rules for using the block parameters

The following rules apply to the use of block parameters within the block:

- Input parameters may only be read.
- Output parameters may only be written.
- In/out parameters may be read and written.

See also

Parameter assignment to function blocks (Page 2189)

Parameter assignment to functions (Page 2187)

General rules for assigning parameters (Page 2186)

Using tags within the program (Page 2203)

Keywords (Page 2197)

Rules for declaring the block interface (Page 2547)

Supplying block parameters during call

General rules for assigning parameters

Introduction

When you call a block with block parameters, assign actual parameters to its formal parameters. The rules described below apply here.

Compatible data types

The data types of actual and formal parameters must be identical or convertible according to the rules of data type conversion.

Transferring ARRAYs

You can transfer ARRAYs as parameters. If a block has an input parameter of ARRAY type, you must transfer as actual parameter an ARRAY with identical structure. You can also transfer individual elements of an ARRAY as actual parameter if the element corresponds to the data type of the formal parameter.

Transferring PLC data types

You can also transfer tags that are declared as PLC data type as actual parameters. If the formal parameter is declared as PLC data type in the tag declaration, you must transfer a tag that has the same PLC data type as actual parameter.

An element of a tag declared by means of PLC data type can also be transferred as actual parameter at block call, provided that the data type of the element of the tag matches the data type of the formal parameter.

Transferring structures (STRUCT)

You can transfer structures as parameters. If a block has an input parameter of the STRUCT type, you must transfer as actual parameter a STRUCT with identical structure. This means that the names and data types of all structure components have to be identical.

You can also transfer individual elements of an STRUCT as actual parameter if the element corresponds to the data type of the formal parameter.

Note

We recommend programming structures as PLC data types. PLC data types make programming easier, since they can be used multiple times and modified centrally.

See also

Parameter assignment to function blocks (Page 2189)

Parameter assignment to functions (Page 2187)

Basics of block parameters (Page 2185)

STL Basics (Page 2650)

Fully qualified addresses in STL (Page 223)

Partially qualified addresses in STL (Page 224)

PLC data types (Page 2280)

Parameter assignment to functions

Parameters of functions (FC)

Functions have no data memory in which values of block parameters can be stored. Therefore, when a function is called, all formal parameters must be assigned actual parameters.

Input parameters (Input)

Input parameters are read only once per cycle, namely before the block call. Therefore, the rule is that writing an input parameter within the block does not affect the actual parameter. Only the formal parameter is written.

Output parameters (Output)

Output parameters are read only once per cycle, namely after the block call. Therefore, the rule is that output parameters should not be read within the block. If you nevertheless read an output parameter, please note that only the value of the formal parameter is read. The value of the actual parameter cannot be read within the block.

If an output parameter of a function is not written in this function, the value that is predefined for the specified data type is used. For example, the value "false" is predefined for BOOL. However, structured output parameters are not pre-assigned with a value.

To prevent unintentional further processing of the predefined value or an undefined value, note the following when programming the block:

- Make sure that the output parameters are written with values for all possible program paths within the block. In doing so, note that jump commands may skip instruction sequences in which outputs are set, for example.
- Note that the set and reset commands are dependent on the result of the logic operation. If the value of an output parameter is determined with these commands and RLO = 0, a value will not be generated.
- If possible, assign a default value for the output parameters of functions.

In/out parameters (InOut)

In/out parameters are read before the block call and written after the block call. If you read or write the parameter within the block, you only access its formal parameter.

An exception is in/out parameters with a structured data type. Structured data types consist of several data elements, for example ARRAY or STRUCT. These are passed to the called block through a POINTER. You therefore always access the actual parameter when you read or write a structured in/out parameter within a block.

When an in/out parameter of a function is not written to this function, the old output value or the input value is used as a value. Nevertheless, you should observe the information provided above for output parameters so that old values are not inadvertently processed further.

Temporary local data (Temp)

Temporary local data is only available within a cycle. It is treated differently depending on the block type:

- Standard access
  The following rule applies to code blocks with standard access as well as to all tags with retentivity setting "Set in IDB":
  If you use temporary local data, you have to make sure that the values are written within the cycle in which you want to read them. Otherwise, the values will be random. Temporary data of the STRING data type is an exception: It is automatically pre-assigned the correct length information.
- Optimized access
  The following rule applies to code blocks with optimized access:
  If a temporary tag is not written within a function, the value that is predefined for the specified data type is used. For example, the value "false" is predefined for BOOL. Elements of the PLC data types are pre-assigned with the default value that is specified in the declaration of the PLC data type (UDT). ARRAY elements are pre-assigned with the value "0", even if they are used within a PLC data type. Strings are pre-assigned the correct length information.

Function value (Return)

Functions normally calculate a function value. This function value can be returned to the calling block via the RET_VAL output parameter. For this, the RET_VAL output parameter must be declared in the interface of the function. RET_VAL is always the first output parameter of a function. All data types are permitted for the RET_VAL parameter except ARRAY and STRUCT, as well as parameter types TIMER and COUNTER.

In the SCL programming language functions can be call directly in an expression. The result of the expression is then formed with the calculated function value. Therefore, the data type ANY is not permitted in SCL for the function value.

See also

Parameter assignment to function blocks (Page 2189)

Basics of block parameters (Page 2185)

General rules for assigning parameters (Page 2186)

Calling functions (Page 2707)

Examples for calling functions in SCL (Page 2710)

Parameter assignment to function blocks

Supplying parameters of function blocks (FB)

In the case of function blocks the parameter values will be stored in the instance data.

If the input, output, or in-out parameters of a function block were not assigned with values, the stored values are used.

In some cases, it is mandatory to specify an actual parameter.

The following table shows which parameters of a function block must be assigned actual parameters:

Parameter	Elementary data type	Structured data type	Parameter type
Input (Input)	optional	optional	required
Output (Output)	optional	optional	required
In-out (InOut)	optional	required	Permitted with S7-1200 only, parameter assignment required
Temporary (Temp)	required S7-1500: Optional with optimized block access	required S7-1500: Optional with optimized block access	required

See also

Basics of block parameters (Page 2185)

General rules for assigning parameters (Page 2186)

Parameter assignment to functions (Page 2187)

Parameter types (Page 2279)

Access to block parameters during program execution

Introduction

Block parameters of functions and function blocks are processed differently during program execution. The type of access varies depending on the CPU family, block type and data type of the block parameter.

We generally distinguish between the two following types of access:

- Parameters are transferred as parameter pointers
  A parameter pointer is transferred to the called block.
  This means that the called block can directly access the operand that is specified as actual parameter. Writing a parameter in the called block results in a change of the actual parameter in the calling block. Read access to a block parameter reads the actual parameter directly.
- Parameters are transferred as copy
  The value of the actual parameter is copied to the temporary data of the called blocks during a block call.
  This means that the called block always works with the value that the actual parameter had at the block call. It cannot directly access the operand that is specified as actual parameter. Writing a parameter in the called block does not result in a change of the actual parameter in the calling block. During read access, the formal parameter and not the actual parameter is accessed.
  The procedure for copying is as follows:
  - Input parameter:
    The value of the actual parameter is copied to the formal parameter of the called blocks during a block call.
  - Output parameter:
    The value of the formal parameter is copied to the actual parameter after leaving the block.
  - In/out parameter:
    The value of the actual parameter is copied to the formal parameter of the called block prior to the block call and copied back to the actual parameter after exiting the function.

Parameter transfer for elementary data types

The following table shows how the different CPU families transfer block parameters with elementary data type. Elementary data types are, for example, Bool, Int, Byte.

Called block	Actual parameter in the area	S7-300/400	S7-1500
FC	I, Q, M, P, L	Pointer	Pointer
	Partially qualified DB address (e.g. DBW 2)	Pointer	Partially qualified DB addresses are not supported in S7-1500.
	Fully qualified DB address (e.g. "MyDB".value), forwarded parameters of the caller, static parameters of the caller	Copy	Copy
FB	All actual parameters	Copy	Copy

Parameter transfer for structured data types

The following table shows how the different CPU families transfer block parameters with structured data type. Structured data types are data types that consist of several data elements, e.g. ARRAY or STRUCT.

Called block	Actual parameter in the area	S7-300/400	S7-1500
FC	IN, OUT, InOut	Pointer	Pointer
FB	IN, OUT	Copy	Copy
	InOut	Pointer	Pointer

Forwarding of block parameters

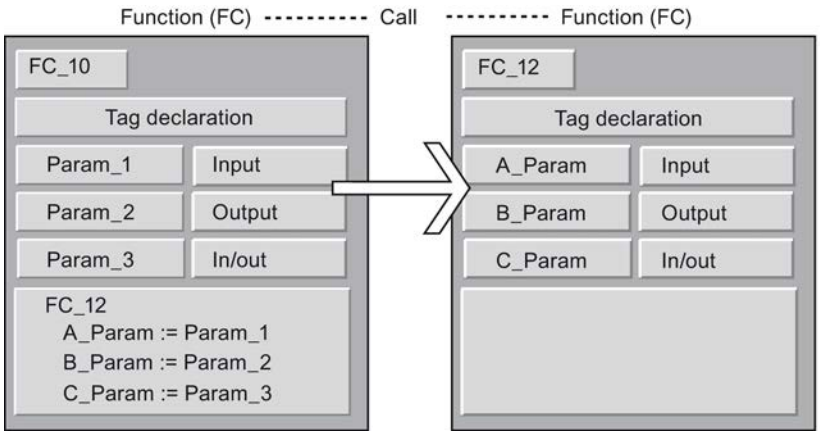
Basic information on forwarding block parameters

Introduction

Definition

The "Forwarding" of block parameters is a special type of parameter use. In this case the block parameters of the calling block are forwarded to the parameters of the called block. The called block uses the values that are currently present at the block parameters of the calling block as the actual parameters.

The following figure shows how the parameters of the function FC_10 are forwarded to the function FC_12:



Rules for LAD/FBD

The following general rules apply in LAD and FBD:

- Input parameters can only be forwarded to input parameters.
- Output parameters can only be forwarded to output parameters.
- In/out parameters can be forwarded to all parameter types.

- In S7-300/400, the two block parameters must have the same data type.
- In S7-1200/1500, the parameters can also be converted according to the rules of implicit conversion.

Rules for STL

The following general rules apply in STL:

- Input parameters can only be forwarded to input parameters.
- Output parameters can only be forwarded to output parameters.
- In/out parameters can be forwarded to all parameter types.
- Both block parameters must have the same data type. In STL, this rule applies to all CPU families.

Rules for SCL

The rules for SCL are less stringent. So that programs from previous SCL versions can be taken over more easily, additional parameter transfer options are permissible, but subject to warning. You can, for example, forward an in/out parameter to an input parameter, but a warning is output as the transferred in/out parameter cannot be written by the program.

Additional rules are described in detail in the following chapters.

See also

Calling a function by another function (Page 2193)

Call of a function by a function block (Page 2194)

Call of a function block by a function (Page 2195)

Call of a function block by another function block (Page 2196)

Calling a function by another function

Permissible data types for the call of a function by another function

Specific rules apply to the forwarding of formal parameters. The following table shows the rules according to which parameters can be forwarded in the various CPU families:

FC calls FC		Data types					
Actual parameter (calling block)	Formal parameters (called block)	Standard data types	ARRAY, STRUCT, STRING, WSTRING, DT	ANY, POINTER	VARIANT	Parameter types (TIMER, COUNTER, BLOCK_XX)	DB_Any

Input	Input	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	S7-1500	S7-1200 S7-1500	S7-1500	S7-1200 as of V2 S7-1500
Output	Output	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	-	S7-1200 S7-1500	-	-
InOut	Input	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	S7-1500	S7-1200 S7-1500	-	-
InOut	Output	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	-	S7-1200 S7-1500	-	-
InOut	InOut	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	S7-1500	S7-1200 S7-1500	-	-

See also

Basic information on forwarding block parameters (Page 2192)

Call of a function by a function block

Permissible data types for the call of a function by a function block

Specific rules apply to the forwarding of formal parameters. The following table shows the rules according to which parameters can be forwarded in the various CPU families:

FB calls FC		Data types					
Actual parameter (calling block)	Formal parameters (called block)	Standard data types	ARRAY, STRUCT, STRING, WSTRING, DT	ANY, POINTER	VARIANT	Parameter types (TIMER, COUNTER, BLOCK_XX)	DB_Any
Input	Input	S7-300/400 S7-1200 S7-1500	S7-300/400 S7-1200 S7-1500	S7-1500	S7-1200 S7-1500	S7-1500	S7-1200 as of V2 S7-1500
Output	Output	S7-300/400 S7-1200 S7-1500	S7-300/400 S7-1200 S7-1500	-	S7-1200 S7-1500	-	-
InOut	Input	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	S7-1500	S7-1200 S7-1500	-	-

InOut	Output	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	-	S7-1200 S7-1500	-	-
InOut	InOut	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	S7-1500	S7-1200 S7-1500	-	-

See also

Basic information on forwarding block parameters (Page 2192)

Call of a function block by a function

Permissible data types for the call of a function block by a function

Specific rules apply to the forwarding of formal parameters. The following table shows the rules according to which parameters can be forwarded in the various CPU families:

FC calls FB		Data types					
Actual parameter (calling block)	Formal parameters (called block)	Standard data types	ARRAY, STRUCT, STRING, WSTRING, DT	ANY, POINTER	VARIANT	Parameter types (TIMER, COUNTER, BLOCK_XX)	DB_Any
Input	Input	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	S7-1500	S7-1200 S7-1500	S7-300/400 S7-1500	S7-1200 as of V2 S7-1500
Output	Output	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	-	S7-1200 S7-1500	-	-
InOut	Input	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	S7-1500	S7-1200 S7-1500	-	-
InOut	Output	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	-	S7-1200 S7-1500	-	-
InOut	InOut	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	S7-1500	S7-1200 S7-1500	-	-

See also

Basic information on forwarding block parameters (Page 2192)

Call of a function block by another function block

Permissible data types for the call of a function block by another function block

Specific rules apply to the forwarding of formal parameters. The following table shows the rules according to which parameters can be forwarded in the various CPU families:

FB calls FB		Data types					
Actual parameter (calling block)	Formal parameters (called block)	Standard data types	ARRAY, STRUCT, STRING, WSTRING, DT	ANY, POINTER	VARIANT	Parameter types (TIMER, COUNTER, BLOCK_XX)	DB_Any
Input	Input	S7-300/400 S7-1200 S7-1500	S7-300/400 S7-1200 S7-1500	S7-1500	S7-1200 S7-1500	S7-300/400 S7-1500	S7-1200 as of V2 S7-1500
Output	Output	S7-300/400 S7-1200 S7-1500	S7-300/400 S7-1200 S7-1500	-	S7-1200 S7-1500	-	-
InOut	Input	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	S7-1500	S7-1200 S7-1500	-	-
InOut	Output	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	-	S7-1200 S7-1500	-	-
InOut	InOut	S7-300/400 S7-1200 S7-1500	S7-1200 S7-1500	S7-1500	S7-1200 S7-1500	-	-

See also

Basic information on forwarding block parameters (Page 2192)

9.1.1.4 Using and addressing operands

Basic information about operands

Introduction

When you program instructions you must specify which data values the instruction should process. These values are referred to as operands. You can, for example, use the following elements as operands:

- PLC tags
- Constants
- Tags in instance data blocks
- Tags in global data blocks

Absolute address and symbolic name

Operands are identified by means of an absolute address and a symbolic name. You define the names and addresses in the PLC tag table or in the tag declaration of the blocks.

Data blocks with optimized access (S7-1200, S7-1500)

Data elements in data blocks with optimized access only receive a symbolic name and no absolute address in the declaration. For more information on this, refer to "See also".

See also

Displaying symbolic and absolute addresses (Page 2533)

Basics of block access (Page 2175)

Keywords

SIMATIC recognizes a range of key words whose definitions are fixed and which have a certain meaning in the program. You should not use these keywords as names for tags or constants.

Table of keywords

The following table shows all keywords.

Keywords German mnemonics	Keywords English mnemonics	Description
&	&	And logical operation of logical expressions
A	Q	Output, bit
A1	CC1	Condition code bit

Keywords German mnemonics	Keywords English mnemonics	Description
A0	CC0	Condition code bit
AB	QB	Output, byte
AD	QD	Output, double word
AND	AND	And logical operation of logical expressions
ANY	ANY	Data type, pointer
AR1	AR1	Address Register 1
AR2	AR2	Address Register 2
ARRAY	ARRAY	Introduces the specification of an array and is followed by the index list between "[" and "]"
AT	AT	Overlaying tag declaration
AUTHOR	AUTHOR	Name of the author, company name, department name, or other name (max. 8 characters, no spaces)
AW	QW	Output, word
B	B	Byte
BEGIN	BEGIN	Introduces the instruction part for code blocks or initialization part for a data block
BIE	BR	Binary result
BLOCK_FB	BLOCK_FB	Parameter type for specification of an FB
BLOCK_FC	BLOCK_FC	Parameter type for specification of an FC
BLOCK_SDB	BLOCK_SDB	Parameter type for specification of an SDB
BOOL	BOOL	Data type
BY	BY	Increment of the FOR loop
BYTE	BYTE	Data type
CALL	CALL	Call
CASE	CASE	Introduction to the CASE statement
CHAR	CHAR	Elementary data type
CODE_VERSION1	CODE_VERSION1	Label, whether an FB is multiple instance capable or not. If you want to declare multiple instances, the FB must not have this characteristic.
CONST	CONST	Start of the constant declaration
CONTINUE	CONTINUE	Instruction to exit a loop in SCL
COUNTER	COUNTER	Parameter type for specification of a counter
DATA_BLOCK	DATA_BLOCK	Introduces the data block
DATE	DATE	Data type

Keywords German mnemonics	Keywords English mnemonics	Description
DATE_AND_TIME	DATE_AND_TIME	Data type
DB	DB	Data block
DB_ANY	DB_ANY	Data type
DBB	DBB	Data block, data byte
DBD	DBD	Data block, data double word
DBLG	DBLG	Data block length
DBNO	DBNO	Data block number
DBW	DBW	Data block, data word
DBX	DBX	Data block, data bit
DI	DI	Instance data block
DIB	DIB	Instance data block, data byte
DID	DID	Instance data block, data double word
DILG	DILG	Instance data block length
DINO	DINO	Instance data block number
DINT	DINT	Data type
DIW	DIW	Instance data block, data word
DIX	DIX	Instance data block, data bit
DO	DO	Introduction of the instruction part in FOR and WHILE instruction
DT	DT	Data type
DTL	DTL	Data type
DWORD	DWORD	Data type
E	I	Input (via process image), bit
EB	IB	Input (via process image), byte
ED	ID	Input (via process image), double word
ELSE	ELSE	Alternative branch in IF and CASE statement
ELSIF	ELSIF	Alternative condition of the IF instruction
EN	EN	System operand of the EN/ENO mechanism
ENO	ENO	System operand of the EN/ENO mechanism
END_CASE	END_CASE	End of the CASE statement
END_DATA_BLOCK	END_DATA_BLOCK	Ends the data block
END_FOR	END_FOR	End of the FOR statement
END_FUNCTION	END_FUNCTION	Ends the function
END_FUNCTION_BLOCK	END_FUNCTION_BLOCK	Ends the function block
END_IF	END_IF	End of the IF instruction
END_ORGANIZATION_BLOCK	END_ORGANIZATION_BLOCK	Ends the organization block
END_REPEAT	END_REPEAT	End of the REPEAT statement

Keywords German mnemonics	Keywords English mnemonics	Description
END_STRUCT	END_STRUCT	Ends the specification of a structure
END_SYSTEM_FUNCTION	END_SYSTEM_FUNCTION	Ends the system function
END_SYSTEM_FUNCTION_BLOCK	END_SYSTEM_FUNCTION_BLOCK	Ends the system function block
END_TYPE	END_TYPE	Ends the PLC data type
END_VAR	END_VAR	Ends a declaration block
END_WHILE	END_WHILE	End of the WHILE instruction
EW	IW	Input (via process image), word
EXIT	EXIT	Instruction to exit a loop in SCL
FALSE	FALSE	Predefined Boolean constant: Logical condition false, value equal to 0
FAMILY	FAMILY	Block family name: e.g. controller
FB	FB	Function block
FC	FC	Function
FOR	FOR	Introduction of the FOR statement
FUNCTION	FUNCTION	Introduces the function
FUNCTION_BLOCK	FUNCTION_BLOCK	Introduces the function block
GOTO	GOTO	Introduction of the GOTO statement
IF	IF	Introduction of the IF instruction
INSTANCE	INSTANCE	Data type
INT	INT	Data type
KNOW_HOW_PROTECT	KNOW_HOW_PROTECT	Block protection
L	L	Local data bit
LB	LB	Local data byte
LD	LD	Local data double word
LDT	LDT	Data type
LINT	LINT	Data type
LTIME	LTIME	Data type
LTOD	LTOD	Data type
LW	LW	Local data word
LWORD	LWORD	Data type
M	M	Memory bit
MB	MB	Memory byte
MD	MD	Memory double word
MOD	MOD	Modulo operator
MW	MW	Memory word
NAME	NAME	Block name
NETWORK	NETWORK	Network
NOT	NOT	Logic inversion

Keywords German mnemonics	Keywords English mnemonics	Description
NULL	NULL	Zero pointer
OB	OB	Organization block
OF	OF	Introduction of the data type specification / Introduction of the instruction part of the CASE statement
OR	OR	Or logical operation of logical expressions
ORGANIZATION_BLOCK	ORGANIZATION_BLOCK	Introduces the organization block
OS	OS	Save overflow
OV	OV	Overflow
PA	PQ	Output (direct peripherals), bit
PAB	PQB	Output (direct peripherals), byte
PAD	PQD	Output (direct peripherals), double word
PAW	PQW	Output (direct peripherals), word
PE	PI	Input (direct peripherals), bit
PEB	PIB	Input (direct peripherals), byte
PED	PID	Input (direct peripherals), double word
PEW	PIW	Input (direct peripherals), word
POINTER	POINTER	Data type
READ_ONLY	READ_ONLY	Write protection for data blocks
REAL	REAL	Data type
REPEAT	REPEAT	Introduction of the REPEAT statement
RET_VAL	RET_VAL	Return value
RETURN	RETURN	RETURN statement in SCL
S5T	S5T	Syntax for data type S5TIME
S5TIME	S5TIME	Data type
S7_	S7_	Keywords for system attributes
SDB	SDB	System data block
SFB	SFB	System function block
SFC	SFC	System function
SINT	SINT	Data type
STRING	STRING	Data type
STRUCT	STRUCT	Introduces the specification of a structure and is followed by a list of components
STW	STW	Status word
SYSTEM_FUNCTION	SYSTEM_FUNCTION	System function
SYSTEM_FUNCTION_BLOCK	SYSTEM_FUNCTION_BLOCK	System function block
T	T	Time element (timer)

Keywords German mnemonics	Keywords English mnemonics	Description
THEN	THEN	Introduction of the instruction part of an IF instruction
THIS	THIS	Syntax for access to an ARRAY data block
TIME	TIME	Elementary data type for time information
TIME_OF_DAY	TIME_OF_DAY	Data type
TIMER	TIMER	Parameter type for specification of a timer
TITLE	TITLE	Optional block title or network title
TO	TO	Definition of the full-scale value of a FOR statement
TOD	TOD	Data type
TRUE	TRUE	Predefined Boolean constant: Logical condition true, value not equal to 0
TYPE	TYPE	Introduction of the PLC data type
UDT	UDT	Global or PLC data type
UDINT	UDINT	Data type
UINT	UINT	Data type
ULINT	ULINT	Data type
UNLINKED	UNLINKED	Marking 'non runtime-related'
UNTIL	UNTIL	End of the instruction part of a REPEAT statement
USINT	USINT	Data type
UO	UO	Query after (Q1=1) AND (Q0=1)
VAR	VAR	Introduces a declaration block
VAR_IN_OUT	VAR_IN_OUT	Introduces a declaration block
VAR_INPUT	VAR_INPUT	Introduces a declaration block
VAR_OUTPUT	VAR_OUTPUT	Introduces a declaration block
VAR_TEMP	VAR_TEMP	Introduces a declaration block
VARIANT	VARIANT	Data type
VERSION	VERSION	Version number of the block
VOID	VOID	Function has no return value
WCHAR	WCHAR	Data type
WSTRING	WSTRING	Data type
WHILE	WHILE	Introduction of a WHILE instruction
WORD	WORD	Data type
XOR	XOR	Logic operation
Z	C	Counter

Using tags within the program

Definition

A variable is a placeholder for a data value that can be changed in the program. The format of the data value is defined. The use of variables makes your program more flexible. For example, you can assign different values to variables that you have declared in the block interface for each block call. As a result, you can reuse a block you have already programmed for various purposes.

A variable consists of the following elements:

- Name
- Data type
- Absolute address
  - PLC tags and DB tags in blocks with standard access have an absolute address.
  - DB variables in blocks with optimized access have no absolute address.
- Value (optional)

Declaring Variables

You can define variables with different scopes for your program:

- PLC tags that apply in all areas of the CPU
- DB variables in global data block that can be used by all blocks throughout the CPU.
- DB tags in instance data blocks that are predominantly used within the block in which they are declared.

The following table shows the difference between the variable types:

	PLC tags	Variables in instance DBs	Variables in global DBs
Range of application	• Are valid throughout the entire CPU. • Can be used by all blocks on the CPU. • The name is unique within the CPU.	• Are predominantly used in the block in which they are defined. • The name is unique within the instance DB.	• Can be used by all blocks on the CPU. • The name is unique within the global DB.
Permissible characters	• Letters, numbers, special characters • Quotation marks are not permitted. • Reserved keywords are not permitted.	• Letters, numbers, special characters • Reserved keywords are not permitted.	• Letters, numbers, special characters • Reserved keywords are not permitted.
Use	• I/O signals (I, IB, IW, ID, Q, QB, QW, QD) • Bit memory (M, MB, MW, MD)	• Block parameters (input, output and in-out parameters), • Static data of a block	• Static data
Location of definition	PLC tag table	Block interface	Declaration table of the global DB

See also

Keywords (Page 2197)

Basic information about operands (Page 2197)

Displaying symbolic and absolute addresses (Page 2533)

Valid names of PLC tags (Page 2475)

Permissible addresses and data types of PLC tags (Page 2476)

Constants

Basics of constants

Definition

Constants are data with a fixed value that you cannot change during program runtime. Constants can be read by various program elements during the execution of the program but cannot be overwritten. There are defined notations for the value of a constant, depending on the data type and data format. A distinction is made between the typed and non-typed notation.

Non-typed constants

In the non-typed notation, you only enter the value of the constant without a data type. Non-typed constants do not receive their type until the first arithmetic or logical operation in which they are used.

The example below shows the non-typed notation:

SCL

```
#My_Int1 := #My_Int2 + 12345        (*The data type of the constant "12345"
                                     results from the addition with My_Int 2. "12345"
                                     receives the data type INT.*)
#My_Real1 := #My_Real2 + 12345      (*The data type of the constant "12345" results
                                     from the addition with My_Real2. "12345"
                                     receives the data type REAL.*)
```

Typed constants

In the typed notation, you specify a data type in addition to the value of the constant.

The example below shows the typed notation:

SCL

```
#My_Int1 :=  INT#12345                    (*The data type of the constant is always
                                          INT.*)
```

Note

Constants of the BOOL type

Constants of the BOOL type may not be used as inputs in LAD or FBD in S7-300/400.

Additional information

Additional information on the data types of constants, their input formats and value ranges:

Data types (Page 2236)

See also

Declaring global constants (Page 2488)

Declaring local tags and constants (Page 2550)

Layout of the block interface (Page 2544)

Declaration of symbolic names for constants

Symbolic constants

You have the option of declaring symbolic names for constants and thus making constant values available under a name in the program. This makes a program more readable and easier to maintain when changing constant values.

A symbolic constant consists of the following elements:

- Name
- Data type
  Symbolic constants always have a data type; non-typed notation is not possible for symbolic constants.
- Constant value
  You can select any value from the value range of the specified data type as constant value. For information on the value ranges, refer to the "Data types" chapter.

Declaration of constants

You can define constants with different scopes of validity:

- Global constants that apply to all areas of the CPU
- Local constants that only apply within a block

The table below shows the difference between the constant types:

	Global constants	Local constants
Scope of validity	• Are valid throughout the entire CPU • The name is unique within the CPU.	• Are valid only in the block in which they were declared. • The name is unique within the block.
Permitted characters	• The permitted characters in constant names are letters, digits and special characters.	• The permitted characters in constant names are letters, digits and special characters.
Location of definition	"Constants" tab from the PLC tag table	Block interface
Representation	In quotation marks, for example.: `"Glob_Const"`	Prefixed with a number sign, for example: `#Loc_Const`

Note

Downloading constant declarations (S7-300/400)

Local and global constant declarations are not downloaded into the CPU. If you download a program from a device, the constant declarations may no longer be available.

Additional information

Additional information on the procedure for declaring constants:

Data types (Page 2236)

Declaring global constants (Page 2488)

Declaring local tags and constants (Page 2550)

Data types of constants

Permitted data types

For constants, all basic data types as well as all derived data types are permitted:

- Binary numbers
- Bit strings
- Integers
- Floating-point numbers
- Timers
- Date and time
- Character strings

All general rules for explicit and implicit type conversion apply.

Data types of non-typed constants

Non-typed constants do not contain an explicit type specification. They do not receive their type until the first arithmetic or logical operation in which they are used.

The example below shows how non-typed constants are used:

SCL

```
#My_Int1 := #My_Int2 + 12345        (*The data type of the constant "12345"
                                     results from the addition with My_Int 2. "12345"
                                     receives the data type INT.*)
#My_Real1 := #My_Real2 + 12345      (*The data type of the constant "12345" results
                                     from the addition with My_Real2. "12345"
                                     receives the data type REAL.*)
```

Note

STEP 7 always uses the data type with the highest possible precision

Unless the data type of a constant can be clearly defined in an expression, the data type with the highest precision available on the current CPU is used.

Example:
`#My_Real := #My_Int / 3.5`

In this expression an integer tag is combined with a non-typed floating point constant. In S7-300/400 the right part of the assignment is calculated in the REAL format. In S7-1200/1500, calculation is performed using the highest possible precision, which in this case means LREAL. As a result, the assignment to a REAL tag is invalid or generates a warning.

To precisely define the data type of a constant, use the typed notation.

Example:

`#My_Real := #My_Int / REAL#3.5`

Additional information

Additional information on the data types of constants, their input formats and value ranges:

Data types (Page 2236)

See also

Declaring global constants (Page 2488)

Declaring local tags and constants (Page 2550)

Examples of using constants

Use in instructions, assignments and expressions

Constants can be used in place of tags in instructions or assignments. You can also use constants in expressions in SCL. But because constants cannot be written, they may only be used as inputs.

The example below shows possible uses of constants:

SCL

```
#My_Int :=   3;
#My_Real1 := #My_Real2 * 3;
#My_Real1 := #My_Real2 * #My_local_const;
#My_Real1 := #My_Real2 * "My_global_const";
```

Use as a default value

You can also use a constant as the default value of a tag. To do so, enter either the value or the symbolic name of the constant in the "Default value" column of the block interface. The data type of the constant must match the data type of the tag or be convertible with it according to the implicit conversion with IEC check.

Name conflicts can occur if you have declared a local and a global constant with the same symbolic name and have used this doubly specified name as the default value of a tag. In this case, the local constant is automatically used.

Use as an ARRAY limit

Local constants of data type INT or DINT can also be used as ARRAY limits.

The example below shows the use of local constants as ARRAY limits:

SCL

```
Array[My_local_const1..My_local_const1] of REAL
```

Note

Constants as ARRAY limits

Note the following information on the topic of "Constants as ARRAY limits":

- Global constants are not permitted as ARRAY limits.
- Local constants which are used as ARRAY limits cannot be changed if the memory reserve of the block is activated.

See also

Declaring global constants (Page 2488)
Declaring local tags and constants (Page 2550)
Data types (Page 2236)

Addressing operands

Addressing global variables

Addressing global variables

To address a global PLC variable, you can use the absolute address or the symbolic name.

Note

The LWORD, LINT, ULINT, LREAL, LTIME, LTOD and LDT data types can only be addressed by means of their symbolic name.

Addressing global variables in symbolic form

When you use addressing in symbolic form, you enter the variable name from the PLC tag table. The symbolic name of global variables are automatically enclosed in quotation marks.

You address structured tags that are based on a PLC data type with the symbolic name of the PLC tag. You can also indicate the names of the individual components separated by a dot.

Addressing global variables in absolute form

When you use addressing in absolute form, you enter the address of the variables from the PLC tag table. The absolute address uses numerical addresses starting with zero for each operand range. The address identifier % is set automatically as prefix for the absolute address of global tags.

Examples

The following examples show applications of symbolic and absolute addressing:

Addressing	Description
%Q1.0	Absolute address: Output 1.0
%I16.4	Absolute address: Input 16.4
%IW4	Absolute address: Input word 4
"Motor"	Symbolic address "Motor"
"Value"	Symbolic address "Value"
"Structured_Tag"	Symbolic address of a tag that is based on a PLC data type

Addressing	Description
"Structured_Tag".Component	Symbolic address of the component of a structured tag.

See also: Permissible addresses and data types of PLC tags (Page 2476)

See also

Displaying symbolic and absolute addresses (Page 2533)

Declaring PLC tags (Page 2473)

Accessing I/O devices (Page 2210)

Accessing I/O devices

Description

The process image of the CPU is updated once in a cycle. In time-critical applications, however, it can be that the current state of a digital input or output has to be read or transferred more often than once per cycle. For this purpose you can use a suffix for I/O access identifiers on the operand to directly access the I/O.

If you want to read the input directly from the peripherals, use the peripheral inputs memory area (PI) instead of the process input image (I). The peripherals memory area can be read as a bit, byte, word, or double word.

If you want to write directly to the output, use the peripheral output (PQ) memory area instead of the process output image (Q). The peripheral output memory area can be written as a bit, byte, word, or double word.

To read or write a signal directly from a peripheral input, you can add the suffix for I/O access ":P", to the operand.

Components of structured PLC tags can also be addressed with ":P". However, access to the higher-level tag with ":P" is not possible.

WARNING

Direct writing of the I/O

Immediate writing to the I/O can lead to hazardous states, for example when writing multiple times to an output in one program cycle.

Syntax

```
<Operand>:P
```

Example

The following example shows applications of I/O access identifiers:

Addressing	Description
"Motor"	Addresses the "Motor" tag in the process image.
"Motor":P	Addresses the "Motor" tag in the I/O memory area (PI or PQ).
"Structured_Tag".Component	Addresses the component of a structured PLC tag in the process image.
"Structured_Tag".Component:P	Addresses the component of a structured PLC tag in the I/O memory area (PI or PQ).

See also

Addressing global variables (Page 2209)

Addressing variables in data blocks

Addressing variables in global data blocks

Description

Tags in global data blocks can be addressed in symbolic or absolute form. For symbolic addressing, you use the name of the data block and the name of the tag, separated by a dot. The name of the data block is enclosed in quotation marks.

For absolute addressing, you use the number of the data block and the absolute address of the tags in the data block, separated by a dot. The address identifier % is set automatically as prefix for the absolute address.

The S7-1200/1500 provides you with an option of accessing a data block that is not yet known during programming. For this purpose, create a block parameter of data type DB_Any in the block interface of the accessing block. The data block name or data block number is transferred to this parameter during runtime. In order to access the internal tags of the data block, use the name of the block parameter of data type DB_Any and the absolute address of the tag, separated by a dot.

Note

Absolute addressing is not possible for the following tags:

- Tags in blocks with optimized access.
- Tags of data type LWORD, LINT, ULINT, LREAL, LTIME, LTOD and LDT.

Best practice is to use the more convenient symbolic addressing for these tags.

ARRAY data blocks

ARRAY data blocks are a particular type of global data block. These consist of an ARRAY of any data type. For example, an ARRAY of a PLC data type (UDT) is possible.

You address elements in ARRAY data blocks with the help of the keyword "THIS". The index is then specified in square brackets. The index can be a constant as well as a tag. Integers with a width of up to 32 bits are permitted as tags for the index.
Extended options for addressing ARRAY DBs are available in the "Move" section of the "Instructions" task card. These instructions give you the option, for example, to also address the DB name indirectly.

Syntax

```
"<DBname>".TagName
%<DBnumber>.absoluteAddress
#<DBAny_name>.%absoluteAddress
"<ArrayDBname".THIS[#i].<Component>.<ComponentElement>
```

SCL:
```
"<ArrayDBname"."THIS"[#i].<Component>.<ComponentElement>
```

The following table show the possible absolute addresses of tags in data blocks:

Data type	Absolute address	Example	Description
BOOL	%DBn.DBXx.y	%DB1.DBX1.0	Data bit 1.0 in DB1
BYTE, CHAR, SINT, USINT	%DBn.DBBy	%DB1.DBB1	Data bit 1 in DB1
WORD, INT, UINT	%DBn. DBWy	%DB1.DBW1	Data word 1 in DB1
DWORD, DINT, UDINT, REAL, TIME	%DBn.DBDy	%DB1.DBD1	Data double word 1 in DB1

Example

The following examples show the addressing of tags in global data blocks:

Addressing	Description
"Motor".Value	Symbolic addressing of the "Value" tag in the "Motor" global data block.
%DB1.DBX1.0	Absolute addressing of the "DBX1.0" tags in the "DB1" global data block.
#MyDBAny.%DBX30.0	Absolute addressing of the "DBX30.0" tag in the global DB that is transferred at runtime at the "MyDBany" parameter.
"MyARRAY_DB".THIS[#MyIndex].MyComponent.MyComponentElement	Addressing an ARRAY data block. The ARRAY index is specified with the "MyIndex" tag. The ARRAY element has two additional substructures: "MyComponent" and "MyComponentElement".

See also

Addressing structured variables (Page 2214)

Addressing areas of a tag with slice access (Page 2216)

Basics of indirect addressing (Page 2220)

Addressing instance data (Page 2213)

Addressing instance data

Description

You can address data elements from the interface of the current block. These tags are stored in the instance data block.

Note

Tags in blocks with optimized access can only be addressed in symbolic form.

To address a tag from the interface of the current block, enter the character # followed by the symbolic tag name.

You can also access the tags of a multiple instance block. Within the multiple instance block, also use the character # followed by the tag name to address the data. You access the data of the multiple instance block from the calling block using #<Multiple instanceName.TagName>.

Syntax

Use the following syntax for addressing tags in instance data blocks:

```
#<TagName>
#<Multiple instanceName.TagName>
```

Examples

The following examples show the addressing of tags in instance data blocks:

Addressing	Description
#Value	Addressing the "Value" tag in the instance data block.
#On	Addressing the "On" tag within the multiple instance block
#Multi.On	Addressing the "On" tag of the multiple instance block from the calling block

See also

Call function blocks as single or multi-instances (Page 2182)

Addressing variables in global data blocks (Page 2211)

Addressing structured variables (Page 2214)

Addressing areas of a tag with slice access (Page 2216)

Basics of indirect addressing (Page 2220)

Addressing structured variables

Addressing data elements of an ARRAY

You access an element in an ARRAY using the syntax `ArrayName[i,j,k...]`.

The index of the element is specified in square brackets. The index includes an integer value (-2147483648 ... 2147483647) for each ARRAY dimension.

Access errors result when you access an element during runtime which is located outside the declared ARRAY limits. The various CPU families react differently to violations of the ARRAY limits:

- S7-300/400
  - The CPU changes to "STOP" mode.
  - You can program the program execution error OB (OB 85) to prevent this.
  - In SCL, you also have the option of enabling the attribute "Check ARRAY limits" in the block properties. This causes the enable output ENO to be set to FALSE in the case of ARRAY access errors.
- S7-1200
  - The CPU generates a diagnostic buffer entry and remains in "RUN" mode.
- S7-1500
  - The CPU changes to "STOP" mode.
  - You can program the programming error OB (OB 121) to prevent this.
  - You also have the option of programming the local error handling with the instructions "GET_ERROR: Get error locally" or "GET_ERROR_ID: Get error ID locally".

Note

Monitoring ARRAY access errors with ENO

The enable output ENO is not set to the signal state FALSE if the ARRAY limits are violated during execution of an instruction. The only exception is SCL blocks on CPUs of the S7-300/400 series for which the block property "Check ARRAY limits" is set.

See also:

Array (Page 2267)

Indirect indexing of ARRAY components (Page 2222)

Addressing ARRAY data blocks

ARRAY data blocks are a particular form of the ARRAY. ARRAY data blocks are global data blocks that consist of exactly one ARRAY. You address elements in ARRAY data blocks using the following syntax:

```
"<GlobArrayDBname".THIS[#i].<componentname>."<elementname>"
```

SCL:

```
"<GlobArrayDBname"."THIS"[#i].<componentname>."<elementname>".
```

The "Move" section of the "Instructions" task card offers extended options for addressing ARRAY DBs. These instructions give you the option, for example, to also address the DB name indirectly.

Addressing data elements in structures

You access individual elements in a structure using `StructureName.ElementName`.

See also:

Structures (Page 2272)

Addressing data elements of an PLC data type

The syntax `PLCDataTypeName.ElementName` is used to access elements of a PLC data type.

See also:

Programming PLC data types (Page 2854)

Addressing individual characters of a STRING or WSTRING

Use the syntax `StringName[i]` to access an individual character of a STRING or WSTRING tag. The counting index "i" begins with "1". Thus, you access the first character of the string with StringName[1].

You cannot access individual characters of a STRING or WSTRING constant.

Errors result when you access a character during runtime which is located outside the STRING length. The various CPU families react differently to violations of the STRING length:

- S7-300/400
  - If the instruction has the enable output ENO, ENO is set to the signal state FALSE.
  - If the instruction does not have an enable output ENO, violation of the STRING length is not monitored:
    If access takes place outside the STRING, the wrong memory area is accessed.
    If the access takes place outside the data block, the CPU goes to STOP. You can program the program execution error OB (OB 85) to prevent this.
- S7-1200/1500
  - If the access takes place outside the declared length, the CPU goes to STOP. You can choose whether you want to respond to violations with the global error handing of the CPU or with separate local error handling.
  - If the access is located within the declared STRING length but outside the actual STRING length, violation of the STRING length is not monitored and the CPU does not go to STOP. You receive the character '$00' or '$0000' with read access; write access is not executed.
    If the instruction has the enable output ENO, ENO is set to the signal state FALSE.

See also:

Character strings (Page 2263)

Examples:

The following examples show the addressing of structured data type tags:

Addressing	Description
Motor.Value_1x3[2]	Addressing of a one-dimensional array
Motor.Value_2x4[2,4]	Addressing of a two-dimensional array
Motor.Value_4x7[2,4,1,3]	Addressing of a four-dimensional array
Batch_1.Temperature	Addressing of the element "Temperature" in the structure "Batch_1"
Values.Temperature	Addressing of the "Temperature" element in the "Values" tag, which is based on a PLC data type.
STRING[3]	Addresses the third character of the string.

See also

Basics of indirect addressing (Page 2220)

Addressing areas of a tag with slice access

Description

You have the option to specifically address areas within declared tags. You can access areas of the 1-bit, 8-bit, 16-bit, or 32-bit width. The type of access is referred to as "slice access".

Constants cannot be addressed by slice access.

Syntax

The following syntax is used for addressing:

```
<Tag>.X<Bit number>
<Tag>.B<BYTE number>
<Tag>.W<WORD number>
<Tag>.D<DWORD number>
```

The syntax has the following components:

Part	Description
<Tag>	Tag that you access. The tag must be of the "Bit string" data type. In the case of activated IEC check, the access to tags of the "Integer" data type is also possible.
X B W D	ID for the access width "Bit (1Bit)" ID for the access width "Byte (8 Bit)" ID for the access width "Word (16 Bit)" ID for access width "DWord (32-bit)"
<BIT number>	Bit number within <tag> that is accessed. Number 0 accesses the least significant BIT.
<BYTE number>	Byte number within <tag> that is accessed. The number 0 accesses the least significant BYTE.
<WORD number>	Word number within <tag> that is accessed. The number 0 accesses the least significant WORD.
<DWORD number>	DWord number within <tag> that is accessed. The number 0 accesses the least significant DWORD.

Examples

The following examples show the addressing of individual bits:

Addressing	Description
"Engine".Motor.X0 "Engine".Motor.X7	"Motor" is a tag of the BYTE, WORD, DWORD or LWORD data type in the global data block "Engine". X0 addresses the bit address 0, X7 the bit address 7 within "Motor".
"Engine".Speed.B0 "Engine".Speed.B1	"Speed" is a tag of the WORD, DWORD or LWORD data type in the global data block "Engine". B0 addresses the byte address 0, B1 the byte address 1 within "Speed".
"Engine".Fuel.W0 "Engine".Fuel.W1	"FUEL" is a tag of the DWORD or LWORD data type in the global data block "Engine". W0 addresses the word address 0, W1 the word address 1 within "Fuel".
"Engine".Data.D0 "Engine".Data.D1	"Data" is a tag of the LWORD data type in the global data block "Engine".

Addressing	Description
	D0 addresses the double word address 0, D1 the double word address 1 within "Data".

Overlaying tags with AT

Description

To access data areas within a declared tag, you can overlay the declared tags with an additional declaration. This provides you with the option of addressing an already declared tag with a different data type. You can, for example, address the individual bits of a tag of WORD data type with an ARRAY of BOOL.

Rules

The following general rules are valid for tag overlaying:

- Overlaying is possible in S7-1200 and S7-1500 in STL, LAD, FBD and GRAPH.
- SCL supports overlaying in all CPU families.
- Overlaying of tags is possible in the following blocks:
  - In code blocks with standard access
  - In code blocks with optimized access for tags with the retain setting "Set in IDB"
- The data width of the overlaying tag must be equal to or less than that of the overlaid tag.
- It is not possible to overlay tags of the VARIANT and INSTANCE data types.
- Blocks from libraries which are declared as parameters in the interface cannot be overlaid.
- Structured PLC tags that are declared as parameters in the interface cannot be overlaid.

Note

S7-1200/1500: Using AT in FCs

The data widths of the overlaying tag and the overlaid tag must be identical for FCs in S7-1200/1500. If this is not possible in your program, you can also address areas within existing tags with the .X, .B, .W or .D syntax.

See also: Addressing individual areas of a tag (Page 2216)

The following combination rules are also valid:

		Overlaying tag	Overlaid tag			
Elementary	Structured *	Any/Pointer	DB_ANY			
FB	Input	Elementary	x	x		x
		Structured *	x	x	x	x
		Any/Pointer		x		

		Overlaying tag	Overlaid tag			
	Temp	Elementary	x	x		
		Structured	x	x	x	
		Any/Pointer		x		
	Static, Output	Elementary	x	x		x
		Structured	x	x		x
		Any/Pointer				
	InOut	Elementary	x			x
		Structured		x		
		Any/Pointer				
FC	Temp	Elementary	x	x		
		Structured	x	x	x	
		Any/Pointer		x		
	Input, Output, InOut	Elementary (both tags must have the same bit width)	x			x
		Structured		x	x	
		Any/Pointer				
OB	Temp	Elementary	x	x		
		Structured	x	x	x	
		Any/Pointer		x		

* Structured data types consist of several data elements, e.g. ARRAY or STRUCT.

Declaration

To overlay a tag, declare an additional tag directly after the tag that is to be overlaid and identify it with the keyword "AT".

Example

The following figure shows the declaration of an overlaid tag in the interface of a FB:

Input		
MyByte		Byte
AT	AT "MyByte"	Array [0..7] of Bool
AT[0]		Bool
AT[1]		Bool
AT[2]		Bool
AT[3]		Bool
AT[4]		Bool
AT[5]		Bool
AT[6]		Bool
AT[7]		Bool

When a block is called with the shown tag declaration, the "MyByte" tag is assigned. Within the block there are now two options for interpreting the data:

- as a byte
- As one-dimensional ARRAY of BOOL

See also

Setting retentivity (Page 2839)

Addressing operands indirectly

Basics of indirect addressing

Introduction

Indirect addressing offers the option of addressing operands whose address is not calculated until during runtime. With indirect addressing, program sections can be executed several times and a different operand can be used during each run.

WARNING

Risk of access errors

Since operands are only calculated during runtime with indirect addressing, there is a risk that access errors may occur and that the program will operate with incorrect values. In addition, memory areas may inadvertently be overwritten with incorrect values. The automation system can then react in unexpected manner.

Therefore, use indirect addressing only with caution.

Indirect addressing

Basics of indirect addressing

General indirect addressing options in S7-1200 and S7-1500

The following indirect addressing options are available in all programming languages:

- Indirect addressing via pointer
- Indirect indexing of ARRAY components
- Indirect addressing of a data block via DB_ANY data type.

Language-specific options of indirect addressing

The following specific addressing options are also available in the various programming languages:

- In STL, you can address operands indirectly via the address register.
- In SCL, you can read or write a variable memory area with the following instructions:
  - POKE - Write memory address
  - POKE_BOOL - Write memory bit
  - PEEK - Read memory address
  - PEEK_BOOL - Read memory bit
  - POKE_BLK - Write memory area

For a detailed description of these addressing options, refer to "See also".

See also

Addressing variables in global data blocks (Page 2211)

POKE: Write memory address (Page 4213)

POKE_BOOL: Write memory bit (Page 4214)

PEEK: Read memory address (Page 4209)

PEEK_BOOL: Read memory bit (Page 4211)

POKE_BLK: Write memory area (Page 4216)

Indirect addressing via pointer (Page 2221)

Indirect indexing of ARRAY components (Page 2222)

Indirect addressing in STL (Page 2227)

Indirect addressing via pointer

Description

For indirect addressing, a special data format is required that contains the address and possibly also the range and the data type of an operand. This data format is referred to as pointer. The following types of pointers are available to you:

- POINTER (S7-1500)
- ANY (S7-1500, only for blocks with standard access)
- VARIANT (S7-1200/1500)

For more information on the pointer data types, refer to "See also".

Note

In SCL the use of the POINTER is restricted. The only option available is to forward it to the called block.

Example

The following example shows an indirect addressing with an area-internal pointer:

Addressing in STL	Explanation
L P#10.0	// Load pointer (P#10.0) in accumulator 1
T MD20	// Transfer pointer to the operand MD20
L MW [MD20]	// Load MW10 in accumulator 1
....	// Any program
L MD [MD20]	// Load MD10 in accumulator 1
....	// Any program
= M [MD20]	// If RLO=1, set the memory bit M10.0

The pointer P#10.0 is transferred to the operand MD20. If the operand MD20 in square brackets is programmed, this will be replaced in runtime by the address that is contained in the pointer.

See also

Basics of indirect addressing (Page 2220)

Pointer (Page 2273)

Indirect indexing of ARRAY components

Description

For addressing the components of an ARRAY, you can specify tags of the integer data type as well as constants as the index. Integers with a length of up to 32 bits are allowed here. When tags are used, the index is calculated during runtime. You can, for example, use a different index for each cycle in program loops.

Note

When you call a block and transfer an indirectly indexed ARRAY component ("<Data block>".<ARRAY>["i"]) to it as in/out parameter (InOut), you cannot change the value of the index tag [i] while the block is being executed. The value is therefore always written back to the same ARRAY component from which it was read.

Syntax

The following syntax is used for the indirect indexing of a ARRAY:

```
"<Data block>".<ARRAY>["i"] // one-dimensional ARRAY
"<Data block>".<ARRAY>["i"] // one-dimensional ARRAY of STRUCT
"<Data block>".<ARRAY>["i"] // multidimensional ARRAY
"<Data block>".<ARRAY>["i"] // multidimensional ARRAY of STRUCT
```

The syntax has the following components:

Part	Description
Data block	Name of the data block in which the ARRAY is located
ARRAY	Tag of the ARRAY data type
i, j	PLC tags of the integer data type that are used as pointers
a	Additional partial tag of the structure

Examples

The following example shows indirect array indexing of an ARRAY component in STL:

Several axes traverse at different angles. The values for axis number and angle are stored in the two-dimensional ARRAY "control_axis".

You can use the "SEL" instruction to select the components of the "control_axis" ARRAY to be written at the "#out" output parameter.

The axis number is defined by the constants "Constant_Axis_NoX" and "Constant_Axis_NoY"; the angle is defined by the "#Angle" tag.

Addressing in STL

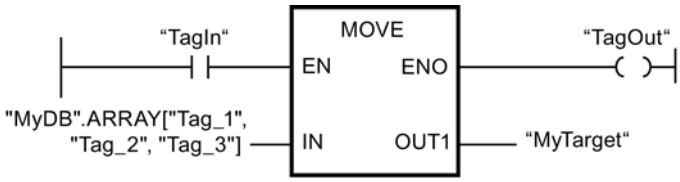
```
CALL  SEL
    value_type:=Int
    G   :="Select"
    IN0 :=#control_axis["Constant_Axis_NoX", #Angle]
    IN1 :=#control_axis["Constant_Axis_NoY", #Angle]
    OUT :=#out
```

The following examples are based on SCL and demonstrate indirect indexing of an ARRAY component. "MOTOR" is a one-dimensional ARRAY_of_INT with three rows. "VALUES" is a PLC tag of data type "Integer".

Addressing in SCL	Explanation
`MOTOR[2] := VALUES;`	`(*Direct addressing: Assignment of VALUES to the second row of the ARRAY MOTOR*)`
`MOTOR["Tag_1"] := VALUES;`	`(*Indirect addressing: Assignment of VALUES to the rows of ARRAY MOTOR*) specified by "Tag_1"`

Addressing in SCL	Explanation
`#MOTOR["Tag_2"+"Tag_3"] := #Values;`	`(*Indirect addressing: Assignment of VALUES to the row of the MOTOR*) ARRAY specified by the expression "Tag_2"+"Tag_3"`

The following example shows the indirect indexing of an ARRAY component as an example of LAD. "ARRAY" is a three-dimensional ARRAY. "Tag_1", " "Tag_2" and ""Tag_3" are PLC tags of the "Integer" data type. Depending on their values, one of the "ARRAY" components will be copied to the "MyTarget" tag.



Indexing ARRAY components using the "FieldRead" and "FieldWrite" instructions

You may also use the following instructions for indirect indexing of ARRAY components in LAD and FBD:

- FieldWrite - Write field
- FieldRead - Read field
  For more information on these instructions, refer to the "References" chapter.

Additional information

For more information on the ARRAY data type, refer to "See also".

See also

Basics of indirect addressing (Page 2220)

Array (Page 2267)

Addressing structured variables (Page 2214)

Indirect addressing of individual characters of a STRING

Description

For addressing the individual characters of a STRING or WSTRING, you can specify constants and also tags as the index. The tags must be of the Integer data type. When tags are used, the index is calculated during runtime. You can, for example, use a different index for each cycle in program loops.

If you transfer a STRING or WSTRING with variable index to an in/out parameter during a block call, please note that: The index tag [i] is read once at the start of the block call and cannot be changed by the called block while it is being executed.

Note

Monitoring STRING access in runtime

When a STRING or WSTRING that exceeds the defined length is written in runtime, unwanted reactions may occur in the program. Violation of the STRING or WSTRING length is monitored in S7-1200/1500. You can choose whether you want to respond to violations with the global error handing of the CPU or with separate local error handling.

Syntax

The following syntax is used for the indirect indexing of a STRING or WSTRING:

```
"<Data block>".<STRING>["i"]
"<Data block>".<WSTRING>["i"]
```

Example

The indirect indexing of a STRING is illustrated below based on the example of SCL. "STRING" is a tag of the STRING data type. "Tag_1" is a PLC tag of the "Integer" data type.

Addressing in SCL	Description
`STRING["Tag_1"] := CHARACTER;`	`(*Indirect addressing: Assignment of "CHARACTER" to the characters of the STRING specified by "Tag_1" *)`

Additional information

For additional information on the STRING data type, refer to "See also".

See also

STRING (Page 2264)

Indirect addressing in STL

Basic information about address registers

Introduction

Two address registers are available for the indirect addressing of operands: address register 1 (AR1), and address register 2 (AR2). The address registers are equal and are 32 bits in length. You can store area-internal and cross-area pointers in the address registers. To define the address of an operand, you can call the stored data in the program.

Data is exchanged between the registers and the other available memory areas with the assistance of load and transfer instructions.

Note

In S7-1500, special rules apply to data exchange via address register and data block register:

- The values in the registers do not remain in existence beyond the block limits.
- The registers are reset when the language is changed within a block.
- You can only reference data in blocks with optimized access if these have the retain setting "Set in IDB".
- It is not possible to reference local data in blocks with optimized access with the help of the address registers (across areas).

Additional information

For more information on the statements that address registers use and on indirect addressing, refer to "See also".

See also

LAR1: Load AR1 with contents of accumulator 1 (Page 5997)

LAR1 <D>: Load AR1 with double word or area pointer (Page 5998)

LAR1 AR2: Load AR1 with contents of AR2 (Page 6000)

LAR2: Load AR2 with contents of accumulator 1 (Page 6001)

CAR: Switch AR1 and AR2 (Page 6006)

TAR1: Transfer AR1 to accumulator 1 (Page 6007)

TAR1 <D>: Transfer AR1 to double word (Page 6008)

TAR1 AR2: Transfer AR1 to AR2 (Page 6009)

TAR2: Transfer AR2 to accumulator 1 (Page 6010)

+AR1: Add accumulator 1 to AR1 (Page 6108)

+AR2: Add accumulator 1 to AR2 (Page 6110)

Indirect addressing in STL (Page 2227)

Load and transfer (Page 3991)

Data exchange in STL (Page 2650)

Addressing areas of a tag with slice access (Page 2216)

Setting retentivity (Page 2839)

Indirect addressing in STL

In STL, the following options are available for indirect addressing:

- Memory-indirect addressing
- Register-indirect area-internal addressing
- Register-indirect cross-area addressing

Memory-indirect addressing

In the case of memory-indirect addressing, you store the address in a tag. The tag can be of WORD or DWORD data type. The tag can be located in the memory areas "Data" (DB or DI), "Bit memory" (M) or "Temporary local data" (L). In S7-1500, FB parameters can also be used to store the address. If the tag is located in a data block, it must be a data black with standard access.

The following example shows applications of memory-indirect addressing:

Addressing in STL	Explanation
`U E [MD 2]`	`// Execute an AND logic operation with a variable input bit. The address of the input bit is located in the memory double word MD2.`
`= DIX [DBD 2]`	`// Assign the RLO to a variable data bit. The address of the data bit is located in the data double word DBD2.`
`L EB [DID 4]`	`// Load a variable input byte to ACCU 1. The address of the input byte is located in the instance double word DID4.`
`AUF DB [LW 2]`	`// Open a variable data block. The number of the data block is located in the local data word LW2.`

Register-indirect area-internal addressing

Register-indirect addressing uses one of the address registers (AR1 or AR2) to pick up the address of the operand.

In the case of register-indirect, area-internal addressing, you index only the bit address and the byte address via the address register (e.g. `P#10.0`). You do not enter the memory area for which the address in the address register is to apply until during programming of the instruction. The address in the address register then moves to the memory area specified in the instruction.

Possible memory areas are "Inputs" (I), "Outputs" (Q), "I/O" (PI or PQ), "Bit memory" (M), "Temporary local data" (L) and "Data" (DB or DI). If the operand is located in a data block, it must be a data black with standard access.

When you enter register-indirect, area-internal addressing, specify an offset after the specification of the address register. This offset is added to the contents of the address register without changing the address register. This offset also has the format of a pointer. The specification of a pointer is mandatory and must be entered as constant (e.g. `P#0.0` or `P#2.0`).

The following example shows an application of register-indirect area-internal addressing:

STL	Explanation
LAR1 P#10.0	// Load pointer (P#10.0) to address register 1
L IW [AR1, P#2.0]	// Increase contents of address register 1 (P#10.0) by offset P#2.0.
	// Load contents of input word IW12 into accumulator 1
L IW [AR1, P#0.0]	// Increase contents of address register 1 (P#10.0) by offset P#0.0.
	// Load contents of input word IW10 into accumulator 1

Register-indirect cross-area addressing

In the case of register-indirect, cross-area addressing, use the address register to index the entire address of the operand, in other words, the bit address and byte address, as well as the memory area. Possible memory areas are "Inputs" (I), "Outputs" (Q), "I/O" (P), "Bit memory" (M), "Temporary local data" (L) and "Data" (DB or DI). If the operand is located in a data block, it must be a data black with standard access or the operand must have the retain setting "Set in IDB".

In the instruction, program only the operand width. Possible operand widths are bit, byte, word, and double word.

The following example shows an application of register-indirect cross-area addressing:

LAR1 P#M10.0	// Load cross-area pointer (P#M10.0) to address register 1
L W [AR1, P#2.0]	// Increase contents of address register 1 (P#M10.0) by offset P#2.0.
	// Load contents of memory word "MW12" into accumulator 1
LAR1 P#A10.0	// Load cross-area pointer (P#A10.0) to address register 1
L W [AR1, P#2.0]	// Add contents of address register 1 (P#A10.0) by offset P#2.0
	// Load contents of output word QW12.0 into accumulator 1

Note

Special features in S7-1500

In S7-1500, special rules apply to data exchange via address register and data block register:

- The values in the registers do not remain in existence beyond the block limits. The registers are also reset when the language is changed within a block.
- If you access an operand of the BYTE, WORD or DWORD type using register-indirect addressing, the address must begin at a byte limit.
  Examples:
  LAR1 P#0.0
  L MW [AR1, P#0.0] // P#0.0 + P#0.0 = P#0.0 - The addressing is allowed, because P#0.0 points to a byte limit.
  L MW [AR1, P#2.1] // P#0.0 + P#2.1 = P#2.1 - The addressing is not allowed, because P#2.1 does not point to a byte limit.

See also

Basics of indirect addressing (Page 2220)
Indirect addressing of data blocks (Page 2230)
Addressing structured variables (Page 2214)
Basic information about address registers (Page 2225)
Data exchange in STL (Page 2650)

Indirect addressing

Indirect addressing via pointer

Description

For indirect addressing, a special data format is required that contains the address and possibly also the range and the data type of an operand. This data format is referred to as pointer. The following types of pointer are available to you:

- POINTER (S7-300/400)
- ANY (S7-300/400)
- VARIANT (S7-1200/1500)

For more information on the pointer data type, refer to "See also".

Note

In SCL the use of the pointer data type is restricted. The only option available is to forward it to the called block.

Example

The following example shows an indirect addressing with an area-internal pointer:

Addressing in STL	Description
L P#10.0	// Load pointer (P#10.0) in accumulator 1
T MD20	// Transfer pointer to the operand MD20
L MW [MD20]	// Load MW10 in accumulator 1
....	// Any program
L MD [MD20]	// Load MD10 in accumulator 1
....	// Any program
= M [MD20]	// If RLO=1, set the memory bit M10.0

The pointer P#10.0 is transferred to the operand MD20. If the operand MD20 in square brackets is programmed, this will be replaced in runtime by the address that is contained in the pointer.

See also

Basics of indirect addressing (Page 2220)

Pointer (Page 2273)

Indirect addressing in SCL

Indirect addressing of data blocks

Indirect addressing of data blocks in SCL

You can use the conversion function WORD_TO_BLOCK_DB to address data blocks indirectly. The DB number is therefore specified as the tag or expression with WORD data type.

Syntax

The following syntax is used for the indirect addressing of a data block:
`WORD_TO_BLOCK_DB(Index).Operand ID (Address)`

Examples

Example 1: The global tag "Address index" of WORD data type is used as the number of the DB.

Addressing in SCL

```
%M0.0:=WORD_TO_BLOCK_DB("Addressindex").DX(0,0);
%MW0:=WORD_TO_BLOCK_DB("Addressindex").DW(4);
```

Example 2: The global tag "Address index" of WORD data type is used as the number of the DB. The data element within the DB is also specified by means of an index:

Addressing in SCL

```
%M0.0:=WORD_TO_BLOCK_DB("Addressindex").DX(#i,#y);
%MW0:=WORD_TO_BLOCK_DB("Addressindex").DW(#y);
```

See also

Basics of indirect addressing (Page 2220)

Indirect addressing in STL (Page 2227)

Addressing structured variables (Page 2214)

Indirect addressing of tags

Indirect addressing of tags in SCL

Indirect addressing is performed similarly to absolute addressing. An offset in round brackets is specified instead of the address. The offset consists of one byte tag or, in the case of Boolean operands, one byte tag and one bit tag. Byte and bit tags must be of data type INT.

Timers and counters from the PLC tag table cannot be addressed indirectly in this way.

Syntax

The following syntax is used for the indirect addressing of a global tag:

```
Operand ID (Byte tag)
Operand ID (Byte tag.Bit tag)
```

The following syntax is used for the indirect addressing of a DB tag:

```
MyDB.OperandID(ByteTag)
MyDB.OperandID(ByteTag.BitTag)
```

Examples

Example 1: One of the input words 2 to 8 is set to 0, depending on the value of run tag i.

Addressing in SCL

```
#i:=2;
FOR #i := 2 TO 8 DO
    %IW(#i) := 0 ;
END_FOR;
```

Example 2: One of the words 2 to 8 in DB10 is set to 0, depending on the value of run tag i.

Addressing in SCL

```
#i:=2;
FOR #i := 2 TO 8 DO
    %DB10.DW(#i) := 0 ;
END_FOR;
```

Indirect addressing of ARRAY components

Description

For addressing the components of an ARRAY, you can enter tags of the integer data type as well as constants as the index. When using tags, the index will be calculated during runtime. You can, for example, use a different index for each cycle in program loops.

Note

The index tag [i] is read once at the start of the block call and cannot be changed by the called block while it is being processed.

When you call a block and transfer an indirectly indexed ARRAY component ("<DataBlock>".<ARRAY>["i"]) to it as in/out parameter (InOut), you cannot change the value of the index tag while the block is being processed. The value is therefore always written back to the same ARRAY component from which it was read.

Syntax

The following syntax is used for the indirect indexing of a ARRAY:

```
"<Data block>".<ARRAY>["i"] // one-dimensional ARRAY
"<Data block>".<ARRAY>["i"] // one-dimensional ARRAY of STRUCT
"<Data block>".<ARRAY>["i"] // multidimensional ARRAY
"<Data block>".<ARRAY>["i"] // multidimensional ARRAY of STRUCT
```

The syntax has the following components:

Part	Description
Data block	Name of the data block in which the ARRAY is located
ARRAY	Tag of the ARRAY data type
i, j	PLC tags of the integer data type that are used as pointers
a	Additional partial tag of the structure

Examples

The following examples are based on SCL and demonstrate indirect indexing of an ARRAY component. MOTOR is a one-dimensional ARRAY_of_INT with three rows. VALUES is a PLC tag of data type "Integer".

Addressing in SCL	Description
`MOTOR[2] := VALUES;`	`(*Direct addressing: Assignment of VALUES to the second row of the ARRAY MOTOR*)`
`MOTOR["Tag_1"] := VALUES;`	`(*Indirect addressing: Assignment of VALUES to the rows of ARRAY MOTOR*) specified by "Tag_1"`

Addressing in SCL	Description
`#MOTOR["Tag_2"+"Tag_3"] := #Values;`	`(*Indirect addressing: Assignment of VALUES to the row of the MOTOR*) ARRAY specified by the expression "Tag_2"+"Tag_3"`

More information

For more information on the ARRAY data type, refer to "See also".

Indirect addressing in STL

Basic information about address registers

Introduction

Two address registers are available for the indirect addressing of operands: address register 1 (AR1), and address register 2 (AR2). The address registers are equal and are 32 bits in length. You can store area-internal and cross-area pointers in the address registers. To define the address of an operand, you can call the stored data in the program.

Data is exchanged between the registers and the other available memory areas with the assistance of load and transfer instructions.

For more information on the statements that address registers use and on indirect addressing, refer to "See also".

See also

LAR1: Load AR1 with contents of accumulator 1 (Page 5997)

LAR1 <D>: Load AR1 with double word or area pointer (Page 5998)

LAR1 AR2: Load AR1 with contents of AR2 (Page 6000)

LAR2: Load AR2 with contents of accumulator 1 (Page 6001)

CAR: Switch AR1 and AR2 (Page 6006)

TAR1: Transfer AR1 to accumulator 1 (Page 6007)

TAR1 <D>: Transfer AR1 to double word (Page 6008)

TAR1 AR2: Transfer AR1 to AR2 (Page 6009)

TAR2: Transfer AR2 to accumulator 1 (Page 6010)

+AR1: Add accumulator 1 to AR1 (Page 6108)

+AR2: Add accumulator 1 to AR2 (Page 6110)

Indirect addressing in STL (Page 2234)

Load and transfer (Page 3991)

Data exchange in STL (Page 2650)

Indirect addressing in STL

In STL, the following options are available for indirect addressing:

- Memory-indirect addressing
- Register-indirect area-internal addressing
- Register-indirect cross-area addressing

Memory-indirect addressing

In the case of memory-indirect addressing, you store the address in a tag. The tag can be of WORD or DWORD data type. The tag can be located in the memory areas "Data" (DB or DI), "Bit memory" (M) or "Temporary local data" (L).

The following example shows applications of memory-indirect addressing:

Addressing in STL	Description
`U E [MD 2]`	`// Execute an AND logic operation with a variable input bit. The address of the input bit is located in the memory double word MD2.`
`= DIX [DBD 2]`	`// Assign the RLO to a variable data bit. The address of the data bit is located in the data double word DBD2.`
`L EB [DID 4]`	`// Load a variable input byte to ACCU 1. The address of the input byte is located in the instance double word DID4.`
`AUF DB [LW 2]`	`// Open a variable data block. The number of the data block is located in the local data word LW2.`

Register-indirect area-internal addressing

Register-indirect addressing uses one of the address registers (AR1 or AR2) to pick up the address of the operand.

In the case of register-indirect, area-internal addressing, you index only the bit address and the byte address via the address register (e.g. `P#10.0`). You do not enter the memory area for which the address in the address register is to apply until during programming of the instruction. The address in the address register then moves to the memory area specified in the instruction.

Possible memory areas are "Inputs" (I), "Outputs" (Q), "I/O" (PI or PQ), "Bit memory" (M), "Temporary local data" (L) and "Data" (DB or DI).

When you enter register-indirect, area-internal addressing, specify an offset after the specification of the address register. This offset is added to the contents of the address register without changing the address register. This offset also has the format of a pointer. The specification of a pointer is mandatory and must be entered as constant (e.g. `P#0.0` or `P#2.0`).

The following example shows an application of register-indirect area-internal addressing:

STL	Description
LAR1 P#10.0	// Load pointer (P#10.0) to address register 1
L IW [AR1, P#2.0]	// Increase content of address register 1 (P#10.0) by offset P#2.0.
	// Load content of input word IW12 into accumulator 1
L IW [AR1, P#0.0]	// Increase content of address register 1 (P#10.0) by offset P#0.0.
	// Load content of input word IW10 into accumulator 1

Register-indirect cross-area addressing

In the case of register-indirect, cross-area addressing, use the address register to index the entire address of the operand, i.e., the bit address and byte address, as well as the memory area. Possible memory areas are "Inputs" (I), "Outputs" (Q), "I/O" (P), "Bit memory" (M), "Temporary local data" (L) and "Data" (DB or DI). In the instruction, program only the operand width. Possible operand widths are bit, byte, word, and double word.

The following example shows an application of register-indirect cross-area addressing:

LAR1 P#M10.0	// Load cross-area pointer (P#M10.0) to address register 1
L W [AR1, P#2.0]	// Increase content of address register 1 (P#M10.0) by offset P#2.0.
	// Load content of memory word "MW12" into accumulator 1
LAR1 P#A10.0	// Load cross-area pointer (P#A10.0) to address register 1
L W [AR1, P#2.0]	// Add content of address register 1 (P#A10.0) by offset P#2.0
	// Load content of output word QW12.0 into accumulator 1

See also

Basics of indirect addressing (Page 2220)

Indirect addressing of data blocks (Page 2230)

Addressing structured variables (Page 2214)

Basic information about address registers (Page 2233)

Data exchange in STL (Page 2650)

9.1.1.5 Data types

Overview of the valid data types

Validity of data type groups

The data type groups define the properties of the data, for example, the representation of the contents and the valid memory areas. In the user program, you can use predefined data type or also data types that you have defined.

The following tables show the availability of predefined data types in the various S7-CPUs:

Table 9-1 Binary numbers

Binary numbers	S7-300/400	S7-1200	S7-1500
BOOL (Page 2240)	X	X	X
Bit strings			
BYTE (Page 2240)	X	X	X
WORD (Page 2241)	X	X	X
DWORD (Page 2242)	X	X	X
LWORD (Page 2243)	-	-	X

Table 9-2 Integers

Integers	S7-300/400	S7-1200	S7-1500
SINT (Page 2244)	-	X	X
INT (Page 2245)	X	X	X
DINT (Page 2247)	X	X	X
USINT (Page 2245)	-	X	X
UINT (Page 2246)	-	X	X
UDINT (Page 2248)	-	X	X
LINT (Page 2249)	-	-	X
ULINT (Page 2250)	-	-	X

Table 9-3 Floating-point numbers

Floating-point numbers	S7-300/400	S7-1200	S7-1500
REAL (Page 2252)	X	X	X
LREAL (Page 2253)	-	X	X

Table 9-4 Timers

Timers	S7-300/400	S7-1200	S7-1500
S5TIME (Page 2255)	X	-	X
TIME (Page 2257)	X	X	X
LTIME (Page 2257)	-	-	X

Table 9-5 Date and time

Date and time	S7-300/400	S7-1200	S7-1500
DATE (Page 2258)	X	X	X
TIME_OF_DAY (TOD) (Page 2259)	X	X	X
LTOD (LTIME_OF_DAY) (Page 2259)	-	-	X
DT (DATE_AND_TIME) (Page 2260)	X	-	X
LDT (Page 2261)	-	-	X
DTL (Page 2262)	-	X	X

Table 9-6 Character

Character	S7-300/400	S7-1200	S7-1500
CHAR (Page 2263)	X	X	X
WCHAR (Page 2263)	-	-	X
STRING (Page 2264)	X	X	X
WSTRING (Page 2265)	-	-	X

Table 9-7 Array

Array	S7-300/400	S7-1200	S7-1500
ARRAY [….] OF <type> (Page 2267)	X	X	X

Table 9-8 Structures

Structures	S7-300/400	S7-1200	S7-1500
STRUCT (Page 2272)	X	X	X

Table 9-9 Pointer

Pointer	S7-300/400	S7-1200	S7-1500
POINTER (Page 2273)	X	-	X
ANY (Page 2275)	X	-	X
VARIANT (Page 2277)	-	X	X

Table 9-10 Parameter types

Parameter types	S7-300/400	S7-1200	S7-1500
TIMER (Page 2279)	X	-	X
COUNTER (Page 2279)	X	-	X
BLOCK_FC (Page 2279)	X	-	X
BLOCK_FB (Page 2279)	X	-	X
BLOCK_DB (Page 2279)	X	-	-
BLOCK_SDB (Page 2279)	X	-	-
BLOCK_SFB (Page 2279)	X	-	-
BLOCK_SFC (Page 2279)	X	-	-

Parameter types	S7-300/400	S7-1200	S7-1500
BLOCK_OB (Page 2279)	X	X	X
VOID (Page 2279)	X	X	X

Table 9-11 PLC data types

PLC data types	S7-300/400	S7-1200	S7-1500
PLC data type (Page 2280)	X	X	X

Table 9-12 System data types

System data types	S7-300/400	S7-1200	S7-1500
IEC_TIMER (Page 2281)	X[1)]	X	X
IEC_LTIMER (Page 2281)	-	-	X
IEC_SCOUNTER (Page 2281)	-	X	X
IEC_USCOUNTER (Page 2281)	-	X	X
IEC_COUNTER (Page 2281)	X[2)]	X	X
IEC_UCOUNTER (Page 2281)	-	X	X
IEC_DCOUNTER (Page 2281)	-	X	X
IEC_UDCOUNTER (Page 2281)	-	X	X
IEC_LCOUNTER (Page 2281)	-	-	X
IEC_ULCOUNTER (Page 2281)	-	-	X
ERROR_STRUCT (Page 2281)	-	X	X
NREF (Page 2281)	-	X	X
CREF (Page 2281)	-	X	X
FBTREF (Page 2281)	-	-	-
VREF (Page 2281)	-	-	-
STARTINFO (Page 2281)	X	-	X
SSL_HEADER (Page 2281)	X	-	X
CONDITIONS (Page 2281)	-	X	X
TADDR_Param (Page 2281)	-	X	X
TCON_Param (Page 2281)	-	X	X
[1)] For S7-300/400 CPUs, the data type is represented by TP, TON and TOF. [2)] For S7-300/400 CPUs, the data type is represented by CTU, CTD and CTUD.			

Table 9-13 Hardware data types

Hardware data types	S7-300/400	S7-1200	S7-1500
REMOTE (Page 2283)	-	X	X
GEOADDR (Page 2283)	-	-	X
HW_ANY (Page 2283)	-	X	X
HW_DEVICE (Page 2283)	-	X	X
HW_DPMASTER (Page 2283)	-	-	X
HW_DPSLAVE (Page 2283)	-	X	X
HW_IO (Page 2283)	-	X	X

Hardware data types	S7-300/400	S7-1200	S7-1500
HW_IOSYSTEM (Page 2283)	-	X	X
HW_SUBMODULE (Page 2283)	-	X	X
HW_MODULE (Page 2283)	-	-	X
HW_INTERFACE (Page 2283)	-	X	X
HW_IEPORT (Page 2283)	-	X	X
HW_HSC (Page 2283)	-	X	X
HW_PWM (Page 2283)	-	X	X
HW_PTO (Page 2283)	-	X	X
AOM_AID (Page 2283)	-	X	X
AOM_IDENT (Page 2283)	-	X	X
EVENT_ANY (Page 2283)	-	X	X
EVENT_ATT (Page 2283)	-	X	X
EVENT_HWINT (Page 2283)	-	X	X
OB_ANY (Page 2283)	-	X	X
OB_DELAY (Page 2283)	-	X	X
OB_TOD (Page 2283)	-	X	X
OB_CYCLIC (Page 2283)	-	X	X
OB_ATT (Page 2283)	-	X	X
OB_PCYCLE (Page 2283)	-	X	X
OB_HWINT (Page 2283)	-	X	X
OB_DIAG (Page 2283)	-	X	X
OB_TIMEERROR (Page 2283)	-	X	X
OB_STARTUP (Page 2283)	-	X	X
PORT (Page 2283)	-	X	X
RTM (Page 2283)	-	X	X
PIP (Page 2283)	-	-	X
CONN_ANY (Page 2283)	-	X	X
CONN_PRG (Page 2283)	-	X	X
CONN_OUC (Page 2283)	-	X	X
CONN_R_ID (Page 2283)	-	-	X
DB_ANY (Page 2283)	-	X	X
DB_WWW (Page 2283)	-	X	X
DB_DYN (Page 2283)	-	X	X

Note

Depending on the CPU version, the actually valid data types can deviate slightly from the table.

Binary numbers

BOOL (bit)

Description

An operand of data type BOOL represents a bit value and contains one of the following values:

- TRUE
- FALSE

The following table shows the properties of data type BOOL:

Length (bits)	Format	Range of values	Examples of value input
1	Boolean	FALSE or TRUE BOOL#0 or BOOL#1 BOOL#FALSE or BOOL#TRUE	TRUE BOOL#1 BOOL#TRUE
	Unsigned integers	0 or 1	1
	Binary numbers	2#0 or 2#1	2#0
	Octal numbers	8#0 or 8#1	8#1
	Hexadecimal numbers	16#0 or 16#1	16#1

Note

Applies to CPUs of the S7-1500 series

For a block with the block property "Optimized block access", the bit has a length of 1 byte.

See also

Overview of the valid data types (Page 2236)

Data type conversion (Page 2286)

Basics of constants (Page 2204)

Bit strings

BYTE (byte)

Description

An operand of data type BYTE is a bit string of 8 bits.

The following table shows the properties of data type BYTE:

Length (bits)	Format	Value range	Examples of value input
8	Unsigned integers[1)]	-128 to +127 or 0 to +255	15, BYTE#15, B#15
	Binary numbers	2#0 to 2#11111111	2#00001111, BYTE#2#00001111, B#2#00001111
	Octal numbers	8#0 to 8#377	8#17, BYTE#8#17, B#8#17
	Hexadecimal numbers	B#16#0 to B#16#FF, 16#0 to 16#FF	16#0F, BYTE#16#0F, B#16#0F
[1)] The value range depends on the relevant interpretation or conversion.			

Note

The BYTE data type cannot be compared for more than or less than. It can only be supplied with the same decimal data that can be processed by the SINT and USINT data types.

See also

Overview of the valid data types (Page 2236)

Data type conversion (Page 2286)

Basics of constants (Page 2204)

WORD

Description

An operand of data type WORD is a bit string of 16 bits.

The following table shows the properties of data type WORD:

Length (bits)	Format	Value range	Examples of value input
16	Unsigned integers	-32768 to 65535	61680, WORD#61680, W#61680
	Binary numbers	2#0 to 2#1111111111111111	2#1111000011110000, WORD#2#1111000011110000, W#2#1111000011110000
	Octal numbers	8#0 to 8#177777	8#170360, WORD#8#170360, W#8#170360
	Hexadecimal numbers	W#16#0 to W#16#FFFF, 16#0 to 16#FFFF	16#F0F0, WORD#16#F0F0, W#16#F0F0
	BCD	C#0 to C#999	C#55
	Decimal sequence	B#(0, 0) to B#(255, 255)	B#(127, 200)

Note

The WORD data type cannot be compared for more than or less than. It can only be supplied with the same decimal data that can be processed by the INT and UINT data types.

See also

Overview of the valid data types (Page 2236)

Data type conversion (Page 2286)

Basics of constants (Page 2204)

DWORD

Description

An operand of data type DWORD is a bit string of 32 bits.

The following table shows the properties of data type DWORD:

Length (bits)	Format	Value range	Examples of value input
32	Unsigned integers	-2147483648 to 4294967295	15793935, DWORD#15793935, DW#15793935
	Binary numbers	2#0 to 2#11111111111111111111111111111111	2#00000000111100001111111100001111, DWORD#2#00000000111100001111111100001111, DW#2#00000000111100001111111100001111
	Octal numbers	8#0 to 8#37777777777	8#74177417, DWORD#8#74177417, DW#8#74177417
	Hexadecimal numbers	DW#16#00000000 to DW#16#FFFFFFFF, 16#00000000 to 16#FFFFFFFF	16#00F0FF0F, DWORD#16#00F0FF0F, DW#16#00F0FF0F
	Decimal sequence	B#(0, 0, 0, 0) to B#(255, 255, 255, 255)	B#(127, 200, 127, 200)

Note

The DWORD data type cannot be compared for more than or less than. It can only be supplied with the same decimal data that can be processed by the DINT and UDINT data types.

See also

Overview of the valid data types (Page 2236)
Basics of constants (Page 2204)
Data type conversion (Page 2286)

LWORD

Description

An operand of data type LWORD is a bit string of 64 bits.

The following table shows the properties of data type LWORD:

Length (bits)	Format	Value range	Examples of value input
64	Unsigned integers	-9223372036854775808 to 18446744073709551615	26123590360715, LWORD#26123590360715, LW#26123590360715
	Binary numbers	2#0 to 2#1111111111111111111111111111111111111111111111111111111111111111	2#0000000000000000000101111100001001011110101001011011011010001011, LWORD#2#0000000000000000000101111100001001011110101001011011101010001011, LW#2#0000000000000000000101111100001001011110101001011011101010001011
	Octal numbers	8#0 to 8#1777777777777777777777	8#13724557213, LWORD#8#13724557213, LW#8#13724557213
	Hexadecimal numbers	LW#16#00000000 to LW#16#FFFFFFFFFFFFFFFF, 16#00000000 to 16#FFFFFFFFFFFFFFFF	16#000000005F52DE8B, LWORD#16#000000005F52DE8B, LW#16#000000005F52DE8B
	Decimal sequence	B#(0, 0, 0, 0, 0, 0, 0, 0) to B#(255, 255, 255, 255, 255, 255, 255, 255)	B#(127, 200, 127, 200, 127, 200, 127, 200)

Note

The LWORD data type cannot be compared for more than or less than. It can only be supplied with the same decimal data that can be processed by the LINT and ULINT data types.

See also

Overview of the valid data types (Page 2236)
Implicit conversion (Page 2288)

Explicit conversion (Page 2322)

Overview of data type conversion (Page 2286)

Basics of constants (Page 2204)

Data type conversion (Page 2286)

Integers

SINT (8-bit integers)

Description

An operand of data type SINT (Short INT) has a length of 8 bits and consists of two components: a sign and a numerical value in the two's complement. The signal states of bits 0 to 6 represent the number value. The signal state of bit 7 represents the sign. The sign may assume "0" for the positive, or "1" for the negative signal state.

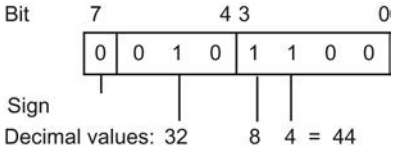
An operand of data type SINT occupies one BYTE in the memory.

The following table shows the properties of data type SINT:

Length (bits)	Format	Range of values	Examples of value input
8	Signed integers	-128 to 127	+44, SINT#+44
	Binary numbers	2#0 to 2#01111111	2#00101100, SINT#2#00101100
	Octal numbers (only positive)	8#0 to 8#177	8#54, SINT#8#54
	Hexadecimal numbers (only positive)	16#0 to 16#7F	16#2C, SINT#16#2C

Example

The following figure shows the integer +44 as a binary number:



See also

Overview of the valid data types (Page 2236)

Data type conversion (Page 2286)

Basics of constants (Page 2204)

USINT (8-bit integers)

Description

An operand of data type USINT (Unsigned Short INT) has a length of 8 bits and contains unsigned numerical values:

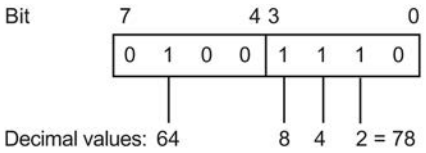
An operand of data type USINT occupies one BYTE in the memory.

The following table shows the properties of data type USINT:

Length (bits)	Format	Range of values	Examples of value input
8	Unsigned integers	0 to 255	78, USINT#78
	Binary numbers	2#0 to 2#11111111	2#01001110, USINT#2#01001110
	Octal numbers	8#0 to 8#377	8#116, USINT#8#116
	Hexadecimal numbers	16#0 to 16#FF	16#4E, USINT#16#4E

Example

The following figure shows the integer 78 as a binary number:



See also

Overview of the valid data types (Page 2236)

Data type conversion (Page 2286)

Basics of constants (Page 2204)

INT (16-bit integers)

Description

An operand of data type INT has a length of 16 bits and consists of two components: a sign and a numerical value in the two's complement. The signal states of bits 0 to 14 represent the number value. The signal state of bit 15 represents the sign. The sign may assume "0" for the positive, or "1" for the negative signal state.

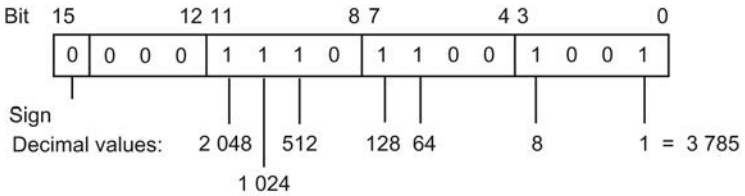
An operand of data type INT occupies two BYTE in the memory.

The following table shows the properties of data type INT:

Length (bits)	Format	Range of values	Examples of value input
16	Signed integers	-32768 to 32767	+3785, INT#+3785
	Binary numbers (only positive)	2#0 to 2#0111111111111111	2#0000111011001001, INT#2#0000111011001001
	Octal numbers	8#0 to 8#77777	8#7311, INT#8#7311
	Hexadecimal numbers (only positive)	16#0 to 16#7FFF	16#0EC9, INT#16#0EC9

Example

The following figure shows the integer +3785 as a binary number:



See also

Overview of the valid data types (Page 2236)

Data type conversion (Page 2286)

Basics of constants (Page 2204)

UINT (16-bit integers)

Description

An operand of data type UINT (Unsigned INT) has a length of 16 bits and contains unsigned numerical values.

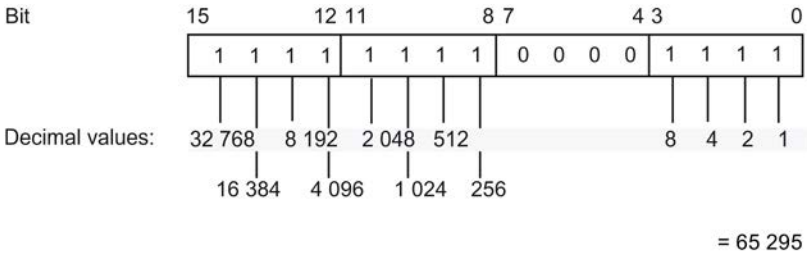
An operand of data type UINT occupies two BYTE in the memory.

The following table shows the properties of data type UINT:

Length (bits)	Format	Range of values	Examples of value input
16	Unsigned integers	0 to 65535	65295, UINT#65295
	Binary numbers	2#0 to 2#1111111111111111	2#1111111100001111, UINT#2#1111111100001111
	Octal numbers	8#0 to 8#177777	8#177417, UINT#8#177417
	Hexadecimal numbers	16#0 to 16#FFFF	16#FF0F, UINT#16#FF0F

Example

The following figure shows the integer 65295 as a binary number:



See also

Overview of the valid data types (Page 2236)

Data type conversion (Page 2286)

Basics of constants (Page 2204)

DINT (32-bit integers)

Description

An operand of data type DINT (Double INT) has a length of 32 bits and consists of two components: a sign and a numerical value in the two's complement. The signal states of bits 0 to 30 represent the number value. The signal state of bit 31 represents the sign. The sign may assume "0" for the positive, or "1" for the negative signal state.

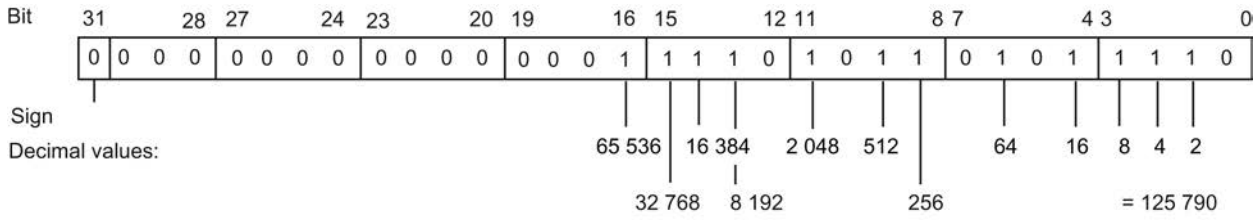
An operand of data type DINT occupies four BYTE in the memory.

The following table shows the properties of data type DINT:

Length (bits)	Format	Range of values	Examples of value input
32	Signed integers	-2147483648 to +2147483647	125790, DINT#125790, L#275
	Binary numbers (only positive)	2#0 to 2#01111111111111111111111111111111	2#00000000000000001111010 1101011110, DINT#2#00000000000000011 110101101011110
	Octal numbers (only positive)	8#0 to 8#17777777777	8#365536, DINT#8#365536
	Hexadecimal numbers	16#00000000 to 16#7FFFFFFF	16#0001EB5E, DINT#16#0001EB5E

Example

The following figure shows the integer +125790 as a binary number:



See also

Overview of the valid data types (Page 2236)

Data type conversion (Page 2286)

Basics of constants (Page 2204)

UDINT (32-bit integers)

Description

An operand of data type UDINT (Unsigned Double INT) has a length of 32 bits and contains unsigned numerical values.

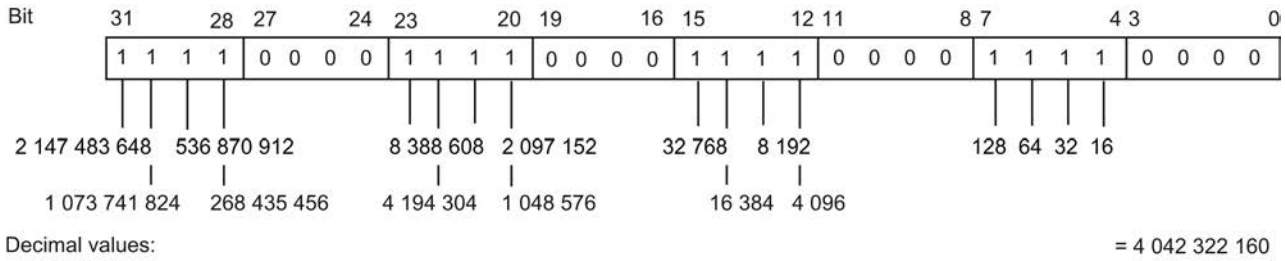
An operand of data type UDINT occupies four BYTE in the memory.

The following table shows the properties of data type UDINT:

Length (bits)	Format	Range of values	Examples of value input
32	Unsigned integers	0 to 4294967295	4042322160, UDINT#4042322160
	Binary numbers	2#0 to 2#11111111111111111111111111111111	2#11110000111100001111000011110000, UDINT#2#11110000111100001111000011110000
	Octal numbers	8#0 to 8#37777777777	8#36074170360, UDINT#8#36074170360
	Hexadecimal numbers	16#00000000 to 16#FFFFFFFF	16#F0F0F0F0, UDINT#16#F0F0F0F0

Example

The following figure shows the integer 4042322160 as a binary number:



See also

Overview of the valid data types (Page 2236)

Data type conversion (Page 2286)

Basics of constants (Page 2204)

LINT (64-bit integers)

Description

An operand of data type LINT (Long INT) has a length of 64 bits and consists of two components: a sign and a numerical value in the two's complement. The signal states of bits 0 to 62 represent the number value. The signal state of bit 63 represents the sign. The sign may assume "0" for the positive, or "1" for the negative signal state.

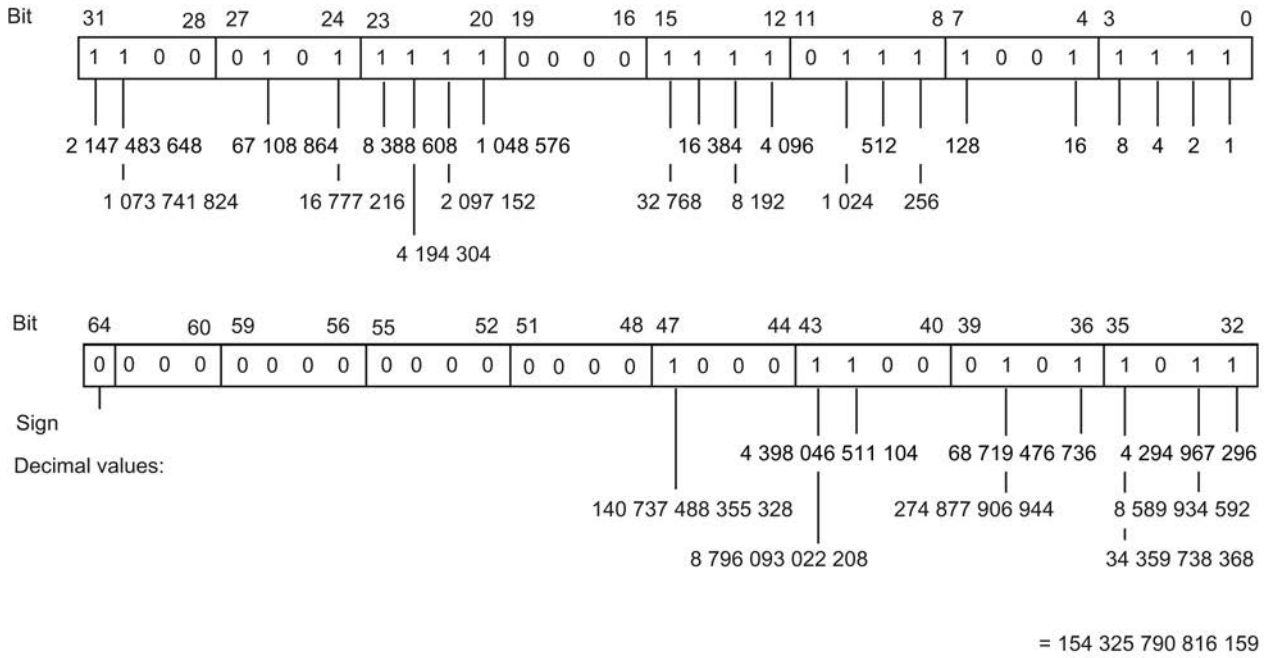
An operand of data type LINT occupies eight BYTE in the memory.

The following table shows the properties of data type LINT:

Length (bits)	Format	Range of values	Examples of value input
64	Signed integers	-9223372036854775808 to +9223372036854775807	+154325790816159, LINT# +154325790816159
	Binary numbers (only positive)	2#0 to 2#0111111111111111111111111111111111111111111111111111111111111111	2#0000000000000000100011000101101111000101111100001111011110011111, LINT#2#0000000000000000100011000101101111000101111100001111011110111110011111
	Octal numbers	8#0 to 8#7777777777777777777	8#4305570574173637, LINT#8#4305570574173637
	Hexadecimal numbers (only positive)	16#0 to 16#7FFFFFFFFFFFFFFF	16#00008C5BC5F0F79F, LINT#16#00008C5BC5F0F79F

Example

The following figure shows the integer +154325790816159 as a binary number:



See also

Overview of the valid data types (Page 2236)

Overview of data type conversion (Page 2286)

Implicit conversion (Page 2288)

Explicit conversion (Page 2322)

Data type conversion (Page 2286)

Basics of constants (Page 2204)

ULINT (64-bit integers)

Description

An operand of data type ULINT (Unsigned Long INT) has a length of 64 bits and contains unsigned numerical values.

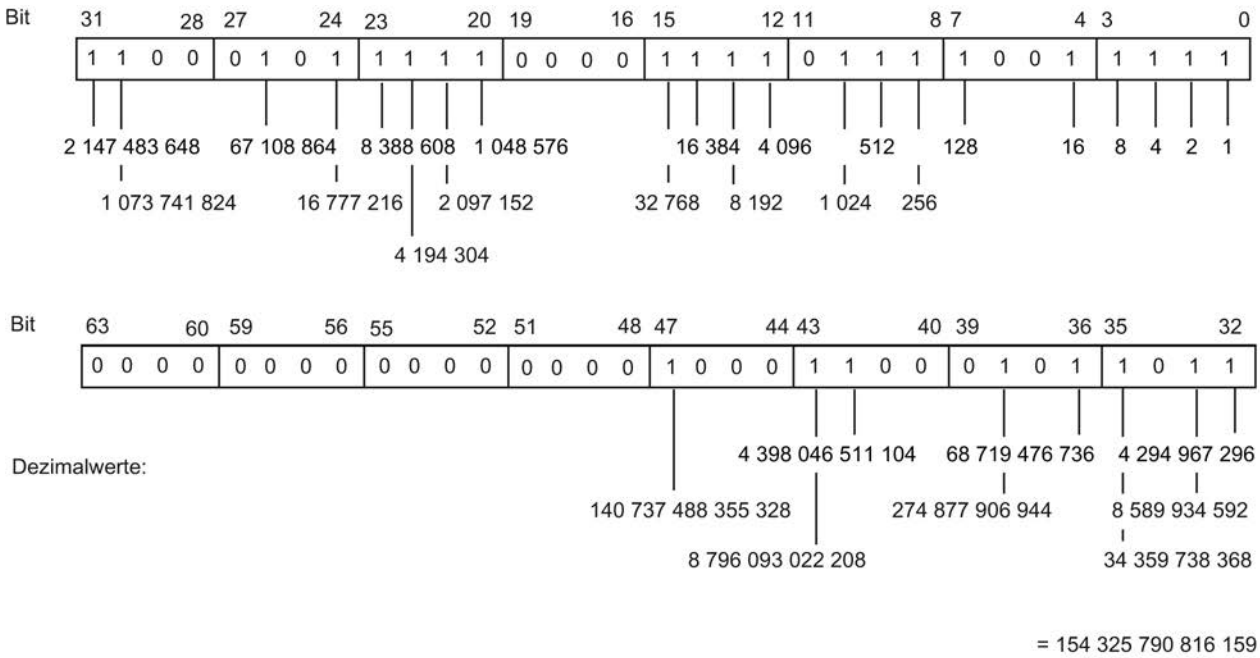
An operand of data type ULINT occupies eight BYTE in the memory.

The following table shows the properties of data type ULINT:

Length (bits)	Format	Value range	Examples of value input
64	Unsigned integers	0 to 18446744073709551615	154325790816159, ULINT#154325790816159
	Binary numbers	2#0 to 2#1111111111111111111111111111111111111111111111111111111111111111	2#0000000000000000100011000101101111000101111100001111011110011111, ULINT#2#0000000000000000100011000101101111000101111100001111011110111111
	Octal numbers	8#0 to 8#1777777777777777777777	8#4305570574173637, ULINT#8#4305570574173637
	Hexadecimal numbers	16#0 to 16#FFFFFFFFFFFFFFFF	16#00008C5BC5F0F79F, ULINT#16#00008C5BC5F0F79F

Example

The following figure shows the integer 154325790816159 as a binary number:



See also

Overview of the valid data types (Page 2236)

Overview of data type conversion (Page 2286)

Implicit conversion (Page 2288)

Explicit conversion (Page 2322)

Data type conversion (Page 2286)

Basics of constants (Page 2204)

Floating-point numbers

REAL

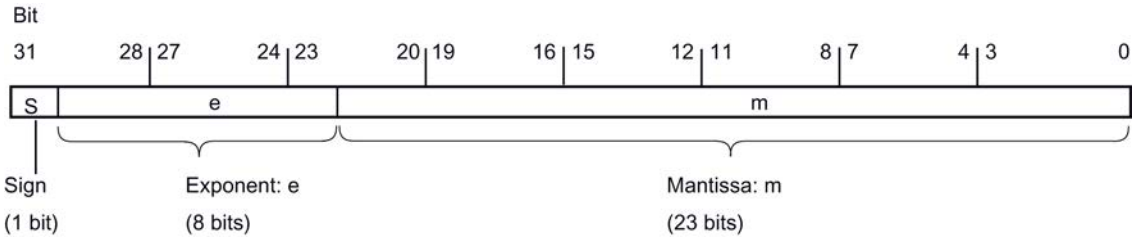
Description

Operands of the data type REAL have a length of 32 bits and are used to display floating-point numbers. An operand of the REAL data type consists of the following three components:

- Sign: The sign is determined by the signal state of bit 31. The bit 31 assume the value "0" (positive) or "1" (negative).
- 8-bit exponents to basis 2: The exponent is increased by a constant (base, +127), so that it has a value range of 0 to 255.
- 23-bit mantissa: Only the fraction part of the mantissa is shown. The integer part of the mantissa is always 1 with normalized floating-point numbers and is not stored.

The REAL data type is processed with a precision of 7 digits after the decimal point.

The following figure shows the structure of the REAL data type:



Note

With floating-point numbers, only the precision defined by the IEEE754 standard is stored. Additionally specified decimals are rounded off according to IEEE754.

The number of decimal places may decrease for frequently nested arithmetic calculations.

If more decimal places are specified than can be stored by the data type, the number is rounded to the corresponding value of the precision allowed by this value range .

The following table shows the properties of data type REAL:

Length (bits)	Format	Value range	Examples of value input
32	Floating-point numbers according to IEEE754	-3.402823e+38 to -1.175495e-38 ±0.0 +1.175495e-38 to +3.402823e+38	1.0e-5, REAL#1.0e-5
	Floating-point numbers		1.0, REAL#1.0

See also

Overview of the valid data types (Page 2236)

Data type conversion (Page 2286)

Basics of constants (Page 2204)

LREAL

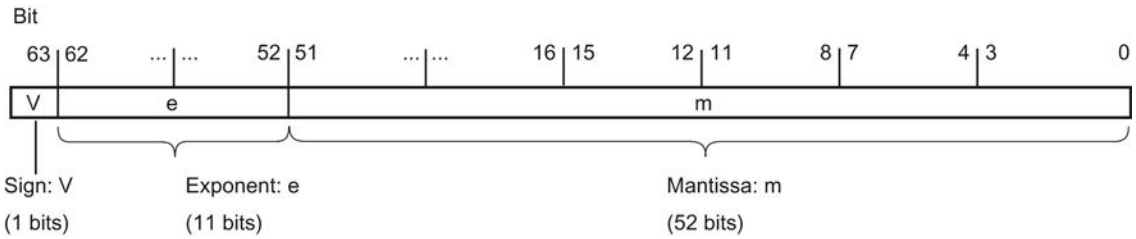
Description

Operands of the data type LREAL have a length of 64 bits and are used to represent floating-point numbers. An operand of the LREAL data type consists of the following three components:

- Sign: The sign is determined by the signal state of bit 63. The bit 63 assumes the value "0" (positive) or "1" (negative).
- 11-bit exponents to base 2: The exponent is increased by a constant (base, +1023), so that it has a value range of 0 to 2047.
- 52-bit mantissa: Only the fraction part of the mantissa is shown. The integer part of the mantissa is always 1 with normalized floating-point numbers and is not stored.

The LREAL data type is processed with a precision of 15 digits after the decimal point.

The following figure shows the structure of the LREAL data type:



The following table shows the properties of data type LREAL:

Length (bits)	Format	Value range	Examples of value input
64	Floating-point numbers according to IEEE754	-1.7976931348623158e+308 to -2.2250738585072014e-308 ±0.0	1.0e-5, LREAL#1.0e-5
	Floating-point numbers	+2.2250738585072014e-308 to +1.7976931348623158e+308	1.0, LREAL#1.0

Note

With floating-point numbers, only the precision defined by the IEEE754 standard is stored. Additionally specified decimals are rounded off according to IEEE754.

The number of decimal places may decrease for frequently nested arithmetic calculations.

If more decimal places are specified than can be stored by the data type, the number is rounded to the corresponding value of the precision allowed by this value range .

See also

Overview of the valid data types (Page 2236)

Data type conversion (Page 2286)

Basics of constants (Page 2204)

Invalid floating-point numbers

Description

We distinguish between four number ranges for data types REAL and LREAL:

- normalized numbers stored with full accuracy
- denormalized numbers not stored with full accuracy
- Infinite numbers: +Inf/-Inf (infinity)
- Invalid numbers: NaN (Not a Number)

Note

Floating-point numbers are stored as specified by the IEEE754 standard. Results of conversion or arithmetic functions with a denormalized, infinite or NaN (Not a Number) floating point depend on the CPU.

If you are not working with normalized floating-point numbers in mathematical functions, the result will show significant differences depending on the series of the CPU which you are using.

A CPU cannot calculate with denormalized floating-point numbers, with the exception of older CPU versions of the S7-300 and S7-400 series. The bit pattern of a denormalized number is interpreted as a zero. If the result of calculation falls into this range, it is continued with zero; the status bits OV and OS are set (number range undershoot).

Even though the values of invalid floating-point numbers can only be displayed with limited accuracy for mathematical functions, numbers with an exponent of -39 (e.g., 2.4408e-039) can

be monitored in the TIA Portal and therefore do not necessarily represent a faulty result. This means that floating-point values may be located outside the range of valid numerical values.

Note

The following applies to CPUs of the series S7-1200 V1, V2 and V3:

The comparison operation "Equal" uses the bit pattern of the invalid floating-point number. If two "NaN numbers" with the same bit pattern are compared, the output of the "Equal" comparison operation returns the result TRUE.

Note

The following applies to CPUs of the S7-1200 V4 and S7-1500 series:

If two invalid numbers (NaN) are compared with each other, the result is always FALSE, regardless of the bit pattern of the invalid number or the relation (>, >, ...).

Note

Comparison of denormalized floating-point numbers

For the comparison operation "Equal" with two denormalized floating-point numbers, the output for CPUs of the S7-300/400 series is set to the signal state "0" and for CPUs of the S7-1200/1500 series to the signal state "1".

If the input variables of a mathematical function represent an invalid floating-point number, an invalid floating-point number will also be output as result.

You have the following options to evaluate possible errors caused by invalid floating-point numbers:

- In LAD/FBD and SCL, you can query the enable output ENO for FALSE
- In STL, you can evaluate the status bit OV

See also

Overview of the valid data types (Page 2236)

Data type conversion (Page 2286)

Basics of constants (Page 2204)

Timers

S5TIME (duration)

Format

Data type S5TIME stores the duration in BCD format. The duration is the product from a time in the range 0 to 999 and a time basis. The time basis indicates the interval at which a timer

decrements the time value by one unit until it reaches "0". The resolution of the times can be controlled via the time basis.

The following table shows the range of values for data type S5TIME:

Length (bits)	Format	Range of values	Examples of value input
16	S7 time in increments of 10 ms (default value)	S5T#0MS to S5T#2H_46M_30S_0MS	S5T#10s, S5TIME#10s
	Hexadecimal numbers	16#0 to 16#3999	16#2

The following table shows the time base coding for S5TIME:

Time basis	Binary code for time basis
10 ms	00
100 ms	01
1 s	10
10 s	11

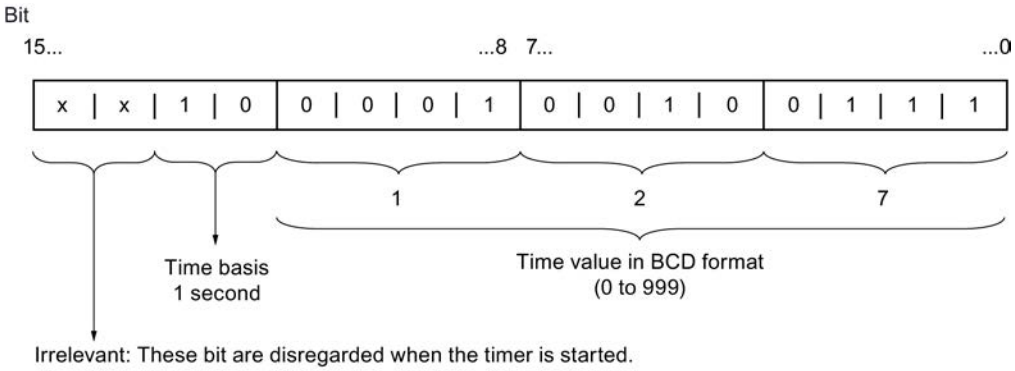
Always observe range limits and the resolution of time values when using data type S5TIME with timers. The table below specifies the range associated with each resolution:

Resolution	Range
0.01 s	10 ms to 9 s 990 ms
0.1 s	100 ms to 1 min 39 s 900 ms
1 s	1 s to 16 min 39 s
10 s	10 s to 2 h 46 min 30 s

Values that exceed 2h46m30s are not accepted.

Example

The following figure shows the content of the time operand for a time value of 127 and a time base of 1 s:



See also

Overview of the valid data types (Page 2236)

Data type conversion (Page 2286)

Basics of constants (Page 2204)

TIME (IEC time)

Description

The contents of an operand of the data type TIME is interpreted as milliseconds. The representation contains information for days (d), hours (h), minutes (m), seconds (s) and milliseconds (ms).

The following table shows the properties of data type TIME:

Length (bits)	Format	Value range	Examples of value input
32	Signed duration	T#-24d20h31m23s648ms to T#+24d20h31m23s647ms	T#10d20h30m20s630ms, TIME#10d20h30m20s630ms
	Hexadecimal numbers	16#00000000 to 16#7FFFFFFF	16#0001EB5E

It is not necessary to specify all time units. T#5h10s is a valid entry, for example. If only one unit is specified, the absolute value of days, hours, and minutes must not exceed the high or low limits. When more than one time unit is specified, the value must not exceed 24 days, 23 hours, 59 minutes, 59 seconds or 999 milliseconds.

See also

Overview of the valid data types (Page 2236)

Data type conversion (Page 2286)

Basics of constants (Page 2204)

LTIME (IEC time)

Description

The contents of an operand of data type LTIME is interpreted as nanoseconds. The representation contains information for days (d), hours (h), minutes (m), seconds (s) and milliseconds (ms), microseconds (us) and nanoseconds (ns).

The following table shows the properties of data type LTIME:

Length (bits)	Format	Range of values	Examples of value input
64	Signed duration	LT#-106751d23h47m16s854ms775us808ns to LT#+106751d23h47m16s854ms775us807ns	LT#11350d20h25m14s830ms652us315ns, LTIME#11350d20h25m14s830ms652us315ns
	Hexadecimal numbers	16#0 to 16#8000000000000000	16#2

It is not necessary to specify all time units. LT#5h10s is therefore a valid entry, for example. If only one unit is specified, the absolute value of days, hours, and minutes must not exceed the high or low limits. When more than one time unit is specified, the value must not exceed 106751 days, 23 hours, 59 minutes, 59 seconds, 999 milliseconds, 999 microseconds or 999 nanoseconds.

See also

Overview of the valid data types (Page 2236)

Overview of data type conversion (Page 2286)

Implicit conversion (Page 2288)

Explicit conversion (Page 2322)

Data type conversion (Page 2286)

Basics of constants (Page 2204)

Date and time

DATE

Format

The DATE data type saves the date as an unsigned integer. The representation contains the year, the month, and the day.

The contents of an operand of DATE data type correspond in hexadecimal format to the number of days since 01-01-1990 (16#0000).

The following table shows the properties of data type DATE:

Length (bytes)	Format	Range of values	Examples of value input
2	IEC date (Year-Month-Day)	D#1990-01-01 to D#2168-12-31	D#2009-12-31, DATE#2009-12-31
	Hexadecimal numbers	16#0000 to 16#FF62	16#00F2

See also

Overview of the valid data types (Page 2236)

Data type conversion (Page 2286)

Basics of constants (Page 2204)

TIME_OF_DAY (TOD)

Format

Data type TOD (TIME_OF_DAY) occupies a double word and stores the number of milliseconds since the beginning of the day (0:00 h) as unsigned integer.

The following table shows the properties of data type TOD:

Length (bytes)	Format	Range of values	Examples of value input
4	Time-of-day (hours:minutes:seconds)	TOD#00:00:00.000 to TOD#23:59:59.999	TOD#10:20:30.400, TIME_OF_DAY#10:20:30.400

You always need to specify the hours, minutes and seconds. The specification of milliseconds is optional.

See also

Overview of the valid data types (Page 2236)

Data type conversion (Page 2286)

Basics of constants (Page 2204)

LTOD (LTIME_OF_DAY)

Format

Data type LTOD (LTIME_OF_DAY) occupies two double words and stores the number of nanoseconds since the beginning of the day (0:00 h) as unsigned integer.

The following table shows the properties of data type LTOD:

Length (bytes)	Format	Range of values	Examples of value input
8	Time-of-day (hours:minutes:seconds.milliseconds.microseconds.nanoseconds)	LTOD#00:00:00.000000000 to LTOD#23:59:59.999999999	LTOD#10:20:30.400_365_215, LTIME_OF_DAY#10:20:30.400_365_215

You always need to specify the hours, minutes and seconds. The specification of milliseconds, microseconds and nanoseconds is optional.

See also

Overview of the valid data types (Page 2236)
Overview of data type conversion (Page 2286)
Implicit conversion (Page 2288)
Explicit conversion (Page 2322)
Data type conversion (Page 2286)
Basics of constants (Page 2204)

DATE_AND_TIME (date and time of day)

Format

The DT (DATE_AND_TIME) data type saves the information on date and time of day in BCD format.

The following table shows the properties of data type DT:

Length (bytes)	Format	Range of values	Example of value input
8	Date and time (year-month-day-hour:minute:second:millisecond [3])	Min.: DT#1990-01-01-00:00:00.000 Max.: DT#2089-12-31-23:59:59.999	DT#2008-10-25-8:12:34.567, DATE_AND_TIME#2008-10-25-08:12:34.567

The following table shows the structure of the DT data type:

Byte	Contents	Range of values
0	Year	0 to 99 (Years 1990 to 2089) BCD#90 = 1990 ... BCD#0 = 2000 ... BCD#89 = 2089
1	Month	BCD#0 to BCD#12
2	Day	BCD#1 to BCD# 31
3	Hour	BCD#0 to BCD#23
4	Minute	BCD#0 to BCD#59
5	Second	BCD#0 to BCD#59
6	The two most significant digits of MSEC	BCD#0 to BCD#999
7 (4MSB) [1]	The least significant digit of MSEC	BCD#0 to BCD#9

Byte	Contents	Range of values
7 (4LSB) [2]	Weekday	BCD#1 to BCD#7 BCD#1 = Sunday ... BCD#7 = Saturday
[1] MSB: Most significant bit [2] LSB: Least significant bit [3] Fixed point number		

See also

Overview of the valid data types (Page 2236)

Data type conversion (Page 2286)

Basics of constants (Page 2204)

LDT (DATE_AND_LTIME)

Format

Data type LDT (DATE_AND_LTIME) stores the date and time-of-day information in nanoseconds since 01/01/1970 0:0.

The following table shows the properties of data type LDT:

Length (bytes)	Format	Range of values	Example of value input
8	Date and time (Year-Month-Day-Hour:Minute:Second)	Min.: LDT#1970-01-01-0:0:0.0000 00000 Max.: LDT#2200-12-31-23:59:59.9 99999999	LDT#2008-10-25-8:12:34.567
	Hexadecimal numbers	16#0 to 16#7FFF_FFFF_FFFF_FFF F	16#7FFF

See also

Overview of the valid data types (Page 2236)

Overview of data type conversion (Page 2286)

Implicit conversion (Page 2288)

Explicit conversion (Page 2322)

Data type conversion (Page 2286)

Basics of constants (Page 2204)

DTL

Description

An operand of data type DTL has a length of 12 bytes and stores date and time information in a predefined structure.

The following table shows the properties of data type DTL:

Length (bytes)	Format	Value range	Example of value input
12	Date and time (Year-Month-Day-Hour:Minute:Second.Nanoseconds)	Min.: DTL#1970-01-01-00:00:00.0 Max.: DTL#2200-12-31-23:59:59.999999999	DTL#2008-12-16-20:30:20.250

The structure of data type DTL consists of several components each of which can contain a different data type and range of values. The data type of a specified value must match the data type of the corresponding components.

The following table shows the structure components of data type DTL and their properties:

Byte	Component	Data type	Value range
0	Year	UINT	1970 to 2200
1			
2	Month	USINT	1 to 12
3	Day	USINT	1 to 31
4	Weekday	USINT	1(Sunday) to 7(Saturday) The weekday is not considered in the value entry.
5	Hour	USINT	0 to 23
6	Minute	USINT	0 to 59
7	Second	USINT	0 to 59
8	Nanosecond	UDINT	0 to 999999999
9			
10			
11			

See also

Overview of the valid data types (Page 2236)

Data type conversion (Page 2286)

Basics of constants (Page 2204)

Character strings

CHAR (character)

Description

An operand of data type CHAR has a length of 8 bits and occupies one BYTE in the memory.

The CHAR data type stores a single character in ASCII format. You can find information on the encoding of special characters under "See also > STRING".

The following table shows the value range of the CHAR data type:

Length (bits)	Format	Value range	Example of value inputs
8	ASCII characters	ASCII character set	'A', CHAR#'A'

See also

Overview of the valid data types (Page 2236)

Data type conversion (Page 2286)

Basics of constants (Page 2204)

STRING (Page 2264)

WCHAR (character)

Description

An operand of data type WCHAR (Wide Characters) has a length of 16 bits and occupies two BYTE in the memory.

The WCHAR data type saves a single character of an expanded character set which is stored in Unicode format. However, only a subset of the entire Unicode range is covered. When a control character is entered, it is represented with a dollar sign.

The following table shows the value range of the WCHAR data type:

Length (bits)	Format	Range of values	Example of value input
16	Unicode	$0000 - $D7FF	WCHAR#'a'

See also

Overview of the valid data types (Page 2236)

Constants (Page 2204)

STRING

Description

An operand of the STRING data type saves several characters in a character string that can consist of up to 254 characters. In a character string, all characters of the ASCII code are permitted. The characters are specified in single quotation marks.

The following table shows the properties of a STRING tag:

Length (bytes)	Format	Value range	Example of value input
n + 2 *	ASCII character string incl. special characters	0 to 254 characters	'Name', STRING#'NAME'
* An operand of the STRING data type occupies two bytes more than the specified maximum length in the memory.			

A character string can also contain special characters. The escape character $ is used to identify control characters, dollar signs and single quotation marks.

The table below shows examples for the notation of special characters:

Character	Hex	Meaning	Example
$L or $l	0A	Line feed	'$LText', '$0AText'
$N	0A and 0D	Line break The line break occupies 2 characters in the character string.	'$NText', '$0A$0DText'
$P or $p	0C	Page feed	'$PText', '$0CText'
$R or $r	0D	Carriage return (CR)	'$RText','$0DText'
$T or $t	09	Tab	'$TText', '$09Text'
$$	24	Dollar sign	'100$$t', '100$26'
$'	27	Single quotation mark	'$'Text$','$27Text$27'

The maximum length of the character string can be specified during the declaration of an operand using square brackets after the keyword STRING (for example, STRING[4]). If the specification of the maximum length is omitted, the standard length of 254 characters is set for the respective operand.

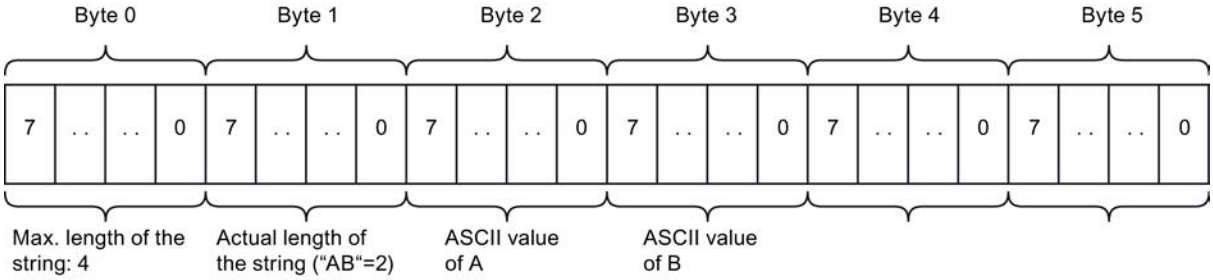
If the actual length of a specified character string is shorter than the declared maximum length, the characters are written to the character string left-justified and the remaining character spaces remain undefined. Only occupied character spaces are considered in the value processing.

Note

For S7-300/400 CPUs, please note: If a temporary tag of the STRING data type was defined, you must describe the BYTE "Max. length of string" with the defined length before you use the tags in the user program.

Example

The example below shows the byte sequence if the STRING[4] data type is specified with output value 'AB':



See also

Overview of the valid data types (Page 2236)

Data type conversion (Page 2286)

Basics of constants (Page 2204)

WSTRING

Description

An operand of data type WSTRING (Wide String) stores several Unicode characters of data type WCHAR in one character string. If you do not specify a length, the character string has a preset length of 254 characters. In a character string, all characters of the Unicode format are permitted. This means you can also use Chinese characters in a character string.

When declaring an operand of data type WSTRING you can define its length using square brackets (for example WSTRING[10]). If you do not specify a length, the length of the WSTRING is set to 254 characters by default. You can declare a length of up to 16382 characters (WSTRING[16382]).

Note

Use in the watch table

If you wish to monitor or control a tag with data type WSTRING in the watch table, it cannot contain more than 254 characters.

The specification of the characters occurs in single quotes and always with the qualifier WSTRING#.

The table below shows the properties of a WSTRING tag:

Length (WORD)	Format	Range of values	Example of value input
n + 2 *	Unicode character string; n specifies the length of the character string.	Preset value: 0 to 254 characters Max. possible value: 0 to 16382	WSTRING#'Hello World'
* An operand of the WSTRING data type occupies two WORDs more in the memory than the specified maximum length.			

A character string can also contain special characters. The escape character $ is used to identify control characters, dollar signs and single quotation marks.

The table below shows examples for the notation of special characters:

Character	Hex	Meaning	Example
$L or $l	000A	Line feed	'$LText', '$000AText'
$N	000A and 000D	Line break The line break occupies 2 characters in the character string.	'$NText', '$000A $000DText'
$P or $p	000C	Page feed	'$PText', '$000CText'
$R or $r	000D	Carriage return (CR)	'$RText','$000DText'
$T or $t	0009	Tab	'$TText', '$0009Text'
$$	0024	Dollar sign	'100$$t', '100$0024t'
$'	0027	Single quotation mark	'$'Text$','$0027Text $0027'

The maximum length of the character string can be specified during the declaration of an operand using square brackets after the keyword WSTRING (for example, WSTRING[4]). If the specification of the maximum length is omitted, the standard length of 254 characters is set for the respective operand.

If the actual length of a specified character string is shorter than the declared maximum length, the characters are written to the character string left-justified and the remaining character spaces remain undefined. Only occupied character spaces are considered in the value processing.

Note

Conversion of WSTRING tags

Implicit conversion of the WSTRING data type is not possible. Explicit conversion from the WSTRING data type to STRING is generally possible. However, normally only the conversion of characters in the code range from 0 - 127 will work in all Windows code pages. For all characters outside this range, the code page character and the Unicode character must be in exactly the same position for the conversion to work without errors.

Access to block parameters of data type WSTRING

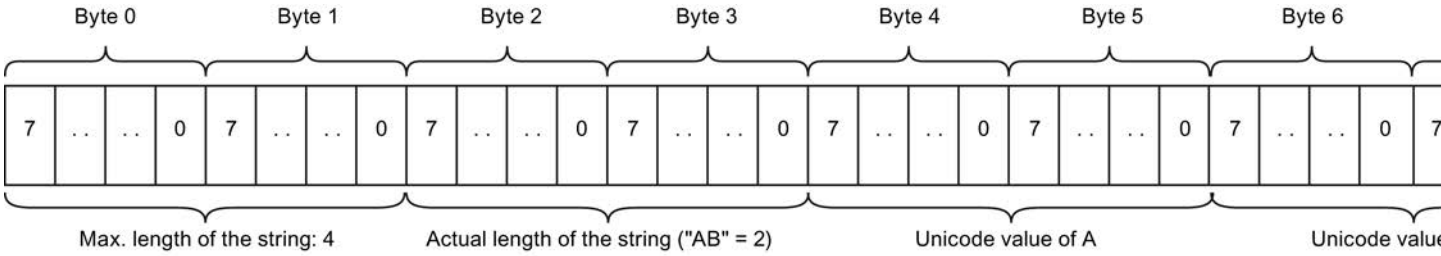
Operands of the data type WSTRING can be transferred as parameters up to the maximum length for blocks with "optimized" access.

For function blocks (FB) with "standard" access, operands of the data type WSTRING can be declared as parameters in all sections of the block interface except in the section "InOut". For a function (FC) with "standard" access, only operands of the data type STRING can be transferred as parameters.

The function value of an FC in the "Return" section and expressions in the SCL programming language are another exception to this rule. In these cases, the WSTRING tag must not be longer than 1022 characters. If you need a WSTRING tag with more than 1022 characters, declare a tag of the data type "WSTRING" with the required character length in the section "Temp" and assign the function value to the tag.

Example

The example below shows the byte sequence if the WSTRING[4] data type is specified with output value 'AB':



See also

Overview of the valid data types (Page 2236)

Constants (Page 2204)

Array

Format of array (16-bit limits)

Description

The Array data type represents a data structure that consists of a fixed number of components of the same data type. All data types except Array are permitted.

A tag with the Array data type always starts at a WORD limit.

The array components are addressed by means of an index. In the array declaration, the index limits are defined in square brackets after the keyword Array. The low limit must be smaller than or equal to the high limit. An array may contain up to six dimensions, the limits of which can be specified separated by a comma.

The following table shows the property of the Array data type:

Length	Format	Index limits	Data type
Number of components * length of the data type	Array [low limit...high limit] of <data type>	[-32768..32767] of <data type>	Bit strings, integers, floating-point numbers, timers, character strings, structures

Example

The following example shows how operands of data type Array can be declared:

Name	Declaration	Comment
Measured value	Array[1..20] of REAL	One-dimensional array with 20 components
Time-of-day	Array[-5..5] of INT	One-dimensional array with 11 components
Character	Array[1..2, 3..4] of CHAR	Two-dimensional array with 4 components

Maximum array limits

The maximum Array limits depend on the following factors:

- Data type of the Array elements
- Maximum storage capacity of the CPU (you can find more information in the relevant device manual)

See also

Overview of the valid data types (Page 2236)

Indirect indexing of ARRAY components (Page 2222)

Format of array (32-bit limits)

Description

The Array data type represents a data structure that consists of a fixed number of components of the same data type. All data types except Array are permitted.

The array components are addressed by means of an index. In the array declaration, the index limits are defined in square brackets after the keyword Array. The low limit must be smaller than or equal to the high limit. An array may contain up to six dimensions, the limits of which can be specified separated by a comma.

Note

Depending on the CPU, the storage capacity of a data block is limited and the number of components of the Array is therefore also limited. However, you may initialize the addressing of the array components at any position within index limits.

The following table shows the property of the Array data type:

Length	Format	Index limits	Data type
Number of components * length of the data type	Array [low limit...high limit] of <data type>	[-2147483648..2147483647] of <data type>	Bit strings, integers, floating-point numbers, timers, character strings, structures

Note

The length of the array depends on whether the block was created with the "Standard" or "with optimized access" block property.

Note

Applies to CPUs of the S7-1500 series

For a block with the block property "Optimized block access", a bit requires 1 byte of memory. This is also true when you use an ARRAY of <data type>. An ARRAY [0..1] of BOOL, for example, thus requires 2 bytes in an optimized block.

Example

The following example shows how operands of data type Array can be declared:

Name	Declaration	Comment
Measured value	Array[1..20] of REAL	One-dimensional array with 20 components
Time-of-day	Array[-5..5] of INT	One-dimensional array with 11 components
Character	Array[1..2, 3..4] of CHAR	Two-dimensional array with 4 components

Maximum array limits

The maximum array limits depend on the following factors:

- Data type of the array elements
- Memory reserve (only in blocks with optimized access)
  You can find additional information on this topic in the section "Loading block changes without reinitialization".
- Maximum size of a data block for a CPU (you can find more information in the respective device manual)
- The entire length of the array is available within a data block. Within a program block (OB, FB or FC), the possible length is reduced by the memory capacity required by the program code.

Example based on a CPU of the S7-1200 series

The following table shows the maximum number of elements within a block with the "with optimized access" block property:

Data type width (bits)	Maximum number of elements	Note
1	524272	= 65534*8
8	65534	Refer to the respective device manual of the CPU for the value.
16	32767	= 65534/2 (integer division, remainder 0)
32	16383	= 65534/4 (integer division, remainder 2)
64	8191	= 65534/8 (integer division, remainder 6)

Due to various technical/internal constraints, the actual usable memory area may be approximately 70 - 100 bytes less. The memory area may be further restricted due to a default setting, for example, by the "Load without reinitialization" block property.

See also

Overview of the valid data types (Page 2236)

Indirect indexing of ARRAY components (Page 2222)

Basic information on loading block extensions without reinitialization (Page 2914)

Activating memory reserve (Page 2917)

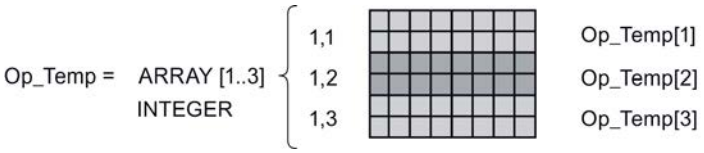
Example of a one-dimensional array

Declaration

The following table shows the declaration of a one-dimensional Array tag:

Name	Data type	Comment
Op_Temp	Array[1..3] of INT	One-dimensional array tag with 3 components.

The following figure shows the structure of the declared array tag:



Access to ARRAY components

The individual array components are accessed via an index.. The index of the first ARRAY component is [1], of the second [2], and of the third [3]. To access the value of the second ARRAY component, you need to declare "Op_Temp[2]" in the program.

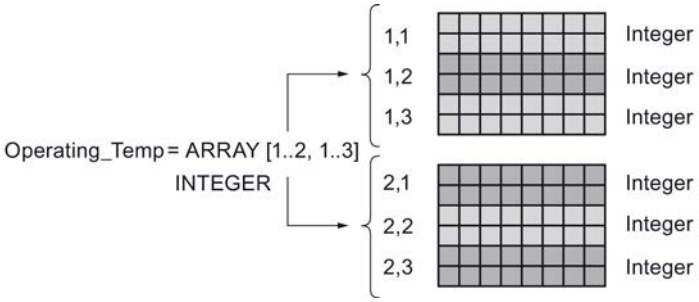
Example of a multi-dimensional array

Declaration

The following table shows the declaration of a two-dimensional Array tag:

Name	Data type	Value	Comment
Betr_Temp	Array[1..2, 1..3] of INT	1,1,4(0)	Two-dimensional array tag with 6 components. The first two components are assigned the value "1". The remaining four components are assigned the value "0".

The following figure shows the structure of the declared array tag:



Access to the array components

The values of the individual array components are accessed via an index. The index of the first array component is, for example, [1,1] and the index of the fourth array component is [2,1]. For example, you need to declare "Betr_Temp[2,1]" in the program to enable access to the value of the fourth array component.

Additional access option

You can also declare the "Betr_Temp" TAG as a six-dimensional array. The following table shows an example of the declaration of a six-dimensional Array tag:

Name	Data type	Value	Comment
Betr_Temp	Array[1..3, 1..2, 1..3, 1..4, 1..3, 1..4] of INT	-	Six-dimensional array tag

The index of the first array component is in this case [1,1,1,1,1,1] and the index of the last component is [3,2,3,4,3,4]. Intermediate values are accessed by entering the corresponding value for each dimension.

Structures

STRUCT

Description

Data type STRUCT represents a data structure that consists of a fixed number of components of various data types. Components of STRUCT or ARRAY data type can also be nested in a structure. The nesting depth is hereby limited to eight levels. Structures can be used to group data according to the process control system and to transfer parameters as one data unit.

You can create up to 65534 structures for a CPU of the S7-1200 or S7-1500 series. Each of these structures can include up to 252 components. In addition, you can create up to 65534 function blocks, 65535 functions and 65535 organization blocks with up to 252 components each.

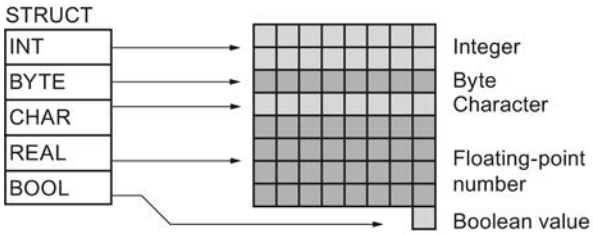
A component of the ARRAY data type always starts at a WORD limit.

The following table shows the properties of data type STRUCT:

Length	Format	Range of values	Example of value input
A STRUCT variable starts with one byte with even address and occupies the memory up to the next word limit.	STRUCT	The value ranges of the used data types apply.	The value input rules of the used data types apply.

Example

The following figure shows an example of the structure of a STRUCT variable:



See also

Overview of the valid data types (Page 2236)

Pointer

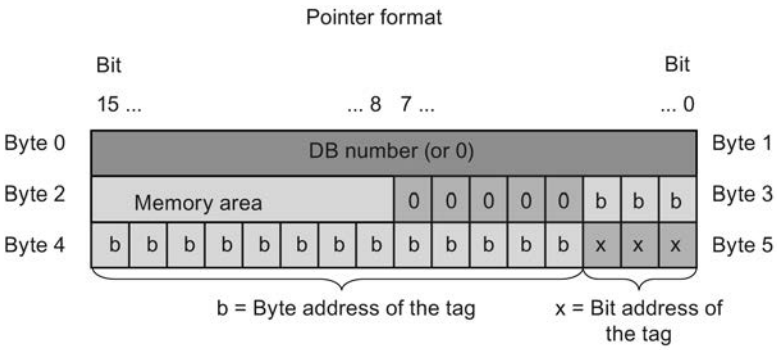
POINTER

Description

A parameter of the type POINTER is a pointer that can point to a specific tag. It occupies 6 bytes (48 bits) in memory and may contain the following tag information:

- DB number, or 0 if the data is not stored in a DB
- Memory area in the CPU
- Tag address

The following figure shows the structure of parameter type POINTER:



Types of pointer

Depending on the information, you can use parameter type POINTER to declare the following four types of pointer:

- Area-internal pointer:
  An area-internal pointer contains information on the address of a tag.
- Cross-area pointer:
  A cross-area pointer contains information on the memory area and the address of a tag.
- DB pointer:
  You can use a DB pointer to point to a data block tag. A DB pointer also contains a data block number in addition to the memory area and the address of a tag.
- Zero pointer:
  Use the zero pointer to indicate a missing value. A missing value may indicate that no value exists, or that the value is not yet known. A zero value represents the absence of a value, but is also a value.

The following table shows the formats for the declaration of various pointer types:

P#ByteRepresentation	Format	Example of value input	Description
Symbolic	P#Byte.Bit	"MyTag"	Area-internal pointer
	P#OperandAreaByte.Bit	"MyTag"	Cross-area pointer
	P#Data_block.Data_operand	"MyDB"."MyTag"	DB pointer
	P#Zero value	-	Zero pointer
Absolute	P#Byte.Bit	P#20.0	Area-internal pointer
	P#OperandAreaByte.Bit	P#M20.0	Cross-area pointer
	P#Data_block.Data_operand	P#DB10.DBX20.0	DB pointer
	P#Zero value	P#0.0, ZERO	Zero pointer

You can enter a parameter of the type POINTER without prefix (P#). The entry is then automatically converted to the POINTER format.

Note

If you use the prefix P#, you can only point to memory areas with "standard" access mode.

Memory areas

The following table shows the hexadecimal codes of the memory areas for parameter type POINTER:

Hexadecimal code	Memory area	Description
B#16#80[1)]	P	Peripherals on a CPU S7-300/400
16#1	P	Peripheral inputs on a CPU S7-1500
16#2	P	Peripheral outputs on a CPU S7-1500
B#16#81	I	Memory area of inputs
B#16#82	Q	Memory area of outputs
B#16#83	M	Memory area of bit memory
B#16#84	DBX	Data block
B#16#85	DIX	Instance data block
B#16#86	L	Local data
B#16#87	V	Previous local data
[1)] These data types can only be used for the POINTER pointer on a CPU S7-300/400.		

See also

Basics of indirect addressing (Page 2220)

Overview of the valid data types (Page 2236)

Data type conversion (Page 2286)

Basics of constants (Page 2204)

ANY

Description

An ANY type parameter points to the start of a data area and specifies its length. An ANY pointer occupies 10 bytes of memory and may contain the following information:

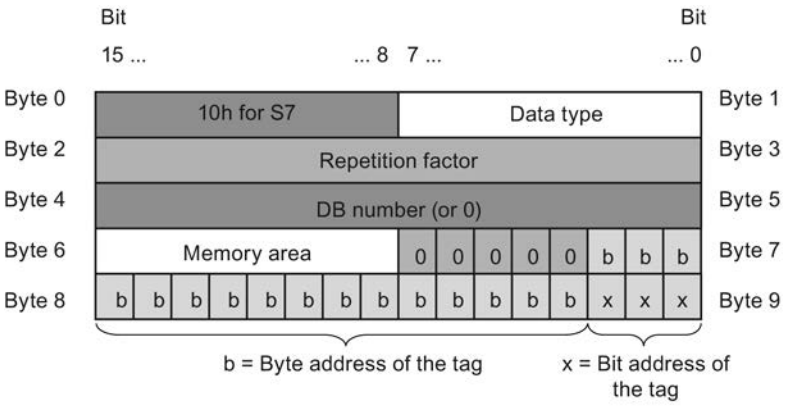
- Data type:
  Data type of the elements of the data area
- Repetition factor:
  Number of elements of the data area
- DB number:
  Data block that contains the declaration of data area elements.
- Memory area:
  Memory area of the CPU that stores the data area elements.
- Start address of the data in the format "byte.bit":
  Data area start identified by the ANY pointer.
- Zero pointer:
  Use the zero pointer to indicate a missing value. A missing value may indicate that no value exists, or that the value is not yet known. A zero value represents the absence of a value, but is also a value.

Note

Memory area

For an S7-1500 CPU, the ANY pointer can also only point to memory areas with "Standard" access mode.

The following figure shows the structure of the ANY pointer:



An ANY pointer cannot identify structures. It can only be assigned to local tags.

The following table shows the formats for the declaration of an ANY pointer:

Representation	Format	Example of value input	Description
Symbolic	P#DataBlock.MemoryArea DataAddress Type Number	"MyDB".StructTag.InitialComponents	Area with 10 words in global DB11 starting with DBB20.0
	P#MemoryArea DataAddress Type Number	"MyMarkerTag"	Area with 4 bytes starting with MB 20.0
		"MyTag"	Input I1.0
	P#Zero value	-	Zero value
Absolute	P#DataBlock.MemoryArea DataAddress Type Number	P#DB11.DBX20.0 INT 10	Area with 10 words in global DB11 starting with DBB20.0
	P#MemoryArea DataAddress Type Number	P#M20.0 BYTE 10	Area with 10 bytes starting with MB 20.0
		P#E1.0 BOOL 1	Input I1.0
	P#Zero value	P#0.0 VOID 0, ZERO	Zero value

Coding of data types

The following table lists the coding of data types for the ANY pointer:

Hexadecimal code	Data type	Description
B#16#00	NIL	Null pointer
B#16#01[1)]	BOOL	Bits
B#16#02	BYTE	bytes, 8 bits
B#16#03	CHAR	8-bit characters
B#16#04	WORD	16-bit words
B#16#05	INT	16-bit integers
B#16#06	DWORD	32-bit words
B#16#07	DINT	32-bit integers
B#16#08	REAL	32-bit floating-point numbers
B#16#0B	TIME	Time duration
B#16#0C	S5TIME	Time duration
B#16#09	DATE	Date
B#16#0A	TOD	Date and time
B#16#0E	DT	Date and time
B#16#13	STRING	Character string
B#16#17[1)]	BLOCK_FB	Function block
B#16#18[1)]	BLOCK_FC	Function
B#16#19[1)]	BLOCK_DB	Data block
B#16#1A[1)]	BLOCK_SDB	System data block
B#16#1C[1)]	COUNTER	Counter

Hexadecimal code	Data type	Description
B#16#1D[1)]	TIMER	Timer
[1)] These data types can only be used for the ANY pointer on a CPU S7-300/400.		

Coding of the memory area

The following table lists the coding of the memory areas for the ANY pointer:

Hexadecimal code	Area	Description
B#16#80[1)]	P	I/O
B#16#81	I	Memory area of inputs
B#16#82	Q	Memory area of outputs
B#16#83	M	Memory area of bit memory
B#16#84	DBX	Data block
B#16#85	DIX	Instance data block
B#16#86	L	Local data
B#16#87	V	Previous local data
[1)] These memory areas can only be used for the ANY pointer on an S7-300/400 CPU.		

See also

Basics of indirect addressing (Page 2220)

Overview of the valid data types (Page 2236)

Data type conversion (Page 2286)

Basics of constants (Page 2204)

VARIANT

Description

A parameter of the VARIANT type is a pointer that can point to tags of different data types other than an instance. The VARIANT pointer can be an object of an elementary data type, such as INT or REAL. It can also be a STRING, DTL, array of STRUCT, UDT, or array of UDT. The VARIANT pointer can recognize structures and point to individual structure components. An operand of data type VARIANT occupies no space in the instance DB or L stack. However, it will occupy memory space on the CPU.

A tag of the VARIANT type is not an object but rather a reference to another object. Individual elements of the VARIANT type can only be declared within the block interface of a function. For this reason, it cannot be declared in a data block or in the static section of the block interface of a function block, for example, because its size is unknown. The size of the referenced objects can change.

The VARIANT data type can only be defined on formal parameters in the block interface of a block.

Note

You can only point to a complete data block if it was originally derived from a user-defined data type (UDT).

The following table shows the properties of the VARIANT pointer:

Length (bytes)	Representation	Format	Example of value input
0	Symbolic	Operand	"TagResult"
		NameDataBlock.NameOperand.Component	"Data_TIA_Portal".StructVariable.FirstComponent
	Absolute	Operand	%MW10
		DataBlockNumber.Operand Type Length (valid only for blocks with standard access)	P#DB10.DBX10.0 INT 12
		P#Zero value	P#0.0 VOID 0, ZERO

Note

If you use the prefix P#, you can only point to memory areas with "standard" access mode.

Coding of data types

If you use absolute addressing with P#, the following data types are permitted:

Hexadecimal code	Data type	Description
B#16#00	NIL	Null pointer
B#16#01	BOOL	Bits
B#16#02	BYTE	bytes, 8 bits
B#16#03	CHAR	8-bit characters
B#16#04	WORD	16-bit words
B#16#05	INT	16-bit integers
B#16#06	DWORD	32-bit words
B#16#07	DINT	32-bit integers
B#16#08	REAL	32-bit floating-point numbers
B#16#0B	TIME	Time duration
B#16#0C	S5TIME	Time duration
B#16#09	DATE	Date
B#16#0A	TOD	Date and time
B#16#0E	DT	Date and time
B#16#13	STRING	Character string
B#16#17	BLOCK_FB	Function block

Hexadecimal code	Data type	Description
B#16#18	BLOCK_FC	Function
B#16#19	BLOCK_DB	Data block
B#16#1A	BLOCK_SDB	System data block
B#16#1C	COUNTER	Counter
B#16#1D	TIMER	Timer

Example

The following example shows how VARIANT works using the "MOVE: Move value" instruction of STL:

STL	Description
CALL MOVE	// The "Move value" instruction is called.
VARIANT	// Data type of the instruction
IN := "Data_TIA_Portal".StructVariable.FirstComponent	// The contents of the "FirstComponent" operand are moved from the "Data_TIA_Portal" DB.
OUT := "MotorDB".StructResult.TagResult	// And transferred to the "TagResult" operand from the "MotorDB" DB.

See also

Basics of indirect addressing (Page 2220)

Overview of the valid data types (Page 2236)

Data type conversion (Page 2286)

Basics of constants (Page 2204)

Basics of block access (Page 2175)

Principles of block calls (Page 2180)

Using VARIANT instructions (Page 263)

Parameter types

Parameter types

Description

The parameter types are data types for formal parameters that are transferred to called blocks. A parameter type can also be a PLC data type.

The following table shows the available parameter data types and their purpose:

Parameter type	Length (bits)	Description
TIMER	16	Is used to specify a timer that is used in the called code block. If you supply a formal parameter of the TIMER parameter type, the associated actual parameter must be a timer. Example: T1
COUNTER	16	Is used to specify a counter that is used in the called code block. If you supply a formal parameter of the COUNTER parameter type, the associated actual parameter must be a counter. Example: C10
BLOCK_FC	16	Is used to specify a block that is used as input in the called code block. The declaration of the parameter determines the block type (for example FB, FC, DB) that is to be used. If you supply a formal parameter of the BLOCK parameter type, specify a block address as the actual parameter. Example: DB3
BLOCK_FB	16	
BLOCK_DB	16	
BLOCK_SDB	16	
BLOCK_SFB	16	
BLOCK_SFC	16	
BLOCK_OB	16	
VOID	-	The VOID parameter type does not save any values. This parameter type is used if the return values of an output are not required. The VOID parameter type can be specified at the STATUS output, for example, if no error information is required.

See also

Overview of the valid data types (Page 2236)

Basics of PLC data types (Page 2854)

PLC data types

PLC data types

Description

PLC data types are data structures that you define and that can be used multiple times within the program. The structure of a PLC is made up of several components, each of which can contain different data types. You define the type of components during the declaration of the PLC data type.

You can create up to 65534 PLC data types for a CPU of the S7-1200 or S7-1500 series. Each of these PLC data types can include up to 252 components.

PLC data types can be used for the following applications:

- PLC data types can be used as data types for variables in the variable declaration of logic blocks or in data blocks.
- PLC data types can be used as templates for the creation of global data blocks with identical data structures.
- PLC data types can be used in S7-1200 and S7-1500 as a template for the creation of structured PLC tags.

See also

Programming PLC data types (Page 2854)

Addressing structured variables (Page 2214)

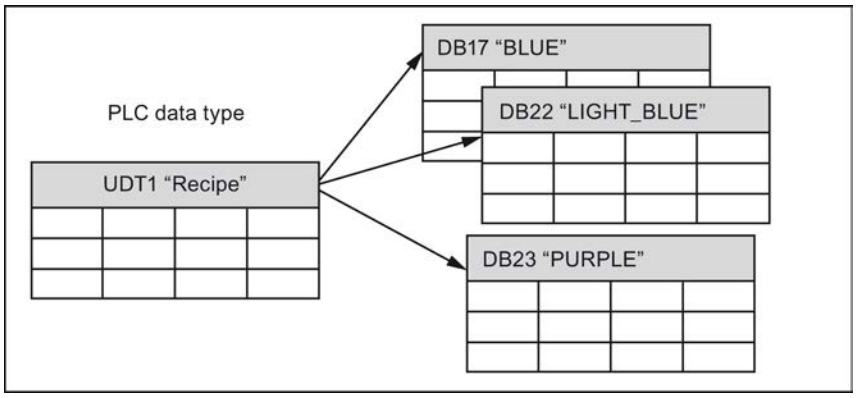
Example of a PLC data type

Example

You can declare PLC data types as the type when creating data blocks. Based on this type you can create a number of data blocks, all of which have the same data structure. These data blocks can be adapted by entering different actual values for the corresponding task.

For instance, create a PLC data type for a recipe for blending paints. You can then assign this data type to several data blocks, each of which contains other quantity information.

The following figure shows this application:



System data types

System data types

Description

The system data types (SDT) are made available by the system and have a predefined structure. The structure of a system data type consists of a fixed number of components that can have various data types. It is not possible to change the structure of a system data type.

The system data types can only be used for specific instructions. The following table shows the available system data types and their purpose:

System data type	Length (bytes)	Description
IEC_TIMER	16	Structure of a timer whose timer values are of TIME data type. This data type is used for the "TP", "TOF", "TON", "TONR", "RT" and "PT" instructions, for example.
IEC_LTIMER	32	Structure of a timer whose timer values are of LTIME data type. This data type is used for the "TP", "TOF", "TON", "TONR", "RT" and "PT" instructions, for example.
IEC_SCOUNTER	3	Structure of a counter whose count values are of SINT data type. This data type is used for the "CTU", "CTD" and "CTUD" instructions, for example.
IEC_USCOUNTER	3	Structure of a counter whose count values are of USINT data type. This data type is used for the "CTU", "CTD" and "CTUD" instructions, for example.
IEC_COUNTER	6	Structure of a counter whose count values are of INT data type. This data type is used for the "CTU", "CTD" and "CTUD" instructions, for example.
IEC_UCOUNTER	6	Structure of a counter whose count values are of UINT data type. This data type is used for the "CTU", "CTD" and "CTUD" instructions, for example.
IEC_DCOUNTER	12	Structure of a counter whose count values are of DINT data type. This data type is used for the "CTU", "CTD" and "CTUD" instructions, for example.
IEC_UDCOUNTER	12	Structure of a counter whose count values are of UDINT data type. This data type is used for the "CTU", "CTD" and "CTUD" instructions, for example.
IEC_LCOUNTER	24	Structure of a counter with count values of data type UDINT. This data type is used for the "CTU", "CTD" and "CTUD" instructions, for example.
IEC_ULCOUNTER	24	Structure of a counter with count values of data type UINT. This data type is used for the "CTU", "CTD" and "CTUD" instructions, for example.
ERROR_STRUCT	28	Structure of an error information to a programming or I/O access error. This data type is used, for example, for the "GET_ERROR" instruction.
CREF	8	Components of the ERROR_STRUCT data type, in which information about the address of a block is saved.
NREF	8	Components of the ERROR_STRUCT data type, in which information about the address of an operand is saved.

System data type	Length (bytes)	Description
VREF	12	Is used for storage of a VARIANT pointer. This data type is, for example, used for instructions from S7-1200 Motion Control.
STARTINFO	12	Specifies the data structure in which the start information is saved. This data type is used, for example, for the "RD_SINFO" instruction.
SSL_HEADER	4	Specifies the data structure in which information about the data records are saved during the reading of the system status lists. This data type is used, for example, for the "RDSYSST" instruction.
CONDITIONS	52	User-defined data structure defining the conditions for start and end of a data reception. This data type is used, for example, for the "RCV_CFG" instruction.
TADDR_Param	8	Specifies the structure of a data block which stores descriptions of connections for Open User Communication via UDP. This data type is used for the "TUSEND" and "TURSV" instructions, for example.
TCON_Param	64	Specifies the structure of a data block which stores descriptions of connections for Open User Communication via Industrial Ethernet (PROFINET). This data type is used for the "TSEND" and "TRSV" instructions, for example.

See also

Overview of the valid data types (Page 2236)

Hardware data types

Hardware data types

Description

The hardware data types are made available by the CPU. The number of available hardware data types depends on the CPU.

Constants of a specific hardware data type are stored based on the modules set in the hardware configuration. When an instruction for controlling or activating a configured module is inserted in the user program, the available constants can be used for the parameters.

The following table shows the available hardware data types and their purpose:

Data type	Basic data type	Description
REMOTE	ANY	Serves to specify the address of a remote CPU. This data type is used, for example, for the "PUT" and "GET" instructions.
GEOADDR	HW_IOSYSTEM	Geographical address information
HW_ANY	WORD	Identification of any hardware component, e.g. a module.
HW_DEVICE	HW_ANY	Identification of a DP slave/PROFINET IO device
HW_DPMASTER	HW_INTERFACE	Identification of a DP master
HW_DPSLAVE	HW_DEVICE	Identification of a DP slave
HW_IO	HW_ANY	Identification number of the CPU or the interface The number is automatically allocated and is stored in the properties of the CPU or of the interface in the hardware configuration.
HW_IOSYSTEM	HW_ANY	Identification of a PN/IO system or DP master system
HW_SUBMODULE	HW_IO	Identification of a central hardware component
HW_MODULE	HW_IO	Identification of a module
HW_INTERFACE	HW_SUBMODULE	Identification of an interface component
HW_IEPORT	HW_SUBMODULE	Identification of a port (PN/IO)
HW_HSC	HW_SUBMODULE	Identification of a high-speed counter This data type is used, for example, for the "CTRL_HSC" instruction.
HW_PWM	HW_SUBMODULE	Identification of a pulse width modulation This data type is used, for example, for the "CTRL_PWM" instruction.
HW_PTO	HW_SUBMODULE	Identification of a pulse encoder This data type is used for Motion Control.
AOM_AID	DWORD	Is used only in connection with a system function block.
AOM_IDENT	DWORD	Identification of an object in the runtime system of the AS
EVENT_ANY	AOM_IDENT	Used to identify any event
EVENT_ATT	EVENT_ANY	Is used to specify an event that can be assigned dynamically to an OB This data type is used, for example, for the "ATTACH" and "DETACH" instructions.
EVENT_HWINT	EVENT_ATT	Is used to specify a hardware interrupt event
OB_ANY	INT	Serves to specify any organization block.
OB_DELAY	OB_ANY	Used to specify an organization block that is called when a time-delay interrupt occurs. This data type is used, for example, for the "SRT_DINT" and "CAN_DINT" instructions.
OB_TOD	OB_ANY	Specifies the number of a time-of-day interrupt OB. This data type is used, for example, for the "SET_TINT", "CAN_TINT", "ACT_TINT" and "QRY_TINT" instructions.

Data type	Basic data type	Description
OB_CYCLIC	OB_ANY	Is used to specify an organization block that is called when a watchdog interrupt occurs.
OB_ATT	OB_ANY	Is used to specify an organization block that can be assigned dynamically to an event. This data type is used, for example, for the "ATTACH" and "DETACH" instructions.
OB_PCYCLE	OB_ANY	Is used to specify an organization block that can be assigned to an event of the "Cyclic program" event class.
OB_HWINT	OB_ATT	Is used to specify an organization block that is called when a hardware interrupt occurs.
OB_DIAG	OB_ANY	Is used to specify an organization block that is called when a diagnostic interrupt occurs.
OB_TIMEERROR	OB_ANY	Is used to specify an organization block that is called when a time error occurs.
OB_STARTUP	OB_ANY	Is used to specify an organization block that is called when a startup event occurs.
PORT	HW_SUBMODULE	Serves to specify a communication port. This data type is used for point-to-point communication.
RTM	UINT	Serves to specify the number of an operating hours counter. This data type is used, for example, for the "RTM" instruction.
PIP	UINT	Is used to create and connect a "Synchronous Cycle" OB. This data type is used for the SFCs 26, 27, 126 and 127.
CONN_ANY	WORD	Serves to specify any connection.
CONN_PRG	CONN_ANY	Serves to specify a connection for open communication over UDP.
CONN_OUC	CONN_ANY	Used to specify a connection for open communication over Industrial Ethernet (PROFINET).
CONN_R_ID	DWORD	Data type for the R_ID parameter on the S7 communication blocks.
DB_ANY	UINT	Identification (number) of any DB The data type "DB_ANY" has the length 0 in the section "Temp".
DB_WWW	DB_ANY	Number of a DB generated via the Web application (for example, "WWW" instruction) The data type "DB_WWW" has the length 0 in the section "Temp".
DB_DYN	DB_ANY	Number of a DB generated by the user program

See also

Overview of the valid data types (Page 2236)

Data type conversion

Data type conversion

Overview of data type conversion

Introduction

If you combine several operands in an instruction, you must make sure that the data types are compatible. This also applies when you assign or supply block parameters. If the operands are not of the same data type, a conversion has to be carried out.

There are two options for conversion:

- Implicit conversion
  Implicit conversion is supported by the programming languages LAD, FBD, SCL and GRAPH. Implicit conversion is not possible in the STL programming language.
- Explicit conversion
  You use an explicit conversion instruction before the actual instruction is executed.

Note

Converting bit strings to SCL

All bit strings (BYTE, WORD, DWORD and LWORD) are handled like the corresponding unsigned integers (USINT, UINT, UDINT and ULINT) in expressions. Therefore, implicit conversion from DWORD to REAL is carried out like a conversion from UDINT to REAL, for example.

Implicit conversion

Implicit conversion is executed automatically if the data types of the operands are compatible. This compatibility test can be carried out according to strict or less strict criteria:

- With IEC check (default setting)
  If IEC check is set, the following rules are applied:
  - Implicit conversion of BOOL into other data types is not possible.
  - Only the data types REAL, BYTE, WORD, DWORD, DINT, INT, SINT, UDINT, UINT, USINT, TIME, LDT, DTL, DT, TOD, WCHAR and CHAR can be converted implicitly.
  - The bit length of the source data type may not exceed the bit length of the destination data type. A WORD data type operand, for example, cannot be specified at a parameter at which the BYTE data type is expected.
- Without IEC check
  If IEC check is not set, the following rules are applied:
  - Implicit conversion of BOOL into other data types is not possible.
  - Only the data types REAL, LREAL, BYTE, WORD, DWORD, LWORD, SINT, INT, DINT, LINT, USINT, UINT, UDINT, ULINT, TIME, LTIME, S5TIME, LDT, DTL, TOD, LTOD, DATE, STRING, WSTRING, WCHAR and CHAR can be converted implicitly.
  - The bit length of the source data type may not exceed the bit length of the destination data type. A DWORD data type operand, for example, cannot be specified at a parameter at which the WORD data type is expected.
  - The bit length of an operand entered at in-out parameters (InOut) must be the same as the programmed bit length for the parameter in question.

  Note

  Implicit conversion without IEC check

  The programming editor uses a gray rectangle to mark operands that are implicitly converted. The dark gray rectangle signals that an implicit conversion is possible without any accuracy loss, for example, if you convert the data type INT to DINT. A light gray rectangle signals that implicit conversion is possible, but errors could occur during runtime. If, for example, you are converting the data type LINT to DINT and an overflow occurs, the enable output ENO is set to "0".

For more information about the setting of the IEC check and the implicit conversion, refer to "See also".

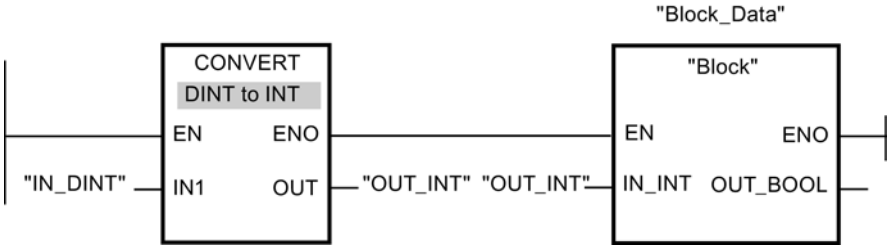
Explicit conversion

If the operands are not compatible and implicit conversion is therefore not possible, you can perform an explicit conversion. You can do this by using the conversion instructions in the "Instructions" Task Card or you can insert the conversion into the program manually. You can find the format for explicit conversion functions under "See also".

Any overflow will be displayed on the ENO enable output. An overflow occurs, for example, if the value of the source data type is greater than the value of the destination data type.

You can find additional information on explicit conversion under "See also".

The following figure shows an example in which explicit data type conversion must be carried out:



The "Block" function block expects a tag of the INT data type at the "IN_INT" input parameter. The value of the "IN_DINT" tag has therefore to the converted first from DINT to INT. Conversion is performed if the value of the "IN_DINT" tag is within the admissible value range for the INT data type. Otherwise an overflow is reported. However, a conversion takes place even in the event of an overflow, but the values are truncated and the ENO enable output is set to "0".

See also

Implicit conversion (Page 2288)

Explicit conversion (Page 2322)

Implicit conversion

Setting and disabling the IEC check

The data types of the operands used are checked for compatibility. This compatibility test can be carried out according to criteria that are more or less strict. If "IEC check" is activated, stricter criteria are applied.

You can set the IEC check centrally for all new blocks of the project or for individual blocks.

Setting IEC check for new blocks

To set the IEC check for all new blocks in the project, proceed as follows:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "PLC programming > General" group in the area navigation.
3. Select or clear the "IEC check" check box in the "Default settings for new blocks" group.
   The IEC check is enabled or disabled for all new blocks in the program.

Setting IEC check for a block

To set the IEC check for a block, proceed as follows:

1. Open the block.
2. Open the "Properties" tab in the inspector window.
3. Select the "Attributes" group in the area navigation.
4. Select or clear the "IEC check" check box.
   The IEC check is enabled or disabled for this block. The setting is stored together with the project.

Binary numbers

Implicit conversion of BOOL

Implicit conversion options

The BOOL data type cannot be implicitly converted.

See also

BOOL (bit) (Page 2240)

Overview of data type conversion (Page 2286)

Setting and disabling the IEC check (Page 2288)

Explicit conversion of BOOL (Page 2322)

Bit strings

Implicit conversion of BYTE

Implicit conversion options

The following table shows the options for the implicit conversion of BYTE data type:

Source	Destination	With IEC check	Without IEC check	Explanation
BYTE	BOOL	-	-	No implicit conversion
	WORD	X	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	DWORD	X	X	
	LWORD	X	X	
	SINT	-	X	
	USINT	-	X	
	INT	-	X	

Source	Destination	With IEC check	Without IEC check	Explanation
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	LINT	-	X	
	ULINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	TIME	-	-	
	LTIME	-	-	
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

BYTE (byte) (Page 2240)
Setting and disabling the IEC check (Page 2288)
Overview of data type conversion (Page 2286)
Explicit conversion of BYTE (Page 2323)

Implicit conversion of WORD

Implicit conversion options

The following table shows the options for the implicit conversion of WORD data type:

Source	Destination	With IEC check	Without IEC check	Explanation
WORD	BOOL	-	-	No implicit conversion
	BYTE	-	X	The low byte is transferred to the destination data type; the high byte is ignored.

Source	Destination	With IEC check	Without IEC check	Explanation
	DWORD	X	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	LWORD	X	X	
	SINT	-	X	The low byte is transferred to the destination data type; the high byte is ignored.
	USINT	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	LINT	-	X	
	ULINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	TIME	-	-	
	LTIME	-	-	
	S5TIME	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	LDT	-	-	No implicit conversion
	DTL	-	-	
	DT	-	-	
	DATE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	TOD	-	-	No implicit conversion
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

WORD (Page 2241)

Setting and disabling the IEC check (Page 2288)

Overview of data type conversion (Page 2286)

Explicit conversion of WORD (Page 2325)

Implicit conversion of DWORD

Implicit conversion options

The following table shows the options for the implicit conversion of DWORD data type:

Source	Destination	With IEC check	Without IEC check	Explanation
DWORD	BOOL	-	-	No implicit conversion
	BYTE	-	X	The right byte is transferred to the destination data type; the left byte is ignored.
	WORD	-	X	
	LWORD	X	X	
	SINT	-	X	
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	UDINT	-	X	
	LINT	-	X	
	ULINT	-	X	
	REAL	-	X	The value is converted to the destination data type format. (The value "-1", for example, is converted to the value "-1.0".)
	LREAL	-	-	No implicit conversion
	TIME	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	LTIME	-	-	No implicit conversion
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	LTOD	-	-	No implicit conversion
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

DWORD (Page 2242)

Setting and disabling the IEC check (Page 2288)

Overview of data type conversion (Page 2286)
Explicit conversion of DWORD (Page 2327)

Implicit conversion of LWORD

Implicit conversion options

The following table shows the options for the implicit conversion of LWORD data type:

Source	Destination	With IEC check	Without IEC check	Explanation
LWORD	BOOL	-	-	No implicit conversion
	BYTE	-	X	The right byte is transferred to the destination data type; the left byte is ignored.
	WORD	-	X	
	DWORD	-	X	
	SINT	-	X	
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	LINT	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	ULINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	X	The value is converted to the destination data type format. (The value "-1", for example, is converted to the value "-1.0".)
	TIME	-	-	No implicit conversion
	LTIME	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	S5TIME	-	-	No implicit conversion
	LDT	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	DTL	-	-	No implicit conversion
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	STRING	-	-	No implicit conversion
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 2288)
Overview of data type conversion (Page 2286)
LWORD (Page 2243)
Explicit conversion of LWORD (Page 2329)

Integers

Implicit conversion of SINT

Implicit conversion options

The following table shows the options for the implicit conversion of SINT data type:

Source	Destination	With IEC check	Without IEC check	Explanation
SINT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. The remaining bits are filled with "0".
	WORD	-	X	
	DWORD	-	X	
	LWORD	-	X	
	USINT	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value transfer from SINT #-1 -> INT #-1, not filled with "0".)
	INT	X	X	
	UINT	-	X	
	DINT	X	X	
	UDINT	-	X	
	LINT	X	X	
	ULINT	-	X	
	REAL	X	X	The value is converted to the destination data type format. (The value "-1", for example, is converted to the value "-1.0".)
	LREAL	X	X	
	TIME	-	-	No implicit conversion
	LTIME	-	-	
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	

Source	Destination	With IEC check	Without IEC check	Explanation
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 2288)
Overview of data type conversion (Page 2286)
SINT (8-bit integers) (Page 2244)
Explicit conversion of SINT (Page 2332)

Implicit conversion of USINT

Implicit conversion options

The following table shows the options for the implicit conversion of USINT data type:

Source	Destination	With IEC check	Without IEC check	Explanation
USINT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. The remaining bits are filled with "0".
	WORD	-	X	
	DWORD	-	X	
	LWORD	-	X	
	SINT	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from USINT #10 -> DINT #10 or USINT #128 -> SINT #-128)
	INT	X	X	
	UINT	X	X	
	DINT	X	X	
	UDINT	X	X	
	LINT	X	X	
	ULINT	X	X	
	REAL	X	X	The value is converted to the destination data type format. (The value "-1", for example, is converted to the value "-1.0".)
	LREAL	X	X	
	TIME	-	-	No implicit conversion
	LTIME	-	-	
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	

Source	Destination	With IEC check	Without IEC check	Explanation
	TOD	-	-	
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 2288)

Overview of data type conversion (Page 2286)

USINT (8-bit integers) (Page 2245)

Explicit conversion of USINT (Page 2335)

Implicit conversion of INT

Implicit conversion options

The following table shows the options for the implicit conversion of INT data type:

Source	Destination	With IEC check	Without IEC check	Explanation
INT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WORD	-	X	
	DWORD	-	X	
	LWORD	-	X	
	SINT	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from INT #-1 -> SINT #-1, or INT #-32 767 -> UINT #32 769)
	USINT	-	X	
	UINT	-	X	
	DINT	X	X	
	UDINT	-	X	
	LINT	X	X	
	ULINT	-	X	
	REAL	X	X	The value is converted to the destination data type format. (The value "-1", for example, is converted to the value "-1.0".)
	LREAL	X	X	
	TIME	-	-	No implicit conversion
	LTIME	-	-	
	S5TIME	-	-	

Source	Destination	With IEC check	Without IEC check	Explanation
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	TOD	-	-	No implicit conversion
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 2288)
Overview of data type conversion (Page 2286)
INT (16-bit integers) (Page 2245)
Explicit conversion of INT (Page 2337)

Implicit conversion of UINT

Implicit conversion options

The following table shows the options for the implicit conversion of UINT data type:

Source	Destination	With IEC check	Without IEC check	Explanation
UINT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WORD	-	X	
	DWORD	-	X	
	LWORD	-	X	
	SINT	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from UINT #100 -> DINT #100 or UINT #60 000 -> INT #-5536)
	USINT	-	X	
	INT	-	X	
	DINT	X	X	
	UDINT	X	X	
	LINT	X	X	
	ULINT	X	X	

Source	Destination	With IEC check	Without IEC check	Explanation
	REAL	X	X	The value is converted to the destination data type format. (The value "-1", for example, is converted to the value "-1.0".)
	LREAL	X	X	
	TIME	-	-	No implicit conversion
	LTIME	-	-	
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	TOD	-	-	No implicit conversion
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 2288)

Overview of data type conversion (Page 2286)

UINT (16-bit integers) (Page 2246)

Explicit conversion of UINT (Page 2339)

Implicit conversion of DINT

Implicit conversion options

The following table shows the options for the implicit conversion of DINT data type:

Source	Destination	With IEC check	Without IEC check	Explanation
DINT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WORD	-	X	
	DWORD	-	X	
	LWORD	-	X	
	SINT	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from DINT #-1 -> SINT #-1 or DINT #-1 -> USINT #255)
	USINT	-	X	

Source	Destination	With IEC check	Without IEC check	Explanation
	INT	-	X	
	UINT	-	X	
	UDINT	-	X	
	LINT	X	X	
	ULINT	-	X	
	REAL	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from DINT #-1 -> REAL #-1.0, but there is a loss in accuracy for numbers with an absolute value greater than 8 388 608)
	LREAL	X	X	The value is converted to the destination data type format. (The value "-1", for example, is converted to the value "-1.0".)
	TIME	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	LTIME	-	-	No implicit conversion
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	LTOD	-	-	No implicit conversion
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 2288)

Overview of data type conversion (Page 2286)

DINT (32-bit integers) (Page 2247)

Explicit conversion of DINT (Page 2341)

Implicit conversion of UDINT

Implicit conversion options

The following table shows the options for the implicit conversion of UDINT data type:

Source	Destination	With IEC check	Without IEC check	Explanation
UDINT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WORD	-	X	
	DWORD	-	X	
	LWORD	-	X	
	SINT	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from DINT #-1 -> SINT #-1 or DINT #-1 -> USINT #255)
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	LINT	X	X	
	ULINT	X	X	
	REAL	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from DINT #-1 -> REAL #-1.0, but there is a loss in accuracy for numbers with an absolute value greater than 8 388 608)
	LREAL	X	X	The value is converted to the destination data type format. (The value "-1", for example, is converted to the value "-1.0".)
	TIME	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	LTIME	-	-	No implicit conversion
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	LTOD	-	-	No implicit conversion
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 2288)
Overview of data type conversion (Page 2286)
UDINT (32-bit integers) (Page 2248)
Explicit conversion of UDINT (Page 2344)

Implicit conversion of LINT

Implicit conversion options

The following table shows the options for the implicit conversion of LINT data type:

Source	Destination	With IEC check	Without IEC check	Explanation
LINT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WORD	-	X	
	DWORD	-	X	
	LWORD	-	X	
	SINT	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from LINT #-1 -> SINT #-1 or LINT #-1 -> USINT #255)
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	ULINT	-	X	
	REAL	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from LINT #-1 -> REAL #-1.0, but there is a loss in accuracy for numbers with an absolute value greater than 8 388 608)
	LREAL	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from LINT #-1 -> REAL #-1.0, but there is a loss in accuracy for numbers with an absolute value greater than 9 007 199 254 740 992)
	TIME	-	-	No implicit conversion
	LTIME	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. (duration in nanoseconds)
	S5TIME	-	-	No implicit conversion
	LDT	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. (nanoseconds since 1/1/1970)
	DTL	-	-	No implicit conversion
	DT	-	-	
	DATE	-	-	

Source	Destination	With IEC check	Without IEC check	Explanation
	TOD	-	-	
	LTOD	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	STRING	-	-	No implicit conversion
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 2288)
Overview of data type conversion (Page 2286)
LINT (64-bit integers) (Page 2249)
Explicit conversion of LINT (Page 2346)

Implicit conversion of ULINT

Implicit conversion options

The following table shows the options for the implicit conversion of ULINT data type:

Source	Destination	With IEC check	Without IEC check	Explanation
ULINT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WORD	-	X	
	DWORD	-	X	
	LWORD	-	X	
	SINT	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from ULINT #-1 -> SINT #-1 or ULINT #-1 -> USINT #255)
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	LINT	-	X	
	REAL	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from LINT #-1 -> REAL #-1.0, but there is a loss in accuracy for numbers with an absolute value greater than 8 388 608)

Source	Destination	With IEC check	Without IEC check	Explanation
	LREAL	-	X	The bit pattern of the source value is converted and transferred to the destination data type. (for example, value conversion from LINT #-1 -> REAL #-1.0, but there is a loss in accuracy for numbers with an absolute value greater than 9 007 199 254 740 992)
	TIME	-	-	No implicit conversion
	LTIME	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. (duration in nanoseconds)
	S5TIME	-	-	No implicit conversion
	LDT	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. (nanoseconds since 1/1/1970)
	DTL	-	-	No implicit conversion
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	STRING	-	-	No implicit conversion
	WSTRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	WCHAR	-	X	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 2288)

Overview of data type conversion (Page 2286)

ULINT (64-bit integers) (Page 2250)

Explicit conversion of ULINT (Page 2349)

Floating-point numbers

Implicit conversion of REAL

Implicit conversion options

The following table shows the options for the implicit conversion of REAL data type:

Source	Destination	With IEC check	Without IEC check	Explanation
REAL	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	LWORD	-	-	No implicit conversion
	SINT	-	X	The bit pattern of the source value is rounded off and converted and transferred to the destination data type. (For example, rounding off and value conversion of REAL #2.5 -> INT #2, or negative numbers REAL #-2.5 -> INT #-2 -> USINT #254. With an overflow, the remainder is determined REAL #305.5 -> INT #306 -> USINT #50)
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	LINT	-	X	
	ULINT	-	X	
	LREAL	X	X	The value is transferred to the destination data type.
	TIME	-	-	No implicit conversion
	LTIME	-	-	
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 2288)
Overview of data type conversion (Page 2286)
REAL (Page 2252)
Explicit conversion of REAL (Page 2351)

Implicit conversion of LREAL

Implicit conversion options

The following table shows the options for the implicit conversion of LREAL data type:

Source	Destination	With IEC check	Without IEC check	Explanation
LREAL	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	-	
	LWORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	SINT	-	X	The bit pattern of the source value is rounded off and converted and transferred to the destination data type. (For example, rounding off and value conversion of LREAL #2.5 -> INT #2, or negative numbers LREAL #-2.5 -> INT #-2 -> USINT #254. With an overflow, the remainder is determined LREAL #305.5 -> INT #306 -> USINT #50)
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	LINT	-	X	
	ULINT	-	X	
	REAL	-	X	The value is transferred to the destination data type.
	TIME	-	-	No implicit conversion
	LTIME	-	-	
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	-	

Source	Destination	With IEC check	Without IEC check	Explanation
	WCHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 2288)
Overview of data type conversion (Page 2286)
LREAL (Page 2253)
Explicit conversion of LREAL (Page 2352)

Timers

Implicit conversion of S5TIME

Implicit conversion options

The following table shows the options for the implicit conversion of data type S5TIME:

Source	Destination	With IEC check	Without IEC check	Explanation
S5TIME	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of conversion shows the duration in milliseconds.
	DWORD	-	-	No implicit conversion
	LWORD	-	-	
	SINT	-	-	
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	-	
	UDINT	-	-	
	LINT	-	-	
	ULINT	-	-	
	REAL	-	-	
	LREAL	-	-	
	TIME	-	-	
	LTIME	-	-	
	LDT	-	-	

Source	Destination	With IEC check	Without IEC check	Explanation
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 2288)
Overview of data type conversion (Page 2286)
S5TIME (duration) (Page 2255)
Explicit conversion of S5TIME (Page 2354)

Implicit conversion of TIME

Implicit conversion options

The following table shows the options for the implicit conversion of TIME data type:

Source	Destination	With IEC check	Without IEC check	Explanation
TIME	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of conversion shows the duration in milliseconds.
	LWORD	-	-	No implicit conversion
	SINT	-	-	
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of conversion shows the duration in milliseconds.
	UDINT	-	X	
	LINT	-	-	No implicit conversion

Source	Destination	With IEC check	Without IEC check	Explanation
	ULINT	-	-	
	REAL	-	-	
	LREAL	-	-	
	S5TIME	-	-	
	LTIME	X	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion shows the duration in nanoseconds. (1 ms = 1 000 000 ns)
	LDT	-	-	No implicit conversion
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	X	If the source value is between 0 s and 84599.999 s, the bit pattern of the source value is transferred unchanged to the destination data type. (Display in nanoseconds) The target value remains unchanged otherwise. The result of the conversion shows the time that has passed since midnight.
	LTOD	-	-	No implicit conversion
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 2288)

Overview of data type conversion (Page 2286)

TIME (IEC time) (Page 2257)

Explicit conversion of TIME (Page 2355)

Implicit conversion of LTIME

Implicit conversion options

The following table shows the options for the implicit conversion of LTIME data type:

Source	Destination	With IEC check	Without IEC check	Explanation
LTIME	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	

Source	Destination	With IEC check	Without IEC check	Explanation
	DWORD	-	-	
	LWORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion shows the duration in nanoseconds.
	SINT	-	-	No implicit conversion
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	-	
	UDINT	-	-	
	LINT	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion shows the duration in nanoseconds.
	ULINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	S5TIME	-	-	
	TIME	-	X	If the source value is outside the value range of the destination data type, the target value remains unchanged. (0.123456789 s becomes 0.123 s)
	LDT	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion shows the duration in nanoseconds since 1/1/1970.
	DTL	-	-	No implicit conversion
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion shows the duration in nanoseconds since 1/1/1970.
	STRING	-	-	No implicit conversion
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 2288)

Overview of data type conversion (Page 2286)

LTIME (IEC time) (Page 2257)

Explicit conversion of LTIME (Page 2357)

Date and time-of-day

Implicit conversion of DT

Implicit conversion options

The following table shows the options for the implicit conversion of DT data type:

Source	Destination	With IEC check	Without IEC check	Explanation
DT	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	-	
	LWORD	-	-	
	SINT	-	-	
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	-	
	UDINT	-	-	
	LINT	-	-	
	ULINT	-	-	
	REAL	-	-	
	LREAL	-	-	
	TIME	-	-	
	LTIME	-	-	
	S5TIME	-	-	
	LDT	X	X	The source value is transferred unchanged with the same value to the destination data type. (12/24/2012 14:30 remains 12/24/2012 14:30)
	DTL	X	X	
	DATE	-	-	No implicit conversion
	TOD	-	-	
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 2288)
Overview of data type conversion (Page 2286)
DATE_AND_TIME (date and time of day) (Page 2260)
Explicit conversion of DT (Page 2358)

Implicit conversion of LDT

Implicit conversion options

The following table shows the options for the implicit conversion of LDT data type:

Source	Destination	With IEC check	Without IEC check	Explanation
LDT	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	-	
	LWORD	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. (nanoseconds since 1/1/1970)
	SINT	-	-	No implicit conversion
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	-	
	UDINT	-	-	
	LINT	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. (nanoseconds since 1/1/1970)
	ULINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	TIME	-	-	
	LTIME	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. (nanoseconds since 1/1/1970)
	S5TIME	-	-	No implicit conversion
	DT	-	X	The bit pattern of the source value is converted with a loss in accuracy to the destination data type. (12/24/2012 12:34:56.123456789 becomes 12/24/2012 12:34:56.123)
	DTL	X	X	The source value is transferred unchanged with the same value to the destination data type. (12/24/2012 14:30 remains 12/24/2012 14:30)
	DATE	-	-	No implicit conversion
	TOD	-	-	

Source	Destination	With IEC check	Without IEC check	Explanation
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 2288)
Overview of data type conversion (Page 2286)
Explicit conversion of LDT (Page 2360)
LDT (DATE_AND_LTIME) (Page 2261)

Implicit conversion of DTL

Implicit conversion options

The following table shows the options for the implicit conversion of DTL data type:

Source	Destination	With IEC check	Without IEC check	Explanation
DTL	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	-	
	LWORD	-	-	
	SINT	-	-	
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	-	
	UDINT	-	-	
	LINT	-	-	
	ULINT	-	-	
	REAL	-	-	
	LREAL	-	-	
	TIME	-	-	
	LTIME	-	-	
	S5TIME	-	-	

Source	Destination	With IEC check	Without IEC check	Explanation
	LDT	X	X	The source value is transferred unchanged with the same value to the destination data type. (12/24/2012 14:30 remains 12/24/2012 14:30)
	DT	-	X	The bit pattern of the source value is converted with a loss in accuracy to the destination data type. (12/24/2012 12:34:56.123456789 becomes 12/24/2012 12:34:56.123)
	DATE	-	-	No implicit conversion
	TOD	-	-	
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 2288)

Overview of data type conversion (Page 2286)

DTL (Page 2262)

Explicit conversion of DTL (Page 2361)

Implicit conversion of DATE

Implicit conversion options

The following table shows the options for the implicit conversion of data type DATE:

Source	Destination	With IEC check	Without IEC check	Explanation
DATE	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of conversion is equal to the number of days since 01/01/1990.
	DWORD	-	-	No implicit conversion
	LWORD	-	-	
	SINT	-	-	
	USINT	-	-	
	INT	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of conversion is equal to the number of days since 01/01/1990.
	UINT	-	X	

Source	Destination	With IEC check	Without IEC check	Explanation
	DINT	-	-	No implicit conversion
	UDINT	-	-	
	LINT	-	-	
	ULINT	-	-	
	REAL	-	-	
	LREAL	-	-	
	TIME	-	-	
	LTIME	-	-	
	S5TIME	-	-	
	LDT	-	-	
	DT	-	-	
	DTL	-	-	
	TOD	-	-	
	LTOD	-	-	
	STRING	-	-	
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 2288)
Overview of data type conversion (Page 2286)
DATE (Page 2258)
Explicit conversion of DATE (Page 2362)

Implicit conversion of TOD

Implicit conversion options

The following table shows the options for the implicit conversion of TOD data type:

Source	Destination	With IEC check	Without IEC check	Explanation
TOD	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	

Source	Destination	With IEC check	Without IEC check	Explanation
	DWORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion corresponds to the number of milliseconds since the start of day (0:00).
	LWORD	-	-	No implicit conversion
	SINT	-	-	
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion corresponds to the number of milliseconds since the start of day (0:00).
	UDINT	-	X	
	LINT	-	-	No implicit conversion
	ULINT	-	-	
	REAL	-	-	
	LREAL	-	-	
	TIME	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion corresponds to the number of milliseconds since the start of day (0:00).
	LTIME	-	-	No implicit conversion
	S5TIME	-	-	
	LDT	-	-	
	DT	-	-	
	DTL	-	-	
	DATE	-	-	
	LTOD	X	X	The source value is transferred unchanged with the same value to the destination data type. (12:34:56.123 becomes 12:34:56.123000000)
	STRING	-	-	No implicit conversion
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 2288)

Overview of data type conversion (Page 2286)

TIME_OF_DAY (TOD) (Page 2259)

Explicit conversion of TOD (Page 2364)

Implicit conversion of LTOD

Implicit conversion options

The following table shows the options for the implicit conversion of LTOD data type:

Source	Destination	With IEC check	Without IEC check	Explanation
LTOD	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	-	
	LWORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion corresponds to the number of nanoseconds since the start of day (0:00 hrs).
	SINT	-	-	No implicit conversion
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	-	
	UDINT	-	-	
	LINT	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion corresponds to the number of nanoseconds since the start of day (0:00 hrs).
	ULINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	TIME	-	-	
	LTIME	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion corresponds to the number of nanoseconds since the start of day (0:00 hrs).
	S5TIME	-	-	No implicit conversion
	LDT	-	-	
	DT	-	-	
	DTL	-	-	
	DATE	-	-	
	TOD	-	X	The source value is transferred unchanged with the same value rounded off to the destination data type. (12:34:56.123456789 becomes 12:34:56.123)
	STRING	-	-	No implicit conversion
	WSTRING	-	-	
	CHAR	-	-	
	WCHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 2288)
Overview of data type conversion (Page 2286)
Explicit conversion of LTOD (Page 2365)
LTOD (LTIME_OF_DAY) (Page 2259)

String

Implicit conversion of CHAR

Implicit conversion options

The following table shows the options for the implicit conversion of CHAR data type:

Source	Destination	With IEC check	Without IEC check	Explanation
CHAR	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. The remaining bits are filled from the left with "0".
	WORD	-	X	
	DWORD	-	X	
	LWORD	-	X	
	SINT	-	X	
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	LINT	-	X	
	ULINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	TIME	-	-	
	LTIME	-	-	
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	-	
	WCHAR	-	-	
	STRING	X	X	The STRING is shortened to length 1 and includes the character.

Source	Destination	With IEC check	Without IEC check	Explanation
	WSTRING	-	-	No implicit conversion
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 2288)
Overview of data type conversion (Page 2286)
CHAR (character) (Page 2263)
Explicit conversion of CHAR (Page 2367)

Implicit conversion of WCHAR

Implicit conversion options

The following table shows the options for the implicit conversion of WCHAR data type:

Source	Destination	With IEC check	Without IEC check	Explanation
WCHAR	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. The remaining bits are filled from the left with "0".
	WORD	-	X	
	DWORD	-	X	
	LWORD	-	X	
	SINT	-	X	
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	LINT	-	X	
	ULINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	TIME	-	-	
	LTIME	-	-	
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	

Source	Destination	With IEC check	Without IEC check	Explanation
	TOD	-	-	
	LTOD	-	-	
	CHAR	-	-	
	STRING	-	-	
	WSTRING	X	X	The WSTRING is shortened to length 1 and includes the character.
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 2288)
Overview of data type conversion (Page 2286)
WCHAR (character) (Page 2263)
Explicit conversion of WCHAR (Page 2368)

Implicit conversion of STRING

Implicit conversion options

The following table shows the options for the implicit conversion of STRING data type:

Source	Destination	With IEC check	Without IEC check	Explanation
STRING	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	-	
	LWORD	-	-	
	SINT	-	-	
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	-	
	UDINT	-	-	
	LINT	-	-	
	ULINT	-	-	
	REAL	-	-	
	LREAL	-	-	
	TIME	-	-	
	LTIME	-	-	

Source	Destination	With IEC check	Without IEC check	Explanation
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	-	
	CHAR	-	X	The first character of the STRING is returned if the STRING includes one or more characters. Otherwise, the character is output with coding $00.
	WCHAR	-	-	No implicit conversion
	WSTRING	-	-	
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 2288)
Overview of data type conversion (Page 2286)
STRING (Page 2264)
Explicit conversion of STRING (Page 2369)

Implicit conversion of WSTRING

Implicit conversion options

The following table shows the options for the implicit conversion of WSTRING data type:

Source	Destination	With IEC check	Without IEC check	Explanation
WSTRING	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	-	
	LWORD	-	-	
	SINT	-	-	
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	-	
	UDINT	-	-	

Source	Destination	With IEC check	Without IEC check	Explanation
	LINT	-	-	
	ULINT	-	-	
	REAL	-	-	
	LREAL	-	-	
	TIME	-	-	
	LTIME	-	-	
	S5TIME	-	-	
	LDT	-	-	
	DTL	-	-	
	DT	-	-	
	DATE	-	-	
	TOD	-	-	
	LTOD	-	-	
	CHAR	-	-	
	WCHAR	-	X	The first character of the WSTRING is returned if the WSTRING includes one or more characters. Otherwise, the character is output with coding $0000.
	STRING	-	-	No implicit conversion
x: Conversion possible -: Conversion not possible				

See also

Setting and disabling the IEC check (Page 2288)

Overview of data type conversion (Page 2286)

WSTRING (Page 2265)

Explicit conversion of WSTRING (Page 2371)

Explicit conversion

Binary numbers

Explicit conversion of BOOL

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the BOOL data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
BOOL	BYTE	X	Only the LSB (Least Significant Bit) is set in the destination data type. The enable output ENO is always "1".	BOOL_TO_BYTE
	WORD	X		BOOL_TO_WORD
	DWORD	X		BOOL_TO_DWORD
	LWORD	X		BOOL_TO_LWORD
	SINT	X		BOOL_TO_SINT
	USINT	X		BOOL_TO_USINT
	INT	X		BOOL_TO_INT
	UINT	X		BOOL_TO_UINT
	DINT	X		BOOL_TO_DINT
	UDINT	X		BOOL_TO_UDINT
	LINT	X		BOOL_TO_LINT
	ULINT	X		BOOL_TO_ULINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	TIME	-		-
	LTIME	-		-
	S5TIME	-		-
	LDT	-		-
	DT	-		-
	DTL	-		-
	TOD	-		-
	LTOD	-		-
	DATE	-		-
	STRING	-		-
	WSTRING	-		-
	CHAR	-		-
	WCHAR	-		-
x: Conversion possible - : Conversion not possible				

See also

Overview of data type conversion (Page 2286)
Implicit conversion of BOOL (Page 2289)
BOOL (bit) (Page 2240)

Bit strings

Explicit conversion of BYTE

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the BYTE data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
BYTE[1)]	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bits are not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	BYTE_TO_BOOL
	WORD[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	BYTE_TO_WORD
	DWORD[1)]	X		BYTE_TO_DWORD
	LWORD[1)]	X		BYTE_TO_LWORD
	SINT	X		BYTE_TO_SINT
	USINT	X		BYTE_TO_USINT
	INT	X		BYTE_TO_INT
	UINT	X		BYTE_TO_UINT
	DINT	X		BYTE_TO_DINT
	UDINT	X		BYTE_TO_UDINT
	LINT	X		BYTE_TO_LINT
	ULINT	X		BYTE_TO_ULINT
	REAL	-	No explicit conversion	-
	LREAL	-		-

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	TIME	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	BYTE_TO_TIME
	LTIME	X		BYTE_TO_LTIME
	S5TIME	-	No explicit conversion	-
	LDT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	BYTE_TO_LDT
	DT	-	No explicit conversion	-
	DTL	-		-
	TOD	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	BYTE_TO_TOD
	LTOD	X		BYTE_TO_LTOD
	DATE	X		BYTE_TO_DATE
	STRING	-	No explicit conversion	-
	WSTRING	-		-
	CHAR	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	BYTE_TO_CHAR
	WCHAR	X		BYTE_TO_WCHAR
x: Conversion possible - : Conversion not possible 1) Bit strings (BYTE, WORD, DWORD, LWORD) are interpreted as an unsigned integer with the same bit length. The data type BYTE is interpreted as USINT, WORD as UINT, DWORD as UDINT and LWORD as ULINT.				

See also

Overview of data type conversion (Page 2286)

Implicit conversion of BYTE (Page 2289)

BYTE (byte) (Page 2240)

Explicit conversion of WORD

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the WORD data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
WORD[1)]	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bits are not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	WORD_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	WORD_TO_BYTE
	DWORD[1)]	X		WORD_TO_DWORD
	LWORD[1)]	X		WORD_TO_LWORD

	SINT	X	ENO = TRUE
			#sint1 := WORD_TO_SINT(16#FFFF); // -1 to
			#sint1 := WORD_TO_SINT(16#FF80); // -128
			#sint1 := WORD_TO_SINT(16#0); // 0 to
			#sint1 := WORD_TO_SINT(16#007F); // 127
			
			ENO = FALSE
			#sint1 := WORD_TO_SINT(16#FF7F); // -129 to
			#sint1 := WORD_TO_SINT(16#8000); // -32768
			#sint1 := WORD_TO_SINT(16#0080); // 128 to
			#sint1 := WORD_TO_SINT(16#7FFF); // 32767
				WORD_TO_SINT

	USINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	WORD_TO_USINT
	INT	X		WORD_TO_INT
	UINT	X		WORD_TO_UINT
	DINT	X		WORD_TO_DINT
	UDINT	X		WORD_TO_UDINT
	LINT	X		WORD_TO_LINT
	ULINT	X		WORD_TO_ULINT
	REAL	-	No explicit conversion	-
	LREAL	-		-

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	TIME	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	WORD_TO_TIME
	LTIME	X		WORD_TO_LTIME
	S5TIME	X		WORD_TO_S5TIME
	LDT	X		WORD_TO_LDT
	DT	-	No explicit conversion	-
	DTL	-		-
	TOD	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	WORD_TO_TOD
	LTOD	X		WORD_TO_LTOD
	DATE	X		WORD_TO_DATE
	STRING	-	No explicit conversion	-
	WSTRING	-		-
	CHAR	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	WORD_TO_CHAR
	WCHAR	X		WORD_TO_WCHAR
WORD_ BCD16	INT	X	The value to be converted has data type WORD and is accepted as a BCD-coded value between -999 and +999. The result is available after conversion as an integer (in binary notation) of the type INT. A real conversion takes place. If the bit pattern includes an invalid tetrad, a synchronous error is not triggered but only the status bit OV is set instead.	WORD_BCD16_TO_INT
BCD16	INT	X		BCD16_TO_INT

x: Conversion possible

- : Conversion not possible

[1] Bit strings (BYTE, WORD, DWORD, LWORD) are interpreted as an unsigned integer with the same bit length. The data type BYTE is interpreted as USINT, WORD as UINT, DWORD as UDINT and LWORD as ULINT.

See also

Implicit conversion of WORD (Page 2290)

Overview of data type conversion (Page 2286)

WORD (Page 2241)

Explicit conversion of DWORD

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the DWORD data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
DWORD[1)]	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bits are not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	DWORD_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	DWORD_TO_BYTE
	WORD[1)]	X		DWORD_TO_WORD
	LWORD[1)]	X		DWORD_TO_LWORD

	SINT	X	ENO = TRUE
			#sint1 := DWORD_TO_SINT(16#FFFF_FFFF); // -1 to
			#sint1 := DWORD_TO_SINT(16#FFFF_FF80); // -128
			#sint1 := DWORD_TO_SINT(16#0); // 0 to
			#sint1 := DWORD_TO_SINT(16#0000_007F); // 127
			
			ENO = FALSE
			#sint1 := DWORD_TO_SINT(16#FFFF_FF7F); // -129
			#sint1 := DWORD_TO_SINT(16#8000_0000); // -2147483648
			#sint1 := DWORD_TO_SINT(16#0000_0080); // 128 to
			#sint1 := DWORD_TO_SINT(16#7FFF_FFFF); // 2147483647
				DWORD_TO_SINT

	USINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	DWORD_TO_USINT

Source	Destination	Conversion	Explanation	Mnemonics of the instruction

	INT	X	ENO = TRUE
			#int1 := DWORD_TO_INT(16#FFFF_FFFF); // -1 to
			#int1 := DWORD_TO_INT(16#FFFF_8000); // -32768
			#int1 := DWORD_TO_INT(16#0); // 0 to
			#int1 := DWORD_TO_INT(16#0000_7FFF); // 32767
			
			ENO = FALSE
			#int1 := DWORD_TO_INT(16#FFFF_7FFF); // -32769
			#int1 := DWORD_TO_INT(16#8000_0000); // -2147483648
			#int1 := DWORD_TO_INT(16#8000); // 32768 to
			#int1 := DWORD_TO_INT(16#7FFF_FFFF); // 2147483647
				DWORD_TO_INT

	UINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If no errors occur during the conversion, the signal state of ENO = 1; if an error occurs during processing, the signal state of ENO = 0.	DWORD_TO_UINT
	DINT	X		DWORD_TO_DINT
	UDINT	X		DWORD_TO_UDINT
	LINT	X		DWORD_TO_LINT
	ULINT	X		DWORD_TO_ULINT
	REAL	X		DWORD_TO_REAL
	LREAL	-	No explicit conversion	-
	TIME	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	DWORD_TO_TIME
	LTIME	X		DWORD_TO_LTIME
	S5TIME	-	No explicit conversion	-
	LDT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	DWORD_TO_LDT
	DT	-	No explicit conversion	-
	DTL	-		-
	TOD	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	DWORD_TO_TOD
	LTOD	X		DWORD_TO_LTOD
	DATE	X		DWORD_TO_DATE
	STRING	-	No explicit conversion	-
	WSTRING	-		-
	CHAR	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	DWORD_TO_CHAR

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	WCHAR	X		DWORD_TO_WCHAR
DWORD_BCD32	DINT	X	The value to be converted has data type DWORD and is accepted as a BCD-coded value between -9999999 and +9999999. The result is available after conversion as an integer (in binary notation) of the type DINT. A real conversion takes place. If the bit pattern includes an invalid tetrad, a synchronous error is not triggered but only the status bit OV is set instead.	DWORD_BCD32_TO_DINT
BCD32	DINT	X		BCD32_TO_DINT
x: Conversion possible - : Conversion not possible 1) Bit strings (BYTE, WORD, DWORD, LWORD) are interpreted as an unsigned integer with the same bit length. The data type BYTE is interpreted as USINT, WORD as UINT, DWORD as UDINT and LWORD as ULINT.				

See also

Implicit conversion of DWORD (Page 2292)

Overview of data type conversion (Page 2286)

DWORD (Page 2242)

Explicit conversion of LWORD

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the LWORD data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
LWORD[1)]	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bit is not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	LWORD_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LWORD_TO_BYTE
	WORD[1)]	X		LWORD_TO_WORD
	DWORD[1)]	X		LWORD_TO_DWORD

Source	Destination	Conversion	Explanation	Mnemonics of the instruction

	SINT	X	ENO = TRUE
			#sint1 := LWORD_TO_SINT(16#FFFF_FFFF_FFFF_FFFF); // -1 to
			#sint1 := LWORD_TO_SINT(16#FFFF_FFFF_FFFF_FF80); // -128
			#sint1 := LWORD_TO_SINT(16#0); // 0 to
			#sint1 := LWORD_TO_SINT(16#0000_0000_0000_007F); // 127
			
			ENO = FALSE
			#sint1 := LWORD_TO_SINT(16#FFFF_FFFF_FFFF_FF7F); // -129
			#sint1 := LWORD_TO_SINT(16#8000_0000_0000_0000); // -9223372036854775808
			#sint1 := LWORD_TO_SINT(16#0000_0000_0000_0080); // 128
			#sint1 := LWORD_TO_SINT(16#7FFF_FFFF_FFFF_FFFF); // 9223372036854775807
				LWORD_TO_SINT

	USINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LWORD_TO_USINT

	INT	X	ENO = TRUE
			#int1 := LWORD_TO_INT(16#FFFF_FFFF_FFFF_FFFF); // -1 to
			#int1 := LWORD_TO_INT(16#FFFF_FFFF_FFFF_8000); // -32768
			#int1 := LWORD_TO_INT(16#0); // 0 to
			#int1 := LWORD_TO_INT(16#0000_0000_0000_7FFF); // 32767
			
			ENO = FALSE
			#int1 := LWORD_TO_INT(16#FFFF_FFFF_FFFF_7FFF); // -32769
			#int1 := LWORD_TO_INT(16#8000_0000_0000_0000); // -2147483648
			#int1 := LWORD_TO_INT(16#0000_0000_0000_8000); // 32768 to
			#int1 := LWORD_TO_INT(16#7FFF_FFFF_FFFF_FFFF); // 2147483647
				LWORD_TO_INT

	UINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LWORD_TO_UINT

Source	Destination	Conversion	Explanation	Mnemonics of the instruction

	DINT	X	ENO = TRUE
			#dint1 := LWORD_TO_DINT(16#FFFF_FFFF_FFFF_FFFF); // -1 to
			#dint1 := LWORD_TO_DINT(16#FFFF_FFFF_8000_0000); // -2147483648
			#dint1 := LWORD_TO_DINT(16#0); // 0 to
			#dint1 := LWORD_TO_DINT(16#0000_0000_7FFF_FFFF); // 2147483647
			
			ENO = FALSE
			#dint1 := LWORD_TO_DINT(16#FFFF_FFFF_7FFF_FFFF); // -2147483649 to
			#dint1 := LWORD_TO_DINT(16#8000_0000_0000_0000); // -9223372036854775808
			#dint1 := LWORD_TO_DINT(16#0000_0000_8000_0000); // 2147483648 to
			#dint1 := LWORD_TO_DINT(16#7FFF_FFFF_FFFF_FFFF); // 9223372036854775807
				LWORD_TO_DINT

	UDINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LWORD_TO_UDINT
	LINT	X		LWORD_TO_LINT
	ULINT	X		LWORD_TO_ULINT
	REAL	-	No explicit conversion	-
	LREAL	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If no errors occur during the conversion, the signal state of ENO = 1; if an error occurs during processing, the signal state of ENO = 0.	LWORD_TO_LREAL
	TIME	X		LWORD_TO_TIME
	LTIME	X		LWORD_TO_LTIME
	S5TIME	-	No explicit conversion	-
	LDT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LWORD_TO_LDT
	DT	-	No explicit conversion	-
	DTL	-		-
	TOD	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LWORD_TO_TOD
	LTOD	X		LWORD_TO_LTOD
	DATE	X		LWORD_TO_DATE
	STRING	-	No explicit conversion	-
	WSTRING	-		-
	CHAR	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LWORD_TO_CHAR

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	WCHAR	X		LWORD_TO_WCHAR
x: Conversion possible - : Conversion not possible 1) Bit strings (BYTE, WORD, DWORD, LWORD) are interpreted as an unsigned integer with the same bit length. The data type BYTE is interpreted as USINT, WORD as UINT, DWORD as UDINT and LWORD as ULINT.				

See also

Implicit conversion of LWORD (Page 2293)

Overview of data type conversion (Page 2286)

LWORD (Page 2243)

Integers

Explicit conversion of SINT

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the SINT data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
SINT	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bit is not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	SINT_TO_BOOL
	BYTE1)	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0".	SINT_TO_BYTE
	WORD1)	X		SINT_TO_WORD
	DWORD1)	X		SINT_TO_DWORD
	LWORD1)	X		SINT_TO_LWORD

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	USINT	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0".	SINT_TO_USINT
	INT	X		SINT_TO_INT
	UINT	X		SINT_TO_UINT
	DINT	X		SINT_TO_DINT
	UDINT	X		SINT_TO_UDINT
	LINT	X		SINT_TO_LINT
	ULINT	X		SINT_TO_ULINT
	REAL	X	The value is converted to the destination data type format. (The value "-1" is changed to the value "-1.0", for example, with the instruction "Convert value" (CONVERT).	SINT_TO_REAL, NORM_X
	LREAL	X		SINT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the destination data type and interpreted as milliseconds.	SINT_TO_TIME
	LTIME	X	The value is transferred to the destination data type and interpreted as nanoseconds.	SINT_TO_LTIME
	S5TIME	-	No explicit conversion	-
	LDT	X	The result is returned in nanoseconds since 1970-1-1-0:0:0.0.	SINT_TO_LDT
	DT	-	No explicit conversion	-
	DTL	-		-
	TOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in milliseconds since 0:0)	SINT_TO_TOD
	LTOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in nanoseconds since 0:0)	SINT_TO_LTOD

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	DATE	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in days since 1990-1-1)	SINT_TO_DATE
	STRING	X	The value is converted into a character string. The character string is shown preceded by a sign. If the permitted length of the character string is violated, the enable output ENO is set to "0".	SINT_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		SINT_TO_WSTRING
	CHAR[1)]	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0".	SINT_TO_CHAR
	WCHAR[1)]	X		SINT_TO_WCHAR

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width including sign, and then the bits are copied. The source type determines the interpretation.

See also

Implicit conversion of SINT (Page 2294)

Overview of data type conversion (Page 2286)

SINT (8-bit integers) (Page 2244)

Explicit conversion of USINT

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the USINT data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
USINT	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bit is not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	USINT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	USINT_TO_BYTE
	WORD[1)]	X		USINT_TO_WORD
	DWORD[1)]	X		USINT_TO_DWORD
	LWORD[1)]	X		USINT_TO_LWORD
	SINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If the sign is changed during the conversion, the enable output ENO is set to "0".	USINT_TO_SINT
	INT	X	The bit pattern of the source value is converted and transferred to the destination data type.	USINT_TO_INT
	UINT	X		USINT_TO_UINT
	DINT	X		USINT_TO_DINT
	UDINT	X		USINT_TO_UDINT
	LINT	X		USINT_TO_LINT
	ULINT	X		USINT_TO_ULINT
	REAL	X	The value is converted to the destination data type format. (The value "1" is changed to the value "1.0", for example, with the instruction "Convert value" (CONVERT).	USINT_TO_REAL, NORM_X
	LREAL	X		USINT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the destination data type and interpreted as milliseconds.	USINT_TO_TIME

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	LTIME	X	The value is transferred to the destination data type and interpreted as nanoseconds.	USINT_TO_LTIME
	S5TIME	-	No explicit conversion	-
	LDT	X	The result is returned in nanoseconds since 1970-1-1-0:0:0.0.	USINT_TO_LDT
	DT	-	No explicit conversion	-
	DTL	-		-
	TOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (interpretation in milliseconds since 0:0)	USINT_TO_TOD
	LTOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (interpretation in nanoseconds since 0:0)	USINT_TO_LTOD
	DATE	X	The bit pattern of the source value is converted and transferred to the destination data type. (interpretation in days since 1990-1-1)	USINT_TO_DATE
	STRING	X	The value is converted into a character string. If the permitted length of the character string is violated, the enable output ENO is set to "0".	USINT_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		USINT_TO_WSTRING
	CHAR[1)]	X	The bit pattern of the source value is converted and transferred to the destination data type.	USINT_TO_CHAR
	WCHAR[1)]	X		USINT_TO_WCHAR

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width (the non-existing sign is replaced with zeros), and then the bits are copied. The source type determines the interpretation.

See also

Implicit conversion of USINT (Page 2295)

Overview of data type conversion (Page 2286)

USINT (8-bit integers) (Page 2245)

Explicit conversion of INT

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the INT data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
INT	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bit is not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	INT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0".	INT_TO_BYTE
	WORD[1)]	X		INT_TO_WORD
	DWORD[1)]	X		INT_TO_DWORD
	LWORD[1)]	X		INT_TO_LWORD
	SINT	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0".	INT_TO_SINT
	USINT	X		INT_TO_USINT
	UINT	X		INT_TO_UINT
	DINT	X		INT_TO_DINT
	UDINT	X		INT_TO_UDINT
	LINT	X		INT_TO_LINT
	ULINT	X		INT_TO_ULINT
	REAL	X	The value is converted to the destination data type format. (The value "-1" is changed to the value "-1.0", for example, with the instruction "Convert value" (CONVERT).	INT_TO_REAL, NORM_X
	LREAL	X		INT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the destination data type and interpreted as milliseconds.	INT_TO_TIME
	LTIME	X	The value is transferred to the destination data type and interpreted as nanoseconds.	INT_TO_LTIME
	S5TIME	-	No explicit conversion	-
	LDT	X	The result is returned in nanoseconds since 1970-1-1-0:0:0.0.	INT_TO_LDT
	DT	-	No explicit conversion	-

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	DTL	-		-
	TOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in milliseconds since 0:0; check for 24h limit)	INT_TO_TOD
	LTOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in nanoseconds since 0:0; check for 24h limit)	INT_TO_LTOD
	DATE	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in days since 1990-1-1; check for negative value)	INT_TO_DATE
	STRING	X	The value is converted into a character string. The character string is shown preceded by a sign. If the permitted length of the character string is violated, the enable output ENO is set to "0".	INT_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		INT_TO_WSTRING
	CHAR[1)]	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0".	INT_TO_CHAR
	WCHAR[1)]	X		INT_TO_WCHAR
	BCD16	X	The value to be converted has type INT and is accepted as an integer with a value between -999 and +999. The result is available after conversion as a BCD-coded number of the type WORD. A real conversion takes place. If the value is outside the target area, a synchronous error is not	INT_TO_BCD16

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	BCD16_WORD	X	triggered, but rather only the status bit OV is set.	INT_TO_BCD16_WORD
x: Conversion possible - : Conversion not possible 1) Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width including sign, and then the bits are copied. The source type determines the interpretation.				

See also

Implicit conversion of INT (Page 2296)

Overview of data type conversion (Page 2286)

INT (16-bit integers) (Page 2245)

Explicit conversion of UINT

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the UINT data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
UINT	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bits are not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	UINT_TO_BOOL
	BYTE1)	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If bits are lost in the process, the enable output ENO is set to "0".	UINT_TO_BYTE
	WORD1)	X		UINT_TO_WORD
	DWORD1)	X		UINT_TO_DWORD
	LWORD1)	X		UINT_TO_LWORD
	SINT	X		UINT_TO_SINT
	USINT	X		UINT_TO_USINT

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	INT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If the sign bit is changed during the conversion, the enable output ENO is set to "0".	UINT_TO_INT
	DINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	UINT_TO_DINT
	UDINT	X		UINT_TO_UDINT
	LINT	X		UINT_TO_LINT
	ULINT	X		UINT_TO_ULINT
	REAL	X	The value is converted to the destination data type format. (The value "1" is changed to the value "1.0", for example, with the instruction "Convert value" (CONVERT).	UINT_TO_REAL, NORM_X
	LREAL	X		UINT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the destination data type and interpreted as milliseconds.	UINT_TO_TIME
	LTIME	X	The value is transferred to the destination data type and interpreted as nanoseconds.	UINT_TO_LTIME
	S5TIME	-	No explicit conversion	-
	LDT	X	The result is returned in nanoseconds since 1970-1-1-0:0:0.0.	UINT_TO_LDT
	DT	-	No explicit conversion	-
	DTL	-		-
	TOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (interpretation in milliseconds since 0:0; check for 24h limit)	UINT_TO_TOD
	LTOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (interpretation in nanoseconds since 0:0; check for 24h limit)	UINT_TO_LTOD
	DATE	X	The bit pattern of the source value is converted and transferred to the destination data type. (interpretation in days since 1990-1-1; check for negative value)	UINT_TO_DATE, T_CONV
	STRING	X	The value is converted into a character string. If the permitted length of the character string is violated, the enable output ENO is set to "0".	UINT_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		UINT_TO_WSTRING
	CHAR[1)]	X	The bit pattern of the source value is transferred unchanged to the destination data type. The enable	UINT_TO_CHAR

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	WCHAR[1)]	X	output ENO is set to "0" in the event of overflow.	UINT_TO_WCHAR
x: Conversion possible - : Conversion not possible [1)] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width (the non-existing sign is replaced with zeros), and then the bits are copied. The source type determines the interpretation.				

See also

Implicit conversion of UINT (Page 2297)

Overview of data type conversion (Page 2286)

UINT (16-bit integers) (Page 2246)

Explicit conversion of DINT

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the DINT data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
DINT	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bits are not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	DINT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0".	DINT_TO_BYTE
	WORD[1)]	X		DINT_TO_WORD
	DWORD[1)]	X		DINT_TO_DWORD
	LWORD[1)]	X		DINT_TO_LWORD
	SINT	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative	DINT_TO_SINT
	USINT	X		DINT_TO_USINT
	INT	X		DINT_TO_INT
	UINT	X		DINT_TO_UINT

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	UDINT	X	value is converted into an unsigned destination data type, the enable output ENO is set to "0".	DINT_TO_UDINT
	LINT	X		DINT_TO_LINT
	ULINT	X		DINT_TO_ULINT
	REAL	X	The value is converted to the destination data type format. (The value "-1" is changed to the value "-1.0", for example, with the instruction "Convert value" (CONVERT).	DINT_TO_REAL, NORM_X
	LREAL	X		DINT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the destination data type and interpreted as milliseconds.	DINT_TO_TIME, T_CONV
	LTIME	X	The value is transferred to the destination data type and interpreted as nanoseconds.	DINT_TO_LTIME, T_CONV
	S5TIME	-	No explicit conversion	-
	LDT	X	The result is returned in nanoseconds since 1970-1-1-0:0:0.0.	DINT_TO_LDT
	DT	-	No explicit conversion	-
	DTL	-		-
	TOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in milliseconds since 0:0)	DINT_TO_TOD
	LTOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in nanoseconds since 0:0)	DINT_TO_LTOD
	DATE	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in days since 1990-1-1)	DINT_TO_DATE

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	STRING	X	The value is converted into a character string. The character string is shown preceded by a sign. If the permitted length of the character string is violated, the enable output ENO is set to "0".	DINT_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		DINT_TO_WSTRING
	CHAR[1)]	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0".	DINT_TO_CHAR
	WCHAR[1)]	X		DINT_TO_WCHAR
	BCD32	X	The value to be converted has type DINT and is accepted as an integer with a value between –999999 and +9999999. The result is available after conversion as a BCD-coded number of the type DWORD. The enable output is set to "0" in the event of overflow. A real conversion takes place. If the value is outside the target area, a synchronous error is not triggered, but rather only the status bit OV is set.	DINT_TO_BCD32
	BCD32_DWORD	X		DINT_TO_BCD32_DWORD

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width including sign, and then the bits are copied. The source type determines the interpretation.

See also

Implicit conversion of DINT (Page 2298)

Overview of data type conversion (Page 2286)

DINT (32-bit integers) (Page 2247)

Explicit conversion of UDINT

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the UDINT data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
UDINT	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bits are not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	UDINT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If bits are lost in the process, the enable output ENO is set to "0".	UDINT_TO_BYTE
	WORD[1)]	X		UDINT_TO_WORD
	DWORD[1)]	X		UDINT_TO_DWORD
	LWORD[1)]	X		UDINT_TO_LWORD
	SINT	X		UDINT_TO_SINT
	USINT	X		UDINT_TO_USINT
	INT	X		UDINT_TO_INT
	UINT	X		UDINT_TO_UINT
	DINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If the sign bit is changed during the conversion, the enable output ENO is set to "0".	UDINT_TO_DINT
	LINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	UDINT_TO_LINT
	ULINT	X		UDINT_TO_ULINT
	REAL	X	The value is converted to the destination data type format. (The value "1" is changed to the value "1.0", for example, with the instruction "Convert value" (CONVERT).	UDINT_TO_REAL, NORM_X
	LREAL	X		UDINT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the destination data type and interpreted as milliseconds.	UDINT_TO_TIME

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	LTIME	X	The value is transferred to the destination data type and interpreted as nanoseconds.	UDINT_TO_LTIME
	S5TIME	-	No explicit conversion	-
	LDT	X	The result is returned in nanoseconds since 1970-1-1-0:0:0.0.	UDINT_TO_LDT
	DT	-	No explicit conversion	-
	DTL	-		-
	TOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in milliseconds since 0:0; check for 24h limit)	UDINT_TO_TOD, T_CONV
	LTOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in nanoseconds since 0:0; check for 24h limit)	UDNT_TO_LTOD, T_CONV
	DATE	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in days since 1990-1-1; check for negative value)	UDINT_TO_DATE
	STRING	X	The value is converted into a character string. If the permitted length of the character string is violated, the enable output ENO is set to "0".	UDINT_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		UDINT_TO_WSTRING
	CHAR[1)]	X	The bit pattern of the source value is transferred unchanged to the destination data type. The enable output ENO is set to "0" in the event of overflow.	UDINT_TO_CHAR

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	WCHAR[1)]	X		UDINT_TO_WCHAR
x: Conversion possible - : Conversion not possible [1)] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width (the non-existing sign is replaced with zeros), and then the bits are copied. The source type determines the interpretation.				

See also

Implicit conversion of UDINT (Page 2300)

Overview of data type conversion (Page 2286)

UDINT (32-bit integers) (Page 2248)

Explicit conversion of LINT

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the LINT data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
LINT	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bits are not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	LINT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0".	LINT_TO_BYTE
	WORD[1)]	X		LINT_TO_WORD
	DWORD[1)]	X		LINT_TO_DWORD
	LWORD[1)]	X		LINT_TO_LWORD
	SINT	X		LINT_TO_SINT
	USINT	X		LINT_TO_USINT
	INT	X		LINT_TO_INT
	UINT	X		LINT_TO_UINT
	DINT	X		LINT_TO_DINT

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	UDINT	X		LINT_TO_UDINT
	ULINT	X		LINT_TO_ULINT
	REAL	X	The value is converted to the destination data type format. (The value "-1" is changed to the value "-1.0", for example, with the instruction "Convert value" (CONVERT).	LINT_TO_REAL, NORM_X
	LREAL	X		LINT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the destination data type and interpreted as milliseconds.	LINT_TO_TIME, T_CONV
	LTIME	X	The value is transferred to the destination data type and interpreted as nanoseconds.	LINT_TO_LTIME, T_CONV
	S5TIME	-	No explicit conversion	-
	LDT	X	The result is returned in nanoseconds since 1970-1-1-0:0:0.0.	LINT_TO_LDT
	DT	-	No explicit conversion	-
	DTL	-		-
	TOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in milliseconds since 0:0)	LINT_TO_TOD
	LTOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in nanoseconds since 0:0)	LINT_TO_LTOD
	DATE	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0". (interpretation in days since 1990-1-1)	LINT_TO_DATE
	STRING	X	The value is converted into a character string. The character	LINT_TO_STRING, S_CONV, VAL_STRG

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	WSTRING	X	string is shown preceded by a sign. If the permitted length of the character string is violated, the enable output ENO is set to "0".	LINT_TO_WSTRING
	CHAR[1)]	X	The bit pattern of the source value is converted and transferred to the destination data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF)). If a negative value is converted into an unsigned destination data type, the enable output ENO is set to "0".	LINT_TO_CHAR
	WCHAR[1)]	X		LINT_TO_WCHAR
x: Conversion possible - : Conversion not possible [1)] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width including sign, and then the bits are copied. The source type determines the interpretation.				

See also

Implicit conversion of LINT (Page 2301)

Overview of data type conversion (Page 2286)

LINT (64-bit integers) (Page 2249)

Explicit conversion of ULINT

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the ULINT data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
ULINT	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bits are not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	ULINT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	ULINT_TO_BYTE
	WORD[1)]	X		ULINT_TO_WORD
	DWORD[1)]	X		ULINT_TO_DWORD
	LWORD[1)]	X		ULINT_TO_LWORD
	SINT	X		ULINT_TO_SINT
	USINT	X		ULINT_TO_USINT
	INT	X		ULINT_TO_INT
	UINT	X		ULINT_TO_UINT
	DINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If the sign bit is overwritten during the conversion, the enable output ENO is set to "0".	ULINT_TO_DINT
	UDINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	ULINT_TO_UDINT
	LINT	X		ULINT_TO_LINT
	REAL	X	The value is converted to the destination data type format. (The value "1" is changed to the value "1.0", for example, with the instruction "Convert value" (CONVERT).	ULINT_TO_REAL, NORM_X
	LREAL	X		ULINT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the destination data type and interpreted as milliseconds.	ULINT_TO_TIME

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	LTIME	X	The value is transferred to the destination data type and interpreted as nanoseconds.	ULINT_TO_LTIME
	S5TIME	-	No explicit conversion	-
	LDT	X	The result is returned in nanoseconds since 1970-1-1-0:0:0.0.	ULINT_TO_LDT
	DT	-	No explicit conversion	-
	DTL	-		-
	TOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (interpretation in milliseconds since 0:0)	ULINT_TO_TOD, T_CONV
	LTOD	X	The bit pattern of the source value is converted and transferred to the destination data type. (interpretation in nanoseconds since 0:0)	ULINT_TO_LTOD, T_CONV
	DATE	X	The bit pattern of the source value is converted and transferred to the destination data type. (interpretation in days since 1990-1-1)	ULINT_TO_DATE
	STRING	X	The value is converted into a character string. If the permitted length of the character string is violated, the enable output ENO is set to "0".	ULINT_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		ULINT_TO_WSTRING
	CHAR[1)]	X	The bit pattern of the source value is converted and transferred to the destination data type.	ULINT_TO_CHAR
	WCHAR[1)]	X		ULINT_TO_WCHAR

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width (the non-existing sign is replaced with zeros), and then the bits are copied. The source type determines the interpretation.

See also

Implicit conversion of ULINT (Page 2302)

Overview of data type conversion (Page 2286)

ULINT (64-bit integers) (Page 2250)

Floating-point numbers

Explicit conversion of REAL

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the REAL data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
REAL	BOOL	-	No explicit conversion	-
	BYTE	-		-
	WORD	-		-
	DWORD	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	REAL_TO_DWORD
	LWORD	-	No explicit conversion	-
	SINT	X	The value is converted into the destination data type. The result of conversion depends on the instruction employed. The enable output ENO is set to "0" if the admissible range of values of the destination data type is exceeded during conversion or the value to be converted has an invalid floating-point number.	REAL_TO_SINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	USINT	X		REAL_TO_USINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	INT	X		REAL_TO_INT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	UINT	X		REAL_TO_UINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	DINT	X		REAL_TO_DINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	UDINT	X		REAL_TO_UDINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	LINT	X		REAL_TO_LINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	ULINT	X		REAL_TO_ULINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	LREAL	X	The value is converted into the destination data type. The result of the conversion depends on the instruction used, e.g. TRUNC(2.5) = 2.0; CEIL(2.5) = 3.0	REAL_TO_LREAL, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	TIME	-	No explicit conversion	-
	LTIME	-		-
	S5TIME	-		-
	LDT	-		-
	DT	-		-
	DTL	-		-
	TOD	-		-
	LTOD	-		-
	DATE	-		-

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	STRING	X	The value is converted into a character string. The enable output ENO is set to "0" if the character string length is exceeded or the value to be converted has an invalid floating-point number. The min. length of the character string is 14 characters.	REAL_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		REAL_TO_WSTRING
	CHAR	-	No explicit conversion	-
	WCHAR	-		
x: Conversion possible - : Conversion not possible				

See also

Implicit conversion of REAL (Page 2304)

Overview of data type conversion (Page 2286)

REAL (Page 2252)

Explicit conversion of LREAL

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the LREAL data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
LREAL	BOOL	-	No explicit conversion	-
	BYTE	-		-
	WORD	-		-
	DWORD	-		-
	LWORD	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LREAL_TO_LWORD
	SINT	X	The value is converted into the destination data type. The result of conversion depends on the instruction employed. The enable output ENO is set to "0" if the admissible range of values of the destination data type is exceeded during conversion or the value to be converted has an invalid floating-point number.	LREAL_TO_SINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	USINT	X		LREAL_TO_USINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	INT	X		LREAL_TO_INT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	UINT	X		LREAL_TO_UINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	DINT	X		LREAL_TO_DINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	UDINT	X		LREAL_TO_UDINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	LINT	X		LREAL_TO_LINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	ULINT	X		LREAL_TO_ULINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	REAL	X	The value is converted into the destination data type. If the permitted value range of the destination data type is violated during the conversion or if the value to be converted is an invalid floating-point number, the enable output ENO is set to "0". A loss in accuracy is tolerated.	LREAL_TO_LREAL, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	TIME	-	No explicit conversion	-
	LTIME	-		-
	S5TIME	-		-
	LDT	-		-
	DT	-		-
	DTL	-		-
	TOD	-		-
	LTOD	-		-
	DATE	-		-
	STRING	X	The value is converted into a character string. The enable output ENO is set to "0" if the character string length is exceeded or the value to be converted has an invalid floating-point number. The min. length of the character string is 21 characters.	LREAL_TO_STRING, S_CONV, VAL_STRG
	WSTRING	X		LREAL_TO_WSTRING
	CHAR	-	No explicit conversion	-
	WCHAR	-		-
x: Conversion possible - : Conversion not possible				

See also

Implicit conversion of LREAL (Page 2305)

Overview of data type conversion (Page 2286)

LREAL (Page 2253)

Timers

Explicit conversion of S5TIME

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the S5TIME data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
S5TIME	BOOL	-	No explicit conversion	-
	BYTE	-		-
	WORD[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	S5TIME_TO_WORD
	DWORD	-	No explicit conversion	-
	LWORD	-		-
	SINT	-		-
	USINT	-		-
	INT	-		-
	UINT	-		-
	DINT	-		-
	UDINT	-		-
	LINT	-		-
	ULINT	-		-
	REAL	-		-
	LREAL	-		-
	TIME	X	A conversion of the source value into the destination data type takes place here (for example, s5t#10ms becomes T#10ms)	S5TIME_TO_TIME
	LTIME	X	A conversion of the source value into the destination data type takes place here (for example, s5t#10ms becomes LTIME#10ms)	S5TIME_TO_LTIME
	LDT	-	No explicit conversion	-
	DT	-		-
	DTL	-		-
	TOD	-		-
	LTOD	-		-
	DATE	-		-
	STRING	-		-
	WSTRING	-		-
	CHAR	-		-

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	WCHAR	-		-
x: Conversion possible - : Conversion not possible 1) Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width, and then the bits are copied. The source type determines the interpretation.				

See also

Implicit conversion of S5TIME (Page 2306)

Overview of data type conversion (Page 2286)

S5TIME (duration) (Page 2255)

Explicit conversion of TIME

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the TIME data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
TIME	BOOL	-	No explicit conversion	-
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	TIME_TO_BYTE
	WORD[1)]	X		TIME_TO_WORD
	DWORD[1)]	X		TIME_TO_DWORD
	LWORD[1)]	X		TIME_TO_LWORD
	SINT	X	The bit pattern of the source value is transferred unchanged right-justified and interpreted as milliseconds to the destination data type.	TIME_TO_SINT
	USINT	X		TIME_TO_USINT
	INT	X		TIME_TO_INT
	UINT	X		TIME_TO_UINT
	DINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. The result of conversion shows the duration in milliseconds.	TIME_TO_DINT, T_CONV
	UDINT	X	The bit pattern of the source value is transferred unchanged right-justified and interpreted as milliseconds to the destination data type. A change in sign has the result that the enable output ENO is set to "0".	TIME_TO_UDINT
	LINT	X		TIME_TO_LINT
	ULINT	X		TIME_TO_ULINT
	REAL	-	No explicit conversion	-
	LREAL	-		-

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	S5TIME	X	A conversion of the source value into the destination data type takes place here (for example, T#10ms becomes s5t#10ms)	TIME_TO_TIME
	LTIME	X	A conversion of the source value into the destination data type takes place here (for example, T#10ms becomes LTIME#10ms)	TIME_TO_LTIME
	LDT	-	No explicit conversion	-
	DT	-		-
	DTL	-		-
	TOD	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If the source value is outside the value range of TOD, the destination data type remains unchanged.	TIME_TO_TOD
	LTOD	-	No explicit conversion	-
	DATE	-		-
	STRING	-		-
	WSTRING	-		-
	CHAR	-		-
	WCHAR	-		-

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width, and then the bits are copied. The source type determines the interpretation.

See also

Implicit conversion of TIME (Page 2307)

Overview of data type conversion (Page 2286)

TIME (IEC time) (Page 2257)

Explicit conversion of LTIME

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the LTIME data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
LTIME	BOOL	-	No explicit conversion	-
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LTIME_TO_BYTE
	WORD[1)]	X		LTIME_TO_WORD
	DWORD[1)]	X		LTIME_TO_DWORD
	LWORD[1)]	X		LTIME_TO_LWORD
	SINT	X	The bit pattern of the source value is transferred unchanged right-justified and interpreted as nanoseconds to the destination data type. A change in sign has the result that the enable output ENO is set to "0".	LTIME_TO_SINT
	USINT	X		LTIME_TO_USINT
	INT	X		LTIME_TO_INT
	UINT	X		LTIME_TO_UINT
	DINT	X		LTIME_TO_DINT
	UDINT	X		LTIME_TO_UDINT
	LINT	X		LTIME_TO_LINT, T_CONV
	ULINT	X		LTIME_TO_ULINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	S5TIME	X	A conversion of the source value into the destination data type takes place here (for example, LTIME#10ms becomes s5t#10ms)	LTIME_TO_S5TIME
	TIME	X	A conversion of the source value into the destination data type takes place here (for example, LTIMET#10ms becomes T#10ms)	LTIME_TO_TIME
	LDT	X	A conversion of the source value into the destination data type takes place here (for example, LTIMET#10ms becomes LDT#1970-1-1-0:0:0.010000000)	LTIME_TO_LDT
	DT	-	No explicit conversion	-
	DTL	-		-
	TOD	-		-
	LTOD	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If the source value is outside the value range of LTOD, the destination data type remains unchanged.	LTIME_TO_LTOD
	DATE	-	No explicit conversion	-
	STRING	-		-

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	WSTRING	-		-
	CHAR	-		-
	WCHAR	-		-
x: Conversion possible - : Conversion not possible 1) Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width, and then the bits are copied. The source type determines the interpretation.				

See also

Implicit conversion of LTIME (Page 2308)

Overview of data type conversion (Page 2286)

LTIME (IEC time) (Page 2257)

Date and time-of-day

Explicit conversion of DT

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the DT data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
DT	BOOL	-	No explicit conversion	-
	BYTE	-		-
	WORD	-		-
	DWORD	-		-
	LWORD	-		-
	SINT	-		-
	USINT	-		-
	INT	-		-
	UINT	-		-
	DINT	-		-
	UDINT	-		-
	LINT	-		-
	ULINT	-		-
	REAL	-		-
	LREAL	-		-
	S5TIME	-		-
	TIME	-		-
	LTIME	-		-

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	LDT	X	A conversion of the source value into the destination data type takes place here (for example, DT#1990-1-2-0:0:1.0 becomes LDT#1990-1-2-0:0:1.0)	DT_TO_LDT
	DTL	X	A conversion of the source value into the destination data type takes place here (for example, DT#1990-1-2-0:0:1.0 becomes DTL#1990-1-2-0:0:1.0)	DT_TO_DTL
	TOD	X	A conversion of the source value into the destination data type takes place here (for example, DT#1990-1-2-0:0:1.0 becomes TOD#1990-1-2-0:0:1.0)	DT_TO_TOD
	LTOD	X	A conversion of the source value into the destination data type takes place here (for example, DT#1990-1-2-0:0:1.0 becomes LTOD#1990-1-2-0:0:1.0)	DT_TO_LTOD
	DATE	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	DT_TO_DATE
	STRING	-	No explicit conversion	-
	WSTRING	-		-
	CHAR	-		-
	WCHAR	-		-
x: Conversion possible - : Conversion not possible				

See also

Implicit conversion of DT (Page 2310)

Overview of data type conversion (Page 2286)

DATE_AND_TIME (date and time of day) (Page 2260)

Explicit conversion of LDT

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the LDT data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
LDT	BOOL	-	No explicit conversion	-
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LDT_TO_BYTE
	WORD[1)]	X		LDT_TO_WORD
	DWORD[1)]	X		LDT_TO_DWORD
	LWORD[1)]	X		LDT_TO_LWORD
	SINT	X		LDT_TO_SINT
	USINT	X		LDT_TO_USINT
	INT	X		LDT_TO_INT
	UINT	X		LDT_TO_UINT
	DINT	X		LDT_TO_DINT
	UDINT	X		LDT_TO_UDINT
	LINT	X		LDT_TO_LINT
	ULINT	X		LDT_TO_ULINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	S5TIME	-		-
	TIME	-		-
	LTIME	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LDT_TO_LTIME
	DT	X	A conversion of the source value into the destination data type takes place here (for example, LDT#1990-1-2-0:0:1.0 becomes DT#1990-1-2-0:0:1.0)	LDT_TO_DT
	DTL	X	A conversion of the source value into the destination data type takes place here (for example, LDT#1990-1-2-0:0:1.0 becomes DTL#1990-1-2-0:0:1.0)	LDT_TO_DTL
	TOD	X	A conversion of the source value into the destination data type takes place here (for example, LDT#1990-1-2-0:0:1.0 becomes TOD#1990-1-2-0:0:1.0)	LDT_TO_TOD
	LTOD	X	A conversion of the source value into the destination data type takes place here (for example, LDT#1990-1-2-0:0:1.0 becomes LTOD#1990-1-2-0:0:1.0)	LDT_TO_LTOD

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	DATE	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LDT_TO_DATE
	STRING	-	No explicit conversion	-
	WSTRING	-		-
	CHAR	-		-
	WCHAR	-		-
x: Conversion possible - : Conversion not possible 1) Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width, and then the bits are copied. The source type determines the interpretation.				

See also

Implicit conversion of LDT (Page 2311)

Overview of data type conversion (Page 2286)

LDT (DATE_AND_LTIME) (Page 2261)

Explicit conversion of DTL

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the DTL data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
DTL	BOOL	-	No explicit conversion	-
	BYTE	-		-
	WORD	-		-
	DWORD	-		-
	LWORD	-		-
	SINT	-		-
	USINT	-		-
	INT	-		-
	UINT	-		-
	DINT	-		-
	UDINT	-		-
	LINT	-		-
	ULINT	-		-
	REAL	-		-
	LREAL	-		-
	S5TIME	-		-
	TIME	-		-

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	LTIME	-		-
	LDT	X	A conversion of the source value into the destination data type takes place here (for example, DTL#1990-1-2-0:0:1.0 becomes LDT#1990-1-2-0:0:1.0)	DTL_TO_LDT
	DT	X	A conversion of the source value into the destination data type takes place here (for example, DTL#1990-1-2-0:0:1.0 becomes DT#1990-1-2-0:0:1.0)	DTL_TO_DT
	TOD	X	During the conversion, the time of day is extracted from the DTL format and written to the destination data type.	DTL_TO_TOD, T_CONV
	LTOD	X		DTL_TO_LTOD
	DATE	X	During the conversion, the date is extracted from the DTL format and written to the destination data type. The enable output ENO is set to "0" in the event of overflow.	DTL_TO_DATE, T_CONV
	STRING	-	No explicit conversion	-
	WSTRING	-		-
	CHAR	-		-
	WCHAR	-		-
x: Conversion possible - : Conversion not possible				

See also

Implicit conversion of DTL (Page 2312)

Overview of data type conversion (Page 2286)

DTL (Page 2262)

Explicit conversion of DATE

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the DATE data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
DATE	BOOL	-	No explicit conversion	-
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	DATE_TO_BYTE
	WORD[1)]	X		DATE_TO_WORD
	DWORD[1)]	X		DATE_TO_DWORD

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	LWORD[1)]	X		DATE_TO_LWORD
	SINT	X	The number of days since 1/1/1990 is returned as result.	DATE_TO_SINT
	USINT	X		DATE_TO_USINT
	INT	X		DATE_TO_INT
	UINT	X		DATE_TO_UINT
	DINT	X		DATE_TO_DINT
	UDINT	X		DATE_TO_UDINT
	LINT	X		DATE_TO_LINT
	ULINT	X		DATE_TO_ULINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	S5TIME	-		-
	TIME	-		-
	LTIME	-		-
	DT	X	The conversion sets the date in the destination type.	DATE_TO_DT
	LDT	X		DATE_TO_LDT
	DTL	X		DATE_TO_DTL
	TOD	-	No explicit conversion	-
	LTOD	-		-
	STRING	-		-
	WSTRING	-		-
	CHAR	-		-
	WCHAR	-		-

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width, and then the bits are copied. The source type determines the interpretation.

See also

Implicit conversion of DATE (Page 2313)

Overview of data type conversion (Page 2286)

DATE (Page 2258)

Explicit conversion of TOD

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the TOD data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
TOD	BOOL	-	No explicit conversion	-
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	TOD_TO_BYTE
	WORD[1)]	X		TOD_TO_WORD
	DWORD[1)]	X		TOD_TO_DWORD
	LWORD[1)]	X		TOD_TO_LWORD
	SINT	X	The number of milliseconds since midnight is returned as result.	TOD_TO_SINT
	USINT	X		TOD_TO_USINT
	INT	X		TOD_TO_INT
	UINT	X		TOD_TO_UINT
	DINT	X		TOD_TO_DINT
	UDINT	X	The result of the conversion corresponds to the number of milliseconds since the start of day (0:00).	TOD_TO_UDINT, T_CONV
	LINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	TOD_TO_LINT
	ULINT	X		TOD_TO_ULINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	S5TIME	-		-
	TIME	X	The duration since midnight is returned as result.	TOD_TO_TIME
	LTIME	-	No explicit conversion	-
	DT	X	The conversion sets the time in the destination data type.	TOD_TO_DT
	LDT	X		TOD_TO_LDT
	DTL	X		TOD_TO_DTL
	DATE	-	No explicit conversion	-
	LTOD	X	A conversion of the source value into the destination data type takes place here (for example, TOD#1:0:0.0 becomes LTOD#1:0:0.0)	TOD_TO_LTOD
	STRING	-	No explicit conversion	-
	WSTRING	-		-
	CHAR	-		-

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	WCHAR	-		-
x: Conversion possible - : Conversion not possible 1) Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width, and then the bits are copied. The source type determines the interpretation.				

See also

Implicit conversion of TOD (Page 2314)

Overview of data type conversion (Page 2286)

TIME_OF_DAY (TOD) (Page 2259)

Explicit conversion of LTOD

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the LTOD data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
LTOD	BOOL	-	No explicit conversion	-
	BYTE1)	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LTOD_TO_BYTE
	WORD1)	X		LTOD_TO_WORD
	DWORD1)	X		LTOD_TO_DWORD
	LWORD1)	X		LTOD_TO_LWORD
	SINT	X	The number of nanoseconds since midnight is returned as result.	LTOD_TO_SINT
	USINT	X		LTOD_TO_USINT
	INT	X		LTOD_TO_INT
	UINT	X		LTOD_TO_UINT
	DINT	X		LTOD_TO_DINT
	UDINT	X		LTOD_TO_UDINT
	LINT	X		LTOD_TO_LINT
	ULINT	X		LTOD_TO_ULINT, T_CONV
	REAL	-	No explicit conversion	-
	LREAL	-		-
	S5TIME	-		-
	TIME	-		-
	LTIME	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	LTOD_TO_LTIME

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	DT	X	A conversion of the source value into the destination data type takes place here (for example, LTOD#1:0:0.0 becomes DT#1:0:0.0)	LTOD_TO_DT
	LDT	X	A conversion of the source value into the destination data type takes place here (for example, LTOD#1:0:0.0 becomes LDT#1:0:0.0)	LTOD_TO_LDT
	DTL	X	A conversion of the source value into the destination data type takes place here (for example, LTOD#1:0:0.0 becomes DTL#1:0:0.0)	LTOD_TO_DTL
	DATE	-	No explicit conversion	-
	TOD	X	A conversion of the source value into the destination data type takes place here (for example, LTOD#1:0:0.0 becomes TOD#1:0:0.0)	LTOD_TO_TOD
	STRING	-	No explicit conversion	-
	WSTRING	-		-
	CHAR	-		-
	WCHAR	-		-

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD, LWORD) and the data types CHAR and WCHAR are initially extended to the necessary width, and then the bits are copied. The source type determines the interpretation.

See also

Implicit conversion of LTOD (Page 2316)

Overview of data type conversion (Page 2286)

LTOD (LTIME_OF_DAY) (Page 2259)

String

Explicit conversion of CHAR

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the CHAR data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
CHAR	BOOL	-	No explicit conversion	-
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	CHAR_TO_BYTE
	WORD[1)]	X		CHAR_TO_WORD
	DWORD[1)]	X		CHAR_TO_DWORD
	LWORD[1)]	X		CHAR_TO_LWORD
	SINT	X		CHAR_TO_SINT
	USINT	X		CHAR_TO_USINT
	INT	X		CHAR_TO_INT
	UINT	X		CHAR_TO_UINT
	DINT	X		CHAR_TO_DINT
	UDINT	X		CHAR_TO_UDINT
	LINT	X		CHAR_TO_LINT
	ULINT	X		CHAR_TO_ULINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	S5TIME	-		-
	TIME	-		-
	LTIME	-		-
	DT	-		-
	LDT	-		-
	DTL	-		-
	TOD	-		-
	LTOD	-		-
	DATE	-		-
	STRING	X	The value is converted into the first character in the character string (STRING). The length "1" is set after conversion if the character string length is not defined. If the character string length is defined, it remains unchanged following conversion.	CHAR_TO_STRING, S_CONV
	WSTRING	-	No explicit conversion	-

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	WCHAR	X		CHAR_TO_WCHAR
x: Conversion possible - : Conversion not possible 1) Bit strings (BYTE, WORD, DWORD, LWORD) and the data type WCHAR are initially extended to the necessary width, and then the bits are copied. The source type determines the interpretation.				

See also

Implicit conversion of CHAR (Page 2317)

Overview of data type conversion (Page 2286)

CHAR (character) (Page 2263)

Explicit conversion of WCHAR

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the WCHAR data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
WCHAR	BOOL	-	No explicit conversion	-
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	WCHAR_TO_BYTE
	WORD[1)]	X		WCHAR_TO_WORD
	DWORD[1)]	X		WCHAR_TO_DWORD
	LWORD[1)]	X		WCHAR_TO_LWORD
	SINT	X		WCHAR_TO_SINT
	USINT	X		WCHAR_TO_USINT
	INT	X		WCHAR_TO_INT
	UINT	X		WCHAR_TO_UINT
	DINT	X		WCHAR_TO_DINT
	UDINT	X		WCHAR_TO_UDINT
	LINT	X		WCHAR_TO_LINT
	ULINT	X		WCHAR_TO_ULINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	S5TIME	-		-
	TIME	-		-
	LTIME	-		-
	DT	-		-
	LDT	-		-
	DTL	-		-
	TOD	-		-

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	LTOD	-		-
	DATE	-		-
	STRING	-		-
	WSTRING	X	The value is converted into the first character in the character string (WSTRING). The length "1" is set after conversion if the character string length is not defined. If the character string length is defined, it remains unchanged following conversion.	WCHAR_TO_WSTRING
	CHAR	X		WCHAR_TO_CHAR
x: Conversion possible - : Conversion not possible 1) Bit strings (BYTE, WORD, DWORD, LWORD) and the data type CHAR are initially extended to the necessary width, and then the bits are copied. The source type determines the interpretation.				

See also

Overview of data type conversion (Page 2286)

WCHAR (character) (Page 2263)

Implicit conversion of WCHAR (Page 2318)

Explicit conversion of STRING

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the STRING data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
STRING	BOOL	-	No explicit conversion	-
	BYTE	-		-
	WORD	-		-
	DWORD	-		-
	LWORD	-		-
	SINT	X	Conversion begins with the first character in the character string (STRING) and stops at the end of the string or at the first inadmissible character. The following characters are admissible for conversion: • Digit • Sign • Dot	STRING_TO_SINT, S_CONV, STRG_VAL
	USINT	X		STRING_TO_USINT, S_CONV, STRG_VAL
	INT	X		STRING_TO_INT, S_CONV, STRG_VAL
	UINT	X		STRING_TO_UINT, S_CONV, STRG_VAL

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	DINT	X	The first character in the character string may be a sign (+, -) or a digit. Leading spaces are ignored. The dot is used as separation for the conversion of floating-point numbers. The exponential notation "e" or "E" is not permitted. The comma as thousand separator to the left of the decimal point is permitted but is ignored. If the layout of the character string is invalid for the conversion or an overflow occurs, then the enable output ENO will be set to "0".	STRING_TO_DINT, S_CONV, STRG_VAL
	UDINT	X		STRING_TO_UDINT, S_CONV, STRG_VAL
	LINT	X		STRING_TO_LINT, S_CONV, STRG_VAL
	ULINT	X		STRING_TO_ULINT, S_CONV, STRG_VAL
	REAL	X		STRING_TO_REAL, S_CONV, STRG_VAL
	LREAL	X		STRING_TO_LREAL, S_CONV, STRG_VAL
	S5TIME	-	No explicit conversion	-
	TIME	-		-
	LTIME	-		-
	DT	-		-
	LDT	-		-
	DTL	-		-
	TOD	-		-
	LTOD	-		-
	DATE	-		-
	CHAR	X	The first character in the character string (STRING) is transferred to the destination data type. The value "0" is written to the destination data type if the character string is empty.	STRING_TO_CHAR, S_CONV
	WCHAR	-	No explicit conversion	-
	WSTRING	X		STRING_TO_WSTRING
x: Conversion possible - : Conversion not possible				

See also

Implicit conversion of STRING (Page 2319)

Overview of data type conversion (Page 2286)

STRING (Page 2264)

Explicit conversion of WSTRING

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the WSTRING data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
WSTRING	BOOL	-	No explicit conversion	-
	BYTE	-		-
	WORD	-		-
	DWORD	-		-
	LWORD	-		-
	SINT	X	Conversion begins with the first character in the character string (WSTRING) and stops at the end of the string or at the first inadmissible character. The following characters are admissible for conversion: • Digit • Sign • Dot The first character in the character string may be a sign (+, -) or a digit. Leading spaces are ignored. The dot is used as separation for the conversion of floating-point numbers. The exponential notation "e" or "E" is not permitted. The comma as thousand separator to the left of the decimal point is permitted but is ignored. If the layout of the character string is invalid for the conversion or an overflow occurs, then the enable output ENO will be set to "0".	WSTRING_TO_SINT, S_CONV, STRG_VAL
	USINT	X		WSTRING_TO_USINT, S_CONV, STRG_VAL
	INT	X		WSTRING_TO_INT, S_CONV, STRG_VAL
	UINT	X		WSTRING_TO_UINT, S_CONV, STRG_VAL
	DINT	X		WSTRING_TO_DINT, S_CONV, STRG_VAL
	UDINT	X		WSTRING_TO_UDINT, S_CONV, STRG_VAL
	LINT	X		WSTRING_TO_LINT, S_CONV, STRG_VAL
	ULINT	X		WSTRING_TO_ULINT, S_CONV, STRG_VAL
	REAL	X		WSTRING_TO_REAL, S_CONV, STRG_VAL
	LREAL	X		WSTRING_TO_LREAL, S_CONV, STRG_VAL
	S5TIME	-	No explicit conversion	-
	TIME	-		-
	LTIME	-		-
	DT	-		-
	LDT	-		-
	DTL	-		-
	TOD	-		-
	LTOD	-		-
	DATE	-		-
	CHAR	-		-

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	WCHAR	X	The first character in the character string (WSTRING) is transferred to the destination data type. The value "0" is written to the destination data type if the character string is empty.	WSTRING_TO_WCHAR
	STRING	X		WSTRING_TO_STRING
x: Conversion possible - : Conversion not possible				

See also

Overview of data type conversion (Page 2286)

WSTRING (Page 2265)

Implicit conversion of WSTRING (Page 2320)

Data type conversion

Overview of data type conversion

Introduction

If you link several operands in an instruction, you must make sure that the data types are compatible. This applies also for assignments or for supplying block parameters. If the operands are not the same data type, a conversion has to be carried out.

There are two options for the conversion:

- Implicit conversion
  The conversion take place automatically when the instruction is executed.
- Explicit conversion
  You use an explicit conversion instruction before the actual instruction is executed.

Note

The data type conversion options described always refer to the latest version of the CPU (V. 4) Conversions marked as possible may not be available in CPU versions 1 - 3.

Note

Converting bit strings in SCL

All bit strings (BYTE, WORD, DWORD and LWORD) are handled like the corresponding unsigned integers (USINT, UINT, UDINT and ULINT) in expressions. Therefore, implicit conversion from DWORD to REAL is carried out like a conversion from UDINT to REAL, for example.

Implicit conversion

An implicit conversion is executed automatically if the data types of the operands are compatible. This compatibility test can be carried out according to criteria that are more or less strict:

- With IEC check (default)
  If IEC check is set, the following rules are applied:
  - Implicit conversion of BOOL to other data types is not possible.
  - Only the REAL, BYTE, WORD, DINT, INT, SINT, UDINT, UINT, USINT, TIME, DT, STRING and CHAR data types can be converted implicitly.
  - The bit length of the source data type must not exceed the bit length of the target data type. An operand of data type WORD, for example, cannot be declared at a parameter at which data type BYTE is expected.
- Without IEC check
  If IEC check is not set, the following rules are applied:
  - Implicit conversion of BOOL to other data types is not possible.
  - Only the REAL, LREAL, BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT, TIME, DTL, TOD, DATE, STRING and CHAR data types can be converted implicitly.
  - The bit length of the source data type must not exceed the bit length of the target data type. An operand of data type DWORD, for example, cannot be declared at a parameter at which data type WORD is expected.
  - The bit length of an operand entered in-out parameters InOut) must be the same as the programmed bit length for the parameter in question.

Note

Implicit conversion without IEC check

The programming editor uses a gray rectangle to mark operands that are implicitly converted. The dark gray rectangle signals that an implicit conversion is possible without any accuracy loss, for example, if you convert the data type SINT to INT. A light gray rectangle signals that implicit conversion is possible, but errors could occur during runtime. If, for example, you are converting the data type DINT to INT and an overflow occurs, the enable output ENO is set to "0".

For more information about the setting of the IEC check and the implicit conversion, refer to "See also".

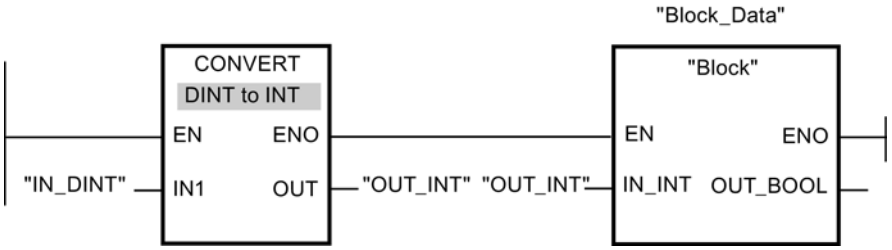
Explicit conversion

If the operands are not compatible and an implicit conversion is therefore not possible, you can use an explicit conversion instruction. You can find the conversion instructions in the "Instructions" task card.

A possible overflow is displayed at the ENO enable output. An overflow is created, for example, if the value of the source data type is greater than the value of the target data type.

For more information about explicit conversion, refer to "See also".

The following figure shows an example in which an explicit data type conversion must be carried out:



The "Block" function block expects a tag of the INT data type at the "IN_INT" input parameter. Therefore, the value of the "IN_DINT" tag must first be converted from DINT to INT. If the value of the "IN_DINT" tag is within the permitted value range of the INT data type, the conversion takes place. Otherwise, an overflow is signaled. A conversion still takes place even in case of an overflow, but the values are cut off and the enable output ENO is set to "0".

See also

Implicit conversion (Page 2374)

Explicit conversion (Page 2391)

Setting and canceling the IEC check (Page 2374)

Implicit conversion

Setting and canceling the IEC check

The data types of the operands used are check for compatibility. This compatibility test can be carried out according to criteria that are more or less strict. If "IEC check" is activated, stricter criteria are applied.

You can set the IEC check centrally for all new blocks of the project or for individual blocks.

Setting IEC check for new blocks

To set the IEC check for all new blocks in the project, proceed as follows:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "PLC programming > General" group in the area navigation.
3. Select or clear the "IEC check" check box in the "Default settings for new blocks" group.
   The IEC check is enabled or disabled for all new blocks in the program.

Setting IEC check for a block

To set the IEC check for a block, proceed as follows:

1. Open the block.
2. Open the "Properties" tab in the Inspector window.
3. Select the "Attributes" group in the area navigation.
4. Select or clear the "IEC check" check box.
   The IEC check is enabled or disabled for this block. The setting is stored together with the project.

Binary numbers

Implicit conversion of BOOL

Options for implicit conversion

The implicit conversion of the BOOL data type is not possible.

See also

BOOL (bit) (Page 2240)

Bit strings

Implicit conversion of BYTE

Options for implicit conversion

The following table shows the options for implicit conversion of the BYTE data type:

Source	Target	With IEC check	Without IEC check	Description
BYTE	BOOL	-	-	No implicit conversion
	WORD	x	x	The bit pattern of the source value is transferred unchanged right-justified to the target data type.
	DWORD	x	x	
	SINT	-	x	
	USINT	-	x	
	INT	-	x	
	UINT	-	x	
	DINT	-	x	
	UDINT	-	x	
	REAL	-	-	No implicit conversion

Source	Target	With IEC check	Without IEC check	Description
	LREAL	-	-	
	TIME	-	-	
	DTL	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	
	CHAR	-	x	The bit pattern of the source value is transferred unchanged to the target data type.
x: Conversion possible -: Conversion not possible				

See also

BYTE (byte) (Page 2240)
Setting and canceling the IEC check (Page 2374)
Overview of data type conversion (Page 2372)
Explicit conversion of BYTE (Page 2392)

Implicit conversion of WORD

Options for implicit conversion

The following table shows the options for implicit conversion of the WORD data type:

Source	Target	With IEC check	Without IEC check	Description
WORD	BOOL	-	-	No implicit conversion
	BYTE	-	X	The least significant byte is transferred to the target data type, while the most significant byte is ignored.
	DWORD	X	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.
	SINT	-	X	The least significant byte is transferred to the target data type, while the most significant byte is ignored.
	USINT	-	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	TIME	-	-	

Source	Target	With IEC check	Without IEC check	Description
	DTL	-	-	
	TOD	-	-	
	DATE	-	X	The bit pattern of the source value is transferred unchanged to the target data type.
	STRING	-	-	No implicit conversion
	CHAR	-	X	The bit pattern of the source value is transferred unchanged to the target data type.
x: Conversion possible -: Conversion not possible				

See also

WORD (Page 2241)
Setting and canceling the IEC check (Page 2374)
Overview of data type conversion (Page 2372)
Explicit conversion of WORD (Page 2393)

Implicit conversion of DWORD

Options for implicit conversion

The following table shows the options for implicit conversion of the DWORD data type:

Source	Target	With IEC check	Without IEC check	Description
DWORD	BOOL	-	-	No implicit conversion
	BYTE	-	X	The right bytes are transferred to the target data type; the left bytes are ignored.
	WORD	-	X	
	SINT	-	X	
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.
	UDINT	-	X	
	REAL	-	X	The value is converted to the format of the target data type. (The value "-1", for example, is converted to the value "-1.0".)
	LREAL	-	-	No implicit conversion
	TIME	-	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.
	DTL	-	-	No implicit conversion
	TOD	-	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.

Source	Target	With IEC check	Without IEC check	Description
	DATE	-	-	No implicit conversion
	STRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.
x: Conversion possible -: Conversion not possible				

See also

DWORD (Page 2242)

Setting and canceling the IEC check (Page 2374)

Overview of data type conversion (Page 2372)

Explicit conversion of DWORD (Page 2395)

Integers

Implicit conversion of SINT

Options for implicit conversion

The following table shows the options for implicit conversion of the SINT data type:

Source	Target	With IEC check	Without IEC check	Description
SINT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type. The remaining bits are filled with "0".
	WORD	-	X	
	DWORD	-	X	
	USINT	-	X	The bit pattern of the source value is converted and transferred to the target data type. (for example, value transfer from SINT #-1 -> INT #-1, not filled with "0".)
	INT	X	X	
	UINT	-	X	
	DINT	X	X	
	UDINT	-	X	
	REAL	X	X	The value is converted to the format of the target data type. (The value "-1", for example, is converted to the value "-1.0".)
	LREAL	X	X	
	TIME	-	-	No implicit conversion
	DTL	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	

Source	Target	With IEC check	Without IEC check	Description
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.
x: Conversion possible -: Conversion not possible				

See also

SINT (8-bit integers) (Page 2244)
Setting and canceling the IEC check (Page 2374)
Overview of data type conversion (Page 2372)
Explicit conversion of SINT (Page 2398)

Implicit conversion of USINT

Options for implicit conversion

The following table shows the options for implicit conversion of the USINT data type:

Source	Target	With IEC check	Without IEC check	Description
USINT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type. The remaining bits are filled with "0".
	WORD	-	X	
	DWORD	-	X	
	SINT	-	X	The bit pattern of the source value is converted and transferred to the target data type. (for example, value conversion from USINT #10 -> DINT #10 or USINT #128 -> SINT #-128)
	INT	X	X	
	UINT	X	X	
	DINT	X	X	
	UDINT	X	X	
	REAL	X	X	The value is converted to the format of the target data type. (The value "1", for example, is converted to the value "1.0".)
	LREAL	X	X	
	TIME	-	-	No implicit conversion
	DTL	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.
x: Conversion possible -: Conversion not possible				

See also

USINT (8-bit integers) (Page 2245)
Setting and canceling the IEC check (Page 2374)
Overview of data type conversion (Page 2372)
Explicit conversion of USINT (Page 2400)

Implicit conversion of INT

Options for implicit conversion

The following table shows the options for implicit conversion of the INT data type:

Source	Target	With IEC check	Without IEC check	Description
INT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.
	WORD	-	X	
	DWORD	-	X	
	SINT	-	X	The bit pattern of the source value is converted and transferred to the target data type. (for example, value conversion from INT #-1 -> SINT #-1, or INT #-32 767 -> UINT #32 769)
	USINT	-	X	
	UINT	-	X	
	DINT	X	X	
	UDINT	-	X	
	REAL	X	X	The value is converted to the format of the target data type. (The value "-1", for example, is converted to the value "-1.0".)
	LREAL	X	X	
	TIME	-	-	No implicit conversion
	DTL	-	-	
	TOD	-	-	
	DATE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.
	STRING	-	-	No implicit conversion
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.
x: Conversion possible -: Conversion not possible				

See also

INT (16-bit integers) (Page 2245)
Setting and canceling the IEC check (Page 2374)
Overview of data type conversion (Page 2372)
Explicit conversion of INT (Page 2401)

Implicit conversion of UINT

Options for implicit conversion

The following table shows the options for implicit conversion of the UINT data type:

Source	Target	With IEC check	Without IEC check	Description
UINT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.
	WORD	-	X	
	DWORD	-	X	
	SINT	-	X	The bit pattern of the source value is converted and transferred to the target data type. (for example, value conversion from UINT #100 -> DINT #100 or UINT #60 000 -> INT #-5536)
	USINT	-	X	
	INT	-	X	
	DINT	X	X	
	UDINT	X	X	
	REAL	X	X	The value is converted to the format of the target data type. (The value "1", for example, is converted to the value "1.0".)
	LREAL	X	X	
	TIME	-	-	No implicit conversion
	DTL	-	-	
	TOD	-	-	
	DATE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.
	STRING	-	-	No implicit conversion
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.
x: Conversion possible -: Conversion not possible				

See also

UINT (16-bit integers) (Page 2246)

Setting and canceling the IEC check (Page 2374)

Overview of data type conversion (Page 2372)

Explicit conversion of UINT (Page 2403)

Implicit conversion of DINT

Options for implicit conversion

The following table shows the options for implicit conversion of the DINT data type:

Source	Target	With IEC check	Without IEC check	Description
DINT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.
	WORD	-	X	
	DWORD	-	X	
	SINT	-	X	The bit pattern of the source value is converted and transferred to the target data type. (for example, value conversion from DINT #-1 -> SINT #-1 or DINT #-1 -> USINT #255)
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	UDINT	-	X	
	REAL	-	X	The bit pattern of the source value is converted and transferred to the target data type. (for example, value conversion from DINT #-1 -> REAL #-1.0, but there is a loss in accuracy for numbers with an absolute value greater than 8 388 608)
	LREAL	X	X	The value is converted to the format of the target data type. (The value "-1", for example, is converted to the value "-1.0".)
	TIME	-	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.
	DTL	-	-	No implicit conversion
	TOD	-	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.
	DATE	-	-	No implicit conversion
	STRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.
x: Conversion possible -: Conversion not possible				

See also

DINT (32-bit integers) (Page 2247)

Setting and canceling the IEC check (Page 2374)

Overview of data type conversion (Page 2372)

Explicit conversion of DINT (Page 2405)

Implicit conversion of UDINT

Options for implicit conversion

The following table shows the options for implicit conversion of the UDINT data type:

Source	Target	With IEC check	Without IEC check	Description
UDINT	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.
	WORD	-	X	
	DWORD	-	X	
	SINT	-	X	The bit pattern of the source value is converted and transferred to the target data type. (for example, value conversion from DINT #-1 -> SINT #-1 or DINT #-1 -> USINT #255)
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	REAL	-	X	The bit pattern of the source value is converted and transferred to the target data type. (for example, value conversion from DINT #-1 -> REAL #-1.0, but there is a loss in accuracy for numbers with an absolute value greater than 8 388 608)
	LREAL	X	X	The value is converted to the format of the target data type. (The value "1", for example, is converted to the value "1.0".)
	TIME	-	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.
	DTL	-	-	No implicit conversion
	TOD	-	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.
	DATE	-	-	No implicit conversion
	STRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.
x: Conversion possible -: Conversion not possible				

See also

UDINT (32-bit integers) (Page 2248)

Setting and canceling the IEC check (Page 2374)

Overview of data type conversion (Page 2372)

Explicit conversion of UDINT (Page 2407)

Floating-point numbers

Implicit conversion of REAL

Options for implicit conversion

The following table shows the options for implicit conversion of the REAL data type:

Source	Target	With IEC check	Without IEC check	Description
REAL	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.
	SINT	-	X	The bit pattern of the source value is rounded off and converted and transferred to the target data type. (for example, rounding off and value conversion of REAL #2.5 -> INT #2 or negative number REAL #-2.5 -> INT #-2 -> USINT #254. With an overflow, the remainder is determined REAL #305.5 -> INT #306 -> USINT #50)
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	LREAL	X	X	The value is transferred to the target data type.
	TIME	-	-	No implicit conversion
	DTL	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

REAL (Page 2252)
Setting and canceling the IEC check (Page 2374)
Overview of data type conversion (Page 2372)
Explicit conversion of REAL (Page 2408)

Implicit conversion of LREAL

Options for implicit conversion

The following table shows the options for implicit conversion of the LREAL data type:

Source	Target	With IEC check	Without IEC check	Explanation
LREAL	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	-	
	SINT	-	X	The bit pattern of the source value is rounded off and converted and transferred to the target data type. (for example, rounding off and value conversion of REAL #2.5 -> INT #2 or negative number REAL #-2.5 -> INT #-2 -> USINT #254. With an overflow, the remainder is determined REAL #305.5 -> INT #306 -> USINT #50)
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	REAL	-	-	No implicit conversion
	TIME	-	-	
	DTL	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

Explicit conversion of LREAL (Page 2410)

Timers

Implicit conversion of TIME

Options for implicit conversion

The following table shows the options for implicit conversion of the TIME data type:

Source	Target	With IEC check	Without IEC check	Description
TIME	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	X	The bit pattern of the source value is transferred unchanged to the target data type. The result of the conversion shows the duration in milliseconds.
	SINT	-	-	No implicit conversion
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	X	The bit pattern of the source value is transferred unchanged to the target data type. The result of the conversion shows the duration in milliseconds.
	UDINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	DTL	-	-	
	TOD	-	X	The bit pattern of a source value that is less than 24 h (86 400 00 ms) is transferred without changes to the target data type. No further changes are made to the target value. The result of the conversion shows the time that has passed since midnight.
	DATE	-	-	No implicit conversion
	STRING	-	-	
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

TIME (IEC time) (Page 2257)
Setting and canceling the IEC check (Page 2374)
Overview of data type conversion (Page 2372)
Explicit conversion of TIME (Page 2411)

Date and time

Implicit conversion of DTL

Options for implicit conversion

The DTL data type cannot be implicitly converted.

See also

Explicit conversion of DTL (Page 2414)

Implicit conversion of TOD

Options for implicit conversion

The following table shows the options for implicit conversion of the TOD data type:

Source	Target	With IEC check	Without IEC check	Description
TOD	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	X	The bit pattern of the source value is transferred unchanged to the target data type. The result of the conversion corresponds to the number of milliseconds since the start of day (0:00 hrs).
	SINT	-	-	No implicit conversion
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	X	The bit pattern of the source value is transferred unchanged to the target data type. The result of the conversion corresponds to the number of milliseconds since the start of day (0:00 hrs).
	UDINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	TIME	-	X	The bit pattern of the source value is transferred unchanged to the target data type. The result of the conversion corresponds to the number of milliseconds since the start of day (0:00 hrs).
	DTL	-	-	No implicit conversion
	DATE	-	-	
	STRING	-	-	
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

TIME_OF_DAY (TOD) (Page 2259)
Setting and canceling the IEC check (Page 2374)
Overview of data type conversion (Page 2372)
Explicit conversion of TOD (Page 2413)

Implicit conversion of DATE

Options for implicit conversion

The following table shows the options for implicit conversion of the DATE data type:

Source	Target	With IEC check	Without IEC check	Description
DATE	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	X	The bit pattern of the source value is transferred unchanged to the target data type. The result of the conversion corresponds to the number of days since 01-01-1990.
	DWORD	-	-	No implicit conversion
	SINT	-	-	
	USINT	-	-	
	INT	-	X	The bit pattern of the source value is transferred unchanged to the target data type. The result of the conversion corresponds to the number of days since 01-01-1990.
	UINT	-	X	
	DINT	-	-	No implicit conversion
	UDINT	-	-	
	REAL	-	-	
	LREAL	-	-	
	TIME	-	-	
	DTL	-	X	The bit pattern of the source value is transferred unchanged to the target data type. The result of the conversion corresponds to the number of days since 01-01-1990.
	TOD	-	-	No implicit conversion
	STRING	-	-	
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

DATE (Page 2258)
Setting and canceling the IEC check (Page 2374)

Overview of data type conversion (Page 2372)
Explicit conversion of DATE (Page 2412)

Character strings

Implicit conversion of CHAR

Options for implicit conversion

The following table shows the options for implicit conversion of the CHAR data type:

Source	Target	With IEC check	Without IEC check	Description
CHAR	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type. The remaining bits are filled from the left with "0".
	WORD	-	X	
	DWORD	-	X	
	SINT	-	X	
	USINT	-	X	
	INT	-	X	
	UINT	-	X	
	DINT	-	X	
	UDINT	-	X	
	REAL	-	-	No implicit conversion
	LREAL	-	-	
	TIME	-	-	
	DTL	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	X	X	The STRING is shortened to length 1 and includes the character.
x: Conversion possible -: Conversion not possible				

See also

CHAR (character) (Page 2263)
Setting and canceling the IEC check (Page 2374)
Overview of data type conversion (Page 2372)
Explicit conversion of CHAR (Page 2415)

Implicit conversion of STRING

Options for implicit conversion

The following table shows the options for implicit conversion of the STRING data type:

Source	Target	With IEC check	Without IEC check	Description
STRING	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	-	
	SINT	-	-	
	USINT	-	-	
	INT	-	-	
	UINT	-	-	
	DINT	-	-	
	UDINT	-	-	
	REAL	-	-	
	LREAL	-	-	
	TIME	-	-	
	DTL	-	-	
	DATE	-	-	
	TOD	-	-	
	CHAR	-	X	The first character of the STRING is returned if the STRING includes one or more characters. Otherwise, the character is output with coding $00.
x: Conversion possible -: Conversion not possible				

See also

Explicit conversion of STRING (Page 2416)

Explicit conversion

Binary numbers

Explicit conversion of BOOL

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the BOOL data type:

Source	Target	Conversion	Description	Mnemonics of the instruction
BOOL	BYTE	X	Only the LSB (Least Significant Bit) is set in the target data type. The enable output ENO is always "1".	BOOL_TO_BYTE
	WORD	X		BOOL_TO_WORD
	DWORD	X		BOOL_TO_DWORD
	SINT	X		BOOL_TO_SINT
	USINT	X		BOOL_TO_USINT
	INT	X		BOOL_TO_INT
	UINT	X		BOOL_TO_UINT
	DINT	X		BOOL_TO_DINT
	UDINT	X		BOOL_TO_UDINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	TIME	-		-
	DTL	-		-
	TOD	-		-
	DATE	-		-
	STRING	-		-
	CHAR	-		-
x: Conversion possible - : Conversion not possible				

See also

BOOL (bit) (Page 2240)

Implicit conversion of BYTE (Page 2375)

Overview of data type conversion (Page 2372)

Bit strings

Explicit conversion of BYTE

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the BYTE data type:

Source	Target	Conversion	Description	Mnemonics of Instruction
BYTE[1)]	BOOL	X	The following scenarios are possible: • If the source is "0", the target data type is also "0" and enable output ENO is "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the target data type is also "1" and enable output ENO is "1". • If bits are not equal to LSB in the source value, the target data type is set according to LSB and enable output ENO is "0".	BYTE_TO_BOOL
	WORD[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.	BYTE_TO_WORD
	DWORD[1)]	X		BYTE_TO_DWORD
	SINT	X		BYTE_TO_SINT
	USINT	X		BYTE_TO_USINT
	INT	X		BYTE_TO_INT
	UINT	X		BYTE_TO_UINT
	DINT	X		BYTE_TO_DINT
	UDINT	X		BYTE_TO_UDINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	TIME	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.	BYTE_TO_TIME
	DTL	-	No explicit conversion	-
	TOD	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.	BYTE_TO_TOD
	DATE	X		BYTE_TO_DATE
	STRING	-	No explicit conversion	-

Source	Target	Conversion	Description	Mnemonics of Instruction
	CHAR	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.	BYTE_TO_CHAR
x: Conversion possible - : Conversion not possible 1) Bit strings (BYTE, WORD, DWORD) are interpreted as an unsigned integer with the same bit length.. Data type BYTE is interpreted as USINT, WORD as UINT and DWORD as UDINT.				

See also

BYTE (byte) (Page 2240)

Implicit conversion of BYTE (Page 2375)

Overview of data type conversion (Page 2372)

Explicit conversion of WORD

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the WORD data type:

Source	Target	Conversion	Description	Mnemonics of the instruction
WORD[1)]	BOOL	X	The following scenarios are possible: • If the source is "0", the target data type is also "0" and enable output ENO is "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the target data type is also "1" and enable output ENO is "1". • If bit is not equal to LSB in the source value, the target data type is set according to LSB and enable output ENO is "0".	WORD_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.	WORD_TO_BYTE
	DWORD[1)]	X		WORD_TO_DWORD

Source	Target	Conversion	Description	Mnemonics of the instruction

	SINT	X	ENO = TRUE
			#sint1 := WORD_TO_SINT(16#FFFF); // -1 to
			#sint1 := WORD_TO_SINT(16#FF80); // -128
			#sint1 := WORD_TO_SINT(16#0); // 0 to
			#sint1 := WORD_TO_SINT(16#007F); // 127
			
			ENO = FALSE
			#sint1 := WORD_TO_SINT(16#FF7F); // -129 to
			#sint1 := WORD_TO_SINT(16#8000); // -32768
			#sint1 := WORD_TO_SINT(16#0080); // 128 to
			#sint1 := WORD_TO_SINT(16#7FFF); // 32767
				WORD_TO_SINT

	USINT	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.	WORD_TO_USINT
	INT	X		WORD_TO_INT
	UINT	X		WORD_TO_UINT
	DINT	X		WORD_TO_DINT
	UDINT	X		WORD_TO_UDINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	TIME	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.	WORD_TO_TIME
	DTL	-	No explicit conversion	-
	TOD	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.	WORD_TO_TOD
	DATE	X		WORD_TO_DATE
	STRING	-	No explicit conversion	-
	CHAR	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.	WORD_TO_CHAR
WORD_BCD16	INT	X	The value to be converted has data type WORD and is accepted as a BCD-coded value between -999 and +999. The result is available after conversion as an integer (in binary notation) of the type INT. A real conversion takes place. If the bit pattern includes an invalid tetrad, a synchronous error is not triggered but only the status bit OV is set instead.	WORD_BCD16_TO_INT
BCD16	INT	X		BCD16_TO_INT

x: Conversion possible

- : Conversion not possible

[1] Bit strings (BYTE, WORD, DWORD) are interpreted as an unsigned integer with the same bit length.. Data type BYTE is interpreted as USINT, WORD as UINT and DWORD as UDINT.

See also

WORD (Page 2241)
Implicit conversion of WORD (Page 2376)
Overview of data type conversion (Page 2372)

Explicit conversion of DWORD

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the DWORD data type:

Source	Target	Conversion	Description	Mnemonics of the instruction
DWORD[1)]	BOOL	X	The following scenarios are possible: • If the source is "0", the target data type is also "0" and enable output ENO is "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the target data type is also "1" and enable output ENO is "1". • If bit is not equal to LSB in the source value, the target data type is set according to LSB and enable output ENO is "0".	DWORD_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.	DWORD_TO_BYTE
	WORD[1)]	X		DWORD_TO_WORD

Source	Target	Conversion	Description	Mnemonics of the instruction

	SINT	X	ENO = TRUE
			#sint1 := DWORD_TO_SINT(16#FFFF_FFFF); // -1 bis
			#sint1 := DWORD_TO_SINT(16#FFFF_FF80); // -128
			#sint1 := DWORD_TO_SINT(16#0); // 0 bis
			#sint1 := DWORD_TO_SINT(16#0000_007F); // 127
			
			ENO = FALSE
			#sint1 := DWORD_TO_SINT(16#FFFF_FF7F); // -129
			#sint1 := DWORD_TO_SINT(16#8000_0000); // -2147483648
			#sint1 := DWORD_TO_SINT(16#0000_0080); // 128 bis
			#sint1 := DWORD_TO_SINT(16#7FFF_FFFF); // 2147483647
				DWORD_TO_SINT

	USINT	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.	DWORD_TO_USINT

Source	Target	Conversion	Description	Mnemonics of the instruction

	INT	X	ENO = TRUE
			#int1 := DWORD_TO_INT(16#FFFF_FFFF); // -1 bis
			#int1 := DWORD_TO_INT(16#FFFF_8000); // -32768
			#int1 := DWORD_TO_INT(16#0); // 0 bis
			#int1 := DWORD_TO_INT(16#0000_7FFF); // 32767
			
			ENO = FALSE
			#int1 := DWORD_TO_INT(16#FFFF_7FFF); // -32769
			#int1 := DWORD_TO_INT(16#8000_0000); // -2147483648
			#int1 := DWORD_TO_INT(16#8000); // 32768 bis
			#int1 := DWORD_TO_INT(16#7FFF_FFFF); // 2147483647
				DWORD_TO_INT

	UINT	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.	DWORD_TO_UINT
	DINT	X		DWORD_TO_DINT
	UDINT	X		DWORD_TO_UDINT
	REAL	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type. If no errors occur during the conversion, the signal state of ENO = 1; if an error occurs during processing, the signal state of ENO = 0.	DWORD_TO_REAL
	LREAL	-	No explicit conversion	-
	TIME	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.	DWORD_TO_TIME
	DTL	-	No explicit conversion	-
	TOD	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.	DWORD_TO_TOD
	DATE	X		DWORD_TO_DATE
	STRING	-	No explicit conversion	-
	CHAR	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.	DWORD_TO_CHAR
DWORD_BCD32	DINT	X	The value to be converted has data type DWORD and is accepted as a BCD-coded value between -9999999	DWORD_BCD32_TO_DINT

Source	Target	Conversion	Description	Mnemonics of the instruction
BCD32	DINT	X	and +9999999. The result is available after conversion as an integer (in binary notation) of the type DINT. A real conversion takes place. If the bit pattern includes an invalid tetrad, a synchronous error is not triggered but only the status bit OV is set instead.	BCD32_TO_DINT
x: Conversion possible - : Conversion not possible 1) Bit strings (BYTE, WORD, DWORD) are interpreted as an unsigned integer with the same bit length.. Data type BYTE is interpreted as USINT, WORD as UINT and DWORD as UDINT.				

See also

DWORD (Page 2242)

Implicit conversion of DWORD (Page 2377)

Overview of data type conversion (Page 2372)

Integers

Explicit conversion of SINT

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the SINT data type:

Source	Target	Conversion	Description	Mnemonics of the instruction
SINT	BOOL	X	The following scenarios are possible: • If the source is "0", the target data type is also "0" and enable output ENO is "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the target data type is also "1" and enable output ENO is "1". • If bit is not equal to LSB in the source value, the target data type is set according to LSB and enable output ENO is "0".	SINT_TO_BOOL
	BYTE1)	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type. If a negative value is converted to an unsigned target data type, the enable output ENO is set to "0".	SINT_TO_BYTE
	WORD1)	X		SINT_TO_WORD
	DWORD1)	X		SINT_TO_DWORD
	USINT	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF).	SINT_TO_USINT
	INT	X		SINT_TO_INT
	UINT	X		SINT_TO_UINT

Source	Target	Conversion	Description	Mnemonics of the instruction
	DINT	X	Enable output ENO is set to "0" if a negative value is converted to an unsigned target data type.	SINT_TO_DINT
	UDINT	X		SINT_TO_UDINT
	REAL	X	The value is converted into the format of the target data type (the value "-1" will be converted into the value "-1.0" with the "Convert value" (CONVERT) instruction.	SINT_TO_REAL, NORM_X
	LREAL	X		SINT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the target data type and interpreted as milliseconds.	SINT_TO_TIME
	DTL	-	No explicit conversion	-
	TOD	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). Enable output ENO is set to "0" if a negative value is converted to an unsigned target data type. (interpretation in milliseconds since 0:0)	SINT_TO_TOD
	DATE	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). Enable output ENO is set to "0" if a negative value is converted to an unsigned target data type. (interpretation in days since 1990-1-1)	SINT_TO_DATE
	STRING	X	The value is converted to a character string. The character string is shown preceded by a sign. If the permitted length of the character string is violated, the enable output ENO is set to "0".	SINT_TO_STRING, S_CONV, VAL_STRG
	CHAR[1)]	X	The bit pattern of the source value is transferred unchanged to the target data type. If negative values are converted, the enable output ENO is set to "0".	SINT_TO_CHAR

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD) and data type CHAR are first extended to the necessary width including sign, and then the bits are copied. The source type determines the interpretation.

See also

SINT (8-bit integers) (Page 2244)

Implicit conversion of SINT (Page 2378)

Overview of data type conversion (Page 2372)

Explicit conversion of USINT

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the USINT data type:

Source	Target	Conversion	Description	Mnemonics of the instruction
USINT	BOOL	X	The following scenarios are possible: • If the source is "0", the target data type is also "0" and enable output ENO is "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the target data type is also "1" and enable output ENO is "1". • If bit is not equal to LSB in the source value, the target data type is set according to LSB and enable output ENO is "0".	USINT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.	USINT_TO_BYTE
	WORD[1)]	X		USINT_TO_WORD
	DWORD[1)]	X		USINT_TO_DWORD
	SINT	X	The bit pattern of the source value is transferred unchanged to the target data type. If the sign is changed during the conversion, the enable output ENO is set to "0".	USINT_TO_SINT
	INT	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.	USINT_TO_INT
	UINT	X		USINT_TO_UINT
	DINT	X		USINT_TO_DINT
	UDINT	X		USINT_TO_UDINT
	REAL	X	The value is converted into the format of the target data type (the value "1" will be converted into the value "1.0" with the "Convert value" (CONVERT) instruction.	USINT_TO_REAL, NORM_X
	LREAL	X		USINT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the target data type and interpreted as milliseconds.	USINT_TO_TIME
	DTL	-	No explicit conversion	-
	TOD	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.	USINT_TO_TOD
	DATE	X		USINT_TO_DATE
	STRING	X	The value is converted to a character string. If the permitted length of the character string is violated, the enable output ENO is set to "0".	USINT_TO_STRING, S_CONV, VAL_STRG

Source	Target	Conversion	Description	Mnemonics of the instruction
	CHAR[1)]	X	The bit pattern of the source value is transferred unchanged to the target data type.	USINT_TO_CHAR
x: Conversion possible - : Conversion not possible [1)] Bit strings (BYTE, WORD, DWORD) and data type CHAR are first extended to the necessary width (the non-existing sign is replaced with zeros) and then the bits are copied. The source type determines the interpretation.				

See also

USINT (8-bit integers) (Page 2245)

Implicit conversion of USINT (Page 2379)

Overview of data type conversion (Page 2372)

Explicit conversion of INT

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the INT data type:

Source	Target	Conversion	Description	Mnemonics of the instruction
INT	BOOL	X	The following scenarios are possible: • If the source is "0", the target data type is also "0" and enable output ENO is "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the target data type is also "1" and enable output ENO is "1". • If bit is not equal to LSB in the source value, the target data type is set according to LSB and enable output ENO is "0".	INT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type. If a negative value is converted to an unsigned target data type, the enable output ENO is set to "0".	INT_TO_BYTE
	WORD[1)]	X		INT_TO_WORD
	DWORD[1)]	X		INT_TO_DWORD
	SINT	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). Enable output ENO is set to "0" if a negative value is converted to an unsigned target data type.	INT_TO_SINT
	USINT	X		INT_TO_USINT
	UINT	X		INT_TO_UINT
	DINT	X		INT_TO_DINT
	UDINT	X		INT_TO_UDINT
	REAL	X	The value is converted to the format of the target data type (the value "-1", for example, is converted with the instruction "Convert value" (CONVERT) to the value "-1.0").	INT_TO_REAL, NORM_X
	LREAL	X		INT_TO_LREAL, NORM_X

Source	Target	Conversion	Description	Mnemonics of the instruction
	TIME	X	The value is transferred to the target data type and interpreted as milliseconds.	INT_TO_TIME
	DTL	-	No explicit conversion	-
	TOD	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). Enable output ENO is set to "0" if a negative value is converted to an unsigned target data type. (interpretation in milliseconds since 0:0; check for 24h limit)	INT_TO_TOD
	DATE	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). Enable output ENO is set to "0" if a negative value is converted to an unsigned target data type. (interpretation in days since 1990-1-1; check for negative value)	INT_TO_DATE
	STRING	X	The value is converted to a character string. The character string is shown preceded by a sign. If the permitted length of the character string is violated, the enable output ENO is set to "0".	INT_TO_STRING, S_CONV, VAL_STRG[)]
	CHAR[1)]	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). Enable output ENO is set to "0" if a negative value is converted to an unsigned target data type.	INT_TO_CHAR
	BCD16	X	The value to be converted has type INT and is accepted as an integer with a value between -999 and +999. The result is available after conversion as a BCD-coded number of the type WORD. A real conversion takes place. If the value is outside the target area, a synchronous error is not triggered, but rather only the status bit OV is set.	INT_TO_BCD16
	BCD16_WORD	X		INT_TO_BCD16_WORD

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD) and data type CHAR are first extended to the necessary width including sign, and then the bits are copied. The source type determines the interpretation.

See also

INT (16-bit integers) (Page 2245)

Implicit conversion of INT (Page 2380)

Overview of data type conversion (Page 2372)

Explicit conversion of UINT

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the UINT data type:

Source	Target	Conversion	Description	Mnemonics of the instruction
UINT	BOOL	X	The following scenarios are possible: • If the source is "0", the target data type is also "0" and enable output ENO is "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the target data type is also "1" and enable output ENO is "1". • If bit is not equal to LSB in the source value, the target data type is set according to LSB and enable output ENO is "0".	UINT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type. If bits are lost in the process, the enable output ENO is set to "0".	UINT_TO_BYTE
	WORD[1)]	X		UINT_TO_WORD
	DWORD[1)]	X		UINT_TO_DWORD
	SINT	X		UINT_TO_SINT
	USINT	X		UINT_TO_USINT
	INT	X	The bit pattern of the source value is transferred unchanged to the target data type. If the sign bit is changed during the conversion, the enable output ENO is set to "0".	UINT_TO_INT
	DINT	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.	UINT_TO_DINT
	UDINT	X		UINT_TO_UDINT
	REAL	X	The value is converted to the format of the target data type (the value "1", for example, is converted with the instruction "Convert value" (CONVERT) to the value "1.0").	UINT_TO_REAL, NORM_X
	LREAL	X		UINT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the target data type and interpreted as milliseconds.	UINT_TO_TIME
	DTL	-	No explicit conversion	-
	TOD	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). Enable output ENO is set to "0" if a negative value is converted to an unsigned target data type. (interpretation in milliseconds since 0:0; check for 24h limit)	UINT_TO_TOD

Source	Target	Conversion	Description	Mnemonics of the instruction
	DATE	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). Enable output ENO is set to "0" if a negative value is converted to an unsigned target data type. (interpretation in days since 1990-1-1; check for negative value)	UINT_TO_DATE, T_CONV
	STRING	X	The value is converted to a character string. If the permitted length of the character string is violated, the enable output ENO is set to "0".	UINT_TO_STRING, S_CONV, VAL_STRG
	CHAR[1)]	X	The bit pattern of the source value is transferred unchanged to the target data type. The enable output ENO is set to "0" in the event of overflow.	UINT_TO_CHAR

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD, LWORD) and the data type CHAR are initially extended to the necessary width (the non-existing sign is replaced with zeros) and then the bits are copied. The source type determines the interpretation.

See also

UINT (16-bit integers) (Page 2246)

Implicit conversion of UINT (Page 2381)

Overview of data type conversion (Page 2372)

Explicit conversion of DINT

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the DINT data type:

Source	Target	Conversion	Description	Mnemonics of the instruction
DINT	BOOL	X	The following scenarios are possible: • If the source is "0", the target data type is also "0" and enable output ENO is "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the target data type is also "1" and enable output ENO is "1". • If bit is not equal to LSB in the source value, the target data type is set according to LSB and enable output ENO is "0".	DINT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type. If a negative value is converted to an unsigned target data type, the enable output ENO is set to "0".	DINT_TO_BYTE
	WORD[1)]	X		DINT_TO_WORD
	DWORD[1)]	X		DINT_TO_DWORD
	SINT	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). Enable output ENO is set to "0" if a negative value is converted to an unsigned target data type.	DINT_TO_SINT
	USINT	X		DINT_TO_USINT
	INT	X		DINT_TO_INT
	UINT	X		DINT_TO_UINT
	UDINT	X		DINT_TO_UDINT
	REAL	X	The value is converted to the format of the target data type (the value "-1", for example, is converted with the instruction "Convert value" (CONVERT) to the value "-1.0").	DINT_TO_REAL, NORM_X
	LREAL	X		DINT_TO_LREAL, NORM_X
	TIME	X	The value is transferred to the target data type and interpreted as milliseconds.	DINT_TO_TIME, T_CONV
	DTL	-	No explicit conversion	-
	TOD	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). Enable output ENO is set to "0" if a negative value is converted to an unsigned target data type. (interpretation in milliseconds since 0:0)	DINT_TO_TOD

Source	Target	Conversion	Description	Mnemonics of the instruction
	DATE	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). Enable output ENO is set to "0" if a negative value is converted to an unsigned target data type. (interpretation in days since 1990-1-1)	DINT_TO_DATE
	STRING	X	The value is converted to a character string. The character string is shown preceded by a sign. If the permitted length of the character string is violated, the enable output ENO is set to "0".	DINT_TO_STRING, S_CONV, VAL_STRG
	CHAR[1)]	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). Enable output ENO is set to "0" if a negative value is converted to an unsigned target data type.	DINT_TO_CHAR
	BCD32	X	The value to be converted has type DINT and is accepted as an integer with a value between –999999 and +9999999. The result is available after conversion as a BCD-coded number of the type DWORD. The enable output is set to "0" in the event of overflow. A real conversion takes place. If the value is outside the target area, a synchronous error is not triggered, but rather only the status bit OV is set.	DINT_TO_BCD32
	BCD32_DWORD	X		DINT_TO_BCD32_DWORD

x: Conversion possible

- : Conversion not possible

[1)] Bit strings (BYTE, WORD, DWORD) and data type CHAR are first extended to the necessary width including sign, and then the bits are copied. The source type determines the interpretation.

See also

DINT (32-bit integers) (Page 2247)

Implicit conversion of DINT (Page 2382)

Overview of data type conversion (Page 2372)

Explicit conversion of UDINT

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the UDINT data type:

Source	Target	Conversion	Description	Mnemonics of the instruction
UDINT	BOOL	X	The following scenarios are possible: • If the source is "0", the target data type is also "0" and enable output ENO is "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the target data type is also "1" and enable output ENO is "1". • If bit is not equal to LSB in the source value, the target data type is set according to LSB and enable output ENO is "0".	UDINT_TO_BOOL
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged to the target data type. If bits are lost in the process, the enable output ENO is set to "0".	UDINT_TO_BYTE
	WORD[1)]	X		UDINT_TO_WORD
	DWORD[1)]	X		UDINT_TO_DWORD
	SINT	X		UDINT_TO_SINT
	USINT	X		UDINT_TO_USINT
	INT	X		UDINT_TO_INT
	UINT	X		UDINT_TO_UINT
	DINT	X	The bit pattern of the source value is transferred unchanged to the target data type. If the sign bit is changed during the conversion, the enable output ENO is set to "0".	UDINT_TO_DINT
	REAL	X	The value is converted to the format of the target data type (the value "1", for example, is converted with the instruction "Convert value" (CONVERT) to the value "1.0").	UDINT_TO_REAL, NORM_X
	LREAL	X		UDINT_TO_LREAL, NORM_X
	TIME	X	The bit pattern of the source value is transferred unchanged right-justified and interpreted as milliseconds to the target data type.	UDINT_TO_TIME
	DTL	-	No explicit conversion	-
	TOD	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). Enable output ENO is set to "0" if a negative value is converted to an unsigned target data type. (interpretation in milliseconds since 0:0; check for 24h limit)	UDINT_TO_TOD, T_CONV

Source	Target	Conversion	Description	Mnemonics of the instruction
	DATE	X	The bit pattern of the source value is converted and transferred to the target data type. (The value "-1" (16#FF) becomes the value "-1" (16#FFFFFFFF). Enable output ENO is set to "0" if a negative value is converted to an unsigned target data type. (interpretation in days since 1990-1-1; check for negative value)	UDINT_TO_DATE
	STRING	X	The value is converted to a character string. If the permitted length of the character string is violated, the enable output ENO is set to "0".	UDINT_TO_STRING, S_CONV, VAL_STRG
	CHAR[1)]	X	The bit pattern of the source value is transferred unchanged to the target data type. The enable output ENO is set to "0" in the event of overflow.	UDINT_TO_CHAR
x: Conversion possible - : Conversion not possible [1)] Bit strings (BYTE, WORD, DWORD, LWORD) and the data type CHAR are initially extended to the necessary width (the non-existing sign is replaced with zeros) and then the bits are copied. The source type determines the interpretation.				

See also

UDINT (32-bit integers) (Page 2248)

Implicit conversion of UDINT (Page 2383)

Overview of data type conversion (Page 2372)

Floating-point numbers

Explicit conversion of REAL

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the REAL data type:

Source	Target	Conversion	Description	Mnemonics of the instruction
REAL	BOOL	-	No explicit conversion	-
	BYTE	-		-
	WORD	-		-
	DWORD	X	The bit pattern of the source value is transferred unchanged to the target data type.	REAL_TO_DWORD
	SINT	X	The value is converted to the target data type. The result of the conversion depends on the instruction used. Enable output ENO is set to "0" if the valid range of values of the target	REAL_TO_SINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X

Source	Target	Conversion	Description	Mnemonics of the instruction
	USINT	X	data type is exceeded during conversion, or if the value to be converted is an invalid floating-point number.	REAL_TO_USINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	INT	X		REAL_TO_INT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	UINT	X		REAL_TO_UINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	DINT	X		REAL_TO_DINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	UDINT	X		REAL_TO_UDINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	LREAL	X	The value is converted to the target data type. The result of the conversion depends on the instruction used, e.g. TRUNC(2.5) = 2.0; CEIL(2.5) = 3.0	REAL_TO_LREAL, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	TIME	-	No explicit conversion	-
	DTL	-		-
	TOD	-		-
	DATE	-		-
	STRING	X	The value is converted to a character string. Enable output ENO is set to "0" if the character string length is exceeded, or if the value to be converted is an invalid floating-point number. The string has a minimum length of 14 characters.	REAL_TO_STRING, S_CONV, VAL_STRG
	CHAR	-	No explicit conversion	-
x: Conversion possible -: Conversion not possible				

See also

REAL (Page 2252)

Implicit conversion of REAL (Page 2384)

Overview of data type conversion (Page 2372)

Explicit conversion of LREAL

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the LREAL data type:

Source	Target	Conversion	Description	Mnemonics of the instruction
LREAL	BOOL	-	No explicit conversion	-
	BYTE	-		-
	WORD	-		-
	DWORD	-		-
	SINT	X	The value is converted to the target data type. The result of the conversion depends on the instruction used. Enable output ENO is set to "0" if the valid range of values of the target data type is exceeded during conversion, or if the value to be converted is an invalid floating-point number.	LREAL_TO_SINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	USINT	X		LREAL_TO_USINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	INT	X		LREAL_TO_INT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	UINT	X		LREAL_TO_UINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	DINT	X		LREAL_TO_DINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	UDINT	X		LREAL_TO_UDINT, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	REAL	X	The value is converted to the target data type. Enable output ENO is set to "0" if the valid range of values of the target data type is exceeded during conversion, or if the value to be converted is an invalid floating-point number. A loss in accuracy is tolerated.	LREAL_TO_LREAL, ROUND, CEIL, FLOOR, TRUNC, NORM_X, SCALE_X
	TIME	-	No explicit conversion	-
	DTL	-		-
	TOD	-		-
	DATE	-		-
	STRING	X	The value is converted to a character string. Enable output ENO is set to "0" if the character string length is exceeded, or if the value to be converted is an invalid floating-point number. The string has a minimum length of 21 characters.	REAL_TO_STRING, S_CONV, VAL_STRG

Source	Target	Conversion	Description	Mnemonics of the instruction
	CHAR	-	No explicit conversion	-
x: Conversion possible -: Conversion not possible				

See also

LREAL (Page 2253)

Overview of data type conversion (Page 2372)

Timers

Explicit conversion of TIME

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the TIME data type:

Source	Target	Conversion	Description	Mnemonics of the instruction
TIME	BOOL	-	No explicit conversion	-
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged to the target data type.	TIME_TO_BYTE
	WORD[1)]	X		TIME_TO_WORD
	DWORD[1)]	X		TIME_TO_DWORD
	SINT	X	The bit pattern of the source value is transferred unchanged right-justified and interpreted as milliseconds to the target data type.	TIME_TO_SINT
	USINT	X		TIME_TO_USINT
	INT	X		TIME_TO_INT
	UINT	X		TIME_TO_UINT
	DINT	X	The bit pattern of the source value is transferred unchanged to the target data type. The result of the conversion shows the duration in milliseconds.	TIME_TO_DINT, T_CONV
	UDINT	X	The bit pattern of the source value is transferred unchanged right-justified and interpreted as milliseconds to the target data type. Enable output ENO is set to "0" if the sign changes.	TIME_TO_UDINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	DTL	-		-

Source	Target	Conversion	Description	Mnemonics of the instruction
	TOD	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type. If the source value exceeds the range of values of TOD, the target data type remains unchanged.	TIME_TO_TOD
	DATE	-	No explicit conversion	-
	STRING	-		-
	CHAR	-		-
x: Conversion possible -: Conversion not possible 1) Bit strings (BYTE, WORD, DWORD) and data type CHAR are first extended to the necessary width and then the bits are copied. The source type determines the interpretation.				

See also

TIME (IEC time) (Page 2257)

Implicit conversion of TIME (Page 2386)

Overview of data type conversion (Page 2372)

Clock and calendar

Explicit conversion of DATE

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the DATE data type:

Source	Target	Conversion	Description	Mnemonics of the instruction
DATE	BOOL	-	No explicit conversion	-
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.	DATE_TO_BYTE
	WORD[1)]	X		DATE_TO_WORD
	DWORD[1)]	X		DATE_TO_DWORD
	SINT	X	The number of days since 1/1/1990 is returned as result.	DATE_TO_SINT
	USINT	X		DATE_TO_USINT
	INT	X		DATE_TO_INT
	UINT	X		DATE_TO_UINT
	DINT	X		DATE_TO_DINT
	UDINT	X		DATE_TO_UDINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	TIME	-		-

Source	Target	Conversion	Description	Mnemonics of the instruction
	DTL	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.	DATE_TO_DTL
	TOD	-	No explicit conversion	-
	STRING	-		-
	CHAR	-		-

x: Conversion possible

- : Conversion not possible

[1] Bit strings (BYTE, WORD, DWORD) and data type CHAR are first extended to the necessary width and then the bits are copied. The source type determines the interpretation.

See also

DATE (Page 2258)

Implicit conversion of DATE (Page 2388)

Overview of data type conversion (Page 2372)

Explicit conversion of TOD

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the TOD data type:

Source	Target	Conversion	Description	Mnemonics of the instruction
TOD	BOOL	-	No explicit conversion	-
	BYTE[1]	X	The bit pattern of the source value is transferred unchanged to the target data type.	TOD_TO_BYTE
	WORD[1]	X		TOD_TO_WORD
	DWORD[1]	X		TOD_TO_DWORD
	SINT	X	The number of milliseconds since midnight is returned as result.	TOD_TO_SINT
	USINT	X		TOD_TO_USINT
	INT	X		TOD_TO_INT
	UINT	X		TOD_TO_UINT
	DINT	X		TOD_TO_DINT
	UDINT	X	The result of the conversion corresponds to the number of milliseconds since the start of day (0:00 hrs).	TOD_TO_UDINT, T_CONV
	REAL	-	No explicit conversion	-
	LREAL	-		-
	TIME	X	The duration since midnight is returned as result.	TOD_TO_TIME
	DTL	X	The date is set to 1.1.1970 as a result.	TOD_TO_DTL

Source	Target	Conversion	Description	Mnemonics of the instruction
	DATE	-	No explicit conversion	-
	STRING	-		-
	CHAR	-		-
x: Conversion possible -: Conversion not possible 1) Bit strings (BYTE, WORD, DWORD) and data type CHAR are first extended to the necessary width including the sign and then the bits are copied. The source type determines the interpretation.				

See also

TIME_OF_DAY (TOD) (Page 2259)
Implicit conversion of TOD (Page 2387)
Overview of data type conversion (Page 2372)

Explicit conversion of DTL

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the DTL data type:

Source	Target	Conversion	Description	Mnemonics of the instruction
DTL	BYTE	-	No explicit conversion	-
	WORD	-		-
	DWORD	-		-
	SINT	-		-
	USINT	-		-
	INT	-		-
	UINT	-		-
	DINT	-		-
	UDINT	-		-
	REAL	-		-
	LREAL	-		-
	TIME	-		-
	TOD	X	During the conversion, the time-of-day is extracted from the DTL format and written to the target data type.	DTL_TO_TOD, T_CONV
	DATE	X	During the conversion, the date is extracted from the DTL format and written to the target data type. The enable output ENO is set to "0" in the event of overflow.	DTL_TO_DATE, T_CONV
	STRING	-	No explicit conversion	-

Source	Target	Conversion	Description	Mnemonics of the instruction
	CHAR	-		-
x: Conversion possible -: Conversion not possible				

See also

DTL (Page 2262)

Overview of data type conversion (Page 2372)

Character strings

Explicit conversion of CHAR

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the CHAR data type:

Source	Target	Conversion	Description	Mnemonics of the instruction
CHAR	BOOL	-	No explicit conversion	-
	BYTE[1)]	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.	CHAR_TO_BYTE
	WORD[1)]	X		CHAR_TO_WORD
	DWORD[1)]	X		CHAR_TO_DWORD
	SINT	X		CHAR_TO_SINT
	USINT	X		CHAR_TO_USINT
	INT	X		CHAR_TO_INT
	UINT	X		CHAR_TO_UINT
	DINT	X		CHAR_TO_DINT
	UDINT	X		CHAR_TO_UDINT
	REAL	-	No explicit conversion	-
	LREAL	-		-
	TIME	-		-
	DTL	-		-
	TOD	-		-
	DATE	-		-

Source	Target	Conversion	Description	Mnemonics of the instruction
	STRING	X	The value is converted to the first character in the character string (STRING). If the length of the character string is not defined, the length "1" is set after the conversion. If the length of the character string is defined, this remains unchanged after the conversion.	CHAR_TO_STRING
x: Conversion possible - : Conversion not possible 1) Bit strings (BYTE, WORD, DWORD) and data type CHAR are first extended to the necessary width and then the bits are copied. The source type determines the interpretation.				

See also

CHAR (character) (Page 2263)

Implicit conversion of CHAR (Page 2389)

Overview of data type conversion (Page 2372)

Explicit conversion of STRING

Options for explicit conversion

The following table shows the options and instructions for explicit conversion of the STRING data type:

Source	Target	Conversion	Description	Mnemonics of the instruction
STRING	BOOL	-	No explicit conversion	-
	BYTE	-		-
	WORD	-		-
	DWORD	-		-
	SINT	X	Conversion begins with the first character in the character string (STRING) and stops at the end of the string or at the first inadmissible character. The following characters are permitted for conversion: • Digit • Sign • Dot The first character of the string may be a sign (+, -) or a number. Leading spaces will be ignored. The dot is used as separation for the conversion of floating-point numbers. The exponential notation "e" or "E" is not permitted. The comma as thousand separator is permitted to the left of the decimal point but will be ignored. If the layout of the character string is invalid for the	STRING_TO_SINT, S_CONV, STRG_VAL
	USINT	X		STRING_TO_USINT, S_CONV, STRG_VAL
	INT	X		STRING_TO_INT, S_CONV, STRG_VAL
	UINT	X		STRING_TO_UINT, S_CONV, STRG_VAL
	DINT	X		STRING_TO_DINT, S_CONV, STRG_VAL
	UDINT	X		STRING_TO_UDINT, S_CONV, STRG_VAL
	REAL	X		STRING_TO_REAL, S_CONV, STRG_VAL

Source	Target	Conversion	Description	Mnemonics of the instruction
	LREAL	X	conversion or an overflow occurs, then the enable output ENO will be set to "0".	STRING_TO_LREAL, S_CONV, STRG_VAL
	TIME	-	No explicit conversion	-
	DTL	-		-
	TOD	-		-
	DATE	-		-
	CHAR[1)]	X	The first character in the character string (STRING) is transferred to the destination data type. If the string is empty, then the value "0" will be written in the destination data type.	STRING_TO_CHAR, S_CONV
x: Conversion possible - : Conversion not possible				

See also

STRING (Page 2264)

Overview of data type conversion (Page 2372)

Data type conversion

Overview of data type conversion

Introduction

If you combine several operands in an instruction, you must make sure that the data types are compatible. This also applies when you assign or supply block parameters. If the operands are not of the same data type, a conversion has to be carried out.

There are two options for conversion:

- Implicit conversion
  Implicit conversion is supported by the programming languages LAD, FBD, SCL and GRAPH. Implicit conversion is not possible in the STL programming language.
- Explicit conversion

Note

Converting bit strings to SCL

All bit strings (BYTE, WORD, DWORD and LWORD) are handled like the corresponding unsigned integers (USINT, UINT, UDINT and ULINT) in expressions. Therefore, implicit conversion from DWORD to REAL is carried out like a conversion from UDINT to REAL, for example.

Implicit conversion

Implicit conversion is executed automatically if the data types of the operands are compatible. This compatibility test can be carried out according to strict or less strict criteria:

- With IEC check
  The following rules apply in the LAD, FBD and GRAPH programming languages when the IEC check has been set:
  - The only data types for which implicit conversion is possible are BYTE and WORD.
  - The bit length of the source data type may not exceed the bit length of the target data type. A WORD data type operand, for example, cannot be specified at a parameter at which the BYTE data type is expected.

  The following rules apply in the SCL programming language when the IEC check has been set:
  - Implicit conversion of bit strings into other data types is not possible. A WORD data type operand, for example, cannot be specified at a parameter at which the INT data type is expected.
  - The bit length of the source data type may not exceed the bit length of the target data type. A WORD data type operand, for example, cannot be specified at a parameter at which the BYTE data type is expected.
- Without IEC check (default setting)
  The following rules apply in the LAD, FBD and GRAPH programming languages when the IEC check has not been set:
  - Implicit conversion of data types BYTE, WORD, DWORD, INT, DINT, TIME, S5TIME, TOD, DATE and CHAR is possible.
  - The bit length of the source data type may not exceed the bit length of the target data type. A DWORD data type operand, for example, cannot be specified at a parameter at which the WORD data type is expected.

  The following rules apply in the SCL programming language when the IEC check has not been set:
  - Bit strings can be implicitly converted into other data types. A WORD data type operand can, for example, be given at a parameter at which the WORD data type is expected.
  - Bit strings cannot be implicitly converted into floating-point numbers. A WORD data type operand, for example, cannot be specified at a parameter at which the REAL data type is expected.
  - Bit strings can only be implicitly converted into the data types TIME, TOD, DATE and CHAR if these have the same bit length. A DWORD data type operand, for example, cannot be specified at a parameter at which the DATE data type is expected.
  - The bit length of the source data type may not exceed the bit length of the target data type. A DINT data type operand, for example, cannot be specified at a parameter at which the INT data type is expected.
  - The bit length of an operand entered at in-out parameters must be the same as the programmed bit length for the parameter in question.

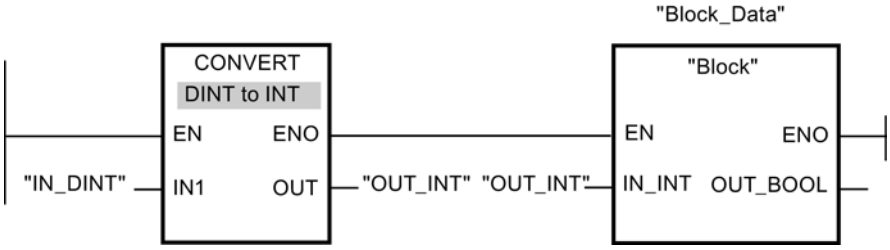
Explicit conversion

If the operands are not compatible and implicit conversion is therefore not possible, you can use an explicit conversion instruction. You can find the conversion instructions in the "Instructions" task card.

Any overflow will be displayed on the ENO enable output. An overflow occurs, for example, if the value of the source data type is greater than the value of the target data type.

You can find additional information on explicit conversion under "See also".

The following figure shows an example in which explicit data type conversion must be carried out:



The "Block" function block expects a tag of the INT data type at the "IN_INT" input parameter. The value of the "IN_DINT" tag has therefore to the converted first from DINT to INT. Conversion is performed if the value of the "IN_DINT" tag is within the admissible value range for the INT data type. Otherwise an overflow is reported. However, a conversion takes place even in the event of an overflow, but the values are truncated and the ENO enable output is set to "0".

See also

Setting and disabling the IEC check (Page 2419)

Implicit conversion (Page 2419)

Explicit conversion (Page 2430)

Implicit conversion

Setting and disabling the IEC check

The data types of the operands used are checked for compatibility. This compatibility test can be carried out according to criteria that are more or less strict. If "IEC check" is activated, stricter criteria are applied.

You can set the IEC check centrally for all new blocks of the project or for individual blocks.

Setting IEC check for new blocks

To set the IEC check for all new blocks in the project, proceed as follows:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "PLC programming > General" group in the area navigation.
3. Select or clear the "IEC check" check box in the "Default settings for new blocks" group.
   The IEC check is enabled or disabled for all new blocks in the program.

Setting IEC check for a block

To set the IEC check for a block, proceed as follows:

1. Open the block.
2. Open the "Properties" tab in the inspector window.
3. Select the "Attributes" group in the area navigation.
4. Select or clear the "IEC check" check box.
   The IEC check is enabled or disabled for this block. The setting is stored together with the project.

See also

Overview of data type conversion (Page 2417)

Binary numbers

Implicit conversion of BOOL

Implicit conversion options

The BOOL data type cannot be implicitly converted.

See also

BOOL (bit) (Page 2240)

Bit strings

Implicit conversion of BYTE

Implicit conversion options

The following table shows the options for the implicit conversion of BYTE data type:

Source	Target	With IEC check	Without IEC check	Explanation
BYTE	BOOL	-	-	No implicit conversion
	WORD	X	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.
	DWORD	X	X	
	INT	-	-	No implicit conversion
	DINT	-	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.
	REAL	-	-	No implicit conversion
	TIME	-	-	
	S5TIME	-	-	
	DT	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	
	CHAR	-	X	The bit pattern of the source value is transferred unchanged to the target data type.
x: Conversion possible -: Conversion not possible				

See also

BYTE (byte) (Page 2240)

Overview of data type conversion (Page 2417)

Explicit conversion of BYTE (Page 2431)

Setting and disabling the IEC check (Page 2419)

Implicit conversion of WORD

Implicit conversion options

The following table shows the options for the implicit conversion of WORD data type:

Source	Target	With IEC check	Without IEC check	Explanation
WORD	BOOL	-	-	No implicit conversion
	BYTE	-	X	Only the low byte is transferred to the destination data type.
	DWORD	X	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.
	INT	-	X	
	DINT	-	X	
	REAL	-	-	No implicit conversion
	TIME	-	-	
	S5TIME	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	DT	-	-	No implicit conversion
	TOD	-	-	
	DATE	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	STRING	-	-	No implicit conversion
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

WORD (Page 2241)
Overview of data type conversion (Page 2417)
Explicit conversion of WORD (Page 2432)
Setting and disabling the IEC check (Page 2419)

Implicit conversion of DWORD

Implicit conversion options

The following table shows the options for the implicit conversion of DWORD data type:

Source	Target	With IEC check	Without IEC check	Explanation
DWORD	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	WORD	-	X	

Source	Target	With IEC check	Without IEC check	Explanation
	INT	-	-	No implicit conversion
	DINT	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	REAL	-	-	No implicit conversion
	TIME	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	S5TIME	-	-	No implicit conversion
	DT	-	-	
	TOD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	DATE	-	-	No implicit conversion
	STRING	-	-	
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

DWORD (Page 2242)
Overview of data type conversion (Page 2417)
Explicit conversion of DWORD (Page 2434)
Setting and disabling the IEC check (Page 2419)

Integers

Implicit conversion of INT

Implicit conversion options

The following table shows the options for the implicit conversion of INT data type:

Source	Target	With IEC check	Without IEC check	Explanation
INT	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	DWORD	-	-	No implicit conversion
	DINT	X	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	REAL	X	X	
	TIME	-	-	No implicit conversion

Source	Target	With IEC check	Without IEC check	Explanation
	S5TIME	-	-	
	DT	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

INT (16-bit integers) (Page 2245)
Overview of data type conversion (Page 2417)
Explicit conversion of INT (Page 2436)
Setting and disabling the IEC check (Page 2419)

Implicit conversion of DINT

Implicit conversion options

The following table shows the options for the implicit conversion of DINT data type:

Source	Target	With IEC check	Without IEC check	Explanation
DINT	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	INT	-	-	No implicit conversion
	REAL	-	-	
	TIME	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	S5TIME	-	-	No implicit conversion
	DT	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

DINT (32-bit integers) (Page 2247)
Overview of data type conversion (Page 2417)
Explicit conversion of STRING (Page 2446)
Setting and disabling the IEC check (Page 2419)

Floating-point numbers

Implicit conversion of REAL

Implicit conversion options

The REAL data type cannot be implicitly converted.

See also

REAL (Page 2252)
Overview of data type conversion (Page 2417)
Explicit conversion of CHAR (Page 2445)
Setting and disabling the IEC check (Page 2419)

Timers

Implicit conversion of TIME

Implicit conversion options

The following table shows the options for the implicit conversion of TIME data type:

Source	Target	With IEC check	Without IEC check	Explanation
TIME	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of conversion shows the duration in milliseconds.
	INT	-	-	No implicit conversion
	DINT	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of conversion shows the duration in milliseconds.
	REAL	-	-	No implicit conversion

Source	Target	With IEC check	Without IEC check	Explanation
	S5TIME	-	-	
	DT	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

TIME (IEC time) (Page 2257)
Overview of data type conversion (Page 2417)
Explicit conversion of TIME (Page 2440)
Setting and disabling the IEC check (Page 2419)

Implicit conversion of S5TIME

Implicit conversion options

The following table shows the options for the implicit conversion of S5TIME data type:

Source	Target	With IEC check	Without IEC check	Explanation
S5TIME	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of conversion shows the duration in milliseconds.
	DWORD	-	-	No implicit conversion
	INT	-	-	
	DINT	-	-	
	REAL	-	-	
	TIME	-	-	
	DT	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	-	-	
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

S5TIME (duration) (Page 2255)
Overview of data type conversion (Page 2417)
Explicit conversion of S5TIME (Page 2441)
Setting and disabling the IEC check (Page 2419)

Date and time

Implicit conversion of DATE

Implicit conversion options

The following table shows the options for the implicit conversion of DATE data type:

Source	Target	With IEC check	Without IEC check	Explanation
DATE	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion corresponds to the number of days since 01/01/1990.
	DWORD	-	-	No implicit conversion
	INT	-	-	
	DINT	-	-	
	REAL	-	-	
	TIME	-	-	
	S5TIME	-	-	
	DT	-	-	
	TOD	-	-	
	STRING	-	-	
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

DATE (Page 2258)
Overview of data type conversion (Page 2417)
Setting and disabling the IEC check (Page 2419)

Implicit conversion of TOD

Implicit conversion options

The following table shows the options for the implicit conversion of TOD data type:

Source	Target	With IEC check	Without IEC check	Explanation
TOD	BOOL	-	-	No implicit conversion
	BYTE	-	-	
	WORD	-	-	
	DWORD	-	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of the conversion corresponds to the number of milliseconds since the start of day (0:00).
	INT	-	-	No implicit conversion
	DINT	-	-	
	REAL	-	-	
	TIME	-	-	
	S5TIME	-	-	
	DT	-	-	
	DATE	-	-	
	STRING	-	-	
	CHAR	-	-	
x: Conversion possible -: Conversion not possible				

See also

TIME_OF_DAY (TOD) (Page 2259)
Overview of data type conversion (Page 2417)
Setting and disabling the IEC check (Page 2419)

Implicit conversion of DT

Implicit conversion options

The DT data type cannot be implicitly converted.

See also

DATE_AND_TIME (date and time of day) (Page 2260)
Overview of data type conversion (Page 2417)

Explicit conversion of DT (Page 2444)
Setting and disabling the IEC check (Page 2419)

Character strings

Implicit conversion of CHAR

Implicit conversion options

The following table shows the options for the implicit conversion of CHAR data type:

Source	Target	With IEC check	Without IEC check	Explanation
CHAR	BOOL	-	-	No implicit conversion
	BYTE	-	X	The bit pattern of the source value is transferred unchanged to the destination data type.
	WORD	-	-	No implicit conversion
	DWORD	-	-	
	INT	-	-	
	DINT	-	-	
	REAL	-	-	
	TIME	-	-	
	S5TIME	-	-	
	DT	-	-	
	TOD	-	-	
	DATE	-	-	
	STRING	X	X	The bit pattern of the source value is transferred unchanged to the destination data type.
x: Conversion possible -: Conversion not possible				

See also

CHAR (character) (Page 2263)
Overview of data type conversion (Page 2417)
Explicit conversion of CHAR (Page 2445)
Setting and disabling the IEC check (Page 2419)

Implicit conversion of STRING

Implicit conversion options

The STRING data type cannot be implicitly converted.

See also

STRING (Page 2264)
Overview of data type conversion (Page 2417)
Explicit conversion of STRING (Page 2446)
Setting and disabling the IEC check (Page 2419)

Explicit conversion

Binary numbers

Explicit conversion of BOOL

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the BOOL data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
BOOL	BYTE	X	Only the LSB (Least Significant Bit) is set in the destination data type. The enable output ENO is always "1".	CONVERT
	WORD	X		
	DWORD	X		
	INT	X		BOOL_TO_INT
	DINT	X		BOOL_TO_DINT
	REAL	-	No explicit conversion	-
	TIME	-		
	S5TIME	-		
	DT	-		
	TOD	-		
	DATE	-		
	STRING	-		
	CHAR	-		
x: Conversion possible - : Conversion not possible				

See also

Implicit conversion (Page 2419)
Overview of data type conversion (Page 2417)
BOOL (bit) (Page 2240)

Bit strings

Explicit conversion of BYTE

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the BYTE data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
BYTE	BOOL	X	The following possibilities can occur: • If the source is "0", the target data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the target data type is also "1" and the enable output ENO "1". • If bits are not equal to LSB in the source value, the target data type is set according to LSB and the enable output ENO is "0".	CONVERT
	WORD	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type. The remaining bits are set to "0".	CONVERT
	DWORD	X		
	INT	X		
	DINT	X		
	REAL	-	No explicit conversion	-
	TIME	-		
	S5TIME	-		
	DT	-		
	TOD	-		
	DATE	-		
	STRING	-		

Source	Target	Conversion	Explanation	Mnemonics of the instruction
	CHAR	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type. The remaining bits are set to "0".	CONVERT
x: Conversion possible - : Conversion not possible				

See also

BYTE (byte) (Page 2240)

Overview of data type conversion (Page 2417)

Implicit conversion of BYTE (Page 2421)

Explicit conversion of WORD

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the WORD data type:

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
WORD	BOOL	X	The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bit is not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	CONVERT
	BYTE	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. If the permitted value range of the destination data type is violated, the enable output ENO is set to "0". The result of the conversion is invalid in such a case.	
	DWORD	X		
	INT	X		
	DINT	X		
	REAL	-	No explicit conversion	-
	TIME	-		

Source	Destination	Conversion	Explanation	Mnemonics of the instruction
	S5TIME	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	MOVE
	DT	-	No explicit conversion	-
	TOD	-		
	DATE	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	MOVE
	STRING	-	No explicit conversion	-
	CHAR	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	WORD_TO_CHAR
	BLOCK_DB	X	The bit pattern of WORD is interpreted as a data block number.	WORD_TO_BLOCK_DB
WORD_BCD[1)]	INT	X	If the permitted value range of the destination data type is violated or if there is an invalid tetrad in the A - F range, the enable output ENO is set to "0". The result of the conversion is invalid in such a case.	WORD_BCD_TO_INT
BCD[1)]	INT	X		BCD_TO_INT

x: Conversion possible

- : Conversion not possible

[1)] The value to be converted has data type WORD and is accepted as a BCD-coded value between -999 and +999. The result is available after conversion as an integer (in binary notation) of the type INT.

See also

WORD (Page 2241)

Overview of data type conversion (Page 2417)

Implicit conversion of WORD (Page 2422)

Explicit conversion of DWORD

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the DWORD data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
DWORD	BOOL	X	The following possibilities can occur: • If the source is "0", the target data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the target data type is also "1" and the enable output ENO "1". • If bit is not equal to LSB in the source value, the target data type is set according to LSB and the enable output ENO is "0".	CONVERT
	BYTE	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.	
	WORD	X		

Source	Target	Conversion	Explanation	Mnemonics of the instruction

	INT	X	ENO = TRUE
			#int1 := DWORD_TO_INT(16#FFFF_FFFF); // -1 bis
			#int1 := DWORD_TO_INT(16#FFFF_8000); // -32768
			#int1 := DWORD_TO_INT(16#0); // 0 bis
			#int1 := DWORD_TO_INT(16#0000_7FFF); // 32767
			
			ENO = FALSE
			#int1 := DWORD_TO_INT(16#FFFF_7FFF); // -32769
			#int1 := DWORD_TO_INT(16#8000_0000); // -2147483648
			#int1 := DWORD_TO_INT(16#8000); // 32768 bis
			#int1 := DWORD_TO_INT(16#7FFF_FFFF); // 2147483647	

	DINT	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.	MOVE
	REAL	X		
	TIME	X		
	S5TIME	-	No explicit conversion	-
	DT	-		
	TOD	X	The bit pattern of the source value is transferred unchanged right-justified to the target data type.	MOVE
	DATE	-	No explicit conversion	-
	STRING	-		
	CHAR	-		
DWORD_BCD[1)]	DINT	X	If the permitted value range of the target data type is violated or if there is an invalid tetrad in the A - F range, the enable output ENO is set to "0". The result of the conversion is invalid in such a case.	DWORD_BCD_TO_DINT
BCD[1)]	DINT	X		BCD_TO_DINT

x: Conversion possible

- : Conversion not possible

[1)] The value to be converted has data type DWORD and is accepted as a BCD-coded value between -9999999 and +9999999. The result is available after conversion as an integer (in binary notation) of the type DINT.

See also

DWORD (Page 2242)
Overview of data type conversion (Page 2417)
Implicit conversion of DWORD (Page 2422)

Integers

Explicit conversion of INT

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the INT data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
INT	BOOL	X	The value is first converted to WORD and then to BOOL. The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bit is not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	INT_TO_BOOL
	BYTE	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. The enable output ENO is set to "0" if a negative value is converted to an unsigned destination data type or if overflow occurs.	CONVERT
	WORD	X		MOVE
	DWORD	X		
	DINT	X		CONVERT, ITD
	REAL	X	The value is converted into the format of the destination data type (the value "1", for example, is converted with the instruction "Convert value" (CONVERT) into the value "1.0").	CONVERT, SCALE, ITR
	TIME	-	No explicit conversion	-
	S5TIME	-		
	DT	-		
	TOD	-		
	DATE	-		

Source	Target	Conversion	Explanation	Mnemonics of the instruction
	STRING	X	The value is converted into a character string. The character string is shown preceded by a sign. If the permitted length of the character string is violated, the enable output ENO is set to "0".	S_CONV, CONVERT
	CHAR	X	The bit pattern of the source value is transferred unchanged to the destination data type. With the conversion of negative values or with an overflow, the enable output ENO will be set to "0".	CONVERT
	BCD[1)]	X	If the permitted value range of the destination data type is violated or if there is an invalid tetrad in the A - F range, the enable output ENO is set to "0". The result of the conversion is invalid in such a case.	INT_TO_BCD
	BCD_WORD [1)]	X		INT_TO_BCD_WORD
x: Conversion possible - : Conversion not possible [1)] The value to be converted has type INT and is accepted as an integer with a value between -999 and +999. The result is available after conversion as a BCD-coded number of the type WORD.				

See also

INT (16-bit integers) (Page 2245)

Overview of data type conversion (Page 2417)

Implicit conversion of INT (Page 2423)

Explicit conversion of DINT

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the DINT data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
DINT	BOOL	X	The value is first converted to DWORD and then to BOOL. The following possibilities can occur: • If the source is "0", the destination data type is also "0" and the enable output ENO "1". • If only the LSB (Least Significant Bit) "1" is set in the source value, the destination data type is also "1" and the enable output ENO "1". • If bit is not equal to LSB in the source value, the destination data type is set according to LSB and the enable output ENO is "0".	DINT_TO_BOOL
	BYTE	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type. The enable output ENO is set to "0" if a negative value is converted to an unsigned destination data type or if overflow occurs.	CONVERT
	WORD	X		
	DWORD	X		MOVE
	INT	X		CONVERT
	REAL	X	The value is converted into the format of the destination data type (the value "1", for example, is converted with the instruction "Convert value" (CONVERT) into the value "1.0").	CONVERT, DTR, SCALE
	TIME	X	The bit pattern of the source value is transferred unchanged to the destination data type.	T_CONV, MOVE
	S5TIME	-	No explicit conversion	-
	DT	-		
	TOD	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	CONVERT
	DATE	X		
	STRING	X	The value is converted into a character string. The character string is shown preceded by a sign. If the permitted length of the character string is violated, the enable output ENO is set to "0".	S_CONV, CONVERT
	CHAR	X	The bit pattern of the source value is transferred unchanged to the destination data type. With the conversion of negative values or with an overflow, the enable output ENO will be set to "0".	DINT_TO_CHAR

Source	Target	Conversion	Explanation	Mnemonics of the instruction
	BCD[1)]	X	If the permitted value range of the destination data type is violated or if there is an invalid tetrad in the A - F range, the enable output ENO is set to "0". The result of the conversion is invalid in such a case.	DINT_TO_BCD
	BCD_DWORD[1)]	X		DINT_TO_BCD_DWORD
x: Conversion possible - : Conversion not possible				

See also

DINT (32-bit integers) (Page 2247)

Overview of data type conversion (Page 2417)

Implicit conversion (Page 2419)

Floating-point numbers

Explicit conversion of REAL

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the REAL data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
REAL	BOOL	-	No explicit conversion	-
	BYTE	-		
	WORD	-		
	DWORD	X	The bit pattern of the source value is transferred unchanged to the destination data type.	CONVERT, MOVE
	INT	X	The value is converted into the destination data type. The result of conversion depends on the instruction employed. The enable output ENO is set to "0" if the admissible range of values of the destination data type is exceeded during conversion or the value to be converted has an invalid floating-point number.	CONVERT, ROUND, RND, CEIL, RND+, FLOOR, RND-, TRUNC, UNSCALE
	DINT	X		
	TIME	-	No explicit conversion	-
	S5TIME	-		
	DT	-		
	TOD	-		
	DATE	-		

Source	Target	Conversion	Explanation	Mnemonics of the instruction
	STRING	X	The value is converted into a character string. The enable output ENO is set to "0" if the character string length is exceeded or the value to be converted has an invalid floating-point number.	S_CONV, CONVERT
	CHAR	-	No explicit conversion	-
x: Conversion possible - : Conversion not possible				

See also

REAL (Page 2252)

Overview of data type conversion (Page 2417)

Implicit conversion (Page 2419)

Timer operations

Explicit conversion of TIME

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the TIME data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
TIME	BOOL	-	No explicit conversion	-
	BYTE	-		
	WORD	-		
	DWORD	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of conversion shows the duration in milliseconds.	CONVERT
	INT	-	No explicit conversion	-
	DINT	X	The bit pattern of the source value is transferred unchanged to the destination data type. The result of conversion shows the duration in milliseconds.	T_CONV, CONVERT
	REAL	-	No explicit conversion	-
	S5TIME	X	The value is converted to the destination data type format. The enable output ENO is set to "0" in the event of overflow.	T_CONV, CONVERT

Source	Target	Conversion	Explanation	Mnemonics of the instruction
	DT	-	No explicit conversion	-
	TOD	-		
	DATE	-		
	STRING	-		
	CHAR	-		
x: Conversion possible - : Conversion not possible				

See also

TIME (IEC time) (Page 2257)

Overview of data type conversion (Page 2417)

Implicit conversion of TIME (Page 2425)

Explicit conversion of S5TIME

Options for explicit conversion in LAD, FBD, STL and GRAPH

The following table shows the options and instructions for the explicit conversion of the S5TIME data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
S5TIME	BOOL	-	No explicit conversion	-
	BYTE	-		
	WORD	X	The value is converted to the destination data type format.	S5TIME_TO_WORD
	DWORD	-	No explicit conversion	-
	INT	-		
	DINT	-		
	REAL	-		
	TIME	X	The value is converted to the destination data type format.	T_CONV, CONVERT
	DT	-	No explicit conversion	-
	TOD	-		
	DATE	-		
	STRING	-		
	CHAR	-		
x: Conversion possible - : Conversion not possible				

See also

S5TIME (duration) (Page 2255)
Overview of data type conversion (Page 2417)
Implicit conversion of S5TIME (Page 2426)

Date and time

Explicit conversion of DATE

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the DATE data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
DATE	BOOL	-	No explicit conversion	-
	BYTE	-		
	WORD	X	The value is converted to the destination data type format.	DATE_TO_WORD
	DWORD	-	No explicit conversion	-
	INT	X	The value is converted to the destination data type format.	DATE_TO_INT
	DINT	X		DATE_TO_DINT
	REAL	-	No explicit conversion	-
	TIME	-		
	S5TIME	-		
	DT	-		
	TOD	-		
	STRING	-		
	CHAR	-		
x: Conversion possible - : Conversion not possible				

See also

DATE (Page 2258)
Overview of data type conversion (Page 2417)
Implicit conversion of DATE (Page 2427)

Explicit conversion of TOD

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the TOD data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
TOD	BOOL	-	No explicit conversion	-
	BYTE	-		
	WORD	-		
	DWORD	X	The value is converted to the destination data type format.	TOD_TO_DWORD
	INT	-	No explicit conversion	-
	DINT	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	TOD_TO_DINT
	REAL	-	No explicit conversion	-
	TIME	X	The value is converted to the destination data type format.	TOD_TO_TIME
	S5TIME	-	No explicit conversion	-
	DT	-		
	DATE	-		
	STRING	-		
	CHAR	-		
x: Conversion possible - : Conversion not possible				

See also

TIME_OF_DAY (TOD) (Page 2259)

Overview of data type conversion (Page 2417)

Implicit conversion of TOD (Page 2428)

Explicit conversion of DT

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the DT data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
DT	BOOL	-	No explicit conversion	-
	BYTE	-		
	WORD	-		
	DWORD	-		
	INT	-		
	DINT	-		
	REAL	-		
	TIME	-		
	S5TIME	-		
	TOD	X	During conversion, times are extracted from the DTL format and transferred to the destination data type.	T_CONV, CONVERT
	DATE	X	During the conversion, the date information is extracted from the DTL format and transferred to the destination data type.	
	STRING	-	No explicit conversion	-
	CHAR	-		
x: Conversion possible - : Conversion not possible				

See also

DATE_AND_TIME (date and time of day) (Page 2260)

Overview of data type conversion (Page 2417)

String

Explicit conversion of CHAR

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the CHAR data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
CHAR	BOOL	-	No explicit conversion	-
	BYTE	X	The bit pattern of the source value is transferred unchanged right-justified to the destination data type.	CONVERT
	WORD	X		
	DWORD	X		
	INT	X		
	DINT	X		
	REAL	-	No explicit conversion	-
	TIME	-		
	S5TIME	-		
	DT	-		
	TOD	-		
	DATE	-		
	STRING	X	The value is converted into the first character in the character string (STRING). The length "1" is set after conversion if the character string length is not defined. If the character string length is defined, it remains unchanged following conversion.	S_CONV, CONVERT
x: Conversion possible - : Conversion not possible				

See also

CHAR (character) (Page 2263)

Overview of data type conversion (Page 2417)

Implicit conversion of CHAR (Page 2429)

Explicit conversion of STRING

Options for explicit conversion

The following table shows the options and instructions for the explicit conversion of the STRING data type:

Source	Target	Conversion	Explanation	Mnemonics of the instruction
STRING	BOOL	-	No explicit conversion	-
	BYTE	-		
	WORD	-		
	DWORD	-		
	INT	X	Conversion begins with the first character in the character string (STRING) and stops at the end of the string or at the first inadmissible character. The following characters are admissible for conversion: • Digit • Sign • Dot The first character in the character string may be a sign (+, -) or a digit. Leading spaces are ignored. The dot is used as separation for the conversion of floating-point numbers. The exponential notation "e" or "E" is not permitted. The comma as thousand separator to the left of the decimal point is permitted but is ignored. If the layout of the character string is invalid for the conversion or an overflow occurs, then the enable output ENO will be set to "0".	S_CONV, CONVERT
	DINT	X		
	REAL	X		
	TIME	-	No explicit conversion	-
	S5TIME	-		
	DT	-		
	TOD	-		
	DATE	-		
	CHAR	X	The first character in the character string (STRING) is transferred to the target data type. The value "0" is written to the target data type if the character string is empty.	S_CONV, CONVERT
x: Conversion possible - : Conversion not possible				

See also

STRING (Page 2264)
Overview of data type conversion (Page 2417)
Implicit conversion (Page 2419)

Additional conversion functions

Additional explicit conversion functions

Additional options for explicit conversion in SCL

The following table shows the additional options and operations for explicit conversion in SCL:

Source	Target	Explanation	Mnemonics of the instruction
WORD	BLOCK_DB	The bit pattern of WORD is interpreted as a data block number.	WORD_TO_BLOCK_DB
BLOCK_DB	WORD	The data block number is interpreted as a bit pattern of WORD.	BLOCK_DB_TO_WORD

9.1.1.6 Program flow control

EN/ENO mechanism

Basics of the EN/ENO mechanism

Introduction

Runtime errors that require a program abort can occur during the processing of instructions. You can use the EN/ENO mechanism to avoid such program aborts. This mechanism can be used at two levels:

- EN/ENO mechanism for individual instructions
- EN/ENO mechanism for block calls

EN/ENO mechanism for instructions in LAD/FBD

In LAD and FBD, certain instructions have an enable input EN and an enable output ENO.

You can use the enable input EN to make the execution of the instruction dependent on conditions. The instructions are only executed if the signal state is "1" at the enable input EN.

You can use the enable output ENO to query runtime errors in instructions and react to these.

The enable output ENO returns the signal state "1" if one of the following conditions applies:

- No error occurred during processing.

The enable output ENO returns signal state "0" if one of the following conditions applies:

- The EN input has signal state "0".
- An error occurred during processing.

The EN/ENO mechanism is used for the following basic instructions:

- Mathematical functions
- Move operations
- Conversion operations
- Word logic operations
- Shift + rotate

In LAD and FBD, you can switch the evaluation of the enable output ENO on and off by means of the shortcut menu specifically for certain instructions.

EN/ENO mechanism for block calls in LAD/FBD

All blocks that you call in LAD or FBD are provided with an enable input EN and an enable output ENO when called. This applies to all called blocks, regardless of the programming language in which they were created.

You can use the enable input EN to call the block depending on conditions. The block is only executed if the signal state is "1" at the enable input EN.

You can query the error status of the block with the enable output ENO. It has signal "1" as soon as the execution of the block starts. If you do not explicitly set the output ENO to "0" in the program code, it retains signal "1".
However, you can explicitly set it to "0" to return an error statement to the called block. In LAD or FBD, the output ENO is set with the instruction "RET: Return".

See also:

Example of the use of the EN/ENO mechanism in LAD (Page 2450)

Example of the use of the EN/ENO mechanism in FBD (Page 2450)

EN/ENO mechanism for STL

In STL, the EN/ENO mechanism is not required for individual instructions. This function is mapped by language-specific instruction sequences.

Blocks that you call from an STL block are not provided with the EN and ENO parameters. Regardless of the programming language in which they were created, you can transfer an error statement to the calling STL block using the BR bit of the status word.

In STL, you can evaluate the error status of the called block by linking the BR bit of the status word with the RLO. It has signal "1" as soon as the execution of the block starts. If you do not explicitly set it to "0" in the program code, it retains signal "1".
However, you can explicitly set it to "0" to return an error statement to the calling block. In STL, the error statement is set with the instructions "SAVE" or "JNB".

See also: Example of the simulation of the EN/ENO mechanism in STL (Page 2452)

EN/ENO mechanism in SCL

With SCL, the use of the EN/ENO mechanism for instructions is optional. You can activate it with the block property "Set ENO automatically". If the property is active, all blocks implicitly receive an error handling.

You can implement a conditional block call with the enable input EN. Use the enable input EN in the parameter list as a normal input parameter. If EN has signal "1" or when EN is not used, the block is called. If EN has signal "0", the block is not called.

Note

When you call functions in SCL, you cannot use the release mechanism via EN. Use an IF statement instead to call functions conditionally.

You can query the error status of the block with the enable output ENO. If the ENO has signal "1", the block was processed without errors. If the ENO has signal "0", an error occurred during processing. To query the state of the enable output, insert an additional output parameter with the name ENO in the parameter list during a block call.

See also: Example of the use of the EN/ENO mechanism in SCL (Page 2451)

ENO in GRAPH (S7-1500)

In GRAPH, you can use the ENO operand to evaluate whether an action has been completed successfully. You can select the "Set ENO automatically" option for this.

- Select the option in the program properties in order for it to be used as the default setting for new GRAPH blocks.
- Select the option in the block properties in order for it to be used for specific blocks.

If this option is selected, the ENO operand is displayed when testing with program status. This operand has the value "TRUE" if the action was successful and "FALSE" if the action failed.

The ENO operand can be used in the following cases:

- When blocks are called that have the enable output ENO.
- For instructions in which errors can occur, e.g., conversions or mathematical functions.

EN/ENO mechanism for memory and I/O access errors

You cannot evaluate memory and I/O access errors with the EN/ENO mechanism. You do this either with the global troubleshooting via OBs (S7-300/400 and S7-1200/1500) or local troubleshooting using the "GetError" instruction (S7-1200/1500 only). If a memory access error occurred for an instruction, you can evaluate the associated ENO.

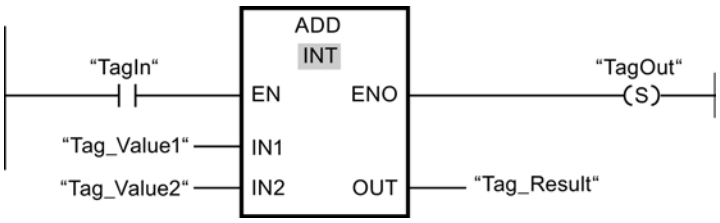
See also

Enabling and disabling the EN/ENO mechanism (Page 2600)

Example of the use of the EN/ENO mechanism in LAD

Description

The following figure shows an ADD instruction with EN and ENO protective circuit:



After the normally open contact, the RLO contains the previous result of logic operation:

- If "TagIn" signal is "0", the addition is not executed. EN and ENO both lead to the signal state "0".
- If "TagIn" signal is "1", EN is also "1" and the addition is executed. If no errors occur during the processing of the instruction, the output ENO also has the signal state "1" and the output ""TagOut"" is set.

See also

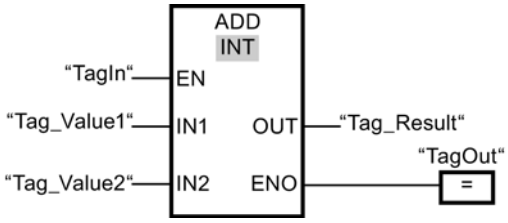
Basics of the EN/ENO mechanism (Page 2447)

ADD: Add (Page 3267)

Example of the use of the EN/ENO mechanism in FBD

Description

The following figure shows an ADD instruction with EN and ENO protective circuit:



- If "TagIn" signal is "1", EN is also "1" and the addition is executed. If no errors occur during the processing of the instruction, the output ENO also has the signal state "1" and the output ""TagOut"" is set.
- If "TagIn" signal is "0", the addition is not executed. EN and ENO both lead to the signal state "0".

See also

Basics of the EN/ENO mechanism (Page 2447)

Example of the use of the EN/ENO mechanism in SCL

Example of the EN/ENO mechanism for basic instructions

To use the EN/ENO mechanism for instructions in SCL, you have to activate the block property "Set ENO automatically". The following example shows the use of the enable output ENO for the "a/b" instruction.

SCL

```
"MyoutputREAL" :=a/b;
IF ENO
    THEN "MyOutputBool":=1;
    ELSE "MyOutputBool":=0;
END_IF;
```

If the "a/b" instruction is executed error-free, MyOutputBool has signal "1".

Example of the use of the EN/ENO mechanism in block calls

The following example shows the use of the enable output ENO for a block call.

SCL

```
"MyDB"."MyFB"(EN:="MyTag1">"MyTag2",
              in1:="MyInputBool1",
              in2:="MyInputBool1",
              EN0=>"MyOutputBool");
```

If MyTag1 is not greater than MyTag2 the block call is not processed. EN and ENO both lead to the signal state "0".

If MyTag1 is greater than MyTag2, EN has signal "1" and the block call is executed.

If all instructions within MyFB are executed error-free, MyOutputBool has signal "1".

See also

Basics of the EN/ENO mechanism (Page 2447)

Example of the simulation of the EN/ENO mechanism in STL

Description

The following example shows an program section for adding values with EN and ENO connected:

STL	Description
A"Tag_Input_1"	// Query whether the signal state of the operand is "1" and AND with current RLO
JNBMyLABEL	// Evaluation of the EN input
	// If RLO="0" jump to jump label "MyLABEL" and save the current RLO in the BR
	// Execute next instruction if RLO="1"
L"Tag_Input_2"	// Load first value of addition
L"Tag_Input_3"	// Load second value of addition
+I	// Add values
T "Tag_Result"	// Transfer sum to the operand "Tag_Result"
AN OV	// Query if errors occurred
SAVE	// Transfer signal state of the RLO to the BR bit
CLR	// Reset RLO to "0"
MyLABEL: U BR	// Jump label "MyLABEL"
	// Query BR and AND with RLO
= "Tag_Output"	// Assign signal state of the RLO to the operand "Tag_Output"

The query of the operand "U" Tag_Input_1"" provides the result of the preceding logic operation (RLO). The instruction "Jump at RLO = 0 and save RLO (SPBNB)" saves the RLO to the BR. The instruction "Jump if RLO = 0 and save RL0" also evaluates the RLO and executes one of the following actions depending on the evaluation:

- If the RLO is "0", the processing of the program is continued at the jump label "MyLABEL" with the query of the BR. The addition is not executed. Assign the current RLO to the operand "Tag_Output".
- If the RLO is "1", the addition is executed. A query of the overflow bit (OV) shows if an error occurred during the addition. The query result is saved in the BR. The operation "CLR" resets the RLO to "0". The BR is then queried for "1" and AND'd with the current RLO. The result is assigned to the operand "Tag_Output". The signal state of the BR and of the operand "Tag_Output" shows if the addition was carried out with any error

See also

Basics of the EN/ENO mechanism (Page 2447)

Status word

Basic information on the status word

Description

The status word combines status bits that the CPU uses to control the binary logic operations and sets for digital processing. You can query the status bits and influence them individually.

The following table shows the arrangement of the status bits in the status word:

	Status word									
Bit number	15-9	8	7	6	5	4	3	2	1	0
Content	0	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC

The status bits /FC, RLO, STA, OR and BR are binary codes which the CPU uses for binary instructions. The status bits OS, OV, CC 0 and CC 1 are digital codes which generally express math function results.

/FC (first check)

The /FC bit signal state controls a logic operation string.

A logic operation string begins with an /FC bit signal state of "0" and a binary scan instruction (first scan). The first scan is the first binary bit logic operation or scan in a network. This sets the /FC bit to "1". The logic operation string ends with a binary value assignment (e.g. the "Set" instruction), a conditional jump or a change of block. This sets the /FC bit to "0".

RLO (result of logic operation)

The status bit RLO is the buffer in binary logic operations.

The CPU transfers the scan results of the first scan to the RLO. The scan results for each subsequent scan are combined with the RLO saved and the result obtained is then saved in the RLO.

You can set, reset or negate the RLO or save it in the binary result (BR) using the corresponding instructions.

The RLO is used to control the memory, timer and counter instructions and execute certain jump instructions.

STA (status)

The STA status bit is equal to the signal state of the binary operand queried.

The status of a logic operation that reads memory is always the same as the value of the addressed bit. The status of a logic operation that writes to memory (e.g. the "Set" or "Reset" instructions) always has the same value as the bit written by the instruction. If the instruction does not write to memory, the status has the same value as the addressed bit. The status bit has no significance for logic operation instructions that do not access the memory. These

instructions set the status bit to "1". The status bit is not queried by instructions. It is only evaluated when displaying the online status of program variables.

OR (OR bit)

The OR status bit is used when an AND logic operation is executed before an OR logic operation.

The OR bit is set if the RLO of the AND logic operation is "1". This anticipates the result of the OR logic operation. Every other binary instruction resets the OR bit.

OS (Overflow, stored)

The OS status bit saves the set OV status bit.

When the CPU sets the OV status bit, it also sets the OS status bit. However, while the next properly executed instruction resets the OV bit, the OS status bit remains set. You are therefore able to query a number range overflow or, for example, the use of invalid floating-point numbers at a later stage in the current CPU cycle.

OV (Overflow)

The OV status bit shows a number range overflow or the use of invalid floating-point numbers.

The OV status bit can be influenced by math functions, conversion instructions, and floating-point number comparisons.

CC 0 and CC 1 (condition code bits)

The CC 0 and CC 1 status bits provide information on the results of the following instructions:

- Comparison instructions
- Math functions
- Word logic operations
- Shift and rotate instructions

BR (binary result)

The BR status bit is used to execute the EN/ENO mechanism for boxes or as a condition in certain jump instructions (STL). You can influence the BR status bit using certain instructions (e.g. SAVE).

Setting status bits

Setting the status bit for bit logic operations

Description

Binary instructions affect the OR, STA, RLO and /FC status bits.

Setting the status bits in LAD

The following table shows how the status bits are set in the LAD programming language:

Instruction		Status bit			
Mnemonics	Title	OR	STA	RLO	/FC
-| |-	Normally open contact	x	x	x	1
-| / |-	Normally closed contact	x	x	x	1
-|NOT|-	Invert RLO	-	1	x	-
-( )-	Assignment	0	x	-	0
-(R)-	Reset output	0	x	-	0
-(S)-	Set output	0	x	-	0
SR	Set/reset flip-flop	x	x	x	1
RS	Reset/set flip-flop	x	x	x	1
-|P|-	Scan operand for positive signal edge	0	1	x	1
-|N|-	Scan operand for negative signal edge	0	1	x	1
P_TRIG	Scan RLO for positive signal edge	0	x	x	1
N_TRIG	Scan RLO for negative signal edge	0	x	x	1
x: The instruction can write "1" or "0" 1: The instruction sets the status bit to "1". 0: The instruction resets the status bit to "0". -: The status bit is not influenced.					

Setting the status bits in FBD

The following table shows how the status bits are set in the FBD programming language:

Instruction		Status bit			
Mnemonics	Title	OR	STA	RLO	/FC
&	AND logic operation	x	x	x	1
<=1	OR logic operation	x	x	x	1
X	EXCLUSIVE OR logic operation	x	x	x	1
-|	Insert binary input	-	1	x	-
-o|	Invert RLO	-	1	x	-
-( )-	Assignment	0	x	-	0

Instruction		Status bit			
Mnemonics	Title	OR	STA	RLO	/FC
R	Reset output	0	x	-	0
S	Set output	0	x	-	0
SR	Set/reset flip-flop	x	x	x	1
RS	Reset/set flip-flop	x	x	x	1
P	Scan operand for positive signal edge	0	1	x	1
N	Scan operand for negative signal edge	0	1	x	1
P_TRIG	Scan RLO for positive signal edge	0	x	x	1
N_TRIG	Scan RLO for negative signal edge	0	x	x	1
x: The instruction can write "1" or "0" 1: The instruction sets the status bit to "1". 0: The instruction resets the status bit to "0". -: The status bit is not influenced.					

Setting the status bits in STL

The following table shows how the status bits are set in the STL programming language:

Instruction		Status bit			
Mnemonics	Title	OR	STA	RLO	/FC
A	AND logic operation	0	x	x	1
AN	Negated AND logic operation	0	x	x	1
O	OR logic operation	0	x	x	1
ON	Negated OR logic operation	0	x	x	1
X	EXCLUSIVE OR logic operation	0	x	x	1
XN	Negated EXCLUSIVE OR logic operation	0	x	x	1
O	OR logic operation from AND functions	0	1	-	0
A(	AND logic operation with branch	0	1	-	0
AN(	Negated AND logic operation with branch	0	1	-	0
O(	OR logic operation with branch	0	1	-	0
ON(	Negated OR logic operation with branch	0	1	-	0
X(	EXCLUSIVE OR logic operation with branch	0	1	-	0
XN(	Negated EXCLUSIVE OR logic operation with branch	0	1	-	0
=	Assignment	0	x	-	0
R	Reset output	0	x	-	0
S	Set output	0	x	-	0
NOT	Invert RLO	-	1	x	-
SET	Set RLO to 1	0	1	1	0
CLR	Reset RLO to 0	0	0	0	0
SAVE	Save RLO in BR bit	-	-	-	-
FN	Scan RLO for negative signal edge	0	x	x	1

Instruction		Status bit			
Mnemonics	Title	OR	STA	RLO	/FC
FP	Scan RLO for positive signal edge	0	x	x	1
x: The instruction can write "1" or "0" 1: The instruction sets the status bit to "1". 0: The instruction resets the status bit to "0". -: The status bit is not influenced.					

Setting status bits in timer and counter instructions

Description

The timer and counter instructions affect the OR, STA, RLO and /ER status bits.

Setting the status bits in LAD and FBD

The following table shows how the status bits are set in the LAD and FDB programming languages:

Instruction		Status bit				
Mnemonics	Title	OS	OR	STA	RLO	/FC
S_PULSE	Assign pulse timer parameters and start	-	x	x	x	1
S_PEXT	Assign extended pulse timer parameters and start	-	x	x	x	1
S_ODT	Assign on-delay timer parameters and start	-	x	x	x	1
S_ODTS	Assign retentive on-delay timer parameters and start	-	x	x	x	1
S_OFFDT	Assign off-delay timer parameters and start	-	x	x	x	1
SP	Start pulse timer	-	0	-	-	0
SE	Start extended pulse timer	-	0	-	-	0
SD	Start on-delay timer	-	0	-	-	0
SS	Start retentive on-delay timer	-	0	-	-	0
SF	Start off-delay timer	-	0	-	-	0
S_CUD	Assign parameters and count up / down	-	x	x	x	1
S_CU	Assign parameters and count up	-	x	x	x	1
S_CD	Assign parameters and count down	-	x	x	x	1
SC	Set counter value	-	0	x	-	0
CU	Count up	-	0	x	-	0
CD	Count down	-	0	-	-	0
x: The instruction can write "1" or "0" 1: The instruction sets the status bit to "1". 0: The instruction resets the status bit to "0". -: The status bit is not influenced.						

Setting the status bits in STL

The following table shows how the status bits are set in the STL programming language:

Instruction		Status bit				
Mnemonics	Title	OS	OR	STA	RLO	/FC
FR	Enable timer	-	0	-	-	0
R	Reset timer	-	0	-	-	0
SP	Start pulse timer	-	0	-	-	0
SE	Start extended pulse timer	-	0	-	-	0
SD	Start on-delay timer	-	0	-	-	0
SS	Start retentive on-delay timer	-	0	-	-	0
SF	Start off-delay timer	-	0	-	-	0
FR	Enable counters	-	0	-	-	0
R	Reset counter	-	0	-	-	0
S	Set counter	-	0	-	-	0
CU	Count up	-	0	-	-	0
CD	Count down	-	0	-	-	0
x: The instruction can write "1" or "0" 1: The instruction sets the status bit to "1". 0: The instruction resets the status bit to "0". -: The status bit is not influenced.						

Setting the status bits in instructions with integers

Description

Instructions with integers affect status bits CC 1, CC 0, OV and OS in the status word.

With a negative result, status bits CC 1 and CC 0 are set to "0". A positive result sets status bit CC 1 to "1" and status bit CC 0 to "0". Status bit CC 1 is set to "0" and status bit CC 0 is set to "1" if the result is negative.

In the case of a number range overflow, status bits OV and OS are set to "1". A divide by zero is indicated by status bits CC 1, CC 0, OV and OS having signal status "0".

The following tables show how the status bits are set when using integers:

Valid range	CC 1	CC 0	OV	OS
0 (zero)	0	0	0	*
16 bits: -32768 <=result < 0 (negative number) 32 bits: -2147483648 <=result < 0 (negative number)	0	1	0	*
16 bits: 32767 >= result > 0 (positive number) 32 bits: 2147483647 >= result > 0 (positive number)	1	0	0	*
* The OS bit is not influenced by the result of the operation.				

Invalid range	CC 1	CC 0	OV	OS
Underflow at addition 16 bits: Result = -65536 32 bits: Result = -4294967296	0	0	1	1
Underflow at multiplication 16 bits: Result < -32768 (negative number) 32 bits: Result < -2147483648 (negative number)	0	1	1	1
Overflow at addition, subtraction 16 bits: Result > 32767 (positive number) 32 bits: Result > 2147483647 (positive number)	0	1	1	1
Overflow at multiplication, division 16 bits: Result > 32767 (positive number) 32 bits: Result > 2147483647 (positive number)	1	0	1	1
Underflow at addition, subtraction 16 bits: Result < -32768 (negative number) 32 bits: Result < -2147483648 (negative number)	1	0	1	1
Division by 0	1	1	1	1

Setting the status bits in instructions with floating-point numbers

Description

Instructions with floating point numbers affect status bits CC 1, CC 0, OV and OS in the status word.

With a negative result, status bits CC 1 and CC 0 are set to "0". A positive result sets status bit CC 1 to "1" and status bit CC 0 to "0". Status bit CC 1 is set to "0" and status bit CC 0 is set to "1" if the result is negative.

In the case of a number range overflow, status bits OV and OS are set to "1". An invalid floating point number is indicated by status bits CC 1, CC 0, OV and OS having signal status "0".

The following tables show how the status bits are set when using floating point numbers:

Valid range	CC 1	CC 0	OV	OS
+0, -0 (zero)	0	0	0	*
-3.402823E+38 < result < -1.175494E-38 (negative number)	0	1	0	*
+1.175494E-38 < result < 3.402824E+38 (positive number)	1	0	0	*
* The OS bit is not influenced by the result of the operation.				

Invalid range	CC 1	CC 0	OV	OS
Underflow -1.175494E-38 < result < -1.401298E-45 (negative number)	0	0	1	1
Underflow +1.401298E-45 < result < +1.175494E-38 (positive number)	0	0	1	1
Overflow Result < -3.402823E+38 (negative number)	0	1	1	1
Overflow Result > 3.402823E+38 (positive number)	1	0	1	1
Not a valid floating-point number or invalid operation (input value outside of the valid value range)	1	1	1	1

Setting the status bit for comparison instructions

Description

Comparison statements affect status bits CC 1, CC 0, OV, OS in the status word.

Comparison statements set status bits CC 1 and CC 0 regardless of the comparison relationship.

When invalid floating point numbers are used, status bits CC 1, CC 0, OV and OS are set to "1".

The following table shows how the status bits are set with comparison statements:

Result	CC 1	CC 0	OV	OS
Equal to (ACCU2 = ACCU1)	0	0	0	*
Greater than (ACCU2 > ACCU1)	1	0	0	*
Smaller than (ACCU2 < ACCU1)	0	1	0	*
Invalid floating-point number	1	1	1	1
* The OS bit is not influenced by the result of the operation.				

Setting status bits in program control operation instructions

Description

The instructions for program control affect the OS, OR, STA, RLO and /FC status bits.

Setting the status bits in LAD and FBD

The following table shows how the status bits are set in the LAD and FDB programming languages:

Instruction		Status bit				
Mnemonics	Title	OS	OR	STA	RLO	/FC
JMP	Jump if RLO = 1	-	0	1	1	0
JMPN	Jump if RLO = 0	-	0	1	1	0
RET	Return	0	0	1	1	0
CALL	Call block	0	0	1	-	0
OPN	Open global data block	-	-	-	-	-
OPNI	Open instance data block	-	-	-	-	-
MCR<	Open MCR ranges	-	0	1	-	0
MCR>	Close MCR ranges	-	0	1	-	0
MCRA	Enable MCR range	-	-	-	-	-
MCRD	Disable MCR range	-	-	-	-	-
x: The instruction can write "1" or "0" 1: The instruction sets the status bit to "1". 0: The instruction resets the status bit to "0". -: The status bit is not influenced.						

Setting the status bits in STL

The following table shows how the status bits are set in the STL programming language:

Instruction		Status bit				
Mnemonics	Title	OS	OR	STA	RLO	/FC
JU	Unconditional jump	-	-	-	-	-
JC	Jump if RLO = 1	-	0	1	1	0
JCN	Jump if RLO = 0	-	0	1	1	0
JCB	Jump if RLO = 1 and save RLO	-	0	1	1	0
JNB	Jump if RLO = 0 and save RLO	-	0	1	1	0
JBI	Jump if BR = 1	-	0	1	-	0
JNBI	Jump if BR = 1	-	0	1	-	0
JO	Jump if OV = 1	-	-	-	-	-
JOS	Jump if OS = 1	0	-	-	-	-
JZ	Jump if the result is zero	-	-	-	-	-
JN	Jump if the result is not zero	-	-	-	-	-
JP	Jump if the result is greater than zero	-	-	-	-	-
JM	Jump if the result is less than zero	-	-	-	-	-
JPZ	Jump if the result is greater than or equal to zero	-	-	-	-	-
JMZ	Jump if the result is less than or equal to zero	-	-	-	-	-
JUO	Jump if the result is invalid	-	-	-	-	-

Instruction		Status bit				
Mnemonics	Title	OS	OR	STA	RLO	/FC
JL	Defining the jump list	-	-	-	-	-
LOOP	Loop	-	-	-	-	-
OPN	Open global data block	-	-	-	-	-
OPN_DI	Open instance data block	-	-	-	-	-
CALL	Call block	0	0	1	-	0
CC	Conditional block call	0	0	1	1	0
UC	Unconditional block call	0	0	1	-	0
BE	Block end	0	0	1	-	0
BEC	Conditional block end	x	0	1	1	0
BEU	Conditional block end	0	0	1	-	0
MCR(	Open MCR ranges	-	0	1	-	0
)MCR	Close MCR ranges	-	0	1	-	0
MCRA	Enable MCR range	-	-	-	-	-
MCRD	Disable MCR range	-	-	-	-	-
x: The instruction can write "1" or "0" 1: The instruction sets the status bit to "1". 0: The instruction resets the status bit to "0". -: The status bit is not influenced.						

Setting the status bit for word logic operations

Description

Word logic instructions affect status bits CC 1, CC 0, and OV in the status word.

If all the bits in the result of a word logic instruction are "0", status bit CC 1 is reset to "0". If at least one bit in the result is "1", status bit CC 1 is set to "1".

The following table shows how the status bits are set with word logic instructions:

Result	CC 1	CC 0	OV
Zero	0	0	0
Not zero	1	0	0

Setting the status bits for shift and rotate instructions

Description

Shift and rotate instructions affect status bits CC 1, CC 0, and OV in the status word.

In the case of shift and rotate instructions, the signal status of the last shifted or rotated bit is transferred to status bit CC 1. Status bits CC 0 and OV are reset.

The following table shows the setting of the status bit for shift and rotate instructions:

Displaced/rotated bit	CC 1	CC 0	OV
"0"	0	0	0
"1"	1	0	0
For shift or rotate number 0	*	*	*
* The bit is not influenced by the result of the operation.			

Querying status bits in STL

Querying status bits with bit logic operation instructions

Description

You can use the following bit logic instructions to query the signal state of bits in the status word:

- A: AND logic operation
- AN: Negated AND logic operation
- O: OR logic operation
- ON: Negated OR logic operation
- X: EXCLUSIVE OR logic operation
- XN: Negated EXCLUSIVE OR logic operation

The following table shows the operands which can be used to scan the signal state of the bits in the status word:

Operand	Querying the status word
== 0	Scan for ((CC 0 = 0) AND (CC 1 = 0))
<> 0	Scan for CC 0 <>CC 1
> 0	Scan for ((CC 0 = 0) AND (CC 1 = 1))
< 0	Scan for ((CC 0 = 1) AND (CC 1 = 0))
>= 0	Scan for CC 0 = 0
<= 0	Scan for CC 1 = 0
UO	Scan for ((CC 1 = 1) AND (CC 0 = 1))
OV	Scan for OV = 1
OS	Scan for OS = 1
BR	Scan for BR = 1

See also

A: AND logic operation (Page 5903)

AN: Negated AND logic operation (Page 5905)

O: OR logic operation (Page 5906)

ON: Negated OR logic operation (Page 5908)
X: EXCLUSIVE OR logic operation (Page 5909)
XN: Negated EXCLUSIVE OR logic operation (Page 5911)
O: OR logic operation of AND functions (Page 5912)
A(: AND logic operation with branch (Page 5913)
AN(: Negated AND logic operation with branch (Page 5914)
O(: OR logic operation with branch (Page 5916)
ON(: Negated OR logic operation with branch (Page 5917)
X(: EXCLUSIVE OR logic operation with branch (Page 5918)
XN(: Negated EXCLUSIVE OR logic operation with branch (Page 5919)
): Close branch (Page 5920)
=: Assignment (Page 5921)
R: Reset (Page 5923)
S: Set (Page 5924)
NOT: Invert RLO (Page 5925)
SET: Set RLO to 1 (Page 5926)
CLR: Reset RLO to 0 (Page 5927)
SAVE: Save RLO in BR bit (Page 5928)
FN: Scan RLO for negative signal edge (Page 5929)
FP: Scan RLO for positive signal edge (Page 5930)

Querying status bits with jump instructions

Description

The following table shows the jump instruction evaluation of the status bits:

RLO	BR	CC 1	CC 0	OV	OS	Executed instruction mnemonics
1	-	-	-	-	-	JC, JCB
0	-	-	-	-	-	JCN, JNB
-	1	-	-	-	-	JBI
-	0	-	-	-	-	JNBI
-	-	0	0	-	-	JZ, JMZ, JPZ
-	-	1	0	-	-	JN, JP, JPZ
-	-	0	1	-	-	JN, JM, JMZ
-	-	1	1	-	-	JUO
-	-	-	-	1	-	JO
-	-	-	-	-	1	JOS

The following table shows the dependency between status bits CC1 and CC0 and the conditional jump instructions:

CC 1	CC 0	Result	Executed instruction mnemonics
0	0	= 0	JZ
1 or 0	0 or 1	<> 0	JN
1	0	> 0	JP
0	1	< 0	JM
0 or 1	0	>= 0	JPZ
0	0 or 1	<= 0	JMZ
1	1	UO (invalid)	JUO

See also

JU: Unconditional jump (Page 6032)

JC: Jump if RLO = 1 (Page 6034)

JCN: Jump if RLO = 0 (Page 6035)

JCB: Jump if RLO = 1 and save RLO (Page 6036)

JNB: Jump if RLO = 0 and save RLO (Page 6038)

JBI: Jump if BR = 1 (Page 6039)

JNBI: Jump if BR = 0 (Page 6040)

JO: Jump if OV = 1 (Page 6041)

JOS: Jump if OS = 1 (Page 6043)

JZ: Jump if the result is zero (Page 6044)

JN: Jump if the result is not zero (Page 6046)

JP: Jump if the result is greater than zero (Page 6048)

JM: Jump if the result is less than zero (Page 6050)

JPZ: Jump if the result is greater than or equal to zero (Page 6051)

JMZ: Jump if the result is less than or equal to zero (Page 6053)

JUO: Jump if the result is invalid (Page 6055)

JL: Define jump list (Page 6057)

LOOP: Loop (Page 6058)

Status word

Basic information on the status word

Description

The status word combines status bits that the CPU uses to control the binary logic operations and sets for digital processing. You can query the status bits and influence them individually.

The following table shows the arrangement of the status bits in the status word:

	Status word									
Bit number	15-9	8	7	6	5	4	3	2	1	0
Content	0	BR	CC 1	CC 0	OV	OS	0	0	0	0

The status bits OS, OV, CC 0 and CC 1 are digital codes which generally express math function results.

OS (Overflow, stored)

The OS status bit saves the set OV status bit.

When the CPU sets the OV status bit, it also sets the OS status bit. However, while the next properly executed instruction resets the OV bit, the OS status bit remains set. You are therefore able to query a number range overflow or, for example, the use of invalid floating-point numbers at a later stage in the current CPU block.

OV (Overflow)

The OV status bit shows a number range overflow or the use of invalid floating-point numbers.

The OV status bit can be influenced by math functions, conversion instructions, and floating-point number comparisons.

CC 0 and CC 1 (condition code bits)

The CC 0 and CC 1 status bits provide information on the results of the following instructions:

- Comparison instructions
- Math functions
- Word logic operations
- Shift and rotate instructions

BR (binary result)

The BR status bit is used to execute the EN/ENO mechanism for boxes or as a condition in certain jump instructions (STL). You can influence the BR status bit using certain instructions (e.g. SAVE).

RLO (result of logic operation)

Status bit RLO is the buffer for binary logic operations and is not a component of the status word.

The CPU transfers the scan results of the first scan to the RLO. The scan results for each subsequent scan are combined with the RLO saved and the result obtained is then saved in the RLO.

You can set or reset the RLO using the corresponding instructions.

The RLO is used to control the memory, timer and counter instructions and execute certain jump instructions.

Note

Since the status bits are only partly needed for internal processing, they are only displayed if they are available for further program execution.

Querying and setting status bits in STL

Querying status bits with bit logic operation instructions

Description

You can use the following bit logic instructions to query the signal state of bits in the status word:

- A: AND logic operation
- AN: Negated AND logic operation
- O: OR logic operation
- ON: Negated OR logic operation
- X: EXCLUSIVE OR logic operation
- XN: Negated EXCLUSIVE OR logic operation

The following table shows the operands which can be used to scan the signal state of the bits in the status word:

Operand	Querying the status word
== 0	Scan for ((CC 0 = 0) AND (CC 1 = 0))
<> 0	Scan for CC 0 <>CC 1
> 0	Scan for ((CC 0 = 0) AND (CC 1 = 1))
< 0	Scan for ((CC 0 = 1) AND (CC 1 = 0))
>= 0	Scan for CC 0 = 0
<= 0	Scan for CC 1 = 0
UO	Scan for ((CC 1 = 1) AND (CC 0 = 1))

Operand	Querying the status word
OV	Scan for OV = 1
OS	Scan for OS = 1
BR	Scan for BR = 1

See also

A: AND logic operation (Page 3899)
AN: Negated AND logic operation (Page 3900)
O: OR logic operation (Page 3902)
ON: Negated OR logic operation (Page 3903)
X: EXCLUSIVE OR logic operation (Page 3905)
XN: Negated EXCLUSIVE OR logic operation (Page 3906)
O: OR logic operation of AND functions (Page 3908)
A(: AND logic operation with branch (Page 3909)
AN(: Negated AND logic operation with branch (Page 3910)
O(: OR logic operation with branch (Page 3911)
ON(: Negated OR logic operation with branch (Page 3912)
X(: EXCLUSIVE OR logic operation with branch (Page 3913)
XN(: Negated EXCLUSIVE OR logic operation with branch (Page 3914)
): Close branch (Page 3915)
=: Assignment (Page 3917)
R: Reset (Page 3918)
S: Set (Page 3919)
NOT: Invert RLO (Page 3920)
SET: Set RLO to 1 (Page 3921)
CLR: Reset RLO to 0 (Page 3922)
SAVE: Save RLO in BR bit (Page 3923)
FN: Scan RLO for negative signal edge (Page 3924)
FP: Scan RLO for positive signal edge (Page 3926)

Querying status bits with jump instructions

Description

The following table shows the jump instruction evaluation of the status bits:

RLO	BR	CC 1	CC 0	OV	OS	Executed instruction mnemonics
1	-	-	-	-	-	JC, JCB
0	-	-	-	-	-	JCN, JNB
-	1	-	-	-	-	JBI
-	0	-	-	-	-	JNBI
-	-	0	0	-	-	JZ, JMZ, JPZ
-	-	1	0	-	-	JN, JP, JPZ
-	-	0	1	-	-	JN, JM, JMZ
-	-	1	1	-	-	JUO
-	-	-	-	1	-	JO
-	-	-	-	-	1	JOS

The following table shows the dependency between status bits CC 0 and CC 1 and the conditional jump instructions:

CC 1	CC 0	Result	Executed instruction mnemonics
0	0	= 0	JZ
1 or 0	0 or 1	<> 0	JN
1	0	> 0	JP
0	1	< 0	JM
0 or 1	0	>= 0	JPZ
0	0 or 1	<= 0	JMZ
1	1	UO (invalid)	JUO

See also

JU: Unconditional jump (Page 6032)

JC: Jump if RLO = 1 (Page 6034)

JCN: Jump if RLO = 0 (Page 6035)

JCB: Jump if RLO = 1 and save RLO (Page 6036)

JNB: Jump if RLO = 0 and save RLO (Page 6038)

JBI: Jump if BR = 1 (Page 6039)

JNBI: Jump if BR = 0 (Page 6040)

JO: Jump if OV = 1 (Page 6041)

JOS: Jump if OS = 1 (Page 6043)

JZ: Jump if the result is zero (Page 6044)

JN: Jump if the result is not zero (Page 6046)

JP: Jump if the result is greater than zero (Page 6048)

JM: Jump if the result is less than zero (Page 6050)

JPZ: Jump if the result is greater than or equal to zero (Page 6051)

JMZ: Jump if the result is less than or equal to zero (Page 6053)

JUO: Jump if the result is invalid (Page 6055)

JL: Define jump list (Page 6057)

LOOP: Loop (Page 6058)

Jump label (Page 4029)

JU: Unconditional jump (Page 4030)

JC: Jump if RLO = 1 (Page 4032)

JCN: Jump if RLO = 0 (Page 4033)

JCB: Jump if RLO = 1 and save RLO (Page 4034)

JNB: Jump if RLO = 0 and save RLO (Page 4036)

JBI: Jump if BR = 1 (Page 4037)

JNBI: Jump if BR = 0 (Page 4038)

JO: Jump if OV = 1 (Page 4040)

JOS: Jump if OS = 1 (Page 4041)

JZ: Jump if the result is zero (Page 4043)

JN: Jump if the result is not zero (Page 4044)

JP: Jump if the result is greater than zero (Page 4046)

JM: Jump if the result is less than zero (Page 4048)

JPZ: Jump if the result is greater than or equal to zero (Page 4050)

JMZ: Jump if the result is less than or equal to zero (Page 4052)

JUO: Jump if the result is invalid (Page 4053)

JL: Define jump list (Page 4055)

LOOP: Loop (Page 4057)

Setting the status bit for word logic operations

Description

Word logic instructions affect status bits CC 0 , CC 1, and OV in the status word. The status bits are only set if they are used in the program sequence.

If all the bits in the result of a word logic instruction are "0", status bit CC 1 is reset to "0". If at least one bit in the result is "1", status bit CC 1 is set to "1".

The following table shows how the status bits are set with word logic instructions:

Result	CC 1	CC 0	OV
Zero	0	0	0
Not zero	1	0	0

See also

AW: AND logic operation word by word (Page 4075)

OW: OR logic operation word by word (Page 4077)

XOW: EXCLUSIVE OR logic operation word by word (Page 4079)

AD: AND logic operation double word by double word (Page 4081)

OD: OR logic operation double word by double word (Page 4083)

XOD: EXCLUSIVE OR logic operation double word by double word (Page 4085)

Master control relay

Master control relay

Description of the MCR functionality

With conventional programmable controllers, the Master Control Relay (MCR) enables or disables the signal flow in one part of the controller.

Note

Refer to the information in the "Important notes on using MCR functions (Page 2472)" section

WARNING

To prevent personal injury or property damage, never use the MCR to replace a hard-wired mechanical master control relay that is being used as an emergency stop or safety switching device.

The MCR affects all instructions that write a value back to the memory. When MCR dependency is enabled these instructions respond regardless of any previous binary or digital logic operation.

The following instructions are available for the implementation of the MCR:

- MCRA: Enable MCR area
- MCR: Open MCR ranges
- MCR: Close MCR ranges
- MCRD: Disable MCR area

The MCR dependency for this area is enabled when the result of logic operation is "1" directly before the opening of an MCR area. If the RLO of the area is "0" when it is opened, the processing in this area is without MCR dependency.

Important notes on using MCR functions

CAUTION

Note the following for blocks in which the master control relay was enabled with the instruction "Enable MCR range" (MCRA):

- When the MCR is switched off, the value "0" is written by all assignments in program sections between the instructions "Open MCR range" (MCR( ) and "Close MCR range".
- The MCR is then switched off exactly at the point when the result of logic operation (RLO) was "0" before an "Open MCR range" (MCR( ) instruction.

DANGER

CPU in STOP or undefined runtime characteristics!

For address calculations and also writes to local data, the compiler accesses the temporary variables defined after VAR_TEMP. The following command sequences therefore change the AS to STOP or lead to undefined runtime characteristics (refer to following explanations).

Formal parameter access

- Access to components of complex FC parameters of the type STRUCT, PLC data type (UDT), ARRAY, STRING
- Access to components of complex FB parameters of the type STRUCT, PLC data type (UDT), ARRAY, STRING from the InOut area in a block with multiple instance capability.
- Access to parameters of a function block with multiple instance capability if its address is greater than 8180.0.
- Access in a function block with multiple instance capability to a parameter of the type BLOCK_DB opens DB0. Any subsequent data access changes the CPU to STOP. T 0, C 0, FC0, or FB0 are also always used for TIMER, COUNTER, BLOCK_FC, and BLOCK_FB.

Parameter transfer

In block calls in which parameters are transferred.

Remedy

Free the above commands from their dependence on the MCR:

1. Use the instruction "Disable MCR range" (MCRD) to disable the master control relay before the assignment or network in question.
2. Use the instruction "Enable MCR range" (MCRA) to enable the master control relay before the assignment or network in question.

9.1.2 Declaring PLC tags

9.1.2.1 Overview of PLC tag tables

Introduction

PLC tag tables contain the definitions of the PLC tags and symbolic constants that are valid throughout the CPU. A PLC tag table is created automatically for each CPU used in the project. You can create additional tag tables and use these to sort and group tags and constants.

In the project tree there is a "PLC tags" folder for each CPU of the project. The following tables are included:

- "All tags" table
- Standard tag table
- Optional: Other user-defined tag tables

All tags

The "All tags" table gives an overview of all PLC tags, user constants and system constants of the CPU. This table cannot be deleted or moved.

Standard tag table

There is one standard tag table for each CPU of the project. It cannot be deleted, renamed or moved. The default tag table contains PLC tags, user constants and system constants. You can declare all PLC tags in the default tag table, or create additional user-defined tag tables as you want.

User-defined tag tables

You can create multiple user-defined tag tables for each CPU to group tags according to your requirements. You can rename, gather into groups, or delete user-defined tag tables. User-defined tag tables can contain PLC tags and user constants.

See also

Structure of the PLC tag tables (Page 2474)

Using tags within the program (Page 2203)

Basics of constants (Page 2204)

9.1.2.2 Structure of the PLC tag tables

Introduction

Each PLC tag table contains a tab for tags and a tab for user constants. The default tag table and the "All tags" table also have a "System constants" tab.

Structure of the "PLC tags" tab

In the "Tags" tab you declare the global PLC tags that you require in the program. The following figure shows the tab structure. The number of columns shown may vary.

	Name	Data type	Address	Retain	Visible in HMI	Accessible from HMI	Comment
	Motor1	Bool	%Q3.1	☐	☑	☑	
	Motor2	Bool	%Q3.2	☐	☑	☑	
	Control	Bool	%I3.3	☐	☑	☑	

The following table shows the meaning of the individual columns. The number of columns shown may vary. You can show or hide the columns as required.

Column	Description
	Symbol you can click on to drag-and-drop a tag to a program for use as an operand.
Name	Unique name for the constants throughout the CPU.
Data type	Data type of the tags.
Address	Tag address.
Retain	Marks the tag as retentive. The values of retentive tags are retained even after the power supply is switched off.
Accessible from HMI	Shows whether HMI can access this tag during runtime.
Visible in HMI	Shows whether the tag is visible by default in the operand selection of HMI.
Monitor value	Current data value in the CPU. This column only appears if an online connection is established and you select the "Monitor all" button.
Tag table	Shows which tag table includes the tag declaration. This column is only available in the "All tags" table.
Comment	Comment to document the tags.

Structure of the "User constants" and "System constants" tabs

In the "User constants" you define symbolic constants that are valid throughout the CPU. The constants required by the system are shown in the "Systems constants" tab. System constants can be hardware IDs, for example, which can be used for identification of modules.

The following figure shows the structure of both tabs. The number of columns shown may vary.

	Name	Data type	Value	Comment
	Const_1	Bool	true	
	Const_2	Byte	12	
	Const_3	Bool	false	
	Const_4	Real	1.0	

The following table shows the meaning of the individual columns. You can show or hide the columns as required.

Column	Description
	Symbol you can click to move a tag into a network via a drag-and-drop operation for use as an operand.
Name	Unique name for the constants throughout the CPU.
Data type	Data type of the constants
Value	Value of the constants
Tag table	Shows which tag table includes the constant declaration. This column is only available in the "All tags" table.
Comment	Comments to document the tags.

See also

Hardware data types (Page 2283)
Using tags within the program (Page 2203)
Basics of constants (Page 2204)
Data types (Page 2236)
Overview of PLC tag tables (Page 2473)
Show and hide table columns (Page 2496)
Editing tables (Page 316)

9.1.2.3 Rules for PLC tags

Valid names of PLC tags

Permissible characters

The following rules apply to the use of names for PLC tags:

- Letters, numbers, special characters are permitted.
- Quotation marks are not permitted.

Unique tag names

The names of the PLC tags must be unique throughout the CPU, even if the tags are located in different tag tables of a CPU. A name that is already used for a block, another PLC tag or a constant within the CPU, cannot be used for a new PLC tag. The uniqueness check does not differentiate between use of small and capital letters.

If you enter an already assigned name another time, a sequential number is automatically appended to the second name entered. For example, if you enter the name "Motor" a second time, the second entry is changed to "Motor(1)".

Unique table names

The names of the PLC tag tables must also be unique throughout the CPU. A unique name is automatically suggested when user-defined PLC tag tables are being created.

See also

Using tags within the program (Page 2203)

Permissible addresses and data types of PLC tags (Page 2476)

Entering a PLC tag declaration (Page 2481)

Keywords (Page 2197)

Permissible addresses and data types of PLC tags

The addresses of PLC tags are made up of the particulars of the operand area and the address within this area.

The addresses must be unique throughout the CPU. If you enter an address that is already assigned to another tag, the address will be highlighted at both places in yellow and an error message will be issued.

Operand areas

The following table shows the possible operand areas. The available data types depend on the CPU you use:

Operand area		Description	Data type	Format	Address area:		
International mnemonics	German mnemonics				S7-1200	S7-300/400	S7-1500
I	E	Input bit	BOOL	I x.y E x.y	0.0..1023.7	0.0..65535.7	0.0..32767.7
I	E	Input (64-bit)	LWORD, LINT, ULINT, LTIME, LTOD, LDT, LREAL, PLC data type	I x.0 E x.0	-	-	0.0..32760.0
IB	EB	Input byte	BYTE, CHAR, SINT, USINT, PLC data type	IB x EB y	0..1023	0..65535	0..32767
IW	EW	Input word	WORD, INT, UINT, DATE, S5TIME, PLC data type	IW x EW y	0..1022	0..65534	0..32766
ID	ED	Input double word	DWORD, DINT, UDINT, REAL, TIME, TOD, PLC data type	ID x ED y	0..1020	0..65532	0..32764
Q	A	Output bit	BOOL	Q x.y A x.y	0.0..1023.7	0.0..65535.7	0.0..32767.7
Q	A	Output (64-bit)	LWORD, LINT, ULINT, LTIME, LTOD, LDT, LREAL, PLC data type	Q x.0 A x.0	-	-	0.0..32760.0
QB	AB	Output byte	BYTE, CHAR, SINT, USINT, PLC data type	QB x AB y	0..1023	0..65535	0..32767
QW	AW	Output word	WORD, INT, UINT, DATE, S5TIME, PLC data type	QW x AW y	0..1022	0..65534	0..32766

Operand area		Description	Data type	Format	Address area:		
International mnemonics	German mnemonics				S7-1200	S7-300/400	S7-1500
QD	AD	Output double word	DWORD, DINT, UDINT, REAL, TIME, TOD, PLC data type	QD x AD y	0..1020	0..65532	0..32764
M	M	Memory bit	BOOL	M x.y	0.0..8191.7	0.0..65535.7	0.0..16383.7
M	M	Bit memory (64-bit)	LREAL	M x.0	0.0..8184.0	-	0.0..16376.0
M	M	Bit memory (64-bit)	LWORD, LINT, ULINT, LTIME, LTOD, LDT	M x.0	-	-	0.0..16376.0
MB	MB	Memory byte	BYTE, CHAR, SINT, USINT	MB x	0..8191	0..65535	0..16383
MW	MW	Memory word	WORD, INT, UINT, DATE, S5TIME	MW x	0..8190	0..65534	0..16382
MD	MD	Memory double word	DWORD, DINT, UDINT, REAL, TIME, TOD	MD x	0..8188	0..65532	0..16380
T	T	Time function (for S7-300/400 only)	Timer	T n	-	0..65535	0..2047
C	Z	Counter function (for S7-300/400 only)	Counter	Z n C n	-	0..65535	0..2047

Addresses

The following table shows the possible addresses of tags:

Data type	Address	Example
BOOL	Tags with BOOL data type are addressed with a byte number and a bit number. The numbering of the bytes begins for each operand area at 0. The numbering of the bits goes from 0 to 7.	A 1.0
BYTE, CHAR, SINT, USINT	Tags with BYTE, CHAR, SINT, and USINT data type are addressed with a byte number.	MB 1

Data type	Address	Example
WORD, INT, UINT, DATE, S5TIME	Tags with WORD, INT, UINT, DATE, S5TIME data type consist of two bytes. They are addressed with the number of the lowest byte.	IW 1
DWORD, DINT, UDINT, REAL, TIME, TOD	Tags with DWORD, DINT, UDINT, REAL, TIME, TOD data type consist of four bytes. They are addressed with the number of the lowest byte.	AD 1
LWORD, LINT, ULINT, LTIME, LTOD, LDT, LREAL	Tags of data type LWORD, LINT, ULINT, LTIME, LTOD, LDT, and LREAL consist of eight bytes. They are addressed with it number 0 and the number of the lower byte.	I 1.0

Mnemonics used

The addresses that you enter in the PLC tag table are automatically adapted to the set mnemonics.

See also

Setting the mnemonics (Page 2533)

Using tags within the program (Page 2203)

Valid names of PLC tags (Page 2475)

Entering a PLC tag declaration (Page 2481)

Overview of the valid data types (Page 2236)

9.1.2.4 Creating and managing PLC tag tables

Creating a PLC tag table

You can created multiple user-defined PLC tag tables in a CPU. Each tag table must have have a unique name throughout the CPU.

Requirement

The project view is open.

Procedure

To created a new PLC tag table, follow these steps:

1. Open the "PLC tags" folder under the CPU in the project tree.
2. Double-click the "Add new tag table" entry.
   A new PLC tag table with the default name "TagTable_x" is created.

3. Select the PLC tag table in the project tree.
4. Select the "Rename" command in the shortcut menu.
5. Type in a name that is unique throughout the CPU.

Result

A new PLC tag table is created. You can now declare tag and constants in this table.

See also

Overview of PLC tag tables (Page 2473)

Structure of the PLC tag tables (Page 2474)

Declaring PLC tags (Page 2481)

Declaring global constants (Page 2488)

Editing PLC tags with external editors (Page 2496)

Grouping PLC tag tables

You can gather the user-defined tag tables of the CPU into groups. You cannot, however, move the standard tag table and the "All tags" table into a group.

Requirement

Multiple user-defined tag tables are contained in the "PLC tags" folder of the CPU.

Procedure

To gather multiple PLC tag tables into a group, follow these steps:

1. Select the "PLC tags" folder under the CPU in the project tree.
2. Select the "Insert > Group" menu command.
   A new group with the standard name "Group_x" is inserted.
3. Select the newly inserted group in the project tree.
4. Select the "Rename" command in the shortcut menu.
5. Assign the new group a unique name throughout the CPU.
6. Drag to the new group the tables you want to group together.

Result

The tag tables are gathered in the new group.

See also

Overview of PLC tag tables (Page 2473)
Structure of the PLC tag tables (Page 2474)
Declaring PLC tags (Page 2481)
Declaring global constants (Page 2488)

Opening the PLC tag table

Procedure

To open the PLC tag table in a CPU, proceed as follows:

1. Open the "PLC tags" folder under the CPU in the project tree.
2. Double-click the PLC tag table in the folder.
3. Select the desired tab in the upper corner.

Result

The PLC tag table associated with the CPU opens. You can declare the required tags and constants.

See also

Overview of PLC tag tables (Page 2473)
Structure of the PLC tag tables (Page 2474)
Declaring PLC tags (Page 2481)
Declaring global constants (Page 2488)

9.1.2.5 Declaring PLC tags

Entering a PLC tag declaration

Declaring tags in the PLC tag table

Requirements

The "Tags" tab of the PLC tag table is open.

Procedure

To define PLC tags, follow these steps:

1. Enter a tag name in the "Name" column.
2. Enter the required data type in the "Data type" column. You will be supported by autocompletion during input.
   An address corresponding to the data type is automatically appended.
3. Optional: Click on the arrow key in the "Address" column and enter an operand identifier, an operand type, an address and a bit number in the dialog which then opens.
4. Optional: Enter a comment in the "Comments" column.
5. Repeat steps 1 to 4 for all the tags you require.

See also: Permissible addresses and data types of PLC tags (Page 2476)

Syntax check

A syntax check is performed automatically after each entry, and any errors found are displayed in red. You do not have to correct these errors immediately - you can continue editing and make any corrections later. As long as the tag declaration contains syntax errors and the tag is used in the program, you will not be able to compile the program.

See also

Editing PLC tags with external editors (Page 2496)

Using autocompletion (Page 2540)

Valid names of PLC tags (Page 2475)

Declaring PLC tags in the program editor (Page 2482)

Editing PLC tag tables (Page 2493)

Structure of the PLC tag tables (Page 2474)

Editing tables (Page 316)

Declaring PLC tags in the program editor

Requirement

- The program editor is open.

Procedure

To declare operands as global PLC tags, follow these steps:

1. Insert an instruction in your program.
   The "<???>", "<??.?>" or "..." strings represent operand placeholders.
2. Replace an operand placeholder with the name of the PLC tag to be created.

3. Select the tag name.
   If you want to declare multiple PLC tags, select the names of all the tags to be declared.
4. Select the "Define tag" command in the shortcut menu.
   The "Define tag" dialog box opens. This dialog displays a declaration table in which the name of the tag is already entered.
5. Click the arrow key in the "Section" column and select one of the following entries:
   - Global Memory
   - Global Input
   - Global Output
6. In the other columns, enter the address, data type, and comments.
   See also: Permissible addresses and data types of PLC tags (Page 2476)
7. If the CPU contains multiple PLC tag tables, you can use an entry in the "PLC tag table" column to indicate in which table the tag is to be inserted. If you make no entry in the column, the new tag will be inserted in the default tag table.
8. Click the "Define" button to complete your entry.

Result

The tag declaration is written to the PLC tag table and is valid for all blocks in the CPU.

See also

Valid names of PLC tags (Page 2475)

Editing tables (Page 316)

Declaring tags in the PLC tag table (Page 2481)

Using autocompletion (Page 2540)

Setting the retentivity of PLC tags

Retentive behavior of PLC tags

Retentive PLC tags

Each CPU has a memory area whose content remains available even after the supply voltage has been switched off. This area is referred to as retentive memory area.

To avoid data loss during power failure, you can save specific PLC tags to this memory area. You specify the retain setting of PLC tags in the PLC tag table.

Depending on the CPU family, the retentive memory area can accommodate various type of PLC tags. The following table provides an overview of the options of the various CPUs:

CPU type	Retentive bit memories	Retentive SIMATIC timers	Retentive SIMATIC counters
S7-300/400 series	✓	-	-
S7-1200 series	✓	-	-
S7-1500 series	✓	✓	✓

See also

Setting the retentive behavior of PLC tags (Page 2484)

Setting the retentive behavior of PLC tags

Introduction

In the PLC tag table you can specify the width of the retentive memory area for PLC tags. All tags with addresses in this memory area are then designated as retentive. You can recognize the retentivity setting of a tag by the check mark set in the "Retain" column of the PLC tag table.

Requirement

The "PLC tags" tab of the PLC tag table is open.

Procedure

To define the width of the retentive memory area for PLC tags, follow these steps:

1. On the toolbar, click the "Retain" button.
   The "Retain memory" dialog will open.
2. Specify the width of the retentive memory area by entering the number of retentive bytes, timers or counters in the input field.
3. Click the "OK" button.

Result

The width if the retentive memory area is defined. In the "Retain" column of the tag table a check mark is automatically set for all tags that are located within the retentive memory area.

See also

Retentive behavior of PLC tags (Page 2483)

Editing tables (Page 316)

9.1.2.6 Grouping PLC tags for inputs and outputs in structures

Other useful information regarding structured PLC tags

Use of structured PLC tags (S7-1200 V4 and higher/S7-1500)

To make your program easier to view, you can group several input or output addresses in a higher-level PLC tag. The higher-level PLC tag represents a structure that contains several logically related inputs or outputs. When the block is called, you transfer the higher-level tag and thus need only an input or output parameter for all associated inputs or outputs.

Principle of operation

To create a structured PLC tag, you initially define a PLC data type (UDT). In it you declare the necessary data elements and specify their names and data types.

Then, you switch to the PLC tag table and specify the higher-level PLC tag there. As a data type for the tag, you select your PLC data types. The system now reserves a certain number of input or output addresses starting from the start address of the higher-level tag. The number of reserved addresses depends on the length of your PLC data type.

If you call a block that requires the reserved inputs or outputs for program execution, you transfer the higher-level tag as a block parameter.

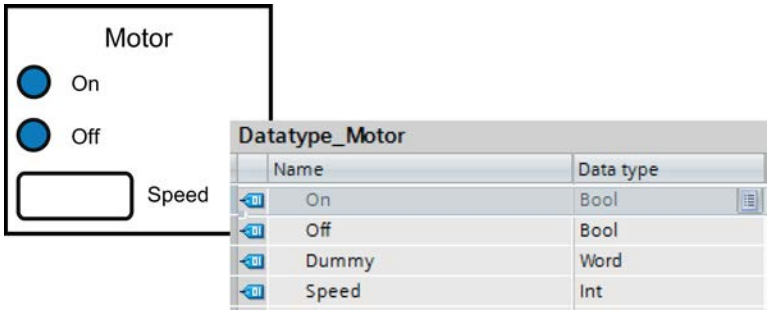
You can address the individual PLC tags like structure elements in the program code.

A detailed description of the individual handling steps can be found in the following chapters.

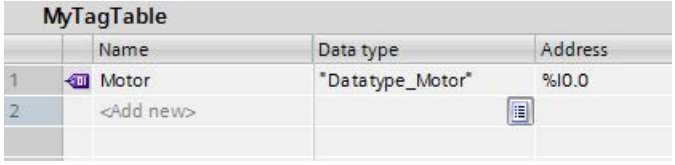
Application example

You can use structured PLC tags in order to group the inputs or outputs of a function module. The following figure shows the schematic representation of a motor: A component was created in the "Datatype_Motor" PLC data type for each of the three inputs.

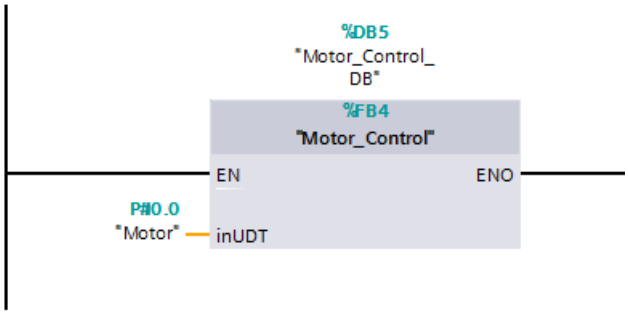
The memory areas of the declared tags must not overlap. In the example, you see that the "Speed" component has the data type "Integer" and therefore must start at a word address. For this reason, the first input word has been filled with the "Dummy" fill tag. This means that "Speed" is located on the second input word.



The following figure shows the higher-level "Motor" PLC tag that is based on the "Data type_Motor" data type. With the declaration of "Motor", the addresses IW0 and IW1 are reserved on the input module.



The following figure shows the transfer of the "Motor" PLC tag as an input parameter of the "Motor_Control" block.



You can address the individual components of the tag in the "Motor_Control" block.

Addressing	Description
"Motor"	Addressing the higher-level PLC tag.
"Motor".On	Addressing a component of a structured PLC tag.
"Motor".On:P	Addressing an I/O input or output (PI or PO).

Rules for use of structured PLC tags

Note the following rules when creating and using structured PLC tags.

- Structured PLC tags can be used in the "Inputs" and "Outputs" operand areas.
- Structured tags are not permitted in the bit memory address area.
- Structured PLC tags cannot be addressed from HMI.

Observe the following rules when creating the PLC data type that is to serve as the basis for a PLC tag:

- The memory areas of the individual elements must not overlap.
  See also: Permissible addresses and data types of PLC tags (Page 2476)
- Do not group inputs and outputs in one PLC data type but create different PLC data types for inputs and outputs.

- Do not group inputs or outputs from different modules in a PLC data type because it is not guaranteed that the process images of the modules are updated synchronously.
- In the lower-level PLC data types, all data types are permitted except for the "STRING" data type.

See also

Creating structured PLC tags (Page 2487)

Addressing global variables (Page 2209)

Data types (Page 2236)

Creating structured PLC tags

Rules

Note the following rules when creating structured PLC tags:

- Use separate PLC data types for the "Inputs" and "Outputs" operand areas.
- Structured tags are not permitted in the bit memory address area.
- Do not group inputs or outputs from different modules in a PLC data type because it is not guaranteed that the process images of the modules are updated synchronously.

Procedure

To create a structured PLC tag, follow these steps:

1. Double-click the "Add new data type" command in the "PLC data types" folder in the project tree.
   A new declaration table for creating a PLC data type will be created and opened.
2. Declare all the necessary components in the PLC type. All data types except the "STRING" data type are permitted.
3. Select the PLC data type in the project tree and select the "Compile > Software (only changes)" command from the shortcut menu.
   The PLC data type is compiled and can then be used in the PLC tag table.
   Even when you make changes to existing PLC data types, you must recompile the program. This updates all locations of use of the PLC data type.
4. Open a PLC tag table within the same CPU.
5. Declare a new tag or select an existing tag.
6. In the "Data type" column, select the PLC data type and assign it to the PLC tag.
   The PLC tag receives the structure of the PLC data type. A suitable address is assigned automatically. Structured PLC tags always start at word addresses.
   The highest structure element is displayed without its subelements in the table.

Note

Assignment rules and default values

- For the declaration of the PLC data type, note that the memory areas of the individual tags must not overlap. For example, tags of data type "Integer" must start at a word limit. If necessary, insert "fill tags" to prevent overlaps.
  See also: Permissible addresses and data types of PLC tags (Page 2476)
- It is not possible to assign default values for the individual components. Values that you enter in the "Default value" column are not evaluated. Tags of data types "DT" and "DTL" may therefore contain invalid values.

See also

Other useful information regarding structured PLC tags (Page 2485)

Addressing global variables (Page 2209)

Declaring PLC tags (Page 2481)

Programming PLC data types (Page 2854)

Data types (Page 2236)

9.1.2.7 Declaring global constants

Rules for global constants

Permitted characters

Names of global constants may consist of the following characters:

- Letters, numbers, special characters are permitted.
- Quotation marks are not permitted.

Unique constant names

The names of global constants must be unique throughout the CPU, even if the constants are located in different tag tables of a CPU. A name that is already used for a block, a PLC tag or another constant within the CPU, cannot be used for new constant. The uniqueness check does not differentiate between use of small and capital letters.

If you enter an already assigned name another time, a sequential number is automatically appended to the second name entered. For example, if you enter the name "Motor" a second time, the second entry is changed to "Motor(1)".

Permitted data types

For constants, all data types supported by the CPU are permitted, with the exception of structured data types.

Permitted values

You can select any value from the value range of the specified data type as constant value. For information on the value ranges, refer to the "Data types" chapter.

See also: Data types (Page 2236)

See also

Basics of constants (Page 2204)

Declaring global constants (Page 2489)

Constants (Page 2204)

Declaring global constants

Introduction

You declare constants in the "User constants" tab of a PLC tag table. During declaration you have to to enter a symbolic name, a data type and a fixed value for each constant. You can select any value from the value range of the specified data type as constant value. For information on the value ranges, refer to the "Data types" chapter.

See also: Data types (Page 2236)

Procedure

To declare constants, follow these steps:

1. Open a PLC tag table.
2. Open the "User constants" tab.
   The constants table opens.
3. Enter a constant name in the "Name" column.
4. Enter the required data type in the "Data type" column. You will be supported by autocompletion during input.
5. Enter a constant value in the "Value" column; this constant value must be valid for the selected data type.
6. If you want, enter comments on the constants in the "Comments" column. The entry of a comment is optional.
7. If you want to declare additional constants, place the cursor in the next row and repeat steps 3 to 6.

Syntax check

A syntax check is performed automatically after each entry, and any errors found are displayed in red. You do not have to correct these errors immediately - you can continue editing and make any corrections later. As long as the tag declaration contains syntax errors and the constant is used in the program, you will not be able to compile the program.

See also

Opening the PLC tag table (Page 2481)

Inserting a table row in the PLC tag table (Page 2493)

Structure of the PLC tag tables (Page 2474)

Rules for global constants (Page 2488)

Editing tables (Page 316)

Using autocompletion (Page 2540)

Constants (Page 2204)

9.1.2.8 Editing properties

Editing the properties of PLC tags

Properties of PLC tags

Overview

The following table provides an overview of the properties of PLC tags. The display of properties may vary depending on the CPU type.

Group	Property	Description
General	Name	A unique name within the CPU.
	Data type	Data type of the tags.
	Address	Tag address.
	Retain	Shows whether the tag is in the retentive memory area.
	Comment	Comment on the tag.
History	Date created	Time when the tag was created (cannot be changed).
	Last modified	Time when the tag was last changed (cannot be changed).
Usage	Visible in HMI	Shows whether the tag is visible by default in the HMI selection list.
	Accessible from HMI	Shows whether HMI can access this tag during runtime.

See also

Editing the properties of PLC tags (Page 2491)

Editing the properties of PLC tags

Editing properties in a PLC tag table

To edit the properties of one or more tags, follow these steps:

1. In the project tree, double-click the PLC tag table that contains the tags.
   The PLC tag table opens.
2. Change the entries in the columns.

Editing addresses in the program editor

To edit the address of a tag in the program editor, follow these steps:

1. Select the tag name.
2. Select the "Rewire tag" command in the shortcut menu.
   The "Rewire tag" dialog will open. The dialog shows a declaration table.
3. Enter the new address in the "Address" column.
4. Click the "Change" button to confirm the input.

Editing names in the program editor

To edit the name of a tag in the program editor, follow these steps:

1. Select the tag name.
2. Select the "Rename tag" command in the shortcut menu.
   The "Rename tag" dialog opens. The dialog shows a declaration table.
3. Enter the new name in the "Name" column.
4. Click the "Change" button to confirm the input.

Effect in the program

In the case of a change of the tag's name, data type or address, each location of use of the tag is automatically updated in the program.

See also

Properties of PLC tags (Page 2490)

Editing properties of global constants

Properties of global constants

Overview

The following table gives an overview of the properties of constants:

Group	Property	Description
General	Name	A unique name within the table
	Data type	Data type of the constants
	Value	Value that you defined for the constants. This value must be compatible with the declared data type. See also: Data types (Page 2236)
	Comment	Comment on the constants
History	Date created	Time when the constant was created (cannot be changed)
	Last modified	Time when the constant was last changed (cannot be changed)

Editing properties of global constants

Editing properties in a PLC tag table

To edit the properties of one or more constants, follow these steps:

1. In the project tree, double-click the PLC tag table that contains the constants.
   The PLC tag table opens.
2. Open the "User constants" tab.
3. Change the entries in the "Name", "Data type", "Value", or "Comments" column.

Effect in the program

In the case of a change of a constant's name, data type or value, each location of use of the constant is automatically updated in the program.

See also

Editing tables (Page 316)

9.1.2.9 Monitoring of PLC tags

Monitoring of PLC tags

You can monitor the current data values of the tags on the CPU directly in the PLC tag table.

Requirements

An online connection to the CPU is available.

Procedure

To monitor the data values, proceed as follows:

1. Open a PLC tag table.
2. Start monitoring by clicking the "Monitor all" button.
   The additional "Monitor value" column is displayed in the table. This shows the current data values.
3. End the monitoring by clicking the "Monitor all" button again.

Note

You also have the option of copying PLC tags to a monitor or force table so that you can monitor, control or force them in the table.

See also

Structure of the PLC tag tables (Page 2474)

Introduction to testing with the watch table (Page 2994)

Introduction for testing with the force table (Page 3021)

Copying entries in the PLC tag table (Page 2494)

9.1.2.10 Editing PLC tag tables

Inserting a table row in the PLC tag table

Procedure

Proceed as follows to insert a row above the selected row:

1. Select the row in front of which you want to insert a new row.
2. Click the "Insert row" button on the toolbar of the table.

Result

A new row is inserted above the selected row.

See also

Editing tables (Page 316)

Copying entries in the PLC tag table

You can copy PLC tags within a table or to other tables.

Procedure

To copy a tag, follow these steps:

1. Select the tags you want to copy.
   You can also select several tags by clicking on them one after the other while holding down the <Ctrl> key or by pressing and holding down <Shift> and clicking on the first and last tag.
2. Select "Copy" in the shortcut menu.
3. Position the insertion pointer at the location where you want to insert the tags.
4. Select "Paste" in the shortcut menu.

Or

1. Select the tag.
2. Hold down the left mouse button.
3. At the same time, press <Ctrl>.
4. Drag the tag to the destination.

Result

- The tag is copied to the destination.
- If there is a name conflict, a number is automatically appended to the tag name. For example, "Tag" becomes "Tag(1)".
- All other properties of the tag remain unchanged.

See also

Editing tables (Page 316)

Deleting entries in the PLC tag table

Procedure

Follow the steps below to delete elements:

1. Select the row with the element to be deleted. You can also select several rows by clicking on them one after the other while holding down the <Ctrl> key or by pressing and holding down <Shift> and clicking on the first and last row.
2. Select the "Delete" command in the shortcut menu.

See also

Editing tables (Page 316)

Sorting rows in the PLC tag tables

You can sort the rows in the tables alphanumerically by name, data type, or address.

Procedure

To sort the table rows, follow these steps:

1. Select the column by which you want to sort.
2. Click the column header.
   The column will be sorted in order of increasing values.
   An up arrow shows the sort sequence.
3. In order to change the sort sequence, click the arrow.
   The column will be sorted in order of decreasing values. A down arrow shows the sort sequence.
4. To restore the original sequence, click a third time on the column header.

See also

Editing tables (Page 316)

Automatically filling in cells in the PLC tag table

You can load the contents of one or several table cells into the cells below, automatically filling in the successive cells.

If you automatically fill in cells in the "Name" column, a consecutive number will be appended to each name. For example, "Motor" will become "Motor_1".

If you fill the cells in the column "Address" automatically, the addresses will be increased depending on the indicated data type.

Procedure

To automatically fill in successive cells, follow these steps:

1. Select the cells to be loaded.
2. Click the "Fill" symbol in the bottom right corner of the cell.
   The mouse pointer is transformed into a crosshair.
3. Keep the mouse button pressed and drag the mouse pointer downwards over the cells that you want to fill in automatically.
4. Release the mouse button.
   The cells are filled in automatically. If entries are already present in the cells that are to be automatically updated, a dialog appears in which you can indicate whether you want to overwrite the existing entries or whether you want to insert new rows for the new tags.

See also

Editing tables (Page 316)

Show and hide table columns

You can show or hide the columns in a table as needed.

Procedure

To show or hide table columns, follow these steps:

1. Click a column header.
2. Select the "Show/Hide" command in the shortcut menu.
   The selection of available columns is displayed.
3. To show a column, select the column's check box.
4. To hide a column, clear the column's check box.

See also

Editing tables (Page 316)

Editing PLC tags with external editors

Basics for importing and exporting

Introduction

You can export PLC tag tables to a standardized XLSX format for editing with external table editors. Similarly, you can import PLC tag tables created with external table editors to the TIA Portal.

Overwriting existing PLC tags and constants during import

Existing entries of the same name will be overwritten during import if they have the same name as the entries that will be imported.

Link to existing objects

References to PLC tags or constants that already exist in the project are updated automatically during import. The update is executed based on the name of the PLC tags and constants.

See also

Format of the export file (Page 2497)

Exporting PLC tags (Page 2498)

Importing PLC tags (Page 2498)

Format of the export file

Introduction

During the export of PLC tag tables, a standardized XSLX format will be generated that you can edit with external table editors.

This format is also expected during the import of tables.

Format of the export file

The sheet is always named "PLC Tags". This sheet can contain the displayed columns. The sorting order of columns can vary. The sheet does not necessarily have to include all columns. During import, the following values will be identified by a <no value> entry.

The names of the column headers are also clearly defined and are always expected in English.

The following table specifies the contents expected for the individual columns:

Element	Explanation
Name	Name of the tags
Path	Group and name of the PLC tag table
Data Type	The notation of the data type corresponds to the notation used in the PLC tag table.
Logical Address	The address can be specified with German or international mnemonics.
Comment	Free-form comments
Hmi Visible	The value TRUE or FALSE is expected.
Hmi Accessible	The value TRUE or FALSE is expected.

See also

Basics for importing and exporting (Page 2496)

Exporting PLC tags (Page 2498)

Importing PLC tags (Page 2498)

Exporting PLC tags

Requirement

A PLC tag table is open.

Procedure

To export PLC tags and constants, follow these steps:

1. In the PLC tag table, click the "Export" button.
   The "Export to Excel" dialog opens.
2. Select the path to which you want to save the export file.
3. Select whether to export tags and/or constants.
4. Click the "OK" button.

Result

The export file will be generated. Errors and warnings generated during export are indicated in the "Info" tab of the Inspector window.

See also

Basics for importing and exporting (Page 2496)

Format of the export file (Page 2497)

Importing PLC tags (Page 2498)

Importing PLC tags

Requirement

A table exists and it conforms to format specifications.

Procedure

To import a PLC tag table, follow these steps:

1. Open the "All tags" table.
2. Click the "Import" button.
   The "Import from Excel" dialog opens.
3. Select whether to import PLC tags and/or constants.
4. Select the table you want to import.
5. Click the "OK" button.

Result

The PLC tag table will be imported.

Errors and warnings generated during export are indicated in the "Info" tab of the Inspector window.

See also

Basics for importing and exporting (Page 2496)

Format of the export file (Page 2497)

Exporting PLC tags (Page 2498)

Editing individual PLC tags with external editors

To edit individual PLC tags in external editors outside the TIA portal, you can export or import these tags using copy and paste. However, you cannot copy structured tags to an editor.

Requirement

A PLC tag table and an external editor are opened.

Procedure

To export and import individual PLC tags, follow these steps:

1. Select one or more PLC tags.
2. Select "Copy" in the shortcut menu.
3. Switch to the external editor and paste the copied tags.
4. Edit the tags as required.
5. Copy the tags in the external editor.
6. Switch back to the PLC tag table.
7. Select "Paste" in the shortcut menu.

Note

You also have the option of export or importing PLC tags as mass data.

See also: Editing PLC tags with external editors (Page 2496)

9.1.3 Creating and managing blocks

9.1.3.1 Creating blocks

Block folder

Function

You can find a "Program blocks" folder in the project tree, in which you can create and manage the following blocks:

- Organization blocks (OB)
- Function blocks (FB)
- Functions (FCs)
- Data blocks (DB)

A "System blocks" subfolder containing another subfolder, "Program resources", is also created in the "Program blocks" folder the first time you drag an instruction to your program which is an internal system function block. The instance data block of the internal system function block is also pasted to the "Program resources" folder. You can move or copy such instance data blocks from the "Program resources" folder to any other folder and rename or delete them. You can also move your blocks into the "Program resources" folder. Blocks in the "Program resources" folder that are not required to run the user program are removed during the next compilation. If the "Program resources" folder contains no more blocks then it is also deleted with the "System blocks" folder.

A program cycle OB is automatically generated for each device and inserted in the "Program blocks" folder.

See also

Creating functions and function blocks (Page 2502)

Creating data blocks (Page 2503)

Creating organization blocks (Page 2501)

Using blocks from libraries (Page 2504)

Creating organization blocks

Requirement

The "Program blocks" folder in the project tree is open.

Procedure

To create an organization block, follow these steps:

1. Double-click the "Add new block" command.
   The "Add new block" dialog box opens.
2. Click the "Organization block (OB)" button.
3. Select the type of new organization block.
4. Enter a name for the new organization block.
5. Enter the properties of the new organization block.
6. To enter additional properties for the new organization block, click "Additional information". An area with further input fields is displayed.
7. Enter all the properties you require.
8. Activate the "Add new and open" check box if the organization block does not open as soon as it is created.
9. Confirm your entries with "OK".

Result

The new organization block is created. You can find the organization block in the project tree in the "Program blocks" folder. You can assign additional parameters to some organization blocks in the inspector window or device view after they have been created. The organization block description will state whether the newly created organization block has additional parameters.

See also

Organization blocks (OB) (Page 2170)

Block folder (Page 2500)

Creating functions and function blocks (Page 2502)

Creating data blocks (Page 2503)

Using blocks from libraries (Page 2504)

Entering a block title (Page 2510)

Entering a block comment (Page 2511)

Creating functions and function blocks

Requirement

The "Program blocks" folder in the project tree is open.

Procedure

To create a function (FC) or a function block (FB), follow these steps:

1. Double-click the "Add new block" command.
   The "Add new block" dialog box opens.
2. Click the "Function block (FB)" or "Function (FC)" button.
3. Enter a name for the new block.
4. Enter the properties of the new block.
5. To enter additional properties for the new block, click "Additional information".
   An area with further input fields is displayed.
6. Enter all the properties you require.
7. Activate the "Add new and open" check box if the block does not open as soon as it is created.
8. Confirm your entries with "OK".

Result

The new block is created. You can find the block in the project tree in the "Program blocks" folder.

See also

Function blocks (FB) (Page 2171)

Functions (FCs) (Page 2170)

Basics of block access (Page 2175)

Block folder (Page 2500)

Creating organization blocks (Page 2501)

Creating data blocks (Page 2503)

Using blocks from libraries (Page 2504)

Entering a block title (Page 2510)

Entering a block comment (Page 2511)

Creating data blocks

Requirement

The "Program blocks" folder in the project tree is open.

Procedure

To create a data block, follow these steps:

1. Double-click the "Add new block" command.
   The "Add new block" dialog box opens.
2. Click the "Data block (DB)" button.
3. Select the type of the data block. You have the following options available to you:
   - To create a global data block, select the list entry "Global DB".
   - To create an ARRAY data block, select the "ARRAY DB" entry in the list.
   - To create an instance data block, select the function block to which you want to assign the instance data block from the list. The list contains only the function blocks that were previously created for the CPU.
   - To create a data block based on a PLC data type, select the PLC data type from the list. The list contains only the PLC data types that were previously created for the CPU.
   - To create a data block based on a system data type, select the system data type from the list. The list contains only those system data types that have already been inserted to program blocks in the CPU.
4. Enter a name for the data block.
5. Enter the properties of the new data block.
6. If you have selected an ARRAY DB as the data block type, enter the ARRAY data type and the high limit for the ARRAY.
   You can change the high limit for the ARRAY at any time in the property window of the created block. The ARRAY data type cannot be changed subsequently.
7. To enter additional properties of the new data block, click "Additional information".
   An area with further input fields is displayed.
8. Enter all the properties you require.
9. Activate the "Add new and open" check box if the block does not open as soon as it is created.
10. Confirm your entry with "OK".

Result

The new data block is created. You can find the data block in the project tree in the "Program blocks" folder.

See also

Global data blocks (DB) (Page 2172)

Instance data blocks (Page 2174)

Block folder (Page 2500)

Creating organization blocks (Page 2501)

Creating functions and function blocks (Page 2502)

Using blocks from libraries (Page 2504)

Basics of block access (Page 2175)

Programming data blocks (Page 2825)

Programming PLC data types (Page 2854)

System data types (Page 2281)

Using blocks from libraries

You can save blocks in the project library or in a global library, so that you can use them more than once within a user program. You can insert the blocks as master copies or as types.

See also: Library basics (Page 415)

Requirement

- The "Libraries" task card is displayed.
- No write protection is set for global libraries.

Adding blocks as master copies to the project library or to a global library

To add new blocks as master copies to the project library or to a global library, follow these steps:

1. Maximize the project library or the global library.
2. Use drag-and-drop to move the block you wish to add to the library to the "Master copies" folder or any one of the "Master copies" subfolders in the project library or a global library. Do not release the left mouse button until a small plus sign appears underneath the mouse pointer.

Or:

1. Copy the element you want to add as master copy.
2. Maximize the project library or the global library.
3. Right-click the "Master copies" folder or any of its subfolders.
4. Select "Paste" in the shortcut menu.

Adding blocks as types to the project library or to a global library

To add new blocks as types to the project library or to a global library, follow these steps:

1. Maximize the project library or the global library.
2. Drag-and-drop the element you want to add as a type into the "Types" folder or any of its subfolders in the project library or a global library. Do not release the left mouse button until a small plus sign appears underneath the mouse pointer.

Or:

1. Copy the element you want to add as a type.
2. Maximize the project library or the global library.
3. Right-click the "Types" folder or any of its subfolders.
4. Select "Paste" in the shortcut menu.

Using blocks of the project library or a global library

To use a block from the project library or a global library in your project, follow these steps:

1. Maximize the project library or the global library so that you can see the block you wish to use.
2. Use a drag-and-drop operation to move the block to the CPU block folder. If the selected insertion points is not allowed, the mouse pointer will appear as a circle with a slash.

Note

If you derive an instance from a type in a global library, the type is also inserted into the project library. The instance is then only linked to the type in the project library.

See also

Using libraries (Page 415)

Copying and pasting blocks

Basics of copying and pasting blocks

Function

You can also create new blocks by copying existing blocks and pasting the copy. Note the following principles when copying to CPUs of the same device family:

- You can copy organization blocks (OBs), functions (FCs), function blocks (FBs), and global data blocks (DBs) without restriction.
- You can copy instance data blocks only for the same function block, since the assignment to the function block cannot be changed afterwards. However, the assignment is canceled if you copy the instance data block to a different CPU. If a function block with the same name exists there, the instance data block will be assigned to this function block. If you copy the instance data block together with the function block into the other CPU, the instance data block is assigned to the copy of the function block.

The various device families sometimes support different blocks, especially in the case of organization blocks. However, function blocks and functions can also be programmed on the various devices with different access types. Therefore, not all blocks are supported on all devices. Note the following principles when copying to a different device family:

- Copying to an S7-1200 CPU:
  - Organization blocks with "Optimized" access type can be copied to an S7-1200. If the copied OB type is supported by the S7-1200 CPU, the copied OB retains the properties of its event. You must, however, compile it again.
  - Although organization blocks with the "Standard" access type can be copied to an S7-1200, they are not supported by the CPU.
  - Function blocks (FBs), functions (FCs) and global data blocks (DBs) with "Optimized" access type can be copied to an S7-1200. However, they must be recompiled after this.
  - Although function blocks (FBs), functions (FCs) and global data blocks (DBs) with "Standard" access type can be copied to an S7-1200, they are not supported by the CPU.
  - Instance data blocks: If there is a function block in the target CPU with the name that was assigned to the instance data block in the source CPU, the instance data block is assigned to the function block in the target CPU. If you copy the instance data block together with the function block to which it was assigned in the source CPU into the target CPU, the instance data block is assigned to the copy of the function block.
- Copying to an S7-1500 CPU:
  - Organization blocks with "Optimized" access type can be copied to an S7-1500. If the copied OB type is supported by the S7-1500 CPU, the copied OB retains the properties of its event. You must, however, compile it again. OB types that are not supported receive a no parking symbol.
  - Organization blocks with "Standard" access type can be copied to an S7-1500. If the OB derives from an S7-300/400 CPU, it receives the standard event of the corresponding OB type. If the OB derives from an S7-1200/1500 CPU, it receives the properties of its event. However, it must be compiled again.
  - Function blocks (FBs), functions (FCs) and global data blocks (DBs) with "Optimized" access type can be copied to an S7-1500. However, they must be recompiled after this.
  - Although function blocks (FBs), functions (FCs) and global data blocks (DBs) with "Standard" access type can be copied to an S7-1500, they are not supported by the CPU.
  - Instance data blocks: If there is a function block in the target CPU with the name that was assigned to the instance data block in the source CPU, the instance data block is assigned to the function block in the target CPU. If you copy the instance data block

together with the function block to which it was assigned in the source CPU into the target CPU, the instance data block is assigned to the copy of the function block.

- Copying to S7-300/400 CPUs:
  - Organization blocks can be copied as required between S7-300 and S7-400.
  - Although organization blocks from S7-1200/1500 CPUs can be copied to S7-300/400 CPUs, they are not supported by the target CPU.
  - Function blocks (FBs), functions (FCs) and global data blocks (DBs) can be copied as required between S7-300 and S7-400.
  - Although function blocks (FBs), functions (FCs) and global data blocks (DBs) can be copied from S7-1200/1500 CPUs to S7-300/400 CPUs, they are not supported by the target CPU.
  - Instance data blocks: If there is a function block in the target CPU with the name that was assigned to the instance data block in the source CPU, the instance data block is assigned to the function block in the target CPU. If you copy the instance data block together with the function block to which it was assigned in the source CPU into the target CPU, the instance data block is assigned to the copy of the function block.

In the project tree, blocks that are not supported are indicated by the no parking symbol. Blocks with a no parking symbol cannot be edited, but only used again as copy source.

Copying data

With paste, all the block data is copied and forwarded to the copy. This data includes:

- Block interface tags
- All networks
- Comments in all existing compilations
- Messages defined in the block
- The entire program code of the copied block including the call instructions contained in the block.
  However, called blocks and their associated instance data blocks are not copied.

Avoiding name conflicts during pasting

When pasting copied blocks with identical names to already existing blocks, the following mechanisms are used to avoid name conflicts:

- Pasting the copied block into the same CPU:
  The copy of the block gets a name that is extended by a number. For example, if block "A" is copied, a possible name for the copy is "A_1". Consecutive numbering is not used, but rather the smallest free number. The copy of block "A" can also get the name "A_25", if no lower number is available.
- Pasting the copied block into another CPU:
  A dialog box opens in which you can select whether the block with the same name will be replaced or the copied block will be pasted with a duplicate designation (Name_Number).

Note

Number conflicts

Number conflicts may occur, if the pasted block has the same block number as an existing block. The block number is not automatically changed during pasting. This double number may have an effect on block calls. When you copy blocks you should therefore check the block number carefully and correct duplicate block numbers manually or using the block properties. Duplicate block numbers lead to a compilation error.

See also

Copying blocks (Page 2509)

Pasting blocks (Page 2509)

Copying blocks

Requirement

The "Program blocks" folder is opened in the project tree.

Procedure

To copy a block, follow these steps:

1. Right-click the block that you want to copy.
2. Select "Copy" in the shortcut menu.

Result

A copy of the block is now on the clipboard and can be pasted either into the same CPU or into another one.

See also

Basics of copying and pasting blocks (Page 2506)

Pasting blocks (Page 2509)

Pasting blocks

Requirement

You have copied a block.

Procedure

To paste a copied block and its data into a CPU, follow these steps:

1. In the project tree, open the folder structure for the CPU into which you want to paste the copied block.

   Note

   Please note that you can only paste the copied block into a CPU which supports the block programming language and type.

2. Right-click on the "Program blocks" folder.
3. Select "Paste" in the shortcut menu.
   - If you are pasting the block into the same CPU as the original block, "_<consecutive number>" will be appended to the name of the copy.
   - If you are pasting the block into a different CPU where a block of the same name already exists, the "Paste" dialog box opens. Select the required option and confirm with "OK".

See also

Basics of copying and pasting blocks (Page 2506)

Copying blocks (Page 2509)

Entering a block title

The block title is the title of the block. It is not the same thing as the block name, which was assigned when the block was created. The length of the block title is restricted to one row. You can enter the block title for open and closed blocks.

Requirement

A code block is available.

Enter block title for open block

To insert the block title in an open block, follow these steps:

1. Click on the title bar of the block in the program editor.
2. Enter the block title.

Enter block title for closed blocks

To insert the block title in a closed block, follow these steps:

1. Right-click the block in the project tree.
2. Select the "Properties" command in the shortcut menu.
   The dialog with the properties of the block opens.

3. Select the entry "Information" in the area navigation.
4. Enter the block title in the "Title" input field.
5. Confirm your entry with "OK".

See also

Creating organization blocks (Page 2501)

Creating functions and function blocks (Page 2502)

Entering a block comment (Page 2511)

Entering a block comment

You can use the block comment to document the entire code block. For example, you can indicate the purpose of the block or draw attention to special characteristics. You can enter the block comment for open and closed blocks.

Requirement

A code block is available.

Enter block comment for open blocks

To insert a block comment in an open block, proceed as follows:

1. Click the small arrow in front of the block title.
   The right arrow becomes a down arrow, and the comment area is displayed.
2. Click "Comment" in the comment area.
   The "Comment" text passage is selected.
3. Enter the block comment.

Enter block comments for closed blocks

To insert the block comment in a closed block, follow these steps:

1. Right-click the block in the project tree.
2. Select the "Properties" command in the shortcut menu.
   The dialog with the properties of the block opens.
3. Select the entry "Information" in the area navigation.
4. Enter the block comment in the "Comment" input field.
5. Confirm your entry with "OK".

See also

Creating organization blocks (Page 2501)
Creating functions and function blocks (Page 2502)
Entering a block title (Page 2510)

9.1.3.2 Specifying block properties

Basics of block properties

Block properties

Each block has certain properties, which you can display and edit. These properties are used amongst other things to:

- Identify the block
- Display the memory requirements and the compilation status of the block
- Display the time stamp
- Display the reference information
- Specify the access protection

See also

Overview of block properties (Page 2512)
Block time stamps (Page 2515)
Displaying and editing block properties (Page 2517)
Setting the mnemonics (Page 2533)

Overview of block properties

Overview

The properties of the blocks are block and CPU-specific. Not all properties are therefore available for all blocks or in all CPU families. The following table gives an overview of block properties:

Group	Property	Description
General	Name	Unique block name within the station
	Constant name	Name of the constant pasted for the OB in the PLC tag table
	Type	Block type (cannot be changed)
	Number	Block number
	Event class	Event class of an OB (cannot be changed)

Group	Property	Description
	Language	Programming language of the block
	Language in networks	Language used to program conditions in GRAPH blocks.
	Process image partition number	Display of the process image partitions that are assigned to the organization block (cannot be changed)
	ARRAY data type	Data type of an ARRAY data block (cannot be changed)
	ARRAY limit	High limit of an ARRAY data block The "Move operations" section of the "Instructions" task card offers options for addressing of ARRAY data blocks.
Information	Title	Block title
	Comment	Block comment
	Version	Version number of the block
	Family	Block family name
	Author	Name of the author, company name, department name, or other names
	User-defined ID	ID created by the user
Time stamps	Block	Times of creation and time of change of the block (cannot be changed)
	Interface	Time of creation of the block interface (cannot be changed)
	Code	Code modification time (non-editable)
	Data	Data modification time (non-editable)
Compilation	Status	Details of the last compilation run (cannot be changed)
	Lengths	Details of the block lengths (cannot be changed)
Protection	Protection	Setting up know-how protection and copy protection for the block See also: Protecting blocks (Page 2929)
Attributes	Optimized block access	The tag declaration for blocks with optimized access contains only the symbolic names of the data elements. The system automatically optimizes and manages the addresses. The CPU performance increases and access errors, for example, from SIMATIC HMI, are prevented. See also: Blocks with optimized access (Page 2175)
	IEC check	The compatibility of the operands in comparison operations and arithmetic operations are tested according to IEC 61131. You have to explicitly convert non-compatible operands. See also: Overview of data type conversion (Page 2372)
	Handle errors within block	Troubleshooting inside the block with the GET_ERROR or GET_ERR_ID instruction (cannot be changed). See also: Handling program execution errors (Page 2821)
	Create extended status information	Allows you to monitor all tags in an SCL block. This option does, however, increase the program memory space required and the execution times.
	Check ARRAY limits	Checks whether array indexes are within the range declared for an ARRAY during the runtime of an SCL block. The block enable output ENO is set to "0" if an array index is outside of the permitted range.
	Set ENO automatically	Checks whether errors occur in the processing of certain instructions during the runtime of an SCL block. The block enable output ENO is set to "0" if a runtime error occurs.

Group	Property	Description
	Create minimized DB	Creates instance data blocks for GRAPH blocks of the S7-300 and S7-400 in minimized format. This option reduces the GRAPH FB memory space required, however you will only receive limited program status information.
	Skip steps	If the transitions before and after a step become valid simultaneously in a GRAPH block, the step will not be activated and will be skipped.
	Acknowledge supervision alarms	Any supervision error which occurs during the operation of a GRAPH block must be acknowledged before the program can continue.
	Permanent processing of all interlocks in manual mode	Permanently monitors all interlock conditions in a GRAPH block in manual mode.
	Lock operating mode selection	Prevents a GRAPH block operating mode being selected.
	Data block write-protected in the device	Indicates whether the data block is read-only in the target system, and cannot be overwritten while the program is running (for data blocks only)
	Only store in load memory	On activation the data block is stored only in the load memory, occupies no space in the work memory, and is not linked into the program. The "Move operations" section of the "Instructions" task card offers options for transferring data blocks to the work memory (for data blocks only).
	Start information	Here you specify the structure of the start information of the OB for S7-1500 CPUs: either like for S7-300 and S7-400 CPUs or optimized start information.
	Priority	Shows the priority set for organization blocks. The CPU family used and the type of the organization block determine whether you can change the priority.
	Parameter assignment by means of registers	Enables parameter input by means of registers in an STL block of S7-1500. This feature allows you to use the "Conditional call of blocks" (CC) and "Unconditional call of blocks" (UC) instructions in the block.
	Block can be used as know-how protected library item	Shows whether or not the block in the library can be used with know-how protection.
	Enable readback	Enables you to mark individual parameters of the block for readback. The "Tag readback" function is relevant if the block is used in a CFC chart.
	Block representation	Specifies how the block is represented in a CFC chart.
Time-of-day interrupt	Time-of-day interrupt	Parameters of the time-of-day interrupt OB: active (yes or no), execution, start date and time, local time or system time
Cyclic interrupt	Cyclic interrupt	Cycle time and phase shift of the cyclic interrupt OB
Triggers	Triggers	Display of start events of the hardware interrupt OB
Isochronous mode	Isochronous mode	Parameters of the isochronous mode interrupt OB: application cycles, automatic setting (yes or no), delay time. Also indicated is the PROFINET IO system or DP master system whose IO devices or DP slaves are assigned to the isochronous mode interrupt OB.
Download without re-initialization	Reserve in volatile memory	Defines the reserve in volatile memory that can be used for interface extensions. The number of currently available bytes is displayed in parentheses. This information is updated with each compilation.

Group	Property	Description
	Activate download without re-initialization for retentive tags	Enables definition of a reserve in retentive memory.
	Reserve in retentive memory	Defines the reserve in retentive memory that can be used for interface extensions. The number of currently available bytes is displayed in parentheses. This information is updated with each compilation.

See also

Basics of block properties (Page 2512)

Block time stamps (Page 2515)

Displaying and editing block properties (Page 2517)

Basics of block access (Page 2175)

General settings for the PLC programming (Page 2542)

Block time stamps

Introduction

Blocks receive a number of different time stamps which show you when the block was created and when it was last changed. These time stamps are also used for automatic consistency checks before compilation.

Code block time stamps

The following time stamps are generated for code blocks (OBs, FBs, FCs):

- Block: Created on, Modified on
- Interface: Modified on
- Code/data: Modified on

A time stamp conflict is displayed during compilation if the time stamp for the block calling is older than that of the interface for the block called.

Time stamps for code blocks are updated as follows:

- Block: The time stamp for the last block modification is always the same as the time stamp either of the interface or of the code depending on which area was modified last.
- Interface: The interface time stamp is updated each time the interface is modified. Even if you manually undo a change to the interface, for example change the name back, that is also a change which updates the time stamp. However, if you undo the change using the "Undo" function, the time stamp will be reset to the value it had before the undone change.
- Code/data: The code time stamp is updated each time the block code is changed. Even if you manually undo a change to the code, for example remove an instruction, that is also a change which will update the time stamp. However, if you undo the change using the "Undo" function, the time stamp will be reset to the value it had before the undone change.

Global data block time stamps

The following time stamps are generated for global data blocks:

- Block: Created on, Modified on
- Interface: Modified on
- Data: Modified on

Time stamp conflict is reported during compilation of a global data block based on a PLC data type if the time stamp of the global data block is older than the time stamp of the PLC data type used.

The time stamps for global data blocks are updated as follows:

- Block: The time stamp for the last change to a global data block is always the same as the time stamp for the interface and the data.
- Interface and data: The interface time stamps and the data are updated each time the global data block is changed. Even if you manually undo a change, for example remove a tag, that is also a change which will update the time stamp. However, if you undo the change using the "Undo" function, the time stamps will be reset to the value they had before the undone change.

Instance data block time stamps

The following time stamps are generated for instance data blocks:

- Block: Created on, Modified on
- Interface: Modified on
- Data: Modified on

A time stamp conflict will be reported during compilation of an instance data block if the interface time stamps of the instance data block are not identical to those of the function block.

The time stamps for instance data blocks are updated as follows:

- Block: The time stamp for the last change to an instance data block is always the same as the time stamp for the interface and the data.
- Interface and data: The interface time stamps and the data are updated each time the instance data block is changed. Even if you manually undo a change, for example cancel the tag retain setting, that is also a change which will update the time stamps. However, if you undo the change using the "Undo" function, the time stamps will be reset to the value they had before the undone change.

PLC data type time stamps

The following time stamps are generated for PLC data types:

- Block: Created on, Modified on
- Interface: Modified on

The time stamps for PLC data types are updated as follows:

- Block: The time stamp for the last change to a PLC data type is always the same as the interface time stamp.
- Interface: The interface time stamp is updated each time the PLC data type is changed. Even if you manually undo a change, for example delete the content of a PLC data type, that is also a change which will update the time stamp. However, if you undo the change using the "Undo" function, the time stamp will be reset to the value it had before the undone change.

See also

Basics of block properties (Page 2512)

Overview of block properties (Page 2512)

Displaying and editing block properties (Page 2517)

Basic information on compiling blocks (Page 2898)

Displaying and editing block properties

The properties of the blocks are block and CPU-specific. Not all properties are therefore available for all blocks or in all CPU families. Properties that can only be displayed are write-protected.

Displaying and editing properties of a closed block

To display and edit the properties of a closed block, follow these steps:

1. Open the "Program blocks" folder in the project tree.
2. Right-click the block whose properties you want to display or edit.
3. Select the "Properties" command in the shortcut menu.
   The properties dialog box of the block opens.
4. In the area navigation, click a group whose properties you want to display or edit.
5. Change the relevant property.
6. Confirm your entries with "OK".

Displaying and editing properties of an open block

To display or edit the properties of an open block, follow these steps:

1. Select the "Inspector window" check box in the "View" menu.
   The Inspector window opens.
2. Click the "Properties" tab.
   The properties of the block are shown in the "Properties" tab of the Inspector window.
3. In the area navigation, click a group whose properties you want to display or edit.
4. Change the relevant property.

Result

The properties of the block will be changed. The changes are not saved until the project is saved.

See also

Basics of block properties (Page 2512)

Overview of block properties (Page 2512)

Block time stamps (Page 2515)

9.1.3.3 Managing blocks

Opening blocks

You can open a block in several different ways:

- Open block directly
  You can open a block directly if the corresponding block folder is open in the project tree or in the overview window.
- Find and open block
  You can look for blocks within a project, a device or the "Program blocks" folder and then open them.

Open block directly

To open a block directly, follow these steps:

1. Open the folder with the block you wish to open in the project tree or in the overview window.
2. Double-click on the block you wish to open.

Find and open block

To find and then open a block, follow these steps:

1. Right-click on the project, a device, the "Program blocks" folder or a lower-level folder of "Program blocks" in the project tree.
2. Select the "Search in PLC and open" command from the shortcut menu.
   The "Search in PLC and open" dialog box is displayed.
3. Enter the name, the address or the type of block you are looking for.
   The block list is filtered further with each letter entered. The block list is closed if there is no block that matches the input. You can show the complete block list at any time by clicking the button to the right of the text box. Keep in mind that there is no filtering in this case. If you want to filter for your inputs again, click the button once again.
4. In the block list, click the block you wish to open.

Result

The block will open in the program editor.

See also

Saving blocks (Page 2519)
Closing blocks (Page 2519)
Renaming blocks (Page 2520)
Deleting blocks offline (Page 2521)
Deleting blocks online (Page 2521)
Opening know-how protected blocks (Page 2933)

Saving blocks

Blocks are always saved together with the project. Faulty blocks can also be saved. This allows the error to be resolved at a convenient time.

Procedure

To save a block, follow these steps:

1. Select the "Save" or "Save as" command in the "Project" menu.
   See also: Saving projects (Page 338)

See also

Opening blocks (Page 2518)
Closing blocks (Page 2519)
Renaming blocks (Page 2520)
Deleting blocks offline (Page 2521)
Deleting blocks online (Page 2521)

Closing blocks

Procedure

To close a block, follow these steps:

1. Click the "Close" button in the title bar of the program editor.

Note

Note that the block will not be saved on closing.

See also

Opening blocks (Page 2518)
Saving blocks (Page 2519)
Renaming blocks (Page 2520)
Deleting blocks offline (Page 2521)
Deleting blocks online (Page 2521)

Renaming blocks

Requirement

The "Program blocks" folder is opened in the project tree.

Procedure

To change the name of a block, follow these steps:

1. Right-click the block that you want to rename.
2. Select the "Rename" command in the shortcut menu.
   The block name in the project tree changes to an input field.
3. Input the new name for the block.
4. Confirm your entry with the Enter key.

Result

The name of the block is now changed at all points of use in the program.

See also

Opening blocks (Page 2518)
Saving blocks (Page 2519)
Closing blocks (Page 2519)
Deleting blocks offline (Page 2521)
Deleting blocks online (Page 2521)

Deleting blocks offline

Requirement

The "Program blocks" folder opens in the project tree.

Procedure

To delete a block that exists offline, proceed as follows:

1. In the project tree in the "Program blocks" folder, right-click on the block that you want to delete.
2. Select the "Delete" command in the shortcut menu.
3. Confirm the safety prompt with "Yes".
   The block is deleted offline from the project.

Note

If you are deleting organization blocks, note that events may be assigned to these blocks. If you delete such organization block the program cannot respond to parameterized events.

See also

Opening blocks (Page 2518)

Saving blocks (Page 2519)

Closing blocks (Page 2519)

Renaming blocks (Page 2520)

Deleting blocks online (Page 2521)

Downloading blocks (Page 2903)

Deleting blocks online

Requirement

The "Program blocks" folder in the project tree is open.

Procedure

To delete a block that exists online, follow these steps:

1. In the "Program blocks" folder in the project tree, right-click on the block that you wish to delete from the device.
2. Select the "Delete" command from the shortcut menu.
   The "Delete" dialog opens.

3. Select the "Delete from device" check box.
4. Click "Yes".
   The block will be deleted from the device online.

See also

Opening blocks (Page 2518)

Saving blocks (Page 2519)

Closing blocks (Page 2519)

Renaming blocks (Page 2520)

Deleting blocks offline (Page 2521)

Deleting CPU data blocks

You may delete CPU data blocks both in offline and online mode.

Deleting CPU data blocks in offline mode

Proceed as follows to delete a CPU data block from the offline project:

1. Right-click the CPU data blocks that you want to delete in the project navigation, "Program blocks" folder.
2. Select the "Delete" command from the shortcut menu.
3. Confirm the safety prompt with "Yes".
   The CPU data block is deleted from the offline project.

Deleting CPU data blocks in online mode

Proceed as follows to delete a CPU data block from the online project:

1. Establish an online connection to the device containing the CPU data block that you want to delete.
2. Right-click the CPU data block that you want to delete from the device in the project navigation, "Program blocks" folder.
3. Select the "Delete" command from the shortcut menu.
   The "Delete" dialog opens.
4. Select the "Delete from device" check box.
5. Click "Yes".
   The CPU data block is deleted from the online device.

See also

CPU data blocks (Page 2174)

9.1.4 Programming blocks

9.1.4.1 Program editor

Overview of the program editor

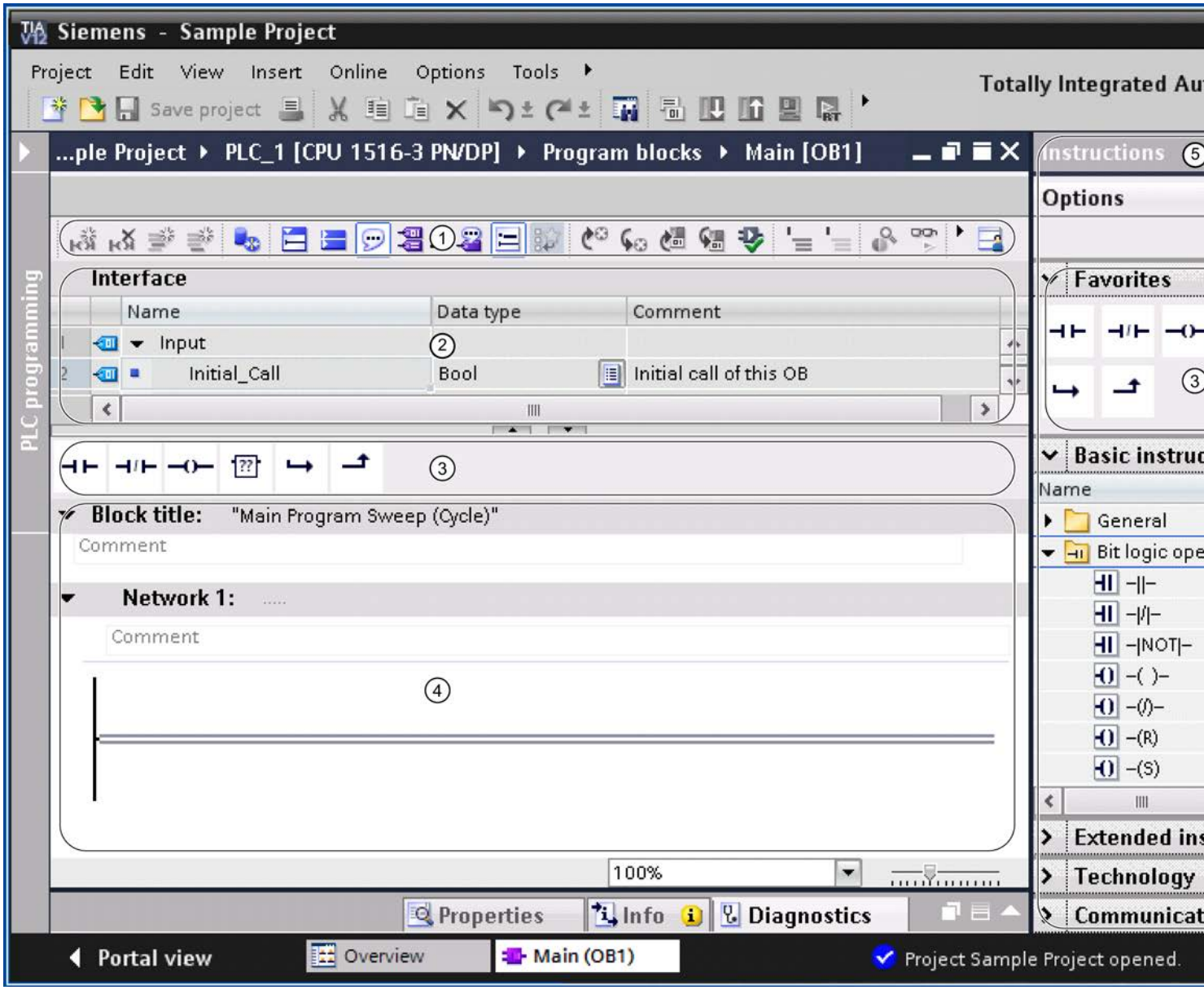
Function of the program editor

The program editor is the integrated development environment for programming functions, function blocks, and organization blocks. If offers comprehensive support for programming and troubleshooting.

The appearance and functionality of the program editor can vary depending on the CPU, programming language and block type used.

Structure of the program editor

Using LAD as an example, the following figure shows the components of the program editor:



① Toolbar
② Block interface
③ "Favorites" pane in the "Instructions" task card and favorites in the program editor
④ Programming window
⑤ "Instructions" task card
⑥ "Testing" task card

Toolbar

The toolbar allows you to access the principal functions of the program editor, such as:

- Show and hide absolute operands
- Showing and hiding favorites
- Skip to syntax errors
- Update block calls
- Show and hide program status

The functions available in the toolbar can vary depending on the programming language used.

Block interface

The block interface contains the declarations for local tags used solely within the block. The sections available depend on the block type.

Favorites

You can save frequently used instructions as favorites. These favorites are then displayed in the "Instructions" task card and the "Favorites" pane. You can also display favorites in the program editor using the program editor toolbar. This allows you to access your favorites even when the "Instructions" task card is not visible.

Programming window

The programming window is the work area of the program editor. You can enter the program code in this window. The appearance and functionality of the program window can vary depending on the programming language used.

"Instructions" task card

The "Instructions" task card offers you easy access to all instructions available for creating your program. The instructions are broken down by area into a number of different palettes. You can show additional information on the instructions via the "Show column headers and additional columns" button in the task card toolbar. You can modify the arrangement of columns by clicking a column header and moving the column to the new position by means of drag-and-drop.

If an instruction profile is active, the offered instructions may vary.
See also: Using instruction profiles (Page 2535)

"Testing" task card

You can set settings in the "Testing" task card for troubleshooting using the program status. The functions of the "Testing" task card are only available in online mode. It contains the

following panes, which are displayed depending on the selected CPU and the configured programming language of the block:

- CPU operator panel
  The CPU operator panel allows you to switch the CPU operating mode.
- Breakpoints
  You can test blocks you created in the textual programming languages STL and SCL in single step mode. To do this, set breakpoints in the program code.
  In the "Breakpoints" pane, you can find all breakpoints that you have set and you can activate, delete, navigate to, or set the call environment for the individual breakpoints.
- PLC register
  This pane allows you to read off the values for the PLC registers and the accumulators.
- Sequence control
  In this pane you can set the operating mode for testing sequencers for GRAPH blocks.
- Test settings
  This pane allows you to specify the test settings for the GRAPH block.
- Call environment
  This pane allows you to specify the call environment for the block.
- Call hierarchy
  In this pane, you can trace the call hierarchy of the blocks. You only see the call hierarchy during block monitoring.

See also

Declaring the block interface (Page 2544)

Layout of the block interface (Page 2544)

Creating program code (Page 2568)

Enlarging the programming window (Page 2531)

Using the keyboard in the program editor

Navigating in the editor

Function	Shortcut key
Open "Instructions" task card	<Ctrl+Shift+C>
Open "Testing" task card	<Ctrl+Shift+O>
Add new block	<Ctrl+N>
Expand all networks	<Alt+F11>
Collapse all networks	<Alt+F12>
Navigate to the next point of use of the selected block or operand	<Ctrl+Shift+F>
Navigate to the previous point of use of the selected block or operand	<Ctrl+Shift+G>

Function	Shortcut key
Navigate to the next read/write access	<Alt+F8>
Navigate to the previous read/write access	<Alt+F9>

Navigating in the program code (LAD/FBD)

Function	Selected object	Shortcut key
Navigating between objects in the network	Object in the network	Arrow keys
Navigate to the first element of the network	Object in the network	<Home>
Navigate to the last element of the network	Object in the network	<End>
Navigate to the next element of the network	Object in the network	<Tab>
Navigate to the previous element of the network	Object in the network	<Shift+Tab>
Insert network	Any	<Ctrl+R>

Navigating in the program code (STL/SCL)

Function	Position of the cursor	Shortcut key
Navigating in the program code	Line	Arrow keys
One word to the right/left	Line	<Ctrl+arrow keys>
Cursor to start of line	Line	<Home>
Cursor to end of line	Line	<End>
Cursor to start of code section	Line	<Ctrl+Home>
Cursor to end of code section	Line	<Ctrl+End>
To the next network (STL only)	Network title	<Down arrow>
To the next network (STL only)	Line	<Tab> Repeat the shortcut keys until the insertion point is in the next network.
To the previous network (STL only)	Network title	<Up arrow>
To the previous network (STL only)	Line	<Shift+Tab> Repeat the shortcut keys until the insertion point is in the previous network.
Insert network	Any	<Ctrl+R>

Inserting instructions (LAD)

Function	Selected object	Shortcut key
Insert normally open contact	Rung	<Shift+F2>
Insert normally closed contact	Rung	<Shift+F3>
Insert empty box	Rung	<Shift+F5>
Insert assignment	Rung	<Shift+F7>

Function	Selected object	Shortcut key
"Insert "Open branch"	Rung	<Shift+F8>
"Insert "Close branch"	Rung	<Shift+F9>

Inserting instructions (FBD)

Function	Selected object	Shortcut key
Insert assignment	Network, input or output	<Shift+F7>
Insert empty box	Network	<Shift+F5>
"Insert "Open branch"	Connection line between two boxes	<Shift+F8>
Invert RLO	Network, input or output	<Ctrl+Shift+4>
Insert input	Network, input or output	<Ctrl+Shift+3>

Entering operands (LAD/FBD/GRAPH)

Function	Selected object	Shortcut key
Activate the input field for the first operand of the instruction	Instruction	<Return> Or <Any letters/numbers> An empty input field opens on <Return>, any letters or numbers will be entered in the input field.
Activate input field for the operand	Operand	<F2>
Delete operand	Operand	<Del>
Define tag	Operand	<Ctrl+Shift+I>
Rewire tag	Operand	<Ctrl+Shift+P>
Rename tag	Operand	<Ctrl+Shift+T>
Entering operands	Input field for operands	<Any letters/numbers>
Confirm entry of the operand	Input field for operands	<Return>
Open autocompletion	Input field for operands	<Ctrl+I>
Discard current change	Input field for operands	<Esc> The input field is deactivated and the previous contents restored.

Process instructions (STL/SCL)

Function	Selected object	Shortcut key
Indent line (SCL only)	Line	<Tab> or <Ctrl+R>
Outdent line (SCL only)	Line	<Shift+Tab> or <Ctrl+Shift+R>
Automatically formatting selected text (SCL only)		<Ctrl+Shift+W>
Open "Call options" dialog	Cursor after block call	<Return>
Define tag	Operand	<Ctrl+Shift+I>
Rewire tag	Operand	<CtrlShift+P>
Rename tag	Operand	<Ctrl+Shift+T>
Expand/collapse parameter list (SCL only)	Operand	<Ctrl+Shift+Space>
Expand/collapse code section	Insertion mark within the code section	<Ctrl+Shift+Num+> <Ctrl+Shift+Num->
Expand/collapse all code sections	Any	<Ctrl+Shift+Num*> <Ctrl+Shift+Num/>
Open autocompletion	Any	<Ctrl+I> or <Ctrl+Spacebar>
Set/delete bookmarks		<Ctrl+Shift+M>
To the next bookmark		<Ctrl+Shift+6>
To the previous bookmark		<Ctrl+Shift+5>
Comment out selection	Line	<Ctrl+Shift+Y>
Remove comment	Line	<Ctrl+Shift+U>

Programming window of GRAPH

Function	Area	Shortcut key
Page Up/Down	Navigation view, single step view, sequence view, permanent instructions	<Page Up>/ <Page Down>
Navigate in the navigation view	Navigation view	<Up arrow> <Down arrow>
Expand/collapse object	Navigation view	<+> or <Right arrow> <-> or <Left arrow>
Switch between single step view and sequence view when a step or a transition is selected	Navigation view	<Return>
Switch between navigation view and work area	Navigation view, single step view, sequence view, permanent instructions	<ALT+F6>

Function	Area	Shortcut key
Go to first element in a network	Single step view	<Home>
Go to last element in a network	Single step view	<End>
Switch to interlock	Single step view	<Ctrl+Home>
Switch to transition	Single step view	<Ctrl+End>
Navigate in the structure	Sequence view	Arrow keys
Go to first step	Sequence view	<Home> or <Ctrl+Home>
Go to last step	Sequence view	<End> or <Ctrl+End>
Open branch	Sequence view	<Shift+F8>
Close branch	Sequence view	<Shift+F9>
Insert sequence end	Sequence view	<Shift+F7>
Insert jump	Sequence view	<Shift+F12>
Insert step and transition	Sequence view	<Shift+F5>
Delete element	Sequence view	<Del>
Go to first editable element	Permanent instructions	<Home>
Go to next editable element	Permanent instructions	<Tab>
Go to last editable element	Permanent instructions	<End>
Go to previous editable element	Permanent instructions	<Shift+Tab>
Jump to the start of the "Action" cell	Actions	<Home>
Jump to the end of the "Action" cell	Actions	<End>
Insert new action	Actions	<Return>

Monitor program

Function	Shortcut key
Set/remove breakpoint (STL, SCL)	<Ctrl+Shift+F9>
Step over breakpoint (STL, SCL)	<Ctrl+Shift+F10>
Jump to lower-level block (STL, SCL)	<Ctrl+Shift+F11>
Return to calling block (STL, SCL)	<Ctrl+Shift+F12>
Run program up to marker (cursor position) (STL, SCL)	<Ctrl+F3>
Display program status (SCL, STL)	<Ctrl+T>
Enable all breakpoints (STL, SCL)	<Ctrl+Shift+F2>
Disable all breakpoints (STL, SCL)	<Ctrl+Shift+F3>
Modify to 0 (LAD, FBD)	<Ctrl+Shift+9>
Modify to 1 (LAD, FBD)	<Ctrl+Shift+1>
Modify operand (LAD, FBD)	<Ctrl+Shift+2>

See also

Keyboard operation in the TIA Portal (Page 308)
Using project-related functions (Page 310)
Arranging windows (Page 310)
Editing tables (Page 316)
Text editing (Page 315)

Enlarging the programming window

Introduction

The programming window is relatively small when all components of the application are shown. If the program code is large, you may find you have to rearrange the work area constantly. To avoid this problem, you can hide or minimize the display of the following components of the application and of the program editor:

- Project tree
- Task cards
- Block interface
- Favorites
- Comments
- Networks

Note

You can also use the "Reduce automatically" option for the task cards, project tree, and Inspector window. These windows will then be minimized automatically when you do not need them.

See also: Maximizing and minimizing the work area

Hiding and showing the project tree

The project tree allows you to access all areas of the project. You can hide the project tree while you are creating a program so you have more space for the programming window.

To show and hide the project tree, follow these steps:

1. To hide the project tree, deselect the "Project tree" check box in the "View" menu, or click on "Collapse" on the project tree title bar.
2. To show the project tree, select the "Project tree" check box in the "View" menu or click on "Extend" on the project tree title bar.

Opening and closing task cards

The task cards are located at the right-hand edge of the programming window.

To open or close the task cards, follow these steps:

1. To close the task cards, deselect the "Task card" check box in the "View" menu or click "Collapse" on the task cards title bar.
2. To open the task cards, select the "Task card" check box in the "View" menu or click "Expand" on the task cards title bar.

Hiding and showing the block interface

The block interface is shown in the upper section of the program editor. During programming you can show and hide it as required.

To show and hide the block interface, follow these steps:

1. In the lower part of the interface within the window splitter, click on the Up arrow or Down arrow.

Showing and hiding favorites

To hide or show the favorites in the program editor, follow these steps:

1. Click the "Display favorites in the editor" button in the program editor toolbar.

Showing and hiding comments

Within a block you can enter a comment for the block or for each network. These two types of comments are shown and hidden differently.

To show or hide a block comment, follow these steps:

1. Click the the triangle at the start of the line with the block title.

To show or hide network comments, follow these steps:

1. Click "Network comments on/off" on the program editor toolbar.

Note

The comments available can vary depending on the programming language used.

Opening and closing networks

Some programming languages use networks. You can open or close these networks as required.

To open or close networks, follow these steps:

1. If you want to open a network, click the right arrow in front of the network title. If you want to close a network, click the down arrow in front of the network title.

To open and close all networks, follow these steps:

1. Click "Open all networks" or "Close all networks" in the program editor toolbar.

Note

Networks are not used in every programming language.

See also

Overview of the program editor (Page 2523)

Maximizing and minimizing the work area (Page 287)

Setting the mnemonics

You can program blocks using German or international mnemonics. If you open the TIA portal for the first the international mnemonics is set as default. You can change the mnemonics at any time.

Procedure

To set the mnemonics, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "General" group in the area navigation.
3. In the "General settings" group, select the mnemonics that you want to use.
   The mnemonics is changed in all blocks.

Displaying symbolic and absolute addresses

You have the following options for displaying operands in the program editor:

- Symbolic representation
  The symbolic operands are displayed in the program. The corresponding absolute addresses are shown in tooltips if you hold the mouse pointer over the operand.
- Absolute representation
  The absolute addresses are displayed in the program. The corresponding symbolic operands are displayed in tooltips.
- Symbolic and absolute representation
  Symbolic operands and absolute addresses are displayed in program.

Requirement

The program editor is open.

Procedure

To change the representation of the operands, follow these steps:

1. Click the "Absolute/symbolic operands" button in the program editor toolbar.
   Each time you click the button, the representation and the symbol on the button change.

Or:

1. Click the small arrow next to the "Absolute/symbolic operands" button in the program editor toolbar.
   A drop-down list is displayed.
2. Select the required representation from the drop-down list.
   The symbol on the button changes.

See also

Basic information about operands (Page 2197)

Use instruction versions

Basic information on instruction versions

The instructions available to you for programming the user program are managed in system libraries. If a new version of a system library is installed by an update, the newer versions of the instructions of this system library may also be installed.

If there are several versions for an instruction, these are listed in the "Instructions" task card after the respective instruction. If the instruction versions are not shown, you can show them via the "Show column headers and additional columns" button in the toolbar of the "Instructions" task card. You can then select the versions of the instructions to be used in the program from the dropdown-list box of the "Version" column. If you do not select any versions, the most recent versions are used.

Note

Note the following information:

- You can only ever use the same version of an instruction within a device.
- If you change the version of an instruction that other instructions depend on, the versions of the dependent instructions are also changed.
- If you select a version for an instruction that can not be run on the CPU used, the instruction is shaded out. This means that you cannot use this instruction in this version with your CPU.
- When you change the version of an instruction, you must compile the block before the new version number is displayed in the properties of the instruction.
- If an instruction is a block within the system, the block also receives its own block version number. This version number can be different from the instruction version and can vary on different CPU types.

Changes in the versions

New versions can be main versions or secondary versions. New versions, such as 2.0 or 3.0, have more substantial changes to them. New main versions may therefore result in changes to the block interface. New secondary versions, such as 1.3 or 1.4, contain lesser changes or remedies to errors.

Using instruction versions

You can decide within a device which version of an instruction you want to use. If you select another version for an instruction, the new version is specified for all locations of use of this instruction within your program. These instructions are identified in the program by a red frame. You must then download your program to the device to use the new instruction version.

Using instruction profiles

Basics of instruction profiles

Introduction

The TIA Portal provides you with numerous instructions that you can use to program the user program. However, you may filter out specific instructions that you do not want to use. To this end you can create instruction profiles in which you can explicitly specify the instructions to be listed in the "Instructions" task card. However, although you may create several instruction profiles in a project, only one of these profiles may be active at any given time. You can exchange instruction profiles with other users by means of shared libraries.

Note

Please note the following:

- The use of instructions that are not allowed in the active profile in a block will trigger the output of a block compilation error. Such a situation may be triggered if you drag-and-drop a block from the library to your program.
- Instructions of a profile that are not supported by the currently installed products are deleted from the profile the next time it is edited. If you transfer this profile to an engineering system in which these instructions are supported by the installed products, the instructions are again present in the profile but they are disabled. You can enable these instructions as required at any time.
- If you want to make changes to the active profile, you must recompile the blocks in the project. This is also necessary when you disable or delete the active profile or when you enable a profile.

See also

Creating new instruction profiles (Page 2536)

Opening and editing instruction profiles (Page 2537)

Activating and deactivating instruction profiles (Page 2538)

Deleting instruction profiles (Page 2539)

Creating new instruction profiles

Requirement

The "Common data > Instruction profiles" folder is open in the project navigation.

Procedure

Proceed as follows to create a new instruction profile:

1. Double-click the "Add new profile" command.
   The Instruction Profile Editor opens and displays the new instruction profile. All instructions are activated for the new instruction profile.
2. Edit the new instruction profile to suit your requirements.

If necessary, you can rename the new instruction profile. To do this, follow these steps:

1. Right-click on the new instruction profile.
2. Select the "Rename" command in the shortcut menu.
3. Enter a name for the new instruction profile.

Note

The first instruction profile that you create will be used as active profile. In this case, compile all blocks in the project. If other instruction profiles are already available you must explicitly activate the new one in order to use it as active profile. You can identify the active profile by its icon in the project navigation.

See also

Basics of instruction profiles (Page 2535)

Opening and editing instruction profiles (Page 2537)

Activating and deactivating instruction profiles (Page 2538)

Deleting instruction profiles (Page 2539)

Opening and editing instruction profiles

Once you have opened an instruction profile, you can edit it as follows:

- Activating and deactivating instructions
  You can explicitly specify the instructions to be allowed in the instruction profile.

  Note

  Note that dependencies exist between some instructions. As a result, it is possible that several instructions may be activated or deactivated by an action. The check box icon indicates the folders in which instructions are deactivated.

- Activating and deactivating instruction versions
  Certain instructions are available with different versions. You can explicitly specify the instruction versions to be allowed in the instruction profile.
- Renumbering blocks
  An instruction representing an internal function block (FB) or function (FC) in the system is assigned a specific block number by the system. You can replace this block number with your own block number. Within a version, there are several implementations for certain instructions. The block numbers in such instructions can only be changed for the specific implementation.

  Note

  If an instruction from the instruction profile is used in the program and the specified block number is already in use by a different block, the specified block number of the instruction will be replaced by a free block number.

Requirement

The "Common data > Instruction profiles" folder is open in the project navigation.

Opening instruction profiles

Proceed as follows to open an instruction profile:

1. Double-click the instruction profile that you want to edit.
   The instruction profile opens in the Instruction Profile Editor.

Editing instruction profiles

Proceed as follows to edit an instruction profile in the Instruction Profile Editor:

1. Select the device that you want to edit from the "Device family" drop-down list box.
2. Select the programming language for which you want to edit the instruction profile from the "Language" drop-down list box.
3. Deactivate the instructions or instruction versions that you want to exclude from the instruction profile. You can deactivate a folder to deactivate all subordinate instructions.

4. Activate the instructions or instruction versions that you want to allow in the instruction profile.
5. You may assign your own block numbers.

Note

You can assign the number up to 65535 for CPUs of the S7-1200/1500 series. For CPUs of the S7300/400 series you find the restrictions of the number ranges in the respective CPU manual.

Note

A new compilation process is required for all blocks in the project when you change the active profile.

See also

Basics of instruction profiles (Page 2535)

Creating new instruction profiles (Page 2536)

Activating and deactivating instruction profiles (Page 2538)

Deleting instruction profiles (Page 2539)

Use instruction versions (Page 2534)

Activating and deactivating instruction profiles

You first need to activate an instruction profile in order to include filtering of its instructions. You can always deactivate the instruction profile to reset the instructions task card to the default scope of instructions.

Note

A new compilation process is required for all blocks in the project.

Requirement

The "Common data > Instruction profiles" folder is open in the project navigation.

Activating instruction profiles

Proceed as follows to activate an instruction profile:

1. Right-click on the instruction profile that you want to activate.
2. Select the "Activate profile" command from the shortcut menu.
   The selected instruction profile is now active. Instructions can now only be used in accordance with the settings of this profile.

Deactivating instruction profiles

Proceed as follows to deactivate the instruction profile:

1. Right-click on the instruction profile that you want to deactivate.
2. Select the "Deactivate profile" command from the shortcut menu.
   No instruction profile is active and the "Instructions" task card once again shows all instructions that are available for use.

See also

Basics of instruction profiles (Page 2535)

Creating new instruction profiles (Page 2536)

Opening and editing instruction profiles (Page 2537)

Deleting instruction profiles (Page 2539)

Deleting instruction profiles

Requirement

The "Common data > Instruction profiles" folder is open in the project tree.

Procedure

Proceed as follows to delete an instruction profile:

1. Right-click on the instruction profile that you want to delete.
2. Select the "Delete" command in the shortcut menu.

Note

A new compilation process is required for all blocks in the project when you delete the active profile.

Result

The selected instruction profile is deleted. If you deleted the active instruction profile, no more active profiles are available and the "Instructions" task card once again shows all instructions that are available for use.

See also

Basics of instruction profiles (Page 2535)

Creating new instruction profiles (Page 2536)

Opening and editing instruction profiles (Page 2537)

Activating and deactivating instruction profiles (Page 2538)

Using autocompletion

Basics of autocompletion

Function

You can use autocompletion in the program window of the program editor as an easy way to access available tags or instructions during programming. Autocompletion means a context-specific list appears in a dialog from which you can select the tags or instructions you need.

See also

Using autocompletion in graphic programming languages (Page 2540)

Using autocompletion in textual programming languages (Page 2541)

Using autocompletion in graphic programming languages

Inserting tags using autocompletion

To insert tags in graphic programming languages using autocompletion, follow these steps:

1. Select an operand of the instruction to which you wish to assign a tag.
   The input field for the operand opens. The autocompletion button will appear beside the input field.
2. Either click the autocompletion button or use the shortcut <Ctrl+I>.
   Autocompletion opens. It contains only the local and global tags, data blocks and multiple instances which are admissible for the operand in the given context. You can exit autocompletion at any time by pressing <Esc>.

3. Select the required tag from the list. If necessary, you can also filter the list:
   - For example, enter the first few letters of the name of the tag or instruction you wish to insert. Autocompletion will be filtered further with each letter entered. If there is no tag or instruction starting with the letters entered, autocompletion will remain at the last match.
   - Enter # to access the local tags from the block interface.
   - Enter " to access the global tags.
   - Enter % to access absolute addresses.

   If the tag is a structured tag, a data block or a multiple instance, then an arrow is displayed at the end of the row. Click on the arrow to display the lower-level elements. You can navigate to the very last level in this way. If a structure is allowed as a data type for the operand, you can choose "None" from the list. This assigns the entire structure to the operand as a tag. Use the <Backspace> key to return to the previous level.
4. Press the <Return> key to apply the tag.

See also

Basics of autocompletion (Page 2540)

Using autocompletion in textual programming languages (Page 2541)

Using autocompletion in textual programming languages

Inserting tags and instructions using autocompletion

To insert tags and instructions in textual programming languages using autocompletion, follow these steps:

1. Enter the first few letters of the name of the tag or instruction you wish to insert. If necessary, you can directly filter the kind of tags:
   - Enter # to access the local tags from the block interface.
   - Enter " to access the global tags.
   - Enter % to access absolute addresses.

   Autocompletion opens. It contains only the local and global tags, data blocks, multiple instances and instructions which are admissible at the current position. You can exit autocompletion at any time by pressing <Esc>.
2. Enter more letters of the name of the tag or instruction you wish to insert. You can use <Enter> or <Tab> to apply the tag or instruction and close autocompletion. Autocompletion will be filtered further with each letter entered. If there is no tag or instruction starting with the letters entered, autocompletion only contains the previous matches.

3. Select the tag or instruction required from the list.
   If a tag is a structured tag, a data block or a multiple instance, first select the tag, the data block or multiple instance from the autocompletion and apply the selection with <Enter>. To select the additional components of the structure, data block, or multiple instance, enter a period. Autocompletion then reopens and you can select the next component.
4. Press the <Return> key to apply the tag.

See also

Basics of autocompletion (Page 2540)

Using autocompletion in graphic programming languages (Page 2540)

General settings for the PLC programming

Overview of the general settings

Overview

The following table shows the general settings that you can make:

Group	Setting	Description
View	With comments	Network comments are shown.
	Tag information	Additional information for the tags used is displayed in the program editor. When you select the option "Tag information with hierarchy", the comments of the higher structure levels are also displayed for structured tags.
Compilation	Delete actual parameters on interface update	Actual parameters are deleted if the associated formal parameters were deleted in the called block, and you run the "Update block call" function or compile the block.
Default settings for new blocks	IEC check	The compatibility of operands in comparison operations and arithmetic operations are tested according to IEC rules. You have to explicitly convert non-compatible operands.
Additional settings	Set network title automatically	Sets the title of a network based on the comment of the output parameter of the first writing instruction in the network. See also: Inserting network title (Page 2575)
	Show autocompletion list	The autocompletion list is displayed.
	Mnemonics	German or international designation of operations and operands

Group	Setting	Description
Download without reinitialization	Memory reserve	Defines the size of reserve in volatile memory that can be used for interface extensions.
Block interface / data blocks	Set "HMI accessible" for new elements and ARRAY data blocks	Enables the option "HMI accessible" for new tags in the block interface and in data blocks. The use of this option makes sense especially when working with large amounts of data in ARRAY data blocks. See also: Basic principles for programming of data blocks (Page 2825)

See also

Layout of the block interface (Page 2544)

Changing the settings (Page 2543)

Permissible addresses and data types of PLC tags (Page 2476)

Overview of the print settings (Page 276)

Basics of block access (Page 2175)

Setting and canceling the IEC check (Page 2374)

Using autocompletion (Page 2540)

Changing the settings

Procedure

To change the settings, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. Change the settings.

Result

The change will be loaded directly, there is no need to save it explicitly.

See also

Overview of the general settings (Page 2542)

9.1.4.2 Programming code blocks

Declaring the block interface

Layout of the block interface

Introduction

The interface contains the declarations of local tags and constants that are used within the block. The tags are subdivided into two groups:

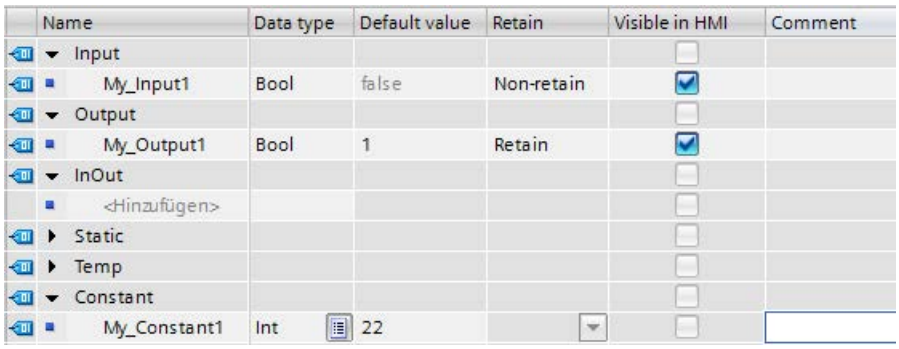
- Block parameters that form the block interface when it is called in the program.
- Local data that are used for storage of intermediate results.

You use the tag declaration to define the call interface of a block in the program and the names and data types of tags and constants that you want to use in the block.

The interface of function blocks also defines the structure of the instances that are assigned to the function block.

Layout of the block interface

The following figure shows the structure of the block interface. The number of columns and sections varies depending on the type of block.



Block parameters

The following table shows the types of block parameters:

Type	Section	Function	Available in
Input parameters	Input	Parameters whose values are read by the block.	Functions, function blocks and some types of organization blocks
Output parameters	Output	Parameters whose values are written by the block.	Functions and function blocks

Type	Section	Function	Available in
In/out parameters	InOut	Parameters whose values are read by the block when it is called, and whose values are written again by the block after execution.	Functions and function blocks
Return value	Return	Value that is returned to the calling block.	Functions

Depending on the type of block opened, additional sections may be displayed.

Local data

The following table shows the types of local data:

Type	Section	Function	Available in
Temporary local data	Temp	Tags that are used to store temporary intermediate results. Temporary local data are retained for only one cycle. If you use temporary local data, you have to make sure that the values are written within the cycle in which you want to read them. Otherwise the values will be random.	Functions, function blocks and organization blocks Note: Temporary local data is not shown in instance data blocks.
Static local data	Static	Tags that are used for storage of static intermediate results in the instance data block. Static data is retained until overwritten, which may be after several cycles. The names of the blocks, which are called in this code block as multiple instance, will also be stored in the static local data.	Function blocks
Constant	Constant	Constants with declared symbolic names that are used within the block.	Functions, function blocks and organization blocks Note: Local constants are not shown in instance data blocks.

Meaning of the columns

The following table shows the meaning of the individual columns. You can show or hide the columns as required. The number of columns displayed varies depending on the CPU series and the type of the open object.

Column	Description
	Symbol you can click on to drag-and-drop an element to a program for use as an operand.
Name	Name of the element.
Data type	Data type of the element.
Offset	Relative address of a tag. The column is only visible in blocks with standard access.

Column	Description
Default value	Value with which you can pre-assign specific tags in the interface of the code block, or the value of a local constant. Specification of the default value is optional for tags. If you do not specify any value the predefined value for the indicated data type is used. For example, the value "false" is predefined for BOOL. The default value of a tag is applied as the start value in the corresponding instance data block. You can replace these values with instance-specific start values in the instance data block. Constants always have the default value declared in the block interface. They are not shown in instance data blocks and cannot be assigned instance-specific values there.
Retentivity	Marks a tag as retentive. The values of retentive tags are retained even after the power supply is switched off. This column is only visible in the interface of the function block with optimized access.
Visible in HMI	Shows whether a tag is visible by default in the HMI selection list.
Accessible from HMI	Shows whether HMI can access this tag during runtime.
Setting value	Marks a tag as a setpoint. Setting values are the values that will probably have to be fine tuned during commissioning. The column is only available in the interface of function blocks.
Comment	Comments on documentation of the element.

See also

Using tags within the program (Page 2203)

Keywords (Page 2197)

Parameter transfer at block call (Page 2185)

Valid data types in the block interface (Page 2549)

Setting the retentivity of local tags (Page 2561)

Setting data values during commissioning (Page 2850)

Basics of constants (Page 2204)

Rules for declaring the block interface

General rules for declaring the block interface

Using POINTER

The following rules apply to the use of block parameters within the block:

- Input parameters may only be read.
- Output parameters may only be written.
- In/out parameters may be read and written.

Assigning default values to block parameters

You can assign default values to specific parameters in the function block interface. The possibility of the assignment depends on the declaration subsection and data type of the parameter.

The following table shows which parameters can be assigned a default value:

Parameter type	Section	Assignment of a default value is possible		
		Elementary data types	Structured data types	Parameter types
Input parameters	Input	X	X	-
Output parameters	Output	X	X	-
In/out parameters	InOut	X	- (1)	-
Static local data	Static	X	X	-
Temporary local data	Temp	-	-	-
Constants	Constant	X	-	-

(1) Exception: In blocks with optimized access, you have the option to use PLC data types as default values under certain conditions.

See also: Declaring PLC data types (UDT) as actual parameters for in-out parameters. (Page 2564)

See also

Using tags within the program (Page 2203)

Keywords (Page 2197)

Valid data types in the block interface

Valid data types in the block interface in S7-300/400

The following table shows which data types you can assign to the parameters in the individual sections of the interface.

Section	Standard Data types	ARRAY STRUCT STRING DT	Parameter types	VOID	POINTER	ANY
Organization block						
Temp	X	X	-	-	-	X
Constant	X	X(3)	-	-	-	-
Function block						
Input	X	X	X	-	X	X
Output	X	X	-	-	-	-
InOut	X	X (1)	-	-	X	X
Static	X	X	-	-	-	-
Temp	X	X	-	-	-	X
Constant	X	X(3)	-	-	-	-
Function						
Input	X	X (1)	X	-	X	X
Output	X	X (1)	-	-	X	X
InOut	X	X (1)	-	-	X	X
Temp	X	X	-	-	-	X
Return	X	X	-	X	X	X (2)
Constant	X	X(3)	-	-	-	-

(1) STRING can only be defined in the standard length of 254 characters.
(2) In SCL, ANY is not permitted as a function value.
(3) Constants with the ARRAY or STRUCT data type are not permitted.

See also

Parameter transfer at block call (Page 2185)

Data types (Page 2236)

Valid data types in the block interface

Valid data types in the block interface in S7-1200

The following table shows which data types you can assign in the parameters in the individual sections of the interface.

Section	Standard Data types	ARRAY STRUCT STRING DT	VOID	VARIANT
Organization block				
Temp	X	X	-	X
Constant	X	X(2)	-	-
Function block				
Input	X	X	-	X
Output	X	X	-	-
InOut	X	X(1)	-	X
Static	X	X	-	-
Temp	X	X	-	X
Constant	X	X(2)	-	-
Function				
Input	X	X(1)	-	X
Output	X	X(1)	-	X
InOut	X	X(1)	-	X
Temp	X	X	-	X
Return	X	X	X	-
Constant	X	X(2)	-	-
(1) STRING can only be defined in the standard length of 254 characters. (2) Constants with the ARRAY or STRUCT data type are not permitted.				

Valid data types in the block interface in S7-1500

The following table shows which data types you can assign in the parameters in the individual sections of the interface.

Section	Standard Data types	ARRAY STRUCT STRING / WSTRING DT	Parameter types	VOID	DB_ANY	POINTER	ANY	VARIANT
Organization block								
Temp	X	X	-(4)	-	X	-	X(3)	X
Constant	X	X(5)	-	-	-	-	-	-

Section	Standard Data types	ARRAY STRUCT STRING / WSTRING DT	Parameter types	VOID	DB_ANY	POINTER	ANY	VARIANT
Function block								
Input	X	X	X	-	X	X	X	X
Output	X	X	-	-	X	-	-	-
InOut	X	X(1)	-(4)	-	X	X	X	X
Static	X	X	-	-	X	-	-	-
Temp	X	X	-(4)	-	-	-	X(3)	X
Constant	X	X(5)	-	-	-	-	-	-
Function								
Input	X	X(1)	X	-	X	X	X	X
Output	X	X(1)	-	-	X	X	X	X
InOut	X	X(1)	-(4)	-	X	X	X	X
Temp	X	X	-(4)	-	X	-	X(3)	X
Return	X	X	-	X	X	X	x(2)	-
Constant	X	X(5)	-	-	-	-	-	-

(1) You cannot make length declarations for STRING and WSTRING in these sections. Here, STRINGs always have the standard length 254, WSTRINGs the standard length 16832. A declaration in the format MyString[3] would not be permitted. WSTRING is only permitted in blocks with optimized access in these sections.

(2) In SCL, ANY is not permitted as a function value.

(3) ANY can only be used in blocks with standard access in the "Temp" section.

(4) The "INSTANCE" parameter type is the only exception permissible in the "TEMP" and "InOut" sections.

(5) Constants with the ARRAY or STRUCT data type are not permitted.

See also

Parameter transfer at block call (Page 2185)

Data types (Page 2236)

Declaring local tags and constants

Declaring local tags and constants in the block interface

Requirement

The block interface is open.

Procedure

To declare a tag or constant of the elementary data type, follow these steps:

1. Select the appropriate declaration section in the interface.
2. Enter a name for the element in the "Name" column.
3. Enter the required data type in the "Data type" column. You will be supported by autocompletion during input.
4. For constants, enter a value in the "Default value" column.
5. Optional: Change the properties that are displayed in the other columns of the block interface.

Result

The element is created.

Syntax check

A syntax check is performed after each entry, and any errors found are displayed in red. You do not have to correct these errors immediately - you can continue editing and make any corrections later. However, you will not be able to compile the program if the tag declaration contains syntax errors.

Note

If you change the interface of a block, the calls of the block in the program will possibly become inconsistent. The call locations are automatically updated, if possible.

If an automatic updating is not possible, the inconsistent blocks have to be updated manually.

See also:

Updating block calls in LAD (Page 2588)

Updating block calls in FBD (Page 2630)

See also

Editing tables (Page 316)

Editing the block interface (Page 2565)

Basic information on start values (Page 2836)

Using tags within the program (Page 2203)

Keywords (Page 2197)

Properties of local tags and constants (Page 2560)

Setting the retentivity of local tags (Page 2561)

Rules for declaring the block interface (Page 2547)

Constants (Page 2204)

Declaring a local tag in the program editor

Requirement

The program editor is open.

Procedure

To declare a local tag, follow these steps:

1. Insert an instruction in your program.
   The "<???>", "<??.?>" or "..." strings represent operand placeholders.
2. Replace an operand placeholder with the name of the tag to be created.
3. Select the name of the element.
   If you want to declare multiple elements, select the names of all the elements to be declared.
4. Select the "Define tag" command in the shortcut menu.
   The "Define tag" dialog box opens. It displays a declaration table in which the name of the element is already entered.
5. To declare a local tag, select one of the following sections:
   - Local In
   - Local Out
   - Local InOut
   - Local Static
   - Local Temp
6. In the other columns, enter data type and comments.
7. Click the "Define" button to complete your entry.

Result

The declaration is written directly into the block interface and is valid within the entire block.

Note

If you change the interface of a block, the calls of the block in the program will possibly become inconsistent. The call locations are automatically updated, if possible.

If an automatic updating is not possible, the inconsistent blocks have to be updated manually.

See also:

Updating block calls in LAD (Page 2588)

Updating block calls in FBD (Page 2630)

See also

Editing tables (Page 316)
Using tags within the program (Page 2203)
Keywords (Page 2197)
Editing the block interface (Page 2565)
Basic information on start values (Page 2836)
Properties of local tags and constants (Page 2560)
Setting the retentivity of local tags (Page 2561)
Rules for declaring the block interface (Page 2547)

Declaring tags of the ARRAY data type

Requirement

The block interface is open.

Procedure

To declare a tag of the ARRAY data type, follow these steps:

1. Select the appropriate declaration section in the interface.
2. Enter a tag name in the "Name" column.
3. In the "Data type" column, click the button for the data type selection.
   A list of the permissible data types is opened.
4. Select the "Array" data type.
   The "Array" dialog opens.
5. In the "Data type" text box, specify the data type of the array elements.
6. In the "ARRAY limits" text box, specify the high and low limit for each dimension.
   Example of a one-dimensional ARRAY:
   `[0..3]`
   Example of a three-dimensional ARRAY:
   `[0..3, 0..15, 0..33]`
7. Confirm your entry.
8. Optional: Change the properties of the tags that are displayed in the other columns of the block interface.

Result

The tag of ARRAY data type is created.

Note

You cannot define specific default values for ARRAY elements. However, you can assign them start values in the instance.

See also

Using autocompletion (Page 2540)
Rules for declaring the block interface (Page 2547)
Array (Page 2267)
Using tags within the program (Page 2203)
Keywords (Page 2197)
Properties of local tags and constants (Page 2560)
Setting the retentivity of local tags (Page 2561)
Editing tables (Page 316)

Declaring tags of STRUCT data type

Requirement

The block interface is open.

Procedure

To declare a tag of the STRUCT data type, follow these steps:

1. Select the appropriate declaration section in the interface:
2. Enter a tag name in the "Name" column.
3. Enter "Struct" in the "Data type" column. You will be supported by autocompletion during input.
   An empty, indented row is inserted after the new tag.
4. Insert the first structural element in the first empty row.
   An additional empty row is inserted after the element.
5. Select a data type for the structure element.
6. Optional: Change the properties of the structural element that is displayed in the other columns of the block interface.

7. Repeat the step 4 to 7 for all additional structure elements.
   It is not necessary to end the structure explicitly. The structure ends with the last element that is entered.
8. To insert a new tag after the structure, leave a blank row after the end of the structure and then start the new tag in the second empty row.

Result

The tag of STRUCT data type is created.

See also

Using autocompletion (Page 2540)

Rules for declaring the block interface (Page 2547)

Using tags within the program (Page 2203)

Keywords (Page 2197)

Properties of local tags and constants (Page 2560)

Setting the retentivity of local tags (Page 2561)

Editing tables (Page 316)

Declaring tags based on a PLC data type

Requirement

A PLC data type is declared in the current CPU.

Procedure

To declare a tag based on a PLC data type, follow these steps:

1. Select the appropriate declaration section in the interface:
2. Enter the PLC data type in the "Data type" column. You will be supported by Autocomplete during input.

Result

The tag is created.

Note

You define the default values of tags within a PLC data type when the PLC data type is created. You cannot change these values at the point of use of the PLC data type.

If you change or delete PLC data types that are used in the block interface, the interface becomes inconsistent. To remedy this inconsistency, the interface has to be updated.

See also: Updating the block interface (Page 2557)

See also

Editing tables (Page 316)

Basics of PLC data types (Page 2854)

Declaring higher-level tags

Introduction

To access data areas within a declared tag, you can overlay the declared tags with an additional declaration. This provides you with the option of addressing an already declared tag with a different data type. You can, for example, address the individual bits of a tag of WORD data type with an ARRAY of BOOL.

Overlaying tags

To overlay a tag with a new data type, follow these steps:

1. Open the block interface.
2. In the interface, select the tag that you want to overlay with a new data type.
3. Click "Add row" in the toolbar.
   A row is inserted after the tag to be overlaid. The overlaying tag must be declared in the row directly after the tag that is to be overlaid.
4. Enter a tag name in the "Name" column.
5. Enter the "AT" entry in the "Data type" column. You will be supported by Autocomplete in this step.
   The following is added to the entry in the "Name" column.
   `"AT<Name of the higher-level tag>"`
6. Click the data type selection button again and select the data type for the new tag.
   The tag is created. It points to the same data as the higher-level tag, however interprets this data with the new data type.

Removing overlay

To remove the overlay of a tag, follow these steps:

1. Select the overlaid tag that you want to remove.
2. Select the "Delete" command in the shortcut menu.
3. The overlay is removed.

See also

Editing tables (Page 316)
Overlaying tags with AT (Page 2218)

Declaring multi-instances

Requirement

- The function block to be called exists in project tree and is multi-instance capable.
- The block interface of the calling function block is open.

Procedure

To declare a function block to be called as a multi-instance, follow these steps:

1. In the "Name" column of the "Static" section, enter a designation for the block call.
2. In the "Data type" column, enter the symbolic name for the function block to be called.

Note

The program editor will declare the multi-instance automatically if you program a block call in a network and then specify in the "Call options" dialog that you want to call the block as a multiple instance.

See also

Updating the block interface (Page 2557)

Updating the block interface

Introduction

If you change or delete PLC data types or multiple instances that are used in the block interface, the interface will become inconsistent. To remedy this inconsistency, the interface has to be updated.

You have two options for updating the block interface:

- Explicit updating of the block interface.
  The used PLC data types and multiple instances will be updated. The instance data blocks that belong to the block are not implicitly updated during this process.
- Implicit updating during compilation.
  All used PLC data types and multiple instances as well as the related instance data blocks will be updated.

Explicit updating of the block interface

To explicitly update the block interface, follow these steps:

1. Open the block interface.
2. Select the "Update" command in the shortcut menu.

Implicit Updating during Compilation

Proceed as follows to implicitly update all uses of PLC data types and multiple instances as well as the instance data blocks during compilation:

1. Open the project tree.
2. Select the "Program blocks" folder.
3. Select the command "Compile > Software (rebuild all blocks)" in the shortcut menu.

See also

Basics of PLC data types (Page 2854)

Declaring tags based on a PLC data type (Page 2555)

Editing tables (Page 316)

Editing the block interface (Page 2565)

Basic information on start values (Page 2836)

Using tags within the program (Page 2203)

Keywords (Page 2197)

Properties of local tags and constants (Page 2560)

Setting the retentivity of local tags (Page 2561)

Updating block calls in LAD (Page 2588)

Declaring multi-instances (Page 2557)

Rules for declaring the block interface (Page 2547)

Extending the block interface

Description

In order to enable the editing of PLC programs that have already been commissioned and that are running without error on a system, CPUs of the S7-1500 series and most CPUs of the S7-1200 V4 series support the option of extending the interfaces of function blocks during runtime.

You can download the modified blocks without setting the CPU to STOP and without affecting the values of already loaded tags.

This is a simple means of implementing program changes. This load process (download without reinitialization) will not have a negative impact on the controlled process.

Principle of operation

Each function block is always assigned a default memory reserve. The memory reserve is not used initially. Activate the memory reserve if you decide on loading interface changes after having compiled and downloaded the block. All tags that you subsequently declare will be saved to the memory reserve. The subsequent download does not influence any tags that are already loaded or have a negative impact on runtime.

If you decide to review your program at a later time while the plant is not in operation, you are also provided an option of reworking the memory layout of individual or several blocks in a single pass. With this action, you move all tags from the reserve area to the regular area. The memory reserve is now cleared and made available for further interface extensions.

Requirements

This "Download without reinitialization" function is available if the following requirements are met:

- The project is in the "TIA Portal V12" format or a higher version.
- You are working with a CPU that supports "Download without reinitialization".
- The blocks were created in LAD, FBD, STL, or SCL.
- The blocks were created by the user, i.e. they are not included with the blocks delivered in your package.
- These blocks are assigned the optimized access attribute.

Basic steps

Perform the following steps if you want to extend the interface of a function block and then load the block without re-initialization.

1. All blocks have a default memory reserve of 100 bytes. You can adapt this memory reserve to suit your requirements.
2. Activate the memory reserve.
3. Extend the block interface.

4. Compile the block.
5. Download the block to the CPU as usual.

For more information on the various steps, refer to chapter "Loading blocks (S7-1200/1500) ".

Note

The full scope of the "Download without reinitialization" function is only available on CPUs of the S7-1500 and S7-1200 V4 series.

However, all CPU families support the option of extending the interface of function blocks and downloading newly declared tags without repercussion:

- You may add new tags in the "Temp" section and download these without influencing the process.
- You may create new tags of a structured data type in the "InOut" section and download these without influencing the process.

Editing the properties of local tags and constants

Properties of local tags and constants

Properties

The table below provides an overview of the properties of local tags and constants:

Group	Property	Description
General	Name	Name of the element.
	Data type	Data type of the element.
	Default value	Value with which you can pre-assign specific tags in the interface of the code block, or the value of a local constant. Specification of the default value is optional for tags. If you do not specify any value the predefined value for the indicated data type is used. For example, the value "false" is predefined for BOOL. The default value of a tag is applied as the start value in the corresponding instance. You can then replace these adopted values with instance-specific start values.
	Comment	Comment for the element.
Attributes	Retain	Marks the tag as retentive. The values of retentive tags are retained even after the power supply is switched off. This attribute is only available in the interface of the function block with optimized access.

Group	Property	Description
	Accessible from HMI	Indicates whether the tag can be used in HMI. When the attribute is set, you have read or write access to the tag from the HMI. When the attribute is not set, you cannot access the tag from the HMI. Please note, however, that you cannot implement general access protection for the tag with the "Accessible from HMI" attribute. Read or write access from other applications is possible even if the attribute is not enabled.
	Visible in HMI	Shows whether the tag is visible by default in the HMI selection list.
	Hidden parameter	Indicates whether the tag should be hidden for the block call. This is only possible if you have specified a valid predefined actual parameter beforehand.
	Predefined actual parameter	Defines a parameter that is to be used as actual parameter during the block call.
	Configurable	Indicates whether a parameter is configurable in CFC.
	For test	Indicates whether a parameter is registered for the CFC test mode.
	Visible	Indicates whether a parameter is visible in CFC.
	Interconnectable	Indicates whether a parameter is interconnectable in CFC.
	Enable tag readback	Indicates whether a parameter is relevant for the "Read back chart" function in CFC.
	Enumeration texts	Assigns a parameter to an enumeration in CFC.
	Engineering unit	Assigns a parameter to a unit in CFC.
	Low limit	Defines the low limit for the parameter in CFC.
	High limit	Defines the high limit for the parameter in CFC.

See also

Setting the retentivity of local tags (Page 2561)

Changing properties of local tags and constants (Page 2562)

Keywords (Page 2197)

Setting the retentivity of local tags

Introduction

Function blocks store their data in an instance. To prevent data loss in the event of power failure, you can mark the data as retentive. This data is stored in a retentive memory area. The option of setting the retentivity depends on the set access type of the function block.

Retentive behavior in blocks with standard access

In blocks with standard access you cannot set the retentive behavior of individual tags. You can only define them as retentive in the assigned instance. All tags contained in the block are then considered as retentive.

Retentivity for optimized block access

In data blocks with optimized access you can define the retentive behavior of individual tags.

For structured data type tags, the retentivity setting always applies to the entire structure. You can make no individual retentivity setting for individual elements within the structure.

You cannot create retentive tags of the structured data type in the "InOut" section. In/out parameters with structured data type, for example ARRAY, STRUCT, or STRING, are always non-retentive.

The following settings are available:

- Retentive
  The values of the tags or the structure are available even after a power failure.
- Non-retentive
  The values of the tags or the structure are lost in the event of a power failure.
- Set in IDB
  The retentivity can be set in the instance data block. The setting that is made in the instance data block than applies, however, centrally to all tags that are selected with "Set in IDB".

See also

Properties of local tags and constants (Page 2560)

Basics of block access (Page 2175)

Changing properties of local tags and constants

Editing properties of an element in the block interface

To edit the properties of an element in the block interface, follow these steps:

1. Open the block interface.
2. Select the required element in the table.
3. Change the entries in the columns.

Editing properties of several elements in the block interface

You can also simultaneously set or reset the "Retain", "Visible in HMI", "Accessible in HMI" and "Setpoint" columns for one or more selected elements.

To change one of these properties for several elements, follow these steps:

1. Open the block interface.
2. Hold down the CTRL key.
3. In the required column, select each of the table cells whose value you want to change.
4. Select the "Set <property>" or "Reset <property>" command in the shortcut menu.

Editing properties in the properties window

To edit the properties of an individual tag or constant, follow these steps:

1. Select an element in the table.
   The properties of the element are shown in the Inspector window.
2. Change the entries in the inspector window.

Renaming tags directly in the program editor

To rename one or more elements, follow these steps:

1. Select one or more elements in the program.
2. Select the "Rename tag" command in the shortcut menu.
   The "Rename tag" dialog opens. It displays a declaration table with the selected elements.
3. Change the entries in the "Name" column.
4. Confirm the input by clicking the "Change" button.

Editing the data type or comment in the program editor

Proceed as follows to edit the data type or tag comment in the program editor:

1. Select the name of the tag.
2. Select the "Rewire tag" command in the shortcut menu.
   The "Rewire tag" dialog will open. The dialog shows a declaration table.
3. Change the entry in the "Data type" or "Comment" columns.
4. Click the "Change" button to confirm the input.

Effect in the program

In case of a change of the name, data type or address of a tag or constant, each location of use of the tag is automatically updated in the program.

Note

If you change the interface of a block, the program may become inconsistent. The inconsistencies are automatically updated, if possible.

If an automatic updating is not possible, the inconsistent calls are marked in red. You than have to manually updated the inconsistencies.

See also:

Updating block calls in LAD (Page 2588)

Updating block calls in FBD (Page 2630)

See also

Layout of the block interface (Page 2544)

Editing tables (Page 316)

Editing the block interface (Page 2565)

Properties of local tags and constants (Page 2560)

Setting the retentivity of local tags (Page 2561)

Basic information on start values (Page 2836)

Using tags within the program (Page 2203)

Keywords (Page 2197)

Updating the block interface (Page 2557)

Rules for declaring the block interface (Page 2547)

Declaring PLC data types (UDT) as actual parameters for in-out parameters.

Use of PLC data types as predefined actual parameters

In blocks with optimized access, you have the option to use PLC data types as actual parameters for in/out parameters (InOut) under certain conditions. This can be useful if you are using program blocks as library elements and want to store information about the actual parameters to be used along with the library element.

Additionally, you have the option of hiding in/out parameters that have a vaild predefined actual parameter when the block is called. Hidden parameters are not visible initially when the block is called but can be displayed via a small arrow at the bottom edge of the box.

Requirements

- The in/out parameter (InOut) is based on a PLC data type or a system data type.
- The in/out parameter has the retentivity setting "Retain" or "Non-retain". For in/out parameters with setting "Set in IDB", it is not possible to predefine actual parameters.
- The program block is a block with optimized access.

Procedure

To predefine the actual parameter of an in/out parameter, follow these steps:

1. Open the block interface.
2. Select an in/out parameter (InOut) in the block interface.
3. Open the "Properties" tab in the Inspector window.
4. Select the "Attributes" group in the area navigation.
5. Enter the required actual parameter in the "Predefined actual parameter" input box.
6. Select the "Hidden parameter" check box (optional).

Result

- An actual parameter is predefined. If you save the program block as a library element, it thus also contains information about the actual parameter to be used.
- If the library element is used in the program, a check is made to determine whether the actual parameter you have predefined can be addressed. If so, it is automatically used as the actual parameter.
- If the actual parameter is not found in the program, a syntax error is signaled. The parameter is not hidden and the parameter must be initialized manually.

Editing the block interface

Inserting table rows

Procedure

Proceed as follows to insert a row above the selected row:

1. Select the row in front of which you want to insert a new row.
2. Click the "Insert row" button on the toolbar of the table.

Result

A new row is inserted above the selected row.

See also

Editing tables (Page 316)

Inserting table rows

Procedure

Proceed as follows to insert a row below the selected row:

1. Select the row below which you want to insert a new row.
2. Click the "Add row" button on the table toolbar.

Result

A new empty row will be inserted below the selected row.

See also

Editing tables (Page 316)

Deleting tags

Procedure

Follow the steps below to delete elements:

1. Select the row with the element to be deleted. You can also select several rows by clicking on them one after the other while holding down the <Ctrl> key or by pressing and holding down <Shift> and clicking on the first and last row.
2. Select the "Delete" command in the shortcut menu.

See also

Editing tables (Page 316)

Automatically filling in successive cells

You can load the contents of one or several table cells into the cells below, automatically filling in the successive cells.

If you automatically fill in cells in the "Name" column, a consecutive number will be appended to each name. For example, "Motor" will become "Motor_1".

You can define individual or more cells as well as entire rows as source area.

If less rows exist in the open table than you want to fill, then you will first have to insert additional empty rows.

Requirement

- The table is open.
- Sufficient declaration rows are available.

Procedure

To automatically fill in successive cells, follow these steps:

1. Select the cells to be loaded.
2. Click the "Fill" symbol in the bottom right corner of the cell.
   The mouse pointer is transformed into a crosshair.
3. Keep the mouse button pressed and drag the mouse pointer downwards over the cells that you want to fill in automatically.
4. Release the mouse button.
   The cells are filled in automatically.
5. If entries are already present in the cells that are to be automatically filled in, a dialog appears. In this dialog you can indicate whether you want to overwrite the existing entries or insert new rows for the new tags.

See also

Editing tables (Page 316)

Show and hide table columns

You can show or hide the columns in a table as needed.

Procedure

To show or hide table columns, follow these steps:

1. Click a column header.
2. Select the "Show/Hide" command in the shortcut menu.
   The selection of available columns is displayed.
3. To show a column, select the column's check box.
4. To hide a column, clear the column's check box.

Editing tags with external editors

To edit individual tags in external table editors, such as Excel, you can export or import these tags using copy and paste. However, you cannot copy structured tags to an editor.

Requirements

The block interface and an external editor are opened.

Procedure

To export individual tags to an external editor and import them again, follow these steps:

1. Select one or more tags.
2. Select "Copy" in the shortcut menu.
3. Switch to the external editor and paste the copied tags.
4. Edit the tags as required.
5. Copy the tags in the external editor.
6. Select the tags in the external editor.
7. Switch back to the block interface.
8. Select "Paste" in the shortcut menu.

Creating program code

Creating LAD programs

Basic information on LAD

LAD programming language

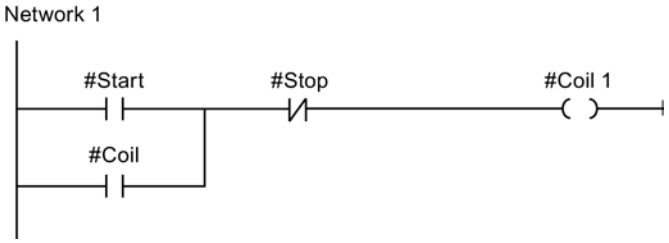
Overview of the Ladder Logic (LAD) programming language

LAD is a graphical programming language. The representation is based on circuit diagrams.

The program is mapped in one or more networks. A network contains a power rail on the left where the rungs originate. The binary signal scans are arranged in the form of contacts on the rungs. The serial arrangement of the elements on a rung creates a series connection; arrangement on simultaneous branches creates a parallel connection. Complex functions are represented by boxes.

Example of networks in LAD

The following figure shows a LAD network with two normally open contacts, one normally closed contact and one coil:



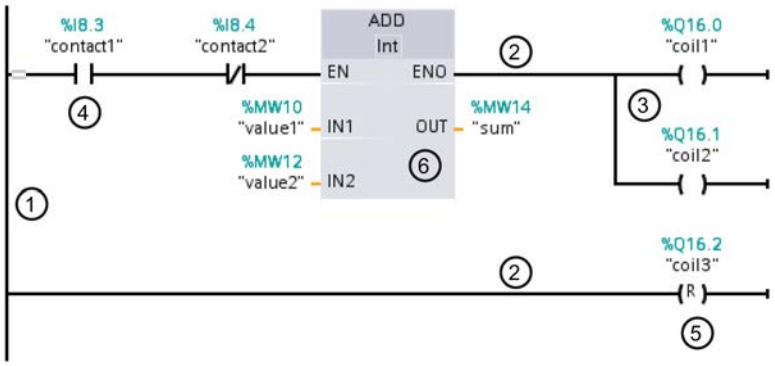
Overview of the LAD elements

LAD elements

A LAD program consists of separate elements that you can arrange in series or parallel on the power rail of a network. Most program elements must be supplied with tags.

There is at least one rung from the power rail. Network programming starts at the left edge of the rung. You can expand the power rail by several rungs and branches.

For example, the following figure shows elements of a LAD network:



1) Power rail

2) Rung

3) Branch

4) Contact

5) Coil

6) Box

Power rail

Each LAD network consists of a power rail that contains at least one rung. A network can be extended by adding additional rungs. You can use branches to program parallel connections in the specific rungs.

Contacts

You can use contacts to create or interrupt a current-carrying connection between two elements. The current is relayed from left to right. You can use contacts to query the signal state or the value of an operand and control it depending on the result of the current flow.

The following types of contact are available to you in a LAD program:

- Normally open contact:
  Normally open contacts forward the current if the signal state of a specified binary operand is "1".
- Normally closed contacts:
  Normally closed contacts forward the current if the signal state of a specified binary operand is "0".
- Contact with additional function:
  Contacts with additional function forward the current if a specific condition is met. With these contacts you can also execute an additional function, such as an RLO edge detection and a comparison.

Coils

You can use coils to control binary operands. Coils can set or reset a binary operand depending on the signal state of the result of logic operation.

The following types of coils are available to you in a LAD program:

- Standard coils:
  Standard coils set a binary operand if current flows in the coil. The "Assignment" instruction is an example of a standard coil.
- Coils with additional function:
  These coils have additional functions in addition to the evaluation of the logic operation result. Coils for RLO edge detection and program control are examples of coils with additional function.

Boxes

Boxes are LAD elements with complex functions. The empty box is an exception. You can use the empty box as a placeholder in which you can select the required instruction.

The following types of boxes are available to you in a LAD program:

- Boxes without EN/ENO mechanism:
  A box is executed depending on the signal state at the box inputs. The error status of the processing cannot be queried.
- Boxes with EN/ENO mechanism:
  A box is only executed if the enable input "EN" carries the signal state "1". If the box is processed correctly, the "ENO" enable output has signal state "1". If an error occurs during the processing, the "ENO" enable output is reset.

Calls of code block are also shown in the network as boxes with EN/ENO mechanism.

See also

Rules for the use of LAD elements (Page 2579)

EN/ENO mechanism (Page 2447)

Settings for LAD

Overview of the settings for LAD

Overview

The following table shows the settings that you can make:

Group	Setting	Description
Font	Font size	Font size in program editor
View	Layout	Compact or wide Changes the vertical spacing between operands and other objects (such as operand and contact). The change becomes visible once the block is reopened.
	With absolute information	Additional display of the absolute addresses
Operand field	Maximum width	Maximum number of characters that can be entered horizontally in the operand field. This setting recalculates the layout of the networks.
	Maximum height	Maximum number of characters that can be entered vertically in the operand field. This setting recalculates the layout of the networks.

See also

Changing the settings (Page 2571)

Changing the settings

Procedure

To change the settings, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. Change the settings.

Result

The change will be loaded directly, there is no need to save it explicitly.

See also

Overview of the settings for LAD (Page 2571)

Working with networks

Using networks

Function

The user program is created in the block within networks. For a code block to be programmed, it must contain at least one network. To achieve a better overview of the user program, you can also subdivide your program into several networks.

See also

Inserting network title (Page 2575)

Entering a network comment (Page 2577)

Navigating networks (Page 2578)

Inserting networks

Requirement

A block is open.

Procedure

To insert a new network, follow these steps:

1. Select the network after which you want to insert a new network.
2. Select the "Insert network" command in the shortcut menu.

Result

A new empty network is inserted into the block.

See also

Selecting networks (Page 2573)

Copying and pasting networks (Page 2573)

Deleting networks (Page 2574)

Expanding and collapsing networks (Page 2575)

Inserting network title (Page 2575)

Entering a network comment (Page 2577)

Navigating networks (Page 2578)

Selecting networks

Requirements

A network is available.

Selecting a network

To select a network, follow these steps:

1. Click the title bar of the network that you want to select.

Selecting several networks

Proceed as follows to select several individual networks:

1. Press and hold down the <Ctrl> key.
2. Click all the networks that you want to select.

To select several successive networks, follow these steps:

1. Press and hold down the <Shift> key.
2. Click the first network that you want to select.
3. Click the last network that you want to select.
   The first and last networks and all those in between are selected.

See also

Inserting networks (Page 2572)

Copying and pasting networks (Page 2573)

Deleting networks (Page 2574)

Expanding and collapsing networks (Page 2575)

Inserting network title (Page 2575)

Entering a network comment (Page 2577)

Navigating networks (Page 2578)

Copying and pasting networks

Copied networks can be pasted within the block or in another block. Networks that were created in LAD or FBD can also be inserted in blocks of the respective other programming language.

Requirement

A network is available.

Procedure

To copy and paste a network, follow these steps:

1. Select the network or networks to be copied.
2. Select "Copy" in the shortcut menu.
3. Select the network after which you want to paste in the copied network.
4. Select "Paste" in the shortcut menu.

See also

Inserting networks (Page 2572)
Selecting networks (Page 2573)
Deleting networks (Page 2574)
Expanding and collapsing networks (Page 2575)
Inserting network title (Page 2575)
Entering a network comment (Page 2577)
Navigating networks (Page 2578)

Deleting networks

Requirement

A network is available.

Procedure

To delete a network, follow these steps:

1. Select the network that you want to delete.
2. Select the "Delete" command in the shortcut menu.

See also

Inserting networks (Page 2572)
Selecting networks (Page 2573)
Copying and pasting networks (Page 2573)
Expanding and collapsing networks (Page 2575)
Inserting network title (Page 2575)

Entering a network comment (Page 2577)
Navigating networks (Page 2578)

Expanding and collapsing networks

Requirements

A network is available.

Opening and closing a network

To open a network, follow these steps:

1. Click on the right arrow in the network title bar.

To close a network, follow these steps:

1. Click on the down arrow in the network title bar.

Opening and closing all networks

To open and close all networks, follow these steps:

1. In the toolbar, click "Open all networks" or "Close all networks".

See also

Inserting networks (Page 2572)
Selecting networks (Page 2573)
Copying and pasting networks (Page 2573)
Deleting networks (Page 2574)
Inserting network title (Page 2575)
Entering a network comment (Page 2577)
Navigating networks (Page 2578)

Inserting network title

The network title is the header of a network. The length of the network title is limited to one line. You can enter the title manually or set it automatically. When you set it automatically, you can do this for individual networks or use the settings to specify that the network title is always set automatically.

For automatic insertion of the network title, the comment of the operand in one of the following instructions in the network is evaluated:

- Assignment
- Set output
- Reset output

The instruction that is listed first in the network is used.

The network title is only inserted automatically when the following conditions are fulfilled:

- The network does not have a title yet.
- The operand of the instruction used for the comment has a comment.

Note

Note the following restrictions for automatic insertion of the network title:

- The network title is not adapted if you change the comment of the operand at a later time.
- The network title is not adapted if you change the operand of the instruction.
- The network title is only set by the writing instructions listed above.
- If the operand is of the data type array, the comment of the array is used and not the comments of the array elements.
- Comments of invalid operands are not taken into consideration.

Entering the network title manually

To enter a network title, follow these steps:

1. Click on the title bar of the network.
2. Enter the network title.

Setting the network title automatically

To specify that the network titles are always set automatically, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. Select the "Set network title automatically" check box in the "Additional settings" group.
   The network titles are set automatically as of this time if the conditions listed above are fulfilled.

To set an individual network title automatically, follow these steps:

1. Right-click "Network <Number of the network>" in the title bar of a network.
2. Select the "Set network title automatically" command in the shortcut menu.
   The title of the selected network is set based on the comment of the operand if the conditions listed above are fulfilled.

See also

Using networks (Page 2572)
Inserting networks (Page 2572)
Selecting networks (Page 2573)
Copying and pasting networks (Page 2573)
Deleting networks (Page 2574)
Expanding and collapsing networks (Page 2575)
Entering a network comment (Page 2577)
Navigating networks (Page 2578)

Entering a network comment

You can use network comments to provide comments on the program contents of individual networks. For example, you can indicate the function of the network or draw attention to special characteristics.

Requirement

A network is available.

Procedure

To enter a network comment, follow these steps:

1. Click on the right arrow before the network title.
2. If the comment area is not visible, click "Network comments on/off" in the toolbar.
   The comment area is displayed.
3. Click "Comment" in the comment area.
   The "Comment" text passage is selected.
4. Enter the network comment.

See also

Using networks (Page 2572)
Inserting networks (Page 2572)
Selecting networks (Page 2573)
Copying and pasting networks (Page 2573)
Deleting networks (Page 2574)
Expanding and collapsing networks (Page 2575)

Inserting network title (Page 2575)

Navigating networks (Page 2578)

Navigating networks

You can navigate straight to a specific position within a block.

Procedure

To navigate to a specific position within a block, follow these steps:

1. Right-click in the code area of the programming window.
2. Select the "Go to > Network/line" command in the shortcut menu.
   The "Go to" dialog will open.
3. Enter the network to which you want to navigate.
4. Enter the line number of the network to which you want to navigate.
5. Confirm your entry with "OK".

Result

The relevant line will be displayed if this is possible. If the network or line requested does not exist, the last existing network or the last existing line in the network requested will be displayed.

See also

Using networks (Page 2572)

Inserting networks (Page 2572)

Selecting networks (Page 2573)

Copying and pasting networks (Page 2573)

Deleting networks (Page 2574)

Expanding and collapsing networks (Page 2575)

Inserting network title (Page 2575)

Entering a network comment (Page 2577)

Inserting LAD elements

Rules for the use of LAD elements

Rules

Note the following rules when inserting LAD elements:

- Every LAD network must terminate with a coil or a box. However, the following LAD elements must not be used to terminate a network:
  - Comparator boxes
  - Instructions for positive and negative RLO edge detection
- The starting point of the branch for a box connection must always be the power rail. Logic operations or other boxes can be present in the branch before the box.
- Only contacts can be inserted into simultaneous branches with preceding logic operations. The contact for negating the result of logic operation (-|NOT|-) is an exception here. The contact for negating the result of logic operation, as well as coils and boxes, can be used in simultaneous branches if they originate directly from the power rail.
- Constants (e.g. TRUE or FALSE) cannot be assigned to normally closed or normally open contacts. Instead, use operands of the BOOL data type.
- Only one jump instruction can be inserted in each network.
- Only one jump label can be inserted in each network.
- Instructions with positive or negative edge detection may not be arranged directly at the left margin of the rung as they requires a prior logic operation.

Placement rules for S7-1200/1500 CPUs

The following table sets out the instructions that can only be positioned at the end of the network:

Instruction		Preceding logic operation required
Mnemonics	Name	
SET_BF	Set bit field	No
RESET_BF	Reset bit field	No
JMP	Jump if RLO = 1	No
JMPN	Jump if RLO = 0	Yes
JMP_LIST	Define jump list	No
SWITCH	Jump distributor	No
RET	Return	No

Placement rules for S7-300/400 CPUs

The following table sets out the instructions that can only be positioned at the end of the network:

Instruction		Preceding logic operation required
Mnemonics	Name	
S	Set output	Yes
R	Reset output	Yes
SP	Start pulse timer	Yes
SE	Start extended pulse timer	Yes
SD	Start on-delay timer	Yes
SS	Start retentive on-delay timer	Yes
SF	Start off-delay timer	Yes
SC	Set counter value	Yes
CU	Count up	Yes
CD	Count down	Yes
JMP	Jump if RLO = 1	No
JMPN	Jump if RLO = 0	Yes
RET	Return	No
OPN	Open global data block	No
OPNI	Open instance data block	No
CALL	Call block	No
SAVE	Save RLO in BR bit	No
MCRA	Enable MCR range	No
MCRD	Disable MCR range	No
MCR<	Open MCR ranges	No
MCR>	Close MCR ranges	No

See also

EN/ENO mechanism (Page 2447)

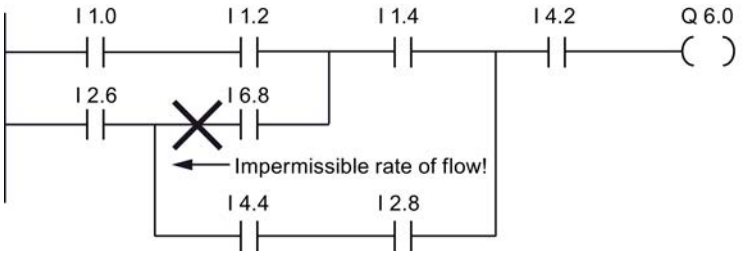
Prohibited interconnections in LAD (Page 2581)

Overview of the LAD elements (Page 2569)

Prohibited interconnections in LAD

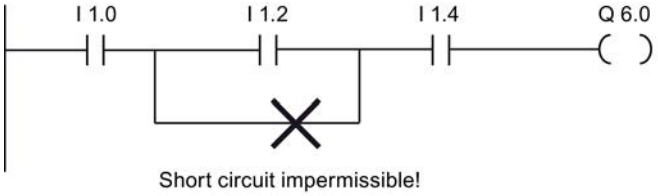
Power flow from right to left

No branches can be programmed that could result in a power flow in the reverse direction.



Short-circuit

No branches may be programmed that would cause a short-circuit.



Logic operations

The following rules apply to logic operations:

- Only Boolean inputs can be combined with preceding logic operations.
- Only the first Boolean output can be combined with a further logic operation.
- Only one complete logical path can exist per network. Paths that are not connected can be linked.

See also

Rules for the use of LAD elements (Page 2579)

Inserting LAD elements using the "Instructions" task card

Requirement

A network is available.

Procedure

To insert a LAD element into a network using the "Instructions" task card, follow these steps:

1. Open the "Instructions" task card.
2. Navigate to the LAD element that you want to insert.
3. Use drag-and-drop to move the element to the desired place in the network.
   If the element is an internal system function block (FB), the "Call options" dialog opens. In this dialog you can create an instance data block of the single-instance or multiple-instance type for the function block in which the data of the inserted element will be saved. You will find the new instance data block created in the project tree in the "Program resources" folder under "Program blocks > System blocks". If you have selected "multiple instance", these are located in the block interface in the "Static" section.

Or:

1. Select the point in the network at which you want to insert the element.
2. Open the "Instructions" task card.
3. Double-click on the element you want to insert.
   If the element is an internal system function block (FB), the "Call options" dialog opens. In this dialog you can create an instance data block of the single-instance or multiple-instance type for the function block in which the data of the inserted element will be saved. You will find the new instance data block created in the project tree in the "Program resources" folder under "Program blocks > System blocks". If you have selected "multiple instance", these are located in the block interface in the "Static" section.

Result

The selected LAD element is inserted with placeholders for the parameters.

Inserting LAD elements using an empty box

Requirement

A network is available.

Procedure

To insert an LAD element into a network using an empty box, follow these steps:

1. Open the "Instructions" task card.
2. Navigate to "General > Empty box" in the "Basic instructions" palette.
3. Use a drag-and-drop operation to move the "Empty box" element to the desired place in the network.

4. Position the cursor over the triangle in the top right-hand corner of the empty box.
   A drop-down list is displayed.
5. Select the required LAD element from the drop-down list.
   If the element is an internal system function block (FB), the "Call option" dialog opens. In this dialog you can create an instance data block of the single-instance or multiple-instance type for the function block in which the data of the inserted element will be saved. You will find the new instance data block created in the project tree in the "Program resources" folder under "Program blocks > System blocks". If you have selected "multi-instance", these are located in the block interface in the "Static" section.

Result

The empty box is changed to the respective LAD element. Placeholders are inserted for the parameters.

Selecting the data type of a LAD element

Selecting a data type

Introduction

Some instructions can be executed with several different data types. If you use one of these instructions in the program, you have to specify a valid data type for the instruction at the specific point in the program. For some instructions, you have to select the data types for the inputs and outputs separately.

Note

The valid data type (BOOL) for the tags on the enable input EN and the enable output ENO is predefined by the system and cannot be changed.

The valid data types for an instruction are listed in the instruction drop-down list. You specify the data type of the instruction by selecting an entry from the drop-down list. If the data type of an operand differs from the data type of the instruction and cannot be converted implicitly, the operand is displayed in red and a rollout with the corresponding error message appears.

Data type selection of mathematical instructions

Some mathematical instructions provide you with the option of having the data type automatically set corresponding to the data types of the operand. In the drop-down list for data type selection, these instructions have the entry "Auto" in addition to the actual data types. If you select this entry and then allocate the first operand, the data type of the operand is selected as data type for the instruction. The entry in the drop-down list changes to "Auto (<Data type>)",

e.g. "Auto (Real)". If you allocate additional operands, the automatically set data type of the instruction is adjusted according to the following criteria:

- You supply all other operands with tags of the same data type:
  The data type of the instruction is not changed.
- You supply all other operands with tags whose data type is smaller than the data type of the instruction:
  The data type of the instruction is not changed. For the operand with the smaller data type, an implicit conversion is conducted if necessary.
- You supply an additional operand with a tag whose data type is greater than the data type of the instruction:
  The data type of the instruction is changed to the larger data type. An implicit conversion is performed, if necessary, for operands that deviate from the newly set data type of the instruction.

Each change in the data type of an operand can result in a change of the data type of the instruction. Other operands may possibly be implicitly converted as a result. Operands for which an implicit conversion is performed are marked with a gray square.

Note

Please also observe the information on data type conversion for your device and, in particular, the notes on the IEC check.

See also: Data type conversion (Page 2286)

See also

Defining the data type of an instruction (Page 2584)

Defining the data type of an instruction

Introduction

Some instructions can be executed with several different data types. When you insert such instructions into your program, you must specify the data type for these instructions at the actual point in the program.

Specifying the data type by means of the drop-down list

To define the data type of an instruction using the drop-down list, follow these steps:

1. Insert the instruction at the required point in the program using drag-and-drop.
   The entry "???" (undefined) is displayed in the drop-down list of the inserted instruction.
2. Click the triangle in the upper corner of the drop-down list.
   The drop-down list will open to display the data types valid for the instruction.

3. Select a data type from the drop-down list.
   The selected data type is displayed.
4. If the instruction has two drop-down lists, select the data type for the instruction inputs in the left-hand drop-down list and the data type for the instruction outputs in the right-hand drop-down list.

Specifying data type by assigning tags

To define the data type of an instruction by assigning tags, follow these steps:

1. Insert the instruction at the required point in the program using drag-and-drop.
   The entry "???" (undefined) is displayed in the drop-down list of the inserted instruction.
2. At an input or output, specify a valid tag, the data type of which is to be applied as the instruction data type.
   The data type of the tag is displayed in the drop-down list.
3. Enter a valid tag at an input and a valid tag at an output if data types need to be defined for both the inputs and outputs of the instruction. The tag specified at the input determines the data type of the inputs; the tag specified at the output determines the data type of the outputs of the instruction.

Automatically specifying the data type of mathematical instructions

To automatically specify the data type for mathematical instructions, follow these steps:

1. Insert the mathematical instruction at the required point in the program using drag-and-drop.
   The entry "???" (undefined) is displayed in the drop-down list of the inserted instruction.
2. Select the "Auto" entry from the drop-down list.
3. Enter a valid tag at an input or output.
   The data type of the tag is applied as data type of the instruction. The entry in the drop-down list changes to "Auto (<Data type>)".

See also: Selecting a data type (Page 2583)

See also

Selecting a data type (Page 2583)

Using favorites in LAD

Adding LAD elements to Favorites

Requirement

- A block is open.
- The multipane mode is set for the "Instructions" task card or the Favorites are also displayed in the editor.

Procedure

To add SCL instructions to the Favorites, follow these steps:

1. Open the "Instructions" task card.
2. Maximize the "Basic instructions" pane.
3. Navigate in the "Basic instructions" pane to the instruction that you want to add to the Favorites.
4. Drag-and-drop the instruction into the "Favorites" pane or into the Favorites area in the program editor.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Removing LAD elements from Favorites (Page 2587)

Overview of the program editor (Page 2523)

Inserting LAD elements using favorites

Requirement

- A block is open.
- Favorites are available.

Procedure

To insert an instruction into a program using Favorites, follow these steps:

1. Drag-and-drop the desired instruction from Favorites to the desired position.

Or:

1. Select the position in the program where you want to insert the instruction.
2. In the Favorites, click on the instruction you want to insert.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Removing LAD elements from Favorites (Page 2587)
Overview of the program editor (Page 2523)

Removing LAD elements from Favorites

Requirement

A code block is open.

Procedure

To remove instructions from Favorites, follow these steps:

1. Right-click on the instruction you want to remove.
2. Select the "Remove instruction" command in the shortcut menu.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Adding LAD elements to Favorites (Page 2585)
Inserting LAD elements using favorites (Page 2586)
Overview of the program editor (Page 2523)

Insert block calls in LAD

Inserting block calls using a drag-and-drop operation

You can insert calls for existing functions (FC) and function blocks (FB) using a drag-and-drop operation from the project tree. If you call function blocks from other function blocks, you can either call them as single-instance or multi-instance blocks. If a function block is called as single instance, it will store its data in a data block of its own. If a function block is called as multi-instance, it will store its data in the instance data block of the calling function block.

Requirement

- A network is available.
- The block that is to be called is available.

Inserting a call of a function (FC)

To insert a call of a function (FC) into a network using a drag-and-drop operation, follow these steps:

1. Drag the function from the project tree to the required network.

Inserting a call for a function block (FB)

To insert a call for a function block (FB), follow these steps:

1. Drag the function block from the project tree to the required network.
   The "Call options" dialog opens.
2. Enter in the dialog whether you wish to call the block as single or multi-instance.
   - If you click on the "Single instance" button, you will have to enter a name in the "Name" text box for the data block that you want to assign to the function block.
   - If you click on the "Multi-instance" button, you will have to enter the name of the tag in the "Name in the interface" text box; this is the name that you use to enter the called function block as a static tag in the interface of the calling block.
3. Confirm your entries with "OK".

Result

The function or the function block is inserted with its parameters. You can then assign the parameters.

See also: Parameter transfer at block call (Page 2185)

Note

If when calling a function block you specify an instance data block that does not exist, it will be created. If you have called a function block as a multi-instance, this will be entered as a static tag in the interface.

See also

Updating block calls in LAD (Page 2588)

Changing the instance type (Page 2589)

Single instances (Page 2183)

Multi-instances (Page 2183)

Block calls (Page 2180)

Updating block calls in LAD

If interface parameters of a called block are changed, the block call can no longer be executed correctly. You can avoid such inconsistent block calls by updating the block calls.

You have two options to update the block calls:

- Explicit updating in the program editor.
  The block calls in the open block will be updated.
- Implicit updating during compilation.
  All block calls in the program as well as the used PLC data types will be updated.

Update blocks in the program editor

To update a block call within a block, follow these steps:

1. Open the block in the program editor.
2. Click "Update inconsistent block calls" in the toolbar.

Or:

1. Open the block in the program editor.
2. Right-click on the instruction with the block call.
3. Select the "Update" command in the shortcut menu.
   The "Interface update" dialog opens. This dialog shows the differences between the block interface in use and the changed interface of the called block.
4. If you want to update the block call, click "OK". To cancel the update, click "Cancel".

Update block calls during compilation

Follow these steps to update all block calls and uses of PLC data types during compilation implicitly:

1. Open the project tree.
2. Select the "Program blocks" folder.
3. Select the command "Compile > Software (rebuild all blocks)" in the shortcut menu.

See also

Block calls (Page 2180)

Inserting block calls using a drag-and-drop operation (Page 2587)

Changing the instance type (Page 2589)

Changing the instance type

Instance type

There are two ways of calling function blocks:

- As a single instance
- As a multiple instance

See also: Call function blocks as single or multi-instances (Page 2182)

You can modify a defined instance type at any time.

Requirement

The user program contains a block call.

Procedure

To change the instance type of a function block, follow these steps:

1. Open the code block and select the block call.
2. Select the "Change instance" command in the shortcut menu.
   The "Call options" dialog opens.
3. Click the "Single instance" or "Multi instance" button.
   - If you select the "Single instance" instance type, enter a name for the data block that is to be assigned to the function block.
   - If you select "Multiple instance" as the instance type, enter in the "Name in the interface" text field the name of the tag with which the called function block is to be entered as a static tag in the interface of the calling block.
4. Confirm your entries with "OK".

Note

The previous single and multiple instances will not be deleted automatically.

See also

Block calls (Page 2180)

Inserting block calls using a drag-and-drop operation (Page 2587)

Updating block calls in LAD (Page 2588)

Inserting complex LAD instructions

Using the "Calculate" instruction

Requirement

A network is available.

Procedure

Proceed as follows to use the "Calculate" instruction:

1. Open the "Instructions" task card.
2. Navigate to "Math functions > CALCULATE" in the "Basic instructions" pane.
3. Use drag-and-drop to move the element to the desired place in the network.
   The instruction "Calculate" will be inserted for the data type with a placeholder expression and question mark.
4. Enter the data type for the calculation.
5. Enter the operands for the calculation.

   Note

   The calculation is run with the inputs of the "Calculate" instruction. If you want to use constants you must also insert appropriate inputs for them.

6. Click on the "Edit 'Calculate' instruction" button to replace the placeholder expression with the correct expression.
   The "Edit 'Calculate' instruction" dialog will open.
7. Enter the required expression in the "OUT:= " text box.

   Note

   In the "Example" area you can find an example of a valid expression and possible instructions that you can use.

   To determine a value with the help of Pythagoras' theorem, for example, enter "OUT := SQRT (SQR (IN1) + SQR (IN2))".

8. Confirm your entry with "OK".

See also

CALCULATE: Calculate (Page 3264)

Using free-form comments

Basic information on using free-form comments in LAD

Introduction

Free-form comments allow you to add comments to the source code for graphic programming languages similar to line comments for textual languages.

Free-form comments can be used for the following elements:

- Boxes
- Coils

See also

Inserting free-form comments (Page 2592)
Editing free-form comments (Page 2592)
Deleting free-form comments (Page 2594)

Inserting free-form comments

Requirement

A network with instructions is available.

Procedure

To insert a free comment on an instruction, proceed as follows:

1. If necessary, activate the "Free-form comments on/off" button in the toolbar.
2. Right-click on the instruction for which you want to insert a free-form comment.
3. Select the "Insert comment" command in the shortcut menu.
   A comment box with a standard comment opens. The comment box is connected by an arrow to the corresponding instruction.
4. Enter the required comment in the comment box.

See also

Basic information on using free-form comments in LAD (Page 2591)
Editing free-form comments (Page 2592)
Deleting free-form comments (Page 2594)

Editing free-form comments

Introduction

Free-form comments can be edited as follows:

- Changing the comment text
- Changing the position and size of the comment box
- Attaching a comment to another element
- Showing and hiding free comments

Changing the comment text

To change the text of free-form comments, follow these steps:

1. Click on the comment box.
2. Enter the desired text.

Changing the position of the comment box

To change the positioning of the comment box, follow the steps below:

1. Left-click the comment box and keep the mouse button pressed.
2. Drag the comment box to the desired location.

Changing the size of the comment box

To change the size of the comment box, follow the steps below:

1. Click on the comment box.
2. Drag the comment box on the move handle in the lower right corner to the desired size.

Attaching a comment to another element

To attach a free-form comment to another element, follow these steps:

1. Left-click the point of the arrow that links the comment box with the instruction and keep the mouse button pressed.
2. Drag the arrow to the element to which you want to attach the comment. Possible insertion points are marked with a green square.
3. Release the mouse button.

Showing and hiding free comments

To show or hide a free-form comments, follow these steps:

1. Click the "Free-form comment on/off" button in the toolbar.

See also

Basic information on using free-form comments in LAD (Page 2591)

Inserting free-form comments (Page 2592)

Deleting free-form comments (Page 2594)

Deleting free-form comments

Procedure

To delete a free-form comment, proceed as follows:

1. Right-click on the free-form comment that you want to delete.
2. Select the "Delete" command in the shortcut menu.

See also

Basic information on using free-form comments in LAD (Page 2591)
Inserting free-form comments (Page 2592)
Editing free-form comments (Page 2592)

Editing LAD elements

Selecting LAD elements

You can select several individual elements or all elements in a network.

Requirement

LAD elements are available

Selecting several individual LAD elements

To select several individual LAD elements, follow these steps:

1. Press and hold down the <Ctrl> key.
2. Click on all the LAD elements you wish to select.
3. Now release the <Ctrl> key.

Selecting all LAD elements in a network

To select all LAD elements in a network, follow these steps:

1. Go to the network whose elements you wish to select.
2. Select the "Select all" command in the "Edit" menu or press <Ctrl A>.

See also

Copying LAD elements (Page 2595)
Cutting LAD elements (Page 2596)
Pasting an LAD element from the clipboard (Page 2596)
Replacing LAD elements (Page 2597)
Inserting additional inputs and outputs in LAD elements (Page 2598)
Removing inputs and outputs (Page 2599)
Enabling and disabling the EN/ENO mechanism (Page 2600)
Deleting LAD elements (Page 2601)

Copying LAD elements

Requirement

An LAD element is available.

Procedure

To copy a LAD element, follow these steps:

1. Right-click the LAD element that you want to copy.
2. Select "Copy" in the shortcut menu.

Result

The LAD element will be copied and saved to the clipboard.

See also

Selecting LAD elements (Page 2594)
Cutting LAD elements (Page 2596)
Pasting an LAD element from the clipboard (Page 2596)
Replacing LAD elements (Page 2597)
Inserting additional inputs and outputs in LAD elements (Page 2598)
Removing inputs and outputs (Page 2599)
Enabling and disabling the EN/ENO mechanism (Page 2600)
Deleting LAD elements (Page 2601)

Cutting LAD elements

Requirement

An LAD element is available.

Cutting

To cut a LAD element, follow these steps:

1. Right-click the LAD element that you want to cut.
2. Select "Cut" in the shortcut menu.

Result

The LAD element will be cut and saved to the clipboard.

See also

Selecting LAD elements (Page 2594)
Copying LAD elements (Page 2595)
Pasting an LAD element from the clipboard (Page 2596)
Replacing LAD elements (Page 2597)
Inserting additional inputs and outputs in LAD elements (Page 2598)
Removing inputs and outputs (Page 2599)
Enabling and disabling the EN/ENO mechanism (Page 2600)
Deleting LAD elements (Page 2601)

Pasting an LAD element from the clipboard

Requirement

An LAD element is available.

Procedure

To paste an LAD element from the clipboard, follow these steps:

1. Copy a LAD element or cut a LAD element.
2. Right-click the point in the network where you want to paste the element.
3. Select "Paste" in the shortcut menu.

See also

Selecting LAD elements (Page 2594)
Copying LAD elements (Page 2595)
Cutting LAD elements (Page 2596)
Replacing LAD elements (Page 2597)
Inserting additional inputs and outputs in LAD elements (Page 2598)
Removing inputs and outputs (Page 2599)
Enabling and disabling the EN/ENO mechanism (Page 2600)
Deleting LAD elements (Page 2601)

Replacing LAD elements

You can easily exchange LAD elements with other LAD elements of the same type. This has the advantage that the parameters are retained and need not be entered again. For example, you can exchange normally open contacts and normally closed contacts or RS FlipFlop and SR FlipFlop.

Requirements

A network with at least one LAD element is present.

Procedure

To replace an LAD element with another LAD element, follow these steps:

1. Select the LAD element that you want to replace.
2. Position the cursor over the triangle in the top right-hand corner of the LAD element.
   A drop-down list is displayed.
3. From the drop-down list, select the LAD element that you want to use to replace the existing LAD element.

See also

Selecting LAD elements (Page 2594)
Copying LAD elements (Page 2595)
Cutting LAD elements (Page 2596)
Pasting an LAD element from the clipboard (Page 2596)
Inserting additional inputs and outputs in LAD elements (Page 2598)
Removing inputs and outputs (Page 2599)
Enabling and disabling the EN/ENO mechanism (Page 2600)
Deleting LAD elements (Page 2601)

Inserting additional inputs and outputs in LAD elements

Introduction

You can expand LAD elements which execute commutative arithmetic instructions by adding additional inputs. Such elements are, for example, the instructions "Add" (ADD) and "Multiply" (MUL). You can expand the MOVE and DEMUX instruction boxes by adding additional outputs.

Requirement

An LAD element is available that permits the insertion of additional inputs and outputs.

Inserting an additional input

To add an additional input to the box of a LAD element, follow these steps:

1. Right-click on an existing input of the LAD element.
2. Select "Insert input" in the shortcut menu.
   An additional input is added to the box of the LAD element.

Or:

1. Click on the yellow star symbol beside the last input in the instruction box.
   An additional input is added to the box of the LAD element.

Inserting an additional output

To add an additional output to the box of a LAD element, follow these steps:

1. Right-click on an existing output of the LAD element.
2. Select "Insert output" from the shortcut menu.
   An additional output is added to the box of the LAD element.

Or:

1. Click on the yellow star symbol beside the last input in the instruction box.
   An additional output is added to the box of the LAD element.

See also

Inserting LAD elements (Page 2579)

Selecting LAD elements (Page 2594)

Copying LAD elements (Page 2595)

Cutting LAD elements (Page 2596)

Pasting an LAD element from the clipboard (Page 2596)

Replacing LAD elements (Page 2597)

Removing inputs and outputs (Page 2599)

Enabling and disabling the EN/ENO mechanism (Page 2600)

Deleting LAD elements (Page 2601)

Removing inputs and outputs

Introduction

Inputs and outputs which you have added to an instruction can be removed.

Requirement

An LAD element is available to which you have added additional inputs and outputs.

Remove input

To remove an input, follow these steps:

1. Select the input that you want to remove.
2. Select the "Delete" command in the shortcut menu.
   The input of the LAD element is removed.

Remove output

To remove an output, follow these steps:

1. Select the output that you want to remove.
2. Select the "Delete" command in the shortcut menu.
   The output of the LAD element will be removed.

See also

Inserting LAD elements (Page 2579)

Selecting LAD elements (Page 2594)

Copying LAD elements (Page 2595)

Cutting LAD elements (Page 2596)

Pasting an LAD element from the clipboard (Page 2596)

Replacing LAD elements (Page 2597)

Inserting additional inputs and outputs in LAD elements (Page 2598)

Enabling and disabling the EN/ENO mechanism (Page 2600)

Deleting LAD elements (Page 2601)

Enabling and disabling the EN/ENO mechanism

In LAD and FBD, certain instructions have an enable output ENO and thus use the EN/ENO mechanism. This allows you to query runtime errors in instructions and react to them. In order to increase the performance of the CPU, the EN/ENO mechanism is disabled in the default setting. This means that you are not initially able to react to runtime errors of the instruction using the ENO value. However, you can enable the EN/ENO mechanism again at any time, if required.

You can enable the EN/ENO mechanism individually for each instruction in order to generate the ENO. If you enable the EN/ENO mechanism for an instruction, other instructions that you subsequently add to your program are also inserted with the EN/ENO mechanism enabled. You can disable the EN/ENO mechanism again at any time if you do not want to use the evaluation of ENO for an instruction. Further instructions that you subsequently add to your program will then be inserted without the EN/ENO mechanism.

See also: Basics of the EN/ENO mechanism (Page 2447)

Activating the EN/ENO mechanism

Proceed as follows to activate the EN/ENO mechanism of an instruction:

1. In your program, right-click the instruction at which you want to activate the EN/ENO mechanism.
2. Select the "Generate ENO" command from the shortcut menu.
   The ENO value is again generated for the instruction. Other instructions are inserted with the enable output.

Deactivating the EN/ENO mechanism

Proceed as follows to deactivate the EN/ENO mechanism of an instruction:

1. In your program, right-click the instruction at which you want to deactivate the EN/ENO mechanism.
2. Select the "Do not generate ENO" command from the shortcut menu.
   The ENO value is no longer generated for the instruction. Other instructions are inserted without enable output.

See also

Selecting LAD elements (Page 2594)

Copying LAD elements (Page 2595)

Cutting LAD elements (Page 2596)

Pasting an LAD element from the clipboard (Page 2596)

Replacing LAD elements (Page 2597)

Inserting additional inputs and outputs in LAD elements (Page 2598)

Removing inputs and outputs (Page 2599)

Deleting LAD elements (Page 2601)

Deleting LAD elements

Requirement

An LAD element is available.

Procedure

To delete a LAD element, follow these steps:

1. Right-click the LAD element that you want to delete.
2. Select the "Delete" command in the shortcut menu.

See also

Inserting LAD elements (Page 2579)
Selecting LAD elements (Page 2594)
Copying LAD elements (Page 2595)
Cutting LAD elements (Page 2596)
Pasting an LAD element from the clipboard (Page 2596)
Replacing LAD elements (Page 2597)
Inserting additional inputs and outputs in LAD elements (Page 2598)
Removing inputs and outputs (Page 2599)
Enabling and disabling the EN/ENO mechanism (Page 2600)

Inserting operands into LAD instructions

Inserting operands

The character strings "<???>", "<??.?>" and "..." are inserted as placeholders for the parameters when an LAD element is inserted. The "<???>" and "<??.?>" strings displayed in red indicate parameters that need to be connected. The "..." string displayed in black indicates parameters that may be connected. "<??.?>" stands for Boolean placeholders.

Note

If you position the cursor over the placeholder, the expected data type will be displayed.

Requirement

An LAD element is available.

Procedure

To connect the parameters of a LAD element, follow these steps:

1. Double-click the placeholder of the parameter.
   An entry field opens, and the placeholder is selected.
2. Enter the appropriate parameter.

   Note

   If you enter the absolute address of a parameter that has already been defined, this absolute address will be changed to the symbolic name of the parameter as soon as the input is confirmed. If you have not yet defined the parameter, a new tag with this absolute address and the default name "Tag_<n>" will be entered in the PLC tag table. When you confirm your input, the absolute address will be replaced with the symbolic name "Tag_<n>".

3. Confirm the parameter with the Enter key.
4. If you have not yet defined the parameter, you can define it directly in the program editor using the shortcut menu.
   See also:
   Declaring PLC tags in the program editor (Page 2482)
   Declaring local tags in the program editor (Page 2552)

Or drag from it the PLC tag table:

1. In the project tree, select the "PLC tags" folder or open the PLC tag table.
2. If you have opened the PLC tag table, drag the symbol from the first column of the desired tag to the appropriate place in your program. If you have not opened the PLC tag table yet, open the detail view now. Drag the desired tag from the detail view to the appropriate place in your program.

Or drag from it the block interface:

1. Open the block interface.
2. Drag the required operand from the block interface to the instruction window.

Result

- If the syntax is error-free, the displayed parameter is black. The editor then jumps to the next placeholder.
- If there is an error in the syntax, the cursor stays in the entry field and a corresponding error message is displayed in the status line. If you press the Enter key again, the entry field is closed and the faulty entry is displayed in red italics.

See also

Using and addressing operands (Page 2197)

Using autocompletion (Page 2540)

Wiring hidden parameters

Introduction

Depending on the CPU used, you can use complex instructions in your program that are dispatched with the TIA portal. These instructions can contain parameters that are declared as hidden.

If an instruction contains hidden parameters, the instruction box has a small arrow on the lower edge. You can recognize hidden parameters by their white font.

You can show and wire the hidden parameters at any time.

Showing or hiding hidden parameters

To show or hide hidden parameters, follow these steps:

1. Click on the down arrow at the bottom edge of the instruction box to show hidden parameters.
2. Click on the up arrow at the bottom edge of the instruction box to hide hidden parameters.

Wiring hidden parameters

To wire parameters, follow these steps:

1. Wire the hidden parameters like normally visible parameters.
   The hidden parameter is transformed into a visible parameter.

See also

Using libraries (Page 415)

Displaying or hiding variable information

Introduction

You can display the following information about the tags to be used in the Program editor:

- Name of the tag
- Address of the tag
- Simple or hierarchical comments for tag documentation

The information is taken from the block interface for local tags and DB tags and from the PLC tag table for tags that are valid CPU-wide.

You can display the tag information either for all the blocks or for individually opened blocks. If you display the tag information for all the blocks, the tag information for all the blocks currently opened and opened in future is shown.

You can hide the tag information at any time again. If you have hidden the tag information for all blocks, you can display it again for individual blocks that are open.

When you select the display of tag information with hierarchical comments, the comments of the higher structure levels are also displayed for structured tags. The comments are shown in brackets after the tag comment; the comments of the individual levels are separated by a dot. If there is no comment for a tag on a structure level, it is not displayed. This is indicated by two successive dots.

Displaying or hiding tag information for all the blocks

Proceed as follows to display or hide the tag information for all the blocks:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. If you want to display the tag information, select the "Show" option or the "Tag information with hierarchical comments" option in the drop-down list, depending on whether you want to display simple or hierarchical comments.
4. If you want to hide the tag information, select the option "Collapse" in the "Tag information" drop-down list.
   The tag information is displayed or hidden for all the blocks. When you open additional blocks, the tag information is displayed or hidden depending on the selected setting.

Displaying or hiding tag information for an opened block

Proceed as follows to display or hide the tag information for an opened block:

1. If you want to display the tag information, select the "Show tag information" option or the "Tag information with hierarchical comments" option in the "Shows the tag information" drop-down list, depending on whether you want to display simple or hierarchical comments.
2. If you want to hide the tag information, select the option "Hide tag information" in the "Shows the tag information" drop-down list.
   The tag information is displayed or hidden.

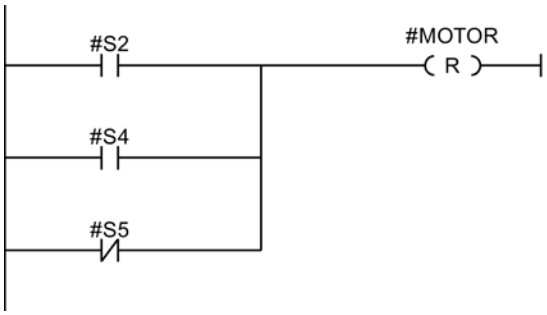
Branches in LAD

Basic information on branches in LAD

Definition

You use branches to program parallel circuits with the Ladder Logic (LAD) programming language. Branches are inserted in the main rung. You can insert several contacts into the branch and thus achieve a parallel circuit of series connections. This allows you to program complex ladder logic.

The figure below shows an example of the use of branches:



MOTOR carries signal 1, if one of the following conditions is fulfilled:

- Signal 1 is pending on S2 or S4
- Signal 0 is pending on S5.

See also

Rules for branches in LAD (Page 2605)

Inserting branches into the LAD network (Page 2606)

Closing branches in the LAD network (Page 2606)

Deleting branches in LAD networks (Page 2607)

Rules for branches in LAD

Rules

The following rules apply to simultaneous branches:

- A simultaneous branch can only be inserted if the main branch already contains an LAD element.
- Simultaneous branches are opened downwards or are connected directly to the power rail. They are terminated upwards.
- Simultaneous branches are opened after the selected LAD element.
- Simultaneous branches are terminated after the selected LAD element.
- To delete a simultaneous branch, you must delete all LAD elements of this branch. When the last LAD element is removed from the branch, the rest of the branch is also removed.

See also

Basic information on branches in LAD (Page 2604)

Inserting branches into the LAD network (Page 2606)

Deleting branches in LAD networks (Page 2607)

Closing branches in the LAD network (Page 2606)

Inserting branches into the LAD network

You can create several branches in a network.

Requirement

- A network is available.
- The network contains elements.

Procedure

To insert a new branch in a network, follow these steps:

1. Open the "Instructions" task card.
2. Navigate to "General > Open branches" in the "Simple instructions" palette.
3. Use drag-and-drop to move the element to the desired place in the network.
   If you want to connect the new branch directly to the power rail, drag the element to the power rail.

See also

Basic information on branches in LAD (Page 2604)

Rules for branches in LAD (Page 2605)

Deleting branches in LAD networks (Page 2607)

Closing branches in the LAD network

Branches must be closed again at suitable places. If necessary, branches will be arranged so that they do not cross each other.

Requirement

A branch is available.

Procedure

To close an open branch, follow these steps:

1. Select the open branch.
2. Press and hold down the left mouse button.
   A dashed line will appear as soon as the cursor is moved.
3. Drag the dashed line to a suitable place on the network. Permissible connections are indicated by green lines.
4. Release the left mouse button.

See also

Basic information on branches in LAD (Page 2604)
Rules for branches in LAD (Page 2605)

Deleting branches in LAD networks

Requirement

A branch is available.

Procedure

To delete a branch, follow these steps:

1. Select the connection line that links the branch to the main branch.
2. Select the "Delete" command in the shortcut menu.

See also

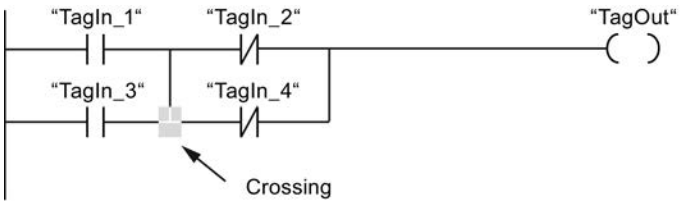
Basic information on branches in LAD (Page 2604)
Rules for branches in LAD (Page 2605)
Inserting branches into the LAD network (Page 2606)

Crossings in LAD

Basic information on crossings in LAD

Definition

A crossing is a place in a LAD network where one branch is closed and at the same time another branch is opened.



"TagOut" receives signal 1, if the following two conditions are met:

- "TagIn_1" or "TagIn_3" has signal 1
- "TagIn_2" or "TagIn_4" has signal 0

Inserting crossings

You can insert crossings in a LAD network by creating connections between the main branch and an additional branch or between different branches.

Requirements

A branch is available.

Procedure

To insert a new crossing in an LAD network, follow these steps:

1. Open the "Instructions" task card.
2. Navigate to "General > Open branches" in the "Simple instructions" palette.
3. Drag the element behind the existing branch.
4. Insert any element into the open branch.
5. Click the arrow of the open branch after the inserted element.
6. Hold down the left mouse button and drag the dashed connecting line to the main branch.
7. Release the left mouse button.

See also

Rearranging crossings (Page 2608)

Deleting crossings (Page 2609)

Inserting branches into the LAD network (Page 2606)

Rearranging crossings

Requirement

A crossing is available.

Procedure

To rearrange a connection, follow these steps:

1. Select the connection line that defines the crossings in the respective branches.
2. Select the "Delete" command in the shortcut menu.
3. Open the "Instructions" task card.
4. Navigate to "General > Open branches" in the "Simple instructions" palette.
5. Use a drag-and-drop operation to move the element to the place in the network where you want to insert the new crossing.
6. Click on the arrow for the open branch.

7. Hold down the left mouse button and drag the dashed connecting line to the subsidiary branch in which you wish to insert the new crossing.
8. Release the left mouse button.

See also

Inserting crossings (Page 2608)
Deleting crossings (Page 2609)

Deleting crossings

Requirement

A crossing is available.

Procedure

To delete a crossing, follow these steps:

1. Select the connection line that defines the crossings in the respective branches.
2. Select the "Delete" command in the shortcut menu.

See also

Inserting crossings (Page 2608)
Rearranging crossings (Page 2608)

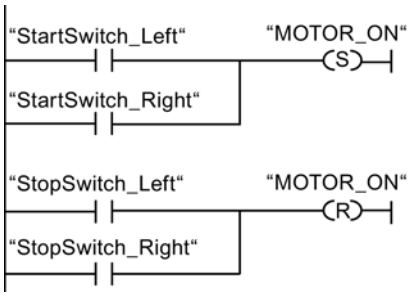
Rungs in LAD

Basic information on rungs in LAD

Using rungs

The program is mapped in one or more networks. A network contains a power rail on the left where one or more rungs originate. The binary signal scans are arranged in the form of contacts on the rungs. The serial arrangement of the elements on a rung creates a series connection; arrangement on simultaneous branches creates a parallel connection. A rung is closed by a coil or a box in which the result of logic operation will be written.

The figure below shows an example of the use of several rungs within a network:



Rules

Remember the following rules when using several rungs:

- Connections are not permitted between rungs.
- Only one jump instruction is permissible per network. The positioning rules for jump instructions remain valid.

Running rungs

Rungs and networks are executed from top to bottom and from left to right. This means that the first instruction in the first rung of the first network is processed first. All instructions of this rung are then processed. After this come all other rungs of the first network. The next network is processed only after all rungs have first been run.

Differences between branches and rungs

The difference between branches and rungs is that the rungs are independent branches that can also stand in a different network. Branches, on the other hand, permit the programming of a parallel connection.

See also

Insert rung (Page 2610)

Deleting a rung (Page 2611)

Insert rung

Requirement

- A block is open.
- A network is available.

Procedure

To insert a new rung in a network, proceed as follows:

1. Insert any coil on the power rail.
   A new rung will be inserted and the coil positioned at the end of the rung.
2. Insert additional instructions in the new rung.

See also

Basic information on rungs in LAD (Page 2609)

Deleting a rung (Page 2611)

Deleting a rung

Requirement

A rung is available.

Procedure

To delete a rung, proceed as follows:

1. Hold down the left mouse button and draw a frame around the rung. At the same time, make sure that you select all instructions. Alternatively, you can hold down the <Shift> key and select the first the last instruction of the rung.
2. Right-click on one of the instructions in the rung.
3. Select the "Delete" command in the shortcut menu.

See also

Basic information on rungs in LAD (Page 2609)

Insert rung (Page 2610)

Creating FBD programs

Basic information on FBD

FBD programming language

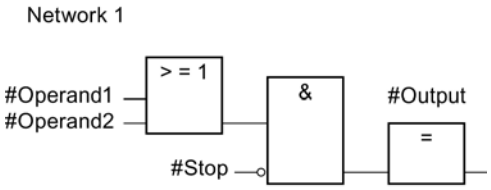
Overview of the Function Block Diagram (FBD) programming language

FBD is a graphical programming language. The representation is based on electronic circuit systems.

The program is mapped in one or more networks. A network contains one or more logic operation paths. The binary signal scans are linked by boxes. The representation of the logic is based on the graphical logic symbols used in Boolean algebra.

Example of networks in FBD

The following figure shows an FBD network with AND and OR boxes and an assignment:



See also

Working with networks (Page 2615)

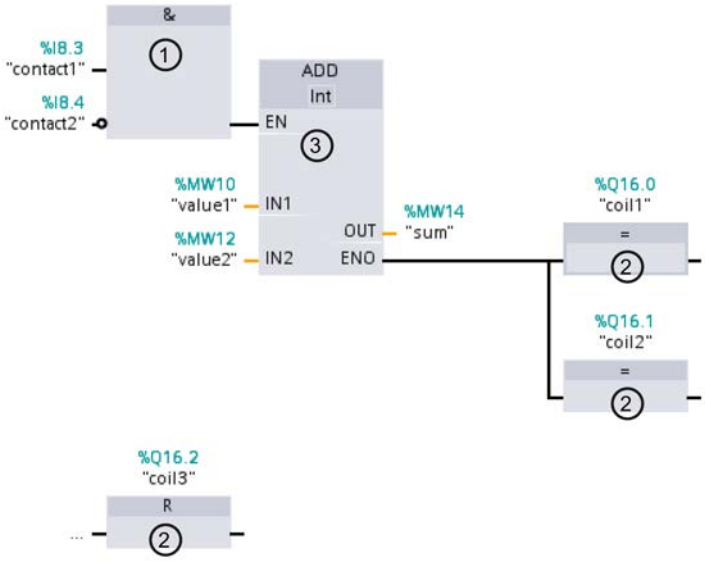
Overview of the FBD elements

FBD elements

An FBD program consists of separate elements that are linked by means of a binary signal flow. Most program elements must be supplied with tags.

A FBD network is programmed from left to right.

For example, the following figure shows elements of an FBD network:



1) Binary function

2) Standard box

3) Complex box

Binary functions

You can use binary functions to query binary operands and to combine their signal states. The following operations are examples of binary functions: "AND operation", "OR operation" and "EXCLUSIVE OR operation".

Standard boxes:

You can use standard boxes to control binary operands, perform RLO edge detection or execute jump functions in the program. Standard boxes generally have only one single input.

Complex boxes

Complex boxes represent program elements with complex functions. The empty box is an exception. You can use the empty box as a placeholder in which you can select the required instruction.

The following types of boxes are available to you in an FBD program:

- Complex boxes without EN/ENO mechanism:
  A box is executed independently of the signal state at the box inputs. The error status of the processing cannot be queried.
- Complex boxes with EN/ENO mechanism:
  A box is only executed if the enable input "EN" has the signal state "1". If the box is processed correctly, the "ENO" enable output has signal state "1". If an error occurs during processing, the "ENO" output is reset.
  If the EN enable input is not interconnected, the box is always executed.

Calls of code block are also shown in the network as complex boxes with EN/ENO mechanism.

See also

EN/ENO mechanism (Page 2447)

Settings for FBD

Overview of the settings for FBD

Overview

The following table shows the settings that you can make:

Group	Setting	Description
Font	Font size	Font size in program editor
View	Layout	Compact or wide Changes the vertical spacing between operands and other objects (such as operand and contact). The change becomes visible once the block is reopened.
	With absolute information	Additional display of the absolute addresses
Operand field	Maximum width	Maximum number of characters that can be entered horizontally in the operand field. This setting recalculates the layout of the networks.
	Maximum height	Maximum number of characters that can be entered vertically in the operand field. This setting recalculates the layout of the networks.

See also

Changing the settings (Page 2614)

Changing the settings

Procedure

To change the settings, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. Change the settings.

Result

The change will be loaded directly, there is no need to save it explicitly.

See also

Overview of the settings for FBD (Page 2614)

Working with networks

Using networks

Function

The user program is created in the block within networks. For a code block to be programmed, it must contain at least one network. To achieve a better overview of the user program, you can also subdivide your program into several networks.

See also

Inserting network title (Page 2618)

Entering a network comment (Page 2620)

Navigating networks (Page 2620)

Inserting networks

Requirement

A block is open.

Procedure

To insert a new network, follow these steps:

1. Select the network after which you want to insert a new network.
2. Select the "Insert network" command in the shortcut menu.

Result

A new empty network is inserted into the block.

See also

Inserting network title (Page 2618)

Entering a network comment (Page 2620)

Navigating networks (Page 2620)

Selecting networks

Requirements

A network is available.

Selecting a network

To select a network, follow these steps:

1. Click the title bar of the network that you want to select.

Selecting several networks

Proceed as follows to select several individual networks:

1. Press and hold down the <Ctrl> key.
2. Click all the networks that you want to select.

To select several successive networks, follow these steps:

1. Press and hold down the <Shift> key.
2. Click the first network that you want to select.
3. Click the last network that you want to select.
   The first and last networks and all those in between are selected.

See also

Inserting networks (Page 2615)

Inserting network title (Page 2618)

Entering a network comment (Page 2620)

Navigating networks (Page 2620)

Copying and pasting networks

Copied networks can be pasted within the block or in another block. Networks that were created in LAD or FBD can also be inserted in blocks of the respective other programming language.

Requirement

A network is available.

Procedure

To copy and paste a network, follow these steps:

1. Select the network or networks to be copied.
2. Select "Copy" in the shortcut menu.

3. Select the network after which you want to paste in the copied network.
4. Select "Paste" in the shortcut menu.

See also

Inserting networks (Page 2615)
Selecting networks (Page 2616)
Inserting network title (Page 2618)
Entering a network comment (Page 2620)
Navigating networks (Page 2620)

Deleting networks

Requirement

A network is available.

Procedure

To delete a network, follow these steps:

1. Select the network that you want to delete.
2. Select the "Delete" command in the shortcut menu.

See also

Inserting networks (Page 2615)
Selecting networks (Page 2616)
Copying and pasting networks (Page 2616)
Inserting network title (Page 2618)
Entering a network comment (Page 2620)
Navigating networks (Page 2620)

Expanding and collapsing networks

Requirements

A network is available.

Opening and closing a network

To open a network, follow these steps:

1. Click on the right arrow in the network title bar.

To close a network, follow these steps:

1. Click on the down arrow in the network title bar.

Opening and closing all networks

To open and close all networks, follow these steps:

1. In the toolbar, click "Open all networks" or "Close all networks".

See also

Inserting networks (Page 2615)
Selecting networks (Page 2616)
Copying and pasting networks (Page 2616)
Deleting networks (Page 2617)
Inserting network title (Page 2618)
Entering a network comment (Page 2620)
Navigating networks (Page 2620)

Inserting network title

The network title is the header of a network. The length of the network title is limited to one line. You can enter the title manually or set it automatically. When you set it automatically, you can do this for individual networks or use the settings to specify that the network title is always set automatically.

For automatic insertion of the network title, the comment of the operand in one of the following instructions in the network is evaluated:

- Assignment
- Set output
- Reset output

The instruction that is listed first in the network is used.

The network title is only inserted automatically when the following conditions are fulfilled:

- The network does not have a title yet.
- The operand of the instruction used for the comment has a comment.

Note

Note the following restrictions for automatic insertion of the network title:

- The network title is not adapted if you change the comment of the operand at a later time.
- The network title is not adapted if you change the operand of the instruction.
- The network title is only set by the writing instructions listed above.
- If the operand is of the data type array, the comment of the array is used and not the comments of the array elements.
- Comments of invalid operands are not taken into consideration.

Entering the network title manually

To enter a network title, follow these steps:

1. Click on the title bar of the network.
2. Enter the network title.

Setting the network title automatically

To specify that the network titles are always set automatically, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. Select the "Set network title automatically" check box in the "Additional settings" group.
   The network titles are set automatically as of this time if the conditions listed above are fulfilled.

To set an individual network title automatically, follow these steps:

1. Right-click "Network <Number of the network>" in the title bar of a network.
2. Select the "Set network title automatically" command in the shortcut menu.
   The title of the selected network is set based on the comment of the operand if the conditions listed above are fulfilled.

See also

Using networks (Page 2615)

Inserting networks (Page 2615)

Selecting networks (Page 2616)

Copying and pasting networks (Page 2616)

Deleting networks (Page 2617)

Expanding and collapsing networks (Page 2617)

Entering a network comment (Page 2620)

Navigating networks (Page 2620)

Entering a network comment

You can use network comments to provide comments on the program contents of individual networks. For example, you can indicate the function of the network or draw attention to special characteristics.

Requirement

A network is available.

Procedure

To enter a network comment, follow these steps:

1. Click on the right arrow before the network title.
2. If the comment area is not visible, click "Network comments on/off" in the toolbar.
   The comment area is displayed.
3. Click "Comment" in the comment area.
   The "Comment" text passage is selected.
4. Enter the network comment.

See also

Using networks (Page 2615)

Inserting networks (Page 2615)

Selecting networks (Page 2616)

Copying and pasting networks (Page 2616)

Deleting networks (Page 2617)

Expanding and collapsing networks (Page 2617)

Inserting network title (Page 2618)

Navigating networks (Page 2620)

Navigating networks

You can navigate straight to a specific position within a block.

Procedure

To navigate to a specific position within a block, follow these steps:

1. Right-click in the code area of the programming window.
2. Select the "Go to > Network/line" command in the shortcut menu.
   The "Go to" dialog will open.
3. Enter the network to which you want to navigate.
4. Enter the line number of the network to which you want to navigate.
5. Confirm your entry with "OK".

Result

The relevant line will be displayed if this is possible. If the network or line requested does not exist, the last existing network or the last existing line in the network requested will be displayed.

See also

Using networks (Page 2615)

Inserting networks (Page 2615)

Selecting networks (Page 2616)

Copying and pasting networks (Page 2616)

Deleting networks (Page 2617)

Expanding and collapsing networks (Page 2617)

Inserting network title (Page 2618)

Entering a network comment (Page 2620)

Inserting FBD elements

Rules for the use of FBD elements

Rules

Note the following rules when inserting FBD elements:

- An FBD network can consist of several elements. All elements of a logic path must be linked to each other according to IEC 61131-3.
- Standard boxes (flip flops, counters, timers, math operations, etc.) can be added as output to boxes with binary logic operations (for example, AND, OR). Comparison boxes are excluded from this rule.
- Only Boolean inputs in an instruction can be combined with preceding logic operations.
- Only the bottom Boolean output in an instruction can be combined with an additional logic operation.

- Enable input EN or enable output ENO can be connected to boxes, but this is not mandatory.
- Constants (for example, TRUE or FALSE) cannot be assigned to binary logic operations. Instead, use tags of the BOOL data type.
- Only one jump instruction can be inserted in each network.
- Only one jump label can be inserted in each network.
- Instructions for positive or negative RLO edge detection may not be arranged right at the left of the network as this requires a prior logic operation.

Placement rules for S7-1200/1500 CPUs

The following table sets out the instructions that can only be positioned at the end of the network:

Instruction		Preceding logic operation required
Mnemonics	Name	
SET_BF	Set bit field	No
RESET_BF	Reset bit field	No
JMP	Jump if RLO = 1	No
JMPN	Jump if RLO = 0	Yes
JMP_LIST	Define jump list	No
SWITCH	Jump distributor	No
RET	Return	No

Placement rules for S7-300/400 CPUs

The following table sets out the instructions that can only be positioned at the end of the network:

Instruction		Preceding logic operation required
Mnemonics	Name	
S	Set output	Yes
R	Reset output	Yes
SP	Start pulse timer	Yes
SE	Start extended pulse timer	Yes
SD	Start on-delay timer	Yes
SS	Start retentive on-delay timer	Yes
SF	Start off-delay timer	Yes
SC	Set counter value	Yes
CU	Count up	Yes
CD	Count down	Yes
JMP	Jump if RLO = 1	No
JMPN	Jump if RLO = 0	Yes

Instruction		Preceding logic operation required
Mnemonics	Name	
RET	Return	No
OPN	Open global data block	No
OPNI	Open instance data block	No
CALL	Call block	No
SAVE	Save RLO in BR bit	No
MCRA	Enable MCR range	No
MCRD	Disable MCR range	No
MCR<	Open MCR ranges	No
MCR>	Close MCR ranges	No

Inserting FBD elements using the "Instructions" task card

Requirement

A network is available.

Procedure

To insert FBD elements into a network using the "Instructions" task card, follow these steps:

1. Open the "Instructions" task card.
2. Navigate to the FBD element that you want to insert.
3. Use drag-and-drop to move the element to the desired place in the network.
   If the element is an internal system function block (FB), the "Call options" dialog opens. In this dialog you can create an instance data block of the single-instance or multiple-instance type for the function block in which the data of the inserted element will be saved. You will find the new instance data block created in the project tree in the "Program resources" folder under "Program blocks > System blocks". If you have selected "multiple instance", these are located in the block interface in the "Static" section.

Or:

1. Select the point in the network at which you want to insert the element.
2. Open the "Instructions" task card.
3. Double-click on the element you want to insert.
   If the element is an internal system function block (FB), the "Call options" dialog opens. In this dialog you can create an instance data block of the single-instance or multiple-instance type for the function block in which the data of the inserted element will be saved. You will find the new instance data block created in the project tree in the "Program resources" folder under "Program blocks > System blocks". If you have selected "multiple instance", these are located in the block interface in the "Static" section.

Result

The selected FBD element is inserted with dummy entries for the parameters.

See also

Rules for the use of FBD elements (Page 2621)

Inserting FBD elements using an empty box

Requirement

A network is available.

Procedure

To insert FBD elements into a network using an empty box, follow these steps:

1. Open the "Instructions" task card.
2. Navigate to "General > Empty box" in the "Basic instructions" palette.
3. Use a drag-and-drop operation to move the "Empty box" element to the desired place in the network.
4. Position the cursor over the triangle in the top right-hand corner of the empty box.
   A drop-down list is displayed.
5. Select the desired FBD element from the drop-down list.
   If the element is an internal system function block (FB), the "Call options" dialog opens. In this dialog you can create an instance data block of the single-instance or multiple-instance type for the function block in which the data of the inserted element will be saved. You will find the new instance data block created in the project tree in the "Program resources" folder under "Program blocks > System blocks". If you have selected "multi-instance", these are located in the block interface in the "Static" section.

Result

The empty box is changed to the respective FBD element. Placeholders are inserted for the parameters.

Selecting the data type of an FBD element

Selecting a data type

Introduction

Some instructions can be executed with several different data types. If you use one of these instructions in the program, you have to specify a valid data type for the instruction at the

specific point in the program. For some instructions, you have to select the data types for the inputs and outputs separately.

Note

The valid data type (BOOL) for the tags on the enable input EN and the enable output ENO is predefined by the system and cannot be changed.

The valid data types for an instruction are listed in the instruction drop-down list. You specify the data type of the instruction by selecting an entry from the drop-down list. If the data type of an operand differs from the data type of the instruction and cannot be converted implicitly, the operand is displayed in red and a rollout with the corresponding error message appears.

Data type selection of mathematical instructions

Some mathematical instructions provide you with the option of having the data type automatically set corresponding to the data types of the operand. In the drop-down list for data type selection, these instructions have the entry "Auto" in addition to the actual data types. If you select this entry and then allocate the first operand, the data type of the operand is selected as data type for the instruction. The entry in the drop-down list changes to "Auto (<Data type>)", e.g. "Auto (Real)". If you allocate additional operands, the automatically set data type of the instruction is adjusted according to the following criteria:

- You supply all other operands with tags of the same data type:
  The data type of the instruction is not changed.
- You supply all other operands with tags whose data type is smaller than the data type of the instruction:
  The data type of the instruction is not changed. For the operand with the smaller data type, an implicit conversion is conducted if necessary.
- You supply an additional operand with a tag whose data type is greater than the data type of the instruction:
  The data type of the instruction is changed to the larger data type. An implicit conversion is performed, if necessary, for operands that deviate from the newly set data type of the instruction.

Each change in the data type of an operand can result in a change of the data type of the instruction. Other operands may possibly be implicitly converted as a result. Operands for which an implicit conversion is performed are marked with a gray square.

Note

Please also observe the information on data type conversion for your device and, in particular, the notes on the IEC check.

See also: Data type conversion (Page 2286)

See also

Defining the data type of an instruction (Page 2626)

Defining the data type of an instruction

Introduction

Some instructions can be executed with several different data types. When you insert such instructions into your program, you must specify the data type for these instructions at the actual point in the program.

Specifying the data type by means of the drop-down list

To define the data type of an instruction using the drop-down list, follow these steps:

1. Insert the instruction at the required point in the program using drag-and-drop.
   The entry "???" (undefined) is displayed in the drop-down list of the inserted instruction.
2. Click the triangle in the upper corner of the drop-down list.
   The drop-down list will open to display the data types valid for the instruction.
3. Select a data type from the drop-down list.
   The selected data type is displayed.
4. If the instruction has two drop-down lists, select the data type for the instruction inputs in the left-hand drop-down list and the data type for the instruction outputs in the right-hand drop-down list.

Specifying data type by assigning tags

To define the data type of an instruction by assigning tags, follow these steps:

1. Insert the instruction at the required point in the program using drag-and-drop.
   The entry "???" (undefined) is displayed in the drop-down list of the inserted instruction.
2. At an input or output, specify a valid tag, the data type of which is to be applied as the instruction data type.
   The data type of the tag is displayed in the drop-down list.
3. Enter a valid tag at an input and a valid tag at an output if data types need to be defined for both the inputs and outputs of the instruction. The tag specified at the input determines the data type of the inputs; the tag specified at the output determines the data type of the outputs of the instruction.

Automatically specifying the data type of mathematical instructions

To automatically specify the data type for mathematical instructions, follow these steps:

1. Insert the mathematical instruction at the required point in the program using drag-and-drop.
   The entry "???" (undefined) is displayed in the drop-down list of the inserted instruction.
2. Select the "Auto" entry from the drop-down list.
3. Enter a valid tag at an input or output.
   The data type of the tag is applied as data type of the instruction. The entry in the drop-down list changes to "Auto (<Data type>)".

See also: Selecting a data type (Page 2624)

See also

Selecting a data type (Page 2624)

Using favorites in FBD

Adding FBD elements to Favorites

Requirement

- A block is open.
- The multipane mode is set for the "Instructions" task card or the Favorites are also displayed in the editor.

Procedure

To add SCL instructions to the Favorites, follow these steps:

1. Open the "Instructions" task card.
2. Maximize the "Basic instructions" pane.
3. Navigate in the "Basic instructions" pane to the instruction that you want to add to the Favorites.
4. Drag-and-drop the instruction into the "Favorites" pane or into the Favorites area in the program editor.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Removing FBD elements from Favorites (Page 2628)

Overview of the program editor (Page 2523)

Inserting FBD elements using favorites

Requirement

- A block is open.
- Favorites are available.

Procedure

To insert an instruction into a program using Favorites, follow these steps:

1. Drag-and-drop the desired instruction from Favorites to the desired position.

Or:

1. Select the position in the program where you want to insert the instruction.
2. In the Favorites, click on the instruction you want to insert.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Overview of the program editor (Page 2523)

Removing FBD elements from Favorites (Page 2628)

Removing FBD elements from Favorites

Requirement

A code block is open.

Procedure

To remove instructions from Favorites, follow these steps:

1. Right-click on the instruction you want to remove.
2. Select the "Remove instruction" command in the shortcut menu.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Adding FBD elements to Favorites (Page 2627)

Inserting FBD elements using favorites (Page 2627)

Overview of the program editor (Page 2523)

Inserting block calls in FBD

Inserting block calls using a drag-and-drop operation

You can insert calls for existing functions (FC) and function blocks (FB) using a drag-and-drop operation from the project tree. If you call function blocks from other function blocks, you can either call them as single-instance or multi-instance blocks. If a function block is called as single instance, it will store its data in a data block of its own. If a function block is called as multi-instance, it will store its data in the instance data block of the calling function block.

Requirement

- A network is available.
- The block that is to be called is available.

Inserting a call of a function (FC)

To insert a call of a function (FC) into a network using a drag-and-drop operation, follow these steps:

1. Drag the function from the project tree to the required network.

Inserting a call for a function block (FB)

To insert a call for a function block (FB), follow these steps:

1. Drag the function block from the project tree to the required network.
   The "Call options" dialog opens.
2. Enter in the dialog whether you wish to call the block as single or multi-instance.
   - If you click on the "Single instance" button, you will have to enter a name in the "Name" text box for the data block that you want to assign to the function block.
   - If you click on the "Multi-instance" button, you will have to enter the name of the tag in the "Name in the interface" text box; this is the name that you use to enter the called function block as a static tag in the interface of the calling block.
3. Confirm your entries with "OK".

Result

The function or the function block is inserted with its parameters. You can then assign the parameters.

See also: Parameter transfer at block call (Page 2185)

Note

If when calling a function block you specify an instance data block that does not exist, it will be created. If you have called a function block as a multi-instance, this will be entered as a static tag in the interface.

See also

Updating block calls in FBD (Page 2630)

Changing the instance type (Page 2631)

Single instances (Page 2183)

Multi-instances (Page 2183)

Updating block calls in FBD

If interface parameters of a called block are changed, the block call can no longer be executed correctly. You can avoid such inconsistent block calls by updating the block calls.

You have two options to update the block calls:

- Explicit updating in the program editor.
  The block calls in the open block will be updated.
- Implicit updating during compilation.
  All block calls in the program as well as the used PLC data types will be updated.

Update blocks in the program editor

To update a block call within a block, follow these steps:

1. Open the block in the program editor.
2. Click "Update inconsistent block calls" in the toolbar.

Or:

1. Open the block in the program editor.
2. Right-click on the instruction with the block call.
3. Select the "Update" command in the shortcut menu.
   The "Interface update" dialog opens. This dialog shows the differences between the block interface in use and the changed interface of the called block.
4. If you want to update the block call, click "OK". To cancel the update, click "Cancel".

Update block calls during compilation

Follow these steps to update all block calls and uses of PLC data types during compilation implicitly:

1. Open the project tree.
2. Select the "Program blocks" folder.
3. Select the command "Compile > Software (rebuild all blocks)" in the shortcut menu.

See also

Block calls (Page 2180)
Inserting block calls using a drag-and-drop operation (Page 2629)
Changing the instance type (Page 2631)

Changing the instance type

Instance type

There are two ways of calling function blocks:

- As a single instance
- As a multiple instance

See also: Call function blocks as single or multi-instances (Page 2182)

You can modify a defined instance type at any time.

Requirement

The user program contains a block call.

Procedure

To change the instance type of a function block, follow these steps:

1. Open the code block and select the block call.
2. Select the "Change instance" command in the shortcut menu.
   The "Call options" dialog opens.
3. Click the "Single instance" or "Multi instance" button.
   - If you select the "Single instance" instance type, enter a name for the data block that is to be assigned to the function block.
   - If you select "Multiple instance" as the instance type, enter in the "Name in the interface" text field the name of the tag with which the called function block is to be entered as a static tag in the interface of the calling block.
4. Confirm your entries with "OK".

Note

The previous single and multiple instances will not be deleted automatically.

See also

Inserting block calls using a drag-and-drop operation (Page 2629)
Updating block calls in FBD (Page 2630)

Inserting complex FBD instructions

Using the "Calculate" instruction

Requirement

A network is available.

Procedure

Proceed as follows to use the "Calculate" instruction:

1. Open the "Instructions" task card.
2. Navigate to "Math functions > CALCULATE" in the "Basic instructions" pane.
3. Use drag-and-drop to move the element to the desired place in the network.
   The instruction "Calculate" will be inserted for the data type with a placeholder expression and question mark.
4. Enter the data type for the calculation.
5. Enter the operands for the calculation.

   Note

   The calculation is run with the inputs of the "Calculate" instruction. If you want to use constants you must also insert appropriate inputs for them.

6. Click on the "Edit 'Calculate' instruction" button to replace the placeholder expression with the correct expression.
   The "Edit 'Calculate' instruction" dialog will open.
7. Enter the required expression in the "OUT:= " text box.

   Note

   In the "Example" area you can find an example of a valid expression and possible instructions that you can use.

   To determine a value with the help of Pythagoras' theorem, for example, enter "OUT := SQRT (SQR (IN1) + SQR (IN2))".

8. Confirm your entry with "OK".

See also

CALCULATE: Calculate (Page 3560)

Using free-form comments

Basic information on using free comments in FBD

Introduction

Free-form comments allow you to add comments to the source code for graphic programming languages similar to line comments for textual languages.

Free-form comments can be used for all non-binary boxes.

See also

Inserting free-form comments (Page 2633)

Editing free-form comments (Page 2634)

Deleting free-form comments (Page 2635)

Inserting free-form comments

Requirement

A network with instructions is available.

Procedure

To insert a free comment on an instruction, proceed as follows:

1. If necessary, activate the "Free-form comments on/off" button in the toolbar.
2. Right-click on the instruction for which you want to insert a free-form comment.
3. Select the "Insert comment" command in the shortcut menu.
   A comment box with a standard comment opens. The comment box is connected by an arrow to the corresponding instruction.
4. Enter the required comment in the comment box.

See also

Basic information on using free comments in FBD (Page 2633)

Editing free-form comments (Page 2634)

Deleting free-form comments (Page 2635)

Editing free-form comments

Introduction

Free-form comments can be edited as follows:

- Changing the comment text
- Changing the position and size of the comment box
- Attaching a comment to another element
- Showing and hiding free comments

Changing the comment text

To change the text of free-form comments, follow these steps:

1. Click on the comment box.
2. Enter the desired text.

Changing the position of the comment box

To change the positioning of the comment box, follow the steps below:

1. Left-click the comment box and keep the mouse button pressed.
2. Drag the comment box to the desired location.

Changing the size of the comment box

To change the size of the comment box, follow the steps below:

1. Click on the comment box.
2. Drag the comment box on the move handle in the lower right corner to the desired size.

Attaching a comment to another element

To attach a free-form comment to another element, follow these steps:

1. Left-click the point of the arrow that links the comment box with the instruction and keep the mouse button pressed.
2. Drag the arrow to the element to which you want to attach the comment. Possible insertion points are marked with a green square.
3. Release the mouse button.

Showing and hiding free comments

To show or hide a free-form comments, follow these steps:

1. Click the "Free-form comment on/off" button in the toolbar.

See also

Basic information on using free comments in FBD (Page 2633)
Inserting free-form comments (Page 2633)
Deleting free-form comments (Page 2635)

Deleting free-form comments

Procedure

To delete a free-form comment, proceed as follows:

1. Right-click on the free-form comment that you want to delete.
2. Select the "Delete" command in the shortcut menu.

See also

Basic information on using free comments in FBD (Page 2633)
Inserting free-form comments (Page 2633)
Editing free-form comments (Page 2634)

Editing FBD elements

Selecting FBD elements

You can select several individual elements or all elements in a network.

Requirement

FBD elements are available

Selecting several individual FBD elements

To select several individual FBD elements, follow these steps:

1. Press and hold down the <Ctrl> key.
2. Click on all the FBD elements you wish to select.
3. Now release the <Ctrl> key.

Selecting all FBD elements in a network

To select all FBD elements in a network, follow these steps:

1. Go to the network whose elements you wish to select.
2. Select the "Select all" command in the "Edit" menu or press <Ctrl+A>.

See also

Copying FBD elements (Page 2636)
Cutting FBD elements (Page 2637)
Pasting an FBD element from the clipboard (Page 2637)
Replacing FBD elements (Page 2638)
Adding additional inputs and outputs to FBD elements (Page 2639)
Removing instruction inputs and outputs (Page 2640)
Enabling and disabling the EN/ENO mechanism (Page 2641)
Deleting FBD elements (Page 2642)

Copying FBD elements

Requirement

An FBD element is available.

Procedure

To copy an FBD element, follow these steps:

1. Right-click the FBD element that you want to copy.
2. Select "Copy" in the shortcut menu.

Result

The FBD element will be copied and saved to the clipboard.

See also

Selecting FBD elements (Page 2635)
Cutting FBD elements (Page 2637)
Pasting an FBD element from the clipboard (Page 2637)
Replacing FBD elements (Page 2638)
Adding additional inputs and outputs to FBD elements (Page 2639)
Removing instruction inputs and outputs (Page 2640)

Enabling and disabling the EN/ENO mechanism (Page 2641)
Deleting FBD elements (Page 2642)

Cutting FBD elements

Requirement

An FBD element is available.

Cutting

To cut an FBD element, follow these steps:

1. Right-click the FBD element that you want to cut.
2. Select "Cut" in the shortcut menu.

Result

The FBD element will be cut and saved to the clipboard.

See also

Selecting FBD elements (Page 2635)
Copying FBD elements (Page 2636)
Pasting an FBD element from the clipboard (Page 2637)
Replacing FBD elements (Page 2638)
Adding additional inputs and outputs to FBD elements (Page 2639)
Removing instruction inputs and outputs (Page 2640)
Enabling and disabling the EN/ENO mechanism (Page 2641)
Deleting FBD elements (Page 2642)

Pasting an FBD element from the clipboard

Requirement

An FBD element is available.

Procedure

To paste an FBD element from the clipboard, follow these steps:

1. Copy an FBD element or cut an FBD element.
2. Right-click the point in the network where you want to paste the element.
3. Select "Paste" in the shortcut menu.

See also

Selecting FBD elements (Page 2635)
Copying FBD elements (Page 2636)
Cutting FBD elements (Page 2637)
Replacing FBD elements (Page 2638)
Adding additional inputs and outputs to FBD elements (Page 2639)
Removing instruction inputs and outputs (Page 2640)
Enabling and disabling the EN/ENO mechanism (Page 2641)
Deleting FBD elements (Page 2642)

Replacing FBD elements

You can easily exchange FBD elements with other FBD elements of the same type. This has the advantage that the parameters are retained and need not be entered again. For example, you can exchange OR and AND, RS-FlipFlop and SR-FlipFlop, comparison functions or jump instructions.

Requirements

A network with at least one FBD element is present.

Procedure

To replace an FBD element with another FBD element, follow these steps:

1. Select the FBD element that you want to replace.
   If elements compatible with the selected FBD element are available, a triangle will appear in the upper right-hand corner of the element.
2. Position the cursor above the triangle of the FBD element.
   A drop-down list is displayed.
3. From the drop-down list, select the FBD element that you want to use to replace the existing FBD element.

See also

Selecting FBD elements (Page 2635)
Copying FBD elements (Page 2636)
Cutting FBD elements (Page 2637)
Pasting an FBD element from the clipboard (Page 2637)
Adding additional inputs and outputs to FBD elements (Page 2639)
Removing instruction inputs and outputs (Page 2640)
Enabling and disabling the EN/ENO mechanism (Page 2641)
Deleting FBD elements (Page 2642)

Adding additional inputs and outputs to FBD elements

Introduction

You can expand several FBD elements with additional inputs that execute arithmetic or binary operations. Such elements are, for example, the instructions "Add" (ADD), "Multiply" (MUL), AND or OR. You can expand the "MOVE value" (MOVE) and "Demultiplex" (DEMUX) instruction boxes by adding additional outputs.

The name of the new inputs and outputs is comprised of the type of inserted element and a consecutive number. The name of a new input is may be "IN2"; the name of a new output may be "OUT2".

Requirements

An FBD element is available that permits the insertion of additional inputs and outputs.

Inserting an additional input

To add an additional input to the box of an FBD element, follow these steps:

1. Right-click on an existing input of the FBD element.
2. Select "Insert input" in the shortcut menu.
   An additional input is added to the box of the FBD element.

Or:

1. Click on the yellow star symbol beside the last input in the instruction box.
   An additional input is added to the box of the FBD element.

Inserting an additional output

To add an additional output to the box of an FBD element, follow these steps:

1. Right-click on an existing output of the FBD element.
2. Select "Insert output" from the shortcut menu.
   An additional output is added to the box of the FBD element.

Or:

1. Click on the yellow star symbol beside the last output of the instruction box.
   An additional output is added to the box of the FBD element.

See also

Inserting FBD elements (Page 2621)

Selecting FBD elements (Page 2635)

Copying FBD elements (Page 2636)

Cutting FBD elements (Page 2637)

Pasting an FBD element from the clipboard (Page 2637)

Replacing FBD elements (Page 2638)

Removing instruction inputs and outputs (Page 2640)

Enabling and disabling the EN/ENO mechanism (Page 2641)

Deleting FBD elements (Page 2642)

Removing instruction inputs and outputs

Introduction

Inputs and outputs which you have added to an instruction can be removed.

Requirement

An FBD element is available, which you have expanded with additional inputs or outputs.

Remove input

To remove an input, follow these steps:

1. Select the input that you want to remove.
2. Select the "Delete" command in the shortcut menu.
   The input of the FBD element is removed.

Remove output

To remove an output, follow these steps:

1. Select the output that you want to remove.
2. Select the "Delete" command in the shortcut menu.
   The output of the FBD element will be removed.

See also

Inserting FBD elements (Page 2621)

Selecting FBD elements (Page 2635)

Copying FBD elements (Page 2636)

Cutting FBD elements (Page 2637)

Pasting an FBD element from the clipboard (Page 2637)

Replacing FBD elements (Page 2638)

Adding additional inputs and outputs to FBD elements (Page 2639)

Enabling and disabling the EN/ENO mechanism (Page 2641)

Deleting FBD elements (Page 2642)

Enabling and disabling the EN/ENO mechanism

In LAD and FBD, certain instructions have an enable output ENO and thus use the EN/ENO mechanism. This allows you to query runtime errors in instructions and react to them. In order to increase the performance of the CPU, the EN/ENO mechanism is disabled in the default setting. This means that you are not initially able to react to runtime errors of the instruction using the ENO value. However, you can enable the EN/ENO mechanism again at any time, if required.

You can enable the EN/ENO mechanism individually for each instruction in order to generate the ENO. If you enable the EN/ENO mechanism for an instruction, other instructions that you subsequently add to your program are also inserted with the EN/ENO mechanism enabled. You can disable the EN/ENO mechanism again at any time if you do not want to use the evaluation of ENO for an instruction. Further instructions that you subsequently add to your program will then be inserted without the EN/ENO mechanism.

See also: Basics of the EN/ENO mechanism (Page 2447)

Activating the EN/ENO mechanism

Proceed as follows to activate the EN/ENO mechanism of an instruction:

1. In your program, right-click the instruction at which you want to activate the EN/ENO mechanism.
2. Select the "Generate ENO" command from the shortcut menu.
   The ENO value is again generated for the instruction. Other instructions are inserted with the enable output.

Deactivating the EN/ENO mechanism

Proceed as follows to deactivate the EN/ENO mechanism of an instruction:

1. In your program, right-click the instruction at which you want to deactivate the EN/ENO mechanism.
2. Select the "Do not generate ENO" command from the shortcut menu.
   The ENO value is no longer generated for the instruction. Other instructions are inserted without enable output.

See also

Selecting FBD elements (Page 2635)
Copying FBD elements (Page 2636)
Cutting FBD elements (Page 2637)
Pasting an FBD element from the clipboard (Page 2637)
Replacing FBD elements (Page 2638)
Adding additional inputs and outputs to FBD elements (Page 2639)
Removing instruction inputs and outputs (Page 2640)
Deleting FBD elements (Page 2642)

Deleting FBD elements

Requirement

An FBD element is available.

Procedure

To delete an FBD element, follow these steps:

1. Right-click the FBD element that you want to delete.
2. Select the "Delete" command in the shortcut menu.

See also

Inserting FBD elements (Page 2621)
Selecting FBD elements (Page 2635)
Copying FBD elements (Page 2636)
Cutting FBD elements (Page 2637)
Pasting an FBD element from the clipboard (Page 2637)
Replacing FBD elements (Page 2638)
Adding additional inputs and outputs to FBD elements (Page 2639)

Removing instruction inputs and outputs (Page 2640)

Enabling and disabling the EN/ENO mechanism (Page 2641)

Inserting operands in FBD instructions

Inserting operands

The character strings "<???>", "<??.?>" and "..." are inserted as placeholders for the parameters when a FBD element is inserted. The "<???>" and "<??.?>" strings displayed in red indicate parameters that need to be connected. The "..." string displayed in black indicates parameters that may be connected. "<??.?>" stands for Boolean placeholders.

Note

To display the available data types in a tooltip, move the cursor over the placeholder.

Requirement

An FBD element is available.

Procedure

To connect the parameters of an FBD element, follow these steps:

1. Click the placeholder of the parameter.
   An input field is opened.
2. Enter the corresponding parameters, for example a PLC tag, a local tag or a constant.

   Note

   If you enter the absolute address of a parameter that has already been defined, this absolute address will be changed to the symbolic name of the parameter as soon as the input is confirmed. If you have not yet defined the parameter, a new tag with this absolute address and the default name "Tag_1" will be entered in the PLC tag table. When you confirm your input, the absolute address will be replaced with the symbolic name "Tag_1".

3. Confirm the parameter with the Enter key.
4. If you have not yet defined the parameter, you can define it directly in the program editor using the shortcut menu.
   See also: "Declaring PLC tags in the program editor (Page 2482)".

Or drag from it the PLC tag table:

1. In the project tree, select the "PLC tags" folder and open the PLC tag table.
2. If you have opened the PLC tag table, drag the desired tag to the corresponding location in your program. If you have not opened the PLC tag table yet, open the detail view now. Drag the desired tag from the detail view to the appropriate place in your program.

Or drag from it the block interface:

1. Open the block interface.
2. Drag the desired operand from the block interface to the corresponding location in your program.

Result

- If the syntax is error-free, the displayed parameter is black.
- If there is an error in the syntax, the cursor stays in the input field and a corresponding error message is displayed in the inspector window in the "Info > Syntax" register.

See also

Using and addressing operands (Page 2197)

Using autocompletion (Page 2540)

Wiring hidden parameters

Introduction

Depending on the CPU used, you can use complex instructions in your program that are dispatched with the TIA portal. These instructions can contain parameters that are declared as hidden.

If an instruction contains hidden parameters, the instruction box has a small arrow on the lower edge. You can recognize hidden parameters by their white font.

You can show and wire the hidden parameters at any time.

Showing or hiding hidden parameters

To show or hide hidden parameters, follow these steps:

1. Click on the down arrow at the bottom edge of the instruction box to show hidden parameters.
2. Click on the up arrow at the bottom edge of the instruction box to hide hidden parameters.

Wiring hidden parameters

To wire parameters, follow these steps:

1. Wire the hidden parameters like normally visible parameters.
   The hidden parameter is transformed into a visible parameter.

See also

Using libraries (Page 415)

Displaying or hiding variable information

Introduction

You can display the following information about the tags to be used in the Program editor:

- Name of the tag
- Address of the tag
- Simple or hierarchical comments for tag documentation

The information is taken from the block interface for local tags and DB tags and from the PLC tag table for tags that are valid CPU-wide.

You can display the tag information either for all the blocks or for individually opened blocks. If you display the tag information for all the blocks, the tag information for all the blocks currently opened and opened in future is shown.

You can hide the tag information at any time again. If you have hidden the tag information for all blocks, you can display it again for individual blocks that are open.

When you select the display of tag information with hierarchical comments, the comments of the higher structure levels are also displayed for structured tags. The comments are shown in brackets after the tag comment; the comments of the individual levels are separated by a dot. If there is no comment for a tag on a structure level, it is not displayed. This is indicated by two successive dots.

Displaying or hiding tag information for all the blocks

Proceed as follows to display or hide the tag information for all the blocks:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. If you want to display the tag information, select the "Show" option or the "Tag information with hierarchical comments" option in the drop-down list, depending on whether you want to display simple or hierarchical comments.
4. If you want to hide the tag information, select the option "Collapse" in the "Tag information" drop-down list.
   The tag information is displayed or hidden for all the blocks. When you open additional blocks, the tag information is displayed or hidden depending on the selected setting.

Displaying or hiding tag information for an opened block

Proceed as follows to display or hide the tag information for an opened block:

1. If you want to display the tag information, select the "Show tag information" option or the "Tag information with hierarchical comments" option in the "Shows the tag information" drop-down list, depending on whether you want to display simple or hierarchical comments.
2. If you want to hide the tag information, select the option "Hide tag information" in the "Shows the tag information" drop-down list.
   The tag information is displayed or hidden.

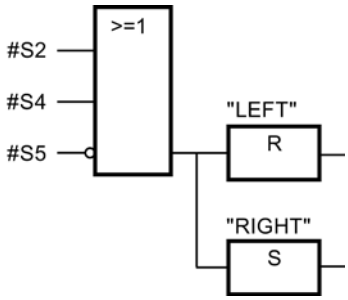
Branches in FBD

Basic information on branches in FBD

Definition

You can use the Function Block Diagram (FBD) programming language to program parallel branches. This is done using branches that are inserted between the boxes. You can insert additional boxes within the branch and in this way build up complex function block diagrams.

The figure below shows an example of the use of branches:



See also

Rules for branches in FBD (Page 2646)

Inserting branches in FBD networks (Page 2647)

Deleting branches in FBD networks (Page 2647)

Rules for branches in FBD

Rules

The following rules apply to the use of branches in FBD:

- Branches are opened downward.
- Branches can be inserted only between FBD elements.
- To delete a branch, you must delete all FBD elements, including the branch itself.
- If you delete the connection between two branches, the FBD elements of the interrupted branch will be positioned freely in the network.

See also

Basic information on branches in FBD (Page 2646)

Inserting branches in FBD networks (Page 2647)

Deleting branches in FBD networks (Page 2647)

Inserting branches in FBD networks

Requirement

A network is available.

Procedure

To insert a new branch in a network, follow these steps:

1. Open the "Instructions" task card.
2. Navigate to "General > Branch" in the "Basic instructions" palette.
3. Drag the element from the "Elements" pane to the a required location on a connection line between two boxes.

See also

Rules for branches in FBD (Page 2646)

Basic information on branches in FBD (Page 2646)

Deleting branches in FBD networks (Page 2647)

Deleting branches in FBD networks

Requirement

A branch is available.

Procedure

To delete a branch, follow these steps:

1. Select the connection line that links the branch to the main branch.
2. Select the "Delete" command in the shortcut menu.

Result

The branch is now deleted. Boxes connected to the deleted branch are placed freely within the network.

See also

Rules for branches in FBD (Page 2646)

Basic information on branches in FBD (Page 2646)

Inserting branches in FBD networks (Page 2647)

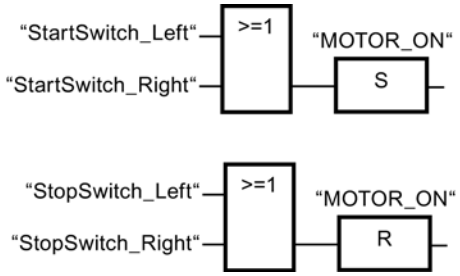
Logic paths in FBD

Basic information on logic paths in FBD

Use of logic paths

The user program will be mapped in one or more networks. The networks can contain one or more logic paths on which the binary signals are arranged in the form of boxes.

The following figure shows an example of the use of several logic paths within a network:



Rules

Remember the following rules when using logic paths:

- Connections are not permitted between logic paths.
- Only one jump instruction is permissible per network. The positioning rules for jump instructions remain valid.

Executing logic paths

Logic paths are executed from top to bottom and from left to right. This means that the first instruction in the first logic path of the first network is executed first. All instructions of this logic path are then executed. After this come all other logic paths of the first network. The next network is executed only after all logic paths have first been executed.

When jumps are used the regular execution of the logic paths is circumvented and the instruction is executed at the jump destination.

Differences between branches and logic paths

The difference between branches and logic paths is that the logic paths are independent branches that can also stand in a different network. Branches, on the other hand, permit the programming of a parallel connection and have a common preceding logic operation.

See also

Inserting a logic path (Page 2649)

Deleting a logic operation path (Page 2649)

Inserting a logic path

Requirement

- A block is open.
- A network is available.

Procedure

To insert a new logic path in a network, follow these steps:

1. Insert any instruction in a network in such a way that it has no connection to existing instructions.
   A new logic path is inserted.
2. Insert an assignment at the end of the new logic path.
3. Insert additional instructions in the new logic path.

See also

Basic information on logic paths in FBD (Page 2648)

Deleting a logic operation path (Page 2649)

Deleting a logic operation path

Requirement

A logic path is available.

Procedure

To delete a logic path, proceed as follows:

1. Hold down the left mouse button and draw a frame around the logic path. At the same time, make sure that you select all instructions of the logic path. Alternatively, you can hold down the <Shift> key and select the first the last instruction of the logic path.
2. Right-click on one of the instructions in the logic path.
3. Select the "Delete" command in the shortcut menu.

See also

Basic information on logic paths in FBD (Page 2648)

Inserting a logic path (Page 2649)

Creating STL programs

STL Basics

STL programming language

Overview of the STL (Statement List) programming language

STL is a text-based programming language you can use to program logic blocks.

The STL program is divided into networks. Each network can contain one or more rows. The numbering of the rows begins with 1 in each network and is continued in ascending order with each new row. The individual STL instructions are programmed in the rows of a network, where only one STL instruction can be specified per row. Each statement represents an instruction to the CPU. The CPU executes the instructions from top to bottom.

The following example shows the STL programming of a network:

STL	Explanation
A"Tag_Input_1"	// Check whether the signal state of the operand is "1" and AND with current RLO
A "Tag_Input_2"	// Check whether the signal state of the operand is "1" and AND with current RLO
S "Tag_Output"	// Sets the operand to "1" if the RLO is "1"

Data exchange in STL

S7-300/400 data exchange

For S7-300/400 CPUs, information is exchanged between different memory areas in STL via accumulators. Accumulators are special registers in the process which serve as buffers. All S7-300/400 CPUs have two buffers, namely accumulator 1 (ACCU 1) and accumulator 2 (ACCU 2). The S7-400 CPUs and the S7-318 CPU also have two additional buffers, accumulators 3 (ACCU 3) and 4 (ACCU 4).

S7-1500 data exchange

Although S7-1500 CPUs no longer have a register in the processor, they can emulate the address register, ACCU 1, ACCU 2, the status word and the data block registers DB and DI. Data is exchanged primarily via the block interface, global data blocks or via PLC tags. However, using accumulators for data exchange slows downs the program processing.

When programming in STL for S7-1500, note the following basic instructions regarding the register:

- The contents of the registers, accumulators and the status word are only available within STL networks. When an LAD or FBD network follows an STL network, you cannot access the register contents that were previously set in STL from the LAD or FBD network. However, the register contents are available again in a downstream STL network.
  The RLO bit is an exception: It is set to "undefined" at a language change and is no longer available in downstream networks.
- The values from the registers, accumulators and the status word are never transferred to called blocks. The instructions "CC" and "UC" are the only exceptions. If you use "UC" or "CC" and want to pass the parameters to the called block by using registers, the status word or accumulators, you have to select the "Parameter passing via registers" option in the properties of the called block. Note that this option is only available for STL blocks with standard access and the block can have no formal parameters. If the option is enabled, you can transfer register contents between blocks. The RLO bit is an exception here as well: It is set to "undefined" when the block is exited and is no longer available after a block call.
- The data block register DB is set to "0" after each access to the data block with specification of a fully qualified address (for example, %DB10.DBW10). Subsequent partially qualified access leads to an error during compilation.
- To pass an error message to the calling block, you can use the BR bit. You first need to store the error message in the called block in the BR bit with the instruction "SAVE". You can then read the BR bit in the calling block.
- If you symbolically address (e.g. with the instruction L #myIn) a local formal parameter from the block interface in S7-1500, you always access the data block that you specified as an instance in the block call. Although the OPN DI, L AR2, +AR2, CDB, CAR instructions change the contents of the DI or address register, the registers are no longer evaluated when local formal parameters are addressed.

Load and transfer

During data exchange, the direction of the information flow is marked. If data is transferred from a memory area to accumulator 1, this is referred to as loading data. Data transfer from accumulator 1 to a memory area is referred to as "transfer". Load and transfer instructions are used for load and transfer in STL.

Load and transfer is necessary above all for editing digital values. If you want to add two values, for example, you have to load them to the buffer and execute the appropriate instruction. First, the first value to be added is loaded to accumulator 1. When the second summand is loaded, the first value, which is located in accumulator 1, is moved to accumulator 2. After execution of the add instruction (e.g. I+), the sum is saved in accumulator 1. The sum can be transferred to any operand using the transfer instruction.

All accumulators are 32 bits long and are organized byte-oriented. Data transfer between the memory areas and accumulator 1 may be byte-oriented, word-oriented or double word-oriented.

Data exchange between accumulators

Data can be exchanged between individual accumulators. The accumulator instructions "TAK", "PUSH" and "POP" are used for this purpose. This allows the transfer of data between the

available accumulators and the exchange of bytes in Accumulator 1. Please note that the accumulator instructions "ENT" and "LEAVE" are then no longer available for the S7-1500.

See also

Load (Page 5994)

Transfer (Page 6003)

Accumulators (Page 6102)

Parameter assignment via registers during block call in STL (Page 2652)

Parameter passing via registers

Parameter assignment via registers during block call in STL

Block calls can be implemented using the following instructions in STL:

- CALL: Call block
- UC: Unconditional block call
- CC: Conditional block call

In a block call, values are usually transferred between the called and the calling block. The data exchange options are different for the various CPU families.
See also: Data exchange in STL (Page 2650)

Values transferred from the calling block to the called block

The following table indicates which registers, accumulators and status bits are directly available after a block call in the called block prior to processing the first network:

	S7-300/400		S7-1500	
	UC / CC	CALL	UC / CC with option "Parameter passing via registers"	CALL
Status bit OS	0	0	0	0
Status bit OV	W	W	W	0
Status bit CC0	W	W	W	0
Status bit CC1	W	W	W	0
BR	W	W	W	1
RLO	W	W	W	A
Accu 1	W	W	W	A
Accu 2	W	W	W	A
Address register 1	W	W	W	0
Address register 2	W	Call FC: W Call FB: Instance offset	W	Call FC: 0 Call FB: Instance offset

	S7-300/400		S7-1500	
DB register	W	W	W	A
DI register	W	Call FC: W Call FB: Instance DB number	W	Call FC: A Call FB: Instance DB number

Legend:
0 = Register is set to "0" at block change.
1 = Register is set to "1" at block change.
W = Register content can be set in the calling block and read in the called block.
A = Register content is not transferred to the called block. The contents are undefined, access results in an error during compilation.
Instance offset = Offset in instance DB
Instance DB number = Number of the instance DB

Values returned from the called block to the calling block

The following table indicates which registers, accumulators and status bits are directly available after processing of the block call in the calling block:

	S7-300/400		S7-1500	
	UC / CC	CALL	UC / CC with option "Parameter passing via registers"	CALL
Status bit OS	0	0	0	0
Status bit OV	W	W	W	-
Status bit CC0	W	W	W	-
Status bit CC1	W	W	W	-
BR	W	W	W	W
RLO	W	W1	W	A
Accu 1	W	W1	W	-
Accu 2	W	W1	W	-
Address register 1	W	W1	W	-
Address register 2	W	Call FC: W Call FB: -	W	-
DB register	-	X	-	-
DI register	-	X	-	-

Legend:
0 = Register is set to "0" at block change.
1 = Register is set to "1" at block change.
W = Register content can be set in the called block and read in the calling block.
W1 = The returned value can be changed by the output parameters and in/out parameters of the called block.
- = Register contents is not returned to the calling block, content is not changed by the block call.
A = Register content is not transferred to the called block. The content is undefined, access results in an error during compilation.
X = Register content is not returned to the calling block. The value can be changed implicitly

by the parameters of the called block.

See also

Passing values using registers or the status word with a block call (Page 221)
Passing values using registers with the CC and UC instructions (Page 222)
Activating parameter passing via registers (Page 2654)

Activating parameter passing via registers

Requirement

- The "Optimized block access" option is disabled.
- The block interface does not include any parameters.

Procedure

To enable parameter passing via registers, follow these steps:

1. Open the "Program blocks" folder in the project tree.
2. Right-click the block for which you want to set up the parameter passing via registers.
3. Select the "Properties" command in the shortcut menu.
4. The properties dialog box of the block opens.
5. Click "Attributes" in the area navigation.
6. Select the option "Parameter passing via registers".
7. Confirm your entries with "OK".

See also

Parameter assignment via registers during block call in STL (Page 2652)
Data exchange in STL (Page 2650)

Settings for STL

Overview of the settings for STL

Overview

The following table shows the settings that you can make for STL:

Group	Setting	Description
View	Line break on Enter	Inserts a line break when you press the Enter key.

See also

Changing the settings (Page 2655)

Changing the settings

Procedure

To change the settings, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. Change the settings.

Result

The change will be loaded directly, there is no need to save it explicitly.

See also

Overview of the settings for STL (Page 2655)

Working with networks

Using networks

Function

The user program is created in the block within networks. For a code block to be programmed, it must contain at least one network. To achieve a better overview of the user program, you can also subdivide your program into several networks.

See also

Inserting networks (Page 2656)
Selecting networks (Page 2657)
Copying and pasting networks (Page 2658)
Deleting networks (Page 2658)
Expanding and collapsing networks (Page 2659)
Inserting network title (Page 2660)
Entering a network comment (Page 2661)
Navigating networks (Page 2662)

Inserting networks

Requirement

A block is open.

Procedure

To insert a new network, follow these steps:

1. Select the network after which you want to insert a new network.
2. Select the "Insert network" command in the shortcut menu.

Result

A new empty network is inserted into the block.

See also

Using networks (Page 2656)
Selecting networks (Page 2657)
Copying and pasting networks (Page 2658)

Deleting networks (Page 2658)
Expanding and collapsing networks (Page 2659)
Inserting network title (Page 2660)
Entering a network comment (Page 2661)
Navigating networks (Page 2662)

Selecting networks

Requirements

A network is available.

Selecting a network

To select a network, follow these steps:

1. Click the title bar of the network that you want to select.

Selecting several networks

Proceed as follows to select several individual networks:

1. Press and hold down the <Ctrl> key.
2. Click all the networks that you want to select.

To select several successive networks, follow these steps:

1. Press and hold down the <Shift> key.
2. Click the first network that you want to select.
3. Click the last network that you want to select.
   The first and last networks and all those in between are selected.

See also

Using networks (Page 2656)
Inserting networks (Page 2656)
Copying and pasting networks (Page 2658)
Deleting networks (Page 2658)
Expanding and collapsing networks (Page 2659)
Inserting network title (Page 2660)
Entering a network comment (Page 2661)
Navigating networks (Page 2662)

Copying and pasting networks

Copied networks can be pasted within the block or in another block. Networks that were created in LAD or FBD can also be inserted in blocks of the respective other programming language.

Requirement

A network is available.

Procedure

To copy and paste a network, follow these steps:

1. Select the network or networks to be copied.
2. Select "Copy" in the shortcut menu.
3. Select the network after which you want to paste in the copied network.
4. Select "Paste" in the shortcut menu.

See also

Using networks (Page 2656)
Inserting networks (Page 2656)
Selecting networks (Page 2657)
Deleting networks (Page 2658)
Expanding and collapsing networks (Page 2659)
Inserting network title (Page 2660)
Entering a network comment (Page 2661)
Navigating networks (Page 2662)

Deleting networks

Requirement

A network is available.

Procedure

To delete a network, follow these steps:

1. Select the network that you want to delete.
2. Select the "Delete" command in the shortcut menu.

See also

Using networks (Page 2656)
Inserting networks (Page 2656)
Selecting networks (Page 2657)
Copying and pasting networks (Page 2658)
Expanding and collapsing networks (Page 2659)
Inserting network title (Page 2660)
Entering a network comment (Page 2661)
Navigating networks (Page 2662)

Expanding and collapsing networks

Requirements

A network is available.

Opening and closing a network

To open a network, follow these steps:

1. Click on the right arrow in the network title bar.

To close a network, follow these steps:

1. Click on the down arrow in the network title bar.

Opening and closing all networks

To open and close all networks, follow these steps:

1. In the toolbar, click "Open all networks" or "Close all networks".

See also

Using networks (Page 2656)
Inserting networks (Page 2656)
Selecting networks (Page 2657)
Copying and pasting networks (Page 2658)
Deleting networks (Page 2658)
Inserting network title (Page 2660)
Entering a network comment (Page 2661)
Navigating networks (Page 2662)

Inserting network title

The network title is the header of a network. The length of the network title is limited to one line. You can enter the title manually or set it automatically. When you set it automatically, you can do this for individual networks or use the settings to specify that the network title is always set automatically.

For automatic insertion of the network title, the comment of the operand in one of the following instructions in the network is evaluated:

- Assignment
- Set output
- Reset output

The instruction that is listed first in the network is used.

The network title is only inserted automatically when the following conditions are fulfilled:

- The network does not have a title yet.
- The operand of the instruction used for the comment has a comment.

Note

Note the following restrictions for automatic insertion of the network title:

- The network title is not adapted if you change the comment of the operand at a later time.
- The network title is not adapted if you change the operand of the instruction.
- The network title is only set by the writing instructions listed above.
- If the operand is of the data type array, the comment of the array is used and not the comments of the array elements.
- Comments of invalid operands are not taken into consideration.

Entering the network title manually

To enter a network title, follow these steps:

1. Click on the title bar of the network.
2. Enter the network title.

Setting the network title automatically

To specify that the network titles are always set automatically, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. Select the "Set network title automatically" check box in the "Additional settings" group.
   The network titles are set automatically as of this time if the conditions listed above are fulfilled.

To set an individual network title automatically, follow these steps:

1. Right-click "Network <Number of the network>" in the title bar of a network.
2. Select the "Set network title automatically" command in the shortcut menu.
   The title of the selected network is set based on the comment of the operand if the conditions listed above are fulfilled.

See also

Using networks (Page 2656)

Inserting networks (Page 2656)

Selecting networks (Page 2657)

Copying and pasting networks (Page 2658)

Deleting networks (Page 2658)

Expanding and collapsing networks (Page 2659)

Entering a network comment (Page 2661)

Navigating networks (Page 2662)

Entering a network comment

You can use network comments to provide comments on the program contents of individual networks. For example, you can indicate the function of the network or draw attention to special characteristics.

Requirement

A network is available.

Procedure

To enter a network comment, follow these steps:

1. Click on the right arrow before the network title.
2. If the comment area is not visible, click "Network comments on/off" in the toolbar.
   The comment area is displayed.
3. Click "Comment" in the comment area.
   The "Comment" text passage is selected.
4. Enter the network comment.

See also

Using networks (Page 2656)

Inserting networks (Page 2656)

Selecting networks (Page 2657)

Copying and pasting networks (Page 2658)
Deleting networks (Page 2658)
Expanding and collapsing networks (Page 2659)
Inserting network title (Page 2660)
Navigating networks (Page 2662)

Navigating networks

You can navigate straight to a specific position within a block.

Procedure

To navigate to a specific position within a block, follow these steps:

1. Right-click in the code area of the programming window.
2. Select the "Go to > Network/line" command in the shortcut menu.
   The "Go to" dialog will open.
3. Enter the network to which you want to navigate.
4. Enter the line number of the network to which you want to navigate.
5. Confirm your entry with "OK".

Result

The relevant line will be displayed if this is possible. If the network or line requested does not exist, the last existing network or the last existing line in the network requested will be displayed.

See also

Using networks (Page 2656)
Inserting networks (Page 2656)
Selecting networks (Page 2657)
Copying and pasting networks (Page 2658)
Deleting networks (Page 2658)
Expanding and collapsing networks (Page 2659)
Inserting network title (Page 2660)
Entering a network comment (Page 2661)

Inserting STL instructions

Rules for STL

Rules

The following rules must be observed when entering STL instructions incrementally:

- An instruction consists of the specification of an optional jump label, the operation, the operand, and an optional comment. Example:
  `M001: A I1.0 //comment`
- Each instruction is in its own separate line.
- You can enter a maximum of 800 networks per block.
- The entry of operations or absolute addresses is not case-sensitive.
- If you would like support for block calls, you must program the called block before the calling block.

Impermissible instructions are highlighted in red. A rollout and the inspector window give information for troubleshooting.

Jumps

For jump operations, make sure that the jump destination is always at the start of a logic sequence and is within a different nesting level (not applicable to CPU 318-2). The jump designation must not be located within a logic sequence or a nesting level.

See also

Entering STL instructions manually (Page 2663)

Inserting STL instructions using the "Instructions" task card (Page 2665)

Using favorites in STL (Page 2669)

Inserting block calls in STL (Page 2671)

Inserting comments (Page 2676)

Inserting operands (Page 2667)

Entering STL instructions manually

When an empty network is created automatically or manually, a two-line entry field is generated within the network. The STL instructions are entered in this entry field.

Note

S7-1500: Instructions from the "Instructions" task card that use emulated registers and accumulators cannot use 64-bit tags, such as LInt, as operands. If you want to use 64-bit tags, use the instructions from the "Long Functions" global library.

Requirement

A network is available.

Procedure

To enter STL instructions in a network, follow these steps:

1. Click the first empty row of the entry field in a network.
2. Enter an instruction.
3. Enter a space.
4. Enter the operand.

   Note

   You can also use a drag-and-drop operation to insert an already defined operand from the PLC tag table or from the detail view.

5. Enter a line comment. This step is optional.
6. Confirm the instruction with the Enter key.
7. Repeat steps 2 to 6 until you have entered all the required instructions.

Result

After the instruction is closed by pressing the Enter key, the program editor performs the following actions:

- Syntax check: After the check, faulty entries are displayed in red italics. In addition, you also receive a detailed error message in the "Syntax" tab of the information window.
- Change of lower and upper case: Instructions are changed to upper case.
- If you have entered absolute addresses in symbolic programming, these will be converted into symbolic names. If no symbolic name is available, "Tag_< number>" is inserted.
- Block call: the necessary parameters are inserted and displayed.
- An additional line is inserted in the entry field.

See also

Rules for STL (Page 2663)

Inserting STL instructions using the "Instructions" task card (Page 2665)

Inserting operands (Page 2667)

Using favorites in STL (Page 2669)

Inserting block calls in STL (Page 2671)

Inserting comments (Page 2676)

Inserting STL instructions using the "Instructions" task card

When an empty network is created automatically or manually, a two-line entry field is generated within the network. The STL instructions are entered in this entry field.

Note

S7-1500: Instructions from the "Instructions" task card that use emulated registers and accumulators cannot use 64-bit tags, such as LInt, as operands. If you want to use 64-bit tags, use the instructions from the "Long Functions" global library.

Requirement

A network is available.

Procedure

To insert STL instructions into a network using the "Instructions" task card, follow these steps:

1. Open the "Instructions" task card.
2. To insert the instruction, select one of the following steps:
   - Go to the STL element you want to insert and drag it to the required location in the network.
   - Select the location in the network at which you want to insert the element and then double-click on the element you want to insert.
3. Enter the required operands.

   Note

   You can also use a drag-and-drop operation to insert an already defined operand from the PLC tag table, a data block, the detail view or the block interface.

4. Enter a line comment. This step is optional.
5. Repeat steps 2 to 4 until you have entered all the required instructions.

Result

After the instruction is closed by pressing the Enter key, the program editor performs the following actions:

- Syntax check: After the check, faulty entries are displayed in red italics. In addition, you also receive a detailed error message in the information window.
- Change of lower and upper case: Instructions are changed to upper case.
- If you have entered absolute addresses in symbolic programming, these will be converted into symbolic names. If no symbolic name is available, "Tag_< number>" is inserted.
- Block call: the necessary parameters are inserted and displayed.
- An additional line is inserted in the entry field.

See also

Rules for STL (Page 2663)
Entering STL instructions manually (Page 2663)
Inserting operands (Page 2667)
Using favorites in STL (Page 2669)
Inserting block calls in STL (Page 2671)
Inserting comments (Page 2676)

Selecting the data type of an STL instruction

Selecting a data type

Introduction

Some instructions can be executed with several different data types. If you use one of these instructions in the program, you have to specify a valid data type for the instruction at the specific point in the program.

The valid data types for an instruction are listed in the instruction drop-down list. You specify the data type of the instruction by selecting an entry from the drop-down list. If the data type of an operand differs from the data type of the instruction and cannot be converted implicitly, the operand is displayed in red and a rollout with the corresponding error message appears.

Defining the data type of an instruction

Introduction

Some instructions can be executed with several different data types. When you insert such instructions into your program, you must specify the data type for these instructions at the actual point in the program.

Procedure

To define the data type of an instruction using the drop-down list, follow these steps:

1. Insert the instruction at the required point in the program using drag-and-drop.
   The entry "???" (undefined) is displayed in the drop-down list of the inserted instruction.
2. Click the triangle in the upper corner of the drop-down list.
   The drop-down list opens to display the data types valid for the instruction.
3. Select a data type from the drop-down list.
   The selected data type is displayed.

Inserting operands

You can enter the operands of an STL instruction either manually or by dragging them from the PLC tag table, a data block, the detail view, or the block interface.

Note

S7-1500: Instructions from the "Instructions" task card that use emulated registers and accumulators cannot use 64-bit tags, such as LInt, as operands. If you want to use 64-bit tags, use the instructions from the "Long Functions" global library.

Procedure

To enter the operand of an STL instruction manually, follow these steps:

1. Position the cursor to the right of the operation.
2. Enter a space and then the operand.

   Note

   If you enter the absolute address of a parameter that has already been defined in symbolic programming, this absolute address will be changed to the symbolic name of the parameter as you confirm your input. If you have not yet defined the parameter, a new tag with this absolute address and the default name "Tag_1" will be entered in the PLC tag table. When you confirm your input, the absolute address will be replaced with the symbolic name "Tag_1".

3. Confirm the parameter with the Enter key.
4. If you have not yet defined the parameter, you can define it directly in the program editor using the shortcut menu.
   See also: "Declaring tags directly in the program editor (Page 2482)"

Or drag from it the PLC tag table:

1. In the project tree, select the "PLC tags" folder or open the PLC tag table.
2. If you have opened the PLC tag table, drag the symbol from the first column of the desired tag to the appropriate place in your program. If you have not opened the PLC tag table yet, open the detail view now. Drag the desired tag from the detail view to the appropriate place in your program.

Or drag from it the block interface:

1. Open the block interface.
2. Drag the required operand from the block interface to the instruction window.

See also

Rules for STL (Page 2663)

Entering STL instructions manually (Page 2663)

Inserting STL instructions using the "Instructions" task card (Page 2665)

Using favorites in STL (Page 2669)

Inserting block calls in STL (Page 2671)

Inserting comments (Page 2676)

Using and addressing operands (Page 2197)

Using autocompletion (Page 2540)

Displaying or hiding tag information

Introduction

You can hide or show simple or hierarchical comments on the documentation of the tags, regardless of the absolute or symbolic representation of the operands. For local tags and DB tags, this information is obtained from the block interface. For tags that are valid for the entire CPU, the information is obtained from the PLC tag table.

You can display the tag information either for all the blocks or for individually opened blocks. If you display the tag information for all the blocks, the tag information for all blocks currently opened and opened in future is shown.

You can hide the tag information at any time again. If you have hidden the tag information for all blocks, you can display it again for individual ones that you have opened.

If you select the display of tag information with hierarchical comments, the comments of the higher structure levels of structured tags will also be displayed. The display is in brackets after the comment of the tags; the comments of the individual levels are separated by a period. If there is no comment at a structure level for a tag, it is omitted in the display and this is recognizable because there are two periods.

Displaying or hiding tag information for all blocks

Follow the steps below to display or hide the tag information for all blocks:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. If you want to show the tag information, either select the "Expand" option in the "Tag information" drop-down list or the "Tag information with hierarchy" depending on whether you want to display simple or hierarchical comments.
4. If you want to hide the tag information, select the "Collapse" option in the "Tag information" drop-down list.
   The tag information is displayed or hidden for all blocks. When you open further blocks, the tag information is displayed or hidden depending on the selected setting.

Displaying or hiding tag information for an opened block

Follow the steps below to display or hide the tag information for an opened block:

1. If you want to show the tag information, either select the "Show tag information" option in the "Shows the tag information" drop-down list or the "Tag information with hierarchy" depending on whether you want to display simple or hierarchical comments.
2. If you want to hide the tag information, select the "Hide tag information" option in the "Hides tag information" drop-down list.
   The tag information is displayed or hidden.

Using favorites in STL

Adding STL elements to Favorites

Requirement

- A block is open.
- The multipane mode is set for the "Instructions" task card or the Favorites are also displayed in the editor.

Procedure

To add SCL instructions to the Favorites, follow these steps:

1. Open the "Instructions" task card.
2. Maximize the "Basic instructions" pane.
3. Navigate in the "Basic instructions" pane to the instruction that you want to add to the Favorites.
4. Drag-and-drop the instruction into the "Favorites" pane or into the Favorites area in the program editor.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Inserting STL instructions using favorites (Page 2670)

Removing STL instructions from Favorites (Page 2670)

Overview of the program editor (Page 2523)

Inserting STL instructions using favorites

Requirement

- A block is open.
- Favorites are available.

Procedure

To insert an instruction into a program using Favorites, follow these steps:

1. Drag-and-drop the desired instruction from Favorites to the desired position.

Or:

1. Select the position in the program where you want to insert the instruction.
2. In the Favorites, click on the instruction you want to insert.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Overview of the program editor (Page 2523)
Adding STL elements to Favorites (Page 2669)
Entering STL instructions manually (Page 2663)
Inserting STL instructions using the "Instructions" task card (Page 2665)
Removing STL instructions from Favorites (Page 2670)

Removing STL instructions from Favorites

Requirement

A code block is open.

Procedure

To remove instructions from Favorites, follow these steps:

1. Right-click on the instruction you want to remove.
2. Select the "Remove instruction" command in the shortcut menu.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Adding STL elements to Favorites (Page 2669)

Inserting STL instructions using favorites (Page 2670)

Overview of the program editor (Page 2523)

Inserting block calls in STL

Manually inserting block calls

You can insert calls for functions (FCs) and function blocks (FBs). Function blocks can be called either as a single instance or a multi-instance. If a function block is called as a single instance, it saves its data in a separate data block. If a function block is called as a multi-instance, it stores its data in the instance data block of the calling function block.

Requirement

A network is available.

Inserting a call for a function (FC)

To insert a function call in an STL network, follow these steps:

1. Enter "CALL <Name of the function to be called>".
   Replace "<Name of the function to be called>" with the name of the function that you want to call.
2. Confirm your entry with the Enter key.
   - If the function already exists, all parameters of the function are inserted in the call.
   - If the function does not already exist, you must enter the parameters yourself.

Inserting a call for a function block (FB)

To insert the call for a function block (FB), follow these steps:

1. Enter "CALL <Name of the function block to be called>[, <Name of the instance data block>]".
   Replace "<Name of the function to be called>" with the name of the function block that you want to call. The instance data block entry is optional.
2. Confirm your entry with the Enter key.
   If you have not specified the instance data block or the instance data block you have specified does not exist, the "Call options" dialog is opened. In this case, perform steps 3 and 4.
3. In the dialog, enter whether you want to call the block as a single instance or multi-instance.
   – If you click the "Single instance" button, enter a name for the data block that will be assigned to the call in the entry field "Name".
   – If you click the "Multi instance" button, enter the name of the tag in the "Name in the interface" entry field, with which the called function block will be entered as a static tag in the interface of the calling block.
4. Confirm your entries with "OK".

Result

When you call existing functions or function blocks with an instance data block, the parameters of the function or function block are inserted into the call. You must then assign these parameters.

See also: Parameter transfer at block call (Page 2185)

If when calling a function block you specify an instance data block that does not exist, it will be created.

See also

Single instances (Page 2183)

Multi-instances (Page 2183)

Inserting block calls using a drag-and-drop operation (Page 2672)

Updating block calls in STL (Page 2674)

Changing the instance type (Page 2675)

Parameter assignment via registers during block call in STL (Page 2652)

Inserting block calls using a drag-and-drop operation

You can insert calls for existing functions (FC) and function blocks (FB) using a drag-and-drop operation from the project tree. Function blocks can be called either as a single instance or as a multi-instance. If a function block is called as a single instance, it saves its data in a separate data block. If a function block is called as a multi-instance, it stores its data in the instance data block of the calling function block.

Requirement

- A network is available.
- The function to be called (FC) or the function block (FB) to be called is present.

Inserting a call for a function (FC)

To insert a function call in an STL network using a drag-and-drop operation, follow these steps:

1. Drag the function from the project tree into the network.

Inserting a call for a function block (FB)

To insert the call for a function block (FB), follow these steps:

1. Drag the function block from the project tree into the network.
   The "Call options" dialog opens.
2. In the dialog, enter whether you want to call the block as a single instance or multi-instance.
   - If you click the "Single instance" button, enter a name for the instance data block to be assigned to the call in the entry field "Name".
   - If you click the "Multi-instance" button, in the "Name in the interface" box, enter a name of the tag with which the called function block will be entered as a static tag in the interface of the calling block.
3. Confirm your entries with "OK".

Result

The function or the function block is inserted with its parameters. You must then assign these parameters.

See also: Parameter transfer at block call (Page 2185)

If when calling a function block you specify an instance data block that does not exist, it will be created. If you have called a function block as a multi-instance, this will be entered as a static tag in the interface.

See also

Single instances (Page 2183)

Multi-instances (Page 2183)

Manually inserting block calls (Page 2671)

Updating block calls in STL (Page 2674)

Changing the instance type (Page 2675)

Updating block calls in STL

If interface parameters of a called block are changed, the block call can no longer be executed correctly. You can avoid such inconsistent block calls by updating the block calls.

You have two options to update the block calls:

- Explicit updating in the program editor.
  The block calls in the open block will be updated.
- Implicit updating during compilation.
  All block calls in the program as well as the used PLC data types will be updated.

Update blocks in the program editor

To update a block call within a block, follow these steps:

1. Open the block in the program editor.
2. Click "Update inconsistent block calls" in the toolbar.

Or:

1. Open the block in the program editor.
2. Right-click on the instruction with the block call.
3. Select the "Update" command in the shortcut menu.
   The "Interface update" dialog opens. This dialog shows the differences between the block interface in use and the changed interface of the called block.
4. If you want to update the block call, click "OK". To cancel the update, click "Cancel".

Update block calls during compilation

Follow these steps to update all block calls and uses of PLC data types during compilation implicitly:

1. Open the project tree.
2. Select the "Program blocks" folder.
3. Select the command "Compile > Software (rebuild all blocks)" in the shortcut menu.

See also

Block calls (Page 2180)

Manually inserting block calls (Page 2671)

Inserting block calls using a drag-and-drop operation (Page 2672)

Changing the instance type (Page 2675)

Changing the instance type

Instance type

There are two ways of calling function blocks:

- As a single instance
- As a multiple instance

See also: Call function blocks as single or multi-instances (Page 2182)

You can modify a defined instance type at any time.

Requirement

The user program contains a block call.

Procedure

To change the instance type of a function block, follow these steps:

1. Open the code block and select the block call.
2. Select the "Change instance" command in the shortcut menu.
   The "Call options" dialog opens.
3. Click the "Single instance" or "Multi instance" button.
   - If you select the "Single instance" instance type, enter a name for the data block that is to be assigned to the function block.
   - If you select "Multiple instance" as the instance type, enter in the "Name in the interface" text field the name of the tag with which the called function block is to be entered as a static tag in the interface of the calling block.
4. Confirm your entries with "OK".

Note

The previous single and multiple instances will not be deleted automatically.

See also

Manually inserting block calls (Page 2671)

Inserting block calls using a drag-and-drop operation (Page 2672)

Updating block calls in STL (Page 2674)

Inserting comments

Commenting program code

Various options are available for commenting STL programs:

- Line comment
  A line comment starts with "//" and extends to the end of the line.
- Comment section within a line
  A comment section is started with "(* and completed by "*)". A comment section can be located at any point in the line, but cannot extend over several lines.

Requirement

A network is available.

Inserting line comments

To insert a line comment into an instruction line, follow these steps:

1. Enter "//" at the end of the instruction line.
2. Enter the comment.

Inserting a comment section within a line

Proceed as follows to insert a comment section within an instruction line:

1. Enter a "(*" at any point in the instruction line.
2. Enter the comment after "(*".
3. Enter "*)" after the comment.

Disabling one or more lines with comments

To disable program code with with comments, follow these steps:

1. Select the code lines you want to comment out.
2. Click the "Comment selection" button in the editor.
3. "//" is inserted at the beginning of the line in the selected lines. The code that follows is interpreted as a comment. If lines already containing a line comment are disabled, "//" is inserted as well. If the lines are then enabled again, original comments are retained.

Enabling comment lines

To enable lines that have been commented out to be enabled as code again, proceed as follows:

1. Select the code lines you want to enable.
2. Click the "Remove comment" button in the editor.
3. The "//" mark for line comments at the beginning of the line is removed.

Result

Comments are formatted with highlighting.

See also

Rules for STL (Page 2663)
Entering STL instructions manually (Page 2663)
Inserting STL instructions using the "Instructions" task card (Page 2665)
Inserting operands (Page 2667)
Using favorites in STL (Page 2669)
Inserting block calls in STL (Page 2671)

Editing STL instructions

Selecting STL instructions

You can select individual instructions or all the instructions of a network.

Requirement

STL instructions are available.

Selecting individual STL instructions

To select STL instructions, follow these steps:

1. Set the insertion mark before the first character that you want to select.
2. Press and hold down the left mouse button.
3. Move the cursor to a position after the last character that you want to select.
4. Release the left mouse button.

Selecting all STL instructions of a network

To select all instructions of a network, follow these steps:

1. Click in the network whose instructions you want to select.
2. Select "Select All" from the "Edit" menu or use the keyboard shortcut <Ctrl+A>.

See also

Copying STL instructions (Page 2678)
Cutting STL instructions (Page 2678)
Pasting STL instructions from the Clipboard (Page 2679)
Deleting STL instructions (Page 2680)

Copying STL instructions

Requirement

An STL instruction is available.

Procedure

To copy an STL instruction, follow these steps:

1. Select the STL instruction you want to copy.
2. Select "Copy" in the shortcut menu.

Result

The STL instruction is copied and saved to the clipboard.

See also

Selecting STL instructions (Page 2677)
Cutting STL instructions (Page 2678)
Pasting STL instructions from the Clipboard (Page 2679)
Deleting STL instructions (Page 2680)

Cutting STL instructions

Requirement

An STL instruction is available.

Procedure

To cut an STL instruction, follow these steps:

1. Select the STL instruction you want to cut.
2. Select "Cut" in the shortcut menu.

Result

The STL instruction is cut and saved to the clipboard.

See also

Selecting STL instructions (Page 2677)
Copying STL instructions (Page 2678)
Pasting STL instructions from the Clipboard (Page 2679)
Deleting STL instructions (Page 2680)

Pasting STL instructions from the Clipboard

Requirement

An STL instruction is available.

Procedure

To paste an STL instruction from the Clipboard, follow these steps:

1. Copy an STL instruction or cut an STL instruction.
2. Click the place in the network where you want to insert the STL instruction.
3. Select "Paste" in the shortcut menu.

See also

Selecting STL instructions (Page 2677)
Copying STL instructions (Page 2678)
Cutting STL instructions (Page 2678)
Deleting STL instructions (Page 2680)

Deleting STL instructions

Requirement

An STL instruction is available.

Procedure

To delete an STL instruction, follow these steps:

1. Select the STL instruction you want to delete.
2. Select the "Delete" command in the shortcut menu.

See also

Selecting STL instructions (Page 2677)
Copying STL instructions (Page 2678)
Cutting STL instructions (Page 2678)
Pasting STL instructions from the Clipboard (Page 2679)

Creating SCL programs

Basics of SCL

Programming language SCL

Programming language SCL

SCL (Structured Control Language) is a high-level programming language based on PASCAL. The language is based on DIN EN 61131-3 (international IEC 1131-3).

The standard standardizes programming languages for programmable logic controllers. The SCL programming language fulfills the PLCopen Basis Level of ST language (Structured Text) defined in this standard.

Language elements

SCL also contains higher programming languages in addition to the typical elements of the PLC, such as inputs, outputs, timers or memory bits.

- Expressions
- Value assignments
- Operators

Program control

SCL provides convenient instructions for controlling the program allowing you, for example, to create program branches, loops or jumps.

Application

SCL is therefore particularly suitable for the following areas of application:

- Data management
- Process optimization
- Recipe management
- Mathematical / statistical tasks

Expressions

Description

Expressions are calculated during the runtime of the program and return a value. An expression consists of operands (such as constants, tags or function calls) and optionally out of operators (such as *, /, + or -). Expressions can be linked together or nested within each other by operators.

Evaluation order

The evaluation of the expression occurs in a specific order that is defined by the following factors:

- Priority of the operators involved
- Left-to-right order
- Brackets

Types of expressions

The following expression types are available depending on the operator:

- Arithmetic expressions
  Arithmetic expressions consist of either a numerical value or combine two values or expressions with arithmetic operators.
- Relational expressions
  Relational expressions compare the values of two operands and yield a Boolean value. The result is TRUE if the comparison is true, and FALSE if it is not met.
- Logical expressions
  Logical expressions combine two operands with logical operators (AND, OR, XOR) or negating operands (NOT).

How expressions are used

You can use the result of an expression in different ways:

- As a value assignment for a tag
- As as a condition for a control instruction
- As a parameter for a calling a block or instruction

See also

Operators and operator precedence (Page 2687)

Arithmetic expressions (Page 2682)

Relational expressions (Page 2684)

Logical expressions (Page 2686)

Arithmetic expressions

Description

Arithmetic expressions consist of either a numerical value or combine two values or expressions with arithmetic operators.

Arithmetic operators can process the data types that are allowed in the CPU in use. If two operands are involved in the operation, the data type of the result is determined based on the following criteria:

- If both operands are integers with sign and have different lengths, the result receives the data type of the longer integer (e. g. INT + DINT = DINT).
- If both operands are integers without sign and have different lengths, the result receives the data type of the longer integer (e. g. USINT + UDINT = UDINT).
- If one operand is an integer with sign and the other integer is an operand without sign, the result receives the next larger data type with sign that covers the integer without sign (e. g. SINT + USINT = INT).
  You can only execute an operation with such operands if the IEC check is not set.
- If one operand is an integer and the other operand is a floating-point number, the result receives the data type of the floating-point number (e. g. INT + REAL = REAL).
- If both operands are floating-point numbers and have different lengths, the result receives the data type of the longer floating-point number (e. g. REAL + LREAL = LREAL).
- The data type of the result of an operation that involves operands of the data type groups "Times" and "Date and time" can be found in the table in section "Data types of arithmetic expressions".
  You cannot use data types of the data type groups "Times" and "Date and time" when the IEC check is set.

Data types of arithmetic expressions

The following table shows the data types you can use in arithmetic expressions:

Operation	Operator	1st Operand	2nd Operand	Result
Power	**	Integer/floating-point number	Integer/floating-point number	Integer/floating-point number
Unary plus	+	Integer/floating-point number TIME, LTIME	-	Integer/floating-point number TIME, LTIME
Unary minus	-	Integer/floating-point number TIME, LTIME	-	Integer/floating-point number TIME, LTIME
Multiplication	*	Integer/floating-point number	Integer/floating-point number	Integer/floating-point number
		TIME, LTIME	Integer	TIME, LTIME
Division	/	Integer/floating-point number	Integer/floating-point number (not equal 0)	Integer/floating-point number
		TIME, LTIME	Integer	TIME, LTIME
Modulo function	MOD	Integer	Integer	Integer
Addition	+	Integer/floating-point number	Integer/floating-point number	Integer/floating-point number
		TIME	TIME	TIME
		TIME	DINT	TIME
		LTIME	TIME, LTIME	LTIME
		LTIME	LINT	LTIME
		TOD	TIME	TOD
		TOD	DINT	TOD
		LTOD	TIME, LTIME	LTOD
		LTOD	LINT	LTOD
		DATE	LTOD	DTL
		DATE	TOD	• S7-300/400: DT • S7-1200/1500: DTL
		DT	TIME	DT
		LDT	TIME, LTIME	LDT
		DTL	TIME, LTIME	DTL
Subtraction	-	Integer/floating-point number	Integer/floating-point number	Integer/floating-point number
		TIME	TIME	TIME
		TIME	DINT	TIME
		LTIME [1)]	TIME, LTIME	LTIME
		LTIME	LINT	LTIME
		TOD	TIME	TOD
			DINT	TOD

Operation	Operator	1st Operand	2nd Operand	Result
		LTOD	TIME, LTIME	LTOD
		LTOD	LINT	LTOD
		DATE	DATE	• S7-300/400/1200: TIME • S7-1500: LTIME
		DT	TIME	DT
		LDT	TIME, LTIME	LDT
		DTL	TIME, LTIME	DTL
		DTL	DTL	• S7-1200: TIME • S7-1500: LTIME
[1] Combinations between nanoseconds and milliseconds are not possible within expressions.				

For additional information on valid data types, refer to "See also".

Example

The following example shows an arithmetic expression:

```
SCL
"MyTag1":= "MyTag2" * "MyTag3";
```

See also

Expressions (Page 2681)

Data types (Page 2236)

Operators and operator precedence (Page 2687)

Relational expressions

Description

Relational expressions compare the values of two operands and yield a Boolean value. The result is TRUE if the comparison is true, and FALSE if it is not met.

Relational operators can process the data types that are allowed in the CPU in use. The data type of the result always is BOOL.

Note the following rules when forming relational expressions:

- All tags are comparable within the following data type groups:
  - Integers/floating-point numbers
  - Binary numbers
  - String
- With the following data types/data groups, only tags of the same type can be compared:
  - TIME, LTIME
  - Date and time
- The comparison of strings is based on the ASCII character set. The length of the tags and the numerical value of each ASCII character are used for the comparison.
- S5TIME tags are not permitted as comparison operands. An explicit conversion from S5TIME to TIME or LTIME is necessary.

Data types of relational expressions

The following table shows the data types/data type groups you can use in relational expressions:

Operation	Operator	1st Operand	2nd Operand	Result
Compare for equal, not equal	=, <>	Integer/floating-point number	Integer/floating-point number	BOOL
		Binary number	Binary number	BOOL
		String	String	BOOL
		TIME, LTIME	TIME, LTIME	BOOL
		Date and time	Date and time	BOOL
Compare for less than, less than-equal to, greater than, greater than or equal to	<, <=, >, >=	Integer/floating-point number	Integer/floating-point number	BOOL
		Bit strings (S7-1200/1500 only)	Bit strings (S7-1200/1500 only)	BOOL
		String	String	BOOL
		TIME, LTIME	TIME, LTIME	BOOL
		Date and time	Date and time	BOOL

For additional information on valid data types, refer to "See also".

Example

The following example shows a relational expression:

SCL

```
IF a > b THEN c:= a;
IF A > 20 AND B < 20 THEN C:= TRUE;
IF A<>(B AND C) THEN C:= FALSE;
```

Note

The comparison for STRING and DT are executed internally in the S7-300/400 by extended instructions. The following operands are not permitted for these functions:

- Parameter of a FC
- In-out parameter of an FB of type STRUCT or ARRAY

See also

Expressions (Page 2681)

Data types (Page 2236)

Operators and operator precedence (Page 2687)

Logical expressions

Description

Logical expressions combine two operands with logical operators AND OR XOR or negating operands NOT.

Logical operators can process the data types that are allowed in the CPU in use. The result of a logical expression is of BOOL data type, if both operands are of BOOL data type. If at least one of both operands is a bit string, then the result is also a bit string and is determined by the type of the highest operand. For example, when you link a BYTE type operand to a WORD type operand, the result is type WORD.

To link a BOOL type operand with a bit string, you must first explicitly convert it to a bit string.

Data types of logical expressions

The following table shows the data types you can use in logical expressions:

Operation	Operator	1st Operand	2nd Operand	Result
Negation	NOT	BOOL	-	BOOL
AND logic operation	AND or &	BOOL	BOOL	BOOL
		Bit string	Bit string	Bit string
OR logic operation	OR	BOOL	BOOL	BOOL
		Bit string	Bit string	Bit string
EXCLUSIVE OR logic operation	XOR	BOOL	BOOL	BOOL
		Bit string	Bit string	Bit string

Example

The following example shows a logical expression:

```
SCL
IF "MyTag1" AND NOT "MyTag2" THEN c:=a;
MyTag:=ALPHA OR BETA;
```

See also

Expressions (Page 2681)

Data types (Page 2236)

Operators and operator precedence (Page 2687)

Operators and operator precedence

Operators and their order of evaluation

Expressions can be linked together or nested within each other by operators.

The order of evaluation for expressions depends on the precedence of operators and brackets. The following basic rules apply:

- Arithmetic operators are evaluated before relational operators and relational operators are evaluated before logical operators.
- Operators with no precedence are evaluated according to their occurrence from left to right.
- Operations in brackets are evaluated first.

The following table provides an overview of the operators and their precedence:

Operator	Operation	Precedence
Arithmetic expressions		
+ (Page 2682)	Unary plus	2
- (Page 2682)	Unary minus	2
** (Page 2682)	Power	3
* (Page 2682)	Multiplication	4
/ (Page 2682)	Division	4
MOD (Page 2682)	Modulo function	4
+ (Page 2682)	Addition	5
- (Page 2682)	Subtraction	5
Relational expressions		
< (Page 2684)	Less than	6
> (Page 2684)	Greater than	6
<= (Page 2684)	Less than or equal	6
>= (Page 2684)	Greater than or equal	6
= (Page 2684)	Equal	7

Operator	Operation	Precedence
<> (Page 2684)	Not equal	7
Logical expressions		
NOT (Page 2686)	Negation	3
AND (Page 2686) or & (Page 2686)	Boolean AND	8
XOR (Page 2686)	Exclusive OR	9
OR (Page 2686)	Boolean OR	10
Miscellaneous operations		
( ) (Page 2681)	Brackets	1
:= (Page 2688)	Assignment	11

Value assignments

Definition

You can use a value assignment to assign the value of an expression to a tag. On the left side of the assignment is the tag that takes the value of the expression on the right.

The name of a function can also be specified as an expression. The function is called by the value assignment and returns its function value to the tag on the left.

The data type of value assignment is defined by the data type of the tag on the left. The data type of the expression on the right must match this type.

For additional information on compatibility and conversion of data types, refer to "See also".

Value assignments for STRUCT data type or PLC data types

An entire structure can be assigned to another if the structures are identically organized and the data types as well as the names of the structural components match.

You can assign a tag, an expression or another structural element to an individual structural element.

Value assignments for the ARRAY data type

An entire ARRAY can be assigned to another ARRAY if both the data types of the ARRAY elements as well as the ARRAY limits match.

You can assign a tag, an expression or another ARRAY element to an individual ARRAY element.

Value assignments for the STRING data type

An entire STRING can be assigned to another STRING.

You can assign another STRING element to an individual STRING element.

Value assignments for the ANY data type

You can assign a tag with the ANY data type only to the following objects:

- Input parameters or temporary local data of FBs that also have the data type ANY.
- Temporary local data of FCs that also have the data type ANY.

Note that you can only point to memory areas with "standard" access mode with the ANY pointer.

Value assignments for the POINTER data type

You cannot use POINTER in value assignments in SCL.

Examples

The following table shows examples for value assignments:

SCL	
`"MyTag1" := "MyTag2";`	`(* Assignment of a tag*)`
`"MyTag1" := "MyTag2" * "MyTag3";`	`(* Assignment of an expression*)`
`"MyTag" := "MyFC"();`	`(* Call for a function that assigns its function value to the "MyTag" tag*)`
`#MyStruct.MyStructElement := "MyTag";`	`(* Assignment of a tag to a structure element*)`
`#MyArray[2] := "MyTag";`	`(* Assignment of a tag to an ARRAY element*)`
`"MyTag" := #MyArray[1,4];`	`(* Assignment of an ARRAY element to a tag*)`
`#MyString[2] := #MyOtherString[5];`	`(* Assignment of a STRING element to another STRING element*)`

See also

Addressing operands (Page 2209)

Data types (Page 2236)

Operators and operator precedence (Page 2687)

Settings for SCL

Overview of the settings for SCL

Overview

The following tables show the settings you can make for SCL:

Editor settings

Group	Setting	Description
View	Keyword highlighting	Notation used to represent the keywords of the programming language. You can choose between uppercase and lowercase letters or a notation corresponding to the conventions of the Pascal programming language.

Default settings for new blocks

If you create new blocks, the following settings are set as default values. You can change these in the block properties at a later point in time.

Group	Setting	Description
Compile	Create extended status information	Allows all tags in a block to be monitored. The memory requirements of the program and execution times increase, however, with this option.
	Check ARRAY limits[1)]	Checks at runtime whether array indices are within the declared range for an ARRAY. If an array index exceeds the permissible range, the enable output ENO of the block is set to "0".
	Set ENO automatically	Checks at runtime whether errors occur in the processing of certain instructions. If a runtime error occurs, the enable output ENO of the block is set to "0".
[1)]For CPUs of the S7-300/400 series: When the ARRAY limits are violated, the enable output ENO is set to FALSE. For CPUs of the S7-1200/1500 series: When the ARRAY limits are violated, the enable output ENO is not set to FALSE. See "Addressing structured variables (Page 2214)" for error query options.		

See also

EN/ENO mechanism (Page 2447)

Changing the settings (Page 2690)

Changing the settings

Procedure

To change the settings, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. Change the settings.

Result

The change will be loaded directly, there is no need to save it explicitly.

See also

Overview of the settings for SCL (Page 2689)

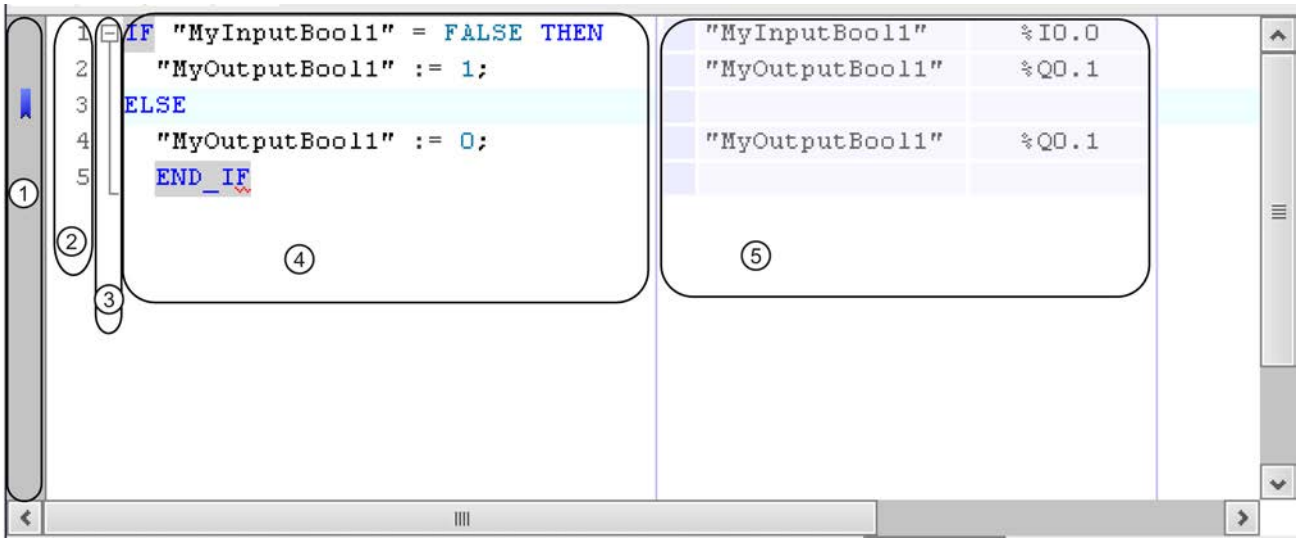
The programming window of SCL

Overview of the programming window

Function

The programming window is the work area, where you enter the SCL program.

The following figure shows the programming window of SCL:



The programming window consists of the following sections:

Section	Meaning
① Sidebar	You can set bookmarks and breakpoints in the sidebar.
② Line numbers	The line numbers are displayed to the left of the program code.
③ Outline view	The outline view highlights related code sections.
④ Code area	You edit the SCL program in the code area.
⑤ Display of the absolute operands	This table shows the assignment of symbolic operands to absolute addresses.

See also

Using bookmarks (Page 2695)

Customizing the programming window (Page 2692)

Formatting SCL code (Page 2693)

Expanding and collapsing sections of code (Page 2694)

Customizing the programming window

Introduction

You can customize the appearance of the programming window and the program code in the following way:

- By setting the font, size and color
- By setting the tab spacing
- By displaying the line numbers
- By showing or hiding the absolute operands

Setting the font, size and color

To set the font, size and color, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "General > Script/text editors" group.
3. Select the desired font and font size or choose a font color for the individual language elements.

Setting the tab spacing

To provide a better overview of the program, lines are indented according to syntax. Define the depth of indentation with the tab spacing.

To set the tab spacing, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "General > Script/text editors" group.
3. Set the tab spacing.

Show line numbers

To display the line numbers, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "General > Script/text editors" group.
3. Select the "Show line numbers" option.

Show or hide the absolute operands

You can show the assignment of symbolic and absolute operands in a table next to the program code, if required.

To hide or show the display of the absolute operands, follow these steps:

1. Click the "Absolute/symbolic operands" icon in the toolbar.
   The display of the absolute operands appears.
2. To move the display, click the table and drag it to the desired position while holding down the mouse button.
3. To change the width of the table, click on the right or left table border and drag it to the right or left while holding down the mouse button.

See also

Using bookmarks (Page 2695)

Overview of the programming window (Page 2691)

Formatting SCL code (Page 2693)

Expanding and collapsing sections of code (Page 2694)

Formatting SCL code

Introduction

To make the program clearer, you can indent or outdent individual lines manually or format code sections. Note the following information about formatting code sections:

- The type of formatting is based on the general settings for indents, but at least the line or the section is always indented. If you selected the setting "Smart", unnecessary spaces within the SCL instruction are also removed.
- Only syntactically correct code sections can be formatted.
- If you place the insertion point in the first or last line of an instruction for program control, for example in an IF instruction in the line with the "IF", the entire instruction is formatted.
- If you select text, only the selected text is formatted.

Indenting or outdenting lines

To indent or outdent individual lines, follow these steps:

1. Click on the line you want to indent or outdent.
2. Press the "Indent text", "Outdent text" button into the toolbar of the programming editor.

Note

You can set the width of the indent in "Options > Settings".

Formatting code sections

To format code sections, follow these steps:

1. Select the text that you want to format or place the insertion point in the appropriate line.
2. Select the "Format selected text automatically" button into the toolbar of the programming editor.

See also

Using bookmarks (Page 2695)

Overview of the programming window (Page 2691)

Customizing the programming window (Page 2692)

Expanding and collapsing sections of code (Page 2694)

Overview of the script and text editor settings (Page 275)

Expanding and collapsing sections of code

Introduction

SCL instructions can span several lines. Examples for this are program control instructions or block calls.

These instructions are identified as follows:

- An outline view between the display line number and the program code marks the entire code section.
- When you select the opening keyword, the closing keyword is automatically highlighted.

Procedure

To expand or collapse the code section, follow these steps:

1. Click the minus sign in the outline view.
   The code section closes.
2. Click the plus sign in the outline view.
   The code section opens.

See also

Using bookmarks (Page 2695)

Overview of the programming window (Page 2691)

Customizing the programming window (Page 2692)

Formatting SCL code (Page 2693)

Using bookmarks

Basics of bookmarks

Function

You can use bookmarks to mark program locations in extensive programs so that you can find them quickly later if they need revising. Bookmarks are displayed in the sidebar of the programming window. You can navigate between multiple bookmarks within a block using menu commands.

Bookmarks are saved with the project and are therefore available for anyone who wants to edit the block. However, they are not loaded to a device.

Bookmarks are not evaluated when blocks are compared.

See also

Setting bookmarks (Page 2695)

Navigating between bookmarks (Page 2696)

Deleting bookmarks (Page 2696)

Setting bookmarks

Requirement

The SCL block is open.

Procedure

To set a bookmark, follow these steps:

1. Right-click on the desired line in the sidebar.
2. Select the "Bookmarks > Set" command in the shortcut menu.

Or:

1. Click on the line in which you want to place the bookmark.
2. Click the "Set/delete bookmark" button in the toolbar.

Or:

1. Hold down the <Ctrl> key.
2. Click on the line in the sidebar in which you want to place the bookmark.

Result

A bookmark is placed in the program code.

See also

Basics of bookmarks (Page 2695)
Navigating between bookmarks (Page 2696)
Deleting bookmarks (Page 2696)

Navigating between bookmarks

Requirement

Several bookmarks are set in a block.

Procedure

To navigate between bookmarks, follow these steps:

1. Set the insertion cursor in the program code.
2. In the "Edit" menu, select the "Go to > Next bookmark" or "Go to > Previous bookmark" command.

Or:

1. Set the insertion cursor in the program code.
2. In the toolbar of the programming editor, click the "Go to next bookmark", "Go to previous bookmark" button.

Or:

1. Click in the sidebar.
2. Select the "Bookmarks > Next" or "Bookmarks > Previous" command in the shortcut menu.

Result

The line with the bookmark is highlighted.

See also

Basics of bookmarks (Page 2695)
Setting bookmarks (Page 2695)
Deleting bookmarks (Page 2696)

Deleting bookmarks

You can delete individual bookmarks or all bookmarks from the block or the CPU.

Deleting individual bookmarks

To delete an individual bookmark, follow these steps:

1. Right-click in the sidebar on the line in which you want to delete the bookmark.
2. Select the "Bookmarks > Remove" command in the shortcut menu.

Or:

1. Click on the line in which you want to delete the bookmark.
2. In the "Edit" menu, select the "Bookmarks > Remove" command.

Or:

1. Click on the line in which you want to delete the bookmark.
2. Click the "Set/delete bookmark" button in the toolbar.

Deleting all bookmarks from the block

To delete all bookmarks from the block, follow these steps:

1. Right-click in the sidebar.
2. Select the "Bookmarks > Delete all from block" command in the shortcut menu.

Or:

1. In the "Edit" menu, select the "Bookmarks > Delete all from block" command.

See also

Basics of bookmarks (Page 2695)
Setting bookmarks (Page 2695)
Navigating between bookmarks (Page 2696)

Entering SCL instructions

Rules for SCL instructions

Instructions in SCL

SCL recognizes the following types of instructions:

- Value assignments
  Value assignments are used to assign a tag a constant value, the result of an expression or the value of another tag.
- Instructions for program control
  Instructions for program control are used to implement program branches, loops or jumps.

- Additional instructions from the "Instructions" task card
  The "Instructions" task card offers a wide selection of standard instructions that you can use in your SCL program.
- Block calls
  Block calls are used to call up subroutines that have been placed in other blocks and to further process their results.

Rules

You need to observe the following rules when entering SCL instructions:

- Instructions can span several lines.
- Each instruction ends with a semicolon (;).
- No distinction is made between upper and lower case.
- Comments serve only for documentation of the program. They do not affect the program execution.

Examples

The following examples shows the various types of instructions:

SCL

```
// Example of a value assignment
"MyTag":= 0;
// Example of a block call
"MyDB"."MyFB" (ParamInput:= 10);
// Example of a program control instruction
WHILE "Counter" < 10 DO
    "MyTAG" := "MyTag" + 2;
END_WHILE;
```

See also

Basics of SCL (Page 2680)

Entering SCL instructions manually

Requirement

An SCL block is open.

Procedure

To enter SCL instructions, follow these steps:

1. Enter the syntax of the instruction using the keyboard.
   You are supported by the auto-complete function when performing this task. It offers all the instructions and operands that are allowed at the current location.
2. Select the required instruction or the desired operand from the auto-complete function.
   If you select an instruction that requires specification of operands, placeholders for the operands are inserted into the program. The placeholders for the operands are highlighted in yellow. The first placeholder is selected.
3. Replace this placeholder with an operand.
4. Use the <TAB> key to navigate to all other placeholders and replace them with operands.

Note

You can also drag-and-drop a defined operand from the PLC tag table or from the block interface into the program. To replace an operand that has already been inserted, hover the mouse pointer briefly over the operand to be replaced before releasing the mouse button. This selects the operand and when you release the mouse button it is replaced by the new operand.

Result

The instruction is inserted.

The programming editor performs a syntax check. Incorrect entries are displayed in red and italics. In addition, you also receive a detailed error message in the inspector window.

See also

Using autocompletion in textual programming languages (Page 2541)

Data type conversion (Page 2286)

Expanding and reducing the parameter list (Page 2714)

Inserting SCL instructions using the "Instructions" task card

The "Instructions" task card offers a wide selection of instructions that you can use in your SCL program. The SCL-specific instructions for program control are available in the "Instructions" task card.

Requirement

An SCL block is open.

Procedure

To insert SCL instructions into a program using the "Instructions" task card, follow these steps:

1. Open the "Instructions" task card.
2. To insert the instruction, select one of the following steps:
   - Navigate to the SCL instruction you want to insert and drag-and-drop it to the required line in the program code. The insertion location is highlighted by a green rectangle.
   - Select the location in the program code where you want to insert the instruction and then double-click on the instruction you want to insert.

   The instruction is inserted in the program. The placeholders for the operands are highlighted in yellow. The first placeholder is selected.
3. Replace this placeholder with an operand. You can also drag a tag from the interface or the PLC tag table with drag-and-drop to the placeholder.
4. Use the <TAB> key to navigate to all other placeholders and replace them with operands.

Result

The instruction is inserted.

The programming editor performs a syntax check. Incorrect entries are displayed in red and italics. In addition, you also receive a detailed error message in the inspector window.

See also

Using autocompletion in textual programming languages (Page 2541)

Data type conversion (Page 2286)

Expanding and reducing the parameter list (Page 2714)

Defining the data type of an SCL instruction

Basic information on the data types of SCL instructions

Introduction

The SCL instructions that you employ for block programming use specific data types to calculate function values. Certain SCL instructions only support the use of a specific data type. You cannot change the data type for these instructions. However, most of the SCL instructions support the use of different data types. We differentiate between the following two types of such instructions:

- Instructions for which the data type of the function value is determined by the data type of the input parameters. This is the case for most instructions.
- Instructions with default data type. The instructions listed in the following table are of this type.

You will have to change the default data type if this is incompatible with the data type of the input parameter used. You can always change the data type based on the following syntax:
`_<data type>`

SCL instructions with default data type

The following table lists the SCL instructions with default data types:

Instruction	Default data type
CEIL	DINT
DECO	DWORD
FLOOR	DINT
NORM_X	REAL
PEEK	BYTE
SCALE_X	INT
TRUNC	DINT
CONCAT	STRING

See also

Changing the data type of an SCL instruction (Page 2701)

Example for changing the data type of an SCL instruction (Page 2702)

Changing the data type of an SCL instruction

Procedure

Proceed as follows to insert an SCL instruction and change its data type:

1. Insert the instruction at the required point in the program using drag-and-drop.
2. Specify the operands for the instruction.
   The data type of the function value is specified based on the input parameters, or the default data type of the instruction is used.
3. Append the "_<data type>" string to the instruction name.
   "<data type>" represents the data type you need for the instruction.

See also

Basic information on the data types of SCL instructions (Page 2700)

Example for changing the data type of an SCL instruction (Page 2702)

Modifying the data types of IEC timers and IEC counters

IEC timers and IEC counters are internal system function blocks and require an instance data block. You can create the instance data blocks either as single or multi-instance. The data type of the instance data block is determined according to the associated instruction. For CPUs of the S7-1200 and S7-1500 series, you can, however execute the instructions with different data types, depending on your requirements.

If the newly set data type of the instance data block does not match the data type of the input parameter, an implicit conversion takes place if possible. If the conversion is not possible, you will receive an error message.

Procedure

To change the data type of an IEC timer or IEC-counter instance data block, proceed as follows:

1. Open the block in which you call the IEC timer or IEC counter.
   Depending on the instance type of the instance data block, there is a green-bordered box before (multi-instance) or after (single instance) the name of the instance data block.
2. Click the green-bordered box.
   A drop-down list box with the valid data types for the instance data block is opened.
3. Select the desired data type.

Example for changing the data type of an SCL instruction

Changing the default data type of the "Decode" instruction (DECO)

Data type DWORD is set as default if you insert the "Decode" instruction in the program.

```
"Tag_Result" := DECO(IN := "Tag_Value");
```

Modify the program code as follows to convert the data type from DWORD to BYTE:

```
"Tag_Result_BYTE" := DECO_BYTE(IN := "Tag_Value");
```

See also

Basic information on the data types of SCL instructions (Page 2700)

Changing the data type of an SCL instruction (Page 2701)

Displaying or hiding tag information

Introduction

Regardless of whether the operands are represented in absolute or symbolic form, you can show and hide simple or hierarchical comments used to document global tags. This information is taken from the PLC tag table.

You can display the tag information either for all the blocks or for individually opened blocks. If you display the tag information for all the blocks, the tag information for all blocks currently opened and opened in future is shown.

You can hide the tag information at any time again. If you have hidden the tag information for all blocks, you can display it again for individual ones that you have opened.

If you select the display of tag information with hierarchical comments, the comments of the higher structure levels of structured tags will also be displayed. The display is in brackets after the comment of the tags; the comments of the individual levels are separated by a period. If there is no comment at a structure level for a tag, it is omitted in the display and this is recognizable because there are two periods.

Displaying or hiding tag information for all blocks

Follow the steps below to display or hide the tag information for all blocks:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. If you want to show the tag information, either select the "Expand" option in the "Tag information" drop-down list or the "Tag information with hierarchy" depending on whether you want to display simple or hierarchical comments.
4. If you want to hide the tag information, select the "Collapse" option in the "Tag information" drop-down list.
   The tag information is displayed or hidden for all blocks. When you open further blocks, the tag information is displayed or hidden depending on the selected setting.

Displaying or hiding tag information for an opened block

Follow the steps below to display or hide the tag information for an opened block:

1. If you want to show the tag information, either select the "Show tag information" option in the "Shows the tag information" drop-down list or the "Tag information with hierarchy" depending on whether you want to display simple or hierarchical comments.
2. If you want to hide the tag information, select the "Hide tag information" option in the "Hides tag information" drop-down list.
   The tag information is displayed or hidden.

Using Favorites in SCL

Adding SCL instructions to the Favorites

Requirement

- A block is open.
- The multipane mode is set for the "Instructions" task card or the Favorites are also displayed in the editor.

Procedure

To add SCL instructions to the Favorites, follow these steps:

1. Open the "Instructions" task card.
2. Maximize the "Basic instructions" pane.
3. Navigate in the "Basic instructions" pane to the instruction that you want to add to the Favorites.
4. Drag-and-drop the instruction into the "Favorites" pane or into the Favorites area in the program editor.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Overview of the program editor (Page 2523)

Inserting SCL instructions using Favorites (Page 2704)

Removing SCL instructions from the Favorites (Page 2705)

Inserting SCL instructions using Favorites

Requirement

- A block is open.
- Favorites are available.

Procedure

To insert an instruction into a program using Favorites, follow these steps:

1. Drag-and-drop the desired instruction from Favorites to the desired position.

Or:

1. Select the position in the program where you want to insert the instruction.
2. In the Favorites, click on the instruction you want to insert.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Overview of the program editor (Page 2523)
Adding SCL instructions to the Favorites (Page 2703)
Removing SCL instructions from the Favorites (Page 2705)

Removing SCL instructions from the Favorites

Requirement

A code block is open.

Procedure

To remove instructions from Favorites, follow these steps:

1. Right-click on the instruction you want to remove.
2. Select the "Remove instruction" command in the shortcut menu.

Note

To additionally display the Favorites in the program editor, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Overview of the program editor (Page 2523)
Adding SCL instructions to the Favorites (Page 2703)
Inserting SCL instructions using Favorites (Page 2704)

Insert block calls in SCL

Basic information on the block call in SCL

Calling function blocks

Syntax of a call

The following syntax is used to call a function block as a single or multi-instance:

- Single instance:
  - If the function block originates from the project:
    `<DBName> (Parameter list)`
  - If the function block originates from the "Instructions" task card:
    `<DB name>.<Instruction name> (Parameter list)`
- Multi-instance
  `<#Instance name> (Parameter list)`

Calling as single instance or multi-instance

Function blocks can be called either as a single instance or a multi-instance.

- Calling as a single instance
  The called function block stores its data in a data block of its own.
- Calling as a multi-instance
  The called function block stores its data in the instance data block of the calling function block.

For additional information on the types of calls, refer to "See also".

Parameter list

If you call another code block from a SCL block, you can supply the formal parameters of the called block with actual parameters.

The specification of the parameters has the form of a value assignment. This value assignment enables you to assign values (actual parameters) to the parameters you have defined in the called block.

The formal parameters of the called code block are listed in brackets directly after the call. Input and in-out parameters have the assignment identifier ":=", output parameters have the assignment identifier "=>". A placeholder placed after the parameter shows the required data type and the type of the parameter.

Rules for supplying parameters

The following rules apply to supplying parameters:

- Constants, tags and expressions can be used as actual parameters.
- The assignment order is not of importance.
- The data types of formal and actual parameters must match.
- The individual assignments are separated by commas.
- If the called block has only one parameter, it is sufficient to specify the actual parameter in the brackets. The formal parameter need not be specified.

See also

Manually inserting block calls (Page 2711)

Inserting block calls with drag-and-drop (Page 2712)

Examples for calling a function block in SCL (Page 2709)

Block calls (Page 2180)

Using and addressing operands (Page 2197)

Calling functions

Syntax of a call

The following syntax is used to call a function:

```
<Function name> (Parameter list);     //Standard call
<Operand>:=<Function name> (Parameter list); // Call in an expression
```

Function value

Functions that provide a return value can be used in any expression in place of an operand. For this reason, the return value is also known as the "function value" in SCL.

The call options of functions depend on whether the function returns a function value to the calling block.

The function value is defined in the RET_VAL parameter. If the RET_VAL parameter is of the VOID data type, then the function will not return a value to the calling block. If the RET_VAL parameter has another data type, then the function returns a function value of this data type.

In SCL, all data types are permitted for the RET_VAL parameter except ANY, ARRAY, STRUCT and VARIANT, as well as the parameter types TIMER and COUNTER.

Call options

There are two possibilities for calling functions in SCL:

- Standard call for functions with and without a function value
  With a standard call, the results of the function is made available as an output and in-out parameter.
- Call in an expression for functions with a function value
  Functions that return a function value can be used in any expression in place of an operand, for example, a value assignment.
  The function calculates the function value, which has the same name as the function and returns it to the calling block. There the value replaces the function call.
  Following the call, the results of the function is made available as a function value or as an output and in-out parameter.

Parameter list

If you call another code block from a SCL block, you need to supply the formal parameters of the called block with actual parameters.

The specification of the parameters has the form of a value assignment. This value assignment enables you to assign values (actual parameters) to the parameters you have defined in the called block.

The formal parameters of the called code block are listed in brackets directly after the call. Input and in-out parameters have the assignment identifier ":=", output parameters have the assignment identifier "=>". A gray placeholder placed after the parameter shows the required data type and the type of the parameter.

Rules for supplying parameters

The following rules apply to supplying parameters to functions:

- All parameters of the function must be supplied.
- The assignment order is not of importance.
- Constants, tags and expressions can be used as actual parameters.
- The data types of formal and actual parameters must match.
- The individual assignments are separated by commas.
- If the called block has only one parameter, it is sufficient to specify the actual parameter in the brackets. The formal parameter need not be specified.
- When you call functions in SCL, you cannot use the release mechanism via EN. Use an IF statement instead to call functions conditionally.

See also

Manually inserting block calls (Page 2711)

Inserting block calls with drag-and-drop (Page 2712)

Examples for calling functions in SCL (Page 2710)

Block calls (Page 2180)

Using and addressing operands (Page 2197)

Examples for calling a function block in SCL

Calling as a single instance

The following example shows the call of an FB as a single instance:

SCL

```
// Call as a single instance
"MyDB" (MyInput:=10, MyInout:= "Tag1");
```

Result

After the call is executed, the value determined for the "MyInout" in/out parameter is available in "Tag1" in the "MyDB" data block.

Calling as a multi-instance

The following example shows the call of an FB as a multi-instance:

SCL

```
// Call as a multi-instance
"MyFB" (MyInput:= 10, MyInout:= "Tag1");
```

Result

After the "MyFB" block is executed, the value determined for the "MyInout" in-out parameter is made available in "Tag1" in the data block of the calling code block.

See also

Calling function blocks (Page 2706)

Block calls (Page 2180)

Using and addressing operands (Page 2197)

Manually inserting block calls (Page 2711)

Inserting block calls with drag-and-drop (Page 2712)

Examples for calling functions in SCL

Standard call

The following example shows a standard function call:

```
SCL
// Standard function call
"MyFC" (MyInput := 10, MyInOut := "Tag1");
```

Result

After the "MyFC" block is executed, the value determined for the "MyInOut" in/out parameter is available in "Tag1" in the calling block and needs to be further processed there.

Call in a value assignment

The following example shows a function call in a value assignment:

```
SCL
(*Call in a value assignment, a function value was defined for "MyFC" *)
#MyOperand := "MyFC" (MyInput1 := 3, MyInput2 := 2, MyInput3 := 8.9,
MyInOut := "Tag1");
```

Result

The function value of "MyFC" is transferred to "#MyOperand".

Call in an arithmetic expression

The following example shows a function call in an arithmetic expression:

```
SCL
(*Call in a mathematical expression, a function value was defined for
"MyFC"  *)
#MyOperand := "Tag2" + "MyFC" (MyInput1 := 3, MyInput2 := 2, MyInput3 :=
8.9);
```

Result

The function value of "MyFC" will be added to "Tag2" and the result will be transferred to "MyOperand".

See also

Calling functions (Page 2707)

Manually inserting block calls (Page 2711)

Inserting block calls with drag-and-drop (Page 2712)

Manually inserting block calls

You can insert calls for functions (FCs) and function blocks (FBs).

Inserting a call for a function (FC)

Proceed as follows to insert a function call:

1. Enter the function name.
2. Confirm your entry with the Return key.
   The syntax for the function call including the parameter list is added to the SCL program. The placeholders for the actual parameters are highlighted in yellow. The first placeholder is selected.
3. Replace this placeholder with an actual parameter. You can also drag a tag from the interface or the PLC tag table with drag-and-drop to the placeholder.
4. Use the <TAB> key to navigate to all other placeholders and replace them with actual parameters.

Inserting a call for a function block (FB)

To insert a call for a function block (FB), follow these steps:

1. Enter the name of the function block.
2. Confirm your entry with the Return key.
   The "Call options" dialog opens.
3. In the dialog, specify whether you want to call the block as a single or multi-instance.
   - If you click the "Single instance" button, in the "Name" field enter a name for the data block to be assigned to the call.
   - If you click the "Multi-instance" button, in the "Name in the interface" field enter a name of the tag with which the called function block is to be entered as a static tag in the interface of the calling block.
4. Confirm your entries with "OK".
   The syntax for the function block call including the parameter list is added to the SCL program. The placeholders for the actual parameters are highlighted in yellow. The first placeholder is selected.
5. Replace this placeholder with an actual parameter. You can also drag a tag from the interface or the PLC tag table with drag-and-drop to the placeholder.
6. Use the <TAB> key to navigate to all other placeholders and replace them with actual parameters.

Result

The block call is inserted.

If you specify an instance data block that does not exist when calling a function block, it is created.

See also

Updating block calls (Page 2713)

Expanding and reducing the parameter list (Page 2714)

Basic information on the block call in SCL (Page 2706)

Block calls (Page 2180)

Using autocompletion in textual programming languages (Page 2541)

Inserting block calls with drag-and-drop

You can insert calls for existing functions (FC) and function blocks (FB) using a drag-and-drop operation from the project tree.

Requirement

The function to be called (FC) or the function block (FB) to be called is present.

Inserting a call for a function (FC)

To insert a function call using drag-and-drop, follow these steps:

1. Drag the function from the project tree into the program.
   The syntax for the function call including the parameter list is added to the SCL program. The placeholders for the actual parameters are highlighted in yellow. The first placeholder is selected.
2. Replace this placeholder with an actual parameter. You can also drag a tag from the interface or the PLC tag table with drag-and-drop to the placeholder.
3. Use the <TAB> key to navigate to all other placeholders and replace them with actual parameters.

Inserting a call for a function block (FB)

To insert a call for a function block (FB) using drag-and-drop, follow these steps:

1. Drag the function block from the project tree and drop it into the program.
   The "Call options" dialog opens.
2. In the dialog, specify whether you want to call the block as a single or multi-instance.
   - If you click the "Single instance" button, in the "Name" field enter a name for the data block to be assigned to the call.
   - If you click the "Multi-instance" button, in the "Name in the interface" field enter a name of the tag with which the called function block is to be entered as a static tag in the interface of the calling block.
3. Confirm your entries with "OK".
   The syntax for the function block call including the parameter list is added to the SCL program. The placeholders for the actual parameters are highlighted in yellow. The first placeholder is selected.
4. Replace this placeholder with an actual parameter. You can also drag a tag from the interface or the PLC tag table with drag-and-drop to the placeholder.
5. Use the <TAB> key to navigate to all other placeholders and replace them with actual parameters.

Result

The block call is inserted.

If you specify an instance data block that does not exist when calling a function block, it is created.

See also

Updating block calls (Page 2713)

Expanding and reducing the parameter list (Page 2714)

Basic information on the block call in SCL (Page 2706)

Block calls (Page 2180)

Using autocompletion in textual programming languages (Page 2541)

Updating block calls

If interface parameters of a called block are changed, the block call can no longer be executed correctly. You can avoid such inconsistent block calls by updating the block calls.

You have two options to update the block calls:

- Explicit updating in the program editor.
  The inconsistencies within the open block are displayed and can be updated.
- Implicit updating during compilation.
  All block calls in the program as well as the used PLC data types will be updated.

Update blocks in the program editor

To update a block call within a block, follow these steps:

1. Open the block in the program editor.
2. Click "Update inconsistent block calls" in the toolbar.
   Inconsistent calls are displayed.
3. Correct the inconsistencies.

Update block calls during compilation

Proceed as follows to update all block calls and uses of PLC data types during compilation implicitly:

1. Open the project tree.
2. Select the "Program blocks" folder.
3. Select the command "Compile > Software (rebulid all blocks)" in the shortcut menu.

See also

Manually inserting block calls (Page 2711)

Inserting block calls with drag-and-drop (Page 2712)

Expanding and reducing the parameter list

In SCL, if you call blocks or insert instructions that are system-internal function blocks, the syntax and the parameter list with the placeholders for the actual parameters are inserted in the SCL program. To make the program code easier to read, the unused optional parameters are removed from the parameter list when you edit other instructions. You can restore these at any time. You can also explicitly reduce the parameter list when you have finished assigning the parameters.

Expanding the parameter list

To expand the parameter list, follow these steps:

1. Right-click in the block call or the instruction.
2. Select the "Expand parameter list" command from the shortcut menu or press the key combination <Ctrl+Shift+Space bar>.
   The parameter list is displayed in full again.

Reducing the parameter list

To reduce the parameter list, follow these steps:

1. Right-click in the block call or the instruction.
2. Select the "Reduce parameter list" command from the shortcut menu or press the key combination <Ctrl+Shift+Space bar>.
   All unused optional parameters are hidden.

See also

Entering SCL instructions manually (Page 2698)

Inserting SCL instructions using the "Instructions" task card (Page 2699)

Manually inserting block calls (Page 2711)

Inserting block calls with drag-and-drop (Page 2712)

Inserting comments

Commenting program code

You have various options for commenting SCL programs:

- Line comment
  A line comment starts with "//" and extends to the end of the line.
- Comment section
  A comment section is started with "(* and completed by "*)". It can span several lines.

Inserting a line comment

To insert line comments, follow these steps:

1. Type "//" at the position where you want to place the comment. This does not have to be the beginning of the line.
2. Enter the comment text.

Inserting a comment section

To insert a comment section, follow these steps:

1. Type "(*" at the position where you want to place the comment. This does not have to be the beginning of the line.
2. Enter the comment text.
3. Complete the comment with "*)".

Disabling one or more lines with comments

To disable program code with comments, follow these steps:

1. Select the code lines you want to comment out.
2. Click the "Comment selection" button in the editor.
   "//" is inserted at the beginning of the line in the selected lines. The code that follows is interpreted as a comment. If lines already containing a line comment are disabled, "//" is inserted as well. If these lines are enabled again, the original comments are retained.

Enabling comment lines

To enable lines that have been commented out to be enabled as code again, proceed as follows:

1. Select the code lines you want to enable.
2. Click the "Remove comment" button in the editor.
   The "//" mark for line comments at the beginning of the line is removed.

Example

The following code contains comment sections and line comments

```
(*****************************************************************************
  A description of the instructions that follow can be placed here
*****************************************************************************)
IF "MyVal1" > 0 THEN //No division by 0
    "MyReal" := "MyVal2" (* input value *) / "MyVal1" (* measured value *);
END_IF;
//Data type conversion
"MyInt" := REAL_TO_INT("MyReal");
```

Editing SCL instructions

Selecting instructions

You can select individual instructions or all instructions of a block.

Requirement

An SCL block is open.

Selecting individual instructions

To select individual instructions, follow these steps:

1. Set the insertion mark before the first character that you want to select.
2. Press and hold down the left mouse button.
3. Move the cursor to a position after the last character that you want to select.
4. Release the left mouse button.

Selecting all the instructions of a program

To select all instructions, follow these steps:

1. In the "Edit" menu, select the "Select All" command or use the keyboard shortcut <Ctrl+A>.

Note

When you select the opening keyword of an instructing, the closing keyword is automatically highlighted.

Copying, cutting and pasting instructions

Copying an instruction

To copy an instruction, follow these steps:

1. Select the instruction you want to copy.
2. Select "Copy" in the shortcut menu.

Cutting an instruction

To cut an instruction, follow these steps:

1. Select the instruction you want to cut.
2. Select the "Cut" command in the shortcut menu.

Inserting an instruction from the clipboard

To insert an instruction from the clipboard, follow these steps:

1. Copy or cut an instruction.
2. Click on the position at which you want to insert the instruction.
3. Select "Paste" in the shortcut menu.

Deleting instructions

Requirement

An SCL block is open.

Procedure

To delete an instruction, follow these steps:

1. Select the instruction you want to delete.
2. Select the "Delete" command in the shortcut menu.

Creating GRAPH programs

Basic information on GRAPH

GRAPH programming language

Overview

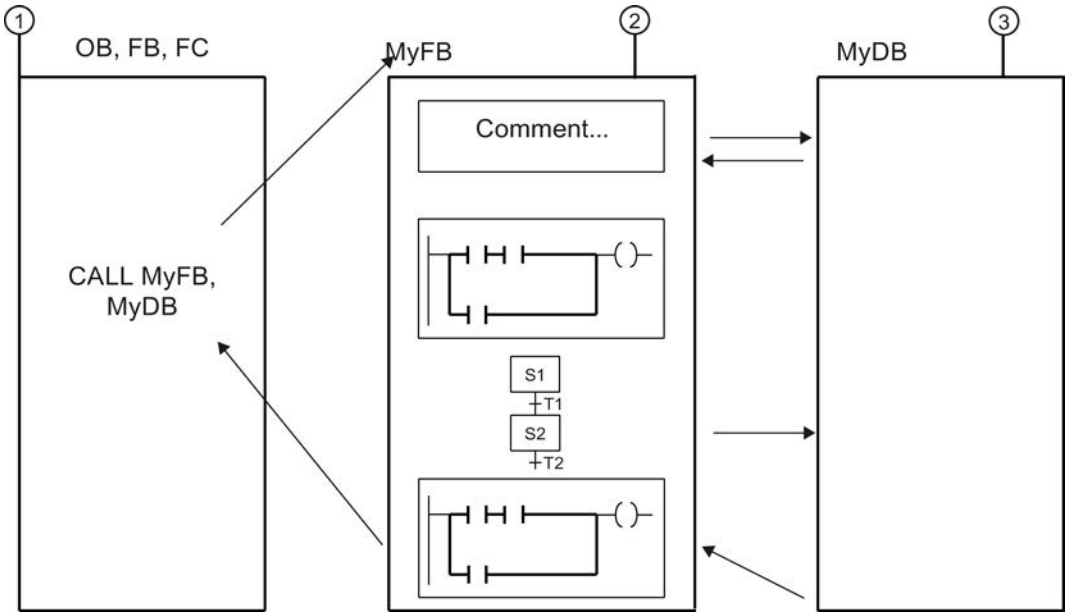
GRAPH is a graphic programming language for creating sequential control systems. Sequences can be programmed in a fast straightforward manner using sequencers. The process is broken down into individual steps, each with a clear scope of functions, and organized into sequencers. The actions to be executed are defined within the individual steps. There are transitions between the steps. These contain conditions for advancing to the next step.

Sequential control system modules

A sequential control system controls the process in a pre-defined order and subject to certain conditions. The complexity of the sequential control system is specified by the automation task. At least three blocks always belong to a sequential control system.

- GRAPH function block
  In the GRAPH function block, you describe the individual steps and transitions of the sequential control system within one or more sequencers.
- Instance data block
  The instance DB contains the data and parameters of the sequential control system. It is assigned to the GRAPH function block and can be automatically generated by the system.
- Calling code block
  In order for the GRAPH function block to be executed within the cycle, it must be called from a higher-level code block. This block can be an organization block (OB), a function (FC), or another function block (FB). The GRAPH function block is always called as a single instance.

The following figure shows how the blocks work together in a sequential control system:



① Calling block

② GRAPH function block

③ Instance data block

The cycle of a GRAPH function block is defined by the cycle of the calling block. In each cycle, the permanent pre-instructions are executed first in the GRAPH function block. Then the actions of the active steps are processed. Last, the permanent post-instructions are executed.

Note

Permanent instructions are executed in each cycle, even if no step is active.

See also

Sequencers (Page 2720)

Steps and transitions (Page 2724)

GRAPH-specific operands (Page 2727)

Actions (Page 2728)

Permanent instructions (Page 2739)

Conditions (Page 2739)

Interlock and supervision alarms (Page 2740)

Block interface of a GRAPH function block (Page 2741)

Sequencers

Basics on sequencers

Introduction

Within a GRAPH function block, you can write your program in the form of sequencers. The sequencers can either process independent tasks, or you can break down a complex task into multiple sequencers. If the sequencers are processing independent tasks, they are processed in parallel in the program flow. If you are using multiple sequencers to break down a complex task into smaller segments, you must link the associated sequencers with jumps.

You subdivide the program into individual steps within a sequencer. In the simplest case, the steps are processed linearly one after the other. However, the use of alternative or simultaneous branches allows you to create sequencers with more complex structures. The program execution always begins with a step that you have defined as the initial step. A sequencer can have one or more initial steps. The initial steps can be at any position in the sequencer.

When a step is active, its actions are executed. More than one step can be be active simultaneously, e.g., within a simultaneous branch. While a step is active, the conditions of the following transition are constantly checked. As soon as all conditions are satisfied and no supervision errors are pending, the transition advances to the next step. This now becomes the active step. You can complete a sequencer either with a jump or a sequence end. The destination of the jump can be any step of the same sequencer or another sequencer. This enables cyclic processing of the sequencer.

Processing principle of a sequencer

The processing of a sequencer begins when the initial step(s) of the sequence become active. Several initial steps are permitted only in simultaneous branches. As long as a step is active, its actions are executed. The interlock conditions for the individual actions are taken into consideration here. Once all possible actions have been executed, a check is first made for the existence of a supervision error. If no such error exists and the following transition condition is satisfied, the next step of the sequence becomes active. If a supervision error exists or the transition condition is not satisfied, the step remains active until the error is eliminated or the transition condition is satisfied. At the end of a sequencer, you can use jumps to enable cyclic processing of the sequencer or a sequence end to terminate the sequencer.

See also

Elements of a sequencer (Page 2721)

Programming the structure of a sequencer (Page 2774)

GRAPH programming language (Page 2718)

GRAPH-specific operands (Page 2727)

Permanent instructions (Page 2739)

Conditions (Page 2739)

Interlock and supervision alarms (Page 2740)

Steps and transitions (Page 2724)

Actions (Page 2728)

Block interface of a GRAPH function block (Page 2741)

Elements of a sequencer

Overview

The following table shows the possible elements of a sequencer along with their icons:

Icon	Name
	Step and transition (Page 2724)
	Step (Page 2724)
	Transition (Page 2724)
	Simultaneous branch
	Alternative branch
	Close branch
	Jump to step
	Sequence end

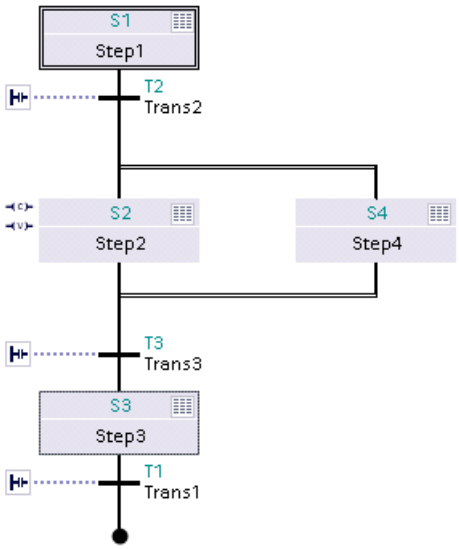
Simultaneous branch

You can use simultaneous branches to program AND branches. This means that you use one transition to activate multiple steps whose actions are then executed. Simultaneous branches always start with a step.

The subsequent transitions of the simultaneous branches are located on the main branch, in which case the various simultaneous branches can be connected to various points of the main branch. Note that branches that you join together in a transition will only advance to the next step when all branches are completely processed.

You can program up to 249 simultaneous branches in a sequencer.

The following figure shows the graphic representation of a simultaneous branch:

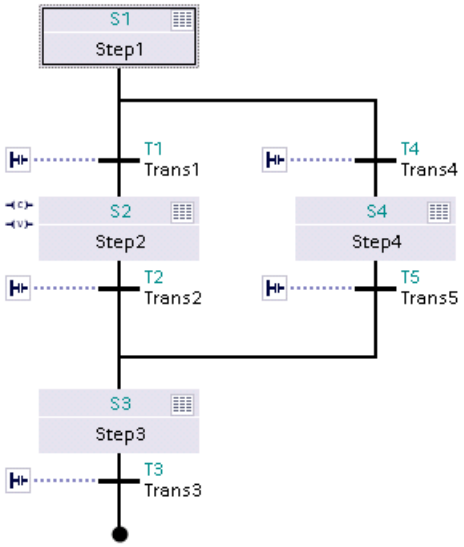


Alternative branch

You can use alternative branches to program OR branches. This means that you insert branches that start with a transition after a step. Depending of which transition's condition is satisfied first, the corresponding branch is executed. If the conditions of multiple transitions are satisfied simultaneously, the transition farthest to the left has the highest priority and the corresponding branch is executed.

You can program up to 125 alternative branches in a sequencer.

The following figure shows the graphic representation of an alternative branch:



Close branch

You can use the "Close branch" element to close simultaneous and alternative branches to their parent branch. This is necessary if you do not want to complete the branch with a jump or a sequence end.

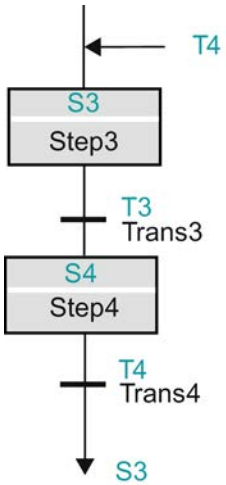
Jump to step

You can use a jump to continue the program execution at any step within the GRAPH function block. Jumps can be inserted at the end of the main branch, an alternative branch, or a simultaneous branch to enable cyclic processing of the sequencer. The jump and jump destination are represented in the sequencer as arrows, in which case the return transition is specified for the jump destination and the destination step is specified for the return.

Note

Avoid jumps from a transition to a directly preceding step. If you need this type of jump, insert an empty step with a transition without condition.

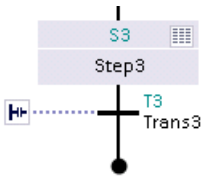
The following figure shows the graphic representation of a jump:



Sequence end

You can complete the sequencer or a branch with the "Sequence end" element. This terminates the sequencer or the branch. In a simultaneous branch, a transition must precede the sequence end. The "Sequence end" element is represented in the sequencer with a black dot.

The following figure shows the graphic representation of a sequence end:



Note

If all branches of a sequencer are completed with a sequence end, the sequencer can be restarted by the "INIT_SQ" parameter or using the "Initialize" button in the "Sequence control" pane of the "Testing" task card.

See also

Basics on sequencers (Page 2720)

Programming the structure of a sequencer (Page 2774)

GRAPH programming language (Page 2718)

Steps and transitions (Page 2724)

GRAPH-specific operands (Page 2727)

Actions (Page 2728)

Permanent instructions (Page 2739)

Conditions (Page 2739)

Interlock and supervision alarms (Page 2740)

Block interface of a GRAPH function block (Page 2741)

Steps and transitions

Basics on steps

Introduction

Steps are used to subdivide complex automation tasks into straightforward subtasks that can then be accomplished using actions. The individual steps are organized in sequencers so that every step can be executed in the specified order in the program flow. Each step must be assigned a unique name and number.

The actions of a step are carried out when the step becomes active. For this a step must either be an initial step or the transition condition preceding it must be satisfied. A step can also be called by an event-dependent action. In addition, you can program interlock and supervision conditions in a step.

You are free to choose the step at which the processing of the sequencer is to start. For this you specify any step as the initial step. In simultaneous branches you can specify multiple steps as initial steps. During cyclic operation of a sequencer, the initial step is treated like every other step, provided there is no initialization of the sequencer using the "INIT_SQ" parameter. If the sequencer is reinitialized, all initial steps are activated when there is a positive signal edge at "INIT_SQ". All other steps are deactivated. Initial steps are framed in the navigation view and sequence view to make them easy to recognize.

A step in which no actions are programmed is an empty step. Since an empty step does not have to process an action, the sequencer advances immediately to the following transition. This allows you to program two consecutive transitions.

You can change the name and number of a step at any time. In addition, you have the option of documenting steps using comments.

See also

Step elements (Page 2725)

Basics on transitions (Page 2726)

Programming the structure of a sequencer (Page 2774)

GRAPH programming language (Page 2718)

Sequencers (Page 2720)

GRAPH-specific operands (Page 2727)

Actions (Page 2728)

Permanent instructions (Page 2739)

Conditions (Page 2739)

Interlock and supervision alarms (Page 2740)

Block interface of a GRAPH function block (Page 2741)

Step elements

You can program the following components in a step:

- Interlock
- Supervision
- Actions

Interlock

You use an interlock to program conditions on which the execution of the individual actions depends. Actions of a step that are linked with the interlock are only executed when the interlock condition is satisfied. If the interlock condition is not satisfied, a fault exists. You can specify alarms for this case, which are then displayed. The fault does not affect the advancement to the next step, however. Note also that the interlock is automatically canceled when the step is deactivated. Optionally, you can specify that the operator must acknowledge an interlock alarm in order to close it.

You also have the option to not program any conditions in the interlock but to still link the interlock with actions. In this case, the interlock condition is always satisfied.

Supervision

You use a supervision to program conditions with which you can monitor a whole step. If the condition is satisfied, a fault exists and the step remains active. That is, in this case the

sequencer does not advance to the next step but instead remains in the active step until the fault has been eliminated. You can also specify alarms and mandatory acknowledgement for the supervision.

Actions

Actions are programmed within the steps. For example, actions control inputs or outputs, activate or deactivate other steps of the sequencer, or call blocks. Actions thus contain instructions for process control.

Actions are differentiated into standard actions and event-dependent actions. Standard actions are executed as long as the step is active. Event-dependent actions are those that you have linked with an event.

See also

Basics on steps (Page 2724)

Basics on transitions (Page 2726)

GRAPH programming language (Page 2718)

GRAPH-specific operands (Page 2727)

Permanent instructions (Page 2739)

Conditions (Page 2739)

Interlock and supervision alarms (Page 2740)

Programming the structure of a sequencer (Page 2774)

Sequencers (Page 2720)

Actions (Page 2728)

Block interface of a GRAPH function block (Page 2741)

Basics on transitions

Introduction

Transitions are located between the steps and contain the conditions for advancing to the next step. That is, the sequencer only switches to the subsequent steps when the conditions of the transition are satisfied. In the process, the steps belonging to the transition are deactivated and the subsequent steps are activated. If the conditions are not satisfied, the steps belonging to the transition remain active. Each transition must be assigned a unique name and number. You can program the conditions of a transition in LAD or FBD.

A transition in which no conditions are programmed is an empty transition. In this case, the sequencer advances immediately to the subsequent steps.

You can change the name and number of a transition.

See also

Programming the structure of a sequencer (Page 2774)
GRAPH programming language (Page 2718)
Sequencers (Page 2720)
GRAPH-specific operands (Page 2727)
Actions (Page 2728)
Permanent instructions (Page 2739)
Conditions (Page 2739)
Interlock and supervision alarms (Page 2740)
Block interface of a GRAPH function block (Page 2741)
Step elements (Page 2725)

GRAPH-specific operands

Introduction

GRAPH provides you with predefined operands you can use to read and then process certain information. You can use these operands in transitions, interlocks, supervisions, actions, and permanent instructions.

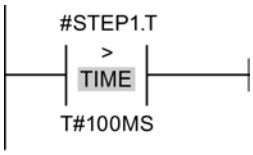
The following table provides an overview of GRAPH-specific operands:

Operand	Meaning	Usage
<Step_name>.T	Current or last activation time of the specified step	Comparator, assignment
<Step_name>.U	Total activation time of the specified step without the fault time	Comparator, assignment
<Step_name>.X	Indicates whether the specified step is active	Normally open contact/Normally closed contact
<Transition_name>.TT	Indicates whether all conditions of the specified transition are satisfied	Normally open contact/Normally closed contact

Example of use of the "Greater than" instruction with the "<Step_name>.T" operand

To monitor the duration of a process, you can program a supervision condition using the "Greater than" instruction and the "<Step_name>.T" operand. This monitors the entire duration of a step.

The following figure shows how the instruction works:

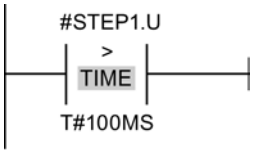


As long as the activation time of "Step1" is less than 100 ms, the result of logic operation (RLO) is "0". As soon as the activation time exceeds 100 ms, the RLO changes to "1".

Example of use of the "Greater than" instruction with the "<Step_name>.U" operand

To determine the activation time without fault times, you use the "<Step_name>.U" operand. This subtracts the times during which the step was in fault state from the activation time.

The following figure shows how the instruction works:



As long as the activation time minus any faults of "Step1" is less than 100 ms, the RLO is "0". As soon as the activation time exceeds 100 ms, the RLO changes to "1".

See also

GRAPH programming language (Page 2718)

Sequencers (Page 2720)

Steps and transitions (Page 2724)

Actions (Page 2728)

Permanent instructions (Page 2739)

Conditions (Page 2739)

Interlock and supervision alarms (Page 2740)

Block interface of a GRAPH function block (Page 2741)

Actions

Basics on actions

Introduction

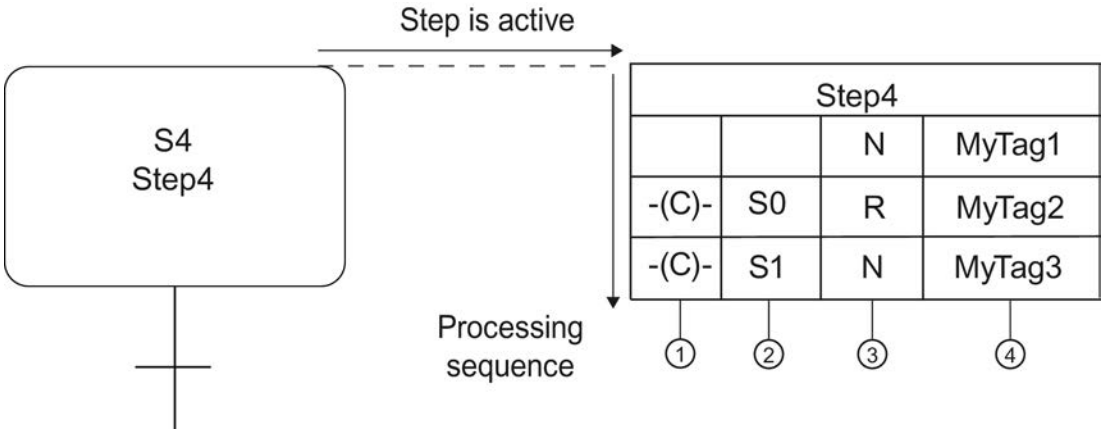
You can use actions, for example, to control inputs or outputs, activate or deactivate other steps of the sequencer, or call blocks. Actions are executed when the associated step becomes active (standard actions) or when an event linked with the action occurs (event-dependent actions). You can also use timers and counters in actions.

The following table shows examples of programming of actions:

Interlock	Event	Identifier	Action	Description
		N	MyTag1	Action consisting of instruction and operand without interlock
-(C)-		N	MyTag2	Action consisting of instruction and operand with interlock
	S1	S	MyTag3	Action consisting of event, instruction, and operand without interlock

Actions are executed in order from "top" to "bottom" when the step is active or the event occurs.

The following figure shows an example of the order in which actions are processed in a step:



① Interlock

② Optional event

③ Identifier

④ Operand, assignment, or block call

An action consists of the following components:

- Interlock (optional)
  You can link an action with an interlock in order to influence the execution of the action.
- Event (optional)
  An event defines when an action is to be executed. You must specify an event for some identifiers.

- Identifier (mandatory)
  The identifier defines the type of action to be executed, e.g., the setting or resetting of an operand.
- Action (mandatory)
  The action determines the operands used to execute the action.

See also

Standard actions (Page 2730)

Event-dependent actions (Page 2732)

Block calls (Page 2735)

Timers (Page 2736)

Counter operations (Page 2737)

Programming actions (Page 2792)

GRAPH programming language (Page 2718)

Sequencers (Page 2720)

Steps and transitions (Page 2724)

GRAPH-specific operands (Page 2727)

Permanent instructions (Page 2739)

Conditions (Page 2739)

Interlock and supervision alarms (Page 2740)

Block interface of a GRAPH function block (Page 2741)

Standard actions

Introduction

Standard actions are executed as long as a step is active. The following table shows the possible standard actions and their identifiers:

Identifier	Data type of the operand	Meaning
N - Set as long as step is active	BOOL FB, FC, SFB, SFC	The signal state of the operand is "1" as long as the step is active. The specified block is called as long as the step is active.
S - Set to 1	BOOL	As soon as the step is active, the operand is set to "1" and then remains at "1".
R - Set to 0	BOOL	As soon as the step is active, the operand is set to "0" and then remains at "0".

Identifier	Data type of the operand	Meaning
D - On delay	BOOL and Time/ DWORD	n seconds after the step activation, the operand is set to "1" and remains at "1" for the duration of the step activation. This does not apply if the duration of the step activation is shorter than n seconds. You can specify the time as a constant or as a PLC tag of data type TIME/ DWORD.
L - Set for limited time	BOOL and Time/ DWORD	When the step is active, the operand is set to "1" for n seconds. The operand is then reset. The operand is also reset if the step activation time is less than the time duration. You can specify the time as a constant or as a PLC tag of data type TIME/ DWORD.

You can link all of these standard actions with an interlock. If an action is linked with an interlock, the action is only executed if the conditions of the interlock are satisfied.

The following table shows some examples of the use of time constants:

Identifier	Action	Description
D	"MyTag", T#2s	2 seconds after step activation, the "MyTag" operand is set to "1" and remains at "1" for the duration of the step activation. This does not apply if the duration of the step activation is shorter than 2 seconds. The operand is reset with step deactivation (not latching).
L	"MyTag", T#20s	If the step is active, the "MyTag" operand is set to "1" for 20 seconds. The operand is then reset (not latching). The operand is also reset if the step activation time is less than 20 seconds.

See also

Basics on actions (Page 2728)

Event-dependent actions (Page 2732)

Block calls (Page 2735)

Timers (Page 2736)

Counter operations (Page 2737)

Programming actions (Page 2792)

GRAPH programming language (Page 2718)

Sequencers (Page 2720)

Steps and transitions (Page 2724)

GRAPH-specific operands (Page 2727)

Permanent instructions (Page 2739)

Conditions (Page 2739)

Interlock and supervision alarms (Page 2740)

Block interface of a GRAPH function block (Page 2741)

Event-dependent actions

Introduction

You have the option of linking an action with an event in order to make the execution of the action dependent on certain conditions. For actions with identifiers "D", "L", and "TF", the linking with events is not possible. If an action is linked with an event, the signal state of the event is detected by edge evaluation. This means that the instructions are only executed in the cycle in which the event occurs.

The following table shows the events that you can link with actions:

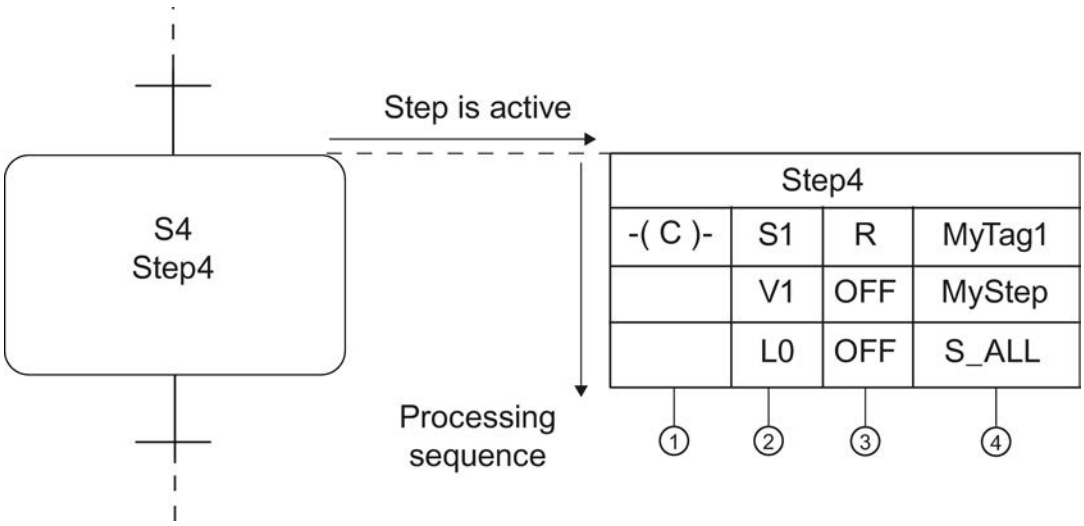
Event	Signal evaluation	Description
S1	Positive edge	Step is activated (signal state = "1")
S0	Negative edge	Step is deactivated (signal state = "0")
V1	Positive edge	Supervision condition is satisfied, i.e., fault occurs (signal state = "1")
V0	Negative edge	Supervision condition is no longer satisfied, i.e., fault is eliminated (signal state = "0")
L0	Positive edge	Interlock condition is satisfied, i.e., fault is eliminated (signal state = "1")
L1	Negative edge	Interlock condition is not satisfied, i.e., fault occurs (signal state = "0")
A1	Positive edge	An alarm is acknowledged.
R1	Positive edge	Incoming registration.

The signal states of an event can be used to program additional actions. This allows you to monitor and influence not only the individual steps but also the entire sequential control system.

Actions for which you use the identifiers "ON" (activate step) and "OFF" (deactivate step) must always be linked with an event. The event determines the timing of the activation or deactivation of the step. If a step is both activated and deactivated in a cycle, the deactivation has higher priority.

You can additionally connect actions for which you use the "S1", "V1", "A1", or "R1" events with an interlock. As a result of this, the actions are only executed if the conditions of the interlock are satisfied.

The following figure shows an example of a step that contains event-dependent actions:



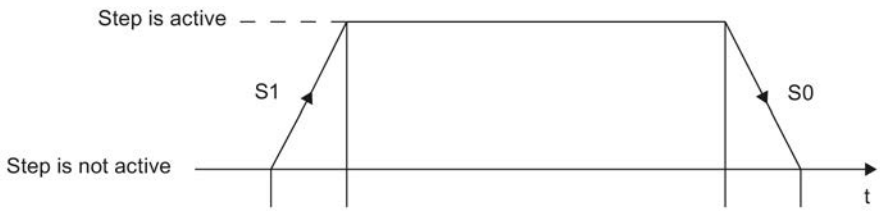
As soon as the "Step4" step becomes active and the interlock condition is satisfied, "MyTag1" is set to "0" and then remains at "0". If a supervision error occurs, the "MyStep" step is deactivated. If an interlock error occurs, all steps are deactivated.

Registration

A registration is an event that is triggered outside the block. Registrations are queried via a positive signal edge at the "REG_S" or "REG_EF" input parameter. If the registration occurs via the "REG_S" input parameter, the event is only forwarded to the active step that is indicated in the "S_NO" output parameter. If the registration is via the "REG_EF" input parameter, the event is forwarded to all currently active steps.

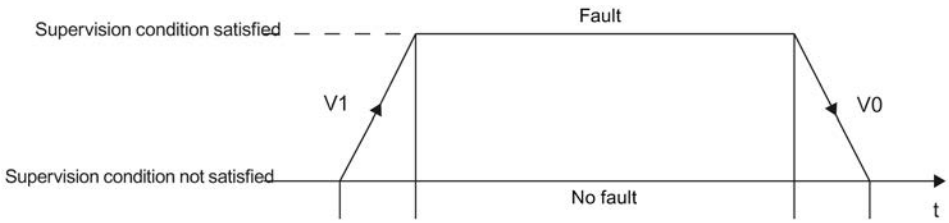
Examples of the signal evaluation of events

The following figure shows the signal evaluation for a step:



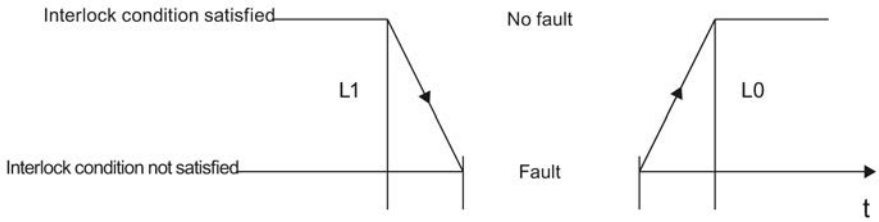
- S1: Step becomes active
- S0: Step becomes inactive

The following figure shows the signal evaluation for a supervision:



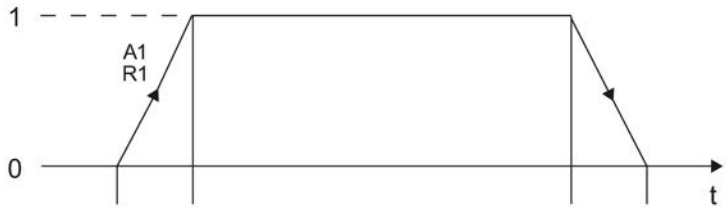
- V1: Supervision error occurs
- V0: Supervision error is eliminated

The following figure shows the signal evaluation for an interlock:



- L1: Interlock condition no longer satisfied (incoming fault)
- L0: Interlock condition satisfied (outgoing fault)

The following figure shows the signal evaluation for an alarm and the registration:



- A1: An alarm is acknowledged.
- R1: Incoming registration (positive edge at input REG_EF/REG_S)

See also

Basics on actions (Page 2728)
Standard actions (Page 2730)
Block calls (Page 2735)
Timers (Page 2736)
Counter operations (Page 2737)
Programming actions (Page 2792)
GRAPH programming language (Page 2718)
Sequencers (Page 2720)
Steps and transitions (Page 2724)

GRAPH-specific operands (Page 2727)

Permanent instructions (Page 2739)

Conditions (Page 2739)

Interlock and supervision alarms (Page 2740)

Block interface of a GRAPH function block (Page 2741)

Block calls

Introduction

Within actions you can call other function blocks and functions in which you perform certain subtasks. This allows you to better structure your program.

Call of function blocks (FB)

Use the following syntax for calling a function block:
```
CALL "<FBName>", "<DBName>" (list of parameters)
```

The following table shows an example of the call of a function block:

Interlock	Event	Identifier	Action
		N	CALL "FB_KOP", "FB_KOP_DB" (MyInOut := "MyTag2")

The "FB_KOP" function block is called using the "CALL" instruction. The "CALL" instruction is always connected with the "N" identifier. After the name of the called block is the name of the data block containing the data and parameters of the function block. In the list of parameters, the "MyTag2" tag is assigned to the "MyInOut" parameter.

Call of functions (FC)

Use the following syntax for the call of a function:
```
CALL "<FCName>" (list of parameters)
```

The following table shows an example of the call of a function:

Interlock	Event	Identifier	Action
		N	CALL "FC_KOP" (MyInput := "MyTag1")

The "FC_KOP" function is called using the "CALL" instruction. In the list of parameters, the "MyTag1" tag is assigned to the "MyInput" parameter.

See also

Block calls (Page 2180)

Inserting block calls manually (Page 2812)

Inserting block calls using drag-and-drop operation (Page 2813)
Updating block calls in GRAPH (Page 2814)
Basics on actions (Page 2728)
Standard actions (Page 2730)
Event-dependent actions (Page 2732)
Timers (Page 2736)
Counter operations (Page 2737)
GRAPH programming language (Page 2718)
GRAPH-specific operands (Page 2727)
Permanent instructions (Page 2739)
Conditions (Page 2739)
Interlock and supervision alarms (Page 2740)
Programming actions (Page 2792)
Sequencers (Page 2720)
Steps and transitions (Page 2724)
Block interface of a GRAPH function block (Page 2741)

Timers

Introduction

You can use timers within actions. Except for "TF", all timers are dependent on an event that determines when the timer is activated. The "TF" timer is activated by the step itself.

When the "TL", "TD", and "TF" timers are used, you must specify the duration. Use the following syntax for this:

```
<Duration> = IWy, QWy, MWy, LWy, DBWy, DIWy; tag of type S5TIME or
WORD; S5T#Constant
```

y = 0 ... 65534

You can also enter the constants as normal time information, such as "5 s".

The following table shows the timers you can use in actions:

Event	Identifier	Description
S1, S0, L1, L0, V1, V0, A1, R1	TL	Extended pulse: As soon as the defined event occurs, the timer starts. During the specified duration, the timer status has signal state "1". After the time expires, the timer status has signal state "0".
S1, S0, L1, L0, V1, V0, A1, R1	TD	Retentive on-delay: As soon as the defined event occurs, the timer starts. During the specified duration, the timer status has signal state "0". After the time expires, the timer status has signal state "1".
S1, S0, L1, L0, V1, V0, A1, R1	TR	Stop timer and reset: As soon as the defined event occurs, the timer is stopped. Timer status and time value are reset to 0.
-	TF	Off delay: The counter status is set to "1" as soon as the step is activated. The timer starts running when the step is deactivated, but the timer status is reset to "0" only after the time expires.

See also

Basics on actions (Page 2728)

Standard actions (Page 2730)

Event-dependent actions (Page 2732)

Block calls (Page 2735)

Counter operations (Page 2737)

Programming actions (Page 2792)

GRAPH programming language (Page 2718)

Sequencers (Page 2720)

Steps and transitions (Page 2724)

GRAPH-specific operands (Page 2727)

Permanent instructions (Page 2739)

Conditions (Page 2739)

Interlock and supervision alarms (Page 2740)

Block interface of a GRAPH function block (Page 2741)

Counter operations

Introduction

You can use counters within actions. Counters always require an event in order to specify when the counter is activated. This means that the counter is activated when the associated event occurs. You can additionally connect actions for which you use the S1, V1, A1, or R1 events with an interlock. As a result of this, the actions are only executed if the conditions of the interlock are satisfied.

The following table shows the counters you can use in actions:

Event	Identifier	Data type of the operand	Meaning
S1, S0, L1, L0, V1, V0, A1, R1	CS	COUNTER	Set initial counter value: As soon as the defined event occurs, the counter is set to the specified count value. You can specify the counter value as a tag or constant of data type WORD (C#0 to C#999).
S1, S0, L1, L0, V1, V0, A1, R1	CU	COUNTER	Count up: As soon as the defined event occurs, the counter is incremented by "1". The counter value can be incremented until the limit of "999" is reached. When the limit is reached, the counter value is no longer incremented on a positive edge.
S1, S0, L1, L0, V1, V0, A1, R1	CD	COUNTER	Count down: As soon as the defined event occurs, the counter is decremented by "1". The counter value can be decremented until the limit of "0" is reached. When the limit is reached, the counter value is no longer decremented on a positive edge.
S1, S0, L1, L0, V1, V0, A1, R1	CR	COUNTER	Reset counter: As soon as the defined event occurs, the counter is reset to "0".

See also

Basics on actions (Page 2728)

Standard actions (Page 2730)

Event-dependent actions (Page 2732)

Block calls (Page 2735)

Timers (Page 2736)

Programming actions (Page 2792)

GRAPH programming language (Page 2718)

Sequencers (Page 2720)

Steps and transitions (Page 2724)

GRAPH-specific operands (Page 2727)

Permanent instructions (Page 2739)

Conditions (Page 2739)

Interlock and supervision alarms (Page 2740)

Block interface of a GRAPH function block (Page 2741)

Permanent instructions

Introduction

Permanent instructions allow you to create program code that is executed before or after the sequencer is processed. This means you can program conditions and block calls that are processed in every cycle independently of the sequencer.

You can create permanent instructions with the LAD and FBD programming languages. A maximum of 32 LAD or FBD instructions can be programmed in one permanent instruction. You can use up to 250 permanent pre-instructions and 250 permanent post-instructions within a GRAPH function block. You transfer the result of logic operation of the conditions to an assignment for which the "Set" and "Reset" memory functions are also available. The operand used (e.g., a bit memory) can be queried in transition, interlock, supervision, and other permanent networks.

See also

"Permanent pre-instructions" view (Page 2764)

"Permanent post-instructions" view (Page 2769)

GRAPH programming language (Page 2718)

Sequencers (Page 2720)

Steps and transitions (Page 2724)

GRAPH-specific operands (Page 2727)

Actions (Page 2728)

Conditions (Page 2739)

Interlock and supervision alarms (Page 2740)

Block interface of a GRAPH function block (Page 2741)

Conditions

Introduction

Conditions are binary states of the process that can be linked together as LAD or FBD elements (e.g., normally open contact, normally closed contact, AND logic operation, OR logic operation, comparator) according to the Boolean logic in the ladder/function block diagram. The result of logic operation (RLO) can influence individual actions of a step, the whole step, the advancement to the next step, or the entire sequencer.

You use conditions to program the following components of your GRAPH program:

- Permanent instructions
- Interlocks
- Supervisions
- Transitions

Depending on the network, you can program up to 32 instructions to which a maximum of 32 operands are interconnected. You can have additional information about the utilized tags displayed by showing the tag information.

See also

GRAPH programming language (Page 2718)

Sequencers (Page 2720)

Steps and transitions (Page 2724)

GRAPH-specific operands (Page 2727)

Actions (Page 2728)

Permanent instructions (Page 2739)

Interlock and supervision alarms (Page 2740)

Block interface of a GRAPH function block (Page 2741)

Interlock and supervision alarms

Introduction

You can use interlocks and supervisions to detect faults. In order to quickly eliminate these faults, you have the option of specifying alarms that describe the faults in detail. To do this, you first activate the alarm generation and then adapt the alarm texts to meet your requirements. You can also specify that the alarms have to be acknowledged.

See also

Specifying interlock and supervision alarms (Page 2815)

GRAPH programming language (Page 2718)

Sequencers (Page 2720)

Steps and transitions (Page 2724)

GRAPH-specific operands (Page 2727)

Actions (Page 2728)

Permanent instructions (Page 2739)

Conditions (Page 2739)

Block interface of a GRAPH function block (Page 2741)

Block interface of a GRAPH function block

Basics on the block interface of a GRAPH function block

Interface parameter sets

When creating a GRAPH block, you can choose between the following interface parameter sets, each of which provides you with a different set of input and output parameters:

- Maximum interface parameter set
- Standard interface parameter set

The standard interface parameter set enables operation of the sequencer in various operating modes and provides the option of acknowledging alarms. The maximum interface parameter set provides additional diagnostics. You can manually delete or insert individual parameters in both parameter sets.

Note

When selecting, also take into consideration the available memory space in your CPU, because the memory space required for the GRAPH block and the associated instance data block increases as the number of parameters increases.

See also:

Input parameters (Page 2743)

Output parameters (Page 2746)

Memory space models for the instance data block (S7-300/400)

For CPUs of the S7-300/400 series, you can additionally specify the memory space model of the instance data block. You have the following options:

- Standard data block
- Minimized data block

You define the number of static parameters via the selection of the memory model. Generally, both memory models store information on the individual steps and transitions as static parameters. A separate structure with detailed information is created for each step and each transition. You receive more detailed information in a standard data block than in a minimized data block. However a standard data block requires considerably more memory space.

The following restrictions apply to a minimized data block:

- Some elements of the steps and transition structures are unavailable.
- The "Skip steps" option is unavailable.
- No alarms are available.
- Step and transition numbers must be consecutive. If not, the steps and transitions are automatically renumbered during compilation.
- No data is generated for the criteria analysis.

See also: Static parameters (Page 2749)

Access to interface parameters

Generally, with an S7-1500 CPU, you do not have write access to the GRAPH-internal interface parameters. However, the following components of the "MOP" structure in the "Static" section that manage the operating modes allow write access:

- AUTO
- MAN
- TAP
- TOP
- LOCK
- SUP
- OFF
- INIT
- HOLD
- TMS_HALT
- OPS_ZERO
- S_PREV
- S_NEXT
- S_ON
- S_OFF
- T_PUSH
- ACK
- IL_PERM
- T_PERM
- ILP_MAN

These parameters react to a rising edge. For S7-300/400 CPUs, also avoid write access to parameters other than those listed. Otherwise, error-free processing of the sequencer cannot be guaranteed.

See also: Other static parameters (Page 2752)

See also

Specifying the interface parameter set (Page 2759)

Specifying the memory space model of the data block (Page 2760)

GRAPH programming language (Page 2718)

Sequencers (Page 2720)

GRAPH-specific operands (Page 2727)
Actions (Page 2728)
Steps and transitions (Page 2724)
Permanent instructions (Page 2739)
Conditions (Page 2739)
Interlock and supervision alarms (Page 2740)

Input parameters

Overview of the input parameters of a GRAPH function block

The input parameters of a GRAPH function block react to a rising edge. The following table provides an overview of the input parameters of a GRAPH function block:

Parameters	Data type	Description	Standard parameter set	Maximum parameter set
OFF_SQ	BOOL	OFF_SEQUENCE: Switch off sequencer, i.e., deactivate all steps Type: Request	x	x
INIT_SQ	BOOL	INIT_SEQUENCE: Activate initial steps, reset sequencer Type: Request	x	x
ACK_EF	BOOL	ACKNOWLEDGE_ERROR_FAULT: Acknowledge a fault, force advance to next step Type: Request	x	x
REG_EF	BOOL	REGISTRATE_ERROR_FAULT: Register all errors and faults Type: Request	-	x
ACK_S	BOOL	ACKNOWLEDGE_STEP: Acknowledge step indicated in "S_NO" output parameter Type: Request	-	x
REG_S	BOOL	REGISTRATE_STEP: Register step indicated in "S_NO" output parameter Type: Request	-	x
HALT_SQ	BOOL	HALT_SEQUENCE: Stop/reactivate sequencer Type: Status, this is reset by the next positive edge	-	x
HALT_TM	BOOL	HALT_TIMES: Stop/reactivate all step activation times and time-dependent operations (L and D) of the sequencer Type: Status, this is reset by the next positive edge	-	x

Parameters	Data type	Description	Standard parameter set	Maximum parameter set
ZERO_OP	BOOL	ZERO_OPERANDS: Reset to zero all operands with identifier N, D, L in active steps and do not execute CALL instructions in actions/ reactivate operands and CALL instructions Type: Status, this is reset by the next positive edge	-	x
EN_IL	BOOL	ENABLE_INTERLOCKS: Disable interlock (sequencer behavior is same as when interlock condition is satisfied)/re-enable Type: Status, this is reset by the next positive edge	-	x
EN_SV	BOOL	ENABLE_SUPERVISIONS: Disable supervision (sequencer behavior is same as when supervision condition is not satisfied)/re-enable Type: Status, this is reset by the next positive edge	-	x
EN_ACKREQ	BOOL	ENABLE_ACKNOWLEDGE_REQUIRED: Enable mandatory acknowledgement Type: Status, this is reset by the next positive edge	-	x
EN_SSKIP	BOOL	ENABLE_STEP_SKIPPING: Enable step skipping Type: Status, this is reset by the next positive edge	-	x
DISP_SACT	BOOL	DISPLAY_ACTIVE_STEPS: Display only active steps Type: Status, this is reset by the next positive edge at DISP_SEF, DISP_SALL	-	x
DISP_SEF	BOOL	DISPLAY_STEPS_WITH_ERROR_OR_FAULT: Display only steps with errors and faults Type: Status, this is reset by the next positive edge at DISP_SACT, DISP_SALL	-	x
DISP_SALL	BOOL	DISPLAY_ALL_STEPS: Display all steps Type: Status, this is reset by the next positive edge at DISP_SACT, DISP_SEF	-	x
S_PREV	BOOL	PREVIOUS_STEP:Automatic mode: Page up through the currently active steps, display of step number in "S_NO" parameter Manual mode: Display preceding step in "S_NO" (smaller number) Type: Request	x	x
S_NEXT	BOOL	NEXT_STEP: Automatic mode: Page down through the currently active steps, display of step number in "S_NO" parameter Manual mode: Display next step in S_NO (larger number) Type: Request	x	x

Parameters	Data type	Description	Standard parameter set	Maximum parameter set
SW_AUTO	BOOL	SWITCH_MODE_AUTOMATIC: Operating mode switchover: Automatic mode Type: Status, this is reset by the next positive edge at SW_TAP, SW_TOP, SW_MAN	x	x
SW_TAP	BOOL	SWITCH_MODE_TRANSITION: Operating mode switchover: Semi-automatic mode Type: Status, this is reset by the next positive edge at SW_AUTO, SW_TOP, SW_MAN	x	x
SW_TOP	BOOL	SWITCH_MODE_TRANSITION_OR_PUSH: Operating mode switchover: Automatic or semi-automatic mode Type: Status, this is reset by the next positive edge at SW_AUTO, SW_TAP, SW_MAN	x	x
SW_MAN	BOOL	SWITCH_MODE_MANUAL: Operating mode switchover: Manual mode, an independent sequence is not initiated Type: Status, this is reset by the next positive edge at SW_AUTO, SW_TAP, SW_TOP	x	x
S_SEL	INT	STEP_SELECT: Select step number for "S_NO" output parameter in manual mode, enable/disable with "S_ON", "S_OFF" Type: N/A	x	x
S_SELOK	BOOL	STEP_SELECT_OK: Apply value in "S_SEL" for "S_NO" output parameter Type: Request	-	x
S_ON	BOOL	STEP_ON: Manual mode: Activate displayed step Type: Request	x	x
S_OFF	BOOL	STEP_OFF: Manual mode: Deactivate displayed step Type: Request	x	x
T_PREV	BOOL	PREVIOUS_TRANSITION: Show previous valid transition in "T_NO" output parameter Type: Request	-	x
T_NEXT	BOOL	NEXT_TRANSITION: Show next valid transition in "T_NO" output parameter Type: Request	-	x
T_PUSH	BOOL	PUSH_TRANSITION: Transition advances to next step, when condition is satisfied and "T_PUSH" (edge) Requirement: Automatic mode or manual mode Type: Request	x	x

See also

Basics on the block interface of a GRAPH function block (Page 2741)
Output parameters (Page 2746)
Static parameters (Page 2749)
Specifying the interface parameter set (Page 2759)
Specifying the memory space model of the data block (Page 2760)
GRAPH programming language (Page 2718)
Sequencers (Page 2720)
Steps and transitions (Page 2724)
GRAPH-specific operands (Page 2727)
Actions (Page 2728)
Permanent instructions (Page 2739)
Conditions (Page 2739)
Interlock and supervision alarms (Page 2740)

Output parameters

Overview of the output parameters of a GRAPH function block

The following table provides an overview of the output parameters of a GRAPH function block:

Parameters	Data type	Description	Standard parameter set	Maximum parameter set
S_NO	INT	STEP_NUMBER: Display of step number	x	x
S_MORE	BOOL	MORE_STEPS: Additional steps are active	x	x
S_ACTIVE	BOOL	STEP_ACTIVE: Displayed step is active	x	x
S_TIME	TIME	STEP_TIME: Step activation time	-	x
S_TIMEOK	TIME	STEP_TIME_OK: Step activation time is error free	-	x
S_CRITLOC (S7-300/400 only)	DWORD	STEP_CRITERIA_INTERLOCK: Interlock criteria bits	-	x
S_CRITLOCERR (S7-300/400 only)	DWORD	S_CRITERIA_IL_LAST_ERROR: Interlock criteria bits for event L1	-	x
S_CRITSUP (S7-300/400 only)	DWORD	STEP_CRITERIA_SUPERVISION: Supervision criteria bits	-	x

Parameters	Data type	Description	Standard parameter set	Maximum parameter set
S_STATE	WORD	STEP_STATE: Step status bits	-	x
T_NO	INT	TRANSITION: Valid transition number	-	x
T_MORE	BOOL	MORE_TRANSITIONS: Additional valid transitions available for display	-	x
T_CRIT	DWORD	TRANSITION_CRITERIA: Transition criteria bits	-	x
T_CRITOLD	DWORD	T_CRITERIA_LAST_CYCLE: Transition criteria bits from the last cycle	-	x
T_CRITFLT	DWORD	T_CRITERIA_LAST_FAULT: Transition criteria bits for event V1	-	x
ERROR	BOOL	INTERLOCK_ERROR: Interlock error (any step)	-	x
FAULT	BOOL	SUPERVISION_FAULT: Supervision error (any step)	-	x
ERR_FLT	BOOL	IL_ERROR_OR_SV_FAULT: General fault	x	x
SQ_ISOFF	BOOL	SEQUENCE_IS_OFF: Sequencer is switched off (no step is active)	-	x
SQ_HALTED	BOOL	SEQUENCE_IS_HALTED: Sequencer is stopped	-	x
TM_HALTED	BOOL	TIMES_ARE_HALTED: Timers are stopped	-	x
OP_ZEROED	BOOL	OPERANDS_ARE_ZEROED: Operands are reset	-	x
IL_ENABLED	BOOL	INTERLOCK_IS_ENABLED: Interlock is taken into consideration	-	x
SV_ENABLED	BOOL	SUPERVISION_IS_ENABLED: Supervision is taken into consideration	-	x
ACKREQ_ENABLED	BOOL	ACKNOWLEDGE_REQUIRED_IS_ENABLED: Mandatory acknowledgement is enabled	-	x
SSKIP_ENABLED	BOOL	STEP_SKIPPING_IS_ENABLED: Step skipping is enabled	-	x
SACT_DISP	BOOL	ACTIVE_STEPS_WERE_DISPLAYED: Only active steps are displayed in "S_NO"	-	x

Parameters	Data type	Description	Standard parameter set	Maximum parameter set
SEF_DISP	BOOL	STEPS_WITH_ERROR_FAULT_WERE_DISPLAYED: Only steps with errors and faults are displayed in "S_NO"	-	x
SALL_DISP	BOOL	ALL_STEPS_WERE_DISPLAYED: All steps are displayed in "S_NO"	-	x
AUTO_ON	BOOL	AUTOMATIC_IS_ON: Display of automatic mode	x	x
TAP_ON	BOOL	T_AND_PUSH_IS_ON: Display of semi-automatic mode	x	x
TOP_ON	BOOL	T_OR_PUSH_IS_ON: Display of semi-automatic mode	x	x
MAN_ON	BOOL	MANUAL_IS_ON: Display of manual mode	x	x

See also

Basics on the block interface of a GRAPH function block (Page 2741)
Input parameters (Page 2743)
Static parameters (Page 2749)
Specifying the interface parameter set (Page 2759)
Specifying the memory space model of the data block (Page 2760)
GRAPH programming language (Page 2718)
Sequencers (Page 2720)
Steps and transitions (Page 2724)
GRAPH-specific operands (Page 2727)
Actions (Page 2728)
Permanent instructions (Page 2739)
Conditions (Page 2739)
Interlock and supervision alarms (Page 2740)

Static parameters

Static parameters of a step

Overview of the static parameters of a step

The following table shows the parameters that are created for each step of the GRAPH function block in the "Static" section of the block interface:

Component	Description	Data type	Minimized DB	Standard DB
S1	Step is activated	BOOL	x	x
L1	Interlock is outgoing	BOOL	x	x
V1	Supervision incoming	BOOL	x	x
R1	Reserved	BOOL	x	x
A1	Error is acknowledged	BOOL	x	x
S0	Step is deactivated	BOOL	x	x
L0	Incoming interlock	BOOL	x	x
V0	Outgoing supervision	BOOL	x	x
X	Step is active	BOOL	x	x
LA	Interlock is unsatisfied	BOOL	x	x
VA	Pending supervision	BOOL	x	x
RA	Reserved	BOOL	x	x
AA	Reserved	BOOL	x	x
SS	System internal	BOOL	x	x
LS	Direct result of the programmed interlock	BOOL	x	x
VS	Direct result of the programmed supervision	BOOL	x	x
SNO	User step number	INT		x
T	Total step activation time	TIME	x	x
U	Uninterrupted step activation time	TIME		x
CRIT_LOC (S7-300/400 only)	State of the max. 32 LAD/FBD elements in interlock in the current process cycle	DWORD		x
CRIT_LOC_ ERR (S7-300/400 only)	Copy of CRIT_LOC, if interlock is outgoing	DWORD		x
CRIT_SUP (S7-300/400 only)	State of the max. 32 LAD/FBD elements in supervision in the current process cycle	DWORD		x
SM	System internal	BOOL		x
LP (S7-300/400 only)	System internal	BOOL	x	x
LN (S7-300/400 only)	System internal	BOOL	x	x
VP (S7-300/400 only)	System internal	BOOL	x	x

Component	Description	Data type	Minimized DB	Standard DB
VN (S7-300/400 only)	System internal	BOOL	x	x
H_IL_ERR	System internal	BYTE		x
H_SV_FLT	System internal	BYTE		x
RESERVED (S7-300/400 only)	Reserve	DWORD		x

Access to components

Write access is not recommended because the parameter values influence the sequence. Reading access takes place via the step name followed by the structure element name. In conditions or actions, e.g. the following accesses are thereby possible:

- TROCKNEN.X
- TROCKNEN.T

See also

Basics on the block interface of a GRAPH function block (Page 2741)

Input parameters (Page 2743)

Output parameters (Page 2746)

Static parameters of a transition (Page 2751)

Other static parameters (Page 2752)

Specifying the interface parameter set (Page 2759)

Specifying the memory space model of the data block (Page 2760)

GRAPH programming language (Page 2718)

Sequencers (Page 2720)

Steps and transitions (Page 2724)

GRAPH-specific operands (Page 2727)

Actions (Page 2728)

Permanent instructions (Page 2739)

Conditions (Page 2739)

Interlock and supervision alarms (Page 2740)

Static parameters of a transition

Transition structure in the interface

The following table shows the parameters that are created for each transition of the GRAPH function block in the "Static" section of the block interface:

Component	Description	Data type	Minimized DB	Standard DB
TV	Transition is valid	BOOL	x	x
TT	Transition is satisfied	BOOL	x	x
TS	Transition is switching	BOOL	x	x
CF_IV (S7-300/400 only)	Entry "CRIT_FLT" is invalid	BOOL		x
TNO	Display of the user-defined transition number	INT		x
CRIT (S7-300/400 only)	State of the max. 32 LAD/FBD transition elements in the current process cycle	DWORD		x
CRIT_OLD (S7-300/400 only)	State of the max. 32 LAD/FBD transition elements in the previous process cycle	DWORD		x
CRIT_FLT (S7-300/400 only)	Copy of "CRIT", if an error occurs	DWORD		x

Access to components

Write access is not recommended because the parameter values influence the sequence. Reading access takes place via the transition name followed by the structure element name. In conditions or actions, e.g. the following accesses are thereby possible:

- VENTIL_EIN.TT
- VENTIL_EIN.CRIT

See also

Basics on the block interface of a GRAPH function block (Page 2741)

Input parameters (Page 2743)

Output parameters (Page 2746)

Static parameters of a step (Page 2749)

Other static parameters (Page 2752)

Specifying the interface parameter set (Page 2759)

Specifying the memory space model of the data block (Page 2760)

GRAPH programming language (Page 2718)

Sequencers (Page 2720)

Steps and transitions (Page 2724)
GRAPH-specific operands (Page 2727)
Actions (Page 2728)
Permanent instructions (Page 2739)
Conditions (Page 2739)
Interlock and supervision alarms (Page 2740)

Other static parameters

Overview

The static parameters include the step and transition parameters as well as the following information:

- Internal parameters "S_DISPLAY", "S_SEL_OLD", "S_DISPIDX", "T_DISPIDX"
- Component "Operating mode"
- Component "Clock speeds"
- Component "Sequence bit memory"

Note

Additional parameters are contained in a GRAPH-specific instance data block. They are used however only for internal application. The use of these parameters in the program is not recommended.

Internal parameters

The following table shows the internal parameters that are located in the "Static" section. They are used to manage the step and transition numbers:

Component	Description	Data type	Length in bytes	Minimized DB	Standard DB
S_DISPLAY	Copy of the "S_NO" output parameter	INT	2	x	x
S_SEL_OLD	Step number from the "S_SEL" input parameter from the last CPU cycle	INT	2	x	x
S_DISPIDX	Step index for the step in "S_NO" output parameter	BYTE	1	x	x
T_DISPIDX	Transition index for the transition in "T_NO" output parameter	BYTE	1	x	x

Operating mode

The following table shows the "MOP" structure components in the section "Static" that manage the operating modes:

Component	Description	Data type	Minimized DB	Standard DB
AUTO	Switch to automatic mode	BOOL	x	x
MAN	Switch to manual mode	BOOL	x	x
TAP	Switch to "Jog" mode	BOOL	x	x
TOP	Switch to "Automatic" or "Advance" mode	BOOL	x	x
ACK_S	Requirement: Acknowledge step that is indicated in "S_NO" output parameter	BOOL	x	x
REG_S	Requirement: Register step that is indicated in "S_NO" output parameter	BOOL	x	x
T_PREV	Requirement: Output previous valid transition in "T_NO" output parameter	BOOL	x	x
T_NEXT	Requirement: Output next valid transition in "T_NO" output parameter	BOOL	x	x
LOCK	Turn on/turn off interlock handling	BOOL	x	x
SUP	Turn on/turn off supervision handling	BOOL	x	x
ACKREQ	Requirement: Acknowledge step that is indicated in "S_NO" output parameter	BOOL	x	x
SSKIP	Status: "Skip step" is activated	BOOL	x	x
OFF	Deactivate sequencer	BOOL	x	x
INIT	Activate initial steps	BOOL	x	x
HOLD	Stop/resume sequence	BOOL	x	x
TMS_HALT	Stop/resume timers	BOOL	x	x
OPS_ZERO	Switch off/switch on actions	BOOL	x	x
SACT_DISP	Status: Display only active steps	BOOL	x	x
SEF_DISP	Status: Display only steps with errors and faults	BOOL	x	x
SALL_DISP	Status: Display all steps	BOOL	x	x
S_PREV	"Automatic" mode: Display previous simultaneously active step in "S_NO" output parameter "Manual" mode: Display next smaller step number	BOOL	x	x
S_NEXT	"Automatic" mode: Display next simultaneously active step in "S_NO" output parameter "Manual" mode: Display next higher step number	BOOL	x	x
S_SELOK	Requirement: Output step number that is indicated in "S_SEL" input parameter in "S_NO" output parameter	BOOL	x	x
S_ON	"Manual" mode: Activate displayed step	BOOL	x	x
S_OFF	"Manual" mode: Deactivate displayed step	BOOL	x	x

Component	Description	Data type	Minimized DB	Standard DB
T_PUSH	"Jog" mode: Transition switches, when the satisfied condition is applied with "T_PUSH"	BOOL	x	x
REG	Requirement: Register all interlock and supervision windows	BOOL	x	x
ACK	Acknowledgement of the fault	BOOL	x	x
IL_PERM	Permanent interlock processing	BOOL	x	x
T_PERM	Permanent transition processing	BOOL	x	x
ILP_MAN	Permanent interlock processing in "Manual" mode	BOOL	x	x

With CPUs of the S7-1500 series, the "MOP" structure is located in the "RT_DATA" structure.

Clock pulses (S7-300/400 only)

The following table shows the "TICKS" structure components in the section "Static" that manages the information for activation time calculation. This information can be evaluated only internally in the system.

Component	Description	Data type	Minimized DB	Standard DB
DELTA	Time difference between two cycles	TIME	x	x
OLD	10 ms clock counter value in the last cycle	TIME	x	x
NEW	10 ms clock counter value in this cycle	TIME	x	x

Sequence bit memory

The following table shows the "SQ_FLAGS" structure components in the section "Static" that monitor the internal sequence status:

Component	Description	Data type	Minimized DB	Standard DB
ERR_FLT	General fault	BOOL	x	x
ERROR	Interlock error	BOOL	x	x
FAULT	Supervision error	BOOL	x	x
RT_FAIL	Runtime error	BOOL	x	x
NO_SNO	Selected step does not exist	BOOL	x	x
NF_OFL	Too many steps that have to be activated or deactivated	BOOL	x	x
SA_OFL	Too many active steps	BOOL	x	x
TV_OFL	Too many valid transitions	BOOL	x	x
MSG_OFL	Too many alarms	BOOL	x	x
NO_SWI	Do not switch in this cycle	BOOL	x	x
CYC_OP	Cyclic processing of the sequencer after initialization	BOOL	x	x
AS_MSG	Activate or deactivate alarms during runtime on instruction	BOOL	x	x

Component	Description	Data type	Minimized DB	Standard DB
AS_SEND	Send alarms from "WR_USMSG" or enter only in diagnostics buffer	BOOL	x	x
SQ_BUSY	Internal edge memory bit for sequence processing	BOOL	x	x
SA_BUSY	Internal edge memory bit for sequence processing	BOOL	x	x
AS_SIG	Edge memory bit for alarms from "Alarm_S" and "Alarm_SQ"	BOOL	x	x

With CPUs of the S7-1500 series, the "SQ_FLAGS" structure is located in the "RT_DATA" structure.

Access to components

Write access is not recommended because the parameter values influence the sequence. Read access is however possible.

See also

Basics on the block interface of a GRAPH function block (Page 2741)

Input parameters (Page 2743)

Output parameters (Page 2746)

Static parameters of a step (Page 2749)

Static parameters of a transition (Page 2751)

Specifying the interface parameter set (Page 2759)

Specifying the memory space model of the data block (Page 2760)

GRAPH programming language (Page 2718)

Sequencers (Page 2720)

Steps and transitions (Page 2724)

GRAPH-specific operands (Page 2727)

Actions (Page 2728)

Permanent instructions (Page 2739)

Conditions (Page 2739)

Interlock and supervision alarms (Page 2740)

Settings for GRAPH

Overview of the settings for GRAPH

Overview

The following table shows the settings that you can make:

Group	Area	Setting	Description
View		Length of step and transition names	Specifies whether step and transition names are to be displayed in one or two lines. The "Show two lines" setting allows longer names to be viewed.
Load		Turn off sequence before downloading DB	All active steps are deactivated before the instance data block is loaded.
Default for time monitoring		Uninterrupted step activation time (U)	Activation time for the time monitoring of a step by the "Greater than uninterrupted step activation time" instruction.
		Step activation time (T)	Activation time for the time monitoring of a step by the "Greater than step activation time" instruction.
Default settings for new blocks	Editor	Language in networks	Selection of the programming language for the programming of conditions.
	Interface	Maximum interface parameters	Inserts the maximum number of parameters into the block interface when new GRAPH function blocks are created.
	Compile	Create minimized DB	Creates an instance data block with a reduced set of static parameters when compiling. This limits the functionality, however. You can change this setting again subsequently for individual blocks. See also: Basics on the block interface of a GRAPH function block (Page 2741)
	Sequence properties	Skip steps	Skips steps in which the next transition condition is already satisfied when the step is activated. In this case the step does not become active. You can change this setting subsequently for individual blocks.
		Acknowledge supervision alarms	Specifies that supervision errors must be acknowledged before the sequencer will continue to be processed.
		Permanent processing of all interlocks in manual mode	Allows the processing of all interlocks in manual mode.
		Lock operating mode selection	Prevents the manual change of operating mode via the "Testing" task card. You can change this setting subsequently for individual blocks.
		Set ENO automatically	Displays the additional "ENO" operand for S7-1500 CPUs when testing with program status.

Group	Area	Setting	Description
	Alarm properties	Enable alarms	Enables the interlock and supervision alarms for new GRAPH function blocks. You can change this setting subsequently for individual blocks.
	Interlock alarms	Acknowledge interlock alarms	Specifies that interlock alarms must be acknowledged before the step can continue to be processed. You can change this setting subsequently for individual blocks.
		Alarm text	Specifies the text for interlock alarms. You can change this text subsequently for individual blocks.
	Supervision alarms	Acknowledge supervision alarms	Specifies that supervision alarms must be acknowledged in order to close them.
		Alarm text	Specifies the text for supervision alarms. You can change this text subsequently for individual blocks.
Synchronization		Preceding transition condition satisfied	Marks the steps whose preceding transition condition is satisfied when determining possible synchronization points. The subsequent transition condition is not satisfied in this case.
		Interlock condition satisfied	Marks the steps whose interlock condition is satisfied when determining possible synchronization points. The subsequent transition condition is satisfied in this case.

See also

Overview of block properties (Page 2512)

Changing the settings (Page 2757)

Specifying the programming language in networks (Page 2758)

Specifying the interface parameter set (Page 2759)

Specifying the memory space model of the data block (Page 2760)

Modifying settings for internal parameters (Page 2761)

Changing the settings

Procedure

To change the settings, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "PLC programming > GRAPH" group in the area navigation.
3. Change the settings.

Result

The change is applied and does not have to explicitly saved.

See also

Overview of the settings for GRAPH (Page 2756)

Specifying the programming language in networks (Page 2758)

Specifying the interface parameter set (Page 2759)

Specifying the memory space model of the data block (Page 2760)

Modifying settings for internal parameters (Page 2761)

Specifying the programming language in networks

For GRAPH function blocks, you cannot change the programming language of the block. However, you can specify the programming language of the networks for conditions. You can choose here between LAD and FBD. You can specify the programming language either in the project tree or in the inspector window.

Specifying the programming language in networks in the project tree

To specify the programming language in networks in the project tree, follow these steps:

1. Right-click the GRAPH function block in the project tree.
2. Select the "Properties" command in the shortcut menu.
   A dialog containing the properties of the block opens.
3. Select the "General" entry in the area navigation.
4. In the "Block" area, select the programming language you want to use from the "Language in networks" drop-down list.
5. Confirm your selection with "OK".

Specifying the programming language in networks in the inspector window

To specify the programming language in networks in the inspector window, follow these steps:

1. Open the GRAPH function block whose programming language you want to change.
2. Open the "Properties" tab in the inspector window.
3. Select the "General" entry in the area navigation in the inspector window.
4. In the "Block" area, select the programming language you want to use from the "Language in networks" drop-down list.

See also

Overview of the settings for GRAPH (Page 2756)

Changing the settings (Page 2757)

Specifying the interface parameter set (Page 2759)

Specifying the memory space model of the data block (Page 2760)

Modifying settings for internal parameters (Page 2761)

Specifying the interface parameter set

You can specify the interface parameter set for GRAPH function blocks as follows:

- Globally for all new GRAPH function blocks
- Locally for an open GRAPH function block

The local setting has priority over the globally setting.

Note

You can manually add or delete individual input and output parameters in the block interface at any time.

Specifying the interface parameter set globally for all new GRAPH function blocks

In order to specify the interface parameter set globally for all new GRAPH function blocks, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "PLC programming > GRAPH" group in the area navigation.
3. If you want to use the maximum interface parameter set, select the "Maximum interface parameters" option.
4. If you want to use the standard interface parameter set, clear the "Maximum interface parameters" option.

Specifying the interface parameter set locally for an open GRAPH function block

In order to specify the interface parameter set for an open GRAPH function block, follow these steps:

1. If you want to use the maximum interface parameter set, select the "Maximum interface parameters" command in the "Edit" menu.
2. If you want to use the standard interface parameter set, select the "Default interface parameter set" command in the "Edit" menu.

The block interface of the GRAPH function block is adapted.

See also

Basics on the block interface of a GRAPH function block (Page 2741)

Overview of the settings for GRAPH (Page 2756)

Changing the settings (Page 2757)

Specifying the programming language in networks (Page 2758)

Specifying the memory space model of the data block (Page 2760)

Modifying settings for internal parameters (Page 2761)

Specifying the memory space model of the data block

You can specify the memory space model for GRAPH function blocks as follows:

- Globally for all new GRAPH function blocks
- Locally for an open GRAPH function block

The local setting has priority over the globally setting.

Note

GRAPH-specific parameters that are not available in the minimized data block cannot be manually inserted in the block interface.

Specifying the memory space model globally for all new GRAPH function blocks

In order to specify the memory space model globally for all new GRAPH function blocks, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "PLC programming > GRAPH" group in the area navigation.
3. If you want to use a minimized data block, select the "Create minimized DB" option.
4. If you want to use a data block with the standard memory space model, clear the "Create minimized DB" option.

Specifying the memory space model locally for an open GRAPH function block

In order to specify the memory space model for an open GRAPH function block, follow these steps:

1. Open the "Properties" tab in the inspector window.
2. Select the "Attributes" entry in the area navigation in the inspector window.
3. If you want to use a minimized data block, select the "Create minimized DB" option in the "Compile" group.
4. If you want to use a data block with the standard memory space model, clear the "Create minimized DB" option in the "Compile" group.

The interface of the GRAPH block is adapted.

See also

Basics on the block interface of a GRAPH function block (Page 2741)

Overview of the settings for GRAPH (Page 2756)

Changing the settings (Page 2757)

Specifying the programming language in networks (Page 2758)

Specifying the interface parameter set (Page 2759)

Modifying settings for internal parameters (Page 2761)

Modifying settings for internal parameters

You can declare the internal parameters of a GRAPH block as visible and accessible from HMI. For S7-1500 CPUs, you can also declare the internal parameters as retentive. If you declare the internal parameters as retentive, their values are retained even after an operating state change from "STOP" to "RUN", for example following a power failure. As a result, the sequencer remains in the current state and continues running from the point at which it was interrupted after the change to "RUN".

Parameters that you have declared yourself in the interface of a GRAPH block are not affected by these settings.

Declaring internal parameters as visible and accessible from HMI

Proceed as follows to declare all internal parameters of a GRAPH block as visible and accessible from HMI:

1. Open the GRAPH block for which you want to change the setting.
2. In the "Edit" menu, select the "Internal parameters visible/accessible from HMI" check box. Clear this check box if you want to reset the setting.

Declaring internal parameters as retentive (only S7-1500)

Proceed as follows to declare all internal parameters of a GRAPH block as retentive:

1. Open the GRAPH block for which you want to change the setting.
2. In the "Edit" menu, select the "Retentive internal parameters" check box. Clear this check box if you want to reset the setting.

See also

Overview of the settings for GRAPH (Page 2756)

Changing the settings (Page 2757)

Specifying the programming language in networks (Page 2758)

Specifying the interface parameter set (Page 2759)

Specifying the memory space model of the data block (Page 2760)

Programming window of GRAPH

Overview of the programming window

Function

The programming window is the area, where you create the GRAPH program. In addition to the actual work area, it contains a navigation view that allows you to easily access all areas of the GRAPH function block.

You can perform the following tasks in the programming window:

- Programming of permanent pre-instructions and post-instructions
- Programming of the sequencers
- Specifying of the interlock and supervision alarms

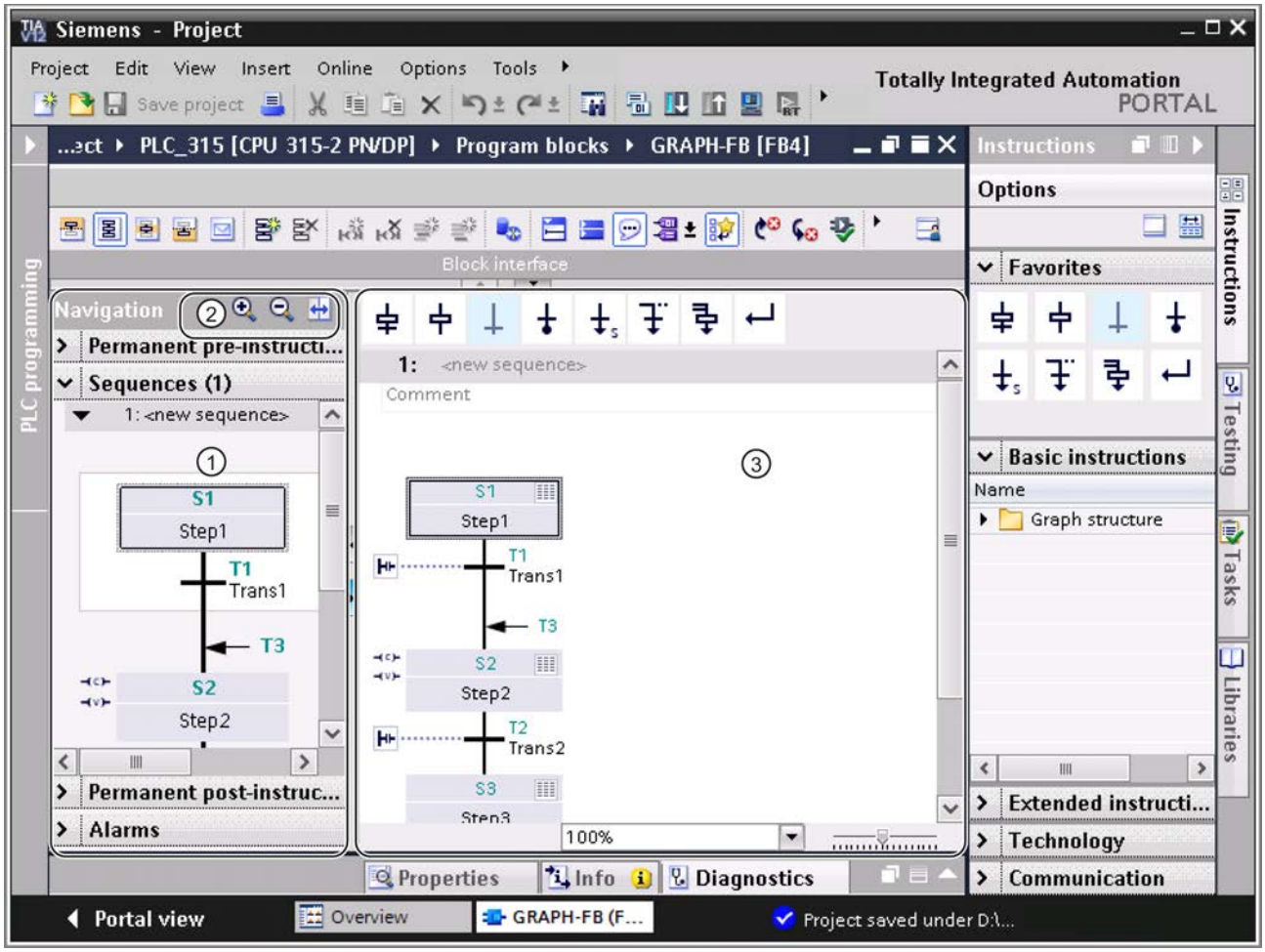
Depending on what you want to program, you can switch between the following views:

- Permanent pre-instructions (Page 2764)
- Sequence view (Page 2765)
- Single step view (Page 2767)
- Permanent post-instructions (Page 2769)
- Alarm view (Page 2770)

The work area and the available instructions and favorites vary depending on the view.

Structure of the programming window

The following figure shows the programming window of GRAPH in the sequence view:



① Navigation view

② Navigation toolbar

③ Work area

Navigation view

The navigation view contains the following pallets you can use to open the following views:

- Permanent pre-instructions
- Sequence view
- Permanent post-instructions
- Alarm view

In addition, the navigation view shows graphic overviews of the permanent instructions and sequencers and offers you basic processing options via the shortcut menu.

You can double-click a step in the navigation view to switch between the sequence and single step views.

Navigation toolbar

You can perform the following actions in the navigation toolbar:

- Zoom in or zoom out of elements within the navigation
  You can use the zoom in and zoom out buttons to zoom in on or zoom out from the graphic elements of the permanent instructions and sequences within the navigation view. You can set the zoom factor separately for each pallet. When the GRAPH function block is closed, any changes to the zoom factor setting are discarded. If you want to retain the zoom factor, you can do so via the "Remember layout" button.
- Synchronizing navigation
  When the button is enabled, the navigation and the work area are synchronized so that the same elements are always displayed. When you disable the button, you can display different objects in navigation and work area.

Work area

You program the individual components of your sequencer in the work area. You can display the GRAPH program in different views for this. You can scale these views using a zoom function.

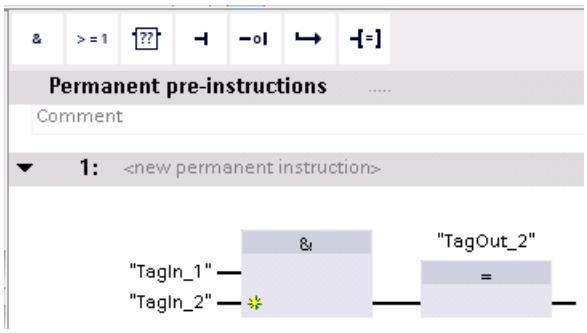
"Permanent pre-instructions" view

Function of the "Permanent pre-instructions" view

In the "Permanent pre-instructions" view, you can program instructions that are executed before the sequencer is processed.

Structure of the "Permanent pre-instructions" view

The following figure shows an example of the "Permanent pre-instructions" view with the FBD programming language:



See also

Permanent instructions (Page 2739)

Overview of the programming window (Page 2762)

Sequence view (Page 2765)

Single step view (Page 2767)

"Permanent post-instructions" view (Page 2769)

Alarm view (Page 2770)

Sequence view

Function of the sequence view

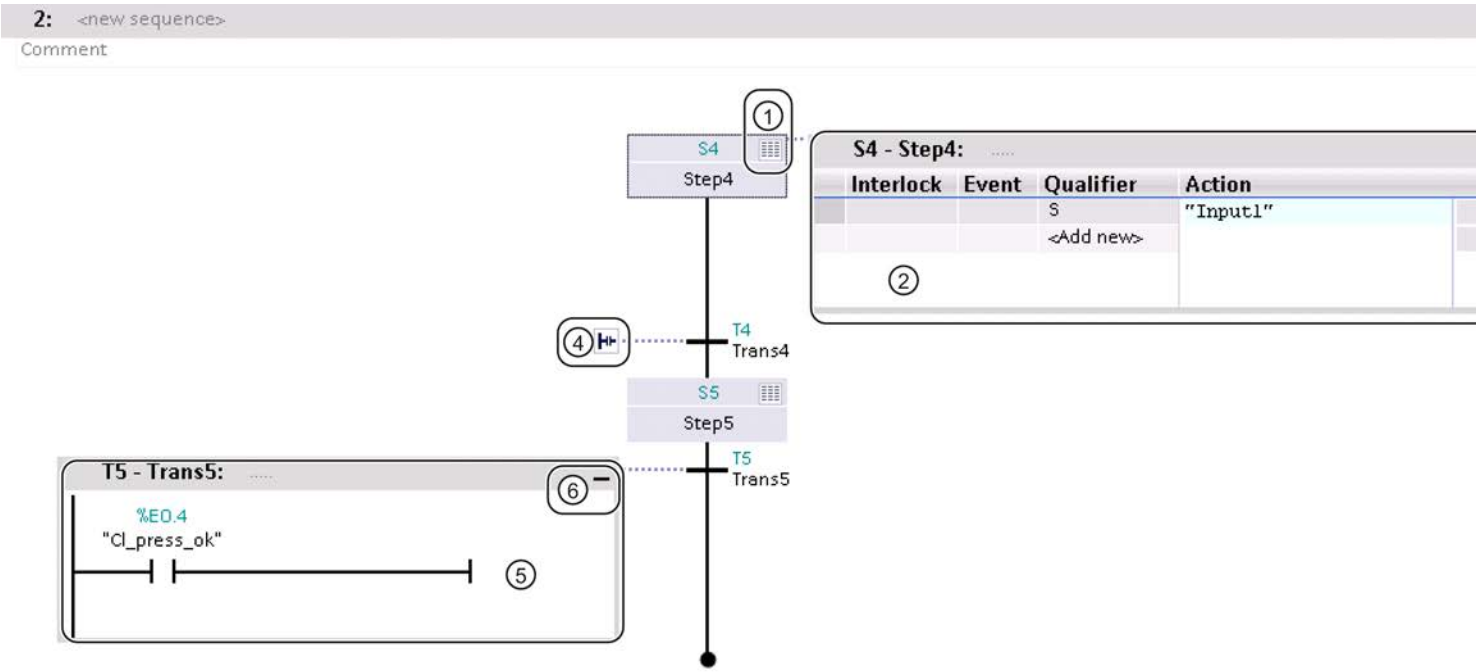
The sequence view shows you the structure of your sequencer in an easy to read format and enables you to add the following elements to it:

- Steps
- Transitions
- Jumps
- Branches
- Sequence ends

In addition, you can expand steps and transitions at the click of a mouse in order to display or edit the actions of a step and the conditions of a transition.

Structure of the sequence view

The following figure shows an example of the sequence view:



① Button for expanding or reducing a step

② Step in the expanded view

③ Button for reducing a step

④ Button for expanding a transition

⑤ Transition in the expanded view

⑥ Button for reducing a transition

See also

Overview of the programming window (Page 2762)

"Permanent pre-instructions" view (Page 2764)

Single step view (Page 2767)

"Permanent post-instructions" view (Page 2769)

Alarm view (Page 2770)

Single step view

Function of the single step view

The single step view provides you the option of programming the following elements of a step:

- Interlock
- Supervision
- Actions
- Transition

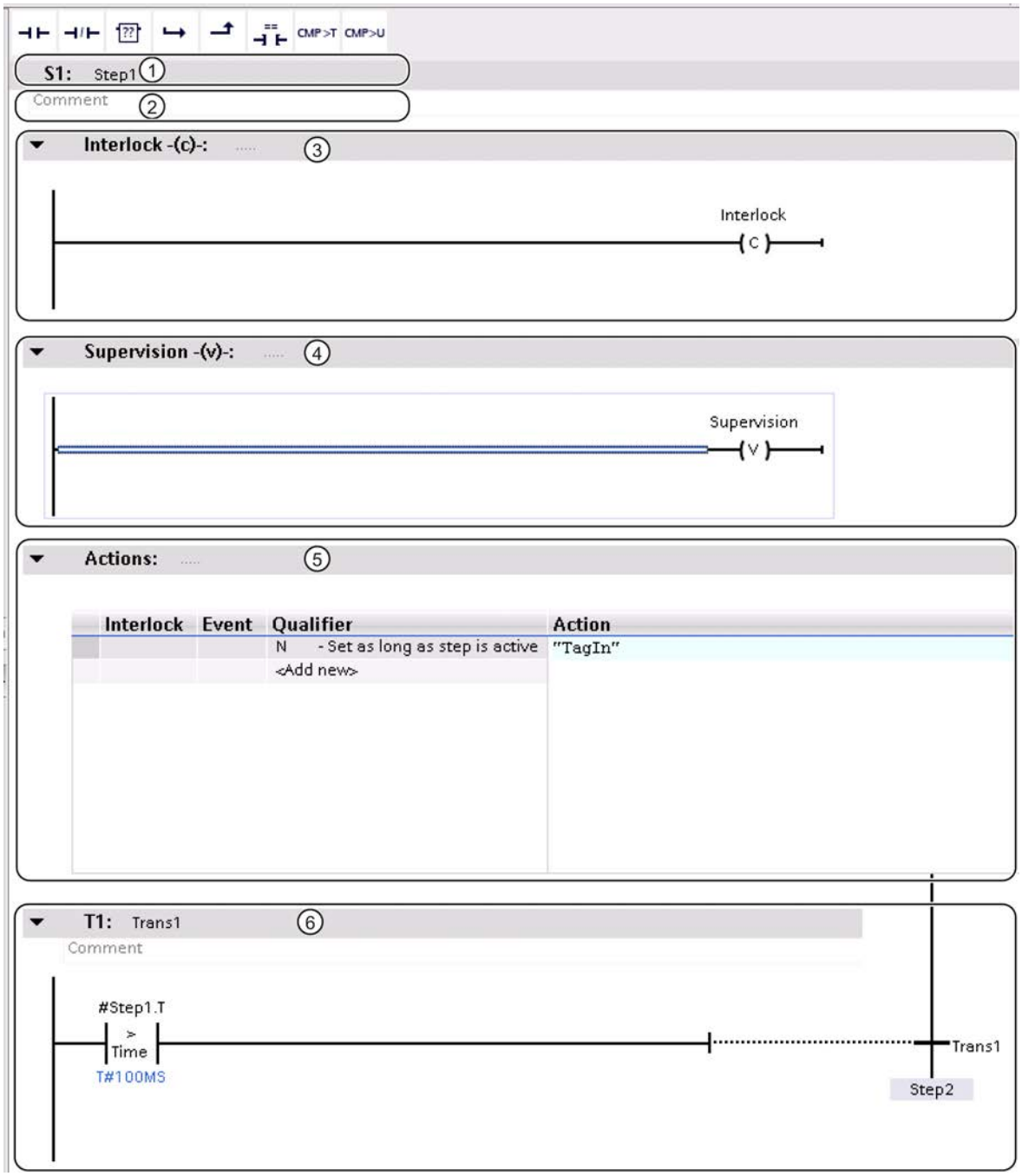
In addition, you can specify the title of the step and a comment for the step

Note

You can program the actions of the step and the associated transition additionally in the sequence view.

Structure of the single step view

The following figure shows a step in the single step view:



① Title of the step

② Comment for the step

③ Interlock

④ Supervision

⑤ Actions

⑥ Transition

See also

Step elements (Page 2725)

Overview of the programming window (Page 2762)

"Permanent pre-instructions" view (Page 2764)

Sequence view (Page 2765)

"Permanent post-instructions" view (Page 2769)

Alarm view (Page 2770)

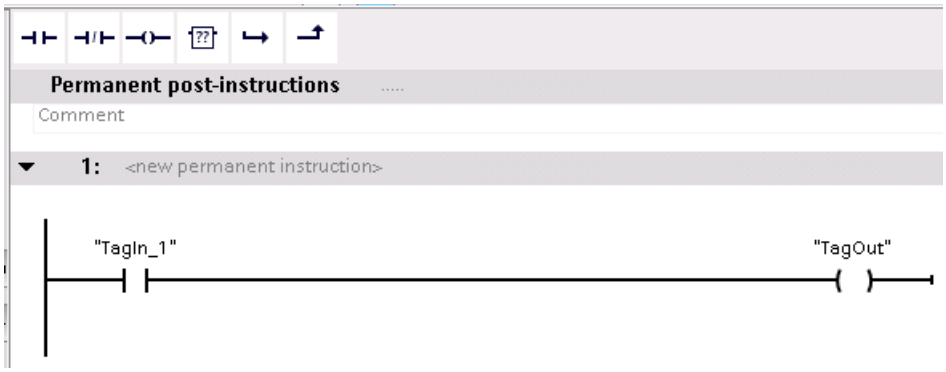
"Permanent post-instructions" view

Function of the "Permanent post-instructions" view

In the "Permanent post-instructions" view, you can program instructions that are executed after the sequencer is processed.

Structure of the "Permanent post-instructions" view

The following figure shows an example of the "Permanent post-instructions" view with the LAD programming language:



See also

Permanent instructions (Page 2739)

Overview of the programming window (Page 2762)

"Permanent pre-instructions" view (Page 2764)

Sequence view (Page 2765)

Single step view (Page 2767)

Alarm view (Page 2770)

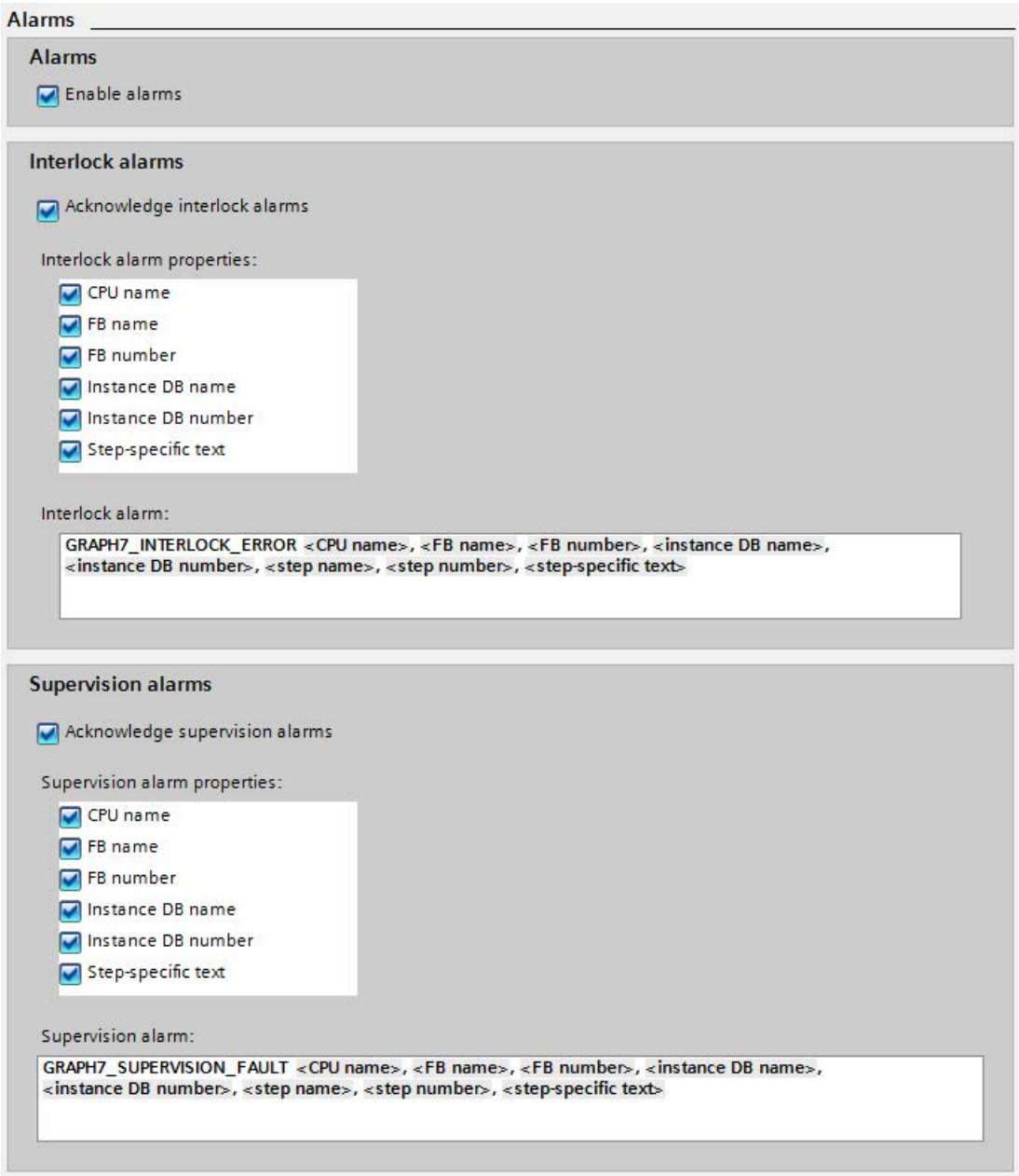
Alarm view

Function of the alarm view

In the alarm view, you can enable the alarm display and specify your own interlock and supervision alarms. You also have the option of specifying whether the alarms have to be acknowledged.

Structure of the alarm view

The following figure shows the alarm view containing the standard texts for interlock and supervision:



See also

Specifying interlock and supervision alarms (Page 2815)

Overview of the programming window (Page 2762)

"Permanent pre-instructions" view (Page 2764)

Sequence view (Page 2765)
Single step view (Page 2767)
"Permanent post-instructions" view (Page 2769)

Using Favorites in GRAPH

Adding GRAPH elements to Favorites

Requirement

- A GRAPH function block is open in one of the following views:
  - Permanent pre-instructions
  - Sequence view
  - Single step view
  - Permanent post-instructions
- The multipane mode is set for the "Instructions" task card or the Favorites are also displayed in the GRAPH programming window.

Procedure

In order to add instructions to Favorites, follow these steps:

1. Open the "Instructions" task card.
2. Maximize the "Basic instructions" pane.
3. Navigate in the "Basic instructions" pane to the instruction that you want to add to Favorites.
4. Drag the instruction to the "Favorites" pane or to the Favorites area in the GRAPH programming window.

Note

To additionally display the Favorites in the programming window, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Inserting GRAPH elements using Favorites (Page 2773)
Removing GRAPH elements from Favorites (Page 2773)

Inserting GRAPH elements using Favorites

Requirement

- A GRAPH function block is open in one of the following views:
  - Permanent pre-instructions
  - Sequence view
  - Single step view
  - Permanent post-instructions
- Favorites are available.

Procedure

To insert an instruction into a program using Favorites, follow these steps:

1. Drag the desired instruction from Favorites to the desired position.

Or:

1. Select the position in the program where you want to insert the instruction.
2. In Favorites, click the instruction to be inserted.

Note

To additionally display the Favorites in the programming window, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Adding GRAPH elements to Favorites (Page 2772)
Removing GRAPH elements from Favorites (Page 2773)

Removing GRAPH elements from Favorites

Requirement

- A GRAPH function block is open in one of the following views:
  - Permanent pre-instructions
  - Sequence view
  - Single step view
  - Permanent post-instructions
- Favorites are available.

Procedure

To remove instructions from Favorites, follow these steps:

1. Right-click the instruction in the Favorites area that you want to remove.
2. Select the "Remove instruction" command in the shortcut menu.

Note

To additionally display the Favorites in the programming window, click the "Display favorites in the editor" button in the program editor toolbar.

See also

Adding GRAPH elements to Favorites (Page 2772)

Inserting GRAPH elements using Favorites (Page 2773)

Programming the structure of a sequencer

Inserting the sequencers

When you open a GRAPH function block, the "Sequences" pane is expanded by default in the navigation view. A new GRAPH function block already contains a sequencer with one step and one transition. You can insert additional sequencers at any time.

Requirement

A GRAPH function block is open.

Procedure

To insert a new sequencer, follow these steps:

1. In the navigation view, right-click the title bar of the "Sequences" pane or the title bar of a sequencer that is already present.
2. Select the "Insert sequence" command in the shortcut menu.
   The sequencer is inserted after the selected sequencer. If no sequencer is selected, it is inserted before the first sequencer.

See also

Basics on sequencers (Page 2720)

Elements of a sequencer (Page 2721)

Renaming sequencers (Page 2775)

Adding comments for sequencers (Page 2775)

Inserting elements in a sequencer (Page 2776)
Ending sequencers (Page 2777)
Deleting sequencers (Page 2778)
Using steps and transitions (Page 2778)
Using branches (Page 2787)
Using jumps (Page 2790)

Renaming sequencers

Requirement

A GRAPH function block is open in the sequence view.

Procedure

To rename a sequencer, follow these steps:

1. Click the title bar of the sequencer in the sequence view.
   The "<new sequence>" placeholder is provided by default.
2. Enter the new name of the sequencer.

See also

Basics on sequencers (Page 2720)
Elements of a sequencer (Page 2721)
Inserting the sequencers (Page 2774)
Adding comments for sequencers (Page 2775)
Inserting elements in a sequencer (Page 2776)
Ending sequencers (Page 2777)
Deleting sequencers (Page 2778)
Using steps and transitions (Page 2778)
Using branches (Page 2787)
Using jumps (Page 2790)

Adding comments for sequencers

Requirement

A GRAPH function block is open in the sequence view.

Procedure

To enter a comment for a sequencer, follow these steps:

1. Click inside the comment area of the sequencer in the sequence view.
   A "Comment" placeholder is provided by default.
2. Enter a comment on the sequencer.

See also

Basics on sequencers (Page 2720)

Elements of a sequencer (Page 2721)

Inserting the sequencers (Page 2774)

Renaming sequencers (Page 2775)

Inserting elements in a sequencer (Page 2776)

Ending sequencers (Page 2777)

Deleting sequencers (Page 2778)

Using steps and transitions (Page 2778)

Using branches (Page 2787)

Using jumps (Page 2790)

Inserting elements in a sequencer

Requirement

A sequencer is open in the sequence view.

Procedure

To insert elements in a sequencer, follow these steps:

1. Right-click the point in the sequencer where you want to insert a new element.
2. Select the element you want to insert in the "Insert element" submenu of the shortcut menu.
   Only the elements that are permitted at the selected point in the sequencer are available for selection.

Or:

1. Open the "Basic instructions" pane of the "Instructions" task card.
2. Open the "Graph structure" folder.
3. Drag the element you want to insert to the desired point in the sequencer. Note here that not all elements can be inserted at all points in the sequencer. Allowed points in the sequencer are indicated by a small green square.

See also

Basics on sequencers (Page 2720)
Elements of a sequencer (Page 2721)
Inserting the sequencers (Page 2774)
Renaming sequencers (Page 2775)
Adding comments for sequencers (Page 2775)
Ending sequencers (Page 2777)
Deleting sequencers (Page 2778)
Using steps and transitions (Page 2778)
Using branches (Page 2787)
Using jumps (Page 2790)

Ending sequencers

Every sequencer must be ended with a jump to another step in the same sequencer or another sequencer or with the "Sequence end" element. A jump to another step enables cyclic processing of the sequencer.

Requirement

A sequencer is open in the sequence view.

Procedure

To end a sequence, follow these steps:

1. Right-click the double arrow at the end of the sequencer.
2. Select the "Sequence end" element in the "Insert element" submenu of the shortcut menu.

Or:

1. Right-click the double arrow at the end of the sequencer.
2. Select the "Jump" element in the "Insert element" submenu of the shortcut menu.
   An overview containing all available steps of the GRAPH function block opens.
3. Select a step as the jump destination.

See also

Basics on sequencers (Page 2720)
Elements of a sequencer (Page 2721)
Inserting the sequencers (Page 2774)
Renaming sequencers (Page 2775)
Adding comments for sequencers (Page 2775)

Inserting elements in a sequencer (Page 2776)
Deleting sequencers (Page 2778)
Using steps and transitions (Page 2778)
Using branches (Page 2787)
Using jumps (Page 2790)

Deleting sequencers

You can delete sequencers again at any time. However, the GRAPH function block must always contain at least one sequencer. Note that when you delete sequencers, this may result in the deletion of jump destinations.

Procedure

To delete a sequencer, follow these steps:

1. Right-click the title bar of the sequencer in the navigation view.
2. Select the "Delete" command in the shortcut menu.

See also

Basics on sequencers (Page 2720)
Elements of a sequencer (Page 2721)
Inserting the sequencers (Page 2774)
Renaming sequencers (Page 2775)
Adding comments for sequencers (Page 2775)
Inserting elements in a sequencer (Page 2776)
Ending sequencers (Page 2777)
Using steps and transitions (Page 2778)
Using branches (Page 2787)
Using jumps (Page 2790)

Using steps and transitions

Insert steps and transitions

When you insert a new GRAPH function block in your project, it already contains a sequencer with one step and one transition. You can insert additional steps and transitions into the sequencer.

Note the following rules when inserting steps and transitions:

- Steps and transitions must always alternate. This means that you cannot insert two steps or two transitions in a row.
- It follows from this that you can insert steps and transitions only at the end of a branch or at positions at which a step or transition is missing.
- A simultaneous branch must always start with a step.
- An alternative branch must always start with a transition.
- You can insert a step and a transition together as a pair. Because this method ensures that the step is followed by a transition, it is possible to do this at any position in the sequencer.

Requirement

A GRAPH function block is open in the sequence view.

Inserting a step

To insert a step, follow these steps:

1. Right-click a position in the sequencer where it is permitted to insert a step.
2. Select the "Insert element > Step" command from the shortcut menu.

Or:

1. Drag the "Step" element from Favorites or the "Instructions" task card to the appropriate position in the sequencer.

Inserting a transition

To insert a transition, follow these steps:

1. Right-click a position in the sequencer where it is permitted to insert a transition.
2. Select the "Insert element > Transition" command from the shortcut menu.

Or:

1. Drag the "Transition" element from Favorites or the "Instructions" task card to the appropriate position in the sequencer.

Inserting a step and transition together as a pair

To insert a step and a transition together as a pair, follow these steps:

1. Right-click the position in the sequencer where you want to insert the step and the transition.
2. Select the "Insert element > Step and transition" command from the shortcut menu.

Or:

1. Drag the "Step and transition" element from Favorites or the "Instructions" task card to the appropriate position in the sequencer.

See also

Steps and transitions (Page 2724)

Specifying and canceling the initial step (Page 2780)

Renumbering steps and transitions (Page 2781)

Renaming steps and transitions (Page 2783)

Adding comments for steps (Page 2784)

Copying and inserting steps and transitions (Page 2785)

Skip steps (Page 2786)

Deleting steps and transitions (Page 2786)

Specifying and canceling the initial step

When you insert a new GRAPH function block in your project, it already contains a sequencer with one step and one transition. By default, this step is specified as the initial step. This means that the sequencer is processed starting with this step. You can define a different step as the initial step at any time. In simultaneous branches, it is also permitted to have multiple initial steps. However, the maximum number of initial steps is 250. You can change an initial step to a normal step subsequently by canceling it.

Requirement

A GRAPH function block is open in the sequence view.

Specifying an initial step

To specify a step as an initial step, follow these steps:

1. Right-click the step you want to specify as an initial step.
2. Select the "Initial step" check box in the shortcut menu.

Canceling an initial step

To cancel an initial step and use it as a normal step again, follow these steps:

1. Right-click the step you want to specify as an initial step.
2. Clear the "Initial step" check box in the shortcut menu.

See also

Steps and transitions (Page 2724)

Insert steps and transitions (Page 2778)

Renumbering steps and transitions (Page 2781)
Renaming steps and transitions (Page 2783)
Adding comments for steps (Page 2784)
Copying and inserting steps and transitions (Page 2785)
Skip steps (Page 2786)
Deleting steps and transitions (Page 2786)

Renumbering steps and transitions

When you insert steps and transitions in your sequencer, the next available number is assigned. You can change this number subsequently. You have the following two options for this:

- You manually assign new numbers for the steps and transitions you want to renumber. Note that the same number must not be assigned more than once.
- You perform program-supported renumbering This method allows you to have all steps and transitions of a branch, sequencer, or block to be renumbered automatically. Observe the following differences when using this method:
  - Renumbering within a branch:
    Within the branch, all steps or transitions starting from the selected step or transition are renumbered. The renumbering ends at the start of a new branch.
  - Renumbering within a sequencer:
    Within the sequencer, all steps or transitions are renumbered. The first step or the first transition of the sequencer is assigned the start number you entered. Note that steps in other sequencers may also be renumbered in the process if duplicate numbers would otherwise result.
  - Renumbering within a block:
    Within the GRAPH block, all steps or transitions are renumbered. The first step or the first transition within the first sequencer of the block is assigned the start number you entered.

  You can also renumber only the steps or only the transitions.

Note

Note, however, that this option is not available for GRAPH function blocks for which you have selected the "Create minimized DB" attribute.

Requirement

A GRAPH function block is open in the sequence view.

Manual renumbering of steps and transitions

To manually renumber steps or transitions, follow these steps:

1. Select the step whose number you want to change.
2. Click the number of the step.
3. Enter the new number.

Program-supported renumbering of steps and transitions within a branch

For program-supported renumbering of steps or transitions within a branch, follow these steps:

1. Right-click the step or transition where you want to start renumbering the branch.
2. Select the "Renumber" command in the shortcut menu.
   The "Renumber" dialog opens.
3. If you want to renumber the steps, select the "Steps" check box and enter the number to be assigned to the selected step.
4. If you want to renumber the transitions, select the "Transitions" check box and enter the number to be assigned to the transition following the selected step.
5. Select the "Branch" option button in the "Renumber in" area.
6. Click "OK" to confirm.

Program-supported renumbering of steps and transitions within a sequencer

For program-supported renumbering of steps or transitions within a sequencer, follow these steps:

1. Right-click a step or transition within the sequencer for which you want to carry out renumbering.
2. Select the "Renumber" command in the shortcut menu.
   The "Renumber" dialog opens.
3. If you want to renumber the steps, select the "Steps" check box and enter the number to be assigned to the first step of the sequencer.
4. If you want to renumber the transitions, select the "Transitions" check box and enter the number to be assigned to the first transition of the sequencer.
5. Select the "Sequence" option button in the "Renumber in" area.
6. Click "OK" to confirm.

Program-supported renumbering of steps and transitions within a GRAPH function block

For program-supported renumbering of steps or transitions within a GRAPH function block, follow these steps:

1. Right-click a step or transition within the GRAPH function block for which you want to carry out renumbering.
2. Select the "Renumber" command in the shortcut menu.
   The "Renumber" dialog opens.

3. If you want to renumber the steps, select the "Steps" check box and enter the number to be assigned to the first step of the GRAPH function block.
4. If you want to renumber the transitions, select the "Transitions" check box and enter the number to be assigned to the first transition of the GRAPH function block.
5. Select the "Block" option button in the "Renumber in" area.
6. Click "OK" to confirm.

See also

Steps and transitions (Page 2724)

Insert steps and transitions (Page 2778)

Specifying and canceling the initial step (Page 2780)

Renaming steps and transitions (Page 2783)

Adding comments for steps (Page 2784)

Copying and inserting steps and transitions (Page 2785)

Skip steps (Page 2786)

Deleting steps and transitions (Page 2786)

Renaming steps and transitions

When you insert steps and transitions into your sequencer, they are initially named "Step<StepNumber>" and "Trans<StepNumber>". You can change these names in order to make your program easier to understand. Note that the same name must not be assigned more than once.

Requirement

A GRAPH function block is open in the sequence view.

Renaming a step

To rename a step, follow these steps:

1. Select the step whose name you want to change.
2. Click the step name.
3. Enter a new name.

Renaming a transition

To rename a transition, follow these steps:

1. Double-click the transition name you want to change.
2. Enter a new name.

See also

Steps and transitions (Page 2724)
Insert steps and transitions (Page 2778)
Specifying and canceling the initial step (Page 2780)
Renumbering steps and transitions (Page 2781)
Adding comments for steps (Page 2784)
Copying and inserting steps and transitions (Page 2785)
Skip steps (Page 2786)
Deleting steps and transitions (Page 2786)

Adding comments for steps

You have the option of adding comments for steps. For example, you can use comments to indicate the purpose of the step or to reference particular details.

Requirement

A step is open in the single step view.

Procedure

To add a comment for a step, follow these steps:

1. Click "Comment" in the comment area.
   The "Comment" text is deleted and an insertion point appears.
2. Enter the comment.

See also

Steps and transitions (Page 2724)
Insert steps and transitions (Page 2778)
Specifying and canceling the initial step (Page 2780)
Renumbering steps and transitions (Page 2781)
Renaming steps and transitions (Page 2783)
Copying and inserting steps and transitions (Page 2785)
Skip steps (Page 2786)
Deleting steps and transitions (Page 2786)

Copying and inserting steps and transitions

You can copy steps and transitions and insert them in the same sequencer or a different sequencer. This allows you to reduce the programming effort required in cases where a new step is to perform similar tasks as an existing step. Note the following particularities:

- When steps and transitions are copied, the alternating sequence of a step and a transition must also be retained. For this reason, placeholders are inserted ("Missing Step", "Missing Transition"), if two steps or two transitions in a row result after the insertion operation.
- A renaming and renumbering of the inserted element is carried out to prevent duplicate names and numbers.

Requirement

A GRAPH function block is open in the sequence view.

Copying steps and transitions

To copy a step or a transition, follow these steps:

1. To copy a step, right-click the step you want to copy. To copy a transition, right-click the transition you want to copy.
2. Select the "Copy" command in the shortcut menu.

Insert steps and transitions

To insert a step or transition, follow these steps:

1. Right-click the position in the sequencer where you want to insert the step or transition.
2. Select the "Insert" command in the shortcut menu.
3. If necessary, replace the placeholders with valid elements.

See also

Steps and transitions (Page 2724)

Insert steps and transitions (Page 2778)

Specifying and canceling the initial step (Page 2780)

Renumbering steps and transitions (Page 2781)

Renaming steps and transitions (Page 2783)

Adding comments for steps (Page 2784)

Skip steps (Page 2786)

Deleting steps and transitions (Page 2786)

Skip steps

You have the option of preventing the execution of steps for which the preceding and subsequent transitions are simultaneously valid. For CPUs of the S7-300/400 series, however, this option cannot be used in combination with the "Create minimized DB" option.

Procedure

To skip steps for which the preceding and subsequent transitions are simultaneously valid, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "PLC programming > GRAPH" group in the area navigation.
3. Select the "Skip steps" option under "Sequence properties".

See also

Steps and transitions (Page 2724)

Insert steps and transitions (Page 2778)

Specifying and canceling the initial step (Page 2780)

Renumbering steps and transitions (Page 2781)

Renaming steps and transitions (Page 2783)

Adding comments for steps (Page 2784)

Copying and inserting steps and transitions (Page 2785)

Deleting steps and transitions (Page 2786)

Deleting steps and transitions

You can remove steps and transitions from a sequencer again. Note the following particularities:

- When steps and transitions are deleted, the alternating sequence of a step and a transition must also be retained. For this reason, placeholders are inserted ("Missing Step", "Missing Transition"), if two steps or two transitions in a row occur after the deletion operation.
- You cannot delete all the steps of a sequencer since it must always contain at least one step.

Requirement

A GRAPH function block is open in the sequence view.

Procedure

To delete a step or a transition, follow these steps:

1. To delete a step, right-click the step you want to delete. To delete a transition, right-click the transition you want to delete.
2. Select the "Delete" command in the shortcut menu.
3. If necessary, replace the placeholders with valid elements.

See also

Steps and transitions (Page 2724)

Insert steps and transitions (Page 2778)

Specifying and canceling the initial step (Page 2780)

Renumbering steps and transitions (Page 2781)

Renaming steps and transitions (Page 2783)

Adding comments for steps (Page 2784)

Copying and inserting steps and transitions (Page 2785)

Skip steps (Page 2786)

Using branches

Creating alternative branches

Requirement

A sequencer is open in the sequence view.

Procedure

To create an alternative branch, follow these steps:

1. Right-click the step after which you want to insert an alternative branch.
2. Select the "Open alternative branch" command in the "Insert element" submenu of the shortcut menu.

Or:

1. Open the "Basic instructions" pane of the "Instructions" task card.
2. Open the "Graph structure" folder.
3. Drag the "Alternative branch" element to the desired position in the sequencer. Allowed positions are indicated by a green square.

See also

Elements of a sequencer (Page 2721)

Creating a simultaneous branch (Page 2788)

Closing branches (Page 2788)

Deleting branches (Page 2789)

Creating a simultaneous branch

Requirements

A sequencer is open in the sequence view.

Procedure

To create a simultaneous branch, follow these steps:

1. Right-click the transition after which you want to insert a simultaneous branch.
2. Select the "Open simultaneous branch" command in the "Insert element" submenu of the shortcut menu.

Or:

1. Open the "Basic instructions" pane of the "Instructions" task card.
2. Open the "Graph structure" folder.
3. Drag the "Simultaneous branch" element to the desired position in the sequencer. Allowed positions are indicated by a green square.

See also

Elements of a sequencer (Page 2721)

Creating alternative branches (Page 2787)

Closing branches (Page 2788)

Deleting branches (Page 2789)

Closing branches

You can close simultaneous and alternative branches as follows:

- You can join the branch to its parent branch.
- You can close the branch with a sequence end.
- You can close the branch with a jump.

In a simultaneous branch, however, you can only insert a sequence end or a jump after a transition.

Requirement

An alternative or simultaneous branch is present.

Joining a branch to its parent branch

To join an open branch to its parent branch, follow these steps:

1. Select the double arrow at the end of the branch.
2. Press and hold down the left mouse button.
   A dashed line will appear as soon as the cursor is moved.
3. Drag the dashed line to a suitable position in the network. Allowed positions are indicated by a green square.
4. Release the left mouse button.

Closing a branch with a sequence end

To close a branch with a sequence end, follow these steps:

1. If required, insert a transition at the end of the branch in a simultaneous branch.
2. Right-click the double arrow at the end of the branch.
3. Select the "Sequence end" element in the "Insert element" submenu of the shortcut menu.

Closing a branch with a jump

To close a branch with a jump, follow these steps:

1. If required, insert a transition at the end of the branch in a simultaneous branch.
2. Right-click the double arrow at the end of the branch.
3. Select the "Jump" element in the "Insert element" submenu of the shortcut menu.
   An overview containing all available steps of the GRAPH function block opens.
4. Select a step as the jump destination.

See also

Elements of a sequencer (Page 2721)

Creating alternative branches (Page 2787)

Creating a simultaneous branch (Page 2788)

Deleting branches (Page 2789)

Deleting branches

Requirement

An alternative or simultaneous branch is present.

Procedure

To delete a branch, follow these steps:

1. Select the branch you want to delete.
2. Right-click the selected branch.
3. Select the "Delete" command in the shortcut menu.

See also

Elements of a sequencer (Page 2721)

Creating alternative branches (Page 2787)

Creating a simultaneous branch (Page 2788)

Closing branches (Page 2788)

Using jumps

Inserting jumps

You can insert jumps only at the end of a sequencer or branch. In a simultaneous branch, the last element must be a transition for this.

Requirement

A sequencer is open in the sequence view.

Procedure

To insert a jump in a sequencer or branch, follow these steps:

1. If required, insert a transition at the end of the branch in a simultaneous branch.
2. Right-click the double arrow at the end of the sequencer or branch.
3. Select the "Jump" element in the "Insert element" submenu of the shortcut menu.
   An overview containing all available steps of the GRAPH function block opens.
4. Select a step as the jump destination.

Result

The jump is inserted. The jump and jump destination are represented in the sequencer as arrows, in which case the return transition is indicated for the jump destination and the destination step is indicated for the return.

See also

Elements of a sequencer (Page 2721)
Changing the jump destination (Page 2791)
Navigating to the jump destination (Page 2791)
Deleting jumps (Page 2792)

Changing the jump destination

You can change the destination of a jump subsequently.

Requirement

- A GRAPH function block is open in the sequence view.
- A jump is available.

Procedure

To change the jump destination subsequently, follow these steps:

1. Double-click the step number of the current jump destination.
   An overview containing all available steps of the GRAPH function block opens.
2. Select a step as the new jump destination.

See also

Elements of a sequencer (Page 2721)
Inserting jumps (Page 2790)
Navigating to the jump destination (Page 2791)
Deleting jumps (Page 2792)

Navigating to the jump destination

You can navigate from a jump to the step that you have specified as the jump destination.

Requirement

- A GRAPH function block is open in the sequence view.
- A jump is available.

Procedure

To navigate from a jump to the step specified as the destination, follow these steps:

1. Right-click on the jump.
2. Select the "Go to > Jump destination" command in the shortcut menu.

See also

Elements of a sequencer (Page 2721)
Inserting jumps (Page 2790)
Changing the jump destination (Page 2791)
Deleting jumps (Page 2792)

Deleting jumps

Requirement

- A GRAPH function block is open in the sequence view.
- A jump is available.

Procedure

To delete a jump, follow these steps:

1. Right-click on the jump you want to delete.
2. Select the "Delete" command in the shortcut menu.

See also

Elements of a sequencer (Page 2721)
Inserting jumps (Page 2790)
Changing the jump destination (Page 2791)
Navigating to the jump destination (Page 2791)

Programming actions

Inserting actions

Introduction

You can specify actions that will be implemented when the step is processed. In order to add comments for actions or adapt block calls, you can enable a multi-line input. Single-line input is the default setting. You can insert actions either in the single step view or the sequence view.

For a better overview of your programmed actions you can show descriptions for the events and identifiers.

In actions you can assign an operand with the value of another operand or mathematical expression or the result of an instruction. Use the following syntax for this:

- Assigning the value of an operand: A:=B
- Assigning the value of a mathematical expression: A:=B<Operator>C
  You can use the following operators: + (Addition), - (Subtraction), * (Multiplication), / (Division), % (Modulo), AND (AND logic operation), OR (OR logic operation), XOR (Exclusive OR logic operation)
- Assigning the result of an instruction: A:=<Instruction>(B)
  You obtain the available instructions from the "Instructions" task card.

Note that the data types of the operands must be compatible with one another.

Inserting actions in the single step view

To insert an action in the single step view, follow these steps:

1. Open the single step view.
2. If the actions are collapsed, click the small arrow in front of "Actions".
   The Right arrow becomes a Down arrow, and the table containing the actions is displayed.
3. If you want to enable multi-line input, right-click in the line of the new action and select the "Allow multi-line mode" command in the shortcut menu.
4. If you want to link the new action with an interlock, click in the cell of the "Interlock" column and select the "-(C)- - Interlock" entry in the drop-down list.
5. If you want to link the new action with an event, click in the cell of the "Event" column and select the appropriate event in the drop-down list.
6. Click in the cell of the "Identifier" column and select the identifier of the new action in the drop-down list.
7. In the "Action" column, specify the action that is to be performed. You have the following options:
   - You can replace the placeholders with the operands or values that you want to use for the action. You can also insert these using drag-and-drop operation or via autocompletion.
   - You can use instructions from the "Instructions" task card. To do so, drag-and-drop them from the task card into the "Action" column.
   - You can drag-and-drop blocks from the project tree into the "Action" column in order to call them.

Note

When data-type-dependent instructions are called, you must specify a data type. You can either manually enter the data type or select it from a drop-down list by clicking the small green rectangle that is shown below the existing data type.

Inserting actions in the sequence view

To insert an action in the sequence view, follow these steps:

1. Open the sequence view.
2. Expand the step in which you want to insert actions. To do so, click the button at the top right corner of the step or select the step and press <Enter>.
3. If you want to enable multi-line input, right-click in the line of the new action and select the "Allow multi-line mode" command in the shortcut menu.
4. If you want to link the new action with an interlock, click in the cell of the "Interlock" column and select the "-(C)- - Interlock" entry in the drop-down list.
5. If you want to link the new action with an event, click in the cell of the "Event" column and select the appropriate event in the drop-down list.
6. Click in the cell of the "Identifier" column and select the identifier of the new action in the drop-down list.
7. In the "Action" column, specify the action that is to be performed. You have the following options:
   - You can replace the placeholders with the operands or values that you want to use for the action. You can also insert these using drag-and-drop operation or via autocompletion.
   - You can use instructions from the "Instructions" task card. To do so, drag-and-drop them from the task card into the "Action" column.
   - You can drag-and-drop blocks from the project tree into the "Action" column in order to call them.

Note

When data-type-dependent instructions are called, you must specify a data type. You can either manually enter the data type or select it from a drop-down list by clicking the small green rectangle that is shown below the existing data type.

Showing the event descriptions

To show the description of the events, follow these steps:

1. Right-click in the action table.
2. Select the "Show event descriptions" command in the shortcut menu.

Showing the identifier descriptions

To show the description of the identifiers, follow these steps:

1. Right-click in the action table.
2. Select the "Show qualifier descriptions" command in the shortcut menu.

See also

Actions (Page 2728)
Changing actions (Page 2797)
Copying, cutting, and inserting actions (Page 2798)
Moving actions (Page 2799)
Deleting actions (Page 2799)

Defining the data type of a GRAPH instruction

Basic information on the data types of GRAPH instructions

Introduction

GRAPH instructions can be used by means of the following two mechanisms:

- As block call
  For instructions that you insert in your program as block calls, you can select the data type via a drop-down list.
- As GRAPH-internal function
  You assign the GRAPH-internal functions directly to a tag. The operand must have a data type which supports this function. The following table provides an overview of the permitted data types of GRAPH-internal functions.

GRAPH-internal functions

The following table shows the GRAPH-internal functions and their data types:

Function	S7-300/400	S7-1500
BCD_TO_INT	WORD	WORD
BCD_TO_DINT	DWORD	DWORD
INT_TO_BCD	INT	INT
DINT_TO_BCD	DINT	DINT
INT_TO_DINT	INT	INT
DINT_TO_REAL	DINT	DINT
ROUND	REAL	REAL, LREAL
TRUNC	REAL	REAL, LREAL
CEIL	REAL	REAL, LREAL
FLOOR	REAL	REAL, LREAL
NEG	INT, DINT, REAL	SINT, INT, DINT, REAL, LINT, LREAL
ABS	REAL	SINT, INT, DINT, LINT, REAL, LREAL
SQR	REAL	REAL, LREAL
SQRT	REAL	REAL, LREAL
LN	REAL	REAL, LREAL

Function	S7-300/400	S7-1500
EXP	REAL	REAL, LREAL
SIN	REAL	REAL, LREAL
COS	REAL	REAL, LREAL
TAN	REAL	REAL, LREAL
ASIN	REAL	REAL, LREAL
ACOS	REAL	REAL, LREAL
ATAN	REAL	REAL, LREAL
NOT	WORD, DWORD	USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, BYTE, WORD, DWORD, LWORD
RLDA	DWORD	-
RRDA	DWORD	-
SWAP	WORD, DWORD	WORD, DWORD, LWORD
AND	BYTE, WORD, DWORD	BYTE, WORD, DWORD, LWORD
OR	BYTE, WORD, DWORD	BYTE, WORD, DWORD, LWORD
XOR	BYTE, WORD, DWORD	BYTE, WORD, DWORD, LWORD
SHL	WORD, DWORD	BYTE, USINT, SINT, WORD, UINT, INT, DWORD, UDINT, DINT, LWORD, ULINT, LINT
SHR	WORD, DWORD	BYTE, WORD, DWORD, USINT, UINT, UDINT, SINT, INT, DINT, LWORD, ULINT, LINT
ROL	DWORD	BYTE, WORD, DWORD, LWORD
ROR	DWORD	BYTE, WORD, DWORD, LWORD
MOD	DINT	USINT, UINT, UDINT, SINT, INT, DINT, ULINT, LINT
-: Function not available		

Changing the data type of a GRAPH instruction

GRAPH instructions that you insert in your program via a block call are inserted with a preset data type. If necessary, you can change this data type. Data types which can be changed are indicated by a green rectangle under the first letter of the data type. You can change the operand for GRAPH-internal functions.

Changing the data type of a called instruction

To change the data type of a called instruction, follow these steps:

1. Click on the data type of the instruction.
   The drop-down list for data type selection opens.
2. Click on the data type that you wish to use.

Changing the data type of a GRAPH-internal instruction

To change the data type of a GRAPH-internal function, follow these steps:

1. Enter a value or a tag as the operand. Observe the permitted range of values for the operand.

Or:

1. Holding the right mouse button down, drag a tag with the permitted data type over the operand.
2. Wait until the current operand has a green border.
3. Release the mouse button.

Changing actions

You can make the following changes for actions:

- Linking an action with an interlock or canceling the link with an interlock
- Linking an action with an event or canceling or changing the link with an event.
- Changing the action identifier
- Changing the action, e.g., replacing operands or using a different instruction

Note that certain action identifiers require the use of events and that the link to an event cannot be canceled for these identifiers.

Requirement

A GRAPH function block is open in the single step view.

Procedure

To change an action, follow these steps:

1. If you want to link an action with an interlock, click in the cell of the "Interlock" column and select the "-(C)- - Interlock" entry in the drop-down list.
2. If you want to cancel the link with an interlock, click in the cell of the "Interlock" column and select the "No condition" entry in the drop-down list.
3. If you want to link the action with an event or change the linked event, click in the cell of the "Event" column and select the appropriate event in the drop-down list.
4. If you want to cancel the link with an event, click in the cell of the "Event" column and select the "No event" entry in the drop-down list. If the action identifier requires an event, the background of the cell turns red and "<???>" is displayed. In this case, you must assign an event again or choose a different action identifier.

5. If you want to change the action identifier, click in the cell of the "Qualifier" column and select the new action identifier in the drop-down list.
6. If you want to change the action, click in the action in the "Action" column and make one of the following changes:
   - Change the operand(s) of the action.
   - To change the data type of an operand, click the small green rectangle and select the new data type in the drop-down list.
   - To use a different instruction, delete the existing action and drag a different instruction from the "Instructions" task card into the "Action" column. Enter the operands.
   - To call a different block, delete the existing action and drag-and-drop a block from the project tree into the "Action" column in order to call it.

See also

Actions (Page 2728)

Inserting actions (Page 2792)

Copying, cutting, and inserting actions (Page 2798)

Moving actions (Page 2799)

Deleting actions (Page 2799)

Copying, cutting, and inserting actions

Requirement

A GRAPH function block is open in the single step view.

Copying actions

To copy an action, follow these steps:

1. Right-click in the first cell of the action you want to copy.
2. Select the "Copy" command in the shortcut menu.

Cutting actions

To cut an action, follow these steps:

1. Right-click in the first cell of the action you want to cut.
2. Select the "Cut" command in the shortcut menu.

Pasting actions from the clipboard

To insert an action from the clipboard, follow these steps:

1. Right-click the position at which you want to insert the action.
2. Select the "Insert" command in the shortcut menu.

See also

Actions (Page 2728)
Inserting actions (Page 2792)
Changing actions (Page 2797)
Moving actions (Page 2799)
Deleting actions (Page 2799)

Moving actions

You can move actions within the action table.

Requirement

A GRAPH function block is open in the single step view.

Procedure

To move an action within the action table, follow these steps:

1. Click in the first cell of the action you want to move and keep the mouse button pressed. This selects the action.
2. Drag the action to the desired position.

See also

Actions (Page 2728)
Inserting actions (Page 2792)
Changing actions (Page 2797)
Copying, cutting, and inserting actions (Page 2798)
Deleting actions (Page 2799)

Deleting actions

Requirement

A GRAPH function block is open in the single step view.

Procedure

To delete an action, follow these steps:

1. Right-click in the first cell of the action you want to delete.
2. Select the "Delete" command in the shortcut menu.

See also

Actions (Page 2728)

Inserting actions (Page 2792)

Changing actions (Page 2797)

Copying, cutting, and inserting actions (Page 2798)

Moving actions (Page 2799)

Programming conditions

Programming interlocks

You can use up to 32 instructions in your interlock network to which a maximum of 32 operands are interconnected. In LAD, the instructions can be inserted both in a rung and in the power rail.

Requirement

A GRAPH function block is open in the single step view.

Procedure

To program interlocks, follow these steps:

1. Open the interlock network by clicking the small arrow in front of "Interlock -(c)-".
2. Open the "Instructions" task card.
3. Navigate to the instruction you want to insert.
4. Drag the instruction to the desired position in the network.
5. In the "Interlock" column of the action table, click in the cell of the action you want to link with the interlock and select the "-(C)- - Interlock" entry in the drop-down list.

Or:

1. Open the interlock network by clicking the small arrow in front of "Interlock -(c)-".
2. Open the "Instructions" task card.
3. Navigate to "LAD/FBD > General > Empty box" in the "Basic instructions" pane.
4. Drag the "Empty box" element to the desired position in the network.

5. Click on "??".
   A text box opens.
6. Either click the autocompletion button or use the shortcut <Ctrl+I>.
   Autocompletion opens. It contains only the permitted elements for the particular context.
7. Select the desired element from the drop-down list.
8. In the "Interlock" column of the action table, click in the cell of the action you want to link with the interlock and select the "-(C)- - Interlock" entry in the drop-down list.

See also

Conditions (Page 2739)

Programming supervisions (Page 2801)

Programming transitions (Page 2802)

Programming permanent instructions (Page 2803)

Displaying or hiding tag information (Page 2810)

Specifying the programming language in networks (Page 2758)

Programming supervisions

You can use up to 32 instructions in your supervision network to which a maximum of 32 operands are interconnected. In LAD, the instructions can be inserted both in a rung and in the power rail.

Requirement

A GRAPH function block is open in the single step view.

Procedure

To program supervisions, follow these steps:

1. Open the supervision network by clicking the small arrow in front of "Supervision -(v)-".
2. Open the "Instructions" task card.
3. Navigate to the instruction you want to insert.
4. Drag the instruction to the desired position in the network.

Or:

1. Open the supervision network by clicking the small arrow in front of "Supervision -(v)-".
2. Open the "Instructions" task card.
3. Navigate to "LAD/FBD > General > Empty box" in the "Basic instructions" pane.
4. Drag the "Empty box" element to the desired position in the network.
5. Click on "??".
   A text box opens.

6. Either click the autocompletion button or use the shortcut <Ctrl+I>.
   Autocompletion opens. It contains only the permitted elements for the particular context.
7. Select the desired element from the drop-down list.

See also

Conditions (Page 2739)

Programming interlocks (Page 2800)

Programming transitions (Page 2802)

Programming permanent instructions (Page 2803)

Displaying or hiding tag information (Page 2810)

Specifying the programming language in networks (Page 2758)

Programming transitions

You can use up to 32 instructions in your transition network to which a maximum of 32 operands are interconnected. In LAD, the instructions can be inserted both in a rung and in the power rail. You can program transitions in either the single step view or the sequence view.

Programming transitions in the single step view

To program transitions in the single step view, follow these steps:

1. Open the single step view.
2. Open the transition network by clicking the small arrow in front of "<TransitionNumber> - <TransitionName>".
3. Open the "Instructions" task card.
4. Navigate to the instruction you want to insert.
5. Drag the instruction to the desired position in the network.

Or:

1. Open the transition network by clicking the small arrow in front of "<TransitionNumber> - <TransitionName>".
2. Open the "Instructions" task card.
3. Navigate to "LAD/FBD > General > Empty box" in the "Basic instructions" pane.
4. Drag the "Empty box" element to the desired position in the network.
5. Click on "??".
   A text box opens.
6. Either click the autocompletion button or use the shortcut <Ctrl+I>.
   Autocompletion opens. It contains only the permitted elements for the particular context.
7. Select the desired element from the drop-down list.

Programming transitions in the sequence view

To program transitions in the sequence view, follow these steps:

1. Open the sequence view.
2. Expand the transition you want to program. To do so, click the button next to the transition or select the transition and press <Enter>.
3. Open the "Instructions" task card.
4. Navigate to the instruction you want to insert.
5. Drag the instruction to the desired position in the network.

Or:

1. Open the "Instructions" task card.
2. Navigate to "LAD/FBD > General > Empty box" in the "Basic instructions" pane.
3. Drag the "Empty box" element to the desired position in the network.
4. Click on "??".
   A text box opens.
5. Either click the autocompletion button or use the shortcut <Ctrl+I>.
   Autocompletion opens. It contains only the permitted elements for the particular context.
6. Select the desired element from the drop-down list.

See also

Conditions (Page 2739)

Programming interlocks (Page 2800)

Programming supervisions (Page 2801)

Programming permanent instructions (Page 2803)

Displaying or hiding tag information (Page 2810)

Specifying the programming language in networks (Page 2758)

Programming permanent instructions

Working with networks

Using networks

Function

Permanent instructions are programmed inside networks in a GRAPH function block. At least one network must be available in order to program a permanent instruction. To achieve a better overview of the program, you can also subdivide your program into several networks.

See also

Inserting networks (Page 2804)
Selecting networks (Page 2804)
Copying and pasting networks (Page 2805)
Deleting networks (Page 2806)
Expanding and collapsing networks (Page 2806)
Inserting network title (Page 2807)
Navigating networks (Page 2809)

Inserting networks

Requirement

A block is open.

Procedure

To insert a new network, follow these steps:

1. Select the network after which you want to insert a new network.
2. Select the "Insert network" command in the shortcut menu.

Result

A new empty network is inserted into the block.

See also

Using networks (Page 2803)
Selecting networks (Page 2804)
Copying and pasting networks (Page 2805)
Deleting networks (Page 2806)
Expanding and collapsing networks (Page 2806)
Inserting network title (Page 2807)
Navigating networks (Page 2809)

Selecting networks

Requirements

A network is available.

Selecting a network

To select a network, follow these steps:

1. Click the title bar of the network that you want to select.

Selecting several networks

Proceed as follows to select several individual networks:

1. Press and hold down the <Ctrl> key.
2. Click all the networks that you want to select.

To select several successive networks, follow these steps:

1. Press and hold down the <Shift> key.
2. Click the first network that you want to select.
3. Click the last network that you want to select.
   The first and last networks and all those in between are selected.

See also

Using networks (Page 2803)
Inserting networks (Page 2804)
Copying and pasting networks (Page 2805)
Deleting networks (Page 2806)
Expanding and collapsing networks (Page 2806)
Inserting network title (Page 2807)
Navigating networks (Page 2809)

Copying and pasting networks

Copied networks can be pasted within the block or in another block. Networks that were created in LAD or FBD can also be inserted in blocks of the respective other programming language.

Requirement

A network is available.

Procedure

To copy and paste a network, follow these steps:

1. Select the network or networks to be copied.
2. Select "Copy" in the shortcut menu.
3. Select the network after which you want to paste in the copied network.
4. Select "Paste" in the shortcut menu.

See also

Using networks (Page 2803)
Inserting networks (Page 2804)
Selecting networks (Page 2804)
Deleting networks (Page 2806)
Expanding and collapsing networks (Page 2806)
Inserting network title (Page 2807)
Navigating networks (Page 2809)

Deleting networks

Requirement

A network is available.

Procedure

To delete a network, follow these steps:

1. Select the network that you want to delete.
2. Select the "Delete" command in the shortcut menu.

See also

Using networks (Page 2803)
Inserting networks (Page 2804)
Selecting networks (Page 2804)
Copying and pasting networks (Page 2805)
Expanding and collapsing networks (Page 2806)
Inserting network title (Page 2807)
Navigating networks (Page 2809)

Expanding and collapsing networks

Requirements

A network is available.

Opening and closing a network

To open a network, follow these steps:

1. Click on the right arrow in the network title bar.

To close a network, follow these steps:

1. Click on the down arrow in the network title bar.

Opening and closing all networks

To open and close all networks, follow these steps:

1. In the toolbar, click "Open all networks" or "Close all networks".

See also

Using networks (Page 2803)

Inserting networks (Page 2804)

Selecting networks (Page 2804)

Copying and pasting networks (Page 2805)

Deleting networks (Page 2806)

Inserting network title (Page 2807)

Navigating networks (Page 2809)

Inserting network title

The network title is the header of a network. The length of the network title is limited to one line. You can enter the title manually or set it automatically. When you set it automatically, you can do this for individual networks or use the settings to specify that the network title is always set automatically.

For automatic insertion of the network title, the comment of the operand in one of the following instructions in the network is evaluated:

- Assignment
- Set output
- Reset output

The instruction that is listed first in the network is used.

The network title is only inserted automatically when the following conditions are fulfilled:

- The network does not have a title yet.
- The operand of the instruction used for the comment has a comment.

Note

Note the following restrictions for automatic insertion of the network title:

- The network title is not adapted if you change the comment of the operand at a later time.
- The network title is not adapted if you change the operand of the instruction.
- The network title is only set by the writing instructions listed above.
- If the operand is of the data type array, the comment of the array is used and not the comments of the array elements.
- Comments of invalid operands are not taken into consideration.

Entering the network title manually

To enter a network title, follow these steps:

1. Click on the title bar of the network.
2. Enter the network title.

Setting the network title automatically

To specify that the network titles are always set automatically, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. Select the "Set network title automatically" check box in the "Additional settings" group.
   The network titles are set automatically as of this time if the conditions listed above are fulfilled.

To set an individual network title automatically, follow these steps:

1. Right-click "Network <Number of the network>" in the title bar of a network.
2. Select the "Set network title automatically" command in the shortcut menu.
   The title of the selected network is set based on the comment of the operand if the conditions listed above are fulfilled.

See also

Using networks (Page 2803)

Inserting networks (Page 2804)

Selecting networks (Page 2804)

Copying and pasting networks (Page 2805)

Deleting networks (Page 2806)

Expanding and collapsing networks (Page 2806)

Navigating networks (Page 2809)

Navigating networks

You can navigate straight to a specific position within a block.

Procedure

To navigate to a specific position within a block, follow these steps:

1. Right-click in the code area of the programming window.
2. Select the "Go to > Network/line" command in the shortcut menu.
   The "Go to" dialog will open.
3. Enter the network to which you want to navigate.
4. Enter the line number of the network to which you want to navigate.
5. Confirm your entry with "OK".

Result

The relevant line will be displayed if this is possible. If the network or line requested does not exist, the last existing network or the last existing line in the network requested will be displayed.

See also

Using networks (Page 2803)

Inserting networks (Page 2804)

Selecting networks (Page 2804)

Copying and pasting networks (Page 2805)

Deleting networks (Page 2806)

Expanding and collapsing networks (Page 2806)

Inserting network title (Page 2807)

Inserting permanent instructions

When you expand a pane for permanent pre-instructions or post-instructions, it always contains at least one empty network by default. You can use up to 32 instructions per network to which a maximum of 32 operands are interconnected. In LAD, the instructions can be inserted both in a rung and in the power rail. You can use up to 250 permanent pre-instructions and 250 permanent post-instructions within a GRAPH function block.

Requirement

A GRAPH function block is open in the sequence view.

Procedure

To program permanent instructions, follow these steps:

1. In the navigation view, open the "Permanent pre-instructions" or "Permanent post-instructions" pane.
2. Click on a network.
   The view for permanent pre-instructions or post-instructions opens in the work area.
3. Open the "Instructions" task card.
4. Navigate to the instruction you want to insert.
5. Drag the instruction to the desired position in the network.

Or:

1. Open the "Instructions" task card.
2. Navigate to "General > Empty box" in the "Basic instructions" pane.
3. Drag the "Empty box" element to the desired position in the network.
4. Click on "??".
   A text box opens.
5. Either click the autocompletion button or use the shortcut <Ctrl+I>.
   Autocompletion opens. It contains only the permitted elements for the particular context.
6. Select the desired element from the drop-down list.

See also

Conditions (Page 2739)

Programming interlocks (Page 2800)

Programming supervisions (Page 2801)

Programming transitions (Page 2802)

Displaying or hiding tag information (Page 2810)

Specifying the programming language in networks (Page 2758)

Displaying or hiding tag information

Introduction

For permanent instructions, interlocks, supervisions, and transitions, you can display the following information on the utilized tags in the programming window:

- Name of the tags
- Address of the tags
- Simple or hierarchical comments to document the tags

The information is taken from the block interface for local tags and DB tags and from the PLC tag table for tags that are valid CPU-wide.

You can display the tag information either for all the blocks or for individually opened blocks. If you display the tag information for all the blocks, the tag information for all blocks currently opened and opened in future is shown.

You can hide the tag information at any time again. If you have hidden the tag information for all blocks, you can display it again for individual ones that you have opened.

If you select the display of tag information with hierarchical comments, the comments of the higher structure levels of structured tags will also be displayed. The display is in brackets after the comment of the tags; the comments of the individual levels are separated by a period. If there is no comment at a structure level for a tag, it is omitted in the display and this is recognizable because there are two periods.

Displaying or hiding tag information for all blocks

Follow the steps below to display or hide the tag information for all blocks:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. In the area navigation, select the "PLC programming" group.
3. If you want to show the tag information, either select the "Expand" option in the "Tag information" drop-down list or the "Tag information with hierarchy" depending on whether you want to display simple or hierarchical comments.
4. If you want to hide the tag information, select the "Collapse" option in the "Tag information" drop-down list.
   The tag information is displayed or hidden for all blocks. When you open further blocks, the tag information is displayed or hidden depending on the selected setting.

Displaying or hiding tag information for an opened block

Follow the steps below to display or hide the tag information for an opened block:

1. If you want to show the tag information, either select the "Show tag information" option in the "Shows the tag information" drop-down list or the "Tag information with hierarchy" depending on whether you want to display simple or hierarchical comments.
2. If you want to hide the tag information, select the "Hide tag information" option in the "Hides tag information" drop-down list.
   The tag information is displayed or hidden.

See also

Programming interlocks (Page 2800)

Programming supervisions (Page 2801)

Programming transitions (Page 2802)

Programming permanent instructions (Page 2803)

Inserting block calls in GRAPH

Inserting block calls manually

You can call other blocks in actions and permanent instructions.

Inserting block calls manually in an action

To manually insert a block call in an action, follow these steps:

1. Open the GRAPH function block in the single step view.
2. Click in the cell of the action in the "Action" column.
3. Enter "CALL <Name of block to be called>" and confirm your entry with the Enter key.
4. Replace "<Name of block to be called>" with the name of the block you want to call.
   If the block to be called is a function block, the "Call options" dialog opens. In this case, perform steps 5 and 6.
5. In the dialog, enter whether you want to call the block as a single instance or multi-instance.
   - If you click on the "Single instance" button, you will have to enter a name in the "Name" text box for the data block that you want to assign to the function block. Select the type of numbering and enter the number for the data block, if necessary.
   - If you click on the "Multi-instance" button, you will have to enter the name of the tag in the "Name in the interface" text box; this is the name that you use to enter the called function block as a static tag in the interface of the calling block.
6. Confirm your entry with "OK".

Inserting a block call manually in a permanent instruction

To manually insert a block call in a permanent instruction, follow these steps:

1. Open the view for "Permanent pre-instructions" or "Permanent post-instructions".
2. Right-click the position in the network where you want to insert the block call.
3. Select the "Insert empty box" command in the shortcut menu.
4. Click on "??".
   A text box opens.
5. Enter the name of the block you want to call or open the autocompletion and select the block.
   If the block to be called is a function block, the "Call options" dialog opens. In this case, perform steps 6 and 7.

6. In the dialog, enter whether you want to call the block as a single instance or multi-instance.
   - If you click on the "Single instance" button, you will have to enter a name in the "Name" text box for the data block that you want to assign to the function block. Select the type of numbering and enter the number for the data block, if necessary.
   - If you click on the "Multi-instance" button, you will have to enter the name of the tag in the "Name in the interface" text box; this is the name that you use to enter the called function block as a static tag in the interface of the calling block.
7. Confirm your entry with "OK".

See also

Block calls (Page 2735)

Inserting block calls using drag-and-drop operation (Page 2813)

Updating block calls in GRAPH (Page 2814)

Inserting block calls using drag-and-drop operation

You can call other blocks in actions and permanent instructions. Note however, that you can call function blocks only as a single instance.

Inserting block calls in an action using drag-and-drop operation

To insert a block call in an action using drag-and-drop operation, follow these steps:

1. Open the GRAPH function block in the single step view.
2. Drag the block you want to call from the project tree to the cell of the action in the "Action" column.
   If the block to be called is a function block, the "Call options" dialog opens. In this case, perform steps 3 and 4.
3. In the dialog, enter whether you want to call the block as a single instance or multi-instance.
   - If you click on the "Single instance" button, you will have to enter a name in the "Name" text box for the data block that you want to assign to the function block. Select the type of numbering and enter the number for the data block, if necessary.
   - If you click on the "Multi-instance" button, you will have to enter the name of the tag in the "Name in the interface" text box; this is the name that you use to enter the called function block as a static tag in the interface of the calling block.
4. Confirm your entry with "OK".

Inserting block calls in a permanent instruction using drag-and-drop operation

To insert a block call in a permanent instruction using drag-and-drop operation, follow these steps:

1. Open the view for "Permanent pre-instructions" or "Permanent post-instructions".
2. Drag the block you want to call from the project tree to the position of the network of the permanent instruction where you want to insert the block call.
   If the block to be called is a function block, the "Call options" dialog opens. In this case, perform steps 3 and 4.
3. In the dialog, enter whether you want to call the block as a single instance or multi-instance.
   – If you click on the "Single instance" button, you will have to enter a name in the "Name" text box for the data block that you want to assign to the function block. Select the type of numbering and enter the number for the data block, if necessary.
   – If you click on the "Multi-instance" button, you will have to enter the name of the tag in the "Name in the interface" text box; this is the name that you use to enter the called function block as a static tag in the interface of the calling block.
4. Confirm your entry with "OK".

See also

Block calls (Page 2735)

Inserting block calls manually (Page 2812)

Updating block calls in GRAPH (Page 2814)

Updating block calls in GRAPH

If interface parameters of a called block are changed, the block call can no longer be executed correctly. You can avoid such inconsistent block calls by updating the block calls.

You have two options for updating block calls:

- Explicit updating in the program editor.
  The block calls within the open block will be updated.
- Implicit updating during compilation.
  All block calls in the program as well as the used PLC data types will be updated.

Updating block calls in the program editor

To update a block call within a block, follow these steps:

1. Open the block in the program editor.
2. Click the "Update inconsistent block calls" button in the toolbar.

Updating block calls during compilation

To implicitly update all block calls and uses of PLC data types during compilation, follow these steps:

1. Open the project tree.
2. Select the "Program blocks" folder.
3. Select the "Compile > Software (rebuild all blocks)" command in the shortcut menu.

See also

Block calls (Page 2735)

Inserting block calls manually (Page 2812)

Inserting block calls using drag-and-drop operation (Page 2813)

Specifying interlock and supervision alarms

Enabling and disabling the alarm view

You can enable or disable the alarm view globally for new GRAPH blocks or locally for individual blocks. The local setting of a block has priority over the global setting.

Enabling or disabling the alarm view globally

To globally enable or disable the alarm view for new GRAPH function blocks, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "PLC programming > GRAPH" group in the area navigation.
3. If you want to enable the alarm view, select the "Enable alarms" check box in the "Default settings for new blocks > Alarm properties > Alarms" area. If you want to disable the alarm view, clear the "Enable alarms" check box.

Enabling or disabling the alarm view locally

To enable or disable the alarm view for an existing GRAPH function block, follow these steps:

1. Open the alarm view for the GRAPH function block.
2. If you want to enable the alarm view, select the "Enable alarms" check box in the "Alarms" area. If you want to disable the alarm view, clear the "Enable alarms" check box.

Note

For GRAPH function blocks in S7-300/400 for which you are using the "Create minimized DB" option, you cannot change the alarm view.

See also

Interlock and supervision alarms (Page 2740)

Editing alarms (Page 2816)

Specifying mandatory acknowledgment (Page 2817)

Editing alarms

You can edit the interlock and supervision alarms as follows:

- Globally for all new GRAPH function blocks
- Locally for existing GRAPH function blocks
- For single steps

Editing interlock and supervision alarms globally for all new GRAPH function blocks

To globally edit the interlock and supervision alarms for new GRAPH function blocks, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "PLC programming > GRAPH" group in the area navigation.
3. If you want to edit the interlock alarms, enter the desired text in the "Alarm text" text box in the "Interlock alarms" area.
4. If you want to edit the supervision alarms, enter the desired text in the "Alarm text" text box in the "Supervision alarms" area.

Editing interlock and supervision alarms locally for an existing GRAPH function block

To locally edit the interlock and supervision alarms for an existing GRAPH function block, follow these steps:

1. Open the alarm view for the GRAPH function block.
2. In the "Interlock alarms" and "Supervision alarms" areas, select the alarm blocks that are to be displayed in the alarm.
3. If you want to edit the alarm text, enter the desired text in front of the alarm blocks. If necessary, replace the standard text or the globally specified text.

Note

For GRAPH function blocks in S7-300/400 for which you are using the "Create minimized DB" option, you cannot edit the alarm texts.

Editing the interlock alarm of a single step

To edit the interlock alarm of a single step, follow these steps:

1. Open the step in the single step view.
2. Select the "-(C)-" interlock.
3. Open the Inspector window and switch to the "Properties" tab, if necessary.
4. Enter the desired alarm text in the "Alarm text" text box.
   This text is appended at the end of the interlock alarm by the "<Step-specific text>" alarm block.

Editing the supervision alarm of a single step

To edit the supervision alarm of a single step, follow these steps:

1. Open the step in the single step view.
2. Select the "-(v)-" supervision.
3. Open the Inspector window and switch to the "Properties" tab, if necessary.
4. Enter the desired alarm text in the "Alarm text" text box.
   This text is appended at the end of the supervision alarm by the "<Step-specific text>" alarm block.

See also

Interlock and supervision alarms (Page 2740)

Enabling and disabling the alarm view (Page 2815)

Specifying mandatory acknowledgment (Page 2817)

Specifying mandatory acknowledgment

You can specify that interlock and supervision alarms must be acknowledged by the operator.

Procedure

To specify mandatory acknowledgment for interlock or supervision alarms, follow these steps:

1. Open the alarm view for the GRAPH function block.
2. If you want to enable mandatory acknowledgment for interlock alarms, select the "Acknowledge interlock alarms" check box in the "Interlock alarms" area. If you want to cancel mandatory acknowledgment, clear the "Acknowledge interlock alarms" check box.
3. If you want to enable mandatory acknowledgment for supervision alarms, select the "Acknowledge supervision alarms" check box in the "Supervision alarms" area. If you want to cancel mandatory acknowledgment, clear the "Acknowledge supervision alarms" check box.

Note

For GRAPH function blocks in S7-300/400 for which you are using the "Create minimized DB" option, you cannot change the mandatory acknowledgment.

See also

Interlock and supervision alarms (Page 2740)

Enabling and disabling the alarm view (Page 2815)

Editing alarms (Page 2816)

Eliminating syntax errors in the program

Basic information on syntax errors

Syntax errors

Below are some examples of syntax errors:

- Missing separators or the use of too many separators
- Incorrect keyword spelling
- Incorrect jump label spelling/notation
- Notation which does not match the set mnemonics (for example, "I2.3" instead of "E2.3")
- The use of key words as operands

Identification of syntax errors

Syntax errors are underlined in red or appear in red type.

This identification allows you to recognise incorrect inputs at a glance and jump from error to error to eliminate them. Syntax errors are also listed in the "Info" tab of the inspector window with an error message.

See also

Finding syntax errors in the program (Page 2819)

Finding syntax errors in the program

Procedure

To find syntax errors in the program, follow these steps:

1. Select the position in the program in which you wish to look for errors.
2. Click "Go to next error" in the toolbar.
   The first error after the position you have selected will be marked.

You can use "Go to next error" and "Go to previous error" in the toolbar to find and correct all errors in the block.

Or:

1. Open the error list in the inspector window with "Info > Syntax".
   All syntax errors are listed in the table with a short description of the error.
2. If there are any errors, click on the blue question mark next to the error text to obtain information on eliminating the problem.
3. Double-click the error you want to correct.
   The corresponding error is highlighted.

See also

Basic information on syntax errors (Page 2818)

Changing the programming language

Rules for changing the programming language

Rules

Observe the following rules if you want to change the programming language for a block:

- All CPU series:
  - You can only change the programming language of entire blocks. The programming language cannot be changed for individual networks.
  - You cannot switch blocks programmed in the programming languages SCL or GRAPH. In GRAPH blocks, however, you can change between LAD and FBD as network languages.
- S7-300/400:
  - You can only change between the programming languages LAD, FBD and STL.
  - You can create networks within a block using another programming language and then copy them into the desired block.
  - If the language of individual networks of the block cannot be changed, these networks is displayed in their original language.
- S7-1200/1500:
  - You can change between the programming languages LAD and FBD.
- S7-1500:
  - You can create STL networks within the LAD and FBD blocks. However, you cannot copy between STL and LAD/FBD.

Change the programming language

Procedure

To change the programming language, follow these steps:

1. Right-click the block in the project tree.
2. Select the "Properties" command in the shortcut menu.
   The dialog with the properties of the block opens.
3. Select the "General" entry in the area navigation.
4. Select the new programming language in the "Language" drop-down list.
5. Confirm your selection with "OK".

See also

Rules for changing the programming language (Page 2820)

Handling program execution errors

Basics of error handling

Introduction

Program execution errors are programming or I/O access errors. You have a number of different options for responding to program execution errors depending on the CPU used.

Handling program execution errors in S7-300/400

You can program the program execution error OB (OB 85) for S7-300/400 CPUs. If a program execution error occurs and you do not use the program execution error OB, the CPU will switch to "STOP" mode.

You will find additional information about the program execution error OB in the description of the mode of operation of S7-300/400 CPUs.

Handling program execution errors in S7-1200/1500

You can select the type of error handling for CPUs of the S7-1200 and S7-1500 series. You have the following two options:

- Use the CPU's global troubleshooting:
  - S7-1200: The CPU generates a diagnostic buffer entry and remains in "RUN" mode.
  - S7-1500: You can program the programming error OB (OB 121) and the I/O access error OB (OB 122) for S7-1500 CPUs. If no programming error OB exists in the CPU, the CPU switches to "STOP" mode when a programming error occurs. In the event of an I/O access error, the CPU always remains in "RUN" mode, regardless of whether the I/O access error OB is present.
    Please note, however, that an existing I/O access or programming error OB is not called synchronously to the error. Therefore, depending on the selected priority, the execution of I/O access or programming error OBs may be delayed instead of taking place immediately when the error occurs. If other errors occur before execution of the I/O access or programming error OB is complete, no further I/O access or programming error OB is called. If you want to prevent I/O access or programming error OBs from being discarded, set the priority correspondingly high.
    You can use the enable output ENO to detect I/O access and programming errors for the instructions "Read field" (FieldRead), "Write field" (FieldWrite) , "Read memory address" (PEEK) and "Write memory address" (POKE).
    You can find more information about these error OBs in the description of the mode of operation of S7-1500 CPUs.
- You use separate local error handling. Local error handling is error handling within a block. Local error handling has the following advantages:
  - The error information is stored in the system memory, which you can query and evaluate.
  - You can use the error information to program a response in the block to the error that has occurred.
  - Programmed error evaluation and error reactions do not interrupt the program cycle.
  - The system performance is not unnecessarily burdened by the local error handling. If no errors occur, programmed error analyses and reactions are not executed.
    Local error handling applies only to blocks for which it has been set explicitly. If local error handling is set for a block, no global error handling is conducted for errors in this block.

Note

Note the following information:

- All memory access errors and I/O access errors must be captured either by global or local error handling.
- If the parameters of an instruction do not cause any memory access errors, you can query the associated ENO.

See also

GET_ERROR: Get error locally (Page 3688)

GET_ERR_ID: Get error ID locally (Page 3692)

GET_ERROR: Get error locally (Page 3388)

GET_ERR_ID: Get error ID locally (Page 3392)

Querying and fixing errors in the program code (Page 258)

Local error handling

Principles of local error handling

Introduction

Local error handling makes it possible to query the occurrence of errors within a block and evaluate the associated error information. You can set local error handling for organization blocks (OBs), function blocks (FBs), and functions (FCs). If local error handling is enabled, the system reaction is ignored.

Local error handling applies only to blocks for which it has been set explicitly. The local error handling setting is not assumed by a calling block, nor is it transferred to called blocks. For higher-level blocks and lower-level blocks, the system settings still apply provided dedicated error handling has not been programmed for these blocks.

General procedure for local error handling

When errors occur while a block is being executed with local error handling, a predefined response is initiated based on the following error types:

- Write errors: These errors are ignored, and program execution simply continues.
- Read errors: Program execution continues with the substitute value "0".
- Execution errors: Execution of the instruction is aborted. Program execution resumes with the next instruction.

Information about the first error that occurs is stored in the system memory. This information can be queried and output with an instruction (GET_ERROR or GET_ERR_ID). Error information is output in a format that can undergo additional processing. You can use additional instructions to analyze error information and program a reaction to the error based.

When information about the first error is queried, the error memory space in the system memory is enabled. Then, when additional errors occur, information about the next error is output.

Instructions for local error handling

You can use the following instructions for local error handling:

- GET_ERROR: Get error locally
- GET_ERR_ID: Get error ID locally

The instructions differ in the amount of error information that is output with each one.

For additional information on the instructions, refer to "See also".

See also

GET_ERROR: Get error locally (Page 3688)

GET_ERR_ID: Get error ID locally (Page 3392)

GET_ERR_ID: Get error ID locally (Page 3692)

GET_ERROR: Get error locally (Page 3388)

Error output priorities

Overview of the priorities

In local error handling, information about the first error that occurred is displayed. If multiple errors occur at the same time while an instruction is being executed, these errors are displayed according to their priority. The following table shows the priority of different types of errors.

Priority	Error type
1	Error in the program code
2	Missing reference
3	Invalid range
4	DB does not exist
5	Operands are not compatible
6	Width of specified area is not sufficient
7	Timers or counters do not exist
8	No write access to a DB
9	I/O error
10	Instruction does not exist
11	Block does not exist
12	Invalid nesting depth

The highest priority is 1 and the lowest priority is 12.

See also

GET_ERROR: Get error locally (Page 3688)

GET_ERR_ID: Get error ID locally (Page 3392)

GET_ERR_ID: Get error ID locally (Page 3692)

GET_ERROR: Get error locally (Page 3388)

Enabling local error handling for a block

Introduction

Local error handling is enabled for a block if you insert one of the following instructions in a network.

- GET_ERROR: Get error locally
- GET_ERR_ID: Get error ID locally

For additional information on the instructions, refer to "See also".

If local error handling is enabled for a block, the system reactions for this block are ignored.

Requirement

- The block is open.
- Die "Instructions" task card is open.

Procedure

To enable local error handling for a block, proceed as follows:

1. Navigate to the "Basic instructions" pane of the "Instructions" task card.
2. Open the "Program Control" folder.
3. Drag the instruction "Get error locally" (GET_ERROR) or "Get error ID locally" (GET_ERR_ID) to the required network.

Result

Local error handling is enabled for the open block. The "Handle errors within block" check box is selected in the Inspector window under "Properties > Attributes". This setting cannot be edited in the Inspector window. Local error handling can be deactivated by deleting the inserted instructions on local error handling.

See also

GET_ERROR: Get error locally (Page 3688)

GET_ERR_ID: Get error ID locally (Page 3392)

GET_ERR_ID: Get error ID locally (Page 3692)

GET_ERROR: Get error locally (Page 3388)

9.1.4.3 Programming data blocks

Basic principles for programming of data blocks

A data block (DB) is used to save the values that are written during execution of the program.

In contrast to the code block, the data block contains only tag declarations. It contains no networks or instructions. The tag declarations define the structure of the data block.

Types of data blocks

There are two types of data blocks:

- Global data blocks
  The global data block is not assigned to a code block. You can access the values of a global data block from any code block. A global data block contains only static tags.
  The structure of the global data block can be freely defined. In the declaration table for data blocks, you declare the data elements that are to contained in the global data block.
- Instance data blocks
  The instance data block is assigned directly to a function block (FB). The structure of an instance data block cannot be freely defined, but is instead determined by the interface declaration of the function block. The instance data block contains exactly those block parameters and tags that are declared there.
  However, you can define instance-specific values in the instance data block, for example, start values for the declared tags.

ARRAY data blocks (S7-1500)

ARRAY data blocks are global data blocks that consist of an ARRAY. This ARRAY can be based on any data type. For example, an ARRAY of a PLC data type (UDT) is possible. The DB contains no other elements besides the ARRAY. Because of their flat structure, ARRAY data blocks facilitate access to the ARRAY elements and their transfer to called blocks.

The "Move operations" section of the "Instructions" task card offers options for addressing of ARRAY DBs.

PLC data types as a template for global data blocks

PLC data types can be used as templates for the creation of global data blocks with identical data structures. You create the structure as PLC data type only once and then generate the required data blocks by assigning the PLC data type.

System data types as a template for global data blocks

System data types can also be used as templates for creating global data blocks with identical data structure. System data types already have a pre-defined structure. You insert the system data type in the program only once and then generate additional data blocks with an identical structure by assigning the system data type.

Access modes

There are two different modes of accessing data values in data blocks:

- Data blocks with optimized access (only S7-1200)
  Data blocks with optimized access have no fixed defined structure. In the declaration, the data elements are assigned only a symbolic name and no fixed address within the block. You access the data values in these block via symbolic names.
  The "Optimized block access" attribute is always enabled for ARRAY data blocks.
- Data blocks with standard access (all CPU families)
  Data blocks with standard access have a fixed structure. In the declaration, the data elements are assigned both a symbolic name and a fixed address within the block. You can access the data values in these blocks via symbolic names or the address.
  ARRAY data blocks with standard access are not possible.

Retentivity of data values

To prevent data loss in the event of power failure, you can store the data values in a retentive memory area.

See also

Creating data blocks (Page 2503)

Blocks with optimized access (Page 2175)

Global data blocks (DB) (Page 2172)

Instance data blocks (Page 2174)

Structure of the declaration table for data blocks

Structure of the declaration table for data blocks

The figure below shows the structure of the declaration table for data blocks. The display will vary depending on type of block and type of access.

	Name	Data type	Start value	Retain	Visible in HMI	Comment
	Input					
	MyInput1	Bool	false	☑	☑	
	Output					
	MyOutput1	Byte	0	☐	☑	
	InOut					
	Static					

Display of instance-specific values

In instance data blocks, you can apply the already defined values from the interface of the assigned function block or define instance-specific start values. Values that are applied from

the function block cannot be edited. You can replace the grayed-out values with instance-specific values. Values that were already changed instance specific are not grayed out.

Meaning of the columns

The following table shows the meaning of the individual columns. You can show or hide the columns as required. The number of columns displayed varies depending on the CPU series.

Column	Explanation
	Symbol you can click to move or copy the tag. You can, for example, drag-and-drop the tag into a program and use it there as operand.
Name	Name of the tags.
Data type	Data type of the tags.
Offset	Relative address of the tags. The column is only visible in data blocks with standard access.
Default value	Default value of the tag in the interface of a higher-level code block or in a PLC data type. The values contained in the "Default value" column can only be changed in the higher-level code block or PLC data type. The values are only displayed in the data block.
Start value	Value that the tag should assume at startup. The default values defined in a code block are used as start values during the creation of the data block. You can then replace these adopted values with instance-specific start values. Specification of an start value is optional. If you do not specify any value, the tag assumes the default value at startup. If a default is not defined either, the default value valid for the data type is used. For example, the value "FALSE" is specified as standard for BOOL.
Monitor value	Current data value in the CPU. This column only appears if an online connection is available and you click "Monitor".
Snapshot	Shows values that were loaded from the device.
Retentivity	Marks the tag as retentive. The values of retentive tags are retained even after the power supply is switched off.
Visible in HMI	Shows whether the tag is visible by default in the HMI selection list.
Accessible from HMI	Shows whether HMI can access this tag during runtime.
Setting value	Setting values are the values that will probably have to be fine tuned during commissioning. After commissioning, the values of these tags can be transferred to the offline program as start values and stored there.
Comment	Comment to document the tags.

See also

Editing the properties of tags in data blocks (Page 2841)

Setting data values during commissioning (Page 2850)

Creating data blocks (Page 2503)

Creating a data structure for global data blocks (Page 2833)

Basic information on start values (Page 2836)
Define start values (Page 2836)
Setting retentivity (Page 2839)

Creating data blocks

Requirement

The "Program blocks" folder in the project tree is open.

Procedure

To create a data block, follow these steps:

1. Double-click the "Add new block" command.
   The "Add new block" dialog box opens.
2. Click the "Data block (DB)" button.
3. Select the type of the data block. You have the following options available to you:
   - To create a global data block, select the list entry "Global DB".
   - To create an ARRAY data block, select the "ARRAY DB" entry in the list.
   - To create an instance data block, select the function block to which you want to assign the instance data block from the list. The list contains only the function blocks that were previously created for the CPU.
   - To create a data block based on a PLC data type, select the PLC data type from the list. The list contains only the PLC data types that were previously created for the CPU.
   - To create a data block based on a system data type, select the system data type from the list. The list contains only those system data types that have already been inserted to program blocks in the CPU.
4. Enter a name for the data block.
5. Enter the properties of the new data block.
6. If you have selected an ARRAY DB as the data block type, enter the ARRAY data type and the high limit for the ARRAY.
   You can change the high limit for the ARRAY at any time in the property window of the created block. The ARRAY data type cannot be changed subsequently.
7. To enter additional properties of the new data block, click "Additional information".
   An area with further input fields is displayed.
8. Enter all the properties you require.
9. Activate the "Add new and open" check box if the block does not open as soon as it is created.
10. Confirm your entry with "OK".

Result

The new data block is created. You can find the data block in the project tree in the "Program blocks" folder.

See also

Instance data blocks (Page 2174)

Programming data blocks (Page 2825)

Programming PLC data types (Page 2854)

Global data blocks (DB) (Page 2172)

Blocks with optimized access (Page 2175)

Overview of block properties (Page 2512)

Updating data blocks

Introduction

Changes in the interface of a function block or a PLC data type can lead to the corresponding data blocks becoming inconsistent. These inconsistencies are marked in red in the declaration table and at the call point of the block. To remedy these inconsistencies, the data blocks must be updated.

You have three options to update block calls:

- Explicit updating in the declaration table for data blocks.
  The data block is updated. Changes from the interface of the assigned function block and changes to the used PLC data types are applied.
- Explicit updating in the program editor.
  The block calls in the open block will be updated. The associated instance data block is also adjusted in the process.
- Implicit updating during compilation.
  All block calls in the program as well as the used PLC data types and the corresponding instance data blocks are updated.

Explicit Updating in the Declaration Table for Data Blocks

To explicitly update an individual data block, follow these steps:

1. Open the data block.
2. Select "Update interface" in the shortcut menu.

Explicit Updating in the Program Editor

To update all block calls or a specific call within a block, follow these steps:

1. Open the block in the program editor.
2. Right-click on the instruction with the block call.
3. Select the "Update" command in the shortcut menu.
4. The "Interface update" dialog opens. This dialog shows the differences between the block interface in use and the changed interface of the called block.
5. If you want to update the block call, click "OK". To cancel the update, click "Cancel".

Implicit Updating during Compilation

To implicitly update all block calls and uses of PLC data types as well as the instance data blocks during the compiling, follow these steps:

1. Open the project tree.
2. Select the "Program blocks" folder.
3. Select the command "Compile > Software (rebuild all blocks)" in the shortcut menu.

See also

Changing the properties of tags in instance data blocks (Page 2843)

Extending data blocks

Description

In order to enable the editing of PLC programs that have already been commissioned and that are running without error on a system, CPUs of the S7-1500 series and most CPUs of the S7-1200 V4 series support the option of extending global data blocks during runtime.

You can download the modified blocks without setting the CPU to STOP and without affecting the values of already loaded tags.

This is a simple means of implementing program changes. This load process (download without reinitialization) will not have a negative impact on the controlled process.

Principle of operation

Each data block is always assigned a default memory reserve. The memory reserve is not used initially. Activate the memory reserve if you decide on loading interface changes after having compiled and downloaded the block. All tags that you subsequently declare will be saved to the memory reserve. A subsequent download has no impact on the values of tags that have already been loaded.

If you decide to review your program at a later time while the plant is not in operation, you are also provided an option of reworking the memory layout of individual or several blocks in a

single pass. With this action, you move all tags from the reserve area to the regular area. The memory reserve is now cleared and made available for further interface extensions.

Requirements

This "Download without reinitialization" function is available if the following requirements are met:

- The project is in the "TIA Portal V12" format or a higher version.
- You are working with a CPU that supports "Download without reinitialization".
- The blocks were created in LAD, FBD, STL, or SCL.
- The blocks were created by the user, i.e. they are not included with the blocks delivered in your package.
- These blocks are assigned the optimized access attribute.

Basic steps

Perform the following steps if you want to extend the data block and then load the block without re-initialization.

1. All blocks have a default memory reserve of 100 bytes. You can adapt this memory reserve to suit your requirements.
2. Activate the memory reserve.
3. Extend the block interface.
4. Compile the block.
5. Download the block to the CPU as usual.

Reference

For more information on the various steps, refer to chapter "Loading blocks (S7-1200/1500) ".

Creating a data structure for global data blocks

Declaring tags of elementary data type

Requirement

A global data block is open.

Note

You cannot change the structure of instance data blocks and of data blocks based on a PLC data type directly, since the structures of these blocks are defined by the respective function block or the PLC data type.

The type of the data block is entered in the block properties.

Procedure

To declare a tag of the elementary data type, follow these steps:

1. Enter a tag name in the "Name" column.
2. In the "Data type" column, click the button for the data type selection.
   A list of the permissible data types is opened.
3. Select the desired data type.
4. Optional: Change the properties of the tags that are displayed in the other columns.
5. Repeat steps 1 to 4 for all tags that are to be declared.

See also

Displaying and editing block properties (Page 2517)
Declaring tags of the ARRAY data type (Page 2834)
Declaring tags of STRUCT data type (Page 2835)
Editing tables (Page 316)
Editing the declaration table for data blocks (Page 2846)
Define start values (Page 2836)
Setting retentivity (Page 2839)
Monitoring data values online (Page 2849)

Declaring tags of the ARRAY data type

Requirement

A global data block is open.

Procedure

To declare a tag of the ARRAY data type, follow these steps:

1. Enter a tag name in the "Name" column.
2. Enter the "Array" data type in the "Data type" column. You will be supported by autocompletion in this step.
   The "Array" dialog opens.
3. In the "Data type" text box, specify the data type of the array elements.
4. In the "ARRAY limits" text box, specify the high and low limit for each dimension.
   Example of a one-dimensional ARRAY:
   `[0..3]`
   Example of a three-dimensional ARRAY:
   `[0..3, 0..15, 0..33]`
5. Confirm your entry.
6. Optional: Change the properties of the tags that are displayed in the other columns.

The tag is created but remains collapsed. To expand the ARRAY, click the triangle in front of the tag. Not that you cannot expand very large ARRAYs for reasons of clarity.

Entering start values of ARRAY elements

To set default start values for the individual elements of an ARRAY, follow these steps:

1. Click the triangle in front of the ARRAY data type tags.
   The ARRAY opens and the individual ARRAY elements are shown in separate rows.
2. Enter the required value in the "Start value" column.

See also

Define start values (Page 2836)

Setting retentivity (Page 2839)

Editing the declaration table for data blocks (Page 2846)

Array (Page 2267)

Using autocompletion (Page 2540)

Declaring tags of STRUCT data type

Requirement

A global data block is open.

Procedure

To declare a tag of the STRUCT data type, follow these steps:

1. Enter a tag name in the "Name" column.
2. Enter "Struct" in the "Data type" column. You will be supported by autocompletion during input.
   An empty, indented row is inserted after the new tag.
3. Insert the first structural element in the first empty row.
   An additional empty row is inserted after the element.
4. Select a data type for the structure element.
5. Optional: Change the properties of the structural element that is displayed in the other columns of the block interface.
6. Repeat the step 4 to 7 for all additional structure elements.
   It is not necessary to end the structure explicitly. The structure ends with the last element that is entered.
7. To insert a new tag after the structure, leave a blank row after the end of the structure and then start the new tag in the second empty row.

Result

The tag of STRUCT data type is created.

Enter start values of structure elements

To set default start values for the individual elements of a structure, follow these steps:

1. Click the triangle in front of the STRUCT data type tags.
   The structure opens and the individual structure elements are shown in separate rows.
2. Enter the required value in the "Start value" column.

See also

Define start values (Page 2836)

Setting retentivity (Page 2839)

Editing the declaration table for data blocks (Page 2846)

STRUCT (Page 2272)

Using autocompletion (Page 2540)

Declaring tags based on a PLC data type

Requirements

- A global data block is open.
- A PLC data type is declared in the current CPU.

Procedure

To declare a tag based on a PLC data type, follow these steps:

1. Enter a tag name in the "Name" column.
2. Enter the PLC data type in the "Data type" column. You will be supported by autocompletion during input.
3. Optional: Change the properties of the tags that are displayed in the other columns of the table.

Result

The tag is created.

See also

Layout of the block interface (Page 2544)

Editing the declaration table for data blocks (Page 2846)

Using autocompletion (Page 2540)

Define start values

Basic information on start values

Definition of "Start value"

The start value of a tag is a value defined by you which the tag assumes after a CPU startup.

The retentive tags have a special status. Their values take the defined start value only after a "cold restart". After a "warm restart", they retain their values and are not reset to the start value.

Definition of "Default value"

The structure of the data blocks can be derived from higher-level elements.

- An instance data block is based, for example, on the interface of a higher-level code block.
- A global data block can be based on a predefined PLC data type.

In this case you can define a default value for each tag in the higher-level element. These default values are used as start values during the creation of the data block. You can then replace these values with instance-specific start values in the data block.

Specification of an start value is optional. If you do not specify any value, the tag assumes the default value at startup. If a default is not defined either, the default value valid for the data type is used. For example, the value "FALSE" is specified as standard for BOOL.

See also

Define start values (Page 2837)

Structure of the declaration table for data blocks (Page 2827)

Declaring local tags and constants in the block interface (Page 2550)

Applying values from the online program as start values (Page 2853)

Define start values

Define start values

To define the start values for the tags of a data block, follow these steps:

1. Open the data block.
   The "Default value" column shows the default values that were defined for the tags in the interface of a higher-level code block or in a PLC data type.
2. Click the "Expanded mode" button to show all elements of structured data types.
3. Enter the desired start values in the "Start value" column. The value must match the data type of the tag and should not exceed the range of the data type.
   The start values are defined. The tag takes the defined value at startup, provided it was not declared as retentive.

Resetting a tag to the default value

To reset a tag for which you have defined a start value to the default value, follow these steps:

1. Select a modified value in the table.
2. Delete the value.
   The default value is entered. The default value is displayed.

Resetting all tags to the default value

To reset to the default value all tags for which you have defined an start value, follow these steps:

1. Select the "Reset start values" icon in the toolbar.
   The default values are transferred to the "Start value" column. Write-protected start values are not overwritten.

See also

Basic information on start values (Page 2836)

Applying values from the online program as start values (Page 2853)

Editing the declaration table for data blocks (Page 2846)

Loading changed values

Introduction

To apply the changed start values from the offline program to the online program, you must load the changes. The following cases must be distinguished:

- Loading changed start values of non-retentive tags
- Loading changed start values of retentive tags
- Loading changed start values of setting values

Requirement

The start values in the offline program were changed.

Procedure

To load changed start values of non-retentive tags, follow these steps:

1. Select the blocks to be loaded in the project tree.
2. Select the "Download to device > Software (only changes)" command from the shortcut menu.
   The blocks are compiled and loaded.
   The start values of the newly defined tags are placed in the load memory of the CPU. The program runs with the new start values at the next transition from STOP to RUN.

To load changed start values of retentive tags, follow these steps:

1. Select the blocks to be loaded in the project tree.
2. Select the "Compile > Software (rebuild all blocks)" command in the shortcut menu.
3. In the "Online" menu, select the "Download and reset PLC program" command.
   The online blocks are deleted and replaced with the new blocks. This reinitializes all tags, including the retentive tags.

Information about loading changed setting values and general information about loading can be found under "See also".

See also

Compiling and downloading blocks (Page 2898)

Initializing setting values in the online program (Page 2852)

Setting retentivity

Retentivity of tags in data blocks

Retentive behavior

To prevent data loss in the event of power failure, you can mark the data as retentive. This data is stored in a retentive memory area. The options for setting the retentivity depend on the type of data block and the type of block access that is set.

See also

Setting retentivity in an instance data block (Page 2839)

Setting retentivity in a global data block (Page 2840)

Setting retentivity in an instance data block

Introduction

In an instance data block, the editability of the retentive behavior depends on the type of access of the higher-level function block:

- Function block with standard access
  You can define the instance data both as retentive or non-retentive. Individual retentivity settings are not possible for individual tags.
- Function block with optimized access
  In the instance data block, you can define the retentivity settings of the tags that are selected in the block interface with "Set in IDB". With these tags also, you cannot individually set the retentive behavior for each tag. The retentivity setting has an impact on all tags that are selected in the block interface with "Set in IDB".

Setting Retentivity for Standard Access

To centrally set the retentivity of all tags in the data block with standard access, follow these steps:

1. Open the instance data block.
2. Select the check box in the "Retain" column of a tag.
   All tags are defined as retentive.
3. To reset the retentivity setting for all tags, clear the check box in the "Retain" column of a tag.
   All tags will be defined as non-retentive.

Setting Retentivity for Optimized Access

To set the retentive behavior of the tags that are selected with "Set in IDB" in data blocks with optimized access, follow these steps:

1. Open the instance data block.
2. Select the check box in the "Retain" column of a tag.
   All tags selected with "Set in IDB" in the data block interface are defined as retentive.
3. To reset the retentivity setting for the tags, clear the check box in the "Retain" column of a tag.
   All tags selected with "Set in IDB" in the data block interface will be defined as non-retentive.

See also

Basics of block access (Page 2175)

Retentivity of tags in data blocks (Page 2839)

Editing the declaration table for data blocks (Page 2846)

Setting retentivity in a global data block

Introduction

In a global data block, the editability of the retentive behavior depends on the type of access:

- Global data block with standard access
  You can define the data both as retentive or non-retentive. Individual retentivity settings are not possible for individual tags.
- Global data block with optimized access
  You can individually define the retentivity settings of the tags. For tags with structured data types, retentivity settings are transferred for all tag elements.

Setting Retentivity for Standard Access

To centrally set the retentivity of all tags in the data block with standard access, follow these steps:

1. Open the global data block.
2. Select the check box in the "Retain" column of a tag.
   All tags are defined as retentive.
3. To reset the retentivity setting for all tags, clear the check box in the "Retain" column of a tag.
   All tags are defined as non-retentive.

Setting Retentivity for Optimized Access

To individually set the retentivity of all tags in data blocks with optimized access, follow these steps:

1. Open the global data block.
2. In the "Retain" column, select the check box for the tags for which you want to set a retentive behavior.
   The selected tag is defined as retentive.
3. To reset the retentivity setting for the tags, clear the check box in the "Retain" column of a tag.
   All selected tags are defined as non-retentive.

See also

Basics of block access (Page 2175)

Retentivity of tags in data blocks (Page 2839)

Editing the declaration table for data blocks (Page 2846)

Editing the properties of tags in data blocks

Properties of the tags in data blocks

Properties

The following table provides an overview of the properties of tags in data blocks:

Group	Property	Description
General	Name	Name of the tags.
	Data type	Data type of the tags.

Group	Property	Description
	Default value	Default value of the tag in the interface of a higher-level code block or in a PLC data type. The values contained in the "Default value" column can only be changed in the higher-level code block or PLC data type. The values are only displayed in the data block.
	Start value	Value that the tag should assume at CPU startup. The default values defined in a code block are used as start values during the creation of the data block. You can then replace these adopted values with instance-specific start values. Specification of an start value is optional. If you do not specify any value, the tag assumes the default value at startup. If a default is not defined either, the default value valid for the data type is used. For example, the value "FALSE" is specified as standard for BOOL.
	Comment	Comment on the tag.
Attributes	Retain	Marks the tag as retentive. The values of retentive tags are retained even after the power supply is switched off. This attribute is only available in the interface of the function block with optimized access.
	Accessible from HMI	Indicates whether the tag can be used in HMI. When the attribute is set, you have read or write access to the tag from the HMI. When the attribute is not set, you cannot access the tag from the HMI. Please note, however, that you cannot implement general access protection for the tag with the "Accessible from HMI" attribute. Read or write access from other applications is possible even if the attribute is not enabled.
	Visible in HMI	Shows whether the tag is visible by default in the HMI selection list.
	Hidden parameter	Indicates whether the tag should be hidden for the block call. This is only possible if you have specified a valid predefined actual parameter beforehand.
	Predefined actual parameter	Defines a parameter that is to be used as actual parameter during the block call.
	Visible	Indicates whether a parameter is visible in CFC.
	Configurable	Indicates whether a parameter is configurable in CFC.
	For test	Indicates whether a parameter is registered for the CFC test mode.
	Interconnectable	Indicates whether a parameter is interconnectable in CFC.
	Enable tag readback	Indicates whether a parameter is relevant for the "Read back chart" function in CFC.
	Enumeration texts	Assigns a parameter to an enumeration in CFC.
	Engineering unit	Assigns a parameter to a unit in CFC.
	Low limit	Defines the low limit for the parameter in CFC.
	High limit	Defines the high limit for the parameter in CFC.

See also

Changing the properties of tags in instance data blocks (Page 2843)

Changing the properties of tags in global data blocks (Page 2844)

Setting retentivity (Page 2839)

Define start values (Page 2836)

Changing the properties of tags in instance data blocks

Instance-specific tag properties

Two options are available for defining the tag properties:

- The tag properties are applied from the interface of the assigned function block. Properties that are applied from the function block are displayed grayed out out in the columns of the declaration table. The "Name" and "Data type" properties are always applied.
- You define instance-specific properties.
  You can change some properties instance specific. Changeable values are, for example, "Comment" or "Visible in HMI". Properties that were changed instance specific are not grayed out in the columns of the declaration table. The instance-specific changes are retained, even if the interface of the higher-level function block is changed and the instance data blocks are subsequently updated.

Editing properties of an element in the declaration table

To edit the properties of an element, follow these steps:

1. Open the instance data block.
2. Select the required element in the table.
3. Change the entries in the columns.

Editing properties of several elements in the declaration table

You can also simultaneously set or reset the "Retain", "Visible in HMI", "Accessible in HMI" and "Setpoint" columns for one or more selected elements.

To change one of these properties for several elements, follow these steps:

1. Open the data block.
2. Hold down the CTRL key.
3. In the required column, select each of the table cells whose value you want to change.
4. Select the "Set <property>" or "Reset <property>" command in the shortcut menu.

Editing properties in the properties window

To edit the properties of an individual tag, follow these steps:

1. Select a tag in the table.
2. Select the "Properties" command in the shortcut menu.
   The properties window opens. It shows the properties of the tag in the "General" and "Attributes" areas.
3. Select the required area in the area navigation.
4. Change the entries in the text boxes.

Reset individual properties to the default value.

To reset individual tag properties to the value that was defined as default in the function block, follow these steps:

1. Select an instance-specific, modified value in the table.
2. Delete the value.
   The instance-specific value will be deleted and the default value from the interface of the function block entered. The default value is displayed grayed out.

See also

Updating data blocks (Page 2830)

Properties of the tags in data blocks (Page 2841)

Editing the declaration table for data blocks (Page 2846)

Changing the properties of tags in global data blocks

Introduction

Two options are available for defining the tag properties:

- The tag properties are applied from the PLC data type.
  Properties that are applied from the PLC data type are shown grayed out in the columns of the declaration table. The "Name" and "Data type" properties are always applied.
- You define specific properties.
  You can change some properties in the global data block. Changeable values are, for example, "Comment" or "Visible in HMI". Properties that were changed are not grayed out in the columns of the declaration table. The changes are retained, even if the PLC data type changes and the global data block is subsequently updated.

Editing properties of an element in the declaration table

To edit the properties of an element, follow these steps:

1. Open the global data block.
2. Select the required element in the table.
3. Change the entries in the columns.

Editing properties of several elements in the declaration table

You can also simultaneously set or reset the "Retain", "Visible in HMI", "Accessible in HMI" and "Setpoint" columns for one or more selected elements.

To change one of these properties for several elements, follow these steps:

1. Open the data block.
2. Hold down the CTRL key.
3. In the required column, select each of the table cells whose value you want to change.
4. Select the "Set <property>" or "Reset <property>" command in the shortcut menu.

Editing properties in the properties window

To edit the properties of an individual tag, follow these steps:

1. Select a tag in the table.
2. Select the "Properties" command in the shortcut menu.
   The properties window opens. It shows the properties of the tag in the "General" and "Attributes" areas.
3. Select the required area in the area navigation.
4. Change the entries in the text boxes.

Reset individual properties to the default value.

To reset individual tag properties to the value that was defined as default in the PLC data type, follow these steps:

1. Select a modified value in the table.
2. Delete the value.
   The default value from the PLC data type is entered. The default value is displayed grayed out.

See also

Properties of the tags in data blocks (Page 2841)

Editing the declaration table for data blocks (Page 2846)

Editing the declaration table for data blocks

Inserting table rows

Procedure

Proceed as follows to insert a row above the selected row:

1. Select the row in front of which you want to insert a new row.
2. Click the "Insert row" button on the toolbar of the table.

Result

A new row is inserted above the selected row.

See also

Editing tables (Page 316)

Inserting table rows

Procedure

Proceed as follows to insert a row below the selected row:

1. Select the row below which you want to insert a new row.
2. Click the "Add row" button on the table toolbar.

Result

A new empty row will be inserted below the selected row.

See also

Editing tables (Page 316)

Deleting tags

Requirements

A global data block is open.

Procedure

To delete a tag, follow these steps:

1. Select the row with the tag to be deleted. You can also select several rows by clicking on them one after the other while holding down the <Ctrl> key or by pressing and holding down <Shift> and clicking on the first and last row.
2. Select the "Delete" command in the shortcut menu.

Note

You cannot directly change the structure of instance data blocks and of global data blocks based on a PLC data type, since the structures of these blocks are defined in the higher-level object.

The type of the data block is entered in the block properties.

See also: Displaying and editing block properties (Page 2517)

See also

Editing tables (Page 316)

Automatically filling in successive cells

You can load the contents of one or several table cells into the cells below, automatically filling in the successive cells.

If you automatically fill in cells in the "Name" column, a consecutive number will be appended to each name. For example, "Motor" will become "Motor_1".

You can define individual or more cells as well as entire rows as source area.

If less rows exist in the open table than you want to fill, then you will first have to insert additional empty rows.

Requirement

- The table is open.
- Sufficient declaration rows are available.

Procedure

To automatically fill in successive cells, follow these steps:

1. Select the cells to be loaded.
2. Click the "Fill" symbol in the bottom right corner of the cell.
   The mouse pointer is transformed into a crosshair.

3. Keep the mouse button pressed and drag the mouse pointer downwards over the cells that you want to fill in automatically.
4. Release the mouse button.
   The cells are filled in automatically.
5. If entries are already present in the cells that are to be automatically filled in, a dialog appears. In this dialog you can indicate whether you want to overwrite the existing entries or insert new rows for the new tags.

Show and hide table columns

You can show or hide the columns in a table as needed.

Procedure

To show or hide table columns, follow these steps:

1. Click a column header.
2. Select the "Show/Hide" command in the shortcut menu.
   The selection of available columns is displayed.
3. To show a column, select the column's check box.
4. To hide a column, clear the column's check box.

Editing tags with external editors

To edit individual tags in external editors outside the TIA portal, you can export or import these tags using copy & paste. However, you cannot copy structured tags to an editor.

Requirement

The data block and an external editor are opened.

Procedure

To export and re-import individual tags by drag-and-drop operation, follow these steps:

1. Select one or more tags.
2. Select "Copy" in the shortcut menu.
3. Switch to the external editor and paste the copied tags.
4. Edit the tags as required.
5. Copy the tags in the external editor.
6. Switch back to the declaration table.
7. Select "Paste" in the shortcut menu.

Monitoring data values online

Monitoring data values in data blocks online

You can monitor the current data values of the tags in the CPU directly in the declaration table.

Requirement

- An online connection is available.
- The data block has been loaded to the CPU.
- The program execution is active (CPU in "RUN").
- The data block is open.

Procedure

To monitor the data values, proceed as follows:

1. Start monitoring by clicking the "Monitor all" button.
   The additional "Monitor value" column is displayed in the table. This shows the current data values.
   See also: Structure of the declaration table for data blocks (Page 2827)
2. End the monitoring by clicking the "Monitor all" button again.

Displaying data values loaded from the device

During the loading of a data block from a device, the current tag values are also loaded. You can display these values.

Requirement

A data block was loaded from the device.

Procedure

To display the current values, follow these steps:

1. Open the data block.
2. Click a column header.
3. In the shortcut menu, select the "Show/hide columns" command.
   The selection of available columns is displayed.
4. Select the check box in the "Snapshot" column.

Result

The current values will be applied in the "Snapshot" column.

Note

If you subsequently change the structure of the data block, the display of the current values gets lost. The "Snapshot" column will then be empty.

Setting data values during commissioning

Basic information on adjusting data values during commissioning

Introduction

During commissioning of a plant, data values have to be frequently adjusted in order to optimally adapt the program to the general operating conditions on site. The declaration table for data blocks offers the following functions for this purpose:

- Initialize data values in "RUN" mode
  This function enables you to change data values online in order to quickly determine the optimum tag values.
- Transfer tag values from the online program to the offline program as start values
  When you have determined the optimum tag values, you can apply these as start values in the offline program. This allows you to ensure that the program starts with the optimized values the next time it is loaded.

To use the function, first define specific tags as "Setting values" in the program. Setting values are the values that will probably have to be fine tuned during commissioning.

See also

Monitoring data values online (Page 2849)

Define start values (Page 2836)

Marking data as values that can be set

You can mark specific tags in the program as "Setting values". Setting values are the values that will probably have to be fine tuned during commissioning.

Rules

You can mark tags as "Setting value" in the following block types:

- In function blocks (FB), but only in the "Static" section
- in global data blocks (DB)
- in PLC data types (UDT)
  In the case of PLC data types (UDT), however, the setting is only effective, if the UDT is used in the "Static" section of a function block or data block.

It is not possible to define setting values in the following block types:

- In data blocks based on a PLC data type, and in instance data blocks. These inherit the setting from the higher-level FB or UDT.
- You cannot mark tags as a "Setting value" in ARRAY data blocks.
- You also cannot mark tags as a "Setting value" at the call point of a multi-instance. You have to make the setting in the interface of the function block that is called as multiple instance.
- You cannot change the "Setting value" marking in know-how-protected blocks. To do so, you must first remove the know-how protection.

Requirement

A function block, a global data block or a PLC data type (UDT) is open.

Procedure

To mark a tag as "Setting value", follow these steps:

1. Select a tag from the "Static" section.
2. Select the check box in the "Setting value" column.
   - You cannot define the higher-level element of a structure or a PLC data type as "Setting value". You have to make the setting for the lower-level elements individually.
   - In the case of ARRAYs, you can only mark the higher-level element as "Setting value". The lower-level elements inherit the setting.
   - For ARRAYs of STUCT, you can only mark the elements below the first structure as setting values. The elements of other structures inherit the setting.

Result

The tags are marked as setting values. During commissioning, these tags can be initialized online. You do not need to set the CPU to "STOP" mode; it can remain in "RUN". In addition, the current tag values can be transferred as start values to the offline program and saved there.

See also

Monitoring data values online (Page 2849)

Define start values (Page 2836)

Initializing setting values in the online program

Basics on initializing setting values

You can initialize all tags marked as a "Setting value" with new values in the online program. At the same time, the start values will be loaded from the offline program to the online program. The CPU remains in "RUN" mode. All tags that are marked as a setting value are initialized once at the next cycle control point. This applies both to retentive and non-retentive tags. The program execution is then continued with the new tag values.

DANGER

Danger when changing tag values

Changing the tag values while the plant is operating can cause serious damage to property or injury to persons if there are functional disturbances or program errors!

Make sure that no dangerous situations can arise before you re-initialize the setting values.

Requirement

- An online connection to the CPU exists.
- The structure of the data block is identical offline and online.
- One or more tags are marked as a "Setting value".

Procedure

To initialize all setting values of the data block, follow these steps:

1. Open a global data block or an instance data block.
2. Enter the desired values in the "Start value" column. The start values must correspond to the indicated data type.
3. Click the "Initialize setting values" button.

Result

The setting values in the online program are initialized with the start values from the offline program at the next cycle control point.

The maximum number of tags that can be initialized is dependent on the CPU. If too many setting values are marked, an alarm informs you about this. In this case, you can insert the tags in a watch table and initialize them using the "Modify" function in the watch table. Alternatively, you can also load the entire data block. For more information on this, refer to "See also".

See also

Loading changed values (Page 2838)

Applying values from the online program as start values

In order to apply tag values from the online program to the offline program as start values, first create a snapshot of the tag values from the online program. You can then apply them to the offline program. Note that the values from the snapshot are always copied. There is no check to determine whether all values originate from the same cycle.

Write-protected start values are not overwritten.

You have the following basic options for applying the values:

- Applying the values of an open data block
  You can apply all values or only the values of the tags marked as a "setting value" as start values in an open data block.
- Applying the values of multiple blocks in the project tree
  You can either apply all setting values or all retentive values as start values in the project tree.

Requirement

- An online connection to the CPU is available.
- As least one data block has been loaded to the CPU.

Procedure

In order to apply all values or only the values of the tags marked as a "setting value" in a data block, follow these steps:

1. Open the data block.
2. Start monitoring by clicking the "Monitor all" button.
   The "Monitor value" column is displayed in the table. This shows the current data values.
3. On the toolbar, click "Snapshot of monitored values".
   The latest monitored values will be applied in the "Snapshot" column.
4. Click one of the following buttons on the toolbar:
   – "Apply setting values from the snapshot as start values"
   – "Apply all values from the snapshot as start values"

The values from the "Snapshot" column are applied to the "Start value" column.

To apply the monitored values of multiple data blocks in the project tree, follow these steps:

1. Select the blocks in the project tree.
2. Select the "Snapshot of the monitored values" command in the shortcut menu.
   The current monitored values of all selected blocks will be applied in the "Snapshot" column.
   An alarm is shown in the Inspector window after the operation is complete.
3. Then select one of the following commands in the shortcut menu:
   – "Apply monitor values as start values > Only setpoints"
   – "Apply monitor values as start values > Only retain values"

The values from the "Snapshot" column are applied to the "Start value" column.

Result

The new start values are stored in the offline program.

Note

Applying values of individual tags

You can also transfer the values of individual tags that were not marked as a setting value beforehand from the "Snapshot" column to the "Start values" column. Use the "Copy" and "Paste" commands from the shortcut menu to copy the values and insert them in the "Start value" column. Note that only the values that are currently located in the visible area of the table are copied.

See also

Basic information on start values (Page 2836)

Define start values (Page 2837)

Monitoring data values online (Page 2849)

9.1.4.4 Programming PLC data types

Basics of PLC data types

Description

PLC data types are data structures that you define and that can be used multiple times within the program. The structure of a PLC is made up of several components, each of which can contain different data types. You define the type of components during the declaration of the PLC data type.

You can create up to 65534 PLC data types for a CPU of the S7-1200 or S7-1500 series. Each of these PLC data types can include up to 252 components.

PLC data types can be used for the following applications:

- PLC data types can be used as data types for variables in the variable declaration of logic blocks or in data blocks.
- PLC data types can be used as templates for the creation of global data blocks with identical data structures.
- PLC data types can be used in S7-1200 and S7-1500 as a template for the creation of structured PLC tags.

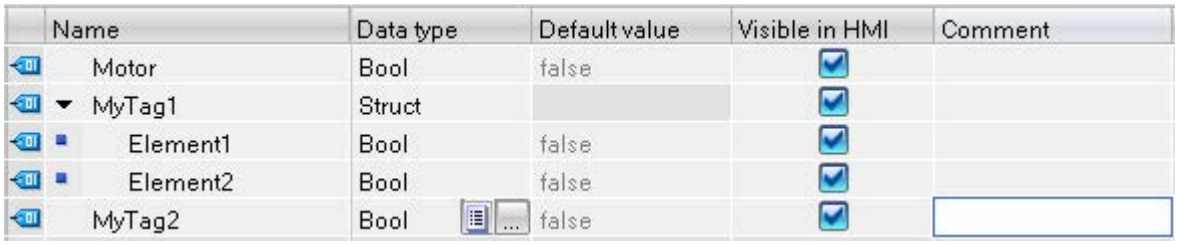
See also

Creating PLC data types (Page 2856)

Structure of the declaration table for PLC data types

Structure of the declaration table for PLC data types

The figure below shows the structure of the declaration table for PLC data types.



Meaning of the columns

The following table shows the meaning of the individual columns. You can show or hide the columns as required. The number of columns displayed varies depending on the CPU series.

Column	Explanation
	Symbol you can click to move or copy the tag.
Name	Name of the tags.
Data type	Data type of the tags.
Default value	Value with which you predefine the tag in the declaration of the PLC data type. Specification of the default value is optional. If you do not specify any value the predefined value for the indicated data type is used. For example, the value "false" is predefined for BOOL.
Visible in HMI	Shows whether the tag is visible by default in the HMI selection list.
Accessible from HMI	Shows whether HMI can access this tag during runtime.
Setting value	Setting values are the values that will probably have to be fine tuned during commissioning. After commissioning, the values of these tags can be transferred to the offline program as start values and stored there.
Comment	Comment to document the tags.

See also

Creating PLC data types (Page 2856)

Show and hide table columns (Page 2863)

Setting data values during commissioning (Page 2850)

Creating PLC data types

Requirement

The "PLC data types" folder opens in the project tree.

Procedure

To create a PLC data type, proceed as follows:

1. In the "PLC data types" folder, click the "Add new data type" command.
   A new declaration table for creating a PLC data type will be created and opened.
2. Select the PLC data type and select the "Rename" command in the shortcut menu.
3. Enter the name of the PLC data type.

Result

The new PLC data type is created. You can find the PLC data type in the project tree in the "PLC data types" folder.

See also

Structure of the declaration table for PLC data types (Page 2855)

Basics of PLC data types (Page 2854)

Delete PLC data types

Requirement

The PLC data type you want to delete is not open.

Procedure

To delete a PLC data type, follow these steps:

1. In the project tree, open the "PLC data types" folder.
2. Select the PLC data type to be deleted. You can also select several PLC data types by clicking on them one after the other while holding down the <Ctrl> key or by pressing and holding down <Shift> and clicking on the first and last data type.
3. Select the "Delete" command in the shortcut menu.

Note

If you delete a PLC data type, the blocks that use the data type will become inconsistent. These inconsistencies are marked in red in the block used. To remedy these inconsistencies, the data blocks have to be updated.

See also:

Updating the block interface (Page 2557)

Updating data blocks (Page 2830)

Programming the structure of PLC data types

Declaring tags of elementary data type

Requirement

A PLC data type is open.

Procedure

To declare a tag, follow these steps:

1. Enter a tag name in the "Name" column.
2. Enter the required data type in the "Data type" column. You will be supported by autocompletion during input.
3. Optional: Change the properties of the tags that are displayed in the other columns.
4. Repeat steps 1 to 3 for all tags that are to be declared.

See also

Editing tables (Page 316)

Editing the declaration table for PLC data types (Page 2861)

Declaring tags of the ARRAY data type

Requirement

A PLC data type is open.

Procedure

To declare a tag of the ARRAY data type, follow these steps:

1. Enter a tag name in the "Name" column.
2. Enter the "Array" data type in the "Data type" column. You will be supported by autocompletion in this step.
   The "Array" dialog opens.
3. In the "Data type" text box, specify the data type of the array elements.
4. In the "ARRAY limits" text box, specify the high and low limit for each dimension.
   Example of a one-dimensional ARRAY:
   `[0..3]`
   Example of a three-dimensional ARRAY:
   `[0..3, 0..15, 0..33]`
5. Confirm your entry.
6. Optional: Change the properties of the tags that are displayed in the other columns.

Note

You cannot define specific default values for ARRAY elements. You can, however, assign them start values at the usage point in the data block.

See also

Array (Page 2267)

Structure of the declaration table for PLC data types (Page 2855)

Using autocompletion (Page 2540)

Declaring tags of STRUCT data type

Requirements

A PLC data type is open.

Procedure

To declare a tag of the STRUCT data type, follow these steps:

1. Enter a tag name in the "Name" column.
2. Enter "Struct" in the "Data type" column. You will be supported by autocompletion during input.
   An empty, indented row is inserted after the new tag.
3. Insert the first structural element in the first empty row.
   An additional empty row is inserted after the element.
4. Select a data type for the structure element.

5. Optional: Change the properties of the structural element that is displayed in the other columns.
6. Repeat steps 3 to 5 for all additional structure elements.
   It is not necessary to end the structure explicitly. The structure ends with the last element that is entered.
7. To insert a new tag after the structure, leave a blank row after the end of the structure and then start the new tag in the second empty row.

Result

The tag of STRUCT data type is created.

See also

STRUCT (Page 2272)

Structure of the declaration table for PLC data types (Page 2855)

Using autocompletion (Page 2540)

Declaring tags based on a different PLC data type

Requirements

- A global data block is open.
- A PLC data type is declared in the current CPU.

Procedure

To declare a tag based on a different PLC data type, follow these steps:

1. Enter a tag name in the "Name" column.
2. Enter the PLC data type in the "Data type" column. You will be supported by autocompletion during input.

Result

The tag is created.

Note

You define the default values of tags within a PLC data type when the PLC data type is created. You cannot change these values at the point of use of the PLC data type.

See also

Basics of PLC data types (Page 2854)
Structure of the declaration table for PLC data types (Page 2855)
Using autocompletion (Page 2540)

Editing tag properties in PLC data types

Properties of tags in PLC data types

Properties

The following table gives an overview of tag properties in PLC data types:

Group	Property	Description
General	Name	Name of the tags.
	Data type	Data type of the tags.
	Default value	Default value of the tag in the interface of a higher-level code block or in a PLC data type. The values contained in the "Default value" column can only be changed in the higher-level code block or PLC data type. The values are only displayed in the data block.
	Start value	Not relevant in PLC data types
	Comment	Comment on the tag.
Attributes	Retain	Not relevant in PLC data types
	Visible	Indicates whether a parameter is visible in CFC.
	Configurable	Indicates whether a parameter is configurable in CFC.
	For test	Indicates whether a parameter is registered for the CFC test mode.
	Interconnectable	Indicates whether a parameter is interconnectable in CFC.

See also

Changing the properties of tags in PLC data types (Page 2861)
Basics of PLC data types (Page 2854)
Structure of the declaration table for PLC data types (Page 2855)

Changing the properties of tags in PLC data types

Editing general properties in the declaration table

To edit the general properties of one or more tags, follow these steps:

1. Open the PLC data type.
2. Change the entries in the columns.

Editing detailed properties in the properties window

To edit the detailed properties of an individual tag, follow these steps:

1. Select a tag in the table.
2. Select the "Properties" command in the shortcut menu.
   The inspector window shows the properties of the tag in the "General" and "Attributes" areas.
3. Select the required area in the area navigation.
4. Change the entries in the text boxes.

See also

Updating the block interface (Page 2557)

Updating data blocks (Page 2830)

Editing the declaration table for PLC data types

Inserting table rows

Procedure

Proceed as follows to insert a row above the selected row:

1. Select the row in front of which you want to insert a new row.
2. Click the "Insert row" button on the toolbar of the table.

Result

A new row is inserted above the selected row.

Inserting table rows

Procedure

Proceed as follows to insert a row below the selected row:

1. Select the row below which you want to insert a new row.
2. Click the "Add row" button on the table toolbar.

Result

A new empty row will be inserted below the selected row.

Deleting tags

Procedure

Follow the steps below to delete elements:

1. Select the row with the element to be deleted. You can also select several rows by clicking on them one after the other while holding down the <Ctrl> key or by pressing and holding down <Shift> and clicking on the first and last row.
2. Select the "Delete" command in the shortcut menu.

See also

Updating the block interface (Page 2557)

Updating data blocks (Page 2830)

Automatically filling in successive cells

You can load the contents of one or several table cells into the cells below, automatically filling in the successive cells.

If you automatically fill in cells in the "Name" column, a consecutive number will be appended to each name. For example, "Motor" will become "Motor_1".

You can define individual or more cells as well as entire rows as source area.

If less rows exist in the open table than you want to fill, then you will first have to insert additional empty rows.

Requirement

- The table is open.
- Sufficient declaration rows are available.

Procedure

To automatically fill in successive cells, follow these steps:

1. Select the cells to be loaded.
2. Click the "Fill" symbol in the bottom right corner of the cell.
   The mouse pointer is transformed into a crosshair.
3. Keep the mouse button pressed and drag the mouse pointer downwards over the cells that you want to fill in automatically.
4. Release the mouse button.
   The cells are filled in automatically.
5. If entries are already present in the cells that are to be automatically filled in, a dialog appears. In this dialog you can indicate whether you want to overwrite the existing entries or insert new rows for the new tags.

Show and hide table columns

You can show or hide the columns in a table as needed.

Procedure

To show or hide table columns, follow these steps:

1. Click a column header.
2. Select the "Show/Hide" command in the shortcut menu.
   The selection of available columns is displayed.
3. To show a column, select the column's check box.
4. To hide a column, clear the column's check box.

9.1.4.5 Using external source files

Basics of using external source files

Function

The textual programming languages STL and SCL allow you to enter the program code in any ASCII editor and save it as an external source file. This enables you to perform a range of tasks, for example:

- Declaring tags
- Specify block properties
- Programming blocks

You can import these source files to your project and use them to generate blocks. You can generate a number of different blocks from one source file. Observe the following special features when generating blocks from a source file:

- A block that exists under the same name in the project will be overwritten.
- Always name organization blocks according to their type, e.g., "Main" or "Cyclic interrupt" so that the event assignment is retained. If you are using multiple organization blocks of the same type, you can add underscores and numbers to the name, e.g., "Main_2" or "Main_2_1".
- If a block was programmed with its absolute block number instead of a symbolic name in the source file and this number is already assigned by a block in the project, the new generated block is initially assigned the next free symbolic name.
- If you have not explicitly defined the access mode for a block in the external source file, the block access mode is set depending on the CPU series used:
  - Blocks generated for a CPU of the S7-1200/1500 series are assigned "optimized" access mode by default.
  - Blocks generated for a CPU of the S7-300/400 product range are assigned "standard" access mode by default.

  Organization blocks are the exception in this case, as they are always assigned the "standard" access mode by default, regardless of the CPU series. You have the option of changing the block access mode manually.
- It is possible that not all comments from the source file will be applied in the block.
- If you use absolute addressing in the external source file, a symbolic tag is created for each absolute address during the generation of the block. The names of these tags are made up of "Tag_" and a time stamp. This may result in relatively long tag names, which you can change manually if required.
- If you are using instructions in a different version in the external source file than in the target device, this may result in compilation errors. Correct the respective instructions in this case and start the compilation process once again. You can also select a different version for the target device.

You also have the option of saving existing blocks as external source files.

See also

Rules for programming external source files (Page 2864)

Saving blocks as external source files (Page 2865)

Inserting external source files (Page 2866)

Opening and editing external source files (Page 2867)

Generating blocks from external source files (Page 2868)

Rules for programming external source files

An external source file basically consists of continuous text. To compile the source into blocks, certain structures and syntax rules must however be adhered to.

Syntax rules

The syntax of the instructions in external source files is very similar to that in the creation of user programs in the program editor with STL or SCL. Note, however, the following additional syntax rules:

- Block call
  When calling a block, transfer the parameters in the defined order in the ASCII editor. If you do not, the comment assignments for these lines may not match.
  Enter the parameters in brackets. The individual parameters are separated by a comma.
- Upper or lower case
  The program editor generally disregards upper or lower case. Jump labels are an exception to this. Character string entries are also case-sensitive ("STRING" data type). Keywords are displayed in upper case. For compilation purposes, however, case is disregarded; you can therefore specify keywords in upper or lower case or a mixture of the two.
- Semicolon
  Mark the end of every instruction and every tag declaration with a semicolon. You can enter several instructions per line.
- Forward slashes
  Begin every comment with two forward slashes (//) and end the comment with the <Enter> key.

See also

Basics of using external source files (Page 2863)

Saving blocks as external source files (Page 2865)

Inserting external source files (Page 2866)

Opening and editing external source files (Page 2867)

Generating blocks from external source files (Page 2868)

Saving blocks as external source files

Depending on the programming language used for the block, you have the following options for storing blocks as external source files:

- STL and SCL: copy a block as text
- SCL: generate external source file from one or more blocks

Copying a block as text

To copy a block as text and export it to an external source file, follow these steps:

1. In the project tree, right-click on the block you want to export to an external source file.
2. Select the "Copy as text" command in the shortcut menu.
3. Open an external text editor.

4. Paste the copied text from the clipboard.
5. Save the file with one of the following file name extensions:
   – ".scl" if you wish to generate a source file for SCL
   – ".stl" if you wish to generate a source file for STL

Generating an external source file from SCL blocks

To generate an external source file from SCL blocks, follow these steps:

1. In the project tree or in the overview window, select the SCL blocks from which you want to generate an external source file.
2. Select the "Generate source from blocks" command from the shortcut menu.
   The "Generate source from blocks" dialog opens.
3. Specify a path and a name for the external source.
4. Click "OK".

You can also generate an external source file from an open SCL block. To do this, follow these steps:

1. Click on the "Generate source from block" button in the programming editor.
   The "Generate source from blocks" dialog opens.
2. Specify a path and a name for the external source.
3. Click "OK".

Result

The block has been saved as an external source file. You can include this source file in a project in the TIA portal and use it to generate other blocks. However, please note that you can use STL source files only in S7-300/400/1500 CPUs.

See also

Basics of using external source files (Page 2863)

Rules for programming external source files (Page 2864)

Inserting external source files (Page 2866)

Opening and editing external source files (Page 2867)

Generating blocks from external source files (Page 2868)

Inserting external source files

Requirement

- An external source file is available and complies with the syntax and structure rules.
- The "External source files" folder is open in the project tree.

Procedure

Follow these steps to insert an external source file:

1. Double-click on the "Add new external file" command.
   The "Open" dialog box is opened.
2. Navigate to and select existing external source files.
3. Confirm your selection with "Open".

Result

The new source file will be added to the "External source files" folder.

See also

Basics of using external source files (Page 2863)

Rules for programming external source files (Page 2864)

Saving blocks as external source files (Page 2865)

Opening and editing external source files (Page 2867)

Generating blocks from external source files (Page 2868)

Opening and editing external source files

By linking the files with the file name extensions "stl" and "scl" to an editor you will be able to open and edit external source files with these formats directly. Use Notepad as editor because other text editors may not let you open several sources at the same time.

This means you do not need to insert the external source files again after editing.

Linking files with the file name extensions "stl" and "scl" file types to an editor

Proceed as follows to link files with the file name extensions "stl" and "scl" to an editor:

1. Open Windows Explorer.
2. Right-click on an STL file.
3. Select "Properties" in the shortcut menu.
   The "Properties" dialog box opens.
4. Click "Change" in the "File type" area on the "General" tab.
   The "Open with" dialog box opens.
5. Select the text editor you want to link to the "stl" file type.
6. Confirm your selection with "OK".
7. Close the "Properties" dialog with "OK".
8. Repeat steps 2 to 7 with an SCL file.

Opening and editing an external source file

To open an external source file, follow these steps:

1. Open the "External source files" folder in the project tree.
2. Double-click on the external source file you want to open.
   The external source file will open in the linked editor and can be edited.

See also

Basics of using external source files (Page 2863)

Rules for programming external source files (Page 2864)

Saving blocks as external source files (Page 2865)

Inserting external source files (Page 2866)

Generating blocks from external source files (Page 2868)

Generating blocks from external source files

Requirement

- The "External source files" folder is open in the project tree.
- An external source file is available.

Procedure

To generate blocks from an external source file, follow these steps:

1. Open the external source file from which you wish to generate blocks.
2. Select the "Generate blocks from source" command in the "Edit" menu.
3. A prompt will appear telling you any existing blocks will be overwritten.
4. Confirm the safety prompt with "Yes".

Result

The external source file blocks will be generated and inserted in the "Program blocks" folder in the project tree. In the event of errors, information about the errors which have occurred will be displayed in the inspector window. This information, however, relates to the external source file and not to the block generated.

See also

Specifying block properties (Page 2512)

Basics of using external source files (Page 2863)

Rules for programming external source files (Page 2864)

Saving blocks as external source files (Page 2865)
Inserting external source files (Page 2866)
Opening and editing external source files (Page 2867)

9.1.5 Comparing PLC programs

9.1.5.1 Basic information on comparing PLC programs

Introduction to comparing PLC programs

Function

You can compare the following objects of a PLC program in order to detect any differences:

- Code blocks with other code blocks
- Data blocks with other data blocks
- PLC tags of a PLC tag table with the PLC tags of another PLC tag table
- PLC data types with other PLC data types

Types and levels of comparison

Two different basic types of comparison can be used:

- Online/offline comparison:
  The objects in the project are compared with the objects of the corresponding device. An online connection to the device is necessary for this comparison.
- Offline/offline comparison:
  The objects of two devices either within a project or from different projects or libraries are compared. No online connection is required for this comparison.

Please note that you cannot carry out an unlimited number of comparisons at the same time, but only one comparison per comparison type (online/offline or offline/offline).

You can choose between the following levels of comparison depending on how in-depth an object comparison you require:

- Compare editor
- Detailed comparison

When you start a comparison, you will first receive an overview in the compare editor. For some comparison objects, you can then start a detailed comparison in which the objects compared will be opened side-by-side, each in its own program editor instance. Any differences will be highlighted.

The table below gives an overview of the types and levels of comparison you can apply for each object:

Object	Online/offline		Offline/offline	
Compare editor	Detailed comparison	Compare editor	Detailed comparison	
LAD block	X	X	X	X
FBD block	X	X	X	X
STL block[1]	X	X	X	X
SCL block	X	X[3 4]	X	X
GRAPH block[2]	X	X[4]	X	X[5]
Global data block	X	X	X	X
Instance data block	X	X	X	X
PLC tags	-	-	X	X
PLC data type	X[4]	X[4]	X	X

Legend:

X: available
-: not available

[1]: STL is not available for S7-1200

[2]: GRAPH is not available for S7-1200

[3]: not for S7-1200 prior to version 2.0

[4]: not for S7-300/400

[5]: only comparable with a GRAPH block from the same CPU family

Note

Please note the following:

- You cannot perform a detailed comparison for know-how protected blocks.
- If the detail comparison detects differences only with respect to the data types of local tags, with offline being an interrupt data type (C_ALARM C_ALARM_S C_ALARM_8 C_ALARM_8P C_ALARM_T C_AR_SEND C_NOTIFY C_NOTIFY_8P) and online a DWORD, this difference is not marked as such.
- You cannot run a detailed comparison for types and master copies from libraries.

See also

Basics of project data comparison (Page 363)

Comparison of code blocks (Page 2871)

Comparison of data blocks (Page 2872)

Comparing PLC tags and PLC data types (Page 2872)

Carrying out an online/offline comparison (Page 364)

Carrying out offline/offline comparisons (Page 364)

Using the compare editor (Page 365)

Performing detailed block comparisons (Page 2875)

Comparison of code blocks

Introduction

The blocks to be compared in a code block comparison are assigned for comparison on the basis of the following criteria:

- Online/offline comparison: Addresses, e.g. FB100
- Offline/offline comparison: Symbolic names of the blocks

The comparison involves an evaluation of the block time stamps. The results are displayed as an overview in the comparison editor. You can then use actions to define what is to be done about the differences. You can also start detailed comparisons for the individual blocks. The versions of a block compared are opened beside each other and the differences are highlighted.

For the comparison of code blocks, both the block interfaces and the individual networks are compared. Any differing tag names are also determined. All comments and other block attributes are excluded from an online/offline comparison.

If the block interface changes, the time stamp of the code block interface will also change. This change means a change in the time stamp of the program code. The first step in comparing block interfaces is therefore a comparison of the program code time stamps. If these time stamps are the same, it is assumed that the interfaces are the same. If the time stamps of the interfaces differ, the next step is to compare the data types of the interfaces, section by section. Multiple instances and PLC data types are included in the comparison. If the data types in the sections are the same, the start values of the tags are compared. All differences are displayed.

When networks are compared, first inserted or deleted networks are detected. Then the other networks are compared. Instructions are the same if the operator and operand are the same. The first difference in each instruction is displayed. However, several differences per network can be displayed.

See also

Introduction to comparing PLC programs (Page 2869)

Comparison of data blocks (Page 2872)

Comparing PLC tags and PLC data types (Page 2872)

Carrying out an online/offline comparison (Page 364)

Carrying out offline/offline comparisons (Page 364)

Using the compare editor (Page 365)

Performing detailed block comparisons (Page 2875)

Comparison of data blocks

Introduction

The blocks to be compared in a data block comparison are assigned for comparison on the basis of the following criteria:

- Online/offline comparison: Addresses, e.g. DB100
- Offline/offline comparison: Symbolic names of the blocks

The first step in data block comparison is comparing the time stamps of the data block. If these time stamps are the same, it is assumed that the data structures are the same. If the time stamps differ, the structures are then compared until the first difference is found. If the data structures in the sections are the same, the initial and current values of the tags are then compared. All differences are displayed. Any differing tag names are also determined. Comments and PLC data type structures used in the data block are not included in the comparison.

See also

Introduction to comparing PLC programs (Page 2869)

Comparison of code blocks (Page 2871)

Comparing PLC tags and PLC data types (Page 2872)

Carrying out an online/offline comparison (Page 364)

Carrying out offline/offline comparisons (Page 364)

Using the compare editor (Page 365)

Performing detailed block comparisons (Page 2875)

Comparing PLC tags and PLC data types

Introduction

The device PLC tag tables and the device PLC data types will also be shown in the comparison editor if you carry out an offline/offline comparison. The PLC tag tables and the PLC data types will be matched by name and you will receive the following information:

- Status: A symbol shows whether the PLC tags/PLC data types are identical or differ.
- Missing PLC tag tables / PLC data types: You can see at a glance whether the PLC tag tables / PLC data types are available in both devices.

You obtain the following information with an online/offline comparison of CPUs of the S7-1200/1500 series:

- PLC tags: A symbol shows whether the PLC tags are identical or differ. Because PLC tag tables are not downloaded to the device during loading, they cannot be displayed during an online/offline comparison.
- PLC data types: You receive the status symbol for each PLC data type. You can see at a glance whether the PLC data types are available in both devices.

See also

Introduction to comparing PLC programs (Page 2869)
Comparison of code blocks (Page 2871)
Comparison of data blocks (Page 2872)
Carrying out an online/offline comparison (Page 364)
Carrying out offline/offline comparisons (Page 364)
Using the compare editor (Page 365)
Performing detailed block comparisons (Page 2875)

9.1.5.2 Comparing blocks

Comparing blocks in the compare editor

You have the following options for comparing blocks in the compare editor:

- Online/offline comparison
  The blocks in the project are compared with the blocks of the selected device.
- Automatic offline/offline comparison
  All blocks of the selected devices are compared offline.
- Manual offline/offline comparison
  The selected blocks of the devices are compared offline.

Carrying out an online/offline comparison of blocks

To perform an online/offline comparison, follow these steps:

1. In the project tree, select a device that allows online/offline comparison.
2. Select the "Compare > Offline/online" command in the shortcut menu.
3. If you have not already established an online connection to this device, the "Go online" dialog opens. In this case, set all the necessary parameters for the connection and click "Connect".
   The online connection is established and the compare editor opens.
4. Open the "Program blocks" folder.
   You can identify the status based on the symbols in the status and action area. You can define certain actions depending on the status of the objects. Note, however, that you can only perform actions in one direction in a synchronization action.

Carrying out an automatic offline/offline comparison of blocks

To perform an automatic offline/offline comparison of blocks, follow these steps:

1. Select a device in the project tree that allows offline/offline comparison.
2. Select the "Compare > Offline/offline" command in the shortcut menu.
   The compare editor opens and the selected device is displayed in the left area.

3. Drag-and-drop an additional device to the drop area of the right pane. The device to be compared can originate from the same project, a reference project or the library.
4. Open the "Program blocks" folder.
   You can identify the status of the objects based on the symbols in the status and action area. You can define certain actions depending on the status of the objects. When you select an object, the object's properties and the corresponding object of the assigned device are clearly shown in the properties comparison.

You can drag any other device to the drop area at any time to perform further comparisons.

Carrying out a manual offline/offline comparison of blocks

To perform a manual offline/offline comparison of blocks, follow these steps:

1. Select a device in the project tree that allows offline/offline comparison.
2. Select the "Compare > Offline/offline" command in the shortcut menu.
   The compare editor opens and the selected device is displayed in the left area.
3. Drag-and-drop an additional device to the drop area of the right pane. The device to be compared can originate from the same project, a reference project or the library.
4. In the status and action area, click on the button for switching between automatic and manual comparison.
5. Select the objects that you want to compare.
   The properties comparison is displayed. You can identify the status of the objects based on the symbols. You can define certain actions depending on the status of the objects.

You can drag any other device to the drop area at any time to perform further comparisons.

See also

Introduction to comparing PLC programs (Page 2869)

Using the compare editor (Page 365)

Performing detailed block comparisons (Page 2875)

Comparing PLC tags (Page 2895)

Comparing PLC data types (Page 2896)

Performing detailed block comparisons

Starting a detailed comparison

You can start a detailed comparison for blocks. The versions of a block compared are opened beside each other and the differences highlighted.

Note

Please note the following:

- For blocks that are created in the programming language SCL, the detail comparison is not available for S7-1200 series CPUs with a version older than 2.0.
- You can only start the detailed comparison with the compare editor for blocks created in the GRAPH programming language.

Starting detailed comparisons using the compare editor

To start a detailed comparison for a block using the compare editor, follow these steps:

1. First, perform an online/offline or an offline/offline comparison.
   The compare editor opens.
2. In the compare editor, select the block for which you want to perform a detailed comparison.
3. Click the "Start detailed comparison" button in the toolbar.

Starting detailed comparisons in the program editor

For the comparison type online/offline, you can start the detailed comparison directly in the programming editor. To do this, follow these steps:

1. Open the block for which you wish to carry out a detailed comparison.
2. Establish an online connection.
   See also: Go online and Go offline

   Note

   Please note that the block must be available online in order for you to be able to start the detailed comparison for the block within the programming editor.

3. Click the "Detailed comparison" button in the toolbar.
4. Confirm the dialog for closing the block with "Yes".

Result

One instance of the program editor will be opened for each version of the block compared and the two instances are displayed side by side. Any differences will be highlighted.

If there are structural differences for GRAPH blocks, the comparison results are only displayed up to the first structural difference in the sequence view.

See also

Basic information on comparing PLC programs (Page 2869)

Carrying out offline/offline comparisons (Page 364)

Carrying out an online/offline comparison (Page 364)

Using the compare editor (Page 365)

Visualization of the comparison result for LAD/FBD (Page 2876)

Navigating in the detailed comparison (Page 2892)

Changing blocks during detailed comparison (Page 2893)

Updating comparison results (Page 2894)

Visualization of the comparison result

Visualization of the comparison result for LAD/FBD

Introduction

The detailed comparison allows you to identify the exact places where versions of a block differ. The following color coding allows you to find these places as quickly as possible:

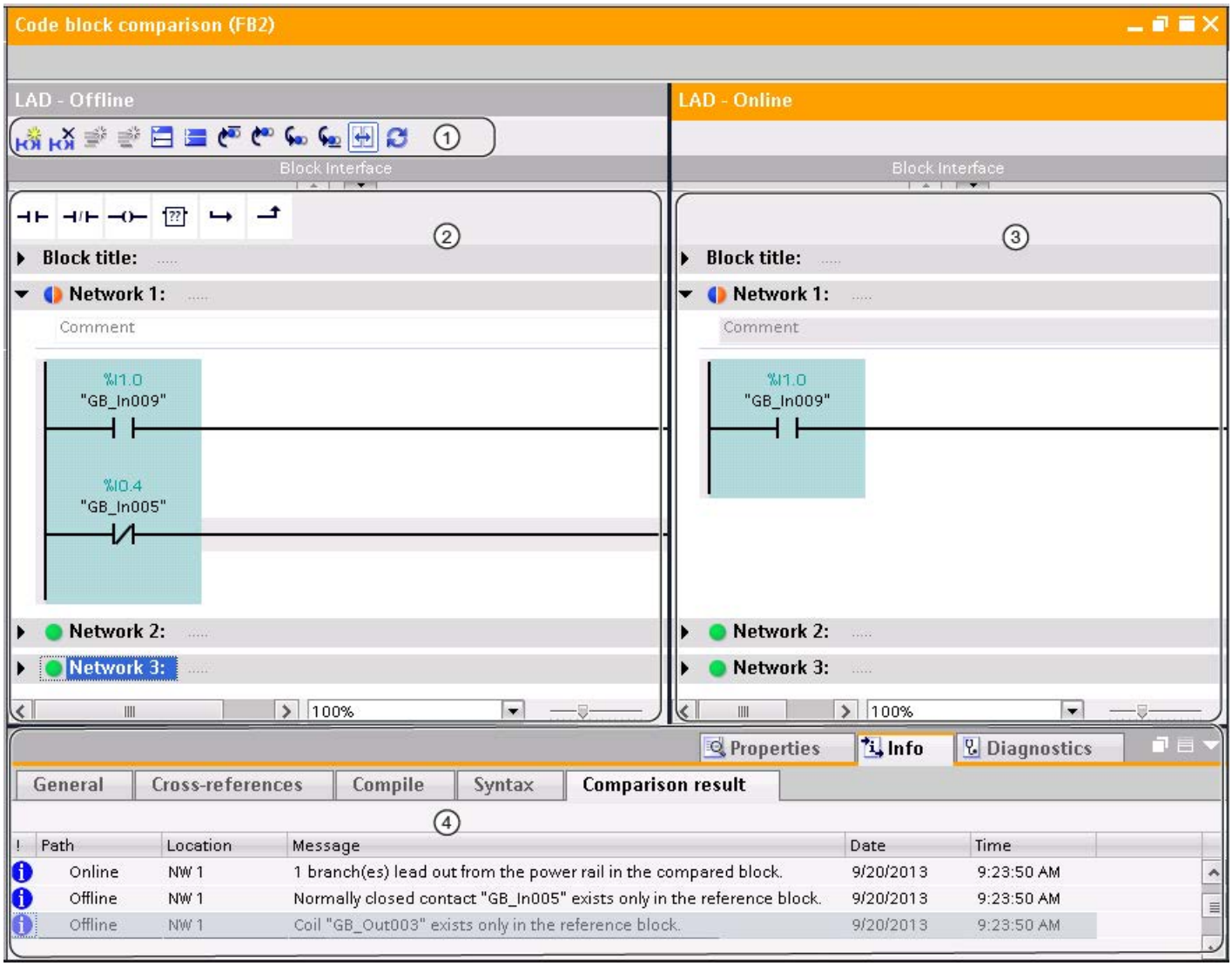
- The lines where there are differences are highlighted in gray.
- Differing operands and instructions are highlighted in green.
- If the number of networks differs, pseudo-networks are added to allow the display of identical networks to be synchronized. These pseudo-networks are highlighted in gray and contain the text "No corresponding network found" in the title bar of the network. Pseudo-networks cannot be edited.

For the sake of clarity, not all the differences are highlighted but rather the first difference of an operation in each case. For example, if all the inputs in an instruction with multiple inputs are different in the offline and online versions of the block, only the first input is highlighted as a difference. You can resolve this difference and update the comparison list. The next input will then be highlighted as a difference.

The number of differences highlighted within a network therefore depends on the number of instructions.

Structure of the detailed comparison

The following figure shows an example of the online/offline detailed comparison for the LAD programming language:



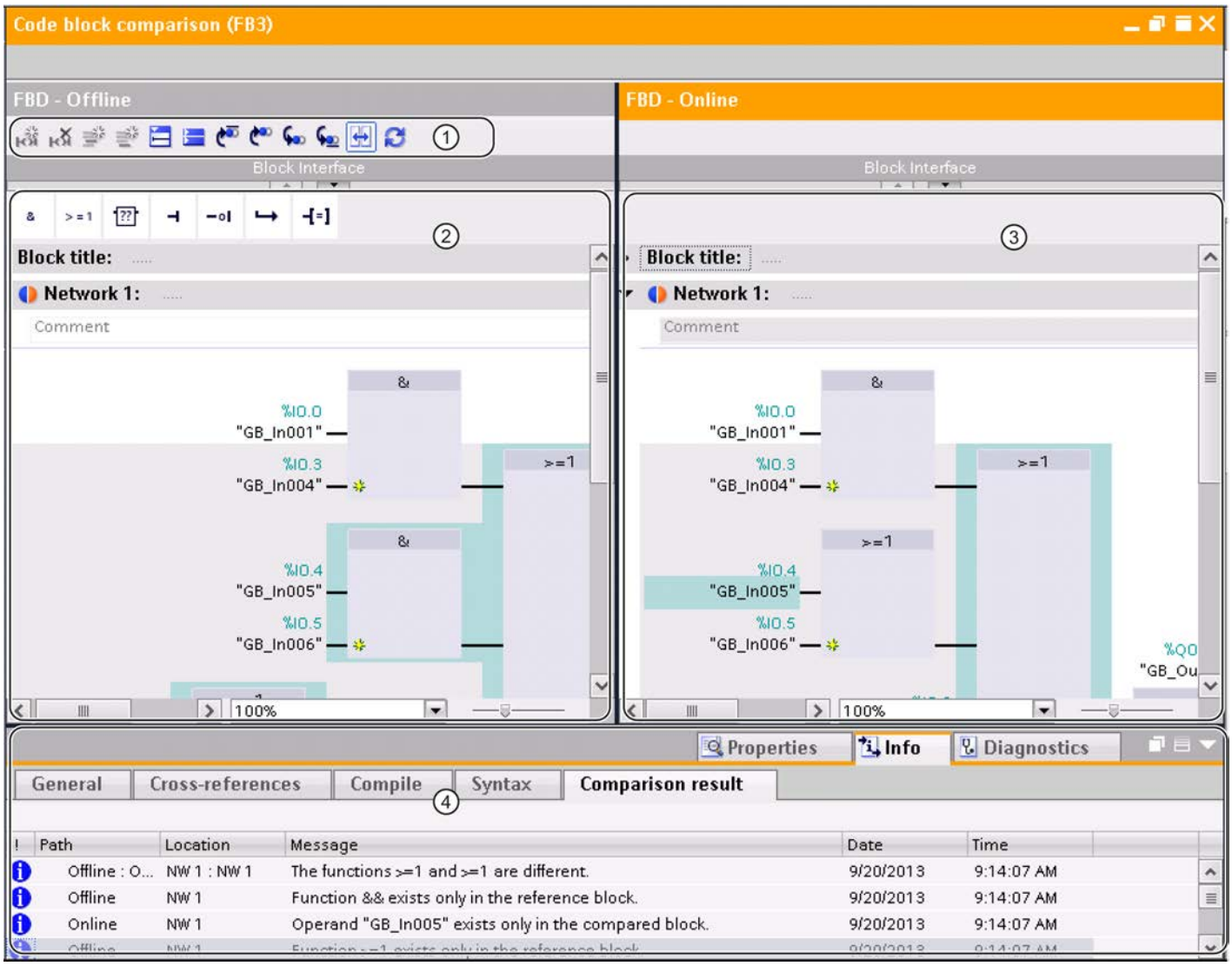
① Toolbar of the detailed comparison for LAD

② Reference block

③ Compared block

④ Comparison result in the Inspector window

The following figure shows an example of the online/offline detailed comparison for the FBD programming language:



① Toolbar of the detailed comparison for FBD

② Reference block

③ Compared block

④ Comparison result in the Inspector window

Note

Display of the symbolic names of the online version of the block is only possible for S7-1200 and S7-1500.

Toolbar of the detailed comparison

With the toolbar, you can access the following functions:

- General functions
  - Insert network
  - Delete network
  - Insert row
  - Add row
  - Open all networks
  - Close all networks
- Comparison-specific functions
  - Position on first difference
  - Position on previous difference
  - Position on next difference
  - Position on last difference
  - Synchronize scrolling between editors
  - Update comparison results

Reference block

The reference block is displayed in the left window. In an online/offline comparison, the reference block is the offline version of the block.

Compared block

The compared block is displayed in the right window. In an online/offline comparison, the compared block is the online version of the block.

Comparison result in the Inspector window

The differences are displayed in the form of a table in the "Info > Comparison result" tab of the Inspector window. Double-click on a row to navigate to the corresponding difference in the block.

See also

Basic information on comparing PLC programs (Page 2869)

Carrying out an online/offline comparison (Page 364)

Carrying out offline/offline comparisons (Page 364)

Using the compare editor (Page 365)

Starting a detailed comparison (Page 2875)

Navigating in the detailed comparison (Page 2892)
Changing blocks during detailed comparison (Page 2893)
Updating comparison results (Page 2894)

Visualization of the comparison result for STL

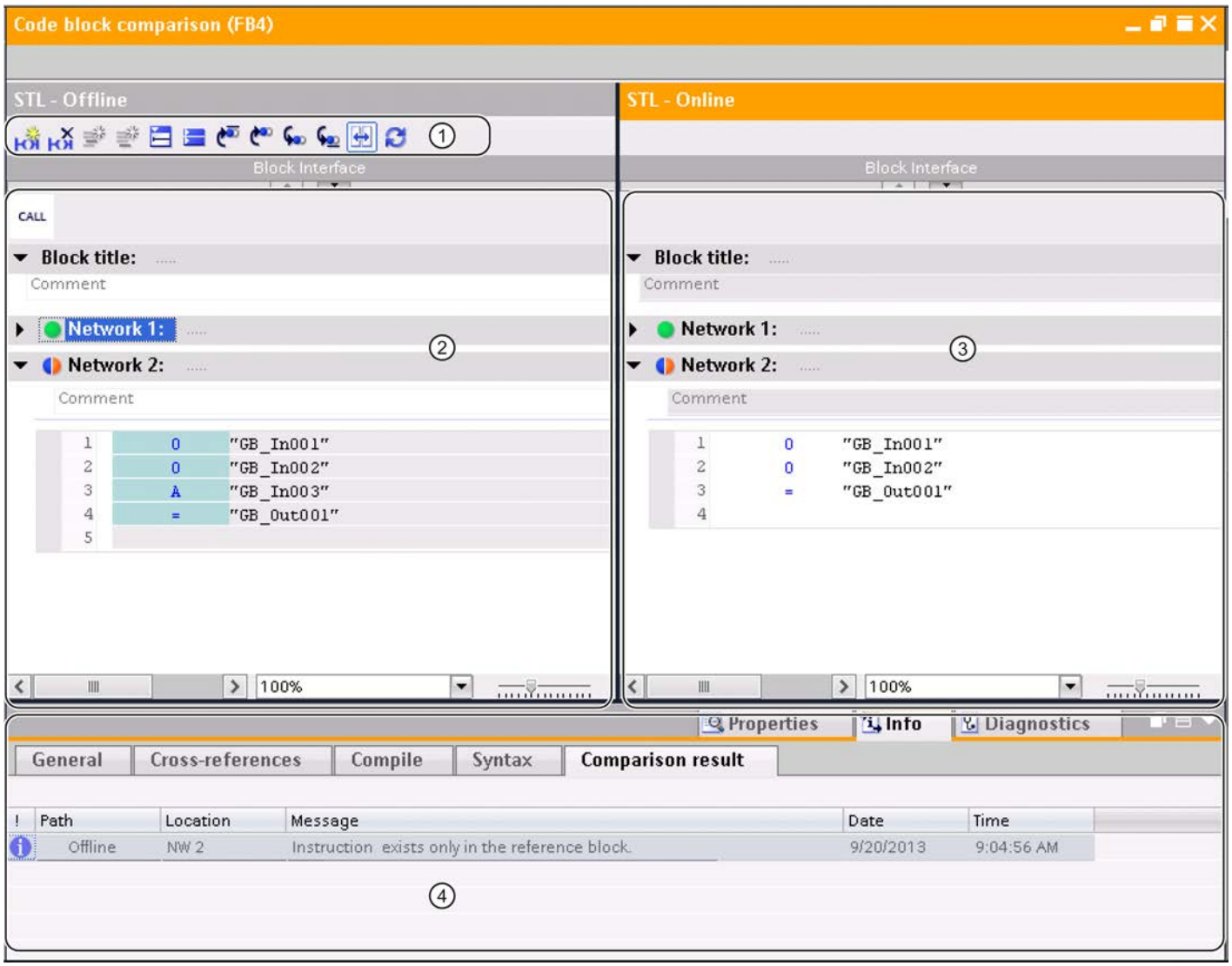
Introduction

The detailed comparison allows you to identify the exact places where versions of a block differ. The following color coding allows you to find these places as quickly as possible:

- The lines where there are differences are highlighted in gray.
- Differing operands and instructions are highlighted in green.
- If the number of networks differs, pseudo-networks are added to allow the display of identical networks to be synchronized. These pseudo-networks are highlighted in gray and contain the text "No corresponding network found" in the title bar of the network. Pseudo-networks cannot be edited.

Structure of the detailed comparison

The following figure shows an example of the online/offline detailed comparison for the STL programming language:



① Toolbar of the detailed comparison for STL

② Reference block

③ Compared block

④ Comparison result in the Inspector window

Note

The display of the symbolic names of the online version of the block is only possible for S7-1500.

Toolbar of the detailed comparison

With the toolbar, you can access the following functions:

- General functions
  - Insert network
  - Delete network
  - Insert row
  - Add row
  - Open all networks
  - Close all networks
- Comparison-specific functions
  - Position on first difference
  - Position on previous difference
  - Position on next difference
  - Position on last difference
  - Synchronize scrolling between editors
  - Update comparison results

Reference block

The reference block is displayed in the left window. In an online/offline comparison, the reference block is the offline version of the block.

Compared block

The compared block is displayed in the right window. In an online/offline comparison, the compared block is the online version of the block.

Comparison result in the Inspector window

The differences are displayed in the form of a table in the "Info > Comparison result" tab of the Inspector window. Double-click on a row to navigate to the corresponding difference in the block.

See also

Basic information on comparing PLC programs (Page 2869)

Carrying out an online/offline comparison (Page 364)

Carrying out offline/offline comparisons (Page 364)

Using the compare editor (Page 365)

Starting a detailed comparison (Page 2875)

Navigating in the detailed comparison (Page 2892)
Changing blocks during detailed comparison (Page 2893)
Updating comparison results (Page 2894)

Visualization of the comparison result for SCL

Introduction

The detailed comparison allows you to identify the exact places where versions of a block differ. The following color coding allows you to find these places as quickly as possible:

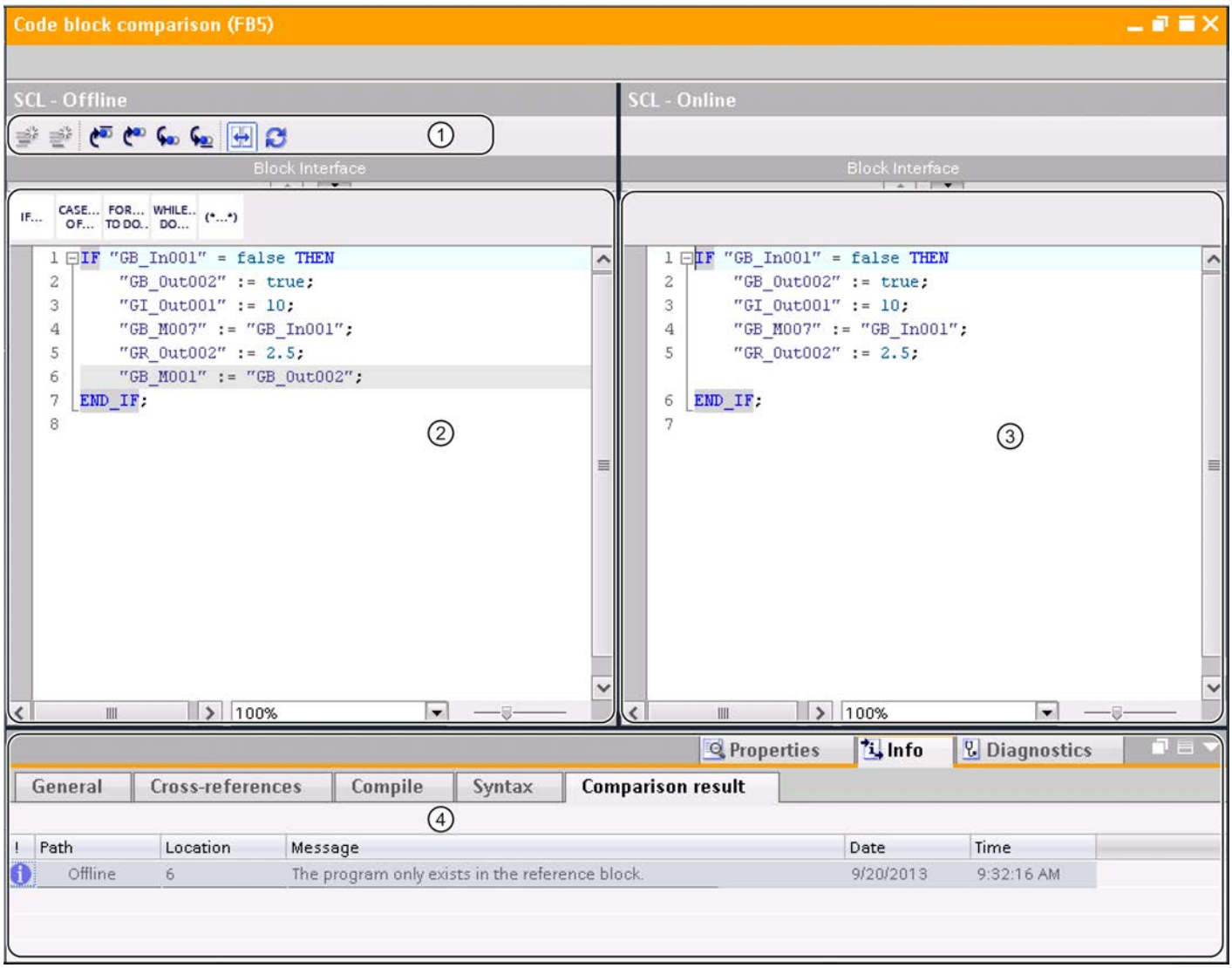
- The lines where there are differences are highlighted in gray.
- Differing operands and instructions are highlighted in green.

Note

The online/offline detailed comparison is not available for the CPU families S7-300/400 and for S7-1200 with a version less than 2.0.

Structure of the detailed comparison

The following figure shows an example of the online/offline detailed comparison for the SCL programming language:



① Toolbar of the detailed comparison for SCL

② Reference block

③ Compared block

④ Comparison result in the Inspector window

Note

The display of the symbolic name of the online version of the block is only possible for S7-1200 and S7-1500.

Toolbar of the detailed comparison

With the toolbar, you can access the following functions:

- General functions
  - Insert row
  - Add row
- Comparison-specific functions
  - Position on first difference
  - Position on previous difference
  - Position on next difference
  - Position on last difference
  - Synchronize scrolling between editors
  - Update comparison results

Reference block

The reference block is displayed in the left window. In an online/offline comparison, the reference block is the offline version of the block.

Compared block

The compared block is displayed in the right window. In an online/offline comparison, the compared block is the online version of the block.

Comparison result in the Inspector window

The differences are displayed in the form of a table in the "Info > Comparison result" tab of the Inspector window. Double-click on a row to navigate to the corresponding difference in the block.

See also

Basic information on comparing PLC programs (Page 2869)

Carrying out an online/offline comparison (Page 364)

Carrying out offline/offline comparisons (Page 364)

Using the compare editor (Page 365)

Starting a detailed comparison (Page 2875)

Navigating in the detailed comparison (Page 2892)

Changing blocks during detailed comparison (Page 2893)

Updating comparison results (Page 2894)

Visualization of the comparison result for GRAPH

Introduction

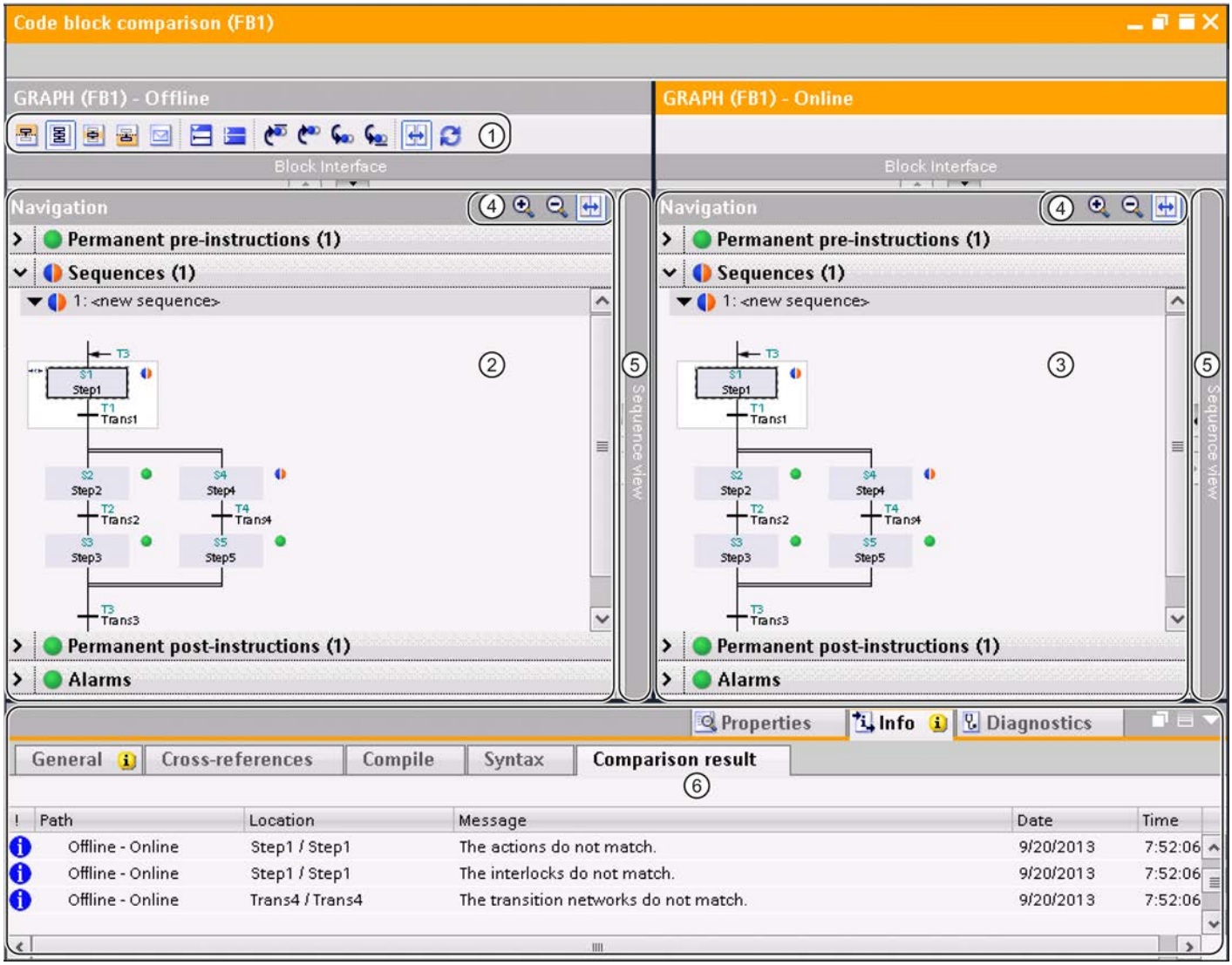
The detailed comparison allows you to identify the exact places where versions of a block differ. When you start a detailed comparison for a GRAPH block, navigation is opened first. Use the dividers to toggle between the navigation and the currently set view. You can select other views with the toolbar of the detailed comparison.

The result of the comparison is indicated by the comparison symbols.

See also: Overview of the comparison editor (Page 365)

Structure of the detailed comparison

The following figure shows an example for the navigation view with an online/offline detailed comparison for the GRAPH programming language:



① Toolbar of the detailed comparison for GRAPH

② Reference block

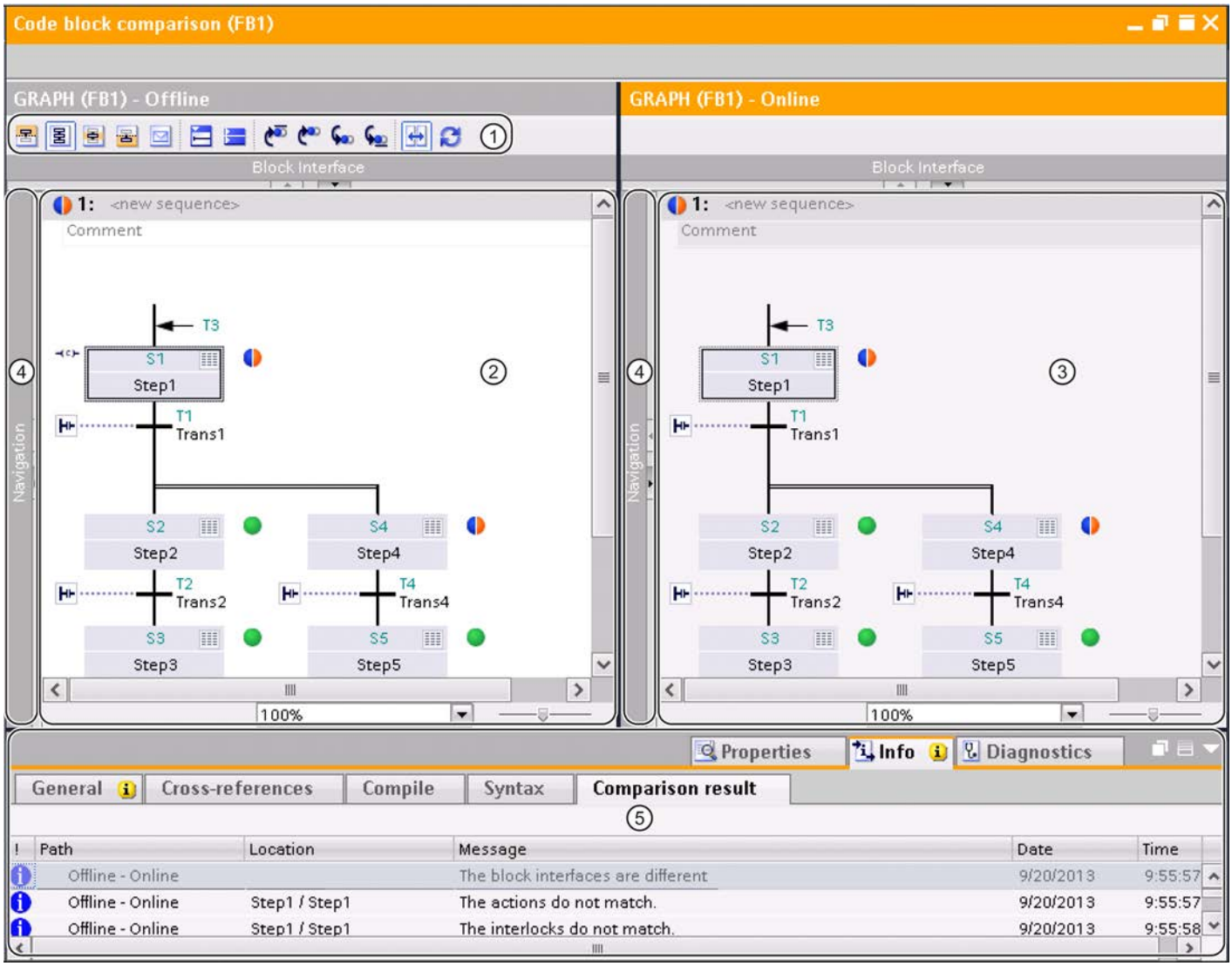
③ Compared block

④ Navigation toolbar

⑤ Dividers

⑥ Comparison result in the Inspector window

The following figure shows an example for the sequence view with an online/offline detailed comparison for the GRAPH programming language:



① Toolbar of the detailed comparison for GRAPH

② Reference block

③ Compared block

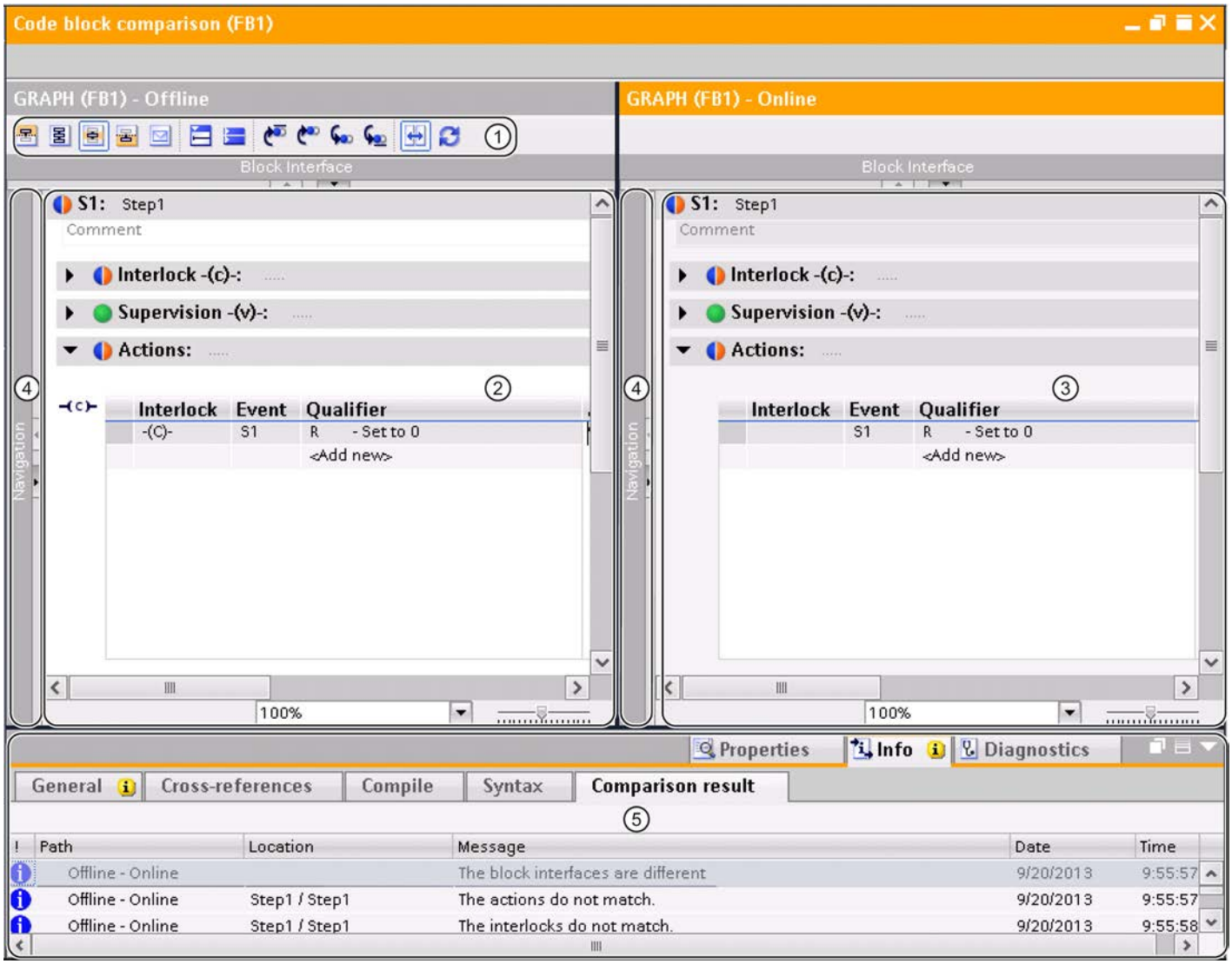
④ Dividers

⑤ Comparison result in the Inspector window

Note

If there are structural differences between the blocks, the comparison results are only displayed up to the first structural difference in the sequence view.

The following figure shows an example for the single step view with an online/offline detailed comparison for the GRAPH programming language:



① Toolbar of the detailed comparison for GRAPH

② Reference block

③ Compared block

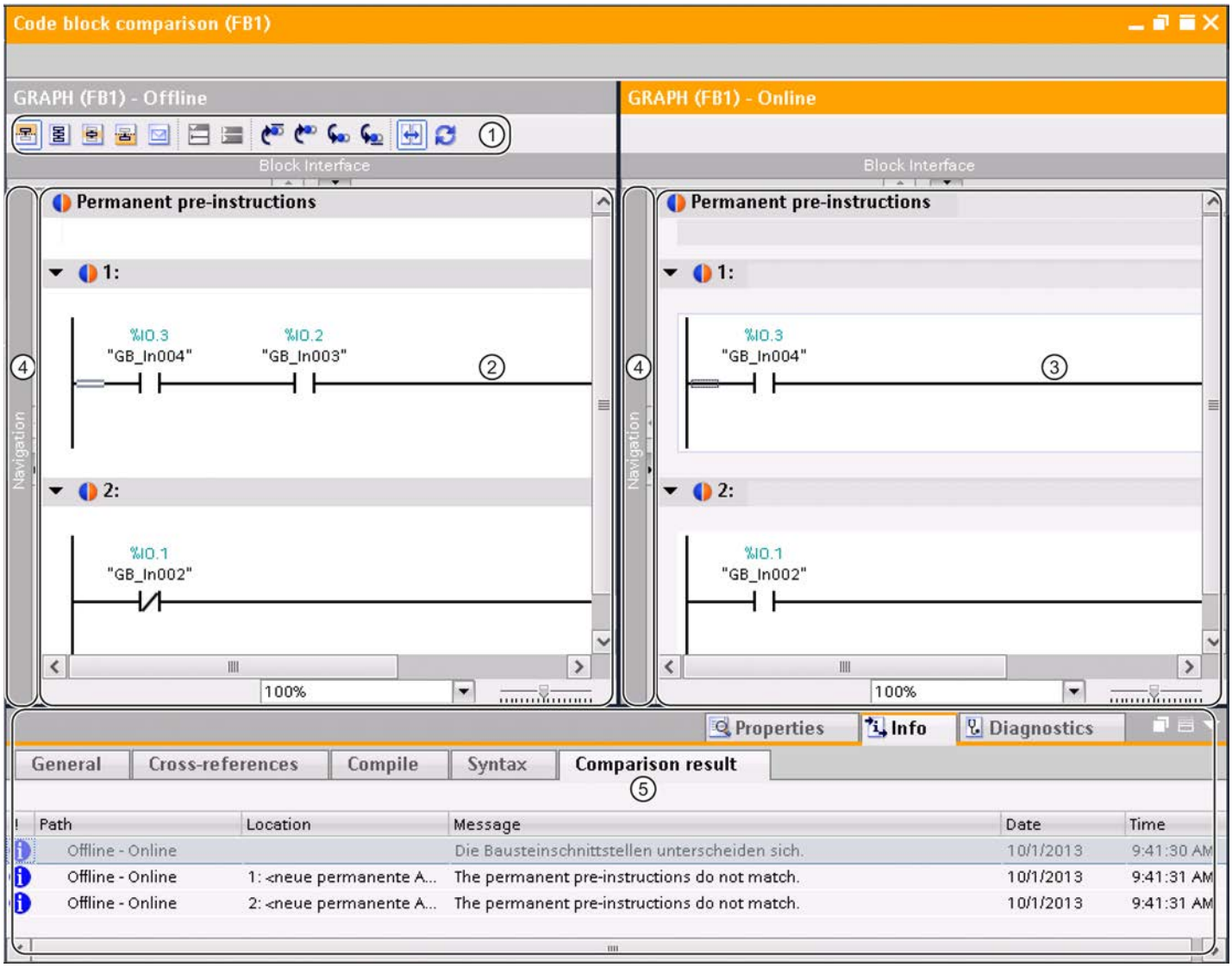
④ Dividers

⑤ Comparison result in the Inspector window

Note

The result of the comparison refers to the complete network Differences within the networks are not identified.

The following figure shows an example for the view of permanent instructions with an online/offline detailed comparison for the GRAPH programming language:



① Toolbar of the detailed comparison for GRAPH

② Reference block

③ Compared block

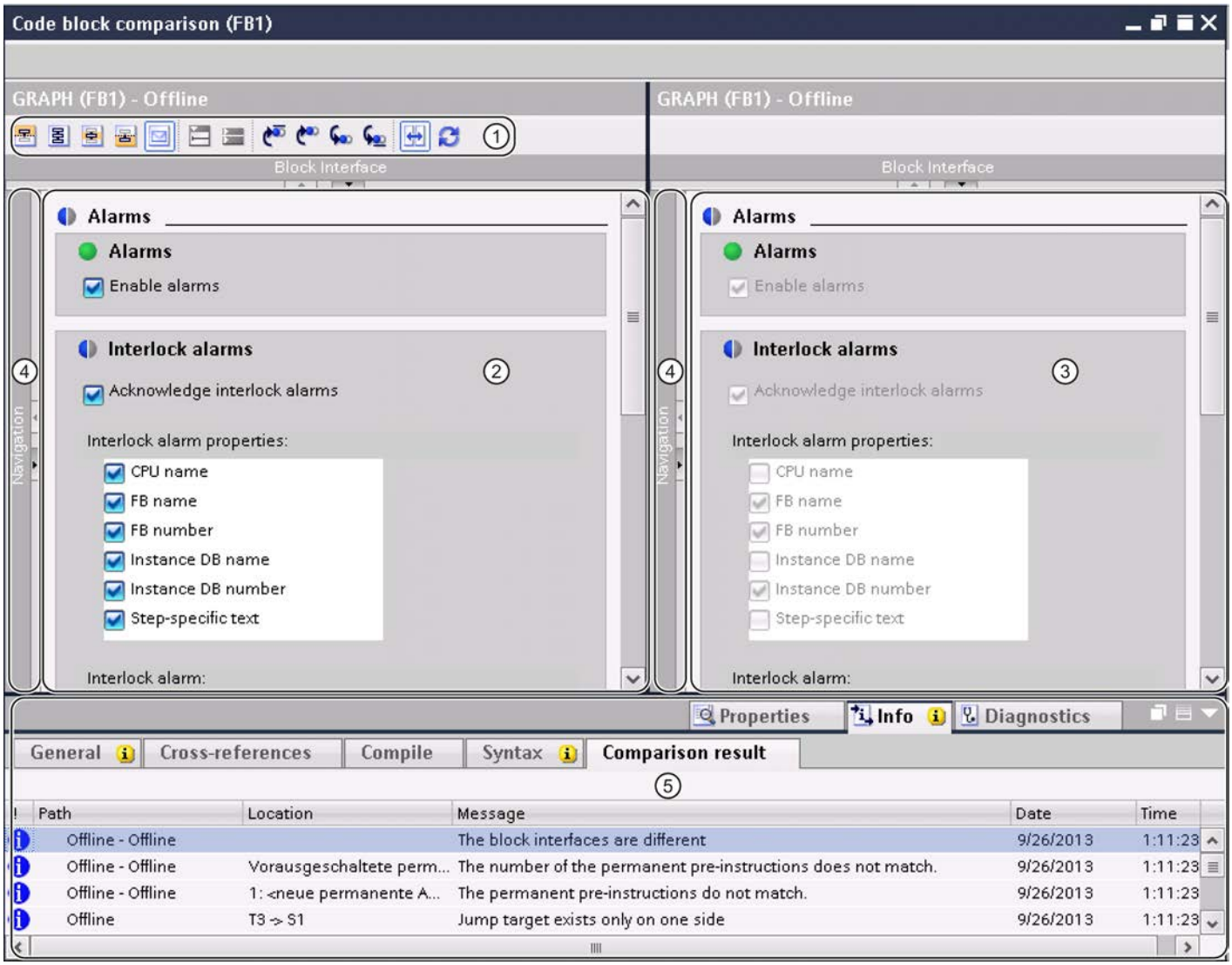
④ Dividers

⑤ Comparison result in the Inspector window

Note

The result of the comparison refers to the complete network Differences within the networks are not identified.

The following figure shows an example for the alarm view with an online/offline detailed comparison for the GRAPH programming language:



① Toolbar of the detailed comparison for GRAPH

② Reference block

③ Compared block

④ Dividers

⑤ Comparison result in the Inspector window

Toolbars

With the toolbar of the detailed comparison, you can access the following functions:

- General functions
  - Change to permanent pre-instructions
  - Change to sequence view
  - Change to single step view
  - Change to permanent post-instructions
  - Change to alarm view
  - Open all networks
  - Close all networks
- Comparison-specific functions
  - Position on first difference
  - Position on previous difference
  - Position on next difference
  - Position on last difference
  - Synchronize scrolling between editors
  - Update comparison results

The navigation has its own toolbar with the following functions:

- Zoom in or zoom out of elements within the navigation
- Synchronize navigation

Reference block

The reference block is displayed in the left window. In an online/offline comparison, the reference block is the offline version of the block.

Compared block

The compared block is displayed in the right window. In an online/offline comparison, the compared block is the online version of the block.

Dividers

You can click the dividers to toggle quickly between the navigation and the current view.

Comparison result in the Inspector window

The differences are displayed in the form of a table in the "Info > Comparison result" tab of the Inspector window. Double-click on a row to navigate to the corresponding difference in the block.

See also

Basic information on comparing PLC programs (Page 2869)
Carrying out an online/offline comparison (Page 364)
Carrying out offline/offline comparisons (Page 364)
Using the compare editor (Page 365)
Starting a detailed comparison (Page 2875)
Navigating in the detailed comparison (Page 2892)
Changing blocks during detailed comparison (Page 2893)
Updating comparison results (Page 2894)

Navigating in the detailed comparison

Requirement

You have run a detailed comparison.

Navigate to the differences

To navigate to a difference between the two blocks, follow these steps:

1. Open the list of results for the detailed comparison under "Info > Comparison result" in the Inspector window.
2. Double-click a difference.
   The difference is selected in both editors.

Or:

1. Click one of the following navigation buttons on the toolbar:
   - Position on first difference
     Navigates to the first difference in the block, and displays the difference in both editors.
   - Position on previous difference
     Navigates to the previous difference starting from the current position, and displays the difference in both editors.
   - Position on next difference
     Navigates to the next difference starting from the current position, and displays the difference in both editors.
   - Position on last difference
     Navigates to the last difference in the block, and displays the difference in both editors.

Switching off/on the synchronization of the vertical scrolling between the editors

The scrolling for both editors is synchronized to ensure that the corresponding networks are visible parallel to each other during vertical scrolling. You can switch this mode off and on. To do this, follow these steps:

1. To switch off synchronized scrolling, click the "Synchronize scrolling between editors" button in the toolbar.
2. To switch on synchronized scrolling again, click the "Synchronize scrolling between editors" button one more time in the toolbar.

See also

Basic information on comparing PLC programs (Page 2869)

Carrying out an online/offline comparison (Page 364)

Carrying out offline/offline comparisons (Page 364)

Using the compare editor (Page 365)

Starting a detailed comparison (Page 2875)

Visualization of the comparison result for LAD/FBD (Page 2876)

Changing blocks during detailed comparison (Page 2893)

Updating comparison results (Page 2894)

Changing blocks during detailed comparison

Changing offline blocks

You can change offline blocks that serve as a reference block of the comparison at any time. The compared block, on the other hand, is write protected and cannot be changed.

Changing online blocks

You cannot change online blocks.

See also

Basic information on comparing PLC programs (Page 2869)
Carrying out an online/offline comparison (Page 364)
Carrying out offline/offline comparisons (Page 364)
Using the compare editor (Page 365)
Starting a detailed comparison (Page 2875)
Visualization of the comparison result for LAD/FBD (Page 2876)
Navigating in the detailed comparison (Page 2892)
Updating comparison results (Page 2894)

Updating comparison results

As soon as you change an object, the comparison results are no longer valid and must be updated.

Requirement

You have run a detailed comparison.

Procedure

To update the comparison results, follow these steps:

1. Click "Update the comparison result" in the toolbar.

See also

Basic information on comparing PLC programs (Page 2869)
Carrying out an online/offline comparison (Page 364)
Carrying out offline/offline comparisons (Page 364)
Using the compare editor (Page 365)
Starting a detailed comparison (Page 2875)
Visualization of the comparison result for LAD/FBD (Page 2876)
Navigating in the detailed comparison (Page 2892)
Changing blocks during detailed comparison (Page 2893)

9.1.5.3 Comparing PLC tags

You have the following options for comparing PLC tags:

- Automatic offline/offline comparison in the compare editor
  The PLC tag tables of the selected devices are compared offline.
- Manual offline/offline comparison in the compare editor
  The selected PLC tag tables of the devices are compared offline.
- Detailed comparison
  Use the detailed comparison to determine differences within the PLC tag tables.

Performing automatic offline/offline comparison in the compare editor

To perform an automatic offline/offline comparison of PLC tag tables, follow these steps:

1. Select a device in the project tree that allows offline/offline comparison.
2. Select the "Compare > Offline/offline" command in the shortcut menu.
   The compare editor opens and the selected device is displayed in the left area.
3. Drag-and-drop an additional device to the drop area of the right pane. The device to be compared can originate from the same project, a reference project or the library.
4. Open the "PLC tags" folder.
   You can identify the status of the PLC tag tables based on the symbols in the status and action area. You can define certain actions depending on the status.

You can drag any other device to the drop area at any time to perform further comparisons.

Performing manual offline/offline comparison in the compare editor

To perform a manual offline/offline comparison of PLC tag tables, follow these steps:

1. Select a device in the project tree that allows offline/offline comparison.
2. Select the "Compare > Offline/offline" command in the shortcut menu.
   The compare editor opens and the selected device is displayed in the left area.
3. Drag-and-drop an additional device to the drop area of the right pane. The device to be compared can originate from the same project, a reference project or the library.
4. In the status and action area, click on the button for switching between automatic and manual comparison.
5. Select the PLC tag tables that you want to compare.
   The properties comparison is displayed. You can identify the status based on the symbols.

You can drag any other device to the drop area at any time to perform further comparisons.

Running a detailed comparison

To start a detailed comparison for a PLC tag table, follow these steps:

1. Perform an automatic or manual offline/offline comparison.
2. For an automatic offline/offline comparison in the compare editor, select the PLC tag table for which you want to run a detailed comparison. Note that two PLC tag tables must be selected for comparison for a manual offline/offline comparison.
3. Click the "Start detailed comparison" button in the toolbar.
   A separate compare editor opens. All existing PLC tags of the selected PLC tag tables are displayed depending on the settings of the compare editor. User and system constants are not shown, however. You can identify the status of the PLC tags based on the symbols. You can define certain actions depending on the status of the PLC tags.

See also

Introduction to comparing PLC programs (Page 2869)

Using the compare editor (Page 365)

Comparing blocks (Page 2873)

Comparing PLC data types (Page 2896)

9.1.5.4 Comparing PLC data types

You have the following options for comparing PLC data types:

- Online/offline comparison (S7-1200/1500 only)
  The PLC data types in the project are compared with the PLC data types of the selected device.
- Automatic offline/offline comparison in the compare editor
  The PLC data types of the selected devices are compared offline.
- Manual offline/offline comparison in the compare editor
  The selected PLC data types of the devices are compared offline.
- Detailed comparison
  Use the detailed comparison to determine differences between PLC data types.

Performing online/offline comparison of PLC data types

To perform an online/offline comparison, follow these steps:

1. In the project tree, select a device that allows online/offline comparison.
2. Select the "Compare > Offline/online" command in the shortcut menu.
   If you have not already established an online connection to this device, the "Go online" dialog opens. In this case, set all the necessary parameters for the connection and click "Connect".
   The online connection is established and the compare editor opens.
3. Open the "PLC data types" folder.
   You can identify the status based on the symbols in the status and action area. When you select an object, the properties of the PLC data type and the corresponding PLC data type of the assigned device are displayed in the properties comparison.

Performing automatic offline/offline comparison in the compare editor

To perform an automatic offline/offline comparison of PLC tag tables, follow these steps:

1. Select a device in the project tree that allows offline/offline comparison.
2. Select the "Compare > Offline/offline" command in the shortcut menu.
   The compare editor opens and the selected device is displayed in the left area.
3. Drag-and-drop an additional device to the drop area of the right pane. The device to be compared can originate from the same project, a reference project or the library.
4. Open the "PLC data types" folder.
   You can identify the status of the PLC tag tables based on the symbols in the status and action area. You can define certain actions depending on the status.

You can drag any other device to the drop area at any time to perform further comparisons.

Performing manual offline/offline comparison in the compare editor

To perform a manual offline/offline comparison of PLC data types, follow these steps:

1. Select a device in the project tree that allows offline/offline comparison.
2. Select the "Compare > Offline/offline" command in the shortcut menu.
   The compare editor opens and the selected device is displayed in the left area.
3. Drag-and-drop an additional device to the drop area of the right pane. The device to be compared can originate from the same project, a reference project or the library.
4. In the status and action area, click on the button for switching between automatic and manual comparison.
5. Select the PLC data types that you want to compare.
   The properties comparison is displayed. You can identify the status based on the symbols.

You can drag any other device to the drop area at any time to perform further comparisons.

Running a detailed comparison

To start a detailed comparison for a PLC data type, follow these steps:

1. Perform an offline/offline comparison. You can also perform an online/offline comparison for CPUs of the S7-1200/1500 series.
2. For an automatic offline/offline comparison in the compare editor, select the PLC data type for which you want to run a detailed comparison. Note that two PLC data types must be selected for comparison with a manual offline/offline comparison.
3. Click the "Start detailed comparison" button in the toolbar.
   The two PLC data types are opened next to each other so that you can easily identify the differences.

See also

Introduction to comparing PLC programs (Page 2869)

Using the compare editor (Page 365)

Comparing blocks (Page 2873)

Comparing PLC tags (Page 2895)

9.1.6 Compiling and downloading blocks

9.1.6.1 Compiling blocks

Basic information on compiling blocks

Introduction

A user program must first be compiled before the CPU can execute it. You need to recompile your program each time you make a change.

The following procedures take place during compilation:

- The user program is checked for syntax errors.
- Unneeded instructions are removed from the user program.
- All the block calls within the compiled blocks are checked. In case of changes to the interface of called blocks, errors will be shown in the "Compilation" tab of the information window. You have to correct these errors first.

- The blocks must be numbered uniquely in the user program. If more than one block has the same number, the blocks with number conflicts are renumbered automatically during compilation. A block will not be renumbered in the following cases:
  - The block was selected either individually or as part of a multiselection for the compilation.
  - The block is know-how protected.
  - The number assignment is set to "manual" in the properties of the block.

  Number conflicts that cannot be resolved by automatic renumbering must be corrected manually. Note the messages in the Inspector window for this. You can only correct know-how-protected blocks manually if you know the password since the blocks must then be recompiled.
- Finally, the user program is compiled into a code that can be read by the CPU.

Compilation methods

You can start compilation in the following windows or editors:

- Compiling blocks in the project tree
  Serves to compile individual blocks or the simultaneous compilation of one or several blocks in the "Program blocks" folder.
- Compiling blocks in the program editor
  This is intended for compilation of a single open block.
- Compiling blocks in the call or dependency structure
  Used to compile individual blocks.
  See also: Call structure (Page 2945), Dependency structure (Page 2952)

Compilation options

If you are compiling blocks in project tree, you have further options:

- Software (changes only)
  All program changes of the selected blocks are compiled. If you have selected a block folder, all program changes to the bocks contained in the folder are compiled.
- Software (compile all blocks)
  All blocks are compiled. This is recommended for the first compilation and after major revisions.
- Software (reset memory reserve)
  All tags declared in the reserve area of the interface of selected blocks are moved to the standard area of the interface. Memory reserve is now available for further interface extensions.

  Note

  This option is only available for CPUs of the S7-1500 and S7-1200 V4 series.

Consistency check

Changing the interfaces of blocks called or PLC data types used can result in inconsistencies between calling blocks and called blocks or between the PLC data types and the global data blocks which use these PLC data types.

To avoid such inconsistencies in the user program, the system performs an automatic consistency check before each compilation process. The time stamps are compared and compilation is then either carried out or canceled depending on the results of the comparison.

- The calling block can only be compiled if the time stamps of the interfaces of the called blocks are older than those of the calling block.
- A global data block based on a PLC data type can only be compiled correctly if the time stamp of the global data block is newer than the time stamp of the PLC data type used.
- The instance data block can only be compiled correctly if the interface time stamps for the interface of the instance data block are identical to those of the assigned function block.

If the compilation process is cancelled, an alarm is displayed in the inspector window. Update the block calls in the relevant blocks and the PLC data types in the global data blocks and then restart compilation. The consistency check also finds know-how protected blocks which cannot be compiled. The corresponding messages will also be shown in the inspector window.

If you start loading immediately instead of first compiling, the blocks selected will be automatically compiled and the block call and global data blocks implicitly updated. Please note the following differences between the CPU families:

- S7-1200/1500: All blocks affected are loaded to ensure no inconsistencies can arise.
- S7-300/400: Only the block selected is loaded.

See also

Compiling blocks in the project tree (Page 2900)

Compiling blocks in the program editor (Page 2902)

Correcting compilation errors (Page 2902)

Block time stamps (Page 2515)

Updating block calls in LAD (Page 2588)

Updating block calls in FBD (Page 2630)

Updating block calls in STL (Page 2674)

Compiling project data (Page 353)

Compiling blocks in the project tree

You can compile one block, multiple blocks or all of the blocks in the project tree.

For CPUs of the S7-1500 and S7-1200 V4 series, you can also reset the memory layout of blocks with memory reserve by running a compilation. For more information on memory reserve, refer to chapter "Loading blocks (S7-1200/1500) > Loading block changes without reinitialization".

Requirement

The project tree is open.

Compiling one or more blocks in the project tree

To compile multiple blocks in the project tree, follow these steps:

1. Open the "Program blocks" folder in project tree.
2. Select the blocks you want to compile.
3. Select the "Compile > Software (only changes)" command from the shortcut menu.

Compiling all blocks in the project tree

To compile all blocks in the "Program blocks" folder in project tree, follow these steps:

1. Select the "Program blocks" folder in the project tree.
2. You can select one of two different options for the compilation:
   - If you want to compile only the changes since the last compilation, select the "Compile > Software (only changes)" command in the shortcut menu.
   - If you want to compile all blocks completely, select the "Compile > Software (compile all blocks)" command in the shortcut menu.

Resetting memory layout (S7-1500/S7-1200 V4)

Proceed as follows to reset the memory layout of blocks:

1. Select the "Program blocks" folder, or specific blocks in this folder.
2. Select the "Compile > Software (Reset memory reserve)" command from the shortcut menu.

Result

The code for the blocks will be generated if the consistency check has been successful. Instance data blocks generated by the system which are no longer needed will be deleted.

The message under "Info > Compilation" in the inspector window reports whether the compilation was successful.

See also

Basic information on compiling blocks (Page 2898)

Compiling blocks in the program editor (Page 2902)

Correcting compilation errors (Page 2902)

Finding syntax errors in the program (Page 2819)

Compiling blocks in the program editor

Note

Note that the block is recompiled even if you have not made any changes and the time stamp of the block is modified.

Requirement

The block to be compiled is open.

Procedure

To compile a block in the program editor, follow these steps:

1. Right-click in the instruction window of the programming editor.
2. Select the "Compile" command in the shortcut menu.

Result

The code for the block is generated. Instance data blocks generated by the system which are no longer needed will be deleted.

The message under "Info > Compilation" in the inspector window reports whether the compilation was successful.

See also

Basic information on compiling blocks (Page 2898)

Compiling blocks in the project tree (Page 2900)

Correcting compilation errors (Page 2902)

Correcting compilation errors

In the Inspector window in "Info > Compile", you can see whether any compilation was successful or whether errors were detected in the program. If errors occur, you will need to correct them and then start the compilation again.

Procedure

To correct errors following compilation, follow these steps:

1. Open the error list in the Inspector window with "Info > Compile".
2. If there is one, click on the blue question mark next to the error text for information on remedying errors.
3. Double-click the error you want to correct.
   The corresponding error is highlighted.

4. Correct the error.
5. Restart compilation.

See also

Basic information on compiling blocks (Page 2898)
Compiling blocks in the program editor (Page 2902)
Compiling blocks in the project tree (Page 2900)

9.1.6.2 Downloading blocks

Introduction to downloading blocks

Downloading blocks to device

So that the CPU can execute the user program, the program must first be compiled and then downloaded to the device. The following options are available for downloading:

- Downloading blocks to the program editor
  You can load a single open block in the program editor.
- Downloading blocks to the project tree
  You can download several or all of the blocks in the block folder via the project tree.

During the loading operation, all information that is required for the reconstruction of the program, including symbolic information such as the names and comments for code and data blocks, is also loaded in the current project language. If you change the project language, you must therefore re-load the program.

The symbolic information is not loaded to the work memory, but rather to the load memory.

After the data has been loaded from a device, the symbolic information is available again in your program, which increases the readability of your program code. Please note, however, that loading to and from a device is not a substitute for storing data in an offline project, as watch tables or multi-language capability of projects cannot be reproduced by loading to and from a device.

After loading from a device, you can only display all data from know-how-protected blocks by entering the correct password.

Note

To avoid inconsistencies between calling and called blocks, all blocks affected are compiled and loaded after each global change, such as a change in the block interface.

Note

S7-1200 Version 1.0

If you download an element of your project to the CPU, for example a program block, a data block or the hardware configuration, the CPU runs a cold restart the next time it changes to RUN mode. Apart from deleting the inputs, initializing the outputs and deleting the non-retentive memory, cold restart also deletes the retentive memory areas. All subsequent changes from STOP to RUN are warm restarts in which the retentive memory is not deleted.

Note

S7-1500

The load memory of S7-1500 series CPUs is on the SIMATIC memory card. Therefore, a SIMATIC memory card absolutely must be inserted in order to operate the CPU.

Uploading blocks from device

You can load the blocks of a device to your project. This is necessary, for example, if you want to edit blocks that only exist in this device. You have the option of loading either all available blocks (organization blocks, function blocks, functions and data blocks) and global PLC tags or individual blocks to the project.

Uploading blocks from or downloading blocks to a memory card

Memory cards are plug-in cards used with an S7-1200 series CPU, for example, to replace the load memory of a device. In the case of S7-1500 series CPUs, they contain the load memory. Only Siemens SD cards can be used for devices of the S7-1200 and S7-1500 product range.

To use a memory card as load memory, you must download the user program or individual blocks to a memory card. You can just as well upload blocks from a memory card back into the project.

Note

S7-1200

Note the following when uploading to or downloading from a memory card:

- If the CPU contains no previous program and you insert an empty memory card in the CPU the program will be loaded from the PG/PC to the memory card and not to the CPU.
- If you insert an empty memory card prior to the startup of the CPU, the program that is on the CPU will be transferred automatically to the memory card. The program on the CPU will then be deleted.
- If you insert a memory card with a program in the CPU prior to the startup of the CPU and the CPU already contains a program, the program on the memory card will be executed and not the program on the CPU. The program on the CPU will be deleted.

Loading GRAPH function blocks

If you load a GRAPH function block together with its instance data block, the processing of the sequencer starts over at the initial step. As a result, problems may occur when synchronizing the sequencer with the process. You can avoid these problems by switching off the sequencer before loading.

Loading block changes without reinitialization

It often proves necessary to edit or expand a PLC program that was already commissioned and that is running on the plant without error. Such operations should be performed without causing any major interruptions of current operations.

S7-1500 therefore offers the option of extending the interfaces of function or data blocks during runtime and loading the modified blocks without setting the CPU to STOP or affecting the value of tags that are already loaded. This is a simple means of implementing program changes. This load process (download without reinitialization) will not have a negative impact on the controlled process.

Effects of a load operation on the tag values of a data blocks

When data blocks are downloaded to a device in STOP mode, the next transition of the device to RUN affects the current tag values as follows:

- Tags not marked as being retentive retain their defined start values.
- Retentive tags of the S7-1200 only retain their values if the following conditions are met:
  - You loaded the data block by means of "Download to device > Software (changes only)".
  - You made no changes to the DB structure.

  Otherwise the retentive tags will also retain their defined start values.
- Retentive tags of the S7-1500 only retain their values if the following conditions are met:
  - You loaded the data block by means of "Download to device > Software (changes only)".
  - You made no changes to the structure of the data block or modified it within the memory reserve.

  Otherwise the retentive tags will also retain their defined start values.

Loading blocks with synchronization

In team engineering, it is possible for several users to work on one project with several engineering systems at the same time and access one S7-1500 CPU. To ensure consistency within the shared project, it is necessary to synchronize the changed data prior to loading so that nothing gets overwritten unintentionally.

If differences are determined between the online and offline data management within the shared project during loading that were caused by a different engineering system, automatic synchronization of the data to be loaded is offered during loading.

In this case, the "Synchronization" dialog displays the data to be synchronized with the current status (online-offline comparison) and the possible actions.

The following options are available for synchronization:

Application case	Recommendation	Synchronization
One or more blocks on the CPU (online) are more recent than in the engineering system (offline).	These blocks should be downloaded from the CPU to the engineering system before loading.	Automatic synchronization is possible: The blocks in the engineering system are updated prior to loading.
One or more new blocks have been created and exist only in the CPU (online).	These blocks should be downloaded from the CPU to the engineering system before loading.	Automatic synchronization is possible: The new blocks are added prior to the download in the engineering system.
One or more blocks on the CPU have been deleted.	The blocks should also be deleted prior to the download in the engineering system.	Automatic synchronization is not possible. The blocks deleted on the CPU should be manually deleted in the offline project in the engineering system.
One or more blocks on the CPU and in the engineering system are different. This is the case when a different user has changed blocks to which you have also made corrections and has already downloaded them to the CPU.	These blocks with competing changes must be adapted manually. You decide in this case which changes you are going to accept. If the blocks on the CPU are to be retained, you should accept these blocks prior to download from the CPU to your engineering system. If the blocks that you have changed are to be applied, you can continue with the download without synchronization.	Automatic synchronization is not possible: The affected blocks on the CPU or in the engineering system must be adapted manually. One of the existing block versions (online or offline) will be overwritten in the process.
There are differences in the hardware configuration on the CPU (online) and in the engineering system (offline).	Differences in the hardware configuration must be adapted manually. You decide in this case which hardware configuration you are going to accept. If the existing hardware configuration on the CPU is to be retained, you should apply these in your engineering system prior to loading. If you want to apply the changed hardware configuration, you can continue with the download without synchronization.	Automatic synchronization is not possible: The hardware configuration must be adapted manually. One of the existing hardware configurations (online or offline) will be overwritten in the process.

You can use the "Force download to device" command to download blocks without synchronization, if desired.

Downloading blocks in the "RUN" operating mode to the device

Basics on downloading blocks in the "RUN" operating mode

When you download modified blocks to the device, it is not always necessary to switch the device to the "STOP" operating mode. Prior to a download operation, the Engineering System checks whether the device must be stopped before downloading. The result of this check is

displayed in the "Load preview" dialog. If it is necessary to change to the "STOP" operating mode, you cannot continue the download process until you have set the appropriate option.

Note

Actual parameters are not overwritten by a download process in the "RUN" operating mode. Changes to the actual parameters will not become effective until the next time you change the operating mode from "STOP" to "RUN".

The following table shows the actions after which you can execute the download process in the "RUN" operating mode:

Action	Download in "RUN" operating mode possible	
	S7-1200	S7-1500
Downloading individual blocks	Yes	Yes
Downloading all blocks	No	No
Adding or deleting OBs	No	Yes
Adding or deleting DBs, FCs or FBs	Yes	Yes
Changing block interfaces for FBs	No	Only within the memory reserve
Changing block interfaces for FCs	Yes	Yes
Changing the structure of a DB or an instance DB	No	Only within the memory reserve
Changing the hardware configuration	No	No
Changing the retentivity settings of bit memories and DBs	No	No
Changing the program code of FC, FB or OB	Yes	Yes
Changing the attributes of OBs	Yes	Yes
Adding comments	Yes	Yes
Adding input, output or bit memory areas	Yes	Yes
Changing tag names	Yes	Yes
Maximum number of blocks that can be downloaded for the device used in "RUN" not exceeded	Yes	Yes

See also

Downloading blocks from program editor to device (Page 2907)

Downloading blocks from the project tree to the device (Page 2909)

Downloading project data to a device (Page 357)

Downloading blocks from program editor to device

Requirement

The block to be downloaded is open.

Procedure

To download a block from the program editor to the device, follow these steps:

1. Right-click in the instruction window of the programming editor.
2. Select the "Download to device" command in the shortcut menu.
   - If you have not already established an online connection, the "Extended download to device" dialog opens. In this case, set all parameters required for the connection and click "Load". You have the option of showing all compatible devices by selecting the corresponding option and clicking the "Start search" command. You can also open the "Extended download to device" dialog explicitly via the "Online" menu.
     See also: Go online and Go offline
   - If you have already specified an online connection, then the project data is compiled if necessary and the "Load preview" dialog opens. This dialog displays messages and proposes actions necessary for loading.
3. Check the messages and, where necessary, enable the actions in the "Action" column.

   Note
   Actions

   Performing the proposed actions during ongoing plant operation can cause serious damage to property or injury to persons if there are functional faults or program errors.

   Make sure that no dangerous situations can arise before you start the actions.

   As soon as downloading becomes possible, the "Load" button is enabled.
4. Click "Load".
   If there is a need for synchronization, the system automatically displays the "Synchronization" dialog. This dialog displays messages and suggests actions that are needed for the synchronization. You have the option of performing these actions or forcing the download without synchronization by clicking "Force download to device". If you have performed the suggested actions, you will be asked whether you want to continue with the download. Click "Continue download" in order to download the block. The "Load results" dialog then opens and shows you the status and the actions after the download operation.
5. If you want to start the modules again directly after downloading, select the "Start all" check box.
6. To close the "Load results" dialog box, click "Finish".

Result

The code for the block will be downloaded to the device. If the changes affect additional blocks, these will be compiled and also downloaded to the device. Blocks that only exist online in the device are deleted. Existing CPU data blocks are retained, however. Inconsistencies between the blocks in the user program are avoided by loading all blocks affected and deleting the unneeded blocks in the device.

The messages under "Info > General" in the Inspector window show whether the downloading process was successful.

See also

Downloading blocks from the project tree to the device (Page 2909)

Downloading project data to a device (Page 357)

Downloading blocks in the "RUN" operating mode to the device (Page 2906)

Downloading blocks from the project tree to the device

In the project tree you can download one block, multiple blocks or all blocks to a device.

Downloading one or more blocks from the project tree to the device

To download one block or multiple blocks to the device from the project tree, follow these steps:

1. Open the "Program blocks" folder in project tree.
2. Select the blocks you want to download.
3. Select the "Download to device > Software (only changes)" command from the shortcut menu.
   - If you have not already established an online connection, the "Extended download to device" dialog opens. In this case, set all parameters required for the connection and click "Load". You have the option of showing all compatible devices by selecting the corresponding option and clicking the "Start search" command. You can also open the "Extended download to device" dialog explicitly via the "Online" menu.
     See also: Go online and Go offline
   - If you have already specified an online connection, then the project data is compiled if necessary and the "Load preview" dialog opens. This dialog displays messages and proposes actions necessary for loading.
4. Check the messages and, where necessary, enable the actions in the "Action" column.

   Note

   Performing the proposed actions during ongoing plant operation can cause serious damage to property or injury to persons if there are functional faults or program errors.

   Make sure that no dangerous situations can arise before you start the actions.

   As soon as downloading becomes possible, the "Load" button is enabled.
5. Click "Load".
   If there is a need for synchronization, the system automatically displays the "Synchronization" dialog. This dialog displays messages and suggests actions that are needed for the synchronization. You have the option of performing these actions or forcing the download without synchronization by clicking "Force download to device". If you have performed the suggested actions, you will be asked whether you want to continue with the download. Click "Continue download" in order to download the block. The "Load results" dialog then opens and shows you the status and the actions after the download operation.
6. If you want to start the modules again directly after downloading, select the "Start all" check box.
7. To close the "Load results" dialog box, click "Finish".

Downloading blocks from the project tree to the device

To download all blocks in the "Program blocks" folder to the device from the project tree, follow these steps:

1. Select the "Program blocks" folder in the project tree.
2. Select the "Download to device" submenu in the shortcut menu.
3. If you only want to download the changes since the last download, select the "Software (only changes)" option. If all blocks are to be fully loaded and all values are to be reset to their start values, select "Download PLC program to the device and reset".
   - If you have not already established an online connection, the "Extended download to device" dialog opens. In this case, set all parameters required for the connection and click "Load". You have the option of showing all compatible devices by selecting the corresponding option and clicking the "Start search" command. You can also open the "Extended download to device" dialog explicitly via the "Online" menu.
     See also: Go online and Go offline
   - If you have already specified an online connection, then the project data is compiled if necessary and the "Load preview" dialog opens. This dialog displays messages and proposes actions necessary for loading.
4. Check the messages and, where necessary, enable the actions in the "Action" column.

   Note

   Performing the proposed actions during ongoing plant operation can cause serious damage to property or injury to persons if there are functional faults or program errors.

   Make sure that no dangerous situations can arise before you start the actions.

   As soon as downloading becomes possible, the "Load" button is enabled.
5. Click "Load".
   If there is a need for synchronization, the system automatically displays the "Synchronization" dialog. This dialog displays messages and suggests actions that are needed for the synchronization. You have the option of performing these actions or forcing the download without synchronization by clicking "Force download to device". If you have performed the suggested actions, you will be asked whether you want to continue with the download. Click "Continue download" in order to download the block. The "Load results" dialog then opens and shows you the status and the actions after the download operation.
6. If you want to start the modules again directly after downloading, select the "Start all" check box.
7. To close the "Load results" dialog box, click "Finish".

Result

The code for the blocks is downloaded to the device. If the changes affect additional blocks, these will be compiled and also downloaded to the device. Blocks that only exist online in the device are deleted. Inconsistencies between the blocks in the user program are avoided by loading all blocks affected and deleting the unneeded blocks in the device.

The messages under "Info > General" in the Inspector window show whether the downloading process was successful.

See also

Downloading blocks from program editor to device (Page 2907)

Downloading project data to a device (Page 357)

Downloading blocks in the "RUN" operating mode to the device (Page 2906)

Uploading blocks from device

You can load either all blocks or individual blocks from a device into your project.

Note

Note the following information:

- Please note that when you load individual blocks, no tags or other required blocks to which you may refer are loaded together with the individual blocks. During the loading operation, reference to tags and blocks are reassigned where possible based on the names. After the loading operation, check whether these assignments are correct.
- S7-1500: When GRAPH function blocks are loaded from a device to your project, the step-specific alarm texts for the interlock and supervision alarms are not loaded.

Requirement

The online and offline versions of a block to be loaded are different or the blocks only exists online.

Uploading all blocks from a device

To upload all blocks from a device, follow these steps:

1. Establish an online connection with the device from which you want to upload the blocks.
   See also: Establishing and terminating an online connection
2. In the project tree, select the device folder from which you want to upload blocks.
3. In the "Online" menu, select the "Upload from device" command.
   The "Upload preview" dialog box opens. This dialog displays alarms and proposes actions necessary for loading.
4. Check the alarms and, where necessary, enable the actions in the "Action" column.
   The "Upload from device" button will be enabled as soon as uploading becomes possible.
5. Click on the "Upload from device" button.
   The load is executed.

Uploading individual blocks from a device

To upload individual blocks from a device, follow these steps:

1. Establish an online connection with the device from which you want to upload the blocks.
   See also: Establishing and terminating an online connection
2. In the project tree, select the blocks that you want to upload from the device.

3. In the "Online" menu, select the "Upload from device" command.
   The "Upload preview" dialog box opens. This dialog displays alarms and proposes actions necessary for loading.
4. Check the alarms and, where necessary, enable the actions in the "Action" column.
   The "Upload from device" button will be enabled as soon as uploading becomes possible.
5. Click on the "Upload from device" button.
   The load is executed.

Result

The blocks will be uploaded from the device to the project. You can edit them as normal, recompile them and download them to the device again.

Downloading blocks to a memory card

Requirement

- The memory card is marked as a program card.
- The "Program blocks" folder of the memory card is open.

Procedure

To download blocks to a memory card, follow these steps:

1. Open the "Program blocks" folder of the device in the project tree.
2. Select the blocks you want to download to the memory card.
3. Drag the blocks in project tree to the "Program blocks" folder of the memory card.
   The "Load preview" dialog opens. This dialog displays messages and proposes actions necessary for loading.
4. Check the messages and, where necessary, enable the actions in the "Action" column.
5. As soon as downloading becomes possible, the "Load" button is enabled.
6. Click the "Load" button.
   If there is a need for synchronization, the system automatically displays the "Synchronization" dialog. This dialog displays messages and suggests actions that are needed for the synchronization. You have the option of performing these actions or forcing the download without synchronization by clicking "Force download to device". If you have performed the suggested actions, you will be asked whether you want to continue with the download. Click "Continue download" in order to download the block. The "Load results" dialog then opens and shows you the status and the actions after the download operation.
7. Click "Finish".

Result

The block is downloaded to the memory card If the changes affect additional blocks, these will also be downloaded to the memory card. Blocks that exist only on the memory card are deleted.

Inconsistencies between the blocks in the user program are avoided by downloading all affected blocks and the deleting of the non-required blocks on the memory card.

The messages under "Info > General" in the Inspector window show whether the downloading process was successful.

See also

Uploading blocks from a memory card (Page 2913)

Accessing memory cards (Page 414)

Uploading blocks from a memory card

You can only upload all blocks from one memory card back into your project.

Requirement

The memory card is displayed.

See also: Accessing memory cards (Page 414)

Procedure

To upload blocks from a memory card to your project, follow these steps:

1. In the project tree, drag the folder of the memory card to the folder of the device in the project.
   The "Upload preview" dialog box opens. This dialog displays alarms and proposes actions necessary for loading.
2. Check the alarms and, where necessary, enable the actions in the "Action" column.
   The "Upload from device" button will be enabled as soon as uploading becomes possible.
3. Click on the "Upload from device" button.

See also

Downloading blocks to a memory card (Page 2912)

Switching off the sequencer prior to loading a GRAPH DB

You can specify the switching off of the sequencer prior to loading an instance data block either globally or during the load operation.

Globally switching off the sequencer

To switch off the sequencer globally for each loading operation of an instance data block, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "PLC programming > GRAPH" group in the area navigation.
3. Select the "Turn off sequence before downloading DB" check box.
   For future loading operations, the sequencer is switched off prior to loading of the instance data block.

Switching off the sequencer during the loading operation

To switch off the sequencer during the loading operation, follow these steps:

1. Load the GRAPH function block to the device.
   During the loading operation, the "Load preview" dialog opens. This dialog displays alarms and suggests the required actions for loading. If the instance data block must be loaded together with the GRAPH function block, the "Load preview" dialog suggests the action "Turn off sequence before downloading the DB".
2. Select the "Turn off sequence before downloading DB" check box.

Loading block extensions without reinitialization

Basic information on loading block extensions without reinitialization

Description

To enable the subsequent editing of PLC programs that have already been commissioned and are running without error on a system, CPUs of the S7-1200 V4 and S7-1500 series support the option of extending the interfaces of function or data blocks during runtime.

You can download the modified blocks without setting the CPU to STOP and without affecting the values of already loaded tags.

This is a simple means of implementing program changes. This load process (download without reinitialization) will not have a negative impact on the controlled process.

Principle of operation

In principle, every function or data block has a memory reserve by default, which can be used for subsequent interface changes. The memory reserve is not used initially. Activate the memory reserve if you decide on loading interface changes after having compiled and downloaded the block. All tags that you subsequently declare will be saved to the memory reserve. A subsequent download has no impact on the values of tags that have already been loaded.

If you decide to revise your program at a later time while the plant is not in operation, you also have the option of resetting the memory layout of one or more blocks in one step. With this

action, you move all tags from the reserve area to the regular area. The memory reserve is now cleared and made available for further interface extensions.

Requirements

This "Download without reinitialization" function is available if the following requirements are met:

- The project is in the "TIA Portal V12" format or a higher version.
- You are working with a CPU of the S7-1200 V4 or S7-1500 series.
- The blocks were created in LAD, FBD, STL, or SCL.
- The blocks were created by the user, i.e. they were not included in your delivery package.
- They are blocks with "optimized access".

Basic steps

Perform the following steps if you want to extend the interface of a function or data block and then load the block without reinitialization. A detailed description of the individual steps can be found in the following chapters:

1. By default, all blocks have a memory reserve of 100 bytes in the non-retentive memory. If needed, you can adjust the amount of memory reserve or define additional memory reserve in the retentive memory.
2. Activate the memory reserve.
3. Extend the block interface.
4. Compile the block.
5. Download the block to the CPU as usual.

Note

The following blocks do not offer the option of downloading block extensions without reinitialization:

- GRAPH blocks
- ARRAY data blocks

See also

General information on loading (Page 355)

Initializing setting values in the online program (Page 2852)

Setting the reserved memory for block extensions (Page 2916)

Activating memory reserve (Page 2917)

Deactivating memory reserve (Page 2918)

Extending the block interface or data block (Page 2919)

Resetting the reserved memory (Page 2920)

Setting the reserved memory for block extensions

Introduction

Essentially, each function or data block has a reserve of 100 bytes by default. This reserve is not located in the block's retentive memory area. By default, no reserve is available in the retentive memory area, since the retentive memory is restricted to the CPU.

You can change the default value of 100 bytes, which applies to all newly created blocks in the project. Furthermore, you can change the size of the reserved memory individually for specific blocks and define a reserve in the retentive memory area for specific blocks.

Setting the amount of reserved memory for new blocks

To set the size of the reserved memory for all newly created blocks in the project, proceed as follows:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "PLC programming > General" group in the area navigation.
3. In the "Reserved memory for download without reinitialization" group, enter in the "Memory reserve" input box the desired number of bytes to be allocated for a later extension of the block interface.
   Newly created function blocks and global data blocks now get a reserved memory in the specified size. Blocks that already exist are not affected by this change.

Note

The setting is only valid for blocks that support the "Download without reinitialization" function. Other blocks are not influenced by this setting.

Setting the amount of reserved memory for an existing block

To set the amount of reserved memory for an existing block, proceed as follows:

1. Select the block in the project tree.
2. Select the "Properties" command in the shortcut menu.
   The "Properties" dialog opens.
3. Select the "Download without reinitialization" group in the area navigation.

4. Enter the desired number of bytes in the "Memory reserve" input box.
5. To define a reserve in retentive memory, select the "Enable download without reinitialization for retentive tags" check box and enter the desired number of bytes in the "Retentive reserved memory" input box.

Note

Please note the following:

- If you have already activated reserved memory for the current block, you can no longer change the size of the reserved memory.
- In the memory, data elements are sorted according to specific rules that allow the fastest possible access while the program is running. This is why the available memory is sometimes reduced by more than the actual size of the data element when a new data element is added.
  For example, the available memory area may be reduced by four bytes when an element of the "WORD" type and an element of the "BYTE" type are added, since all data elements are aligned to a WORD limit.

See also

Basic information on loading block extensions without reinitialization (Page 2914)

Activating memory reserve

Introduction

Each function or data block is always assigned a default memory reserve. However, this memory reserve is not used initially. Activate the memory reserve if you have compiled and loaded the block and then decide that you want to load interface changes later. All tags that you subsequently declare will be saved to the memory reserve.

Requirements

- The block is currently compiled.
- The block contains a memory reserve.

Procedure

To activate the memory reserve for a function block or global data block, proceed as follows:

1. Open the block.
2. Select the "Download without reinitialization" button on the toolbar.

Result

- As of now, all newly declared tags will be stored in the memory reserve and can therefore be downloaded without affecting the process that is running.
- It is no longer possible to delete existing tags. Their properties can no longer be changed, except for the properties "Default value", "Start value" and "Comment".
- Local constants which are used as ARRAY limits can no longer be changed, because changing the constant value could have an effect on the memory layout of the block.

See also

Basic information on loading block extensions without reinitialization (Page 2914)

Deactivating memory reserve

Introduction

In order to revise the memory layout of a block at a later time when the plant is not in operation, you can deactivate the reserved memory. With this action, you move all tags from the reserve area to the regular area. The reserved memory remains in existence and is now made available for further interface extensions. It then has the size which is defined in the block properties again.

Note

Please note that the block has to be compiled and loaded again after you have deactivated the reserved memory. It is not possible to perform a "Download without reinitialization" after deactivating the reserved memory.

Requirement

The block's reserved memory is activated.

Procedure

To activate the reserved memory for a function block or global data block, proceed as follows:

1. Open the block.
2. Select the "Download without reinitialization" button on the toolbar.

Result

With this action, you move all tags that were previously located in the reserve area to the regular area. As a result, the block has to compiled and loaded again. The values of the tags in the CPU are reinitialized during loading.

See also

Basic information on loading block extensions without reinitialization (Page 2914)

Extending the block interface or data block

Introduction

You can add new tags after you have activated the reserved memory of a function block or global data block.

Requirement

The reserved memory is activated.

Procedure

To declare additional tags, follow these steps:

1. Select a declaration section, for example "Input", "Output", "InOut" or "Static".
2. In the selected section, declare as usual one or more tags and enter their properties in the columns. In the "Retain" column of an FB, you can select only between the "Retain" and "Non-retain" settings. The "Set in IDB" setting is not available in the reserved memory.

   Note

   It is not possible to add new tags within a tag with structured data type. You cannot declare any new structure elements within an existing structure, for example.

   However, you can create new structures within the reserved memory.

3. Compile the changed blocks, for example by selecting them in the project tree and selecting the command "Compile> Software (only changes)" in the shortcut menu.
   During compiling, the newly declared tags are inserted into the reserved memory. If the reserved memory is too small, the compiling is terminated and an error message notifies you about this.
4. Load the changed blocks, for example by selecting them in the project tree and selecting the command "Load > Software (only changes)" in the shortcut menu.
   When you load block extensions, only the newly added tags with defined start values are initialized. The existing online tags are not reinitialized.

See also

Basic information on loading block extensions without reinitialization (Page 2914)

Resetting the reserved memory

Introduction

If you decide to revise your program at a later time while the plant is not in operation, you have the option of resetting the reserved memory of one or more blocks in one step. With this action, you move all tags from the reserved memory to the regular memory area. The reserved memory is now cleared and made available for further interface extensions.

Requirement

- The block's reserved memory is activated.
- The reserved memory contains tags.

Procedure

To reset the reserved memory of one or more blocks, follow these steps:

1. Select the "Program blocks" folder, or specific blocks in this folder.
2. Select the "Compile > Software (Reset memory reserve)" command from the shortcut menu.

Result

All tags in the selected blocks that were previously located in the reserved memory area are moved from this area to the regular area. The block is recompiled. The tags are reinitialized during the next loading. The block's configured reserved memory is retained and continues to be active.

DANGER

Reinitialization at the next load

When you reset the reserved memory, all the tags that were previously in the reserved memory are reinitialized the next time you load. This is true for both retentive and non-retentive tags.

Changing tag values during ongoing plant operation can cause serious material damage and personal injuries! It is therefore very important that you check the program thoroughly in a test environment after resetting the reserved memory before putting it in operation.

See also

Basic information on loading block extensions without reinitialization (Page 2914)

9.1.6.3 Downloading blocks

Introduction to loading of blocks

Downloading blocks to device

So that the CPU can execute the user program, the program must first be compiled and then downloaded to the device. The following options are available for downloading:

- Downloading blocks to the program editor
  You can download an individual open block in the program editor.
- Downloading blocks to the project tree
  You can download several blocks or all of the blocks in the block folder using the project tree.

Note

If you download a know-how-protected block to a device, no restore information is installed along with it. This means that you cannot reopen a know-how-protected block if you upload it from the device.

Uploading blocks from device

You can load the blocks of a device to your project. This is necessary, for example, if you want to edit blocks that only exist in this device. You can either upload all existing blocks (organization blocks, function blocks, functions, data blocks) and global PLC tags or just individual blocks to the project.

Uploading blocks from or downloading blocks to a memory card

Memory cards are insertable cards used, for example, to expand the memory of a CPU and to protect data against power failures. You can use different types of memory cards depending on the CPU. For additional information on memory cards approved for your CPU and possible applications, see "Additional information on configurations" in the chapter "Functional description of S7-300/400 CPUs".

Loading GRAPH function blocks

If you load a GRAPH function block together with its instance data block, the processing of the sequencer starts over at the initial step. As a result, problems may occur when synchronizing the sequencer with the process. You can avoid these problems by switching off the sequencer before loading.

Effects of a load operation on the tag values of a data blocks

When data blocks are downloaded to a device in STOP operating state, the next transition of the device to RUN affects the current tag values as follows:

- Tags of non-retentive data blocks retain their defined start values.
- Tags of retentive data blocks only retain their values if the following conditions are met:
  - You loaded the data block by means of "Download to device > Software (changes only)".
  - You made no changes to the DB structure.

  Otherwise the tags of retentive data blocks also retain their defined start values.

Downloading blocks in the "RUN" operating state to the device

Basics about downloading blocks in the "RUN" operating state

When you download modified blocks to a device, it is not always necessary to change the device to the "STOP" operating state. During a downloading process the Engineering System therefore checks whether the device has to be stopped before downloading. The result of the check is displayed in the "Load preview" dialog. If a change to the "STOP" operating state is necessary, you cannot continue the downloading process until you have set the corresponding option.

Fundamentally there are the following limitations for a downloading process in the "RUN" operating state:

- Depending on the device used, the number and the type of blocks that you can download in the "RUN" operating state can be limited.
- It is not possible to load all the blocks using "Download PLC program to the device and reset".
- If you have changed the hardware configuration of the device that is to be used as the target of the downloading process, downloading in the "RUN" operating state is not possible.

See also

Downloading blocks in the program editor to device (Page 2922)

Downloading blocks from the project tree to the device (Page 2924)

Downloading project data to a device (Page 357)

Downloading blocks in the program editor to device

Requirement

The block to be downloaded is open.

Procedure

To download a block from the program editor to the device, follow these steps:

1. Right-click in the Instruction window of the program editor.
2. Select the "Download to device" command in the shortcut menu.
   - If you have not already established an online connection, the "Extended download to device" dialog opens. In this case set all parameters required for the connection and click on "Download". You can also open the "Extended download to device" dialog with the "Online" menu.
     See also: Establishing and terminating an online connection
   - If you have already specified an online connection, then the project data will be compiled if necessary and the dialog "Download preview" opens. This dialog displays alarms and proposes actions necessary for uploading.
3. Check the alarms and, where necessary, enable the actions in the "Action" column.
   As soon as downloading becomes possible, the "Download" button is enabled.

   Note
   Actions

   Performing the proposed actions during operation of the plant can cause serious damage to property or injury to persons if there are functional disturbances or program errors!

   Make sure that no dangerous situations can arise before you start the actions!

   Note

   To avoid inconsistencies between calling and called blocks, download all affected blocks each time you make global changes, such as changes in the block interface. Select the "Consistent download" action.

4. Click "Download".
   The block is downloaded and the "Download results" dialog opens. This dialog shows you the status and the actions after downloading.
5. To close the "Download results" dialog box, click "Finish".

Result

The code for the block is downloaded to the device. If you have selected "Consistent download", then all blocks that are subject to this change will be compiled and downloaded to the device.

The alarms under "Info > General" in the Inspector window report whether the downloading process was successful.

See also

Downloading blocks from the project tree to the device (Page 2924)

Downloading project data to a device (Page 357)

Downloading blocks in the "RUN" operating state to the device (Page 2922)

Downloading blocks from the project tree to the device

In the project tree you can download one block, multiple blocks or all blocks to a device.

Loading one or more blocks from the project tree to the device

To download one block or multiple blocks to the device from the project tree, follow these steps:

1. Open the "Program blocks" folder in project tree.
2. Select the blocks you want to download.
3. Select the "Download to device > Software (only changes)" command from the shortcut menu.
   - If you have not already established an online connection, the "Extended download to device" dialog opens. In this case, set all parameters required for the connection and click "Load". You can also open the "Extended download to device" dialog with the "Online" menu.
     See also: Go online and Go offline
   - If you have already specified an online connection, then the project data is compiled if necessary and the "Load preview" dialog opens. This dialog displays alarms and proposes actions necessary for loading.
4. Check the alarms and, where necessary, enable the actions in the "Action" column.
   As soon as downloading becomes possible, the "Load" button is enabled.

   Note

   Performing the proposed actions during ongoing plant operation can cause serious damage to property or injury to persons if there are functional faults or program errors.

   Make sure that no dangerous situations can arise before you start the actions.

   Note

   To avoid inconsistencies between calling and called blocks, download all affected blocks each time you make global changes, such as changes in the block interface. Select the "Consistent download" action.

5. Click "Load".
   The block is loaded and the "Load results" dialog opens. This dialog shows you the status and the actions after downloading.
6. To close the "Load results" dialog box, click "Finish".

Loading blocks from the project tree to the device

To download all blocks in the "Program blocks" folder to the device from the project tree, follow these steps:

1. Select the "Program blocks" folder in the project tree.
2. Select the "Download to device" submenu in the shortcut menu.

3. If you only want to download the changes since the last download, select the "Software (only changes)" option. If all blocks are to be fully loaded and all values are to be reset to their start values, select "Download PLC program to the device and reset".
   - If you have not already established an online connection, the "Extended download to device" dialog opens. In this case, set all parameters required for the connection and click "Load". You can also open the "Extended download to device" dialog with the "Online" menu.
     See also: Go online and Go offline
   - If you have already specified an online connection, then the project data is compiled if necessary and the "Load preview" dialog opens. This dialog displays alarms and proposes actions necessary for loading.
4. Check the alarms and, where necessary, enable the actions in the "Action" column.
   As soon as downloading becomes possible, the "Load" button is enabled.

   Note

   Performing the proposed actions during ongoing plant operation can cause serious damage to property or injury to persons if there are functional faults or program errors.

   Make sure that no dangerous situations can arise before you start the actions.

   Note

   To avoid inconsistencies between calling and called blocks, download all affected blocks each time you make global changes, such as changes in the block interface. Select the "Consistent download" action.

5. Click "Load".
   The block is loaded and the "Load results" dialog opens. This dialog shows you the status and the actions after downloading.
6. To close the "Load results" dialog box, click "Finish".

Result

The code for the blocks is downloaded to the device. If the changes affect additional blocks, these will be compiled and also downloaded to the device. Blocks that only exist online in the device are deleted. Inconsistencies between the blocks in the user program are avoided by loading all blocks affected and deleting the unneeded blocks in the device.

The messages under "Info > General" in the Inspector window show whether the downloading process was successful.

See also

Downloading blocks in the program editor to device (Page 2922)

Downloading project data to a device (Page 357)

Downloading blocks in the "RUN" operating state to the device (Page 2922)

Uploading blocks from device

You can upload either all the blocks or only individual blocks from a device to your project.

Note

SCL blocks and GRAPH function blocks cannot be uploaded from a device.

Requirements

The online and offline versions of a blocks to be loaded differ or the blocks only exist online.

Uploading all blocks from a device

To upload all blocks from a device, follow these steps:

1. Establish an online connection with the device from which you want to upload the blocks.
   See also: Establishing and terminating an online connection
2. In the project tree, select the device folder from which you want to upload blocks.
3. In the "Online" menu, select the "Upload from device" command.
   The "Upload preview" dialog box opens. This dialog displays alarms and proposes actions necessary for uploading.
4. Check the alarms and, where necessary, enable the actions in the "Action" column.
   As soon as uploading becomes possible, the "Upload from device" button is enabled.
5. Click the "Upload from device" button.
   The load is executed.

Uploading individual blocks from a device

To upload individual blocks from a device, follow these steps:

1. Establish an online connection with the device from which you want to upload the blocks.
   See also: Establishing and terminating an online connection
2. In the project tree, select the blocks that you want to upload from the device.
3. In the "Online" menu, select the "Upload from device" command.
   The load is executed.

Result

The blocks will be uploaded from the device to the project. You can edit them as normal, recompile them and download them to the device again.

Downloading blocks to a memory card

Requirement

The "Program blocks" folder of the memory card is open.

Procedure

To download blocks to a memory card, follow these steps:

1. Open the "Program blocks" folder of the device in the project tree.
2. Select the blocks you want to download to the memory card.
3. Drag the blocks in project tree to the "Program blocks" folder of the memory card.
   The "Download preview" dialog opens. This dialog displays messages and proposes actions necessary for uploading.
4. Check the messages and, where necessary, enable the actions in the "Action" column.
   As soon as downloading becomes possible, the "Load" button is enabled.

   Note

   Performing the proposed actions during operation of the plant can cause serious damage to property or injury to persons if there are functional disturbances or program errors!

   Make sure that no dangerous situations can arise before you start the actions!

5. Click the "Load" button.
   The download is performed. The "Load results" dialog then opens. In this dialog, you can check whether or not the download was successful and take any further action that may be necessary.
6. Click the "Finish" button.

Result

The selected blocks will be downloaded to the memory card. The messages under "Info > General" in the Inspector window report whether the downloading process was successful.

See also

Uploading blocks from a memory card (Page 2927)

Accessing memory cards (Page 414)

Uploading blocks from a memory card

You can only upload all blocks from a memory card to your project. Note that no symbolic information is loaded when you upload from a memory card.

Requirement

The memory card is displayed.

See also: Accessing memory cards (Page 414)

Procedure

To upload blocks from a memory card to your project, follow these steps:

1. In the project tree, drag the memory card folder to the device folder in the project.
   The "Upload preview" dialog box opens. This dialog displays alarms and proposes actions necessary for uploading.
2. Check the alarms and, where necessary, enable the actions in the "Action" column.
   As soon as uploading becomes possible, the "Upload from device" button is enabled.
3. Click the "Upload from device" button.

See also

Downloading blocks to a memory card (Page 2927)

Switching off the sequencer prior to loading a GRAPH DB

You can specify the switching off of the sequencer prior to loading an instance data block either globally or during the load operation.

Globally switching off the sequencer

To switch off the sequencer globally for each loading operation of an instance data block, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "PLC programming > GRAPH" group in the area navigation.
3. Select the "Turn off sequence before downloading DB" check box.
   For future loading operations, the sequencer is switched off prior to loading of the instance data block.

Switching off the sequencer during the loading operation

To switch off the sequencer during the loading operation, follow these steps:

1. Load the GRAPH function block to the device.
   During the loading operation, the "Load preview" dialog opens. This dialog displays alarms and suggests the required actions for loading. If the instance data block must be loaded together with the GRAPH function block, the "Load preview" dialog suggests the action "Turn off sequence before downloading the DB".
2. Select the "Turn off sequence before downloading DB" check box.

9.1.7 Protecting blocks

9.1.7.1 Protecting blocks

Introduction

You can use a password to protect one or more blocks of the OB, FB, FC type and global data blocks from unauthorized access. Note the following particularities:

- You can not manually protect instance data blocks; they depend on the know-how protection of the assigned FB. This means that when you create an instance data block for a know-how protected FB, the instance data block also receives this know-how protection. This is independent of whether you explicitly create the instance data block or if it is created by a block call.
- The know-how protection acts as write protection in global blocks and in instance data blocks.
- You cannot provide ARRAY data blocks with know-how protection.

If a block is know-how protected, only the following data is readable without the correct password:

- Interface parameters Input, Output, InOut, Return, Static
- Block title
- Block comment
- Block properties
- Tags of global data blocks without specification of the location of use

The following actions can be performed with a know-how protected block:

- Copying and deleting
- Calling in a program
- Online/offline comparison
- Load

The code of the block, on the other hand, is protected from unauthorized reading and modification. For S7-1200/1500 CPUs, you can also set up copy protection which binds execution of the block to the CPU, or to the SD card with the defined serial number.

Note

Note the following information:

- S7-1200 version 1.0 and S7-300/400 (only GRAPH and SCL blocks): If you download a know-how-protected block to a device, no restore information is loaded along with it. This means that you cannot open a know-how-protected block again even with the correct password if you upload it from the device.
- Only the non-protected data is compared in offline-online comparison of know-how protected blocks.
- You will no longer be able to access the block if you do not have the password.
- If you add a know-how-protected block to a library, the master copy created will also be know-how protected.
- For S7-1500, you can select the "Block can be used as know-how protected library item" check box in the block properties to obtain the information on whether or not the block can be used as protected library item. For this purpose, the block cannot use any tags from the operand areas Output (Q), Input (I), Bit memory (M), Timer function (T) or Count function (C), and cannot access data blocks.
- Cross references to used tags, bit memories, inputs and outputs in know-how protected blocks are not displayed even after the correct password is entered..
- If you change the number of a block, the loadable binary component of the block is out of date. This means that the block must be recompiled before loading it to a device. For know-how-protected blocks, this is only possible with the correct password. Keep this in mind particularly if you want to copy a know-how-protected block to another device in which there is already a block with the same number.
- Always pass on a project that includes know-how-protected blocks as a project archive or library archive. In this way, you ensure that the know-how protection cannot be bypassed.
- If you wish to assign know-how protection to several blocks using multiple selection, no instance data block can be included in the selection. Otherwise, it will not be possible to set know-how protection.
- You cannot change the know-how protection settings for an open, know-how protected block.

See also

Setting up and removing block copy protection (Page 2931)

Setting up block know-how protection (Page 2932)

Opening know-how protected blocks (Page 2933)

Printing know-how protected blocks (Page 2933)

Removing block know-how protection (Page 2935)

Changing a password (Page 2935)

Archiving projects (Page 349)

Archiving global libraries (Page 437)

9.1.7.2 Setting up and removing block copy protection

For S7-1200 /1500 CPUs, you can set up copy protection which binds execution of the block to a specific CPU or SD card. The block can then only be executed if it is in the device with the set serial number.

It is important that you also know-how-protect any block for which you have set up copy protection. If you do not, anyone can reset the copy protection.

Note

S7-1500 and S7-1200 V2.2 and higher: If you download a copy protected block to a device that does not match the specified serial number, the entire download operation will be rejected. This means that blocks without copy protection, too, will not be downloaded.

Setting up copy protection

To set up copy protection for a block, follow these steps:

1. Open the block you wish to copy-protect.
2. Open the "Properties" tab in the inspector window.
3. Select "Protection" in the area navigation in the inspector window.
4. Select either "Bind to serial number of the CPU" or "Bind to serial number of the memory card" from the drop-down list in the "Copy protection" area.
5. Enter the serial number of the CPU or the memory card for a S7-1500 CPU. You can either enter the serial number directly for a S7-1200 CPU or enable the option "Serial number inserted when downloading to a device or memory card" if the serial number is to be inserted automatically during loading.
6. You can now set up the know-how protection for the block in the "Know-how protection" area, if the block does not already have know-how protection.

Removing copy protection

To remove copy protection, follow these steps:

1. Open the block for which you wish to remove copy protection.
2. Open the "Properties" tab in the inspector window.
3. Select "Protection" in the area navigation in the inspector window.
4. Select "No binding" in the drop-down list in the "Copy protection" area.

See also

Protecting blocks (Page 2929)
Setting up block know-how protection (Page 2932)
Opening know-how protected blocks (Page 2933)
Printing know-how protected blocks (Page 2933)
Removing block know-how protection (Page 2935)
Changing a password (Page 2935)

9.1.7.3 Setting up block know-how protection

You can set up know-how protection for blocks in the devices in your project.

Procedure

To set up block know-how protection, follow these steps:

1. Select the blocks with no know-how protection which you want to protect.
2. Select the command "Know-how protection" in the "Edit" menu.
   The "Know-how protection" dialog will open.
3. Click "Define".
   The "Define password" dialog box opens.
4. Enter a password in the "New" field.
5. Enter the same password in the "Confirm" field.
6. Confirm your entries with "OK".
7. Close the "Know-how protection" dialog by clicking on "OK".

Result

The blocks selected will be know-how-protected. Know-how protected blocks are marked with a lock in the project tree. The password entered is valid for all blocks selected.

See also

Protecting blocks (Page 2929)
Setting up and removing block copy protection (Page 2931)
Opening know-how protected blocks (Page 2933)
Printing know-how protected blocks (Page 2933)
Removing block know-how protection (Page 2935)
Changing a password (Page 2935)

9.1.7.4 Opening know-how protected blocks

You can only open multiple know-how protected blocks at once if they are protected with the same password.

Procedure

To open a know-how protected block, follow these steps:

1. Double-click on the block you wish to open.
   The "Access protection" dialog will open.
2. Enter the password for the know-how protected block.
3. Confirm your entry with "OK".

Result

The know-how protected block will open provided you have entered the correct password. However, the block will remain know-how protected. If you copy the block or add it to a library, for example, the copies will also be know-how protected.

Once you have opened the block, you can edit the program code and the block interface of the block for as long as the block or TIA portal is open. The password must be entered again the next time the block is opened. If you close the "Access protection" dialog with "Cancel", the block will open but the block code will not be displayed and you will not be able to edit the block.

See also

Protecting blocks (Page 2929)

Setting up and removing block copy protection (Page 2931)

Setting up block know-how protection (Page 2932)

Printing know-how protected blocks (Page 2933)

Removing block know-how protection (Page 2935)

Changing a password (Page 2935)

9.1.7.5 Printing know-how protected blocks

You can only print complete know-how protected blocks if they have been opened with the correct password. If you print a closed block or if the block was not opened with the correct password, only the non-protected block data will be printed.

Procedure

To print a know-how protected block in full, follow these steps:

1. Open the know-how protected block you wish to print.
   See also: Opening know-how protected blocks (Page 2933)
2. Select the "Print" command in the "Project" menu.
   The "Print" dialog will open.
3. Select the printer in the "Name" field.
4. Click "Advanced" to modify the Windows printer settings.
5. Select the documentation information set in the "Document information" drop-down list that you want to use for the frame layout.
6. Under "Print objects/area" select whether you want to print all objects or the complete area, or only a selection.
7. Under "Properties" select the print scope.
   - Select "All" to print the complete block.
   - Choose "Visible" to print all the information within the block that is visible on the screen.
   - Select "Compact" to print a shortened form of the block.
8. Click "Preview" to generate a print preview in advance.
   A print preview is created in the work area.
9. Click "Print" to start the printout.

See also

Printing project contents (Page 380)

Protecting blocks (Page 2929)

Setting up and removing block copy protection (Page 2931)

Setting up block know-how protection (Page 2932)

Removing block know-how protection (Page 2935)

Changing a password (Page 2935)

9.1.7.6 Changing a password

Procedure

To change the password, follow these steps:

1. Select the know-how protected blocks for which you want to change the password.

   Note

   You can only change the password for several blocks at once if all blocks selected have the same password.

2. Select the command "Know-how protection" in the "Edit" menu.
   The "Know-how protection" dialog will open.
3. Click the "Change" button.
4. Enter the old password in the "Old" field.
5. Enter the new password in the "New" field.
6. Enter the new password again in the "Confirm" field.
7. Confirm your entries with "OK".
8. Close the "Know-how protection" dialog by clicking on "OK".

See also

Protecting blocks (Page 2929)

Setting up and removing block copy protection (Page 2931)

Setting up block know-how protection (Page 2932)

Opening know-how protected blocks (Page 2933)

Printing know-how protected blocks (Page 2933)

Removing block know-how protection (Page 2935)

9.1.7.7 Removing block know-how protection

Procedure

To remove block know-how protection, follow these steps:

1. Select the blocks for which you want to remove know-how protection.

   Note

   You can only remove know-how protection for several blocks at once if all blocks selected have the same password.

2. Select the command "Know-how protection" in the "Edit" menu.
   The "Know-how protection" dialog will open.

3. Deactivate the check box "Hide code (know-how protection)".
4. Enter the password.
5. Confirm your entries with "OK".

Result

Know-how protection will be disabled for the blocks selected.

See also

Protecting blocks (Page 2929)
Setting up and removing block copy protection (Page 2931)
Setting up block know-how protection (Page 2932)
Opening know-how protected blocks (Page 2933)
Printing know-how protected blocks (Page 2933)
Changing a password (Page 2935)

9.2 Displaying program information

9.2.1 Overview of available program information

Program information

The program information of a user program contains the view specified in the following table.

View	Application
Assignment list (Page 2937)	Provides an overview of the address bits for the I, Q, and M memory areas already allocated within the user program. Also indicates if an address has been allocated by access from an S7 program or if the address has been assigned to a SIMATIC S7 module.
Call structure (Page 2945)	Shows the call structure of the blocks within the user program and provides an overview of the blocks used and their relationships.

View	Application
Dependency structure (Page 2952)	Shows the list of blocks used in the user program. A block is shown at the first level and blocks that call or use this block are indented below it. In contrast to the call structure, instance blocks are listed separately.
Resources (Page 2957)	Shows the hardware resources of the CPU for objects (OB, FC, FB, DB, user-defined data types and PLC tags), for CPU memory areas and for the existing I/O modules.

Displaying several views simultaneously

You can generate and display several views for one or more user programs to facilitate testing and changing your user program.

Displaying multiple views, for example, enables you to:

- Display all program information for a user program next to one another
- Compare different user programs

9.2.2 Displaying an assignment list

9.2.2.1 Introduction to the assignment list

Program information in the assignment list

The assignment list shows if an address has been allocated by access from an S7 program or if the address has been assigned to a SIMATIC S7 module. It is therefore an important basis for locating errors or changes in the user program.

In the assignment list, you have a CPU-specific overview of which bit is used in which byte of the memory areas listed below:

- Input (I)
- Output (O)
- Bit memory (M)
- Timer (T)
- Counter (C)
- I/O (P)

Display of the assignment list

The assignment list of inputs, outputs, and bit memory is displayed in several separate work windows.

Filters

You can filter the display within the assignment list. You can use predefined filters or create your own.

Displaying cross-reference information

You have the option of displaying cross-reference information for selected addresses in the assignment list.

You can display the cross-references for a selected address in the Inspector window using the "Cross-reference information" shortcut menu command. The command "Tools > Cross-references" allows you to also open the cross-reference list for the selected object.

Displaying the PLC tag table

You can open the PLC tag table from the assignment list and edit the properties of the tags used.

To do this select an address of the assignment list and select the "Open editor" command in the shortcut menu.

Enabling the display of retentivity

You can enable and disable the display of the retentive state of bit memory by selecting the "Hide/show retain area" toolbar button.

See also

Symbols in the assignment list (Page 2939)

Layout of the assignment list (Page 2938)

9.2.2.2 Layout of the assignment list

Layout of the assignment list

Depending on the CPU, the assignment list is displayed in several work windows with the following operands.

For S7-300/400 CPUs:

- Inputs
- Outputs
- Bit memory
- Timers
- Counters

For S7-1200 CPUs:

- Inputs
- Outputs
- Bit memory

Displaying inputs, outputs, bit memory, timers and counters

It shows all operands used and their assignment in the S7 program.

For all displayed operands, each line in the assignment list is dedicated to a byte of the memory area, in which the corresponding eight bits from 7 to 0 are labeled according to their access. In conclusion, a "bar" indicates if access is made by a byte (B), word (W) or double word (D).

You can find an explanation of the symbols in the assignment list here. (Page 2939)

See also

Introduction to the assignment list (Page 2937)

9.2.2.3 Symbols in the assignment list

Meaning of the symbols in the assignment list

The following table shows the meaning of the symbols in the assignment list:

Symbol	Meaning
	Indicates the address assignment in the selected state.
	Indicates the address assignment in the non-selected state.
	Indicates that a pointer start address and a tag address access the same address range and that they are selected.
	Indicates that a pointer start address and a tag address access the same address range and that they are not selected.
	Indicates the pointer assignment in the selected state.
	Indicates the pointer assignment in the non-selected state.
	Indicates that the byte is in use with byte access and the corresponding tag is selected. The shortcut menu allows you to display cross-reference information for the selected variables as well as the PLC tag table.
	Indicates that the byte is in use with byte access and the corresponding tag is not selected.
	Indicates that the byte is in use with word access and the corresponding tag is selected. The shortcut menu allows you to display cross-reference information for the selected variables as well as the PLC tag table.
	Indicates that the byte is in use with word access and the corresponding tag is not selected.

Symbol	Meaning
	Indicates that the byte is in use with double word access and the corresponding tag is selected. The shortcut menu allows you to display cross-reference information for the selected variables as well as the PLC tag table.
	Indicates that the byte is in use with double word access and the corresponding tag is not selected.
Background color: gray	Indicates that a byte is in use with byte, word or double word access and that the address is also in use by the hardware. The gray background color indicates overlapping memory access.
Background color: yellow	Indicates that the address is not in use by the hardware.
	Indicates that the memory area has been defined as system memory.
	Indicates that the memory area has been defined as clock memory.

See also

Layout of the assignment list (Page 2938)

Introduction to the assignment list (Page 2937)

9.2.2.4 Displaying an assignment list

Requirement

A project has been created with programmed blocks.

Procedure

Proceed as follows to display the assignment list:

1. Select the "Program blocks" folder or one or more of the blocks it contains.
2. Select the "Assignment list" command in the "Tools" menu.

Result

The assignment list for the selected program is displayed.

View options in the assignment list

Refer to view respective view options that are set to display the desired information in the assignment list.

See also

Setting the view options for the assignment list (Page 2941)

Layout of the assignment list (Page 2938)

9.2.2.5 Setting the view options for the assignment list

Introduction

The following view options are available for the assignment list:

- Used addresses:
  When this check box is activated, the addresses, I/Os and pointers used in the program are displayed.
- Free hardware addresses:
  When this check box is activated, only the free hardware addresses are displayed.

Requirement

- A project has been created with programmed blocks.
- The assignment list is open.

Procedure

Proceed as follows to set the view options for the assignment list:

1. Click on the arrow of the symbol ("View options") in the task bar.
   The view options for the assignment list are opened. Check marks are set in front of the activated view options.
2. If you want to activate or deactivate a view option, click on the respective check box and set or remove the check mark.

Result

The view options are set and the desired information is displayed in the assignment list.

9.2.2.6 Filter options in the assignment list

Filter settings

You can define your own filter settings for the assignment list. The following options are available for defining filters:

- Display all addresses of the address areas specified.
- Display of single, defined addresses from the selected address area, for example, "0" and "200".
- Display of complete areas from the selected address area, for example, "0 - 256".

The following table provides an overview of all available options:

Selection in the	Selection	Symbol	Meaning
Address area	All CPU-dependent displayed addresses (I, O, M, T, C) can be activated as they are by default, or individual address areas can be activated.	Check box is activated	Only the activated address areas (I, O, M, T, C) are shown in the assignment list.
Filter area	Show assignment for all addresses	*	Displays the assignment of all addresses of the enabled address areas (I, Q, M).
	Show assignment for selected addresses, for example, for the inputs "IB 0" and "IB 256"	0;256 Separate individual addresses and areas by a semicolon.	Assignments of selected addresses for the activated address areas (I) are shown.
	Show assignment for selected areas, for example, for the inputs "IB 0 to IB 100" and "IB 200 to IB 256".	0-100;200-256 Contiguous areas should be connected by a hyphen.	Assignments of selected areas for the activated address areas (I) are shown.

9.2.2.7 Defining filters for assignment list

Requirement

- A project has been created with programmed blocks.
- The assignment list is open.

Defining filter

Proceed as follows to define a filter for the assignment list:

1. Click on the symbol ("Filter") in the task bar.
   The "Assignment List Filter" dialog opens.
2. Click on the symbol ("Create new filter") in the task bar.
   A new filter is created with the name "Filter_1". The check boxes for all addresses (inputs, outputs, memory bits, timers and counters) are activated by default for the filter.

3. If you want to change the name of the filter, click on the drop-down list in the task bar and enter a new filter name.
4. Deactivate the check boxes of addresses that are not to be affected by the filter.
5. Enter one of the following options in the filter area of the activated address:
   - Show all addresses used = "*"
   - Show single, defined addresses, for example, IB 0" and IB 25 = "0.25". Individual addresses and address areas are separated by commas or semicolons.
   - Show complete address areas, for example, IB 0 to IB 256 = "0-256". Complete address areas should be connected by a hyphen.
6. Confirm your entries with "OK".
   The newly defined filter is shown in the task bar of the assignment list under the specified name.

Delete filter

Proceed as follows to delete a filter:

1. Click on the symbol ("Filter") in the task bar.
   The filter dialog for the assignment list opens.
2. In the drop-down list of the task bar, select the filter you want to delete.
3. Click on the symbol ("Delete selected filter") in the task bar.
   The selected filter is deleted.

See also

Filter options in the assignment list (Page 2942)

Displaying an assignment list (Page 2940)

Introduction to the assignment list (Page 2937)

9.2.2.8 Filtering an assignment list

Requirement

- A project has been created with programmed blocks.
- The assignment list is open.

Procedure

1. Click on the arrow on the drop-down list.
   The available filter are displayed.
2. Select the desired filter.

Result

The assignment list is filtered according to the settings of the selected filter.

Note

The filter settings are saved when the project is closed.

9.2.2.9 Defining retentive memory areas for bit memories

Introduction

In the assignment list you can define the width of the retentive memory area for bit memories. The content of tags which are addressed in retentive memory is retained after power off and at the STOP to RUN transition after power on.

The display of retentive bit memories can be enabled and disabled in the assignment list. If their display is enabled, retentive bit memories are identified by an icon in the "Address" column.

Requirement

The assignment list is open.

Procedure

Proceed as follows to define the width of the retentive memory area for bit memories:

1. Click "Retain" in the toolbar.
   The "Retain memory" dialog will open.
2. Starting at the count of 0, define the width of the retentive memory area by entering the last byte of this area in the input field. Watch out for any addresses of tags already assigned to the retentive area.
3. Load the block to the target system. Select the "Program blocks" folder in the Project tree and select the "Download to device" submenu in the shortcut menu.

Result

The width of the retentive memory area is defined. If enabled in the assignment list, an icon will indicate the retentive state of all tags in the "Address" column.

9.2.2.10 Enabling the display of retentive bit memories

Introduction

In the assignment list you can enable and disable the display of retentive bit memories. The retentive bit memories are identified by means of an icon in the "Address" column if the display of retentivity is enabled.

Requirement

The assignment list is open.

Procedure

Proceed as follows to enable and disable the display of retentive bit memories:

1. Click "Display/hide retentivity" in the toolbar.

Result

The retentive tags are identified by means of an icon in the "Address" column of the bit memory area if the display of retentivity is enabled. The icons in the "Address" column are hidden if the display of retentivity is disabled.

9.2.3 Displaying the call structure

9.2.3.1 Introduction to the call structure

Call structure

The call structure describes the call hierarchy of the block within an S7 program.

It provides an overview of:

- The blocks used
- Jumps to the places of use of the blocks
- Relationships between blocks
- Local data requirements of the blocks
- Status of the blocks

Information in the call structure

Displaying the call stucture provides you with a list of the blocks used in the user program. The first level of the call structure is highlighted in color and shows the blocks that are not called by any other block in the program. Organization blocks are always shown on the first level of the call structure. Functions, function blocks and data blocks are only shown on the first level if they are not called by an organization block. When a block calls other blocks or functions, they are listed indented under the calling block. Instructions and blocks are shown in the call structure only if they are called by a block.

View options

The following view options are available for the call structure:

- Show conflicts only:
  When this check box is activated, only the conflicts within the call structure are displayed.
- Group multiple calls together:
  When this check box is activated, several block calls are grouped together. The number of block calls is displayed in the "Call frequency" column. The links to the various call locations are offered in a drop-down list in the "Details" column.

Displaying the block calls

You can display the block calls in a block by clicking on the arrow in front of the block title. To display the call information of all blocks, click on the "Expand list" icon in the toolbar.

You can hide the total overview by clicking the "Collapse list" icon.

Displaying cross-reference information

You can display the cross-reference information for a block in the Inspector window by right-clicking on the relevant block and selecting the "Cross-reference information" command from the shortcut menu.

To open the "Cross-references" view, click the "Cross-references" command in the shortcut menu.

Displaying blocks in the program editor

You can open the program editor and edit blocks there from the call structure.

To do this select the required block in the call structure and select the "Open editor" command in the shortcut menu.

Displaying deleted blocks

The rows belonging to deleted blocks are identified by an icon.

Note

Please note that any existing local data can only be displayed or updated after compiling a block.

See also

Symbols in the call structure (Page 2947)

9.2.3.2 Symbols in the call structure

Meaning of the symbols in the call structure

The following table shows the meaning of the symbols in the call structure:

Symbol	Meaning
	Indicates an organization block (OB).
	Indicates a function block (FB).
	Indicates a function (FC).
	Indicates a data block (DB).
	Indicates that the block is declared as a multiinstance.
	The object has an interface dependency to an object connected to the left.
	Indicates that the block needs to be compiled again.
	Indicates that the data block needs to be compiled again.
	Indicates that the object is not available.
	Indicates that the interface causes a time stamp conflict.
	Indicates that the variable causes a time stamp conflict.
	Indicates that the block is not called directly or indirectly from an OB.
	Indicates that an object has know-how protection.

Symbol	Meaning
	Indicates that the block is normally called recursively.
	Indicates that a tag declaration in the interface has a recursive dependency: • Scenario 1: FB1 calls FB2 and this then calls FB1. The instance data blocks of these FBs have a recursion in the interface. • Scenario 2: A multiple instance FB uses the instance DB of its parent FB as a global DB.

9.2.3.3 Layout of the call structure

Layout of the call structure

The view of the call structure consists of the following columns:

Column	Content/meaning
Call structure	Shows an overview of the blocks called If the viewing option "Group multiple calls together" is enabled, several block calls are grouped together and the "Number of calls" column is displayed.
Call type (!)	Shows the type of call, for example recursive block call.
Address	Shows the absolute address of the block. With a function block, the absolute address of the corresponding instance data block is also shown.
Call frequency	Indicates the number of multiple calls of blocks.
Details	Shows the network or interface of the calling block. All information are offered as a link in this column. With this link, you can jump to the location of the block call in the program editor. If the viewing option "Group multiple calls together" option is enabled, the calls are grouped together and are available as links in a drop-down list.
Local data (in path)	Indicates the local data requirement of the full path. Blocks with optimized access have higher local data requirements because the information for the symbolic addressing is stored with them. Please note that any existing local data can only be displayed or updated after compiling a block.
Local data (for blocks)	Show the local data requirements of the block. Blocks with optimized access have higher local data requirements because the information for the symbolic addressing is stored with them. Please note that any existing local data can only be displayed or updated after compiling a block.

See also

Symbols in the call structure (Page 2947)
Introducing the consistency check in the call structure (Page 2950)

9.2.3.4 Displaying the call structure

Requirement

A project has been created with blocks.

Procedure

Proceed as follows to display the call structure:

1. Select the "Program blocks" folder or one or more of the blocks it contains.
2. Select the "Call structure" command in the "Tools" menu.

Result

The call structure for the selected program is displayed.

Note

Please note that any existing local data can only be displayed or updated after compiling a block.

See also

Setting the view options for the call structure (Page 2950)

9.2.3.5 Setting the view options for the call structure

Introduction

The following view options are available for the call structure:

- Show conflicts only:
  Only the blocks causing conflicts within the call structure are displayed if this check box is activated.
  The following blocks cause conflicts:
  - Blocks executing any calls with older or newer code time stamps.
  - Blocks calling a block with modified interface.
  - Blocks using a tag with modified address and/or data type.
  - Block called neither directly, nor indirectly by an OB.
  - Blocks calling a block which no longer exists.
- Group multiple calls together:
  When this viewing option is enabled, several block calls and data block accesses are grouped together. The number of block calls is displayed in the "Call frequency" column. The links to the various call locations are offered in a drop-down list in the "Details" column.

Requirement

- A project has been created with programmed blocks.
- The call structure is open.

Procedure

Proceed as follows to set the view options for the call structure:

1. Click on the arrow of the symbol ("View options") in the task bar.
   The view options for the call structure opens. Check marks are set in front of the activated view options.
2. If you want to activate or deactivate a view option, click on the respective check box and set or remove the check mark.

Result

The view options are set and the required information is displayed in the call structure.

9.2.3.6 Introducing the consistency check in the call structure

Consistency check

Changing the time stamp of a block during or after the program is generated can lead to time stamp conflicts, which in turn cause inconsistencies among the blocks that are calling and being called.

Using the consistency check

The "Consistency check" function is used to visualize inconsistencies when time stamp conflicts occur. Whe the consistency check is performed, the inconsistent blocks are shown in the call structure and marked with the correspoinding symbols.

- Most time stamp and interface conflicts can be rectified by recompiling the blocks.
- If compilation fails to clear up inconsistencies you can use the link in the "Details" column to go to the source of the problem in the program editor and manually eliminate any inconsistencies.
- The blocks marked in red must be recompiled.

See also

Symbols in the call structure (Page 2947)

9.2.3.7 Checking block consistency in the call structure

Requirement

- A project has been created with programmed blocks.
- The call structure is open.

Procedure

Proceed as follows to check the block consistency:

1. Click on the symbol ("Consistency check") in the task bar.
   The block consistency is checked. Blocks found to be inconsistent are marked accordingly by a symbol.
2. If a block is inconsistent, click on the arrow in front of the block title in the call structure.
   The inconsistent blocks are displayed. The exact problem locations are listed as links in the "Details" column.
3. Click on the respective link in the "Details" column to jump to the location in the block requiring correction.
4. Check and correct the inconsistencies in the blocks.
5. Recompile the blocks by selecting the required blocks and clicking on the command "Compile" in the shortcut menu.
6. Download the corrected blocks to the target system by clicking the command "Download to device" in the shortcut menu.

Result

The block consistency is checked. The inconsistencies in the blocks are corrected. The corrected blocks are loaded to the target system.

See also

Symbols in the call structure (Page 2947)

9.2.4 Displaying the dependency structure

9.2.4.1 Introduction to the dependency structure

Introduction

The dependency structure shows the dependencies each block has to other blocks in the program.

Information in the dependency structure

Displaying the dependency structure provides you with a list of the blocks used in the user program. A block is shown at the far left and blocks that call or use this block are indented below it.

The dependency structure also shows the status of the individual blocks using symbols.

Objects causing a time stamp conflict and perhaps leading to an inconsistency in the program are marked with various symbols.

The dependency structure is an extension of the cross-reference list for objects.

View options

The following view options are available for the dependency structure:

- Show conflicts only:
  When this check box is activated, only the conflicts within the dependency structure are displayed.
- Group multiple calls together:
  When this check box is activated, several block calls are grouped together. The number of block calls is shown numerically in the "Dependency structure" column. The links to the various call locations are offered in a drop-down list in the "Details" column.

Displaying the dependency structure

Clicking on the arrow in front of the block title displays the blocks that call or use this block. To display the dependencies of all blocks,

click the "Expand list" icon in the toolbar.

You can hide the total overview by clicking the "Collapse list" icon.

Displaying cross-reference information

You can display the cross-reference information for a block in the Inspector window by right-clicking on the respective block and selecting the "Display Usage" command from the shortcut menu.

Displaying blocks in the program editor

You can open the program editor and edit blocks there from the dependency structure. To do this select the required block in the dependency structure and select the "Open editor" command in the shortcut menu.

9.2.4.2 Layout of the dependency structure

Layout of the dependency structure

The view of the dependency structure consists of the following columns:

Column	Content/meaning
Dependency	It indicates the dependencies between each block and the other blocks in the program.
Call type (!)	Shows the type of call, for example recursive block call.
Address	Shows the absolute address of the block.
Call frequency	Indicates the number of multiple calls of blocks.
Details	Shows the network or interface of the called block. All information are offered as a link in this column. With this link, you can jump to the location of the block call in the program editor. If the viewing option "Group multiple calls together" option is enabled, the calls are grouped together and are available as links in a drop-down list.

See also

Symbols in the dependency structure (Page 2954)

9.2.4.3 Symbols in the dependency structure

Meaning of the symbols in the dependency structure

The following table shows the meaning of the symbols in the dependency structure:

Symbol	Meaning
	Indicates an organization block (OB).
	Indicates a function block (FB).
	Indicates a function (FC).
	Indicates a data block (DB).
	The object has an interface dependency to an object connected to the left.
	Indicates that the block needs to be compiled again.
	Indicates that the data block needs to be compiled again.
	Indicates that there is an inconsistency with this object.
	Indicates that an object has know-how protection.
	Indicates that a tag declaration in the interface has a recursive dependency: • Scenario 1: FB1 calls FB2 and this then calls FB1. The instance data blocks of these FBs have a recursion in the interface. • Scenario 2: A multiple instance FB uses the instance DB of its parent FB as a global DB.

9.2.4.4 Displaying the dependency structure

Requirement

A project has been created with programmed blocks.

Procedure

Proceed as follows to display the dependency structure:

1. Select the block folder or one or more of the blocks contained therein.
2. Select the "Dependency structure" command in the "Tools" menu.

Result

The dependency structure for the selected program is displayed.

See also

Setting the view options for the dependency structure (Page 2955)

9.2.4.5 Setting the view options for the dependency structure

Introduction

The following view options are available for the dependency structure:

- Show conflicts only:
  When this check box is activated, only the conflicts within the dependency structure are displayed.
  The following blocks cause conflicts:
  - Blocks executing any calls with older or newer code time stamps.
  - Blocks called by a block with modified interface.
  - Blocks using a tag with modified address and/or data type.
  - Block called neither directly, nor indirectly by an OB.
- Group multiple calls together:
  When this check box is activated, several block calls are grouped together. The number of block calls is shown in the relevant column. The links to the various call locations are offered in a drop-down list in the "Details" column.

Requirement

- A project has been created with programmed blocks.
- The dependency structure is open.

Procedure

Proceed as follows to set the view options for the dependency structure:

1. Click on the arrow of the symbol ("View options") in the task bar.
   The view options for the dependency structure are opened. Check marks are set in front of the activated view options.
2. If you want to activate or deactivate a view option, click on the respective check box and set or remove the check mark.

Result

The view options are set and the required information is displayed in the dependency structure.

9.2.4.6 Introducing the consistency check in the dependency structure

Consistency check

Changing the time stamp of a block during or after the program is generated can lead to time stamp conflicts, which in turn cause inconsistencies among the blocks that are calling and being called.

Using the consistency check

The "Consistency check" function is used to visualize inconsistencies. Whe the consistency check is performed, the inconsistent blocks are shown in the dependency structure and marked with the correspoinding symbols.

- Most time stamp and interface conflicts can be rectified by recompiling the blocks.
- If compilation fails to clear up inconsistencies you can use the link in the "Details" column to go to the source of the problem in the program editor and manually eliminate any inconsistencies.
- The blocks marked in red must be recompiled.

See also

Layout of the dependency structure (Page 2953)

Symbols in the dependency structure (Page 2954)

9.2.4.7 Checking block consistency in the dependency structure

Requirement

- A project has been created with programmed blocks.
- The dependency structure is open.

Procedure

Proceed as follows to check the block consistency:

1. Click on the symbol ("Consistency check") in the task bar.
   The block consistency is checked. Blocks found to be inconsistent are marked accordingly by a symbol.
2. If a block is inconsistent, click on the arrow in front of the block title in the dependency structure.
   The inconsistent blocks are displayed. The exact problem locations are listed as links in the "Details" column.
3. Check and correct the inconsistencies in the blocks.

4. Recompile the blocks by selecting the required blocks and clicking on the command "Compile" in the shortcut menu.
5. Download the corrected blocks to the target system by clicking the command "Download to device" in the shortcut menu.

Result

The block consistency is checked. The inconsistencies in the blocks are corrected. The corrected blocks are loaded to the target system.

See also

Symbols in the dependency structure (Page 2954)

9.2.5 Displaying CPU resources

9.2.5.1 Introducing resources

Introduction

The "Resources" tab indicates the hardware resources of the configured CPU for:

- the used programming objects,
- the assignment of the different memory areas within the CPU and
- the assigned inputs and outputs of the existing input and output modules.

Information provided in the "Resources" tab

The resources tab provides an overview of the hardware resources. The display in this tab depends on the CPU which you are using. The following information is displayed:

- the programming objects used in the CPU (e.g. OB, FC, FB, DB, data types and PLC tags)
- the memory areas available on the CPU (load memory, work memory - divided into code work memory and data work memory depending on the CPU -, retentive memory), their maximum size and utilization by the programming objects stated above
- the I/O of modules which can be configured for the CPU (I/O modules, digital input modules, digital output modules, analog input modules, and analog output modules), including the I/O already in use.

Display of the maximum available load memory

The maximum size of available load memory can be selected from a drop-down list box in the "Total" row of the "Load memory" column.

Display of the maximum available work memory

The maximum size of available work memory is displayed in the "Work memory" column or in the "Code work memory" and "Data work memory" columns in the "Total" row.

Display of the maximum available retentive memory

The maximum size of available retentive memory can be selected from a drop-down list box in the "Total" row of the "Retentive memory" column.

Note

Retentive memory data

All bit memories and data blocks specified as retentive will be integrated in the calculation of the retentive data.

Updating the display in the "Resources" tab

Click the "Update view" toolbar button to update the display of objects.

Benefits of the display in the "Resources" tab

The "Resources" tab of the program information dialog provides a detailed list of all objects and of the corresponding memory area used.

The tab also indicates shortage of resources and helps to avoid such states.

Blocks which are not compiled can be identified as their size is indicated by a question mark.

See also

Layout of the "Resources" tab (Page 2959)

Displaying resources (Page 2960)

Selecting the maximum load memory available (Page 2961)

9.2.5.2 Layout of the "Resources" tab

Layout of the "Resources" tab in the program information

The view of the "Resources" tab consists of the following columns:

Column	Content/meaning
Objects	The "Details" area provides an overview of the programming objects available in the CPU, including their memory assignments.
Load memory	Displays the maximum load memory resources of the CPU as a percentage and as absolute value. The values displayed under "Total" provide information on the maximum memory available in the load memory. The values displayed under "Used" provide information on the memory actually used in the load memory. If a value is displayed in red, the available memory capacity has been exceeded.
Work memory or code and data work memory	Displays the maximum work memory resources of the CPU as a percentage and as absolute value. The work memory depends on the CPU and is divided into "Code work memory" and "Data work memory" for a CPU from the S7-400 or S7-1500 series, for example. The values displayed under "Total" provide information on the maximum memory available in the work memory. The values displayed under "Used" provide information on the memory space actually used in the work memory. If a value is displayed in red, the available memory capacity has been exceeded.
Retentive memory	Displays the maximum resources for retentive memory in the CPU as a percentage and as absolute value. The values displayed under "Total" provide information on the maximum memory available in the retentive memory. The values displayed under "Used" provide information on the memory actually used in the retentive memory. If a value is displayed in red, the available memory capacity has been exceeded.

Column	Content/meaning
I/O	Displays the I/Os which are available on the CPU, including their module-specific availability in the next columns. The values displayed at "Configured" provide information about the maximum number of I/O available. The values displayed under "Used" provide information on the actually used inputs and outputs.
DI / DQ / AI / AQ	Displays the number of configured and used inputs/outputs: DI = Digital inputs DQ = Digital outputs AI = Analog inputs AQ = Analog outputs The values displayed at "Configured" provide information about the maximum number of I/O available. The values displayed under "Used" provide information on the actually used inputs and outputs.

See also

Displaying resources (Page 2960)

Selecting the maximum load memory available (Page 2961)

Introducing resources (Page 2957)

9.2.5.3 Displaying resources

Requirement

A project with programmed blocks has been created.

Procedure

Proceed as follows to display the resources of the respective CPU memory areas:

1. Select the block folder below the relevant CPU, or one or several of the blocks contained therein.
2. Select the "Resources" command in the "Tools" menu.

Result

The memory resources of the assigned CPU are displayed.

9.2.5.4 Selecting the maximum load memory available

Requirement

A project with programmed blocks has been created.

Procedure

Proceed as follows to display the available maximum of load memory resources:

1. Select the block folder below the relevant CPU, or one or several of the blocks contained therein.
2. Select the "Resources" command in the "Tools" menu.
3. In the dialog that is displayed, open the drop-down list in the "Total" field of the "Load memory" column by clicking the icon.
4. Select a corresponding value for the CPU used by clicking it in the drop-down list box.

Result

The "Total" field displays the selected maximum memory resources.

Note

Display of maximum memory resources

If a value is displayed in red for the maximum memory resources, the available memory capacity has been exceeded.

In this case, adapt the memory capacity as described above.

9.3 Displaying cross-references

9.3.1 General information about cross references

Introduction

The cross-reference list provides an overview of the use of operands and tags within the user program.

Uses of cross-references

The cross-reference list offers you the following advantages:

- When creating and changing a program, you retain an overview of the operands, tags and block calls you have used.
- From the cross-references, you can jump directly to the point of use of operands and tags.
- During a program test or when troubleshooting, you are informed of the following:
  - Which operand is processed by which command in which block.
  - Which tag is used in which picture.
  - Which block is called by which other block.
  - Cross-reference information for subordinate and higher-level structures.
- As part of the project documentation, the cross-references provide a comprehensive overview of all operands, memory areas, blocks, tags and pictures used.

See also

Structure of the cross-reference list (Page 2962)

Displaying the cross-reference list (Page 2963)

Displaying cross-references in the Inspector window (Page 2965)

9.3.2 Structure of the cross-reference list

Views of the cross-reference list

There are two views of the cross-reference list. The difference between the two views is in the objects displayed in the first column:

- Used by:
  Display of the referenced objects. Here, you can see where the object is used.
- Used:
  Display of the referencing objects. Here, you can see the users of the object.

The assigned tool tips provide additional information about each object.

Structure of the cross-reference list

The cross-reference list has the following structure:

Column	Content/meaning
Object	Name of the object that uses the lower-level objects or that is being used by the lower-level objects.
Number	Number of uses
Point of use	Each point of use, for example, network

Column	Content/meaning
Property	Special properties of referenced objects, for example, the tag names in multi-instance declarations.
as	Shows additional information about the object, e.g., that an instance DB is used as template or as multiple instance.
Access	Type of access, whether access to the operand is read access (R) and/ or write access (W).
Address	Address of the operand
Type	Information on the type and language used to create the object
Path	Path of object in project tree

Depending on the installed products, additional columns or different columns are displayed for the cross-references.

Settings in the cross-reference list

You can make the following settings using the buttons in the toolbar of the cross-reference list:

- Update cross-reference list
  Updates the current cross-reference list.
- Making settings for the cross-reference list
  Here, you select check boxes to specify whether all used, all unused, all defined or all undefined objects will be displayed. If the "Undefined objects" option is enabled, references to previously deleted objects are also displayed.
- Collapse entries
  Reduces the entries in the current cross-reference list by closing the lower-level objects.
- Expand entries
  Expands the entries in the current cross-reference list by opening the low-level objects.

Sorting in the cross-reference list

You can sort the entries in the "Object" column, including other product-specific columns, in ascending or descending order. To do this, click on the relevant column title.

See also

General information about cross references (Page 2961)

Displaying the cross-reference list (Page 2963)

9.3.3 Displaying the cross-reference list

Requirements

You have created a project.

Introduction

There are several ways of displaying cross-references depending on whether you are in the Portal view or in the Project view and which object you have selected in the project tree.

In the Portal view, you can only display cross-references for the entire CPU; in the Project view, you can, for example, display cross-references for the following objects:

- "PLC tags" folder
- "PLC data types" folder
- "Program blocks" folder
- "Tags and connections" folder
- Individual tags
- Individual PLC data types
- Individual blocks
- Technological objects
- Watch tables

Displaying cross-references

Proceed as follows to display cross-references:

1. Select the required action in the Portal view, for example "PLC programming" and the "Show cross-references" command or select one of the objects listed above in the Project view and select the "Cross-references" command in the "Tools" menu.
   The cross-reference list is displayed.
2. Click the "Used by" button to display where the objects shown in the cross-reference list are used.
3. Click the "Uses" button to view the users of the objects displayed in the cross-reference list.
4. You can perform the following actions using the buttons in the toolbar:
   - Update cross-reference list
   - Making settings for the cross-reference list
   - Collapse entries
   - Expand entries
5. You can sort the entries in the "Object" and "Address" columns in ascending or descending order by clicking on the relevant column title.
6. To go to the point of use of the object, click on the displayed link.

See also

General information about cross references (Page 2961)

Structure of the cross-reference list (Page 2962)

9.3.4 Displaying cross-references in the Inspector window

Introduction

The Inspector window displays cross-reference information about an object you have selected in the "Info > Cross-references" tab. This tab displays the instances where a selected object is being used and the other objects using it.

The Inspector window also includes blocks which are only available online in the cross-references.

You can use the "Show overlapping access..." shortcut menu command to also have overlapping access across block borders displayed for selected objects.

Structure

The Inspector window displays the cross-reference information in tabular format. Each column contains specific and detailed information on the selected object and its application. The table below shows the additional information listed in the "Info > Cross-reference" tab:

Column	Meaning
Object	Name of the object that uses the lower-level objects or that is being used by the lower-level objects.
Number	Number of uses
Point of use	Each location of use, for example, network
Property	Special properties of referenced objects, for example, the tag name in multi-instance declarations
as	Shows additional information about the object, e.g., that an instance DB is used as template or as multiple instance.
Access	Access mode Shows whether the operand is accessed by a read (R) and/or write (W) operation.
Address	Address of the operand
Monitor value	This column will only be displayed when the program editor is open.
Type	Information about the type and language used to create the object
Path	Path of object in project tree

Depending on the installed products, additional columns or different columns are displayed for the cross-references.

9.4 Testing the user program

9.4.1 Basics of testing the user program

Functions

You have the option of testing the running of your user program on the device. You can then monitor signal states and values of tags and can assign values to tags to simulate certain situations in the running of the program.

Requirement

There must be an executable program loaded on the device.

Test options

The following test options are available:

- Testing with program status
  The program status allows you to monitor the running of the program. You can display the values of operands and the results of logic operations (RLO) allowing you to recognize and fix logical errors in your program.
- Testing in single step mode (S7-300/400 only)
  You can test blocks you created in STL or SCL in the single step mode. You do this by setting breakpoints in the program code at which program execution stops. You can then continue to run the program one step at a time. Within a CPU, you can test either with program status or in single step mode. You cannot, however, use both test options at the same time within a CPU.
- Testing with the watch table
  With the watch table, you can monitor and modify the current values of individual tags in the user program or on a CPU. You can assign values to individual tags for testing and run the program in a variety of different situations. You can also assign fixed values to the I/O outputs of a CPU in STOP mode, for example to check the wiring.
- Testing with the force table
  With the force table, you can monitor and force the current values of individual tags in the user program or on a CPU. When you force, you overwrite individual tags with specified values. This allows you to test your user program and run through various situations. When forcing, make sure that you keep to the necessary safety measures for forcing (Page 3039)!

See also

Introduction to testing with program status (Page 2967)

Introduction to testing with the watch table (Page 2994)

Introduction for testing with the force table (Page 3021)

9.4.2 Testing with program status

9.4.2.1 Introduction to testing with program status

Program Status function

If you display the program status, you can monitor the execution of the program. This provides you with an overview of the values of the individual operands and the results of the logic operations and you can check whether the components of the automation system are correctly controlled.

The display of the program execution in the program status can differ slightly, depending on the CPU family used.

Testing with program status for S7-300/400

During testing with program status, the CPU cycle time can become extended in test mode because the recording of all test data can deviate from the duration of the programmed instructions due to the CPU capacity and therefore not run in realtime.

During the execution of the following test functions an alarm indicating the danger of a time-out is displayed once for each online session.

- During testing with call conditions
- During testing with breakpoints

You can only perform these test functions after you have acknowledged the alarm.

Note

With older CPUs from the S7-300/400 CPU family, you will need to change the operating response using the hardware configuration and then download the hardware configuration to the device. You have the option to set the "Process operation" or "Test operation" operating response.

Restrictions with the "Program status" function

The monitoring of loops can increase the cycle time significantly, depending in each case on the number of tags to be monitored and on the actual number of loops processed.

To ensure that the cycle time is influenced as little as possible, the "Program status" function is restricted as follows:

- The status display of a programmed loop is stopped at the return point.

> **WARNING**
>
> **Testing with program status**
>
> A test with the "Program status" function can cause serious damage to property or injury to persons if there are functional disturbances or program errors.
>
> Make sure that no dangerous situations can arise before you conduct a test with the "Program status" function.

9.4.2.2 Setting the call environment

Basics of the call environment

Function

You can define the call environment for blocks and for breakpoints. With this, you specify the conditions for recording the program status of a block or for stopping program execution at a breakpoint. You can select one of the following conditions:

- No condition defined
  If no other option was selected this option is the default.
- Instance data block
  The program status of a function block will only be recorded when you call the function block with the selected instance data block.
- Call path
  The program status of a block will only be recorded when you call the block with a specific block or from a specific path.

You can also specify a separate condition for each breakpoint. If you do not specify the call environment, the program status of a block call is selected at random and recorded within the call structure and program execution is always interrupted at the relevant breakpoint. You should therefore always specify the call environment if you want to display the program status for a specific call.

See also

Setting the call environment of the block (Page 2968)

Setting the call environment of the block

By setting the call environment, you can specify when the program status of a block is recorded.

The section "Setting the call environment for breakpoints (Page 2972)" describes how to set the call environment for breakpoints.

Requirements

- The block is open.

Procedure

To set the call environment, follow these steps:

1. Open the "Testing" task card.
2. Click the "Change" button in the "Call environment" pane.
   The "Call environment of a block" dialog opens.
3. Select the condition you want to apply.
   See also: Basics of the call environment (Page 2968)
4. Confirm your selection with "OK".

See also

Introduction to testing with program status (Page 2967)

9.4.2.3 Testing in single step mode

Basics of testing in single step mode

Introduction

You can test blocks you created in the textual programming languages STL and SCL in single step mode. You do this by first setting breakpoints in the program code, establishing an online connection, and enabling the breakpoints on the CPU. Following this, you can execute the program up to the first breakpoint and then run the program step-by-step with the following functions:

- Run
  Continue to the next enabled breakpoint.
- Run to cursor
  Continue to the cursor position.
- Step over
  The currently selected instruction is executed.
- Step into
  Jump into a lower-level block or call a lower-level block.
- Step out
  If a block call is being processed and the program execution in the called block was interrupted at a breakpoint, you can use "Step out" to jump back to the called block.

You can set any number of breakpoints in the program code. However, the number of breakpoints you can enable on the CPU depends on the particular CPU. For individual breakpoints, you can use the call environment to specify when program execution is interrupted at the breakpoints.

If you terminate the online connection to the CPU, the breakpoints are disabled.

Note

The breakpoints are not saved with the project. If you close the project, the breakpoints are lost.

See also

Setting breakpoints (Page 2970)

Setting the call environment for breakpoints (Page 2972)

Enabling and disabling breakpoints (Page 2972)

Running a test in single step mode (Page 2974)

Deleting breakpoints (Page 2976)

Setting breakpoints

Requirement

The block was created in STL or SCL.

Procedure

To set a breakpoint in the program code, follow these steps:

1. Open the block in which you want to set a breakpoint.
2. Right-click in the line in which you want to set the breakpoint and double-click in the gray column in front of the line number.
3. Select the "Set breakpoint" command in the shortcut menu.

Or:

1. Open the block in which you want to set a breakpoint.
2. Click in the line in which you want to set the breakpoint and double-click in the gray column in front of the line number.

Result

A breakpoint is set in the program code. The breakpoint is displayed in the "Breakpoints" pane of the "Testing" task card and the breakpoint is also displayed as a grey circle in front of the corresponding line in the program code. Before the program is interrupted at the newly set breakpoint, the breakpoint must first be enabled.

See also

Basics of testing in single step mode (Page 2969)

Setting the call environment for breakpoints (Page 2972)

Enabling and disabling breakpoints (Page 2972)

Running a test in single step mode (Page 2974)

Deleting breakpoints (Page 2976)

Navigating to the breakpoints

To navigate to the breakpoints in your program, you can either jump from breakpoint to breakpoint or to the point in the program where a specific breakpoint is located.

Navigating from breakpoint to breakpoint

To navigate from breakpoint to breakpoint in your program, follow these steps:

1. Open a block and select the programming editor.
2. In the "Online > Breakpoints" menu, select the "Next" command to jump to the next breakpoint after the current cursor position. When you reach the last breakpoint, you jump to the first breakpoint.
3. In the "Online > Breakpoints" menu, select the "Previous" command to jump to the previous breakpoint before the current cursor position. When you reach first breakpoint, you jump to the last breakpoint.

Navigating to a specific breakpoint

To navigate to a specific breakpoint in your program, follow these steps:

1. Open the "Testing" task card.
   In the "Breakpoints" pane, you will see a list with all the breakpoints set for the current CPU.
2. Double-click on the breakpoint to which you want to navigate.
   The block with the breakpoint is displayed in the programming editor and the line with the breakpoint is highlighted.

Or:

1. Open the "Testing" task card.
   In the "Breakpoints" pane, you will see a list with all the breakpoints set for the current CPU.
2. Right-click on the breakpoint to which you want to navigate.
3. Select the "Go to breakpoint" command in the shortcut menu.
   The block with the breakpoint is displayed in the programming editor and the line with the breakpoint is highlighted.

Setting the call environment for breakpoints

By setting the call environment, you can specify when the program is interrupted at the breakpoints.

Requirement

Breakpoints exist in the program.

Procedure

To set the call environment for a breakpoint, follow these steps:

1. Open the "Testing" task card.
2. On the "Breakpoints" pane, click the "..." button after the breakpoint for which you want to set the call environment.
   The "Call environment of a block" dialog opens.
3. Select the condition you want to apply.
   See also: Basics of the call environment (Page 2968)
4. Confirm your selection with "OK".
   The call environment is specified for the breakpoint and the symbol for the breakpoint changes from a single gray circle to a gray circle with vertical lines before and after it.

See also

Basics of testing in single step mode (Page 2969)

Setting breakpoints (Page 2970)

Enabling and disabling breakpoints (Page 2972)

Running a test in single step mode (Page 2974)

Deleting breakpoints (Page 2976)

Enabling and disabling breakpoints

To be able to use breakpoints when testing, the set breakpoints must first be enabled. By enabling them, the breakpoints are installed and enabled on the CPU. You can enable and disable breakpoints either individually or all together. After you have disabled breakpoints, they are ignored when the program executes.

Note

The number of breakpoints that can be enabled on the CPU depends on the particular CPU. Once you have reached the maximum number, you cannot enable any more breakpoints. Since the single step functions "Step over", "Step into" and "Run to cursor" also count as breakpoints, these can also no longer be used. Before you can enable further breakpoints or use these single step functions, you first need to disable already enabled breakpoints.

Requirements

- Breakpoints have been set.
- There is an online connection to the CPU.
- The CPU you are using does not support the use of breakpoints.
- The maximum number of breakpoints that can be enabled for the CPU has not been reached.

Enabling individual breakpoints

To enable individual breakpoints, follow these steps:

1. Open the "Testing" task card.
   In the "Breakpoints" pane, you will see a list with all the breakpoints set for the current CPU.
2. Right-click on the breakpoint you want to enable.
3. Select the "Enable breakpoint" command in the shortcut menu.
   The selected breakpoint is installed on the CPU and enabled and the program is interrupted at the breakpoint when it reaches that point in the program. The display of the breakpoint changes from a gray circle to a red circle.

Activate all breakpoints

To enable all breakpoints, follow these steps:

1. Open the "Testing" task card.
   In the "Breakpoints" pane, you will see a list with all the breakpoints set for the current CPU.
2. Make sure that no breakpoint is selected.
3. Click "Enable/disable breakpoint" in the toolbar of the "Breakpoints" pane.
   Depending on the capacity of the CPU, all breakpoints are installed and enabled on the CPU and the program is interrupted at all breakpoints that are reached when it executes. If you have set more breakpoints than allowed by the CPU, only the permitted number of breakpoints is enabled. The display of the enabled breakpoints changes from gray circles to red circles.

Disabling individual breakpoints

To disable individual breakpoints, follow these steps:

1. Open the "Testing" task card.
   In the "Breakpoints" pane, you will see a list with all the breakpoints set for the current CPU.
2. Right-click on the breakpoint you want to enable.
3. Select the "Disable breakpoint" command in the shortcut menu.
   The selected breakpoint is uninstalled on the CPU and disabled and the program is no longer interrupted at the breakpoint. The display of the breakpoint changes from a red circle to a gray circle.

Deactivate all breakpoints

To disable all breakpoints, follow these steps:

1. Open the "Testing" task card.
   In the "Breakpoints" pane, you will see a list with all the breakpoints set for the current CPU.
2. Make sure that no breakpoint is selected.
3. Click "Enable/disable breakpoint" in the toolbar of the "Breakpoints" pane.
   All breakpoints are uninstalled on the CPU and disabled and the program is no longer interrupted at any breakpoint. The display of the breakpoints changes from red circles to gray circles.

See also

Basics of testing in single step mode (Page 2969)

Setting breakpoints (Page 2970)

Setting the call environment for breakpoints (Page 2972)

Running a test in single step mode (Page 2974)

Deleting breakpoints (Page 2976)

Running a test in single step mode

If the program has been interrupted at a breakpoint, you can continue the program step-by-step.

Requirements

- Breakpoints are set and enabled in the program code.
- The maximum number of breakpoints that can be enabled for the CPU has not been reached.
- The block is open.

Procedure

To run the program in single step mode, follow these steps:

1. Establish an online connection to the CPU.
2. Enable the breakpoints.
   The program code executes as far as the enabled breakpoint and is then interrupted, in other words, the CPU changes to "HOLD" mode. The red circle for the breakpoint has a yellow arrow superimposed on it.

3. When using the "Run" and "Run to cursor" functions, if you want to enable the outputs at the next change to "RUN" mode, select the "Enable output in run" check box in the "Breakpoints" pane of the "Testing" task card.
4. To continue running the program step by step, select one of the following commands from the toolbar:
   - Run
     The program is continued. If there are other breakpoints, the program is interrupted at the next breakpoint.
   - Run to cursor
     The program continues as far as the point at which the insert point is set. If there is a further breakpoint before the insert point, the program is interrupted first at this breakpoint.
   - Step over
     The next instruction after the breakpoint is executed. If the instruction is a block call, the call is processed and execution jumps to the next instruction following the block call.
   - Step into
     The next instruction after the breakpoint is executed. If the instruction is a block call, execution jumps to the called block. There, you can continue testing in single step mode or set further breakpoints. At the end of the block, execution jumps back to the next instruction following the block call.
   - Step out
     If a block call is being processed and the program execution in the called block was interrupted at a breakpoint, you can use "Jump back" to jump back to the called block. In the process the remaining instructions of the called block will be executed and the program execution is halted again in the calling block after the block call.

See also

Basics of testing in single step mode (Page 2969)

Setting breakpoints (Page 2970)

Setting the call environment for breakpoints (Page 2972)

Enabling and disabling breakpoints (Page 2972)

Deleting breakpoints (Page 2976)

Introduction to testing with program status (Page 2967)

Exiting testing in single step mode

Procedure

To stop running the program in single step mode, follow these steps:

1. Open the "Testing" task card.
2. Disable the breakpoints of the CPU.
3. Select the "Enable output in run" check box.

4. Right-click in the gray column in front of the line number.
5. In the shortcut menu, select the "Run" command.

Result

The CPU changes to RUN and program execution continues.

See also

Introduction to testing with program status (Page 2967)

Deleting breakpoints

You can delete breakpoints either individually or all together.

Deleting individual breakpoints

To delete an individual breakpoint from the program, follow these steps:

1. Open the "Testing" task card.
   In the "Breakpoints" pane, you will see a list with all the breakpoints set for the current CPU.
2. Select the breakpoint you want to delete.
3. Right-click on the selected breakpoint.
4. Select the "Delete breakpoint" command in the shortcut menu.

Or:

1. Open the block in which you want to delete a breakpoint.
2. Click in the line containing the breakpoint you want to delete and double-click in the gray column in front of the line number.

Deleting all breakpoints

To delete all breakpoints, follow these steps:

1. Open the "Testing" task card.
   In the "Breakpoints" pane, you will see a list with all the breakpoints set for the current CPU.
2. Make sure that no breakpoint is selected.
3. Click "Remove breakpoints" in the toolbar of the "Breakpoints" pane.

See also

Basics of testing in single step mode (Page 2969)

Setting breakpoints (Page 2970)

Setting the call environment for breakpoints (Page 2972)

Enabling and disabling breakpoints (Page 2972)

Running a test in single step mode (Page 2974)

9.4.2.4 Switching test with program status on/off

You can monitor all blocks by switching on the program status of the block. This function is available to you for all code blocks, regardless of the programming language used. For blocks that were programmed with LAD or FBD, you can also enable the program status from a specific position or for a specific selection. You can switch on the program status for an open block directly, or open a block from the calling block and view the program status.

Note

The resources for testing with program status are limited. If there are not enough resources for the current test, earlier tests will be terminated.

Requirement

The identical block exists in the device.

Switching the program status on or off directly in the block

To switch the program status for a block on or off directly in the block, follow these steps:

1. Open the block for which you wish to switch on the program status.
2. Click the "Monitoring on/off" button in the toolbar.
   If you have not already established an online connection, the "Go online" dialog opens. In this dialog, you can establish an online connection.
   See also: Establishing and terminating an online connection
3. Click the "Monitoring on/off" button in the toolbar again to switch the program status off.

Switching on or off program status starting at a specific point in a network

To start the program status for LAD and FBD at a specific point, follow these steps:

1. Open the block for which you wish to switch on the program status.
2. Click the "Monitoring on/off" button in the toolbar.
3. Right-click on the tag you want program status to start from.
4. Select "Modify > "Monitor from here" in the shortcut menu.
5. Click the "Monitoring on/off" button in the toolbar again to switch the program status off.

Switching on or off program status for selected tags

To start the program status for LAD and FBD for selected tags, follow these steps:

1. Open the block for which you wish to switch on the program status.
2. Click the "Monitoring on/off" button in the toolbar.
3. Select the tags for which you want to start the program status.
4. Select "Modify > Monitor selection" in the shortcut menu.
5. Click the "Monitoring on/off" button in the toolbar again to switch the program status off.

Switching on program status from the calling block

To switch on the program status for a block from the calling block (e.g. OB1), follow these steps:

1. Open the calling block.
2. Right-click on the block call.
3. Select the command "Open and monitor" in the shortcut menu.
   The block will open in the program editor. An online connection is established and the program status is displayed.

Note

If the calling block is programmed in SCL and the called block is an FB, the FB is opened with "Open and monitor". To monitor the FB, you then need to switch on the program status directly in the FB.

Result

If you enable the display of the program status, an online connection is established and the program status is displayed. When you turn off the display of the program status, you can terminate the online connection at the same time.

The call path of the block is shown under the block interface. For S7-300/400 and S7-1500 CPUs, you can change the call condition if required using the button on the right edge.

See also

Setting the call environment (Page 2968)

9.4.2.5 Editing blocks during the program test

If you edit blocks while the test with program status is still running, online monitoring will be interrupted and you will be able to edit the block offline. If the block is not available offline in the project, you will first have to load it from the device to the project. After editing the block, you will also have to compile and download it again.

Procedure

To edit blocks while the test with program status is still running, follow these steps:

1. Edit the block as necessary.
   The test with program status is interrupted and the block is switched offline assuming it exists offline.
2. If the block does not exist offline, load it to the project from the device.
3. Compile the block.
   See also: Compiling blocks (Page 2898)
4. Download the block to the device.
   See also: Downloading blocks (Page 2903)

Result

The block now contains your modifications both online and offline. The online connection is re-established and testing with program status continues.

9.4.2.6 Modifying tags in the program status

While testing with the program status, you have the option of modifying tags to the following values once and immediately:

- Modify to 1
  Modifies tags of the "Bool" data type to the value "True".
- Modify to 0
  Modifies tags of the "Bool" data type to the value "False"
- Modify operand
  You can enter a modify value for tags that do not belong to the "Bool" data type.

Note that you cannot modify peripheral inputs, for example, via TagName:P.

Procedure

To modify tags during testing with the program status, proceed as follows:

1. Right-click on the tag you want to modify.
2. Select one of the following commands in the shortcut menu:
   – "Modify > Modify to 1"
   – "Modify > Modify to 0"
   – "Modify > Modify operand"
3. If you select "Modify operand", the "Modify operand" dialog opens. Enter the value you require in the "Modify value" box and confirm with "OK".

9.4.2.7 Switching display formats in the program status

Introduction

The display formats for tags are generally displayed in "integer" form. In the program status, you have the option of switching the current display format by means of the shortcut menu. The possible display formats for a tag are offered in a list. This is useful, for example, when you need a hexadecimal display in order to search for a hexadecimal error code.

Procedure

To switch the display format, follow these steps:

1. Open the desired block in the programming editor.
2. Switch on the program status by clicking "Monitoring on/off" in the toolbar.
   If you have not already established an online connection, the "Go online" dialog opens. In this dialog, you can establish an online connection.
3. Select the tags for which you want to start the program status.
4. Select "Modify > Monitor selection" in the shortcut menu to start monitoring this tag.
5. Select the desired tag at the corresponding block output and then select the desired display format in the shortcut menu, for example, "Modify > Display format > Hexadecimal".

Result

The display format for the selected tag is shown in hexadecimal form.

Note

Switching the display format in the program status

Please note that it is not possible to switch the display format for unconnected outputs, as no monitoring value is output in this case.

9.4.2.8 Special considerations when testing GRAPH programs

Basics on testing GRAPH programs

Introduction

You can display the program status of a GRAPH program to check the logic, consistency and function of a sequential control system.

In the process you can, for example, determine the following errors:

- Program errors, for example, shifts between the programmed steps and conditions and the actual process sequence
- Programming errors in the sequential control system, such as incorrectly defined monitoring times in supervisions

To be able to test a user program, an online connection to the CPU must be available.

⚠ WARNING
Danger when testing during ongoing plant operation Testing while the plant is operating can cause serious damage to property or injury to persons if there are functional disturbances or program errors. Make sure that no dangerous situations can arise before you conduct a test.

Test functions of GRAPH

The following test functions are available for testing sequential control systems that were created in GRAPH:

- Program status for sequences
- Program status for conditions and actions
- Program status for interlocks and supervision
- Program status for permanent pre-instructions and post-instructions

During a test, you have the option to control the sequencer or to synchronize it with the current process status.

Operating modes for testing GRAPH programs

The following three operating modes are available for the testing GRAPH programs:

- Automatic mode
  In this operating mode, the sequence automatically advances to the next step as soon as the transition is fulfilled.
- Semi-automatic mode
  In this operating mode, the sequence advances to the next step if one of the following conditions is fulfilled:
  - Transition is satisfied.
  - There is a rising signal edge at the "T_PUSH" parameter.
  - You continue advance manually.
- Manual mode
  In this operating mode, you can either advance from one step to the next manually or specifically select a step.

System synchronization

If a process is brought to another state in manual mode, it may no longer be synchronized with the sequencer. To synchronize the process with the program once again, you can have the program look for synchronization points and subsequently perform the synchronization.

You can select from the following methods to find the synchronization points:

- Preceding transition satisfied
  All steps whose preceding transition has been fulfilled are selected.
- Interlock condition satisfied
  All steps whose interlock condition has been fulfilled are selected.

Only those steps for which the subsequent transition is not fulfilled are selected for both options.

See also

Basics of testing the user program (Page 2966)

Introduction to testing with program status (Page 2967)

Overview of the test settings for GRAPH (Page 2982)

Controlling a sequencer (Page 2984)

Acknowledging supervision errors (Page 2984)

Testing in various operating modes (Page 2985)

Performing a system synchronization (Page 2986)

Specifying the test settings

Overview of the test settings for GRAPH

Overview

The following table provides an overview of the settings for testing GRAPH programs:

Setting	Description
Track active step	The step of the sequencer which is active in each case is displayed in the single-step view or in the sequence view.
Skip steps	A step whose preceding or subsequent transition is valid will not be activated. The default setting of this option is adopted from the corresponding block settings. This option is not available when you use the "Create minimized DB" option for the GRAPH function block.
Acknowledge supervision alarms	Specifies that supervision errors must be acknowledged before the program continues. This option is only available if the "Acknowledge supervision alarms" option is selected in the settings of the GRAPH function block.
Stop sequence	The sequencer is stopped, even if the next transition is fulfilled.
Stop timers	If this option is enabled, all step activation timers are stopped. If this option is disabled, all timers continue to run.

Setting	Description
Process all interlocks	All interlocks, even those in step that are not active, are processed. This option is only available when both of the following conditions are fulfilled: • The sequence property "Permanent processing of all interlocks in manual mode" is enabled in the settings of the GRAPH function block. • The sequencer is in "Manual mode". For this the sequencer must be synchronized with the device. You can recognize the interlocks that are fulfilled in the sequence view.
Process all transitions	All transitions are always processed. You can recognize the transitions that are fulfilled in the sequence view.
Activate actions	Activates the actions of the active step. This option is enabled by default. When you disable this option, no additional actions are executed in the active step.
Activate supervisions	Activates the supervision of the active step. This option is enabled by default. When you disable this option, the conditions of supervision are ignored in the active step.
Activate interlocks	Activates the interlocks of the active step. This option is enabled by default. When you disable this option, the conditions of the interlock are ignored in the active step.

See also

Basics on testing GRAPH programs (Page 2980)

Changing test settings (Page 2983)

Changing test settings

WARNING

Changing the test settings

Keep in mind that changes to the test settings can have unintended effects on the process. Make sure that no dangerous states can occur before you change the test settings.

Requirement

A GRAPH function block is open.

Procedure

To change the test settings of GRAPH, follow these steps:

1. Start the test with program status for the GRAPH function block.
   See also: Switching test with program status on/off (Page 2977)
2. Open the "Testing" task card.
3. Change the settings in the "Test settings" pallet.

See also

Overview of the test settings for GRAPH (Page 2982)

Controlling a sequencer during testing

Controlling a sequencer

You can re-initialize the sequencer during testing with the program status. The sequencer then starts with the initial steps once again. You also have the option to disable all steps for testing.

Requirement

The program status is displayed.

Initialize sequencer

To initialize the sequencer, follow these steps:

1. Open the "Testing" task card.
2. Click the "Initialize" in the "Sequence control" pallet.
   The sequencer is executed starting with the initial steps.

Disable all steps

To disable all steps for testing of a sequencer, follow these steps:

1. Open the "Testing" task card.
2. Click "Disable all" in the "Sequence control" pallet.
   All steps of the sequencer are disabled.

See also

Acknowledging supervision errors (Page 2984)

Testing in various operating modes (Page 2985)

Performing a system synchronization (Page 2986)

Basics on testing GRAPH programs (Page 2980)

Introduction to testing with program status (Page 2967)

Switching test with program status on/off (Page 2977)

Acknowledging supervision errors

You can specify in the test settings that supervision errors have to acknowledged during testing before the program continues.

Requirement

The program status is displayed.

Procedure

To acknowledge supervision errors, follow these steps:

1. Open the "Testing" task card.
2. Click "Acknowledge - (V) -" in the "Sequence control" pallet.

See also

Controlling a sequencer (Page 2984)
Testing in various operating modes (Page 2985)
Performing a system synchronization (Page 2986)
Basics on testing GRAPH programs (Page 2980)
Overview of the test settings for GRAPH (Page 2982)
Changing test settings (Page 2983)
Introduction to testing with program status (Page 2967)
Switching test with program status on/off (Page 2977)

Testing in various operating modes

Requirement

The program status is displayed.

Testing in automatic mode

Proceed as follows to test a sequencer in automatic mode:

1. Open the "Testing" task card.
2. Select "Automatic" as operating mode in the "Sequence control" pallet.

Testing in semiautomatic mode

Proceed as follows to test a sequencer in semiautomatic mode:

1. Open the "Testing" task card.
2. Select "Semiautomatic mode" as operating mode in the "Sequence control" pallet.
3. When switching does not continue automatically by a fulfilled transition or via the parameter "T_PUSH", click the "Ignore transition" button.

Testing in manual mode

To test a sequencer in manual mode, follow these steps:

1. Open the "Testing" task card.
2. Select "Manual mode" as operating mode in the "Sequence control" pallet.
3. Advance to another step:
   - If you want to enable the next step, click the "Next" button.
   - If you want to enable or disable any given step, enter the step number in the "Select step manually" field or select a step in the sequencer and click the "Enable" or "Disable" button.

See also

Controlling a sequencer (Page 2984)

Acknowledging supervision errors (Page 2984)

Performing a system synchronization (Page 2986)

Switching test with program status on/off (Page 2977)

Basics on testing GRAPH programs (Page 2980)

Introduction to testing with program status (Page 2967)

Performing a system synchronization

Requirement

- The program status is displayed.
- The test takes place in "Manual mode" operating mode.

Procedure

Proceed as follows to perform a system synchronization:

1. Open the "Testing" task card.
2. Select the "Enable synchronization" check box in the "Sequence control" pallet.
3. Select one of these two methods for locating synchronization points:
   - "Preceding transition satisfied"
     All steps for which the preceding transition has been fulfilled are selected.
   - "Interlock condition satisfied"
     All steps whose interlock condition has been fulfilled are selected.
4. Select the step with which the system synchronization should start.
5. Click the "Activate" button.

Result

The process and the sequencers are synchronized with each other.

See also

Basics on testing GRAPH programs (Page 2980)
Controlling a sequencer (Page 2984)
Acknowledging supervision errors (Page 2984)
Testing in various operating modes (Page 2985)
Introduction to testing with program status (Page 2967)
Switching test with program status on/off (Page 2977)

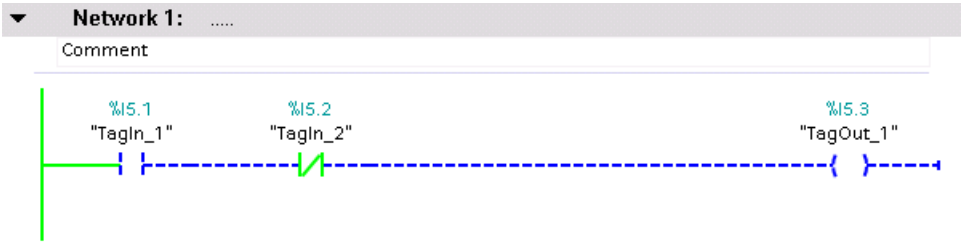
9.4.2.9 Examples of program status display

Program status display for LAD programs

Displays in program status

The display of the program status is updated cyclically.

The following figure shows an example of the program status display for LAD:



Representation of the program status

You can recognize the status of individual instructions and lines of a network quickly based on the color and type of lines and symbols. The following table shows the relationship between representation and status:

Representation	Status
Green solid	Satisfied
Blue dashed	Not satisfied
Gray solid	Unknown or not executed
Black	Not interconnected

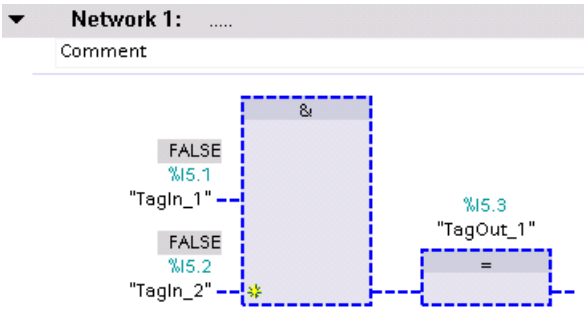
Representation	Status
Parameter in a frame with a saturation of 100 %	Value is current
Parameter in a frame with a saturation of 50 %	Value originates from an earlier cycle. The point in the program was not executed in the current cycle.

Program status display for FBD programs

Displays in program status

The display of the program status is updated cyclically.

The following figure shows an example of the program status display for FBD:



Representation of the program status

You can recognize the status of individual instructions and lines of a network quickly based on the color and type of lines and symbols. The following table shows the relationship between representation and status:

Representation	Status
Green solid	Satisfied
Blue dashed	Not satisfied
Gray solid	Unknown or not executed
Black	Not interconnected
Parameter in a frame with a saturation of 100 %	Value is current
Parameter in a frame with a saturation of 50 %	Value originates from an earlier cycle. The point in the program was not executed in the current cycle.

The values of the operands are displayed above the relevant operand name in a gray box.

Note

Program status display for outputs which are not interconnected

Please note that a monitor value cannot be displayed for outputs which are not interconnected.

Program status display for STL programs

Displays in program status

The display of the program status is updated cyclically and shown in tables. The tables are shown directly next to the STL program. You can read the program status for each line of the program. The display depends on the CPU in use (S7-300, S7-400, S7-1200 or S7-1500).

The tables to be displayed in the program status always contain the following information:

- RLO
  The "RLO" column shows the result of logic operation for each line of program. You can recognize the value of the RLO based on the background color of the table cell. Here, green means an RLO of 1 and lilac an RLO of 0.
- Value
  The current value of the operand is shown in the "Value" column.
- Extra
  The "Extra" column shows additional information depending on the particular operation, for example, relevant status bits for mathematical instructions, time or count values for timers and counters or memory addresses for indirect addressing.

The following figure shows an example of the program status display of a CPU S7-300 under STL:

Network 1: .....

Comment

			RLO	Value	Extra
1	OPN "DB1"	%DB1	0	DB1	
2	L %DBW2	%DBW2	0	16#4D2	DB1
3	L %DBW4	%DBW4	0	16#3333	DB1
4	+I		0	14341	OS=0,OV=0,A0=0,A1=1
5	T "Tag_15"	%MW4	0	14341	
6					
7	L 3.0	3.0	0	3.0	
8	T "Tag_24"	%MD10	0	3.0	
9	L 5.0	5.0	0	5.0	
10	T "Tag_52"	%MD20	0	5.0	
11	L MD ["Tag_29"]	%MD26	0	16#40400000	P#10.0
12	LAR1 P#M16.0	P#M16.0	0	P#M16.0	
13	L MD [AR1 , P#4.0]	P#4.0	0	16#40A00000	P#M20.0
14	+R		0	8.0	OS=0,OV=0,A0=0,A1=1
15	T "Tag_17"	%MD30	0	8.0	
16					
17	OPN DB ["Tag_48"]	%MW14	0	DB1	
18	L DBW ["Tag_38"]	%MD60	0	16#3333	P#4.0
19	L W#16#7777	W#16#7777	0	16#7777	
20	XOW		0	16#4444	
21	T "Tag_47"	%MW70	0	16#4444	
22					
23	A "Tag_1"	%I0.0	1	1	
24	L S5T#55S	S5T#55S	1	S5T#55S	
25	SE "Tag_49"	%T5	1	S5T#47S600MS	
26	A "Tag_49"	%T5	1	1	S5T#47S600MS
27	= "Tag_3"	%Q0.0	1	1	
28					
29	A "Tag_55"	%I0.2	1	1	
30	CU "Tag_23"	%C1	1	C#556	
31	A "Tag_23"	%C1	1	1	C#556
32	= "Tag_3"	%Q0.0	1	1	
33					
34	A "Tag_31"	%M10.0	0	0	
35	AN M ["Tag_51"]	%MD64	0	0	P#5.1
36	O DBX ["Tag_38"]	%MD60	1	1	P#4.0
37	LAR2 P#M10.0	P#M10.0	1	P#M10.0	
38	X [AR2 , P#3.4]	P#3.4	1	0	P#M13.4
39	= Q ["Tag_51"]	%MD64	1	1	P#5.1
40					

Program status display for SCL programs

Displays in program status

The display of the program status is updated cyclically and shown in a table. The table is displayed immediately beside the SCL program and you can see the program status for each line of the program. The table contains the following information:

- Tag names
- Value

You can move the table to the left or right at any time.

The following figure shows an example of the program status display for SCL:

```
1 IF "TagIn_1"                    "TagIn_1"    FALSE
2 THEN "TagIn_2" :=1;             "TagIn_2"    TRUE
3 END_IF;
4 IF "TagIn_2" = false            "TagIn_2"    FALSE
5 THEN "TagIn_3" :=1;             "TagIn_3"    TRUE
6 END_IF;
7
```

In the first column, you can see the name of the tag for which the current value is being displayed. If the line includes the "IF", "WHILE" or "REPEAT" instruction, the result of the instruction is displayed in the line as "True" or "False". If the line contains more than one tag, the value of the first tag is displayed. In both cases, all tags of these lines are displayed with their values in a separate list as soon as you select a line. If you place the cursor in a tag in the program code, this is shown in bold face in the list. You can also display the other tags of a line explicitly by clicking the arrow right located in front of lines containing more than one tag.

If the code of the line is not executed, the tag name is displayed in the values table in gray text.

The current values of the tags are displayed in the last column. If no values can be displayed for a tag, the line has a yellow background and three question marks are shown. In this case, select the "Create extended status information" check box in the properties of the block and download the block to the device again. All values are then displayed.

Program status display for GRAPH programs

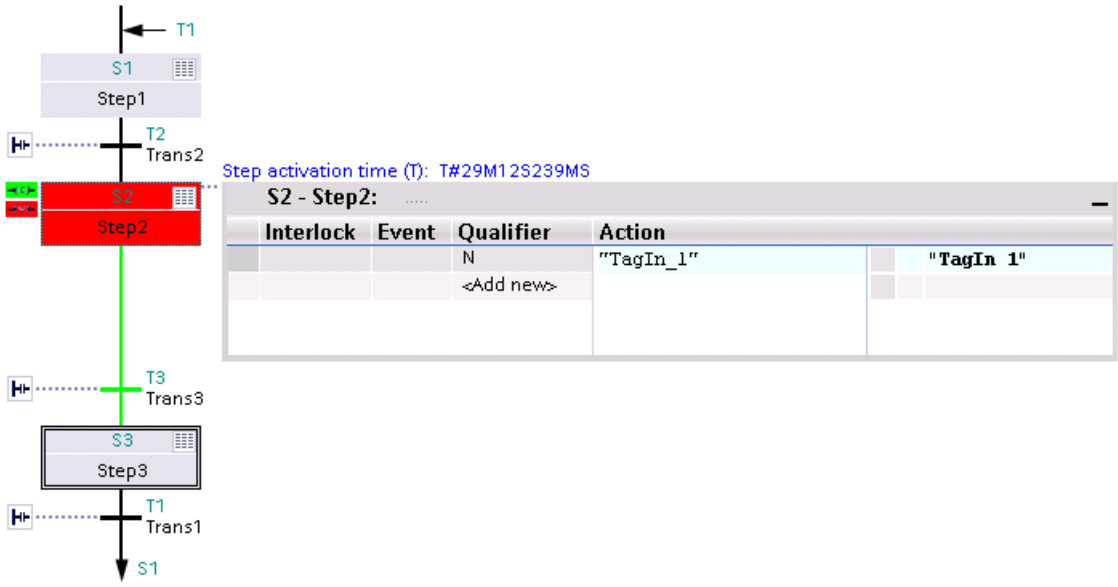
Displays in program status

For GRAPH programs you can have the program status displayed in sequence view and in single step view and for the permanent instructions. The program status display of permanent instructions corresponds to the program status display for LAB/FBD programs. The display of the program status is updated cyclically.

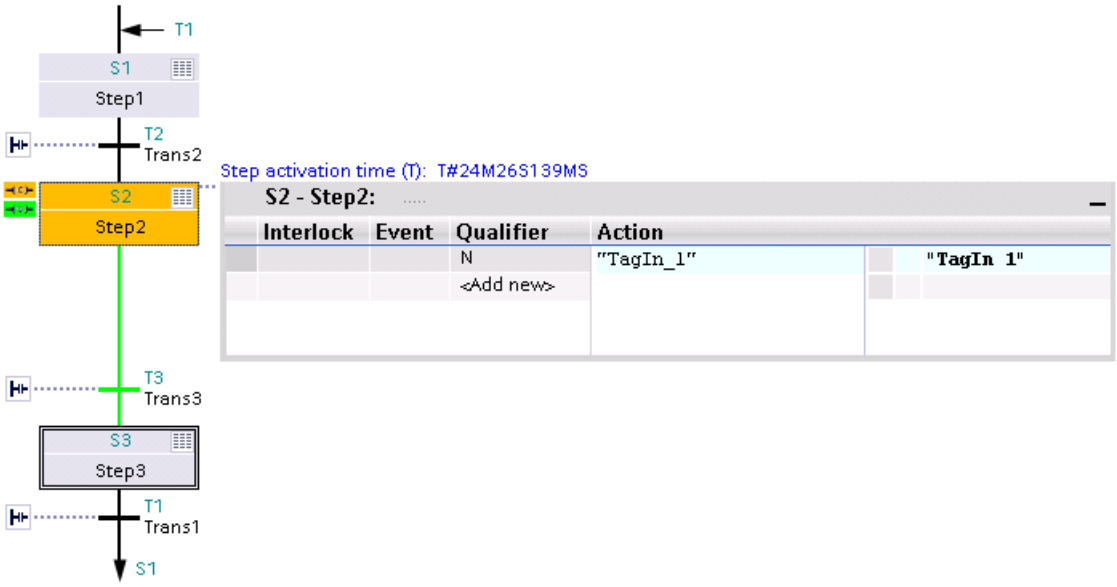
The following table shows the relationship between representation and status:

Representation	Area	Status
Green	Step, sequencer	There are no faults.
	Condition	The transition is fulfilled.
Red	Step, sequencer	There is a supervision error.
Yellow	Step, sequencer	There is an interlock error.
Black	Condition	The transition is not fulfilled.

The following figures show examples of the display for the program status in the sequence view:

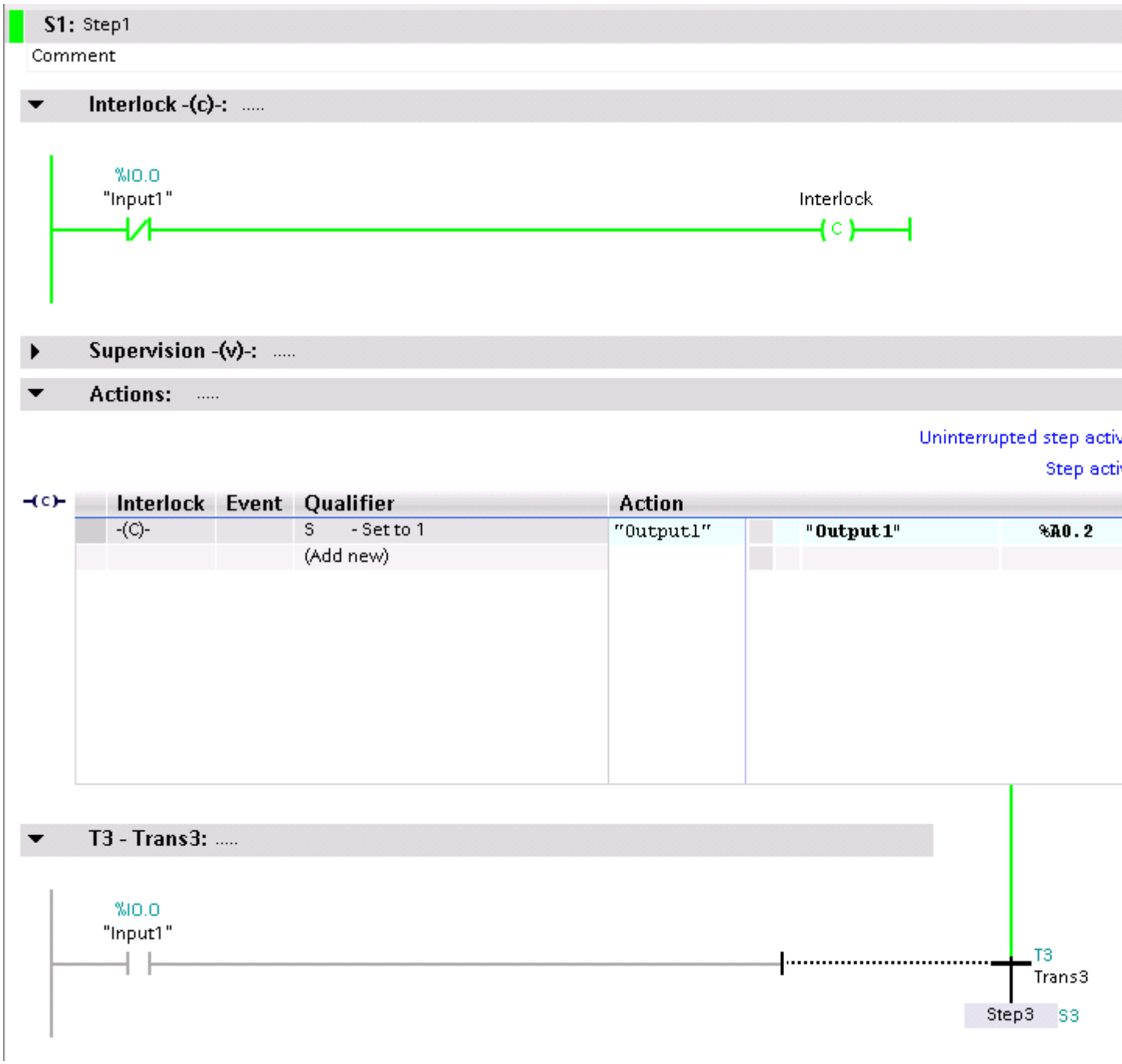


Step 2 contains a supervision error. The transition for switching to step 3 is fulfilled.



Step 2 contains an interlock error. The transition for switching to step 3 is fulfilled.

The figure below shows an example of the display for the program status in the single step view of a S7-300:



The step does not contain an interlock error. The subsequent transition is not satisfied.

9.4.3 Testing with the watch table

9.4.3.1 Introduction to testing with the watch table

Overview

The following functions are available in the watch table:

- **Monitoring tags**
  This displays the current values of the individual tags of a user program or a CPU on the programming device or PC.
- **Modifying tags**
  You can use this function to assign specific values to the individual tags of a user program or CPU. Modifying is also possible with Test with program status (Page 2967).
- **"Enable peripheral outputs" and "Modify now"**
  These two functions enable you to assign specific values to individual peripheral outputs of a CPU in STOP mode. You can also use them to check your wiring.

Monitoring and modifying tags

The following tags can be monitored and modified:

- Inputs, outputs, and bit memory
- Contents of data blocks
- I/O

Possible applications

The advantage of the watch table is that a variety of test environments can be stored. This enables you to reproduce tests during commissioning or for service and maintenance purposes.

See also

Creating and editing watch tables (Page 2998)

Layout of the watch table (Page 2995)

Basic mode and expanded mode in the watch table (Page 2996)

Icons in the watch table (Page 2997)

9.4.3.2 Layout of the watch table

Introduction

A watch table contains the tags you defined for the entire CPU. A "Watch and force tables" folder is automatically generated for each CPU created in the project. You create a new watch table in this folder by selecting the "Add new Watch table" command.

Layout of the watch table

The columns displayed in the watch table depend on the mode you are working in: basic mode or expanded mode.

The following additional columns are shown in expanded mode:

- Monitor with trigger
- Modify with trigger

The names of the columns can also be changed dynamically based on the action.

Meaning of the columns

The following table shows the meaning of the individual columns in basic mode and expanded mode:

Mode	Column	Meaning
Basic mode	i	Identifier column
	Name	Name of the inserted tag
	Address	Address of the inserted tag
	Display format	Selected display format
	Monitor value	Values of the tags, depending on the selected display format.
	Modify value	Value with which the tag is modified.
		Select the tag to be modified by clicking the corresponding check box.
	Comment	Comment for documentation of the tags
The following additional columns are shown in expanded mode:	Monitor with trigger	Display of selected monitoring mode
	Modify with trigger	Display of selected modify mode

See also

Icons in the watch table (Page 2997)

9.4.3.3 Basic mode and expanded mode in the watch table

Difference between basic mode and expanded mode in the watch table

Depending on the mode specified, the watch table displays different columns and column headings that can be used to perform different actions.

You will find a detailed list of the columns in Layout of the watch table (Page 2995).

Switching between basic mode and expanded mode

You have the following options of toggling between the basic and expanded mode:

- Click the icon "Show/hide advanced setting columns". Click this icon again to return to the basic mode.
  Or:
- Activate the check box for the "Expanded Mode" command in the "Online" menu. Deactivate this check box to return to the basic mode.

Functionality in expanded mode

The following functionality is only possible in expanded mode:

- Monitor with trigger
- Modify with trigger
- Enable peripheral outputs
- Monitoring peripheral inputs
- Controlling peripheral outputs

NOTICE
Danger of a time-out while monitoring peripheral inputs and controlling peripheral outputs Note that the monitoring of peripheral inputs and the controlling of peripheral outputs in the watch table can result in a time-out. The CPU assumes the "STOP" mode.

See also

Setting the monitoring and modify mode (Page 3009)

9.4.3.4 Icons in the watch table

Meaning of the icons

The following table shows the meaning of the icons in the watch table:

Icon	Meaning
	Identifies a table inside the project tree as a watch table.
	Shows information in the identifier column.
	Inserts a row before the selected row.
	Inserts a row after the selected row.
	Modifies the addresses of all selected tags immediately and once. This command is executed once and as quickly as possible without reference to a defined trigger point in the user program.
	Modifies the addresses of all selected tags with reference to a defined trigger point in the user program.
	Disables the command output disable of the peripheral outputs. You can then modify the peripheral outputs when the CPU is in STOP mode.
	Displays all columns of expanded mode. If you click this icon again, the columns of expanded mode will be hidden.
	Displays all modify columns. If you click this icon again, the modify columns will be hidden.
	Starts monitoring of the visible tags in the active watch table. The default setting for the monitoring mode in basic mode is "permanent". In expanded mode, you can set defined trigger points for the monitoring of tags.
	Starts monitoring of the visible tags in the active watch table. This command is executed immediately and the tags are monitored once.
	Displays the check box for the selection of tags to be modified.
	Indicates that the value of the selected tag has been modified to "1".
	Indicates that the value of the selected tag has been modified to "0".
	Indicates that the address is being used multiple times.
	Indicates that the substitute value is being used. Substitute values are values that are output to the process in case of signal output module faults or are used instead of a process value in the user program in case of signal input module faults. The substitute values can be assigned by the user (e.g., retain old value).
	Indicates that the address is blocked because it is already being modified.
	Indicates that the address cannot be modified.
	Indicates that the address cannot be monitored.
	Indicates that an address is being forced.
	Indicates that an address is being partly forced.

Icon	Meaning
	Indicates that an associated I/O address is being fully or partly forced.
	Indicates that an address cannot be fully forced. Example: It is indeed possible to force the address QW0:P, but it is not possible to force the address QD0:P since this address area is eventually not available on the CPU.
	Indicates that a syntax error occurred.
	Indicates that the address is selected but at the moment e.g. has not yet been modified.

See also

Layout of the watch table (Page 2995)

9.4.3.5 Creating and editing watch tables

Creating a watch table

Introduction

The watch table allows you to monitor and modify tags in the user program. Once you have created a watch table, you can save it, duplicate it, and print it and use it again and again to monitor and modify tags.

Requirement

A project is open.

Procedure

To create a watch table, follow these steps:

1. Click "Project view" in the status bar.
   The project view is displayed.
2. In the project tree, double-click the CPU for which you want to create a watch table.
3. Double-click the "Watch and force tables" folder and then the "Add new watch table" command.
   A new watch table is added.
4. In the "Name" column or in the "Address" column, enter the name or the absolute address for the tags that you want to monitor or modify.
5. You can select a display format from the drop-down list in the "Display format" column if you want to change this default setting.
6. Now decide whether you want to monitor or modify the entered tags and, if applicable, enter the desired values for modifying.

Opening a watch table

Requirement

A watch table has been created.

Procedure

To open a watch table, follow these steps:

1. Open the "Watch and force tables" folder below the desired CPU.
2. Double-click on the required watch table in the folder.

Result

The selected watch table opens.

Copying and pasting a watch table

Requirement

A watch table has been created.

Procedure

To copy a watch table, follow these steps:

1. Right-click the watch table that you want to copy.
2. In the context menu, select "Copy".
3. In the project tree, open the folder structure for the CPU in which you want to paste the copied watch table.
4. Right-click on the "Watch and force tables" folder.
5. In the context menu, select "Paste".
6. Alternatively, you can select the entire contents of the watch table and Drag & Drop it onto another watch table.

Result

A copy of the selected watch table is placed in the "Watch and force tables" folder of the relevant CPU.

Saving a watch table

Prerequisite

A watch table has been created.

Procedure

To save a watch table, follow these steps:

1. In the project tree select the watch table you want to save.
2. If you wish to change the preset name of the table, select the "Rename" command in the context menu and enter a new name for the table.
3. In the "Project" menu, select "Save". Note that this save operation will save the entire project.

Result

The contents of the watch table and the project are saved.

Note

You can reuse saved watch tables to monitor and modify tags when retesting your program.

9.4.3.6 Entering tags in the watch table

Basic information on entering tags in the watch table

Recommended procedure

- Select the tags whose values you want to monitor or modify, and enter them in the watch table.
- When entering tags into the watch table, please note that these tags must be previously defined in the PLC tag table.
- When entering tags, work from the outside to the inside. This means that you start by entering the tags for the inputs in the watch table. Then, you enter the tags that are affected by the inputs or that affect the outputs. Finally, you enter the tags for the outputs.

Example of filling out a watch table

- Enter the absolute address to be monitored or modified in the "Address" column.
- Enter the symbolic name for the tag in the "Name" column.

- Select the display format you require from the drop-down list in the "Display format" column, if you do not want to use the default setting.
- Now decide whether you want to monitor or modify the entered tags. Enter the desired values for modifying as well as a comment in the corresponding columns of the watch table.

Create comment row

If required, you can create a comment row by entering the string "//" in the "Name" column.

Syntax check

When you enter the tags in the watch table, the syntax of each cell is checked when you exit the cell. Incorrect entries are marked in red.

Note

When you place the mouse pointer in a cell marked in red, brief information is displayed with additional notes on the error.

See also

Permitted operands for the watch table (Page 3001)

Permissible modify values for the watch table (Page 3002)

Permitted operands for the watch table

Permissible operands for the watch table

The following table shows the operands that are permitted for the watch table:

Permitted operand	Example of data type	Example (International mnemonics)
Input/output/bit memory	BOOL	I1.0, Q1.7, M10.1 I0.0:P; Q0.0:P
Input/output/bit memory	BYTE	IB1/QB10/MB100 IB1:P; QB1:P
Input/output/bit memory	WORD	IW1; QW10; MW100 IW2:P; QW3:P
Input/output/bit memory	DWORD	ID4; QD10; MD100 ID2:P; QD1:P
Timers	TIMER	T1
Counters	COUNTER	C1
Data block	BOOL	DB1.DBX1.0
Data block	BYTE	DB1.DBB1

Permitted operand	Example of data type	Example (International mnemonics)
Data block	WORD	DB1.DBW1
Data block	DWORD	DB1.DBD1

Note

Please observe the following notes to work with the watch table.

- You cannot enter "DB0..." because it is used by the system!
- Peripheral outputs can be modified but not monitored.
- Peripheral inputs can be monitored but not modified.

NOTICE

Danger of a time-out while monitoring peripheral inputs and controlling peripheral outputs

Note that the monitoring of peripheral inputs and the controlling of peripheral outputs in the watch table can result in a time-out.

The CPU assumes the "STOP" mode.

See also

Basic information on entering tags in the watch table (Page 3000)

Overview of the valid data types (Page 2236)

Permissible modify values for the watch table

Entry of modify values in the watch table

The following table shows the operands that are permitted for the entry of modify values in the watch table:

Table 9-14 Bit operands

Possible bit operands	Example for permitted modify values
I1.0	True
M1.7	False
Q1.0	0
Q1.1:P	1
DB1.DBX1.1	2#0
M1.6	2#1

Table 9-15 Byte operands

Possible byte operands	Example for permitted modify values
IB1	2#00110011
MB12	B#16#1F
QB10	1F
QB11:P	'a'
DB1.DBB1	10

Table 9-16 Word operands

Possible word operands	Example for permitted modify values
IW1	2#0011001100110011
MW12	W#16#ABCD
MW14	ABCD
QW10	B#(12, 34)
QW12:P	12345
DB1.DBW1	'ab'
MW16	S5T#9s_340ms
MW18	C#123
MW9	D#2006-12-31

Table 9-17 Double word operands

Possible double word operands	Example for permitted modify values
ID1	2#00110011001100110011001100110011
QD10	Dw#16#abcdef10
QD12:P	ABCDEF10
DB1.DBD2	b#(12,34,56,78)
MD8	L#-12
MD12	L#12
MD16	123456789
MD20	123456789
MD24	T#12s345ms
MD28	Tod#1:2:34.567
MD32	P#e0.0

Table 9-18 Timers

Possible operands of the "Timer" type	Permitted control values	Explanation
T1	0 ms	Time value in milliseconds (ms)
T12	20 ms	Time value in milliseconds (ms)
T14	12345 ms	Time value in milliseconds (ms)
T16	S5t#12s340ms	Time value 12s 340 ms

Table 9-19 Counters

Possible operands of the "Counter" type	Permitted control values
C1	0
C14	20
C16	C#123

Notes on timers and counters

- Timers

Note

Modifying a timer influences only the value, not the status. Timer T1 can be modified to the value "0", but the result of logic operation for A T1 is not changed.

The time sequences "s5t" and "s5time" can be written in both lower-case and upper-case characters.

- Counter

Note

Modifying a counter influences only the value, not the status. Counter C1 can be changed to the value "0", but the result of the logic operation for A C1 is not changed.

Overview of the display formats

Display formats in the watch table

The display format you select specifies the representation of a tag value.

When entering the address a display format is automatically preset. If you want to change this, you can select a display format from the drop-down list in the "Display formats" column. The drop-down list only offers the display formats which are valid for this data type. The display format that appears first in the list is the pre-selected format.

Example

The following table shows the 32-bit data types permitted for all CPU families in the watch table and their possible display formats:

Data type	Possible display formats
BOOL	Bool, Hex, BCD, Octal, Bin, Dec, Dec+/-
BYTE	Hex, BCD, Octal, Bin, Dec, Dec+/-, Character
WORD	Hex, BCD, Octal, Bin, Dec, Dec+/-, Dec_Sequence, Character, Unicode_Character, SIMATIC_Timer, Date, Counter
DWORD	Hex, BCD, Octal, Bin, Dec, Dec+/-, Dec_Sequence, Character, Unicode_Character, Floating-point number, Time of day, Timer, Pointer

Data type	Possible display formats
SINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character
INT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, SIMATIC_Timer, Counter, Date
DINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, Floating-point number, Time of day, Timer, Pointer
USINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character
UINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, SIMATIC_Timer, Counter, Date
UDINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, Floating-point number, Time of day, Timer, Pointer
REAL	Floating-point number, Hex, BCD, Octal, Bin, Character, Unicode_Character,Dec, Dec+/-, Dec_Sequence, Time of day, Timer, Pointer
DATE	Date, Dec, Hex, BCD, Bin
TIME_OF_DAY	Time of day, Dec, Hex, BCD, Bin
TIME	Timer, Hex, BCD, Bin
DATE_AND_TIME	Date and time,
TIMER	SIMATIC_Timer, Hex, BCD, Bin
CHAR	Character, Hex, BCD, Octal, Bin, Dec, Dec+/-
WCHAR	Unicode_Character, Character, Hex, BCD, Octal, Bin, Dec, Dec+/-
STRING	Character string
WSTRING	Unicode_character string
POINTER	Pointer, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec, Dec+/-, Dec_Sequence, Floating-point number, Time of day, Timer, Block number
COUNTER	Counter, Hex, BCD, Bin
S5TIME	SIMATIC_Timer, Hex, BCD, Bin

For the S7-1200 CPU family, all 32-bit data types are permitted (see table above), as well as the 64-bit data type LREAL with the following possible display formats:

Data type	Possible display formats
LREAL	In a project created with TIA Portal < V12: Floating-point number Note: The display of LREAL is limited to 13 digits plus exponent.
LREAL	In a project created with TIA Portal >= V12: Floating-point number, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec, Dec+/-, Dec_Sequence, Time of day, Timer, Date and time Note: The display of LREAL is limited to 13 digits plus exponent.

For the S7-1500 CPU family, in addition to 32-bit data types, the 64-bit data types listed in the table are also permitted with the following possible display formats:

Data type	Possible display formats
LWORD	Hex, Octal, BCD, Bin, Character, Unicode_Character, Dec, Dec+/-, Dec_Sequence, Floating-point number, Time of day, Timer, Date and time
LINT	Dec+/-, Dec, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, Time of day, Timer, Date and time
ULINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, Time of day, Timer, Date and time
LREAL	Floating-point number, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec, Dec+/-, Dec_Sequence, Time of day, Timer, Date and time
LTIME	Timer, Dec+/-, Dec, Hex
LTOD	Time of day, Dec, Hex, BCD, Bin
LDT	Date and time, Dec, Hex

For more information, refer to the description of the valid data types (Page 2236).

Note

Rounding of floating-point numbers

In the watch table, floating-point numbers are stored as binary numbers in IEEE format. Because not every floating point number (real, longreal) that can be displayed on the user interface can be mapped to the IEEE format, there is a possibility that floating-point numbers will be rounded. If a rounded floating-point number in the watch table is copied and, in turn, inserted in another input field, the rounding may cause a slight difference.

Note

Only symbolic addressing is possible

In the watch table, LongDataTypes such as LWORD or LREAL can only be addressed symbolically.

Selecting the display format for tags

Procedure

To select the display format of the tags, follow these steps:

1. Enter the desired address in the watch table.
2. Click the desired cell in the "Display format" column, and open the drop-down list. The permissible display formats are shown in the drop-down list.
3. Select the desired display format from the drop-down list.

Note

If the selected display format cannot be applied, then the last selected display format will be displayed automatically.

See also

Overview of the valid data types (Page 2236)

Creating and editing comment lines

Basic principles of comment lines

In addition to the row related comments in the comment column, you can now also create complete comment lines to enhance the structure of the watch table.

The contents of the comment line are stored in the "Languages & Resources" folder in the "Project texts" tab and can be compiled in other project languages.

Creating comment lines

To create comment lines, follow these steps:

1. Open the watch table and enter the required addresses.
2. To create a comment line, enter the character string "//" in the "Name" column. No comment lines can be created in the other columns.
3. Enter the required comment in the comment line. The entered comment is shown in green.
4. To show all comments you entered, double-click "Project texts" in the project tree under "Languages & Resources".
5. If you are working in multi-lingual projects and want your comment to be translated into other languages, you can set the project languages required in addition to the editing language in the project tree under "Languages & Resources > Project languages".

Deleting comment lines

To delete comment lines, follow these steps:

1. Open a watch table containing comment lines.
2. Delete the entire comment including the introductory string "//", if you no longer required this.
3. Alternatively, delete only the introductory string "//". In this case the existing comment is retained and is displayed in the "Comment" column in the watch table.

Note

Deleting comment lines

When you delete comment lines the project languages and any existing translations for these comments are also deleted.

9.4.3.7 Monitoring tags in the watch table

Introduction to monitoring tags in the watch table

Introduction

The watch table allows you to monitor the tags of the configured input and output modules in the CPU, depending on the monitoring and modify mode (Page 3009) selected. To monitor tags, an online connection to the CPU must exist.

NOTICE
Danger of a time-out while monitoring peripheral inputs Note that the monitoring of peripheral inputs can result in a time-out. The CPU assumes the "STOP" mode.

Options for monitoring tags

The following options are available for monitoring tags:

- Monitor now
  This command starts the monitoring of the visible tags in the active watch table immediately and once only.
- Monitor all
  This command starts the monitoring of all visible tags in the active watch table, depending on the selected watch mode:
  - In basic mode, the monitoring mode is set to "permanent" by default.
  - In expanded mode, you can specify defined trigger points for the monitoring of tags.

  Note

  If the monitoring mode is changed while in expanded mode and then a switch is made to basic mode, the monitoring mode set before will also be applied in basic mode.

CPU-specific limitations when monitoring tags

The following CPU-specific differences exist:

- CPU S7-300/400:
  CPUs from this family can only monitor the first 30 characters of a string.
- CPU S7-1200/1500:
  CPUs from this family can monitor a string up to the total size of 254 characters.

Setting the monitoring and modify mode

Introduction

By selecting the monitoring and modify mode, you specify the trigger point and the duration of the tag monitoring in the watch table and the force table.

Possible monitoring and modify modes (duration of monitoring or modifying)

The following monitoring and modifying modes are available:

- Permanent
- Once only, at start of scan cycle
- Once only, at end of scan cycle
- Permanently, at start of scan cycle
- Permanently, at end of scan cycle

- Once only, at transition to STOP
- Permanently, at transition to STOP

Special features when using "Permanent" mode

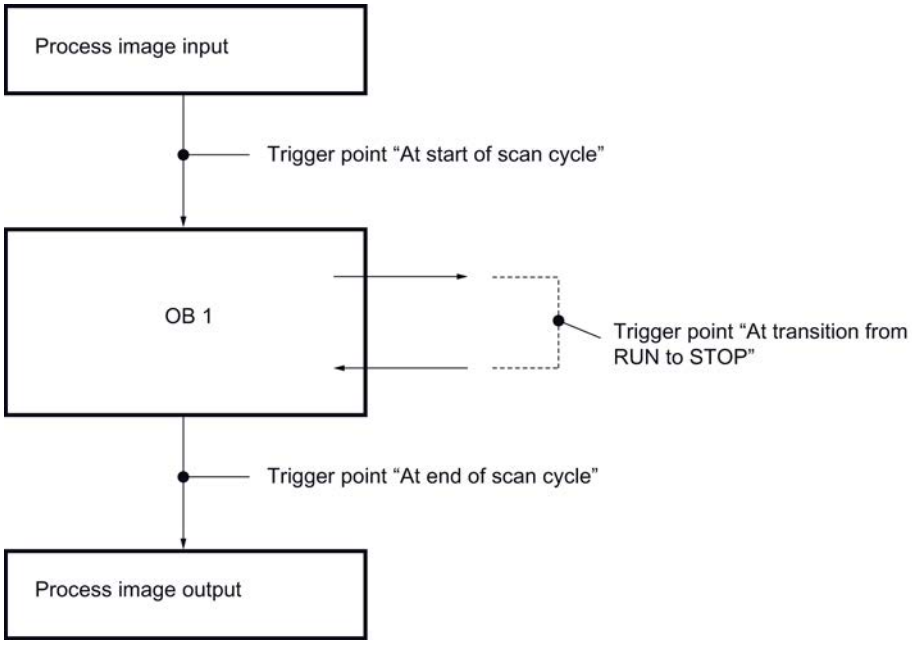
The "Permanent" mode is executed differently for the monitoring and modifying of tags.

- Monitoring: The inputs are monitored at the end of the cycle and the outputs at the start of the cycle.
- Modifying: The inputs are monitored at the start of the cycle and the outputs at the end of the cycle.

Selecting the trigger point

The trigger points "Beginning of scan cycle", "End of scan cycle", and "Switch to stop" specify the time at which the tags are to be read from the CPU or updated in the CPU.

The following diagram shows the position of these trigger points:



Position of the trigger points

From the position of the trigger points, it follows that:

- Modifying of inputs is only appropriate at the beginning of the scan cycle (corresponding to the beginning of the user program OB 1), because otherwise the process image input is updated again after modifying and is thus overwritten.
- Modifying of outputs is only appropriate at the end of the scan cycle (corresponding to the end of the user program OB 1), because otherwise the process image output can be overwritten by the user program.
- The modified value is displayed in the "Monitor value" column, provided that monitoring is active and the modified value is not overwritten by the user program.

Monitoring of tags

When tags are being modified, the following applies to the trigger points:

- If you have specified the modify mode as "once only", you will receive a message if the selected tags cannot be modified.
- In "permanent" modify mode, you do not receive a message.

Note regarding the "Modify now" command

You can modify the values of selected tags immediately using the "Online > Modify >Modify now" command. This command is executed once only and as quickly as possible without reference to a defined position (trigger point) in the user program. This function is used mainly for modifying when the CPU is in STOP mode.

"Monitor all" command for tags

Introduction

The "Monitor all" command allows you to start monitoring the visible tags in the active watch table. The default setting for the monitoring mode in basic mode of the watch table is "permanent". In expanded mode, you can specify defined trigger points for the monitoring of tags. In this case, the tags are monitored with reference to the specified trigger points.

NOTICE
Danger of a time-out while monitoring peripheral inputs Note that the monitoring of peripheral inputs can result in a time-out. The CPU assumes the "STOP" mode.

Requirements

- A watch table has been created.
- An online connection to the CPU exists.

Procedure

To execute the "Monitor all" command, follow these steps:

1. Enter the tags to be monitored and the corresponding addresses in the watch table.
2. Switch to expanded mode by clicking
   the icon "Show/hide advanced setting columns" in the toolbar.
3. If you want to change the default monitoring mode for a tag, click the appropriate cell in the "Monitor with trigger" column and select the desired monitoring mode from the drop-down list.
4. Click the "Monitor all" icon in the toolbar.

Result

The tags of the active watch table are monitored using the monitoring mode selected.

See also

Icons in the watch table (Page 2997)

Entering tags in the watch table (Page 3000)

Basic mode and expanded mode in the watch table (Page 2996)

"Monitor now" command for tags

Introduction

The "Monitor now" command starts the monitoring of tags immediately without reference to defined trigger points. The tag values are read out once only and displayed in the watch table.

NOTICE
Danger of a time-out while monitoring peripheral inputs Note that the monitoring of peripheral inputs can result in a time-out. The CPU assumes the "STOP" mode.

Requirements

- A watch table has been created.
- An online connection to the CPU exists.

Procedure

To execute the "Monitor now" command, follow these steps:

1. Enter the tags to be monitored and the corresponding addresses in the watch table.
2. Click the "Monitor now" icon in the toolbar.

Result

The tags of the active watch table are monitored immediately and once only.

See also

Icons in the watch table (Page 2997)
Entering tags in the watch table (Page 3000)
Basic mode and expanded mode in the watch table (Page 2996)

9.4.3.8 Modifying tags in the watch table

Introduction to modifying tags

Introduction

The watch table allows you to modify the tags of the configured input and output modules in the CPU, depending on the monitoring and modify mode (Page 3009) selected. To monitor the tags, an online connection to the CPU must exist.

DANGER

Danger when modifying:

Serious personal injury and material damage can result from changes in the tags or addresses during plant operation in the event of malfunctions or program errors!
Make sure that dangerous conditions cannot occur before you execute the "Modify" function.

NOTICE
Danger of a time-out while controlling peripheral outputs Note that the controlling of peripheral outputs in the watch table can result in a time-out. The CPU assumes the "STOP" mode.

Options for modifying tags

The following options are available for modifying tags:

- Modify to "0"
  This command modifies the selected address to the modify value "0".
- Modify to "1"
  This command modifies the selected address to the modify value "1".
- Modify once only and immediately
  This command modifies all selected addresses in the active watch table "once only and immediately".
- Modify with trigger
  This command modifies all selected addresses in the active watch table using the monitoring and modify mode (Page 3009) selected.
  The "Modify with trigger" function is only available in expanded mode. You will not receive a message indicating whether or not the selected addresses were actually modified with the specified value. You should use the "Modify once only and immediately" function if you require such a confirmation.
- Enable peripheral outputs
  This command disables the command output disable.
  This function can only be executed in expanded mode, when the CPU is in STOP and the option Force (Page 3038) of tags is not enabled. If desired, deactivate this function in the force table.

Note

When modifying, note the following:

Modifying can **not** be undone.

Modify tags to "0"

Introduction

You can assign one-time values to tags independent of the monitoring and modify mode and modify them. The modify command is executed as fast as possible, similar to a "Trigger now" command, without reference to a defined position in the user program.

DANGER

Danger when modifying:

Serious personal injury and material damage can result from changes in the tags or addresses during plant operation in the event of malfunctions or program errors!
Make sure that dangerous conditions cannot occur before you execute the "Modify" function.

NOTICE

Danger of a time-out while controlling peripheral outputs

Note that the controlling of peripheral outputs in the watch table can result in a time-out.

The CPU assumes the "STOP" mode.

Requirements

- A watch table has been created.
- An online connection to the CPU exists.

Procedure

To modify tags to "0", follow these steps:

1. Enter the desired address in the watch table.
2. Select the "Online > Modify > Modify to 0" command in order to modify the selected address with the specified value.

Result

The selected address is modified to "0".

Note

When modifying, note the following:

Modifying can **not** be undone!

Modify tags to "1"

Introduction

You can assign one-time values to tags independent of the monitoring and modify mode and modify them. The modify command is executed as fast as possible, similar to a "Trigger now" command, without reference to a defined position in the user program.

DANGER

Danger when modifying:

Serious personal injury and material damage can result from changes in the tags or addresses during plant operation in the event of malfunctions or program errors!
Make sure that dangerous conditions cannot occur before you execute the "Modify" function.

NOTICE

Danger of a time-out while controlling peripheral outputs

Note that the controlling of peripheral outputs in the watch table can result in a time-out.

The CPU assumes the "STOP" mode.

Requirements

- A watch table has been created.
- An online connection to the CPU exists.

Procedure

To modify tags to "1", follow these steps:

1. Enter the desired address in the watch table.
2. Select the "Online > Modify > Modify to 1" command in order to modify the selected address with the specified value.

Result

The selected address is modified to "1".

Note

When modifying, note the following:

Modifying can **not** be undone!

"Modify now" command for tags

Introduction

You can assign one-time values to tags independent of the monitoring and modify mode and modify them immediately. The modify command is executed as fast as possible, similar to a "Trigger now" command, without reference to a defined position in the user program.

DANGER

Danger when modifying:

Serious personal injury and material damage can result from changes in the tags or addresses during plant operation in the event of malfunctions or program errors!
Make sure that dangerous conditions cannot occur before you execute the "Modify" function.

NOTICE

Danger of a time-out while controlling peripheral outputs

Note that the controlling of peripheral outputs in the watch table can result in a time-out.

The CPU assumes the "STOP" mode.

Requirements

- A watch table has been created.
- An online connection to the CPU exists.

Procedure

To modify tags immediately, follow these steps:

1. Enter the desired addresses and modify values in the watch table.
2. Select the addresses to be modified by selecting the check boxes for modifying in the column after the "Modify value".
   A yellow triangle appears behind the selected check box, indicating that the address is now selected for modifying but has not yet been modified.
3. Select the "Online > Modify > Modify once and now" command in order to immediately modify the selected address once only with the specified value.

Result

The selected addresses are modified immediately and once only.

Note

When modifying, note the following:

Modifying can **not** be undone!

"Modify with trigger" command for tags

Introduction

You can assign values to addresses dependent on the defined monitoring and modify mode and modify them. The modify command is executed as specified in the monitoring and modify mode, with reference to the defined trigger position in the user program.

DANGER

Danger when modifying:

Serious personal injury and material damage can result from changes in the tags or addresses during plant operation in the event of malfunctions or program errors!
Make sure that dangerous conditions cannot occur before you execute the "Modify" function.

NOTICE

Danger of a time-out while controlling peripheral outputs

Note that the controlling of peripheral outputs in the watch table can result in a time-out.

The CPU assumes the "STOP" mode.

Requirements

- A watch table has been created.
- An online connection to the CPU exists.
- The watch table has to be in expanded mode.

Procedure

To modify tags "with trigger", follow these steps:

1. Enter the desired addresses and modify values in the watch table.
2. Select the addresses to be modified by selecting the check boxes for modifying in the column after the "Modify value".
   A yellow triangle appears behind the selected check box, indicating that the address is now selected for modifying but has not yet been modified.
3. Switch to expanded mode using the icon "Show/hide advanced settings columns" in the toolbar or the "Online > Expanded mode" command.
   The "Monitor with trigger" and "Modify with trigger" columns are displayed.
4. In the "Modify with trigger" column, select the desired modify mode from the drop-down list box. The following options are available:
   - Permanent
   - Permanently, at start of scan cycle
   - Once only, at start of scan cycle
   - Permanently, at end of scan cycle
   - Once only, at end of scan cycle
   - Permanently, at transition to STOP
   - Once only, at transition to STOP
5. Start modifying using the "Online > Modify > Modify with trigger" command.
6. Confirm the prompt with "Yes" if you want to start modifying with trigger.

Result

The selected tags are modified using the selected monitoring and modify mode. The yellow triangle is no longer displayed.

Note

When modifying, note the following:

Modifying can **not** be undone!

Enable peripheral outputs

Introduction

The "Enable peripheral outputs" function deactivates the command output disable of the peripheral outputs. You can then modify the peripheral outputs when the CPU is in STOP mode. This function is available in the watch table in "Expanded mode" only.

DANGER

Danger when enabling the peripheral outputs:

Attention, the enabling of the peripheral outputs can cause serious personal injury and material damage!
Make certain that dangerous conditions cannot occur before you execute the "Enable peripheral outputs" function.

Prerequisites

- A watch table has been created.
- An online connection to the CPU exists.
- The CPU is in STOP mode before you can enable the peripheral outputs.
- The watch table has to be in expanded mode.
- The option Force (Page 3038) of tags must not be enabled.

Note

"Enable peripheral outputs" function

- This function is possible only in STOP mode. The function is exited by an operating state change of the CPU and by the termination of the online connection.
- While the function is enabled, forcing is not possible.

Procedure

To enable the peripheral outputs in STOP mode, follow these steps:

1. Enter the desired addresses and modify values in the watch table.
2. Select the addresses to be modified by selecting the check boxes for modifying in the column after the "Modify value".
   A yellow triangle appears behind the selected check box, indicating that the address is now selected for modifying but has not yet been modified.
3. Switch to expanded mode using the icon "Show/hide advanced settings columns" in the toolbar or the "Online > Expanded mode" command.
   The "Monitor with trigger" and "Modify with trigger" columns are displayed.

4. Change the relevant CPU to STOP using the operator panel.
5. Right-click to open the shortcut menu and select "Enable peripheral outputs".
6. Confirm the prompt with "Yes" if you want to unlock the command output disable for the peripheral outputs.
7. Modify the peripheral outputs using the "Online > Modify > Modify now" command.

Result

The peripheral outputs are modified with the selected modify values. The yellow triangle is no longer displayed.

Enabling the peripheral outputs

The "Enable peripheral outputs" function remains active until:

- The "Enable peripheral outputs" command is deactivated again via the shortcut menu or via the "Online > Modify > Enable peripheral outputs" command.
- The CPU is no longer in STOP mode.
- The online connection is terminated.

Note

When modifying, note the following:

Modifying can **not** be undone!

9.4.4 Testing with the force table

9.4.4.1 Introduction for testing with the force table

Overview

You can use the force table to assign permanent values to individual tags

of the user program. This action is referred to as "forcing".

The following functions are available in the force table:

- **Monitoring tags**
  This displays the current values of the individual tags of a user program or a CPU on the programming device or PC. Tags can be monitored with or without a trigger condition.
- **Forcing tags**
  This function lets you assign a permanent value to individual peripheral tags of the user program.

Monitoring and forcing tags

The monitoring and forcing of tags is always dependent on the operand scope of the CPU used.

The following tags can be monitored:

- Inputs, outputs, and bit memories
- Contents of data blocks
- Peripheral inputs

The following tags can be forced:

- Peripheral inputs
- Peripheral outputs

Example

- Independent of the CPU used, only I/O can be forced, such as: "Tag_1":P or "QW0:P" or "IW0:P". Note that "Tag_1":P must not be the symbolic name of a bit memory.

Possible applications

One advantage of the force table is that you can simulate different test environments and overwrite tags in the CPU with a permanent value. This enables you to intervene in the ongoing process for regulating purposes.

See also

Layout of the force table (Page 3023)

Basic mode and expanded mode in the force table (Page 3024)

Icons in the force table (Page 3025)

Open and edit force table (Page 3026)

9.4.4.2 Safety precautions when forcing tags

Safety precautions when forcing tags

Because the forcing function allows you to intervene permanently in the process, observance of the following notices is essential:

DANGER

Prevent personal injury and material damage!

Note that an incorrect action when executing the "Force" function can:

- Harm persons or pose a health hazard.
- Cause damage to machinery or the entire plant.

CAUTION

Prevent personal injury and material damage!

- Before you start the "Force" function, you should ensure that no one else is currently executing this function on the same CPU.
- Forcing can only be stopped by clicking the "Stop forcing" icon or using the "Online > Force > Stop forcing" command. Closing the active force table does **not** stop the forcing!
- Forcing can **not** be undone!
- Review the differences between " modifying tags" (Page 3013) and "forcing tags" (Page 3038).
- If a CPU does not support the "Force" function, the relevant icons cannot be selected.
- If the function "Enable peripheral outputs" is active on your CPU, then forcing is **not** possible on this CPU. If desired, deactivate this function in the watch table.

9.4.4.3 Layout of the force table

Introduction

In the force table, enter the CPU-wide tags that you have defined and selected and which are to be forced in the allocated CPU. Only peripheral inputs and peripheral outputs can be forced.

For each CPU created in the project, a force table will automatically be created in the "Watch and force tables" folder. Only one force table can be allocated to a CPU. This force table displays all the addresses forced in the allocated CPU.

Layout of the force table

The columns displayed in the force table depend on the mode you are working in: basic mode or expanded mode.

In expanded mode the "Monitor with trigger" column is also displayed.

Meaning of the columns

The following table shows the meaning of the individual columns in basic mode and expanded mode:

Mode	Column	Meaning
Basic mode	i	Identification column
	Name	Name of the inserted tag
	Address	Address of the inserted tag
	Display format	Selected display format
	Monitor value	Values of the tags, dependent on the selected display format.
	Force value	Value with which the tag is forced.
	F ("Force")	Select the tag to be forced by activating the corresponding check box.
	Comment	Comment for documentation of the tags
The following additional column is shown in expanded mode:	Monitor with trigger	Display of selected monitoring mode

See also

Icons in the force table (Page 3025)

Basic mode and expanded mode in the force table (Page 3024)

9.4.4.4 Basic mode and expanded mode in the force table

Difference between basic mode and expanded mode in the force table

In expanded mode the "Monitor with trigger" column is also displayed in the force table.

You will find a detailed list of the columns under Layout of the force table (Page 3023).

Switching between basic mode and expanded mode

You have the following options of toggling between the basic and expanded mode:

- Click the icon "Show/hide advanced setting columns". Click this icon again to return to the basic mode.
  Or:
- Activate the check box for the "Expanded mode" command in the "Online" menu. Deactivate this check box to return to the basic mode.

Functionality in expanded mode

The following functionality is only possible in expanded mode:

- Monitor with trigger
- Monitor peripheral inputs

9.4.4.5 Icons in the force table

Meaning of the icons

The following table shows the meaning of the icons in the force table:

Icon	Meaning
	Identifies a table inside the project tree as a force table.
	Identification column
	Inserts a row before the selected row.
	Inserts a row after the selected row.
	Displays all columns of expanded mode. If you click this icon again, the columns of the expanded mode will be hidden.
	Starts forcing for all addresses of the selected tags. If forcing is already running, the previous action is replaced without interruption.
	Stops forcing of addresses in the force table.
	Starts monitoring of the visible tags in the force table. The default setting for monitoring in basic mode is "permanent". In expanded mode an additional column is shown and you can set certain trigger points for monitoring tags.
	Starts monitoring of the visible tags in the force table. This command is executed immediately and the tags are monitored once.
	Displays the check box for the selection of tags to be forced.
	Indicates that an address cannot be forced.
	Indicates that an address cannot be fully forced. Example: It is possible to force the address QW0:P, but it is not possible to force the address QD0:P because this address area is potentially not available on the CPU.

Icon	Meaning
	Indicates that an address cannot be monitored.
	Indicates that an address is being forced.
	Indicates that an address is being partly forced.
	Indicates that the associated peripheral address is being forced.
	Indicates that a syntax error occurred.
	Indicates that the address is selected but has not been forced yet.

See also

Layout of the force table (Page 3023)

9.4.4.6 Open and edit force table

Display force table

Introduction

You cannot create a new force table; one force table already exists for each CPU. It is permanently allocated to this CPU and cannot be copied or duplicated.

Requirements

A project with an allocated CPU has to be open.

Displaying a force table

The force table is always displayed below a CPU in the "Watch and force tables" folder.

Open force table

Requirements

A project with an allocated CPU must be created.

Procedure

Proceed as follows to open a force table:

1. Open the "Watch and force tables" folder below the desired CPU.
2. Double-click the "Force table" in this folder.

Result

The selected force table opens.

Save force table

Requirements

A project with an allocated CPU has been created.

Procedure

Proceed as follows to save a force table:

1. Enter the desired changes in the force table.
2. Select the "Save" command in the "Project" menu or click the "Save project" icon in the toolbar. Note that this save operation will save the entire project.

Result

The contents of the force table and the associated project are saved.

Note

You cannot rename a force table.

9.4.4.7 Entering tags in the force table

Basic principles for entering tags in the force table

Recommended procedure

Select the tags whose values you want to monitor or force, and enter them in the force table.

When entering tags in the force table, please note that these tags must be previously defined in the PLC tag table.

Example of filling out a force table

- You can enter the absolute address that is to be forced or monitored in the "Address" column or you can enter the symbolic name of the tag in the "Name" column.
- Select the display format you require from the drop-down list in the "Display format" column, if you do not want to use the default setting.

- Now you have to decide whether you want to monitor or force the entered tags. Enter the required force value and a comment in the appropriate columns of the force table.
- Note that only peripheral inputs and peripheral outputs can be forced and review the Safety precautions when forcing tags (Page 3039).

Create comment line

If required, you can create a comment row by entering the string "//" in the "Name" column.

Syntax check

When you enter tags in the force table, the syntax of each cell will be checked when you exit the cell. Incorrect entries are marked in red.

Note

When you place the mouse pointer in a cell marked in red, brief information is displayed with additional notes on the error.

Permitted operands for the force table

Permitted operands for the force table

The following table shows the operands that are permitted for forcing in the force table:

Permitted operand	Example of data type	Example (International mnemonics)
Peripheral input/peripheral output	BOOL	I0.0:P; Q0.0:P
Peripheral input/peripheral output	BYTE	IB1:P; QB1:P
Peripheral input/peripheral output	WORD	IW2:P; QW3:P
Peripheral input/peripheral output	DWORD	ID2:P; QD1:P

The following table shows the operands that are permitted for monitoring in the force table:

Permitted operand	Example of data type	Example (International mnemonics)
Input/output/bit memory	BOOL	I1.0, Q1.7, M10.1 I0.0:P
Input/output/bit memory	BYTE	IB1/QB10/MB100 IB1:P
Input/output/bit memory	WORD	IW1; QW10; MW100 IW2:P

Permitted operand	Example of data type	Example (International mnemonics)
Input/output/bit memory	DWORD	ID4; QD10; MD100 ID2:P
Timers	TIMER	T1
Counters	COUNTER	C1
Data block	BOOL	DB1.DBX1.0
Data block	BYTE	DB1.DBB1
Data block	WORD	DB1.DBW1
Data block	DWORD	DB1.DBD1

Note

You cannot enter "DB0..." because it is used by the system!

Permitted force values for the force table

Entering force values in the force table

The following table shows the operands that are permitted for entering force values in the force table:

Table 9-20 Bit operands

Possible bit operands	Example for permitted force values
I1.0:P	True
I1.1:P	False
Q1.0P	0
Q1.1:P	1
I2.0:P	2#0
I2.1:P	2#1

Table 9-21 Byte operands

Possible byte operands	Example for permitted force values
IB1:P	2#00110011
IB2:P	B#16#1F
QB14:P	1F
QB10:P	'a'
IB3:P	10

Table 9-22 Word operands

Possible word operands	Example for permitted force values
IW0:P	2#0011001100110011
IW2:P	W#16#ABCD
QW10:P	ABCD
QW12:P	B#(12, 34)
IW4:P	'ab'
IW6:P	12345
IW8:P	S5T#9S_340ms
IW10:P	C#123
IW12:P	D#2006-12-31

Table 9-23 Double word operands

Possible double word operands	Example for permitted force values
ID0:P	2#00110011001100110011001100110011
ID4:P	1.2
QD10:P	1.234.e4
QD14:P	Dw#16#abcdef10
ID8:P	16#ABCDEF10
ID12:P	b#(12,34,56,78)
ID16:P	L#-12
ID20:P	L#12
ID24:P	123456789
ID28:P	123456789
ID32:P	T#12s345ms
ID36:P	Tod#14:20:40.645
ID40:P	P#e0.0

Overview of the display formats

Display formats in the force table

The display format you select specifies the representation of a tag value.

When entering the address a display format is automatically preset. If you want to change this, you can select a display format from the drop-down list in the "Display formats" column. The drop-down list only offers the display formats which are valid for this data type. The display format that appears first in the list is the pre-selected format.

Example

The following table shows the 32-bit data types permitted for all CPU families in the force table and their possible display formats:

Data type	Possible display formats
BOOL	Bool, Hex, BCD, Octal, Bin, Dec, Dec+/-
BYTE	Hex, BCD, Octal, Bin, Dec, Dec+/-, Character
WORD	Hex, BCD, Octal, Bin, Dec, Dec+/-, Dec_Sequence, Character, SIMATIC_Timer, Date, Unicode_Character, Counter
DWORD	Hex, BCD, Octal, Bin, Dec, Dec+/-, Dec_Sequence, Character, Floating-point number, Time of day, Timer, Pointer, Unicode_Character
SINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character
INT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, SIMATIC_Timer, Counter, Date
DINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, Floating-point number, Time of day, Timer, Pointer
USINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character
UINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, SIMATIC_Timer, Counter, Date
UDINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, Floating-point number, Time of day, Timer, Pointer
REAL	Floating-point number, Hex, BCD, Octal, Bin, Character, Unicode_Character,Dec, Dec+/-, Dec_Sequence, Time of day, Timer, Pointer
DATE	Date, Dec, Hex, BCD, Bin
TIME_OF_DAY	Time of day, Dec, Hex, BCD, Bin
TIME	Timer, Hex, BCD, Bin
DATE_AND_TIME	Date and time,
TIMER	SIMATIC_Timer, Hex, BCD, Bin
CHAR	Character, Hex, BCD, Octal, Bin, Dec, Dec+/-
WCHAR	Unicode_Character, Character, Hex, BCD, Octal, Bin, Dec, Dec+/-
STRING	Character string
WSTRING	Unicode_character string
POINTER	Pointer, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec, Dec+/-, Dec_Sequence, Floating-point number, Time of day, Timer, Number
COUNTER	Counter, Hex, BCD, Bin
S5TIME	SIMATIC_Timer, Hex, BCD, Bin

For the S7-1200 CPU family, all 32-bit data types are permitted (see table above), as well as the 64-bit data type LREAL with the following possible display formats:

Data type	Possible display formats
LREAL	In a project created with TIA Portal < V12: Floating-point number Note: The display of LREAL is limited to 13 digits plus exponent.
LREAL	In a project created with TIA Portal >= V12: Floating-point number, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec, Dec+/-, Dec_Sequence, Time of day, Timer, Date and time Note: The display of LREAL is limited to 13 digits plus exponent.

For the S7-1500 CPU family, in addition to 32-bit data types, the 64-bit data types listed in the table are also permitted with the following possible display formats:

Data type	Possible display formats
LWORD	Hex, Octal, BCD, Bin, Character, Unicode_Character, Dec, Dec+/-, Dec_Sequence, Floating-point number, Time of day, Timer, Date and time
LINT	Dec+/-, Dec, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, Time of day, Timer, Date and time
ULINT	Dec, Dec+/-, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec_Sequence, Time of day, Timer, Date and time
LREAL	Floating-point number, Hex, BCD, Octal, Bin, Character, Unicode_Character, Dec, Dec+/-, Dec_Sequence, Time of day, Timer, Date and time
LTIME	Timer, Dec+/-, Dec, Hex
LTOD	Dec, Hex, BCD, Bin, Time of day
LDT	Dec, Hex, Date and time

For more information, refer to the description of the valid data types (Page 2236).

Note

Rounding of floating-point numbers

In the force table, floating-point numbers are stored as binary numbers in IEEE format. Because not every floating point number (real, longreal) that can be displayed on the user interface can be mapped to the IEEE format, there is a possibility that floating-point numbers will be rounded. If a rounded floating-point number in the force table is copied and, in turn, inserted in another input field, the rounding may cause a slight difference.

Note

Only symbolic addressing is possible

In the force table, LongDataTypes such as LWORD or LREAL can only be addressed symbolically.

Selecting the display format for tags

Procedure

To select the display format of the tags, follow these steps:

1. Enter the desired address in the force table.
2. Click the desired cell in the "Display format" column, and open the drop-down list.
   The permitted display formats are shown in the drop-down list.
3. Select the desired display format from the drop-down list.

Note

If the selected display format cannot be applied, then the last selected display format will be displayed automatically.

Creating and editing comment lines

Basic principles of comment lines

In addition to the row-related comments in the comment column, you can now also create complete comment lines to enhance the structure of the force table.

The contents of the comment line are stored in the "Languages & Resources" folder in the "Project texts" tab and can be compiled in other project languages.

Creating comment lines

To create comment lines, follow these steps:

1. Open the force table and enter the required addresses.
2. To create a comment line, enter the character string "//" in the "Name" column. No comment lines can be created in the other columns.

3. Enter the required comment in the comment line. The entered comment is shown in green.
4. To show all comments you entered, double-click "Project texts" in the project tree under "Languages & Resources".
5. If you are working in multi-lingual projects and want your comment to be translated into other languages, you can set the project languages required in addition to the editing language in the project tree under "Languages & Resources > Project languages".

Deleting comment lines

To delete comment lines, follow these steps:

1. Open a force table containing comment lines.
2. Delete the entire comment including the introductory string "//", if you no longer require this.
3. Alternatively, delete only the introductory string "//". In this case the existing comment is retained and is displayed in the "Comment" column in the force table.

Note

Deleting comment lines

When you delete comment lines the project languages and any existing translations for these comments are also deleted.

9.4.4.8 Monitoring tags in the force table

Introduction to monitoring tags in the force table

Introduction

Use the force table to monitor the tags of the configured input and output modules in the CPU, dependent on the monitoring mode (Page 3035) you have selected. An online connection to the CPU must exist to monitor tags.

Options for monitoring tags

The following options are available for monitoring tags:

- Monitor all
  This command starts the monitoring of all visible tags in the active force table, dependent on the selected monitoring mode:
  - In basic mode, the monitoring mode is set to "permanent" by default.
  - In expanded mode, you can specify defined trigger points for the monitoring of tags.

  Note

  If the monitoring mode is changed while in expanded mode and then a switch is made to basic mode, the monitoring mode set before will also be applied in basic mode.

- Monitor now
  This command starts the monitoring of the visible tags in the active force table immediately and once only.

CPU-specific limitations when monitoring tags

The following CPU-specific differences exist:

- CPU S7-300/400:
  CPUs from this family can only monitor the first 30 characters of a string.
- CPU S7-1200:
  CPUs from this family can monitor a string up to the total size of 254 characters.

Setting the monitoring mode in the force table

Introduction

By selecting the monitoring mode, you specify the trigger point and the duration of tag monitoring in the force table.

Possible monitoring mode (duration of monitoring)

The following selection options are available:

- Permanent
- Once only, at start of scan cycle
- Once only, at end of scan cycle
- Permanently, at start of scan cycle
- Permanently, at end of scan cycle
- Once only, at transition to STOP
- Permanently, at transition to STOP

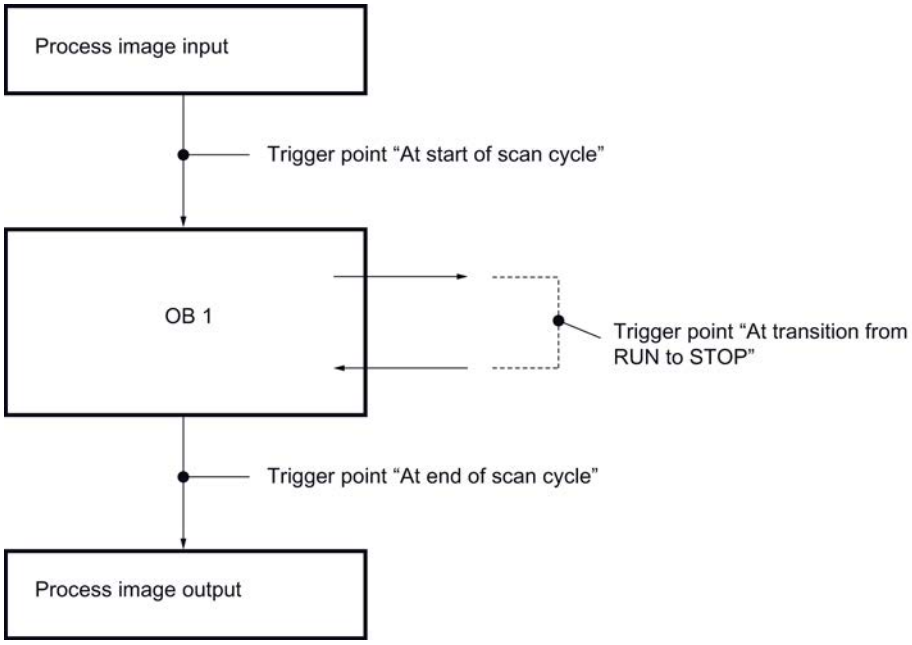
Special features when using "Permanent" mode

The "Permanent" mode is executed as follows for the monitoring of tags: The inputs are monitored at the end of the cycle and the outputs at the start of the cycle.

Selecting the trigger point

The trigger points "Beginning of scan cycle", "End of scan cycle", and "Switch to stop" specify the time at which the tags are to be read from the CPU or updated in the CPU.

The following diagram shows the position of these trigger points:



"Monitor all" command for tags

Introduction

Use the "Monitor all" command to start monitoring the visible tags in the active force table. In basic mode of the force table, the default setting for the monitoring mode is "permanent". In expanded mode, you can specify defined trigger points for the monitoring of tags. In this case, the tags are monitored with reference to the specified trigger points.

Requirements

An online connection to the CPU exists.

Procedure

To execute the "Monitor all" command, follow these steps:

1. Enter the tags to be monitored and the corresponding addresses in the force table.
2. Switch to expanded mode by clicking
   the icon "Show/hide advanced setting columns" in the toolbar.
3. If you want to change the default monitoring mode for a tag, click the appropriate cell in the "Monitor with trigger" column and select the desired monitoring mode from the drop-down list.
4. Click the "Monitor all" icon in the toolbar.

Result

The tags of the active force table will be monitored using the set monitoring mode.

"Monitor now" command for tags

Introduction

The "Monitor now" command starts the monitoring of tags immediately without reference to defined trigger points. The tag values are read out only once and displayed in the force table.

Requirements

An online connection to the CPU exists.

Procedure

To execute the "Monitor now" command, follow these steps:

1. Enter the tags to be monitored and the corresponding addresses in the force table.
2. Click the "Monitor now" icon in the toolbar.

Result

The tags of the active force table are monitored immediately and once only.

9.4.4.9 Forcing tags in the force table

Introduction to forcing tags

Introduction

You can use the force table to assign permanent values to individual tags of the user program. This action is referred to as forcing. Only peripheral inputs and peripheral outputs can be forced.

To use the forcing function, you must have an online connection to the CPU and the utilized CPU must support this functionality.

If you open a force table in the "Watch and force tables" folder below a CPU, all values forced in the allocated CPU will be displayed in this force table, provided that an online connection to the CPU exists.

Possible applications

By permanently assigning defined values to tags, you can specify defined default settings for your user program and, thus, test the programmed functions. Forcing is possible in basic mode and in expanded mode (Page 3024).

Caution when forcing tags

Before forcing, you must review the safety precautions (Page 3039) for this procedure.

DANGER

Prevent personal injury and material damage!

Note that an incorrect action when executing the "Force" function can:

- Harm persons or pose a health hazard.
- Cause damage to machinery or the entire plant.

Options for forcing tags

The following options are available for forcing tags:

- Force to "0"
  This command forces the selected address in the CPU to the force value "0".
- Force to "1"
  This command forces the selected address in the CPU to the force value "1".
- Force all
  This command starts the forcing of enabled addresses in the active force table or replaces an existing force job without interruption.
- Stop forcing
  This command stops the forcing of all addresses in the active force table.

Constraints when forcing tags

Note the following constraints when forcing:

- Forcing is always dependent on the operand scope of the CPU used.
- In principle, only peripheral inputs and peripheral outputs can be forced.
- If the function "Enable peripheral outputs" is active on your CPU, then forcing is not possible. If desired, deactivate this function in the watch table.

Unique aspects when forcing tags

Note that forcing of tags will overwrite values in the CPU and will continue even after the online connection to the CPU is terminated.

- Stop forcing
  Terminating the online connection is not sufficient to stop the forcing operation! To stop forcing, you must select the "Online > Force > Stop forcing" command. Only then will the tags that are visible in the active force table no longer be forced.
- Stop forcing of individual tags
  The "Online > Force > Stop forcing" command always applies to all tags displayed in the force table. To stop forcing individual tags, you must clear the check mark for forcing of these tags in the force table and restart forcing using the "Online > Force > Force all" command.

Safety precautions when forcing tags

Safety precautions when forcing tags

Because the forcing function allows you to intervene permanently in the process, observance of the following notices is essential:

DANGER

Prevent personal injury and material damage!

Note that an incorrect action when executing the "Force" function can:

- Harm persons or pose a health hazard.
- Cause damage to machinery or the entire plant.

CAUTION

Prevent personal injury and material damage!

- Before you start the "Force" function, you should ensure that no one else is currently executing this function on the same CPU.
- Forcing can only be stopped by clicking the "Stop forcing" icon or using the "Online > Force > Stop forcing" command. Closing the active force table does **not** stop the forcing!
- Forcing can **not** be undone!
- Review the differences between " modifying tags" (Page 3013) and "forcing tags" (Page 3038).
- If a CPU does not support the "Force" function, the relevant icons cannot be selected.
- If the function "Enable peripheral outputs" is active on your CPU, then forcing is **not** possible on this CPU. If desired, deactivate this function in the watch table.

Force tags to "0"

Introduction

You can use the force function to assign permanent values to individual tags of a user program.

Caution when forcing tags

Before forcing, you must review the safety precautions when forcing tags (Page 3039).

DANGER

Prevent personal injury and material damage!

Note that an incorrect action when executing the "Force" function can:

- Pose a risk to the life or health of persons.
- Cause damage to machinery or the entire plant.

Requirements

- An online connection to the CPU exists.
- The utilized CPU supports the force function.
- The "Enable peripheral outputs" function is **not** enabled on the CPU on which the tags are to be forced. If desired, deactivate this function in the watch table.

Procedure

To force tags to "0", follow these steps:

1. Open the force table.
2. Enter the desired address in the force table.
3. Select the "Online > Force> Force to 0" command in order to force the selected address with the specified value.
4. Confirm the next dialog with "Yes".

Result

The selected address is forced to "0". The yellow triangle is no longer displayed. A red "F" is displayed in the first column, for example, indicating that the tag is being forced.

Stop forcing

To stop forcing, follow these steps:

1. Open the force table.
2. Select the "Online > Force > Stop forcing" command.
3. Confirm the next dialog with "Yes".

Result

Forcing of the selected values is stopped. The red "F" in the first column is no longer displayed. The yellow triangle reappears behind the check box again to indicate that the address is selected for forcing but is not being forced at the moment.

Note

When forcing, note the following:

- Forcing **cannot** be undone!
- Terminating the online connection does **not** stop the forcing!
- To stop forcing, the forced address must be visible in the active force table.

Force tags to "1"

Introduction

You can use the force function to assign permanent values to individual tags of a user program.

Caution when forcing tags

Before forcing, you must review the safety precautions when forcing tags (Page 3039).

DANGER

Prevent personal injury and material damage!

Note that an incorrect action when executing the "Force" function can:

- Endanger the life or health of personnel
- Cause damage to machinery or the entire plant.

Requirements

- An online connection to the CPU exists.
- The utilized CPU supports the force function.
- The "Enable peripheral outputs" function is **not** enabled on the CPU on which the tags are to be forced. If desired, deactivate this function in the watch table.

Procedure

To force tags to "1", follow these steps:

1. Open the force table.
2. Enter the desired address in the force table.
3. Select the "Online > Force> Force to 1" command in order to force the selected address with the specified value.
4. Confirm the next dialog with "Yes".

Result

The selected address is forced to "1". The yellow triangle is no longer displayed. A red "F" is displayed in the first column, for example, indicating that the tag is being forced.

Stop forcing

To stop forcing, follow these steps:

1. Open the force table.
2. Select the "Online > Force > Stop forcing" command.
3. Confirm the next dialog with "Yes".

Result

Forcing of the selected values is stopped. The red "F" in the first column is no longer displayed. The yellow triangle reappears behind the check box again to indicate that the address is selected for forcing but is not being forced at the moment.

Note

When forcing, note the following:

- Forcing **cannot** be undone!
- Terminating the online connection does **not** stop the forcing!
- To stop forcing, the forced address must be visible in the active force table.

"Force all" command for tags

Introduction

You can use the force function to assign permanent values to individual tags of a user program.

If forcing is already active, this forcing operation is replaced without interruption by the "Online > Force > Force all" command. Any forced addresses that are not selected will no longer be forced.

Caution when forcing tags

Before forcing, you must review the safety precautions when forcing tags (Page 3039).

DANGER

Prevent personal injury and material damage!

Note that an incorrect action when executing the "Force" function can:

- Pose a risk to the life or health of persons.
- Cause damage to machinery or the entire plant.

Requirements

- An online connection to the CPU exists.
- The utilized CPU supports the force function.
- The "Enable peripheral outputs" function is **not** enabled on the CPU on which the tags are to be forced. If desired, deactivate this function in the watch table.

Procedure

To force tags with the "Online > Force > Force all" command, follow these steps:

1. Open the force table.
2. Enter the desired addresses and force values in the force table.
3. Select the addresses to be forced by selecting the check boxes for forcing in the column after the "Force value".
   A yellow triangle appears behind the selected check box, indicating that the address is selected for forcing but is not being forced at the moment.
4. Select the "Online > Force> Force all" command in order to force the selected addresses with the specified values.
5. Confirm the next dialog with "Yes".

Result

The selected addresses are forced to the specified values. The yellow triangle is no longer displayed. A red "F" is displayed in the first column, for example, indicating that the tag is being forced.

Stop forcing

To stop forcing, follow these steps:

1. Open the force table.
2. Select the "Online > Force > Stop forcing" command.
3. Confirm the next dialog with "Yes".

Result

Forcing of the selected addresses is stopped. The red "F" in the first column is no longer displayed. The yellow triangle reappears behind the check box again to indicate that the address is selected for forcing but is not being forced at the moment.

Note

When forcing, note the following:

- Forcing **cannot** be undone!
- Terminating the online connection does **not** stop the forcing!
- To stop forcing, the forced address must be visible in the active force table.

9.4.4.10 Stop forcing tags

Stop forcing all tags

Introduction

Note the following before you stop forcing tags:

- Stopping forcing **cannot** be undone!
- Terminating the online connection does **not** stop the forcing!
- To stop forcing, the forced address must be visible in the active force table.

Caution when forcing tags

Before forcing, you must review the safety precautions when forcing tags (Page 3039).

DANGER

Prevent personal injury and material damage!

Note that an incorrect action when stopping the "Force" function can:

- Pose a risk to the life or health of persons.
- Cause damage to machinery or the entire plant.

Requirements

- Tags are forced in a force table.
- An online connection to the CPU exists.
- The utilized CPU supports the force function.
- The "Enable peripheral outputs" function is not enabled on the CPU on which the tags are to be forced. If desired, deactivate this function in the watch table.

Procedure

Proceed as follows to stop **forcing all tags** :

1. Open the force table.
2. Select the "Online > Force > Stop forcing" command in order to stop forcing the displayed addresses.
3. Confirm the "Stop forcing" dialog with "Yes".

Result

The forcing of all tags is stopped. The red "F" in the first column is no longer displayed. The yellow triangle reappears behind the check box again to indicate that the address is flagged for forcing but is not being forced at the moment.

Stop forcing individual tags

Introduction

Note the following before you stop forcing tags:

- Stopping forcing **cannot** be undone!
- Terminating the online connection does **not** stop the forcing!
- To stop forcing, the forced address must be visible in the active force table.

Caution when forcing tags

Before forcing, you must review the safety precautions when forcing tags (Page 3023).

DANGER

Prevent personal injury and material damage!

Note that an incorrect action when stopping the "Force" function can:

- Pose a risk to the life or health of persons.
- Cause damage to machinery or the entire plant.

Requirements

- Tags are forced in a force table.
- An online connection to the CPU exists.
- The utilized CPU supports the force function.
- The "Enable peripheral outputs" function is not enabled on the CPU on which the tags are to be forced. If desired, deactivate this function in the watch table.

Procedure

Proceed as follows to stop **forcing individual tags** :

1. Open the force table.
2. Deactivate the check boxes for the addresses that are no longer to be forced.
3. Reselect the "Online > Force" command.

Result

Forcing of the disabled addresses will be stopped. The red "F" in the first column is no longer displayed. The yellow triangle reappears behind the check box again to indicate that the address is flagged for forcing but is not being forced at the moment.

9.5 Configuring alarms

9.5.1 Introduction to alarm configuration

Overview

Alarms allow you to detect errors in process control in the automation system quickly, to localize them precisely, and to eliminate them. This leads to a significant reduction in down times in the plant.

Before alarms can be output, they need to be configured.

You can create, edit and compile event-dependent alarms along with their alarm texts and alarm attributes and display them on display devices.

The table below lists the alarm types along with a brief description of their functions.

Alarm type	Description
Program alarms	Program alarms are used to report program-synchronous events and are each assigned to a block. They are created in the program editor and edited in the alarm editor.
System diagnostic alarms	System diagnostic alarms are configuration-dependent module events and are activated or deactivated in the hardware configuration. They can only be viewed, not edited, in the alarm editor.
User diagnostic alarms	By means of user diagnostic alarms, you can write a user entry to the diagnostics buffer and send a corresponding alarm. They are assigned to a CPU. They are created in the alarm editor and can be edited there.

9.5.2 Assigning alarm numbers

Assigning numbers

The alarms are identified by a number that is unique within the CPU. This means that it is not necessary to recompile after copying complete programs. Copying a single block is an

exception to this rule. Here, recompilation is necessary to include the changed alarm number in the program.

9.5.3 Components of an alarm

Overview

How an alarm is displayed depends on the alarm method, the alarm block used and the display device.

The possible components of an alarm are listed in the following table:

Component	Description
Time stamp	Generated when the alarm event occurs in the automation system.
Alarm state	The following are possible: Incoming, outgoing, outgoing without acknowledgment, outgoing with acknowledgment.
Associated value	With some alarms, it is possible to add a process value that can be evaluated by the alarm instruction being used.
Image	If there is a system crash, the resulting alarms can be displayed later on HMI devices.
Alarm number	A number assigned by the system that is unique within the CPU and identifies an alarm.
Alarm texts	Are configured by the user.

9.5.4 Available alarm blocks

S7-300/400: Overview of the alarm blocks

You can choose from the following alarm blocks:

- ALARM
- ALARM_8
- ALARM_8P
- NOTIFY
- ALARM_S
- ALARM_SQ

- AR_SEND (for sending archives; no configuration of alarm texts and alarm attributes possible)
- NOTIFY_8P
- ALARM_DQ
- ALARM_D

S7-300/400: When do I use which alarm block?

The following table will help you to decide which alarm block to use for your application. The choice of alarm block depends on the following factors:

- The number of channels available in the block and therefore the number of signals to be monitored per block call.
- The possibility of acknowledging alarms.
- The possibility of including associated values.
- The display devices to be used.
- The configuration limits of the CPU.

Alarm block	Channels	Acknowl-edgment	Associated values	Special features
ALARM	1	Yes	up to 10	Sends an alarm on rising or falling edge
ALARM_8	8	Yes	no	Sends an alarm on rising or falling edge of one or more signals
ALARM_8P	8	Yes	up to 10	as ALARM_8
NOTIFY	1	no	up to 10	as ALARM
NOTIFY_8P	8	no	up to 10	as ALARM
AR_SEND	1	-	-	Used to send an archive; no configuration of alarm texts and alarm attributes possible
ALARM_SQ	1	Yes	1	With each block call and a signal change compared with the previous block call, an alarm is generated
ALARM_S	1	no	1	as ALARM_SQ
ALARM_DQ	1	Yes	1	as ALARM_SQ
ALARM_D	1	no	1	as ALARM_SQ

S7-1500: Overview of the alarm blocks

The following alarm block is available:

- Program_Alarm

The selection of the corresponding alarm class determines whether the block requires acknowledgment or not. The default setting is that the block does not require acknowledgement.

Alarm block	Channels	Acknowl-edgment	Associated values	Special features
Program_Alarm	1	possible	up to 10	Sends an alarm with time stamp on rising or falling edge

9.5.5 Alarm type and alarms

Using the program and alarm editors, you have the option of creating either an alarm type (i.e. an FB as template for instance DBs) or alarms themselves (i.e. instance DBs as template of an alarm type).

The block with alarm capability can be an FB or an instance DB

- With an FB you create an alarm type that serves as a template for alarms. All the input you make for the alarm type is automatically included for alarms derived from it. If you assign an instance DB to the FB, messages are generated automatically for the instance DB according to the template of the alarm type and alarm numbers are assigned.
- With an instance DB, you can modify the alarms generated based on the alarm type for the specific instance.

The visible difference is that alarm numbers are assigned to alarms whereas they are not assigned to an alarm type. The alarm types and corresponding instances are arranged one under the other in the alarm editor.

Locking the data for an alarm type

You enter texts and attributes for event-dependent alarms in the alarm editor. While you are doing this, you can, for example, specify how the alarms are displayed on certain display devices (for example using the alarm class). Use the alarm types as templates to facilitate creation of alarms.

- When you enter data (attributes and texts) for the alarm type, you can specify whether or not they are locked. If you lock attributes, the icon of a closed chain link is shown beside the input box. If the attribute is not locked, the chain link is open.
- If you lock data in the alarm type, you can no longer change this in the alarms of the specific instance. The data is simply displayed.
- If, however, you want to make changes, you will need to go back to the alarm type, cancel the lock and make any changes there. The changes do not, however, apply to instances generated prior to the change.

Changing the data for an alarm type

If you change data for an alarm type, these changes are automatically included in the instances. Exceptions: You have already changed this data in the instance or have locked or unlocked data later in the alarm type.

Note

If you copy instances to a different program without copying the alarm type, then the instance will not be completely displayed. In this case, copy the alarm type to the new program as well.

Resetting instance-specific data to the value of the alarm type

If attributes or texts were overwritten in an alarm instance, a type icon is displayed beside the attribute. You can decide whether to use the value of the alarm type again for each attribute. In this case, no type icon is displayed.

See also

Creating and editing an alarm type (Page 3056)

Creating and editing an instance DB (Page 3056)

9.5.6 Formal parameters, alarm data types and alarm blocks

Formal parameters as alarm number inputs

For each alarm or group of alarms, you require a formal parameter (name of the alarm) in your program that you specify as an IN parameter in the tag overview of your program. The formal parameter is used as an alarm number input and forms the basis for an alarm.

Supplying the formal parameter with the suitable data type

The formal parameter must be assigned an alarm data type in keeping with the alarm block being used.

Alarm data types and corresponding alarm blocks

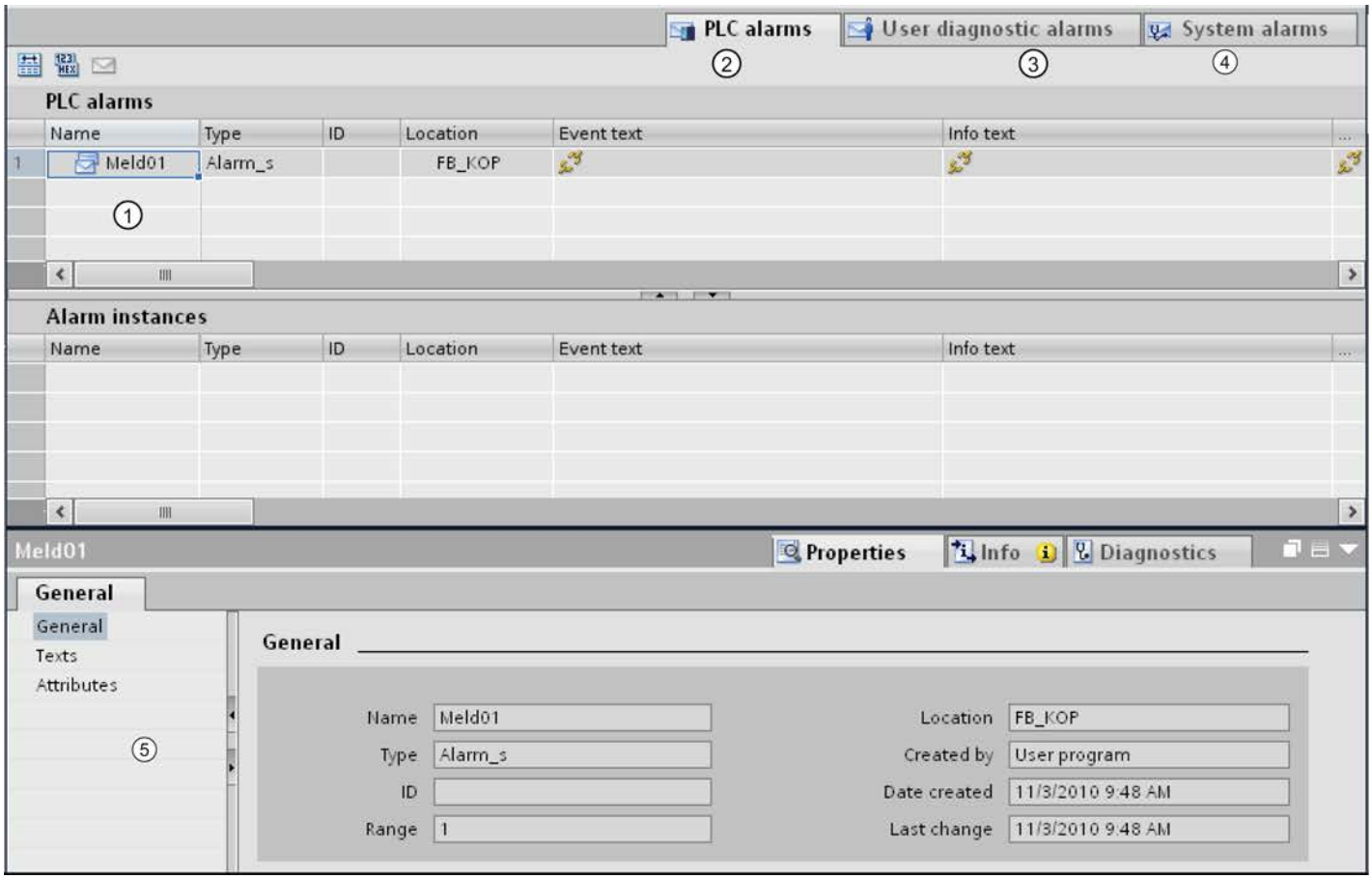
The following table shows the alarm data types with their corresponding alarm blocks and their properties. The values of the data types have the same names as those of the alarm blocks (exception: "alarm_s") and and have the prefix "C_".

SIMATIC system	Data type	Alarm block	Properties
S7-400	C_Alarm_8	ALARM_8	8 channels, with acknowledgment, no associated values
S7-400	C_Alarm_8p	ALARM_8P	8 channels, with acknowledgment, up to 10 associated values per channel
S7-400	C_Notify	NOTIFY	1 channel, with acknowledgment, up to 10 associated values
S7-400	C_Alarm	ALARM	1 channel, with acknowledgment, up to 10 associated values
S7-300/400	C_Alarm_s	ALARM_S	1 channel, no acknowledgment, up to 1 associated value
S7-300/400	C_Alarm_s	ALARM_SQ	1 channel, with acknowledgment, up to 1 associated value
S7-400	C_Ar_Send	AR_SEND	Used to send an archive
S7-400	C_Notify_8p	NOTIFY_8P	8 channels, no acknowledgment, up to 10 associated values
S7-300/400	C_Alarm_s	ALARM_DQ	1 channel, with acknowledgment, up to 1 associated value
S7-300/400	C_Alarm_s	ALARM_D	1 channel, no acknowledgment possible, up to 1 associated value
S7-1500	Program_Alarm	Program_Alarm	1 channel, acknowledgment possible by parameter assignment, up to 10 associated values

9.5.7 Layout of the alarm editor

Layout of the alarm editor

The following graphic shows an example of the components in the alarm editor:



1		Tabular display of the alarms in the work area, see: Auto-Hotspot The table above contains the alarm type; the table below contains the instances (instance DBs) derived from the alarm type (FB).
2		"Program alarms" tab: You can edit program alarms here.
3		"System alarms" tab: System alarms can only be viewed and cannot be edited. To edit these alarms, you must change to the device view ("Go to device" menu command), where you can modify the system alarms in the Inspector window.
4		"User diagnostic alarms" tab: User diagnostic alarms can be created and edited here. This function is not available for S7-1500 CPUs.
5		Auto-Hotspot

You can enter or modify the necessary parameters, texts and attributes in the table or in the Inspector window.

9.5.8 Creating and editing alarms

9.5.8.1 Creating program alarms

Requirement

You have created a function block.

Procedure

To create a program alarm, follow these steps:

1. In the "Program blocks" folder in the project navigation, select the function block (FB) for which you want to create a program alarm and double-click the block to open it.
2. Fill in the block interface. For each alarm block that will be called in the FB, you will need to declare tags in the calling FB.
   For this purpose, enter the following tags, for example:
   – For the "IN" parameter, a name for the alarm block input, for example "Alarm01" (for alarm input 01) and the data type.
3. In the instruction window of the FB, insert the call for the selected alarm block, e.g. "CALL ALARM_S" and complete your input with the RETURN key.
   Result: The instruction part of the FB displays the input tags of the called alarm block, in this case the ALARM_S block.
4. Assign the name that you entered for the alarm block input in step 2 to the "EV_ID" tag, in this case "Alarm01".

   Note

   If, instead of an alarm block in the CPU, you call an FB with multiple instances in which alarms are also configured, you also need to configure the alarms of the FB with multiple instances in the calling block.

5. Repeat steps 2 through 4 for all alarm block calls in this FB.

9.5.8.2 Editing program alarms in the alarm editor

Requirement

You have created a program alarm.

Procedure

To edit program alarms, follow these steps:

1. Double-click "PLC alarms" in the project navigation. The alarm editor opens.
2. Enter the required texts and attributes in the appropriate columns.

Note

When you edit alarm types, you can lock texts and attributes. You do this by clicking on the icon in front of the relevant column.

If you edit alarms in instance DBs in which the texts/attributes are not locked in the alarm type, a type icon is displayed in front of the relevant column. When you click on this icon, modified texts/attributes are reset to the values of the alarm type.

9.5.8.3 Editing program alarms in the program editor

Requirement

You have created a program alarm.

Procedure

To edit program alarms, follow these steps:

1. Select the appropriate line in the block interface.
2. Move to the "Alarm" tab in the Inspector window and select the required group.
3. Enter the required texts and attributes in the appropriate fields.

Note

When you edit alarm types, you can lock texts and attributes. You do this by clicking on the icon next to the relevant field.

If you edit alarms in instance DBs in which the texts/attributes are not locked in the alarm type, a type icon is displayed next to the relevant field. When you click on this icon, modified texts/attributes are reset to the values of the alarm type.

9.5.8.4 Deleting program alarms

Procedure

To delete a program alarm, follow these steps:

1. Open the block containing the alarm you want to delete.
2. Select the corresponding line in the block interface and select "Delete" in the shortcut menu.

Result

The alarm is deleted.

9.5.8.5 Creating and editing an alarm type

Procedure

To edit an alarm type, follow these steps:

1. Select the required alarm block.
2. Enter the texts you require in the appropriate columns or select the required attributes.
   If you have selected a multichannel alarm block (for example, "ALARM_8"), you can assign a separate text and certain separate attributes to each subnumber.
3. If you do not want the texts or attributes of the instance to be changed, lock them in the alarm type.

9.5.8.6 Creating and editing an instance DB

Prerequisites

You have already created an FB and created at least one alarm in it.

Procedure

To assign instance data blocks (DBs) to an alarm type and to edit the alarms for these DBs for specific instances, follow the steps below:

1. Double-click on "Add new block" in the project navigation, click on the "Data block (BD)" button that appears in the dialog and select the function block (alarm type) to which you want to assign the instance DB from the "Type" drop-down list.
2. Now click in the inspector window on the "Alarm" tab and select the group you want.
   Or:
   Double-click in the project navigation on "PLC alarms" to open the alarm configuration.
3. Make the changes you require for the particular instance DB.

Note

If the properties of the instance DB are write-protected then you must first unlock them in alarm type (FB).

Result

This completes the alarm configuration for the selected instance DB.

9.5.8.7 Creating user diagnostic alarms

User diagnostic alarms are assigned to a CPU. They are created and edited in the alarm editor.

Procedure

To create a user diagnostic alarm, follow these steps:

1. Double-click on the "PLC alarms" folder in project navigation to open the alarm editor.
2. Select the "User diagnostic alarms" tab in the alarm editor.
3. Click in the table and select "Insert new alarm" in the shortcut menu.

Result

You have created a user diagnostic alarm.

9.5.8.8 Editing user diagnostic alarms

Requirement

You have created a user diagnostic alarm.

The alarm editor is open.

Procedure

To edit a user diagnostic alarm, follow these steps:

1. Enter the required texts and attributes in the appropriate columns.

9.5.8.9 Deleting user diagnostic alarms

You can delete a selected alarm. The texts you configured for this alarm are deleted and the alarm number is released for use.

Procedure

To delete a user diagnostic alarm, follow these steps:

1. Select the corresponding row in the table and select "Delete" in the shortcut menu.

Result

The alarm is deleted. It is no longer displayed in the table.

9.5.8.10 Insert associated values in alarms

To add current information to alarms, e.g. from the process, you can insert associated values at any location within an alarm text.

Procedure

To insert an associated value into an alarm, follow these steps:

Alarm SFB for the S7-400 (e.g. Alarm8)

1. Configure a block as follows:
   @<No. of associated value>%<Element type><Format specification>@.
2. Insert this block at the locations in the alarm text at which the associated value is to be shown.

Alarm SFC for the S7-300/400 (e.g. Alarm_S)

The maximum length of an associated value in the AlarmS/SQ is a maximum of 12 bytes.

1. Configure a block as follows:
   @<Index>%<Element type><Format specification>@.
   Index: Index in the associated value. The index is interpreted as an array index starting with 1.
   The byte offset is calculated with the following formula:
   Offset = ((Index - 1) * data width (element type))
   Use only element types Y, W, X, or R. The element type is used here only to determine the data width. The display type is determined by the format specification.
2. Insert this block at the locations in the alarm text at which the associated value is to be shown.

See also

Structure of associated values (Page 3059)

Examples of associated values (Page 3060)

9.5.8.11 Structure of associated values

Associated values are comprised of the following:

Element type

This uniquely configures the data type of the associated value:

Element type	Data type
Y	BYTE
W	WORD
X	DWORD
I	Integer
D	DINT
B	BOOL
C	CHAR
R	REAL

The element type only uniquely identifies the data type transferred by the AS. It is not used as Casting Operator.

Format

Determine the output format for the associated value on the display device. The format is preceded by the "%" sign. The following fixed formats apply to alarm texts:

Format	Description
%[i]X	Hexadecimal number with i digits
%[i]u	Decimal number without sign with i digits
%[i]d	Decimal number with sign with i digits
%[i]b	Binary number with i digits
%[i][.y]f	Floating-point number with sign with y digits after the decimal point and total number of digits i
%[i]s	String (ANSI string) with i digits Characters are printed up to the first 0 Byte (00Hex).
%t#<Name of text library>	Access to text library

If the format is too small, the value is nevertheless output in full.

If the format is too large, an appropriate number of empty characters is output before the value.

If the format is too large and if a sign is output (for decimal/floating-point numbers), the number of empty characters is reduced by one. The same applies to periods or commas within floating-point numbers.

Example: Value = 1234,567 Format=%3,8f Output=1234,567; that is, no empty character is output even though there are only 7 digits.

Note

Please note that you can optionally enter "[i]" (without the square brackets).

See also

Insert associated values in alarms (Page 3058)

Examples of associated values (Page 3060)

9.5.8.12 Examples of associated values

Examples of associated values:

@1I%6d@: The value from associated value 1 is shown as a decimal number with a max. 6 digits.

@2R%6f@: The value "5.4", for instance from associated value 2, is shown as a fixed number of points "5.4" (three leading blanks).

@2R%2f@: The value "5.4", for instance from associated value 2, is shown as a fixed number of points "5.4" (if number of digits is too small then these are not cut off).

@1W%t#Textbib1@: Associated value 1 of the WORD data type is the index by which the text to be inserted is referenced in the Textbib1 text library.

Examples of formats and output values:

Value=255 Format=%5X Output="000FF"

Value=255 Format=%5X Output="000FF"

Value=123 Format=%5u Output=" 123"

Value=-200 Format=%4d Output="-200"

Value=19 Format=%8b Output="2#00010011"

Value = 1234,567 Format=%8.3f Output="1234.567"; that is, no empty character is output even though there are only 7 digits.

Note

If you want to transfer more than one associated value to an ALARM_S block then you can transfer an array with a max. length of 12 bytes. These could be, e.g. maximum 12 Byte or Char, maximum 6 Word or Int or maximum 3 DWord, Real or Dint.

See also

Insert associated values in alarms (Page 3058)

Structure of associated values (Page 3059)

9.5.8.13 Deleting associated values

You can delete associated values by deleting the string that represents an associated value in the alarm text.

Procedure:

Follow the steps below to delete associated values:

1. Find the block in the alarm text that corresponds to the associated value you want to delete. The block begins with the "@" character followed by the location ID by which you can recognize the associated value, there is then format information and it ends with the "@" character.
2. Delete the block you have found from the alarm text.

9.5.9 Creating and editing alarms

9.5.9.1 Creating program alarms

Requirements

You have created a function block.

Procedure

To create a program alarm, follow these steps:

1. In the "Program blocks" folder in the project navigation, select the function block (FB) for which you want to create a program alarm and double-click the block to open it.
2. In the instruction window of the FB, insert the call for the selected alarm block ("Program_Alarm") and complete your input with the RETURN key. The block interface is automatically provided with the necessary information in the "Static" area.
   Result: The instruction part of the FB displays the input tags of the called alarm block, in this case the Program_Alarm block.
3. Repeat steps 3 through 4 for all alarm block calls in this FB.

Please note that you must specify the alarm class when you create alarms for S7-1500 CPUs. Indicates whether the alarm requires acknowledgment. Alarms that do not require acknowledgment can also be used only for information purposes.

As a result, there are three types of alarm:

- The alarm has to be acknowledged and has the status "incoming" or "outgoing".
- The alarm does not have to be acknowledged and has the status "incoming" or "outgoing".
- The alarm is used only for information purposes. The check mark must be set in the "Information only" column.

Note

The default setting is without acknowledgement with the status "incoming" or "outgoing".

See also

Creating alarm classes (Page 3073)

9.5.9.2 Editing program alarms in the alarm editor

Requirement

You have created a program alarm.

Procedure

To edit program alarms, follow these steps:

1. Double-click "PLC alarms" in the project navigation. The alarm editor opens.
2. Enter the required texts and attributes in the appropriate columns.

Note

When you edit alarm types, you can lock texts and attributes. You do this by clicking on the icon in front of the relevant column.

If you edit alarms in instance DBs in which the texts/attributes are not locked in the alarm type, a type icon is displayed in front of the relevant column. When you click on this icon, modified texts/attributes are reset to the values of the alarm type.

9.5.9.3 Editing program alarms in the program editor

Requirements

You have created a program alarm.

Procedure

To edit program alarms, follow these steps:

1. Select the appropriate line in the block interface.
2. Move to the "Alarm" tab in the Inspector window and select the required group.
3. Enter the required texts and attributes in the appropriate fields.

Note

When you edit alarm types, you can lock texts and attributes. You do this by clicking on the icon next to the relevant field.

If you edit alarms in instance DBs in which the texts/attributes are not locked in the alarm type, a type icon is displayed next to the relevant field. When you click on this icon, modified texts/attributes are reset to the values of the alarm type.

"Program_alarm" alarm block

If you use the "Program_alarm" alarm block, the additional "Configuration" tab is displayed in the Inspector window. You can enter additional associated values, which for example do not occur in texts, at the empty input channels SD_1 to SD_10.

For this, click the selection box next to the parameter and select the desired value.

9.5.9.4 Deleting program alarms

Procedure

To delete a program alarm, follow these steps:

1. Open the block containing the alarm you want to delete.
2. Select the corresponding line in the block interface and select "Delete" in the shortcut menu.

Result

The alarm is deleted.

9.5.9.5 Creating and editing an alarm type

Procedure

To edit an alarm type, follow these steps:

1. Select the required alarm block.
2. Enter the texts you require in the appropriate columns or select the required attributes.
3. If you do not want the texts or attributes of the instance to be changed, lock them in the alarm type.

9.5.9.6 Creating and editing an instance DB

Prerequisites

You have already created an FB and created at least one alarm in it.

Procedure

To assign instance data blocks (DBs) to an alarm type and to edit the alarms for these DBs for specific instances, follow the steps below:

1. Double-click on "Add new block" in the project navigation, click on the "Data block (BD)" button that appears in the dialog and select the function block (alarm type) to which you want to assign the instance DB from the "Type" drop-down list.
2. Now click in the inspector window on the "Alarm" tab and select the group you want.
   Or:
   Double-click in the project navigation on "PLC alarms" to open the alarm configuration.
3. Make the changes you require for the particular instance DB.

Note

If the properties of the instance DB are write-protected then you must first unlock them in alarm type (FB).

Result

This completes the alarm configuration for the selected instance DB.

9.5.9.7 Insert associated values in alarms

To add current information to alarms, for example from the process, you can insert associated values at various positions within an alarm text.

Procedure

To insert an associated value into an alarm, follow these steps:

1. Open the shortcut menu at the desired position, and use the corresponding menu command to select whether you want to insert a keyword, a tag, or an entry from a text list.
2. Enter the desired parameters in the dialog that appears, and confirm your entries.

Note

You can configure associated values both in the alarm editor and the block editor. If you work in the alarm editor, you should have opened the block for which you are configuring the alarm in the block editor. Otherwise, only already configured tags, but no newly added or created tags, can be interconnected or shown in the "Tag" selection box.

Result

The associated value is integrated into the alarm. If there is a syntax error, the text is displayed with a red background and must be corrected.

When tags and text list entries are entered, the "Address" and alarm parameter fields are automatically filled out. They are not displayed until the dialog is opened again. The associated value is added automatically at input channel SD1 to SD10.

See also

Examples of associated values (Page 3067)

Settings when entering associated values (Page 3066)

9.5.9.8 Settings when entering associated values

You can insert associated values in alarms both in the alarm editor and the block editor. If you work in the alarm editor, you should have opened the block for which you are configuring the alarm in the block editor. As a result of this, the "Tag" selection box will display the available parameters, which you can then apply. If the block editor is closed, you will see only the previously used parameters.

Entering a keyword as an associated value

The possible keywords that you can define as an associated value are available for selection in the drop-down list.

Entering a text list entry as an associated value

You select the desired text list in the drop-down list. You then select the desired text list entry in the "Tag" selection box.

Entering a tag as an associated value

You select the desired tag in the "Tag" selection box. The address and the alarm parameter are added automatically by the program. In the "Format" section, you must now define the display type suitable for the specified tag, the minimum length, and, if applicable, the number of decimal places.

Note

Projects created with an older versions of the TIA Portal can still contain the "old" (manual) notation with embedded associated values. This corresponds to the syntax @<Number of the associated value>%<Format specification>@. For information on the "Format specification" parameter, refer to Structure of associated values (Page 3059).

Use only the above described dialog boxes to enter associated values in new projects.

See also

Keywords for alarm texts (Page 3068)
Insert associated values in alarms (Page 3065)

9.5.9.9 Examples of associated values

Examples of associated values:

Keyword:

<Keyword: CpuName>

The name of the CPU is entered as an associated value.

Tag:

<Tag: #Temperature>

The temperature is entered as an associated value.

Text list entry:

<Text list: USER_1: "tag_1">

The specified text list entry is entered as an associated value.

See also

Insert associated values in alarms (Page 3065)

9.5.9.10 Deleting associated values

Procedure:

Follow the steps below to delete associated values:

1. Open the "Configuration" tab in the program editor.
2. Select the required associated value and delete it.

Or

1. Select the required input channel in the program editor.
2. Delete the associated value.

9.5.10 Texts and attributes

9.5.10.1 Keywords for alarm texts

Overview of keywords for alarm texts

You can enter the following keywords in alarm texts, info texts, or additional texts. These are then replaced with their real values by the engineering system.

PLC name

Name of the configured PLC.

Example: <Keyword: CpuName> = "PLC_1"

Name of the instance DB

Instance of instance alarm, usually a data block.

Example: <Keyword: Instance> = "Block_3_DB"

FB nesting for alarm

If you are using nested blocks in the project, the path is shown that leads to there from the function block from which the data block is generated.

Example: <Keyword: Path> = "myFB2\myFB1"

Note

For blocks that are not nested, the "Path" keyword is replaced with an empty string.

Name of the alarm

Name of the alarm.

Example: <Keyword: Name> = "myAlarm"

Note

Keywords cannot be entered in text lists.

9.5.10.2 Entering texts

You can enter the texts for alarms manually or you can use the default values.

Text template from the alarm type

All the texts in the alarm type are available as a template for creating alarm texts. If the alarm type already contains a general text, all instances of this alarm type include the same attributes and texts. Where necessary, you simply need to modify them.

Info text

The info text is a text that can be specified for certain display devices. With certain groups of devices (for example WinCC), it can be changed during run time.

Additional texts

Additional texts are texts that can be displayed by certain HMI devices. Click in the relevant row and type in the text. If you want to protect the text from being overwritten, click the option in the column. The texts can include line breaks.

See also

Locking texts (Page 3069)

9.5.10.3 Locking texts

"Locked" option in the alarm type

You can only lock texts when you edit the alarm type. The locked texts in alarms derived from the alarm type are write-protected. The icon displayed beside the input field indicates whether or not they are locked.

Locking texts

Follow the steps below to lock texts:

1. Start by editing the alarm types.
2. Click on the icon beside the input box you want to lock.
   Result: The icon changes to a closed chain link.

Unlocking texts

To unlock texts, follow the steps below:

1. Start by editing the alarm types.
2. Click on the icon beside the input box you want to unlock.
   Result: The icon changes to an open chain link.

9.5.10.4 Locking attributes

Locking attributes in the alarm type

You can only lock attributes when you edit the alarm type. The locked attributes in alarms derived from the alarm type are write-protected. The icon in front of the input field indicates whether the attribute is locked.

Locking attributes

Follow the steps below to lock attributes:

1. Start by editing the alarm types.
2. Click on the icon to the left of the input box you want to lock in the table.
   Result: The icon changes to a closed chain link.

Unlocking attributes

Follow the steps below to unlock attributes:

1. Start by editing the alarm types.
2. Click on the icon to the left of the input box you want to unlock in the table.
   Result: The icon changes to an open chain link.

9.5.11 Text lists for alarms

9.5.11.1 Basics of alarm text lists

You can adapt existing text lists (user-defined and system-defined text lists) to your requirements and edit texts and attributes. You can then translate the texts into the project language(s) you want to use.

If a signaling block which references a text list is copied into another program, you must either also copy the associated text lists, create another text list with the same name and contents, or change the reference in the alarm text

Note

Associated values within text list entries are not supported or are not resolved and displayed as plain text.

Detailed information on text lists is contained in the "Working with text lists" chapter.

See also

Text lists (Page 409)

Creating user-defined text lists (Page 410)

Editing user-defined text lists (Page 411)

Editing system-defined text lists (Page 411)

9.5.11.2 Editing text lists for alarms

Requirements

- You have created a user-defined text list.

Procedure

Follow the steps below to edit user-defined text lists:

1. Double-click on the "Text lists" command below the device in the project navigation.
   The text list editor opens.
2. Select the text list you want to edit from the table.
3. Change the values as required.

You can change the following values:

- Text titles:
  With the exception of the info text, the titles for the alarm texts (alarm text, additional texts) can be freely configured to suit your purposes.
- Static texts:
  Here you can change the previously created texts or insert new texts in an existing text list. Note that the entry of associated values in text lists is not permitted.

See also

Editing system-defined text lists (Page 411)

9.5.11.3 Integrating texts from text lists in alarms

You can integrate texts from various text lists in an alarm. The texts can be positioned anywhere you want which means that they can be used in alarms in foreign languages.

Procedure S7-300/400

To integrate texts from text lists in alarms, follow the steps below:

1. Double-click "PLC alarms" in the project navigation. The alarm editor opens.
2. Put a placeholder in the format @[Index][Data type]%t#[Text list]@ at the point in the alarm at which you want the text from the text list to appear.

Note

S7-300/400: [Index] = e.g. 1, where 1 is the first associated value of the alarm.

Procedure S7-1500

To integrate texts from text lists in alarms, follow the steps below:

1. Double-click "PLC alarms" in the project navigation. The alarm editor opens.
2. Go to the point in the alarm at which you want the text from the text list to appear, and select from the shortcut menu the option to insert a text list.
3. You select the desired text list in the drop-down list. You then select the required text list entry in the "Tag" selection box.

9.5.11.4 Example of integrating texts from text lists in alarms

S7-300/400: Configured alarm text: Pressure has risen @2W%t#textlist1@.

S7-1500: Configured alarm text: Pressure has <Text list: USER_1: "tag_1"> increased.

Text list with the name "textlist1":

Index	German	English
1734	zu hoch	too high

The associated value is supplied with the value 1734.

The following alarm text is displayed (if the project language is set to "English" on the operator panel): Pressure has risen too high.

See also

Integrating texts from text lists in alarms (Page 3071)

9.5.12 Alarm classes

9.5.12.1 Creating alarm classes

You can configure alarm classes to suit your purposes. You can create and edit them in the alarm class editor. An alarm can then by assigned to an alarm class in the alarm editor.

Two alarm classes "Ackownledgement" (for alarms with acknowledgment) and "No Ackownledgement" (for alarms without acknowledgment) are already created by the system. These cannot be modified.

A total of 16 alarm classes can be created. The "With acknowledgment" property of the alarm classes indicates that its is mandatory to acknowledge the assigned alarms.

Requirement

You have opened the "Common data" folder in project navigation.

Procedure

To configure an alarm class, follow these steps:

1. Double click on the "Alarm classes" entry in project navigation.
   The alarm class editor opens.
2. Select "Insert new alarm class" in the shortcut menu.
3. Enter a unique name for the new alarm class in the "Name" column.
   The language of the name you assign here is neutral.
4. Enter a display name in the "Display name" column. This name is translatable.
5. Specify whether or not alarms of this alarm class require acknowledgment in the "With acknowledgment" column.

9.5.12.2 Editing alarm classes

You can change the settings (name, display name, or acknowledgment) for an alarm class at any time even if alarms have already been assigned to the alarm class. The changes are adopted automatically in the alarms.

Copying alarm classes

To copy alarm classes, follow these steps:

1. Select the row with the alarm class you want to copy.
2. Select "Copy" in the shortcut menu.
3. Select "Paste" in the shortcut menu.

Result

The copied alarm class is appended to the end of the table under a new name.

The name of the new alarm class is made up as follows:

<old name><no.>

no.: This the lowest free natural number.

9.6 Using Team Engineering

9.6.1 Exchanging data with Inter Project Engineering (IPE)

9.6.1.1 Basics of Inter Project Engineering (IPE)

Introduction to Inter Project Engineering (IPE)

The functionality of Inter Project Engineering, hereinafter referred to as IPE, is used to exchange the controller data in a source project between different projects with the help of a device proxy.

You can then transfer this data to other projects, for visualization in HMI, for example, and use it there for further configuration.

Inter Project Engineering gives you a simple way of providing controller data from a PLC programming across multiple projects for an HMI configuration.

Using the object "Device Proxy Data", you exchange controller data across multiple projects consistently and easily without redundant configuration overheads.

This makes it possible for HMI project engineers to work on a project without the hardware configuration having to be present in their project. This means they can work on the PLC project and on the HMI project at the same time.

You can exchange the following controller data between projects via the device proxy:

- Program blocks
- Technological objects
- PLC tags
- PLC alarms

The following data is automatically included in the exchange:

- Controller interfaces
- Configured communication processors and communication modules

This automatic data exchange ensures that the interfaces and the configured communication processors and modules are transferred along with the data you selected. This data is required to allow you to continue working consistently and without problems in your target project.

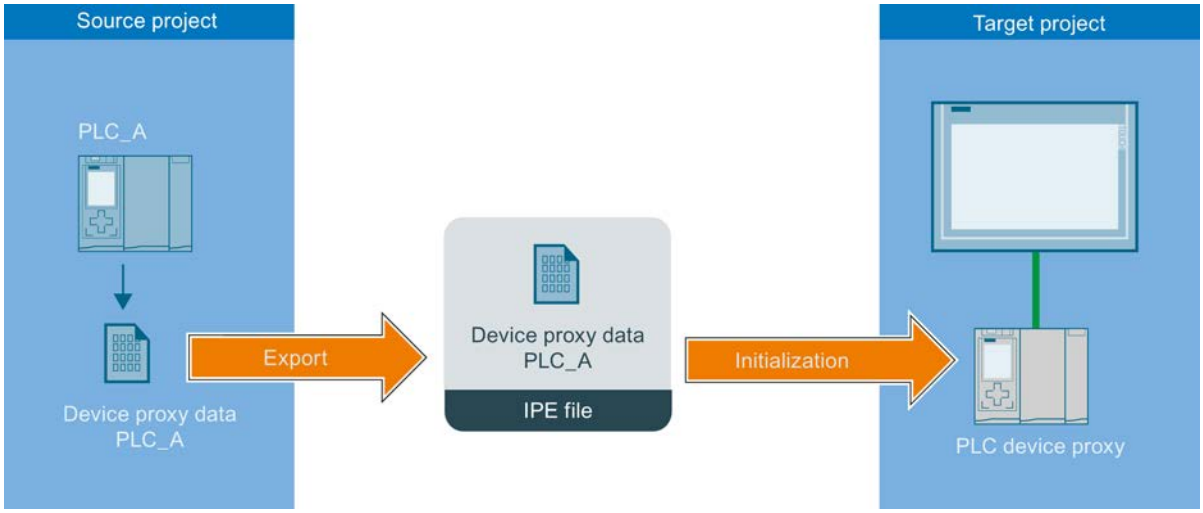
Cross-project data exchange

The following options are available for cross-project data exchange with Inter Project Engineering:

- Exchanging controller data via IPE file
- Exchanging controller data via project file

Exchanging controller data via IPE file

The figure below shows cross-project data exchange via an IPE file.

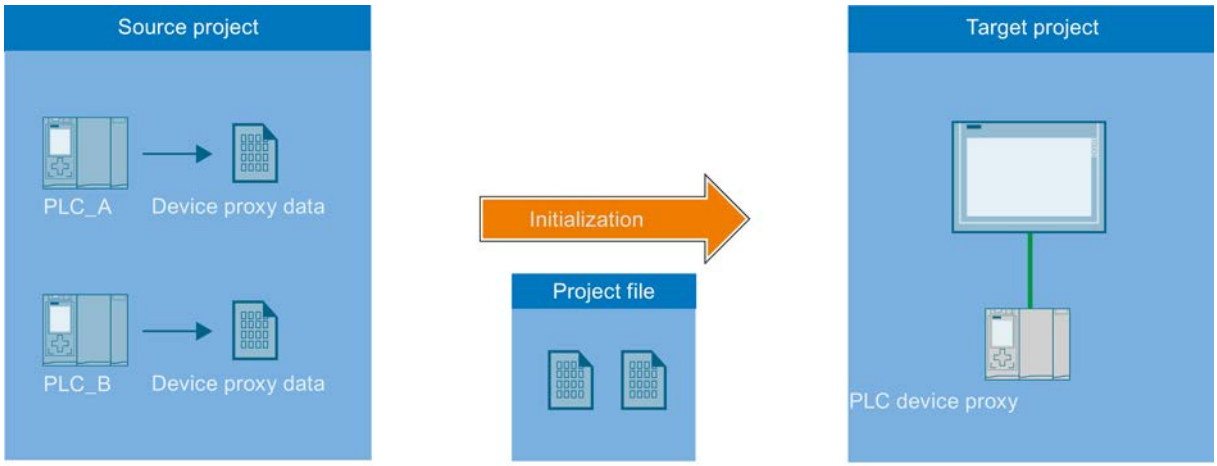


The PLC controller data from the source project is transferred to the PLC in the target project via the object "Device Proxy Data".

Controller data that is exchanged via an IPE file must originate from TIA Portal Projects V13 or higher.

Exchanging controller data via a project file

The figure below shows cross-project data exchange via a project file.



The PLC controller data from the source project is transferred to the PLC in the target project via a project file.

You can easily transfer and continue to use controller data from older TIA Portal projects as well as STEP 7 Classic projects that are not integrated in TIA Portal in your current TIA Portal projects.

9.6.1.2 Requirements for Inter Project Engineering (IPE)

Software and hardware requirements

To use the functionality of Inter Project Engineering, the minimum software and hardware requirements for the installation of TIA Portal V13 or higher must be met.

The following requirements also apply:

- You have installed TIA Portal V13 or higher and the "SIMATIC STEP 7 Professional" software package.
- You are using a CPU of the S7-1200/1500 or S7-300/400 series.
- The same software version must be installed on all involved controllers.

Requirements for IPE

The requirements for accepting controller data from projects are as follows:

- You have created a project with hardware configuration and PLC data that can be transferred to another project.
- You have made sure that the project is consistent before you transfer the controller data by means of the device proxy data.
- You are familiar with the defined procedures for Inter Project Engineering.

Notes on compatibility mode

The functions for Inter Project Engineering are not available in compatibility mode.

9.6.1.3 Procedure for working with Inter Project Engineering (IPE)

Exchanging controller data across projects

You can use the functionality of Inter Project Engineering to exchange existing controller data between different projects with the help of a device proxy and to continue using this data in another project. The device proxy provides you with a consistent and convenient way of exchanging data between projects without redundant configuration overheads.

Procedure for exchanging controller data via an IPE file

To exchange data via an IPE file, follow these steps:

1. Using TIA Portal V13 or higher, create a project with all required controller data and an executable user program.
2. Configure the hardware required for the project.
3. Compile the project to ensure consistency in your project.
4. In your project, create the object "Device Proxy Data" below the required CPU by clicking the corresponding command "Add device proxy data" in the "Device Proxy Data" folder.
   Result: The "Device Proxy Data" object has been created.
5. Select the required device proxy data in the project tree and double-click the object to open it in the editor.
6. Enter the required data for the device proxy data in the editor under "General" or apply the default setting. You can change the name and enter a comment for export of the controller data.
   In the "Define content" area of the editor select which of the existing controller data you wish to export as IPE file and click "Export device proxy data".
7. In the subsequent dialog, enter the name and storage location for the IPE file to be created and click "Save". You will be notified as soon as the export has been completed successfully.
   Result: The selected controller data are exported as IPE file.
8. In your target project, create a device proxy to be able to import the controller data contained in the IPE file. To do so, click the command "Add new device" and select the required device proxy in the dialog that opens.
9. Click the newly created device proxy and select the "Initialize device proxy" command from the shortcut menu.
10. Select the previously created IPE file to be used for the initialization, and confirm the selection in the following dialog with "OK". This starts the initialization.
    Result: The controller data from the source project contained in the IPE file is transferred to the device proxy in the target project.

After the successful exchange of controller data, you can continue with the configuration in your target project and, for example, connect PLC tags with HMI tags. The controller data can be updated as often as required when it is changed in the source project.

Procedure for exchanging controller data via a project file

When exchanging controller data via a project file, device proxy data can either be predefined in the source project or it can be defined and selected from the target project.

Follow these steps to exchange data via a project file:

1. Using TIA Portal V13 or higher, create a project with all required controller data and an executable user program.
2. Configure the hardware required for the project.
3. Compile the project to ensure consistency in your project.
4. Open the TIA Portal project from which you want to import the controller data into the target project.
5. In your project, create the object "Device Proxy Data" below the required CPU by clicking the corresponding command "Add device proxy data" in the "Device Proxy Data" folder.
   Result: The "Device Proxy Data" object has been created.
6. Enter the required data for the device proxy data in the editor under "General" or apply the default setting. You can change the name and enter a comment for the device proxy data.
7. In the "Define content" area of the editor select which of the existing controller data you want to provide by means of the device proxy data.
8. Click "Save" to save the changes in the project.
   Result: You have successfully saved the selected controller data in your device proxy data.
9. In your target project, create a device proxy to import the controller data contained in the project file. To do so, click the command "Add new device" and select the required device proxy in the dialog that opens.
10. Click the newly created device proxy and select the "Initialize device proxy" command from the shortcut menu.
11. Select the previously created project file with the prepared device proxy data to be used for the initialization, and confirm the selection in the following dialog with "OK". This starts the initialization.
    Result: The controller data from the source project contained in the project file is transferred to the device proxy in the target project.

After the successful exchange of controller data, you can continue with the configuration in your target project and, for example, connect PLC tags with HMI tags. The controller data can be updated as often as required when it is changed in the source project.

Procedure for updating already transferred controller data

System requirements for the update:

The following requirements must be met for updating already transferred controller data with the help of the already created "Device Proxy Data" object:

- You are using the same project for the update as for the previous transfer of the controller data.
- You are using the already created "Device Proxy Data" for transfer of the controller data.
- You have not made any changes to the hardware configuration or the communication interfaces in the meantime. You can add new hardware components.

Updating procedure:

1. Open the TIA Portal project from which you want to update the controller data in the target project.
2. Double-click the existing "Device Proxy Data" object and select under content which controller data you want to make available for the update.
3. Save the project or export the IPE file once again.
4. Open the required target project in the TIA Portal.
5. Click the existing device proxy in the target project and select the "Update device proxy" command from the shortcut menu.
6. Select the source which is to be used for the update, and confirm the selection.
7. In the following dialog, select under "Device" the CPU that was previously used as source, from which the controller data is to be re-imported.
8. Under "Defined device proxy data", select the required object and exit the dialog with "OK".
9. The update is then started and the selected device proxy data is transferred from the source project to the device proxy in the target project.

After the successful update of the controller data, you can continue with the configuration in your target object and, for example, connect PLC tags with HMI tags. The controller data can be updated as often as required when it is changed in the source project.

9.6.1.4 Creating device proxy data

Creating device proxy data

To create a device proxy data, follow these steps:

1. Open the folder "Device Proxy Data" below the CPU to which you want to create a device proxy data.
2. Click the command "Add new device proxy data".

Result

The new device proxy data is created and displayed in the project tree.

See also

Basics of Inter Project Engineering (IPE) (Page 3074)

Requirements for Inter Project Engineering (IPE) (Page 3076)

Procedure for working with Inter Project Engineering (IPE) (Page 3077)

Opening device proxy data (Page 3080)

Exchange of controller data via IPE file (Page 3080)

9.6.1.5 Opening device proxy data

Opening device proxy data

To open a device proxy data in the editor, follow these steps:

1. Open the required "Device Proxy Data" with a double-click in the "Device Proxy Data" folder in the project tree below the CPU.

Result

The device proxy data is opened in the editor and displays the controller data assigned to it.

You can either apply the defaults displayed in the editor or customize them as needed.

See also

Exchange of controller data via IPE file (Page 3080)

Creating device proxy data (Page 3079)

Basics of Inter Project Engineering (IPE) (Page 3074)

Requirements for Inter Project Engineering (IPE) (Page 3076)

Procedure for working with Inter Project Engineering (IPE) (Page 3077)

9.6.1.6 Exchange of controller data via IPE file

Exchange of controller data via "IPE file"

To export the controller data of the higher-level CPU displayed in the device proxy data, follow these steps:

1. Open the TIA Portal project from which you want to import the controller data into the target project.
2. Compile the project to ensure consistency in your project.
3. In your project, create the object "Device Proxy Data" below the required CPU by clicking the corresponding command "Add Device Proxy Data" in the "Device Proxy Data" folder.
   Result: The "Device Proxy Data" object has been created.
4. Select the device proxy data in the project tree and double-click the object to open it in the editor.
5. Enter the required data for the device proxy data in the editor under "General" or apply the default setting. You can change the name and enter a comment.

6. In the "Define content" area of the editor select which of the existing controller data you want to provide by means of the device proxy data.
7. Click the command "Export device proxy data" to export.
8. In the subsequent dialog, enter the name and storage location for the data to be exported and click "Save". The ".ipe" format is automatically specified as file format for the device proxy data file during the export.

Result

The selected device proxy data is exported and a progress bar for the export is displayed.

You receive a message as soon as the export is successfully completed.

The controller data of the selected CPU were stored in the saved ".ipe" file and can be conveniently forwarded, for example, by e-mail. The controller data can be imported in other devices without additional configuration, for example, for visualization and can be used further.

See also

Basics of Inter Project Engineering (IPE) (Page 3074)

Requirements for Inter Project Engineering (IPE) (Page 3076)

Procedure for working with Inter Project Engineering (IPE) (Page 3077)

Creating device proxy data (Page 3079)

Opening device proxy data (Page 3080)

9.6.1.7 Exchange of controller data via project file

Exchange of controller data via a project file

When exchanging controller data via a project file, device proxy data must be predefined in the source project. To export the controller data of the higher-level CPU displayed in the device proxy data as project file, follow these steps:

1. Open the TIA Portal project from which you want to import the controller data into the target project.
2. Compile the project to ensure consistency in your project.
3. In your project, create the object "Device Proxy Data" below the required CPU by clicking the corresponding command "Add Device Proxy Data" in the "Device Proxy Data" folder.
   Result: The "Device Proxy Data" object has been created.
4. Select the device proxy data in the project tree and double-click the object to open it in the editor.
5. Enter the required data for the device proxy data in the editor under "General" or apply the default setting. You can change the name and enter a comment.

6. In the "Define content" area of the editor select which of the existing controller data you want to provide by means of the device proxy data.
7. Click "Save" to save the changes in the project.
   Result: You have successfully saved the selected controller data in your device proxy data.
8. Open the target project into which the controller data are to be imported. In your target project, create a device proxy to import the controller data contained in the project file. To do so, click the command "Add new device" and select the required device proxy in the dialog that opens.
9. Click "Save" to save the changes in the project.
10. Click the newly created device proxy and select the "Initialize device proxy" command from the shortcut menu.
11. Select the previously created project file with the prepared device proxy data to be used for the initialization, and confirm the selection in the following dialog with "OK".
12. In the following dialog, select the required CPU and select the prepared device proxy data. Confirm the dialog with "OK". This starts the initialization.
    Result: The controller data from the source project contained in the project file is transferred to the device proxy in the target project.

After the successful exchange of controller data, you can continue with the configuration in your target project and, for example, connect PLC tags with HMI tags. The controller data can be updated as often as required when it is changed in the source project.

9.7 Programming examples

9.7.1 Programming examples (S7-1200, S7-1500)

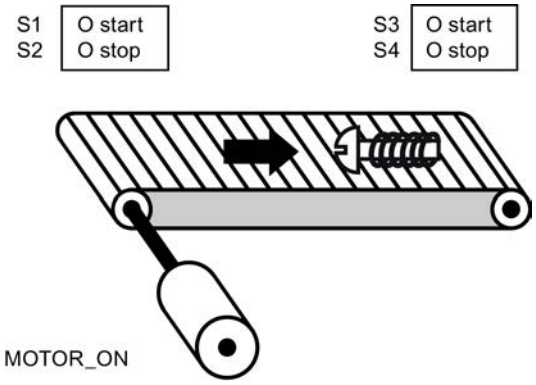
9.7.1.1 LAD programming examples

Example of controlling a conveyor belt

Controlling a conveyor belt

The following figure shows a conveyor belt that can be activated electrically. There are two pushbutton switches at the beginning of the conveyor belt: S1 for START and S2 for STOP.

There are also two pushbutton switches at the end of the conveyor belt: S3 for START and S4 for STOP. It is possible to start and stop the conveyor belt from either end.



Implementation

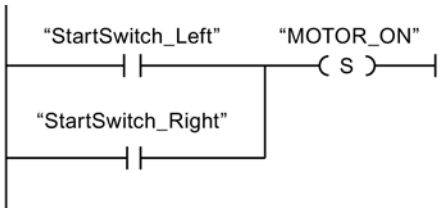
The following table shows the definition of the tags used:

Name	Data type	Description
StartSwitch_Left (S1)	BOOL	Start switch on the left side of the conveyor belt
StopSwitch_Left (S2)	BOOL	Stop switch on the left side of the conveyor belt
StartSwitch_Right (S3)	BOOL	Start switch on the right side of the conveyor belt
StopSwitch_Right (S4)	BOOL	Stop switch on the right side of the conveyor belt
MOTOR_ON	BOOL	Turn on the conveyor belt motor

The following networks show the LAD programming for solving this task:

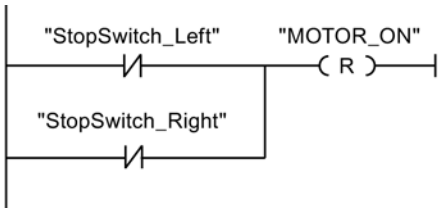
Network 1:

The conveyor belt motor is switched on when Start switch "S1" or "S3" is pressed.



Network 2:

The conveyor belt motor is switched off when Stop switch "S2" or "S4" is pressed.



See also

---(S)---: Set output (Page 3161)

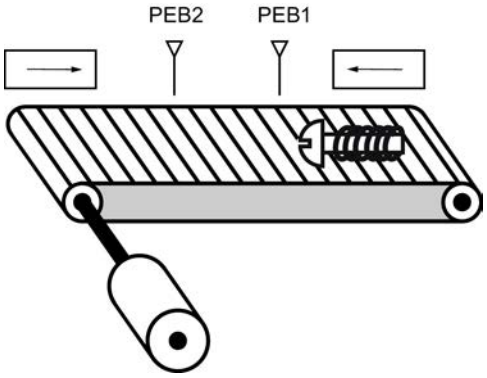
---| |---: Normally open contact (Page 3156)

---(R)---: Reset output (Page 3160)

Example of detecting the direction of a conveyor belt

Detecting the direction of a conveyor belt

The following figure shows a conveyor belt that is equipped with two photoelectric barriers (PEB1 and PEB2). The photoelectric barriers are designed to detect the direction in which an object is moving on the conveyor belt.



Implementation

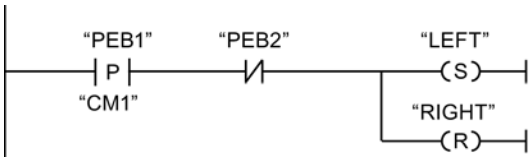
The following table shows the definition of the tags used:

Name	Data type	Description
PEB1	BOOL	Photoelectric barrier 1
PEB2	BOOL	Photoelectric barrier 2
RIGHT	BOOL	Display during movement to right
LEFT	BOOL	Display during movement to left
CM1	BOOL	Edge bit memory 1
CM2	BOOL	Edge bit memory 2

The following networks show the LAD programming for solving this task:

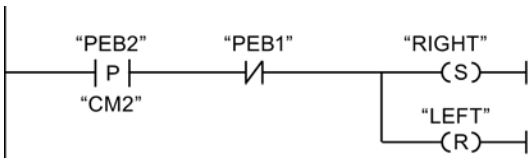
Network 1:

If the signal state changes from "0" to "1" (positive edge) at photoelectric barrier "PEB1" and, at the same time, the signal state at "PEB2" is "0", the object on the belt is moving to the left.



Network 2:

If the signal changes from "0" to "1" (positive edge) at photoelectric barrier "PEB2" and, at the same time, the signal state at "PEB1" is "0", the object on the belt is moving to the right.



See also

---(S)---: Set output (Page 3161)

---| / |---: Normally closed contact (Page 3157)

---(R)---: Reset output (Page 3160)

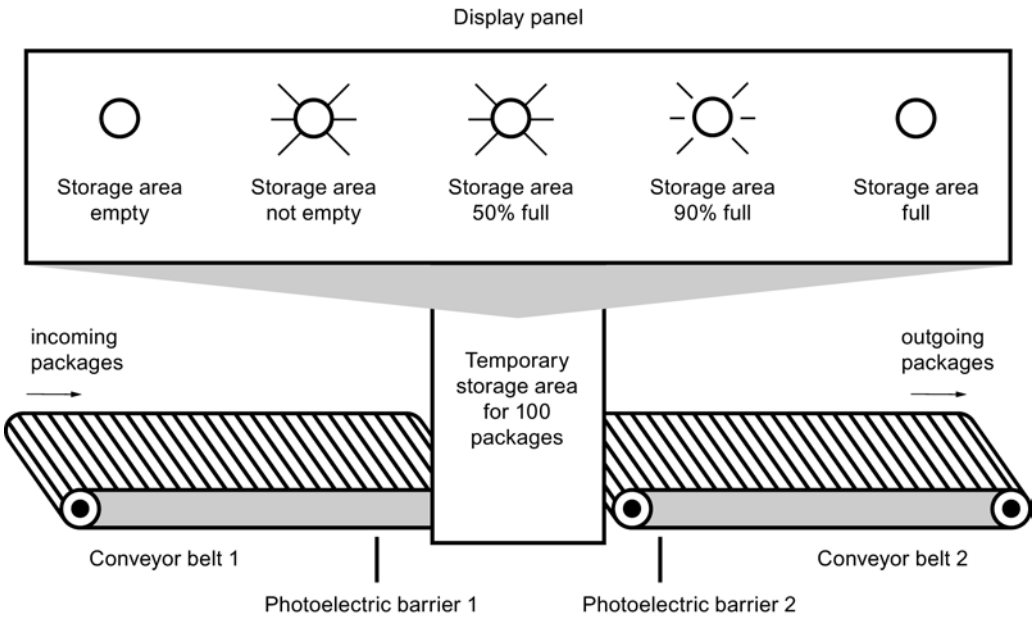
--|P|--: Scan operand for positive signal edge (Page 3167)

Example of detecting the fill level of a storage area

Detecting the fill level of a storage area

The following figure shows a system with two conveyor belts and a temporary storage area between them. Conveyor belt 1 delivers packages to the storage area. A photoelectric barrier at the end of conveyor belt 1 near the storage area detects how many packages are delivered to the storage area. Conveyor belt 2 transports packages from the temporary storage area to a loading dock onto which the packages are loaded for delivery to customers by truck. A photoelectric barrier at the storage area exit detects how many packages leave the storage area to be transported to the loading dock. Five display lamps indicate the capacity of the temporary storage area.

When a conveyor belt is restarted, the current count value is set to the number of packages available in the storage area.



Implementation

The following table shows the definition of the tags used:

Name	Data type	Description
PEB1	BOOL	Photoelectric barrier 1
PEB2	BOOL	Photoelectric barrier 2
RESET	BOOL	Reset counter
LOAD	BOOL	Set counter to value of "PV" parameter
STOCK	INT	Stock at restart
PACKAGECOUNT	INT	Number of packages in the storage area (current count value)
STOCK_PACKAGES	BOOL	Is set if the current count value is greater than or equal to the value of the tag "STOCK".
STOR_EMPTY	BOOL	Display lamp: Storage area empty
STOR_NOT_EMPTY	BOOL	Display lamp: Storage area not empty
STOR_50%_FULL	BOOL	Display lamp: Storage area 50% full
STOR_90%_FULL	BOOL	Display lamp: Storage area 90% full
STOR_FULL	BOOL	Display lamp: Storage area full
VOLUME_50	INT	Comparison value: 50 packages

Name	Data type	Description
VOLUME_90	INT	Comparison value: 90 packages
VOLUME_100	INT	Comparison value: 100 packages

The following networks show the LAD programming for activating the lamps:

Network 1:

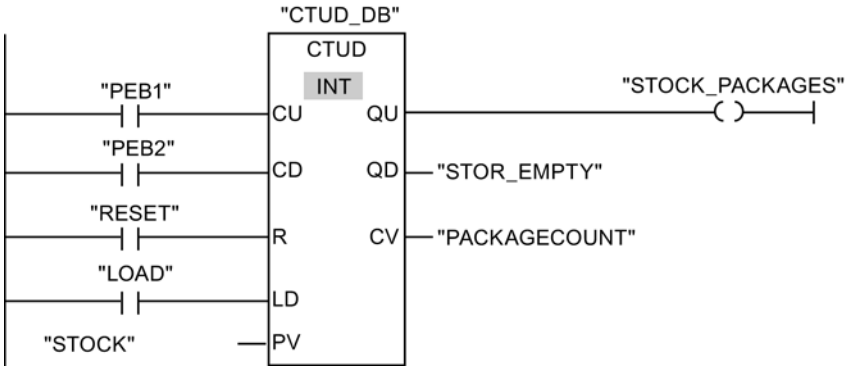
When a package is delivered to the storage area, the signal state at "PEB1" switches from "0" to "1" (positive signal edge). On a positive signal edge at "PEB1", the "Up" counter is enabled, and the current count value of "PACKAGECOUNT" is increased by one.

When a package is delivered from the storage area to the loading dock, the signal state at "PEB2" switches from "0" to "1" (positive signal edge). On a positive signal edge at "PEB2", the "Down" counter is enabled, and the current count value of "PACKAGECOUNT" is decreased by one.

If there are no packages in the storage area ("PACKAGECOUNT" = "0"), the "STOR_EMPTY" tag is set to signal state "1", and the "Storage area empty" lamp is switched on.

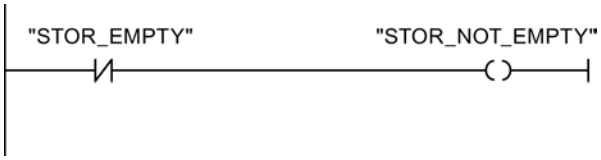
The current count value can be reset to "0" if the "RESET" tag is set to signal state "1".

If the "LOAD" tag is set to signal state "1", the current count value is set to the value of the "STOCK" tag. If the current count value is greater than or equal to the value of the "STOCK" tag, the "STOCK_PACKAGES" tag supplies the signal state "1".



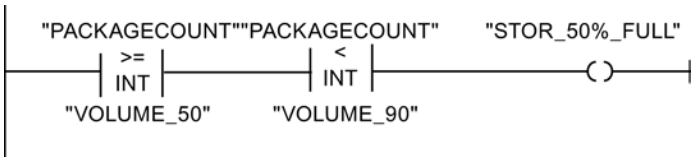
Network 2:

As long as there are packages in the storage area, the "STOR_NOT_EMPTY" tag is set to signal state "1", and the "Storage area not empty" lamp is switched on.



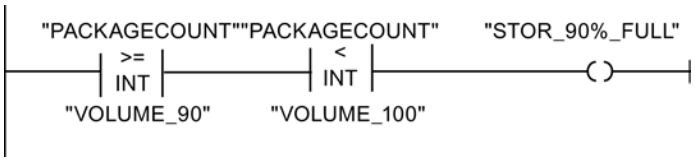
Network 3:

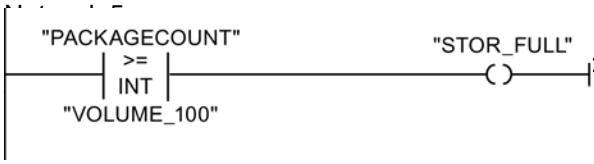
If the number of packages in the storage area is greater than or equal to 50, the lamp for the•"Storage area 50% full" message switches on.



Network 4:

If the number of packages in the storage area is greater than or equal to 90, the "Storage area 90% full" lamp switches on.





ches 100, the lamp for the "Storage area

See also

---( )---: Assignment (Page 3159)

CTUD: Count up and down (Page 3226)

CMP >=: Greater or equal (Page 3243)

CMP <: Less than (Page 3249)

---| |---: Normally open contact (Page 3156)

---| / |---: Normally closed contact (Page 3157)

Example of controlling room temperature

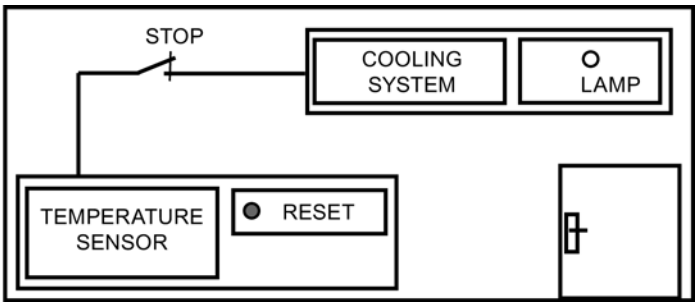
Controlling room temperature

In a cold room, the temperature must be maintained below zero degrees Celsius. Any temperature fluctuations are monitored by a sensor. If the temperature rises above zero degrees Celsius, the cooling system switches on for a preset time. The "Cooling system On" lamp is lit during this time.

The cooling system and the lamp are turned off if one of the following conditions is met:

- The sensor reports a temperature fall below zero degrees Celsius.
- The preset cooling time has elapsed.
- The pushbutton switch "STOP" has been pressed.

If the preset cooling time has expired and the temperature in the cold room is still too high, the cooling system can be restarted by means of the pushbutton switch "RESET".



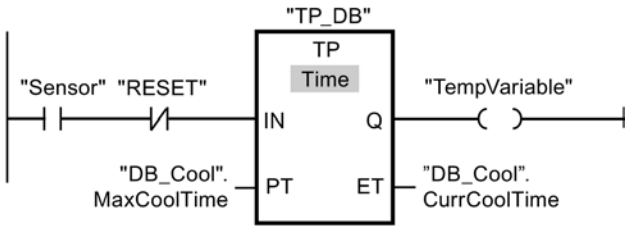
Implementation

The following table shows the definition of the tags used:

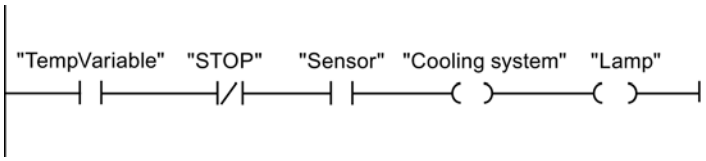
Name	Data type	Comment
Sensor	BOOL	Temperature sensor signal
RESET	BOOL	Restart
STOP	BOOL	The cooling system is switched off.
MaxCoolTime	TIME	Preset cooling time This tag is defined in the "DB_Cool" data block.
CurrCoolTime	TIME	Currently elapsed cooling time This tag is defined in the "DB_Cool" data block.
Cooling system	BOOL	The cooling system is switched on.
Lamp	BOOL	The lamp for the "Cooling system on" message is switched on.
TempVariable	BOOL	Temporary tag This tag stores the signal state of the IEC time TP.

The following network shows the LAD programming for controlling room temperature:

Network 1:



Network 2:



When the temperature in the cold room rises above zero degrees Celsius, the signal state at the "Sensor" operand switches from "0" to "1" (positive signal edge). In the case of a positive signal edge at the input IN of the time function, the preset cooling time is started and the "TempVariable" receives the signal state "1". The signal state "1" of the "TempVariable" has the result in network 2 that the cooling system as well as the display lamp are turned on. The outputs "Sensor", "Cooling system" and "Lamp" must be programmed in network 2, because you can program only one coil at output Q of the time function.

If the temperature in the cold room falls below zero degrees Celsius, the signal state of the sensor switches back to "0". This switches the cooling system and lamp off.

If the sensor does not signal a temperature drop, the cooling system and lamp are switched off after the preset cooling time has elapsed, at the latest. In this case, the cooling process can be restarted by pressing the "RESET" pushbutton switch. Pressing and releasing the pushbutton switch generates a new positive signal edge at input IN, which restarts the cooling system.

Using the pushbutton switch "STOP", the cooling system and the display lamp can be turned off at any time.

See also

---( )---: Assignment (Page 3159)

TP: Generate pulse (Page 3177)

---| |---: Normally open contact (Page 3156)

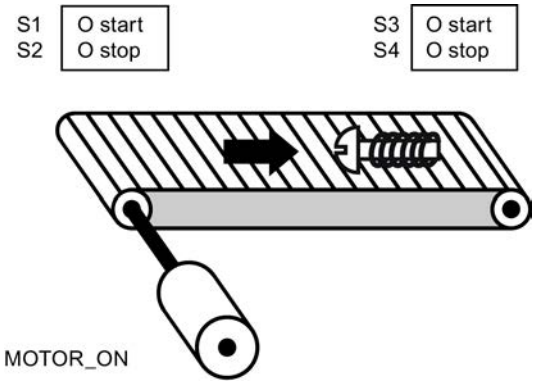
9.7.1.2 FBD programming examples

Example of controlling a conveyor belt

Controlling a conveyor belt

The following figure shows a conveyor belt that can be activated electrically. There are two pushbutton switches at the beginning of the conveyor belt: S1 for START and S2 for STOP.

There are also two pushbutton switches at the end of the conveyor belt: S3 for START and S4 for STOP. It is possible to start and stop the conveyor belt from either end.



Implementation

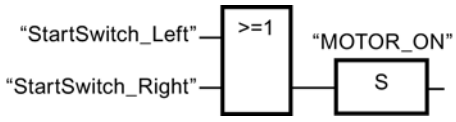
The following table shows the definition of the tags used:

Name	Data type	Description
StartSwitch_Left (S1)	BOOL	Start switch on the left side of the conveyor belt
StopSwitch_Left (S2)	BOOL	Stop switch on the left side of the conveyor belt
StartSwitch_Right (S3)	BOOL	Start switch on the right side of the conveyor belt
StopSwitch_Right (S4)	BOOL	Stop switch on the right side of the conveyor belt
MOTOR_ON	BOOL	Turn on the conveyor belt motor

The following networks show the FBD programming for solving this task:

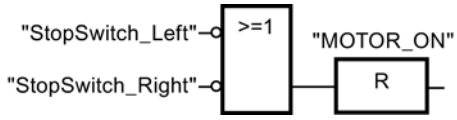
Network 1:

The conveyor belt motor is switched on when Start switch "S1" or "S3" is pressed.



Network 2:

The conveyor belt motor is switched off when stop switch "S2" or "S4" is pressed.



See also

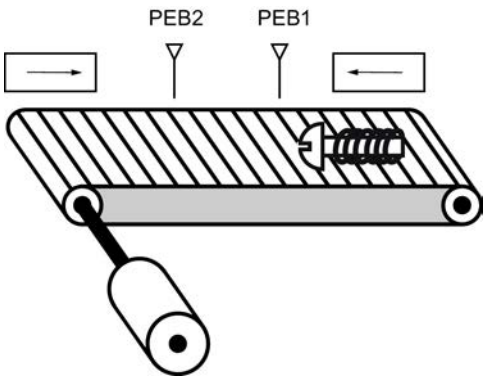
>=1: OR logic operation (Page 3447)

R: Reset output (Page 3454)

Example of detecting the direction of a conveyor belt

Detecting the direction of a conveyor belt

The following figure shows a conveyor belt that is equipped with two photoelectric barriers (PEB1 and PEB2). The photoelectric barriers are designed to detect the direction in which an object is moving on the conveyor belt.



Implementation

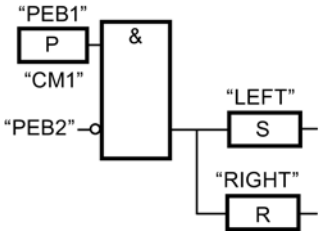
The following table shows the definition of the tags used:

Name	Data type	Description
PEB1	BOOL	Photoelectric barrier 1
PEB2	BOOL	Photoelectric barrier 2
RIGHT	BOOL	Display during movement to right
LEFT	BOOL	Display during movement to left
CM1	BOOL	Edge bit memory 1
CM2	BOOL	Edge bit memory 2

The following networks show the FBD programming for solving this task:

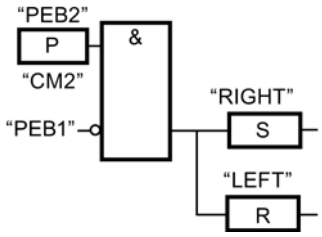
Network 1:

If the signal state changes from "0" to "1" (positive edge) at photoelectric barrier "PEB1" and, at the same time, the signal state at "PEB2" is "0", the object on the belt is moving to the left.



Network 2:

If the signal changes from "0" to "1" (positive edge) at photoelectric barrier "PEB2" and, at the same time, the signal state at "PEB1" is "0", the object on the belt is moving to the right.



See also

&: AND logic operation (Page 3445)

R: Reset output (Page 3454)

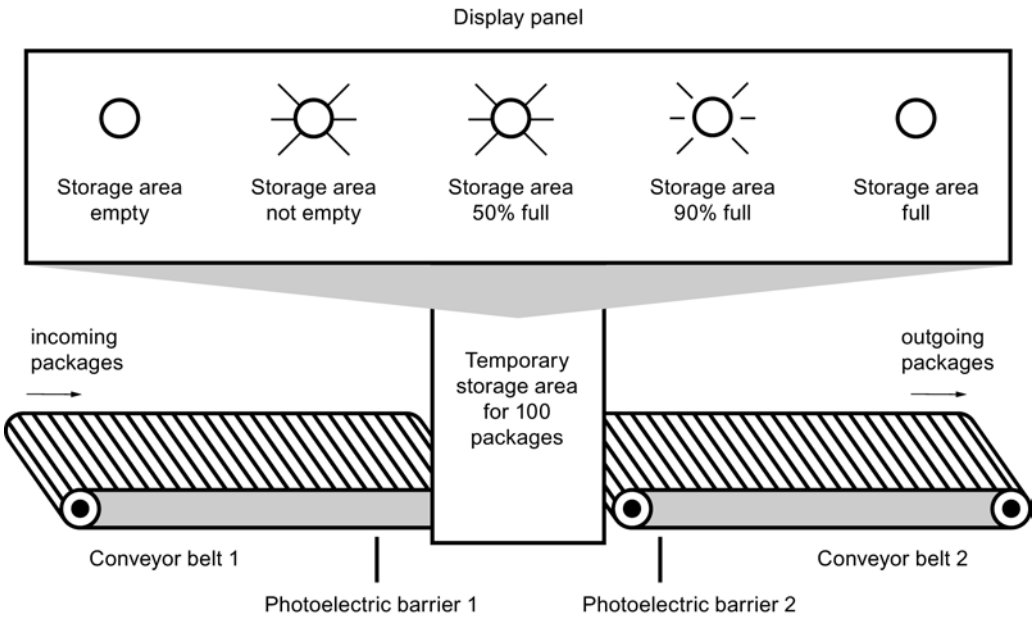
P: Scan operand for positive signal edge (Page 3461)

Example of detecting the fill level of a storage area

Detecting the fill level of a storage area

The following figure shows a system with two conveyor belts and a temporary storage area between them. Conveyor belt 1 delivers packages to the storage area. A photoelectric barrier at the end of conveyor belt 1 near the storage area detects how many packages are delivered to the storage area. Conveyor belt 2 transports packages from the temporary storage area to a loading dock onto which the packages are loaded for delivery to customers by truck. A photoelectric barrier at the storage area exit detects how many packages leave the storage area to be transported to the loading dock. Five display lamps indicate the capacity of the temporary storage area.

When a conveyor belt is restarted, the current count value is set to the number of packages available in the storage area.



Implementation

The following table shows the definition of the tags used:

Name	Data type	Description
PEB1	BOOL	Photoelectric barrier 1
PEB2	BOOL	Photoelectric barrier 2
RESET	BOOL	Reset counter
LOAD	BOOL	Set counter to value of "CV" parameter
STOCK	INT	Stock at restart
PACKAGECOUNT	INT	Number of packages in the storage area (current count value)
STOCK_PACKAGES	BOOL	Is set if the current count value is greater than or equal to the value of the tag "STOCK".
STOR_EMPTY	BOOL	Display lamp: Storage area empty
STOR_NOT_EMPTY	BOOL	Display lamp: Storage area not empty
STOR_50%_FULL	BOOL	Display lamp: Storage area 50% full
STOR_90%_FULL	BOOL	Display lamp: Storage area 90% full
STOR_FULL	BOOL	Display lamp: Storage area full
VOLUME_50	INT	Comparison value: 50 packages

Name	Data type	Description
VOLUME_90	INT	Comparison value: 90 packages
VOLUME_100	INT	Comparison value: 100 packages

The following networks show the FBD programming for activating the lamps:

Network 1:

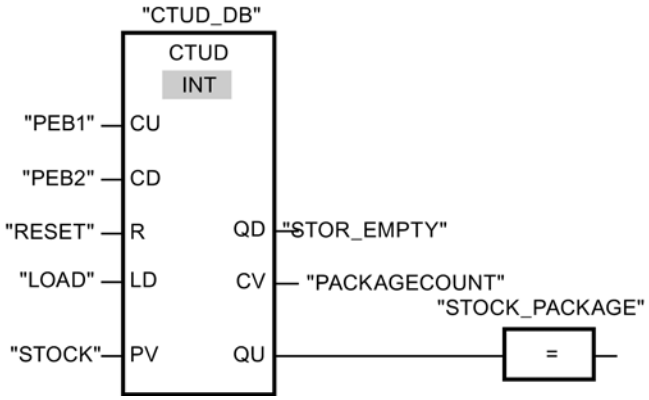
When a package is delivered to the storage area, the signal state at "PEB1" switches from "0" to "1" (positive signal edge). On a positive signal edge at "PEB1", the "Up" counter is enabled, and the current count value of "PACKAGECOUNT" is increased by one.

When a package is delivered from the storage area to the loading dock, the signal state at "PEB2" switches from "0" to "1" (positive signal edge). On a positive signal edge at "PEB2", the "Down" counter is enabled, and the current count value of "PACKAGECOUNT" is decreased by one.

If there are no packages in the storage area ("PACKAGECOUNT" = "0"), the "STOR_EMPTY" tag is set to signal state "1", and the "Storage area empty" lamp is switched on.

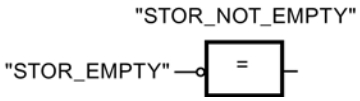
The current count value can be reset to "0" if the "RESET" tag is set to signal state "1".

If the "LOAD" tag is set to signal state "1", the current count value is set to the value of the "STOCK" tag. If the current count value is greater than or equal to the value of the "STOCK" tag, the "STOCK_PACKAGES" tag supplies the signal state "1".



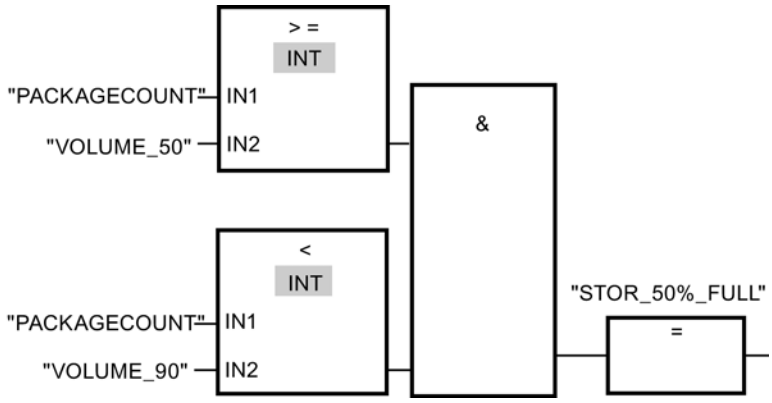
Network 2:

As long as there are packages in the storage area, the "STOR_NOT_EMPTY" tag is set to signal state "1", and the "Storage area not empty" lamp is switched on.



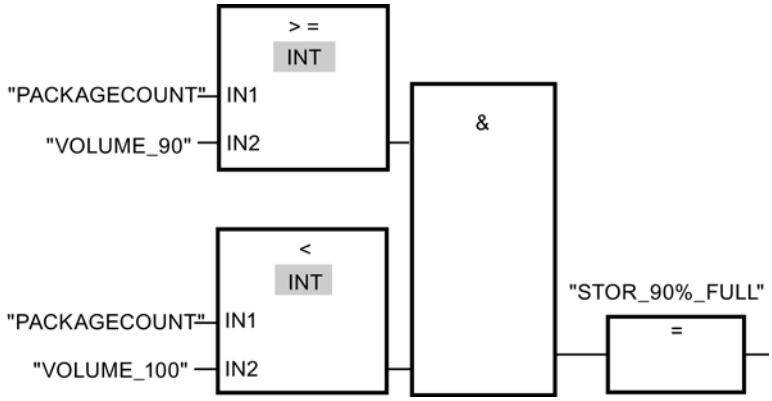
Network 3:

If the number of packages in the storage area is greater than or equal to 50, the lamp for the "Storage area 50% full" message switches on.



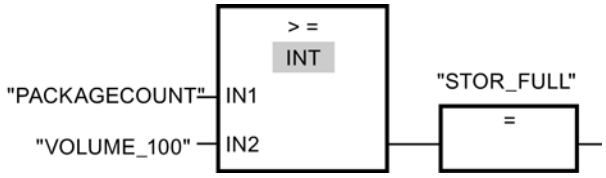
Network 4:

If the number of packages in the storage area is greater than or equal to 90, the "Storage area 90% full" lamp switches on.



Network 5:

If the number of packages in the storage area reaches 100, the lamp for the "Storage area full" message switches on.



See also

CTUD: Count up and down (Page 3226)

=: Assignment (Page 3452)

CTUD: Count up and down (Page 3523)

CMP >=: Greater or equal (Page 3540)

CMP <: Less than (Page 3545)

Example of controlling room temperature

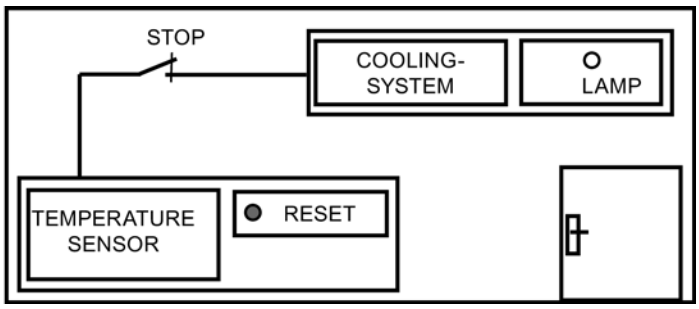
Controlling room temperature

In a cold room, the temperature must be maintained below zero degrees Celsius. Any temperature fluctuations are monitored by a sensor. If the temperature rises above zero degrees Celsius, the cooling system switches on for a preset time. The "Cooling system on" lamp is lit during this time.

The cooling system and the lamp are turned off if one of the following conditions is met:

- The sensor reports a temperature fall below zero degrees Celsius.
- The preset cooling time has elapsed.
- The pushbutton switch "Stop" has been pressed.

If the preset cooling time has expired and the temperature in the cold room is still too high, the cooling system can be restarted by means of the pushbutton switch "RESET".



Implementation

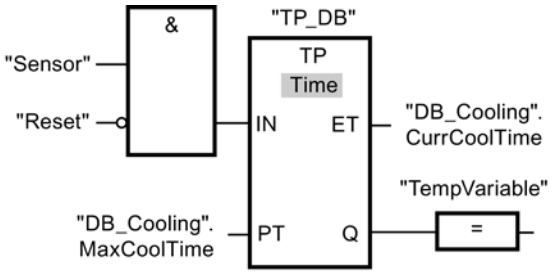
The following table shows the definition of the tags used:

Name	Data type	Comment
Sensor	BOOL	Temperature sensor signal
RESET	BOOL	Restart
STOP	BOOL	The cooling system is switched off.
MaxCoolTime	TIME	Preset cooling time This tag is defined in the "DB_Cool" data block.
CurrCoolTime	TIME	Currently elapsed cooling time This tag is defined in the "DB_Cool" data block.
Cooling system	BOOL	The cooling system is switched on.

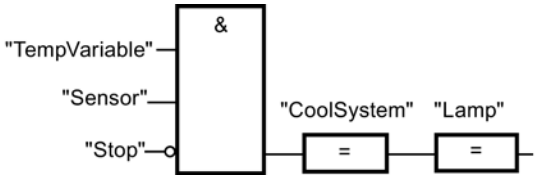
Name	Data type	Comment
Lamp	BOOL	The lamp for the "Cooling system on" message is switched on.
TempVariable	BOOL	Temporary tag This tag stores the signal state of the IEC time TP.

The following network shows the FBD programming for controlling room temperature:

Network 1:



Network 2:



When the temperature in the cold room rises above zero degrees Celsius, the signal state at the "Sensor" operand switches from "0" to "1" (positive signal edge). In the case of a positive signal edge at the input IN of the time function, the preset cooling time is started and the "TempVariable" receives the signal state "1". The signal state "1" of the "TempVariable" has the result in network 2 that the cooling system as well as the display lamp are turned on. The outputs "Sensor", "Cooling system" and "Lamp" must be programmed in network 2, because you can program only one coil at output Q of the time function.

If the temperature in the cold room falls below zero degrees Celsius, the signal state of the sensor switches back to "0". This switches the cooling system and lamp off.

If the sensor does not signal a temperature drop, the cooling system and lamp are switched off after the preset cooling time has elapsed, at the latest. In this case, the cooling process can be restarted by pressing the pushbutton switch "RESET". Pressing and releasing the pushbutton switch generates a new positive signal edge at input IN, which restarts the cooling system.

Using the pushbutton switch "STOP", the cooling system and the display lamp can be turned off at any time.

See also

TP: Generate pulse (Page 3472)

=: Assignment (Page 3452)

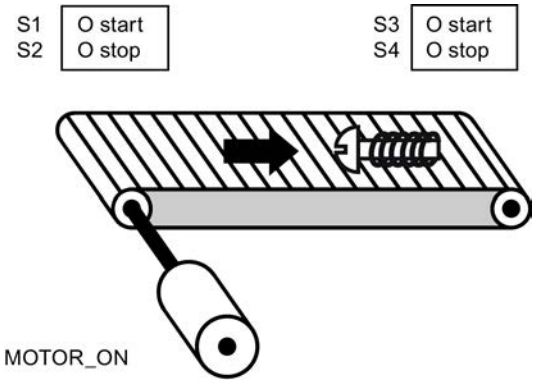
&: AND logic operation (Page 3445)

9.7.1.3 STL programming examples

Example: Bit logic instructions

Controlling a conveyor belt

The following figure shows a conveyor belt that can be activated electrically. There are two pushbutton switches at the beginning of the conveyor belt: S1 for START and S2 for STOP. There are also two pushbutton switches at the end of the conveyor belt: S3 for START and S4 for STOP. It is possible to start and stop the conveyor belt from either end.



Implementation

The following table shows the definition of the tags used:

Operand	Declaration	Data type	Description
StartSwitch_Left (S1)	Input	BOOL	Start switch on the left side of the conveyor belt
StopSwitch_Left (S2)	Input	BOOL	Stop switch on the left side of the conveyor belt
StartSwitch_Right (S3)	Input	BOOL	Start switch on the right side of the conveyor belt
StopSwitch_Right (S4)	Input	BOOL	Stop switch on the right side of the conveyor belt
MOTOR_ON/OFF	Output	BOOL	Switching the conveyor belt motor on/off

The following STL program shows how to implement this task:

STL	Explanation
O #S1	// Scan start switch S1 for "1".
O #S3	// Scan start switch S3 for "1".
S #"MOTOR_ON/OFF"	// If one of the start switches (S1 or S3) returns the signal state "1", the conveyor belt motor will be turned on.
O #S2	// Scan stop switch S2 for "1".
O #S4	// Scan stop switch S4 for "1".
R #"MOTOR_ON/OFF"	// If one of the stop switches (S2 or S4) returns the signal state "1", the conveyor belt motor will be turned off.

See also

O: OR logic operation (Page 5906)

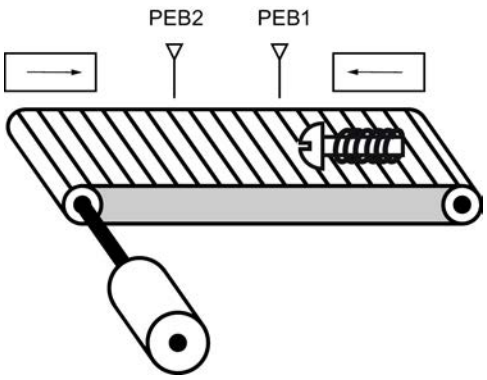
R: Reset (Page 5923)

S: Set (Page 5924)

Example of detecting the direction of a conveyor belt

Detecting the direction of a conveyor belt

The following figure shows a conveyor belt that is equipped with two photoelectric barriers (PEB1 and PEB2). The photoelectric barriers are designed to detect the direction in which an object is moving on the conveyor belt.



Implementation

The following table shows the definition of the tags used:

Name	Declaration	Data type	Description
S1	Input	BOOL	Photoelectric barrier 1
S2	Input	BOOL	Photoelectric barrier 2
TM1	Input	BOOL	Edge bit memory 1
TM2	Input	BOOL	Edge bit memory 2

Name	Declaration	Data type	Description
RIGHT	Output	BOOL	Display for movement to the right
LEFT	Output	BOOL	Display for movement to the left

The following STL program shows how to implement this example:

```
STL                 Explanation
A #S1               // Scan photoelectric barrier "S1" for "1"
FP #TM1             // Query positive edge
AN #S2              // Scan photoelectric barrier "S2" for "0"
S #LEFT             //If the signal state changes from "0" to "1" (positive edge)
                    at the photoelectric barrier "S1" and, at the same time, the
                    signal state at the photoelectric barrier "S2" is "0", the
                    object on the belt is moving to the left.
                    // The display for movement to the left is activated.
A #S2               // Scan photoelectric barrier "S2" for "1"
FP #TM2             // Query positive edge
AN #S1              // Scan photoelectric barrier "S1" for "0"
S #RIGHT            //If the signal state changes from "0" to "1" (positive edge)
                    at photoelectric barrier "S2" and, at the same time, the signal
                    state at photoelectric barrier "S1" is "0", the object on the
                    belt is moving to the right.
                    // The display for movement to the right is activated.
AN #S1              // Scan photoelectric barrier "S1" for "0"
AN #S2              // Scan photoelectric barrier "S2" for "0"
R #LEFT             // The display for movement to the left will be turned off when
                    the signal state at both photoelectric barriers is "0".
R #RIGHT            // The display for movement to the right will be turned off when
                    the signal state at both photoelectric barriers is "0".
```

See also

O: OR logic operation (Page 5906)

R: Reset (Page 5923)

S: Set (Page 5924)

A: AND logic operation (Page 5903)

AN: Negated AND logic operation (Page 5905)

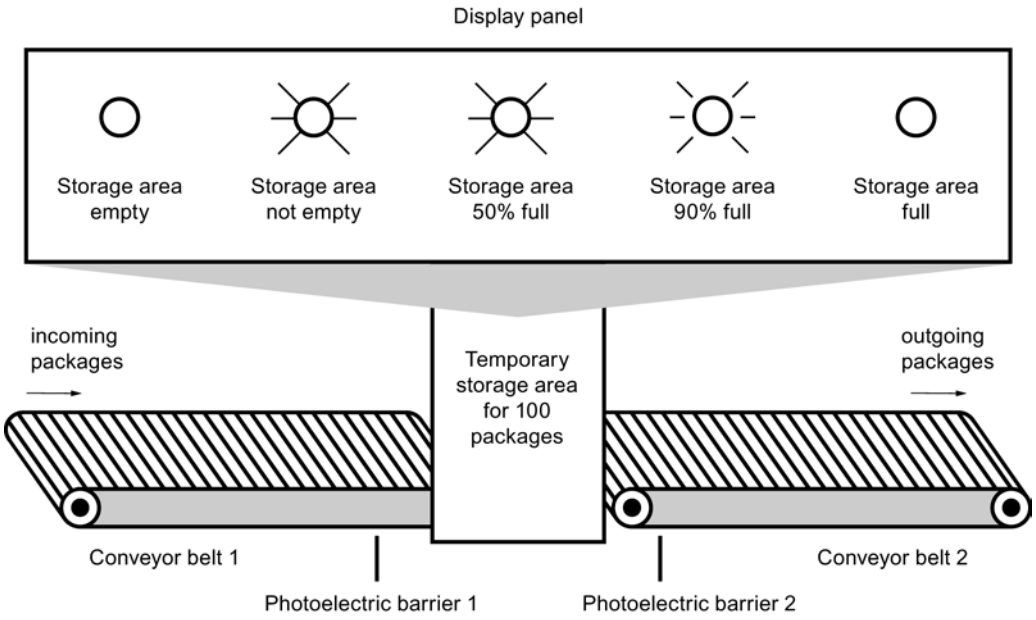
FP: Scan RLO for positive signal edge (Page 5930)

Example of detecting the fill level of a storage area

Detecting the fill level of a storage area

The following figure shows a system with two conveyor belts and a temporary storage area between them. Conveyor belt 1 delivers packages to the storage area. A photoelectric barrier at the end of conveyor belt 1 near the storage area detects how many packages are delivered to the storage area. Conveyor belt 2 transports packages from the temporary storage area to a loading dock onto which the packages are loaded for delivery to customers by truck. A photoelectric barrier at the storage area exit detects how many packages leave the storage area to be transported to the loading dock. Five display lamps indicate the capacity of the temporary storage area.

When a conveyor belt is restarted, the current count value is set to the number of packages available in the storage area.



Implementation

The following table shows the definition of the tags used:

Name	Data type	Address	Description
PACKAGECOUNT	COUNTER	C1	Number of packages in the storage area (current count value)

Name	Section	Data type	Description
LS1	Input	BOOL	Photoelectric barrier 1
LS2	Input	BOOL	Photoelectric barrier 2

Name	Section	Data type	Description
STOR_EMPTY	Output	BOOL	Display lamp: Storage area empty
STOR_NOT_EMPTY	Output	BOOL	Display lamp: Storage area not empty
STOR_50%_FULL	Output	BOOL	Display lamp: Storage area 50% full
STOR_90%_FULL	Output	BOOL	Display lamp: Storage area 90% full
STOR_FULL	Output	BOOL	Display lamp: Storage area full

The following STL program shows how to implement this example:

```
STL                 Explanation
A #LS1              // Scan photoelectric barrier "LS1" for "1".
CU "PACKAGECOUNT"   // At a positive edge at photoelectric barrier "LS1", the count
                    value of counter "PACKAGECOUNT" is increased by one.
A #LS2              // Scan photoelectric barrier "LS2" for "1".
CD "PACKAGECOUNT"   // At a positive edge at photoelectric barrier "LS2", the count
                    value of counter "PACKAGECOUNT" is decreased by one.
AN "PACKAGECOUNT"   // Scan count value for "0".
= #STOR_EMPTY       // With a count value of "0" the display lamp "storage area
                    empty" is switched on.
A "PACKAGECOUNT"    // Scan count value for "1".
= #STOR_NOT_EMPTY   // With a count value greater than "0" the display lamp "Storage
                    area not empty" is switched on.
L 50                // Load the comparison value "50" to accumulator 1.
L "PACKAGECOUNT"    // Move the comparison value to accumulator 2.
                    // Load the current count value to accumulator 1.
<=I                 // Compare values
= #"STOR_50%_FULL"  // With a count value greater than or equal to "50" the display
                    lamp "Storage area 50% full" is switched on.
L 90                // Move the counter value to accumulator 2.
                    // Load the comparison value "90" to accumulator 1.
>=I                 // Compare values
= #"STOR_90%_FULL"  // With a count value greater than or equal to "90" the display
                    lamp "Storage area 90% full" is switched on.
L "PACKAGECOUNT"    // Load the current count value to accumulator 1.
L 100               // Move the counter value to accumulator 2.
                    // Load the comparison value "100" to accumulator 1.
>=I                 // Compare values
= #STOR_FULL        // At a count value greater than "100" the display lamp "Storage
                    area full" is switched on.
```

See also

O: OR logic operation (Page 5906)

R: Reset (Page 5923)

S: Set (Page 5924)

A: AND logic operation (Page 5903)

AN: Negated AND logic operation (Page 5905)

FP: Scan RLO for positive signal edge (Page 5930)

=: Assignment (Page 5921)

CU: Count up (Page 5956)

CD: Count down (Page 5957)

L: Load (Page 5994)

Example of calculating an equation

Calculating an equation

The sample program shows you how to use three math instructions to calculate the following equation:

RESULT = ((A + B) x C) /D

Implementation

The following table shows the declaration of the operands used in the PLC tag table:

Name	Data type	Comment
A	INT	First value for addition
B	INT	Second value for addition
C	INT	Multiplier
D	INT	Divisor
RESULT	INT	End result

The following STL program shows how to implement this example:

```
 STL                 Explanation
L "A"                // Load value of the operand "A" to accumulator 1
L "B"                // Load value of operand "A" to accumulator 2
                     // Load value of the operand "B" to accumulator 1
+I                   // Add the values of accumulator 1 and 2
                     // Store sum in accumulator 1
L "C"                // Move the sum to accumulator 2
                     // Load value of the operand "C" to accumulator 1
*I                   // Multiply the values of accumulator 1 and 2
```

STL	Explanation
	// Store product in accumulator 1
L "D"	// Move the product to accumulator 2
	// Load value of the operand "D" to accumulator 1
/I	// Divide the value of accumulator 2 by the value of accumulator 1
	// Store result in accumulator 1
T "RESULT"	// Transfer the result to operand "RESULT"

See also

+I: Add integers (16 bit) (Page 5964)

-I: Subtract integers (16 bit) (Page 5965)

*I: Multiply integers (16 bit) (Page 5966)

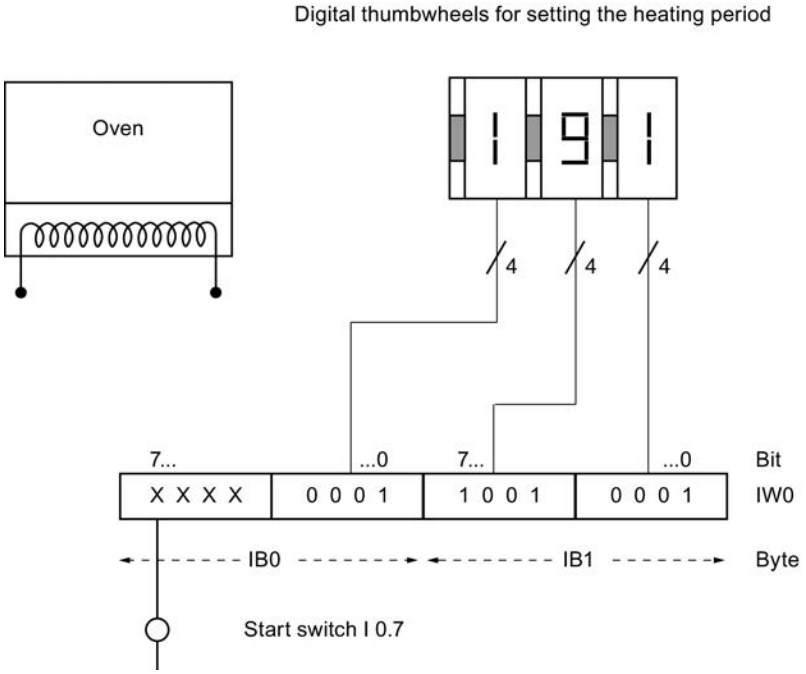
L: Load (Page 5994)

T: Transfer (Page 6003)

Example: Word logic instructions

Heating an oven

The following illustration shows an oven that is turned on with a start switch. The heating process is started when the start switch is pressed. The heating period is set with digital thumbwheels. The heating period is set in seconds using the BCD format.



Implementation

The following table shows the declaration of the operands used in the PLC tag table:

Name	Data type	Address	Comment
DURATION	WORD	EW0	Heating period in seconds • I1.0 to I1.3: Thumbwheels for ones • I1.4 to I1.7: Thumbwheels for tens • I0.0 to I0.3: Thumbwheels for hundreds
HEATING	TIMER	T1	Time that is started with the preset heating period.

The following table shows the declaration of the operands used in the block interface of the code block:

Name	Section	Data type	Comment
START	Input	BOOL	Start switch
START_HEATING	Output	BOOL	Start of the heating process

The following STL program shows how to implement this example:

```
STL                 Explanation
A "HEATING"         // Scan to see if time has started
= #START_HEATING    // Start the heating process
BEC                 // For RLO=1 End processing of block
                    // This prevents the "HEATING" time from being restarted if the
                    button is pressed
L "DURATION"        // Load the heating period in accumulator 1
AW W#16#0FFF        // Reset input bits I0.4 to I0.7 to "0"
OW W#16#4000        // Set seconds in bits I1.2 and I1.3 of accumulator 1
A #START            // Scan start switch for "1"
SE "HEATING"        // Start heating process as an extended pulse with a positive
                    edge at the start switch
```

See also

L: Load (Page 5994)

SE: Start extended pulse timer (Page 5941)

A: AND logic operation (Page 5903)

=: Assignment (Page 5921)

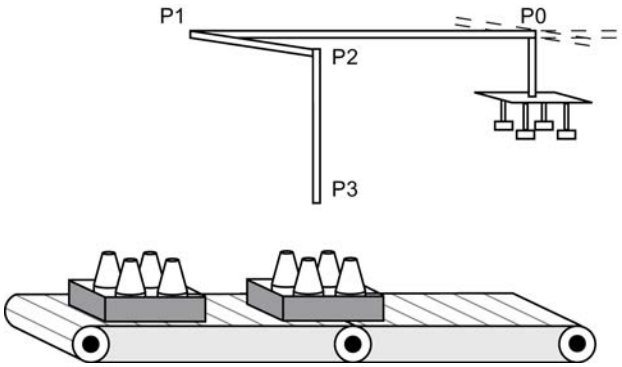
AW: AND logic operation word by word (Page 6074)

OW: OR logic operation word by word (Page 6076)

Example of a step sequence

Programming a step sequence

The following figure shows a station for removing glass containers from a pallet. The pallets are transported on a conveyor belt to the station. When a pallet with glass containers reaches the station, the conveyor belt will be stopped and a gripper moves from its basic position (P0) to the position above the pallet (P2). Once the gripper is above the pallet, the gripping clamps will open and the gripper lowered. Sensors detect the actual position of the gripper and the status of the gripping clamps. The sequence of the gripper movement in this example is implemented by a step sequence. You can program the additional steps required for removing the bottles and transporting them on an additional conveyor belt.



Implementation

The following table shows the declaration of the operands used in the PLC tag table:

Name	Data type	Comment
NUMBER	INT	Step number
Tag_Error	BOOL	Operand that is set when the step number is greater than 3 or if one of the steps was not executed.

The following table shows the declaration of the operands used in the block interface of the code block:

Name	Section	Data type	Comment
POS_0	Input	BOOL	Gripper in basic position (P0)
POS_1	Input	BOOL	Gripper in position 1 (P1)
POS_2	Input	BOOL	Gripper in position 2 (P2)
GRIPPER_OPEN	Input	BOOL	Gripping clamps open
OUT_POS_1	Output	BOOL	Move gripper to position 1
OUT_POS_2	Output	BOOL	Move gripper to position 2
OUT_GRIPPER	Output	BOOL	Open gripping clamps
OUT_POS_3	Output	BOOL	Move gripper to position 3

The following STL program shows how to implement this example:

```
STL                          Explanation
L "NUMBER"                   // Load the step number to accumulator 1.
JL END                       // Start of the jump list
JU POSITION_0                // At a value of "0" in the accumulator 1 jump to jump
                             label "POSITION_0".
JU POSITION_1                // At a value of "1" in the accumulator 1 jump to jump
                             label "POSITION_1".
JU POSITION_2                // At a value of "2" in the accumulator 1 jump to jump
                             label "POSITION_2".
JU POSITION_3                // At a value of "3" in the accumulator 1 jump to jump
                             label "POSITION_3".
END: JU ERROR                // End of the jump list
                             // At a step number greater than 3 jump to jump label
                             "ERROR".
POSITION_0: A #POS_0         // Jump label "POSITION_0"
                             // Scan to see if gripper is in basic position (P0).
= #OUT_POS_1                 // If condition is met, set output "OUT_POS_1" and
                             move gripper to position 1 (P1).
JCN ERROR                    // With RLO "0" jump to jump label "ERROR".
JC NEXT                      // With RLO "1" jump to jump label "NEXT".
POSITION_1: A #POS_1         // Jump label "POSITION_1"
                             // Scan to see if gripper is in position 1 (P1).
= #OUT_POS_2                 // If condition is met, set output "OUT_POS_2" and
                             move gripper to position 2 (P2).
JCN ERROR                    // With RLO "0" jump to jump label "ERROR".
JC NEXT                      // With RLO "1" jump to jump label "NEXT".
POSITION_2: A #POS_2         // Jump label "POSITION_2"
                             // Scan to see if gripper is in position 2 (P2).
= #OUT_GRIPPER               // If condition is met, set output "OUT_GRIPPER" and
                             open gripping clamps.
```

STL	Explanation
JCN ERROR	// With RLO "0" jump to jump label "ERROR".
JC NEXT	// With RLO "1" jump to jump label "NEXT".
POSITION_3: A #POS_2	// Jump label "POSITION_3"
	// Scan to see if gripper is in position 2 (P2).
A #GRIPPER_OPEN	// Scan to see if the gripping clamps are open
= #OUT_POS_3	// If condition is met, set output "OUT_POS_3" and move gripper to position 3 (P3).
JCN ERROR	// With RLO "0" jump to jump label "ERROR".
JC NEXT	// With RLO "1" jump to jump label "NEXT".
NEXT: INC 1	// Jump label "NEXT"
	// Increase step number in accumulator 1 by one.
T "NUMBER"	// Transfer step number to operand "NUMBER".
L 3	// Move the current step number to accumulator 2.
	// Load value 3 to accumulator 1.
>I	// Scan to see if current step number is greater than 3.
JC RESET_NUMBER	// With a scan result "1" jump to jump label "RESET_NUMBER" and continue with program processing
BEU	// End block
RESET_NUMBER: L 0	// Jump label "RESET_NUMBER"
	// Load value "0" in accumulator 1.
T "NUMBER"	// Assign value "0" to operands "NUMBER" (step number).
BEU	// End block
ERROR: NOT	// Jump label "ERROR"
= "Tag_Error"	// Assign negated RLO to the operand "Tag_Error".
BEU	// End block

See also

A: AND logic operation (Page 5903)

=: Assignment (Page 5921)

L: Load (Page 5994)

T: Transfer (Page 6003)

? I: Compare 16-bit integers (Page 5959)

JU: Unconditional jump (Page 6032)

JC: Jump if RLO = 1 (Page 6034)

JCN: Jump if RLO = 0 (Page 6035)

JL: Define jump list (Page 6057)

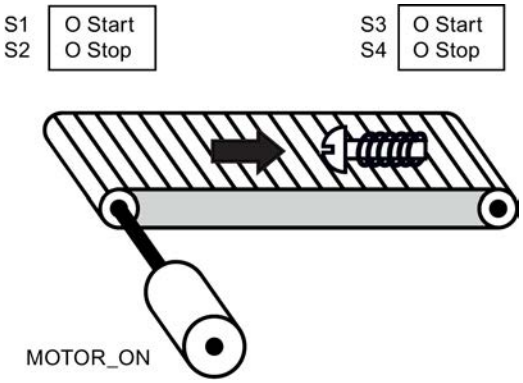
BEU: Unconditional block end (Page 6069)

9.7.1.4 SCL programming examples

Example: Bit logic instructions

Controlling a conveyor belt

The following figure shows a conveyor belt that can be activated electrically. There are two pushbutton switches at the beginning of the conveyor belt: S1 for START and S2 for STOP. There are also two pushbutton switches at the end of the conveyor belt: S3 for START and S4 for STOP. It is possible to start and stop the conveyor belt from either end.



Implementation

The following table shows the definition of the tags used:

Operand	Declaration	Data type	Description
StartSwitch_Left (S1)	Input	BOOL	Start switch on the left side of the conveyor belt
StopSwitch_Left (S2)	Input	BOOL	Stop switch on the left side of the conveyor belt
StartSwitch_Right (S3)	Input	BOOL	Start switch on the right side of the conveyor belt
StopSwitch_Right (S4)	Input	BOOL	Stop switch on the right side of the conveyor belt
MOTOR_ON	Output	BOOL	Turn on the conveyor belt motor
MOTOR_OFF	Output	BOOL	Turn off the conveyor belt motor

The following SCL program shows how to implement this task:

SCL

```
IF "StartSwitch_Left" OR "StartSwitch_Right" THEN
"MOTOR_ON" := 1;
END_IF;
```

SCL

```
IF "StopSwitch_Left" OR "StopSwitch_Right" THEN
"MOTOR_OFF" := 1;
END_IF;
```

The conveyor belt motor is switched on when start switch "StartSwitch_Left" or "StartSwitch_Right" is pressed. The conveyor belt motor is switched off when stop switch "StopSwitch_Left" or "StopSwitch_Right" is pressed.

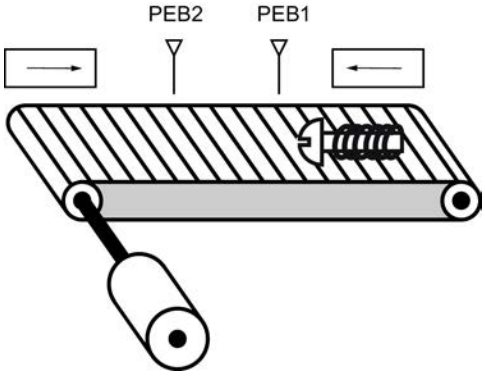
See also

Logical expressions (Page 2686)

Example of detecting the direction of a conveyor belt

Detecting the direction of a conveyor belt

The following figure shows a conveyor belt that is equipped with two photoelectric barriers (PEB1 and PEB2). The photoelectric barriers are designed to detect the direction in which an object is moving on the conveyor belt.



Implementation

The following table shows the definition of the tags used:

Name	Declaration	Data type	Description
Photoelectric barrier LS1	Input	BOOL	Photoelectric barrier 1
Photoelectric barrier LS2	Input	BOOL	Photoelectric barrier 2
RIGHT	Output	BOOL	Display for movement to the right
LEFT	Output	BOOL	Display for movement to the left

The following SCL program shows how to implement this example:

SCL

```
IF "LS1" = 1 AND NOT "LS2" = 0
THEN "LEFT" = 1;
IF "LS2" = 1 AND NOT "LS1" = 0
THEN "RIGHT" := 1;
IF "LS2" = 0
THEN "RIGHT" := 0;
IF "LS1" = 0
THEN "LEFT" = 0;
```

If the photoelectric barrier "PEB1" has signal state "1" and the photoelectric barrier "PEB2" has signal state "0" at the same time, the object on the belt is moving to the left. If the photoelectric barrier "PEB2" has signal state "1" and the photoelectric barrier "PEB1" has signal state "0" at the same time, the object on the belt is moving to the right. The displays for a movement to the left or right will be turned off when the signal state at both photoelectric barriers is "0".

See also

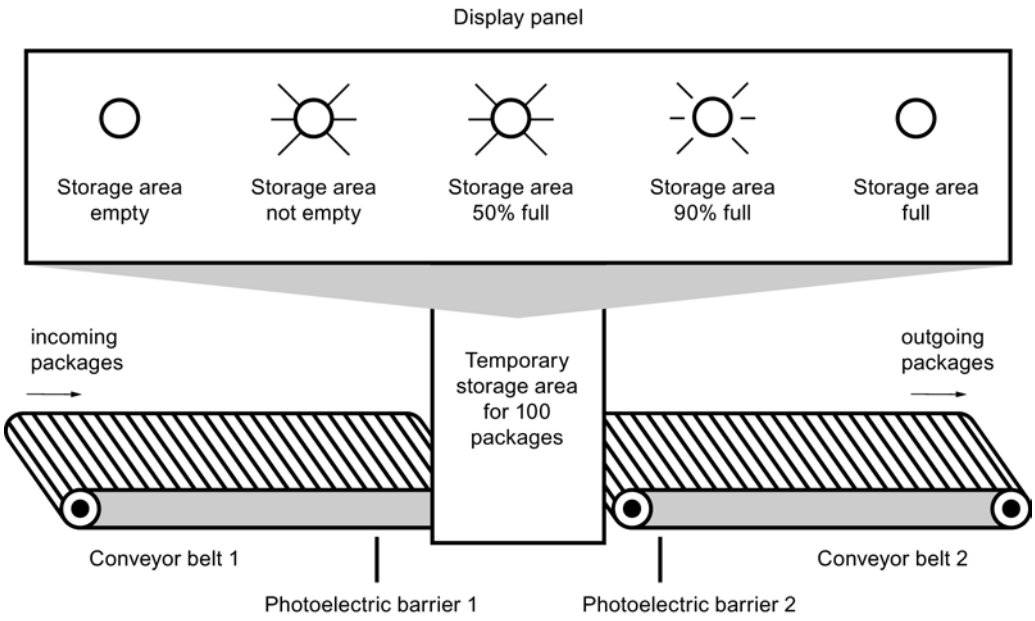
Logical expressions (Page 2686)

Example of detecting the fill level of a storage area

Detecting the fill level of a storage area

The following figure shows a system with two conveyor belts and a temporary storage area between them. Conveyor belt 1 delivers packages to the storage area. A photoelectric barrier at the end of conveyor belt 1 near the storage area detects how many packages are delivered to the storage area. Conveyor belt 2 transports packages from the temporary storage area to a loading dock onto which the packages are loaded for delivery to customers by truck. A photoelectric barrier at the storage area exit detects how many packages leave the storage area to be transported to the loading dock. Five display lamps indicate the capacity of the temporary storage area.

When a conveyor belt is restarted, the current count value is set to the number of packages available in the storage area.



Implementation

The following table shows the definition of the tags used:

Name	Declaration	Data type	Description
PEB1	Input	BOOL	Photoelectric barrier 1
PEB2	Input	BOOL	Photoelectric barrier 2
RESET	Input	BOOL	Reset counter
LOAD	Input	BOOL	Set counter to value of "CV" parameter
STOCK	Input	INT	Stock at restart
PACKAGECOUNT	Output	INT	Number of packages in the storage area (current count value)
STOCK_PACKAGES	Output	BOOL	Is set if the current count value is greater than or equal to the value of the tag "STOCK".
STOR_EMPTY	Output	BOOL	Display lamp: Storage area empty
STOR_NOT_EMPTY	Output	BOOL	Display lamp: Storage area not empty
STOR_50%_FULL	Output	BOOL	Display lamp: Storage area 50% full
STOR_90%_FULL	Output	BOOL	Display lamp: Storage area 90% full
STOR_FULL	Output	BOOL	Display lamp: Storage area full

Name	Declaration	Data type	Description
VOLUME_50	Input	INT	Comparison value: 50 packages
VOLUME_90	Input	INT	Comparison value: 90 packages
VOLUME_100	Input	INT	Comparison value: 100 packages

The following SCL program shows how to implement this example:

When a package is delivered to the storage area, the signal state at "PEB1" switches from "0" to "1" (positive signal edge). On a positive signal edge at "PEB1", the "Up" counter is enabled, and the current count value of "PACKAGECOUNT" is increased by one.

When a package is delivered from the storage area to the loading dock, the signal state at "PEB2" switches from "0" to "1" (positive signal edge). On a positive signal edge at "PEB2", the "Down" counter is enabled, and the current count value of "PACKAGECOUNT" is decreased by one.

If there are no packages in the storage area ("PACKAGECOUNT" = "0"), the "STOR_EMPTY" tag is set to signal state "1", and the "Storage area empty" lamp is switched on.

The current count value can be reset to "0" if the "RESET" tag is set to signal state "1".

If the "LOAD" tag is set to signal state "1", the current count value is set to the value of the "STOCK" tag. If the current count value is greater than or equal to the value of the "STOCK" tag, the "STOCK_PACKAGES" tag supplies the signal state "1".

SCL

```
"CTUD_DB".CTUD(CU := "PEB1",
               CD := "PEB2",
               R  := "RESET",
               LD := "LOAD",
               PV := "STOCK",
               QU := "STOCK_PACKAGES",
               QD := "STOR_EMPTY",
               CV := "PACKAGECOUNT");
```

As long as there are packages in the storage area, the "STOR_NOT_EMPTY" tag is set to signal state "1", and the "Storage area not empty" lamp is switched on.

SCL

```
"STOR_NOT_EMPTY" := NOT "STOR_EMPTY"
```

If the number of packages in the storage area is greater than or equal to 50, the lamp for the•"Storage area 50% full" message switches on.

SCL

```
IF "PACKAGECOUNT" >= "VOLUME_50" THEN "STOR_50%_FULL" := 1;
IF "PACKAGECOUNT" <= "VOLUME_90" THEN "STOR_50%_FULL" := 1;
```

If the number of packages in the storage area is greater than or equal to 90, the "Storage area 90% full" lamp switches on.

```
SCL
IF "PACKAGECOUNT" >= "VOLUME_90" THEN "STOR_90%_FULL" := 1;
IF "PACKAGECOUNT" <= "VOLUME_100" THEN "STOR_90%_FULL" := 1;
```

If the number of packages in the storage area reaches 100, the lamp for the "Storage area full" message switches on.

```
SCL
IF "PACKAGECOUNT" >= "VOLUME_100" THEN "STOR_FULL" := 1;
```

See also

Logical expressions (Page 2686)

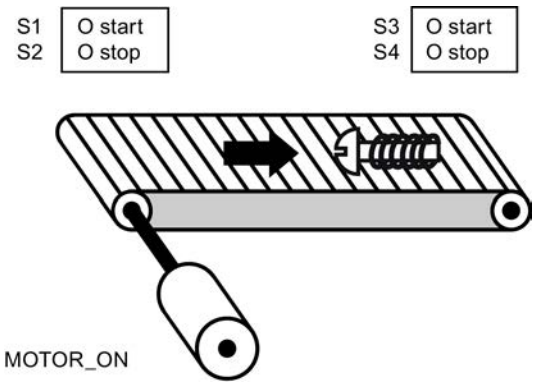
9.7.2 Programming examples (S7-300, S7-400)

9.7.2.1 LAD programming examples

Example of controlling a conveyor belt

Controlling a conveyor belt

The following figure shows a conveyor belt that can be activated electrically. There are two pushbutton switches at the beginning of the conveyor belt: S1 for START and S2 for STOP. There are also two pushbutton switches at the end of the conveyor belt: S3 for START and S4 for STOP. It is possible to start and stop the conveyor belt from either end.



Implementation

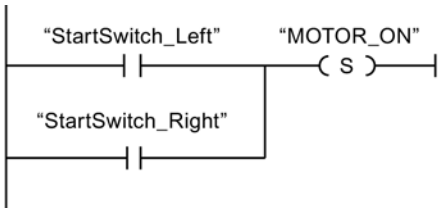
The following table shows the definition of the tags used:

Name	Data type	Description
StartSwitch_Left (S1)	BOOL	Start switch on the left side of the conveyor belt
StopSwitch_Left (S2)	BOOL	Stop switch on the left side of the conveyor belt
StartSwitch_Right (S3)	BOOL	Start switch on the right side of the conveyor belt
StopSwitch_Right (S4)	BOOL	Stop switch on the right side of the conveyor belt
MOTOR_ON	BOOL	Turn on the conveyor belt motor

The following networks show the LAD programming for solving this task:

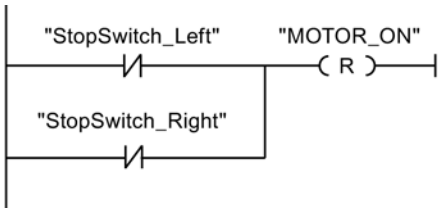
Network 1:

The conveyor belt motor is switched on when Start switch "S1" or "S3" is pressed.



Network 2:

The conveyor belt motor is switched off when Stop switch "S2" or "S4" is pressed.



See also

---(S)---: Set output (Page 5379)

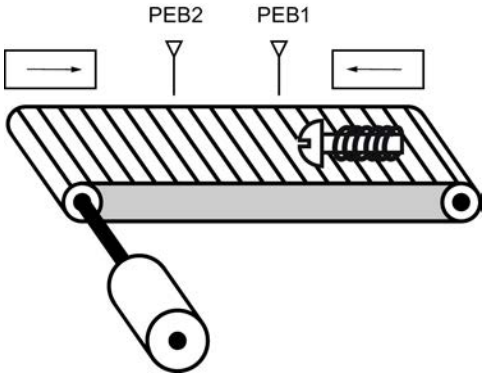
---| |---: Normally open contact (Page 5374)

---(R)---: Reset output (Page 5378)

Example of detecting the direction of a conveyor belt

Detecting the direction of a conveyor belt

The following figure shows a conveyor belt that is equipped with two photoelectric barriers (PEB1 and PEB2). The photoelectric barriers are designed to detect the direction in which an object is moving on the conveyor belt.



Implementation

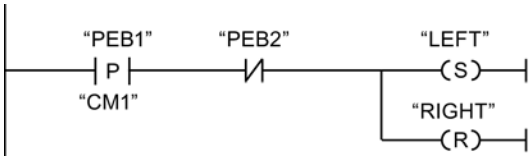
The following table shows the definition of the tags used:

Name	Data type	Description
PEB1	BOOL	Photoelectric barrier 1
PEB2	BOOL	Photoelectric barrier 2
RIGHT	BOOL	Display during movement to right
LEFT	BOOL	Display during movement to left
CM1	BOOL	Edge bit memory 1
CM2	BOOL	Edge bit memory 2

The following networks show the LAD programming for solving this task:

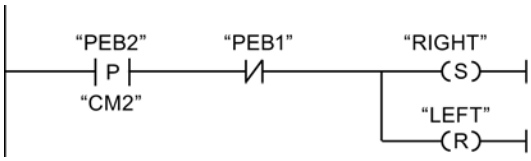
Network 1:

If the signal state changes from "0" to "1" (positive edge) at photoelectric barrier "PEB1" and, at the same time, the signal state at "PEB2" is "0", the object on the belt is moving to the left.



Network 2:

If the signal changes from "0" to "1" (positive edge) at photoelectric barrier "PEB2" and, at the same time, the signal state at "PEB1" is "0", the object on the belt is moving to the right.



See also

--|P|--: Scan operand for positive signal edge (Page 5383)

---(S)---: Set output (Page 5379)

---| |---: Normally open contact (Page 5374)

---(R)---: Reset output (Page 5378)

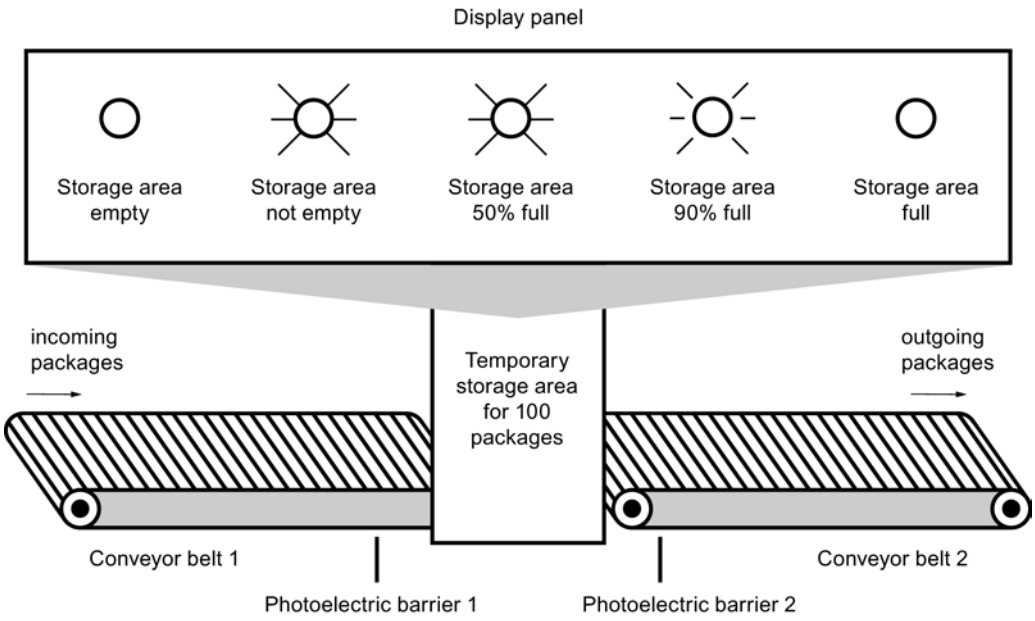
---| / |---: Normally closed contact (Page 5375)

Example of detecting the fill level of a storage area

Detecting the fill level of a storage area

The following figure shows a system with two conveyor belts and a temporary storage area between them. Conveyor belt 1 delivers packages to the storage area. A photoelectric barrier at the end of conveyor belt 1 near the storage area detects how many packages are delivered to the storage area. Conveyor belt 2 transports packages from the temporary storage area to a loading dock onto which the packages are loaded for delivery to customers by truck. A photoelectric barrier at the storage area exit detects how many packages leave the storage area to be transported to the loading dock. Five display lamps indicate the capacity of the temporary storage area.

When a conveyor belt is restarted, the current count value is set to the number of packages available in the storage area.



Implementation

The following table shows the definition of the tags used:

Name	Data type	Description
PEB1	BOOL	Photoelectric barrier 1
PEB2	BOOL	Photoelectric barrier 2
RESET	BOOL	Reset counter
LOAD	BOOL	Set counter to value of "PV" parameter
STOCK	INT	Stock at restart
PACKAGECOUNT	INT	Number of packages in the storage area (current count value)
STOCK_PACKAGES	BOOL	Is set if the current count value is greater than or equal to the value of the tag "STOCK".
STOR_EMPTY	BOOL	Display lamp: Storage area empty
STOR_NOT_EMPTY	BOOL	Display lamp: Storage area not empty
STOR_50%_FULL	BOOL	Display lamp: Storage area 50% full
STOR_90%_FULL	BOOL	Display lamp: Storage area 90% full
STOR_FULL	BOOL	Display lamp: Storage area full
VOLUME_50	INT	Comparison value: 50 packages

Name	Data type	Description
VOLUME_90	INT	Comparison value: 90 packages
VOLUME_100	INT	Comparison value: 100 packages

The following networks show the LAD programming for activating the lamps:

Network 1:

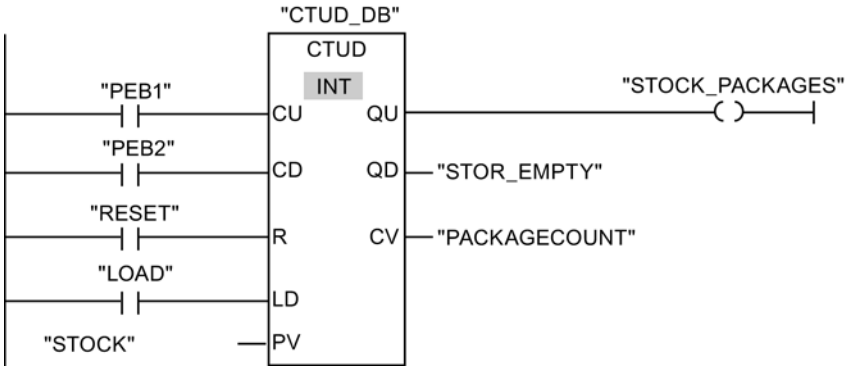
When a package is delivered to the storage area, the signal state at "PEB1" switches from "0" to "1" (positive signal edge). On a positive signal edge at "PEB1", the "Up" counter is enabled, and the current count value of "PACKAGECOUNT" is increased by one.

When a package is delivered from the storage area to the loading dock, the signal state at "PEB2" switches from "0" to "1" (positive signal edge). On a positive signal edge at "PEB2", the "Down" counter is enabled, and the current count value of "PACKAGECOUNT" is decreased by one.

If there are no packages in the storage area ("PACKAGECOUNT" = "0"), the "STOR_EMPTY" tag is set to signal state "1", and the "Storage area empty" lamp is switched on.

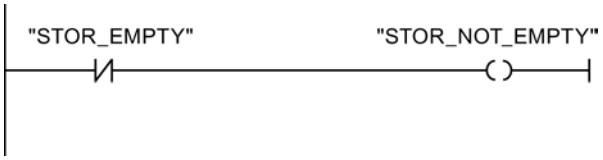
The current count value can be reset to "0" if the "RESET" tag is set to signal state "1".

If the "LOAD" tag is set to signal state "1", the current count value is set to the value of the "STOCK" tag. If the current count value is greater than or equal to the value of the "STOCK" tag, the "STOCK_PACKAGES" tag supplies the signal state "1".



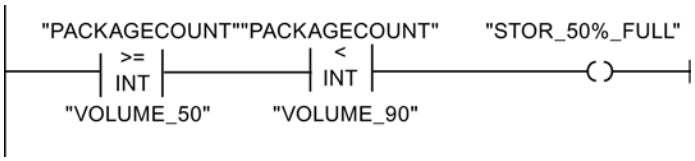
Network 2:

As long as there are packages in the storage area, the "STOR_NOT_EMPTY" tag is set to signal state "1", and the "Storage area not empty" lamp is switched on.



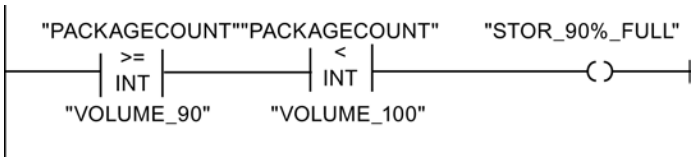
Network 3:

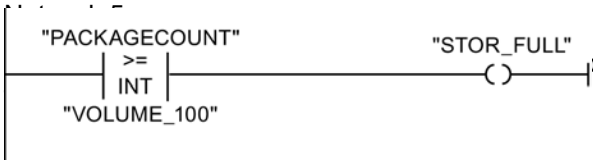
If the number of packages in the storage area is greater than or equal to 50, the lamp for the•"Storage area 50% full" message switches on.



Network 4:

If the number of packages in the storage area is greater than or equal to 90, the "Storage area 90% full" lamp switches on.





ches 100, the lamp for the "Storage area

See also

---( )---: Assignment (Page 5377)

CMP >=: Greater or equal (Page 5435)

CMP <: Less than (Page 5438)

Example of calculating an equation

Calculating an equation

The following program example shows how you can use three mathematic functions to produce the same result as the following equation:

RESULT = ((A + B) x 15) / E

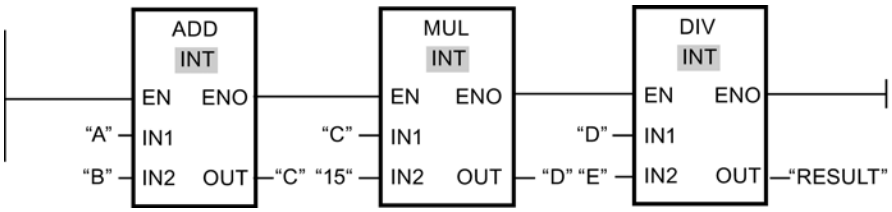
Implementation

The following table shows the definition of the tags used:

Name	Data type	Comment
A	INT	First value for addition
B	INT	Second value for addition
C	INT	First intermediate result
15	INT	Multiplier
D	INT	Second intermediate result

Name	Data type	Comment
E	INT	Divisor
RESULT	INT	End result

The following network shows the LAD programming for calculating the equation:



The value of operand "A" is added to the value of operand "B". The sum is stored in operand "C". The value of operand "C" is multiplied by "15". The multiplication result is stored in operand "D". The value stored in operand "D" is then divided by the value of operand "E". The end result is stored in the "RESULT" operand.

See also

ADD: Add (Page 5440)

MUL: Multiply (Page 5443)

DIV: Divide (Page 5444)

Example of controlling room temperature

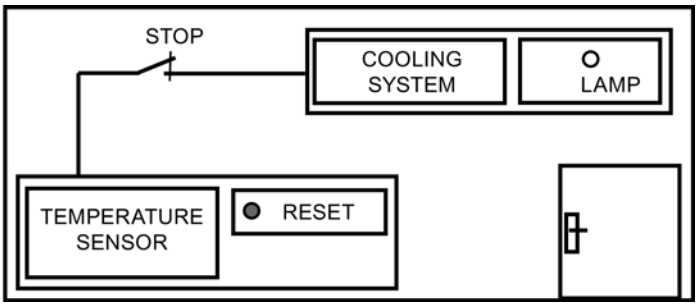
Controlling room temperature

In a cold room, the temperature must be maintained below zero degrees Celsius. Any temperature fluctuations are monitored by a sensor. If the temperature rises above zero degrees Celsius, the cooling system switches on for a preset time. The "Cooling system On" lamp is lit during this time.

The cooling system and the lamp are turned off if one of the following conditions is met:

- The sensor reports a temperature fall below zero degrees Celsius.
- The preset cooling time has elapsed.
- The pushbutton switch "STOP" has been pressed.

If the preset cooling time has expired and the temperature in the cold room is still too high, the cooling system can be restarted by means of the pushbutton switch "RESET".



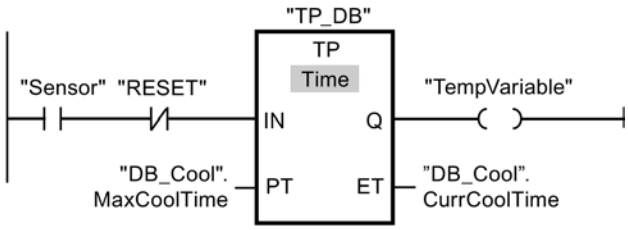
Implementation

The following table shows the definition of the tags used:

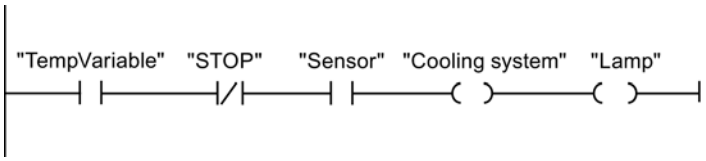
Name	Data type	Comment
Sensor	BOOL	Temperature sensor signal
RESET	BOOL	Restart
STOP	BOOL	The cooling system is switched off.
MaxCoolTime	TIME	Preset cooling time This tag is defined in the "DB_Cool" data block.
CurrCoolTime	TIME	Currently elapsed cooling time This tag is defined in the "DB_Cool" data block.
Cooling system	BOOL	The cooling system is switched on.
Lamp	BOOL	The lamp for the "Cooling system on" message is switched on.
TempVariable	BOOL	Temporary tag This tag stores the signal state of the IEC time TP.

The following network shows the LAD programming for controlling room temperature:

Network 1:



Network 2:



When the temperature in the cold room rises above zero degrees Celsius, the signal state at the "Sensor" operand switches from "0" to "1" (positive signal edge). In the case of a positive signal edge at the input IN of the time function, the preset cooling time is started and the "TempVariable" receives the signal state "1". The signal state "1" of the "TempVariable" has the result in network 2 that the cooling system as well as the display lamp are turned on. The outputs "Sensor", "Cooling system" and "Lamp" must be programmed in network 2, because you can program only one coil at output Q of the time function.

If the temperature in the cold room falls below zero degrees Celsius, the signal state of the sensor switches back to "0". This switches the cooling system and lamp off.

If the sensor does not signal a temperature drop, the cooling system and lamp are switched off after the preset cooling time has elapsed, at the latest. In this case, the cooling process can be restarted by pressing the "RESET" pushbutton switch. Pressing and releasing the pushbutton switch generates a new positive signal edge at input IN, which restarts the cooling system.

Using the pushbutton switch "STOP", the cooling system and the display lamp can be turned off at any time.

See also

TP: Generate pulse (Page 5389)

---( )---: Assignment (Page 5377)

---| |---: Normally open contact (Page 5374)

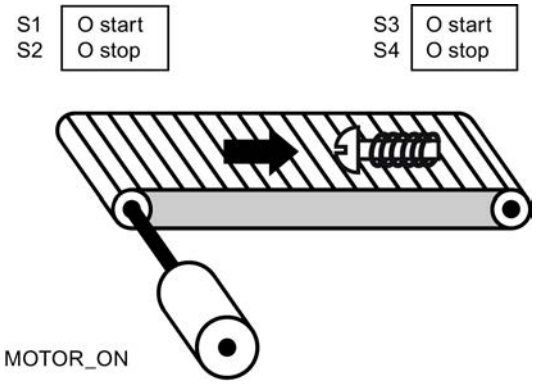
9.7.2.2 FBD programming examples

Example of controlling a conveyor belt

Controlling a conveyor belt

The following figure shows a conveyor belt that can be activated electrically. There are two pushbutton switches at the beginning of the conveyor belt: S1 for START and S2 for STOP.

There are also two pushbutton switches at the end of the conveyor belt: S3 for START and S4 for STOP. It is possible to start and stop the conveyor belt from either end.



Implementation

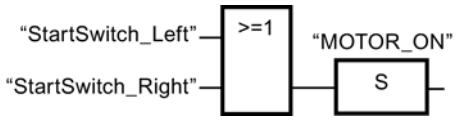
The following table shows the definition of the tags used:

Name	Data type	Description
StartSwitch_Left (S1)	BOOL	Start switch on the left side of the conveyor belt
StopSwitch_Left (S2)	BOOL	Stop switch on the left side of the conveyor belt
StartSwitch_Right (S3)	BOOL	Start switch on the right side of the conveyor belt
StopSwitch_Right (S4)	BOOL	Stop switch on the right side of the conveyor belt
MOTOR_ON	BOOL	Turn on the conveyor belt motor

The following networks show the FBD programming for solving this task:

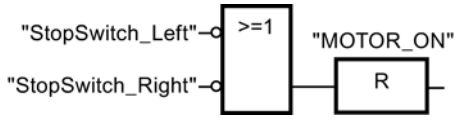
Network 1:

The conveyor belt motor is switched on when Start switch "S1" or "S3" is pressed.



Network 2:

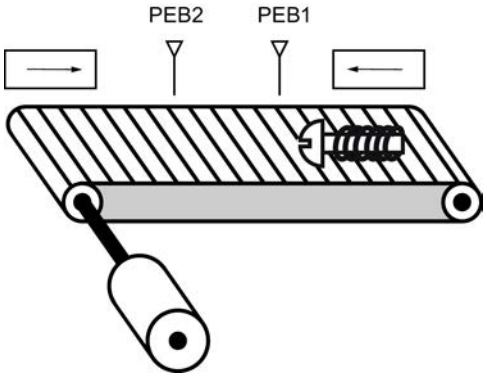
The conveyor belt motor is switched off when stop switch "S2" or "S4" is pressed.



Example of detecting the direction of a conveyor belt

Detecting the direction of a conveyor belt

The following figure shows a conveyor belt that is equipped with two photoelectric barriers (PEB1 and PEB2). The photoelectric barriers are designed to detect the direction in which an object is moving on the conveyor belt.



Implementation

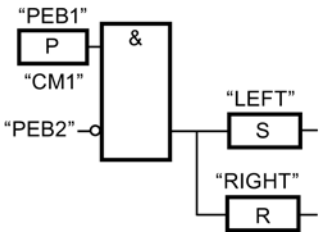
The following table shows the definition of the tags used:

Name	Data type	Description
PEB1	BOOL	Photoelectric barrier 1
PEB2	BOOL	Photoelectric barrier 2
RIGHT	BOOL	Display during movement to right
LEFT	BOOL	Display during movement to left
CM1	BOOL	Edge bit memory 1
CM2	BOOL	Edge bit memory 2

The following networks show the FBD programming for solving this task:

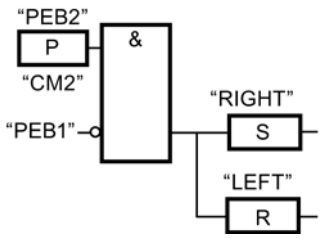
Network 1:

If the signal state changes from "0" to "1" (positive edge) at photoelectric barrier "PEB1" and, at the same time, the signal state at "PEB2" is "0", the object on the belt is moving to the left.



Network 2:

If the signal changes from "0" to "1" (positive edge) at photoelectric barrier "PEB2" and, at the same time, the signal state at "PEB1" is "0", the object on the belt is moving to the right.



See also

Invert RLO (Page 5587)

S: Set output (Page 5590)

R: Reset output (Page 5589)

&: AND logic operation (Page 5581)

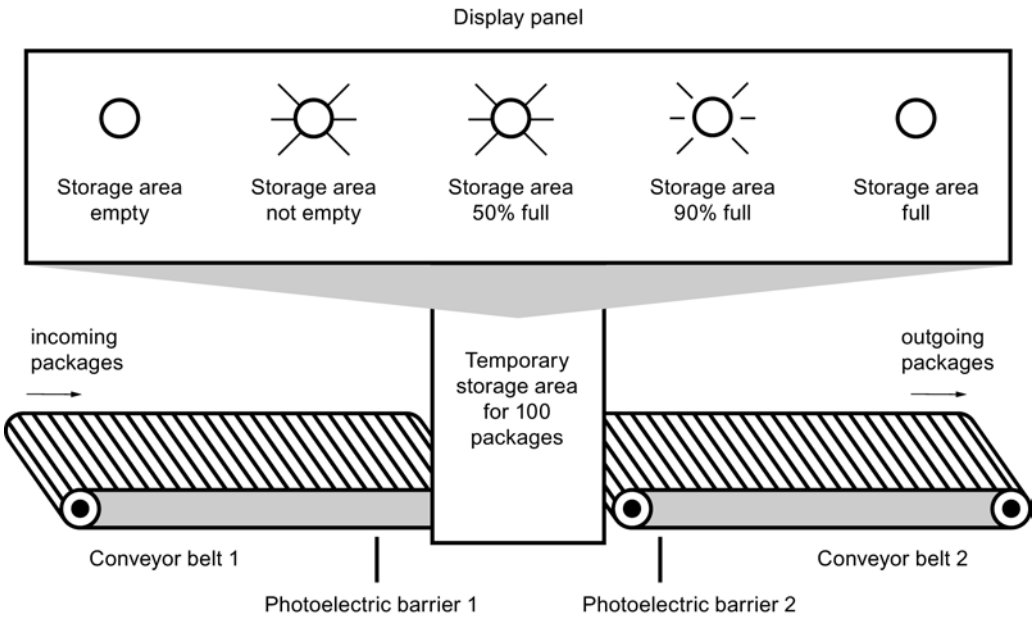
P: Scan operand for positive signal edge (Page 5594)

Example of detecting the fill level of a storage area

Detecting the fill level of a storage area

The following figure shows a system with two conveyor belts and a temporary storage area between them. Conveyor belt 1 delivers packages to the storage area. A photoelectric barrier at the end of conveyor belt 1 near the storage area detects how many packages are delivered to the storage area. Conveyor belt 2 transports packages from the temporary storage area to a loading dock onto which the packages are loaded for delivery to customers by truck. A photoelectric barrier at the storage area exit detects how many packages leave the storage area to be transported to the loading dock. Five display lamps indicate the capacity of the temporary storage area.

When a conveyor belt is restarted, the current count value is set to the number of packages available in the storage area.



Implementation

The following table shows the definition of the tags used:

Name	Data type	Description
PEB1	BOOL	Photoelectric barrier 1
PEB2	BOOL	Photoelectric barrier 2
RESET	BOOL	Reset counter
LOAD	BOOL	Set counter to value of "CV" parameter
STOCK	INT	Stock at restart
PACKAGECOUNT	INT	Number of packages in the storage area (current count value)
STOCK_PACKAGES	BOOL	Is set if the current count value is greater than or equal to the value of the tag "STOCK".
STOR_EMPTY	BOOL	Display lamp: Storage area empty
STOR_NOT_EMPTY	BOOL	Display lamp: Storage area not empty
STOR_50%_FULL	BOOL	Display lamp: Storage area 50% full
STOR_90%_FULL	BOOL	Display lamp: Storage area 90% full
STOR_FULL	BOOL	Display lamp: Storage area full
VOLUME_50	INT	Comparison value: 50 packages

Name	Data type	Description
VOLUME_90	INT	Comparison value: 90 packages
VOLUME_100	INT	Comparison value: 100 packages

The following networks show the FBD programming for activating the lamps:

Network 1:

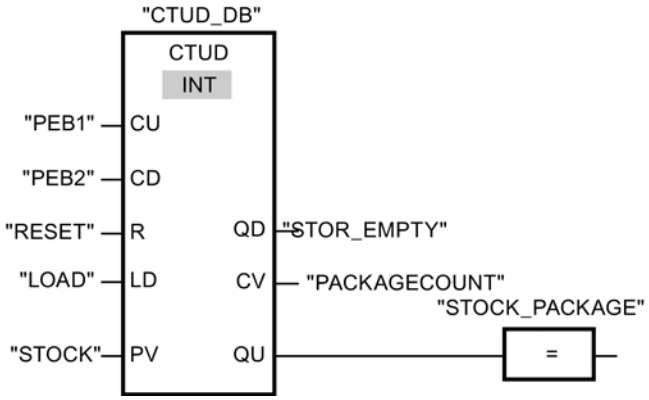
When a package is delivered to the storage area, the signal state at "PEB1" switches from "0" to "1" (positive signal edge). On a positive signal edge at "PEB1", the "Up" counter is enabled, and the current count value of "PACKAGECOUNT" is increased by one.

When a package is delivered from the storage area to the loading dock, the signal state at "PEB2" switches from "0" to "1" (positive signal edge). On a positive signal edge at "PEB2", the "Down" counter is enabled, and the current count value of "PACKAGECOUNT" is decreased by one.

If there are no packages in the storage area ("PACKAGECOUNT" = "0"), the "STOR_EMPTY" tag is set to signal state "1", and the "Storage area empty" lamp is switched on.

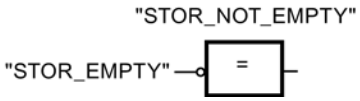
The current count value can be reset to "0" if the "RESET" tag is set to signal state "1".

If the "LOAD" tag is set to signal state "1", the current count value is set to the value of the "STOCK" tag. If the current count value is greater than or equal to the value of the "STOCK" tag, the "STOCK_PACKAGES" tag supplies the signal state "1".



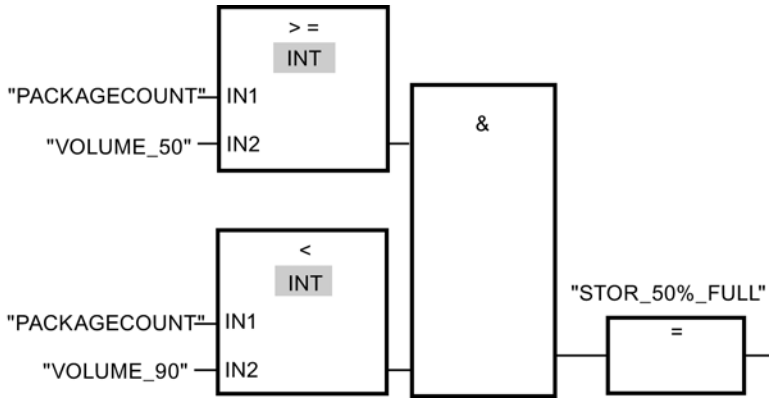
Network 2:

As long as there are packages in the storage area, the "STOR_NOT_EMPTY" tag is set to signal state "1", and the "Storage area not empty" lamp is switched on.



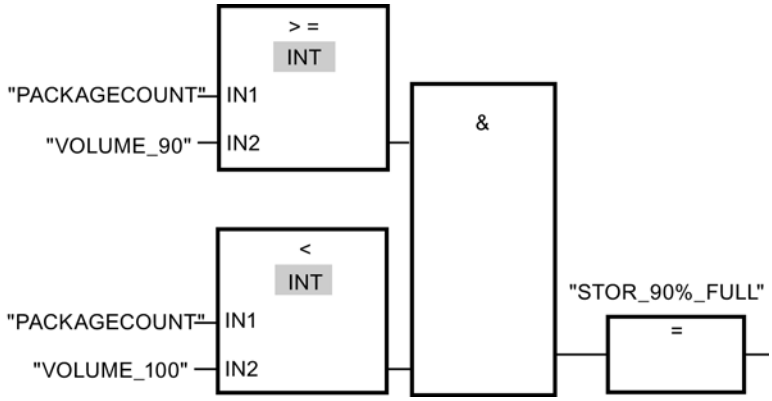
Network 3:

If the number of packages in the storage area is greater than or equal to 50, the lamp for the "Storage area 50% full" message switches on.



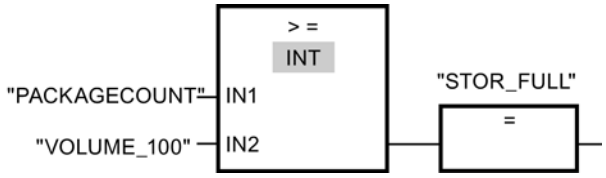
Network 4:

If the number of packages in the storage area is greater than or equal to 90, the "Storage area 90% full" lamp switches on.



Network 5:

If the number of packages in the storage area reaches 100, the lamp for the "Storage area full" message switches on.



See also

Invert RLO (Page 5587)

=: Assignment (Page 5588)

&: AND logic operation (Page 5581)

CMP >=: Greater or equal (Page 5647)

CMP <: Less than (Page 5651)

Example of calculating an equation

Calculating an equation

The following program example shows how you can use three mathematic functions to produce the same result as the following equation:

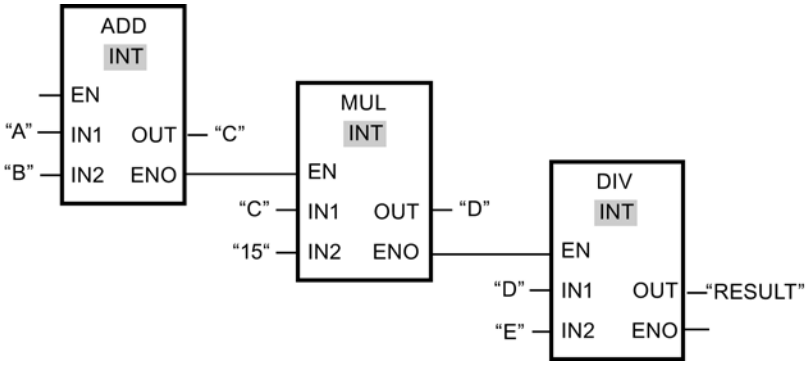
RESULT = ((A + B) x 15) / E

Implementation

The following table shows the definition of the tags used:

Name	Data type	Comment
A	INT	First value for addition
B	INT	Second value for addition
C	INT	First intermediate result
15	INT	Multiplier
D	INT	Second intermediate result
E	INT	Divisor
RESULT	INT	End result

The following network shows the FBD programming for calculating the equation:



The value of operand "A" is added to the value of operand "B". The sum is stored in operand "C". The value of operand "C" is multiplied by "15". The multiplication result is stored in operand "D". The value stored in operand "D" is then divided by the value of operand "E". The end result is stored in the "RESULT" operand.

See also

ADD: Add (Page 5652)

DIV: Divide (Page 5656)

Example of controlling room temperature

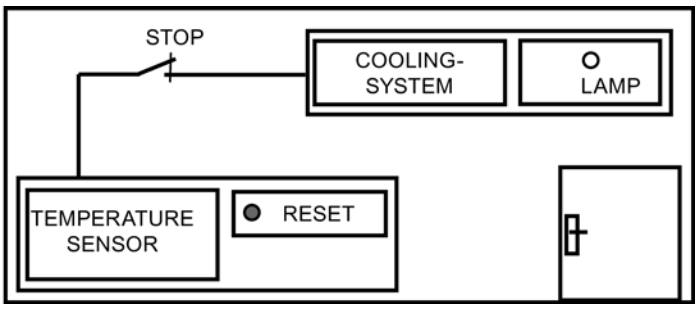
Controlling room temperature

In a cold room, the temperature must be maintained below zero degrees Celsius. Any temperature fluctuations are monitored by a sensor. If the temperature rises above zero degrees Celsius, the cooling system switches on for a preset time. The "Cooling system on" lamp is lit during this time.

The cooling system and the lamp are turned off if one of the following conditions is met:

- The sensor reports a temperature fall below zero degrees Celsius.
- The preset cooling time has elapsed.
- The pushbutton switch "Stop" has been pressed.

If the preset cooling time has expired and the temperature in the cold room is still too high, the cooling system can be restarted by means of the pushbutton switch "RESET".



Implementation

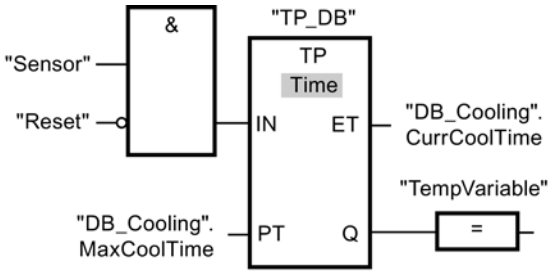
The following table shows the definition of the tags used:

Name	Data type	Comment
Sensor	BOOL	Temperature sensor signal
RESET	BOOL	Restart
STOP	BOOL	The cooling system is switched off.
MaxCoolTime	TIME	Preset cooling time This tag is defined in the "DB_Cool" data block.
CurrCoolTime	TIME	Currently elapsed cooling time This tag is defined in the "DB_Cool" data block.
Cooling system	BOOL	The cooling system is switched on.

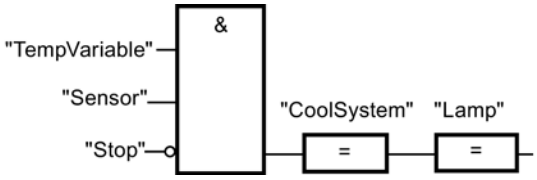
Name	Data type	Comment
Lamp	BOOL	The lamp for the "Cooling system on" message is switched on.
TempVariable	BOOL	Temporary tag This tag stores the signal state of the IEC time TP.

The following network shows the FBD programming for controlling room temperature:

Network 1:



Network 2:



When the temperature in the cold room rises above zero degrees Celsius, the signal state at the "Sensor" operand switches from "0" to "1" (positive signal edge). In the case of a positive signal edge at the input IN of the time function, the preset cooling time is started and the "TempVariable" receives the signal state "1". The signal state "1" of the "TempVariable" has the result in network 2 that the cooling system as well as the display lamp are turned on. The outputs "Sensor", "Cooling system" and "Lamp" must be programmed in network 2, because you can program only one coil at output Q of the time function.

If the temperature in the cold room falls below zero degrees Celsius, the signal state of the sensor switches back to "0". This switches the cooling system and lamp off.

If the sensor does not signal a temperature drop, the cooling system and lamp are switched off after the preset cooling time has elapsed, at the latest. In this case, the cooling process can be restarted by pressing the pushbutton switch "RESET". Pressing and releasing the pushbutton switch generates a new positive signal edge at input IN, which restarts the cooling system.

Using the pushbutton switch "STOP", the cooling system and the display lamp can be turned off at any time.

See also

&: AND logic operation (Page 5581)

=: Assignment (Page 5588)

TP: Generate pulse (Page 5600)

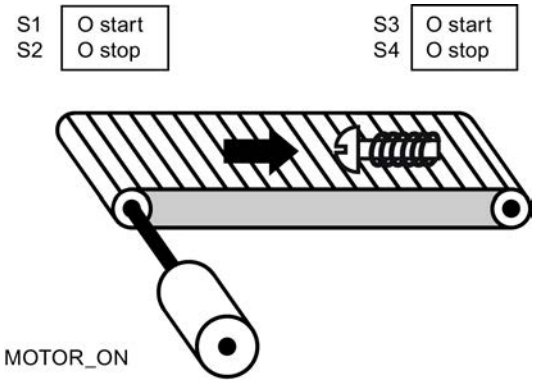
Invert RLO (Page 5587)

9.7.2.3 STL programming examples

Example: Bit logic instructions

Controlling a conveyor belt

The following figure shows a conveyor belt that can be activated electrically. There are two pushbutton switches at the beginning of the conveyor belt: S1 for START and S2 for STOP. There are also two pushbutton switches at the end of the conveyor belt: S3 for START and S4 for STOP. It is possible to start and stop the conveyor belt from either end.



Implementation

The following table shows the definition of the tags used:

Operand	Declaration	Data type	Description
StartSwitch_Left (S1)	Input	BOOL	Start switch on the left side of the conveyor belt
StopSwitch_Left (S2)	Input	BOOL	Stop switch on the left side of the conveyor belt
StartSwitch_Right (S3)	Input	BOOL	Start switch on the right side of the conveyor belt
StopSwitch_Right (S4)	Input	BOOL	Stop switch on the right side of the conveyor belt
MOTOR_ON/OFF	Output	BOOL	Switching the conveyor belt motor on/off

The following STL program shows how to implement this task:

```
STL                 Explanation
O #S1               // Scan start switch S1 for "1".
```

```
STL                 Explanation
O #S3               // Scan start switch S3 for "1".
S #"MOTOR_ON/OFF"   // If one of the start switches (S1 or S3) returns the signal
                    state "1", the conveyor belt motor will be turned on.
O #S2               // Scan stop switch S2 for "1".
O #S4               // Scan stop switch S4 for "1".
R #"MOTOR_ON/OFF"   // If one of the stop switches (S2 or S4) returns the signal
                    state "1", the conveyor belt motor will be turned off.
```

See also

O: OR logic operation (Page 5906)

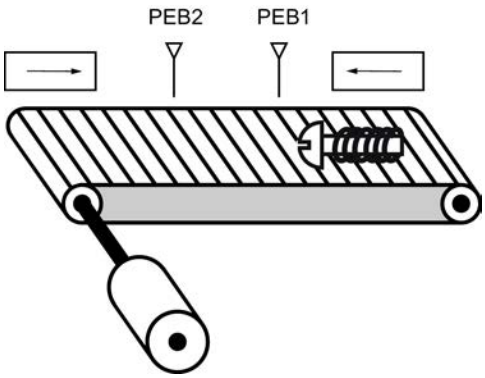
S: Set (Page 5924)

R: Reset (Page 5923)

Example of detecting the direction of a conveyor belt

Detecting the direction of a conveyor belt

The following figure shows a conveyor belt that is equipped with two photoelectric barriers (PEB1 and PEB2). The photoelectric barriers are designed to detect the direction in which an object is moving on the conveyor belt.



Implementation

The following table shows the definition of the tags used:

Name	Declaration	Data type	Description
S1	Input	BOOL	Photoelectric barrier 1
S2	Input	BOOL	Photoelectric barrier 2
TM1	Input	BOOL	Edge bit memory 1
TM2	Input	BOOL	Edge bit memory 2

Name	Declaration	Data type	Description
RIGHT	Output	BOOL	Display for movement to the right
LEFT	Output	BOOL	Display for movement to the left

The following STL program shows how to implement this example:

```
STL                 Explanation
A #S1               // Scan photoelectric barrier "S1" for "1"
FP #TM1             // Query positive edge
AN #S2              // Scan photoelectric barrier "S2" for "0"
S #LEFT             //If the signal state changes from "0" to "1" (positive edge)
                    at the photoelectric barrier "S1" and, at the same time, the
                    signal state at the photoelectric barrier "S2" is "0", the
                    object on the belt is moving to the left.
                    // The display for movement to the left is activated.
A #S2               // Scan photoelectric barrier "S2" for "1"
FP #TM2             // Query positive edge
AN #S1              // Scan photoelectric barrier "S1" for "0"
S #RIGHT            //If the signal state changes from "0" to "1" (positive edge)
                    at photoelectric barrier "S2" and, at the same time, the signal
                    state at photoelectric barrier "S1" is "0", the object on the
                    belt is moving to the right.
                    // The display for movement to the right is activated.
AN #S1              // Scan photoelectric barrier "S1" for "0"
AN #S2              // Scan photoelectric barrier "S2" for "0"
R #LEFT             // The display for movement to the left will be turned off when
                    the signal state at both photoelectric barriers is "0".
R #RIGHT            // The display for movement to the right will be turned off when
                    the signal state at both photoelectric barriers is "0".
```

See also

A: AND logic operation (Page 5903)

AN: Negated AND logic operation (Page 5905)

S: Set (Page 5924)

R: Reset (Page 5923)

FP: Scan RLO for positive signal edge (Page 5930)

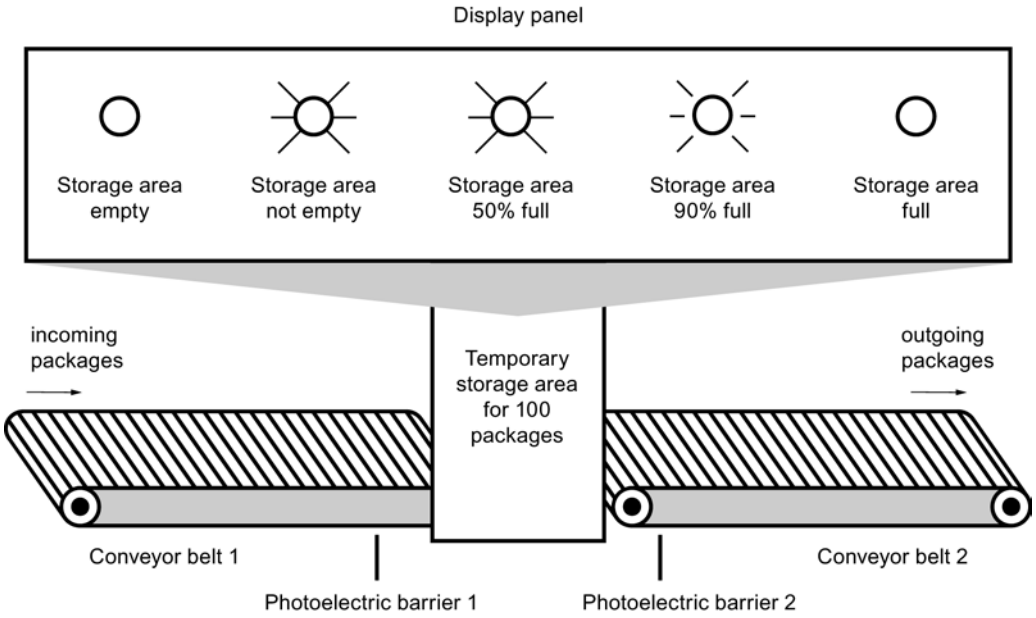
Example of detecting the fill level of a storage area

Detecting the fill level of a storage area

The following figure shows a system with two conveyor belts and a temporary storage area between them. Conveyor belt 1 delivers packages to the storage area. A photoelectric barrier

at the end of conveyor belt 1 near the storage area detects how many packages are delivered to the storage area. Conveyor belt 2 transports packages from the temporary storage area to a loading dock onto which the packages are loaded for delivery to customers by truck. A photoelectric barrier at the storage area exit detects how many packages leave the storage area to be transported to the loading dock. Five display lamps indicate the capacity of the temporary storage area.

When a conveyor belt is restarted, the current count value is set to the number of packages available in the storage area.



Implementation

The following table shows the definition of the tags used:

Name	Data type	Address	Description
PACKAGECOUNT	COUNTER	C1	Number of packages in the storage area (current count value)

Name	Section	Data type	Description
LS1	Input	BOOL	Photoelectric barrier 1
LS2	Input	BOOL	Photoelectric barrier 2
STOR_EMPTY	Output	BOOL	Display lamp: Storage area empty
STOR_NOT_EMPTY	Output	BOOL	Display lamp: Storage area not empty
STOR_50%_FULL	Output	BOOL	Display lamp: Storage area 50% full

Name	Section	Data type	Description
STOR_90%_FULL	Output	BOOL	Display lamp: Storage area 90% full
STOR_FULL	Output	BOOL	Display lamp: Storage area full

The following STL program shows how to implement this example:

```
STL                 Explanation
A #LS1              // Scan photoelectric barrier "LS1" for "1".
CU "PACKAGECOUNT"   // At a positive edge at photoelectric barrier "LS1", the count
                    value of counter "PACKAGECOUNT" is increased by one.
A #LS2              // Scan photoelectric barrier "LS2" for "1".
CD "PACKAGECOUNT"   // At a positive edge at photoelectric barrier "LS2", the count
                    value of counter "PACKAGECOUNT" is decreased by one.
AN "PACKAGECOUNT"   // Scan count value for "0".
= #STOR_EMPTY       // With a count value of "0" the display lamp "storage area
                    empty" is switched on.
A "PACKAGECOUNT"    // Scan count value for "1".
= #STOR_NOT_EMPTY   // With a count value greater than "0" the display lamp "Storage
                    area not empty" is switched on.
L 50                // Load the comparison value "50" to accumulator 1.
L "PACKAGECOUNT"    // Move the comparison value to accumulator 2.
                    // Load the current count value to accumulator 1.
<=I                 // Compare values
= #"STOR_50%_FULL"  // With a count value greater than or equal to "50" the display
                    lamp "Storage area 50% full" is switched on.
L 90                // Move the counter value to accumulator 2.
                    // Load the comparison value "90" to accumulator 1.
>=I                 // Compare values
= #"STOR_90%_FULL"  // With a count value greater than or equal to "90" the display
                    lamp "Storage area 90% full" is switched on.
L "PACKAGECOUNT"    // Load the current count value to accumulator 1.
L 100               // Move the counter value to accumulator 2.
                    // Load the comparison value "100" to accumulator 1.
>=I                 // Compare values
= #STOR_FULL        // At a count value greater than "100" the display lamp "Storage
                    area full" is switched on.
```

See also

A: AND logic operation (Page 5903)

AN: Negated AND logic operation (Page 5905)

=: Assignment (Page 5921)

CD: Count down (Page 5957)

CU: Count up (Page 5956)

L: Load (Page 5994)

? I: Compare 16-bit integers (Page 5959)

Example of calculating an equation

Calculating an equation

The sample program shows you how to use three math instructions to calculate the following equation:

RESULT = ((A + B) x C) /D

Implementation

The following table shows the declaration of the operands used in the PLC tag table:

Name	Data type	Comment
A	INT	First value for addition
B	INT	Second value for addition
C	INT	Multiplier
D	INT	Divisor
RESULT	INT	End result

The following STL program shows how to implement this example:

```
 STL                 Explanation
L "A"                // Load value of the operand "A" to accumulator 1
L "B"                // Load value of operand "A" to accumulator 2
                     // Load value of the operand "B" to accumulator 1
+I                   // Add the values of accumulator 1 and 2
                     // Store sum in accumulator 1
L "C"                // Move the sum to accumulator 2
                     // Load value of the operand "C" to accumulator 1
*I                   // Multiply the values of accumulator 1 and 2
                     // Store product in accumulator 1
L "D"                // Move the product to accumulator 2
                     // Load value of the operand "D" to accumulator 1
/I                   // Divide the value of accumulator 2 by the value of accumulator
                     1
                     // Store result in accumulator 1
T "RESULT"           // Transfer the result to operand "RESULT"
```

See also

L: Load (Page 5994)

T: Transfer (Page 6003)

+I: Add integers (16 bit) (Page 5964)

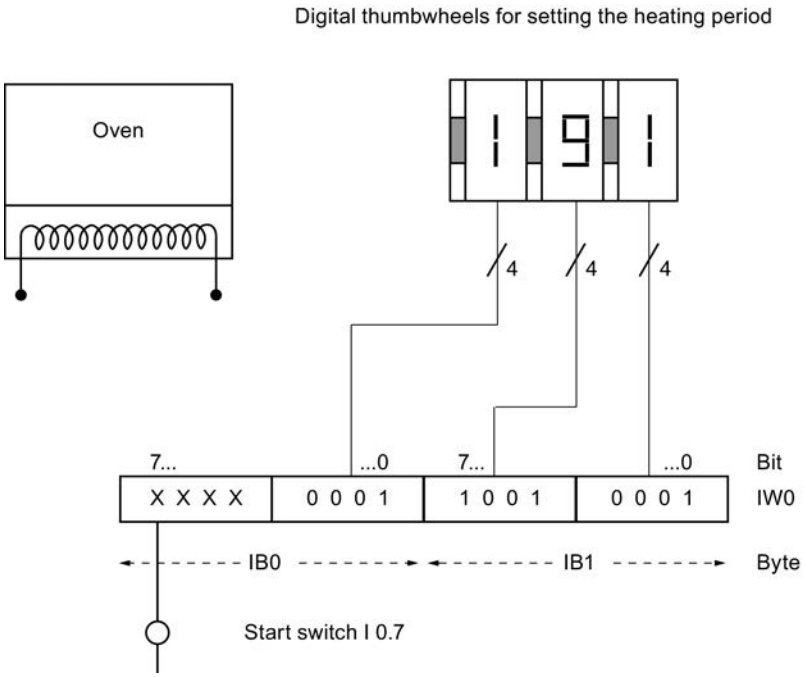
*I: Multiply integers (16 bit) (Page 5966)

/I: Divide integers (16 bit) (Page 5967)

Example: Word logic instructions

Heating an oven

The following illustration shows an oven that is turned on with a start switch. The heating process is started when the start switch is pressed. The heating period is set with digital thumbwheels. The heating period is set in seconds using the BCD format.



Implementation

The following table shows the declaration of the operands used in the PLC tag table:

Name	Data type	Address	Comment
DURATION	WORD	EW0	Heating period in seconds • I1.0 to I1.3: Thumbwheels for ones • I1.4 to I1.7: Thumbwheels for tens • I0.0 to I0.3: Thumbwheels for hundreds
HEATING	TIMER	T1	Time that is started with the preset heating period.

The following table shows the declaration of the operands used in the block interface of the code block:

Name	Section	Data type	Comment
START	Input	BOOL	Start switch
START_HEATING	Output	BOOL	Start of the heating process

The following STL program shows how to implement this example:

```
STL                 Explanation
A "HEATING"         // Scan to see if time has started
= #START_HEATING    // Start the heating process
BEC                 // For RLO=1 End processing of block
                    // This prevents the "HEATING" time from being restarted if the
                    button is pressed
L "DURATION"        // Load the heating period in accumulator 1
AW W#16#0FFF        // Reset input bits I0.4 to I0.7 to "0"
OW W#16#4000        // Set seconds in bits I1.2 and I1.3 of accumulator 1
A #START            // Scan start switch for "1"
SE "HEATING"        // Start heating process as an extended pulse with a positive
                    edge at the start switch
```

See also

L: Load (Page 5994)

SE: Start extended pulse timer (Page 5941)

A: AND logic operation (Page 5903)

=: Assignment (Page 5921)

BEC: Conditional block end (Page 6067)

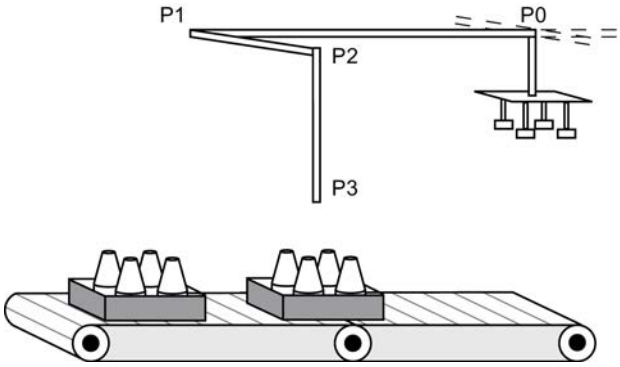
AW: AND logic operation word by word (Page 6074)

OW: OR logic operation word by word (Page 6076)

Example of a step sequence

Programming a step sequence

The following figure shows a station for removing glass containers from a pallet. The pallets are transported on a conveyor belt to the station. When a pallet with glass containers reaches the station, the conveyor belt will be stopped and a gripper moves from its basic position (P0) to the position above the pallet (P2). Once the gripper is above the pallet, the gripping clamps will open and the gripper lowered. Sensors detect the actual position of the gripper and the status of the gripping clamps. The sequence of the gripper movement in this example is implemented by a step sequence. You can program the additional steps required for removing the bottles and transporting them on an additional conveyor belt.



Implementation

The following table shows the declaration of the operands used in the PLC tag table:

Name	Data type	Comment
NUMBER	INT	Step number
Tag_Error	BOOL	Operand that is set when the step number is greater than 3 or if one of the steps was not executed.

The following table shows the declaration of the operands used in the block interface of the code block:

Name	Section	Data type	Comment
POS_0	Input	BOOL	Gripper in basic position (P0)
POS_1	Input	BOOL	Gripper in position 1 (P1)
POS_2	Input	BOOL	Gripper in position 2 (P2)
GRIPPER_OPEN	Input	BOOL	Gripping clamps open
OUT_POS_1	Output	BOOL	Move gripper to position 1
OUT_POS_2	Output	BOOL	Move gripper to position 2
OUT_GRIPPER	Output	BOOL	Open gripping clamps
OUT_POS_3	Output	BOOL	Move gripper to position 3

The following STL program shows how to implement this example:

```
STL                          Explanation
L "NUMBER"                   // Load the step number to accumulator 1.
JL END                       // Start of the jump list
JU POSITION_0                // At a value of "0" in the accumulator 1 jump to jump
                             label "POSITION_0".
JU POSITION_1                // At a value of "1" in the accumulator 1 jump to jump
                             label "POSITION_1".
JU POSITION_2                // At a value of "2" in the accumulator 1 jump to jump
                             label "POSITION_2".
JU POSITION_3                // At a value of "3" in the accumulator 1 jump to jump
                             label "POSITION_3".
END: JU ERROR                // End of the jump list
                             // At a step number greater than 3 jump to jump label
                             "ERROR".
POSITION_0: A #POS_0         // Jump label "POSITION_0"
                             // Scan to see if gripper is in basic position (P0).
= #OUT_POS_1                 // If condition is met, set output "OUT_POS_1" and
                             move gripper to position 1 (P1).
JCN ERROR                    // With RLO "0" jump to jump label "ERROR".
JC NEXT                      // With RLO "1" jump to jump label "NEXT".
POSITION_1: A #POS_1         // Jump label "POSITION_1"
                             // Scan to see if gripper is in position 1 (P1).
= #OUT_POS_2                 // If condition is met, set output "OUT_POS_2" and
                             move gripper to position 2 (P2).
JCN ERROR                    // With RLO "0" jump to jump label "ERROR".
JC NEXT                      // With RLO "1" jump to jump label "NEXT".
POSITION_2: A #POS_2         // Jump label "POSITION_2"
                             // Scan to see if gripper is in position 2 (P2).
= #OUT_GRIPPER               // If condition is met, set output "OUT_GRIPPER" and
                             open gripping clamps.
```

STL	Explanation
JCN ERROR	// With RLO "0" jump to jump label "ERROR".
JC NEXT	// With RLO "1" jump to jump label "NEXT".
POSITION_3: A #POS_2	// Jump label "POSITION_3"
	// Scan to see if gripper is in position 2 (P2).
A #GRIPPER_OPEN	// Scan to see if the gripping clamps are open
= #OUT_POS_3	// If condition is met, set output "OUT_POS_3" and move gripper to position 3 (P3).
JCN ERROR	// With RLO "0" jump to jump label "ERROR".
JC NEXT	// With RLO "1" jump to jump label "NEXT".
NEXT: INC 1	// Jump label "NEXT"
	// Increase step number in accumulator 1 by one.
T "NUMBER"	// Transfer step number to operand "NUMBER".
L 3	// Move the current step number to accumulator 2.
	// Load value 3 to accumulator 1.
>I	// Scan to see if current step number is greater than 3.
JC RESET_NUMBER	// With a scan result "1" jump to jump label "RESET_NUMBER" and continue with program processing
BEU	// End block
RESET_NUMBER: L 0	// Jump label "RESET_NUMBER"
	// Load value "0" in accumulator 1.
T "NUMBER"	// Assign value "0" to operands "NUMBER" (step number).
BEU	// End block
ERROR: NOT	// Jump label "ERROR"
= "Tag_Error"	// Assign negated RLO to the operand "Tag_Error".
BEU	// End block

See also

JC: Jump if RLO = 1 (Page 6034)

JCN: Jump if RLO = 0 (Page 6035)

JU: Unconditional jump (Page 6032)

JL: Define jump list (Page 6057)

BEU: Unconditional block end (Page 6069)

L: Load (Page 5994)

T: Transfer (Page 6003)

? I: Compare 16-bit integers (Page 5959)

A: AND logic operation (Page 5903)

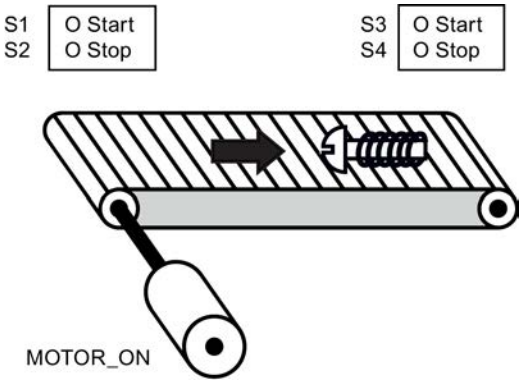
=: Assignment (Page 5921)

9.7.2.4 SCL programming languages

Example: Bit logic instructions

Controlling a conveyor belt

The following figure shows a conveyor belt that can be activated electrically. There are two pushbutton switches at the beginning of the conveyor belt: S1 for START and S2 for STOP. There are also two pushbutton switches at the end of the conveyor belt: S3 for START and S4 for STOP. It is possible to start and stop the conveyor belt from either end.



Implementation

The following table shows the definition of the tags used:

Operand	Declaration	Data type	Description
StartSwitch_Left (S1)	Input	BOOL	Start switch on the left side of the conveyor belt
StopSwitch_Left (S2)	Input	BOOL	Stop switch on the left side of the conveyor belt
StartSwitch_Right (S3)	Input	BOOL	Start switch on the right side of the conveyor belt
StopSwitch_Right (S4)	Input	BOOL	Stop switch on the right side of the conveyor belt
MOTOR_ON	Output	BOOL	Turn on the conveyor belt motor
MOTOR_OFF	Output	BOOL	Turn off the conveyor belt motor

The following SCL program shows how to implement this task:

SCL

```
IF "StartSwitch_Left" OR "StartSwitch_Right" THEN
"MOTOR_ON" := 1;
END_IF;
```

SCL

```
IF "StopSwitch_Left" OR "StopSwitch_Right" THEN
"MOTOR_OFF" := 1;
END_IF;
```

The conveyor belt motor is switched on when start switch "StartSwitch_Left" or "StartSwitch_Right" is pressed. The conveyor belt motor is switched off when stop switch "StopSwitch_Left" or "StopSwitch_Right" is pressed.

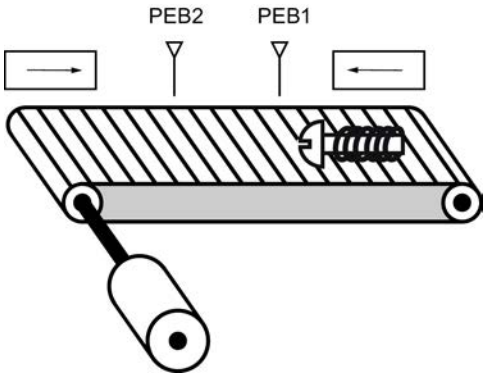
See also

Logical expressions (Page 2686)

Example of detecting the direction of a conveyor belt

Detecting the direction of a conveyor belt

The following figure shows a conveyor belt that is equipped with two photoelectric barriers (PEB1 and PEB2). The photoelectric barriers are designed to detect the direction in which an object is moving on the conveyor belt.



Implementation

The following table shows the definition of the tags used:

Name	Declaration	Data type	Description
Photoelectric barrier LS1	Input	BOOL	Photoelectric barrier 1
Photoelectric barrier LS2	Input	BOOL	Photoelectric barrier 2
RIGHT	Output	BOOL	Display for movement to the right
LEFT	Output	BOOL	Display for movement to the left

The following SCL program shows how to implement this example:

SCL

```
IF "LS1" = 1 AND NOT "LS2" = 0
THEN "LEFT" = 1;
IF "LS2" = 1 AND NOT "LS1" = 0
THEN "RIGHT" := 1;
IF "LS2" = 0
THEN "RIGHT" := 0;
IF "LS1" = 0
THEN "LEFT" = 0;
```

If the photoelectric barrier "PEB1" has signal state "1" and the photoelectric barrier "PEB2" has signal state "0" at the same time, the object on the belt is moving to the left. If the photoelectric barrier "PEB2" has signal state "1" and the photoelectric barrier "PEB1" has signal state "0" at the same time, the object on the belt is moving to the right. The displays for a movement to the left or right will be turned off when the signal state at both photoelectric barriers is "0".

See also

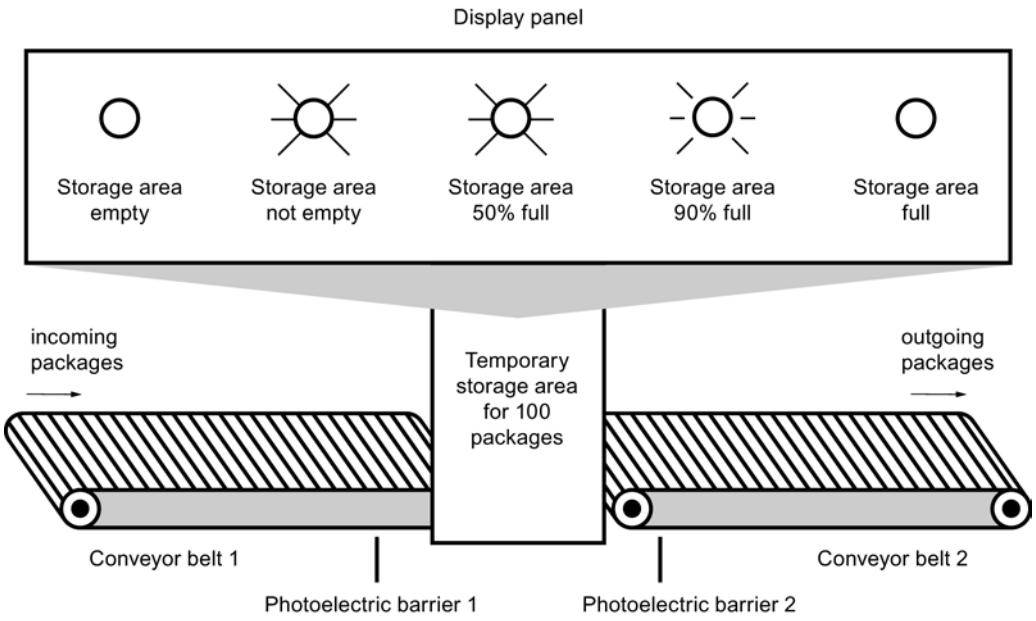
Logical expressions (Page 2686)

Example of detecting the fill level of a storage area

Detecting the fill level of a storage area

The following figure shows a system with two conveyor belts and a temporary storage area between them. Conveyor belt 1 delivers packages to the storage area. A photoelectric barrier at the end of conveyor belt 1 near the storage area detects how many packages are delivered to the storage area. Conveyor belt 2 transports packages from the temporary storage area to a loading dock onto which the packages are loaded for delivery to customers by truck. A photoelectric barrier at the storage area exit detects how many packages leave the storage area to be transported to the loading dock. Five display lamps indicate the capacity of the temporary storage area.

When a conveyor belt is restarted, the current count value is set to the number of packages available in the storage area.



Implementation

The following table shows the definition of the tags used:

Name	Declaration	Data type	Description
PEB1	Input	BOOL	Photoelectric barrier 1
PEB2	Input	BOOL	Photoelectric barrier 2
RESET	Input	BOOL	Reset counter
LOAD	Input	BOOL	Set counter to value of "CV" parameter
STOCK	Input	INT	Stock at restart
PACKAGECOUNT	Output	INT	Number of packages in the storage area (current count value)
STOCK_PACKAGES	Output	BOOL	Is set if the current count value is greater than or equal to the value of the tag "STOCK".
STOR_EMPTY	Output	BOOL	Display lamp: Storage area empty
STOR_NOT_EMPTY	Output	BOOL	Display lamp: Storage area not empty
STOR_50%_FULL	Output	BOOL	Display lamp: Storage area 50% full
STOR_90%_FULL	Output	BOOL	Display lamp: Storage area 90% full
STOR_FULL	Output	BOOL	Display lamp: Storage area full

Name	Declaration	Data type	Description
VOLUME_50	Input	INT	Comparison value: 50 packages
VOLUME_90	Input	INT	Comparison value: 90 packages
VOLUME_100	Input	INT	Comparison value: 100 packages

The following SCL program shows how to implement this example:

When a package is delivered to the storage area, the signal state at "PEB1" switches from "0" to "1" (positive signal edge). On a positive signal edge at "PEB1", the "Up" counter is enabled, and the current count value of "PACKAGECOUNT" is increased by one.

When a package is delivered from the storage area to the loading dock, the signal state at "PEB2" switches from "0" to "1" (positive signal edge). On a positive signal edge at "PEB2", the "Down" counter is enabled, and the current count value of "PACKAGECOUNT" is decreased by one.

If there are no packages in the storage area ("PACKAGECOUNT" = "0"), the "STOR_EMPTY" tag is set to signal state "1", and the "Storage area empty" lamp is switched on.

The current count value can be reset to "0" if the "RESET" tag is set to signal state "1".

If the "LOAD" tag is set to signal state "1", the current count value is set to the value of the "STOCK" tag. If the current count value is greater than or equal to the value of the "STOCK" tag, the "STOCK_PACKAGES" tag supplies the signal state "1".

SCL

```
"CTUD_DB".CTUD(CU := "PEB1",
               CD := "PEB2",
               R  := "RESET",
               LD := "LOAD",
               PV := "STOCK",
               QU := "STOCK_PACKAGES",
               QD := "STOR_EMPTY",
               CV := "PACKAGECOUNT");
```

As long as there are packages in the storage area, the "STOR_NOT_EMPTY" tag is set to signal state "1", and the "Storage area not empty" lamp is switched on.

SCL

```
"STOR_NOT_EMPTY" := NOT "STOR_EMPTY"
```

If the number of packages in the storage area is greater than or equal to 50, the lamp for the•"Storage area 50% full" message switches on.

SCL

```
IF "PACKAGECOUNT" >= "VOLUME_50" THEN "STOR_50%_FULL" := 1;
IF "PACKAGECOUNT" <= "VOLUME_90" THEN "STOR_50%_FULL" := 1;
```

If the number of packages in the storage area is greater than or equal to 90, the "Storage area 90% full" lamp switches on.

```
SCL
IF "PACKAGECOUNT" >= "VOLUME_90" THEN "STOR_90%_FULL" := 1;
IF "PACKAGECOUNT" <= "VOLUME_100" THEN "STOR_90%_FULL" := 1;
```

If the number of packages in the storage area reaches 100, the lamp for the "Storage area full" message switches on.

```
SCL
IF "PACKAGECOUNT" >= "VOLUME_100" THEN "STOR_FULL" := 1;
```

9.8 References

9.8.1 References (S7-1200, S7-1500)

9.8.1.1 General parameters of the instructions

Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions

Asynchronous instructions

For instructions that work asynchronously, the function is executed with several calls.

Identification of the job

If you use asynchronous instructions to trigger a process interrupt, output control commands to DP slaves, start a data transfer, or abort a non-configured connection with one of the SFCs listed above and then call the same SFC again before the current job is completed, then the reaction of the SFC will depend on whether the second call involves the same job.

Parameter REQ

The input parameter REQ (request) is used solely to start the job:

- If you call the instruction for a job that is not currently active, the job is started with REQ = 1 (case 1).
- If a particular job has been started and not yet completed and you call the instruction again to perform the same job (for example, in a cyclic interrupt OB), then REQ is not evaluated by the instruction (case 2).

Parameter RET_VAL and BUSY

The output parameters RET_VAL and BUSY indicate the status of the job.

Pay attention to the note in section: Evaluating errors with output parameter RET_VAL (Page 3152)

- In case 1 (first call with REQ=1), the input parameter will be entered in RET_VALW#16#7001 if system resources are available and supply is correct. BUSY will be set.
  If the required system resources are currently being used or the input parameters have errors, the corresponding error code is entered in RET_VAL and BUSY has the value 0.
- In case 2 (interim call) W#16#7002 will be entered in RET_VAL (this corresponds to a warning: Job still being processed!), and BUSY will be set.
- The following applies to the last call for a job:
  - For instruction "DPNRM_DG (Page 4614)", the number of data in bytes will entered as integer in RET_VAL in case there are no errors in data transmission. BUSY has the value "0" in this case.
    If there is an error, then the error information will be entered in RET_VAL and you should not evaluate BUSY in this case.
  - For all other instructions, "0" will be entered in RET_VAL if the job was executed without errors and BUSY has the value "0" in this case. If there is an error, the error code is entered in RET_VAL and BUSY has the value "0" in this case.

  Note

  If the first and last call coincide, the reaction is the same for RET_VAL and BUSY as described for the last call.

Overview

The following table provides you with an overview of the relationships explained above. In particular, it shows the possible values of the output parameters if the execution of the job is not completed after an instruction call has been completed.

Note

Following every call, you must evaluate the relevant output parameters in your program.

Relationship between call, REQ, RET_VAL and BUSY during execution of a "running" job.

Number of the call	Type of call	REQ	RET_VAL	BUSY
1	First call	1	W#16#7001	1
			Error code	0
2 to (n - 1)	Intermediate call	irrelevant	W#16#7002	1
n	Last call	irrelevant	W#16#0000, if no errors have occurred.	0
			Error code if errors occurred	0

Evaluating errors with output parameter RET_VAL

Types of error information

An executed instruction indicates in the user program whether or not the CPU was able to execute the function of the instruction successfully.

You can obtain information about any errors that occurred in two ways:

- In the BR bit of the status word
- in the output parameter RET_VAL (return value).

Note

Before evaluating the output parameters specific to an instruction, you should always follow the steps below:

- First, evaluate the BR bit of the status word.
- Then check the output parameter RET_VAL.

If the BR bit indicates that an error has occurred or if RET_VAL contains a general error code, you should not evaluate the instruction-specific output parameters.

Error information in the return value

An instruction indicates that an error occurred during its execution by entering the value "0" in the binary result bit (BR) of the status word. Some instructions provide an additional error code at an output parameter known as the return value (RET_VAL). If a general error is entered in the output parameter RET_VAL (see below for explanation), this is only indicated by the value "0" in the BR bit of the status word.

The return value is of the data type integer (INT). The relationship of the return value to the value "0" indicates whether or not an error occurred during execution of the function.

CPU execution of the instruction	BR	Return value	Sign of the integer
With error(s)	0	less than "0"	negative (sign bit is "1")
Without error	1	greater than or equal to "0"	positive (sign bit is "0")

Reacting to error information

There are two types of error codes in RET_VAL:

- A general error code that all instructions can output and
- A specific error code that an instruction can output and which relates to its specific function.

You can write your program so that it reacts to the errors that occur during execution of an instruction. This way you prevent further errors occurring as a result of the first error.

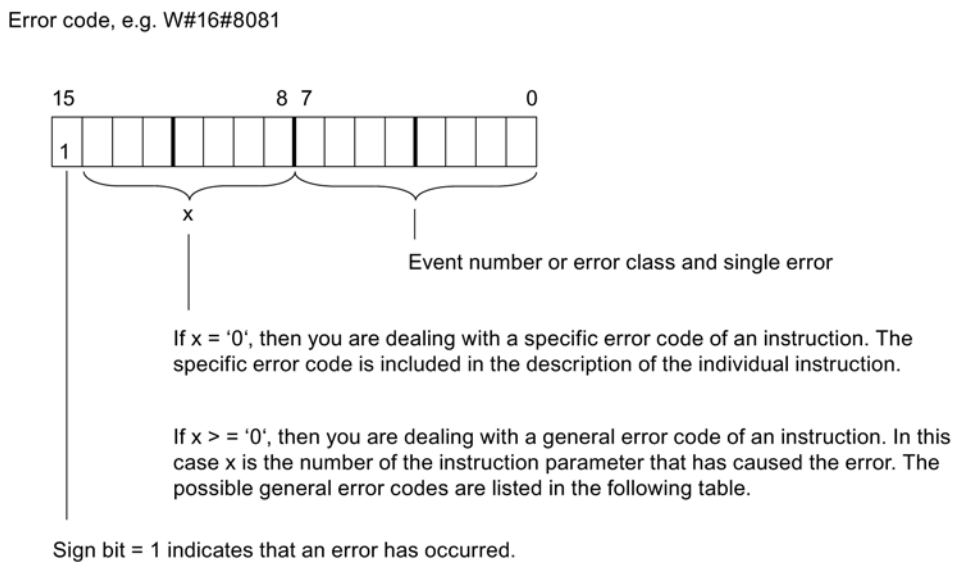
General and specific error information

The return value (RET_VAL) of an instruction provides one of the two following types of error codes:

- A general error code that relates to errors that can occur in any instruction.
- A specific error code that relates only to the particular instruction.

Even though the data type of the output parameter RET_VAL is an integer (INT), the error codes for the instruction are grouped according to hexadecimal values. If you want to examine a return value and compare the value with the error codes listed in this documentation, then display the error code in hexadecimal format.

The figure below shows the structure of a system function error code in hexadecimal format.

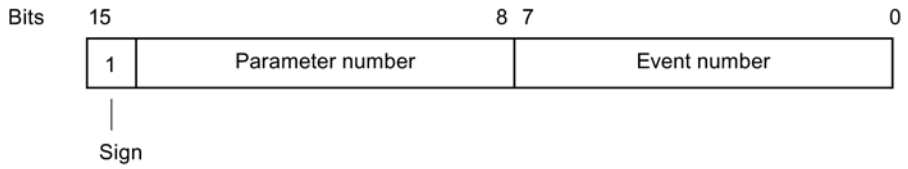


General error information

The general error code indicates errors that can occur in all instructions. A general error code consists of the following two numbers:

- A parameter number from 1 to 111, where 1 indicates the first parameter of the called instruction, 2 the second parameter, and so forth.
- An event number from 0 to 127. The event number indicates that a synchronous error occurred.

The following table lists the codes for general errors and an explanation of each error.



Note

If a general error code was entered in RET_VAL, then the following situations are possible:

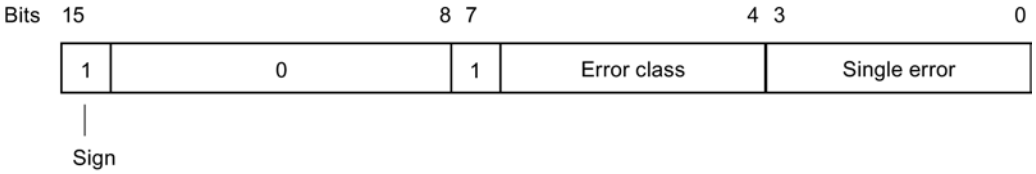
- The action associated with the instruction may have been started or already completed.
- A specific instruction error may have occurred when the action was performed. As a result of a general error that occurred later, the specific error could, however, no longer be indicated.

Specific error information

Some instructions have a return value that provides an error code specific for the instruction. A specific error code indicates errors that can occur only in specific instructions.

A specific error code consists of the following two numbers:

- An error class from 0 to 7.
- An error number from 0 to 15.



General error codes

The following table explains the general error codes of a return value. The error code is shown in hexadecimal format. The letter x in each code number is simply a place holder and represents the number of the system function parameter that caused the error.

General error codes

Error code (W#16#...)	Explanation
8x7F	Internal error This error code indicates an internal error at parameter x.
8x01	Illegal syntax ID at an VARIANT parameter
8x22	Range length error when reading a parameter.
8x23	Range length error when writing a parameter. This error code indicates that the parameter x is located either entirely or partly outside the range of an address, or that the length of a bit range is not a multiple of 8 with an VARIANT parameter.
8x24	Range error when reading a parameter.

Error code (W#16#...)	Explanation
8x25	Range error when writing a parameter. This error code indicates that the parameter x is located in a range that is illegal for the system function. Refer to the descriptions of the individual functions for information about the illegal ranges.
8x26	The parameter contains a timer cell number that is too high. This error code indicates that the timer cell specified in parameter x does not exist.
8x27	The parameter contains a counter cell number that is too high (counter number error). This error code indicates that the counter cell specified in parameter x does not exist.
8x28	Alignment error when reading a parameter.
8x29	Alignment error when writing a parameter. This error code indicates that the reference to parameter x is an operand with bit address that is not equal to 0.
8x30	The parameter is located in a read-only global DB.
8x31	The parameter is located in a read-only instance DB. This error code indicates that parameter x is located in a read-only data block. If the data block was opened by the system function itself, the system function always returns the value W#16#8x30.
8x32	The parameter contains a DB number that is too high (DB number error).
8x34	The parameter contains an FC number that is too high (FC number error).
8x35	The parameter contains an FB number that is too high (FB number error). This error code indicates that parameter x contains a block number higher than the highest permitted number.
8x3A	The parameter contains the number of a DB that is not loaded.
8x3C	The parameter contains the number of an FC that is not loaded.
8x3E	The parameter contains the number of an FB that is not loaded.
8x42	An access error occurred while the system was attempting to read a parameter from the peripheral input area.
8x43	An access error occurred while the system was attempting to write a parameter to the peripheral output area.
8x44	Error in the nth (n > 1) read access after an error occurred.
8x45	Error in the nth (n > 1) write access after an error occurred. This error code indicates that access to the required parameter is denied.

9.8.1.2 Basic instructions

LAD

Bit logic operations

---| |---: Normally open contact

Description

The activation of the normally open contact depends on the signal state of the associated operand. When the operand has signal state "1", the normally open contact closes and the signal state at the output is set to the signal state of the input.

When the operand has signal state "0", the normally open contact is not activated and the signal state at the output of the instruction is reset to "0".

Two or more normally open contacts are linked bit-by-bit by AND when connected in series. With a series connection, power flows when all contacts are closed.

The normally open contacts are linked by OR when connected in parallel. With a parallel connection, power flows when one of the contacts is closed.

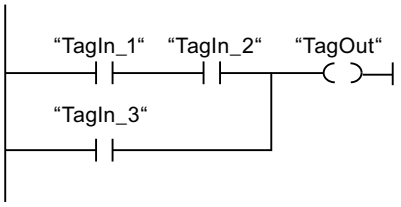
Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand>	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Operand whose signal state is queried.

Example

The following example shows how the instruction works:



The "TagOut" operand is set when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The operand "TagIn_3" has the signal state "1".

See also

Overview of the valid data types (Page 2236)

Inserting operands into LAD instructions (Page 2601)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

---| / |---: Normally closed contact

Description

The activation of the normally closed contact depends on the signal state of the associated operand. When the operand has signal state "1", the normally closed contact opens and the signal state at the output of the instruction is reset to "0".

When the operand has signal state "0", the normally closed contact is not enabled and the signal state of the input is transferred to the output.

Two or more normally closed contacts are linked bit-by-bit by AND when connected in series. With a series connection, power flows when all contacts are closed.

The normally closed contacts are linked by OR when connected in parallel. With a parallel connection, power flows when one of the contacts is closed.

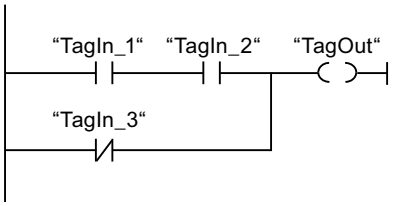
Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand>	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Operand whose signal state is queried.

Example

The following example shows how the instruction works:



The "TagOut" operand is set when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The operand "TagIn_3" has the signal state "0".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

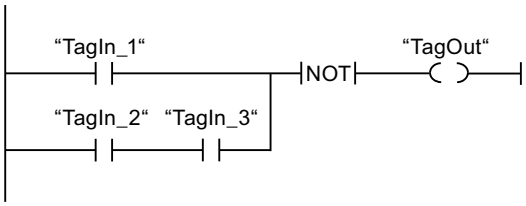
--|NOT|--: Invert RLO

Description

You use the "Invert RLO" instruction to invert the signal state of the result of logic operation (RLO). If the signal state is "1" at the input of the instruction, the output of the instruction has signal state "0". If the signal state is "0" at the input of the instruction, the output has the signal state "1".

Example

The following example shows how the instruction works:



Operand "TagOut" is reset when one of the following conditions is fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The signal state of the operands "TagIn_2" and "TagIn_3" is "1".

See also

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

---( )---: Assignment

Description

You can use the "Assignment" instruction to set the bit of a specified operand. If the result of logic operation (RLO) at the input of the coil has signal state "1", the specified operand is set to signal state "1". If the signal state is "0" at the input of the coil, the bit of the specified operand is reset to "0".

The instruction does not influence the RLO. The RLO at the input of the coil is sent directly to the output.

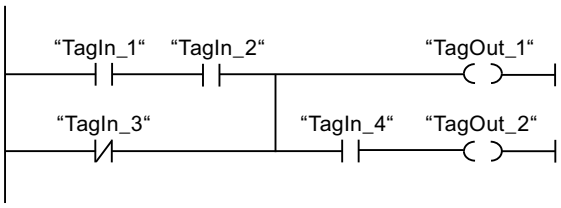
Parameter

The following table shows the parameters of the "Assignment" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand to which the RLO is assigned.

Example

The following example shows how the instruction works:



The "TagOut_1" operand is set when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The signal state of the operand "TagIn_3" is "0".

The "TagOut_2" operand is set when one of the following conditions is fulfilled:

- Operands "TagIn_1", "TagIn_2", and "TagIn_4" have signal state "1".
- The signal state of the "TagIn_3" operand is "0" and the signal state of the "TagIn_4" operand is "1".

See also

Inserting LAD elements (Page 2579)

Inserting operands into LAD instructions (Page 2601)

Editing LAD elements (Page 2594)

Overview of the valid data types (Page 2236)

--(/)--: Negate assignment

Description

The "Negate assignment" instruction inverts the result of logic operation (RLO) and assigns it to the specified operand. When the RLO at the input of the coil is "1", the operand is reset. When the RLO at the input of the coil is "0", the operand is set to signal state "1".

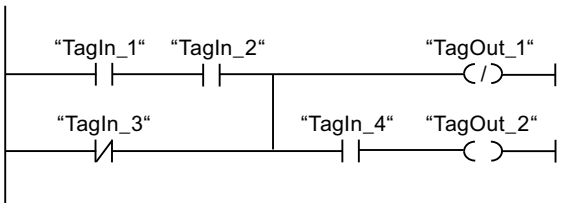
Parameter

The following table shows the parameters of the "Negate assignment" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand to which the RLO is assigned.

Example

The following example shows how the instruction works:



Operand "TagOut_1" is reset when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The signal state of the operand "TagIn_3" is "0".

See also

Inserting LAD elements (Page 2579)

Inserting operands into LAD instructions (Page 2601)

Editing LAD elements (Page 2594)

Overview of the valid data types (Page 2236)

---(R)---: Reset output

Description

You can use the "Reset output" instruction to reset the signal state of a specified operand to "0".

The instruction is only executed if the result of logic operation (RLO) at the input of the coil is "1". If power flows to the coil (RLO = "1"), the specified operand is reset to "0". If the RLO at the input of the coil is "0" (no signal flow to the coil), the signal state of the specified operand remains unchanged.

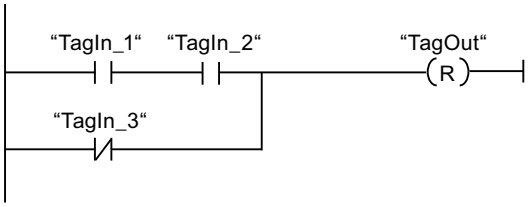
Parameter

The following table shows the parameters of the "Reset output" instruction:

Parameter	Declaration	Data type	Memory area		
			S7-1200	S7-1500	Description
<Operand>	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Operand that is reset when RLO = "1".

Example

The following example shows how the instruction works:



Operand "TagOut" is reset when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The signal state of the operand "TagIn_3" is "0".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

---(S)---: Set output

Description

You can use the "Set output" instruction to set the signal state of a specified operand to "1".

The instruction is only executed if the result of logic operation (RLO) at the input of the coil is "1". If power flows to the coil (RLO = "1"), the specified operand is set to "1". If the RLO at the

input of the coil is "0" (no signal flow to the coil), the signal state of the specified operand remains unchanged.

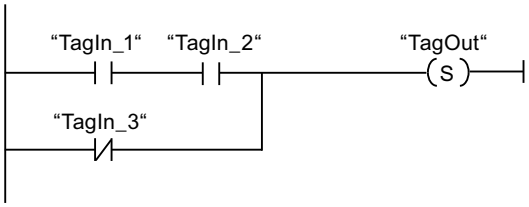
Parameter

The following table shows the parameters of the "Set output" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand which is set with RLO = "1".

Example

The following example shows how the instruction works:



The "TagOut" operand is set when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The signal state of the operand "TagIn_3" is "0".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

SET_BF: Set bit field

Description

You use the instruction "Set bit field" to set multiple bits starting from a certain address.

You determine the number of bits to be set using the value of <Operand1>. The address of the first bit to be set is defined by <Operand2>. If the value of <Operand1> is greater than the number of bits in a selected byte, then the bits of the next byte will be set. The bits remain set until they are explicitly reset, for example, by another instruction.

The instruction is only executed if the result of logic operation (RLO) at the input of the coil is "1". If the RLO at the input of the coil is "0", the instruction does not execute.

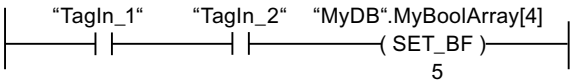
Parameter

The following table shows the parameters of the "Set bit field" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand2>	Output	BOOL	I, Q, M In the case of a DB or an IDB, an element of an ARRAY[..] of BOOL	Pointer to the first bit to be set.
<Operand1>	Input	UINT	Constant	Number of bits to be set.

Example

The following example shows how the instruction works:



If the operands "TagIn_1" and "TagIn_2" have the signal state "1", 5 bits are set starting at the address of the operand "MyDB".MyBoolArray[4]

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

RESET_BF: Reset bit field

Description

You use the "Reset bit field" instruction to reset several bits starting from a certain address.

You specify the number of bits to be reset using the value of <Operand1>. The address of the first bit to be reset is specified by <Operand2>. If the value of <Operand1> is greater than the number of bits in a selected byte, the bits of the next byte will be reset. The bits remain reset until they are explicitly set, for example, by another instruction.

The instruction is only executed if the result of logic operation (RLO) at the input of the coil is "1". If the RLO at the input of the coil is "0", the instruction does not execute.

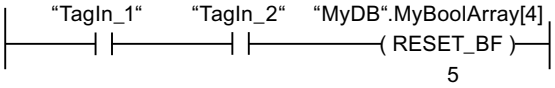
Parameter

The following table shows the parameters of the "Reset bit field" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand2>	Output	BOOL	I, Q, M In the case of a DB or an IDB, an element of an ARRAY[..] of BOOL	Pointer to the first bit to be reset.
<Operand1>	Input	UINT	Constant	Number of bits to be reset.

Example

The following example shows how the instruction works:



If the operands "TagIn_1" and "TagIn_2" have the signal state "1", 5 bits are reset starting at the address of the operand "MyDB".MyBoolArray[4]

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

SR: Set/reset flip-flop

Description

Use the instruction "Set/reset flip-flop" to set or reset the bit of the specified operand, depending on the signal state of the inputs S and R1. If the signal state is "1" at input S and "0" at input R1, the specified operand is set to "1". If the signal state is "0" at input S and "1" at input R1, the specified operand will be reset to "0".

Input R1 takes priority over input S. When the signal state is "1" on both inputs S and R1, the signal state of the specified operand is reset to "0".

The instruction is not executed if the signal state at the two inputs S and R1 is "0". The signal state of the operand then remains unchanged.

The current signal state of the operand is transferred to output Q and can be queried there.

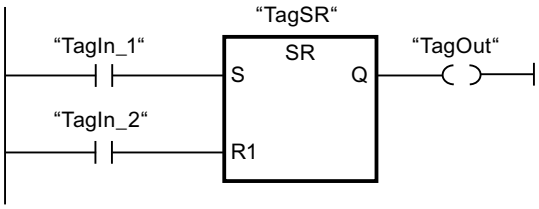
Parameter

The following table shows the parameters of the "Set/reset flip-flop" instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
S	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable setting
R1	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable resetting
<Operand>	InOut	BOOL	I, Q, M, D, L	I, Q, M, D, L	Operand that is set or reset.
Q	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Signal state of the operand

Example

The following example shows how the instruction works:



The operands "TagSR" and "TagOut" are set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The operand "TagIn_2" has the signal state "0".

The operands "TagSR" and "TagOut" are reset when one of the following conditions is fulfilled:

- The operand "TagIn_1" has signal state "0" and the operand "TagIn_2" has signal state "1".
- The operands "TagIn_1" and "TagIn_2" have signal state "1".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

RS: Reset/set flip-flop

Description

You can use the "Reset/set flip-flop" instruction to reset or set the bit of a specified operand based on the signal state of the inputs R and S1. If the signal state is "1" at input R and "0" at

input S1, the specified operand will be reset to "0". If the signal state is "0" at input R and "1" at input S1, the specified operand is set to "1".

Input S1 takes priority over input R. When the signal state is "1" at both inputs R and S1, the signal state of the specified operand is set to "1".

The instruction is not executed if the signal state at the two inputs R and S1 is "0". The signal state of the operand then remains unchanged.

The current signal state of the operand is transferred to output Q and can be queried there.

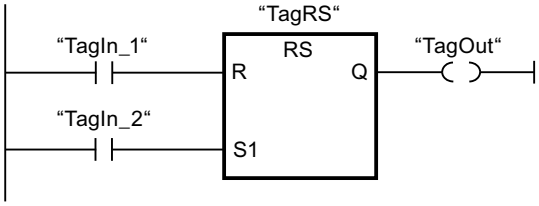
Parameter

The following table shows the parameters of the "Reset/set flip-flop" instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
R	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable resetting
S1	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable setting
<Operand>	InOut	BOOL	I, Q, M, D, L	I, Q, M, D, L	Operand that is reset or set.
Q	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Signal state of the operand

Example

The following example shows how the instruction works:



The operands "TagRS" and "TagOut" are reset when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The operand "TagIn_2" has the signal state "0".

The operands "TagRS" and "TagOut" are set when one of the following conditions is fulfilled:

- The operand "TagIn_1" has signal state "0" and the operand "TagIn_2" has signal state "1".
- The operands "TagIn_1" and "TagIn_2" have signal state "1".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

--|P|--: Scan operand for positive signal edge

Description

You can use the "Scan operand for positive signal edge" instruction to determine if there is a "0" to "1" change in the signal state of a specified operand (<Operand1>). The instruction compares the current signal state of <Operand1> with the signal state of the previous scan that is saved in an edge memory bit (<Operand2>). If the instruction detects a change in the result of logic operation (RLO) from "0" to "1", there is a positive, rising edge.

If a positive edge is detected, the output of the instruction has the signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Specify the operand to be queried (<Operand1>) in the operand placeholder above the instruction. Specify the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This step influences the edge evaluation and the result is therefore no longer unique. The memory area of the edge memory bit must be located in a DB (static area for FB) or in the bit memory area.

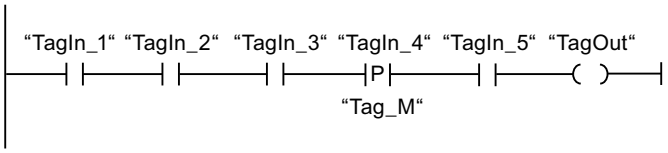
Parameter

The following table shows the parameters of the "Scan operand for positive signal edge" instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand1>	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Signal to be scanned
<Operand2>	InOut	BOOL	I, Q, M, D, L	I, Q, M, D, L	Edge memory bit in which the signal state of the previous query is saved.

Example

The following example shows how the instruction works:



Operand "TagOut" is set when the following conditions are fulfilled:

- The operands "TagIn_1", "TagIn_2", and "TagIn_3" have signal state "1".
- There is a rising edge at operand "TagIn_4". The signal state of the previous scan is stored in the edge memory bit "Tag_M".
- The signal state of the operand "TagIn_5" is "1".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

--|N|--: Scan operand for negative signal edge

Description

You can use the "Scan operand for negative signal edge" instruction to determine if there is a "1" to "0" change in the signal state of a specified operand (<Operand1>). The instruction compares the current signal state of <Operand1> with the signal state of the previous scan that is saved in an edge memory bit <Operand2>. If the instruction detects a change in the result of logic operation (RLO) from "1" to "0", there is a negative, falling edge.

If a negative signal edge is detected, the output of the instruction has the signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Specify the operand to be queried (<Operand1>) in the operand placeholder above the instruction. Specify the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This step influences the edge evaluation and the result is therefore no longer unique. The memory area of the edge memory bit must be located in a DB (static area for FB) or in the bit memory area.

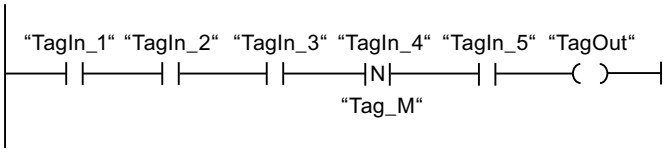
Parameter

The following table shows the parameters of the "Scan operand for negative signal edge" instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand1>	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Signal to be scanned
<Operand2>	InOut	BOOL	I, Q, M, D, L	I, Q, M, D, L	Edge memory bit in which the signal state of the previous query is saved.

Example

The following example shows how the instruction works:



Operand "TagOut" is set when the following conditions are fulfilled:

- The operands "TagIn_1", "TagIn_2", and "TagIn_3" have signal state "1".
- There is a negative signal edge at operand "TagIn_4". The signal state of the previous scan is stored in the edge memory bit "Tag_M".
- The signal state of the operand "TagIn_5" is "1".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

--(P)--: Set operand on positive signal edge

Description

You can use the "Set operand on positive signal edge" instruction to set a specified operand (<Operand1>) when there is a "0" to "1" change in the result of logic operation (RLO). The instruction compares the current RLO with the RLO from the previous query, which is saved in the edge memory bit (<Operand2>). If the instruction detects a change in the RLO from "0" to "1", there is a positive signal edge.

When a positive signal edge is detected, <Operand1> is set to signal state "1" for one program cycle. In all other cases, the operand has the signal state "0".

Specify the operand to be set (<Operand1>) in the operand placeholder above the instruction. Specify the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This step influences the edge evaluation and the result is therefore no longer unique. The memory area of the edge memory bit must be located in a DB (static area for FB) or in the bit memory area.

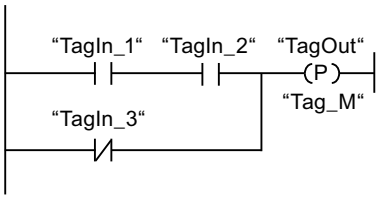
Parameter

The following table shows the parameters of the "Set operand on positive signal edge" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Output	BOOL	I, Q, M, D, L	Operand which is set by a positive edge.
<Operand2>	InOut	BOOL	I, Q, M, D, L	Edge memory bit

Example

The following example shows how the instruction works:



Operand "TagOut" is set for one program cycle, when the signal state at the input of the coil switches from "0" to "1" (positive signal edge). In all other cases, the operand "TagOut" has the signal state "0".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

--(N)--: Set operand on negative signal edge

Description

You can use the "Set operand on negative signal edge" instruction to set a specified operand (<Operand1>) when there is a "1" to "0" change in the result of logic operation (RLO). The instruction compares the current RLO with the RLO from the previous query, which is saved in the edge memory bit (<Operand2>). If the instruction detects a change in the RLO from "1" to "0", there is a negative edge.

When a negative signal edge is detected, <Operand1> is set to signal state "1" for one program cycle. In all other cases, the operand has the signal state "0".

Specify the operand to be set (<Operand1>) in the operand placeholder above the instruction. Specify the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This step influences the edge evaluation and the result is therefore no longer unique. The memory area of the edge memory bit must be located in a DB (static area for FB) or in the bit memory area.

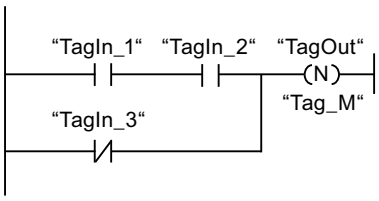
Parameter

The following table shows the parameters of the "Set operand on negative signal edge" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Output	BOOL	I, Q, M, D, L	Operand which is set by a negative edge.
<Operand2>	InOut	BOOL	I, Q, M, D, L	Edge memory bit

Example

The following example shows how the instruction works:



Operand "TagOut" is set for one program cycle, when the signal state at the input of the coil switches from "1" to "0" (negative signal edge). In all other cases, the operand "TagOut" has the signal state "0".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

P_TRIG: Scan RLO for positive signal edge

Description

Use the "Scan RLO for positive signal edge" instruction to query a "0" to "1" change in the signal state of the result of logic operation (RLO). The instruction compares the current signal state of the RLO with the signal state of the previous query, which is saved in an edge memory bit (<operand>). If the instruction detects a change in the RLO from "0" to "1", there is a positive signal edge.

If a positive edge is detected, the output of the instruction has the signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This step influences the edge evaluation and the result is therefore no longer unique. The memory area of the edge memory bit must be located in a DB (static area for FB) or in the bit memory area.

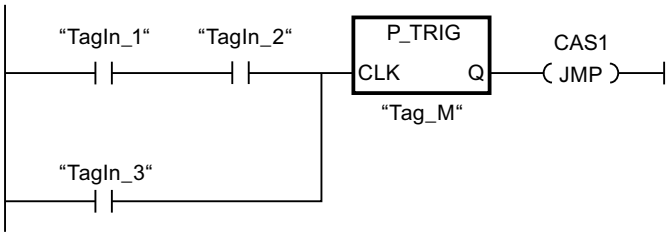
Parameter

The following table shows the parameters of the "Scan RLO for positive signal edge" instruction:

Parameter	Declaration	Data type	Memory area	Description
CLK	Input	BOOL	I, Q, M, D, L	Current RLO
<Operand>	InOut	BOOL	M, D	Edge memory bit in which the RLO of the previous query is saved.
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:



The RLO of the previous query is saved in the edge memory bit "Tag_M". If a "0" to "1" change is detected in the signal state of the RLO, the program jumps to jump label CAS1.

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

N_TRIG: Scan RLO for negative signal edge

Description

Use the "Scan RLO for negative signal edge" instruction to query a "1" to "0" change in the signal state of the result of logic operation (RLO). The instruction compares the current signal state of the RLO with the signal state of the previous query, which is saved in an edge memory bit (<operand>). If the instruction detects a change in the RLO from "1" to "0", there is a negative edge.

If a negative signal edge is detected, the output of the instruction has the signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This step influences the edge evaluation and the result is therefore no longer unique. The memory area of the edge memory bit must be located in a DB (static area for FB) or in the bit memory area.

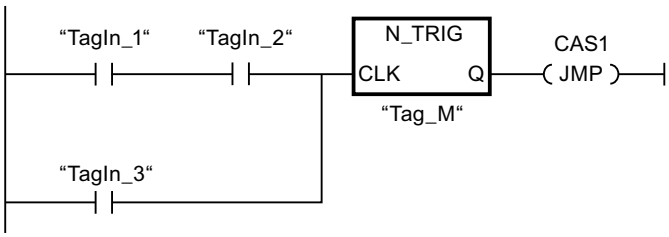
Parameter

The following table shows the parameters of the "Scan RLO for negative signal edge" instruction:

Parameters	Declaration	Data type	Memory area	Description
CLK	Input	BOOL	I, Q, M, D, L	Current RLO
<Operand>	InOut	BOOL	M, D	Edge memory bit in which the RLO of the previous query is saved.
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:



The RLO of the previous query is saved in the edge memory bit "Tag_M". If a "1" to "0" change is detected in the signal state of the RLO, the program jumps to jump label CAS1.

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

R_TRIG: Set tag on positive signal edge

Description

You can use the "Set tag on positive signal edge" instruction to set a specified tag in the instance DB when there is a "0" to "1" change in the result of logic operation (RLO). The instruction compares the current RLO at the input CLK with the RLO from the previous query, which is saved in the specified instance DB. If the instruction detects a change in the RLO from "0" to "1", there is a positive signal edge.

If a positive edge is detected, the tag in the instance DB is set to signal state "1" and the output Q returns the signal state "1" In all other cases, the signal state at the output of the instruction is "0".

When you insert the instruction in the program, the "Call options" dialog opens automatically. In this dialog you can specify whether the edge memory bit is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

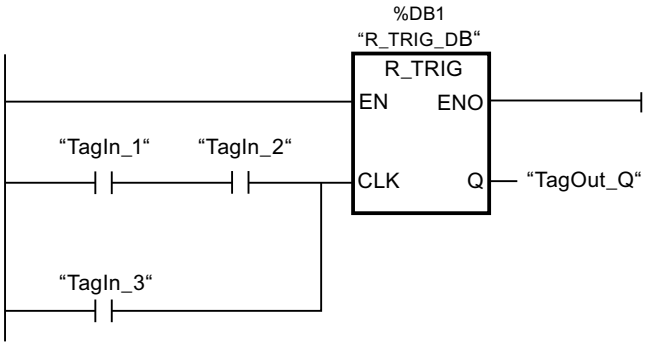
Parameter

The following table shows the parameters of the instruction "Set tag on positive signal edge":

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
CLK	Input	BOOL	I, Q, M, D, L or constant	Incoming signal, the edge of which is to be queried.
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:



The RLO of the preceding query is saved in the instance DB "R_TRIG_DB". If a change is the signal state of the RLO from "0" to "1" is detected in the operands "TagIn_1" and "TagIn_2" or in the operand "TagIn_3", the output "TagOut_Q" has signal state "1".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

F_TRIG: Set tag on negative signal edge

Description

You can use the "Set tag on negative signal edge" instruction to set a specified tag in the instance DB when there is a "1" to "0" change in the result of logic operation (RLO). The instruction compares the current RLO at the input CLK with the RLO from the previous query, which is saved in the specified instance DB. If the instruction detects a change in the RLO from "1" to "0", there is a negative edge.

If a negative edge is detected, the tag in the instance DB is set to signal state "1" and the output Q returns the signal state "1" In all other cases, the signal state at the output of the instruction is "0".

When you insert the instruction in the program, the "Call options" dialog opens automatically. In this dialog you can specify whether the edge memory bit is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

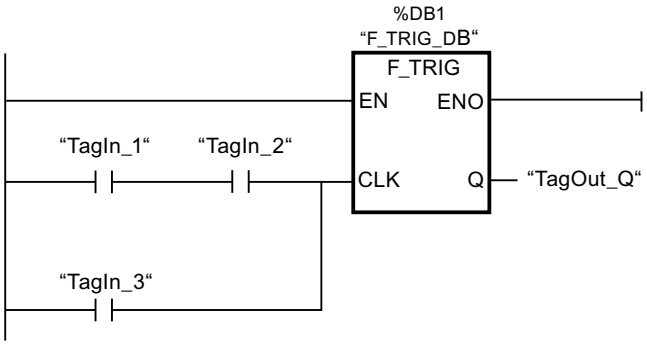
Parameter

The following table shows the parameters of the instruction "Set tag on negative signal edge":

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
CLK	Input	BOOL	I, Q, M, D, L or constant	Incoming signal, the edge of which is to be queried.
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:



The RLO of the preceding query is saved in the instance DB "F_TRIG_DB". If a change is the signal state of the RLO from "1" to "0" is detected in the operands "TagIn_1" and "TagIn_2" or in the operand "TagIn_3", the output "TagOut_Q" has signal state "1".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Timer operations

IEC Timers

TP: Generate pulse

Description

You can use the "Generate pulse" instruction to set the output Q for a programmed duration. The instruction is started when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. Output Q is set for the duration PT, regardless of the subsequent course of the input signal. Even if a new positive signal edge is detected, the signal state at the output Q is not affected as long as the PT time duration is running.

You can scan the current time value at the ET output. The time value starts at T#0s and ends when the value of duration PT is reached. When the duration PT is reached and the signal state at input IN is "0", the ET output is reset.

Each call of the "Generate pulse" instruction must be assigned to an IEC timer in which the instruction data is stored.

Note

If the timer is not called in the program because it is skipped, for example, the ET output returns a constant value as soon as the timer has expired.

For S7-1200 CPU

An IEC timer is a structure of the data type IEC_TIMER or TP_TIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TP_TIME, TP_LTIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

An IEC timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TP_TIME or TP_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TP_TIME, TP_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when the instruction is called and also each time the outputs Q or ET are accessed.

The execution of the "Generate pulse" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

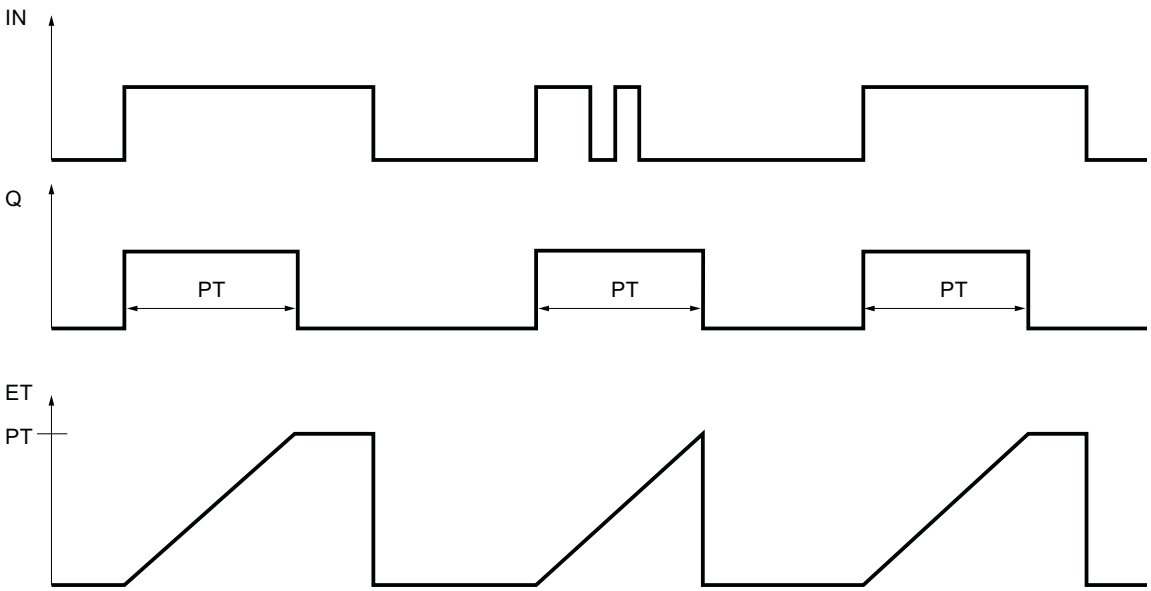
The following table shows the parameters of the "Generate pulse" instruction:

Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
IN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, P	Start input
PT	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	I, Q, M, D, L, P or constant	Duration of the pulse The value of the PT parameter must be positive.
Q	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, P	Pulse output
ET	Output	TIME	TIME, LTIME	I, Q, M, D, L	I, Q, M, D, L, P	Current time value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Generate pulse" instruction:



See also

Overview of the valid data types (Page 2236)

Call function blocks as single or multi-instances (Page 2182)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

TON: Generate on-delay

Description

You can use the "Generate on-delay" instruction to delay setting of the Q output by the programmed duration PT. The instruction is started when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. When the duration PT expires, the output Q has the signal state "1". Output Q remains set as long as the start input is still "1". When the signal state at the start input changes from "1" to "0", the Q output is reset. The timer function is started again when a new positive signal edge is detected at the start input.

The current time value can be queried at the ET output. The time value starts at T#0s and ends when the value of duration PT is reached. The ET output is reset as soon as the signal state at the IN input changes to "0".

Each call of the "Generate on-delay" instruction must be assigned to an IEC timer in which the instruction data is stored.

Note

If the timer is not called in the program because it is skipped, for example, the ET output returns a constant value as soon as the timer has expired.

For S7-1200 CPU

An IEC timer is a structure of the data type IEC_TIMER or TON_TIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TON_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

An IEC timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TON_TIME or TON_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TON_TIME, TON_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when the instruction is called and also each time the outputs Q or ET are accessed.

The execution of the "Generate on-delay" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

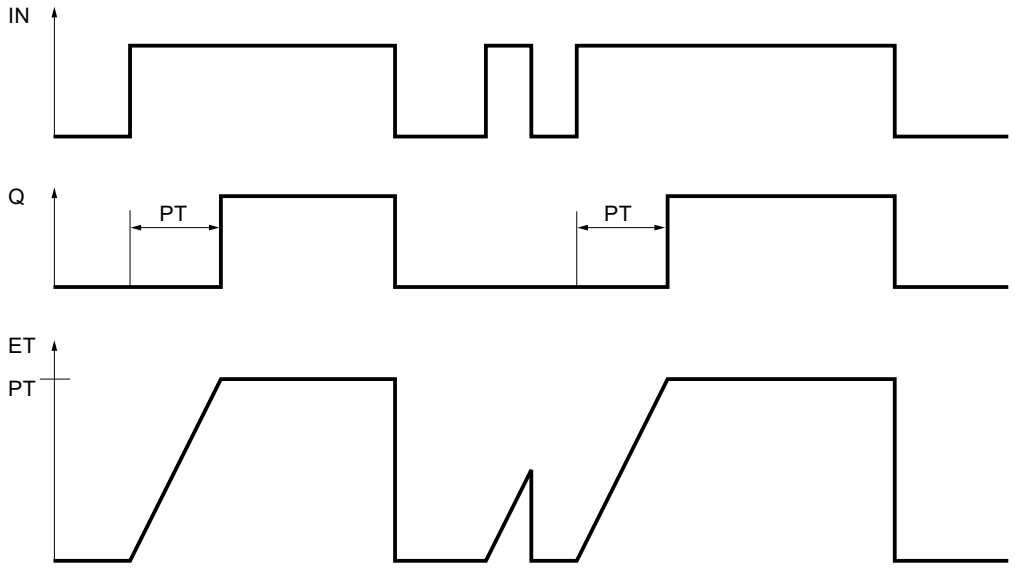
The following table shows the parameters of the "Generate on-delay" instruction:

Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
IN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, P	Start input
PT	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	I, Q, M, D, L, P or constant	Duration of the on-delay The value of the PT parameter must be positive.
Q	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, P	Output that is set when the time PT expires.
ET	Output	TIME	TIME, LTIME	I, Q, M, D, L	I, Q, M, D, L, P	Current time value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Generate on-delay" instruction:



See also

Overview of the valid data types (Page 2236)

Call function blocks as single or multi-instances (Page 2182)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Selecting the data type of a LAD element (Page 2583)

Inserting operands into LAD instructions (Page 2601)

TOF: Generate off-delay

Description

You can use the "Generate off-delay" instruction to delay resetting of the Q output by the programmed duration PT. The Q output is set when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive signal edge). When the signal state at input IN changes back to "0", the programmed time PT starts. Output Q remains set as long as the duration PT is running. When duration PT expires, the Q output is reset. If the signal state at input IN changes to "1" before the PT time duration expires, the timer is reset. The signal state at the output Q continues to be "1".

The current time value can be queried at the ET output. The time value starts at T#0s and ends when the value of duration PT is reached. When the time duration PT expires, the ET output remains set to the current value until the IN input changes back to "1". If input IN switches to "1" before the duration PT has expired, the ET output is reset to the value T#0s.

Each call of the "Generate off-delay" instruction must be assigned to an IEC timer in which the instruction data is stored.

Note

If the timer is not called in the program because it is skipped, for example, the ET output returns a constant value as soon as the timer has expired.

For S7-1200 CPU

An IEC timer is a structure of the data type IEC_TIMER or TOF_TIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TOF_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

An IEC timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TOF_TIME or TOF_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TOF_TIME, TOF_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when the instruction is called and also each time the outputs Q or ET are accessed.

The execution of the "Generate off-delay" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

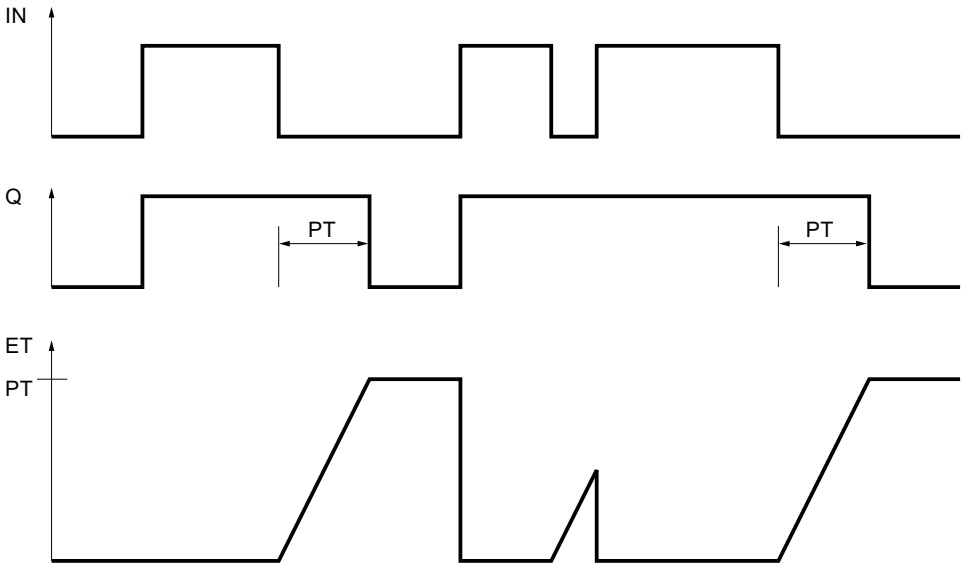
The following table shows the parameters of the "Generate off-delay" instruction:

Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
IN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, P	Start input
PT	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	I, Q, M, D, L, P or constant	Duration of the off delay The value of the PT parameter must be positive.
Q	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, P	Output that is reset when the timer PT expires.
ET	Output	TIME	TIME, LTIME	I, Q, M, D, L	I, Q, M, D, L, P	Current time value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Generate off-delay" instruction:



See also

Overview of the valid data types (Page 2236)

Call function blocks as single or multi-instances (Page 2182)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Selecting the data type of a LAD element (Page 2583)

Inserting operands into LAD instructions (Page 2601)

TONR: Time accumulator

Description

The "Time accumulator" instruction is used to accumulate time values within a period set by the PT parameter. When the signal state at input IN changes from "0" to "1" (positive signal edge), the instruction executes and the duration PT starts. While the duration PT is running, the time values are accumulated that are recorded when the IN input has signal state "1". The accumulated time is written to output ET and can be queried there. When the duration PT expires, the output Q has the signal state "1". The Q parameter remains set to "1", even when the signal state at the IN parameter changes from "1" to "0" (negative signal edge).

The R input resets the outputs ET and Q regardless of the signal state at the start input.

Each call of the "Time accumulator" instruction must be assigned to an IEC timer in which the instruction data is stored.

For S7-1200 CPU

An IEC timer is a structure of the data type IEC_TIMER or TONR_TIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TONR_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

An IEC timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TONR_TIME or TONR_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TONR_TIME, TONR_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when the instruction is called and also each time the outputs Q or ET are accessed.

The execution of the Time accumulator" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Time accumulator" instruction:

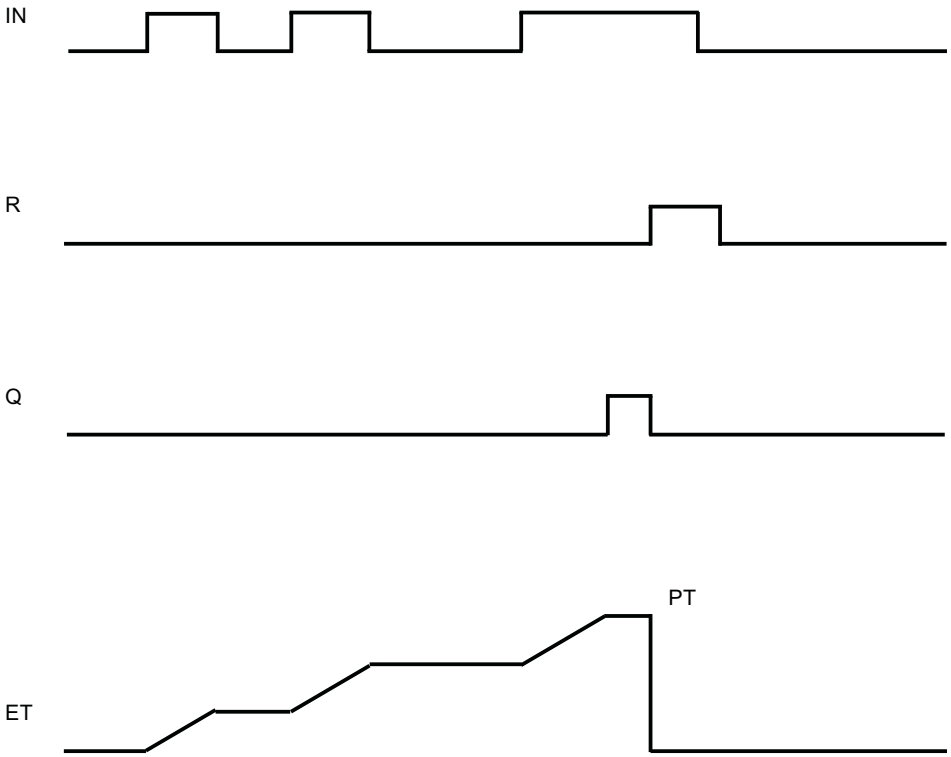
Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
IN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, P	Start input
R	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, P or constant	Reset input
PT	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	I, Q, M, D, L, P or constant	Maximum duration of time recording The value of the PT parameter must be positive.

Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
Q	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, P	Output that is set when the time PT expires.
ET	Output	TIME	TIME, LTIME	I, Q, M, D, L	I, Q, M, D, L, P	Accumulated time

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Time accumulator" instruction:



See also

Overview of the valid data types (Page 2236)

Call function blocks as single or multi-instances (Page 2182)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Selecting the data type of a LAD element (Page 2583)

Inserting operands into LAD instructions (Page 2601)

---(TP)---: Start pulse timer

Description

Use the "Start pulse timer" instruction to start an IEC timer with a specified duration as pulse. The IEC timer is started when the result of logic operation (RLO) changes from "0" to "1" (positive signal edge). The IEC timer runs for the specified duration regardless of any subsequent changes in the RLO. The run of the IEC timer is also not affected by the detection of a new positive signal edge. As long as the IEC timer is running, the querying of the timer status for "1" returns the signal state "1". When the IEC timer has expired, the timer status returns the signal state "0".

Note

You can start and query the IEC timer at various execution levels, as each querying of the outputs Q or ET updates the IEC_TIMER structure.

For S7-1200 CPU

The instruction "Start pulse timer" stores its data in a structure of the data type IEC_TIMER or TP_TIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TP_LTIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

The instruction "Start pulse timer" stores its data in a structure of the data type IEC_TIMER, IEC_LTIMER, TP_TIME or TP_LTIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TP_TIME, TP_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated both when the instruction is called and also each time the assigned timer is accessed.

The current timer status is stored in the Q structure component of the IEC timer. You can use a normally open contact to query timer status for "1" or a normally closed contact for "0". The query on Q or ET (for example, "MyTimer".Q or "MyTimer".ET) updates the IEC_TIMER structure.

The execution of the "Start pulse timer" instruction assumes a preceding logic operation. It can be placed only at the end of the network.

Parameter

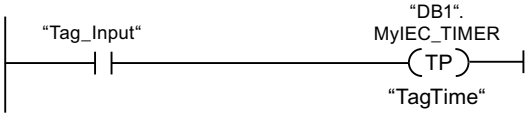
The following table shows the parameters of the "Start pulse timer" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<Time duration>	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	Duration with which the IEC timer runs.
<IEC timer>	InOut	IEC_TIMER, TP_TIME	IEC_TIMER, IEC_LTIMER, TP_TIME, TP_LTIME	D, L	IEC timer that is started.

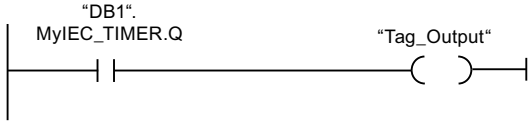
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "Start pulse timer" instruction is executed when the signal state of the operand "Tag_Input" changes from "0" to "1". The timer "DB1".MyIEC_TIMER is started for the time stored in the operand "TagTime".



As long as the timer "DB1". MyIEC_TIMER is running, the timer status ("DB1".MyIEC_TIMER.Q) has signal state "1" and the operand "Tag_Output" is set. When the IEC timer has expired, the signal state of the time status changes back to "0" and the "Tag_Output" operand is reset.

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

---(TON)---: Start on-delay timer

Description

Use the "Start on-delay timer" instruction to start an IEC timer with a specified duration as on-delay. The IEC timer is started when the result of logic operation (RLO) changes from "0" to "1" (positive signal edge). The IEC timer runs for the specified time. The output returns the signal state "1" if the RLO at the input of the instruction has the signal state "1". If the RLO changes to "0" before the time expires, the IEC timer is reset. In this case, querying the timer status for "1" returns signal state "0". The IEC timer restarts when the next positive signal edge is detected at the input of the instruction.

Note

You can start and query the IEC timer at various execution levels, as each querying of the outputs Q or ET updates the IEC_TIMER structure.

For S7-1200 CPU

The instruction "Start on-delay timer" stores its data in a structure of the data type IEC_TIMER or TON_TIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TON_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

The "Start on-delay timer" instruction stores its data in a structure of the data type IEC_TIMER, IEC_LTIMER, TON_TIME or TON_LTIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TON_TIME, TON_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated both when the instruction is called and also each time the assigned timer is accessed.

The current timer status is stored in the ET structure component of the IEC timer. You can use a normally open contact to query timer status for "1" or a normally closed contact for "0". The query on Q or ET (for example, "MyTimer".Q or "MyTimer".ET) updates the IEC_TIMER structure.

The execution of the "Start on-delay timer" instruction assumes a preceding logic operation. It can be placed only at the end of the network.

Parameter

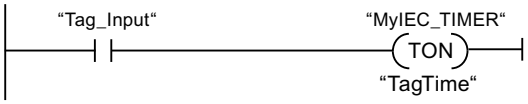
The following table shows the parameters of the "Start on-delay timer" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<Time duration>	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	Duration with which the IEC timer runs.
<IEC timer>	InOut	IEC_TIMER, TON_TIME	IEC_TIMER, IEC_LTIMER, TON_TIME, TON_LTIME	D, L	IEC timer that is started.

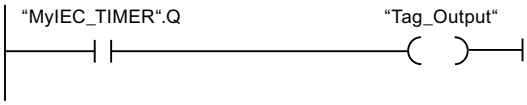
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "Start on-delay timer" instruction is executed when the signal state of the operand "Tag_Input" changes from "0" to "1". The "MyIEC_TIMER timer is started for the time stored in the "TagTime" operand.



If the timer "MyIEC_TIMER" has expired and the operand "Tag_Input" has the signal state "1", querying the timer status ("MyIEC_TIMER).Q) returns signal state "1" and the "Tag_Output" operand is set. When the signal state of the operand "Tag_Input" changes to "0", the querying of the timer status returns the signal state "0" and the operand "Tag_Output" is reset.

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

---(TOF)---: Start off-delay timer

Description

Use the "Start off-delay timer" instruction to start an IEC timer with a specified duration as on-delay. The query of the timer status for "1" returns the signal state "0" if the result of logic operation (RLO) at the input of the instruction has the signal state "1". When the RLO changes from "1" to "0" (negative signal edge), the IEC timer starts with the specified time. The timer status remains at signal state "1" as long as the IEC timer is running. When the timer has run out and the RLO at the input of the instruction has the signal state "0", the timer status is set to the signal state "0". If the RLO changes to "1" before the time expires, the IEC timer is reset and the timer status remains at signal state "1".

Note

You can start and query the IEC timer at various execution levels, as each querying of the outputs Q or ET updates the IEC_TIMER structure.

For S7-1200 CPU

The instruction "Start off-delay timer" stores its data in a structure of the data type IEC_TIMER or TOF_TIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TOF_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

The "Start off-delay timer" instruction stores its data in a structure of the data type IEC_TIMER, IEC_LTIMER, TOF_TIME or TOF_LTIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TOF_TIME, TOF_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated both when the instruction is called and also each time the assigned timer is accessed.

The current timer status is stored in the ET structure component of the IEC timer. You can use a normally open contact to query timer status for "1" or a normally closed contact for "0". The query on Q or ET (for example, "MyTimer".Q or "MyTimer".ET) updates the IEC_TIMER structure.

The execution of the "Start off-delay timer" instruction assumes a preceding logic operation. It can be placed only at the end of the network.

Parameters

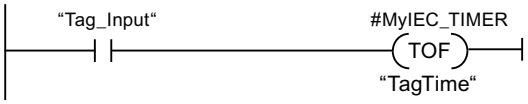
The following table shows the parameters of the "Start off-delay timer" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<Time duration>	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	Duration with which the IEC timer runs.
<IEC timer>	InOut	IEC_TIMER, TOF_TIME	IEC_TIMER, IEC_LTIMER, TOF_TIME, TOF_LTIME	D, L	IEC timer that is started.

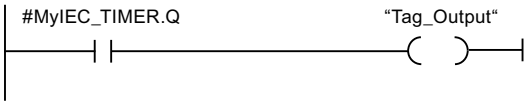
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "Start off-delay timer" instruction is executed when the signal state of the operand "Tag_Input" changes from "1" to "0". The timer #MyIEC_TIMER is started for the time stored in the operand "TagTime".



As long as timer #MyIEC_TIMER is running, the query of the time status (#MyIEC_TIMER.Q) returns the signal state "1" and operand "Tag_Output" is set. If the timer has expired and the operand "Tag_Input" has the signal state "0", the query of the timer status returns the signal state "0". If the signal state of the operand "Tag_Input" changes to "1" before timer #MyIEC_TIMER expires, the timer is reset. When the signal state of the operand "Tag_Input" is "1", the query of the timer status returns the signal state "1".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

---(TONR)---: Time accumulator

Description

You can use the "Time accumulator" instruction to record how long the signal is at the input of instruction "1". The instruction is started when the result of logic operation (RLO) changes from "0" to "1" (positive signal edge). The time is recorded as long at the RLO is "1". If the RLO changes to "0", the instruction is halted. If the RLO changes back to "1", the time recording is continued. The query of the timer status for "1" returns the signal state "1" if the recorded time exceeds the value of the specified duration and the RLO at the input of coil is "1".

The timer status and the currently expired timer can be reset to "0" using the "Reset timer" instruction.

Note

You can start and query the IEC timer at various execution levels, as each querying of the outputs Q or ET updates the IEC_TIMER structure.

For S7-1200 CPU

The "Time accumulator" instruction stores its data in a structure of the data type IEC_TIMER or TONR_TIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TONR_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

The "Time accumulator" instruction stores its data in a structure of the data type IEC_TIMER, IEC_LTIMER, TONR_TIME or TONR_LTIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TONR_TIME, TONR_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated both when the instruction is called and also each time the assigned timer is accessed.

The current timer status is stored in the ET structure component of the IEC timer. You can use a normally open contact to query timer status for "1" or a normally closed contact for "0". The query on Q or ET (for example, "MyTimer".Q or "MyTimer".ET) updates the IEC_TIMER structure.

The execution of the Time accumulator" instruction requires a preceding logic operation. It can be placed only at the end of the network.

Parameter

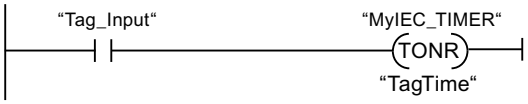
The following table shows the parameters of the "Time accumulator" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<Time duration>	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	Duration with which the IEC timer runs.
<IEC timer>	InOut	IEC_TIMER, TONR_TIME	IEC_TIMER, IEC_LTIMER, TONR_TIME, TONR_LTIME	D, L	IEC timer that is started.

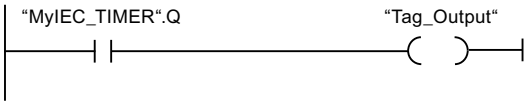
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "Time accumulator" instruction executes on a positive signal edge in the RLO. The time is recorded as long as the operand "Tag_Input" has the signal state "1".



If the recorded time exceeds the value of the operand "TagTime", then the query of the timer status ("MyIEC_TIMER".Q) will return the signal state "1" and the operand "Tag_Output" will be set.

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

---(RT)---: Reset timer

Description

You can use the "Reset timer" instruction to reset an IEC timer to "0". The instruction is only executed if the result of logic operation (RLO) at the input of the coil is "1". If current is flowing to the coil (RLO is "1"), the structure components of the timer in the specified data block are reset to "0". If the RLO at the input of the instruction is "0", the timer remains unchanged.

The instruction does not influence the RLO. The RLO at the input of the coil is sent directly to the output of the coil.

You assign the "Reset timer" instruction an IEC timer that has been declared in the program.

The instruction data is updated only when the instruction is called and not each time the assigned IEC timer is accessed. Querying the data is only identical from the call of the instruction to the next call of the instruction.

Parameters

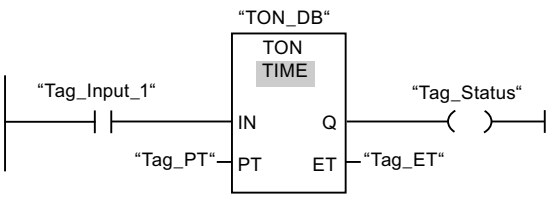
The following table shows the parameters of the "Reset timer" instruction":

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<IEC timer>	Output	IEC_TIMER, TP_TIME, TON_TIME, TOF_TIME, TONR_TIME	IEC_TIMER, IEC_LTIMER, TP_TIME, TP_LTIME, TON_TIME, TON_LTIME, TOF_TIME, TOF_LTIME, TONR_TIME, TONR_LTIME	D, L	IEC timer that is reset.

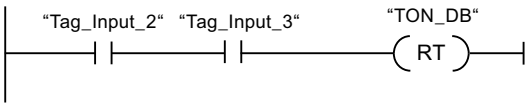
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "Generate on-delay" instruction executes when the signal state of the "Tag_Input_1" operand changes from "0" to "1". The timer stored in the "TON_DB" instance data block starts running for the time duration specified by operand "Tag_PT".



If operands "Tag_Input_2" and "Tag_Input_3" have the signal state "1", the "Reset timer" instruction is executed and the timer stored in the "TON_DB" data block.

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

---(PT)---: Load time duration

Description

You can use the "Load time duration" instruction to set the time for an IEC timer. The instruction is executed in every cycle when the result of logic operation (RLO) at the input of the instruction has the signal state "1". The instruction writes the specified time to the structure of the specified IEC timer.

Note

If the specified IEC timer is running while the instruction executes, the instruction overwrites the current time of the specified IEC timer. This can change the timer status of the IEC timer.

You assign an IEC timer declared in the program to the "Load time duration" instruction.

The instruction data is updated only when the instruction is called and each time the assigned IEC timer is accessed. The query on Q or ET (for example, "MyTimer".Q or "MyTimer".ET) updates the IEC_TIMER structure.

Parameter

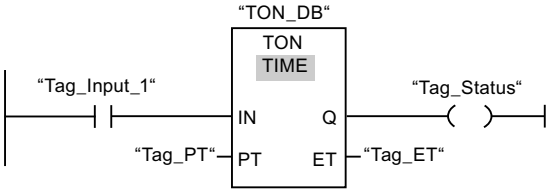
The following table shows the parameters of the "Load time duration" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<Time duration>	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	Duration with which the IEC timer runs.
<IEC timer>	Output	IEC_TIMER, TP_TIME, TON_TIME, TOF_TIME, TONR_TIME	IEC_TIMER, IEC_LTIMER, TP_TIME, TP_LTIME, TON_TIME, TON_LTIME, TOF_TIME, TOF_LTIME, TONR_TIME, TONR_LTIME	D, L	IEC timer, the duration of which is set.

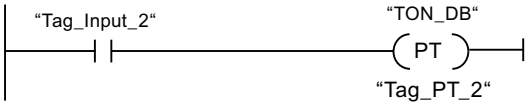
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "Generate on-delay" instruction executes when the signal state of the "Tag_Input_1" operand changes from "0" to "1". The IEC timer stored in the instance data block "TON_DB" is started with the time duration that is specified by the operand "Tag_PT".



The "Load time duration" instruction is executed when the operand "Tag_Input_2" has the signal state "1". The instruction writes the time duration "Tag_PT_2" in the instance data block "TON_DB" and at the same time overwrites the value of the operand "Tag_PT" within the data block. The signal state of the timer status may therefore change at the next query or when "MyTimer".Q or "MyTimer".ET are accessed.

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

SIMATIC Timers

S_PULSE: Assign pulse timer parameters and start

Description

The "Assign pulse timer parameters and start" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV) as soon as the signal state at input S is "1". If the signal state at input S changes to "0" before the programmed duration expires, the timer is stopped. In this case, the signal state at output Q is "0".

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed. The current time value is output BI-coded at output BI and BCD-coded at output BCD.

If the timer is running and the signal state at input R changes to "1" then the current time value and the time base are also set to zero. If the timer is not running, the signal state "1" at the R input has no effect.

The "Assign pulse timer parameters and start" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

The instruction data is updated at every access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

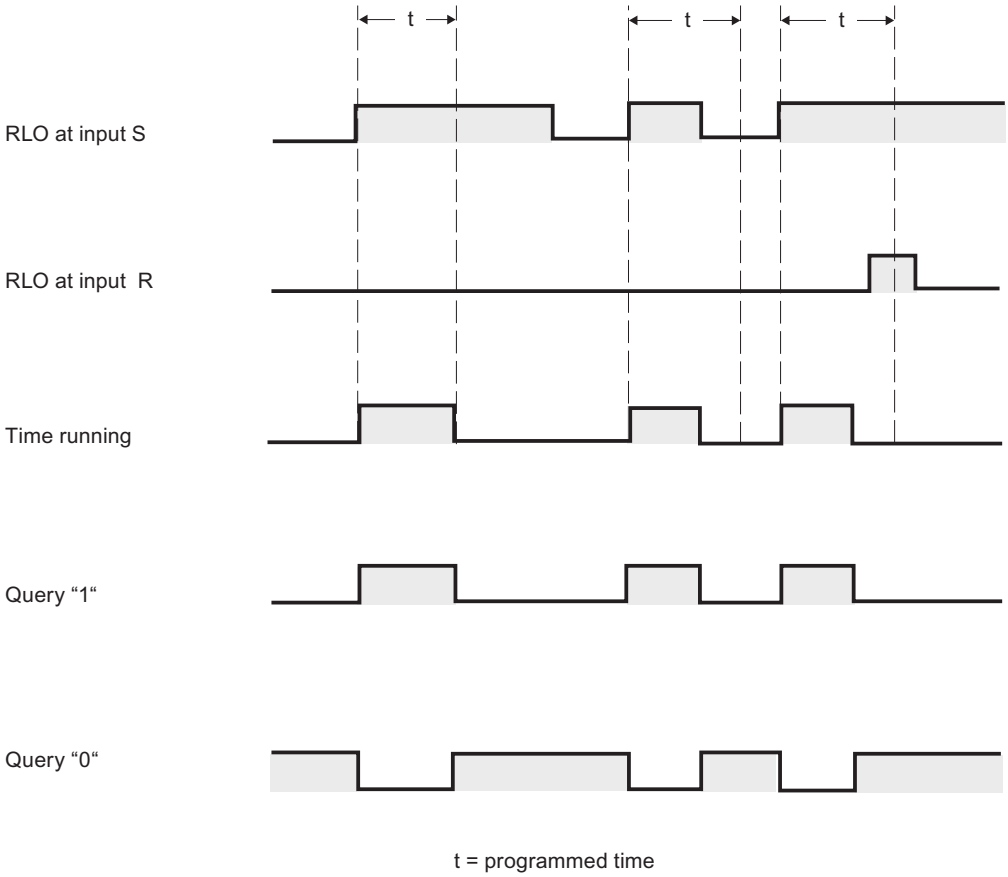
The following table shows the parameters of the instruction "Assign pulse timer parameters and start":

Parameter	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Preset time value
R	Input	BOOL	I, Q, M, T, C, D, L, P or constant	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (BI-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the timer

For additional information on valid data types, refer to "See also".

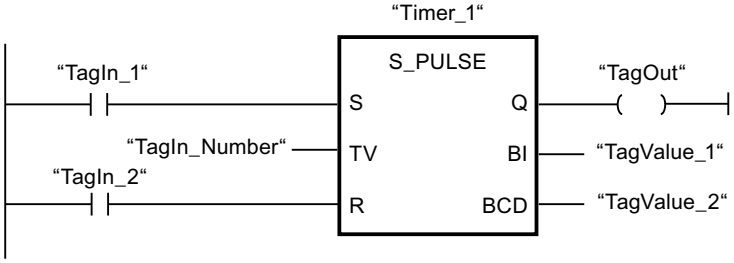
Pulse timing diagram

The following figure shows the pulse timing diagram of the instruction "Assign pulse timer parameters and start":



Example

The following example shows how the instruction works:



The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "0" to "1". The timer expires with the time value of the operand "TagIn_Number" as long as the operand "TagIn_1" has the signal state "1". If the signal state of the operand "TagIn_1" changes from "1" to "0" before the timer expires, the timer "Timer_1" is stopped. The operand "TagOut" is reset to "0" in this case.

The operand "TagOut" has the signal state "1" as long as the timer is running and the operand "TagIn_1" has the signal state "1". When the timer expires or is reset, the operand "TagOut" is reset to "0".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

L: Load timer value (Page 3929)

S_PEXT: Assign extended pulse timer parameters and start

Description

The "Assign extended pulse timer parameters and start" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV), even if the signal state at input S changes to "0". As long as the timer is running, the output Q has the signal state "1". When the timer expires, the output Q is reset to "0". If the signal state at input S changes from "0" to "1" while the timer is running, the timer is restarted with the duration programmed at input TV.

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed. The current time value is output BI-coded at output BI and BCD-coded at output BCD.

If the timer is running and the signal state at input R changes to "1" then the current time value and the time base are also set to zero. If the timer is not running, the signal state "1" at the R input has no effect.

The "Assign extended pulse timer parameters and start" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

The instruction data is updated at every access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameter

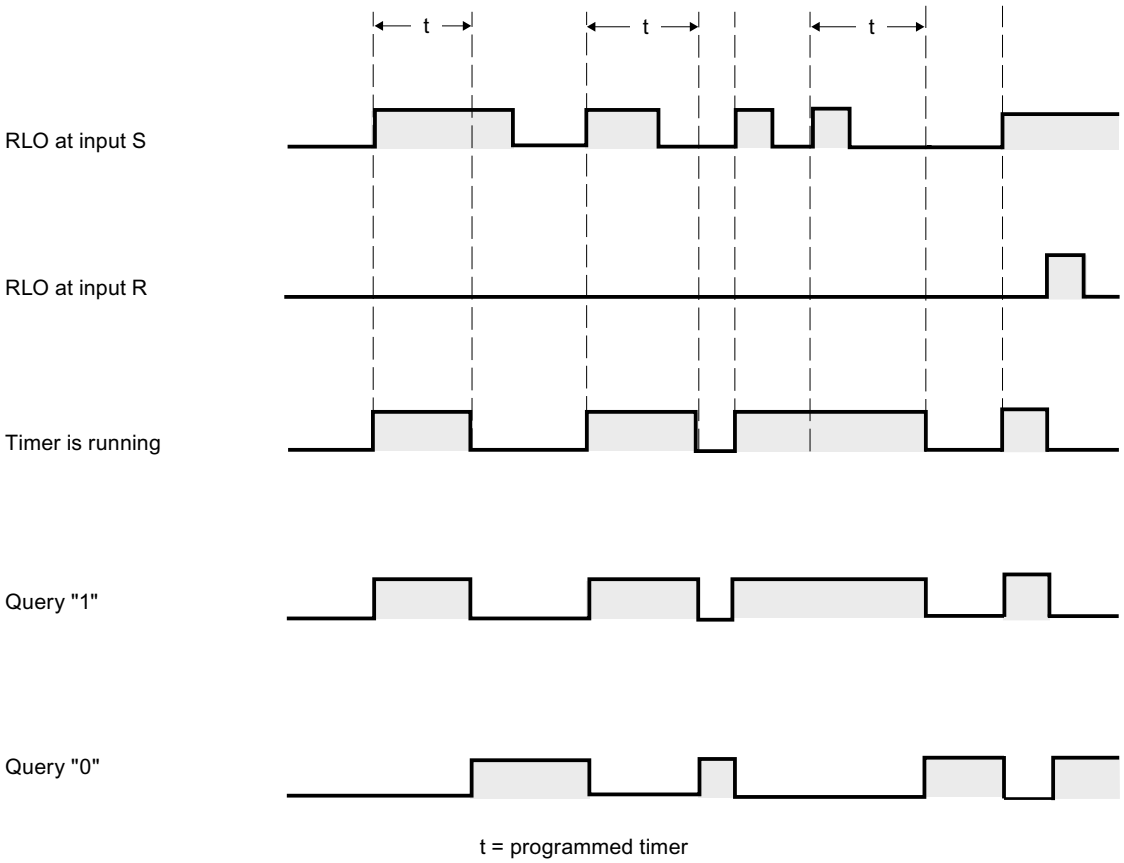
The following table shows the parameters of the instruction "Assign extended pulse timer parameters and start":

Parameter	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Preset time value
R	Input	BOOL	I, Q, M, T, C, D, L, P or constant	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (BI-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the timer

For additional information on valid data types, refer to "See also".

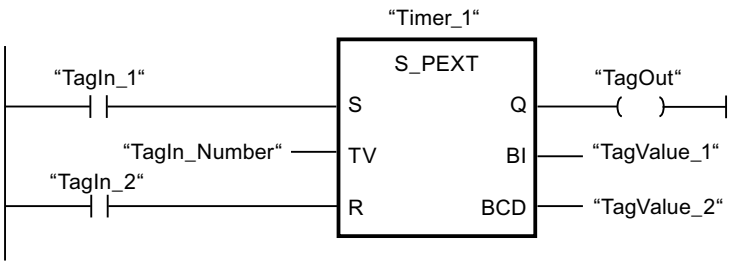
Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign extended pulse timer parameters and start" instruction:



Example

The following example shows how the instruction works:



The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "0" to "1". The timer expires with the time value of the operand "TagIn_Number" without being affected by a negative edge at input S. If the signal state of the operand "TagIn_1" changes from "0" to "1" before the timer expires, the timer is restarted.

The operand "TagOut" has the signal state "1" as long as the timer is running. When the timer expires or is reset, the operand "TagOut" is reset to "0".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

L: Load timer value (Page 3929)

S_ODT: Assign on-delay timer parameters and start

Description

The "Assign on-delay timer parameters and start" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV) as soon as the signal state at input S is "1". If the timer expires correctly and input S still has signal state "1" then output Q returns signal state "1". If the signal state at input S changes from "1" to "0" while the timer is running, the timer is stopped. In this case, output Q is reset to signal state "0".

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed. The current time value is output BI-coded at output BI and BCD-coded at output BCD.

If the time is running and the signal state at input R changes from "0" to "1" then the current time value and the time base are also set to zero. In this case, the signal state at output Q is "0". The timer is reset if the signal state is "1" at the R input even if the timer is not running and the RLO at input S is "1".

Specify the timer of the instruction in the placeholder above the box. The timer must be declared with the data type TIMER.

The "Assign on-delay timer parameters and start" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

The instruction data is updated at every access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameter

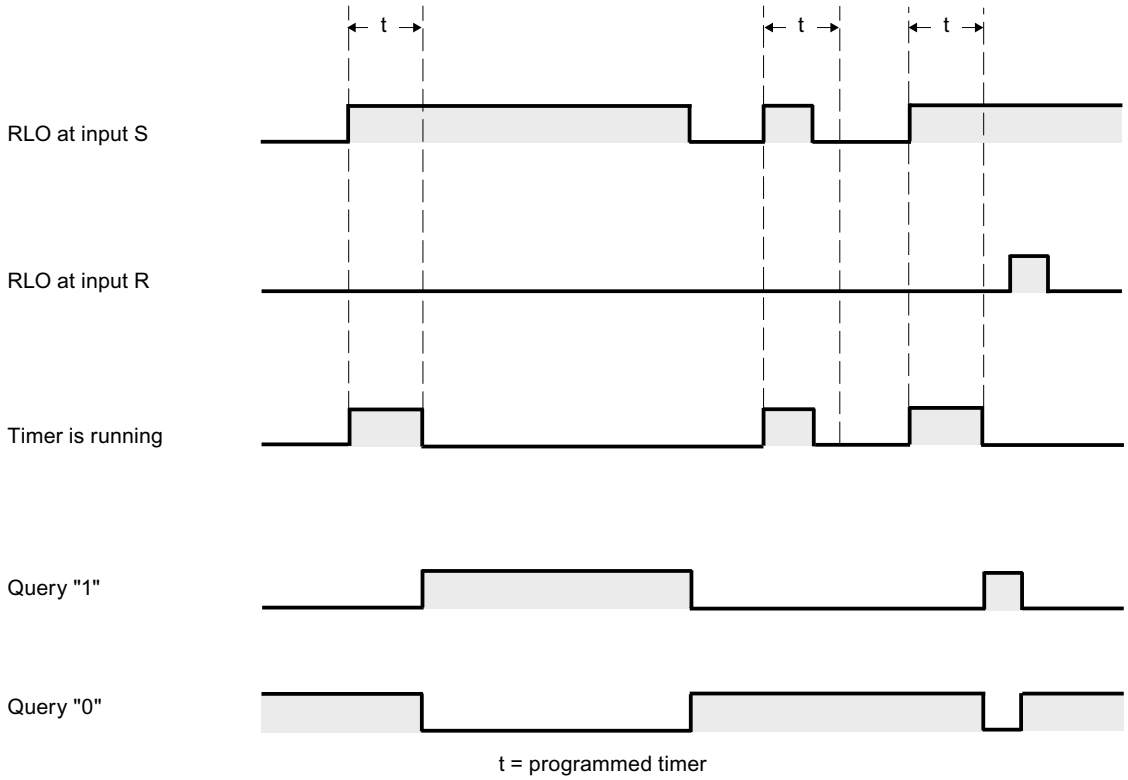
The following table shows the parameters of the "Assign on-delay timer parameters and start" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Preset time value
R	Input	BOOL	I, Q, M, T, C, D, L, P or constant	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (BI-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the timer

For additional information on valid data types, refer to "See also".

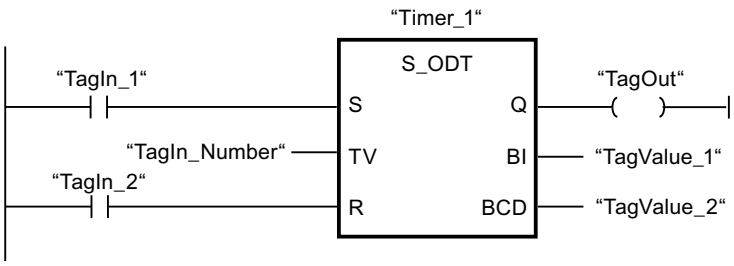
Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign on-delay timer parameters and start" instruction:



Example

The following example shows how the instruction works:



The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "0" to "1". The timer expires with the time value of the operand "TagIn_Number". If the timer expires and the operand has the signal state "1", the operand "TagOut" is set to "1". If the signal state of the operand "TagIn_1" changes from "1" to "0" before the timer expires, the timer is stopped. The operand "TagOut" has the signal state "0" in this case.

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

L: Load timer value (Page 3929)

S_ODTS: Assign retentive on-delay timer parameters and start

Description

The "Assign retentive on-delay timer parameters and start" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV), even if the signal state at input S changes to "0". If the timer expires, the "Q" output returns signal state "1" regardless of the signal state at input "S". If the signal state at input S changes from "0" to "1" while the timer is running, the timer is restarted with the duration programmed at input (TV).

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed. The current time value is output BI-coded at output BI and BCD-coded at output BCD.

Signal state "1" at input R resets the current time value and time base to "0" regardless of the signal state at start input S. In this case, the signal state at output Q is "0".

The "Assign retentive on-delay timer parameters and start" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

The instruction data is updated at every access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameter

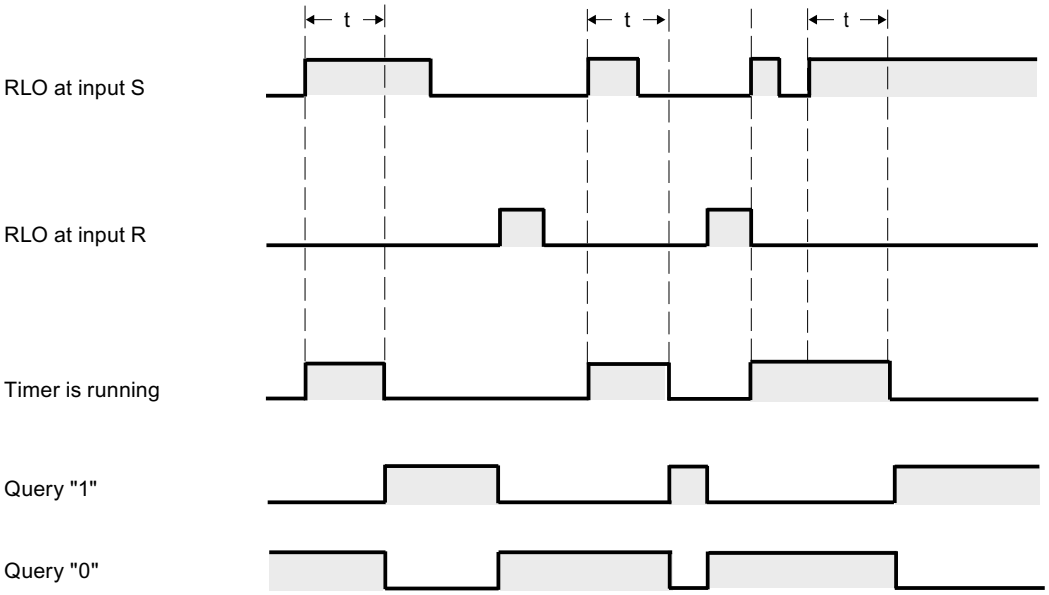
The following table shows the parameters of the "Assign retentive on-delay timer parameters and start" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Preset time value
R	Input	BOOL	I, Q, M, T, C, D, L, P or constant	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (BI-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the timer

For additional information on valid data types, refer to "See also".

Pulse timing diagram

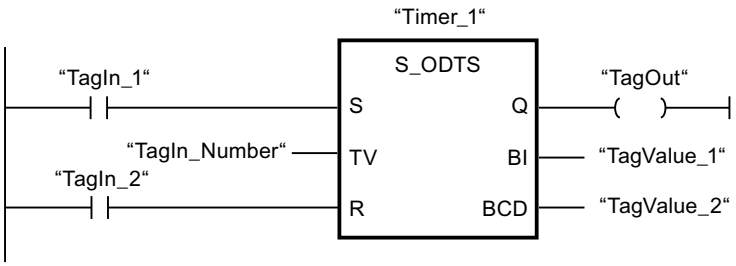
The following figure shows the pulse timing diagram of the "Assign retentive on-delay timer parameters and start" instruction:



t = programmed timer

Example

The following example shows how the instruction works:



The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "0" to "1". The timer expires with the time value of the operand "TagIn_Number", even if the signal state of the operand "TagIn_1" changes to "0". When the timer expires, the operand "TagOut" is set to "1". If the signal state of the operand "TagIn_1" changes from "0" to "1" while the timer is running, the timer is restarted.

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

L: Load timer value (Page 3929)

S_OFFDT: Assign off-delay timer parameters and start

Description

The "Assign off-delay timer parameters and start" instruction starts a programmed timer when a change from "1" to "0" (negative signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV). As long as the timer is running or input S returns signal state "1", then output Q has signal state "1". When the timer expires and the signal state at input S is "0", output Q is reset to the signal state "0". If the signal state at input S changes from "0" to "1" while the timer is running, the timer is stopped. The timer is only restarted after a falling signal edge is detected at input S.

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed. The current time value is output BI-coded at output BI and BCD-coded at output BCD.

Signal state "1" at input R resets the current time value and time base to "0". In this case, the signal state at output Q is "0".

The "Assign off-delay timer parameters and start" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

The instruction data is updated at every access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameter

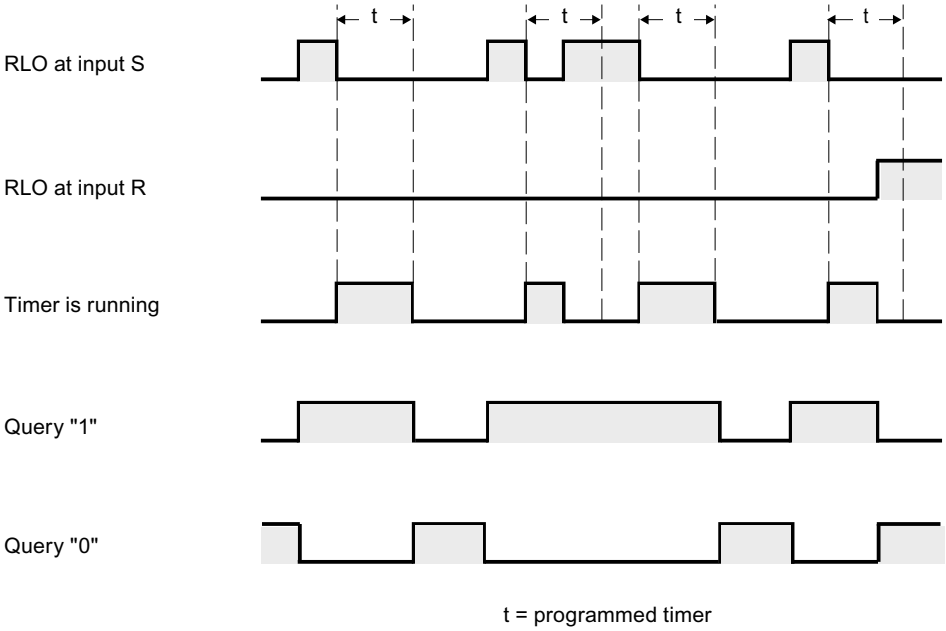
The following table shows the parameters of the "Assign off-delay timer parameters and start" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Preset time value
R	Input	BOOL	I, Q, M, T, C, D, L, P or constant	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (BI-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the timer

For additional information on valid data types, refer to "See also".

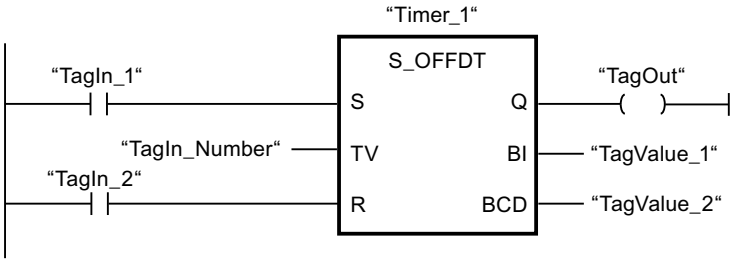
Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign off-delay timer parameters and start" instruction:



Example

The following example shows how the instruction works:



The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "1" to "0". The timer expires with the time value of the operand "TagIn_Number". The operand "TagOut" is set to "1" when the timer is running and when the operand "TagIn_1" has the signal state "0". If the signal state of the operand "TagIn_1" changes from "0" to "1" while the timer is running, the timer is reset.

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

L: Load timer value (Page 3929)

---(SP): Start pulse timer

Description

The "Start pulse timer" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO). The time runs with the specified duration as long as the RLO has the signal state "1". As long as the timer is running, the querying of the timer status for "1" returns the signal state "1". If there is a change from "1" to "0" in the RLO before the time value has elapsed, the timer stops. In this case, the querying of the timer status for "1" returns the signal state "0".

The duration is made up internally of a time value and a time base. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed.

The "Start pulse timer" instruction requires a preceding logic operation for edge evaluation and can only be placed at the right side of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

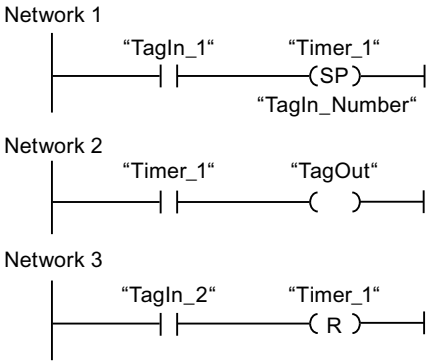
The following table shows the parameters of the "Start pulse timer" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Time duration>	Input	S5TIME, WORD	I, Q, M, D, L or constant	Duration with which the timer expires.
<Timer>	InOut/Input	TIMER	T	Timer which is started. The number of timers depends on the CPU.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "0" to "1". The timer expires with the time value of the operand "TagIn_Number" as long as the signal state of the operand "TagIn_1" is "1". If the signal state of the operand "TagIn_1" changes from "1" to "0" before the timer expires, the timer is stopped. As long as the timer is running, the operand "TagOut" has the signal state "1". A signal state change of the operand "TagIn_1" from "0" to "1" resets the timer, which stops the timer and sets the current time value to "0".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

L: Load timer value (Page 3929)

---(SE): Start extended pulse timer

Description

The "Start extended pulse timer" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO). The timer expires with the specified duration, even if the RLO changes to the signal state "0". As long as the timer is running, the querying of the timer status for "1" returns the signal state "1". If the RLO changes from "0" to "1" while the timer is running, the timer is restarted with the programmed duration. The querying of the timer status for "1" returns the signal state "0" when the timer expires.

The duration is made up internally of a time value and a time base. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed.

The "Start extended pulse timer" instruction requires a preceding logic operation for edge evaluation and can only be placed at the right side of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameter

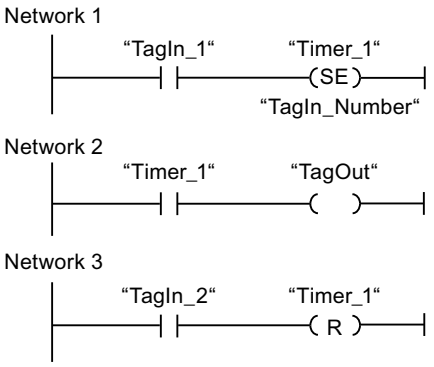
The following table shows the parameters of the "Start extended pulse timer" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Time duration>	Input	S5TIME, WORD	I, Q, M, D, L or constant	Duration with which the timer expires.
<Timer>	InOut/Input	TIMER	T	Timer which is started. The number of timers depends on the CPU.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "0" to "1". The timer expires with the time value of the operand "TagIn_Number" without being affected by a negative edge in the RLO. As long as the timer is running, the operand "TagOut" has the signal state "1". If the signal state of the operand "TagIn_1" changes from "0" to "1" before the timer expires, the timer is restarted.

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

L: Load timer value (Page 3929)

---(SD): Start on-delay timer

Description

The "Start on-delay timer" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO). The timer runs for the specified duration as long as the RLO is "1". If the timer expires and the RLO still has the signal state "1", the query of the timer status for "1" returns the signal state "1". If the RLO changes from "1" to "0" while the timer is running, the timer is stopped. In this case, the querying of the timer status for "1" returns the signal state "0".

The duration is made up internally of a time value and a time base. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed.

The "Start on-delay timer" instruction requires a preceding logic operation for edge evaluation and can only be placed at the right side of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameter

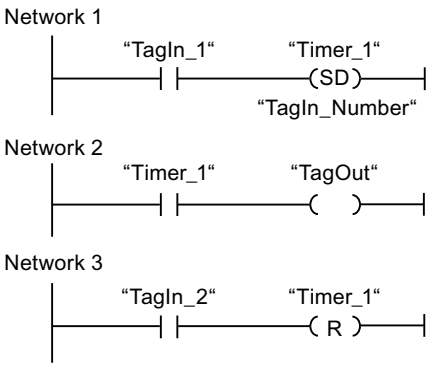
The following table shows the parameters of the "Start on-delay timer" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Time duration>	Input	S5TIME, WORD	I, Q, M, D, L or constant	Duration with which the timer expires.
<Timer>	InOut/Input	TIMER	T	Timer which is started. The number of timers depends on the CPU.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "0" to "1". The timer expires with the time value of the operand "TagIn_Number". If the timer expires and the RLO has the signal state "1", the operand "TagOut" is set to "1". If the signal state of the operand "TagIn_1" changes from "1" to "0" before the timer expires, the timer is stopped. If the signal state of the operand "TagIn_2" is "1", the timer "Timer_1" is reset, which stops the timer and sets the current time value to "0".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

L: Load timer value (Page 3929)

---(SS): Start retentive on-delay timer

Description

The "Start retentive on-delay timer" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO). The timer expires with the specified duration, even if the RLO changes to the signal state "0". When the timer expires, the querying of the timer status for "1" returns the signal state "1". After expiry of the timer, the timer can only be restarted if it is explicitly reset.

The duration is made up internally of a time value and a time base. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed.

The "Start retentive on-delay timer" instruction requires a preceding logic operation for edge evaluation and can only be placed at the right side of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

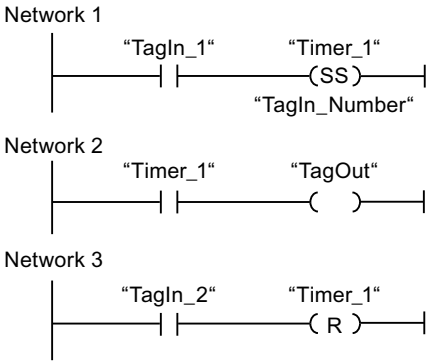
The following table shows the parameters of the "Start retentive on-delay timer" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Time duration>	Input	S5TIME, WORD	I, Q, M, D, L or constant	Duration with which the timer expires.
<Timer>	InOut/Input	TIMER	T	Timer which is started. The number of timers depends on the CPU.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "0" to "1". The timer expires with the time value of the operand "TagIn_Number". When the timer expires, the operand "TagOut" is set to "1". If the signal state of the operand "TagIn_1" changes from "0" to "1" while the timer is running, the timer is restarted. If the signal state of the operand "TagIn_2" is "1", the timer "Timer_1" is reset, which stops the timer and sets the current time value to "0".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

L: Load timer value (Page 3929)

---(SF): Start off-delay timer

Description

The "Start off-delay timer" instruction starts a programmed timer when a change from "1" to "0" (negative signal edge) is detected in the result of logic operation (RLO). The timer expires with the specified duration. As long as the timer is running, the querying of the timer status for "1" returns the signal state "1". If the RLO changes from "0" to "1" while the timer is running, the timer is reset. The timer is always restarted when the RLO changes from "1" to "0".

The duration is made up internally of a time value and a time base. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed.

The "Start off-delay timer" instruction requires a preceding logic operation for edge evaluation and can only be placed at the right side of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameter

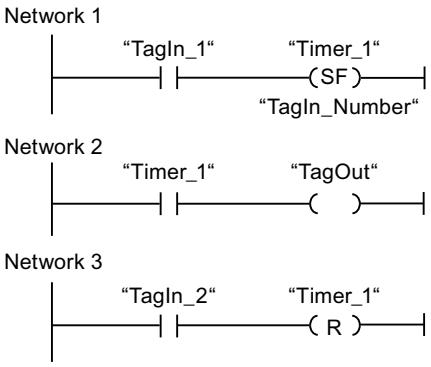
The following table shows the parameters of the "Start off-delay timer" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Time duration>	Input	S5TIME, WORD	I, Q, M, D, L or constant	Duration with which the timer expires.
<Timer>	InOut/Input	TIMER	T	Timer which is started. The number of timers depends on the CPU.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "1" to "0". The timer expires with the time value of the operand "TagIn_Number". As long as the timer is running, the operand "TagOut" is set to "1". If the signal state of the operand "TagIn_1" changes from "1" to "0" while the timer is running, the timer is restarted. If the signal

state of the operand "TagIn_2" is "1", the timer "Timer_1" is reset, which stops the timer and sets the current time value to "0".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

L: Load timer value (Page 3929)

Counter operations

IEC Counters

CTU: Count up

Description

You can use the "Count up" instruction to increment the value at output CV. When the signal state at the CU input changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value at the CV output is incremented by one. When the instruction executes for the first time, the current counter value at the CV output is set to zero. The counter is incremented each time a positive signal edge is detected, until it reaches the high limit for the specified data type at output CV. When the high limit is reached, the signal state at the CU input no longer has an effect on the instruction.

You can scan the counter status at the Q output. The signal state at the Q output is determined by the PV parameter. If the current counter value is greater than or equal to the value of the PV parameter, the Q output is set to signal state "1". In all other cases, the Q output has signal state "0".

The value at the CV output is reset to zero when the signal state at input R changes to "1". As long as the R input has the signal state "1", the signal state at the CU input has no effect on the instruction.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count up" instruction must be assigned to an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

For S7-1200 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER	• CTU_SINT / CTU_USINT • CTU_INT / CTU_UINT • CTU_DINT / CTU_UDINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER

For S7-1500 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER	• CTU_SINT / CTU_USINT • CTU_INT / CTU_UINT • CTU_DINT / CTU_UDINT • CTU_LINT / CTU_ULINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER")
- Declaration as a local tag of the type CTU_<Data type> or IEC_<Counter> in the "Static" section of a block (for example #MyIEC_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

If you create the IEC counter in a separate data block (single instance), the instance data block is created by default with "optimized block access" and the individual tags defined as retentive. For additional information on setting retentivity in a instance data block, refer to "See also".

If you create the IEC counter as local tag (multi-instance) in a function block with "optimized block access", this is defined as retentive in the block interface.

The execution of the "Count up" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table lists the parameters of the "Count up" instruction:

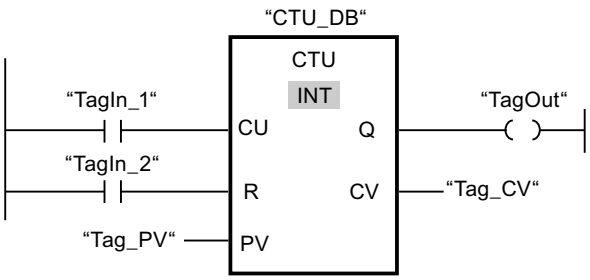
Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
CU	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Count input
R	Input	BOOL	I, Q, M, D, L, P or constant	I, Q, M, T, C, D, L, P or constant	Reset input
PV	Input	Integers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Value at which the output Q is set.
Q	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Counter status
CV	Output	Integers, CHAR, DATE	I, Q, M, D, L, P	I, Q, M, D, L, P	Current counter value

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



When the signal state of the "TagIn_1" operand changes from "0" to "1", the "Count up" instruction executes and the current counter value of the "Tag_CV" operand is incremented by one. With each additional positive signal edge, the counter value is incremented until the high limit of the data type (INT = 32767) is reached.

The value of the PV parameter is adopted as the limit for determining the "TagOut" output. The "TagOut" output has signal state "1" as long as the current counter value is greater than or equal to the value of the "Tag_PV" operand. In all other cases, the "TagOut" output has signal state "0".

See also

Overview of the valid data types (Page 2236)

Call function blocks as single or multi-instances (Page 2182)

Setting the retentivity of local tags (Page 2561)

Setting retentivity in an instance data block (Page 2839)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Selecting the data type of a LAD element (Page 2583)

Inserting operands into LAD instructions (Page 2601)

CTD: Count down

Description

You can use the "Count down" instruction to decrement the value at output CV. When the signal state at the CD input changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value at the CV output is decremented by one. When the instruction executes the first time, the counter value of the CV parameter is set to the value of the PV parameter. Each time a positive signal edge is detected, the counter value is decremented until it reaches the low limit of the specified data type. When the low limit is reached, the signal state at the CD input no longer has an effect on the instruction.

You can scan the counter status at the Q output. If the current counter value is less than or equal to zero, the Q output is set to signal state "1". In all other cases, the Q output has signal state "0".

The value at the CV output is set to the value of the PV parameter when the signal state at the LD input changes to "1". As long as the LD input has the signal state "1", the signal state at the CD input has no effect on the instruction.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count down" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

For S7-1200 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER	• CTD_SINT / CTD_USINT • CTD_INT / CTD_UINT • CTD_DINT / CTD_UDINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER

For S7-1500 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER	• CTD_SINT / CTD_USINT • CTD_INT / CTD_UINT • CTD_DINT / CTD_UDINT • CTD_LINT / CTD_ULINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER")
- Declaration as a local tag of the type CTD_<Data type> or IEC_<Counter> in the "Static" section of a block (for example #MyIEC_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

If you create the IEC counter in a separate data block (single instance), the instance data block is created by default with "optimized block access" and the individual tags defined as retentive. For additional information on setting retentivity in a instance data block, refer to "See also".

If you create the IEC counter as local tag (multi-instance) in a function block with "optimized block access", this is defined as retentive in the block interface.

The execution of the "Count down" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table lists the parameters of the "Count down" instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
CD	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Count input
LD	Input	BOOL	I, Q, M, D, L, P or constant	I, Q, M, T, C, D, L, P or constant	Load input
PV	Input	Integers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Value at which the output Q is set.

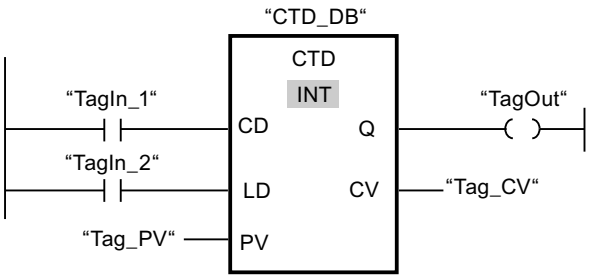
Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
Q	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Counter status
CV	Output	Integers, CHAR, DATE	I, Q, M, D, L, P	I, Q, M, D, L, P	Current counter value

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



When the signal state of the "TagIn_1" operand changes from "0" to "1", the "Count down" instruction executes and the value at the "Tag_CV" output is decremented by one. With each additional positive signal edge, the counter value is decremented until the low limit of the specified data type (INT = -32768) is reached.

The "TagOut" output has signal state "1" as long as the current counter value is less than or equal to zero. In all other cases, the "TagOut" output has signal state "0".

See also

Overview of the valid data types (Page 2236)

Call function blocks as single or multi-instances (Page 2182)

Setting retentivity in an instance data block (Page 2839)

Setting the retentivity of local tags (Page 2561)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Selecting the data type of a LAD element (Page 2583)

Inserting operands into LAD instructions (Page 2601)

CTUD: Count up and down

Description

You can use the "Count up and down" instruction to increment and decrement the counter value at the CV output. If the signal state at the CU input changes from "0" to "1" (positive signal edge), the current counter value is incremented by one and stored at the CV output. If the signal state at the CD input changes from "0" to "1" (positive signal edge), the counter value at the CV output is decremented by one. If there is a positive signal edge at the CU and CD inputs in one program cycle, the current counter value at the CV output remains unchanged.

The counter value can be incremented until it reaches the high limit of the data type specified at the CV output. When the high limit is reached, the counter value is no longer incremented on a positive signal edge. When the low limit of the specified data type is reached, the counter value is not decremented any further.

When the signal state at the LD input changes to "1", the counter value at the CV output is set to the value of the PV parameter. As long as the LD input has the signal state "1", the signal state at the CU and CD inputs has no effect on the instruction.

The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has the signal state "1", a change in the signal state of the CU, CD and LD inputs has no effect on the "Count up and down" instruction.

You can scan the current status of the up counter at the QU output. If the current counter value is greater than or equal to the value of the PV parameter, the QU output is set to signal state "1". In all other cases, the QU output has signal state "0".

You can scan the current status of the down counter at the QD output. If the current counter value is less than or equal to zero, the QD output is set to signal state "1". In all other cases, the QD output has signal state "0".

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count up and down" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

For S7-1200 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER	• CTUD_SINT / CTUD_USINT • CTUD_INT / CTUD_UINT • CTUD_DINT / CTUD_UDINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER

For S7-1500 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER	• CTUD_SINT / CTUD_USINT • CTUD_INT / CTUD_UINT • CTUD_DINT / CTUD_UDINT • CTUD_LINT / CTUD_ULINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER")
- Declaration as a local tag of the type CTUD_<Data type> or IEC_<Counter> in the "Static" section of a block (for example #MyIEC_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

If you create the IEC counter in a separate data block (single instance), the instance data block is created by default with "optimized block access" and the individual tags defined as retentive. For additional information on setting retentivity in a instance data block, refer to "See also".

If you create the IEC counter as local tag (multi-instance) in a function block with "optimized block access", this is defined as retentive in the block interface.

The execution of the "Count up and down" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table lists the parameters of the "Count up and down" instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
CU	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Count up input
CD	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Count down input
R	Input	BOOL	I, Q, M, D, L, P or constant	I, Q, M, T, C, D, L, P or constant	Reset input
LD	Input	BOOL	I, Q, M, D, L, P or constant	I, Q, M, T, C, D, L, P or constant	Load input

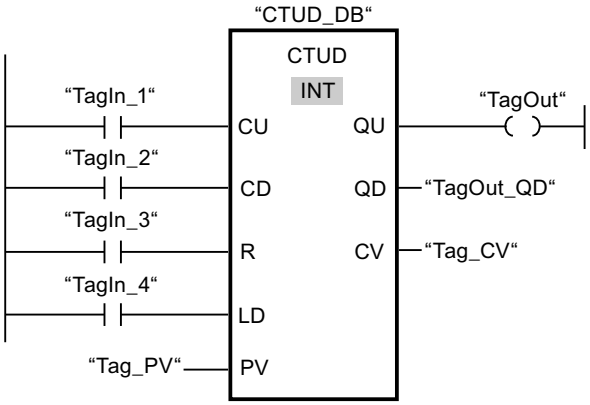
Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
PV	Input	Integers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Value at which the output QU is set.
QU	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Status of the up counter
QD	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Status of the down counter
CV	Output	Integers, CHAR, DATE	I, Q, M, D, L, P	I, Q, M, D, L, P	Current counter value

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the signal state at the "TagIn_1" or "TagIn_2" input changes from "0" to "1" (positive signal edge), the "Count up and down" instruction is executed. When there is a positive signal edge at the "TagIn_1" input, the current counter value is incremented by one and stored at the "Tag_CV" output. When there is a positive signal edge at the "TagIn_2" input, the counter value is decremented by one and stored at the "Tag_CV" output. When there is a positive signal edge at the CU input, the counter value is incremented until it reaches the high limit of 32767. If input CD has a positive signal edge, the counter value is decremented until it reaches the low limit of INT = -32768.

The "TagOut" output has signal state "1" as long as the current counter value is greater than or equal to the value at the "Tag_PV" input. In all other cases, the "TagOut" output has signal state "0".

The "TagOut_QD" output has signal state "1" as long as the current counter value is less than or equal to zero. In all other cases, the "TagOut_QD" output has signal state "0".

See also

Overview of the valid data types (Page 2236)
Call function blocks as single or multi-instances (Page 2182)
Setting retentivity in an instance data block (Page 2839)
Setting the retentivity of local tags (Page 2561)
Inserting LAD elements (Page 2579)
Editing LAD elements (Page 2594)
Selecting the data type of a LAD element (Page 2583)
Inserting operands into LAD instructions (Page 2601)

SIMATIC Counters

S_CUD: Assign parameters and count up / down

Description

You can use the "Assign parameters and count up / down" instruction to increment or decrement the value of a counter. If the signal state at the CU input changes from "0" to "1" (positive signal edge), the current counter value is incremented by one. If the signal state at the CD input changes from "0" to "1" (positive signal edge), the counter value is decremented by one. The current counter value is output as a hexadecimal value at output CV and BCD-coded at output CV_BCD. If there is a positive signal edge at the CU and CD inputs in one program cycle, the counter value remains unchanged.

The counter value can be incremented until the high limit of "999" is reached. When the high limit value is reached, the counter value is no longer incremented on a positive signal edge. When the low limit "0" is reached, the counter value is not decremented any further.

When the signal state at input S changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if RLO is "1" at the inputs CU and CD, the counter counts accordingly in the next scan cycle, even if no change in the signal edge was detected.

The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has the signal state "1", processing of the signal state of the CU, CD and S inputs has no effect on the counter value.

The signal state at output Q is "1" if the counter value is greater than zero. If the counter value is equal to zero, output Q has the signal state "0".

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

The "Assign parameters and count up / down" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

Parameters

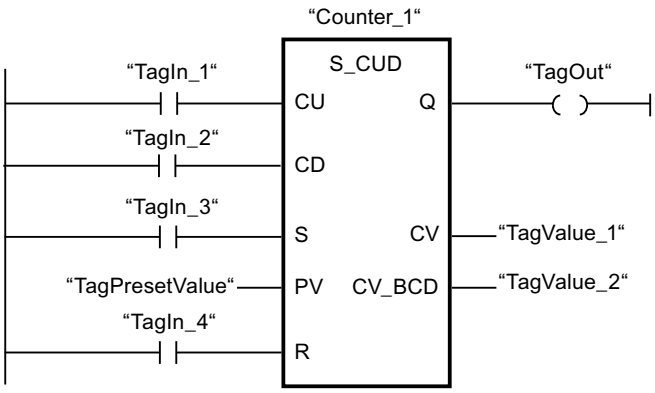
The following table shows the parameters of the "Assign parameters and count up / down" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Counter>	InOut/Input	COUNTER	C	Counter of the instruction The number of counters depends on the CPU.
CU	Input	BOOL	I, Q, M, D, L	Count up input
CD	Input	BOOL	I, Q, M, D, L, T, C or constant	Count down input
S	Input	BOOL	I, Q, M, D, L, T, C or constant	Input for presetting counter
PV	Input	WORD	I, Q, M, D, L or constant	Preset counter value (C#0 to C#999)
R	Input	BOOL	I, Q, M, D, L, T, C or constant	Reset input
BI	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value (hexadecimal)
CV_BCD	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the counter

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the signal state at the "TagIn_1"" or "TagIn_2" input changes from "0" to "1" (positive signal edge), the "Assign parameters and count up / down" instruction is executed. When there is a

positive signal edge at the "TagIn_1" input and the current counter value is less than "999", the counter value is incremented by one. When there is a positive signal edge at the "TagIn_2" input and the current counter value is greater than "0", the counter value is decremented by one.

When the signal state at input "TagIn_3" changes from "0" to "1", the counter value is set to the value of the operand "TagPresetValue". The counter value is reset to "0" when the "TagIn_4" operand has signal state "1".

The current counter value is saved as a hexadecimal value in the operand "TagValue_1" and BCD-coded in the operand "TagValue_2".

The "TagOut" output has the signal state "1" as long as the current counter value is not equal to "0".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

S_CU: Assign parameters and count up

Description

You can use the "Assign parameters and count up" instruction to increment the value of a counter. If the signal state at the CU input changes from "0" to "1" (positive signal edge), the current counter value is incremented by one. The current counter value is output as a hexadecimal value at output CV and BCD-coded at output CV_BCD. The counter value can be incremented until the limit of "999" is reached. When the limit is reached, the counter value is no longer incremented on a positive signal edge.

When the signal state at input S changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if RLO at input CU is "1", the counter will count accordingly in the next scan cycle, even when no change has been detected in the signal edge.

The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has the signal state "1", processing of the signal state of the CU and S inputs has no effect on the counter value.

The signal state at output Q is "1" if the counter value is greater than zero. If the counter value is equal to zero, output Q has the signal state "0".

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

The "Assign parameters and count up" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

Parameter

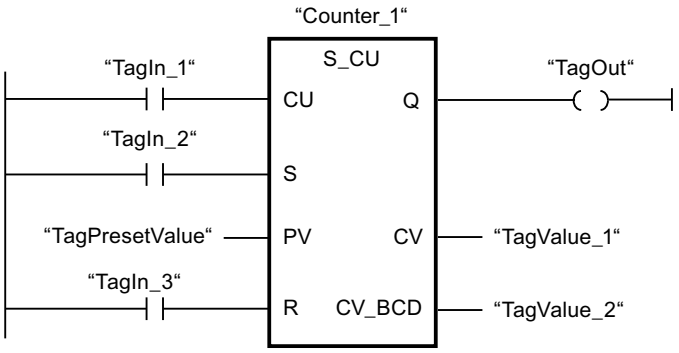
The following table shows the parameters of the "Assign parameters and count up" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Counter>	InOut/Input	COUNTER	C	Counter of the instruction The number of counters depends on the CPU.
CU	Input	BOOL	I, Q, M, D, L	Count up input
S	Input	BOOL	I, Q, M, D, L, T, C or constant	Input for presetting counter
PV	Input	WORD	I, Q, M, D, L or constant	Preset counter value (C#0 to C#999)
R	Input	BOOL	I, Q, M, D, L, T, C or constant	Reset input
CV	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value (hexadecimal)
CV_BCD	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the counter

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



When the signal state at the "TagIn_1" input changes from "0" to "1" (positive signal edge) and the current counter value is less than "999", the counter value is incremented by one. When the signal state at input "TagIn_2" changes from "0" to "1", the counter value is set to the value of the operand "TagPresetValue". The counter value is reset to "0" when the "TagIn_3" operand has signal state "1".

The current counter value is saved as a hexadecimal value in the operand "TagValue_1" and BCD-coded in the operand "TagValue_2".

The "TagOut" output has the signal state "1" as long as the current counter value is not equal to "0".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

S_CD: Assign parameters and count down

Description

You can use the "Assign parameters and count down" instruction to decrement the value of a counter. If the signal state at the CD input changes from "0" to "1" (positive signal edge), the counter value is decremented by one. The current counter value is output as a hexadecimal value at output CV and BCD-coded at output CV_BCD. The counter value is decremented until the low limit of "0" is reached. When the low limit is reached, the counter value is no longer decremented on a positive signal edge.

When the signal state at input S changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if RLO at input CD is "1", the counter will count accordingly in the next scan cycle, even when no change has been detected in the signal edge.

The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has the signal state "1", processing of the signal state of the CD and S inputs has no effect on the counter value.

The signal state at output Q is "1" if the counter value is greater than zero. If the counter value is equal to zero, output Q has the signal state "0".

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

The "Assign parameters and count down" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

Parameter

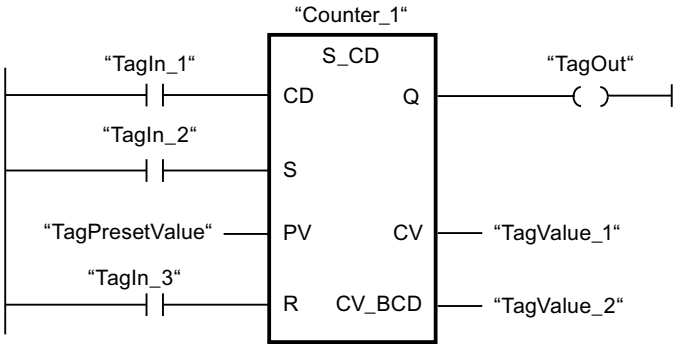
The following table shows the parameters of the "Assign parameters and count down" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Counter>	InOut/Input	COUNTER	C	Counter of the instruction The number of counters depends on the CPU.
CD	Input	BOOL	I, Q, M, D, L or constant	Count down input
S	Input	BOOL	I, Q, M, D, L, T, C or constant	Input for presetting counter
PV	Input	WORD	I, Q, M, D, L or constant	Preset counter value (C#0 to C#999)
R	Input	BOOL	I, Q, M, D, L, T, C or constant	Reset input
CV	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value (hexadecimal)
CV_BCD	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the counter

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



When the signal state at the "TagIn_1" input changes from "0" to "1" (positive signal edge) and the current counter value is greater than "0", the counter value is decremented by one. When the signal state at input "TagIn_2" changes from "0" to "1", the counter value is set to the value of the operand "TagPresetValue". The counter value is reset to "0" when the "TagIn_3" operand has signal state "1".

The current counter value is saved as a hexadecimal value in the operand "TagValue_1" and BCD-coded in the operand "TagValue_2".

The "TagOut" output has the signal state "1" as long as the current counter value is not equal to "0".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

---(SC): Set counter value

Description

You can use the "Set counter value" instruction to set the value of a counter. The instruction is executed when the result of logic operation (RLO) at the input changes from "0" to "1". When the instruction is executed, the counter is set to the specified counter value.

The "Set counter value" instruction needs a preceding logic operation for the edge evaluation and can only be placed on the right edge of the network.

Parameter

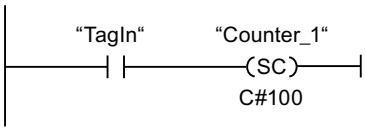
The following table lists the parameters of the "Set counter value" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Count value>	Input	WORD	I, Q, M, D, L or constant	Value with which the counter is preset in the BCD format. (C#0 to C#999)
<Counter>	InOut/Input	COUNTER	C	Counter that is preset.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



the counter "Counter_1" starts with the value "100" when the signal state of the "TagIn" operand changes from "0" to "1".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

---(CU): Count up

Description

With the "Count up" instruction you can increment the value of the specified counter by one if there is a positive signal edge in the result of logic operation (RLO). The counter value can be incremented until the limit of "999" is reached. When the limit is reached, the counter value is no longer incremented on a positive signal edge.

The "Count up" instruction requires a preceding logic operation for edge evaluation and can only be placed at the right side of the network.

Parameter

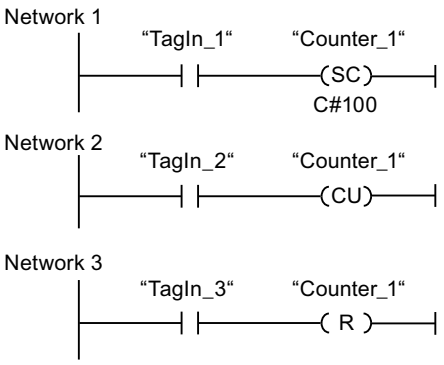
The following table shows the parameters of the "Count up" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Counter>	InOut/Input	COUNTER	C	Counter whose value is incremented.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



When the signal state of the "TagIn_1" operand changes from "0" to "1" (positive signal edge), the counter "Counter_1" is preset with the value "100".

The value of the counter "Counter_1" is incremented by one when the signal state of the operand "TagIn_2" changes from "0" to "1".

When the operand "TagIn_3" has the signal state "1", the value of the counter "Counter_1" is reset to "0".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

---(CD): Count down

Description

With the "Count down" instruction you can increment the value of the specified counter by one if there is a positive signal edge in the result of logic operation (RLO). The counter value can be decremented until the limit of "0" is reached. When the limit is reached, the counter value is no longer changed on a positive signal edge.

The "Count down" instruction requires a preceding logic operation for edge evaluation and can only be placed at the right side of the network.

Parameter

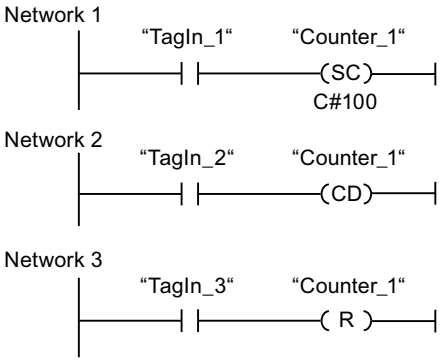
The following table shows the parameters of the "Count down" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Counter>	InOut/Input	COUNTER	C	Counter whose value is decremented.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



When the signal state of the "TagIn_1" operand changes from "0" to "1" (positive signal edge), the counter "Counter_1" is preset with the value "100".

The value of the counter "Counter_1" is decremented by one when the signal state of the operand "TagIn_2" changes from "0" to "1".

When the operand "TagIn_3" has the signal state "1", the value of the counter "Counter_1" is reset to "0".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Comparator operations

CMP ==: Equal

Description

You can use the "Equal" instruction to determine if a first comparison value (<Operand1>) is equal to a second comparison value (<Operand2>).

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify the first comparison value (<Operand1>) at the operand placeholder above the instruction. Specify the second comparison value (<Operand2>) at the operand placeholder below the instruction.

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison.

The following table lists examples of string comparisons:

<Operand1>	<Operand2>	RLO of the instruction
'AA'	'AA'	1
'Hello World'	'HelloWorld'	0
'AA'	'aa'	0

You can also use the "Equal" instruction to compare individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

If IEC check is enabled, the operands to be compared must be of the same data type. If IEC check is not enabled, the width (length) of the operands must be the same. When floating-point numbers are compared, the operands to be compared must have the same data type regardless of the setting for the IEC Check.

Note

Comparison of floating-point numbers

If you want to compare the data types REAL or LREAL, instead of the instruction "CMP ==: Equal", use the instruction "IN_RANGE: Value within range".

Note

Comparison of the data type PORT

To be able to compare the operands of the PORT data type with the "Equal" instruction, you need to select the WORD data type from the drop-down list of the instructions box.

Parameters

The following table lists the parameters of the "Equal" instruction:

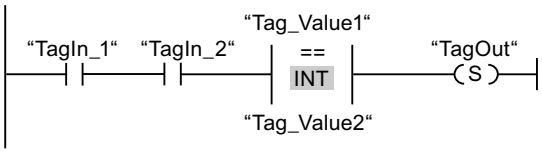
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<Operand1>	Input	Bit strings, integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Bit strings, integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	First comparison value
<Operand2>	Input	Bit strings, integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Bit strings, integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" = "Tag_Value2".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

CMP <>: Not equal

Description

You can use the "Not equal" instruction to determine if a first comparison value (<Operand1>) is not equal to a second comparison value (<Operand2>).

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify the first comparison value (<Operand1>) at the operand placeholder above the instruction. Specify the second comparison value (<Operand2>) at the operand placeholder below the instruction.

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison.

The following table lists examples of string comparisons:

<Operand1>	<Operand2>	RLO of the instruction
'AA'	'aa'	1
'Hello World'	'HelloWorld'	1
'AA'	'AA'	0

You can also use the "Not equal" instruction to compare individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

If IEC check is enabled, the operands to be compared must be of the same data type. If IEC check is not enabled, the width (length) of the operands must be the same. If the floating-point numbers are being compared, the operands to be compared must be of the same data type regardless of the IEC check setting.

Note

Comparison of the data type PORT

To be able to compare the operands of the PORT data type with the "Not equal" instruction, you need to select the WORD data type from the drop-down list of the instructions box.

Parameters

The following table lists the parameters of the "Not equal" instruction:

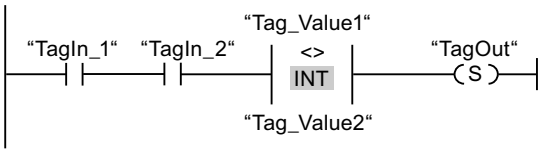
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<Operand1>	Input	Bit strings, integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Bit strings, integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	First comparison value
<Operand2>	Input	Bit strings, integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Bit strings, integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" <> "Tag_Value2".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

CMP >=: Greater or equal

Description

You can use the "Greater or equal" instruction to determine if a first comparison value (<Operand1>) is greater than or equal to a second comparison value (<Operand2>). Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify the first comparison value (<Operand1>) at the operand placeholder above the instruction. Specify the second comparison value (<Operand2>) at the operand placeholder below the instruction.

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the longer string is considered greater.

The following table lists examples of string comparisons:

<Operand1>	<Operand2>	RLO of the instruction
'BB'	'AA'	1
'AAA'	'AA'	1
'Hello World'	'Hello World'	1
'Hello World'	'HelloWorld'	0
'AA'	'aa'	0
'AAA'	'a'	0

You can also use the "Greater or equal" instruction to compare individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

When time values are compared, the RLO of the instruction is "1" if the point of time at <Operand1> is greater (more recent) than or equal to the point of time at <Operand2>.

Parameters

The following table lists the parameters of the "Greater or equal" instruction:

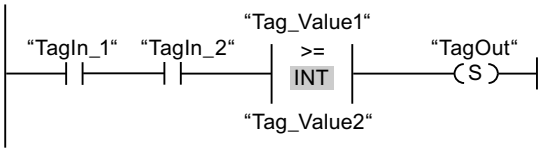
Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<Operand1>	Input	Integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	First comparison value
<Operand2>	Input	Integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" >= "Tag_Value2".

See also

Overview of the valid data types (Page 2236)

Example of detecting the fill level of a storage area (Page 3085)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

CMP <=: Less or equal

Description

You can use the "Less or equal" instruction to determine if a first comparison value (<Operand1>) is less than or equal to a second comparison value (<Operand2>). Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify the first comparison value (<Operand1>) at the operand placeholder above the instruction. Specify the second comparison value (<Operand2>) at the operand placeholder below the instruction.

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the shorter string is considered smaller.

The following table lists examples of string comparisons:

<Operand1>	<Operand2>	RLO of the instruction
'AA'	'aa'	1
'AAA'	'a'	1
'Hello World'	'Hello World'	1
'HelloWorld'	'Hello World'	0
'BB'	'AA'	0
'AAA'	'AA'	0

You can also use the "Less or equal" instruction to compare individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

When time values are compared, the RLO of the instruction is "1" if the point of time at <Operand1> is less (less recent) than or equal to the point of time at <Operand2>.

Parameters

The following table lists the parameters of the "Less or equal" instruction:

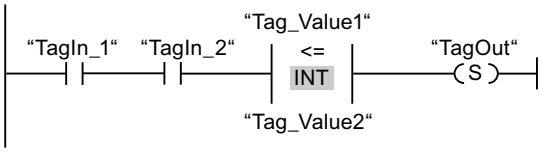
Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<Operand1>	Input	Integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	First comparison value
<Operand2>	Input	Integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" <= "Tag_Value2".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

CMP >: Greater than

Description

You can use the "Greater than" instruction to determine if a first comparison value (<Operand1>) is greater than a second comparison value (<Operand2>). Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify the first comparison value (<Operand1>) at the operand placeholder above the instruction. Specify the second comparison value (<Operand2>) at the operand placeholder below the instruction.

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the longer string is considered greater.

The following table lists examples of string comparisons:

<Operand1>	<Operand2>	RLO of the instruction
'BB'	'AA'	1
'AAA'	'AA'	1
'AA'	'aa'	0
'AAA'	'a'	0

You can also use the "Greater than" instruction to compare individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

When time values are compared, the RLO of the instruction is "1" if the point of time at <Operand1> is greater (more recent) than the point of time at <Operand2>.

Parameters

The following table lists the parameters of the "Greater than" instruction:

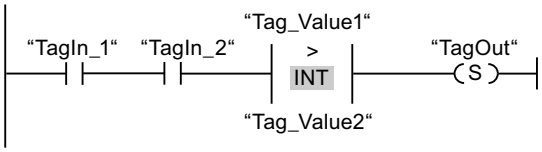
Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<Operand1>	Input	Integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	First comparison value
<Operand2>	Input	Integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" > "Tag_Value2".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

CMP <: Less than

Description

You can use the "Less than" instruction to determine if a first comparison value (<Operand1>) is less than a second comparison value (<Operand2>). Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify the first comparison value (<Operand1>) at the operand placeholder above the instruction. Specify the second comparison value (<Operand2>) at the operand placeholder below the instruction.

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the shorter string is considered smaller.

The following table lists examples of string comparisons:

<Operand1>	<Operand2>	RLO of the instruction
'AA'	'aa'	1
'AAA'	'a'	1
'BB'	'AA'	0
'AAA'	'AA'	0

You can also use the "Less than" instruction to compare individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

When time values are compared, the RLO of the instruction is "1" if the point of time at <Operand1> is less (less recent) than the point of time at <Operand2>.

Parameters

The following table lists the parameters of the "Less than" instruction:

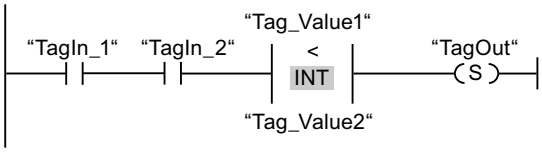
Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<Operand1>	Input	Integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	First comparison value
<Operand2>	Input	Integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" < "Tag_Value2".

See also

Overview of the valid data types (Page 2236)

Example of detecting the fill level of a storage area (Page 3085)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

IN_RANGE: Value within range

Description

You can use the "Value within range" instruction to determine if the value at the VAL input is within a specific value range.

You specify the limits of the value range with the MIN and MAX inputs. The "Value within range" instruction compares the value at the VAL input with the values of the MIN and MAX inputs and sends the result to the box output. If the value at the VAL input fulfills the comparison MIN <= VAL or VAL <= MAX, the box output has the signal state "1". If the comparison is not fulfilled, the box output has the signal state "0".

If the box input has the signal state "0", the "Value within range" instruction is not executed.

The comparison function can only execute if the values to be compared are of the same data type and the box input is interconnected.

Parameters

The following table shows the parameters of the "Value within range" instruction:

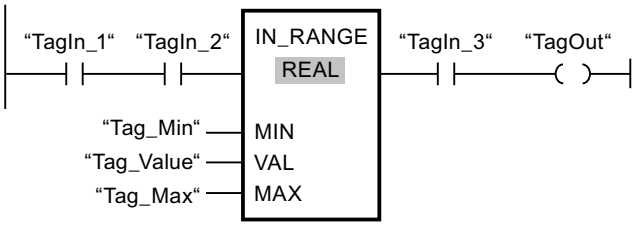
Parameter	Declaration	Data type	Memory area	Description
Box input	Input	BOOL	I, Q, M, D, L	Result of the previous logic operation
MIN	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Low limit of the value range
VAL	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Comparison value
MAX	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	High limit of the value range
Box output	Output	BOOL	I, Q, M, D, L	Result of the comparison

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also":

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The value of the operand "Tag_Value" is within the value range that is specified by the current values of the operands "Tag_Min" and "Tag_Max" (MIN <= VAL or VAL <= MAX).
- The operand "TagIn_3" has the signal state "1".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

OUT_RANGE: Value outside range

Description

You can use the "Value outside range" instruction to determine if the value at the VAL input is outside a specific value range.

You specify the limits of the value range with the MIN and MAX inputs. The "Value outside range" instruction compares the value at the VAL input with the values of the MIN and MAX inputs and sends the result to the box output. If the value at the VAL input fulfills the comparison MIN > VAL or VAL > MAX, the box output has the signal state "1". The box output also has the signal state "1" if a specified operand with the REAL data type shows an invalid value.

The box output returns the signal state "0", if the value at input VAL does not satisfy the MIN > VAL or VAL > MAX condition.

If the box input has the signal state "0", the "Value outside range" instruction is not executed.

The comparison function can only execute if the values to be compared are of the same data type and the box input is interconnected.

Parameters

The following table shows the parameters of the "Value outside range" instruction:

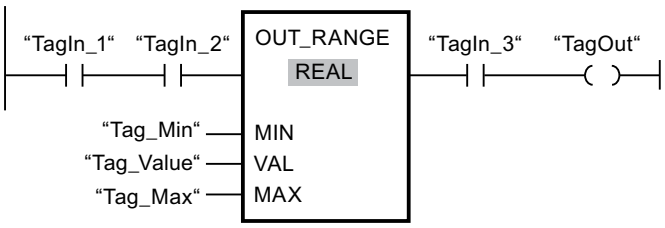
Parameter	Declaration	Data type	Memory area	Description
Box input	Input	BOOL	I, Q, M, D, L	Result of the previous logic operation
MIN	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Low limit of the value range
VAL	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Comparison value
MAX	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	High limit of the value range
Box output	Output	BOOL	I, Q, M, D, L	Result of the comparison

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The value of the operand "Tag_Value" is outside the value range that is specified by the values of the operands "Tag_Min" and "Tag_Max" (MIN > VAL or VAL > MAX).
- The operand "TagIn_3" has the signal state "1".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

----I OK I----: Check validity

Description

You can use the "Check validity" instruction to check if the value of an operand (<operand>) is a valid floating-point number. The query is started in each program cycle when the signal state at the input of the instruction is "1".

The output of the instruction has signal state "1" when the value of the operand is a valid floating-point number at the time of the query and the input of the instruction has signal state "1". In all other cases, the signal state at the output of the "Check validity" instruction is "0".

You can use the "Check validity" instruction together with the EN mechanism. If you connect the instruction box to an EN enable input, the enable input is set only when the result of the validity query of the value is positive. You can use this function to ensure that an instruction is enabled only when the value of the specified operand is a valid floating-point number.

Parameters

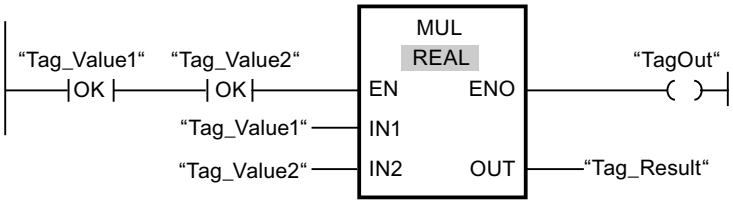
The following table shows the parameters of the "Check validity" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L	Value to be queried.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



When the values of the operands "Tag_Value1" and "Tag_Value2" show valid floating-point numbers, the "Multiply" (MUL) instruction is activated and the ENO output is set. During the execution of the "Multiply" (MUL) instruction, the value of the operand "Tag_Value1" is multiplied with the value of the operand "Tag_Value2". The product of the multiplication is then stored in the operand "Tag_Result". If the instruction is executed without errors, the ENO and "TagOut" outputs are set to signal state "1".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

----I NOT_OK I----: Check invalidity

Description

You can use the "Check invalidity" instruction to check if the value of an operand (<operand>) is an invalid floating-point number. The query is started in each program cycle when the signal state at the input of the instruction is "1".

The output of the instruction has signal state "1" when the value of the operand is an invalid floating-point number at the time of the query and the input of the instruction has signal state "1". In all other cases, the signal state at the output of the "Check invalidity" instruction is "0".

Parameters

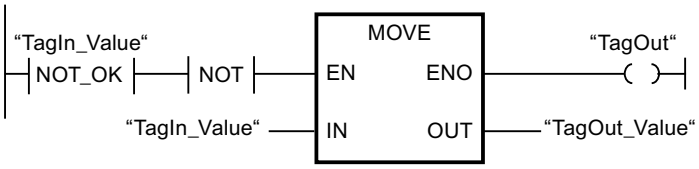
The following table shows the parameters of the "Check invalidity" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L	Value to be queried.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



When the value of operand "TagIn_Value" is an invalid floating-point number, the "Move value" (MOVE) instruction will not be executed. The "TagOut" output is reset to signal state "0".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

VARIANT instructions

EQ_Type: Compare data type for EQUAL with the data type of a tag

Description

You can use the "Compare data type for EQUAL with the data type of a tag" instruction to query the data type of a tag to which a VARIANT points. You compare the data type of the tag (<Operand1>) to the data type of a tag (<Operand2>) that you have declared in the block interface to determine if they are "Equal".

<Operand1> must have the VARIANT data type. <Operand2> can be an elementary data type or a PLC data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify <Operand1> at the operand placeholder above the instruction. Specify <Operand2> at the operand placeholder below the instruction.

Parameters

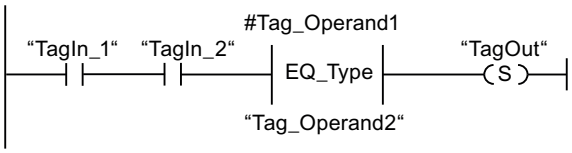
The following table lists the parameters of the "Compare data type for EQUAL with the data type of a tag" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Input	VARIANT	L	First operand
<Operand2>	Input	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY, PLC data types	I, Q, M, D, L	Second operand

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the "#Tag_Operand1" operand is equal to "Tag_Operand2".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

NE_Type: Compare data type for UNEQUAL with the data type of a tag

Description

You can use the "Compare data type for EQUAL with the data type of a tag" instruction to query the data type of a tag to which a VARIANT points. You compare the data type of the tag (<Operand1>) to the data type of a tag (<Operand2>) that you have declared in the block interface to determine if they are "Not equal".

<Operand1> must have the VARIANT data type. <Operand2> can be an elementary data type or a PLC data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify <Operand1> at the operand placeholder above the instruction. Specify <Operand2> at the operand placeholder below the instruction.

Parameters

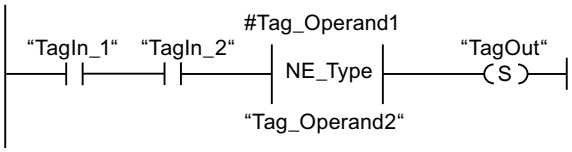
The following table lists the parameters of the "Compare data type for UNEQUAL with the data type of a tag" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Input	VARIANT	L	First operand
<Operand2>	Input	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY, PLC data types	I, Q, M, D, L	Second operand

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the "#Tag_Operand1" operand is not equal to "Tag_Operand2".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

EQ_ElemType: Compare element data type for EQUAL with the data type of a tag

Description

You can use the "Compare element data type for EQUAL with the data type of a tag" instruction to query the data type of a tag to which a VARIANT points. You compare the data type of the tag (<Operand1>) to the data type of a tag (<Operand2>) that you have declared in the block interface to determine if they are "Equal".

<Operand1> must have the VARIANT data type. <Operand2> can be an elementary data type or a PLC data type.

If the data type of the VARIANT tag (<Operand1>) is an ARRAY, the data type of the ARRAY elements is compared.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify <Operand1> at the operand placeholder above the instruction. Specify <Operand2> at the operand placeholder below the instruction.

Parameters

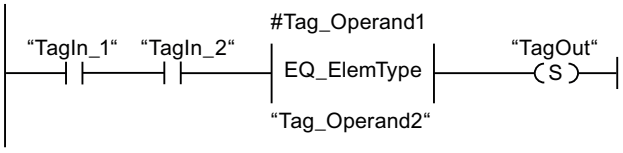
The following table lists the parameters of the "Compare element data type for EQUAL with the data type of a tag" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Input	VARIANT	L	First operand
<Operand2>	Input	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY, PLC data types	I, Q, M, D, L	Second operand

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the "#Tag_Operand1" operand is equal to "Tag_Operand2".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

NE_ElemType: Compare element data type for UNEQUAL with the data type of a tag

Description

You can use the "Compare element data type for UNEQUAL with the data type of a tag" instruction to query the data type of a tag to which a VARIANT points. You compare the data type of the tag (<Operand1>) to the data type of a tag (<Operand2>) that you have declared in the block interface to determine if they are "Not equal".

<Operand1> must have the VARIANT data type. <Operand2> can be an elementary data type or a PLC data type.

If the data type of the VARIANT tag (<Operand1>) is an ARRAY, the data type of the ARRAY elements is compared.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify <Operand1> at the operand placeholder above the instruction. Specify <Operand2> at the operand placeholder below the instruction.

Parameters

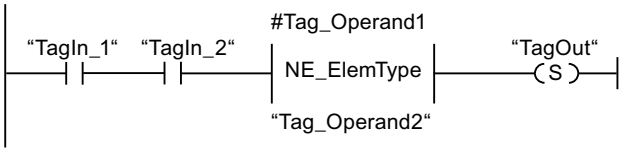
The following table lists the parameters of the "Compare element data type for UNEQUAL with the data type of a tag" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand1>	Input	VARIANT	L	First operand
<Operand2>	Input	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY, PLC data types	I, Q, M, D, L	Second operand

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the "#Tag_Operand1" operand is not equal to "Tag_Operand2".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

IS_NULL: Compare for EQUALS ZERO

Description

You can use the instruction "Compare for EQUALS ZERO" to query whether the VARIANT does not point to an object.

The <Operand> must have the VARIANT data type.

Note

VARIANT tag points to an ANY pointer

If the VARIANT tag points to an ANY pointer, the instruction always returns the result RLO = "0" even if the ANY pointer is ZERO.

Parameters

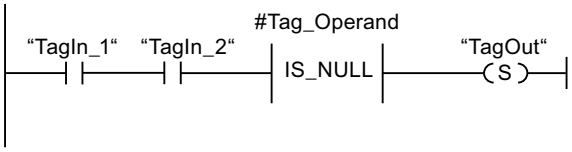
The following table lists the parameters of the "Compare for EQUALS ZERO" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	VARIANT	L	Operand that is compared for EQUAL ZERO

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the "#Tag_Operand" operand does not point to an object.

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

NOT_NULL: Compare for UNEQUALS ZERO

Description

You can use the instruction "Compare for UNEQUALS ZERO" to query whether the VARIANT points to an object.

The <Operand> must have the VARIANT data type.

Note

VARIANT tag points to an ANY pointer

If the VARIANT tag points to an ANY pointer, the instruction always returns the result RLO = "1" even if the ANY pointer is ZERO.

Parameters

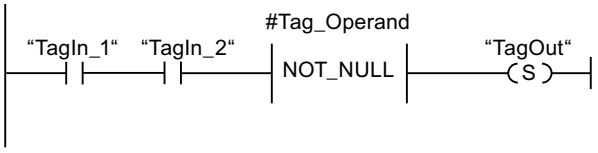
The following table lists the parameters of the "Compare for UNEQUALS ZERO" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	VARIANT	L	Operand that is compared for NOT EQUAL ZERO

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the "#Tag_Operand" operand points to an object.

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

IS_ARRAY: Check for ARRAY

Description

You can use the instruction "Check for ARRAY" to query whether the VARIANT points to a tag of the ARRAY data type.

The <Operand> must have the VARIANT data type.

Parameters

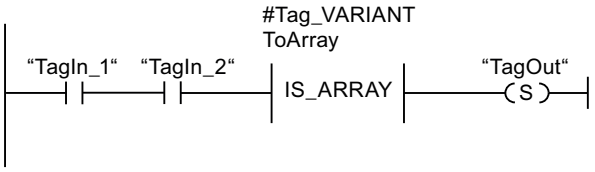
The following table lists the parameters of the "Check for ARRAY" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	VARIANT	L	Operand that is query for ARRAY

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the "#Tag_VARIANTToArray" operand has the ARRAY data type.

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

Math functions

CALCULATE: Calculate

Description

The "Calculate" instruction is used to define and execute an expression for the calculation of mathematical operations or complex logic operations depending on the selected data type.

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box. Depending on the data type selected, you can combine the functions of certain instructions to perform a complex calculation. The expression to be calculated is specified via a dialog you can open via the "Calculator" icon at the top of the instruction box. The expression can contain names of input parameters and the syntax of the instructions. Operand names and operand addresses cannot be specified.

In its initial state, the instruction box contains at least 2 inputs (IN1 and IN2). The number of inputs can be extended. The inserted inputs are numbered in ascending order in the box.

The values of the inputs are used to execute the specified expression. Not all of the defined inputs have to be used in the expression. The result of the instruction is transferred to the output OUT.

Note

If one of the mathematical operations fails in the expression, no result is transferred to the output OUT and the enable output ENO returns the signal state "1".

If you use inputs in the expression that are not available in the box, they are inserted automatically. This requires that there are no gaps in the numbering of the inputs to be newly defined in the expression. For example, you cannot use the IN4 input in the expression unless the IN3 input has been defined.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The result of the "Calculate" instruction is outside the range permitted for the data type specified at the OUT output.
- A floating-point number has an invalid value.
- An error occurred during execution of one of the instructions in the expression.

The following table shows the instructions that can be executed together in the expression of the instruction "Calculate", depending on the selected data type:

Data type	Instruction	Syntax	Example
Bit strings	AND: AND logic operation	AND	IN1 AND IN2 OR IN3
	OR: OR logic operation	OR	
	XOR: EXCLUSIVE OR logic operation	XOR	
	INV: Create ones complement	NOT	
	SWAP: Swap [1]	SWAP	
Integers	ADD: Add	+	(IN1 + IN2) * IN3; (ABS(IN2)) * (ABS(IN1))
	SUB: Subtract	-	
	MUL: Multiply	*	
	DIV: Divide	/	
	MOD: Return remainder of division	MOD	
	INV: Create ones complement	NOT	
	NEG: Create twos complement	-(in1)	
	ABS: Form absolute value	ABS( )	
Floating-point numbers	ADD: Add	+	((SIN(IN2) * SIN(IN2) + (SIN(IN3) * SIN(IN3)) / IN3)); (SQR(SIN(IN2)) + (SQR(COS(IN3)) / IN2))
	SUB: Subtract	-	
	MUL: Multiply	*	
	DIV: Divide	/	
	EXPT: Exponentiate	**	
	ABS: Form absolute value	ABS( )	
	SQR: Form square	SQR( )	
	SQRT: Form square root	SQRT( )	
	LN: Form natural logarithm	LN( )	
	EXP: Form exponential value	EXP( )	
	FRAC: Return fraction	FRAC( )	
	SIN: Form sine value	SIN( )	
	COS: Form cosine value	COS( )	
	TAN: Form tangent value	TAN( )	
	ASIN: Form arcsine value	ASIN( )	
	ACOS: Form arcosine value	ACOS( )	
	ATAN: Form arctangent value	ATAN( )	
	NEG: Create twos complement	-(in1)	
	TRUNC: Truncate numerical value	TRUNC( )	
	ROUND: Round numerical value	ROUND( )	
	CEIL: Generate next higher integer from floating-point number	CEIL( )	
	FLOOR: Generate next lower integer from floating-point number	FLOOR( )	

[1] Not possible for data type BYTE.

Parameters

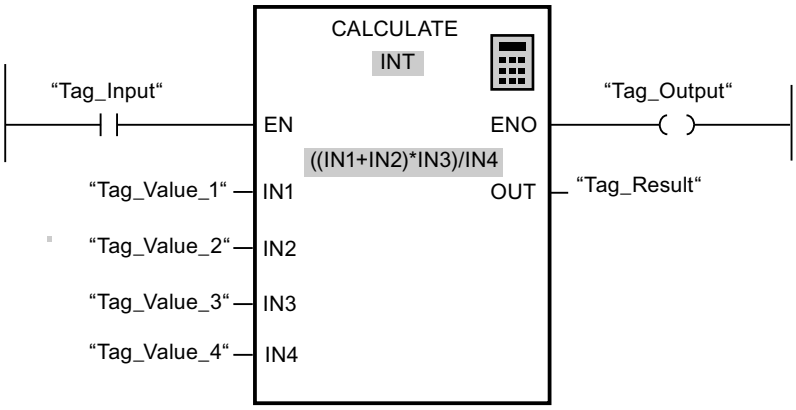
The following table shows the parameters of the "Calculate" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P or constant	First available input
IN2	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P or constant	Second available input
INn	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P or constant	Additionally inserted inputs
OUT	Output	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P	Output to which the end result is to be transferred.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN1	Tag_Value_1	4
IN2	Tag_Value_2	4
IN3	Tag_Value_3	3
IN4	Tag_Value_4	2
OUT	Tag_Result	12

If the "Tag_Input" input has the signal state"1", the "Calculate" instruction is executed. The value of operand "Tag_Value_1" is added to the value of operand "Tag_Value_2". The sum is

multiplied with the value of operand "Tag_Value_3". The product is divided by the value of operand "Tag_Value_4". The quotient is transferred as end result to the operand "Tag_Result" at the OUT output of the instruction. If the instruction is executed without errors, the enable output ENO and operand "Tag_Output" are set to "1".

See also

Using the "Calculate" instruction (Page 2590)

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Inserting additional inputs and outputs in LAD elements (Page 2598)

Removing inputs and outputs (Page 2599)

Basics of the EN/ENO mechanism (Page 2447)

Selecting the data type of a LAD element (Page 2583)

ADD: Add

Description

You can use the "Add" instruction to add the value at input IN1 and the value at input IN2 and query the sum at output OUT (OUT := IN1+IN2).

In its initial state, the instruction box contains at least 2 inputs (IN1 and IN2). The number of inputs can be extended. The inserted inputs are numbered in ascending order in the box. When the instruction is executed, the values of all available input parameters are added. The sum is stored at the OUT output.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The enable input EN has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at the OUT output.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Add" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	First number to be added

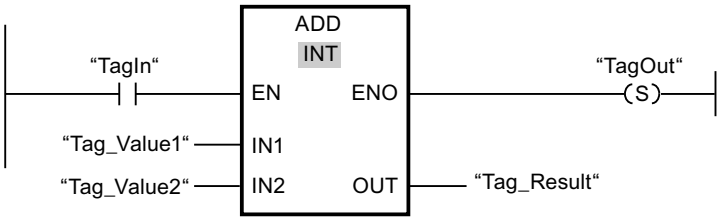
Parameter	Declaration	Data type	Memory area	Description
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Second number to be added
INn	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Optional input values that are added.
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Sum

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also":

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state "1", the "Add" instruction is executed. The value of operand "Tag_Value1" is added to the value of operand "Tag_Value2". The result of the addition is stored in operand "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Removing inputs and outputs (Page 2599)

Basics of the EN/ENO mechanism (Page 2447)

Selecting a data type (Page 2583)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Selecting the data type of a LAD element (Page 2583)

Inserting operands into LAD instructions (Page 2601)

Inserting additional inputs and outputs in LAD elements (Page 2598)

SUB: Subtract

Description

You can use the "Subtract" instruction to subtract the value at input IN2 from the value at input IN1 and query the difference at output OUT (OUT := IN1-IN2).

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at the OUT output.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Subtract" instruction:

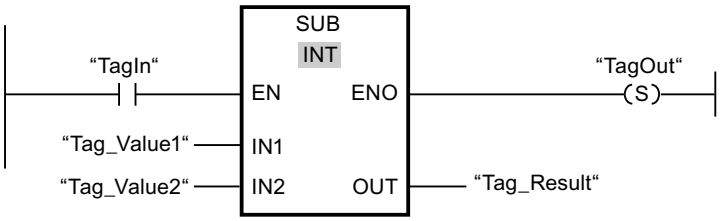
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Minuend
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Subtracting
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Difference

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state "1", the "Subtract" instruction is executed. The value of operand "Tag_Value2" is subtracted from the value of operand "Tag_Value1". The result of the subtraction is stored in operand "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Selecting a data type (Page 2583)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

MUL: Multiply

Description

You can use the "Multiply" instruction to multiply the value at input IN1 with the value at input IN2 and query the product at output OUT (OUT := IN1*IN2).

The number of inputs can be expanded in the instruction box. The added inputs are numbered in ascending order in the box. When the instruction is executed, the values of all available input parameters are multiplied. The product is stored at the OUT output.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN input has the signal state "0".
- The result is outside the range permitted for the data type specified at output OUT.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Multiply" instruction:

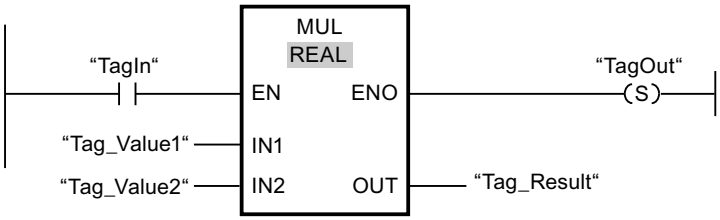
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Multiplier
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Number being multiplied
INn	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Optional input values that can be multiplied.
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Product

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state "1", the "Multiply" instruction is executed. The value of operand "Tag_Value1" is multiplied by the value of operand "Tag_Value2". The result of the multiplication is stored in operand "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Removing inputs and outputs (Page 2599)

Basics of the EN/ENO mechanism (Page 2447)

Selecting a data type (Page 2583)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Inserting additional inputs and outputs in LAD elements (Page 2598)

Selecting the data type of a LAD element (Page 2583)

DIV: Divide

Description

You can use the "Divide" instruction to divide the value at input IN1 by the value at input IN2 and query the quotient at output OUT (OUT := IN1/IN2).

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at the OUT output.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Divide" instruction:

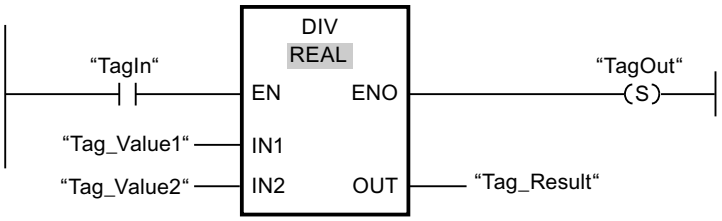
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Dividend
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Divisor
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Quotient value

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state "1", the "Divide" instruction is executed. The value of operand "Tag_Value1" is divided by the value of operand "Tag_Value2". The division result is stored in operand "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Selecting a data type (Page 2583)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

MOD: Return remainder of division

Description

You can use the "Return remainder of division" instruction to divide the value at input IN1 by the value at input IN2 and query the remainder of division at output OUT.

Parameters

The following table shows the parameters of the "Return remainder of division" instruction:

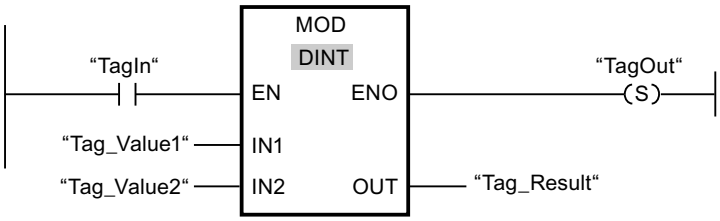
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers	I, Q, M, D, L, P or constant	Dividend
IN2	Input	Integers	I, Q, M, D, L, P or constant	Divisor
OUT	Output	Integers	I, Q, M, D, L, P	Remainder of division

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state "1", the "Return remainder of division" instruction is executed. The value of operand "Tag_Value1" is divided by the value of operand "Tag_Value2". The remainder is stored in operand "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Selecting a data type (Page 2583)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

NEG: Create twos complement

Description

You can use the "Create twos complement" instruction to change the sign of the value at the IN input and query the result at the OUT output. If there is a positive value at input IN, for example, the negative equivalent of this value is sent to output OUT.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at the OUT output.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Create twos complement" instruction:

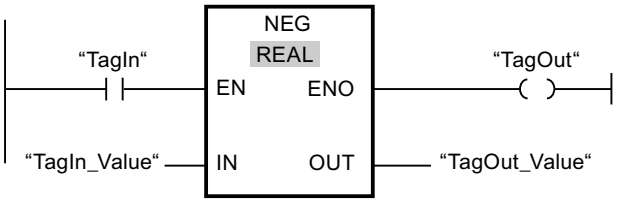
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
IN	Input	SINT, INT, DINT, floating-point numbers	SINT, INT, DINT, LINT, floating-point numbers	I, Q, M, D, L, P or constant	Input value
OUT	Output	SINT, INT, DINT, floating-point numbers	SINT, INT, DINT, LINT, floating-point numbers	I, Q, M, D, L, P	Twos complement of the input value

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state "1", the "Create twos complement" instruction is executed. The sign of the value at input "TagIn_Value" is changed and the result is provided at "TagOut_Value" output. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

INC: Increment

Description

You can use the "Increment" instruction to change the value of the operand at the IN/OUT parameter to the next higher value and query the result. The "Increment" instruction is only started when the signal state at the EN enable input is "1". If no overflow error occurs during the execution, the ENO enable output also has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Increment" instruction:

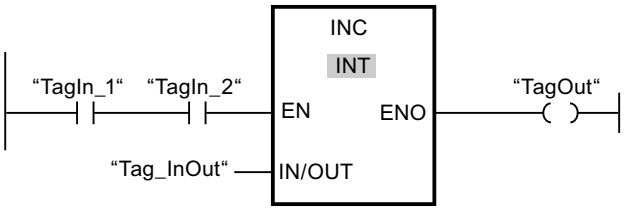
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN/OUT	InOut	Integers	I, Q, M, D, L	Value to be incremented.

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operands "TagIn_1" and "TagIn_2" have the signal state "1", the value of operand "Tag_InOut" is incremented by one and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

DEC: Decrement

Description

You can use the "Decrement" instruction to change the value of the operand at the IN/OUT parameter to the next lower value and query the result. The "Decrement" instruction is only started when the signal state at the EN enable input is "1". If the range of values of the selected data type is not exceeded during processing, the ENO output also has the signal state "1".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- A floating-point number has an invalid value.

Parameter

The following table shows the parameters of the "Decrement" instruction:

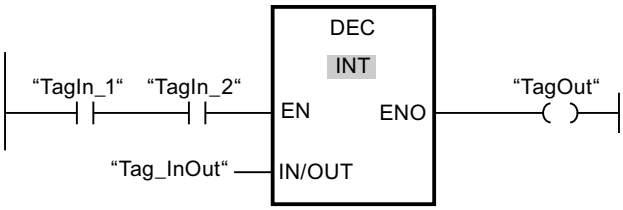
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN/OUT	InOut	Integers	I, Q, M, D, L	Value to be decremented.

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operands "TagIn_1" and "TagIn_2" have the signal state "1", the value of operand "Tag_InOut" is decremented by one and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Selecting the data type of a LAD element (Page 2583)

Inserting operands into LAD instructions (Page 2601)

ABS: Form absolute value

Description

You can use the "Form absolute value" instruction to calculate the absolute value of the value specified at input IN. The result of the instruction is sent to the OUT output and can be queried there.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Form absolute value" instruction:

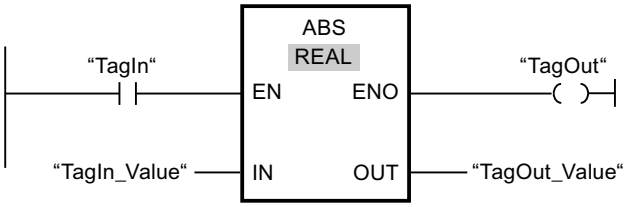
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
IN	Input	SINT, INT, DINT, floating-point numbers	SINT, INT, DINT, LINT, floating-point numbers	I, Q, M, D, L, P or constant	Input value
OUT	Output	SINT, INT, DINT, floating-point numbers	SINT, INT, DINT, LINT, floating-point numbers	I, Q, M, D, L, P	Absolute value of the input value

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	TagIn_Value	-6.234
OUT	TagOut_Value	6.234

If operand "TagIn" has the signal state "1", the "Form absolute value" instruction is executed. The instruction calculates the absolute value of the value at input "TagIn_Value" and sends the result to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

MIN: Get minimum

Description

The "Get minimum" instruction compares the values at the available inputs and writes the lowest value to the OUT output. The number of inputs can be expanded at the instruction box by additional inputs. The inputs are numbered in ascending order in the box.

A minimum of two and a maximum of 100 inputs must be specified for the execution of the instruction.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- Enable input EN has the signal state "0".
- The implicit conversion of the data types fails during execution of the instruction.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Get minimum" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output

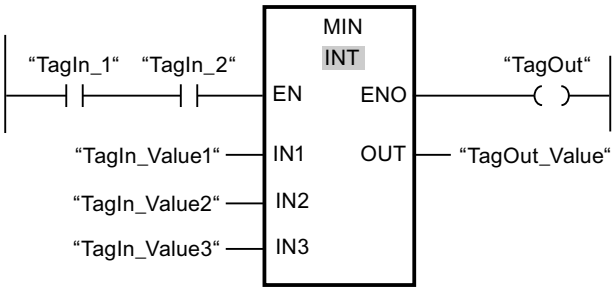
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN1	Input	Integers, floating-point numbers, TIME, TOD, DATE	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	First input value
IN2	Input	Integers, floating-point numbers, TIME, TOD, DATE	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	Second input value
INn	Input	Integers, floating-point numbers, TIME, TOD, DATE	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	Additionally inserted inputs whose values are to be compared
OUT	Output	Integers, floating-point numbers, TIME, TOD, DATE	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	Result
The data types TOD, LTOD, DATE, and LDT can only be used if the IEC test is not enabled.					

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444

Parameter	Operand	Value
IN3	TagIn_Value3	13333
OUT	TagOut_Value	12222

If the "TagIn_1" and "TagIn_2" operands have signal state "1", the "Get minimum" instruction is executed. The instruction compares the values of the specified operands and copies the lowest value ("TagIn_Value1") to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Inserting additional inputs and outputs in LAD elements (Page 2598)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

MAX: Get maximum

Description

The "Get maximum" instruction compares the values at the available inputs and writes the highest value to the OUT output. The number of inputs can be expanded at the instruction box by additional inputs. The inputs are numbered in ascending order in the box.

A minimum of two and a maximum of 100 inputs must be specified for the execution of the instruction.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- Enable input EN has the signal state "0".
- The implicit conversion of the data types fails during execution of the instruction.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Get maximum" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output

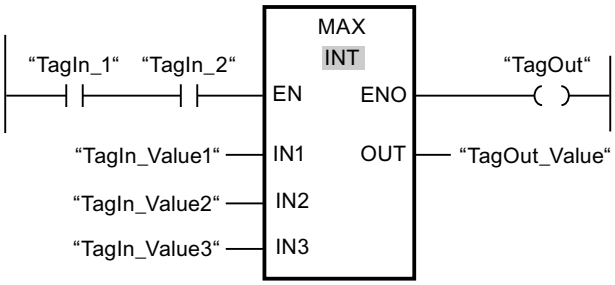
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN1	Input	Integers, floating-point numbers, TIME, TOD, DATE	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	First input value
IN2	Input	Integers, floating-point numbers, TIME, TOD, DATE	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	Second input value
INn	Input	Integers, floating-point numbers, TIME, TOD, DATE	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	Additionally inserted inputs whose values are to be compared
OUT	Output	Integers, floating-point numbers, TIME, TOD, DATE	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	Result
The data types TOD, LTOD, DATE, and LDT can only be used if the IEC test is not enabled.					

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444

Parameter	Operand	Value
IN3	TagIn_Value3	13333
OUT	TagOut_Value	14444

If the "TagIn_1" and "TagIn_2" operands have signal state "1", the "Get maximum" instruction is executed. The instruction compares the values of the specified operands and copies the highest value ("TagIn_Value2") to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Inserting additional inputs and outputs in LAD elements (Page 2598)

Removing inputs and outputs (Page 2599)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Selecting the data type of a LAD element (Page 2583)

Inserting operands into LAD instructions (Page 2601)

LIMIT: Set limit value

Description

You can use the "Set limit value" instruction to limit the value at input IN to the values at the inputs MN and MX. If the value at the IN input meets the condition MN <= IN <= MX, it is copied to the OUT output. If the condition is not fulfilled and the input value IN is below the low limit MN, the output OUT is set to the value of the MN input. If the MX high limit is exceeded, the OUT output is set to the value of the MX input.

If the value at the MN input is greater than at the MX input, the result is undefined and the enable output ENO is "0".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- Enable input EN has the signal state "0".
- The specified tags are not of the same data type.
- An operand has an invalid value.
- The value at the MN input is greater than the value of the MX input.

Parameters

The following table shows the parameters of the "Set limit value" instruction:

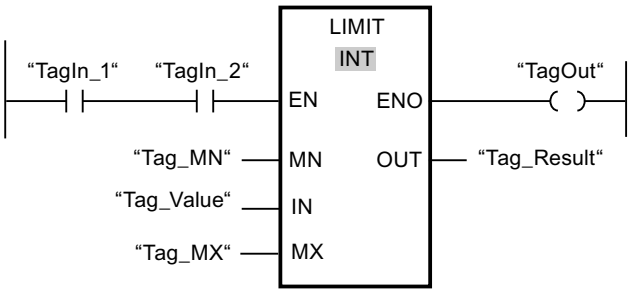
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
MN	Input	Integers, floating-point numbers, TIME, TOD, DATE	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	Low limit
IN	Input	Integers, floating-point numbers, TIME, TOD, DATE	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	Input value
MX	Input	Integers, floating-point numbers, TIME, TOD, DATE	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	High limit
OUT	Output	Integers, floating-point numbers, TIME, TOD, DATE	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	Result
The data types TOD, LTOD, DATE, and LDT can only be used if the IEC test is not enabled.					

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
MN	Tag_MN	12000
IN	Tag_Value	8000
MX	Tag_MX	16000
OUT	Tag_Result	12000

If operands "TagIn_1" and "TagIn_2" have the signal state "1", the "Set limit value" instruction is executed. The value of operand "Tag_Value" is compared with the values of operands "Tag_MN" and "Tag_MX". Since the value of operand "Tag_Value" is less than the low limit, the value of operand "Tag_MN" is copied to the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Selecting the data type of a LAD element (Page 2583)

Inserting operands into LAD instructions (Page 2601)

SQR: Form square

Description

You can use the "Form square" instruction to square the value at the IN input and query the result at the OUT output.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The value at the IN input is not a valid floating-point number.

Parameters

The following table shows the parameters of the "Form square" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

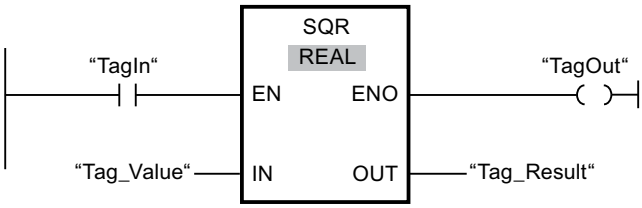
Parameter	Declaration	Data type	Memory area	Description
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Square of the input value

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_Value	5.0
OUT	Tag_Result	25.0

If operand "TagIn" has the signal state "1", the "Form square" instruction is executed. The instruction squares the value of operand "Tag_Value" and sends the result to the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

SQRT: Form square root

Description

You can use the "Form square root" instruction to find the square root of the value at the IN input and query the result at the OUT output. The instruction outputs a positive result if the

input value is greater than zero. If input values are less than zero, the OUT output returns an invalid floating-point number. If the value at input IN is "0", then the result is also "0".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The value at the IN input is not a valid floating-point number.
- The value at the IN input is negative.

Parameters

The following table shows the parameters of the "Form square root" instruction:

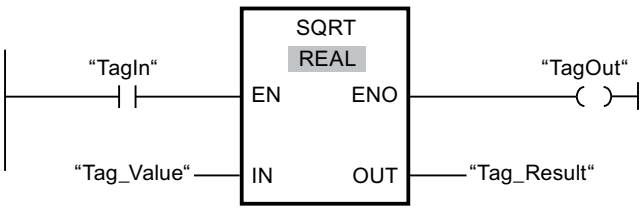
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Square root of the input value

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_Value	25.0
OUT	Tag_Result	5.0

If operand "TagIn" has the signal state "1", the "Form square root" instruction is executed. The instruction finds the square root of the value of operand "Tag_Value" and stores the result in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

LN: Form natural logarithm

Description

You can use the "Form natural logarithm" instruction to calculate the natural logarithm to base e (e = 2.718282) of the value at input IN. The result is sent to the OUT output and can be queried there. The instruction outputs a positive result if the input value is greater than zero. If input values are less than zero, the OUT output returns an invalid floating-point number.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The enable input EN has the signal state "0".
- The value at the IN input is not a valid floating-point number.
- The value at the IN input is negative.

Parameters

The following table shows the parameters of the "Form natural logarithm" instruction:

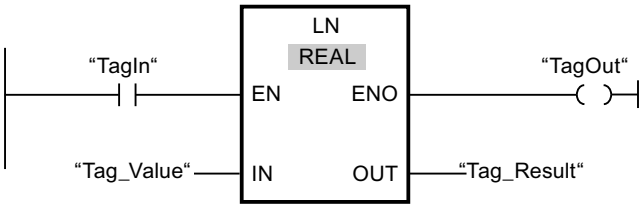
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Natural logarithm of the input value

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state "1", the "Form natural logarithm" instruction is executed. The instruction forms the natural logarithm of the value at the "Tag_Value" input and stores the result in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

EXP: Form exponential value

Description

You can use the "Form exponential value" instruction to calculate the exponent from the base e (e = 2.718282) and the value specified at input IN. The result is provided at output OUT and can be queried there (OUT = e^{IN}).

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The enable input EN has the signal state "0".
- The value at the IN input is not a valid floating-point number.

Parameters

The following table shows the parameters of the "Form exponential value" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

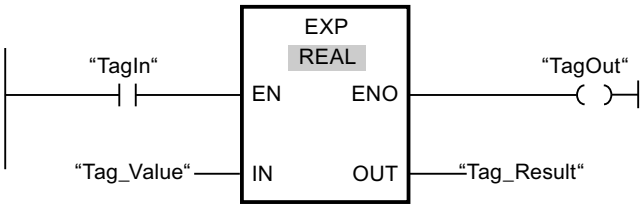
Parameters	Declaration	Data type	Memory area	Description
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Exponential value of input value IN

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state "1", the "Form exponential value" instruction is executed. The instruction calculates the exponent from base e and the value of operand "Tag_Value" and sends the result to the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

SIN: Form sine value

Description

Use the "Form sine value" instruction to calculate the sine of the angle. The size of the angle is specified in radians at the IN input. The result of the instruction is sent to the OUT output and can be queried there.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The value at the IN input is not a valid floating-point number.

Parameters

The following table shows the parameters of the "Form sine value" instruction:

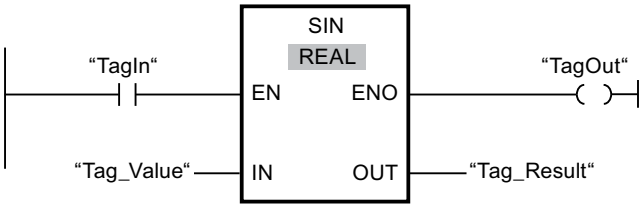
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Size of angle in radians
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Sine of the specified angle

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_Value	+1.570796 (π/2)
OUT	Tag_Result	1.0

If operand "TagIn" has the signal state "1", the "Form sine value" instruction is executed. The instruction calculates the sine of the angle specified at the "Tag_Value" input and stores the result in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

COS: Form cosine value

Description

Use the "Form cosine value" instruction to calculate the cosine of the angle. The size of the angle is specified in radians at the IN input. The result of the instruction is sent to the OUT output and can be queried there.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The value at the IN input is not a valid floating-point number.

Parameters

The following table shows the parameters of the "Form cosine value" instruction:

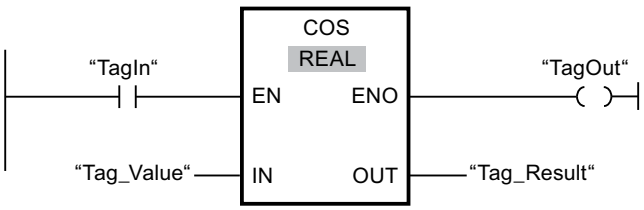
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Size of angle in radians
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Cosine of the specified angle

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_Value	+1.570796 (π/2)
OUT	Tag_Result	0

If operand "TagIn" has the signal state "1", the "Form cosine value" instruction is executed. The instruction calculates the cosine of the angle specified at the "Tag_Value" input and stores

the result in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

TAN: Form tangent value

Description

Use the "Form tangent value" instruction to calculate the tangent of the angle. The size of the angle is specified in radians at the IN input. The result of the instruction is sent to the OUT output and can be queried there.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The value at the IN input is not a valid floating-point number.

Parameters

The following table shows the parameters of the "Form tangent value" instruction:

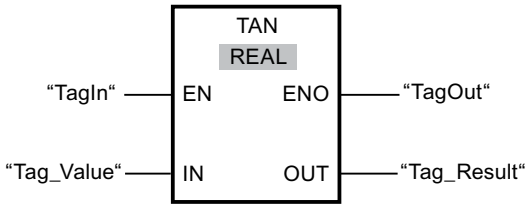
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Size of angle in radians
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Tangent of the specified angle

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_Value	+3.141593 (π)
OUT	Tag_Result	0

If operand "TagIn" has the signal state "1", the "Form tangent value" instruction is executed. The instruction calculates the tangent of the angle specified at the "Tag_Value" input and stores the result in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

ASIN: Form arcsine value

Description

You can use the "Form arcsine value" instruction to calculate the size of the angle from the arcsine value specified at input IN, which corresponds to this value. Only valid floating-point numbers within the range -1 to +1 can be specified at the IN input. The calculated angle size is output in radians at the OUT output and can range in value from -π/2 to +π/2.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The value at the IN input is not a valid floating-point number.
- The value at the IN input is outside the permitted value range (-1 to +1).

Parameters

The following table shows the parameters of the "Form arcsine value" instruction:

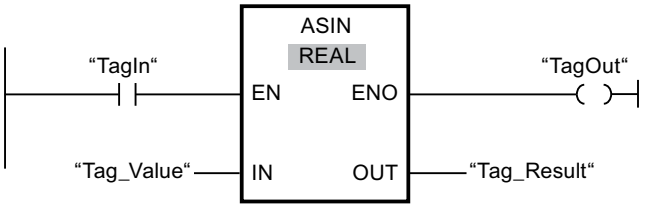
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Sine value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Size of angle in radians

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	1.0
OUT	Tag_Result	+1.570796 (π/2)

If operand "TagIn" has the signal state "1", the "Form arcsine value" instruction is executed. The instruction calculates the size of the angle corresponding to the sine value at the "Tag_Value" input. The result of the instruction is stored in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

ACOS: Form arccosine value

Description

You can use the "Form arccosine value" instruction to calculate the size of the angle from the cosine value specified at input IN, which corresponds to this value. Only valid floating-point numbers within the range -1 to +1 can be specified at the IN input. The calculated angle size is output in radians at the OUT output and can range in value from 0 to +π.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The value at the IN input is not a valid floating-point number.
- The value at the IN input is outside the permitted value range (-1 to +1).

Parameters

The following table shows the parameters of the "Form arccosine value" instruction:

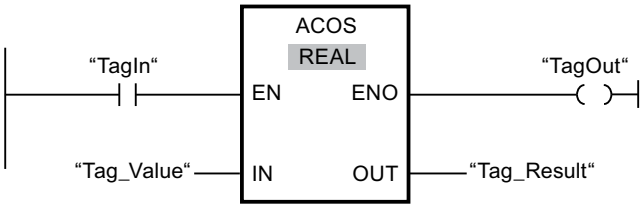
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Cosine value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Size of angle in radians

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	0
OUT	Tag_Result	+1.570796 (π/2)

If operand "TagIn" has the signal state "1", the "Form arccosine value" instruction is executed. The instruction calculates the size of the angle corresponding to the cosine value at the "Tag_Value" input. The result of the instruction is stored in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

ATAN: Form arctangent value

Description

You can use the "Form arctangent value" instruction to calculate the size of the angle from the tangent value specified at input IN, which corresponds to this value. Only valid floating-point numbers may be specified at the IN input. The calculated angle size is output in radians at the OUT output and can range in value from -π/2 to +π/2.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The value at the IN input is not a valid floating-point number.

Parameters

The following table shows the parameters of the "Form arctangent value" instruction:

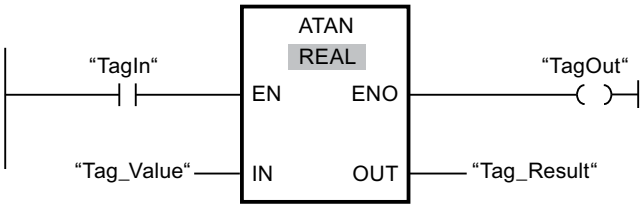
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Tangent value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Size of angle in radians

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	1.0
OUT	Tag_Result	+0.785398 (π/4)

If operand "TagIn" has the signal state"1", the "Form arctangent value" instruction is executed. The instruction calculates the size of the angle corresponding to the tangent value at the "Tag_Value" input. The result of the instruction is stored in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

FRAC: Return fraction

Description

You can use the "Return fraction" instruction to determine the decimal places of the value at the IN input. The result of the query is stored at the OUT output and can be queried there. If, for example, the IN input has the value 123.4567, the OUT output returns the value 0.4567.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the "Return fraction" instruction:

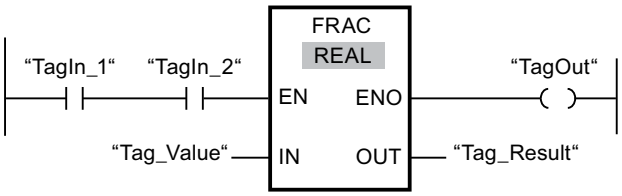
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Value, whose decimal places are to be determined.
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Decimal places of the value at the IN input

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_Value	2.555
OUT	Tag_Result	0.555

If operands "TagIn_1" and "TagIn_2" have the signal state "1", the "Return fraction" instruction is started. The decimal places from the value of operand "Tag_Value" are copied to operand "Tag_Result". If the instruction is executed without errors, the ENO output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Selecting the data type of a LAD element (Page 2583)

Inserting operands into LAD instructions (Page 2601)

EXPT: Exponentiate

Description

You can use the "Exponentiate" instruction to raise the value at the IN1 input to a power specified with the value at the IN2 input. The result of the instruction is provided at the OUT output and can be queried there (OUT =$IN1^{IN2}$).

The IN1 input can only be assigned valid floating-point numbers. Integers can also be assigned to the IN2 input.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- Errors occur during execution of the instruction, for example, an overflow occurs.

Parameters

The following table shows the parameters of the "Exponentiate" instruction:

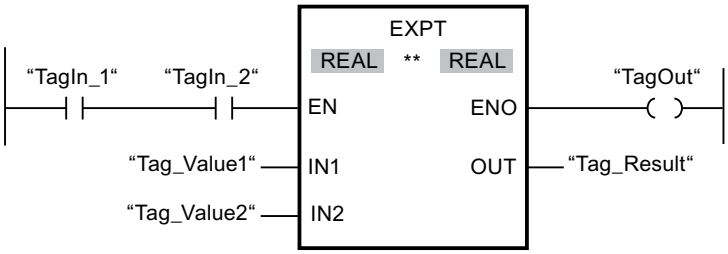
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Base value
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Value with which the base value is exponentiated
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Result

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operands "TagIn_1" and "TagIn_2" have the signal state "1", the "Exponentiate" instruction is started. The value of operand "Tag_Value1" is raised to a power specified with the value of operand "Tag_Value2". The result is stored in the "Tag_Result" output. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Selecting the data type of a LAD element (Page 2583)

Inserting operands into LAD instructions (Page 2601)

Move operations

MOVE: Move value

Description

You use the "Move value" instruction to transfer the content of the operand at the IN input to the operand at the OUT1 output. The transfer is always made in the direction of ascending address.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The data type at the IN parameter does not correspond to the specified data type at the OUT1 parameter.

The following table lists the possible transfers for the S7-1200 CPU series:

Source (IN)	Destination (OUT1)	
	With IEC check	Without IEC check
BYTE	BYTE, WORD, DWORD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE ,TOD, CHAR
WORD	WORD, DWORD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD, CHAR
DWORD	DWORD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME, DATE, TOD, CHAR
SINT	SINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
USINT	USINT, UINT, UDINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
INT	INT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
UINT	UINT, UDINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
DINT	DINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
UDINT	UDINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
REAL	REAL	DWORD, REAL
LREAL	LREAL	LREAL
TIME	TIME	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME
DATE	DATE	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, DATE
TOD	TOD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TOD
DTL	DTL	DTL
CHAR	CHAR	BYTE, WORD, DWORD, CHAR, Character of a string[1)]
Character of a string[1)]	Character of a string	Character of a string CHAR
ARRAY[2)]	ARRAY	ARRAY
STRUCT	STRUCT	STRUCT
PLC data type (UDT)	PLC data type (UDT)	PLC data type (UDT)
IEC_TIMER	IEC_TIMER	IEC_TIMER
IEC_SCOUNTER	IEC_SCOUNTER	IEC_SCOUNTER
IEC_USCOUNTER	IEC_USCOUNTER	IEC_USCOUNTER
IEC_COUNTER	IEC_COUNTER	IEC_COUNTER
IEC_UCOUNTER	IEC_UCOUNTER	IEC_UCOUNTER

Source (IN)	Destination (OUT1)	
	With IEC check	Without IEC check
IEC_DCOUNTER	IEC_DCOUNTER	IEC_DCOUNTER
IEC_UDCOUNTER	IEC_UDCOUNTER	IEC_UDCOUNTER

The following table lists the possible transfers for the S7-1500 CPU series:

Source (IN)	Destination (OUT1)	
	With IEC check	Without IEC check
BYTE	BYTE, WORD, DWORD, LWORD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE ,TOD, LTOD, CHAR
WORD	WORD, DWORD, LWORD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, S5TIME, TIME, LTIME, LDT, DATE, TOD, LTOD, CHAR
DWORD	DWORD, LWORD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, TIME, LTIME, LDT, DATE, TOD, LTOD, CHAR
LWORD	LWORD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, LREAL, TIME, LTIME, LDT, DATE, TOD, LTOD, CHAR
SINT	SINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
USINT	USINT, UINT, UDINT, ULINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
INT	INT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
UINT	UINT, UDINT, ULINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD. LTOD
DINT	DINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
UDINT	UDINT, ULINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
LINT	LINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
ULINT	ULINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
REAL	REAL	DWORD, REAL
LREAL	LREAL	LWORD, LREAL
S5TIME	S5TIME	WORD, S5TIME

Source (IN)	Destination (OUT1)	
	With IEC check	Without IEC check
TIME	TIME	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME
LTIME	LTIME	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, LTIME
DATE	DATE	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, DATE
DT	DT	DT
LDT	LDT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, LDT
TOD	TOD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TOD
LTOD	LTOD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, LTOD
DTL	DTL	DTL
CHAR	CHAR	BYTE, WORD, DWORD, LWORD, CHAR, Character of a string[1)]
WCHAR	WCHAR	BYTE, WORD, DWORD, LWORD, CHAR, WCHAR, Character of a string[1)]
Character of a string[1)]	Character of a string	CHAR, WCHAR, Character of a string
ARRAY[2)]	ARRAY	ARRAY
STRUCT	STRUCT	STRUCT
COUNTER	COUNTER, WORD, INT	WORD, DWORD, INT, UINT, DINT, UDINT
TIMER	TIMER, WORD, INT	WORD, DWORD, INT, UINT, DINT, UDINT
PLC data type (UDT)	PLC data type (UDT)	PLC data type (UDT)
IEC_TIMER	IEC_TIMER	IEC_TIMER
IEC_LTIMER	IEC_LTIMER	IEC_LTIMER
IEC_SCOUNTER	IEC_SCOUNTER	IEC_SCOUNTER
IEC_USCOUNTER	IEC_USCOUNTER	IEC_USCOUNTER
IEC_COUNTER	IEC_COUNTER	IEC_COUNTER
IEC_UCOUNTER	IEC_UCOUNTER	IEC_UCOUNTER
IEC_DCOUNTER	IEC_DCOUNTER	IEC_DCOUNTER
IEC_UDCOUNTER	IEC_UDCOUNTER	IEC_UDCOUNTER
IEC_LCOUNTER	IEC_LCOUNTER	IEC_LCOUNTER
IEC_ULCOUNTER	IEC_ULCOUNTER	IEC_ULCOUNTER

[1)] You can also use the "Move value" instruction to transfer individual characters of a string to operands of the CHAR or WCHAR data type. The number of the character to be transferred is specified in square brackets next to the operand name. "MyString[2]", for example, transfers

the second character of the "MyString" string. It is also possible to transfer from operands of the data type CHAR to the individual characters of a string. You can also replace a specific character of a string with the character of another string.

[2)] Transferring entire arrays (ARRAY) is possible only when the array components of the operands at input IN and at output OUT1 are of the same data type.

If the bit length of the data type at input IN exceeds the bit length of the data type at output OUT1, the higher-order bits of the source value are lost. If the bit length of the data type at input IN is less than the bit length of the data type at output OUT1, the higher-order bits of the destination value will be overwritten with zeros.

In its initial state, the instruction box contains 1 output (OUT1). The number of outputs can be extended. The added outputs are numbered in ascending order on the box. During the execution of the instruction, the content of the operand at the input IN is transferred to all available outputs. The instruction box cannot be extended if structured data types (DTL, STRUCT, ARRAY) or characters of a string are transferred.

You can also use the "Move block" (MOVE_BLK) and "Move block uninterruptible" (UMOVE_BLK) instructions to move operands of the ARRAY data type. You can move operands of the STRING data type with the instruction "Move character string" (S_MOVE).

Parameter

The following table lists the parameters of the "Move value" instruction:

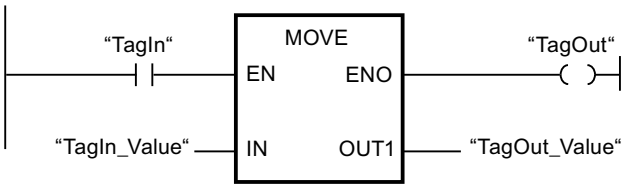
Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	Bit strings, integers, floating-point numbers, DATE, TIME, TOD, DTL, CHAR, STRUCT, ARRAY, IEC data types, PLC data type (UDT)	Bit strings, integers, floating-point numbers, DATE, DT, LDT, S5TIME, TIME, LTIME, TOD, LTOD, DTL, CHAR, STRUCT, ARRAY, TIMER, COUNTER, IEC data types, PLC data type (UDT)	I, Q, M, D, L or constant	Source value
OUT1	Output	Bit strings, integers, floating-point numbers, DATE, TIME, TOD, DTL, CHAR, STRUCT, ARRAY, IEC data types, PLC data type (UDT)	Bit strings, integers, floating-point numbers, DATE, DT, LDT, S5TIME, TIME, LTIME, TOD, LTOD, DTL, CHAR, STRUCT, ARRAY, TIMER, COUNTER, IEC data types, PLC data type (UDT)	I, Q, M, D, L	Operands to which in the source value is transferred.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
OUT1	TagOut_Value	0011 1111 1010 1111

If the operand "TagIn" has the signal state "1", the "Move value" instruction is executed. The instruction copies the contents of operand "TagIn_Value" to operand "TagOut_Value" and sets output "TagOut" to signal state "1".

See also

Overview of the valid data types (Page 2236)

Removing inputs and outputs (Page 2599)

Basics of the EN/ENO mechanism (Page 2447)

MOVE_BLK: Move block (Page 3318)

UMOVE_BLK: Move block uninterruptible (Page 3324)

S_MOVE: Move character string (Page 4519)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting additional inputs and outputs in LAD elements (Page 2598)

Inserting operands into LAD instructions (Page 2601)

Deserialize: Deserialize

Description

You can use the "Deserialize" instruction to convert the sequential representation of a PLC data type (UDT) back to a PLC data type and to fill its entire contents.

The memory area in which the sequential representation of a PLC data type is located must have the ARRAY of BYTE data type and be declared with standard access. Make sure that there is enough memory space prior to the conversion.

The instruction enables you can convert multiple sequential representations of converted PLC data types back to their original state step-by-step.

Note

If you only want to convert back a single sequential representation of a PLC data type (UDT), you can also direct the instruction "TRCV: Receive data via communication connection".

Parameters

The following table lists the parameters of the "Deserialize" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRC_ARRAY	Input	VARIANT	I, Q, D, L	Global data block in which the generated data stream is stored
DEST_VARIABLE	InOut	VARIANT	I, Q, D, L	Tag in which the returned converted PLC data type (UDT) is stored
POS	InOut	DINT	I, Q, M, D, L	Number of bytes that the converted PLC data types use. The POS parameter is calculated zero-based.
RET_VAL	Output	INT	I, Q, M, D, L	Error information

For additional information on valid data types, refer to "See also".

RET_VAL parameter

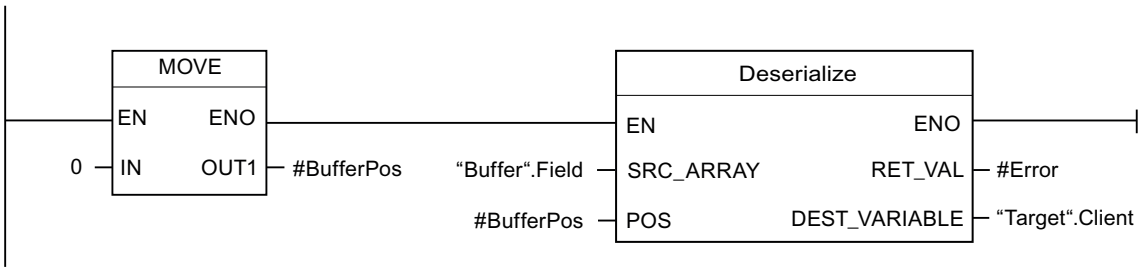
The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B0	The memory areas for the SRC_ARRAY and DEST_VARIABLE parameters overlap.
8136	The data block at the DEST_VARIABLE parameter is not a block with standard access.
8150	The VARIANT data type at the SRC_ARRAY parameter contains no value.
8151	Code generation error at the SRC_ARRAY parameter
8153	There is not enough free memory available at the SRC_ARRAY parameter.
8250	The VARIANT data type at the DEST_VARIABLE parameter contains no value.
8251	Code generation error at the DEST_VARIABLE parameter
8254	Invalid data type in the DEST_VARIABLE parameter.
8382	The value at the POS parameter is outside the limits of the array.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

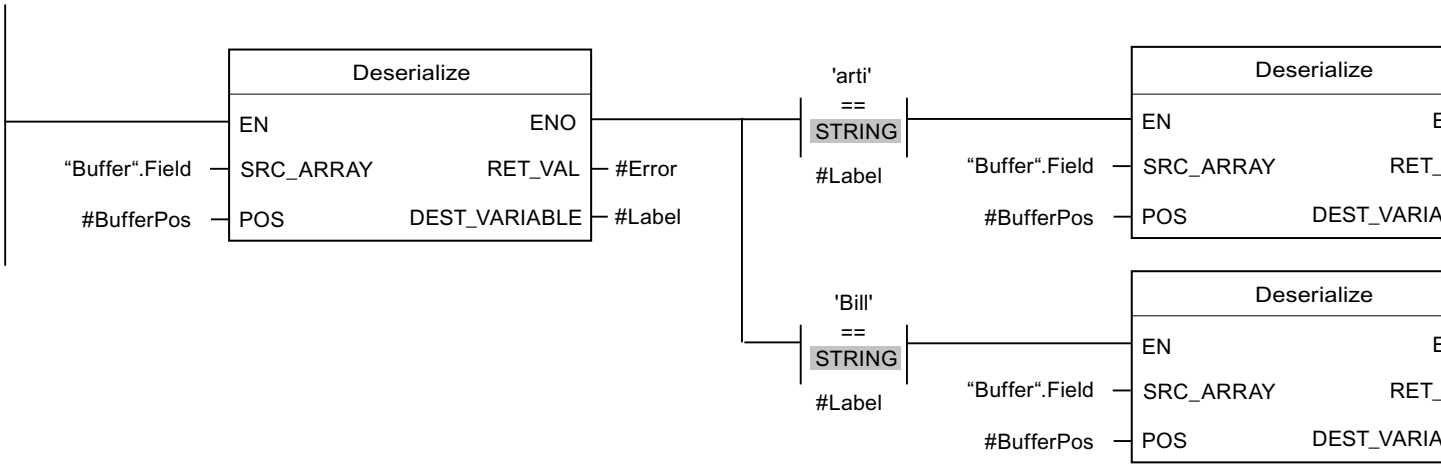
The following example shows how the instruction works:

Network 1:



The "Move value" instruction copies the value "0" to the "#BufferPos" operand. The "Deserialize" instruction deserializes the sequential representation of the customer data from the "Buffer" data block and writes it to the "Target" data block. The number of bytes that show the converted customer data is stored in the "#BufferPos" operand.

Network 2:



The "Deserialize" instruction deserializes the sequential representation of the separator sheet, which was saved with the sequential representation after the customer data, from the "Buffer" data block and writes the characters to the "#Label" operand. The characters are compared using comparison instructions "arti" and "Bill". If the comparison for "arti" = TRUE, then this is article data that has been deserialized and written to the "Target" data block. If the comparison for "Bill" = TRUE, then this is billing data that has been deserialized and written to the "Target" data block.

The following table lists the declaration of the operands:

Operand	Data type	Declaration
DeliverPos	INT	In the "Input" section of the block interface
BufferPos	DINT	In the "Temp" section of the block interface
Error	INT	In the "Temp" section of the block interface
Label	STRING[4]	In the "Temp" section of the block interface

The following table lists the declarations of the PLC data types:

Name the PLC data types	Name	Data type
Article	Number	DINT
	Declaration	STRING
	Colli	INT
Client	Title	INT
	First name	STRING[10]
	Surname	STRING[10]

The following table lists the declaration of the data blocks:

Name of the data blocks	Name	Data type
Target	Client	"Client" (PLC data type)
	Article	Array[0..] of "Article" (PLC data type)
	Bill	Array[0..10] of INT
Buffer	Field	Array[0..294] of BYTE

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Switching display formats in the program status (Page 2980)

PLC data types (Page 2280)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting additional inputs and outputs in LAD elements (Page 2598)

Inserting operands into LAD instructions (Page 2601)

Serialize: Serialize

Description

You can use the "Serialize" instruction to convert several PLC data types (UDT) to a sequential representation without losing parts of their structure.

You can use the instruction to cache multiple structured data from your program in a buffer, which is preferably in a global data block, and send it to another CPU. The memory area in which the converted PLC data types must be stored must have the ARRAY of BYTE data type and be declared with standard access. Make sure that there is enough memory space prior to the conversion.

The operand at the POS parameter contains information about how many bytes the converted PLC data types occupy.

Note

If you want to send a single PLC data type (UDT), you can directly call the instruction "TSEND: Send data via communication connection".

Parameters

The following table lists the parameters of the "Serialize" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRC_VARIABLE	Input	VARIANT	I, Q, D, L	PLC data type (UDT) that is converted to sequential representation.
DEST_ARRAY	InOut	VARIANT	I, Q, D, L	Data block in which the generated data stream is stored.
POS	InOut	DINT	I, Q, M, D, L	Number of bytes that the converted PLC data types use. The POS parameter is calculated zero-based.
RET_VAL	Output	INT	I, Q, M, D, L	Error information

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

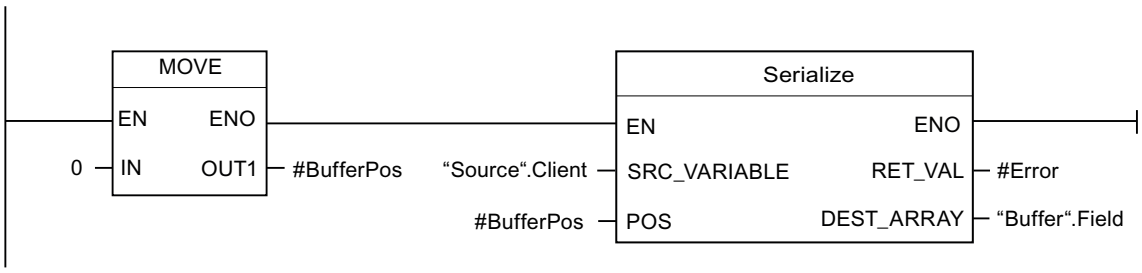
Error code* (W#16#...)	Explanation
0000	No error
80B0	The memory areas for the SRC_VARIABLE and DEST_ARRAY parameters overlap.
8150	The VARIANT data type at the SRC_VARIABLE parameter contains no value.
8152	Code generation error at the SRC_VARIABLE parameter
8236	The data block at the DEST_ARRAY parameter is not a block with standard access.
8250	The VARIANT data type at the DEST_ARRAY parameter contains no value.
8252	Code generation error at the DEST_ARRAY parameter
8253	There is not enough free memory available at the DEST_ARRAY parameter.
8254	Invalid data type in the DEST_ARRAY parameter.

Error code* (W#16#...)	Explanation
8382	The value at the POS parameter is outside the limits of the array.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

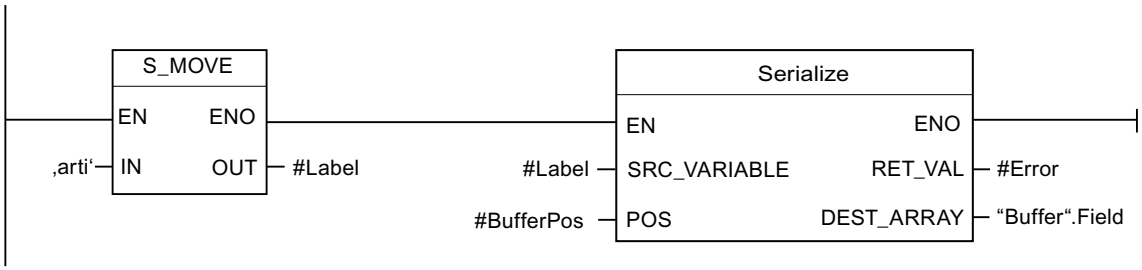
The following example shows how the instruction works:

Network 1:



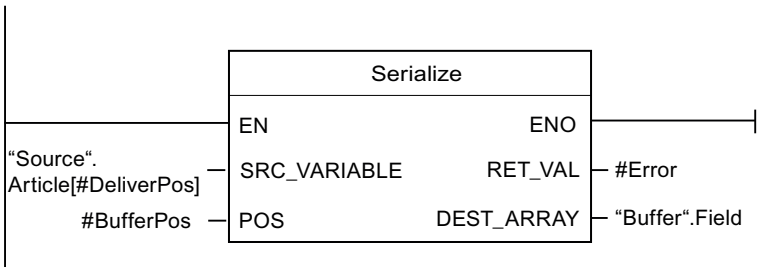
The "Move value" instruction copies the value "0" to the "#BufferPos" operand. The "Serialize" instruction serializes the customer data from the "Source" data block and writes it in sequential representation to the "Buffer" data block. The number of bytes occupied by the sequential representation is stored in the "#BufferPos" operand.

Network 2:



A kind of separator sheet is inserted now to later make it easier to deserialize the sequential representation. The "Move string" instruction copies the "arti" characters to the "#Label" operand. The "Serialize" instruction writes these characters after the customer data to the "Buffer" data block. The number of bytes that the characters need in addition to the number already stored are added to the "#BufferPos" operand.

Network 3:



The "Serialize" instruction serializes the data of a specific article, which is calculated in runtime, from the "Source" data block and writes it in sequential representation to the "Buffer" data block after the "arti" characters.

The following table lists the declaration of the operands:

Operand	Data type	Declaration
DeliverPos	INT	In the "Input" section of the block interface
BufferPos	DINT	In the "Temp" section of the block interface
Error	INT	In the "Temp" section of the block interface
Label	STRING[4]	In the "Temp" section of the block interface

The following table lists the declarations of the PLC data types:

Name the PLC data types	Name	Data type
Article	Number	DINT
	Declaration	STRING
	Colli	INT
Client	Title	INT
	First name	STRING[10]
	Surname	STRING[10]

The following table lists the declaration of the data blocks:

Name of the data blocks	Name	Data type
Source	Client	"Client" (PLC data type)
	Article	Array[0..10] of "Article" (PLC data type)
Buffer	Field	Array[0..294] of BYTE

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Switching display formats in the program status (Page 2980)

PLC data types (Page 2280)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting additional inputs and outputs in LAD elements (Page 2598)

Inserting operands into LAD instructions (Page 2601)

FieldRead: Read field

Description

You can use the "Read field" instruction to read out a specific component from the field specified at input MEMBER and transfer its content to the tag at output VALUE. You specify the index of the field component to be read at input INDEX. Specify the first component of the field from which reading is to occur at input MEMBER.

The data types of the field component at input MEMBER and the tags at output VALUE must correspond to the data type of the instruction "Read field".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The field component specified at input INDEX is not defined in the field specified at output MEMBER.
- Errors such as an overflow occur during execution.

Parameters

The following table shows the parameters of the "Read field" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
INDEX	Input	DINT	DINT	I, Q, M, D, L, P or constant	Index of field component whose content is read out

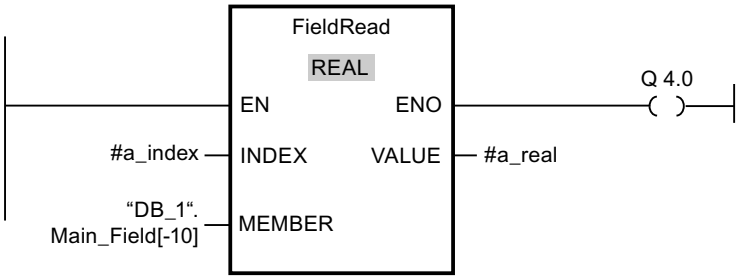
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
MEMBER	Input	Binary numbers, integers, floating-point numbers, TIME, DATE, TOD and CHAR as components of an ARRAY tag	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, LTOD and CHAR as components of an ARRAY tag	D, L	First component of the field from which reading occurs.
VALUE	Output	Bit strings, integers, floating-point numbers, TIME, DATE, TOD and CHAR	Bit strings, integers, floating-point numbers, timers, DATE, TOD, LTOD and CHAR	I, Q, M, D, L, P	Operand to which the content of the field component is transferred

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Tag	Value
INDEX	a_index	4
MEMBER	"DB_1".Main_Field[-10]	First component of the "Main_Field[-10..10] of REAL" field in the "DB_1" data block
VALUE	a_real	Component with index 4 of the "Main_Field[-10..10] of REAL" field

The field component with index 4 is read from the "Main_Field[-10...10] of REAL" field and written to the "a_real" tag. The field component to be read is specified by the value at input INDEX.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

FieldWrite: Write field

Description

The "Write field" instruction is used to transfer the content of the tag at the VALUE input to a specific component of the field at the MEMBER output. You use the value at the INDEX input to specify the index of the field component that is described. At the MEMBER output, enter the first component of the field which is to be written to.

The data types of the field component specified at the MEMBER output and the tags at the VALUE input have to match the data type of the "Write field" instruction.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The field component specified at the input INDEX is not defined in the field specified at the output MEMBER.
- Errors such as an overflow occur during execution.

Parameters

The following table shows the parameters of the "Write field" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
INDEX	Input	DINT	DINT	I, Q, M, D, L, P or constant	Index of the field component that is being written with the content of VALUE.

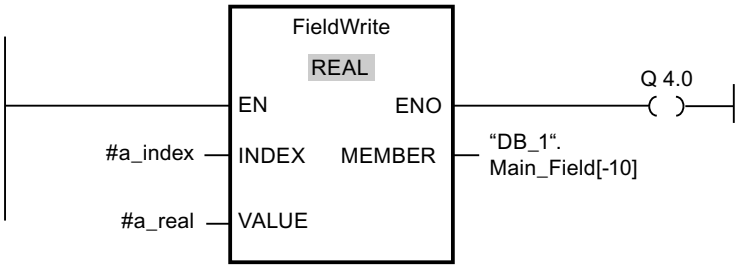
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
VALUE	Input	Bit strings, integers, floating-point numbers, TIME, DATE, TOD and CHAR	Bit strings, integers, floating-point numbers, timers, DATE, TOD, LTOD and CHAR	I, Q, M, D, L, P or constant	Operand whose content is copied.
MEMBER	Output	Binary numbers, integers, floating-point numbers, TIME, DATE, TOD and CHAR as components of an ARRAY tag	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, LTOD and CHAR as components of an ARRAY tag	D, L	First component of the field to which the content of VALUE is written.

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
INDEX	a_index	4
VALUE	a_real	10.54
MEMBER	"DB_1".Main_Field[-10]	First component of the "Main_Field[-10..10] of REAL" field in the "DB_1" data block

The value "10.54" of the "a_real" tag is written to the field component with index 4 of the "Main_Field[-10..10] of REAL" field. The index of the field component to which the content of the tag "a_real" is transferred is specified by the value at the input INDEX.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

MOVE_BLK: Move block

Description

You can use the "Move block" instruction to move the content of a memory area (source area) to another memory area (destination area). The number of elements to be moved to the destination area is specified at input COUNT. The width of the elements to be moved is defined by the width of the element at the IN input.

The instruction can only be executed if the source area and the destination area are of the same data type.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- More data is copied than is made available at the IN input or OUT output.

When a Array of BOOL is copied, the enable output ENO for an overflow is set to "1" until the byte limit of the ARRAY structure is exceeded. If the byte limit of the ARRAY structure is exceeded by the value at the COUNT input, the ENO enable output is reset to "0".

Parameters

The following table lists the parameters of the "Move block" instruction:

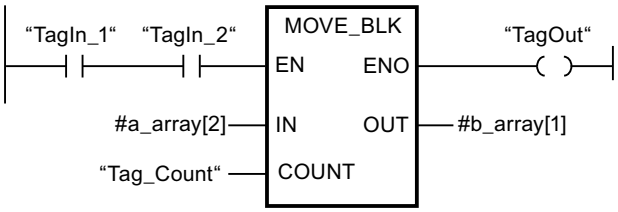
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
IN [1)]	Input	Binary numbers, integers, floating-point numbers, TIME, DATE, CHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, LTOD	D, L	The first element of the source area that is being copied

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P or constant	Number of elements to be copied from the source area to the destination area.
OUT [1]	Output	Binary numbers, integers, floating-point numbers, TIME, DATE, CHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, LTOD	D, L	The first element of the destination area to which the contents of the source area are being copied
[1] The specified data types can only be used as elements of an ARRAY structure.					

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	a_array[2]	The data type of the a_array operand is Array [0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the b_array operand is Array [0..6] of INT. It consists of seven elements of data type INT.

If operands "TagIn_1" and "TagIn_2" have the signal state "1", the "Move block" instruction is executed. Starting from the third element, the instruction selects three INT elements from the #a_array tag and copies their contents to the #b_array output tag, beginning with the second element. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting additional inputs and outputs in LAD elements (Page 2598)

Inserting operands into LAD instructions (Page 2601)

MOVE_BLK_VARIANT: Move block

Description

You can use the "Move block" instruction to move the content of a memory area (source area) to another memory area (destination area). You can copy elements of an array to another array of the same data type. The size (number of elements) of the source and destination array may be different. You can copy several elements within an array or individual elements.

If you are using the instruction, the array must not yet be known at the time the block is created, as the source and the destination are transferred using VARIANT.

The counting at the parameters SRC_INDEX and DEST_INDEX always starts with the low limit "0", regardless of the later declaration of the array.

If the VARIANT pointer (source or destination) is of the data type BOOL, it must be addressed absolutely and the length specified must be divisible by 8; otherwise the instruction is not executed.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- More data is copied than is made available.

Parameters

The following table lists the parameters of the "Move block" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRC	Input	VARIANT (array or single element)	L	Source block from which to copy

Parameter	Declaration	Data type	Memory area	Description
COUNT	Input	UDINT	I, Q, M, D, L or constant	Number of elements which are copied Set the value at the parameter COUNT to "1", if no Array is specified at the parameter SRC or the parameter DEST.
SRC_INDEX	Input	DINT	I, Q, M, D, L or constant	• The SRC_INDEX parameter is calculated zero-based. If an Array is specified at parameter SRC, the integer at the SRC_INDEX parameter specifies the first element within the source area from which copying is to take place. This is independent of the declared array limits. • If no Array is set at the SRC parameter or only an individual element of an array is specified, assign the SRC_INDEX parameter the value "0".

Parameter	Declaration	Data type	Memory area	Description
DEST_INDEX	Input	DINT	I, Q, M, D, L or constant	• The DEST_INDEX parameter is calculated zero-based. If an Array is specified at the DEST parameter, the integer at the DEST_INDEX parameter specifies the first element within the destination area to which copying is to take place. This is independent of the declared array limits. • If no Array is specified at the DEST parameter, assign the DEST_INDEX parameter the value "0".
DEST	Output	VARIANT	L	Destination area into which the contents of the source block are copied.
RET_VAL	Output	INT	I, Q, M, D, L	Error information: If an error occurs during the execution of the instruction, an error code is output at the RET_VAL parameter.

For additional information on valid data types, refer to "See also".

RET_VAL parameter

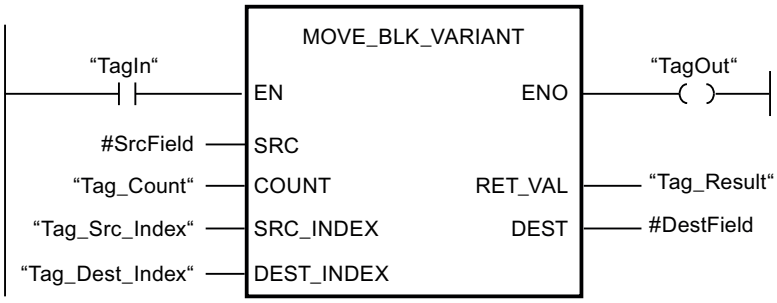
The following table lists the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	Data types do not correspond
8151	Access to the SRC parameter is not possible.
8152	The operand at the SRC parameter is not typed.
8153	Code generation error at the SRC parameter
8154	The operand at the SRC parameter has the data type BOOL.
8281	The COUNT parameter has an invalid value

Error code* (W#16#...)	Explanation
8382	The value at the SRC_INDEX parameter is outside the limits of the VARIANT.
8383	The value at the SRC_INDEX parameter is outside the high limit of the array.
8482	The value at the DEST_INDEX parameter is outside the limits of the VARIANT.
8483	The value at the DEST_INDEX parameter is outside the high limit of the array.
8534	The DEST parameter is write protected
8551	Access to the DEST parameter is not possible.
8552	The operand at the DEST parameter is not typed.
8553	Code generation error at the DEST parameter
8554	The operand at the DEST parameter has the data type BOOL.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
SRC	#SrcField	The local operand #SrcField uses a UDT that was still unknown at the time when the block was programmed. (Array [0 to 10] of "MOVE_UDT"
COUNT	Tag_Count	2
SRC_INDEX	Tag_Src_Index	3
DEST_INDEX	Tag_Dest_Index	3
DEST	#DestField	The local operand #DestField uses a UDT that was still unknown at the time when the block was programmed. (Array [10 to 20] of "MOVE_UDT"

If the operand "TagIn" has the signal state "1", the "Move block" instruction is executed. 2 elements are copied from the source area into the destination area beginning with the fourth element of the array of UDT. The copies are inserted in the array of UDT beginning with the fourth element. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Switching display formats in the program status (Page 2980)

VariantGet: Read out VARIANT tag value (Page 3343)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting additional inputs and outputs in LAD elements (Page 2598)

Inserting operands into LAD instructions (Page 2601)

UMOVE_BLK: Move block uninterruptible

Description

You can use the "Move block uninterruptible" instruction to move the content of a memory area (source area) to another memory area (destination area). The number of elements to be moved to the destination area is specified with the COUNT parameter. The width of the elements to be moved is defined by the width of the element at the IN input.

The instruction can only be executed if the source area and the destination area have the same data type.

Note

The copy operation cannot be interrupted by other operating system activities. This is why the interrupt reaction times of the CPU increase during the execution of the "Move block uninterruptible" instruction.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- More data is copied than is made available at the IN input or OUT output.

When a Array of BOOL is copied, the enable output ENO for an overflow is set to "1" until the byte limit of the ARRAY structure is exceeded. If the byte limit of the ARRAY structure is exceeded by the value at the COUNT input, the ENO enable output is reset to "0".

You can use the "Move block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Parameters

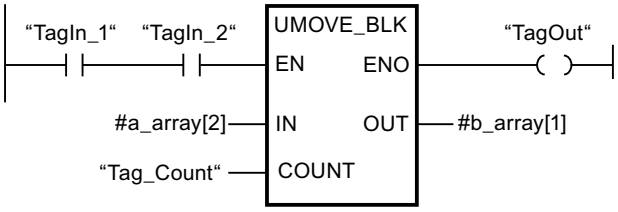
The following table shows the parameters of the "Move block uninterruptible" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
IN [1]	Input	Binary numbers, integers, floating-point numbers, TIME, DATE, CHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, LTOD	D, L	The first element of the source area that is being copied
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P or constant	Number of elements to be moved from the source area to the destination area.
OUT [1]	Output	Binary numbers, integers, floating-point numbers, TIME, DATE, CHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, LTOD	D, L	The first element of the destination area to which the contents of the source area are being copied
[1] The specified data types can only be used as elements of an ARRAY structure.					

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	a_array[2]	The data type of the a_array operand is Array [0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the b_array operand is Array [0..6] of INT. It consists of seven elements of data type INT.

If operands "TagIn_1" and "TagIn_2" have the signal state "1", the "Move block uninterruptible" instruction is executed. Starting from the third element, the instruction selects three INT elements from the #a_array tag and copies their contents to the #b_array output tag, beginning with the second element. The copy operation cannot be interrupted by other operating system activities. If the instruction is executed without errors, the ENO output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Inserting additional inputs and outputs in LAD elements (Page 2598)

FILL_BLK: Fill block

Description

You can use the "Fill block" instruction to fill a memory area (destination area) with the value of the IN input. The destination area is filled starting from the address specified at the OUT output. The number of repeated copy operations is specified with the COUNT parameter. When the instruction is executed, the value at the IN input is selected and copied to the destination area as often as specified by the value in the COUNT parameter.

The instruction can only be executed if the source area and the destination area are of the same data type.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- More data is copied than is made available at the IN input or OUT output.

When a Array of BOOL is copied, the enable output ENO for an overflow is set to "1" until the byte limit of the ARRAY structure is exceeded. If the byte limit of the ARRAY structure is exceeded by the value at the COUNT input, the ENO enable output is reset to "0".

Parameters

The following table shows the parameters of the "Fill block" instruction:

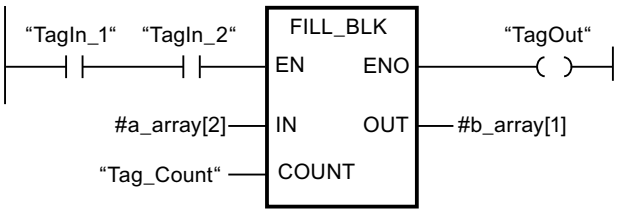
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
IN [1]	Input	Binary numbers, integers, floating-point numbers, TIME, DATE, TOD, CHAR	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, TOD, LTOD	I, Q, M, D, L, P or constant	Element used to fill the destination area
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P or constant	Number of repeated move operations
OUT [2]	Output	Binary numbers, integers, floating-point numbers, TIME, DATE, TOD, CHAR	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, TOD, LTOD	D, L	Address in destination area from which filling starts.

[1] The specified data types can also be used as elements of an ARRAY structure.
[2] The specified data types can only be used as elements of an ARRAY structure.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	a_array[2]	The data type of the a_array operand is Array [0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the b_array operand is Array [0..6] of INT. It consists of seven elements of data type INT.

If operands "TagIn_1" and "TagIn_2" have the signal state "1", the "Fill block" instruction is executed. The instruction copies the third element (#a_array[2]) of the #a_array tag three times to the #b_array output tag. If the instruction is executed without errors, the ENO and "TagOut" enable outputs are set to signal state "1".

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Inserting additional inputs and outputs in LAD elements (Page 2598)

UFILL_BLK: Fill block uninterruptible

Description

You can use the "Fill block uninterruptible" instruction to fill a memory area (destination area) with the value of the IN input without interruption. The destination area is filled starting from the address specified at the OUT output. The number of repeated copy operations is specified with the COUNT parameter. When the instruction is executed, the value at the IN input is selected and copied to the destination area as often as specified by the value in the COUNT parameter.

The instruction can only be executed if the source area and the destination area have the same data type.

Note

The copy operation cannot be interrupted by other operating system activities. This is why the alarm reaction times of the CPU increase during the execution of the "Fill block uninterruptible" instruction.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- More data is copied than is made available at the IN input or OUT output.

When a Array of BOOL is copied, the enable output ENO for an overflow is set to "1" until the byte limit of the ARRAY structure is exceeded. If the byte limit of the ARRAY structure is exceeded by the value at the COUNT input, the ENO enable output is reset to "0".

You can use the "Fill block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Parameters

The following table shows the parameters of the "Fill block uninterruptible" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
IN [1]	Input	Binary numbers, integers, floating-point numbers, TIME, DATE, CHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, TOD, LTOD	I, Q, M, D, L, P or constant	Element used to fill the destination area.
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P or constant	Number of repeated move operations
OUT [2]	Output	Binary numbers, integers, floating-point numbers, TIME, DATE, CHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, TOD, LTOD	D, L	Address in destination area from which filling begins.

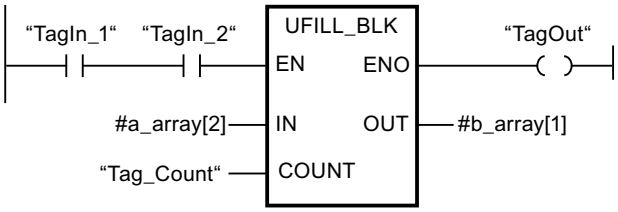
[1] The specified data types can also be used as elements of an ARRAY structure.

[2] The specified data types can only be used as elements of an ARRAY structure.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	a_array[2]	The data type of the a_array operand is Array [0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the b_array operand is Array [0..6] of INT. It consists of seven elements of data type INT.

If operands "TagIn_1" and "TagIn_2" have the signal state "1", the "Fill block uninterruptible" instruction is executed. The instruction copies the third element (#a_array[2]) of the #a_array tag three times to the #b_array output tag. The copy operation cannot be interrupted by other operating system activities. If the instruction is executed without errors, the ENO and "TagOut" enable outputs are set to signal state "1".

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

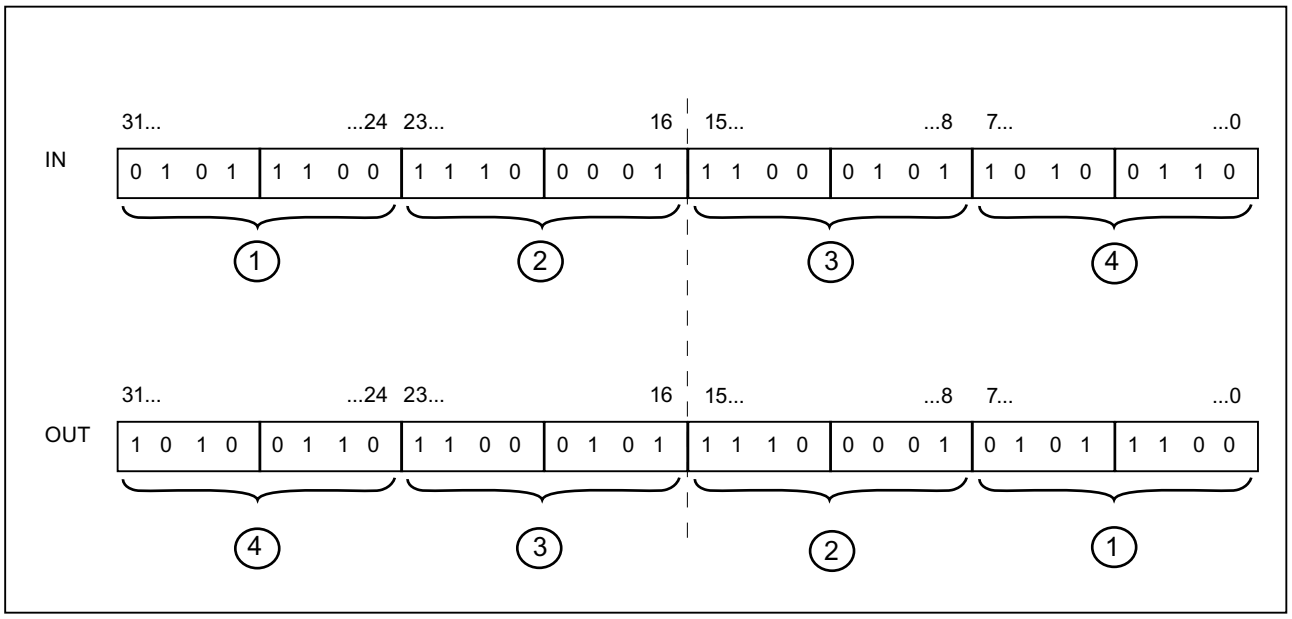
Inserting additional inputs and outputs in LAD elements (Page 2598)

SWAP: Swap

Description

You can use the "Swap" instruction to change the order of the bytes at input IN and query the result at output OUT.

The following figure shows how the bytes of an operand of the DWORD data type are swapped using the "Swap" instruction:



Parameter

The following table lists the parameters of the "Swap" instruction:

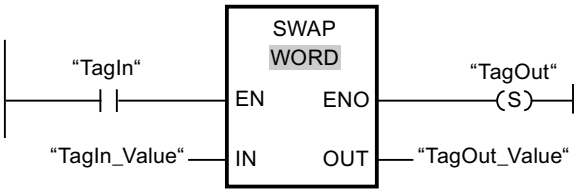
Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
IN	Input	WORD, DWORD	WORD, DWORD, LWORD	I, Q, M, D, L or, P constant	Operand whose bytes are swapped.
OUT	Output	WORD, DWORD	WORD, DWORD, LWORD	I, Q, M, D, L, P	Result

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	TagIn_Value	0000 1111 0101 0101
OUT	TagOut_Value	0101 0101 0000 1111

If operand "TagIn" has the signal state "1", the "Swap" instruction is executed. The order of the bytes is changed and stored in operand "TagOut_Value".

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

Inserting additional inputs and outputs in LAD elements (Page 2598)

ARRAY DB instructions

ReadFromArrayDB: Read from array data block

Description

You can use the "Read from array data block" instruction to read data from an array DB and write them to a destination area.

An array DB is a data block which consists of exactly one array of [data type]. The elements of the array can be UDT data type or any other elementary data type. The counting of the array always starts with the low limit "0", regardless of the later declaration of the array.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- If an error occurs during the execution of the instruction.

Parameters

The following table shows the parameters of the "Read from array data block" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

Parameter	Declaration	Data type	Memory area	Description
DB	Input	DB_ANY	I, Q, M, D, L	Data block that is read
INDEX	Input	DINT	I, Q, M, D, L, P or constant	Element that is read
VALUE	Output	VARIANT	L	Value that is read and output
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output at the RET_VAL parameter.

For additional information on valid data types, refer to "See also".

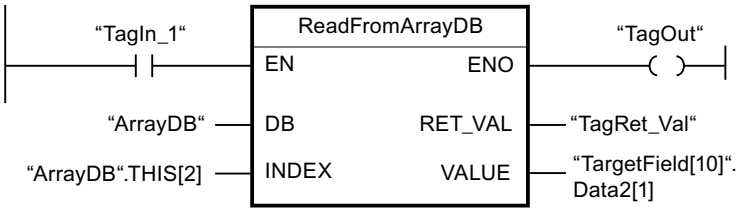
RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	Data types do not correspond
80B5	Copying process was interrupted
8132	The data block does not exist or is too short
8133	The data block is located in the load memory
8134	The data block is write protected
8135	The array DB contains invalid values
8154	The data type is not correctly declared
8282	The value at the INDEX parameter is outside the limits of the array.
8350	The VARIANT data type contains no value
8352	Code generation error
8353	Code generation error
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
DB	ArrayDB	The data type of the ArrayDB operand is Array [0 to 10] of INT.
INDEX	ArrayDB.THIS[2]	Third element of the "ArrayDB"
VALUE	TargetField[10].Data2[1]	The data type of the TargetField operand is Array [10 to 20] of "MOVE_UDT".

If the "TagIn1" operand has the signal state "1", the "Read from the ARRAY data block" instruction is executed. The third element is read from "ArrayDB" and written to the "TargetField[10].Data2[1]" operand. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Switching display formats in the program status (Page 2980)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting additional inputs and outputs in LAD elements (Page 2598)

Inserting operands into LAD instructions (Page 2601)

WriteToArrayDB: Write to array data block

Description

You can use the "Write to array data block" instruction to write data to an array DB.

An array DB is a data block which consists of exactly one array of [data type]. The elements of the array can be UDT data type or any other elementary data type. The counting of the array always starts with the low limit "0", regardless of the later declaration of the array.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- If an error occurs during the execution of the instruction.

Parameters

The following table shows the parameters of the "Write to array data block" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
DB	Input	DB_ANY	I, Q, M, D, L	Data block to which data is written
INDEX	Input	DINT	I, Q, M, D, L, P or constant	Element in the DB to which data is written
VALUE	Input	VARIANT	L	Value to be written
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output at the RET_VAL parameter.

For additional information on valid data types, refer to "See also".

RET_VAL parameter

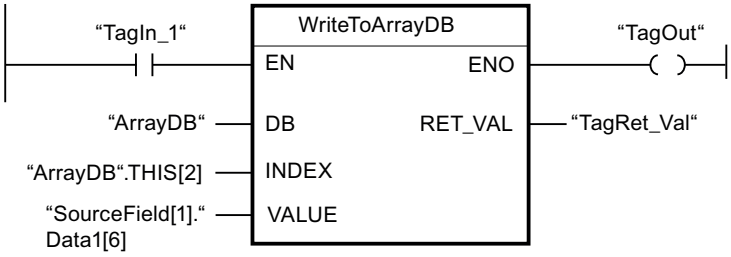
The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	Data types do not correspond
80B5	Copying process was interrupted
8132	The data block does not exist or is too short
8133	The data block is located in the load memory
8134	The data block is write protected
8135	The DB is not an array data block.
8282	The value at the INDEX parameter is outside the limits of the array.
8350	The VARIANT data type contains no value
8352	Code generation error

Error code* (W#16#...)	Explanation
8353	Code generation error
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
DB	ArrayDB	The data type of the ArrayDB operand is Array [0 to 10] of INT.
INDEX	ArrayDB.THIS[2]	Third element of the "ArrayDB"
VALUE	SourceField[1].Data1[6]	The data type of the SourceField operand is Array [0 to 10] of "MOVE_UDT".

If the "TagIn1" operand has the signal state "1", the "Write to the ARRAY data block" instruction is executed. From the "SourceField" operand, the "Data1[6]" element from the second element is written to the "ArrayDB". The third element is written to the "ArrayDB". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Switching display formats in the program status (Page 2980)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting additional inputs and outputs in LAD elements (Page 2598)

Inserting operands into LAD instructions (Page 2601)

ReadFromArrayDBL: Read from array data block in load memory

Description

You can use the "Read from array data block in load memory" instruction to read data from an array DB in load memory.

An array DB is a data block which consists of exactly one array of [data type]. The elements of the array can be UDT data type or any other elementary data type. The counting of the array always starts with the low limit "0", regardless of the later declaration of the array.

If the array DB is marked with the "Only store in load memory" block attribute, it is only stored in load memory.

The instruction is executed when a positive signal edge is detected at the REQ parameter. The BUSY parameter then has the signal state "1". The instruction is terminated if a negative signal edge is detected at the BUSY parameter. The DONE parameter has the signal state "1" for one program cycle and the read value is output at the VALUE parameter within this cycle. With all other program cycles, the value at the VALUE parameter is not changed.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Note

The array DB must be located in a data block with the block property "Optimized".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- If an error occurs during the execution of the instruction.

Parameters

The following table lists the parameters of the "Read from array data block in load memory" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = "1": Starting with the reading of the array DB
DB	Input	DB_ANY	I, Q, M, D, L	Data block that is read
INDEX	Input	DINT	I, Q, M, D, L, P or constant	Element that is read

Parameter	Declaration	Data type	Memory area	Description
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = "1": The array DB is still being read
DONE	Output	BOOL	I, Q, M, D, L	DONE = "1": The instruction was executed successfully
VALUE	InOut	VARIANT	D, L	Value that is read and output
ERROR	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output at the ERROR parameter.

For additional information on valid data types, refer to "See also".

ERROR parameter

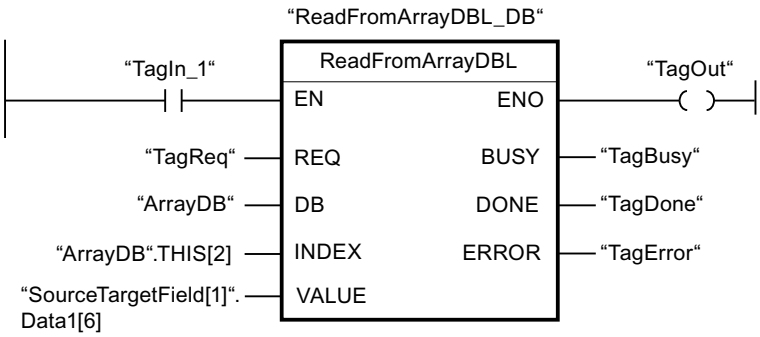
The following table shows the meaning of the values of the ERROR parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	Data types do not correspond
80B5	Copying process was interrupted
80C3	Too many concurrent calls
8132	The data block does not exist or is too short
8133	The data block is not located in the load memory
8134	The data block is write protected
8135	The DB is not an array data block.
8282	The value at the INDEX parameter is outside the limits of the array.
8350	The VARIANT data type contains no value
8352	Code generation error
8353	Code generation error
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

You can find descriptions of the error codes triggered by the instructions "READ_DBL: Read from data block in the load memory" and "WRIT_DBL: Write to data block in the load memory" under "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
REQ	TagReq	BOOL
DB	ArrayDB	The data type of the ArrayDB operand is Array [0 to 10] of INT.
INDEX	ArrayDB.THIS[2]	Third element of the "ArrayDB"
VALUE	SourceTargetField[1].Data1[6]	The data type of the SourceTargetField operand is Array [0 to 10] of "MOVE_UDT".
BUSY	TagBusy	BOOL
DONE	TagDone	BOOL

If the "TagIn1" operand has the signal state "1" and a positive edge is detected at the "TagReq"operand, the "Read from ARRAY data block in load memory" instruction is executed. The third element is read from "ArrayDB" and written to the "SourceTargetField[1].Data1[6]" operand. As soon as a negative signal edge is detected at the "TagBusy" operand, the instruction is terminated and the value at the VALUE parameter is no longer changed. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set. After the instruction has been processed, the "TagDone" operand the signal state "1".

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Switching display formats in the program status (Page 2980)

READ_DBL: Read from data block in the load memory (Page 4801)

WRIT_DBL: Write to data block in the load memory (Page 4803)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting additional inputs and outputs in LAD elements (Page 2598)

Inserting operands into LAD instructions (Page 2601)

WriteToArrayDBL: Write to array data block in load memory

Description

You can use the "Write to array data block in load memory" instruction to write data to an array DB in load memory.

An array DB is a data block which consists of exactly one array of [data type]. The elements of the array can be UDT data type or any other elementary data type. The counting of the array always starts with the low limit "0", regardless of the later declaration of the array.

If the array DB is marked with the "Only store in load memory" block attribute, it is only stored in load memory.

The instruction is executed when a positive signal edge is detected at the REQ parameter. The BUSY parameter then has the signal state "1". If a negative signal edge is detected at the BUSY parameter, the instruction is terminated and the value at the VALUE parameter is written to the data block. The DONE parameter then has the signal state "1" for one program cycle.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Note

The array DB must be located in a data block with the block property "Optimized".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- If an error occurs during the execution of the instruction.

Parameters

The following table lists the parameters of the "Write to array data block in load memory" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = "1": Start writing into the array DB

Parameter	Declaration	Data type	Memory area	Description
DB	Input	DB_ANY	D	Data block to which data is written
INDEX	Input	DINT	I, Q, M, D, L, P or constant	Element in the DB to which data is written
VALUE	Input	VARIANT	D, L	Value to be written
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = "1": The array DB is still being written
DONE	Output	BOOL	I, Q, M, D, L	DONE = "1": The instruction was executed successfully
ERROR	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output at the ERROR parameter.

For additional information on valid data types, refer to "See also".

ERROR parameter

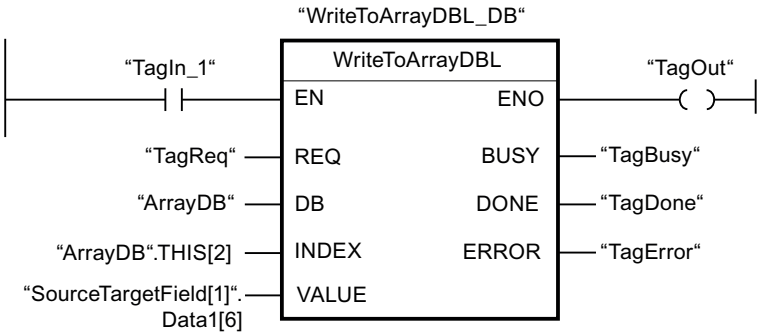
The following table shows the meaning of the values of the ERROR parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	Data types do not correspond
80B5	Copying process was interrupted
80C3	Too many concurrent calls
8132	The data block does not exist or is too short
8133	The data block is not located in the load memory
8134	The data block is write protected
8135	The DB is not an array data block.
8282	The value at the INDEX parameter is outside the limits of the array.
8350	The VARIANT data type contains no value
8352	Code generation error
8353	Code generation error
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

You can find descriptions of the error codes triggered by the instructions "READ_DBL: Read from data block in the load memory" and "WRIT_DBL: Write to data block in the load memory" under "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
REQ	TagReq	BOOL
DB	ArrayDB	The data type of the ArrayDB operand is Array [0 to 10] of INT.
INDEX	ArrayDB.THIS[2]	Third element of the "ArrayDB"
VALUE	SourceTargetField[1].Data1[6]	The data type of the SourceTargetField operand is Array [0 to 10] of "MOVE_UDT".
BUSY	TagBusy	BOOL
DONE	TagDone	BOOL

If the "TagIn1" operand has the signal state "1" and a positive edge is detected at the "TagReq"operand, the "Write to ARRAY data block in load memory" instruction is executed. As soon as a negative signal edge is detected at the "TagBusy" operand, the instruction is terminated and the value at the VALUE parameter is written in the third element in the "ArrayDB". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set. After the instruction has been processed, the "TagDone" operand the signal state "1".

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Switching display formats in the program status (Page 2980)

READ_DBL: Read from data block in the load memory (Page 4801)

WRIT_DBL: Write to data block in the load memory (Page 4803)

Inserting LAD elements (Page 2579)
Editing LAD elements (Page 2594)
Inserting additional inputs and outputs in LAD elements (Page 2598)
Inserting operands into LAD instructions (Page 2601)

VARIANT instructions

VariantGet: Read out VARIANT tag value

Description

You can use the "Read out VARIANT tag value" instruction to read the value of the tag that points to the VARIANT at the SRC parameter, and write it to the tag at the DST parameter.

The SRC parameter must have the same VARIANT data type. Any data type except for VARIANT can be specified at the DST parameter.

The data type of the tag at the DST parameter must match the data type to which the VARIANT points.

Note

The tag to which the VARIANT at parameter SRC points must be a PLC data type.

To copy structures and ARRAYs, you can use the "MOVE_BLK_VARIANT: Move block" instruction. For additional information, refer to "See also".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The data types do not correspond. (No value is transferred.)

Parameter

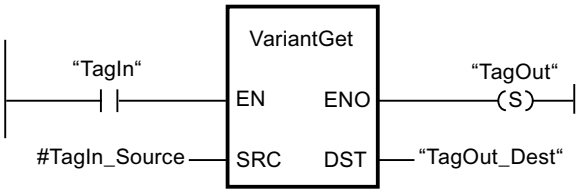
The following table lists the parameters of the "Read out VARIANT tag value" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRC	Input	VARIANT	L	Tag to be read
DST	Output	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the "TagIn" operand has the signal state "1", the "Read out VARIANT tag value" instruction is executed. The value of the tag to which the VARIANT at the "#TagIn_Source" operand points, is read and written to the "TagOut_Dest" operand.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Switching display formats in the program status (Page 2980)

VARIANT (Page 2277)

MOVE_BLK_VARIANT: Move block (Page 3320)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting additional inputs and outputs in LAD elements (Page 2598)

Inserting operands into LAD instructions (Page 2601)

VariantPut: Write VARIANT tag value

Description

You can use the "Write VARIANT tag value" instruction to write the value of the tag at the SRC parameter and to the tag at the DST parameter to which the VARIANT points.

The DST parameter must have the same VARIANT data type. Any data type except for VARIANT can be specified at the SRC parameter.

The data type of the tag at the SRC parameter must match the data type to which the VARIANT points.

Note

The tag to which the VARIANT at parameter DST points must be a PLC data type.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The data types do not correspond. (No value is transferred.)

Parameters

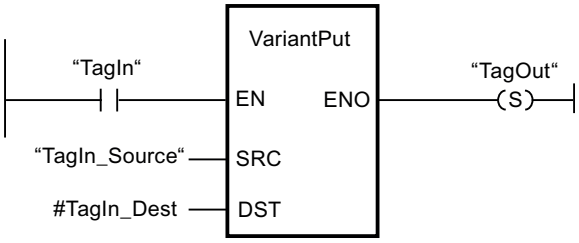
The following table lists the parameters of the "Write VARIANT tag value" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRC	Input	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY	I, Q, M, D, L	Tag to be read
DST	Input	VARIANT	L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the "TagIn" operand has the signal state "1", the "Write VARIANT tag value" instruction is executed. The value of the "TagIn_Source" operand is written to the tag to which the VARIANT at the "#TagIn_Dest" operand points.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Switching display formats in the program status (Page 2980)

VARIANT (Page 2277)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting additional inputs and outputs in LAD elements (Page 2598)

Inserting operands into LAD instructions (Page 2601)

CountOfElements: Get number of ARRAY elements

Description

You can use the "Get number of ARRAY elements" instruction to query how many ARRAY elements a tag pointing to an VARIANT has.

If it is a one-dimensional ARRAY, the difference between the high and low limit +1 is output as a result. If it is a multi-dimensional ARRAY, the product of all dimensions is output as the result.

The IN parameter must have the same VARIANT data type.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The VARIANT tag is not an ARRAY. (The result is "0".)

If the VARIANT points to an ARRAY of BOOL, the fill elements are also included in the count. (For example, 8 is returned for an ARRAY[0..1] of BOOL)

Parameters

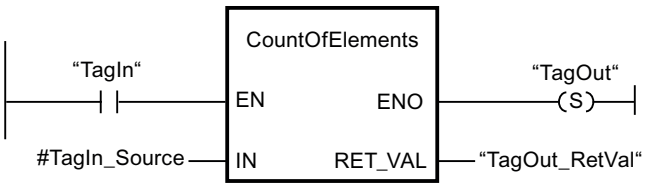
The following table lists the parameters of the "Get number of ARRAY elements" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	VARIANT	L	Tag to be queried
RET_VAL	Output	UDINT	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the "TagIn" operand has the signal state "1", the "Get number of ARRAY elements" instruction is executed. The number of ARRAY elements of the tag to which the VARIANT at the "#TagIn_Source" operand points is read and output at the "TagOut_RetVal" operand.

See also

Format of array (16-bit limits) (Page 2267)
Format of array (32-bit limits) (Page 2268)
Overview of the valid data types (Page 2236)
Basics of the EN/ENO mechanism (Page 2447)
Switching display formats in the program status (Page 2980)
VARIANT (Page 2277)
Inserting LAD elements (Page 2579)
Editing LAD elements (Page 2594)
Inserting additional inputs and outputs in LAD elements (Page 2598)
Inserting operands into LAD instructions (Page 2601)

Other

BLKMOV: Move block

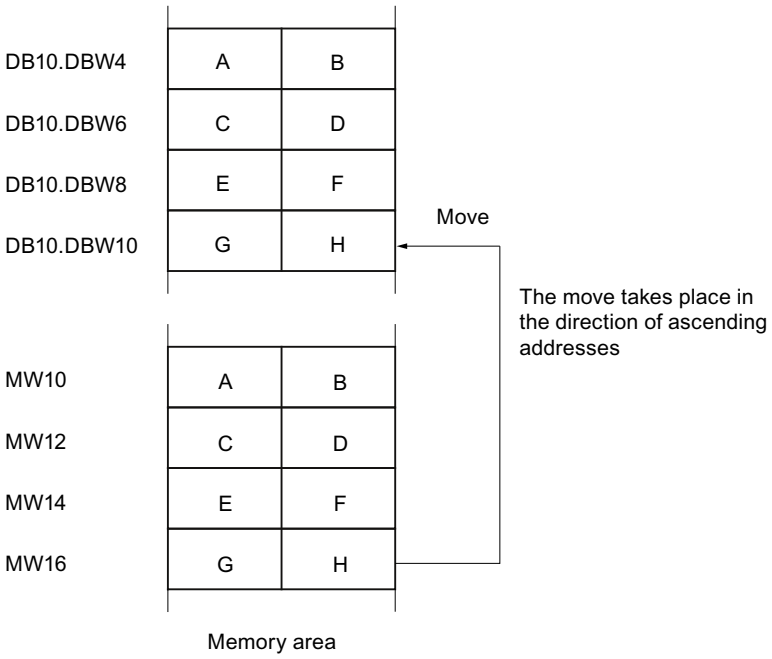
Description

You can use the "Move block" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use VARIANT to define the source and destination area.

Note

The tags of the instruction can only be used in data blocks in which the "Optimized block access" attribute is not set. If the tag with the retentivity setting "Set in IDB" has been declared, you can also use the tags "with optimized access".

The following figure shows the principle of the move operation:



Consistency of the source data and the target data

Make sure that the source data remain unchanged during the execution of the "Move block" instruction. Otherwise the consistency of the target data cannot be ensured.

Interruptibility

There is no limit to the nesting depth.

Memory areas

You can use the "Move block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image of the inputs
- Process image of the outputs

General rules for moving

The source and destination area must not overlap. If the source and destination area have different lengths, only the length of the smaller area will be moved.

If the source area is less than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is less than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a block of data type BOOL is moved, the tag must be addressed absolutely and the specified length of the area must be divisible by 8, otherwise the instruction cannot be executed.

Rules for moving character strings

You can use the "Move block" instruction to also move source and destination areas of the STRING data type. If only the source area is STRING data type, the characters will be moved that are actually contained in the character string. Information on the actual and maximum length is also written to the destination area. If the source and destination area are each STRING data type, the current length of the character string in the destination area is set to the number of actually moved characters.

If you want to move the information on the maximum and actual length of a character string, specify the areas in bytes to the SRCBLK and DSTBLK parameters.

Parameters

The following table lists the parameters of the "Move block" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRCBLK	Input	VARIANT	L, P	Specifies the memory area to be moved (source area).
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output at the RET_VAL parameter.
DSTBLK	Output	VARIANT	L, P	Specifies the memory area to which the block is to be moved (destination area).

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8152	The WSTRING, WCHAR and BOOL data types are not supported.
8092	The source or target area is only available in the load memory.
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)
Evaluating errors with output parameter RET_VAL (Page 5370)
Basics of the EN/ENO mechanism (Page 2447)
Inserting LAD elements (Page 2579)
Selecting the data type of a LAD element (Page 2583)
Editing LAD elements (Page 2594)
Inserting operands into LAD instructions (Page 2601)
Switching display formats in the program status (Page 2980)
GET_ERR_ID: Get error ID locally (Page 3392)
Inserting additional inputs and outputs in LAD elements (Page 2598)

UBLKMOV: Move block uninterruptible

Description

You can use the "Move block uninterruptible" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use VARIANT to define the source and destination area.

The move operation cannot be interrupted by other operating system activities. As a result the alarm reaction time of the CPU can increase during the execution of the "Move block uninterruptible" instruction.

Note

The tags of the instruction can only be used in data blocks in which the "Optimized block access" attribute is not set. If the tag with the retentivity setting "Set in IDB" has been declared, you can also use the tags "with optimized access".

Memory areas

You can use the "Move block uninterruptible" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image of the inputs
- Process image of the outputs

General rules for moving

The source and destination area must not overlap during the execution of the "Move block uninterruptible" instruction. If the source area is less than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is less than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a source or destination area defined as a formal parameter is less than a destination or source area specified on the SRCBLK or DSTBLK parameter, no data is transferred.

If a block of data type BOOL is moved, the tag must be addressed absolutely and the specified length of the area must be divisible by 8, otherwise the instruction cannot be executed.

You can use the "Move block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Rules for moving character strings

You can use the "Move block uninterruptible" instruction to also move source and destination areas of the STRING data type. If only the source area is STRING data type, the characters will be moved that are actually contained in the character string. Information on the actual and maximum length are not written in the destination area. If the source and destination area are each STRING data type, the current length of the character string in the destination area is set to the number of actually moved characters. If areas of the data type STRING are moved, you have to specify "1" as area length.

Parameters

The following table shows the parameters of the "Move block uninterruptible" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRCBLK	Input	VARIANT	L, P	Specifies the memory area to be moved (source area).
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output at the RET_VAL parameter.
DSTBLK	Output	VARIANT	L, P	Specifies the memory area to which the block is to be moved (destination area).

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8152	The WSTRING, WCHAR and BOOL data types are not supported.
8091	The source area is in a data block that is not relevant for program execution.
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Evaluating errors with output parameter RET_VAL (Page 5370)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Switching display formats in the program status (Page 2980)

GET_ERR_ID: Get error ID locally (Page 3392)

Inserting additional inputs and outputs in LAD elements (Page 2598)

FILL: Fill block

Description

You can use the "Fill block" instruction to fill a memory area (destination area) with the content of another memory area (source area). The "Fill block" instruction moves the content of the source area to the destination area until the destination area is completely written. The move operation takes place in the direction of ascending addresses.

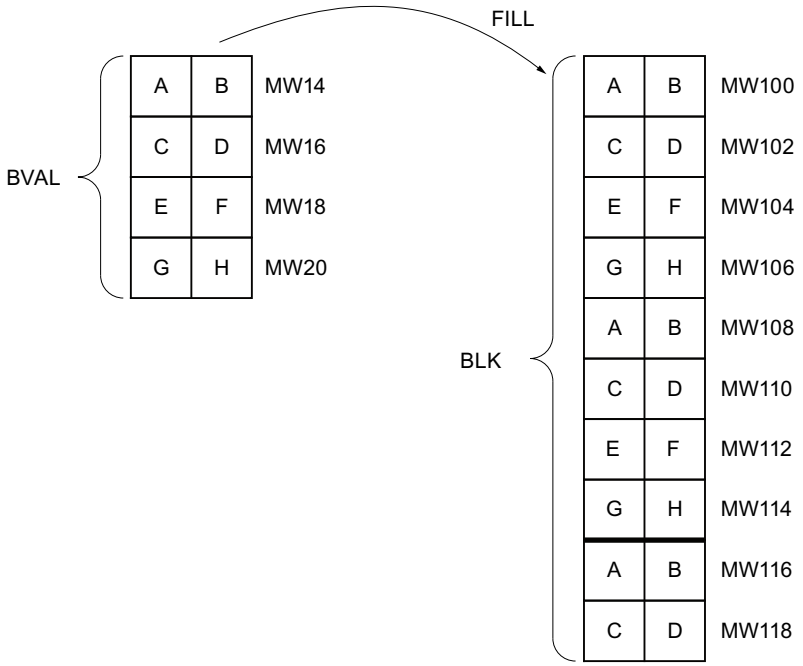
You use VARIANT to define the source and destination area.

Note

The tags of the instruction can only be used in data blocks in which the "Optimized block access" attribute is not set. If the tag with the retentivity setting "Set in IDB" has been declared, you can also use the tags "with optimized access".

For blocks with the "Optimized block access" attribute, you can use the instruction "FILL_BLK: Fill block" instruction.

The following figure shows the principle of the move operation:



Example: The contents of the range MW100 to MW118 are to be preassigned with the contents of the memory words MW14 to MW20.

Consistency of the source data and the target data

Please note that during execution of the instruction "Fill block" that the source data remain unchanged, otherwise the consistency of the target data is not ensured.

Memory areas

You can use the "Fill block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image of the inputs
- Process image of the outputs

General rules for moving

The source and destination area must not overlap. If the destination block to be preset is not an integer multiple of the length of the input parameter BVAL, the destination block is nevertheless written up to the last byte.

If the destination area to be preset is smaller than the source area, the function only copies as much data as can be written to the destination area.

If the destination or source area actually present is smaller than the assigned memory area for the source or destination block (BVAL BLK parameters), no data is transferred.

If the ANY pointer (source or destination) is of the data type BOOL, it must be addressed absolutely and the length specified must be divisible by 8; otherwise the instruction is not executed.

If the destination block is of data type STRING, the instruction describes the entire string including the administration information.

Parameters

The following table shows the parameters of the "Fill block" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
BVAL	Input	VARIANT	L, P	Specification of the memory area (source area) with the content used to fill the destination area at the BLK parameter.
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output at the RET_VAL parameter.
BLK	Output	VARIANT	L, P	Specification of the memory area that is filled with the content of the source area.

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8152	The WSTRING, WCHAR and BOOL data types are not supported.
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)
Evaluating errors with output parameter RET_VAL (Page 5370)
Basics of the EN/ENO mechanism (Page 2447)
Inserting LAD elements (Page 2579)
Selecting the data type of a LAD element (Page 2583)
Editing LAD elements (Page 2594)
Inserting operands into LAD instructions (Page 2601)
Switching display formats in the program status (Page 2980)
GET_ERR_ID: Get error ID locally (Page 3392)
Inserting additional inputs and outputs in LAD elements (Page 2598)

Conversion operations

CONVERT: Convert value

Description

The "Convert value" instruction reads the content of the IN parameter and converts it according to the data types selected in the instruction box. The converted value is sent to the output OUT.

For information on possible conversions, refer to the "Explicit conversion" section at "See also".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- Errors such as an overflow occur during execution.

Parameters

The following table shows the parameters of the "Convert value" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

Parameter	Declaration	Data type	Memory area	Description
IN	Input	Bit strings, integers, floating-point numbers, CHAR, BCD16, BCD32	I, Q, M, D, L, P or constant	Value to be converted.
OUT	Output	Bit strings, integers, floating-point numbers, CHAR, BCD16, BCD32	I, Q, M, D, L, P	Result of the conversion

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

Bit strings (BYTE, WORD, DWORD, LWORD) cannot be selected in the instruction box. If you specify an operand of data type BYTE, WORD, DWORD or LWORD at a parameter of the instruction, it is interpreted according to the data type of the input or output parameter. Data type BYTE is then interpreted as USINT, WORD as UINT, DWORD as UDINT, and LWORD as LINT, for example.

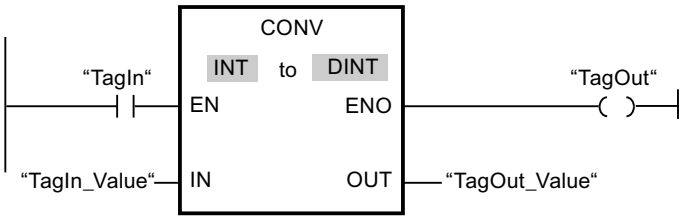
Note

For S7-1500 CPU: The data types DWORD and LWORD can be selected, if REAL or LREAL is selected as IN data type.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state "1", the content of operand "TagIn_Value" is read and converted to an integer (32-bit). The result is stored in operand "TagOut_Value". The output "TagOut" is set to "1" if the instruction was executed without errors.

See also

Overview of the valid data types (Page 2236)

Explicit conversion (Page 2391)

Explicit conversion of CHAR (Page 2415)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

ROUND: Round numerical value

Description

You can use the "Round numerical value" instruction to round the value at input IN to the nearest integer. The instruction interprets the value at input IN as a floating-point number and converts this to an integer of data type DINT. If the input value is exactly between an even and odd number, the even number is selected. The result of the instruction is sent to the OUT output and can be queried there.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The enable input EN has the signal state "0".
- Errors such as an overflow occur during execution.

Parameters

The following table shows the parameters of the "Round numerical value" instruction:

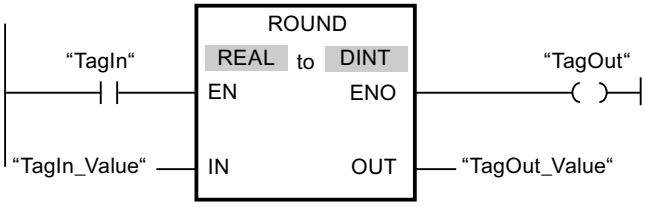
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value to be rounded.
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result of rounding

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
IN	TagIn_Value	1.50000000	-1.50000000
OUT	TagOut_Value	2	-2

If operand "TagIn" has the signal state "1", the "Round numerical value" instruction is executed. The floating-point number at input "TagIn_Value" is rounded to the nearest even integer and sent to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Editing LAD elements (Page 2594)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Inserting operands into LAD instructions (Page 2601)

CEIL: Generate next higher integer from floating-point number

Description

You can use the "Generate next higher integer from floating-point number" instruction to round the value at input IN to the next higher integer. The instruction interprets the value at the IN input as a floating-point number and converts this number to the next higher integer. The result of the instruction is sent to the OUT output and can be queried there. The output value can be greater than or equal to the input value.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The enable input EN has the signal state "0".
- Errors such as an overflow occur during execution.

Parameters

The following table shows the parameters of the "Generate next higher integer from floating-point number" instruction:

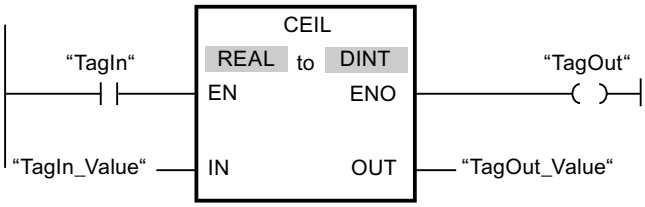
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result with next higher integer

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
IN	TagIn_Value	0.50000000	-0.50000000
OUT	TagOut_Value	1	0

If operand "TagIn" has the signal state "1", the "Generate next higher integer from floating-point number" instruction is executed. The floating-point number at the "TagIn_Value" input is rounded to the next higher integer and sent to the "TagOut_Value" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Editing LAD elements (Page 2594)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Inserting operands into LAD instructions (Page 2601)

FLOOR: Generate next lower integer from floating-point number

Description

You can use the "Generate next lower integer from floating-point number" instruction to round the value at input IN to the next lower integer. The instruction interprets the value at input IN as a floating-point number and converts this to the next lower integer. The result of the instruction is sent to the OUT output and can be queried there. The output value can be less than or equal to the input value.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The enable input EN has the signal state "0".
- Errors such as an overflow occur during execution.

Parameters

The following table shows the parameters of the "Generate next lower integer from floating-point number" instruction:

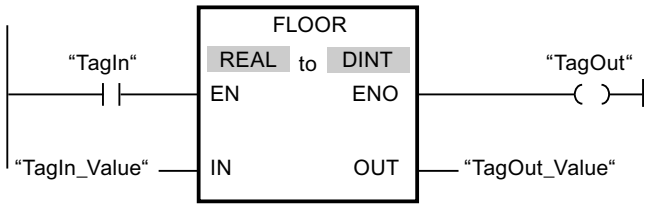
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result with next lower integer

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
IN	TagIn_Value	0.50000000	-0.50000000
OUT	TagOut_Value	0	-1

If operand "TagIn" has the signal state "1", the "Generate next lower integer from floating-point number" instruction is executed. The floating-point number at input "TagIn_Value" is rounded to the next lower integer and sent to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

TRUNC: Truncate numerical value

Description

You can use the "Truncate numerical value" instruction to form an integer from the value at the IN input. The value at the IN input is interpreted as a floating-point number. The instruction selects only the integer part of the floating-point number and sends this to the OUT output without decimal places.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN input has the signal state "0".
- Errors such as an overflow occur during execution.

Parameters

The following table shows the parameters of the "Truncate numerical value" instruction:

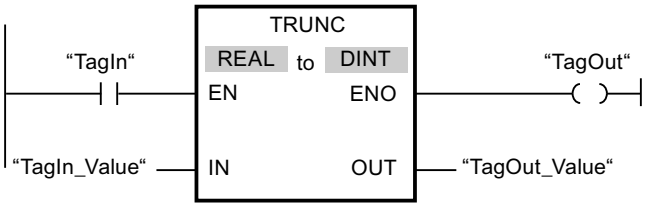
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L or constant	Input value
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L	Integer part of the input value

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
IN	TagIn_Value	1.50000000	-1.50000000
OUT	TagOut_Value	1	-1

If operand "TagIn" has the signal state "1", the "Truncate numerical value" instruction is executed. The integer part of the floating-point number at the "TagIn_Value" input is converted to an integer and sent to the "TagOut_Value" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

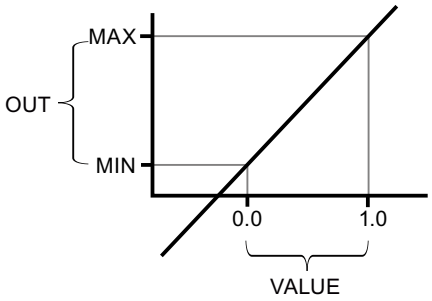
Selecting the data type of a LAD element (Page 2583)

SCALE_X: Scale

Description

You can use the "Scale" instruction to scale the value at the VALUE input by mapping it to a specified value range. When the "Scale" instruction is executed, the floating-point value at the VALUE input is scaled to the value range that was defined by the MIN and MAX parameters. The result of the scaling is an integer, which is stored in the OUT output.

The following figure shows an example of how values can be scaled:



The "Scale" instruction works with the following equation:

OUT = [VALUE * (MAX – MIN)] + MIN

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The value at the MIN input is greater than or equal to the value at the MAX input.
- The value of a specified floating-point number is outside the range of the normalized numbers according to IEEE-754.
- An overflow occurs.
- The value at the VALUE input is NaN (Not a Number = result of an invalid arithmetic operation).

Note

For more information on the converting of analog values, refer to the respective manual.

Parameters

The following table shows the parameters of the "Scale" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
MIN	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Low limit of the value range

Parameter	Declaration	Data type	Memory area	Description
VALUE	Input	Floating-point numbers	I, Q, M, D, L or constant	Value to be scaled. If you enter a constant, you must declare it.
MAX	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	High limit of the value range
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L	Result of scaling

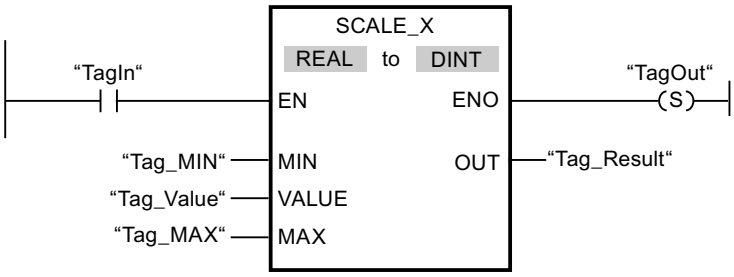
You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

For additional information on declaring constants, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
MIN	Tag_MIN	10
VALUE	Tag_Value	0.5
MAX	Tag_MAX	30
OUT	Tag_Result	20

If operand "TagIn" has the signal state "1", the "Scale" instruction is executed. The value at the "Tag_Value" input is scaled to the range of values defined by the values at the "Tag_MIN" and "Tag_MAX" inputs. The result is stored in the "Tag_Result" output. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

NORM_X: Normalize (Page 3365)

Declaring global constants (Page 2489)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

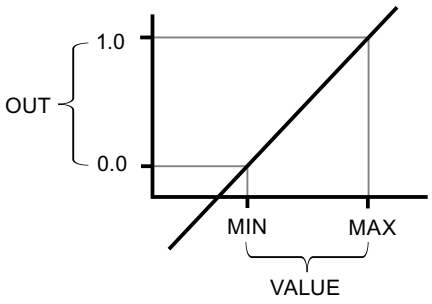
Selecting the data type of a LAD element (Page 2583)

NORM_X: Normalize

Description

You can use the instruction "Normalize" to normalize the value of the tag at the VALUE input by mapping it to a linear scale. You can use the MIN and MAX parameters to define the limits of a value range that is applied to the scale. The result at the OUT output is calculated and stored as a floating-point number depending on the location of the value to be normalized within this value range. If the value to be normalized is equal to the value at the MIN input, the OUT output has the value "0.0". If the value to be normalized is equal to the value at input MAX, output OUT returns the value "1.0".

The following figure shows an example of how values can be normalized:



The "Normalize" instruction works with the following equation:

OUT = (VALUE – MIN) / (MAX – MIN)

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The value at the MIN input is greater than or equal to the value at the MAX input.
- The value of a specified floating-point number is outside the range of the normalized numbers according to IEEE-754.
- The value at the VALUE input is NaN (result of an invalid arithmetic operation).

Note

For more information on the converting of analog values, refer to the respective manual.

Parameter

The following table shows the parameters of the "Normalize" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
MIN [1]	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Low limit of the value range
VALUE [1]	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Value to be normalized.
MAX [1]	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	High limit of the value range
OUT	Output	Floating-point numbers	I, Q, M, D, L	Result of the normalization
[1] If you use constants in these three parameters, you only need to declare one of them.				

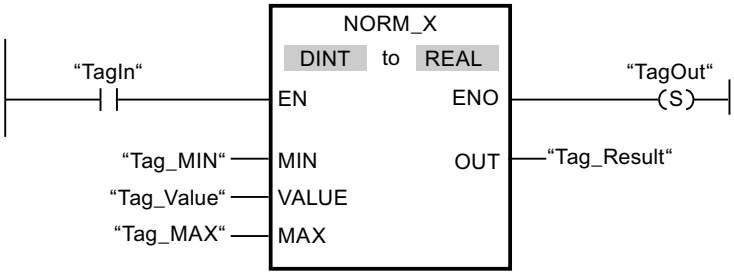
You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

For additional information on declaring constants, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
MIN	Tag_MIN	10
VALUE	Tag_Value	20

Parameter	Operand	Value
MAX	Tag_MAX	30
OUT	Tag_Result	0.5

If operand "TagIn" has the signal state "1", the "Normalize" instruction is executed. The value at the "Tag_Value" input is mapped to the range of values that were defined by the values at the "Tag_MIN" and "Tag_MAX" inputs. The tag value at the "Tag_Value" input is normalized to the defined value range. The result is stored as a floating-point number in the "Tag_Result" output. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

SCALE_X: Scale (Page 3362)

Declaring global constants (Page 2489)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

Other

SCALE: Scale

Description

Use the "Scale" instruction to convert the integer at the IN parameter into a floating-point number, which can be scaled in physical units between a low limit value and a high limit value. You can use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is scaled. The result of the instruction is output on the OUT parameter.

The "Scale" instruction works with the following equation:

OUT = [((FLOAT (IN) – K1)/(K2–K1)) * (HI_LIM–LO_LIM)] + LO_LIM

The values of the constants "K1" and "K2" are determined by the signal state on the BIPOLAR parameter. The following signal states are possible on the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case the "K1" constant has the value "-27648.0" and the "K2" constant the value "+27648.0".
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case the "K1" constant has the value "0,0" and the "K2" constant the value "+27648.0".

When the value at the IN parameter is greater than the value of the constant "K2", the result of the instruction is set to the value of the high limit (HI_LIM) and an error is output.

When the value at the IN parameter is less than the value of the constant "K1", the result of the instruction is set to the value of the low limit value (LO_LIM) and an error is output.

When the indicated low limit value is greater than the high limit value (LO_LIM > HI_LIM), the result is scaled in reverse proportion to the input value.

Parameter

The following table shows the parameters of the "Scale" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	INT	I, Q, M, D, L, P or constant	Input value to be scaled.
HI_LIM	Input	REAL	I, Q, M, D, L, P or constant	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P or constant	Low limit
BIPOLAR	Input	BOOL	I, Q, M, D, L or constant	Indicates if the value at IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
OUT	Output	REAL	I, Q, M, D, L, P	Result of the instruction
RET_VAL	Output	WORD	I, Q, M, D, L, P	Error information

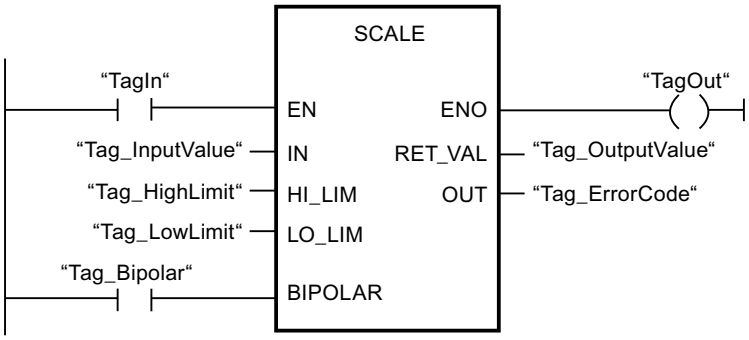
Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the constant "K2" or less than the value of the constant "K1"
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_InputValue	22
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
OUT	Tag_OutputValue	50.03978588
RET_VAL	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 2236)

Evaluating errors with output parameter RET_VAL (Page 5370)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Switching display formats in the program status (Page 2980)

GET_ERR_ID: Get error ID locally (Page 3392)

UNSCALE: Unscale

Description

The "Unscale" instruction is used to unscale the floating-point number on the IN parameter into physical units between a low limit and a high limit and convert it into an integer. You can use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range

to which the input value is unscaled. The result of the instruction is output on the OUT parameter.

The "Unscale" instruction works with the following equation:

OUT = [((IN–LO_LIM)/(HI_LIM–LO_LIM)) * (K2–K1)] + K1

The values of the constants "K1" and "K2" are determined by the signal state on the BIPOLAR parameter. The following signal states are possible on the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case the "K1" constant has the value "-27648.0" and the "K2" constant the value "+27648.0".
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case the "K1" constant has the value "0.0" and the "K2" constant the value "+27648.0".

When the value at the IN parameter is greater than the value of the constant "HI_LIM", the result of the instruction is set to the value of the constant (K2) and an error is output.

When the value at the IN parameter is less than the value of the constant of the low limit (LO_LIM), the result of the instruction is set to the value of the constant (K1) and an error is output.

Parameters

The following table shows the parameters of the "Unscale" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Input	BOOL	I, Q, M, D, L	Enable output
IN	Input	REAL	I, Q, M, D, L, P or constant	Input value to be unscaled to an integer value.
HI_LIM	Input	REAL	I, Q, M, D, L, P or constant	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P or constant	Low limit
BIPOLAR	Input	BOOL	I, Q, M, D, L or constant	Indicates if the value at IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
OUT	Output	INT	I, Q, M, D, L, P	Result of the instruction
RET_VAL	Output	WORD	I, Q, M, D, L, P	Error information

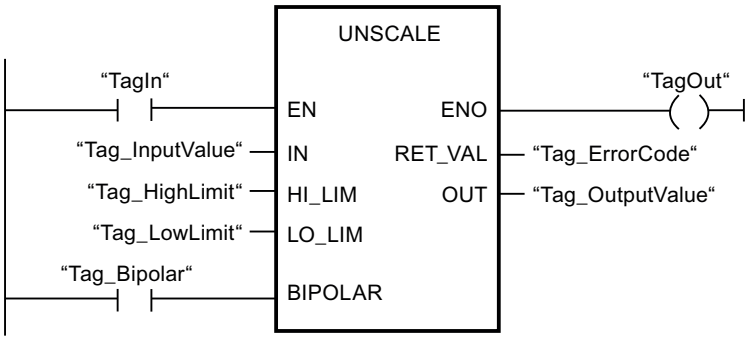
Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the high limit (HI_LIM) or less than the value of the low limit (LO_LIM).
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
OUT	Tag_OutputValue	22
RET_VAL	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 2236)

Evaluating errors with output parameter RET_VAL (Page 5370)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Switching display formats in the program status (Page 2980)

GET_ERR_ID: Get error ID locally (Page 3392)

Program control operations

---(JMP): Jump if RLO = 1

Description

You can use the "Jump if RLO = 1" instruction to interrupt the linear execution of the program and resume it in another network. The destination network must be identified by a jump label (LABEL). The name of this jump label is specified in the placeholder above the instruction.

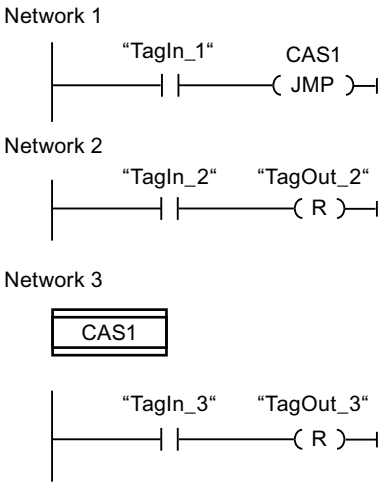
The specified jump label must be in the same block in which the instruction is executed. The name you specify can only occur once in the block. Only one jumping coil is permitted within a network.

If the result of logic operation (RLO) at the input of the instruction is "1", the jump to the network identified by the specified jump label is executed. The jump direction can be towards higher or lower network numbers.

If the condition at the input of the instruction is not fulfilled (RLO = 0), execution of the program continues in the next network.

Example

The following example shows how the instruction works:



If operand "TagIn_1" has the signal state "1", the "Jump if RLO = 1" instruction is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. If the "TagIn_3" input has the signal state "1", the "TagOut_3" output is set.

See also

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

---(JMPN): Jump if RLO = 0

Description

You can use the instruction "Jump if RLO = 0" to interrupt the linear execution of the program and resume it in another network, when the result of logic operation at the input of the instruction is "0". The destination network must be identified by a jump label (LABEL). The name of this jump label is specified in the placeholder above the instruction.

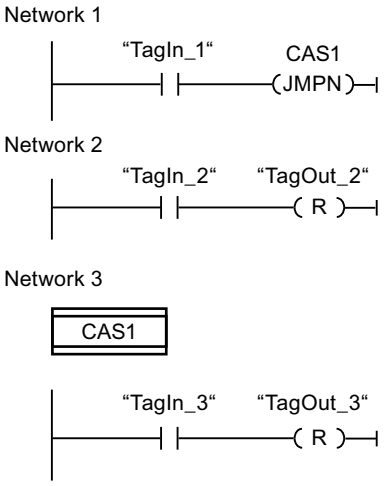
The specified jump label must be in the same block in which the instruction is executed. The name you specify can only occur once in the block. Only one jumping coil is permitted within a network.

If the result of logic operation (RLO) at the input of the instruction is "0", the jump to the network identified by the specified jump label is executed. The jump direction can be towards higher or lower network numbers.

If the result of logic operation at the input of the instruction is "1", execution of the program continues in the next network.

Example

The following example shows how the instruction works:



If operand "TagIn_1" has the signal state "0", the "Jump if RLO = 0" instruction is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. If the "TagIn_3" input has the signal state "1", the "TagOut_3" output is set.

See also

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Overview of the valid data types (Page 2236)

Inserting operands into LAD instructions (Page 2601)

LABEL: Jump label

Description

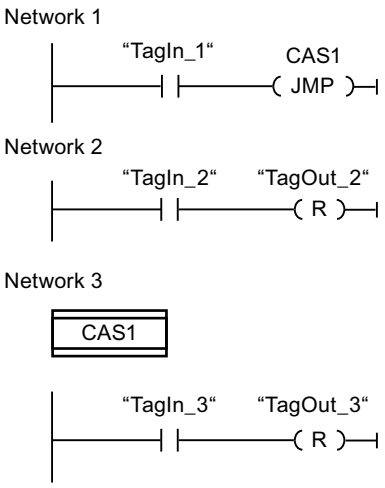
You can use a jump label to identify a destination network, in which the program execution should resume when a jump is executed.

The jump label and the instruction in which the jump label is specified must be located in the same block. The name of a jump label can only be assigned once in a block. You can declare up to 32 jump labels when you use a CPU S7-1200 and a maximum of 256 jump labels when you use a CPU S7-1500.

Only one jump label can be placed in a network. Each jump label can jump to several locations.

Example

The following example shows how the instruction works:



If operand "TagIn_1" has the signal state "1", the "Jump if RLO = 1" instruction is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. If the "TagIn_3" input has the signal state "1", the "TagOut_3" output is set.

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Inserting additional inputs and outputs in LAD elements (Page 2598)

Removing inputs and outputs (Page 2599)

Selecting the data type of a LAD element (Page 2583)

JMP_LIST: Define jump list

Description

You can use the "Define jump list" instruction to define several conditional jumps and continue the program execution in a specific network depending on the value of the K parameter.

You define the jumps with jump labels (LABEL), which you specify at the outputs of the instruction box. The number of outputs can be expanded in the instruction box. You can declare up to 32 outputs when you use a CPU S7-1200 and a maximum of 99 outputs when you use a CPU S7-1500.

The numbering of the outputs begins with the value "0" and continues in ascending order with each new output. Only jump labels can be specified at the outputs of the instruction. Instructions or operands cannot be specified.

The value of the K parameter specifies the number of the output and thus the jump label where the program execution is to be resumed. If the value in the K parameter is greater than the number of available outputs, the program execution is resumed in the next network of the block.

The "Define jump list" instruction is only executed if the signal state is "1" at the EN enable input.

Parameter

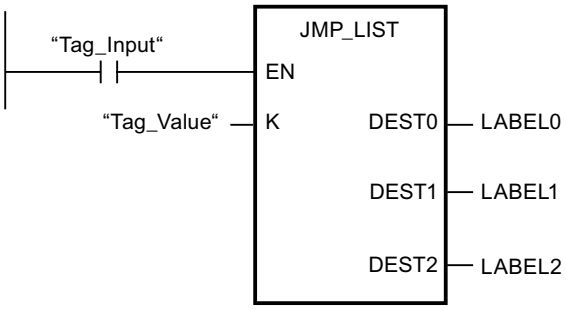
The following table shows the parameters of the "Define jump list" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
K	Input	UINT	I, Q, M, D, L or constant	Specifies the number of the output and thus the jump that is to be made.
DEST0	-	-	-	First jump label
DEST1	-	-	-	Second jump label
DESTn	-	-	-	Optional jump labels

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand/Jump label	Value
K	Tag_Value	1
Dest0	LABEL0	Jump to the network that is identified with the jump label "LABEL0".
Dest1	LABEL1	Jump to the network that is identified with the jump label "LABEL1".
Dest2	LABEL2	Jump to the network that is identified with the jump label "LABEL2".

If operand "Tag_Input" has the signal state "1", the "Define jump list" instruction is executed. The program execution is resumed according to the value of operand "Tag_Value" in the network that is identified with the jump label "LABEL1".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Inserting additional inputs and outputs in LAD elements (Page 2598)

Removing inputs and outputs (Page 2599)

Selecting the data type of a LAD element (Page 2583)

SWITCH: Jump distributor

Description

You can use the "Jump distributor" instruction to define multiple program jumps to be executed depending on the result of one or more comparison instructions.

You specify the value to be compared in the K parameter. This value is compared with the values that are provided by the various inputs. You can select the comparison method for each individual input. The availability of the various comparison instructions depends on the data type of the instruction.

The following table shows the comparison instructions that are available depending on the selected data type:

Data type		Instruction	Syntax
S7-1200	S7-1500		
Bit strings	Bit strings	Equal	==
		Not equal	<>
Integers, floating-point numbers, TIME, DATE, TOD	Integers, floating-point numbers, TIME, LTIME, DATE, TOD, LTOD, LDT	Equal	==
		Not equal	<>
		Greater or equal	>=
		Less or equal	<=
		Greater than	>
		Less than	<

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box. If you select a comparison instruction and the data type of the instruction is not yet defined, the "<???>" drop-down list only offers the data types that are permitted for the selected comparison instruction.

Execution of the instruction begins with the first comparison and runs until a comparison condition is met. If a comparison condition is met, the subsequent comparison conditions are not considered. If none of the specified comparison conditions are met, the jump at the ELSE output is executed. If no program jump is defined at the ELSE output, execution of the program continues in the next network.

The number of outputs can be expanded in the instruction box. The numbering of the outputs begins with the value "0" and continues in ascending order with each new output. Specify jump labels (LABEL) at the outputs of the instruction. Instructions or operands cannot be specified at the outputs of the instruction.

An input is automatically inserted for each additional output. The jump programmed at an output is executed if the comparison condition of the corresponding input is fulfilled.

Parameters

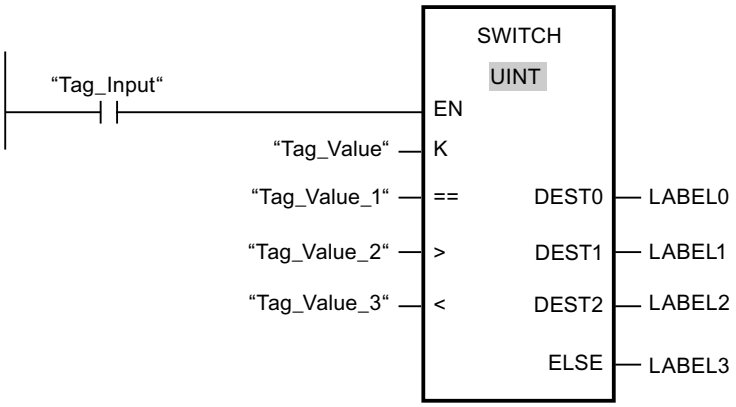
The following table shows the parameters of the "Jump distributor" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
K	Input	UINT	UINT	I, Q, M, D, L or constant	Specifies the value to be compared.
<Comparison values>	Input	Bit strings, integers, floating-point numbers, TIME, DATE, TOD	Bit strings, integers, floating-point numbers, TIME, LTIME, DATE, TOD, LTOD, LDT	I, Q, M, D, L or constant	Input value with which the value of the K parameter is compared.
DEST0	-	-	-	-	First jump label
DEST1	-	-	-	-	Second jump label
DEST(n)	-	-	-	-	Optional jump labels(n = 2 to 99)
ELSE	-	-	-	-	Program jump that is executed when none of the comparison conditions are fulfilled.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand/Jump label	Value
K	Tag_Value	23
==	Tag_Value_1	20
>	Tag_Value_2	21
<	Tag_Value_3	19
Dest 0	LABEL0	Jump to jump label "LABEL0", if the value of the K parameter equals 20.
Dest 1	LABEL1	Jump to jump label "LABEL1" if the value of the K parameter is greater than 21.
Dest 2	LABEL2	Jump to jump label "LABEL2", if the value of the K parameter is less than 19.
ELSE	LABEL 3	Jump to jump label "LABEL3", if the none of the comparison conditions are fulfilled.

If the operand "Tag_Input" changes to signal state "1", the instruction "Jump distributor" is executed. The execution of the program is continued in the network that is identified with the jump label "LABEL1".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Inserting additional inputs and outputs in LAD elements (Page 2598)

Removing inputs and outputs (Page 2599)

Selecting the data type of a LAD element (Page 2583)

--(RET): Return

Description

You can use the "Return" instruction to stop the execution of a block. The results is three types through which the block processing can be completed.

- Without call of the "Return" instruction
  The block is exited after execution of the last network. The ENO of the function call is set to the signal state "1".
- Call of the "Return" instruction with preceding logic operation (see example)
  If the left connector has the signal state "1", the block will be exited. The ENO of the function call corresponds to the operand.
- Call of the "Return" instruction without previous logic operation
  The block is exited. The ENO of the function call corresponds to the operand.

Note

Only one jumping coil may be used in a network ("Return", "Jump if RLO=1", "Jump if RLO=0").

If the result of logic operation (RLO) at the input of the "Return" instruction is "1", program execution is terminated in the currently called block and resumed after the call function in the calling block (for example, in the calling OB). The status (ENO) of the call function is determined by the parameter of the instruction. This can assume the following values:

- RLO
- TRUE/FALSE
- <Operand>

To set the parameter values, double-click the instruction and select the corresponding value in the drop-down list.

The following table shows the status of the call function if the "Return" instruction is programmed in a network within the called block:

RLO	Parameter value	ENO of the call function
1	RLO	1
	TRUE	1
	FALSE	0
	<Operand>	<Operand>
0	RLO	The program execution continues in the next network of the called block.
	TRUE	
	FALSE	
	<Operand>	

If an OB is completed, another block will be selected and started or executed by the priority class system:

- If the program cycle OB was completed, it will be restarted.
- If an OB is completed that has interrupted another block (for example, an alarm OB), then the interrupted block (for example, program cycle OB) will be executed.

Parameters

The following table shows the parameters of the "Return" instruction:

Parameter	Declaration	Data type	Memory area	Description
Status of the calling function with RLO = 1:				
RLO	-	-		Is set to the signal status of the RLO.
TRUE	-	-		1
FALSE	-	-		0
<Operand>	Input	BOOL	I, Q, M, D, L	Signal state of the specified operand

Example

The following example shows how the instruction works:

If operand "TagIn" has the signal state "1", the "Return" instruction is executed. Program execution is terminated in the called block and continues in the calling block. The ENO enable output of the call function is reset to signal state "0".

See also

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Overview of the valid data types (Page 2236)

Inserting operands into LAD instructions (Page 2601)

Runtime control

ENDIS_PW: Limit and enable password legitimation

Description

You use the "Limit and enable password legitimation" instruction to specify whether configured passwords may be legitimated or not for the CPU. You can prevent legitimated connections, even when the correct password is known.

When you call the command and the REQ parameter has the signal state "0", only the currently set state is displayed at the output parameters. If changes are made to the input parameters, these changes are not transferred to the output parameters.

If the instruction is called and REQ parameter has the signal state "1", the signal state is applied from the input parameters (F_PWD, FULL_PWD, R_PWD, HMI_PWD). FALSE means that legitimation per password is not permitted; TRUE means the password can be used.

Disable or enabling of the passwords can be allowed or prohibited individually. For example, all passwords can be prohibited, except the fail-safe password. You can thus limit access options to a small user group. The output parameters (F_PWD_ON, FULL_PWD_ON, R_PWD_ON, HMI_PWD_ON) always show the current status of the password use, regardless of the REQ parameter.

Non-configured passwords must have the TRUE signal state at the input and return the TRUE signal state at the output. The fail-safe password can only be configured for an F-CPU and, therefore, must always be interconnected with theTRUE signal state in a standard CPU. If the instruction return an error, the call remains without effect, i.e. the previous lock remains in force.

Disabled passwords can be enabled again under the following conditions:

- The CPU is reset to its factory settings.
- The front panel of the S7-1500 CPU supports a dialog that allows you to navigate to the appropriate menu in which the passwords can be enabled again.
- When you call the "Limit and enable password legitimation" instruction, the input parameter of the desired password has the signal state "1".
- Set the operating mode switch to STOP. The restriction to password legitimation is re-established as soon as the switch is set back to RUN.
- Inserting an empty memory card (transfer card or program card) into S7-1200 CPU.
- The transition from POWER OFF to POWER ON disables the protection in the S7-1200 CPU. The "Limit and enable password legitimation" instruction must be called again in the program (e.g. in the startup OB).

Note

The "Limit and enable password legitimation" instruction blocks access to the HMI system, if the HMI password has not been enabled.

Note

Existing legitimized connections retain their access rights and you cannot use the "Limit and enable password legitimation" instruction to restrict them.

Preventing accidental lockout out with an S7-1500 CPU

The same setting can be made on the front panel of the S7-1500 CPU and the CPU saves the most recent setting.

To prevent accidental lockout, the protection can be disabled by setting the mode selector to STOP with an S7-1500 CPU. The protection is automatically enabled again by setting mode selector to RUN without having to call the "Limit and enable password legitimation" instruction again or taking additional actions on the front panel.

Preventing accidental lockout with an S7-1200 CPU

The protection for POWER OFF - POWER ON is disabled because a S7-1200 CPU has no operating mode switch. As a result it is possible and advisable to use certain program sequences within your program to prevent an accidental lockout.

To do this, program a time control using either a cyclic interrupt OB or a timer in the main OB (OB 1). This provides you with the option to call the "Limit and enable password legitimation" instruction again without delay in the corresponding OB (e.g. OB 1 or OB 35) after a transition from POWER OFF - POWER ON. Call the instruction in the startup OB (OB 100) in order to keep as short as possible the time window in which the application is not active and thus ensure that no restrictions exist in the password legitimation. This procedure offers you the largest possible protection against unauthorized access.

If an unintended lockout occurs you can skip the call in the startup OB (for example, by querying an input parameter) and have the set time (for example 10 seconds to 1 minute), to establish a connection to the CPU before the lock becomes active again.

If you have set no timer in your program code and are locked out, insert an empty transfer card or empty program card into the CPU. The empty transfer card or program card deletes the internal load memory of the CPU. You must then load the user program again from STEP 7 in the CPU.

Procedure for lost passwords with an S7-1200 CPU

If you have lost the password for a password-protected S7-1200 CPU, delete the password-protected program with an empty transfer card or program card. The empty transfer card or

program card deletes the internal load memory of the CPU. You can then load a new user program from STEP 7 Basic in the CPU.

WARNING

Inserting empty transfer card

When you insert a transfer card in a CPU during runtime, the CPU goes to STOP mode. If operating states are unstable, controllers may fail and thus cause the uncontrolled operation of the controlled devices. This leads to an unforeseen operation of the automation system which can cause deadly or serious injuries and/or damage to property.

After you remove the transfer card the content of the transfer card is in the internal load memory. Make sure at this point that the card contains no program.

WARNING

Inserting empty program card

When you insert a program card in a CPU during runtime, the CPU goes to STOP mode. If operating states are unstable, controllers may fail and thus cause the uncontrolled operation of the controlled devices. This leads to an unforeseen operation of the automation system which can cause deadly or serious injuries and/or damage to property.

Make sure that the program card is empty. The internal load memory is copied to the empty program card. The internal load memory is empty after the removal of the formerly empty program card.

You must remove the transfer card or program card before you put the CPU into RUN mode.

Effects of password use on the operating modes

The following table shows what effects the password use by means of the "Limit and enable password legitimation" instruction has on the operating modes and the corresponding user actions.

Action	Password protection by means of the instruction
Basic state after • Operating mode switch to STOP • Reset memory manually (PG, switch, change of MC (Motion Control)) • Reset to factory setting	Not activated (no restrictions)
Basic state after POWER ON	• S7-1200-CPU: the lock is disabled and the instruction must be called again in the program (in the startup OB, for example). • S7-1500 CPU: Enabled (if a lock was activated before POWER OFF). The option of not allowing passwords is retentive.
Operating mode transition RUN/STARTUP/HOLD -> STOP (by terminating the instruction, an error or communication) or STOP -> STARTUP/RUN/HOLD	Activated Passwords still cannot be used.

Parameters

The following table shows the parameters of the "Limit and enable password legitimation" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
REQ	Input	BOOL	I, Q, M, D, L or constant	When the REQ parameter has the signal state "0", the currently set signal state of the passwords is queried.
F_PWD	Input	BOOL	I, Q, M, D, L or constant	Read/write access including fail-safe • F_PWD = "0": Do not allow password • F_PWD = "1": Allow password
FULL_PWD	Input	BOOL	I, Q, M, D, L or constant	Read/write access • FULL_PWD = "0": Do not allow password • FULL_PWD = "1": Allow password
R_PWD	Input	BOOL	I, Q, M, D, L or constant	Read access • R_PWD = "0": Do not allow password • R_PWD = "1": Allow password
HMI_PWD	Input	BOOL	I, Q, M, D, L or constant	HMI access • HMI_PWD = "0": Do not allow password • HMI_PWD = "1": Allow password

Parameter	Declaration	Data type	Memory area	Description
F_PWD_ON	Output	BOOL	I, Q, M, D, L	Read/write access including fail-safe • F_PWD_ON = "0": Password not allowed • F_PWD_ON = "1": Password allowed
FULL_PWD_ON	Output	BOOL	I, Q, M, D, L	Read/write access status • FULL_PWD_ON = "0": Password not allowed • FULL_PWD_ON = "1": Password allowed
R_PWD_ON	Output	BOOL	I, Q, M, D, L	Read-access status • R_PWD_ON = "0": Password not allowed • R_PWD_ON = "1": Password allowed
HMI_PWD_ON	Output	BOOL	I, Q, M, D, L	HMI-access status • HMI_PWD_ON = "0": Password not allowed • HMI_PWD_ON = "1": Password allowed
RET_VAL	Output	WORD	I, Q, M, D, L	Error information

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8090	The "Limit and enable password legitimation" instruction is not supported
80D0	The password for fail-safe is not configured. With standard CPUs, the signal state must be TRUE.
80D1	The read/write access is not configured
80D2	The read access is not configured
80D3	The HMI access is not configured
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Overview of the valid data types (Page 2236)

Evaluating errors with output parameter RET_VAL (Page 5370)

Basics of the EN/ENO mechanism (Page 2447)

Selecting the data type of a LAD element (Page 2583)

Inserting operands into LAD instructions (Page 2601)

Switching display formats in the program status (Page 2980)

RE_TRIGR: Restart cycle monitoring time

Description

You can use the "Restart cycle monitoring time" instruction to restart the cycle time monitoring of the CPU. The maximum cycle time then starts over again for the duration you have set in the CPU configuration.

The instruction "Restart cycle monitoring time" can be called regardless of the priority in all blocks.

If the instruction is called in a block with a higher priority, such as a hardware interrupt, diagnostic error interrupt, or cyclic interrupt, the instruction is not executed and the ENO enable output is set to signal state "0".

The instruction "Restart cycle monitoring time" can be called a maximum of 10 times in a program cycle.

Parameter

The "Restart cycle monitoring time" instruction has no parameters.

See also

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Overview of the valid data types (Page 2236)

Evaluating errors with output parameter RET_VAL (Page 5370)

Basics of the EN/ENO mechanism (Page 2447)

Selecting the data type of a LAD element (Page 2583)

Inserting operands into LAD instructions (Page 2601)

STP: Exit program

Description

You use the "Exit program" instruction to set the CPU to STOP mode and therefore to terminate the execution of the program. The effects of changing from RUN to STOP depend on the CPU configuration.

When the result of logic operation (RLO) at the input of the instruction is "1", the CPU changes to STOP mode and program execution is terminated. The signal state at the output of the instruction is not evaluated.

When the RLO is "0" at the input of the instruction, then the instruction will not be executed.

Parameters

The "Exit program" instruction has no parameters.

See also

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Overview of the valid data types (Page 2236)

Inserting operands into LAD instructions (Page 2601)

GET_ERROR: Get error locally

Description

The "Get error locally" instruction is used to query the occurrence of errors within a block. This is usually to access error. If the system reports an error during the processing of the block, detailed information is stored in the operand at the ERROR output for the first error to occur during the execution of the block since the last execution of the instruction.

Only operands of the "ErrorStruct" system data type can be specified at the ERROR output. The "ErrorStruct" system data type specifies the exact structure in which the information about the error is stored. Using additional instructions, you can evaluate this structure and program an appropriate response. If several errors occur in the block, the error information about the next error to occur in the instruction is output only after the first error that occurred has been remedied.

Note

The ERROR output is only changed if error information is present. To set the output back to "0" after handling an error, you have the following options:

- Declare the tag in the "Temp" section of the block interface.
- Reset the tag to "0" before calling the instruction.
- Query the enable output ENO.

The enable output ENO of the instruction "Get error locally" instruction is set only if the enable input EN returns signal state "1" and error information is present. If one of these conditions is not fulfilled, the remaining program execution is not affected by the "Get error locally" instruction.

You can find an example of how you can implement the instruction in combination with other troubleshooting options under "See also".

Note

The "Get error locally" instruction enables local error handling within a block. If "Get error locally" is inserted in the program code of a block, any predefined system reactions are ignored if errors occur.

Parameters

The following table lists the parameters of the "Get error locally" instruction:

Parameter	Declaration	Data type	Memory area	Description
ERROR	Output	ErrorStruct	D, L	Error information

Data type "ErrorStruct"

The following table lists the structure of the "ErrorStruct" data type:

Structure component		Data type	Description
ERROR_ID		WORD	Error ID
FLAGS		BYTE	Shows if an error occurred during a block call. 16#01: Error during a block call 16#00: No error during a block call
REACTION		BYTE	Default reaction: 0: Ignore (write error) 1: Continue with substitute value "0" (read error) 2: Skip instruction (system error)
CODE_ADDRESS		CREF	Information about the address and type of block
	BLOCK_TYPE	BYTE	Type of block where the error occurred: 1: OB 2: FC 3: FB
	CB_NUMBER	UINT	Number of the code block
	OFFSET	UDINT	Reference to the internal memory
MODE		BYTE	Information about the address of an operand
OPERAND_NUMBER		UINT	Operand number of the machine command
POINTER_NUMBER_LOCATION		UINT	(A) Internal pointer
SLOT_NUMBER_SCOPE		UINT	(B) Storage area in internal memory
DATA_ADDRESS		NREF	Information about the address of an operand

Structure component		Data type	Description
	AREA	BYTE	(C) Memory area: L: 16#40 – 4E, 86, 87, 8E, 8F, C0 – CE I: 16#81 Q: 16#82 M: 16#83 DB: 16#84, 85, 8A, 8B Range violations for a directly editable tag of data type DINT: 16#04
	DB_NUMBER	UINT	(D) Number of the data block
	OFFSET	UDINT	(E) Relative address of the operand

Structure component "ERROR_ID"

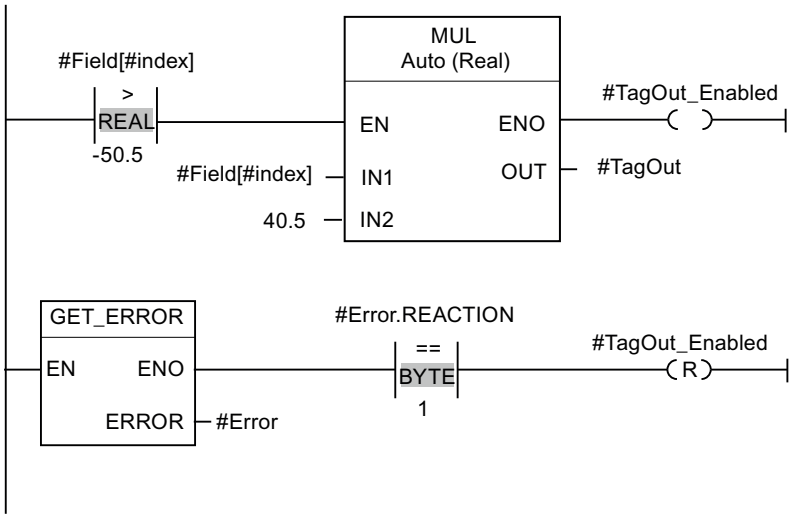
The following table lists the values that can be output at the structure component "ERROR_ID":

ID* (hexadecimal)	ID* (decimal)	Description
0	0	No error
2503	9475	Invalid pointer
2520	9504	Invalid STRING
2522	9506	Read error: Operand outside the valid range
2523	9507	Write error: Operand outside the valid range
2524	9508	Read error: Invalid operand
2525	9509	Write error: Invalid operand
2528	9512	Read error: Data alignment
2529	9513	Write error: Data alignment
252C	9516	Invalid pointer
2530	9520	Write error: Data block
2533	9523	Invalid pointer used
2538	9528	Access error: DB does not exist
2539	9529	Access error: Wrong DB used
253A	9530	Global data block does not exist
253C	9532	Faulty information or the function does not exist
253D	9533	System function does not exist
253E	9534	Faulty information or the function block does not exist
253F	9535	System block does not exist
2550	9552	Access error: DB does not exist
2551	9553	Access error: Wrong DB used
2575	9589	Error in the program nesting depth
2577	9591	The block property "Parameter passing via registers" is not selected.
2576	9590	Error in the local data distribution

ID* (hexadecimal)	ID* (decimal)	Description
25A0	9632	Internal error in TP
25A1	9633	Tag is write-protected
25A2	9634	Invalid numerical value of tag
2942	10562	Read error: Input
2943	10563	Write error: Output
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".		

Example

The following example shows how the instruction works:



An error occurred accessing the "#Field[#index]" tag. The enable output ENO of the "Multiply" instruction and the "#TagOut_Enabled" operand have signal state "1" despite the read/access error and the multiplication is performed with the value "0.0". If this error scenario occurs, it is recommended to program the "Get error locally" instruction after the "Multiply" instruction to get the error. The error information supplied by the "Get error locally" instruction is evaluated with the comparison instruction "Equal". If the "#Error.REACTION" structure component has the value "1", this involves a read/access error and the "#TagOut_Enabled" output is reset.

See also

Overview of the valid data types (Page 2236)

Inserting operands into LAD instructions (Page 2601)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Switching display formats in the program status (Page 2980)

Handling program execution errors (Page 2821)

Querying and fixing errors in the program code (Page 258)

GET_ERR_ID: Get error ID locally

Description

The "Get error ID locally" instruction is used to query the occurrence of errors within a block. This is usually to access error. If the system reports during the block processing errors within the block execution since the last execution of the instruction, the error ID of the first error that occurred in the tag is stored at the ID output.

Only operands of the WORD data type can be specified at the ID output. If several errors occur in the block, the error ID of the next error to occur in the instruction is output only after the first error that occurred has been remedied.

Note

The ID output is only changed if error information is present. To set the output back to "0" after handling an error, you have the following options:

- Declare the tag in the "Temp" section of the block interface.
- Reset the tag to "0" before calling the instruction.
- Query the enable output ENO.

The enable output ENO of the instruction "Get error ID locally" instruction is set only if the enable input EN returns signal state "1" and error information is present. If one of these conditions is not fulfilled, the remaining program execution is not affected by the "Get error ID locally" instruction.

You can find an example of how you can implement the instruction in combination with other troubleshooting options under "See also".

Note

The "Get error ID locally" instruction enables local error handling within a block. If the "Get error ID locally" instruction is inserted in the program code of a block, any predefined system reactions are ignored if errors occur.

Parameters

The following table lists the parameters of the "Get error ID locally" instruction:

Parameter	Declaration	Data type	Memory area	Description
ID	Output	WORD	I, Q, M, D, L	Error ID

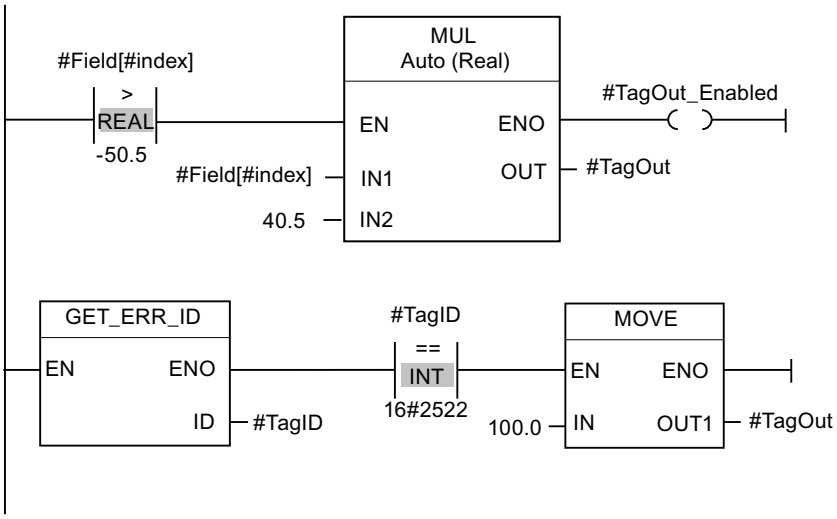
ID parameter

The following table lists the values that can be output at the ID parameter:

ID* (hexadecimal)	ID* (decimal)	Description
0	0	No error
2503	9475	Invalid pointer
2520	9504	Invalid STRING
2522	9506	Read error: Operand outside the valid range
2523	9507	Write error: Operand outside the valid range
2524	9508	Read error: Invalid operand
2525	9509	Write error: Invalid operand
2528	9512	Read error: Data alignment
2529	9513	Write error: Data alignment
252C	9516	Invalid pointer
2530	9520	Write error: Data block
2533	9523	Invalid pointer used
2538	9528	Access error: DB does not exist
2539	9529	Access error: Wrong DB used
253A	9530	Global data block does not exist
253C	9532	Faulty information or the function does not exist
253D	9533	System function does not exist
253E	9534	Faulty information or the function block does not exist
253F	9535	System block does not exist
2550	9552	Access error: DB does not exist
2551	9553	Access error: Wrong DB used
2575	9589	Error in the program nesting depth
2576	9590	Error in the local data distribution
2577	9591	The block property "Parameter passing via registers" is not selected.
25A0	9632	Internal error in TP
25A1	9633	Tag is write-protected
25A2	9634	Invalid numerical value of tag
2942	10562	Read error: Input
2943	10563	Write error: Output
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".		

Example

The following example shows how the instruction works:



An error occurred accessing the "#Field[#index]" tag. The enable output ENO of the "Multiply" instruction and the "#TagOut_Enabled" operand have signal state "1" despite the read/access error and the multiplication is performed with the value "0.0". If this error scenario occurs, it is recommended to program the "Get error ID locally" instruction after the "Multiply" instruction to get the error. The error information supplied by the "Get error ID locally" instruction is evaluated with the comparison instruction "Equal". If the operand "#TagID" returns the ID 2522, this involves a read/access error and the value "100.0" is written to the "#TagOut" output.

See also

Overview of the valid data types (Page 2236)

Inserting operands into LAD instructions (Page 2601)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Switching display formats in the program status (Page 2980)

Handling program execution errors (Page 2821)

Querying and fixing errors in the program code (Page 258)

INIT_RD: Initialize all retain data

Description

You can use the "Initialize all retain data" instruction to reset the retain data of all data blocks, bit memories and SIMATIC timers and counters at the same time. The instruction can only be executed within a startup OB because the execution would exceed the program cycle duration.

Parameter

The following table shows the parameters of the "Initialize all retain data" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
REQ	Input	BOOL	I, Q, M, D, L, T, C or constant	If the input "REQ" has the signal state "1", all retain data are reset.
RET_VAL	Output	INT	I, Q, M, D, L	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.

For additional information on valid data types, refer to "See also".

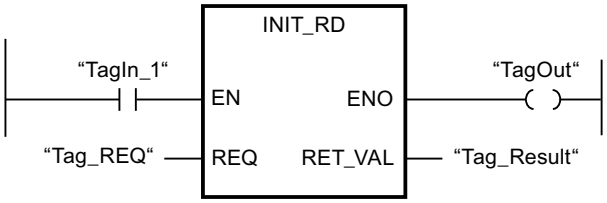
Parameters RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B5	The instruction cannot be executed because it was not programmed within a startup OB.
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:



If the operands "TagIn_1" and "Tag_REQ" have signal state "1", the instruction is executed. The retain data of all data blocks, bit memories and SIMATIC timers and counters are reset If the instruction is executed without errors, the ENO enable output has the signal state "1".

See also

Inserting LAD elements (Page 2579)
Editing LAD elements (Page 2594)
Overview of the valid data types (Page 2236)
Evaluating errors with output parameter RET_VAL (Page 5370)
Basics of the EN/ENO mechanism (Page 2447)
Selecting the data type of a LAD element (Page 2583)
Inserting operands into LAD instructions (Page 2601)
Switching display formats in the program status (Page 2980)
GET_ERR_ID: Get error ID locally (Page 3392)

WAIT: Configure time delay

Description

The "Configure time delay" instruction is used to halt the execution of the program for a set period of time. You indicate the period of time in microseconds on the WT parameter of the instruction.

You can configure time delays of up to 32767 microseconds (µs). The smallest possible delay time depends on the respective CPU and corresponds to the execution time of the "Configure time delay" instruction.

The execution of the instruction can be interrupted by higher priority events.

The "Configure time delay" instruction supplies no error information.

Parameters

The following table shows the parameters of the "Configure time delay" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
WT	Input	INT	I, Q, M, D, L, P or constant	Time delay in microseconds (µs)

See also

Inserting LAD elements (Page 2579)
Editing LAD elements (Page 2594)
Overview of the valid data types (Page 2236)
Basics of the EN/ENO mechanism (Page 2447)

Selecting the data type of a LAD element (Page 2583)

Inserting operands into LAD instructions (Page 2601)

RUNTIME: Measure program runtime

Description

The "Measure program runtime" instruction is used to measure the runtime of the entire program, individual blocks or command sequences.

If you want to measure the runtime of your entire program, call the instruction "Measure program runtime" in OB1. Measurement of the runtime is started with the first call and the output RET_VAL returns the runtime of the program after the second call. The measured runtime includes all CPU processes that can occur during the program execution, for example, interruptions caused by higher-level events or communication. The instruction "Measure program runtime" reads an internal counter of the CPU and write the value to the IN-OUT parameter MEM. The instruction calculates the current program runtime according to the internal counter frequency and writes it to output RET_VAL.

If you want to measure the runtime of individual blocks or individual command sequences, you need three separate networks. Call the instruction "Measure program runtime" in an individual network within your program. You set the starting point of the runtime measurement with this first call of the instruction. Then you call the required program block or the command sequence in the next network. In another network, call the "Measure program runtime" instruction a second time and assign the same memory to the IN-OUT parameter MEM as you did during the first call of the instruction. The "Measure program runtime" instruction in the third network reads an internal CPU counter and calculates the current runtime of the program block or the command sequence according to the internal counter frequency and writes it to the output RET_VAL.

The Measure program runtime" instruction uses an internal high-frequency counter to calculate the time. If the counter overruns, the instruction returns values <= 0.0. Ignore these runtime values.

Note

The runtime of a command sequence cannot be determined exactly, because the sequence of instructions within a command sequence is changed during optimized compilation of the program.

Parameter

The following table lists the parameters of the "Measure program runtime" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

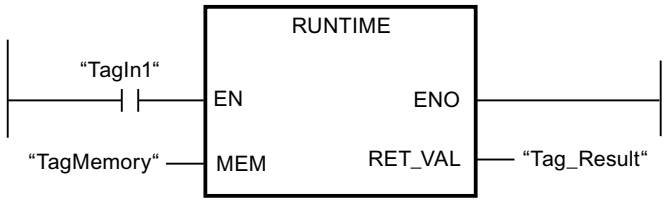
Parameter	Declaration	Data type	Memory area	Description
MEM	InOut	LREAL	M, D, L	Saves the starting point of the runtime measurement.
RET_VAL	Output	LREAL	I, Q, M, D, L	Returns the measured runtime in seconds

For additional information on valid data types, refer to "See also".

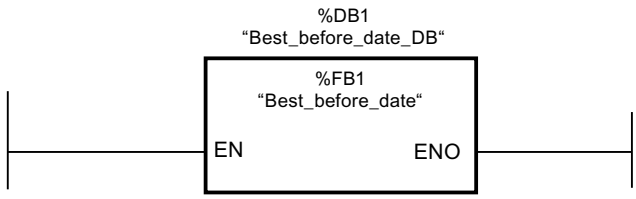
Example

The following example shows how the instruction works based on the runtime calculation of a program block:

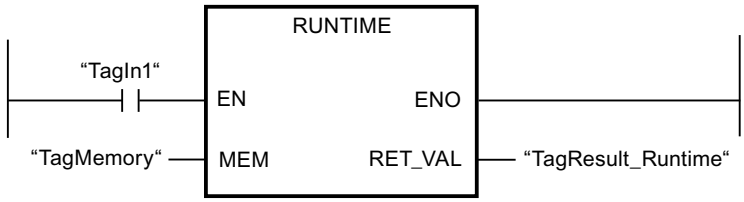
Network 1:



Network 2:



Network 3:



When the "TagIn1" operand in network 1 has the signal state "1", the instruction is executed. The starting point for the runtime measurement is set with the first call of the instruction and buffered as reference for the second call of the instruction in the "TagMemory" operand.

The "Best_before_date" program block FB1 is called in network 2.

When the FB1 program block has been executed and the "TagIn1" operand has the signal state "1", the instruction in network 3 is executed. The second call of the instruction calculates the runtime of the program block and writes the result to the output RET_VAL.

See also

Inserting LAD elements (Page 2579)
Editing LAD elements (Page 2594)
Overview of the valid data types (Page 2236)
Evaluating errors with output parameter RET_VAL (Page 5370)
Basics of the EN/ENO mechanism (Page 2447)
Selecting the data type of a LAD element (Page 2583)
Inserting operands into LAD instructions (Page 2601)
Switching display formats in the program status (Page 2980)

Word logic operations

AND: AND logic operation

Description

You can use the "AND logic operation" instruction to combine the value at the IN1 input and the value at the IN2 input bit-by-bit by AND logic and query the result at the OUT output.

When the instruction is executed, bit 0 of the value at the IN1 input and bit 0 of the value at the IN2 input are logically ANDed. The result is stored in bit 0 of output OUT. The same logic operation is executed for all other bits of the specified values.

The number of inputs can be expanded in the instruction box. The added inputs are numbered in ascending order in the box. When the instruction is executed, the values of all available input parameters are combined with AND logic (ANDed). The result is stored in the OUT output.

The result bit has signal state "1" only when both of the bits in the logic operation also have signal state "1". If one of the two bits of the logic operation has signal state "0", the corresponding result bit is reset.

Parameters

The following table shows the parameters of the "AND logic operation" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Bit strings	I, Q, M, D, L, P or constant	First value for logic operation
IN2	Input	Bit strings	I, Q, M, D, L, P or constant	Second value for logic operation

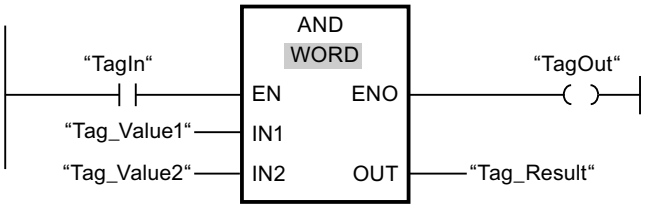
Parameter	Declaration	Data type	Memory area	Description
INn	Input	Bit strings	I, Q, M, D, L, P or constant	Other inputs whose values are logically combined.
OUT	Output	Bit strings	I, Q, M, D, L, P	Result of the instruction

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	Tag_Value1	0101 0101 0101 0101
IN2	Tag_Value2	0000 0000 0000 1111
OUT	Tag_Result	0000 0000 0000 0101

If operand "TagIn" has the signal state "1", the "AND logic operation" instruction is executed. The value of operand "Tag_Value1" and the value of operand "Tag_Value2" are ANDed. The result is mapped bit-for-bit and output in operand "Tag_Result". The ENO enable output and the "TagOut" output are set to signal state "1".

See also

Overview of the valid data types (Page 2236)

Inserting additional inputs and outputs in LAD elements (Page 2598)

Removing inputs and outputs (Page 2599)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

OR: OR logic operation

Description

You can use the "OR logic operation" instruction to combine the value at the IN1 input and the value at the IN2 input bit-by-bit by OR logic and query the result at the OUT output.

When the instruction is executed, bit 0 of the value at the IN1 input and bit 0 of the value at the IN2 input are combined by OR logic operation. The result is stored in bit 0 of output OUT. The same logic operation is executed for all bits of the specified tags.

The number of inputs can be expanded in the instruction box. The added inputs are numbered in ascending order in the box. When the instruction is executed, the values of all available input parameters are combined with OR logic (ORed). The result is stored in the OUT output.

The result bit has signal state "1" when at least one of the two bits in the logic operation has the signal state "1". If both of the bits of the logic operation have signal state "0", the corresponding result bit is reset.

Parameters

The following table shows the parameters of the "OR logic operation" instruction:

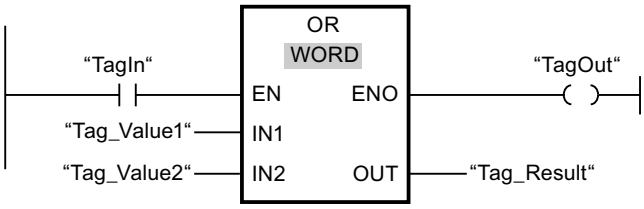
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Bit strings	I, Q, M, D, L, P or constant	First value for logic operation
IN2	Input	Bit strings	I, Q, M, D, L, P or constant	Second value for logic operation
INn	Input	Bit strings	I, Q, M, D, L, P or constant	Other inputs whose values are logically combined.
OUT	Output	Bit strings	I, Q, M, D, L, P	Result of the instruction

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	Tag_Value1	0101 0101 0101 0101
IN2	Tag_Value2	0000 0000 0000 1111
OUT	Tag_Result	0101 0101 0101 1111

If operand "TagIn" has the signal state "1", the "OR logic operation" instruction is executed. The value of operand "Tag_Value1" and the value of operand "Tag_Value2" are ORed. The result is mapped bit-for-bit and output in operand "Tag_Result". The ENO enable output and the "TagOut" output are set to signal state "1".

See also

Overview of the valid data types (Page 2236)

Inserting additional inputs and outputs in LAD elements (Page 2598)

Removing inputs and outputs (Page 2599)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

XOR: EXCLUSIVE OR logic operation

Description

You can use the "EXCLUSIVE OR logic operation" instruction to combine the value at the IN1 input and the value at the IN2 input bit-by-bit by EXCLUSIVE OR logic and query the result at the OUT output.

When the instruction is executed, bit 0 of the value at the IN1 input and bit 0 of the value at the IN2 input are combined by EXCLUSIVE OR logic operation. The result is stored in bit 0 of output OUT. The same logic operation is executed for all other bits of the specified value.

The number of inputs can be expanded in the instruction box. The added inputs are numbered in ascending order in the box. When the instruction is executed, the values of all available input parameters are combined with EXCLUSIVE OR logic. The result is stored in the OUT output.

The result bit has signal state "1" when one of the two bits in the logic operation has the signal state "1". If both of the bits of the logic operation have signal state "1" or "0", the corresponding result bit is reset.

Parameters

The following table shows the parameters of the "EXCLUSIVE OR logic operation" instruction:

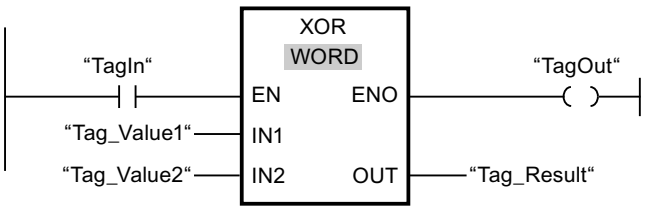
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Bit strings	I, Q, M, D, L, P or constant	First value for logic operation
IN2	Input	Bit strings	I, Q, M, D, L, P or constant	Second value for logic operation
INn	Input	Bit strings	I, Q, M, D, L, P or constant	Other inputs whose values are logically combined.
OUT	Output	Bit strings	I, Q, M, D, L, P	Result of the instruction

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	Tag_Value1	0101 0101 0101 0101
IN2	Tag_Value2	0000 0000 0000 1111
OUT	Tag_Result	0101 0101 0101 1010

If operand "TagIn" has the signal state "1", the "EXCLUSIVE OR logic operation" instruction is executed. The value of operand "Tag_Value1" and the value of operand "Tag_Value2" are combined by EXCLUSIVE OR logic. The result is mapped bit-for-bit and output in operand "Tag_Result". The ENO enable output and the "TagOut" output are set to signal state "1".

See also

Overview of the valid data types (Page 2236)

Inserting additional inputs and outputs in LAD elements (Page 2598)

Removing inputs and outputs (Page 2599)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

INV: Create ones complement

Description

You can use the "Create ones complement" instruction to invert the signal state of the bits at the IN input. When the instruction is processed, the value at the IN input and a hexadecimal template (W#16#FFFF for 16-bit numbers or DW#16#FFFF FFFF for 32-bit numbers) are combined by EXCLUSIVE OR logic. This inverts the signal state of the individual bits that are then stored at output OUT.

Parameters

The following table shows the parameters of the "Create ones complement" instruction:

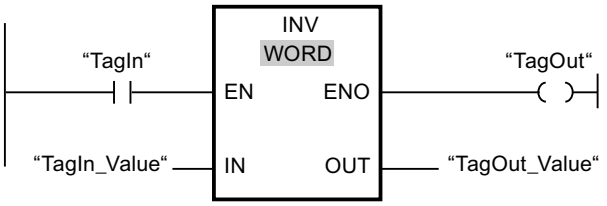
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers	I, Q, M, D, L, P or constant	Input value
OUT	Output	Bit strings, integers	I, Q, M, D, L, P	Ones complement of the value at input IN

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
IN	TagIn_Value	W#16#000F	W#16#7E
OUT	TagOut_Value	W#16#FFF0	W#16#81

If operand "TagIn" has the signal state "1", the "Create ones complement" instruction is executed. The instruction inverts the signal state of the individual bits at input TagIn_Value" and writes the result to output "TagOut_Value". The ENO enable output and the "TagOut" output are set to signal state "1".

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

DECO: Decode

Description

You can use the "Decode" instruction to set a bit in the output value specified by the input value.

The "Decode" instruction reads the value at the IN input and sets the bit in the output value whose bit position corresponds to the read value. The other bits in the output value are filled with zeroes. When the value at the IN input is greater than 31, a modulo 32 instruction is executed.

Parameters

The following table shows the parameters of the "Decode" instruction:

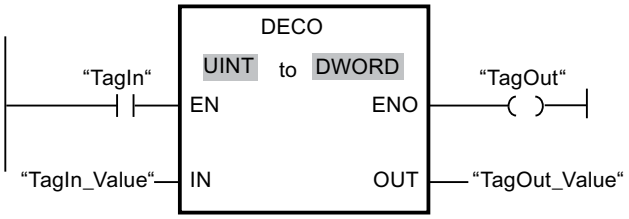
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	UINT	I, Q, M, D, L, P or constant	Position of the bit in the output value which is set.
OUT	Output	Bit strings	I, Q, M, D, L, P	Output value

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

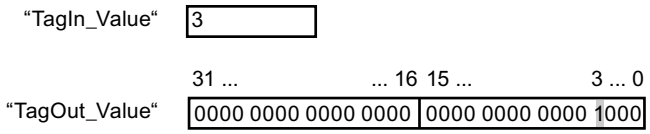
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following figure shows how the instruction works using specific operand values:



If operand "TagIn" has the signal state "1", the "Decode" instruction is executed. The instruction reads bit number "3" from the value of the operand "TagIn_Value" at the input and sets the third bit to the value of the operand "TagOut_Value" at the output.

If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

ENCO: Encode

Description

You can use the "Encode" instruction to read the bit number of the least significant bit in the input value and to send it to the OUT output.

The "Encode" instruction selects the least significant bit of the value at the IN input and writes its bit number to the tag in the OUT output.

Parameters

The following table shows the parameters of the "Encode" instruction:

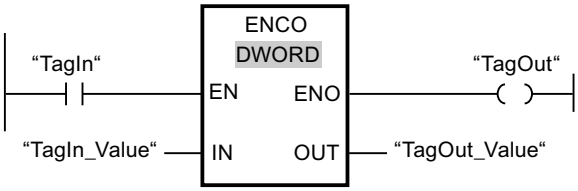
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings	I, Q, M, D, L, P or constant	Input value
OUT	Output	INT	I, Q, M, D, L, P	Output value

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

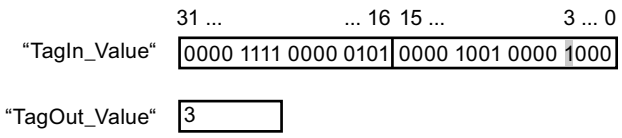
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following figure shows how the instruction works using specific operand values:



If operand "TagIn" has the signal state "1", the "Encode" instruction is executed. The instruction selects the least significant bit at the "TagIn_Value" input and writes bit position "3" to the tag in the "TagOut_Value" output.

If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

SEL: Select

Description

Depending on a switch (G input), the "Select" instruction selects one of the inputs, IN0 or IN1 and copies its content to the OUT output. When the G input has the signal state "0", the value at the IN0 input is moved. When the G input has the signal state "1", the value at the IN1 input is moved to the OUT output.

All tags at all parameters must have the same data type.

Parameters

The following table shows the parameters of the "Select" instruction:

Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
G	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Switch
IN0	Input	Bit strings, integers, floating-point numbers, TIME, TOD, CHAR, DATE	Bit strings, integers, floating-point numbers, timers, TOD, LTOD, LDT, CHAR, DATE	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	First input value

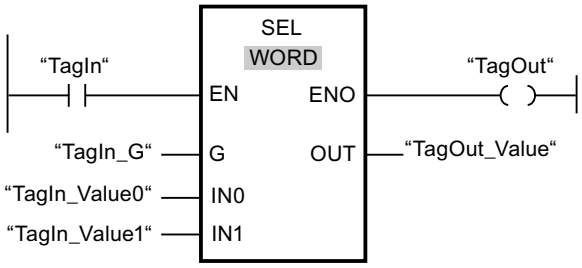
Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
IN1	Input	Bit strings, integers, floating-point numbers, TIME, TOD, CHAR, DATE	Bit strings, integers, floating-point numbers, timers, TOD, LTOD, LDT, CHAR, DATE	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Second input value
OUT	Output	Bit strings, integers, floating-point numbers, TIME, TOD, CHAR, DATE	Bit strings, integers, floating-point numbers, timers, TOD, LTOD, LDT, CHAR, DATE	I, Q, M, D, L, P	I, Q, M, D, L, P	Result

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
G	TagIn_G	0	1
IN0	TagIn_Value0	W#16#0000	W#16#4C
IN1	TagIn_Value1	W#16#FFFF	W#16#5E
OUT	TagOut_Value	W#16#0000	W#16#5E

If operand "TagIn" has the signal state "1", the "Select" instruction is executed. Based on the signal state at the "TagIn_G" input, the value at the "TagIn_Value0" or "TagIn_Value1" input is selected and copied to the "TagOut_Value" output. If the instruction is executed without errors, enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

MUX: Multiplex

Description

You can use the "Multiplex" instruction to copy the content of a selected input to the OUT output. The number of selectable inputs of the instruction box can be expanded. You can declare a maximum of 32 inputs.

The inputs are automatically numbered in the box. Numbering starts at IN0 and continues consecutively with each new input. You use the K parameter to define the input whose content is to be copied to the OUT output. If the value of the K parameter is greater than the number of available inputs, the content of the ELSE parameter is copied to the OUT output and the ENO enable output is assigned the signal state "0".

The "Multiplex" instruction can only be executed, when the tags in all inputs and in the OUT output have the same data type. The K parameter is an exception, since only integers can be specified for it.

The ENO enable output is reset if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The value of the K parameter is greater than the number of available inputs.
- Errors occurred during execution of the instruction.

Parameters

The following table shows the parameters of the "Multiplex" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output

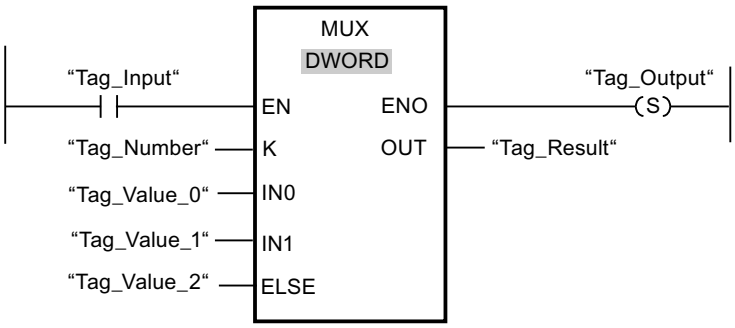
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
K	Input	Integers	Integers	I, Q, M, D, L, P or constant	Specifies the input whose content is to be copied. • If K = 0 => Parameter IN0 • If K = 1 => Parameter IN1, etc.
IN0	Input	Binary numbers, integers, floating-point numbers, strings, TIME, TOD, DATE	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	First input value
IN1	Input	Binary numbers, integers, floating-point numbers, strings, TIME, TOD, DATE	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	Second input value
INn	Input	Binary numbers, integers, floating-point numbers, strings, TIME, TOD, DATE	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	Optional input values
ELSE	Input	Binary numbers, integers, floating-point numbers, strings, TIME, TOD, DATE	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	Specifies the value to be copied when K > n.
OUT	Output	Binary numbers, integers, floating-point numbers, strings, TIME, TOD, DATE	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	Output to which the value is to be copied.

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
K	Tag_Number	1
IN0	Tag_Value_0	DW#16#00000000
IN1	Tag_Value_1	DW#16#003E4A7D
ELSE	Tag_Value_2	DW#16#FFFF0000
OUT	Tag_Result	DW#16#003E4A7D

If operand "Tag_Input" has the signal state "1", the "Multiplex" instruction is executed. Depending on the value of the operand "Tag_Number", the value at input "Tag_Value_1" is copied and assigned to the operand at output "Tag_Result". If the instruction is executed without errors, the "ENO" and "Tag_Output" enable outputs are set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

DEMUX: Demultiplex

Description

You can use the "Demultiplex" instruction to copy the content of the IN input to the selected output. The number of selectable outputs can be expanded in the instruction box. The outputs are automatically numbered in the box. Numbering starts at OUT0 and continues consecutively with each new output. You use the K parameter to define the output to which the content of the IN input is to be copied. The other outputs are not changed. If the value of the parameter K is greater than the number of available outputs, then the content of input IN in the parameter ELSE and the enable output ENO will be assigned to the signal state "0".

The "Demultiplex" instruction can only be executed if the tags in the IN input and in all outputs have the same data type. The K parameter is an exception, since only integers can be specified for it.

The ENO enable output is reset if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The value of the K parameter is greater than the number of available outputs.
- Errors occurred during execution of the instruction.

Parameters

The following table shows the parameters of the "Demultiplex" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
K	Input	Integers	Integers	I, Q, M, D, L, P or constant	Specifies the output to which the input value (IN) is copied. • If K = 0 => Parameter OUT0 • If K = 1 => Parameter OUT1, etc.
IN	Input	Binary numbers, integers, floating-point numbers, strings, TIME, TOD, DATE	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	Input value

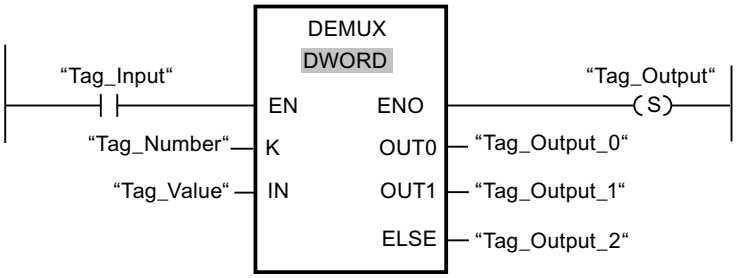
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
OUT0	Output	Binary numbers, integers, floating-point numbers, strings, TIME, TOD, DATE	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	First output
OUT1	Output	Binary numbers, integers, floating-point numbers, strings, TIME, TOD, DATE	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	Second output
OUTn	Output	Binary numbers, integers, floating-point numbers, strings, TIME, TOD, DATE	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	Optional outputs
ELSE	Output	Binary numbers, integers, floating-point numbers, strings, TIME, TOD, DATE	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	Output to which the input value (IN) at K > n is copied.

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on available data types, refer to "See also".

Example

The following example shows how the instruction works:



The following tables show how the instruction works using specific operand values:

Table 9-24 Input values of the "Demultiplex" instruction before network execution

Parameter	Operand	Values	
K	Tag_Number	1	4
IN	Tag_Value	DW#16#FFFFFFFF	DW#16#003E4A7D

Table 9-25 Output values of the "Demultiplex" instruction after network execution

Parameter	Operand	Values	
OUT0	Tag_Output_0	Unchanged	Unchanged
OUT1	Tag_Output_1	DW#16#FFFFFFFF	Unchanged
ELSE	Tag_Output_2	Unchanged	DW#16#003E4A7D

If the "Tag_Input" input has the signal state"1", the "Demultiplex" instruction is executed. Depending on the value of operand "Tag_Number", the value at the IN input is copied to the corresponding output.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

Shift and rotate

SHR: Shift right

Description

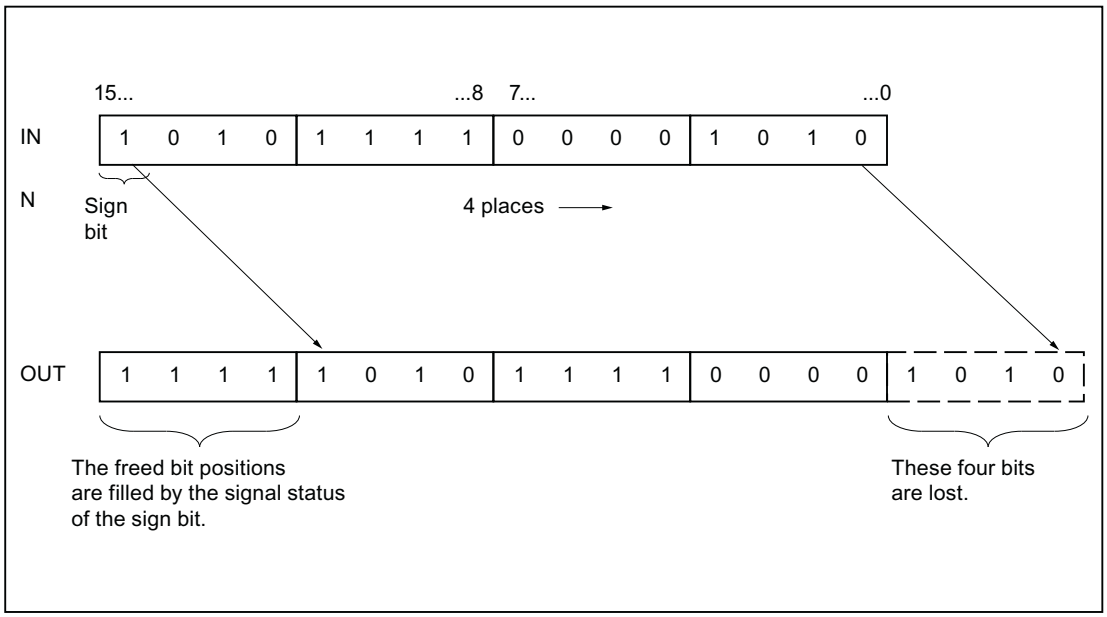
You can use the "Shift right" instruction to rotate the content of the operand at the input INbit-by-bit to the right and query the result at the OUT output. You use the N parameter to specify the number of bit positions by which the specified value is to be shifted.

When the value at the N parameter is "0", the value at the IN input is copied to the operand at the OUT output.

When the value at the N parameter is greater than the number of available bit positions, the operand value at the IN input is shifted by the available number of bit positions to the right.

In the case of unsigned values, the freed bit positions in the left area of the operand are filled with zeroes when shifting occurs. If the specified value has a sign, the free bit positions are filled with the signal state of the sign bit.

The following figure show how the content of an operand of integer data type is shifted four bit positions to the right:



Parameters

The following table shows the parameters of the "Shift right" instruction:

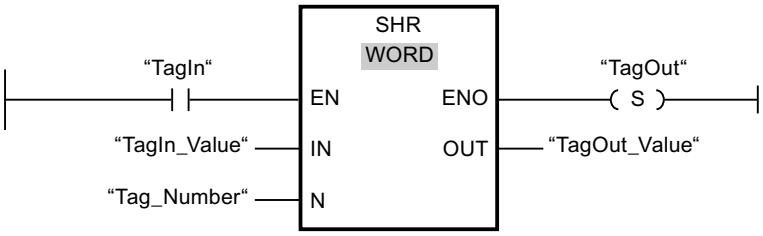
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers	Bit strings, integers	I, Q, M, D, L or constant	Value to be shifted
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L or constant	Number of bit positions by which the value is shifted
OUT	Output	Bit strings, integers	Bit strings, integers	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
N	Tag_Number	3
OUT	TagOut_Value	0000 0111 1111 0101

If operand "TagIn" has the signal state "1", the "Shift right" instruction is executed. The content of operand "TagIn_Value" is shifted three bit positions to the right. The result is sent to the "TagOut_Value" output. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

SHL: Shift left

Description

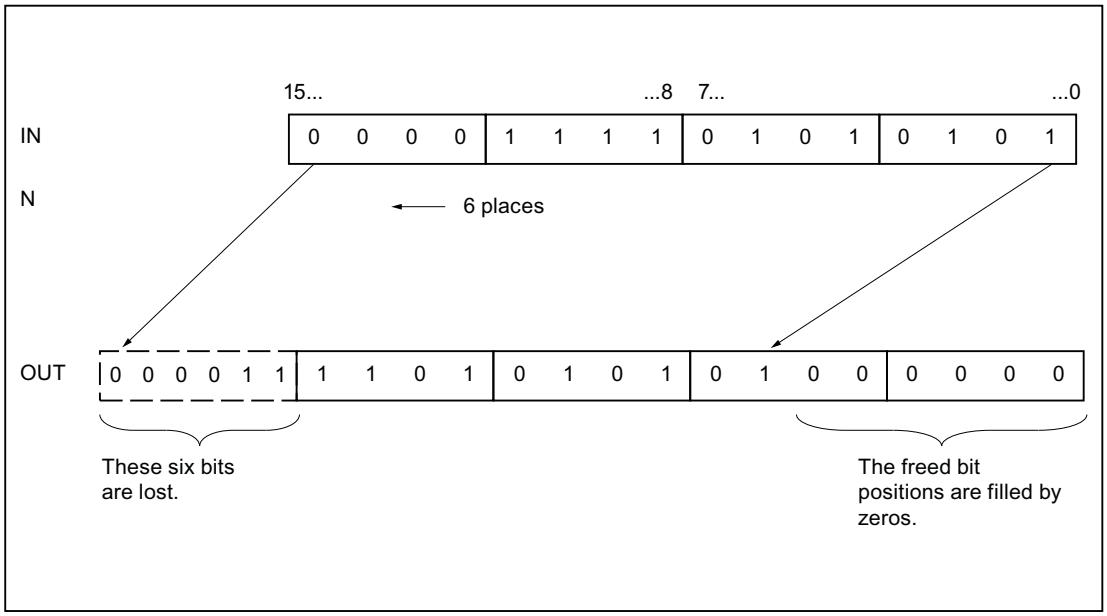
You can use the "Shift left" instruction to rotate the content of the operand at the IN input bit-by-bit to the left and query the result at the OUT output. You use the N parameter to specify the number of bit positions by which the specified value is to be shifted.

When the value at the N parameter is "0", the value at the IN input is copied to the operand at the OUT output.

When the value at the N parameter is greater than the number of available bit positions, the operand value at the IN input is shifted by the available number of bit positions to the left.

The bit positions in the right part of the operand freed by shifting are filled with zeros.

The following figure shows how the content of an operand of the WORD data type is shifted by six bit positions to the left:



Parameters

The following table shows the parameters of the "Shift left" instruction:

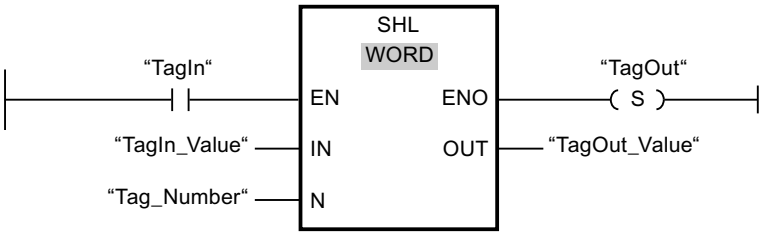
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers	Bit strings, integers	I, Q, M, D, L or constant	Value to be shifted
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L or constant	Number of bit positions by which the value is shifted
OUT	Output	Bit strings, integers	Bit strings, integers	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
N	Tag_Number	4
OUT	TagOut_Value	1111 1010 1111 0000

If operand "TagIn" has the signal state "1", the "Shift left" instruction is executed. The content of operand "TagIn_Value" is shifted four bit positions to the left. The result is sent to the "TagOut_Value" output. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

ROR: Rotate right

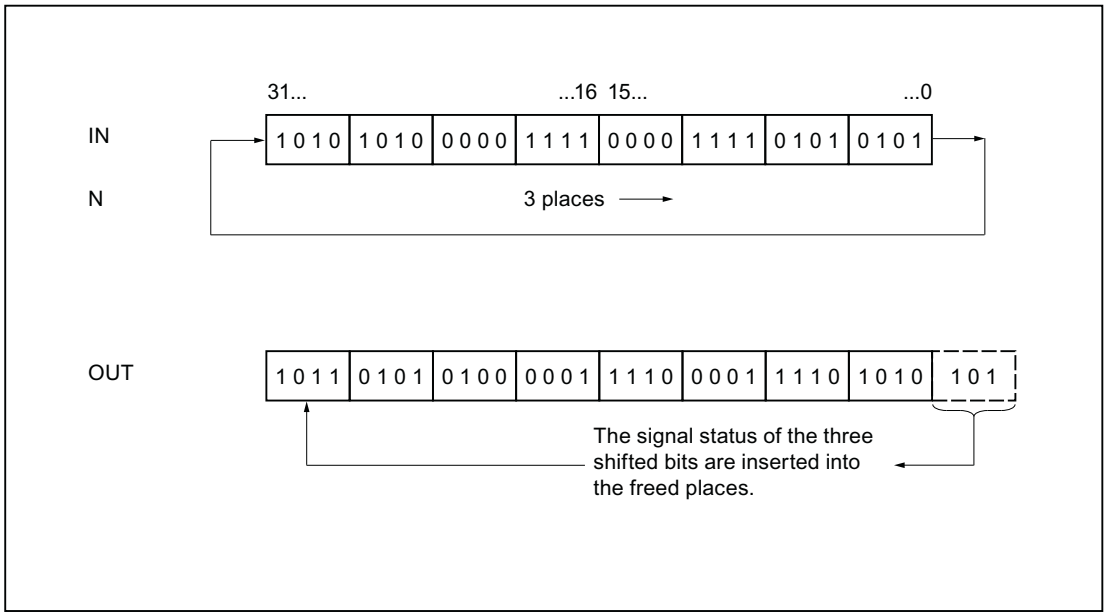
Description

You can use the "Rotate right" instruction to rotate the content of the operand at the IN input bit-by-bit to the right and query the result at the OUT output. You use the N parameter to specify the number of bit positions by which the specified value is to be rotated. The bit positions freed by rotating are filled with the bit positions that are pushed out.

When the value at the N parameter is "0", the value at the IN input is copied to the operand at the OUT output.

When the value at the N parameter is greater than the number of available bit positions, the operand value at the IN input is nevertheless rotated by the specified number of bit positions.

The following figure shows how the content of an operand of DWORD data type is rotated three positions to the right:



Parameters

The following table shows the parameters of the "Rotate right" instruction:

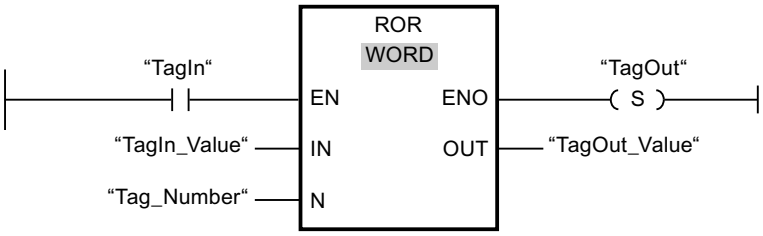
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers	Bit strings, integers	I, Q, M, D, L or constant	Value to be rotated
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L or constant	Number of bit positions by which the value is rotated
OUT	Output	Bit strings, integers	Bit strings, integers	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	TagIn_Value	0000 1111 1001 0101
N	Tag_Number	5
OUT	TagOut_Value	1010 1000 0111 1100

If operand "TagIn" has the signal state "1", the "Rotate right" instruction is executed. The content of operand "TagIn_Value" is rotated five bit positions to the right. The result is sent to the "TagOut_Value" output. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

ROL: Rotate left

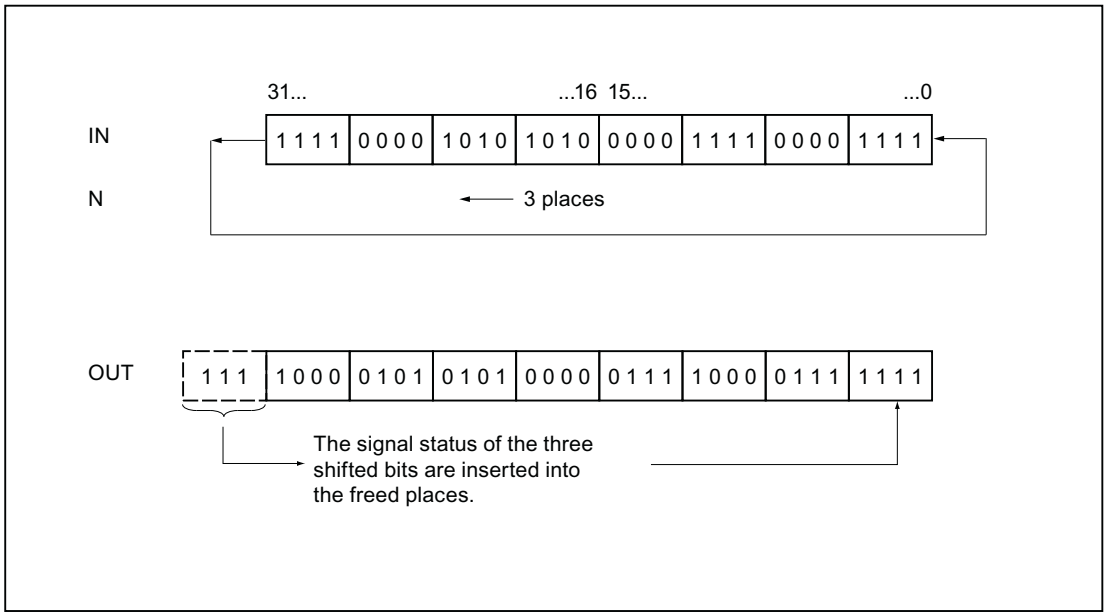
Description

You can use the "Rotate left" instruction to rotate the content of the operand at the IN input bit-by-bit to the left and query the result at the OUT output. You use the N parameter to specify the number of bit positions by which the specified value is to be rotated. The bit positions freed by rotating are filled with the bit positions that are pushed out.

When the value at the N parameter is "0", the value at the IN input is copied to the operand at the OUT output.

When the value at the N parameter is greater than the number of available bit positions, the operand value at the IN input is nevertheless rotated by the specified number of bit positions.

The following figure shows how the content of an operand of DWORD data type is rotated three positions to the left:



Parameters

The following table shows the parameters of the "Rotate left" instruction:

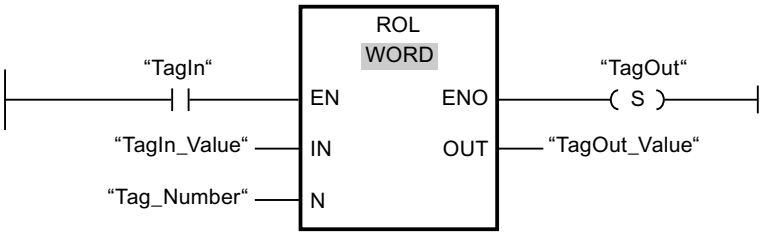
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers	Bit strings, integers	I, Q, M, D, L or constant	Value to be rotated
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L or constant	Number of bit positions by which the value is rotated
OUT	Output	Bit strings, integers	Bit strings, integers	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	TagIn_Value	1010 1000 1111 0110
N	Tag_Number	5
OUT	TagOut_Value	0001 1110 1101 0101

If the "TagIn" input has the signal state "1", the "Rotate left" instruction is executed. The content of operand "TagIn_Value" is rotated five bit positions to the left. The result is sent to the "TagOut_Value" output. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Selecting the data type of a LAD element (Page 2583)

Additional instructions

DRUM: Implement sequencer

Description

You can use the "Implement sequencer" instruction to assign the programmed values of the OUT_VAL parameter of the corresponding step to the programmed output bits (OUT1 to OUT16) and the output word (OUT_WORD). The specific step must thereby satisfy the conditions of the programmed enable mask on the S_MASK parameter while the instruction remains at this step. The instruction advances to the next step if the event for the step is true and the programmed time for the current step elapses or if the value at the JOG parameter

changes from "0" to "1". The instruction is reset if the signal state on the RESET parameter changes to "1". The current step is hereby equated to the preset step (DSP).

The amount of time spent on a step is determined by the product of the preset timebase (DTBP) and the preset counter value (S_PRESET) for each step. At the start of a new step, this calculated value is loaded into the DCC parameter, which contains the time remaining for the current step. If, for example the value of the DTBP parameter is "2" and the preset value for the first step is "100" (100 ms), the DCC parameter has the value "200" (200 ms).

A step can be programmed with a timer value, an event or both. Steps that have an event bit and the timer value "0" advance to the next step as soon as the signal state of the event bit is "1". Steps that are programmed only with a timer value start the time immediately. Steps that are programmed with an event bit and a timer value greater than "0" start the time when the signal state of the event bit is "1". The event bits are initialized with a signal state of "1".

When the sequencer is on the last programmed step (LST_STEP) and the time for this step has expired, the signal state on the Q parameter is set to "1"; otherwise it is set to "0". When the parameter Q is set, the instruction remains on the step until it is reset.

In the configurable mask (S_MASK) you can selected the separate bits in the output word (OUT_WORD) and set or reset the output bits (OUT1 to OUT16) by means of the output values (OUT_VAL). If a bit of the configurable mask is in the signal state "1", the value OUT_VAL sets or resets the corresponding bit. If the signal state of a bit of the configurable mask is "0", the corresponding bit is left unchanged. All the bits of the configurable mask for all 16 steps are initialized with a signal state of "1".

The output bit at the OUT1 parameter corresponds to the least significant bit of the output word (OUT_WORD). The output bit at the OUT16 parameter corresponds to the most significant bit of the output word (OUT_WORD).

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the "Implement sequencer" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
RESET	Input	BOOL	I, Q, M, D, L or constant	The signal state "1" indicates the reset condition.
JOG	Input	BOOL	I, Q, M, D, L or constant	When the signal state changes from "0" to "1", the instruction advances to the next step.

Parameter	Declaration	Data type	Memory area	Description
DRUM_EN	Input	BOOL	I, Q, M, D, L or constant	The signal state "1" causes the sequencer to advance based on the event and time criteria.
LST_STEP	Input	BYTE	I, Q, M, D, L or constant	Number of the last programmed step.
EVENT(i), 1 ≤ i ≤ 16	Input	BOOL	I, Q, M, D, L or constant	Event bit (i); Initial signal state is "1".
OUT(j), 1 ≤ j ≤ 16	Output	BOOL	I, Q, M, D, L	Output bit (j)
Q	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that the time for the last step has elapsed.
OUT_WORD	Output	WORD	I, Q, M, D, L, P	Word address to which the sequencer writes the output values.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
JOG_HIS	Static	BOOL	I, Q, M, D, L	JOG parameter history bit
EOD	Static	BOOL	I, Q, M, D, L or constant	The signal state "1" indicates that the time for the last step has elapsed.
DSP	Static	BYTE	I, Q, M, D, L, P or constant	Preset step of the sequencer
DSC	Static	BYTE	I, Q, M, D, L, P or constant	Current step of the sequencer
DCC	Static	DWORD	I, Q, M, D, L, P or constant	Current numerical value of the sequencer
DTBP	Static	WORD	I, Q, M, D, L, P or constant	Preset timebase of the sequencer
PREV_TIME	Static	DWORD	I, Q, M, D, L or constant	Previous system time
S_PRESET	Static	ARRAY of WORD	I, Q, M, D, L or constant	Preset count for each step [1 to 16] where 1 clock pulse = 1 ms.

Parameter	Declaration	Data type	Memory area	Description
OUT_VAL	Static	ARRAY of BOOL	I, Q, M, D, L or constant	Output values for each step [1 to 16, 0 to 15].
S_MASK	Static	ARRAY of BOOL	I, Q, M, D, L or constant	Configurable mask for each step [1 to 16, 0 to 15]. Initial signal states are "1".

For additional information on valid data types, refer to "See also".

Parameters ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

ERR_CODE*	Explanation
W#16#0000	No error
W#16#000B	The value at the LST_STEP parameter is less than 1 or greater than 16.
W#16#000C	The value at the DSC parameter is less than 1 or greater than the value of the LST_STEP parameter.
W#16#000D	The value at the DSP parameter is less than 1 or greater than the value of the LST_STEP parameter.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Switching display formats in the program status (Page 2980)

DCAT: Discrete control-timer alarm

Description

The instruction "Discrete control-timer alarm" is used to accumulate the time from the point at which the CMD parameter issued the command to open or close. The time is accumulated until the preset time (PT) is exceeded or the information is received that the device was opened or closed (O_FB or C_FB) within the specified time. If the preset time is exceeded before the information on the opening or closing of the device is received, the corresponding alarm is

activated. If the signal state of the command input changes state before the preset time, the time is restarted.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The "Discrete control-timer alarm" instruction has the following reactions to the input conditions:

- When the signal state of the CMD parameter changes from "0" to "1", the signal states of the parameters Q, CMD_HIS, ET (only if ET < PT), OA and CA are influenced as follows:
  - The parameters Q and CMD_HIS are set to "1".
  - The parameters ET, OA and CA are reset to "0".
- When the signal state of the parameter CMD changes from "1" to "0", the parameters Q, ET (only if ET < PT), OA, CA and CMD_HIS are reset to "0".
- When the signal state of the parameters CMD and CMD_HIS is "1" and the parameter O_FB is set to "0", the time difference (ms) since the last execution of the instruction is added to the value at the parameter ET. If the value of the parameter ET exceeds the value of the parameter PT, the signal state on the parameter OA is set to "1". If the value of the ET parameter does not exceed the value of the parameter PT, the signal state on the parameter OA is reset to "0". The value at the parameter CMD_HIS is reset to the value of the parameter CMD.
- If the signal state of the CMD, CMD_HIS and O_FB parameters are set to "1" and the parameter C_FB has the value "0", the signal state of the parameter OA is set to "0". The value of the parameter ET is set to the value of the parameter PT. If the signal state of the O_FB parameter changes to "0", the alarm is set the next time the instruction is executed. The value of the parameter CMD_HIS is set to the value of the parameter CMD.
- If the parameters CMD, CMD_HIS and C_FB have the value "0", the time difference (ms) since the last execution of the instruction is added to the value of the parameter ET. If the value of the ET parameter exceeds the value of the parameter PT, the signal state of the parameter CA is reset to "1". If the value at the parameter PT is not exceeded, the parameter CA has the signal state "0". The value of the parameter CMD_HIS is set to the value of the parameter CMD.
- If the parameters CMD, CMD_HIS and O_FB have the signal state "0" and the parameter C_FB is set to "1", the parameter CA is set to "0". The value of the parameter ET is set to the value of the parameter PT. If the signal state of the C_FB parameter changes to "0", the alarm is set the next time the instruction is executed. The value of the parameter CMD_HIS is set to the value of the parameter CMD.
- If the parameters O_FB and C_FB simultaneously have the signal state "1", the signal states of both alarm outputs are set to "1".

The "Discrete control-timer alarm" instruction returns no error information.

Parameters

The following table shows the parameters of the "Discrete control-timer alarm" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
CMD	Input	BOOL	I, Q, M, D, L or constant	The signal state "0" indicates a "close" command. The signal state "1" indicates the "open" command.
O_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when closing
Q	Output	BOOL	I, Q, M, D, L	Shows the status of the parameter CMD
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
ET	Static	DINT	D, L or constant	Currently elapsed time, where one clock pulse = 1 ms
PT	Static	DINT	D, L or constant	Preset timer value, where one clock pulse = 1 ms
PREV_TIME	Static	DWORD	D, L or constant	Previous system time
CMD_HIS	Static	BOOL	D, L or constant	CMD history bit

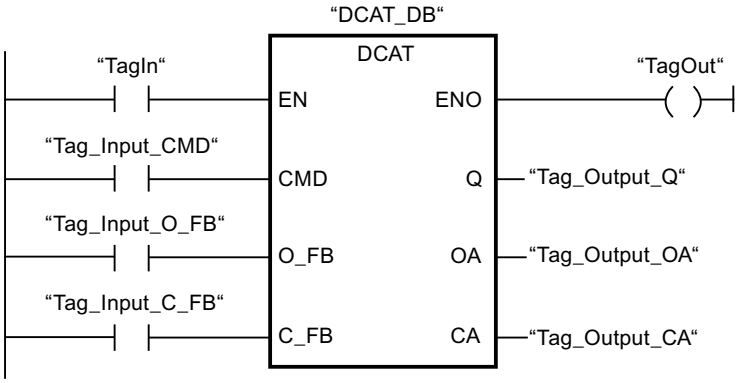
For additional information on valid data types, refer to "See also".

Example

In the following example the parameter CMD changes from "0" to "1". After the execution of the instruction the parameter Q is set to "1" and the two alarm outputs OA and CA have the signal state "0". The parameter CMD_HIS of the instance data block is set to the signal state "1" and the parameter ET is reset to "0".

Note

You can initialize static parameters in the data block.



The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameter	Operand	Value
CMD	Tag_Input_CMD	TRUE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
Q	Tag_Output_Q	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the instance data block "DCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#12
PT	DBD8	L#222
CMD_HIS	DBX16.0	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
Q	Tag_Output_Q	TRUE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the instance data block "DCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#0
CMD_HIS	DBX16.0	TRUE

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

MCAT: Motor control-timer alarm

Description

The instruction "Motor control-timer alarm" is used to accumulate the time from the point in time as of which one of the command inputs (open or close) is activated. The time is accumulated until the preset time is exceeded or the relevant feedback input indicates that the device has executed the requested operation within the specified time. If the preset time is exceeded before the feedback is received, the corresponding alarm is triggered.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Execution of the "Motor control-timer alarm" instruction

The following table shows the reactions of the "Motor control-timer alarm" instruction to the various input conditions:

Input parameters								Output parameters								
ET	O_HIS	C_HIS	O_CMD	C_CMD	S_CMD	O_FB	C_FB	OO	CO	OA	CA	ET	O_HIS	C_HIS	Q	Status
X	1	1	X	X	X	X	X	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	X	1	1	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	1	X	X	0	0	0	0	X	0	0	1	Stop
X	X	X	1	1	X	X	X	0	0	0	0	X	0	0	1	Stop
X	0	X	1	0	0	X	X	1	0	0	0	0	1	0	1	Start opening
<PT	1	0	X	0	0	0	X	1	0	0	0	INC	1	0	1	Open
X	1	0	X	0	0	1	0	0	0	0	0	PT	1	0	1	Opened
>=PT	1	0	X	0	0	0	X	0	0	1	0	PT	1	0	0	Opening alarm
X	X	0	0	1	0	X	X	0	1	0	0	0	0	1	1	Start closing
<PT	0	1	0	X	0	X	0	0	1	0	0	INC	0	1	1	Close
X	0	1	0	X	0	0	1	0	0	0	0	PT	0	1	1	Closed

Input parameters								Output parameters								
>=PT	0	1	0	X	0	X	0	0	0	0	1	PT	0	1	0	Closing alarm
X	0	0	0	0	0	X	X	0	0	0	0	X	0	0	1	Stopped

Legend:	
INC	Add the time difference (ms) since the last processing of the FB to ET
PT	PT is set to the same value as ET
X	Cannot be used
<PT	ET < PT
>=PT	ET >= PT

If the input parameters O_HIS and C_HIS both have the signal state "1", they are immediately set to signal state "0". In this case, the last row in the table (X) mentioned above is valid. Because it is therefore no longer possible to check whether the input parameters O_HIS and C_HIS have the signal state "1", the output parameters are set as follows in this case:

OO = FALSE

CO = FALSE

OA = FALSE

CA = FALSE

ET = PT

Q = TRUE

The "Motor control-timer alarm" instruction returns no error information.

Parameters

The following table shows the reactions of the "Motor control-timer alarm" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
O_CMD	Input	BOOL	I, Q, M, D, L or constant	"Open" command input
C_CMD	Input	BOOL	I, Q, M, D, L or constant	"Close" command input
S_CMD	Input	BOOL	I, Q, M, D, L or constant	"Stop" command input
O_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when closing
OO	Output	BOOL	I, Q, M, D, L	"Open" output
CO	Output	BOOL	I, Q, M, D, L	"Close" output
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing

Parameter	Declaration	Data type	Memory area	Description
Q	Output	BOOL	I, Q, M, D, L	The signal state "0" indicates an error condition.
ET	Static	DINT	D, L or constant	Currently elapsed time, where one clock pulse = 1 ms
PT	Static	DINT	D, L or constant	Preset timer value, where one clock pulse = 1 ms
PREV_TIME	Static	DWORD	D, L or constant	Previous system time
O_HIS	Static	BOOL	D, L or constant	"Open" history bit
C_HIS	Static	BOOL	D, L or constant	"Close" history bit

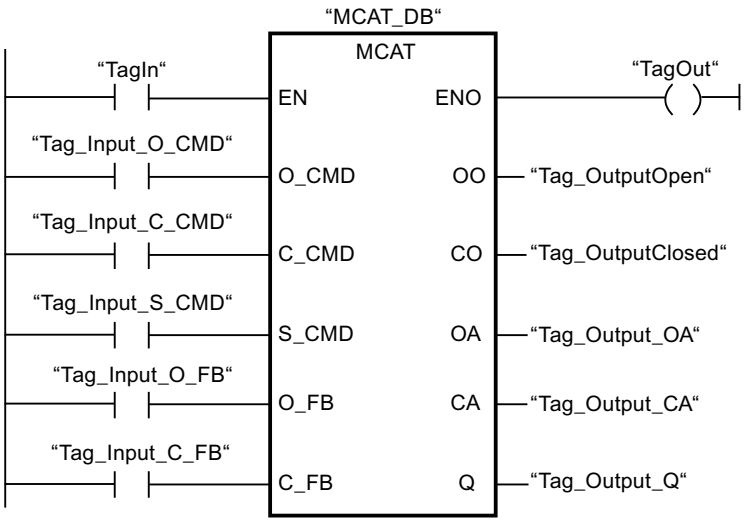
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.



The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameter	Operand	Value
O_CMD	Tag_Input_O_CMD	TRUE
C_CMD	Tag_Input_C_CMD	FALSE
S_CMD	Tag_Input_S_CMD	FALSE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
OO	Tag_OutputOpen	FALSE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	FALSE

The following values are saved in the instance data block "MCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#2
PT	DBD8	L#22
O_HIS	DBX16.0	TRUE
C_HIS	DBX16.1	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OO	Tag_OutputOpen	TRUE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	TRUE

The following values are saved in the instance data block "MCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#0
O_HIS	DBX16.0	TRUE
CMD_HIS	DBX16.1	FALSE

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

IMC: Compare input bits with the bits of a mask

Description

The instruction "Compare input bits with the bits of a mask" is used to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bits of a mask. Up to 16 steps with masks can be programmed. The value of the IN_BIT0 parameter is compared with the value of the mask CMP_VAL[x,0], with "x" indicating the step number. On the CMP_STEP parameter, you specify the step number of the mask that is used for the comparison. All programmed values are compared in the same manner. Unprogrammed input bits or unprogrammed bits of the mask have the default signal state FALSE.

If a match is found in the comparison, the signal state of the OUT parameter is set to "1". Otherwise, the OUT parameter is set to "0".

If the value of the CMP_STEP parameter is greater than 15, the instruction is not executed. An error message is output at the ERR_CODE parameter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the "Compare input bits with the bits of a mask" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN_BIT0	Input	BOOL	I, Q, M, D, L or constant	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L or constant	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L or constant	Input bit 2 is compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L or constant	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L or constant	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L or constant	Input bit 5 is compared with bit 5 of the mask.

Parameter	Declaration	Data type	Memory area	Description
IN_BIT6	Input	BOOL	I, Q, M, D, L or constant	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L or constant	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L or constant	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L or constant	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L or constant	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L or constant	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L or constant	Input bit 12 is compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L or constant	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L or constant	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L or constant	Input bit 15 is compared with bit 15 of the mask.
CMP_STEP	Input	BYTE	I, Q, M, D, L, P or constant	The step number of the mask used for the comparison.
OUT	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that a match was found. The signal state "0" indicates that a match was found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
CMP_VAL	Static	ARRAY OF WORD	I, Q, M, D, L or constant	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

For additional information on valid data types, refer to "See also".

Parameters ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000A	The value at the CMP_STEP parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Switching display formats in the program status (Page 2980)

SMC: Compare scan matrix

Description

The "Compare scan matrix" instruction is used to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bits of the comparison masks for each step. Processing starts at step 1 and is continued until the last programmed step (LAST) or until a match is found. The input bit of the IN_BIT0 parameter is compared with the value of the mask CMP_VAL[x,0], with "x" indicating the step number. All programmed values are compared in the same manner. If a match is found the signal state of the OUT parameter is set to "1" and the step number with the matching mask is written in the OUT_STEP parameter. Unprogrammed input bits or unprogrammed bits of the mask have the default signal state FALSE. If more than one step has a matching mask, only the first one found is indicated in the OUT_STEP parameter. If no match is found, the signal state of the OUT parameter is set to "0". In this case the value at the OUT_STEP parameter is greater by "1" than the value of the LAST parameter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the "Compare scan matrix" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN_BIT0	Input	BOOL	I, Q, M, D, L or constant	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L or constant	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L or constant	Input bit 2 is compared with bit 2 of the mask.

Parameter	Declaration	Data type	Memory area	Description
IN_BIT3	Input	BOOL	I, Q, M, D, L or constant	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L or constant	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L or constant	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L or constant	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L or constant	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L or constant	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L or constant	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L or constant	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L or constant	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L or constant	Input bit 12 is compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L or constant	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L or constant	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L or constant	Input bit 15 is compared with bit 15 of the mask.
OUT	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that a match was found. The signal state "0" indicates that no match was found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
OUT_STEP	Output	BYTE	I, Q, M, D, L, P	Contains the step number with the matching mask or the step number which is greater by "1" than the value of the LAST parameter, provided no match is found.
LAST	Static	BYTE	I, Q, M, D, L, P or constant	Specifies the step number of the last step to be scanned for a matching mask.
CMP_VAL	Static	ARRAY OF WORD	I, Q, M, D, L or constant	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

For additional information on valid data types, refer to "See also".

Parameters ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000E	The value at the LAST parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Switching display formats in the program status (Page 2980)

LEAD_LAG: Lead and lag algorithm

Description

The "Lead and lag algorithm" instruction is used to process signals with an analog tag. The gain value at the GAIN parameter must be greater than zero. The result of the instruction "Lead and lag algorithm" is calculated using the following equation:

$$OUT = \left[\frac{LG_TIME}{LG_TIME + SAMPLE_T}\right] PREV_OUT + GAIN \left[\frac{LD_TIME + SAMPLE_T}{LG_TIME + SAMPLE_T}\right] IN - GAIN \left[\frac{LD_TIME}{LG_TIME + SAMPLE_T}\right] * PREV_IN$$

The "Lead and lag algorithm" instruction supplies plausible results only when processing is in fixed program cycles. The same units must be specified for the LD_TIME, LG_TIME, and SAMPLE_T parameters. At LG_TIME > 4 + SAMPLE_T, the instruction approaches the following function:

OUT = GAIN * ((1 + LD_TIME * s) / (1 + LG_TIME * s)) * IN

When the value of the GAIN parameter is less than or equal to zero, the calculation is not performed and an error information is output on the ERR_CODE parameter.

You can use the "Lead and lag algorithm" instruction in conjunction with loops as a compensator in dynamic feed-forward control. The instruction consists of two operations. The "Lead" operation shifts the phase of the OUT output so that the output leads the input. The "Lag" operation, on the other hand, shifts the output so that the output lags behind the input. Because the "Lag" operation is equivalent to an integration, it can be used as a noise suppressor or as a low-pass filter. The "Lead" operation is equivalent to a differentiation and can therefore be used as a high-pass filter. The two operations together (Lead and Lag) result in the output phase lagging behind the the input at lower frequencies and leading it at higher

frequencies. This means that the "Lead and lag algorithm" instruction can be used as a band pass filter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the "Lead and lag algorithm" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	REAL	I, Q, M, D, L, P or constant	The input value of the current sample time (cycle time) to be processed. Constants can also be specified on the IN parameter.
SAMPLE_T	Input	INT	I, Q, M, D, L, P or constant	Sample time Constants can also be specified on the SAMPLE_T parameter.
OUT	Output	REAL	I, Q, M, D, L	Result of the instruction
ERR_CODE	Output	WORD	I, Q, M, D, L	Error information
LD_TIME	Static	REAL	I, Q, M, D, L, P or constant	Lead time in the same unit as sample time.
LG_TIME	Static	REAL	I, Q, M, D, L, P or constant	Lag time in in the same unit as sample time.
GAIN	Static	REAL	I, Q, M, D, L, P or constant	Gain as % / % (the ratio of the change in output to a change in input as a steady state).
PREV_IN	Static	REAL	I, Q, M, D, L, P or constant	Previous input
PREV_OUT	Static	REAL	I, Q, M, D, L, P or constant	Previous output

For additional information on valid data types, refer to "See also".

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

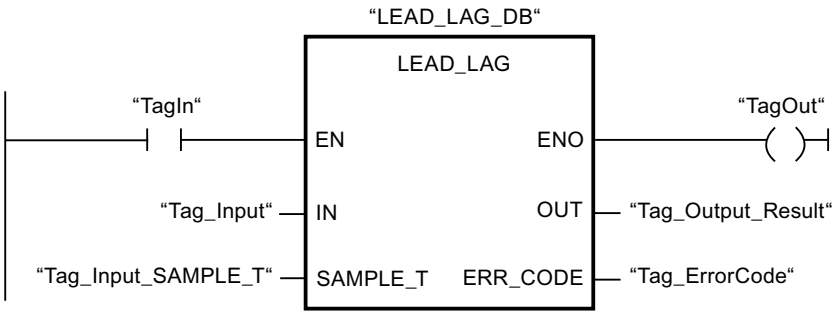
Error code* (W#16#...)	Explanation
0000	No error
0009	The value at the GAIN parameter is less than or equal to zero.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.



The following table shows how the instruction works using specific operand values:

Before processing

In this example the following values are used for the input parameters:

Parameter	Operand	Value
IN	Tag_Input	2.0
SAMPLE_T	Tag_InputSampleTime	10

The following values are saved in the instance data block "LEAD_LAG_DB" of the instruction:

Parameter	Address	Value
LD_TIME	DBD12	2.0
LG_TIME	DBD16	2.0
GAIN	DBD20	1.0
PREV_IN	DBD24	6.0
PREV_OUT	DBD28	6.0

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OUT	Tag_Output_Result	2.0

The following values are saved in the instance data block "LEAD_LAD_DB" of the instruction:

Parameter	Operand	Value
PREV_IN	DBD24	2.0
PREV_OUT	DBD28	2.0

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Switching display formats in the program status (Page 2980)

SEG: Create bit pattern for seven-segment display

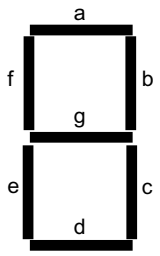
Description

The instruction "Create bit pattern for seven-segment display" is used to convert each of the four hexadecimal digits of the specified source word (IN) into an equivalent bit pattern for a seven-segment display. The result of the instruction is output in the double word on the OUT parameter.

The following relation exists between the hexadecimal digits and the assignment of the 7 segments (a, b, c, d, e, f, g):

Input digit (Binary)	Assignment of the segments - g f e d c b a	Display (Hexadecimal)
0000	00111111	0
0001	00000110	1
0010	01011011	2
0011	01001111	3
0100	01100110	4
0101	01101101	5
0110	01111101	6
0111	00000111	7

Seven-segment display



1000	01111111	8
1001	01100111	9
1010	01110111	A
1011	01111100	B
1100	00111001	C
1101	01011110	D
1110	01111001	E
1111	01110001	F

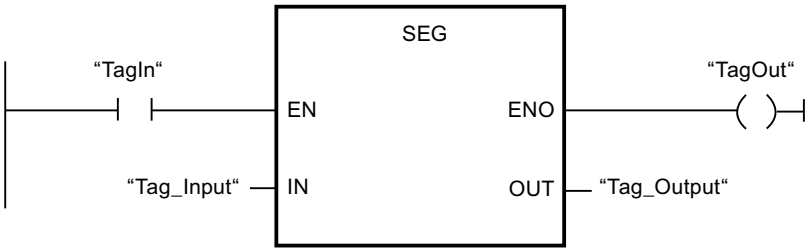
Parameter

The following table shows the parameters of the "Create bit pattern for seven-segment display" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	WORD	I, Q, M, D, L, P or constant	Source word with four hexadecimal digits
OUT	Output	DWORD	I, Q, M, D, L, P	Bit pattern for the seven-segment display

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
Hexadecimal	Binary		
IN	Tag_Input	W#16#1234	0001 0010 0011 0100
OUT	Tag_Output	DW#16065B4F66	000 00110 0101 1011 0100 1111 0110 0110 Display: 1234

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

BCDCPL: Create tens complement

Description

You can use the "Create tens complement" instruction to create the tens complement of a seven-digit BCD number specified in the IN parameter. This instruction uses the following mathematical formula to calculate:

10000000 (as BCD)

– 7-digit BCD value

Tens complement (as BCD)

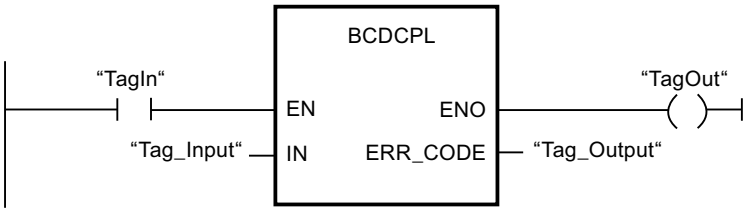
Parameters

The following table shows the parameters of the "Create tens complement" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings	I, Q, M, D, L, P or constant	7-digit BCD number
ERR_CODE	Output	DWORD	I, Q, M, D, L, P	Result of the instruction

Example

The following example shows how the instruction works:



The following table shows how the instruction functions using specific values:

Parameter	Operand	Value*
IN	Tag_Input	DW#16#01234567
ERR_CODE	Tag_Output	DW#16#08765433
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".		

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Switching display formats in the program status (Page 2980)

BITSUM: Count number of set bits

Description

The "Count number of set bits" instruction is used to count the number of bits of an operand that is set to the signal state "1". The operand whose bits are to be counted is specified on the IN parameter. The result of the instruction is output on the RET_VAL parameter.

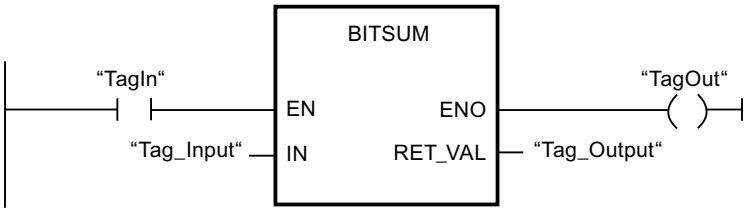
Parameters

The following table shows the parameters of the "Count number of set bits" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	DWORD	I, Q, M, D, L, P or constant	Operand whose set bits are counted
RET_VAL	Output	INT	I, Q, M, D, L, P	Number of bits to be set

Example

The following example shows how the instruction works:



The following table shows how the instruction functions using specific values:

Parameter	Operand	Value*
IN	Tag_Input	DW#16#12345678
RET_VAL	Tag_Output	W#16#000D (13 Bits)
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".		

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Switching display formats in the program status (Page 2980)

FBD

Bit logic operations

&: AND logic operation

Description

You can use the instruction "AND logic operation" to query the signal states of two or more specified operands and evaluate them according to the AND truth table.

If the signal state of all the operands is "1", then the condition is fulfilled and the instruction returns the result "1". If the signal state of one of the operands is "0", then the condition is not fulfilled and the instruction generates the result "0".

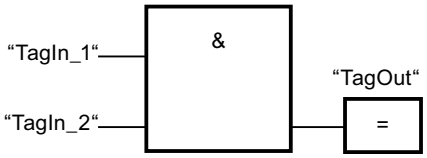
Parameters

The following table shows the parameters of the instruction "AND logic operation":

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand>	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	The operand indicates the bit whose signal state will be queried.

Example

The following example shows how the instruction works:



Output "TagOut" is set, when the signal state of the operands "TagIn_1" and "TagIn_2" is "1" and reset when the state of the operands "TagIn_1" and "TagIn_2" is "0".

See also

AND truth table (Page 3447)

Example of detecting the direction of a conveyor belt (Page 3092)

Example of controlling room temperature (Page 3097)

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Adding additional inputs and outputs to FBD elements (Page 2639)

Inserting operands in FBD instructions (Page 2643)

Insert input (Page 3451)

AND truth table

Results of the logic operation

The following table shows the results that arise from the AND logic operation of two operands:

Signal state of the first operand	Signal state of the second operand	Result of the logic operation
1	1	1
0	1	0
1	0	0
0	0	0

See also

&: AND logic operation (Page 3445)

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

>=1: OR logic operation

Description

You can use the instruction "OR logic operation" to query the signal states of two or more specified operands and evaluate them according to the OR truth table.

If the signal state of one of the operands is "1", then the condition is fulfilled and the instruction returns the result "1". If the signal state of all the operands is "0", then the condition is not fulfilled and the instruction generates the result "0".

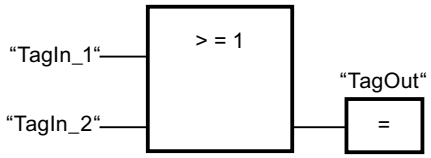
Parameters

The following table shows the parameters of the instruction "OR logic operation":

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand>	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	The operand indicates the bit whose signal state will be queried.

Example

The following example shows how the instruction works:



Output "TagOut" is set, when the signal state of the operands "TagIn_1" or "TagIn_2" is "1".

See also

OR truth table (Page 3448)
Example of controlling a conveyor belt (Page 3090)
Adding additional inputs and outputs to FBD elements (Page 2639)
Inserting FBD elements (Page 2621)
Editing FBD elements (Page 2635)
Inserting operands in FBD instructions (Page 2643)
Overview of the valid data types (Page 2236)
Insert input (Page 3451)

OR truth table

Results of the logic operation

The following table shows the results that arise from the OR logic operation of two operands:

Signal state of the first operand	Signal state of the second operand	Result of the logic operation
1	0	1
0	1	1
1	1	1
0	0	0

See also

>=1: OR logic operation (Page 3447)
Overview of the valid data types (Page 2236)
Inserting FBD elements (Page 2621)
Editing FBD elements (Page 2635)
Inserting operands in FBD instructions (Page 2643)

X: EXCLUSIVE OR logic operation

Description

You can use the instruction "EXCLUSIVE OR logic operation" to query the result of a signal state query according to the EXCLUSIVE OR truth table.

With an instruction "EXCLUSIVE OR logic operation", the signal state is "1" when the signal state of one of the two specified operands is "1". When more than two operands are queried, the common RLO is "1" if an odd number of the queried operands returns the result "1".

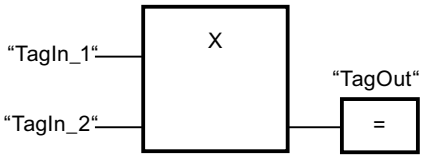
Parameters

The following table shows the parameters of the instruction "EXCLUSIVE OR logic operation":

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand>	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	The operand indicates the bit whose signal state will be queried.

Example

The following example shows how the instruction works:



Output "TagOut" is set when the signal state of the operands "TagIn_1" and "TagIn_2" is "1". When both operands return the signal state "1" or "0", the output "TagOut" is reset.

See also

EXCLUSIVE OR truth table (Page 3450)

Adding additional inputs and outputs to FBD elements (Page 2639)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Overview of the valid data types (Page 2236)

Insert input (Page 3451)

EXCLUSIVE OR truth table

Results of the logic operation

The following table shows the results that arise from the EXCLUSIVE OR logic operation of two operands:

Signal state of the first operand	Signal state of the second operand	Result of the logic operation
1	0	1
0	1	1
1	1	0
0	0	0

The following table shows the results that arise from the EXCLUSIVE OR logic operation of three operands:

Signal state of the first operand	Signal state of the second operand	Signal state of the third operand	Result of the logic operation
1	0	0	1
0	1	1	0
0	1	0	1
1	0	1	0
0	0	1	1
1	1	0	0
1	1	1	1
0	0	0	0

See also

X: EXCLUSIVE OR logic operation (Page 3449)

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Insert input

Description

The "Insert input" instruction is used to add an input to the box of one of the following instructions:

- "AND logic operation"
- "OR logic operation"
- "EXCLUSIVE OR logic operation"

You can query the signal state of several operands by the extension of an instruction box.

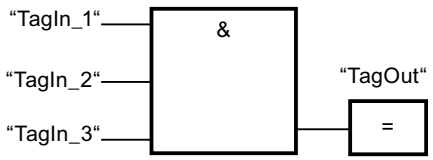
Parameters

The following table shows the parameters of the instruction "Insert input":

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand>	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	The operand indicates the bit whose signal state will be queried.

Example

The following example shows how the instruction works:



The box of the instruction "AND logic operation" was extended by an additional input at which the signal state of the operand "TagIn_3" is queried. The output "TagOut" is set, when the signal state of the operands "TagIn_1", "TagIn_2" and "TagIn_3" returns the signal state "1".

See also

&: AND logic operation (Page 3445)

>=1: OR logic operation (Page 3447)

X: EXCLUSIVE OR logic operation (Page 3449)

Inserting operands in FBD instructions (Page 2643)

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

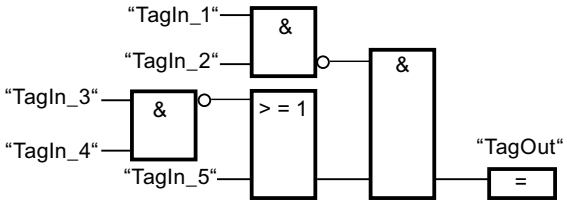
Invert RLO

Description

You use the "Invert RLO" instruction to invert the signal state of the result of logic operation (RLO).

Example

The following example shows how the instruction works:



The output "TagOut" is set when the following conditions are fulfilled:

- The input "TagIn_1" and/or "TagIn_2" has signal state "0".
- The input "TagIn_3" and/or "TagIn_4" has signal state "0" or the input "TagIn_5" has signal state "1".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

=: Assignment

Description

You can use the instruction "Assignment" to set the bit of a specified operand. If the result of logic operation (RLO) at the box input has the signal state "1", the specified operand is set to signal state "1". If the signal state at the box input is "0", the bit of the specified operand is reset to "0".

The instruction does not influence the RLO. The RLO at the box input is assigned directly to the operand above the assignment box.

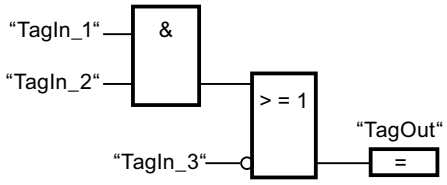
Parameters

The following table shows the parameters of the instruction "Assignment":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand to which the RLO is assigned.

Example

The following example shows how the instruction works:



The operand "TagOut" is set at the output of the "Assignment" instruction when one of the following conditions is fulfilled:

- The inputs "TagIn_1" and "TagIn_2" have the signal state "1".
- The signal state at the input "TagIn_3" is "0".

See also

Overview of the valid data types (Page 2236)

Example of detecting the fill level of a storage area (Page 3093)

Example of controlling room temperature (Page 3097)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

/=: Negate assignment

Description

The instruction "Negate assignment" inverts the result of logic operation (RLO) and assigns this to the operand above the box. If the RLO at the input of the box is "1", the binary operand is reset. If the RLO at the input of the box is "0", the operand is set to signal state "1".

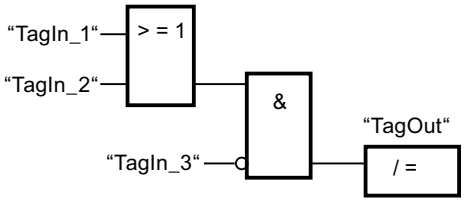
Parameters

The following table shows the parameters of the instruction "Negate assignment":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand to which the negated RLO is assigned.

Example

The following example shows how the instruction works:



The operand "TagOut" is reset when the following conditions are fulfilled:

- The operand "TagIn_1" or "TagIn_2" has the signal state "1".
- The operand "TagIn_3" has the signal state "0".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

R: Reset output

Description

You can use the "Reset output" instruction to reset the signal state of a specified operand to "0".

The instruction is only executed if the result of logic operation (RLO) at the box input is "1". If the box input has the signal state "1", the specified operand is reset to "0". If there is an RLO of "0" at the box input, the signal state of the specified operand remains unchanged.

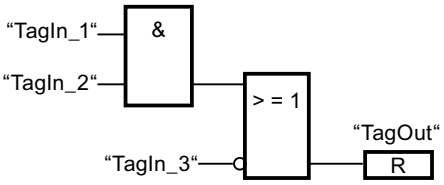
Parameters

The following table shows the parameters of the "Reset output" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand>	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Operand that is reset if RLO = "1".

Example

The following example shows how the instruction works:



Operand "TagOut" is reset when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The operand "TagIn_3" has the signal state "0".

See also

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Example of controlling a conveyor belt (Page 3090)

Example of detecting the direction of a conveyor belt (Page 3092)

Overview of the valid data types (Page 2236)

S: Set output

Description

You can use the instruction "Set output" to set the signal state of a specified operand to "1".

The instruction is only executed if the result of logic operation (RLO) at the box input is "1". If the box input has the signal state "1", the specified operand is set to "1". If there is an RLO of "0" at the box input, the signal state of the specified operand remains unchanged.

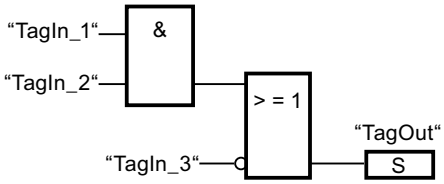
Parameters

The following table shows the parameters of the instruction "Set output":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand which is set with RLO = "1".

Example

The following example shows how the instruction works:



The "TagOut" operand is set when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The operand "TagIn_3" has the signal state "0".

See also

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Overview of the valid data types (Page 2236)

SET_BF: Set bit field

Description

You can use the instruction "Set bit field" to set multiple bits starting from a certain address.

You can use the input N to define the number of bits to be set. The address of the first bit to be set is defined by (<Operand>). If the value of the N input is greater than the number of bits in a selected byte, the bits of the next byte are set. The bits remain set until they are explicitly reset, for example, by another instruction.

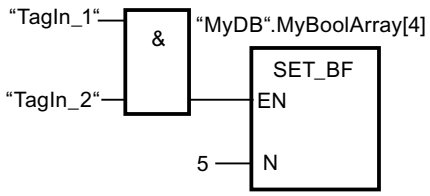
Parameters

The following table shows the parameters of the instruction "Set bit field":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
N	Input	UINT	Constant	Constant	Number of bits to be set
<Operand>	Output	BOOL	I, Q, M In the case of a DB or an IDB, an element of an ARRAY[..] of BOOL	I, Q, M In the case of a DB or an IDB, an element of an ARRAY[..] of BOOL	Pointer to the first bit to be set.

Example

The following example shows how the instruction works:



If the operands "TagIn_1" and "TagIn_2" have the signal state "1", 5 bits are set starting at the address of the operand "MyDB".MyBoolArray[4].

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

RESET_BF: Reset bit field

Description

You can use the instruction "Reset bit field" to reset several bits starting from a certain address.

You can use the value of the N input to define the number of bits to be reset. The address of the first bit to be reset is defined by (<Operand>). If the value of the input N is greater than the

number of bits in a selected byte, the bits of the next byte are reset. The bits remain reset until they are explicitly set, for example, by another instruction.

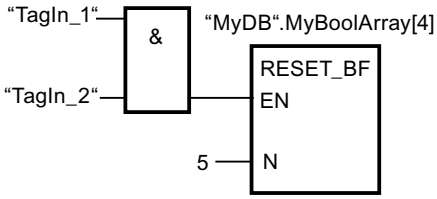
Parameters

The following table shows the parameters of the instruction "Reset bit field":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
N	Input	UINT	Constant	Constant	Number of bits to be reset.
<Operand>	Output	BOOL	I, Q, M In the case of a DB or an IDB, an element of an ARRAY[..] of BOOL	I, Q, M In the case of a DB or an IDB, an element of an ARRAY[..] of BOOL	Pointer to the first bit to be reset.

Example

The following example shows how the instruction works:



If the operands "TagIn_1" and "TagIn_2" have the signal state "1", 5 bits are reset starting at the address of the operand "MyDB".MyBoolArray[4].

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

SR: Set/reset flip-flop

Description

You can use the instruction "Set/reset flip-flop" to set or reset the bit of a specified operand based on the signal state of the inputs S and R1. If the signal state is "1" at input S and "0" at input R1, the specified operand is set to "1". If the signal state is "0" at input S and "1" at input R1, the specified operand will be reset to "0".

Input R1 takes priority over input S. When the signal state is "1" on both inputs S and R1, the signal state of the specified operand is reset to "0".

The instruction is not executed if the signal state at the two inputs S and R1 is "0". The signal state of the operand then remains unchanged.

The current signal state of the operand is transferred to output Q and can be queried there.

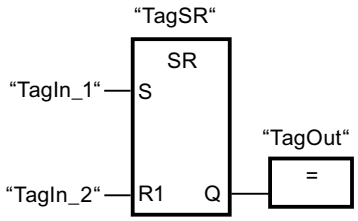
Parameters

The following table shows the parameters of the instruction "Set/reset flip-flop":

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
S	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable setting
R1	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable resetting
<Operand>	InOut	BOOL	I, Q, M, D, L	I, Q, M, D, L	Operand that is set or reset
Q	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Signal state of the operand

Example

The following example shows how the instruction works:



The operands "TagSR" and "TagOut" are set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The operand "TagIn_2" has the signal state "0".

The operands "TagSR" and "TagOut" are reset when one of the following conditions is fulfilled:

- The operand "TagIn_1" has signal state "0" and the operand "TagIn_2" has signal state "1".
- The operands "TagIn_1" and "TagIn_2" have signal state "1".

See also

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Overview of the valid data types (Page 2236)

RS: Reset/set flip-flop

Description

You can use the instruction "Reset/set flip-flop" to reset or set the bit of a specified operand based on the signal state of the inputs R and S1. If the signal state is "1" at input R and "0" at input S1, the specified operand will be reset to "0". If the signal state is "0" at input R and "1" at input S1, the specified operand is set to "1".

Input S1 takes priority over input R. When the signal state is "1" at both inputs R and S1, the signal state of the specified operand is set to "1".

The instruction is not executed if the signal state at the two inputs R and S1 is "0". The signal state of the operand then remains unchanged.

The current signal state of the operand is transferred to output Q and can be queried there.

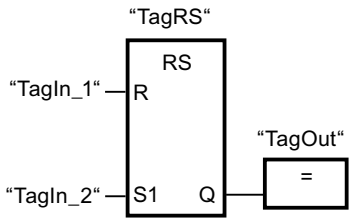
Parameters

The following table shows the parameters of the instruction "Reset/set flip-flop":

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
R	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable resetting
S1	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable setting
<Operand>	InOut	BOOL	I, Q, M, D, L	I, Q, M, D, L	Operand that is reset or set.
Q	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Signal state of the operand

Example

The following example shows how the instruction works:



The operands "TagRS" and "TagOut" are reset when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The operand "TagIn_2" has the signal state "0".

The operands "TagRS" and "TagOut" are set when the following conditions are fulfilled:

- The operand "TagIn_1" has signal state "0" and the operand "TagIn_2" has signal state "1".
- The operands "TagIn_1" and "TagIn_2" have signal state "1".

See also

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Overview of the valid data types (Page 2236)

P: Scan operand for positive signal edge

Description

You can use the instruction "Scan operand for positive signal edge" to determine whether there is a "0" to "1" change in the signal state of a specified operand (<Operand1>). The instruction compares the current signal state of <Operand1> with the signal state of the previous scan, which is saved in an edge memory bit (<Operand2>). If the instruction detects a change in the result of logic operation (RLO) from "0" to "1", there is a positive, rising edge.

If a rising edge is detected, the output of the instruction has the signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Specify the operand to be queried (<Operand1>) in the operand placeholder above the instruction. Specify the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the memory bit is overwritten. This would influence edge evaluation and the result would no longer be unequivocal. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

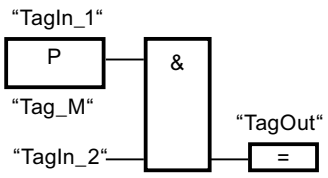
Parameters

The following table shows the parameters of the "Scan operand for positive signal edge" instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand1>	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Signal to be scanned
<Operand2>	InOut	BOOL	I, Q, M, D, L	I, Q, M, D, L	Edge memory bit in which the signal state of the previous query is saved.

Example

The following example shows how the instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- There is a rising edge at input "TagIn_1".
- The signal state of the operand "TagIn_2" is "1".

See also

Overview of the valid data types (Page 2236)

Example of detecting the direction of a conveyor belt (Page 3092)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

N: Scan operand for negative signal edge

Description

You can use the instruction "Scan operand for negative signal edge" to determine whether there is a "1" to "0" change in the signal state of a specified operand (<Operand1>). The instruction compares the current signal state of <Operand1> with the signal state of the previous scan, which is saved in an edge memory bit (<Operand2>). If the instruction detects a change in the result of logic operation (RLO) from "1" to "0", there is a negative, falling edge.

If a falling edge is detected, the output of the instruction has the signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Specify the operand to be queried (<Operand1>) in the operand placeholder above the instruction. Specify the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the memory bit is overwritten. This would influence edge evaluation and the result would no longer be unequivocal. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

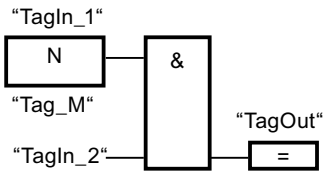
Parameters

The following table shows the parameters of the "Scan operand for negative signal edge" instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
<Operand1>	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Signal to be scanned
<Operand2>	InOut	BOOL	I, Q, M, D, L	I, Q, M, D, L	Edge memory bit in which the signal state of the previous query is saved.

Example

The following example shows how the instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- There is a falling edge at input "TagIn_1".
- The signal state of the operand "TagIn_2" is "1".

See also

Inserting FBD elements (Page 2621)

Inserting operands in FBD instructions (Page 2643)

Overview of the valid data types (Page 2236)

Editing FBD elements (Page 2635)

P=: Set operand on positive signal edge

Description

You can use the instruction "Set operand on positive signal edge" to set a specified operand (<Operand2>) when there is a "0" to "1" change in the result of logic operation (RLO). The instruction compares the current RLO with the RLO from the previous query, which is saved in the edge memory bit (<Operand1>). If the instruction detects a change in the result of logic operation (RLO) from "0" to "1", there is a positive, rising edge.

When a positive edge is detected, <Operand2> is set to signal state "1" for one program cycle. In all other cases, the operand has the signal state "0".

You specify the operand (<Operand2>) to be set in the operand placeholder above the instruction. Specify the edge memory bit (<Operand1>) in the operand placeholder below the instruction.

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the memory bit is overwritten. This would influence edge evaluation and the result would no longer be unequivocal. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

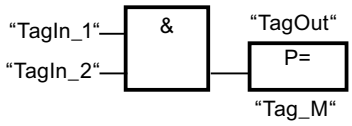
Parameters

The following table shows the parameters of the instruction "Set operand on positive signal edge":

Parameter	Declaration	Data type	Memory area	Description
<Operand2>	Output	BOOL	I, Q, M, D, L	Operand which is set when there is a positive signal edge.
<Operand1>	InOut	BOOL	I, Q, M, D, L	Edge memory bit

Example

The following example shows the parameters of the instruction:



The "TagOut" output is set for one program cycle, when the signal state at the input of the instruction box switches from "0" to "1" (positive signal edge). In all other cases, the "TagOut" output has signal state "0".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

N=: Set operand on negative signal edge

Description

You can use the instruction "Set operand on negative signal edge" to set a specified operand (<Operand1>) when there is a "1" to "0" change in the result of logic operation (RLO). The instruction compares the current RLO with the RLO from the previous query, which is saved in the edge memory bit (<Operand2>). If the instruction detects a change in the result of logic operation (RLO) from "1" to "0", there is a negative, falling edge.

When a negative edge is detected, <Operand1> is set to signal state "1" for one program cycle. In all other cases, the operand has the signal state "0".

You specify the operand (<Operand1>) to be set in the operand placeholder above the instruction. Specify the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the memory bit is overwritten. This would influence edge evaluation and the result would no longer be unequivocal. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

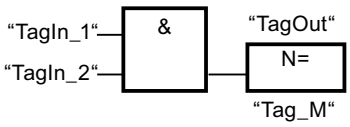
Parameters

The following table shows the parameters of the instruction "Set operand on negative signal edge":

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Output	BOOL	I, Q, M, D, L	Operand which is set when there is a negative signal edge.
<Operand2>	InOut	BOOL	I, Q, M, D, L	Edge memory bit

Example

The following example shows how the instruction works:



The operand "TagOut" is set for one program cycle if the signal state at the input of the instruction box changes from "1" to "0" (negative signal edge). In all other cases, the operand "TagOut" has the signal state "0".

See also

Inserting FBD elements (Page 2621)

Inserting operands in FBD instructions (Page 2643)

Overview of the valid data types (Page 2236)

Editing FBD elements (Page 2635)

P_TRIG: Scan RLO for positive signal edge

Description

Use the instruction "Scan RLO for positive signal edge" to query a "0" to "1" change in the signal state of the result of logic operation (RLO). The instruction compares the current signal state of the RLO with the signal state of the previous query, which is saved in an edge memory bit (<Operand>). If the instruction detects a change in the result of logic operation (RLO) from "0" to "1", there is a positive, rising edge.

If a rising edge is detected, the output of the instruction has the signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the memory bit is overwritten. This would influence edge evaluation and the result would no longer be unequivocal. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

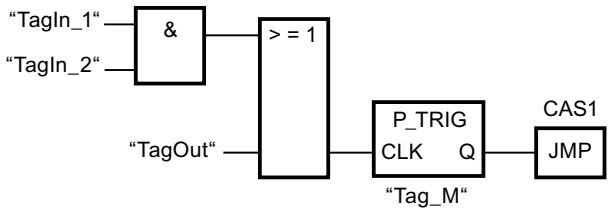
Parameters

The following table shows the parameters of the instruction "Scan RLO for positive signal edge":

Parameter	Declaration	Data type	Memory area	Description
CLK	Input	BOOL	I, Q, M, D, L	Current RLO
<Operand>	InOut	BOOL	M, D	Edge memory bit in which the RLO of the previous query is saved.
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:



The RLO of the preceding bit logic operation is saved in the edge memory bit "Tag_M". If a "0" to "1" change is detected in the signal state of the RLO, the program jumps to jump label CAS1.

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

N_TRIG: Scan RLO for negative signal edge

Description

Use the instruction "Scan RLO for negative signal edge" to query a "1" to "0" change in the signal state of the result of logic operation (RLO). The instruction compares the current signal state of the RLO with the signal state of the previous query saved in the edge memory bit (<Operand>). If the instruction detects a change in the result of logic operation (RLO) from "1" to "0", there is a negative, falling edge.

If a falling edge is detected, the output of the instruction has the signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the memory bit is overwritten. This would influence edge evaluation and the result would no longer be unequivocal. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

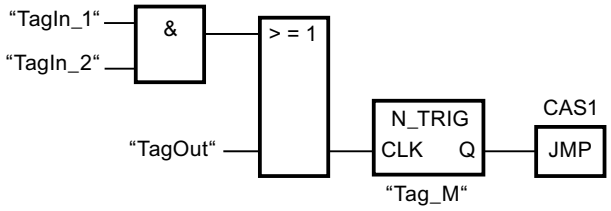
Parameters

The following table shows the parameters of the instruction "Scan RLO for negative signal edge":

Parameter	Declaration	Data type	Memory area	Description
CLK	Input	BOOL	I, Q, M, D, L	Current RLO
<Operand>	InOut	BOOL	M, D	Edge memory bit in which the RLO of the previous query is saved.
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:



The RLO of the preceding bit logic operation is saved in the edge memory bit "Tag_M". If a "1" to "0" change is detected in the signal state of the RLO, the program jumps to jump label CAS1.

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

R_TRIG: Set tag on positive signal edge

Description

You can use the "Set tag on positive signal edge" instruction to set a specified tag in the instance DB when there is a "0" to "1" change in the result of logic operation (RLO). The instruction compares the current RLO at the input CLK with the RLO from the previous query, which is saved in the specified instance DB. If the instruction detects a change in the result of logic operation (RLO) from "0" to "1", there is a positive, rising edge.

If a positive edge is detected, the tag in the instance DB is set to signal state "1" and the output Q returns the signal state "1" In all other cases, the signal state at the output of the instruction is "0".

When you insert the instruction in the program, the "Call options" dialog opens automatically. In this dialog you can specify whether the edge memory bit is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

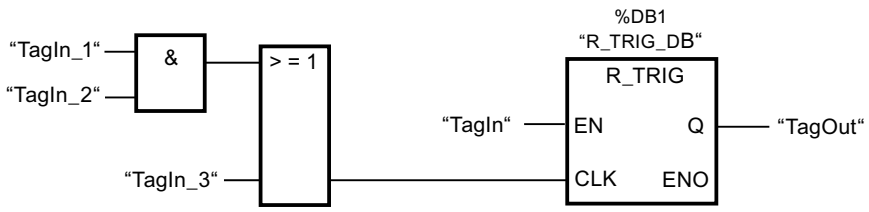
Parameters

The following table shows the parameters of the instruction "Set tag on positive signal edge":

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
CLK	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Incoming signal, the edge of which is to be queried.
Q	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:



The RLO of the preceding query is saved in the instance DB "R_TRIG_DB". If a change is the signal state of the RLO from "0" to "1" is detected in the operands "TagIn_1" and "TagIn_2" or in the operand "TagIn_3", the output "TagOut" has signal state "1".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

F_TRIG: Set tag on negative signal edge

Description

You can use the "Set tag on negative signal edge" instruction to set a specified tag in the instance DB when there is a "1" to "0" change in the result of logic operation (RLO). The instruction compares the current RLO at the input CLK with the RLO from the previous query,

which is saved in the specified instance DB. If the instruction detects a change in the result of logic operation (RLO) from "1" to "0", there is a negative, falling edge.

If a negative edge is detected, the tag in the instance DB is set to signal state "1" and the output Q returns the signal state "1" In all other cases, the signal state at the output of the instruction is "0".

When you insert the instruction in the program, the "Call options" dialog opens automatically. In this dialog you can specify whether the edge memory bit is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

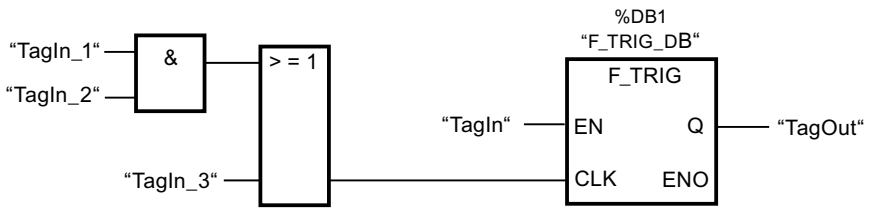
Parameters

The following table shows the parameters of the instruction "Set tag on negative signal edge":

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
CLK	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Incoming signal, the edge of which is to be queried.
Q	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:



The RLO of the preceding query is saved in the instance DB "F_TRIG_DB". If a change is the signal state of the RLO from "1" to "0" is detected in the operands "TagIn_1" and "TagIn_2" or in the operand "TagIn_3", the output "TagOut" has signal state "1".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Timer operations

IEC Timers

TP: Generate pulse

Description

You can use the instruction "Generate pulse" to set output Q for the duration PT. The instruction is started when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive signal edge). The configured time duration PT begins when the instruction starts. Output Q is set for the time duration PT, regardless of the subsequent course of the input signal (positive edge). Even when a new positive signal edge is detected, the signal state of the Q output is not affected as long as the PT duration is running.

The current time value can be queried at the output ET. The time value starts at T#0s and ends when the value of the time duration PT is reached. If the configured time duration PT is reached and the signal state at input IN is "0", the ET output is reset.

Each call of the "Generate pulse" instruction must be assigned an IEC Timer in which the instruction data is stored.

Note

If the timer is not called in the program because it is skipped, for example, output ET returns a constant value as soon as the timer has expired.

For S7-1200 CPU

An IEC Timer is a structure of the data type IEC_TIMER or TP_TIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TP_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

An IEC Timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TP_TIME or TP_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TP_TIME, TP_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC Timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when the instruction is called and also each time the outputs Q or ET are accessed.

The execution of the "Generate pulse" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

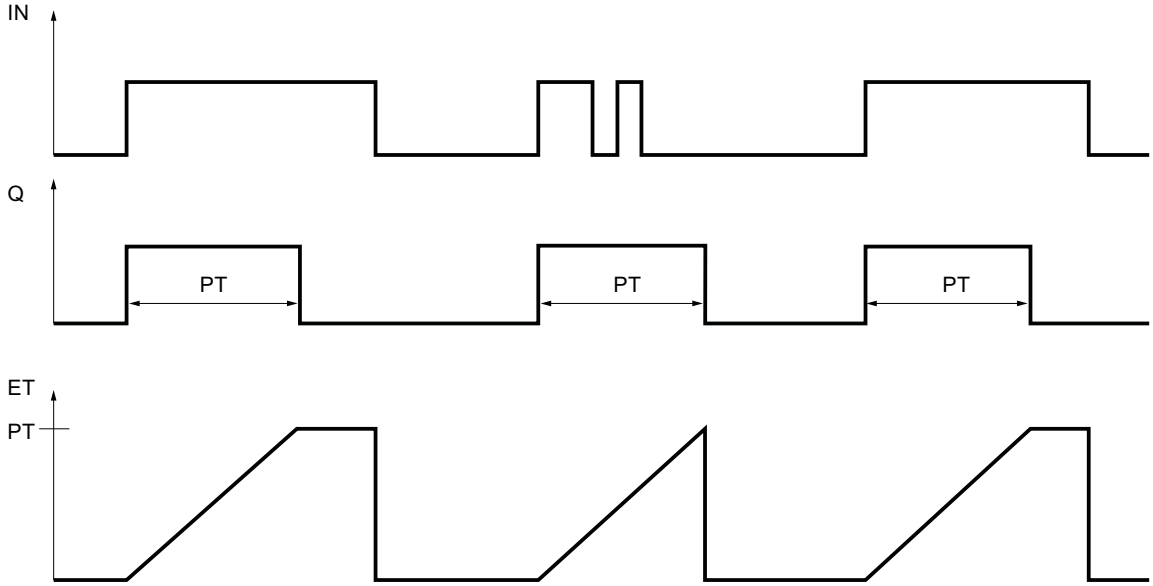
The following table shows the parameters of the "Generate pulse" instruction:

Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
IN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C, P	Start input
PT	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	I, Q, M, D, L, P or constant	Duration of the pulse. The value of the PT parameter must be positive.
Q	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, P	Pulse output
ET	Output	TIME	TIME, LTIME	I, Q, M, D, L	I, Q, M, D, L, P	Current time value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Generate pulse" instruction:



See also

Overview of the valid data types (Page 2236)

Example of controlling room temperature (Page 3097)

Call function blocks as single or multi-instances (Page 2182)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Selecting the data type of an FBD element (Page 2624)

Inserting operands in FBD instructions (Page 2643)

TON: Generate on-delay

Description

You can use the instruction "Generate on-delay" to delay setting of the Q output by the time configured with the PT time. The instruction is started when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. When the duration PT expires, output Q has the signal state "1". Output Q remains set as long as the start input is still "1". When the signal state at the start input changes from "1" to "0", the Q output is reset. The timer function is started again when a new positive signal edge is detected at the start input.

The current time value can be queried at the output ET. The time value starts at T#0s and ends when the value of the time duration PT is reached. The ET output is reset as soon as the signal state at the IN input changes to "0".

Each call of the "Generate on-delay" instruction must be assigned an IEC Timer in which the instruction data is stored.

Note

If the timer is not called in the program because it is skipped, for example, output ET returns a constant value as soon as the timer has expired.

For S7-1200 CPU

An IEC Timer is a structure of the data type IEC_TIMER or TON_TIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TON_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

An IEC Timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TON_TIME or TON_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TON_TIME, TON_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC Timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, it is saved to the "Program resources" folder in the "Program blocks > System blocks" path of the project tree. For additional information on this topic, refer to "See also".

The instruction data is updated both when the instruction is called and also each time the outputs Q or ET are accessed.

The execution of the "Generate on-delay" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

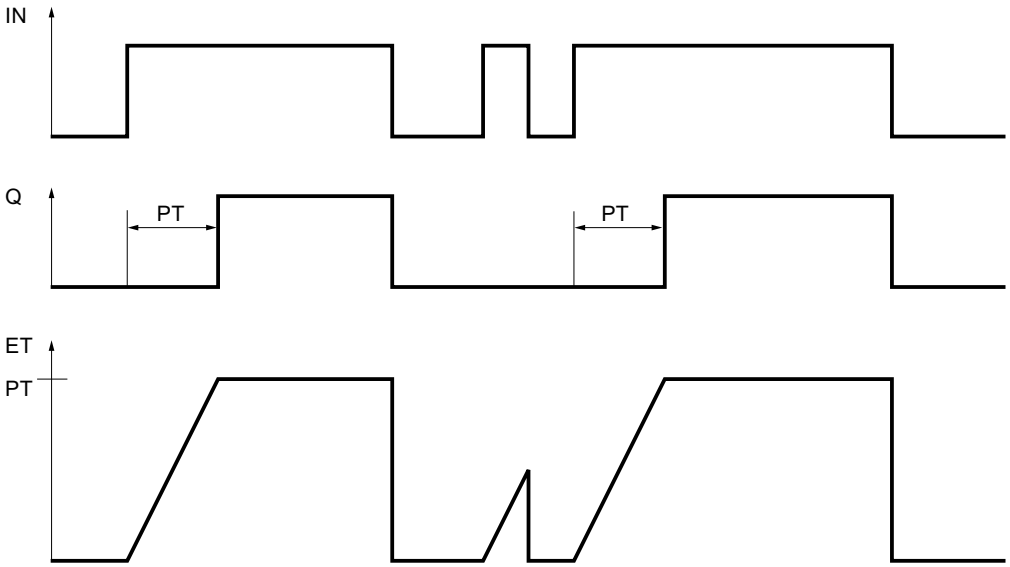
The following table shows the parameters of the "Generate on-delay" instruction:

Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
IN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C, P	Start input
PT	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	I, Q, M, D, L, P or constant	Duration of the on delay. The value of the PT parameter must be positive.
Q	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, P	Output that is set when the time PT expires.
ET	Output	TIME	TIME, LTIME	I, Q, M, D, L	I, Q, M, D, L, P	Current time value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Generate on-delay" instruction:



See also

Overview of the valid data types (Page 2236)

Call function blocks as single or multi-instances (Page 2182)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Selecting the data type of an FBD element (Page 2624)

Inserting operands in FBD instructions (Page 2643)

TOF: Generate off-delay

Description

You can use the "Generate off-delay" instruction to delay setting of the Q output by the time configured with the PT time. The output Q is set when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive signal edge). When the signal state at input IN changes back to "0" (positive signal edge), the configured time duration PT starts. Output Q remains set as long as the time duration PT is running. When the PT time duration expires, the Q output is reset. If the signal state at input IN changes to "1" before the PT time duration expires, the timer is reset. The signal state at the output Q will continue to be "1".

The current time value can be queried at the output ET. The time value starts at T#0s and ends when the value of the time duration PT is reached. When the time duration PT expires, the ET output remains set to the current value until input IN changes back to "1". If input IN changes to "1" before the time duration PT has expired, the ET output is reset to the value T#0s.

Each call of the "Generate off-delay" instruction must be assigned to an IEC Timer in which the instruction data is stored.

Note

If the timer is not called in the program because it is skipped, for example, output ET returns a constant value as soon as the timer has expired.

For S7-1200 CPU

An IEC Timer is a structure of the data type IEC_TIMER or TOF_TIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TOF_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

An IEC Timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TOF_TIME or TOF_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TOF_TIME, TOF_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC Timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when the instruction is called and also each time the outputs Q or ET are accessed.

The execution of the "Generate off-delay" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

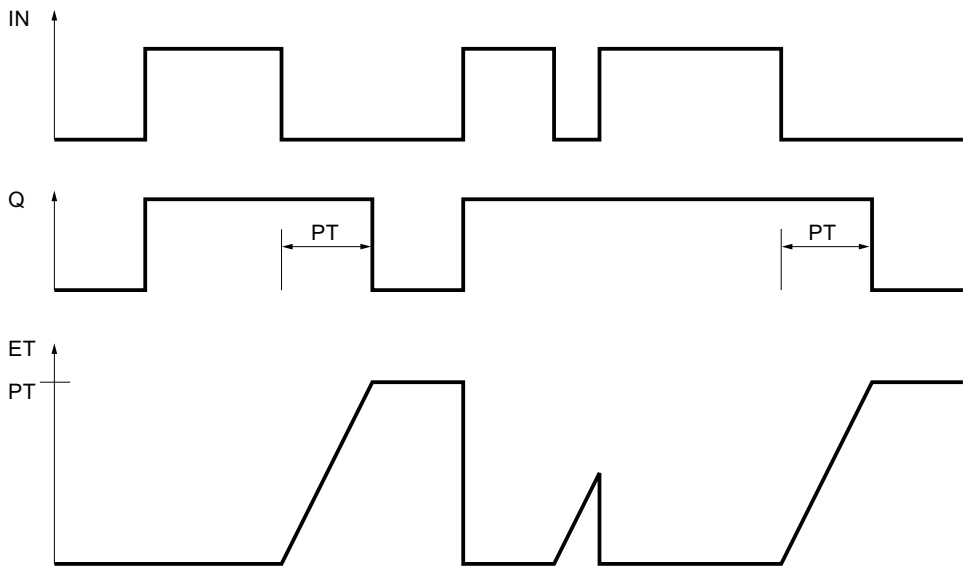
The following table shows the parameters of the "Generate off-delay" instruction:

Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
IN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C, P	Start input
PT	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	I, Q, M, D, L, P or constant	Duration of the off delay The value of the PT parameter must be positive.
Q	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, P	Output that is reset when the timer PT expires.
ET	Output	TIME	TIME, LTIME	I, Q, M, D, L	I, Q, M, D, L, P	Current time value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Generate off-delay" instruction:



See also

Overview of the valid data types (Page 2236)

Call function blocks as single or multi-instances (Page 2182)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Selecting the data type of an FBD element (Page 2624)

Inserting operands in FBD instructions (Page 2643)

TONR: Time accumulator

Description

The instruction "Time accumulator" is used to accumulate time values within a period set by the parameter PT. The instruction is executed and the configured time duration PT is started when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive edge). While the time set at PT is running, the time values are accumulated that are recorded at signal state "1" at input IN. The accumulated time is written to output ET and can be queried there. When the current time value PT is reached, the output Q has the signal state "1". Output Q remains set at "1", even when the signal state at input IN changes to "0".

The R input resets the outputs ET and Q regardless of the signal state at the start input.

Each call of the "Time accumulator" instruction must be assigned to an IEC Timer in which the instruction data is stored.

For S7-1200 CPU

An IEC Timer is a structure of the data type IEC_TIMER or TONR_TIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TONR_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

An IEC Timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TONR_TIME or TONR_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TONR_TIME, TONR_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC Timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when the instruction is called and also each time the outputs Q or ET are accessed.

The execution of the "Time accumulator" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Time accumulator" instruction:

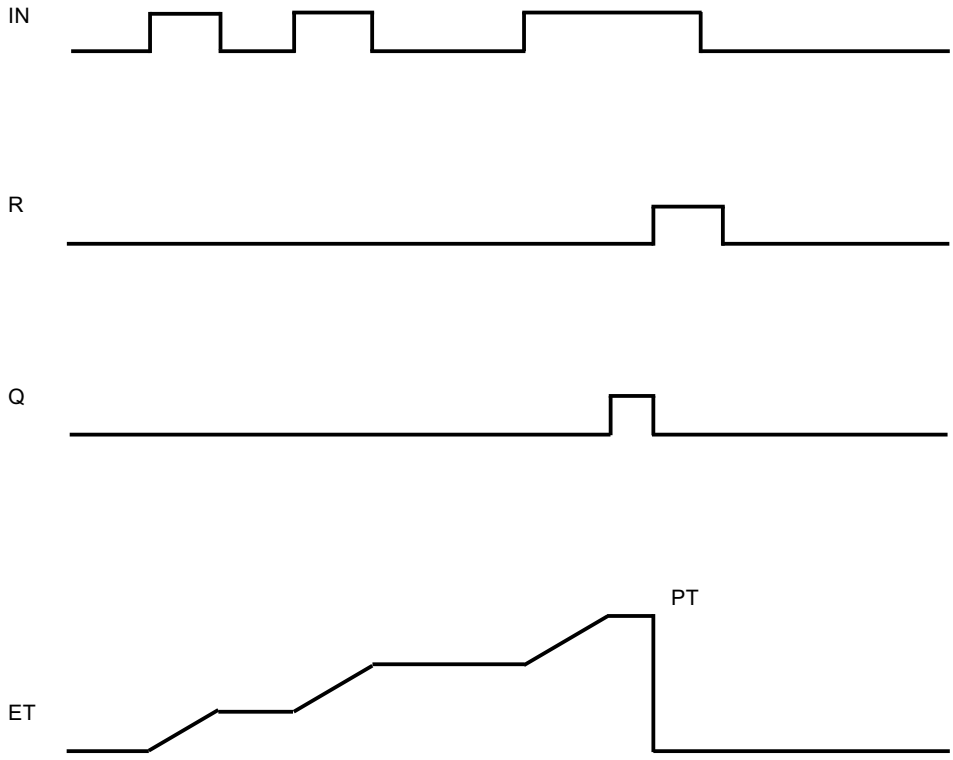
Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
IN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C, P	Start input
R	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, P or constant	Reset input
PT	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	I, Q, M, D, L, P or constant	Maximum duration of time recording. The value of the PT parameter must be positive.

Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
Q	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, P	Output that is set when the time PT expires.
ET	Output	TIME	TIME, LTIME	I, Q, M, D, L	I, Q, M, D, L, P	Current time value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Time accumulator" instruction:



See also

Overview of the valid data types (Page 2236)

Call function blocks as single or multi-instances (Page 2182)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Selecting the data type of an FBD element (Page 2624)

Inserting operands in FBD instructions (Page 2643)

TP: Start pulse timer

Description

Use the instruction "Start pulse timer" to start an IEC Timer with a specified duration as pulse. The IEC Timer is started when the result of logic operation (RLO) changes from "0" to "1" (positive signal edge). The IEC Timer runs for the specified time duration regardless of the subsequent course of the RLO. The expiry of the IEC Timer is also not affected by the detection of a new rising edge. As long as the IEC Timer is running, the querying of the timer status for "1" returns the signal state "1". When the IEC Timer has expired, the timer status returns the signal state "0".

Note

The start and the query of the IEC Timer may be on different expiry levels as each query of the outputs Q or ET updates the IEC_TIMER structure.

For S7-1200 CPU

The instruction "Start pulse timer" stores its data in a structure of the data type IEC_TIMER or TP_TIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TP_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

The instruction "Start pulse timer" stores its data in a structure of the data type IEC_TIMER, IEC_LTIMER, TP_TIME or TP_LTIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TP_TIME, TP_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated both when the instruction is called and also each time the specified timer is accessed.

The current timer status is saved in the structure components Q of the IEC Timer. You can query the timer status with the help of a binary logic operation. The query for Q or ET (e. g. "MyTimer".Q or "MyTimer".ET) updates the IEC_TIMER structure.

The execution of the instruction "Start pulse timer" assumes a preceding logic operation. It can be placed only at the end of the network.

Parameters

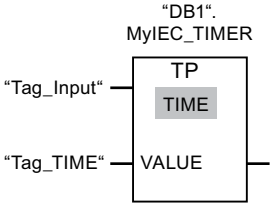
The following table shows the parameters of the instruction "Start pulse timer":

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
VALUE	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	Duration with which the IEC Timer runs.
<IEC Timer>	InOut	IEC_TIMER, TP_TIME	IEC_TIMER, IEC_LTIMER, TP_TIME, TP_LTIME	D, L	IEC Timer which is started.

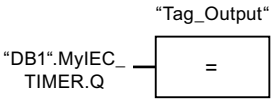
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The instruction "Start pulse timer" is executed when the signal state of the operand "Tag_Input" changes from "0" to "1". The timer "DB1".MyIEC_TIMER is started for the time stored in the operand "TagTime".



As long as the timer "DB1".MyIEC_TIMER is running, the timer status ("DB1".MyIEC_TIMER.Q) has signal state "1" and the operand "Tag_Output" is set. When the IEC Timer has expired, the signal state of the time status changes back to "0" and the "Tag_Output" operand is reset.

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

TON: Start on-delay timer

Description

Use the "Start on-delay timer" instruction to start an IEC Timer with a specified duration as on-delay. The IEC Timer is started when the result of logic operation (RLO) changes from "0" to "1" (positive signal edge). The IEC Timer runs for the specified time duration. The output returns the signal state "1" if the RLO at the input of the instruction has the signal state "1". If the RLO changes to "0" before the end of the timer, the running IEC Timer is reset. The query of the timer status for "1" returns the signal state "0". The IEC Timer restarts when the next rising signal edge is detected at the input of the instruction.

Note

The start and the query of the IEC Timer may be on different expiry levels as each query of the outputs Q or ET updates the IEC_TIMER structure.

For S7-1200 CPU

The instruction "Start on-delay timer" stores its data in a structure of the data type IEC_TIMER or TON_TIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TON_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

The "Start on-delay timer" instruction stores its data in a structure of the data type IEC_TIMER, IEC_LTIMER, TON_TIME or TON_LTIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TON_TIME, TON_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated both when the instruction is called and also each time the specified timer is accessed.

The current timer status is saved in the structure components ET of the IEC Timer. You can query the timer status with the help of a binary logic operation. The query for Q or ET (e. g. "MyTimer".Q or "MyTimer".ET) updates the IEC_TIMER structure.

The execution of the "Start on-delay timer" instruction assumes a preceding logic operation. It can be placed only at the end of the network.

Parameters

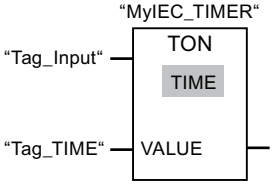
The following table shows the parameters of the instruction "Start on-delay timer":

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
VALUE	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	Duration with which the IEC Timer runs.
<IEC Timer>	InOut	IEC_TIMER, TON_TIME	IEC_TIMER, IEC_LTIMER, TON_TIME, TON_LTIME	D, L	IEC Timer which is started.

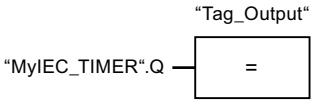
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "Start on-delay timer" instruction is executed when the signal state of the operand "Tag_Input" changes from "0" to "1". The "MyIEC_TIMER" timer is started for the time stored in the "Tag_TIME" operand.



If the timer "MyIEC_TIMER" has expired and the operand "Tag_Input" has the signal state "1", querying the timer status ("MyIEC_TIMER".Q) returns signal state "1" and the "Tag_Output" operand is set. When the signal state of the operand "Tag_Input" changes to "0", the querying of the timer status returns the signal state "0" and the operand "Tag_Output" is reset.

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

TOF: Start off-delay timer

Description

Use the "Start off-delay timer" instruction to start an IEC Timer with a specified duration as off-delay. The query of the timer status for "1" returns the signal state "0" if the result of the logic operation (RLO) at the input of the instruction has the signal state "1". When the RLO changes from "1" to "0" (negative signal edge), the IEC Timer starts with the specified time duration. The timer status remains at signal state "1" as long as the IEC Timer is running. When the timer has run out and the RLO at the input of the instruction has the signal state "0", the timer status is set to the signal state "0". If the RLO changes to "1" before the end of the timer, the running IEC Timer is reset and the timer status remain at the signal state "1".

Note

The start and the query of the IEC Timer may be on different expiry levels as each query of the outputs Q or ET updates the IEC_TIMER structure.

For S7-1200 CPU

The instruction "Start off-delay timer" stores its data in a structure of the data type IEC_TIMER or TOF_TIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TOF_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

The "Start off-delay timer" instruction stores its data in a structure of the data type IEC_TIMER, IEC_LTIMER, TOF_TIME or TOF_LTIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TOF_TIME, TOF_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated both when the instruction is called and also each time the specified timer is accessed.

The current timer status is saved in the structure components ET of the IEC Timer. You can query the timer status with the help of a binary logic operation. The query for Q or ET (e. g. "MyTimer".Q or "MyTimer".ET) updates the IEC_TIMER structure.

The execution of the "Start off-delay timer" instruction assumes a preceding logic operation. It can be placed only at the end of the network.

Parameters

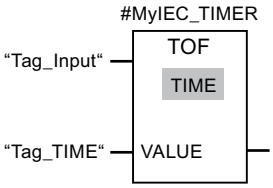
The following table shows the parameters of the instruction "Start off-delay timer":

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
VALUE	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	Duration with which the IEC Timer runs.
<IEC Timer>	InOut	IEC_TIMER, TOF_TIME	IEC_TIMER, IEC_LTIMER, TOF_TIME, TOF_LTIME	D, L	IEC Timer which is started.

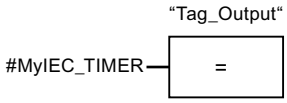
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "Start off-delay timer" instruction is executed when the signal state of the operand "Tag_Input" changes from "1" to "0". The #MyIEC_TIMER timer is started for the time stored in the operand "Tag_TIME".



As long as timer #MyIEC_TIMER is running, the query of the time status (#MyIEC_TIMER.Q) returns the signal state "1" and operand "Tag_Output" is set. If the timer has expired and the operand "Tag_Input" has the signal state "0", the query of the timer status returns the signal state "0". If the signal state of the operand "Tag_Input" changes to "1" before timer #MyIEC_TIMER expires, the timer is reset. When the signal state of the operand "Tag_Input" is "1", the query of the timer status returns the signal state "1".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

TONR: Time accumulator

Description

You can use the "Time accumulator" instruction to record how long the signal is at the input of instruction "1". The instruction is started when the result of logic operation (RLO) changes from "0" to "1" (positive signal edge). The time is recorded as long at the RLO is "1". If the RLO changes to "0", the instruction is halted. If the RLO changes back to "1", the time recording is continued. The query of the timer status for "1" returns the signal state "1" if the recorded time exceeds the value of the specified time duration and the RLO at the input of coil is "1".

The timer status and the currently expired timer can be reset to "0" using the "Reset timer" instruction.

Note

The start and the query of the IEC Timer may be on different expiry levels as each query of the outputs Q or ET updates the IEC_TIMER structure.

For S7-1200 CPU

The "Time accumulator" instruction stores its data in a structure of the data type IEC_TIMER or TONR_TIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TONR_TIME or IEC_TIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

For S7-1500 CPU

The "Time accumulator" instruction stores its data in a structure of the data type IEC_TIMER, IEC_LTIMER, TONR_TIME or TONR_LTIME. You can declare the structure as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TONR_TIME, TONR_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

The instruction data is updated both when the instruction is called and also each time the specified timer is accessed.

The current timer status is saved in the structure components ET of the IEC Timer. You can query the timer status with the help of a binary logic operation. The query for Q or ET (e. g. "MyTimer".Q or "MyTimer".ET) updates the IEC_TIMER structure.

The execution of the "Time accumulator" instruction requires a preceding logic operation. It can be placed only at the end of the network.

Parameters

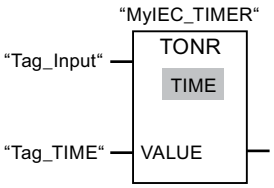
The following table shows the parameters of the "Time accumulator" instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
VALUE	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	Duration with which the IEC Timer runs.
<IEC Timer>	InOut	IEC_TIMER, TONR_TIME	IEC_TIMER, IEC_LTIMER, TONR_TIME, TONR_LTIME	D, L	IEC Timer which is started.

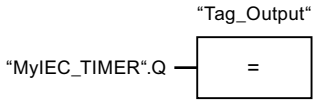
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "Time accumulator" instruction is executed if there is a positive signal edge in the RLO. The time is recorded as long as the operand "Tag_Input" has the signal state "1".



If the recorded time exceeds the value of the operand "Tag_TIME", then the query of the timer status ("MyIEC_TIMER".Q) will return the signal state "1" and the operand "Tag_Output" will be set.

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

RT: Reset timer (Page 3490)

RT: Reset timer

Description

You can use the "Reset timer" instruction to reset an IEC Timer to "0". You specify the IEC Timer to be reset by entering the name of the data block that contains the structure of the IEC Timer in the placeholder above the instruction.

The instruction is only executed if the result of logic operation (RLO) at the box input is "1". When the instruction is executed the structure components of the IEC Timer are reset to "0" in the specified data block. If the RLO at box input is "0", the instruction is not executed.

The instruction does not influence the RLO. The RLO at the box input is transferred directly to the box output.

You must assign a IEC Timer declared in the program to the "Reset timer" instruction.

The instruction data is updated only when the instruction is called and not each time the assigned IEC Timer is accessed. The query of the data is only identical from the call of the instruction to the next call of the instruction.

Parameters

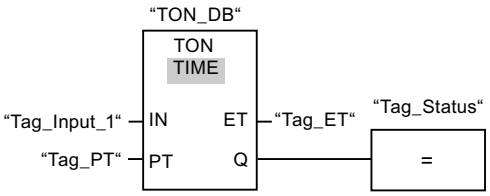
The following table shows the parameters of the instruction "Reset timer":

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
<IEC Timer>	InOut	IEC_TIMER, TP_TIME, TON_TIME, TOF_TIME, TONR_TIME	IEC_TIMER, IEC_LTIMER, TP_TIME, TP_LTIME, TON_TIME, TON_LTIME, TOF_TIME, TOF_LTIME, TONR_TIME, TONR_LTIME	D, L	IEC Timer, which is reset.

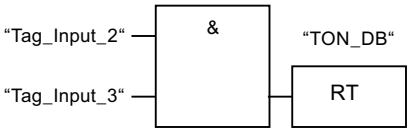
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "Generate on-delay" instruction executes when the signal state of the "Tag_Input_1" operand changes from "0" to "1". The IEC Timer stored in the instance data block "TON_DB" is started with the time duration that is specified by the operand "Tag_PT".



If the operands "Tag_Input_2" and "Tag_Input_3" have the signal state "1", the "Reset timer" instruction is executed and the IEC Timer stored in the data block "TON_DB" is reset.

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

PT: Load time duration

Description

Use the "Load time duration" instruction to set the time duration of an IEC Timer. The instruction is executed in every cycle when the result of logic operation (RLO) at the input of the instruction has the signal state "1". The instruction writes the specified time duration to the structure of the specified IEC Timer.

Note

If the specified IEC Timer is running during the execution, the instruction overwrites the current time duration of the specified IEC Timer. As a result, the timer status of the IEC Timer can change.

You must assign a IEC Timer declared in the program to the "Load time duration" instruction.

The instruction data is updated when the instruction is called and each time the assigned IEC Timer is accessed. The query for Q or ET (e. g. "MyTimer".Q or "MyTimer".ET) updates the IEC_TIMER structure.

Parameters

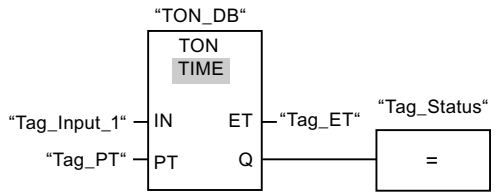
The following table shows the parameters of the instruction "Load time duration":

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
PT	Input	TIME	TIME, LTIME	I, Q, M, D, L or constant	Time duration
<IEC Timer>	InOut	IEC_TIMER, TP_TIME, TON_TIME, TOF_TIME, TONR_TIME	IEC_TIMER, IEC_LTIMER, TP_TIME, TP_LTIME, TON_TIME, TON_LTIME, TOF_TIME, TOF_LTIME, TONR_TIME, TONR_LTIME	D, L	IEC Timer, the duration of which is set.

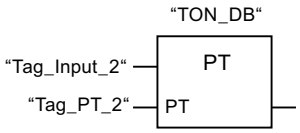
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "Generate on-delay" instruction executes when the signal state of the "Tag_Input_1" operand changes from "0" to "1". The IEC Timer stored in the instance data block "TON_DB" is started with the time duration that is specified by the operand "Tag_PT".



The "Load time duration" instruction is executed when the operand "Tag_Input_2" has the signal state "1". The instruction writes the time duration "Tag_PT_2" in the instance data block "TON_DB" and at the same time overwrites the value of the operand "Tag_PT" within the data block. As a result, the signal state of the timer status can change at the next query or upon access to "MyTimer".Q or "MyTimer".ET.

Note

The "Tag_Input_2" is executed as pulse flag in order that the time duration is loaded only throughout one program cycle.

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

SIMATIC Timers

S_PULSE: Assign pulse timer parameters and start

Description

The "Assign pulse timer parameters and start" instruction starts a programmed timer when a transition from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV) as long as the signal state at input S is "1". If the signal state at input S changes to "0" before the programmed duration expires, the timer is stopped. In this case, the signal state at output Q is "0".

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value. The current time value is output binary-coded at output BI and BCD-coded at output BCD.

If the timer is running and the signal state at input R changes to "1", the current time value and the time base are also set to zero. If the timer is not running, the signal state "1" at the R input has no effect.

The "Assign pulse timer parameters and start" instruction needs a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

The instruction data is updated with each access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

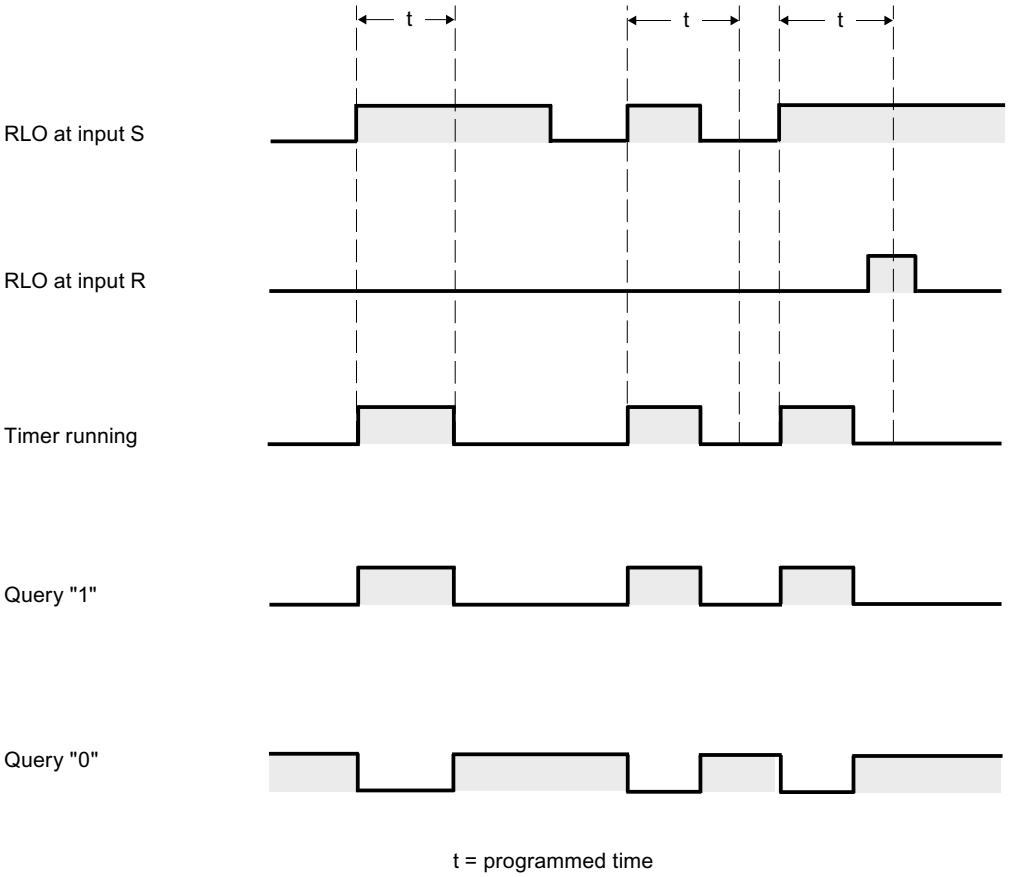
The following table shows the parameters of the "Assign pulse timer parameters and start" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
R	Input	BOOL	I, Q, M, T, C, D, L, P or constant	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (binary-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer

For additional information on valid data types, refer to "See also".

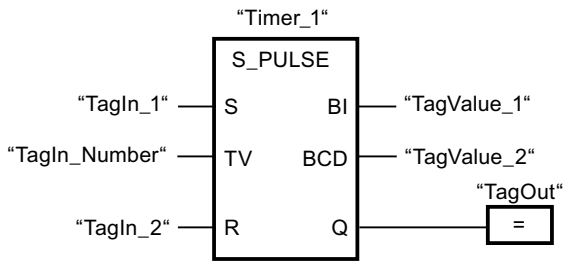
Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign pulse timer parameters and start" instruction:



Example

The following example shows how the instruction works:



"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer runs for the time value of the "TagIn_Number" operand as long as the "TagIn_1" operand has the signal state "1". If the signal state of the operand "TagIn_1" changes from "1" to "0" before the timer expires, the timer "Timer_1" is stopped. The "TagOut" operand is reset to signal state "0".

The "TagOut" operand has the signal state "1" as long as the timer is running and the "TagIn_1" operand has the signal state "1". When the time has expired or is reset, the "TagOut" operand is reset to "0".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

L: Load timer value (Page 3929)

S_PEXT: Assign extended pulse timer parameters and start

Description

The "Assign extended pulse timer parameters and start" instruction starts a programmed timer when a transition from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV), even if the signal state at input S changes to "0". As long as the timer is running, output Q has the signal state "1". When the timer has expired, output Q is reset to "0". If the signal state at input S changes from "0" to "1" while the timer is running, the timer is restarted with the duration programmed at input TV.

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value. The current time value is output binary-coded at output BI and BCD-coded at output BCD.

If the timer is running and the signal state at input R changes to "1", the current time value and the time base are also set to zero. If the timer is not running, the signal state "1" at the R input has no effect.

The "Assign extended pulse timer parameters and start" instruction needs a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

The instruction data is updated with each access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

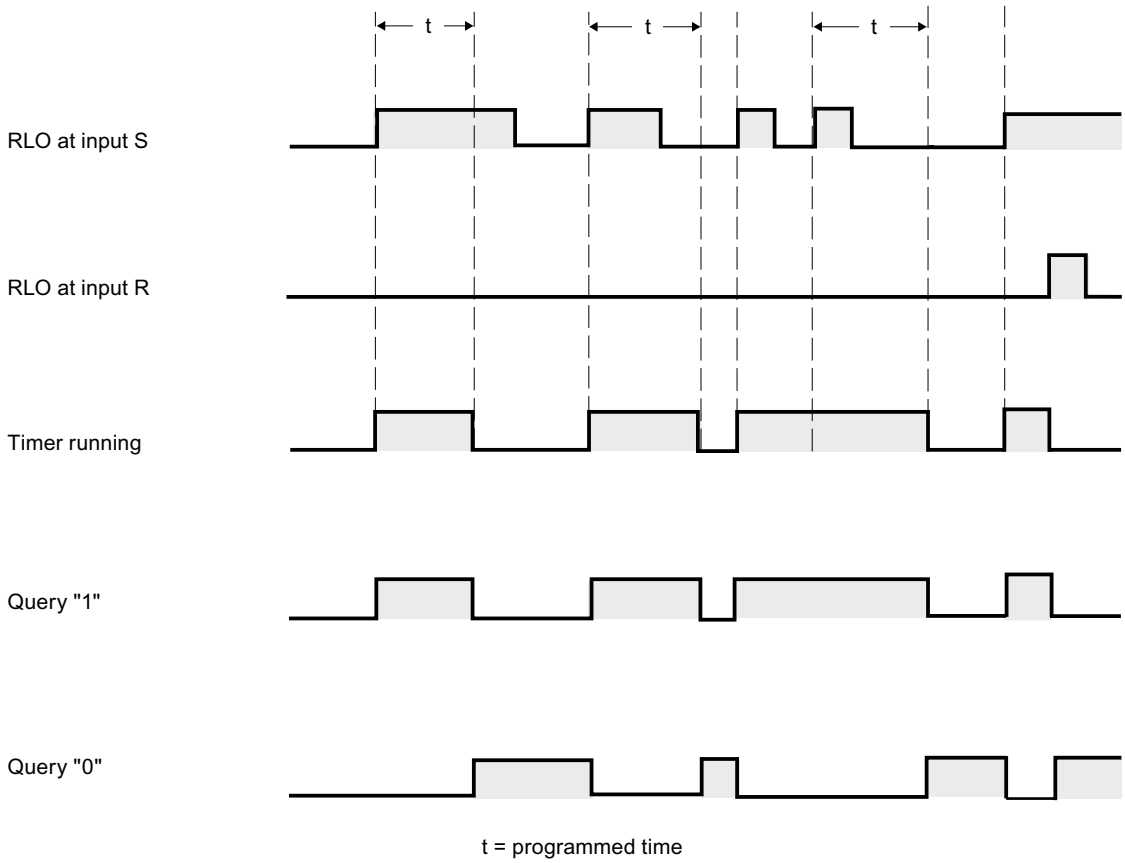
The following table shows the parameters of the "Assign extended pulse timer parameters and start" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
R	Input	BOOL	I, Q, M, T, C, D, L, P or constant	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (binary-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer

For additional information on valid data types, refer to "See also".

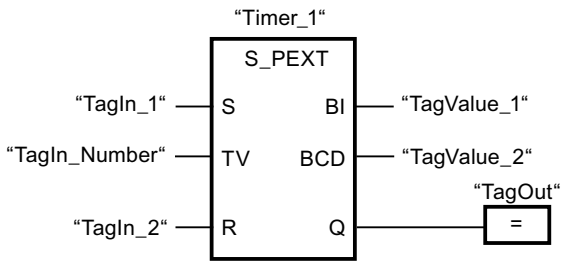
Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign extended pulse timer parameters and start" instruction:



Example

The following example shows how the instruction works:



"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer runs for the time value of the "TagIn_Number" operand without being affected by a negative edge at the S input. If the signal state at the "TagIn_1" operand changes from "0" to "1" before the timer expires, the timer is restarted.

The "TagOut" operand has the signal state "1" as long as the timer is running. When the time has expired or is reset, the "TagOut" operand is reset to "0".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

L: Load timer value (Page 3929)

S_ODT: Assign on-delay timer parameters and start

Description

The "Assign on-delay timer parameters and start" instruction starts a programmed timer when a transition from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV) as long as the signal state at input S is "1". If the timer expires correctly and input S still has signal state "1", output Q returns signal state "1". If the signal state at input S changes from "1" to "0" while the timer is running, the timer is stopped. In this case, output Q is reset to signal state "0".

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value. The current time value is output binary-coded at output BI and BCD-coded at output BCD.

If the time is running and the signal state at input R changes from "0" to "1", the current time value and the time base are also set to zero. In this case, the signal state at output Q is "0". The timer is reset if the signal state is "1" at the R input even if the timer is not running and the RLO at input S is "1".

The "Assign on-delay timer parameters and start" instruction needs a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

The instruction data is updated with each access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

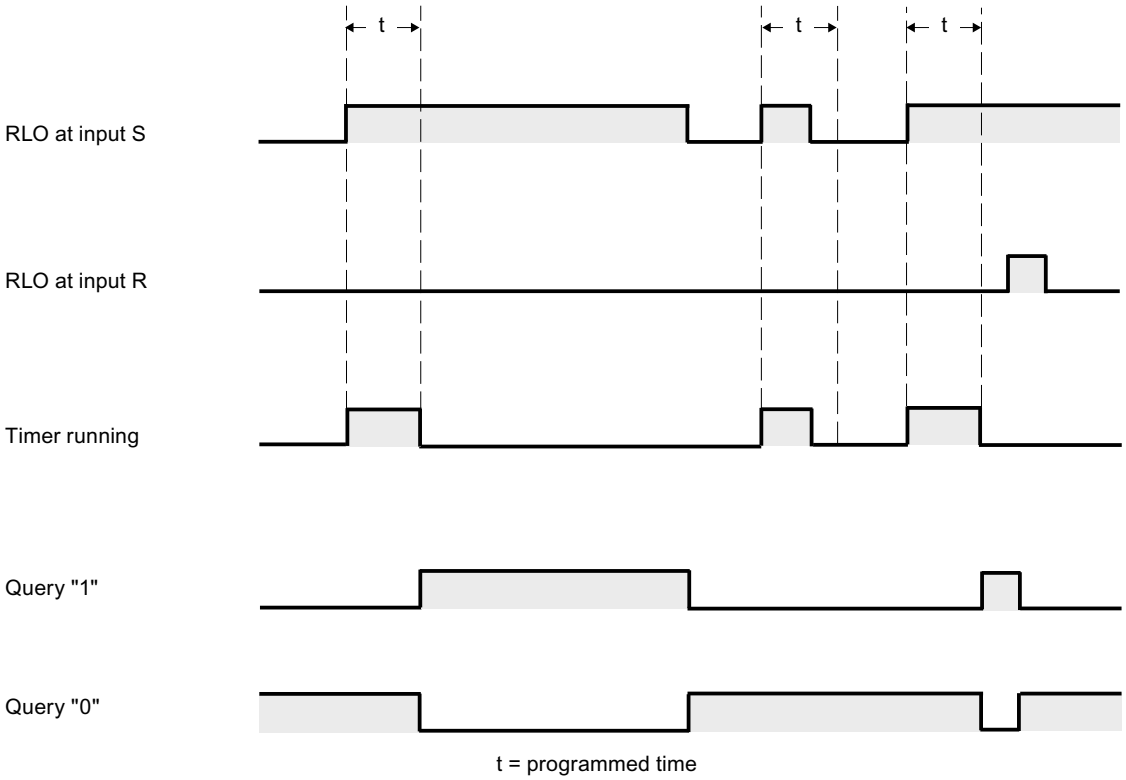
The following table shows the parameters of the "Assign on-delay timer parameters and start" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
R	Input	BOOL	I, Q, M, T, C, D, L, P or constant	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (binary-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer

For additional information on valid data types, refer to "See also".

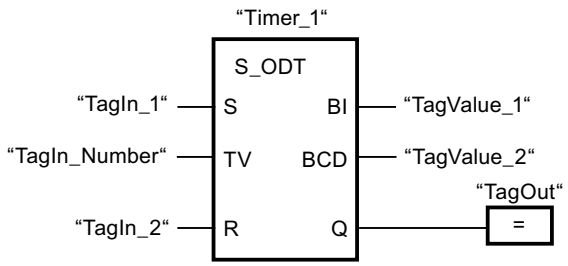
Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign on-delay timer parameters and start" instruction:



Example

The following example shows how the instruction works:



"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer expires with the value of operand "TagIn_Number". If the timer expires and the signal state of the operand is "1", the "TagOut" operand is set to "1". If the signal state at the "TagIn_1" operand changes from "1" to "0" before the timer expires, the timer is stopped. The "TagOut" operand has the signal state "0".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

L: Load timer value (Page 3929)

S_ODTS: Assign retentive on-delay timer parameters and start

Description

The "Assign retentive on-delay timer parameters and start" instruction starts a programmed timer when a transition from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV), even if the signal state at input S changes to "0". If the timer expires, the "Q" output returns signal state "1" regardless of the signal state at input "S". If the signal state at input S changes from "0" to "1" while the timer is running, the timer is restarted with the duration programmed at input (TV).

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value. The current time value is output binary-coded at output BI and BCD-coded at output BCD.

Signal state "1" at input R resets the current time value and time base to "0" regardless of the signal state at start input S. In this case, the signal state at output Q is "0".

The "Assign retentive on-delay timer parameters and start" instruction needs a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

The instruction data is updated with each access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

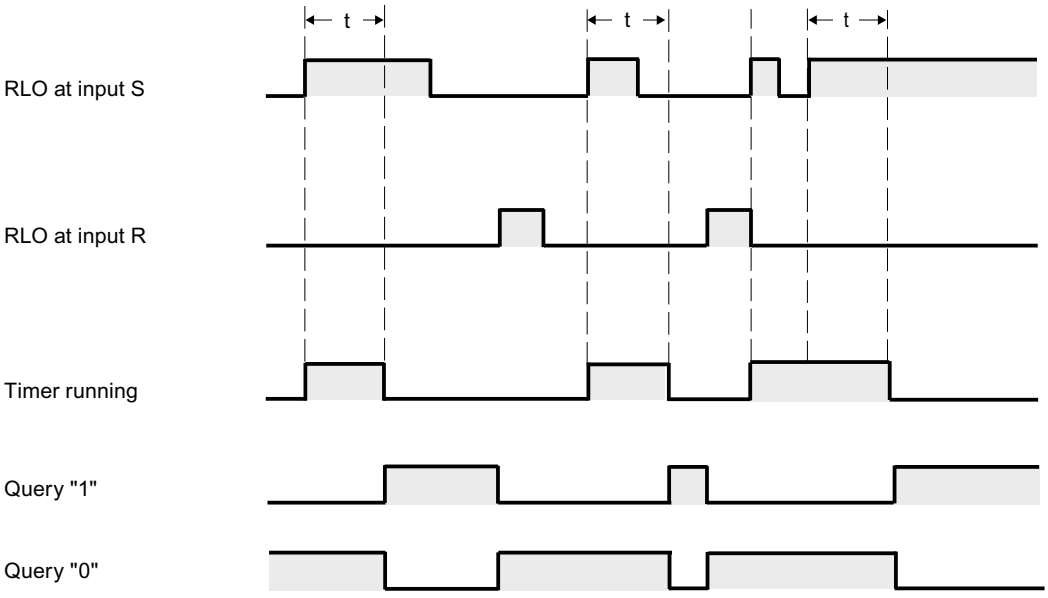
The following table shows the parameters of the "Assign retentive on-delay timer parameters and start" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
R	Input	BOOL	I, Q, M, T, C, D, L, P or constant	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (binary-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer

For additional information on valid data types, refer to "See also".

Pulse timing diagram

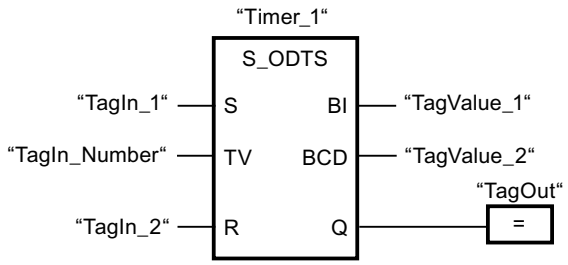
The following figure shows the pulse timing diagram of the "Assign retentive on-delay timer parameters and start" instruction:



t = programmed time

Example

The following example shows how the instruction works:



"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer expires with the time value of the "TagIn_Number" operand, even when the signal state of the "TagIn_1" operand changes to "0". When the time expires, the "TagOut" operand is reset to "1". If the signal state at the "TagIn_1" operand changes from "0" to "1" while the timer is running, the timer is restarted.

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

L: Load timer value (Page 3929)

S_OFFDT: Assign off-delay timer parameters and start

Description

The "Assign off-delay timer parameters and start" instruction starts a programmed timer when a transition from "1" to "0" (negative signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV). As long as the timer is running or input S returns signal state "1", output Q has signal state "1". If the timer expires and the signal state is "0", output Q is reset to signal state "0". If the signal state at input S changes from "0" to "1" while the timer is running, the timer is stopped. The timer is only restarted after a falling signal edge is detected at input S.

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value. The current time value is output binary-coded at output BI and BCD-coded at output BCD.

Signal state "1" at input R resets the current time value and time base to "0". In this case, the signal state at output Q is "0".

The "Assign off-delay timer parameters and start" instruction needs a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

The instruction data is updated with each access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

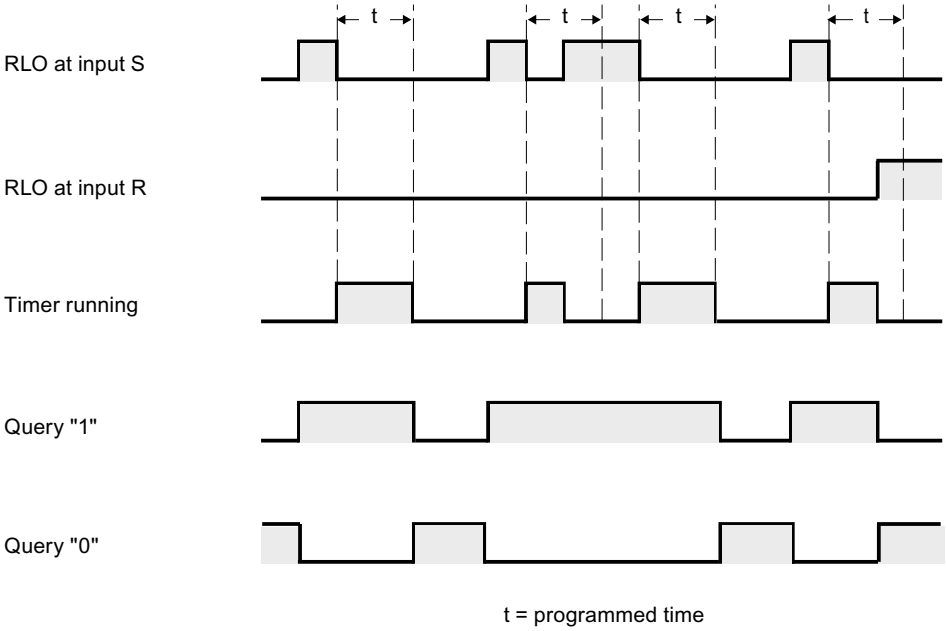
The following table shows the parameters of the "Assign off-delay timer parameters and start" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
R	Input	BOOL	I, Q, M, T, C, D, L, P or constant	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (binary-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer

For additional information on valid data types, refer to "See also".

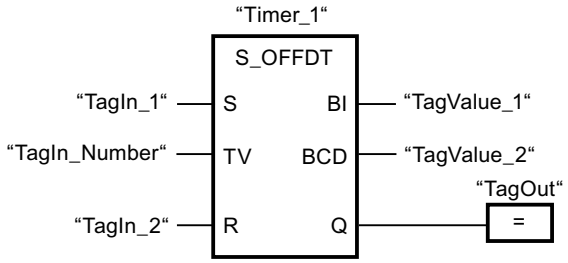
Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign off-delay timer parameters and start" instruction:



Example

The following example shows how the instruction works:



"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "1" to "0". The timer expires with the value of operand "TagIn_Number". The "TagOut" operand is set to "1" if the timer is running or the "TagIn_1" operand has the signal state "0". If the signal state at the "TagIn_1" operand changes from "0" to "1" while the timer is running, the timer is reset.

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

L: Load timer value (Page 3929)

SP: Start pulse timer

Description

The instruction "Start pulse timer" starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at the start input. The timer runs with the specified duration as long as the RLO has the signal state "1". As long as the timer is running, the query of timer status "1" returns the signal state "1". If there is a change from "1" to "0" in the RLO before the time value has elapsed, the timer stops. In this case, the query for timer status for "1" returns the signal state "0".

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value.

The "Start pulse timer" instruction needs a preceding logic operation for the edge evaluation and can only be placed on the right edge of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

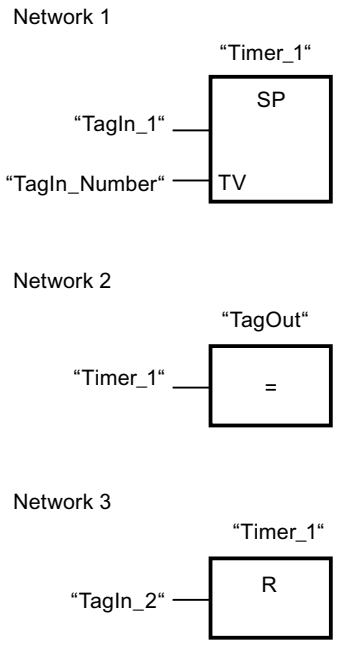
The following table shows the parameters of the instruction "Start pulse timer":

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
<Timer>	InOut/Input	TIMER	T	Timer which is started. The number of timers depends on the CPU.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer expires with the time value of the "TagIn_Number" operand as long as the signal state of the "TagIn_1" operand is "1". If the signal state at the "TagIn_1" operand changes from "1" to "0" before the timer expires, the timer is stopped. As long as the timer is running, the "TagOut" operand returns signal state "1". If the signal state of the "TagIn_1" operand changes from "0" to "1", the timer is reset, i.e. the timer is stopped and the current time value is set to "0".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

L: Load timer value (Page 3929)

SE: Start extended pulse timer

Description

The "Start extended pulse timer" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at the start input. The timer runs for the specified time period even when the RLO changes to signal state "0". As long as the timer is running, the query of timer status "1" returns the signal state "1". If the RLO changes from "0" to "1" while the timer is running, the timer is restarted with the

programmed time period. When the timer expires, the query for timer status for "1" returns the signal state "0".

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value.

The "Start extended pulse timer" instruction needs a preceding logic operation for the edge evaluation and can only be placed on the right edge of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

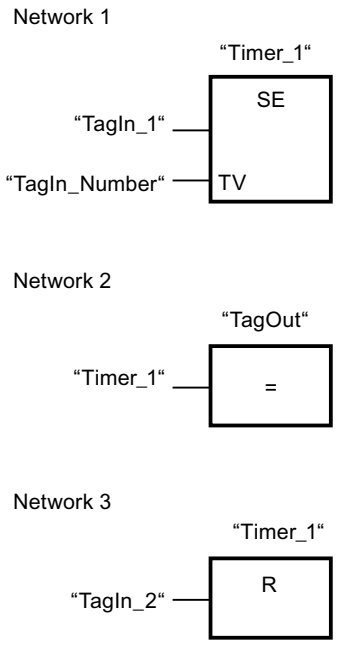
The following table shows the parameters of the instruction "Start extended pulse timer":

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
<Timer>	InOut/Input	TIMER	T	Timer which is started. The number of timers depends on the CPU.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer expires with the time value of the "TagIn_Number" operand without being affected by a negative edge at the RLO. As long as the timer is running, the "TagOut" operand returns signal state "1". If the signal state at the "TagIn_1" operand changes from "0" to "1" before the timer expires, the timer is restarted.

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

L: Load timer value (Page 3929)

SD: Start on-delay timer

Description

The "Start on-delay timer" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at the start input. The timer runs for the specified period of time as long as the RLO is "1". If the timer expires and the RLO has the signal state "1", the query timer status "1" returns the signal state "1". If the RLO changes from "1" to "0" while the timer is running, the timer is stopped. In this case, querying the timer status for "1" returns the signal state "0".

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value.

The "Start on-delay timer" instruction needs a preceding logic operation for the edge evaluation and can only be placed on the right edge of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

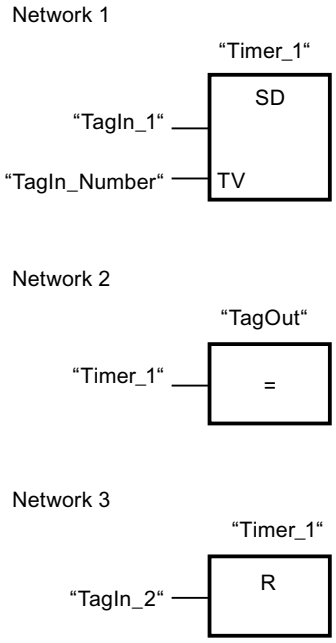
The following table shows the parameters of the instruction "Start on-delay timer":

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
<Timer>	InOut/Input	TIMER	T	Timer which is started. The number of timers depends on the CPU.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer expires with the value of operand "TagIn_Number". If the timer expires and the RLO has the signal state "1", the "TagOut" operand is set to "1". If the signal state at the "TagIn_1" operand changes from "1" to "0" before the timer expires, the timer is stopped. If the signal state of the "TagIn_2" operand changes to "1", "Timer_1" is reset, i.e. the timer is stopped and the current time value is set to "0".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

L: Load timer value (Page 3929)

SS: Start retentive on-delay timer

Description

The "Start retentive on-delay timer" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at the start input. The timer runs for the specified time period even when the RLO changes to signal state "0". When the timer expires, the query for timer status for "1" returns the signal state "1". When the timer expires, the timer can only be restarted if it is explicitly reset.

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value.

The "Start retentive on-delay timer" instruction needs a preceding logic operation for the edge evaluation and can only be placed on the right edge of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

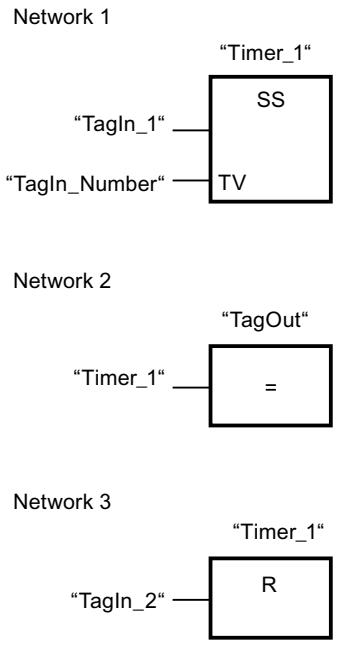
The following table shows the parameters of the instruction "Start retentive on-delay timer":

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
<Timer>	InOut/Input	TIMER	T	Timer which is started. The number of timers depends on the CPU.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer expires with the value of operand "TagIn_Number". When the time expires, the "TagOut" operand is reset to "1". If the signal state at the "TagIn_1" operand changes from "0" to "1" while the timer is running, the timer is restarted. If the signal state of the "TagIn_2" operand changes to "1", "Timer_1" is reset, i.e. the timer is stopped and the current time value is set to "0".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

L: Load timer value (Page 3929)

SF: Start off-delay timer

Description

The "Start off-delay timer" instruction starts a programmed timer when a change from "1" to "0" (negative signal edge) is detected in the result of logic operation (RLO) at the start input. The timer runs for the specified time period. As long as the timer is running, the query of timer status "1" returns the signal state "1". If the RLO changes from "0" to "1" while the timer is

running, the timer is reset. The timer is always restarted when the RLO changes from "1" to "0".

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value.

The "Start off-delay timer" instruction needs a preceding logic operation for the edge evaluation and can only be placed on the right edge of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

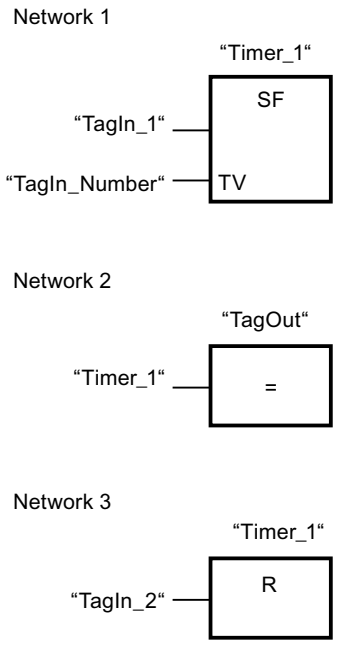
The following table shows the parameters of the instruction "Start off-delay timer":

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
<Timer>	InOut/Input	TIMER	T	Timer which is started. The number of timers depends on the CPU.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "1" to "0". The timer expires with the value of operand "TagIn_Number". As long as the timer is running, the "TagOut" operand is set to "1". If the signal state at the "TagIn_1" operand changes from "1" to "0" while the timer is running, the timer is restarted. If the signal state of the "TagIn_2" operand changes to "1", "Timer_1" is reset, i.e. the timer is stopped and the current time value is set to "0".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

L: Load timer value (Page 3929)

Counter operations

IEC Counters

CTU: Count up

Description

You can use the "Count up" instruction to increment the value at output CV. When the signal state at the CU input changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value at the CV output is incremented by one. When the instruction executes for the first time, the current counter value at the CV output is set to zero. The counter is incremented each time a positive signal edge is detected, until it reaches the high limit for the specified data type at output CV. When the high limit is reached, the signal state at the CU input no longer has an effect on the instruction.

You can scan the counter status at the Q output. The signal state at the Q output is determined by the PV parameter. If the current counter value is greater than or equal to the value of the PV parameter, the Q output is set to signal state "1". In all other cases, the Q output has signal state "0". You can also specify a constant for the PV parameter.

The value at the CV output is reset to "0" and saved to an edge memory bit when the signal state at input R changes to "1". As long as the R input has the signal state "1", the signal state at the CU input has no effect on the instruction.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count up" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

For S7-1200 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER	• CTU_SINT / CTU_USINT • CTU_INT / CTU_UINT • CTU_DINT / CTU_UDINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER

For S7-1500 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER	• CTU_SINT / CTU_USINT • CTU_INT / CTU_UINT • CTU_DINT / CTU_UDINT • CTU_LINT / CTU_ULINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER")
- Declaration as a local tag of the type CTU_<Data type> or IEC_<Counter> in the "Static" section of a block (for example #MyIEC_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, it is saved to the "Program resources" folder in the "Program blocks > System blocks" path of the project tree. For additional information on this topic, refer to "See also".

When you set up the IEC counter in a separate data block (single instance), the instance data block is created by default with "optimized block access" and the individual tags are defined as retentive. For additional information on setting retentivity in an instance data block, refer to "See also".

When you set up the IEC counter as local tag (multiple instance) in a function block with "optimized block access", it is defined as retentive in the block interface.

The execution of the "Count up" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Count up" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
CU	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Count input
R	Input	BOOL	I, Q, M, D, L, P or constant	I, Q, M, T, C, D, L, P or constant	Reset input
PV	Input	Integers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Value at which the output Q is set.

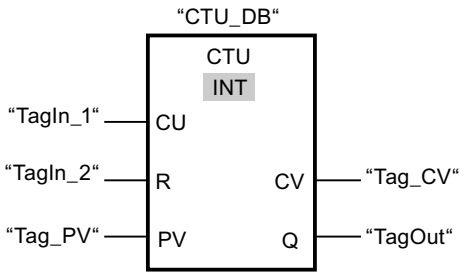
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
Q	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Counter status
CV	Output	Integers, CHAR, DATE	I, Q, M, D, L, P	I, Q, M, D, L, P	Current counter value

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



When the signal state of the "TagIn_1" operand changes from "0" to "1", the "Count up" instruction executes and the current counter value of the "Tag_CV" operand is incremented by one. With each additional positive signal edge, the counter value is incremented until the high limit of the specified data type (INT = 32767) is reached.

The value of the PV parameter is adopted as the limit for determining the "TagOut" output. The "TagOut" output has signal state "1" as long as the current counter value is greater than or equal to the value of the "Tag_PV" operand. In all other cases, the "TagOut" output returns the signal state "0".

See also

Overview of the valid data types (Page 2236)

Call function blocks as single or multi-instances (Page 2182)

Setting retentivity in an instance data block (Page 2839)

Setting the retentivity of local tags (Page 2561)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Selecting the data type of an FBD element (Page 2624)

CTD: Count down

Description

You can use the "Count down" instruction to decrement the value at output CV. When the signal state at the CD input changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value at the CV output is decremented by one. When the instruction is executed the first time, the counter value of the CV parameter is set to the value of the PV parameter. Each time a positive signal edge is detected, the counter value is decremented until it reaches the low limit of the specified data type. When the low limit is reached, the signal state at the CD input no longer has an effect on the instruction.

You can scan the counter status at the Q output. If the current counter value is less than or equal to "0", the Q output is set to signal state "1". In all other cases, the Q output has signal state "0". You can also specify a constant for the PV parameter.

The value at the CV output is set to the value of the PV parameter and saved to a edge memory bit when the signal state at the LD input changes from "0" to "1". As long as the LD input has the signal state "1", the signal state at the CD input has no effect on the instruction.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count down" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

For S7-1200 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER	• CTD_SINT / CTD_USINT • CTD_INT / CTD_UINT • CTD_DINT / CTD_UDINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER

For S7-1500 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER	• CTD_SINT / CTD_USINT • CTD_INT / CTD_UINT • CTD_DINT / CTD_UDINT • CTD_LINT / CTD_ULINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER")
- Declaration as a local tag of the type CTD_<Data type> or IEC_<Counter> in the "Static" section of a block (for example #MyIEC_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, it is saved to the "Program resources" folder in the "Program blocks > System blocks" path of the project tree. For additional information on this topic, refer to "See also".

When you set up the IEC counter in a separate data block (single instance), the instance data block is created by default with "optimized block access" and the individual tags are defined as retentive. For additional information on setting retentivity in an instance data block, refer to "See also".

When you set up the IEC counter as local tag (multiple instance) in a function block with "optimized block access", it is defined as retentive in the block interface.

The execution of the "Count down" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Count down" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
CD	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Count input
LD	Input	BOOL	I, Q, M, D, L, P or constant	I, Q, M, T, C, D, L, P or constant	Load input
PV	Input	Integers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Value at which the output Q is set.

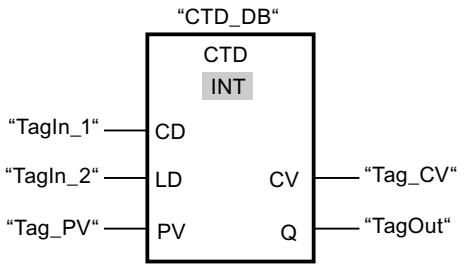
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
Q	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Counter status
CV	Output	Integers, CHAR, DATE	I, Q, M, D, L, P	I, Q, M, D, L, P	Current counter value

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



When the signal state of the "TagIn_1" operand changes from "0" to "1", the "Count down" instruction executes and the value at the "Tag_CV" output is decremented by one. With each additional positive signal edge, the counter value is decremented until the low limit of the specified data type (INT = -32768) is reached.

The value of the PV parameter is adopted as the limit for determining the "TagOut" output. The "TagOut" output has signal state "1" as long as the current counter value is less than or equal to "0". In all other cases, the "TagOut" output returns the signal state "0".

See also

Overview of the valid data types (Page 2236)

Call function blocks as single or multi-instances (Page 2182)

Setting retentivity in an instance data block (Page 2839)

Setting the retentivity of local tags (Page 2561)

Editing FBD elements (Page 2635)

Inserting FBD elements (Page 2621)

Inserting operands in FBD instructions (Page 2643)

Selecting the data type of an FBD element (Page 2624)

CTUD: Count up and down

Description

You can use the "Count up and down" instruction to increment and decrement the counter value at the CV output. If the signal state at the CU input changes from "0" to "1" (positive signal edge), the current counter value is incremented by one and stored at the CV output. If the signal state at the CD input changes from "0" to "1" (positive signal edge), the current counter value at the CV output is decremented by one. If there is a positive signal edge at the CU and CD inputs in one program cycle, the current counter value at the CV output remains unchanged.

The counter value can be incremented until it reaches the high limit of the data type specified at the CV output. When the high limit is reached, the counter value is no longer incremented on a positive signal edge. When the low limit of the specified data type is reached, the counter value is not decremented any further.

When the signal state at the LD input changes to "1", the counter value at the CV output is set to the value of the PV parameter and stored in a edge memory bit. As long as the LD input has the signal state "1", the signal state at the CU and CD inputs has no effect on the instruction.

The counter value is set to "0" and stored in an edge memory bit when the signal state at input R changes to "1". As long as the R input has the signal state "1", a change in the signal state of the CU, CD and LD inputs has no effect on the "Count up and down" instruction.

You can scan the current status of the up counter at the QU output. If the current counter value is greater than or equal to the value of the PV parameter, the QU output is set to signal state "1". In all other cases, the QU output has signal state "0". You can also specify a constant for the PV parameter.

You can scan the current status of the down counter at the QD output. If the current counter value is less than or equal to zero, the QD output is set to signal state "1". In all other cases, the QD output has signal state "0".

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count up and down" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

For S7-1200 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER	• CTUD_SINT / CTUD_USINT • CTUD_INT / CTUD_UINT • CTUD_DINT / CTUD_UDINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER

For S7-1500 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER	• CTUD_SINT / CTUD_USINT • CTUD_INT / CTUD_UINT • CTUD_DINT / CTUD_UDINT • CTUD_LINT / CTUD_ULINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER")
- Declaration as a local tag of the type CTUD_<Data type> or IEC_<Counter> in the "Static" section of a block (for example #MyIEC_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, it is saved to the "Program resources" folder in the "Program blocks > System blocks" path of the project tree. For additional information on this topic, refer to "See also".

When you set up the IEC counter in a separate data block (single instance), the instance data block is created by default with "optimized block access" and the individual tags are defined as retentive. For additional information on setting retentivity in an instance data block, refer to "See also".

When you set up the IEC counter as local tag (multiple instance) in a function block with "optimized block access", it is defined as retentive in the block interface.

The execution of the "Count up and down" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

The following table shows the parameters of the "Count up and down" instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
CU	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Count up input
CD	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Count down input
R	Input	BOOL	I, Q, M, D, L, P or constant	I, Q, M, T, C, D, L, P or constant	Reset input
LD	Input	BOOL	I, Q, M, D, L, P or constant	I, Q, M, T, C, D, L, P or constant	Load input

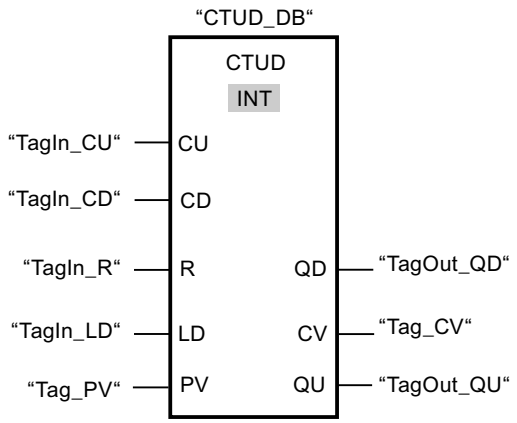
Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
PV	Input	Integers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Value at which the output QU is set.
QU	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Up-counter status
QD	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Down-counter status
CV	Output	Integers, CHAR, DATE	I, Q, M, D, L, P	I, Q, M, D, L, P	Current counter value

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the signal state at the "TagIn_CU" or "TagIn_CD" input changes from "0" to "1" (positive signal edge), the "Count up and down" instruction is executed. When there is a positive signal edge at the "TagIn_CU" input, the current counter value is incremented by one and stored at the "Tag_CV" output. When there is a positive signal edge at the "TagIn_CD" input, the counter value is decremented by one and stored at the "Tag_CV" output. When there is a positive edge at the CU input, the counter value is incremented until it reaches the high limit (INT = 32767). If input CD has a positive signal edge, the counter value is decremented until it reaches the low limit (INT = -32768).

The "TagOut_GU" output has signal state "1" as long as the current counter value is greater than or equal to the value at the "Tag_PV" input. In all other cases, the "TagOut_QU" output returns the signal state "0".

The "TagOut_QD" output has signal state "1" as long as the current counter value is less than or equal to "0". In all other cases, the "TagOut_QD" output has signal state "0".

See also

Overview of the valid data types (Page 2236)
Example of detecting the fill level of a storage area (Page 3093)
Call function blocks as single or multi-instances (Page 2182)
Setting retentivity in an instance data block (Page 2839)
Setting the retentivity of local tags (Page 2561)
Inserting FBD elements (Page 2621)
Editing FBD elements (Page 2635)
Inserting operands in FBD instructions (Page 2643)
Selecting the data type of an FBD element (Page 2624)

SIMATIC counters

S_CUD: Assign parameters and count up / down

Description

You can use the "Assign parameters and count up / down" instruction to increment or decrement the value of a counter. If the signal state at the CU input changes from "0" to "1" (positive signal edge), the current counter value is incremented by one. If the signal state at the CD input changes from "0" to "1" (positive signal edge), the counter value is decremented by one. The current counter value is output as a hexadecimal value at output CV and BCD-coded at output CV_BCD. If there is a positive signal edge at the CU and CD inputs in one program cycle, the counter value remains unchanged.

The counter value is incremented until the high limit of "999" is reached. When the high limit value is reached, the counter value is no longer incremented on a positive signal edge. When the low limit of "0" is reached, the counter value is no longer decremented.

When the signal state at input S changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if RLO is "1" at the inputs CU and CD, the counter counts accordingly in the next scan cycle, even if no change in the signal edge is detected.

The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has the signal state "1", processing of the signal state of the CU, CD and S inputs has no effect on the counter value.

The signal state at output Q is "1" if the counter value is greater than zero. If the counter value is equal to zero, output Q has the signal state "0".

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

The "Assign parameters and count up / down" instruction needs a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

Parameters

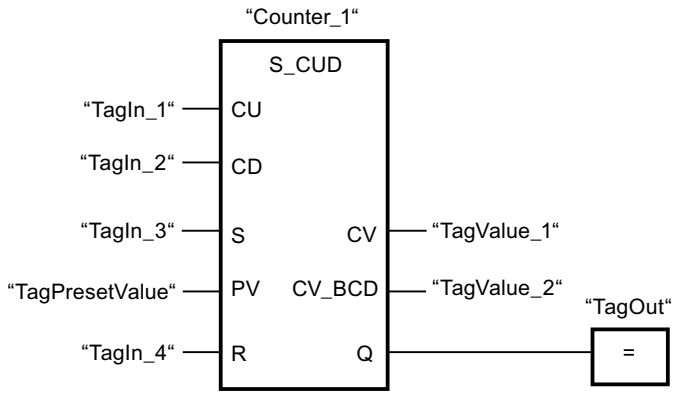
The following table shows the parameters of the "Assign parameters and count up / down" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Counter>	InOut/Input	COUNTER	C	Counter of the instruction The number of counters depends on the CPU.
CU	Input	BOOL	I, Q, M, D, L, T, C	Count up input
CD	Input	BOOL	I, Q, M, D, L, T, C or constant	Count down input
S	Input	BOOL	I, Q, M, D, L, T, C or constant	Input for presetting counter
PV	Input	WORD	I, Q, M, D, L or constant	Preset counter value (C#0 to C#999)
R	Input	BOOL	I, Q, M, D, L, T, C or constant	Reset input
CV	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value (hexadecimal)
CV_BCD	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the counter

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the signal state at the "TagIn_1" or "TagIn_2" input changes from "0" to "1" (positive signal edge), the "Assign parameters and count up / down" instruction is executed. When there is a

positive signal edge at the "TagIn_1" input and the current counter value is less than "999", the counter value is incremented by one. When there is a positive signal edge at the "TagIn_2" input and the current counter value is greater than "0", the counter value is decremented by one.

When the signal state at the "TagIn_3" input changes from "0" to "1", the counter value is set to the value of the "TagPresetValue" operand. The counter value is reset to "0" when the "TagIn_4" operand has signal state "1".

The current counter value is hexadecimal in the "TagValue_1" operand and BCD-coded in the "TagValue_2" operand.

The "TagOut" output has the signal state "1" as long as the current counter value is not equal to "0".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

S_CU: Assign parameters and count up

Description

You can use the "Assign parameters and count up" instruction to increment the value of a counter. If the signal state at the CU input changes from "0" to "1" (positive signal edge), the current counter value is incremented by one. The current counter value is output as a hexadecimal value at output CV and BCD-coded at output CV_BCD. The count is incremented until the limit of "999" is reached. When the limit is reached, the counter value is no longer incremented on a positive signal edge.

When the signal state at input S changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if RLO at input CU is "1", the counter will count accordingly in the next scan cycle, even when no change has been detected in the signal edge.

The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has the signal state "1", processing of the signal state of the CU and S inputs has no effect on the counter value.

The signal state at output Q is "1" if the counter value is greater than zero. If the counter value is equal to zero, output Q has the signal state "0".

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

The "Assign parameters and count up" instruction needs a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

Parameters

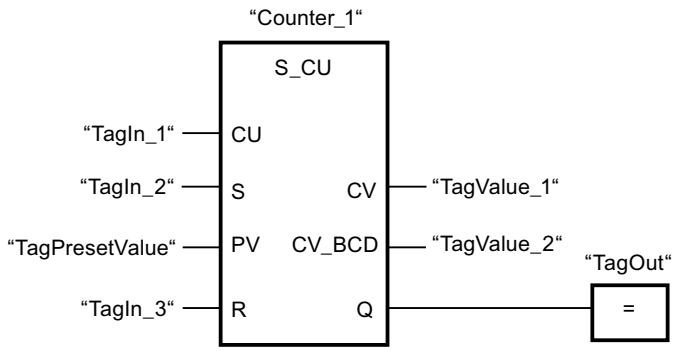
The following table shows the parameters of the "Assign parameters and count up" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Counter>	InOut/Input	COUNTER	C	Counter of the instruction The number of counters depends on the CPU.
CU	Input	BOOL	I, Q, M, D, L, T, C	Count up input
S	Input	BOOL	I, Q, M, D, L, T, C or constant	Input for presetting counter
PV	Input	WORD	I, Q, M, D, L or constant	Preset counter value (C#0 to C#999)
R	Input	BOOL	I, Q, M, D, L, T, C or constant	Reset input
CV	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value (hexadecimal)
CV_BCD	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the counter

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



When the signal state at the "TagIn_1" input changes from "0" to "1" (positive signal edge) and the current counter value is less than "999", the counter value is incremented by one. When the signal state at the "TagIn_2" input changes from "0" to "1", the counter value is set to the value of the "TagPresetValue" operand. The counter value is reset to "0" when the "TagIn_3" operand has signal state "1".

The current counter value is hexadecimal in the "TagValue_1" operand and BCD-coded in the "TagValue_2" operand.

The "TagOut" output has the signal state "1" as long as the current counter value is not equal to "0".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

S_CD: Assign parameters and count down

Description

You can use the "Assign parameters and count down" instruction to decrement the value of a counter. If the signal state at the CD input changes from "0" to "1" (positive signal edge), the counter value is decremented by one. The current counter value is output as a hexadecimal value at output CV and BCD-coded at output CV_BCD. The count is decremented until the low limit of "0" is reached. When the low limit is reached, the counter value is no longer decremented on a positive signal edge.

When the signal state at input S changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if RLO at input CD is "1", the counter will count accordingly in the next scan cycle, even when no change has been detected in the signal edge.

The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has the signal state "1", processing of the signal state of the CD and S inputs has no effect on the counter value.

The signal state at output Q is "1" if the counter value is greater than zero. If the counter value is equal to zero, output Q has the signal state "0".

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

The "Assign parameters and count down" instruction needs a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

Parameters

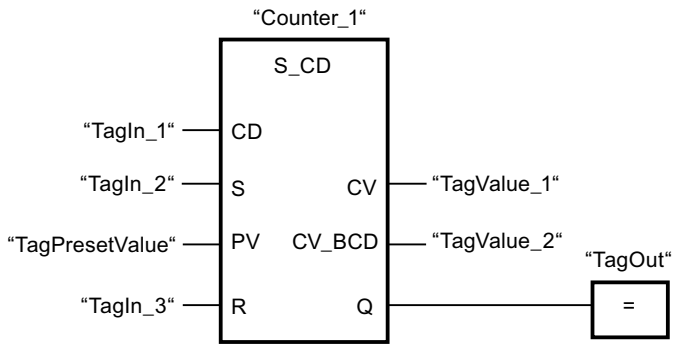
The following table shows the parameters of the "Assign parameters and count down" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Counter>	InOut/Input	COUNTER	C	Counter of the instruction The number of counters depends on the CPU.
CD	Input	BOOL	I, Q, M, D, L or constant	Count down input
S	Input	BOOL	I, Q, M, D, L, T, C or constant	Input for presetting counter
PV	Input	WORD	I, Q, M, D, L or constant	Preset counter value (C#0 to C#999)
R	Input	BOOL	I, Q, M, D, L, T, C or constant	Reset input
CV	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value (hexadecimal)
CV_BCD	Output	WORD, S5TIME, DATE	I, Q, M, D, L	Current counter value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the counter

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



When the signal state at the "TagIn_1" input changes from "0" to "1" (positive signal edge) and the current counter value is greater than "0", the counter value is decremented by one. When the signal state at the "TagIn_2" input changes from "0" to "1", the counter value is set to the value of the "TagPresetValue" operand. The counter value is reset to "0" when the "TagIn_3" operand has signal state "1".

The current counter value is hexadecimal in the "TagValue_1" operand and BCD-coded in the "TagValue_2" operand.

The "TagOut" output has the signal state "1" as long as the current counter value is not equal to "0".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

SC: Set counter value

Description

You can use the "Set counter value" instruction to set the value of a counter. The instruction is executed when the result of logic operation (RLO) at the start input of the instruction changes from "0" to "1". When the instruction is executed, the counter is set to the specified counter value.

The "Set counter value" instruction needs a preceding logic operation for the edge evaluation and can only be placed on the right edge of the network.

Parameters

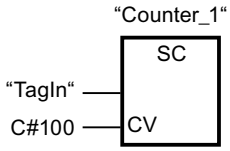
The following table lists the parameters of the "Set counter value" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L	Start input
CV	Input	WORD	I, Q, M, D, L or constant	Value with which the counter is preset in the BCD format. (C#0 to C#999)
<Counter>	InOut/Input	COUNTER	C	Counter that is preset.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



the counter "Counter_1" starts with the value "100" when the signal state of the "TagIn" operand changes from "0" to "1".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

CU: Count up

Description

You use the "Count up" instruction to increment the value of the specified counter by the count of one on a rising edge at the start input. The count is incremented until the limit of "999" is reached. When the limit is reached, the counter value is no longer incremented on a positive signal edge.

The "Count up" instruction needs a preceding logic operation for the edge evaluation and can only be placed on the right edge of the network.

Parameters

The following table shows the parameters of the "Count up" instruction:

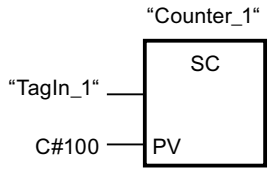
Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L	Start input
<Counter>	InOut/Input	COUNTER	C	Counter whose value is incremented.

For additional information on valid data types, refer to "See also".

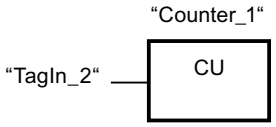
Example

The following example shows how the instruction works:

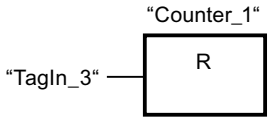
Network 1



Network 2



Network 3



When the signal state of the operand "TagIn_1" changes from "0" to "1" (positive signal edge), the counter "Counter_1" is preset with the value "100".

The value of "Counter_1" is incremented by the count of one when the signal state of the "TagIn_2" operand changes from "0" to "1".

If the "TagIn_3" operand returns signal state "1", the value of "Counter_1" is reset to "0".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

CD: Count down

Description

You use the "Count down" instruction to decrement the value of the specified counter by the count of one on a rising edge at the start input. The count is decremented until the limit of "0" is reached. When the limit is reached, the counter value is no longer changed on a positive signal edge.

The "Count down" instruction needs a preceding logic operation for the edge evaluation and can only be placed on the right edge of the network.

Parameters

The following table shows the parameters of the "Count down" instruction:

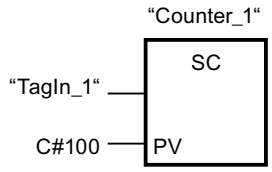
Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L	Start input
<Counter>	InOut/Input	COUNTER	C	Counter whose value is decremented.

For additional information on valid data types, refer to "See also".

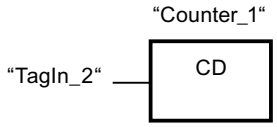
Example

The following example shows how the instruction works:

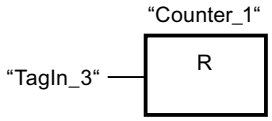
Network 1



Network 2



Network 3



When the signal state of the operand "TagIn_1" changes from "0" to "1" (positive signal edge), the counter "Counter_1" is preset with the value "100".

The value of "Counter_1" is incremented by the count of one when the signal state of the "TagIn_2" operand changes from "0" to "1".

If the "TagIn_3" operand returns signal state "1", the value of "Counter_1" is reset to "0".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Comparator operations

CMP ==: Equal

Description

You can use the "Equal" instruction to query whether the value at input IN1 is equal to the value at input IN2.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison.

The following table lists examples of string comparisons:

IN1	IN2	RLO of the instruction
'AA'	'AA'	1
'Hello World'	'HelloWorld'	0
'AA'	'aa'	0

The "Equal" instruction also compares individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

If IEC check is enabled, the operands to be compared must be of the same data type. If IEC check is not enabled, the width (length) of the operands must be the same. If the floating-point numbers are being compared, the operands to be compared must be of the same data type regardless of the IEC check setting.

Note

Comparison of floating-point numbers

If you want to compare the data types REAL or LREAL, instead of the instruction "CMP ==: Equal", use the instruction "IN_RANGE: Value within range".

Note

Comparison of the data type PORT

To be able to compare the operands of the PORT data type with the "Equal" instruction, you need to select the WORD data type from the drop-down list of the instructions box.

Parameters

The following table lists the parameters of the "Equal" instruction:

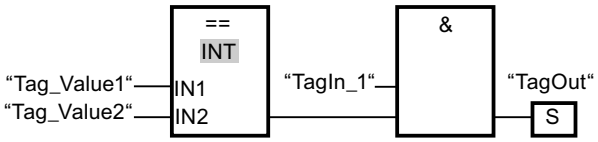
Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN1	Input	Bit strings, integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Bit strings, integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	First comparison value
IN2	Input	Bit strings, integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Bit strings, integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also":

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" = "Tag_Value2".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Selecting the data type of an FBD element (Page 2624)

CMP <>: Not equal

Description

You can use the "Not equal" instruction to query whether the value at input IN1 is not equal to the value at input IN2.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison.

The following table lists examples of string comparisons:

IN1	IN2	RLO of the instruction
'AA'	'aa'	1
'Hello World'	'HelloWorld'	1
'AA'	'AA'	0

The "Not equal" instruction also compares individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

If IEC check is enabled, the operands to be compared must be of the same data type. If IEC check is not enabled, the width (length) of the operands must be the same. If the floating-point numbers are being compared, the operands to be compared must be of the same data type regardless of the IEC check setting.

Note

Comparison of the data type PORT

To be able to compare the operands of the PORT data type with the "Not equal" instruction, you need to select the WORD data type from the drop-down list of the instructions box.

Parameters

The following table lists the parameters of the "Not equal" instruction:

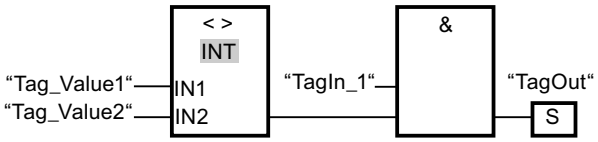
Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN1	Input	Bit strings, integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Bit strings, integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	First comparison value
IN2	Input	Bit strings, integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Bit strings, integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" <> "Tag_Value2".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Selecting the data type of an FBD element (Page 2624)

CMP >=: Greater or equal

Description

You can use the "Greater or equal" instruction to query whether the value at input IN1 is greater or equal to the value at input IN2. Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the longer string is considered greater.

The following table lists examples of string comparisons:

IN1	IN2	RLO of the instruction
'BB'	'AA'	1
'AAA'	'AA'	1
'Hello World'	'Hello World'	1
'Hello World'	'HelloWorld'	0
'AA'	'aa'	0
'AAA'	'a'	0

The "Greater or equal" instruction also compares individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

In comparing timer values, the RLO of the instruction is "1" if the timer at input IN1 is greater (more recent) than or equal to the timer at input IN2.

Parameters

The following table lists the parameters of the "Greater or equal" instruction:

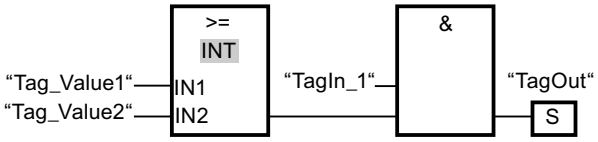
Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN1	Input	Integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	First comparison value
IN2	Input	Integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" >= "Tag_Value2".

See also

Overview of the valid data types (Page 2236)

Example of detecting the fill level of a storage area (Page 3093)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Selecting the data type of an FBD element (Page 2624)

CMP <=: Less or equal

Description

You can use the "Less or equal" instruction to query whether the value at input IN1 is less or equal to the value at input IN2. Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the shorter string is considered smaller.

The following table lists examples of string comparisons:

IN1	IN2	RLO of the instruction
'AA'	'aa'	1
'AAA'	'a'	1
'Hello World'	'Hello World'	1
'HelloWorld'	'Hello World'	0
'BB'	'AA'	0
'AAA'	'AA'	0

The "Less or equal" instruction also compares individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

In comparing timer values, the RLO of the instruction is "1" if the timer at input IN1 is smaller (less recent) than or equal to the timer at input IN2.

Parameters

The following table lists the parameters of the "Less or equal" instruction:

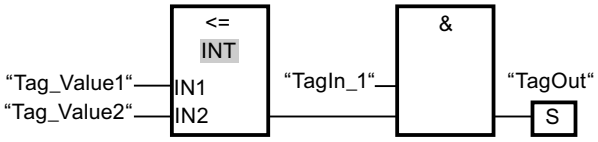
Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN1	Input	Integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	First comparison value
IN2	Input	Integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" <= "Tag_Value2".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Selecting the data type of an FBD element (Page 2624)

CMP >: Greater than

Description

You can use the "Greater than" instruction to query whether the value at input IN1 is greater than the value at input IN2. Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the longer string is considered greater.

The following table lists examples of string comparisons:

IN1	IN2	RLO of the instruction
'BB'	'AA'	1
'AAA'	'AA'	1
'AA'	'aa'	0
'AAA'	'a'	0

The "Greater than" instruction also compares individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

In comparing timer values, the RLO of the instruction is "1" if the timer at input IN1 is greater (more recent) than the timer at input IN2.

Parameters

The following table lists the parameters of the "Greater than" instruction:

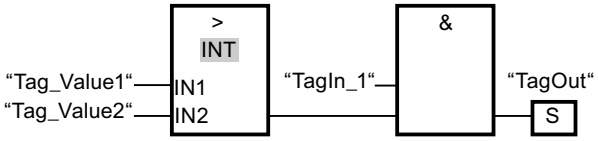
Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN1	Input	Integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	First comparison value
IN2	Input	Integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" > "Tag_Value2".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Selecting the data type of an FBD element (Page 2624)

CMP <: Less than

Description

You can use the "Less than" instruction to query whether the value at input IN1 is less than the value at input IN2. Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

The individual characters are compared by means of their code (for example, 'a' is greater than 'A') during the comparison of the strings. The comparison is performed from left to right. The first character to be different decides the result of the comparison. If the left part of the longer string is identical to the shorter string, the shorter string is considered smaller.

The following table lists examples of string comparisons:

<Operand1>	<Operand2>	RLO of the instruction
'AA'	'aa'	1
'AAA'	'a'	1

<Operand1>	<Operand2>	RLO of the instruction
'BB'	'AA'	0
'AAA'	'AA'	0

The "Less than" instruction also compares individual characters of a string. The number of the character to be compared is specified in square brackets next to the operand name. "MyString[2]", for example, compares the second character of the "MyString" string.

In comparing timer values, the RLO of the instruction is "1" if the timer at input IN1 is less (less recent) than the timer at input IN2 .

Parameters

The following table lists the parameters of the "Less than" instruction:

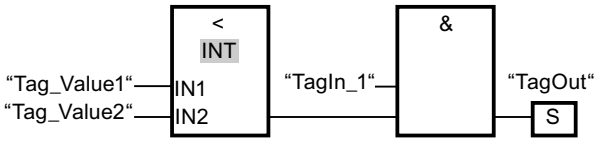
Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN1	Input	Integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	First comparison value
IN2	Input	Integers, floating-point numbers, character strings, TIME, DATE, TOD, DTL	Integers, floating-point numbers, character strings, TIME, LTIME, DATE, TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" < "Tag_Value2".

See also

Overview of the valid data types (Page 2236)

Example of detecting the fill level of a storage area (Page 3093)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Selecting the data type of an FBD element (Page 2624)

IN_RANGE: Value within range

Description

You can use the "Value within range" instruction to query whether of the value at input VAL is within a specific value range.

You specify the limits of the value range with the MIN and MAX inputs. The "Value within range" instruction compares the value at input VAL with the values of the inputs MIN and MAX and sends the result to the box output. If the value at input VAL satisfies the comparison MIN <= VAL or VAL <=MAX, the box output has the signal state "1". If the comparison is not fulfilled, the signal state is "0" at the box output.

The comparison function can only execute if the values to be compared are of the same data type.

Parameters

The following table shows the parameters of the "Value within range" instruction:

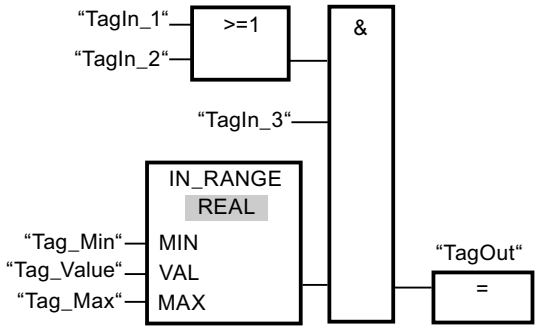
Parameters	Declaration	Data type	Memory area	Description
MIN	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Low limit of the value range
VAL	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Comparison value
MAX	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	High limit of the value range
Box output	Output	BOOL	I, Q, M, D, L	Result of the comparison

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also":

Example

The following example shows how the instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- The operands "TagIn_1" or "TagIn_2" have signal state "1".
- The operand "TagIn_3" has the signal state "1".
- The value of the operand "Tag_Value" is within the value range that is specified by the current values of the operands "Tag_Min" and "Tag_Max" (MIN <= VAL or VAL <= MAX).

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Selecting the data type of an FBD element (Page 2624)

OUT_RANGE: Value outside range

Description

You can use the "Value outside range" instruction to query whether of the value at input VAL is outside a specific value range.

You specify the limits of the value range with the MIN and MAX inputs. The "Value outside range" instruction compares the value at input VAL with the values of the inputs MIN and MAX and sends the result to the box output. If the value at input VAL satisfies the comparison MIN > VAL or VAL >MAX, the box output has the signal state "1". The box output has the signal state "1" if a specified operand of data type REAL has an invalid value.

The box output returns the signal state "0", if the value at input VAL does not satisfy the MIN > VAL or VAL > MAX condition.

The comparison function can only execute if the values to be compared are of the same data type.

Parameters

The following table shows the parameters of the "Value outside range" instruction:

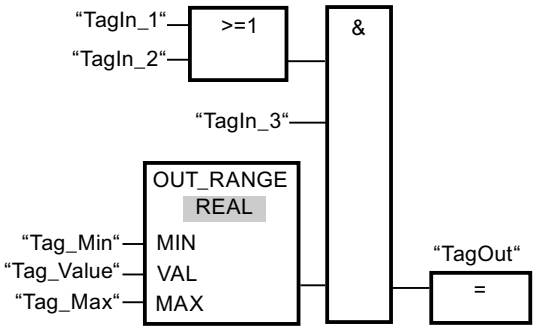
Parameters	Declaration	Data type	Memory area	Description
MIN	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Low limit of the value range
VAL	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Comparison value
MAX	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	High limit of the value range
Box output	Output	BOOL	I, Q, M, D, L	Result of the comparison

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The operand "TagIn_3" has the signal state "1".
- The value of the operand "Tag_Value" is outside the value range that is specified by the values of the operands "Tag_Min" and "Tag_Max" (MIN > VAL or VAL > MAX).

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Selecting the data type of an FBD element (Page 2624)

OK: Check validity

Description

You can use the "Check validity" instruction to check if the value of an operand (<Operand>) is a valid floating-point number. The check is performed in every program cycle. If the operand value at the time of the query is a valid floating-point number, the output box will return the signal state"1". In all other cases, the signal state at the output of the "Check validity" instruction is "0".

You can use the "Check validity" instruction together with the EN mechanism. If you connect the instruction box to an EN enable input, the enable input is set only when the result of the validity query of the value is positive. You can use this function to ensure that an instruction is enabled only when the value of the specified operand is a valid floating-point number.

Parameters

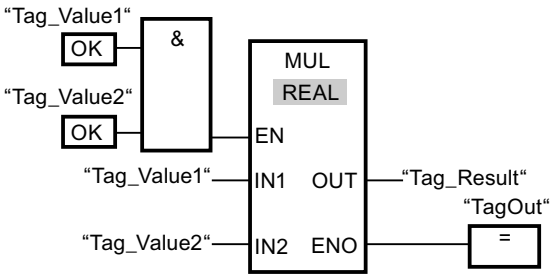
The following example shows how the "Check validity" instruction works:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L	Value to be checked.

For additional information on valid data types, refer to "See also":

Example

The following example shows how the instruction works:



When the values of the operands "Tag_Value1" and "Tag_Value2" show valid floating-point numbers, the "Multiply" (MUL) instruction is activated and the ENO output is set. During the execution of the "Multiply" (MUL) instruction, the value of the operand "Tag_Value1" is multiplied by the value of operand "Tag_Value2". The product of the multiplication is then stored in the operand "Tag_Result". If no errors occur during the execution of the instruction, the outputs ENO and "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Selecting the data type of an FBD element (Page 2624)

NOT_OK: Check invalidity

Description

You can use the "Check invalidity" instruction to check if the value of an operand (<Operand>) is an invalid floating-point number. The check is performed in every program cycle. If the operand value at the time of the query is a valid floating-point number, then the output box will return the signal state "1". In all other cases, the signal state on the output box is "0".

Parameters

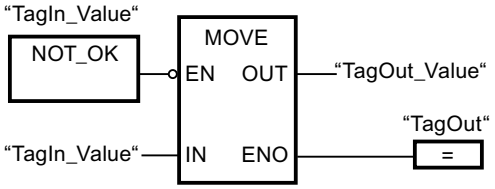
The following table shows the parameters of the instruction "Check invalidity":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L	Value to be checked.

For additional information on valid data types, refer to "See also":

Example

The following example shows how the instruction works:



When the value of operand "TagIn_Value" is an invalid floating-point number, the instruction "Move value" (MOVE) is not executed. The "TagOut" output is reset to signal state "0".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Selecting the data type of an FBD element (Page 2624)

VARIANT instructions

EQ_Type: Compare data type for EQUAL with the data type of a tag

Description

You can use the "Compare data type for EQUAL with the data type of a tag" instruction to query the data type of a tag to which a VARIANT points. You compare the data type of the tag to the data type of a tag that you have declared in the block interface to determine if they are "Equal".

The IN1 parameter must have the same VARIANT data type. The IN2 parameter can be an elementary data type or a PLC data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Parameters

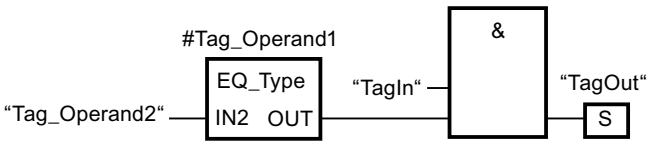
The following table shows the parameters of the "Compare data type for EQUAL with the data type of a tag" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	VARIANT	L	First operand
IN2	Input	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY, PLC data types	I, Q, M, D, L	Second operand

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operand "TagIn" has the signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the "#Tag_Operand1" operand is equal to "Tag_Operand2".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Selecting the data type of an FBD element (Page 2624)

NE_Type: Compare data type for UNEQUAL with the data type of a tag

Description

You can use the "Compare data type for UNEQUAL with the data type of a tag" instruction to query which data type the tag to which a VARIANT points does not have. You compare the data type of the tag to the data type of a tag that you have declared in the block interface to determine if they are "Unequal".

The IN1 parameter must have the same VARIANT data type. The IN2 parameter can be an elementary data type or a PLC data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Parameters

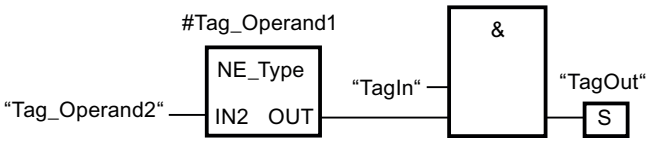
The following table shows the parameters of the "Compare data type for UNEQUAL with the data type of a tag" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	VARIANT	L	First operand
IN2	Input	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY, PLC data types	I, Q, M, D, L	Second operand

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operand "TagIn" has the signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the "#Tag_Operand1" operand is not equal to "Tag_Operand2".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Selecting the data type of an FBD element (Page 2624)

EQ_ElemType: Compare element data type for EQUAL with the data type of a tag

Description

You can use the "Compare element data type for EQUAL with the data type of a tag" instruction to query the data type of a tag to which a VARIANT points. You compare the data type of the tag to the data type of a tag that you have declared in the block interface to determine if they are "Equal".

The IN1 parameter must have the same VARIANT data type. The IN2 parameter can be an elementary data type or a PLC data type.

If the data type of the VARIANT tag is an ARRAY, the data type of the ARRAY elements is compared.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Parameters

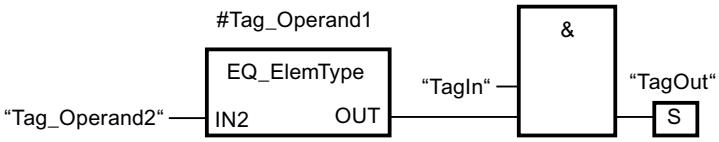
The following table shows the parameters of the "Compare element data type for EQUAL with the data type of a tag" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	VARIANT	L	First operand
IN2	Input	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY, PLC data types	I, Q, M, D, L	Second operand

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operand "TagIn" has the signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the "#Tag_Operand1" operand is equal to "Tag_Operand2".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Selecting the data type of an FBD element (Page 2624)

NE_ElemType: Compare element data type for UNEQUAL with the data type of a tag

Description

You can use the "Compare element data type for UNEQUAL with the data type of a tag" instruction to query which data type the tag to which a VARIANT points does not have. You compare the data type of the tag to the data type of a tag that you have declared in the block interface to determine if they are "Unequal".

The IN1 parameter must have the same VARIANT data type. The IN2 parameter can be an elementary data type or a PLC data type.

If the data type of the VARIANT tag is an ARRAY, the data type of the ARRAY elements is compared.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Parameters

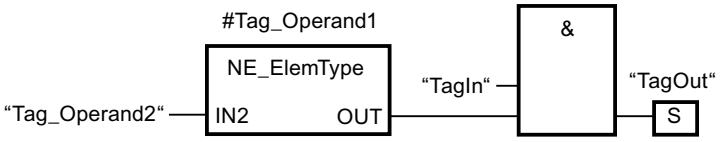
The following table shows the parameters of the "Compare element data type for UNEQUAL with the data type of a tag" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	VARIANT	L	First operand
IN2	Input	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY, PLC data types	I, Q, M, D, L	Second operand

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operand "TagIn" has the signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the "#Tag_Operand1" operand is not equal to "Tag_Operand2".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Selecting the data type of an FBD element (Page 2624)

IS_NULL: Compare for EQUALS ZERO

Description

You can use the instruction "Compare for EQUALS ZERO" to query whether the VARIANT does not point to an object.

The <Operand> must have the VARIANT data type.

Note

VARIANT tag points to an ANY pointer

If the VARIANT tag points to an ANY pointer, the instruction always returns the result RLO = "0" even if the ANY pointer is ZERO.

Parameters

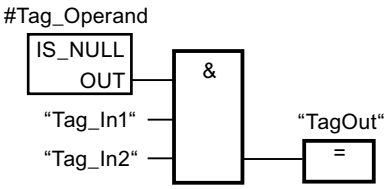
The following table shows the parameters of the "Compare for EQUALS ZERO" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	VARIANT	L	Operand that is compared for EQUAL ZERO

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "Tag_In1" and "Tag_In2" have signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the "#Tag_Operand" operand does not point to an object.

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Selecting the data type of an FBD element (Page 2624)

NOT_NULL: Compare for UNEQUALS ZERO

Description

You can use the instruction "Compare for UNEQUALS ZERO" to query whether the VARIANT points to an object.

The <Operand> must have the VARIANT data type.

Note

VARIANT tag points to an ANY pointer

If the VARIANT tag points to an ANY pointer, the instruction always returns the result RLO = "1" even if the ANY pointer is ZERO.

Parameters

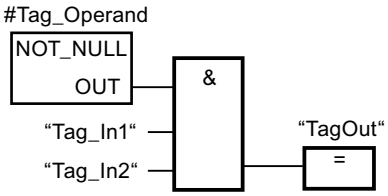
The following table shows the parameters of the "Compare for UNEQUALS ZERO" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	VARIANT	L	Operand that is compared for NOT EQUAL ZERO

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "Tag_In1" and "Tag_In2" have signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the "#Tag_Operand" operand points to an object.

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Selecting the data type of an FBD element (Page 2624)

IS_ARRAY: Check for ARRAY

Description

You can use the "Check for ARRAY" instruction to query whether the VARIANT points to a tag of the ARRAY data type.

The <Operand> must have the VARIANT data type.

Parameters

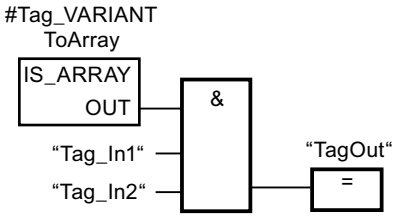
The following table shows the parameters of the "Check for ARRAY" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	VARIANT	L	Operand that is queried for ARRAY

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "Tag_In1" and "Tag_In2" have signal state "1".
- The condition of the comparison instruction is fulfilled, i.e. the "#Tag_VARIANTToArray" operand has the ARRAY data type.

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Selecting the data type of an FBD element (Page 2624)

Math functions

CALCULATE: Calculate

Description

The "Calculate" instruction is used to define and execute an expression for the calculation of mathematical operations or complex logic operations depending on the selected data type.

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box. Depending on the selected data type, you can combine the functionality of specific instructions to execute a complex calculation. The expression to be calculated is specified via a dialog you can open via the "Calculator" icon at the top of the instruction box. The expression can contain the names of the input parameters and the syntax of the instructions. It is not permitted to specify operand names or operand addresses.

In its initial state, the instruction box contains at least 2 inputs (IN1 and IN2). The number of inputs can be extended. The inserted inputs are numbered in ascending order in the box.

The values of the inputs are use to execute the specified expression. Not all defined inputs have to be used in the expression. The result of the instruction is transferred to the box output OUT.

Note

If one of the mathematical operations fails in the expression, then no result is transferred to the OUT output and the ENO enable output has the signal state "1".

If, in the expression, you use inputs that are not available in the box, these inputs are automatically inserted. Provided that there are no gaps in the numbering of the inputs that are to be newly defined in the expression. You cannot, for example, use the input IN4 in the expression if the input IN3 is not defined.

Enable output ENO has the signal state "0" if one of the following conditions is fulfilled:

- Enable input EN has the signal state "0".
- The result or an interim result of the "Calculate" instruction is outside the range permitted for the data type specified at output OUT.
- A floating-point number has an invalid value.
- An error occurred during the execution of one of the instructions specified in the expression.

The following table shows the instructions that, depending on the selected data type, can be combined and executed in the expression of the "Calculate" instruction:

Data type	Instruction	Syntax	Example
Bit strings	AND: AND logic operation	AND	IN1 AND IN2 OR IN3
	OR: OR logic operation	OR	
	XOR: EXCLUSIVE OR logic operation	XOR	
	INV: Create ones complement	NOT	
	SWAP: Swap [1)]	SWAP	

Data type	Instruction	Syntax	Example
Integers	ADD: Add	+	(IN1 + IN2) * IN3; (ABS(IN2)) * (ABS(IN1))
	SUB: Subtract	-	
	MUL: Multiply	*	
	DIV: Divide	/	
	MOD: Return remainder of division	MOD	
	INV: Create ones complement	NOT	
	NEG: Create twos complement	-(in1)	
	ABS: Form absolute value	ABS( )	
Floating-point numbers	ADD: Add	+	((SIN(IN2) * SIN(IN2) + (SIN(IN3) * SIN(IN3)) / IN3)); (SQR(SIN(IN2)) + (SQR(COS(IN3)) / IN2))
	SUB: Subtract	-	
	MUL: Multiply	*	
	DIV: Divide	/	
	EXPT: Exponentiate	**	
	ABS: Form absolute value	ABS( )	
	SQR: Form square	SQR( )	
	SQRT: Form square root	SQRT( )	
	LN: Form natural logarithm	LN( )	
	EXP: Form exponential value	EXP( )	
	FRAC: Return fraction	FRAC( )	
	SIN: Form sine value	SIN( )	
	COS: Form cosine value	COS( )	
	TAN: Form tangent value	TAN( )	
	ASIN: Form arcsine value	ASIN( )	
	ACOS: Form arccosine value	ACOS( )	
	ATAN: Form arctangent value	ATAN( )	
	NEG: Create twos complement	-(in1)	
	TRUNC: Truncate numerical value	TRUNC( )	
	ROUND: Round numerical value	ROUND( )	
	CEIL: Generate next higher integer from floating-point number	CEIL( )	
	FLOOR: Generate next lower integer from floating-point number	FLOOR( )	
[1] Not possible for data type BYTE.			

Parameters

The following table shows the parameters of the instruction "Calculate":

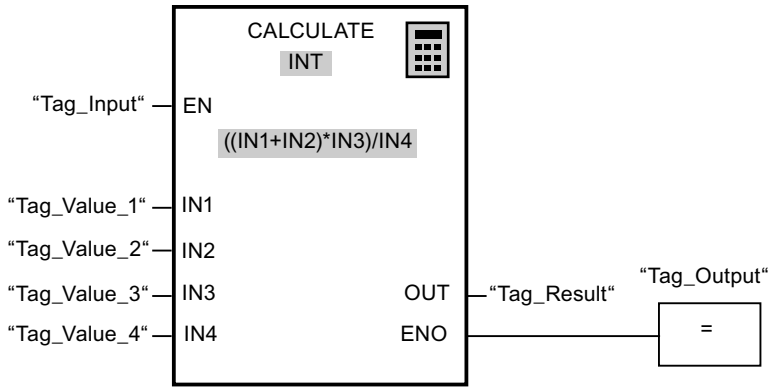
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
IN1	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	First available input
IN2	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Second available input
INn	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Additionally inserted inputs
OUT	Output	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Output to which the end result is to be transferred.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN1	Tag_Value_1	4
IN2	Tag_Value_2	4
IN3	Tag_Value_3	3
IN4	Tag_Value_4	2
OUT	Tag_Result	12

The "Calculate" instruction is executed when input "Tag_Input" has the signal state "1". The value of operand "Tag_Value_1" is added to the value of operand "Tag_Value_2". The sum is multiplied with the value of the operand "Tag_Value_3". The product is divided by the value of the operand "Tag_Value_4". The quotient is transferred as end result to the operand

"Tag_Result" at the OUT output of the instruction. If no errors occur during the execution of the individual instructions, output ENO and the operand "Tag_Output" are set to "1".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Adding additional inputs and outputs to FBD elements (Page 2639)

Removing instruction inputs and outputs (Page 2640)

Selecting the data type of an FBD element (Page 2624)

Basics of the EN/ENO mechanism (Page 2447)

ADD: Add

Description

You can use the "Add" instruction to add the value at input IN1 to the value at input IN2 and query the sum at output OUT (OUT := IN1+IN2).

In its initial state the instruction box contains at least 2 inputs (IN1 and IN2). The number of inputs can be extended. The inserted inputs are numbered in ascending order in the box. During the execution of the instruction, the values of all available input parameters are added. The sum is stored at output OUT.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at output OUT.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the instruction "Add":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	First number to be added

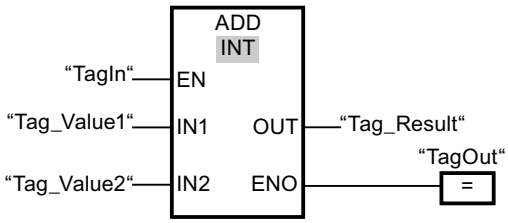
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Second number to be added
INn	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Optional input values, which are added.
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Sum

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also":

Example

The following example shows how the instruction works:



If the operand "TagIn" has the signal state "1", the "Add" instruction is executed. The value of operand "Tag_Value1" is added to the value of operand "Tag_Value2". The result of the addition is stored in the operand "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Removing instruction inputs and outputs (Page 2640)

Basics of the EN/ENO mechanism (Page 2447)

Selecting a data type (Page 2624)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Selecting the data type of an FBD element (Page 2624)

Adding additional inputs and outputs to FBD elements (Page 2639)

SUB: Subtract

Description

You can use the "Subtract" instruction to subtract the value at input IN2 from the value at input IN1 and query the difference at output OUT (OUT := IN1-IN2).

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at output OUT.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the instruction "Subtract":

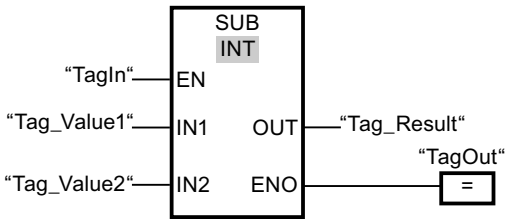
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Minuend
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Subtrahend
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Difference

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the operand "TagIn" has the signal state "1", the "Subtract" instruction is executed. The value of operand "Tag_Value2" is subtracted from the value of operand "Tag_Value1". The result of the subtraction is stored in the operand "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Selecting a data type (Page 2624)

MUL: Multiply

Description

You can use the "Multiply" instruction to multiply the value at input IN1 with the value at input IN2 and query the total at output OUT (OUT := IN1*IN2).

In its initial state the instruction box contains at least 2 inputs (IN1 and IN2). The number of inputs can be extended. The inserted inputs are numbered in ascending order in the box. When the instruction is executed, the values of all available input parameters are multiplied. The product is stored at the OUT output.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The result is outside the range permitted for the data type specified at output OUT.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the instruction "Multiply":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	First value for multiplication

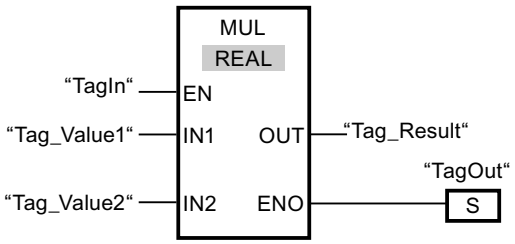
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Second value for multiplication
INn	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Optional input values, which are multiplied.
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Product

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the operand "TagIn" has the signal state "1", the "Multiply" instruction is executed. The value of operand "Tag_Value1" is multiplied with the value of operand "Tag_Value2". The result of the multiplication is stored in the operand "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Removing instruction inputs and outputs (Page 2640)

Basics of the EN/ENO mechanism (Page 2447)

Selecting a data type (Page 2624)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Adding additional inputs and outputs to FBD elements (Page 2639)

Selecting the data type of an FBD element (Page 2624)

DIV: Divide

Description

You can use the "Divide" instruction to divide the value at input IN1 by the value at input IN2 and query the quotient at output OUT (OUT := IN1/IN2).

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at output OUT.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the instruction "Divide":

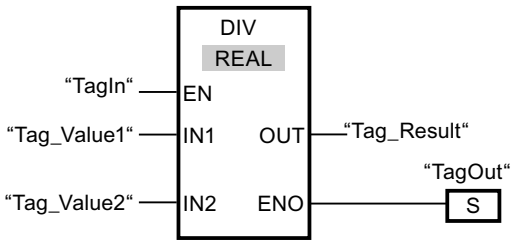
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Dividend
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Divisor
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Quotient value

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the operand "TagIn" has the signal state "1", the "Divide" instruction is executed. The value of operand "Tag_Value1" is divided by the value of operand "Tag_Value2". The result of the division is stored in the operand "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Selecting a data type (Page 2624)

MOD: Return remainder of division

Description

You can use the "Return remainder of division" instruction to divide the value at input IN1 by the value at input IN2 and query the quotient at output OUT.

Parameters

The following table shows the parameters of the instruction "Return remainder of division":

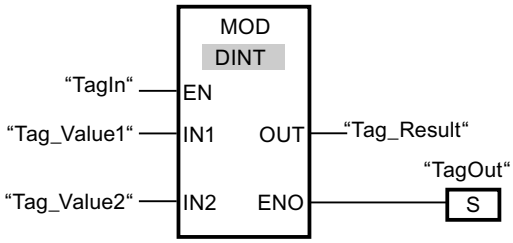
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN1	Input	Integers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Dividend
IN2	Input	Integers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Divisor
OUT	Output	Integers	I, Q, M, D, L, P	I, Q, M, D, L, P	Remainder of division

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the operand "TagIn" has the signal state "1", the "Return remainder of division" instruction is executed. The value of operand "Tag_Value1" is divided by the value of operand "Tag_Value2". The remainder of division is stored in operand "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Selecting a data type (Page 2624)

NEG: Create twos complement

Description

You can use the "Create twos complement" instruction to change the sign of the value at input IN and query the result at output OUT. If there is a positive value at input IN, for example, the negative equivalent of this value is sent to output OUT.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at output OUT.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Create twos complement" instruction:

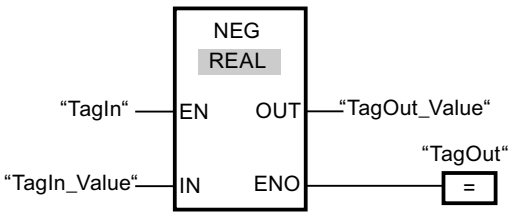
Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	SINT, INT, DINT, Floating-point numbers	SINT, INT, DINT, LINT, Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value
OUT	Output	SINT, INT, DINT, Floating-point numbers	SINT, INT, DINT, LINT, Floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Twos complement of the input value

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the operand "TagIn" has the signal state "1", the "Create twos complement" instruction is executed. The sign of the value at input "TagIn_Value" is changed and the result is stored at output "TagOut_Value". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Selecting the data type of an FBD element (Page 2624)

Adding additional inputs and outputs to FBD elements (Page 2639)

INC: Increment

Description

You can use the "Increment" instruction to change the value of the operand at parameter IN/OUT to the next higher value and query the result.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Increment" instruction:

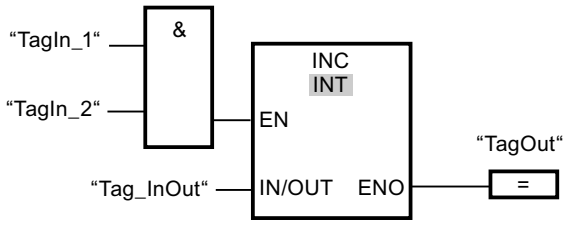
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN/OUT	InOut	Integers	I, Q, M, D, L	I, Q, M, D, L	Value to be incremented.

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the operands TagIn_1 and TagIn_2 have the signal state "1", the value of the operand "Tag_InOut" is incremented by one and the output "TagOut" is set.

See also

Overview of the valid data types (Page 2236)
Basics of the EN/ENO mechanism (Page 2447)
Inserting FBD elements (Page 2621)
Editing FBD elements (Page 2635)
Inserting operands in FBD instructions (Page 2643)
Selecting the data type of an FBD element (Page 2624)
Adding additional inputs and outputs to FBD elements (Page 2639)

DEC: Decrement

Description

You can use the "Decrement" instruction to change the value of the operand at parameter IN/OUT to the next lower value and query the result.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the instruction "Decrement":

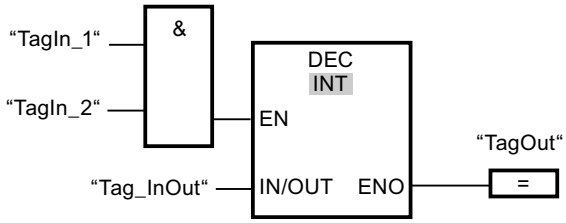
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN/OUT	InOut	Integers	I, Q, M, D, L	I, Q, M, D, L	Value to be decremented.

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the operands "TagIn_1" and "TagIn_2" have the signal state "1", the value of the operand "Tag_InOut" is decremented by one and the output "TagOut" is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Selecting the data type of an FBD element (Page 2624)

Inserting operands in FBD instructions (Page 2643)

Adding additional inputs and outputs to FBD elements (Page 2639)

ABS: Form absolute value

Description

You can use the "Form absolute value" instruction to calculate the absolute value of the value specified at input IN. The result of the instruction is provided at output OUT and can be queried there.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the instruction "Form absolute value":

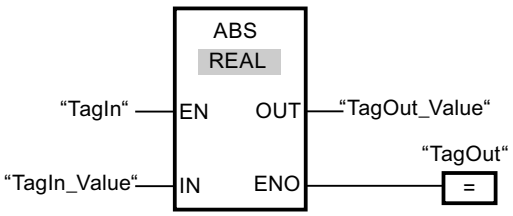
Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	SINT, INT, DINT, floating-point numbers	SINT, INT, DINT, LINT, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value
OUT	Output	SINT, INT, DINT, floating-point numbers	SINT, INT, DINT, LINT, floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Absolute value of the input value

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	TagIn_Value	-6.234
OUT	TagOut_Value	6.234

If the operand "TagIn" has the signal state "1", the "Form absolute value" instruction is executed. The instruction calculates the absolute value of the value at input "TagIn_Value" and sends the result to output "TagOut_Value". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Selecting the data type of an FBD element (Page 2624)

Adding additional inputs and outputs to FBD elements (Page 2639)

MIN: Get minimum

Description

The "Get minimum" instruction compares the values at the available inputs and writes the lowest value to the OUT output. The number of inputs can be expanded at the instruction box by additional inputs. The inputs are numbered in ascending order in the box.

In its initial state the instruction contains at least two inputs (IN1 and IN2) and no more than 100 inputs.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- Enable input EN has the signal state "0".
- The implicit conversion of the data types fails during execution of the instruction.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Get minimum" instruction:

Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers, TIME, TOD, DATE	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	First input value

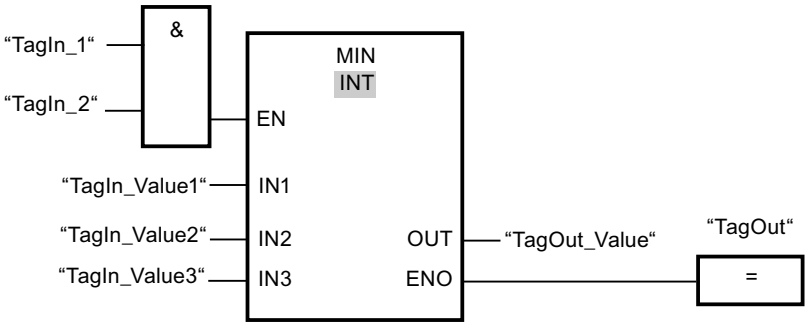
Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
IN2	Input	Integers, floating-point numbers, TIME, TOD, DATE	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Second input value
INn	Input	Integers, floating-point numbers, TIME, TOD, DATE	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Additionally inserted inputs whose values are to be compared
OUT	Output	Integers, floating-point numbers, TIME, TOD, DATE	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	I, Q, M, D, L, P	Result
The data types TOD, LTOD, DATE, and LDT can only be used if the IEC test is not enabled.						

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
IN3	TagIn_Value3	13333
OUT	TagOut_Value	12222

If the "TagIn_1" and "TagIn_2" operands have signal state "1", the "Get minimum" instruction is executed. The instruction compares the values of the specified operands and copies the lowest value ("TagIn_Value1") to output "TagOut_Value". If the instruction is executed without errors, the output "TagOut" is set.

See also

Overview of the valid data types (Page 2236)

Removing instruction inputs and outputs (Page 2640)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Selecting the data type of an FBD element (Page 2624)

Adding additional inputs and outputs to FBD elements (Page 2639)

MAX: Get maximum

Description

The "Get maximum" instruction compares the values at the available inputs and writes the highest value to the OUT output. The number of inputs can be expanded at the instruction box by additional inputs. The inputs are numbered in ascending order in the box.

In its initial state the instruction contains at least two inputs (IN1 and IN2) and no more than 100 inputs.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- Enable input EN has the signal state "0".
- The implicit conversion of the data types fails during execution of the instruction.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Get maximum" instruction:

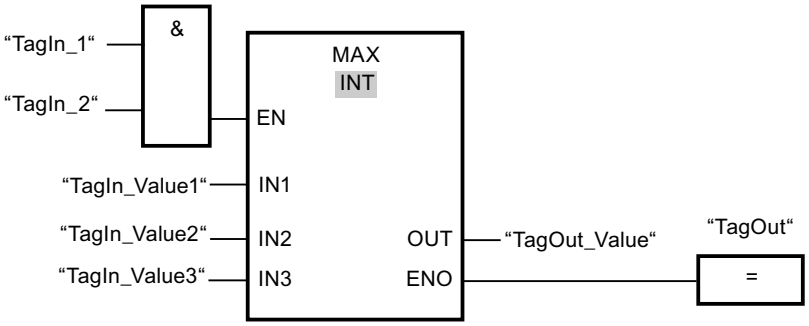
Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers, TIME, TOD, DATE	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	First input value
IN2	Input	Integers, floating-point numbers, TIME, TOD, DATE	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Second input value
INn	Input	Integers, floating-point numbers, TIME, TOD, DATE	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Additionally inserted inputs whose values are to be compared.
OUT	Output	Integers, floating-point numbers, TIME, TOD, DATE	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	I, Q, M, D, L, P	Result
The data types TOD, LTOD, DATE, and LDT can only be used if the IEC test is not enabled.						

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
IN3	TagIn_Value3	13333
OUT	TagOut_Value	14444

If the "TagIn_1" and "TagIn_2" operands have signal state "1", the "Get maximum" instruction is executed. The instruction compares the values of the specified operands and copies the highest value ("TagIn_Value2") to output "TagOut_Value". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 2236)

Removing instruction inputs and outputs (Page 2640)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Selecting the data type of an FBD element (Page 2624)

Adding additional inputs and outputs to FBD elements (Page 2639)

LIMIT: Set limit value

Description

You can use the "Set limit value" instruction to limit the value at input IN to the values at the inputs MN and MX. If the value at the IN input meets the MN condition <= IN <= MX, it is copied to the OUT output. If the condition is not fulfilled and the input value IN is below the low limit

MN, output OUT is set to the value of the input MN. If the high limit MX is exceeded, output OUT is set to the value of the input MX.

If the value at input MN is greater than that at input MX, the result is undefined and the enable output ENO is "0".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The specified tags are not of the same data type.
- An operand has an invalid value.
- The value at the MN input is greater than the value at input MX.

Parameters

The following table shows the parameters of the "Set limit value" instruction:

Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
MN	Input	Integers, floating-point numbers, TIME, TOD, DATE	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Low limit
IN	Input	Integers, floating-point numbers, TIME, TOD, DATE	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value
MX	Input	Integers, floating-point numbers, TIME, TOD, DATE	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	High limit

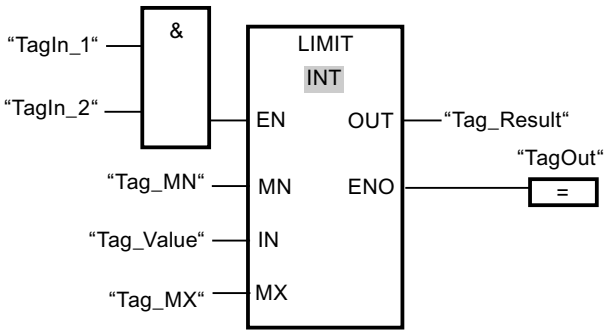
Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
OUT	Output	Integers, floating-point numbers, TIME, TOD, DATE	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	I, Q, M, D, L, P	Result
The data types TOD, LTOD, DATE, and LDT can only be used if the IEC test is not enabled.						

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
MN	Tag_MN	12000
IN	Tag_Value	8000
MX	Tag_MX	16000
OUT	Tag_Result	12000

If the "TagIn_1" and "TagIn_2" operands have the signal state "1", the "Set limit value" instruction is executed. The value of operand "Tag_Value" is compared with the values of operands "Tag_MN" and "Tag_MX". Since the value at the operand "Tag_Value" is less than the low limit, the value of the operand "Tag_MN" is copied to output "Tag_Result". If the instruction is executed without errors, the output "TagOut" is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Adding additional inputs and outputs to FBD elements (Page 2639)

Selecting the data type of an FBD element (Page 2624)

SQR: Form square

Description

You can use the "Form square" instruction to square the value at input IN and query the result at output OUT.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form square":

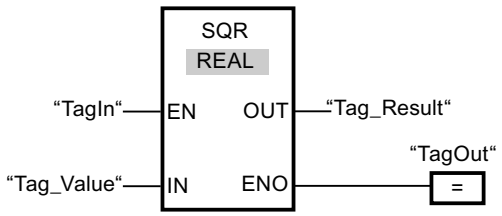
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Square of the input value

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	5.0
OUT	Tag_Result	25.0

If the operand "TagIn" has the signal state "1", the "Form square" instruction is executed. The instruction squares the value of the operand "Tag_Value" and sends the result to output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

SQRT: Form square root

Description

You can use the "Form square root" instruction to form the square root of the value at input IN and query the result at output OUT. The instruction has a positive result if the input value is greater than zero. If input values are less than zero, output OUT returns an invalid floating-point number. If the value at input IN is "0", the result is also "0".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.
- The value at input IN is negative.

Parameters

The following table shows the parameters of the instruction "Form square root":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output

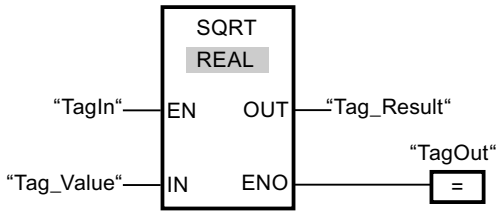
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L	I, Q, M, D, L	Square root of the input value

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	25.0
OUT	Tag_Result	5.0

If the operand "TagIn" has the signal state "1", the "Form square root" instruction is executed. The instruction calculates the square root of the operand "Tag_Value" and sends the result to output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

LN: Form natural logarithm

Description

You can use the "Form natural logarithm" instruction to calculate the natural logarithm to base e (e = 2.718282) of the value at input IN. The result is sent to output OUT and can be queried there. The instruction has a positive result if the input value is greater than zero. If input values are less than zero, output OUT returns an invalid floating-point number.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.
- The value at input IN is negative.

Parameters

The following table shows the parameters of the instruction "Form natural logarithm":

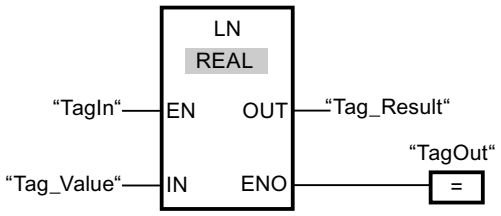
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Natural logarithm of the input value

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the operand "TagIn" has the signal state "1", the "Form natural logarithm" instruction is executed. The instruction forms the natural logarithm of the value at input "Tag_Value" and sends the result to output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Inserting FBD elements (Page 2621)
Editing FBD elements (Page 2635)
Inserting operands in FBD instructions (Page 2643)
Overview of the valid data types (Page 2236)
Basics of the EN/ENO mechanism (Page 2447)

EXP: Form exponential value

Description

You can use the "Form exponential value" instruction to calculate the exponent from the base e (e = 2.718282 and the value specified at input IN. The result is provided at output OUTand can be queried there (OUT = e^{IN}).

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form exponential value":

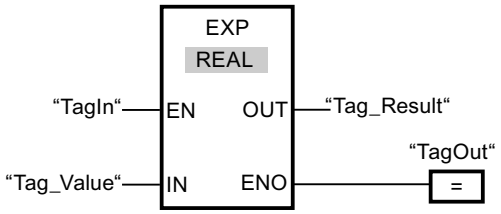
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Exponential value of the input value IN

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the operand "TagIn" has the signal state "1", the "Form exponential value" instruction is executed. The instruction forms the exponent from base e and the value of the operand "Tag_Value" and sends the result to output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

SIN: Form sine value

Description

You can use the "Form sine value" instruction to calculate the sine of an angle. The size of the angle is specified in the radian measure at input IN. The result of the instruction is provided at output OUT and can be queried there.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form sine value":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output

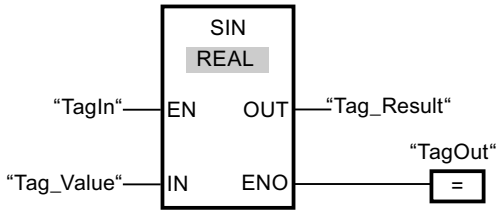
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Size of angle in the radian measure
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Sine of the specified angle

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	+1.570796 (π/2)
OUT	Tag_Result	1.0

If the operand "TagIn" has the signal state "1", the "Form sine value" instruction is executed. The instruction calculates the sine of the angle specified at input "Tag_Value" and sends the result to output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

COS: Form cosine value

Description

You can use the "Form cosine value" instruction to calculate the cosine of an angle. The size of the angle is specified in the radian measure at input IN. The result of the instruction is provided at output OUT and can be queried there.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form cosine value":

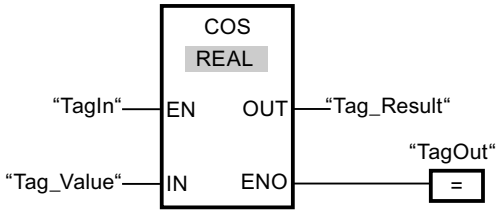
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Size of angle in the radian measure
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Cosine of the specified angle

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	+1.570796 (π/2)
OUT	Tag_Result	0

If the operand "TagIn" has the signal state "1", the "Form cosine value" instruction is executed. The instruction calculates the cosine of the angle specified at input "Tag_Value" and sends the result to output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

TAN: Form tangent value

Description

You can use the "Form tangent value" instruction to calculate the tangent of an angle. The size of the angle is specified in the radian measure at input IN. The result of the instruction is provided at output OUT and can be queried there.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form tangent value":

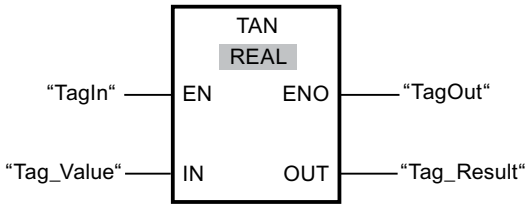
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Size of angle in the radian measure
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Tangent of the specified angle

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	+3.141593 (π)
OUT	Tag_Result	0

If the operand "TagIn" has the signal state "1", the "Form tangent value" instruction is executed. The instruction calculates the tangent of the angle specified at input "Tag_Value" and stores the result at output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

ASIN: Form arcsine value

Description

You can use the "Form arcsine value" instruction to calculate the size of the angle from the sine value specified at input IN, which corresponds to this value. Only valid floating-point numbers within the range -1 to +1 can be specified at input IN. The calculated angle size is output in radians at output OUT and can range in value from -π/2 to +π/2.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.
- The value at input IN is outside the permitted range (-1 to +1).

Parameters

The following table shows the parameters of the instruction "Form arcsine value":

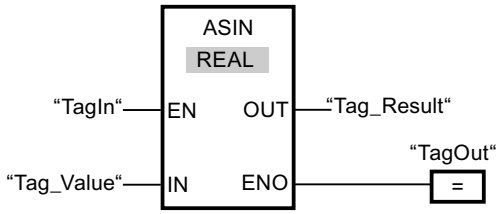
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Sine value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Size of angle in the radian measure

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	1.0
OUT	Tag_Result	+1.570796 (π/2)

If the operand "TagIn" has the signal state "1", the "Form arcsine value" instruction is executed. The instruction calculates the size of the angle corresponding to the sine value at input "Tag_Value". The result of the instruction is stored at output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

ACOS: Form arccosine value

Description

You can use the "Form arccosine value" instruction to calculate the size of the angle from the cosine value specified at input IN, which corresponds to this value. Only valid floating-point numbers within the range -1 to +1 can be specified at input IN. The calculated angle size is output in radians at output OUT and can range in value from 0 to +π.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.
- The value at input IN is outside the permitted range (-1 to +1).

Parameters

The following table shows the parameters of the instruction "Form arccosine value":

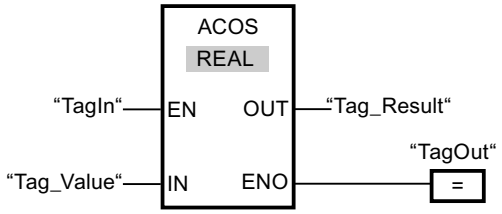
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Cosine value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Size of angle in the radian measure

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	0
OUT	Tag_Result	+1.570796 (π/2)

If the operand "TagIn" has the signal state "1", the "Form arccosine value" instruction is executed. The instruction calculates the size of the angle corresponding to the cosine value at input "Tag_Value". The result of the instruction is stored at output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

ATAN: Form arctangent value

Description

You can use the "Form arctangent value" instruction to calculate the size of the angle from the tangent value specified at input IN, which corresponds to this value. Only valid floating-point numbers may be specified at input IN. The calculated angle size is output in radians at the output OUT and can range in value from -π/2 to +π/2.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form arctangent value":

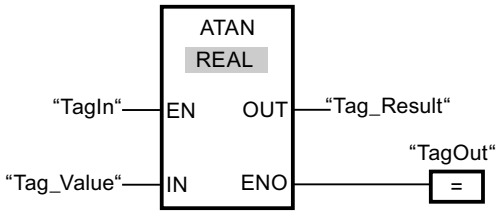
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Tangent value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Size of angle in the radian measure

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	1.0
OUT	Tag_Result	+0.785398 (π/4)

If the operand "TagIn" has the signal state "1", the "Form arctangent value" instruction is executed. The instruction calculates the size of the angle corresponding to the tangent value at input "Tag_Value". The result of the instruction is stored at output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

FRAC: Return fraction

Description

You can use the "Return fraction" instruction to determine the decimal places of the value at the IN input. The result of the query is stored at output OUT and can be queried there. If the input IN has e.g. the value 123.4567, the output OUT has the value 0.4567.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- Errors occur during the execution of the instruction, for example there is no valid floating-point number at the input.

Parameters

The following table shows the parameters of the "Return fraction" instruction:

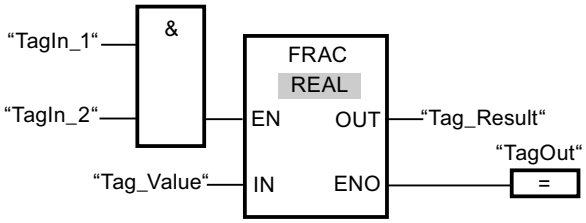
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value, whose decimal places are to be determined.
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Decimal places of the input value at input IN

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	2.555
OUT	Tag_Result	0.555

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the "Return fraction" instruction is executed. The decimal places from the value at the operand "Tag_Value" are copied to the operand "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Selecting the data type of an FBD element (Page 2624)

Inserting operands in FBD instructions (Page 2643)

Adding additional inputs and outputs to FBD elements (Page 2639)

EXPT: Exponentiate

Description

You can use the "Exponentiate" instruction to raise the value at the input IN1 by a power specified with the value at input IN2. The result of the instruction is stored at output OUT and can be queried there (OUT = $IN1^{IN2}$).

The value at input IN1 must be a valid floating-point number. Integers are also allowed for setting the input IN2.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- Errors occur during the instruction processing, for example, if there is an overflow.

Parameters

The following table shows the parameters of the instruction "Exponentiate":

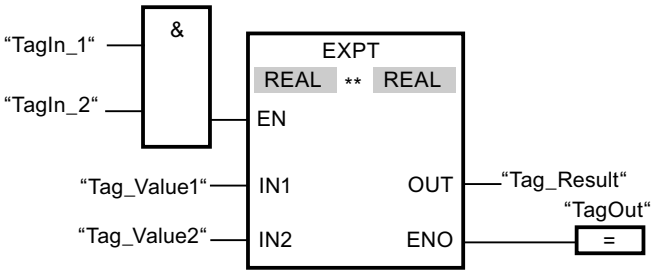
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN1	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Base value
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Value with which the base value is exponentiated
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Result

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the operands "TagIn_1" and "TagIn_2" have signal state "1", the "Exponentiate" instruction is executed. The value of operand "Tag_Value1" is raised by the power of the value of the operand "Tag_Value2". The result is stored at output "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Selecting the data type of an FBD element (Page 2624)

Inserting operands in FBD instructions (Page 2643)

Adding additional inputs and outputs to FBD elements (Page 2639)

Move operations

MOVE: Move value

Description

You use the "Move value" instruction to transfer the content of the operand at the IN input to the operand at the OUT1 output. The transfer is always made in the direction of ascending address.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The data type at the IN parameter does not correspond to the specified data type at the OUT1 parameter.

The following table lists the possible transfers for the S7-1200 CPU series:

Source (IN)	Destination (OUT1)	
	With IEC check	Without IEC check
BYTE	BYTE, WORD, DWORD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE ,TOD, CHAR
WORD	WORD, DWORD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD, CHAR
DWORD	DWORD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME, DATE, TOD, CHAR
SINT	SINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
USINT	USINT, UINT, UDINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
INT	INT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
UINT	UINT, UDINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
DINT	DINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
UDINT	UDINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
REAL	REAL	DWORD, REAL
LREAL	LREAL	LREAL
TIME	TIME	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME
DATE	DATE	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, DATE

Source (IN)	Destination (OUT1)	
	With IEC check	Without IEC check
TOD	TOD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TOD
DTL	DTL	DTL
CHAR	CHAR	BYTE, WORD, DWORD, CHAR, Character of a string[1)]
Character of a string[1)]	Character of a string	Character of a string CHAR
ARRAY[2)]	ARRAY	ARRAY
STRUCT	STRUCT	STRUCT
PLC data type (UDT)	PLC data type (UDT)	PLC data type (UDT)
IEC_TIMER	IEC_TIMER	IEC_TIMER
IEC_SCOUNTER	IEC_SCOUNTER	IEC_SCOUNTER
IEC_USCOUNTER	IEC_USCOUNTER	IEC_USCOUNTER
IEC_COUNTER	IEC_COUNTER	IEC_COUNTER
IEC_UCOUNTER	IEC_UCOUNTER	IEC_UCOUNTER
IEC_DCOUNTER	IEC_DCOUNTER	IEC_DCOUNTER
IEC_UDCOUNTER	IEC_UDCOUNTER	IEC_UDCOUNTER

The following table lists the possible transfers for the S7-1500 CPU series:

Source (IN)	Destination (OUT1)	
	With IEC check	Without IEC check
BYTE	BYTE, WORD, DWORD, LWORD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE ,TOD, LTOD, CHAR
WORD	WORD, DWORD, LWORD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, S5TIME, TIME, LTIME, LDT, DATE, TOD, LTOD, CHAR
DWORD	DWORD, LWORD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, TIME, LTIME, LDT, DATE, TOD, LTOD, CHAR
LWORD	LWORD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, LREAL, TIME, LTIME, LDT, DATE, TOD, LTOD, CHAR
SINT	SINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
USINT	USINT, UINT, UDINT, ULINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD

Source (IN)	Destination (OUT1)	
	With IEC check	Without IEC check
INT	INT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
UINT	UINT, UDINT, ULINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD. LTOD
DINT	DINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
UDINT	UDINT, ULINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
LINT	LINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
ULINT	ULINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
REAL	REAL	DWORD, REAL
LREAL	LREAL	LWORD, LREAL
S5TIME	S5TIME	WORD, S5TIME
TIME	TIME	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME
LTIME	LTIME	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, LTIME
DATE	DATE	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, DATE
DT	DT	DT
LDT	LDT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, LDT
TOD	TOD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TOD
LTOD	LTOD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, LTOD
DTL	DTL	DTL
CHAR	CHAR	BYTE, WORD, DWORD, LWORD, CHAR, Character of a string[1)]
WCHAR	WCHAR	BYTE, WORD, DWORD, LWORD, CHAR, WCHAR, Character of a string[1)]
Character of a string[1)]	Character of a string	CHAR, WCHAR, Character of a string
ARRAY[2)]	ARRAY	ARRAY
STRUCT	STRUCT	STRUCT
COUNTER	COUNTER, WORD, INT	WORD, DWORD, INT, UINT, DINT, UDINT
TIMER	TIMER, WORD, INT	WORD, DWORD, INT, UINT, DINT, UDINT
PLC data type (UDT)	PLC data type (UDT)	PLC data type (UDT)

Source (IN)	Destination (OUT1)	
	With IEC check	Without IEC check
IEC_TIMER	IEC_TIMER	IEC_TIMER
IEC_LTIMER	IEC_LTIMER	IEC_LTIMER
IEC_SCOUNTER	IEC_SCOUNTER	IEC_SCOUNTER
IEC_USCOUNTER	IEC_USCOUNTER	IEC_USCOUNTER
IEC_COUNTER	IEC_COUNTER	IEC_COUNTER
IEC_UCOUNTER	IEC_UCOUNTER	IEC_UCOUNTER
IEC_DCOUNTER	IEC_DCOUNTER	IEC_DCOUNTER
IEC_UDCOUNTER	IEC_UDCOUNTER	IEC_UDCOUNTER
IEC_LCOUNTER	IEC_LCOUNTER	IEC_LCOUNTER
IEC_ULCOUNTER	IEC_ULCOUNTER	IEC_ULCOUNTER

1) You can also use the "Move value" instruction to transfer individual characters of a string to operands of the CHAR or WCHAR data type. The number of the character to be transferred is specified in square brackets next to the operand name. "MyString[2]", for example, transfers the second character of the "MyString" string. It is also possible to transfer from operands of the data type CHAR to the individual characters of a string. You can also replace a specific character of a string with the character of another string.

2) Transferring entire arrays (ARRAY) is possible only when the array components of the operands at input IN and at output OUT1 are of the same data type.

If the bit length of the data type at input IN exceeds the bit length of the data type at output OUT1, the higher-order bits of the source value are lost. If the bit length of the data type at input IN is less than the bit length of the data type at output OUT1, the higher-order bits of the destination value will be overwritten with zeros.

In its initial state, the instruction box contains 1 output (OUT1). The number of outputs can be extended. The added outputs are numbered in ascending order on the box. During the execution of the instruction, the content of the operand at the input IN is transferred to all available outputs. The instruction box cannot be extended if structured data types (DTL, STRUCT, ARRAY) or characters of a string are transferred.

You can also use the "Move block" (MOVE_BLK) and "Move block uninterruptible" (UMOVE_BLK) instructions to move operands of the ARRAY data type. You can move operands of the STRING data type with the instruction "Move character string" (S_MOVE).

Parameters

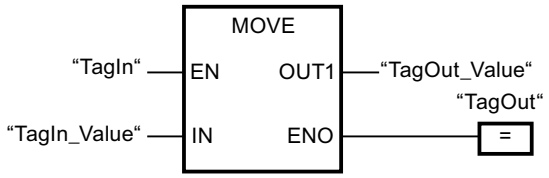
The following table lists the parameters of the "Move value" instruction:

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers, floating-point numbers, DATE, TIME, TOD, DTL, CHAR, STRUCT, ARRAY, IEC data types, PLC data type (UDT)	Bit strings, integers, floating-point numbers, DATE, DT, LDT, S5TIME, TIME, LTIME, TOD, LTOD, DTL, CHAR, STRUCT, ARRAY, TIMER, COUNTER, IEC data types, PLC data type (UDT)	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Element used to overwrite the destination address.
OUT1	Output	Bit strings, integers, floating-point numbers, DATE, TIME, TOD, DTL, CHAR, STRUCT, ARRAY, IEC data types, PLC data type (UDT)	Bit strings, integers, floating-point numbers, DATE, DT, LDT, S5TIME, TIME, LTIME, TOD, LTOD, DTL, CHAR, STRUCT, ARRAY, TIMER, COUNTER, IEC data types, PLC data type (UDT)	I, Q, M, D, L	I, Q, M, D, L	Destination address

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table lists how the instruction works using specific operand values:

Parameters	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
OUT1	TagOut_Value	0011 1111 1010 1111

If the operand "TagIn" has the signal state "1", the "Move value" instruction is executed. The instruction copies the content of the "TagIn_Value" operand to the "TagOut_Value" operand. If the instruction is executed without errors, the ENO and "TagOut" enable outputs are set to signal state "1".

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Removing instruction inputs and outputs (Page 2640)

MOVE_BLK: Move block (Page 3616)

UMOVE_BLK: Move block uninterruptible (Page 3623)

S_MOVE: Move character string (Page 4519)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Adding additional inputs and outputs to FBD elements (Page 2639)

Deserialize: Deserialize

Description

You can use the "Deserialize" instruction to convert the sequential representation of a PLC data type (UDT) back to a PLC data type and to fill its entire contents.

The memory area in which the sequential representation of a PLC data type is located must have the ARRAY of BYTE data type and be declared with standard access. Make sure that there is enough memory space prior to the conversion.

The instruction enables you to convert multiple sequential representations of converted PLC data types back to their original state step-by-step.

Note

If you only want to convert back a single sequential representation of a PLC data type (UDT), you can also use the instruction "TRCV: Receive data via communication connection" directly.

Parameter

The following table shows the parameters of the "Deserialize" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRC_ARRAY	Input	VARIANT	I, Q, D, L	Global data block in which the generated data stream is stored
DEST_VARIABLE	InOut	VARIANT	I, Q, D, L	Tag in which the returned converted PLC data type (UDT) is stored
POS	InOut	DINT	I, Q, M, D, L	Number of bytes that the converted PLC data types use. The POS parameter is calculated zero-based.
RET_VAL	Output	INT	I, Q, M, D, L	Error information

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

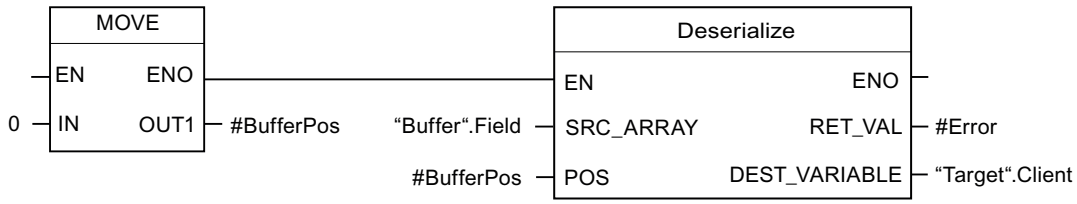
Error code* (W#16#...)	Explanation
0000	No error
80B0	The memory areas for the SRC_ARRAY and DEST_VARIABLE parameters overlap.
8136	The data block at the DEST_VARIABLE parameter is not a block with standard access.
8150	The VARIANT data type at the SRC_ARRAY parameter contains no value.
8151	Code generation error at SRC_ARRAY parameter
8153	There is not enough free memory available at the SRC_ARRAY parameter.
8250	The VARIANT data type at the DEST_VARIABLE parameter contains no value.
8251	Code generation error at DEST_VARIABLE parameter
8254	Invalid data type at the DEST_VARIABLE parameter

Error code* (W#16#...)	Explanation
8382	The value at the POS parameter is outside the limits of the array.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

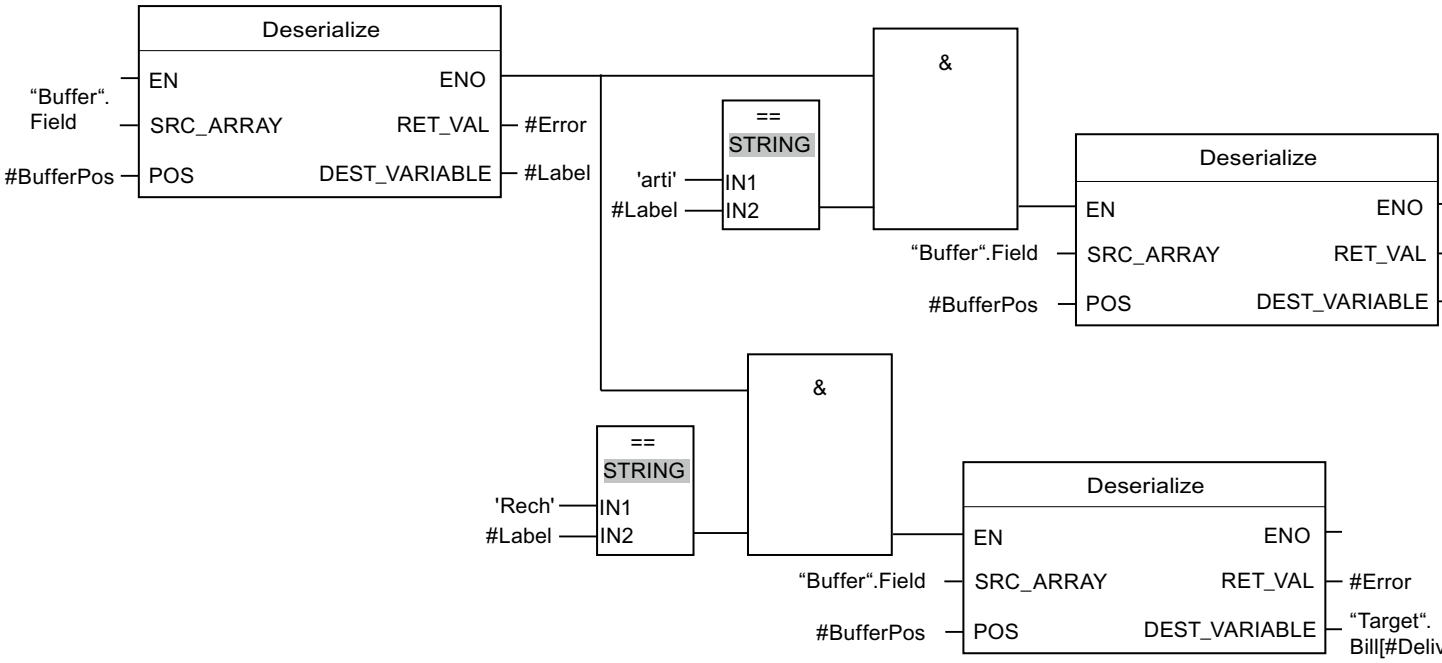
The following example shows how the instruction works:

Network 1:



The "Move value" instruction moves the value "0" to the "#BufferPos" operand. The "Deserialize" instruction deserializes the sequential representation of the customer data from the "Buffer" data block and writes it to the "Target" data block. The number of bytes used by the converted customer data is stored in the "#BufferPos" operand.

Network 2:



The "Deserialize" instruction deserializes the sequential representation of the separator sheet, which was saved with the sequential representation after the customer data, from the "Buffer" data block and writes the characters to the "#Label" operand. The characters are compared using comparison instructions for "arti" and "Bill". If the comparison for "arti" = TRUE, these

are article data that have been deserialized and written to the "Target" data block. If the comparison for "Bill" = TRUE, these are billing data that have been deserialized and written to the "Target" data block.

The following table shows the declaration of the operands:

Operand	Data type	Declaration
DeliverPos	INT	In the "Input" section of the block interface
BufferPos	DINT	In the "Temp" section of the block interface
Error	INT	In the "Temp" section of the block interface
Label	STRING[4]	In the "Temp" section of the block interface

The following table lists the declarations of the PLC data types:

Name of the PLC data types	Name	Data type
Article	Number	DINT
	Declaration	STRING
	Colli	INT
Client	Title	INT
	First name	STRING[10]
	Surname	STRING[10]

The following table shows the declaration of the data blocks:

Name of the data blocks	Name	Data type
Target	Client	"Client" (PLC data type)
	Article	Array[0..] of "Article" (PLC data type)
	Bill	Array[0..10] of INT
Buffer	Field	Array[0..294] of BYTE

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Switching display formats in the program status (Page 2980)

PLC data types (Page 2280)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Serialize: Serialize

Description

You can use the "Serialize" instruction to convert several PLC data types (UDT) to a sequential representation without losing parts of their structure.

You can use the instruction to temporarily save multiple structured data items from your program in a buffer, which should preferably be in a global data block, and send them to another CPU. The memory area in which the converted PLC data types are stored must have the ARRAY of BYTE data type and be declared with standard access. Make sure that there is enough memory space prior to the conversion.

The operand at the POS parameter contains information about the number of bytes used by the converted PLC data types.

Note

If you want to send a single PLC data type (UDT), you can directly call the instruction "TSEND: Send data via communication connection".

Parameters

The following table shows the parameters of the "Serialize" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRC_VARIABLE	Input	VARIANT	I, Q, D, L	PLC data type (UDT) that is converted to sequential representation.
DEST_ARRAY	InOut	VARIANT	I, Q, D, L	Data block in which the generated data stream is stored.
POS	InOut	DINT	I, Q, M, D, L	Number of bytes that the converted PLC data types use. The POS parameter is calculated zero-based.
RET_VAL	Output	INT	I, Q, M, D, L	Error information

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

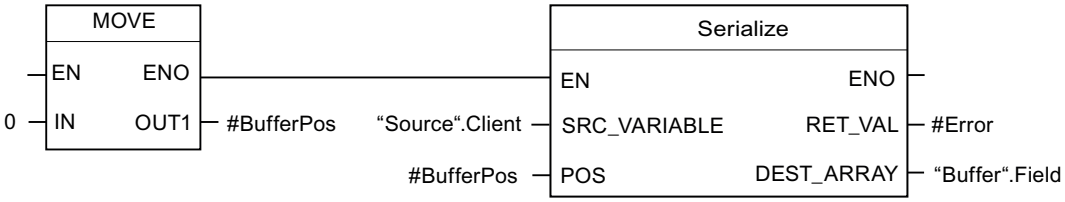
Error code* (W#16#...)	Explanation
0000	No error
80B0	The memory areas for the SRC_VARIABLE and DEST_ARRAY parameters overlap.

Error code* (W#16#...)	Explanation
8150	The VARIANT data type at the SRC_VARIABLE parameter contains no value.
8152	Code generation error at SRC_VARIABLE parameter
8236	The data block at the DEST_ARRAY parameter is not a block with standard access.
8250	The VARIANT data type at the DEST_ARRAY parameter contains no value.
8252	Code generation error at DEST_ARRAY parameter
8253	There is not enough free memory available at the DEST_ARRAY parameter.
8254	Invalid data type at the DEST_ARRAY parameter
8382	The value at the POS parameter is outside the limits of the array.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

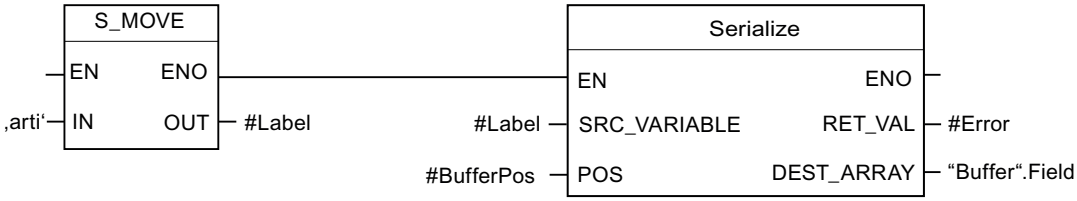
The following example shows how the instruction works:

Network 1:



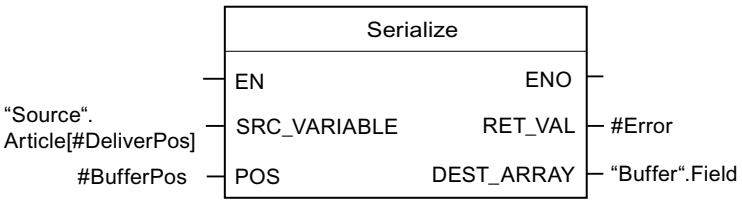
The "Move value" instruction moves the value "0" to the "#BufferPos" operand. The "Serialize" instruction serializes the customer data from the "Source" data block and writes them in sequential representation to the "Buffer" data block. The number of bytes used by the sequential representation is stored in the "#BufferPos" operand.

Network 2:



A kind of separator sheet is inserted now to make it easier to deserialize the sequential representation later. The "Move character string" instruction moves the "arti" characters to the "#Label" operand. The "Serialize" instruction writes these characters after the customer data to the "Buffer" data block. The number of bytes that the characters need is added to the number already stored in the "#BufferPos" operand.

Network 3:



The "Serialize" instruction serializes the data of a specific article, which is calculated in runtime, from the "Source" data block and writes them in sequential representation to the "Buffer" data block after the "arti" characters.

The following table shows the declaration of the operands:

Operand	Data type	Declaration
DeliverPos	INT	In the "Input" section of the block interface
BufferPos	DINT	In the "Temp" section of the block interface
Error	INT	In the "Temp" section of the block interface
Label	STRING[4]	In the "Temp" section of the block interface

The following table lists the declarations of the PLC data types:

Name of the PLC data types	Name	Data type
Article	Number	DINT
	Declaration	STRING
	Colli	INT
Client	Title	INT
	First name	STRING[10]
	Surname	STRING[10]

The following table shows the declaration of the data blocks:

Name of the data blocks	Name	Data type
Source	Client	"Client" (PLC data type)
	Article	Array[0..10] of "Article" (PLC data type)
Buffer	Field	Array[0..294] of BYTE

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Switching display formats in the program status (Page 2980)

PLC data types (Page 2280)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

FieldRead: Read field

Description

You can use the "Read field" instruction to read out a specific component from the field specified in the MEMBER parameter and transfer its content to the tag in the VALUE parameter. You use the parameter INDEX to define the index of the field components that are to be read. At the parameter MEMBER you specify the first component of the field to be read.

The data types of the field component at parameter MEMBER and the tags at parameter VALUE must correspond to the data type of the instruction "Read field".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The field component specified at the parameter INDEX is not defined in the field specified at the parameter MEMBER.
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the instruction "Read field":

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
INDEX	Input	DINT	DINT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Index of field components whose content is read out

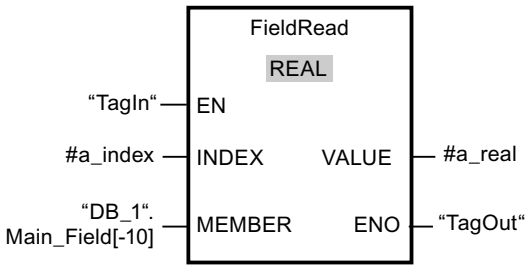
Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
MEMBER	Input	Binary numbers, integers, floating-point numbers, TIME, DATE, TOD and CHAR as components of an ARRAY tag	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, LTOD and CHAR as components of an ARRAY tag	D, L	D, L	First component of the field from which will be read
VALUE	Output	Bit strings, integers, floating-point numbers, TIME, DATE, TOD and CHAR	Bit strings, integers, floating-point numbers, timers, DATE, TOD, LTOD and CHAR	I, Q, M, D, L, P	I, Q, M, D, L, P	Operand to which the contents of the field component are transferred.

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Tag	Value
INDEX	a_index	4
MEMBER	"DB_1".Main_Field[-10]	First component of the field "Main_Field[-10..10] of REAL" in the data block "DB_1"
VALUE	a_real	Component with Index 4 of the field "Main_Field[-10..10] of REAL"

The field component with index 4 is read out from the field "Main_Field[-10...10] of REAL" and written to the tag "a_real". The field component to be read is defined by the value at the parameter INDEX.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Selecting the data type of an FBD element (Page 2624)

FieldWrite: Write field

Description

The "Write field" instruction is used to transfer the content of the tag at the VALUE input to a specific component of the field at the MEMBER output. You use the value at the INDEX input to specify the index of the field component that is described. At the MEMBER output, enter the first component of the field which is to be written to.

The data types of the field component specified at the MEMBER output and the tags at the VALUE input have to match the data type of the "Write field" instruction.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The field component specified at the input INDEX is not defined in the field specified at the output MEMBER.
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the instruction "Write field":

Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output

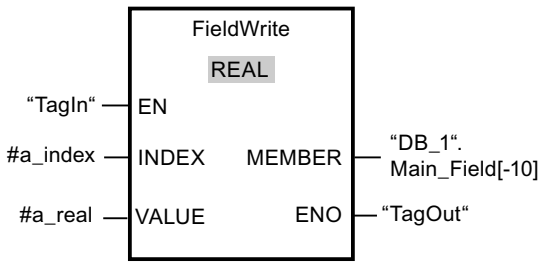
Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
INDEX	Input	DINT	DINT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Index of field component that is written with the content of VALUE
VALUE	Input	Bit strings, integers, floating-point numbers, TIME, DATE, TOD and CHAR	Bit strings, integers, floating-point numbers, timers, DATE, TOD, LTOD and CHAR	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Operand whose contents are copied
MEMBER	Output	Binary numbers, integers, floating-point numbers, TIME, DATE, TOD and CHAR as components of an ARRAY tag	Binary numbers, integers, floating-point numbers, timers, DATE, TOD, LTOD and CHAR as components of an ARRAY tag	D, L	D, L	First component of the field to which the content of VALUE is written.

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
INDEX	a_index	4
VALUE	a_real	10.54
MEMBER	"DB_1".Main_Field[-10]	First component of the field "Main_Field[-10..10] of REAL" in the data block "DB_1"

The value "10.54" of the "a_real" tag is written to the field component with index 4 of the "Main_Field[-10 ... 10] of REAL" field. The index of the field component to which the content of the tag "a_real" is transferred is specified by the value at the input INDEX.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Selecting the data type of an FBD element (Page 2624)

MOVE_BLK: Move block

Description

You can use the "Move block" instruction to move the content of a memory area (source area) to another memory area (destination area). The number of elements to be moved to the destination area is specified with the COUNT parameter. The width of the elements to be moved is defined by the width of the element at input IN.

The instruction can only be executed if the source area and the destination area are of the same data type.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- More data is moved than is made available at input IN or output OUT.

When a Array of BOOL is copied, the enable output ENO for an overflow is set to "1" until the byte limit of the Array structure is exceeded. If the byte limit of the Array structure is exceeded by the value at the COUNT input, the ENO enable output is reset to "0".

Parameters

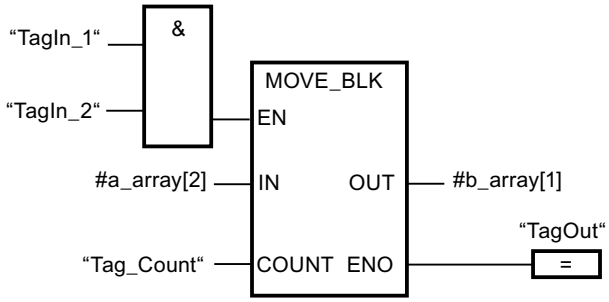
The following table lists the parameters of the "Move block" instruction:

Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN [1]	Input	Binary numbers, integers, floating-point numbers, TIME, DATE, CHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, TOD, LTOD	D, L	D, L	The first element of the source area that is being copied
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Number of elements to be copied from the source area to the destination area
OUT [1]	Output	Binary numbers, integers, floating-point numbers, TIME, DATE, CHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, TOD, LTOD	D, L	D, L	The first element of the destination area to which the contents of the source area are being copied
[1] The specified data types can only be used as elements of an Array structure.						

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	a_array[2]	The data type of the a_array operand is Array [0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the b_array operand is Array [0..6] of INT. It consists of seven elements of data type INT.

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the "Move block" instruction is executed. Starting from the third element, the instruction selects three INT elements from the #a_array tag and copies their contents to the #b_array output tag, beginning with the second element. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

MOVE_BLK_VARIANT: Move block

Description

You can use the "Move block" instruction to move the content of a memory area (source area) to another memory area (destination area). You can copy elements of an array to another array

of the same data type. The size (number of elements) of the source and destination array may be different. You can copy several elements within an array or individual elements.

If you are using the instruction, the array must not yet be known at the time the block is created, as the source and the destination are transferred using VARIANT.

The counting at the parameters SRC_INDEX and DEST_INDEX always starts with the low limit "0", regardless of the later declaration of the array.

If the VARIANT pointer (source or destination) is of the data type BOOL, it must be addressed absolutely and the length specified must be divisible by 8; otherwise the instruction is not executed.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- More data is copied than is made available.

Parameters

The following table shows the parameters of the "Move block" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRC	Input	VARIANT (array or single element)	L	Source block from which to copy
COUNT	Input	UDINT	I, Q, M, D, L or constant	Number of elements which are copied Set the value at the parameter COUNT to "1", if no Array is specified at the parameter SRC or the parameter DEST.

Parameter	Declaration	Data type	Memory area	Description
SRC_INDEX	Input	DINT	I, Q, M, D, L or constant	• The SRC_INDEX parameter is calculated zero-based. If an Array is specified at parameter SRC, the integer at the SRC_INDEX parameter specifies the first element within the source area from which copying is to take place. This is independent of the declared array limits. • If no Array is set at the SRC parameter or only an individual element of an array is specified, assign the SRC_INDEX parameter the value "0".
DEST_INDEX	Input	DINT	I, Q, M, D, L or constant	• The DEST_INDEX parameter is calculated zero-based. If an Array is specified at the DEST parameter, the integer at the DEST_INDEX parameter specifies the first element within the destination area to which copying is to take place. This is independent of the declared array limits. • If no Array is specified at the DEST parameter, assign the DEST_INDEX parameter the value "0".

Parameter	Declaration	Data type	Memory area	Description
DEST	Output	VARIANT	L	Destination area into which the contents of the source block are copied.
RET_VAL	Output	INT	I, Q, M, D, L	Error information: If an error occurs during the execution of the instruction, an error code is output at the RET_VAL parameter.

For additional information on valid data types, refer to "See also".

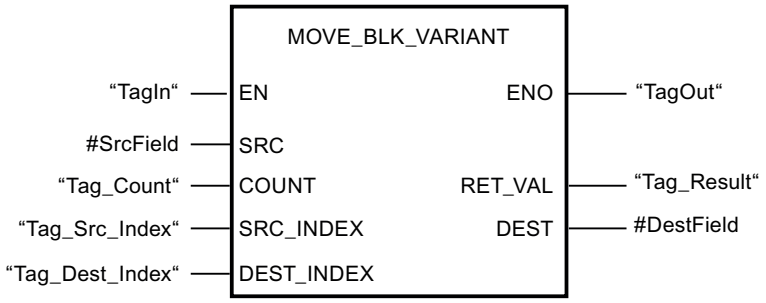
RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	Data types do not correspond
8151	Access to the SRC parameter is not possible.
8152	The operand at the SRC parameter is not typed.
8153	Code generation error at SRC parameter
8154	The operand at the SRC parameter has the data type BOOL.
8281	The COUNT parameter has an invalid value
8382	The value at the SRC_INDEX parameter is outside the limits of the VARIANT.
8383	The value at the SRC_INDEX parameter is outside the high limit of the array.
8482	The value at the DEST_INDEX parameter is outside the limits of the VARIANT.
8483	The value at the DEST_INDEX parameter is outside the high limit of the array.
8534	The DEST parameter is write protected
8551	Access to the DEST parameter is not possible.
8552	The operand at the DEST parameter is not typed.
8553	Code generation error at DEST parameter
8554	The operand at the DEST parameter has the data type BOOL.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
SRC	#SrcField	The local operand #SrcField uses a UDT that was still unknown at the time when the block was programmed. (Array [0 to 10] of "MOVE_UDT"
COUNT	Tag_Count	2
SRC_INDEX	Tag_Src_Index	3
DEST_INDEX	Tag_Dest_Index	3
DEST	#DestField	The local operand #DestField uses a UDT that was still unknown at the time when the block was programmed. (Array [10 to 20] of "MOVE_UDT"

If the operand "TagIn" has the signal state "1", the "Move block" instruction is executed. 2 elements are copied from the source area into the destination area beginning with the fourth element of the array of UDT. The copies are inserted in the array of UDT beginning with the fourth element. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Switching display formats in the program status (Page 2980)

VariantGet: Read out VARIANT tag value (Page 3642)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

UMOVE_BLK: Move block uninterruptible

Description

You can use the "Move block uninterruptible" instruction to move the content of a memory area (source area) to another memory area (destination area). The number of elements to be moved to the destination area is specified with the COUNT parameter. The width of the elements to be moved is defined by the width of the element at input IN.

The instruction can only be executed if the source area and the destination area have the same data type.

Note

The move operation cannot be interrupted by other operating system activities. This is why the alarm reaction times of the CPU increase during the execution of the "Move block uninterruptible" instruction.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- More data is moved than is made available at input IN or output OUT.

When a Array of BOOL is copied, the enable output ENO for an overflow is set to "1" until the byte limit of the Array structure is exceeded. If the byte limit of the Array structure is exceeded by the value at the COUNT input, the ENO enable output is reset to "0".

You can use the "Move block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Parameters

The following table shows the parameters of the "Move block uninterruptible" instruction:

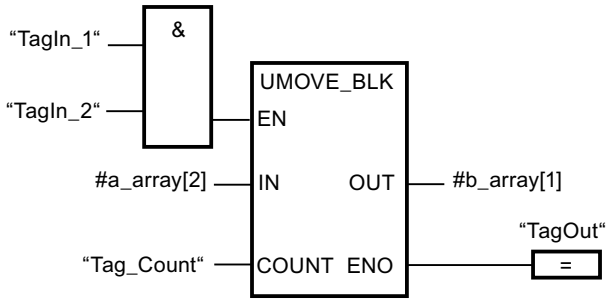
Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN [1)]	Input	Binary numbers, integers, floating-point numbers, TIME, DATE, CHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, TOD, LTOD	D, L	D, L	The first element of the source area that is being copied

Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Number of elements to be copied from the source area to the destination area
OUT [1]	Output	Binary numbers, integers, floating-point numbers, TIME, DATE, CHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, TOD, LTOD	D, L	D, L	The first element of the destination area to which the contents of the source area are being copied
[1] The specified data types can only be used as elements of an array structure.						

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	a_array[2]	The data type of the a_array operand is Array [0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the b_array operand is Array [0..6] of INT. It consists of seven elements of data type INT.

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the "Move block uninterruptible" instruction is executed. Starting from the third element, the instruction selects three INT elements from the #a_array tag and copies their contents to the #b_array output tag, beginning with the second element. The move operation cannot be interrupted by other operating system activities. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

FILL_BLK: Fill block

Description

You can use the "Fill block" instruction to fill a memory area (destination area) with the value of input IN. The destination area is filled beginning with the address specified at the OUT output. The number of repeated move operations is specified with the COUNT parameter. When the instruction is executed, the value at the input IN is selected and moved to the destination area as often as specified by the value of the COUNT parameter.

The instruction can only be executed if the source area and the destination area are of the same data type.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- More data is moved than is made available at input IN or output OUT.

When a Array of BOOL is copied, the enable output ENO for an overflow is set to "1" until the byte limit of the Array structure is exceeded. If the byte limit of the Array structure is exceeded by the value at the COUNT input, the ENO enable output is reset to "0".

Parameters

The following table shows the parameters of the "Fill block" instruction:

Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output

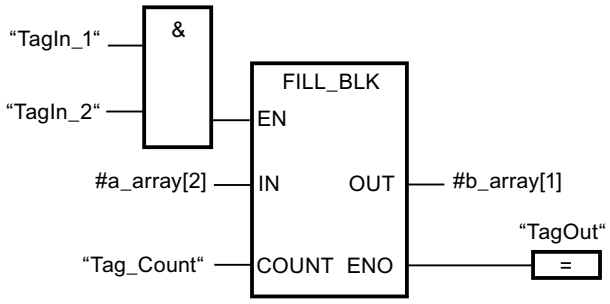
Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
IN [1]	Input	Binary numbers, integers, floating-point numbers, TIME, DATE, TOD, CHAR	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, TOD, LTOD	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Element used to fill the destination area
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Number of repeated move operations
OUT [2]	Output	Binary numbers, integers, floating-point numbers, TIME, DATE, TOD, CHAR	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, TOD, LTOD	D, L	D, L	Address in destination area from which filling starts.

[1] The specified data types can also be used as elements of an Array structure.
[2] The specified data types can only be used as elements of an Array structure.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	a_array[2]	The data type of the a_array operand is Array [0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the b_array operand is Array [0..6] of INT. It consists of seven elements of data type INT.

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the "Fill block" instruction is executed. The instruction copies the third element (#a_array[2]) of the #a_array tag three times to the #b_array output tag. If no errors occur during the execution of the instruction, the outputs ENO and "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

UFILL_BLK: Fill block uninterruptible

Description

You can use the instruction "Fill block uninterruptible" to fill a memory area (destination area) with the value of input IN uninterruptibly. The destination area is filled beginning with the address specified at the OUT output. The number of repeated move operations is specified with the COUNT parameter. When the instruction is executed, the value at the input IN is selected and moved to the destination area as often as specified by the value of the COUNT parameter.

The instruction can only be executed if the source area and the destination area have the same data type.

Note

The move operation cannot be interrupted by other operating system activities. This is why the alarm reaction times of the CPU increase during the execution of the "Fill block uninterruptible" instruction.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- More data is moved than is made available at input IN or output OUT.

When a Array of BOOL is copied, the enable output ENO for an overflow is set to "1" until the byte limit of the Array structure is exceeded. If the byte limit of the Array structure is exceeded by the value at the COUNT input, the ENO enable output is reset to "0".

You can use the "Fill block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Parameters

The following table shows the parameters of the "Fill block uninterruptible" instruction:

Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN [1]	Input	Binary numbers, integers, floating-point numbers, TIME, DATE, CHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, TOD, LTOD	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Element used to fill the destination area
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Number of repeated move operations
OUT [2]	Output	Binary numbers, integers, floating-point numbers, TIME, DATE, CHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, TOD, LTOD	D, L	D, L	Address in destination area from which filling starts.

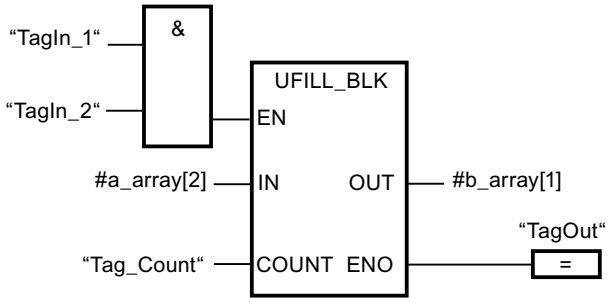
[1] The specified data types can also be used as elements of an Array structure.

[2] The specified data types can only be used as elements of an Array structure.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	a_array[2]	The data type of the a_array operand is Array [0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the b_array operand is Array [0..6] of INT. It consists of seven elements of data type INT.

If the operands "TagIn_1" and "TagIn_2" have signal state "1", the "Fill block uninterruptible" instruction is executed. The instruction copies the third element (#a_array[2]) of the #a_array tag three times to the #b_array output tag. The move operation cannot be interrupted by other operating system activities. If no errors occur during the execution of the instruction, the outputs ENO and "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

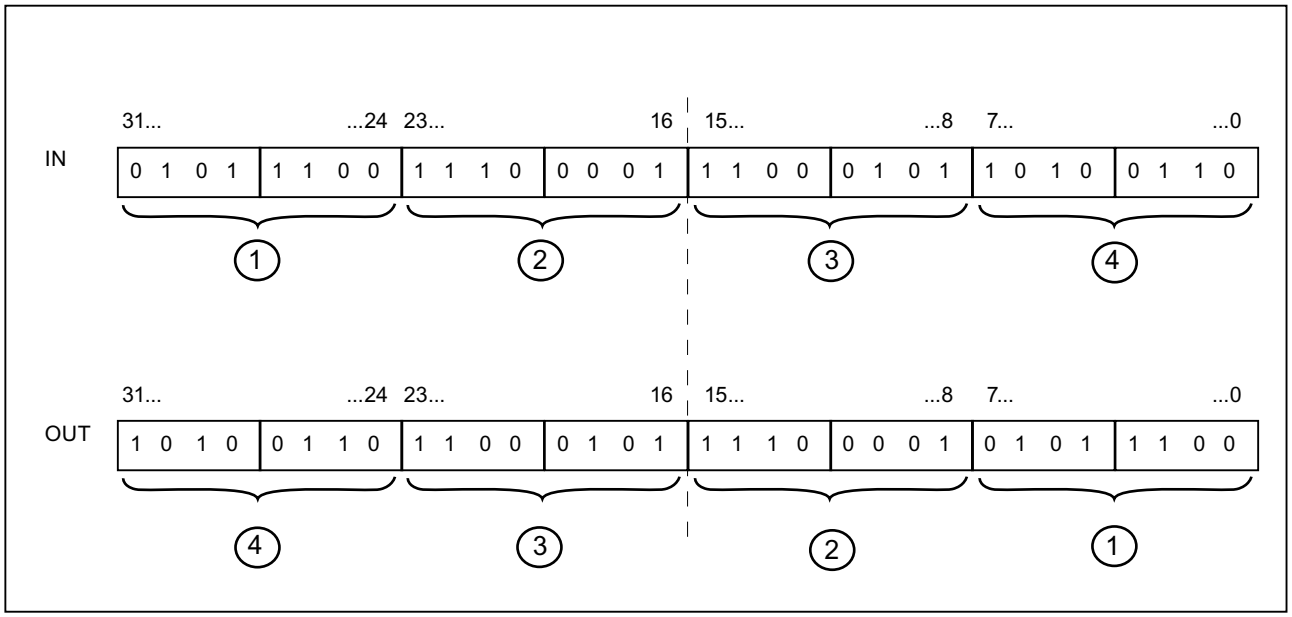
Inserting operands in FBD instructions (Page 2643)

SWAP: Swap

Description

You can use the instruction "Swap" to change the order of the bytes within the tag at input IN and query the result at output OUT.

The following figure shows how the bytes of a DWORD data type operand are swapped using the "Swap" instruction:



Parameters

The following table shows the parameters of the "Swap" instruction:

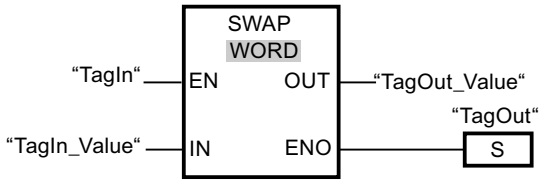
Parameters	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	WORD, DWORD	WORD, DWORD, LWORD	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Operand whose bytes are swapped.
OUT	Output	WORD, DWORD	WORD, DWORD, LWORD	I, Q, M, D, L, P	I, Q, M, D, L, P	Result

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	TagIn_Value	0000 1111 0101 0101
OUT	TagOut_Value	0101 0101 0000 1111

If the operand "TagIn" has the signal state "1", the "Swap" instruction is executed. The arrangement of the bytes is changed and stored in the operand "TagOut_Value".

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Selecting the data type of an FBD element (Page 2624)

Inserting operands in FBD instructions (Page 2643)

ARRAY DB instructions

ReadFromArrayDB: Read from array data block

Description

You can use the "Read from array data block" instruction to read data from an array DB and write them to a destination area.

An array DB is a data block which consists of exactly one array of [data type]. The elements of the array can be UDT data type or any other elementary data type. The counting of the array always starts with the low limit "0", regardless of the later declaration of the array.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- If an error occurs during the execution of the instruction.

Parameters

The following table shows the parameters of the "Read from array data block" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
DB	Input	DB_ANY	I, Q, M, D, L	Data block that is read
INDEX	Input	DINT	I, Q, M, D, L, P or constant	Element that is read
VALUE	Output	VARIANT	L	Value that is read and output
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output at the RET_VAL parameter.

For additional information on valid data types, refer to "See also".

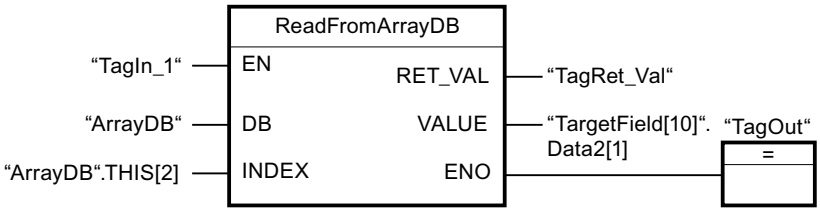
Parameters RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	Data types do not correspond
80B5	Copying process was interrupted
8132	The data block does not exist or is too short
8133	The data block is located in the load memory
8134	The data block is write protected
8135	The array DB contains invalid values
8154	The data type is not correctly declared
8282	The value at the INDEX parameter is outside the limits of the array.
8350	The VARIANT data type contains no value
8352	Code generation error
8353	Code generation error
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
DB	ArrayDB	The data type of the "ArrayDB" operand is Array [0 to 10] of INT.
INDEX	ArrayDB.THIS[2]	Third element of the "ArrayDB"
VALUE	TargetField[10].Data2[1]	The data type of the "TargetField" operand is Array [10 to 20] of "MOVE_UDT".

If the "TagIn1" operand has the signal state "1", the "Read from array data block" instruction is executed. The third element is read from "ArrayDB" and written to the "TargetField[10].Data2[1]" operand. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

WriteToArrayDB: Write to array data block

Description

You can use the "Write to array data block" instruction to write data to an array DB.

An array DB is a data block which consists of exactly one array of [data type]. The elements of the array can be UDT data type or any other elementary data type. The counting of the array always starts with the low limit "0", regardless of the later declaration of the array.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- If an error occurs during the execution of the instruction.

Parameters

The following table shows the parameters of the "Write to array data block" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
DB	Input	DB_ANY	I, Q, M, D, L	Data block to which data is written
INDEX	Input	DINT	I, Q, M, D, L, P or constant	Element in the DB to which data is written
VALUE	Input	VARIANT	L	Value to be written
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output at the RET_VAL parameter.

For additional information on valid data types, refer to "See also".

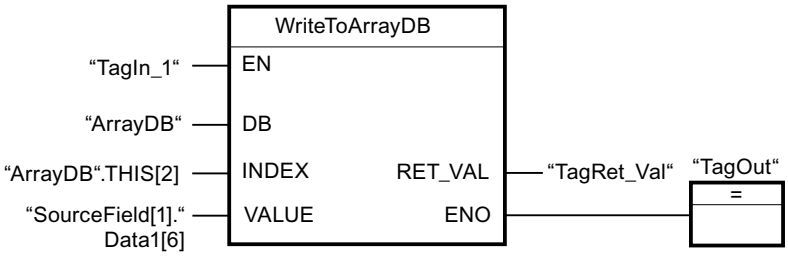
Parameters RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	Data types do not correspond
80B5	Copying process was interrupted
8132	The data block does not exist or is too short
8133	The data block is located in the load memory
8134	The data block is write protected
8135	The DB is not an array data block.
8282	The value at the INDEX parameter is outside the limits of the array.
8350	The VARIANT data type contains no value
8352	Code generation error
8353	Code generation error
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
DB	ArrayDB	The data type of the "ArrayDB" operand is Array [0 to 10] of INT.
INDEX	ArrayDB.THIS[2]	Third element of the "ArrayDB"
VALUE	SourceField[1].Data1[6]	The data type of the "SourceField" operand is Array [0 to 10] of "MOVE_UDT".

If the "TagIn1" operand has the signal state "1", the "Write to the ARRAY data block" instruction is executed. From the "SourceField" operand, the "Data1[6]" element from the second element is written to the "ArrayDB". The third element is written to the "ArrayDB". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

ReadFromArrayDBL: Read from array data block in load memory

Description

You can use the "Read from array data block in load memory" instruction to read data from an array DB in load memory.

An array DB is a data block which consists of exactly one array of [data type]. The elements of the array can be UDT data type or any other elementary data type. The counting of the array always starts with the low limit "0", regardless of the later declaration of the array.

If the array DB is marked with the "Only store in load memory" block attribute, it is only stored in load memory.

The instruction is executed when a positive signal edge is detected at the REQ parameter. The BUSY parameter then has the signal state "1". The instruction is terminated if a negative signal edge is detected at the BUSY parameter. The DONE parameter has the signal state "1" for one program cycle and the read value is output at the VALUE parameter within this cycle. With all other program cycles, the value at the VALUE parameter is not changed.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, it is saved to the "Program resources" folder in the "Program blocks > System blocks" path of the project tree. For additional information on this topic, refer to "See also".

Note

The array DB must be located in a data block with the block property "Optimized".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- If an error occurs during the execution of the instruction.

Parameters

The following table shows the parameters of the "Read from array data block in load memory" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = "1": Starting with the reading of the array DB
DB	Input	DB_ANY	I, Q, M, D, L	Data block that is read
INDEX	Input	DINT	I, Q, M, D, L, P or constant	Element that is read
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = "1": The array DB is still being read
DONE	Output	BOOL	I, Q, M, D, L	DONE = "1": The instruction was executed successfully

Parameter	Declaration	Data type	Memory area	Description
VALUE	InOut	VARIANT	D, L	Value that is read and output
ERROR	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output at the ERROR parameter.

For additional information on valid data types, refer to "See also".

ERROR parameter

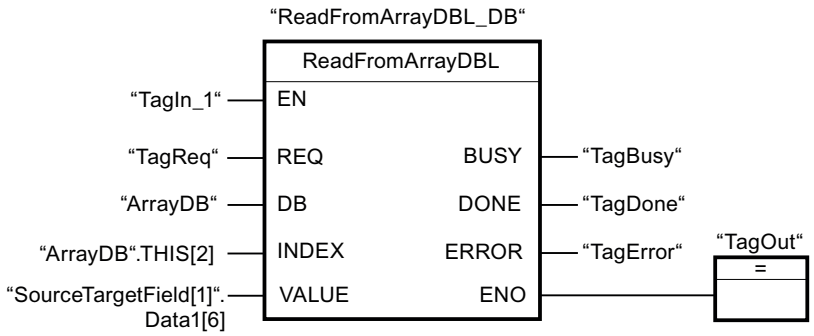
The following table shows the meaning of the values of the ERROR parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	Data types do not correspond
80B5	Copying process was interrupted
80C3	Too many concurrent calls
8132	The data block does not exist or is too short
8133	The data block is not located in load memory
8134	The data block is write protected
8135	The DB is not an array data block.
8282	The value at the INDEX parameter is outside the limits of the array.
8350	The VARIANT data type contains no value
8352	Code generation error
8353	Code generation error
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

You can find descriptions of the error codes triggered by the instructions "READ_DBL: Read from data block in load memory" and "WRIT_DBL: Write to data block in load memory" under "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
REQ	TagReq	BOOL
DB	ArrayDB	The data type of the "ArrayDB" operand is Array [0 to 10] of INT.
INDEX	ArrayDB.THIS[2]	Third element of the "ArrayDB"
VALUE	SourceTargetField[1].Data1[6]	The data type of the "SourceTargetField" operand is Array [0 to 10] of "MOVE_UDT".
BUSY	TagBusy	BOOL
DONE	TagDone	BOOL

If the "TagIn1" operand has the signal state "1" and a positive edge is detected at the "TagReq"operand, the "Read from ARRAY data block in load memory" instruction is executed. The third element is read from "ArrayDB" and written to the "SourceTargetField[1].Data1[6]" operand. As soon as a negative signal edge is detected at the "TagBusy" operand, the instruction is terminated and the value at the VALUE parameter is no longer changed. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set. After the instruction has been processed, the "TagDone" operand the signal state "1".

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

WRIT_DBL: Write to data block in the load memory (Page 4803)

READ_DBL: Read from data block in the load memory (Page 4801)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

WriteToArrayDBL: Write to array data block in load memory

Description

You can use the "Write to array data block in load memory" instruction to write data to an array DB in load memory.

An array DB is a data block which consists of exactly one array of [data type]. The elements of the array can be UDT data type or any other elementary data type. The counting of the array always starts with the low limit "0", regardless of the later declaration of the array.

If the array DB is marked with the "Only store in load memory" block attribute, it is only stored in load memory.

The instruction is executed when a positive signal edge is detected at the REQ parameter. The BUSY parameter then has the signal state "1". If a negative signal edge is detected at the BUSY parameter, the instruction is terminated and the value at the VALUE parameter is written to the data block. The DONE parameter then has the signal state "1" for one program cycle.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, it is saved to the "Program resources" folder in the "Program blocks > System blocks" path of the project tree. For additional information on this topic, refer to "See also".

Note

The array DB must be located in a data block with the block property "Optimized".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- If an error occurs during the execution of the instruction.

Parameters

The following table shows the parameters of the "Write to array data block in load memory" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = "1": Start writing to the array DB
DB	Input	DB_ANY	D	Data block to which data is written
INDEX	Input	DINT	I, Q, M, D, L, P or constant	Element in the DB to which data is written

Parameter	Declaration	Data type	Memory area	Description
VALUE	Input	VARIANT	D, L	Value to be written
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = "1": The array DB is still being written
DONE	Output	BOOL	I, Q, M, D, L	DONE = "1": The instruction was executed successfully
ERROR	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output at the ERROR parameter.

For additional information on valid data types, refer to "See also".

ERROR parameter

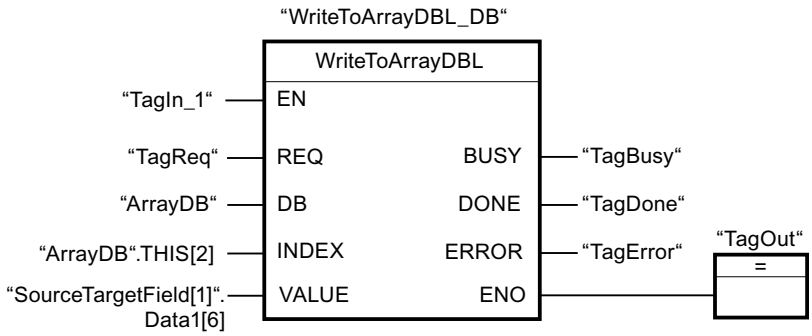
The following table shows the meaning of the values of the ERROR parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	Data types do not correspond
80B5	Copying process was interrupted
8132	The data block does not exist or is too short
8133	The data block is not located in the load memory
8134	The data block is write protected
8135	The DB is not an array data block.
8282	The value at the INDEX parameter is outside the limits of the array.
8350	The VARIANT data type contains no value
8352	Code generation error
8353	Code generation error
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

You can find descriptions of the error codes triggered by the instructions "READ_DBL: Read from data block in load memory" and "WRIT_DBL: Write to data block in load memory" under "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
REQ	TagReq	BOOL
DB	ArrayDB	The data type of the "ArrayDB" operand is Array [0 to 10] of INT.
INDEX	ArrayDB.THIS[2]	Third element of the "ArrayDB"
VALUE	SourceTargetField[1].Data1[6]	The data type of the "SourceTargetField" operand is Array [0 to 10] of "MOVE_UDT".
BUSY	TagBusy	BOOL
DONE	TagDone	BOOL

If the "TagIn1" operand has the signal state "1" and a positive edge is detected at the "TagReq"operand, the "Write to ARRAY data block in load memory" instruction is executed. As soon as a negative signal edge is detected at the "TagBusy" operand, the instruction is terminated and the value at the VALUE parameter is written in the third element in the "ArrayDB". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set. After the instruction has been processed, the "TagDone" operand the signal state "1".

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

WRIT_DBL: Write to data block in the load memory (Page 4803)

READ_DBL: Read from data block in the load memory (Page 4801)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

VARIANT instructions

VariantGet: Read out VARIANT tag value

Description

You can use the "Read out VARIANT tag value" instruction to read the value of the tag to which the VARIANT at the SRC parameter points and write it in the tag at the DST parameter.

The SRC parameter must have the VARIANT data type. Any data type except for VARIANT can be specified at the DST parameter.

The data type of the tag at the DST parameter must match the data type to which the VARIANT points.

Note

The tag to which the VARIANT at parameter SRC points must be a PLC data type.

To copy structures and ARRAYs, you can use the "MOVE_BLK_VARIANT: Move block" instruction. For additional information, refer to "See also".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The data types do not correspond. (No value is transferred.)

Parameters

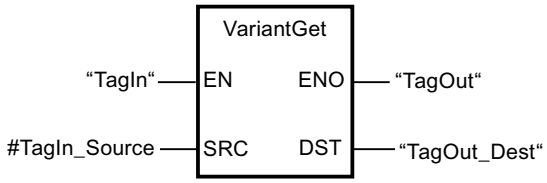
The following table shows the parameters of the "Read out VARIANT tag value" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRC	Input	VARIANT	L	Tag to be read
DST	Output	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the "TagIn" operand has the signal state "1", the "Read out VARIANT tag value" instruction is executed. The value of the tag to which the VARIANT at the "#TagIn_Source" operand points is read and written to the "TagOut_Dest" operand.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Switching display formats in the program status (Page 2980)

VARIANT (Page 2277)

MOVE_BLK_VARIANT: Move block (Page 3618)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Selecting the data type of an FBD element (Page 2624)

Inserting operands in FBD instructions (Page 2643)

VariantPut: Write VARIANT tag value

Description

You can use the "Write VARIANT tag value" instruction to write the value of the tag at the SRC parameter to the tag at the DST parameter to which the VARIANT points.

The DST parameter must have the VARIANT data type. Any data type except for VARIANT can be specified at the SRC parameter.

The data type of the tag at the SRC parameter must match the data type to which the VARIANT points.

Note

The tag to which the VARIANT at parameter DST points must be a PLC data type.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The data types do not correspond. (No value is transferred.)

Parameters

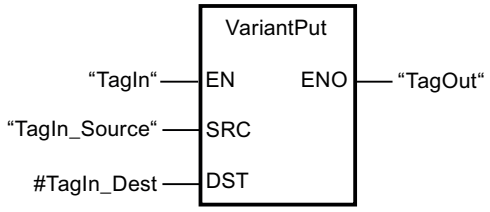
The following table shows the parameters of the "Write VARIANT tag value" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRC	Input	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY	I, Q, M, D, L	Tag to be read
DST	Input	VARIANT	L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the "TagIn" operand has the signal state "1", the "Write VARIANT tag value" instruction is executed. The value of the "TagIn_Source" operand is written to the tag to which the VARIANT at the "#TagIn_Dest" operand points.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Switching display formats in the program status (Page 2980)

VARIANT (Page 2277)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Selecting the data type of an FBD element (Page 2624)

Inserting operands in FBD instructions (Page 2643)

CountOfElements: Get number of ARRAY elements

Description

You can use the "Get number of ARRAY elements" instruction to query how many ARRAY elements a tag pointing to an VARIANT has.

If it is a one-dimensional ARRAY, the difference between the high and low limit +1 is output as a result. If it is a multi-dimensional ARRAY, the product of all dimensions is output as the result.

The IN parameter must have the same VARIANT data type.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The VARIANT tag is not an ARRAY. (The result is "0".)

If the VARIANT points to an ARRAY of BOOL, the fill elements are included in the count. (For example, 8 is returned for an ARRAY[0..1] of BOOL)

Parameters

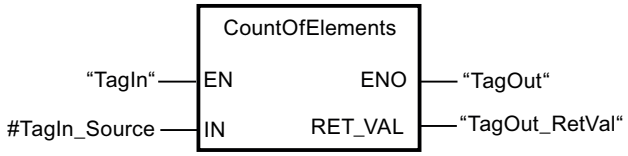
The following table shows the parameters of the "Get number of ARRAY elements" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	VARIANT	L	Tag to be queried
RET_VAL	Output	UDINT	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the "TagIn" operand has the signal state "1", the "Get number of ARRAY elements" instruction is executed. The number of ARRAY elements of the tag to which the VARIANT at the "#TagIn_Source" operand points is read and output at the "TagOut_RetVal" operand.

See also

Format of array (16-bit limits) (Page 2267)

Format of array (32-bit limits) (Page 2268)

Overview of the valid data types (Page 2236)
Basics of the EN/ENO mechanism (Page 2447)
Switching display formats in the program status (Page 2980)
VARIANT (Page 2277)
Inserting FBD elements (Page 2621)
Editing FBD elements (Page 2635)
Selecting the data type of an FBD element (Page 2624)
Inserting operands in FBD instructions (Page 2643)

Other

BLKMOV: Move block

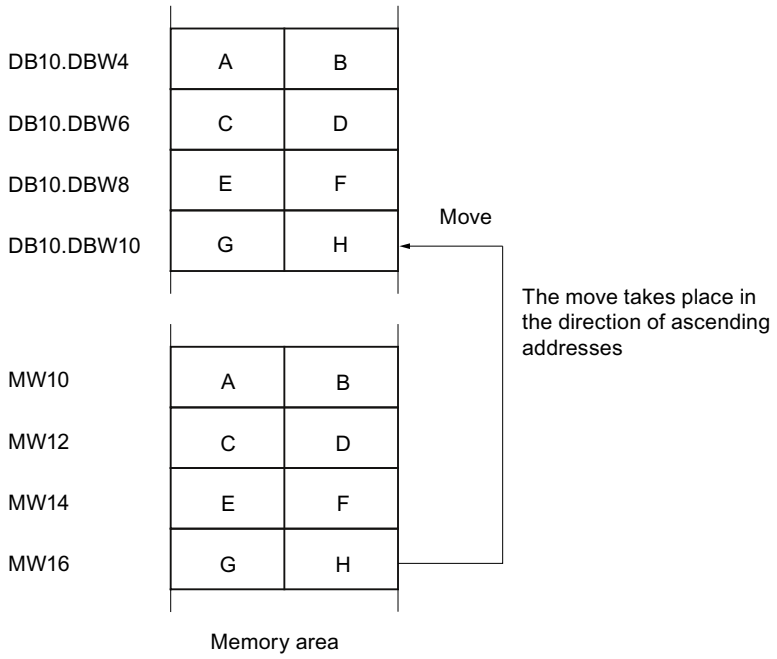
Description

You can use the "Move block" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use VARIANT to define the source and destination area.

Note

The tags of the instruction can only be used in data blocks in which the "Optimized block access" attribute is not set. If the tag with the retentive setting "Set in IDB" has been declared, you can also use the tags "with optimized access".

The following figure shows the principle of the move operation:



Consistency of the source data and the target data

Make sure that the source data remain unchanged during the execution of the "Move block" instruction. Otherwise the consistency of the target data cannot be ensured.

Interruptibility

There is no limit to the nesting depth.

Memory areas

You can use the "Move block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image of the inputs
- Process image of the outputs

General rules for moving

The source and destination area must not overlap. If the source and destination area have different lengths, only the length of the smaller area will be moved.

If the source area is less than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is less than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a block of data type BOOL is moved, the tag must be addressed absolutely and the specified length of the area must be divisible by 8, otherwise the instruction cannot be executed.

Rules for moving character strings

You can use the "Move block" instruction to also move source and destination areas of the STRING data type. If only the source area is STRING data type, the characters will be moved that are actually contained in the character string. Information on the actual and maximum length is also written to the destination area. If the source and destination area are each STRING data type, the current length of the character string in the destination area is set to the number of actually moved characters.

If you want to move the information on the maximum and actual length of a character string, specify the areas in bytes in the SRCBLK and DSTBLK parameters.

Parameters

The following table shows the parameters of the "Move block" instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
SRCBLK	Input	VARIANT	L, P	L, P	Specifies the memory area to be moved (source area).
RET_VAL	Output	INT	I, Q, M, D, L, P	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output at the RET_VAL parameter.
DSTBLK	Output	VARIANT	L, P	L, P	Specifies the memory area to which the block is to be moved (destination area).

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8152	The data types WSTRING, WCHAR and BOOL are not supported.
8092	The source or target area is only available in the load memory.

Error code* (W#16#...)	Explanation
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Evaluating errors with output parameter RET_VAL (Page 5370)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Switching display formats in the program status (Page 2980)

GET_ERR_ID: Get error ID locally (Page 3692)

UBLKMOV: Move block uninterruptible

Description

You can use the "Move block uninterruptible" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use VARIANT to define the source and destination area.

The move operation cannot be interrupted by other operating system activities. As a result the interrupt reaction time of the CPU can increase during the execution of the "Move block uninterruptible" instruction.

Note

The tags of the instruction can only be used in data blocks in which the "Optimized block access" attribute is not set. If the tag with the retentive setting "Set in IDB" has been declared, you can also use the tags "with optimized access".

Memory areas

You can use the "Move block uninterruptible" instruction to move the following memory areas:

- Areas of a data block
- Bit memory

- Process image of the inputs
- Process image of the outputs

General rules for moving

The source and destination area must not overlap during the execution of the "Move block uninterruptible" instruction. If the source area is less than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is less than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a source or destination area defined as a formal parameter is less than a destination or source area specified on the SRCBLK or DSTBLK parameter, no data is transferred.

If a block of data type BOOL is moved, the tag must be addressed absolutely and the specified length of the area must be divisible by 8, otherwise the instruction cannot be executed.

You can use the "Move block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Rules for moving character strings

You can use the "Move block uninterruptible" instruction to also move source and destination areas of the STRING data type. If only the source area is STRING data type, the characters will be moved that are actually contained in the character string. Information on the actual and maximum length are not written in the destination area. If the source and destination area are each STRING data type, the current length of the character string in the destination area is set to the number of actually moved characters. If areas of the data type STRING are moved, you have to specify "1" as area length.

Parameters

The following table shows the parameters of the "Move block uninterruptible" instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
SRCBLK	Input	VARIANT	L, P	L, P	Specifies the memory area to be moved (source area).

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
RET_VAL	Output	INT	I, Q, M, D, L, P	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output at the RET_VAL parameter.
DSTBLK	Output	VARIANT	L, P	L, P	Specifies the memory area to which the block is to be moved (destination area).

Parameters RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code (W#16#...)	Explanation
0000	No error
8152	The data types WSTRING, WCHAR and BOOL are not supported.
8091	The source area is in a data block that is not relevant for program execution.
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Evaluating errors with output parameter RET_VAL (Page 5370)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Switching display formats in the program status (Page 2980)

GET_ERR_ID: Get error ID locally (Page 3692)

FILL: Fill block

Description

You can use the "Fill block" instruction to fill a memory area (destination area) with the content of another memory area (source area). The "Fill block" instruction moves the content of the source area to the destination area until the destination area is completely written. The move operation takes place in the direction of ascending addresses.

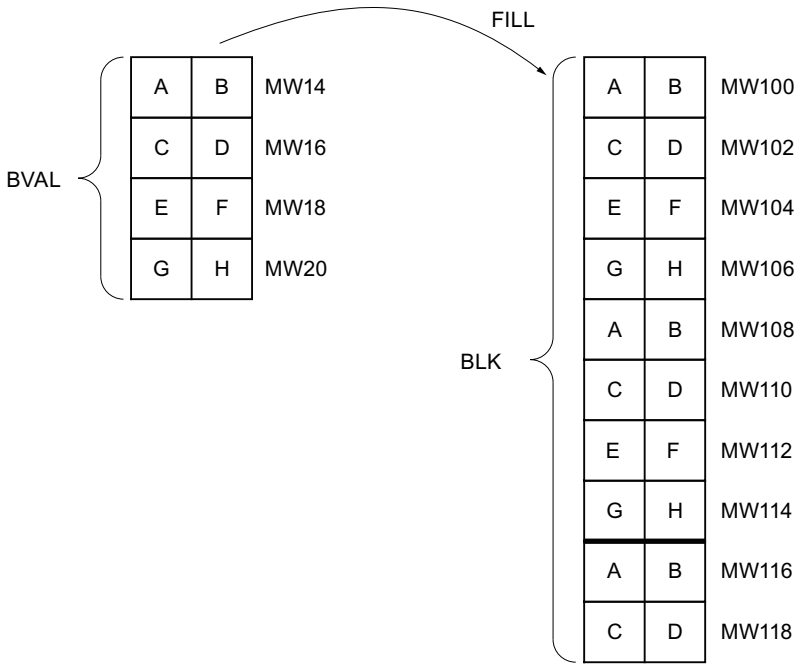
You use VARIANT to define the source and destination area.

Note

The tags of the instruction can only be used in data blocks in which the "Optimized block access" attribute is not set. If the tag with the retentive setting "Set in IDB" has been declared, you can also use the tags "with optimized access".

For blocks with the "Optimized block access" attribute, you can use the "FILL_BLK: Fill block" instruction.

The following figure shows the principle of the move operation:



Consistency of the source data and the target data

Note that while the instruction "Fill block" is being executed, the source data remains unchanged; otherwise the consistency of the destination data cannot be guaranteed.

Memory areas

You can use the "Fill block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image of the inputs
- Process image of the outputs

General rules for moving

The source and destination area must not overlap. If the destination area to be preset is not an integer multiple of the length of the input parameter BVAL, the destination area is nevertheless written up to the last byte.

If the destination area to be preset is smaller than the source area, the function only copies as much data as can be written to the destination area.

If the destination or source area actually present is smaller than the assigned memory area for the source or destination area (BVAL, BLK parameters), no data is transferred.

If the ANY pointer (source or destination) is of the data type BOOL, it must be addressed absolutely and the length specified must be divisible by 8; otherwise the instruction is not executed.

If the destination area is of data type STRING, the instruction writes the entire string including the administration information.

Parameters

The following table shows the parameters of the "Fill block" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
BVAL	Input	VARIANT	L, P	Specification of the memory area (source area) with the content used to fill the destination area at the BLK parameter.
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output at the RET_VAL parameter.
BLK	Output	VARIANT	L, P	Specification of the memory area that is filled with the content of the source area.

Parameters RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8152	The data types WSTRING, WCHAR and BOOL are not supported.
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Evaluating errors with output parameter RET_VAL (Page 5370)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Switching display formats in the program status (Page 2980)

GET_ERR_ID: Get error ID locally (Page 3692)

Conversion operations

CONVERT: Convert value

Description

The "Convert value" instruction reads the content of the IN parameter and converts it according to the data types configured in the instruction box. The converted value is provided at output OUT.

For information on possible conversions, refer to section "Explicit conversion" under "See also".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the "Convert value" instruction:

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers, floating-point numbers, CHAR, BCD16, BCD32	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Value to be converted.
OUT	Output	Bit strings, integers, floating-point numbers, CHAR, BCD16, BCD32	I, Q, M, D, L, P	I, Q, M, D, L, P	Result of the conversion

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

Bit strings (BYTE, WORD, DWORD, LWORD) cannot be selected in the instruction box. If you specify an operand of data type BYTE, WORD, DWORD or LWORD at a parameter of the instruction, it is interpreted according to the data type of the input or output parameter. Data type BYTE is then interpreted as USINT, WORD as UINT, DWORD as UDINT, and LWORD as LINT, for example.

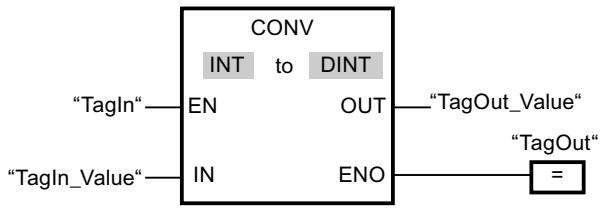
Note

For S7-1500 CPU: You can select the data types DWORD and LWORD if REAL or LREAL were selected as IN data type.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the operand "TagIn" has the signal state "1", the content of the operand "TagIn_Value" is read and converted to an integer (16 bits). The result is stored in the operand "TagOut_Value". The output "TagOut" is set to "1" if the instruction was executed without errors.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Explicit conversion (Page 2391)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Selecting the data type of an FBD element (Page 2624)

ROUND: Round numerical value

Description

You can use the "Round numerical value" instruction to round the value at input IN to the nearest integer. The instruction interprets the value at input IN as a floating-point number and converts this to the nearest integer. If the input value is exactly between an even and odd number, then the even number is converted. The result of the instruction is provided at output OUT and can be queried there.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the "Round numerical value" instruction:

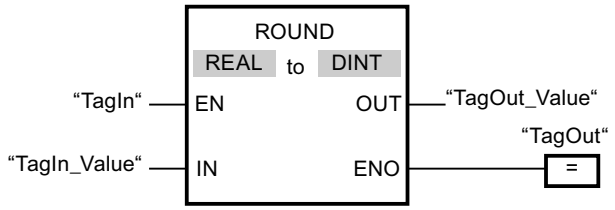
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value to be rounded.
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Result of the rounding

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value	
IN	TagIn_Value	1.50000000	-1.50000000
OUT	TagOut_Value	2	-2

If the operand "TagIn" has the signal state "1", the "Round numerical value" instruction is executed. The floating-point number at input "TagIn_Value" is rounded to the nearest even integer and sent to output "TagOut_Value". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Explicit conversion (Page 2391)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Selecting the data type of an FBD element (Page 2624)

CEIL: Generate next higher integer from floating-point number

Description

You can use the "Generate next higher integer from floating-point number" instruction to round the value at input IN to the next higher integer. The instruction interprets the value at input IN as a floating-point number and converts this to the next higher integer. The result of the instruction is provided at output OUT and can be queried there. The output value can be greater than or equal to the input value.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the instruction "Generate next higher integer from floating-point number":

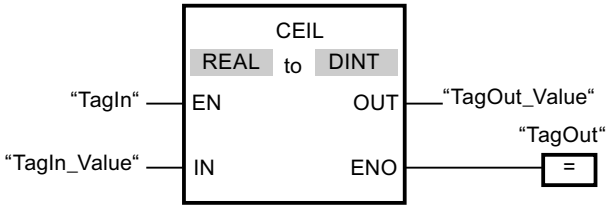
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value as floating-point number
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Result with the next higher integer

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value	
IN	TagIn_Value	0.50000000	-0.50000000
OUT	TagOut_Value	1	0

If the operand "TagIn" has the signal state "1", the instruction "Generate next higher integer from floating-point number" is executed. The floating-point number at input "TagIn_Value" is rounded to the next higher integer and sent to output "TagOut_Value". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 2236)
Basics of the EN/ENO mechanism (Page 2447)
Explicit conversion (Page 2391)
Inserting FBD elements (Page 2621)
Editing FBD elements (Page 2635)
Inserting operands in FBD instructions (Page 2643)
Selecting the data type of an FBD element (Page 2624)

FLOOR: Generate next lower integer from floating-point number

Description

You can use the "Generate next lower integer from floating-point number" instruction to round the value at input IN to the next lower integer. The instruction interprets the value at input IN as a floating-point number and converts this to the next lower integer. The result of the instruction is provided at output OUT and can be queried there. The output value can be less than or equal to the input value.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the instruction "Generate next lower integer from floating-point number":

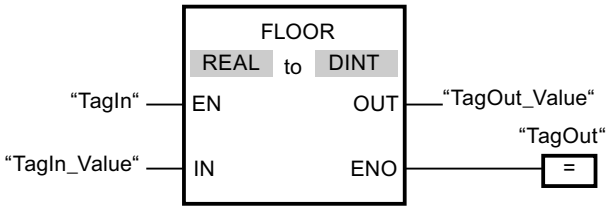
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value as floating-point number
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	I, Q, M, D, L, P	Result with the next lower integer

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value	
IN	TagIn_Value	0.50000000	-0.50000000
OUT	TagOut_Value	0	-1

If the operand "TagIn" has the signal state "1", the instruction "Generate next lower integer from floating-point number" is executed. The floating-point number at input "TagIn_Value" is rounded to the next lower integer and displayed at output "TagOut_Value". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Explicit conversion (Page 2391)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Selecting the data type of an FBD element (Page 2624)

TRUNC: Truncate numerical value

Description

You can use the "Truncate numerical value" instruction to form an integer from the value at input IN. The value at input IN is interpreted as a floating-point number. The instruction selects only the integer part of the floating-point number and sends this to output OUT without decimal places.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the instruction "Truncate numerical value":

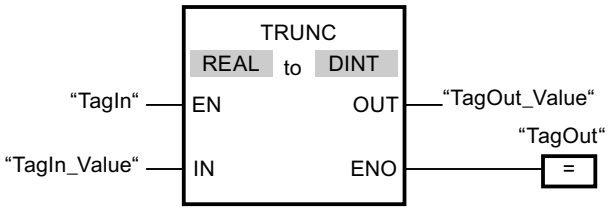
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L or constant	Input value as floating-point number
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L	Result with integer part of the floating-point number

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value	
IN	TagIn_Value	1.50000000	-1.50000000
OUT	TagOut_Value	1	-1

If the operand "TagIn" has the signal state "1", the "Truncate numerical value" instruction is executed. The integer part of the floating-point number at input "TagIn_Value" is converted to an integer and sent to output "TagOut_Value". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Explicit conversion (Page 2391)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

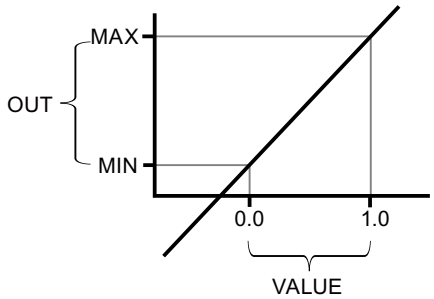
Selecting the data type of an FBD element (Page 2624)

SCALE_X: Scale

Description

You can use the "Scale" instruction to scale the value at the VALUE input by mapping it to a specified value range. When the instruction "Scale" is executed, the floating-point value at input VALUE is scaled to the value range, which is defined by the parameters MIN and MAX. The result of the scaling is an integer, which is stored at output OUT.

The following figure shows an example of how values can be scaled:



The "Scale" instruction works with the following equation:

OUT = [VALUE * (MAX – MIN)] + MIN

Enable output ENO has the signal state "0" if one of the following conditions is fulfilled:

- Enable input EN has the signal state "0".
- The value at input MIN is greater than or equal to the value at input MAX.
- The value of a specified floating-point number is outside the range of the normalized numbers according to IEEE-754.
- An overflow occurs.
- The value at input VALUE is NaN (Not a number = result of an invalid arithmetic operation).

Note

For more information on the conversion of analog values, refer to the respective manual.

Parameters

The following table shows the parameters of the "Scale" instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
MIN	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Low limit of the value range
VALUE	Input	Floating-point numbers	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Value to be scaled. If you enter a constant, you must declare it.
MAX	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	I, Q, M, D, L or constant	High limit of the value range
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L	I, Q, M, D, L	Result of scaling

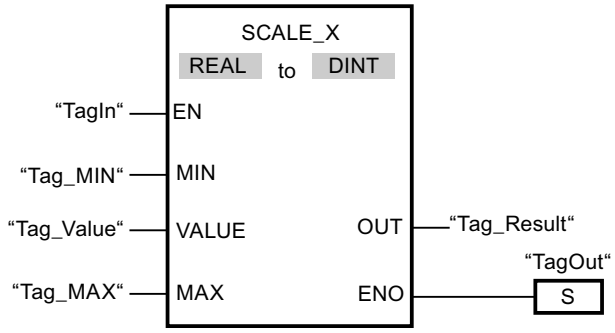
You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

For additional information on declaring constants, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
MIN	Tag_MIN	10
VALUE	Tag_Value	0.5
MAX	Tag_MAX	30
OUT	Tag_Result	20

If the operand "TagIn" has the signal state "1", the "Scale" instruction is executed. The value at input "Tag_Value" is scaled to the range of values defined by the values at inputs "Tag_MIN" and "Tag_MAX". The result is stored at output "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

NORM_X: Normalize (Page 3664)

Declaring global constants (Page 2489)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Inserting operands in FBD instructions (Page 2643)

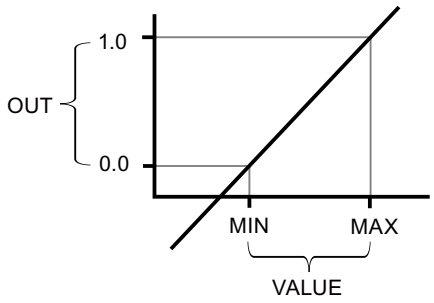
NORM_X: Normalize

Description

You can use the instruction "Normalize" to normalize the value of the tag at the VALUE input by mapping it to a linear scale. You can use the parameters MIN and MAX to define the limits of a value range that is applied to the scale. Depending on the location of the normalized value in this value range, the result at output OUT is calculated and stored as a floating-point number.

If the value to be normalized is equal to the value at input MIN, output OUT returns the value "0.0". If the value to be normalized is equal to the value at input MAX, output OUT returns the value "1.0".

The following figure shows an example of how values can be normalized:



The "Normalize" instruction works with the following equation:

OUT = (VALUE – MIN) / (MAX – MIN)

Enable output ENO has the signal state "0" if one of the following conditions is fulfilled:

- Enable input EN has the signal state "0".
- The value at input MIN is greater than or equal to the value at input MAX.
- The value of a specified floating-point number is outside the range of the normalized numbers according to IEEE-754.
- The value at input VALUE is NaN (result of an invalid arithmetic operation).

Note

For more information on the conversion of analog values, refer to the respective manual.

Parameters

The following table shows the parameters of the instruction "Normalize":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
MIN [1]	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Low limit of the value range
VALUE [1]	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Value to be normalized.
MAX [1]	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	I, Q, M, D, L or constant	High limit of the value range
OUT	Output	Floating-point numbers	I, Q, M, D, L	I, Q, M, D, L	Result of the normalization
[1] If you use constants in these three parameters, you only need to declare one of them.					

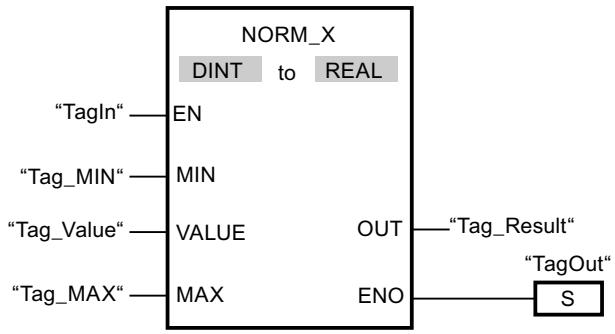
You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

For additional information on declaring constants, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
MIN	Tag_MIN	10
VALUE	Tag_Value	20

Parameter	Operand	Value
MAX	Tag_MAX	30
OUT	Tag_Result	0.5

If the operand "TagIn" has the signal state "1", the "Normalize" instruction is executed. The value at input "Tag_Value" is assigned to the range of values defined by the values at inputs "Tag_MIN" and "Tag_MAX". The tag value at input "Tag_Value" is normalized corresponding to the defined value range. The result is stored as a floating-point number at output "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

SCALE_X: Scale (Page 3662)

Declaring global constants (Page 2489)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Inserting operands in FBD instructions (Page 2643)

Other

SCALE: Scale

Description

Use the Scale instruction to convert the integer at the IN parameter into a floating-point number, which can be scaled in physical units between a low limit value and a high limit value. You can use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is scaled. The result of the instruction is output at the OUT parameter.

The "Scale" instruction works with the following equation:

OUT = [((FLOAT (IN) – K1)/(K2–K1)) * (HI_LIM–LO_LIM)] + LO_LIM

The values of the constants "K1" and "K2" are determined by the signal state on the BIPOLAR parameter. The following signal states are possible on the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case the "K1" constant has the value "-27648.0" and the "K2" constant the value "+27648.0".
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case the "K1" constant has the value "0.0" and the "K2" constant the value "+27648.0".

When the value at the IN parameter is greater than the value of the constant "K2", the result of the instruction is set to the value of the high limit (HI_LIM) and an error is output.

When the value at the IN parameter is less than the value of the constant "K1", the result of the instruction is set to the value of the low limit value (LO_LIM) and an error is output.

When the indicated low limit value is greater than the high limit value (LO_LIM > HI_LIM), the result is scaled in reverse proportion to the input value.

Parameters

The following table shows the parameters of the "Scale" instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	INT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value to be scaled.
HI_LIM	Input	REAL	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Low limit
BIPOLAR	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Indicates if the value at IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
OUT	Output	REAL	I, Q, M, D, L, P	I, Q, M, D, L, P	Result of the instruction
RET_VAL	Output	WORD	I, Q, M, D, L, P	I, Q, M, D, L, P	Error information

Parameters RET_VAL

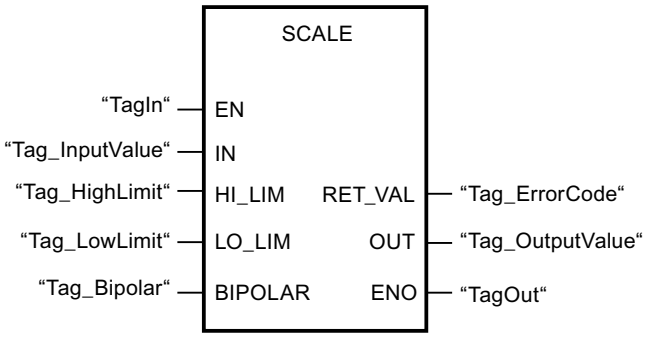
The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the "K2" constant or less than the value of the "K1" constant.

Error code* (W#16#...)	Explanation
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_InputValue	22
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
OUT	Tag_OutputValue	50.03978588
RET_VAL	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 2236)
Evaluating errors with output parameter RET_VAL (Page 5370)
Basics of the EN/ENO mechanism (Page 2447)
Inserting FBD elements (Page 2621)
Editing FBD elements (Page 2635)
Inserting operands in FBD instructions (Page 2643)
Switching display formats in the program status (Page 2980)
GET_ERR_ID: Get error ID locally (Page 3692)

UNSCALE: Unscale

Description

The "Unscale" instruction unscales the floating-point number at the IN parameter into physical units between a low limit value and a high limit and converts them into integers. You can use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is unscaled. The result of the instruction is output at the OUT parameter.

The "Unscale" instruction works with the following equation:

OUT = [((IN–LO_LIM)/(HI_LIM–LO_LIM)) * (K2–K1)] + K1

The values of the constants "K1" and "K2" are determined by the signal state on the BIPOLAR parameter. The following signal states are possible on the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case the "K1" constant has the value "-27648.0" and the "K2" constant the value "+27648.0".
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case the "K1" constant has the value "0.0" and the "K2" constant the value "+27648.0".

When the value at the IN parameter is greater than the value of the constant "HI_LIM", the result of the instruction is set to the value of the constant (K2) and an error is output.

When the value at the IN parameter is less than the value of the constant of the low limit (LO_LIM), the result of the instruction is set to the value of the constant (K1) and an error is output.

Parameters

The following table shows the parameters of the "Unscale" instruction:

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	REAL	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value to be unscaled to an integer value.
HI_LIM	Input	REAL	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Low limit

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
BIPOLAR	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Indicates if the value at the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
OUT	Output	INT	I, Q, M, D, L, P	I, Q, M, D, L, P	Result of the instruction
RET_VAL	Output	WORD	I, Q, M, D, L, P	I, Q, M, D, L, P	Error information

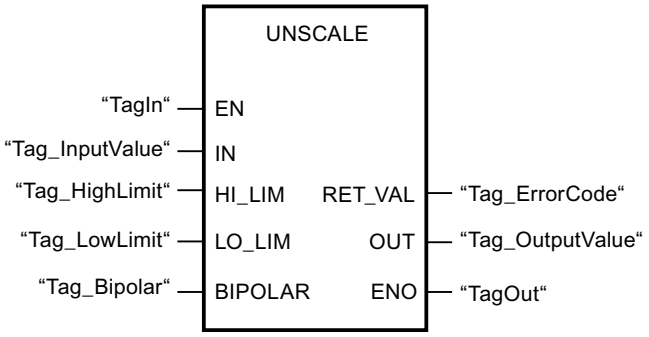
Parameters RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the high limit (HI_LIM) or less than the value of the low limit (LO_LIM).
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
OUT	Tag_OutputValue	22
RET_VAL	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 2236)

Evaluating errors with output parameter RET_VAL (Page 5370)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Switching display formats in the program status (Page 2980)

GET_ERR_ID: Get error ID locally (Page 3692)

Program control operations

JMP: Jump if RLO = 1

Description

You can use the "Jump if RLO = 1" instruction to interrupt the linear execution of the program and resume it in another network. The target network must be identified by a jump label (LABEL). The jump label description is entered in the placeholder above the instruction box.

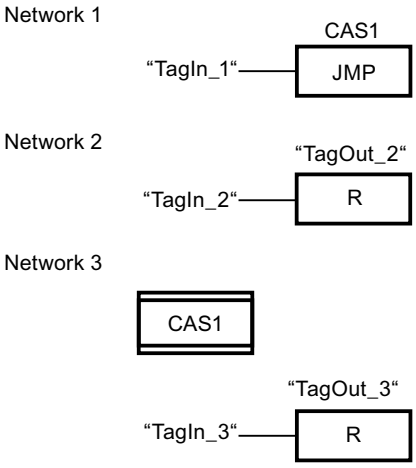
The specified jump label must be in the same block in which the instruction is executed. The name you specify can only occur once in the block. Only one jumping coil can occur in a network.

If the result of logic operation (RLO) at the input of the instruction is "1", the jump to the network identified by the jump label is executed. The jump direction can be towards higher or lower network numbers.

If the condition at the input of the instruction is not fulfilled (RLO = 0), execution of the program continues in the next network.

Example

The following example shows how the instruction works:



If the operand "TagIn_1" has the signal state "1", the instruction "Jump if RLO = 1" is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. If input "TagIn_3" has the signal state "1", output "TagOut_3" is reset.

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

JMPN: Jump if RLO = 0

Description

You can use the instruction "Jump if RLO = 0" to interrupt the linear execution of the program and resume it in another network, when the result of logic operation at the input of the instruction is "0". The target network must be identified by a jump label (LABEL). The jump label description is entered in the placeholder above the instruction box.

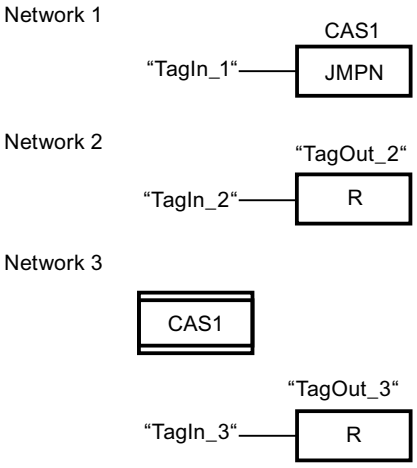
The specified jump label must be in the same block in which the instruction is executed. The name you specify can only occur once in the block. Only one jumping coil can occur in a network.

If the result of logic operation (RLO) at the input of the instruction is "0", the jump to the network identified by the jump label is executed. The jump direction can be towards higher or lower network numbers.

If the result of the logic operation RLO at the input of the instruction is "1", execution of the program continues in the next network.

Example

The following example shows how the instruction works:



If the operand "TagIn_1" has the signal state "0", the instruction "Jump if RLO = 0" is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. If input "TagIn_3" has the signal state "1", output "TagOut_3" is reset.

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

LABEL: Jump label

Description

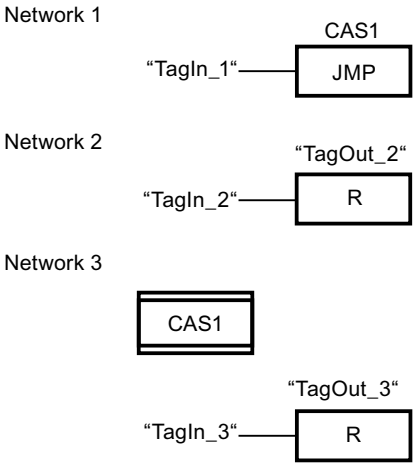
The jump label identifies a destination network in which the execution of the program can be resumed after the execution of a jump instruction.

The jump label and the instruction in which the jump label is specified must be located in the same block. The name of a jump label can only be assigned once in a block. You can declare up to 32 jump labels when you use a CPU S7-1200 and a maximum of 256 jump labels when you use a CPU S7-1500.

Only one jump label can be placed in a network. Each jump label can jump to several locations.

Example

The following example shows how the instruction works:



If the operand "TagIn_1" has the signal state "1", the instruction "Jump if RLO = 1" is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. If input "TagIn_3" has the signal state "1", output "TagOut_3" is reset.

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Adding additional inputs and outputs to FBD elements (Page 2639)

Removing instruction inputs and outputs (Page 2640)

Selecting the data type of an FBD element (Page 2624)

JMP_LIST: Define jump list

Description

You can use the "Define jump list" instruction to define several conditional jumps and continue the program execution in a specific network depending on the value of the K parameter.

You define the jumps with jump labels (LABEL), which you specify at the outputs of the instruction box. The number of outputs can be expanded in the instruction box. You can declare up to 32 outputs when you use a CPU S7-1200 and a maximum of 99 outputs when you use a CPU S7-1500.

The numbering of the outputs begins with the value "0" and is continued in ascending order with each new output. Only jump labels can be specified at the outputs of the instruction. It is not permitted to specify instructions or operands.

The value of the K parameter specifies the number of the output and thus the jump label where the program execution is to be resumed. If the value in the K parameter is greater than the number of available outputs, the program execution is resumed in the next network of the block.

The "Define jump list" instruction is only executed if the signal state is "1" at the EN enable input.

Parameters

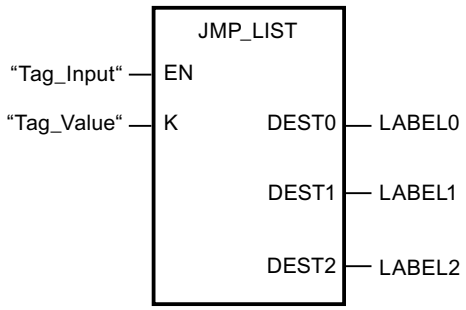
The following table shows the parameters of the "Define jump list" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, L, D	Enable input
K	Input	UINT	I, Q, M, L, D or constant	Specifies the number of the output and thus the jump that is executed.
DEST0	-	-	-	First jump label
DEST1	-	-	-	Second jump label
DESTn	-	-	-	Optional jump labels

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand/Jump label	Value
K	Tag_Value	1
DEST0	LABEL0	Jump to the network that is identified with the jump label "LABEL0".

Parameters	Operand/Jump label	Value
DEST1	LABEL1	Jump in the network that is identified with the jump label "LABEL1".
DEST2	LABEL2	Jump in the network that is identified with the jump label "LABEL2".

If the operand "Tag_Input" has the signal state "1", the instruction "Define jump list" is executed. The execution of the program is continued according to the value of the operand "Tag_Value" in the network that is identified with the jump label "LABEL1".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Adding additional inputs and outputs to FBD elements (Page 2639)

Removing instruction inputs and outputs (Page 2640)

Selecting the data type of an FBD element (Page 2624)

SWITCH: Jump distributor

Description

You can use the "Jump distributor" instruction to define multiple program jumps to be executed depending on the result of one or more comparison instructions.

At the parameter K, you specify the value to be compared. This value is compared with the values that the individual inputs return. You select the type of comparison for each input. The availability of various comparison instructions depends on the data type of the instruction.

The following table shows the comparison instructions that are available depending on the selected data type:

Data type		Instruction	Syntax
S7-1200	S7-1500		
Bit strings	Bit strings	Equal	==
		Not equal	<>
Integers, floating-point numbers, TIME, DATE, TOD	Integers, floating-point numbers, TIME, LTIME, DATE, TOD, LTOD, LDT	Equal	==
		Not equal	<>
		Greater or equal	>=
		Less or equal	<=
		Greater than	>
		Less than	<

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box. If you select a comparison instruction and the data type of the instruction is not yet defined, then the "<???>" drop-down list will only list those data types that are permitted for the selected comparison instruction.

The execution of the instruction begins with the first comparison and is executed until a comparison condition is fulfilled. When a comparison condition is fulfilled the subsequent comparison conditions are not considered. If none of the specified comparison conditions are fulfilled, the jump is executed at output ELSE. If not jump label is defined at output ELSE, the linear execution of the program is not interrupted but instead continued in the next network.

In its initial state the instruction box contains at least 2 outputs (DEST0 and DEST1). The number of outputs can be extended. The numbering of the outputs begins with the value "0" and is continued in ascending order with each new output. Specify jump labels (LABEL) at the outputs of the instruction. It is not permitted to specify instructions or operands at the outputs of the instruction.

An input is automatically inserted to each additional output. The jump programmed at an output is executed when the comparison condition of the corresponding is fulfilled.

Parameters

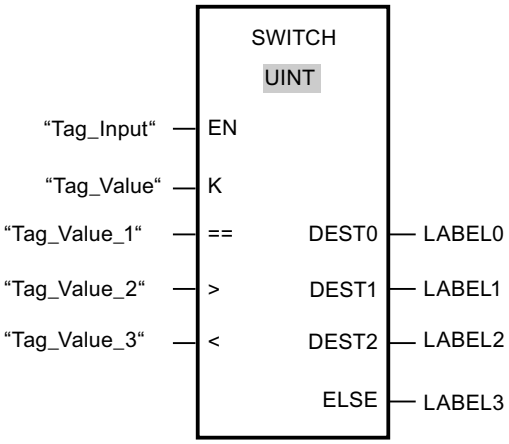
The following table shows the parameters of the instruction "Jump distributor":

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I, Q, M, D, L	Enable input
K	Input	UINT	UINT	I, Q, M, D, L or constant	Specifies the value to be compared.
<Comparison values>	Input	Bit strings, integers, floating-point numbers, TIME, DATE, TOD	Bit strings, integers, floating-point numbers, TIME, LTIME, DATE, TOD, LTOD, LDT	I, Q, M, D, L or constant	Input values with which the value of the parameter K is compared.
DEST0	-	-	-	-	First jump label
DEST1	-	-	-	-	Second jump label
DEST(n)	-	-	-	-	Optional jump labels (n = 2 to 99)
ELSE	-	-	-	-	Program jump which is executed if none of the comparison conditions are fulfilled.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand/Jump label	Value
K	Tag_Value	23
==	Tag_Value_1	20
>	Tag_Value_2	21
<	Tag_Value_3	19
DEST0	LABEL0	Jump to jump label "LABEL0", if the value of parameter K is equal to 20.
DEST1	LABEL1	Jump to jump label "LABEL1", if the value of parameter K is greater than 21.
DEST2	LABEL2	Jump to jump label "LABEL2", if the value of parameter K is less than 19.
ELSE	LABEL 3	Jump to jump label "LABEL3", if none of the comparison conditions are fulfilled.

If the operand "Tag_Input" changes to signal state "1", the instruction "Jump distributor" is executed. The execution of the program is continued in the network that is identified with the jump label "LABEL1".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Adding additional inputs and outputs to FBD elements (Page 2639)

Removing instruction inputs and outputs (Page 2640)

Selecting the data type of an FBD element (Page 2624)

RET: Return

Description

You can use the instruction "Return" to stop the execution of a block. This results in three types, in which the block processing can be completed:

- Without call of the "Return" instruction
  The block is exited after the execution of the last network. The ENO of the call function is set to the signal state "1".
- Call of the "Return" instruction with logic operation (see example)
  If the left connector has the signal state "1", then the block is exited. The ENO of the function call corresponds to the operand.
- Call of the "Return" instruction without logic operation
  The block is exited. The ENO of the function call corresponds to the operand.

Note

Only one jumping coil may be used in a network ("Return", "Jump if RLO = 1", "Jump if RLO = 0").

If the result of logic operation (RLO) at the input of the instruction "Return" is "1", the execution of the program is terminated in the currently called block and continued after the call function in the calling block (for example, in the calling OB). The status (ENO) of the call function is determined by the parameter of the instruction. This can assume the following values:

- RLO
- TRUE/FALSE
- <Operand>

To set the parameter values, double-click the instruction and select the corresponding value in the drop-down list.

The following table shows the status of the call function when the instruction "Return" is programmed in a network within the called block:

RLO	Parameter value	ENO of the call function
1	RLO	1
	TRUE	1
	FALSE	0
	<Operand>	<Operand>
0	RLO	In this case, the execution of the program continues in the next network of the called block.
	TRUE	
	FALSE	
	<Operand>	

If an OB is completed, another block is selected by the priority class system and started or re-executed.

- If the OB program cycle was completed, it is restarted.
- If an OB, which interrupted another block (e.g. an Alarm OB), is completed, then the interrupted block (e.g. OB program cycle) is executed.

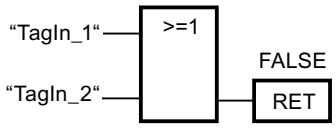
Parameters

The following table shows the parameters of the instruction "Return":

Parameters	Declaration	Data type	Memory area	Description
Status of the calling function when RLO = 1:				
RLO	-	-	-	Is set to the signal state of the RLO.
TRUE	-	-	-	1
FALSE	-	-	-	0
<Operand>	Input	BOOL	I, Q, M, D, L	Signal state of the specified operand

Example

The following example shows how the instruction works:



If the operands "TagIn_1" or "TagIn_2" have signal state "1", the instruction "Return" is executed. Program execution in the called block is terminated and continues in the calling block. The enable output ENO of the call function is reset to signal state "0".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Runtime control

ENDIS_PW: Limit and enable password legitimation

Description

You use the "Limit and enable password legitimation" instruction to specify whether passwords configured for the CPU are legitimized or not. In this way, you can prevent legitimized connections, even if the correct password is known.

When the instruction is called and the REQ parameter has the signal state "0", the currently set state is displayed at the output parameters. If changes have been made to the input parameters, these are not transferred to the output parameters.

When the instruction is called and the REQ parameter has the signal state "1", the signal state is taken from the input parameters (F_PWD, FULL_PWD, R_PWD, HMI_PWD). FALSE means that legitimation per password is not permitted; TRUE means the password can be used.

Disable or enabling of the passwords can be allowed or prohibited individually. For example, all passwords can be prohibited except the fail-safe password. You can thus limit access options to a small user group. The output parameters (F_PWD_ON, FULL_PWD_ON, R_PWD_ON, HMI_PWD_ON) always show the current status of the password use, regardless of the REQ parameter.

Passwords that are not configured must have the signal state TRUE at the input and return the signal state TRUE at the output. The fail-safe password can only be configured for an F-CPU and must therefore always be interconnected with the signal state TRUE in a standard CPU. If the instruction returns an error, the call has no effect and the previous lock is still in effect.

Disabled passwords can be enabled again under the following conditions:

- The CPU is reset to its factory settings.
- The front panel of the S7-1500 CPU supports a dialog that allows you to navigate to the appropriate menu in which the passwords can be enabled again.
- When you call the "Limit and enable password legitimation" instruction, the input parameter of the desired password has the signal state "1".
- Set the mode selector to STOP. The restriction to password legitimation is re-established as soon as the switch is set back to RUN.
- Plugging an empty memory card (transfer module or program card) into an S7-1200 CPU.
- The transition from POWER OFF to POWER ON disables the protection in the S7-1200 CPU. The "Limit and enable password legitimation" instruction must be called once again in the program (e.g. in the startup OB).

Note

The "Limit and enable password legitimation" instruction blocks access from HMI systems if the HMI password has not been enabled.

Note

Existing legitimized connections retain their access rights and you cannot use the "Limit and enable password legitimation" instruction to restrict them.

Preventing unintentional lockout on an S7-1500 CPU

The settings can be made on the front panel of the CPU and the CPU saves the most recent setting.

To prevent accidental lockout, the protection can be disabled by setting the mode selector to STOP with an S7-1500 CPU. The protection is automatically enabled again by setting mode selector to RUN without having to call the "Limit and enable password legitimation" instruction again or taking additional actions on the front panel.

Preventing unintentional lockout on an S7-1200 CPU

Because the S7-1200 CPU does not have a mode switch, protection is disabled with POWER OFF - POWER ON. This means that it is possible and advisable to prevent accidental lockout with the help of specific program sequences within your program.

To do so, program a time control either by means of a cyclic interrupt OB or a timer in the Main OB (OB 1). This gives you the option of calling the "Limit and enable password legitimation" instruction in the respective OB (e.g., OB 1 or OB 35) relatively quickly after a transition from POWER OFF to POWER ON and the associated disabling of the protection. Call the instruction in the startup OB (OB 100) to keep the time window in which the instruction is inactive and there are no limitations for password legitimation relatively small. This procedure will give you the best possible protection from unauthorized access.

If there has been an accidental lockout, you can skip the call in the startup OB (by querying an input parameter, for example) and you have the set time (for example, 10 seconds up to 1 minute) to establish a connection to the CPU before the lock becomes active once again.

If there is no timer in your program code and a lockout has occurred, insert an empty transfer card or an empty program card into the CPU. The empty transfer card or program card deletes the internal load memory of the CPU. You must then download the user program once again from STEP 7 to the CPU.

Procedure for lost passwords with an S7-1200 CPU

If you have lost the password for a password-protected S7-1200 CPU, delete the password-protected program with an empty transfer card or program card. The empty transfer card or

program card deletes the internal load memory of the CPU. You can then load a new user program from STEP 7 Basic in the CPU.

WARNING

Inserting an empty transfer card

When you insert a transfer card in a CPU during runtime, the CPU goes to STOP mode. If operating states are unstable, controllers may fail and thus cause the uncontrolled operation of the controlled devices. This leads to an unforeseen operation of the automation system which can cause deadly or serious injuries and/or damage to property.

Once you have pulled the transfer card, the content of the transfer card is available in the internal load memory. Make sure that the card does not include a program at this point.

WARNING

Inserting an empty program card

When you insert a program card in a CPU during runtime, the CPU goes to STOP mode. If operating states are unstable, controllers may fail and thus cause the uncontrolled operation of the controlled devices. This leads to an unforeseen operation of the automation system which can cause deadly or serious injuries and/or damage to property.

Make sure that the program card is empty. The internal load memory is copied to the empty program card. Once you have pulled the previously empty program card, the internal load memory is empty.

You must remove the transfer card or program card before you put the CPU into RUN mode.

Effects of password use on the operating modes

The following table shows what effects the password use by means of the "Limit and enable password legitimation" instruction has on the operating modes and the corresponding user actions.

Action	Password protection by means of the instruction
Basic state after • Operating mode switch to STOP • Reset memory manually (PG, switch, change of MC (Motion Control)) • Reset to factory setting	Not activated (no restrictions)
Basic state after POWER ON	• S7-1200 CPU: The lock is disabled and the instruction must be called once again in the program (e.g., in the startup OB). • S7-1500 CPU: Enabled (if a lock was activated before POWER OFF). The option of not allowing passwords is retentive.
Operating mode transition RUN/STARTUP/HOLD -> STOP (by terminating the instruction, an error or communication) or STOP -> STARTUP/RUN/HOLD	Activated Passwords still cannot be used.

Parameters

The following table shows the parameters of the "Limit and enable password legitimation" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
REQ	Input	BOOL	I, Q, M, D, L or constant	When the REQ parameter has the signal state "0", the currently set signal state of the passwords is queried.
F_PWD	Input	BOOL	I, Q, M, D, L or constant	Read/write access including fail-safe • F_PWD = "0": Do not allow password • F_PWD = "1": Allow password
FULL_PWD	Input	BOOL	I, Q, M, D, L or constant	Read/write access • FULL_PWD = "0": Do not allow password • FULL_PWD = "1": Allow password
R_PWD	Input	BOOL	I, Q, M, D, L or constant	Read access • R_PWD = "0": Do not allow password • R_PWD = "1": Allow password
HMI_PWD	Input	BOOL	I, Q, M, D, L or constant	HMI access • HMI_PWD = "0": Do not allow password • HMI_PWD = "1": Allow password

Parameter	Declaration	Data type	Memory area	Description
F_PWD_ON	Output	BOOL	I, Q, M, D, L	Read/write access including fail-safe • F_PWD_ON = "0": Password not allowed • F_PWD_ON = "1": Password allowed
FULL_PWD_ON	Output	BOOL	I, Q, M, D, L	Read/write access status • FULL_PWD_ON = "0": Password not allowed • FULL_PWD_ON = "1": Password allowed
R_PWD_ON	Output	BOOL	I, Q, M, D, L	Read-access status • R_PWD_ON = "0": Password not allowed • R_PWD_ON = "1": Password allowed
HMI_PWD_ON	Output	BOOL	I, Q, M, D, L	HMI-access status • HMI_PWD_ON = "0": Password not allowed • HMI_PWD_ON = "1": Password allowed
RET_VAL	Output	WORD	I, Q, M, D, L	Error information

For additional information on valid data types, refer to "See also".

Parameters RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8090	The "Limit and enable password legitimation" instruction is not supported
80D0	The password for fail-safe is not configured. The signal state must be TRUE for standard CPUs.
80D1	The read/write access is not configured
80D2	The read access is not configured
80D3	The HMI access is not configured
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Overview of the valid data types (Page 2236)

Inserting operands in FBD instructions (Page 2643)

Basics of error handling (Page 2821)

Principles of local error handling (Page 2823)

Error output priorities (Page 2824)

Enabling local error handling for a block (Page 2825)

RE_TRIGR: Restart cycle monitoring time

Description

You can use the "Restart cycle monitoring time" instruction to restart the cycle time monitoring of the CPU. The cycle monitoring time then starts over for the duration you have set in the CPU configuration.

The instruction "Restart cycle monitoring time" can be called regardless of the priority in all blocks.

If the instruction is called in a block with a higher priority, such as a hardware interrupt, a diagnostics interrupt or a cyclic interrupt, the instruction is not executed and the ENO enable output is set to signal state "0".

The instruction "Restart cycle monitoring time" can be called a maximum of 10 times in a program cycle.

Parameters

The instruction "Restart cycle monitoring time" has no parameters.

See also

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Overview of the valid data types (Page 2236)

Inserting operands in FBD instructions (Page 2643)

Basics of error handling (Page 2821)

Principles of local error handling (Page 2823)

Error output priorities (Page 2824)

Enabling local error handling for a block (Page 2825)

STP: Exit program

Description

The instruction "Exit program" instruction is used to set the CPU to STOP mode and therefore to terminate the execution of the program. The effects of changing from RUN to STOP depend on the CPU configuration.

When the (RLO) at the input of the instruction is "1", the CPU changes to STOP mode and the execution of the program is terminated. The signal state at the output of the instruction is not evaluated.

If the RLO at the input of the instruction is "0", then the instruction will not be executed.

Parameters

The instruction "Exit program" has no parameters.

See also

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Overview of the valid data types (Page 2236)

Inserting operands in FBD instructions (Page 2643)

GET_ERROR: Get error locally

Description

The "Get error locally" instruction is used to query the occurrence of errors within a block. This is usually to access error. If the system reports an error during the processing of the block, detailed information is stored in the operand at the ERROR output for the first error to occur during the execution of the block since the last execution of the instruction.

Only operands of the "ErrorStruct" system data type can be specified at the ERROR output. The "ErrorStruct" system data type specifies the exact structure in which the information about the error is stored. Using additional instructions, you can evaluate this structure and program an appropriate response. If several errors occur in the block, the error information about the next error to occur in the instruction is output only after the first error that occurred has been remedied.

Note

The ERROR output is only changed if error information is present. To set the output back to "0" after handling an error, you have the following options:

- Declare the tag in the "Temp" section of the block interface.
- Reset the tag to "0" before calling the instruction.
- Query the enable output ENO.

The enable output ENO of the instruction "Get error locally" instruction is set only if the enable input EN returns signal state "1" and error information is present. If one of these conditions is not fulfilled, the remaining program execution is not affected by the "Get error locally" instruction.

You can find an example of how you can implement the instruction in combination with other troubleshooting options under "See also".

Note

The "Get error locally" instruction enables local error handling within a block. If "Get error locally" is inserted in the program code of a block, any predefined system reactions are ignored if errors occur.

Parameters

The following table lists the parameters of the "Get error locally" instruction:

Parameter	Declaration	Data type	Memory area	Description
ERROR	Output	ErrorStruct	D, L	Error information

Data type "ErrorStruct"

The following table lists the structure of the "ErrorStruct" data type:

Structure component		Data type	Description
ERROR_ID		WORD	Error ID
FLAGS		BYTE	Shows if an error occurred during a block call. 16#01: Error during a block call 16#00: No error during a block call
REACTION		BYTE	Default reaction: 0: Ignore (write error), 1: Continue with substitute value "0" (read error), 2: Skip instruction (system error)
CODE_ADDRESS		CREF	Information about the address and type of block
	BLOCK_TYPE	BYTE	Type of block where the error occurred: 1: OB 2: FC 3: FB
	CB_NUMBER	UINT	Number of the code block
	OFFSET	UDINT	Reference to the internal memory
MODE		BYTE	Information about the address of an operand
OPERAND_NUMBER		UINT	Operand number of the machine command
POINTER_NUMBER_LOCATION		UINT	(A) Internal pointer
SLOT_NUMBER_SCOPE		UINT	(B) Storage area in internal memory
DATA_ADDRESS		NREF	Information about the address of an operand

Structure component		Data type	Description
	AREA	BYTE	(C) Memory area: L: 16#40 – 4E, 86, 87, 8E, 8F, C0 – CE I: 16#81 Q: 16#82 M: 16#83 DB: 16#84, 85, 8A, 8B Range violations for a directly editable tag of data type DINT: 16#04
	DB_NUMBER	UINT	(D) Number of the data block
	OFFSET	UDINT	(E) Relative address of the operand

Structure component "ERROR_ID"

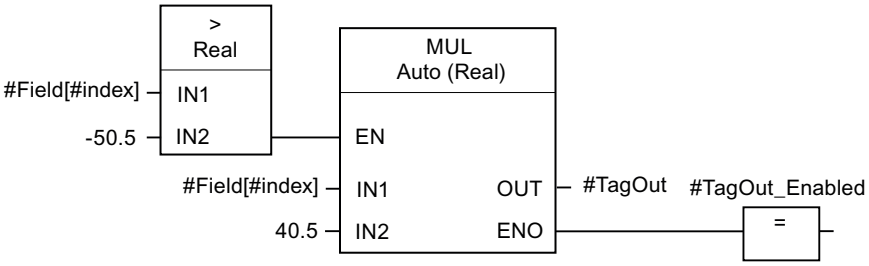
The following table lists the values that can be output at the structure component "ERROR_ID":

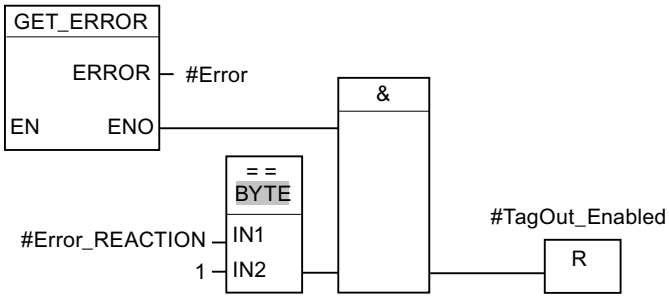
ID* (hexadecimal)	ID* (decimal)	Description
0	0	No error
2503	9475	Invalid pointer
2520	9504	Invalid STRING
2522	9506	Read error: Operand outside the valid range
2523	9507	Write error: Operand outside the valid range
2524	9508	Read error: Invalid operand
2525	9509	Write error: Invalid operand
2528	9512	Read error: Data alignment
2529	9513	Write error: Data alignment
252C	9516	Invalid pointer
2530	9520	Write error: Data block
2533	9523	Invalid pointer used
2538	9528	Access error: DB does not exist
2539	9529	Access error: Wrong DB used
253A	9530	Global data block does not exist
253C	9532	Faulty information or the function does not exist
253D	9533	System function does not exist
253E	9534	Faulty information or the function block does not exist
253F	9535	System block does not exist
2550	9552	Access error: DB does not exist
2551	9553	Access error: Wrong DB used
2575	9589	Error in the program nesting depth
2576	9590	Error in the local data distribution
2577	9591	The block property "Parameter passing via registers" is not selected.
25A0	9632	Internal error in TP

ID* (hexadecimal)	ID* (decimal)	Description
25A1	9633	Tag is write-protected
25A2	9634	Invalid numerical value of tag
2942	10562	Read error: Input
2943	10563	Write error: Output
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".		

Example

The following example shows how the instruction works:





An error occurred accessing the "#Field[#index]" tag. The enable output ENO of the "Multiply" instruction and the "#TagOut_Enabled" operand have signal state "1" despite the read/access error and the multiplication is performed with the value "0.0". If this error scenario occurs, it is recommended to program the "Get error locally" instruction after the "Multiply" instruction to get the error. The error information supplied by the "Get error locally" instruction is evaluated with the comparison instruction "Equal". If the "#Error.REACTION" structure component has the value "1", this involves a read/access error and the "#TagOut_Enabled" output is reset.

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Basics of error handling (Page 2821)

Principles of local error handling (Page 2823)

Error output priorities (Page 2824)

Enabling local error handling for a block (Page 2825)

Switching display formats in the program status (Page 2980)

Querying and fixing errors in the program code (Page 258)

GET_ERR_ID: Get error ID locally

Description

The "Get error ID locally" instruction is used to query the occurrence of errors within a block. This is usually to access error. If the system reports during the block processing errors within the block execution since the last execution of the instruction, the error ID of the first error that occurred in the tag is stored at the ID output.

Only operands of the WORD data type can be specified at the ID output. If several errors occur in the block, the error ID of the next error to occur in the instruction is output only after the first error that occurred has been remedied.

Note

The ID output is only changed if error information is present. To set the output back to "0" after handling an error, you have the following options:

- Declare the tag in the "Temp" section of the block interface.
- Reset the tag to "0" before calling the instruction.
- Query the enable output ENO.

The enable output ENO of the instruction "Get error ID locally" instruction is set only if the enable input EN returns signal state "1" and error information is present. If one of these conditions is not fulfilled, the remaining program execution is not affected by the "Get error ID locally" instruction.

You can find an example of how you can implement the instruction in combination with other troubleshooting options under "See also".

Note

The "Get error ID locally" instruction enables local error handling within a block. If the "Get error ID locally" instruction is inserted in the program code of a block, any predefined system reactions are ignored if errors occur.

Parameters

The following table lists the parameters of the "Get error ID locally" instruction:

Parameter	Declaration	Data type	Memory area	Description
ID	Output	WORD	I, Q, M, D, L	Error ID

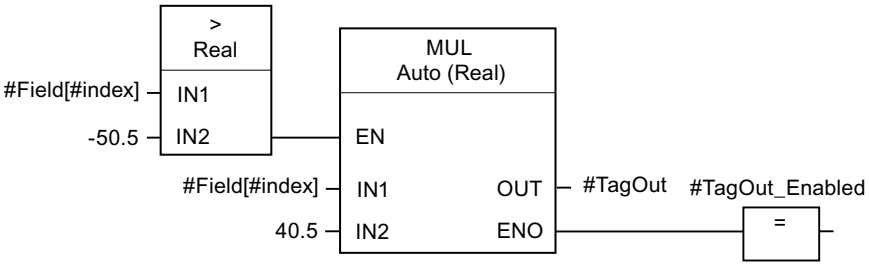
ID parameter

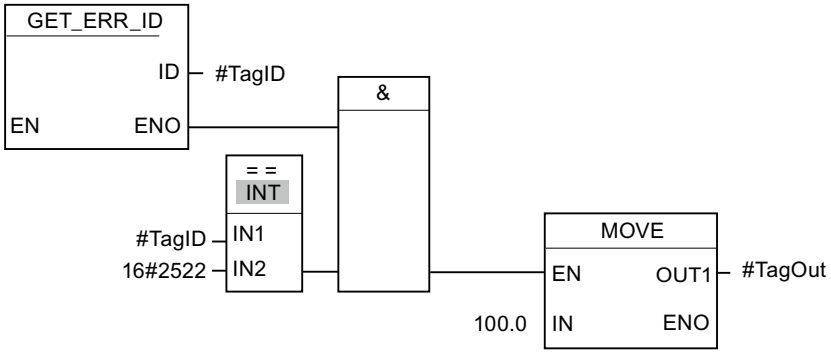
The following table lists the values that can be output at the ID parameter:

ID* (hexadecimal)	ID* (decimal)	Description
0	0	No error
2503	9475	Invalid pointer
2520	9504	Invalid STRING
2522	9506	Read error: Operand outside the valid range
2523	9507	Write error: Operand outside the valid range
2524	9508	Read error: Invalid operand
2525	9509	Write error: Invalid operand
2528	9512	Read error: Data alignment
2529	9513	Write error: Data alignment
252C	9516	Invalid pointer
2530	9520	Write error: Data block
2533	9523	Invalid pointer used
2538	9528	Access error: DB does not exist
2539	9529	Access error: Wrong DB used
253A	9530	Global data block does not exist
253C	9532	Faulty information or the function does not exist
253D	9533	System function does not exist
253E	9534	Faulty information or the function block does not exist
253F	9535	System block does not exist
2550	9552	Access error: DB does not exist
2551	9553	Access error: Wrong DB used
2575	9589	Error in the program nesting depth
2576	9590	Error in the local data distribution
2577	9591	The block property "Parameter passing via registers" is not selected.
25A0	9632	Internal error in TP
25A1	9633	Tag is write-protected
25A2	9634	Invalid numerical value of tag
2942	10562	Read error: Input
2943	10563	Write error: Output
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".		

Example

The following example shows how the instruction works:





An error occurred accessing the "#Field[#index]" tag. The enable output ENO of the "Multiply" instruction and the "#TagOut_Enabled" operand have signal state "1" despite the read/access error and the multiplication is performed with the value "0.0". If this error scenario occurs, it is recommended to program the "Get error ID locally" instruction after the "Multiply" instruction to get the error. The error information supplied by the "Get error ID locally" instruction is evaluated with the comparison instruction "Equal". If the operand "#TagID" returns the ID 2522, this involves a read/access error and the value "100.0" is written to the "#TagOut" output.

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Basics of error handling (Page 2821)

Principles of local error handling (Page 2823)

Error output priorities (Page 2824)

Enabling local error handling for a block (Page 2825)

Switching display formats in the program status (Page 2980)

Querying and fixing errors in the program code (Page 258)

INIT_RD: Initialize all retain data

Description

The "Initialize all retain data" instruction is used to reset the retentive data of all data blocks, bit memories and SIMATIC timers and counters at the same time. The instruction can only be executed within a startup OB because the execution exceeds the program cycle duration.

Parameters

The following table shows the parameters of the "Initialize all retain data" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
REQ	Input	BOOL	I, Q, M, D, L, T, C or constant	If the input "REQ" has the signal state "1", all retentive data are reset.
RET_VAL	Output	INT	I, Q, M, D, L	Error information: If an error occurs during the execution of the instruction, an error code is output at the RET_VAL parameter.

For additional information on valid data types, refer to "See also".

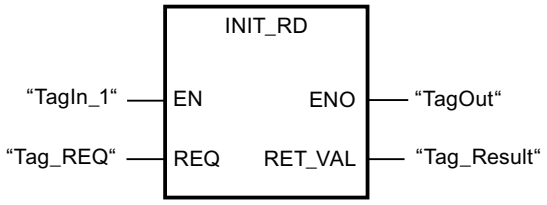
Parameters RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B5	The instruction cannot be executed because it was not programmed within a startup OB.
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal values in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:



If the operands "TagIn_1" and "Tag_REQ" have signal state "1", the instruction is executed. All retentive data of all data blocks, bit memories and SIMATIC timers and counters are reset. If the instruction is executed without errors, the ENO enable output has the signal state "1".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Basics of error handling (Page 2821)

Principles of local error handling (Page 2823)

Error output priorities (Page 2824)

Enabling local error handling for a block (Page 2825)

Evaluating errors with output parameter RET_VAL (Page 5370)

Basics of the EN/ENO mechanism (Page 2447)

Selecting the data type of an FBD element (Page 2624)

Switching display formats in the program status (Page 2980)

GET_ERR_ID: Get error ID locally (Page 3692)

WAIT: Configure time delay

Description

The "Configure time delay" instruction is used to halt the execution of the program for a set period of time. You indicate the period of time in microseconds on the WT parameter of the instruction.

You can configure time delays of up to 32767 microseconds (µs). The smallest possible delay time depends on the respective CPU and corresponds to the execution time of the "Configure time delay" instruction.

The execution of the instruction can be interrupted by higher priority events.

The "Configure time delay" instruction supplies no error information.

Parameters

The following table shows the parameters of the "Configure time delay" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
WT	Input	INT	I, Q, M, D, L, P or constant	Time delay in microseconds (µs)

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

RUNTIME: Measure program runtime

Description

The "Measure program runtime" instruction is used to measure the runtime of the entire program, individual blocks or command sequences.

If you want to measure the runtime of your entire program, call the instruction "Measure program runtime" in OB1. Measurement of the runtime is started with the first call and the output RET_VAL returns the runtime of the program after the second call. The measured runtime includes all CPU processes that can occur during the program execution, for example, interruptions caused by higher-level events or communication. The instruction "Measure program runtime" reads an internal counter of the CPU and write the value to the IN-OUT parameter MEM. The instruction calculates the current program runtime according to the internal counter frequency and writes it to output RET_VAL.

If you want to measure the runtime of individual blocks or individual command sequences, you need three separate networks. Call the instruction "Measure program runtime" in an individual network within your program. You set the starting point of the runtime measurement with this first call of the instruction. Then you call the required program block or the command sequence in the next network. In another network, call the "Measure program runtime" instruction a second time and assign the same memory to the IN-OUT parameter MEM as you did during the first call of the instruction. The "Measure program runtime" instruction in the third network reads an internal CPU counter and calculates the current runtime of the program block or the command sequence according to the internal counter frequency and writes it to the output RET_VAL.

The "Measure program runtime" instruction uses an internal high-frequency counter to calculate the time. If the counter overruns, the instruction returns values <= 0.0. Ignore these runtime values.

Note

The runtime of a command sequence cannot be determined exactly, because the sequence of instructions within a command sequence is changed during optimized compilation of the program.

Parameters

The following table lists the parameters of the "Measure program runtime" instruction:

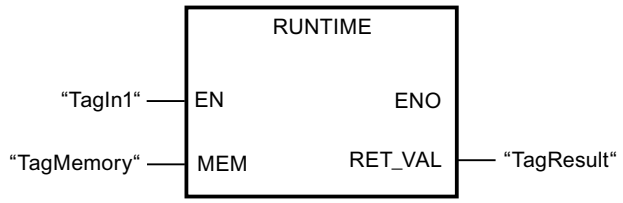
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
MEM	InOut	LREAL	M, D, L	Saves the starting point of the runtime measurement.
RET_VAL	Output	LREAL	I, Q, M, D, L	Returns the measured runtime in seconds

For additional information on valid data types, refer to "See also".

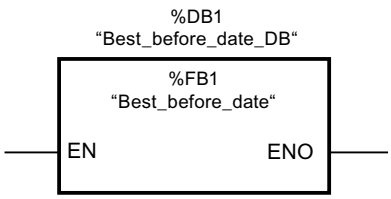
Example

The following example shows how the instruction works based on the runtime calculation of a program block:

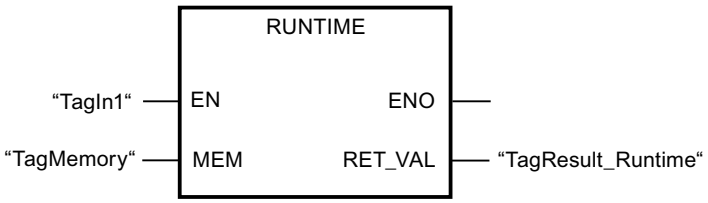
Network 1:



Network 2:



Network 3:



When the "TagIn1" operand in network 1 has the signal state "1", the instruction is executed. The starting point for the runtime measurement is set with the first call of the instruction and buffered as reference for the second call of the instruction in the "TagMemory" operand.

The "Best_before_date" program block FB1 is called in network 2.

When the FB1 program block has been executed and the "TagIn1" operand has the signal state "1", the instruction in network 3 is executed. The second call of the instruction calculates the runtime of the program block and writes the result to the output RET_VAL.

See also

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Overview of the valid data types (Page 2236)

Inserting operands in FBD instructions (Page 2643)

Basics of error handling (Page 2821)

Principles of local error handling (Page 2823)

Error output priorities (Page 2824)

Enabling local error handling for a block (Page 2825)

Word logic operations

AND: AND logic operation

Description

You can use the instruction "AND logic operation" to link the value at input IN1 to the value at input IN2 bit-by-bit by AND logic and query the result at the output OUT.

When the instruction is executed, bit 0 of the value at input IN1 is linked by AND logic to bit 0 of the value at input IN2. The result is stored in bit 0 of output OUT. The same logic operation is executed for all other bits of the specified values.

In its initial state the instruction box contains at least 2 inputs (IN1 and IN2). The number of inputs can be extended. The inserted inputs are numbered in ascending order in the box. During the execution of the instruction, the values of all available input parameters are linked by AND logic. The result is stored at output "OUT".

The result bit has the signal state "1" only when both of the bits in the logic operation also have signal state "1". If one of the two bits of the logic operation has signal state "0", the corresponding result bit is reset.

Parameters

The following table shows the parameters of the instruction "AND logic operation":

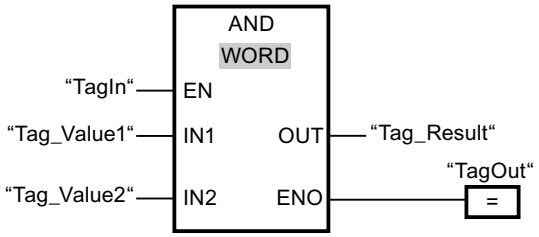
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN1	Input	Bit strings	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	First value for logic operation
IN2	Input	Bit strings	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Second value for logic operation
INn	Input	Bit strings	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Optional input values
OUT	Output	Bit strings	I, Q, M, D, L, P	I, Q, M, D, L, P	Result of the instruction

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN1	Tag_Value1	0101 0101 0101 0101
IN2	Tag_Value2	0000 0000 0000 1111
OUT	Tag_Result	0000 0000 0000 0101

If the operand "TagIn" has the signal state "1", the instruction "AND logic operation" is executed. The value of operand "Tag_Value1" is linked by AND to the value of the operand

"Tag_Value2". The result is mapped bit-for-bit and sent to the operand "Tag_Result". The enable output ENO and the output "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 2236)

Adding additional inputs and outputs to FBD elements (Page 2639)

Removing instruction inputs and outputs (Page 2640)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Selecting the data type of an FBD element (Page 2624)

Inserting operands in FBD instructions (Page 2643)

OR: OR logic operation

Description

You can use the instruction "OR logic operation" to link the value at input IN1 to the value at input IN2 bit-by-bit by OR logic and query the result at the output OUT.

When the instruction is executed, bit 0 of the value at input IN1 is linked by OR logic to bit 0 of the value at input IN2. The result is stored in bit 0 of output OUT. The same logic operation is executed for all bits of the specified tags.

In its initial state the instruction box contains at least 2 inputs (IN1 and IN2). The number of inputs can be extended in the instruction box. The inserted inputs are numbered in ascending order in the box. During the execution of the instruction, the values of all available input parameters are linked by OR logic. The result is stored at output "OUT".

The result bit has the signal state "1" when at least one of the two bits in the logic operation has the signal state "1". If both of the bits of the logic operation have signal state "0", the corresponding result bit is reset.

Parameters

The following table shows the parameters of the instruction "OR logic operation":

Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN1	Input	Bit strings	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	First value for logic operation

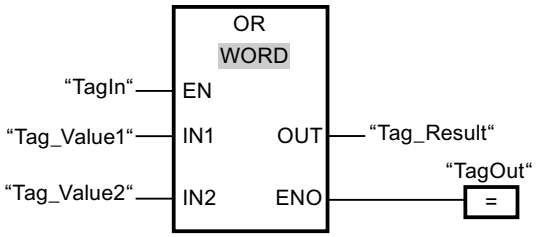
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
IN2	Input	Bit strings	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Second value for logic operation
INn	Input	Bit strings	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Optional input values
OUT	Output	Bit strings	I, Q, M, D, L, P	I, Q, M, D, L, P	Result of the instruction

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN1	Tag_Value1	0101 0101 0101 0101
IN2	Tag_Value2	0000 0000 0000 1111
OUT	Tag_Result	0101 0101 0101 1111

If the operand "TagIn" has the signal state "1", the instruction "OR logic operation" is executed. The value of operand "Tag_Value1" is linked by OR to the value of the operand "Tag_Value2". The result is mapped bit-for-bit and sent to the operand "Tag_Result". The enable output ENO and the output "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 2236)

Adding additional inputs and outputs to FBD elements (Page 2639)

Removing instruction inputs and outputs (Page 2640)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Selecting the data type of an FBD element (Page 2624)

XOR: EXCLUSIVE OR logic operation

Description

You can use the instruction "EXCLUSIVE OR logic operation" to link the value at input IN1 to the value at input IN2 bit-by-bit by EXCLUSIVE OR logic and query the result at the output OUT.

When the instruction is executed, bit 0 of the value at input IN1 is linked by EXCLUSIVE OR logic to bit 0 of the value at input IN2. The result is stored in bit 0 of output OUT. The same logic operation is executed for all other bits of the specified value.

In its initial state the instruction box contains at least 2 inputs (IN1 and IN2). The number of inputs can be extended in the instruction box. The inserted inputs are numbered in ascending order in the box. During the execution of the instruction, the values of all available input parameters are linked by EXCLUSIVE OR logic. The result is stored at output "OUT".

The result bit has the signal state "1" when one of the two bits in the logic operation has the signal state "1". If both of the bits of the logic operation have signal state "1" or "0", the corresponding result bit is reset.

Parameters

The following table shows the parameters of the instruction "EXCLUSIVE OR logic operation":

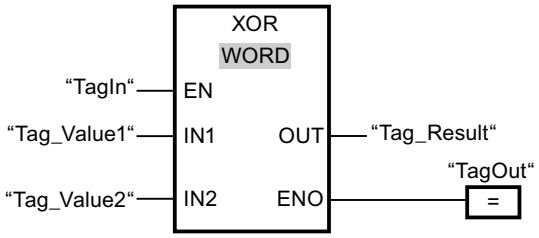
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN1	Input	Bit strings	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	First value for logic operation
IN2	Input	Bit strings	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Second value for logic operation
INn	Input	Bit strings	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Optional input values
OUT	Output	Bit strings	I, Q, M, D, L, P	I, Q, M, D, L, P	Result of the instruction

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN1	Tag_Value1	0101 0101 0101 0101
IN2	Tag_Value2	0000 0000 0000 1111
OUT	Tag_Result	0101 0101 0101 1010

If the operand "TagIn" has the signal state "1", the instruction "EXCLUSIVE OR logic operation" is executed. The value of operand "Tag_Value1" is linked by EXCLUSIVE OR to the value of the operand "Tag_Value2". The result is mapped bit-for-bit and sent to the operand "Tag_Result". The enable output ENO and the output "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 2236)

Adding additional inputs and outputs to FBD elements (Page 2639)

Removing instruction inputs and outputs (Page 2640)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Selecting the data type of an FBD element (Page 2624)

Inserting operands in FBD instructions (Page 2643)

INV: Create ones complement

Description

You can use the instruction "Create ones complement" to invert the signal status of the bits at input IN. When the instruction is processed, the value at input IN is linked to EXCLUSIVE OR by a hexadecimal mask (W#16#FFFF for 16-bit numbers or DW#16#FFFF FFFF for 32-bit number). This inverts the signal state of the individual bits that are then stored at output OUT.

Parameters

The following table shows the parameters of the instruction "Create ones complement":

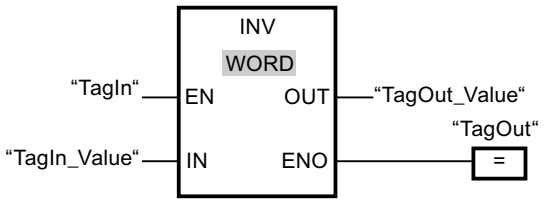
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value
OUT	Output	Bit strings, integers	I, Q, M, D, L, P	I, Q, M, D, L, P	Ones complement of the value at input IN

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value	
IN	TagIn_Value	W#16#000F	W#16#7E
OUT	TagOut_Value	W#16#FFF0	W#16#81

If the operand "TagIn" has the signal state "1", the "Create ones complement" instruction is executed. The instruction inverts the signal state of the individual bits at input "TagIn_Value" and writes the result to output "TagOut_Value". The enable output ENO and the output "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Selecting the data type of an FBD element (Page 2624)

DECO: Decode

Description

You can use the "Decode" instruction to set a bit in the output value specified by the input value.

The "Decode" instruction reads the value at the IN input and sets the bit in the output value whose bit position corresponds to the read value. The other bits in the output value are filled with zeroes. If the value at input IN is greater than 31, a modulo 32 instruction is executed.

Parameters

The following table shows the parameters of the "Decode" instruction:

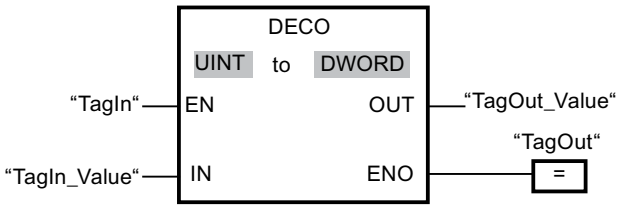
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	UINT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Position of the bit in the output value which is set.
OUT	Output	Bit strings	I, Q, M, D, L, P	I, Q, M, D, L, P	Output value

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

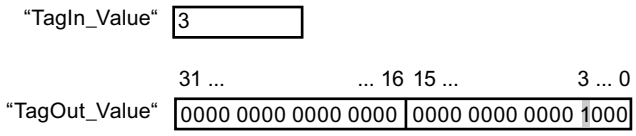
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following figure shows how the instruction works using specific operand values:



If the operand "TagIn" has the signal state "1", the "Decode" instruction is executed. The instruction reads bit number "3" from the value of the operand "TagIn_Value" and sets the third bit to the value of the operand "TagOut_Value".

If no errors occur during the execution of the instruction, the output ENO has the signal state "1" and the output "TagOut" is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Selecting the data type of an FBD element (Page 2624)

Inserting operands in FBD instructions (Page 2643)

ENCO: Encode

Description

The instruction "Encode" is used to read the bit number of the lowest bit in the input value and output it to the output OUT.

The instruction "Encode" selects the least significant bit of the value at the IN input and writes its bit number to the tag in the output OUT.

Parameters

The following table shows the parameters of the instruction "Encode":

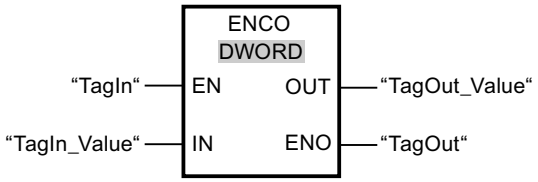
Parameters	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
EN	Input	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Bit strings	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value
OUT	Output	INT	I, Q, M, D, L, P	I, Q, M, D, L, P	Output value

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

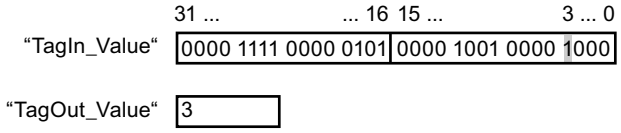
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following figure shows how the instruction works using specific operand values:



If the operand "TagIn" has the signal state "1", the "Encode" instruction is executed. The instruction selects bit position "3" as the least significant bit at input "TagIn_Value" and writes the value "3" to the tag at the output "TagOut_Value.

If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Editing FBD elements (Page 2635)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Inserting operands in FBD instructions (Page 2643)

SEL: Select

Description

Depending on the signal state at switch (input G), the "Select" instruction selects one of the inputs IN0 or IN1 and moves its content to the output OUT. When the input G has the signal state "0", the value at the input IN0 is moved. When the input G has the signal state "1", the value at the input IN1 is moved to the output OUT.

The instruction can only be executed if the enable input EN has the signal state "1" and the tags at all parameters have the same data type.

Parameters

The following table shows the parameters of the "Select" instruction:

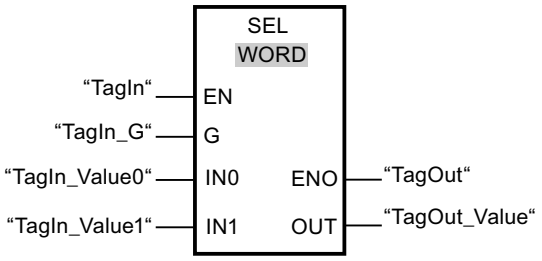
Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
G	Input	BOOL	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L, T, C or constant	Switch
IN0	Input	Bit strings, integers, floating-point numbers, TIME, TOD, DATE, CHAR	Bit strings, integers, floating-point numbers, timers, TOD, LTOD, DATE, CHAR	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	First input value
IN1	Input	Bit strings, integers, floating-point numbers, TIME, TOD, DATE, CHAR	Bit strings, integers, floating-point numbers, timers, TOD, LTOD, DATE, CHAR	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Second input value
OUT	Output	Bit strings, integers, floating-point numbers, TIME, TOD, DATE, CHAR	Bit strings, integers, floating-point numbers, timers, TOD, LTOD, DATE, CHAR	I, Q, M, D, L, P	I, Q, M, D, L, P	Result

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
G	TagIn_G	0	1
IN0	TagIn_Value0	W#16#0000	W#16#4C
IN1	TagIn_Value1	W#16#FFFF	W#16#5E
OUT	TagOut_Value	W#16#0000	W#16#5E

If the operand "TagIn" has the signal state "1", the instruction "Select" is executed. Based on the signal state at the "TagIn_G" input, the value of the "TagIn_Value0" or "TagIn_Value1" input is selected and moved to the "TagOut_Value" output. If the instruction is executed without errors, the enable output ENO has the signal state "1" and the output "TagOut" is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Selecting the data type of an FBD element (Page 2624)

MUX: Multiplex

Description

You can use the instruction "Multiplex" to copy the content of a selected input to output OUT. In its initial state the instruction box contains at least 2 inputs (IN0 and IN1). The number of selectable inputs of the instruction box can be expanded. You can declare a maximum of 32 inputs.

The inputs are numbered automatically in the box. Numbering starts at IN0 and is incremented continuously with each new input. You can use the parameter K to determine the input whose content should be copied to output OUT. If the value of the parameter K is greater than the number of available inputs, the content of the parameter ELSE is copied to output OUT and enable output ENO is assigned the signal state "0".

The "Multiplex" instruction can only be executed if the tags have the same data type in all inputs and in the OUT output. The exception here is the parameter K, which can only be specified as an integer.

The enable output ENO is reset if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value of the parameter K is greater than the number of available inputs.
- Errors occurred during the execution of the instruction.

Parameters

The following table shows the parameters of the "Multiplex" instruction:

Parameter	Declaring	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
K	Input	Integers	Integers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Specifies the input whose content is to be copied. • If K = 0 => Parameter IN0 • If K = 1 => Parameter IN1, etc.
IN0	Input	Binary numbers, integers, floating-point numbers, strings, TIME, TOD, DATE	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	First input value
IN1	Input	Binary numbers, integers, floating-point numbers, strings, TIME, TOD, DATE	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Second input value

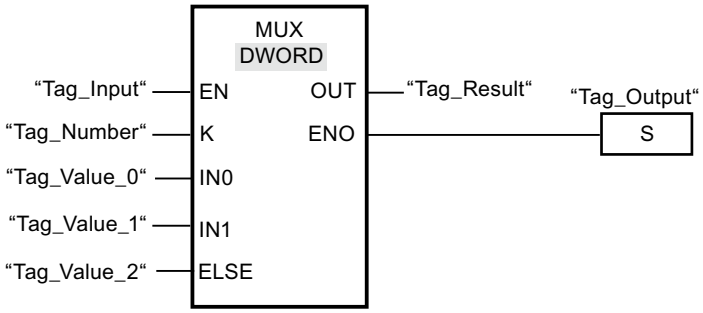
Parameter	Declaring	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
INn	Input	Binary numbers, integers, floating-point numbers, strings, TIME, TOD, DATE	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Optional input values
ELSE	Input	Binary numbers, integers, floating-point numbers, strings, TIME, TOD, DATE	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Specifies the value to be copied when K > n.
OUT	Output	Binary numbers, integers, floating-point numbers, strings, TIME, TOD, DATE	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	I, Q, M, D, L, P	Output to which the value is to be copied.

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
K	Tag_Number	1
IN0	Tag_ValueI_0	DW#16#00000000
IN1	Tag_Value_1	DW#16#003E4A7D
ELSE	Tag_Value_2	DW#16#FFFF0000
OUT	Tag_Result	DW#16#003E4A7D

If the operand "Tag_Input" has the signal state "1", the "Multiplex" instruction is executed. Depending on the value of the operand "Tag_Number", the value at input "Tag_Value_1" is copied and assigned to the operand at output "Tag_Result". If no errors occur during the execution of the instruction, the outputs ENO and "Tag_Output" are set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Selecting the data type of an FBD element (Page 2624)

DEMUX: Demultiplex

Description

You can use the instruction "Demultiplex" to copy the content of the input IN to a selected output. In its initial state the instruction box contains at least 2 outputs (OUT0 and OUT1). The number of selectable outputs can be extended in the instruction box. The outputs are numbered automatically in the box. Numbering starts at OUT0 and continues consecutively with each new output. You can use the parameter K to define the output to which the content of input IN will be copied. The other outputs will not be changed. If the value of the parameter K is greater

than the number of available outputs, then the content of input IN in the parameter ELSE and the enable output ENO will be assigned to the signal state "0".

The instruction "Demultiplex" can only be executed if the tags at the input IN and at all outputs are of the same data type. The exception here is the parameter K, which can only be specified as an integer.

The enable output ENO is reset if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value of the parameter K is greater than the number of available outputs.
- Errors occurred during the execution of the instruction.

Parameters

The following table shows the parameters of the instruction "Demultiplex":

Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
K	Input	Integers	Integers	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Specifies the output to which the input value (IN) will be copied. • If K = 0 => Parameter OUT0 • If K = 1 => Parameter OUT1, etc.
IN	Input	Binary numbers, integers, floating-point numbers, strings, TIME, TOD, DATE	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	I, Q, M, D, L, P or constant	Input value

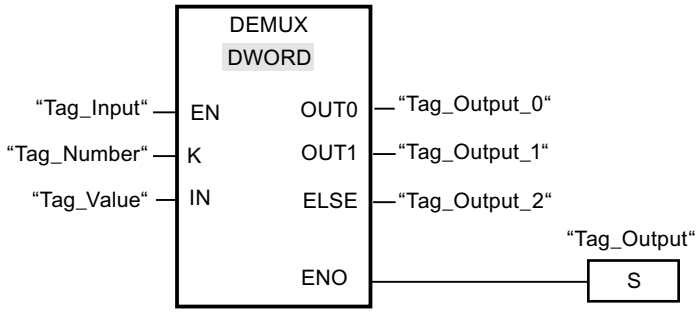
Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
OUT0	Output	Binary numbers, integers, floating-point numbers, strings, TIME, TOD, DATE	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	I, Q, M, D, L, P	First output
OUT1	Output	Binary numbers, integers, floating-point numbers, strings, TIME, TOD, DATE	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	I, Q, M, D, L, P	Second output
OUTn	Output	Binary numbers, integers, floating-point numbers, strings, TIME, TOD, DATE	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	I, Q, M, D, L, P	Optional outputs
ELSE	Output	Binary numbers, integers, floating-point numbers, strings, TIME, TOD, DATE	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	I, Q, M, D, L, P	Output to which the input value (IN) at K > n is copied.

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on available data types, refer to "See also".

Example

The following example shows how the instruction works:



The following tables show how the instruction works using specific operand values:

Table 9-26 Input values of the "Demultiplex" instruction before network execution

Parameter	Operand	Values	
K	Tag_Number	1	4
IN	Tag_Value	DW#16#FFFFFFFF	DW#16#003E4A7D

Table 9-27 Output values of the "Demultiplex" instruction after network execution

Parameter	Operand	Values	
OUT0	Tag_Output_0	Unchanged	Unchanged
OUT1	Tag_Output_1	DW#16#FFFFFFFF	Unchanged
ELSE	Tag_Output_2	Unchanged	DW#16#003E4A7D

The "Demultiplex" instruction is executed when input "Tag_Input" has the signal state "1". Depending on the value of the operand "Tag_Number", the value at input "IN" is copied to the corresponding output.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Selecting the data type of an FBD element (Page 2624)

Shift and rotate

SHR: Shift right

Description

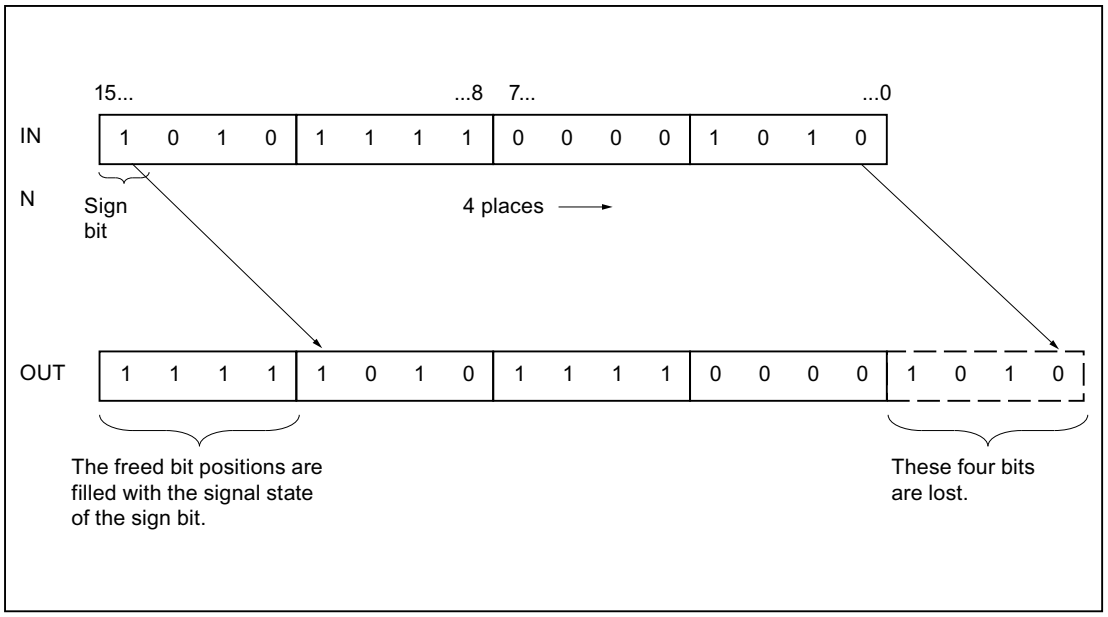
You can use the "Shift right" instruction to shift the content of the operand at the input IN bit-by-bit to the right and query the result at the OUT output. The input N is used to specify the number of bit positions by which the specified value should be moved.

If the value at the input N is "0", the value at input IN is copied unchanged to the operand at output OUT.

If the value at the input N is greater than the number of available bit positions, the operand value at input IN is shifted to the right by the available number of bit positions.

The freed bit positions in the left area of the operand are filled by zeroes when values without signs are shifted. If the specified value has a sign, the free bit positions are filled with the signal state of the sign bit.

The following figure show how the content of an integer data type operand is shifted four bit positions to the right:



Parameters

The following table shows the parameters of the instruction "Shift right":

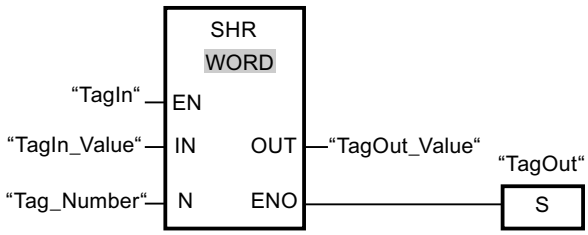
Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers	Bit strings, integers	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Value to be shifted
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Number of bit positions by which the value is shifted
OUT	Output	Bit strings, integers	Bit strings, integers	I, Q, M, D, L	I, Q, M, D, L	Result of the instruction

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
N	Tag_Number	3
OUT	TagOut_Value	0000 0111 1111 0101

If the operand "TagIn" has the signal state "1", the instruction "Shift right" is executed. The content of the operand "TagIn_Value" is shifted three bit positions to the right. The result is sent at output "TagOut_Value". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Selecting the data type of an FBD element (Page 2624)

SHL: Shift left

Description

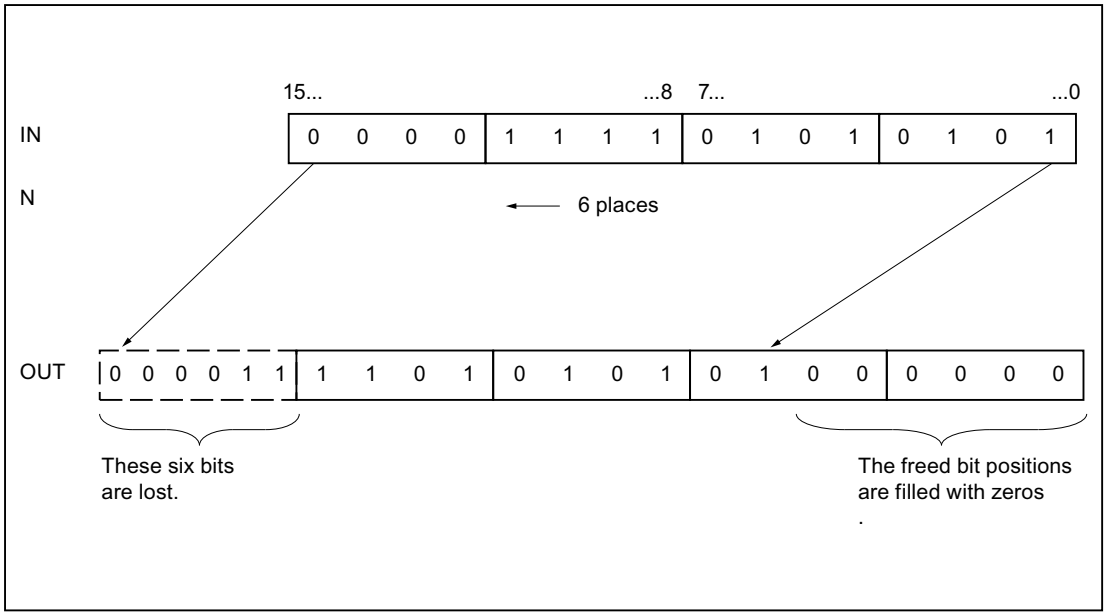
You can use the "Shift left" instruction to shift the content of the operand at the input IN bit-by-bit to the left and query the result at the OUT output. The input N is used to specify the number of bit positions by which the specified value should be moved.

If the value at the input N is "0", the value at input IN is copied unchanged to the operand at output OUT.

If the value at the input N is greater than the number of available bit positions, the operand value at input IN is shifted to the left by the available number of bit positions.

The bit positions in the right part of the operand freed by shifting are filled with zeros.

The following figure show how the content of an operand of the data type WORD is shifted six bit positions to the left:



Parameters

The following table shows the parameters of the instruction "Shift left":

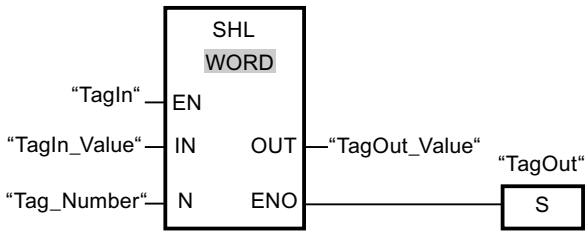
Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers	Bit strings, integers	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Value to be shifted.
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Number of bit positions by which the value is shifted.
OUT	Output	Bit strings, integers	Bit strings, integers	I, Q, M, D, L	I, Q, M, D, L	Result of the instruction

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
N	Tag_Number	4
OUT	TagOut_Value	1111 1010 1111 0000

If the operand "TagIn" has the signal state "1", the instruction "Shift left" is executed. The content of the operand "TagIn_Value" is shifted four bit positions to the left. The result is sent at output "TagOut_Value". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Selecting the data type of an FBD element (Page 2624)

ROR: Rotate right

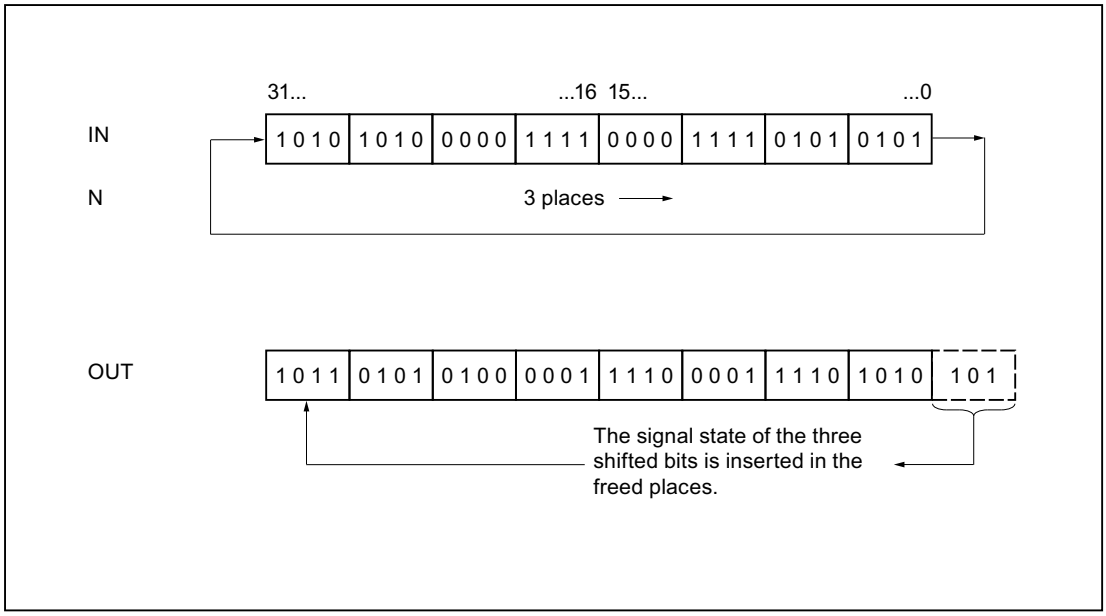
Description

You can use the "Rotate right" instruction to rotate the content of the operand at the input IN bit-by-bit to the right and query the result at the OUT output. The input N is used to specify the number of bit positions by which the specified value should be rotated. The bit positions freed by rotating on the left-hand side are filled true-to-position with the bit positions that are pushed out from the left-hand side.

If the value at the input N is "0", the value at input IN is copied unchanged to the operand at output OUT.

If the value at the parameter N is greater than the number of available bit positions, the operand value at input IN is nevertheless rotated by the specified number of bit positions.

The following figure shows how the content of an operand of the data type DWORD is rotated three bit positions to the right:



Parameters

The following table shows the parameters of the instruction "Rotate right":

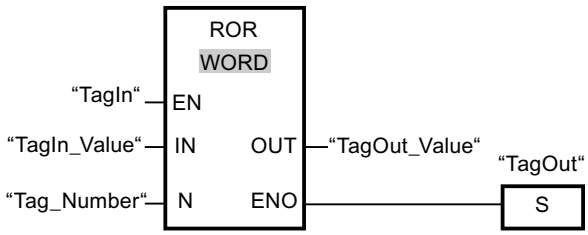
Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers	Bit strings, integers	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Value to be rotated
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Number of bit positions by which the value is rotated
OUT	Output	Bit strings, integers	Bit strings, integers	I, Q, M, D, L	I, Q, M, D, L	Result of the instruction

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	TagIn_Value	0000 1111 1001 0101
N	Tag_Number	5
OUT	TagOut_Value	1010 1000 0111 1100

If the operand "TagIn" has the signal state "1", the instruction "Rotate right" is executed. The content of the operand "TagIn_Value" is rotated five bit positions to the right. The result is sent at output "TagOut_Value". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Selecting the data type of an FBD element (Page 2624)

Inserting operands in FBD instructions (Page 2643)

ROL: Rotate left

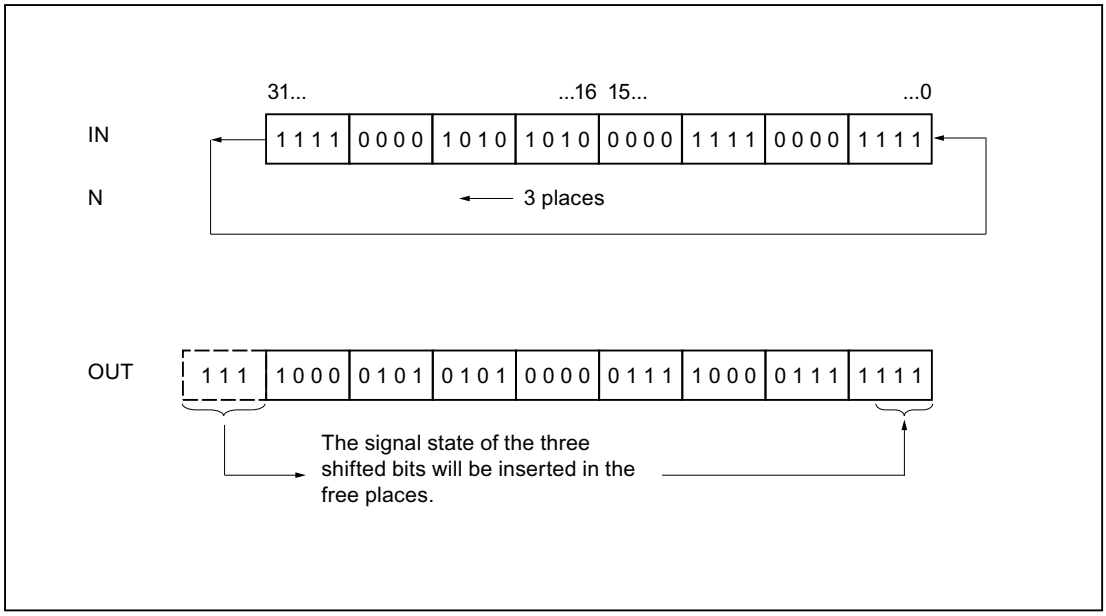
Description

You can use the "Rotate left" instruction to rotate the content of the operand at the input IN bit-by-bit to the left and query the result at the OUT output. The input N is used to specify the number of bit positions by which the specified value should be rotated. The bit positions freed by rotating on the right-hand side are filled true-to-position with the bit positions that are pushed out from the left-hand side.

If the value at the input N is "0", the value at input IN is copied to the operand at output OUT.

If the value at the parameter N is greater than the number of available bit positions, the operand value at input IN is nevertheless rotated by the specified number of bit positions.

The following figure shows how the content of an operand of the data type DWORD is rotated three bit positions to the left:



Parameters

The following table shows the parameters of the instruction "Rotate left":

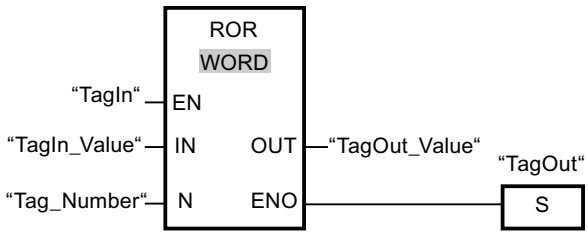
Parameter	Declaration	Data type		Memory area		Description
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	BOOL	I, Q, M, D, L	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers	Bit strings, integers	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Value to be rotated.
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Number of bit positions by which the value is rotated.
OUT	Output	Bit strings, integers	Bit strings, integers	I, Q, M, D, L	I, Q, M, D, L	Result of the instruction

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	TagIn_Value	1010 1000 1111 0110
N	Tag_Number	5
OUT	TagOut_Value	0001 1110 1101 0101

If input "TagIn" has the signal state "1", the instruction "Rotate left" is executed. The content of the operand "TagIn_Value" is rotated five bit positions to the left. The result is sent at output "TagOut_Value". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)
Basics of the EN/ENO mechanism (Page 2447)
Inserting FBD elements (Page 2621)
Editing FBD elements (Page 2635)
Inserting operands in FBD instructions (Page 2643)
Selecting the data type of an FBD element (Page 2624)

Additional instructions

DRUM: Implement sequencer

Description

You can use the "Implement sequencer" instruction to assign the programmed values of the OUT_VAL parameter of the corresponding step to the programmed output bits (OUT1 to OUT16) and the output word (OUT_WORD). The specific step must thereby satisfy the conditions of the programmed enable mask on the S_MASK parameter while the instruction remains at this step. The instruction advances to the next step if the event for the step is true and the programmed time for the current step elapses or if the value at the JOG parameter changes from "0" to "1". The instruction is reset if the signal state on the RESET parameter changes to "1". The current step is hereby equated to the preset step (DSP).

The amount of time spent on a step is determined by the product of the preset timebase (DTBP) and the preset counter value (S_PRESET) for each step. At the start of a new step, this calculated value is loaded into the DCC parameter, which contains the time remaining for the current step. If, for example the value of the DTBP parameter is "2" and the preset value for the first step is "100" (100 ms), the DCC parameter has the value "200" (200 ms).

A step can be programmed with a timer value, an event or both. Steps that have an event bit and the timer value "0" advance to the next step as soon as the signal state of the event bit is "1". Steps that are programmed only with a timer value start the time immediately. Steps that are programmed with an event bit and a timer value greater than "0" start the time when the signal state of the event bit is "1". The event bits are initialized with a signal state of "1".

When the sequencer is on the last programmed step (LST_STEP) and the time for this step has expired, the signal state on the Q parameter is set to "1"; otherwise it is set to "0". When the Q parameter is set, the instruction remains on the step until it is reset.

In the configurable mask (S_MASK), you can selected the individual bits in the output word (OUT_WORD) and set or reset the output bits (OUT1 to OUT16) by means of the output values (OUT_VAL). If a bit of the configurable mask has signal state "1", the OUT_VAL value sets or resets the corresponding bit. If the signal state of a bit of the configurable mask is "0", the corresponding bit is left unchanged. All the bits of the configurable mask for all 16 steps are initialized with a signal state of "1".

The output bit at the OUT1 parameter corresponds to the least significant bit of the output word (OUT_WORD). The output bit at the OUT16 parameter corresponds to the most significant bit of the output word (OUT_WORD).

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the "Implement sequencer" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
RESET	Input	BOOL	I, Q, M, D, L or constant	The signal state "1" indicates the reset condition.
JOG	Input	BOOL	I, Q, M, D, L or constant	When the signal state changes from "0" to "1", the instruction advances to the next step.
DRUM_EN	Input	BOOL	I, Q, M, D, L or constant	The signal state "1" causes the sequencer to advance based on the event and time criteria.
LST_STEP	Input	BYTE	I, Q, M, D, L or constant	Number of the last programmed step
EVENT(i), 1 ≤ i ≤ 16	Input	BOOL	I, Q, M, D, L or constant	Event bit (i); Initial signal state is "1".
OUT(j), 1 ≤ j ≤ 16	Output	BOOL	I, Q, M, D, L	Output bit (j)
Q	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that the time for the last step has elapsed.
OUT_WORD	Output	WORD	I, Q, M, D, L, P	Word address to which the sequencer writes the output values.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
JOG_HIS	Static	BOOL	I, Q, M, D, L or constant	JOG parameter history bit
EOD	Static	BOOL	I, Q, M, D, L or constant	The signal state "1" indicates that the time for the last step has elapsed.

Parameter	Declaration	Data type	Memory area	Description
DSP	Static	BYTE	I, Q, M, D, L, P or constant	Preset step of the sequencer
DSC	Static	BYTE	I, Q, M, D, L, P or constant	Current step of the sequencer
DCC	Static	DWORD	I, Q, M, D, L, P or constant	Current numerical value of the sequencer
DTBP	Static	WORD	I, Q, M, D, L, P or constant	Preset timebase of the sequencer
PREV_TIME	Static	DWORD	I, Q, M, D, L or constant	Previous system time
S_PRESET	Static	ARRAY of WORD	I, Q, M, D, L or constant	Preset count for each step [1 to 16]; where 1 clock pulse = 1 ms.
OUT_VAL	Static	ARRAY of BOOL	I, Q, M, D, L or constant	Output values for each step [1 to 16, 0 to 15].
S_MASK	Static	ARRAY of BOOL	I, Q, M, D, L or constant	Configurable mask for each step [1 to 16, 0 to 15]. Initial signal states are "1".

For additional information on valid data types, refer to "See also".

Parameters ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

ERR_CODE*	Explanation
W#16#0000	No error
W#16#000B	The value at the LST_STEP parameter is less than 1 or greater than 16.
W#16#000C	The value at the DSC parameter is less than 1 or greater than the value of the LST_STEP parameter.
W#16#000D	The value at the DSP parameter is less than 1 or greater than the value of the LST_STEP parameter.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Switching display formats in the program status (Page 2980)

DCAT: Discrete control-timer alarm

Description

The "Discrete control-timer alarm" instruction is used to accumulate the time from the point at which the CMD parameter issued the command to open or close. The time is accumulated until the preset time (PT) is exceeded or the information is received that the device was opened or closed (O_FB or C_FB) within the specified time. If the preset time is exceeded before the information on the opening or closing of the device is received, the corresponding alarm is activated. If the signal state of the command input changes state before the preset time, the time is restarted.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The "Discrete control-timer alarm" instruction has the following reactions to the input conditions:

- When the signal state of the CMD parameter changes from "0" to "1", the signal states of the parameters Q, CMD_HIS, ET (only if ET is < PT) OA and CA are influenced as follows:
  - The parameters Q and CMD_HIS are set to "1".
  - The parameters ET, OA and CA are reset to "0".
- When the signal state of the parameter CMD changes from "1" to "0", the parameters Q, ET (only if ET < PT), OA, CA and CMD_HIS are reset to "0".
- When the signal state of the CMD and CMD_HIS parameters is "1" and the parameter O_FB is set to "0", the time difference (ms) since the last execution of the instruction is added to the value at the parameter ET. If the value of the parameter ET exceeds the value of the parameter PT, the signal state at the parameter OA is set to "1". If the value of the ET parameter does not exceed the value of the PT parameter, the signal state at the OA parameter is reset to "0". The value at the parameter CMD_HIS is reset to the value of the parameter CMD.
- If the signal state of the CMD, CMD_HIS and O_FB parameters are set to "1" and the parameter C_FB has the value "0", the signal state of the parameter OA is set to "0". The value of the parameter ET is set to the value of the parameter PT. If the signal state of the O_FB parameter changes to "0", the alarm is set the next time the instruction is executed. The value of the parameter CMD_HIS is set to the value of the parameter CMD.
- If the CMD, CMD_HIS and C_FB parameters return signal state "0", the time difference (ms) since the last execution of the instruction is added to the value of the ET parameter. If the value of the ET parameter exceeds the value of the PT parameter, the signal state of the parameter CA is reset to "1". If the value at the parameter PT is not exceeded, the parameter CA has the signal state "0". The value of the parameter CMD_HIS is set to the value of the parameter CMD.

- If the parameters CMD, CMD_HIS and O_FB have the signal state "0" and the parameter C_FB is set to "1", the parameter CA is set to "0". The value of the parameter ET is set to the value of the parameter PT. If the signal state of the C_FB parameter changes to "0", the alarm is set the next time the instruction is executed. The value of the parameter CMD_HIS is set to the value of the parameter CMD.
- If the parameters O_FB and C_FB simultaneously have the signal state "1", the signal states of both alarm outputs are set to "1".

The "Discrete control-timer alarm" instruction returns no error information.

Parameters

The following table shows the parameters of the "Discrete control-timer alarm" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
CMD	Input	BOOL	I, Q, M, D, L or constant	The signal state "0" indicates a "close" command. The signal state "1" indicates the "open" command.
O_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when closing
Q	Output	BOOL	I, Q, M, D, L	Shows the status of the parameter CMD
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
ET	Static	DINT	D, L or constant	Currently elapsed time, where one count = 1 ms.
PT	Static	DINT	D, L or constant	Preset timer value, where one count = 1 ms.
PREV_TIME	Static	DWORD	D, L or constant	Previous system time
CMD_HIS	Static	BOOL	D, L or constant	CMD history bit

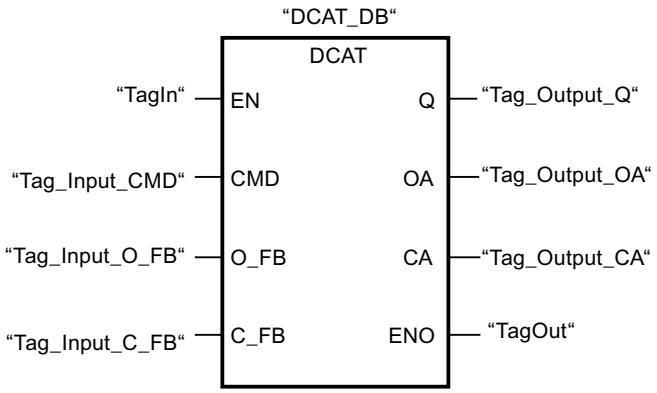
For additional information on valid data types, refer to "See also".

Example

In the following example the parameter CMD changes from "0" to "1". After the execution of the instruction the parameter Q is set to "1" and the two alarm outputs OA and CA have the signal state "0". The parameter CMD_HIS of the instance data block is set to the signal state "1" and the parameter ET is reset to "0".

Note

You can initialize static parameters in the data block.



The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameter	Operand	Value
CMD	Tag_Input_CMD	TRUE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
Q	Tag_Output_Q	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the instance data block "DCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#12
PT	DBD8	L#222
CMD_HIS	DBX16.0	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
Q	Tag_Output_Q	TRUE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the instance data block "DCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#0
CMD_HIS	DBX16.0	TRUE

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

MCAT: Motor control-timer alarm

Description

The instruction "Motor control-timer alarm" is used to accumulate the time from the point in time as of which one of the command inputs (open or close) is activated. The time is accumulated until the preset time is exceeded or the relevant feedback input indicates that the device has executed the requested operation within the specified time. If the preset time is exceeded before the feedback is received, the corresponding alarm is triggered.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The "Motor control-timer alarm" instruction returns no error information.

Execution of the "Motor control-timer alarm" instruction

The following table shows the reactions of the "Motor control-timer alarm" instruction to the various input conditions:

Input parameters								Output parameters								
ET	O_HIS	C_HIS	O_CMD	C_CMD	S_CMD	O_FB	C_FB	OO	CO	OA	CA	ET	O_HIS	C_HIS	Q	Status
X	1	1	X	X	X	X	X	0	0	1	1	PT	0	0	0	Alarm

Input parameters								Output parameters								
X	X	X	X	X	X	1	1	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	1	X	X	0	0	0	0	X	0	0	1	Stop
X	X	X	1	1	X	X	X	0	0	0	0	X	0	0	1	Stop
X	0	X	1	0	0	X	X	1	0	0	0	0	1	0	1	Start opening
<PT	1	0	X	0	0	0	X	1	0	0	0	INC	1	0	1	Open
X	1	0	X	0	0	1	0	0	0	0	0	PT	1	0	1	Opened
>=PT	1	0	X	0	0	0	X	0	0	1	0	PT	1	0	0	Opening alarm
X	X	0	0	1	0	X	X	0	1	0	0	0	0	1	1	Start closing
<PT	0	1	0	X	0	X	0	0	1	0	0	INC	0	1	1	Close
X	0	1	0	X	0	0	1	0	0	0	0	PT	0	1	1	Closed
>=PT	0	1	0	X	0	X	0	0	0	0	1	PT	0	1	0	Closing alarm
X	0	0	0	0	0	X	X	0	0	0	0	X	0	0	1	Stopped

Legend:	
INC	Add the time difference (ms) since the last processing of the FB to ET
PT	PT is set to the same value as ET
X	Cannot be used
<PT	ET < PT
>=PT	ET >= PT

If the input parameters O_HIS and C_HIS both have the signal state "1", they are immediately set to signal state "0". In this case, the last row of the above-named table (X) is valid. Because it is therefore no longer possible to check whether the input parameters O_HIS and C_HIS have the signal state "1", the output parameters are set as follows in this case:

OO = FALSE

CO = FALSE

OA = FALSE

CA = FALSE

ET = PT

Q = TRUE

Parameters

The following table shows the parameters of the "Motor control-timer alarm" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
O_CMD	Input	BOOL	I, Q, M, D, L or constant	"Open" command input
C_CMD	Input	BOOL	I, Q, M, D, L or constant	"Close" command input
S_CMD	Input	BOOL	I, Q, M, D, L or constant	"Stop" command input

Parameter	Declaration	Data type	Memory area	Description
O_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when closing
OO	Output	BOOL	I, Q, M, D, L	"Open" output
CO	Output	BOOL	I, Q, M, D, L	"Close" output
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
Q	Output	BOOL	I, Q, M, D, L	The signal state "0" indicates an error condition.
ET	Static	DINT	D, L or constant	Currently elapsed time, where one count = 1 ms
PT	Static	DINT	D, L or constant	Preset timer value, where one count = 1 ms
PREV_TIME	Static	DWORD	D, L or constant	Previous system time
O_HIS	Static	BOOL	D, L or constant	"Open" history bit
C_HIS	Static	BOOL	D, L or constant	"Close" history bit

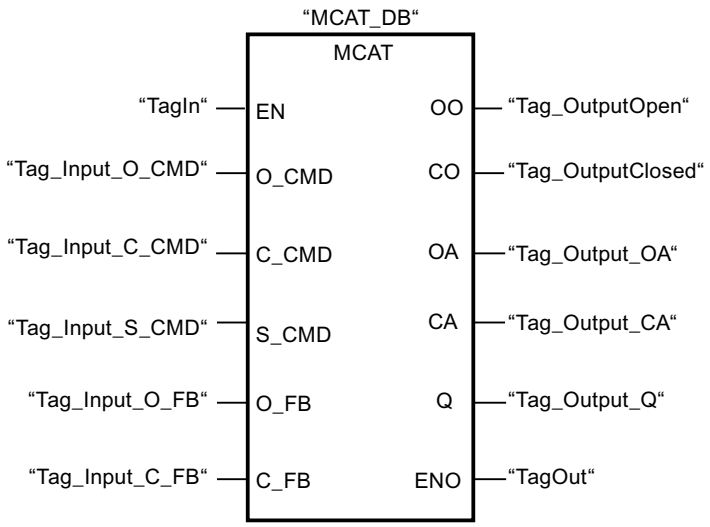
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.



The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameter	Operand	Value
O_CMD	Tag_Input_O_CMD	TRUE
C_CMD	Tag_Input_C_CMD	FALSE
S_CMD	Tag_Input_S_CMD	FALSE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
OO	Tag_OutputOpen	FALSE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	FALSE

The following values are saved in the instance data block "MCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#2
PT	DBD8	L#22
O_HIS	DBX16.0	TRUE
C_HIS	DBX16.1	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OO	Tag_OutputOpen	TRUE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	TRUE

The following values are saved in the instance data block "MCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#0
O_HIS	DBX16.0	TRUE
CMD_HIS	DBX16.1	FALSE

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

IMC: Compare input bits with the bits of a mask

Description

The instruction "Compare input bits with the bits of a mask" is used to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bits of a mask. Up to 16 steps with masks can be programmed. The value of the IN_BIT0 parameter is compared with the value of the mask CMP_VAL[x,0], with "x" indicating the step number. At the CMP_STEP parameter, specify the step number of the mask that is used for the comparison. All programmed values are compared in the same manner. Unprogrammed input bits or unprogrammed bits of the mask have the default signal state FALSE.

If a match is found in the comparison, the signal state of the OUT parameter is set to "1". Otherwise, the OUT parameter is set to "0".

If the value of the CMP_STEP parameter is greater than 15, the instruction is not executed. An error message is output at the ERR_CODE parameter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the "Compare input bits with the bits of a mask" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN_BIT0	Input	BOOL	I, Q, M, D, L or constant	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L or constant	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L or constant	Input bit 2 is compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L or constant	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L or constant	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L or constant	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L or constant	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L or constant	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L or constant	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L or constant	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L or constant	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L or constant	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L or constant	Input bit 12 is compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L or constant	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L or constant	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L or constant	Input bit 15 is compared with bit 15 of the mask.
CMP_STEP	Input	BYTE	I, Q, M, D, L, P or constant	The step number of the mask used for the comparison.
OUT	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that a match was found. The signal state "0" indicates that no match was found.

Parameter	Declaration	Data type	Memory area	Description
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
CMP_VAL	Static	ARRAY OF WORD	I, Q, M, D, L or constant	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

For additional information on valid data types, refer to "See also".

Parameters ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000A	The value at the CMP_STEP parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Switching display formats in the program status (Page 2980)

SMC: Compare scan matrix

Description

The instruction "Compare scan matrix" is used to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bits of the comparison masks for each step. Processing starts at step 1 and is continued until the last programmed step (LAST) or until a match is found. The input bit of the IN_BIT0 parameter is compared with the value of the mask CMP_VAL[x,0], with "x" indicating the step number. All programmed values are compared in the same manner. If a match is found, the signal state of the OUT parameter is set to "1" and the step number with the matching mask is written to the OUT_STEP parameter. Unprogrammed input bits or unprogrammed bits of the mask have the default signal state FALSE. If more than one step has a matching mask, only the first one found is indicated in the OUT_STEP parameter. If no match is found, the signal state of the OUT parameter is set to "0". In this case the value at the OUT_STEP parameter is greater by "1" than the value of the LAST parameter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the "Compare scan matrix" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN_BIT0	Input	BOOL	I, Q, M, D, L or constant	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L or constant	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L or constant	Input bit 2 is compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L or constant	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L or constant	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L or constant	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L or constant	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L or constant	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L or constant	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L or constant	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L or constant	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L or constant	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L or constant	Input bit 12 is compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L or constant	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L or constant	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L or constant	Input bit 15 is compared with bit 15 of the mask.

Parameter	Declaration	Data type	Memory area	Description
OUT	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that a match was found. The signal state "0" indicates that no match was found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
OUT_STEP	Output	BYTE	I, Q, M, D, L, P	Contains the step number with the matching mask or the step number which is greater by "1" than the value of the LAST parameter, provided no match is found.
LAST	Static	BYTE	I, Q, M, D, L, P or constant	Specifies the step number of the last step to be scanned for a matching mask.
CMP_VAL	Static	ARRAY OF WORD	I, Q, M, D, L or constant	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

For additional information on valid data types, refer to "See also".

Parameters ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000E	The value at the LAST parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Switching display formats in the program status (Page 2980)

LEAD_LAG: Lead and lag algorithm

Description

The "Lead and lag algorithm" instruction is used to process signals with an analog tag. The gain value at the GAIN parameter must be greater than zero. The result of the instruction "Lead and lag algorithm" is calculated using the following equation:

$$OUT = \left[\frac{LG_TIME}{LG_TIME + SAMPLE_T}\right] PREV_OUT + GAIN \left[\frac{LD_TIME + SAMPLE_T}{LG_TIME + SAMPLE_T}\right] IN - GAIN \left[\frac{LD_TIME}{LG_TIME + SAMPLE_T}\right] * PREV_IN$$

The "Lead and lag algorithm" instruction supplies plausible results only when processing is in fixed program cycles. The same units must be specified for the LD_TIME, LG_TIME, and SAMPLE_T parameters. At LG_TIME > 4 + SAMPLE_T, the instruction approaches the following function:

OUT = GAIN * ((1 + LD_TIME * s) / (1 + LG_TIME * s)) * IN

When the value of the GAIN parameter is less than or equal to zero, the calculation is not performed and an error information is output on the ERR_CODE parameter.

You can use the "Lead and lag algorithm" instruction in conjunction with loops as a compensator in dynamic feed-forward control. The instruction consists of two operations. The "Lead" operation shifts the phase of the OUT output so that the output leads the input. The "Lag" operation, on the other hand, shifts the output so that the output lags behind the input. Because the "Lag" operation is equivalent to an integration, it can be used as a noise suppressor or as a low-pass filter. The "Lead" operation is equivalent to a differentiation and can therefore be used as a high-pass filter. The two instructions together (Lead and Lag) result in the output phase lagging behind the the input at lower frequencies and leading it at higher frequencies. This means that the "Lead and lag algorithm" instruction can be used as a band pass filter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the "Lead and lag algorithm" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	REAL	I, Q, M, D, L, P or constant	The input value of the current sample time (cycle time) to be processed.
SAMPLE_T	Input	INT	I, Q, M, D, L, P or constant	Sample time

Parameter	Declaration	Data type	Memory area	Description
OUT	Output	REAL	I, Q, M, D, L, P	Result of the instruction
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
LD_TIME	Static	REAL	I, Q, M, D, L, P or constant	Lead time in the same unit as sample time.
LG_TIME	Static	REAL	I, Q, M, D, L, P or constant	Lag time in the same unit as sample time
GAIN	Static	REAL	I, Q, M, D, L, P or constant	Gain as % / % (the ratio of the change in output to a change in input as a steady state).
PREV_IN	Static	REAL	I, Q, M, D, L, P or constant	Previous input
PREV_OUT	Static	REAL	I, Q, M, D, L, P or constant	Previous output

For additional information on valid data types, refer to "See also".

Parameters ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

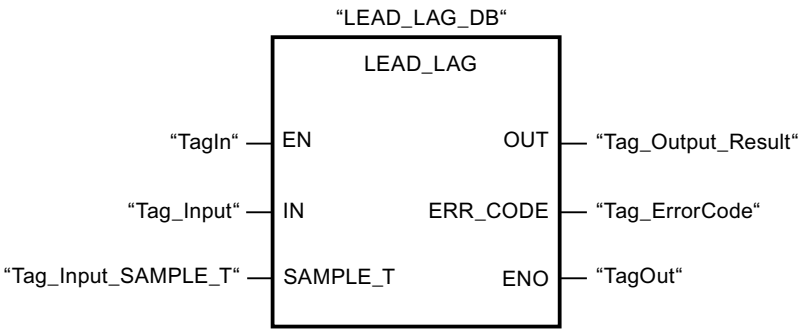
Error code* (W#16#...)	Explanation
0000	No error
0009	The value at the GAIN parameter is less than or equal to zero.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.



The following table shows how the instruction works using specific operand values:

Before processing

In this example the following values are used for the input parameters:

Parameters	Operand	Value
IN	Tag_Input	2.0
SAMPLE_T	Tag_InputSampleTime	10

The following values are saved in the instance data block "LEAD_LAG_DB" of the instruction:

Parameters	Address	Value
LD_TIME	DBD12	2.0
LG_TIME	DBD16	2.0
GAIN	DBD20	1.0
PREV_IN	DBD24	6.0
PREV_OUT	DBD28	6.0

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameters	Operand	Value
OUT	Tag_Output_Result	2.0

The following values are saved in the instance data block "LEAD_LAD_DB" of the instruction:

Parameters	Operand	Value
PREV_IN	DBD24	2.0
PREV_OUT	DBD28	2.0

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

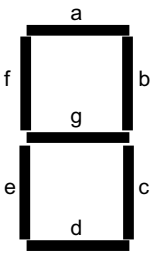
Switching display formats in the program status (Page 2980)

SEG: Create bit pattern for seven-segment display

Description

The instruction "Create bit pattern for seven-segment display" is used to convert each of the four hexadecimal digits of the specified source word (IN) into an equivalent bit pattern for a seven-segment display. The result of the instruction is output in the double word on the OUT parameter.

The following relation exists between the hexadecimal digits and the assignment of the 7 segments (a, b, c, d, e, f, g):

Input digit (Binary)	Assignment of the segments - g f e d c b a	Display (Hexadecimal)	Seven-segment display
0000	00111111	0	
0001	00000110	1	
0010	01011011	2	
0011	01001111	3	
0100	01100110	4	
0101	01101101	5	
0110	01111101	6	
0111	00000111	7	
1000	01111111	8	
1001	01100111	9	
1010	01110111	A	
1011	01111100	B	
1100	00111001	C	
1101	01011110	D	
1110	01111001	E	
1111	01110001	F	

Parameters

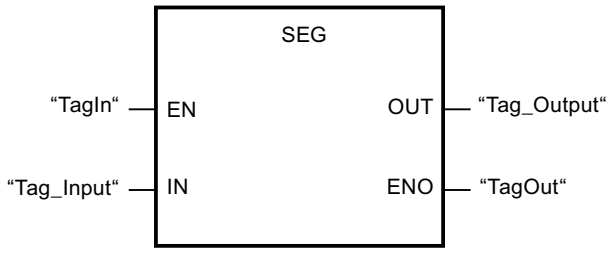
The following table shows the parameters of the "Create bit pattern for seven-segment display" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

Parameter	Declaration	Data type	Memory area	Description
IN	Input	WORD	I, Q, M, D, L, P or constant	Source word with four hexadecimal digits
OUT	Output	DWORD	I, Q, M, D, L, P	Bit pattern for the seven-segment display

Example

The following example shows how the instruction works:



The following table shows how the instruction functions using specific values:

Parameters	Operand	Value	
Hexadecimal	Binary		
IN	Tag_Input	W#16#1234	0001 0010 0011 0100
OUT	Tag_Output	DW#16065B4F66	00000110 01011011 01001111 01100110 Display: 1234

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

BCDCPL: Create tens complement

Description

You can use the "Create tens complement" instruction to create the tens complement of a seven-digit BCD number specified in the IN parameter. This instruction uses the following mathematical formula to calculate:

10000000 (as BCD)

– 7-digit BCD value

Tens complement (as BCD)

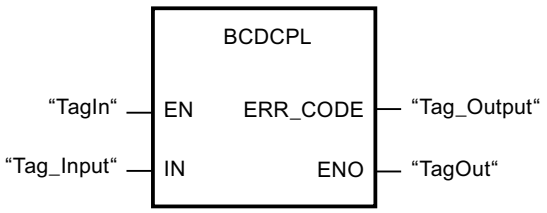
Parameters

The following table shows the parameters of the "Create tens complement" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings	I, Q, M, D, L, P or constant	7-digit BCD number
ERR_CODE	Output	DWORD	I, Q, M, D, L, P	Result of the instruction

Example

The following example shows how the instruction works:



The following table shows how the instruction functions using specific values:

Parameter	Operand	Value*
IN	Tag_Input	DW#16#01234567
ERR_CODE	Tag_Output	DW#16#08765433
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".		

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Switching display formats in the program status (Page 2980)

BITSUM: Count number of set bits

Description

The "Count number of set bits" instruction is used to count the number of bits of an operand that is set to the signal state "1". The operand whose bits are to be counted is specified on the IN parameter. The result of the instruction is output at the RET_VAL parameter.

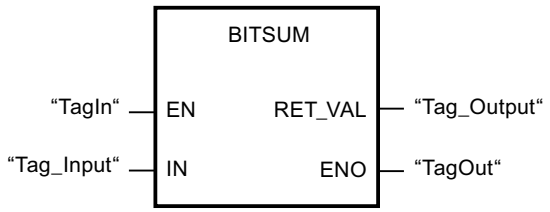
Parameters

The following table shows the parameters of the "Count number of set bits" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	DWORD	I, Q, M, D, L, P or constant	Operand whose set bits are counted.
RET_VAL	Output	INT	I, Q, M, D, L, P	Number of bits to be set

Example

The following example shows how the instruction works:



The following table shows how the instruction functions using specific values:

Parameter	Operand	Value*
IN	Tag_Input	DW#16#12345678
RET_VAL	Tag_Output	W#16#000D (13 Bits)
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".		

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Switching display formats in the program status (Page 2980)

STL

Bit logic operations

R_TRIG: Set tag on positive signal edge

Description

You can use the "Set tag on positive signal edge" instruction to set a specified tag in the instance DB when there is a "0" to "1" change in the result of logic operation (RLO). The instruction compares the current RLO at the input CLK with the RLO from the previous query, which is saved in the specified instance DB. If the instruction detects a change in the result of logic operation (RLO) from "0" to "1", there is a positive, rising edge.

If a positive edge is detected, the tag in the instance DB is set to signal state "1" and the output Q returns the signal state "1" In all other cases, the signal state at the output of the instruction is "0".

When you insert the instruction in the program, the "Call options" dialog opens automatically. In this dialog you can specify whether the edge memory bit is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Set tag on positive signal edge" instruction:

```
CALL R_TRIG, "R_TRIG_DB"
CLK := <Operand>
Q := <Operand>
```

Parameters

The following table shows the parameters of the instruction "Set tag on positive signal edge":

Parameter	Declaration	Data type	Memory area	Description
CLK	Input	BOOL	I, Q, M, D, L or constant	Incoming signal, the edge of which is to be queried.
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:

STL	Explanation
CALL R_TRIG, "R_TRIG_DB"	// The "Set tag on positive signal edge" instruction is called.
CLK := "TagIn"	// Positive signal edge is detected.
Q := "TagOut"	// Signal state "1" on positive signal edge.

The RLO of the preceding query is saved in the instance DB "R_TRIG_DB". If a change in the signal state of the RLO from "0" to "1" is detected in the operand "TagIn", the output "TagOut" has signal state "1".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

F_TRIG: Set tag on negative signal edge

Description

You can use the "Set tag on negative signal edge" instruction to set a specified tag in the instance DB when there is a "1" to "0" change in the result of logic operation (RLO). The instruction compares the current RLO at the input CLK with the RLO from the previous query, which is saved in the specified instance DB. If the instruction detects a change in the result of logic operation (RLO) from "1" to "0", there is a negative, falling edge.

If a negative edge is detected, the tag in the instance DB is set to signal state "1" and the output Q returns the signal state "1" In all other cases, the signal state at the output of the instruction is "0".

When you insert the instruction in the program, the "Call options" dialog opens automatically. In this dialog you can specify whether the edge memory bit is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Set tag on negative signal edge" instruction:

```
CALL F_TRIG, "F_TRIG_DB"
CLK := <Operand>
Q := <Operand>
```

Parameters

The following table shows the parameters of the instruction "Set tag on negative signal edge":

Parameter	Declaration	Data type	Memory area	Description
CLK	Input	BOOL	I, Q, M, D, L or constant	Incoming signal, the edge of which is to be queried.
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:

STL	Explanation
CALL F_TRIG, "F_TRIG_DB"	// The "Set tag on negative signal edge" instruction is called.
CLK := "TagIn"	// Negative signal edge is detected.
Q := "TagOut"	// Signal state "1" on negative signal edge.

The RLO of the preceding query is saved in the instance DB "R_TRIG_DB". If a change in the signal state of the RLO from "1" to "0" is detected in the operand "TagIn", the output "TagOut" has signal state "1".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Timer operations

TP: Generate pulse

Description

You can use the "Generate pulse" instruction to set the Q output for a programmed duration. The instruction is started when the result of logic operation (RLO) at the IN parameter changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. The Q parameter is set for the time PT, regardless of the subsequent changes in the input signal. Even when a new positive signal edge is detected, the signal state of the Q parameter is not affected as long as time PT is running.

You can query the current time value at the ET output. The time value starts at T#0s and ends when the value of the time PT is reached. If the time duration PT is reached and the signal state at input IN is "0", the ET output is reset.

You can use the "Call block" (CALL) instruction in the program code to call the "Generate pulse" instruction.

Note

If the timer is not called in the program because it is skipped, for example, the ET output returns a constant value as soon as the time has expired.

Each call of the "Generate pulse" instruction must be assigned an IEC timer in which the instruction data is stored. An IEC timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TP_TIME or TP_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TP_TIME or TP_LTIME in the "Static" section of a block (for example, #MyTP_TIMER)

After the selection of the data type from the drop-down list "???", the "Call options" dialog opens in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when the instruction is called and also each time the outputs Q or ET are accessed.

The operating system resets the instances of the "Generate pulse" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized in a startup OB with the value "0" set for the PT parameter. If instances of the "Generate pulse" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Syntax

The following syntax is used for the "Generate pulse" instruction:

```
CALL TP, "IEC timer"
<Data type>
IN := <Operand>
PT := <Operand>
Q := <Operand>
ET := <Operand>
```

Parameters

The following table shows the parameters of the "Generate pulse" instruction:

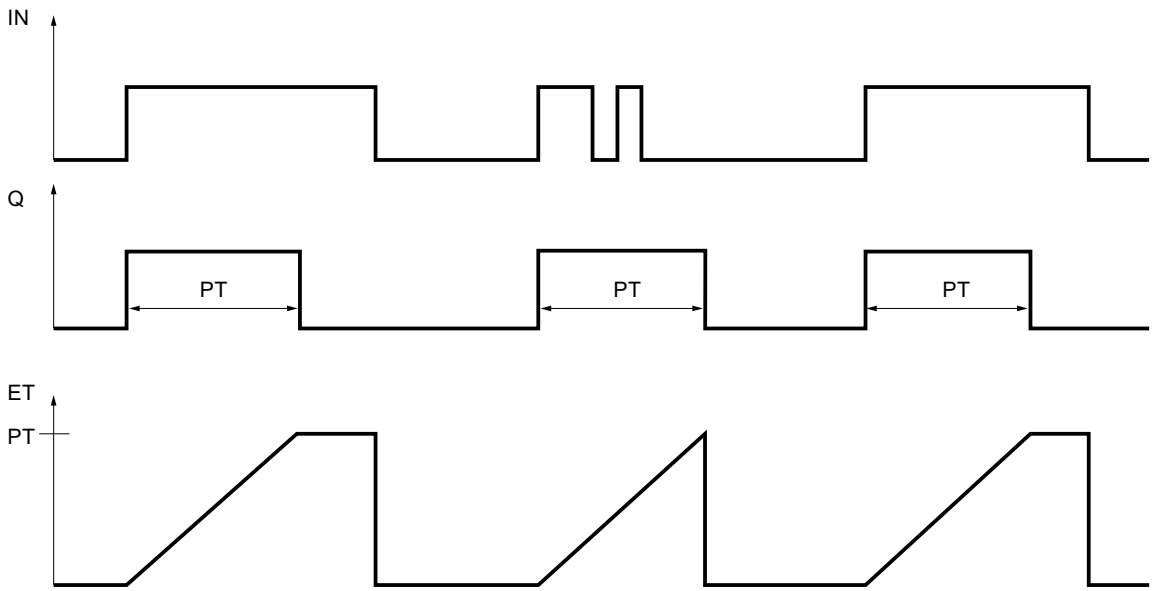
Parameter	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L, P or constant	Start input
PT	Input	TIME, LTIME	I, Q, M, D, L, P or constant	Duration of the pulse. The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L, P	Pulse output
ET	Output	TIME, LTIME	I, Q, M, D, L, P	Current time value

You can select the data type of the instruction from the "<???>" drop-down list.

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the behavior of the "Generate pulse" instruction after the start:



Example

The following example shows how the instruction works:

STL	Explanation
CALL TP, "TP_DB"	// The "Generate pulse" instruction is called. The "TP_DB" data block is assigned to the instruction.
TIME	// Data type of the instruction
IN := "Tag_Start"	// The instruction executes on a rising signal edge of the operand.
PT := "Tag_PresetTIME"	// Duration of the pulse
Q := "Tag_Status"	// The operand is set for the duration that was specified by the "Tag_PresetTIME" operand.
ET := "Tag_ElapsedTIME"	// Current time value

See also

Overview of the valid data types (Page 2236)
Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)
Call function blocks as single or multi-instances (Page 2182)
Basic information on the status word (Page 2466)
Querying and setting status bits in STL (Page 2467)

TON: Generate on-delay

Description

You can use the "Generate on-delay" instruction to delay setting of the Q output by the programmed time PT. The instruction is started when the result of logic operation (RLO) at the IN parameter changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. When the time PT has elapsed, the Q parameter has the signal state "1". The Q parameter remains set as long as the start input IN is still "1". When the signal state at the start input changes from "1" to "0", the Q parameter is reset. The timer function is started again when a new positive signal edge is detected at the start input.

You can query the current time value at the ET output. The time value starts at T#0s and ends when the value of the time PT is reached. The ET parameter is reset as soon as the signal state at the IN parameter changes to "0".

Note

If the timer is not called in the program because it is skipped, for example, the ET output returns a constant value as soon as the time has expired.

In the program code, you call the "Generate on-delay" instruction with the "Call block" (CALL) instruction.

Each call of the "Generate on-delay" instruction must be assigned an IEC timer in which the instruction data is stored. An IEC timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TON_TIME or TON_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TON_TIME or TON_LTIME in the "Static" section of a block (for example, #MyTON_TIMER)

After the selection of the data type from the drop-down list "???", the "Call options" dialog opens in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when the instruction is called and also each time the outputs Q or ET are accessed.

The operating system resets the instances of the "Generate on-delay" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized in a startup OB with the value "0" set for the PT parameter. If instances of the "Generate on-delay" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Syntax

The following syntax is used for the "Generate on-delay" instruction:

```
CALL TON, "IEC timer"
<Data type>
IN := <Operand>
PT := <Operand>
Q := <Operand>
ET := <Operand>
```

Parameters

The following table shows the parameters of the "Generate on-delay" instruction:

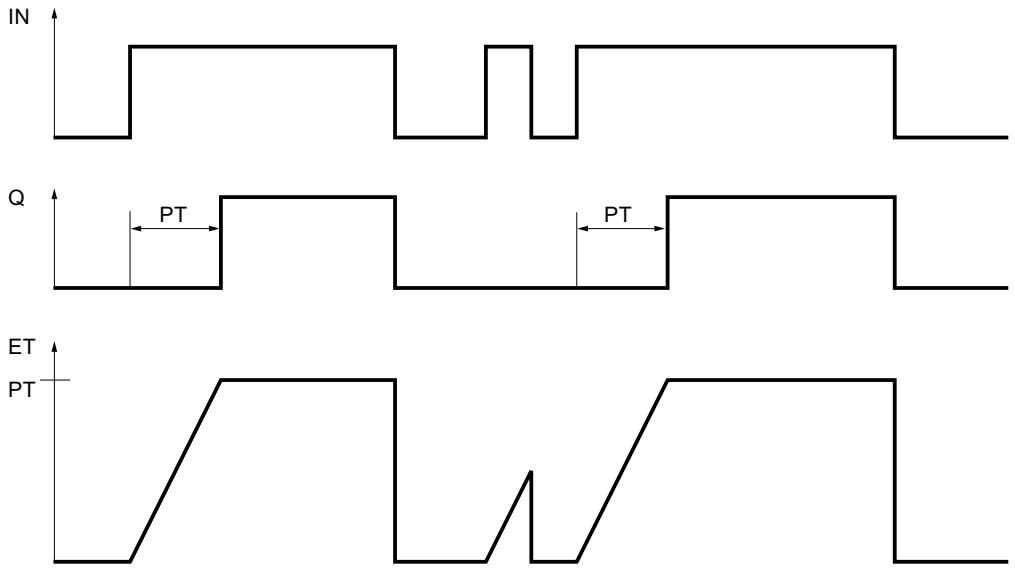
Parameter	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L, P or constant	Start input
PT	Input	TIME, LTIME	I, Q, M, D, L, P or constant	Duration of the on-delay The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L, P	Signal state that is delayed by the time PT.
ET	Output	TIME, LTIME	I, Q, M, D, L, P	Current time value

You can select the data type of the instruction from the "<???>" drop-down list.

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the behavior of the "Generate on-delay" instruction after the start:



Example

The following example shows how the instruction works:

STL	Explanation
CALL "TON", "TON_DB"	// The "Generate on-delay" instruction is called. The "TON_DB" data block is assigned to the instruction.
IN := "Tag_Start"	// The instruction executes on a rising signal edge of the operand.
PT := "Tag_PresetTIME"	// Specifies the time by which the rising signal edge of the IN parameter is delayed.
Q := "Tag_Status"	// The operand is set if the duration PT that was specified by the "Tag_PresetTIME" tag has expired.
	// The Q parameter remains set as long as the "Tag_Start" variable is "1".
	// When the signal state at the start input changes from "1" to "0", the operand of the Q parameter is reset.
ET := "Tag_ElapsedTIME"	// Current time value

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

TOF: Generate off-delay

Description

You can use the "Generate off-delay" instruction to delay resetting of the Q output by the programmed time PT. The Q parameter is set when the result of logic operation (RLO) of the IN parameter changes from "0" to "1" (positive signal edge). When the signal state at the IN parameter changes back to "0", the programmed time PT starts. The Q parameter remains set as long as time PT is running. When the time PT expires, the Q parameter is reset. If the signal state of the IN parameter changes to "1" before the time PT expires, the timer is reset. The signal state of the Q parameter remains set to "1".

The current time value can be queried in the ET parameter. The time value starts at T#0s and ends when the value of the time PT is reached. When the time PT expires, the ET parameter remains set to the current value until the IN parameter changes back to "1". If input IN changes to "1" before the time PT elapses, the ET output is reset to the value T#0s.

Note

If the timer is not called in the program because it is skipped, for example, output ET returns a constant value as soon as the time has expired.

In the program code, you call the "Generate off-delay" instruction with the "Call block" (CALL) instruction.

Each call of the "Generate off-delay" instruction must be assigned an IEC timer in which the instruction data is stored. An IEC timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TOF_TIME or TOF_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TOF_TIME or TOF_LTIME in the "Static" section of a block (for example, #MyTOF_TIMER)

After the selection of the data type from the drop-down list "???", the "Call options" dialog opens in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when the instruction is called and also each time the outputs Q or ET are accessed.

The operating system resets the instances of the "Generate off-delay" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized in a startup OB with the value "0" set for the PT parameter. If instances of the "Generate off-delay" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Syntax

The following syntax is used for the "Generate off-delay" instruction:

```
CALL TOF, "IEC timer"
<Data type>
IN := <Operand>
PT := <Operand>
Q := <Operand>
ET := <Operand>
```

Parameters

The following table shows the parameters of the "Generate off-delay" instruction:

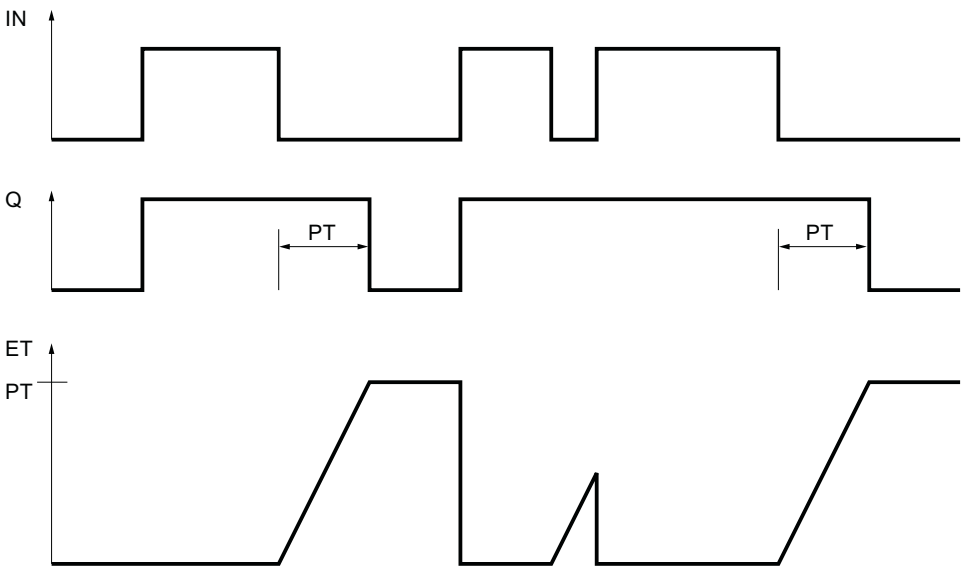
Parameter	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L, P or constant	Start input
PT	Input	TIME, LTIME	I, Q, M, D, L, P or constant	Duration of the off delay The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L, P	Signal state that is delayed by the time PT.
ET	Output	TIME, LTIME	I, Q, M, D, L, P	Current time value

You can select the data type of the instruction from the "<???>" drop-down list.

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the behavior of the "Generate off-delay" instruction after the start:



Example

The following example shows how the instruction works:

STL	Explanation
CALL TOF, "TOF_DB"	// The "Generate off-delay" instruction is called. The "TOF_DB" data block is assigned to the instruction.
TIME	// Data type of the instruction
IN := "Tag_Start"	// The instruction executes on a rising signal edge of the operand.
PT := "Tag_PresetTIME"	// Specifies the time by which the falling signal edge of the IN parameter is delayed.
Q := "Tag_Status"	// The operand is set when the instruction is started by a rising signal edge of the IN parameter.
	// When the value at the IN parameter changes from "1" to "0", the "Tag_Status" operand remains set as long as the time specified by the "Tag_PresetTIME" tag is running.
	// When the time of the PT parameter expires, the operand is reset.
ET := "Tag_ElapsedTIME"	// Current time value

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

TONR: Time accumulator

Description

The "Time accumulator" instruction is used to accumulate time values within a period set by the PT parameter. When the signal state at the IN input changes from "0" to "1" (positive signal edge), the instruction is executed and time PT starts. While the time PT is running, the time values are accumulated that are recorded when the IN input has the signal state "1". The accumulated time is written to the ET output and can be queried there. When the time PT expires, the output Q has the signal state "1". The Q parameter remains set to "1", even when the signal state at the IN parameter changes from "1" to "0" (negative signal edge).

The R input resets the ET and Q outputs regardless of the signal state at the start input.

In the program code, you call the "Time accumulator" instruction with the "Call block" (CALL) instruction.

Each call of the "Time accumulator" instruction must be assigned an IEC timer in which the instruction data is stored. An IEC timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TONR_TIME or TONR_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TONR_TIME or TONR_LTIME in the "Static" section of a block (for example, #MyTONR_TIMER)

After the selection of the data type from the drop-down list "???", the "Call options" dialog opens in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when the instruction is called and also each time the outputs Q or ET are accessed.

The execution of the "Time accumulator" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Syntax

The following syntax is used for the "Time accumulator" instruction:

```
CALL TONR, "IEC timer"
<Data type>
IN := <Operand>
```

```
R := <Operand>
PT := <Operand>
Q := <Operand>
ET := <Operand>
```

Parameters

The following table shows the parameters of the "Time accumulator" instruction:

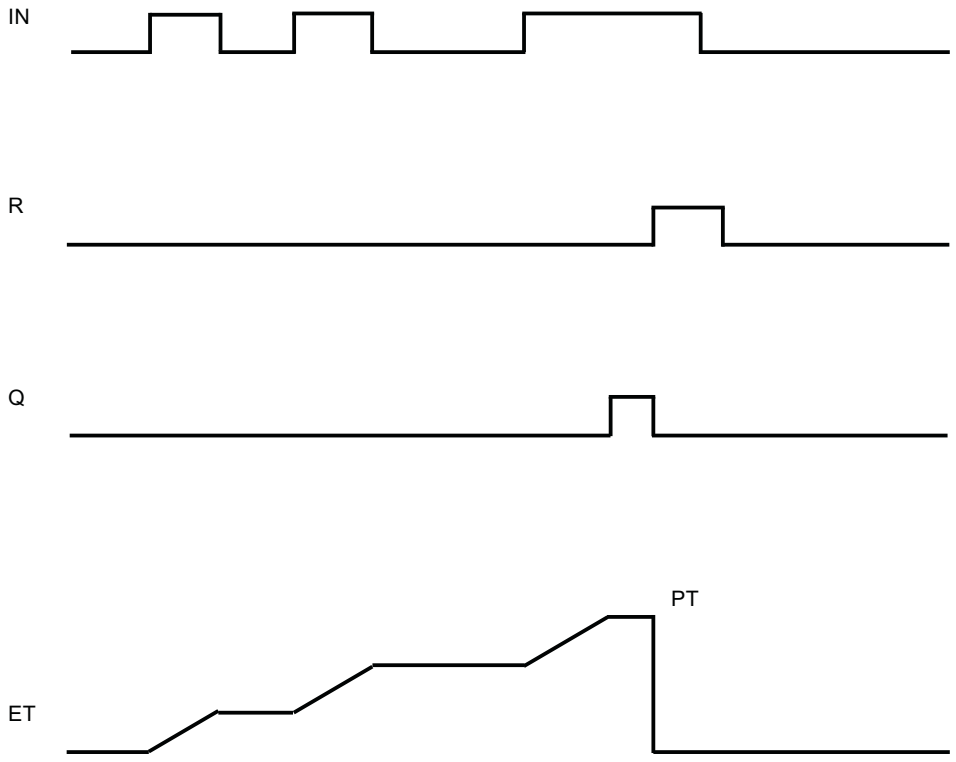
Parameter	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L, P or constant	Start input
R	Input	BOOL	I, Q, M, D, L, P or constant	Reset input
PT	Input	TIME, LTIME	I, Q, M, D, L, P or constant	Maximum duration of time recording The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L, P	Output that is set when the time PT expires.
ET	Output	TIME, LTIME	I, Q, M, D, L, P	Accumulated time

You can select the data type of the instruction from the "<???>" drop-down list.

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Time accumulator" instruction:



Example

The following example shows how the instruction works:

STL	Explanation
CALL TONR, "TONR_DB"	// The "Time accumulator" instruction is called. The "TONR_DB" data block is assigned to the instruction.
TIME	// Data type of the instruction
IN := "Tag_Start"	// The instruction executes on a rising signal edge of the operand and the time duration at the PT input is started.
	While the time PT is running, the time values are accumulated that are recorded when the IN input has the signal state "1".
R := "Tag_Reset"	// The R input resets the outputs ET and Q regardless of the signal state at the start input.
PT := "Tag_PresetTIME"	// Specifies how long the time values are accumulated.
Q := "Tag_Status"	// The operand is set if the time at PT has expired.
	// The Q parameter remains set to "1", even when the signal state at the IN parameter changes from "1" to "0" (negative signal edge).

STL	Explanation
ET := "Tag_ElapsedTIME"	// The accumulated time is written to output ET and can be queried there.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

RESET_TIMER: Reset timer

Description

You can use the "Reset timer" instruction to reset an IEC timer to "0". The structure components of the timer in the specified data block are reset to "0".

The instruction does not influence the RLO. At the TIMER parameter, the "Reset timer" instruction is assigned an IEC timer declared in the program.

The instruction data is updated only when the instruction is called and not each time the assigned IEC timer is accessed. Querying the data is only identical from the call of the instruction to the next call of the instruction.

Syntax

The following syntax is used for the "Reset timer" instruction:

```
CALL RESET_TIMER
<Data type>
TIMER := <Operand>
```

Parameters

The following table shows the parameters of the "Reset timer" instruction:

Parameter	Declaration	Data type	Memory area	Description
TIMER	Output	IEC_TIMER, IEC_LTIMER, TP_TIME, TP_LTIME, TON_TIME, TON_LTIME, TOF_TIME, TOF_LTIME, TONR_TIME, TONR_LTIME	D, L	IEC timer that is reset

You can select the data type of the instruction from the "<???>" drop-down list.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL TON, "TON_DB"           // The "Generate on-delay" instruction is called. The
                             "TON_DB" instance data block is assigned to the
                             instruction.
time_type := TIME            // Data type of the instruction
IN := "Tag_Start"            // The instruction executes on a rising signal edge of
                             the operand "Tag_Start".
PT := "Tag_PresetTIME"       // The IEC timer stored in the instance data block
                             "TON_DB" is started with the time duration that is
                             specified by the operand "Tag_PresetTIME".
Q := "Tag_Status"            // The operand "Tag_Status" is set if the time
                             duration PT that was specified by the
                             "Tag_PresetTIME" operand has expired.
                             // The parameter Q remains set as long as the
                             "Tag_Start" operand still has the signal state "1".
                             // When the signal state at the start input changes
                             from "1" to "0", the operand of the Q parameter is
                             reset.
ET := "Tag_ElapsedTIME"      // Current time value

A "Tag_Input_1"              // When the operand "Tag_Input_1" and
A "Tag_Input_2"              // The operand "Tag_Input_2" returns the signal state
                             "1",

CALL RESET_TIMER             // The "Reset timer" instruction is called.
timer_type := IEC_TIMER      // Data type of the IEC timer
TIMER := "TON_DB"            // IEC timer is reset.
```

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

PRESET_TIMER: Load time duration

Description

You can use the "Load time duration" instruction to set the time for an IEC timer. The instruction is executed in each cycle. The instruction writes the specified time to the structure of the specified IEC timer.

You assign an IEC timer declared in the program to the "Load time duration" instruction.

The instruction data is updated only when the instruction is called and each time the assigned IEC timer is accessed. The query on Q or ET (for example, "MyTimer".Q or "MyTimer".ET) updates the IEC_TIMER structure.

Note

If the specified IEC timer is running while the instruction executes, the instruction overwrites the current time of the specified IEC timer. This can change the timer status of the IEC timer.

Syntax

The following syntax is used for the "Load time duration" instruction:

```
CALL PRESET_TIMER
<Data type> <Data type
PT := <Operand>
TIMER := <Operand>
```

Parameter

The following table shows the parameters of the "Load time duration" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Time duration>	Input	TIME, LTIME	I, Q, M, D, L or constant	Duration with which the IEC timer runs
<IEC timer>	Output	IEC_TIMER, IEC_LTIMER, TP_TIME, TP_LTIME, TON_TIME, TON_LTIME, TOF_TIME, TOF_LTIME, TONR_TIME, TONR_LTIME	D, L	IEC timer whose duration is set

You can select the data type of the instruction from the "<???>" drop-down list.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
CALL TON, "TON_DB"	// The "Generate on-delay" instruction is called. The "TON_DB" instance data block is assigned to the instruction.
time_type := TIME	// Data type of the instruction
IN := "Tag_Start"	// The instruction executes on a rising signal edge of the operand "Tag_Start".
PT := "Tag_PresetTIME"	// The IEC timer stored in the instance data block "TON_DB" is started with the time duration that is specified by the operand "Tag_PresetTIME".
Q := "Tag_Status"	// The operand "Tag_Status" is set if the time duration PT that was specified by the "Tag_PresetTIME" operand has expired.
	// The parameter Q remains set as long as the "Tag_Start" operand still has the signal state "1".
	// When the signal state at the start input changes from "1" to "0", the operand of the parameter Q is reset.
ET := "Tag_ElapsedTIME"	// Current time value
A "Tag_Input_1"	// When the operand "Tag_Input_1" and
A "Tag_Input_2"	// The operand "Tag_Input_2" returns the signal state "1",

STL	Explanation
CALL PRESET_TIMER	// The "Time accumulator" instruction is called.
time_type := TIME	// Data type of the instruction
timer_type := IEC_TIMER	// Data type of the IEC timer
PT := "Tag_PresetTIME_new"	// The instruction writes the time duration "Tag_PresetTIME_new" in the instance data block "TON_DB", thereby overwriting the time value of the operand "Tag_PresetTIME" within the instance data block. The signal state of the timer status may therefore change at the next query or when "MyTimer".Q or "MyTimer".ET are accessed.
TIMER := "TON_DB"	// IEC timer is reset.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Counter operations

CTU: Count up

Description

You can use the "Count up" instruction to increment the value at the CV parameter. When the signal state at the CU parameter changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value of the CV parameter is incremented by one. When the instruction is executed for the first time the current count of the CV parameter is set to zero. The counter value is incremented each time a positive signal edge is detected, until it reaches the high limit for the data type specified at the CV output. When the high limit is reached, the signal state of the CU parameter no longer has an effect on the instruction.

You can query the count status of the Q parameter. The signal state of the Q parameter is determined by the PV parameter. When the current counter value is greater than or equal to the value of the PV parameter, the Q parameter is set to signal state "1". In all other cases, the signal state of the Q parameter is "0".

The value of the CV parameter is reset to zero when the signal state at the R parameter changes to "1". As long as the R parameter has the signal state "1", the signal state of the CU parameter has no effect on the instruction.

In the program code, you call the "Count up" instruction with the "Call block" (CALL) instruction.

Note

Only use a counter at a single location in the program to avoid counting errors.

Each call of the "Count up" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

Data block of system data type IEC_<Counter> (Shared DB)

- IEC_SCOUNTER / IEC_USCOUNTER
- IEC_COUNTER / IEC_UCOUNTER
- IEC_DCOUNTER / IEC_UDCOUNTER
- IEC_LCOUNTER / IEC_ULCOUNTER

Local tag

- CTU_SINT / CTU_USINT
- CTU_INT / CTU_UINT
- CTU_DINT / CTU_UDINT
- CTU_LINT / CTU_ULINT
- IEC_SCOUNTER / IEC_USCOUNTER
- IEC_COUNTER / IEC_UCOUNTER
- IEC_DCOUNTER / IEC_UDCOUNTER
- IEC_LCOUNTER / IEC_ULCOUNTER

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER")
- Declaration as a local tag of the type CTU_<Data type> or IEC_<Counter> in the "Static" section of a block (for example #MyCTU_COUNTER)

After the selection of the data type from the drop-down list "???", the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

When you set up the IEC counter in a separate data block (single instance), the instance data block is created by default with "optimized block access" and the individual tags are defined as retentive. For additional information on setting retentivity in an instance data block, refer to "See also".

When you set up the IEC counter as local tag (multiple instance) in a function block with "optimized block access", it is defined as retentive in the block interface.

The operating system resets the instances of the "Count up" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized in a startup OB with the value "1" set for the R parameter of the instruction. If instances of the "Count up" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Syntax

The following syntax is used for the "Count up" instruction:

```
CALL CTU, "IEC counters"
<Data type>
CU := <Operand>
R := <Operand>
PV := <Operand>
Q := <Operand>
CV := <Operand>
```

Parameters

The following table shows the parameters of the "Count up" instruction:

Parameters	Declaration	Data type	Memory area	Description
CU	Input	BOOL	I, Q, M, D, L or constant	Count input
R	Input	BOOL	I, Q, M, D, L, P or constant	Reset input
PV	Input	Integers	I, Q, M, D, L, P or constant	Value at which the Q output is set.
Q	Output	BOOL	I, Q, M, D, L	Counter status
CV	Output	Integers, CHAR, DATE	I, Q, M, D, L, P	Current counter value

You can select the data type of the instruction from the "<???>" drop-down list.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
CALL CTU, "CTU_DB"	// The "Count up" instruction is called. The "CTU_DB" data block is assigned to the instruction.
INT	// Data type of the instruction
CU := "Tag_StartCTU"	// When the signal state at the "Tag_StartCTU" operand changes from "0" to "1", the instruction is executed

STL	Explanation
	and the current counter value of the "Tag_CounterValue" operand is incremented by one.
	// The counter value is incremented until the high limit of INT = 32767 is reached.
R := "Tag_ResetCounter"	// When the signal state at the "Tag_ResetCounter" operand changes to "1", the "Tag_CounterValue" operand is reset to "0".
PV := "Tag_PresetValue"	// The operand determines the value at which the operand of the Q parameter is set.
Q := "Tag_CounterStatus"	// The operand is set as long as the current counter value is greater than or equal to the value of the PV parameter.
CV := "Tag_CounterValue"	// Current counter value

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Setting retentivity in an instance data block (Page 2839)

Setting the retentivity of local tags (Page 2561)

CTD: Count down

Description

The "Count down" instruction is used to decrement the value at the parameter CV. When the signal state at the CD parameter changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value of the CV parameter is decremented by one. When the instruction is executed the first time, the counter value of the CV parameter is set to the value of the PV parameter. Each time a positive signal edge is detected, the counter value is decremented until it reaches the low limit of the specified data type. When the low limit is reached, the signal state of the CD parameter no longer has an effect on the instruction.

You can query the count status of the Q parameter. If the current counter value is less than or equal to zero, the Q parameter is set to signal state "1". In all other cases, the signal state of the Q parameter is "0".

The value of the CV parameter is set to the value of the PV parameter when the signal state of the LD parameter changes to "1". As long as the LD parameter has the signal state "1", the signal state of the CD parameter has no effect on the instruction.

In the program code, you call the "Count down" instruction with the "Call block" (CALL) instruction.

Note

Only use a counter at a single location in the program to avoid counting errors.

Each call of the "Count down" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

Data block of system data type IEC_<Counter> (Shared DB)

- IEC_SCOUNTER / IEC_USCOUNTER
- IEC_COUNTER / IEC_UCOUNTER
- IEC_DCOUNTER / IEC_UDCOUNTER
- IEC_LCOUNTER / IEC_ULCOUNTER

Local tag

- CTD_SINT / CTD_USINT
- CTD_INT / CTD_UINT
- CTD_DINT / CTD_UDINT
- CTD_LINT / CTD_ULINT
- IEC_SCOUNTER / IEC_USCOUNTER
- IEC_COUNTER / IEC_UCOUNTER
- IEC_DCOUNTER / IEC_UDCOUNTER
- IEC_LCOUNTER / IEC_ULCOUNTER

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER")
- Declaration as a local tag of the type CTD_<Data type> or IEC_<Counter> in the "Static" section of a block (for example #MyCTD_COUNTER)

After the selection of the data type from the drop-down list "???", the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

When you set up the IEC counter in a separate data block (single instance), the instance data block is created by default with "optimized block access" and the individual tags are defined as retentive. For additional information on setting retentivity in an instance data block, refer to "See also".

When you set up the IEC counter as local tag (multiple instance) in a function block with "optimized block access", it is defined as retentive in the block interface.

The operating system resets the instances of "Count down" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized with the value "1" for the LD parameter of the instruction in a startup OB. In this case the desired initial value for the CV parameter is specified in the PV parameter. If instances of the "Count down" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Syntax

The following syntax is used for the "Count down" instruction:

```
CALL CTD, "IEC counters"
<Data type>
CD := <Operand>
LD := <Operand>
PV := <Operand>
Q := <Operand>
CV := <Operand>
```

Parameters

The following table shows the parameters of the "Count down" instruction:

Parameter	Declaration	Data type	Memory area	Description
CD	Input	BOOL	I, Q, M, D, L or constant	Count input
LD	Input	BOOL	I, Q, M, D, L, P or constant	Load input
PV	Input	Integers	I, Q, M, D, L, P or constant	Value at which the Q output is set.
Q	Output	BOOL	I, Q, M, D, L	Counter status
CV	Output	Integers, CHAR, DATE	I, Q, M, D, L, P	Current counter value

You can select the data type of the instruction from the "<???>" drop-down list.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL CTD, "CTD_DB"           // The "Count down" instruction is called. The
                             "CTD_DB" data block is assigned to the instruction.
INT                          // Data type of the instruction
```

STL	Explanation
CD := "Tag_StartCTD"	// When the signal state at the "Tag_StartCTD" operand changes from "0" to "1", the instruction is executed and the current counter value of the "Tag_CounterValue" operand is decremented by one. // The counter value of the CV parameter is decremented until the low limit of INT = -32768 is reached.
LD := "Tag_LoadPV"	// When the signal state at the "Tag_LoadPV" operand changes to "1", the "Tag_CounterValue" operand is set to the value of the "Tag_PresetValue" operand.
PV := "Tag_PresetValue"	// Specifies the value to which the counter is set when the signal state is "1" in the LD parameter.
Q := "Tag_CounterStatus"	// The operand is set when the current counter value is less than or equal to zero.
CV := "Tag_CounterValue"	// Current counter value

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Setting retentivity in an instance data block (Page 2839)

Setting the retentivity of local tags (Page 2561)

CTUD: Count up and down

Description

Use the "Count up and down" instruction to increment or decrement the counter value at the CV parameter. If the signal state of the CU parameter changes from "0" to "1" (positive signal edge), the current counter value is incremented by one and stored in the CV parameter. When the signal state at the CD parameter changes from "0" to "1" (positive signal edge), the counter value of the CV parameter is decremented by one. If there is a positive signal edge at the CU and CD inputs in a program cycle, the current counter value of the CV parameter remains unchanged.

The counter value can be incremented until it reaches the high limit of the data type specified for the CV parameter. When the high limit is reached, the counter value is no longer incremented on a positive signal edge. The counter value is no longer decremented once the low limit of the specified data type has been reached.

When the signal state at the LD parameter changes to "1", the counter value of the CV parameter is set to the value of the PV parameter. As long as the LD parameter has signal state "1", the signal state of the CU and CD inputs has no effect on the instruction.

The counter value is set to zero when the signal state of the R parameter changes to "1". As long as the R parameter has signal state "1", a change in the signal state of the CU, CD and LD parameters has no effect on the "Count up and down" instruction.

You can query the status of the up counter at the QU parameter. When the current counter value is greater than or equal to the value of the PV parameter, the QU parameter is set to signal state "1". In all other cases, the signal state of the QU parameter is "0". You can also specify a constant for the PV parameter.

You can query the status of the down counter at the QD parameter. If the current counter value is less than or equal to zero, the QD parameter is set to signal state "1". In all other cases, the signal state of the QD parameter is "0".

In the program code, you call the "Count up and down" instruction with the "Call block" (CALL) instruction.

Note

Only use a counter at a single location in the program to avoid counting errors.

Each call of the "Count up and down" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

Data block of system data type IEC_<Counter> (Shared DB)

- IEC_SCOUNTER / IEC_USCOUNTER
- IEC_COUNTER / IEC_UCOUNTER
- IEC_DCOUNTER / IEC_UDCOUNTER
- IEC_LCOUNTER / IEC_ULCOUNTER

Local tag

- CTUD_SINT / CTUD_USINT
- CTUD_INT / CTUD_UINT
- CTUD_DINT / CTUD_UDINT
- CTUD_LINT / CTUD_ULINT
- IEC_SCOUNTER / IEC_USCOUNTER
- IEC_COUNTER / IEC_UCOUNTER
- IEC_DCOUNTER / IEC_UDCOUNTER
- IEC_LCOUNTER / IEC_ULCOUNTER

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER")
- Declaration as a local tag of the type CTUD_<Data type> or IEC_<Counter> in the "Static" section of a block (for example #MyCTUD_COUNTER)

After the selection of the data type from the drop-down list "???", the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

When you set up the IEC counter in a separate data block (single instance), the instance data block is created by default with "optimized block access" and the individual tags are defined as retentive. For additional information on setting retentivity in an instance data block, refer to "See also".

When you set up the IEC counter as local tag (multiple instance) in a function block with "optimized block access", it is defined as retentive in the block interface.

The operating system resets the instances of "Count up and down" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, you must call the instances to be initialized with the following parameter values in a restart OB:

- When used as an up counter, the value at the R parameter must be set to "1".
- When used as a down counter, the value at the LD parameter must be set to "1". In this case you specify the desired initial value for the CV parameter in the PV parameter.

If instances of the "Count up and down" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Syntax

The following syntax is used for the "Count up and down" instruction:

```
CALL CTUD, "IEC counters"
<Data type>
CU := <Operand>
CD := <Operand>
R := <Operand>
LD := <Operand>
PV := <Operand>
QU := <Operand>
QD := <Operand>
CV := <Operand>
```

Parameters

The following table shows the parameters of the "Count up and down" instruction:

Parameter	Declaration	Data type	Memory area	Description
CU	Input	BOOL	I, Q, M, D, L or constant	Count up input
CD	Input	BOOL	I, Q, M, D, L or constant	Count down input
R	Input	BOOL	I, Q, M, D, L, P or constant	Reset input
LD	Input	BOOL	I, Q, M, D, L, P or constant	Load input
PV	Input	Integers	I, Q, M, D, L, P or constant	Value at which the QU output is set.
QU	Output	BOOL	I, Q, M, D, L	Status of the up counter
QD	Output	BOOL	I, Q, M, D, L	Status of the down counter
CV	Output	Integers, CHAR, DATE	I, Q, M, D, L, P	Current counter value

You can select the data type of the instruction from the "<???>" drop-down list.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
CALL CTUD, "CTUD_DB"	// The "Count up and down" instruction is called. The "CTUD_DB" data block is assigned to the instruction.
INT	// Data type of the instruction
CU := "Tag_StartCTU"	// When the signal state at the "Tag_StartCTU" operand changes from "0" to "1", the instruction is executed and the current counter value of the "Tag_CounterValue" operand is incremented by one.
	// The counter value is incremented on a rising signal edge of the CU parameter until it reaches the high limit of INT = 32767.
CD := "Tag_StartCTD"	// When the signal state at the "Tag_StartCTD" operand changes from "0" to "1", the instruction is executed and the current counter value of the "Tag_CounterValue" operand is decremented by one.
	// The counter value of the CV parameter is decremented until the low limit of -32768 is reached.
R := "Tag_ResetCounter"	// When the signal state at the "Tag_ResetCounter" operand changes to "1", the "Tag_CounterValue" operand is reset to "0".

STL	Explanation
LD := "Tag_LoadPV"	// When the signal state at the "Tag_LoadPV" operand changes to "1", the "Tag_CounterValue" operand is set to the value of the "Tag_PresetValue" operand.
PV := "Tag_PresetValue"	// Specifies the value to which the counter is set when the signal state is "1" in the LD parameter.
QU := "Tag_CounterStatus"	// The operand is set as long as the current counter value is greater than or equal to the value of the PV parameter.
QD := "Tag_CounterStatus"	// The operand is set when the current counter value is less than or equal to zero.
CV := "Tag_CounterValue"	// Current counter value

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Setting retentivity in an instance data block (Page 2839)

Setting the retentivity of local tags (Page 2561)

Math functions

MIN: Get minimum

Description

The "Get minimum" instruction compares the values at the inputs IN1, IN2 and IN3 and writes the lowest value to the OUT output. The instruction is only executed if the tags of all inputs are of the same data type.

The value of the OUT parameter is invalid if one of the following conditions is met:

- The specified tags are not of the same data type.
- A floating-point number has an invalid value.

Syntax

The following syntax is used for the "Get minimum" instruction:

```
CALL MIN
```

```
<Data type>
IN1 := <Operand>
IN2 := <Operand>
IN3 := <Operand>
OUT := <Operand>
```

You can select the data type of the instruction from the "<Data type>" drop-down list.

Parameters

The following table shows the parameters of the "Get minimum" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	First input value
IN2	Input	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	Second input value
IN3	Input	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	Third input value
OUT	Output	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	Result
The data types TOD, LTOD, DATE, and LDT can only be used if the IEC test is not enabled.				

You can select the data type of the instruction from the "<???>" drop-down list.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL MIN                     // The "Get minimum" instruction is called.
INT                          // Data type of the instruction
IN1 := "TagIn_Value1"        // First input value to be compared.
IN2 := "TagIn_Value2"        // Second input value to be compared.
IN3 := "TagIn_Value3"        // Third input value to be compared.
OUT := "Tag_Minimum"         // Lowest value of the comparison
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
IN3	TagIn_Value3	13333
OUT	Tag_Minimum	12222

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

MAX: Get maximum

Description

The "Get maximum" instruction compares the values at the inputs IN1, IN2 and IN3 and writes the highest value to the OUT output. The instruction is only executed if the tags of all inputs are of the same data type.

The value of the OUT output is invalid if one of the following conditions is fulfilled:

- The specified tags are not of the same data type.
- A floating-point number has an invalid value.

Syntax

The following syntax is used for the "Get maximum" instruction:

```
CALL MAX
<Data type>
IN1 := <Operand>
IN2 := <Operand>
IN3 := <Operand>
OUT := <Operand>
```

You can select the data type of the instruction from the "<Data type>" drop-down list.

Parameters

The following table shows the parameters of the "Get maximum" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	First input value
IN2	Input	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	Second input value
IN3	Input	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	Third input value
OUT	Output	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	Result
The data types TOD, LTOD, DATE, and LDT can only be used if the IEC test is not enabled.				

You can select the data type of the instruction from the "<???>" drop-down list.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL MAX                     // The "Get maximum" instruction is called.
INT                          // Data type of the instruction
IN1 := "TagIn_Value1"        // First input value is compared.
IN2 := "TagIn_Value2"        // Second input value to be compared.
IN3 := "TagIn_Value3"        // Third input value to be compared.
OUT := "Tag_Maximum"         // Highest value of the comparison
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
IN3	TagIn_Value3	13333
OUT	Tag_Maximum	14444

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

LIMIT: Set limit value

Description

You can use the "Set limit value" instruction to limit the value at the IN input to the values at the inputs MN and MX. If the value at the IN input meets the MN condition <= IN <= MX, it is copied to the OUT output. If the condition is not fulfilled and the IN input value is below the low limit MN, the OUT output is set to the value of the MN input. If the high limit MX is exceeded, the OUT output is set to the value of the MX input.

The instruction is only executed if all the tags at the inputs are of the same data type.

The value of the OUT output is invalid if one of the following conditions is fulfilled:

- The specified tags are not of the same data type.
- An operand has an invalid value.
- The value at the MN parameter is greater than the value of the MX parameter.

Syntax

The following syntax is used for the "Set limit value" instruction:

```
CALL LIMIT
<Data type>
MN := <Operand>
IN := <Operand>
MX := <Operand>
OUT := <Operand>
```

You can select the data type of the instruction from the "<Data type>" drop-down list.

Parameters

The following table shows the parameters of the "Set limit value" instruction:

Parameter	Declaration	Data type	Memory area	Description
MN	Input	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	Low limit
IN	Input	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	Input value
MX	Input	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	High limit
OUT	Output	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	Result
The data types TOD, LTOD, DATE, and LDT can only be used if the IEC test is not enabled.				

You can select the data type of the instruction from the "<???>" drop-down list.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL LIMIT                   // The "Set limit value" instruction is called.
INT                          // Data type of the instruction
MN := "Tag_LowLimit"         // Low limit
IN := "Tag_InputValue"       // Input value
MX := "Tag_HighLimit"        // High limit
OUT := "Tag_Result"          // Result
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
MN	Tag_LowLimit	12000
IN	Tag_InputValue	8000
MX	Tag_HighLimit	16000
OUT	Tag_Result	12000

The "Tag_Result" operand is written with the value of the "Tag_LowLimit" operand because the value of the "Tag_InputValue" operand is outside the defined limit range.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Move operations

MOVE: Move value

Description

You can use the "Move value" instruction to transfer the content of the operand at the IN input to the operand at the OUT output. The operands at input IN and at output OUT must have the identical data type.

The following table shows the possible transfers:

Source (IN)	Destination (OUT)
BYTE	BYTE
WORD	WORD
DWORD	DWORD
LWORD	LWORD
SINT	SINT
USINT	USINT
INT	INT
UINT	UINT
DINT	DINT
UDINT	UDINT
LINT	LINT
ULINT	ULINT
REAL	REAL
LREAL	LREAL
S5TIME	S5TIME
TIME	TIME
LTIME	LTIME
DATE	DATE

Source (IN)	Destination (OUT)
DT	DT
LDT	LDT
TOD	TOD
LTOD	LTOD
DTL	DTL
CHAR	CHAR
WCHAR	WCHAR
Character of a string[1)]	Character of a string
ARRAY	ARRAY
STRUCT	STRUCT
PLC data type (UDT)	PLC data type (UDT)
IEC_TIMER	IEC_TIMER
IEC_LTIMER	IEC_LTIMER
IEC_SCOUNTER	IEC_SCOUNTER
IEC_USCOUNTER	IEC_USCOUNTER
IEC_COUNTER	IEC_COUNTER
IEC_UCOUNTER	IEC_UCOUNTER
IEC_DCOUNTER	IEC_DCOUNTER
IEC_UDCOUNTER	IEC_UDCOUNTER
IEC_LCOUNTER	IEC_LCOUNTER
IEC_ULCOUNTER	IEC_ULCOUNTER
DB_ANY	DB_ANY

[1)] You can also use the "Move value" instruction to transfer individual characters of a string to operands of the CHAR or WCHAR data type. The number of the character to be transferred is specified in square brackets next to the operand name. "MyString[2]", for example, transfers the second character of the "MyString" string. It is also possible to transfer from operands of the data type CHAR to the individual characters of a string. You can also replace a specific character of a string with the character of another string.

You can use the "Move block" (MOVE_BLK), "Move block uninterruptible" (UMOVE_BLK) and "Move block" (MOVE_BLK_VARIANT) instructions to move operands of the ARRAY data type. You can move operands of the STRING data type with the instruction "Move character string" (S_MOVE).

Syntax

The following syntax is used for the "Move value" instruction:

```
CALL MOVE
VARIANT
IN := <Operand>
OUT := <Operand>
```

Parameter

The following table shows the parameters of the "Move value" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	Bit strings, integers, floating-point numbers, timers, date and time, strings, ARRAY, IEC data types, PLC data type (UDT), DB_ANY	I, Q, M, D, L or constant	Element used to overwrite the destination address
OUT	Output	Bit strings, integers, floating-point numbers, timers, date and time, strings, ARRAY, IEC data types, PLC data type (UDT), DB_ANY	I, Q, M, D, L	Destination address

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL MOVE                    // The "Move value" instruction is called.
VARIANT                      // Data type of the instruction
IN := "TagIn_LREAL"          // The contents of the operand "TagIn_LREAL" are
                             moved.
OUT := "TagOut_LREAL"        // And transferred to the operand "TagOut_LREAL".
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	TagIn_LREAL	1.23456789098e55
OUT	TagOut_LREAL	1.23456789098e55

If operand "TagIn" has the signal state "1", the "Move value" instruction is executed. The instruction moves the contents of the "TagIn_LREAL" operand to the "TagOut_LREAL" operand.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Deserialize: Deserialization

Description

You can use the "Deserialize" instruction to convert the sequential representation of a PLC data type (UDT) back to a PLC data type and to fill its entire contents.

The memory area in which the sequential representation of a PLC data type is located must have the ARRAY of BYTE data type and be declared with standard access. Make sure that there is enough memory space prior to the conversion.

The instruction enables you to convert multiple sequential representations of converted PLC data types back to their original state step-by-step.

Note

If you only want to convert back a single sequential representation of a PLC data type (UDT), you can also use the instruction "TRCV: Receive data via communication connection" directly.

Syntax

The following syntax is used for the "Deserialize" instruction:

```
CALL Deserialize
SRC_ARRAY := <Operand>
RET_VAL := <Operand>
DEST_VARIABLE :=
<Operand>
POS := <Operand>
```

Parameter

The following table shows the parameters of the "Deserialize" instruction:

Parameter	Declaration	Data type	Memory area	Description
SRC_ARRAY	Input	VARIANT	I, Q, D, L	Global data block in which the generated data stream is stored
DEST_VARIABLE	InOut	VARIANT	I, Q, D, L	Tag in which the returned converted PLC data type (UDT) is stored

Parameter	Declaration	Data type	Memory area	Description
POS	InOut	DINT	I, Q, M, D, L	Number of bytes that the converted PLC data types use. The POS parameter is calculated zero-based.
RET_VAL	Output	INT	I, Q, M, D, L	Error information

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B0	The memory areas for the SRC_ARRAY and DEST_VARIABLE parameters overlap.
8136	The data block at the DEST_VARIABLE parameter is not a block with standard access.
8150	The VARIANT data type at the SRC_ARRAY parameter contains no value.
8151	Code generation error at SRC_ARRAY parameter
8153	There is not enough free memory available at the SRC_ARRAY parameter.
8250	The VARIANT data type at the DEST_VARIABLE parameter contains no value.
8251	Code generation error at DEST_VARIABLE parameter
8254	Invalid data type at the DEST_VARIABLE parameter
8382	The value at the POS parameter is outside the limits of the array.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

STL	Explanation
L 0	// The value "0" is loaded to accumulator 1.
T #BufferPos	// The value "0" is transferred to the "#BufferPos" operand.
CALL Deserialize	// The "Deserialize" instruction is called.
SRC_ARRAY := "Buffer".Field	// The customer data from the "Buffer" data block are deserialized.
RET_VAL := #Error	// Error information
DEST_VARIABLE := "Target".Client	// The deserialized customer data are written to the "Target" data block.

STL	Explanation
POS := #BufferPos	// The number of bytes used by the converted customer data is stored in the "#BufferPos" operand.
AN BR JC end	// If an error occurs during execution of the instruction, the processing of the program jumps to the end.
CALL Deserialize	// The "Deserialize" instruction is called.
SRC_ARRAY := "Buffer".Field	// The separator sheet from the "Buffer" data block is deserialized. The separator sheet was inserted in the form of 'arti' or 'Bill' characters to allow better differentiation of the customer data.
RET_VAL := #Error	// Error information
DEST_VARIABLE := #Label	// The deserialized characters of the separator sheet are written to the "#Label" operand.
POS := #BufferPos	// The number of bytes used by the converted characters is stored in the "#BufferPos" operand.
AN BR JC end	// If an error occurs during execution of the instruction, the processing of the program jumps to the end.
CALL S_COMP	// The "Compare character string" instruction is called.
src_type := STRING	// Data type of the instruction
relation := EQ	
IN1 := 'arti' IN2 := #Label	// Comparison of characters 'arti' with characters that were stored in the #Label operand.
OUT := #cmp	// Result of the instruction
A #cmp JCN noarticle	// If the comparison returns the signal state "0", the characters in the #Label operand are not 'arti' characters and the processing of the program jumps to the "noarticle" jump label.
CALL Deserialize	// If the result of the comparison was "1", the "Deserialize" instruction is called.
SRC_ARRAY := "Buffer".Field	// The article data are deserialized.
RET_VAL := #Error	// Error information
DEST_VARIABLE := "Target".Article[#DeliverPos]	// The deserialized article data are written to the "Target" data block.
POS := #BufferPos	// The number of bytes used by the converted article data is stored in the "#BufferPos" operand.
AN BR JC end	// If no error occurred during processing, you have reached the block end.

```
STL                                          Explanation
BE

noarticle: NOP 0                             // If the comparison for 'arti' = "0", processing
                                             of the program continues here.

CALL S_COMP                                  // The "Compare character string" instruction is
                                             called.
src_type := STRING                           // Data type of the instruction
relation := EQ
IN1 := 'Bill'                                // Comparison of characters 'Bill' with characters
IN2 := #Label                                that were stored in the #Label operand.

OUT := #cmp                                  // Result of the instruction

A #cmp                                       // If the comparison returns the signal state "1",
JCN end                                      the characters in the #Label operand are 'Bill'
                                             characters and processing of the program continues.

CALL Deserialize                             // If the result of the comparison was "1", the
                                             "Deserialize" instruction is called.
SRC_ARRAY := "Buffer".Field                  // The bill data are deserialized.
RET_VAL := #Error                            // Error information
DEST_VARIABLE := "Target".Bill[#DeliverPos]  // The deserialized customer data are written to
                                             the "Target" data block.
POS := #BufferPos                            // The number of bytes used by the converted
                                             article data is stored in the "#BufferPos" operand.

AN BR                                        // If no error occurred during processing, you
JC end                                       have reached the block end.
BE

end: CLR                                     // A jump to jump label "end" takes place in the
                                             case of an error.
SAVE                                         // The RLO is saved in the BR bit.
```

The following table shows the declaration of the operands:

Operand	Data type	Declaration
DeliverPos	INT	In the "Input" section of the block interface
BufferPos	DINT	In the "Temp" section of the block interface
Error	INT	In the "Temp" section of the block interface
Label	STRING[4]	In the "Temp" section of the block interface

The following table lists the declarations of the PLC data types:

Name of the PLC data types	Name	Data type
Article	Number	DINT
	Declaration	STRING
	Colli	INT
Client	Title	INT
	First name	STRING[10]
	Surname	STRING[10]

The following table shows the declaration of the data blocks:

Name of the data blocks	Name	Data type
Target	Client	"Client" (PLC data type)
	Article	Array[0..] of "Article" (PLC data type)
	Bill	Array[0..10] of INT
Buffer	Field	Array[0..294] of BYTE

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Switching display formats in the program status (Page 2980)

PLC data types (Page 2280)

Serialize: Serialization

Description

You can use the "Serialize" instruction to convert several PLC data types (UDT) to a sequential representation without losing parts of their structure.

You can use the instruction to temporarily save multiple structured data items from your program in a buffer, which should preferably be in a global data block, and send them to another CPU. The memory area in which the converted PLC data types are stored must have the ARRAY of BYTE data type and be declared with standard access. Make sure that there is enough memory space prior to the conversion.

The operand at the POS parameter contains information about the number of bytes used by the converted PLC data types.

Note

If you want to send a single PLC data type (UDT), you can directly call the instruction "TSEND: Send data via communication connection".

Syntax

The following syntax is used for the "Serialize" instruction:

```
CALL Serialize
SRC_VARIABLE := <Operand>
RET_VAL := <Operand>
DEST_ARRAY := <Operand>
POS := <Operand>
```

Parameter

The following table shows the parameters of the "Serialize" instruction:

Parameter	Declaration	Data type	Memory area	Description
SRC_VARIABLE	Input	VARIANT	I, Q, D, L	PLC data type (UDT) that is converted to sequential representation.
DEST_ARRAY	InOut	VARIANT	I, Q, D, L	Data block in which the generated data stream is stored.
POS	InOut	DINT	I, Q, M, D, L	Number of bytes that the converted PLC data types use. The POS parameter is calculated zero-based.
RET_VAL	Output	INT	I, Q, M, D, L	Error information

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B0	The memory areas for the SRC_VARIABLE and DEST_ARRAY parameters overlap.
8150	The VARIANT data type at the SRC_VARIABLE parameter contains no value.
8152	Code generation error at SRC_VARIABLE parameter

Error code* (W#16#...)	Explanation
8236	The data block at the DEST_ARRAY parameter is not a block with standard access.
8250	The VARIANT data type at the DEST_ARRAY parameter contains no value.
8252	Code generation error at DEST_ARRAY parameter
8253	There is not enough free memory available at the DEST_ARRAY parameter.
8254	Invalid data type at the DEST_ARRAY parameter
8382	The value at the POS parameter is outside the limits of the array.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

```
STL                                  Explanation
L 0                                  // The value "0" is loaded to accumulator 1.
T #BufferPos                         // The value "0" is transferred to the
                                     "#BufferPos" operand.

CALL Serialize                       // The "Serialize" instruction is called.
SRC_VARIABLE := "Source".Client      // The customer data from the "Source".Client
                                     data block are serialized.
RET_VAL := #Error                    // Error information
DEST_ARRAY := "Buffer".Field         // The sequential representation is written
                                     to the "Buffer".Field data block.
POS := #BufferPos                    // The number of bytes occupied by the
                                     sequential representation is stored in the
                                     "#BufferPos" operand.

CALL S_MOVE                          // The "Move character string" instruction is
                                     called.
src_type := STRING                   // Data types of the instruction
dest_type := STRING
IN := 'arti'                         // The 'arti' characters are copied to the
OUT := #Label                        "#Label" operand.

CALL Serialize                       // The "Serialize" instruction is called.
SRC_VARIABLE := #Label               / / The characters are serialized.
RET_VAL := #Error                    // Error information
DEST_ARRAY := "Buffer".Field         // The sequential representation is written
                                     to the "Buffer".Field data block after the
                                     customer data.
```

```
STL                                     Explanation
POS := #BufferPos                       // The number of bytes that the characters
                                        need is added to the number already stored in
                                        the "#BufferPos" operand.

CALL Serialize                          // The "Serialize" instruction is called.
SRC_VARIABLE :=                         // The data of a specific article from the
"Source".Article[#DeliverPos]           "Source".Article[#DeliverPos] data block,
                                        which are calculated in runtime, are
                                        serialized.
RET_VAL := #Error                       // Error information
DEST_ARRAY := "Buffer".Field            // The sequential representation is written
                                        to the "Buffer".Field data block after the
                                        'arti' characters.
POS := #BufferPos                       // Number of bytes
```

The following table shows the declaration of the operands:

Operand	Data type	Declaration
DeliverPos	INT	In the "Input" section of the block interface
BufferPos	DINT	In the "Temp" section of the block interface
Error	INT	In the "Temp" section of the block interface
Label	STRING[4]	In the "Temp" section of the block interface

The following table lists the declarations of the PLC data types:

Name of the PLC data types	Name	Data type
Article	Number	DINT
	Declaration	STRING
	Colli	INT
Client	Title	INT
	First name	STRING[10]
	Surname	STRING[10]

The following table shows the declaration of the data blocks:

Name of the data blocks	Name	Data type
Source	Client	"Client" (PLC data type)
	Article	Array[0..10] of "Article" (PLC data type)
Buffer	Field	Array[0..294] of BYTE

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Switching display formats in the program status (Page 2980)

PLC data types (Page 2280)

MOVE_BLK: Move block

Description

You can use the "Move block" instruction to move the content of a memory area (source area) to another memory area (destination area). The number of elements to be copied to the destination area is specified with the COUNT parameter. The width of the elements to be moved is defined by the width of the element at the IN parameter.

The instruction can only be executed if the source area and the destination area have the same data type.

The value of the OUT output is invalid if the following condition is met:

- More data is moved than is made available at the IN parameter or OUT parameter.

Syntax

The following syntax is used for the "Move block" instruction:

```
CALL MOVE_BLK
<Data type> <Data type>
IN := <Operand>
COUNT := <Operand>
OUT := <Operand>
```

Parameters

The following table shows the parameters of the "Move block" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN [1]	Input	Binary numbers, integers, floating-point numbers, timers, TOD, LTOD, DATE, CHAR	D, L	The first element of the source area that is being copied
COUNT	Input	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P or constant	Number of elements to be copied from the source area to the destination area
OUT [1]	Output	Binary numbers, integers, floating-point numbers, timers, TOD, LTOD, DATE, CHAR	D, L	The first element of the destination area to which the contents of the source area are being copied
[1] The specified data types can only be used as elements of an Array structure.				

You can select the data type of the instruction from the "<???>" drop-down list.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL MOVE_BLK                // The "Move block" instruction is called.
value_type := INT            // Data types of the instruction
count_type := UINT           // Data type of the COUNT parameter
IN := #a_array[2]            // First element to be moved.
COUNT := "Tag_Count"         // Number of elements to be moved from the source area
                             to the destination area.
OUT := #b_array[1]           // First element to which is moved.
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	a_array[2]	The data type of the a_array operand is Array [0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the b_array operand is Array [0..6] of INT. It consists of seven elements of data type INT.

Starting from the third element, the instruction selects three INT elements from the #a_array tag and copies their contents to the #b_array output tag, beginning with the second element.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

MOVE_BLK_VARIANT: Move block

Description

The instruction "Move block" copies the contents of a memory area (source area) to another memory area (destination area). You can copy elements of an array to another array of the same data type. The size (number of elements) of the source and destination array may be different. You can copy several elements within an array or individual elements.

If you are using the instruction, the array must not yet be known at the time the block is created, as the source and the destination are transferred using VARIANT.

The counting at the parameters SRC_INDEX and DEST_INDEX always starts with the low limit "0", regardless of the later declaration of the array.

The instruction is not executed if more data is copied than is made available.

If the VARIANT pointer (source or destination) is of the data type BOOL, it must be addressed absolutely and the length specified must be divisible by 8; otherwise the instruction is not executed.

Syntax

The following syntax is used for the "Move block" instruction:

```
CALL MOVE_BLK_VARIANT
SRC := <Operand>
COUNT := <Operand>
SRC_INDEX := <Operand>
DEST_INDEX := <Operand>
DEST := <Operand>
RET_VAL := <Operand>
```

Parameter

The following table shows the parameters of the "Move block" instruction:

Parameter	Declaration	Data type	Memory area	Description
SRC	Input	VARIANT (array or single element)	L	Source block from which to copy
COUNT	Input	UDINT	I, Q, M, D, L or constant	Number of elements which are copied Set the value at the parameter COUNT to "1", if no Array is specified at the parameter SRC or the parameter DEST.

Parameter	Declaration	Data type	Memory area	Description
SRC_INDEX	Input	DINT	I, Q, M, D, L or constant	• The SRC_INDEX parameter is calculated zero-based. If an Array is specified at parameter SRC, the integer at the SRC_INDEX parameter specifies the first element within the source area from which copying is to take place. This is independent of the declared array limits. • If no Array is set at the SRC parameter or only an individual element of an array is specified, assign the SRC_INDEX parameter the value "0".
DEST_INDEX	Input	DINT	I, Q, M, D, L or constant	• The DEST_INDEX parameter is calculated zero-based. If an Array is specified at the DEST parameter, the integer at the DEST_INDEX parameter specifies the first element within the destination area to which copying is to take place. This is independent of the declared array limits. • If no Array is specified at the DEST parameter, assign the DEST_INDEX parameter the value "0".

Parameter	Declaration	Data type	Memory area	Description
DEST	Output	VARIANT	L	Destination area into which the contents of the source block are copied.
RET_VAL	Output	INT	I, Q, M, D, L	Error information: If an error occurs during the execution of the instruction, an error code is output on the RET_VAL parameter.

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	Data types do not correspond
8151	Access to the SRC parameter is not possible.
8152	The operand at the SRC parameter is not typed.
8153	Code generation error at SRC parameter
8154	The operand at the SRC parameter has the data type BOOL.
8281	The COUNT parameter has an invalid value
8382	The value at the SRC_INDEX parameter is outside the limits of the VARIANT.
8383	The value of the SRC_INDEX parameter is outside the high limit of the array.
8482	The value at the DEST_INDEX parameter is outside the limits of the VARIANT.
8483	The value of the DEST_INDEX parameter is outside the high limit of the array.
8534	The DEST parameter is write protected
8551	Access to the DEST parameter is not possible.
8552	The operand at the DEST parameter is not typed.
8553	Code generation error at DEST parameter
8554	The operand at the DEST parameter has the data type BOOL.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

```
STL                                    Explanation
CALL MOVE_BLK_VARIANT                  // The "Move block" instruction is called.
SRC := #SrcField                       // First element to moved
COUNT := "Tag_Count"                   // Number of elements to be moved from the
                                       source area to the destination area.
SRC_INDEX := "Tag_Src_Index"           // First element at the SRC parameter to be
                                       moved
DEST_INDEX := "Tag_Dest_Index"         // First element at the DEST parameter to be
                                       moved
DEST := #DestField                     // The first element of the source area to be
                                       moved.
RET_VAL := "Tag_Result"                // Error information
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
SRC	#SrcField	The local operand #SrcField uses a UDT that was still unknown at the time when the block was programmed. (Array [0 to 10] of "MOVE_UDT"
COUNT	Tag_Count	2
SRC_INDEX	Tag_Src_Index	3
DEST_INDEX	Tag_Dest_Index	3
DEST	#DestField	The local operand #DestField uses a UDT that was still unknown at the time when the block was programmed. (Array [10 to 20] of "MOVE_UDT"

2 elements are copied from the source area into the destination area beginning with the fourth element of the array of UDT. The copies are inserted in the array of UDT beginning with the fourth element.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Switching display formats in the program status (Page 2980)

UMOVE_BLK: Move block uninterruptible

Description

You can use the "Move block uninterruptible" instruction to move the content of a memory area (source area) to another memory area (destination area). The instruction cannot be interrupted. The number of elements to be copied to the destination area is specified with the COUNT parameter. The width of the elements to be copied is defined by the width of the element at the IN input.

The instruction can only be executed if the source area and the destination area have the same data type.

Note

The copy operation cannot be interrupted by other operating system activities. This is why the interrupt reaction times of the CPU increase during the execution of the "Move block uninterruptible" instruction.

The value of the OUT output is invalid if the following condition is met:

- More data is moved than is made available at the IN parameter or OUT parameter.

You can use the "Move block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Syntax

The following syntax is used for the "Move block uninterruptible" instruction:

```
CALL UMOVE_BLK
<Data type> <Data type>
IN := <Operand>
COUNT := <Operand>
OUT := <Operand>
```

Parameters

The following table shows the parameters of the "Move block uninterruptible" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN [1]	Input	Binary numbers, integers, floating-point numbers, timers, TOD, LTOD, DATE, CHAR	D, L	The first element of the source area that is being copied
COUNT	Input	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P or constant	Number of elements to be copied from the source area to the destination area
OUT [1]	Output	Binary numbers, integers, floating-point numbers, timers, TOD, LTOD, DATE, CHAR	D, L	The first element of the destination area to which the contents of the source area are being copied
[1] The specified data types can only be used as elements of an Array structure.				

You can select the data type of the instruction from the "<???>" drop-down list.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL UMOVE_BLK               // The "Move block uninterruptible" instruction is
                             called.
value_type := INT            // Data types of the instruction
count_type := UINT           // Data type of the COUNT parameter
IN := #a_array[2]            // First element from where the move begins.
COUNT := "Tag_Count"         // Number of elements to be moved from the source area
                             to the destination area.
OUT := #b_array[1]           // First element to which is moved.
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	a_array[2]	The data type of the a_array operand is Array [0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the b_array operand is Array [0..6] of INT. It consists of seven elements of data type INT.

Starting from the third element, the instruction selects three INT elements from the #a_array tag and copies their contents to the #b_array output tag, beginning with the second element. The copy operation cannot be interrupted by other operating system activities.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

FILL_BLK: Fill block

Description

You can use the "Fill block" instruction to fill a memory area (destination area) with the value of the IN input. The destination area is filled starting from the address specified at the OUT output. The number of repeated copy operations is specified with the COUNT parameter. When the instruction is executed, the value at the IN input is selected and copied to the destination area as often as specified by the value in the COUNT parameter.

The instruction can only be executed if the source area and the destination area have the same data type.

The value of the OUT output is invalid if the following condition is met:

- More data is moved than is made available at input IN or output OUT.

Syntax

The following syntax is used for the "Fill block" instruction:

```
CALL FILL_BLK
<Data type> <Data type>
```

```
IN := <Operand>
COUNT := <Operand>
OUT := <Operand>
```

Parameters

The following table shows the parameters of the "Fill block" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN [1]	Input	Binary numbers, integers, floating-point numbers, timers, TOD, LTOD, DATE, CHAR	I, Q, M, D, L, P or constant	Element used to fill the destination area
COUNT	Input	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P or constant	Number of repeated copy operations
OUT [2]	Output	Binary numbers, integers, floating-point numbers, timers, TOD, LTOD, DATE, CHAR	D, L	Address in destination area from which filling starts
[1] The specified data types can also be used as elements of an Array structure. [2] The specified data types can only be used as elements of an Array structure.				

You can select the data type of the instruction from the "<???>" drop-down list.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL FILL_BLK                // The "Fill block" instruction is called.
value_type := INT            // Data types of the instruction
count_type := UINT           // Data type of the COUNT parameter
IN := a_array[2]             // Element to be moved.
COUNT := "Tag_Count"         // Number of repeated copy operations
OUT := b_array[1]            // The destination area is filled starting from the
                             address specified at the OUT output.
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	a_array[2]	The data type of the a_array operand is Array [0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the b_array operand is Array [0..6] of INT. It consists of seven elements of data type INT.

The instruction copies the third element (#a_array[2]) of the #a_array tag three times to the #b_array output tag.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

UFILL_BLK: Fill block uninterruptible

Description

You can use the "Fill block uninterruptible" instruction to fill a memory area (destination area) with the value of the IN input. The instruction cannot be interrupted. The destination area is filled starting from the address specified at the OUT output. The number of repeated copy operations is specified with the COUNT parameter. When the instruction is executed, the value at the IN input is selected and copied to the destination area as often as specified by the value in the COUNT parameter.

The instruction can only be executed if the source area and the destination area have the same data type.

Note

The copy operation cannot be interrupted by other operating system activities. This is why the interrupt reaction times of the CPU increase during the execution of the "Fill block uninterruptible" instruction.

The value of the OUT output is invalid if the following condition is met:

- More data is moved than is made available at input IN or output OUT.

You can use the "Fill block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Syntax

The following syntax is used for the "Fill block uninterruptible" instruction:

```
CALL UFILL_BLK
<Data type> <Data type>
IN := <Operand>
COUNT := <Operand>
OUT := <Operand>
```

Parameters

The following table shows the parameters of the "Fill block uninterruptible" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN [1]	Input	Binary numbers, integers, floating-point numbers, timers, TOD, LTOD, DATE, CHAR	I, Q, M, D, L, P or constant	Element used to fill the destination area
COUNT	Input	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P or constant	Number of repeated copy operations
OUT [2]	Output	Binary numbers, integers, floating-point numbers, timers, TOD, LTOD, DATE, CHAR	D, L	Address in destination area from which filling starts
[1] The specified data types can also be used as elements of an ARRAY structure. [2] The specified data types can only be used as elements of an ARRAY structure.				

You can select the data type of the instruction from the "<???>" drop-down list.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
CALL UFILL_BLK	// The "Fill block uninterruptible" instruction is called.

```
STL                          Explanation
value_type := INT            // Data types of the instruction
count_type := UINT           // Data type of the COUNT parameter
IN := "Tag_IN"               // Element to be moved
COUNT := "Tag_Count"         // Number of repeated copy operations
OUT := "Tag_OUT"             // The destination area is filled starting from the
                             address specified at the OUT output.
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	a_array[2]	The data type of the a_array operand is Array [0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the b_array operand is Array [0..6] of INT. It consists of seven elements of data type INT.

The instruction copies the third element (#a_array[2]) of the #a_array tag three times to the #b_array output tag. The copy operation cannot be interrupted by other operating system activities.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

ARRAY DB instructions

ReadFromArrayDB: Read from array data block

Description

You can use the "Read from array data block" instruction to read data from an array DB and write them to a destination area.

An array DB is a data block which consists of exactly one array of [data type]. The elements of the array can be UDT data type or any other elementary data type. The counting of the array always starts with the low limit "0", regardless of the later declaration of the array.

Syntax

The following syntax is used for the "Read from the ARRAY data block" instruction:

```
CALL ReadFromArrayDB
DB := <Operand>
INDEX := <Operand>
RET_VAL := <Operand>
VALUE := <Operand>
```

Parameter

The following table shows the parameters of the "Read from array data block" instruction:

Parameter	Declaration	Data type	Memory area	Description
DB	Input	DB_ANY	I, Q, M, D, L	Data block that is read
INDEX	Input	DINT	I, Q, M, D, L, P or constant	Element that is read
VALUE	Output	VARIANT	L	Value that is read and output
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output on the RET_VAL parameter.

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	Data types do not correspond
80B5	Copying process was interrupted
8132	The data block does not exist or is too short

Error code* (W#16#...)	Explanation
8133	The data block is located in the load memory
8134	The data block is write protected
8135	The array DB contains invalid values
8154	The data type is not correctly declared
8282	The value of the INDEX parameter is outside the limits of the array.
8350	The VARIANT data type contains no value
8352	Code generation error
8353	Code generation error
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

```
STL                                          Explanation
CALL ReadFromArrayDB                         // Calling the instruction
DB := "ArrayDB"                              // Data block from which data is read.
INDEX := "ArrayDB".THIS[2]                   // The element [2] is read from the data
                                             block.
RET_VAL := "TagRet_Val"                      // Error information
VALUE := "TargetField[10]".Data2[1]          // The read value is provided at the
                                             parameter VALUE.
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
DB	ArrayDB	The data type of the ArrayDB operand is Array [0 to 10] of DINT.
INDEX	ArrayDB.THIS[2]	Third element of the "ArrayDB"
VALUE	TargetField[10].Data2[1]	The data type of the TargetField operand is Array [10 to 20] of "MOVE_UDT".

The third element is read from "ArrayDB" and written to the "TargetField[10].Data2[1]" operand.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)
Basic information on the status word (Page 2466)
Querying and setting status bits in STL (Page 2467)

WriteToArrayDB: Write to array data block

Description

You can use the "Write to the array data block" instruction to write data to an array DB.

An array DB is a data block which consists of exactly one array of [data type]. The elements of the array can be UDT data type or any other elementary data type. The counting of the array always starts with the low limit "0", regardless of the later declaration of the array.

Syntax

The following syntax is used for the "Write to the ARRAY data block" instruction:

```
CALL WriteToArrayDB
DB := <Operand>
INDEX := <Operand>
RET_VAL := <Operand>
VALUE := <Operand>
```

Parameter

The following table shows the parameters of the "Write to the array data block" instruction:

Parameter	Declaration	Data type	Memory area	Description
DB	Input	DB_ANY	I, Q, M, D, L	Data block to which data is written
INDEX	Input	DINT	I, Q, M, D, L, P or constant	Element in the DB to which data is written
VALUE	Input	VARIANT	L	Value to be written
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output on the RET_VAL parameter.

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	Data types do not correspond
80B5	Copying process was interrupted
8132	The data block does not exist or is too short
8133	The data block is located in the load memory
8134	The data block is write protected
8135	The DB is not an array data block.
8282	The value of the INDEX parameter is outside the limits of the array.
8350	The VARIANT data type contains no value
8352	Code generation error
8353	Code generation error
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

```
STL                                        Explanation
CALL WriteToArrayDB                        // Calling the instruction
DB := "ArrayDB"                            // Data block to which data is written
INDEX := "ArrayDB".THIS[2]                 // The value is written to the element
                                           [2] of the data block.
RET_VAL := "TagRet_Val"                    // Error information
VALUE := "SourceField[1]".Data1[6]         // The value that is written into the
                                           data block.
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
DB	ArrayDB	The data type of the ArrayDB operand is Array [0 to 10] of DINT.
INDEX	ArrayDB.THIS[2]	Third element of the "ArrayDB"
VALUE	SourceField[1].Data1[6]	The data type of the SourceField operand is Array [0 to 10] of "MOVE_UDT".

From the "SourceField" operand, the "Data1[6]" element from the second element is written to the "ArrayDB". The third element is written to the "ArrayDB".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

ReadFromArrayDBL: Read from array data block in load memory

Description

You can use the "Read from array data block in load memory" instruction to read data from an array DB in load memory.

An array DB is a data block which consists of exactly one array of [data type]. The elements of the array can be UDT data type or any other elementary data type. The counting of the array always starts with the low limit "0", regardless of the later declaration of the array.

If the array DB is marked with the "Only store in load memory" block attribute, it is only stored in load memory.

The instruction is executed when a positive signal edge is detected at the REQ parameter. The BUSY parameter then has the signal state "1". The instruction is terminated if a negative signal edge is detected at the BUSY parameter. The DONE parameter has the signal state "1" for one program cycle and the read value is output at the VALUE parameter within this cycle. With all other program cycles, the value at the VALUE parameter is not changed.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Note

The array DB must be located in a data block with the block property "Optimized".

Syntax

The following syntax is used for the "Read from the ARRAY data block in the load memory" instruction:

```
CALL ReadFromArrayDBL,
"Instance DB"
REQ := <Operand>
```

```
DB := <Operand>
INDEX := <Operand>
BUSY := <Operand>
DONE := <Operand>
ERROR := <Operand>
VALUE := <Operand>
```

Parameter

The following table shows the parameters of the "Read from array data block in load memory" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = "1": Starting with the reading of the array DB
DB	Input	DB_ANY	I, Q, M, D, L	Data block that is read
INDEX	Input	DINT	I, Q, M, D, L, P or constant	Element that is read
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = "1": The array DB is still being read
DONE	Output	BOOL	I, Q, M, D, L	DONE = "1": The instruction was executed successfully
VALUE	InOut	VARIANT	D, L	Value that is read and output
ERROR	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output on the ERROR parameter.

For additional information on valid data types, refer to "See also".

ERROR parameter

The following table shows the meaning of the values of the ERROR parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	Data types do not correspond
80B5	Copying process was interrupted
80C3	Too many concurrent calls
8132	The data block does not exist or is too short
8133	The data block is not located in the load memory
8134	The data block is write protected
8135	The DB is not an array data block.
8282	The value of the INDEX parameter is outside the limits of the array.
8350	The VARIANT data type contains no value
8352	Code generation error
8353	Code generation error
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

You can find descriptions of the error codes triggered by the instructions "READ_DBL: Read from data block in the load memory" and "WRIT_DBL: Write to data block in the load memory" under "See also".

Example

The following example shows how the instruction works:

STL	Explanation
`CALL ReadFromArrayDBL, "ReadFromArrayDBL_DB"`	`// Calling the instruction`
`REQ := "TagReg"`	`// The instruction executes on a positive signal edge.`
`DB := "ArrayDB"`	`// Data block from which data is read.`
`INDEX := "ArrayDB".THIS[2]`	`// The element [2] is read from the data block.`
`BUSY := "TagBusy"`	`// Indicates whether the instruction is still in process.`
`DONE := "TagDone"`	`// Indicates whether the instruction is closed.`
`ERROR := "TagError"`	`// Error information`
`VALUE := "SourceTargetField[1]".Data1[6]`	`// The read value is provided at the parameter VALUE.`

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
REQ	TagReq	BOOL
DB	ArrayDB	The data type of the ArrayDB operand is Array [0 to 10] of DINT.
INDEX	ArrayDB.THIS[2]	Third element of the "ArrayDB"
VALUE	SourceTargetField[1].Data1[6]	The data type of the SourceTargetField operand is Array [0 to 10] of "MOVE_UDT".
BUSY	TagBusy	BOOL
DONE	TagDone	BOOL

If a positive edge is detected at the "TagReq" operand, the "Read from the ARRAY data block in the load memory" instruction is executed. The third element is read from "ArrayDB" and written to the "SourceTargetField[1].Data1[6]" operand. As soon as a negative signal edge is detected at the "TagBusy" operand, the instruction is terminated and the value at the VALUE parameter is no longer changed. After the instruction has been processed, the "TagDone" operand the signal state "1".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

READ_DBL: Read from data block in the load memory (Page 4801)

WRIT_DBL: Write to data block in the load memory (Page 4803)

WriteToArrayDBL: Write to array data block in load memory

Description

You can use the "Write to the array data block in load memory" instruction to write data to an array DB in load memory.

An array DB is a data block which consists of exactly one array of [data type]. The elements of the array can be UDT data type or any other elementary data type. The counting of the array always starts with the low limit "0", regardless of the later declaration of the array.

If the array DB is marked with the "Only store in load memory" block attribute, it is only stored in load memory.

The instruction is executed when a positive signal edge is detected at the REQ parameter. The BUSY parameter then has the signal state "1". If a negative signal edge is detected at the

BUSY parameter, the instruction is terminated and the value at the VALUE parameter is written to the data block. The DONE parameter then has the signal state "1" for one program cycle.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Note

The array DB must be located in a data block with the block property "Optimized".

Syntax

The following syntax is used for the "Write to ARRAY data block in the load memory" instruction:

```
CALL WriteToArrayDBL,
"Instance DB"
REQ := <Operand>
DB := <Operand>
INDEX := <Operand>
VALUE := <Operand>
BUSY := <Operand>
DONE := <Operand>
ERROR := <Operand>
```

Parameters

The following table shows the parameters of the "Write to the array data block in load memory" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = "1": Start writing to the array DB
DB	Input	DB_ANY	D	Data block to which data is written
INDEX	Input	DINT	I, Q, M, D, L, P or constant	Element in the DB to which data is written
VALUE	Input	VARIANT	D, L	Value to be written
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = "1": The array DB is still being written

Parameter	Declaration	Data type	Memory area	Description
DONE	Output	BOOL	I, Q, M, D, L	DONE = "1": The instruction was executed successfully
ERROR	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output on the ERROR parameter.

For additional information on valid data types, refer to "See also".

ERROR parameter

The following table shows the meaning of the values of the ERROR parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	Data types do not correspond
80B5	Copying process was interrupted
80C3	Too many concurrent calls
8132	The data block does not exist or is too short
8133	The data block is not located in the load memory
8134	The data block is write protected
8135	The DB is not an array data block.
8282	The value of the INDEX parameter is outside the limits of the array.
8350	The VARIANT data type contains no value
8352	Code generation error
8353	Code generation error
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

You can find descriptions of the error codes triggered by the instructions "READ_DBL: Read from data block in the load memory" and "WRIT_DBL: Write to data block in the load memory" under "See also".

Example

The following example shows how the instruction works:

STL	Explanation
`CALL WriteToArrayDBL, "WriteToArrayDBL_DB"`	`// Calling the instruction`
`REQ := "TagReg"`	`// The instruction executes on a positive signal edge.`
`DB := "ArrayDB"`	`// Data block from which data is read.`
`INDEX := "ArrayDB".THIS[2]`	`// The element [2] is read from the data block.`
`BUSY := "TagBusy"`	`// Indicates whether the instruction is still in process.`
`DONE := "TagDone"`	`// Indicates whether the instruction is closed.`
`ERROR := "TagError"`	`// Error information`
`VALUE := "SourceTargetField[1]".Data1[6]`	`// The read value is provided at the parameter VALUE.`

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
REQ	TagReq	BOOL
DB	ArrayDB	The data type of the ArrayDB operand is Array [0 to 10] of DINT.
INDEX	ArrayDB.THIS[2]	Third element of the "ArrayDB"
VALUE	SourceTargetField[1].Data1[6]	The data type of the SourceTargetField operand is Array [0 to 10] of "MOVE_UDT".
BUSY	TagBusy	BOOL
DONE	TagDone	BOOL

If a positive edge is detected at the "TagReq" operand, the "Write to ARRAY data block in the load memory" instruction is executed. As soon as a negative signal edge is detected at the "TagBusy" operand, the instruction is terminated and the value at the VALUE parameter is written in the third element in the "ArrayDB". After the instruction has been processed, the "TagDone" operand the signal state "1".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

READ_DBL: Read from data block in the load memory (Page 4801)

WRIT_DBL: Write to data block in the load memory (Page 4803)

Other

BLKMOV: Move block

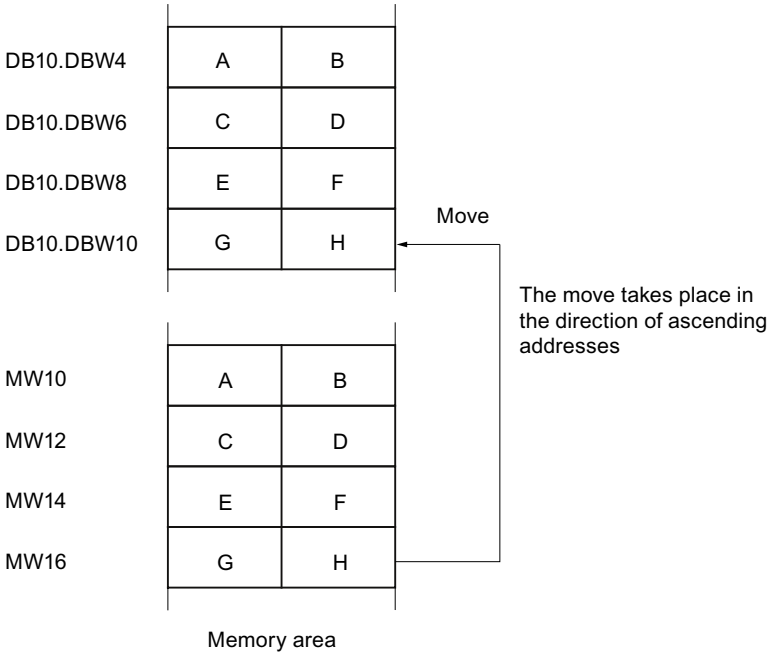
Description

The instruction "Move block" copies the contents of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use VARIANT to define the source and destination areas.

Note

The tags of the instruction can only be used in data blocks in which the attribute "Optimized block access" is not set. If the tag with the retentive setting "Set in IDB" has been declared, you can also use the tags "with optimized access".

The following figure shows the principle of the move operation:



Consistency of source data and destination data

Make sure that the source data remain unchanged during execution of the "Move block" instruction. Otherwise the consistency of the destination data cannot be guaranteed.

Interruptibility

There is no limit to the nesting depth.

Memory areas

The instruction "Move block" moves the following memory areas:

- Areas of a data block
- Bit memory
- Process image of the inputs
- Process image of the outputs

General rules for moving

The source and destination areas must not overlap. If the source and destination area have different lengths, only the length of the smaller area will be moved.

If the source area is smaller than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is smaller than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a block of data type BOOL is moved, the tag must be addressed absolutely and the specified length of the area must be divisible by 8, otherwise the instruction cannot be executed.

Rules for moving character strings

You can use the "Move block" instruction to move source and destination areas of the STRING data type. If only the source area is of data type STRING, the characters will be moved that are actually contained in the character string. Information on the actual and maximum length is also written to the destination area. If the source and destination area are both STRING data type, the current length of the character string in the destination area is set to the number of characters actually moved.

If you want to move the information on the maximum and actual length of a character string, specify the areas in bytes in the SRCBLK and DSTBLK parameters.

Syntax

The following syntax is used for the "Move block" instruction:

```
CALL BLKMOV
VARIANT
SRCBLK := <Operand>
RET_VAL := <Operand>
DSTBLK := <Operand>
```

Parameters

The following table shows the parameters of the "Move block" instruction:

Parameter	Declaration	Data type	Memory area	Description
SRCBLK	Input	VARIANT	L, P	Specifies the memory area to be moved (source area).
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output on the RET_VAL parameter.
DSTBLK	Output	VARIANT	L, P	Specifies the memory area to which the block is to be moved (destination area).

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8152	The data types WSTRING, WCHAR and BOOL are not supported.
8092	The source or target area is only available in the load memory.
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

```
STL                              Explanation
CALL BLKMOV                      // The "Move block" instruction is called.
VARIANT                          // Data type of the instruction
SRCBLK := P#M100.0 BYTE 10       // 10 Bytes starting from MB100 are moved.
RET_VAL := "Tag_ErrorCode"       // If an error occurs during the move operation, its
                                 error code is output in the variable "Tag_ErrorCode".
DSTBLK := P#DB1.DBX0.0 BYTE      // The moved bytes are transferred to DB1.
10
```

See also

Overview of the valid data types (Page 2236)
Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)
Evaluating errors with output parameter RET_VAL (Page 5370)
Call function blocks as single or multi-instances (Page 2182)
Basic information on the status word (Page 2466)
Querying and setting status bits in STL (Page 2467)
Switching display formats in the program status (Page 2980)
GET_ERR_ID: Get error ID locally (Page 3852)

UBLKMOV: Move block uninterruptible

Description

The instruction "Move block uninterruptible" copies the contents of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use VARIANT to define the source and destination areas.

The move operation cannot be interrupted by other operating system activities. As a result the interrupt reaction time of the CPU can increase during the execution of the instruction "Move block uninterruptible".

Note

The tags of the instruction can only be used in data blocks in which the attribute "Optimized block access" is not set. If the tag with the retentive setting "Set in IDB" has been declared, you can also use the tags "with optimized access".

Memory areas

The instruction "Move block uninterruptible" copies the following memory areas:

- Areas of a data block
- Bit memory
- Process image of the inputs
- Process image of the outputs

General rules for moving

The source and destination area must not overlap during the execution of the instruction "Move block uninterruptible". If the source area is smaller than the destination area, the entire source

area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is smaller than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a source or destination area defined as a formal parameter is less than a destination or source area specified at the SRCBLK or DSTBLK parameter, no data is transferred.

If a block of data type BOOL is moved, the tag must be addressed absolutely and the specified length of the area must be divisible by 8, otherwise the instruction cannot be executed.

You can use the "Move block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Rules for moving character strings

The instruction "Move block uninterruptible" copies source and destination areas of the data type STRING. If only the source area is of data type STRING, the characters will be moved that are actually contained in the character string. Information on the actual and maximum length are not written in the destination area. If the source and destination area are both STRING data type, the current length of the character string in the destination area is set to the number of characters actually moved. If areas of the STRING data type are moved, specify "1" as the area length.

Syntax

The following syntax is used for the "Move block uninterruptible" instruction:

```
CALL UBLKMOV
VARIANT
SRCBLK := <Operand>
RET_VAL := <Operand>
DSTBLK := <Operand>
```

Parameters

The following table shows the parameters of the "Move block uninterruptible" instruction:

Parameter	Declaration	Data type	Memory area	Description
SRCBLK	Input	VARIANT	L, P	Specifies the memory area to be moved (source area).
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output on the RET_VAL parameter.
DSTBLK	Output	VARIANT	L, P	Specifies the memory area to which the block is to be moved (destination area).

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8152	The data types WSTRING, WCHAR and BOOL are not supported.
8091	The source and destination area are in a data block that is not relevant for program execution.
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

```
STL                              Explanation
CALL UBLKMOV                     // The "Move block uninterruptible" instruction is
                                 called.
VARIANT                          // Data type of the instruction
SRCBLK := P#M100.0 BYTE 10       // 10 Bytes starting from MB100 are moved.
RET_VAL := "Tag_ErrorCode"       // If an error occurs during the move operation, its
                                 error code is output in the variable "Tag_ErrorCode".
DSTBLK := P#M200.0 BYTE 10       // The moved bytes are written to the 10 bytes
                                 starting from MB200
```

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Switching display formats in the program status (Page 2980)

GET_ERR_ID: Get error ID locally (Page 3852)

FILL: Fill block

Description

The instruction "Fill block" fills a memory area (destination area) with the content of another memory area (source area). The instruction "Fill block" moves the content of the source area to the destination area until the destination area is completely written. The move operation takes place in the direction of ascending addresses.

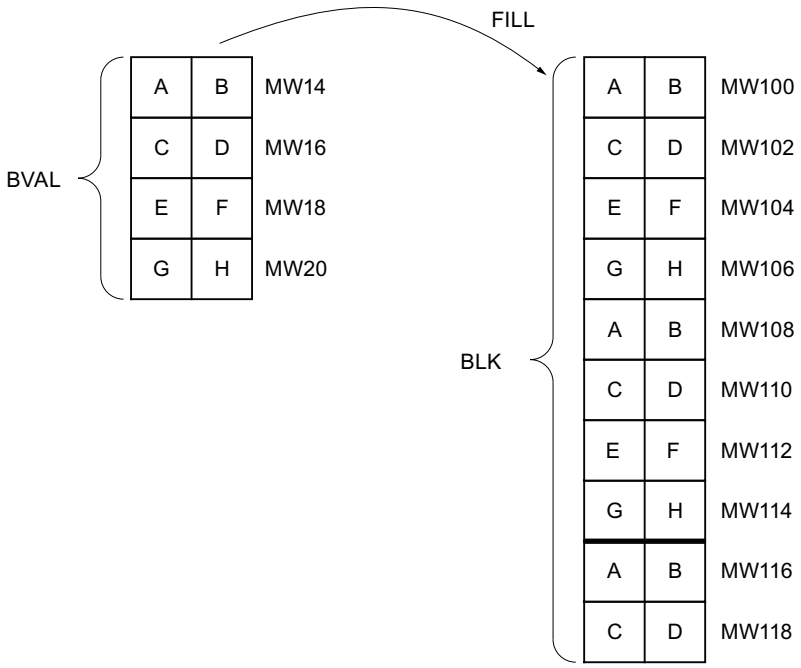
You use VARIANT to define the source and destination areas.

Note

The tags of the instruction can only be used in data blocks in which the attribute "Optimized block access" is not set. If the tag with the retentive setting "Set in IDB" has been declared, you can also use the tags "with optimized access".

For blocks with the "Optimized block access" attribute, you can use the "FILL_BLK: Fill block" instruction.

The following figure shows the principle of the move operation:



Example: The contents of the range MW100 to MW118 are to be pre-assigned with the contents of the memory words MW14 to MW20.

Consistency of source data and destination data

Note that while the instruction "Fill block" is being executed, the source data remains unchanged; otherwise the consistency of the destination data cannot be guaranteed.

Memory areas

The instruction "Fill block" copies the following memory areas:

- Areas of a data block
- Bit memory
- Process image of the inputs
- Process image of the outputs

General rules for moving

The source and destination areas must not overlap. If the destination area to be preset is not an integer multiple of the length of the input parameter BVAL, the destination area is nevertheless written up to the last byte.

If the destination area to be preset is smaller than the source area, the function only copies as much data as can be written to the destination area.

If the destination or source block actually present is smaller than the assigned memory area for the source or destination area (BVAL, BLK parameters), no data will be transferred.

If the ANY pointer (source or destination) is of the data type BOOL, it must be addressed absolutely and the length specified must be divisible by 8; otherwise the instruction is not executed.

If the destination area is STRING data type, the instruction writes the entire string including the administration information.

Rules for moving structures

When you transfer a structure as an input parameter you must bear in mind that the length of a structure is always oriented to an even number of bytes. The structure will need one byte of additional memory space if you declare a structure with an odd number of bytes.

Syntax

The following syntax is used for the instruction "Fill block":

```
CALL FILL
VARIANT
BVAL := <Operand>
RET_VAL := <Operand>
BLK := <Operand>
```

Parameters

The following table shows the parameters of the "Fill block" instruction:

Parameter	Declaration	Data type	Memory area	Description
BVAL	Input	VARIANT	L, P	Specification of the memory area (source area) with the content used to fill the destination area at the BLK parameter.
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output on the RET_VAL parameter.
BLK	Output	VARIANT	L, P	Specification of the memory area that is filled with the content of the source area.

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8152	The data types WSTRING, WCHAR and BOOL are not supported.
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL FILL                    // The "Fill block" instruction is called.
VARIANT                      // Data type of the instruction
BVAL := P#M14.0 WORD 4       // The area from MW14 to MW20 is moved
RET_VAL := "Tag_ErrorCode"   // Error information
BLK := P#M100.0 WORD 10      // The memory area from MW100 to MW118 is filled with
                             the content of 4 words that are contained in the
                             memory area on the BVAL parameter
```

See also

Overview of the valid data types (Page 2236)
Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)
Evaluating errors with output parameter RET_VAL (Page 5370)
Call function blocks as single or multi-instances (Page 2182)
Basic information on the status word (Page 2466)
Querying and setting status bits in STL (Page 2467)
Switching display formats in the program status (Page 2980)
GET_ERR_ID: Get error ID locally (Page 3852)

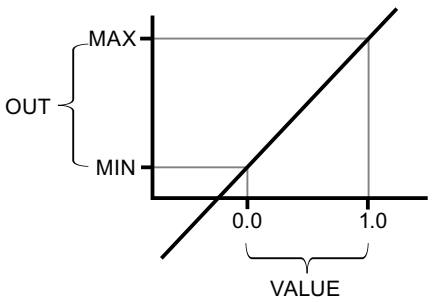
Conversion operations

SCALE_X: Scale

Description

You can use the "Scale" instruction to scale the value at the VALUE input by mapping it to a specified value range. When the "Scale" instruction is executed, the floating-point value at the VALUE input is scaled to the value range that was defined by the MIN and MAX parameters. The result of the scaling is an integer, which is stored in the RET_VAL output.

The following figure shows an example of how values can be scaled:



The instruction "Scale" works with the following equation:

OUT = [VALUE * (MAX – MIN)] + MIN

Note

For more information on the conversion of analog values, refer to the respective manual.

Syntax

The following syntax is used for the "Scale" instruction:

```
CALL SCALE_X
<Data type> <Data type>
MIN := <Operand>
VALUE := <Operand>
MAX := <Operand>
RET_VAL := <Operand>
```

Parameter

The following table shows the parameters of the "Scale" instruction:

Parameter	Declaration	Data type	Memory area	Description
MIN	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Low limit of the value range
VALUE	Input	Floating-point numbers	I, Q, M, D, L or constant	Value to be scaled. If you enter a constant, you must declare it.
MAX	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	High limit of the value range
RET_VAL	Output	Integers, floating-point numbers	I, Q, M, D, L	Result of scaling

You can select the data type of the instruction from the "<???>" drop-down list.

For additional information on valid data types, refer to "See also".

For additional information on declaring constants, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL SCALE_X                 // The "Scale" instruction is called
dest_type := INT             // Data types of the instruction
src_type := REAL             // Data type of the VALUE parameter
```

```
STL                          Explanation
MIN := "Tag_Minimum"         // Low limit
VALUE := "Tag_Value"         // Floating-point value
MAX := "Tag_Maximum"         // High limit
RET_VAL := "Tag_Result"      // Result of the instruction
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
MIN	Tag_Minimum	10
VALUE	Tag_Value	0.5
MAX	Tag_Maximum	30
OUT	Tag_Result	20

The value at the "Tag_Value" input is scaled to the range of values defined by the values at the "Tag_Minimum" and "Tag_Maximum" inputs. The result is stored in the "Tag_Result" output.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

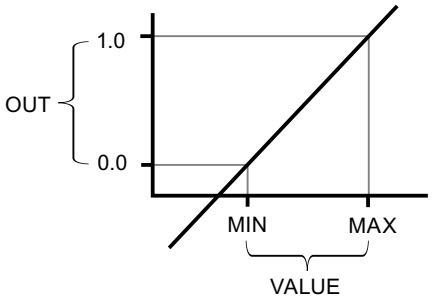
Declaring global constants (Page 2489)

NORM_X: Normalize

Description

You can use the instruction "Normalize" to normalize the value of the tag at the VALUE input by mapping it to a linear scale. You can use the MIN and MAX parameters to define the limits of a value range that is applied to the scale. The result at the RET_VAL output is calculated and stored as a floating-point number depending on the location of the value to be normalized within this value range. If the value to be normalized is equal to the value at the MIN input, the OUT output has the value "0.0". If the value to be normalized is equal to the value at input MAX, output OUT returns the value "1.0".

The following figure shows an example of how values can be normalized:



The "Normalize" instruction works with the following equation:

OUT = (VALUE – MIN) / (MAX – MIN)

Note

For more information on the conversion of analog values, refer to the respective manual.

Syntax

The following syntax is used for the "Normalize" instruction:

```
CALL NORM_X
<Data type> <Data type>
MIN := <Operand>
VALUE := <Operand>
MAX := <Operand>
RET_VAL := <Operand>
```

Parameters

The following table shows the parameters of the instruction "Normalize":

Parameter	Declaration	Data type	Memory area	Description
MIN [1)]	Input	Integers, floating-point numbers	I, Q, M, D, L	Low limit of the value range
VALUE [1)]	Input	Integers, floating-point numbers	I, Q, M, D, L	Value to be normalized

Parameter	Declaration	Data type	Memory area	Description
MAX [1]	Input	Integers, floating-point numbers	I, Q, M, D, L	High limit of the value range
RET_VAL	Output	Floating-point numbers	I, Q, M, D, L	Result of the normalization
[1] If you use constants in these three parameters, you only need to declare one of them.				

You can select the data type of the instruction from the "<???>" drop-down list.

For additional information on valid data types, refer to "See also".

For additional information on declaring constants, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL NORM_X                  // The "Normalize" instruction is called.
src_type := INT              // Data types of the instruction
dest_type := REAL            // Data type of the RET_VAL parameter
MIN := "Tag_Minimum"         // Low limit
VALUE := "Tag_Value"         // Floating-point value
MAX := "Tag_Maximum"         // High limit
RET_VAL := "Tag_Result"      // Result of the instruction
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
MIN	Tag_Minimum	10
VALUE	Tag_Value	20
MAX	Tag_Maximum	30
RET_VAL	Tag_Result	0.5

The value at the "Tag_Value" input is mapped to the range of values that were defined by the values at the "Tag_Minimum" and "Tag_Maximum" inputs. The tag value at the "Tag_Value" input is normalized to the defined value range. The result is stored as a floating-point number in the "Tag_Result" output.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Declaring global constants (Page 2489)

VARIANT instructions

VARIANT_TO_DB_ANY: Convert VARIANT to DB_ANY

Description

The instruction "Convert VARIANT to DB_ANY" is used to read the number of the data block from a VARIANT and convert it to the data type DB_ANY.

Requirements

If the conditions are met, the instruction is executed. If the conditions are not met, "0" is output as data block number.

The output tag...	homed...	Conversion options
VARIANT	...a data block which is an instance data block of an UDT or SDT	The output tag can be converted to a data block number.
VARIANT	...an Array DB	The output tag can be converted to a data block number.
VARIANT	...an object with an elementary data type	It is not possible to convert the output tag to a database number because a data block can never comprise only one elementary data type.
VARIANT	...a structure within a data block	It is not possible to convert the output tags to a database number because it is only a part within the data block.

Syntax

The following syntax is used for the "Convert VARIANT To DB_ANY" instruction:

```
CALL VARIANT_TO_DB_ANY
IN := <Operand>
ERR := <Operand>
RET_VAL := <Operand>
```

Parameter

The following table shows the parameters of the "Convert VARIANT to DB_ANY" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	VARIANT	L	Tag to be read
ERR	Output	INT	I, Q, M, D, L	Error information
RET_VAL	Output	DB_ANY	I, Q, M, D, L	Result: Number of the DB

For additional information on valid data types, refer to "See also".

ERR parameter

The following table shows the meaning of the values of the ERR parameter:

Error code* (W#16#...)	Explanation
0000	No error
252C	The VARIANT data type has the value "0"
8130	The number of the data block is "0"
8131	The data block does not exist
8132	The data block is too short
8134	The data block is write protected
8150	The VARIANT data type has the value "0"
8154	The data type is not correctly declared
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

STL	Explanation
CALL VARIANT_TO_DB_ANY	// The "Convert VARIANT to DB_ANY" instruction is called.
IN := #InputofTypeVARIANT	// The tag #InputofTypeVARIANT is read.
RET_VAL := "OutputofTypeDBANY"	// The number of the DB is written to the "OutputofTypeDBANY" data block.
ERR := "Tag_Error"	// Error information

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)
Call function blocks as single or multi-instances (Page 2182)
Basic information on the status word (Page 2466)
Querying and setting status bits in STL (Page 2467)
Declaring global constants (Page 2489)
VARIANT (Page 2277)

DB_ANY_TO_VARIANT: Convert DB_ANY to VARIANT

Description

The instruction "Convert DB_ANY to VARIANT" is used to read the number of the data block from a DB_ANY and convert it to the data type VARIANT.

Requirements

If the conditions are met, the instruction is executed. If the conditions are not met or the data block does not exist, the value "0" is output. All additional accesses to this value "0" fail.

The output tag...	homed...	Conversion options
DB_ANY	...a data block which is an instance data block of an UDT or SDT	It is possible to convert the output tag to the VARIANT data type.
DB_ANY	...a DB which is an ARRAY DB	It is possible to convert the output tag to the VARIANT data type.
DB_ANY	...a data block which is an instance data block of a function block or a global data block	It is not possible to convert the output tags to the VARIANT data type.

Syntax

The following syntax is used for the "Convert DB_ANY to VARIANT" instruction:

```
CALL DB_ANY_TO_VARIANT
IN := <Operand>
ERR := <Operand>
RET_VAL := <Operand>
```

Parameter

The following table shows the parameters of the "Convert DB_ANY to VARIANT" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	DB_ANY	I, Q, M, D, L	Tag to be read
ERR	Output	INT	I, Q, M, D, L	Error information
RET_VAL	Output	VARIANT	L	Result

For additional information on valid data types, refer to "See also".

ERR parameter

The following table shows the meaning of the values of the ERR parameter:

Error code* (W#16#...)	Explanation
0000	No error
8130	The number of the data block is "0"
8131	The data block does not exist
8132	The data block is too short
8134	The data block is write protected
8154	The data type is not correctly declared
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

STL	Explanation
CALL DB_ANY_TO_VARIANT	// The "Convert DB_ANY to VARIANT" instruction is called.
IN := #InputofTypeDBANY	// The tag #InputofTypeDBANY is read.
RET_VAL := "OutputofTypeVARIANT"	// The number of the DB is written to the "OutputofTypeVARIANT" data block.
ERR := "Tag_Error"	// Error information

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Declaring global constants (Page 2489)

VARIANT (Page 2277)

Other

SCALE: Scale

Description

You can use the "Scale" instruction to convert the integer at the IN parameter into a floating-point number that can be scaled in physical units between a low limit and a high limit. You can use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is scaled. The result of the instruction is output at the OUT parameter.

The "Scale" instruction works with the following equation:

OUT = [((FLOAT (IN) – K1)/(K2–K1)) * (HI_LIM–LO_LIM)] + LO_LIM

The values of the constants "K1" and "K2" are determined by the signal state of the BIPOLAR parameter. The following signal states are possible at the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and lies in a value range between -27 648 and 27 648. In this case the constant "K1" has the value "-27 648.0" and the constant "K2" the value "+27 648.0".
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and lies in a value range between 0 and 27 648. In this case the constant "K1" has the value "0.0" and the constant "K2" the value "+27 648.0".

When the value at the IN parameter is greater than the value of the constant "K2", the result of the instruction is set to the value of the high limit (HI_LIM) and an error is output.

When the value at the IN parameter is less than the value of the constant "K1", the result of the instruction is set to the value of the low limit (LO_LIM) and an error is output.

When the indicated low limit is greater than the high limit (LO_LIM > HI_LIM), the result is scaled inversely proportionate to the input value.

Syntax

The following syntax is used for the "Scale" instruction:

```
CALL SCALE
IN := <Operand>
HI_LIM := <Operand>
LO_LIM := <Operand>
BIPOLAR := <Operand>
```

```
RET_VAL := <Operand>
OUT := <Operand>
```

Parameter

The following table shows the parameters of the "Scale" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	INT	I, Q, M, D, L, P or constant	Input value to be scaled
HI_LIM	Input	REAL	I, Q, M, D, L, P or constant	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P or constant	Low limit
BIPOLAR	Input	BOOL	I, Q, M, D, L or constant	Indicates whether the value at the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
RET_VAL	Output	WORD	I, Q, M, D, L, P	Error information
OUT	Output	REAL	I, Q, M, D, L, P	Result of the instruction

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the constant "K2" or less than the value of the constant "K1".
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL SCALE                   // The "Scale" instruction is called
IN := "Tag_InputValue"       // Specifies the value to be converted and scaled
HI_LIM := "Tag_HighLimit"    // High limit
LO_LIM := "Tag_LowLimit"     // Low limit
BIPOLAR := "Tag_Bipolar"     // Indicates whether the value at the IN parameter is
                             to be interpreted as bipolar or unipolar
RET_VAL := "Tag_ErrorCode"   // Error information
OUT := "Tag_OutputValue"     // Result of the instruction
```

The following table shows the values of the various operands before execution of the instruction:

Parameter	Operand	Value
IN	Tag_InputValue	22
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	0.0

The following table shows the values of the various operands after execution of the instruction:

Parameter	Operand	Value
IN	Tag_InputValue	22
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	50.03978588

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Switching display formats in the program status (Page 2980)

GET_ERR_ID: Get error ID locally (Page 3852)

UNSCALE: Unscale

Description

The "Unscale" instruction unscales the floating-point number at the IN parameter in physical units between a low limit value and a high limit value and converts it to an integer. You can use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is unscaled. only the currently set state at the OUT parameter.

The "Unscale" instruction works with the following equation:

OUT = [((IN–LO_LIM)/(HI_LIM–LO_LIM)) * (K2–K1)] + K1

The values of the constants "K1" and "K2" are determined by the signal state of the BIPOLAR parameter. The following signal states are possible at the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and lies in a value range between -27 648 and 27 648. In this case the constant "K1" has the value "-27 648.0" and the constant "K2" the value "+27 648.0".
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and lies in a value range between 0 and 27 648. In this case the constant "K1" has the value "0.0" and the constant "K2" the value "+27 648.0".

When the value at the IN parameter is greater than the value of the constant "HI_LIM", the result of the instruction is set to the value of the constant (K2) and an error is output.

When the value at the IN parameter is less than the value of the constant of the low limit (LO_LIM), the result of the instruction is set to the value of the constant (K1) and an error is output.

Syntax

The following syntax is used for the "Unscale" instruction:

```
CALL UNSCALE
IN := <Operand>
HI_LIM := <Operand>
LO_LIM := <Operand>
BIPOLAR := <Operand>
RET_VAL := <Operand>
OUT := <Operand>
```

Parameters

The following table shows the parameters of the "Unscale" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	REAL	I, Q, M, D, L, P or constant	Input value to be unscaled to an integer value
HI_LIM	Input	REAL	I, Q, M, D, L, P or constant	High limit
LO_LIM	Input	REAL	I, Q, M, D, L or constant	Low limit
BIPOLAR	Input	BOOL	I, Q, M, D, L or constant	Indicates whether the value at the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
RET_VAL	Output	WORD	I, Q, M, D, L, P	Error information
OUT	Output	INT	I, Q, M, D, L, P	Result of the instruction

Parameters RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the high limit (HI_LIM) or less than the value of the low limit (LO_LIM).
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL UNSCALE                 // The "Unscale" instruction is called
IN := "Tag_InputValue"       // Specifies the value to be unscaled
HI_LIM := "Tag_HighLimit"    // High limit
```

```
STL                              Explanation
LO_LIM := "Tag_LowLimit"         // Low limit
BIPOLAR := "Tag_Bipolar"         // Indicates whether the value at the IN parameter is
                                 to be interpreted as bipolar or unipolar
RET_VAL := "Tag_ErrorCode"       // Error information
OUT := "Tag_OutputValue"         // Result of the instruction
```

The following table shows the values of the various operands before execution of the instruction:

Parameter	Operand	Value
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	0.0

The following table shows the values of the various operands after execution of the instruction:

Parameter	Operand	Value
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	22

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Switching display formats in the program status (Page 2980)

GET_ERR_ID: Get error ID locally (Page 3852)

Program control operations

Runtime control

ENDIS_PW: Limit and enable password legitimation

Description

You use the "Limit and enable password legitimation" instruction to specify whether passwords configured for the CPU are legitimized or not. In this way, you can prevent legitimized connections, even if the correct password is known.

When you call the instruction and the REQ parameter has the signal state "0", the currently set state is displayed at the output parameters. If changes have been made to the input parameters, these are not transferred to the output parameters.

When you call the instruction and the REQ parameter has the signal state "1", the signal state is taken from the input parameters (F_PWD, FULL_PWD, R_PWD, HMI_PWD). FALSE means that legitimation per password is not permitted; TRUE means the password can be used.

Disable or enabling of the passwords can be allowed or prohibited individually. For example, all passwords can be prohibited except the fail-safe password. You can thus limit access options to a small user group. The output parameters (F_PWD_ON, FULL_PWD_ON, R_PWD_ON, HMI_PWD_ON) always show the current status of the password use, regardless of the REQ parameter.

Passwords that are not configured must have the signal state TRUE at the input and return the signal state TRUE at the output. The fail-safe password can only be configured for an F-CPU and must therefore always be interconnected with the signal state TRUE in a standard CPU. If the instruction returns an error, the call has no effect and the previous lock is still in effect.

Disabled passwords can be enabled again under the following conditions:

- The CPU is reset to its factory settings.
- The front panel of the S7-1500 CPU supports a dialog that allows you to navigate to the appropriate menu in which the passwords can be enabled again.
- When you call the "Limit and enable password legitimation" instruction, the input parameter of the desired password has the signal state "1".
- Set the mode selector to STOP. The restriction to password legitimation is re-established as soon as the switch is set back to RUN.
- Plugging an empty memory card (transfer module or program card) into an S7-1200 CPU.
- The transition from POWER OFF to POWER ON disables the protection in the S7-1200 CPU. The "Limit and enable password legitimation" instruction must be called once again in the program (e.g. in the startup OB).

Note

The "Limit and enable password legitimation" instruction blocks access from HMI systems if the HMI password has not been enabled.

Note

Existing legitimized connections retain their access rights and you cannot use the "Limit and enable password legitimation" instruction to restrict them.

Preventing accidental lockout of an S7-1500 CPU

The settings can be made on the front panel of the CPU and the CPU saves the most recent setting.

To prevent accidental lockout, the protection can be disabled by setting the mode switch to STOP with an S7-1500 CPU. The protection is automatically enabled again by setting mode selector to RUN without having to call the "Limit and enable password legitimation" instruction again or taking additional actions on the front panel.

Preventing accidental lockout of an S7-1200 CPU

Because the S7-1200 CPU does not have a mode switch, protection is disabled with POWER OFF - POWER ON. This means that it is possible and advisable to prevent accidental lockout with the help of specific program sequences within your program.

To do so, program a time control either by means of a cyclic interrupt OB or a timer in the Main OB (OB 1). This gives you the option of calling the "Limit and enable password legitimation" instruction in the respective OB (e.g., OB 1 or OB 35) relatively quickly after a transition from POWER OFF to POWER ON and the associated disabling of the protection. Call the instruction in the startup OB (OB 100) to keep the time window in which the instruction is inactive and there are no limitations for password legitimation relatively small. This procedure will give you the best possible protection from unauthorized access.

If there has been an accidental lockout, you can skip the call in the startup OB (by querying an input parameter, for example) and you have the set time (for example, 10 seconds up to 1 minute) to establish a connection to the CPU before the lock becomes active once again.

If there is no timer in your program code and a lockout has occurred, insert an empty transfer card or an empty program card into the CPU. The empty transfer card or program card deletes the internal load memory of the CPU. You must then download the user program once again from STEP 7 to the CPU.

Procedure for lost passwords with an S7-1200 CPU

If you have lost the password for a password-protected S7-1200 CPU, delete the password-protected program with an empty transfer card or program card. The empty transfer card or

program card deletes the internal load memory of the CPU. You can then load a new user program from STEP 7 Basic in the CPU.

WARNING

Inserting an empty transfer card

When you insert a transfer card in a CPU during runtime, the CPU goes to STOP mode. If operating states are unstable, controllers may fail and thus cause the uncontrolled operation of the controlled devices. This leads to an unforeseen operation of the automation system which can cause deadly or serious injuries and/or damage to property.

Once you have pulled the transfer card, the content of the transfer card is available in the internal load memory. Make sure that the card does not include a program at this point.

WARNING

Inserting an empty program card

When you insert a program card in a CPU during runtime, the CPU goes to STOP mode. If operating states are unstable, controllers may fail and thus cause the uncontrolled operation of the controlled devices. This leads to an unforeseen operation of the automation system which can cause deadly or serious injuries and/or damage to property.

Make sure that the program card is empty. The internal load memory is copied to the empty program card. Once you have pulled the previously empty program card, the internal load memory is empty.

You must remove the transfer card or program card before you put the CPU into RUN mode.

Effects of password use on the operating modes

The following table shows what effects the password use by means of the "Limit and enable password legitimation" instruction has on the operating modes and the corresponding user actions.

Action	Password protection by means of the instruction
Basic state after • Operating mode switch to STOP • Reset memory manually (PG, switch, change of MC (Motion Control)) • Reset to factory setting	Not activated (no restrictions)
Basic state after POWER ON	• S7-1200 CPU: The lock is disabled and the instruction must be called once again in the program (e.g., in the startup OB). • S7-1500 CPU: Enabled (if a lock was activated before POWER OFF). The option of not allowing passwords is retentive.
Operating mode transition RUN/STARTUP/HOLD -> STOP (by terminating the instruction, an error or communication) or STOP -> STARTUP/RUN/HOLD	Activated Passwords still cannot be used.

Syntax

The following syntax is used for the "Limit and enable password legitimation" instruction:

```
CALL ENDIS_PW
REQ := <Operand>
F_PWD := <Operand>
FULL_PWD := <Operand>
R_PWD := <Operand>
HMI_PWD := <Operand>
F_PWD_ON := <Operand>
FULL_PWD_ON := <Operand>
R_PWD_ON := <Operand>
HMI_PWD_ON := <Operand>
RET_VAL := <Operand>
```

Parameter

The following table shows the parameters of the "Limit and enable password legitimation" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	When the REQ parameter has the signal state "0", the currently set signal state of the passwords is queried.
F_PWD	Input	BOOL	I, Q, M, D, L or constant	Read/write access including fail-safe • F_PWD = "0": Do not allow password • F_PWD = "1": Allow password
FULL_PWD	Input	BOOL	I, Q, M, D, L or constant	Read/write access • FULL_PWD = "0": Do not allow password • FULL_PWD = "1": Allow password
R_PWD	Input	BOOL	I, Q, M, D, L or constant	Read access • R_PWD = "0": Do not allow password • R_PWD = "1": Allow password
HMI_PWD	Input	BOOL	I, Q, M, D, L or constant	HMI access • HMI_PWD = "0": Do not allow password • HMI_PWD = "1": Allow password
F_PWD_ON	Output	BOOL	I, Q, M, D, L	Read/write access including fail-safe • F_PWD_ON = "0": Password not allowed • F_PWD_ON = "1": Password allowed
FULL_PWD_ON	Output	BOOL	I, Q, M, D, L	Read/write access status • FULL_PWD_ON = "0": Password not allowed • FULL_PWD_ON = "1": Password allowed
R_PWD_ON	Output	BOOL	I, Q, M, D, L	Read-access status • R_PWD_ON = "0": Password not allowed • R_PWD_ON = "1": Password allowed
HMI_PWD_ON	Output	BOOL	I, Q, M, D, L	HMI-access status • HMI_PWD_ON = "0": Password not allowed • HMI_PWD_ON = "1": Password allowed
RET_VAL	Output	WORD	I, Q, M, D, L	Error information

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8090	The "Limit and enable password legitimation" instruction is not supported

Error code* (W#16#...)	Explanation
80D0	The password for fail-safe is not configured. The signal state must be TRUE for standard CPUs.
80D1	The read/write access is not configured
80D2	The read access is not configured
80D3	The HMI access is not configured
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

RE_TRIGR: Restart cycle monitoring time

Description

You can use the "Restart cycle monitoring time" instruction to restart the cycle time monitoring of the CPU. The cycle time monitoring then restarts with the time you have set in the CPU configuration.

The instruction "Restart cycle monitoring time" can be called, regardless of the priority, in all blocks.

If the instruction is called in a block with a higher priority, for example, a hardware interrupt, diagnostic interrupt or cyclic interrupt, the instruction is not be executed.

The instruction "Restart cycle monitoring time" can be called a maximum of 10 times in a program cycle.

Syntax

The following syntax is used for the "Restart cycle monitoring time" instruction:

```
CALL RE_TRIGR
```

Parameters

The "Restart cycle monitoring time" instruction has no parameters and returns no error information.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

STP: Exit program

Description

You can use the "Exit program" instruction to set the CPU to STOP mode and therefore to terminate the execution of the program. The effects of changing from RUN to STOP depend on the CPU configuration.

Syntax

The following syntax is used for the "Exit program" instruction:

```
CALL STP
```

Parameters

The "Exit program" instruction has no parameters and returns no error information.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

GET_ERROR: Get error locally

Description

The "Get error locally" instruction is used to query the occurrence of errors within a block. This is usually to access error. If the system reports an error during the processing of the block, detailed information is stored in the operand at the OUT output for the first error to occur during the execution of the block since the last execution of the instruction.

Only operands of the "ErrorStruct" system data type can be specified at the OUT output. The "ErrorStruct" system data type specifies the exact structure in which the information about the error is stored. Using additional instructions, you can evaluate this structure and program an appropriate response. If several errors occur in the block, the error information about the next error to occur in the instruction is output only after the first error that occurred has been remedied.

Note

The OUT output is only changed if error information is present. To set the output back to "0" after handling an error, you have the following options:

- Declare the tag in the "Temp" section of the block interface.
- Reset the tag to "0" before calling the instruction.

You can find an example of how you can implement the instruction in combination with other troubleshooting options under "See also".

Note

The "Get error locally" instruction enables local error handling within a block. If "Get error locally" is inserted in the program code of a block, any predefined system reactions are ignored if errors occur.

Syntax

The following syntax is used for the "Get error locally" instruction:

```
CALL GET_ERROR
OUT := <Operand>
```

Parameters

The following table shows the parameters of the "Get error locally" instruction:

Parameter	Declaration	Data type	Memory area	Description
OUT	Output	ErrorStruct	D, L	Error information

Data type "ErrorStruct"

The following table shows the structure of the "ErrorStruct" data type:

Structure component		Data type	Description
ERROR_ID		WORD	Error ID
FLAGS		BYTE	Shows if an error occurred during a block call. 16#01: Error during a block call 16#00: No error during a block call
REACTION		BYTE	Default reaction: 0: Ignore (write error), 1: Continue with substitute value "0" (read error), 2: Skip instruction (system error)
CODE_ADDRESS		CREF	Information about the address and type of block
	BLOCK_TYPE	BYTE	Type of block where the error occurred: 1: OB 2: FC 3: FB
	CB_NUMBER	UINT	Number of the code block
	OFFSET	UDINT	Reference to the internal memory
MODE		BYTE	Information about the address of an operand
OPERAND_NUMBER		UINT	Operand number of the machine command
POINTER_NUMBER_LOCATION		UINT	(A) Internal pointer
SLOT_NUMBER_SCOPE		UINT	(B) Storage area in internal memory
DATA_ADDRESS		NREF	Information about the address of an operand
	AREA	BYTE	(C) Memory area: L: 16#40 – 4E, 86, 87, 8E, 8F, C0 – CE I: 16#81 Q: 16#82 M: 16#83 DB: 16#84, 85, 8A, 8B Range violations for a directly editable tag of data type DINT: 16#04
	DB_NUMBER	UINT	(D) Number of the data block
	OFFSET	UDINT	(E) Relative address of the operand

Structure component "ERROR_ID"

The following table shows the values that can be output at the structure component "ERROR_ID":

ID* (hexadecimal)	ID* (decimal)	Description
0	0	No error
2503	9475	Invalid pointer
2520	9504	Invalid STRING
2522	9506	Read error: Operand outside the valid range
2523	9507	Write error: Operand outside the valid range
2524	9508	Read error: Invalid operand
2525	9509	Write error: Invalid operand
2528	9512	Read error: Data alignment
2529	9513	Write error: Data alignment
252C	9516	Invalid pointer
2530	9520	Write error: Data block
2533	9523	Invalid pointer used
2538	9528	Access error: DB does not exist
2539	9529	Access error: Wrong DB used
253A	9530	Global data block does not exist
253C	9532	Faulty information or the function does not exist
253D	9533	System function does not exist
253E	9534	Faulty information or the function block does not exist
253F	9535	System block does not exist
2550	9552	Access error: DB does not exist
2551	9553	Access error: Wrong DB used
2575	9589	Error in the program nesting depth
2576	9590	Error in the local data distribution
2577	9591	The block property "Parameter passing via registers" is not selected.
25A0	9632	Internal error in TP
25A1	9633	Tag is write-protected
25A2	9634	Invalid numerical value of tag
2942	10562	Read error: Input
2943	10563	Write error: Output
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".		

Example

The following example shows how the instruction works:

```
                                Explanation
LABEL: L #Field[#index]         // Access to the #Field[#index] tag
```

	Explanation
L 40.5	// Load value 40.5
*R	// Multiply value of the tag by the value 40.5.
T #TagOut	// Transfer product to the operand #TagOut.
CALL GET_ERROR	// The "Get error locally" instruction is called.
OUT := #Error	// If an error occurs, detailed information is stored in the #Error operand.
T #Error.REACTION	// The structure component #Error.REACTION is read.
L 1	// Load value 1
==I	// The value of the structure component is compared with the value 1.
JC LABEL	// If the comparison of the two values is positive, the program execution returns to the jump label LABEL.

An error occurred accessing the "#Field[#index]" tag. The "#TagOut" operand returns the signal state "1" despite the read/access error and the multiplication is performed with the value "0.0". If this error scenario occurs, we recommend that you program the "Get error locally" instruction after the multiplication to get the error. The error information supplied by the "Get error locally" instruction is evaluated with a comparison. If the "#Error.REACTION" structure component has the value "1", there is a read/access error and the program execution begins again at the jump label LABEL.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Switching display formats in the program status (Page 2980)

Querying and fixing errors in the program code (Page 258)

Handling program execution errors (Page 2821)

GET_ERR_ID: Get error ID locally

Description

The "Get error ID locally" instruction is used to query the occurrence of errors within a block. This is usually to access error. If the system reports during the block processing errors within the block execution since the last execution of the instruction, the error ID of the first error that occurred in the tag is stored at the RET_VAL output.

Only operands of the WORD data type can be specified at the RET_VAL output. If several errors occur in the block, the error ID of the next error to occur in the instruction is output only after the first error that occurred has been remedied.

Note

The RET_VAL output is only changed if error information is present. To set the output back to "0" after handling an error, you have the following options:

- Declare the tag in the "Temp" section of the block interface.
- Reset the tag to "0" before calling the instruction.

The output of the "Get error ID locally" instruction is only set if error information is present. If one of these conditions is not fulfilled, the remaining program execution is not affected by the "Get error ID locally" instruction.

You can find an example of how you can implement the instruction in combination with other troubleshooting options under "See also".

Note

The "Get error ID locally" instruction enables local error handling within a block. If the "Get error ID locally" instruction is inserted in the program code of a block, any predefined system reactions are ignored if errors occur.

Syntax

The following syntax is used for the "Get error ID locally" instruction:

```
CALL GET_ERR_ID
RET_VAL := <Operand>
```

Parameters

The following table shows the parameters of the "Get error ID locally" instruction:

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Output	WORD	I, Q, M, D, L	Error ID

RET_VAL parameter

The following table shows the values that can be output at the RET_VAL parameter:

RET_VAL* (Hexadecimal)	RET_VAL* (Decimal)	Description
0	0	No error
2503	9475	Invalid pointer

RET_VAL* (Hexadecimal)	RET_VAL* (Decimal)	Description
2520	9504	Invalid STRING
2522	9506	Read error: Operand outside the valid range
2523	9507	Write error: Operand outside the valid range
2524	9508	Read error: Invalid operand
2525	9509	Write error: Invalid operand
2528	9512	Read error: Data alignment
2529	9513	Write error: Data alignment
252C	9516	Invalid pointer
2530	9520	Write error: Data block
2533	9523	Invalid pointer used
2538	9528	Access error: DB does not exist
2539	9529	Access error: Wrong DB used
253A	9530	Global data block does not exist
253C	9532	Faulty information or the function does not exist
253D	9533	System function does not exist
253E	9534	Faulty information or the function block does not exist
253F	9535	System block does not exist
2550	9552	Access error: DB does not exist
2551	9553	Access error: Wrong DB used
2575	9589	Error in the program nesting depth
2576	9590	Error in the local data distribution
2577	9591	The block property "Parameter passing via registers" is not selected.
25A0	9632	Internal error in TP
25A1	9633	Tag is write-protected
25A2	9634	Invalid numerical value of tag
2942	10562	Read error: Input
2943	10563	Write error: Output
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".		

Example

The following example shows how the instruction works:

```
                                  Explanation
L #Field[#index]                  // Access to the #Field[#index] tag
L 40.5                            // Load value 40.5
*R                                // Multiply value of the tag by the value 40.5.
T #TagOut                         // Transfer product to the operand #TagOut.

CALL GET_ERR_ID                   // The "Get error ID locally" instruction is called.
RET_VAL := #TagID                 // If an error occurs, an error ID is stored in the #TagID operand.
```

```
                                    Explanation

T #TagID                            // The error ID at the #TagID operand is read.
L 16#2522                           // Load value 16#2522
==I                                 // The value of the error ID is compared with the value of 16#2522.

CALL MOVE                           // The "Move value" instruction is called.
VARIANT                             // Data type of the instruction
IN := 100.0                         // The value 100.0 is copied to the #TagOut operand.
OUT := #TagOut                      // Output of the instruction
```

An error occurred accessing the "#Field[#index]" tag. The "#TagOut" operand returns the signal state "1" despite the read/access error and the multiplication is performed with the value "0.0". If this error scenario occurs, we recommend that you program the "Get error ID locally" instruction after the multiplication to get the error. The error ID supplied by the "Get error ID locally" instruction is evaluated with a comparison. If the "#TagID" operand returns ID 16#2522, there is a read/access error and the value "100.0" is moved to the "#TagOut" output.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Switching display formats in the program status (Page 2980)

Querying and fixing errors in the program code (Page 258)

Handling program execution errors (Page 2821)

INIT_RD: Initialize all retain data

Description

The "Initialize all retain data" instruction is used to reset the retentive data of all data blocks, bit memories and SIMATIC timers and counters at the same time. The instruction can only be executed within a startup OB because the execution exceeds the program cycle duration.

Syntax

The following syntax is used for the "Initialize all retain data" instruction:

```
CALL INIT_RD
REQ := <Operand>
RET_VAL := <Operand>
```

Parameters

The following table shows the parameters of the "Initialize all retain data" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	If the input "REQ" has the signal state "1", all retentive data are reset.
RET_VAL	Output	INT	I, Q, M, D, L	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B5	The instruction cannot be executed because it was not programmed within a startup OB.
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

STL	Explanation
CALL INIT_RD	// The "Initialize all retain data" instruction is called
REQ := "Tag_REQ"	// If the operand "Tag_REQ" has the signal state "1", all retentive data are reset.
RET_VAL := "Tag_RET_VAL"	// Error code

If the "Tag_REQ" operand has signal state "1", the instruction is executed. All retentive data of all data blocks, bit memories and SIMATIC timers and counters are reset.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Switching display formats in the program status (Page 2980)

GET_ERR_ID: Get error ID locally (Page 3852)

WAIT: Configure time delay

Description

The "Configure time delay" instruction is used to halt the execution of the program for a set period of time. You indicate the period of time in microseconds on the WT parameter of the instruction.

You can configure time delays of up to 32767 microseconds (µs). The smallest possible delay time depends on the particular CPU and corresponds to the execution time of the "Configure time delay" instruction.

The execution of the instruction can be interrupted by higher priority events.

The "Configure time delay" instruction returns no error information.

Syntax

The following syntax is used for the "Configure time delay" instruction:

```
CALL WAIT
WT := <Operand>
```

Parameters

The following table shows the parameters of the "Configure time delay" instruction:

Parameters	Declaration	Data type	Memory area	Description
WT	Input	INT	I, Q, M, D, L, P or constant	Time delay in microseconds (µs)

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

RUNTIME: Measure program runtime

Description

The "Measure program runtime" instruction is used to measure the runtime of the entire program, individual blocks or command sequences.

If you want to measure the runtime of your entire program, call the instruction "Measure program runtime" in OB1. Measurement of the runtime is started with the first call and the output RET_VAL returns the runtime of the program after the second call. The measured runtime includes all CPU processes that can occur during the program execution, for example, interruptions caused by higher-level events or communication. The instruction "Measure program runtime" reads an internal counter of the CPU and write the value to the IN-OUT parameter MEM. The instruction calculates the current program runtime according to the internal counter frequency and writes it to output RET_VAL.

If you want to measure the runtime of individual blocks or individual command sequences, you need three separate networks. Call the instruction "Measure program runtime" in an individual network within your program. You set the starting point of the runtime measurement with this first call of the instruction. Then you call the required program block or the command sequence in the next network. In another network, call the "Measure program runtime" instruction a

second time and assign the same memory to the IN-OUT parameter MEM as you did during the first call of the instruction. The "Measure program runtime" instruction in the third network reads an internal CPU counter and calculates the current runtime of the program block or the command sequence according to the internal counter frequency and writes it to the output RET_VAL.

The Measure program runtime" instruction uses an internal high-frequency counter to calculate the time. If the counter overruns, the instruction returns values <= 0.0. Ignore these runtime values.

Note

The runtime of a command sequence cannot be determined exactly, because the sequence of instructions within a command sequence is changed during optimized compilation of the program.

Syntax

The following syntax is used for the "Measure program runtime" instruction:

```
CALL RUNTIME
RET_VAL := <Operand>
MEM := <Operand>
```

Parameters

The following table shows the parameters of the "Measure program runtime" instruction:

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Output	LREAL	I, Q, M, D, L	Returns the measured runtime in seconds
MEM	InOut	LREAL	M, D, L	Saves the starting point of the runtime measurement

For additional information on valid data types, refer to "See also".

Example

The following example shows the how the instruction works based on the runtime calculation of a program block:

```
STL                                  Explanation
CALL RUNTIME                         // The "Measure program runtime" instruction is called.
   RET_VAL := "TagResult"            // Intermediate result
   MEM := "TagMemory"                // Saves the starting point of the runtime measurement.
```

STL	Explanation
CALL "Best_before_date", "Best_before_date_DB"	// The "Best_before_date" block is called and processed.
CALL RUNTIME	// The "Measure program runtime" instruction is called a second time.
RET_VAL := "TagResult"	// Result of the instruction
MEM := "TagMemory"	// Uses the start point of the runtime measurement as basis for the runtime measurement.

The starting point for the runtime measurement is set with the first call of the instruction and buffered as reference for the second call of the instruction in the "TagMemory" operand.

The "Best_before_date" program block FB1 is called.

When the program block FB1 has been processed, the instruction is executed a second time. The second call of the instruction calculates the runtime of the program block and writes the result to the "TagResult" operand on the RET_VAL output.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Timer operations (Page 3750)

Word logic operations

DECO: Decode

Description

You can use the "Decode" instruction to set a bit in the output value specified by the input value.

The "Decode" instruction reads the value of the IN parameter and sets the bit in the OUT parameter whose bit position corresponds to the value read. The other bits in the output value are filled with zeroes. When the value at the IN parameter is greater than 31, a modulo-32 operation is executed.

Syntax

The following syntax is used for the "Decode" instruction:

```
CALL DECO
UINT <Data type>
IN := <Operand>
OUT := <Operand>
```

Parameter

The following table shows the parameters of the "Decode" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	UINT	I, Q, M, D, L, P or constant	Position of the bit in the output value which is set
OUT	Output	Bit strings	I, Q, M, D, L, P	Output value

You can select the data type of the instruction from the "<???>" drop-down list.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
CALL DECO	// The "Decode" instruction is called
UINT WORD	// Data types of the instruction
IN := "Tag_Input"	// Input value that specifies the bit to be set in the output value
OUT := "Tag_Output"	// Result

The following table shows how the instruction functions using specific values:

Tag_Input	3	
	31 16	15 ... 3 ... 0
Tag_Output	0000 0000 0000 0000	0000 0000 0000 1000

The instruction reads bit number "3" from the value of the operand "Tag_Input" and sets the third bit to the value of the operand "Tag_Output".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

ENCO: Encode

Description

The instruction "Encode" is used to read the bit number of the least significant set bit in the input value and to output it at the parameter OUT.

The "Encode" instruction selects the least significant bit of the value at the IN parameter and writes this bit number to the operand at the OUT parameter. If the IN parameter has the value DW#16#00000001 or DW#16#00000000, the value "0" is output at the OUT parameter.

Syntax

The following syntax is used for the "Encode" instruction:

```
CALL ENCO
<Data type>
IN := <Operand>
OUT := <Operand>
```

Parameters

The following table shows the parameters of the "Encode" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	Bit strings	I, Q, M, D, L, P or constant	Input value
OUT	Output	INT	I, Q, M, D, L, P	Output value

You can select the data type of the instruction from the "<???>" drop-down list.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
CALL ENCO	// The "Encode" instruction is called

STL	Explanation
WORD	// Data type of the instruction
IN := "Tag_Input"	// Input value the least significant bit of which is read.
OUT := "Tag_Output"	// Bit number of least significant bit set in the input value

The following table shows how the instruction functions using specific values:

	31 16	15 ... 3 ... 0
“Tag_Input“	0000 1111 0000 0101	0000 1001 0000 1000
“Tag_Output“	3	

The instruction selects the least significant set bit of the variable "Tag_Input" and writes its bit position "3" to the variable "Tag_Output".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

SEL: Select

Description

Depending on a switch (G input), the "Select" instruction selects one of the inputs, IN0 or IN1 and copies its content to the OUT output. When the G input has the signal state "0", the value at the IN0 input is moved. If input G has the signal state "1", the value at input IN1 is copied to output OUT.

The instruction is only executed if the tags of all parameters have the same data type.

Syntax

The following syntax is used for the "Select" instruction:

```
CALL SEL
<Data type>
G := <Operand>
IN0 := <Operand>
```

```
IN1 := <Operand>
OUT := <Operand>
```

Parameter

The following table shows the parameters of the "Select" instruction:

Parameter	Declaration	Data type	Memory area	Description
G	Input	BOOL	I, Q, M, D, L or constant	Switch
IN0	Input	Bit strings, integers, floating-point numbers, timers, CHAR, TOD, LTOD, DATE, DT, LDT	I, Q, M, D, L, P or constant	First input value
IN1	Input	Bit strings, integers, floating-point numbers, TIME, CHAR, TOD, DATE, DT, LDT	I, Q, M, D, L, P or constant	Second input value
OUT	Output	Bit strings, integers, floating-point numbers, TIME, CHAR, TOD, DATE, DT, LDT	I, Q, M, D, L, P	Result

You can select the data type of the instruction from the "<data type>" drop-down list.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL SEL                     // The "Select" instruction is called
DWORD                        // Data type of the instruction
G := "Tag_Input_G"           // Signal state of the switch
IN0 := "Tag_Input0"          // First input value
IN1 := "Tag_Input1"          // Second input value
OUT := "Tag_Output"          // Value of the selected input
```

The following table shows how the instruction functions using specific values:

Parameter	Operand	Value
G	Tag_Input_G	1
IN0	Tag_Input0	W#16#0000
IN1	Tag_Input1	W#16#FFFF
OUT	Tag_Output	W#16#FFFF

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Additional instructions

DRUM: Implement sequencer

Description

The "Implement sequencer" instruction is used to assign the programmed output bits (OUT1 to OUT16) and the output word (OUT_WORD) with the programmed values of the OUT_VAL parameter of the relevant step. The specific step must thereby satisfy the conditions of the programmed enable mask in the S_MASK parameter while the instruction remains at this step. The instruction advances to the next step if the event for the step is true and the programmed time for the current step elapses, or if the value at the JOG parameter changes from "0" to "1". The instruction is reset when the signal state of the RESET parameter changes to "1". The current step is hereby equated to the preset step (DSP).

The amount of time spent on a step is determined by the product of the preset timebase (DTBP) and the preset counter value (S_PRESET) for each step. At the start of a new step, this calculated value is loaded into the DCC parameter, which contains the time remaining for the current step. If, for example the value of the DTBP parameter is "2" and the preset value for the first step is "100" (100 ms), the DCC parameter has the value "200" (200 ms).

A step can be programmed with a time value, an event or both. Steps that have an event bit and the time value "0" advance to the next step as soon as the signal state at the event bit is "1". Steps that are programmed only with a timer value start the time immediately. Steps that are programmed with an event bit and a timer value greater than "0" start the time when the signal state of the event bit is "1". The event bits are initialized with a signal state of "1".

When the sequencer is on the last programmed step (LST_STEP) and the time for this step has expired, the signal state at the Q parameter is set to "1"; otherwise it is set to "0". When the Q parameter is set, the instruction remains on the step until it is reset.

In the configurable mask (S_MASK), you can select the individual bits in the output word (OUT_WORD) and set or reset the output bits (OUT1 to OUT16) by means of the output values (OUT_VAL). If a bit of the configurable mask has signal state "1", the OUT_VAL value sets or resets the corresponding bit. If the signal state of a bit of the configurable mask is "0", the corresponding bit is left unchanged. All the bits of the configurable mask for all 16 steps are initialized with a signal state of "1".

The output bit at the OUT1 parameter corresponds to the least significant bit of the output word (OUT_WORD). The output bit at the OUT16 parameter corresponds to the most significant bit of the output word (OUT_WORD).

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Implement sequencer" instruction:

```
CALL DRUM, "Instance DB"
RESET := <Operand>
JOG := <Operand>
DRUM_EN := <Operand>
LST_STEP := <Operand>
EVENT1 - 16 := <Operand>
OUT1 - 16 := <Operand>
Q := <Operand>
OUT_WORD := <Operand>
ERR_CODE := <Operand>
```

Parameters

The following table shows the parameters of the "Implement sequencer" instruction:

Parameters	Declaration	Data type	Memory area	Description
RESET	Input	BOOL	I, Q, M, D, L or constant	The signal state "1" indicates a reset condition.
JOG	Input	BOOL	I, Q, M, D, L or constant	When the signal state changes from "0" to "1", the instruction advances to the next step.

Parameters	Declaration	Data type	Memory area	Description
DRUM_EN	Input	BOOL	I, Q, M, D, L or constant	The signal state "1" causes the sequencer to continue counting based on the event and time criteria.
LST_STEP	Input	BYTE	I, Q, M, D, L or constant	Number of the last programmed step.
EVENT(i), 1 ≤ i ≤ 16	Input	BOOL	I, Q, M, D, L or constant	Event bit (i); Initial signal state is "1".
OUT(j), 1 ≤ j ≤ 16	Output	BOOL	I, Q, M, D, L	Output bit (j)
Q	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that the time for the last step has elapsed.
OUT_WORD	Output	WORD	I, Q, M, D, L, P	Word address to which the sequencer writes the output values.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
JOG_HIS	Static	BOOL	I, Q, M, D, L or constant	JOG parameter history bit
EOD	Static	BOOL	I, Q, M, D, L or constant	The signal state "1" indicates that the time for the last step has elapsed.
DSP	Static	BYTE	I, Q, M, D, L, P or constant	Preset first step of the sequencer (1 to 16)
DSC	Static	BYTE	I, Q, M, D, L, P or constant	Current step of the sequencer
DCC	Static	DWORD	I, Q, M, D, L, P or constant	Remaining processing time for the current step
DTBP	Static	WORD	I, Q, M, D, L, P or constant	Preset timebase of the sequencer
PREV_TIME	Static	DWORD	I, Q, M, D, L or constant	System time of the previous call
S_PRESET	Static	ARRAY of WORD	I, Q, M, D, L or constant	Preset counter value for each step [1 to 16] where 1 clock pulse = 1 ms.

Parameters	Declaration	Data type	Memory area	Description
OUT_VAL	Static	ARRAY of BOOL	I, Q, M, D, L or constant	Output values for each step [1 to 16, 0 to 15].
S_MASK	Static	ARRAY of BOOL	I, Q, M, D, L or constant	Configurable mask for each step [1 to 16, 0 to 15]. Initial signal states are "1".

For additional information on valid data types, refer to "See also".

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

ERR_CODE*	Explanation
W#16#0000	No error
W#16#000B	The value of the LST_STEP parameter is less than 1 or greater than 16.
W#16#000C	The value of the DSC parameter is less than 1 or greater than the value of the LST_STEP parameter.
W#16#000D	The value of the DSP parameter is less than 1 or greater than the value of the LST_STEP parameter.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

In the following example the instruction advances from step 1 to step 2. The output bits (OUT1 to OUT16) and the output word (OUT_WORD) are set according to the mask configured for step 2 and the values of the OUT_VAL parameter.

Note

You can initialize static parameters in the data block.

```
STL                              Explanation
CALL DRUM, "DRUM_DB"             // The "Implement sequencer" instruction is called
                                 and the instance data block "DRUM_DB" is created.
RESET := "Tag_Reset"             // Reset input
JOG := "Tag_Input_Jog"           // On a positive signal edge, the instruction advances
                                 to the next step.
DRUM_EN := "Tag_Input_DrumEN"    // The signal state "1" allows the sequencer to
                                 advance based on the event and time criteria.
LST_STEP :=                      // Number of the last programmed step
"Tag_Number_LastStep"
EVENT1 := "MyTag_Event_1"        // Event bit 1
```

STL	Explanation
EVENT2 := "MyTag_Event_2"	// Event bit 2
EVENT3 := "MyTag_Event_3"	// Event bit 3
EVENT4 := "MyTag_Event_4"	// Event bit 4
EVENT5 := "MyTag_Event_5"	// Event bit 5
EVENT6 := "MyTag_Event_6"	// Event bit 6
EVENT7 := "MyTag_Event_7"	// Event bit 7
EVENT8 := "MyTag_Event_8"	// Event bit 8
EVENT9 := "MyTag_Event_9"	// Event bit 9
EVENT10 := "MyTag_Event_10"	// Event bit 10
EVENT11 := "MyTag_Event_11"	// Event bit 11
EVENT12 := "MyTag_Event_12"	// Event bit 12
EVENT13 := "MyTag_Event_13"	// Event bit 13
EVENT14 := "MyTag_Event_14"	// Event bit 14
EVENT15 := "MyTag_Event_15"	// Event bit 15
EVENT16 := "MyTag_Event_16"	// Event bit 16
OUT1 := "MyTag_Output_1"	// Output bit 1
OUT2 := "MyTag_Output_2"	// Output bit 2
OUT3 := "MyTag_Output_3"	// Output bit 3
OUT4 := "MyTag_Output_4"	// Output bit 4
OUT5 := "MyTag_Output_5"	// Output bit 5
OUT6 := "MyTag_Output_6"	// Output bit 6
OUT7 := "MyTag_Output_7"	// Output bit 7
OUT8 := "MyTag_Output_8"	// Output bit 8
OUT9 := "MyTag_Output_9"	// Output bit 9
OUT10 := "MyTag_Output_10"	// Output bit 10
OUT11 := "MyTag_Output_11"	// Output bit 11
OUT12 := "MyTag_Output_12"	// Output bit 12
OUT13 := "MyTag_Output_13"	// Output bit 13
OUT14 := "MyTag_Output_14"	// Output bit 14
OUT15 := "MyTag_Output_15"	// Output bit 15
OUT16 := "MyTag_Output_16"	// Output bit 16
Q := "Tag_Output_Q"	// The signal state "1" indicates that the time for the last step has elapsed
OUT_WORD := "Tag_OutputWord"	// Word address to which the sequencer writes the output values
ERR_CODE := "Tag_ErrorCode"	// Error information

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for initializing the input parameters:

Parameters	Operand	Address	Value
RESET	Tag_Reset	M0.0	FALSE
JOG	Tag_Input_JOG	M0.1	FALSE
DRUM_EN	Tag_Input_DrumEN	M0.2	TRUE
LST_STEP	Tag_Number_LastStep	MB1	B#16#08
EVENT2	MyTag_Event_2	M20.0	FALSE
EVENT4	MyTag_Event_4	M20.1	FALSE
EVENT6	MyTag_Event_6	M20.2	FALSE
EVENT8	MyTag_Event_8	M20.3	FALSE
EVENT10	MyTag_Event_10	M20.4	FALSE
EVENT12	MyTag_Event_12	M20.5	FALSE
EVENT14	MyTag_Event_14	M20.6	FALSE
EVENT16	MyTag_Event_16	M20.7	FALSE

The following values are saved in the "DRUM_DB" instance data block of the instruction:

Parameters	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSP	DBB13	W#16#0001
DSC	DBB14	W#16#0001
DCC	DBD16	DW#16#0000000A
DTBP	DBW20	W#16#0001
S_PRESET[1]	DBW26	W#16#0064
S_PRESET[2]	DBW28	W#16#00C8
OUT_VAL[1,0]	DBX58.0	TRUE
OUT_VAL[1,1]	DBX58.1	TRUE
OUT_VAL[1,2]	DBX58.2	TRUE
OUT_VAL[1,3]	DBX58.3	TRUE
OUT_VAL[1,4]	DBX58.4	TRUE
OUT_VAL[1,5]	DBX58.5	TRUE
OUT_VAL[1,6]	DBX58.6	TRUE
OUT_VAL[1,7]	DBX58.7	TRUE
OUT_VAL[1,8]	DBX59.0	TRUE
OUT_VAL[1,9]	DBX59.1	TRUE
OUT_VAL[1,10]	DBX59.2	TRUE
OUT_VAL[1,11]	DBX59.3	TRUE
OUT_VAL[1,12]	DBX59.4	TRUE
OUT_VAL[1,13]	DBX59.5	TRUE
OUT_VAL[1,14]	DBX59.6	TRUE
OUT_VAL[1,15]	DBX59.7	TRUE

Parameters	Address	Value
OUT_VAL[2,0]	DBX60.0	FALSE
OUT_VAL[2,1]	DBX60.1	FALSE
OUT_VAL[2,2]	DBX60.2	FALSE
OUT_VAL[2,3]	DBX60.3	FALSE
OUT_VAL[2,4]	DBX60.4	FALSE
OUT_VAL[2,5]	DBX60.5	FALSE
OUT_VAL[2,6]	DBX60.6	FALSE
OUT_VAL[2,7]	DBX60.7	FALSE
OUT_VAL[2,8]	DBX61.0	FALSE
OUT_VAL[2,9]	DBX61.1	FALSE
OUT_VAL[2,10]	DBX61.2	FALSE
OUT_VAL[2,11]	DBX61.3	FALSE
OUT_VAL[2,12]	DBX61.4	FALSE
OUT_VAL[2,13]	DBX61.5	FALSE
OUT_VAL[2,14]	DBX61.6	FALSE
OUT_VAL[2,15]	DBX61.7	FALSE
S_MASK[2,0]	DBX92.0	FALSE
S_MASK[2,1]	DBX92.1	TRUE
S_MASK[2,2]	DBX92.2	TRUE
S_MASK[2,3]	DBX92.3	TRUE
S_MASK[2,4]	DBX92.4	TRUE
S_MASK[2,5]	DBX92.5	FALSE
S_MASK[2,6]	DBX92.6	TRUE
S_MASK[2,7]	DBX92.7	TRUE
S_MASK[2,8]	DBX93.0	FALSE
S_MASK[2,9]	DBX93.1	FALSE
S_MASK[2,10]	DBX93.2	TRUE
S_MASK[2,11]	DBX93.3	TRUE
S_MASK[2,12]	DBX93.4	TRUE
S_MASK[2,13]	DBX93.5	TRUE
S_MASK[2,14]	DBX93.6	FALSE
S_MASK[2,15]	DBX93.7	TRUE

The output parameters are set to the following values before the execution of the instruction:

Parameters	Operand	Address	Value
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#FFFF
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	TRUE
OUT3	MyTag_Output_3	M4.2	TRUE
OUT4	MyTag_Output_4	M4.3	TRUE
OUT5	MyTag_Output_5	M4.4	TRUE

Parameters	Operand	Address	Value
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	TRUE
OUT8	MyTag_Output_8	M4.7	TRUE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	TRUE
OUT12	MyTag_Output_12	M5.3	TRUE
OUT13	MyTag_Output_13	M5.4	TRUE
OUT14	MyTag_Output_14	M5.5	TRUE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	TRUE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameters	Operand	Address	Value
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	FALSE
OUT3	MyTag_Output_3	M4.2	FALSE
OUT4	MyTag_Output_4	M4.3	FALSE
OUT5	MyTag_Output_5	M4.4	FALSE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	FALSE
OUT8	MyTag_Output_8	M4.7	FALSE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	FALSE
OUT12	MyTag_Output_12	M5.3	FALSE
OUT13	MyTag_Output_13	M5.4	FALSE
OUT14	MyTag_Output_14	M5.5	FALSE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	FALSE
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#4321
ERR_CODE	Tag_ErrorCode	MW10	W#16#0000

The following values are changed in the instance data block "DRUM_DB" of the instruction after the execution of the instruction:

Parameters	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSC	DBB14	W#16#0002
DCC	DBD16	DW#16#000000C8

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Switching display formats in the program status (Page 2980)

DCAT: Discrete control-timer alarm

Description

You can use the "Discrete control-timer alarm" to count the time from the point at which the CMD parameter issues the command to open or close. The time is accumulated until the preset time (PT) is exceeded or the information is received that the device was opened or closed (O_FB or C_FB) within the specified time. If the preset time is exceeded before the information on the opening or closing of the device is received, the corresponding alarm is activated. If the signal state of the command input changes before the preset time, the time is restarted.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The "Discrete control-timer alarm" instruction has the following effects on the input conditions:

- When the signal state at the CMD parameter changes from "0" to "1", the signal states of the parameters Q, CMD_HIS, ET (only if ET < PT), OA and CA are influenced as follows:
  - The parameters Q and CMD_HIS are set to "1".
  - The parameters ET, OA and CA are reset to "0".
- When the signal state at the parameter CMD changes from "1" to "0", the parameters Q, ET (only if ET < PT), OA, CA and CMD_HIS are reset to "0".

- When the signal state at the CMD and CMD_HIS parameters is "1" and the parameter O_FB is set to "0", the time difference (ms) since the last execution of the instruction is added to the value at the parameter ET. If the value of the ET parameter exceeds the value of the PT parameter, the signal state at the OA parameter is set to "1". If the value of the ET parameter does not exceed the value of the PT parameter, the signal state of the OA parameter is reset to "0". The value of the parameter CMD_HIS is reset to the value of the parameter CMD.
- If the signal state of the CMD, CMD_HIS, and O_FB parameters are set to "1" and the C_FB parameter has the value "0", the signal state of the OA parameter is set to "0". The value of the ET parameter is set to the value of the PT parameter. If the signal state of the O_FB parameter changes to "0", the alarm is set the next time the instruction is executed. The value of the CMD_HIS parameter is set to the value of the CMD parameter.
- If the parameters CMD, CMD_HIS and C_FB have the value "0", the time difference (ms) since the last execution of the instruction is added to the value of the parameter ET. If the value of the ET parameter exceeds the value of the PT parameter, the signal state of the parameter CA is set to "1". If the value at the parameter PT is not exceeded, the parameter CA has the signal state "0". The value of the CMD_HIS parameter is set to the value of the CMD parameter.
- If the parameters CMD, CMD_HIS and O_FB have the signal state "0" and the parameter C_FB is set to "1", the parameter CA is set to "0". The value of the ET parameter is set to the value of the PT parameter. If the signal state of the C_FB parameter changes to "0", the alarm is set the next time the instruction is executed. The value of the CMD_HIS parameter is set to the value of the CMD parameter.
- If the parameters O_FB and C_FB simultaneously have the signal state "1", the signal states of both alarm outputs are set to "1".

The "Discrete control-timer alarm" instruction returns no error information.

Syntax

The following syntax is used for the "Discrete control-timer alarm" instruction:

```
CALL DCAT, "Instance DB"
CMD := <Operand>
O_FB := <Operand>
C_FB := <Operand>
Q := <Operand>
OA := <Operand>
CA := <Operand>
```

Parameters

The following table shows the parameters of the "Discrete control-timer alarm" instruction:

Parameters	Declaration	Data type	Memory area	Description
CMD	Input	BOOL	I, Q, M, D, L or constant	The signal state "0" indicates the "close" command. The signal state "1" indicates the "open" command.
O_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when closing
Q	Output	BOOL	I, Q, M, D, L	Shows the status of the CMD parameter
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
ET	Static	DINT	D, L or constant	Currently elapsed time, where one clock pulse = 1 ms
PT	Static	DINT	D, L or constant	Preset timer value, where one clock pulse = 1 ms
PREV_TIME	Static	DWORD	D, L or constant	Previous system time
CMD_HIS	Static	BOOL	D, L or constant	CMD history bit

For additional information on valid data types, refer to "See also".

Example

In the following example the parameter CMD changes from "0" to "1". After the execution of the instruction the parameter Q is set to "1" and the two alarm outputs OA and CA have the signal state "0". The parameter CMD_HIS of the instance data block is set to the signal state "1" and the parameter ET is reset to "0".

Note

You can initialize static parameters in the data block.

```
STL                          Explanation
CALL DCAT, "DCAT_DB"         // The "Discrete control-timer alarm" instruction is
                             called and the instance data block "DCAT_DB" is
                             created.
CMD := "Tag_Input_CMD"       // Opening or closing of device
O_FB := "Tag_Input_O_FB"     // Feedback when opening
C_FB := "Tag_Input_C_FB"     // Feedback when closing
Q := "Tag_Output_Q"          // Shows the status of the CMD parameter
```

```
STL                          Explanation
OA := "Tag_Output_OA"        // Alarm output when opening
CA := "Tag_Output_CA"        // Alarm output when closing
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameters	Operand	Value
CMD	Tag_Input_CMD	TRUE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
Q	Tag_Output_Q	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the "DCAT_DB" instance data block of the instruction:

Parameters	Address	Value
ET	DBD4	L#12
PT	DBD8	L#222
CMD_HIS	DBX16.0	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameters	Operand	Value
Q	Tag_Output_Q	TRUE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the "DCAT_DB" instance data block of the instruction:

Parameter	Address	Value
ET	DBD4	L#0
CMD_HIS	DBX16.0	TRUE

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

MCAT: Motor control-timer alarm

Description

The instruction "Motor control-timer alarm" is used to accumulate the time from the point in time as of which one of the command inputs (open or close) is activated. The time is accumulated until the preset time is exceeded or the relevant feedback input indicates that the device has executed the requested operation within the specified time. If the preset time is exceeded before the feedback is received, the corresponding alarm is triggered.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Execution of the "Motor control-timer alarm" instruction

The following table shows the reactions of the "Motor control-timer alarm" instruction to the various input conditions:

Input parameters								Output parameters								
ET	O_HIS	C_HIS	O_CMD	C_CMD	S_CMD	O_FB	C_FB	OO	CO	OA	CA	ET	O_HIS	C_HIS	Q	Status
X	1	1	X	X	X	X	X	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	X	1	1	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	1	X	X	0	0	0	0	X	0	0	1	Stop
X	X	X	1	1	X	X	X	0	0	0	0	X	0	0	1	Stop
X	0	X	1	0	0	X	X	1	0	0	0	0	1	0	1	Start opening
<PT	1	0	X	0	0	0	X	1	0	0	0	INC	1	0	1	Open
X	1	0	X	0	0	1	0	0	0	0	0	PT	1	0	1	Opened
>=PT	1	0	X	0	0	0	X	0	0	1	0	PT	1	0	0	Opening alarm
X	X	0	0	1	0	X	X	0	1	0	0	0	0	1	1	Start closing
<PT	0	1	0	X	0	X	0	0	1	0	0	INC	0	1	1	Close
X	0	1	0	X	0	0	1	0	0	0	0	PT	0	1	1	Closed
>=PT	0	1	0	X	0	X	0	0	0	0	1	PT	0	1	0	Closing alarm
X	0	0	0	0	0	X	X	0	0	0	0	X	0	0	1	Stopped

Input parameters	Output parameters
Legend:	
INC	Add the time difference (ms) since the last processing of the FB to ET
PT	PT is set to the same value as ET
X	Cannot be used
<PT	ET < PT
>=PT	ET >= PT
If the input parameters O_HIS and C_HIS both have the signal state "1", they are immediately set to the signal state "0". In this case, the last row in the table (X) mentioned above is valid. Because it is therefore not possible to check whether the input parameters O_HIS and C_HIS have the signal state "1", the output parameters are set as follows in this case: OO = FALSE CO = FALSE OA = FALSE CA = FALSE ET = PT Q = TRUE	

Syntax

The following syntax is used for the "Motor control-timer alarm" instruction:

```
CALL MCAT, "Instance DB"
O_CMD := <Operand>
C_CMD := <Operand>
S_CMD := <Operand>
O_FB := <Operand>
C_FB := <Operand>
OO := <Operand>
CO := <Operand>
OA := <Operand>
CA := <Operand>
Q := <Operand>
```

Parameters

The following table shows the parameters of the "Motor control-timer alarm" instruction:

Parameter	Declaration	Data type	Memory area	Description
O_CMD	Input	BOOL	I, Q, M, D, L or constant	"Open" command input
C_CMD	Input	BOOL	I, Q, M, D, L or constant	"Close" command input
S_CMD	Input	BOOL	I, Q, M, D, L or constant	"Stop" command input

Parameter	Declaration	Data type	Memory area	Description
O_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when closing
OO	Output	BOOL	I, Q, M, D, L	"Open" output
CO	Output	BOOL	I, Q, M, D, L	"Close" output
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
Q	Output	BOOL	I, Q, M, D, L	The signal state "0" indicates an error condition
ET	Static	DINT	D, L or constant	Currently elapsed time, where one clock pulse = 1 ms
PT	Static	DINT	D, L or constant	Preset timer value, where one clock pulse = 1 ms
PREV_TIME	Static	DWORD	D, L or constant	Previous system time
O_HIS	Static	BOOL	D, L or constant	"Open" history bit
C_HIS	Static	BOOL	D, L or constant	"Close" history bit

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.

```
STL                              Explanation
CALL MCAT, "MCAT_DB"             // The "Motor control-timer alarm" instruction is
                                 called and the instance data block "MCAT_DB" is
                                 created.
O_CMD := "Tag_Input_O_CMD"       // "Open" command input
C_CMD := "Tag_Input_C_CMD"       // "Close" command input
S_CMD := "Tag_Input_S_CMD"       // "Stop" command input
O_FB := "Tag_Input_O_FB"         // Feedback when opening
C_FB := "Tag_Input_C_FB"         // Feedback when closing
OO := "Tag_OutputOpen"           // "Open" output
CO := "Tag_OutputClosed"         // "Close" output
OA := "Tag_Output_OA"            // Alarm output when opening
```

```
STL                          Explanation
CA := "Tag_Output_CA"        // Alarm output when closing
Q := "Tag_Output_Q"          // Error information
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameter	Operand	Value
O_CMD	Tag_Input_O_CMD	TRUE
C_CMD	Tag_Input_C_CMD	FALSE
S_CMD	Tag_Input_S_CMD	FALSE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
OO	Tag_OutputOpen	FALSE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	FALSE

The following values are saved in the "MCAT_DB" instance data block of the instruction:

Parameter	Address	Value
ET	DBD4	L#2
PT	DBD8	L#22
O_HIS	DBX16.0	TRUE
C_HIS	DBX16.1	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OO	Tag_OutputOpen	TRUE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	TRUE

The following values are saved in the "MCAT_DB" instance data block of the instruction:

Parameter	Address	Value
ET	DBD4	L#0
O_HIS	DBX16.0	TRUE
CMD_HIS	DBX16.1	FALSE

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

IMC: Compare input bits with the bits of a mask

Description

You can use the "Compare input bits with the bits of a mask" instruction to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bits of a mask. Up to 16 steps with masks can be programmed. The value of the IN_BIT0 parameter is compared with the value of the CMP_VAL[x,0] mask, where "x" indicates the step number. In the CMP_STEP parameter, you specify the step number of the mask that is used for the comparison. All programmed values are compared in the same manner. Unprogrammed input bits or unprogrammed bits of the mask have the default signal state FALSE.

If a match is found in the comparison, the signal state of the OUT parameter is set to "1". Otherwise, the OUT parameter is set to "0".

If the value of the CMP_STEP parameter is greater than 15, the instruction is not executed. An error message is output at the ERR_CODE parameter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Compare input bits with the bits of a mask" instruction:

```
CALL IMC, "Instance DB"
IN_BIT0 - 15 := <Operand>
CMP_STEP := <Operand>
```

```
OUT := <Operand>
ERR_CODE := <Operand>
```

Parameters

The following table shows the parameters of the "Compare input bits with the bits of a mask" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN_BIT0	Input	BOOL	I, Q, M, D, L or constant	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L or constant	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L or constant	Input bit 2 is compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L or constant	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L or constant	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L or constant	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L or constant	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L or constant	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L or constant	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L or constant	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L or constant	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L or constant	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L or constant	Input bit 12 is compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L or constant	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L or constant	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L or constant	Input bit 15 is compared with bit 15 of the mask.

Parameters	Declaration	Data type	Memory area	Description
CMP_STEP	Input	BYTE	I, Q, M, D, L, P or constant	The step number of the mask used for the comparison.
OUT	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that a match was found. The signal state "0" indicates that no match was found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
CMP_VAL	Static	ARRAY OF WORD	I, Q, M, D, L or constant	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

For additional information on valid data types, refer to "See also".

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000A	The value of the CMP_STEP parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

In the following example all 16 input bits are compared with the mask for step 2. The signal state of the OUT parameter is set to TRUE, since the input bits match the mask for step 2.

Note

You can initialize static parameters in the data block.

STL	Explanation
CALL IMC, "IMC_DB"	// The instruction "Compare input bits with the bits of a mask" is called and the instance data block "IMC_DB" is created.
IN_BIT0 := "Tag_Input_BIT0"	// Input bit 0
IN_BIT1 := "Tag_Input_BIT1"	// Input bit 1
IN_BIT2 := "Tag_Input_BIT2"	// Input bit 2

```
STL                              Explanation
IN_BIT3 := "Tag_Input_BIT3"    // Input bit 3
IN_BIT4 := "Tag_Input_BIT4"    // Input bit 4
IN_BIT5 := "Tag_Input_BIT5"    // Input bit 5
IN_BIT6 := "Tag_Input_BIT6"    // Input bit 6
IN_BIT7 := "Tag_Input_BIT7"    // Input bit 7
IN_BIT8 := "Tag_Input_BIT8"    // Input bit 8
IN_BIT9 := "Tag_Input_BIT9"    // Input bit 9
IN_BIT10 := "Tag_Input_BIT10"  // Input bit 10
IN_BIT11 := "Tag_Input_BIT11"  // Input bit 11
IN_BIT12 := "Tag_Input_BIT12"  // Input bit 12
IN_BIT13 := "Tag_Input_BIT13"  // Input bit 13
IN_BIT14 := "Tag_Input_BIT14"  // Input bit 14
IN_BIT15 := "Tag_Input_BIT15"  // Input bit 15
CMP_STEP := "Tag_CMP_STEP"     // Step number of the mask
OUT := "Tag_Output"            // Indicates whether or not a match was found.
ERR_CODE := "Tag_ErrorCode"    // Error information
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameters	Operand	Value
IN_BIT0	Tag_Input_BIT0	TRUE
IN_BIT1	Tag_Input_BIT1	TRUE
IN_BIT2	Tag_Input_BIT2	FALSE
IN_BIT3	Tag_Input_BIT3	TRUE
IN_BIT4	Tag_Input_BIT4	TRUE
IN_BIT5	Tag_Input_BIT5	FALSE
IN_BIT6	Tag_Input_BIT6	TRUE
IN_BIT7	Tag_Input_BIT7	TRUE
IN_BIT8	Tag_Input_BIT8	FALSE
IN_BIT9	Tag_Input_BIT9	TRUE
IN_BIT10	Tag_Input_BIT10	TRUE
IN_BIT11	Tag_Input_BIT11	FALSE
IN_BIT12	Tag_Input_BIT12	TRUE
IN_BIT13	Tag_Input_BIT13	TRUE
IN_BIT14	Tag_Input_BIT14	FALSE
IN_BIT15	Tag_Input_BIT15	TRUE
CMP_STEP	Tag_CMP_STEP	B#16#02
OUT	Tag_Output	FALSE
ERR_CODE	Tag_ErrorCode	W#16#0000

The following values for the mask of step 2 are saved in the "IMC_DB" instance data block of the instruction:

Parameter	Address	Value
CMP_VAL [2,0]	DBX12.0	TRUE
CMP_VAL [2,1]	DBX12.1	TRUE
CMP_VAL [2,2]	DBX12.2	FALSE
CMP_VAL [2,3]	DBX12.3	TRUE
CMP_VAL [2,4]	DBX12.4	TRUE
CMP_VAL [2,5]	DBX12.5	FALSE
CMP_VAL [2,6]	DBX12.6	TRUE
CMP_VAL [2,7]	DBX12.7	TRUE
CMP_VAL [2,8]	DBX13.0	FALSE
CMP_VAL [2,0]	DBX13.1	TRUE
CMP_VAL [2,10]	DBX13.2	TRUE
CMP_VAL [2,11]	DBX13.3	FALSE
CMP_VAL [2,12]	DBX13.4	TRUE
CMP_VAL [2,13]	DBX13.5	TRUE
CMP_VAL [2,14]	DBX13.6	FALSE
CMP_VAL [2,15]	DBX13.7	TRUE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OUT	Tag_Output	TRUE
ERR_CODE	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Switching display formats in the program status (Page 2980)

SMC: Compare scan matrix

Description

You can use the "Compare scan matrix" instruction to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bit of the comparison masks for each step. Processing starts at step 1 and is continued until the last programmed step (LAST) or until a match is found. The input bit of the IN_BIT0 parameter is compared with the value of the CMP_VAL[x,0] mask, where "x" indicates the step number. All programmed values are compared in the same manner. If a match is found the signal state of the OUT parameter is set to "1" and the step number with the matching mask is written in the OUT_STEP parameter. Unprogrammed input bits or unprogrammed bits of the mask have the default signal state FALSE. If more than one step has a matching mask, only the first one found is indicated in the OUT_STEP parameter. If no match is found, the signal state of the OUT parameter is set to "0". In this case the value at the OUT_STEP parameter is greater by "1" than the value of the LAST parameter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Compare scan matrix" instruction:

```
CALL SMC, "Instance DB"
IN_BIT0 - 15 := <Operand>
OUT := <Operand>
OUT_STEP := <Operand>
ERR_CODE := <Operand>
```

Parameters

The following table shows the parameters of the "Compare scan matrix" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN_BIT0	Input	BOOL	I, Q, M, D, L or constant	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L or constant	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L or constant	Input bit 2 is compared with bit 2 of the mask.

Parameters	Declaration	Data type	Memory area	Description
IN_BIT3	Input	BOOL	I, Q, M, D, L or constant	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L or constant	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L or constant	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L or constant	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L or constant	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L or constant	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L or constant	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L or constant	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L or constant	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L or constant	Input bit 12 is compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L or constant	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L or constant	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L or constant	Input bit 15 is compared with bit 15 of the mask.
OUT	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that a match was found. The signal state "0" indicates that no match was found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information

Parameters	Declaration	Data type	Memory area	Description
OUT_STEP	Output	BYTE	I, Q, M, D, L, P	Contains the step number with the matching mask or the step number which is greater by "1" than the value of the LAST parameter, provided no match is found.
LAST	Static	BYTE	I, Q, M, D, L, P or constant	Specifies the step number of the last step to be scanned for a matching mask.
CMP_VAL	Static	ARRAY OF WORD	I, Q, M, D, L or constant	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

For additional information on valid data types, refer to "See also".

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000E	The value of the LAST parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

In this example, all 16 input bits are compared with the mask for steps 0 through 5 until a match is found. Only the masks for steps 0 through 2 are scanned, since the mask for step 2 matches the input bit.

Note

You can initialize static parameters in the data block.

```
STL                              Explanation
CALL SMC, "SMC_DB"               // The "Compare scan matrix" instruction is called and
                                 the instance data block "SMC_DB" is created.
IN_BIT0 := "Tag_Input_BIT0"      // Input bit 0
IN_BIT1 := "Tag_Input_BIT1"      // Input bit 1
IN_BIT2 := "Tag_Input_BIT2"      // Input bit 2
IN_BIT3 := "Tag_Input_BIT3"      // Input bit 3
IN_BIT4 := "Tag_Input_BIT4"      // Input bit 4
IN_BIT5 := "Tag_Input_BIT5"      // Input bit 5
IN_BIT6 := "Tag_Input_BIT6"      // Input bit 6
IN_BIT7 := "Tag_Input_BIT7"      // Input bit 7
IN_BIT8 := "Tag_Input_BIT8"      // Input bit 8
IN_BIT9 := "Tag_Input_BIT9"      // Input bit 9
IN_BIT10 := "Tag_Input_BIT10"    // Input bit 10
IN_BIT11 := "Tag_Input_BIT11"    // Input bit 11
IN_BIT12 := "Tag_Input_BIT12"    // Input bit 12
IN_BIT13 := "Tag_Input_BIT13"    // Input bit 13
IN_BIT14 := "Tag_Input_BIT14"    // Input bit 14
IN_BIT15 := "Tag_Input_BIT15"    // Input bit 15
OUT := "Tag_Output"              // Indicates whether or not a match was found.
OUT_STEP := "Tag_Output_STEP"    // Contains the step number with the relevant mask.
ERR_CODE := "Tag_ErrorCode"      // Error information
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameters	Operand	Value
IN_BIT0	Tag_Input_BIT0	TRUE
IN_BIT1	Tag_Input_BIT1	TRUE
IN_BIT2	Tag_Input_BIT2	FALSE
IN_BIT3	Tag_Input_BIT3	TRUE
IN_BIT4	Tag_Input_BIT4	TRUE
IN_BIT5	Tag_Input_BIT5	FALSE
IN_BIT6	Tag_Input_BIT6	TRUE
IN_BIT7	Tag_Input_BIT7	TRUE
IN_BIT8	Tag_Input_BIT8	FALSE
IN_BIT9	Tag_Input_BIT9	TRUE
IN_BIT10	Tag_Input_BIT10	TRUE
IN_BIT11	Tag_Input_BIT11	FALSE
IN_BIT12	Tag_Input_BIT12	TRUE
IN_BIT13	Tag_Input_BIT13	TRUE
IN_BIT14	Tag_Input_BIT14	FALSE

Parameters	Operand	Value
IN_BIT15	Tag_Input_BIT15	TRUE
OUT	Tag_Output	FALSE
OUT_STEP	Tag_Output_STEP	B#16#00
ERR_CODE	Tag_ErrorCode	W#16#0000

The following values for the mask of step 2 are saved in the "SMC_DB" instance data block of the instruction:

Parameters	Address	Value
CMP_VAL [2,0]	DBX12.0	TRUE
CMP_VAL [2,1]	DBX12.1	TRUE
CMP_VAL [2,2]	DBX12.2	FALSE
CMP_VAL [2,3]	DBX12.3	TRUE
CMP_VAL [2,4]	DBX12.4	TRUE
CMP_VAL [2,5]	DBX12.5	FALSE
CMP_VAL [2,6]	DBX12.6	TRUE
CMP_VAL [2,7]	DBX12.7	TRUE
CMP_VAL [2,8]	DBX13.0	FALSE
CMP_VAL [2,0]	DBX13.1	TRUE
CMP_VAL [2,10]	DBX13.2	TRUE
CMP_VAL [2,11]	DBX13.3	FALSE
CMP_VAL [2,12]	DBX13.4	TRUE
CMP_VAL [2,13]	DBX13.5	TRUE
CMP_VAL [2,14]	DBX13.6	FALSE
CMP_VAL [2,15]	DBX13.7	TRUE
LAST	DB84	B#16#05

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OUT	Tag_Output	TRUE
OUT_STEP'	Tag_Output_STEP	B#16#02
ERR_CODE	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Switching display formats in the program status (Page 2980)

LEAD_LAG: Lead and lag algorithm

Description

You can use the "Lead and lag algorithm" instruction to process signals with an analog tag. The gain value at the GAIN parameter must be greater than zero. The result of the "Lead and lag algorithm" instruction is calculated using the following equation:

$$OUT = \left[\frac{LG_TIME}{LG_TIME + SAMPLE_T}\right] PREV_OUT + GAIN \left[\frac{LD_TIME + SAMPLE_T}{LG_TIME + SAMPLE_T}\right] IN - GAIN \left[\frac{LD_TIME}{LG_TIME + SAMPLE_T}\right] * PREV_IN$$

The instruction "Lead and lag algorithm" supplies plausible results only when processing is in fixed program cycles. The same units must be specified at the parameters LD_TIME, LG_TIME and SAMPLE_T. At LG_TIME > 4 + SAMPLE_T, the instruction approaches the following function:

OUT = GAIN * ((1 + LD_TIME * s) / (1 + LG_TIME * s)) * IN

If the value of the GAIN parameter is less than or equal to zero, the calculation is not performed and an error information is output at the ERR_CODE parameter.

You can use the "Lead and lag algorithm" instruction in conjunction with loops as a compensator in dynamic feed-forward control. The instruction consists of two operations. The "Lead" operation shifts the phase of the OUT output so that the output leads the input. The "Lag" operation, on the other hand, shifts the output so that the output lags behind the input. Because the "Lag" operation is equivalent to an integration, it can be used as a noise suppressor or as a low-pass filter. The "Lead" operation is equivalent to a differentiation and can therefore be used as a high-pass filter. The two operations together (Lead and Lag) result in the output phase lagging behind the input at lower frequencies and leading it at higher frequencies. This means that the "Lead and lag algorithm" instruction can be used as a band pass filter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Lead and lag algorithm" instruction:

```
CALL LEAD_LAG, "Instance
DB"
IN := <Operand>
```

```
SAMPLE_T := <Operand>
OUT := <Operand>
ERR_CODE := <Operand>
```

Parameters

The following table shows the parameters of the "Lead and lag algorithm" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	REAL	I, Q, M, D, L, P or constant	The input value of the current sample time (cycle time) to be processed
SAMPLE_T	Input	INT	I, Q, M, D, L, P or constant	Sample time
OUT	Output	REAL	I, Q, M, D, L, P	Result of the instruction
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
LD_TIME	Static	REAL	I, Q, M, D, L, P or constant	Lead time in the same unit as sample time.
LG_TIME	Static	REAL	I, Q, M, D, L, P or constant	Lag time in the same unit as sample time
GAIN	Static	REAL	I, Q, M, D, L, P or constant	Gain as % / % (the ratio of the change in output to a change in input as a steady state).
PREV_IN	Static	REAL	I, Q, M, D, L, P or constant	Previous input
PREV_OUT	Static	REAL	I, Q, M, D, L, P or constant	Previous output

For additional information on valid data types, refer to "See also".

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
0009	The value of the GAIN parameter is less than or equal to zero.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.

```
STL                              Explanation
CALL LEAD_LAG, "LEAD_LAG_DB"     // The "Lead and lag algorithm" instruction is called
                                 and the instance data block "LEAD_LAG_DB" is created.
IN := "Tag_Input"                // The input value of the current sample time (cycle
                                 time) to be processed.
SAMPLE_T :=                      // Sample time
"Tag_Input_SAMPE_T"
OUT := "Tag_Output_Result"       // Result of the instruction
ERR_CODE := "Tag_ErrorCode"      // Error information
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input parameters:

Parameters	Operand	Value
IN	Tag_Input	2.0
SAMPLE_T	Tag_InputSampleTime	10

The following values are saved in the "LEAD_LAG_DB" instance data block of the instruction:

Parameters	Address	Value
LD_TIME	DBD12	2.0
LG_TIME	DBD16	2.0
GAIN	DBD20	1.0
PREV_IN	DBD24	6.0
PREV_OUT	DBD28	6.0

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameters	Operand	Value
OUT	Tag_Output_Result	2.0

The following values are saved in the "LEAD_LAD_DB" instance data block of the instruction:

Parameters	Operand	Value
PREV_IN	DBD24	2.0
PREV_OUT	DBD28	2.0

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Switching display formats in the program status (Page 2980)

SEG: Create bit pattern for seven-segment display

Description

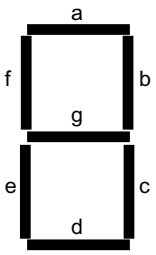
The "Create bit pattern for seven-segment display" instruction converts each of the four hexadecimal digits in the specified source word (IN) into four equivalent codes for a 7-segment display and writes these to the double word of the output (OUT).

The "Create bit pattern for seven-segment display" does not recognize any error conditions.

The relationship between the input hexadecimal values and the output bit patterns is as follows:

Digit	-gfedcba	Display
0000	00111111	0
0001	00000110	1
0010	01011011	2
0011	01001111	3
0100	01100110	4
0101	01101101	5
0110	01111101	6
0111	00000111	7
1000	01111111	8
1001	01100111	9
1010	01110111	A
1011	01111100	B
1100	00111001	C
1101	01011110	D

Seven-segment display



1110	01111001	E
1111	01110001	F

Syntax

The following syntax is used for the "Create bit pattern for seven-segment display" instruction:

```
CALL SEG
IN := <Operand>
OUT := <Operand>
```

Parameter

The following table shows the parameters of the "Create bit pattern for seven-segment display" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	IN	WORD	I, Q, M, D, L, P or constant	Source word with four hexadecimal digits.
OUT	OUT	DWORD	I, Q, M, D, L, P	Destination bit pattern with four bytes.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
CALL SEG	// The "Create bit pattern for seven-segment display" instruction is called.
IN := "Tag_Input"	// Source word
OUT := "Tag_Output"	// Bit pattern for the seven-segment display

The following table shows how the instruction functions using specific values:

Parameter	Operand	Value	
Hexadecimal	Binary		
IN	Tag_Input	W#16#1234	0001 0010 0011 0100
OUT	Tag_Output	DW#16#065B4F66	00000110 01011011 01001111 01100110 Display: 1234

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

BCDCPL: Create tens complement

Description

The "Create tens complement" instruction is used to create the tens complement of a seven-digit BCD number specified by the IN parameter. This instruction uses the following mathematical formula to calculate:

10000000 (as BCD)

– 7-digit BCD value

Tens complement (as BCD)

Syntax

The following syntax is used for the "Create tens complement" instruction:

```
CALL BCDCPL
IN := <Operand>
ERR_CODE := <Operand>
```

Parameters

The following table shows the parameters of the "Create tens complement" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	Bit strings	I, Q, M, D, L, P or constant	7-digit BCD number
ERR_CODE	Output	DWORD	I, Q, M, D, L, P	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL BCDCPL                  // The "Create tens complement" instruction is called.
IN := "Tag_Input"            // 7-digit BCD number
ERR_CODE := "Tag_Output"     // Tens complement of the specified BCD number
```

The following table shows how the instruction functions using specific values:

Parameter	Operand	Value*
IN	Tag_Input	DW#16#01234567
ERR_CODE	Tag_Output	DW#16#08765433
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".		

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Switching display formats in the program status (Page 2980)

BITSUM: Count number of set bits

Description

The "Count number of set bits" instruction is used to count the number of bits of an operand that is set to the signal state "1". The operand whose bits are to be counted is specified on the IN parameter. The result of the instruction is output at the RET_VAL parameter.

Syntax

The following syntax is used for the "Count number of set bits" instruction:

```
CALL BITSUM
IN := <Operand>
RET_VAL := <Operand>
```

Parameters

The following table shows the parameters of the "Count number of set bits" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	DWORD	I, Q, M, D, L, P or constant	Operand whose set bits are counted
RET_VAL	Output	INT	I, Q, M, D, L, P	Number of set bits

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL BITSUM                  // The "Count number of set bits" instruction is
                             called.
IN := "Tag_Input"            // Operand whose set bits are counted.
RET_VAL := "Tag_Output"      // Number of set bits of the operand "Tag_Input".
```

The following table shows how the instruction functions using specific values:

Parameter	Operand	Value*
IN	Tag_Input	DW#16#12345678
RET_VAL	Tag_Output	W#16#000D (13 bits)
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".		

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Switching display formats in the program status (Page 2980)

Basic instructions

Bit logic operations

A: AND logic operation

Description

Use the instruction "AND logic operation" to query whether the signal state of a binary operand is "1" and to AND the query result with the signal state of the result of logic operation (RLO). The query result is hereby identical with the signal state of the checked operand.

After execution of the instruction the RLO will be "1" if both linked signal states are also "1". If one of the linked signal states is "0", the RLO is "0" after executing the instruction.

If you use the instruction multiple times consecutively, all query results must be "1" for the common result of the logic operation to also be "1".

You can also use the instruction "AND logic operation" to query the signal state of the status word. To do this, specify the relevant status bit (==0, <>0, >0, <0, >=0, <=0, OV, OS, UO, BR) as parameter of the instruction.

Syntax

Use the following syntax for the "AND logic operation" instruction:

```
A <operand>
```

Parameters

The following table shows the parameters of the "AND logic operation" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, D, L, T, C	Operand whose signal state is queried.

Example

The following example shows how the instruction works:

STL program	Relay circuit diagram
	Power rail
A “Tag_Input_1“	Signal state 1: Normally open contact
A “Tag_Input_2“	Signal state 1: Normally open contact
= “Tag_Output“	Signal state 1: Coil
Shows closed switch	

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

AN: Negated AND logic operation

Description

Use the instruction "Negated AND logic operation" to query whether the signal state of a binary operand is "0" and to AND the query result with the signal state of the result of logic operation (RLO). The result of the query is hereby the negated signal state of the checked operand. If the operand has signal state "0," the result of the query is "1". If the operand has signal state "1," the result of the query is "0".

After the instruction executes, the RLO is "1" if the result of the query and the RLO before the execution of the instruction have signal state "1".

After the instruction executes, the RLO is "0" if the result of the query or the RLO before the execution of the instruction have signal state "0".

You can also use the instruction "Negated AND logic operation" to query the signal state of the status word. To do this, specify the relevant status bit (==0, <>0, >0, <0, >=0, <=0, OV, OS, UO, BR) as parameter of the instruction.

Syntax

Use the following syntax for the "Negated AND logic operation" instruction:

```
AN <operand>
```

Parameters

The following table shows the parameters of the "Negated AND logic operation" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, D, L, T, C	Operand whose signal state is queried.

Example

The following example shows how the instruction works:

STL program		Relay circuit diagram	
		Power rail	
A	“Tag_Input_1“	Signal state 0:	Normally open contact
AN	“Tag_Input_2“	Signal state 1:	Normally closed contact
=	“Tag_Output“	Signal state 0:	Coil

See also

Overview of the valid data types (Page 2236)
Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)
Evaluating errors with output parameter RET_VAL (Page 5370)
Call function blocks as single or multi-instances (Page 2182)
Basic information on the status word (Page 2466)
Querying and setting status bits in STL (Page 2467)

O: OR logic operation

Description

Use the instruction "OR logic operation" to query whether the signal state of a binary operand is "1" and to OR the query result with the signal state of the result of logic operation (RLO). The query result is hereby identical with the signal state of the checked operand.

After the instruction executes, the RLO is "1" if one of the linked signal states is "1". If both linked signal states are "0", the RLO is "0" after the instruction executes.

If you use the instruction multiple times consecutively, it is sufficient that one of the results of the query is "1" for the common result of the logic operation to also be "1".

You can also use the instruction "OR logic operation" to query the signal state of the status word. To do this, specify the relevant status bit (==0, <>0, >0, <0, >=0, <=0, OV, OS, UO, BR) as parameter of the instruction.

Syntax

Use the following syntax for the "OR logic operation" instruction:

```
O <Operand>
```

Parameters

The following table shows the parameters of the "OR logic operation" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, D, L, T, C	Operand whose signal state is queried.

Example

The following example shows how the instruction works:

STL program	Relay circuit diagram
	Power rail
O “Tag_Input_1“	Signal state 1 normally open contact (“Tag_Input_1“) / Signal state 0 normally open contact (“Tag_Input_2“)
O “Tag_Input_2“	
= “Tag_Output“	Signal state 1: Coil
	Shows closed switch

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

ON: Negated OR logic operation

Description

Use the instruction "Negated OR logic operation" to query whether the signal state of a binary operand is "0" and to OR the query result with the signal state of the result of logic operation (RLO). The result of the query is hereby the negated signal state of the checked operand. If the operand has signal state "0," the result of the query is "1". If the operand has signal state "1," the result of the query is "0".

After the instruction executes, the RLO is "1" if the result of the query or the RLO before the execution of the instruction have signal state "1". If both linked signal states are "0", the RLO is also "0" after the instruction executes.

If you use the instruction multiple times consecutively, it is sufficient that one of the results of the query is "1" for the common result of the logic operation to also be "1".

You can also use the instruction "Negated OR logic operation" to query the signal state of the status word. To do this, specify the relevant status bit (==0, <>0, >0, <0, >=0, <=0, OV, OS, UO, BR) as parameter of the instruction.

Syntax

Use the following syntax for the "Negated OR logic operation" instruction:

```
ON <Operand>
```

Parameters

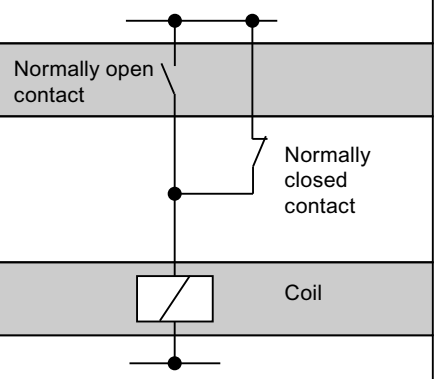
The following table shows the parameters of the "Negated OR logic operation" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, D, L, T, C	Operand whose signal state is checked

Example

The following example shows how the instruction works:

STL program	Relay circuit diagram
	Power rail
O "Tag_Input_1"	Signal state 0
ON "Tag_Input_2"	Signal state 1
= "Tag_Output"	Signal state 1



See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

X: EXCLUSIVE OR logic operation

Description

Use the instruction "EXCLUSIVE OR logic operation" to query whether the signal state of a binary operand is "1" and to XOR the query result with the signal state of the result of logic operation (RLO). The query result is hereby identical with the signal state of the checked operand. After the instruction EXECUTES, the RLO is "1" if the linked signal states are not equal. The RLO is "0" if both signal states are the same.

If, for example, the RLO before execution of the instruction is "1" and the queried operand is "1", the RLO is reset to "0". If the RLO is "1" and the operand has signal state "0", the RLO is set to "1" after the instruction executes.

If you use the instruction several times consecutively, it is sufficient that one of the query results is "1" for the common RLO to be "1" as well. If all query results are "1" or "0" the RLO is reset to "0".

You can also use the instruction "EXCLUSIVE OR logic operation" to query the signal state of the status word. To do this, specify the relevant status bit (==0, <>0, >0, <0, >=0, <=0, OV, OS, UO, BR) as parameter of the instruction.

Syntax

Use the following syntax for the "EXCLUSIVE OR logic operation" instruction:

```
X <Operand>
```

Parameters

The following table shows the parameters of the "EXCLUSIVE OR logic operation" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, D, L, T, C	Operand whose signal state is queried.

Example

The following example shows how the instruction works:

STL program	Relay circuit diagram
	Power rail
X “Tag_Input_1“	Contact
X “Tag_Input_2“	Contact
= “Tag_Output“	Coil

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

XN: Negated EXCLUSIVE OR logic operation

Description

Use the "Negated EXCLUSIVE OR logic operation" instruction to query whether the signal state of a binary operand is "0" and to OR the query result with the signal state of the result of logic operation (RLO). The result of the query is hereby the negated signal state of the checked operand. After the instruction EXECUTES, the RLO is "1" if the linked signal states are not equal. The RLO is "0" if both signal states are the same.

If, for example, the queried operand has signal state "0," the result of the query is "1". XORing the query result with an RLO "1" results in signal state "0". The logic operation with an RLO "0" results in signal state "1".

You can also use the "Negated EXCLUSIVE OR logic operation" instruction to query the signal state of the status word. To do this, specify the relevant status bit (==0, <>0, >0, <0, >=0, <=0, OV, OS, UO, BR) as parameter of the instruction.

Syntax

Use the following syntax for the "Negated EXCLUSIVE OR logic operation" instruction:

```
XN <Operand>
```

Parameters

The following table shows the parameters of the "Negated EXCLUSIVE OR logic operation" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, D, L, T, C	Operand whose signal state is queried.

Example

The following example shows how the instruction works:

STL program		Relay circuit diagram	
		Power rail	
X	“Tag_Input_1“	Contact	
XN	“Tag_Input_2“	Contact	
=	“Tag_Output“	Coil	

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

O: OR logic operation of AND functions

Description

Use the instruction "OR logic operation of AND functions" to execute several AND logic operations before an OR logic operation. The overall result of the AND logic operations is saved temporarily and ORed with further signal queries.

Syntax

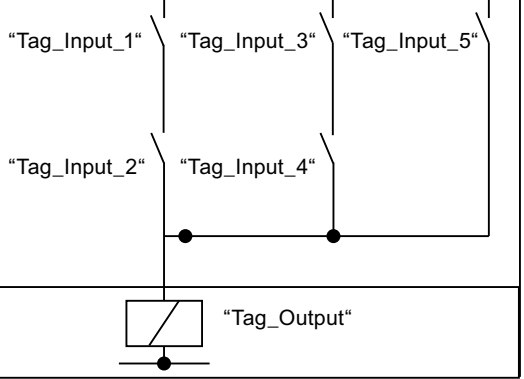
Use the following syntax for the "OR logic operation of AND functions" instruction:

```
O
```

No brackets are needed for the syntax of this instruction.

Example

The following example shows how the instruction works:

STL program	Relay circuit diagram
	Power rail
A “Tag_Input_1“ A “Tag_Input_2“ O A “Tag_Input_3“ A “Tag_Input_4“ O “Tag_Input_5“ = “Tag_Output“	

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

A(: AND logic operation with branch

Description

Use the instruction "AND logic operation with branch" to execute the instructions of a parenthesized expression before an AND logic operation.

When the instruction executes, the CPU saves the current result of logic operation (RLO), as well as the binary result BR. The saved RLO is ANDed with the result of logic operation of the complete nested expression. If additional signal queries follow after the nested expression, these are also ANDed.

Parenthesized expressions can also be nested by programming an instruction within the parenthesized expression that also contains a parenthesized expression. The nesting depth is hereby limited to 7 levels.

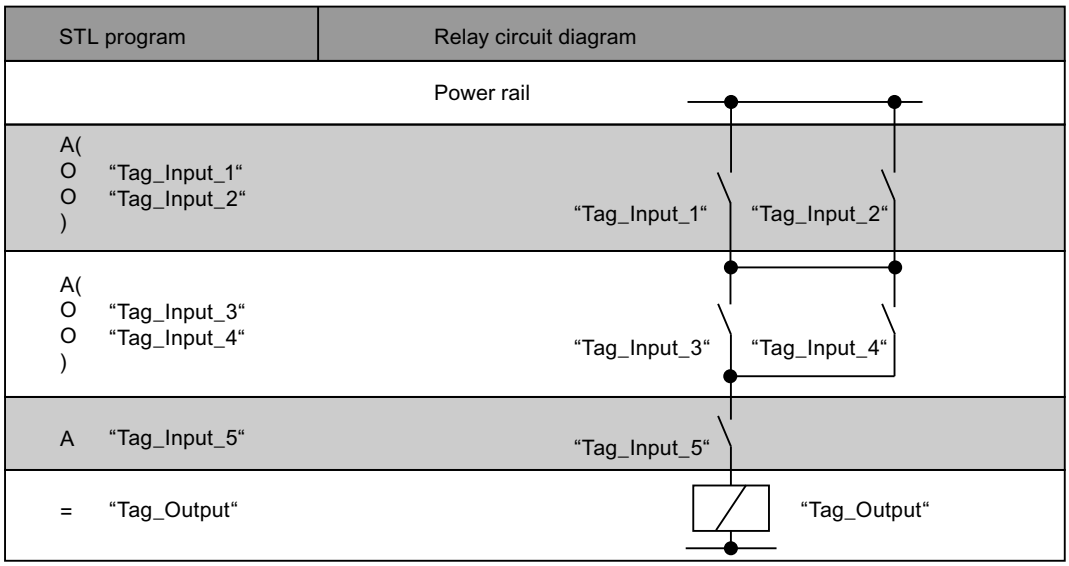
Syntax

Use the following syntax for the instruction "AND logic operation with branch":

```
A (
```

Example

The following example shows how the instruction works:



See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

AN(: Negated AND logic operation with branch

Description

Use the "Negated AND logic operation with branch" instruction to execute the instructions of a parenthesized expression before an AND logic operation and to negate the complete result of the parenthesized expression.

When the instruction executes, the CPU saves the current result of logic operation (RLO), as well as the binary result BR. After the instructions in the parenthesized expression have executed, the RLO of the complete parenthesized expression is negated and ANDed with the saved RLO. If additional signal queries follow after the parenthesized expression, these are also ANDed.

Parenthesized expressions can also be nested by programming an instruction within the parenthesized expression that also contains a parenthesized expression. The nesting depth is hereby limited to 7 levels.

Syntax

Use the following syntax for the instruction "Negated AND logic operation with branch":

```
AN(
```

Example

The following example shows how the instruction works:

STL	Explanation
AN(	// Start of the nested expression
	// AND the RLO of the parenthesized expression and negate the complete result
O "Tag_Input_1"	// Query the signal state of the operand for "1" and OR with the current RLO.
O "Tag_Input_2"	// Query the signal state of the operand for "1" and OR with the current RLO.
)	// End of the nested expression
AN "Tag_Input_3"	// Query the signal state of the operand for "0" and AND with the current RLO.
= "Tag_Output"	// Assign signal state of the RLO to the operand.

See also

Overview of the valid data types (Page 2236)
Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)
Evaluating errors with output parameter RET_VAL (Page 5370)
Call function blocks as single or multi-instances (Page 2182)
Basic information on the status word (Page 2466)
Querying and setting status bits in STL (Page 2467)

O(: OR logic operation with branch

Description

Use the instruction "OR logic operation with branch" to execute the instructions of a parenthesized expression before an OR logic operation.

When the instruction executes, the CPU saves the current result of logic operation (RLO), as well as the binary result BR. After the instructions contained in the parenthesized expression have executed, the saved RLO is ORed with the RLO of the complete parenthesized expression. If additional signal queries follow after the parenthesized expression, these are also ORed.

Parenthesized expressions can also be nested by programming an instruction within the parenthesized expression that also contains a parenthesized expression. The nesting depth is hereby limited to 7 levels.

Syntax

Use the following syntax for the "OR logic operation with branch" instruction:

```
O(
```

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input_1"	// Query the signal state of the operand for "1" and AND with the current RLO.
O(	// Start of the nested expression
	// OR the RLO of the parenthesized expression.
A "Tag_Input_2"	// Query the signal state of the operand for "1" and AND with the current RLO.
A "Tag_Input_3"	// Query the signal state of the operand for "1" and AND with the current RLO.

STL	Explanation
)	// End of the nested expression
= "Tag_Output"	// Assign signal state of the RLO to the operand.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

ON(: Negated OR logic operation with branch

Description

Use the "Negated OR logic operation with branch" instruction to execute the instructions of a parenthesized expression before an OR logic operation and to negate the complete result of the parenthesized expression.

When the instruction executes, the CPU saves the current result of logic operation (RLO), as well as the binary result BR. After the instructions in the parenthesized expression have executed, the RLO of the complete parenthesized expression is negated and ORed with the saved RLO. If additional signal queries follow after the parenthesized expression, these are also ORed.

Parenthesized expressions can also be nested by programming an instruction within the parenthesized expression that also contains a parenthesized expression. The nesting depth is hereby limited to 7 levels.

Syntax

Use the following syntax for the instruction "Negated OR logic operation with branch":

```
ON (
```

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input_1"	// Query the signal state of the operand for "1" and AND with the current RLO.

STL	Explanation
ON(	// Start of the nested expression
	// OR the RLO of the parenthesized expression and negate the complete result
A "Tag_Input_2"	// Query the signal state of the operand for "1" and AND with the current RLO.
A "Tag_Input_3"	// Query the signal state of the operand for "1" and AND with the current RLO.
)	// End of the nested expression
= "Tag_Output"	// Assign signal state of the RLO to the operand.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

X(: EXCLUSIVE OR logic operation with branch

Description

Use the instruction "EXCLUSIVE OR logic operation with branch" to execute the instructions of a parenthesized expression before an EXCLUSIVE OR logic operation.

When the instruction executes, the CPU saves the current result of logic operation (RLO), as well as the binary result BR. After the instructions in the parenthesized expression have executed, the saved RLO is XORed with the RLO of the complete parenthesized expression. If additional signal queries follow after the parenthesized expression, these are also XORed.

Parenthesized expressions can also be nested by programming an instruction within the parenthesized expression that also contains a parenthesized expression. The nesting depth is hereby limited to 7 levels.

Syntax

Use the following syntax for the "EXCLUSIVE OR logic operation with branch" instruction:

```
X(
```

Example

The following example shows how the instruction works:

STL	Explanation
X(	// Start of the nested expression
	// EXCLUSIVE OR the RLO of the parenthesized expression.
A "Tag_Input_1"	// Query the signal state of the operand for "1" and AND with the current RLO.
A "Tag_Input_2"	// Query the signal state of the operand for "1" and AND with the current RLO.
)	// End of the nested expression
X "Tag_Input_3"	// Query the signal state of the operand for "1" and AND with the current RLO.
= "Tag_Output"	// Assign signal state of the RLO to the operand.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

XN(: Negated EXCLUSIVE OR logic operation with branch

Description

Use the "Negated EXCLUSIVE OR logic operation with branch" instruction to execute the instructions of a parenthesized expression before an EXCLUSIVE OR logic operation and to negate the complete result of the parenthesized expression.

When the instruction executes, the CPU saves the current result of logic operation (RLO), as well as the binary result BR. After the instructions contained in the parenthesized expression have been executed, the RLO of the complete parenthesized expression will be negated and EXCLUSIVE ORed with the saved RLO. If additional signal queries follow after the parenthesized expression, these are also XORed.

Parenthesized expressions can also be nested by programming an instruction within the parenthesized expression that also contains a parenthesized expression. The nesting depth is hereby limited to 7 levels.

Syntax

Use the following syntax for the instruction "Negated EXCLUSIVE OR logic operation with branch":

```
XN(
```

Example

The following example shows how the instruction works:

STL	Explanation
XN(	// Start of the nested expression
	// EXCLUSIVE OR the RLO of the parenthesized expression and negate the complete result
A "Tag_Input_1"	// Query the signal state of the operand for "1" and AND with the current RLO.
A "Tag_Input_2"	// Query the signal state of the operand for "1" and AND with the current RLO.
)	// End of the nested expression
XN "Tag_Input_3"	// Query the signal state of the operand for "1" and AND with the current RLO.
= "Tag_Output"	// Assign signal state of the RLO to the operand.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

): Close branch

Description

Use the "Close branch" instruction to specify the end of a parenthesized expression. After processing of the instruction the complete result of the nested expression is linked according to the instruction that was used to open the nested expression.

Use the following instructions to open a nested expression:

- A(: AND logic operation with branch
- AN(: Negated AND logic operation with branch

- O(: OR logic operation with branch
- ON(: Negated OR logic operation with branch
- X(: EXCLUSIVE OR logic operation with branch
- XN(: Negated EXCLUSIVE OR logic operation with branch

After the "Close branch" instruction, the CPU sets the binary result BR to the signal state that it had before the parenthesized expression.

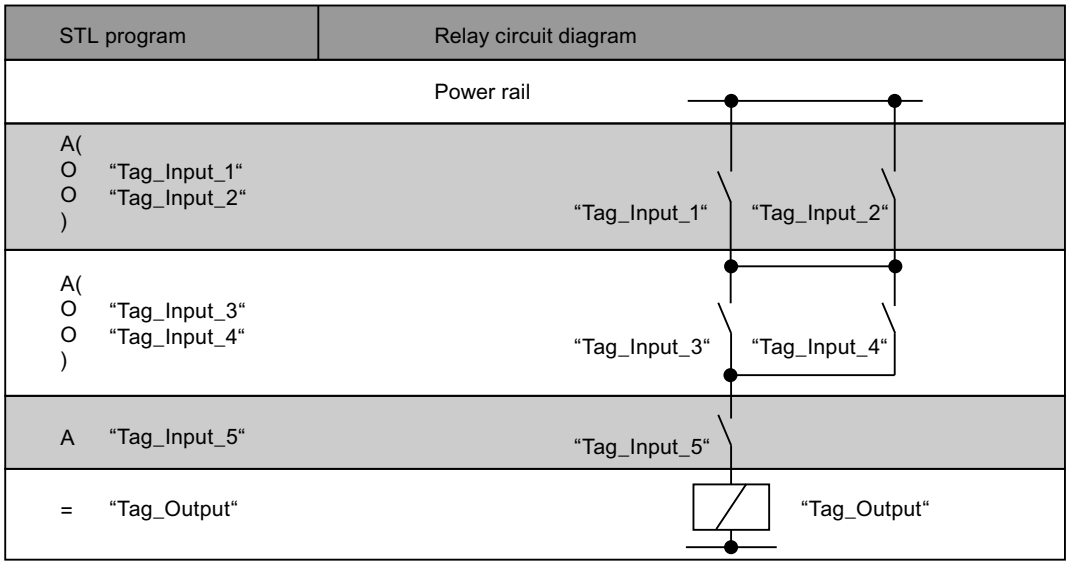
Syntax

Use the following syntax for the "Close branch" instruction:

```
)
```

Example

The following example shows how the instruction works:



See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

=: Assignment

Description

Use the "Assignment" instruction to assign the signal state of a result of logic operation (RLO) saved in the CPU to a specified operand. If the RLO has signal state "1", the operand is set. If the signal is "0", the operand is reset to "0".

Syntax

Use the following syntax for the "Assignment" instruction:

```
= <Operand>
```

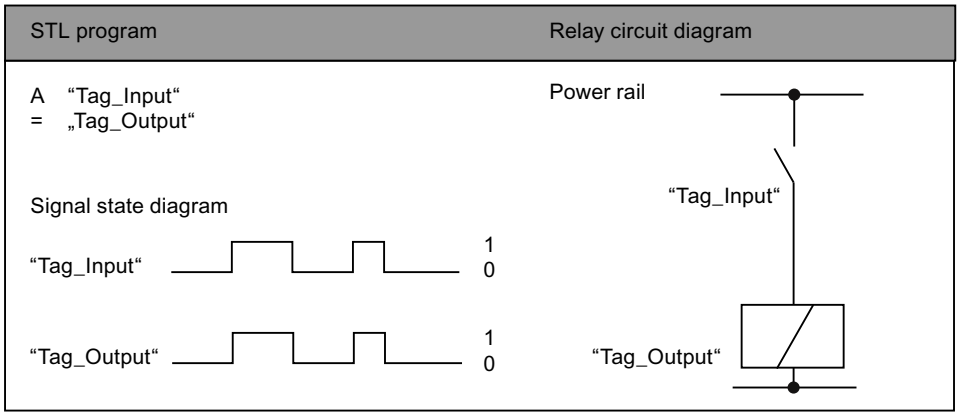
Parameters

The following table shows the parameters of the "Assignment" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand to which the current RLO is assigned.

Example

The following example shows how the instruction works:



See also

Overview of the valid data types (Page 2236)
Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)
Evaluating errors with output parameter RET_VAL (Page 5370)
Call function blocks as single or multi-instances (Page 2182)
Basic information on the status word (Page 2466)
Querying and setting status bits in STL (Page 2467)

R: Reset

Description

Use the "Reset" instruction to reset the signal state of a specified operand to "0".

The instruction is only executed when the current result of logic operation (RLO) is "1". After execution of the instruction the specified operand is reset to "0". If there is a current RLO of "0", the signal state of the specified operand remains unchanged.

Syntax

Use the following syntax for the "Reset" instruction:

```
R <Operand>
```

Parameters

The following table shows the parameters of the "Reset" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand that is reset when RLO = "1".

Example

The following example shows how the instruction works:

```
STL                          Explanation
A "Tag_Input_1"              // Query the signal state of the operand for "1" and
                             AND with the current RLO.
A "Tag_Input_2"              // Query the signal state of the operand for "1" and
                             AND with the current RLO.
R "Tag_Output"               // Reset the operand to "0" if RLO is "1"
```

See also

Overview of the valid data types (Page 2236)
Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)
Evaluating errors with output parameter RET_VAL (Page 5370)
Call function blocks as single or multi-instances (Page 2182)
Basic information on the status word (Page 2466)
Querying and setting status bits in STL (Page 2467)

S: Set

Description

Use the "Set" instruction to set the signal state of a specified operand to "1".

The instruction is only executed when the current result of logic operation (RLO) is "1". After the instruction executes, the specified operand is set to "1". If there is a current RLO of "0", the signal state of the specified operand remains unchanged.

Syntax

Use the following syntax for the "Set" instruction:

```
S <Operand>
```

Parameters

The following table shows the parameters of the "Set" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand that is set when RLO = "1".

Example

The following example shows how the instruction works:

```
STL                          Explanation
A "Tag_Input_1"              // Query the signal state of the operand for "1" and
                             AND with the current RLO.
A "Tag_Input_2"              // Query the signal state of the operand for "1" and
                             AND with the current RLO.
S "Tag_Output"               // Sets the operand to "1" if RLO = "1".
```

See also

Overview of the valid data types (Page 2236)
Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)
Evaluating errors with output parameter RET_VAL (Page 5370)
Call function blocks as single or multi-instances (Page 2182)
Basic information on the status word (Page 2466)
Querying and setting status bits in STL (Page 2467)

NOT: Invert RLO

Description

Use the "Invert RLO" instruction to invert the signal state of the result of logic operation (RLO). You can use the "Invert RLO" instruction anywhere, even within a logic operation.

Syntax

Use the following syntax for the "Invert RLO" instruction:

```
NOT
```

Example

The following example shows how the instruction works:

STL	Explanation
`A "Tag_Input_1"`	`// Query the signal state of the "Tag_Input_1" operand for "1" and AND with the current RLO.`
`A "Tag_Input_2"`	`// Query the signal state of the "Tag_Input_2" operand for "1" and AND with the current RLO.`
`A "Tag_Input_3"`	`// Query the signal state of the "Tag_Input_3" operand for "1" and AND with the current RLO.`
`NOT`	`// Invert signal state of the RLO`
`= "Tag_Output"`	`// Assign signal state of the RLO to the "Tag_Output" operand.`

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Input_1	1	1	1	0	0	0	1	0
Tag_Input_2	0	1	1	0	1	1	0	1

Operand	Value							
Tag_Input_3	0	0	1	0	0	1	1	1
Tag_Output	1	1	0	1	1	1	1	1

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

SET: Set RLO to 1

Description

Use the "Set RLO to 1" instruction to set the current result of logic operation (RLO) to signal state "1".

Syntax

Use the following syntax for the "Set RLO to 1" instruction:

```
SET
```

Example

The following example shows how the instruction works:

STL program	Signal state	Result of logic operation (RLO)
SET		1
= “Tag_Output_1“	1	
= “Tag_Output_2“	1	
= “Tag_Output_3“	1	
CLR		0
= “Tag_Output_4“	0	
= “Tag_Output_5“	0	

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

CLR: Reset RLO to 0

Description

Use the "Reset RLO to 0" instruction to set the current result of logic operation (RLO) to signal state "0".

Syntax

Use the following syntax for the "Reset RLO to 0" instruction:

```
CLR
```

Example

The following example shows how the instruction works:

STL program	Signal state	Result of logic operation (RLO)
SET		1
= “Tag_Output_1“	1	
= “Tag_Output_2“	1	
= “Tag_Output_3“	1	
CLR		0
= “Tag_Output_4“	0	
= “Tag_Output_5“	0	

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

SAVE: Save RLO in BR bit

Description

Use the "Save RLO in BR bit" instruction to save the result of logic operation (RLO) in the binary result (BR). When it executes, the instruction transfers the signal state of the current result of logic operation to the status bit BR. The instruction operates independent of the conditions and does not affect any other status bits.

Note

After executing the "Save RLO in BR bit" instruction, the BR bit can be modified again by the following instructions in the same or lower-level blocks.

Use the "Save RLO in BR bit" instruction to check the execution status of a block. If, for example, you use the instruction "Save RLO in BR bit" at the end of a block, the BR bit is set to the signal state of the current result of logic operation of the block.

The BR bit helps to implement the EN/ENO mechanism for block calls. For more information on this topic, refer to "See also".

Syntax

Use the following syntax for the "Save RLO in BR bit" instruction:

```
SAVE
```

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input"	// Query signal state of the Tag_Input operand for "1".
SAVE	// Transfer signal state of the RLO to the BR bit.
BEC	// End of block if RLO = "1".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

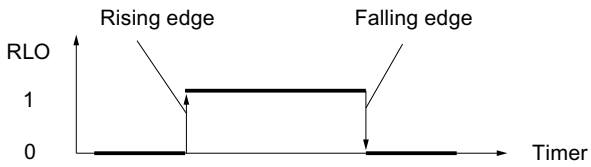
Querying and setting status bits in STL (Page 2467)

FN: Scan RLO for negative signal edge

Description

Use the "Scan RLO for negative signal edge" instruction to query a "1" to "0" change in the signal state of the result of logic operation (RLO). The instruction compares the current signal state of the result of logic operation (RLO) with the signal state of the previous query that is saved in the operand of the instruction. If the instruction detects a change in the RLO from "1" to "0", there is a negative, falling edge.

The following figure shows the change in the RLO in case of a falling and a rising edge:



The negative edge is queried each time the instruction executes. When the instruction detects a falling edge, the RLO is set to signal state "1" after edge evaluation. If no falling edge is detected, the signal state of the result of logic operation is "0".

Note

The operand for the edge evaluation must be located in bit memory or in a data block.

Syntax

Use the following syntax for the "Scan RLO for negative signal edge" instruction:

```
FN <Operand>
```

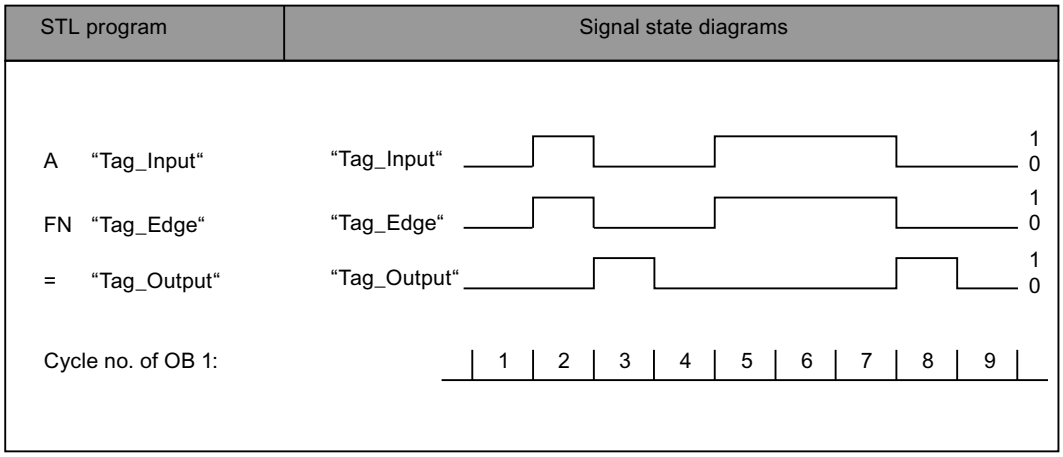
Parameters

The following table shows the parameters of the "Scan RLO for negative signal edge" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Edge memory bit in which the RLO of the last query is saved.

Example

The following example shows how the instruction works:



See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

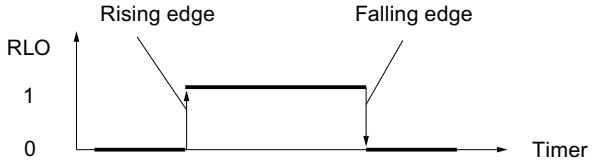
Querying and setting status bits in STL (Page 2467)

FP: Scan RLO for positive signal edge

Description

Use the "Scan RLO for positive signal edge" instruction to query a "0" to "1" change in the signal state of the result of logic operation (RLO). The instruction compares the current signal state of the result of logic operation (RLO) with the signal state of the previous query that is saved in the operand of the instruction. If the instruction detects a change in the RLO from "0" to "1", there is a positive, rising edge.

The following figure shows the change in the RLO in case of a falling and a rising edge:



The positive edge is queried each time the instruction executes. When the instruction detects a rising edge, the RLO after the edge evaluation is set to signal state "1". If no rising edge is detected, the signal state of the result of logic operation is "0".

Note

The operand for the edge evaluation must be located in bit memory or in a data block.

Syntax

Use the following syntax for the "Scan RLO for positive signal edge" instruction:

```
FP <Operand>
```

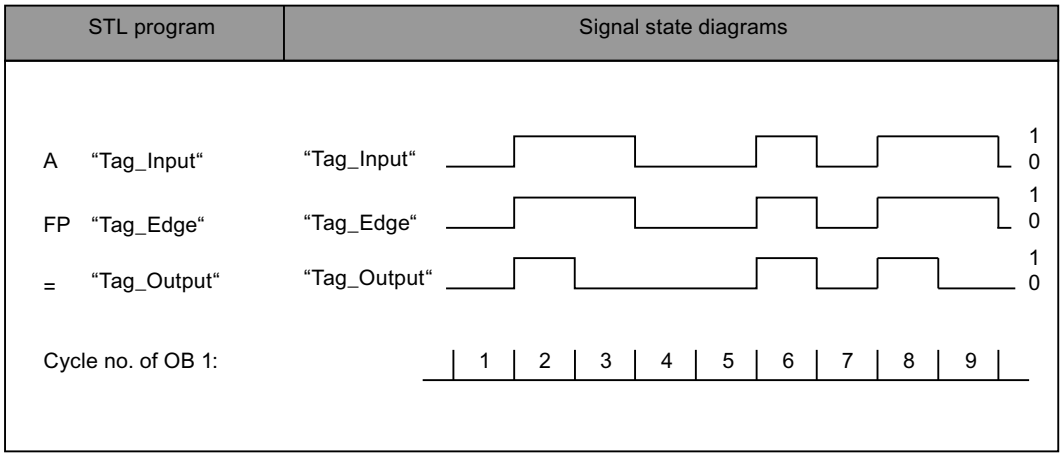
Parameters

The following table shows the parameters of the "Scan RLO for positive signal edge" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Edge memory bit in which the RLO of the last query is saved.

Example

The following example shows how the instruction works:



See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Timer operations

FR: Enable timer

Description

You can use the instruction "Enable timer" to restart a timer. The instruction is executed with a positive edge and resets the internal edge trigger flag for the starting of a timer. During the processing of an instruction for starting the timer with a result of logic operation "1", the timer restarts even if no positive signal edge is present before the start instruction. If the result of logic operation is "0" before a start instruction, the resetting of the internal edge trigger flag does not affect the timer.

A change of the result of logic operation to "0" before the "Enable timer" instruction has no affect after the restart of the timer.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Syntax

The following syntax is used for the "Enable timer" instruction:

```
FR <Timer>
```

Parameters

The following table shows the parameters of the "Enable timer" instruction:

Operand	Declaration	Data type	Memory area	Description
<timer>	Input	TIMER	T	Enabled timer

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_EnableInput"	// Query operand for "1" and AND with the current RLO.
FR "MyTimer"	// If the signal edge is positive, enable timer.
A "Tag_StartInput"	// Query operand for "1" and AND with the current RLO.
L S5T#10s	// Load period of time (10s)
SP "MyTimer"	// Start pulse timer if there is a positive signal edge at the "Tag_StartInput" operand or at the "Tag_EnableInput" operand and the "Tag_StartInput" operand has signal state "1".
A "Tag_Reset"	// Query operand for "1" and AND with the current RLO.
R "MyTimer"	// If RLO = "1", reset timer.
A "MyTimer"	// Query status of timer for "1".
= "Tag_TimerStatus"	// Assign timer status to the operand.
L "MyTimer"	// Load current time value of timer in accumulator 1.
T "Tag_TimerValue"	// Transfer current time value to operand "Tag_TimerValue".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

L: Load timer value (Page 3929)

L: Load timer value

Description

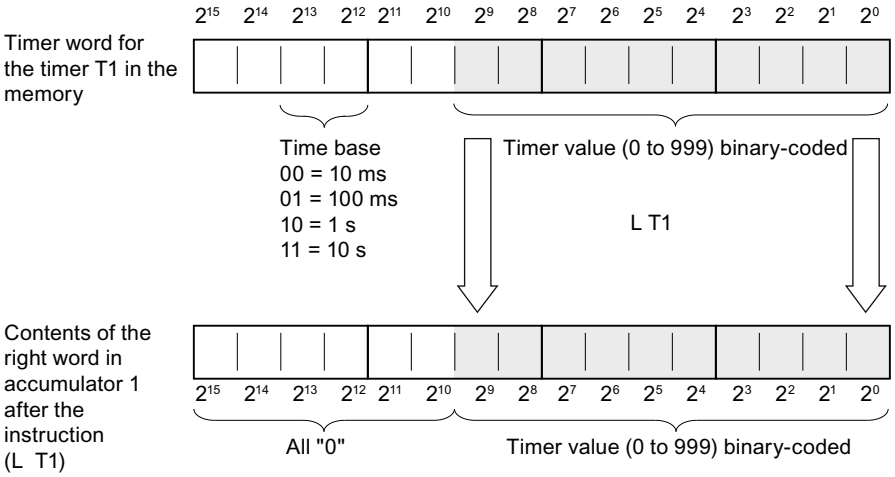
The "Load timer value" instruction loads the time value of a specified timer in accumulator 1. The binary-coded timer value is loaded without the timebase. Zeros are written to accumulator 1 instead of the timebase.

After the loading, the value in accumulator 1 corresponds to a positive number of the data type INT.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

The following figure is an example of how the time value is loaded in the accumulator:



Syntax

The following syntax is used for the "Load timer value" instruction:

```
L <Timer>
```

Parameters

The following table shows the parameters of the "Load timer value" instruction:

Operand	Declaration	Data type	Memory area	Description
<timer>	Input	TIMER	T	Timer whose current time value is loaded

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_EnableInput"	// Query operand for "1" and AND with the current RLO.
FR "MyTimer"	// If edge is positive, enable timer.
A "Tag_StartInput"	// Query operand for "1" and AND with the current RLO.
L S5T#10s	// Load period of time (10s)
SP "MyTimer"	// Start timer as pulse if there is a positive edge at the "Tag_StartInput" operand or at the "Tag_EnableInput" operand and the "Tag_StartInput" operand has signal state "1".
A "Tag_Reset"	// Query operand for "1" and AND with the current RLO.
R "MyTimer"	// If RLO = "1", reset timer.
A "MyTimer"	// Query status of timer for "1".
= "Tag_TimerStatus"	// Assign timer status to the operand.
L "MyTimer"	// Load current time value of timer in accumulator 1.
T "Tag_TimerValue"	// Transfer current time value to operand "Tag_TimerValue".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

LC: Load BCD-coded timer value

Description

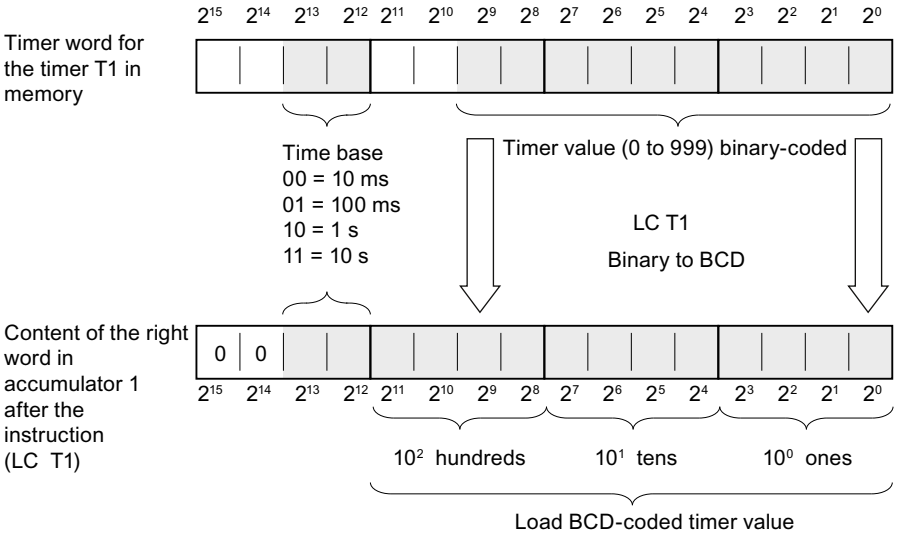
The "Load BCD-coded timer value" instruction loads a binary-coded timer value of a specified timer in BCD format in accumulator 1. The timebase is transferred to accumulator 1 during loading.

After loading, the value in accumulator 1 corresponds to a time duration in S5TIME format. The left word in accumulator 1 is padded with zeros.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

The following figure shows how the time value is loaded in the accumulator:



Syntax

The following syntax is used for the "Load BCD-coded timer value" instruction:

```
LC <Timer>
```

Parameters

The following table shows how the parameters of the "Load BCD-coded timer value" instruction:

Operand	Declaration	Data type	Memory area	Description
<timer>	Input	TIMER	T	Timer whose current BCD-coded timer value is loaded.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_EnableInput"	// Query operand for "1" and AND with the current RLO.
FR "MyTimer"	// If edge is positive, enable timer.
A "Tag_StartInput"	// Query operand for "1" and AND with the current RLO.
L S5T#10s	// Load period of time (10s)
SP "MyTimer"	// Start timer as pulse if there is a positive edge at the "Tag_StartInput" operand or at the "Tag_EnableInput" operand and the "Tag_StartInput" operand has signal state "1".
A "Tag_Reset"	// Query operand for "1" and AND with the current RLO.
R "MyTimer"	// If RLO = "1", reset timer.
A "MyTimer"	// Query status of timer for "1".
= "Tag_TimerStatus"	// Assign timer status to the operand.
LC "MyTimer"	// Load current BCD-coded timer value of timer in accumulator 1.
T "Tag_TimerValue"	// Transfer current time value to operand "Tag_TimerValue".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

L: Load timer value (Page 3929)

R: Reset timer

Description

You can use the "Reset timer" instruction to reset a specified timer to "0". The instruction is executed when the current result of logic operation (RLO) is "1". As long as the RLO is "1"

before the instruction, the specified timer returns the value "0". The instruction also resets the time value and the time basis of the programmed period of time to "0".

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

Note

The "Reset timer" instruction does not reset the internal edge memory bit. To reset the internal edge memory bit, program the "Enable timer" instruction or execute an instruction to start the time with the signal state "0".

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Syntax

The following syntax is used for the "Reset timer" instruction:

```
R <Timer>
```

Parameters

The following table shows the parameters of the "Reset timer" instruction:

Operand	Declaration	Data type	Memory area	Description
<timer>	Input	TIMER	T	Timer, which is reset

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
A "Tag_Reset"                // Query operand for "1" and AND with the current RLO.
R "MyTimer"                  // If RLO = "1", reset timer.
A "MyTimer"                  // Query status of timer for "1".
= "Tag_TimerStatus"          // Assign timer status to the operand.
L "MyTimer"                  // Load current time value of timer in accumulator 1.
T "Tag_TimerValue"           // Transfer current time value to operand
                             "Tag_TimerValue".
```

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

L: Load timer value (Page 3929)

SP: Start pulse timer

Description

You can use the "Start pulse timer" instruction to start a specified timer on a positive signal edge. If a positive signal edge is detected, the instruction is executed and the timer started. The timer then runs down with the period of time specified in accumulator 1, as long as the result of logic operation (RLO) before the instruction remains "1". If the RLO changes to "0" before the end of the period of time, the timer is stopped. The queries of the timer status for "1" return the query result "1", as long as the period of time is running.

The timer can be restarted with a positive signal edge before the "Start pulse timer" instruction or using the "Enable timer" instruction. This restart can be effected provided the timer is not reset.

The period of time in accumulator 1 is made up of a time value and a time base. If a specified timer is started with the "Start pulse timer" instruction, the time value is counted down according to the timebase. The timer runs until the counter value is counted down to zero.

A valid period of time in S5TIME format must be present in accumulator 1, even if the timer is not started during execution of the instruction. If the accumulator 1 does not include a valid BCD value and the RLO of the preceding logic operation returns the value "0", the status bit OV is set to "1". No synchronous error handling is started. If the accumulator 1 does include a valid BCD value, the status bit OV is reset to "0".

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Syntax

The following syntax is used for the "Start pulse timer" instruction:

```
SP <Timer>
```

Parameters

The following table shows the parameters of the "Start pulse timer" instruction:

Operand	Declaration	Data type	Memory area	Description
<timer>	Input	TIMER	T	Timer that is started as pulse.

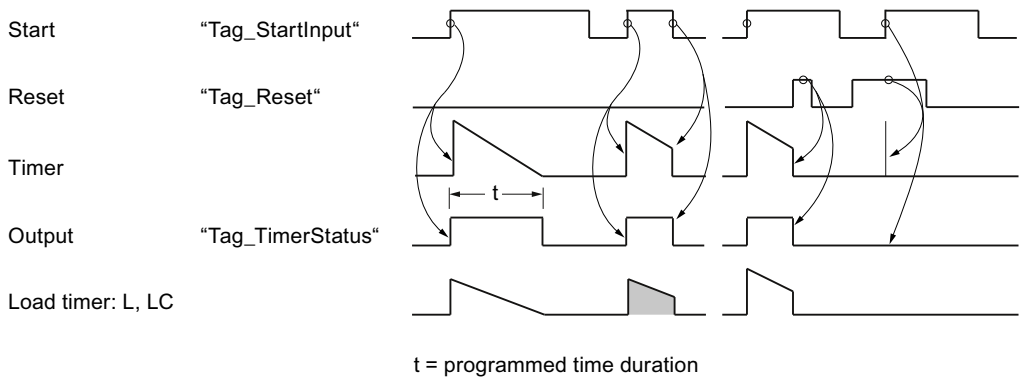
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_StartInput"	// Query operand for "1" and AND with the current RLO.
L S5T#10s	// Load period of time (10s)
SP "MyTimer"	// Start pulse timer if there is a positive signal edge at the "Tag_StartInput" operand or at the "Tag_EnableInput" operand and the "Tag_StartInput" operand has signal state "1".
A "Tag_Reset"	// Query operand for "1" and AND with the current RLO.
R "MyTimer"	// If RLO = "1", reset timer and stop timing
A "MyTimer"	// Query status of timer for "1".
= "Tag_TimerStatus"	// Assign timer status to the operand.
L "MyTimer"	// Load current time value of timer in accumulator 1.
T "Tag_TimerValue"	// Transfer current time value to operand "Tag_TimerValue".
LC "MyTimer"	// Load current time value in BCD format in accumulator 1.
T "Tag_TimerValue_BCD"	// Transfer time value in BCD format to operand "Tag_TimerValue_BCD"

The figure below shows the timer diagram of the example:



See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

L: Load timer value (Page 3929)

SE: Start extended pulse timer

Description

You can use the "Start extended pulse timer" to start a specified timer on a positive signal edge. If a positive signal edge is detected, the instruction is executed and the timer started. The timer then runs down with the period of time specified in accumulator 1, as long as the result of logic operation changes to "0" before the instruction. The queries of the timer status for "1" return the query result "1", as long as the period of time is running.

With each positive signal edge, the instruction restarts the timer with the programmed time duration, even if the timer has still not run down.

The period of time in accumulator 1 is made up of a time value and a time base. If a specified timer is started with the "Start extended pulse timer" instruction, the time value is counted down according to the timebase. The timer runs until the counter value is counted down to zero.

A valid period of time in S5TIME format must be present in accumulator 1, even if the timer is not started during execution of the instruction. If the accumulator 1 does not include a valid BCD value and the RLO of the preceding logic operation returns the value "0", the status bit

OV is set to "1". No synchronous error handling is started. If the accumulator 1 does include a valid BCD value, the status bit OV is reset to "0".

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Syntax

The following syntax is used for the "Start extended pulse timer" instruction:

```
SE <Timer>
```

Parameters

The following table shows the parameters of the "Start extended pulse timer" instruction:

Operand	Declaration	Data type	Memory area	Description
<timer>	Input	TIMER	T	Timer, which is started as extended pulse

For additional information on valid data types, refer to "See also".

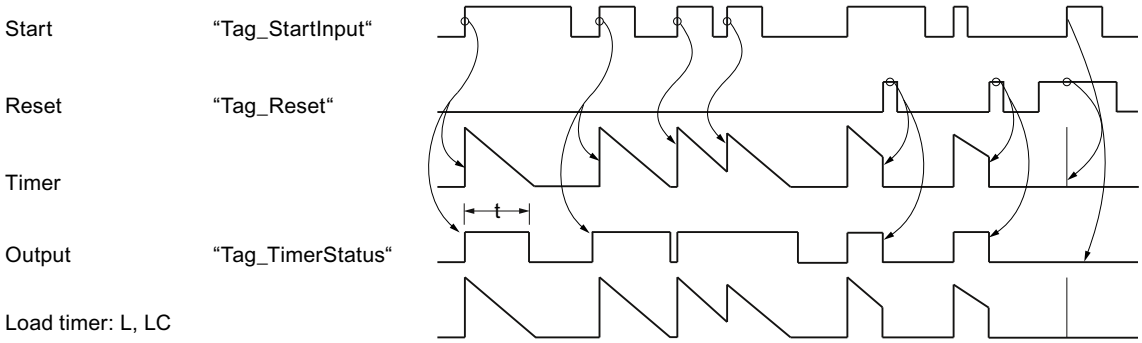
Example

The following example shows how the instruction works:

```
STL                          Explanation
A "Tag_StartInput"           // Query operand for "1" and AND with the current RLO.
L S5T#10s                    // Load period of time (10s)
SE "MyTimer"                 // Start timer as extended pulse if there is a
                             positive signal edge at the "Tag_StartInput" operand
                             or at the "Tag_EnableInput" operand and the
                             "Tag_StartInput" operand has signal state "1".
A "Tag_Reset"                // Query operand for "1" and AND with the current RLO.
R "MyTimer"                  // If RLO = "1", reset timer and stop timing
A "MyTimer"                  // Query status of timer for "1".
= "Tag_TimerStatus"          // Assign timer status to the operand.
L "MyTimer"                  // Load current time value of timer in accumulator 1.
```

STL	Explanation
`T "Tag_TimerValue"`	`// Transfer current time value to operand "Tag_TimerValue".`
`LC "MyTimer"`	`// Load current time value in BCD format in accumulator 1.`
`T "Tag_TimerValue_BCD"`	`// Transfer time value in BCD format to operand "Tag_TimerValue_BCD"`

The figure below shows the timer diagram of the example:



t = programmed time duration

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

L: Load timer value (Page 3929)

SD: Start on-delay timer

Description

You can use the "Start on-delay timer" instruction to start a specified timer on a positive signal edge. If a positive signal edge is detected, the instruction is executed and the timer started. The timer then runs down with the period of time specified in accumulator 1, as long as the result of logic operation (RLO) before the instruction remains "1". If the result of logic operation changes to "0" before the end of the period of time, the timer is stopped.

The queries of the timer status for "1" return the query result "1" if the time expired correctly and the current RLO before the instruction is "1". If the time has not expired or if the RLO changes to "0", the queries of the timer status for "1" return the query result "0".

The period of time in accumulator 1 is made up of a time value and a time base. If a specified timer is started with the "Start on-delay timer" instruction, the time value is counted down according to the timebase. The timer runs until the counter value is counted down to zero.

A valid period of time in S5TIME format must be present in accumulator 1, even if the timer is not started during execution of the instruction. If the accumulator 1 does not include a valid BCD value and the RLO of the preceding logic operation returns the value "0", the status bit OV is set to "1". No synchronous error handling is started. If the accumulator 1 does include a valid BCD value, the status bit OV is reset to "0".

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Syntax

The following syntax is used for the "Start on-delay timer" instruction:

```
SD <Timer>
```

Parameters

The following table shows the parameters of the "Start on-delay timer" instruction:

Operand	Declaration	Data type	Memory area	Description
<timer>	Input	TIMER	T	Timer, which is started as on delay

For additional information on valid data types, refer to "See also".

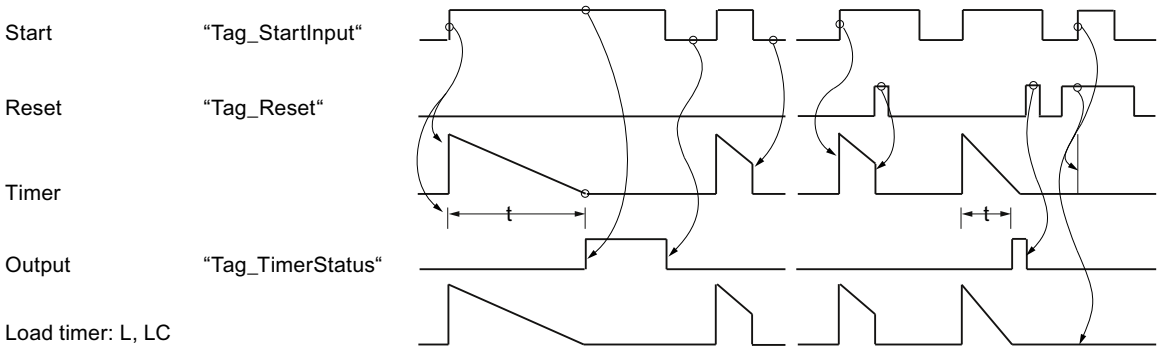
Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_StartInput"	// Query operand for "1" and AND with the current RLO.
L S5T#10s	// Load period of time (10s)
SD "MyTimer"	// Start timer as an on delay if there is a positive signal edge at the "Tag_StartInput" operand or at the

STL	Explanation
	"Tag_EnableInput" operand and the "Tag_StartInput" operand has signal state "1".
A "Tag_Reset"	// Query operand for "1" and AND with the current RLO.
R "MyTimer"	// If RLO = "1", reset timer and stop timing
A "MyTimer"	// Query status of timer for "1".
= "Tag_TimerStatus"	// Assign timer status to the operand.
L "MyTimer"	// Load current time value of timer in accumulator 1.
T "Tag_TimerValue"	// Transfer current time value to operand "Tag_TimerValue".
LC "MyTimer"	// Load current time value in BCD format in accumulator 1.
T "Tag_TimerValue_BCD"	// Transfer time value in BCD format to the operand "Tag_TimerValue_BCD"

The figure below shows the timer diagram of the example:



t = programmed time duration

See also

Overview of the valid data types (Page 2236)
Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)
Evaluating errors with output parameter RET_VAL (Page 5370)
Call function blocks as single or multi-instances (Page 2182)
Basic information on the status word (Page 2466)
Querying and setting status bits in STL (Page 2467)
L: Load timer value (Page 3929)

SS: Start retentive on-delay timer

Description

You can use the "Start retentive on-delay timer" instruction to start a specified timer on a positive signal edge. If a positive signal edge is detected, the instruction is executed and the timer started. The timer then runs with the period of time specified in accumulator 1, even if the result of logic operation (RLO) changes to "0" before the instruction. When the time has run out, queries of the timer status for "1" return the query result "1". The query result is not hereby affected by the signal state of the current result of logic operation before the instruction.

With each positive edge, the instruction restarts the timer with the programmed period of time, even if the time has still not run down.

The period of time in accumulator 1 is made up of a time value and a time base. If a specified timer is started with the "Start retentive on-delay timer" instruction, the time value is counted down according to the timebase. The timer runs until the counter value is counted down to zero.

A valid period of time in S5TIME format must be present in accumulator 1, even if the timer is not started during execution of the instruction. If the accumulator 1 does not include a valid BCD value and the RLO of the preceding logic operation returns the value "0", the status bit OV is set to "1". No synchronous error handling is started. If the accumulator 1 does include a valid BCD value, the status bit OV is reset to "0".

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Syntax

The following syntax is used for the "Start retentive on-delay timer" instruction:

```
SS <Timer>
```

Parameters

The following table shows the parameters of the "Start retentive on-delay timer" instruction:

Operand	Declaration	Data type	Memory area	Description
<timer>	Input	TIMER	T	Timer that is started as retentive on delay

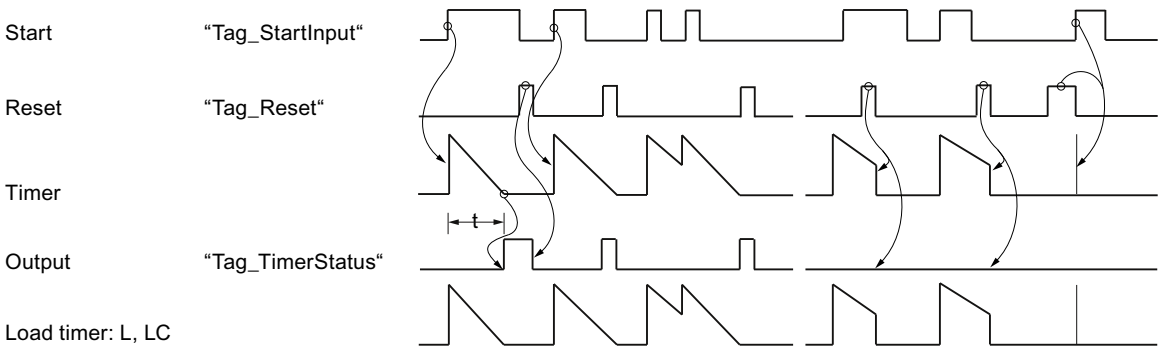
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_StartInput"	// Query operand for "1" and AND with the current RLO.
L S5T#10s	// Load period of time (10s)
SS "MyTimer"	// Start retentive on-delay timer if there is a positive signal edge at the "Tag_StartInput" operand or at the "Tag_EnableInput" operand and the "Tag_StartInput" operand has signal state "1".
A "Tag_Reset"	// Query operand for "1" and AND with the current RLO.
R "MyTimer"	// If RLO = "1", reset timer and stop timing
A "MyTimer"	// Query status of timer for "1".
= "Tag_TimerStatus"	// Assign timer status to the operand.
L "MyTimer"	// Load current time value of timer in accumulator 1.
T "Tag_TimerValue"	// Transfer current time value to operand "Tag_TimerValue".
LC "MyTimer"	// Load current time value in BCD format in accumulator 1.
T "Tag_TimerValue_BCD"	// Transfer time value in BCD format to the operand "Tag_TimerValue_BCD"

The figure below shows the timer diagram of the example:



t = programmed time duration

See also

Overview of the valid data types (Page 2236)
Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)
Evaluating errors with output parameter RET_VAL (Page 5370)
Call function blocks as single or multi-instances (Page 2182)
Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

L: Load timer value (Page 3929)

SF: Start off-delay timer

Description

You can use the "Start off-delay timer" instruction to start a specified timer on a positive signal edge. If a negative signal edge is detected, the instruction is executed and the timer started. The timer then runs down with the period of time specified in accumulator 1. If the result of logic operation (RLO) changes to "1" before the end of the period of time, the timer is reset. The timer is only restarted if a negative signal edge is detected during the processing of the instruction.

The queries of the timer status for "1" return the query result "1" if the signal state of the result of logic operation is "1" at the execution of the instruction or if the programmed time is running. If the time does not elapse or the RLO is "0", queries of the timer status for "1" return the query result "0".

The period of time in accumulator 1 is made up of a time value and a time base. If a specified timer is started with the "Start off-delay timer" instruction, the time value is counted down according to the timebase. The timer runs until the counter value is counted down to zero.

A valid period of time in S5TIME format must be present in accumulator 1, even if the timer is not started during execution of the instruction. If the accumulator 1 does not include a valid BCD value and the RLO of the preceding logic operation returns the value "0", the status bit OV is set to "1". No synchronous error handling is started. If the accumulator 1 does include a valid BCD value, the status bit OV is reset to "0".

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Syntax

The following syntax is used for the "Start off-delay timer" instruction:

```
SF <Timer>
```

Parameters

The following table shows the parameters of the "Start off-delay timer" instruction:

Operand	Declaration	Data type	Memory area	Description
<timer>	Input	TIMER	T	Timer, which is started as off delay

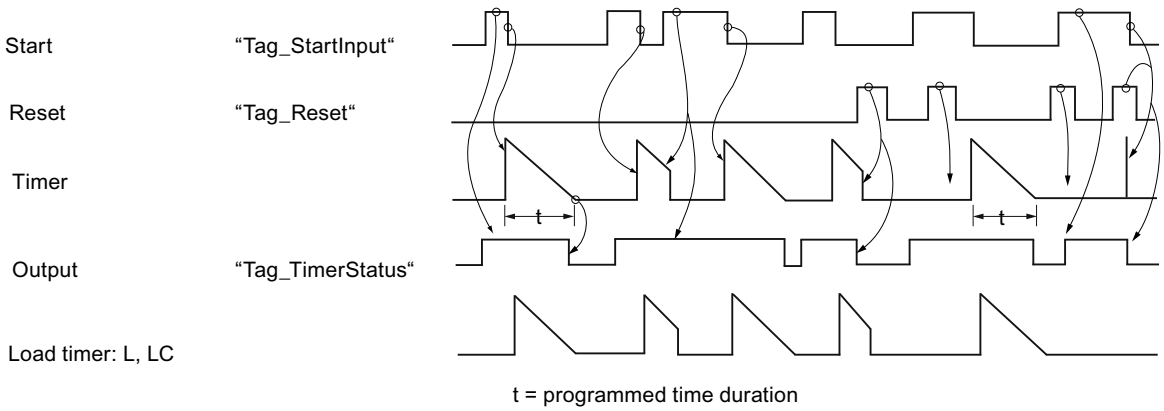
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
A "Tag_StartInput"           // Query operand for "1" and AND with the current RLO.
L S5T#10s                    // Load period of time (10s)
SF "MyTimer"                 // Start timer as an off delay if there is a negative
                             signal edge at the "Tag_StartInput" operand or at the
                             "Tag_EnableInput" operand and the "Tag_StartInput"
                             operand has signal state "1".
A "Tag_Reset"                // Query operand for "1" and AND with the current RLO.
R "MyTimer"                  // If RLO = "1", reset timer and stop timing
A "MyTimer"                  // Query status of timer for "1".
= "Tag_TimerStatus"          // Assign timer status to the operand.
L "MyTimer"                  // Load current time value of timer in accumulator 1.
T "Tag_TimerValue"           // Transfer current time value to operand
                             "Tag_TimerValue".
LC "MyTimer"                 // Load current time value in BCD format in
                             accumulator 1.
T "Tag_TimerValue_BCD"       // Transfer time value in BCD format to operand
                             "Tag_TimerValue_BCD"
```

The figure below shows the timer diagram of the example.



See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

L: Load timer value (Page 3929)

Counter operations

FR: Enable counter

Description

Use the "Enable counter" instruction to execute the instructions "Set counter", "Count up" and "Count down" without the presence of a positive signal edge. During processing, the instruction resets the internal edge flag to the signal state "0". The query before the corresponding count instruction must continue to have signal state "1".

The "Enable counter" instruction is executed when the result of logic operation (RLO) before the instruction changes from "0" to "1".

A counter does not have to be enabled for the count instruction to be executed.

Syntax

Use the following syntax for the "Enable counter" instruction:

```
FR <Operand>
```

Parameters

The following table shows the parameters of the "Enable counter" instruction:

Operand	Declaration	Data type	Memory area	Description
<Operand>	Input	COUNTER	C	Counter that is enabled.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input_1"	// Query signal state for "1" and AND with the current RLO.
FR "MyCounter"	// Enable counter
A "Tag_Input_2"	// Query signal state for "1" and AND with the current RLO.
CD "MyCounter"	// Count down
A "MyCounter"	// Query counter status
= "Tag_Output"	// Assign query result to the operand.

In the example the counter ("MyCounter") is enabled when the signal state of the operand "Tag_Input_1" changes from "0" to "1". When the operand "Tag_Input_2" has signal state "1", the count value is decremented by one when the counter is enabled. When the signal state of the address "Tag_Input_2" is "0", the count value remains unchanged. The count value is also decremented by one regardless of the counter enabling, when the signal state of the address "Tag_Input_2" changes from "0" to "1".

The query of the counter status has the result "1" when the counter reading is greater than zero. If the counter reading is equal to zero, the result of the query is "0".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

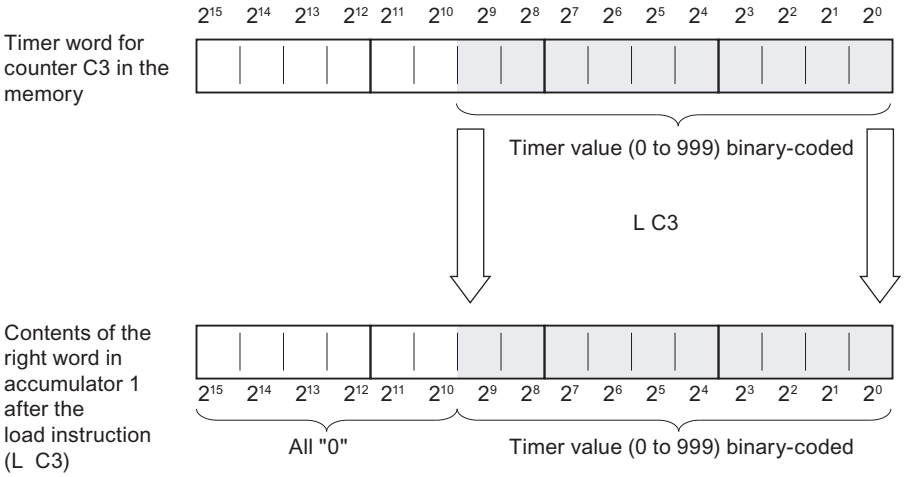
Querying and setting status bits in STL (Page 2467)

L: Load counter

Description

The "Load counter" instruction transfers the current value of a specified counter binary-coded to the right word of accumulator 1. The loaded count value is available as a 16-bit integer for further processing.

The following figure shows how the instruction works:



Syntax

Use the following syntax for the "Load counter" instruction:

```
L <Operand>
```

Parameters

The following table shows the parameters of the "Load counter" instruction:

Operand	Declaration	Data type	Memory area	Description
<Operand>	Input	COUNTER	C	Counter whose current value is loaded.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
L "MyCounter"	// Load current count value binary-coded in accumulator 1
L "Tag_Value"	// Move the counter value to accumulator 2.
	// Load value of the operand "Tag_Value" in accumulator 1
>=I	// Query whether the count value is greater than or equal to the value of the address "Tag_Value"
= "Tag_Output"	// Assign query result to the operand.

See also

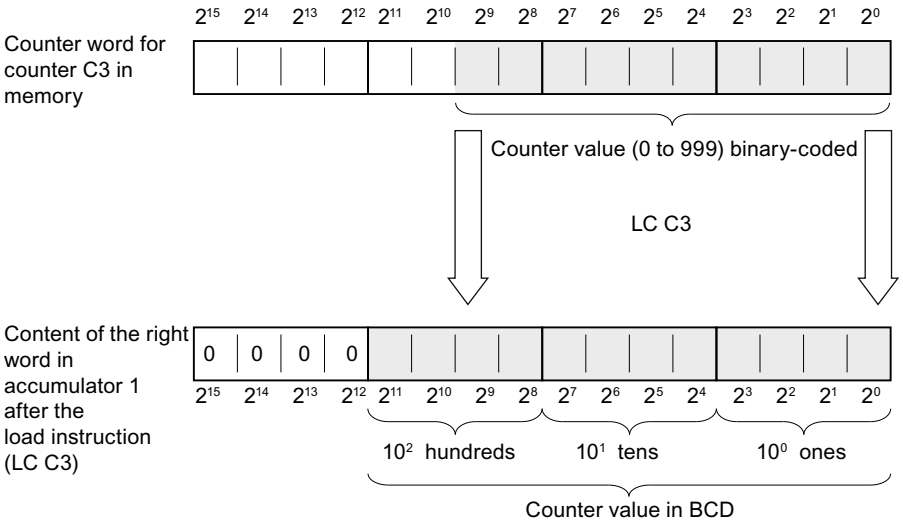
Overview of the valid data types (Page 2236)
Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)
Evaluating errors with output parameter RET_VAL (Page 5370)
Call function blocks as single or multi-instances (Page 2182)
Basic information on the status word (Page 2466)
Querying and setting status bits in STL (Page 2467)

LC: Load BCD-coded counter value

Description

The "Load BCD-coded counter value" instruction transfers the current value of a specified counter as BCD to the right-hand word of the accumulator 1. The loaded counter value is available in BCD form for further processing.

The following figure shows how the "Load BCD-coded counter value" instruction works:



Syntax

Use the following syntax for the "Load BCD-coded counter value" instruction:

```
LC <Operand>
```

Parameters

The following table shows the parameters of the "Load BCD-coded counter value" instruction:

Operand	Declaration	Data type	Memory area	Description
<Operand>	Input	COUNTER	C	Counter whose current value is loaded.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
LC "MyCounter"	// Load current BCD-coded counter value in accumulator 1.
T "Tag_Output"	// Transfer current counter value to the address "Tag_Output" and save.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

R: Reset counter

Description

Use the "Reset counter" instruction to reset the current count value of a specified counter to "0".

The instruction is only executed when the result of logic operation (RLO) before the instruction has signal state "1". As long as the RLO before the "Reset counter" instruction is "1", the queries of the counter status for "1" return the result "0".

When a counter is reset, the internal edge memory bit for the instructions "Set counter", "Count up" and "Count down" is not reset.

Syntax

Use the following syntax for the "Reset counter" instruction:

```
R <Operand>
```

Parameters

The following table shows the parameters of the "Reset counter" instruction:

Operand	Declaration	Data type	Memory area	Description
<Operand>	Input	COUNTER	C	Counter whose count value is reset to "0".

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input"	// Query signal state for "1" and AND with the current RLO.
R "MyCounter"	// Reset counter

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

S: Set counter

Description

Use the "Set counter" instruction to set a specified counter to the value contained in accumulator 1 at the time of the query.

The "Set counter" instruction is executed when the result of logic operation before the instruction changes from "0" to "1". The positive signal edge can also be achieved with the "Enable counter" instruction that resets the internal edge memory bit to "0".

To set a counter, accumulator 1 must contain a valid count value. Valid count values are values in BCD format with three decades. Only positive count values are permitted; the count instruction cannot process negative count values. If a valid BCD value is present in accumulator 1, the status bit OV has signal state "0". If there is no valid BCD value, the status bit OV has signal state "1". No synchronous error handling is started.

Syntax

Use the following syntax for the "Set counter" instruction:

```
S <Operand>
```

Parameters

The following table shows the parameters of the "Set counter" instruction:

Operand	Declaration	Data type	Memory area	Description
<Operand>	Input	COUNTER	C	Counter that is set.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input"	// Query signal state for "1" and AND with the current RLO.
L "Tag_CountValue"	// Load count value in accumulator 1.
S "MyCounter"	// Set counter to the value that is in accumulator 1

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

CU: Count up

Description

Use the "Count up" instruction to increment the current count value by one when the result of logic operation (RLO) before the instruction changes from "0" to "1". The counter value continues to be incremented for each positive signal edge until the high limit of "999" is reached. When the limit is reached the RLO before the instruction no longer has any affect on the count value.

A positive signal edge is always required to execute the "Count up" instruction. The positive signal edge can also be achieved with the "Enable counter" instruction that resets the internal edge memory bit to "0".

Syntax

Use the following syntax for the "Count up" instruction:

```
CU <operand>
```

Parameters

The following table shows the parameters of the "Count up" instruction:

Operand	Declaration	Data type	Memory area	Description
<Operand>	Input	COUNTER	C	Counter, whose count value is incremented by one on a positive edge.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input"	// Query signal state for "1" and AND with the current RLO.
CU "MyCounter"	// Increment count value by one on a positive edge.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

CD: Count down

Description

Use the "Count down" instruction to decrement the current count value by one when the result of logic operation (RLO) before the instruction changes from "0" to "1". The count value continues to be decremented for each positive signal edge until the low limit of "0" is reached. When the limit is reached the RLO before the instruction no longer has any affect on the count value. Counting does not occur when count value is negative.

A positive signal edge is always required to execute the "Count down" instruction. The positive signal edge can also be achieved with the "Enable counter" instruction that resets the internal edge memory bit to "0".

Syntax

The following syntax is used for the "Count down" instruction:

```
CD <operand>
```

Parameters

The following table shows the parameters of the "Count down" instruction:

Operand	Declaration	Data type	Memory area	Description
<Operand>	Input	COUNTER	C	Counter, whose count value is decremented by one on a positive edge.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input"	// Query signal state for "1" and AND with the current RLO.
CD "MyCounter"	// Decrement counter value by one on a positive edge.

See also

Overview of the valid data types (Page 2236)
Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)
Evaluating errors with output parameter RET_VAL (Page 5370)
Call function blocks as single or multi-instances (Page 2182)
Basic information on the status word (Page 2466)
Querying and setting status bits in STL (Page 2467)

Comparator operations

? I: Compare 16-bit integers

Description

Use the "Compare 16-bit integers" instruction to compare the contents of the right word in accumulator 2 with the contents of the right word in accumulator 1. The contents of accumulators 1 and 2 are interpreted as 16-bit integers. The instruction performs the comparison depending on the specified comparison function.

The following table shows the available comparison functions and their meaning:

Comparison function	Meaning
==I	Compare 16-bit integers for "equivalence"
<>I	Compare 16-bit integers for "non-equivalence"
>I	Compare 16-bit integers for "greater than"
<I	Compare 16-bit integers for "less than"
>=I	Compare 16-bit integers for "greater than or equal to"
<=I	Compare 16-bit integers for "less than or equal to"

The result of the comparison influences the result of logic operation (RLO) and the status bits CC 1 and CC 0. If the condition of the comparison is satisfied, the RLO is set to signal state "1". When the condition of the comparison is not met, the RLO is set to signal state "0".

The following table shows how the various comparison functions affect the RLO:

Comparison function	RLO if ACCU 2 > ACCU 1	RLO if ACCU 2 = ACCU 1	RLO if ACCU 2 < ACCU 1
==I	0	1	0
<>I	1	0	1
>I	1	0	0
<I	0	0	1
>=I	1	1	0
<=I	0	1	1

The setting of status bits CC 1 and CC 0 depends on the relationship between the two values involved in the comparison.

The contents of accumulators 1 and 2 are not changed by execution of the comparison functions. After the processing of a comparison function you can evaluate the result of logic operation with assignment and jump functions.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Input_1"              // Load value of the operand "Tag_Input_1" in
                             accumulator 1
L "Tag_Input_2"              // Shift the contents of accumulator 1 (Tag_Input_1)
                             to accumulator 2.
                             // Load value of the operand "Tag_Input_2" in
                             accumulator 1
>I                           // Compare whether the value of the operand
                             "Tag_Input_1" is greater than the value of the operand
                             "Tag_Input_2".
= "Tag_Output"               // Assign result of the comparison to the operand
                             "Tag_Output".
```

The following table shows how the instruction works using specific operand values:

Operand	Values		
Tag_Input_1	10	10	5
Tag_Input_2	5	10	10
Tag_Output	1	0	0

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

? D: Compare 32-bit integers

Description

Use the instruction "Compare 32-bit integers" to compare the contents of accumulator 2 (ACCU 2) with the contents of accumulator 1 (ACCU 1). The contents of accumulators 1 and 2 are interpreted as 32-bit integers. The instruction performs the comparison depending on the specified comparison function.

The following table shows the available comparison functions and their meaning:

Comparison function	Meaning
==D	Compare 32-bit integers for "equivalence"
<>D	Compare 32-bit integers for "non-equivalence"
>D	Compare 32-bit integers for "greater than"
<D	Compare 32-bit integers for "less than"
>=D	Compare 32-bit integers for "greater than or equal to"
<=D	Compare 32-bit integers for "less than or equal to"

The result of the comparison influences the result of logic operation (RLO) and the status bits CC 1 and CC 0. If the condition of the comparison is satisfied, the RLO is set to signal state "1". If the condition of the comparison is not met, the RLO is reset to signal state "0".

The following table shows how the various comparison functions affect the RLO:.

Comparison function	RLO if ACCU 2 > ACCU 1	RLO if ACCU 2 = ACCU 1	RLO if ACCU 2 < ACCU 1
==D	0	1	0
<>D	1	0	1
>D	1	0	0
<D	0	0	1
>=D	1	1	0
<=D	0	1	1

The setting of status bits CC 1 and CC 0 depends on the relationship between the two values involved in the comparison.

The contents of accumulators 1 and 2 are not changed by execution of the comparison functions. After the processing of a comparison function you can evaluate the result of logic operation with assignment and jump functions.

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Input_1"`	`// Load value of the operand "Tag_Input_1" in accumulator 1`

STL	Explanation
L "Tag_Input_2"	// Contents of accumulator 1 (Tag_Input_1) shifted to accumulator 2.
	// Load value of the operand "Tag_Input_2" in accumulator 1
>D	// Compare whether the value of the operand "Tag_Input_1" is greater than the value of the operand "Tag_Input_2".
= "Tag_Output"	// Assign result of the comparison to the operand "Tag_Output".

The following table shows how the instruction works using specific operand values:

Operand	Values		
Tag_Input_1	150 000	150 000	100 000
Tag_Input_2	100 000	150 000	150 000
Tag_Output	1	0	0

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

? R: Compare floating-point numbers

Description

Use the instruction "Compare floating-point numbers" to compare the contents of accumulator 2 (ACCU 2) with the contents of accumulator 1 (ACCU 1). The contents of accumulators 1 and 2 are interpreted as floating-point numbers. For the comparison, accumulators 1 and 2 must contain valid floating-point numbers. The instruction performs the comparison depending on the specified comparison function.

The following table shows the available comparison functions and their meaning:

Comparison function	Meaning
==R	Compare floating-point numbers for "equivalence"
<>R	Compare floating-point numbers for "non-equivalence"
>R	Compare floating-point numbers for "greater than"
<R	Compare floating-point numbers for "less than"

Comparison function	Meaning
>=R	Compare floating-point numbers for "greater than or equal to"
<=R	Compare floating-point numbers for "less than or equal to"

The result of the comparison influences the result of logic operation (RLO) and the status bits CC 1 and CC 0. If the condition of the comparison is satisfied, the RLO is set to signal state "1". If the condition of the comparison is not met, the RLO is reset to signal state "0".

The following table shows how the various comparison functions affect the RLO:

Comparison function	RLO if ACCU 2 > ACCU 1	RLO if ACCU 2 = ACCU 1	RLO if ACCU 2 < ACCU 1
==R	0	1	0
<>R	1	0	1
>R	1	0	0
<R	0	0	1
>=R	1	1	0
<=R	0	1	1

The setting of status bits CC 1 and CC 0 depends on the relationship between the two values involved in the comparison.

Note

During a comparison of invalid floating-point numbers, the CPU sets the result of logic operation to "0" and the status bits CC 0, CC 1, OVand OS to "1".

The contents of accumulators 1 and 2 are not changed by execution of the comparison functions. After the processing of a comparison function you can evaluate the result of logic operation with assignment and jump functions.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Input_1"	// Load value of the operand "Tag_Input_1" in accumulator 1
L "Tag_Input_2"	// Shift the contents of accumulator 1 (Tag_Input_1) to accumulator 2.
	// Load value of the operand "Tag_Input_2" in accumulator 1
>R	// Compare whether the value of the operand "Tag_Input_1" is greater than the value of the operand "Tag_Input_2".
= "Tag_Output"	// Assign result of the comparison to the operand "Tag_Output".

The following table shows how the instruction works using specific operand values:

Operand	Values		
Tag_Input_1	10 515	10 515	5 232
Tag_Input_2	5 232	10 515	10 515
Tag_Output	1	0	0

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Math functions

Integers

+I: Add integers (16 bit)

Description

Use the instruction "Add integers (16 bit)" to add the values in the right words of accumulators 1 and 2. The instruction interprets these values as 16-bit integers.

The instruction saves the sum in the right word of accumulator 1. The left word of accumulator 1 remains unchanged.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the sum is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

The contents of accumulator 2 remain unchanged after the instruction has executed.

Syntax

The following syntax is used for the "Add integers (16 bit)" instruction:

```
+I
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load first value for addition.
L "Tag_Value_2"	// Load second value for addition.
+I	// Add values
T "Tag_Result"	// Transfer sum to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

-I: Subtract integers (16 bit)

Description

Use the "Subtract integers (16 bit)" instruction to subtract the value of the right word of accumulator 1 from the value of the right word of accumulator 2. The instruction interprets the values of accumulators 1 and 2 as 16-bit integers.

The instruction saves the difference in the right word of accumulator 1. The left word of accumulator 1 remains unchanged.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the difference is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

The contents of accumulator 2 remain unchanged after the instruction has executed.

Syntax

The following syntax is used for the "Subtract integers (16 bit)" instruction:

```
-I
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load first value of subtraction into accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load second value for addition in accumulator 1.
-I	// Subtract value of accumulator 1 from value of accumulator 2.
T "Tag_Result"	// Transfer difference to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

*I: Multiply integers (16 bit)

Description

Use the instruction "Multiply integers (16 bit)" to multiply the values in the right words of accumulators 1 and 2. The instruction interprets these values as 16-bit integers.

The instruction saves the product in accumulator 1 as 32-bit integer (DINT).

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the product is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

The contents of accumulator 2 remain unchanged after the instruction has executed.

Syntax

The following syntax is used for the "Multiply integers (16 bit)" instruction:

```
*I
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load first value for multiplication.
L "Tag_Value_2"	// Load second value for multiplication.
*I	// Multiply values
T "Tag_Result"	// Transfer product to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

/I: Divide integers (16 bit)

Description

Use the "Divide integers (16 bit)" instruction to divide the value of the right word of accumulator 2 by the value of the right word of accumulator 1. The instruction interprets the values of the accumulators as 16-bit integers.

The instruction provides two results: the quotients and the remainder of the division. The quotient is the whole number result of the division. The instruction stores the whole number in the right word of accumulator 1. The instruction stores the remainder of the division in the left word of accumulator 1. If the dividend is negative, the remainder of the division is also negative.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the quotient is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

In the case of a division by zero, the instruction reports the value as quotient and in each case the value zero as remainder. In this case, status bits CC 0, CC 1, OV and OS are set to signal state "1".

The contents of accumulator 2 remain unchanged after the instruction has executed.

Syntax

The following syntax is used for the "Divide integers (16 bit)" instruction:

```
/I
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load dividend in accumulator 1.
L "Tag_Value_2"	// Move dividend to accumulator 2.
	// Load divisor in accumulator 1.
/I	// Divide the value of accumulator 2 by the value of accumulator 1.
T "Tag_Result"	// Transfer the result to operand "Tag_Result".

Before processing

The following table shows the contents of the right words in accumulators 1 and 2 before executing the "Divide integers (16 bit)" instruction:

Accumulator	Value
Accumulator 2	13
Accumulator 1	4

After processing

The following table shows the contents of accumulator 1 after executing the "Divide integers (16 bit)" instruction:

Accumulator 1	Value
Bit 0 to 15 (right word)	3
Bit 16 to 31 (left word)	1

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

+D: Add double integers (32 bit)

Description

Use the "Add double integers (32 bit)" instruction to add the values in accumulators 1 and 2. The instruction interprets these values as 32-bit integers. The instruction saves the sum in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the sum is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

The contents of accumulator 2 remain unchanged after the instruction has executed.

Syntax

The following syntax is used for the "Add double integers (32 bit)" instruction:

```
+D
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load first value for addition.
L "Tag_Value_2"	// Load second value for addition.
+D	// Add values
T "Tag_Result"	// Transfer sum to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

-D: Subtract double integers (32 bit)

Description

Use the "Subtract double integers (32 bit)" instruction to subtract the value of accumulator 1 from the value of the accumulator 2. The instruction interprets the values of accumulators 1 and 2 as as 32-bit integers. The instruction saves the result of the subtraction in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the difference is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

The contents of accumulator 2 remain unchanged after the instruction has executed.

Syntax

The following syntax is used for the "Subtract double integers (32 bit)" instruction:

```
-D
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load first value of subtraction into accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load the second value of subtraction into accumulator 1.
-D	// Subtract value of accumulator 1 from value of accumulator 2.
T "Tag_Result"	// Transfer difference to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

*D: Multiply double integers (32 bit)

Description

Use the "Multiply double integers (32 bit)" instruction to multiply the values in accumulators 1 and 2. The instruction interprets these values as 32-bit integers. The instruction saves the product in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the product is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

The contents of accumulator 2 remain unchanged after the instruction has executed.

Syntax

The following syntax is used for the "Multiply double integers (32 bit)" instruction:

```
*D
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load first value for multiplication.
L "Tag_Value_2"	// Load second value for multiplication.
*D	// Multiply values
T "Tag_Result"	// Transfer product to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

/D: Divide double integers (32 bit)

Description

Use the "Divide double integers (32 bit)" instruction to divide the value of accumulator 2 by the value of accumulator 1. The instruction interprets the values of accumulators 1 and 2 as as 32-bit integers. The instruction saves the result of the division (quotient) in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the quotient is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

In the case of a division by zero, the instruction reports the value zero as quotient. In this case, status bits CC 0, CC 1, OV and OS are set to signal state "1".

The contents of accumulator 2 remain unchanged after the instruction has executed.

Syntax

The following syntax is used for the "Divide double integers (32 bit)" instruction:

```
/D
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load dividend in accumulator 1.
L "Tag_Value_2"	// Move dividend to accumulator 2.
	// Load divisor in accumulator 1.
/D	// Divide the value of accumulator 2 by the value of accumulator 1.
T "Tag_Result"	// Transfer quotients to the operand "Tag_Result".

Before processing

The following table shows the contents of accumulators 1 and 2 before executing the "Divide double integers (32 bit)" instruction:

Accumulator	Value
Accumulator 2	13
Accumulator 1	4

After processing

The following table shows the contents of accumulator 1 after executing the "Divide double integers (32 bit)" instruction:

Accumulator	Value
Accumulator 1	3

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

+: Add constants

Description

Use the instruction "Add constants" to add the value of a specified constant to the contents of accumulator 1. You can specify the constant value as a 16-bit or 32-bit integer. The instruction stores the result in accumulator 1. In the case of addition of 16-bit integers, the instruction influences only the right word of accumulator 1.

To execute the addition of a 16-bit integer as DINT math function, enter "L#" before the constant. The "Add constants" instruction permits the use of negative numbers.

The instruction does not affect any status bits.

Syntax

Use the following syntax for the "Add constants" instruction:

```
+ <constant>
```

Parameters

The following table shows the parameters of the "Add constants" instruction:

Parameters	Data type	Description
<constant>	INT, DINT	Constants to be added

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1".
+ 25	// Increment the value of accumulator 1 by 25.
T "Tag_Result_1"	// Transfer contents of accumulator 1 to the operand "Tag_Result_1".
L "Tag_Value_2"	// Load value of the operand "Tag_Value_2" in accumulator 1
L "Tag_Value_3"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_3" in accumulator 1
+ -100	// Decrement the value of accumulator 1 by 100.
<I	// Check whether the value of accumulator 2 is less than the value of accumulator 1
JC NEXT	// If the result of the query is "1", jump to jump label NEXT .
	// If the result of the query is "0", execute the next instruction.
T "Tag_Result_2"	Transfer contents of accumulator 1 to the operand "Tag_Result_2".
L "Tag_Value_4"	// Load value of the operand "Tag_Value_4".
+ L#20	// Increment content of accumulator 1 by 20.
	// The addition is done as a DINT calculation
T "Tag_Result_3"	// Transfer contents of accumulator 1 to the operand "Tag_Result_3".
NEXT: L "Tag_Value_5"	// Jump label "NEXT"
	// Load value of the operand "Tag_Value_5".
+D	// Add values of accumulators 1 and 2.
T "Tag_Result_4"	// Transfer sum to the operand "Tag_Result_4".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

INC: Increment

Description

Use the "Increment" instruction to increment the value in accumulator 1 by the value specified as parameter of the instruction. The specified value can be in a value range from 0 to 255. The value is incremented only in the right byte of accumulator 1. No transfer takes place to be byte located to the left.

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The content of accumulator 2 remains unchanged.

If the parameter value is outside the permitted range of value, the result will be corrupted without generating an error message.

Syntax

Use the following syntax for the "Increment" instruction:

```
INC <constant>
```

Parameters

The following table shows the parameters of the "Increment" instruction:

Parameters	Format	Description
<constant>	8-bit integer	Value by which the value is incremented.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value"                // Load value of the operand in accumulator 1.
INC 5                        // Increment value of accumulator 1 by 5.
T "Tag_Result"               // Transfer contents of accumulator 1 to the operand.
```

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

DEC: Decrement

Description

Use the "Decrement" instruction to decrement the value in accumulator 1 by the value specified as parameter of the instruction. The specified value can be in a value range from 0 to 255. The value is decremented only in the right byte of accumulator 1. No transfer takes place to be byte located on the left side.

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The content of accumulator 2 remains unchanged.

If the parameter value is outside the permitted range of value, the result will be corrupted without generating an error message.

Syntax

Use the following syntax for the "Decrement" instruction:

```
DEC <constant>
```

Parameters

The following table shows the parameters of the "Decrement" instruction:

Parameters	Format	Description
<constant>	8-bit integer	Value by which the value is decremented.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value"                // Load value of the operand in accumulator 1.
DEC 5                        // Decrement the value of accumulator 1 by 5.
T "Tag_Result"               // Transfer contents of accumulator 1 to the operand.
```

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

MOD: Return remainder of division

Description

Use the "Return remainder of division" instruction to divide the value of accumulator 2 by the value of accumulator 1 and save the remainder of the division.

The instruction interprets the values of accumulators 1 and 2 as 32-bit integers. The instruction stores the remainder of the division in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the remainder of division is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

In the case of a division by zero, the instruction returns the value zero as result. In this case, status bits CC 0, CC 1, OV and OS are set to signal state "1".

The contents of accumulator 2 remain unchanged after the instruction has executed.

Syntax

Use the following syntax for the "Return remainder of division" instruction:

```
MOD
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load dividend in accumulator 1.
L "Tag_Value_2"	// Move dividend to accumulator 2.
	// Load divisor in accumulator 1.
MOD	// Divide the value of accumulator 2 by the value of accumulator 1.
T "Tag_Result"	// Transfer remainder of division to the operand "Tag_Result".

Before processing

The following table shows the contents of accumulators 1 and 2 before executing the instruction "Return remainder of division":

Accumulator	Value
Accumulator 2	13
Accumulator 1	4

After processing

The following table shows the contents of accumulator 1 after executing the instruction "Return remainder of division":

Accumulator	Value
Accumulator 1	1

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Floating-point numbers

+R: Add floating-point numbers

Description

Use the "Add floating-point numbers" instruction to add the values contained in accumulators 1 and 2. The instruction interprets these values as floating-point numbers. The instruction saves the sum in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the sum is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If an invalid floating-point number or the sign for infinity (+/- ∞) is entered, the instruction writes an invalid value to accumulator 1 and sets the status bits CC 0, CC 1, OV and OS to "1".

The contents of accumulator 2 remain unchanged after the instruction has executed.

Syntax

The following syntax is used for the instruction "Add floating-point numbers":

```
+R
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load first value for addition.
L "Tag_Value_2"	// Load second value for addition.
+R	// Add values
T "Tag_Result"	// Transfer sum to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

-R: Subtract floating-point numbers

Description

Use the "Subtract floating-point numbers" instruction to subtract the value of accumulator 1 from the value of accumulator 2. The instruction interprets the values of accumulators 1 and 2 as floating-point numbers. The instruction stores the difference in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the difference is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If an invalid floating-point number or the sign for infinity (+/- ∞) is entered, the instruction writes an invalid value to accumulator 1 and sets the status bits CC 0, CC 1, OV and OS to "1".

The contents of accumulator 2 remain unchanged after the instruction has executed.

Syntax

Use the following syntax for the instruction "Subtract floating-point numbers":

```
-R
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load first value of subtraction into accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load the second value of subtraction into accumulator 1.
-R	// Subtract value of accumulator 1 from value of accumulator 2.
T "Tag_Result"	// Transfer difference to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

*R: Multiply floating-point numbers

Description

Use the "Multiply floating-point numbers" instruction to multiply the values contained in accumulators 1 and 2. The instruction interprets these values as floating-point numbers. The instruction saves the product in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the product is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If an invalid floating-point number or the sign for infinity (+/- ∞) is entered, the instruction writes an invalid value to accumulator 1 and sets the status bits CC 0, CC 1, OV and OS to "1".

The contents of accumulator 2 remain unchanged after the instruction has executed.

Syntax

Use the following syntax for the "Multiply floating-point numbers" instruction:

```
*R
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load first value for multiplication.
L "Tag_Value_2"	// Load second value for multiplication.
*R	// Multiply values
T "Tag_Result"	// Transfer product to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

/R: Divide floating-point numbers

Description

Use the "Divide floating-point numbers" instruction to divide the value of accumulator 2 by the value of accumulator 1. The instruction interprets the values of accumulators 1 and 2 as floating-point numbers. The instruction saves the result of the division (quotient) in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the quotient is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

In the case of a division by zero, the instruction reports the value zero as quotient. In this case, status bits CC 0, CC 1, OV and OS are set to signal state "1".

If an invalid floating-point number or the sign for infinity (+/- ∞) is entered, the instruction writes an invalid value to accumulator 1 and sets the status bits CC 0, CC 1, OV and OS to "1".

The contents of accumulator 2 remain unchanged after the instruction has executed.

Syntax

Use the following syntax for the "Divide floating-point numbers" instruction:

```
/R
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load dividend in accumulator 1.
L "Tag_Value_2"	// Move dividend to accumulator 2.
	// Load divisor in accumulator 1.
/R	// Divide the value of accumulator 2 by the value of accumulator 1.
T "Tag_Result"	// Transfer quotients to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

ABS: Form absolute value

Description

Use the "Form absolute value" instruction to form the absolute value of the value in accumulator 1. The instruction interprets the contents of accumulator 1 as a floating-point number and sets the sign of the mantissa to "0". The result is stored in accumulator 1.

The "Form absolute value" instruction does not affect any status bits.

Syntax

Use the following syntax for the "Form absolute value" instruction:

```
ABS
```

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value"                // Load floating-point number into accumulator 1
ABS                          // Form absolute value
T "Tag_Result"               // Transfer the result to operand "Tag_Result".
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	-1.5E+02
Tag_Result	1.5E+02

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Advanced floating-point numbers

SQR: Form square

Description

Use the "Form square" instruction to calculate the square of a floating-point number located in accumulator 1. The result is stored in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If the floating-point numbers are invalid, the instruction sets the status bits CC 0, CC 1, OV and OS to "1".

The instruction influences only the contents of accumulator 1. The contents of accumulator 2 remain unchanged.

Syntax

Use the following syntax for the "Form square" instruction:

```
SQR
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
SQR	// Form square
	// Store result in accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

SQRT: Form square root

Description

Use the "Form square root" instruction to form the square root of the floating-point number located in accumulator 1. The result is stored in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If accumulator 1 has a value less than zero or an invalid floating-point number, the instruction writes an invalid value to accumulator 1 and sets the status bits CC 0, CC 1, OV and OS to "1".

If accumulator 1 has the value "0", the instruction returns the value "0".

The instruction influences only the contents of accumulator 1. The contents of accumulator 2 remain unchanged.

Syntax

Use the following syntax for the "Form square root" instruction:

```
SQRT
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
+R	// Add the values of accumulator 1 and 2.
	// Store sum in accumulator 1.
SQRT	// Get square root
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

EXP: Form exponential value

Description

Use the "Form exponential value" to form the exponent from the base e (e = 2.718282) and the floating-point number in accumulator 1. The instruction stores the result in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If the floating-point numbers are invalid, the instruction sets the status bits CC 0, CC 1, OV and OS to "1".

The instruction influences only the contents of accumulator 1. The contents of accumulator 2 remain unchanged.

Syntax

Use the following syntax for the "Form exponential value" instruction:

```
EXP
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
EXP	// Form exponential value
	// Store result in accumulator 1.
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

LN: Form natural logarithm

Description

Use the "Form natural logarithm" instruction to calculate the natural logarithm of the value in accumulator 1 to base e (e = 2.718282).

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If accumulator 1 has a value less than or equal to zero, the instruction writes an invalid value to accumulator 1 and sets the status bits CC 0, CC 1, OV and OS to "1". If the floating-point numbers are invalid, the instruction also sets the status bits CC 0, CC 1, OV and OS to "1".

The instruction influences only the contents of accumulator 1. The contents of accumulator 2 remain unchanged.

Syntax

Use the following syntax for the "Form natural logarithm" instruction:

```
LN
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
LN	// Form natural logarithm.
	// Store result in accumulator 1.
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

SIN: Form sine value

Description

Use the "Form sine value" instruction to calculate the sine of the angle in accumulator 1. The angle must be specified in radian measure and exist as floating-point number in accumulator 1. The instruction stores the result in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1". If the floating-point numbers are invalid, the instruction sets the status bits CC 0, CC 1, OV and OS to "1".

The instruction influences only the contents of accumulator 1. The contents of accumulator 2 remain unchanged.

Syntax

Use the following syntax for the "Form sine value" instruction:

```
SIN
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
SIN	// Form sine value
	// Store result in accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

ASIN: Form arcsine value

Description

Use the "Form arcsine value" instruction to calculate the size of the angle from the value in accumulator 1 that corresponds to this value. The value of the accumulator 1 must exist as floating-point number in a value range from "-1" to "1".

The result of the instruction is returned in radian measure and stored in accumulator 1. Depending on the value of accumulator 1, the result can be in a value range from $-\pi/2$ to $+\pi/2$ (π= 3.14159).

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If accumulator 1 has a value that exceeds the permitted value range, the instruction writes an invalid floating-point number to accumulator 1 and sets the status bits CC 0, CC 1, OV and OS to "1". If the floating-point numbers are invalid, the instruction also sets the status bits CC 0, CC 1, OV and OS to "1".

The instruction influences only the contents of accumulator 1. The contents of accumulator 2 remain unchanged.

Syntax

Use the following syntax for the "Form arcsine value" instruction:

```
ASIN
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
ASIN	// Form arcsine value
	// Store result in accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.

STL	Explanation
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

COS: Form cosine value

Description

Use the "Form cosine value" instruction to calculate the cosine of the angle in accumulator 1. The size of the angle must be specified in radian measure and exist as floating-point number in accumulator 1. The instruction stores the result in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1". If the floating-point numbers are invalid, the instruction sets the status bits CC 0, CC 1, OV and OS to "1".

The instruction influences only the contents of accumulator 1. The contents of accumulator 2 remain unchanged.

Syntax

Use the following syntax for the "Form cosine value" instruction:

```
COS
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
COS	// Form cosine value
	// Store result in accumulator 1.

STL	Explanation
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

ACOS: Form arccosine value

Description

Use the "Form arccosine value" instruction to calculate the size of the angle from the value in accumulator 1 that corresponds to this value. The value of the accumulator 1 must exist as floating-point number in a value range from "-1" to "1".

The result of the instruction is returned in radian measure and stored in accumulator 1. Depending on the value of accumulator 1, the result can be in a value range from "0" to "π" (π= 3.14159).

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If accumulator 1 has a value that exceeds the permitted value range, the instruction writes an invalid floating-point number to accumulator 1 and sets the status bits CC 0, CC 1, OV and OS to "1". If the floating-point numbers are invalid, the instruction also sets the status bits CC 0, CC 1, OV and OS to "1".

The instruction influences only the contents of accumulator 1. The contents of accumulator 2 remain unchanged.

Syntax

Use the following syntax for the "Form arccosine value" instruction:

```
ACOS
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
ACOS	// Form arccosine value
	// Store result in accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

TAN: Form tangent value

Description

Use the "Form tangent value" instruction to calculate the tangent of the angle in accumulator 1. The size of the angle must be specified in radian measure and exist as floating-point number in accumulator 1. The instruction stores the result in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status

bits OV and OS are set to "1". If the floating-point numbers are invalid, the instruction sets the status bits CC 0, CC 1, OV and OS to "1".

The instruction influences only the contents of accumulator 1. The contents of accumulator 2 remain unchanged.

Syntax

Use the following syntax for the "Form tangent value" instruction:

```
TAN
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
TAN	// Form tangent value
	// Store result in accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

ATAN: Form arctangent value

Description

Use the "Form arctangent value" instruction to calculate the size of the angle from the value in accumulator 1 that corresponds to this value. The value of the accumulator 1 must exist as floating-point number.

The result of the instruction is returned in radian measure and stored in accumulator 1. Depending on the value of accumulator 1, the result can be in a value range from $-\pi/2$ to $+\pi/2$ (π= 3.14159).

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If accumulator 1 has a value that exceeds the permitted value range, the instruction writes an invalid floating-point number to accumulator 1 and sets the status bits CC 0, CC 1, OV and OS to "1". If the floating-point numbers are invalid, the instruction also sets the status bits CC 0, CC 1, OV and OS to "1".

The instruction influences only the contents of accumulator 1. The contents of accumulator 2 remain unchanged.

Syntax

Use the following syntax for the instruction "Form arctangent value":

```
ATAN
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
ATAN	// Form arctangent value
	// Store result in accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)
Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)
Evaluating errors with output parameter RET_VAL (Page 5370)
Call function blocks as single or multi-instances (Page 2182)
Basic information on the status word (Page 2466)
Querying and setting status bits in STL (Page 2467)

Load and transfer

Load

L: Loading

Description

The "Load" instruction loads the content of a specified operand in accumulator 1. The operand to be loaded can be addressed as a byte, word or double word in one of the following memory areas:

- Process image of the inputs and outputs (I, Q)
- Bit memory (M)
- Temporary local data (L)
- Data blocks (DB, DI)
- Pointer
- I/O (PI)
- Timer (T)
- Counter (C)

The memory area of accumulator 1 is organized in bytes and is 32 bits wide.

The instruction writes the contents of an operand that is to be loaded in byte format right aligned into accumulator 1. The remaining bytes in accumulator 1 are filled with "0".

The instruction writes the contents of an operand that is to be loaded in word format into the right word of accumulator 1. The higher addressed byte is hereby transferred into the right byte (bits 0 to 7) of accumulator 1. The lower addressed byte is written to the left of it. The remaining bytes in the left word of accumulator 1 are filled with "0".

The instruction writes the contents of an operand that is to be loaded in double word format into the 32 bits of accumulator 1. The highest addressed byte is hereby transferred to the right byte (bits 0 to 7) of accumulator 1. The lowest addressed byte is written in bits 24 to 31.

The following table shows, as an example, how the contents of accumulator 1 are changed by the loading of operands in byte, word and double word format:

Instruction	Accumulator 1							
	31 . . .	. . . 24	23 . . .	. . . 16	15 . . .	. . . 8	7 . . .	. . . 0
Loading of a byte: L MB10	0000	0000	0000	0000	0000	0000	<MB10>	
Loading of a word: L MW10	0000	0000	0000	0000	<MB10>		<MB11>	
Load a double word: L MD10	<MB10>		<MB11>		<MB12>		<MB13>	

During the loading of the specified operand the previous contents of accumulator 1 are moved to accumulator 2. The instruction "Load" hereby transfers the complete contents of accumulator 1 to accumulator 2. The previous contents of accumulator 2 are lost in the process.

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

Syntax

Use the following syntax for the "Load" instruction:

```
L <Operand>
```

Parameters

The following table shows the parameters of the "Load" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT, timers, REAL, DATE, TOD, CHAR	I, Q, PI, M, L, DB, DI, T, C, pointer, parameter	Operand whose contents are loaded.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1

STL	Explanation
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.
L "Tag_Value_3"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_3" in accumulator 1
+R	// Add the values of accumulator 1 and 2.
	// Store sum in accumulator 1.
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

L STW: Load status word in accumulator 1

Description

Use the "Load status word in accumulator 1" instruction to load the status word in accumulator 1. When it executes, the instruction writes the individual status bits to the right word of accumulator 1. The remaining bits of accumulator 1 are written with "0".

The following table shows the contents of the accumulator after executing the "Load status word in accumulator 1" instruction:

	Accumulator 1									
Bit number	31-9	8	7	6	5	4	3	2	1	0
Content	0	BR	CC 1	CC 0	OV	OS	0	0	RLO	0

The instruction is always executed independently of the signal state of the status bit.

Syntax

Use the following syntax for the "Load status word in accumulator 1" instruction:

```
L STW
```

Example

The following example shows how the instruction works:

STL	Explanation
L STW	// Load status word in accumulator 1.
T "Tag_STW"	// Transfer the contents of accumulator 1 (status bits) to the operand "Tag_STW".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

LAR1: Load AR1 with contents of accumulator 1

Description

Use the "Load AR1 with contents of accumulator 1" instruction to load address register 1 (AR1) with the contents of accumulator 1. The value of accumulator 1 must match the format of an area pointer (POINTER).

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Syntax

Use the following syntax for the "Load AR1 with contents of accumulator 1" instruction:

```
LAR1
```

Example

The following example shows how the instruction works:

STL	Explanation
L P#10.0	// Load pointer (P#10.0) in accumulator 1.
LAR1	// Load address register 1 with contents of accumulator 1.
L MW [AR1,P#4.0]	// Load MW14 in accumulator 1.
L "Tag_Value"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load content of the operand "Tag_Value" in accumulator 1.
>I	// Compare whether the value of accumulator 2 is greater than the value of accumulator 1.
A I [AR1,P#2.1]	// Check whether signal state of bit I12.1 is "1" and AND with current RLO.
= "Tag_Output"	// If the condition is satisfied (RLO = "1"), set the operand "Tag_Output" to "1".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

LAR1 <D>: Load AR1 with double word or area pointer

Description

Use the "Load AR1 with double word or area pointer" instruction to load address register 1 (AR1) with the contents of a double word or an area internal or area crossing pointer. The contents of the double word must match the format of an area pointer.

The pointer and the double word can address one of the following memory areas:

- Bit memory (M)
- Temporary local data (L)
- Data blocks (DB, DI)

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Syntax

Use the following syntax for the instruction "Load AR1 with double word or area pointer":

```
LAR1 <D>
```

Parameters

The following table shows the parameters of the "Load AR1 with double word or area pointer" instruction:

Parameters	Declaration	Data type	Memory area	Description
<D>	Input	DWORD, POINTER	D, M, L	Operand whose contents are loaded.

Example

The following example shows how the instruction works:

STL	Explanation
LAR1 P#10.0	// Load address register 1 with contents of the area internal pointer P#10.0.
L MW [AR1,P#4.0]	// Load MW14 in accumulator 1.
L "Tag_Value"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load content of the operand "Tag_Value" in accumulator 1.
>I	// Compare whether the value of accumulator 2 is greater than the value of accumulator 1.
A I [AR1,P#2.1]	// Check whether signal state of bit I12.1 is "1" and AND with current RLO.
= "Tag_Output"	// If the condition is satisfied (RLO = "1"), set the operand "Tag_Output" to "1".
LAR1 MD20	// Load address register 1 with contents of the MD20 (MD20 = P#30.0)
A I [AR1,P#2.1]	// Check whether signal state of bit I32.1 is "1" and AND with current RLO.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

LAR1 AR2: Load AR1 with contents of AR2

Description

Use the "Load AR1 with contents of AR2" instruction to load the contents of address register 2 (AR2) into address register 1 (AR1).

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Syntax

Use the following syntax for the "Load AR1 with contents of AR2" instruction:

```
LAR1 AR2
```

Example

The following example shows how the instruction works:

STL	Explanation
LAR2 P#20.0	// Load address register 2 with contents of the area internal pointer P#20.0.
LAR1 AR2	// Copy contents of address register 2 into address register 1.
L MW [AR1,P#4.0]	// Load MW24 in accumulator 1.
T "Tag_2"	// Transfer accumulator 1 to "Tag_2".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

LAR2: Load AR2 with contents of accumulator 1

Description

Use the "Load AR2 with contents of accumulator 1" instruction to load address register 2 (AR2) with the contents of accumulator 1. The value of accumulator 1 must match the format of an area pointer (POINTER).

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Syntax

Use the following syntax for the "Load AR2 with contents of accumulator 1" instruction:

```
LAR2
```

Example

The following example shows how the instruction works:

STL	Explanation
L P#10.0	// Load pointer (P#10.0) in accumulator 1.
LAR2	// Load address register 2 with contents of accumulator 1.
L MW [AR2,P#4.0]	// Load MW14 in accumulator 1.
L "Tag_Value"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load content of the operand "Tag_Value" in accumulator 1.
>I	// Compare whether the value of accumulator 2 is greater than the value of accumulator 1.
A I [AR2,P#2.1]	// Check whether signal state of bit I12.1 is "1" and AND with current RLO.
= "Tag_Output"	// If the condition is satisfied (RLO = "1"), set the operand "Tag_Output" to "1".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

LAR2 <D>: Load AR2 with double word or area pointer

Description

Use the "Load AR2 with double word or area pointer" instruction to load address register 2 (AR2) with the contents of a double word or an area-internal or cross-area pointer. The contents of the double word must match the format of an area pointer (POINTER).

The pointer and the double word can address one of the following memory areas:

- Bit memory (M)
- Temporary local data (L)
- Data blocks (DB, DI)

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Syntax

Use the following syntax for the "Load AR2 with double word or area pointer" instruction:

```
LAR2 <D>
```

Parameters

The following table shows the parameters of the "Load AR2 with double word or area pointer" instruction:

Parameters	Declaration	Data type	Memory area	Description
<D>	Input	DWORD, POINTER	D, M, L	Operand whose contents are loaded.

Example

The following example shows how the instruction works:

```
STL                          Explanation
LAR2 P#10.0                  // Load address register 2 with contents of the area
                             internal pointer P#10.0.
L MW [AR2,P#4.0]             // Load MW14 in accumulator 1.
```

STL	Explanation
L "Tag_Value"	// Shift the contents of accumulator 1 to accumulator 2. // Load content of the operand "Tag_Value" in accumulator 1.
>I	// Compare whether the value of accumulator 2 is greater than the value of accumulator 1.
A I [AR2,P#2.1]	// Check whether signal state of bit I12.1 is "1" and AND with current RLO.
= "Tag_Output"	// If the condition is satisfied (RLO = "1"), set the operand "Tag_Output" to "1".
LAR2 MD20	// Load address register 2 with contents of the MD20 (MD20 = P#30.0)
A I [AR2,P#2.1]	// Check whether signal state of bit I32.1 is "1" and AND with current RLO.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Transfer

T: Transfer

Description

Use the "Transfer" instruction to transfer the contents of accumulator 1 in byte, word or double word format to a specified operand. The contents of accumulator 1 are not changed by this instruction. The specified operand must be addressed in one of the following memory areas:

- Process image of the inputs and outputs (I, Q)
- Bit memory (M)
- Temporary local data (L)
- Data blocks (DB, DI)
- I/O (PQ)

The number of bytes to be transferred depends on the format of the specified operand. If the operand exists in byte format, the instruction transfers the contents of the bits 0 to 7 of accumulator 1.

If the operand exists in word format, the instruction transfers the contents of the word that is right-aligned in accumulator 1. The right byte of accumulator 1 is transferred to the higher addressed byte of the operand. The bits 8 to 15 of the accumulator 1 are transferred to the lower byte of the operand.

If the specified operand is available in double word format, the instruction transfers the entire contents of accumulator 1. The right byte of accumulator 1 is transferred to the highest addressed byte of the operand. The bits 24 to 31 of accumulator 1 are transferred to the lowest addressed byte of the operand.

The following table shows an example of how the contents of accumulator 1 are transferred in byte, word and double word format:

Instruction	Accumulator 1			
Bits	31 24	23 16	15 8	7 0
Contents of accumulator 1	byte (n)	byte (n+1)	byte (n+2)	byte (n+3)
Transfer of a byte				byte (n)
Transfer of a word			byte (n)	byte (n+1)
Transferring a double word	byte (n)	byte (n+1)	byte (n+2)	byte (n+3)

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

Syntax

Use the following syntax for the "Transfer" instruction:

```
T <Operand>
```

Parameters

The following table shows the parameters of the "Transfer" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BYTE, WORD, DWORD, SINT, INT, DINT, USINT, UINT, UDINT, timers, REAL, DATE, TOD, CHAR	I, Q, PQ, M, L, DB, DI	Operand to which the contents of accumulator 1 are transferred.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.
T "Tag_Result_1"	// Transfer product (contents of Accumulator 1) to the operand "Tag_Result_1".
L "Tag_Value_3"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_3" in accumulator 1
+R	// Add the values of accumulator 1 and 2.
	// Store sum in accumulator 1.
T "Tag_Result_2"	// Transfer sum (contents of Accumulator 1) to the operand "Tag_Result_2"

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

T STW: Transfer accumulator 1 to status word

Description

Use the "Transfer accumulator 1 to status word" instruction to transfer bits 0 to 8 of accumulator 1 to the status word.

The following table shows the bits in accumulator 1 that will overwrite status bits with processing of the instruction:

	Accumulator 1								
Bit number	8	7	6	5	4	3	2	1	0
Status bits	BR	CC 1	CC 0	OV	OS	0	0	RLO	0

The instruction is always executed independently of the signal state of the status bit.

Syntax

Use the following syntax for the "Transfer accumulator 1 to status word" instruction:

```
T STW
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_STW"	// Load value of the operand "Tag_STW" in accumulator 1.
T STW	// Transfer bits 0 to 8 of accumulator 1 to status word

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

CAR: Switch AR1 and AR2

Description

Use the instruction "Switch AR1 and AR2" to switch the contents of address registers 1 (AR1) and 2 (AR2).

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Syntax

Use the following syntax for the "Switch AR1 and AR2" instruction:

```
CAR
```

Example

The following example shows how the instruction works:

STL	Explanation
LAR1 P#10.0	// Load address register 1 with contents of the area internal pointer P#10.0.
LAR2 P#20.0	// Load address register 2 with contents of the area internal pointer P#20.0.
CAR	// Switch contents of the address registers.
L MD [AR1,P#2.0]	// Load MD22 in accumulator 1.
T MD [AR2,P#2.0]	// Transfer MD12 too accumulator 2.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

TAR1: Transfer AR1 to accumulator 1

Description

Use the "Transfer AR1 to accumulator 1" instruction to transfer the contents of address register 1 (AR1) to accumulator 1. The contents of accumulator 1 are shifted to accumulator 2.

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

Syntax

Use the following syntax for the "Transfer AR1 to accumulator 1" instruction:

```
TAR1
```

Example

The following example shows how the instruction works:

STL	Explanation
LAR1 P#15.0	// Load contents of the area-internal pointer P#15.0 into address register 1.
TAR1	// Transfer contents of address register 1 (P#15.0) to accumulator 1.
T "Tag_Pointer"	// Transfer contents of accumulator 1 to the operand "Tag_Pointer".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

TAR1 <D>: Transfer AR1 to double word

Description

Use the "Transfer AR1 to double word" instruction to transfer the contents of address register 1 (AR1) to a double word. The double word must hereby be addressed in one of the following memory areas:

- Bit memory (M)
- Temporary local data (L)
- Data blocks (DB, DI)

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Syntax

Use the following syntax for the "Transfer AR1 to double word" instruction:

```
TAR1 <D>
```

Parameters

The following table shows the parameters of the "Transfer AR1 to double word" instruction:

Parameters	Declaration	Data type	Memory area	Description
<D>	Output	DWORD	D, M, L	Operand to which the contents of accumulator 1 are transferred.

Example

The following example shows how the instruction works:

```
STL                          Explanation
TAR1 %DBD20                  // Transfer contents of address register 1 to data
                             double word DBD20.
TAR1 %DID30                  // Transfer contents of address register 1 to instance
                             double word DID30.
TAR1 %LD18                   // Transfer contents of address register 1 to local
                             data double word LD18.
TAR1 %MD24                   // Transfer contents of address register 1 to memory
                             double word MD24.
```

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

TAR1 AR2: Transfer AR1 to AR2

Description

Use the "Transfer AR1 to AR2" instruction to copy the contents of address register 1 (AR1) to address register 2 (AR2).

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Syntax

Use the following syntax for the "Transfer AR1 to AR2" instruction:

```
TAR1 AR2
```

Example

The following example shows how the instruction works:

STL	Explanation
LAR1 P#10.4	// Load contents of the area-internal pointer P#10.4 into address register 1.
TAR1 AR2	// Transfer contents of address register 1 to address register 2.
A I [AR2,P#3.2]	// Check whether signal state of bit I13.6 is "1" and AND with current RLO.
= "Tag1"	// Set operand "Tag_1" to "1"

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

TAR2: Transfer AR2 to accumulator 1

Description

Use the "Transfer AR2 to accumulator 1" instruction to transfer the contents of address register 2 (AR2) to accumulator 1. The contents of accumulator 1 are shifted to accumulator 2.

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

Syntax

Use the following syntax for the "Transfer AR2 to accumulator 1" instruction:

```
TAR2
```

Example

The following example shows how the instruction works:

STL	Explanation
LAR2 P#20.0	// Load address register 2 with contents of the area internal pointer P#20.0.
TAR2	// Transfer contents of address register 2 (P#20.0) to accumulator 1.
T "Tag_Pointer"	// Transfer contents of accumulator 1 to the operand "Tag_Pointer".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

TAR2 <D>: Transfer AR2 to double word

Description

Use the "Transfer AR2 to double word" instruction to transfer the contents of address register 2 (AR2) to a double word. The double word must hereby be addressed in one of the following memory areas:

- Bit memory (M)
- Temporary local data (L)
- Data blocks (DB, DI)

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Syntax

Use the following syntax for the "Transfer AR2 to double word" instruction:

```
TAR2 <D>
```

Parameters

The following table shows the parameters of the "Transfer AR2 to double word" instruction:

Parameters	Declaration	Data type	Memory area	Description
<D>	Output	DWORD	D, M, L	Operand to which the contents of accumulator 2 are transferred.

Example

The following example shows how the instruction works:

```
STL                          Explanation
TAR2 %DBD20                  // Transfer contents of address register 2 to data
                             double word DBD20.
TAR2 %DID30                  // Transfer contents of address register 2 to instance
                             double word DID30.
TAR2 %LD18                   // Transfer contents of address register 2 to local
                             data double word LD18.
TAR2 %MD24                   // Transfer contents of address register 2 to memory
                             double word MD24.
```

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Conversion operations

BTI: Convert BCD to integer (16 bit)

Description

You can use the "Convert BCD to integer (16 bit)" instruction to convert the value in the right word of accumulator 1 to a 16-bit integer. The instruction interprets the word that is to be converted as a three-digit binary coded decimal (BCD).

The bits 1 to 11 of accumulator 1 indicate the value that will be converted. Values in the range from "-999" to "+999" are permitted.

The sign of the result value is read from bit 15 of accumulator 1. If the signal state of the bit is "0", the sign is positive. If the signal state of the bit is "1," the sign is negative. Bits 12 through 14 are not used in the conversion.

If the BCD code contains a BCD error, the CPU sets the OV and the OS bit in the status word.

The result of the conversion is saved in the right word of accumulator 1. The contents of the left word in accumulator 1 are not affected by the instruction.

Syntax

The following syntax is used for the "Convert BCD to integer (16 bit)" instruction:

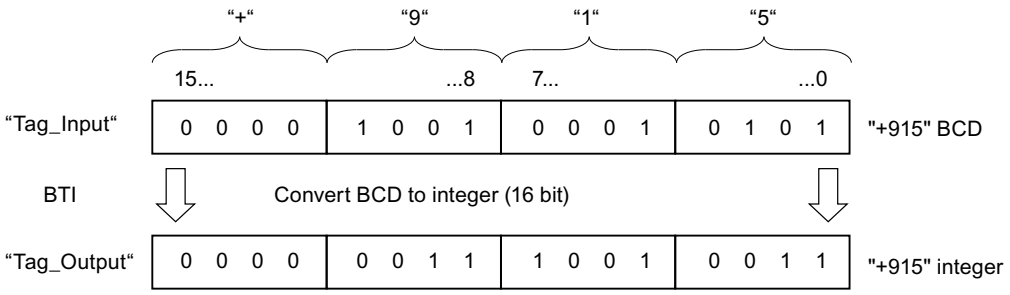
```
BTI
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Input"	// Load BCD in accumulator 1.
BTI	// Convert BCD to 16-bit integer.
T "Tag_Output"	// Transfer the result to operand "Tag_Output".

The following figure shows how the instruction works using specific operand values:



See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

ITB: Convert integer (16 bit) to BCD

Description

Use the "Convert integer (16 bit) to BCD" instruction to convert the value in the right word of accumulator 1 to a three-digit binary-coded decimal (BCD). The instruction interprets the value that is to be converted as a 16-bit integer.

The bits 0 to 11 of accumulator 1 indicate the value that will be converted. Values in the range from "-999" to "+999" are permitted. If the value to be converted is outside this range, the bits OV and OS are set to signal state "1". In this case the conversion is not executed.

The sign of the result value is read from bits 12 through 15 of accumulator 1. If the signal state of the bits is "0", the sign is positive. If the signal state of all four bits is "1", the sign is negative.

The result of the conversion is saved in the right word of accumulator 1. The contents of the left word in accumulator 1 are not affected by the instruction.

Syntax

Use the following syntax for the "Convert integer (16 bit) to BCD" instruction:

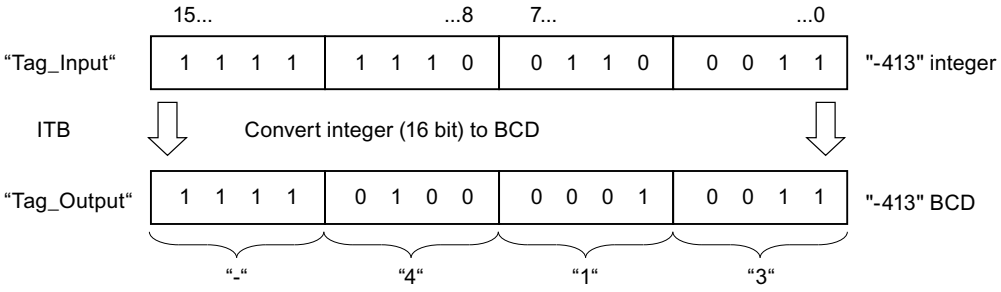
```
ITB
```

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Input"`	`// Load integer (16 bit) in accumulator 1.`
`ITB`	`// Convert integer to BCD.`
`T "Tag_Output"`	`// Transfer the result to operand "Tag_Output".`

The following figure shows how the instruction works using specific operand values:



See also

Overview of the valid data types (Page 2236)
Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)
Evaluating errors with output parameter RET_VAL (Page 5370)
Call function blocks as single or multi-instances (Page 2182)
Basic information on the status word (Page 2466)
Querying and setting status bits in STL (Page 2467)

BTD: Convert BCD to integer (32 bit)

Description

You can use the "Convert BCD to integer (32 bit)" to convert the value in accumulator 1 to a 32-bit integer. The instruction interprets the value that is to be converted as a seven-digit binary coded binary coded decimal (BCD).

The bits 0 to 27 of accumulator 1 indicate the value that will be converted. Values in the range from "-9999999" through "+9999999" are permitted.

The sign of the result value is read from bit 31 of accumulator 1. If the signal state of the bit is "0", the sign is positive. If the signal state of the bit is "1," the sign is negative. Bits 28 through 30 are not used in the conversion.

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

If the BCD code contains a BCD error, the CPU sets the OV and the OS bit in the status word.

Syntax

The following syntax is used for the "Convert BCD to integer (32 bit)" instruction:

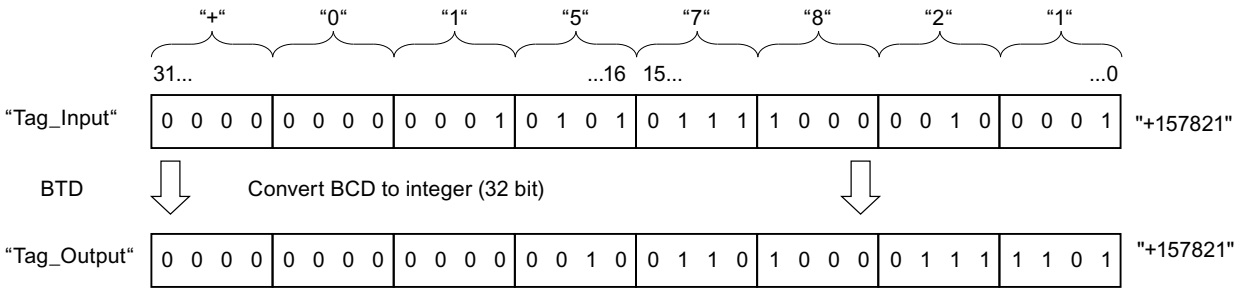
```
BTD
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Input"	// Load BCD in accumulator 1.
BTD	// BCD is converted to 32-bit integer.
T "Tag_Output"	// Transfer the result to operand "Tag_Output".

The following figure shows how the instruction works using specific operand values:



See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

ITD: Convert integer (16 bit) to integer (32 bit)

Description

Use the "Convert integer (16 bit) to integer (32 bit)" instruction to convert the value in the right word of accumulator 1 to a 32-bit integer. The instruction interprets the value that is to be converted as a 16-bit integer.

Bits 0 through 15 are not used in the conversion. The bits 16 through 31 of the result value are filled with the signal state of the bit 15.

The result value is saved in accumulator 1. The content of accumulator 2 remains unchanged.

Syntax

Use the following syntax for the "Convert integer (16 bit) to integer (32 bit)" instruction:

```
ITD
```

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Input"`	`// Load integer (16 bit) in accumulator 1.`

STL	Explanation
ITD	// Convert integer (16 bit) to integer (32 bit).
T "Tag_Output"	// Transfer the result to operand "Tag_Output".

The following table shows the contents of accumulator 1 before and after processing of the instruction:

Status	Accumulator 1							
	31 . . .	. . . .	. . . .	. . . 16	15 . . .	. . . .	. . . .	. . . 0
Before processing	XXXX	XXXX	XXXX	XXXX	1111	1111	1111	0110
After processing	1111	1111	1111	1111	1111	1111	1111	0110
	X: The signal state of the bits is irrelevant for the conversion							

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

DTB: Convert integer (32 bit) to BCD

Description

Use the "Convert integer (32 bit) to BCD" instruction to convert the value in accumulator 1 to a seven-digit binary coded decimal (BCD). The instruction interprets the value that is to be converted as a 32-bit integer.

The bits 0 to 27 of accumulator 1 indicate the value that will be converted. Values in the range from "-9999999" through "+9999999" are permitted. If the value to be converted is outside this range, the bits OV and OS are set to signal state "1". In this case the conversion is not executed.

The sign of the result value is read from bits 28 through 31 of accumulator 1. If the signal state of the bits is "0", the sign is positive. If the signal state of all four bits is "1", the sign is negative.

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

Syntax

Use the following syntax for the "Convert integer (32 bit) to BCD" instruction:

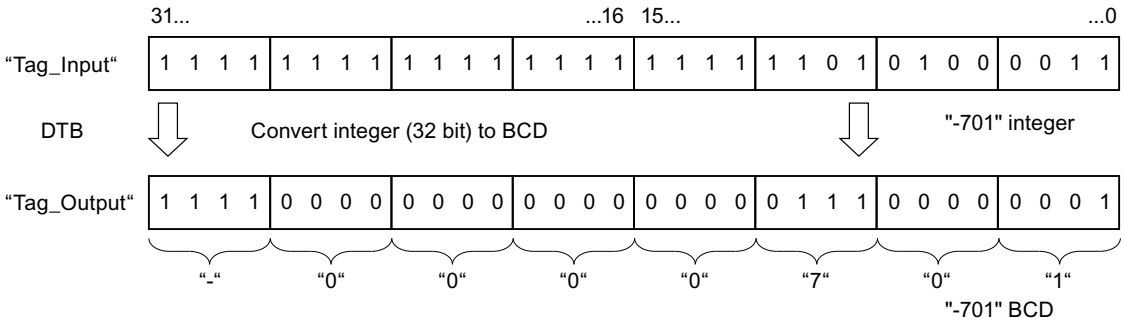
```
DTB
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Input"	// Load integer (32 bit) to accumulator 1
DTB	// Convert integer (32 bit) to BCD
T "Tag_Output"	// Transfer the result to operand "Tag_Output".

The following figure shows how the instruction works using specific operand values:



See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

DTR: Convert integer (32 bit) to floating-point number

Description

Use the "Convert integer (32 bit) to floating-point number" instruction to convert the value in accumulator 1 to a floating-point number. The instruction interprets the value that is to be converted as a 32-bit integer.

Because a 32-bit integer has a higher accuracy than a floating-point number, the result of the conversion is rounded off to the next representable number.

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

Syntax

Use the following syntax for the "Convert integer (32 bit) to floating-point number" instruction:

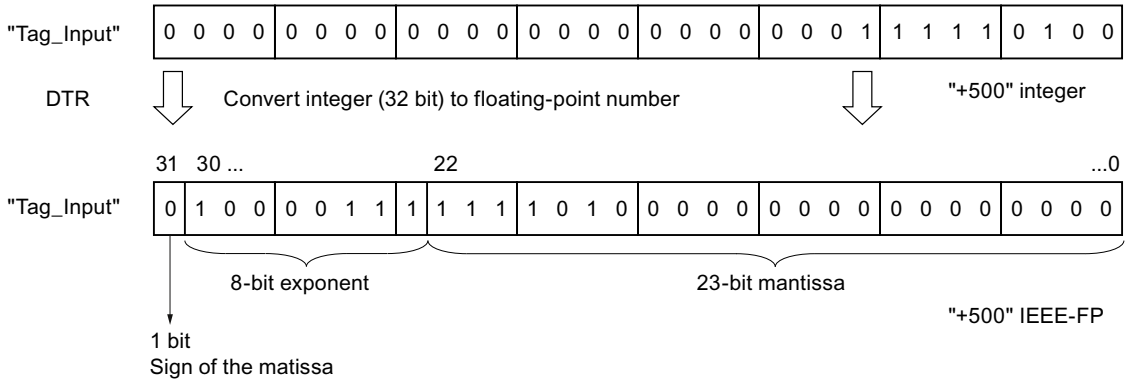
```
DTR
```

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Input"`	`// Load integer (32 bit) to accumulator 1`
`DTR`	`// Convert integer (32 bit) to floating-point number.`
`T "Tag_Output"`	`// Transfer the result to operand "Tag_Output".`

The following figure shows how the instruction works using specific operand values:



See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

INVI: Create ones complement integer (16 bit)

Description

Use the "Create ones complement integer (16 bit)" instruction to negate the value in the right word of accumulator 1 bit for bit.

During processing, the instruction inverts the signal state of each separate bit in the right word of accumulator 1. The ones are replaced with zeros and the zeros with ones.

The result is saved in the right word of accumulator 1. The contents of the left word in accumulator 1 are not affected by the instruction.

Syntax

Use the following syntax for the "Create ones complement integer (16 bit)" instruction:

```
INVI
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Input"	// Load value in accumulator 1
INVI	// Create ones complement
T "Tag_Output"	// Transfer the result to operand "Tag_Output".

The following table shows how the instruction works using specific operand values:

Operand	Value			
	15 . . .	. . . .	. . . .	. . . 0
Tag_Input	0110	0011	1010	1110
Tag_Output	1001	1100	0101	0001

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

INVD: Create ones complement double integer (32 bit)

Description

Use the "Create ones complement double integer (32 bit)" to negate the value in accumulator 1 bit for bit.

During processing, the instruction inverts the signal state of each separate bit in accumulator 1. The ones are replaced with zeros and the zeros with ones.

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

Syntax

Use the following syntax for the "Create ones complement integer (32 bit)" instruction:

```
INVD
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Input"	// Load value in accumulator 1
INVD	// Create ones complement
T "Tag_Output"	// Transfer the result to operand "Tag_Output".

The following table shows how the instruction works using specific operand values:

Operand	Value							
	31 . . .	. . . .	. . . .	. . . 16	15 . . .	. . . .	. . . .	. . . 0
Tag_Input	0110	1111	1000	1100	0110	0011	1010	1110
Tag_Output	1001	0000	0111	0011	1001	1100	0101	0001

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

NEGI: Negate integer (16 bit)

Description

Use the "Negate integer (16 bit)" instruction to change the sign of the value in the right word of accumulator 1 by forming the twos complement. The instruction interprets the value of accumulator 1 as a 16-bit integer.

The processing of the instruction is equal to a multiplication with "-1". The result of the instruction is saved in the right word of accumulator 1. The contents of the left word in accumulator 1 are not affected by the instruction.

The "Negate integer (16 bit)" instruction affects the status bits CC 0, CC 1, OV and OS. The following table shows how the instruction influences the status bit depending on the result:

Result:	CC 1	CC 0	OV	OS
+1 to +32 767	1	0	0	-
0	0	0	0	-
-1 to -32 767	0	1	0	-
(-) 32 768	0	1	1	1

Syntax

The following syntax is used for the "Negate integer (16 bit)" instruction:

```
NEGI
```

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Input"`	`// Load value in accumulator 1`
`NEGI`	`// Negate integer (16 bit)`
`T "Tag_Output"`	`// Transfer the result to operand "Tag_Output".`

The following table shows how the instruction works using specific operand values:

Operand	Value			
	15 . . .	. . . .	. . . .	. . . 0
Tag_Input	0101	1101	0011	1000
Tag_Output	1010	0010	1100	1000

See also

Overview of the valid data types (Page 2236)
Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)
Evaluating errors with output parameter RET_VAL (Page 5370)
Call function blocks as single or multi-instances (Page 2182)
Basic information on the status word (Page 2466)
Querying and setting status bits in STL (Page 2467)

NEGD: Negate integer (32 bit)

Description

Use the "Negate integer (32 bit)" instruction to change the sign of the value in accumulator 1 by forming the twos complement. The instruction interprets the value of accumulator 1 as a 32-bit integer.

The processing of the instruction is equal to a multiplication with "-1". The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

The "Negate integer (32 bit)" instruction affects the status bits CC 0, CC 1, OV and OS. The following table shows how the instruction influences the status bit depending on the result:

Result	CC 1	CC 0	OV	OS
+1 to +2 147 483 647	1	0	0	-
0	0	0	0	-
-1 to -2 147 483 647	0	1	0	-
(-) 2 147 483 648	0	1	1	1

Syntax

The following syntax is used for the "Negate integer (32 bit)" instruction:

```
NEGD
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Input"	// Load value in accumulator 1
NEGD	// Negate integer (32 bit).
T "Tag_Output"	// Transfer the result to operand "Tag_Output".

The following table shows how the instruction works using specific operand values:

Status	Value							
	31 . . .	. . . .	. . . .	. . . 16	15 . . .	. . . .	. . . .	. . . 0
Tag_Input	0101	1111	0110	0100	0101	1101	0011	1000
Tag_Output	1010	0000	1001	1011	1010	0010	1100	1000

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

NEGR: Negate floating-point number

Description

The instruction "Negate floating-point number" inverts the sign of the mantissa and is equal to a multiplication by "-1".

The instruction interprets the value of accumulator 1 as a floating-point number. The sign of the mantissa is in bit 31 of the floating-point number.

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

The instruction does not affect any status bits.

Syntax

Use the following syntax for the "Negate floating-point number" instruction:

```
NEGR
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Input"	// Load floating-point number into accumulator 1
NEGR	// Negate floating-point number.

STL	Explanation
`T "Tag_Output"`	`// Transfer the result to operand "Tag_Output".`

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Input	1.5E+02
Tag_Output	-1.5E+02

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

CAW: Switch bytes in the right word of accumulator 1

Description

Use the "Switch bytes in the right word of accumulator 1" instruction to switch the sequence of the two right bytes in the right word of accumulator 1.

The following table shows the contents of accumulator 1 before and after processing of the instruction:

Status	Bytes in accumulator 1			
Before processing	Value A	Value B	Value C	Value D
After processing	Value A	Value B	Value D	Value C

The result of the instruction is saved in the right word of accumulator 1. The bytes in the left word of accumulator 1 are not influenced by the instruction and remain unchanged.

Syntax

Use the following syntax for the "Switch bytes in the right word of accumulator 1" instruction:

```
CAW
```

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Input"                // Load value in accumulator.
CAW                          // Switch bytes in the right word of accumulator 1.
T "Tag_Output"               // Transfer the result to operand "Tag_Output".
```

The following table shows how the instruction works using specific operand values:

Operand	Value			
Tag_Input	0000	0000	1111	1111
Tag_Output	1111	1111	0000	0000

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

CAD: Switch all bytes in accumulator 1

Description

Use the "Switch all bytes in accumulator 1" instruction to switch the sequence of the bytes in accumulator 1.

The following table shows the contents of accumulator 1 before and after processing of the instruction:

Status	Bytes in accumulator 1			
Before processing	Value A	Value B	Value C	Value D
After processing	Value D	Value C	Value B	Value A

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

Syntax

Use the following syntax for the "Switch all bytes in accumulator 1" instruction:

```
CAD
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Input"	// Load value of the operand in accumulator 1.
CAD	// Switch bytes of accumulator 1.
T "Tag_Output"	// Transfer the result to operand "Tag_Output".

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Input	1111	0000	0000	1111	0000	0000	1111	1111
Tag_Output	1111	1111	0000	0000	0000	1111	1111	0000

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

RND: Round numerical value

Description

Use the "Round numerical value" instruction to convert the value in accumulator 1 to a 32-bit integer. The instruction interprets the value that is to be converted as a floating-point number and round this to the next integer.

If the floating-point number is exactly between an even and odd number, the even number is selected as result.

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

The conversion is not executed and the status bits OV and OS are set, if one of the following conditions applies:

- The value in accumulator 1 is not a valid floating-point number.
- The result is outside the range permitted for an integer of the data type DINT.

Syntax

Use the following syntax for the "Round numerical value" instruction:

```
RND
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Input"	// Load floating-point number into accumulator 1
RND	// Round floating-point number to an integer.
T "Tag_Output"	// Transfer the result to operand "Tag_Output".

The following table shows how the instruction works using specific operand values:

Operand	Values	
Tag_Input	101.5	-101.5
Tag_Output	102	-102

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

TRUNC: Truncate numerical value

Description

Use the "Truncate numerical value" instruction to convert the value in accumulator 1 to a 32-bit integer. The instruction interprets the value to be converted as a floating-point number and returns its whole number as result.

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

The conversion is not executed and the status bits OV and OS are set, if one of the following conditions applies:

- The value in accumulator 1 is not a valid floating-point number.
- The result is outside the range permitted for an integer of the data type DINT.

Syntax

Use the following syntax for the "Truncate numerical value" instruction:

```
TRUNC
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Input"	// Load floating-point number into accumulator 1
TRUNC	// Convert floating-point number to an integer of data type DINT.
T "Tag_Output"	// Transfer the result to operand "Tag_Output".

The following table shows how the instruction works using specific operand values:

Operand	Values	
Tag_Input	101.5	-101.5
Tag_Output	101	-101

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

RND+: Generate next higher integer from floating-point number

Description

Use the "Generate next higher integer from floating-point number" instruction to convert the value in accumulator 1 to a 32-bit integer. The instruction interprets the value that is to be converted as a floating-point number and rounds this number to the next higher integer. The result of the instruction is a number of the data type DINT that is greater than or equal to the floating-point number to be converted.

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

The conversion is not executed and the status bits OV and OS are set, if one of the following conditions applies:

- The value in accumulator 1 is not a valid floating-point number.
- The result is outside the range permitted for an integer of the data type DINT.

Syntax

Use the following syntax for the "Generate next higher integer from floating-point number" instruction:

```
RND+
```

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Input"                // Load floating-point number into accumulator 1
RND+                         // Round floating-point number to next higher integer.
T "Tag_Output"               // Transfer the result to operand "Tag_Output".
```

The following table shows how the instruction works using specific operand values:

Operand	Values	
Tag_Input	100.5	-100.5
Tag_Output	101	-100

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

RND-: Generate next lower integer from floating-point number

Description

Use the "Generate next lower integer from floating-point number" instruction to convert the value in accumulator 1 to a 32-bit integer. The instruction interprets the value that is to be converted as a floating-point number and rounds this number to the next lower integer. The result of the instruction is a number of the data type DINT that is less than or equal to the floating-point number to be converted.

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

The conversion is not executed and the status bits OV and OS are set, if one of the following conditions applies:

- The value in accumulator 1 is not a valid floating-point number.
- The result is outside the range permitted for an integer of the data type DINT.

Syntax

Use the following syntax for the "Generate next lower integer from floating-point number" instruction:

```
RND-
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Input"	// Load floating-point number into accumulator 1
RND-	// Round floating-point number to next lower integer.
T "Tag_Output"	// Transfer the result to operand "Tag_Output".

The following table shows how the instruction works using specific operand values:

Operand	Values	
Tag_Input	100.5	-100.5
Tag_Output	100	-101

See also

Overview of the valid data types (Page 2236)
Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)
Evaluating errors with output parameter RET_VAL (Page 5370)
Call function blocks as single or multi-instances (Page 2182)
Basic information on the status word (Page 2466)
Querying and setting status bits in STL (Page 2467)

Program control operations

Jumps

Jump label

Description

Use a jump label to specify the place in the program at which the program execution should resume after a jump. The name of the jump label can consist of up to 128 letters, numbers or underscores.

The jump label and the instruction in which the jump label is specified must be located in the same block. The name of a jump label can only be assigned once in a block. Each jump label can jump to several locations. You can declare a maximum of 256 jump labels.

Both forward and backward jumps are possible.

Syntax

Use the following syntax for the jump label:

<jump label>:

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Input_1"	// Load first comparison value.
L "Tag_Input_2"	// Load second comparison value.
>I	// Check whether the value of the operand "Tag_Input_1" is greater than the value of the operand "Tag_Input_2"

STL	Explanation
JC MyLABEL	// With a scan result "1" jump to jump label "MyLABEL" and continue with program processing // If the result of the query is "0", execute the next instruction.
L "Tag_Input_3"	// Load content of the operand "Tag_Input_3" in accumulator 1.
T "Tag_Output"	// Load contents of accumulator 1 in operand "Tag_Output".
MyLABEL: A "Tag_Input_4"	// If there was a jump, continue program execution at this point. // Query whether operand "Tag_Input_4" is "1"

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

JU: Unconditional jump

Description

Use the "Unconditional jump" instruction to interrupt the linear program execution and continue at the point that is marked by the specified jump label. The instruction is always executed independently of conditions.

The "Unconditional jump" instruction does not affect the status bits.

Syntax

Use the following syntax for the "Unconditional jump" instruction:

```
JU <jump label>
```

Parameters

The following table shows the parameters of the "Unconditional jump" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Input_1"              // Load first comparison value.
L "Tag_Input_2"              // Load second comparison value.
>I                           // Check whether the value of the operand
                             "Tag_Input_1" is greater than the value of the operand
                             "Tag_Input_2"
JC MyLABEL_1                 // With a scan result "1" jump to jump label
                             "MyLABEL_1" and continue with program processing
                             // If the result of the query is "0", continue linear
                             program execution.
L "Tag_Input_3"              // Load content of the operand "Tag_Input_3" in
                             accumulator 1.
T "Tag_Output_1"             // Load contents of accumulator 1 in operand
                             "Tag_Output_1".
JU MyLABEL_2                 // Jump to jump label "MyLABEL_2" and resume execution
                             of the program there.
MyLABEL_1: L "Tag_Input_4"   // Jump label
                             // Load content of the operand "Tag_Input_4" in
                             accumulator 1.
T "Tag_Output_2"             // Load contents of accumulator 1 in operand
                             "Tag_Output_2".
MyLABEL_2: A "Tag_Input_5"   // Jump label "MyLABEL_2"
                             // Query whether operand "Tag_Input_5" is "1"
```

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

JC: Jump if RLO = 1

Description

Use the "Jump if RLO = 1" instruction to interrupt linear program execution depending on the result of logic operation and resume it at the point marked by the specified jump label.

The jump to the specified jump label is only executed when the current RLO is "1". If the current RLO before the instruction is "0", the jump is not executed and the program is resumed with the next instruction.

The "Jump if RLO = 1" instruction sets the RLO to "1" both if the condition is satisfied or not satisfied.

Syntax

Use the following syntax for the "Jump if RLO = 1" instruction:

```
JC <jump label>
```

Parameters

The following table shows the parameters of the "Jump if RLO = 1" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Input_1"	// Load first comparison value.
L "Tag_Input_2"	// Load second comparison value.
>I	// Check whether the value of the operand "Tag_Input_1" is greater than the value of the operand "Tag_Input_2"
JC MyLABEL	// If RLO = "1", jump to jump label "MyLABEL" and continue program execution there.
	// If the RLO = "0", execute the next instruction.
L "Tag_Input_3"	// Load content of the operand "Tag_Input_3" in accumulator 1.
T "Tag_Output_1"	// Load contents of accumulator 1 in operand "Tag_Output_1".
MyLABEL: L "Tag_Input_4"	// If there was a jump, continue program execution at this point.
	// Load content of the operand "Tag_Input_4" in accumulator 1.

STL	Explanation
`T "Tag_Output_2"`	`// Transfer contents of accumulator 1 to the operand "Tag_Output_2".`

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

JCN: Jump if RLO = 0

Description

Use the "Jump if RLO = 0" instruction to interrupt the linear program execution depending on the result of logic operation (RLO) and resume it at the point marked by the specified jump label.

The jump to the specified jump label is only executed when the current RLO is "0". If the current RLO before the instruction is "1", the jump is not executed and the program is resumed with the next instruction.

The "Jump if RLO = 0" instruction sets the RLO to "1" both if the condition is satisfied or not satisfied.

Syntax

Use the following syntax for the "Jump if RLO = 0" instruction:

```
JCN <jump label>
```

Parameters

The following table shows the parameters of the "Jump if RLO = 0" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input_1"	// Query whether the value of operand "Tag_Input_1" is "1" and AND.
A "Tag_Input_2"	// Query whether the value of operand "Tag_Input_2" is "1" and AND.
JCN MyLABEL	// If RLO = "0", jump to jump label "MyLABEL" and continue program execution there.
	// If the RLO = "1", execute the next instruction.
L "Tag_Input_3"	// Load content of the operand "Tag_Input_3" in accumulator 1.
T "Tag_Output"	// Load contents of accumulator 1 in operand "Tag_Output".
MyLABEL: A "Tag_Input_4"	// If there was a jump, continue program execution at this point.
	// Query whether the value of operand "Tag_Input_4" is "1" and AND with the RLO

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

JCB: Jump if RLO = 1 and save RLO

Description

Use the "Jump if RLO = 1 and save RLO" instruction to interrupt the linear program execution depending on the result of logic operation (RLO) and resume it at the point marked by the specified jump label. At the same time, the signal state of the current RLO is copied to the binary result (BR).

The jump to the specified jump label is only executed when the current RLO is "1". In this case the instruction set the binary result to "1".

If the current RLO before the instruction is "0", the jump is not executed and the program is resumed with the next instruction. In this case the instruction assigns the signal state "0" to the binary result.

The "Jump if RLO = 1 and save RLO" instruction sets the RLO to "1" both if the condition is satisfied or not satisfied.

Syntax

Use the following syntax for the "Jump if RLO = 1 and save RLO" instruction:

```
JCB <jump label>
```

Parameters

The following table shows the parameters of the "Jump if RLO = 1 and save RLO" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Input_1"	// Load first comparison value.
L "Tag_Input_2"	// Load second comparison value.
>I	// Check whether the value of the operand "Tag_Input_1" is greater than the value of the operand "Tag_Input_2"
JCB MyLABEL	// Copy current RLO to BR.
	// If RLO = "1", jump to jump label "MyLABEL" and continue program execution there.
	// If the RLO = "0", execute the next instruction.
L "Tag_Input_3"	// Load content of the operand "Tag_Input_3" in accumulator 1.
T "Tag_Output_1"	// Load contents of accumulator 1 in operand "Tag_Output_1".
MyLABEL: L "Tag_Input_4"	// If there was a jump, continue program execution at this point.
	// Load content of the operand "Tag_Input_4" in accumulator 1.
T "Tag_Output_2"	// Load contents of accumulator 1 in operand "Tag_Output_2".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

JNB: Jump if RLO = 0 and save RLO

Description

Use the "Jump if RLO = 0 and save RLO" instruction to interrupt the linear program execution depending on the result of logic operation (RLO) and resume it at the point marked by the specified jump label. At the same time, the signal state of the current result of logic operation is copied to the binary result (BR).

The jump to the specified jump label is only executed when the current RLO is "0". In this case the instruction sets the binary result to "0".

If the current RLO before the instruction is "1", the jump is not executed and the program is resumed with the next instruction. In this case the instruction assigns the signal state "1" to the binary result.

The "Jump if RLO = 0 and save RLO" instruction sets the RLO to "1" both if the condition is satisfied or not satisfied.

Syntax

Use the following syntax for the "Jump if RLO = 0 and save RLO" instruction:

```
JNB <jump label>
```

Parameters

The following table shows the parameters of the "Jump if RLO = 0 and save RLO" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input_1"	// Query whether the value of operand "Tag_Input_1" is "1" and AND.
A "Tag_Input_2"	// Query whether the value of operand "Tag_Input_2" is "1" and AND.
JNB MyLABEL	// Copy current RLO to BR.

STL	Explanation
	// If RLO = "0", jump to jump label "MyLABEL" and continue program execution there.
	// If the RLO = "1", execute the next instruction.
L "Tag_Input_3"	// Load content of the operand "Tag_Input_3" in accumulator 1.
T "Tag_Output"	// Transfer contents of accumulator 1 to the operand "Tag_Output".
MyLABEL: A "Tag_Input_4"	// If there was a jump, continue program execution at this point.
	// Query whether the value of operand "Tag_Input_4" is "1" and AND with the RLO

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

JBI: Jump if BR = 1

Description

Use the "Jump if BR = 1" instruction to interrupt linear program execution depending on the binary result and resume it at the point marked by the specified jump label.

The jump to the specified jump label is only executed if the status bit BR is "1". If the status bit BR is "0", the jump is not executed and the program resumes with the next instruction.

Syntax

Use the following syntax for the "Jump if BR = 1" instruction:

```
JBI <jump label>
```

Parameters

The following table shows the parameters of the "Jump if BR = 1" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

```
STL                          Explanation
SET                          // Set RLO to "1" at the start of the block.
SAVE                         // Transfer signal state of the RLO to the BR bit.
....                         // Any program
AN OV                        // Query OV bit at end of block.
SAVE                         // If there is an overflow, set BR bit to "0".
                             // Set BR to "1" if there is no overflow.
JBI END                      // If BR = "1", jump to jump label END.
                             // If BR = "0", execute the next instruction.
R "Tag_Output_1"             // Reset operand "Tag_Output_1" to "0".
END: S "Tag_Output_2"        // Set operand "Tag_Output_2" to "1"
```

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

JNBI: Jump if BR = 0

Description

Use the "Jump if BR = 0" instruction to interrupt the linear program execution depending on the binary result (BR) and resume it at the point marked by the specified jump label.

The jump to the specified jump label is only executed if the status bit BR is "0". If the status bit BR before the instruction is "1", the jump is not executed and the program resumes with the next instruction.

Syntax

Use the following syntax for the "Jump if BR = 0" instruction:

```
JNBI <jump label>
```

Parameters

The following table shows the parameters of the "Jump if BR = 0" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

```
STL                          Explanation
SET                          // Set RLO to "1" at the start of the block.
SAVE                         // Transfer signal state of the RLO to the BR bit.
....                         // Any program
AN OV                        // Query OV bit at end of block.
SAVE                         // If there is an overflow, set BR bit to "0".
                             // Set BR to "1" if there is no overflow.
JNBI END                     // If BR = "0", jump to jump label END.
                             // If BR = "1", execute the next instruction.
S "Tag_Output"               // Set operand "Tag_Output" to "1"
END: BE                      // End block
```

See also

Overview of the valid data types (Page 2236)
Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)
Evaluating errors with output parameter RET_VAL (Page 5370)
Call function blocks as single or multi-instances (Page 2182)
Basic information on the status word (Page 2466)
Querying and setting status bits in STL (Page 2467)

JO: Jump if OV = 1

Description

Use the "Jump if OV = 1" instruction to interrupt linear program execution depending on the status bit OV and resume it at the point marked by the specified jump label.

The jump to the specified jump label is only executed if the status bit OV is "1". The status bit OV is set if, for example, the result of a calculation is outside the permitted range or an invalid value is returned during the comparison of floating-point numbers.

If the status bit OV is "0", the jump is not executed and the program continues with the next instruction.

Syntax

Use the following syntax for the "Jump if OV = 1" instruction:

```
JO <jump label>
```

Parameters

The following table shows the parameters of the "Jump if OV = 1" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value
L "Tag_Value_2"	// Load value
*I	// Multiply values
JO OVER	// If the status bit OV has signal state "1", jump to jump label "OVER" and continue program execution there.
	// If the signal state of the status bit OV is "0", execute the next instruction.
T "Tag_Result"	// Transfer the result of the multiplication to the operand "Tag_Result".
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
OVER: SET	// Continue program execution at this point if the condition (OV = "1") is satisfied.
	// Set RLO to signal state "1".
R "Tag_Output"	// Reset operand "Tag_Output" to "0".

STL	Explanation
NEXT: A "MyTag_1"	// Jump label "NEXT"
	// Query whether the value of operand "MyTag_1" is "1" and AND.
A "MyTag_2"	// Query whether the value of operand "MyTag_2" is "1" and AND.
S "Tag_Output_2"	// If RLO = 1, set the "Tag_Output_2" operand to "1".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

JOS: Jump if OS = 1

Description

Use the "Jump if OS = 1" instruction to interrupt linear program execution depending on the status bit OS and resume it at the point marked by the specified jump label.

The jump to the specified jump label is only executed if the status bit OS is "1". The status bit OS is always set when a number range overflow sets the status bit OV to "1". In contrast to status bit OV, the status bit OS remains set even if a result is subsequently in the permitted number range.

If the status bit OS is "0", the jump is not executed and the program continues with the next instruction.

Syntax

Use the following syntax for the "Jump if OS = 1" instruction:

```
JOS <jump label>
```

Parameters

The following table shows the parameters of the "Jump if OS = 1" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value
L "Tag_Value_2"	// Load value
*I	// Multiply loaded values.
	// Store product in accumulator 1.
L "Tag_Value_3"	// Load value
+I	// Add loaded value to the product.
	// Store product in accumulator 1.
L "Tag_Value_4"	// Load value
-I	// Subtract loaded value from the calculated sum.
	// Store product in accumulator 1.
JOS OVER	// If there is an overflow in one of the three preceding instructions, the status bit OS is set to "1"
	// If the signal state of the status bit OS is "1", the program resumes at the jump label "OVER".
	// If the signal state of the status bit OS is "0", the next instruction is executed
T "Tag_Result"	// Transfer the result of the entire calculation to the operand "Tag_Result"
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
OVER: SET	// Continue program execution at this point if the condition (OV = "1") is satisfied.
	// Set RLO to signal state "1".
R "Tag_Output"	// Reset operand "Tag_Output" to "0".
NEXT: A "MyTag_1"	// Jump label "NEXT"
	// Query whether the value of operand "MyTag_1" is "1" and AND.
A "MyTag_2"	// Query whether the value of operand "MyTag_2" is "1" and AND.
S "Tag_Output_2"	// If RLO = 1, set the "Tag_Output_2" operand to "1".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

JZ: Jump if the result is zero

Description

Use the "Jump if the result is zero" instruction to interrupt linear program execution depending on the status bits CC 0 and CC 1 and continue at the point marked by the specified jump label.

The jump to the specified jump label is only executed if the signal state of the status bits CC 0 and CC 1 is "0". The situation occurs when one of the following conditions applies:

- After processing an arithmetic instruction without overflow, accumulator 1 has the value "0".
- A negative overflow occurs when executing the "Add integers (16 bit)" (+I) or "Add double integers (32 bit)" (+D) instruction.
- The value gradually drops below the permitted range when processing an arithmetic instruction (floating-point arithmetic) with overflow.
- The content of accumulator 2 is equal to the content of accumulator 1 after the execution of a comparison instruction.
- The content of accumulator 1 is zero after the execution of a word logic operation.
- The value of the last shifted out bit is "0" after the execution of a shift instruction.

In all other cases the jump is not executed and the processing of the program is resumed with the next instruction.

Syntax

Use the following syntax for the "Jump if the result is zero" instruction:

```
JZ <jump label>
```

Parameters

The following table shows the parameters of the "Jump if the result is zero" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value"	// Load value of the operand "Tag_Value".
SRW 1	// Shift the contents of the right word in accumulator 1 one place to the right.
JZ ZERO	// If the signal state of the last bit shifted is "0", jump to jump label "OVER" and continue with program processing from there

STL	Explanation
	// If the signal state of the last bit shifted is "1", execute the next instruction.
L "MyTag_1"	// Load value of the operand "MyTag_1" in accumulator 1
INC 1	// Add right-hand byte of accumulator 1 with the value "1".
T "Tag_Result_1"	// Transfer the result to operand "Tag_Result_1".
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
ZERO: L "MyTag_2"	// Jump label "ZERO"
	// Load value of the operand "MyTag_2" in accumulator 1
INC 1	// Add right-hand byte of accumulator 1 with the value "1".
T "Tag_Result_2"	// Transfer the result to operand "Tag_Result_2".
NEXT: A "MyTag_3"	// Jump label "NEXT"
	// Query whether the value of operand "MyTag_3" is "1" and AND.
A "MyTag_4"	// Query whether the value of operand "MyTag_4" is "1" and AND.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

JN: Jump if the result is not zero

Description

Use the "Jump if the result is not zero" instruction to interrupt linear program execution depending on the status bits CC 0 and CC 1 and continue at the point marked by the specified jump label.

The jump to the specified jump label is only executed if the signal states of the status bits CC 0 and CC 1 are different. The situation occurs when one of the following conditions applies:

- After processing an arithmetic instruction without overflow, the value of accumulator 1 is not equal to zero.
- A negative overflow occurs when processing an arithmetic instruction with integers (+I, -I, *I, +D, -D, *D).
- A positive overflow occurs when executing a math instruction with integers (+I, -I, *I, /I, +D, -D, *D, /D, NEGI, NEGD).
- A positive or negative overflow occurs when processing an arithmetic instruction with floating-point numbers.
- The content of accumulator 2 is not equal to the content of accumulator 1 after the execution of a comparison instruction.
- The content of accumulator 1 is not zero after the execution of a word logic operation.
- The value of the last shifted out bit is "1" after the execution of a shift instruction.

In all other cases the jump is not executed and the processing of the program is resumed with the next instruction.

Syntax

Use the following syntax for the "Jump if the result is not zero" instruction:

```
JN <jump label>
```

Parameters

The following table shows the parameters of the "Jump if the result is not zero" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Valu_1".
L "Tag_Value_2"	// Load value of the operand "Tag_Value_2".
XOW	// Apply EXCLUSIVE OR logic operation
JN NOZERO	// If the value in accumulator 1 is not equal to zero, the jump to the "NOZERO" jump label is performed and program processing continues from there.
	// If the value in accumulator 1 is equal to zero, the next instruction will be processed.
AN "MyTag_1"	// Query whether the value of operand "MyTag_1" is "0" and AND.

STL	Explanation
S "Tag_Output_1"	// If RLO is "1", set the operand "Tag_Output_1" to "1".
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
NOZERO: AN "MyTag_2"	// Jump label "NOZERO"
	// Query whether the value of operand "MyTag_2" is "0" and AND.
S "Tag_Output_2"	// If RLO is "1", set the operand "Tag_Output_2" to "1"
NEXT: A "MyTag_3"	// Jump label "NEXT"
	// Query whether the value of operand "MyTag_3" is "1" and AND.
A "MyTag_4"	// Query whether the value of operand "MyTag_4" is "1" and AND.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

JP: Jump if the result is greater than zero

Description

Use the "Jump if the result is greater than zero" instruction to interrupt linear program execution depending on the status bits CC 0 and CC 1 and continue at the point marked by the specified jump label.

The jump to the specified jump label is executed only if the signal state of status bit CC 0 is "0" and CC 1 is "1". The situation occurs when one of the following conditions applies:

- After processing an arithmetic instruction without overflow, the value of accumulator 1 is greater than zero.
- A negative overflow occurs when processing an arithmetic instruction with integers (+I, -I, *I, +D, -D, *D).
- A positive overflow occurs when processing an arithmetic instruction with integers (*I, /I, *D, /D).
- A positive overflow occurs when processing an arithmetic instruction with floating-point numbers.

- After the execution of a comparison instruction, the content of accumulator 2 is greater than the content of accumulator 1.
- The contents of accumulator 1 are not zero after the execution of a word logic operation.
- The value of the last shifted out bit is "1" after the execution of a shift instruction.

In all other cases the jump is not executed and the processing of the program is resumed with the next instruction.

Syntax

Use the following syntax for the "Jump if the result is greater than zero" instruction:

```
JP <jump label>
```

Parameters

The following table shows the parameters of the "Jump if the result is greater than zero" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1".
L "Tag_Value_2"	// Load value of the operand "Tag_Value_2".
-I	// Subtract value of the operand "Tag_Value_2" from the value of the operand "Tag_Value_1".
JP POSITIVE	// At a positive result value in accumulator "1" jump to jump label "POSITIVE" and continue program processing from there
	// If the result in accumulator 1 is negative, execute the next instruction.
AN "MyTag_1"	// Query whether the value of operand "MyTag_1" is "0" and AND.
S "Tag_Output_1"	// If RLO is "1", set the operand "Tag_Output_1" to "1".
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
POSITIVE: AN "MyTag_2"	// Jump label "POSITIVE"
	// Query whether the value of operand "MyTag_2" is "0" and AND.
S "Tag_Output_2"	// If RLO is "1", set the operand "Tag_Output_2" to "1".

STL	Explanation
NEXT: A "MyTag_3"	// Jump label "NEXT"
	// Query whether the value of operand "MyTag_3" is "1" and AND.
A "MyTag_4"	// Query whether the value of operand "MyTag_4" is "1" and AND.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

JM: Jump if the result is less than zero

Description

Use the "Jump if the result is less than zero" instruction to interrupt linear program execution depending on the status bits CC 0 and CC 1 and continue at the point marked by the specified jump label.

The jump to the specified jump label is executed only if the signal state of status bit CC 0 is "1" and CC 1 is "0". The situation occurs when one of the following conditions applies:

- After processing an arithmetic instruction without overflow, the value of accumulator 1 is less than zero.
- A negative overflow occurs when processing an arithmetic instruction with integers (*I, *D).
- A positive overflow occurs when processing an arithmetic instruction with integers (+I, -I, +D, -D, NEGI, NEGD).
- A negative overflow occurs when processing an arithmetic instruction with floating-point numbers.
- After the execution of a comparison instruction, the content of accumulator 2 is less than the content of accumulator 1.

In all other cases the jump is not executed and the processing of the program is resumed with the next instruction.

Syntax

Use the following syntax for the "Jump if the result is less than zero" instruction:

```
JM <jump label>
```

Parameters

The following table shows the parameters of the "Jump if the result is less than zero" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1".
L "Tag_Value_2"	// Load value of the operand "Tag_Value_2".
-I	// Subtract value of the operand "Tag_Value_2" from the value of the operand "Tag_Value_1".
JM NEGATIVE	// At a negative result value in accumulator "1" jump to jump label "NEGATIVE" and continue program processing from there
	// If the result in accumulator 1 is positive, execute the next instruction.
AN "MyTag_1"	// Query whether the value of operand "MyTag_1" is "0" and AND.
S "Tag_Output_1"	// If RLO is "1", set the operand "Tag_Output_1" to "1".
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
NEGATIVE: AN "MyTag_2"	// Jump label "NEGATIVE"
	// Query whether the value of operand "MyTag_2" is "0" and AND.
S "Tag_Output_2"	// If RLO is "1", set the operand "Tag_Output_2" to "1".
NEXT: A "MyTag_3"	// Jump label "NEXT"
	// Query whether the value of operand "MyTag_3" is "1" and AND.
A "MyTag_4"	// Query whether the value of operand "MyTag_4" is "1" and AND.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

JPZ: Jump if the result is greater than or equal to zero

Description

Use the "Jump if the result is greater than or equal to zero" instruction to interrupt linear program execution depending on the status bit CC 0 and continue it at the point marked by the specified jump label.

The jump to the specified jump label is only executed if the signal state of the status bit CC 0 is "0". The situation occurs when one of the following conditions applies:

- After processing an arithmetic instruction without overflow, the value of accumulator 1 is greater than or equal to zero.
- A negative overflow occurs when processing an arithmetic instruction with integers (+I, -I, *I, +D, -D, *D).
- A positive overflow occurs when processing an arithmetic instruction with integers (*I, /I, *D, /D).
- A positive overflow occurs when processing an arithmetic instruction with floating-point numbers.
- The value gradually drops below the permitted range when processing an arithmetic instruction (floating-point arithmetic) with overflow.
- After the execution of a comparison instruction, the content of accumulator 2 is greater than or equal to the content of accumulator 1.
- An instruction for word logic operation was executed.
- A shift instruction was executed.

In all other cases the jump is not executed and the processing of the program is resumed with the next instruction.

Syntax

Use the following syntax for the "Jump if the result is greater than or equal to zero" instruction:

```
JPZ <jump label>
```

Parameters

The following table shows the parameters of the "Jump if the result is greater than or equal to zero" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1".
L "Tag_Value_2"	// Load value of the operand "Tag_Value_2".
-I	// Subtract value of the operand "Tag_Value_2" from the value of the operand "Tag_Value_1".
JPZ REGULAR	// At a positive result value or null in accumulator "1" jump to jump label "REGULAR" and continue program processing from there
	// If the result in accumulator 1 is negative, execute the next instruction.
AN "MyTag_1"	// Query whether the value of operand "MyTag_1" is "0" and AND.
S "Tag_Output_1"	// If RLO is "1", set the operand "Tag_Output_1" to "1".
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
REGULAR: AN "MyTag_2"	// Jump label "REGULAR"
	// Query whether the value of operand "MyTag_2" is "0" and AND.
S "Tag_Output_2"	// If RLO is "1", set the operand "Tag_Output_2" to "1".
NEXT: A "MyTag_3"	// Jump label "NEXT"
	// Query whether the value of operand "MyTag_3" is "1" and AND.
A "MyTag_4"	// Query whether the value of operand "MyTag_4" is "1" and AND.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

JMZ: Jump if the result is less than or equal to zero

Description

Use the "Jump if the result is less than or equal to zero" instruction to interrupt linear program execution depending on the status bit CC 1 and continue it at the point marked by the specified jump label.

The jump to the specified jump label is only executed if the signal state of the status bit CC 1 is "0". The situation occurs when one of the following conditions applies:

- After processing an arithmetic instruction without overflow, the value of accumulator 1 is less than or equal to zero.
- A negative overflow occurs when processing an arithmetic instruction with integers (+I, *I, +D, *D).
- A positive overflow occurs when processing an arithmetic instruction with integers (+I, -I, +D, -D, NEGI, NEGD).
- A negative overflow occurs when processing an arithmetic instruction with floating-point numbers.
- The value gradually drops below the permitted range when processing an arithmetic instruction (floating-point arithmetic) with overflow.
- After the execution of a comparison instruction, the content of accumulator 2 is less than or equal to the content of accumulator 1.
- The value of accumulator 1 is equal to zero after processing a word logic.
- The value of the last shifted bit is equal to zero after processing a write instruction.

In all other cases the jump is not executed and the processing of the program is resumed with the next instruction.

Syntax

Use the following syntax for the "Jump if the result is less than or equal to zero" instruction:

```
JMZ <jump label>
```

Parameters

The following table shows the parameters of the "Jump if the result is less than or equal to zero" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1".
L "Tag_Value_2"	// Load value of the operand "Tag_Value_2".
-I	// Subtract value of the operand "Tag_Value_2" from the value of the operand "Tag_Value_1".
JMZ MyLABEL	// At a negative result value or null in accumulator "1" jump to jump label "MyLABEL" and continue program processing from there
	// If the result in accumulator 1 is positive, execute the next instruction.
AN "MyTag_1"	// Query whether the value of operand "MyTag_1" is "0" and AND.
S "Tag_Output_1"	// If RLO = "1", set the operand "Tag_Output_1" to "1"
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
MyLABEL: AN "MyTag_2"	// Jump label "MyLABEL"
	// Query whether the value of operand "MyTag_2" is "0" and AND.
S "Tag_Output_2"	// If RLO="1" set the operand "Tag_Output_2" to "1"
NEXT: A "MyTag_3"	// Jump label "NEXT"
	// Query whether the value of operand "MyTag_3" is "1" and AND.
A "MyTag_4"	// Query whether the value of operand "MyTag_4" is "1" and AND.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

JUO: Jump if the result is invalid

Description

Use the "Jump if the result is invalid" instruction to interrupt linear program execution depending on the status bits CC 0 and CC 1 and continue at the point marked by the specified jump label.

The jump to the specified jump label is only executed if the signal state of the status bit CC 0 and CC 1 is "1". The situation occurs when one of the following conditions applies:

- It is divided by zero with a mathematical instruction (/I, /D, MOD).
- When processing an arithmetic instruction with floating-point numbers an overflow occurs and the result value is an invalid floating-point number.
- When processing a comparative instruction with floating-point numbers an invalid floating-point number was used or occurred as a result value.

In all other cases the jump is not executed and the processing of the program is resumed with the next instruction.

Syntax

Use the following syntax for the instruction "Jump if the result is invalid":

```
JUO <jump label>
```

Parameters

The following table shows the parameters of the "Jump if the result is invalid" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1".
L "Tag_Value_2"	// Load value of the operand "Tag_Value_2".
/I	// Divide value of the operand "Tag_Value_1" by value of the operand "Tag_Value_2".
JUO ERROR	// At a division by zero, jump to jump label "ERROR" and continue with program processing from there
	// Otherwise execute next instruction.
T "Tag_Result"	// Transfer contents of accumulator 1 to the operand "Tag_Result".
A "MyTag_1"	// Query whether the value of operand "MyTag_1" is "1" and AND.
R "MyTag_1"	// If RLO = "1", reset the operand "MyTag_1" to "0"
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
ERROR: AN "MyTag_1"	// Jump label "ERROR"
	// Query whether the value of operand "MyTag_1" is "0" and AND.

STL	Explanation
S "MyTag_1"	// If RLO is "1", set the operand "MyTag_1" to "1".
NEXT: A "MyTag_3"	// Jump label "NEXT"
	// Query whether the value of operand "MyTag_3" is "1" and AND.
A "MyTag_4"	// Query whether the value of operand "MyTag_4" is "1" and AND.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

JL: Define jump list

Description

Use the instruction "Define jump list" to program a list that consists of several entries of the instruction "Unconditional jump" (JU). The list begins directly after the instruction "Define jump list" and may contain up to 255 entries. The numbering of the jump functions in the list begins at zero. The jump list must be programmed without any gaps. The end of the jump list is marked by a jump label that is specified for the "Define jump list" instruction.

Which jump function of the list executes, depends on the value in the right byte of accumulator 1. If, for example, the accumulator 1 has the value "0", the first jump function executes. If the value in accumulator 1 is "1", the second jump function executes. If the value in accumulator 1 is greater than the number of list entries, the "Define jump list" instruction references the end of the list.

The "Jump distributor" instruction is processed independently of conditions and does not affect the status bits.

Syntax

You program the instruction "Define jump list" according to the following general outline:

```
L <Jump_number>
JL <Jump_label_end>
JU <Jump_label>
JU <Jump_label>
```

```
.... <Jump_label>
<Jump_label_end>:
```

The operand "jump number" contains the number of the instruction "Unconditional jump" (JU) to be executed. This number must be loaded in accumulator 1. The end of the jump list is marked by the jump label "jumplabel_end".

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value"	// Load jump number in accumulator 1.
JU END	// Start of the jump list
JU MyLABEL_1	// If the value is "0", execute in accumulator 1 and jump to jump label "MyLABEL_1".
JU MyLABEL_2	// If the value is "1", execute in accumulator 1 and jump to jump label "MyLABEL_2"
JU MyLABEL_3	// If the value is "2", execute in accumulator 1 and jump to jump label "MyLABEL_3".
END: L "MyTag_1"	// End of the jump list
	// Load content of the operand "MyTag_1" in accumulator 1.
L "MyTag_2"	// Load content of the operand "MyTag_2" in accumulator 1.
+I	// Add values
T "Tag_Output_2"	// Load contents of accumulator 1 in operand "Tag_Output_2".
MyLABEL_1: ...	// Jump label "MyLABEL_1"
....	// Any program
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
MyLABEL_2: ...	// Jump label "MyLABEL_2"
....	// Any program
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
MyLABEL_3: ...	// Jump label "MyLABEL_3"
....	// Any program
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
NEXT: ...	// Jump label "NEXT"

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

LOOP: Loop

Description

Use the "Loop" instruction to program loops in the program.

The instruction interprets the right word of accumulator 1 as an unsigned 16-bit integer in the range from 0 to 65 535. During execution the instruction first decrements the content of accumulator 1 by one. If the value in accumulator 1 after the decrement is not zero, the jump is executed at the specified jump label. If the value is zero, the jump is not executed and the processing of the program is resumed with the next instruction. The value in accumulator 1 therefore specifies the number of program loops that have to be executed. This number has to be saved in a loop counter.

The "Loop" instruction does not affect any status bits.

⚠ WARNING
If the loop counter is not preset or is preset with the value "0", the loop counter continues to decrement into negative numbers so that an endless loop results that can change the CPU to STOP mode.

Syntax

The "Loop" instruction is programmed according to the following general outline:

```
L <number>
<jump label>:
T <loop counter>
... Any program
L <loop counter>
LOOP <jump label>
```

The operand "number" contains the number of the loop cycles. The operand "Loop counter" contains the number of the loop cycles still to be executed. At the end of the program loop, the content of the loop counter is loaded in accumulator 1 and decremented by one by the instruction. The jump to the jump label at the start of the program loop is executed if the value in the accumulator is not equal to zero after the decrement.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value"	// Load number of time loop will execute into the right word of accumulator 1.
START: T "Tag_Counter"	// Start of the jump list
	// Transfer contents of accumulator 1 to the loop counter.
L "MyTag_1"	// Load value of the operand "MyTag_1".
L "MyTag_2"	// Load value of the operand "MyTag_2".
*D	// Multiply values
T "MyTag_1"	// Transfer the result of the multiplication to the operand "MyTag_1".
L "Tag_Counter"	// Load value of the loop counter into accumulator 1.
LOOP START	// Decrement value in accumulator 1 by one.
	// Jump to the start of the program loop if the value is not equal to zero.
	// If the value is zero, execute the next instruction.
L "MyTag_2"	// Load value of the operand "MyTag_2".
L 100	// Load the value 100.
>I	// Compare whether the value of the operand "MyTag_2" is greater than 200.
= "MyTag_3"	// Write comparison result to the operand "MyTag_3".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Data blocks

OPN: Open data block in DB register

Description

You can open a global data block (DB) with the "Open data block in DB register" instruction. The number of the data block is transferred to the DB register. Subsequent DB commands access the relevant blocks depending on the content of the register.

The "Open data block in DB register" instruction is processed without any conditions and does not affect either the result of the logic operation or the accumulator contents.

Syntax

Use the following syntax for the "Open data block in DB register" instruction:

```
OPN <data block>
```

Parameters

The following table shows the parameters of the "Open data block in DB register" instruction:

Operand	Declaration	Data block type	Description
<data block>	-	DB	Data block that will be opened.

Example

The following example shows how the instruction works:

```
STL                                  Explanation
OPN "GlobalDataBlock"                // Open global data block and transfer the
                                     number of the block to the DB register.
L %DBW0                              // Load data word DBW0 of the opened global
                                     data block into accumulator 1.
T "MyTag"                            // Transfer contents of accumulator 1 to the
                                     operand "MyTag_1".
```

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)
Basic information on the status word (Page 2466)
Querying and setting status bits in STL (Page 2467)
STL Basics (Page 2650)
Fully qualified addresses in STL (Page 223)
Partially qualified addresses in STL (Page 224)

OPNDI: Open data block in DI register

Description

You can open any data block with the "Open data block in DI register" instruction. The number of the data block is transferred to the DI register. Subsequent DI commands access the relevant blocks depending on the content of the register.

In the symbolic addressing of a local formal parameter from the block interface, you always access the data block whose number you specify when you call the block.

The "Open data block in DI register" instruction is processed without any conditions and does not affect either the result of the logic operation or the accumulator contents.

Syntax

Use the following syntax for the "Open data block in DI register" instruction:

```
OPNDI <data block>
```

Parameters

The following table shows the parameters of the "Open data block in DI register" instruction:

Operand	Declaration	Data block type	Description
<data block>	-	DI	Data block that will be opened.

Example

The following example shows how the instruction works:

STL	Explanation
OPNDI "DataBlock"	// Open data block and transfer the number of the data block to the DI register.
L %DIW0	// Load data word DIW0 of the opened data block into accumulator 1.

STL	Explanation
T "MyTag"	// Transfer contents of accumulator 1 to the operand "MyTag".

See also

Overview of the valid data types (Page 2236)
Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)
Evaluating errors with output parameter RET_VAL (Page 5370)
Call function blocks as single or multi-instances (Page 2182)
Basic information on the status word (Page 2466)
Querying and setting status bits in STL (Page 2467)
STL Basics (Page 2650)
Fully qualified addresses in STL (Page 223)
Partially qualified addresses in STL (Page 224)

CDB: Swap data block register

Description

Use the "Swap data block register" instruction to swap the contents of the data block register. The instruction executes regardless of conditions and does not affect the status bits.

Syntax

Use the following syntax for the "Swap data block register" instruction:

```
CDB
```

Example

The following example shows how the instruction works:

STL	Explanation
OPN "GlobalDataBlock"	// Open global data block and transfer the number of the data block to the DB register.
OPNDI "DataBlock"	// Open data block and transfer the number of the data block to the DI register.
CDB	// Swap data block register
	// The DB register points to "DataBlock" and the DI register points to "GlobalDataBlock".

STL	Explanation
L %DIW0	// Load DW0 from "GlobalDataBlock" into accumulator 1.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

L DBLG: Load the length of a global data block into accumulator 1

Description

Use the "Load the length of a global data block into accumulator 1" instruction to load the length of a global data block opened using the data block register into accumulator 1. The previous contents of accumulator 1 are shifted to accumulator 2.

If no global data block was opened via the data block register prior to the processing of the instruction, the value "0" is loaded into accumulator 1. The length of the data block is equal to the number of the data bytes.

The status bits are not influenced by the instruction.

Syntax

Use the following syntax for the "Load the length of a global data block into accumulator 1" instruction:

```
L DBLG
```

Example

The following example shows how the instruction works:

STL	Explanation
OPN "GlobalDataBlock"	// Open global data block and transfer the number of the data block to the DB register.
L DBLG	// Load the length of the opened data block into accumulator 1.
L "MyTag_1"	// Load comparison value

STL	Explanation
<D	// Compare whether the length of the data block is less than the value of the operand "MyTag_1".
= "MyTag_2"	// Write query result to the "MyTag_2" operand.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

L DBNO: Load the number of a global data block into accumulator 1

Description

Use the "Load the number of a global data block into accumulator 1" instruction to load the number of a global data block opened using the data block register into accumulator 1. The previous contents of accumulator 1 are shifted to accumulator 2.

If no global data block was opened via the data block register prior to the processing of the instruction, the value "0" is loaded into accumulator 1.

The status bits are not influenced by the instruction.

Syntax

Use the following syntax for the "Load the number of a global data block into accumulator 1" instruction:

```
L DBNO
```

Example

The following example shows how the instruction works:

STL	Explanation
OPN "GlobalDataBlock"	// Open global data block and transfer the number of the data block to the DB register.
L DBNO	// Load the number of the opened data block into accumulator 1.

STL	Explanation
L "MyTag_1"	// Load comparison value
==I	// Compare whether the number of the data block is equal to the value of the operand "MyTag_1".
= "MyTag_2"	// Write query result to the "MyTag_2" operand.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

L DILG: Load the length of an instance data block into accumulator 1

Description

Use the "Load the length of an instance data block into accumulator 1" instruction to load the length of an instance data block opened using the data block register into accumulator 1. The previous contents of accumulator 1 are shifted to accumulator 2.

If no instance data block was opened via the data block register prior to the instruction executing, the value "0" is loaded into accumulator 1. The length of the data block is equal to the number of the data bytes.

The status bits are not influenced by the instruction.

Syntax

Use the following syntax for the "Load the length of an instance data block into accumulator 1" instruction:

```
L DILG
```

Example

The following example shows how the instruction works:

STL	Explanation
OPNDI "InstanceDataBlock"	// Open data block and transfer the number of the data block to the DI register.

STL	Explanation
L DILG	// Load the length of the opened data block into accumulator 1.
L "MyTag_1"	// Load comparison value
<I	// Compare whether the length of the data block is less than the value of the operand "MyTag_1".
= "MyTag_2"	// Write query result to the "MyTag_2" operand.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

L DINO: Load the number of an instance data block into accumulator 1

Description

Use the "Load the number of an instance data block into accumulator 1" instruction to load the number of an instance data block opened using the data block register into accumulator 1. The previous contents of accumulator 1 are shifted to accumulator 2.

If no instance data block was opened via the data block register prior to the instruction executing, the value "0" is loaded into accumulator 1.

The status bits are not influenced by the instruction.

Syntax

Use the following syntax for the "Load the number of an instance data block into accumulator 1" instruction:

```
L DINO
```

Example

The following example shows how the instruction works:

STL	Explanation
OPNDI "InstanceDataBlock"	// Open data block and transfer the number of the data block to the DI register.
L DINO	// Load the number of the opened data block into accumulator 1.
L "MyTag_1"	// Load comparison value
==I	// Compare whether the number of the data block is equal to the value of the operand "MyTag_1".
= "MyTag_2"	// Assign query result to the "MyTag_2" operand.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Code blocks

BE: Block end

Description

You can use the "Block end" instruction to end the block currently being processed and go to the point in the program where the block is called. The program processing is then resumed with the instruction that is located directly after the block call.

The "Block end" instruction is executed independent of conditions. If the processing of the instruction is skipped by a jump instruction, the current program execution does not end but instead resumes at the jump destination within the block.

Syntax

The following syntax is used for the "Block end" instruction:

```
BE
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
SIN	// Form sine value
	// Store result in accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.
L "Tag_Value_3"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_3" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.
T "Tag_Result"	// Transfer contents of accumulator 1 to the operand "Tag_Result".
BE	// End block

See also

Overview of the valid data types (Page 2236)
Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)
Evaluating errors with output parameter RET_VAL (Page 5370)
Call function blocks as single or multi-instances (Page 2182)
Basic information on the status word (Page 2466)
Querying and setting status bits in STL (Page 2467)

BEC: Conditional block end

Description

You can use the "Conditional block end" instruction to end the block currently being processed depending on the result of logic operation (RLO) and to go to point in the program where the block is called.

If the RLO is "1", the instruction is executed. The block currently being processed is terminated and the program processing is resumed in the calling block. The program processing is then resumed with the instruction that is located directly after the block call.

If the RLO is "0" when the instruction is processed, the instruction is not executed. In this case the CPU sets the RLO to "1" and executes the next instruction.

If the processing of the instruction is skipped by a jump instruction, the current program execution does not end but instead resumes at the jump destination within the block.

Syntax

The following syntax is used for the "Conditional block end" instruction:

```
BEC
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
>I	// Compare whether the value of accumulator 2 is greater than the value of accumulator 1.
A "Tag_Input"	// Query the value of operand "Tag_Input" for "1" and AND with the RLO.
BEC	// End block if condition satisfied (RLO = "1").
	// If the condition is not satisfied(RLO = "0"), execute the next instruction.
T "Tag_Result"	// Transfer contents of accumulator 1 to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)
Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)
Evaluating errors with output parameter RET_VAL (Page 5370)
Call function blocks as single or multi-instances (Page 2182)
Basic information on the status word (Page 2466)
Querying and setting status bits in STL (Page 2467)

BEU: Unconditional block end

Description

You can use the "Unconditional block end" instruction to end the block currently being processed and go to the point in the program where the block is called. The program processing is then resumed with the instruction that is located directly after the block call.

The "Unconditional block end" instruction is executed independent of conditions and can be programmed multiple times within a block. If the processing of the instruction is skipped by a jump instruction, the current program execution does not end but instead resumes at the jump destination within the block.

Syntax

The following syntax is used for the "Unconditional block end" instruction:

```
BEU
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
>I	// Compare whether the value of accumulator 2 is greater than the value of accumulator 1.
A "Tag_Input"	// Query the value of operand "Tag_Input" for "1" and AND with the RLO.

STL	Explanation
JC NEXT	// If the condition is satisfied (RLO = "1"), continue program execution at jump label"NEXT".
	// If the condition is not satisfied(RLO = "0"), execute the next instruction.
BEU	// End block
NEXT: T "Tag_Result"	// Jump label "NEXT"
	// Transfer contents of accumulator 1 to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

CALL: Call block

Description

Use the "Call block" instruction to call the following block types in the program:

- Functions
- Function blocks

The "Call block" instruction executes without any conditions. After processing of the instruction, program execution is resumed in the called block.

You can feed a called block with data. The data is transferred via the block parameters. The parameters of the called block are listed according to the call instruction in the called block. You can assign the required current parameters to these parameters. During execution of the "Call block" instruction, the data is transferred to the called block. By transferring the data, the content of the status word and the address and data block registers are changed.

Those parameters of a function block that are not fed with parameters, retain their current value. All parameters must be fed when functions are called. If a called block has no parameters, no parameter list is displayed.

If the called block requires an instance data block, give it to the block, separated by a comma when called. The specified data block must be created before the call.

When the called block is processed, the CPU switches back to the calling block and resumes the processing of this block after the call instruction.

Note

The "Conditional block call" instruction can only be used when the block property "Parameter passing via registers" is disabled.

Syntax

When you call a function, use the following syntax for the "Call block" instruction:

```
CALL <FC>
```

When you call a function block, use the following syntax for the "Call block" instruction:

```
CALL <FB>, <FB_DB>
```

Example

The following example shows how the instruction works:

STL	Explanation
CALL "MyFunction"	// Call function "MyFunction"
Input_1 := "Tag_Input_1",	// Assign actual parameters
Input_2 := "Tag_Input_2"	
Output_1 := "Tag_Output_1"	
Output_2 := "Tag_Output_1"	
CALL "MyFunctionBlock", "MyFB_DB"	// Call function block "MyFunctionBlock"
Value_1 := "Tag_Value_1"	// Assign actual parameters
Value_2 := "Tag_Value_2"	
Output := "Tag_Output"	
CALL "LIMIT"	// Call "Set limit value" instruction
MN := "Tag_LowLimit"	// Assign actual parameters
IN := "Tag_InputValue"	
MX := "Tag_HighLimit"	
OUT := "Tag_Output"	
CALL "CTU", "CTU_DB"	// Call "Count up" counter
CU := "Tag_StartCTU"	// Assign actual parameters
R := "Tag_ResetCounter"	
PV := "Tag_PresetValue"	
Q := "Tag_CounterStatus"	
CV := "Tag_CounterValue"	

See also

Overview of the valid data types (Page 2236)
Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)
Evaluating errors with output parameter RET_VAL (Page 5370)
Call function blocks as single or multi-instances (Page 2182)
Basic information on the status word (Page 2466)
Querying and setting status bits in STL (Page 2467)
STL Basics (Page 2650)
Fully qualified addresses in STL (Page 223)
Partially qualified addresses in STL (Page 224)

CC: Conditional block call

Description

Use the "Conditional block call" instruction to call functions (FCs) and function blocks (FBs) depending on the result of logic operation (RLO).

The instruction is only executed if the current result of logic operation (RLO) before execution of the instruction is "1". After processing of the instruction, program execution is resumed in the called block. When the called block is processed, the CPU switches back to the calling block and resumes the processing of this block after the call instruction.

If there is a block change, the status bit OS is reset to "0". The status bits CC 0, CC 1 and OV are affected depending on the calling block.

Note

The "Conditional block call" instruction can only be used when the block property "Parameter passing via registers" is enabled. However, this lowers performance.

The contents of the accumulators and the address registers are not changed by the "Conditional block call" instruction.

If the current RLO is "0", the instruction as well as the block call are not executed and the RLO is set to "1".

Syntax

When you call a function, use the following syntax for the "Conditional block call" instruction:

```
CC <FC>
```

When you call a function block, use the following syntax for the "Conditional block call" instruction:

```
CC <FB>
```

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input_1"	// Query the value of operand "Tag_Input_1" for "1" and AND with the current RLO.
CC "MyFunction"	// If condition is satisfied (RLO = "1"), call the function MyFunction.
	// If the condition is not satisfied(RLO = "0"), execute the next instruction.
A "Tag_Input_2"	// Query the value of operand "Tag_Input_2" for "1" and AND with the current RLO.
CC "MyFunctionBlock"	// If condition is satisfied (RLO = "1"), call the function block MyFunctionBlock.
	// If the condition is not satisfied(RLO = "0"), execute the next instruction.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Overview of block properties (Page 2512)

Parameter assignment via registers during block call in STL (Page 2652)

UC: Unconditional block call

Description

You can use the "Unconditional block call" instruction to call functions (FCs) and function blocks (FBs) that have no parameters and no instance data blocks.

The instruction is executed regardless of conditions. After processing of the instruction, program execution is resumed in the called block. When the called block is processed, the CPU switches back to the calling block and resumes the processing of this block after the call instruction.

If there is a block change, the status bit OS is reset to "0". The status bits CC 0, CC 1 and OV are affected depending on the calling block.

Note

The "Unconditional block call" instruction can only be used when the block property "Parameter passing via registers" is enabled. However, this lowers performance.

The contents of the accumulators and the address registers are not changed by the "Unconditional block call" instruction.

Syntax

When you call a function, use the following syntax for the "Unconditional block call" instruction:

```
UC <FC>
```

When you call a function block, use the following syntax for the "Unconditional block call" instruction:

```
UC <FB>
```

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input_1"	// Query the value of operand "Tag_Input_1" for "1" and AND with the current RLO.
UC "MyFunction"	// Call function "MyFunction".
A "Tag_Input_2"	// Query the value of operand "Tag_Input_2" for "1" and AND with the current RLO.
UC "MyFunctionBlock"	// Call function block "MyFunctionBlock".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Overview of block properties (Page 2512)

Parameter assignment via registers during block call in STL (Page 2652)

Word logic operations

AW: AND logic operation word by word

Description

Use the instruction "AND logic operation word by word" to AND the value of the right word in accumulator 1 with the value of the right word in accumulator 2 or a specified constant bit by bit. The result is saved in the right word of accumulator 1. The contents of the left word in accumulator 1 remain unchanged.

The instruction links bit 0 of accumulator 1 with bit 0 of accumulator 2 or a constant and saves the result to bit 0 of accumulator 1. Bits 1 to 15 are linked in the same manner.

The result bit has signal state "1" if both of the bits in the logic operation also have signal state "1". If one of the two bits of the logic operation has signal state "0", the corresponding result bit is reset.

The following table shows how the result is formed by an AND word logic operation:

Accumulator 2 / constant	0	0	1	1
Accumulator 1	0	1	0	1
Result	0	0	0	1

The instruction is executed independent of conditions and does not affect the result of logic operation.

The instruction affects the status bits CC 0, CC 1 and OV.

Syntax

If the operation is used with a constant, use the following syntax for the "AND logic operation word by word" instruction:

```
AW <constant>
```

If the operation is used with the value of the right word in accumulator 2, use the following syntax for the "AND logic operation word by word" instruction:

```
AW
```

Parameters

The following table shows the parameters of the "AND logic operation word by word" instruction:

Parameters	Data type	Description
<constant>	WORD	Value that is ANDed with the value in the right word of accumulator 1.

Example

The following example shows how the instruction works:

```
STL                         Explanation
L "Tag_Value_1"             // Load value of the operand into the right word of
                            accumulator 1.
AW W#16#F6B5                // AND the value of the right word in accumulator 1
                            with the constant (W#16#F6B5).
                            // Save the result in the right word of accumulator 1.
T "Tag_Result_1"            // Transfer the result to operand "Tag_Result_1".
L "Tag_Value_2"             // Load value of the operand into the right word of
                            accumulator 1.
L "Tag_Value_3"             // Shift the contents of accumulator 1 to accumulator
                            2.
                            // Load value of the operand "Tag_Value_3" into the
                            right word of accumulator 1.
AW                          // AND the value of the right word in accumulator 2
                            ("Tag_Value_2") with the value of the right word in
                            accumulator 1 ("Tag_Value_3")
                            // Save the result in the right word of accumulator 1.
T "Tag_Result_2"            // Transfer the result to operand "Tag_Result_2".
```

The following table shows how the instruction works using specific operand values:

Operand	Value			
Tag_Value_1	0101	1001	0011	1011
Constant (W#16#F6B5)	1111	0110	1011	0101
Tag_Result_1	0101	0000	0011	0001
Tag_Value_2	0110	1100	0010	1010
Tag_Value_3	1101	1010	1001	0011
Tag_Result_2	0100	1000	0000	0010

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Setting the status bit for word logic operations (Page 2470)

OW: OR logic operation word by word

Description

Use the instruction "OR logic operation word by word" to OR the value of the right word in accumulator 1 with the value of the right word in accumulator 2 or a specified constant bit by bit. The result is saved in the right word of accumulator 1. The contents of the left word in accumulator 1 remain unchanged.

The instruction links bit 0 of accumulator 1 with bit 0 of accumulator 2 or a constant and saves the result to bit 0 of accumulator 1. Bits 1 to 15 are linked in the same manner.

The result bit has signal state "1" when at least one of the two bits in the logic operation has signal state "1". If both bits of the logic operation have signal state "0", the corresponding result bit is reset to "0".

The following table shows how the result is formed by an OR word logic operation:

Accumulator 2 / constant	0	0	1	1
Accumulator 1	0	1	0	1
Result	0	1	1	1

The instruction is executed independent of conditions and does not affect the result of logic operation.

The instruction affects the status bits CC 0, CC 1 and OV.

Syntax

If the operation is used with a constant, use the following syntax for the "OR logic operation word by word" instruction:

```
OW <constant>
```

If the operation is used with the value of the right word in accumulator 2, use the following syntax for the "OR logic operation word by word" instruction:

```
OW
```

Parameters

The following table shows the parameters of the "OR logic operation word by word" instruction:

Parameters	Data type	Description
<constant>	WORD	Value that is ORed with the value in the right word of accumulator 1.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand into the right word of
                             accumulator 1.
OW W#16#F6B5                 // OR the value of the right word in accumulator 1
                             with the constant (W#16#F6B5).
                             // Save the result in the right word of accumulator 1.
T "Tag_Result_1"             // Transfer the result to operand "Tag_Result_1".
L "Tag_Value_2"              // Load value of the operand into the right word of
                             accumulator 1.
L "Tag_Value_3"              // Shift the contents of accumulator 1 to accumulator
                             2.
                             // Load value of the operand "Tag_Value_3" into the
                             right word of accumulator 1.
OW                           // OR the value of the right word in accumulator 2
                             ("Tag_Value_2") with the value of the right word in
                             accumulator 1 ("Tag_Value_3")
                             // Save the result in the right word of accumulator 1.
T "Tag_Result_2"             // Transfer the result to operand "Tag_Result_2".
```

The following table shows how the instruction works using specific operand values:

Operand	Value			
Tag_Value_1	0101	1001	0011	1011
Constant (W#16#F6B5)	1111	0110	1011	0101
Tag_Result_1	1111	1111	1011	1111
Tag_Value_2	0110	1100	0010	1010
Tag_Value_3	1101	1010	1001	0011
Tag_Result_2	1111	1110	1011	1011

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Setting the status bit for word logic operations (Page 2470)

XOW: EXCLUSIVE OR logic operation word by word

Description

Use the instruction "EXCLUSIVE OR logic operation word by word" to EXCLUSIVE OR the value of the right word in accumulator 1 with the value of the right word in accumulator 2 or a specified constant bit by bit. The result is saved in the right word of accumulator 1. The contents of the left word in accumulator 1 remain unchanged.

The instruction links bit 0 of accumulator 1 with bit 0 of accumulator 2 or a constant and saves the result to bit 0 of accumulator 1. Bits 1 to 15 are linked in the same manner.

The result bit has signal state "1" if the signal state of the two bits to be linked is different. If both of the bits of the logic operation have the same signal state, the corresponding result bit is reset to "0".

The following table shows how the result is formed by an EXCLUSIVE OR word logic operation:

Accumulator 2 / constant	0	0	1	1
Accumulator 1	0	1	0	1
Result	0	1	1	0

The instruction is executed independent of conditions and does not affect the result of logic operation.

The instruction affects the status bits CC 0, CC 1 and OV.

Syntax

If the operation is used with a constant, use the following syntax for the "EXCLUSIVE OR logic operation word by word" instruction:

```
XOW <constant>
```

If the operation is used with the value of the right word in accumulator 2, use the following syntax for the "EXCLUSIVE OR logic operation word by word" instruction:

```
XOW
```

Parameters

The following table shows the parameters of the "EXCLUSIVE OR logic operation word by word" instruction:

Parameters	Data type	Description
<constant>	WORD	Value that is XORed with the value in the right word of accumulator 1.

Example

The following example shows how the instruction works:

```
STL                         Explanation
L "Tag_Value_1"             // Load value of the operand into the right word of
                            accumulator 1.
XOW W#16#F6B5               // XOR the value of the right word in accumulator 1
                            with the constant (W#16#F6B5).
                            // Save the result in the right word of accumulator 1.
T "Tag_Result_1"            // Transfer the result to operand "Tag_Result_1".
L "Tag_Value_2"             // Load value of the operand into the right word of
                            accumulator 1.
L "Tag_Value_3"             // Shift the contents of accumulator 1 to accumulator
                            2.
                            // Load value of the operand "Tag_Value_3" into the
                            right word of accumulator 1.
XOW                         // XOR the value of the right word in accumulator 2
                            ("Tag_Value_2") with the value of the right word in
                            accumulator 1 ("Tag_Value_3")
                            // Save the result in the right word of accumulator 1.
T "Tag_Result_2"            // Transfer the result to operand "Tag_Result_2".
```

The following table shows how the instruction works using specific operand values:

Operand	Value			
Tag_Value_1	0101	1001	0011	1011
Constant (W#16#F6B5)	1111	0110	1011	0101
Tag_Result_1	1010	1111	1000	1110
Tag_Value_2	0110	1100	0010	1010
Tag_Value_3	1101	1010	1001	0011
Tag_Result_2	1011	0110	1011	1001

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Setting the status bit for word logic operations (Page 2470)

AD: AND logic operation double word by double word

Description

You can use the instruction "AND logic operation double word by double word" to AND the contents of accumulator 1 with the contents of accumulator 2 or with the value of a specified constant, bit by bit. The result is stored in accumulator 1.

The instruction links bit 0 of accumulator 1 with bit 0 of accumulator 2 or a constant and saves the result to bit 0 of accumulator 1. Bits 1 to 31 are linked in the same manner.

The result bit has signal state "1" if both of the bits in the logic operation also have signal state "1". If one of the two bits of the logic operation has signal state "0", the corresponding result bit is reset.

The following table shows how the result is formed by an AND word logic operation:

Accumulator 2 / constant	0	0	1	1
Accumulator 1	0	1	0	1
Result	0	0	0	1

The instruction is executed independent of conditions and does not affect the result of logic operation.

The instruction affects the status bits CC 0, CC 1 and OV.

Syntax

If the operation is used with a constant, use the following syntax for the "AND logic operation double word by double word" instruction:

```
AD <constant>
```

When using the operation with the value of accumulator 2, use the following syntax for the "AND logic operation double word by double word" instruction:

```
AD
```

Parameters

The following table shows the parameters of the instruction "AND logic operation double word by double word":

Parameters	Data type	Description
<constant>	DWORD	Value that is ANDed with the value in accumulator 1.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand into accumulator 1.
AD DW#16#39C657AC            // AND the value in accumulator 1 with the constant
                             (DW#16#39C657AC).
                             // Store result in accumulator 1.
T "Tag_Result_1"             // Transfer the result to operand "Tag_Result_1".
L "Tag_Value_2"              // Load value of the operand into accumulator 1.
L "Tag_Value_3"              // Shift the contents of accumulator 1 to accumulator
                             2.
                             // Load value of the operand "Tag_Value_3" in
                             accumulator 1.
AD                           // AND the value of accumulator 2 Tag_Value_2 with the
                             value of accumulator 1 ("Tag_Value_3")
                             // Store result in accumulator 1.
T "Tag_Result_2"             // Transfer the result to operand "Tag_Result_2".
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	0101	1111	0110	0100	1001	1101	0011	1011
Constant (DW#16#39C657AC)	0011	1001	1100	0110	0101	0111	1010	1100
Tag_Result_1	0001	1001	0100	0100	0001	0101	0010	1000
Tag_Value_2	0110	0101	0100	0011	0101	1101	0010	1011
Tag_Value_3	0011	1001	1100	0110	0101	0111	1010	1100
Tag_Result_2	0010	0001	0100	0010	0101	0101	0010	1010

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Setting the status bit for word logic operations (Page 2470)

OD: OR logic operation double word by double word

Description

You can use the instruction "OR logic operation double word by double word" to OR the contents of accumulator 1 with the contents of accumulator 2 or with the value of a specified constant, bit by bit. The result is stored in accumulator 1.

The instruction links bit 0 of accumulator 1 with bit 0 of accumulator 2 or a constant and saves the result to bit 0 of accumulator 1. Bits 1 to 31 are linked in the same manner.

The result bit has signal state "1" when at least one of the two bits in the logic operation has signal state "1". If both bits of the logic operation have signal state "0", the corresponding result bit is reset to "0".

The following table shows how the result is formed by an OR word logic operation:

Accumulator 2 / constant	0	0	1	1
Accumulator 1	0	1	0	1
Result	0	1	1	1

The instruction is executed independent of conditions and does not affect the result of logic operation.

The instruction affects the status bits CC 0, CC 1 and OV.

Syntax

If the operation is used with a constant, use the following syntax for the "OR logic operation double word by double word" instruction:

```
OD <constant>
```

When using the operation with the value of accumulator 2, use the following syntax for the "OR logic operation double word by double word" instruction:

```
OD
```

Parameters

The following table shows the parameters of the "OR logic operation double word by double word" instruction:

Parameters	Data type	Description
<constant>	DWORD	Value that is ORed with the value in accumulator 1.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand into accumulator 1.
OD DW#16#39C657AC            // OR the value in accumulator 1 with the constant
                             (DW#16#39C657AC).
                             // Store result in accumulator 1.
T "Tag_Result_1"             // Transfer the result to operand "Tag_Result_1".
L "Tag_Value_2"              // Load value of the operand into accumulator 1.
L "Tag_Value_3"              // Shift the contents of accumulator 1 to accumulator
                             2.
                             // Load value of the operand "Tag_Value_3" in
                             accumulator 1.
OD                           // OR the value of accumulator 2 Tag_Value_2 with the
                             value of accumulator 1 ("Tag_Value_3")
                             // Store result in accumulator 1
T "Tag_Result_2"             // Transfer the result to operand "Tag_Result_2".
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	0101	1111	0110	0100	1001	1101	0011	1011
Constant (DW#16#39C657AC)	0011	1001	1100	0110	0101	0111	1010	1100
Tag_Result_1	0111	1111	1110	0110	1101	1111	1011	1111
Tag_Value_2	0110	0101	0100	0011	0101	1101	0010	1011
Tag_Value_3	0011	1001	1100	0110	0101	0111	1010	1100
Tag_Result_2	0111	1101	1100	0111	0101	1111	1110	1111

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Setting the status bit for word logic operations (Page 2470)

XOD: EXCLUSIVE OR logic operation double word by double word

Description

You can use the instruction "EXCLUSIVE OR logic operation double word by double word" to XOR the contents of accumulator 1 with the contents of accumulator 2 bit by bit. The result is stored in accumulator 1.

The instruction links bit 0 of accumulator 1 with bit 0 of accumulator 2 or a constant and saves the result to bit 0 of accumulator 1. Bits 1 to 31 are linked in the same manner.

The result bit has signal state "1" if the signal state of the two bits to be linked is different. If both of the bits of the logic operation have the same signal state, the corresponding result bit is reset to "0".

The following table shows how the result is formed by an EXCLUSIVE OR word logic operation:

Accumulator 2 / constant	0	0	1	1
Accumulator 1	0	1	0	1
Result	0	1	1	0

The instruction is executed independent of conditions and does not affect the result of logic operation.

The instruction affects the status bits CC 0, CC 1 and OV.

Syntax

If the operation is used with a constant, use the following syntax for the "EXCLUSIVE OR logic operation double word by double word" instruction:

```
XOD <constant>
```

When using the operation with the value of accumulator 2, use the following syntax for the "EXCLUSIVE OR logic operation double word by double word" instruction:

```
XOD
```

Parameters

The following table shows the parameters of the "EXCLUSIVE OR logic operation double word by double word" instruction:

Parameters	Data type	Description
<constant>	DWORD	Value that is XORed with the value in accumulator 1

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand into accumulator 1.
XOD DW#16#39C657AC           // XOR the value in accumulator 1 with the constant
                             (DW#16#39C657AC).
                             // Store result in accumulator 1.
T "Tag_Result_1"             // Transfer the result to operand "Tag_Result_1".
L "Tag_Value_2"              // Load value of the operand into accumulator 1.
L "Tag_Value_3"              // Shift the contents of accumulator 1 to accumulator
                             2.
                             // Load value of the operand "Tag_Value_3" in
                             accumulator 1.
XOD                          // XOR the value of accumulator 2 Tag_Value_2 with the
                             value of accumulator 1 ("Tag_Value_3")
                             // Store result in accumulator 1.
T "Tag_Result_2"             // Transfer the result to operand "Tag_Result_2".
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	0101	1111	0110	0100	1001	1101	0011	1011
Constant (DW#16#39C657AC)	0011	1001	1100	0110	0101	0111	1010	1100
Tag_Result_1	0110	0110	1010	0010	1100	1010	1001	0111
Tag_Value_2	0110	0101	0100	0011	0101	1101	0010	1011
Tag_Value_3	0011	1001	1100	0110	0101	0111	1010	1100
Tag_Result_2	0101	1100	1000	0101	0000	1010	1000	0111

See also

Overview of the valid data types (Page 2236)
Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)
Evaluating errors with output parameter RET_VAL (Page 5370)
Call function blocks as single or multi-instances (Page 2182)
Basic information on the status word (Page 2466)
Querying and setting status bits in STL (Page 2467)
Setting the status bit for word logic operations (Page 2470)

Shift and rotate

Shift

SSI: Shift word by word with sign

Description

Use the instruction "Shift word by word with sign" to shift the right word (bits 0 to 15) of accumulator 1 bit by bit to the right. During the shifting, positions that become free are padded with the signal state of bit 15 (sign bit for INT numbers). The bits 16 to 31 of accumulator 1 remain unchanged.

The following options are available for specifying the number of bit positions by which the bits are to be shifted:

- Specification of a positive integer as parameter of the instruction. The specified number must be in a value range from 0 to 15.
- Specification of the value of the right byte in accumulator 2. The byte is interpreted as a positive integer.

The instruction is executed regardless of the RLO. It resets the status bits CC0 and OV to "0" and the status bit CC1 is set to the signal state of the last bit that was shifted out.

If the specified number of places shifted is greater than 15, all bits in the right word of accumulator 1 are padded with the signal state of bit 15.

If the specified number of places shifted is equal to zero, the instruction is still executed. The status bit CC1 is set to "0".

Syntax

If the number of places shifted is specified as a parameter for the "Shift word by word with sign" instruction, the following syntax is used:

```
SSI <number of places shifted>
```

If the number of places shifted is specified using the right byte of accumulator 2, the following syntax is used for the instruction "Shift word by word with sign":

```
SSI
```

Parameters

The following table shows the parameters of the "Shift word by word with sign" instruction:

Parameters	Format	Description
<number of places shifted>	Positive integers SINT, INT, DINT, USINT, UINT, UDINT	Number of bit positions by which the value will be shifted

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand in accumulator 1.
SSI 6                        // Shift bits 0 to 15 of accumulator 1 six positions
                             to the right.
                             // Fill the now empty bit places with the state of bit
                             15.
T "Tag_Result_1"             // Transfer contents of accumulator 1 to the operand.
L 3                          // Load number of places shifted into accumulator 1
L "Tag_Value_2"              // Move number of places shifted to the right byte of
                             accumulator 2.
                             // Load value of the operand in accumulator 1.
SSI                          // Shift bits 0 to 15 of accumulator 1 three positions
                             to the right.
                             // Fill the now empty bit places with the state of bit
                             15.
T "Tag_Result_2"             // Transfer contents of accumulator 1 to the operand.
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	0101	1111	0110	0100	1001	1101	0011	1011
Tag_Result_1	0101	1111	0110	0100	1111	1110	0111	0100
Tag_Value_2	0101	1111	0110	0100	0101	1101	0010	1011
Tag_Result_2	0101	1111	0110	0100	0000	1011	1010	0101

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

SSD: Shift double word by double word with sign

Description

Use the instruction "Shift double word by double word with sign" to shift the complete contents of accumulator 1 bit by bit to the right. During the shifting, positions that become free are padded with the signal state of bit 31 (sign bit for DINT numbers).

The following options are available for specifying the number of bit positions by which the bits are to be shifted:

- Specification of a positive integer as parameter of the instruction. The specified number must be in a value range from 0 to 31.
- Specification of the value of the right byte in accumulator 2. The byte is interpreted as a positive integer.

The instruction is executed regardless of the RLO. It resets the status bits CC0 and OV to "0" and the status bit CC1 is set to the signal state of the last bit that was shifted out.

If the specified number of places shifted is greater than 31, all bits of accumulator 1 are padded with the signal state of bit 31.

If the specified number of places shifted is equal to zero, the instruction is still executed. The status bit CC1 is set to "0".

Syntax

If the number of places shifted is specified as a parameter for the "Shift double word by double word with sign" instruction, the following syntax is used:

```
SSD <number of places shifted>
```

If the number of places shifted is specified using the right byte of accumulator 2, the following syntax is used for the "Shift double word by double word with sign" instruction:

```
SSD
```

Parameters

The following table shows the parameters of the instruction "Shift double word by double word with sign":

Parameters	Format	Description
<number of places shifted>	Positive integers SINT, INT, DINT, USINT, UINT, UDINT	Number of bit positions by which the value will be shifted

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand in accumulator 1.
SSD 7                        // Shift bits 0 to 31 of accumulator 1 seven positions
                             to the right.
                             // Fill the now empty bit places with the state of bit
                             31.
T "Tag_Result_1"             // Transfer contents of accumulator 1 to the operand.
L 4                          // Load number of places shifted into accumulator 1
L "Tag_Value_2"              // Move number of places shifted to the right byte of
                             accumulator 2.
                             // Load value of the operand in accumulator 1.
SSD                          // Shift bits 0 to 31 of accumulator 1 four positions
                             to the right.
                             // Fill the now empty bit places with the state of bit
                             31.
T "Tag_Result_2"             // Transfer contents of accumulator 1 to the operand.
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	1000	1111	0110	0100	0101	1101	0011	1011
Tag_Result_1	1111	1111	0001	1110	1100	1000	1011	1010
Tag_Value_2	0010	1000	1010	0010	1001	1011	1100	1101
Tag_Result_2	0000	0010	1000	1010	0010	1001	1011	1100

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

SLW: Shift left word by word

Description

Use the "Shift left word by word" instruction to shift the right word (bits 0 to 15) of accumulator 1 bit by bit to the left. The bit positions that become free are padded with zeros. The bits 16 to 31 of accumulator 1 remain unchanged.

The following options are available for specifying the number of bit positions by which the bits are to be shifted:

- Specification of a positive integer as parameter of the instruction. The specified number must be in a value range from 0 to 15.
- Specification of the value of the right byte in accumulator 2. The byte is interpreted as a positive integer.

The instruction is executed regardless of the RLO. It resets the status bits CC0 and OV to "0" and the status bit CC1 is set to the signal state of the last bit that was shifted out.

If the specified number of places shifted is greater than 15, all bits in the right word of accumulator 1 are padded with zeros.

If the specified number of places shifted is equal to zero, the instruction is still executed. The status bit CC1 is set to "0".

Syntax

If the number of places shifted is specified as a parameter, the following syntax is used for the "Shift left word by word" instruction:

```
SLW <number of places shifted>
```

If the number of places shifted is specified using the right byte of accumulator 2, the following syntax is used for the "Shift left word by word" instruction:

```
SLW
```

Parameters

The following table shows the parameters of the "Shift left word by word" instruction:

Parameters	Format	Description
<number of places shifted>	Positive integers SINT, INT, DINT, USINT, UINT, UDINT	Number of bit positions by which the value will be shifted

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand in accumulator 1.
SLW 5                        // Shift bits 0 to 15 of accumulator 1 five positions
                             to the left.
                             // Fill the now empty bit places with zeros.
T "Tag_Result_1"             // Transfer contents of accumulator 1 to the operand.
L 4                          // Load number of places shifted into accumulator 1
L "Tag_Value_2"              // Move number of places shifted to the right byte of
                             accumulator 2.
                             // Load value of the operand in accumulator 1.
SLW                          // Shift bits 0 to 15 of accumulator 1 four positions
                             to the left.
                             // Fill the now empty bit places with zeros.
T "Tag_Result_2"             // Transfer contents of accumulator 1 to the operand.
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	0101	1111	0110	0100	0101	1101	0011	1011
Tag_Result_1	0101	1111	0110	0100	1010	0111	0110	0000
Tag_Value_2	0101	1111	0110	0100	0101	1101	0010	1011
Tag_Result_2	0101	1111	0110	0100	1101	0010	1011	0000

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

SRW: Shift right word by word

Description

Use the "Shift right word by word" instruction to shift the right word (bits 0 to 15) of accumulator 1 bit by bit to the right. The bit positions that become free are padded with zeros. The bits 16 to 31 of accumulator 1 remain unchanged.

The following options are available for specifying the number of bit positions by which the bits are to be shifted:

- Specification of a positive integer as parameter of the instruction. The specified number must be in a value range from 0 to 15.
- Specification of the value of the right byte in accumulator 2. The byte is interpreted as a positive integer.

The instruction is executed regardless of the RLO. It resets the status bits CC0 and OV to "0" and the status bit CC1 is set to the signal state of the last bit that was shifted out.

If the specified number of places shifted is greater than 15, all bits in the right word of accumulator 1 are padded with zeros.

If the specified number of places shifted is equal to zero, the instruction is still executed. The status bit CC1 is set to "0".

Syntax

If the number of places shifted is specified as a parameter, the following syntax is used for the "Shift right word by word" instruction:

```
SRW <number of places shifted>
```

If the number of places shifted is specified using the right word of accumulator 2, the following syntax is used for the "Shift right word by word" instruction:

```
SRW
```

Parameters

The following table shows the parameters of the "Shift right word by word" instruction:

Parameters	Format	Description
<number of places shifted>	Positive integers SINT, INT, DINT, USINT, UINT, UDINT	Number of bit positions by which the value will be shifted

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand in accumulator 1.
SRW 6                        // Shift bits 0 to 15 of accumulator 1 six positions
                             to the right.
                             // Fill the now empty bit places with zeros.
T "Tag_Result_1"             // Transfer contents of accumulator 1 to the operand.
L 4                          // Load number of places shifted into accumulator 1
L "Tag_Value_2"              // Move number of places shifted to the right byte of
                             accumulator 2.
                             // Load value of the operand in accumulator 1.
SRW                          // Shift bits 0 to 15 of accumulator 1 four positions
                             to the right.
                             // Fill the now empty bit places with zeros.
T "Tag_Result_2"             // Transfer contents of accumulator 1 to the operand.
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	0101	1111	0110	0100	0101	1101	0011	1011
Tag_Result_1	0101	1111	0110	0100	0000	0001	0111	0100
Tag_Value_2	0101	1111	0110	0100	0101	1101	0010	1011
Tag_Result_2	0101	1111	0110	0100	0000	0101	1101	0010

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

SLD: Shift left double word by double word

Description

Use the "Shift left double word by double word" instruction to shift the complete contents of accumulator 1 bit by bit to the left. The bit positions that become free are padded with zeros.

The following options are available for specifying the number of bit positions by which the bits are to be shifted:

- Specification of a positive integer as parameter of the instruction. The specified number must be in a value range from 0 to 31.
- Specification of the value of the right byte in accumulator 2. The byte is interpreted as a positive integer.

The instruction is executed regardless of the RLO. It resets the status bits CC0 and OV to "0" and the status bit CC1 is set to the signal state of the last bit that was shifted out.

If the specified number of places shifted is greater than 31, all bits of accumulator 1 are padded with zeros.

If the specified number of places shifted is equal to zero, the instruction is still executed. The status bit CC1 is set to "0".

Syntax

If the number of places shifted is specified as a parameter, the following syntax is used for the "Shift left double word by double word" instruction:

```
SLD <number of places shifted>
```

If the number of places shifted is specified using the right byte of accumulator 2, the following syntax is used for the "Shift left double word by double word" instruction:

```
SLD
```

Parameters

The following table shows the parameters of the "Shift left double word by double word" instruction:

Parameters	Format	Description
<number of places shifted>	Positive integers SINT, INT, DINT, USINT, UINT, UDINT	Number of bit positions by which the value will be shifted

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand in accumulator 1.
SLD 5                        // Shift bits 0 to 31 of accumulator 1 five positions
                             to the left.
                             // Fill the now empty bit places with zeros.
T "Tag_Result_1"             // Transfer contents of accumulator 1 to the operand.
L 4                          // Load number of places shifted into accumulator 1
```

```
STL                          Explanation
L "Tag_Value_2"              // Move number of places shifted to the right byte of
                             accumulator 2.
                             // Load value of the operand in accumulator 1.
SLD                          // Shift bits 0 to 31 of accumulator 1 four positions
                             to the left.
                             // Fill the now empty bit places with zeros.
T "Tag_Result_2"             // Transfer contents of accumulator 1 to the operand.
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	0101	1111	0110	0100	0101	1101	0011	1011
Tag_Result_1	1110	1100	1000	1011	1010	0111	0110	0000
Tag_Value_2	1010	1000	1010	0010	1001	1011	1100	1101
Tag_Result_2	1000	1010	0010	1001	1011	1100	1101	0000

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

SRD: Shift right double word by double word

Description

Use the instruction "Shift right double word by double word" to shift the complete contents of accumulator 1 bit by bit to the right. The bit positions that become free are padded with zeros.

The following options are available for specifying the number of bit positions by which the bits are to be shifted:

- Specification of a positive integer as parameter of the instruction. The specified number must be in a value range from 0 to 31.
- Specification of the value of the right byte in accumulator 2. The byte is interpreted as a positive integer.

The instruction is executed regardless of the RLO. It resets the status bits CC0 and OV to "0" and the status bit CC1 is set to the signal state of the last bit that was shifted out.

If the specified number of places shifted is greater than 31, all bits of accumulator 1 are padded with zeros.

If the specified number of places shifted is equal to zero, the instruction is still executed. The status bit CC1 is set to "0".

Syntax

If the number of places shifted is specified as a parameter, the following syntax is used for the "Shift right double word by double word" instruction:

```
SRD <number of places shifted>
```

If the number of places shifted is specified using the right byte of accumulator 2, the following syntax is used for the "Shift right double word by double word" instruction:

```
SRD
```

Parameters

The following table shows the parameters of the "Shift right double word by double word" instruction:

Parameters	Format	Description
<number of places shifted>	Positive integers SINT, INT, DINT, USINT, UINT, UDINT	Number of bit positions by which the value will be shifted

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand in accumulator 1.
SRD 7                        // Shift bits 0 to 31 of accumulator 1 seven positions
                             to the right.
                             // Fill the now empty bit places with zeros.
T "Tag_Result_1"             // Transfer contents of accumulator 1 to the operand.
L 4                          // Load number of places shifted into accumulator 1
L "Tag_Value_2"              // Move number of places shifted to the right byte of
                             accumulator 2.
                             // Load value of the operand in accumulator 1.
SRD                          // Shift bits 0 to 31 of accumulator 1 four positions
                             to the right.
                             // Fill the now empty bit places with zeros.
T "Tag_Result_2"             // Transfer contents of accumulator 1 to the operand.
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	0101	1111	0110	0100	0101	1101	0011	1011
Tag_Result_1	0000	0000	1011	1110	1100	1000	1011	1010
Tag_Value_2	1010	1000	1010	0010	1001	1011	1100	1101
Tag_Result_2	0000	1010	1000	1010	0010	1001	1011	1100

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Rotate

RLD: Rotate left double word by double word

Description

Use the "Rotate left double word by double word" instruction to rotate the complete contents of accumulator 1 bit by bit to the left. During processing the instruction shifts bit 0 to 31 of accumulator 1 bit by bit to the left and pads the positions that become free during writing with the pushed out bit positions.

The following options are available for specifying the number of bit positions by which the bits are to be rotated:

- Specification of a positive integer as parameter of the instruction.
- Specification of the value of the right byte in accumulator 2. The byte is interpreted as a positive integer.

The instruction is executed regardless of the RLO. It resets the status bits CC0 and OV to "0" and the status bit CC1 is set to the signal state of the last bit that was shifted out.

If the specified rotate number is 32, the contents of accumulator 1 remain unchanged. For values greater than 32, the rotation number is calculated with a modulo division by 32. If, for example, the rotate number is 34, the bits of accumulator 1 are rotated by 2 bit positions.

If the specified rotate number is equal to zero, the instruction is still executed. The status bit CC1 is set to "0".

Syntax

If the number of places rotated is specified as a parameter, the following syntax is used for the "Rotate left double word by double word" instruction:

```
RLD <rotate number>
```

If the number of places rotated is specified using the right byte of accumulator 2, the following syntax is used for the "Rotate left double word by double word" instruction:

```
RLD
```

Parameters

The following table shows the parameters of the "Rotate left double word by double word" instruction:

Parameters	Format	Description
<rotate number>	Positive integers SINT, INT, DINT, USINT, UINT, UDINT	Number of bit positions to be rotated.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand in accumulator 1.
RLD 4                        // Rotate bits 0 to 31 of accumulator 1 four positions
                             to the left
T "Tag_Result_1"             // Transfer contents of accumulator 1 to the operand.
L 6                          // Load number of places to be rotated in accumulator
                             1.
L "Tag_Value_2"              // Move number of places to be rotated to right byte
                             of accumulator 2
                             // Load value of the operand in accumulator 1.
RLD                          // Rotate bits 0 to 31 of accumulator 1 six positions
                             to the left
T "Tag_Result_2"             // Transfer contents of accumulator 1 to the operand.
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	**0101**	1111	0110	0100	0101	1101	0011	1011
Tag_Result_1	1111	0110	0100	0101	1101	0011	1011	**0101**

Operand	Value							
Tag_Value_2	**1010**	**10**00	1010	0010	1001	1011	1100	1101
Tag_Result_2	0010	1000	1010	0110	1111	0011	01**10**	**1010**

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

RRD: Rotate right double word by double word

Description

Use the "Rotate right double word by double word" instruction to rotate the complete contents of accumulator 1 bit by bit to the right. During processing the instruction shifts bits 0 to 31 of accumulator 1 bit by bit to the right and pads the positions that become free during writing with the pushed out bit positions.

The following options are available for specifying the number of bit positions by which the bits are to be rotated:

- Specification of a positive integer as parameter of the instruction.
- Specification of the value of the right byte in accumulator 2. The byte is interpreted as a positive integer.

The instruction is executed regardless of the RLO. It resets the status bits CC0 and OV to "0" and the status bit CC1 is set to the signal state of the last bit that was shifted out.

If the specified rotate number is 32, the contents of accumulator 1 remain unchanged. For values greater than 32, the rotation number is calculated with a modulo division by 32. If, for example, the rotate number is 34, the bits of accumulator 1 are rotated by 2 bit positions.

If the specified rotate number is equal to zero, the instruction is still executed. The status bit CC1 is set to "0".

Syntax

If the number of places rotated is specified as a parameter, the following syntax is used for the "Rotate right double word by double word" instruction:

```
RRD <rotate number>
```

If the number of places rotated is specified using the right byte of accumulator 2, the following syntax is used for the "Rotate right double word by double word" instruction:

```
RRD
```

Parameters

The following table shows the parameters of the instruction "Rotate right double word by double word":

Parameters	Format	Description
<rotate number>	Positive integers SINT, INT, DINT, USINT, UINT, UDINT	Number of bit positions to be rotated.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand in accumulator 1.
RRD 4                        // Rotate bits 0 to 31 of accumulator 1 four places to
                             the right.
T "Tag_Result_1"             // Transfer contents of accumulator 1 to the operand.
L 6                          // Load number of places to be rotated in accumulator
                             1.
L "Tag_Value_2"              // Move number of places to be rotated to right byte
                             of accumulator 2
                             // Load value of the operand in accumulator 1.
RRD                          // Rotate bits 0 to 31 of accumulator 1 six places to
                             the right.
T "Tag_Result_2"             // Transfer contents of accumulator 1 to the operand.
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	0101	1111	0110	0100	0101	1101	0011	**1011**
Tag_Result_1	**1011**	0101	1111	0110	0100	0101	1101	0011
Tag_Value_2	1010	1000	1010	0010	1001	1011	11**00**	**1101**
Tag_Result_2	**0011**	**01**10	1010	0010	1000	1010	0110	1111

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

RLDA: Rotate left by status bit CC 1

Description

You can use the "Rotate left by status bit CC 1" instruction to shift the contents of accumulator 1 one bit to the left. The bit positions (0) that become free during shifting are padded with the signal state of status bit CC 1. Status bit CC 1 receives the signal state of the shifted out bit (31).

The instruction is executed regardless of the RLO. It resets the status bit CC 0 to "0".

Syntax

The following syntax is used for the "Rotate left by status bit CC 1" instruction:

```
RLDA
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value"	// Load value of the operand in accumulator 1.
RLDA	// Rotate bits 0 to 31 of accumulator 1 one place to the left
	// Fill bit 0 with the signal state of status bit CC 1.
	// Write the shifted-out bit to the status bit CC 1.
T "Tag_Result"	// Transfer contents of accumulator 1 to the operand.

The following table shows how the instruction works using specific operand values:

Operand	CC 1	Value							
Tag_Value	1	0101	1111	0110	0100	0101	1101	0011	1011
Tag_Result	0	1011	1110	1100	1000	1011	1010	0111	0111

See also

Overview of the valid data types (Page 2236)
Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)
Evaluating errors with output parameter RET_VAL (Page 5370)
Call function blocks as single or multi-instances (Page 2182)
Basic information on the status word (Page 2466)
Querying and setting status bits in STL (Page 2467)

RRDA: Rotate right by status bit CC 1

Description

Use the "Rotate right by status bit CC 1" instruction to shift the contents of accumulator 1 one bit to the right. The bit positions (31) that become free during shifting are padded with the signal state of status bit CC 1. Status bit CC 1 receives the signal state of the shifted out bit (0).

The instruction is executed regardless of the RLO. It resets the status bits CC0 and OV to "0".

Syntax

The following syntax is used for the "Rotate right by status bit CC 1" instruction:

```
RRDA
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value"	// Load value of the operand in accumulator 1.
RRDA	// Rotate bits 0 to 31 of accumulator 1 one place to the right
	// Fill bit 31 with the signal state of status bit CC 1.
	// Write the shifted-out bit to the status bit CC 1.
T "Tag_Result"	// Transfer contents of accumulator 1 to the operand.

The following table shows how the instruction works using specific operand values:

Operand	CC 1	Value							
Tag_Value	1	0101	1111	0110	0100	0101	1101	0011	1011
Tag_Result	0	1010	1111	1011	0010	0010	1110	1001	1101

Status bit CC 1 has the value "1" because the bit 0 of the "Tag_Value" operand also has the value "1".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Additional instructions

Accumulators

TAK: Swap content of accumulators 1 and 2

Description

Use the instruction "Swap content of accumulators 1 and 2" to switch the contents accumulator 1 with the contents of accumulator 2.

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

Syntax

Use the following syntax for the "Swap content of accumulators 1 and 2" instruction:

```
TAK
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.

STL	Explanation
	// Load value of the operand "Tag_Value_2" in accumulator 1
>I	// Compare whether the value of accumulator 2 is greater than the value of accumulator 1.
JC NEXT	// If condition is satisfied (RLO = "1"), jump to jump label NEXT.
	// If the condition is not satisfied(RLO = "0"), execute the next instruction.
TAK	// Swap the contents of accumulators 1 and 2
NEXT: -I	// Jump label "NEXT"
	// Subtract value of accumulator 1 from value of accumulator 2.
T "Tag_Output"	// Transfer contents of accumulator 1 to the operand "Tag_Output".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

PUSH: Shift content to the next higher accumulator

Description

Use the instruction "Shift content to the next higher accumulator" to move the contents of accumulator 1 to accumulator 2.

The content of accumulator 1 is not influenced by the instruction and remains unchanged after processing the instruction. The content of accumulator 2 will be lost.

The following table shows the contents of accumulators 1 to 2 before and after the executing the instruction:

Status	Accumulator	
	1	2
Before processing	Value A	Value B
After processing	Value A	Value A

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

Syntax

Use the following syntax for the instruction "Shift content to the next higher accumulator":

```
PUSH
```

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Value_1"`	`// Load value of the operand in accumulator 1.`
`PUSH`	`// Shift contents of accumulators 1 to 2 to the next higher accumulator.`
`L "Tag_Value_2"`	`// Load value of the operand in accumulator 1.`

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

POP: Shift content to the next lower accumulator

Description

Use the instruction "Shift content to the next lower accumulator" to move the contents of accumulator 2 to accumulator 1.

The content of accumulator 2 is not influenced by the instruction and remains unchanged after processing the instruction. The content of accumulator 1 will be lost.

The following table shows the contents of accumulators 1 to 2 before and after the executing the instruction:

Status	Accumulator	
	1	2
Before processing	Value A	Value B
After processing	Value B	Value B

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

Syntax

Use the following syntax for the instruction "Shift content to the next lower accumulator":

```
POP
```

Example

The following example shows how the instruction works:

STL	Explanation
`T "Tag_Value_1"`	`// Transfer contents of accumulator 1 to the operand "Tag_Value_1".`
`POP`	`// Shift the contents of accumulator 2 to lower-order accumulator 1.`
`T "Tag_Value_2"`	`// Transfer contents of accumulator 1 to the operand "Tag_Value_2".`

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Address register

+AR1: Add accumulator 1 to AR1

Description

Use the "Add accumulator 1 to AR1" instruction to add a value to the content of address register 1. The type of pointer in address register 1 and the operand range are retained.

The following options are available for specifying the value to be added:

- Specification via a constant: The instruction adds the value of the constant to address register 1. The value of the constant must correspond to the format of the range-internal pointer (POINTER).
- Specification via the value of the right word in the accumulator 1: The instruction interprets the value of the right word in accumulator 1 as a 16-bit integer and extends it with the correct sign to 24 bits. The value of accumulator 1 then adds the instruction to address register 1. Values in the range from -32 768 to +32 767 are permitted.

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Syntax

If the addition value is specified using a constant, use the following syntax for the "Add accumulator 1 to AR1" instruction:

```
+AR1 <constant>
```

If the addition value is specified using the right word of accumulator 1, use the following syntax for the "Add accumulator 1 to AR1" instruction:

```
+AR1
```

Parameters

The following table shows the parameters of the "Add accumulator 1 to AR1" instruction:

Parameter	Data type	Description
<constant>	POINTER	Value that is added to address register 1.

Example

The following example shows how the instruction works:

STL	Explanation
+AR1 P#10.0	// Add pointer to address register 1.
L "Tag_Value"	// Load value of the operand "Tag_Value" into the right word of accumulator 1.
+AR1	// Add value of accumulator 1 to address register 1.
TAR1 %MD24	// Transfer contents of address register 1 to double word MD24.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

+AR2: Add accumulator 1 to AR2

Description

Use the "Add accumulator 1 to AR2" instruction to add a value to the content of address register 2. The type of pointer in address register 2 and the operand range are retained.

The following options are available for specifying the value to be added:

- Specification via a constant: The instruction adds the value of the constant to address register 2. The value of the constant must correspond to the format of the range-internal pointer (POINTER).
- Specification via the value of the right word in the accumulator 1: The instruction interprets the value of the right word in accumulator 1 as a 16-bit integer and extends it with the correct sign to 24 bits. The value of accumulator 1 then adds the instruction to address register 2. Values in the range from -32 768 to +32 767 are permitted.

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Note

Address register AR2 is used when processing multiple instances. If you program the "Add accumulator 1 to AR2" instruction, you must first "save" the contents of address register 2 and reload it later.

Syntax

If the addition value is specified using a constant, use the following syntax for the "Add accumulator 1 to AR2" instruction:

```
+AR2 <constant>
```

If the addition value is specified using the right word of accumulator 1, use the following syntax for the "Add accumulator 1 to AR2" instruction:

```
+AR2
```

Parameters

The following table shows the parameters of the "Add accumulator 1 to AR2" instruction:

Parameter	Data type	Description
<constant>	POINTER	Value that is added to address register 1.

Example

The following example shows how the instruction works:

STL	Explanation
+AR2 P#10.0	// Add pointer to address register 2.
L "Tag_Value"	// Load value of the operand "Tag_Value" into the right word of accumulator 1.
+AR2	// Add value of accumulator 1 to address register 2.
TAR2 %MD24	// Transfer contents of address register 1 to double word MD24.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

Null instructions

BLD: Program display (null instruction)

Description

The instruction "Program display (null instruction)" does not execute any function and does not affect the status bits. The instruction is used to recognize code sequences during a parameter transfer or with LAD/FBD networks. It is created automatically when a LAD or FBD program is displayed in STL. The parameter value is the number of the instruction and is generated by the programming device.

Syntax

Use the following syntax for the "Program display (null instruction)" instruction:

```
BLD <number>
```

Parameters

The following table shows the parameters of the "Program display (null instruction)" instruction:

Parameters	Data type	Description
<number>	WORD	Number of the instruction

See also

Overview of the valid data types (Page 2236)
Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)
Evaluating errors with output parameter RET_VAL (Page 5370)
Call function blocks as single or multi-instances (Page 2182)
Basic information on the status word (Page 2466)
Querying and setting status bits in STL (Page 2467)

NOP 0: Null instruction

Description

If "Null instruction" has a parameter 0, it does not execute any function and does not affect the status bits. The instruction code contains a bit pattern with 16 zeros. The instruction is only relevant for the programming device when a program is displayed.

Syntax

Use the following syntax for "Null instruction" with a parameter 0:

```
NOP 0
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
>I	// Compare whether the value of accumulator 2 is greater than the value of accumulator 1.
A	// Query the signal state of the operand for "1" and AND with the current RLO.
JC NEXT	// If condition is satisfied (RLO = "1"), jump to jump label NEXT.
	// If the condition is not satisfied(RLO = "0"), execute the next instruction.
TAK	// Swap the contents of accumulators 1 and 2
NEXT: NOP 0	// Jump label "NEXT"

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Querying and setting status bits in STL (Page 2467)

NOP 1: Null instruction

Description

If "Null instruction" has a parameter 1, it does not execute any function and does not affect the status bits. The instruction code contains a bit pattern with 16 ones. The instruction is only relevant for the programming device when a program is displayed.

Syntax

You can use the following syntax for "Null instruction" with a parameter 1:

```
NOP 1
```

See also

Overview of the valid data types (Page 2236)
Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)
Evaluating errors with output parameter RET_VAL (Page 5370)
Call function blocks as single or multi-instances (Page 2182)
Basic information on the status word (Page 2466)
Querying and setting status bits in STL (Page 2467)

SCL

Bit logic operations

R_TRIG: Set tag on positive signal edge

Description

You can use the "Set tag on positive signal edge" instruction to set a specified tag in the instance DB when there is a "0" to "1" change in the result of logic operation (RLO). The instruction compares the current RLO at the input CLK with the RLO from the previous query, which is saved in the specified instance DB. If the instruction detects a change in the result of logic operation (RLO) from "0" to "1", there is a positive, rising edge.

If a positive edge is detected, the tag in the instance DB is set to signal state "1" and the output Q returns the signal state "1" In all other cases, the signal state at the output of the instruction is "0".

When you insert the instruction in the program, the "Call options" dialog opens automatically. In this dialog you can specify whether the edge memory bit is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Set tag on positive signal edge" instruction:

```
SCL
<Instance_DB>(CLK := <Operand>,
              Q => <Operand>)
```

Parameter

The following table shows the parameters of the instruction "Set tag on positive signal edge":

Parameter	Declaration	Data type	Description
CLK	Input	BOOL	Incoming signal, the edge of which is to be queried
Q	Output	BOOL	Result of edge evaluation

Example

The following example shows how the instruction works:

```
SCL
"R_TRIG_DB"(CLK := "TagIn",
            Q => "TagOut");
```

The RLO of the preceding query is saved in the instance DB "R_TRIG_DB". If a change in the signal state of the RLO from "0" to "1" is detected in the operand "TagIn", the output "TagOut" has signal state "1".

See also

Overview of the valid data types (Page 2236)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

F_TRIG: Set tag on negative signal edge

Description

You can use the "Set tag on negative signal edge" instruction to set a specified tag in the instance DB when there is a "1" to "0" change in the result of logic operation (RLO). The instruction compares the current RLO at the input CLK with the RLO from the previous query,

which is saved in the specified instance DB. If the instruction detects a change in the result of logic operation (RLO) from "1" to "0", there is a negative, falling edge.

If a negative edge is detected, the tag in the instance DB is set to signal state "1" and the output Q returns the signal state "1" In all other cases, the signal state at the output of the instruction is "0".

When you insert the instruction in the program, the "Call options" dialog opens automatically. In this dialog you can specify whether the edge memory bit is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Set tag on negative signal edge" instruction:

SCL

```
<Instance_DB>(CLK := <Operand>,
              Q => <Operand>)
```

Parameter

The following table shows the parameters of the instruction "Set tag on negative signal edge":

Parameter	Declaration	Data type	Description
CLK	Input	BOOL	Incoming signal, the edge of which is to be queried
Q	Output	BOOL	Result of edge evaluation

Example

The following example shows how the instruction works:

SCL

```
"F_TRIG_DB"(CLK := "TagIn",
            Q => "TagOut");
```

The RLO of the preceding query is saved in the instance DB "F_TRIG_DB". If a change in the signal state of the RLO from "1" to "0" is detected in the operand "TagIn", the output "TagOut" has signal state "1".

See also

Overview of the valid data types (Page 2236)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)
Entering SCL instructions (Page 2697)
Editing SCL instructions (Page 2716)

Timer operations

IEC Timers

TP: Generate pulse

Description

You can use the "Generate pulse" instruction to set the Q parameter for the duration PT. The instruction starts when the result of logic operation (RLO) of the IN parameter changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. The Q parameter is set for the time PT, regardless of the subsequent changes in the input signal. Even when a new positive signal edge is detected, the signal state of the Q parameter is not affected as long as the time duration PT is active.

The current time value can be queried in the ET parameter. The time value starts at T#0s and ends when the value of the time duration PT is reached. When the time duration PT is reached and the signal state at the IN parameter is "0", the ET parameter is reset.

Note

If the timer is not called in the program because it is skipped, for example, output ET returns a constant value as soon as the timer has expired.

Each call of the "Generate pulse" instruction must be assigned to an IEC timer in which the instruction data is stored.

For S7-1200 CPU

An IEC Timer is a structure of the data type IEC_TIMER or TP_TIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the TP_TIME type in the "Static" section of a block (for example, #MyTP_TIMER)

For S7-1500 CPU

An IEC timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TP_TIME or TP_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TP_TIME or TP_LTIME in the "Static" section of a block (for example #MyTP_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when an instruction is called and also at each access to the Q or ET outputs.

Syntax

Use the following syntax for the "Generate pulse" instruction:

- Data block of system data type IEC_Timer (global DB):

SCL

```
<IEC_Timer_DB> TP(IN := <Operand>,
                  PT := <Operand>,
                  Q => <Operand>,
                  ET => <Operand>)
```

- Local tag:

SCL

```
#myLocal_timer(IN := <Operand>,
               PT := <Operand>,
               Q => <Operand>,
               ET => <Operand>)
```

The syntax of the instruction consists of the following parts:

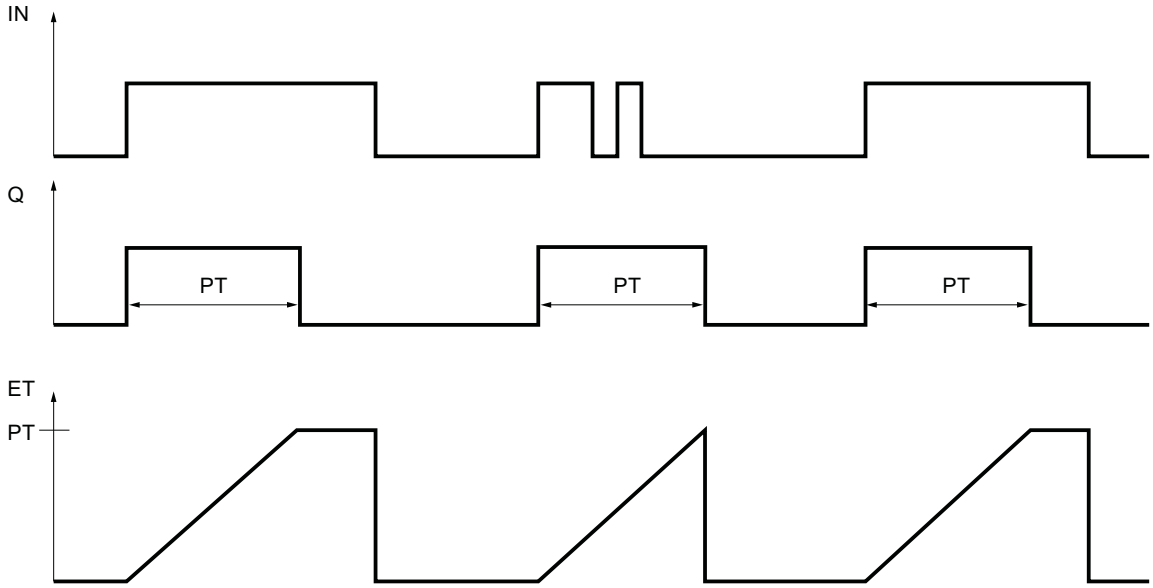
Parameter	Declaration	Data type		Description
		S7-1200	S7-1500	
IN	Input	BOOL	BOOL	Start input
PT	Input	TIME	TIME, LTIME	Duration of the pulse. The value of the PT parameter must be positive.

Parameter	Declaration	Data type		Description
		S7-1200	S7-1500	
Q	Output	BOOL	BOOL	Operand that is set for the PT duration.
ET	Output	TIME	TIME, LTIME	Current time value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Generate pulse" instruction:



Example

The following example shows how the instruction works:

```
SCL
"TP_DB".TP(IN := "Tag_Start",
           PT := "Tag_PresetTime",
           Q => "Tag_Status",
           ET => "Tag_ElapsedTime");
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the time period programmed for the PT parameter is started and the "Tag_Status" operand is set to "1". The current time value is stored in the "Tag_ElapsedTime" operand.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Call function blocks as single or multi-instances (Page 2182)

TON: Generate on-delay

Description

You can use the "Generate on-delay" instruction to delay the setting of the Q parameter for the programmed duration PT. The instruction starts when the result of logic operation (RLO) of the IN parameter changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. When the duration PT has expired, the Q parameter returns signal state "1". The Q parameter remains set as long as the start input is still "1". If the signal state of the IN parameter changes from "1" to "0", the parameter Q will be reset. The timer function is restarted when a new positive signal edge is detected at the IN parameter.

The current time value can be queried in the ET parameter. The time value starts at T#0s and ends when the value of the time duration PT is reached. The ET parameter is reset as soon as the signal state of the IN parameter changes to "0".

Note

If the timer is not called in the program because it is skipped, for example, output ET returns a constant value as soon as the timer has expired.

Each call of the "Generate on-delay" instruction must be assigned to an IEC timer in which the instruction data is stored.

For S7-1200 CPU

An IEC Timer is a structure of the data type IEC_TIMER or TON_TIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the TON_TIME type in the "Static" section of a block (for example, #MyTON_TIMER)

For S7-1500 CPU

An IEC timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TON_TIME or TON_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TON_TIME or TON_LTIME in the "Static" section of a block (for example #MyTON_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find

it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when an instruction is called and also at each access to the Q or ET outputs.

Syntax

Use the following syntax for the "Generate on-delay" instruction:

- Data block of system data type IEC_Timer (global DB):

SCL

```
<IEC_Timer_DB> TON(IN := <Operand>,
                   PT := <Operand>,
                   Q => <Operand>,
                   ET => <Operand>)
```

- Local tag:

SCL

```
#myLocal_timer(IN := <Operand>,
               PT := <Operand>,
               Q => <Operand>,
               ET => <Operand>)
```

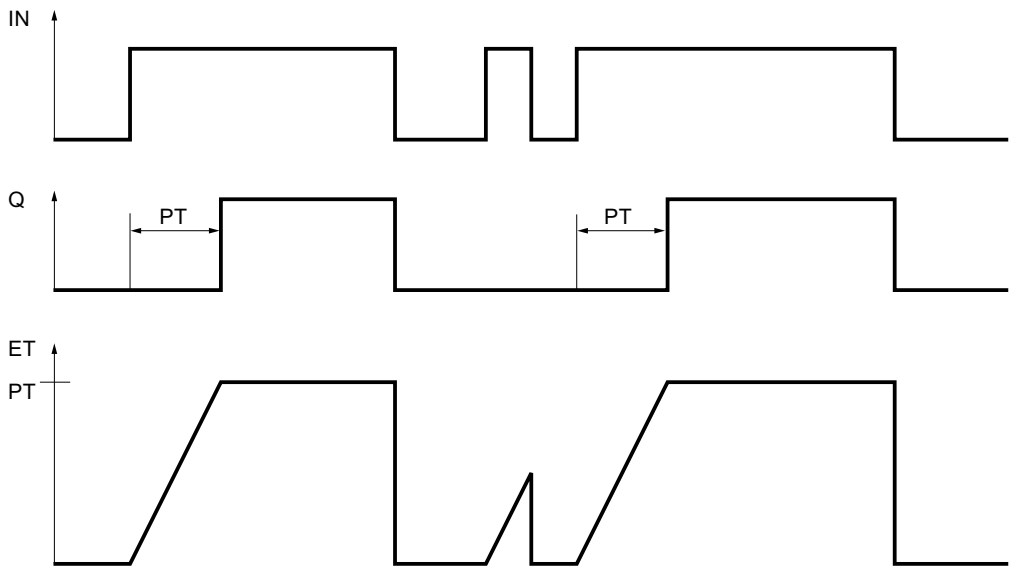
The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type		Description
		S7-1200	S7-1500	
IN	Input	BOOL	BOOL	Start input
PT	Input	TIME	TIME, LTIME	Duration of the on delay. The value of the PT parameter must be positive.
Q	Output	BOOL	BOOL	Operand that is set when the timer PT expires.
ET	Output	TIME	TIME, LTIME	Current time value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Generate on-delay" instruction:



Example

The following example shows how the instruction works:

SCL

```
"TON_DB".TON(IN := "Tag_Start",
             PT := "Tag_PresetTime",
             Q => "Tag_Status",
             ET => "Tag_ElapsedTime");
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the time programmed for the PT parameter is started. After the end of the period the "Tag_Status" operand is set to the signal state "1". The operand Tag_Status remains set to signal state "1" as long as the operand Tag_Start has signal state "1". The current time value is stored in the operand "Tag_ElapsedTime". When the signal state at the operand "Tag_Start" changes from "1" to "0", the "Tag_Status" operand is reset.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Call function blocks as single or multi-instances (Page 2182)

TOF: Generate off-delay

Description

You can use the "Generate off-delay" instruction to delay the resetting of the Q parameter for the programmed duration PT. The Q parameter is set when the result of logic operation (RLO) of the IN parameter changes from "0" to "1" (positive signal edge). When the signal state of the IN parameter changes back to "0", the programmed time PT starts. The Q parameter remains set as long the time duration PT is running. When the time PT expires, the Q parameter is reset. If the signal state of the IN parameter changes to "1" before the time duration PT has expired, the timer is reset. The signal state of the Q parameter remains set to "1".

The current time value can be queried in the ET parameter. The time value starts at T#0s and ends when the value of the time duration PT is reached. When the time duration PT expires, the ET parameter remains set to the current value until the IN parameter changes back to "1". If the IN parameter changes to "1" before the time PT has expired, the ET parameter is reset to the value T#0s.

Note

If the timer is not called in the program because it is skipped, for example, output ET returns a constant value as soon as the timer has expired.

Each call of the "Generate off-delay" instruction must be assigned to an IEC timer in which the instruction data is stored.

For S7-1200 CPU

An IEC Timer is a structure of the data type IEC_TIMER or TOF_TIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the TOF_TIME type in the "Static" section of a block (for example, #MyTOF_TIMER)

For S7-1500 CPU

An IEC timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TOF_TIME or TOF_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TOF_TIME or TOF_LTIME in the "Static" section of a block (for example #MyTOF_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when an instruction is called and also at each access to the Q or ET outputs.

Syntax

Use the following syntax for the "Generate off-delay" instruction:

- Data block of system data type IEC_Timer (global DB):

SCL

```
<IEC_Timer_DB> TOF(IN := <Operand>,
                   PT := <Operand>,
                   Q => <Operand>,
                   ET => <Operand>)
```

- Local tag:

SCL

```
#myLocal_timer(IN := <Operand>,
               PT := <Operand>,
               Q => <Operand>,
               ET => <Operand>)
```

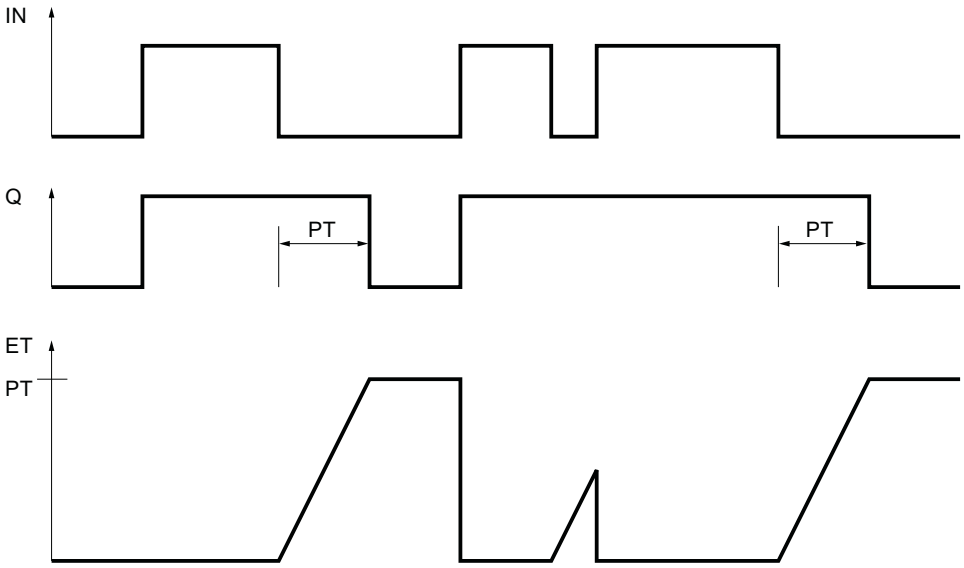
The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type		Description
		S7-1200	S7-1500	
IN	Input	BOOL	BOOL	Start input
PT	Input	TIME	TIME, LTIME	Duration of the off delay. The value of the PT parameter must be positive.
Q	Output	BOOL	BOOL	Operand that is reset when the time PT expires.
ET	Output	TIME	TIME, LTIME	Current time value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the instruction "Generate off-delay":



Example

The following example shows how the instruction works:

```
SCL
"TOF_DB".TOF(IN := "Tag_Start",
             PT := "Tag_PresetTime",
             Q => "Tag_Status",
             ET => "Tag_ElapsedTime");
```

With a change in the signal state of the "Tag_Start" operand from "0" to "1", the "Tag_Status" operand is set. When the signal state of the "Tag_Start" operand changes from "1" to "0", the time programmed for the PT parameter is started. As long as the time is running, the "Tag_Status" operand remains set. When the time has expired, the "Tag_Status" operand is reset. The current time value is stored in the operand "Tag_ElapsedTime".

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Call function blocks as single or multi-instances (Page 2182)

TONR: Time accumulator

Description

The "Time accumulator" instruction is used to accumulate time values within a period set by the PT parameter. When the signal state of the IN parameter changes to "1", the instruction executes and the time duration PT starts. While the time duration PT is running, the time values that are recorded when the IN parameter has signal state "1" are accumulated. The accumulated time is output in the ET parameter and can be queried there. When the time duration PT is reached, the Q parameter has signal state "1". The Q parameter remains set to "1", even when the signal state at the IN parameter changes to "0".

The R parameter resets the ET and Q parameters regardless of the signal state at the IN parameter.

Each call of the "Time accumulator" instruction must be assigned to an IEC timer in which the instruction data is stored.

For S7-1200 CPU

An IEC Timer is a structure of the data type IEC_TIMER or TONR_TIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the TONR_TIME type in the "Static" section of a block (for example, #MyTONR_TIMER)

For S7-1500 CPU

An IEC timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TONR_TIME or TONR_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TONR_TIME or TONR_LTIME in the "Static" section of a block (for example #MyTONR_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when an instruction is called and also at each access to the Q or ET outputs.

Syntax

Use the following syntax for the "Time accumulator" instruction:

- Data block of system data type IEC_Timer (global DB):

```
SCL
<IEC_Timer_DB> TONR(IN := <Operand>,
                    R := <Operand>,
                    PT := <Operand>,
                    Q => <Operand>,
                    ET => <Operand>)
```

- Local tag:

```
SCL
#myLocal_timer(IN := <Operand>,
               R := <Operand>,
               PT := <Operand>,
               Q => <Operand>,
               ET => <Operand>)
```

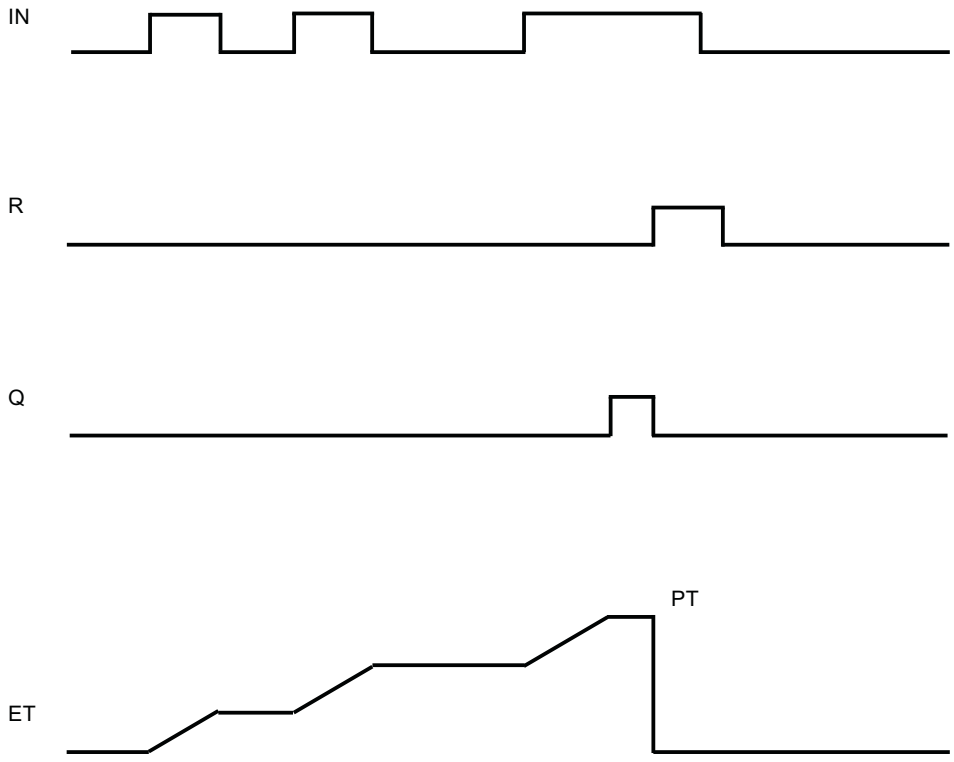
The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type		Description
		S7-1200	S7-1500	
IN	Input	BOOL	BOOL	Start input
R	Input	BOOL	BOOL	Reset of the ET and Q parameters
PT	Input	TIME	TIME, LTIME	Maximum duration of time recording. The value of the PT parameter must be positive.
Q	Output	BOOL	BOOL	Operand that remains set when the timer PT has expired.
ET	Output	TIME	TIME, LTIME	Accumulated time

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Time accumulator" instruction:



Example

The following example shows how the instruction works:

SCL

```
"TONR_DB".TONR(IN := "Tag_Start",
               R := "Tag_Reset",
               PT := "Tag_PresetTime",
               Q => "Tag_Status",
               ET => "Tag_Time");
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the time programmed for the PT parameter is started. While the timer is running, the time values that are recorded at signal state "1" of the operand "Tag_Start" is accumulated. The accumulated times is stored in the "Tag_Time" operand. When the time value displayed at the PT parameter is reached, the "Tag_Status" operand is set to the signal state "1". The current time value is stored in the operand "Tag_Time".

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Call function blocks as single or multi-instances (Page 2182)

RESET_TIMER: Reset timer

Description

You can use the "Reset timer" instruction to reset an IEC timer to "0". The structure components of the timer in the specified data block are reset to "0".

The instruction does not influence the RLO. At the TIMER parameter, the "Reset timer" instruction is assigned an IEC timer declared in the program.

The instruction data is updated only when the instruction is called and not each time the assigned IEC timer is accessed. Querying the data is only identical from the call of the instruction to the next call of the instruction.

Syntax

Use the following syntax for the "Reset timer" instruction:

SCL

```
RESET_TIMER(TIMER := <IEC timer>)
```

Parameters

The following table shows the parameters of the "Reset timer" instruction":

Parameter	Declaration	Data type	Description
<IEC timer>	Output	IEC_TIMER, IEC_LTIMER, TP_TIME, TP_LTIME, TON_TIME, TON_LTIME, TOF_TIME, TOF_LTIME, TONR_TIME, TONR_LTIME	IEC timer that is reset

For more information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
IF #started = false THEN
"TON_DB".TON(IN := "Tag_Start",
```

SCL

```
            PT := "Tag_PresetTime",
            Q => "Tag_Status",
            ET => "Tag_ElapsedTime");
#started := true;
END_IF;

IF "TON_DB".ET < T#25s THEN
RESET_TIMER(TIMER := "TON_DB");
#started := false;
END_IF;
```

When the tag #started has the signal state "0", the instruction "Generate on-delay" is executed when there is positive signal edge on the operand "Tag_Start". The IEC timer stored in the instance data block "TON_DB" is started with the time duration that is specified by the operand "Tag_PresetTime". The operand "Tag_Status" is set if the time duration specified by the operand "Tag_PresetTime" has expired. The parameter Q remains set as long as the operand "Tag_Start" still has the signal state "1". When the signal state of the start input changes from "1" to "0", the operand on the Q parameter is reset.

If the expired time of the IEC timer "TON_DB" is less than 25s, the "Reset timer" instruction is executed and the timer stored in the "TON_DB" instance data block is reset.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Call function blocks as single or multi-instances (Page 2182)

PRESET_TIMER: Load time duration

Description

You can use the "Load time duration" instruction to set the time for an IEC timer. The instruction is executed in every cycle when the result of logic operation (RLO) at the input of the instruction has the signal state "1". The instruction writes the specified time to the structure of the specified IEC timer.

Note

If the specified IEC timer is running while the instruction executes, the instruction overwrites the current time of the specified IEC timer. This can change the timer status of the IEC timer.

You assign an IEC timer declared in the program to the "Load time duration" instruction.

The instruction data is updated only when the instruction is called and each time the assigned IEC timer is accessed. The query on Q or ET (for example, "MyTimer".Q or "MyTimer".ET) updates the IEC_TIMER structure.

Syntax

Use the following syntax for the "Load time duration" instruction:

SCL

```
PRESET_TIMER(PT := <Operand>,
             TIMER := <IEC timer)
```

Parameter

The following table shows the parameters of the "Load time duration" instruction:

Parameter	Declaration	Data type	Description
<Time duration>	Input	TIME, LTIME	Duration with which the IEC timer runs.
<IEC timer>	Output	IEC_TIMER, IEC_LTIMER, TP_TIME, TP_LTIME, TON_TIME, TON_LTIME, TOF_TIME, TOF_LTIME, TONR_TIME, TONR_LTIME	IEC timer whose duration of which is set.

For more information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
IF #started = false THEN
"TON_DB".TON(IN := "Tag_Start",
             PT := "Tag_PresetTime",
             Q => "Tag_Status",
             ET => "Tag_ElapsedTime");
#started := true;
#preset = true
END_IF;

IF "TON_DB".ET < T#10s AND #preset = true THEN
PRESET_TIMER(PT := T#25s,
```

SCL

```
            TIMER := "TON_DB");
#preset := false;
END_IF;
```

When the tag #started has the signal state "0" and the operand "Tag_Start" has a positive signal edge, the instruction "Generate on-delay" is executed. The IEC timer stored in the instance data block "TON_DB" is started with the time duration that is specified by the operand "Tag_PresetTime". The operand "Tag_Status" is set if the time duration PT specified by the operand "Tag_PresetTime" has expired. The parameter Q remains set as long as the operand "Tag_Start" still has the signal state "1". When the signal state of the start input changes from "1" to "0", the operand on the Q parameter is reset.

The instruction "Load time duration" is executed when the expired time of the IEC timer "TON_DB" is less than 10s and the tag #preset has the signal state "1". The instruction writes the time duration that is specified at the parameter "PT" in the instance data block "TON_DB", thereby overwriting the time value of the operand "Tag_PresetTime" within the instance data block. The signal state of the timer status may therefore change at the next query or when "TON_DB".Q or "TON_DB".ET are accessed.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Call function blocks as single or multi-instances (Page 2182)

SIMATIC Timers

S_PULSE: Assign pulse timer parameters and start

Description

The "Assign pulse timer parameters and start" instruction starts the time programmed in the T_NO parameter when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) of the S parameter. The timer runs for the programmed time (TV) as long as the signal state of the S parameter is "1".

When the signal state of the S parameter changes to "0" before the programmed time has expired, the timer is stopped and the "Q" parameter is reset to "0".

Internally, the time is made up of a time value and a time base and is programmed in the TV parameter. When the instruction starts, the programmed time value counts down to zero. The time base specifies the time increment by which the time value changes. The current time value is provided at the parameter BI.

If the timer is running and the signal state at input R changes to "1" then the current time value and the time base are also set to zero. If the timer is not running, the signal state "1" at the R input has no effect.

Parameter Q returns signal state "1" as long as the timer is running and the signal state at parameter S is "1". When the signal state of the S parameter changes to "0" before the programmed time has expired, the Q parameter returns signal state "0". If the timer is reset by parameter R or if the timer has expired then parameter Q also returns signal state "0".

The instruction data is updated with each access. It is therefore possible that the query of the data at the start of the cycle returns different values from those at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Syntax

Use the following syntax for the "Assign pulse timer parameters and start" instruction:

SCL

```
S_PULSE(T_NO := <Operand>,
        S := <Operand>,
        TV := <Operand>,
        R := <Operand>,
        Q => <Operand>,
        BI => <Operand>)
```

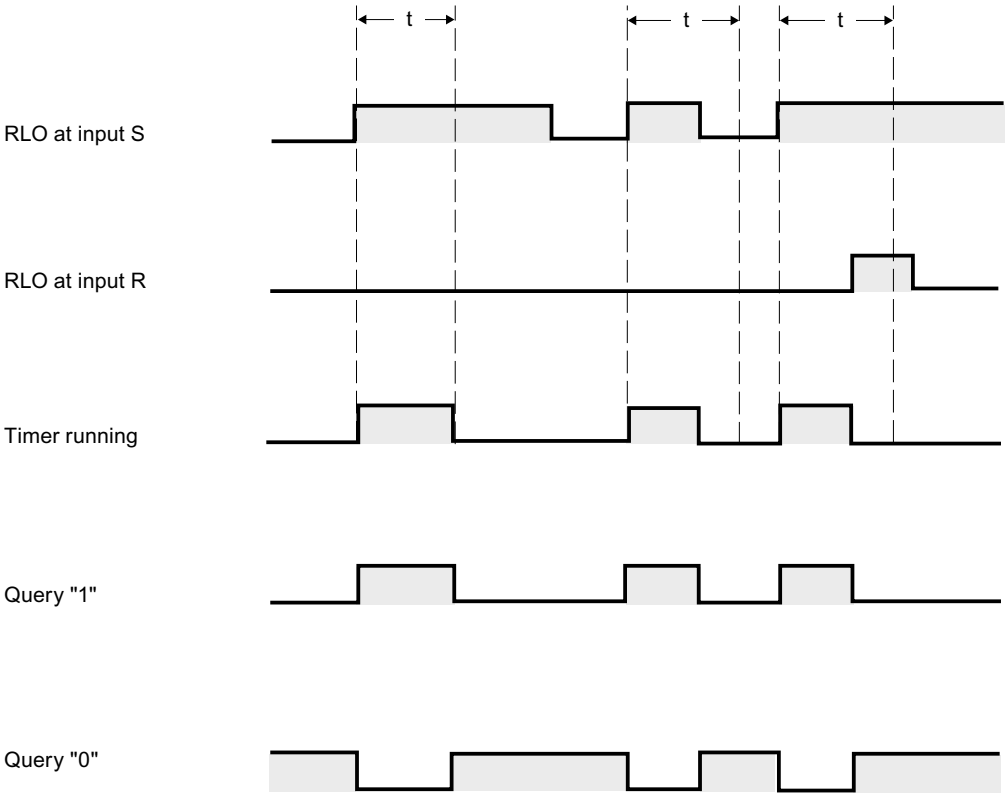
The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
T_NO	Input	TIMER, INT	The timer that is started. The number of timers depends on the CPU.
S	Input	BOOL	Start input
TV	Input	S5TIME, WORD	Preset time value
R	Input	BOOL	Reset input
Q	Output	BOOL	Status of the timer
BI	Output	WORD	Current dual-coded timer value
Function value		S5TIME	Current time value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign pulse timer parameters and start" instruction:



t = programmed time

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := S_PULSE(T_NO := "Timer_1",
                        S := "Tag_1",
                        TV := "Tag_Number",
                        R := "Tag_Reset",
                        Q := "Tag_Status",
                        BI := "Tag_Value");
```

"Timer_1" starts when the signal state of the "Tag_1" operand changes from "0" to "1". The timer counts down with the time value of the operand "Tag_Number" until operand "Tag_1" returns signal state "1".

If the signal state at the S parameter changes to "0" before the programmed time has elapsed, the "Tag_Status" operand is reset to "0". If the timer is reset by the R parameter or if the timer has expired, the "Tag_Status" operand also returns signal state "0".

The current time value is stored both dual-coded at the "Tag_Value" operand and returned as a function value.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

L: Load timer value (Page 3929)

S_PEXT: Assign extended pulse timer parameters and start

Description

The "Assign extended pulse timer parameters and start" instruction starts a programmed timer when a positive signal edge is detected at the S parameter. The timer runs for the programmed time (TV) even if the signal state at the S parameter changes to "0". As long as the timer runs, parameter Q returns the signal state "1".

When the timer has expired, parameter Q is reset to "0". If the signal state at the S parameter changes from "0" to "1" while the timer is running, the timer is restarted with the time programmed in the TV parameter.

Internally, the time is made up of a time value and a time base and is programmed in the TV parameter. When the instruction starts, the programmed time value counts down to zero. The time base specifies the time increment by which the time value changes. The current time value is provided at the parameter BI.

If the timer is running and the signal state at parameter R changes to "1" then the current time value and the time base are also set to zero. If the timer is not running then signal state "1" at parameter R has no effect.

The instruction data is updated with each access. It is therefore possible that the query of the data at the start of the cycle returns different values from those at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Syntax

Use the following syntax for the "Assign extended pulse timer parameters and start" instruction:

SCL

```
S_PEXT(T_NO := <Operand>,
       S := <Operand>,
       TV := <Operand>,
       R := <Operand>,
       Q => <Operand>,
       BI => <Operand>)
```

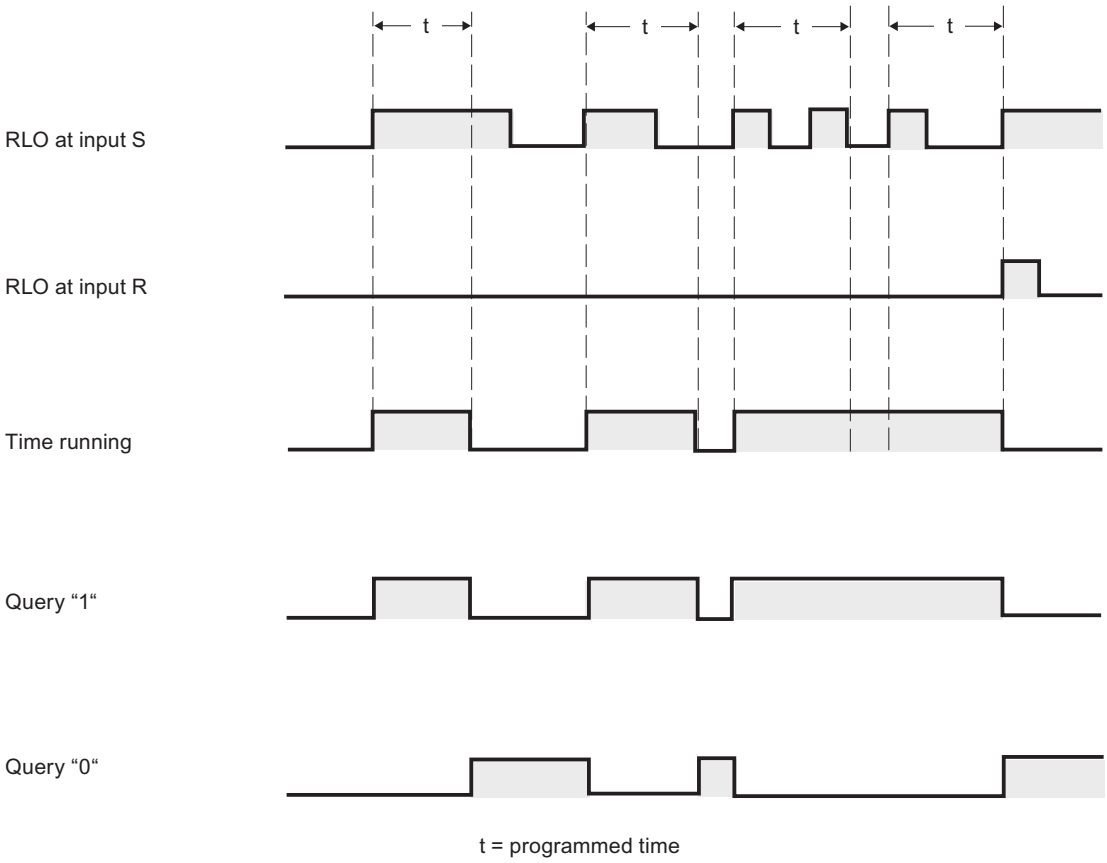
The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
T_NO	Input	TIMER, INT	The timer that is started. The number of timers depends on the CPU.
S	Input	BOOL	Start input
TV	Input	S5TIME, WORD	Preset time value
R	Input	BOOL	Reset input
Q	Output	BOOL	Status of the timer
BI	Output	WORD	Current dual-coded timer value
Function value		S5TIME	Current time value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign extended pulse timer parameters and start" instruction:



Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := S_PEXT(T_NO := "Timer_1",
                       S := "Tag_1",
                       TV := "Tag_Number",
                       R := "Tag_Reset",
                       Q := "Tag_Status",
                       BI := "Tag_Value");
```

"Timer_1" starts when the signal state of the "Tag_1" operand changes from "0" to "1". As long as the timer runs, operand "Tag_Status" returns the signal state "1". When the timer has expired, operand "Tag_Status" is reset to "0". If the signal state at the S input changes from "0" to "1" while the timer is running, the timer is restarted with the time "Tag_Number".

The current time value is stored both dual-coded at the "Tag_Value" operand and returned as a function value.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

L: Load timer value (Page 3929)

S_ODT: Assign on-delay timer parameters and start

Description

The "Assign on-delay timer parameters and start" instruction starts a programmed timer as on-delay when a positive signal edge is detected at the S parameter. The timer runs for the programmed time (TV) as long as the signal state of the S parameter is "1".

If the timer expires correctly and parameter S still has signal state "1" then parameter Q returns signal state "1". If the signal state at the S parameter changes from "1" to "0" while the timer is running, the timer is stopped. In this case, output Q is reset to signal state "0".

Internally, the time is made up of a time value and a time base and is programmed in the TV parameter. When the instruction starts, the programmed time value counts down to zero. The time base specifies the time increment by which the time value changes. The current time value is provided at the parameter BI.

If the time is running and the signal state at input R changes from "0" to "1" then the current time value and the time base are also set to zero. In this case, the signal state at parameter Q is "0". The timer is reset if the signal state at the R parameter is "1", even if the timer is not running and the result of logic operation (RLO) at the S parameter is "1".

The instruction data is updated with each access. It is therefore possible that the query of the data at the start of the cycle returns different values from those at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Syntax

Use the following syntax for the "Assign on-delay timer parameters and start" instruction:

SCL

```
S_ODT(T_NO := <Operand>,
      S := <Operand>,
      TV := <Operand>,
```

SCL

```
      R := <Operand>,
      Q => <Operand>,
      BI => <Operand>)
```

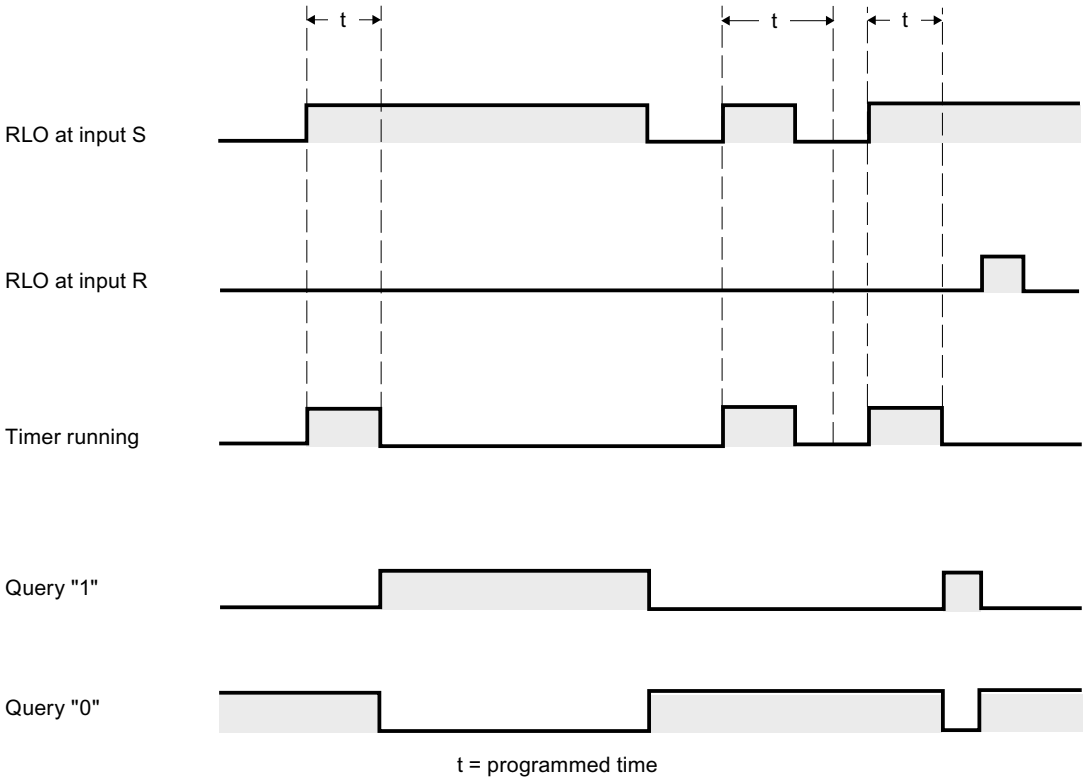
The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
T_NO	Input	TIMER, INT	The timer that is started. The number of timers depends on the CPU.
S	Input	BOOL	Start input
TV	Input	S5TIME, WORD	Preset time value
R	Input	BOOL	Reset input
Q	Output	BOOL	Status of the timer
BI	Output	WORD	Current dual-coded timer value
Function value		S5TIME	Current time value

For additional information on valid data types, refer to "See also".

ulse timing diagram

The following figure shows the pulse timing diagram of the "Assign on-delay timer parameters and start" instruction:



Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := S_ODT(T_NO := "Timer_1",
                      S := "Tag_1",
                      TV := "Tag_Number",
                      R := "Tag_Reset",
                      Q := "Tag_Status",
                      BI := "Tag_Value");
```

"Timer_1" starts when the signal state of the "Tag_1" operand changes from "0" to "1". The timer runs for the duration "Tag_Number" as long as the signal state of operand "Tag_1" is "1".

If the timer expires correctly and operand "Tag_Status" has signal state "1" then operand "Tag_Status" is reset to "1". If the signal state of the "Tag_1" operand changes from "1" to "0" while the timer is running, the timer is stopped. In this case, operand "Tag_Status" returns signal state "0".

The current time value is stored both dual-coded at the "Tag_Value" operand and returned as a function value.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

L: Load timer value (Page 3929)

S_ODTS: Assign retentive on-delay timer parameters and start

Description

The "Assign retentive on-delay timer parameters and start" instruction starts a programmed timer when a positive signal edge is detected at the S parameter. The timer runs for the programmed time (TV) even if the signal state at the S parameter changes to "0".

If the timer expires, the "Q" parameter returns signal state "1" regardless of the signal state of the "S" parameter. If the signal state at the S parameter changes from "0" to "1" while the timer is running, the timer is restarted with the programmed time TV.

Internally, the time is made up of a time value and a time base and is programmed in the TV parameter. When the instruction starts, the programmed time value counts down to zero. The time base specifies the time increment by which the time value changes. The current time value is provided at the parameter BI.

Signal state "1" at parameterR resets the current time value and time base to "0", independent of the signal state at parameter S. In this case, the signal state at parameter Q is "0".

The instruction data is updated with each access. It is therefore possible that the query of the data at the start of the cycle returns different values from those at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Syntax

Use the following syntax for the "Assign retentive on-delay timer parameters and start" instruction:

SCL

```
S_ODTS(T_NO := <Operand>,
       S := <Operand>,
       TV := <Operand>,
       R := <Operand>,
       Q => <Operand>,
       BI =><Operand>)
```

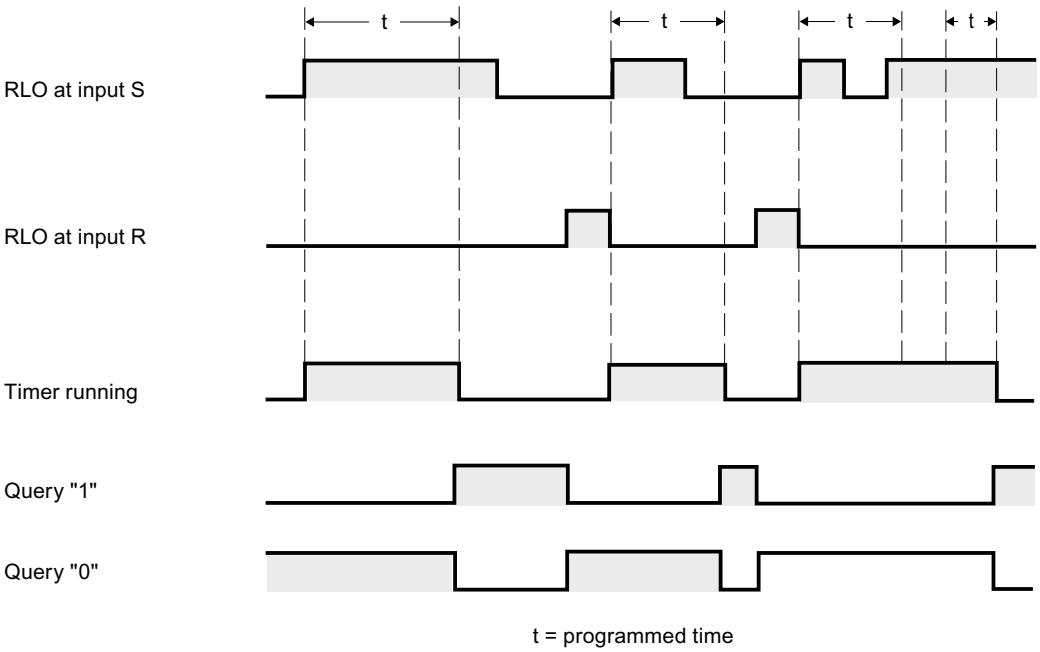
The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
T_NO	Input	TIMER, INT	The timer that is started. The number of timers depends on the CPU.
S	Input	BOOL	Start input
TV	Input	S5TIME, WORD	Preset time value
R	Input	BOOL	Reset input
Q	Output	BOOL	Status of the timer
BI	Output	WORD	Current dual-coded timer value
Function value		S5TIME	Current time value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign retentive on-delay timer parameters and start" instruction:



Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := S_ODTS(T_NO := "Timer_1",
                       S := "Tag_1",
                       TV := "Tag_Number",
                       R := "Tag_Reset",
                       Q := "Tag_Status",
                       BI := "Tag_Value");
```

"Timer_1" starts when the signal state of the "Tag_1" operand changes from "0" to "1". The timer runs for the duration "Tag_Number".

If the timer expires, operand "Tag_Status" returns signal state "1" independent of the signal state of operand "Tag_1". If the signal state of the "Tag_1" operand changes from "0" to "1" while the timer is running, the timer is restarted with the time "Tag_Number".

The current time value is stored both dual-coded at the "Tag_Value" operand and returned as a function value.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

L: Load timer value (Page 3929)

S_OFFDT: Assign off-delay timer parameters and start

Description

The "Assign off-delay timer parameters and start" instruction starts a programmed timer when a negative signal edge is detected at the S parameter. The timer runs for the programmed time (TV). As long as the timer is running or parameter S returns signal state "1", then parameter Q has signal state "1".

If the timer expires and the signal state is "0" then parameter Q is reset to signal state "0". If the signal state at parameter S changes from "0" to "1" while the timer is running, the timer is stopped. The timer is only restarted after a falling signal edge is detected at parameter S.

Internally, the time is made up of a time value and a time base and is programmed in the TV parameter. When the instruction starts, the programmed time value counts down to zero. The time base specifies the time increment by which the time value changes. The current time value is provided at the parameter BI.

Signal state "1" at parameter R resets the current time value and time base to "0". In this case, the signal state at parameter Q is "0".

The instruction data is updated with each access. It is therefore possible that the query of the data at the start of the cycle returns different values from those at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Syntax

Use the following syntax for the "Assign off-delay timer parameters and start" instruction:

SCL

```
S_OFFDT(T_NO := <Operand>,
        S := <Operand>,
        TV := <Operand>,
        R := <Operand>,
        Q => <Operand>,
        BI =><Operand>)
```

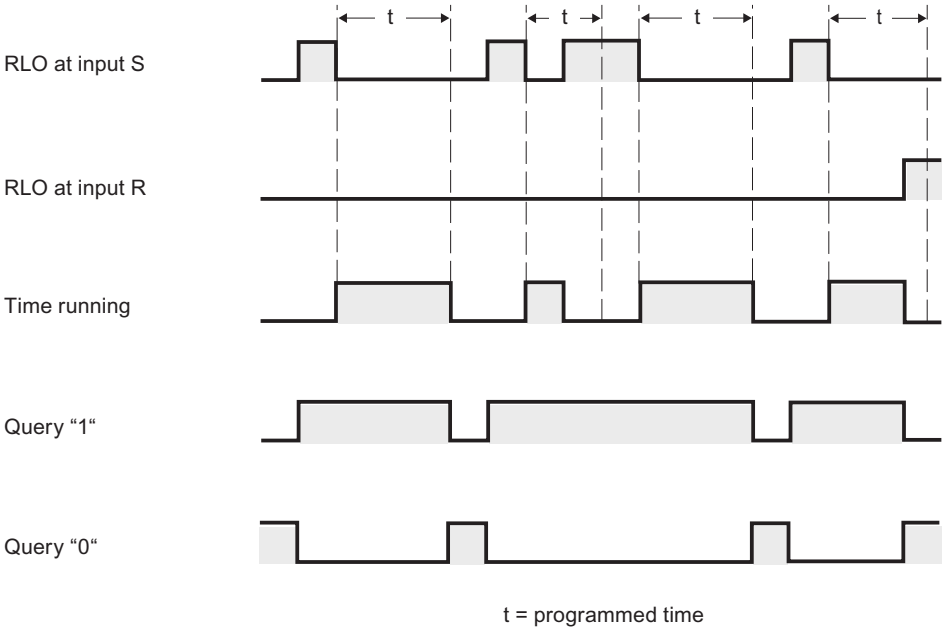
The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
T_NO	Input	TIMER, INT	The timer that is started. The number of timers depends on the CPU.
S	Input	BOOL	Start input
TV	Input	S5TIME, WORD	Preset time value
R	Input	BOOL	Reset input
Q	Output	BOOL	Status of the timer
BI	Output	WORD	Current dual-coded timer value
Function value		S5TIME	Current time value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign off-delay timer parameters and start" instruction:



Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := S_OFFDT(T_NO := "Timer_1",
                        S := "Tag_1",
                        TV := "Tag_Number",
                        R := "Tag_Reset",
                        Q := "Tag_Status",
```

SCL

```
                        BI := "Tag_Value");
```

"Timer_1" starts when the signal state of the "Tag_1" operand changes from "0" to "1". The timer runs for the duration "Tag_Number". As long as the timer is running or operand "Tag_1" returns signal state "1", then operand "Tag_Status" has signal state "1".

If the timer expires and the signal state of operand "Tag_1" is "0" then operand "Tag_Status" is reset to signal state "0". If the signal state of the "Tag_1" operand changes from "0" to "1" while the timer is running, the timer is reset. The timer is only restarted after a falling edge is detected at parameter S.

The current time value is stored both dual-coded at the "Tag_Value" operand and returned as a function value.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

L: Load timer value (Page 3929)

Counter operations

IEC Counters

CTU: Count up

Description

You can use the "Count up" instruction to increment the value at the CV parameter. When the signal state at the CU parameter changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value of the CV parameter is incremented by one. When the instruction is executed for the first time the current count of the CV parameter is set to zero. The counter value is incremented each time a positive signal edge is detected, until it reaches the high limit of the data type specified for the CV parameter. When the high limit is reached, the signal state of the CU parameter no longer has an effect on the instruction.

You can query the count status of the Q parameter. The signal state of the Q parameter is determined by the PV parameter. When the current counter value is greater than or equal to the value of the PV parameter, the Q parameter is set to signal state "1". In all other cases, the signal state of the Q parameter is "0". You can also specify a constant for the PV parameter.

The value of the CV parameter is reset to zero when the signal state at the R parameter changes to "1". As long as the R parameter has the signal state "1", the signal state of the CU parameter has no effect on the instruction.

Note

Only use a counter at a single location in the program to avoid counting errors.

Each call of the "Count up" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

For S7-1200 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER	• CTU_SINT / CTU_USINT • CTU_INT / CTU_UINT • CTU_DINT / CTU_UDINT

For S7-1500 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER	• CTU_SINT / CTU_USINT • CTU_INT / CTU_UINT • CTU_DINT / CTU_UDINT • CTU_LINT / CTU_ULINT

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER")
- Declaration as a local tag of the type CTU_<Data type> in the "Static" section of a block (for example, #MyCTU_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

When you set up the IEC counter in a separate data block (single instance), the instance data block is created by default with "optimized block access" and the individual tags are defined as retentive. For additional information on setting retentivity in an instance data block, refer to "See also".

When you set up the IEC counter as local tag (multiple instance) in a function block with "optimized block access", it is defined as retentive in the block interface.

Syntax

The following syntax is used for the "Count up" instruction:

Table 9-28 Data block of system data type IEC_counter (global DB)

SCL

```
<IEC_Counter_DB>.CTU(CU := <Operand>,
                     R := <Operand>,
                     PV := <Operand>,
                     Q => <Operand>,
                     CV => <Operand>)
```

Table 9-29 Local tag

SCL

```
#myLocal_counter(CU := <Operand>,
                 R := <Operand>,
                 PV := <Operand>,
                 Q => <Operand>,
                 CV => <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
CU	Input	BOOL	Count input
R	Input	BOOL	Reset input
PV	Input	Integers	Value at which the Q output is set
Q	Output	BOOL	Counter status
CV	Output	Integers, CHAR, DATE	Current counter value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"IEC_COUNTER_DB".CTU(CU := "Tag_Start",
                     R := "Tag_Reset",
                     PV := "Tag_PresetValue",
                     Q => "Tag_Status",
                     CV => "Tag_CounterValue");
```

When the signal state at the "Tag_Start" operand changes from "0" to "1", the "Count up" instruction is executed and the current counter value of the "Tag_CounterValue" operand is incremented by one. With each additional positive signal edge, the counter value is incremented until the high limit of the specified data type (INT = 32767) is reached.

The "Tag_Status" output has signal state "1" as long as the current counter value is greater than or equal to the value of the "Tag_PresetValue" operand. In all other cases, the

"Tag_Status" output has signal state "0". The current counter value is stored in the "Tag_CounterValue" operand.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Call function blocks as single or multi-instances (Page 2182)

Setting retentivity in an instance data block (Page 2839)

Setting the retentivity of local tags (Page 2561)

CTD: Count down

Description

The "Count down" instruction is used to decrement the value at the parameter CV. When the signal state at the CD parameter changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value of the CV parameter is decremented by one. When the instruction is executed the first time, the counter value of the CV parameter is set to the value of the PV parameter. Each time a positive signal edge is detected, the counter value is decremented until it reaches the low limit of the specified data type. When the low limit is reached, the signal state of the CD parameter no longer has an effect on the instruction.

You can query the count status of the Q parameter. If the current counter value is less than or equal to zero, the Q parameter is set to signal state "1". In all other cases, the signal state of the Q parameter is "0". You can also specify a constant for the PV parameter.

The value of the CV parameter is set to the value of the PV parameter when the signal state of the LD parameter changes to "1". As long as the LD parameter has the signal state "1", the signal state of the CD parameter has no effect on the instruction.

Note

Only use a counter at a single location in the program to avoid counting errors.

Each call of the "Count down" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

For S7-1200 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER	• CTD_SINT / CTD_USINT • CTD_INT / CTD_UINT • CTD_DINT / CTD_UDINT

For S7-1500 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER	• CTD_SINT / CTD_USINT • CTD_INT / CTD_UINT • CTD_DINT / CTD_UDINT • CTD_LINT / CTD_ULINT

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER")
- Declaration as a local tag of the type CTD_<Data type> in the "Static" section of a block (for example, #MyCTD_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

When you set up the IEC counter in a separate data block (single instance), the instance data block is created by default with "optimized block access" and the individual tags are defined as retentive. For additional information on setting retentivity in an instance data block, refer to "See also".

When you set up the IEC counter as local tag (multiple instance) in a function block with "optimized block access", it is defined as retentive in the block interface.

Syntax

The following syntax is used for the "Count down" instruction:

Table 9-30 Data block of system data type IEC_counter (global DB)

SCL

```
<IEC_Counter_DB>.CTD(CD := <Operand>,
                     LD := <Operand>,
                     PV := <Operand>,
                     Q => <Operand>,
                     CV => <Operand>)
```

Table 9-31 Local tag

SCL

```
#myLocal_counter(CD := <Operand>,
                 LD := <Operand>,
                 PV := <Operand>,
                 Q => <Operand>,
                 CV => <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
CD	Input	BOOL	Count input
LD	Input	BOOL	Load input
PV	Input	Integers	Value at which the Q output is set
Q	Output	BOOL	Counter status
CV	Output	Integers, CHAR, DATE	Current counter value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"IEC_SCOUNTER_DB".CTD(CD := "Tag_Start",
                      LD := "Tag_Load",
                      PV := "Tag_PresetValue",
                      Q => "Tag_Status",
                      CV => "Tag_CounterValue");
```

When the signal state at the "Tag_Start" changes from "0" to "1", the "Count down" instruction executes and the value of the "Tag_CounterValue" operand is decremented by one. With each additional positive signal edge, the counter value is decremented until it reaches the low limit of the specified data type (-128).

The operand "Tag_Status" has the signal state "1" as long as the current counter value is less than or equal to zero. In all other cases, the "Tag_Status" output has signal state "0". The current counter value is stored in the "Tag_CounterValue" operand.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Call function blocks as single or multi-instances (Page 2182)

Setting retentivity in an instance data block (Page 2839)

Setting the retentivity of local tags (Page 2561)

CTUD: Count up and down

Description

Use the "Count up and down" instruction to increment or decrement the counter value at the CV parameter. When the signal state at the CU parameter changes from "0" to "1" (positive signal edge), the current counter value of the CV parameter is incremented by one. When the signal state at the CD parameter changes from "0" to "1" (positive signal edge), the counter

value of the CV parameter is decremented by one. If there is a positive signal edge at the CU and CD inputs in a program cycle, the current counter value of the CV parameter remains unchanged.

The counter value can be incremented until it reaches the high limit of the data type specified for the CV parameter. When the high limit is reached, the counter value is no longer incremented on a positive signal edge. The counter value is no longer decremented once the low limit of the specified data type has been reached.

When the signal state at the LD parameter changes to "1", the counter value of the CV parameter is set to the value of the PV parameter. As long as the LD parameter has the signal state "1", the signal state of the CU and CD parameters has no effect on the instruction.

The counter value is set to zero when the signal state of the R parameter changes to "1". As long as the R parameter has signal state "1", a change in the signal state of the CU, CD and LD parameters has no effect on the "Count up and down" instruction.

You can query the status of the up counter at the QU parameter. When the current counter value is greater than or equal to the value of the PV parameter, the QU parameter is set to signal state "1". In all other cases, the signal state of the QU parameter is "0". You can also specify a constant for the PV parameter.

You can query the status of the down counter at the QD parameter. If the current counter value is less than or equal to zero, the QD parameter is set to signal state "1". In all other cases, the signal state of the QD parameter is "0".

Note

Only use a counter at a single location in the program to avoid counting errors.

Each call of the "Count up and down" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

For S7-1200 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER	• CTUD_SINT / CTUD_USINT • CTUD_INT / CTUD_UINT • CTUD_DINT / CTUD_UDINT

For S7-1500 CPU

Data block of system data type IEC_<Counter> (Shared DB)	Local tag
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER	• CTUD_SINT / CTUD_USINT • CTUD_INT / CTUD_UINT • CTUD_DINT / CTUD_UDINT • CTUD_LINT / CTUD_ULINT

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER")
- Declaration as a local tag of the type CTUD_<Data type> in the "Static" section of a block (for example, #MyCTUD_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

When you set up the IEC counter in a separate data block (single instance), the instance data block is created by default with "optimized block access" and the individual tags are defined as retentive. For additional information on setting retentivity in an instance data block, refer to "See also".

When you set up the IEC counter as local tag (multiple instance) in a function block with "optimized block access", it is defined as retentive in the block interface.

Syntax

The following syntax is used for the "Count up and down" instruction:

Table 9-32 Data block of system data type IEC_counter (global DB)

SCL

```
<IEC_Counter_DB>.CTUD(CU := <Operand>,
                      CD := <Operand>,
                      R := <Operand>,
                      LD := <Operand>,
                      PV := <Operand>,
                      QU=> <Operand>,
                      QD := <Operand>,
                      CV => <Operand>)
```

Table 9-33 Local tag

SCL

```
myLocal_counter(CU := <Operand>,
                CD := <Operand>,
                R := <Operand>,
                LD := <Operand>,
                PV := <Operand>,
                QU=> <Operand>,
                QD := <Operand>,
                CV => <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
CU	Input	BOOL	Count up input
CD	Input	BOOL	Count down input
R	Input	BOOL	Reset input
LD	Input	BOOL	Load input
PV	Input	Integers	Value at which the QU output is set.
QU	Output	BOOL	Status of the up counter
QD	Output	BOOL	Status of the down counter
CV	Output	Integers, CHAR, DATE	Current counter value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"IEC_COUNTER_DB".CTUD(CU := "Tag_Start1",
                      CD := "Tag_Start2",
                      LD := "Tag_Load",
                      R := "Tag_Reset",
                      PV := "Tag_PresetValue",
                      QU => "Tag_CU_Status",
                      QD => "Tag_CD_Status",
                      CV => "Tag_CounterValue");
```

If the "Tag_Start1" operand has a positive signal edge in the signal state, the current counter value is incremented by one and stored in the "Tag_CounterValue" operand. If the "Tag_Start2" operand has a positive signal edge in the signal state, the counter value is decremented by one and is also stored in the "Tag_CounterValue" operand. The counter value is incremented on the positive signal edge of the CU parameter until it reaches the high limit of the specified data type (INT). If the CD parameter has a positive signal edge, the counter value is decremented until it reaches the low limit of the specified data type (INT).

The operand "Tag_CU_Status" has the signal state "1" as long as the current counter value is greater than or equal to the value of the operand "Tag_PresetValue". In all other cases, the "Tag_CU_Status" output has signal state "0".

The operand "Tag_CD_Status" has the signal state "1" as long as the current counter value is less than or equal to zero. In all other cases, the "Tag_CD_Status" output has signal state "0".

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Call function blocks as single or multi-instances (Page 2182)

Setting retentivity in an instance data block (Page 2839)

Setting the retentivity of local tags (Page 2561)

SIMATIC Counters

S_CU: Assign parameters and count up

Description

You can use the "Assign parameters and count up" instruction to increment the value of a counter. When the signal state of the parameter CU changes from "0" to "1" (positive signal edge), the current counter value is incremented by one. The current counter value is provided at the parameter CV. The counter value is incremented until the limit of "999" is reached. When the limit value is reached, the counter value is no longer incremented on a positive signal edge.

When the signal state of the parameter S changes from "0" to "1", the counter value is set to the value of the parameter PV. If the counter is set and if the result of logic operation (RLO) at the CU input is "1", the counter counts once in the next cycle, even when no signal edge change is detected.

The counter value is set to zero when the signal state of the parameter R changes to "1". As long as the parameter R has the signal state "1", a change in the signal state of the parameters CU and S has no effect on the counter value.

The signal state at parameter Q is "1" if the counter value is greater than zero. When the counter value equals zero, parameter Q returns signal state "0".

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Syntax

The following syntax is used for the "Assign parameters and count up" instruction:

SCL

```
S_CU(C_NO := <Operand>,
     CU:= <Operand>,
     S := <Operand>,
     PV := <Operand>,
     R := <Operand>,
     Q => <Operand>,
     CV=> <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
C_NO	Input	COUNTER, INT	Counters The number of counters depends on the CPU.
CU	Input	BOOL	Count up input
S	Input	BOOL	Input for presetting the counter
PV	Input	WORD	Preset counter value (C#0 to C#999) in BCD format
R	Input	BOOL	Reset input
Q	Output	BOOL	Status of the counter
CV	Output	WORD, S5TIME, DATE	Current counter value
Function value		WORD, S5TIME, DATE	Current counter value in BCD format

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := S_CU(C_NO := "Counter_1",
                     CU := "Tag_Start",
                     S := "Tag_1",
                     PV := "Tag_PresetValue",
                     R := "Tag_Reset",
                     Q => "Tag_Status",
                     CV => "Tag_Value");
```

If the signal state of the "Tag_Start" parameter changes from "0" to "1" (positive signal edge) and the current counter value is less than "999", the counter value is incremented by one. If the signal state of the input "Tag_1" changes from "0" to "1", the counter value in BCD format is set to the value of the operand "Tag_PresetValue". The counter value is reset to "0" when the operand "Tag_Reset" has signal state "1".

The current counter value is stored in hexadecimal form in the operand "Tag_Value".

The output "Tag_Status" has the signal state "1" as long as the current counter value is not equal to "0". The current counter value is returned in the "Tag_Value" operand and as a function value.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

S_CD: Assign parameters and count down

Description

You can use the "Assign parameters and count down" instruction to decrement the value of a counter. When the signal state of the parameter CD changes from "0" to "1" (positive signal edge), the current counter value is decreased by one. The current counter value is provided at the parameter CV. The counter value is decreased until the low limit of "0" is reached. When the low limit value is reached, the counter value is no longer decreased on a positive signal edge.

When the signal state of the parameter S changes from "0" to "1", the counter value is set to the value of the parameter PV. If the counter is set and if the result of logic operation (RLO) at the parameter CD is "1", the counter counts once in the next cycle, even when no signal edge change is detected.

The counter value is set to zero when the signal state of the parameter R changes to "1". As long as the parameter R has the signal state "1", a change in the signal state of the parameters CD and S has no effect on the counter value.

The signal state at parameter Q is "1" if the counter value is greater than zero. When the counter value equals zero, parameter Q returns signal state "0".

Note

Only use a counter at a single location in the program to avoid risk of counting errors.

Syntax

The following syntax is used for the "Assign parameters and count down" instruction:

SCL

```
S_CD(C_NO:= <Operand>,
     CD:= <Operand>,
     S:= <Operand>,
     PV:= <Operand>,
     R:= <Operand>,
     Q=> <Operand>,
     CV=> <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
C_NO	Input	COUNTER, INT	Counters The number of counters depends on the CPU.
CD	Input	BOOL	Count down input
S	Input	BOOL	Input for presetting the counter

Parameter	Declaration	Data type	Description
PV	Input	WORD	Preset counter value (C#0 to C#999) in BCD format
R	Input	BOOL	Reset input
Q	Output	BOOL	Status of the counter
CV	Output	WORD, S5TIME, WORD	Current counter value
Function value		WORD, S5TIME, DATE	Current counter value in BCD format

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := S_CD(C_NO := "Counter_1",
                     CD := "Tag_Start",
                     S := "Tag_1",
                     PV := "Tag_PresetValue",
                     R := "Tag_Reset",
                     Q => "Tag_Status",
                     CV => "Tag_Value");
```

When the signal state of the operand "Tag_Start" changes from "0" to "1" (positive signal edge) and the current counter value is greater than "0", the counter value is decreased by one. If the signal state of the operand "Tag_1" changes from "0" to "1", the counter value is set to the value of the operand "Tag_PresetValue". The counter value is reset to "0" when the operand "Tag_Reset" has signal state "1".

The current counter value is stored in the operand "Tag_Value".

The operand "Tag_Status" returns signal state "1" as long as the current counter value is not equal to "0". The current counter value is returned in the "Tag_Value" operand and as a function value.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

S_CUD: Assign parameters and count up / down

Description

You can use the "Assign parameters and count up / down" instruction to increment and decrement the value of a counter. When the signal state of the parameter CU changes from "0" to "1" (positive signal edge), the current counter value is incremented by one. When the signal state of the parameter CD changes from "0" to "1" (positive signal edge), the counter value is decreased by one. The current counter value is provided at the parameter CV. If there is a positive signal edge at the parameters CU and CD in one program cycle, the counter value remains unchanged.

The counter value is incremented until the high limit value of "999" is reached. When the high limit value is reached, the counter value is no longer incremented on a positive signal edge. When the low limit value "0" is reached, the counter value is not decremented any further.

When the signal state of the parameter S changes from "0" to "1", the counter value is set to the value of the parameter PV. If the counter is set and if the result of logic operation (RLO) of the parameters CU and CD is "1", the counter will count once in the next cycle, even if no signal edge change was detected.

The counter value is set to zero when the signal state of the parameter R changes to "1". As long as the parameter R has the signal state "1", processing of the signal state of the parameters CU, CD and S has no effect on the counter value.

The signal state at parameter Q is "1" if the counter value is greater than zero. When the counter value equals zero, parameter Q returns signal state "0".

Note

Only use a counter at a single location in the program to avoid risk of counting errors.

Syntax

The following syntax is used for the "Assign parameters and count up / down" instruction:

SCL

```
S_CUD(C_NO:= <Operand>,
      CU:= <Operand>,
      CD:= <Operand>,
      S:= <Operand>,
      PV:= <Operand>,
      R:= <Operand>,
      Q=> <Operand>,
      CV=> <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
C_NO	Input	COUNTER, INT	Counters The number of counters depends on the CPU.
CU	Input	BOOL	Count up input
CD	Input	BOOL	Count down input
S	Input	BOOL	Input for presetting the counter
PV	Input	WORD	Preset counter value (C#0 to C#999) in BCD format
R	Input	BOOL	Reset input
Q	Output	BOOL	Status of the counter
CV	Output	WORD, S5TIME, DATE	Current counter value (hexadecimal)
Function value		WORD, S5TIME, DATE	Current counter value in BCD format

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := S_CD(C_NO := "Counter_1",
                     CU := "Tag_CU",
                     CD := "Tag_CD",
                     S := "Tag_1",
                     PV := "Tag_PresetValue",
                     R := "Tag_Reset",
                     Q => "Tag_Status",
                     CV => "Tag_Value");
```

When a positive signal edge is detected in the signal state of the operand "Tag_CU" and the current counter value is less than "999", the counter value is incremented by one. When a positive signal edge is detected in the signal state of the operand "Tag_CD" and the current counter value is greater than "0", the counter value is decremented by one.

If the signal state of the operand "Tag_1" changes from "0" to "1", the counter value is set to the value of the operand "Tag_PresetValue". The counter value is reset to "0" when the operand "Tag_Reset" has signal state "1".

The current counter value is stored in the operand "Tag_Value".

The operand "Tag_Status" returns signal state "1" as long as the current counter value is not equal to "0". The current counter value is returned in the "Tag_Value" operand and as a function value.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Comparator operations

TypeOf: Query data type of a VARIANT tag

Description

You can use the "Check data type of a VARIANT tag" instruction to query the data type of a tag to which a VARIANT tag is pointing. You compare the data type of the tag to the data type of a tag that you have declared in the block interface to determine whether they are "Equal" or "Not equal".

The operand must have the VARIANT data type. The comparison operand can be an elementary data type or a PLC data type.

You can only use the "Check data type of a VARIANT tag" instruction within an IF instruction.

Syntax

The following syntax is used for the "Check data type of a VARIANT tag" instruction:

SCL

```
TypeOf(<Operand>)
```

Parameters

The following table shows the parameters of the "Check data type of a VARIANT tag" instruction:

Parameter	Declaration	Data type	Description
<Operand>	Input	VARIANT	First operand

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
IF TypeOf(#TagVARIANT) = TypeOf("TagBOOL") THEN
...;
```

SCL

```
END_IF;
```

If the operand to which the VARIANT "#TagVARIANT" is pointing has the BOOL data type, a specific command is to be executed.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Switching display formats in the program status (Page 2980)

TypeOfElements: Check element data type of a VARIANT tag

Description

You can use the "Check element data type of a VARIANT tag" instruction to query the data type of a tag to which a VARIANT is pointing. You compare the data type of the tag to the data type of a tag that you have declared in the block interface to determine whether they are "Equal" or "Not equal".

The operand must have the VARIANT data type. The comparison operand can be an elementary data type or a PLC data type.

If the data type of the VARIANT tag is an ARRAY, the data type of the ARRAY elements is compared.

You can only use the "Check element data type of a VARIANT tag" instruction within an IF instruction.

Syntax

The following syntax is used for the "Check element data type of a VARIANT tag" instruction:

SCL

```
TypeOfElements(<Operand>)
```

Parameter

The following table shows the parameters of the "Check element data type of a VARIANT tag" instruction:

Parameter	Declaration	Data type	Description
<Operand>	Input	VARIANT	First operand

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL
```
IF TypeOfElements(#Tag_VARIANT) = TypeOf("GlobalDB".Product[1]) THEN
"Tag_Result" := "GlobalDB".Product[1] * 3;
END_IF;
```

The following table shows how the instruction works using specific operand values:

Operand	Value
#Tag_VARIANT	1.5
"GlobalDB".Product[1]	3.5

If the tag to which the VARIANT points and the ""GlobalDB".Product[1]" operand have the REAL data type, the "GlobalDB".Product[1]" operand is then multiplied by 3 and the result is written to the "Tag_Result" operand.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Switching display formats in the program status (Page 2980)

IS_ARRAY: Check for ARRAY

Description

You can use the "Check for ARRAY" instruction to query whether the VARIANT points to a tag of the ARRAY data type.

The <Operand> must have the VARIANT data type.

You can only use the "Check for ARRAY" instruction within an IF instruction.

Syntax

The following syntax is used for the "Check for ARRAY" instruction:

SCL
```
IS_ARRAY(<Operand>)
```

Parameter

The following table shows the parameters of the "Check for ARRAY" instruction:

Parameter	Declaration	Data type	Description
<Operand>	Input	VARIANT	Operand that is queried for ARRAY
Function value		UDINT	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
IF IS_ARRAY(#Tag_VARIANTToArray) THEN
"Tag_Result" := CountOfElements(#Tag_VARIANT);
END_IF;
```

If the tag to which the VARIANT points is an ARRAY, the number of ARRAY elements is output.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Switching display formats in the program status (Page 2980)

Math functions

ABS: Form absolute value

Description

Use the "Form absolute value" instruction to calculate the absolute value of an input value and to save the result in the specified operands.

Syntax

Use the following syntax for the "Form absolute value" instruction:

SCL

```
ABS(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type		Description
		S7-1200	S7-1500	
<Expression>	Input	SINT, INT, DINT, floating-point numbers	SINT, INT, DINT, LINT, floating-point numbers	Input value
Function value		SINT, INT, DINT, floating-point numbers	SINT, INT, DINT, LINT, floating-point numbers	Absolute value of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result1" := ABS("Tag_Value");
"Tag_Result2" := ABS("Tag_Value1"*"Tag_Value2");
```

The absolute value of the input value is returned in the format of the input value as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	-2
Tag_Result1	2
Tag_Value1	4
Tag_Value2	-1
Tag_Result2	4

See also

Overview of the valid data types (Page 2236)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

MIN: Get minimum

Description

The "Get minimum" instruction compares the values of the available inputs and returns the lowest value as the result.

A minimum of two and a maximum of 32 inputs can be specified at the instruction.

The result is invalid if any of the following conditions are met:

- The implicit conversion of the data types fails during execution of the instruction.
- A floating-point number has an invalid value.

Syntax

The following syntax is used for the "Get minimum" instruction:

```
SCL
MIN(IN1:= <Operand>,
    IN2 := <Operand>)
```

Parameter

The following table shows the parameters of the "Get minimum" instruction:

Parameter	Declaration	Data type		Description
		S7-1200	S7-1500	
IN1	Input	Integers, floating-point numbers, TIME, TOD, DATE, DTL	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DT, DTL	First input value
IN2	Input	Integers, floating-point numbers, TIME, TOD, DATE, DTL	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DT, DTL	Second input value
INn	Input	Integers, floating-point numbers, TIME, TOD, DATE, DTL	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DT, DTL	Additionally inserted inputs whose values are to be compared
Function value		Integers, floating-point numbers, TIME, TOD, DATE, DTL	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DT, DTL	Result of the instruction
The data types TOD, LTOD, DATE, and LDT can only be used if the IEC test is not enabled.				

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := MIN(IN1 := "Tag_Value1",
                    IN2 := "Tag_Value2",
                    IN3 := "Tag_Value3");
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	Tag_Value1	12222
IN2	Tag_Value2	14444
IN3	Tag_Value3	13333
Function value	Tag_Result	12222

The instruction compares the values of the available inputs and copies the lowest value ("Tag_Value1") to operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

MAX: Get maximum

Description

The "Get maximum" instruction compares the values of the available inputs and returns the greatest value as the result.

A minimum of two and a maximum of 32 inputs can be specified at the instruction.

The result is invalid if any of the following conditions are met:

- The implicit conversion of the data types fails during execution of the instruction.
- A floating-point number has an invalid value.

Syntax

The following syntax is used for the "Get maximum" instruction:

SCL

```
MAX(IN1:= <Operand>,
    IN2 := <Operand>)
```

Parameters

The following table shows the parameters of the "Get maximum" instruction:

Parameter	Declaration	Data type		Description
		S7-1200	S7-1500	
IN1	Input	Integers, floating-point numbers, TIME, TOD, DATE, DTL	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DT, DTL	First input value
IN2	Input	Integers, floating-point numbers, TIME, TOD, DATE, DTL	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DT, DTL	Second input value
INn	Input	Integers, floating-point numbers, TIME, TOD, DATE, DTL	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DT, DTL	Additionally inserted inputs whose values are to be compared
Function value		Integers, floating-point numbers, TIME, TOD, DATE, DTL	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DT, DTL	Result of the instruction
The data types TOD, LTOD, DATE, and LDT can only be used if the IEC test is not enabled.				

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := MAX(IN1 := "Tag_Value1",
                    IN2 := "Tag_Value2",
                    IN3 := "Tag_Value3");
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	Tag_Value1	12 222
IN2	Tag_Value2	14 444
IN3	Tag_Value3	13 333
Function value	Tag_Result	14 444

The instruction compares the values of the specified operands and copies the greatest value ("Tag_Value2") to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

LIMIT: Set limit value

Description

The "Set limit value" instruction limits the value of the parameter IN to the values of the parameters MN and MX. The value of the parameter MN may not be greater than the value of the parameter MX.

If the value of the IN parameter fulfills the MN condition <=IN <= MX, it is returned as the result of the instruction. If the condition is not fulfilled and the IN input value is less than the MN low limit, the value of the MN parameter is returned as the result. If the high limit MX is exceeded, the value of the MX parameter is returned as the result.

If the value of the MN input is greater than the value of the MX input, the result is undefined.

The instruction is only executed if the operands of all parameters are of the same data type.

Syntax

The following syntax is used for the "Set limit value" instruction:

SCL

```
LIMIT(MN := <Operand>,
      IN := <Operand>,
      MX := <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type		Description
		S7-1200	S7-1500	
MN	Input	Integers, floating-point numbers, TIME, TOD, DATE, DTL	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DT, DTL	Low limit
IN	Input	Integers, floating-point numbers, TIME, TOD, DATE, DTL	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DT, DTL	Input value
MX	Input	Integers, floating-point numbers, TIME, TOD, DATE, DTL	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DT, DTL	High limit
Function value		Integers, floating-point numbers, TIME, TOD, DATE, DTL	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT, DT, DTL	Result of the instruction
The data types TOD, LTOD, DATE, and LDT can only be used if the IEC test is not enabled.				

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := LIMIT(MN := "Tag_Minimum",
                      IN := "Tag_Value",
                      MX := "Tag_Maximum");
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
MN	Tag_Minimum	12 000
IN	Tag_Value	8 000
MX	Tag_Maximum	16 000
Function value	Tag_Result	12 000

The value of operand "Tag_Value" is compared with the values of operands "Tag_Minimum" and "Tag_Maximum". Because the value of the operand "Tag_Value" is less than the low limit value, the value of operand "Tag_Minimum" will be copied to operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)
Expressions (Page 2681)
Operators and operator precedence (Page 2687)
Entering SCL instructions (Page 2697)
Editing SCL instructions (Page 2716)

SQR: Form square

Description

Use the "Form square" instruction to square the input value and save the result in the specified operand.

Syntax

Use the following syntax for the instruction "Form square":

```
SCL
SQR(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
<Expression>	Input	Floating-point numbers	Input value
Function value		Floating-point numbers	Square of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result1" := SQR("Tag_Value");
"Tag_Result2" := SQR((SQR("Tag_Value1"))*"Tag_Value2);
```

The square of the input value is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	2.5
Tag_Result1	6.25

Operand	Value
Tag_Value1	6.0
Tag_Value2	2.0
Tag_Result2	5184.0

See also

Overview of the valid data types (Page 2236)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

SQRT: Form square root

Description

Use the "Form square root" instruction to calculate the square root of the the input value and save the result in the specified operand. The instruction has a positive result if the input value is greater than zero. If input values are less than zero, the instruction returns an invalid floating-point number. If the input value is "-0", the result is also "-0".

Syntax

Use the following syntax for the "Form square root" instruction:

SCL

```
SQRT(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
<Expression>	Input	Floating-point numbers	Input value
Function value		Floating-point numbers	Square root of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result1" := SQRT("Tag_Value");
```

```
SCL
"Tag_Result2" := SQRT((SQR("Tag_Value1"))+"Tag_Value2");
```

The square root of the input value is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	4.0
Tag_Result1	2.0
Tag_Value1	3.0
Tag_Value2	16.0
Tag_Result2	5.0

See also

Overview of the valid data types (Page 2236)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

LN: Form natural logarithm

Description

You can use the "Form natural logarithm" instruction to calculate the natural logarithm to base e (e=2.718282) of the input value. The instruction has a positive result if the input value is greater than zero. If input values are less than zero, the instruction returns an invalid floating-point number.

Syntax

Use the following syntax for the "Form natural logarithm" instruction:

```
SCL
LN(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
<Expression>	Input	Floating-point numbers	Input value
Function value		Floating-point numbers	Natural logarithm of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result1" := LN("Tag_Value");
"Tag_Result2" := LN("Tag_Value1"+"Tag_Value2");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	2.5
Tag_Result1	0.916
Tag_Value1	1.5
Tag_Value2	3.2
Tag_Result2	1.548

See also

Overview of the valid data types (Page 2236)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

EXP: Form exponential value

Description

The "Form exponential value" instruction calculates the exponent from the base e (e = 2.718282) and the input value and saves the result in the specified operand.

Syntax

The following syntax is used for the "Form exponential value" instruction:

```
SCL
EXP(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
<Expression>	Input	Floating-point numbers	Input value
Function value		Floating-point numbers	Exponential value of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result1" := EXP("Tag_Value");
"Tag_Result2" := EXP("Tag_Value1"/"Tag_Value2");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	20.5
Tag_Result1	799 902 200
Tag_Value1	15.5
Tag_Value2	30.2
Tag_Result2	1.671

See also

Overview of the valid data types (Page 2236)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

SIN: Form sine value

Description

Use the "Form sine value" instruction to calculate the sine of the input value. The input value must be given in the radian measure.

Syntax

Use the following syntax for the "Form sine value" instruction:

```
SCL
SIN(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
<Expression>	Input	Floating-point numbers	Input value (size of an angle in the radian measure)
Function value		Floating-point numbers	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := SIN("Tag_Value");
```

The result of the instruction is returned in the "Tag_Result" operand as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	+1.570796 (π/2)
Tag_Result	1.0

See also

Overview of the valid data types (Page 2236)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

COS: Form cosine value

Description

Use the "Form cosine value" instruction to calculate the cosine of the input value. The input value must be given in the radian measure.

Syntax

Use the following syntax for the "Form cosine value" instruction:

```
SCL
COS(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
<Expression>	Input	Floating-point numbers	Input value (size of an angle in the radian measure)
Function value		Floating-point numbers	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := COS("Tag_Value");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	+1.570796 (π/2)
Tag_Result	0

See also

Overview of the valid data types (Page 2236)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

TAN: Form tangent value

Description

Use the "Form tangent value" instruction to calculate the sine of the input value. The input value must be given in the radian measure.

Syntax

Use the following syntax for the "Form tangent value" instruction:

```
SCL
TAN(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
<Expression>	Input	Floating-point numbers	Input value (size of an angle in the radian measure)
Function value		Floating-point numbers	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := TAN("Tag_Value");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	+3.141593 (π)
Tag_Result	0

See also

Overview of the valid data types (Page 2236)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

ASIN: Form arcsine value

Description

You can use the "Form arcsine value" instruction to calculate the size of the angle from a sine value, which corresponds to this value. Only valid floating-point numbers within the range -1

to +1 can be specified as input values. The calculated angle size is given in the radian measure and can range in value from -π/2 to +π/2.

Syntax

Use the following syntax for the "Form arcsine value" instruction:

```
SCL
ASIN(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
<Expression>	Input	Floating-point numbers	Sine value
Function value		Floating-point numbers	Size of angle in the radian measure

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := ASIN("Tag_Value");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	1.0
Tag_Result	+1.570796 (π/2)

See also

Overview of the valid data types (Page 2236)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

ACOS: Form arccosine value

Description

You can use the "Form arccosine value" instruction to calculate the size of the angle from a cosine value, which corresponds to this value. Only valid floating-point numbers within the range -1 to +1 can be specified as input values. The calculated angle size is given in the radian measure and can range in value from 0 to +π.

Syntax

Use the following syntax for the "Form arccosine value" instruction:

SCL

```
ACOS(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
<Expression>	Input	Floating-point numbers	Cosine value
Function value		Floating-point numbers	Size of angle in the radian measure

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := ACOS("Tag_Value");
```

The result of the instruction is returned in the "Tag_Result" operand as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	0
Tag_Result	+1.570796 (π/2)

See also

Overview of the valid data types (Page 2236)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)
Editing SCL instructions (Page 2716)

ATAN: Form arctangent value

Description

You can use the "Form arctangent value" instruction to calculate the size of the angle from a tangent value, which corresponds to this value. Only valid floating-point numbers may be specified as input values. The calculated angle size is given in the radian measure and can range in value from -π/2 to +π/2.

Syntax

Use the following syntax for the "Form arctangent value" instruction:

```
SCL
ATAN(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
<Expression>	Input	Floating-point numbers	Tangent value
Function value		Floating-point numbers	Size of angle in the radian measure

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := ATAN("Tag_Value");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	1.0
Tag_Result	+0.785398 (π/4)

See also

Overview of the valid data types (Page 2236)
Expressions (Page 2681)
Operators and operator precedence (Page 2687)
Entering SCL instructions (Page 2697)
Editing SCL instructions (Page 2716)

FRAC: Return fraction

Description

The result of the instruction "Return fraction" returns the decimal places of a value. Input value 123.4567, for example, returns the value 0.4567.

Syntax

The following syntax is used for the instruction "Return fraction":

```
SCL
FRAC(<Expression>)
FRAC_<Data type>(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
<Expression>	Input	Floating-point numbers	Input value
_<Data type>		Floating-point numbers Default: REAL	Data type of the function value: • You do not need to specify the data type if using the default. • Any other valid data type you may use must be declared explicitly.
Function value		Floating-point numbers	Decimal places of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result1" := FRAC("Tag_Value");
"Tag_Result2" := FRAC_LREAL("Tag_Value");
```

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_Value	2.555	-1.4421
Tag_Result1	0.555	-0.4421

See also

Overview of the valid data types (Page 2236)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Move operations

Deserialize: Deserialization

Description

You can use the "Deserialize" instruction to convert the sequential representation of a PLC data type (UDT) back to a PLC data type and to fill its entire contents.

The memory area in which the sequential representation of a PLC data type is located must have the ARRAY of BYTE data type and be declared with standard access. Make sure that there is enough memory space prior to the conversion.

The instruction enables you to convert multiple sequential representations of converted PLC data types back to their original state step-by-step.

Note

If you only want to convert back a single sequential representation of a PLC data type (UDT), you can also use the instruction "TRCV: Receive data via communication connection" directly.

Syntax

The following syntax is used for the "Deserialize" instruction:

SCL

```
Deserialize(SRC_ARRAY := <Operand>,
            DEST_VARIABLE:= <Operand>,
            POS := <Operand>)
```

Parameter

The following table shows the parameters of the "Deserialize" instruction:

Parameter	Declaration	Data type	Description
SRC_ARRAY	Input	VARIANT	Global data block in which the generated data stream is stored
DEST_VARIABLE	InOut	VARIANT	Tag in which the returned converted PLC data type (UDT) is stored
POS	InOut	DINT	Number of bytes that the converted PLC data types use. The POS parameter is calculated zero-based.
Function value		INT	Error information

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B0	The memory areas for the SRC_ARRAY and DEST_VARIABLE parameters overlap.
8136	The data block at the DEST_VARIABLE parameter is not a block with standard access.
8150	The VARIANT data type at the SRC_ARRAY parameter contains no value.
8151	Code generation error at SRC_ARRAY parameter
8153	There is not enough free memory available at the SRC_ARRAY parameter.
8250	The VARIANT data type at the DEST_VARIABLE parameter contains no value.
8251	Code generation error at DEST_VARIABLE parameter
8254	Invalid data type at the DEST_VARIABLE parameter
8382	The value at the POS parameter is outside the limits of the array.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

SCL

```
#Tag_RetVal := Deserialize(SRC_ARRAY := "Buffer".Field,
                           DEST_VARIABLE := "Target".Client,
                           POS := #BufferPos);

#Tag_RetVal := Deserialize(SRC_ARRAY := "Buffer".Field,
                           DEST_VARIABLE := #Label,
                           POS := #BufferPos);

IF #Label = 'arti' THEN
#Tag_RetVal := Deserialize(SRC_ARRAY := "Buffer".Field,
                           DEST_VARIABLE := "Target".Article[#DeliverPos],
                           POS := #BufferPos);
ELSIF #Label = 'Bill' THEN
#Tag_RetVal := Deserialize(SRC_ARRAY := "Buffer".Field,
                           DEST_VARIABLE := "Target".Bill[#DeliverPos],
                           POS := #BufferPos);
;
ELSE
;
END_IF;
```

The "Move value" instruction moves the value "0" to the "#BufferPos" operand. The "Deserialize" instruction deserializes the sequential representation of the customer data from the "Buffer" data block and writes it to the "Target" data block. The number of bytes used by the converted customer data is stored in the "#BufferPos" operand.

The "Deserialize" instruction deserializes the sequential representation of the separator sheet, which was saved with the sequential representation after the customer data, from the "Buffer" data block and writes the characters to the "#Label" operand. The characters are compared using comparison instructions for "arti" and "Bill". If the comparison for "arti" = TRUE, these are article data that have been deserialized and written to the "Target" data block. If the comparison for "Bill" = TRUE, these are billing data that have been deserialized and written to the "Target" data block.

The following table shows the declaration of the operands:

Operand	Data type	Declaration
DeliverPos	INT	In the "Input" section of the block interface
BufferPos	DINT	In the "Temp" section of the block interface
Error	INT	In the "Temp" section of the block interface
Label	STRING[4]	In the "Temp" section of the block interface

The following table lists the declarations of the PLC data types:

Name of the PLC data types	Name	Data type
Article	Number	DINT
	Declaration	STRING
	Colli	INT
Client	Title	INT
	First name	STRING[10]
	Surname	STRING[10]

The following table shows the declaration of the data blocks:

Name of the data blocks	Name	Data type
Target	Client	"Client" (PLC data type)
	Article	Array[0..] of "Article" (PLC data type)
	Bill	Array[0..10] of INT
Buffer	Field	Array[0..294] of BYTE

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Switching display formats in the program status (Page 2980)

PLC data types (Page 2280)

Serialize: Serialization

Description

You can use the "Serialize" instruction to convert several PLC data types (UDT) to a sequential representation without losing parts of their structure.

You can use the instruction to temporarily save multiple structured data items from your program in a buffer, which should preferably be in a global data block, and send them to another CPU. The memory area in which the converted PLC data types are stored must have the ARRAY of BYTE data type and be declared with standard access. Make sure that there is enough memory space prior to the conversion.

The operand at the POS parameter contains information about the number of bytes used by the converted PLC data types.

Note

If you want to send a single PLC data type (UDT), you can directly call the instruction "TSEND: Send data via communication connection".

Syntax

The following syntax is used for the "Serialize" instruction:

SCL

```
Serialize(SRC_VARIABLE := <Operand>,
          DEST_ARRAY := <Operand>,
          POS := <Operand>)
```

Parameter

The following table shows the parameters of the "Serialize" instruction:

Parameter	Declaration	Data type	Description
SRC_VARIABLE	Input	VARIANT	PLC data type (UDT) that is converted to sequential representation.
DEST_ARRAY	InOut	VARIANT	Data block in which the generated data stream is stored.
POS	InOut	DINT	Number of bytes that the converted PLC data types use. The POS parameter is calculated zero-based.
Function value		INT	Error information

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B0	The memory areas for the SRC_VARIABLE and DEST_ARRAY parameters overlap.
8150	The VARIANT data type at the SRC_VARIABLE parameter contains no value.
8152	Code generation error at SRC_VARIABLE parameter
8236	The data block at the DEST_ARRAY parameter is not a block with standard access.
8250	The VARIANT data type at the DEST_ARRAY parameter contains no value.
8252	Code generation error at DEST_ARRAY parameter
8253	There is not enough free memory available at the DEST_ARRAY parameter.
8254	Invalid data type at the DEST_ARRAY parameter
8382	The value at the POS parameter is outside the limits of the array.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

SCL

```
#Tag_RetVal := Serialize(SRC_VARIABLE := "Source".Client,
                         DEST_ARRAY := "Buffer".Field,
                         POS := #BufferPos);

#Label := STRING_TO_WSTRING('arti');

#Tag_RetVal := Serialize(SRC_VARIABLE := #Label,
                         DEST_ARRAY := "Buffer".Field,
                         POS := #BufferPos);

#Tag_RetVal := Serialize(SRC_VARIABLE := "Source".Article[#DeliverPos],
                         DEST_ARRAY := "Buffer".Field,
                         POS := #BufferPos);
```

The "Move value" instruction moves the value "0" to the "#BufferPos" operand. The "Serialize" instruction serializes the customer data from the "Source" data block and writes them in sequential representation to the "Buffer" data block. The number of bytes used by the sequential representation is stored in the "#BufferPos" operand.

A kind of separator sheet is inserted now to make it easier to deserialize the sequential representation later. The "Move character string" instruction moves the "arti" characters to the "#Label" operand. The "Serialize" instruction writes these characters after the customer data to the "Buffer" data block. The number of bytes that the characters need is added to the number already stored in the "#BufferPos" operand.

The "Serialize" instruction serializes the data of a specific article, which is calculated in runtime, from the "Source" data block and writes them in sequential representation to the "Buffer" data block after the "arti" characters.

The following table shows the declaration of the operands:

Operand	Data type	Declaration
DeliverPos	INT	In the "Input" section of the block interface
BufferPos	DINT	In the "Temp" section of the block interface
Error	INT	In the "Temp" section of the block interface
Label	STRING[4]	In the "Temp" section of the block interface

The following table lists the declarations of the PLC data types:

Name of the PLC data types	Name	Data type
Article	Number	DINT
	Declaration	STRING
	Colli	INT
Client	Title	INT

Name of the PLC data types	Name	Data type
	First name	STRING[10]
	Surname	STRING[10]

The following table shows the declaration of the data blocks:

Name of the data blocks	Name	Data type
Source	Client	"Client" (PLC data type)
	Article	Array[0..10] of "Article" (PLC data type)
Buffer	Field	Array[0..294] of BYTE

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Switching display formats in the program status (Page 2980)

PLC data types (Page 2280)

MOVE_BLK: Move block

Description

The instruction "Move block" copies the contents of a memory area (source area) to another memory area (destination area). The number of elements to be copied to the destination area is specified with the COUNT parameter. The width of the elements to be moved is defined by the width of the first element in the source area.

The instruction can only be executed if the source area and the destination area are of the same data type.

The value of the OUT output is invalid if the following condition is met:

- More data is moved than is made available at the IN parameter or OUT parameter.

Syntax

The following syntax is used for the instruction "Move block":

SCL

```
MOVE_BLK(IN := <Operand>,
         COUNT := <Operand>,
         OUT => <Operand>)
```

Parameter

The following table shows the parameters of the "Move block" instruction:

Parameter	Declaration	Data type		
		S7-1200	S7-1500	Description
IN [1]	Input	Binary numbers, integers, floating-point numbers, TIME, DATE, CHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, TOD, LTOD	The first element of the source area that is being copied
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	Number of elements to be copied from the source area to the destination area
OUT [1]	Output	Binary numbers, integers, floating-point numbers, TIME, DATE, CHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, TOD, LTOD	The first element of the destination area to which the contents of the source area are being copied
[1] The specified data types can only be used as elements of an ARRAY structure.				

For more information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
MOVE_BLK(IN := #a_array[2],
         COUNT := "Tag_Count",
         OUT => #b_array[1]);
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	a_array[2]	The data type of the a_array operand is Array [0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the b_array operand is Array [0..6] of INT. It consists of seven elements of data type INT.

Starting from the third element, the instruction selects three INT elements from the #a_array tag and copies their contents to the #b_array output tag, beginning with the second element.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

MOVE_BLK_VARIANT: Move block

Description

The instruction "Move block" copies the contents of a memory area (source area) to another memory area (destination area). You can copy elements of an array to another array of the same data type. The size (number of elements) of the source and destination array may be different. You can copy several elements within an array or individual elements.

If you are using the instruction, the array must not yet be known at the time the block is created, as the source and the destination are transferred using VARIANT.

The counting at the parameters SRC_INDEX and DEST_INDEX always starts with the low limit "0", regardless of the later declaration of the array.

The instruction is not executed if more data is copied than is made available.

If the VARIANT pointer (source or destination) is of the data type BOOL, it must be addressed absolutely and the length specified must be divisible by 8; otherwise the instruction is not executed.

Syntax

The following syntax is used for the "Move block" instruction:

SCL

```
MOVE_BLK_VARIANT(SRC := <Operand>,
                 COUNT := <Operand>,
                 SRC_INDEX := <Operand>,
                 DEST_INDEX := <Operand>,
                 DEST => <Operand>)
```

Parameter

The following table shows the parameters of the "Move block" instruction:

Parameter	Declaration	Data type	Description
SRC	Input	VARIANT (array or single element)	Source block from which to copy
COUNT	Input	UDINT	Number of elements which are copied Set the value at the parameter COUNT to "1", if no Array is specified at the parameter SRC or the parameter DEST.
SRC_INDEX	Input	DINT	• The SRC_INDEX parameter is calculated zero-based. If an Array is specified at parameter SRC, the integer at the SRC_INDEX parameter specifies the first element within the source area from which copying is to take place. This is independent of the declared array limits. • If no Array is set at the SRC parameter or only an individual element of an array is specified, assign the SRC_INDEX parameter the value "0".
DEST_INDEX	Input	DINT	• The DEST_INDEX parameter is calculated zero-based. If an Array is specified at the DEST parameter, the integer at the DEST_INDEX parameter specifies the first element within the destination area to which copying is to take place. This is independent of the declared array limits. • If no Array is specified at the DEST parameter, assign the DEST_INDEX parameter the value "0".
DEST	Output	VARIANT	Destination area into which the contents of the source block are copied.
Function value (RET_VAL)		INT	Error information

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	Data types do not correspond
8151	Access to the SRC parameter is not possible.
8152	The operand at the SRC parameter is not typed.
8153	Code generation error at SRC parameter
8154	The operand at the SRC parameter has the data type BOOL.
8281	The COUNT parameter has an invalid value
8382	The value at the SRC_INDEX parameter is outside the limits of the VARIANT.
8383	The value of the SRC_INDEX parameter is outside the high limit of the array.
8482	The value at the DEST_INDEX parameter is outside the limits of the VARIANT.
8483	The value of the DEST_INDEX parameter is outside the high limit of the array.
8534	The DEST parameter is write protected
8551	Access to the DEST parameter is not possible.
8552	The operand at the DEST parameter is not typed.
8553	Code generation error at DEST parameter
8554	The operand at the DEST parameter has the data type BOOL.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := MOVE_BLK_VARIANT(SRC := #SrcField,
                                 COUNT := "Tag_Count",
                                 SRC_INDEX := "Tag_Src_Index",
                                 DEST_INDEX := "Tag_Dest_Index",
                                 DEST => #DestField);
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
SRC	#SrcField	The local operand #SrcField uses a UDT that was still unknown at the time when the block was programmed. (Array [0 to 10] of "MOVE_UDT"
COUNT	Tag_Count	2
SRC_INDEX	Tag_Src_Index	3
DEST_INDEX	Tag_Dest_Index	3
DEST	#DestField	The local operand #DestField uses a UDT that was still unknown at the time when the block was programmed. (Array [10 to 20] of "MOVE_UDT"

2 elements are copied from the source area into the destination area beginning with the fourth element of the array of UDT. The copies are inserted in the array of UDT beginning with the fourth element.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Switching display formats in the program status (Page 2980)

VariantGet: Read out VARIANT tag value (Page 4226)

UMOVE_BLK: Move block uninterruptible

Description

The instruction "Move block uninterruptible" copies the contents of a memory area (source area) to another memory area (destination area) without interruption. The number of elements to be copied to the destination area is specified with the COUNT parameter. The width of the elements to be moved is defined by the width of the first element in the source area.

The instruction can only be executed if the source area and the destination area have the same data type.

Note

The copy operation cannot be interrupted by other operating system activities. This is why the alarm reaction times of the CPU increase during the execution of the instruction "Move block uninterruptible".

The value of the OUT output is invalid if the following condition is met:

- More data is moved than is made available at the IN parameter or OUT parameter.

You can use the "Move block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Syntax

The following syntax is used for the instruction "Move block uninterruptible":

SCL

```
UMOVE_BLK(IN := <Operand>,
          COUNT := <Operand>,
          OUT => <Operand>)
```

Parameter

The following table shows the parameters of the "Move block uninterruptible" instruction:

Parameter	Declaration	Data type		Description
		S7-1200	S7-1500	
IN [1]	Input	Binary numbers, integers, floating-point numbers, TIME, DATE, CHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, TOD, LTOD	The first element of the source area that is being copied
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	Number of elements to be copied from the source area to the destination area
OUT [1]	Output	Binary numbers, integers, floating-point numbers, TIME, DATE, CHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, TOD, LTOD	The first element of the destination area to which the contents of the source area are being copied
[1] The specified data types can only be used as elements of an ARRAY structure.				

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
UMOVE_BLK(IN := #a_array[2],
          COUNT := "Tag_Count",
          OUT => #b_array[1]);
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	a_array[2]	The data type of the a_array operand is Array [0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the b_array operand is Array [0..6] of INT. It consists of seven elements of data type INT.

Starting from the third element, the instruction selects three INT elements from the #a_array tag and copies their contents to the #b_array output tag, beginning with the second element. The copy operation cannot be interrupted by other operating system activities.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

FILL_BLK: Fill block

Description

The "Fill block" instruction is used to copy the content of a memory area (source area) to a selected memory area (target area). The number of repeated copy operations is specified with the COUNT parameter. When the instruction is executed, the source area is selected and moved to the destination area as often as specified by the value of the COUNT parameter.

The instruction can only be executed if the source area and the destination area are of the same data type.

The value of the OUT output is invalid if the following condition is met:

- More data is moved than is made available at input IN or output OUT.

Syntax

The following syntax is used for the instruction "Fill block":

SCL

```
FILL_BLK(IN := <Operand>,
         COUNT := <Operand>,
         OUT => <Operand>)
```

Parameter

The following table shows the parameters of the "Fill block" instruction:

Parameter	Declaration	Data type		Description
		S7-1200	S7-1500	
IN [1]	Input	Binary numbers, integers, floating-point numbers, TIME, TOD, DATE, CHAR	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, TOD, LTOD	Element used to fill the destination area
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	Number of repeated copy operations
OUT [2]	Output	Binary numbers, integers, floating-point numbers, TIME, TOD, DATE, CHAR	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, TOD, LTOD	Address in destination area from which filling starts
[1] The specified data types can also be used as elements of an ARRAY structure. [2] The specified data types can only be used as elements of an ARRAY structure.				

For more information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
FILL_BLK(IN := #a_array[2],
         COUNT := "Tag_Count",
         OUT => #b_array[1]);
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	a_array[2]	The data type of the a_array operand is Array [0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the b_array operand is Array [0..6] of INT. It consists of seven elements of data type INT.

The instruction copies the third element (#a_array[2]) of the #a_array tag three times to the #b_array output tag.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

UFILL_BLK: Fill block uninterruptible

Description

The instruction "Fill block uninterruptible" is used to copy the content of a memory area (source area) to a selected memory area (target area). The number of repeated copy operations is specified with the COUNT parameter. When the instruction is executed, the value at the IN input is selected and copied to the destination area as often as specified by the value in the COUNT parameter.

The instruction can only be executed if the source area and the destination area have the same data type.

Note

The copy operation cannot be interrupted by other operating system activities. This is why the alarm reaction times of the CPU increase during the execution of the "Fill block uninterruptible" instruction.

The value of the OUT output is invalid if the following condition is met:

- More data is moved than is made available at input IN or output OUT.

You can use the "Fill block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Syntax

The following syntax is used for the "Fill block uninterruptible" instruction:

SCL

```
UFILL_BLK(IN := <Operand>,
          COUNT := <Operand>,
          OUT => <Operand>)
```

Parameter

The following table shows the parameters of the "Fill block uninterruptible" instruction:

Parameter	Declaration	Data type		Description
		S7-1200	S7-1500	
IN [1]	Input	Binary numbers, integers, floating-point numbers, TIME, DATE, CHAR, TOD	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, TOD, LTOD	Element used to fill the destination area
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	Number of repeated copy operations
OUT [2]	Output	Binary numbers, integers, floating-point numbers, TIME, TOD, DATE, CHAR	Binary numbers, integers, floating-point numbers, timers, DATE, CHAR, TOD, LTOD	Address in destination area from which filling starts
[1] The specified data types can also be used as elements of an ARRAY structure. [2] The specified data types can only be used as elements of an ARRAY structure.				

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
UFILL_BLK(IN := #a_array[2],
          COUNT := "Tag_Count",
          OUT => #b_array[1]);
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	a_array[2]	The data type of the a_array operand is Array [0..5] of INT. It consists of six elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the b_array operand is Array [0..6] of INT. It consists of seven elements of data type INT.

The instruction copies the third element (#a_array[2]) of the #a_array tag three times to the #b_array output tag. The copy operation cannot be interrupted by other operating system activities.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

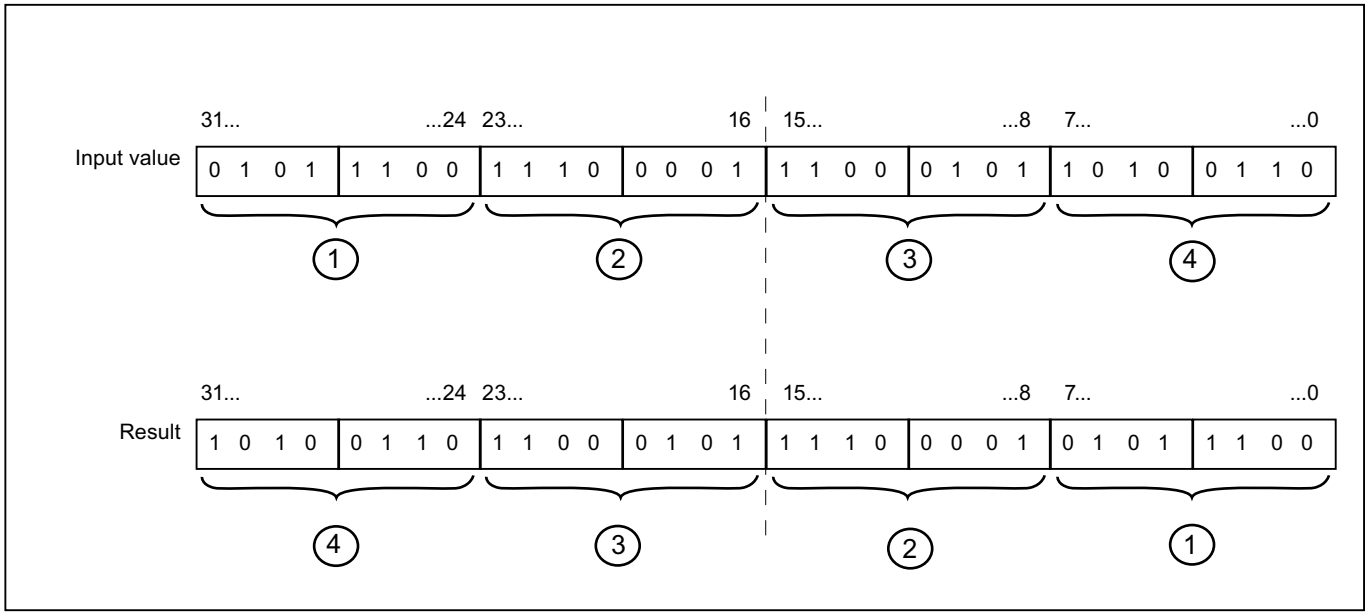
Editing SCL instructions (Page 2716)

SWAP: Swap

Description

You can use the "Swap" instruction to change the arrangement of the bytes of an input value and save the result in the specified operand.

The following figure shows how the bytes of an operand of the DWORD data type are swapped using the "Swap" instruction:



Syntax

The following syntax is used for the "Swap" instruction:

```
SCL
SWAP(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type		Description
		S7-1200	S7-1500	
<Expression>	Input	WORD, DWORD	WORD, DWORD, LWORD	Input value
Function value		WORD, DWORD	WORD, DWORD, LWORD	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := SWAP("Tag_Value");
```

The result of the instruction is returned as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	0000 1111 0101 0101
Tag_Result	0101 0101 0000 1111

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

ARRAY DB instructions

ReadFromArrayDB: Read from array data block

Description

You can use the "Read from array data block" instruction to read data from an array DB and write them to a destination area.

An array DB is a data block which consists of exactly one array of [data type]. The elements of the array can be UDT data type or any other elementary data type. The counting of the array always starts with the low limit "0", regardless of the later declaration of the array.

Syntax

The following syntax is used for the "Read from array data block" instruction:

SCL

```
ReadFromArrayDB(DB := <Operand>,
                INDEX := <Operand>,
                VALUE => <Operand>)
```

Parameter

The following table shows the parameters of the "Read from array data block" instruction:

Parameter	Declaration	Data type	Description
DB	Input	DB_ANY	Data block that is read
INDEX	Input	DINT	Element that is read
VALUE	Output	VARIANT	Value that is read and output
Function value (RET_VAL)		INT	Result of the instruction

For additional information on valid data types, refer to "See also".

Parameters RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	Data types do not correspond
80B5	Copying process was interrupted
8132	The data block does not exist or is too short
8133	The data block is located in the load memory
8134	The data block is write protected
8135	The array DB contains invalid values
8154	The data type is not correctly declared
8282	The value at the INDEX parameter is outside the limit values of the array.
8350	The VARIANT data type contains no value
8352	Code generation error
8353	Code generation error
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

SCL

```
"TagResult" := ReadFromArrayDB(DB := "ArrayDB",
                               INDEX := "ArrayDB"."THIS"[2],
                               VALUE => "TargetField[10]".Data2[1]);
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
DB	ArrayDB	The data type of the "ArrayDB" operand is Array [0 to 10] of INT.
INDEX	ArrayDB."THIS"[2]	Third element of the "ArrayDB"
VALUE	TargetField[10].Data2[1]	The data type of the "TargetField" operand is Array [10 to 20] of "MOVE_UDT".

The third element is read from "ArrayDB" and written to the "TargetField[10].Data2[1]" operand.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

WriteToArrayDB: Write to array data block

Description

You can use the "Write to array data block" instruction to write data to an array DB.

An array DB is a data block which consists of exactly one array of [data type]. The elements of the array can be UDT data type or any other elementary data type. The counting of the array always starts with the low limit "0", regardless of the later declaration of the array.

Syntax

The following syntax is used for the "Write to the ARRAY data block" instruction:

SCL

```
WriteToArrayDB(DB := <Operand>,
               INDEX := <Operand>,
               VALUE := <Operand>)
```

Parameter

The following table shows the parameters of the "Write to array data block" instruction:

Parameter	Declaration	Data type	Description
DB	Input	DB_ANY	Data block to which data is written
INDEX	Input	DINT	Element in the DB to which data is written
VALUE	Input	VARIANT	Value to be written
Function value (RET_VAL)		INT	Result of the instruction

For additional information on valid data types, refer to "See also".

Parameters RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	Data types do not correspond
80B5	Copying process was interrupted
8132	The data block does not exist or is too short
8133	The data block is located in the load memory
8134	The data block is write protected
8135	The DB is not an array data block.
8282	The value at the INDEX parameter is outside the limit values of the array.
8350	The VARIANT data type contains no value
8352	Code generation error
8353	Code generation error
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

SCL

```
"TagResult" := WriteToArrayDB(DB := "ArrayDB",
                              INDEX := "ArrayDB"."THIS"[2],
                              VALUE := "SourceField[1]".Data1[6]);
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
DB	ArrayDB	The data type of the "ArrayDB" operand is Array [0 to 10] of INT.
INDEX	ArrayDB."THIS"[2]	Third element of the "ArrayDB"
VALUE	SourceField[1].Data1[6]	The data type of the "SourceField" operand is Array [0 to 10] of "MOVE_UDT".

From the "SourceField" operand, the "Data1[6]" element from the second element is written to the "ArrayDB". The third element is written to the "ArrayDB".

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

ReadFromArrayDBL: Read from array data block in load memory

Description

You can use the "Read from array data block in load memory" instruction to read data from an array DB in load memory.

An array DB is a data block which consists of exactly one array of [data type]. The elements of the array can be UDT data type or any other elementary data type. The counting of the array always starts with the low limit "0", regardless of the later declaration of the array.

If the array DB is marked with the "Only store in load memory" block attribute, it is only stored in load memory.

The instruction is executed when a positive signal edge is detected at the REQ parameter. The BUSY parameter then has the signal state "1". The instruction is terminated if a negative signal edge is detected at the BUSY parameter. The DONE parameter has the signal state "1" for one program cycle and the read value is output at the VALUE parameter within this cycle. With all other program cycles, the value at the VALUE parameter is not changed.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Note

The array DB must be located in a data block with the block property "Optimized".

Syntax

The following syntax is used for the "Read from array data block in load memory" instruction:

SCL

```
"Instance DB"(REQ := <Operand>,
            DB := <Operand>,
            INDEX := <Operand>,
            BUSY => <Operand>,
            DONE => <Operand>,
            ERROR => <Operand>,
            VALUE := <Operand>)
```

Parameter

The following table shows the parameters of the "Read from array data block in load memory" instruction:

Parameter	Declaration	Data type	Description
REQ	Input	BOOL	REQ = "1": Starting with the reading of the array DB
DB	Input	DB_ANY	Data block that is read
INDEX	Input	DINT	Element that is read
BUSY	Output	BOOL	BUSY = "1": The array DB is still being read
DONE	Output	BOOL	DONE = "1": The instruction was executed successfully
VALUE	InOut	VARIANT	Value that is read and output
ERROR	Output	INT	Error information: If an error occurs during the execution of the instruction, an error code is output on the ERROR parameter.

For additional information on valid data types, refer to "See also".

ERROR parameter

The following table shows the meaning of the values of the ERROR parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	Data types do not correspond

Error code* (W#16#...)	Explanation
80B5	Copying process was interrupted
80C3	Too many concurrent calls
8132	The data block does not exist or is too short
8133	The data block is not located in the load memory
8134	The data block is write protected
8135	The DB is not an array data block.
8282	The value of the INDEX parameter is outside the limits of the array.
8350	The VARIANT data type contains no value
8352	Code generation error
8353	Code generation error
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

You can find descriptions of the error codes triggered by the instructions "READ_DBL: Read from data block in load memory" and "WRIT_DBL: Write to data block in load memory" under "See also".

Example

The following example shows how the instruction works:

SCL

```
"ReadFromArrayDBL_DB"(REQ := "TagReg",
                      DB := "ArrayDB",
                      INDEX := "ArrayDB"."THIS"[2],
                      BUSY => "TagBusy",
                      DONE => "TagDone",
                      ERROR => "TagError",
                      VALUE := "SourceTargetField[1]".Data1[6]);
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
REQ	TagReq	BOOL
DB	ArrayDB	The data type of the "ArrayDB" operand is Array [0 to 10] of INT.
INDEX	ArrayDB."THIS"[2]	Third element of the "ArrayDB"
VALUE	SourceTargetField[1].Data1[6]	The data type of the "SourceTargetField" operand is Array [0 to 10] of "MOVE_UDT".
BUSY	TagBusy	BOOL
DONE	TagDone	BOOL

If a positive edge is detected at the "TagReq" operand, the "Read from array data block in load memory" instruction is executed. The third element is read from "ArrayDB" and written to the "SourceTargetField[1].Data1[6]" operand. As soon as a negative signal edge is detected at the "TagBusy" operand, the instruction is terminated and the value at the VALUE parameter is no longer changed. After the instruction has been processed, the "TagDone" operand the signal state "1".

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

READ_DBL: Read from data block in the load memory (Page 4801)

WRIT_DBL: Write to data block in the load memory (Page 4803)

WriteToArrayDBL: Write to array data block in load memory

Description

You can use the "Write to the array data block in load memory" instruction to write data to an array DB in load memory.

An array DB is a data block which consists of exactly one array of [data type]. The elements of the array can be UDT data type or any other elementary data type. The counting of the array always starts with the low limit "0", regardless of the later declaration of the array.

If the array DB is marked with the "Only store in load memory" block attribute, it is only stored in load memory.

The instruction is executed when a positive signal edge is detected at the REQ parameter. The BUSY parameter then has the signal state "1". If a negative signal edge is detected at the BUSY parameter, the instruction is terminated and the value at the VALUE parameter is written to the data block. The DONE parameter then has the signal state "1" for one program cycle.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Note

The array DB must be located in a data block with the block property "Optimized".

Syntax

The following syntax is used for the "Write to ARRAY data block in load memory" instruction:

SCL

```
"Instance DB"(REQ := <Operand>,
            DB := <Operand>,
            INDEX := <Operand>,
            VALUE := <Operand>,
            BUSY => <Operand>,
            DONE => <Operand>,
            ERROR => <Operand>)
```

Parameter

The following table shows the parameters of the "Write to the array data block in load memory" instruction:

Parameter	Declaration	Data type	Description
REQ	Input	BOOL	REQ = "1": Start writing to the array DB
DB	Input	DB_ANY	Data block to which data is written
INDEX	Input	DINT	Element in the DB to which data is written
VALUE	Input	VARIANT	Value to be written
BUSY	Output	BOOL	BUSY = "1": The array DB is still being written
DONE	Output	BOOL	DONE = "1": The instruction was executed successfully
ERROR	Output	INT	Error information: If an error occurs during the execution of the instruction, an error code is output on the ERROR parameter.

For additional information on valid data types, refer to "See also".

ERROR parameter

The following table shows the meaning of the values of the ERROR parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	Data types do not correspond

Error code* (W#16#...)	Explanation
80B5	Copying process was interrupted
80C3	Too many concurrent calls
8132	The data block does not exist or is too short
8133	The data block is not located in the load memory
8134	The data block is write protected
8135	The DB is not an array data block.
8282	The value of the INDEX parameter is outside the limits of the array.
8350	The VARIANT data type contains no value
8352	Code generation error
8353	Code generation error
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

You can find descriptions of the error codes triggered by the instructions "READ_DBL: Read from data block in load memory" and "WRIT_DBL: Write to data block in load memory" under "See also".

Example

The following example shows how the instruction works:

SCL

```
"WriteToArrayDBL_DB"(REQ := "TagReg",
                     DB := "ArrayDB",
                     INDEX := "ArrayDB"."THIS"[2],
                     VALUE := "SourceTargetField[1]".Data1[6],
                     BUSY => "TagBusy",
                     DONE => "TagDone",
                     ERROR => "TagError");
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
REQ	TagReq	BOOL
DB	ArrayDB	The data type of the "ArrayDB" operand is Array [0 to 10] of INT.
INDEX	ArrayDB."THIS"[2]	Third element of the "ArrayDB"
VALUE	SourceTargetField[1].Data1[6]	The data type of the "SourceTargetField" operand is Array [0 to 10] of "MOVE_UDT".
BUSY	TagBusy	BOOL
DONE	TagDone	BOOL

If a positive edge is detected at the "TagReq" operand, the "Write to ARRAY data block in load memory" instruction is executed. As soon as a negative signal edge is detected at the "TagBusy" operand, the instruction is terminated and the value at the VALUE parameter is written in the third element in the "ArrayDB". After the instruction has been processed, the "TagDone" operand the signal state "1".

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

READ_DBL: Read from data block in the load memory (Page 4801)

WRIT_DBL: Write to data block in the load memory (Page 4803)

Read/write access

PEEK: Read memory address

Description

The "Read memory address" instruction is used to read a memory address from a memory area without specifying a data type.

If you specify the 16#84 area at the AREA parameter for a data block, you can only access data blocks with the "Standard" block property.

Syntax

The following syntax is used for the "Read memory address" instruction:

SCL

```
PEEK(AREA:= <Operand>,
     DBNUMBER := <Operand>,
     BYTEOFFSET:= <Operand>)
```

SCL

```
PEEK_<Data type>(AREA := <Operand>,
                 DBNUMBER := <Operand>,
                 BYTEOFFSET:= <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
AREA	Input	BYTE	The following areas can be selected: • 16#81: Input • 16#82: Output • 16#83: Bit memory • 16#84: DB • 16#1: Peripheral input (S7-1500 only)
DBNUMBER	Input	DINT, DB_ANY	Number of the data block if AREA = DB, otherwise "0"
BYTEOFFSET	Input	DINT	Address to read from Only the 16 least significant bits are used.
_<Data type>		Bit strings default: BYTE	Data type of the function value: • You do not need to specify the data type if using the default. • Any other valid data type you may use must be declared explicitly.
Function value		Bit strings	Result of the instruction

For additional information on valid data types, refer to "See also".

Note

If you read the memory address of the input, output or bit memory areas, you will need to set the "DBNUMBER" parameter with the value "0", as the instruction is faulty otherwise.

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result1" := PEEK(AREA := "Tag_Area",
                      DBNUMBER := "Tag_DBNumber",
                      BYTEOFFSET := "Tag_Byte");
```

SCL

```
"Tag_Result2" := PEEK_WORD(AREA := "Tag_Area",
                           DBNUMBER := "Tag_DBNumber",
```

SCL

```
BYTEOFFSET := "Tag_Byte");
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
AREA	Tag_Area	16#84
DBNUMBER	Tag_DBNumber	5
BYTEOFFSET	Tag_Byte	20
Function value	Tag_Result1	Byte value "20"
Function value	Tag_Result2	Word value "20"

The instruction reads the value of address "20" from the "Tag_Byte" operand at data block "5" and returns the result as a function value at the "Tag_Result" operand.

See also

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Overview of the valid data types (Page 2236)

PEEK_BOOL: Read memory bit

Description

The "Read memory bit" instruction is used to read a memory bit from a memory area without specifying a data type.

If you specify the 16#84 area at the AREA parameter for a data block, you can only access data blocks with the "Standard" block property.

Note

The instruction can only be used to access "Standard" memory areas.

Syntax

The following syntax is used for the "Read memory bit" instruction:

SCL

```
PEEK_BOOL(AREA := <Operand>,
          DBNumber := <Operand>,
          BYTEEOFFSET := <Operand>,
          BITOFFSET := <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
AREA	Input	BYTE	The following areas can be selected: • 16#81: Input • 16#82: Output • 16#83: Bit memory • 16#84: DB • 16#1: Peripheral input (S7-1500 only)
DBNUMBER	Input	DINT, DB_ANY	Number of the data block if AREA = DB, otherwise "0"
BYTEOFFSET	Input	DINT	Address to read from Only the 16 least significant bits are used.
BITOFFSET	Input	INT	Bit to be read from
Function value		BOOL	Result of the instruction

For additional information on valid data types, refer to "See also".

Note

If you read the memory bit from the input, output or bit memory areas, you will need to set the "DBNUMBER" parameter with the value "0", as the instruction is faulty otherwise.

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := PEEK_BOOL(AREA := "Tag_Area",
                          DBNUMBER := "Tag_DBNumber",
                          BYTEOFFSET := "Tag_Byte",
                          BITOFFSET := "Tag_Bit");
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
AREA	Tag_Area	16#84
DBNUMBER	Tag_DBNumber	5
BYTEOFFSET	Tag_Byte	20
BITOFFSET	Tag_Bit	3
Function value	Tag_Result	3

The instruction reads the value of memory bit "3" from the "Tag_Bit" operand at byte "20" of data block "5" and returns the result at the "Tag_Result" operand as function value.

See also

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Overview of the valid data types (Page 2236)

POKE: Write memory address

Description

The "Write memory address" instruction is used to write a memory address to a memory area without specifying a data type.

If you specify the 16#84 area at the AREA parameter for a data block, you can only access data blocks with the "Standard" block property.

Note

The instruction can only be used to access "Standard" memory areas.

Syntax

The following syntax is used for the "Write memory address" instruction:

SCL

```
POKE(AREA := <Operand>,
     DBNUMBER := <Operand>,
     BYTEOFFSET := <Operand>,
     VALUE := <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
AREA	Input	BYTE	The following areas can be selected: • 16#81: Input • 16#82: Output • 16#83: Bit memory • 16#84: DB • 16#2: Peripheral output (S7-1500 only)
DBNUMBER	Input	DINT, DB_ANY	Number of the data block if AREA = DB, otherwise "0"

Parameter	Declaration	Data type	Description
BYTEOFFSET	Input	DINT	Address to be written Only the 16 least significant bits are used.
VALUE	Input	Bit strings	Value to be written

For additional information on valid data types, refer to "See also".

Note

If you write the memory address in the input, output or bit memory areas, you will need to set the "DBNUMBER" parameter with the value "0", as the instruction is faulty otherwise.

Example

The following example shows how the instruction works:

SCL

```
POKE(AREA := "Tag_Area",
     DBNUMBER := "Tag_DBNumber",
     BYTEOFFSET := "Tag_Byte"),
     VALUE := "Tag_Value";
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
AREA	Tag_Area	16#84
DBNUMBER	Tag_DBNumber	5
BYTEOFFSET	Tag_Byte	20
VALUE	Tag_Value	16#11

The instruction overwrites the memory address "20" in the data block "5" with value "16#11".

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

POKE_BOOL: Write memory bit

Description

The "Write memory bit" instruction is used to write a memory bit to a memory area without specifying a data type.

If you specify the 16#84 area at the AREA parameter for a data block, you can only access data blocks with the "Standard" block property.

Note

The instruction can only be used to access "Standard" memory areas.

Syntax

The following syntax is used for the "Write memory bit" instruction:

SCL

```
POKE_BOOL(AREA := <Operand>,
          DBNUMBER := <Operand>,
          BYTEOFFSET := <Operand>,
          BITOFFSET := <Operand>,
          VALUE := <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
AREA	Input	BYTE	The following areas can be selected: • 16#81: Input • 16#82: Output • 16#83: Bit memory • 16#84: DB • 16#2: Peripheral output (S7-1500 only)
DBNUMBER	Input	DINT, DB_ANY	Number of the data block if AREA = DB, otherwise "0"
BYTEOFFSET	Input	DINT	Address to be written Only the 16 least significant bits are used.
BITOFFSET	Input	INT	Bit to be written
VALUE	Input	BOOL	Value to be written

For additional information on valid data types, refer to "See also".

Note

If you write the memory bit in the input, output or bit memory areas, you need to set the "DBNUMBER" parameter with the value "0", as the instruction is faulty otherwise.

Example

The following example shows how the instruction works:

```
SCL
POKE_BOOL(AREA := "Tag_Area",
          DBNUMBER := "Tag_DBNumber",
          BYTEOFFSET := "Tag_Byte",
          BITOFFSET := "Tag_Bit",
          VALUE := "Tag_Value");
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
AREA	Tag_Area	16#84
DBNUMBER	Tag_DBNumber	5
BYTEOFFSET	Tag_Byte	20
BITOFFSET	Tag_Bit	3
VALUE	Tag_Value	M0.0

The instruction overwrites the memory bit "3" in data block "5" in byte "20" with the value "M0.0".

See also

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Overview of the valid data types (Page 2236)

POKE_BLK: Write memory area

Description

The "Write memory area" instruction copies the content a memory area to a different memory area without specifying a data type.

If you specify the 16#84 area at the AREA parameter for a data block, you can only access data blocks with the "Standard" block property.

Note

The instruction can only be used to access "Standard" memory areas.

Syntax

The following syntax is used for the "Write memory area" instruction:

SCL

```
POKE_BLK(AREA_SRC:= <Operand>,
         DBNUMBER_SRC:= <Operand>,
         BYTEOFFSET_SRC:= <Operand>,
         AREA_DEST:= <Operand>,
         DBNUMBER_DEST:= <Operand>,
         BYTEOFFSET_DEST:= <Operand>,
         COUNT:= <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
AREA_SRC	Input	BYTE	The following areas can be selected in the source memory area: • 16#81: Input • 16#82: Output • 16#83: Bit memory • 16#84: DB
DBNUMBER_SRC	Input	DINT, DB_ANY	Number of the data block in the source memory area, if AREA = DB, otherwise "0"
BYTEOFFSET_SRC	Input	DINT	Address in the source memory area to be written Only the 16 least significant bits are used.
AREA_DEST	Input	BYTE	The following areas can be selected in the destination memory area: • 16#81: Input • 16#82: Output • 16#83: Bit memory • 16#84: DB
DBNUMBER_DEST	Input	DINT, DB_ANY	Number of the data block in the destination memory area, if AREA = DB, otherwise "0"
BYTEOFFSET_DEST	Input	DINT	Address in the destination memory area to be written Only the 16 least significant bits are used.
COUNT	Input	DINT	Number of bytes which are copied

For additional information on valid data types, refer to "See also".

Note

If you write the memory address in the input, output or bit memory areas, you will need to set the "DBNUMBER" parameter with the value "0", as the instruction is faulty otherwise.

Example

The following example shows how the instruction works:

SCL

```
POKE_BLK(AREA_SRC := "Tag_Source_Area",
         DBNUMBER_SRC := "Tag_Source_DBNumber",
         BYTEOFFSET_SRC := "Tag_Source_Byte"),
         AREA_DEST := "Tag_Destination_Area",
         DBNUMBER_DEST := "Tag_Destination_DBNumber",
         BYTEOFFSET_DEST := "Tag_Destination_Byte",
         COUNT := "Tag_Count");
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
AREA_SRC	Tag_Source_Area	16#84
DBNUMBER_SRC	Tag_Source_DBNumber	5
BYTEOFFSET_SRC	Tag_Source_Byte	20
AREA_DEST	Tag_Destination_Area	16#83
DBNUMBER_DEST	Tag_Destination_DBNumber	0
BYTEOFFSET_DEST	Tag_Destination_Byte	30
COUNT	Tag_Count	100

The instruction writes 100 byte from data block "5" starting with address "20" in the memory area of the bit memory starting at address "30".

See also

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Overview of the valid data types (Page 2236)

READ_LITTLE: Read data in little endian format

Description

You use the "Read data in little endian format" instruction to read the data to a single tag in the little endian byte sequence. With the little endian format, the byte with the least significant

bits is saved first, which means at the lowest memory address. The memory area that is read from must have the data type ARRAY of BYTE.

The operand at the DEST_VARIABLE parameter must have an elementary data type. The operand at the POS parameter determines the position at which the reading starts.

Note

Reading tag of data type VARIANT or BOOL

If you want to read a tag to which a VARIANT points, use the "Serialize" or "Deserialize" instructions.

If you want to read a tag of the data type BOOL, use a "Slice access".

Syntax

Use the following syntax for the "Read data in little endian format" instruction:

SCL

```
READ_LITTLE(SRC_ARRAY := <Operand>,
            DEST_VARIABLE => <Operand>,
            POS := <Operand>)
```

Parameter

The following table shows the parameters of the "Read data in little endian format" instruction:

Parameter	Declaration	Data type	Description
SRC_ARRAY	Input	ARRAY of BYTE	Memory area from which is read
DEST_VARIABLE	Output	Bit strings, integers, floating-point numbers, timers, LDT, TOD, LTOD, DATE, CHAR, WCHAR	Read value
POS	InOut	DINT	Determines the position at which the reading starts. The POS parameter is calculated zero-based.
Function value		INT	Error information

Example

The following example shows how the instruction works:

SCL

```
#TagResult := READ_LITTLE(SRC_ARRAY := #SourceField,
                          DEST_VARIABLE => #DINTVariable,
```

SCL

```
POS := #TagPos);
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
SRC_ARRAY	#SourceField	ARRAY[0..3] of BYTE := 16#1A, 16#2B, 16#3C, 16#4D
DEST_VARIABLE	#DINTVariable	1295788826 16#4D3C2B1A
POS	#TagPos	0 => 4

The "Read data in little endian format" instruction reads the integer 1_295_788_826 from the memory area "#SourceField" and writes it to the "#DINTVariable" operand in little endian format. The data type at the DEST_VARIABLE parameter specifies how many bytes are read. The quantity 4 is saved at the "#TagPos" operand.

See also

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Overview of the valid data types (Page 2236)

Deserialize: Deserialization (Page 4181)

Serialize: Serialization (Page 4184)

Addressing areas of a tag with slice access (Page 2216)

WRITE_LITTLE: Write data in little endian format

Description

You use the "Write data in little endian format" instruction to write the data of a single tag in the little endian byte sequence to a memory area. With the little endian format, the byte with the least significant bits is saved first, which means at the lowest memory address. The memory area written to must have the data type ARRAY of BYTE.

The operand at the SRC_VARIABLE parameter must have an elementary data type. The operand at the POS parameter determines the position at which the writing starts.

Note

Writing tag of data type VARIANT or BOOL

If you want to write a tag to which a VARIANT points, use the "Serialize" or "Deserialize" instructions.

If you want to write a tag of the data type BOOL, use a "Slice access".

Syntax

Use the following syntax for the "Write data in little endian format" instruction:

SCL

```
WRITE_LITTLE(SRC_VARIABLE := <Operand>,
             DEST_ARRAY := <Operand>,
             POS := <Operand>)
```

Parameter

The following table shows the parameters of the "Write data in little endian format" instruction:

Parameter	Declaration	Data type	Description
SRC_VARIABLE	Input	Bit strings, integers, floating-point numbers, timers, LDT, TOD, LTOD, DATE, CHAR, WCHAR	Tag whose data are written
DEST_ARRAY	InOut	ARRAY of BYTE	Memory area to which the data are written
POS	InOut	DINT	Determines the position at which the writing starts. The POS parameter is calculated zero-based.
Function value		INT	Error information

Example

The following example shows how the instruction works:

SCL

```
#TagResult := WRITE_LITTLE(SRC_VARIABLE := #DINTVariable,
                           DEST_ARRAY := #TargetField,
                           POS := #TagPos);
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
SRC_VARIABLE	#DINTVariable	1295788826 16#4D3C2B1A
DEST_ARRAY	#TargetField	ARRAY[0..10] of BYTE = 16#1A, 16#2B, 16#3C, 16#4D
POS	#TagPos	0 => 4

The "Write data in little endian format" instruction writes the integer 1_295_788_826 from the "#DINTVariable" operand to the "#TargetField" memory area in little endian format. The data type at the SRC_VARIABLE parameter specifies how many bytes are written. The quantity 4 is saved at the "#TagPos" operand.

See also

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Overview of the valid data types (Page 2236)

Deserialize: Deserialization (Page 4181)

Serialize: Serialization (Page 4184)

Addressing areas of a tag with slice access (Page 2216)

READ_BIG: Read data in big endian format

Description

You use the "Read data in big endian format" instruction to read the data to a single tag in the big endian byte sequence. With the big endian format, the byte with the most significant bits is saved first, which means at the lowest memory address. The memory area that is read from must have the data type ARRAY of BYTE.

The operand at the DEST_VARIABLE parameter must have an elementary data type. The operand at the POS parameter determines the position at which the reading starts.

Note

Reading tag of data type VARIANT or BOOL

If you want to read a tag to which a VARIANT points, use the "Serialize" or "Deserialize" instructions.

If you want to read a tag of the data type BOOL, use a "Slice access".

Syntax

Use the following syntax for the "Read data in big endian format" instruction:

SCL

```
READ_BIG(SRC_ARRAY := <Operand>,
         DEST_VARIABLE => <Operand>,
         POS := <Operand>)
```

Parameter

The following table shows the parameters of the "Read data in big endian format" instruction:

Parameter	Declaration	Data type	Description
SRC_ARRAY	Input	ARRAY of BYTE	Memory area from which is read
DEST_VARIABLE	Output	Bit strings, integers, floating-point numbers, timers, LDT, TOD, LTOD, DATE, CHAR, WCHAR	Read value
POS	InOut	DINT	Determines the position at which the reading starts. The POS parameter is calculated zero-based.
Function value		INT	Error information

Example

The following example shows how the instruction works:

SCL

```
#TagResult := READ_BIG(SRC_ARRAY := #SourceField,
                       DEST_VARIABLE => #DINTVariable,
                       POS := #TagPos);
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
SRC_ARRAY	#SourceField	ARRAY[0..10] of BYTE := 16#1A, 16#2B, 16#3C, 16#4D
DEST_VARIABLE	#DINTVariable	439041101 16#1A2B3C4D
POS	#TagPos	0 => 4

The "Read data in big endian format" instruction reads the integer 439_041_101 from the memory area "#SourceField" and writes it to the "#DINTVariable" operand in big endian format. The data type at the DEST_VARIABLE parameter specifies how many bytes are read. The quantity 4 is saved at the "#TagPos" operand.

See also

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Overview of the valid data types (Page 2236)

Deserialize: Deserialization (Page 4181)

Serialize: Serialization (Page 4184)

Addressing areas of a tag with slice access (Page 2216)

WRITE_BIG: Write data in big endian format

Description

You use the "Write data in big endian format" instruction to write the data of a single tag in the big endian byte sequence to a memory area. With the big endian format, the byte with the most significant bits is saved first, which means at the lowest memory address. The memory area written to must have the data type ARRAY of BYTE.

The operand at the SRC_VARIABLE parameter must have an elementary data type. The operand at the POS parameter determines the position at which the writing starts.

Note

Writing tag of data type VARIANT or BOOL

If you want to write a tag to which a VARIANT points, use the "Serialize" or "Deserialize" instructions.

If you want to write a tag of the data type BOOL, use a "Slice access".

Syntax

Use the following syntax for the "Write data in big endian format" instruction:

SCL

```
WRITE_BIG(SRC_VARIABLE := <Operand>,
          DEST_ARRAY := <Operand>,
          POS := <Operand>)
```

Parameter

The following table shows the parameters of the "Write data in big endian format" instruction:

Parameter	Declaration	Data type	Description
SRC_VARIABLE	Input	Bit strings, integers, floating-point numbers, timers, LDT, TOD, LTOD, DATE, CHAR, WCHAR	Tag whose data are written
DEST_ARRAY	InOut	ARRAY of BYTE	Memory area to which the data are written

Parameter	Declaration	Data type	Description
POS	InOut	DINT	Determines the position at which the writing starts. The POS parameter is calculated zero-based.
Function value		INT	Error information

Example

The following example shows how the instruction works:

SCL

```
#TagResult := WRITE_BIG(SRC_VARIABLE := #DINTVariable,
                        DEST_ARRAY := #TargetField,
                        POS := #TagPos);
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
SRC_VARIABLE	#DINTVariable	439041101 16#1A2B3C4D
DEST_ARRAY	#TargetField	ARRAY[0..10] of BYTE = 16#1A, 16#2B, 16#3C, 16#4D
POS	#TagPos	0 => 4

The "Write data in big endian format" instruction writes the integer 439_041_101 from the "#DINTVariable" operand to the "#TargetField" memory area in big endian format. The data type at the SRC_VARIABLE parameter specifies how many bytes are written. The quantity 4 is saved at the "#TagPos" operand.

See also

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Overview of the valid data types (Page 2236)

Deserialize: Deserialization (Page 4181)

Serialize: Serialization (Page 4184)

Addressing areas of a tag with slice access (Page 2216)

VARIANT instructions

VariantGet: Read out VARIANT tag value

Description

You can use the "Read out VARIANT tag value" instruction to read the value of the tag to which the VARIANT at the SRC parameter points and write it in the tag at the DST parameter.

The SRC parameter must have the VARIANT data type. Any data type except for VARIANT can be specified at the DST parameter.

The data type of the tag at the DST parameter must match the data type to which the VARIANT points.

Note

The tag to which the VARIANT at parameter SRC points must be a PLC data type.

To copy structures and ARRAYs, you can use the "MOVE_BLK_VARIANT: Move block" instruction. For additional information, refer to "See also".

Syntax

The following syntax is used for the "Read out VARIANT tag value" instruction:

SCL

```
VariantGet(SRC := <Operand>,
           DST => <Operand>)
```

Parameter

The following table shows the parameters of the "Read out VARIANT tag value" instruction:

Parameter	Declaration	Data type	Description
SRC	Input	VARIANT	Tag to be read
DST	Output	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY, structures	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
VariantGet(SRC := #TagIn_Source,
           DST => "TagOut_Dest");
```

The value of the tag to which the VARIANT at the "#TagIn_Source" operand points is read and written to the "TagOut_Dest" operand.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Switching display formats in the program status (Page 2980)

VARIANT (Page 2277)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

MOVE_BLK_VARIANT: Move block (Page 4189)

VariantPut: Write VARIANT tag value

Description

You can use the "Write VARIANT tag value" instruction to write the value of the tag at the SRC parameter to the tag at the DST parameter to which the VARIANT points.

The DST parameter must have the VARIANT data type. Any data type except for VARIANT can be specified at the SRC parameter.

The data type of the tag at the SRC parameter must match the data type to which the VARIANT points.

Note

The tag to which the VARIANT at parameter DST points must be a PLC data type.

Syntax

The following syntax is used for the "Write VARIANT tag value" instruction:

SCL

```
VariantPut(SRC := <Operand>,
           DST := <Operand>)
```

Parameter

The following table shows the parameters of the "Write VARIANT tag value" instruction:

Parameter	Declaration	Data type	Description
SRC	Input	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY	Tag to be read
DST	Input	VARIANT	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
VariantPut(SRC := "TagIn_Source",
           DST => "#TagIn_Dest");
```

The value of the "TagIn_Source" operand is written to the tag to which the VARIANT at the "#TagIn_Dest" operand points.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Switching display formats in the program status (Page 2980)

VARIANT (Page 2277)

Evaluating errors with output parameter RET_VAL (Page 5370)

Call function blocks as single or multi-instances (Page 2182)

Basic information on the status word (Page 2466)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

CountOfElements: Get number of ARRAY elements

Description

You can use the "Get number of ARRAY elements" instruction to query how many ARRAY elements a tag to which VARIANT is pointing has.

If it is a one-dimensional ARRAY, the difference between the high and low limit +1 is output as a result. If it is a multi-dimensional ARRAY, the product of all dimensions is output as the result.

The <Operand> must have the VARIANT data type.

The result is "0" if the VARIANT tag is not an ARRAY.

If the VARIANT points to an ARRAY of BOOL, the fill elements are included in the count. (For example, 8 is returned for an ARRAY[0..1] of BOOL)

Syntax

The following syntax is used for the "Get number of ARRAY elements" instruction:

SCL

```
CountOfElements(<Operand>)
```

Parameter

The following table shows the parameters of the "Get number of ARRAY elements" instruction:

Parameter	Declaration	Data type	Description
<Operand>	Input	VARIANT	Tag to be queried
Function value		UDINT	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
IF IS_ARRAY(#Tag_VARIANTToArray) THEN
"Tag_Result" := CountOfElements(#Tag_VARIANT);
END_IF;
```

If the tag to which the VARIANT points is an ARRAY, the number of ARRAY elements is output.

See also

Format of array (16-bit limits) (Page 2267)
Format of array (32-bit limits) (Page 2268)
Overview of the valid data types (Page 2236)
Basics of the EN/ENO mechanism (Page 2447)
Switching display formats in the program status (Page 2980)
VARIANT (Page 2277)
Evaluating errors with output parameter RET_VAL (Page 5370)
Call function blocks as single or multi-instances (Page 2182)
Basic information on the status word (Page 2466)
Entering SCL instructions (Page 2697)
Editing SCL instructions (Page 2716)

Other

BLKMOV: Move block

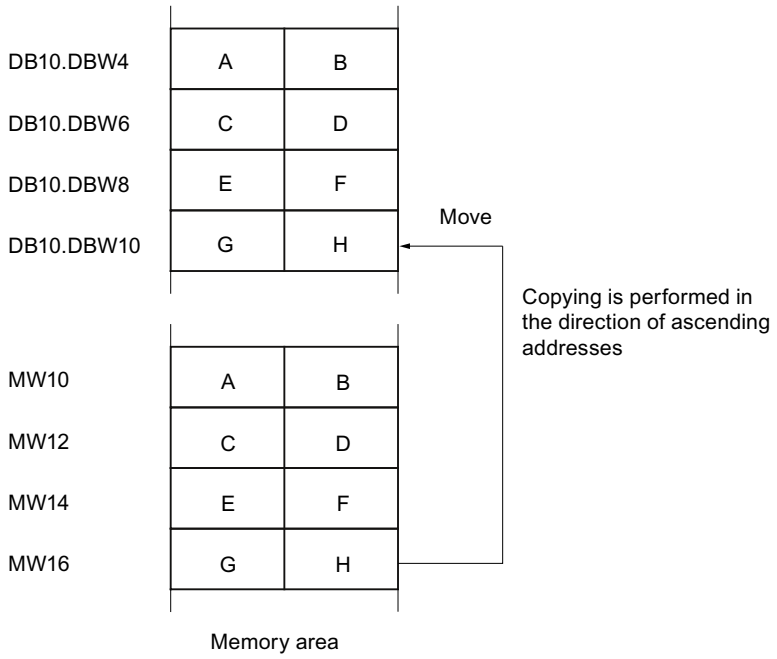
Description

The instruction "Move block" copies the contents of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use VARIANT to define the source and destination areas.

Note

The tags of the instruction can only be used in data blocks in which the attribute "Optimized block access" is not set. If the tag with the retentive setting "Set in IDB" has been declared, you can also use the tags "with optimized access".

The following figure shows the principle of the move operation:



Consistency of source data and destination data

Note that while the instruction "Move block" is being executed, the source data remains unchanged; otherwise the consistency of the destination data cannot be guaranteed.

Interruptibility

There is no limit to the nesting depth.

Memory areas

The instruction "Move block" moves the following memory areas:

- Areas of a data block
- Bit memory
- Process image of the inputs
- Process image of the outputs

General rules for moving

The source and destination areas must not overlap. If the source and destination area have different lengths, only the length of the smaller area will be moved.

If the source area is smaller than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is smaller than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a block of data type BOOL is moved, the tag must be addressed absolutely and the specified length of the area must be divisible by 8, otherwise the instruction cannot be executed.

Rules for moving character strings

You can use the "Move block" instruction to move source and destination areas of the STRING data type. If only the source area is of data type STRING, the characters will be moved that are actually contained in the character string. Information on the actual and maximum length is also written to the destination area. If the source and destination area are both STRING data type, the current length of the character string in the destination area is set to the number of characters actually moved.

If you want to move the information on the maximum and actual length of a character string, specify the areas in bytes in the SRCBLK and DSTBLK parameters.

Syntax

The following syntax is used for the "Move block" instruction:

SCL

```
BLKMOV(SRCBLK:= <Operand>,
       DSTBLK => <Operand>)
```

Parameter

The following table shows the parameters of the "Move block" instruction:

Parameter	Declaration	Data type	Description
SRCBLK	Input	VARIANT	Specifies the memory area to be moved (source area).
DSTBLK	Output	VARIANT	Specifies the memory area to which the block is to be moved (destination area).
Function value (RET_VAL)		INT	Error information

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8152	The data types WSTRING, WCHAR and BOOL are not supported.
8092	The source or target area is only available in the load memory.

Error code* (W#16#...)	Explanation
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Evaluating errors with output parameter RET_VAL (Page 5370)

Switching display formats in the program status (Page 2980)

GET_ERR_ID: Get error ID locally (Page 4282)

UBLKMOV: Move block uninterruptible

Description

The instruction "Move block uninterruptible" copies the contents of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use VARIANT to define the source and destination areas.

The move operation cannot be interrupted by other operating system activities. As a result the interrupt reaction time of the CPU can increase during the execution of the instruction "Move block uninterruptible".

Note

The tags of the instruction can only be used in data blocks in which the attribute "Optimized block access" is not set. If the tag with the retentive setting "Set in IDB" has been declared, you can also use the tags "with optimized access".

Memory areas

The instruction "Move block uninterruptible" copies the following memory areas:

- Areas of a data block
- Bit memory
- Process image of the inputs
- Process image of the outputs

General rules for moving

The source and destination area must not overlap during the execution of the instruction "Move block uninterruptible". If the source area is smaller than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is smaller than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a source or destination area defined as a formal parameter is less than a destination or source area specified at the SRCBLK or DSTBLK parameter, no data is transferred.

If a block of data type BOOL is moved, the tag must be addressed absolutely and the specified length of the area must be divisible by 8, otherwise the instruction cannot be executed.

You can use the "Move block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Rules for moving character strings

The instruction "Move block uninterruptible" copies source and destination areas of the data type STRING. If only the source area is of data type STRING, the characters will be moved that are actually contained in the character string. Information on the actual and maximum length are not written in the destination area. If the source and destination area are both STRING data type, the current length of the character string in the destination area is set to the number of characters actually moved. If areas of the STRING data type are moved, specify "1" as the area length.

Syntax

The following syntax is used for the "Move block uninterruptible" instruction:

SCL

```
UBLKMOV(SRCBLK:= <Operand>,
        DSTBLK => <Operand>)
```

Parameter

The following table shows the parameters of the "Move block uninterruptible" instruction:

Parameter	Declaration	Data type	Description
SRCBLK	Input	VARIANT	Specifies the memory area to be moved (source area).
DSTBLK	Output	ANY	Specifies the memory area to which the block is to be moved (destination area).
Function value (RET_VAL)		VARIANT	Error information

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code (W#16#...)	Explanation
0000	No error
8152	The data types WSTRING, WCHAR and BOOL are not supported.
8091	The source area is in a data block that is not relevant for program execution.
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Evaluating errors with output parameter RET_VAL (Page 5370)

Switching display formats in the program status (Page 2980)

GET_ERR_ID: Get error ID locally (Page 4282)

FILL: Fill block

Description

The instruction "Fill block" fills a memory area (destination area) with the content of another memory area (source area). The instruction "Fill block" moves the content of the source area to the destination area until the destination area is completely written. The move operation takes place in the direction of ascending addresses.

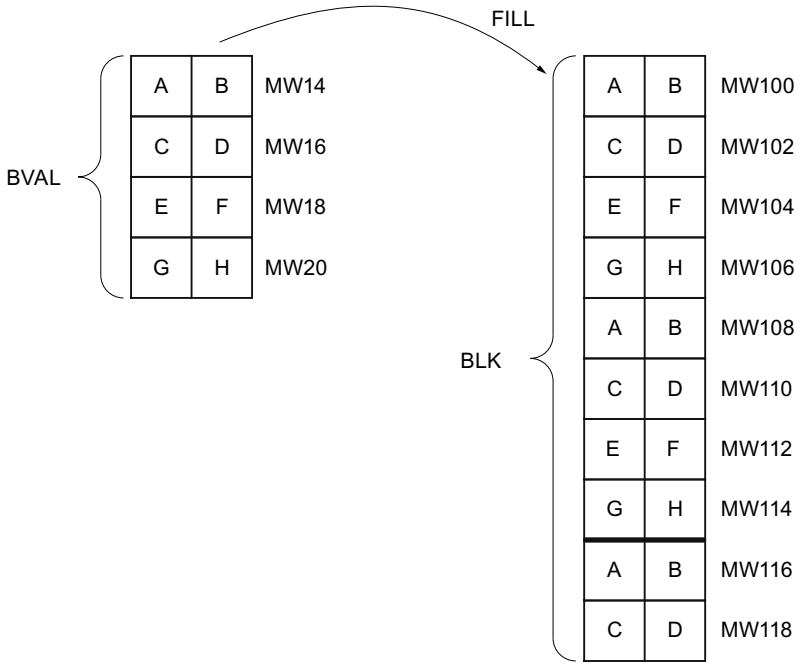
You use VARIANT to define the source and destination areas.

Note

The tags of the instruction can only be used in data blocks in which the attribute "Optimized block access" is not set. If the tag with the retentive setting "Set in IDB" has been declared, you can also use the tags "with optimized access".

For blocks with the "Optimized block access" attribute, you can use the "FILL_BLK: Fill block" instruction.

The following figure shows the principle of the move operation:



Example: The contents of the range MW100 to MW118 are to be preassigned with the contents of the memory words MW14 to MW20.

Consistency of source data and destination data

Note that while the instruction "Fill block" is being executed, the source data remains unchanged; otherwise the consistency of the destination data cannot be guaranteed.

Memory areas

The instruction "Fill block" copies the following memory areas:

- Areas of a data block
- Bit memory
- Process image of the inputs
- Process image of the outputs

The instruction "Fill block" copies the following memory areas:

- Instructions, system data blocks
- Counters
- Timers
- Memory areas of the I/O area

General rules for moving

The source and destination areas must not overlap. If the destination area to be preset is not an integer multiple of the length of the input parameter BVAL, the destination area is nevertheless written up to the last byte.

If the destination area to be preset is smaller than the source area, the function only copies as much data as can be written to the destination area.

If the destination or source block actually present is smaller than the assigned memory area for the source or destination area (BVAL, BLK parameters), no data will be transferred.

If the ANY pointer (source or destination) is of the data type BOOL, it must be addressed absolutely and the length specified must be divisible by 8; otherwise the instruction is not executed.

If the destination area is STRING data type, the instruction writes the entire string including the administration information.

Syntax

The following syntax is used for the instruction "Fill block":

SCL

```
FILL(BVAL:= <Operand>,
     BLK => <Operand>)
```

Parameter

The following table shows the parameters of the "Fill block" instruction:

Parameter	Declaration	Data type	Description
BVAL	Input	VARIANT	Specification of the memory area (source area) with whose content the destination area on the BLK parameter will be filled.
BLK	Output	VARIANT	Specification of the memory area that will be filled with the content of the source area.
Function value (RET_VAL)		INT	Error information

For additional information on valid data types, refer to "See also".

BVAL Parameter

When you transfer a structure as an input parameter, remember that the length of a structure is always based on an even number of bytes. The structure will need one byte of additional memory space if you declare a structure with an odd number of bytes.

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code (W#16#...)	Explanation
0000	No error
8152	The data types WSTRING, WCHAR and BOOL are not supported.
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Evaluating errors with output parameter RET_VAL (Page 5370)

Switching display formats in the program status (Page 2980)

GET_ERR_ID: Get error ID locally (Page 4282)

Conversion operations

CONVERT: Convert value

Description

Use the "Convert value" instruction to program explicit conversions. You determine the data types to be converted in a dialogue box with opens automatically when you insert the instruction. When executed, the instruction reads the source value and converts it into the specified target value.

For information on possible conversions, refer to section "Explicit conversion" under "See also".

Note

For S7-1500 CPU: You can select the data types DWORD and LWORD if REAL or LREAL were selected as IN data type.

Syntax

The following syntax is used for the "Convert value" instruction:

```
SCL
<Target_value> := <Conversion_function>(<Source_value>);
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type		Description
		S7-1200	S7-1500	
<Source_value>	Input, constant	Bit strings, integers, floating-point numbers, TIME, DATE, TOD, DTL, strings	Bit strings, integers, floating-point numbers, TIME, LTIME, DATE, TOD, LTOD, DTL, strings	Value to be converted.
<Conversion_function>	-	-	-	Function that specifies the data type to be converted .
<Target_value>	Output	Bit strings, integers, floating-point numbers, TIME, DATE, TOD, DTL, strings	Bit strings, integers, floating-point numbers, TIME, LTIME, DATE, TOD, LTOD, DTL, strings	Result of the conversion

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_INT" := REAL_TO_INT("Tag_REAL");
```

The following table shows how the instruction works using specific operand values:

Operand	Data type	Value
Tag_REAL	REAL	20.56
Tag_INT	INT	21

During the conversion, the value of the "Tag_REAL" operand will be rounded to the nearest integer and saved in the "Tag_INT" operand.

See also

Overview of the valid data types (Page 2236)
Operators and operator precedence (Page 2687)
Entering SCL instructions (Page 2697)
Editing SCL instructions (Page 2716)
Data type conversion (Page 2286)

ROUND: Round numerical value

Description

The "Round numerical value" instruction is used to round the value at input IN to the nearest integer. The instruction interprets the value at input IN as a floating-point number and converts it into an integer or floating-point number. If the input value is exactly between an even and odd number, the even number is selected.

Syntax

The following syntax is used for the "Round numerical value" instruction:

SCL
```
ROUND(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
<Expression>	Input	Floating-point numbers	Input value to be rounded.
Function value		Integers, floating-point numbers	Result of the rounding

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL
```
"Tag_Result" := ROUND("Tag_Value");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_Value	1.50000000	-1.50000000
Tag_Result	2	-2

See also

Overview of the valid data types (Page 2236)

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Data type conversion (Page 2286)

CEIL: Generate next higher integer from floating-point number

Description

Use the "Generate next higher integer from floating-point number" instruction to round the value to the nearest integer. The instruction interprets the input value as floating-point number and converts it to the next higher integer. The function value can be greater than or equal to the input value.

Syntax

The following syntax is used for the "Generate next higher integer from floating-point number" instruction:

```
SCL
CEIL(<Expression>)
CEIL_<Data type>(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
<Expression>	Input	Floating-point numbers	Input value
_<Data type>		Integers, floating-point numbers. Default: DINT	Data type of the function value: • You do not need to specify the data type if using the default. • Any other valid data type you may use must be declared explicitly.
Function value		Integers, floating-point numbers	Input value rounded up

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result1" := CEIL("Tag_Value");
"Tag_Result2" := CEIL_REAL("Tag_Value);
```

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_Value	0.5	-0.5
Tag_Result1	1	0
Tag_Result2	1.0	0.0

The result of the instruction is returned in the operand "Tag_Result" as a function value.

See also

Overview of the valid data types (Page 2236)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

FLOOR: Generate next lower integer from floating-point number

Description

Use the "Generate next lower integer from floating-point number" instruction to round the value of a floating point number to the next lower integer. The instruction interprets the input value as floating-point number and converts it to the next lower integer. The function value can be equal or less than the input value.

Syntax

Use the following syntax for the "Generate next lower integer from floating-point number" instruction:

```
SCL
FLOOR(<Expression>)
FLOOR_<data type>(<expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
<Expression>	Input	Floating-point numbers	Input value
_<Data type>		Integers, floating-point numbers. Default: DINT	Data type of the function value: • You do not need to specify the data type if using the default. • Any other valid data type you may use must be declared explicitly.
Function value		Integers, floating-point numbers	Input value rounded

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result1" := FLOOR("Tag_Value");
"Tag_Result2" := FLOOR_REAL("Tag_Value");
```

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_Value	0.5	-0.5
Tag_Result1	0	-1
Tag_Result2	0.0	-1.0

The result of the instruction is returned in the operand "Tag_Result" as a function value.

See also

Overview of the valid data types (Page 2236)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

TRUNC: Truncate numerical value

Description

The "Truncate numerical value" instruction is used to generate an integer from the input value without rounding. The instruction selects only the integer part of the input value and returns this part without decimal places as the function value.

Syntax

The following syntax is used for the "Truncate numerical value" instruction:

```
SCL
TRUNC(<Expression>)
TRUNC_<Data type>(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
<Expression>	Input	Floating-point numbers	Input value
_<Data type>		Integers, floating-point numbers Default: DINT	Data type of the function value: • You do not need to specify the data type if using the default. • Any other valid data type you may use must be declared explicitly.
Function value		Integers, floating-point numbers	Integer component of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result1" := TRUNC("Tag_Value1");
"Tag_Result2" := TRUNC("Tag_Value2"+"Tag_Value3");
"Tag_Result3" := TRUNC_SINT("Tag_Value4");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value1	-1.5
Tag_Result1	-1
Tag_Value2	2.1
Tag_Value3	3.2
Tag_Result2	5
Tag_Result3	2
Tag_Value4	2.4

See also

Overview of the valid data types (Page 2236)

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

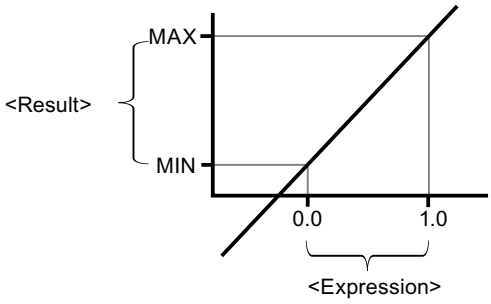
Editing SCL instructions (Page 2716)

SCALE_X: Scale

Description

Use the "Scale" instruction to scale a floating-point number by mapping it to a specific value range. You specify the value range with the MIN and MAX parameters. The result of the scaling is an integer.

The following figure shows an example of how values can be scaled:



The instruction "Scale" works with the following equation:

OUT = [VALUE * (MAX – MIN)] + MIN

Note

For more information on the conversion of analog values, refer to the respective manual.

Syntax

The following syntax is used for the "Scale" instruction:

```
SCL
SCALE_X(MIN := <Operand>,
        VALUE := <Operand>,
        MAX := <Operand>)
SCALE_X_<Data type>(MIN := <Operand>,
                    VALUE := <Operand>,
                    MAX := <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
MIN	Input	Integers, floating-point numbers	Low limit of the value range
VALUE	Input	Floating-point numbers	Value to be scaled. If you enter a constant, you must declare it.
MAX	Input	Integers, floating-point numbers	High limit of the value range
_<Data type>		Integers, floating-point numbers Default: INT	Data type of the function value: • You do not need to specify the data type if using the default. • Any other valid data type you may use must be declared explicitly.
Function value		Integers, floating-point numbers	Result of scaling

For additional information on valid data types, refer to "See also".

For additional information on declaring constants, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result1" := SCALE_X(MIN := "Tag_Value1",
                         VALUE := "Tag_Real",
                         MAX := "Tag_Value2");
"Tag_Result2" := SCALE_X_REAL(MIN := "Tag_Value1",
                              VALUE := "Tag_Real",
                              MAX := "Tag_Value2");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Real	0.5
Tag_Value1	10
Tag_Value2	30
Tag_Result1	20
Tag_Result2	20.0

See also

Overview of the valid data types (Page 2236)

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

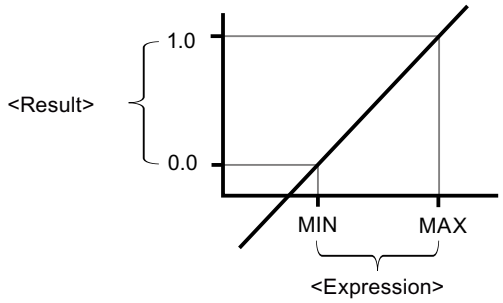
Declaring global constants (Page 2489)

NORM_X: Normalize

Description

You can use the instruction "Normalize" to normalize the value of the tag at the VALUE input by mapping it to a linear scale. You can use the MIN and MAX parameters to define the limits of a value range that is applied to the scale. Depending on the location of the normalized value in this value range, the result at output OUT is calculated and stored as a floating-point number. If the value to be normalized equals the value at input MIN, the instruction returns returns the result "0.0". If the value to be normalized equals the value at input MAX, the instruction returns returns the result "1.0".

The following figure shows an example of how values can be normalized:



The "Normalize" instruction works with the following equation:

OUT = (VALUE – MIN) / (MAX – MIN)

Note

For more information on the conversion of analog values, refer to the respective manual.

Syntax

The following syntax is used for the "Normalize" instruction:

SCL

```
NORM_X(MIN := <Operand>,
       VALUE := <Operand>,
       MAX := <Operand>)
NORM_X_<Data type>(MIN := <Operand>,
                   VALUE := <Operand>,
                   MAX := <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
MIN [1)]	Input	Integers, floating-point numbers	Low limit of the value range
VALUE [1)]	Input	Integers, floating-point numbers	Value to be normalized.
MAX [1)]	Input	Integers, floating-point numbers	High limit of the value range

Parameter	Declaration	Data type	Description
_<Data type>		Floating-point numbers Default: REAL	Data type of the function value: • You do not need to specify the data type if using the default. • Any other valid data type you may use must be declared explicitly.
Function value		Floating-point numbers	Result of the normalization
[1] If you use constants in these three parameters, you only need to declare one of them.			

For additional information on valid data types, refer to "See also".

For additional information on declaring constants, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result1" := NORM_X(MIN := "Tag_Value1",
                        VALUE := "Tag_InputValue",
                        MAX := "Tag_Value2");
"Tag_Result2" := NORM_X_LREAL(MIN := "Tag_Value1",
                              VALUE := "Tag_InputValue",
                              MAX := "Tag_Value2");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InputValue	20
Tag_Value1	10
Tag_Value2	30
Tag_Result1	0.5
Tag_Result2	0.5

See also

Overview of the valid data types (Page 2236)

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Declaring global constants (Page 2489)

VARIANT instructions

VARIANT_TO_DB_ANY: Convert VARIANT to DB_ANY

Description

The instruction "Convert VARIANT to DB_ANY" is used to read the number of the data block from a VARIANT and convert it to the data type DB_ANY.

Requirements

If the conditions are met, the instruction is executed. If the conditions are not met, "0" is output as data block number.

The output tag...	homed...	Conversion options
VARIANT	...a data block which is an instance data block of an UDT or SDT	The output tag can be converted to a data block number.
VARIANT	...an Array DB	The output tag can be converted to a data block number.
VARIANT	...an object with an elementary data type	It is not possible to convert the output tag to a database number because a data block can never comprise only one elementary data type.
VARIANT	...a structure within a data block	It is not possible to convert the output tags to a database number because it is only a part within the data block.

Syntax

The following syntax is used for the "Convert VARIANT To DB_ANY" instruction:

SCL

```
VARIANT_TO_DB_ANY(IN := <Operand>,
                  ERR => <Operand>)
```

Parameter

The following table shows the parameters of the "Convert VARIANT to DB_ANY" instruction:

Parameter	Declaration	Data type	Description
IN	Input	VARIANT	Tag to be read
ERR	Output	INT	Error information
Function value		DB_ANY	Result: Number of the DB

For additional information on valid data types, refer to "See also".

ERR parameter

The following table shows the meaning of the values of the ERR parameter:

Error code* (W#16#...)	Explanation
0000	No error
252C	The VARIANT data type has the value "0"
8130	The number of the data block is "0"
8131	The data block does not exist
8132	The data block is too short
8134	The data block is write protected
8150	The VARIANT data type has the value "0"
8154	The data type is not correctly declared
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := VARIANT_TO_DB_ANY(IN := "Tag_In",
                                  ERR := "Tag_Err");
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_In	The data type of the Tag_In operand is VARIANT. The value is read.
<Function value>	Tag_Result	Number of the block is output.

The third element is read from "Tag_In" operand and written to the "Tag_Result" operand.

See also

Overview of the valid data types (Page 2236)
Operators and operator precedence (Page 2687)
Entering SCL instructions (Page 2697)
Editing SCL instructions (Page 2716)

DB_ANY_TO_VARIANT: Convert DB_ANY to VARIANT

Description

The instruction "Convert DB_ANY to VARIANT" is used to read the number of the data block from a DB_ANY and convert it to the data type VARIANT.

Requirements

If the conditions are met, the instruction is executed. If the conditions are not met or the data block does not exist, the value "0" is output. All additional accesses to this value "0" fail.

The output tag...	homed...	Conversion options
DB_ANY	...a data block which is an instance data block of an UDT or SDT	It is possible to convert the output tag to the VARIANT data type.
DB_ANY	...a DB which is an ARRAY DB	It is possible to convert the output tag to the VARIANT data type.
DB_ANY	...a data block which is an instance data block of a function block or a global data block	It is not possible to convert the output tags to the VARIANT data type.

Syntax

The following syntax is used for the "Convert DB_ANY to VARIANT" instruction:

SCL

```
DB_ANY_TO_VARIANT(IN := <Operand>,
                  ERR => <Operand>)
```

Parameter

The following table shows the parameters of the "Convert DB_ANY to VARIANT" instruction:

Parameter	Declaration	Data type	Description
IN	Input	DB_ANY	Tag to be read
ERR	Output	INT	Error information
Function value		VARIANT	Result

For additional information on valid data types, refer to "See also".

ERR parameter

The following table shows the meaning of the values of the ERR parameter:

Error code* (W#16#...)	Explanation
0000	No error
8130	The number of the data block is "0"
8131	The data block does not exist
8132	The data block is too short
8134	The data block is write protected
8154	The data type is not correctly declared
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := DB_ANY_TO_VARIANT(IN := "Tag_In",
                                  ERR := "Tag_Err");
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_In	The data type of the Tag_In operand is DB_ANY. The value is read.
<Function value>	Tag_Result	Number of the block is converted and output to the VARIANT data type.

The third element is read from "Tag_In" operand and written to the "Tag_Result" operand.

See also

Overview of the valid data types (Page 2236)

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Other

SCALE: Scale

Description

The instruction "Scale" converts the integer on the parameter IN into a floating-point number, which can be scaled in physical units between a low and a high limit. You can use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is scaled. The result of the instruction is output on the OUT parameter.

The instruction "Scale" works with the following equation:

OUT = [((FLOAT (IN) – K1)/(K2–K1)) * (HI_LIM–LO_LIM)] + LO_LIM

The values of the "K1" and "K2" constants are determined by the signal state on the BIPOLAR parameter. The following signal states are possible on the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case the "K1" constant has the value "-27648.0" and the "K2" constant the value "+27648.0".
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case the "K1" constant has the value "0.0" and the "K2" constant the value "+27648.0".

When the value at the IN parameter is greater than the value of the "K2" constant, the result of the instruction is set to the value of the high limit (HI_LIM) and an error is output.

When the value at the IN parameter is less than the value of the "K1" constant, the result of the instruction is set to the value of the low limit value (LO_LIM) and an error is output.

When the indicated low limit is greater than the high limit (LO_LIM >HI_LIM), the result is scaled in reverse proportion to the input value.

Syntax

The following syntax is used for the "Scale" instruction:

SCL

```
SCALE(IN:= <Expression>,
      HI_LIM := <Operand>,
      LO_LIM := <Operand>,
      BIPOLAR := <Operand>,
      OUT => <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
IN	Input	INT	Input value to be scaled.
HI_LIM	Input	REAL	High limit
LO_LIM	Input	REAL	Low limit

Parameter	Declaration	Data type	Description
BIPOLAR	Input	BOOL	Indicates if the value at the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
OUT	Output	REAL	Result of the instruction
Function value (RET_VAL)		WORD	Error information

For additional information on valid data types, refer to "See also".

RET_VAL Parameter

The following table shows the meaning of the values of the parameter RET_VAL:

Error code (W#16#...)	Explanation
0000	No error
0008	The value of the parameter IN is greater than 27 648 or is less than 0 (unipolar) or -27 648 (bipolar).
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

```
"Tag_ErrorCode" := SCALE(IN := "Tag_InputValue",
                         HI_LIM := "Tag_HighLimit"
                         LO_LIM := "Tag_LowLimit"
                         BIPOLAR := "Tag_Bipolar",
                         OUT => "Tag_Result");
```

The error information is returned in the operand "Tag_ErrorCode" as a function value.

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_InputValue	22
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1

Parameter	Operand	Value
OUT	Tag_Result	50.03978588
RET_VAL	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

Evaluating errors with output parameter RET_VAL (Page 5370)

Switching display formats in the program status (Page 2980)

GET_ERR_ID: Get error ID locally (Page 4282)

UNSCALE: Unscale

Description

The instruction "Unscale" unscales the floating-point number on the IN parameter in physical units between a low and a high limit and converts them into an integer. You can use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is unscaled. The result of the instruction is output on the OUT parameter.

The instruction "Unscale" works with the following equation:

OUT = [((IN–LO_LIM)/(HI_LIM–LO_LIM)) * (K2–K1)] + K1

The values of the "K1" and "K2" constants are determined by the signal state on the BIPOLAR parameter. The following signal states are possible on the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case the "K1" constant has the value "-27648.0" and the "K2" constant the value "+27648.0".
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case the "K1" constant has the value "0.0" and the "K2" constant the value "+27648.0".

When the value at the IN parameter is greater than the value of the "HI_LIM" constant, the result of the instruction is set to the value of the constant (K2) and an error is output.

When the value at the IN parameter is less than the value of the constant of the low limit (LO_LIM), the result of the instruction is set to the value of the constant (K1) and an error is output.

Syntax

The following syntax is used for the instruction "Unscale":

```
SCL
UNSCALE(IN := <Expression>,
        HI_LIM := <Operand>,
        LO_LIM := <Operand>,
        BIPOLAR := <Operand>,
        OUT => <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
IN	Input	REAL	Input value to be unscaled to an integer value.
HI_LIM	Input	REAL	High limit
LO_LIM	Input	REAL	Low limit
BIPOLAR	Input	BOOL	Indicates if the value at the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
OUT	Output	INT	Result of the instruction
Function value (RET_VAL)		WORD	Error information

For additional information on valid data types, refer to "See also".

RET_VAL Parameter

The following table shows the meaning of the values of the parameter RET_VAL:

Error code (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the high limit (HI_LIM) or less than the value of the low limit (LO_LIM).
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

```
SCL
"Tag_ErrorCode" := UNSCALE(IN := "Tag_InputValue",
                           HI_LIM := "Tag_HighLimit"
                           LO_LIM := "Tag_LowLimit"
                           BIPOLAR := "Tag_Bipolar",
                           OUT => "Tag_Result");
```

The error information is returned in the operand "Tag_ErrorCode" as a function value.

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
OUT	Tag_Result	22
RET_VAL	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

Evaluating errors with output parameter RET_VAL (Page 5370)

Switching display formats in the program status (Page 2980)

GET_ERR_ID: Get error ID locally (Page 4282)

Program control operations

IF: Run conditionally

Description

The instruction "Run conditionally" branches the program flow depending on a condition. The condition is an expression with Boolean value (TRUE or FALSE). Logical expression or comparative expressions can be stated as conditions.

When the instruction is executed, the stated expressions are evaluated. If the value of an expression is TRUE, the condition is fulfilled; if the value is FALSE, it is not fulfilled.

Syntax

Depending on the type of branch, you can program the following forms of the instruction:

- Branch through IF:

SCL

```
IF <Condition> THEN <Instructions>
END_IF;
```

If the condition is satisfied, the instructions programmed after the THEN are executed. If the condition is not satisfied, the execution of the program continues with the next instruction after the END_IF.

- Branch through IF and ELSE:

SCL

```
IF <Condition> THEN <Instructions1>
ELSE <Instructions0>;
END_IF;
```

If the condition is satisfied, the instructions programmed after the THEN are executed. If the condition is not satisfied, the instructions programmed after the ELSE are executed. Then the execution of the program continues with the next instruction after the END_IF.

- Branch through IF, ELSIF and ELSE:

SCL

```
IF <Condition1> THEN <Instructions1>
ELSIF <Condition2> THEN <Instruction2>
ELSE <Instructions0>;
END_IF;
```

If the first condition (<Condition1>) is satisfied, the instructions (<Instructions1>) after the THEN are executed. After execution of the instructions, the execution of the program continues after the END_IF.

If the first condition is not satisfied, the second condition (<Condition2>) is checked. If the second condition (<Condition2>) is fulfilled, the instructions (<Instructions2>) after the THEN are executed. After execution of the instructions, the execution of the program continues after the END_IF.

If none of the conditions are fulfilled, the instructions (<Instructions0>) after ELSE are executed followed by the execution of the program after END_IF.

You can nest as many combinations of ELSIF and THEN as you like within the IF instruction. The programming of an ELSE branch is optional.

The syntax of the IF instruction consists of the following parts:

Parameter	Data type	Description
<Condition>	BOOL	Expression to be evaluated
<Instructions>	-	Instructions to be executed with satisfied condition. An exception are instructions programmed after the ELSE. These are executed if no condition within the program loop is satisfied.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
IF "Tag_1" = 1
THEN "Tag_Value" := 10;
ELSIF "Tag_2" = 1
THEN "Tag_Value" := 20;
ELSIF "Tag_3" = 1
THEN "Tag_Value" := 30;
ELSE "Tag_Value" := 0;
END_IF;
```

The following table shows how the instruction works using specific operand values:

Operand	Value			
Tag_1	1	0	0	0
Tag_2	0	1	0	0
Tag_3	0	0	1	0
Tag_Value	10	20	30	0

See also

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Overview of the valid data types (Page 2236)

CASE: Create multiway branch

Description

The instruction "Create multiway branch" executes one of several instruction sequences depending on the value of a numerical expression.

The value of the expression must be an integer. When the instruction is executed, the value of the expression is compared with the values of several constants. If the value of the

expression agrees with the value of a constant, the instructions programmed directly after this constant are executed. The constants can assume the following values:

- An integer (for example, 5)
- A range of integers (for example, 15..20)
- An enumeration consisting of integers and ranges (for example, 10, 11, 15..20)

Syntax

The following syntax is used for the "Create multiway branch" instruction:

SCL

```
CASE <expression> OF
<Constant1>: <Instructions1>
<Constant2>: <Instructions2>
<ConstantX>: <InstructionsX>; // X >=3
ELSE <Instructions0>;
END_CASE;
```

The syntax of the instruction consists of the following parts:

Parameter	Data type	Description
<expression>	Integers	Value which is compared to the programmed constant values.
<Constant>	Integers	Constant values which form the condition for the execution of an instruction sequence. The constants can assume the following values: • An integer (for example, 5) • A range of integers (for example, 15..20) • An enumeration consisting of integers and ranges (for example, 10, 11, 15..20)
<Instruction>	-	Any instructions which are executed if the value of the expression agrees with the value of a constant. An exception are instructions programmed after the ELSE. These instructions are executed if the values do not agree.

For additional information on valid data types, refer to "See also".

If the value of the expression agrees with the value of the first constant (<Constant1>), the instructions (<Instructions1>) which are programmed directly after the first constant are executed. Program execution subsequently resumes after the END_CASE.

If the value of the expression does not agree with the value of the first constant (<Constant1>), this value is compared to the value of the constant which is programmed next. In this way, the CASE instruction is executed until the values agree. If the value of the expression does not correspond to any of the programmed constant values, the instructions (<Instructions0>) which are programmed after the ELSE are executed. ELSE is an optional part of the syntax and can be omitted.

The CASE instruction can also be nested by replacing an instruction block with CASE. END_CASE represents the end of the CASE instruction.

Example

The following example shows how the instruction works:

```
SCL
CASE "Tag_Value" OF
     0 :
     "Tag_1" := 1;
     1,3,5 :
     "Tag_2" :=1;
     6..10 :
     "Tag_3" := 1;
     16,17,20..25 :
     "Tag_4" := 1;
ELSE "Tag_5" := 1;
END_CASE;
```

The following table shows how the instruction works using specific operand values:

Operand	Values				
Tag_Value	0	1, 3 , 5	6, 7, 8, 9, 10	16,17, 20, 21, 22, 23, 24, 25	2
Tag_1	1	-	-	-	-
Tag_2	-	1	-	-	-
Tag_3	-	-	1	-	-
Tag_4	-	-	-	1	-
Tag_5	-	-	-	-	1
1: The operand is set to the signal state "1". -: The signal state of the operand remains unaltered.					

See also

CONTINUE: Recheck loop condition (Page 4267)

Overview of the valid data types (Page 2236)

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

EXIT: Exit loop immediately (Page 4268)

FOR: Run in counting loop

Description

The instruction "Run in counting loop" causes repeated execution of a program loop until a loop variable lies within a specified value range.

Program loops can also be nested. Within a program loop, you can program additional program loops with other loop variables.

The current continuous run of a program loop can be ended by the instruction "Recheck loop condition" (CONTINUE). The instruction "Exit loop immediately" (EXIT) ends the entire loop execution. For additional information on this topic, refer to "See also".

Limits for FOR statements

To program "safe" FOR statements that do not run endlessly, observe the following rules and limits:

Rule

```
FOR ii := Start TO End BY Step DO
```

If ...	... then	Note
start < end	end < (PMAX step)	Run tag ii runs in a positive direction
start > end AND step < O	end > (NMAX step)	Run tag ii runs in a negative direction

Limits

Different limits apply to the possible data types:

Data type	PMAX	NMAX
ii of type SINT	127	-128
ii of type INT	32767	-32768
ii of type DINT	2147483647	-2147483648
ii of type LINT	9223372036854775807	-9223372036854775808

Syntax

The following syntax is used for the "Run in counting loop" instruction:

SCL

```
FOR<Run_tag> := <Start_value> TO<End_value> BY<Increment> DO<Instructions>
END_FOR;
```

Parameter

The following table shows the parameters of the "Run in counting loop" instruction:

Parameter	Data type		Description
	S7-1200	S7-1500	
<Run tag>	SINT, INT, DINT	SINT, INT, DINT, LINT	Operand whose value is evaluated with the loop execution. The data type of the loop variable determines the data type of the other parameters.
<Start value>	SINT, INT, DINT	SINT, INT, DINT, LINT	Expression whose value is allocated at the start of the loop execution of the loop variables.
<End value>	SINT, INT, DINT	SINT, INT, DINT, LINT	Expression whose value defines the last run of the program loop. The value of the loop variable is checked before each loop: • End value not reached: The instructions according to DO are executed • End value is reached: The FOR loop is executed one last time • End value exceeded: The FOR loop is completed An alteration to the end value is not permitted during execution of the instruction.
<Increment>	SINT, INT, DINT	SINT, INT, DINT, LINT	Expression by whose value the loop variable is increased (positive increment) or decreased (negative increment) after each loop. Specification of the increment is optional. If no increment is given, the value of the loop variable is increased by 1 after each loop. An alteration of the increment is not permitted during execution of the instruction.
<Instructions>	-	-	Instructions which are carried out with each loop, as long as the value of the loop variable lies within the value range. The value range is defined by the start and end values.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
FOR i := 2 TO 8 BY 2
    DO "a_array[i] := "Tag_Value"*"b_array[i]";
END_FOR;
```

The operand "Tag_Value" is multiplied with the elements (2, 4, 6, 8) of the ARRAY tag "b_array". The result is read in to the elements (2, 4, 6, 8) of the ARRAY tag "a_array".

See also

CONTINUE: Recheck loop condition (Page 4267)

EXIT: Exit loop immediately (Page 4268)

Overview of the valid data types (Page 2236)

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

WHILE: Run if condition is met

Description

The instruction "Run if condition is met" causes a program loop to be repeatedly executed until the implementation condition is satisfied. The condition is an expression with Boolean value (TRUE or FALSE). Logical expression or comparative expressions can be stated as conditions.

When the instruction is executed, the stated expressions are evaluated. If the value of an expression is TRUE, the condition is fulfilled; if the value is FALSE, it is not fulfilled.

Program loops can also be nested. Within a program loop, you can program additional program loops with other loop variables.

The current continuous run of a program loop can be ended by the instruction "Recheck loop condition" (CONTINUE). The instruction "Exit loop immediately" (EXIT) ends the entire loop execution. For additional information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Run if condition is met" instruction:

SCL

```
WHILE <Condition> DO <Instructions>
END_WHILE;
```

The syntax of the WHILE instruction consists of the following parts:

Parameter	Data type	Description
<Condition>	BOOL	Expression which is evaluated before each loop.
<Instructions>	-	Instructions to be executed with satisfied condition. If the condition has not been satisfied, program execution continues after END_WHILE.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
WHILE
      "Tag_Value1" <> "Tag_Value2"
      DO "Tag_Result"
      := "Tag_Input";
END_WHILE;
```

As long as the values of the operands "Tag_Value1" and "Tag_Value2" do not match, the value of the operand "Tag_Input" is allocated to the operand "Tag_Result".

See also

EXIT: Exit loop immediately (Page 4268)

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

CONTINUE: Recheck loop condition (Page 4267)

Overview of the valid data types (Page 2236)

REPEAT: Run if condition is not met

Description

The instruction "Run if condition is not met" causes a program loop to be repeatedly executed until a termination condition is met. The condition is an expression with Boolean value (TRUE or FALSE). Logical expression or comparative expressions can be stated as conditions.

When the instruction is executed, the stated expressions are evaluated. If the value of an expression is TRUE, the condition is fulfilled; if the value is FALSE, it is not fulfilled.

The instructions are executed once, even if the termination condition is fulfilled.

Program loops can also be nested. Within a program loop, you can program additional program loops with other loop variables.

The current continuous run of a program loop can be ended by the instruction "Recheck loop condition" (CONTINUE). The instruction "Exit loop immediately" (EXIT) ends the entire loop execution. For additional information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Run if condition is not met" instruction:

```
SCL
REPEAT <Instructions>
```

```
SCL
UNTIL <Condition>END_REPEAT;
```

The syntax of the REPEAT instruction consists of the following parts:

Parameter	Data type	Description
<Instructions>	-	Instructions that are executed as long as the programmed condition has the value FALSE. The instructions are executed once, even if the termination condition is fulfilled.
<Condition>	BOOL	Expression which is evaluated after each loop. If the expression has the value FALSE, the program loop is executed once again. If the expression has the value TRUE, the program loop continues after END_REPEAT.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
REPEAT "Tag_Result"
        := "Tag_Value";
UNTIL "Tag_Error"
END_REPEAT;
```

As long as the value of the operand "Tag_Error" has the signal state "0", the value of the operand "Tag_Value" is allocated to the operand "Tag_Result".

See also

CONTINUE: Recheck loop condition (Page 4267)

EXIT: Exit loop immediately (Page 4268)

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Overview of the valid data types (Page 2236)

CONTINUE: Recheck loop condition

Description

The "Recheck loop condition" instruction ends the current program run of a FOR, WHILE or REPEAT loop.

After execution of the instruction, the conditions for the continuation of the program loop are evaluated again. The instruction affects the program loop which directly contains the instruction.

Syntax

The following syntax is used for the "Recheck loop condition" instruction:

SCL

```
CONTINUE;
```

Example

The following example shows how the instruction works:

SCL

```
FOR i
     := 1 TO 15 BY 2 DO
     IF (i < 5) THEN
     CONTINUE;
END_IF;
      "DB10".Test[i] := 1;
END_FOR;
```

For additional information on valid data types, refer to "See also".

If the condition i < 5 is satisfied, then the subsequent value allocation ("DB10".Test[i] := 1) will not be executed. The run variable (i) is increased by the increment of "2" and checked to see whether its current value lies in the programmed value range. If the run variable lies in the value range, the IF condition is evaluated again.

If the condition i < 5 is not satisfied, then the subsequent value allocation ("DB10".Test[i] := 1) will be executed and a new loop started. In this case, the run variable is also increased by the increment "2" and checked.

See also

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

EXIT: Exit loop immediately (Page 4268)

Overview of the valid data types (Page 2236)

EXIT: Exit loop immediately

Description

The instruction "Exit loop immediately" cancels the execution of a FOR, WHILE or REPEAT loop at any point regardless of conditions. The execution of the program is continued after the end of the loop (END_FOR, END_WHILE, END_REPEAT).

The instruction affects the program loop which directly contains the instruction.

Syntax

The following syntax is used for the "Exit loop immediately" instruction:

```
SCL
EXIT;
```

Example

The following example shows how the instruction works:

```
SCL
FOR i := 15 TO 1 BY -2 DO
IF (i < 5)
THEN EXIT;
END_IF;
"DB10".Test[i] := 1;
END_FOR;
```

For additional information on valid data types, refer to "See also".

If the condition i < 5 is satisfied, then the execution of the loop will be cancelled. Program execution resumes after the END_FOR.

If the condition i < 5 is not satisfied, then the subsequent value allocation ("DB10".Test[i] := 1) will be executed and a new loop started. The run tag (i) is decreased by the increment of "-2" and it is checked whether its current value lies in the programmed value range. If the (i) run variable lies within the value range, the IF condition is evaluated again.

See also

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

CONTINUE: Recheck loop condition (Page 4267)

Overview of the valid data types (Page 2236)

GOTO: Jump

Description

Use the instruction "Jump" to resume the execution of a program at a given point marked with a jump label.

The jump labels and the instruction "Jump" must be in the same block. The name of a jump label can only be assigned once within a block. Each jump label can be the target of several jump instructions.

A jump from the "outside" into a program loop is not permitted, but a jump from a loop to the "outside" is possible.

Syntax

Use the following syntax for the "Jump" instruction:

SCL

```
GOTO <Jump label>
...
<Jump label>: <Instructions>
```

The syntax of the GOTO instruction consists of the following parts:

Parameter	Data type	Description
<jump label>	-	Jump label to be jumped to
<Instructions>	-	Instructions which are executed after the jump.

Example

The following example shows how the instruction works:

SCL

```
CASE "Tag_Value" OF
1 : GOTO MyLABEL1;
2 : GOTO MyLABEL2;
3 : GOTO MyLABEL3;
ELSE GOTO MyLABEL4;
END_CASE;
MyLABEL1: "Tag_1" := 1;
MyLABEL2: "Tag_2" := 1;
MyLABEL3: "Tag_3" := 1;
MyLABEL4: "Tag_4" := 1;
```

Depending on the value of the "Tag_Value" operand, the execution of the program will resume at the point identified by the corresponding jump label. If the operand "Tag_Value" has the value 2, for example, program execution will resume at the jump label "MyLABEL2". The program line identified by the jump label "MyLABEL1" will be skipped in this case.

See also

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Overview of the valid data types (Page 2236)

RETURN: Exit block

Description

The instruction "Exit block" exits the program execution in the currently edited block and continues in the calling block.

The instruction can be omitted at the end of the block.

Syntax

The following syntax is used for the "Exit block" instruction:

SCL

```
RETURN;
```

Example

The following example shows how the instruction works:

SCL

```
IF "Tag_Error" <>0 THEN RETURN;
END_IF;
```

If the signal state of the "Tag_Error" operand is not zero, execution of the program ends in the block currently being processed.

See also

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Overview of the valid data types (Page 2236)

(*...*): Inserting a comment section

Description

You can use the "Insert comment section" instruction to add a comment section. The text within the parenthesis "(*...*)" is handled as a comment.

Syntax

The following syntax is used for the "Insert comment section" instruction:

```
SCL
(*...*)
```

Example

The following example shows how the instruction works:

```
SCL
(*This is a comment section.*)
```

See also

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Overview of the valid data types (Page 2236)

Runtime control

ENDIS_PW: Limit and enable password legitimation

Description

You use the "Limit and enable password legitimation" instruction to specify whether passwords configured for the CPU are legitimized or not. In this way, you can prevent legitimized connections, even if the correct password is known.

When you call the instruction and the REQ parameter has the signal state "0", the currently set state is displayed at the output parameters. If changes have been made to the input parameters, these are not transferred to the output parameters.

When you call the instruction and the REQ parameter has the signal state "1", the signal state is taken from the input parameters (F_PWD, FULL_PWD, R_PWD, HMI_PWD). FALSE means that legitimation per password is not permitted; TRUE means the password can be used.

Disable or enabling of the passwords can be allowed or prohibited individually. For example, all passwords can be prohibited except the fail-safe password. You can thus limit access options to a small user group. The output parameters (F_PWD_ON, FULL_PWD_ON, R_PWD_ON, HMI_PWD_ON) always show the current status of the password use, regardless of the REQ parameter.

Passwords that are not configured must have the signal state TRUE at the input and return the signal state TRUE at the output. The fail-safe password can only be configured for an F-CPU and must therefore always be interconnected with the signal state TRUE in a standard

CPU. If the instruction returns an error, the call has no effect and the previous lock is still in effect.

Disabled passwords can be enabled again under the following conditions:

- The CPU is reset to its factory settings.
- The front panel of the S7-1500 CPU supports a dialog that allows you to navigate to the appropriate menu in which the passwords can be enabled again.
- When you call the "Limit and enable password legitimation" instruction, the input parameter of the desired password has the signal state "1".
- Set the mode selector to STOP. The restriction to password legitimation is re-established as soon as the switch is set back to RUN.
- Plugging an empty memory card (transfer module or program card) into an S7-1200 CPU.
- The transition from POWER OFF to POWER ON disables the protection in the S7-1200 CPU. The "Limit and enable password legitimation" instruction must be called once again in the program (e.g. in the startup OB).

Note

The "Limit and enable password legitimation" instruction blocks access from HMI systems if the HMI password has not been enabled.

Note

Existing legitimized connections retain their access rights and you cannot use the "Limit and enable password legitimation" instruction to restrict them.

Preventing accidental lockout of an S7-1500 CPU

The settings can be made on the front panel of the CPU and the CPU saves the most recent setting.

To prevent accidental lockout, the protection can be disabled by setting the mode switch to STOP with an S7-1500 CPU. The protection is automatically enabled again by setting mode selector to RUN without having to call the "Limit and enable password legitimation" instruction again or taking additional actions on the front panel.

Preventing accidental lockout of an S7-1200 CPU

Because the S7-1200 CPU does not have a mode switch, protection is disabled with POWER OFF - POWER ON. This means that it is possible and advisable to prevent accidental lockout with the help of specific program sequences within your program.

To do so, program a time control either by means of a cyclic interrupt OB or a timer in the Main OB (OB 1). This gives you the option of calling the "Limit and enable password legitimation" instruction in the respective OB (e.g., OB 1 or OB 35) relatively quickly after a transition from POWER OFF to POWER ON and the associated disabling of the protection. Call the instruction in the startup OB (OB 100) to keep the time window in which the instruction is inactive and

there are no limitations for password legitimation relatively small. This procedure will give you the best possible protection from unauthorized access.

If there has been an accidental lockout, you can skip the call in the startup OB (by querying an input parameter, for example) and you have the set time (for example, 10 seconds up to 1 minute) to establish a connection to the CPU before the lock becomes active once again.

If there is no timer in your program code and a lockout has occurred, insert an empty transfer card or an empty program card into the CPU. The empty transfer card or program card deletes the internal load memory of the CPU. You must then download the user program once again from STEP 7 to the CPU.

Procedure for lost passwords with an S7-1200 CPU

If you have lost the password for a password-protected S7-1200 CPU, delete the password-protected program with an empty transfer card or program card. The empty transfer card or program card deletes the internal load memory of the CPU. You can then load a new user program from STEP 7 Basic in the CPU.

WARNING

Inserting an empty transfer card

When you insert a transfer card in a CPU during runtime, the CPU goes to STOP mode. If operating states are unstable, controllers may fail and thus cause the uncontrolled operation of the controlled devices. This leads to an unforeseen operation of the automation system which can cause deadly or serious injuries and/or damage to property.

Once you have pulled the transfer card, the content of the transfer card is available in the internal load memory. Make sure that the card does not include a program at this point.

WARNING

Inserting an empty program card

When you insert a program card in a CPU during runtime, the CPU goes to STOP mode. If operating states are unstable, controllers may fail and thus cause the uncontrolled operation of the controlled devices. This leads to an unforeseen operation of the automation system which can cause deadly or serious injuries and/or damage to property.

Make sure that the program card is empty. The internal load memory is copied to the empty program card. Once you have pulled the previously empty program card, the internal load memory is empty.

You must remove the transfer card or program card before you put the CPU into RUN mode.

Effects of password use on the operating modes

The following table shows what effects the password use by means of the "Limit and enable password legitimation" instruction has on the operating modes and the corresponding user actions.

Action	Password protection by means of the instruction
Basic state after • Operating mode switch to STOP • Reset memory manually (PG, switch, change of MC (Motion Control)) • Reset to factory setting	Not activated (no restrictions)
Basic state after POWER ON	• S7-1200 CPU: The lock is disabled and the instruction must be called once again in the program (e.g., in the startup OB). • S7-1500 CPU: Enabled (if a lock was activated before POWER OFF). The option of not allowing passwords is retentive.
Operating mode transition RUN/STARTUP/HOLD -> STOP (by terminating the instruction, an error or communication) or STOP -> STARTUP/RUN/HOLD	Activated Passwords still cannot be used.

Syntax

The following syntax is used for the "Limit and enable password legitimation" instruction:

SCL

```
ENDIS_PW(REQ := <Operand>,
         F_PWD := <Operand>,
         FULL_PWD := <Operand>,
         R_PWD := <Operand>,
         HMI_PWD := <Operand>,
         F_PWD_ON => <Operand>,
         FULL_PWD_ON => <Operand>,
         R_PWD_ON => <Operand>,
         HMI_PWD_ON => <Operand>,
         RET_VAL => <Operand>)
```

Parameter

The following table shows the parameters of the "Limit and enable password legitimation" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	When the REQ parameter has the signal state "0", the currently set signal state of the passwords is queried.
F_PWD	Input	BOOL	I, Q, M, D, L	Read/write access including fail-safe • F_PWD = "0": Do not allow password • F_PWD = "1": Allow password
FULL_PWD	Input	BOOL	I, Q, M, D, L	Read/write access • FULL_PWD = "0": Do not allow password • FULL_PWD = "1": Allow password
R_PWD	Input	BOOL	I, Q, M, D, L	Read access • R_PWD = "0": Do not allow password • R_PWD = "1": Allow password
HMI_PWD	Input	BOOL	I, Q, M, D, L	HMI access • HMI_PWD = "0": Do not allow password • HMI_PWD = "1": Allow password
F_PWD_ON	Output	BOOL	I, Q, M, D, L	Read/write access including fail-safe • F_PWD_ON = "0": Password not allowed • F_PWD_ON = "1": Password allowed
FULL_PWD_ON	Output	BOOL	I, Q, M, D, L	Read/write access status • FULL_PWD_ON = "0": Password not allowed • FULL_PWD_ON = "1": Password allowed
R_PWD_ON	Output	BOOL	I, Q, M, D, L	Read-access status • R_PWD_ON = "0": Password not allowed • R_PWD_ON = "1": Password allowed
HMI_PWD_ON	Output	BOOL	I, Q, M, D, L	HMI-access status • HMI_PWD_ON = "0": Password not allowed • HMI_PWD_ON = "1": Password allowed
RET_VAL	Output	WORD	I, Q, M, D, L	Error information

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8090	The "Limit and enable password legitimation" instruction is not supported

Error code* (W#16#...)	Explanation
80D0	The password for fail-safe is not configured. The signal state must be TRUE for standard CPUs.
80D1	The read/write access is not configured
80D2	The read access is not configured
80D3	The HMI access is not configured
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

RE_TRIGR: Restart cycle monitoring time

Description

The instruction "Restart cycle monitoring time" restarts the cycle monitoring of the CPU. The cycle time monitoring then restarts with the time you have set in the CPU configuration.

The instruction "Restart cycle monitoring time" can be called, regardless of the priority, in all blocks.

If the instruction is called in a block with a higher priority, such as a hardware interrupt, diagnostic interrupt, or cyclic interrupt, the instruction is not executed and the ENO enable output is set to signal state "0".

The instruction "Restart cycle monitoring time" can be called a maximum of 10 times in a program cycle.

Note

If the instruction "Restart cycle monitoring time" is started more than 30 times within a LOOP instruction, the CPU switches to STOP because of a runtime error.

Syntax

The following syntax is used for the "Restart cycle monitoring time" instruction:

```
SCL
RE_TRIGR()
```

See also

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

STP: Exit program

Description

The "Exit program" instruction is used to set the CPU to STOP mode and therefore to terminate the execution of the program. The effects of changing from RUN to STOP depend on the CPU configuration.

Syntax

The following syntax is used for the instruction "Exit program":

```
SCL
STP()
```

See also

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

GET_ERROR: Get error locally

Description

The "Get error locally" instruction is used to query the occurrence of errors within a block. This is usually to access error. If the system reports an error during the processing of the block, detailed information is generated for the first error to occur during the execution of the block since the last execution of the instruction.

The error information can only be saved in operands of the "ErrorStruct" system data type. The "ErrorStruct" system data type specifies the exact structure in which the information about the error is stored. Using additional instructions, you can evaluate this structure and program an appropriate response. If several errors occur in the block, the error information about the

next error to occur in the instruction is output only after the first error that occurred has been remedied.

Note

The <Operand> is only changed if error information is present. To set the operand back to "0" after handling the error, you have the following options:

- Declare the operand in the "Temp" section of the block interface.
- Reset the operand to "0" before calling the instruction.

You can find an example of how you can implement the instruction in combination with other troubleshooting options under "See also".

Note

The "Get error locally" instruction enables local error handling within a block. If "Get error locally" is inserted in the program code of a block, any predefined system reactions are ignored if errors occur.

Syntax

The following syntax is used for the "Get error locally" instruction:

SCL

```
GET_ERROR(<Operand>)
```

Parameter

The following table shows the parameters of the "Get error locally" instruction:

Parameter	Data type	Description
<Operand>	ErrorStruct	Information about errors that have occurred

Data type "ErrorStruct"

The following table shows the structure of the "ErrorStruct" data type:

Structure component	Data type	Description
ERROR_ID	WORD	Error ID
FLAGS	BYTE	Shows if an error occurred during a block call. 16#01: Error during a block call 16#00: No error during a block call

Structure component		Data type	Description
REACTION		BYTE	Default reaction: 0: Ignore (write error), 1: Continue with substitute value "0" (read error), 2: Skip instruction (system error)
CODE_ADDRESS		CREF	Information about the address and type of block
	BLOCK_TYPE	BYTE	Type of block where the error occurred: 1: OB 2: FC 3: FB
	CB_NUMBER	UINT	Number of the code block
	OFFSET	UDINT	Reference to the internal memory
MODE		BYTE	Information about the address of an operand
OPERAND_NUMBER		UINT	Operand number of the machine command
POINTER_NUMBER_LOCATION		UINT	(A) Internal pointer
SLOT_NUMBER_SCOPE		UINT	(B) Storage area in internal memory
DATA_ADDRESS		NREF	Information about the address of an operand
	AREA	BYTE	(C) Memory area: L: 16#40 – 4E, 86, 87, 8E, 8F, C0 – CE I: 16#81 Q: 16#82 M: 16#83 DB: 16#84, 85, 8A, 8B Range violations for a directly editable tag of data type DINT: 16#04
	DB_NUMBER	UINT	(D) Number of the data block
	OFFSET	UDINT	(E) Relative address of the operand

Structure component "ERROR_ID"

The following table shows the values that can be output at the structure component "ERROR_ID":

ID* (hexadecimal)	ID* (decimal)	Description
0	0	No error
2503	9475	Invalid pointer
2520	9504	Invalid STRING
2522	9506	Read error: Operand outside the valid range

ID* (hexadecimal)	ID* (decimal)	Description
2523	9507	Write error: Operand outside the valid range
2524	9508	Read error: Invalid operand
2525	9509	Write error: Invalid operand
2528	9512	Read error: Data alignment
2529	9513	Write error: Data alignment
252C	9516	Invalid pointer
2530	9520	Write error: Data block
2533	9523	Invalid pointer used
2538	9528	Access error: DB does not exist
2539	9529	Access error: Wrong DB used
253A	9530	Global data block does not exist
253C	9532	Faulty information or the function does not exist
253D	9533	System function does not exist
253E	9534	Faulty information or the function block does not exist
253F	9535	System block does not exist
2550	9552	Access error: DB does not exist
2551	9553	Access error: Wrong DB used
2575	9589	Error in the program nesting depth
2576	9590	Error in the local data distribution
2577	9591	The block property "Parameter passing via registers" is not selected.
25A0	9632	Internal error in TP
25A1	9633	Tag is write-protected
25A2	9634	Invalid numerical value of tag
2942	10562	Read error: Input
2943	10563	Write error: Output
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".		

Example

The following example shows how the instruction works:

SCL

```
LABEL: #TagOut := #Field[#index] * REAL#40.5;

GET_ERROR(#Error);

IF #Error.REACTION = 1 THEN
   GOTO LABEL;
   ;
END_IF;
```

An error occurred accessing the "#Field[#index]" tag. The "#TagOut" operand returns the signal state "1" despite the read/access error and the multiplication is performed with the value "0.0". If this error scenario occurs, we recommend that you program the "Get error locally" instruction after the multiplication to get the error. The error information supplied by the "Get error locally" instruction is evaluated with a comparison. If the "#Error.REACTION" structure component has the value "1", there is a read/access error and the program execution begins again at the jump label LABEL.

See also

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Switching display formats in the program status (Page 2980)

Querying and fixing errors in the program code (Page 258)

Handling program execution errors (Page 2821)

GET_ERR_ID: Get error ID locally

Description

The "Get error ID locally" instruction is used to query the occurrence of errors within a block. This is usually to access error. If the system reports during the block processing errors within the block execution since the last execution of the instruction, the instruction outputs the error ID of the first error that occurred.

The error ID can only be saved in operands of the WORD data type. If several errors occur in the block, the error ID of the next error to occur in the instruction is output only after the first error that occurred has been remedied.

Note

The <Operand> is only changed if error information is present. To set the operand back to "0" after handling the error, you have the following options:

- Declare the operand in the "Temp" section of the block interface.
- Reset the operand to "0" before calling the instruction.

The output of the "Get error ID locally" instruction is only set if error information is present. If one of these conditions is not fulfilled, the remaining program execution is not affected by the "Get error ID locally" instruction.

You can find an example of how you can implement the instruction in combination with other troubleshooting options under "See also".

Note

The "Get error ID locally" instruction enables local error handling within a block. If the "Get error ID locally" instruction is inserted in the program code of a block, any predefined system reactions are ignored if errors occur.

Syntax

The following syntax is used for the "Get error ID locally" instruction:

SCL

```
GET_ERR_ID()
```

Parameter

The following table shows the parameters of the "Get error ID locally" instruction:

Parameter	Data type	Description
Function value	WORD	Error ID

Error ID

The following table shows the values that can be output:

ID* (hexadecimal)	ID* (decimal)	Description
0	0	No error
2503	9475	Invalid pointer
2520	9504	Invalid STRING
2522	9506	Read error: Operand outside the valid range
2523	9507	Write error: Operand outside the valid range
2524	9508	Read error: Invalid operand
2525	9509	Write error: Invalid operand
2528	9512	Read error: Data alignment
2529	9513	Write error: Data alignment
252C	9516	Invalid pointer
2530	9520	Write error: Data block
2533	9523	Invalid pointer used
2538	9528	Access error: DB does not exist
2539	9529	Access error: Wrong DB used
253A	9530	Global data block does not exist

ID* (hexadecimal)	ID* (decimal)	Description
253C	9532	Faulty information or the function does not exist
253D	9533	System function does not exist
253E	9534	Faulty information or the function block does not exist
253F	9535	System block does not exist
2550	9552	Access error: DB does not exist
2551	9553	Access error: Wrong DB used
2575	9589	Error in the program nesting depth
2576	9590	Error in the local data distribution
2577	9591	The block property "Parameter passing via registers" is not selected.
25A0	9632	Internal error in TP
25A1	9633	Tag is write-protected
25A2	9634	Invalid numerical value of tag
2942	10562	Read error: Input
2943	10563	Write error: Output
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".		

Example

The following example shows how the instruction works:

SCL

```
#TagOut := #Field[#index] * REAL#40.5;

#TagID := GET_ERR_ID();

IF #TagID = 16#2522 THEN
   MOVE_BLK(IN := #TagArrayIn[0],
            COUNT := 1,
            OUT => #TagArrayOut[1]);
END_IF;
```

An error occurred accessing the "#Field[#index]" tag. The "#TagOut" operand returns the signal state "1" despite the read/access error and the multiplication is performed with the value "0.0". If this error scenario occurs, we recommend that you program the "Get error ID locally" instruction after the multiplication to get the error. The error ID supplied by the "Get error ID locally" instruction is evaluated with a comparison. If the "#TagID" operand returns ID 16#2522, there is a read/access error and the value "100.0" of the "#TagArrayIn[0]" operand is written to the "#TagArrayOut[1]" operand.

See also

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Switching display formats in the program status (Page 2980)
Querying and fixing errors in the program code (Page 258)
Handling program execution errors (Page 2821)

INIT_RD: Initialize all retain data

Description

The "Initialize all retain data" instruction is used to reset the retentive data of all data blocks, bit memories and SIMATIC timers and counters at the same time. The instruction can only be executed within a startup OB because the execution exceeds the program cycle duration.

Syntax

Use the following syntax for the "Initialize all retain data" instruction:

SCL

```
INIT_RD(REQ := <Operand>
        RET_VAL := <Operand>)
```

Parameter

The following table shows the parameters of the "Initialize all retain data" instruction:

Parameter	Declaration	Data type	Description
REQ	Input	BOOL	If the input "REQ" has the signal state "1", all retentive data are reset.
RET_VAL	Output	INT	Error information: If an error occurs during the execution of the instruction, an error code is output at the RET_VAL parameter.

For additional information on valid data types, refer to "See also".

RET_VAL Parameter

The following table shows the meaning of the values of the parameter RET_VAL:

Error code* (W#16#...)	Explanation
0000	No error
80B5	The instruction cannot be executed because it was not programmed within a startup OB.

Error code* (W#16#...)	Explanation
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

SCL

```
INIT_RD(REQ := "Tag_REQ",
        RET_VAL := "Tag_Result");
```

If the operand "Tag_REQ" has the signal state "1", the instruction is executed. All retentive data of all data blocks, bit memories and SIMATIC timers and counters are reset.

See also

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Evaluating errors with output parameter RET_VAL (Page 5370)

Overview of the valid data types (Page 2236)

Switching display formats in the program status (Page 2980)

GET_ERR_ID: Get error ID locally (Page 4282)

WAIT: Configure time delay

Description

The instruction "Configure time delay" pauses the program execution for a specific period of time. You indicate the period of time in microseconds on the WT parameter of the instruction.

You can configure time delays of up to 32 767 microseconds (µs). The shortest possible time delay depends on the respective CPU and corresponds to the execution time of the instruction "Configure time delay".

The execution of the instruction can be interrupted by higher priority events.

The instruction "Configure time delay" supplies no error information.

Syntax

The following syntax is used for the instruction "Configure time delay":

```
SCL
WAIT(WT:= <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
WT	Input	INT	Time delay in microseconds (µs)

See also

Evaluating errors with output parameter RET_VAL (Page 5370)

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

RUNTIME: Measure program runtime

Description

The "Measure program runtime" instruction is used to measure the runtime of the entire program, individual blocks or command sequences.

If you want to measure the runtime of your entire program, call the instruction "Measure program runtime" in OB1. Measurement of the runtime is started with the first call and the output RET_VAL returns the runtime of the program after the second call. The measured runtime includes all CPU processes that can occur during the program execution, for example, interruptions caused by higher-level events or communication. The instruction "Measure program runtime" reads an internal counter of the CPU and writes the value to the in/out parameter. The instruction calculates the current program runtime according to the internal counter frequency and writes it to output RET_VAL.

If you want to measure the runtime of individual blocks or individual command sequences, you need three separate networks. Call the instruction "Measure program runtime" in an individual network within your program. You set the starting point of the runtime measurement with this first call of the instruction. Then you call the required program block or the command sequence in the next network. In another network, call the "Measure program runtime" instruction a second time and assign the same memory to the in/out parameter as you did during the first call of the instruction. The "Measure program runtime" instruction in the third network reads an internal CPU counter and calculates the current runtime of the program block or the command sequence according to the internal counter frequency and writes it to the output RET_VAL.

The "Measure program runtime" instruction uses an internal high-frequency counter to calculate the time. If the counter overruns, the instruction returns values <= 0.0. Ignore these runtime values.

Note

The runtime of a command sequence cannot be determined exactly, because the sequence of instructions within a command sequence is changed during optimized compilation of the program.

Syntax

The following syntax is used for the "Measure program runtime" instruction:

SCL

```
RUNTIME(<Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
<Operand>	InOut	LREAL	Saves the starting point of the runtime measurement.
Function value		LREAL	Returns the measured runtime in seconds

For additional information on valid data types, refer to "See also".

Example

The following example shows the how the instruction works based on the runtime calculation of a program block:

SCL

```
"Tag_Result" := RUNTIME("Tag_Memory");

"Best_before_date_DB" ();

"Tag_Result" := RUNTIME("Tag_Memory");
```

The starting point for the runtime measurement is set with the first call of the instruction and buffered as reference for the second call of the instruction in the "TagMemory" operand.

The "Best_before_date" program block FB1 is called.

When the program block FB1 has been processed, the instruction is executed a second time. The second call of the instruction calculates the runtime of the program block and writes the result to the output "Tag_Result".

See also

Entering SCL instructions (Page 2697)
Editing SCL instructions (Page 2716)
Evaluating errors with output parameter RET_VAL (Page 5370)
Overview of the valid data types (Page 2236)
IEC Timers (Page 4116)

Word logic operations

DECO: Decode

Description

The instruction "Decode" sets a bit specified by the input value in the output value.

The instruction "Decode" reads the value of the parameter IN and sets the bit in the output value, whose bit position corresponds to the read value. The other bits in the output value are filled with zeroes. If the value of the IN parameter is greater than 31, a modulo 32 instruction is executed.

Syntax

The following syntax is used for the instruction "Decode":

```
SCL
DECO(IN := <Expression>)
DECO_WORD(IN := <Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
IN	Input	UINT	Position of the bit in the output value which is set.
_<Data type>		Bit strings default: DWORD	Data type of the function value: • You do not need to specify the data type if using the default. • Any other valid data type you may use must be declared explicitly.
Function value		Bit strings	Current output value

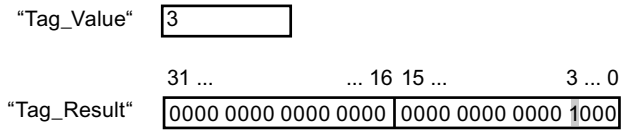
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := DECO(IN := "Tag_Value");
"Tag_Result2" := DECO_BYTE(IN := "Tag_Value2");
```

The following figure shows how the instruction works using specific operand values:



The instruction reads the number "3" from the value of the operand "Tag_Value" and sets the third bit to the value of the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

ENCO: Encode

Description

The instruction "Encode" reads the bit number of the lowest-value bit set in the input value and issues this as a result.

Syntax

The following syntax is used for the instruction "Encode":

```
SCL
ENCO(IN := <Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
IN	Input	Bit strings	Input value
Function value		INT	Bit number of the bit in the input value that is read out.

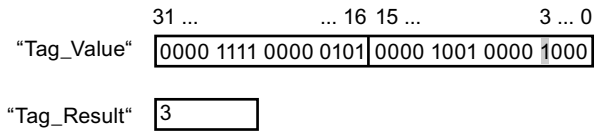
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := ENCO(IN := "Tag_Value");
```

The following figure shows how the instruction works using specific operand values:



The instruction reads the lowest-value set bit of the operand "Tag_Value" and writes the bit position "3" in the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

SEL: Select

Description

The instruction "Select" selects one of the parameters IN0 or IN1 depending on a switch (G parameter) and issues its content as a result. When the parameter G has the signal status "0", the value at parameter IN0 is moved. When the parameter G has the signal status "1", the value at parameter IN1 is moved and returned as a function value.

The instruction is only executed if the tags of all parameters have the same data type class.

Syntax

The following syntax is used for the instruction "Select":

SCL

```
SEL(G:= <Expression>,
    IN0 := <Expression>,
    IN1 := <Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type		
		S7-1200	S7-1500	Description
G	Input	BOOL	BOOL	Switch
IN0	Input	Bit strings, integers, floating-point numbers, TIME, TOD, DATE, CHAR, DT	Bit strings, integers, floating-point numbers, CHAR, timers, DATE, TOD, LTOD, DT, LDT	First input value
IN1	Input	Bit strings, integers, floating-point numbers, TIME, TOD, DATE, CHAR, DT	Bit strings, integers, floating-point numbers, CHAR, timers, DATE, TOD, LTOD, DT, LDT	Second input value
Function value		Bit strings, integers, floating-point numbers, TIME, TOD, DATE, CHAR, DT	Bit strings, integers, floating-point numbers, CHAR, timers, DATE, TOD, LTOD, DT, LDT	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := SEL(G := "Tag_Value",
                    IN0 := "Tag_0",
                    IN1 := "Tag_1");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_Value	0	1
Tag_0	W#16#0000	W#16#4C

Operand	Value	
Tag_1	W#16#FFFF	D#16#5E
Tag_Result	W#16#0000	D#16#5E

See also

Overview of the valid data types (Page 2236)

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

MUX: Multiplex

Description

The "Multiplex" instruction copies the value of a selected input parameter and issues it. You can use the parameter K to determine the number of the input parameter whose value will be moved. Numbering starts at IN0 and is incremented continuously with each new input. You can declare a maximum of 32 inputs.

If the value of the K parameter is greater than the number of inputs and the INELSE parameter is not set, the instruction function value is invalid and the ENO enable output is set to 0.

Numerical data types and time data types are permitted at the inputs. All tags with assigned parameters must be of the same data type.

The function value is invalid if any of the following conditions are met:

- The value of the parameter K is greater than the number of available inputs.
- Errors occurred during the execution of the instruction.

Syntax

The following syntax is used for the "Multiplex" instruction:

SCL

```
MUX(K := <Expression>,
    IN0 := <Expression>,
    IN1 := <Expression>,
    INELSE := <Expression>)
```

Parameter

The following table shows the parameters of the "Multiplex" instruction:

Parameter	Declaration	Data type		Description
		S7-1200	S7-1500	
K	Input	Integers	Integers	Specifies the parameter whose content is to be transferred. • If K = 0 => Parameter IN0 • If K = 1 => Parameter IN1, etc.
IN0	Input	Binary numbers, integers, floating-point numbers, strings, TOD, DATE, TIME, DT	Binary numbers, integers, floating-point numbers, strings, TOD, LTOD, DATE, timers, DT, LDT	First input value
IN1	Input	Binary numbers, integers, floating-point numbers, strings, TOD, DATE, TIME, DT	Binary numbers, integers, floating-point numbers, strings, TOD, LTOD, DATE, timers, DT, LDT	Second input value
INn	Input	Binary numbers, integers, floating-point numbers, strings, TOD, DATE, TIME, DT	Binary numbers, integers, floating-point numbers, strings, TOD, LTOD, DATE, timers, DT, LDT	Optional input values
INELSE	Input	Binary numbers, integers, floating-point numbers, strings, TOD, DATE, TIME, DT	Binary numbers, integers, floating-point numbers, strings, TOD, LTOD, DATE, timers, DT, LDT	Specifies the value to be copied when K <> n.
Function value		Binary numbers, integers, floating-point numbers, strings, TOD, DATE, TIME, DT	Binary numbers, integers, floating-point numbers, strings, TOD, LTOD, DATE, timers, DT, LDT	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := MUX(K := "Tag_Number",
                    IN0 := "Tag_1",
                    IN1 := "Tag_2",
                    INELSE := "Tag_3");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_Number	1	4
Tag_1	DW#16#00000000	DW#16#00000000
Tag_2	DW#16#003E4A7D	DW#16#003E4A7D
Tag_3	DW#16#FFFF0000	DW#16#FFFF0000
Tag_Result	DW#16#003E4A7D	DW#16#FFFF0000

See also

Overview of the valid data types (Page 2236)

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

DEMUX: Demultiplex

Description

The "Demultiplex" instruction transfers the value of the input parameter IN to a selected output parameter. The selection of the input parameter takes place independent of the parameter value K. The K parameter specifies the output parameter number to which the value of the input parameter IN is transferred. The other output parameters are not changed. Numbering starts at OUT0 and continues consecutively with each new output. You can declare a maximum of 32 output parameters.

If the value of the K parameter is greater than the number of output parameters, the value of the ENO enable output is set to "0" and the value is transferred to the input parameter IN and the output parameter OUTELSE.

The function value is invalid if any of the following conditions are met:

- The value of the K parameter is greater than the number of available outputs.
- Errors occurred during execution of the instruction.

Syntax

The following syntax is used for the instruction "Demultiplex":

```
SCL
DEMUX(K := <Expression>,
      IN := <Expression>,
      OUT0 := <Operand>,
      OUT1 := <Operand>,
      OUTELSE := <Operand>)
```

Parameter

The following table shows the parameters of the instruction "Demultiplex":

Parameter	Declaration	Data type		Description
		S7-1200	S7-1500	
K	Input	Integers	Integers	Specifies the output to which the input value (IN) will be copied. • If K = 0 => Parameter OUT0 • If K = 1 => Parameter OUT1, etc.
IN	Input	Binary numbers, integers, floating-point numbers, strings, TIME, TOD, DATE, DT	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, DT, LDT	Input value
OUT0	Output	Binary numbers, integers, floating-point numbers, strings, TIME, TOD, DATE, DT	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, DT, LDT	First output
OUT1	Output	Binary numbers, integers, floating-point numbers, strings, TIME, TOD, DATE, DT	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, DT, LDT	Second output

Parameter	Declaration	Data type		Description
		S7-1200	S7-1500	
OUTn	Output	Binary numbers, integers, floating-point numbers, strings, TIME, TOD, DATE, DT	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, DT, LDT	Optional outputs
OUTELSE	Output	Binary numbers, integers, floating-point numbers, strings, TIME, TOD, DATE, DT	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, DT, LDT	Output to which the value at input IN is copied if K > n.

For additional information on available data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
DEMUX(K := "Tag_Number",
      IN := "Tag_Value",
      OUT0 := "Tag_1",
      OUT1 := "Tag_2",
      OUTELSE := "Tag_3");
```

The following tables show how the instruction works using specific operand values:

Input values of the "Demultiplex" instruction before the network execution

Parameter	Operand	Values	
K	Tag_Number	2	4
IN	Tag_Value	DW#16#FFFFFFFF	DW#16#003E4A7D

Output values of the "Demultiplex" instruction after the network execution

Parameter	Operand	Values	
OUT0	Tag_1	Unchanged	Unchanged
OUT1	Tag_2	DW#16#FFFFFFFF	Unchanged
OUTELSE	Tag_3	Unchanged	DW#16#003E4A7D

See also

Overview of the valid data types (Page 2236)

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Shift and rotate

SHR: Shift right

Description

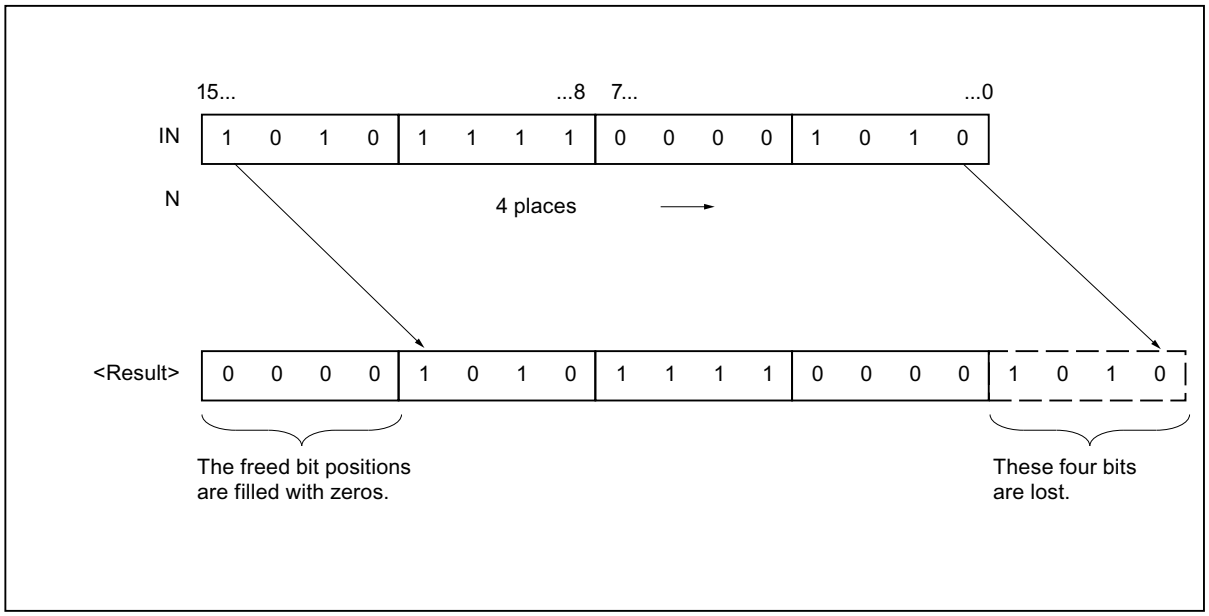
The "Shift right" instruction shifts the contents of the IN parameter bit-by-bit to the right and returns it as a function value. The parameter N is used to specify the number of bit positions by which the specified value should be shifted.

If the value of the N parameter is "0", the value of the IN parameter is given as a result.

If the value of the N parameter is greater than the number of available bit positions, then the value of the IN parameter is shifted to the right by the available number of bit positions.

The bit positions that are freed by shifting in the left operand area are filled with zeros.

The following figure shows how the content of an integer data type operand is shifted by four bit positions to the right:



Syntax

The following syntax is used for the instruction "Shift right":

SCL

```
SHR(IN := <Operand>,
    N := <Operand>)
```

Parameter

The following table shows the parameters of the "Shift right" instruction:

Parameter	Declaration	Data type		Description
		S7-1200	S7-1500	
IN	Input	Bit strings, integers	Bit strings, integers	Value to be shifted
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	Number of bits by which the value (IN) is shifted
Function value		Bit strings, integers	Bit strings, integers	Result of the instruction

For more information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := SHR(IN := "Tag_Value",
                    N := "Tag_Number");
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_Value	0011 1111 1010 1111
N	Tag_Number	3
Function value	Tag_Result	0000 0111 1111 010 1

The content of the "Tag_Value" operand is shifted by three bit positions to the right. The result of the instruction is returned in the operand "Tag_Result" as a function value.

See also

Overview of the valid data types (Page 2236)

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

SHL: Shift left

Description

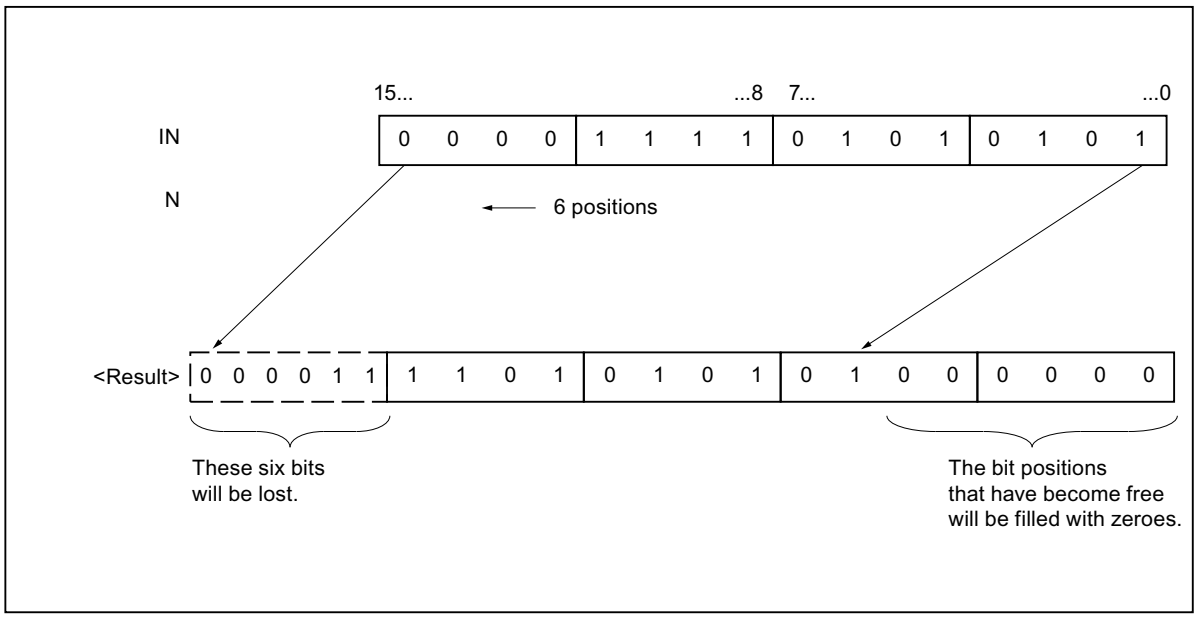
The "Shift left" instruction shifts the contents of the IN parameter bit-by-bit to the left and returns it as a function value. The parameter N is used to specify the number of bit positions by which the specified value should be shifted.

If the value of the N parameter is "0", the value of the IN parameter is given as a result.

If the value of the N parameter is greater than the number of bit positions, the value of the IN parameter is shifted to the left by the available number of bit positions.

The bit positions freed by the shift are filled with zeros in the result value.

The following figure shows how the content of an operand of the WORD data type is shifted six bit positions to the left:



Syntax

The following syntax is used for the instruction "Shift left":

SCL

```
SHL(IN := <Operand>,
    N := <Operand>)
```

Parameter

The following table shows the parameters of the "Shift left" instruction:

Parameter	Declaration	Data type		Description
		S7-1200	S7-1500	
IN	Input	Bit strings, integers	Bit strings, integers	Value to be shifted
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	Number of bits by which the value (IN) is shifted
Function value		Bit strings, integers	Bit strings, integers	Result of the instruction

For more information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := SHL(IN := "Tag_Value",
                    N := "Tag_Number");
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_Value	0011 1111 1010 1111
N	Tag_Number	4
Function value	Tag_Result	1111 1010 1111 0000

The value of the "Tag_Value" operand is shifted by four bit positions to the left. The result of the instruction is returned in the operand "Tag_Result" as a function value.

See also

Overview of the valid data types (Page 2236)

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

ROR: Rotate right

Description

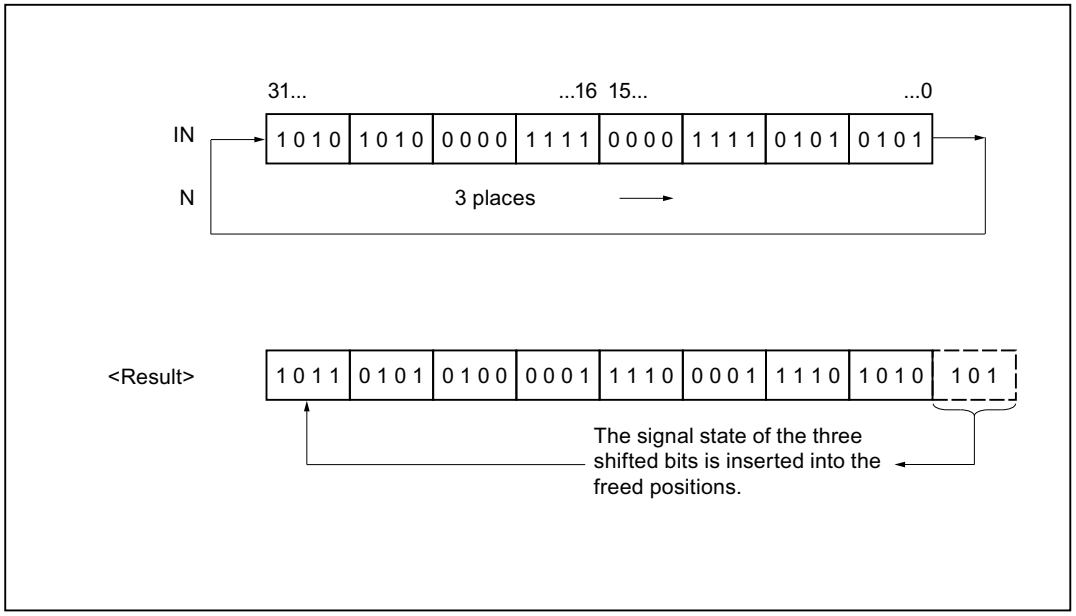
The "Rotate right" instruction rotates the content of the IN parameter bit-by-bit to the right and assigns the result to the specified operand. The parameter N is used to specify the number of

bit positions by which the specified value should be rotated. The bit positions freed by rotating are filled with the bit positions that are pushed out.

If the value of the N parameter is "0", the value at input IN is given as a result.

If the value at the N parameter is greater than the number of available bit positions, the operand value at the IN input is still rotated by the specified number of bit positions.

The following figure shows how the content of an operand of the DWORD data type is rotated three bit positions to the right:



Syntax

The following syntax is used for the instruction "Rotate right":

SCL

```
ROR(IN := <Operand>,
    N := <Operand>)
```

Parameter

The following table shows the parameters of the "Rotate right" instruction:

Parameter	Declaration	Data type		Description
		S7-1200	S7-1500	
IN	Input	Bit strings, integers	Bit strings, integers	Value to be rotated
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	Number of bit positions by which the value (IN) is rotated
Function value		Bit strings, integers	Bit strings, integers	Result of the instruction

For more information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := ROR(IN := "Tag_Value",
                    N := "Tag_Number");
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_Value	0000 1111 1001 0101
N	Tag_Number	5
Function value	Tag_Result	1010 1000 0111 1100

The content of the "Tag_Value" operand is rotated by five bit positions to the right. The result of the instruction is returned in the operand "Tag_Result" as a function value.

See also

Overview of the valid data types (Page 2236)

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

ROL: Rotate left

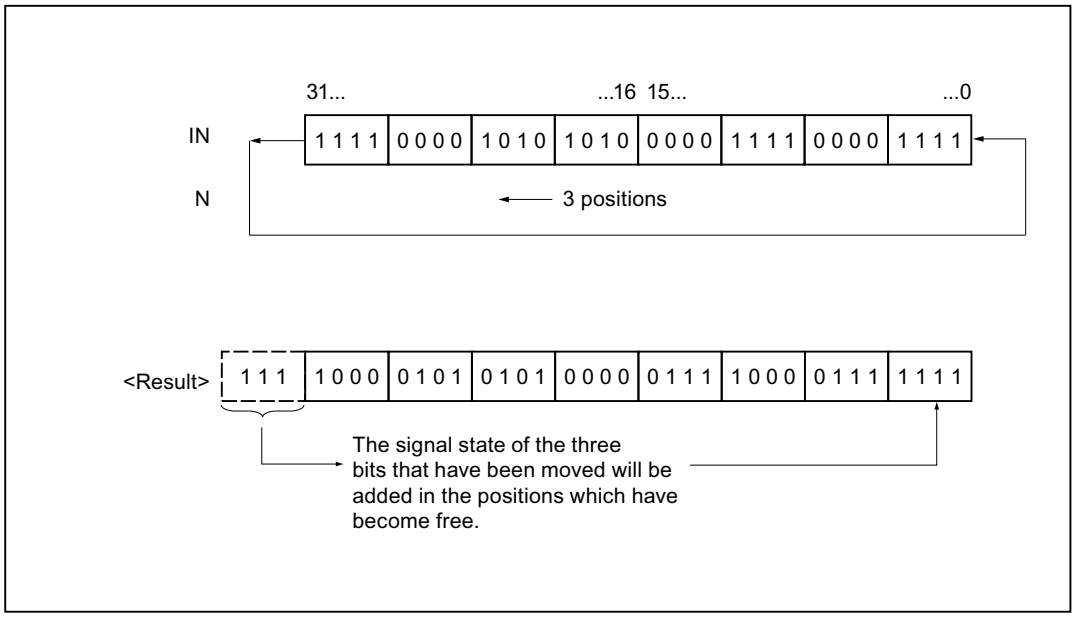
Description

The "Rotate left" instruction rotates the contents of the IN parameter bit-by-bit to the left and returns it as a function value. The parameter N is used to specify the number of bit positions by which the specified value should be rotated. The bit positions freed by rotating are filled with the bit positions that are pushed out.

If the value of the N parameter is "0", the value at input IN is given as a result.

If the value at the N parameter is greater than the number of available bit positions, the operand value at the IN input is still rotated by the specified number of bit positions.

The following figure shows how the content of an operand of the DWORD data type is rotated three bit positions to the left:



Syntax

The following syntax is used for the "Rotate left" instruction:

SCL

```
ROL(IN := <Operand>,
    N := <Operand>)
```

Parameter

The following table shows the parameters of the "Rotate left" instruction:

Parameter	Declaration	Data type		Description
		S7-1200	S7-1500	
IN	Input	Bit strings, integers	Bit strings, integers	Value to be rotated
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	Number of bit positions by which the value (IN) is rotated
Function value		Bit strings, integers	Bit strings, integers	Result of the instruction

For more information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := ROL(IN := "Tag_Value",
                    N := "Tag_Number");
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	1010 1000 1111 0110
N	Tag_Number	5
Function value	Tag_Result	0001 1110 1101 0101

The content of the operand "Tag_Value" is rotated five bit positions to the left. The result of the instruction is returned in the operand "Tag_Result" as a function value.

See also

Overview of the valid data types (Page 2236)

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Additional instructions

DRUM: Implement sequencer

Description

The "Implement sequencer" instruction is used to assign the programmed values of the OUT_VAL parameter of the corresponding step to the programmed output bits (OUT1 to OUT16) and the output word (OUT_WORD). The specific step must thereby satisfy the conditions of the programmed enable mask on the S_MASK parameter while the instruction remains at this step. The instruction advances to the next step if the event for the step is true and the programmed time for the current step elapses, or if the value at the JOG parameter changes from "0" to "1". The instruction is reset if the signal state on the RESET parameter changes to "1". The current step is hereby equated to the preset step (DSP).

The amount of time spent on a step is determined by the product of the preset timebase (DTBP) and the preset counter value (S_PRESET) for each step. At the start of a new step, this calculated value is loaded into the DCC parameter, which contains the time remaining for the current step. If, for example the value at the DTBP parameter is "2" and the preset value for the first step is "100" (100 ms), the DCC parameter has the value "200" (200 ms).

A step can be programmed with a time value, an event, or both. Steps that have an event bit and the time value "0" advance to the next step as soon as the signal state of the event bit is "1". Steps that are programmed only with a time value start the time immediately. Steps that are programmed with an event bit and a time value greater than "0" start the time when the signal state of the event bit is "1". The event bits are initialized with a signal state of "1".

When the sequencer is on the last programmed step (LST_STEP) and the time for this step has expired, the signal state on the Q parameter is set to "1"; otherwise it is set to "0". When the parameter Q is set, the instruction remains on the step until it is reset.

In the configurable mask (S_MASK) you can selected the separate bits in the output word (OUT_WORD) and set or reset the output bits (OUT1 to OUT16) by means of the output values (OUT_VAL). If a bit of the configurable mask is in the signal state "1", the value OUT_VAL sets or resets the corresponding bit. If the signal state of a bit of the configurable mask is "0", the corresponding bit is left unchanged. All the bits of the configurable mask for all 16 steps are initialized with a signal state of "1".

The output bit on the OUT1 parameter corresponds to the least significant bit of the output word (OUT_WORD). The output bit on the OUT16 parameter corresponds to the most significant bit of the output word (OUT_WORD).

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For more information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Implement sequencer" instruction:

```
SCL
<Instance>RESET:= <Operand>,
          JOG := <Operand>,
          DRUM_EN := <Operand>,
          LST_STEP := <Operand>,
          EVENT1 - 16 := <Operand>,
          OUT1 - 16 => <Operand>,
          Q => <Operand>,
          OUT_WORD => <Operand>,
          ERR_CODE => <Operand>)
```

Parameter

The following table shows the parameters of the "Implement sequencer" instruction:

Parameter	Declaration	Data type	Description
RESET	Input	BOOL	A signal state of "1" indicates a reset condition.
JOG	Input	BOOL	When the signal state changes from "0" to "1", the instruction advances to the next step.
DRUM_EN	Input	BOOL	A signal state of "1" allows the sequencer to advance based on the event and time criteria.
LST_STEP	Input	BYTE	Step number of the last step programmed.
EVENT(i), 1 ≤ i ≤ 16	Input	BOOL	Event bit (i); Initial signal state is "1".
OUT(j), 1 ≤ j ≤ 16	Output	BOOL	Output bit (j)
Q	Output	BOOL	A signal state of "1" indicates that the time for the last step has elapsed.
OUT_WORD	Output	WORD	Word address to which the sequencer writes the output values.
ERR_CODE	Output	WORD	Error information
JOG_HIS	Static	BOOL	JOG parameter history bit

Parameter	Declaration	Data type	Description
EOD	Static	BOOL	A signal state of "1" indicates that the time for the last step has elapsed.
DSP	Static	BYTE	Preset step of the sequencer
DSC	Static	BYTE	Current step of the sequencer
DCC	Static	DWORD	Current numerical value of the sequencer
DTBP	Static	WORD	Preset timebase of the sequencer
PREV_TIME	Static	DWORD	Previous system time
S_PRESET	Static	ARRAY of WORD	Count preset for each step [1 to 16] where 1 count = 1 ms.
OUT_VAL	Static	ARRAY of BOOL	Output values for each step [1 to 16, 0 to 15].
S_MASK	Static	ARRAY of BOOL	Configurable mask for each step [1 to 16, 0 to 15]. Initial signal states are "1".

For more information on valid data types, refer to "See also".

The static parameters are not visible when calling the instruction in the program. These are saved in the instance of the instruction.

ERR_CODE Parameter

The following table shows the meaning of the values of the parameter ERR_CODE:

ERR_CODE*	Explanation
W#16#0000	No error
W#16#000B	The value at the LST_STEP parameter is less than 1 or greater than 16.
W#16#000C	The value at the DSC parameter is less than 1 or greater than the value at the LST_STEP parameter.
W#16#000D	The value at the DSP parameter is less than 1 or greater than the value at the LST_STEP parameter.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For more information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Switching display formats in the program status (Page 2980)

DCAT: Discrete control-timer alarm

Description

The "Discrete control-timer alarm" instruction is used to accumulate the time from the point at which the CMD parameter issues the command to open or close. The time is accumulated until the preset time (PT) is exceeded or the information is received that the device was opened or closed (O_FB or C_FB) within the specified time. If the preset time is exceeded before the information on the opening or closing of the device is received, the corresponding alarm is activated. If the signal state on the command input changes state before the preset time, the time is restarted.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For more information on this topic, refer to "See also".

The "Discrete control-timer alarm" instruction has the following reactions to the input conditions:

- When the signal state of the CMD parameter changes from "0" to "1", the signal states of the parameters Q, CMD_HIS, ET (only if ET < PT), OA and CA are influenced as follows:
  - The parameters Q and CMD_HIS are set to "1".
  - The parameters ET, OA and CA are reset to "0".
- When the signal state on the parameter CMD changes from "1" to "0", the parameters Q, ET (only if ET < PT), OA, CA and CMD_HIS are reset to "0".
- When the signal state of the parameters CMD and CMD_HIS is "1" and the parameter O_FB is set to "0", the time difference (ms) since the last execution of the instruction is added to the value at the parameter ET. If the value of the parameter ET exceeds the value of the parameter PT, the signal state on the parameter OA is set to "1". If the value of the parameter ET does not exceed the value of the parameter PT, the signal state on the parameter OA is reset to "0". The value at the parameter CMD_HIS is set to the value of the parameter CMD.
- If the signal state of the parameters CMD, CMD_HIS and O_FB are set to "1" and the parameter C_FB has the value "0", the signal state of the parameter OA is set to "0". The value of parameter ET is set to the value of parameter PT. If the signal state of the parameter O_FB changes to "0", the alarm is set the next time the instruction is executed. The value of parameter CMD_HIS is set to the value of parameter CMD.
- If the parameters CMD, CMD_HIS and C_FB have the value "0", the time difference (ms) since the last execution of the instruction is added to the value of the parameter ET. If the value of the parameter ET exceeds the value of the parameter PT, the signal state of the parameter CA is set to "1". If the value at the parameter PT is not exceeded, the parameter CA has the signal state "0". The value of parameter CMD_HIS is set to the value of parameter CMD.

- If the parameters CMD, CMD_HIS and O_FB have the signal state "0" and the parameter C_FB is set to "1", the parameter CA is set to "0". The value of parameter ET is set to the value of parameter PT. If the signal state of the parameter C_FB changes to "0", the alarm is set the next time the instruction is executed. The value of parameter CMD_HIS is set to the value of parameter CMD.
- If the parameters O_FB and C_FB simultaneously have the signal state "1", the signal states of both alarm outputs are set to "1".

The "Discrete control-timer alarm" instruction has no error information.

Syntax

The following syntax is used for the "Discrete control-timer alarm" instruction:

SCL

```
<Instance>CMD:= <Operand>,
          O_FB := <Operand>,
          C_FB := <Operand>,
          Q => <Operand>,
          OA => <Operand>,
          CA => <Operand>)
```

Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
CMD	Input	BOOL	A signal state of "0" indicates a "close" command. A signal state of "1" indicates an "open" command.
O_FB	Input	BOOL	Feedback input when opening
C_FB	Input	BOOL	Feedback input when closing
Q	Output	BOOL	Shows the status of the parameter CMD
OA	Output	BOOL	Alarm output when opening
CA	Output	BOOL	Alarm output when closing
ET	Static	DINT	Currently elapsed time, where one count = 1 ms.
PT	Static	DINT	Preset time value, where one count = 1 ms.
PREV_TIME	Static	DWORD	Previous system time
CMD_HIS	Static	BOOL	CMD history bit

For more information on valid data types, refer to "See also".

The static parameters are not visible when calling the instruction in the program. These are saved in the instance of the instruction.

Example

In the following example the parameter CMD changes from "0" to "1". After the execution of the instruction the parameter Q is set to "1" and the two alarm outputs OA and CA have the signal state "0". The parameter CMD_HIS of the instance data block is set to the signal state "1" and the parameter ET is reset to "0".

Note

You can initialize static parameters in the data block.

SCL

```
"DCAT_DB"(CMD := "Tag_Input_CMD",
          O_FB := "Tag_Input_O_FB",
          C_FB := "Tag_Input_C_FB",
          Q => "Tag_Output_Q",
          OA => "Tag_Output_OA",
          CA => "Tag_Output_CA");
```

The following tables show how the instruction works using specific values:

Before processing

In this example the following values are used for the input and output parameters:

Parameter	Operand	Value
CMD	Tag_Input_CMD	TRUE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
Q	Tag_Output_Q	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the instance data block "DCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#12
PT	DBD8	L#222
CMD_HIS	DBX16.0	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
Q	Tag_Output_Q	TRUE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the instance data block "DCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#0
CMD_HIS	DBX16.0	TRUE

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

MCAT: Motor control-timer alarm

Description

The "Motor control-timer alarm" instruction is used to accumulate the time from the point at which one of the command inputs (opening or closing) is switched on. The time is accumulated until the preset time is exceeded or the relevant feedback input indicates that the device has executed the requested operation within the specified time. If the preset time is exceeded before the feedback is received, the corresponding alarm is triggered.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Execution of the "Motor control-timer alarm" instruction

The following table shows the reactions of the "Motor control-timer alarm" instruction to the various input conditions:

Input parameters								Output parameters								
ET	O_HIS	C_HIS	O_CMD	C_CMD	S_CMD	O_FB	C_FB	OO	CO	OA	CA	ET	O_HIS	C_HIS	Q	State
X	1	1	X	X	X	X	X	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	X	1	1	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	1	X	X	0	0	0	0	X	0	0	1	Stop
X	X	X	1	1	X	X	X	0	0	0	0	X	0	0	1	Stop
X	0	X	1	0	0	X	X	1	0	0	0	0	1	0	1	Start opening
<PT	1	0	X	0	0	0	X	1	0	0	0	INC	1	0	1	Open
X	1	0	X	0	0	1	0	0	0	0	0	PT	1	0	1	Opened
>=PT	1	0	X	0	0	0	X	0	0	1	0	PT	1	0	0	Opening alarm

Input parameters								Output parameters								
X	X	0	0	1	0	X	X	0	1	0	0	0	0	1	1	Start closing
<PT	0	1	0	X	0	X	0	0	1	0	0	INC	0	1	1	Close
X	0	1	0	X	0	0	1	0	0	0	0	PT	0	1	1	Closed
>=PT	0	1	0	X	0	X	0	0	0	0	1	PT	0	1	0	Closing alarm
X	0	0	0	0	0	X	X	0	0	0	0	X	0	0	1	Stopped

Legend:	
INC	Add the time difference (ms) since the last processing of the FB to ET
PT	PT is set to the same value as ET
X	Cannot be used
<PT	ET < PT
>=PT	ET >= PT

If the input parameters O_HIS and C_HIS both have the signal state "1", they are immediately set to the signal state "0". In this case, the last row in the table (X) mentioned above is valid. Because it is therefore not possible to check whether the input parameters O_HIS and C_HIS have the signal state "1", the output parameters are set as follows in this case:

OO = FALSE

CO = FALSE

OA = FALSE

CA = FALSE

ET = PT

Q = TRUE

Syntax

The following syntax is used for the "Motor control-timer alarm" instruction:

SCL

```
<Instance>O_CMD:= <Operand>,
          C_CMD := <Operand>,
          S_CMD := <Operand>,
          O_FB := <Operand>,
          C_FB := <Operand>,
          OO => <Operand>,
          CO => <Operand>,
          OA => <Operand>,
          CA => <Operand>,
          Q => <Operand>)
```

Parameter

The following table shows the parameters of the "Motor control-timer alarm" instruction:

Parameter	Declaration	Data type	Description
O_CMD	Input	BOOL	"Open" command input
C_CMD	Input	BOOL	"Close" command input

Parameter	Declaration	Data type	Description
S_CMD	Input	BOOL	"Stop" command input
O_FB	Input	BOOL	Feedback input when opening
C_FB	Input	BOOL	Feedback input when closing
OO	Output	BOOL	"Open" output
CO	Output	BOOL	"Close" output
OA	Output	BOOL	Alarm output when opening
CA	Output	BOOL	Alarm output when closing
Q	Output	BOOL	A signal state of "0" indicates an error condition.
ET	Static	DINT	Currently elapsed time, where one count = 1 ms
PT	Static	DINT	Preset time value, where one count = 1 ms
PREV_TIME	Static	DWORD	Previous system time
O_HIS	Static	BOOL	"Open" history bit
C_HIS	Static	BOOL	"Close" history bit

For additional information on valid data types, refer to "See also".

The static parameters are not visible when calling the instruction in the program. These are saved in the instance of the instruction.

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.

SCL

```
"MCAT_DB"(O_CMD := "Tag_Iput_O_CMD",
          C_CMD := "Tag_Input_C_CMD",
          S_CMD := "Tag_Input_S_CMD",
          O_FB := "Tag_Input_O_FB",
          C_FB := "Tag_Input_C_FB",
          OO => "Tag_OutputOpen",
          CO => "Tag_OutputClosed",
          OA => "Tag_Output_OA",
          CA => "Tag_Output_CA",
          Q => "Tag_Output_Q");
```

The following tables show how the instruction works using specific values:

Before processing

In this example the following values are used for the input and output parameters:

Parameter	Operand	Value
O_CMD	Tag_Input_O_CMD	TRUE
C_CMD	Tag_Input_C_CMD	FALSE
S_CMD	Tag_Input_S_CMD	FALSE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
OO	Tag_OutputOpen	FALSE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	FALSE

The following values are saved in the instance data block "MCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#2
PT	DBD8	L#22
O_HIS	DBX16.0	TRUE
C_HIS	DBX16.1	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OO	Tag_OutputOpen	TRUE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	TRUE

The following values are saved in the instance data block "MCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#0
O_HIS	DBX16.0	TRUE
CMD_HIS	DBX16.1	FALSE

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

IMC: Compare input bits with the bits of a mask

Description

The "Compare input bits with the bits of a mask" instruction is used to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bit of a mask. Up to 16 steps with masks can be programmed. The value of the IN_BIT0 parameter is compared with the value of the mask CMP_VAL[x,0], with "x" indicating the step number. On the CMP_STEP parameter, you specify the step number of the mask that is used for the comparison. All programmed values are compared in the same manner. Unprogrammed input bits or unprogrammed bits of the mask have a default signal state FALSE.

If a match is found in the comparison, the signal state of the OUT parameter is set to "1". Otherwise the OUT parameter is set to "0".

If the value of CMP_STEP parameter is greater than 15, the instruction is not executed. An error message is output at the ERR_CODE parameter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For more information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Compare input bits with the bits of a mask" instruction:

SCL

```
<Instance>(IN_BIT0 - 15 := <Operand>,
          CMP_STEP := <Operand>,
          OUT => <Operand>,
          ERR_CODE => <Operand>)
```

Parameters

The following table shows the parameters of the "Compare input bits with the bits of a mask" instruction:

Parameter	Declaration	Data type	Description
IN_BIT0	Input	BOOL	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	Input bit 2 is compared with bit 2 of the mask.

Parameter	Declaration	Data type	Description
IN_BIT3	Input	BOOL	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	Input bit 12 is compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	Input bit 15 is compared with bit 15 of the mask.
CMP_STEP	Input	BYTE	The step number of the mask used for the comparison.
OUT	Output	BOOL	A signal state of "1" indicates that a match was found. A signal state of "0" indicates that no match was found.
ERR_CODE	Output	WORD	Error information
CMP_VAL	Static	ARRAY OF WORD	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

For more information on valid data types, refer to "See also".

The static parameters are not visible when calling the instruction in the program. These are saved in the instance of the instruction.

ERR_CODE Parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000A	The value at the CMP_STEP parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For more information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Switching display formats in the program status (Page 2980)

SMC: Compare scan matrix

Description

You can use the "Compare scan matrix" instruction to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bit of the comparison masks for each step. Processing starts at step 1 and is continued until the last programmed step (LAST) or until a match is found. The input bit of the IN_BIT0 parameter is compared with the value of the mask CMP_VAL[x,0], with "x" indicating the step number. All programmed values are compared in the same manner. If a match is found the signal state of the OUT parameter is set to "1" and the step number with the matching mask is written in the OUT_STEP parameter. Unprogrammed input bits or unprogrammed bits of the mask have a default signal state FALSE. If more than one step has a matching mask, only the first one found is indicated in the OUT_STEP parameter. If no match is found, the signal state of the OUT parameter is set to "0". In this case the value at the OUT_STEP parameter is greater by "1" than the value at the LAST parameter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For more information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Compare scan matrix" instruction:

SCL

```
<Instance>(IN_BIT0 - 15 := <Operand>,
          OUT => <Operand>,
          OUT_STEP => <Operand>,
          ERR_CODE => <Operand>)
```

Syntax

The following table shows the parameters of the "Compare scan matrix" instruction:

Parameter	Declaration	Data type	Description
IN_BIT0	Input	BOOL	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	Input bit 1 is compared with bit 1 of the mask.

Parameter	Declaration	Data type	Description
IN_BIT2	Input	BOOL	Input bit 2 is compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	Input bit 12 is compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	Input bit 15 is compared with bit 15 of the mask.
OUT	Output	BOOL	A signal state of "1" indicates that a match was found. A signal state of "0" indicates that no match was found.
OUT_STEP	Output	BYTE	Contains the step number with the matching mask, or the step number which is greater by "1" than the value at the LAST parameter, provided no match is found.
ERR_CODE	Output	WORD	Error information
LAST	Static	BYTE	Specifies the step number of the last step to be scanned for a matching mask.
CMP_VAL	Static	ARRAY OF WORD	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

For more information on valid data types, refer to "See also".

The static parameters are not visible when calling the instruction in the program. These are saved in the instance of the instruction.

ERR_CODE Parameter

The following table shows the meaning of the values of the parameter ERR_CODE:

Error code* (W#16#...)	Explanation
0000	No error
000E	The value at the LAST parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For more information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Switching display formats in the program status (Page 2980)

LEAD_LAG: Lead and lag algorithm

Description

Use the "Lead and lag algorithm" instruction to process signals with an analog tag. The gain value at the GAIN parameter must be greater than zero. The result of the "Lead and lag algorithm" instruction is calculated using the following equation:

$$OUT = \left[\frac{LG_TIME}{LG_TIME + SAMPLE_T}\right] PREV_OUT + GAIN \left[\frac{LD_TIME + SAMPLE_T}{LG_TIME + SAMPLE_T}\right] IN - GAIN \left[\frac{LD_TIME}{LG_TIME + SAMPLE_T}\right] * PREV_IN$$

The instruction "Lead and lag algorithm" supplies plausible results only when processing is in fixed program cycles. The same units must be specified at the parameters LD_TIME, LG_TIME and SAMPLE_T. At LG_TIME > 4 + SAMPLE_T, the instruction approaches the following function:

OUT = GAIN * ((1 + LD_TIME * s) / (1 + LG_TIME * s)) * IN

When the value of the GAIN parameter is less than or equal to zero, the calculation is not performed and an error information is output on the ERR_CODE parameter.

You can use the "Lead and lag algorithm" instruction in conjunction with loops as a compensator in dynamic feed-forward control. The instruction consists of two operations. The "Lead" operation shifts the phase of output OUT so that the output leads the input. The "Lag" operation, on the other hand, shifts the output so that the output lags behind the input. Because the "Lag" operation is equivalent to an integration, it can be used as a noise suppressor or as a low-pass filter. The "Lead" operation is equivalent to a differentiation and can therefore be used as a high-pass filter. The two instructions together (Lead and Lag) result in the output phase lagging behind the the input at lower frequencies and leading it at higher frequencies. This means that the "Lead and lag algorithm" instruction can be used as a band pass filter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For more information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Lead and lag algorithm" instruction:

SCL

```
<Instance>(IN:= <Operand>,
          SAMPLE_T := <Operand>,
          OUT => <Operand>,
          ERR_CODE => <Operand>)
```

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
IN	Input	REAL	The input value of the current sample time (cycle time) to be processed. Constants can also be specified on the IN parameter.
SAMPLE_T	Input	INT	Sample time Constants can also be specified on the SAMPLE_T parameter.
OUT	Output	REAL	Result of the instruction
ERR_CODE	Output	WORD	Error information
LD_TIME	Static	REAL	Lead time in the same unit as sample time.
LG_TIME	Static	REAL	Lag time in the same unit as sample time
GAIN	Static	REAL	Gain as % / % (the ratio of the change in output to a change in input as a steady state).
PREV_IN	Static	REAL	Previous input
PREV_OUT	Static	REAL	Previous output

For more information on valid data types, refer to "See also".

The static parameters are not visible when calling the instruction in the program. These are saved in the instance of the instruction.

ERR_CODE Parameter

The following table shows the meaning of the values of the parameter ERR_CODE:

Error code* (W#16#...)	Explanation
0000	No error
0009	The value at the GAIN parameter is less than or equal to zero.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For more information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.

SCL

```
"LEAD_LAG_DB"(IN := "Tag_Input",
              SAMPLE_T := "Tag_Input_SAMPLE_T",
              OUT => "Tag_Output_Result",
              ERR_CODE => "Tag_ErrorCode");
```

The following tables show how the instruction works using specific values:

Before processing

In this example the following values are used for the input parameters:

Parameter	Operand	Value
IN	Tag_Input	2.0
SAMPLE_T	Tag_Input_SAMPLE_T	10

The following values are saved in the instance data block "LEAD_LAG_DB" of the instruction:

Parameter	Address	Value
LD_TIME	DBD12	2.0
LG_TIME	DBD16	2.0
GAIN	DBD20	1.0
PREV_IN	DBD24	6.0
PREV_OUT	DBD28	6.0

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OUT	Tag_Output_Result	2.0

The following values are saved in the instance data block "LEAD_LAD_DB" of the instruction:

Parameter	Operand	Value
PREV_IN	DBD24	2.0
PREV_OUT	DBD28	2.0

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

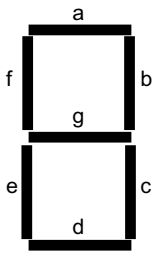
Switching display formats in the program status (Page 2980)

SEG: Create bit pattern for seven-segment display

Description

The "Create bit pattern for seven-segment display" instruction is used to convert each of the four hexadecimal digits of the specified source word (IN) into an equivalent bit pattern for a 7-segment display. The result of the instruction is output in the double word on the OUT parameter.

The following relation exists between the hexadecimal digits and the assignment of the 7 segments (a, b, c, d, e, f, g):

Input digit (Binary)	Assignment of the segments - g f e d c b a	Display (Hexadecimal)	Seven-segment display
0000	00111111	0	
0001	00000110	1	
0010	01011011	2	
0011	01001111	3	
0100	01100110	4	
0101	01101101	5	
0110	01111101	6	
0111	00000111	7	
1000	01111111	8	
1001	01100111	9	
1010	01110111	A	
1011	01111100	B	
1100	00111001	C	
1101	01011110	D	

1110	01111001	E
1111	01110001	F

Syntax

Use the following syntax for the "Create bit pattern for seven-segment display" instruction:

```
SCL
SEG(IN := <Operand>,
    OUT => <Operand>)
```

Parameter

The following table shows the parameters of the "Create bit pattern for seven-segment display" instruction:

Parameter	Declaration	Data type	Description
IN	Input	WORD	Source word with four hexadecimal digits
OUT	Output	DWORD	Bit pattern for the seven-segment display
Function value		VOID	Empty function value

Example

The following example shows how the instruction works:

```
SCL
SEG(IN := "Tag_Input",
    OUT => "Tag_Output");
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
		Hexadecimal	Binary
IN	Tag_Input	W#16#1234	0001 0010 0011 0100
OUT	Tag_Output	DW16#065B4F66	00000110 01011011 01001111 01100110 Display: 1234

See also

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Overview of the valid data types (Page 2236)

BCDCPL: Create tens complement

Description

The "Create tens complement" instruction is used to create the tens complement of a seven-digit BCD number specified by the operand. This instruction uses the following mathematical formula to calculate:

10000000 (as BCD)

– 7-digit BCD value

Tens complement (as BCD)

Syntax

The following syntax is used for the "Create tens complement" instruction:

SCL

```
BCDCPL(<Operand>)
```

Parameter

The following table shows the parameters of the "Create tens complement" instruction:

Parameter	Declaration	Data type	Description
<Operand>	Input	Bit strings	7-digit BCD number
Function value		DWORD	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := BCDCPL("Tag_Input");
```

The following table shows how the instruction functions using specific values:

Operand	Value*
Tag_Input	DW#16#01234567
Tag_Result	DW#16#08765433
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)
Entering SCL instructions (Page 2697)
Editing SCL instructions (Page 2716)
Switching display formats in the program status (Page 2980)

BITSUM: Count number of set bits

Description

The "Count number of set bits" instruction is used to count the number of bits of an operand that are set to the signal state "1".

Syntax

The following syntax is used for the "Count number of set bits" instruction:

```
SCL
BITSUM(<Operand>)
```

Parameter

The following table shows the parameters of the "Count number of set bits" instruction:

Parameter	Declaration	Data type	Description
<Operand>	Input	DWORD	Operand whose set bits are counted
Function value		INT	Result of the instruction

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := BITSUM("Tag_Input");
```

The following table shows how the instruction functions using specific values:

Operand	Value*
Tag_Input	DW#16#12345678
Tag_Result	W#16#000D (13 bits)
*The error codes can be displayed as integer or hexadecimal value in the program editor. For more information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)
Entering SCL instructions (Page 2697)
Editing SCL instructions (Page 2716)
Switching display formats in the program status (Page 2980)

GRAPH

GRAPH structure

Step and transition

Description

You can use the "Step and transition" structure element to insert a step and transition simultaneously in your sequencer.

See also:

Step (Page 4327)
Transition (Page 4328)

See also

Steps and transitions (Page 2724)
Step (Page 4327)
Transition (Page 4328)
Sequence end (Page 4329)
Jump to step (Page 4329)
Alternative branch (Page 4330)
Simultaneous branch (Page 4330)
Close branch (Page 4331)

Step

Description

Steps are used to subdivide complex automation tasks into straightforward subtasks that can then be accomplished using actions. The individual steps are organized in sequencers so that every step can be executed in the specified order in the program execution. Each step must be assigned a unique name and number.

For the execution to actually be performed, the step must be activated by one of the following conditions:

- The step has been defined as an initial step.
- The transition of the previous step has been fulfilled.
- The step has been called by an event-dependent action.

Once all actions have been executed, the step becomes inactive again.

Steps in which no actions are programmed are called empty steps. Such an empty step reacts like an active step and the subsequent transition is always fulfilled.

See also

Basics on steps (Page 2724)

Step and transition (Page 4327)

Transition (Page 4328)

Sequence end (Page 4329)

Jump to step (Page 4329)

Alternative branch (Page 4330)

Simultaneous branch (Page 4330)

Close branch (Page 4331)

Transition

Description

Transitions are located between the steps and contain the conditions for advancing from one step to the next (step-enabling). When the step-enabling conditions of a transition are fulfilled, the next step becomes active and its actions are executed. You can program the conditions of a transition either in LAD or FBD.

See also

Basics on transitions (Page 2726)

Step and transition (Page 4327)

Step (Page 4327)

Sequence end (Page 4329)

Jump to step (Page 4329)

Alternative branch (Page 4330)

Simultaneous branch (Page 4330)

Close branch (Page 4331)

Sequence end

Description

You can use the "Sequence end" structure element to stop a sequencer or a branch. In a simultaneous branch, a transition must precede the sequence end.

Note

If all branches of a sequencer are completed with a sequence end, the sequencer can be restarted by the "INIT_SQ" parameter or using the "Initialize" button in the "Sequence control" pane of the "Testing" task card.

See also

Elements of a sequencer (Page 2721)

Step and transition (Page 4327)

Step (Page 4327)

Transition (Page 4328)

Jump to step (Page 4329)

Alternative branch (Page 4330)

Simultaneous branch (Page 4330)

Close branch (Page 4331)

Jump to step

Description

You can use a jump to continue the program execution at any step within the GRAPH function block. Jumps can be inserted at the end of the main branch or an alternative branch to enable cyclic processing of the sequencer. The jump and jump destination are represented in the sequencer as arrows, in which case the return transition is specified for the jump destination.

See also

Elements of a sequencer (Page 2721)

Step and transition (Page 4327)

Step (Page 4327)

Transition (Page 4328)

Sequence end (Page 4329)

Alternative branch (Page 4330)

Simultaneous branch (Page 4330)

Close branch (Page 4331)

Alternative branch

Description

You can use alternative branches to program OR branches. This means that you insert branches that start with a transition after a step. The corresponding branch is executed depending on which transition is satisfied first. If several transitions are satisfied at the same time, the transition farthest to the left has the highest priority and the corresponding branch is executed. Alternative branches end once again with a transition.

You can program up to 125 alternative branches in a sequencer.

See also

Elements of a sequencer (Page 2721)

Step and transition (Page 4327)

Step (Page 4327)

Transition (Page 4328)

Sequence end (Page 4329)

Jump to step (Page 4329)

Simultaneous branch (Page 4330)

Close branch (Page 4331)

Simultaneous branch

Description

You can use simultaneous branches to program AND branches. This means that you use one transition to activate multiple steps, the actions of which are then executed. Simultaneous branches always start with a step.

The subsequent transitions of the simultaneous branches are located on the main branch, in which case the various simultaneous branches can be connected to various points of the main branch. Note that branches that you join together in a transition only advance to the next step when all branches are completely executed.

You can program up to 249 simultaneous branches in a sequencer.

See also

Elements of a sequencer (Page 2721)

Step and transition (Page 4327)

Step (Page 4327)
Transition (Page 4328)
Sequence end (Page 4329)
Jump to step (Page 4329)
Alternative branch (Page 4330)
Close branch (Page 4331)

Close branch

Description

You can use the "Close branch" element to close simultaneous and alternative branches to their parent branch. This is necessary if you do not want to complete the branch with a jump or a sequence end. In a simultaneous branch, you can insert "Close branch" only after a step.

See also

Elements of a sequencer (Page 2721)
Step and transition (Page 4327)
Step (Page 4327)
Transition (Page 4328)
Sequence end (Page 4329)
Jump to step (Page 4329)
Alternative branch (Page 4330)
Simultaneous branch (Page 4330)

GRAPH actions

Timer operations

TP: Generate pulse

Description

You can use the "Generate pulse" instruction to set the Q output for a programmed duration. The instruction is started when the result of logic operation (RLO) at the IN parameter changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. The Q parameter is set for the time PT, regardless of the subsequent changes in the input signal. Even when a new positive signal edge is detected, the signal state of the Q parameter is not affected as long as time PT is running.

You can query the current time value set in the ET parameter. The time value starts at T#0s and ends when the value of time PT is reached. When the time PT is reached and the signal state of the IN parameter is "0", the ET parameter is reset.

Note

If the time is not called in the program because it is skipped, for example, the ET parameter returns a constant value as soon as the time has expired.

Each call of the "Generate pulse" instruction must be assigned an IEC timer in which the instruction data is stored. An IEC timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TP_TIME or TP_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TP_TIME, TP_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC Timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when the instruction is called and also each time the outputs Q or ET are accessed.

The execution of the "Generate pulse" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Syntax

The following syntax is used for the "Generate pulse" instruction:

GRAPH

```
CALL TP <Data type>, <IEC timer>
     (IN := <Operand>
      PT := <Operand>
      Q => <Operand>
      ET => <Operand>
      )
```

Parameters

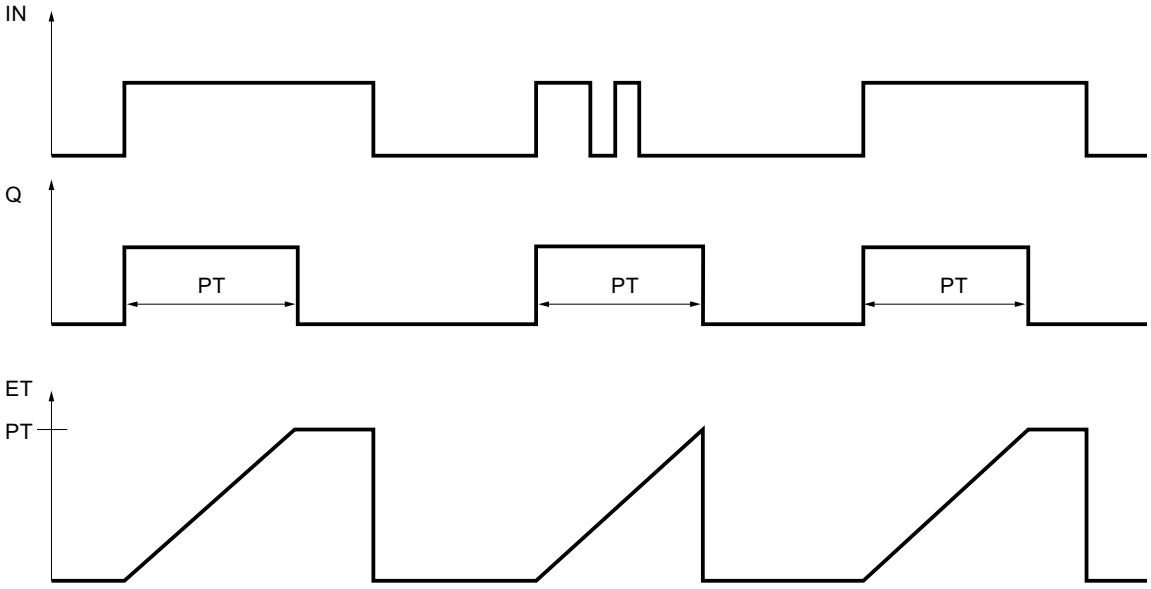
The following table shows the parameters of the "Generate pulse" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L, P or constant	Start input
PT	Input	TIME, LTIME	I, Q, M, D, L, P or constant	Duration of the pulse. The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L, P	Pulse output
ET	Output	TIME, LTIME	I, Q, M, D, L, P	Current time value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Generate pulse" instruction after the start:



Example

The following example shows how the instruction works:

GRAPH

```
CALL TP TIME, "IEC_TP_DB"
     (IN := "Tag_Start"
      PT := "Tag_PresetTIME"
```

GRAPH

```
Q => "Tag_Status"
ET => "Tag_ElapsedTIME"
)
```

When the signal state at the "Tag_Start" operand changes from "0" to "1", the time programmed for the PT parameter is started and the "Tag_Status" operand is set to "1". The current time value is stored in the "Tag_ElapsedTIME" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

TON: Generate on-delay

Description

You can use the "Generate on-delay" instruction to delay setting of the Q output by the programmed time PT. The instruction is started when the result of logic operation (RLO) at the IN parameter changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. When the time PT has elapsed, the Q parameter has the signal state "1". The Q parameter remains set as long as the start input IN is still "1". When the signal state at the start input changes from "1" to "0", the Q parameter is reset. The timer function is started again when a new positive signal edge is detected at the start input.

You can query the current time value at the ET output. The time value starts at T#0s and ends when the value of time PT is reached. The ET parameter is reset as soon as the signal state at the IN parameter changes to "0".

Note

If the timer is not called in the program because it is skipped, for example, the ET output returns a constant value as soon as the time has expired.

Each call of the "Generate on-delay" instruction must be assigned an IEC timer in which the instruction data is stored. An IEC timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TON_TIME or TON_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TON_TIME, TON_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC Timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when the instruction is called and also each time the outputs Q or ET are accessed.

The execution of the "Generate on-delay" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Syntax

The following syntax is used for the "Generate on-delay" instruction:

GRAPH

```
CALL TON <Data type>, <IEC timer>
     (IN := <Operand>
      PT := <Operand>
      Q => <Operand>
      ET => <Operand>
      )
```

Parameters

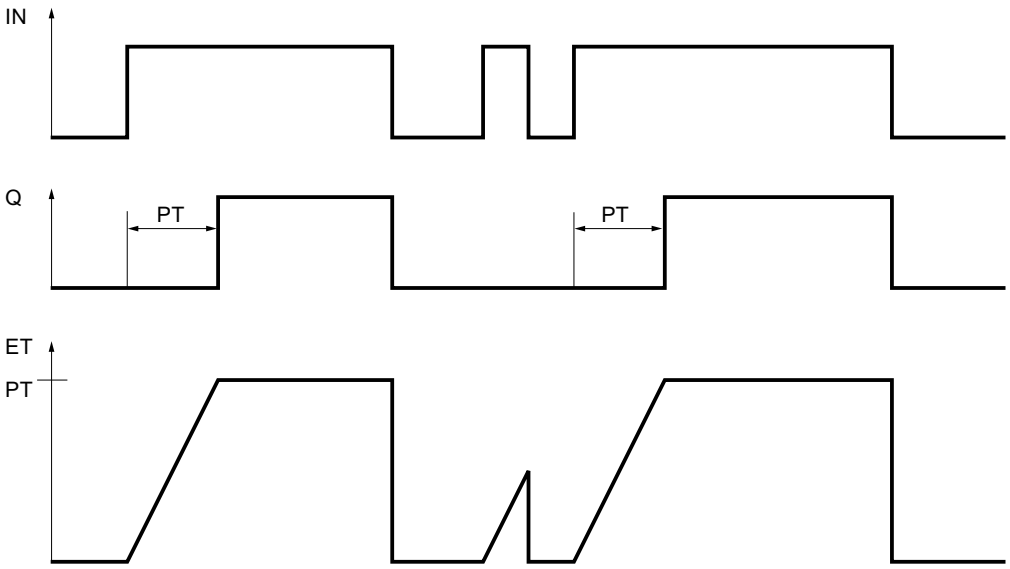
The following table shows the parameters of the "Generate on-delay" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L, P or constant	Start input
PT	Input	TIME, LTIME	I, Q, M, D, L, P or constant	Duration of the on-delay The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L, P	Signal state that is delayed by the time PT.
ET	Output	TIME, LTIME	I, Q, M, D, L, P	Current time value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the behavior of the "Generate on-delay" instruction after the start:



Example

The following example shows how the instruction works:

GRAPH

```
CALL TON TIME, "IEC_TON_DB"
     (IN := "Tag_Start"
      PT := "Tag_PresetTIME"
      Q => "Tag_Status"
      ET => "Tag_ElapsedTIME"
      )
```

When the signal state at the "Tag_Start" operand changes from "0" to "1", the time programmed for the PT parameter is started. When the time expires, the "Tag_Status" operand is set to the signal state "1". The Tag_Status operand remains set to "1" as long as the Tag_Start operand has signal state "1". The current time value is stored in the "Tag_ElapsedTime" operand. When the signal state at the "Tag_Start" operand changes from "1" to "0", the "Tag_Status" operand is reset.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

TOF: Generate off-delay

Description

You can use the "Generate off-delay" instruction to delay resetting of the Q output by the programmed time PT. The Q parameter is set when the result of logic operation (RLO) of the IN parameter changes from "0" to "1" (positive signal edge). When the signal state at the IN parameter changes back to "0", the programmed time PT starts. The Q parameter remains set as long as time PT is running. When the time PT expires, the Q parameter is reset. If the signal state of the IN parameter changes to "1" before time PT expires, the timer is reset. The signal state of the Q parameter remains set to "1".

The current time value can be queried in the ET parameter. The time value starts at T#0s and ends when the value of time PT is reached. When the time PT expires, the ET parameter remains set to the current value until the IN parameter changes back to "1". If the IN input changes to "1" before time PT has expired, the ET output is reset to the value T#0s.

Note

If the timer is not called in the program because it is skipped, for example, the ET output returns a constant value as soon as the time has expired.

Each call of the "Generate off-delay" instruction must be assigned an IEC timer in which the instruction data is stored. An IEC timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TOF_TIME or TOF_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TOF_TIME, TOF_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC Timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when the instruction is called and also each time the outputs Q or ET are accessed.

The execution of the "Generate off-delay" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Syntax

The following syntax is used for the "Generate off-delay" instruction:

GRAPH

```
CALL TOF <Data type>, <IEC timer>
     (IN := <Operand>
      PT := <Operand>
```

GRAPH

```
Q => <Operand>
ET => <Operand>
)
```

Parameters

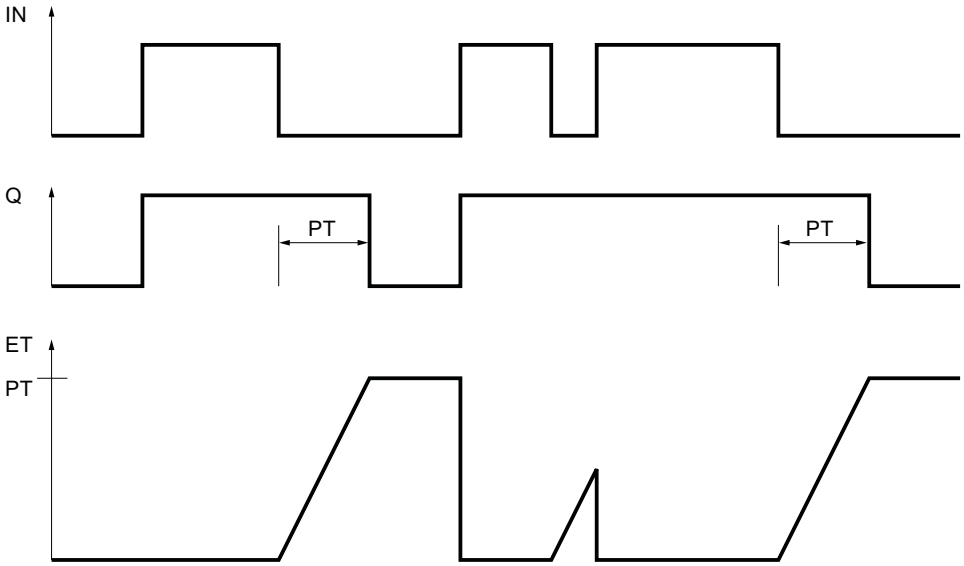
The following table shows the parameters of the "Generate off-delay" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L, P or constant	Start input
PT	Input	TIME, LTIME	I, Q, M, D, L, P or constant	Duration of the off delay The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L, P	Signal state that is delayed by the time PT.
ET	Output	TIME, LTIME	I, Q, M, D, L, P	Current time value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the behavior of the "Generate off-delay" instruction after the start:



Example

The following example shows how the instruction works:

GRAPH

```
CALL TOF TIME, "IEC_TOF_DB"
     (IN := "Tag_Start"
      PT := "Tag_PresetTIME"
      Q => "Tag_Status"
      ET => "Tag_ElapsedTIME"
      )
```

When the signal state at the "Tag_Start" operand changes from "0" to "1", the "Tag_Status" operand is set. When the signal state at the "Tag_Start" operand changes from "1" to "0", the time programmed for the PT parameter is started. As long as the time is running, the "Tag_Status" operand remains set. When the time has expired, the Tag_Status operand is reset. The current time value is stored in the "Tag_ElapsedTIME" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

TONR: Time accumulator

Description

You can use the "Time accumulator" instruction to accumulate time values within a period set by the PT parameter. When the signal state at the IN input changes from "0" to "1" (positive signal edge), the instruction is executed and time PT starts. While the time PT is running, the time values are accumulated that are recorded when the IN input has signal state "1". The accumulated time is written to the ET output and can be queried there. When the time PT expires, the output Q has the signal state "1". The Q parameter remains set to "1", even when the signal state at the IN parameter changes from "1" to "0" (negative signal edge).

The R input resets the ET and Q outputs regardless of the signal state at the start input.

Each call of the "Time accumulator" instruction must be assigned an IEC timer in which the instruction data is stored. An IEC timer is a structure of the data type IEC_TIMER, IEC_LTIMER, TONR_TIME or TONR_LTIME that you can declare as follows:

- Declaration of a data block of system data type IEC_TIMER or IEC_LTIMER (for example, "MyIEC_TIMER")
- Declaration as a local tag of the type TONR_TIME, TONR_LTIME, IEC_TIMER or IEC_LTIMER in the "Static" section of a block (for example, #MyIEC_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC Timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find

it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when the instruction is called and also each time the outputs Q or ET are accessed.

The execution of the "Time accumulator" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Syntax

The following syntax is used for the "Time accumulator" instruction:

GRAPH

```
CALL TONR <Data type>, <IEC timer>
     (IN := <Operand>
      R := <Operand>
      PT := <Operand>
      Q => <Operand>
      ET => <Operand>
      )
```

Parameters

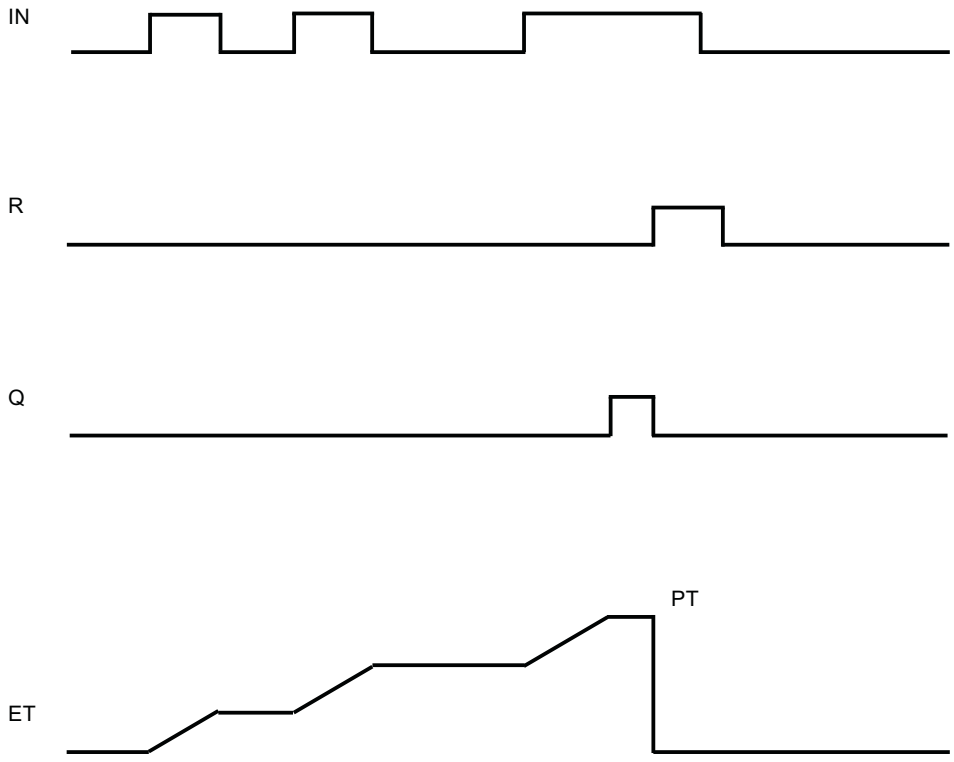
The following table shows the parameters of the "Time accumulator" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L, P or constant	Start input
R	Input	BOOL	I, Q, M, D, L, P or constant	Reset input
PT	Input	TIME, LTIME	I, Q, M, D, L, P or constant	Maximum duration of time recording The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L, P	Output that is set when the time PT expires.
ET	Output	TIME, LTIME	I, Q, M, D, L, P	Accumulated time

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Time accumulator" instruction:



Example

The following example shows how the instruction works:

GRAPH

```
CALL TONR TIME, "IEC_TONR_DB"
     (IN := "Tag_Start"
      R := "Tag_Reset"
      PT := "Tag_PresetTIME"
      Q => "Tag_Status"
      ET => "Tag_ElapsedTIME"
      )
```

When the signal state at the "Tag_Start" operand changes from "0" to "1", the time programmed for the PT parameter is started. While the time is running, the time values are accumulated that are recorded when the "Tag_Start" operand has signal state "1". The accumulated time is stored in the "Tag_ElapsedTIME" operand. When the time value specified for the PT parameter is reached, the "Tag_Status" operand is set to the signal state "1". The current time value is stored in the "Tag_ElapsedTIME" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

Counter operations

CTU: Count up

Description

You can use the "Count up" instruction to increment the value at the CV parameter. When the signal state at the CU parameter changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value of the CV parameter is incremented by one. When the instruction is executed for the first time the current count of the CV parameter is set to zero. The counter value is incremented each time a positive signal edge is detected, until it reaches the high limit for the data type specified at the CV output. When the high limit is reached, the signal state of the CU parameter no longer has an effect on the instruction.

You can query the count status of the Q parameter. The signal state of the Q parameter is determined by the PV parameter. When the current counter value is greater than or equal to the value of the PV parameter, the Q parameter is set to signal state "1". In all other cases, the signal state of the Q parameter is "0".

The value of the CV parameter is reset to zero when the signal state at the R parameter changes to "1". As long as the R parameter has the signal state "1", the signal state of the CU parameter has no effect on the instruction.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count up" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

Data block of system data type IEC_<Counter> (Shared DB)

- IEC_SCOUNTER / IEC_USCOUNTER
- IEC_COUNTER / IEC_UCOUNTER
- IEC_DCOUNTER / IEC_UDCOUNTER
- IEC_LCOUNTER / IEC_ULCOUNTER

Local tag

- CTU_SINT / CTU_USINT
- CTU_INT / CTU_UINT

- CTU_DINT / CTU_UDINT
- CTU_LINT / CTU_ULINT
- IEC_SCOUNTER / IEC_USCOUNTER
- IEC_COUNTER / IEC_UCOUNTER
- IEC_DCOUNTER / IEC_UDCOUNTER
- IEC_LCOUNTER / IEC_ULCOUNTER

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER")
- Declaration as a local tag of the type CTU_<Data type> or IEC_<Counter> in the "Static" section of a block (for example #MyCTU_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

When you set up the IEC counter in a separate data block (single instance), the instance data block is created by default with "optimized block access" and the individual tags are defined as retentive. For additional information on setting retentivity in an instance data block, refer to "See also".

When you set up the IEC counter as local tag (multiple instance) in a function block with "optimized block access", it is defined as retentive in the block interface.

The execution of the "Count up" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Syntax

The following syntax is used for the "Count up" instruction:

GRAPH

```
CALL CTU <Data type>, <IEC counter>
     (CU := <Operand>
      R := <Operand>
      PV := <Operand>
      Q => <Operand>
      CV => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Count up" instruction:

Parameters	Declaration	Data type	Memory area	Description
CU	Input	BOOL	I, Q, M, D, L or constant	Count input
R	Input	BOOL	I, Q, M, D, L, P or constant	Reset input
PV	Input	Integers	I, Q, M, D, L, P or constant	Value at which the Q output is set.
Q	Output	BOOL	I, Q, M, D, L	Counter status
CV	Output	Integers, CHAR, DATE	I, Q, M, D, L, P	Current counter value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL CTU INT, "IEC_CTU_DB"
      (CU := "Tag_Start"
       R := "Tag_ResetCOUNTER"
       PV := "Tag_PresetValue"
       Q => "Tag_Status"
       CV => "Tag_CounterValue"
       )
```

When the signal state at the "Tag_Start" operand changes from "0" to "1", the "Count up" instruction is executed and the current counter value of the "Tag_CounterValue" operand is incremented by one. With each additional positive signal edge, the counter value is incremented until the high limit of the data type (INT = 32767) is reached.

The value of the PV parameter is adopted as the limit for determining the "Tag_Status" output. The "Tag_Status" output has signal state "1" as long as the current counter value is greater than or equal to the value of the "Tag_PresetValue" operand. In all other cases, the "Tag_Status" output has signal state "0".

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

Setting retentivity in an instance data block (Page 2839)

Setting the retentivity of local tags (Page 2561)

CTD: Count down

Description

The "Count down" instruction is used to decrement the value at the parameter CV. When the signal state at the CD parameter changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value of the CV parameter is decremented by one. When the instruction is executed the first time, the counter value of the CV parameter is set to the value of the PV parameter. Each time a positive signal edge is detected, the counter value is decremented until it reaches the low limit of the specified data type. When the low limit is reached, the signal state of the CD parameter no longer has an effect on the instruction.

You can query the count status of the Q parameter. If the current counter value is less than or equal to zero, the Q parameter is set to signal state "1". In all other cases, the signal state of the Q parameter is "0".

The value of the CV parameter is set to the value of the PV parameter when the signal state of the LD parameter changes to "1". As long as the LD parameter has the signal state "1", the signal state of the CD parameter has no effect on the instruction.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count down" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

Data block of system data type IEC_<Counter> (Shared DB)

- IEC_SCOUNTER / IEC_USCOUNTER
- IEC_COUNTER / IEC_UCOUNTER
- IEC_DCOUNTER / IEC_UDCOUNTER
- IEC_LCOUNTER / IEC_ULCOUNTER

Local tag

- CTD_SINT / CTD_USINT
- CTD_INT / CTD_UINT
- CTD_DINT / CTD_UDINT
- CTD_LINT / CTD_ULINT
- IEC_SCOUNTER / IEC_USCOUNTER
- IEC_COUNTER / IEC_UCOUNTER
- IEC_DCOUNTER / IEC_UDCOUNTER
- IEC_LCOUNTER / IEC_ULCOUNTER

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER")
- Declaration as a local tag of the type CTD_<Data type> or IEC_<Counter> in the "Static" section of a block (for example #MyCTD_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

When you set up the IEC counter in a separate data block (single instance), the instance data block is created by default with "optimized block access" and the individual tags are defined as retentive. For additional information on setting retentivity in an instance data block, refer to "See also".

When you set up the IEC counter as local tag (multiple instance) in a function block with "optimized block access", it is defined as retentive in the block interface.

The execution of the "Count down" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Syntax

The following syntax is used for the "Count down" instruction:

GRAPH

```
CALL CTD <Data type>, <IEC counter>
      (CD := <Operand>
       LD := <Operand>
       PV := <Operand>
       Q => <Operand>
       CV => <Operand>
       )
```

Parameters

The following table shows the parameters of the "Count down" instruction:

Parameter	Declaration	Data type	Memory area	Description
CD	Input	BOOL	I, Q, M, D, L or constant	Count input
LD	Input	BOOL	I, Q, M, D, L, P or constant	Load input
PV	Input	Integers	I, Q, M, D, L, P or constant	Value at which the Q output is set.
Q	Output	BOOL	I, Q, M, D, L	Counter status
CV	Output	Integers, CHAR, DATE	I, Q, M, D, L, P	Current counter value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL CTD INT, "IEC_CTD_DB"
     (CD := "Tag_Start"
      LD := "Tag_LoadPV"
      PV := "Tag_PresetValue"
      Q => "Tag_Status"
      CV => "Tag_CounterValue"
      )
```

When the signal state at the "Tag_Start" operand changes from "0" to "1", the "Count down" instruction is executed and the value of the "Tag_CounterValue" output is decremented by one. With each additional positive signal edge, the counter value is decremented until the low limit of the specified data type (INT = -32768) is reached.

The "Tag_Status" output has signal state "1" as long as the current counter value is less than or equal to zero. In all other cases, the "Tag_Status" output has signal state "0".

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

Setting retentivity in an instance data block (Page 2839)

Setting the retentivity of local tags (Page 2561)

CTUD: Count up and down

Description

You can use the "Count up and down" instruction to increment and decrement the value of the CV parameter. When the signal state at the CU parameter changes from "0" to "1" (positive signal edge), the current counter value of the CV parameter is incremented by one. When the signal state at the CD parameter changes from "0" to "1" (positive signal edge), the current counter value of the CV parameter is decremented by one. If there is a positive signal edge at the CU and CD inputs in a program cycle, the current counter value of the CV parameter remains unchanged.

The counter value can be incremented until it reaches the high limit of the data type specified for the CV parameter. When the high limit is reached, the counter value is no longer incremented on a positive signal edge. The counter value is no longer decremented once the low limit of the specified data type has been reached.

When the signal state at the LD parameter changes to "1", the counter value of the CV parameter is set to the value of the PV parameter. As long as the LD parameter has signal state "1", the signal state of the CU and CD inputs has no effect on the instruction.

The counter value is set to zero when the signal state of the R parameter changes to "1". As long as the R parameter has signal state "1", a change in the signal state of the CU, CD and LD parameters has no effect on the "Count up and down" instruction.

You can query the status of the up counter at the QU parameter. When the current counter value is greater than or equal to the value of the PV parameter, the QU parameter is set to signal state "1". In all other cases, the signal state of the QU parameter is "0".

You can query the status of the down counter at the QD parameter. If the current counter value is less than or equal to zero, the QD parameter is set to signal state "1". In all other cases, the signal state of the QD parameter is "0".

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count up and down" instruction must be assigned an IEC counter in which the instruction data is stored. An IEC counter is a structure with one of the following data types:

Data block of system data type IEC_<Counter> (Shared DB)

- IEC_SCOUNTER / IEC_USCOUNTER
- IEC_COUNTER / IEC_UCOUNTER
- IEC_DCOUNTER / IEC_UDCOUNTER
- IEC_LCOUNTER / IEC_ULCOUNTER

Local tag

- CTUD_SINT / CTUD_USINT
- CTUD_INT / CTUD_UINT
- CTUD_DINT / CTUD_UDINT
- CTUD_LINT / CTUD_ULINT
- IEC_SCOUNTER / IEC_USCOUNTER
- IEC_COUNTER / IEC_UCOUNTER
- IEC_DCOUNTER / IEC_UDCOUNTER
- IEC_LCOUNTER / IEC_ULCOUNTER

You can declare an IEC counter as follows:

- Declaration of a data block of system data type IEC_<Counter> (for example, "MyIEC_COUNTER")
- Declaration as a local tag of the type CTUD_<Data type> or IEC_<Counter> in the "Static" section of a block (for example #MyCTUD_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

When you set up the IEC counter in a separate data block (single instance), the instance data block is created by default with "optimized block access" and the individual tags are defined as retentive. For additional information on setting retentivity in an instance data block, refer to "See also".

When you set up the IEC counter as local tag (multiple instance) in a function block with "optimized block access", it is defined as retentive in the block interface.

The execution of the "Count up and down" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Syntax

The following syntax is used for the "Count up and down" instruction:

GRAPH

```
CALL CTUD <Data type>, <IEC counter>
     (CU := <Operand>
      CD := <Operand>
      R := <Operand>
      LD := <Operand>
      PV := <Operand>
      QU => <Operand>
      QD => <Operand>
      CV => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Count up and down" instruction:

Parameter	Declaration	Data type	Memory area	Description
CU	Input	BOOL	I, Q, M, D, L or constant	Count up input
CD	Input	BOOL	I, Q, M, D, L or constant	Count down input
R	Input	BOOL	I, Q, M, D, L, P or constant	Reset input
LD	Input	BOOL	I, Q, M, D, L, P or constant	Load input
PV	Input	Integers	I, Q, M, D, L, P or constant	Value at which the QU output is set.

Parameter	Declaration	Data type	Memory area	Description
QU	Output	BOOL	I, Q, M, D, L	Status of the up counter
QD	Output	BOOL	I, Q, M, D, L	Status of the down counter
CV	Output	Integers, CHAR, DATE	I, Q, M, D, L, P	Current counter value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL CTUD INT, "IEC_CTUD_DB"
      (CU := "Tag_StartCTU"
       CD := "Tag_StartCTD"
       R := "Tag_ResetCOUNTER"
       LD := "Tag_LoadPV"
       PV := "Tag_PresetValue"
       QU => "Tag_CounterStatusUP"
       QD => "Tag_CounterStatusDOWN"
       CV => "Tag_CounterValue"
       )
```

When the signal state at the "Tag_StartCTU" or "Tag_StartCTD" input changes from "0" to "1" (positive signal edge), the "Count up and down" instruction is executed. When there is a positive signal edge in the "Tag_StartCTU" input, the current counter value is incremented by one and stored in the "Tag_CounterValue" output. When there is a positive signal edge in the "Tag_StartCTD" input, the counter value is decremented by one and stored in the "Tag_CounterValue" output. When there is a positive signal edge in the CU input, the counter value is incremented until it reaches the high limit of 32767. When there is a positive signal edged in the CD input, the counter value is decremented until it reaches the low limit of INT = -32768.

The "Tag_CounterStatusUP" output has signal state "1" as long as the current counter value is greater than or equal to the value of the "Tag_PresetValue" input. In all other cases, the "Tag_CounterStatusUP" output has signal state "0".

The "Tag_CounterStatusDOWN" output has signal state "1" as long as the current counter value is less than or equal to zero. In all other cases, the "Tag_CounterStatusDOWN" output has signal state "0".

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

Setting retentivity in an instance data block (Page 2839)

Setting the retentivity of local tags (Page 2561)

Math functions

NEG: Create twos complement

Description

You use the "Create twos complement" instruction to change the sign of the operand value. For example, if the value is positive, the negative equivalent of this value is output.

The value of the OUT output is invalid if the following condition is met:

- The result of the instruction is outside the range permitted for the data type specified for the OUT output (only valid for integers).

Syntax

The following syntax is used for the "Create twos complement" instruction:

GRAPH

```
<Result> := NEG_<Data type>(<Operand>)
```

Parameters

The following table shows the parameters of the "Create twos complement" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	SINT, INT, DINT, LINT, floating-point numbers	I, Q, M, D, L, P, or constant	Input value
<Result>	Output	SINT, INT, DINT, LINT, floating-point numbers	I, Q, M, D, L, P	Twos complement of the input value

You can select the data type of the instruction from the "<???>" drop-down list.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := NEG_REAL("Tag_InValue")
```

The instruction changes the sign of the value of the "Tag_InValue" operand and outputs the result in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

ABS: Form absolute value

Description

You use the "Form absolute value" instruction to calculate the absolute value of the operand value.

The value of the result is invalid if the following condition is met:

- A floating-point number has an invalid value.

Syntax

The following syntax is used for the "Form absolute value" instruction:

GRAPH

```
<Result> := ABS(<Operand>)
```

Parameters

The following table shows the parameters of the "Form absolute value" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	SINT, INT, DINT, LINT, floating-point numbers	I, Q, M, D, L, P, or constant	Input value
<Result>	Output	SINT, INT, DINT, LINT, floating-point numbers	I, Q, M, D, L, P	Absolute value of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := ABS("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	-6.234
Tag_OutValue	6.234

The instruction calculates the absolute value of the value of the "Tag_InValue" operand and outputs the result in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

MIN: Get minimum

Description

You can use the "Get minimum" instruction to compare the values of the available IN1 and IN2 inputs and write the lowest value in the OUT output.

The value of the OUT output is invalid if one of the following conditions is met:

- The implicit conversion of the data types fails during execution of the instruction.
- A floating-point number has an invalid value.

Syntax

The following syntax is used for the "Get minimum" instruction:

GRAPH

```
CALL MIN <Data type>
      (IN1 := <Operand>
       IN2 := <Operand>
       OUT => <Operand>
       )
```

Parameters

The following table shows the parameters of the "Get minimum" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	First input value
IN2	Input	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	Second input value
OUT	Output	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	Result
The data types TOD, LTOD, DATE, and LDT can only be used if the IEC test is not enabled.				

You can select the data type of the instruction from the "<data type>" drop-down list.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL MIN INT
     (IN1 := "TagIn_Value1"
      IN2 := "TagIn_Value2"
      OUT => "Tag_Minimum"
      )
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
OUT	Tag_Minimum	12222

The instruction compares the values of the specified operands and copies the lowest value ("TagIn_Value1") to the "Tag_Minimum" output.

See also

Overview of the valid data types (Page 2236)
Basic information on the status word (Page 2466)
Basic information on GRAPH (Page 2718)

MAX: Get maximum

Description

You can use the "Get maximum" instruction to compare the values of the IN1 and IN2 inputs and write the highest value to the OUT output.

The value of the OUT output is invalid if one of the following conditions is met:

- The implicit conversion of the data types fails during execution of the instruction.
- A floating-point number has an invalid value.

Syntax

The following syntax is used for the "Get maximum" instruction:

GRAPH

```
CALL MAX <Data type>
     (IN1 := <Operand>
      IN2 := <Operand>
      OUT => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Get maximum" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	First input value
IN2	Input	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	Second input value

Parameter	Declaration	Data type	Memory area	Description
OUT	Output	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	Result
The data types TOD, LTOD, DATE, and LDT can only be used if the IEC test is not enabled.				

You can select the data type of the instruction from the "<data type>" drop-down list.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL MAX INT
     (IN1 := "TagIn_Value1"
      IN2 := "TagIn_Value2"
      OUT => "Tag_Maximum"
      )
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
OUT	Tag_Maximum	14444

The instruction compares the values of the specified operands and copies the highest value ("TagIn_Value2") to the "Tag_Maximum" output.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

LIMIT: Set limit value

Description

You can use the "Set limit value" instruction to limit the value of the IN input to the values of the MN and MX inputs. If the value of the IN input meets the condition MN <= IN <= MX, it is copied to the OUT output. If the condition is not met and the IN input value is below the low limit MN, the OUT output is set to the value of the MN input. If the high limit MX is exceeded, the OUT output is set to the value of the MX input.

If the value of the MN input is greater than the value of the MX input, the result is undefined. The instruction is only executed if the tags of all inputs are of the same data type.

The value of the OUT output is invalid if one of the following conditions is met:

- The specified tags are not of the same data type.
- An operand has an invalid value.
- The value of the MN parameter is greater than the value of the MX parameter.

Syntax

The following syntax is used for the "Set limit value" instruction:

GRAPH

```
CALL LIMIT <Data type>
     (MN := <Operand>
      IN := <Operand>
      MX := <Operand>
      OUT => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Set limit value" instruction:

Parameter	Declaration	Data type	Memory area	Description
MN	Input	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	Low limit
IN	Input	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	Input value
MX	Input	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	High limit
OUT	Output	Integers, floating-point numbers, TIME, LTIME, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P	Result
The data types TOD, LTOD, DATE, and LDT can only be used if the IEC test is not enabled.				

You can select the data type of the instruction from the "<data type>" drop-down list.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL LIMIT INT
      (MN := "Tag_LowLimit"
       IN := "Tag_InputValue"
       MX := "Tag_HighLimit"
       OUT => "Tag_Result"
       )
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
MN	Tag_LowLimit	12000
IN	Tag_InputValue	8000
MX	Tag_HighLimit	16000
OUT	Tag_Result	12000

The value of the "Tag_InputValue" operand is compared with the values of the "Tag_LowLimit" and "Tag_HighLimit" operands. Since the value of the "Tag_InputValue" operand is less than the low limit, the value of the "Tag_LowLimit" operand is copied to the "Tag_Result" output.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

SQR: Form square

Description

You use the "Form square" instruction to square the value of the operand.

The value of the result is invalid if the following condition is met:

- The value of the operand is not a valid floating-point number.

Syntax

The following syntax is used for the "Form square" instruction:

GRAPH
```
<Result> := SQR(<Operand>)
```

Parameters

The following table shows the parameters of the "Form square" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Input value
<Result>	Output	Floating-point numbers	I, Q, M, D, L, P	Square of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH
```
"Tag_OutValue" := SQR("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	5.0
Tag_OutValue	25.0

The instruction squares the value of the "Tag_InValue" operand and outputs the result in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

SQRT: Form square root

Description

You use the "Form square root" instruction to calculate the square root of the value of the operand. The instruction outputs a positive result if the input value is greater than zero. If input values are less than zero, the instruction returns an invalid floating-point number. If the value of the operand is "0", then the result is also "0".

The value of the result is invalid if any of the following conditions are met:

- The value of the operand is not a valid floating-point number.
- The value of the operand is negative.

Syntax

The following syntax is used for the "Form square root" instruction:

GRAPH

```
<Result> := SQRT(<Operand>)
```

Parameters

The following table shows the parameters of the "Form square root" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Input value
<Result>	Output	Floating-point numbers	I, Q, M, D, L	Square root of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := SQRT("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	25.0
Tag_OutValue	5.0

The instruction calculates the square root of the value of the "Tag_InValue" operand and outputs the result in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

Invalid floating-point numbers (Page 2254)

LN: Form natural logarithm

Description

You use the "Form natural logarithm" instruction to calculate the natural logarithm to base e (e = 2.718282) from the value of the operand. The instruction outputs a positive result if the input value is greater than zero. If input values are less than zero, the instruction returns an invalid floating-point number.

The value of the result is invalid if any of the following conditions are met:

- The value of the operand is not a valid floating-point number.
- The value of the operand is negative.

Syntax

The following syntax is used for the "Form natural logarithm" instruction:

GRAPH

```
<Result> := LN(<Operand>)
```

Parameters

The following table shows the parameters of the "Form natural logarithm" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Input value
<Result>	Output	Floating-point numbers	I, Q, M, D, L, P	Natural logarithm of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := LN("Tag_InValue")
```

The instruction calculates the natural logarithm of the value in the "Tag_InValue" operand and outputs the result in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

Invalid floating-point numbers (Page 2254)

EXP: Form exponential value

Description

You use the "Form exponential value" instruction to calculate the exponent from the base e (e = 2.718282) and the value of the operand.

The value of the result is invalid if the following condition is met:

- The value of the operand is not a valid floating-point number.

Syntax

The following syntax is used for the "Form exponential value" instruction:

GRAPH

```
<Result> := EXP(<Operand>)
```

Parameters

The following table shows the parameters of the "Form exponential value" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Input value
<Result>	Output	Floating-point numbers	I, Q, M, D, L, P	Exponential value of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := EXP("Tag_InValue")
```

The instruction calculates the exponent from base e and the value of the "Tag_InValue" operand and outputs the result e^{IN}) in the Tag_OutValue operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

Invalid floating-point numbers (Page 2254)

SIN: Form sine value

Description

You use the "Form sine value" instruction to calculate the sine of an angle. The size of the angle is specified in radians in the operand.

The value of the result is invalid if the following condition is met:

- The value of the operand is not a valid floating-point number.

Syntax

The following syntax is used for the "Form sine value" instruction:

GRAPH

```
<Result> := SIN(<Operand>)
```

Parameters

The following table shows the parameters of the "Form sine value" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Size of angle in radians
<Result>	Output	Floating-point numbers	I, Q, M, D, L, P	Sine of the specified angle

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := SIN("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	+1.570796 (π/2)
Tag_OutValue	1.0

The instruction calculates the sine of the angle specified in the "Tag_InValue" operand and outputs the result in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

COS: Form cosine value

Description

You use the "Form cosine value" instruction to calculate the cosine of an angle. The size of the angle is specified in radians in the operand.

The value of the result is invalid if the following condition is met:

- The value of the operand is not a valid floating-point number.

Syntax

The following syntax is used for the "Form cosine value" instruction:

GRAPH

```
<Result> := COS(<Operand>)
```

Parameters

The following table shows the parameters of the "Form cosine value" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Size of angle in radians
<Result>	Output	Floating-point numbers	I, Q, M, D, L, P	Cosine of the specified angle

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH
```
"Tag_OutValue" := COS("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	+1.570796 (π/2)
Tag_OutValue	0

The instruction calculates the cosine of the angle specified in the "Tag_InValue" operand and outputs the result in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

TAN: Form tangent value

Description

You use the "Form tangent value" instruction to calculate the tangent of an angle. The size of the angle is specified in radians in the operand.

The value of the result is invalid if the following condition is met:

- The value of the operand is not a valid floating-point number.

Syntax

Use the following syntax for the "Form tangent value" instruction:

GRAPH
```
<Result> := TAN(<Operand>)
```

Parameters

The following table shows the parameters of the "Form tangent value" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Size of angle in radians
<Result>	Output	Floating-point numbers	I, Q, M, D, L, P	Tangent of the specified angle

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := TAN("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	+3.141593 (π)
Tag_OutValue	0

The instruction calculates the tangent of the angle specified in the "Tag_InValue" operand and outputs the result in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

ASIN: Form arcsine value

Description

You use the "Form arcsine value" instruction to calculate the size of the angle that corresponds to the sine value specified in the operand. Only valid floating-point numbers within the range -1 to +1 can be specified for the operand. The calculated angle size can range in value from -π/2 to +π/2.

The value of the result is invalid if any of the following conditions are met:

- The value of the operand is not a valid floating-point number.
- The value of the operand is outside the permitted value range (-1 to +1).

Syntax

Use the following syntax for the "Form arcsine value" instruction:

GRAPH

```
<Result> := ASIN(<Operand>)
```

Parameters

The following table shows the parameters of the "Form arcsine value" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Sine value
<Result>	Output	Floating-point numbers	I, Q, M, D, L, P	Size of angle in radians

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := ASIN("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	1.0
Tag_OutValue	+1.570796 (π/2)

The instruction calculates the size of the angle corresponding to the sine value in the "Tag_InValue" operand. The result of the instruction is stored in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

ACOS: Form arccosine value

Description

You use the "Form arccosine value" instruction to calculate the size of the angle that corresponds to the cosine value specified in the operand. Only valid floating-point numbers within the range -1 to +1 can be specified for the operand. The calculated angle size can range in value from 0 to +π.

The value of the result is invalid if any of the following conditions are met:

- The value of the operand is not a valid floating-point number.
- The value of the operand is outside the permitted value range (-1 to +1).

Syntax

Use the following syntax for the "Form arccosine value" instruction:

GRAPH

```
<Result> := ACOS(<Operand>)
```

Parameters

The following table shows the parameters of the "Form arccosine value" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Cosine value
<Result>	Output	Floating-point numbers	I, Q, M, D, L, P	Size of angle in radians

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := ACOS("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	0
Tag_OutValue	+1.570796 (π/2)

The instruction calculates the size of the angle corresponding to the cosine value in the "Tag_InValue" operand. The result of the instruction is stored in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 2236)
Basic information on the status word (Page 2466)
Basic information on GRAPH (Page 2718)

ATAN: Form arctangent value

Description

You use the "Form arctangent value" instruction to calculate the size of the angle that corresponds to the tangent value specified in the operand. Only valid floating-point numbers may be specified for the operand. The calculated angle size can range in value from -π/2 to +π/2.

The value of the result is invalid if the following condition is met:

- The value of the operand is not a valid floating-point number.

Syntax

The following syntax is used for the "Form arctangent value" instruction:

GRAPH

```
<Result> := ATAN(<Operand>)
```

Parameters

The following table shows the parameters of the "Form arctangent value" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Tangent value
<Result>	Output	Floating-point numbers	I, Q, M, D, L, P	Size of angle in radians

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := ATAN("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	1.0
Tag_OutValue	+0.785398 (π/4)

The instruction calculates the size of the angle corresponding to the tangent value in the "Tag_InValue" operand. The result of the instruction is stored in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

FRAC: Return fraction

Description

You use the "Return fraction" instruction to determine the decimal places of the value in the operand. For example, if the operand contains the value 123.4567, the result returns the value 0.4567.

The value of the result is invalid if the following condition is met:

- The value of the operand is not a valid floating-point number.

Syntax

The following syntax is used for the "Return fraction" instruction:

GRAPH

```
CALL FRAC <Data type>
     (IN := <Operand>
     RET_VAL => <Operand>
     )
```

Parameters

The following table shows the parameters of the "Return fraction" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Input value, whose decimal places are determined
RET_VAL	Output	Floating-point numbers	I, Q, M, D, L, P	Decimal places of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL FRAC REAL
     (IN := "Tag_InValue"
     RET_VAL => "Tag_OutValue"
     )
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	2.555
Tag_OutValue	0.555

The instruction copies the decimal places from the value of the "Tag_InValue" operand to the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

Move operations

Deserialize: Deserialization

Description

You can use the "Deserialize" instruction to convert the sequential representation of a PLC data type (UDT) back to a PLC data type and to fill its entire contents.

The memory area in which the sequential representation of a PLC data type is located must have the ARRAY of BYTE data type and be declared with standard access. Make sure that there is enough memory space prior to the conversion.

The instruction enables you to convert multiple sequential representations of converted PLC data types back to their original state step-by-step.

Note

If you only want to convert back a single sequential representation of a PLC data type (UDT), you can also use the instruction "TRCV: Receive data via communication connection" directly.

Syntax

The following syntax is used for the "Deserialize" instruction:

GRAPH

```
CALL Deserialize
     (SRC_ARRAY := <Operand>
      DEST_VARIABLE := <Operand>
      POS := <Operand>
      RET_VAL => <Operand>
      )
```

Parameter

The following table shows the parameters of the "Deserialize" instruction:

Parameter	Declaration	Data type	Memory area	Description
SRC_ARRAY	Input	VARIANT	I, Q, D, L	Global data block in which the generated data stream is stored
DEST_VARIABLE	InOut	VARIANT	I, Q, D, L	Tag in which the returned converted PLC data type (UDT) is stored
POS	InOut	DINT	I, Q, M, D, L	Number of bytes that the converted PLC data types use. The POS parameter is calculated zero-based.
RET_VAL	Output	INT	I, Q, M, D, L	Error information

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B0	The memory areas for the SRC_ARRAY and DEST_VARIABLE parameters overlap.
8136	The data block at the DEST_VARIABLE parameter is not a block with standard access.
8150	The VARIANT data type at the SRC_ARRAY parameter contains no value.
8151	Code generation error at SRC_ARRAY parameter
8153	There is not enough free memory available at the SRC_ARRAY parameter.
8250	The VARIANT data type at the DEST_VARIABLE parameter contains no value.
8251	Code generation error at DEST_VARIABLE parameter
8254	Invalid data type at the DEST_VARIABLE parameter

Error code* (W#16#...)	Explanation
8382	The value at the POS parameter is outside the limits of the array.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

GRAPH

```
CALL Deserialize
     (SRC_ARRAY := "Buffer".Field
      DEST_VARIABLE := "Target".Client
      POS := #BufferPos
      RET_VAL => #Error
      )
```

The "Deserialize" instruction deserializes the sequential representation of the customer data from the "Buffer" data block and writes it to the "Target" data block. The number of bytes used by the converted customer data is stored in the "#BufferPos" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

Evaluating errors with output parameter RET_VAL (Page 3152)

Serialize: Serialization

Description

You can use the "Serialize" instruction to convert several PLC data types (UDT) to a sequential representation without losing parts of their structure.

You can use the instruction to temporarily save multiple structured data items from your program in a buffer, which should preferably be in a global data block, and send them to another CPU. The memory area in which the converted PLC data types are stored must have the ARRAY of BYTE data type and be declared with standard access. Make sure that there is enough memory space prior to the conversion.

The operand at the POS parameter contains information about the number of bytes used by the converted PLC data types.

Note

If you want to send a single PLC data type (UDT), you can directly call the instruction "TSEND: Send data via communication connection".

Syntax

The following syntax is used for the "Serialize" instruction:

GRAPH

```
CALL Serialize
     (SRC_VARIABLE := <Operand>
      DEST_ARRAY := <Operand>
      POS := <Operand>
      RET_VAL => <Operand>
      )
```

Parameter

The following table shows the parameters of the "Serialize" instruction:

Parameter	Declaration	Data type	Memory area	Description
SRC_VARIABLE	Input	VARIANT	I, Q, D, L	PLC data type (UDT) that is converted to sequential representation.
DEST_ARRAY	InOut	VARIANT	I, Q, D, L	Data block in which the generated data stream is stored.
POS	InOut	DINT	I, Q, M, D, L	Number of bytes that the converted PLC data types use. The POS parameter is calculated zero-based.
RET_VAL	Output	INT	I, Q, M, D, L	Error information

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B0	The memory areas for the SRC_VARIABLE and DEST_ARRAY parameters overlap.

Error code* (W#16#...)	Explanation
8150	The VARIANT data type at the SRC_VARIABLE parameter contains no value.
8152	Code generation error at SRC_VARIABLE parameter
8236	The data block at the DEST_ARRAY parameter is not a block with standard access.
8250	The VARIANT data type at the DEST_ARRAY parameter contains no value.
8252	Code generation error at DEST_ARRAY parameter
8253	There is not enough free memory available at the DEST_ARRAY parameter.
8254	Invalid data type at the DEST_ARRAY parameter
8382	The value at the POS parameter is outside the limits of the array.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

GRAPH

```
CALL Serialize
     (SRC_VARIABLE := "Source".Client
      DEST_ARRAY := "Buffer".Field
      POS := #BufferPos
      RET_VAL => #Error
      )
```

The "Serialize" instruction serializes the customer data from the "Source" data block and writes them in sequential representation to the "Buffer" data block. The number of bytes used by the sequential representation is stored in the "#BufferPos" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

Evaluating errors with output parameter RET_VAL (Page 3152)

MOVE_BLK: Move block

Description

You can use the "Move block" instruction to move the content of a memory area (source area) to another memory area (destination area). The number of elements to be copied to the destination area is specified with the COUNT parameter. The width of the elements to be moved is defined by the width of the element in the IN parameter.

The instruction can only be executed if the source area and the destination area have the same data type.

The value of the OUT output is invalid if the following condition is met:

- More data is copied than is made available in the IN input or OUT output.

Syntax

The following syntax is used for the "Move block" instruction:

GRAPH

```
CALL MOVE_BLK <Data type>_<Data type>
     (IN := <Operand>
      COUNT := <Operand>
      OUT => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Move block" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN [1]	Input	Binary numbers, integers, floating-point numbers, timers, TOD, LTOD, DATE, CHAR	D, L	The first element of the source area that is being copied
COUNT	Input	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P or constant	Number of elements to be copied from the source area to the destination area
OUT [1]	Output	Binary numbers, integers, floating-point numbers, timers, TOD, LTOD, DATE, CHAR	D, L	The first element of the destination area to which the contents of the source area are being copied
[1] The specified data types can only be used as elements of an ARRAY structure.				

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL MOVE_BLK INT_UINT
     (IN := #a_array[2]
      COUNT := "Tag_Count"
      OUT => #b_array[1]
      )
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	a_array[2]	The data type of the a_array operand is Array [0..5] of INT. It consists of 6 elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the b_array operand is Array [0..6] of INT. It consists of 7 elements of data type INT.

Starting from the third element, the instruction selects three INT elements from the #a_array tag and copies their contents to the #b_array output tag, beginning with the second element.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

MOVE_BLK_VARIANT: Move block

Description

You can use the "Move block" instruction to move the content of a memory area (source area) to another memory area (destination area). You can copy elements of an array to another array of the same data type. The size (number of elements) of the source and destination array may be different. You can copy several elements within an array or individual elements.

If you are using the instruction, the array must not yet be known at the time the block is created, as the source and the destination are transferred using VARIANT.

The counting at the parameters SRC_INDEX and DEST_INDEX always starts with the low limit "0", regardless of the later declaration of the array.

The instruction is not executed if more data is copied than is made available.

If the VARIANT pointer (source or destination) is of the data type BOOL, it must be addressed absolutely and the length specified must be divisible by 8; otherwise the instruction is not executed.

Syntax

The following syntax is used for the "Move block" instruction:

GRAPH

```
CALL MOVE_BLK_VARIANT
     (SRC := <Operand>
      COUNT := <Operand>
      SRC_INDEX := <Operand>
      DEST_INDEX := <Operand>
      DEST := <Operand>
      RET_VAL => <Operand>
      )
```

Parameter

The following table shows the parameters of the "Move block" instruction:

Parameter	Declaration	Data type	Memory area	Description
SRC	Input	VARIANT (array or single element)	L	Source block from which to copy
COUNT	Input	UDINT	I, Q, M, D, L or constant	Number of elements which are copied Set the value at the parameter COUNT to "1", if no Array is specified at the parameter SRC or the parameter DEST.

Parameter	Declaration	Data type	Memory area	Description
SRC_INDEX	Input	DINT	I, Q, M, D, L or constant	• The SRC_INDEX parameter is calculated zero-based. If an Array is specified at parameter SRC, the integer at the SRC_INDEX parameter specifies the first element within the source area from which copying is to take place. This is independent of the declared array limits. • If no Array is set at the SRC parameter or only an individual element of an array is specified, assign the SRC_INDEX parameter the value "0".
DEST_INDEX	Input	DINT	I, Q, M, D, L or constant	• The DEST_INDEX parameter is calculated zero-based. If an Array is specified at the DEST parameter, the integer at the DEST_INDEX parameter specifies the first element within the destination area to which copying is to take place. This is independent of the declared array limits. • If no Array is specified at the DEST parameter, assign the DEST_INDEX parameter the value "0".

Parameter	Declaration	Data type	Memory area	Description
DEST	Output	VARIANT	L	Destination area into which the contents of the source block are copied.
RET_VAL	Output	INT	I, Q, M, D, L	Error information: If an error occurs during the execution of the instruction, an error code is output at the RET_VAL parameter.

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	Data types do not correspond
8151	Access to the SRC parameter is not possible.
8152	The operand at the SRC parameter is not typed.
8153	Code generation error at SRC parameter
8154	The operand at the SRC parameter has the data type BOOL.
8281	The COUNT parameter has an invalid value
8382	The value at the SRC_INDEX parameter is outside the limits of the VARIANT.
8383	The value of the SRC_INDEX parameter is outside the high limit of the array.
8482	The value at the DEST_INDEX parameter is outside the limits of the VARIANT.
8483	The value of the DEST_INDEX parameter is outside the high limit of the array.
8534	The DEST parameter is write protected
8551	Access to the DEST parameter is not possible.
8552	The operand at the DEST parameter is not typed.
8553	Code generation error at DEST parameter
8554	The operand at the DEST parameter has the data type BOOL.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

GRAPH

```
CALL MOVE_BLK_VARIANT
     (SRC := #SrcField
      COUNT := "Tag_Count"
      SRC_INDEX := "Tag_Src_Index"
      DEST_INDEX := "Tag_Dest_Index"
      DEST := #DestField
      RET_VAL := "Tag_Result"
      )
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
SRC	#SrcField	The local operand #SrcField uses a UDT that was still unknown at the time when the block was programmed. (Array [0 to 10] of "MOVE_UDT"
COUNT	Tag_Count	2
SRC_INDEX	Tag_Src_Index	3
DEST_INDEX	Tag_Dest_Index	3
DEST	#DestField	The local operand #DestField uses a UDT that was still unknown at the time when the block was programmed. (Array [10 to 20] of "MOVE_UDT"

2 elements are copied from the source area into the destination area beginning with the fourth element of the array of UDT. The copies are inserted in the array of UDT beginning with the fourth element.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

Switching display formats in the program status (Page 2980)

Evaluating errors with output parameter RET_VAL (Page 3152)

UMOVE_BLK: Move block uninterruptible

Description

You can use the "Move block uninterruptible" instruction to move the content of a memory area (source area) to another memory area (destination area). The instruction cannot be interrupted. The number of elements to be copied to the destination area is specified with the COUNT parameter. The width of the elements to be moved is defined by the width of the element in the IN parameter.

The instruction can only be executed if the source area and the destination area have the same data type.

Note

Move uninterruptible

The move operation cannot be interrupted by other operating system activities. For this reason, the interrupt reaction times of the CPU may be increased during execution of the "Move block uninterruptible" instruction.

The value of the OUT output is invalid if the following condition is met:

- More data is copied than is made available in the IN input or OUT output.

You can use the "Move block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Syntax

The following syntax is used for the "Move block uninterruptible" instruction:

GRAPH

```
CALL UMOVE_BLK <Data type>_<Data type>
      (IN := <Operand>
       COUNT := <Operand>
       OUT => <Operand>
       )
```

Parameters

The following table shows the parameters of the "Move block uninterruptible" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN [1]	Input	Binary numbers, integers, floating-point numbers, timers, TOD, LTOD, DATE, CHAR	D, L	The first element of the source area that is being copied
COUNT	Input	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P or constant	Number of elements to be copied from the source area to the destination area
OUT [1]	Output	Binary numbers, integers, floating-point numbers, timers, TOD, LTOD, DATE, CHAR	D, L	The first element of the destination area to which the contents of the source area are being copied
[1] The specified data types can only be used as elements of an ARRAY structure.				

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL UMOVE_BLK INT_UINT
     (IN := #a_array[2]
      COUNT := "Tag_Count"
      OUT => #b_array[1]
      )
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	a_array[2]	The data type of the a_array operand is Array [0..5] of INT. It consists of 6 elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the b_array operand is Array [0..6] of INT. It consists of 7 elements of data type INT.

Starting from the third element, the instruction selects three INT elements from the #a_array tag and copies their contents to the #b_array output tag, beginning with the second element. The copy operation cannot be interrupted by other operating system activities.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

FILL_BLK: Fill block

Description

You can use the "Fill block" instruction to fill a memory area (destination area) with the value of the IN parameter. The destination area is filled starting from the address specified in the OUT parameter. The number of repeated copy operations is specified with the COUNT parameter. When the instruction is executed, the value of the IN parameter is selected and copied to the destination area as often as specified by the value in the COUNT parameter.

The instruction can only be executed if the source area and the destination area have the same data type.

The value of the OUT output is invalid if the following condition is met:

- More data is copied than is made available in the IN input or OUT output.

Syntax

The following syntax is used for the "Fill block" instruction:

GRAPH

```
CALL FILL_BLK <Data type>_<Data type>
      (IN := <Operand>
       COUNT := <Operand>
       OUT => <Operand>
       )
```

Parameter

The following table shows the parameters of the "Fill block" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN [1]	Input	Binary numbers, integers, floating-point numbers, timers, TOD, LTOD, DATE, CHAR	I, Q, M, D, L, P or constant	Element used to fill the destination area
COUNT	Input	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P or constant	Number of repeated copy operations
OUT [2]	Output	Binary numbers, integers, floating-point numbers, timers, TOD, LTOD, DATE, CHAR	D, L	Address in destination area from which filling starts.
[1] The specified data types can also be used as elements of an ARRAY structure. [2] The specified data types can only be used as elements of an ARRAY structure.				

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL FILL_BLK INT_UINT
     (IN := #a_array[2]
      COUNT := "Tag_Count"
      OUT => #b_array[1]
      )
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	a_array[2]	The data type of the a_array operand is Array [0..5] of INT. It consists of 6 elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the b_array operand is Array [0..6] of INT. It consists of 7 elements of data type INT.

The instruction copies the third element (#a_array[2]) of the #a_array tag three times to the #b_array output tag.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

UFILL_BLK: Fill block uninterruptible

Description

You can use the "Fill block uninterruptible" instruction to fill a memory area (destination area) with the value of the IN parameter. The instruction cannot be interrupted. The destination area is filled starting from the address specified in the OUT parameter. The number of repeated copy operations is specified with the COUNT parameter. When the instruction is executed, the value of the IN parameter is selected and copied to the destination area as often as specified by the value in the COUNT parameter.

The instruction can only be executed if the source area and the destination area have the same data type.

Note

The move operation cannot be interrupted by other operating system activities. For this reason, the interrupt reaction times of the CPU may be increased during execution of the "Fill block uninterruptible" instruction.

The value of the OUT output is invalid if the following condition is met:

- More data is copied than is made available in the IN input or OUT output.

You can use the "Fill block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Syntax

The following syntax is used for the "Fill block uninterruptible" instruction:

GRAPH

```
CALL UFILL_BLK <Data type>_<Data type>
      (IN := <Operand>
       COUNT := <Operand>
       OUT => <Operand>
       )
```

Parameters

The following table shows the parameters of the "Fill block uninterruptible" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN [1]	Input	Binary numbers, integers, floating-point numbers, timers, TOD, LTOD, DATE, CHAR	I, Q, M, D, L, P or constant	Element used to fill the destination area
COUNT	Input	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, P or constant	Number of repeated copy operations
OUT [2]	Output	Binary numbers, integers, floating-point numbers, timers, TOD, LTOD, DATE, CHAR	D, L	Address in destination area from which filling starts.
[1] The specified data types can also be used as elements of an ARRAY structure. [2] The specified data types can only be used as elements of an ARRAY structure.				

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL UFILL_BLK INT_UINT
     (IN := #a_array[2]
      COUNT := "Tag_Count"
      OUT => #b_array[1]
      )
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	a_array[2]	The data type of the a_array operand is Array [0..5] of INT. It consists of 6 elements of data type INT.
COUNT	Tag_Count	3
OUT	b_array[1]	The data type of the b_array operand is Array [0..6] of INT. It consists of 7 elements of data type INT.

The instruction copies the third element (#a_array[2]) of the #a_array tag three times to the #b_array output tag. The copy operation cannot be interrupted by other operating system activities.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

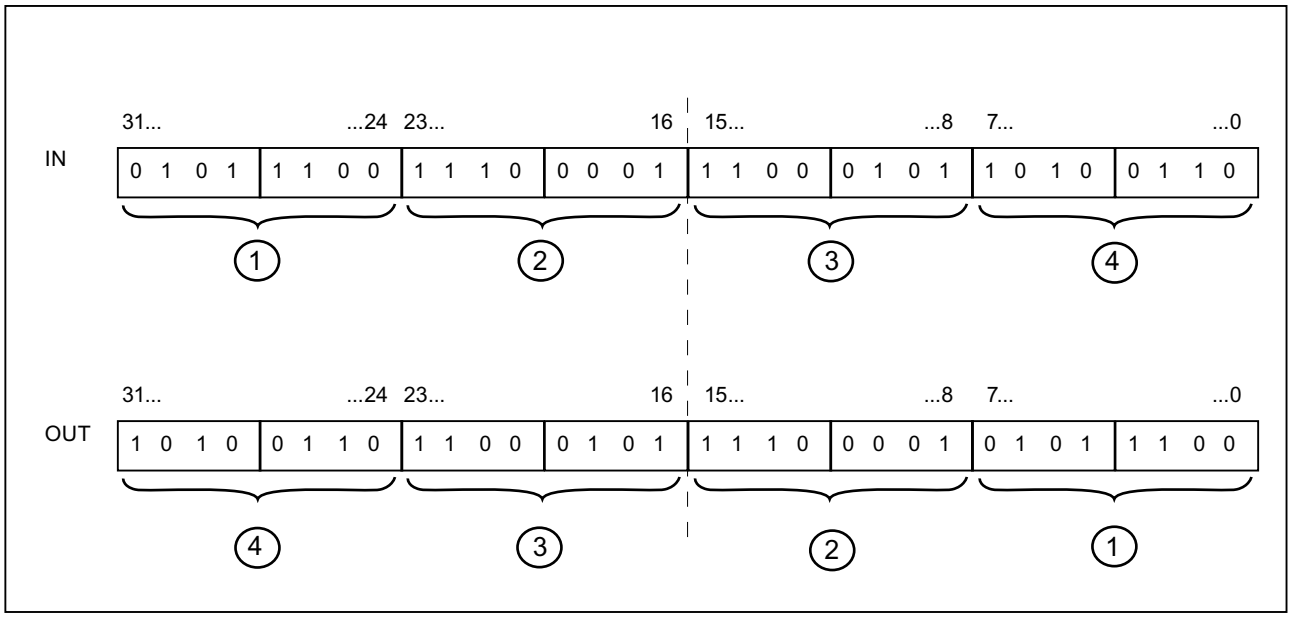
SWAP: Swap

Description

You use the "Swap" instruction to change the order of the bytes of the operand.

Depending on the data type used, you can swap all bytes in accumulator 1 or only the bytes in the right word of accumulator 1.

The following figure shows how the bytes of the operand of data type DWORD are swapped:



Order of bytes in the right word of accumulator 1

For data type WORD, you swap the bytes in the right word of accumulator 1.

The following table shows the contents of accumulator 1 before and after execution of the instruction:

Status	Bytes in accumulator 1			
Before execution	Value A	Value B	Value C	Value D
After execution	Value A	Value B	Value D	Value C

The result of the instruction is saved in the right word of accumulator 1. The bytes in the left word of accumulator 1 are not influenced by the instruction and remain unchanged.

Syntax

The following syntax is used for the "Swap" instruction:

GRAPH

```
<Result> := SWAP(<Operand>)
```

Parameters

The following table shows the parameters of the "Swap" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	WORD, DWORD, LWORD	I, Q, M, D, L, or constant	Operand whose bytes are swapped
<Result>	Output	WORD, DWORD, LWORD	I, Q, M, D, L	Result

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := SWAP("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value			
Tag_InValue	0000	1111	0000	1111
Tag_OutValue	0000	1111	1111	0000

Swapping all bytes in accumulator 1:

For data type DWORD, you swap all bytes in accumulator 1.

The following table shows the contents of accumulator 1 before and after execution of the instruction:

Status	Bytes in accumulator 1			
Before execution	Value A	Value B	Value C	Value D
After execution	Value D	Value C	Value B	Value A

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_InValue	1111	0000	0000	1111	0000	0000	1111	1111
Tag_OutValue	1111	1111	0000	0000	0000	1111	1111	0000

The bytes in the "Tag_InValue" operand are swapped and stored in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

ARRAY DB instructions

ReadFromArrayDB: Read from array data block

Description

You can use the "Read from array data block" instruction to read data from an array DB and write them to a destination area.

An array DB is a data block which consists of exactly one array of [data type]. The elements of the array can be UDT data type or any other elementary data type. The counting of the array always starts with the low limit "0", regardless of the later declaration of the array.

Syntax

The following syntax is used for the "Read from ARRAY data block" instruction:

GRAPH

```
CALL ReadFromArrayDB
     (DB := <Operand>
      INDEX := <Operand>
      VALUE => <Operand>
      RET_VAL => <Operand>
      )
```

Parameter

The following table shows the parameters of the "Read from array data block" instruction:

Parameter	Declaration	Data type	Memory area	Description
DB	Input	DB_ANY	I, Q, M, D, L	Data block that is read
INDEX	Input	DINT	I, Q, M, D, L, P or constant	Element that is read
VALUE	Output	VARIANT	L	Value that is read and output
RET_VAL	Output	INT	I, Q, M, D, L, P	Result of the instruction

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	Data types do not correspond
80B5	Copying process was interrupted
8132	The data block does not exist or is too short
8133	The data block is located in the load memory
8134	The data block is write protected
8135	The array DB contains invalid values
8154	The data type is not correctly declared
8282	The value of the INDEX parameter is outside the limits of the array.
8350	The VARIANT data type contains no value
8352	Code generation error
8353	Code generation error
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

GRAPH

```
CALL ReadFromArrayDB
     (DB := "ArrayDB"
      INDEX := "ArrayDB".THIS[2]
```

GRAPH

```
      VALUE => "TargetField[10]".Data2[1]
      RET_VAL => "TagResult"
      )
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
DB	ArrayDB	The data type of the ArrayDB operand is Array [0 to 10] of INT.
INDEX	ArrayDB.THIS[2]	Third element of the "ArrayDB"
VALUE	TargetField[10].Data2[1]	The data type of the TargetField operand is Array [10 to 20] of "MOVE_UDT".

The third element is read from "ArrayDB" and written to the "TargetField[10].Data2[1]" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

Evaluating errors with output parameter RET_VAL (Page 3152)

WriteToArrayDB: Write to array data block

Description

You can use the "Write to array data block" instruction to write data to an array DB.

An array DB is a data block which consists of exactly one array of [data type]. The elements of the array can be UDT data type or any other elementary data type. The counting of the array always starts with the low limit "0", regardless of the later declaration of the array.

Syntax

The following syntax is used for the "Write to the ARRAY data block" instruction:

GRAPH

```
CALL WriteToArrayDB
      (DB := <Operand>
       INDEX := <Operand>
       VALUE := <Operand>)
       RET_VAL => <Operand>
       )
```

Parameter

The following table shows the parameters of the "Write to array data block" instruction:

Parameter	Declaration	Data type	Memory area	Description
DB	Input	DB_ANY	I, Q, M, D, L	Data block to which data is written
INDEX	Input	DINT	I, Q, M, D, L, P or constant	Element in the DB to which data is written
VALUE	Input	VARIANT	L	Value to be written
RET_VAL	Output	INT	I, Q, M, D, L, P	Result of the instruction

For additional information on valid data types, refer to "See also".

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	Data types do not correspond
80B5	Copying process was interrupted
8132	The data block does not exist or is too short
8133	The data block is located in the load memory
8134	The data block is write protected
8135	The DB is not an array data block.
8282	The value of the INDEX parameter is outside the limits of the array.
8350	The VARIANT data type contains no value
8352	Code generation error
8353	Code generation error
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

GRAPH

```
CALL WriteToArrayDB
     (DB := "ArrayDB"
      INDEX := "ArrayDB".THIS[2]
      VALUE := "SourceField[1]".Data1[6]
```

GRAPH

```
RET_VAL => "TagResult"
)
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
DB	ArrayDB	The data type of the ArrayDB operand is Array [0 to 10] of INT.
INDEX	ArrayDB.THIS[2]	Third element of the "ArrayDB"
VALUE	SourceField[1].Data1[6]	The data type of the SourceField operand is Array [0 to 10] of "MOVE_UDT".

From the "SourceField" operand, the "Data1[6]" element from the second element is written to the "ArrayDB". The third element is written to the "ArrayDB".

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

Evaluating errors with output parameter RET_VAL (Page 3152)

ReadFromArrayDBL: Read from array data block in load memory

Description

You can use the "Read from array data block in load memory" instruction to read data from an array DB in load memory.

An array DB is a data block which consists of exactly one array of [data type]. The elements of the array can be UDT data type or any other elementary data type. The counting of the array always starts with the low limit "0", regardless of the later declaration of the array.

If the array DB is marked with the "Only store in load memory" block attribute, it is only stored in load memory.

The instruction is executed when a positive signal edge is detected at the REQ parameter. The BUSY parameter then has the signal state "1". The instruction is terminated if a negative signal edge is detected at the BUSY parameter. The DONE parameter has the signal state "1" for one program cycle and the read value is output at the VALUE parameter within this cycle. With all other program cycles, the value at the VALUE parameter is not changed.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate

data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Note

The array DB must be located in a data block with the "Optimized" block property.

Syntax

The following syntax is used for the "Read from ARRAY data block in load memory" instruction:

GRAPH

```
CALL ReadFromArrayDBL, "Instance DB"
     (REQ := <Operand>
      DB := <Operand>
      INDEX := <Operand>
      BUSY => <Operand>
      DONE => <Operand>
      ERROR => <Operand>
      VALUE := <Operand>
      )
```

Parameter

The following table shows the parameters of the "Read from array data block in load memory" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = "1": Starting with the reading of the array DB
DB	Input	DB_ANY	I, Q, M, D, L	Data block that is read
INDEX	Input	DINT	I, Q, M, D, L, P or constant	Element that is read
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = "1": The array DB is still being read
DONE	Output	BOOL	I, Q, M, D, L	DONE = "1": The instruction was executed successfully

Parameter	Declaration	Data type	Memory area	Description
VALUE	InOut	VARIANT	D, L	Value that is read and output
ERROR	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output at the ERROR parameter.

For additional information on valid data types, refer to "See also".

ERROR parameter

The following table shows the meaning of the values of the ERROR parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	Data types do not correspond
80B5	Copying process was interrupted
80C3	Too many concurrent calls
8132	The data block does not exist or is too short
8133	The data block is not located in the load memory
8134	The data block is write protected
8135	The DB is not an array data block.
8282	The value of the INDEX parameter is outside the limits of the array.
8350	The VARIANT data type contains no value
8352	Code generation error
8353	Code generation error
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

You can find descriptions of the error codes triggered by the instructions "READ_DBL: Read from data block in load memory" and "WRIT_DBL: Write to data block in load memory" under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL ReadFromArrayDBL, "ReadFromArrayDBL_DB"
     (REQ := "TagReg"
      DB := "ArrayDB"
```

GRAPH

```
INDEX := "ArrayDB".THIS[2]
BUSY => "TagBusy"
DONE => "TagDone"
ERROR => "TagError"
VALUE := "SourceTargetField[1]".Data1[6]
)
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
REQ	TagReq	BOOL
DB	ArrayDB	The data type of the ArrayDB operand is Array [0 to 10] of INT.
INDEX	ArrayDB.THIS[2]	Third element of the "ArrayDB"
VALUE	SourceTargetField[1].Data1[6]	The data type of the SourceTargetField operand is Array [0 to 10] of "MOVE_UDT".
BUSY	TagBusy	BOOL
DONE	TagDone	BOOL

If a positive edge is detected at the "TagReq" operand, the "Read from ARRAY data block in load memory" instruction is executed. The third element is read from "ArrayDB" and written to the "SourceTargetField[1].Data1[6]" operand. As soon as a negative signal edge is detected at the "TagBusy" operand, the instruction is terminated and the value at the VALUE parameter is no longer changed. After the instruction has been processed, the "TagDone" operand the signal state "1".

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

Evaluating errors with output parameter RET_VAL (Page 3152)

WriteToArrayDBL: Write to array data block in load memory

Description

You can use the "Write to array data block in load memory" instruction to write data to an array DB in load memory.

An array DB is a data block which consists of exactly one array of [data type]. The elements of the array can be UDT data type or any other elementary data type. The counting of the array always starts with the low limit "0", regardless of the later declaration of the array.

If the array DB is marked with the "Only store in load memory" block attribute, it is only stored in load memory.

The instruction is executed when a positive signal edge is detected at the REQ parameter. The BUSY parameter then has the signal state "1". If a negative signal edge is detected at the BUSY parameter, the instruction is terminated and the value at the VALUE parameter is written to the data block. The DONE parameter then has the signal state "1" for one program cycle.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Note

The array DB must be located in a data block with the "Optimized" block property.

Syntax

The following syntax is used for the "Write to ARRAY data block in load memory" instruction:

GRAPH

```
CALL WriteToArrayDBL, "Instance DB"
     (REQ := <Operand>
      DB := <Operand>
      INDEX := <Operand>
      VALUE := <Operand>
      BUSY => <Operand>
      DONE => <Operand>
      ERROR => <Operand>
      )
```

Parameter

The following table shows the parameters of the "Write to array data block in load memory" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = "1": Start writing to the array DB
DB	Input	DB_ANY	D	Data block to which data is written
INDEX	Input	DINT	I, Q, M, D, L, P or constant	Element in the DB to which data is written
VALUE	Input	VARIANT	D, L	Value to be written
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = "1": The array DB is still being written

Parameter	Declaration	Data type	Memory area	Description
DONE	Output	BOOL	I, Q, M, D, L	DONE = "1": The instruction was executed successfully
ERROR	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output at the ERROR parameter.

For additional information on valid data types, refer to "See also".

ERROR parameter

The following table shows the meaning of the values of the ERROR parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B4	Data types do not correspond
80B5	Copying process was interrupted
80C3	Too many concurrent calls
8132	The data block does not exist or is too short
8133	The data block is not located in the load memory
8134	The data block is write protected
8135	The DB is not an array data block.
8282	The value of the INDEX parameter is outside the limits of the array.
8350	The VARIANT data type contains no value
8352	Code generation error
8353	Code generation error
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

You can find descriptions of the error codes triggered by the instructions "READ_DBL: Read from data block in load memory" and "WRIT_DBL: Write to data block in load memory" under "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL WriteToArrayDBL, "WriteToArrayDBL_DB"
```

GRAPH

```
(REQ := "TagReg"
 DB := "ArrayDB"
 INDEX := "ArrayDB".THIS[2]
 VALUE := "SourceTargetField[1]".Data1[6]
 BUSY => "TagBusy"
 DONE => "TagDone"
 ERROR => "TagError"
 )
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
REQ	TagReq	BOOL
DB	ArrayDB	The data type of the ArrayDB operand is Array [0 to 10] of INT.
INDEX	ArrayDB.THIS[2]	Third element of the "ArrayDB"
VALUE	SourceTargetField[1].Data1[6]	The data type of the SourceTargetField operand is Array [0 to 10] of "MOVE_UDT".
BUSY	TagBusy	BOOL
DONE	TagDone	BOOL

If a positive edge is detected at the "TagReq" operand, the "Write to ARRAY data block in load memory" instruction is executed. As soon as a negative signal edge is detected at the "TagBusy" operand, the instruction is terminated and the value at the VALUE parameter is written in the third element in the "ArrayDB". After the instruction has been processed, the "TagDone" operand the signal state "1".

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

Evaluating errors with output parameter RET_VAL (Page 3152)

Other

BLKMOV: Move block

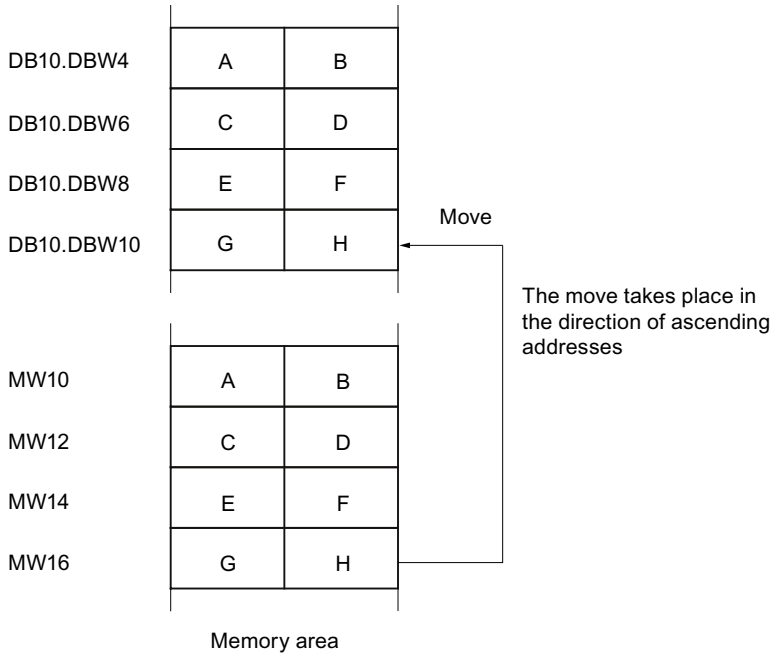
Description

You can use the "Move block" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use VARIANT to define the source and destination area.

Note

The tags of the instruction can only be used in data blocks in which the "Optimized block access" attribute is not set. If the tag with the retentive setting "Set in IDB" has been declared, you can also use the tags "with optimized access".

The following figure shows the principle of the move operation:



Consistency of the source data and the target data

Make sure that the source data remain unchanged during the execution of the "Move block" instruction. Otherwise the consistency of the target data cannot be guaranteed.

Interruptibility

There is no limit to the nesting depth.

Memory areas

You can use the "Move block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image of the inputs
- Process image of the outputs

General rules for moving

The source and destination area must not overlap. If the source and destination area have different lengths, only the length of the smaller area will be moved.

If the source area is less than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is less than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a block of data type BOOL is moved, the tag must be addressed absolutely and the specified length of the area must be divisible by 8, otherwise the instruction cannot be executed.

Rules for moving character strings

You can use the "Move block" instruction to move source and destination areas of data type STRING. If only the source area is STRING data type, the characters will be moved that are actually contained in the character string. Information on the actual and maximum length is also written to the destination area. If the source and destination area are both STRING data type, the current length of the character string in the destination area is set to the number of characters actually moved.

If you want to move the information on the maximum and actual length of a character string, specify the areas in bytes in the SRCBLK and DSTBLK parameters.

Syntax

The following syntax is used for the "Move block" instruction:

GRAPH

```
CALL BLKMOV <Data type>
      (SRCBLK := <Operand>
       RET_VAL => <Operand>
       DSTBLK => <Operand>
       )
```

Parameters

The following table shows the parameters of the "Move block" instruction:

Parameter	Declaration	Data type	Memory area	Description
SRCBLK	Input	VARIANT	L, P	Specifies the memory area to be moved (source area).
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output at the RET_VAL parameter.
DSTBLK	Output	VARIANT	L, P	Specifies the memory area to which the block is to be moved (destination area).

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8152	The data types WSTRING, WCHAR and BOOL are not supported.
8092	The source or target area is only available in the load memory.
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

GRAPH

```
CALL BLKMOV VARIANT
     (SRCBLK := P#M100.0 BYTE 10
      RET_VAL => "Tag_ErrorCode"
      DSTBLK => P#DB1.DBX0.0 BYTE 10
      )
```

The instruction copies 10 bytes starting from MB100 and writes them to DB1. If an error occurs during the move operation, its error code is output in the "Tag_ErrorCode" tag.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

GET_ERR_ID: Get error ID locally (Page 4437)

Evaluating errors with output parameter RET_VAL (Page 3152)

UBLKMOV: Move block uninterruptible

Description

You can use the "Move block uninterruptible" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use VARIANT to define the source and destination area.

The move operation cannot be interrupted by other operating system activities. As a result, the interrupt reaction time of the CPU can increase during the execution of the "Move block uninterruptible" instruction.

Note

The tags of the instruction can only be used in data blocks in which the "Optimized block access" attribute is not set. If the tag with the retentive setting "Set in IDB" has been declared, you can also use the tags "with optimized access".

Memory areas

You can use the "Move block uninterruptible" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image of the inputs
- Process image of the outputs

General rules for moving

The source and destination area must not overlap during the execution of the "Move block uninterruptible" instruction. If the source area is less than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is less than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a source or destination area defined as a formal parameter is smaller than a destination or source area specified in the SRCBLK or DSTBLK parameter, no data is transferred.

If a block of data type BOOL is moved, the tag must be addressed absolutely and the specified length of the area must be divisible by 8, otherwise the instruction cannot be executed.

You can use the "Move block uninterruptible" instruction to move a maximum of 16 KB. Note the CPU-specific restrictions for this.

Rules for moving character strings

You can use the "Move block uninterruptible" instruction to also move source and destination areas of data type STRING. If only the source area is STRING data type, the characters will be moved that are actually contained in the character string. Information on the actual and maximum length are not written in the destination area. If both the source and destination area are of data type STRING, the current length of the character string in the destination area is set to the number of actually moved characters. If areas of the data type STRING are moved, you have to specify "1" as the area length.

Syntax

The following syntax is used for the "Move block uninterruptible" instruction:

GRAPH

```
CALL UBLKMOV <Data type>
     (SRCBLK := <Operand>
      RET_VAL => <Operand>
      DSTBLK => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Move block uninterruptible" instruction:

Parameter	Declaration	Data type	Memory area	Description
SRCBLK	Input	VARIANT	L, P	Specifies the memory area to be moved (source area).
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output at the RET_VAL parameter.
DSTBLK	Output	VARIANT	L, P	Specifies the memory area to which the block is to be moved (destination area).

Parameters RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8152	The data types WSTRING, WCHAR and BOOL are not supported.
8091	The source and destination area are in a data block that is not runtime-relevant.
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

GRAPH

```
CALL UBLKMOV VARIANT
     (SRCBLK := P#M100.0 BYTE 10
      RET_VAL => "Tag_ErrorCode"
      DSTBLK => P#DB1.DBX0.0 BYTE 10
      )
```

The instruction copies 10 bytes starting from MB100 and writes them to DB1. If an error occurs during the move operation, its error code is output in the "Tag_ErrorCode" tag.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

GET_ERR_ID: Get error ID locally (Page 4437)

Evaluating errors with output parameter RET_VAL (Page 3152)

FILL: Fill block

Description

You can use the "Fill block" instruction to fill a memory area (destination area) with the content of another memory area (source area). The "Fill block" instruction moves the content of the source area to the destination area until the destination area is completely written. The move operation takes place in the direction of ascending addresses.

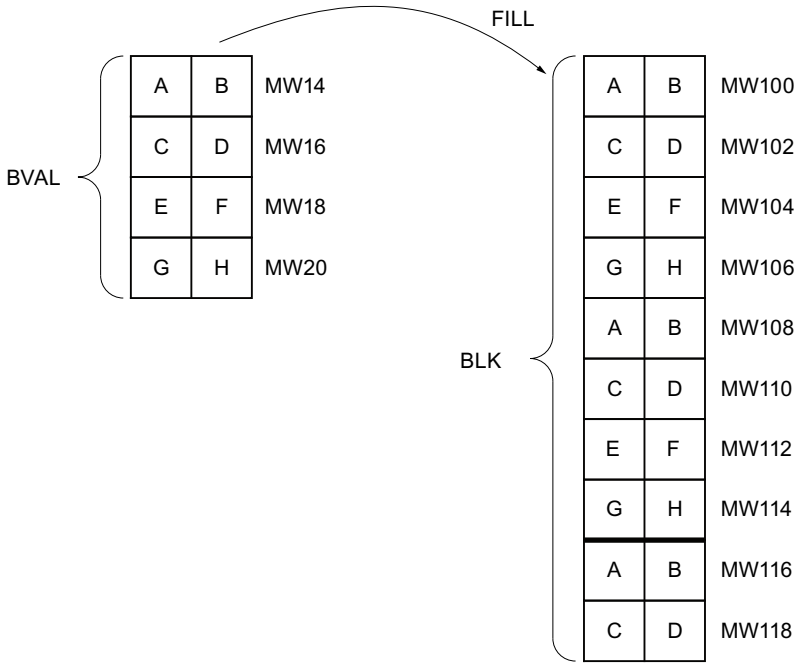
You use VARIANT to define the source and destination area.

Note

The tags of the instruction can only be used in data blocks in which the "Optimized block access" attribute is not set. If the tag with the retentive setting "Set in IDB" has been declared, you can also use the tags "with optimized access".

For blocks with the "Optimized block access" attribute, you can use the "FILL_BLK: Fill block" instruction.

The following figure shows the principle of the move operation:



Example: The contents of the range MW100 to MW118 are to be preassigned with the contents of the memory words MW14 to MW20.

Consistency of the source and the target data

Note that while the instruction "Fill block" is being executed, the source data remains unchanged; otherwise the consistency of the destination data cannot be guaranteed.

Memory areas

You can use the "Fill block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image of the inputs
- Process image of the outputs

General rules for moving

The source and destination area must not overlap. If the destination area to be preset is not an integer multiple of the length of the input parameter BVAL, the destination area is nevertheless written up to the last byte.

If the destination area to be preset is smaller than the source area, the function only copies as much data as can be written to the destination area.

If the destination or source area actually present is smaller than the assigned memory area for the source or destination area (BVAL, BLK parameters), no data is transferred.

If the ANY pointer (source or destination) is of the data type BOOL, it must be addressed absolutely and the length specified must be divisible by 8; otherwise the instruction is not executed.

If the destination area is of data type STRING, the instruction writes the entire string including the administration information.

Rules for moving structures

When you transfer a structure as an input parameter you must bear in mind that the length of a structure is always oriented to an even number of bytes. The structure will need one byte of additional memory space if you declare a structure with an odd number of bytes.

Syntax

The following syntax is used for the "Fill block" instruction:

GRAPH

```
CALL FILL <Data type>
     (BVAL := <Operand>
      RET_VAL => <Operand>
      BLK => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Fill block" instruction:

Parameter	Declaration	Data type	Memory area	Description
BVAL	Input	VARIANT	L, P	Specification of the memory area (source area) with the content used to fill the destination area at the BLK parameter.
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output at the RET_VAL parameter.
BLK	Output	VARIANT	L, P	Specification of the memory area that is filled with the content of the source area.

Parameters RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8152	The data types WSTRING, WCHAR and BOOL are not supported.
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

GRAPH

```
CALL FILL VARIANT
     (BVAL := P#M14.0 WORD 4
      RET_VAL => "Tag_ErrorCode"
      BLK => P#M100.0 WORD 10
      )
```

The instruction copies the source area from MW14 to MW20 and fills the destination area from MW100 to MW118 with the content of the 4 words contained in the memory area in the BVAL parameter.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

GET_ERR_ID: Get error ID locally (Page 4437)

Evaluating errors with output parameter RET_VAL (Page 3152)

Conversion operations

CONVERT: Convert value

Description

You use the "Convert value" instruction to read the content of the operand and convert it according to the assigned data types.

For information on possible conversions, refer to section "Explicit conversion" under "See also".

The value of the result is invalid if any of the following conditions are met:

- Errors, such as an overflow, occur during execution.

Syntax

The following syntax is used for the "Convert value" instruction:

GRAPH

```
<Result> := ???_TO_??? (<Operand>)
```

Parameters

The following table shows the parameters of the "Convert value" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Bit strings, integers, floating-point numbers, CHAR, BCD16, BCD32	I, Q, M, D, L, P or constant	Value to be converted.
<Result>	Output	Bit strings, integers, floating-point numbers, CHAR, BCD16, BCD32	I, Q, M, D, L, P	Result of the conversion

You can select the data type of the instruction from the "<???>" drop-down lists.

Bit strings (BYTE, WORD, DWORD, LWORD) cannot be selected in the instruction box. If you specify an operand of data type BYTE, WORD, DWORD or LWORD at a parameter of the instruction, it is interpreted according to the data type of the input or output parameter. Data type BYTE is then interpreted as USINT, WORD as UINT, DWORD as UDINT, and LWORD as LINT, for example.

Note

The data types DWORD and LWORD can be selected if REAL or LREAL were selected as IN data type.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := INT_TO_DINT("Tag_InValue")
```

The content of the "Tag_InValue" operand is read and converted to an integer (32-bit). The result is stored in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

Explicit conversion (Page 2322)

ROUND: Round numerical value

Description

You use the "Round numerical value" instruction to round the value of the operand to the nearest integer. The instruction interprets the value as a floating-point number and converts it into to the next integer. If the input value is exactly between an even and odd number, the even number is selected.

The value of the result is invalid if the following condition is met:

- Errors such as an overflow occur during execution.

Syntax

The following syntax is used for the "Round numerical value" instruction:

GRAPH

```
<Result> := ROUND(<Operand>)
```

Parameters

The following table shows the parameters of the "Round numerical value" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Input value to be rounded.
<Result>	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result of rounding

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := ROUND("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_InValue	1.50000000	-1.50000000
Tag_OutValue	2	-2

The floating-point number in the "Tag_InValue" operand is rounded to the nearest even integer and output in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

CEIL: Generate next higher integer from floating-point number

Description

You use the "Generate next higher integer from floating-point number" instruction to round the value of the operand to the next higher integer. The instruction interprets the value as a floating-point number and converts this number to the next higher integer. The result can be greater than or equal to the input value.

The value of the result is invalid if the following condition is met:

- Errors such as an overflow occur during execution.

Syntax

The following syntax is used for the "Generate next higher integer from floating-point number" instruction:

GRAPH

```
<Result> := CEIL(<Operand>)
```

Parameters

The following table shows the parameters of the "Generate next higher integer from floating-point number" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Input value as floating-point number
<Result>	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result with next higher integer

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := CEIL("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_InValue	0.50000000	-0.50000000
Tag_OutValue	1	0

The floating-point number in the "Tag_InValue" operand is rounded to the next higher integer and output in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

FLOOR: Generate next lower integer from floating-point number

Description

You use the "Generate next lower integer from floating-point number" instruction to round the value of the operand to the next lower integer. The instruction interprets the value as a floating-point number and converts this number to the next lower integer. The result can be less than or equal to the input value.

The value of the result is invalid if the following condition is met:

- Errors such as an overflow occur during execution.

Syntax

The following syntax is used for the "Generate next lower integer from floating-point number" instruction:

GRAPH

```
<Result> := FLOOR(<Operand>)
```

Parameters

The following table shows the parameters of the "Generate next lower integer from floating-point number" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L, P, or constant	Input value as floating-point number
<Result>	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result with next lower integer

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := FLOOR("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_InValue	0.50000000	-0.50000000
Tag_OutValue	0	-1

The floating-point number in the "Tag_InValue" operand is rounded to the next lower integer and output in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

TRUNC: Truncate numerical value

Description

You use the "Truncate numerical value" instruction to form an integer from the value of the operand. The value is interpreted as a floating-point number. The instruction selects only the integer part of the floating-point number and outputs it without decimal places as the result.

The value of the result is invalid if the following condition is met:

- Errors such as an overflow occur during execution.

Syntax

The following syntax is used for the "Truncate numerical value" instruction:

GRAPH

```
<Result> := TRUNC(<Operand>)
```

Parameters

The following table shows the parameters of the "Truncate numerical value" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Floating-point numbers	I, Q, M, D, L, or constant	Input value as floating-point number
<Result>	Output	Integers, floating-point numbers	I, Q, M, D, L	Result with integer part of the floating-point number

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := TRUNC("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_InValue	1.50000000	-1.50000000
Tag_OutValue	1	-1

The integer part of the floating-point number in the "Tag_InValue" operand is converted to an integer and output in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

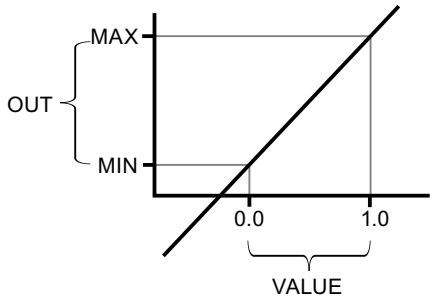
Basic information on GRAPH (Page 2718)

SCALE_X: Scale

Description

You can use the "Scale" instruction to scale the value of the VALUE parameter by mapping it to a specified value range. When the instruction is executed, the floating-point value in the VALUE input is scaled to the value range that was defined by the MIN and MAX parameters. The result of the scaling is an integer, which is stored in the OUT output.

The following figure shows an example of how values can be scaled:



The "Scale" instruction works with the following equation:

OUT = [VALUE * (MAX – MIN)] + MIN

The value of the result is invalid if any of the following conditions are met:

- The value of the MIN parameter is greater than or equal to the value of the MAX parameter.
- The value of a specified floating-point number is outside the range of the normalized numbers according to IEEE-754.
- An overflow occurs.
- The value of the VALUE parameter is NaN (Not a Number = result of an invalid arithmetic operation).

Note

For more information on the conversion of analog values, refer to the respective manual.

Syntax

The following syntax is used for the "Scale" instruction:

GRAPH

```
CALL SCALE_X <Data type>_<Data type>
     (MIN := <Operand>
      VALUE := <Operand>
      MAX := <Operand>
      RET_VAL => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Scale" instruction:

Parameter	Declaration	Data type	Memory area	Description
MIN	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Low limit of the value range
VALUE	Input	Floating-point numbers	I, Q, M, D, L or constant	Value to be scaled. If you enter a constant, you must declare it.
MAX	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	High limit of the value range
RET_VAL	Output	Integers, floating-point numbers	I, Q, M, D, L	Result of scaling

For additional information on valid data types, refer to "See also".

For additional information on declaring constants, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL SCALE_X INT_REAL
     (MIN := "Tag_Minimum"
      VALUE := "Tag_Value"
      MAX := "Tag_Maximum"
      RET_VAL => "Tag_OutputValue"
      )
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
MIN	Tag_Minimum	10
VALUE	Tag_Value	0.5
MAX	Tag_Maximum	30
RET_VAL	Tag_ReturnValue	20

The value of the "Tag_Value" parameter is scaled to the range of values defined by the values of the "Tag_Minimum" and "Tag_Maximum" parameters. The result is stored in the "Tag_ReturnValue" parameter.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

Declaring global constants (Page 2489)

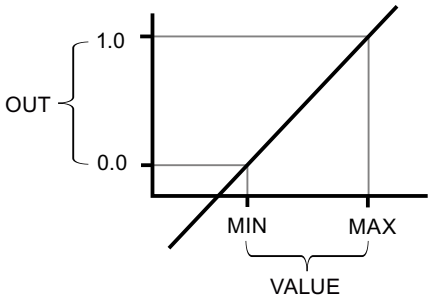
Evaluating errors with output parameter RET_VAL (Page 3152)

NORM_X: Normalize

Description

You can use the "Normalize" instruction to normalize the value of the VALUE parameter by mapping it to a linear scale. You can use the MIN and MAX parameters to define the limits of a value range that is applied to the scale. The result is calculated based on the position of the value to be normalized within this value range and stored as a floating-point number in the OUT parameter. If the value to be normalized is equal to the value of the MIN parameter, the OUT parameter has the value "0.0". If the value to be normalized is equal to the value of the MAX parameter, the OUT parameter has the value "1.0".

The following figure shows an example of how values can be normalized:



The "Normalize" instruction works with the following equation:

OUT = (VALUE – MIN) / (MAX – MIN)

The value of the result is invalid if any of the following conditions are met:

- The value of the MIN parameter is greater than or equal to the value of the MAX parameter.
- The value of a specified floating-point number is outside the range of the normalized numbers according to IEEE-754.
- An overflow occurs.
- The value of the VALUE parameter is NaN (Not a Number = result of an invalid arithmetic operation).

Note

For more information on the conversion of analog values, refer to the respective manual.

Syntax

The following syntax is used for the "Normalize" instruction:

GRAPH

```
CALL NORM_X <Data type>_<Data type>
      (MIN := <Operand>
       VALUE := <Operand>
       MAX := <Operand>
       RET_VAL => <Operand>
       )
```

Parameters

The following table shows the parameters of the "Normalize" instruction:

Parameter	Declaration	Data type	Memory area	Description
MIN [1)]	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Low limit of the value range
VALUE [1)]	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	Value to be normalized
MAX [1)]	Input	Integers, floating-point numbers	I, Q, M, D, L or constant	High limit of the value range
RET_VAL	Output	Floating-point numbers	I, Q, M, D, L	Result of the normalization
[1)] If you use constants in these three parameters, you only need to declare one of them.				

For additional information on valid data types, refer to "See also".

For additional information on declaring constants, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL NORM_X INT_REAL
      (MIN := "Tag_Minimum"
       VALUE := "Tag_Value"
       MAX := "Tag_Maximum"
       RET_VAL => "Tag_OutputValue"
```

GRAPH
)

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
MIN	Tag_Minimum	10
VALUE	Tag_Value	20
MAX	Tag_Maximum	30
RET_VAL	Tag_ReturnValue	0.5

The value of the "Tag_Value" parameter is assigned to the range of values that were defined with the "Tag_Minimum" and "Tag_Maximum" parameters. The tag value in the "Tag_Value" parameter is normalized according the defined value range. The result is stored as a floating-point number in the "Tag_ReturnValue" parameter.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

Declaring global constants (Page 2489)

Evaluating errors with output parameter RET_VAL (Page 3152)

Other

SCALE: Scale

Description

You can use the "Scale" instruction to convert the integer of the IN parameter to a floating-point number that can be scaled in physical units between a low limit and a high limit. You can use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is scaled. The result of the instruction is output at the OUT parameter.

The "Scale" instruction works with the following equation:

OUT = [((FLOAT (IN) – K1)/(K2–K1)) * (HI_LIM–LO_LIM)] + LO_LIM

The values of the constants "K1" and "K2" are determined by the signal state of the BIPOLAR parameter. The following signal states are possible for the BIPOLAR parameter:

- Signal state "1": It is assumed that the value of the IN parameter is bipolar and lies in a value range between -27 648 and 27 648. In this case the constant "K1" has the value "-27 648.0" and the constant "K2" the value "+27 648.0".
- Signal state "0": It is assumed that the value of the IN parameter is unipolar and lies in a value range between 0 and 27 648. In this case the constant "K1" has the value "0.0" and the constant "K2" the value "+27 648.0".

When the value at the IN parameter is greater than the value of the constant "K2", the result of the instruction is set to the value of the high limit (HI_LIM) and an error is output.

When the value at the IN parameter is less than the value of the constant "K1", the result of the instruction is set to the value of the low limit (LO_LIM) and an error is output.

When the specified low limit is greater than the high limit (LO_LIM > HI_LIM), the result is scaled inversely proportional to the input value.

Syntax

The following syntax is used for the "Scale" instruction:

GRAPH

```
CALL SCALE
     (IN := <Operand>
      HI_LIM := <Operand>
      LO_LIM := <Operand>
      BIPOLAR := <Operand>
      RET_VAL => <Operand>
      OUT => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Scale" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	INT	I, Q, M, D, L, P or constant	Input value to be scaled.
HI_LIM	Input	REAL	I, Q, M, D, L, P or constant	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P or constant	Low limit
BIPOLAR	Input	BOOL	I, Q, M, D, L or constant	Indicates whether the value of the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
RET_VAL	Output	WORD	I, Q, M, D, L, P	Error information
OUT	Output	REAL	I, Q, M, D, L, P	Result of the instruction

Parameters RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the constant "K2" or less than the value of the constant "K1"
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

GRAPH

```
CALL SCALE
      (IN := "Tag_InputValue"
       HI_LIM := "Tag_HighLimit"
       LO_LIM := "Tag_LowLimit"
       BIPOLAR := "Tag_Bipolar"
       RET_VAL => "Tag_ErrorCode"
       OUT => "Tag_OutputValue"
       )
```

The following table shows the values of the various operands before execution of the instruction:

Parameter	Operand	Value
IN	Tag_InputValue	22
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	0.0

The following table shows the values of the various operands after execution of the instruction:

Parameter	Operand	Value
IN	Tag_InputValue	22
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1

Parameter	Operand	Value
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	50.03978588

The "Tag_InputValue" operand indicates the value that is to be converted and scaled. The high and low limits are defined by the "Tag_HighLimit" and "Tag_LowLimit" operands. You use operand "Tag_Bipolar" = TRUE to specify that the value of the IN parameter is to be interpreted as bipolar. The result of the instruction is output at the "Tag_OutputValue" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

GET_ERR_ID: Get error ID locally (Page 4437)

Evaluating errors with output parameter RET_VAL (Page 3152)

UNSCALE: Unscale

Description

You can use the "Unscale" instruction to unscale the floating-point number in the IN parameter into physical units between a low limit and a high limit and convert it into an integer. You can use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is unscaled. The result of the instruction is output at the OUT parameter.

The "Unscale" instruction works with the following equation:

OUT = [((IN–LO_LIM)/(HI_LIM–LO_LIM)) * (K2–K1)] + K1

The values of the constants "K1" and "K2" are determined by the signal state of the BIPOLAR parameter. The following signal states are possible for the BIPOLAR parameter:

- Signal state "1": It is assumed that the value of the IN parameter is bipolar and lies in a value range between -27 648 and 27 648. In this case the constant "K1" has the value "-27 648.0" and the constant "K2" the value "+27 648.0".
- Signal state "0": It is assumed that the value of the IN parameter is unipolar and lies in a value range between 0 and 27 648. In this case the constant "K1" has the value "0.0" and the constant "K2" the value "+27 648.0".

When the value at the IN parameter is greater than the value of the constant "HI_LIM", the result of the instruction is set to the value of the constant (K2) and an error is output.

When the value at the IN parameter is less than the value of the constant of the low limit (LO_LIM), the result of the instruction is set to the value of the constant (K1) and an error is output.

Syntax

The following syntax is used for the "Unscale" instruction:

GRAPH

```
CALL UNSCALE
     (IN := <Operand>
      HI_LIM := <Operand>
      LO_LIM := <Operand>
      BIPOLAR := <Operand>
      RET_VAL => <Operand>
      OUT => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Unscale" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	REAL	I, Q, M, D, L, P or constant	Input value to be unscaled to an integer value.
HI_LIM	Input	REAL	I, Q, M, D, L, P or constant	High limit
LO_LIM	Input	REAL	I, Q, M, D, L or constant	Low limit
BIPOLAR	Input	BOOL	I, Q, M, D, L or constant	Indicates whether the value of the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
RET_VAL	Output	WORD	I, Q, M, D, L, P	Error information
OUT	Output	INT	I, Q, M, D, L, P	Result of the instruction

Parameters RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the high limit (HI_LIM) or less than the value of the low limit (LO_LIM).
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

GRAPH

```
CALL UNSCALE
      (IN := "Tag_InputValue"
       HI_LIM := "Tag_HighLimit"
       LO_LIM := "Tag_LowLimit"
       BIPOLAR := "Tag_Bipolar"
       RET_VAL => "Tag_ErrorCode"
       OUT => "Tag_OutputValue"
       )
```

The following table shows the values of the various operands before execution of the instruction:

Parameter	Operand	Value
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	0.0

The following table shows the values of the various operands after execution of the instruction:

Parameter	Operand	Value
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1

Parameter	Operand	Value
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	22

The "Tag_InputValue" operand indicates the value that is to be converted and unscaled. The high and low limits are defined by the "Tag_HighLimit" and "Tag_LowLimit" operands. You use operand "Tag_Bipolar" = TRUE to specify that the value of the IN parameter is to be interpreted as bipolar. The result of the instruction is output at the "Tag_OutputValue" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

GET_ERR_ID: Get error ID locally (Page 4437)

Evaluating errors with output parameter RET_VAL (Page 3152)

Program control operations

Runtime control

ENDIS_PW: Limit and enable password legitimation

Description

You use the "Limit and enable password legitimation" instruction to specify whether passwords configured for the CPU are legitimized or not. In this way, you can prevent legitimized connections, even if the correct password is known.

When you call the instruction and the REQ parameter has the signal state "0", the currently set state is displayed at the output parameters. If changes have been made to the input parameters, these are not transferred to the output parameters.

When you call the instruction and the REQ parameter has the signal state "1", the signal state is taken from the input parameters (F_PWD, FULL_PWD, R_PWD, HMI_PWD). FALSE means that legitimation per password is not permitted; TRUE means the password can be used.

Disable or enabling of the passwords can be allowed or prohibited individually. For example, all passwords can be prohibited, except the fail-safe password. You can thus limit access options to a small user group. The output parameters (F_PWD_ON, FULL_PWD_ON, R_PWD_ON, HMI_PWD_ON) always show the current status of the password use, regardless of the REQ parameter.

Passwords that are not configured must have the signal state TRUE at the input and return the signal state TRUE at the output. The fail-safe password can only be configured for an F-CPU and must therefore always be interconnected with the signal state TRUE in a standard CPU. If the instruction returns an error, the call has no effect and the previous lock is still in effect.

Disabled passwords can be enabled again under the following conditions:

- The CPU is reset to its factory settings.
- The front panel of the S7-1500 CPU supports a dialog that allows you to navigate to the appropriate menu in which the passwords can be enabled again.
- When you call the "Limit and enable password legitimation" instruction, the input parameter of the desired password has the signal state "1".
- Set the mode selector to STOP. The restriction to password legitimation is re-established as soon as the switch is set back to RUN.
- Plugging an empty memory card (transfer module or program card) into an S7-1200 CPU.
- The transition from POWER OFF to POWER ON disables the protection in the S7-1200 CPU. The "Limit and enable password legitimation" instruction must be called once again in the program (e.g. in the startup OB).

Note

The "Limit and enable password legitimation" instruction blocks access from HMI systems if the HMI password has not been enabled.

Note

Existing legitimized connections retain their access rights and you cannot use the "Limit and enable password legitimation" instruction to restrict them.

Preventing unintentional lockout on an S7-1500 CPU

The settings can be made on the front panel of the CPU and the CPU saves the most recent setting.

To prevent accidental lockout, the protection can be disabled by setting the mode selector to STOP with an S7-1500 CPU. The protection is automatically enabled again by setting mode selector to RUN without having to call the "Limit and enable password legitimation" instruction again or taking additional actions on the front panel.

Preventing unintentional lockout on an S7-1200 CPU

Because the S7-1200 CPU does not have a mode switch, protection is disabled with POWER OFF - POWER ON. This means that it is possible and advisable to prevent accidental lockout with the help of specific program sequences within your program.

To do so, program a time control either by means of a cyclic interrupt OB or a timer in the Main OB (OB 1). This gives you the option of calling the "Limit and enable password legitimation" instruction in the respective OB (e.g., OB 1 or OB 35) relatively quickly after a transition from POWER OFF to POWER ON and the associated disabling of the protection. Call the instruction in the startup OB (OB 100) to keep the time window in which the instruction is inactive and there are no limitations for password legitimation relatively small. This procedure will give you the best possible protection from unauthorized access.

If there has been an accidental lockout, you can skip the call in the startup OB (by querying an input parameter, for example) and you have the set time (for example, 10 seconds up to 1 minute) to establish a connection to the CPU before the lock becomes active once again.

If there is no timer in your program code and a lockout has occurred, insert an empty transfer card or an empty program card into the CPU. The empty transfer card or program card deletes the internal load memory of the CPU. You must then download the user program once again from STEP 7 to the CPU.

Procedure for lost passwords with an S7-1200 CPU

If you have lost the password for a password-protected S7-1200 CPU, delete the password-protected program with an empty transfer card or program card. The empty transfer card or program card deletes the internal load memory of the CPU. You can then load a new user program from STEP 7 Basic in the CPU.

WARNING

Inserting an empty transfer card

When you insert a transfer card in a CPU during runtime, the CPU goes to STOP mode. If operating states are unstable, controllers may fail and thus cause the uncontrolled operation of the controlled devices. This leads to an unforeseen operation of the automation system which can cause deadly or serious injuries and/or damage to property.

Once you have pulled the transfer card, the content of the transfer card is available in the internal load memory. Make sure that the card does not include a program at this point.

WARNING

Inserting an empty program card

When you insert a program card in a CPU during runtime, the CPU goes to STOP mode. If operating states are unstable, controllers may fail and thus cause the uncontrolled operation of the controlled devices. This leads to an unforeseen operation of the automation system which can cause deadly or serious injuries and/or damage to property.

Make sure that the program card is empty. The internal load memory is copied to the empty program card. Once you have pulled the previously empty program card, the internal load memory is empty.

You must remove the transfer card or program card before you put the CPU into RUN mode.

Effects of password use on the operating modes

The following table shows what effects the password use by means of the "Limit and enable password legitimation" instruction has on the operating modes and the corresponding user actions.

Action	Password protection by means of the instruction
Basic state after • Operating mode switch to STOP • Reset memory manually (PG, switch, change of MC (Motion Control)) • Reset to factory setting	Not activated (no restrictions)
Basic state after POWER ON	• S7-1200 CPU: The lock is disabled and the instruction must be called once again in the program (e.g., in the startup OB). • S7-1500 CPU: Enabled (if a lock was activated before POWER OFF). The option of not allowing passwords is retentive.
Operating mode transition RUN/STARTUP/HOLD -> STOP (by terminating the instruction, an error or communication) or STOP -> STARTUP/RUN/HOLD	Activated Passwords still cannot be used.

Syntax

The following syntax is used for the "Limit and enable password legitimation" instruction:

GRAPH

```
CALL ENDIS_PW
      (REQ := <Operand>
       F_PWD := <Operand>
       FULL_PWD := <Operand>
       R_PWD := <Operand>
       HMI_PWD := <Operand>
       F_PWD_ON => <Operand>
       FULL_PWD_ON => <Operand>
       R_PWD_ON => <Operand>
       HMI_PWD_ON => <Operand>
       RET_VAL => <Operand>
       )
```

Parameter

The following table shows the parameters of the "Limit and enable password legitimation" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	When the REQ parameter has the signal state "0", the currently set signal state of the passwords is queried.
F_PWD	Input	BOOL	I, Q, M, D, L or constant	Read/write access including fail-safe • F_PWD = "0": Do not allow password • F_PWD = "1": Allow password
FULL_PWD	Input	BOOL	I, Q, M, D, L or constant	Read/write access • FULL_PWD = "0": Do not allow password • FULL_PWD = "1": Allow password
R_PWD	Input	BOOL	I, Q, M, D, L or constant	Read access • R_PWD = "0": Do not allow password • R_PWD = "1": Allow password
HMI_PWD	Input	BOOL	I, Q, M, D, L or constant	HMI access • HMI_PWD = "0": Do not allow password • HMI_PWD = "1": Allow password
F_PWD_ON	Output	BOOL	I, Q, M, D, L	Read/write access including fail-safe • F_PWD_ON = "0": Password not allowed • F_PWD_ON = "1": Password allowed
FULL_PWD_ON	Output	BOOL	I, Q, M, D, L	Read/write access status • FULL_PWD_ON = "0": Password not allowed • FULL_PWD_ON = "1": Password allowed
R_PWD_ON	Output	BOOL	I, Q, M, D, L	Read-access status • R_PWD_ON = "0": Password not allowed • R_PWD_ON = "1": Password allowed
HMI_PWD_ON	Output	BOOL	I, Q, M, D, L	HMI-access status • HMI_PWD_ON = "0": Password not allowed • HMI_PWD_ON = "1": Password allowed
RET_VAL	Output	WORD	I, Q, M, D, L	Error information

For additional information on valid data types, refer to "See also".

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8090	The "Limit and enable password legitimation" instruction is not supported

Error code* (W#16#...)	Explanation
80D0	The password for fail-safe is not configured. The signal state must be TRUE for standard CPUs.
80D1	The read/write access is not configured
80D2	The read access is not configured
80D3	The HMI access is not configured
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

Evaluating errors with output parameter RET_VAL (Page 3152)

RE_TRIGR: Restart cycle monitoring time

Description

You can use the "Restart cycle monitoring time" instruction to restart the maximum cycle time of the CPU. The maximum cycle time then starts over again for the duration you have set in the CPU configuration.

The "Restart cycle monitoring time" instruction can be called in all blocks regardless of the priority.

If the instruction is called in a block with a higher priority, for example, a hardware interrupt, diagnostic interrupt, or cyclic interrupt, the instruction will not be executed.

The "Restart cycle monitoring time" instruction can be called a maximum of 10 times in a program cycle.

Syntax

The following syntax is used for the "Restart cycle monitoring time" instruction:

GRAPH

```
CALL RE_TRIGR ()
```

Parameters

The "Restart cycle monitoring time" instruction has no parameters and supplies no error information.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

STP: Exit program

Description

You use the "Exit program" instruction to set the CPU to STOP mode and therefore to terminate the execution of the program. The effects of changing from RUN to STOP depend on the CPU configuration.

Syntax

The following syntax is used for the "Exit program" instruction:

GRAPH

```
CALL STP ()
```

Parameters

The "Exit program" instruction has no parameters and supplies no error information.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

GET_ERROR: Get error locally

Description

The "Get error locally" instruction is used to query the occurrence of errors within a block. This is usually to access error. If the system reports an error during the processing of the block, detailed information is stored in the operand at the OUT output for the first error to occur during the execution of the block since the last execution of the instruction.

Only operands of the "ErrorStruct" system data type can be specified at the OUT output. The "ErrorStruct" system data type specifies the exact structure in which the information about the error is stored. Using additional instructions, you can evaluate this structure and program an appropriate response. If several errors occur in the block, the error information about the next

error to occur in the instruction is output only after the first error that occurred has been remedied.

Note

The OUT output is only changed if error information is present. To set the output back to "0" after handling an error, you have the following options:

- Declare the tag in the "Temp" section of the block interface.
- Reset the tag to "0" before calling the instruction.

You can find an example of how you can implement the instruction in combination with other troubleshooting options under "See also".

Note

The "Get error locally" instruction enables local error handling within a block. If "Get error locally" is inserted in the program code of a block, any predefined system reactions are ignored if errors occur.

Syntax

The following syntax is used for the "Get error locally" instruction:

GRAPH

```
CALL GET_ERROR
     (OUT => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Get error locally" instruction:

Parameter	Declaration	Data type	Memory area	Description
OUT	Output	ErrorStruct	D, L	Error information

Data type "ErrorStruct"

The following table shows the structure of the "ErrorStruct" data type:

Structure component	Data type	Description
ERROR_ID	WORD	Error ID
FLAGS	BYTE	Shows if an error occurred during a block call. 16#01: Error during a block call 16#00: No error during a block call

Structure component		Data type	Description
REACTION		BYTE	Default reaction: 0: Ignore (write error), 1: Continue with substitute value "0" (read error), 2: Skip instruction (system error)
CODE_ADDRESS		CREF	Information about the address and type of block
	BLOCK_TYPE	BYTE	Type of block where the error occurred: 1: OB 2: FC 3: FB
	CB_NUMBER	UINT	Number of the code block
	OFFSET	UDINT	Reference to the internal memory
MODE		BYTE	Information about the address of an operand
OPERAND_NUMBER		UINT	Operand number of the machine command
POINTER_NUMBER_LOCATION		UINT	(A) Internal pointer
SLOT_NUMBER_SCOPE		UINT	(B) Storage area in internal memory
DATA_ADDRESS		NREF	Information about the address of an operand
	AREA	BYTE	(C) Memory area: L: 16#40 – 4E, 86, 87, 8E, 8F, C0 – CE I: 16#81 Q: 16#82 M: 16#83 DB: 16#84, 85, 8A, 8B Range violations for a directly editable tag of data type DINT: 16#04
	DB_NUMBER	UINT	(D) Number of the data block
	OFFSET	UDINT	(E) Relative address of the operand

Structure component "ERROR_ID"

The following table shows the values that can be output at the structure component "ERROR_ID":

ID* (hexadecimal)	ID* (decimal)	Description
0	0	No error
2503	9475	Invalid pointer
2520	9504	Invalid STRING
2522	9506	Read error: Operand outside the valid range
2523	9507	Write error: Operand outside the valid range
2524	9508	Read error: Invalid operand

ID* (hexadecimal)	ID* (decimal)	Description
2525	9509	Write error: Invalid operand
2528	9512	Read error: Data alignment
2529	9513	Write error: Data alignment
252C	9516	Invalid pointer
2530	9520	Write error: Data block
2533	9523	Invalid pointer used
2534	9524	Block number error FC
2535	9525	Block number error FB
2538	9528	Access error: DB does not exist
2539	9529	Access error: Wrong DB used
253A	9530	Global data block does not exist
253C	9532	Faulty information or the function does not exist
253D	9533	System function does not exist
253E	9534	Faulty information or the function block does not exist
253F	9535	System block does not exist
2550	9552	Access error: DB does not exist
2551	9553	Access error: Wrong DB used
2575	9589	Error in the program nesting depth
2576	9590	Error in the local data distribution
2577	9591	The block property "Parameter passing via registers" is not selected.
25A0	9632	Internal error in TP
25A1	9633	Tag is write-protected
25A2	9634	Invalid numerical value of tag
2942	10562	Read error: Input
2943	10563	Write error: Output
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".		

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

Querying and fixing errors in the program code (Page 258)

Handling program execution errors (Page 2821)

GET_ERR_ID: Get error ID locally

Description

The "Get error ID locally" instruction is used to query the occurrence of errors within a block. This is usually to access error. If the system reports during the block processing errors within the block execution since the last execution of the instruction, the error ID of the first error that occurred in the tag is stored at the RET_VAL output.

Only operands of the WORD data type can be specified at the RET_VAL output. If several errors occur in the block, the error ID of the next error to occur in the instruction is output only after the first error that occurred has been remedied.

Note

The RET_VAL output is only changed if error information is present. To set the output back to "0" after handling an error, you have the following options:

- Declare the tag in the "Temp" section of the block interface.
- Reset the tag to "0" before calling the instruction.

The output of the "Get error ID locally" instruction is only set if error information is present. If one of these conditions is not fulfilled, the remaining program execution is not affected by the "Get error ID locally" instruction.

You can find an example of how you can implement the instruction in combination with other troubleshooting options under "See also".

Note

The "Get error ID locally" instruction enables local error handling within a block. If the "Get error ID locally" instruction is inserted in the program code of a block, any predefined system reactions are ignored if errors occur.

Syntax

The following syntax is used for the "Get error ID locally" instruction:

GRAPH

```
CALL GET_ERR_ID
     (RET_VAL => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Get error ID locally" instruction:

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Output	WORD	I, Q, M, D, L	Error ID

RET_VAL parameter

The following table shows the values that can be output at the RET_VAL parameter:

Error code* (hexadecimal)	Error code* (decimal)	Description
0	0	No error
2503	9475	Invalid pointer
2520	9504	Invalid STRING
2522	9506	Read error: Operand outside the valid range
2523	9507	Write error: Operand outside the valid range
2524	9508	Read error: Invalid operand
2525	9509	Write error: Invalid operand
2528	9512	Read error: Data alignment
2529	9513	Write error: Data alignment
252C	9516	Invalid pointer
2530	9520	Write error: Data block
2533	9523	Invalid pointer used
2534	9524	Block number error FC
2535	9525	Block number error FB
2538	9528	Access error: DB does not exist
2539	9529	Access error: Wrong DB used
253A	9530	Global data block does not exist
253C	9532	Faulty information or the function does not exist
253D	9533	System function does not exist
253E	9534	Faulty information or the function block does not exist
253F	9535	System block does not exist
2550	9552	Access error: DB does not exist
2551	9553	Access error: Wrong DB used
2575	9589	Error in the program nesting depth
2576	9590	Error in the local data distribution
2577	9591	The block property "Parameter passing via registers" is not selected.
25A0	9632	Internal error in TP
25A1	9633	Tag is write-protected
25A2	9634	Invalid numerical value of tag
2942	10562	Read error: Input
2943	10563	Write error: Output
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".		

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

Querying and fixing errors in the program code (Page 258)

Handling program execution errors (Page 2821)

Evaluating errors with output parameter RET_VAL (Page 3152)

INIT_RD: Initialize all retain data

Description

You can use the "Initialize all retain data" instruction to reset the retentive data of all data blocks, bit memories, and SIMATIC timers and counters at the same time. The instruction can only be executed within a startup OB because the execution exceeds the program cycle duration.

Syntax

The following syntax is used for the "Initialize all retain data" instruction:

GRAPH

```
CALL INIT_RD
     (REQ := <Operand>
      RET_VAL => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Initialize all retain data" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	If the input "REQ" has the signal state "1", all retentive data are reset.
RET_VAL	Output	INT	I, Q, M, D, L	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.

For additional information on valid data types, refer to "See also".

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
80B5	The instruction cannot be executed because it was not programmed within a startup OB.
General error information	See also: "GET_ERR_ID: Get error ID locally
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

GRAPH

```
CALL INIT_RD
     (REQ := "Tag_REQ"
      RET_VAL => "Tag_RET_VAL"
      )
```

If the "Tag_REQ" operand has signal state "1", the instruction is executed. All retentive data of all data blocks, bit memories, and SIMATIC timers and counters are reset.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

GET_ERR_ID: Get error ID locally (Page 4437)

Evaluating errors with output parameter RET_VAL (Page 3152)

WAIT: Configure time delay

Description

You use the "Configure time delay" instruction to halt the execution of the program for a set period of time. You indicate the period of time in microseconds in the WT parameter.

You can assign time delays of up to 32767 microseconds (µs). The smallest possible delay time depends on the particular CPU and corresponds to the execution time of the "Configure time delay" instruction.

The execution of the instruction can be interrupted by higher priority events.

The "Configure time delay" instruction supplies no error information.

Syntax

The following syntax is used for the "Configure time delay" instruction:

GRAPH

```
CALL WAIT
     (WT := <Operand>
     )
```

Parameters

The following table shows the parameters of the "Configure time delay" instruction:

Parameter	Declaration	Data type	Memory area	Description
WT	Input	INT	I, Q, M, D, L, P, or constant	Time delay in microseconds (µs)

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

Word logic operations

NOT (INV): Invert

Description

You can use the instruction "Create ones complement" to invert the signal status of the bits of the operand. When the instruction is executed, the value of the operand is linked by EXCLUSIVE OR to a hexadecimal mask (W#16#FFFF for 16-bit numbers or DW#16#FFFF FFFF for 32-bit number). This inverts the signal state of the individual bits that are then output as the result.

Syntax

The following syntax is used for the "Invert" instruction:

GRAPH

```
<Result> := NOT(<Operand>)
```

Parameters

The following table shows the parameters of the instruction "Create ones complement":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	Bit strings, integers	I, Q, M, D, L, P or constant	Input value
<Result>	Output	Bit strings, integers	I, Q, M, D, L, P	Ones complement of the operand value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"TagOut_Value" := NOT("TagIn_Value")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
TagIn_Value	W#16#000F
TagOut_Value	W#16#FFF0

The instruction inverts the signal state of the individual bits at "TagIn_Value" operand and writes the result to the "TagOut_Value" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

DECO: Decode

Description

You use the "Decode" instruction to set a bit in the output value that is specified by the input value .

The "Decode" instruction reads the value of the IN parameter and sets the bit in the OUT parameter whose bit position corresponds to the value read. The other bits in the output value are filled with zeroes. When the value of the IN parameter is greater than 31, a modulo-32 operation is executed.

Syntax

The following syntax is used for the "Decode" instruction:

GRAPH

```
CALL DECO UINT_<DataType>
     (IN := <Operand>
      OUT => <Operand>
     )
```

Parameters

The following table shows the parameters of the "Decode" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	UINT	I, Q, M, D, L, P, or constant	Position of the bit in the output value that is set.
OUT	Output	Bit strings	I, Q, M, D, L, P	Output value

For additional information on valid data types, refer to "See also".

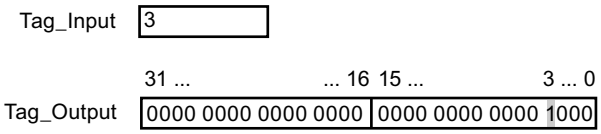
Example

The following example shows how the instruction works:

GRAPH

```
CALL DECO UINT_DWORD
     (IN := "Tag_Input"
      OUT => "Tag_Output"
     )
```

The following figure shows how the instruction functions using specific values:



The instruction reads bit number "3" from the value of the "Tag_Input" operand of the input and sets the third bit in the value of the "Tag_Output" operand of the output.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

ENCO: Encode

Description

You use the "Encode" instruction to read the bit number of the least significant set bit in the input value and output it in the OUT parameter.

The "Encode" instruction selects the least significant bit of the value in the IN parameter and writes its bit number in the operands in the OUT parameter. If the IN parameter has the value DW#16#00000001 or DW#16#00000000, the value "0" is output in the OUT parameter.

Syntax

The following syntax is used for the "Encode" instruction:

GRAPH

```
CALL ENCO <DataType>
      (IN := <Operand>
       OUT => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Encode" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	Bit strings	I, Q, M, D, L, P, or constant	Input value
OUT	Output	INT	I, Q, M, D, L, P	Output value

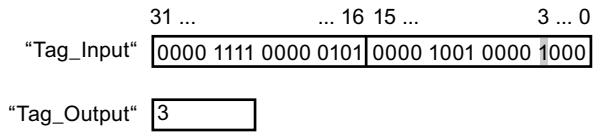
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
CALL ENCO DWORD
      (IN := "Tag_Input"
       OUT => "Tag_Output"
      )
```

The following figure shows how the instruction functions using specific values:



The instruction selects the least significant set bit of the "Tag_Input" tag and writes its bit position "3" to the "Tag_Output" tag.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

SEL: Select

Description

You can use the "Select" instruction to select, dependent on a switch (G input), one of two inputs, IN0 or IN1, and move its content to the OUT output. If the G input has the signal state "0", the value of the IN0 input is moved. If the G input has the signal state "1", the value of the IN1 input is moved to the OUT output.

The instruction is only executed if the tags of all parameters have the same data type.

Syntax

The following syntax is used for the "Select" instruction:

GRAPH

```
CALL SEL <DataType>
    (G := <Operand>
     IN0 := <Operand>
     IN1 := <Operand>
     OUT => <Operand>
    )
```

Parameters

The following table shows the parameters of the "Select" instruction:

Parameter	Declaration	Data type	Memory area	Description
G	Input	BOOL	I, Q, M, D, L or constant	Switch
IN0	Input	Bit strings, integers, floating-point numbers, timers, CHAR, TOD, LTOD, DATE, LDT	I, Q, M, D, L, P or constant	First input value

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	Bit strings, integers, floating-point numbers, TIME, CHAR, TOD, DATE, LDT	I, Q, M, D, L, P or constant	Second input value
OUT	Output	Bit strings, integers, floating-point numbers, TIME, CHAR, TOD, DATE, LDT	I, Q, M, D, L, P	Result

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL SEL CHAR
     (G := "Tag_Input_G"
      IN0 := "Tag_Input0"
      IN1 := "Tag_Input1"
      OUT => "Tag_Output"
     )
```

The following table shows how the instruction functions using specific values:

Parameter	Operand	Value
G	Tag_Input_G	1
IN0	Tag_Input0	W#16#0000
IN1	Tag_Input1	W#16#FFFF
OUT	Tag_Output	W#16#FFFF

Based on the signal state at the "Tag_Input_G" input, the value of the "Tag_Input0" or "Tag_Input1" input is selected and moved to the "Tag_Output" output.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

MUX: Multiplex

Description

You can use the "Multiplex" instruction to copy the content of a selected input to the RET_VAL output. The number of selectable inputs can be extended to a maximum of 32. Numbering starts at IN0 and continues consecutively with each new input. You can use the K parameter to define the input whose content is to be copied to the RET_VAL output. If the value of the K parameter is greater than the number of available inputs, the content of the INELSE parameter is copied to the RET_VAL output.

The "Multiplex" instruction can only be executed if the tags in all inputs and in the RET_VAL output have the same data type. The K parameter is an exception, since only integers can be specified for it.

The value of the RET_VAL parameter is invalid if one of the following conditions is met:

- The value of the K parameter is greater than the number of available inputs.
- Errors occurred during execution of the instruction.

Syntax

The following syntax is used for the "Multiplex" instruction:

GRAPH

```
CALL MUX <DataType>_<DataType>
      (K := <Operand>
      (IN0 := <Operand>
      (IN1 := <Operand>
      (INn := <Operand>
      (INELSE := <Operand>
      (RET_VAL := <Operand>
      )
```

Parameters

The following table shows the parameters of the "Multiplex" instruction:

Parameter	Declaration	Data type	Memory area	Description
K	Input	Integers	I, Q, M, D, L or constant	Specifies the input whose content is to be copied.
IN0	Input	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, LDT	I, Q, M, D, L or constant	First input value

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, LDT	I, Q, M, D, L or constant	Second input value
INn	Input	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, LDT	I, Q, M, D, L or constant	Optional input values
INELSE	Input	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, LDT	I, Q, M, D, L or constant	Specifies the value that is copied when K > n.
RET_VAL	Output	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, DT	I, Q, M, D, L	Output to which the value is copied.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL MUX SINT_SINT
      (K := "Tag_Number"
      (IN0 := "Tag_Value_1"
      (IN1 := "Tag_Value_2"
      (INELSE := "Tag_Value_3"
      (RET_VAL := "Tag_Result"
      )
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
K	Tag_Number	1
IN0	Tag_Value_1	DW#16#00000000
IN1	Tag_Value_2	DW#16#3E4A7D
INELSE	Tag_Value_3	DW#16#FFFF0000
RET_VAL	Tag_Result	DW#16#3E4A7D

Depending on the value of the "Tag_Number" operand, the value of the "Tag_Value_1" input is copied and output to the operand in the "Tag_Result" output.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

Evaluating errors with output parameter RET_VAL (Page 3152)

DEMUX: Demultiplex

Description

You can use the "Demultiplex" instruction to copy the content of the IN input to a selected output. The number of selectable outputs can be extended to a maximum of 32. Numbering starts at OUT0 and continues consecutively with each new output. You can use the K parameter to define the output to which the content of the IN input is to be copied. The other outputs are not changed. If the value of the K parameter is greater than the number of available outputs, the content of the IN input is copied to the OUTELSE parameter.

The "Demultiplex" instruction can only be executed if the tags in the IN input and in all outputs have the same data type. The K parameter is an exception, since only integers can be specified for it.

The value of the OUTELSE output is invalid if one of the following conditions is met:

- The value of the K parameter is greater than the number of available outputs.
- Errors occurred during execution of the instruction.

Syntax

The following syntax is used for the "Demultiplex" instruction:

GRAPH

```
CALL DEMUX <DataType>_<DataType>
      (K := <Operand>
      (IN := <Operand>
      (OUT0 := <Operand>
      (OUT1 := <Operand>
      (OUTn := <Operand>
      (OUTELSE := <Operand>
      )
```

Parameters

The following table shows the parameters of the "Demultiplex" instruction:

Parameter	Declaration	Data type	Memory area	Description
K	Input	Integers	I, Q, M, D, L or constant	Specifies the output to which the input value (IN) is copied.
IN	Input	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, DT, LDT	I, Q, M, D, L or constant	Input value
OUT0	Output	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, DT, LDT	I, Q, M, D, L	First output
OUT1	Output	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, DT, LDT	I, Q, M, D, L	Second output
OUTn	Output	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, DT, LDT	I, Q, M, D, L	Optional outputs
OUTELSE	Output	Binary numbers, integers, floating-point numbers, strings, timers, TOD, LTOD, DATE, DT, LDT	I, Q, M, D, L	Output to which the input value (IN) is copied when K > n.

For additional information on available data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL DEMUX SINT_SINT
      (K := "Tag_Number"
      (IN := "Tag_Value"
      (OUT0 := "Tag_Output_1"
      (OUT1 := "Tag_Output_2"
```

GRAPH

```
(OUTELSE := "Tag_Output_3"
)
```

The following tables show how the instruction works using specific operand values:

Table 9-34 Input values of the "Demultiplex" instruction before network execution:

Parameter	Operand	Values	
K	Tag_Number	1	4
IN	Tag_Value	DW#16#FFFFFFFF	DW#16#3E4A7D

Table 9-35 Output values of the "Demultiplex" instruction after network execution:

Parameter	Operand	Values	
OUT0	Tag_Output_1	Unchanged	Unchanged
OUT1	Tag_Output_2	DW#16#FFFFFFFF	Unchanged
OUTELSE	Tag_Output_3	Unchanged	DW#16#3E4A7D

Based on the value of the "Tag_Number" operand, the value of the IN input is copied to the corresponding output.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

Shift and rotate

SHR: Shift right

Description

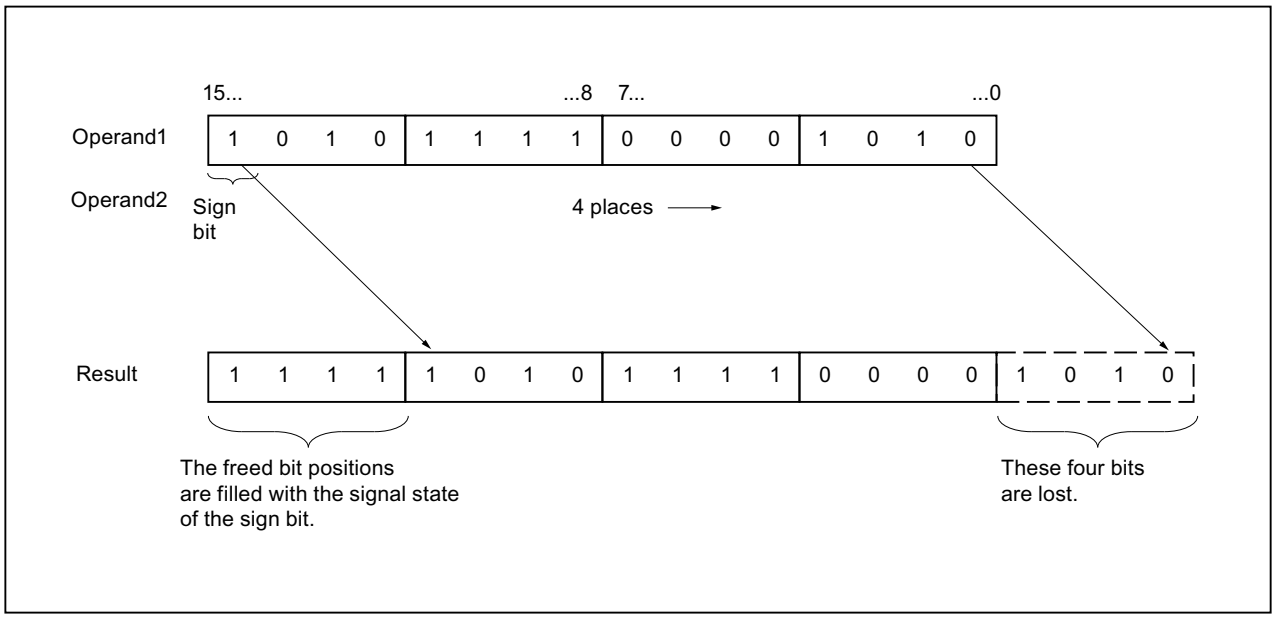
You use the "Shift right" instruction to shift the content of Operand1 to the right bitwise. You use Operand2 to specify the number of bit positions the specified value will be shifted.

If the value of Operand2 is "0", the value of Operand1 is copied to the result.

If the value of Operand2 is greater than the number of available bit positions, the value of Operand1 is shifted to the right by the available number of bit positions.

When unsigned values are shifted, the free bit positions in the left area of Operand1 are filled by zeroes. If the specified value has a sign, the free bit positions are filled with the signal state of the sign bit.

The following figure shows how the content of Operand1 of data type INT is shifted by four bit positions to the right:



Syntax

The following syntax is used for the "Shift right" instruction:

GRAPH

```
<Result> := SHR_<DataType>(<Operand1>,<Operand2>)
```

Parameters

The following table shows the parameters of the "Shift right" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	Bit strings, integers	I, Q, M, D, L, or constant	Value to be shifted
<Operand2>	Input	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, or constant	Number of bit positions by which the value is shifted
<Result>	Output	Bit strings, integers	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "<data type>" drop-down list.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := SHR_INT("Tag_InValue","Tag_Number")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	0011 1111 1010 1111
Tag_Number	3
Tag_OutValue	0000 0111 1111 0101

The content of the "Tag_InValue" operand is shifted three bit positions to the right. The result is output in the "Tag_OutValue" output.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

SHL: Shift left

Description

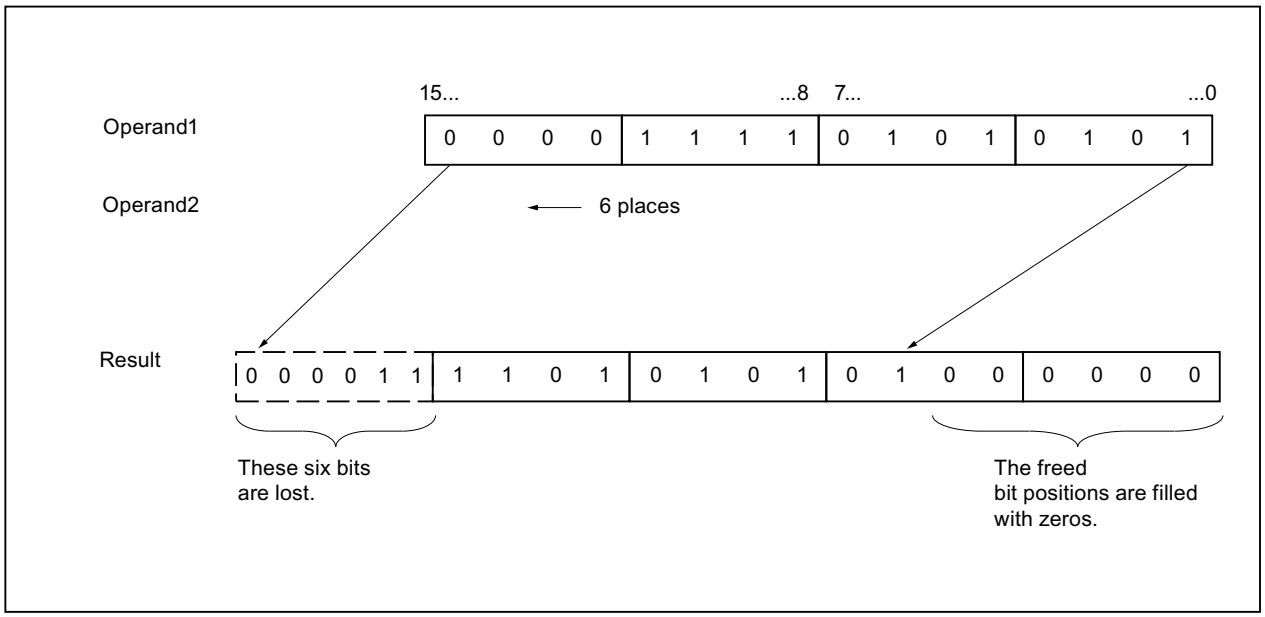
You use the "Shift left" instruction to shift the content of Operand1 to the left bitwise. You use Operand2 to specify the number of bit positions the specified value will be shifted.

If the value of Operand2 is "0", the value of Operand1 is copied to the result.

If the value of Operand2 is greater than the number of available bit positions, the value of Operand1 is shifted to the left by the available number of bit positions.

The bit positions in the right part of Operand1 freed by shifting are filled with zeros.

The following figure shows how the content of Operand1 of data type WORD is shifted by six bit positions to the left:



Syntax

The following syntax is used for the "Shift left" instruction:

GRAPH

```
<Result> := SHL_<DataType>(<Operand1>,<Operand2>)
```

Parameters

The following table shows the parameters of the "Shift left" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	Bit strings, integers	I, Q, M, D, L, or constant	Value to be shifted
<Operand2>	Input	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, or constant	Number of bit positions by which the value is shifted
<Result>	Output	Bit strings, integers	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "<data type>" drop-down list.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := SHL_WORD("Tag_InValue","Tag_Number")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	0011 1111 1010 1111
Tag_Number	4
Tag_OutValue	1111 1010 1111 0000

The content of the "Tag_InValue" operand is shifted four bit positions to the left and output as the result in the Tag_OutValue operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

ROR: Rotate right

Description

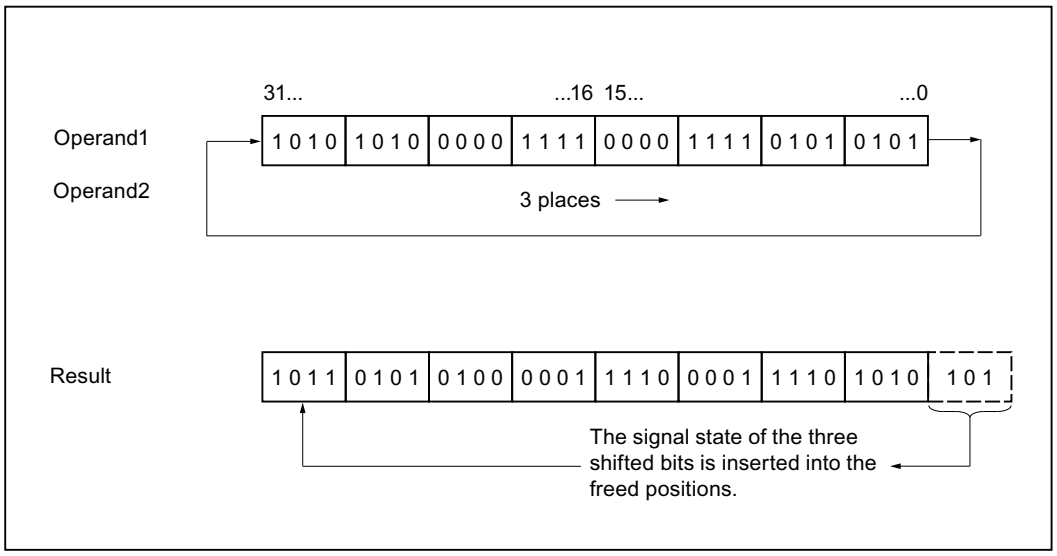
You use the "Rotate right" instruction to rotate the content of Operand1 to the right bitwise. You use Operand2 to specify the number of bit positions the specified value will be rotated.

If the value of Operand2 is "0", the value of Operand1 is copied to the result.

If the value of Operand2 is greater than the number of available bit positions, the value of Operand1 is rotated by the specified number of bit positions.

The bit positions freed by rotating are filled with the bit positions that are pushed out.

The following figure shows how the content of an operand of data type DWORD is rotated three bit positions to the right:



Syntax

The following syntax is used for the "Rotate right" instruction:

GRAPH

```
<Result> := ROR(<Operand1>,<Operand2>)
```

Parameters

The following table shows the parameters of the "Rotate right" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	Bit strings, integers	I, Q, M, D, L, or constant	Value to be rotated
<Operand2>	Input	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, or constant	Number of bit positions by which the value is rotated
<Result>	Output	Bit strings, integers	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := ROR("Tag_InValue","Tag_Number")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	1010 1010 0000 1111 0000 1111 0101 0101
Tag_Number	5
Tag_OutValue	1010 1101 0101 0000 0111 1000 0111 1010

The content of the "Tag_InValue" operand is rotated five bit positions to the right and output as the result in the Tag_OutValue operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

ROL: Rotate left

Description

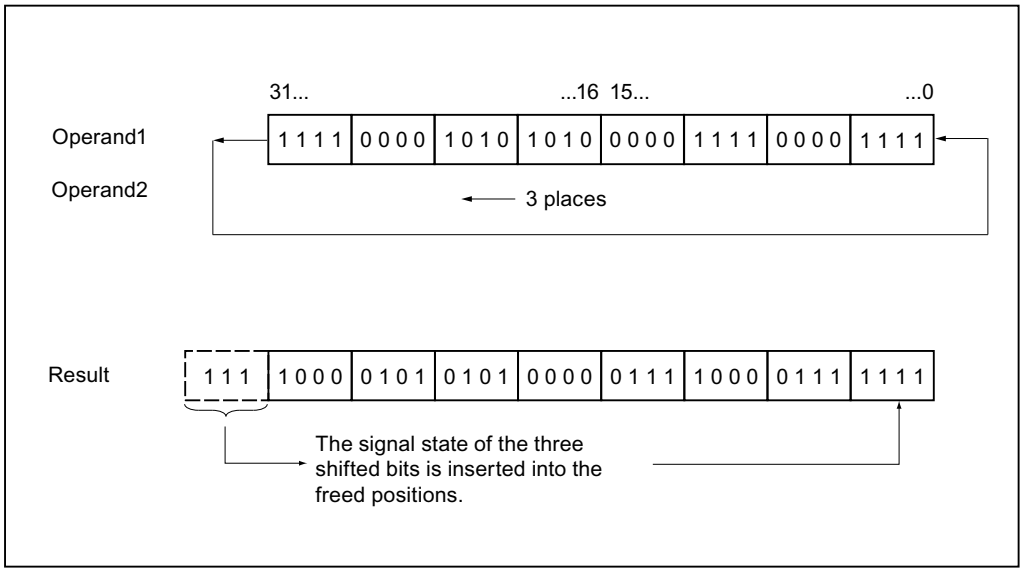
You use the "Rotate left" instruction to rotate the content of Operand1 to the left bitwise. You use Operand2 to specify the number of bit positions the specified value will be rotated.

If the value of Operand2 is "0", the value of Operand1 is copied to the result.

If the value of Operand2 is greater than the number of available bit positions, the value of Operand1 is rotated by the specified number of bit positions.

The bit positions freed by rotating are filled with the bit positions that are pushed out.

The following figure shows how the content of an operand of data type DWORD is rotated three bit positions to the left:



Syntax

The following syntax is used for the "Rotate left" instruction:

GRAPH

```
<Result> := ROL(<Operand1>,<Operand2>)
```

Parameters

The following table shows the parameters of the "Rotate left" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	Bit strings, integers	I, Q, M, D, L, or constant	Value to be rotated
<Operand2>	Input	USINT, UINT, UDINT, ULINT	I, Q, M, D, L, or constant	Number of bit positions by which the value is rotated
<Result>	Output	Bit strings, integers	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := ROL("Tag_InValue","Tag_Number")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	1111 0000 1010 1010 0000 1111 0000 1111
Tag_Number	5
Tag_OutValue	0001 0101 0100 0001 1110 0001 1111 1110

The content of the "Tag_InValue" operand is rotated five bit positions to the left and output as the result in the Tag_OutValue operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

Additional instructions

DRUM: Implement sequencer

Description

The "Implement sequencer" instruction is used to assign the programmed output bits (OUT1 to OUT16) and the output word (OUT_WORD) with the programmed values of the OUT_VAL parameter of the relevant step. The specific step must thereby satisfy the conditions of the programmed enable mask in the S_MASK parameter while the instruction remains at this step. The instruction advances to the next step if the event for the step is true and the programmed time for the current step elapses, or if the value at the JOG parameter changes from "0" to "1". The instruction is reset when the signal state of the RESET parameter changes to "1". The current step is hereby equated to the preset step (DSP).

The amount of time spent on a step is determined by the product of the preset timebase (DTBP) and the preset counter value (S_PRESET) for each step. At the start of a new step, this calculated value is loaded into the DCC parameter, which contains the time remaining for the current step. If, for example the value of the DTBP parameter is "2" and the preset value for the first step is "100" (100 ms), the DCC parameter has the value "200" (200 ms).

A step can be programmed with a time value, an event or both. Steps that have an event bit and the time value "0" advance to the next step as soon as the signal state at the event bit is "1". Steps that are programmed only with a timer value start the time immediately. Steps that are programmed with an event bit and a timer value greater than "0" start the time when the signal state of the event bit is "1". The event bits are initialized with a signal state of "1".

When the sequencer is on the last programmed step (LST_STEP) and the time for this step has expired, the signal state at the Q parameter is set to "1"; otherwise it is set to "0". When the Q parameter is set, the instruction remains on the step until it is reset.

In the configurable mask (S_MASK), you can select the individual bits in the output word (OUT_WORD) and set or reset the output bits (OUT1 to OUT16) by means of the output values (OUT_VAL). If a bit of the configurable mask has signal state "1", the OUT_VAL value sets or resets the corresponding bit. If the signal state of a bit of the configurable mask is "0", the corresponding bit is left unchanged. All the bits of the configurable mask for all 16 steps are initialized with a signal state of "1".

The output bit at the OUT1 parameter corresponds to the least significant bit of the output word (OUT_WORD). The output bit at the OUT16 parameter corresponds to the most significant bit of the output word (OUT_WORD).

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Implement sequencer" instruction:

GRAPH

```
CALL DRUM, <InstanceDB>
     (RESET := <Operand>
      JOG := <Operand>
      DRUM_EN := <Operand>
      LST_STEP := <Operand>
      EVENT1 - 16 := <Operand>
      OUT1 - 16 => <Operand>
      Q => <Operand>
      OUT_WORD => <Operand>
      ERR_CODE => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Implement sequencer" instruction:

Parameter	Declaration	Data type	Memory area	Description
RESET	Input	BOOL	I, Q, M, D, L or constant	The signal state "1" indicates a reset condition.
JOG	Input	BOOL	I, Q, M, D, L or constant	When the signal state changes from "0" to "1", the instruction advances to the next step.
DRUM_EN	Input	BOOL	I, Q, M, D, L or constant	The signal state "1" causes the sequencer to continue counting based on the event and time criteria.
LST_STEP	Input	BYTE	I, Q, M, D, L or constant	Number of the last programmed step.
EVENT(i) 1 ≤ i ≤ 16	Input	BOOL	I, Q, M, D, L or constant	Event bit (i); Initial signal state is "1".
OUT(j), 1 ≤ j ≤ 16	Output	BOOL	I, Q, M, D, L	Output bit (j)

Parameter	Declaration	Data type	Memory area	Description
Q	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that the time for the last step has elapsed.
OUT_WORD	Output	WORD	I, Q, M, D, L, P	Word address to which the sequencer writes the output values.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
JOG_HIS	Static	BOOL	I, Q, M, D, L or constant	JOG parameter history bit
EOD	Static	BOOL	I, Q, M, D, L or constant	The signal state "1" indicates that the time for the last step has elapsed.
DSP	Static	BYTE	I, Q, M, D, L, P or constant	Preset first step of the sequencer (1 to 16)
DSC	Static	BYTE	I, Q, M, D, L, P or constant	Current step of the sequencer
DCC	Static	DWORD	I, Q, M, D, L, P or constant	Remaining processing time for the current step
DTBP	Static	WORD	I, Q, M, D, L, P or constant	Preset timebase of the sequencer
PREV_TIME	Static	DWORD	I, Q, M, D, L or constant	System time of the previous call
S_PRESET	Static	ARRAY of WORD	I, Q, M, D, L or constant	Preset counter value for each step [1 to 16] where 1 clock pulse = 1 ms.
OUT_VAL	Static	ARRAY of BOOL	I, Q, M, D, L or constant	Output values for each step [1 to 16, 0 to 15].
S_MASK	Static	ARRAY of BOOL	I, Q, M, D, L or constant	Configurable mask for each step [1 to 16, 0 to 15]. Initial signal states are "1".

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

ERR_CODE*	Explanation
W#16#0000	No error
W#16#000B	The value of the LST_STEP parameter is less than 1 or greater than 16.
W#16#000C	The value of the DSC parameter is less than 1 or greater than the value of the LST_STEP parameter.
W#16#000D	The value of the DSP parameter is less than 1 or greater than the value of the LST_STEP parameter.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

In the following example the instruction advances from step 1 to step 2. The output bits (OUT1 to OUT16) and the output word (OUT_WORD) are set according to the mask configured for step 2 and the values of the OUT_VAL parameter.

Note

You can initialize static parameters in the data block.

GRAPH

```
CALL DRUM, "DRUM_DB"
     (RESET := "Tag_Reset"
      JOG := "Tag_Input_Jog"
      DRUM_EN := "Tag_Input_Drum_EN"
      LST_STEP := "Tag_Number_LastStep"
      EVENTn := "MyTag_Event_n"
      OUTn => "MyTag_Output_n"
      Q => "Tag_Output_Q"
      OUT_WORD => "Tag_OutputWord"
      ERR_CODE => "Tag_ErrorCode"
      )
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for initializing the input parameters:

Parameter	Operand	Address	Value
RESET	Tag_Reset	M0.0	FALSE
JOG	Tag_Input_JOG	M0.1	FALSE
DRUM_EN	Tag_Input_Drum_EN	M0.2	TRUE
LST_STEP	Tag_Number_LastStep	MB1	B#16#08
EVENT2	MyTag_Event_2	M20.0	FALSE
EVENT4	MyTag_Event_4	M20.1	FALSE
EVENT6	MyTag_Event_6	M20.2	FALSE
EVENT8	MyTag_Event_8	M20.3	FALSE
EVENT10	MyTag_Event_10	M20.4	FALSE
EVENT12	MyTag_Event_12	M20.5	FALSE
EVENT14	MyTag_Event_14	M20.6	FALSE
EVENT16	MyTag_Event_16	M20.7	FALSE

The following values are saved in the "DRUM_DB" instance data block of the instruction:

Parameter	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSP	DBB13	W#16#0001
DSC	DBB14	W#16#0001
DCC	DBD16	DW#16#0000000A
DTBP	DBW20	W#16#0001
S_PRESET[1]	DBW26	W#16#0064
S_PRESET[2]	DBW28	W#16#00C8
OUT_VAL[1,0]	DBX58.0	TRUE
OUT_VAL[1,1]	DBX58.1	TRUE
OUT_VAL[1,2]	DBX58.2	TRUE
OUT_VAL[1,3]	DBX58.3	TRUE
OUT_VAL[1,4]	DBX58.4	TRUE
OUT_VAL[1,5]	DBX58.5	TRUE
OUT_VAL[1,6]	DBX58.6	TRUE
OUT_VAL[1,7]	DBX58.7	TRUE
OUT_VAL[1,8]	DBX59.0	TRUE
OUT_VAL[1,9]	DBX59.1	TRUE
OUT_VAL[1,10]	DBX59.2	TRUE
OUT_VAL[1,11]	DBX59.3	TRUE
OUT_VAL[1,12]	DBX59.4	TRUE
OUT_VAL[1,13]	DBX59.5	TRUE
OUT_VAL[1,14]	DBX59.6	TRUE
OUT_VAL[1,15]	DBX59.7	TRUE

Parameter	Address	Value
OUT_VAL[2,0]	DBX60.0	FALSE
OUT_VAL[2,1]	DBX60.1	FALSE
OUT_VAL[2,2]	DBX60.2	FALSE
OUT_VAL[2,3]	DBX60.3	FALSE
OUT_VAL[2,4]	DBX60.4	FALSE
OUT_VAL[2,5]	DBX60.5	FALSE
OUT_VAL[2,6]	DBX60.6	FALSE
OUT_VAL[2,7]	DBX60.7	FALSE
OUT_VAL[2,8]	DBX61.0	FALSE
OUT_VAL[2,9]	DBX61.1	FALSE
OUT_VAL[2,10]	DBX61.2	FALSE
OUT_VAL[2,11]	DBX61.3	FALSE
OUT_VAL[2,12]	DBX61.4	FALSE
OUT_VAL[2,13]	DBX61.5	FALSE
OUT_VAL[2,14]	DBX61.6	FALSE
OUT_VAL[2,15]	DBX61.7	FALSE
S_MASK[2,0]	DBX92.0	FALSE
S_MASK[2,1]	DBX92.1	TRUE
S_MASK[2,2]	DBX92.2	TRUE
S_MASK[2,3]	DBX92.3	TRUE
S_MASK[2,4]	DBX92.4	TRUE
S_MASK[2,5]	DBX92.5	FALSE
S_MASK[2,6]	DBX92.6	TRUE
S_MASK[2,7]	DBX92.7	TRUE
S_MASK[2,8]	DBX93.0	FALSE
S_MASK[2,9]	DBX93.1	FALSE
S_MASK[2,10]	DBX93.2	TRUE
S_MASK[2,11]	DBX93.3	TRUE
S_MASK[2,12]	DBX93.4	TRUE
S_MASK[2,13]	DBX93.5	TRUE
S_MASK[2,14]	DBX93.6	FALSE
S_MASK[2,15]	DBX93.7	TRUE

The output parameters are set to the following values before the execution of the instruction:

Parameter	Operand	Address	Value
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#FFFF
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	TRUE
OUT3	MyTag_Output_3	M4.2	TRUE
OUT4	MyTag_Output_4	M4.3	TRUE
OUT5	MyTag_Output_5	M4.4	TRUE

Parameter	Operand	Address	Value
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	TRUE
OUT8	MyTag_Output_8	M4.7	TRUE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	TRUE
OUT12	MyTag_Output_12	M5.3	TRUE
OUT13	MyTag_Output_13	M5.4	TRUE
OUT14	MyTag_Output_14	M5.5	TRUE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	TRUE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Address	Value
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	FALSE
OUT3	MyTag_Output_3	M4.2	FALSE
OUT4	MyTag_Output_4	M4.3	FALSE
OUT5	MyTag_Output_5	M4.4	FALSE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	FALSE
OUT8	MyTag_Output_8	M4.7	FALSE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	FALSE
OUT12	MyTag_Output_12	M5.3	FALSE
OUT13	MyTag_Output_13	M5.4	FALSE
OUT14	MyTag_Output_14	M5.5	FALSE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	FALSE
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#4321
ERR_CODE	Tag_ErrorCode	MW10	W#16#0000

The following values are saved in the "DRUM_DB" instance data block of the instruction:

Parameter	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE

Parameter	Address	Value
DSC	DBB14	W#16#0002
DCC	DBD16	DW#16#000000C8

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

DCAT: Discrete control-timer alarm

Description

You can use the "Discrete control-timer alarm" to count the time from the point at which the CMD parameter issues the command to open or close. The time is accumulated until the preset time (PT) is exceeded or the information is received that the device was opened or closed (O_FB or C_FB) within the specified time. If the preset time is exceeded before the information on the opening or closing of the device is received, the corresponding alarm is activated. If the signal state of the command input changes before the preset time, the time is restarted.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The "Discrete control-timer alarm" instruction has the following reactions to the input conditions:

- When the signal state at the CMD parameter changes from "0" to "1", the signal states of the Q, CMD_HIS, ET (only if ET < PT), OA, and CA parameters are influenced as follows:
  - The Q and CMD_HIS parameters are set to "1".
  - The ET, OA, and CA parameters are reset to "0".
- When the signal state at the parameter CMD changes from "1" to "0", the parameters Q, ET (only if ET < PT), OA, CA and CMD_HIS are reset to "0".
- When the signal state at the CMD and CMD_HIS parameters is "1" and the O_FB parameter is set to "0", the time difference (ms) since the last execution of the instruction is added to the value of the parameter ET. If the value of the ET parameter exceeds the value of the PT parameter, the signal state at the OA parameter is set to "1". If the value of the ET parameter does not exceed the value of the PT parameter, the signal state of the OA parameter is reset to "0". The value of the parameter CMD_HIS is reset to the value of the parameter CMD.
- If the signal state of the CMD, CMD_HIS, and O_FB parameters are set to "1" and the C_FB parameter has the value "0", the signal state of the OA parameter is set to "0". The value of the ET parameter is set to the value of the PT parameter. If the signal state of the O_FB parameter changes to "0", the alarm is set the next time the instruction is executed. The value of the CMD_HIS parameter is set to the value of the CMD parameter.

- If the CMD, CMD_HIS, and C_FB parameters have the value "0", the time difference (ms) since the last execution of the instruction is added to the value of the ET parameter. If the value of the ET parameter exceeds the value of the PT parameter, the signal state of the CA parameter is reset to "1". If the value at the parameter PT is not exceeded, the parameter CA has the signal state "0". The value of the CMD_HIS parameter is set to the value of the CMD parameter.
- If the CMD, CMD_HIS, and O_FB parameters have the signal state "0" and the C_FB parameter is set to "1", the CA parameter is set to "0". The value of the ET parameter is set to the value of the PT parameter. If the signal state of the C_FB parameter changes to "0", the alarm is set the next time the instruction is executed. The value of the CMD_HIS parameter is set to the value of the CMD parameter.
- If the O_FB and C_FB parameters simultaneously have the signal state "1", the signal states of both alarm outputs are set to "1".

The "Discrete control-timer alarm" instruction returns no error information.

Syntax

The following syntax is used for the "Discrete control-timer alarm" instruction:

GRAPH

```
CALL DCAT, <InstanceDB>
     (CMD := <Operand>
      O_FB := <Operand>
      C_FB := <Operand>
      Q => <Operand>
      OA => <Operand>
      CA => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Discrete control-timer alarm" instruction:

Parameter	Declaration	Data type	Memory area	Description
CMD	Input	BOOL	I, Q, M, D, L or constant	The signal state "0" indicates the "close" command. The signal state "1" indicates the "open" command.
O_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when closing
Q	Output	BOOL	I, Q, M, D, L	Shows the status of the CMD parameter
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening

Parameter	Declaration	Data type	Memory area	Description
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
ET	Static	DINT	D, L or constant	Currently elapsed time, where one clock pulse = 1 ms.
PT	Static	DINT	D, L or constant	Preset timer value, where one clock pulse = 1 ms.
PREV_TIME	Static	DWORD	D, L or constant	Previous system time
CMD_HIS	Static	BOOL	D, L or constant	CMD history bit

For additional information on valid data types, refer to "See also".

Example

In the following example the CMD parameter changes from "0" to "1". After the execution of the instruction, the Q parameter is set to "1" and the two alarm outputs OA and CA have the signal state "0". The CMD_HIS parameter of the instance data block is set to the signal state "1" and the ET parameter is reset to "0".

Note

You can initialize static parameters in the data block.

GRAPH

```
CALL DCAT, "DCAT_DB"
      (CMD := "Tag_Input_CMD"
       O_FB := "Tag_Input_O_FB"
       C_FB := "Tag_Input_C_FB"
       Q => "Tag_Output_Q"
       OA => "Tag_Output_OA"
       CA => "Tag_Output_CA"
       )
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameter	Operand	Value
CMD	Tag_Input_CMD	TRUE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
Q	Tag_Output_Q	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the "DCAT_DB" instance data block of the instruction:

Parameter	Address	Value
ET	DBD4	L#12
PT	DBD8	L#222
CMD_HIS	DBX16.0	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
Q	Tag_Output_Q	TRUE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the "DCAT_DB" instance data block of the instruction:

Parameter	Address	Value
ET	DBD4	L#0
CMD_HIS	DBX16.0	TRUE

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

MCAT: Motor control-timer alarm

Description

The instruction "Motor control-timer alarm" is used to accumulate the time from the point in time as of which one of the command inputs (open or close) is activated. The time is accumulated until the preset time is exceeded or the relevant feedback input indicates that the device has executed the requested operation within the specified time. If the preset time is exceeded before the feedback is received, the corresponding alarm is triggered.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Execution of the "Motor control-timer alarm" instruction

The following table shows the reactions of the "Motor control-timer alarm" instruction•to the various input conditions:

Input parameters								Output parameters								
ET	O_HIS	C_HIS	O_CMD	C_CMD	S_CMD	O_FB	C_FB	OO	CO	OA	CA	ET	O_HIS	C_HIS	Q	Status
X	1	1	X	X	X	X	X	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	X	1	1	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	1	X	X	0	0	0	0	X	0	0	1	Stop
X	X	X	1	1	X	X	X	0	0	0	0	X	0	0	1	Stop
X	0	X	1	0	0	X	X	1	0	0	0	0	1	0	1	Start opening
<PT	1	0	X	0	0	0	X	1	0	0	0	INC	1	0	1	Open
X	1	0	X	0	0	1	0	0	0	0	0	PT	1	0	1	Opened
>=PT	1	0	X	0	0	0	X	0	0	1	0	PT	1	0	0	Opening alarm
X	X	0	0	1	0	X	X	0	1	0	0	0	0	1	1	Start closing
<PT	0	1	0	X	0	X	0	0	1	0	0	INC	0	1	1	Close
X	0	1	0	X	0	0	1	0	0	0	0	PT	0	1	1	Closed
>=PT	0	1	0	X	0	X	0	0	0	0	1	PT	0	1	0	Closing alarm
X	0	0	0	0	0	X	X	0	0	0	0	X	0	0	1	Stopped

Legend:	
INC	Add the time difference (ms) since the last processing of the FB to ET
PT	PT is set to the same value as ET
X	Cannot be used
<PT	ET < PT
>=PT	ET >= PT

If the input parameters O_HIS and C_HIS both have the signal state "1", they are immediately set to the signal state "0". In this case, the last row in the table (X) mentioned above is valid. Because it is therefore not possible to check whether the input parameters O_HIS and C_HIS have the signal state "1", the output parameters are set as follows in this case:

OO = FALSE
CO = FALSE
OA = FALSE
CA = FALSE
ET = PT
Q = TRUE

Syntax

The following syntax is used for the "Motor control-timer alarm" instruction:

GRAPH

```
CALL MCAT, <InstanceDB>
```

GRAPH

```
(O_CMD := <Operand>
 C_CMD := <Operand>
 S_CMD := <Operand>
 O_FB := <Operand>
 C_FB := <Operand>
 OO => <Operand>
 CO => <Operand>
 OA => <Operand>
 CA => <Operand>
 Q => <Operand>
 )
```

Parameters

The following table shows the parameters of the "Motor control-timer alarm" instruction:

Parameter	Declaration	Data type	Memory area	Description
O_CMD	Input	BOOL	I, Q, M, D, L or constant	"Open" command input
C_CMD	Input	BOOL	I, Q, M, D, L or constant	"Close" command input
S_CMD	Input	BOOL	I, Q, M, D, L or constant	"Stop" command input
O_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L or constant	Feedback input when closing
OO	Output	BOOL	I, Q, M, D, L	"Open" output
CO	Output	BOOL	I, Q, M, D, L	"Close" output
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
Q	Output	BOOL	I, Q, M, D, L	The signal state "0" indicates an error condition
ET	Static	DINT	D, L or constant	Currently elapsed time, where one clock pulse = 1 ms
PT	Static	DINT	D, L or constant	Preset timer value, where one clock pulse = 1 ms
PREV_TIME	Static	DWORD	D, L or constant	Previous system time
O_HIS	Static	BOOL	D, L or constant	"Open" history bit
C_HIS	Static	BOOL	D, L or constant	"Close" history bit

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.

GRAPH

```
CALL MCAT, "MCAT_DB"
      (O_CMD := "Tag_Input_O_CMD"
       C_CMD := "Tag_Input_C_CMD"
       S_CMD := "Tag_Input_S_CMD"
       O_FB := "Tag_Input_O_FB"
       C_FB := "Tag_Input_C_FB"
       OO => "Tag_OutputOpen"
       CO => "Tag_OutputClosed"
       OA => "Tag_Output_OA"
       CA => "Tag_Output_CA"
       Q => "Tag_Output_Q"
       )
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameter	Operand	Value
O_CMD	Tag_Input_O_CMD	TRUE
C_CMD	Tag_Input_C_CMD	FALSE
S_CMD	Tag_Input_S_CMD	FALSE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
OO	Tag_OutputOpen	FALSE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	FALSE

The following values are saved in the "MCAT_DB" instance data block of the instruction:

Parameter	Address	Value
ET	DBD4	L#2
PT	DBD8	L#22

Parameter	Address	Value
O_HIS	DBX16.0	TRUE
C_HIS	DBX16.1	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OO	Tag_OutputOpen	TRUE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	TRUE

The following values are saved in the "MCAT_DB" instance data block of the instruction:

Parameter	Address	Value
ET	DBD4	L#0
O_HIS	DBX16.0	TRUE
CMD_HIS	DBX16.1	FALSE

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

IMC: Compare input bits with the bits of a mask

Description

You can use the "Compare input bits with the bits of a mask" instruction to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bits of a mask. Up to 16 steps with masks can be programmed. The value of the IN_BIT0 parameter is compared with the value of the CMP_VAL[x,0] mask, where "x" indicates the step number. In the CMP_STEP parameter, you specify the step number of the mask that is used for the comparison. All programmed values are compared in the same manner. Unprogrammed input bits or unprogrammed bits of the mask have the default signal state FALSE.

If a match is found in the comparison, the signal state of the OUT parameter is set to "1". Otherwise, the OUT parameter is set to "0".

If the value of the CMP_STEP parameter is greater than 15, the instruction is not executed. An error message is output at the ERR_CODE parameter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Compare input bits with the bits of a mask" instruction:

GRAPH

```
CALL IMC, <InstanceDB>
     (IN_BIT0 - 15 := <Operand>
      CMP_STEP := <Operand>
      OUT => <Operand>
      ERR_CODE => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Compare input bits with the bits of a mask" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN_BIT0	Input	BOOL	I, Q, M, D, L or constant	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L or constant	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L or constant	Input bit 2 is compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L or constant	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L or constant	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L or constant	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L or constant	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L or constant	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L or constant	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L or constant	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L or constant	Input bit 10 is compared with bit 10 of the mask.

Parameter	Declaration	Data type	Memory area	Description
IN_BIT11	Input	BOOL	I, Q, M, D, L or constant	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L or constant	Input bit 12 is compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L or constant	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L or constant	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L or constant	Input bit 15 is compared with bit 15 of the mask.
CMP_STEP	Input	BYTE	I, Q, M, D, L, P or constant	The step number of the mask used for the comparison.
OUT	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that a match was found. The signal state "0" indicates that no match was found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
CMP_VAL	Static	ARRAY OF WORD	I, Q, M, D, L or constant	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

For additional information on valid data types, refer to "See also".

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000A	The value of the CMP_STEP parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

In the following example all 16 input bits are compared with the mask for step 2. The signal state of the OUT parameter is set to TRUE, since the input bits match the mask for step 2.

Note

You can initialize static parameters in the data block.

GRAPH

```
CALL IMC, "IMC_DB"
     (IN_BIT0 - 15 := "Tag_Input_BITn"
      CMP_STEP := "Tag_CMP_STEP"
      OUT => "Tag_Output"
      ERR_CODE => "Tag_ErrorCode"
      )
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameter	Operand	Value
IN_BIT0	Tag_Input_BIT0	TRUE
IN_BIT1	Tag_Input_BIT1	TRUE
IN_BIT2	Tag_Input_BIT2	FALSE
IN_BIT3	Tag_Input_BIT3	TRUE
IN_BIT4	Tag_Input_BIT4	TRUE
IN_BIT5	Tag_Input_BIT5	FALSE
IN_BIT6	Tag_Input_BIT6	TRUE
IN_BIT7	Tag_Input_BIT7	TRUE
IN_BIT8	Tag_Input_BIT8	FALSE
IN_BIT9	Tag_Input_BIT9	TRUE
IN_BIT10	Tag_Input_BIT10	TRUE
IN_BIT11	Tag_Input_BIT11	FALSE
IN_BIT12	Tag_Input_BIT12	TRUE
IN_BIT13	Tag_Input_BIT13	TRUE
IN_BIT14	Tag_Input_BIT14	FALSE
IN_BIT15	Tag_Input_BIT15	TRUE
CMP_STEP	Tag_CMP_STEP	B#16#02
OUT	Tag_Output	FALSE
ERR_CODE	Tag_ErrorCode	W#16#0000

The following values for the mask of step 2 are saved in the "IMC_DB" instance data block of the instruction:

Parameter	Address	Value
CMP_VAL [2,0]	DBX12.0	TRUE
CMP_VAL [2,1]	DBX12.1	TRUE
CMP_VAL [2,2]	DBX12.2	FALSE
CMP_VAL [2,3]	DBX12.3	TRUE
CMP_VAL [2,4]	DBX12.4	TRUE
CMP_VAL [2,5]	DBX12.5	FALSE

Parameter	Address	Value
CMP_VAL [2,6]	DBX12.6	TRUE
CMP_VAL [2,7]	DBX12.7	TRUE
CMP_VAL [2,8]	DBX13.0	FALSE
CMP_VAL [2,0]	DBX13.1	TRUE
CMP_VAL [2,10]	DBX13.2	TRUE
CMP_VAL [2,11]	DBX13.3	FALSE
CMP_VAL [2,12]	DBX13.4	TRUE
CMP_VAL [2,13]	DBX13.5	TRUE
CMP_VAL [2,14]	DBX13.6	FALSE
CMP_VAL [2,15]	DBX13.7	TRUE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OUT	Tag_Output	TRUE
ERR_CODE	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

SMC: Compare scan matrix

Description

You can use the "Compare scan matrix" instruction to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bit of the comparison masks for each step. Processing starts at step 1 and is continued until the last programmed step (LAST) or until a match is found. The input bit of the IN_BIT0 parameter is compared with the value of the CMP_VAL[x,0] mask, where "x" indicates the step number. All programmed values are compared in the same manner. If a match is found the signal state of the OUT parameter is set to "1" and the step number with the matching mask is written in the OUT_STEP parameter. Unprogrammed input bits or unprogrammed bits of the mask have the default signal state FALSE. If more than one step has a matching mask, only the first one found is indicated in the OUT_STEP parameter. If no match is found, the signal state of the OUT parameter is set to "0". In this case the value at the OUT_STEP parameter is greater by "1" than the value of the LAST parameter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single

instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Compare scan matrix" instruction:

GRAPH

```
CALL SMC, <InstanceDB>
     (IN_BIT0 - 15 := <Operand>
      OUT => <Operand>
      OUT_STEP => <Operand>
      ERR_CODE => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Compare scan matrix" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN_BIT0	Input	BOOL	I, Q, M, D, L or constant	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L or constant	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L or constant	Input bit 2 is compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L or constant	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L or constant	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L or constant	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L or constant	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L or constant	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L or constant	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L or constant	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L or constant	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L or constant	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L or constant	Input bit 12 is compared with bit 12 of the mask.

Parameter	Declaration	Data type	Memory area	Description
IN_BIT13	Input	BOOL	I, Q, M, D, L or constant	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L or constant	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L or constant	Input bit 15 is compared with bit 15 of the mask.
OUT	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that a match was found. The signal state "0" indicates that no match was found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
OUT_STEP	Output	BYTE	I, Q, M, D, L, P	Contains the step number with the matching mask or the step number which is greater by "1" than the value of the LAST parameter, provided no match is found.
LAST	Static	BYTE	I, Q, M, D, L, P or constant	Specifies the step number of the last step to be scanned for a matching mask.
CMP_VAL	Static	ARRAY OF WORD	I, Q, M, D, L or constant	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

For additional information on valid data types, refer to "See also".

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000E	The value of the LAST parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

In this example, all 16 input bits are compared with the mask for steps 0 through 5 until a match is found. Only the masks for steps 0 through 2 are scanned, since the mask for step 2 matches the input bit.

Note

You can initialize static parameters in the data block.

GRAPH

```
CALL SMC, "SMC_DB"
     (IN_BIT0 - 15 := "Tag_Input_BITn"
      OUT => "Tag_Output"
      OUT_STEP => "Tag_Output_STEP"
      ERR_CODE => "Tag_ErrorCode"
      )
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameter	Operand	Value
IN_BIT0	Tag_Input_BIT0	TRUE
IN_BIT1	Tag_Input_BIT1	TRUE
IN_BIT2	Tag_Input_BIT2	FALSE
IN_BIT3	Tag_Input_BIT3	TRUE
IN_BIT4	Tag_Input_BIT4	TRUE
IN_BIT5	Tag_Input_BIT5	FALSE
IN_BIT6	Tag_Input_BIT6	TRUE
IN_BIT7	Tag_Input_BIT7	TRUE
IN_BIT8	Tag_Input_BIT8	FALSE
IN_BIT9	Tag_Input_BIT9	TRUE
IN_BIT10	Tag_Input_BIT10	TRUE
IN_BIT11	Tag_Input_BIT11	FALSE
IN_BIT12	Tag_Input_BIT12	TRUE
IN_BIT13	Tag_Input_BIT13	TRUE
IN_BIT14	Tag_Input_BIT14	FALSE
IN_BIT15	Tag_Input_BIT15	TRUE
OUT	Tag_Output	FALSE
OUT_STEP	Tag_Output_STEP	B#16#00
ERR_CODE	Tag_ErrorCode	W#16#0000

The following values for the mask of step 2 are saved in the "SMC_DB" instance data block of the instruction:

Parameter	Address	Value
CMP_VAL [2,0]	DBX12.0	TRUE
CMP_VAL [2,1]	DBX12.1	TRUE
CMP_VAL [2,2]	DBX12.2	FALSE
CMP_VAL [2,3]	DBX12.3	TRUE
CMP_VAL [2,4]	DBX12.4	TRUE
CMP_VAL [2,5]	DBX12.5	FALSE
CMP_VAL [2,6]	DBX12.6	TRUE
CMP_VAL [2,7]	DBX12.7	TRUE
CMP_VAL [2,8]	DBX13.0	FALSE
CMP_VAL [2,0]	DBX13.1	TRUE
CMP_VAL [2,10]	DBX13.2	TRUE
CMP_VAL [2,11]	DBX13.3	FALSE
CMP_VAL [2,12]	DBX13.4	TRUE
CMP_VAL [2,13]	DBX13.5	TRUE
CMP_VAL [2,14]	DBX13.6	FALSE
CMP_VAL [2,15]	DBX13.7	TRUE
LAST	DB84	B#16#05

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OUT	Tag_Output	TRUE
OUT_STEP	Tag_Output_STEP	B#16#02
ERR_CODE	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

LEAD_LAG: Lead and lag algorithm

Description

You can use the "Lead and lag algorithm" instruction to process signals with an analog tag. The gain value at the GAIN parameter must be greater than zero. The result of the "Lead and lag algorithm" instruction is calculated using the following equation:

$$OUT = \left[\frac{LG_TIME}{LG_TIME + SAMPLE_T}\right] PREV_OUT + GAIN \left[\frac{LD_TIME + SAMPLE_T}{LG_TIME + SAMPLE_T}\right] IN - GAIN \left[\frac{LD_TIME}{LG_TIME + SAMPLE_T}\right] * PREV_IN$$

The "Lead and lag algorithm" instruction supplies plausible results only when processing is in fixed program cycles. The same units must be specified at the parameters LD_TIME, LG_TIME and SAMPLE_T. At LG_TIME > 4 + SAMPLE_T, the instruction approaches the following function:

OUT = GAIN * ((1 + LD_TIME * s) / (1 + LG_TIME * s)) * IN

If the value of the GAIN parameter is less than or equal to zero, the calculation is not performed and an error information is output at the ERR_CODE parameter.

You can use the "Lead and lag algorithm" instruction in conjunction with loops as a compensator in dynamic feed-forward control. The instruction consists of two operations. The "Lead" operation shifts the phase of the OUT output so that the output leads the input. The "Lag" operation, on the other hand, shifts the output so that the output lags behind the input. Because the "Lag" operation is equivalent to an integration, it can be used as a noise suppressor or as a low-pass filter. The "Lead" operation is equivalent to a differentiation and can therefore be used as a high-pass filter. The two operations together (Lead and Lag) result in the output phase lagging behind the input at lower frequencies and leading it at higher frequencies. This means that the "Lead and lag algorithm" instruction can be used as a band pass filter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Lead and lag algorithm" instruction:

GRAPH

```
CALL LEAD_LAG, <InstanceDB>
     (IN := <Operand>
      SAMPLE_T := <Operand>
      OUT => <Operand>
      ERR_CODE => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Lead and lag algorithm" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	REAL	I, Q, M, D, L, P or constant	The input value of the current sample time (cycle time) to be processed.
SAMPLE_T	Input	INT	I, Q, M, D, L, P or constant	Sample time
OUT	Output	REAL	I, Q, M, D, L, P	Result of the instruction
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
LD_TIME	Static	REAL	I, Q, M, D, L, P or constant	Lead time in the same unit as sample time.
LG_TIME	Static	REAL	I, Q, M, D, L, P or constant	Lag time in the same unit as sample time
GAIN	Static	REAL	I, Q, M, D, L, P or constant	Gain as % / % (the ratio of the change in output to a change in input as a steady state).
PREV_IN	Static	REAL	I, Q, M, D, L, P or constant	Previous input
PREV_OUT	Static	REAL	I, Q, M, D, L, P or constant	Previous output

For additional information on valid data types, refer to "See also".

ERR_CODE parameter

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
0009	The value of the GAIN parameter is less than or equal to zero.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.

GRAPH

```
CALL LEAD_LAG, "LEAD_LAG_DB"
      (IN := "Tag_Input"
       SAMPLE_T := "Tag_Input_SAMPLE_T"
       OUT => "Tag_Output_Result"
       ERR_CODE => "Tag_ErrorCode"
       )
```

The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input parameters:

Parameter	Operand	Value
IN	Tag_Input	2.0
SAMPLE_T	Tag_Input_SAMPLE_T	10

The following values are saved in the "LEAD_LAG_DB" instance data block of the instruction:

Parameter	Address	Value
LD_TIME	DBD12	2.0
LG_TIME	DBD16	2.0
GAIN	DBD20	1.0
PREV_IN	DBD24	6.0
PREV_OUT	DBD28	6.0

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OUT	Tag_Output_Result	2.0

The following values are saved in the "LEAD_LAD_DB" instance data block of the instruction:

Parameter	Operand	Value
PREV_IN	DBD24	2.0
PREV_OUT	DBD28	2.0

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

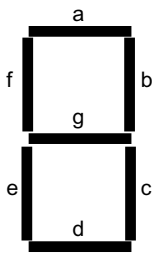
Basic information on GRAPH (Page 2718)

SEG: Create bit pattern for seven-segment display

Description

The "Create bit pattern for seven-segment display" instruction converts each of the four hexadecimal digits in the specified source word (IN) into four equivalent codes for a 7-segment display and writes these to the double word of the output (OUT).

The relationship between the input hexadecimal values and the output bit patterns is as follows:

Digit	-gfedcba	Display	Seven-segment display
0000	00111111	0	
0001	00000110	1	
0010	01011011	2	
0011	01001111	3	
0100	01100110	4	
0101	01101101	5	
0110	01111101	6	
0111	00000111	7	
1000	01111111	8	
1001	01100111	9	
1010	01110111	A	
1011	01111100	B	
1100	00111001	C	
1101	01011110	D	
1110	01111001	E	
1111	01110001	F	

Syntax

The following syntax is used for the "Create bit pattern for seven-segment display" instruction:

GRAPH

```
CALL SEG
```

GRAPH

```
(IN := <Operand>
 OUT => <Operand>
 )
```

Parameters

Parameter	Declaration	Data type	Memory area	Description
IN	Input	WORD	I, Q, M, D, L, P, or constant	Source word with four hexadecimal digits
OUT	Output	DWORD	I, Q, M, D, L, P	Destination bit pattern with four bytes

The "Create bit pattern for seven-segment display" does not recognize any error conditions.

Example

The following example shows how the instruction works:

GRAPH

```
CALL SEG
     (IN := "Tag_Input"
      OUT => "Tag_Output"
      )
```

The following table shows how the instruction functions using specific values:

Parameter	Operand	Value	
Hexadecimal	Binary		
IN	Tag_Input	W#16#1234	0001 0010 0011 0100
OUT	Tag_Output	DW#16065B4F66	000 00110 0101 1011 0100 1111 0110 0110 Display: 1234

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

BCDCPL: Create tens complement

Description

You can use the "Create tens complement" instruction to create the tens complement of a seven-digit BCD number specified in the IN parameter. This instruction uses the following mathematical formula to calculate:

10000000 (as BCD)

– 7-digit BCD value

Tens complement (as BCD)

Syntax

The following syntax is used for the "Lead and lag algorithm" instruction:

GRAPH

```
CALL BCDCPL
     (IN := <Operand>
      ERR_CODE => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Create tens complement" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	Bit strings	I, Q, M, D, L, P or constant	7-digit BCD number
ERR_CODE	Output	DWORD	I, Q, M, D, L, P	Result of the instruction

Example

The following example shows how the instruction works:

GRAPH

```
CALL BCDCPL
     (IN := "Tag_Input"
      ERR_CODE => "Tag_Output"
      )
```

The following table shows how the instruction functions using specific values:

Parameter	Operand	Value*
IN	Tag_Input	DW#16#01234567
ERR_CODE	Tag_Output	DW#16#08765433
*The error codes can be displayed as integer or hexadecimal value in the program editor. For information on toggling display formats, refer to "See also".		

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

Switching display formats in the program status (Page 2980)

BITSUM: Count number of set bits

Description

You use the "Count number of set bits" instruction to count the number of bits of an operand that is set to the signal state "1". The operand whose bits are to be counted is specified in the IN parameter. The result of the instruction is output in the RET_VAL parameter.

Syntax

The following syntax is used for the "Count number of set bits" instruction:

GRAPH

```
CALL BITSUM
      (IN := <Operand>
       RET_VAL => <Operand>
       )
```

Parameters

The following table shows the parameters of the "Count number of set bits" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	DWORD	I, Q, M, D, L, P, or constant	Operand whose set bits are counted
RET_VAL	Output	INT	I, Q, M, D, L, P	Number of set bits

Example

The following example shows how the instruction works:

GRAPH

```
CALL BITSUM
      (IN := "Tag_Input"
       RET_VAL => "Tag_Output"
       )
```

The following table shows how the instruction functions using specific values:

Parameter	Operand	Value*
IN	Tag_Input	DW#16#12345678
RET_VAL	Tag_Output	W#16#000D (13 bits)
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".		

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

LAD instructions Comparator operations

General LAD instructions

Description

Different LAD instructions are available to you when programming in GRAPH.

The following table shows an overview of the instructions and their availability:

LAD instruction	Permanent instruction	Transition/ Supervision/ Interlock
General		
Insert network	x	
Empty box	x	x
Open branch	x	x
Close branch	x	x
-|: Insert input	x	
Bit logic operations		
--| |--: Normally open contact (Page 3156)	x	x

LAD instruction	Permanent instruction	Transition/ Supervision/ Interlock
--| / |--: Normally closed contact (Page 3157)	x	x
--|NOT|--: Invert RLO (Page 3158)	x	x
---( )---: Assignment (Page 3159)	x	
Negate assignment (Page 3160)	x	
---(R)---: Reset output (Page 3160)	x	
---(S)---: Set output (Page 3161)	x	
SET_BF: Set bit field (Page 3162)	x	
RESET_BF: Reset bit field (Page 3163)	x	
SR: Set/reset flip-flop (Page 3164)	x	
RS: Reset/set flip-flop (Page 3165)	x	
--|P|--: Scan operand for positive signal edge (Page 3167)	x	
--|N|--: Scan operand for negative signal edge (Page 3168)	x	
--(P)--: Set operand on positive signal edge (Page 3169)	x	
--(N)--: Set operand on negative signal edge (Page 3171)	x	
P_TRIG: Scan RLO for positive signal edge (Page 3172)	x	
N_TRIG: Scan RLO for negative signal edge (Page 3173)	x	
R_TRIG: Set tag on positive signal edge (Page 3174)	x	
F_TRIG: Set tag on negative signal edge (Page 3176)	x	
Timer operations		
IEC timers		
TP: Generate pulse (Page 3177)	x	
TON: Generate on-delay (Page 3179)	x	
TOF: Generate off-delay (Page 3182)	x	
TONR: Time accumulator (Page 3184)	x	
---(TP)---: Start pulse timer (Page 3187)	x	
---(TON)---: Start on-delay timer (Page 3189)	x	
---(TOF)---: Start off-delay timer (Page 3191)	x	
---(TONR)---: Time accumulator (Page 3193)	x	
---(RT)---: Reset timer (Page 3195)	x	
---(PT)---: Load time duration (Page 3196)	x	
SIMATIC timers (S7-1500)		
S_PULSE: Assign pulse timer parameters and start (Page 3198)	x	
S_PEXT: Assign extended pulse timer parameters and start (Page 3201)	x	
S_ODT: Assign on-delay timer parameters and start (Page 3204)	x	
S_ODTS: Assign retentive on-delay timer parameters and start (Page 3207)	x	
S_OFFDT: Assign off-delay timer parameters and start (Page 3209)	x	
Counter operations		
IEC counters		
CTU: Count up (Page 3220)	x	
CTD: Count down (Page 3223)	x	
CTUD: Count up and down (Page 3226)	x	

LAD instruction	Permanent instruction	Transition/ Supervision/ Interlock
SIMATIC counters (S7-1500)		
S_CU: Assign parameters and count up (Page 3231)	x	
S_CD: Assign parameters and count down (Page 3233)	x	
S_CUD: Assign parameters and count up / down (Page 3229)	x	
Comparator operations		
CMP ==: Equal (Page 3238)	x	x
CMP <>: Not equal (Page 3241)	x	x
CMP >=: Greater or equal (Page 3243)	x	x
CMP <=: Less or equal (Page 3245)	x	x
CMP >: Greater than (Page 3247)	x	x
CMP <: Less than (Page 3249)	x	x
IN_RANGE: Value within range (Page 3251)	x	
OUT_RANGE: Value outside range (Page 3252)	x	
---| OK |---: Check validity (Page 3254)	x	
---| NOT_OK |---: Check invalidity (Page 3255)	x	
EQ_Type: Compare data type for EQUAL with the data type of a tag (Page 3256)	x	
NE_Type: Compare data type for UNEQUAL with the data type of a tag (Page 3257)	x	
EQ_ElemType: Compare element data type for EQUAL with the data type of a tag (Page 3258)	x	
NE_ElemType: Compare element data type for UNEQUAL with the data type of a tag (Page 3259)	x	
IS_NULL: Compare for EQUALS ZERO (Page 3261)	x	
NOT_NULL: Compare for UNEQUALS ZERO (Page 3262)	x	
IS_ARRAY: Check for ARRAY (Page 3263)	x	
CMP>T: Greater than step activation time (Page 4494)		x
CMP>U: Greater than uninterrupted step activation time (Page 4495)		x
Math functions		
CALCULATE: Calculate (Page 3264)	x	
ADD: Add (Page 3267)	x	
SUB: Subtract (Page 3269)	x	
MUL: Multiply (Page 3270)	x	
DIV: Divide (Page 3271)	x	
MOD: Return remainder of division (Page 3273)	x	
NEG: Create twos complement (Page 3274)	x	
INC: Increment (Page 3275)	x	
DEC: Decrement (Page 3276)	x	
ABS: Form absolute value (Page 3278)	x	
MIN: Get minimum (Page 3279)	x	
MAX: Get maximum (Page 3281)	x	
LIMIT: Set limit value (Page 3283)	x	
SQR: Form square (Page 3285)	x	
SQRT: Form square root (Page 3286)	x	

LAD instruction	Permanent instruction	Transition/ Supervision/ Interlock
LN: Form natural logarithm (Page 3288)	x	
EXP: Form exponential value (Page 3289)	x	
SIN: Form sine value (Page 3290)	x	
COS: Form cosine value (Page 3292)	x	
TAN: Form tangent value (Page 3293)	x	
ASIN: Form arcsine value (Page 3294)	x	
ACOS: Form arccosine value (Page 3296)	x	
ATAN: Form arctangent value (Page 3297)	x	
FRAC: Return fraction (Page 3298)		
EXPT: Exponentiate (Page 3300)	x	
Move operations		
MOVE: Move value (Page 3301)	x	
Deserialize: Deserialization (Page 3307)	x	
Serialize: Serialization (Page 3310)	x	
FieldRead: Read field (Page 3314)	x	
FieldWrite: Write field (Page 3316)	x	
MOVE_BLK: Move block (Page 3318)	x	
MOVE_BLK_VARIANT: Move block (Page 3320)	x	
UMOVE_BLK: Move block uninterruptible (Page 3324)	x	
FILL_BLK: Fill block (Page 3326)	x	
UFILL_BLK: Fill block uninterruptible (Page 3328)	x	
SWAP: Swap (Page 3330)	x	
ReadFromArrayDB: Read from the array data block (Page 3332)	x	
WriteToArrayDB: Write to the array data block (Page 3334)	x	
ReadFromArrayDBL: Read from the array data block in the load memory (Page 3337)	x	
WriteToArrayDBL: Write to array data block in the load memory (Page 3340)	x	
VariantGet: Read out VARIANT tag value (Page 3343)	x	
VariantPut: Write VARIANT tag value (Page 3344)	x	
CountOfElements: Get number of ARRAY elements (Page 3346)	x	
BLKMOV: Move block (S7-1500) (Page 3347)	x	
UBLKMOV: Move block uninterruptible (S7-1500) (Page 3350)	x	
FILL: Fill block (S7-1500) (Page 3352)	x	
Conversion operations		
CONVERT: Convert value (Page 3355)	x	
ROUND: Round numerical value (Page 3357)	x	
CEIL: Generate next higher integer from floating-point number (Page 3358)	x	
FLOOR: Generate next lower integer from floating-point number (Page 3359)	x	
TRUNC: Truncate numerical value (Page 3361)	x	
SCALE_X: Scale (Page 3362)	x	
NORM_X: Normalize (Page 3365)	x	
SCALE: Scale (S7-1500) (Page 3367)	x	

LAD instruction	Permanent instruction	Transition/ Supervision/ Interlock
UNSCALE: Unscale (S7-1500) (Page 3369)	x	
Program control operations		
Runtime control		
ENDIS_PW: Limit and enable password legitimation (Page 3382)	x	
RE_TRIGR: Restart cycle monitoring time (Page 3387)	x	
STP: Exit program (Page 3387)	x	
GET_ERROR: Get error locally (Page 3388)	x	
GET_ERR_ID: Get error ID locally (Page 3392)	x	
INIT_RD: Initialize all retain data (Page 3394)	x	
WAIT: Configure time delay (Page 3396)	x	
RUNTIME: Runtime measurement (Page 3397)	x	
Word logic operations		
AND: AND logic operation (Page 3399)	x	
OR: OR logic operation (Page 3401)	x	
XOR: EXCLUSIVE OR logic operation (Page 3402)	x	
INV: Create ones complement (Page 3404)	x	
DECO: Decode (Page 3405)	x	
ENCO: Encode (Page 3407)	x	
SEL: Select (Page 3408)	x	
MUX: Multiplex (Page 3410)	x	
DEMUX: Demultiplex (Page 3413)	x	
Shift and rotate		
SHR: Shift right (Page 3415)	x	
SHL: Shift left (Page 3417)	x	
ROR: Rotate right (Page 3419)	x	
ROL: Rotate left (Page 3421)	x	
Additional instructions		
DRUM: Implement sequencer (Page 3423)	x	
DCAT: Discrete control-timer alarm (Page 3426)	x	
MCAT: Motor control-timer alarm (Page 3430)	x	
IMC: Compare input bits with the bits of the mask (Page 3434)	x	
SMC: Compare scan matrix (Page 3436)	x	
LEAD_LAG: Lead and lag algorithm (Page 3438)	x	
SEG: Create bit pattern for seven-segment display (Page 3441)	x	
BCDCPL: Create tens complement (Page 3443)	x	
BITSUM: Count number of set bits (Page 3444)	x	

CMP>T: Greater than step activation time

Description

You use the "Greater than step activation time" instruction to monitor the total duration of a step in supervision conditions. The times of any events or errors are also recorded in the process.

In the programmed condition, the current or the last activation time of the step (Operand1) is compared with a defined duration (Operand2) in milliseconds. If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

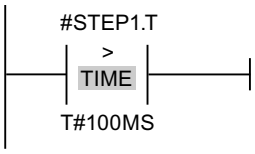
Parameters

The following table shows the parameters of the "Greater than step activation time" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	Integers, floating-point numbers, timers, strings, DATE, DT, DTL, TOD, LTOD, LDT	I, Q, M, D, L, or constant	Current or last activation time of the step
<Operand2>	Input	Integers, floating-point numbers, timers, strings, DATE, DT, DTL, TOD, LTOD, LDT	I, Q, M, D, L, or constant	Duration used for comparison

Example

The following example shows the instruction in the network:



As long as the activation time of #STEP1.T is less than 100 ms, the RLO is "0". As soon as the activation time exceeds 100 ms, the RLO changes to "1".

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

CMP>U: Greater than uninterrupted step activation time

Description

You use the "Greater than uninterrupted step activation time" instruction to monitor the duration of a step, minus any faults, in supervision conditions. The times of any events or faults are not recorded and only the pure step duration is monitored.

In the programmed condition, the total activation time of the step (Operand1) is compared with a defined duration (Operand2) in milliseconds. If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

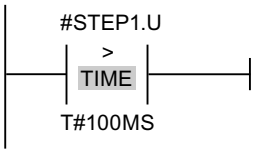
Parameters

The following table shows the parameters of the "Greater than uninterrupted step activation time" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	Integers, floating-point numbers, timers, strings, DATE, DT, DTL, TOD, LTOD, LDT	I, Q, M, D, L, or constant	Current or last activation time of the step minus the faults
<Operand2>	Input	Integers, floating-point numbers, timers, strings, DATE, DT, DTL, TOD, LTOD, LDT	I, Q, M, D, L, or constant	Duration used for comparison

Example

The following example shows the instruction in the network:



As long as the activation time of #STEP1.U, minus any faults, is less than 100 ms, the RLO is "0". As soon as the activation time exceeds 100 ms, the RLO changes to "1".

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2466)

Basic information on GRAPH (Page 2718)

FBD instructions Comparator operations

General FBD instructions

Description

Different FBD instructions are available to you when programming in GRAPH.

The following table shows an overview of the instructions and their availability:

FBD instruction	Permanent operation	Transition/ Supervision/ Interlock
General		
New network	x	
Empty box	x	x
Branch: Open branch	x	x
-|: Insert input (Page 3451)	x	x
-o|: Invert RLO (Page 3452)	x	x
Bit logic operations		
&: AND logic operation (Page 3445)	x	x
>=1: OR logic operation (Page 3447)	x	x
X: EXCLUSIVE OR logic operation (Page 3449)	x	
=: Assignment (Page 3452)	x	
/=: Negate assignment (Page 3453)	x	
R: Reset output (Page 3454)	x	
S: Set output (Page 3455)	x	

FBD instruction	Permanent operation	Transition/ Supervision/ Interlock
SET_BF: Set bit field (Page 3456)	x	
RESET_BF: Reset bit field (Page 3457)	x	
SR: Set/reset flip-flop (Page 3459)	x	
RS: Reset/set flip-flop (Page 3460)	x	
P: Scan operand for positive signal edge (Page 3461)	x	
N: Scan operand for negative signal edge (Page 3463)	x	
P=: Set operand on positive signal edge (Page 3464)	x	
N=: Set operand on negative signal edge (Page 3465)	x	
P_TRIG: Scan RLO for positive signal edge (Page 3467)	x	
N_TRIG: Scan RLO for negative signal edge (Page 3468)	x	
R_TRIG: Set tag on positive signal edge (Page 3469)	x	
F_TRIG: Set tag on negative signal edge (Page 3470)	x	
Timer operations		
IEC timers		
TP: Generate pulse (Page 3472)	x	
TON: Generate on-delay (Page 3474)	x	
TOF: Generate off-delay (Page 3477)	x	
TONR: Time accumulator (Page 3479)	x	
TP: Start pulse timer (Page 3482)	x	
TON: Start on-delay timer (Page 3484)	x	
TOF: Start off-delay timer (Page 3486)	x	
TONR: Time accumulator (Page 3488)	x	
RT: Reset timer (Page 3490)	x	
PT: Load time duration (Page 3491)	x	
SIMATIC Timers		
S_PULSE: Assign pulse timer parameters and start (Page 3493)	x	
S_PEXT: Assign extended pulse timer parameters and start (Page 3496)	x	
S_ODT: Assign on-delay timer parameters and start (Page 3499)	x	
S_ODTS: Assign retentive on-delay timer parameters and start (Page 3502)	x	
S_OFFDT: Assign off-delay timer parameters and start (Page 3504)	x	
Counter operations		
IEC counters		
CTU: Count up (Page 3517)	x	
CTD: Count down (Page 3520)	x	
CTUD: Count up and down (Page 3523)	x	
SIMATIC Counters		
S_CU: Assign parameters and count up (Page 3528)	x	
S_CD: Assign parameters and count down (Page 3530)	x	
S_CUD: Assign parameters and count up / down (Page 3526)	x	
Comparator operations		
CMP==: Equal (Page 3536)	x	x

FBD instruction	Permanent operation	Transition/ Supervision/ Interlock
CMP <>: Not equal (Page 3538)	x	x
CMP>=: Greater or equal (Page 3540)	x	x
CMP<=: Less or equal (Page 3542)	x	x
CMP>: Greater than (Page 3544)	x	x
CMP<: Less than (Page 3545)	x	x
IN_RANGE: Value within range (Page 3547)	x	
OUT_RANGE: Value outside range (Page 3548)	x	
---| OK |---: Check validity (Page 3550)	x	
---| NOT_OK |---: Check invalidity (Page 3551)	x	
EQ_Type: Compare data type for EQUAL with the data type of a tag (Page 3552)	x	
NE_Type: Compare data type for UNEQUAL with the data type of a tag (Page 3553)	x	
EQ_ElemType: Compare element data type for EQUAL with the data type of a tag (Page 3554)	x	
NE_ElemType: Compare element data type for UNEQUAL with the data type of a tag (Page 3555)	x	
IS_NULL: Compare for EQUALS ZERO (Page 3556)	x	
NOT_NULL: Compare for UNEQUALS ZERO (Page 3558)	x	
IS_ARRAY: Check for ARRAY (Page 3559)	x	
CMP>T: Greater than step activation time (Page 4500)		x
CMP>U: Greater than uninterrupted step activation time (Page 4501)		x
Math functions		
CALCULATE: Calculate (Page 3560)	x	
ADD: Add (Page 3563)	x	
SUB: Subtract (Page 3565)	x	
MUL: Multiply (Page 3566)	x	
DIV: Divide (Page 3568)	x	
MOD: Return remainder of division (Page 3569)	x	
NEG: Create twos complement (Page 3570)	x	
INC: Increment (Page 3572)	x	
DEC: Decrement (Page 3573)	x	
ABS: Form absolute value (Page 3574)	x	
MIN: Get minimum (Page 3576)	x	
MAX: Get maximum (Page 3578)	x	
LIMIT: Set limit value (Page 3580)	x	
SQR: Form square (Page 3583)	x	
SQRT: Form square root (Page 3584)	x	
LN: Form natural logarithm (Page 3586)	x	
EXP: Form exponential value (Page 3587)	x	
SIN: Form sine value (Page 3588)	x	
COS: Form cosine value (Page 3590)	x	
TAN: Form tangent value (Page 3591)	x	
ASIN: Form arcsine value (Page 3592)	x	

FBD instruction	Permanent operation	Transition/ Supervision/ Interlock
ACOS: Form arccosine value (Page 3594)	x	
ATAN: Form arctangent value (Page 3595)	x	
FRAC: Return fraction (Page 3597)	x	
EXPT: Exponentiate (Page 3598)	x	
Move operations		
MOVE: Move value (Page 3600)	x	
Deserialize: Deserialization (Page 3605)	x	
Serialize: Serialization (Page 3609)	x	
FieldRead: Read field (Page 3612)	x	
FieldWrite: Write field (Page 3614)	x	
MOVE_BLK: Move block (Page 3616)	x	
MOVE_BLK_VARIANT: Move block (Page 3618)	x	
UMOVE_BLK: Move block uninterruptible (Page 3623)	x	
FILL_BLK: Fill block (Page 3625)	x	
UFILL_BLK: Fill block uninterruptible (Page 3627)	x	
SWAP: Swap (Page 3629)	x	
ReadFromArrayDB: Read from the array data block (Page 3631)	x	
WriteToArrayDB: Write to the array data block (Page 3633)	x	
ReadFromArrayDBL: Read from the array data block in the load memory (Page 3635)	x	
WriteToArrayDBL: Write to array data block in the load memory (Page 3639)	x	
VariantGet: Read out VARIANT tag value (Page 3642)	x	
VariantPut: Write VARIANT tag value (Page 3643)	x	
CountOfElements: Get number of ARRAY elements (Page 3645)	x	
BLKMOV: Move block (S7-1500) (Page 3646)	x	
UBLKMOV: Move block uninterruptible (S7-1500) (Page 3649)	x	
FILL: Fill block (S7-1500) (Page 3652)	x	
Conversion operations		
CONVERT: Convert value (Page 3654)	x	
ROUND: Round numerical value (Page 3656)	x	
CEIL: Generate next higher integer from floating-point number (Page 3657)	x	
FLOOR: Generate next lower integer from floating-point number (Page 3659)	x	
TRUNC: Truncate numerical value (Page 3660)	x	
SCALE_X: Scale (Page 3662)	x	
NORM_X: Normalize (Page 3664)	x	
SCALE: Scale (S7-1500) (Page 3667)	x	
UNSCALE: Unscale (S7-1500) (Page 3670)	x	
Program control operations		
Runtime control		
ENDIS_PW: Limit and enable password legitimation (Page 3682)	x	
RE_TRIGR: Restart cycle monitoring time (Page 3687)	x	
STP: Exit program (Page 3687)	x	

FBD instruction	Permanent operation	Transition/ Supervision/ Interlock
GET_ERROR: Get error locally (Page 3688)	x	
GET_ERR_ID: Get error ID locally (Page 3692)	x	
INIT_RD: Initialize all retain data (Page 3695)	x	
WAIT: Configure time delay (Page 3696)	x	
RUNTIME: Runtime measurement (Page 3697)	x	
Word logic operations		
AND: AND logic operation (Page 3699)	x	
OR: OR logic operation (Page 3701)	x	
XOR: EXCLUSIVE OR logic operation (Page 3703)	x	
INV: Create ones complement (Page 3704)	x	
DECO: Decode (Page 3706)	x	
ENCO: Encode (Page 3707)	x	
SEL: Select (Page 3708)	x	
MUX: Multiplex (Page 3710)	x	
DEMUX: Demultiplex (Page 3713)	x	
Shift and rotate		
SHR: Shift right (Page 3717)	x	
SHL: Shift left (Page 3719)	x	
ROR: Rotate right (Page 3721)	x	
ROL: Rotate left (Page 3723)	x	
Additional instructions		
DRUM: Implement sequencer (Page 3725)	x	
DCAT: Discrete control-timer alarm (Page 3728)	x	
MCAT: Motor control-timer alarm (Page 3731)	x	
IMC: Compare input bits with the bits of the mask (Page 3735)	x	
SMC: Compare scan matrix (Page 3737)	x	
LEAD_LAG: Lead and lag algorithm (Page 3740)	x	
SEG: Create bit pattern for seven-segment display (Page 3743)	x	
BCDCPL: Create tens complement (Page 3744)	x	
BITSUM: Count number of set bits (Page 3746)	x	

CMP>T: Greater than step activation time

Description

You can use the "Greater than step activation time" instruction to monitor the total duration of a step in supervision conditions. The times of any events or errors are also recorded in the process.

In the programmed condition, the current or the last activation time of the step (Operand1) is compared with a defined duration (Operand2) in milliseconds. If the condition of the

comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

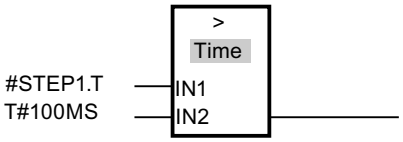
Parameters

The following table shows the parameters of the "Greater than step activation time" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	Integers, floating-point numbers, timers, strings, DATE, DT, DTL, TOD, LTOD, LDT	I, Q, M, D, L, or constant	Current or last activation time of the step
<Operand2>	Input	Integers, floating-point numbers, timers, strings, DATE, DT, DTL, TOD, LTOD, LDT	I, Q, M, D, L, or constant	Duration used for comparison

Example

The following example shows the instruction in the network:



As long as the activation time of #STEP1.T is less than 100 ms, the RLO is "0". As soon as the activation time exceeds 100 ms, the RLO changes to "1".

CMP>U: Greater than uninterrupted step activation time

Description

You use the "Greater than uninterrupted step activation time" instruction to monitor the duration of a step, minus any faults, in supervision conditions. The times of any events or faults are not recorded and only the pure step duration is monitored.

In the programmed condition, the total activation time of a step (Operand1) is compared with a defined duration (Operand2) in milliseconds. If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

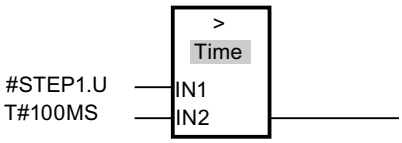
Parameters

The following table shows the parameters of the "Greater than uninterrupted step activation time" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	Integers, floating-point numbers, timers, strings, DATE, DT, DTL, TOD, LTOD, LDT	I, Q, M, D, L, or constant	Current or last activation time of the step minus the faults
<Operand2>	Input	Integers, floating-point numbers, timers, strings, DATE, DT, DTL, TOD, LTOD, LDT	I, Q, M, D, L, or constant	Duration used for comparison

Example

The following example shows the instruction in the network:



As long as the activation time of #STEP1.U, minus any faults, is less than 100 ms, the RLO is "0". As soon as the activation time exceeds 100 ms, the RLO changes to "1".

9.8.1.3 Extended instructions

Date and time-of-day

T_COMP: Compare time tags

Description

This instruction is used to compare the contents of two tags of the data types "Timers" or "Date and time".

The instruction supports comparisons of the following data types: DATE, TIME, LTIME, TOD (TIME_OF_DAY), LTOD (LTIME_OF_DAY), DT (DATE_AND_TIME), LDT (DATE_AND_LTIME), DTL, S5Time.

The data types must be the same length and format to carry out the comparison.

The comparison result is output at the OUT parameter as a return value. For this purpose, the parameter OUT is set to "1" if the condition used for the comparison is satisfied.

The following comparison options can be used:

Symbol	Description
EQ	The return value has the signal state “1" if the time is the same at the parameter IN1 and IN2.
NE	The return value has the signal state “1" if the time at parameters IN1 and IN2 is not identical.
GE	The return value has the signal state "1" if the time at parameter IN1 is greater (more recent) than or equal to the time at parameter IN2 .
LE	The return value has the signal state "1" if the time at parameter IN1 is less (less recent) than or equal to the time at parameter IN2.
GT	The return value has the signal state “1" if the time at parameter IN1 is greater (more recent) than the time at parameter IN2.
LT	The return value has the signal state "1" if the time at parameter IN1 is less (less recent) than the time at parameter IN2 .

Parameters

The following table shows the parameters of the instruction "T_COMP":

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	DATE, TIME, LTIME, TOD, LTOD, DT, LDT, DTL	I, Q, M, D, L, P or constant	First value to be compared.
IN2	Input	DATE, TIME, LTIME, TOD, LTOD, DT, LDT, DTL	I, Q, M, D, L, P or constant	Second value to be compared.
OUT	Output	BOOL	I, Q, M, D, L, P	Return value

You can find additional information on valid data types in "Overview of the valid data types (Page 2236)".

T_CONV: Convert times and extract

Description

You use the instruction "T_CONV" to convert the data type of the IN input parameter to the data type that is output at the OUT output. You select the data formats for the conversion from the instruction boxes of the input and output.

Parameters

The following table shows the parameters of the instruction "T_CONV". If an input and output parameter of the same data type is used, the instruction copies the corresponding value.

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	Integers, TIME, date and time*	WORD, integers, timers, date and time*	I, Q, M, D, L, P or constant	Value to be converted
OUT	Return	Integers, TIME, date and time*	WORD, integers, timers, date and time*	I, Q, M, D, L, P	Result of the conversion
* The range of supported data types depends on the CPU. Please refer to the overviews of valid data types for information on the data types supported by the S7-1200 and S7-1500 modules.					

You can find additional information on valid data types in "Overview of the valid data types (Page 2236)".

T_ADD: Add times

Description

You use this instruction to add the time information in the IN1 input to the time information in the IN2 input. You can query the result in the OUToutput parameter. You can add the following formats:

- Addition of a time period to another time period.
  Example: Addition of a TIME data type to another TIME data type.
- Addition of a time period to a time.
  Example: Addition of a TIME data type to the DTL data type.

The data type for the value at input parameter IN1 and output parameter OUT is defined by the selection in the instruction boxes of the input and output. You can only specify time information in TIME format in the IN2 input parameter (for S7-1500 modules also LTIME).

Parameters

The following tables show the parameters of the "T_ADD" instruction, according to the possible conversions:

Table 9-36 Addition of a time period to another time period

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN1	Input	TIME	TIME, LTIME	I, Q, M, D, L, P or constant	First number to be added
IN2	Input	TIME	TIME, LTIME	I, Q, M, D, L, P or constant	Second number to be added
OUT	Return	DINT, DWORD, TIME, TOD	TIME, LTIME,	I, Q, M, D, L, P	Result of addition The data type selection depends on the data types selected for the IN1 and IN2 input parameters.

Table 9-37 Addition of a time period to a time

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN1	Input	DTL, TOD	DT, TOD, LTOD, LDT, DTL	I, Q, M, D, L, P or constant	First number to be added
IN2	Input	TIME	TIME, LTIME	I, Q, M, D, L, P or constant	Second number to be added
OUT	Return	DINT, DWORD, TIME, TOD, UDINT, DTL	DT, DTL, LDT, TOD, LTOD	I, Q, M, D, L, P	Result of addition The data type selection depends on the data types selected for the IN1 and IN2 input parameters.

You can find additional information on valid data types in "Overview of the valid data types (Page 2236)".

T_SUB: Subtract times

Description

You use this instruction to subtract the time information in the IN2 input parameter from the time information in the IN1 input parameter. You can query the difference in the OUT output parameter. You can subtract the following formats:

- Subtraction of a time period from another time period
  Example: Subtraction of a time period of the data type TIME from another time period of the data type TIME. The result can be output to a tag with the TIME format.
- Subtraction of a time period from a time
  Example: Subtraction of a time period of the data type TIME from a time of the data type DTL. The result can be output to a tag with the DTL format.

You decide the formats of the values in the IN1 input parameter and the OUT output parameter by selecting the data types for the input and output parameters of the instruction.

Parameters

The following tables show the parameters of the "T_SUB" instruction, according to the possible conversions:

Table 9-38 Subtraction of a time period from another time period

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN1	Input	TIME	TIME, LTIME	I, Q, M, D, L, P or constant	Minuend
IN2	Input	TIME	TIME, LTIME	I, Q, M, D, L, P or constant	Subtrahend
OUT	Return	DINT, DWORD, TIME, TOD, UDINT	TIME, LTIME	I, Q, M, D, L, P	Result of subtraction

Table 9-39 Subtraction of a time period from a time

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN1	Input	DTL, TOD	TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P or constant	Minuend
IN2	Input	TIME	TIME, LTIME	I, Q, M, D, L, P or constant	Subtrahend
OUT	Return	DTL, DINT, DWORD, TIME, TOD, UDINT	TOD, LTOD, DTL, DT, LDT	I, Q, M, D, L, P	Result of subtraction

You can find additional information on valid data types in "Overview of the valid data types (Page 2236)".

T_DIFF: Time difference

Description

You use this instruction to subtract the time information in the IN2 input parameter from the time information in the IN1 input parameter. The result is sent at output parameter OUT.

- If the time information at the IN2 input parameter is greater than the time information at the IN1 input parameter, the result is output as a negative value at the OUT output parameter.
- If the result of the subtraction is outside the TIME range, the result is set to "0" (0:00) and the enable output ENO = "0".

Parameters

The following table shows the parameters of the "T_DIFF" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN1	Input	DTL, DATE, TOD	DTL, DATE, DT, TOD, LTOD, LDT	I, Q, M, D, L, P or constant	Minuend
IN2	Input	DTL, DATE, TOD	DTL, DATE, DT, TOD, LTOD, LDT	I, Q, M, D, L, P or constant	Subtrahend
OUT	Return	TIME, INT	TIME, LTIME, INT	I, Q, M, D, L, P	Difference of input parameters

You can find additional information on valid data types in "Overview of the valid data types (Page 2236)".

T_COMBINE: Combine times

Description

The instruction combines the value of a date with the value of a time and converts this into a combined date and time value.

- The date is output at the input parameter IN1. A value of between 1990-01-01 and 2089-12-31 must be used for the data type DATE (this is not checked).
- The time is input at the IN2 input value (TOD/LTOD data type).
- The combined data type for the date and time value is output at the OUT output value.

Parameters

The following table shows the parameters of the instruction "T_COMBINE":

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN1	Input	DATE	DATE	I, Q, M, D, L, P or constant	Input tag of the date
IN2	Input	TOD	TOD, LTOD	I, Q, M, D, L, P or constant	Input tag of the time
OUT	Return	DTL	DT, DTL, LDT	I, Q, M, D, L, P	Return value of the date and time

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

Time-of-day functions

WR_SYS_T: Set time-of-day

Description

You use this instruction to set the date and time-of-day of the CPU clock. Enter the date and time-of-day at the input parameter IN . The value must be in the following range:

- With DT: min. DT#1990-01-01-0:0:0, max. DT#2200-12-31-23:59:59.999
- With LDT: min. LDT#1970-1-1-0:0:0.000000000, max. LDT#2200-12-31-23:59:99.999999999
- With DTL: min. DTL#1970-01-01-00:00:00.0, max. DTL#2200-12-31-23:59:59.999999999

You can query whether errors have occurred during execution of the instruction in the RET_VAL output parameter.

The "WR_SYS_T" instruction cannot be used to pass information about the local time zone or daylight saving time.

Parameters

The following table shows the parameters of the "WR_SYS_T" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	DTL	DT, DTL, LDT	I, Q, M, D, L, P or constant *	Date and time
RET_VAL	Return	INT	INT	I, Q, M, D, L, P	Status of the instruction
* The data types DT and DTL cannot be used for the following memory areas: import, output and bit memory.					

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code* (W#16#....)	Description
0000	No error
8080	Error in date
8081	Error in time
8082	Month invalid
8083	Day invalid
8084	Hour information invalid
8085	Minute information invalid
8086	Second information invalid

Error code* (W#16#....)	Description
8087	Nanosecond information invalid
80B0	The real-time clock has failed
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

RD_SYS_T: Read time-of-day

Description

You use this instruction to read out the current date and current time-of-day of the CPU clock.

The read out dates are output at the OUT output parameter of the instruction. The provided value does not include information about the local time zone or daylight saving time.

You can query whether errors have occurred during execution of the instruction in the RET_VALoutput.

Parameters

The following table shows the parameters of the "RD_SYS_T" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
RET_VAL	Return	INT	INT	I, Q, M, D, L, P	Status of the instruction
OUT	Output	DTL	DT, DTL, LDT	I, Q, M, D, L, P	Date and time of CPU

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code* (W#16#....)	Description
0000	No error
8081	Time value specified at the OUT parameter is outside the valid value range: • With DT: min. DT#1990-01-01-0:0:0, max. DT#2200-12-31-23:59:59.999 • With LDT: min. LDT#1970-1-1-0:0:0.000000000, max. LDT#2200-12-31-23:59:59.999999999 • With DTL: min. DTL#1970-01-01-00:00:00.0, max. DTL#2200-12-31-23:59:59.999999999
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

RD_LOC_T: Read local time

Description

You use this instruction to read the current local time from the CPU clock and output this at the OUT output. Information on the time zone and the start of daylight saving time and standard time, which you have set in the configuration of the CPU clock, is used to output the local time.

Parameter

The following table shows the parameters of the "RD_LOC_T" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
RET_VAL	Return	INT	INT	I, Q, M, D, L, P	Status of the instruction
OUT	Output	DTL	DT, LDT, DTL	I, Q, M, D, L, P	Local time

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

RET_VAL parameter

Error code* (W#16#....)	Description
0000	No error
0001	No error. Local time is output as daylight saving time.
8080	Local time cannot be read.
8081	Time value specified at the OUT parameter is outside the valid value range: • With DT: min. DT#1990-01-01-0:0:0, max. DT#2089-12-31-23:59:59.999 • With LDT: min. LDT#1970-1-1-0:0:0.000000000, max. LDT#2200-12-31-23:59:99.999999999 • With DTL: min. DTL#1970-01-01-00:00:00.0, max. DTL#2200-12-31-23:59:59.999999999
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

WR_LOC_T: Write local time

Description

The instruction "WR_LOC_T" is used to set the date and time of the CPU clock. Enter the date and time-of-day as local time at the input parameter LOCTIME.

The value must be in the following range:

- With DT: min. DT#1990-01-01-00:00:00, max. DT#2089-12-31-23:59:59.999
- With DTL: min. DTL#1970-01-01-00:00:00.0, max. DTL#2200-12-31-23:59:59.999999999
- With LDT: min. LDT#1970-1-1-0:0:0.000000000, max. LDT#2200-12-31-23:59:59.999999999

The granularity of the time information for local time and system time is product-specific and is at least one millisecond. Input values at the LOCTIME parameter which are less than those supported by the CPU are rounded up during system time calculation.

You can query whether errors have occurred during execution of the instruction in the RET_VAL output parameter.

Use of WR_LOC_T during the time change to daylight savings or standard time

- Changeover from standard to daylight saving time
  In the following it is assumed that the time of the changeover is 2:00 am, and that the time is moved forward by one hour. This means there is no hour between 02:00:00:000000000 AM and 02:59:59:999999999 AM.
  When you specify a corresponding time for LOCTIME, the error code W#16#8089 is generated.
  DST is irrelevant.
- Changeover from daylight saving to standard time
  In the following it is assumed that the time of the changeover is 3:00 am, and that the time is moved back by one hour. This means there are two hours between 02:00:00:000000000 AM and 02:59:59:999999999 AM.
  For all times for LOCTIME that are between 02:00:00:000000000 AM and 02:59:59:999999999 AM, you therefore need to declare whether the time is before or after the time changeover. The DST parameter is used for this:
  - With DST=TRUE, the time is in the first of the two hours, i.e. still in daylight savings time.
  - With DST=FALSE, the time is in the second of the two hours, i.e. in standard time.

  For all times for LOCTIME that are outside the double hour, DST is irrelevant.

Parameters

The following table shows the parameters of the "WR_LOC_T" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
LOCTIME	Input	DTL	DT, DTL, LDT	D, L, P or constant	Local time
DST	Input	BOOL	BOOL	I, Q, M, D, L, P, T, C or constant	**D**aylight **S**aving **T**ime Is only evaluated during the "double hour" at changeover to standard time. • TRUE = daylight saving time (first hour) • FALSE = standard time (second hour)
RET_VAL	Return	INT	INT	I, Q, M, D, L, P	Error message (see "RET_VAL parameter")

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

RET_VAL parameter

Error code* (W#16#....)	Description
0000	No error.
8080	The LOCTIME parameter has an invalid value.
8081	Time value specified at the LOCTIME parameter is outside the valid value range: • With DT: min. DT#1990-01-01-00:00:00, max. DT#2089-12-31-23:59:59.999 • With DTL: min. DTL#1970-01-01-00:00:00.0, max. DTL#2200-12-31-23:59:59.999999999 • With LDT: min. LDT#1970-1-1-0:0:0.000000000, max. LDT#2200-12-31-23:59:59.999999999
8082**	Invalid value specified for the month (byte 2 in DTL format).
8083**	Invalid value specified for the day (byte 3 in DTL format).
8084**	Invalid value specified for the hour (byte 5 in DTL format).
8085**	Invalid value specified for the minute (byte 6 in format DTL).
8086**	Invalid value specified for the second (byte 7 in DTL format).
8087**	Invalid value specified for the nanosecond (byte 8 to 11 in DTL format).
8089	Time value does not exist (hour already passed upon changeover to daylight saving time).
80B0	The real-time clock has failed.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	
** Only for local time information at the LOCTIME parameter in DTL format.	

See also

Switching display formats in the program status (Page 2980)

SET_TIMEZONE: Set time zone

Description

Use the instruction "SET_TIMEZONE" to set the parameter for the local time zone and the daylight saving / standard time changeover.

The settings which are carried out with the instruction "SET_TIMEZONE" correspond with the settings for the time-of-day in the properties of the CPU. Define the corresponding parameters in the system data type TimeTransformationRule for execution of the instruction "SET_TIMEZONE".

The local time is calculated based on the system time using the settings for the time zone and the daylight saving / standard time changeover. The system time of the CPU is the UTC time. The system time is used exclusively for communication within the system.

Parameters

The following table shows the parameters of the "SET_TIMEZONE" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, P or constant	Setting the parameters which are stored at the parameter TimeZone.
TimeZone	Input	Time Transformation Rule (Page 4514)	D, L	Store the parameters for the local time zone and the daylight saving / standard time changeover in the system data type TimeTransformationRule.
DONE	Output	BOOL	I, Q, M, D, L, P	Status parameter: • 0: Job not yet started or is still executing • 1: Job completed error-free
BUSY	Output	BOOL	I, Q, M, D, L, P	Status parameter: • 0: Job not yet started or already completed • 1: Job not yet completed. A new job cannot be started.
ERROR	Output	BOOL	I, Q, M, D, L, P	Status parameter: • 0: No error • 1: Error occurred
STATUS	Output	DINT, DWORD, UDINT, WORD	I, Q, M, D, L, P	Detailed error and status information is output at the parameter STATUS. The parameter is only set for the duration of one call. To display the status, you should therefore copy the STATUS parameter to a free data area.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter STATUS

Error code* (W#16#....)	Description
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
808x	Incorrect input at the xth parameter of the system data type TimeTransformationRule. Example: Error code 8084 corresponds with an incorrect input at the 4th parameter (DaylightStartWeek). Only the values 1 to 5 are valid here.
80C3	Temporary resource error: The CPU is currently processing the maximum possible number of simultaneous block calls. "SET_TIMEZONE" cannot be executed unless at least one of the block calls is finished.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

TimeTransformationRule

Description

The times for changeover to daylight saving time and standard time are defined with the system data type TimeTransformationRule.

The following options are available for creating the structure TimeTransformationRule:

- Create a new data block and select as the type TimeTransformationRule.
- Enter TimeTransformationRule as the Data type in an existing data block or a local interface of a function block.

The structure is as follows:

Name		Data type	Description
TimeTransformationRule		STRUCT	
	Bias	INT	Time difference between the local time and the system time (UTC) in minutes.
	DaylightBias	INT	The time difference between standard and daylight saving time in minutes.
	Time specification for change to daylight saving time. The following times always refer to the local time.		
	DaylightStartMonth	USINT	Month in which change is made to daylight saving time: 1 = January 2 = February 3 = March ... 12 = December

Name		Data type	Description
	DaylightStartWeek	USINT	Week in which change is made to daylight saving time. 1 = First occurrence of the weekday in the month ... 5 = Last occurrence of the weekday in the month
	DaylightStartWeekday	USINT	Weekday on which change is made to daylight saving time: 1 = Sunday
	DaylightStartHour	USINT	Hour in which change is made to daylight saving time
	DaylightStartMinute	USINT	Minute in which change is made to daylight saving time
	Time when change is made to standard time, the following times always refer to the local time.		
	StandardStartMonth	USINT	Month in which change is made to standard time: 1 = January 2 = February 3 = March ... 12 = December
	StandardStartWeek	USINT	Week in which change is made to standard time: 1 = First occurrence of the weekday in the month ... 5 = Last occurrence of the weekday in the month
	StandardStartWeekday	USINT	Weekday on which change is made to standard time: 1 = Sunday
	StandardStartHour	USINT	Hour in which change is made to standard time
	StandardStartMinute	USINT	Minute in which change is made to standard time
	TimeZoneName	STRING[80]	Name of the time zone, for example: "(GMT+01:00) Amsterdam, Berlin, Bern, Rome, Stockholm, Vienna"

SNC_RTCB: Synchronize slave clocks

Definition: Synchronization of slave clocks

The synchronization of clock slaves refers to the transfer of the date and time-of-day from the clock master of a bus segment to all clock slaves of this bus segment.

Description

You use this instruction to synchronize all slave clocks present on a bus segment independent of the assigned synchronization interval. Successful synchronization is only possible if "SNC_RTCB" is called on a CPU whose real-time clock was assigned as the master clock for at least one bus segment.

Parameters

The following table shows the parameters of the "SNC_RTCB" instruction:

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Output	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0000	No error occurred during synchronization.
0001	The existing clock was not assigned the master clock function for any of the bus segments.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 3392)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

TIME_TCK: Read system time

Description

With the "TIME_TCK" instruction, you read the system time of the CPU. The system time is a time counter that counts from 0 to a maximum of 2147483647 ms. In case of an overflow, the system time is counted again starting with "0". The time scale and the accuracy of the system time is 1 ms. The system time is only influenced by the operating modes of the CPU. You can use the system time, for example, to measure the duration of processes by comparing the results of two "TIME_TCK" calls. The instruction does not provide any error information.

The following table provides an overview of how the system time changes depending on the operating modes of the CPU.

Mode	System time ...
Startup	... is constantly updated
RUN	

Mode	System time ...
STOP	... is stopped and retains the current value
Warm restart	... is deleted and restarts with "0"

Parameter

The following table shows the parameters of the instruction "TIME_TCK":

Parameters	Declaration	Data type	Memory area	Description
RET_VAL	Return	TIME	I, Q, M, D, L	The RET_VAL parameter contains the read system time in the range from 0 to 2^{31} -1 ms.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

See also

Switching display formats in the program status (Page 2980)

RTM: Runtime meters

Description

You can use this instruction to set, start, stop, and read out a 32-bit runtime meter of your CPU.

Ensure that the runtime meter can also be stopped or restarted during execution of the user program, which may render the saved values incorrect.

Parameters

The following table shows the parameters of the "RTM" instruction:

Parameter	Declaration	Data type	Memory area	Description
NR	Input	RTM	I, Q, M, D, L or constant	Number of the runtime meter Numbering starts with 0. For information on the number of runtime meters of your CPU, refer to the technical data.
MODE	Input	BYTE	I, Q, M, D, L or constant	Job ID: • 0: Read out (the status is then written to CQ and the current value to CV). After the runtime meter has reached (2E31) -1 hours, it stops at the highest value that can be displayed and outputs an "Overflow" error message. • 1: start (at the last counter value) • 2: stop • 4: set (to the value specified in PV) • 5: set (to the value specified in PV) and then start • 6: set (to the value specified in PV) and then stop
PV	Input	DINT	I, Q, M, D, L or constant	New value for the runtime meter
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
CQ	Output	BOOL	I, Q, M, D, L	Status of the runtime meter (1: running)
CV	Output	DINT	I, Q, M, D, L	Current value of the runtime meter

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error
8080	Wrong number for the runtime meter
8081	A negative value was passed to the PV parameter.
8082	Overflow of the runtime meter
8091	The MODE input parameter contains an invalid value.
General error information	See also: Evaluating errors with GET_ERR_ID (Page 4282)

String + Char

S_MOVE: Move character string

Description

You can use this instruction to move the content of a string (W)STRING. The string in the IN input parameter is copied to the OUT output parameter.

The tags at the input parameter IN and the output OUT must have the same data type.

You can use the "MOVE_BLK" and "UMOVE_BLK" instructions to copy tags of data type ARRAY.

Parameters

The following table shows the parameters of the "S_MOVE" instruction:

Parameters	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	STRING	STRING, WSTRING	D, L or constant	Source string
OUT	Output	STRING	STRING, WSTRING	D, L or constant	Destination string

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

S_COMP: Compare character strings

Description

The instruction compares the contents of two tags in the STRING format and outputs the result of the comparison as a return value. The tags that are to be compared will be interconnected at the IN1 and IN2 inputs. You can only assign a symbolically defined tag for the input parameters.

Use the instruction box to select the comparison condition. If the comparison condition (for example, greater than or equal to) is satisfied, the signal state is set to "1" at the output parameter OUT .

The following comparison options can be used:

Symbol	Description
EQ	The return value has the signal state "1" if the string at the IN1 parameter is the same as the string at the IN2 parameter.
NE	The return value has the signal state "1" if the string at the IN1 parameter is not equal to the string at the IN2 parameter.

Symbol	Description
GT (1)	The return value has the signal state “1" if the string at the IN1 parameter is greater than the string at the IN2 parameter.
LT (1)	The return value has the signal state "1" if the string at the IN1 parameter is less than the string at the IN2 parameter.
GE (1)	The return value has the signal state "1" if the string at the IN1 parameter is greater than or equal to the string at the IN2 parameter.
LE (1)	The return value has the signal state "1" if the string at the IN1 parameter is less than or equal to the string at the IN2 parameter.
(1) The characters are compared by their ASCII code (for example, 'a' is greater than 'A'), starting from the left. The first character to be different decides the result of the comparison. If the first characters are the same, the longer string is greater.	

Parameters

The following table shows the parameters of the "S_COMP" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	STRING*	D, L or constant	Input tag in the STRING format.
IN2	Input	STRING*	D, L or constant	Input tag in the STRING format.
OUT	Output	BOOL	I, Q, M, D, L	Result of comparison
* Define the maximum length of the character string if you use the data type STRING in the interface declaration for a temporary tag (you will find further information in the description of the data type).				

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

S_CONV: Convert character string

Description

You use this instruction to convert the value at the IN input to the data format you have specified in the OUT output. You decide the output format of the conversion by selecting a data type for the OUT output parameter.

The following conversions are possible:

- Conversion of a character string (STRING) to:
  - Numerical value (integer or floating-point number)
    The conversion is performed for all characters of the character string specified in the IN input parameter. Permitted characters are the digits "0" to "9", the decimal point, and the plus and minus signs. The first character of the string may be a valid number or a sign. Leading spaces and exponential notations are ignored.
  - A character (CHAR)
    The first character of the character string is transferred to the parameter OUT when a character string is converted to a character (CHAR).
- Conversion of a numerical value or character to a character string (STRING):
  - You decide the format of the numeric value to be converted by selecting a data type for the IN input. A valid tag of the STRING data type must be specified in the OUT output. The length of the character string after conversion depends on the value at the IN input.
  - The conversion result is saved as a string starting at the third byte. The first byte of the string records the maximum length and the second byte the actual length of the character string. Positive numeric value are output without a sign.
  - When you convert zero (e.g. INT_TO_STRING(0)), the string will be 6 characters long.
  - The first character of the character string is filled with spaces when converting to a numerical value. The number of spaces depends on the length of the numerical value.
  - The character is written at the first position in the character string when converting to a character (CHAR).

Note

Exponential notation during conversion from floating-point numbers

Do not use exponential notation ("e" or "E") for the conversion from floating-point numbers with the instruction "S_CONV". Instead, use the instruction "STRG_VAL (Page 4522)" for the conversion of floating-point numbers with exponential notation. You can use the FORMAT parameter of the instruction to select exponential notation as input format.

Parameters

The following tables show the parameters of the "S_CONV" instruction, according to the possible conversions:

Table 9-40 Parameters for converting a character string to a numeric value:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	STRING	D, L or constant	Value to be converted
OUT	Output	CHAR, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, REAL, LREAL	I, Q, M, D, L	Result of the conversion

Table 9-41 Parameter when converting a numerical value or character to a character string:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	CHAR, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, REAL, LREAL	I, Q, M, D, L or constant	Value to be converted
OUT	Output	STRING	D, L	Result of the conversion

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

STRG_VAL: Convert character string to numerical value

Description

The instruction "STRG_VAL" converts a character string to an integer or a floating-point number:

- You specify the character string to be converted in the IN input parameter.
- You define the format of the output value by selecting a data type for the OUT output parameter.

Permitted characters for the conversion are the digits "0" to "9", the decimal point, the decimal comma, notations "E" and "e", and the plus and minus characters. The conversion can be interrupted by invalid characters.

The "STRG_VAL" instruction is not supported by the SCL programming language.

Parameters

The following table shows the parameters of the "STRG_VAL" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	STRING	STRING, WSTRING	D, L or constant	Numeric character string to be converted
FORMAT	Input	WORD	WORD	I, Q, M, D, L, P or constant	Output format of the characters
P	Input	UINT	UINT	I, Q, M, D, L, P or constant	Reference to the first character to be converted (first character = 1, the value "0" or a value > length of the string is invalid)
OUT	Output	USINT, SINT, UINT, INT, UDINT, DINT, REAL, LREAL	USINT, SINT, UINT, INT, UDINT, DINT, ULINT, LINT, REAL, LREAL	I, Q, M, D, L, P	Result of the conversion

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

FORMAT parameter

You use the FORMAT parameter to specify how the characters of a character string are to be interpreted. Exponential values can also be converted and represented with the "STRG_VAL" instruction.

The following table shows the possible values of the FORMAT parameter and their meaning:

Value (W#16#....)	Notation	Decimal representation
0000	Decimal fraction	"."
0001		","
0002	Exponential	"."
0003		","
0004 to FFFF	Invalid values	

P parameter

The conversion starts at the character whose position you specified in the P parameter. If, for example, the value "1" is specified in the P parameter, the conversion starts at the first character of the specified character string.

Example

The following table shows examples of the conversion of a character string to a numeric value:

IN (STRING)	FORMAT (W#16#....)	OUT (data type)	OUT (value)	ENO status
'123'	0000	INT/DINT	123	1
'-00456'	0000	INT/DINT	-456	1
'123.45'	0000	INT/DINT	123	1
'+2345'	0000	INT/DINT	2345	1
'00123AB'	0000	INT/DINT	123	1
'123'	0000	REAL	123.0	1
'-00456'	0001	REAL	-456.0	1
'+00456'	0001	REAL	456.0	1
'123.45'	0000	REAL	123.45	1
'123.45'	0001	REAL	12345.0	1
'123,45'	0000	REAL	12345.0	1
'123,45'	0001	REAL	123.45	1
'.00123AB'	0001	REAL	123.0	1
'1.23e-4'	0000	REAL	1.23	1
'1.23E-4'	0000	REAL	1.23	1

IN (STRING)	FORMAT (W#16#....)	OUT (data type)	OUT (value)	ENO status
'1.23E-4'	0002	REAL	1.23E-4	1
'12,345.67'	0000	REAL	12345.67	1
'12,345.67'	0001	REAL	12.345	1
'3.4e39'	0002	REAL	W#16#7F800000	1
'-3.4e39'	0002	REAL	W#16#FF800000	1
'1.1754943e-38'	0002	REAL	0.0	1
'12345'	-/-	SINT	0	0
'A123'	-/-	-/-	0	0
''	-/-	-/-	0	0
'++123'	-/-	-/-	0	0
'+-123'	-/-	-/-	0	0

VAL_STRG: Convert numerical value to character string

Description

The "VAL_STRG" instruction is used to convert a numerical value into a character string.

- You specify the value to be converted in the IN input parameter. You decide the format of the numeric value by selecting a data type.
- You query the result of the conversion in the OUT output parameter.

Permitted characters for the conversion are the digits "0" to "9", the decimal point, the decimal comma, notations "E" and "e", and the plus and minus characters. The conversion can be interrupted by invalid characters.

The "VAL_STRG" instruction is not supported by the SCL programming language.

Parameters

The following table shows the parameters of the "VAL_STRG" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN	Input	USINT, SINT, UINT, INT, UDINT, DINT, REAL, LREAL	USINT, SINT, UINT, INT, UDINT, DINT, ULINT, LINT, REAL, LREAL	I, Q, M, D, L, P or constant	Value to be converted
SIZE	Input	USINT	USINT	I, Q, M, D, L, P or constant	Number of character positions
PREC	Input	USINT	USINT	I, Q, M, D, L, P or constant	Number of decimal places
FORMAT	Input	WORD	WORD	I, Q, M, D, L, P or constant	Output format of the characters

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
P	InOut	UINT	UINT	I, Q, M, D, L, P or constant	Character starting at which the result is written.
OUT	Output	STRING	STRING, WSTRING	D, L	Result of the conversion

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

P parameter

With the P parameter, you specify the character in the string starting at which the result is written. If, for example, the value "2" is specified in the P parameter, the converted value is saved starting at the second character of the string.

SIZE and P parameters

Use the SIZE parameter to specify how many characters of the character string will be written. This is counted starting from the character specified in the P parameter. If the output value is shorter than the specified length, the result is written to the character string right-justified. The empty character positions are filled with blanks.

FORMAT parameter

Use the FORMAT parameter to specify how the numerical value is interpreted during conversion and written to the character string. You can specify only tags of the FORMAT data type in the USINT parameter.

The following table shows the possible values of the FORMAT parameter and their meaning:

Value (W#16#....)	Notation	Sign	Decimal representation
0000	Decimal fraction	"-"	"."
0001			","
0002	Exponential		"."
0003			","
0004	Decimal fraction	"+" and "-"	"."
0005			","
0006	Exponential		"."
0007			","
0008 to FFFF	Invalid values		

PREC parameter

Use the PREC parameter to define the number of decimal places when converting floating-point numbers. A maximum precision of seven numbers is supported for numerical values of

the REAL data type. If the value to be converted is an integer, you use the PREC parameter to specify the position where the decimal point will be placed.

Example

The following table shows examples of the conversion of numeric values to a character string.

IN (value)	IN (data type)	P	SIZE	FORMAT (W#16#....)	PREC	OUT (STRING)	ENO status
123	UINT	16	10	0000	0	`xxxxxxx123 C`	1
0	UINT	16	10	0000	2	`xxxxxx0.00 C`	1
12345678	UDINT	16	10	0000	3	`x12345.678 C`	1
12345678	UDINT	16	10	0001	3	`x12345.678 C`	1
123	INT	16	10	0004	0	`xxxxxx+123 C`	1
-123	INT	16	10	0004	0	`xxxxxx-123 C`	1
-0.00123	REAL	16	10	0004	4	`xxx-0.0012 C`	1
-0.00123	REAL	16	10	0006	4	`-1.2300E-3 C`	1
-Inf [1]	REAL	16	10	-/-	4	`xxxxxx-INF C`	0
+Inf [2]	REAL	16	10	-/-	4	`xxxxxx+INF C`	0
NaN [3]	REAL	16	10	-/-	4	`xxxxxxxNaN C`	0
12345678	UDINT	16	6	-/-	3	`xxxxxxxxxx C`	0

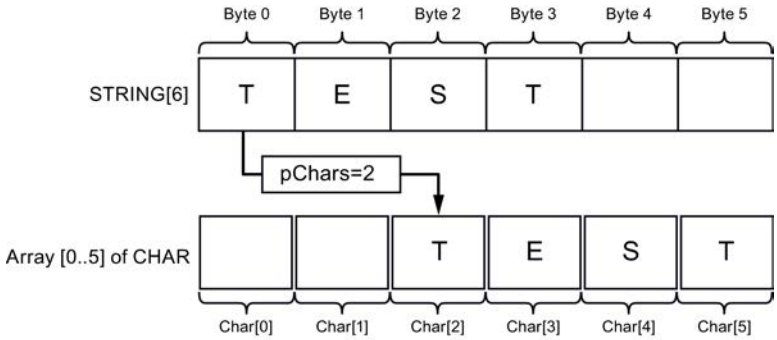
"x" represents blanks
[1]-Inf: Floating-point number representing a negative infinite value.
[2]+Inf: Floating-point number representing a positive infinite value.
[3]NaN: Value returned as the result of invalid math operations.

Strg_TO_Chars: Convert character string to Array of CHAR

Description

Use the instruction to copy a character string (W)STRING into a structure of the data type Array of CHAR or Array of BYTE. Only ASCII characters are valid for the copy process.

- Specify the character string at the input parameter STRG.
- The characters are written to a data type Array of CHAR / BYTE at the parameter CHARS.
  - The number of characters in the structure Array of CHAR / BYTE should be at least the same number as were copied from the character string STRING.
  - If the structure Array of CHAR / BYTE contains less characters than in the character string then the characters are written up to the maximum length of the Array of CHAR / BYTE.
  - If the character string contains a character "$00" then the copy process is only carried out up to the corresponding position.
  - The number of copied characters is output at the parameter CNT.
- Use the parameter PCHARS to define the character in the structure after which Array of CHAR / BYTE should be written.
  - Example: Use the value "2" at the parameter PCHARS if writing should take place after the third character:



  - The lower index limit of the array is used as the standard ("0") (e.g. CHAR[0] at an Array [0..5] of CHAR).

Note

Use of the instruction with S7-1200 V2.0

S7-1200 up to Version 2.0 only supports Array [0 .. n] of CHAR / BYTE. Negative index limits (e.g. Array [-3..2] of CHAR) are not permissible. This restriction is not checked by the software.

Parameters

The following table shows the parameters of the instruction "Strg_TO_Chars":

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
STRG	Input	STRING	STRING, WSTRING	D, L or constant	The copy source Character string with a maximum length of 254 characters.
PCHARS	Input	DINT	DINT	I, Q, M, D, L, P or constant	Position in the structure Array of (W)CHAR / BYTE after which the characters in the character string are written.
CHARS	InOut	VARIANT	VARIANT	D, L	The copy target The characters can be copied to a structure of the data type Array of (W)CHAR or Array of BYTE.
CNT	Output	UINT	UINT	I, Q, M, D, L, P	Number of copied characters.

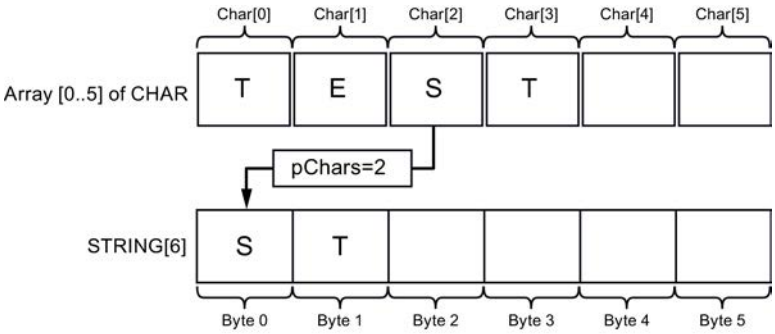
You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

Chars_TO_Strg: Convert Array of CHAR to character string

Description

Use the instruction "Chars_TO_Strg" to copy the character from a structure of a data type Array of CHAR or Array of BYTE to a character string STRING. Only ASCII characters are valid for the copy process.

- At the input parameter CHARS, enter the characters of the structure Array of (W)CHAR / BYTE which are to be copied to a character string.
- The characters are written to a data type STRING at the parameter STRG.
  - The number of characters in the character string STRING should be at least the same number as were copied from the field Array of CHAR.
  - If the character string is shorter than the number of characters in the structure Array of CHAR / BYTE, the maximum length of characters is written in the character string.
  - If the structure Array of CHAR / BYTE contains a character "$00" then the copy process is only carried out up to the corresponding position.
- Use the parameter PCHARS to define the position in the structure Array of CHAR / BYTE after which the character it to be copied.
  - Example: Use the value "2" at the parameter PCHARS if copying is to take place after the third character in the structure:



  - The lower index limit of the array is used as the standard ("0") (e.g. CHAR[0] at Array [0..5] of CHAR).
  - If an index is specified at parameter PCHARS and the index is not contained in the copy source (e.g. "7" at Array [0..5] of CHAR) then the instruction is not executed.

Note

Use of the instruction with S7-1200 V2.0

S7-1200 up to Version 2.0 only supports Array [0 .. n] of CHAR / BYTE. Negative index limits (e.g. Array [-3..2] of CHAR) are not permissible. This restriction is not checked by the software.

Parameters

The following table shows the parameters of the instruction "Chars_TO_Strg":

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
CHARS	Input	VARIANT	VARIANT	D, L	The copy source Structure Array of CHAR / BYTE from which the character is copied.
PCHARS	Input	DINT	DINT	I, Q, M, D, L, P or constant	Position in the structure Array of (W)CHAR / Array of BYTE after which the characters are copied.
CNT	Output	UINT	UINT	I, Q, M, D, L, P	Number of characters to be copied. All characters were copied at "0".
STRG	Output	STRING	STRING, WSTRING	D, L	The copy target Character string

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

MAX_LEN: Determine the length of a character string

Description

A tag of the STRING data type contains two lengths: the maximum length and the current length (this is the number of currently valid characters).

- The maximum length of the character string is specified for each tag in the STRING keyword in square brackets. The number of bytes occupied by a string is 2 greater than the maximum length.
- The current length represents the number of the character places actually used. The current length must be less than or equal to the maximum length.

You use the "MAX_LEN" instruction to query the maximum length of the character string specified at the IN input parameter and output this information as a numerical value at the OUT output parameter.

If errors occur during processing of the instruction, then an empty string will be output.

Note

Reading out the current length

You can also use the LEN (Page 4541) instruction to read out the current length of a string.

Parameter

The following table shows the parameters of the "MAX_LEN" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	STRING	D, L or constant	Character string
OUT	Return	INT	I, Q, M, D, L, P	Maximum number of characters

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

Join: Join multiple strings

Description

The "JOIN" instruction connects multiple strings into an array.

To convert multiple strings into a single one, the instruction provides the following functions:

- Format selection
  You use the first bit in the Mode parameter to select whether you want to use CSV or FSR as the format for the array.
  - With CSV (Comma Separated Values), the contents of the source strings are written consecutively and separated with a delimiter. Example:
    `1963,Miller,John,CitynameA,Roadname`
    `1974,Jackson,Peter,CitynameB,VeryLongRoadname`
  - With FSR (Fixed Size Records), a certain number of characters is defined for each value (each source character string). If fewer characters are required, the characters are padded with delimiters. If more characters are required, the text is truncated. Example:
    `1963Miller,,,,,,,John,,,,,,CitynameA,,,,Roadname,,,,,,`
    `1974Jackson,,,,,,Peter,,,,,CitynameB,,,,VeryLongRoadna`
- Selection of the delimiter for the source strings
  You can use the RecSeparator parameter to select the delimiter to be used for each string. You should select a character based on the content of the input strings at the SrcStruct parameter. If the input strings contain a comma within a string, for example, the comma should not used as a delimiter. The data type that you use for the delimiter must correspond to the destination array at the DstArray parameter in order for the delimiter to be included in what is written in the array.
- Selection of the delimiter for the end of all source strings
  You can use the third bit of the Mode parameter to specify whether an additional character should be written as a delimiter at the end of the copied characters in the destination array (DstArray parameter). Specify the character to be used as a delimiter at the EndSeparator parameter. Make sure to use a character other than the one set for the RecSeparator parameter (delimiter for individual strings). If the two delimiters are indistinguishable, you will obtain unexpected results if you reverse the conversion using the "SPLIT" instruction.

- **Selection of the source strings**
  You specify the source strings at the SrcStruct parameter. For the data type, you can use Array of STRING or Array of WSTRING or structures which only contain the STRING or WSTRING data type. This also applies to user data types or nested structures. You can use these as long as they exclusively contain the data type STRING or WSTRING respectively.
- **Specification of the number of joined strings**
  If you use an Array of STRING or Array of WSTRING at the SrcStruct parameter (source string) (no nested structures), you can use the Count parameter to specify the number of source strings that have been joined to form a single string. If you use a data type other than Array of (W)STRING at the SrcStruct parameter, the Count parameter is ignored. This makes it possible to connect only a part of a large array.
- **Selection of a destination area for writing the array**
  You use the Array of (W)CHAR data type at the DstArray parameter. It is not possible to use the STRING or WSTRING data types here, because the length is limited to 256 characters for STRING.
- **Index of the position in the array (DestArray target parameter)**
  The conversion starts at this position and the instruction reads the Position parameter to determine the position at which the conversion is completed. This enables follow-up calls of the instruction for filling the array.

Parameters

The following table shows the parameters of the instruction "JOIN":

Parameter	Declaration	Data type	Memory area	Description
Mode	Input	DWORD	I, Q, M, D, L or constant	Specifies how the merger to string is performed (see Mode" parameter).
RecSeparator	Input	VARIANT	I, Q, M, D, L	Delimiter for the source strings • With CSV: Character that is used as a delimiter for the individual strings. • With FSR: Character that is used as a fill character for the individual strings.
EndSeparator	Input	VARIANT	I, Q, M, D, L	Delimiter for the end of the conversion Delimiter that is written at the end of the characters if bit 3 = 1 has been set for the Mode parameter.
SrcStruct	Input	VARIANT	I, Q, M, D, L	Pointer to the source strings.
Count	Input	UDINT	I, Q, M, D, L	Number of strings that were joined. The Count parameter can only be used when an Array of (W)STRING has been used at the SrcStruct parameter.
DestArray	InOut	VARIANT	I, Q, M, D, L	Area in which the characters are written after the conversion. Use the Array [0 .. x] of CHAR/WCHAR data type at the DestArray parameter. You can set the length (x) of the array based on the length of the source strings at the SrcStruct parameter.

Parameter	Declaration	Data type	Memory area	Description
Position	InOut	UDINT	I, Q, M, D, L	Index the position in the total string
Ret_Val	Return	UINT	I, Q, M, D, L	Status of instruction (see table "RET_VAL parameter").

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

Mode parameter

Bit	Bit value "0"	Bit value "1"	Description
0	CSV format (Comma Separated Values)	FSR format (Fixed Size Record)	Selection of the format: • With CSV, the source strings are separated by a delimiter in the destination array. • With FSR, the source strings are written into the destination array with the fill characters defined at the RecSeparator parameter.
1	-	-	Irrelevant for the "JOIN" instruction.
2	-	-	Reserved (irrelevant for bit value)
3	Write no additional delimiters.	Write character defined by the EndSeparator parameter at the end of the read characters.	Select whether an additional character is to be written as a delimiter at the end of the characters in the array (DestArray parameter).
4	-	-	Irrelevant for the "JOIN" instruction.

RET_VAL parameter

Error code* (W#16#...)	Explanation
0000	No error.
0001	Additional characters are ignored.
8190	Selection at the Mode parameter is not supported.
8x20	Source strings invalid.
8x50	VARIANT points to empty content.
8x52	VARIANT points to a block that has not been created with optimized access.
8x53	VARIANT points to data structure that is too short.
8x54	Invalid data type
8882	The value at the Position parameter is outside the source string.
8x84	Found additional characters.
80B4	Different data types at parameters SrcStruct (source) and DestArray (destination) or at separators (parameters RecSeparator and EndSeparator).
80B5	Buffer overflow with the instruction. The characters were not fully generated at the DestArray parameter.

Error code* (W#16#...)	Explanation
General error information	See also: GET_ERR_ID: Get error ID locally (Page 3392)

* Note the following for the error codes:

- The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".
- The "x" in the second position of the listed error codes stands for the parameter that caused the error.
  Example: Error code 8352 hex = error occurred at 3rd parameter (EndSeparator), see Parameter table.
- If the error cannot be clearly assigned to a specific parameter, "0" is output.
  Example: The CHAR data type is used for the delimiter (RecSeparator parameter). WCHAR is used as data type for the array at the DestArray parameter. Error code 80B4 is output in this case.

See also

Switching display formats in the program status (Page 2980)

Split: Splitting an array of characters into multiple strings

Description

The "SPLIT" instruction converts an array (Array of CHAR / WCHAR) into several separate strings (Array of STRING / WSTRING or structure).

To convert the array into several strings, specify the following information:

- Selection of array to be read
  You specify the array to be read at the SrcArray parameter. Make sure that the data types used for the input and output parameters match any other parameters used. If you use an array with the CHAR data type at the SrcArray parameter, for example, you must use the CHAR data type for the delimiter (Rec/EndSeparator) and the structure must only contain strings with the STRING data type at the DestStruct parameter.
- Selection of the format for the array to be read
  You use the first bit of the Mode parameter to specify whether the array to be read has the CSV or FSR format (for a definition, see the "JOIN (Page 4531)" instruction).
- Delimiter used for the array to be read
  - If the array to be read has CSV format, specify which delimiter was used at the RecSeparator parameter.
  - If the array to be read has FSR format, specify which fill character was used at the RecSeparator parameter.
- Selection of the delimiter for the end of the total string
  Specify the delimiter after which the reading of the array should stop at the EndSeparator parameter. The "SPLIT" instruction stops at this position and outputs the string that has been found. It should be noted, that the EndSeparator delimiter is evaluated with higher priority than the RecSeparator delimiter. If the delimiter at the EndSeparator parameter is used in a string to be read (between two RecSeparator delimiters), all content after the EndSeparator delimiter is ignored.

- **Specification of the position starting at which the array is read**
  The conversion starts at this position in the array and the instruction reads the Position parameter to determine the position at which the conversion is completed. This enables follow-up calls of the instruction for filling the various strings at the DestStruct parameter.
- **Outputting the number of read strings**
  If you use an Array of STRING at the DestStruct parameter, you can use the Count parameter to output the number of read strings. Only strings with content are counted. If you use a data type other than Array of STRING at the DestStruct parameter, "0" is output at the Count parameter.

Parameters

The following table shows the parameters of the "SPLIT" instruction:

Parameter	Declaration	Data type	Memory area	Description
Mode	Input	DWord	I, Q, M, D, L or constant	Specifies how the split into multiple strings is performed (see "Mode" parameter).
RecSeparator	Input	Variant	I, Q, M, D, L	Delimiter or fill character • With CSV: Character that was used in the array to be read to identify the individual strings. • With FSR: Character that has been used as a fill character in the array to be read.
EndSeparator	Input	Variant	I, Q, M, D, L	Delimiter used to define the end of the total string in the array to be read.
SrcArray	Input	Variant	I, Q, M, D, L	Pointer to the array to be read (Array of CHAR/WCHAR)
DestStruct	InOut	Variant	I, Q, M, D, L	Structure which includes the converted strings (Array of STRING / WSTRING).
Position	InOut	UDInt	I, Q, M, D, L	Position from which the array is to be read at the SrcArray parameter.
Ret_Val	Return	Int	I, Q, M, D, L	Result of instruction execution / error code (see table "Ret_Val parameter")
Count	Output	UDInt	I, Q, M, D, L	Number of strings that were found.

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

Mode parameter

Bit	Bit value "0"	Bit value "1"	Description
0	CSV format (Comma Separated Values)	FSR format (Fixed Size Record)	Basic mode: Selection of CSV or FSR
1	• With CSV: Additional characters are written to the following string. • With FSR: Fill characters remain in string.	• With CSV: Additional characters are removed. • With FSR: Fill characters are removed.	Use bit 1 to specify how you want to deal with additional characters: • With CSV: – If the bit is set, additional characters that do not fit in the destination string are ignored. Example: The instruction writes to a string with a length of 16 characters (STRING[16] data type). No delimiter is contained in the source after the first 16 characters. If bit 1 is set, the extra characters are ignored and the instruction continues to read the array. – If the bit is not set, the instruction stops in such a case and generates an error message at the Ret_Val parameter. • With FSR: – If the bit is set, the fill characters used are not written to the destination string. – If the bit is not set, the fill characters are written to the destination string.
2	-	-	Reserved for use in future versions.
3	Separator is retained.	Separator is removed.	Specification of whether the delimiter is removed at the end of the total string.
4	Leave unwritten strings (STRING) at length.	Set unwritten strings (STRING) to length "0".	Specification of whether unused strings (STRING) at the DestStruct parameter are to be set to the length "0".

RET_VAL parameter

Error code* (W#16#...)	Explanation
0000	No error.
0001	Additional characters are ignored.
8190	Selection at the Mode parameter is not supported.
8x20	Source strings invalid.
8x50	VARIANT points to empty content.
8x52	VARIANT points to a block that has not been created with optimized access.
8x53	VARIANT points to data structure that is too short.
8x54	Invalid data type
8882	The value at the Position parameter is outside the source string.
8x84	Found additional characters.
80B4	Different data types at parameters SrcStruct (source) and DestArray (destination) or at separators (parameters RecSeparator and EndSeparator).
80B5	Buffer overflow with the instruction. The strings were not fully generated at the DestArray parameter.

Error code* (W#16#...)	Explanation
General error information	See also: GET_ERR_ID: Get error ID locally (Page 3392)

* Note the following for the error codes:

- The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".
- The "x" in the second position of the listed error codes stands for the parameter that caused the error.
  Example: Error code 8352 hex = error occurred at 3rd parameter (EndSeparator), see Parameter table.
- If the error cannot be clearly assigned to a specific parameter, "0" is output.
  Example: The CHAR data type is used for the delimiter (RecSeparator parameter). WCHAR is used as data type for the array at the DestArray parameter. Error code 80B4 is output in this case.

See also

Switching display formats in the program status (Page 2980)

ATH: Convert ASCII string to hexadecimal number

Description

You use the instruction "ATH" to convert the ASCII character string specified at the IN input parameter into a hexadecimal number. The result of the conversion is output to the OUT output parameter.

- You can reference the following data types using the pointer in the IN parameter (ASCII): STRING, Array of CHAR, Array of BYTE, WSTRING, Array of WCHAR.
- You can reference the following data types using the pointer in the OUT parameter (hexadecimal): Bit strings, integers, STRING, Array of CHAR, Array of BYTE, WSTRING, Array of WCHAR.

With the N parameter, you specify the number of ASCII characters to be converted. A maximum of 32767 valid ASCII characters can be converted. Only digits "0" to "9", upper case letters "A" to "F", and lower case letters "a" to "f" can be interpreted. All other characters are converted to zeros.

Since 8 bits are required for the ASCII character and only 4 bits for the hexadecimal digit, the output word length is only half of the input word length. The ASCII characters are converted and positioned in the output in the same order as they are read in. If there is an odd number of ASCII characters, the hexadecimal number is padded with zeros in the nibble to the right of the last converted hexadecimal number.

Parameters

The following table shows the parameters of the "ATH" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	VARIANT	I, Q, D, L	Pointer to ASCII character string
N	Input	INT	I, Q, M, D, L or constant	Number of ASCII characters to be converted
RET_VAL	Return	WORD	I, Q, M, D, L	Status of the instruction
OUT	Output	VARIANT	I, Q, M, D, L	Hexadecimal number

You can find additional information on valid data types in "Overview of the valid data types (Page 2236)".

RET_VAL parameter

Error code (W#16#....)*	Description
0000	No error
0007	Invalid character. Only the following ASCII characters may be used: Digits "0" to "9", upper case letters "A" to "F", lower case letters "a" to "f".
8101	Invalid pointer in the IN parameter, e.g., because a non-existing data block is referenced.
8182	Input buffer is too small for data in the N parameter.
8120	Invalid format in the IN parameter.
8151	Non-supported data type in the IN parameter.
8401	Invalid pointer in the OUT parameter, e.g., because a non-existing data block is referenced.
8482	Output buffer is too small for data in the N parameter.
8420	Invalid format in the OUT parameter.
8451	Non-supported data type in the OUT parameter.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

ASCII characters and hexadecimal values

The following table shows the ASCII characters and the corresponding hexadecimal values:

ASCII character	ASCII-coded hexadecimal value	Hexadecimal digit
"0"	30	0
"1"	31	1
"2"	32	2
"3"	33	3
"4"	34	4
"5"	35	5

ASCII character	ASCII-coded hexadecimal value	Hexadecimal digit
"6"	36	6
"7"	37	7
"8"	38	8
"9"	39	9
"A"	41	A
"B"	42	B
"C"	43	C
"D"	44	D
"E"	45	E
"F"	46	F

Example

The following table shows examples of the conversion of ASCII character strings to hexadecimal numbers:

IN	N	OUT	ENO status
'0123'	4	16#0123	1
'123AFx1a23'	10	16#123AF01a23	0

See also

Switching display formats in the program status (Page 2980)

HTA: Convert hexadecimal number to ASCII string

Description

You use the instruction "HTA" to convert the hexadecimal number specified at the IN input into an ASCII character string. The result of the conversion is stored at the address specified in the OUT parameter.

- You can reference the following data types using the pointer in the IN parameter (hexadecimal): Bit strings, integers, STRING, Array of CHAR, Array of BYTE, WSTRING, Array of WCHAR.
- You can reference the following data types using the pointer in the OUT parameter (ASCII): STRING, Array of CHAR, Array of BYTE , WSTRING, Array of WCHAR.

With the N parameter, you specify the number of hexadecimal bytes to be converted. Since 8 bits are required for the ASCII character and only 4 bits for the hexadecimal digit, the output value is twice as long as the input value. Each nibble of the hexadecimal number is converted to a character while maintaining the original order.

A maximum of 32767 characters can be written to the ASCII character string. The result of the conversion is represented by the digits "0" to "9" and upper-case letters "A" to "F".

If the complete result of the conversion cannot be displayed in the OUT parameter, the result is will only be partially written to the parameter.

Parameters

The following table shows the parameters of the "HTA" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	VARIANT	I, Q, M, D, L	Start address of the hexadecimal digits
N	Input	INT	I, Q, M, D, L or constant	Number of hexadecimal bytes to be converted
RET_VAL	Return	WORD	I, Q, M, D, L	Error message
OUT	Output	VARIANT	I, Q, D, L	Address at which the result is stored.

You can find additional information on valid data types in "Overview of the valid data types (Page 2236)".

RET_VAL parameter

Error code* (W#16#....)	Description
0000	No error
8101	Invalid pointer at the IN parameter, e.g., because a non-existing data block is referenced.
8182	Input buffer is too small for data in the N parameter.
8120	Invalid format in the IN parameter.
8151	Non-supported data type in the IN parameter.
8401	Invalid pointer in the OUT parameter, e.g., because a non-existing data block is referenced.
8482	Output buffer is too small for data in the N parameter.
8420	Invalid format in the OUT parameter.
8451	Non-supported data type in the OUT parameter.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

ASCII characters and hexadecimal values

The following table shows the ASCII characters and the corresponding hexadecimal values:

Hexadecimal digit	ASCII-coded hexadecimal value	ASCII character
0	30	"0"
1	31	"1"
2	32	"2"
3	33	"3"
4	34	"4"

Hexadecimal digit	ASCII-coded hexadecimal value	ASCII character
5	35	"5"
6	36	"6"
7	37	"7"
8	38	"8"
9	39	"9"
A	41	"A"
B	42	"B"
C	43	"C"
D	44	"D"
E	45	"E"
F	46	"F"

Example

The following table shows examples of the conversion of hexadecimal numbers to ASCII character strings:

IN	N	OUT	ENO status
W#16#0123	2	'0123'	1
16#123AF01023	4	'123AF010'	0

See also

Switching display formats in the program status (Page 2980)

Other instructions

LEN: Determine the length of a character string

Description

A tag of the STRING data type contains two lengths: the maximum length and the current length (this is the number of currently valid characters).

- The maximum length of the character string is specified for each tag in the STRING keyword in square brackets. The number of bytes occupied by a string is 2 greater than the maximum length.
- The current length represents the number of the character places actually used. The current length must be less than or equal to the maximum length.

You use the instruction "LEN" to query the current length of the character string specified at the IN input parameter and output this information as a numerical value at the OUT output parameter. An empty string ('') has the length zero.

If errors occur during processing of the instruction, then an empty string will be output.

Note

Reading out the maximum length

You can also use the MAX_LEN (Page 4530) instruction to read out the current length of a string.

Parameters

The following table shows the parameters of the "LEN" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	STRING	D, L or constant	Character string
OUT	Return	INT, DINT, REAL, LREAL	I, Q, M, D, L	Number of valid characters

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

CONCAT: Combine character strings

Description

You use the instruction "CONCAT" to combine the character string at the IN1 input parameter with the character string at the IN2 input parameter. The result is output in the OUT output parameter with the STRING format. If the resulting character string is longer than the tag specified in the OUT output parameter, then the resulting character string will be limited to the available length.

If errors occur during processing of the instruction and the OUT output parameter can be written, an empty string will be output.

Parameters

The following table shows the parameters of the "CONCAT" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	STRING	D, L or constant	Character string
IN2	Input	STRING	D, L or constant	Character string
OUT	Return	STRING	D, L	Resulting character string

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

LEFT: Read the left character of a character string

Description

You use the instruction "LEFT" to extract a partial string beginning with the first character of the string at the IN input parameter. You specify the number of characters to be extracted in the L parameter. The extracted characters are output in the OUT output parameter with STRING format.

If the number of characters to be extracted is greater than the current length of the character string, the OUT output parameter returns the input character string as a result. If the L parameter contains the value "0" or the input value is an empty string, an empty string will be returned. If the value in the L parameter is negative, an empty string will be output.

If errors occur during processing of the instruction and the OUT output parameter can be written, an empty string will be output.

Parameters

The following table shows the parameters of the "LEFT" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	STRING	D, L or constant	Character string
L	Input	BYTE, INT, SINT, USINT	I, Q, M, D, L or constant	Number of characters to be extracted
OUT	Return	STRING	D, L	Extracted partial string

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

RIGHT: Read the right characters of a character string

Description

You use this instruction to extract the last L character in a character string in the input parameter IN. You specify the number of characters to be extracted in the L parameter. The extracted characters are output in the OUT output parameter with STRING format.

If the number of characters to be extracted is greater than the current length of the character string, the OUT output parameter returns the input character string as a result. If the L parameter contains the value "0" or the input value is an empty string, an empty string will be returned. If the value in the L parameter is negative, an empty string will be output.

If errors occur during processing of the instruction and the OUT output parameter can be written, an empty string will be output.

Parameters

The following table shows the parameters of the "RIGHT" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	STRING	D, L or constant	Character string
L	Input	BYTE, INT, SINT, USINT	I, Q, M, D, L or constant	Number of characters to be extracted
OUT	Return	STRING	D, L	Extracted partial string

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

MID: Read middle characters of a character string

Description

You use this instruction to extract a portion of the character string in the IN input parameter. With the P parameter, you specify the position of the first character to be extracted. With the L parameter, you define the length of the character string to be extracted. The extracted partial character string is output to the OUT output parameter.

The following rules must be observed when executing the instruction:

- If the number of characters to be extracted exceeds the current length of the character string in the IN input parameter, a partial character string will be output, starting from character position P and continuing to the end of the character string.
- If the character position specified in the P parameter falls outside the current character string length in the IN input parameter, an empty character string will be output in the OUT output parameter.
- If the value of the P or L parameter equals zero or is negative, an empty character string will be output in the OUT output parameter.

If errors occur during processing of the instruction and the OUT output parameter can be written, an empty string will be output.

Parameters

The following table shows the parameters of the "MID" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	STRING	D, L or constant	Character string
L	Input	BYTE, INT, SINT, USINT	I, Q, M, D, L or constant	Length of the string to be extracted
P	Input	BYTE, INT, SINT, USINT	I, Q, M, D, L or constant	Position of the first character to be extracted (first character = 1)
OUT	Return	STRING	D, L	Extracted partial string

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

DELETE: Delete characters in a character string

Description

You use this instruction to delete a portion of the character string in the IN input parameter. With the P parameter, you specify the position of the first character to be deleted. You specify the number of characters to be deleted in the L parameter. The remaining partial character string is output to the OUT output parameter with STRING format.

The following rules must be observed when executing the instruction:

- If the value in the P parameter is less than or equals zero, an empty character string will be output in the OUT output parameter.
- If the value in the P parameter is greater than the current length of the character string in the IN input, the input character string will be returned in the OUT output parameter.
- If the value in the L parameter equals zero, the input character string will be returned in the OUT output parameter.
- If the number of characters to be deleted at the L parameter is greater than the length of the character string in the IN parameter, an empty character string will be output.
- If the value in the L parameter is negative, an empty character string will be output.

If errors occur during processing of the instruction and the OUT output parameter can be written, an empty string will be output.

Parameter

The following table shows the parameters of the "DELETE" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	STRING	D, L or constant	Character string
L	Input	BYTE, INT, SINT, USINT	I, Q, M, D, L or constant	Number of characters to be deleted
P	Input	BYTE, INT, SINT, USINT	I, Q, M, D, L or constant	Position of first character to be deleted
OUT	Return	STRING	D, L	Resulting character string

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

INSERT: Insert characters in a character string

Description

You use this instruction to insert the character string in the IN2 input parameter to the character string in the IN1 input parameter. With the P parameter, you specify the position of the character starting at which the characters are inserted. The result is output in the OUT output parameter with the STRING format.

The following rules must be observed when executing the instruction:

- If the value in the P parameter exceeds the current length of the character string in the IN1 input parameter, the character string of the IN2 input parameter will be appended to the character string of the IN1 input parameter.
- If the value at the P parameter is zero, the character string at the IN2 parameter followed by the character string at the IN1 parameter will be output in the OUT output parameter.
- If the value in the P parameter is negative, an empty character string will be output in the OUT output parameter.
- If the resulting character string is longer than the tag specified in the OUT output parameter, the resulting character string will be limited to the available length.

Parameter

The following table shows the parameters of the "INSERT" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	STRING	D, L or constant	Character string
IN2	Input	STRING	D, L or constant	String to insert
P	Input	BYTE, INT, SINT, USINT	I, Q, M, D, L or constant	Insert position
OUT	Return	STRING	D, L	Resulting character string

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

REPLACE: Replace characters in a character string

Description

You use this instruction to replace a portion of the character string in the IN1 input with the character string in the IN2 input. You specify the position of the first character to be replaced in the P parameter. You specify the number of characters to be replaced in the L parameter. The result is output in the OUT output parameter with the STRING format.

The following rules must be observed when executing the instruction:

- If the value in the P parameter is less than or equals zero, an empty character string will be output in the OUT output parameter.
- If the value in the L parameter is less than zero, an empty character string will be output in the OUT output parameter.
- If P equals one, the character string in the IN1 input will be replaced beginning with (and including) the first character.
- If the value in the P parameter exceeds the current length of the character string in the IN1 input parameter, the character string of the IN2 input parameter will be appended to the character string of the IN1 input parameter.
- If the resulting character string is longer than the tag specified in the OUT output parameter, the resulting character string will be limited to the available length.
- If the value at parameter L is zero, characters are not replaced but inserted. The same conditions apply as with the instruction INSERT. See also: INSERT: Insert characters in a character string (Page 4546)

Parameters

The following table shows the parameters of the "REPLACE" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	STRING	D, L or constant	String with characters to be replaced.
IN2	Input	STRING	D, L or constant	String with characters to be inserted.
L	Input	BYTE, INT, SINT, USINT	I, Q, M, D, L or constant	Number of characters to be replaced
P	Input	BYTE, INT, SINT, USINT	I, Q, M, D, L or constant	Position of first character to be replaced
OUT	Return	STRING	D, L	Resulting character string

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

FIND: Find characters in a character string

Description

You can use this instruction to search through the character string in the IN1 input parameter for a specific character or a specific sequence of characters.

- You specify the value to be searched for in the IN2 input parameter. The search is made from left to right.
- The position of the first occurrence is output in the OUT output parameter. If the search returns no match, the value "0" will be output in the OUT output parameter.

If an invalid character is specified at the IN2 parameter or if an error occurs during processing, the value "0" is output at the OUT parameter.

Parameters

The following table shows the parameters of the "FIND" instruction:

Parameter	Declaration	Data type		Memory area	Description
		S7-1200	S7-1500		
IN1	Input	STRING	STRING, WSTRING	D, L or constant	String searched through
IN2	Input	STRING, CHAR	STRING, WSTRING, CHAR, WCHAR	D, L or constant (For (W)CHAR also I, Q, M)	Characters to search for
OUT	Return	INT	INT	I, Q, M, D, L	Character position

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

Runtime information

GetSymbolName: Read out a tag on the input parameter

Description

The "GetSymbolName" instruction is used to read the name of a tag interconnected at the parameter via the input parameter of a block.

If a block is used multiple times in the project and called each time by a different tag, you can use the "GetSymbolName" instruction to evaluate the name of the calling tag. The process value of the tag is irrelevant in this case.

- You specify the input parameters of the block interface at the VARIABLE parameter. Use only an interface parameter for this parameter and no PLC or data block tags.
- You can limit the length of the read tag name using the SIZE parameter. If the name has been truncated, this is indicated by the characters "..." (Unicode character 16#2026) appears at the end of the name. Note that the "..." character itself has a length of 1. If you enter 6 as the maximum length at the SIZE parameter, for example, and read the tag name "MyPLCTag" via the block interface, the following is output: "MyPLC...".
- The name read is output at the OUT parameter.

Parameters

The following table shows the parameters of the "GetSymbolName" instruction:

Parameter	Declaration	Data type	Memory area	Description
VARIABLE	Input	PARAMETER	L	Input parameters of the block interface
SIZE	Input	UINT	I, Q, M, D, L	Length limit for the read names
OUT	Return	WSTRING	I, Q, M, D, L	Name of the tag that was read via the block interface

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

See also

Switching display formats in the program status (Page 2980)

GetInstanceName: Read out name of the read instance

Description

You can use the "GetInstanceName" instruction to read the name of the instance data block within a function block.

- You can specify how many characters of the instance name should be read out using the SIZE parameter. If you use the "0" value for SIZE, the entire name is read.
- The name of the instance data block is written to the OUT parameter. If the name of the instance data block is longer than the maximum length of WSTRING (254 characters), the name is truncated.

Parameters

The following table shows the parameters of the "GetInstanceName" instruction:

Parameter	Declaration	Data type	Memory area	Description
SIZE	Input	DINT	I, Q, M, D, L or constant	Number of characters up to which the name of the instance data block is to be read
OUT	Output	WSTRING	I, Q, M, D, L	Read name of the instance data block

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

See also

Switching display formats in the program status (Page 2980)

GetBlockName: Read out name of the block

Description

You can use the "GetBlockName" instruction to read the name of the block in which the instruction is called.

- You can specify how many characters of the block name should be read using the SIZE parameter. If you use the "0" value for SIZE, the entire name is read.
- The name of the block is written at the OUT parameter. If the name of the block is longer than the maximum length of WSTRING (254 characters), the name is truncated.

Parameter

The following table shows the parameters of the "GetBlockName" instruction:

Parameter	Declaration	Data type	Memory area	Description
SIZE	Input	UINT	I, Q, M, D, L or constant	Number of characters up to which the name of the instance data block is to be read.
RET_VAL	Output	WSTRING	I, Q, M, D, L	Read name of the instance data block

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

See also

Switching display formats in the program status (Page 2980)

Process image

UPDAT_PI: Update the process image inputs

Description

With the instruction, you update the OB 1 process image (=process image partition 0) of the inputs or a process image partition of the inputs of the inputs defined by configuration.

If you have configured repeated signaling of I/O access errors for the system-side process image update, then the selected process image will be updated constantly.

Otherwise, this update will only be performed if the selected process image partition is not updated by the system, in other words:

- If you have not assigned this process image partition to an interrupt OB
  or
- If you selected process image partition 0 and have disabled updating of the OB 1 process image partition during configuration.

Note

Each logical address that you have assigned to a process image partition of the inputs per configuration no longer belongs to the OB 1 process image of the inputs.

When you update a process image partition with "UPDAT_PI", you must not perform a simultaneous update with the "SYNC_PI (Page 4553)" instruction.

System-side updating of the OB 1 process image of the inputs and the process image partitions of inputs that you have assigned to an interrupt OB takes place independently of "UPDAT_PI" calls.

Parameters

The following table shows the parameters of the "UPDAT_PI"instruction:

Parameter	Declaration	Data type	Memory area	Description
PART	Input	PIP	I, Q, M, D, L or constant	Number of the process image partition of the inputs to be updated. Maximum value range (depending on the CPU): 0 to 31 (0 means OB 1 process image, n where 1 <= n <= 31 means process image partition n).
RET_VAL	Return	INT	I, Q, M, D, L, P	Error information
FLADDR	Output	WORD	I, Q, M, D, L, P	Address of the first byte to cause an error if an access error has occurred.

You can find additional information on valid data types in "Overview of the valid data types (Page 2236)".

RET_VAL parameter

Error code* (W#16#...)	Explanation
0000	No error occurred.
8090	Illegal value for PART parameter
8091	The specified process image partition was not yet defined or is not located in the permitted process image area on the CPU.
8092	The process image partition is being updated by the system and cannot be used for the PART parameter.
8093	The process image partition update is being processed in another OB.
80A0	An access error was detected when accessing the I/O.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 3392)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

Note

If you use the instruction for process image partitions of DP standard slaves for which you have defined a consistency area greater than 32 bytes, the error codes of the "DPRD_DAT (Page 4600)" instruction are also possible.

See also

Switching display formats in the program status (Page 2980)

UPDAT_PO: Update the process image outputs

Description

You use the instruction to transfer the signal states of the OB 1 process image (= process image partition 0) of the outputs or a process image partition of the outputs defined per configuration to the output modules.

If you have specified a consistency range for the selected process image partition, corresponding data is transferred as consistent data to the respective I/O module.

Note

Each logical address you have assigned to a process image partition of the outputs during configuration no longer belongs to the OB 1 process image of the outputs.

Outputs that you update with "UPDAT_PO" must not be updated simultaneously with the "SYNC_PO (Page 4555)" instruction.

The OB 1 process image of the outputs and the process image partitions of the outputs that you have assigned to an interrupt OB are transferred by the system to the output modules independently of "UPDAT_PO" calls.

Parameters

The following table shows the parameters of the "UPDAT_PO" instruction:

Parameter	Declaration	Data type	Memory area	Description
PART	Input	PIP	I, Q, M, D, L or constant	Number of the process image partition of the outputs to be transferred. Maximum value range (depending on the CPU): 0 to 31. (0 means OB 1 process image, n where 1 <= n <= 31 means process image partition n)
RET_VAL	Return	INT	I, Q, M, D, L, P	Error information
FLADDR	Output	WORD	I, Q, M, D, L, P	Address of the first byte to cause an error if an access error has occurred.

You can find additional information on valid data types in "Overview of the valid data types (Page 2236)".

RET_VAL parameter

Error code* (W#16#...)	Explanation
0000	No error occurred.
8090	Illegal value for PART parameter
8091	The specified process image partition was not yet defined or is not located in the permitted process image area on the CPU.
8092	The process image partition is being updated by the system and cannot be used for the PART parameter.
8093	The process image partition update is being processed in another OB.
80A0	An access error was detected when accessing the I/O.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 3392)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

Note

If you use the instruction for process image partitions of DP standard slaves for which you have defined a consistency area greater than 32 bytes, the error codes of the "DPWR_DAT (Page 4601)" instruction are also possible.

See also

Switching display formats in the program status (Page 2980)

SYNC_PI: Synchronize the process image inputs

Description

The "SYNC_PI" is used to update the process image partition of inputs in isochronous mode. A user program linked to a DP cycle or PN send cycle can use this instruction to update input data acquired in a process image partition of the inputs isochronously and consistently.

Call

"SYNC_PI" is interruptible and can only be called in OBs 61, 62, 63 and 64.

Note

A call of the "SYNC_PI" instruction in OBs 61 to 64 is only permitted if you have assigned the affected process image partition to the associated OB in the HW configuration.

A process image partition that you update with "SYNC_PI" must not be updated simultaneously with the "UPDAT_PI (Page 4550)" instruction.

Parameters

The following table shows the parameters of the "SYNC_PI" instruction:

Parameter	Declaration	Data type	Memory area	Range of values	Description
PART	Input	PIP	I, Q, M, D, L or constant	1 to 31	Number of the process image partition of the inputs to be updated isochronously.
RET_VAL	Return	INT	I, Q, M, D, L, P	-	Error information
FLADDR	Output	WORD	I, Q, M, D, L, P		Address of the first byte to cause an error if an access error has occurred.

You can find additional information on valid data types in "Overview of the valid data types (Page 2236)".

RET_VAL parameter

Error code (W#16#...)*	Explanation
0000	No error occurred.
0001	Consistency warning. The update of the process image partition was distributed over two DP or PN cycles. However, the data in one slave or IO device were consistently transferred.
8090	Illegal value on PART parameter or updating the specified process image partition of the inputs is not permitted in this OB. The process image partition of the inputs was not updated.
8091	The specified process image partition was not yet defined or is not located in the permitted process image area on the CPU. The process image partition of the inputs was not updated.
8092	The process image partition is being updated by the system and cannot be used for the PART parameter.
8093	The process image partition update is being processed in another OB.
80A0	During updating an access error was detected. The affected inputs were set to "0".
80A1	The update time is after the permitted access window. The process image partition of the inputs was not updated. The DP or PN cycle is too short to ensure enough time for processing the instruction. You must therefore increase the TDP (also known as T_DC), Ti and To timers.
80A2	Access error with consistency warning In the case of update of the specified process image partition an access error with simultaneous consistency warning was detected. • The data of the incorrect inputs was not read by the I/O. In the process image partition of the inputs the involved inputs are set to zero. • The update of the process image partition of the not involved by an access error input data was distributed over two DP and PN cycles.
80C1	The update time is before the permitted access window. The process image partition of the inputs was not updated.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 3392)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

Note

If you use the "SYNC_PI" instruction for process image partitions of DP standard slaves for which you have defined a consistency area greater than 32 bytes, the error codes of the "DPRD_DAT (Page 4600)" instruction are also possible.

See also

Switching display formats in the program status (Page 2980)

SYNC_PO: Synchronize the process image outputs

Description

The instruction "SYNC_PO" is used to update a process image partition of outputs in isochronous mode. An application program linked to a DP cycle or a PN send cycle can use this instruction for the consistent isochronous transfer of the computed output data of a process image partition from outputs to the I/O devices.

Call

"SYNC_PO" is interruptible and can only be called in OBs 61, 62, 63 and 64.

Note

A call of the "SYNC_PO" instruction in OBs 61 to 64 is only permitted if you have assigned the affected process image partition to the associated OB in the HW configuration.

A process image partition that you update with "SYNC_PO" must not be updated simultaneously with the "UPDAT_PO (Page 4552)" instruction.

Parameters

The following table shows the parameters of the "SYNC_PO" instruction:

Parameter	Declaration	Data type	Memory area	Range of values	Description
PART	Input	PIP	I, Q, M, D, L or constant	1 to 31	Number of the process image partition of the outputs to be updated isochronously.
RET_VAL	Return	INT	I, Q, M, D, P	-	If an error occurs while the instruction is being executed, the return value contains an error code.
FLADDR	Output	WORD	I, Q, M, D, L, P	-	Address of the first byte that has caused the error.

You can find additional information on valid data types in "Overview of the valid data types (Page 2236)".

RET_VAL parameter

Error code (W#16#...)*	Explanation
0000	No error occurred.
0001	Consistency warning. The update of the process image partition was distributed over two DP or PN cycles. However, the data in one slave or IO device were consistently transferred.
8090	Illegal value in PART parameter, or updating the specified process image partition of the outputs is not permitted in this OB. Outputs were not transferred to the I/O. The process image partition of the outputs remains unchanged.
8091	The specified process image partition was not yet defined or is not located in the permitted process image area on the CPU. Outputs were not transferred to the I/O. The process image partition of the outputs remains unchanged.
8092	The process image partition is being updated by the system and cannot be used for the PART parameter.
8093	The process image partition update is being processed in another OB.
80A0	In the case of update of the specified process image partition of the outputs an access error was detected. Incorrect outputs were not transferred to the I/O. During process image partition of the outputs, these outputs remain unchanged.
80A1	Access error with consistency warning In the case of update of the specified process image partition of the outputs an access error with simultaneous consistency warning was detected. • The data of the incorrect outputs were not transferred to the I/O. During process image partition of the outputs, the involved outputs remain unchanged. • The update of the process image partition of the not involved by an access error input data was distributed over two DP and PN cycles.
80A2	The update time is after the permitted access window. Outputs were not transferred to the I/O. The process image partition of the outputs remains unchanged.
80C1	The update time is before the permitted access window. Outputs were not transferred to the I/O. The process image partition of the outputs remains unchanged.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 3392)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

Note

If you use the "SYNC_PO" instruction for process image partitions of DP standard slaves for which you have defined a consistency area greater than 32 bytes, the error codes of the "DPWR_DAT (Page 4601)" instruction are also possible.

See also

Switching display formats in the program status (Page 2980)

Distributed I/O

RDREC: Read data record

Description

You use the instruction "RDREC" to read the data record with the number INDEX from the module addressed using the ID. This may be a module in a central rack or a distributed module (PROFIBUS DP or PROFINET IO).

- Use the parameter ID to select from which module a data record is to be read. Use only the hardware identifier of the module for the "ID" parameter.
- Use the INDEX parameter to select the information to be read from the module. Which data records can be read with which data record number depends on the module. For additional information, see the manual to the specific module.
- The data records readable via "RDREC" have a different length. Use MLEN to specify the maximum number of data record bytes you want to read. If length "0" is selected at the parameter MLEN, the complete data record is written at the parameter RECORD.
- The target range RECORD should have at least the length of MLEN bytes. If you read out the full data record with MLEN=0, use the maximum length of the data record for RECORD. The structure (configuration, data types and length) that you use at the parameter RECORD, also depends on which data record is read out by which module.
- The value TRUE for the output parameter VALID indicates that the data record was successfully transferred to the target range RECORD. In this case, the LEN output parameter contains the length of the read data in bytes.
- If an error has occurred during transfer of the data record, this is indicated by the output parameter ERROR. In this case, the output parameter STATUS contains the error information.

Note

The interface of the "RDREC" instruction is identical to the "RDREC" FB defined in "PROFIBUS Guideline PROFIBUS Communication and Proxy Function Blocks according to IEC 61131-3".

Functional description

"RDREC" is an instruction that works asynchronously, which means its execution extends over multiple calls. You start the data record transfer by calling "RDREC" with REQ= 1.

The job status is displayed via the output parameter BUSY and the two central bytes of output parameter STATUS. The two central bytes of STATUS correspond to the RET_VAL output parameter of the instructions that operate asynchronously.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 3150).

The transfer of the data record is complete when the output parameter BUSY has the value FALSE .

Parameter

The following table shows the parameters of the "RDREC" instruction:

Parameter	Declaration	Data type*	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, T, C or constant	REQ = 1: Transfer data record
ID	Input	HW_IO	I, Q, M, L or constant	Hardware identifier of the hardware module (DP/PROFINET IO) The number is assigned automatically and is stored in the properties of the module or of the interface in the hardware configuration.
INDEX	Input	BYTE, DINT, INT, SINT, UINT, USINT, WORD	I, Q, M, D, L or constant	Data record number
MLEN	Input	BYTE, UINT, USINT	I, Q, M, D, L or constant	Maximum length in bytes of the data record information to be read
VALID	Output	BOOL	I, Q, M, D, L	New data record was received and is valid.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The reading process is not yet complete.
ERROR	Output	BOOL	I, Q, M, D, L	ERROR = 1: An error occurred during the reading process.
STATUS	Output	DWORD	I, Q, M, D, L	Block status or error information
LEN	Output	UINT	I, Q, M, D, L	Length of the read data record information
RECORD	InOut	VARIANT	I, Q, M, D, L	Target range for the data record read.
* There is no implicit conversion in STL, which is why the range of valid data types may be limited. During programming in STL, please note the valid data types. These are displayed in the parameter tooltip.				

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Note

If you use "RDREC" to read a data record for PROFINET IO, then negative values in the INDEX, MLEN, and LEN parameters will be interpreted as an unsigned 16-bit integer.

Parameter STATUS

For interpretation of the STATUS parameter, see Parameter STATUS (Page 4566).

See also

Basics of block access (Page 2175)

WRREC: Write data record

Description

The instruction "WRREC" is used to transfer the RECORD data record to the component addressed using ID. This may be a module in a central rack or a distributed component (PROFIBUS DP or PROFINET IO).

- Use LEN to specify the length of the data record to be transmitted in bytes. The selected length of the source range RECORD should have at least the length of LEN bytes.
- The value TRUE at output parameter DONE indicates that the data record has been successfully transferred.
- If an error has occurred during transfer of the data record, this is indicated by the output parameter ERROR. In this case, the output parameter STATUS contains the error information.

Note

The interface of the "WRREC" instruction is identical to the "WRREC" FB defined in "PROFIBUS Guideline PROFIBUS Communication and Proxy Function Blocks according to IEC 61131-3".

Functional description

"The WRREC" instruction works asynchronously, that is, its execution extends over multiple calls. You start the data record transfer by calling "WRREC" with REQ = 1.

The job status is displayed via the output parameter BUSY and the two central bytes of output parameter STATUS. The two central bytes of STATUS correspond to the RET_VAL output parameter of the instructions that operate asynchronously.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 3150).

Note that you must assign the same value to the actual parameter of RECORD for all "WRREC" calls that belong to one and the same job. The same applies to the actual parameters of LEN.

The transfer of the data record is complete when the output parameter BUSY has the value FALSE.

Parameter

The following table shows the parameters of the instruction "WRREC":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, T, C or constant	REQ= 1: Transfer data record
ID	Input	HW_IO	I, Q, M, L or constant	ID number of the hardware component (DP/PROFINET IO) The number is assigned automatically and is stored in the properties of the component or of the interface in the hardware configuration.
INDEX	Input	DINT	I, Q, M, D, L or constant	Data record number
LEN	Input	BYTE, UINT, USINT	I, Q, M, D, L or constant	(hidden) Maximum length of the data record to be transferred in bytes
DONE	Output	BOOL	I, Q, M, D, L	Data record was transferred
BUSY	Output	BOOL	I, Q, M, D, L	BUSY= 1: The writing process is not yet complete.
ERROR	Output	BOOL	I, Q, M, D, L	ERROR = 1: An error occurred during the writing process.
STATUS	Output	DWORD	I, Q, M, D, L	Block status or error information For interpretation of the STATUS parameter, see Parameter STATUS (Page 4566).
RECORD	InOut	VARIANT	I, Q, M, D, L	Data record

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Note

If you use "WRREC" to write a data record for PROFINET IO, negative values in the INDEX and LEN parameters will be interpreted as an unsigned 16-bit integer.

STATUS parameter

For interpretation of the STATUS parameter, see Parameter STATUS (Page 4566).

GETIO: Read process image

Description

You use the instruction "GETIO" to consistently read out all inputs of a DP standard slave/PROFINET IO device. The instruction "GETIO" calls the instruction "DPRD_DAT (Page 4600)". If there was no error during the data transmission, the data that have been read are entered in the target range indicated by INPUTS .

The target range must have the same length that you configured for the selected component.

If you read from a DP standard slave with a modular configuration or with several DP identifiers, you can only access the data of one component/DP identifier at the configured start address with a "GETIO" call.

Parameter

The following table shows the parameters of the instruction "GETIO":

Parameter	Declaration	Data type	Memory area	Description
ID	Input	HW_SUBMODULE	I, Q, M, D, L or constant	Hardware ID of the DP standard slave / PROFINET IO device.
STATUS	Output	DWORD	I, Q, M, D, L	Contains the error information of "DPRD_DAT (Page 4600)" in the form DW#16#40xxxx00.
LEN	Output	INT	I, Q, M, D, L	Amount of data read in bytes
INPUTS	InOut	VARIANT	I, Q, M, D	Target range for the read data. It must have the same length as the range that you configured for the selected DP standard slave / PROFINET IO device. Only the BYTE data type is permitted.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

STATUS parameter

See also: DPRD_DAT: Read consistent data of a DP standard slave (Page 4600).

SETIO: Transfer process image

Description

You use the instruction "SETIO" to consistently transfer data from the source range defined by the parameter OUTPUTS to the addressed DP standard slave / PROFINET IO device, and, if necessary, to the process image (if you have configured the relevant address area of the DP standard slave / PROFINET IO device as a consistent range in a process image). "SETIO" calls the instruction "DPWR_DAT (Page 4601)".

The source range must have the same length that you configured for the selected component.

In the case of a DP standard slave / PROFINET IO device with modular configuration or with several DP identifiers, you can only access one DP identifier / component per "SETIO" call.

Parameters

The following table shows the parameters of the "SETIO" instruction:

Parameter	Declaration	Data type	Memory area	Description
ID	Input	HW_SUBMODULE	I, Q, M, D, L or constant	Hardware ID of the DP standard slave / PROFINET IO device.
STATUS	Output	DWORD	I, Q, M, D, L	Contains the error information of "DPWR_DAT (Page 4601)" in the form DW#16#40xxxx00.
OUTPUTS	InOut	VARIANT	I, Q, M, D	Source range for the data to be written. It must have the same length as the range that you configured for the selected DP standard slave / PROFINET IO device. Only the BYTE data type is permitted.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter STATUS

See also: DPWR_DAT: Write consistent data of a DP standard slave (Page 4601).

GETIO_PART: Read process image area

Description

You use the instruction "GETIO_PART" to consistently read out a related part of the inputs of an IO module. "GETIO_PART" calls the instruction "DPWR_DAT (Page 4600)".

Use the ID input parameter to select the IO module by means of the hardware ID.

You use the OFFSET and LEN parameters to specify the portion of the process image area to be read. If the input area spanned by OFFSET and LEN is not completely covered by the module, the block returns the error code DW#16#40**80B7**00.

The length of the target range must be larger than or equal to the amount of bytes to be read:

- If there was no error during the data transmission, ERROR receives the value FALSE. The data that are read are written to the target range defined at the INPUTS parameter.
- If there was no error during the data transmission, ERROR receives the value TRUE. The STATUS parameter receives the error information from "DPRD_DAT".
- If the target range is greater than LEN, then the first LEN bytes of the target range will be written. ERROR receives the value FALSE.

Parameters

The following table shows the parameters of the "GETIO_PART" instruction:

Parameter	Declaration	Data type	Memory area	Description
ID	Input	HW_SUBMODULE	I, Q, M, D, L or constant	Hardware identifier of the module
OFFSET	Input	INT	I, Q, M, D, L or constant	Number of the first byte to be read in the process image for the component (smallest possible value: 0).
LEN	Input	INT	I, Q, M, D, L or constant	Number of bytes to be read.
STATUS	Output	DWORD	I, Q, M, D, L	Contains the error information of "DPRD_DAT" in the form DW#16#40xxxx00, if ERROR = TRUE.
ERROR	Output	BOOL	I, Q, M, D, L	Error display: ERROR = TRUE if an error occurs when "DPRD_DAT" is called.
INPUTS	InOut	VARIANT	I, Q, M, D	Target range for read data: If the target range is greater than LEN, the first LEN bytes of the target range are written.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter STATUS

See "RET_VAL" parameter of the "DPRD_DAT (Page 4600)" instruction.

SETIO_PART: Transfer process image area

Description

You can use the "SETIO_PART" instruction to consistently write data from the source area spanned by OUTPUTS to the outputs of an IO module. "SETIO_PART" calls the instruction "DPWR_DAT (Page 4601)".

Use the ID input parameter to select the IO module by means of the hardware ID.
You use the OFFSET and LEN parameters to specify the portion of the process image area to be written for the components addressed by means of ID. If the output area spanned by OFFSET and LEN is not completely covered by the module, the block returns the error code DW#16#40**80B7**00.

The length of the source range must be greater than or equal to the number of bytes to be written:

- If there was no error during the data transmission, ERROR receives the value FALSE.
- If there was an error during the data transmission, ERROR receives the value TRUE, and STATUS receives the error information of "DPWR_DAT".
- If the source range is greater than LEN, the first LEN bytes from OUTPUTS are transferred. ERROR receives the value FALSE.

Parameter

The following table shows the parameters of the instruction "SETIO_PART":

Parameter	Declaration	Data type	Memory area	Description
ID	Input	HW_SUBMODULE	I, Q, M, D, L or constant	Hardware identifier of the IO module.
OFFSET	Input	INT	I, Q, M, D, L or constant	Number of the first byte to be written in the process image for the component (smallest possible value: 0).
LEN	Input	INT	I, Q, M, D, L or constant	Number of bytes to be written.
STATUS	Output	DWORD	I, Q, M, D, L	Contains the error information of "DPWR_DAT" in the form DW#16#40xxxx00, if ERROR = TRUE.
ERROR	Output	BOOL	I, Q, M, D, L	Error display: ERROR = TRUE if an error occurs when "DPWR_DAT" is called.
OUTPUTS	InOut	VARIANT	I, Q, M, D	Source range for the data to be written: If the source range is greater than LEN, the first LEN bytes are transferred from OUTPUTS.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

Parameters STATUS and ERROR

See instruction "DPWR_DAT (Page 4601)".

RALRM: Receive interrupt

Description RALRM

Description

The instruction receives an interrupt with all corresponding information from an I/O module (centralized configuration) or from a module of a DP slave or PROFINET IO device and supplies this information to its output parameters.

The information in the output parameters contains the start information of the called OB as well as information of the interrupt source.

For a central configuration, the configuration of the data structure of the target range AINFO corresponds to the data structure of PROFINET IO.

Call "RALRM" only within the interrupt OB started by the CPU operating system as a result of the I/O interrupt that is to be examined.

Note

If you call "RALRM" in an OB whose start event is not an I/O interrupt, the instruction will provide correspondingly reduced information in its outputs.
Make sure to use different instance DBs when you call "RALRM" in different OBs. If you evaluate data resulting from an "RALRM" call outside of the associated interrupt OB, you should moreover use a separate instance DB per OB start event.

Note

The interface of the "RALRM" instruction is identical to the "RALRM" FB defined in "PROFIBUS Guideline PROFIBUS Communication and Proxy Function Blocks according to IEC 61131-3".

Calling RALRM

"RALRM" can be called in three operating modes (MODE parameter). These are explained in the table below.

MODE	RALRM ...
0	... shows the component that triggered the interrupt in the ID output parameter and writes TRUE in the NEW output parameter.
1	... writes all output parameters, independent of the interrupt triggering component.
2	... checks whether the component specified in the F_ID input parameter has triggered the interrupt. • If not, NEW = FALSE • If yes, NEW = TRUE and all other output parameters are written.

Parameters

The following table shows the parameters of the "RALRM" instruction:

Parameter	Declaration	Data type	Memory area	Description
MODE	Input	INT	I, Q, M, D, L or constant	Mode
F_ID	Input	HW_IO	D, L or constant	Hardware identifier of the module. The number is assigned automatically and is stored in the hardware configuration properties of the component or interface.
MLEN	Input	UINT	I, Q, M, D, L or constant	Maximum length of the interrupt information to be received in bytes. With MLEN = 0, all data specified at the AINFO parameter are read.

Parameter	Declaration	Data type	Memory area	Description
NEW	Output	BOOL	I, Q, M, D, L	A new interrupt was received.
STATUS (Page 4566)	Output	DWORD	I, Q, M, D, L	Error code
ID	Output	HW_IO	I, Q, M, D, L	Hardware identifier of the module from which the interrupt was received.
LEN	Output	UINT	I, Q, M, D, L	Length of the received interrupt information
TINFO (Page 4570)	InOut	VARIANT	I, Q, M, D, L	Destination area for OB start and management information
AINFO (Page 4582)	InOut	VARIANT	I, Q, M, D, L	Destination area for header information and additional interrupt information For AINFO , you should provide a length of at least MLEN bytes.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Note

If you select a target range (Page 4588)TINFO or AINFO that is too short, RALRM cannot enter the full information.

Parameter STATUS

Description

The STATUS output parameter contains error information. If it is interpreted as ARRAY[1...4] of BYTE, the error information has the following structure:

Field element	Name	Meaning
STATUS[1]	Function_Num	• B#16#00, if no error • Function ID from DPV1-PDU: In the event of an error, B#16#80 is output (in the event of an error reading a data record B#16#DE and writing a data record B#16#DF). If no DPV1 protocol element is used, then B#16#C0 will be output.
STATUS[2]	Error_Decode	Location of the error ID
STATUS[3]	Error_Code_1	Error ID
STATUS[4]	Error_Code_2	Manufacturer-specific error ID extension

Field element STATUS[2]

STATUS[2] can have the following values:

Error_Decode (B#16#...)	Source	Meaning
00 to 7F	CPU	No error or no warning
80	DPV1	Error according to IEC 61158-6
81 to 8F	CPU	B#16#8x shows an error in the xth call parameter of the instruction.
FE, FF	DP profile	Profile-specific error

Field element STATUS[3]

STATUS[3] can have the following values:

Error_Decode (B#16#...)	Error_Code_1 (B#16#...)	Explanation according to DVP1	Meaning
00	00		No error, no warning
70	00	reserved, reject	Initial call; no active data record transfer
	01	reserved, reject	Initial call; data record transfer has started
	02	reserved, reject	Intermediate call; data record transfer already active
80	81		The system data type at parameter TINFO does not match the call environment of the instruction. The used system data type must match the organization block in the user program (example: for a time-delay interrupt OB you need the system data type TI_Delay).
	90	reserved, pass	Invalid logical start address
	92	reserved, pass	Illegal type for VARIANT pointer
	93	reserved, pass	The DP component addressed via ID or F_ID is not configured.
	96		The "RALRM (Page 4564)" cannot supply the OB start information, management information, header information, or additional interrupt information. For OBs 4x, 55, 56, 57, 82 and 83, you can use the "DPNRM_DG (Page 4614)" instruction to read the current diagnostic frame of the relevant DP slave asynchronously (address information from OB start information).
	A0	read error	Negative acknowledgment while reading the module
	A1	write error	Negative acknowledgment when writing to the module
	A2	module failure	DP protocol error at layer 2 (e.g., slave failure or bus problems)
	A3	reserved, pass	• PROFIBUS DP: DP protocol error with Direct-Data-Link-Mapper or User-Interface/User • PROFINET IO: General CM error
	A4	reserved, pass	Communication on PBUS+ disrupted

Error_Decode (B#16#...)	Error_Code_1 (B#16#...)	Explanation according to DVP1	Meaning
	A5	reserved, pass	–
	A7	reserved, pass	DP slave or module is occupied (temporary error
	A8	version conflict	DP slave or module reports non-compatible versions
	A9	feature not supported	Function is not supported by DP slave or module
	AA to AF	user specific	DP slave or module reports a manufacturer-specific error in its application. Please check the documentation from the manufacturer of the DP slave or module.
	B0	invalid index	Data record not known in module Illegal data record number ≥ 256
	B1	write length error	Error at length specification: • With "RALRM (Page 4564)": length error in AINFO (Page 4582) • With "RDREC (Page 4557)": length error in MLEN • With "WRREC (Page 4559)": length error in LEN
	B2	invalid slot	The configured slot is not assigned.
	B3	type conflict	Actual module type does not match specified module type
	B4	invalid area	DP slave or module reports access to an invalid area
	B5	state conflict	DP slave or module not ready
	B6	access denied	DP slave or module denies access
	B7	invalid range	DP slave or module reports an invalid range for a parameter or value
	B8	invalid parameter	DP slave or module reports an invalid parameter
	B9	invalid type	DP slave or module reports an invalid type With "RDREC (Page 4557)": buffer too small (subsets cannot be read) With "WRREC (Page 4559)": buffer too small (subsets cannot be written)
	BA to BF	user specific	DP slave or module reports a manufacturer-specific error when accessing. Please check the documentation from the manufacturer of the DP slave or module.
	C0	read constrain conflict	With "WRREC (Page 4559)": the data can only be written when the CPU is in STOP mode. Note: this means that writing by the user program is not possible. You can only write the data online with PG/PC. With "RDREC (Page 4557)": the module routes the data record, but either no data is present or the data can only be read when the CPU is in STOP mode. Note: if data can only be read when the CPU is in STOP mode, then an evaluation by the user program is not possible. In this case, you can only read the data online with PG/PC.
	C1	write constrain conflict	The data of the previous write job on the module for the same data record have not yet been processed by the module.

Error_Decode (B#16#...)	Error_Code_1 (B#16#...)	Explanation according to DVP1	Meaning
	C2	resource busy	The module is currently processing the maximum possible number of jobs for a CPU.
	C3	resource unavailable	The required operating resources are currently occupied.
	C4		Internal temporary error. Job could not be executed. Repeat the job. If this error occurs often, check your installation for sources of electrical interference.
	C5		DP slave or module not available.
	C6		Data record transfer was canceled due to priority class cancellation
	C7		Job aborted due to warm or cold restart on the DP master
	C8 to CF		DP slave or module reports a manufacturer-specific resource error. Please check the documentation from the manufacturer of the DP slave or module.
	Dx	user specific	DP slave specific. Refer to the description of the DP slave.
81	00 to FF		Error in the initial call parameter (with "RALRM (Page 4564)": MODE)
	00		Illegal operating mode
82	00 to FF		Error in the second call parameter
:	:		:
88	00 to FF		Error in the eighth call parameter (with "RALRM (Page 4564)": TINFO (Page 4570))
	01		Wrong syntax ID
	23		Quantity structure exceeded or target range too small
	24		Wrong range ID
	32		DB/DI no. out of user range
	3A		DB/DI no. is NULL for area ID DB/DI or specified DB/DI does not exist
89	00 to FF		Error in the ninth call parameter (with "RALRM (Page 4564)": AINFO (Page 4582))
	01		Wrong syntax ID
	23		Quantity structure exceeded or target range too small
	24		Wrong range ID
	32		DB/DI no. out of user range
	3A		DB/DI no. is NULL for area ID DB/DI or specified DB/DI does not exist
8A	00 to FF		Error in the 10th call parameter
:	:		:
8F	00 to FF		Error in the 15th call parameter
FE, FF	00 to FF		Profile-specific error

Field element STATUS[4]

With DPV1 errors, the DP master passes on STATUS[4] to the CPU and the instruction. Without DPV1 error, this value is set to 0, with the following exceptions for "RDREC":

- STATUS[4] contains the target range length from RECORD, if MLEN > the target range length from RECORD
- STATUS[4]=MLEN, if the actual data record length < MLEN < the target range length from RECORD
- STATUS[4]=0, if STATUS[4]> 255 would have to be set

In PROFINET IO, STATUS[4] has the value "0".

Parameter TINFO

Data structure of the destination area TINFO

The data structure of the destination area TINFO contains the start information of the organization block in which "RALRM" is currently being called.

The TINFO destination area can contain the start information with standard access or optimized access. The format of the start information in the TINFO destination area must always match the start information of the corresponding organization block.

- The start information of an OB with standard access is always stored in the first 20 bytes of the "Temp" section in the block interface. Use the "TI_Classic" data structure for this.
- The start information of an OB with optimized access is always written to the "Input" section. Use a data structure for the specific OB type for these OBs.

Changing the block access (standard/optimized) also changes the block interface.

The following table provides an overview of the data structures that are used depending on the organization block at the TINFO parameter.

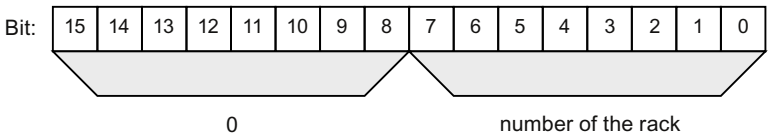
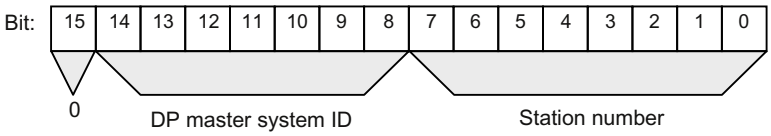
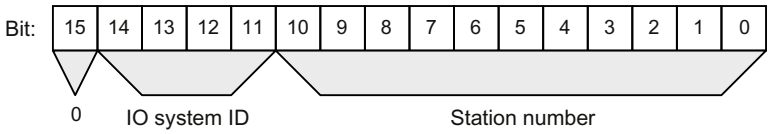
Name of the data structure	S7-1200 CPUs as of version	S7-1500 CPUs as of version	Data structure used for:
Data structure for organization blocks with standard access			
TI_Classic	-	V1	Organization blocks without optimized block access.
Data structure for organization blocks with optimized block access			
TI_ProgramCycle	V2	V1	Cycle OB (program cycle)
TI_Startup	V2	V1	Startup OB (startup)
TI_Delay	V2	V1	Time delay interrupt
TI_Cyclic	V2	V1	Cyclic interrupt OB
TI_HWInterrupt	V2	V1	Hardware interrupt OB
TI_TimeError	V2	V1	Time error OB
TI_DiagnosticInterrupt	V2	V1	Diagnostic error interrupt OB
TI_PlugPullModule	V2	V1	Pull/plug of modules OB
TI_StationFailure	V2	V1	Rack or station failure OB

Name of the data structure	S7-1200 CPUs as of version	S7-1500 CPUs as of version	Data structure used for:
TI_ProgIOAccessError	V2	V1	• Programming error OB • I/O access error OB
TI_TimeOfDay	V2	V1	Time-of-day interrupt OB
TI_SynchCycle	-	V1	Synchronous cycle interrupt OB
TI_Submodule	V2	V1	• Status interrupt OB • Update interrupt OB • OB for manufacturer or profile-specific interrupt

Data structure for organization blocks with standard access

The following shows the TI_Classic data structure:

Parameter	Data type	Byte	Description
TI_Classic			Data structure for organization blocks with optimized block access.
Bytes 0 to 19: Start information of the OB in which "RALRM" is currently being called.*			
EV_CLASS	BYTE	0	Event class Example OB1: • Bits 0-3: ID of the event (1 = incoming event) • Bits 4-7: Event class (1 = Event class 1)
EV_NUM	BYTE	1	Event number (depending on OB type) Example OB1 (SCAN_1): • SCAN_1 = 1 for first call • SCAN_1 = 3 for all additional calls
PRIORITY	BYTE	2	Priority class
NUM	BYTE	3	OB number
TYP2_3	BYTE	4	Additional information Different information is stored in the BYTES "TYP2_3" and "TYPE 1" depending on the OB type used. What these are is described in the documentation for the organization blocks. Example (OB1): • TYP2_3: OB1_RESERVED_1 (reserved) • TYP1: OB1_RESERVED_2 (reserved)
TYP1	BYTE	5	
ZI1	WORD	6 to 7	Additional information Different information is stored in "ZI1" depending on the OB type used. What these are is described in the documentation for the organization blocks. Example (OB1): • ZI1: OB1_PREV_CYCLE (runtime of previous cycle in ms)

Parameter	Data type	Byte	Description
ZI2_3	DWORD	8 to 11	Additional information Different information is stored in "ZI2_3" depending on the OB type used. What these are is described in the documentation for the organization blocks. Example (OB1): • ZI2: OB1_MIN_CYCLE (minimum cycle time (ms) since the last startup) • ZI3: OB1_MAX_CYCLE (maximum cycle time (ms) since the last startup)
OB_DATE_TIME	DATE_AND_TIME (DT)	12 to 19	Date and time-of-day when the OB was called.
Bytes 20 and 21: Address information			
address	WORD	20 and 21	Address information like S7-300/400 CPUs: • In a central configuration, the rack number (0-31):  • For a distributed configuration with PROFIBUS DP: – DP master system ID (1-31) – Station number (0-127).  • In a distributed configuration with PROFINET IO: – The last two positions in the PROFINET IO system ID (0-15). To obtain the complete PROFINET IO system ID, you must add 100 (decimal) to it. – Station number (0-2047). 
Bytes 22 to 31: Management information			
slv_prfl	BYTE	22	Slave profile like S7-300/400 CPUs: • In a central configuration: 0 (data record 0 or data record 1) • In a distributed configuration: – Bit 0 to 3: Slave type - 0000: DP (structure data record 0) - 0001: DPS7 (structure data record 0 or data record 1) - 0010: DPS7 V1 (structure data record 0 or data record 1) - 0011: DPV1 (structure according to PROFIBUS DP standard) - 0100 – 0111: Reserved - 1000: PROFINET IO (structure according to PROFINET IO standard) - from 1001: Reserved – Bit 4 to 7: Profile type (reserved)

Parameter	Data type	Byte	Description
intr_type	BYTE	23	Interrupt info type like S7-300/400 CPUs: • In a central configuration: 0 • In a distributed configuration: – Bit 0 to 3: Interrupt info type - 0000: Interrupt comes from a configured remote module - 0001: Interrupt comes from a non-DPV1 slav / non IO device or a slot that is not configured - 0010: Interrupt generated in the CPU - from 0011: Reserved – Bit 4 to 7: Structure version - 0000: Initial - from 0001: Reserved
flags1	BYTE	24	Flags of the PROFIBUS DP master interface module / PROFINET IO controller interface module like S7-300/400 CPUs: • In a central configuration: 0 • In a distributed configuration: – Bit 0 = 0: Interrupt originating from an integrated interface module (PROFINET IO or PROFIBUS DP) – Bit 0 = 1: Interrupt originating from an external interface module (PROFINET IO or PROFIBUS DP) – Bit 1 to 7: Reserved
flags2	BYTE	25	Flags of the PROFIBUS DP master interface module / PROFINET IO controller interface module like S7-300/400 CPUs: • In a central configuration: 0 • For a distributed configuration with PROFIBUS DP: – Bit 0: EXT_DIAG_FLAG from the diagnostics message frame, or 0 if this bit does not exist in the interrupt The bit is 1 if the DP slave is faulty. – Bit 1 to 7: Reserved • In a distributed configuration with PROFINET IO: – Bit 0: ARDiagnosisState or 0 if there is no information in the interrupt. The bit is 1 if the IO device is faulty. – Bit 1 to 7: Reserved
id	UINT	26 and 27	Management information • In a central configuration: 0 • For a distributed configuration with PROFIBUS DP: PROFIBUS ID number as unique identifier of the PROFIBUS DP slave • In a distributed configuration with PROFINET IO: PROFINET IO device ID number as unique identifier of the PROFINET IO device
manufacturer	UINT	28 and 29	Manufacturer ID (only in a distributed configuration with PROFINET IO).
instance	UINT	30 and 31	Ident number of the instance (only in a distributed configuration with PROFINET IO).
*The start information depends on the OB used. You can find the start information of each OB type at the interface or in the documentation of the OB.			

Data structures for organization blocks with optimized block access

The data structures for organization blocks with optimized block access have the following format:

- Bytes 0 to 3: Format of the start information, class and number of the called OB (same structure for all data structures).
- Bytes 4 to 19: Optimized start information (structure depends on the OB type). The data in bytes 4 to 19 correspond to the structure and content of the corresponding OB interface.
- Bytes 20 to 31: In addition, the address and management information for certain OBs. The data in bytes 20 to 31 correspond to the data of 20 to 31 bytes of the TI_Classic data structure.

The format of the data structures is described in the following tables.

Table 9-42 **Cycle OB: Data structure TI_ProgramCycle**

Parameter		Data type	Byte	Description
TI_ProgramCycle				Data structure for cycle OB
	SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
	OB_Class	USINT	1	OB class (=1)
	OB_Nr	UINT	2	OB number (1...32767).
	Initial_Call	BOOL	4	= TRUE in the first call of this OB: • Transition from STOP or HOLD to RUN • After reloading
	Remanence	BOOL	5	= TRUE, if retentive data are available.

Table 9-43 **Startup OB: Data structure TI_Startup**

Parameter		Data type	Byte	Description
TI_Startup				Data structure for startup OB
	SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
	OB_Class	USINT	1	OB class (=100)
	OB_Nr	UINT	2	OB number (1...32767).
	LostRetentive	BOOL	4	=TRUE, if the contents of retentive data areas have been lost.
	LostRTC	BOOL	5	=TRUE, if the time-of-day of the real-time clock has been lost.

Table 9-44 **Time delay interrupt OB: Data structurer TI_Delay**

Parameter		Data type	Byte	Description
TI_Delay				Data structure for time delay interrupt OB
	SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information

Parameter		Data type	Byte	Description
	OB_Class	USINT	1	OB class (=20)
	OB_Nr	UINT	2	OB number (1...32767).
	Sign	WORD	4	User ID: Input parameter SIGN from the call of the "SRT_DINT (Page 4706)" instruction.

Table 9-45 **Cyclic interrupt OB: Data structure TI_Cyclic**

Parameter		Data type	Byte	Description
TI_Cyclic				Data structure for cyclic interrupt OB
	SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
	OB_Class	USINT	1	OB class (=30)
	OB_Nr	UINT	2	OB number (1...32767).
	Initial_Call	BOOL	4	= TRUE in the first call of this OB • At the transition from STOP or HOLD to RUN • After reloading
	Event_Count	INT	6	Number of discarded start events since the last start of this OB.

Table 9-46 **Hardware interrupt OB: Data structure TI_HWInterrupt**

Parameter		Data type	Byte	Description
TI_HWInterrupt				Data structure for hardware interrupt OB
	SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
	OB_Class	USINT	1	OB class (=40)
	OB_Nr	UINT	2	OB number (1...32767).
	LADDR	HW_IO	4	Hardware identifier of the module that triggers the hardware interrupt.
	USI	WORD	6	Identifier for future extensions (not user-relevant).
	IChannel	USINT	8	Number of the channel that triggered the hardware interrupt.
	EventType	BYTE	9	Identifier for the event type associated with the event triggering the interrupt (e.g., rising edge). You can find this ID in the description of the respective module.
	address	WORD	20	See "address" parameter for the TI_Cassic. data structure
	slv_prfl	BYTE	22	See "slv_prfl" parameter for the TI_Cassic data structure.
	intr_type	BYTE	23	See "intr_type" parameter for the TI_Cassic data structure.
	flags1	BYTE	24	See "flags1" parameter for the TI_Cassic data structure.
	flags2	BYTE	25	See "flags2" parameter for the TI_Cassic data structure.
	id	UNIT	26	See "id" parameter for the TI_Cassic data structure.
	manufacturer	UNIT	28	See "manufacturer" parameter for the TI_Cassic data structure.
	instance	UNIT	30	See "instance" parameter for the TI_Cassic data structure.

Table 9-47 **Time error OB: Data structure TI_TimeError**

Parameter	Data type	Byte	Description
TI_TimeError			Data structure for time error OB
SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
OB_Class	USINT	1	OB class (=80)
OB_Nr	UINT	2	OB number (1...32767).
Csg_OBnr	OB_ANY	4	Number of the OB that was being executed when the time error occurred.
Fault_ID	BYTE	7	Error code. Possible values: • B#16#01: Cycle time exceeded • B#16#02: The requested OB is still being processed. • B#16#05: Expired time-of-day interrupt due to time jump. • B#16#06: Expired time-of-day interrupt upon re-entry into RUN after HOLD. • B#16#07: Overflow of the OB request buffer for the current priority class. • B#16#08: Isochronous mode interrupt - time error. • B#16#09: Interrupt loss due to high interrupt load • B#16#0B: Technology synchronization interrupt - time error
Csg_Prio	UNIT	8	Priority of the OB that was being executed when the time error occurred.

Table 9-48 **Diagnostic interrupt OB: Data structure TI_DiagnosticInterrupt**

Parameter	Data type	Byte	Description
TI_DiagnosticInterrupt			Data structure for diagnostic interrupt OB
SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
OB_Class	USINT	1	OB class (=82)
OB_Nr	UINT	2	OB number (1...32767).
LADDR	HW_ANY	4	Hardware identifier of the hardware object that triggered the diagnostic interrupt.
IO_State	WORD	6	Status of the hardware object: • Bit 0: Good • Bit 1: Disabled • Bit 2: Maintenance required • Bit 3: Maintenance demanded • Bit 4: Error • Bit 5: Not accessible • Bit 6: Qualified • Bit 7: Not available
Channel	UINT	8	Channel number
MultiError	BOOL	10	=TRUE, if there is more than one error.
address	WORD	20	See "address" parameter for the TI_Cassic. data structure
slv_prfl	BYTE	22	See "slv_prfl" parameter for the TI_Cassic data structure.
intr_type	BYTE	23	See "intr_type" parameter for the TI_Cassic data structure.

Parameter		Data type	Byte	Description
	flags1	BYTE	24	See "flags1" parameter for the TI_Cassic data structure.
	flags2	BYTE	25	See "flags2" parameter for the TI_Cassic data structure.
	id	UINT	26	See "id" parameter for the TI_Cassic data structure.
	manufacturer	UINT	28	See "manufacturer" parameter for the TI_Cassic data structure.
	instance	UINT	30	See "instance" parameter for the TI_Cassic data structure.

Table 9-49 **Pull/plug interrupt OB: Data structure** TI_PlugPullModule

Parameter		Data type	Byte	Description
TI_PlugPullModule				Data structure for pull/plug interrupt OB
	SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
	OB_Class	USINT	1	OB class (=83)
	OB_Nr	UINT	2	OB number (1...32767).
	LADDR	HW_IO	4	Hardware identifier of the affected module or submodule
	Event_Class	BYTE	6	• B#16#38: (Sub)module plugged • B#16#39: (Sub)module pulled or not responding
	Fault_ID	BYTE	7	Error code The following table shows the events that cause the start of the pull/plug interrupt OB. • With Event_Class = B#16#38 - (sub-)module plugged: – B#16#54: PROFINET IO submodule inserted and matches configured submodule – B#16#55: PROFINET IO submodule inserted, but does not match configured submodule – B#16#56: PROFINET IO submodule inserted; but error in module parameter assignment – B#16#57: PROFINET IO submodule or module inserted, but with a problem or maintenance – B#16#58: PROFINET IO submodule, access error corrected • With Event_Class = B#16#39 - (sub-)module pulled or cannot be reached: – B#16#51: PROFINET IO module removed – B#16#54: PROFINET IO submodule removed
	address	WORD	20	See "address" parameter for the TI_Cassic. data structure
	slv_prfl	BYTE	22	See "slv_prfl" parameter for the TI_Cassic data structure.
	intr_type	BYTE	23	See "intr_type" parameter for the TI_Cassic data structure.
	flags1	BYTE	24	See "flags1" parameter for the TI_Cassic data structure.
	flags2	BYTE	25	See "flags2" parameter for the TI_Cassic data structure.
	id	UINT	26	See "id" parameter for the TI_Cassic data structure.
	manufacturer	UINT	28	See "manufacturer" parameter for the TI_Cassic data structure.
	instance	UINT	30	See "instance" parameter for the TI_Cassic data structure.

Table 9-50 **Rack error OB: Data structure TI_StationFailure**

Parameter		Data type	Byte	Description
TI_StationFailure				Data structure for rack error OB
	SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
	OB_Class	USINT	1	OB class (=86)
	OB_Nr	UINT	2	OB number (1...32767).
	LADDR	HW_Device	4	Hardware identifier of the defective hardware object.
	Event_Class	BYTE	6	• B#16#38: (Sub)module plugged • B#16#39: (Sub)module pulled or not responding
	Fault_ID	BYTE	7	Error code The following table shows the events that cause the start of the pull/plug interrupt OB. • With Event_Class = B#16#38 - (sub-)module plugged: – B#16#54: PROFINET IO submodule inserted and matches configured submodule – B#16#55: PROFINET IO submodule inserted, but does not match configured submodule – B#16#56: PROFINET IO submodule inserted; but error in module parameter assignment – B#16#57: PROFINET IO submodule or module inserted, but with a problem or maintenance – B#16#58: PROFINET IO submodule, access error corrected • With Event_Class = B#16#39 - (sub-)module pulled or cannot be reached: – B#16#51: PROFINET IO module removed – B#16#54: PROFINET IO submodule removed
	address	WORD	20	See "address" parameter for the TI_Cassic. data structure
	slv_prfl	BYTE	22	See "slv_prfl" parameter for the TI_Cassic data structure.
	intr_type	BYTE	23	See "intr_type" parameter for the TI_Cassic data structure.
	flags1	BYTE	24	See "flags1" parameter for the TI_Cassic data structure.
	flags2	BYTE	25	See "flags2" parameter for the TI_Cassic data structure.
	id	UINT	26	See "id" parameter for the TI_Cassic data structure.
	manufacturer	UINT	28	See "manufacturer" parameter for the TI_Cassic data structure.
	instance	UINT	30	See "instance" parameter for the TI_Cassic data structure.

Table 9-51 **Programming error OB / I/O access error OB: Data structure TI_ProgIOAccessError**

Parameter		Data type	Byte	Description*
TI_ProgIOAccessError				Data structure for programming error OB and I/O access error OB
	SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information

Parameter	Data type	Byte	Description*
OB_Class	USINT	1	OB class: • =121 for programming error OB • =122 for I/O access error OB
OB_Nr	UINT	2	OB number (1...32767).
BlockNr	UINT	4	Number of the block in which the programming error occurred.
Reaction	USINT	6	• 0: Ignore error • 1: Substitute bad value • 2: Skip command
Fault_ID	BYTE	7	Error code • B#16#21: BCD conversion error • B#16#22: Area length error when reading • B#16#23: Area length error when writing • B#16#24: Range error when reading • B#16#25: Range error when writing • B#16#26: Error in timer no. • B#16#27: Error in counter no. • B#16#28: Read access to a byte, word or double word with a pointer whose bit address is not zero • B#16#29: Write access to a byte, word or double word with a pointer whose bit address is not zero • B#16#:30: Write access to a write-protected global DB • B#16#31: Write access to a write-protected instance DB • B#16#32: DB number error when accessing a global DB • B#16#33: DB number error when accessing an instance DB • B#16#34: Number error during FC call • B#16#35: Number error during FB call • B#16#42: I/O access error, reading • B#16#43: I/O access error, writing • B#16#3A: Access to a DB that is not loaded; the DB number is located in the permitted area • B#16#3C: Access to an FC that is not loaded; the FC number is within the permissible range • B#16#3D: Access to an instruction (SFC) that is not loaded; the SFC number is within the permissible range • B#16#3E: Access to an FB that is not loaded; the FB number is within the permissible range • B#16#3F: Access to an SFB that is not available; the SFB number is within the permissible range
BlockType	USINT	8	Type of block in which the error has occurred: • OB: B#16#88 • FC: B#16#8C • FB: B#16#8E

Parameter	Data type	Byte	Description*
Area	USINT	9	Memory area in which the incorrect access occurred: • Local data: B#16#40 to 4E, 86, 87, 8E, 8F, C0 to CE • Process image input: B#16#01 • Process image output: B#16#02 • Technology DB: B#16#04 • I: B#16#81 • Q: B#16#82 • M: B#16#83 • DB: B#16#84, 85, 8A, 8B
DBNr	DB_ANY	10	DB no. if AREA = DB (global DB or instance DB) Only relevant for programming error OB.
Csg_OBNr	OB_ANY	12	OB number: • 121: Programming error OB • 122: I/O access error OB
Csg_Prio	USINT	14	OB priority
Width	USINT	15	Type of access during which the error occurred: • Bit: B#16#00 • Byte: B#16#01 • Word: B#16#02 • DWord: B#16#03 • LWord: B#16#04
* Only certain output values are possible depending on the OB (I/O access error OB or programming error OB) from which information is read.			

Table 9-52 **Time-of-day interrupt OB: Data structure TI_TimeOfDay**

Parameter	Data type	Byte	Description
TI_TimeOfDay			Data structure for time-of-day interrupt OB
SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
OB_Class	USINT	1	OB class (=10)
OB_Nr	UINT	2	OB number (1...32767).
CaughtUp	BOOL	4	=TRUE if an OB call was executed subsequently because time was adjusted ahead
SecondTime	BOOL	5	=TRUE if an OB was called a second time because time was adjusted back. Note: SecondTime is set only once in situations in which the time is set back.

Table 9-53 **Isochronous mode interrupt OB: Data structure TI_SynchCycle**

Parameter	Data type	Byte	Description
TI_SynchCycle			Data structure for isochronous mode interrupt OB
SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information

Parameter		Data type	Byte	Description
	OB_Class	USINT	1	OB class (=61)
	OB_Nr	UINT	2	OB number (1...32767).
	Initial_Call	BOOL	4	= TRUE in the first call of this OB: • Transition from STOP or HOLD to RUN • After reloading
	IO _System	USINT	5	Number of the distributed I/O system triggering the interrupt
	Event_Count	INT	6	• = n: Number of lost cycles • = -1: An unknown number of cycles has been lost (e.g., because cycle has changed).
	PIP_Input	BOOL	10	=TRUE: The associated process image of the inputs is up-to-date
	PIP_Output	BOOL	11	=TRUE: The associated process image of the outputs was transferred to the outputs in good time after the last cycle
	SyncCycleTime	LTIME	16	Calculated cycle time

Table 9-54 **Status interrupt OB / update interrupt OB for manufacturer or profile-specific interrupt: Data structure TI_Submodule**

Parameter		Data type	Byte	Description
TI_Submodule				Data structure for status interrupt OB / update interrupt OB for manufacturer or profile-specific interrupt
	SI_Format	USINT	0	Format of the start information: • 16#FF: None • 16#FE: Optimized start information
	OB_Class	USINT	1	OB class • =55 for status interrupt OB • =56 for update interrupt OB • =57 for manufacturer or profile-specific interrupt OB
	OB_Nr	UINT	2	OB number (1...32767).
	LADDR	HW_IO	4	Hardware address of the component triggering the interrupt
	Slot	UINT	6	Slot number of the component triggering the interrupt
	Specifier	WORD	8	Interrupt specifier from the interrupt frame
	address	WORD	20	See "address" parameter for the TI_Cassic. data structure
	slv_prfl	BYTE	22	See "slv_prfl" parameter for the TI_Cassic data structure.
	intr_type	BYTE	23	See "intr_type" parameter for the TI_Cassic data structure.
	flags1	BYTE	24	See "flags1" parameter for the TI_Cassic data structure.
	flags2	BYTE	25	See "flags2" parameter for the TI_Cassic data structure.
	id	UINT	26	See "id" parameter for the TI_Cassic data structure.
	manufacturer	UINT	28	See "manufacturer" parameter for the TI_Cassic data structure.
	instance	UINT	30	See "instance" parameter for the TI_Cassic data structure.

Parameter AINFO

Data structure of the destination area AINFO with interrupts from PROFIBUS DP

Byte	Meaning	
0 to 3	Header information, for exact description, see below	
4 to 63	Additional interrupt information: data for the respective interrupt:	
	distributed:	ARRAY[0] to ARRAY[59]

Structure of the header information with interrupts from PROFIBUS DP

Byte	Data type	Meaning		
0	BYTE	Length of the received interrupt information in bytes		
		central: distributed:	4 to 224 4 to 63	
1	BYTE	central:	Reserved	
		distributed:	ID for the interrupt type	
			1: 2: 3: 4: 5: 6: 31	Diagnostics interrupt Process interrupt Pull interrupt Plug interrupt Status interrupt Update interrupt Failure of an expansion device, DP master system or DP station
			32 to 126:	Manufacturer-specific interrupt
2	BYTE	Slot number of the component that triggered the interrupt		
3	BYTE	central:	Reserved	
		distributed:	Specifier	
			Bits 0 and 1:	0: No further information; 1: Incoming event, fault at slot 2: Outgoing event, fault at slot cleared 3: Outgoing event, slot still faulty
			Bit 2:	Add_Ack
			Bits 3 to 7:	Sequence number

Data structure of the destination area AINFO with interrupts from PROFINET IO or central I/O devices

Byte	Meaning
0 to 25	Header information, for exact description, see below
26 to 1431	Additional interrupt information: Standardized diagnostic data for the respective interrupt: ARRAY[0] to ARRAY[1405] Note: The additional interrupt information may also be omitted.

Structure of the header information with interrupts from PROFINET IO or central I/O devices

Byte	Data type	Meaning
0 and 1	WORD	• Bits 0 to 7: Block type • Bits 8 to 15: Reserved
2 and 3	WORD	Block length
4 and 5	WORD	Version: • Bits 0 to 7: low byte • Bits 8 to 15: high byte
6 and 7	WORD	ID for interrupt type: • 1: Diagnostics interrupt (incoming) • 2: Process interrupt • 3: Remove module interrupt • 4: Insert module interrupt • 5: Status interrupt • 6: Update interrupt • 7: Redundancy interrupt • 8: Controlled by supervisor • 9: Released by supervisor • 10: Configured module not inserted • 11: Return of the sub-module • 12: Diagnostics interrupt (outgoing) • 13: Slave-to-slave connection alarm • 14: Neighborhood change alarm • 15: Clock synchronization message (bus end) • 16: Clock synchronization alarm (device end) • 17: Network component alarm • 18: Time synchronization alarm (bus end) • 19 to 31: Reserved • 32 to 127: Manufacturer-specific interrupt • 128 to 65535: Reserved
8 to 11	DWORD	API (Application Process Identifier)
12 to 13	WORD	Slot number of the component triggering the interrupt (value range 0 to 65535)
14 to 15	WORD	Submodule slot number of the component triggering the interrupt (value range 0 to 65535)
16 to 19	DWORD	Module identification; specific information on the source of the interrupt

Byte	Data type	Meaning
20 to 23	DWORD	Submodule identification; specific information on the source of the interrupt
24 to 25	WORD	Interrupt specifier: • Bits 0 to 10: Sequence number (value range 0 to 2047) • Bit 11: Channel diagnostics: 0: No channel diagnostics available 1: Channel diagnostic information exists • Bit 12: Status of manufacturer-specific diagnostics: 0: No manufacturer-specific status information available 1: Manufacturer-specific status information available • Bit 13: Status of diagnostics for sub-module: 0: No status information available, all errors have been corrected 1: At least one item of channel diagnostics and/or status information is available • Bit 14: Reserved • Bit 15: Application relationship diagnosis state: – 0: None of the modules configured within this application relationship reports diagnostic information – 1: At least one of the modules configured in this AR is reporting diagnostic information

Structure of the additional interrupt information for interrupts from PROFINET IO or central I/O

The additional interrupt information for PROFINET IO depends on the format identifier. It can comprise multiple data blocks with the same or different format identifier. The following format identifiers are available:

- W#16#0000 to W#16#7FFF: Manufacturer-specific diagnostics

Byte	Data type	Meaning
0 to 1	WORD	Format identifier for the structure of the following data serving as additional interrupt information W#16#0000 to W#16#7FFF: Manufacturer-specific diagnostics
2 to n	BYTE	See manufacturer's manual.

- W#16#8000: Channel diagnostics
  Channel diagnostics is output in blocks of 6 bytes each. The additional interrupt information (without format identifier) is only output for disrupted channels.

Byte	Data type	Meaning	
0 to 1	WORD	Format identifier for the structure of the following data serving as additional interrupt information W#16#8000: Channel diagnostics	
2 to 3	WORD	Channel number of the component triggering the interrupt (value range: 0 to 65535): • W#16#0000 to W#16#7FFF: Channel number of the interface module/ sub-module • W#16#8000: The generic substitute for the entire sub-module • W#16#8001 to W#16#FFFF: Reserved	
4	BYTE	Bits 0 to 2:	Reserved
		Bits 3 to 4:	Type of error: • 0: Reserved • 1: Incoming error • 2: Outgoing error • 3: Outgoing error, other errors present
		Bits 5 to 7:	Type of channel: • 0: Reserved • 1: Input channel • 2: Output channel • 3: Input/output channel

Byte	Data type	Meaning
5	BYTE	Data format: • B#16#00: Free data format • B#16#01: Bit • B#16#02: 2 bits • B#16#03: 4 bits • B#16#04: Byte • B#16#05: Word • B#16#06: Double word • B#16#07: 2 double words • B#16#08 to B#16#FF: Reserved
6 to 7	WORD	Type of error: • W#16#0000: Reserved • W#16#0001: Short circuit • W#16#0002: Undervoltage • W#16#0003: Overvoltage • W#16#0004: Overload • W#16#0005: Overtemperature • W#16#0006: Wire break • W#16#0007: High limit exceeded • W#16#0008: Low limit exceeded • W#16#0009: Error • W#16#000A to W#16#000F: Reserved • W#16#0010 to W#16#001F: Manufacturer-specific • W#16#0020 to W#16#00FF: Reserved • W#16#0100 to w#16#7FFF: Manufacturer-specific • W#16#8000: Device diagnostics available • W#16#8001 to W#16#FFFF: Reserved Not all channels support every error type. For detailed information, refer to the description of the diagnostic data for the specific device.

Note

The section from "channel number" to "type of error" can occur from 0 to n times.

- W#16#8001: MULTIPLE (different types of diagnostic information are transmitted). In this case, the additional interrupt information is transmitted as blocks of variable length.

Byte	Data type	Meaning
0 to 1	WORD	Format identifier for the structure of the following data serving as additional interrupt information W#16#8001: Manufacturer-specific diagnostics and/or channel diagnostics
2 to 3	WORD	Block type
4 to 5	WORD	Block length
6	BYTE	Version: high byte
7	BYTE	Version: low byte
8 to 11	DWORD	API (only if low byte of version = 1)
12 to 13	WORD	Slot number
14 to 15	WORD	Subslot number
16 to 17	WORD	Channel number
18 to 19	WORD	Channel properties
20 to 21	WORD	Format identifier: • W#16#0000 to W#16#7FFF: Manufacturer-specific diagnostics • W#16#8000: Channel diagnostics • W#16#8002: Extended channel diagnostics • W#16#8003: Stepped extended channel diagnostics • W#16#8004 to W#16#80FF: Reserved
22 to n	BYTE	Data depend on the format identifier

Note

The section starting from "block type" can occur from 1 to n times.

- W#16#8002: Extended channel diagnostics

Byte	Meaning
0 to 1	Format identifier W#16#8002
2 to 3	Channel number
4 to 5	Channel properties
6 to 7	Error type
8 to 9	Additional error value
10 to 13	Additional error information

- W#16#8003: Stepped extended channel diagnostics

Byte	Meaning
0 to 1	Format identifier W#16#8003
2 to 3	Channel number
4 to 5	Channel properties
6 to 7	Error type
8 to 9	Additional error value
10 to 13	Additional error information
14 to 17	Qualified channel qualifier

- W#16#8100: Maintenance information

Byte	Meaning
0 to 1	Format identifier W#16#8100
2 to 3	Block type
4 to 5	Block length
6 to 7	Block version
8 to 9	Reserved
10 to 13	Maintenance status

Note

You can find more detailed information about the structure of the additional alarm information in the *Programming Manual SIMATIC PROFINET IO from PROFIBUS DP to PROFINET IO* and the current version of IEC 61158-6-10-1.

Destination area TINFO and AINFO

Destination area TINFO and AINFO

Depending on the respective OB in which "RALRM (Page 4564)" is called, the destination areas TINFO and AINFO are only partially written. Refer to the table below to find out which information is entered respectively.

Interrupt type	OB class	TINFO OB start information	TINFO Management information	AINFO Header information	AINFO Additional interrupt information	
Process interrupt	4x	Yes	Yes	Yes	central:	No
					distributed:	as supplied by PROFIBUS DP slave/ PROFINET IO device
Status interrupt	55	Yes	Yes	Yes	Yes	Yes
Update interrupt	56	Yes	Yes	Yes	Yes	Yes
Manufacturer-specific interrupt	57	Yes	Yes	Yes	Yes	Yes

Interrupt type	OB class	TINFO OB start information	TINFO Management information	AINFO Header information	AINFO Additional interrupt information	
I/O redundancy error	70	Yes	Yes	No	No	No
Diagnostics interrupt	82	Yes	Yes	Yes	central:	Structure according to PROFINET IO standard
					distributed:	as supplied by PROFIBUS DP slave/ PROFINET IO device
Insert/remove interrupt	83	Yes	Yes	Yes	central:	No
					distributed:	as supplied by PROFIBUS DP slave/ PROFINET IO device
Special form of the remove module interrupt: Controlled by supervisor	83	Yes	Yes	Yes	PROFINET IO only	
Special form of the insert module interrupt: Enabled by supervisor	83	Yes	Yes	Yes	PROFINET IO only	
Unconfigured module inserted	83	Yes	Yes	Yes	PROFINET IO only	
Rack failure/ station failure	86	Yes	Yes	No	No	
... all other OBs		Yes	No	No	No	

D_ACT_DP: Enable/disable DP slaves

Description

Use the "D_ACT_DP" instruction to specifically deactivate and reactivate configured DP slaves/ PROFINET IO devices. In addition, you can determine whether each assigned DP slave or PROFINET IO device is currently activated or deactivated.

If you use the instruction to deactivate an IE/PB Link PN IO type of gateway, then all connected PROFIBUS DP slaves will also cease to function. These failures will be reported.

The instruction cannot be used on PROFIBUS PA field devices that are connected by a DP/PA link to a DP master system.

Note

As long as one or more "D_ACT_DP" jobs are active, you cannot load a changed configuration from the programming device to the CPU (within the scope of CiR).
If a changed configuration is being loaded from the programming device to the CPU during ongoing operation (CiR), the CPU will reject activation of a "D_ACT_DP" job.

Several runs through the cycle control point are required to perform the disabling or enabling job. Therefore, you cannot wait for the end of such a job in a programmed loop.

Functional description

"D_ACT_DP" is an instruction that works asynchronously, which means its execution extends over multiple calls. You start the job by calling D_ACT_DP with REQ = 1.

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 3150).

Application

If you configure DP slaves/PROFINET IO devices in a CPU which are not actually present or not currently required, the CPU will nevertheless continue to access these DP slaves/PROFINET IO devices at regular intervals. After the slaves are deactivated, further CPU accessing will stop. With PROFIBUS DP, the fastest possible DP bus cycle can be achieved in this manner and the corresponding error events no longer occur.

Examples

From a machine OEM's point of view, there are numerous device options possible in series production of machines. However, each delivered machine includes only one combination of selected options.

Every one of these possible machine options is configured as a DP slave/PROFINET IO device by the manufacturer in order to create and maintain a common user program having all possible options. Use "D_ACT_DP" to deactivate all DP slaves/PROFINET IO devices not present at machine startup.

A similar situation exists for machine tools having numerous tooling options available but actually using only a few of them at any given time. These tools are implemented as DP slaves/PROFINET IO devices. With "D_ACT_DP", the user program activates the tools currently needed and deactivates those required later.

Identification of a job

If you have started a deactivation or activation job and you call "D_ACT_DP" again before the job is complete, the behavior of the instruction depends on whether or not the new call involves

the same job. If the input parameter LADDR matches, then the call will be interpreted as a follow-on call.

Deactivating DP slaves/PROFINET IO devices

When you deactivate a DP slave/PROFINET IO device with "D_ACT_DP", its process outputs are set to the configured substitute values or to 0 (safe state). The assigned DP master/PROFINET IO controller does not continue to address this component. Deactivated DP slaves/PROFINET IO devices are not identified as faulty or missing by the error LEDs on the DP master/PROFINET IO controller or CPU.

The process image of the inputs of deactivated DP slaves/PROFINET IO devices is updated with 0, that is, it is handled just as it is for failed DP slaves/PROFINET IO devices.

If you are using your program to directly access the user data of a previously deactivated DP slave/PROFINET IO device, the I/O access error OB is called and the corresponding start event is entered in the diagnostics buffer. If you attempt to access a deactivated DP slave/PROFINET IO device via an instruction (such as "RD_REC (Page 4594)"), you will receive the same error information in RET_VAL as for an unavailable DP slave/PROFINET IO device.

Deactivating a DP slave/PROFINET IO device does not start the program error OB, even if its inputs or outputs belong to the system-side process image to be updated. Also there is no entry in the diagnostics buffer.

If a DP station/PNIO station fails after you have deactivated it with "D_ACT_DP", the operating system does not detect the failure.

Applies to PROFIBUS DP: If you wish to deactivate DP slaves functioning as transmitters in slave-to-slave communication, we recommend that you first deactivate the receivers (listeners) that detect which input data the transmitter is transferring to its DP master. Deactivate the transmitter only after you have performed this step.

Activating DP slaves/PROFINET IO devices

When you reactivate a DP slave/PROFINET IO device with "D_ACT_DP", this component is configured and assigned parameters by the associated DP master/PROFINET IO controller (as with the return of a failed DP station/PROFINET IO station). This activation is complete when the component is able to transfer user data.

Activating a DP slave/PROFINET IO device does not start the program error OB, even if its inputs or outputs belong to the system-side process image to be updated. Also there is no entry in the diagnostics buffer.

If you try to activate a DP slave/PROFINET IO device with "D_ACT_DP" that cannot be accessed (for example, because it was physically separated from the bus), the instruction returns the error code W#16#80A7 after expiration of the configured parameter assignment time for distributed I/O. The DP slave/PROFINET IO device is activated and the fact that the activated DP slave/PROFINET IO device cannot be accessed results in a corresponding display in the system diagnostics.

If the DP slave/PROFINET IO device is accessible again afterwards, this results in standard system behavior (for example, a call of the OB configured for this purpose).

Note

Activating a DP slave/PROFINET IO device may be time-consuming. If you wish to cancel a current activation job, start "D_ACT_DP" again with the same value for LADDR and MODE = 2. Repeat the call of "D_ACT_DP" with MODE = 2 until successful cancellation of the activation job is indicated by RET_VAL = 0.

If you wish to activate DP slaves which take part in the slave-to-slave communication, we recommend that you first activate the transmitters and then the receivers (listeners).

Parameters

The following table shows the parameters of the "D_ACT_DP" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Level-triggered control parameter REQ=1: Execute activation or deactivation
MODE	Input	USINT	I, Q, M, D, L or constant	Job identifier Possible values: • 0: Request information on whether the addressed component is activated or deactivated (output via RET_VAL parameter) • 1: Activate the DP slave/PROFINET IO device • 2: Deactivate the DP slave/PROFINET IO device
LADDR	Input	HW_DEVICE	I, Q, M, D, L or constant	Hardware identifier of the DP slave (HW_DPSlave)/PROFINET IO device (HW_Device) The number can be taken from the properties of the DP slave / PROFINET IO device in the Network view or from the "System constants" tab of the standard tag table. If both the identifier for the device diagnostics as well as the ID for operating state transitions are both specified there, you must use the code for the device diagnostics.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
BUSY	Output	BOOL	I, Q, M, D, L	Active code: • BUSY = 1: The job is still active. • BUSY = 0: The job was terminated.

You can find additional information on valid data types in "Overview of the valid data types (Page 2236)".

RET_VAL parameter

Error code* (W#16#...)	Explanation
0000	Job completed without error.
0001	The DP slave/PROFINET IO device is active (this error code is possible only with MODE = 0.)
0002	The DP slave/PROFINET IO device is deactivated (this error code is possible only with MODE = 0.)
7000	First call with REQ=0. The job specified with LADDR is not active; BUSY has the value "0".
7001	First call with REQ=1. The job specified with LADDR was initiated. BUSY has the value "1".
7002	Intermediate call (REQ irrelevant). The activated job is still active; BUSY has the value "1".
8090	• You have not configured a module with the address specified in LADDR. • You operate your CPU as I-Slave and you have specified in LADDR an address of this I-Slave.
8092	The deactivation of the currently addressed DP slave/PROFINET IO device (MODE=2) cannot be canceled by being activated (MODE=1). Activate the component at a later time.
8093	No DP slave / PROFINET IO device that can be activated or deactivated is assigned to the address specified in LADDR.
8094	You have attempted to activate a device which is potential partner for a tool change port. However, another device is already activated on this tool change port at this time. The activated device will remain activated.
80A0	Error during the communication between the CPU and the IO controller.
80A1	Parameters could not be assigned for the addressed component. (This error code is only available when MODE = 1.) Note: "D_ACT_DP" returns this error information only if the activated slaves/devices of this component fails again during parameter assignment. If the parameter assignment of a single module was unsuccessful, "D_ACT_DP" returns the error information W#16#0000.
80A2	The addressed DP slave does not return an acknowledgment. (This error information is not available with PROFINET IO devices. The process is not time-monitored by PROFINET.)
80A3	The DP master/PROFINET IO controller concerned does not support this function.
80A4	The CPU does not support this function for external DP masters/PROFINET IO controller.
80A6	Slot error in the DP slave/PROFINET IO device; not all user data can be accessed (this error code is only available when MODE=1). Note: "D_ACT_DP" returns this error information only if the activated component fails again after parameter assignment and before the end of "D_ACT_DP". If only a single module is unavailable, "D_ACT_DP" returns the error information W#16#0000.
80A7	Activation of a non-accessible device.
80AA	Activation with errors in the DP slave/PROFINET IO device: Differences in the configuration
80AB	Activation with errors in the DP slave/PROFINET IO device: Parameter assignment error
80AC	Activation with errors in the DP slave/PROFINET IO device: Maintenance required
80C1	"D_ACT_DP" has been started and is being continued with another address (this error code is possible when MODE=1 and MODE=2).
80C3	• Temporary resource error: The CPU is currently processing the maximum possible activation and deactivation jobs. (This error code is available only when MODE = 1 and MODE = 2.) • The CPU is busy receiving a modified configuration. Currently you can not enable/disable DP slaves/ PROFINET IO devices.
80C5	DP: Jobs not collected by the user are discarded at restart.
80C6	PROFINET: Jobs not collected by the user are discarded at restart.

Error code* (W#16#...)	Explanation
General error information	See also: GET_ERR_ID: Get error ID locally (Page 3392)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

Others

RD_REC: Read data record from I/O

Description

Use the instruction to read the data record with the number RECNUM from the addressed module. You start the read process by assigning the value "1" to the input parameter REQ during the call. If the read process could be executed immediately, the instruction returns the value "0" at the output parameter BUSY . If BUSY has the value "1", reading is not yet complete.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 3150). The data record read is entered in the target range spanned by the RECORD parameter, providing the data transfer was free of errors.

Note

When you read out a data record with a number greater than one from an FM or a CP you have purchased prior to February 1997 (below referred to as "old modules"), "RD_REC" responds differently than it does in new modules. This special situation is covered in the section "Using old S7-300 FMs and CPs with data record numbers >1" (see below).

If a DPV1 slave is configured via GSD file (GSD rev. 3 and higher) and the DP interface of the DP master is set to "S7 compatible", then you may not read any data records from the I/O modules in the user program with "RD_REC". In this case, the DP master addresses the wrong slot (configured slot + 3).

Remedy: Set the interface of the DP master to "DPV1".

Parameters

The following table shows the parameters of the "RD_REC" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = 1: Read request
LADDR	Input	HW_IO	I, Q, M, D, L or constant	Hardware identifier of the module.
RECNUM	Input	BYTE	I, Q, M, D, L or constant	Data record number (permitted values: 0 to 240)
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code. In addition: the length of the data record actually transferred in bytes (possible values: +1 to +240), if the target range is greater than the transferred data record and if no error occurred during the transfer.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The reading process is not yet complete.
RECORD	Output	ANY	I, Q, M, D, L	Target range for the data record read. With asynchronous execution of "RD_REC", make sure that the actual parameters of RECORD have the same length information for all calls. Only the BYTE data type is permitted. Note: Note that for S7-300 CPUs the parameter RECORD always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and leads to an error message in the user program.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RECORD

Note

If you want to ensure that the entire data record is always read, select a target range with a length of 241 bytes. If the data transfer is without error, RET_VAL contains the actual data record length.

Using old S7-300 FMs and CPs with data record numbers > 1

If you want to use the instruction "RD_REC" to read out a data record with a number greater than one from an old S7-300 FM or old S7-300 CP, remember the following points:

- If the target range is greater than the actual length of the required data record, no data is entered in RECORD . RET_VAL is written with W#16#80B1.
- If the target range is smaller than the actual length of the required data record, the CPU reads as many bytes beginning at the start of the record as are specified in the length information of RECORD and enters this number of bytes inRECORD . RET_VAL has the value "0".
- If the length specified in RECORD is the same as the actual length of the required data record, the CPU reads the data record and enters it in RECORD . RET_VAL is written with "0".

RET_VAL parameter

- If an error has occurred while the function was being executed, the return value contains an error code.
- If no error occurred during the transfer, RET_VAL contains the following:
  - 0, if the entire target range was filled with data from the selected data record (the data record can also be incomplete).
  - The length of the data record actually transferred in bytes (possible values: +1 to +240) if the target range is greater than the transferred data record.

Note

If the general error W#16#8745 occurs, this only indicates that access to at least one byte of the process image was blocked. The data record was read by the module correctly and written to the I/O memory area.

When looking at the "real" error information (error codes W#16#8xyz) in the following table, we distinguish between two cases:

- Temporary errors (error codes W#16#80A2 to 80A3, 80Cx):
  This type of error can possibly be eliminated without user action, meaning it is helpful to call the instruction again (multiple calls, if necessary).
  Example of a temporary error: Resources required are currently in use (W#16#80C3).
- Permanent errors (error codes W#16#809x, 80A0, 80A1, 80Bx):
  This type of error does not correct itself. A new call of the instruction will not be successful until you have eliminated the error. Example of a permanent error: Wrong length specification in RECORD (W#16#80B1).

Note

If you transfer data records to a DPV1 slave with "WR_REC (Page 4599)" or if you read data records from a DPV1 slave with RD_REC and if this DPV1 slave operates in DPV1 mode, the DP master evaluates the error information it received from the slave as follows:

If the error information lies within the range from W#16#8000 to W#16#80FF or W#16#F000 to W#16#FFFF, the DP master passes the error information to the instruction. If the error information is outside this range, the DP master passes the value W#16#80A2 to the instruction and suspends the slave.

For a description of the error information originating from DPV1 slaves, refer to STATUS[3] Parameter STATUS (Page 4566).

Parameter RET_VAL for WR_REC and RD_REC

Error code* (W#16#...)	Explanation	Restriction
0000	No error	-
7000	First call with REQ=0: No data transfer active; BUSY has the value 0.	-
7001	First call with REQ=1: Data transfer started; BUSY has the value 1.	Distributed I/O
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value 1.	Distributed I/O
8090	Address specified at parameter ADDR is invalid.	-
8092	A type other than BYTE is specified in the ANY reference.	-
8093	This instruction is not permitted for the module selected by means of LADDR and IOID .	-
80A0	Negative acknowledgment when reading from the module: Module was removed during the reading process or is defective	With "RD_REC" only
80A1	Negative acknowledgment when writing to the module: Module was removed during the writing process or is defective	With "WR_REC (Page 4599)" only
80A2	• DP protocol error at layer 2 (for example, slave failure or bus problems) • For ET200S, data record cannot be read in DPV0 mode.	Distributed I/O
80A3	DP protocol error with user interface/user	Distributed I/O
80A4	Communication on PBUS+ disrupted	-

Error code* (W#16#...)	Explanation	Restriction
80B0	• Instruction not possible for module type. • The module does not recognize the data record. • Data record number 241 not permitted. • With "WR_REC (Page 4599)", data records 0 and 1 are not permitted.	-
80B1	The length specified in parameter RECORD is incorrect.	• With "WR_REC (Page 4599)": Length incorrect • With "RD_REC" (only when using old S7-300 FMs and S7-300 CPs): specified length > data record length • With DPNRM_DG: specified length < data record length
80B2	The configured slot is not assigned.	-
80B3	Actual module type does not match specified module type	-
80B7	DP slave or module reports an invalid range for a parameter or value.	With "RD_REC" only
80C0	With "WR_REC (Page 4599)": the data can only be written when the CPU is in STOP mode. Note: this means that writing by the user program is not possible. You can only write the data online with PG/PC. With "RD_REC": the module routes the data record, but either no data is present or the data can only be read when the CPU is in STOP mode. Note: if data can only be read when the CPU is in STOP mode, then an evaluation by the user program is not possible. In this case, you can only read the data online with PG/PC. With "DPNRM_DG (Page 4614)": There are no diagnostic data available.	With "WR_REC (Page 4599)" or "RD_REC" or "DPNRM_DG (Page 4614)"
80C1	The data of the previous write job on the module for the same data record have not yet been processed by the module.	-
80C2	The module is currently processing the maximum possible number of jobs for a CPU.	-
80C3	The required resources (memory, etc.) are currently occupied.	-
80C4	Internal temporary error. Job could not be executed. Repeat the job. If this error occurs often, check your installation for sources of electrical interference.	-
80C5	Distributed I/O not available.	Distributed I/O
80C6	Data record transfer stopped due to priority class abort (restart or background)	Distributed I/O
General error information	See also: GET_ERR_ID: Get error ID locally (Page 3392)	-
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

See also

Switching display formats in the program status (Page 2980)

WR_REC: Write data record to I/O

Description

Use the instruction "WR_REC" to transfer the data record RECORD to the addressed module.

The data to be transferred are read from the RECORD parameter during the first call. If the transfer of the data record takes longer than the duration of one call, the contents of the RECORD parameter are no longer relevant for the subsequent instruction calls (for the same job).

You start the writing process by assigning the value "1" to the input parameter REQ during the call. If the writing process could be executed immediately, the instruction returns the value "0" at the output parameter BUSY. If BUSY has the value "1", writing is not yet complete.

Parameters

The following table shows the parameters of the instruction "WR_REC":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, T, C or constant	REQ = 1: Write request
LADDR	Input	HW_IO	I, Q, M, D, L or constant	Hardware identifier of the module.
RECNUM	Input	BYTE	I, Q, M, D, L or constant	Data record number
RECORD	Input	VARIANT	I, Q, M, D, L	Data record
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The writing process is not yet completed.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

See also: RD_REC: Read data record from I/O (Page 4594)

Note

If the general error W#16#8544 occurs, it only indicates that access to at least one byte of the I/O memory area containing the data record was denied. The data transfer was continued.

DPRD_DAT: Read consistent data of a DP standard slave

Description

Use the instruction "DPRD_DAT" to read out consistent data from an I/O module.

The instruction can be used for the central modules as well as for the DP standard slaves and PROFINET IO devices.

You require "DPRD_DAT" because you can only read out a maximum of four continuous bytes using the load commands that access the I/O or the process image input table. If required, you can also read consistent data via the process image of the inputs. Refer to the related documentation to find out if your CPU supports this functionality. For additional information on consistent data of a DP standard slave/PROFINET IO device, refer to Section "Data consistency (Page 5033)".

If necessary, the instruction "DPRD_DAT" can also be used for a data area of 1 byte or larger. For information on the maximum length of the data, refer to the documentation of your CPU (e.g. 64 bytes for an S7-1214).

- Use the parameter LADDR to select the module of the DP normslave/PROFINET IO device. If an access error occurs, the error code W#16#8090 is output.
- Use the RECORD parameter to define the target range of the read data:
  - The target range has to be at least as long as the inputs of the selected module. Only the inputs are transferred, the other bytes are not considered. If you read from a DP standard slave with a modular configuration or with several DP identifiers, you can only access the data of a module of the configured hardware identifier per "DPRD_DAT" call. If you select a target range that is too small, the error code 80B1 is output at the RET_VAL parameter.
  - All bit strings and all integers can be used as data type. It is also possible to use these data types in a data structure of the ARRAY type . The data type STRING is not supported.
- If there was no error during the data transmission, the data that have been read are entered in the target range defined at the RECORD parameter.

Parameters

The following table shows the parameters of the instruction "DPRD_DAT":

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_IO	I, Q, M, L or constant	Hardware ID of the module from which the data is to be read. The hardware ID can be found in the properties of the module in the device view or system constants.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
RECORD	Output	VARIANT	I, Q, M, D, L	Destination area for the user data that were read.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

RET_VAL parameter

Error code* (W#16#...)	Explanation
0000	No error occurred.
8090	• You have not configured a module for the specified hardware identifier or • You have ignored the restriction concerning the length of consistent data, or • you have not specified a hardware identifier as an address at parameter LADDR .
8092	A data type other than (Array of) bit string or integer was specified at the RECORD parameter.
8093	No DP module/PROFINET IO device from which you can read consistent data exists for the hardware identifier specified in LADDR . This error code also occurs if the module addressed by means of LADDR does not have inputs.
80A0	An access error was detected when accessing the I/O.
80B1	The length of the specified target range at parameter RECORD is shorter than the configured user data length.
80C0	The data have not been read yet.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 3392)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Parameter STATUS (Page 4566)

Switching display formats in the program status (Page 2980)

DPWR_DAT: Write consistent data of a DP standard slave

Description

Use the instruction "DPWR_DAT" to transfer the data at the parameters RECORD consistently to the addressed module of the central module or of the DP standard slave/PROFINET IO device and if necessary to the process image (i.e. if the affected address range of the DP standard slaves is projected as consistency range in a process image).

You require "DPWR_DAT" because you can only write a maximum of four continuous bytes using the transfer commands that access the I/O or process image output. If required, you can also write consistent data via the process image outputs. Refer to the related documentation to find out if your CPU supports this functionality. Do not use both possibilities concurrently when writing consistent data: Either use "DPWR_DAT" or write via the process image output table. For additional information on consistent data of a DP standard slave/PROFINET IO

device, refer to Section "Data consistency (Page 5033)". If the DP standard slave has a modular design, you can only access one module of the DP slave.

> **CAUTION**
>
> **I/O access**
>
> When using "DPWR_DAT", avoid accessing I/O areas that have process image partitions with OB6x connections (isochronous mode interrupts) assigned to them.

If necessary, the instruction "DPRD_DAT" can also be used for a data area of 1 byte or larger. For information on the maximum length of the data, refer to the documentation of your CPU (e.g. 64 bytes for an S7-1214).

- Use the LADDR parameter to select the DP standard slave / PROFINET IO device. If an access error occurs on the addressed module, the error code 8090 is output.
- Use the RECORD parameter to define the source range of the data to be written:
  - The source range has to be at least as long as the outputs of the selected module. Only the outputs are transferred, the other bytes are not considered. If the source range at parameter RECORD is longer than the outputs of the configured module, only the data up to the maximum length of the outputs is transferred. If the source range at parameter RECORD is shorter than the outputs of the configured module, the error code 80B1 is output.
  - The following data types can be used: Byte, Char, Word, LWord, DWord, Int, UInt, USInt, SInt, LInt, ULInt, DInt, UDInt. The use of these data types in a data structure of the type ARRAY or STRUCT is also permissible.
  - The data type STRING is not supported.

The data is transferred synchronously, that is, the write process is completed when the instruction is completed.

Parameters

The following table shows the parameters of the instruction "DPWR_DAT":

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_IO	I, Q, M, L or constant	Hardware ID of the module to which the data is to be written. The hardware ID can be found in the properties of the module in the device view or system constants.
RECORD	Input	VARIANT	I, Q, M, D, L	Source area for the user data to be written.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

RET_VAL parameter

Error code* (W#16#...)	Explanation
0000	No error occurred.
8090	• You have not configured a module for the specified hardware identifier or • You have ignored the restriction concerning the length of consistent data, or • you have not specified a hardware identifier at parameter LADDR .
8092	A data type other than (Array of) bit string or integer was specified at the RECORD parameter.
8093	No DP module /PROFINET IO device to which you can write consistent data exists at the HW ID specified in LADDR . This error code also occurs if the DP standard slave / PROFINET IO device addressed via LADDR does not have outputs.
80A1	Access error detected while I/O devices were being accessed.
80B1	The length of the specified source range at the RECORD parameter is shorter than the outputs of the configured DP standard slave / PROFINET IO device.
80C1	The data of the previous write job have not yet been processed by the DP standard slave / PROFINET IO device.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 3392)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Parameter STATUS (Page 4566)

iDevice / iSlave

RCVREC: Receive data record

Description

An I-device can receive a data record from a superordinate controller. The receipt takes place in the user program with the instruction "RCVREC" (receive record).

The instruction has the following operating modes:

- Check whether the I-device has a request for a data record receipt.
- Make the data record available to the output parameters.
- Send an answer to the superordinate controller.

You can determine the operating mode executed by the instruction using the input parameter MODE (see below).

The I-device must be in the RUN or STARTUP mode.

With MLEN, you specify the maximum number of bytes you want to receive. The selected length of the target range RECORD should have at least the length of MLEN bytes.

If a data record was received (MODE=1 or MODE=2), the output parameter NEW will indicate that the data record was stored in RECORD. Note that RECORD has a sufficient length. The output parameter LEN contains the actual length of the data record received in bytes.

Set CODE1 and CODE2 to zero for the positive answer to the superordinate controller. If the received data record is to be rejected, enter the negative answer to the superordinate controller in Error Code 1 of the CODE1 and in Error Code 2 of the CODE2.

Note

If the I-device has received a request for a data record receipt, you must recognize the delivery of this request within a certain duration. After recognition, you must send an answer to the superordinate controller within this time period. Otherwise the I-device will experience a timeout error which causes the operating system of the I-device to send a negative answer to the superordinate controller. For information on the value for the time period, refer to the specifications of your CPU.

The output parameter STATUS receives the error information after the occurrence of an error.

Operating modes

You can determine the operating mode of the instruction "RCVREC" with the input parameter MODE. This step will be explained in the following table.

MODE	Meaning
0	Check whether a request for a data record receipt exists If a data record from a higher-level controller exists on the I-device, the instruction will only write the output parameters NEW, SLOT, SUBSLOT, INDEX and LEN. If you call the instruction several times with MODE=0, then the output parameter will only refer to one and the same request.
1	Receiving a data record for any subslot of the I-device If a data record from a superordinate controller exists on the I-device for any subslot of the I-device, the instruction will write the output parameter and transfer the data record to the parameter RECORD.
2	Receiving a data record for a specific subslot of the I-device If a data record from a superordinate controller exists on the I-device for a specific subslot of the I-device, the instruction will write the output parameter and transfer the data record to the parameter RECORD.
3	Sending a positive answer to the superordinate controller The instruction checks the request of the superordinate controller to receive a data record, accepts the existing data record and sends a positive acknowledgment to the superordinate controller.
4	Sending a negative answer to the superordinate controller The instruction checks the request of the superordinate controller to receive a data record, rejects the existing data record and sends a negative acknowledgment to the superordinate controller. Enter the reason for the rejection in the input parameters CODE1 and CODE2.

Note

After the receipt of a data record (NEW=1) you must call "RCVREC" twice to ensure complete processing. You must do this in the following order:

- First call with MODE=1 or MODE=2
- Second call with MODE=3 or MODE=4

Parameters

The following table shows the parameters of the "RCVREC" instruction:

Parameter	Declaration	Data type	Memory area	Description
MODE	Input	INT	I, Q, M, D, L or constant	Mode
F_ID	Input	HW_SUBMODULE	I, Q, M, D, L or constant	Subslot in the transfer area of the I-device for the data record to be received (only relevant for MODE=2). The high word is always set to zero.
MLEN	Input	INT*	I, Q, M, D, L or constant	Maximum length of the data record to be received in bytes
CODE1	Input	BYTE	I, Q, M, D, L or constant	Zero (for MODE=3) and/or Error Code 1 (forMODE=4)
CODE2	Input	BYTE	I, Q, M, D, L or constant	Zero (for MODE=3) and/or Error Code 2 (forMODE=4)
NEW	Output	BOOL	I, Q, M, D, L	• MODE = 0: New data record was received • MODE=1 or 2: Data record was transferred to RECORD
STATUS	Output	DWORD	I, Q, M, D, L	Error information
SLOT	Output	HW_SUBMODULE	I, Q, M, D, L	Identical to F_ID
SUBSLOT	Output	HW_SUBMODULE	I, Q, M, D, L	Identical to F_ID
INDEX	Output	UINT	I, Q, M, D, L	Number of the data record received
LEN	Output	UINT	I, Q, M, D, L	Length of the data record received
RECORD	InOut	VARIANT	I, Q, M, D, L	Target range for the data record received. Note: Note that for S7-300 CPUs the parameter RECORD always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 Byte 100). The omission of an explicit DB number is not permitted for S7-300 CPUs and results in an error message in the user program.
* Use the data type UINT in the STL programming language.				

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter STATUS

For interpretation of the parameter STATUS refer to section: Parameter STATUS (Page 4566)

PRVREC: Make data record available

Description

An I-device can receive a request from a superordinate controller to make a data record available. The I-device makes the data record available in the user program with the instruction "PRVREC" (provide record).

The instruction has the following operating modes:

- Check whether the I-device has a request for making a data record available.
- Transfer the requested data record to the superordinate controller.
- Sending an answer to the superordinate controller.

You can determine the operating mode executed by the instruction using the input parameter MODE (see below).

The I-device must be in the RUN or STARTUP mode.

Enter the maximum number of bytes the data record to be sent should have with LEN. The selected length of the target range RECORD should have at least the length of LEN bytes.

If a request to make a data record available exists, (MODE=0), the output parameter NEW will be set to TRUE.

If the request for making a data record available is accepted, write RECORD for the positive answer to the superordinate controller with the requested data record and write zero for CODE1 and CODE2. If the request for making a data record available is to be rejected, enter the negative answer to the superordinate controller in Error Code 1 of the CODE1 and in Error Code 2 of theCODE2.

Note

If the I-device has received a request for making a data record available, you must recognize the delivery of this request within a certain time period. After recognition, you must send an answer to the superordinate controller within this time period. Otherwise the I-device will experience a timeout error which causes the operating system of the I-device to send a negative answer to the superordinate controller. For information on the value for the time period, refer to the specifications of your CPU.

The output parameter STATUS receives the error information after the occurrence of an error.

Operating modes

You can determine the operating mode of the instruction "PRVREC" with the input parameter MODE. This step will be explained in the following table.

MODE	Meaning
0	Check whether a request for making a data record available exists If a request from a higher-level controller for making a data record available exists on the I-device, the instruction will only write the output parameters NEW, SLOT, SUBSLOT, INDEX and RLEN. If you call the instruction several times with MODE=0, then the output parameter will only refer to one and the same request.
1	Receiving a request for making a data record available for any subslot of the I-device If such a request from a superordinate controller for any subslot of the I-device exists on the I-device, the instruction will write the output parameter.
2	Receiving a request for making a data record available for a specific subslot of the I-device If such a request from a superordinate controller for a specific subslot of the I-device exists on the I-device, the instruction will write the output parameter.
3	Make the data record available and send a positive answer to the superordinate controller The instruction checks the request of the superordinate controller to make a data record available, makes the request data record available to RECORD and sends a positive acknowledgement to the superordinate controller.
4	Sending a negative answer to the superordinate controller The instruction checks the request of the superordinate controller to make a data record available, rejects this request and sends a negative acknowledgement to the superordinate controller. Enter the reason for the rejection in the input parameters CODE1 and CODE2.

Note

After the receipt of a request (NEW=1) you must call the instruction twice to ensure complete processing. You must do this in the following order:

- First call with MODE=1 or MODE=2
- Second call with MODE=3 or MODE=4

Parameters

The following table shows the parameters of the "PRVREC" instruction:

Parameter	Declaration	Data type	Memory area	Description
MODE	Input	INT	I, Q, M, D, L or constant	Mode
F_ID	Input	HW_SUBMODULE	I, Q, M, D, L or constant	Subslot in the transfer area of the I-device for the data record to be sent (only relevant for MODE=2). The high word is always set to zero.
CODE1	Input	BYTE	I, Q, M, D, L or constant	Zero (for MODE=3) and/or Error Code 1 (forMODE=4)
CODE2	Input	BYTE	I, Q, M, D, L or constant	Zero (for MODE=3) and/or Error Code 2 (forMODE=4)
LEN	Input	UINT	I, Q, M, D, L or constant	Maximum length of the data record to be sent in bytes

Parameter	Declaration	Data type	Memory area	Description
NEW	Output	BOOL	I, Q, M, D, L	The new data record was requested by the superordinate controller.
STATUS	Output	DWORD	I, Q, M, D, L	Error information
SLOT	Output	HW_SUBMODULE	I, Q, M, D, L	Identical to F_ID
SUBSLOT	Output	HW_SUBMODULE	I, Q, M, D, L	Identical to F_ID
INDEX	Output	UINT	I, Q, M, D, L	Number of the data record to be sent
RLEN	Output	UINT	I, Q, M, D, L	Length of the data record to be sent
RECORD	InOut	VARIANT	I, Q, M, D, L	Data record made available Note: Note that for S7-300 CPUs the parameter RECORD always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 Byte 100). The omission of an explicit DB number is not permitted for S7-300 CPUs and results in an error message in the user program.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter STATUS

For interpretation of the parameter STATUS refer to section: Parameter STATUS (Page 4566)

PROFIBUS

DPSYC_FR: Synchronize DP slaves / Freeze inputs

Description

You use the instruction to synchronize one or more groups of DP slaves.

The function involves sending one of the control commands below, or a combination of them, to the relevant groups:

- SYNC (simultaneous output and freezing of output states on the DP slaves)
- UNSYNC (cancels the SYNC control command)
- FREEZE (freeze the input states on the DP slaves and read in the frozen inputs)
- UNFREEZE (cancels the FREEZE control command)

Before you send the control commands listed above, you must assign the DP slaves to groups per configuration. You must know which DP slave is assigned to which group, with which number and know the reactions of the various groups to SYNC/FREEZE.

Note

Note that the control commands SYNC and FREEZE also remain valid when you perform a warm/cold restart.

Please note also that you may initiate only one SYNC/UNSYNC job or only one FREEZE/UNFREEZE job at a time.

Functional description

"DPSYC_FR" works asynchronously, that is, its execution extends over multiple calls. You start the job by calling "DPSYC_FR" with REQ=1.

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 3150).

Identification of a job

If you have triggered a SYNC/FREEZE job and called "DPSYC_FR" again before the first job was completed, the response of the instruction depends on whether the new call is for the same job. If the input parameters LADDR, GROUP and MODE match, the call is interpreted as a follow-on call.

Writing outputs of DP modules

The writing of outputs of DP modules is triggered as follows:

- By transfer commands to the DP I/O
- By writing the process image of the outputs to the modules (by the operating system at the end of OB 1 or by calling the instruction "UPDAT_PO (Page 4552)")
- Calling the "DPWR_DAT (Page 4601)" instruction.

In normal operation, the DP master transfers the output bytes cyclically (within the cycle of the PROFIBUS DP bus) to the outputs of the DP slaves.

If you want to have certain output data (possibly distributed on several slaves) applied to the outputs to the process at exactly the same time, you can send the SYNC control command to the relevant DP master using the "DPSYC_FR" instruction.

What are the effects of SYNC?

With the SYNC control command, the DP slaves of the selected groups are switched to Sync mode. In other words, the DP master transfers the current output data and instructs the DP slaves involved to freeze their outputs. With the following output message frames, the DP

slaves enter the output data in an internal buffer and the state of the outputs remains unchanged.

Following each SYNC control command, the DP slaves of the selected groups apply the output data of their internal buffer to the outputs on the process.

The outputs are only updated cyclically again when you send the UNSYNC control command using the "DPSYC_FR" instruction.

Note

If the DP slaves of the selected group(s) are not currently connected to the network or have failed when the control command was sent, they will not be switched to SYNC mode. This information will not be communicated in the return value of the instruction.

Reading input data of DP modules

The input data of the DP modules are read as follows:

- Using load commands to the DP I/O
- When the process image of the inputs is updated (by the operating system at the start of OB 1 or by calling the "UPDAT_PI (Page 4550)" instruction)
- By calling the "DPRD_DAT (Page 4600)" instruction.

In normal operation, the DP master receives this input data cyclically (within the cycle of the PROFIBUS DP bus) from its DP slaves and makes them available to the CPU.

If you want to have certain input data (possibly distributed on several slaves) to be read from the process at exactly the same time, send the FREEZE control command to the relevant DP master using the "DPSYC_FR" instruction.

What are the effects of FREEZE?

With the FREEZE control command, the DP slaves involved are switched to Freeze mode, in other words the DP master instructs the DP slaves to freeze the current state of the inputs. It then transfers the frozen data to the input area of the CPU.

Following each FREEZE control command, the DP slaves freeze the state of their inputs again.

The DP master only receives the current state of the inputs cyclically again after you have sent the UNFREEZE control command with the "DPSYC_FR" instruction.

Note

If the DP slaves of the selected group(s) are not currently connected to the network or have failed when the control command has been sent, they will not be switched to FREEZE mode. This information will not be communicated in the return value of the instruction.

Data consistency

Because the DPSYC_FR functions are asynchronous and can be interrupted by higher priority classes, you should make sure that the process images are consistent with the actual inputs and outputs of the I/O when using the "DPSYC_FR" instruction.

This is guaranteed if you comply with the following consistency rules:

- Define suitable process image partitions for the "SYNC outputs" and the "FREEZE inputs" (only possible on the S7-400). Call the "UPDAT_PO (Page 4552)" instruction immediately before each first call of a SYNC job. Call the "UPDAT_PI (Page 4550)" instruction immediately after the respective last call of a FREEZE job.
- As an alternative: Use only direct I/O access for outputs involved in a SYNC job and for inputs involved in a FREEZE job. Do not write to these outputs when a SYNC job is active and do not read in these inputs when a FREEZE job is active.

Use of DPWR_DAT and DPRD_DAT

If you use the "DPWR_DAT (Page 4601)" instruction, it must be complete before you send a SYNC job for the outputs involved.

If you use the "DPRD_DAT (Page 4600)" instruction, it must be complete before you send a FREEZE job for the inputs involved.

Startup and "DPSYC_FR"

The user alone must take responsibility for sending the SYNC and FREEZE control commands in the startup OBs.

If you want the outputs of one or more groups to be in SYNC mode when the user program starts, you must initialize these outputs during startup and completely execute the "DPSYC_FR" instruction with the SYNC control command.

If you want the inputs of one or more groups to be in FREEZE mode when the user program starts, you must execute the "DPSYC_FR" instruction with the FREEZE control command completely for these inputs during startup.

Parameters

The following table shows the parameters of the "DPSYC_FR" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Level-triggered control parameter REQ = 1: Triggering of the SYNC/FREEZE job
LADDR	Input	HW_DPMASTER	I, Q, M, D, L or constant	Hardware identifier of the DP master interface The number can be taken from the properties of the DP master interface in the Network view or from the "System constants" tab of the standard tag table.

Parameter	Declaration	Data type	Memory area	Description
GROUP	Input	BYTE	I, Q, M, D, L or constant	Group selection Bit 0 = 1: group 1 selected Bit 1 = 1: group 2 selected : Bit 7 = 1: group 8 selected You can select several groups per job. The value B#16#0 is invalid.
MODE	Input	BYTE	I, Q, M, D, L or constant	Job ID (coding complies with EN 50 170 Volume 2, PROFIBUS) Bit 0: reserved (value 0) Bit 1: reserved (value 0) Bit 2: • = 1: UNFREEZE will be executed • = 0: no meaning Bit 3: • = 1: FREEZE will be executed • = 0: no meaning Bit 4: • = 1: UNSYNC will be executed • = 0: no meaning Bit 5: • = 1: SYNC will be executed • = 0: no meaning Bit 6: reserved (value 0) Bit 7: reserved (value 0) Possible values: • with exactly one ID per job: – B#16#04 (UNFREEZE) – B#16#08 (FREEZE) – B#16#10 (UNSYNC) – B#16#20 (SYNC) • with more than one ID per job: – B#16#14 (UNSYNC, UNFREEZE) – B#16#18 (UNSYNC, FREEZE) – B#16#24 (SYNC, UNFREEZE) – B#16#28 (SYNC, FREEZE)
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code. Make sure that you evaluate RET_VAL each time the block has been executed.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY=1: The SYNC/FREEZE job is not yet complete.

You can find additional information on valid data types in "Overview of the valid data types (Page 2236)".

RET_VAL parameter

Note

If you access DPV1 slaves, error information of these slaves can be forwarded from the DP master to the instruction. For a description of this error information, refer to STATUS[3], STATUS (Page 4566) parameter.

Error code* (W#16#...)	Explanation
0000	Job completed without error.
7000	First call with REQ=0. The job specified with LADDR, GROUP and MODE is not active; BUSY has the value 0.
7001	First call with REQ=1. The job specified with LADDR, GROUP and MODE was triggered; BUSY has the value 1.
7002	Intermediate call (REQ irrelevant). The activated SYNC/FREEZE job is still running; BUSY has the value 1.
8090	The module selected with LADDR is not a DP master.
8093	This instruction is not permitted for the module selected with LADDR (configuration or version of the DP master).
8094	GROUP parameter is incorrect
8095	MODE parameter is incorrect
80B0	The group selected with GROUP is not configured.
80B1	The group selected with GROUP is not assigned to this CPU.
80B2	The SYNC job specified with MODE is not permitted on the group selected with GROUP.
80B3	The FREEZE job specified with MODE is not permitted on the group selected with GROUP .
80C2	Temporary shortage of resources on the DP master: The DP master is currently processing the maximum number of jobs for a CPU.
80C3	This SYNC/UNSYNC job cannot be activated at present because only one SYNC/UNSYNC job can be triggered at a time. Check your user program.
80C4	This FREEZE/UNFREEZE job cannot be activated at present because only one FREEZE/UNFREEZE job can be triggered at a time. Check your user program.
80C5	Short circuit directly at DP interface
80C6	Job aborted due to I/O disconnection by CPU
80C7	Job aborted due to warm or cold restart on the DP master
General error information	See also: GET_ERR_ID: Get error ID locally (Page 3392)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

DPNRM_DG: Read diagnostics data from a DP slave

Description

You use the "DPNRM_DG" instruction to read the current diagnostic data of a DP slave in the form specified in EN 50 170 Volume 2, PROFIBUS.

Refer to the following table for the basic structure of the slave diagnostic data and to the manuals of the DP slaves for further information.

Byte	Meaning
0	Station status 1
1	Station status 2
2	Station status 3
3	Master station number
4	Vendor ID (high byte)
5	Vendor ID (low byte)
6 ...	Additional slave-specific diagnostic information

The data that has been read is entered in the target range indicated by RECORD following without error data transfer. You start the read process by assigning the value "1" to the REQ input parameter when the "DPNRM_DG" instruction is called.

Functional description

The reading process is executed asynchronously, in other words, it can extend over several calls. The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 3150).

Parameters

The following table shows the parameters of the "DPNRM_DG" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = 1: Read request
LADDR	Input	HW_DPSLAVE	I, Q, M, D, L or constant	Configured diagnostic address of the DP slave Note: Address must be specified in hexadecimal form, for example, diagnostic address 1022 means: LADDR:=W#16#3FE.

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Return	DINT, INT, LREAL, REAL	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code. If no error has occurred, the length of the data actually transferred is entered in RET_VAL .
RECORD	Output	VARIANT	I, Q, M, D, L	Destination area for the diagnostic data that were read. Only the BYTE data type is permitted. The minimum length of the data record to be read or the target range is 6. The maximum length of the data record to be sent is 240. Standard slaves can provide more than 240 bytes of diagnostic data up to a maximum of 244 bytes. In this case, the first 240 bytes are transferred to the target range and the overflow bit is set in the data.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The reading process is not yet complete.

You can find additional information on valid data types in "Overview of the valid data types (Page 2236)".

You can find information on data type conversion in the different programming languages under "Overview of data type conversion (Page 2417)".

RECORD parameter

The CPU evaluates the actual length of the read diagnostic data:

If the length specified for RECORD

- is less than the number of data bytes supplied, the data is discarded and a corresponding error code is entered in RET_VAL.
- is greater than or equal to the number of data bytes supplied, the data is accepted in the destination area and the actual length is entered in RET_VAL as a positive value.

Note

You must ensure that the actual parameters of RECORD correspond in all calls belonging to a job.

A job is uniquely identified by the LADDR input parameter.

Standard slaves with more than 240 bytes of diagnostic data

With standard slaves on which the number of standard diagnostic data is between 241 and 244 bytes, note the following points:

If the length specified for RECORD

- is less than 240 bytes, the data is discarded and a corresponding error code is entered in RET_VAL.
- If the length specified for RECORD is greater than or equal to 240 bytes, the first 240 bytes of the standard diagnostic data is transferred to the target range and the overflow bit is set in the data.

RET_VAL parameter

- If an error occurs while the function is being executed, the return value contains an error code.
- If no errors have occurred during data transfer, RET_VAL contains the length of the data read in bytes as a positive number.

Note

The amount of data read in a DP slave depends on its diagnostics status.

For the evaluation of the error information of the RET_VAL parameter, refer to the following table.

Error code (W#16#...)	Explanation	Restriction
7000	First call with REQ = 0: No data transfer active; BUSY has the value "0".	-
7001	First call with REQ = 1: Data transfer triggered; BUSY has the value "1".	Distributed I/O
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".	Distributed I/O
8090	Address specified at parameter LADDR is invalid.	-
8093	This instruction is not permitted for the module selected by means of LADDR and IOID.	-
80A2	• DP protocol error at layer 2 (for example, slave failure or bus problems) • For ET200S, data record cannot be read in DPV0 mode.	Distributed I/O
80A3	DP protocol error with user interface/user	Distributed I/O
80A4	Communication on PBUS+ disrupted	Distributed I/O
80B0	• Instruction not possible for module type. • The module does not recognize the data record. • Data record number 241 not permitted. • With "WR_REC (Page 4559)", data records 0 and 1 are not permitted.	-
80B1	The length specified in parameter RECORD is incorrect.	specified length < data record length
80B2	The configured slot is not assigned.	-
80B3	Actual module type does not match specified module type	-
80C0	There are no diagnostic data available.	-
80C1	The data of the previous write job on the module for the same data record have not yet been processed by the module.	-
80C2	The module is currently processing the maximum possible number of jobs for a CPU.	-
80C3	The required resources (memory, etc.) are currently occupied.	-

Error code (W#16#...)	Explanation	Restriction
80C4	Internal temporary error. Job could not be executed. Repeat the job. If this error occurs often, check your installation for sources of electrical interference.	-
80C5	Distributed I/O not available.	Distributed I/O
80C6	Data record transfer stopped due to priority class abort (restart or background)	Distributed I/O
General error information	See also: GET_ERR_ID: Get error ID locally (Page 3392)	-
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

See also

Switching display formats in the program status (Page 2980)

Overview of data type conversion (Page 2417)

DP_TOPOL: Determine topology for DP master system

Description

You use the instruction to trigger the topology determination for a selected DP master system. Calling the instruction will address all diagnostics repeaters on a DP master system.

Note

The topology can only be determined for one DP master system at a time.

Topology determination is the prerequisite for the detailed display of the error location if line errors occur. After configuration and after every change to the physical configuration of a DP master system, you must repeat the topology determination with "DP_TOPOL".

Changes to the physical configuration are:

- Changes in line lengths
- Adding or removing stations or components with repeater function
- Changing station addresses

If an error is reported by a diagnostics repeater, "DP_TOPOL" writes the DPR and DPRI outputs for the duration of one "DP_TOPOL pass. If errors are reported by multiple diagnostics repeaters of the selected DP master system, "DP_TOPOL" writes DPR and DPRI with information regarding the first diagnostics repeater reporting the error. You can read out the entire diagnostic information on the programming device or using the "DPNRM_DG (Page 4614)" instruction. If no diagnostics repeater reports an error, the DPR and DPRI outputs have the value NULL.

If you want to repeat a topology determination after an error occurs, you must first reset "DP_TOPOL". This is done by calling "DP_TOPOL" with REQ=0 and R=1.

Functional description

"DP_TOPOL" works asynchronously, that is, its execution extends over multiple calls. You start determining the bus topology by calling "DP_TOPOL" with REQ=1 . If you want to cancel the process, call "DP_TOPOL" with R=1 .

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 3150).

Note

Determining the topology may take several minutes.

Identification of a job

The input parameter DP_ID uniquely specifies a job.

If you have called "DP_TOPOL" and you call this instruction again before the topology is re-determined, the manner in which the instruction reacts depends on whether the new call involves the same job: If the DP_ID parameter matches a job that has not yet been completed, then the call is interpreted as a follow-up call and the value W#16#7002 will be entered in RET_VAL. On the other hand, if another job is involved, the CPU rejects it.

Parameters

The following table shows the parameters of the "DP_TOPOL" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ=1: Trigger topology determination
R	Input	BOOL	I, Q, M, D, L or constant	R=1: Cancel topology determination
DP_ID	Input	HW_IOSYSTEM	I, Q, M, D, L or constant	DP master system ID of those master systems for which the topology will be determined
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY=1: Topology determination is not yet complete.
DPR	Output	BYTE	I, Q, M, D, L	PROFIBUS address of the diagnostics repeater reporting the error.
DPRI	Output	BYTE	I, Q, M, D, L	Measuring segment diagnostics repeater reporting the error: • Bit 0 = 1: Temporary faults on segment DP2 • Bit 1 = 1: Permanent faults on segment DP2 • Bit 4 = 1: Temporary faults on segment DP3 • Bit 5 = 1: Permanent faults on segment DP3

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

When looking at the "real" error information (error codes W#16#8xyz) in the following table, we distinguish between two cases:

- Temporary errors (error codes W#16#80A2 to 80A4, 80C3, 80C5):
  It is possible to eliminate this type of error without user action; in other words, it is advisable to call "DP_TOPOL" again (multiple calls, if necessary).
  Example of a temporary error: Resources required are currently in use (W#16#80C3).
- Permanent errors (error codes W#16#8082, 80B0, 80B2):
  This type of error does not correct itself. A new call of "DP_TOPOL" is only advisable after you have eliminated the error. Example of a permanent error: The DP master/CPU does not support this service. (W#16#80B0).

Error code* (W#16#...)	Explanation
0000	Job completed without error.
7000	First call with REQ=0. Topology determination is not triggered. BUSY has the value "0".
7001	First call with REQ=1. The request to execute topology determination was sent. BUSY has the value "1".
7002	Intermediate call (REQ irrelevant): Topology determination is not yet complete. BUSY has the value "1".
7010	You have attempted to cancel topology identification. But there is no running job with the specified DP_ID. BUSY has the value "0".
7011	First call with R=1. The cancellation of the topology determination was triggered. BUSY has the value "1".
7012	Intermediate call: The cancellation of the topology determination is not yet complete. BUSY has the value "1".
7013	Final call: The topology determination was cancelled. BUSY has the value "0".
8082	No DP master system is configured with the specified DP_ID.
80A2	Error during topology determination; for more detailed information, refer to output parameters DPR and DPRI.
80A3	Error during topology determination: Monitoring time has elapsed (timeout).
80A4	Communication on PBUS+ disrupted
80B0	The DP master/CPU does not support this service.
80B2	Error during topology determination: No diagnostics repeater was found at the selected DP master system.
80C3	Resources required are currently in use. Possible cause: You have initiated a second topology determination (only one topology determination is permitted at one time).
80C5	The DP master system is currently not available.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 3392)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

ASi

ASI_CTRL: Controlling ASi master behavior

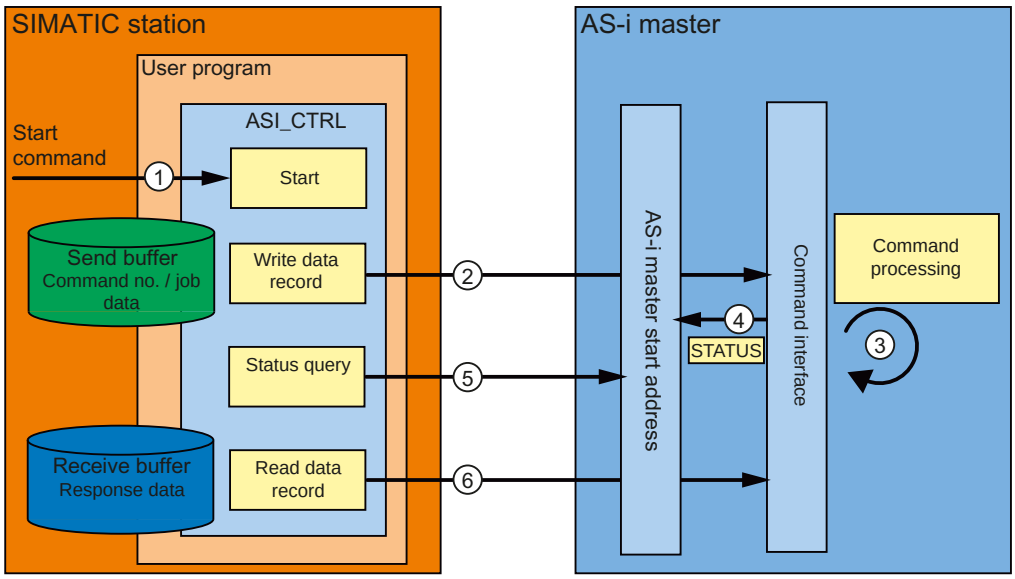
Description of ASI_CTRL

Description

Using the instruction "ASI_CTRL", you can control the behavior of the AS-i master from the user program of the PLC. The instruction processes the command protocol automatically. It also enables parameter assignment on SIMATIC AS-i masters and reading out information data. The functions and operation of the command interface are detailed in the manual for the AS-i master.

Both centrally inserted AS-i masters and distributed AS-i masters via PROFIBUS DP are supported. Combinations with PROFINET IO (e.g. IE/PB Link PN IO) are also possible.

The schematic diagram below shows the functions of the "ASI_CTRL" instruction:



① Start of processing at the REQ parameter.

② The program sends the required command to the AS-i master via the instruction "RDREC".

③ The AS-i master processes the command.

④ The current status of the AS-i master is stored in the input area of the binary data (logical start address).

⑤ The instruction "ASI_CTRL" cyclically queries and evaluates the 4 status bits.

⑥ Once command processing is complete, the command job is completed with "RDREC". Depending on the command, the data field of "RDREC" may contain the response data for the command or additional status information.

Differences in the command call between IE/ AS-i Link and DP/AS-i Links

There are significant differences in how a controller exchanges commands with an AS-i master.

- **IE/AS-i Link** (PROFINET) used the data record interface. The different commands are called with either "Write data record" ("WRREC" instruction) or "Read data record" ("RDREC" instruction) by various data record numbers.
- **DP/AS-i Links** (PROFIBUS) use the command interface. All commands are called by data record number 2 with "Write data record" ("WRREC" instruction) plus "Read data record" ("RDREC" instruction). The type of command is defined by the data content in the write job.

Changes compared to the instruction "ASi_3422".

The instruction "ASI_CTRL" is a revised version of the instruction "ASi_3422 (Page 6529)" (S7-300/400) and provides improved functionality and compatibility. The specific changes are as follows:

- For writing and reading diagnostic data records, the instructions "WR_REC (Page 4599)" and "RD_REC (Page 4594)" have been replaced by the instructions "RDREC (Page 4557)" and "WRREC (Page 4559)". Their function is identical; however, they support data transfer via PROFINET IO.
- The block type of the instruction has been changed from a function (FC) to a function block (FB). "ASI_CTRL" has an instance data block and is multi-instance-capable.
- The designation of the formal parameters of "ASI_CTRL" complies with the SIMATIC system blocks. There is no STARTUP input parameter. The definition of the STATUS parameter is based on the instructions "RDREC (Page 4557)" and "WRREC (Page 4559)". The status identifiers for the parameter DONE and the new parameter BUSY have also been adjusted.

Operation of the "ASI_CTRL" instruction

The instruction "ASI_CTRL" is an asynchronous function block; in other words, its processing extends over multiple calls.

- A job is started with REQ = TRUE.
- The job status is displayed via the BUSY output parameters and the two central bytes of the output parameter STATUS.
- The BUSY parameter is set during job processing. Upon the first call, STATUS is assigned the value 00700100_H. Upon all subsequent calls for this job, it is assigned the value 00700200_H. When the job is complete, the result is output at the parameters DONE or ERROR.
  - If no errors have occurred, DONE is set. For jobs with response data from the AS-i master, this data is provided in the specified receive buffer. In such cases, the STATUS parameter also displays the volume of data supplied in bytes. For jobs without response data, the value 00000000_H is entered in STATUS.
  - If an error occurs during job processing, ERROR is set. The content of the receive buffer is invalid in such a case. An error code is entered in the STATUS parameter for a more detailed description of the error which has occurred.

Number of command calls

If you use the instruction "ASI_CTRL" to send commands, you may not simultaneously send other commands to the same AS-i master with "RDREC (Page 4557)" or "WRREC (Page 4559)". This also applies to multiple calls of the instruction for the same AS-i master.

The instruction "ASI_CTRL" cannot be run with interruptions. Calls therefore cannot be programmed in program priority classes which interrupt each other (for example with calls in OB 1 and in OB 35).

Parameter

The following table shows the parameters of the instruction "ASI_CTRL":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, constant	REQ = TRUE starts a new job unless a job is already in progress. No edge evaluation takes place.
LADDR	Input	WORD	I, Q, M, D, L, constant	Start address of the AS-i master in the S7 address space (logical base address). The start address is defined in the hardware configuration when the master is configured.
SD	Input	VARIANT	I, Q, M, D, L	Send buffer The parameter refers to a memory area in which the command is to be specified (see " ASi commands (Page 4624)"). Example: P#DB101.DBX 0.0 BYTE 223

Parameter	Declaration	Data type	Memory area	Description
RD	Input	VARIANT	I, Q, M, D, L	Receive buffer This buffer is only relevant for commands which deliver answer data. The parameter refers to a memory area in which a command response is stored (see "ASi commands (Page 4624)"). Example: P#DB102.DBX 224.0 BYTE 221
DONE	Output	BOOL	Q, M, D, L	DONE = TRUE: Job completed without errors.
BUSY	Output	BOOL	Q, M, D, L	BUSY = TRUE: Job in progress.
ERROR	Output	BOOL	Q, M, D, L	ERROR = TRUE: Job aborted with error.
STATUS	Output	DWORD	M, D	Job status / error code See the "STATUS parameter" description.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Note

Parameters LADDR, SD, and RD

The parameters LADDR, SD, and RD must not be changed in any block cycle when a job is in progress: they must remain constant.

STATUS parameter

The following table shows the possible STATUS displays dependent upon DONE and ERROR.

DONE	ERROR	STATUS	Meaning
0	0	00700000_H	First call with REQ = FALSE; no active job.
0	0	00700100_H	First call with REQ = TRUE; job has started.
0	0	00700200_H	Subsequent call (REQ irrelevant); job is still in progress.
1	0	00000000_H	Job completed without errors. No response data.
1	0	$0000xx00_H$	Job completed without errors. Number of xx bytes of response data.
0	1	$C0818400_H$	Data type of formal operand RD invalid.
0	1	$C0818500_H$	Error in communication with AS-i master (incorrect address configured at the LADDR parameter).
0	1	$C0838100_H$	Incorrect AS-i slave address.
0	1	$C0838200_H$	AS-i slave is not enabled (not in LAS).
0	1	$C0838300_H$	Error at the AS interface (the SD parameter may have been set too small).
0	1	$C0838400_H$	The command is not valid with the current AS-i master status.
0	1	$C0838500_H$	There is no AS-i slave with the address "0".
0	1	$C0838600_H$	The AS-i slave has invalid configuration data (I/O or ID codes).
0	1	$C083A100_H$	The AS-i slave addressed has not been found at the AS interface.
0	1	$C083A200_H$	There is no AS-i slave with the address "0".

DONE	ERROR	STATUS	Meaning
0	1	$C083A300_H$	There is already an AS-i slave with the new address at the AS interface.
0	1	$C083A400_H$	The AS-i slave address cannot be deleted.
0	1	$C083A500_H$	The AS-i slave address cannot be set.
0	1	$C083A600_H$	The AS-i slave address cannot be permanently saved.
0	1	$C083A700_H$	Error during reading of the extended ID1 code.
0	1	$C083A800_H$	The target address is not plausible (e.g. a B slave address has been used for a standard slave).
0	1	$C083B100_H$	A length error has occurred during string transfer.
0	1	$C083B200_H$	A protocol error has occurred during string transfer.
0	1	$C083F800_H$	Unknown job number or job parameter.
0	1	$C083F900_H$	The AS-i master has detected an EEPROM error.

See also

Switching display formats in the program status (Page 2980)

ASi commands

Description

The command interface allows the controller and AS-i master to exchange parameter assignment and information data.

These commands

- provide the complete functionality of the M4 master profile of the AS-i master specifications.
- enable the AS-i master to be completely configured from the controller.

Note

AS-i commands supported

Please see the manual of the relevant AS-i master for the AS-i commands supported and a detailed description.

General structure of the send buffer

The general structure of the send buffer for commands and job data is set out in the table below. The area for the command number must always be filled. The number of bytes for the

job data can be found in the description of the command (see AS-i master documentation). "q" is the start address of the send buffer.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	Meaning							
q + 0	Command number							
q + 1	Job data							
q + 2	Job data							
q + ...	Job data							

General structure of the receive buffer

The general structure of the receive buffer for the command response data is set out in the table below. The number of bytes for the response data depends on the command. Some commands do not return response data, and therefore only require the definition of a virtual receive buffer which is not filled with data. "n" is the start address of the receive buffer.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	Meaning							
n + 0	Command number (echo)							
n + 1	Response data							
n + 2	Response data							
n + ...	Response data							

CAUTION

Memory areas can be overwritten

If the receive buffer of the "ASI_CTRL" instruction is too short, neighboring memory areas may be overwritten. The length specified in the ANY pointer of the RD parameter when the instruction "ASI_CTRL" is called is irrelevant. The required length for the receive buffer can be found in the description of the command.

The following applies for command numbers 39_H, 41_H, 42_H, 43_H and 44_H:

The receive buffer must be 221 bytes long (bytes 0 to 220), even if the command returns less data. Depending on the command, the highest bytes in the receive buffer may be overwritten with zero values by the AS-i master.

AS-i commands

A selection of possible AS-i commands is set out in the table below.

Name	Parameter	Return	Coding
Set_permanent_parameter (Set_Permanent_Parameter)	Slave address, parameter		00_H
Get_permanent_parameter (Get_Permanent_Parameter)	Slave address	Parameter	01_H
Write_parameter (Write_Parameter)	Slave address, parameter	Parameter echo	02_H
Read_parameter (Read_Parameter)	Slave address	Parameter value	03_H
Store_actual_parameters (Store_Actual_Parameters)			04_H
Set_configuration_data	Slave address, configuration		25_H
Get_configuration_data	Slave address	set configuration data	26_H
Store_actual_configuration (Store_Actual_Configuration)			07_H
Get_actual_configuration	Slave address	Actual configuration data	28_H
Configure_LPS	LPS		29_H
Set_offline_mode	Mode		$0A_H$
Select_auto-program	Mode		$0B_H$
Set_mode	Mode		$0C_H$
Change_AS–iSlave_address (Change_AS–iSlave_Address)	Address1, Address2		$0D_H$
Get_AS-iSlave_status	Slave address	Error record of the AS-i slave	$0F_H$
Read_lists_and_flags		LDS, LAS, LPS, flags	30_H
Get_overall_configuration		Actual configuration data, current parameters, LAS, flags	39_H
Set_overall_configuration	Overall configuration		$3A_H$
Write_parameter_list	Parameter list		$3C_H$
Read_parameter_echo_list		Parameter echo list	33_H
Write_CTT2_request	Slave address CTT2 string	CTT2 string	44_H
Read_version_identifier		Version string	14_H
Read_AS-i_slave	Slave address	ID code	17_H
Read_AS-i_slave_extended_ID1	Slave address	Extended ID1 code	37_H
Write_AS-iSlave_extended_ID1	Extended ID1 code		$3F_H$
Read_AS-iSlave_extended_ID2	Slave address	Extended ID2 code	38_H
Read_AS-iSlave_IO	Slave address	I/O configuration	18_H
Read_I/O_error_list		LPF	$3E_H$

Name	Parameter	Return	Coding
Write_AS–i-slave_parameter_string	Slave address, parameter string		40 $_H$
Read_AS–iSlave_parameter_string	Slave address	Parameter string	41 $_H$
Read_AS-iSlave_ID_string	Slave address	ID string	42 $_H$
Read_AS-iSlave_diagnostic_string	Slave address	Diagnostic string	43 $_H$
Read_AS-i_line_error_counter			4A $_H$
Read_and_clear_AS-i_line_error_counter			4B $_H$
Read_AS-iSlave_error_counter	Slave address		4C $_H$
Read_and_clear_AS-iSlave_error_counter	Slave address		4D $_H$
Additional command for DP/ AS-i F-Link:			
AS–i_status/diag_of_F_slaves		Status/diagnostics of all AS-i safe slaves	51 $_H$

Note

Re-initializing the AS-i master command interface

Another command which is not mentioned in the table is command 77 $_H$. This call re-initializes the command interface of the AS-i master. Any command which the AS-i master specified is currently processing will be terminated.

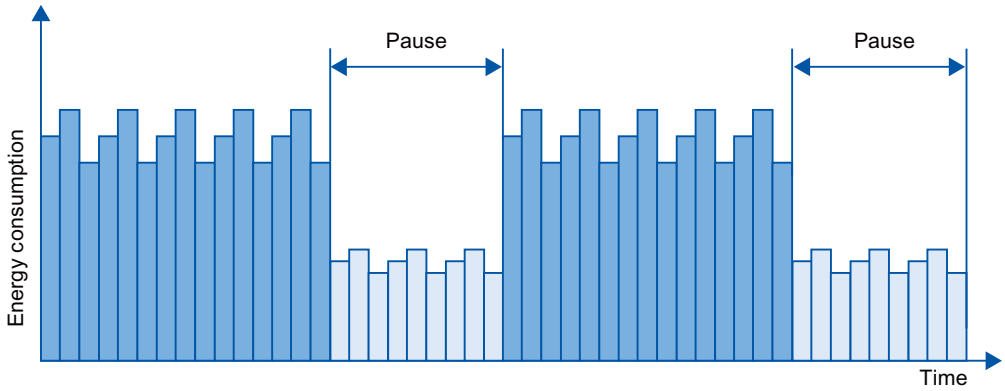
As of version V2.1.20 of DP/AS-i LINK Advanced, command 0E $_H$ is also available. This call enables you to release and block the ground fault monitoring function for a line.

PROFIenergy

Description of PROFIenergy

PROFIenergy

PROFIenergy is a manufacturer- and device-neutral profile for energy management with PROFINET. PROFIenergy enables central, coordinated device shutdown to reduce electricity consumption during breaks in production and unplanned interruptions.



The PROFINET devices/power modules are switched off via special commands in the user program of the PROFINET IO controller. No additional hardware is required. The PROFINET devices interpret the PROFIenergy commands directly.

PROFIenergy controller (PE controller)

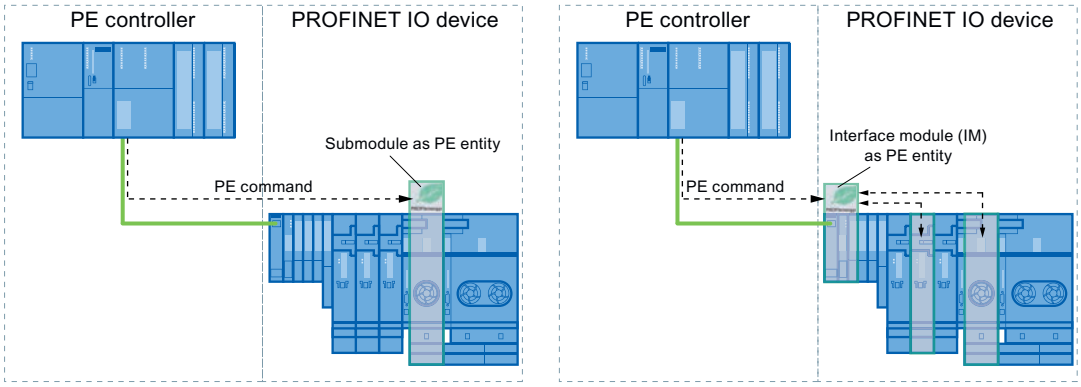
The PE controller is a PLC which enables/disables the idle state in lower-level devices. Individual production components and entire production lines are switched off and reactivated from the user program. Commands (e.g. "Start_Pause" and "End_Pause") are sent to the lower-level device using corresponding instructions (function blocks). The commands are sent via the PROFINET communication protocol.

PROFIenergy entity (PE entity)

The PE entity receives the PROFIenergy commands of the PE controller and executes these accordingly (for example, by returning a measured value or activating an energy saving mode). Implementation of the PE entity in a PROFIenergy-capable device is device- and manufacturer-specific.

The PE entity can, for example, be executed:

- Within the proxy of a submodule: The PE commands are valid for the addressed submodule and any existing lower-level submodules.
- Within the proxy of a module: The PE commands are valid for different submodules within the module.



- For networked submodules without a proxy function: The PE commands in this case are only valid for the respective submodule.

PROFIenergy instructions

- Instructions for IO controller
  - The "PE_START_END (Page 4630)" instruction represents the simplest way to activate or deactivate the idle state of the PROFINET devices (PROFIenergy commands "Start_Pause" and "End_Pause"). This is done using a positive and negative signal edge in the instruction.
  - The "PE_CMD (Page 4634)" instruction allows you to transmit all PROFIenergy commands, including "Start_Pause" and "End_Pause". With other commands, you can query the current status of the PROFINET devices or the behavior during breaks, for example.
  - The instruction "PE_DS3_Write_ET200S (Page 4640)" is used to define the switching characteristic of up to 8 slots of ET 200S. The instruction is not a PROFIenergy instruction; however, it supplements the PROFIenergy functions for an ET 200S.
- Instruction for iDevices
  The "PE_I_DEV (Page 4664)" instruction allows you to implement PROFIenergy on iDevices as well. The instruction receives PROFIenergy commands on the iDevice and forwards these to the user program for processing. After processing the command, the user program calls the "PE_I_DEV (Page 4664)" instruction again in order to send the acknowledgement to the IO controller. For these replies, each command offers you a corresponding auxiliary block that supplies the response data to the instruction "PE_I_DEV (Page 4664)".

PROFIenergy commands (PE commands)

The PE controller transmits PE commands to the PE entity. The PE command can be either a control command to switch a PE entity to the energy-saving mode, or a command to read a status or measured value:

- PI commands for control
  PROFIenergy supports two control commands that can be executed using either the "PE_Start_End (Page 4630)" instruction or the "PE_CMD (Page 4634)" instruction:
  - Start_Pause: Starting a suitable energy-saving mode (PE Energy-saving mode)
  - End_Pause: Exiting the energy-saving mode (switch to PE_ready_to_operate mode)
- PI commands for reading a status or measured value
  Certain condition information can be read by the control system using the following status commands with the instruction "PE_CMD (Page 4634)":
  - PE_Identify: Read out which PE commands are supported by the PE entity.
  - PEM_Status: Read out the currently active mode of a PE entity (e. g. PE_ready_to_operate).
  - Query_Modes: Output an overview of all supported energy-saving modes, including the time and energy information
  - Query_Measurement: Output the measured values of a PE entity

Example applications

Examples for use of PROFIenergy instructions can be found in the Industry Online Support in the entry "PROFIenergy - Saving Energy with SIMATIC S7 (http://support.automation.siemens.com/WW/view/en/41986454)".

See also

Service and Support (http://support.automation.siemens.com/)

IO controller

PE_START_END: Start and exit energy-saving mode

Description

The instruction "PE_START_END" is used to start and exit the energy-saving mode of a specified PE entity (e.g. ET 200S).

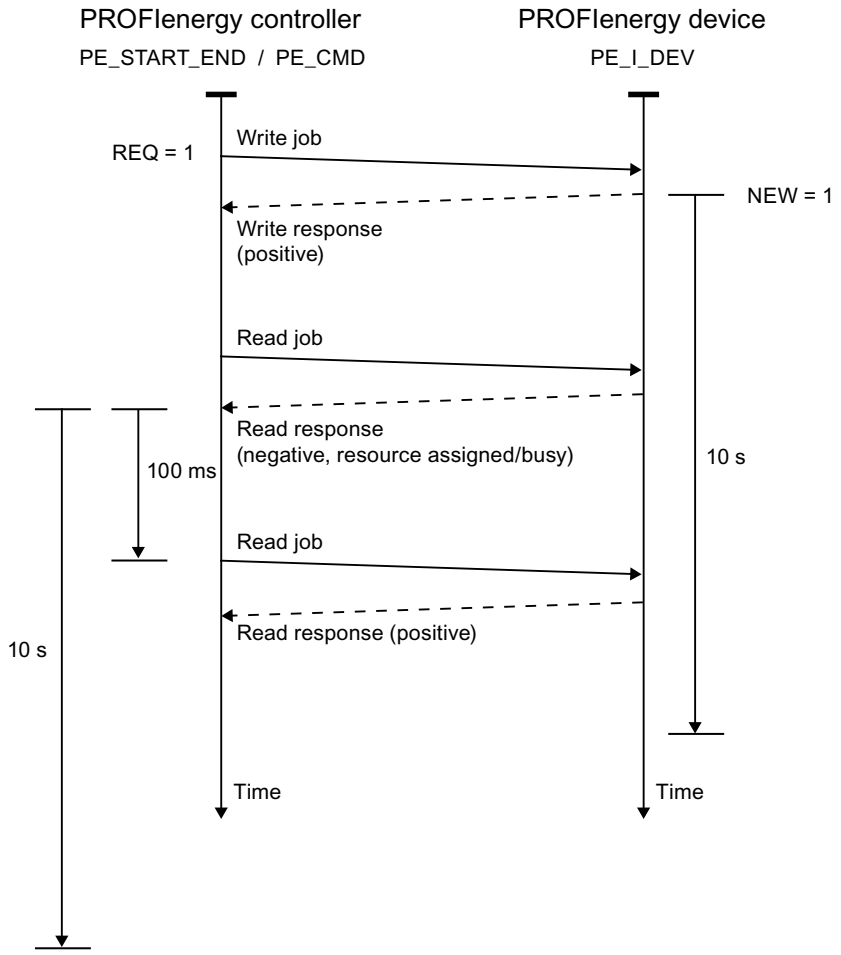
The instruction "PE_START_END" is used in the PE controller, preferably when only field devices from which energy data does not need to be read out are connected to the corresponding PE devices. Alternatively, the instruction "PE_CMD (Page 4634)" can be used to read energy data.

The energy-saving modes are configured in the user program of the PE controller. On completion of the "PE_START_END" instruction, the PE entity reports its current energy-saving mode and outputs this data at parameter PE_MODE_ID.

Write and read jobs of the "PE_START_END" instruction.

The instruction "PE_START_END" sends a PROFIenergy command internally as a write job to the PE entity using "WRREC (Page 4559)". "PE_START_END" then waits for acknowledgment from the PE entity. The acknowledgment data record is read with the instruction "RDREC (Page 4557)" every 100 milliseconds. Until acknowledgment is received from the PE entity, the function repeats the read job for 10 seconds at intervals of 100 milliseconds. The response data of the PE entity is also read with the instruction "RDREC (Page 4557)".

Below is a flowchart of the write and read jobs:



Parameter

The following table shows the parameters of the instruction "PE_START_END":

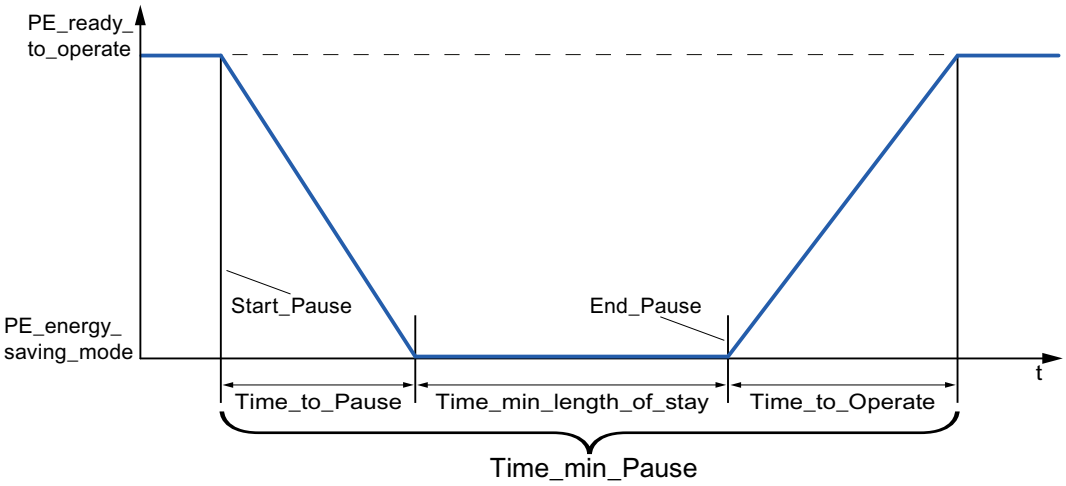
Parameter	Declaration	Data type	Memory area	Description
START	Input	BOOL	I, Q, M, D, L or constant	Transmitting the PE command "Start_Pause" to the PE entity with the address set at parameter ID.
END	Input	BOOL	I, Q, M, D, L or constant	Transmitting the PE command "End_Pause" to the PE entity with the address set at parameter ID.
ID	Input	HW_ SUBMODULE	I, Q, M, D, L or constant	Address of the PE entity Use the hardware ID of the head module for a PROFINET IO device. Refer to the system constants of the assigned IO controller for the hardware ID. The name of the head module is composed of the name of the IO device and the suffix [Head] (example: "IO_Device_1[Head]").
PAUSE_TIME	Input	TIME	I, Q, M, D, L, P or constant	Planned pause duration. • Range: T#1MS to T#24D20H31M23S647MS • Start value: T#0MS
VALID	Output	BOOL	I, Q, M, D, L	PE command successfully sent.
BUSY	Output	BOOL	I, Q, M, D, L	PE command still being processed.
ERROR	Output	BOOL	I, Q, M, D, L	Error occurred during processing. The error message is output at the STATUS parameter.
STATUS	Output	DWORD	I, Q, M, D, L, P	Block status / error number (see "STATUS parameter")
PE_MODE_ID	Output	BYTE	I, Q, M, D, L, P	Identification number of energy-saving mode (energy-saving level for the duration of the pause).

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

PAUSE_TIME parameter

The parameter PAUSE_TIME is used to set the default duration of the energy-saving period at the PE entity. The PE entity checks whether the pause period is of sufficient length and whether it can be implemented. The minimum pause duration (Time_min_Pause) must be

greater than the sum of the times the device needs to switch to energy-saving mode (Time_to_Pause) and to switch back to operating mode (Time_to_Operate).



ET 200S checks whether the scheduled pause duration is greater than or equal to the minimum pause duration (PM-E_Pause_Min) saved on the ET 200S. This is fixed at 10 seconds. If a shorter pause is used, the power modules (PM-E) of the ET 200S will remain on.

The module does not switch back on automatically at the end of the pause; it remains in OFF mode until the "END" command is sent. This prevents uncoordinated switch-on, which can cause undesired peak loads.

STATUS parameter

Error information is output at the STATUS output parameter. If it is interpreted as ARRAY[1...4] of BYTE, the error information has the following structure:

Field element	Name	Meaning
STATUS[1]	Function_Num	Cause of the error • B#16#**00**: No error • B#16#**DE**: Error during reading of data record • B#16#**DF**: Error during writing of data record • B#16#**C0**: Error message from the instruction or from the communication instructions "RDREC (Page 4557)" and "WRREC (Page 4559)" used internally.
STATUS[2]	Error_Decode	Location of the error ID • **80**: DPV1 error as defined in IEC 61158-6 or application-specific • **FE**: DP/PNIO profiles - PROFIenergy-specific error

Field element	Name	Meaning
STATUS[3]	Error_Code_1	Error ID • When Error_Decode = **80**: – **80**: Simultaneous positive edge at the input parameters START and END. – **81**: Length conflict at the CMD_PARAM and CMD_PARAM_LEN parameters. – **82-8F**: Other error messages (reserved) • When Error_Decode = **FE**: – **01**: "Service Request ID" invalid – **02**: Incorrect "Request_Reference" – **03**: "Modifier" invalid – **04**: "Data Structure Identifier RQ" invalid – **05**: "Data Structure Identifier RS" invalid – **06**: "PE energy-saving modes" are not supported – **07**: "Response" is too long (maximum transmissible length exceeded) – **08**: "Count" invalid – **50**: No suitable "energy mode" is available. – **51**: Time value specified is not supported. – **52**: "PE_Mode_ID" invalid
STATUS[4]	Error_Code_2	Manufacturer-specific error ID extension

Note

Error messages of the instructions RDREC and WRREC

The instruction "PE_START_END" uses the instructions "WRREC (Page 4559)" and "RDREC (Page 4557)" for communication. Error messages for these instructions are output in the field elements STATUS[1] to STATUS[4].

Please see the description of the corresponding STATUS (Page 4566) parameter for an explanation of the error codes of the "WRREC (Page 4559)" and "RDREC (Page 4557)" instructions.

See also

Description of PROFIenergy (Page 4628)

PE_CMD: Start and exit energy-saving mode / Read out status information

Description

The instruction "PE_CMD" is used in the PE controller to start or terminate a pause in the energy-saving mode in the PE entity. Additional information and energy measurements can also be read out from a PE entity using "PE_CMD".

The instruction is best used with PE controllers to which assigned PE devices are connected from which energy measurements are to be read out. If this is not the case, the instruction "PE_START_END (Page 4630)" can also be used to start and end the pauses.

Transfer of the PROFIenergy commands (PE commands)

The instruction "PE_CMD" transfers a PROFIenergy command to a PE entity.

The instruction can also be used if additional commands are to be added to the PROFIenergy profile in future. The commands which can be used with the current PROFIenergy profile are listed in the description of the CMD and CMD_MODIFIER parameters (see the "CMD and CMD_MODIFIER parameters" table).

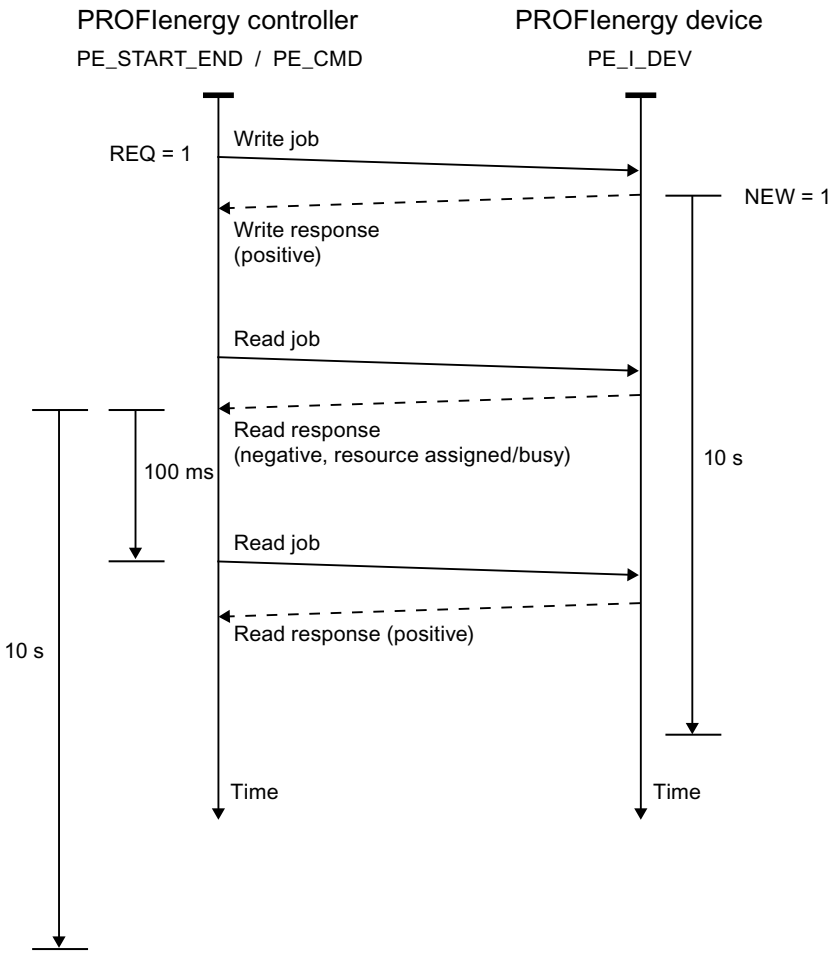
- The individual PE commands which are transferred to the PE entity using the instruction are assigned defined "Service_Request_IDs". The Service_Request_IDs 01 to 05 and 16 are assigned at the CMD parameter.
- The two PE commands 04 (Query_Modes) and 16 (Query_Measurement) are defined in more detail with the CMD_MODIFIER parameter.
- Additional values are transferred at the CMD_PARA parameter for individual PE commands (see the description of the individual PE commands). The parameter CMD_PARA_LEN defines the length of the data at the parameter CMD_PARA.

The commands are transferred without plausibility test. The response data of the PE entity is saved to the RESPONSE_DATA data area that is addressed by VARIANT pointer (for information about the response frame contents, refer to the description of the respective PE command).

Write and read jobs of the "PE_CMD" instruction.

The instruction "PE_CMD" uses "WRREC (Page 4559)" internally to transmit a PROFIenergy command as a write job to the PE entity. "PE_CMD" then waits for the acknowledgment from the PE entity. The acknowledgment data record is read with the instruction "RDREC (Page 4557)" every 100 milliseconds. Until acknowledgment is received from the PE entity, the function repeats the read job for 10 seconds at intervals of 100 milliseconds. The response data of the PE entity is also read with the instruction "RDREC (Page 4557)".

Below is a flowchart of the write and read jobs:



Parameter

The following table shows the parameters of the instruction "PE_CMD":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Starts transferring the PE command upon a positive edge.
ID	Input	HW_ SUBMODULE	I, Q, M, D, L or constant	Address of the PE entity Use the hardware ID of the head module for a PROFINET IO device. Refer to the system constants of the assigned IO controller for the hardware ID. The name of the head module is composed of the name of the IO device and the suffix [Head] (example: "IO_Device_1[Head]").

Parameter	Declaration	Data type	Memory area	Description
CMD	Input	BYTE	I, Q, M, D, L, P or constant	Service-Request-ID of the PROFIenergy command in accordance with the PROFIenergy profile (see "CMD and CMD_MODIFIER parameters"). Further service request IDs are possible following extensions of the PROFIenergy profile.
CMD_MODIFIER	Input	BYTE	I, Q, M, D, L, P or constant	PROFIenergy sub-command (only when CMD=3 or CMD=16, see "CMD and CMD_MODIFIER parameters") Further sub-commands are possible following extensions of the PROFIenergy profile.
CMD_PARA	Input	VARIANT	I, Q, M, D, L	Parameters for the PE commands: • Get mode: PE_mode_ID • Get measurement values: List of Measurement_Ids The complete Service Data Request is entered.
CMD_PARA_LEN	Input	INT	I, Q, M, D, L, P or constant	The actual length of the command parameters (<= length defined in CMD_PARA; is verified by the instruction).
RESPONSE_DATA	InOut	VARIANT	I, Q, M, D, L	PROFIenergy information May be complete response frame including block header, depending on the command. Note: If the buffer is too small, only the number of bytes which is specified in the VARIANT pointer will be entered.
VALID	Output	BOOL	I, Q, M, D, L	Command successfully sent.
BUSY	Output	BOOL	I, Q, M, D, L	Command still being processed.
ERROR	Output	BOOL	I, Q, M, D, L	Error occurred during processing.
STATUS	Output	DWORD	I, Q, M, D, L	Block status / error number (see "STATUS parameter"):

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameters CMD and CMD_MODIFIER

CMD	CMD_ MODIFIER	PROFIenergy command	Description
01	0	Start_Pause (Page 4654)	Start energy-saving mode or switch to a different energy-saving mode.
02	0	End_Pause (Page 4654)	End energy-saving mode.
03	1	Query_Modes - List of energy-saving modes (Page 4655)	Output the energy-saving modes supported.

CMD	CMD_ MODIFIER	PROFIenergy command	Description
	2	Query_Modes - Get mode (Page 4656)	Output attributes of the energy-saving mode currently enabled.
04	0	PEM_Status (Page 4658)	Query status of energy-saving mode.
05	0	PE_Identify (Page 4659)	Read out the number and description of PE commands supported.
16	1	Query_Measurement - Get_Measurement_List (Page 4660)	List of measured values supported by the PE entity.
	2	Query_Measurement - Get_Measurement_Values (Page 4662)	Output the measured values of the PE entity.

STATUS parameter

Error information is output at the STATUS output parameter. If it is interpreted as ARRAY[1...4] of BYTE, the error information has the following structure:

Field element	Name	Meaning
STATUS[1]	Function_Num	Cause of the error • B#16#00: No error • B#16#DE: Error during reading of data record • B#16#DF: Error during writing of data record • B#16#C0: Error message from the internal communication instructions "RDREC (Page 4557)" and "WRREC (Page 4559)".
STATUS[2]	Error_Decode	Location of the error ID • 80: DPV1 error as defined in IEC 61158-6 or application-specific • FE: DP/PNIO profiles - PROFIenergy-specific error

Field element	Name	Meaning
STATUS[3]	Error_Code_1	Error ID • When Error_Decode = **80**: – **81**: Length conflict at the CMD_PARA and CMD_PARA_LEN parameters, or maximum data record length (4095 bytes) exceeded. – **82-8F**: Other error messages (reserved) • When Error_Decode = **FE**: – **01**: "Service Request ID" invalid – **02**: Incorrect "Request_Reference" – **03**: "Modifier" invalid – **04**: "Data Structure Identifier RQ" invalid – **05**: "Data Structure Identifier RS" invalid – **06**: "PE energy-saving modes" are not supported – **07**: "Response" is too long (maximum transmissible length exceeded) – **08**: "Count" invalid – **50**: No suitable energy-saving mode (energy mode) is available. – **51**: Time value specified is not supported. – **52**: "PE_Mode_ID" invalid
STATUS[4]	Error_Code_2	Manufacturer-specific error ID extension

Note

Error messages of the instructions RDREC and WRREC

The instruction "PE_CMD" uses the instructions "WRREC (Page 4559)" and "RDREC (Page 4557)" for communication. Error messages for these instructions are output in the field elements STATUS[1] to STATUS[4].

Please see the description of the corresponding STATUS (Page 4566) parameter for an explanation of the error codes of the "WRREC (Page 4559)" and "RDREC (Page 4557)" instructions.

See also

Description of PROFIenergy (Page 4628)

PE_DS3_Write_ET200S: Set power module switching behavior

Description

The instruction "PE_DS3_Write_ET200S" sends basic settings for power module switching behavior to the ET 200S. The switching behavior of up to 8 slots in the ET 200S (e.g. for power modules) can be defined with the instruction "PE_DS3_Write_ET200S".

Note

This instruction is not part of the PROIenergy profile but instead supplements SIMATIC-specific functions.

Parameter

The following table shows the parameters of the instruction "PE_DS3_Write_ET200S":

Parameter	Declaration	Data type	Memory area	Description
ENABLE	Input	BOOL	I, Q, M, D, L or constant	A positive edge triggers the transfer of the data record. The data record must be transferred again after voltage OFF/ON.
ID	Input	HW_SUBMODULE	I, Q, M, D, L or constant	Address of the ET 200S The address may be taken from the hardware configuration.
SLOT_NO_X	Input	INT	I, Q, M, D, L, P or constant	Slot number of switchable power module X.
FUNC_X	Input	INT	I, Q, M, D, L, P or constant	Function of the module in this slot. Use the parameter FUNC_X to define the switching behavior of the PM-E (power module of the ET 200S): • FALSE: – With "PAUSE_START": -No effect on PM-E -PM-E remains on – With "PAUSE_STOP": -Switches PM_E back on • TRUE: – With "PAUSE_START": -Switches PM_E off – With "PAUSE_STOP": -Switches PM-E back on
BUSY	Output	BOOL	I, Q, M, D, L	Transfer not yet complete.
DONE	Output	BOOL	I, Q, M, D, L	Transfer completed without errors.
ERROR	Output	BOOL	I, Q, M, D, L	Transfer completed with error.
STATUS	Output	DWORD	I, Q, M, D, L, P	Error number (see STATUS parameter of the instruction "PE_Start_End (Page 4630)")

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

See also

Description of PROFIenergy (Page 4628)

PE_WOL: Starting and stopping energy-saving mode via WakeOnLan

Description PE_WOL

Description

The "PE_WOL" instruction sends the PROFIenergy commands "Start_Pause" and "End_Pause" to multiple PROFIenergy-enabled devices in the PROFINET I/O systems.

Multiple PE devices can be coordinated via the instruction provided that the PE devices support the "Wake on LAN" function over a UDP connection.

The "PE_WOL" instruction can only be executed on a CPU with and integrated Ethernet interface. This CPU should be able to load blocks with a size of about 400 KB. You cannot use the block with PROFINET I/O systems that are connected via an Ethernet CP.

The instruction "PE_WOL" is executed asynchronously.

Definition: Wake on LAN

The Wake on LAN function allows data processing equipment to resume working from a state near total shutdown when a special Ethernet packet is received.

For this procedure to work, the data processing equipment must have a network controller that is capable of receiving such a packet.

This packet (Magic Packet™) has a special format. It contains the MAC address of the network adapter 15 times.

Selection of devices

The devices are selected using the user data block at the parameter PENERGY (Type: "PE_PLUS"). The user DB represents the database for processing multiple devices.

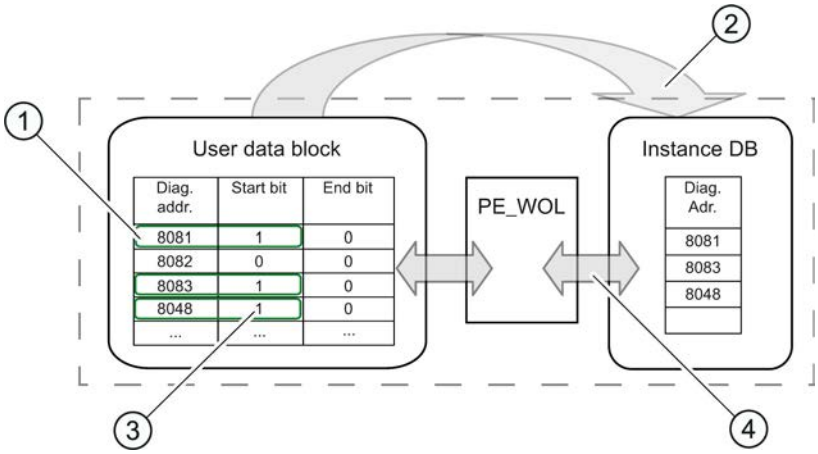
Prior to initialization of "PE_WOL", you must store at least the following information in the user DB:

- ID of the PROFINET I/O system
- Data of the connection which is used for "Wake On LAN".

- Port number which is used for "Wake On LAN".
- For each device
  - Pause time (PauseTime)
  - Switching of the device to the PE_SLEEP_MODE (EnableSleep)

You initialize the "PE_WOL" instruction with the COM_RST parameter. The jobs stored in the user DB are processed one after the other after initialization.

The graphic below illustrates how the PE command "Start_Pause" is transmitted to several devices:



(1) Step 1: The bit "CmdStartPause" of the devices to be shut down is set to "1" by the user.

(2) Step 2: The diagnostic addresses of the devices to be shut down (CmdStartPause = "1") are linked to the queue.

(3) Step 3: The bit "CmdStartPause" is reset automatically once the jobs have been linked.

(4) Step 4: The instruction "PE_WOL" processes the jobs as soon as they are linked.

A PROFIenergy command "CmdStartPause" or "CmdEndPause" can be sent for all recognized devices in the PROFINET IO system via the START and END parameters.

The status of the job processing as well as possible errors during processing are output by the STATUS parameter.

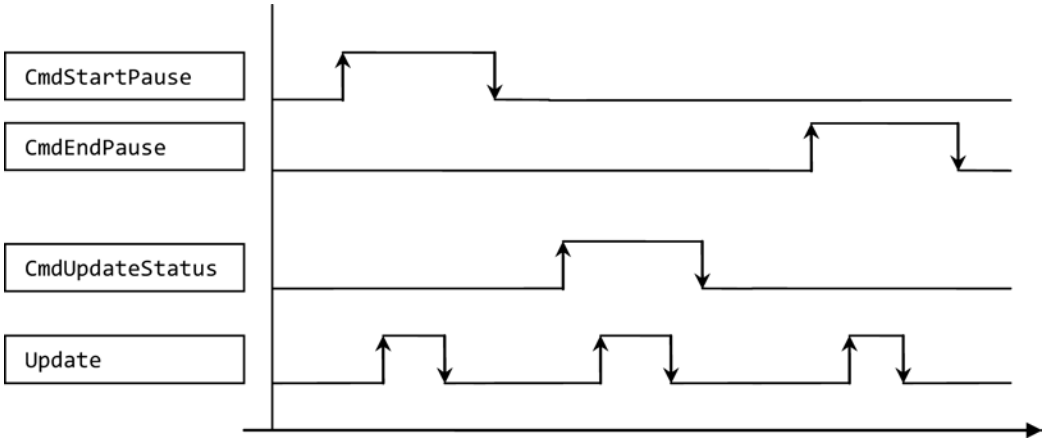
Operation of the instruction using the user DB

The operation of the instruction "PE_WOL" can only take place via the user DB. A basic procedure applies in this case:

1. Selection of the command to be executed for a device:
   - START_PAUSE ("CmdStartPause" in user DB)
   - ENDE_PAUSE ("CmdEndPause" in user DB)
   - UPDATE_STATUS ("CmdUpdateStatus" in user DB)
2. Setting the bit for updating ("Update" in the header of the user DB)
   At least one CPU cycle should pass with an "Update" = False between two updates; otherwise, edge detection cannot be guaranteed.

Prioritizing the PE commands

The graphic below shows the chronological sequence of three possible commands.



These are processed one after the other, regardless of whether the previous command call was successful or completed with an error.

Only one command is executed if you set two commands, such as "CmdEndPause" and "CmdUpdateStatus" simultaneously. A prioritization is present within the block:

- The command "CmdStartPause" has the highest priority and is therefore always executed if selected.
- The command "CmdEndPause" has the second highest priority.
- The command "CmdUpdateStatus" has the lowest priority.

If all three commands are set at the same time, the commands that are not executed remain preselected. In this case, the next rising edge executes the next command.

Parameters

The following table shows the parameters of the "PE_WOL" instruction:

Parameter	Declaration	Data type	Memory area	Description
COM_RST (Page 4644)	Input	BOOL	I, Q, M, D, L	Resets the block and performs a re-initialization. As long as True is set here, the initialization is started but not yet fully completed. Only the falling edge continues the initialization and switches to the normal operating mode after initialization.
START (Page 4645)	Input	BOOL	I, Q, M, D, L	A rising edge performs a "CmdStartPause" PROFIenergy command for all detected devices that support this function.
END (Page 4646)	Input	BOOL	I, Q, M, D, L	A rising edge performs a "CmdEndPause" PROFIenergy command for all detected devices that support this function.

Parameter	Declaration	Data type	Memory area	Description
PENERGY (Page 4646)	InOut	PE_PLUS	D	Pointer to the user DB that contains the database for processing multiple devices.
STATUS (Page 4650)	Output	DWORD	I, Q, M, D, L	Status/error number for the current status of the instruction (see "STATUS parameter").

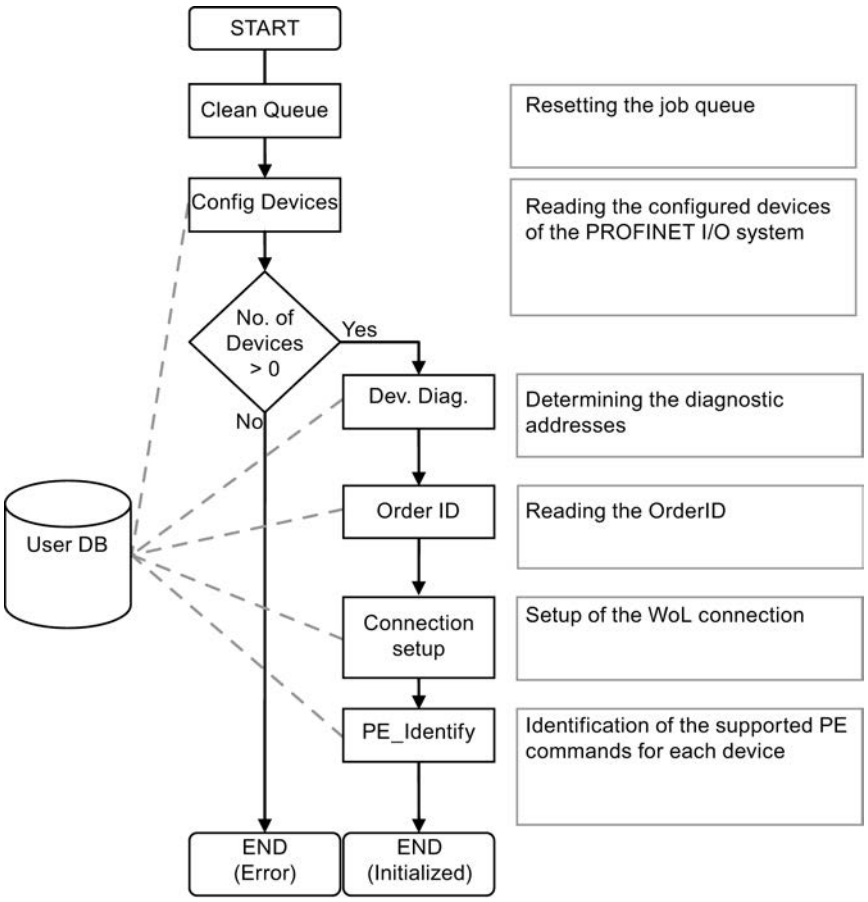
For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter COM_RST

Sequence of the initialization routine

You start the initialization of the "PE_WOL" instruction with the COM_RST parameter.

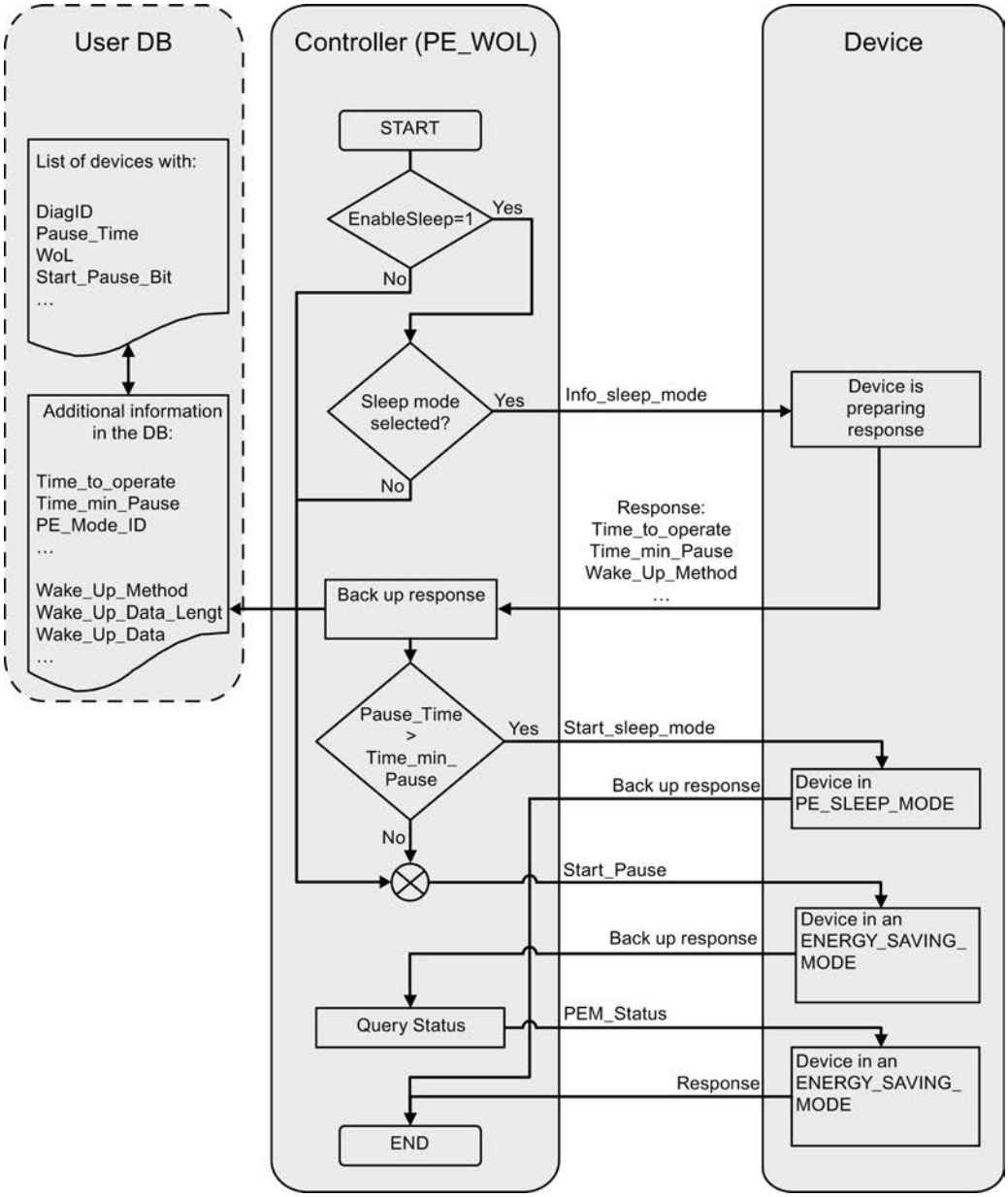
The following flow diagram shows the initialization routine.



Parameter START

Sequence of the CmdStartPause command

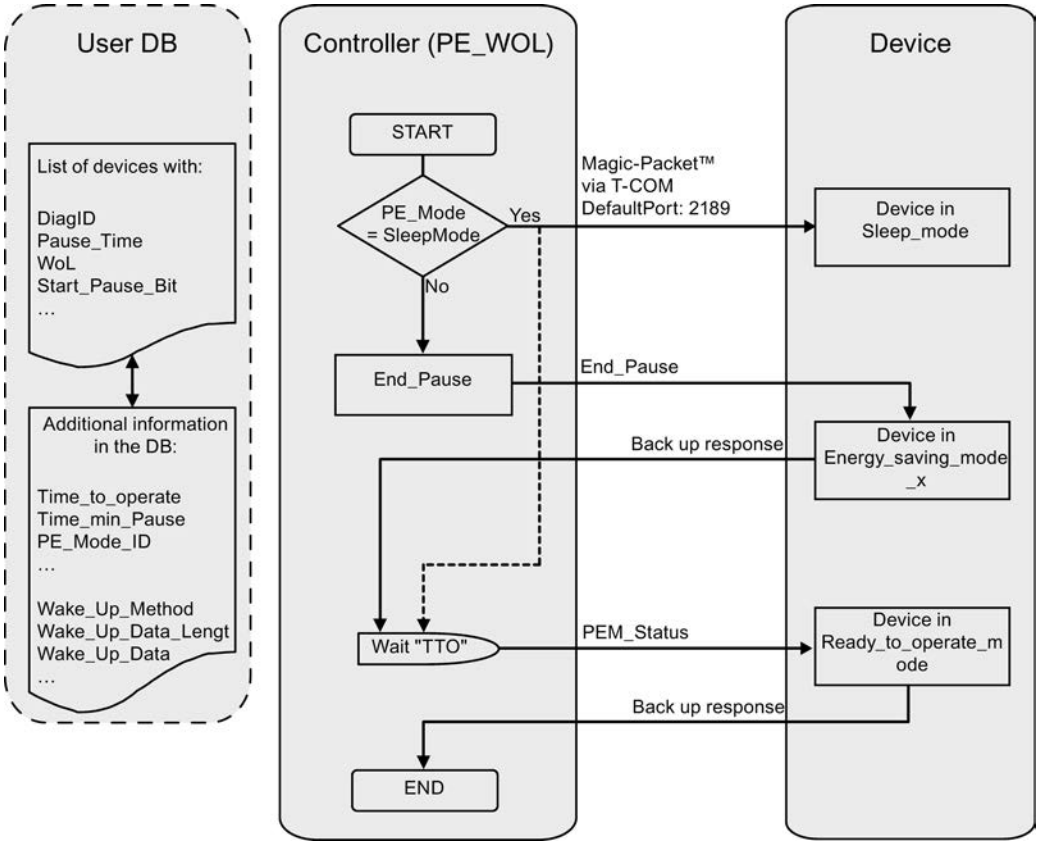
The flow diagram shows the internally used functions and interaction with a device when executing the CmdStartPause command.



Parameter END

Sequence of the CmdEndPause command

The flow diagram shows the internally used functions and interaction with a device when executing the CmdEndPause command.



PENERGY parameter

Data block at the PENERGY parameter

The user DB for the "PE_WOL" PROFIenergy instruction represents a data base for processing multiple devices.

The data block is usually divided into two parts. These are:

- Header 110 bytes
- Device section for a maximum of 256 devices each with 100 bytes (Device). These include:
  - Device-specific data (Device)
  - PROFIenergy-specific data (PE)
  - Data for the job processing (Task)
  - User data (UserData)

The data block operates with optimized access.

Connection parameter "Connection"

The "PE_WOL" instruction reserves a connection resource in the area of "Open User Communication". This is used as a UDP connection. The following parameters must be defined in the data block for this:

- Connection ID ("Connection.id" parameter)
  The connection ID is an integer within the range of 1 to 32. The default setting is 31. It is used to identify the communication resources allocated by the firmware, for example the send and receive buffers.
  The connection ID must be unique throughout the CPU.
- Port number that is used for the "Wake on LAN" function ("Header.PortNo" parameter)
  Number of the UDP port via which a "Wake On LAN" packet is sent. These port numbers are part of the communication resources, which are identified and provided by the firmware through the connection ID. The default setting for port 2189 used here is currently not assigned by IANA. The port number is transferred to the connection configuration. This affects the "Connection.local_tsap_id[1]" and "Connection.rem_tsap_id[1]" parameters.
- Interface ID ("Connection.local_device_id" parameter)
  The interface ID is also part of the connection description. This ID identifies the CPU interface to use for this connection. There are several possible valid values.
  However, these need to be adapted to the employed CPU and interface:
  - B#16#01 for ET200S CPU or WinAC RTX with Ethernet interface in subslot IF1
  - B#16#02 for CPU 315(F)-2PN/DP or CPU 317(F)-2PN/DP
  - B#16#03 for CPU 319(F)-2PN/DP
  - B#16#05 for CPU 41x(F)-3PN/DP
  - B#16#06 for WinAC RTX with Ethernet interface in subslot IF2
  - B#16#0B for WinAC RTX with Ethernet interface in subslot IF3
  - B#16#0F for WinAC RTX with Ethernet interface in subslot IF4

Structure of the data block

The data block has the following structure:

Name	Data type	Offset	Comment
Header	PE_HEADER	-	Header information
Update (1)	BOOL	-	Signal for indicating a change in the data area. • True= indicates a change by the user. • False = indicates the application of changes.
Initialized	BOOL	-	Signal indicating the completed initialization. • True=Initialization finished. • False= indicates that the block is not initialized.
LinkUp	BOOL	-	Indicates successful configuration of the Ethernet interface. • True=interface ready for use. • False=interface not yet configured.
LinkDown	BOOL	-	Indicates an unconfigured interface. • True=interface is not yet configured. • False=interface is currently being configured or is now configured.
PROFINET_ID (1)	INT	-	ID of the PROFINET I/O system
Reserved	ARRAY [1..37] OF BYTE	-	Reserved
LastDeviceID	INT	-	Contains the highest Device-ID in this PROFINET I/O system.
PortNo (1)	WORD	-	Port number that is used for the "Wake on LAN" function (default= 2189).
Connection	TCON_Param	-	Contains the connection configuration of the "Wake on LAN" connection.
BLOCK_LENGTH	UInt	-	Length of the structure (always B#16#40).
ID (1)	CONN_OUC	-	Connection ID
CONNECTION_TYPE (1)	USINT	-	Connection type (UDP = B#16#13)
ACTIVE_EST (1)	BOOL	-	Active connection establishment (always passive for UDP)
LOCAL_DEVICE_ID (1)	USINT	-	Contains the interface ID (CPU dependent)
LOCAL_TSAP_ID_LEN (1)	USINT	-	Contains the length in bytes of the own/local UDP port
REM_SUBNET_ID_LEN (1)	USINT	-	Unused (always B#16#00)
REM_STADDR_LEN (1)	USINT	-	Contains the length of the remote IP address or B#16#00.
REM_TSAP_ID_LEN (1)	USINT	-	Contains the length in bytes of the remote UDP port.
NEXT_STADDR_LEN (1)	USINT	-	Contains the length of the default router address (not relevant).
LOCAL_TSAP_ID (1)	ARRAY[1..16] OF BYTE	-	Contains the own/local port number.
REM_SUBNET_ID (1)	ARRAY[1..6] OF USINT	-	Unused (always B#16#00)
REM_STADDR (1)	ARRAY[1..6] OF USINT	-	Contains the remote IP address.

Name	Data type	Offset	Comment
REM_TSAP_ID (1)	ARRAY[1..16] OF BYTE	-	Contains the remote UDP port number.
NEXT_STADDR (1)	ARRAY[1..6] OF BYTE	-	irrelevant
SPARE (1)	WORD	-	
Device	ARRAY[1..256] OF PE_DEVICE	-	Array of devices
Device	PE_DEV	-	Contains data for each device
DiagID	HW_IO	-	Diagnostic address of the device. Assigned by the hardware configuration.
InterfaceID	WORD	-	Diagnostic address of the device interface. Assigned by the hardware configuration.
MACAdr	ARRAY[1..6] OF BYTE	-	Contains the MAC address of the device.
IPAdr	ARRAY[1..4] OF BYTE	-	Contains the IP address of the device.
OrderID_MxLen	Byte	-	Contains the maximum length of the OrderID.
OrderId_ActLen	Byte		Contains the actual length of the OrderID.
OrderID_Data	ARRAY[1..20] OF CHAR		Contains the OrderID of the device.
PE	PE_PE	-	PROFIenergy-specific data.
ModeID	BYTE	-	PE_MODE_ID according to Profienergy specification.
Result	BYTE	-	PE ErrorCode according to PROFIenergy specification.
PauseTime (1)	TIME	-	Contains the idle time in ms.
TimeToPause	TIME	-	Contains the time it takes for the device to go into Pause mode.
TimeToOperate	TIME	-	Contains the time it takes for the device to go into operating mode.
MinSleepTime	TIME	-	Contains the minimum time of the device in PE_SLEEP_MODE.
SleepToOperate	TIME	-	Contains the time it takes for the device to go from PE_SLEEP_MODE into operating mode.
StatusOperate	BOOL	-	Indicates the operating mode of the device.
StatusPause	BOOL	-	Indicates the device is in Pause mode.
StatusSleep	BOOL	-	Indicates the PE_SLEEP_MODE of the device.
StatusTransitOK	BOOL	-	Indicates the transition from one energy state to another.
StatusInTransit	BOOL	-	Indicates a current state transition.
StatusTransitNOK	BOOL	-	Indicates that the state change was not successful.
StatusError	BOOL	-	Indicates an error with the device.
StatusRetryEx	BOOL	-	Indicates an unsuccessful execution of a command. No more attempts will be made to execute this command.
CmdStartPause (1)	BOOL	-	Places a START_PAUSE command for this device in the queue.
CmdEndPause (1)	BOOL	-	Places a END_PAUSE command for this device in the queue.

Name	Data type	Offset	Comment
CmdUpdateStatus (1)	BOOL	-	Places a PEM_STATUS command for this device in the queue.
EnableSleep (1)	BOOL	-	Allows PE_SLEEP_MODE for this device. • True= device should go to PE_SLEEP_MODE if idle time is sufficient • False=device may not go into PE_SLEEP_MODE.
Services	WORD	-	Shows all supported PROFIenergy services.
UserData (2)	ARRAY[1..24] OF BYTE	-	User-defined data
Task	PE_TASK	-	Job processing
Cmd	BYTE	-	Internal bits for job processing
CmdJ	BYTE	-	Internal bits for job processing
TimeStart	BOOL	-	Starts a delay time.
TimeStarted	BOOL	-	The delay time has just started.
TimeDone	BOOL	-	Shows the expiration of the delay time.
Done	BOOL	-	Indicates that this job has been completed.
DelayedCmd	BOOL	-	Indicates that a delayed command is still pending.
IsV1_0	BOOL	-	Indicates that this device is a Spec. V1.0 device.
IsWakeOnLAN	BOOL	-	Indicates that this device is woken up by "Wake On LAN".
RetryCount	BYTE	-	Repetition counter for PE_COMMANDS
Duration	TIME	-	Contains the value for the delay in ms.
StartTime	TIME	-	Contains the start time for the delay time.
MachineState	INT	-	Contains the internal status of the job.

(1) To be filled in by the user.
(2) Available for use by the user.

Parameter STATUS

Parameter STATUS

The output value at the STATUS parameter is divided into three areas:

- Bits 31 to 24: MESSAGE
- Bit 23 to 16: LOCATION
- Bit 15 to 0: INFORMATION

You can learn the meaning of each error code for the three areas in the following tables:

Table 9-55 Possible values for MESSAGE

Error code (W#16#...)	Description
00	No error.
50	Instruction initialized.

Error code (W#16#...)	Description
51	Determining configuration of the PROFINET I/O system.
52	The instruction could not determine any configured devices in the PROFINET I/O system.
53	Determining the logical addresses of configured devices.
54	Reading interface information out of the devices.
55	Determining I&M data (data record 0 only) of the configured devices.
56	Configuring PROFINET interface for sending the "Wake on LAN" MagicPaket™ via UDP.
57	Determining PROFIenergy capabilities of the connected devices.
62	Invalid PROFINET I/O system ID detected. The causative number is displayed in the INFORMATION field.
70	The instruction is initiated and processing jobs. The value in the INFORMATION field contains the number of currently active jobs.
80	The instruction has lost the initialization during processing of jobs. This generally occurs when either the instance DB or the user DB has been reloaded.
FF	Unknown error has occurred.

Table 9-56 Possible values for LOCATION

Error code (W#16#...)	Description
00	The instruction is either not initialized or it is idle.
70	The instruction is waiting for jobs.
71	The instruction has appended a job to the job list.
72	The instruction is preparing to send a job.
73	The instruction is sending a job to a device.
74	The instruction is waiting for a response from the device.
75	The instruction is evaluating the response of the device.
76	The instruction is removing the job from the job list.
FF	Unknown error has occurred.

Table 9-57 Possible values for INFORMATION

Error code (W#16#...)	Description
0000	No additional information available and no jobs active.
0001 -00FF	1 – 255 jobs are currently being processed.
8001	Error at parameter 1
8002	Error at parameter 2
8003	Error at parameter 3
8004	Error at parameter 4

Error code (W#16#...)	Description
8005	Error at parameter 5. This error is reported if there is no interconnection or an invalid interconnection to the user DB. Possible causes: • The user DB is too small. • The user DB is write-protected. • There is no user DB in the RAM. • The user DB is invalid for the employed CPU.
8100	An attempt has been made to place more than the maximum of 256 possible jobs. This is a temporary error, which is resolved by completing a few jobs. The placed jobs are not accepted and must be placed again.
8200	An attempt has been made to send an invalid or an unsupported PROFIenergy command (PE_COMMAND).
84xx	A communication error has occurred. The number of the device that generated the error is output to "xx".
85xx	An error has been returned from device xx. The number of the device that generated the error is output to "xx".
8600	The requested wake-up method is currently not supported.
FFFF	Unknown error has occurred.

PROFIenergy commands

Structure of the message frames

Structure of the response frame in accordance with the PROFIenergy profile

The table below shows the basic structure of the response frame in accordance with the PROFIenergy profile. The response frame consists of a general section (Header) and a specific section (Service Data Response). The contents of the specific section of the response frame can be found in the description of the relevant PROFIenergy command.

Block definition	Attributes	Value	Data type	Description
BlockHeader	BlockType	801_{hex}	WORD	
	BlockLength		WORD	Number of bytes without consideration of the fields BlockType and BlockLength.
	BlockVersionHigh	1_{hex}	BYTE	
	BlockVersionLow	0_{hex}	BYTE	

Block definition	Attributes	Value	Data type	Description
Response Header	Service_Request_ID	1_{hex} to FF $_{hex}$	BYTE	ID of the PI command executed. The ID of the PE command processed by the PE entity is returned in the response frame: • 01: Start_Pause • 02: End_Pause • 03: Query_Modes • 04: PEM_Status • 05: PE_Identify • 06 to 09: Reserved • 16: Query_Measurement • 11 to CF: Reserved • D0 to FF: Manufacturer-specific
	Request_Reference	1_{hex} to FF $_{hex}$	BYTE	Unique number to identify the query/ response pair (returned by the server in the response).
Service Header Response	Status	1_{hex} to FF $_{hex}$	BYTE	Information whether or not the PI command was executed: • 00: Reserved • 01: Completed • 02: Completed with error • 03: Data incomplete • 04 to CF: Reserved • D0 to FF: Depends on the Service_Request_ID
	Data_Structure_ Identifier_RS	1_{hex} to FF $_{hex}$	BYTE	• 00: Reserved • 01 to FF: Data structure depends on the Service_Request_ID • 0xFF - Error
Service Data Response				Depends on the Service-Request-ID: • For the service request IDs, see the CMD and CMD_MODIFIER parameters of the instruction "PE_CMD (Page 4634)". • The specific contents of the response frame can be found in the description of the PE command (see for example the "Start_Pause (Page 4654)" command).

PI Command "Start_Pause"

Description

Use the PE command "Start_Pause" to start the energy-saving mode. The command Start_Pause can be used to:

- Switch the PE from "ready" state (PE_ready_to_operate) to an energy-saving mode (PE_energy_saving_mode).
- The PE entity can automatically switch between the energy-saving modes.
  Energy consumption can increase or decrease when you switch energy-saving mode.

Calling the PI command "Start_Pause"

The command "Start_Pause" is called with the instruction "PE_CMD (Page 4634)" with the following parameters:

Parameter	Value	Description
CMD	1	Call of PE command "Start_Pause".
CMD_MODIFIER	0	There are no additional specifications of the command call for command "Start_Pause".
CMD_PARA_LEN	4	Length of the CMD_PARA parameter of 4 bytes.
CMD_PARA	VARIANT	VARIANT pointer to the value for "Pause_Time" (TIME).

Response frame (Service Data Response)

The following response frame data of the PE entity is written to the data block that is referenced at parameter RESPONSE_DATA (see instruction "PE_CMD (Page 4634)"):

Attribute	Value	Data type	Description
PE_Mode_ID	1 to 255	BYTE	Identification number of the energy-saving mode
Reserved	0	BYTE	-

PI Command "End_Pause"

Description

The PE command "End_Pause" is used to end the energy-saving mode of the PE entity.

Calling the PE command "End_Pause"

The command "End_Pause" is called with the instruction "PE_CMD (Page 4634)" with the following parameters:

Parameter	Value	Description
CMD	2	Call of PE command "End_Pause".
CMD_MODIFIER	0	There are no additional specifications of the command call for command "End_Pause".
CMD_PARA_LEN	0	Length of the CMD_PARA parameter of 0 bytes.
CMD_PARA	irrelevant	-

Response frame (Service Data Response)

The following data of the PE entity response frame is written to the data block referenced at the RESPONSE_DATA parameter (see "PE_CMD (Page 4634)"):

Attribute	Value	Data type	Description
Time_to_operate	-	DWORD	Expected time it takes to switch to the "PE_ready_to_operate" mode.

PI command "Query_modes" - "List_Energy_Saving_Modes"

Description

Use PE command "Query_modes" and sub-command (modifier) "List_Energy_Saving_Modes" to output all energy-saving modes (PE_Mode_ID) supported by the PE entity.

The query result is written in the form of a response frame to the data block referenced by theRESPONSE_DATA parameter.

Calling the PE command "Query_modes" - "List_Energy_Saving_Modes"

The command "List_Energy_Saving_Modes" is called with the instruction "PE_CMD (Page 4634)" with the following parameters:

Parameter	Value	Description
CMD	3	Call of PE command "Query_modes".
CMD_MODIFIER	1	Specification of the command call: Select the sub-command "List_Energy_Saving_Modes" to output the number and types of energy-saving modes supported.
CMD_PARA_LEN	0	Length of the CMD_PARA parameter of 0 bytes.
CMD_PARA	irrelevant	-

Response frame (Service Data Response)

The following data of the PE entity response frame is written to the data block referenced at the RESPONSE_DATA parameter (see "PE_CMD (Page 4634)"):

Attribute	Value	Data type	Description
Number_of_PE_ Mode_IDs	1	BYTE	The number of energy-saving modes.
PE_Mode_IDs	-	Array [...] of BYTE	Array with the IDs of the supported energy-saving modes. The meaning of the individual IDs depends on the PE entity.

PI command "Query_modes" - "Get_Mode"

Description

You use the PE command "Query_modes" and the sub-command (modifier) "Get_Mode" to output the attributes of the energy-saving mode which is currently enabled.

Calling the PE command "Query_modes" - "Get_Mode"

The call of the command with the instruction "PE_CMD" occurs with the following parameters:

Parameter	Value	Description
CMD	3	Call of the PE command "Query_modes"
CMD_MODIFIER	2	Specification of the command call: Select the sub-command "Get_Mode" to output the status of the mode which is currently enabled.
CMD_PARA_LEN	1	Length of the CMD_PARA parameter of 1 byte.
CMD_PARA	VARIANT	VARIANT pointer to the value for PE_MODE_ID.

Response frame (Service Data Response)

The following data of the PE entity response frame is written to the data block referenced at the RESPONSE_DATA parameter (see "PE_CMD (Page 4634)"):

Attribute	Value	Data type	Description
PE_Mode_ID	• 0 "PE_power_off" mode • 1...254 Energy-saving mode of the PE entity (manufacturer-specific) • 255 "PE_ready_to_operate" mode	BYTE	ID of the energy-saving mode currently enabled.
PE_Mode_Attributes	Bit 0: • = 0: Only static energy consumer and time values are available. • = 1: Dynamic energy consumer and time values are available. Bit 1 to 7: • Reserved	BYTE	
Time_min_Pause [1]	Time difference without date	DWORD	Minimum pause time for the PI mode. The minimum pause time is the sum of the values of the following attributes: • Time_to_Pause • Time_to_operate • Time_min_length_of_stay See the description "PAUSE_TIME parameter" of the instruction "PE_START_END: Start and exit energy-saving mode (Page 4630)".
Time_to_Pause [1]	Time difference without date	DWORD	Switch off time: Duration from call of energy-saving mode to start of energy-saving mode (transition duration of PE_ready_to_operate to PE_energy_saving_mode). The switch-off time depends on the PE entity.
Time_to_operate [1]	Time difference without date	DWORD	Switch-on time: Duration of the transition from energy-saving mode (PE_energy_saving_mode) to ready to operate mode (PE_ready_to_operate). The PE entity calculates the duration dynamically in the output operation.
Time_min_length_of_stay [1]		DWORD	Minimum active period of the energy-saving mode on the PE entity.
Time_max_length_of_stay [1]		DWORD	Maximum active period of the energy-saving mode on the PE entity.
Mode_Power_Consumption [2]		REAL	Power consumption of the PE entity in active energy-saving mode. Unit: kW

Attribute	Value	Data type	Description
Energy_ Consumption_ to_pause [2]		REAL	Energy consumption of the PE entity during transition from ready-to-operate mode (PE_ready_to_operate) to energy-saving mode (PE_energy_saving_mode) Unit: kWh
Energy_ Consumption_ to_operate [2]		REAL	Energy consumption of the PE entity during transition from energy-saving mode (PE_energy_saving_mode) to ready-to-operate mode (PE_ready_to_operate) Unit: kWh
[1] If the duration is infinite, 0xFFFFFFFF will be output. If the duration is zero, "0" will be output. [2] If the data for energy and power consumption is not defined by the PE entity, "0.0" will be output as the value.			

PI command "PEM_Status"

Description

The PE command "PEM_Status" is used to query the status of a PE entity energy-saving mode that is currently enabled.

Calling the PE command "PEM_Status"

The command "PEM_Status" is called with the instruction "PE_CMD" with the following parameters:

Parameter	Value	Description
CMD	4	Call of PE command "PEM_Status".
CMD_MODIFIER	0	There are no other specifications of the command call for command "PEM_Status".
CMD_PARA_LEN	0	Length of the CMD_PARA parameter of 0 bytes.
CMD_PARA	irrelevant	-

Response frame (Service Data Response)

The following data of the PE entity response frame is written to the data block referenced at the RESPONSE_DATA parameter (see "PE_CMD (Page 4634)"):

Attribute	Value	Data type	Description
PE_Mode_ID_ Source	• 0 "PE_power_off" mode • 1 to 254 Energy-saving mode of the PE entity (manufacturer-specific) • 255 "PE_ready_to_operate" mode	BYTE	Mode of the PE entity before a PE command is sent.
PE_Mode_ID_ Destination	• 0 "PE_power_off" mode • 1 to 254 Energy-saving mode of the PE entity (manufacturer-specific) • 255 "PE_ready_to_operate" mode	BYTE	Mode of the PE entity after a PE command is executed.
Time_to_operate	Time difference without date	DWORD	Switch-on time: Duration of the transition from energy-saving mode (PE_energy_saving_mode) to ready to operate mode (PE_ready_to_operate). The PE entity calculates the duration dynamically during output.
Remaining_time_to_ destination	Time difference without date	DWORD	Remaining time to switch to another mode.
Mode_Power_ Consumption		REAL	Power consumption of the PE entity in active energy-saving mode. Unit: kW
Energy_ Consumption_ to_Destination		REAL	Energy consumption for the current PI transition Unit: kWh
Energy_ Consumption_ to_operate		REAL	Energy consumption of the PE entity during transition from energy-saving mode (PE_energy-saving mode) to ready-to-operate mode (PE_ready_to_operate) Unit: kWh

PI command "PE_Identify"

Description

The PE command "PE_Identify" is used to read out the number and description of PE commands supported by the PE entity. The number of specific commands supported depends on the PE entity. As PE_Identify is itself a PE command, at least three supported PE commands will be output upon a positive response: Start_Pause, End_Pause and PE_Identify.

Calling the PE command "PE_Identify"

The command "PE_Identify" is called with the instruction "PE_CMD (Page 4634)" with the following parameters:

Parameter	Value	Description
CMD	5	Call of the command "PE_Identify".
CMD_MODIFIER	0	There are no other specifications of the command call for command "PE_Identify".
CMD_PARA_LEN	0	Length of the CMD_PARA parameter of 0 bytes.
CMD_PARA	irrelevant	-

Response frame (Service Data Response)

The following data of the PE entity response frame is written to the data block referenced at the RESPONSE_DATA parameter (see "PE_CMD (Page 4634)"):

Attribute	Value	Data type	Description
Count [1]	6	BYTE	Number of supported PROFIenergy commands
Start_Pause	1	BYTE	First PE command supported (Service_Request_ID)
End_Pause	2	BYTE	...
Query_Modes	3	BYTE	...
PEM_Status	4	BYTE	...
PE_Identify	5	BYTE	...
Query_Measurement	16	BYTE	Last PE command supported (Service_Request_ID)
[1] The number of commands supported is manufacturer-specific and depends on the PE entity used. The values given are an example of a response frame where all 6 PE commands are supported.			

PI Command "Query_Measurement" - "Get_Measurement_list"

Description

Use the PE command "Query_Measurement" and sub-command (modifier) "Get_measurement_list" to query the specific measured values supported by the PE entity. The supported measured values are output as a list in the data block referenced by the RESPONSE_DATA parameter.

Calling the PE command "Query_Measurement" - "Get_Measurement_list"

The command is called with the instruction "PE_CMD (Page 4634)" with the following parameters:

Parameter	Value	Description
CMD	16	Call of the command "Query_Measurement".
CMD_MODIFIER	1	Specification of the command call: Select the sub-command "Get_Measurement_List" to output a list of supported measured values.
CMD_PARA_LEN	0	Length of the CMD_PARA parameter of 0 bytes.
CMD_PARA	irrelevant	-

Response frame (Service Data Response)

The following data of the PE entity response frame is written to the data block referenced at the RESPONSE_DATA parameter (see "PE_CMD (Page 4634)"):

Attribute	Value	Data type	Description
Count	-	BYTE	Number of measurement IDs
reserved	-	BYTE	
...			
Measurement_ID	-	WORD	First supported Measurement_ID The Measurement_ID is manufacturer-specific. For more information, refer to the manual of the respective PE entity.
Accuracy_Domain	-	BYTE	See "accuracy domain" table.
Accuracy_Class	-	BYTE	See "accuracy classes" table.
Range	-	REAL	Specifies the scale end value of the measured value (only for accuracy domain 1). The attribute range uses the same unit as defined using attribute Measurement_ID (only one unit is used for each Measurement_ID).
...			
Measurement_ID	-	WORD	Last supported Measurement_ID
Accuracy_Domain	-	BYTE	See "accuracy domain" table.
Accuracy_Class	-	BYTE	See "accuracy classes" table.
Range	-	REAL	Specifies the scale end value of the measurement value (only for accuracy domain 1). The attribute range uses the same unit as defined using attribute Measurement_ID (only one unit is used for each Measurement_ID).

Accuracy domains

Accuracy domain	Description
0	Reserved
1	The accuracy deviation is output in percentages of the scale end value. The percentage of the possible deviation is divided into accuracy classes (see table: Accuracy classes of the accuracy domains 1 and 2).

Accuracy domain	Description
2	The accuracy deviation is output in percentages of the current measurement value. The percentage of the possible deviation is divided into accuracy classes (see table: Accuracy classes of the accuracy domains 1 and 2).
3	The measuring accuracy adheres to the standard IEC 61557-12. The function performance classes for performance measurement and monitoring devices (PMD) without external sensors, and the system performance classes for PMD with external sensors, are coded as set out in the "Accuracy classes of accuracy domain 3" table.
4	The entry of the accuracy adheres to the standard EN 50470-3, chapter 8 (see also table: Accuracy classes of accuracy domain 4).

Accuracy classes

Table 9-58 Accuracy classes of the accuracy domains 1 and 2

Accuracy class	0	1	2	3	4	5	6	7	8
Meaning	Reserved	0.01%	0.02%	0.05%	0.1%	0.2%	0.5%	1%	1.5%

Accuracy class	9	10	11	12	13	14	15	>15
Meaning	2%	2.5%	3%	5%	10%	20%	>20%	Not defined

Table 9-59 Accuracy classes of accuracy domain 3

Accuracy class	0	1	2	3	4	5	6	7	8
Meaning	Reserved	0,02	0,05	0,1	0,2	0,5	1	1,5	2

Accuracy class	9	10	11	12	13	14	>13
Meaning	2,5	3	5	10	20	20%	Not defined

Table 9-60 Accuracy classes of accuracy domain 4

Accuracy class	0	1	2	3	4	5	6	>7
Meaning	Reserved	0,5	1,0	1,5	2,0	2,5	3,0	Not defined

PI Command "Query_Measurement" - "Get_Measurement_values"

Description

Use the PE command "Query_Measurement" and sub-command (modifier) "Get_measurement_values" to output a list of measured values supported by the PE entity. The measured values are output as a list in the data block referenced by the RESPONSE_DATA parameter.

Calling the PE command "Query_Measurement" - "Get_Measurement_values"

The command is called with the instruction "PE_CMD (Page 4634)" with the following parameters:

Parameter	Value	Description
CMD	16	Call of the command "Query_Measurement".
CMD_MODIFIER	2	Specification of the command call: Select the command "Get_Measurement_Values" to output a list of supported measurement values.
CMD_PARA_LEN	0	Depends on the number of measurement values. The length of the parameters results from the attribute count and the sum of the lengths of the attributes for the transferred measurement values.
CMD_PARA	VARIANT	VARIANT pointer to data structure with list of measured values to be queried (see "CMD_PARA parameter").

Parameter CMD_PARA

The structure specified with the VARIANT pointer at the CMD_PARA parameter must have the following setup:

Attribute	Value	Data type	Description
Count	-	BYTE	Number of measured values (Measurement-IDs)
reserved	0	BYTE	Not used
Measurement_ID	-	WORD	First measured value queried
...			
Measurement_ID	-	WORD	Last measured value queried

Response frame (Service Data Response)

The following response frame data of the PE entity is written to the data block that is referenced at parameter RESPONSE_DATA (see "PE_CMD (Page 4634)"):

Attribute	Value	Data type	Description
Count [1]	-	BYTE	Number of measured values (Measurement-IDs)
reserved	0	BYTE	Not used
Length_of_Structure	2 to 65535	WORD	Length of the structure in bytes
Measurement_Data_ Structure_ID	1 = simple value	BYTE	Defines the following structure.
Measurement_ID	0 to 65535	WORD	ID of the supported measurement value.
Status_of_ Measurement_Value	1 to 3	BYTE	Status of the measurement value: • 1: Valid • 2: Not supported • 3: Invalid
Transmission_Data_ Type	-	REAL	

Attribute	Value	Data type	Description
End_of_demand	-	TOD	Optional time stamping with data type TimeOfDay.
...			
Length_of_Structure	-	WORD	Length of the structure in bytes
Measurement_Data_ Structure_ID	-	BYTE	Defines the following structure.
Measurement_ID	-	WORD	ID of the supported measurement value.
Status_of_Measurem ent_Value	-	BYTE	Status of the measurement value: • 1: Valid • 2: Not supported • 3: Invalid
Transmission_Data_ Type	-	REAL	
End_of_demand	-	TOD	Optional time stamping with data type TimeOfDay.
[1] If the data length of the measurement values queried exceeds the size of the PDU (Protocol Data Unit) of the protocol layer, the data will not be completely transferred and only the supported measurement values will be output.			

iDevice / iSlave

PE_I_DEV: Control PROFIenergy commands in the I-Device

Description

The instruction "PE_I_DEV" is used to process the PROFIenergy profile in the intelligent IO device (iDevice). The functions which are executed by the firmware in standard PROFIenergy-compatible IO devices, such as the ET 200S, are implemented in the iDevice by the instruction "PE_I_DEV" and the corresponding auxiliary blocks:

- The instruction "PE_I_DEV" is called cyclically by the user program of the iDevice and receives all PROFIenergy commands.
- The PROFIenergy response is generated by configuring an auxiliary block. The response in the pause is fully programmable. The response data must be provided within 10 seconds; otherwise, "State conflict 0x80B5" will occur at the STATUS parameter of the instruction in the IO controller.

No specific knowledge of the PROFIenergy standard is required to use the instruction.

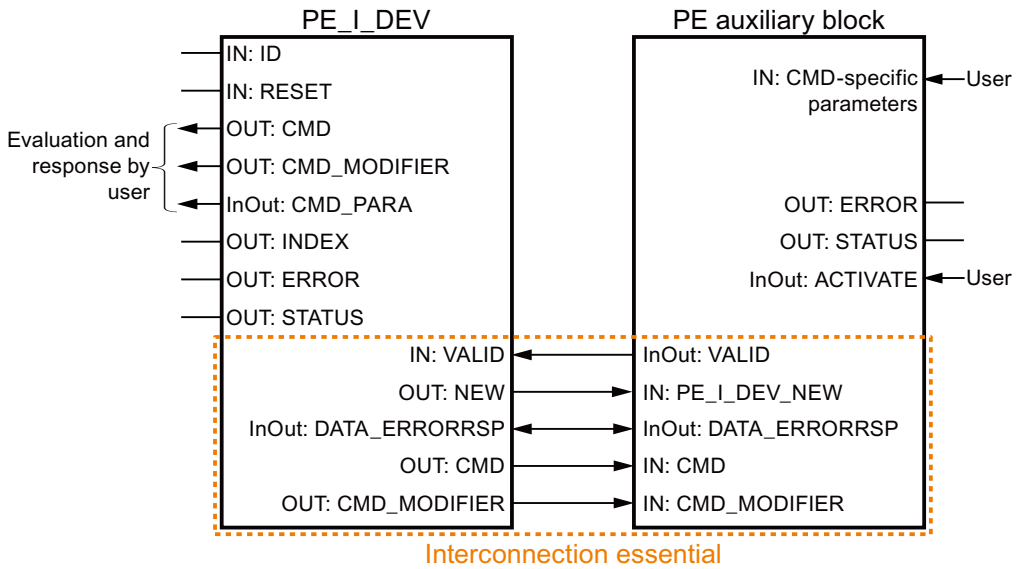
PROFIenergy auxiliary blocks (PE auxiliary blocks)

The PE auxiliary blocks are used to generate the response frame. Enter the response data (in plain text) at the input parameters of the relevant block.

- For each PROFIenergy command, there is a corresponding auxiliary block for a positive response:
  - PE command "Start_Pause": PE_Start_RSP (Page 4669)
  - PE command "End_Pause": PE_End_RSP (Page 4671)
  - PE command "Query_modes" - "List_Energy_Saving_Modes": PE_List_Modes_RSP (Page 4672)
  - PE command "Query_modes" - "Get_Mode": PE_Get_Mode_RSP (Page 4673)
  - PE command "PEM_Status": PE_PEM_Status_RSP (Page 4675)
  - PE command "PE_Identify": PE_Identify_RSP (Page 4677)
  - PE command "Query_Measurement" - "Get_Measurement_list": PE_Measurement_List_RSP (Page 4678)
  - PE command "Query_Measurement" - "Get_Measurement_values": PE_Measurement_Value_RSP (Page 4680)
- Irrespective of the PROFIenergy command, there is an additional shared auxiliary block for a negative response (see PE_Error_RSP (Page 4668)).

Interconnection of the auxiliary blocks

The instruction "PE_I_DEV" and the auxiliary blocks are coordinated. Some of the parameters are simply interconnected. The diagram below shows which of the parameters need to be interconnected.



Parameters

The following table shows the parameters of the instruction "PE_I_DEV":

Parameter	Declaration	Data type	Memory area	Description
RESET	Input	BOOL	I, Q, M, D, L or constant	Resets the instruction.
ID	Input	HW_SUBMODULE	I, Q, M, D, L or constant	Address of the iDevice The address may be taken from the hardware configuration.
VALID	Input	BOOL	I, Q, M, D, L or constant	The response data for the PROFIenergy controller is ready and can be sent.
CMD_PARA	Output	VARIANT	I, Q, M, D, L	Parameters for: • Get mode: PE_mode_ID • Get measurement values: List of Measurement_IDs (list of IDs of the tags to be read; either one tag or multiple tags may be read at any given time). Maximum length: 234 bytes
DATA_ERRORRSP	InOut	VARIANT	I, Q, M, D, L	Pointer to the data area containing the acknowledgment data for the PROFIenergy controller. This must correspond to the pointer which is also used with the auxiliary blocks.
INDEX	Output	INT	I, Q, M, D, L	Data record number of the PROFIenergy record (set at 0x80A0)
CMD	Output	INT	I, Q, M, D, L	Service-Request-ID of the PROFIenergy command in accordance with the PROFIenergy profile (see "CMD and CMD_MODIFIER parameters"). Further PE commands (Service-Request-IDs) are possible following extensions of the PROFIenergy profile.
CMD_MODIFIER	Output	INT	I, Q, M, D, L	PROFIenergy sub-command: • Only when CMD=3 or CMD=16; see "CMD and CMD_MODIFIER parameters". • For all other commands: "0". Further sub-commands are possible following extensions of the PROFIenergy profile.
NEW	Output	BOOL	I, Q, M, D, L	New data available from the PROFIenergy controller.
ERROR	Output	BOOL	I, Q, M, D, L	Command completed with error.
STATUS	Output	DWORD	I, Q, M, D, L	Error information (see "STATUS parameter").

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameters CMD and CMD_MODIFIER

CMD	CMD_ MODIFIER	PROFIenergy command	Description
01	0	Start_Pause	Start energy-saving mode or switch to a different energy-saving mode.
02	0	End_Pause	End energy-saving mode.
03	1	Query_Modes - List energy saving Modes	Output the energy-saving modes supported.
	2	Query_Modes - Get Mode	Output attributes of the energy-saving mode currently enabled.
04	0	PEM_Status	Query status of energy-saving mode.
05	0	PE_Identify	Read out the number and description of PE commands supported.
16	1	Query_Measurement - Get_Measurement_List	List of measured values supported by the PE entity.
	2	Query_Measurement - Get_Measurement_Values	Output of the measured values of the PE entity.

STATUS parameter

Error information is output at the STATUS output parameter. If it is interpreted as ARRAY[1...4] of BYTE, the error information has the following structure:

Field element	Name	Meaning
STATUS[1]	Function_Num	Cause of the error • B#16#**00**: No error • B#16#**DE**: Error during reading of data record • B#16#**DF**: Error during writing of data record • B#16#**C0**: Error message from the instruction "PE_I_DEV" or from the internal communication instructions "RDREC (Page 4557)" and "WRREC (Page 4559)".
STATUS[2]	Error_Decode	Location of the error ID • **80**: DPV1 error as defined in IEC 61158-6 or application-specific
STATUS[3]	Error_Code_1	Error ID (when Error_Decode = **80**): • **B1**: Write length error (error in write length or insufficient length information with the data type VARIANT).
STATUS[4]	Error_Code_2	For PROFINET errors: Output of IO controller error message If no PROFINET error has occurred, the value in STATUS[4] = "0".

Note

Error messages of the instructions RDREC and WRREC

The instruction "PE_I_DEV" uses the instructions "WRREC (Page 4559)" and "RDREC (Page 4557)" for communication. Error messages for these instructions are output in the field elements STATUS[1] to STATUS[4].

Please see the description of the corresponding STATUS (Page 4566) parameter for an explanation of the error codes of the "WRREC (Page 4559)" and "RDREC (Page 4557)" instructions.

See also

Description of PROFIenergy (Page 4628)

Auxiliary blocks of the instruction PE_I_DEV

PE_Error_RSP: Generate negative answer to command

Description

The auxiliary block "PE_Error_RSP" (Response with failure) generates a negative response if the command requested is not supported (either in general or temporarily). Response generation does not depend on the command requested.

Parameter

The following table shows the parameters of the "PE_Error_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 4664)" instruction. The auxiliary block is only processed if the parameter value is set to "1".
ERROR_CODE	Input	BYTE	I, Q, M, D, L or constant	Error number
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 4664)" instruction.

Parameter	Declaration	Data type	Memory area	Description
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 4664)" instruction. The auxiliary block sets the parameter as soon as the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ ERRORRSP	InOut	VARIANT	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 4664)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in VARIANT setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

PE_Start_RSP: Generate answer to command at start of pause

Description

The auxiliary block "PE_Start_RSP" (Start Pause) generates the response to the PE command Start_Pause (Page 4654). The instruction returns the energy-saving mode to which the device has switched (PE_MODE_ID parameter).

If the response differs depending on the length of the pause, you can return this fact in the feedback on the energy-saving mode (PE_Mode_ID = 1 for a brief pause, PE_Mode_ID = 2 for a longer pause, etc.).

Parameter

The following table shows the parameters of the "PE_Start_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 4664)" instruction. The auxiliary block is only processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 4664)" instruction.
PE_MODE_ID	Input	BYTE	I, Q, M, D, L or constant	PE mode that the process assumes
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 4664)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 4664)" instruction. The auxiliary block sets the parameter as soon as the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ERRORRSP	InOut	VARIANT	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 4664)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in VARIANT setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

PE_End_RSP: Generate answer to command at end of pause

Description

The auxiliary block "PE_End_RSP" generates the response to the PE command End_Pause (Page 4654). The response returned is the time required to switch from the current mode to "Ready_To_Operate" mode.

Parameter

The following table shows the parameters of the "PE_End_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 4664)" instruction. The auxiliary block is only processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 4664)" instruction.
Time_to_Operate	Input	DWORD	I, Q, M, D, L or constant	Time required to switch from the current mode to "Ready_To_Operate".
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 4664)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 4664)" instruction. The auxiliary block sets the parameter as soon as the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ ERRORRSP	InOut	VARIANT	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 4664)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes

Parameter	Declaration	Data type	Memory area	Description
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in VARIANT setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

PE_List_Modes_RSP: Generate queried energy savings modes as answer

Description

The auxiliary block "PE_List_Modes_RSP" generates the response to the PE command List_Energy_Saving_Modes (Page 4655). The response generated includes the number and the IDs of energy-saving modes supported.

Parameter

The following table shows the parameters of the "PE_List_Modes_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 4664)" instruction. The auxiliary block is only processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 4664)" instruction.
CMD_MODIFIER	Input	INT	I, Q, M, D, L or constant	PROFIenergy sub-command (evaluated only if CMD=3, or CMD=16). The parameter must be interconnected with the CMD_MODIFIER output parameter of the "PE_I_DEV" instruction.
Number_of_PE_Mode_IDs	Input	BYTE	I, Q, M, D, L or constant	Number of energy-saving modes supported. Permitted values: 1 to 254
PE_MODE_ID	Input	VARIANT	I, Q, M, D, L	Points to the area in which the IDs of the energy-saving modes supported (PE_Mode_ID) are stored. Valid range: 1 to 254.

Parameter	Declaration	Data type	Memory area	Description
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 4664)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 4664)" instruction. The auxiliary block sets the parameter as soon as the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ ERRORRSP	InOut	VARIANT	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 4664)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in VARIANT setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

PE_Get_Mode_RSP: Generate queried energy data as answer

Description

The auxiliary block "PE_Get_Mode_RSP" generates the response to the command Get_Mode (Page 4656). The response frame contains the time, performance, or energy data of an energy-saving mode.

Parameter

The following table shows the parameters of the "PE_Get_Mode_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 4664)" instruction. The auxiliary block is only processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 4664)" instruction.
CMD_MODIFIER	Input	INT	I, Q, M, D, L or constant	PROFIenergy sub-command (evaluated only if CMD=3, or CMD=16). The parameter must be interconnected with the CMD_MODIFIER output parameter of the "PE_I_DEV" instruction.
PE_Mode_ID	Input	BYTE	I, Q, M, D, L or constant	ID of the energy-saving mode currently enabled.
Time_min_Pause	Input	DWORD	I, Q, M, D, L or constant	Minimum pause duration for this PE energy-saving mode. This is the sum of the following three parameters: • Time_to_Pause • Time_to_operate • Time_min_length_of_stay
Time_to_Pause	Input	DWORD	I, Q, M, D, L or constant	Time between the edge at the START parameter (see "PE_I_DEV (Page 4664)") and the requested PE energy-saving mode being reached.
Time_to_Operate	Input	DWORD	I, Q, M, D, L or constant	Maximum switch-on time to "PE_ready_to_operate". The "Time_to_operate" parameter can be used directly for the relevant calculations. The value may be a static maximum, or be calculated dynamically by the PE entity.
Time_min_Lenght_of_stay	Input	DWORD	I, Q, M, D, L or constant	Minimum dwell time of the PE entity in this PE mode.
Time_max_Lenght_of_stay	Input	DWORD	I, Q, M, D, L or constant	Maximum dwell time of the PE entity in this PE mode.
Mode_Power_Consumption	Input	DWORD	I, Q, M, D, L or constant	Energy consumption in the current PE mode in [kW].
Energy_Consum_to_Pause	Input	DWORD	I, Q, M, D, L or constant	Energy consumption from "PE_ready_to_operate" to the current PE mode in [kWh].
Energy_Consum_to_operate	Input	DWORD	I, Q, M, D, L or constant	Energy consumption from the current PE mode to "PE_ready_to_operate" in [kWh].

Parameter	Declaration	Data type	Memory area	Description
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 4664)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 4664)" instruction. The auxiliary block sets the parameter as soon as the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ ERRORRSP	InOut	VARIANT	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 4664)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in VARIANT setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

PE_PEM_Status_RSP: Generate PEM status as answer

Description

The auxiliary block "PE_PEM_Status_RSP" generates the response to the command PEM_Status (Page 4658).

Parameter

The following table shows the parameters of the "PE_PEM_Status_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 4664)" instruction. The auxiliary block is only processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 4664)" instruction.
PE_MODE_ID_ Source	Input	BYTE	I, Q, M, D, L or constant	Source and Destination for PEM_STATUS. Values: • 0x00: PE_POWER_OFF • 0x01 – 0xFE: manufacturer-specific • 0xFF: PE_READY_TO_OPERATE
PE_MODE_ID_ Destination	Input	BYTE	I, Q, M, D, L or constant	
Time_to_Operate[1]	Input	DWORD	I, Q, M, D, L or constant	Maximum switch-on time to "PE_ready_to_operate". "Time_to_operate" can be used directly for the relevant calculations. The value may be a static maximum, or be calculated dynamically by the PE entity.
Remaining_time_ to_destination[1]	Input	DWORD	I, Q, M, D, L or constant	Optional: Time remaining until the requested PE mode. Dynamic value or static maximum value
Mode_Power_ Consumption[2]	Input	DWORD	I, Q, M, D, L or constant	Energy consumption in the current PE mode in [kW].
Energy_ Consumption_ to_Destination[2]	Input	DWORD	I, Q, M, D, L or constant	Energy consumption in the time until the requested PE mode in [kW].
Energy_ Consumption_ to_operate[2]	Input	DWORD	I, Q, M, D, L or constant	Energy consumption from the current PE mode to "PE_ready_to_operate" in [kWh].
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 4664)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 4664)" instruction. The auxiliary block sets the parameter as soon as the response data for the PROFIenergy controller is available and ready for transmission.

Parameter	Declaration	Data type	Memory area	Description
DATA_ ERRORRSP	InOut	VARIANT	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 4664)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in VARIANT setting, e.g. incorrect range
[1] If the time period is unlimited, the maximum value "0xFFFFFFFF" can be specified. If the time period is "Zero", "0x00" can be used. [2] If no energy consumption value has been defined, "0.0" can be specified.				

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

PE_Identify_RSP: Generate supported PROFIenergy commands as answer

Description

The auxiliary block "PE_Identify_RSP" generates the response to the command PE_Identify (Page 4659). In the response to the command, specify which PROFIenergy commands are supported. Please note that PE_IDENTIFY is itself a PE command and has to be included in the response.

Parameter

The following table shows the parameters of the "PE_Identify_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NE W	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 4664)" instruction. The auxiliary block is only processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 4664)" instruction.
Start_Pause	Input	BOOL	I, Q, M, D, L or constant	One parameter for each of the relevant PROFIenergy commands: • 0: PE command is not supported • 1: PE command is supported
End_Pause	Input	BOOL	I, Q, M, D, L or constant	

Parameter	Declaration	Data type	Memory area	Description
Query_Modes	Input	BOOL	I, Q, M, D, L or constant	
PEM_Status	Input	BOOL	I, Q, M, D, L or constant	
PE_Identify	Input	BOOL	I, Q, M, D, L or constant	
Query_ Measurement	Input	BOOL	I, Q, M, D, L or constant	
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 4664)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 4664)" instruction. The auxiliary block sets the parameter as soon as the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ ERRORRSP	InOut	VARIANT	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 4664)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in VARIANT setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

PE_Measurement_List_RSP: Generate list of supported measured values as answer

Description

The auxiliary block "PE_Measurement_List_RSP" generates the response to the command Get_measurement_list (Page 4660). In the response, specify which measured values (Measurement-IDs) are supported.

Parameter

The following table shows the parameters of the "PE_Measurement_List_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 4664)" instruction. The auxiliary block is only processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 4664)" instruction.
CMD_MODIFIER	Input	INT	I, Q, M, D, L or constant	PROFIenergy sub-command (evaluated only if CMD=3, or CMD=16). The parameter must be interconnected with the CMD_MODIFIER output parameter of the "PE_I_DEV" instruction.
Count	Input	BYTE	I, Q, M, D, L or constant	Number of measured values supported (measurement IDs)
Measurement_List	Input	VARIANT	D	Pointer to the array with the Measurement_IDs supported. For information on the structure of the array in accordance with the PROFIenergy profile, see: PI Command "Query_Measurement" - "Get_Measurement_list" (Page 4660)
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 4664)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 4664)" instruction. The auxiliary block sets the parameter as soon as the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ERRORRSP	InOut	VARIANT	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 4664)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes

Parameter	Declaration	Data type	Memory area	Description
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in VARIANT setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

PE_Measurement_Value_RSP: Generate queried measured values as answer

Description

The auxiliary block "PE_Measurement_Value_RSP" generates the response to the command Get_measurement_values (Page 4662). In the response, return the values of the requested measurements.

Parameter

The following table shows the parameters of the "PE_Measurement_Value_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 4664)" instruction. The auxiliary block is only processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 4664)" instruction.
CMD_MODIFIER	Input	INT	I, Q, M, D, L or constant	PROFIenergy sub-command (evaluated only if CMD=3, or CMD=16). The parameter must be interconnected with the CMD_MODIFIER output parameter of the "PE_I_DEV" instruction.
Count	Input	BYTE	I, Q, M, D, L or constant	Number of measured values (Measurement_Values).
Measurement_Values	Input	VARIANT	D	Pointer to the array with the measured values (Measurement_IDs). For information on the structure of the array in accordance with the PROFIenergy profile, see PI Command "Query_Measurement" - "Get_Measurement_values" (Page 4662)

Parameter	Declaration	Data type	Memory area	Description
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 4664)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 4664)" instruction. The auxiliary block sets the parameter as soon as the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ ERRORRSP	InOut	VARIANT	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 4664)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in VARIANT setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

Module parameter assignment

Writing and reading data records

Principle

Some modules have a write-only system data area to which your program can transfer data records. This area contains data records with numbers from 0 to a maximum of 240. Not every module contains all of the data records (see following table).

Modules can also have a read-only system data area in which your program can read data records. This area contains data records with numbers from 0 to a maximum of 240. Not every module contains all of the data records (see following table).

Note

There are modules that have both system data areas. These are physically separate areas and the only thing they have in common is their logical structure.

Write-only system data area

The following table shows the structure of the write-only system data area. This table also shows how the permitted size of the individual data records and which instructions can be used to write them.

Data record number	Contents	Size	Can be written with instruction
0	Parameter	-	WR_DPARM (Page 4689)
1	Parameter	-	WR_DPARM (Page 4689)
2 to 127	User data	Each ≤ 240 bytes	WR_DPARM (Page 4689) WR_REC (Page 4599)
128 to 240	Parameter	Each ≤ 240 bytes	WR_DPARM (Page 4689) WR_REC (Page 4599)

Read-only system data area

The following table shows the structure of the read-only system data area. This table also shows the permitted size of the individual data records and which instructions can be used to read them.

Data record number	Contents	Size	Can be read with instruction
0	Module-specific diagnostics data (set as standard for the entire system)	4 bytes	RD_REC (Page 4594)
1	Channel-specific diagnostics data (incl. data record 0)	4 to 220 Bytes	RD_REC (Page 4594)
2 to 127	User data	Each ≤ 240 bytes	RD_REC (Page 4594)
128 to 240	Diagnostics data	Each ≤ 240 bytes	RD_REC (Page 4594)

System resources

If you start several asynchronous data record transfers one after the other with only short intervals between them, the allocation of system resources by the operating system ensures that all the jobs are executed and that they do not interfere with each other.

If the limits of the system resources are reached, this is indicated in RET_VAL. You can remedy this temporary error situation by simply repeating the job.

The maximum number of "simultaneously" active jobs of a single instruction type depends on the CPU.

RD_DPAR: Read module data record

Description

You use the instruction to read the data record with the number INDEX of the addressed component from the configured system data. This may be a module in a central rack or a distributed component (PROFIBUS DP or PROFINET IO).

The value TRUE for the output parameter VALID indicates that the data record was successfully transferred to the target range RECORD. In this case, the LEN output parameter contains the length of the read data in bytes.

If an error has occurred during transfer of the data record, this is indicated by the output parameter ERROR . In this case, the output parameter STATUS contains the error information.

Functional description

The "RD_DPAR" instruction works asynchronously, that is, its execution extends over multiple calls. You start the data record transfer by calling "RD_DPAR" with REQ = 1.

The output parameter BUSY and bytes 2 and 3 of the output parameter STATUS show the status of the job. Bytes 2 and 3 of STATUS match the output parameter RET_VAL of the instructions that operate asynchronously.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 3150).

The transfer of the data record is complete when the output parameter BUSY has the value FALSE.

Parameter

The following table shows the parameters of the instruction "RD_DPAR":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = 1: Read request
LADDR	Input	HW_IO	I, Q, M, D, L or constant	Identification number of the CPU or the interface. The number is assigned automatically and is stored in the properties of the CPU or of the interface of the hardware configuration.
INDEX	Input	INT	I, Q, M, D, L or constant	Data record number
RECORD	InOut	VARIANT	I, Q, M, D, L	Target range for the read data record
VALID	Output	BOOL	I, Q, M, D, L	New data record was received and is valid
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The job is not yet completed.
ERROR	Output	BOOL	I, Q, M, D, L	ERROR = 1: An error occurred during the reading process.

Parameter	Declaration	Data type	Memory area	Description
STATUS	Output	DWORD	I, Q, M, D, L	• Call ID (bytes 2 and 3) or error code • Byte 1: B#16#00, if no error. Otherwise function ID from DPV1-PDU: In the case of error for data record reading B#16#DE, in the case of error for data record writing B#16#DF. If no DPV1 protocol element is used: B#16#C0. • Byte 4: Manufacturer-specific error ID extension
LEN	Output	INT	I, Q, M, D, L	Length of the read data record information

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

STATUS parameter

Error code* (W#16#...)	Explanation	Restriction
0000	No error	-
7000	First call with REQ=0: No data transfer active; BUSY has the value "0".	-
7001	First call with REQ=1: Data transfer triggered; BUSY has the value "1".	Distributed I/O
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".	Distributed I/O
8090	Address specified at parameter LADDR is invalid.	-
8092	A data type other than (Array of) bit string or integer was specified at the RECORD parameter.	-
8093	This instruction is not valid for the module selected with LADDR.	-
80A1	Negative acknowledgment during sending of data record to module (module defective or unplugged during sending).	-
80A2	DP protocol error at layer 2, possible hardware/interface error in DP slave.	Distributed I/O
80A3	DP protocol error at user interface/user.	Distributed I/O
80A4	Communication problem on com bus.	Error occurs between CPU and external DP interface module
80B0	Instruction for module type not possible, or module does not recognize the data record.	-
80B1	The length of the data record to be sent is incorrect.	-
80B2	The configured slot is not assigned.	-
80B3	Actual module type does not correspond to specified module type.	-
80C1	The data of the previous write job on the module for the same data record have not yet been processed by the module.	-

Error code* (W#16#...)	Explanation	Restriction
80C2	The module is currently processing the maximum possible number of jobs for a CPU.	-
80C3	The required resources (memory, etc.) are currently occupied.	
80C4	Internal temporary error. Job could not be executed. Repeat the job. If this error occurs often, check your installation for sources of electrical interference.	-
80C5	Distributed I/O not available or deactivated.	Distributed I/O
80C6	Data record transfer cancelled due to priority class abort (restart or background).	Distributed I/O
80D0	There is no entry for the module.	-
80D1	The data record number is not configured for the module (data record numbers > 241 are rejected).	-
80D2	The module cannot be assigned parameters according to its type identifier.	-
80D5	The data record is static.	-
General error information	See also: GET_ERR_ID: Get error ID locally (Page 3392)	-
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

See also

RD_DPARM: Read data record from configured system data (Page 4688)

Switching display formats in the program status (Page 2980)

RD_DPARA: Read module data record asynchronously

Description

You use the instruction to read the data record with the number RECNUM of a selected module from the configured system data. The read data record is entered in the target range defined by the parameter RECORD .

Functional description

The "RD_DPARA" instruction works asynchronously, that is, its execution extends over multiple calls. You start the reading process by calling the instruction with REQ = 1.

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 3150).

Parameters

The following table shows the parameters of the instruction "RD_DPARA":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = 1: Read request
LADDR	Input	HW_IO	I, Q, M, D, L or constant	Identification number of the CPU or the interface. The number is assigned automatically and is stored in the properties of the CPU or of the interface of the hardware configuration.
RECNUM	Input	BYTE	I, Q, M, D, L or constant	Data record number (permitted values: 0 to 240)
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code. If no error occurred during the transmission, the following cases are distinguished: • RET_VAL contains the length of the actually read data record in bytes if the target range is larger than the read data record. • RET_VAL contains "0" if the length of the read data record is equal to the length of the target range.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The job is not yet completed.
RECORD	Output	VARIANT	I, Q, M, D, L	Target range for the data record read.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

RET_VAL parameter

Error code* (W#16#...)	Explanation	Restriction
0000	No error	-
7000	First call with REQ=0: No data transfer active; BUSY has the value 0.	-
7001	First call with REQ=1: Data transfer triggered; BUSY has the value "1".	Distributed I/O
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".	Distributed I/O
8090	Specified logical base address invalid.	-
8092	A data type other than (Array of) bit string or integer was specified at the RECORD parameter.	-
8093	This instruction is not valid for the module selected with LADDR.	-
80A1	Negative acknowledgment during sending of data record to module (module defective or unplugged during sending).	-
80A2	DP protocol error at layer 2, possible hardware/interface error in DP slave.	Distributed I/O

Error code* (W#16#...)	Explanation	Restriction
80A3	DP protocol error at user interface/user.	Distributed I/O
80A4	Communication on PBUS+ disrupted	-
80B0	Instruction for module type not possible, or module does not recognize the data record.	-
80B1	The length of the data record to be sent is incorrect. With RD_DPARM (Page 4688): The target range spanned by RECORD is too short.	-
80B2	The configured slot is not assigned.	-
80B3	Actual module type does not correspond to specified module type.	-
80C1	The data of the previous write job on the module for the same data record have not yet been processed by the module.	-
80C2	The module is currently processing the maximum possible number of jobs for a CPU.	-
80C3	The required resources (memory, etc.) are currently occupied.	
80C4	Internal temporary error. Job could not be executed. Repeat the job. If this error occurs often, check your installation for sources of electrical interference.	-
80C5	Distributed I/O not available or deactivated.	Distributed I/O
80C6	Data record transfer cancelled due to priority class abort (restart or background).	Distributed I/O
80D0	There is no entry for the module.	-
80D1	The data record number is not configured for the module (data record numbers > 241 are rejected).	-
80D2	The module cannot be assigned parameters according to its type identifier.	-
80D5	The data record is static.	-
General error information	See also: GET_ERR_ID: Get error ID locally (Page 3392)	-
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

See also

Switching display formats in the program status (Page 2980)

RD_DPARM: Read data record from configured system data

Description

You use instruction to read the data record with the number RECNUM of the addressed module from the configured system data. The read data record is entered into the target range spanned by the RECORD parameter.

Parameters

The following table shows the parameters of the instruction "RD_DPARM":

Parameter	Declaration	Data type	Memory area	Description
IOID	Input	BYTE	I, Q, M, D, L or constant	Address area identifier: • B#16#54 = Peripheral input (PI) • B#16#55 = Peripheral output (PQ) If the module is a mixed module, the area identifier of the lower address must be specified. If the addresses are identical, specify B#16#54.
LADDR	Input	HW_IO	I, Q, M, D, L or constant	Hardware ID of the CPU or the interface. The number is assigned automatically and is stored in the properties of the CPU or of the interface in the hardware configuration.
RECNUM	Input	BYTE	I, Q, M, D, L or constant	Data record number (permitted values: 0 to 240)
RET_VAL	Return	INT	I, Q, M, D, L	Length of the data record read in bytes if the read data record fits in the target range and no error occurred in the transfer. If an error occurs while the instruction is being executed, the return value contains an error code.
RECORD	Output	VARIANT	I, Q, M, D, L	Target range for the data record read.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

RET_VAL parameter

Error code* (W#16#...)	Explanation	Restriction
0000	No error	-
7000	First call with REQ=0: No data transfer active; BUSY has the value "0".	-
7001	First call with REQ=1: Data transfer triggered; BUSY has the value "1".	Distributed I/O
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".	Distributed I/O

Error code* (W#16#...)	Explanation	Restriction
8090	Specified logical base address invalid.	-
8092	A type other than BYTE is specified in the VARIANT reference at the RECORD parameter.	-
8093	This instruction is not valid for the module selected with LADDR.	-
80B1	The target range spanned by RECORD is too short.	-
80C3	The required resources (memory, etc.) are currently occupied.	-
80D0	There is no entry for the module.	-
80D1	The data record number is not configured for the module (data record numbers > 241 are rejected).	-
80D2	The module cannot be assigned parameters according to its type identifier.	-
80D5	The data record is static.	-
General error information	See also: GET_ERR_ID: Get error ID locally (Page 3392)	-
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

See also

Switching display formats in the program status (Page 2980)

WR_DPARM: Transfer data record

Description

You use the instruction "WR_DPARM" to transfer the data record with the number RECNUM from the configuration data to the addressed module. With this instruction, it is irrelevant whether the data record is static or dynamic.

Parameters

The following table shows the parameters of the instruction "WR_DPARM":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = 1: Write request
LADDR	Input	HW_IO	I, Q, M, D, L or constant	Hardware ID of the CPU or of the interface. The number is assigned automatically and is stored in the properties of the CPU or of the interface in the hardware configuration.
RECNUM	Input	BYTE	I, Q, M, D, L or constant	Data record number

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The writing process is not yet completed.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

RET_VAL parameter

Error code* (W#16#...)	Explanation	Restriction
0000	No error	-
7000	First call with REQ=0: No data transfer active; BUSY has the value 0.	-
7001	First call with REQ=1: Data transfer triggered; BUSY has the value "1".	Distributed I/O
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".	Distributed I/O
8090	Address specified at parameter LADDR is invalid.	-
8093	This instruction is not valid for the module selected with LADDR.	-
80A1	Negative acknowledgment during sending of data record to module (module defective or unplugged during sending).	-
80A2	DP protocol error at layer 2, possible hardware/interface error in DP slave.	Distributed I/O
80A3	DP protocol error at user interface/user.	Distributed I/O
80A4	Communication on PBUS+ disrupted	-
80B0	Instruction for module type not possible, or module does not recognize the data record.	-
80B1	The length of the data record to be sent is incorrect.	-
80B2	The configured slot is not assigned.	-
80B3	Actual module type does not correspond to specified module type.	-
80C1	The data of the previous write job on the module for the same data record have not yet been processed by the module.	-
80C2	The module is currently processing the maximum possible number of jobs for a CPU.	-
80C3	The required resources (memory, etc.) are currently occupied.	
80C4	Internal temporary error. Job could not be executed. Repeat the job. If this error occurs often, check your installation for sources of electrical interference.	-
80C5	Distributed I/O not available or deactivated.	Distributed I/O

Error code* (W#16#...)	Explanation	Restriction
80C6	Data record transfer cancelled due to priority class abort (restart or background).	Distributed I/O
80D0	There is no entry for the module.	-
80D1	The data record number is not configured for the module (data record numbers > 241 are rejected).	-
80D2	The module cannot be assigned parameters according to its type identifier.	-
80D5	The data record is static.	-
General error information	See also: GET_ERR_ID: Get error ID locally (Page 3392)	-
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

See also

Switching display formats in the program status (Page 2980)

Interrupts

ATTACH: Attach an OB to an interrupt event

Description

You use the instruction "ATTACH" to assign an organization block (OB) to an event.

You enter the symbolic or numeric name of the organization block in the OB_NR parameter. This will then be assigned to the event specified in the EVENT parameter.

If the event in the EVENT parameter occurs following error-free execution of the "ATTACH" instruction, the organization block in the OB_NR parameter will be called and its program executed.

With the ADD parameter, you specify whether previous assignments of the organization block to other events should be canceled or retained. If the ADD parameter has the value "0", the existing assignments will be replaced by the current assignment.

Parameters

The following table shows the parameters of the "ATTACH" instruction:

Parameter	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_ATT	I, Q, M, D, L or constant	Organization block (numbers up to 32767 are supported.)
EVENT	Input	EVENT_ATT	D, L or constant	Event, e.g., creation of a process event (failure of a hardware module; read hardware ID (16#C0xxyyzz) and query at block)
ADD	Input	BOOL	I, Q, M, D, L or constant	Effects on previous assignments: • ADD=0 (default): This event replaces all previous event assignments for this OB. • ADD=1: This event is added to the previous event assignments for this OB.
RET_VAL	Return	INT	I, Q, M, D, L	Status of the instruction

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code* (W#16#....)	Description
0	No error
8090	OB does not exist
8091	OB is incorrect type
8093	Event does not exist
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

DETACH: Detach an OB from an interrupt event

Description

You use this instruction to cancel the existing assignment of an organization block to one or more events during runtime.

- If you have selected a single event, the assignment of the OB to this event will be cancelled. All other currently existing assignments remain active. You can select an individual event using the drop-down list of the operand placeholder at the EVENT parameter.
- If you have not selected an event, all currently existing assignments of the organization block to events will be canceled.

You enter the symbolic or numeric name of the organization block in the OB_NR parameter. The assignment of this organization block to the event specified in the EVENT parameter will then be canceled.

Parameters

The following table shows the parameters of the "DETACH" instruction:

Parameters	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_ATT	I, Q, M, D, L or constant	Organization block (numbers up to 32767 are supported.)
EVENT	Input	EVENT_ATT	D, L or constant	Event
RET_VAL	Return	INT	I, Q, M, D, L	Status of the instruction

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code* (W#16#....)	Description
0	No error
1	No assignment exists (warning)
8090	OB does not exist
8091	OB has incorrect type
8093	Event does not exist
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

Cyclic interrupt

SET_CINT: Set cyclic interrupt parameters

Description

You use this instruction to set the parameters for a cyclic interrupt OB. The start time for a cyclic interrupt OB is generated from the respective time interval of the OB and the phase offset.

- The time interval of an OB is the interval at which the OB is periodically called. For example, if the time interval is 100 µs, the OB will be called every 100 µs during program execution.
- The phase offset is a time interval by which the call of a cyclic interrupt OB is offset. You can use the phase offset to process low priority organization blocks in a precise time base.

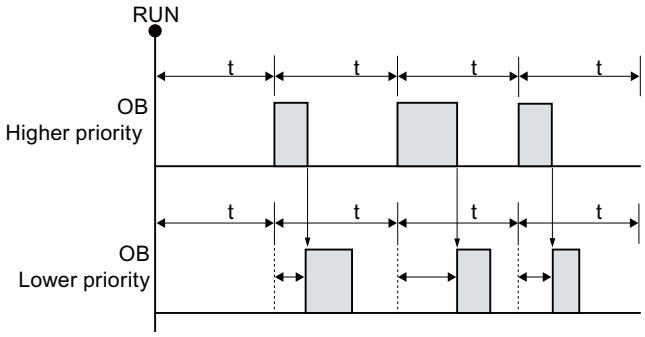
If the OB does not exist or if the time interval used is not supported, a corresponding error alarm is output in the RET_VAL parameter.

A time interval in the CYCLE parameter of "0" means that the OB will not be called.

Functional description

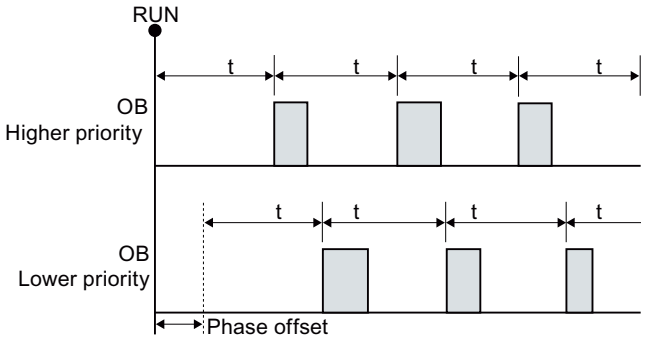
If a lower priority OB and a higher priority OB are called in the same time interval, the lower priority OB will only be called once the higher priority OB has been executed. The call time for the lower priority OB can be offset according to the length of time to execute the higher-priority OB.

OB call without phase offset



If a phase offset is configured for the lower priority OB and the phase offset is greater than the current execution time of the respective higher priority OB, then the block will be called in a fixed time base.

OB call with phase offset



Parameters

The following table shows the parameters of the "SET_CINT" instruction:

Parameter	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_CYCLIC	I, Q, M, D, L or constant	OB number (<32768)
CYCLE	Input	UDINT	I, Q, M, D, L or constant	Time interval in microseconds
PHASE	Input	UDINT	I, Q, M, D, L or constant	Phase offset
RET_VAL	Return	INT	I, Q, M, D, L	Status of the instruction

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code* (W#16#....)	Description
0	No error
8090	OB does not exist or is of the wrong type
8091	Incorrect time interval
8092	Incorrect phase offset
80B2	No event assigned to OB
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

QRY_CINT: Query cyclic interrupt parameters

Description

You can use this instruction to query the current parameters of a cyclic interrupt OB. The cyclic interrupt OB is identified using the OB_NR parameter.

The values of the queried cyclic interrupt parameters correspond to those at the time the "QRY_CINT" instruction is executed.

Parameters

The following table shows the parameters of the "QRY_CINT" instruction:

Parameter	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_CYCLIC (INT)	I, Q, M, D, L or constant	OB number (<32768) or symbolic addressing via the name of the OB (e.g. OB_MyOB)
CYCLE	Output	UDINT	I, Q, M, D, L	Time interval in microseconds
PHASE	Output	UDINT	I, Q, M, D, L	Phase offset
STATUS	Output	WORD	I, Q, M, D, L	Status of the cyclic interrupt: • Bit 0 to bit 4: see parameter STATUS • Other bits: Always "0"
RET_VAL	Return	INT	I, Q, M, D, L	Status of the instruction

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter STATUS

Bit	Value	Meaning
0	0	Not used (always "0").
1	0	The cyclic interrupt is enabled.
	1	The cyclic interrupt is delayed.
2	0	The cyclic interrupt is not enabled or has expired.
	1	The cyclic interrupt is enabled.
3	0	Not used (always "0").
4	0	An OB with the specified number does not exist.
	1	An OB with the specified number exists.
Other bits		Not used (always "0").

Parameter RET_VAL

If an error occurs, the relevant error code will be displayed in the RET_VAL parameter and the STATUS parameter is set to "0".

Error code* (W#16#....)	Description
0	No error.
8090	OB does not exist or is of the wrong type
80B2	No result assigned to OB.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

Time-of-day interrupt

SET_TINT: Set time-of-day interrupt

Description

The instruction "SET_TINT" is used to set the start data and time of the time-of-day interrupt organization blocks from the user program, without having to make settings in the hardware configuration.

- Enter the number of the time-of-day interrupt-OBs at the parameter OB_NR for which you would like to set the start date and the time-of-day.
- Use the parameter SDT and PERIOD to specify when and how often the time-of-day interrupt OB should be called:
  - One-time call: Enter at the parameter SDT the date and the time-of-day and use the value "0" at the parameter PERIOD.
  - Repeated call: Enter the date and the time-of-day of the first call at the parameter SDT. Use the parameter PERIOD to define the time interval in which subsequent calls of the OBs should take place.

Observe the following when setting the start date and start time:

- The date and the time-of-day specification at the parameter SDT refers to the system time.
- The seconds and milliseconds specification is ignored and set to "0" at the start time.
- Only the starting date days 1, 2, ... 28 are possible if you want to specify a monthly time-of-day interrupt OBs. This restriction prevents skipping the monthly call (e.g. 30-day months or in February).
  The setting "end of the month" (W#16#2001) can be used at the parameter PERIOD as an alternative to the 29th 30th and the 31st of the month.

After setting the time-of-day interrupt with "SET_TINT" it still must be activated with the instruction "ACT_TINT".

Note

Additional information about time-of-day interrupt OBs

Other special features for the use of time-of-day interrupt OBs can be found in the descriptions of the organization blocks of the respective CPU:

For the S7-1200: Auto-Hotspot

For the S7-1500: Auto-Hotspot

The settings at the parameters SDT and PERIOD coincide with the settings for time-of-day interrupt in the properties of the time-of-day interrupt OBs.

Parameters

The following table shows the parameters of the "SET_TINT" instruction:

Parameters	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_TOD	I, Q, M, D, L or constant	Number of the time-of-day interrupt OBs • The numbers 10 to 17 are available for the time-of-day interrupt OBs. • An OB number starting with 123 can also be assigned as an alternative. The OB number is displayed in the program block folder and in the system constants.
SDT	Input	DT	D, L or constant	Start date and start time
PERIOD	Input	WORD	I, Q, M, D, L or constant	Execution intervals from the starting point SDT to: • W#16#0000 = single execution • W#16#0201 = once every minute • W#16#0401 = once hourly • W#16#1001 = once daily • W#16#1201 = once weekly • W#16#1401 = once monthly • W#16#1801 = once yearly • W#16#2001 = at month's end
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs during execution of the instruction, the actual parameter of RET_VAL contains an error code.

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

RET_VAL parameter

Error code* (W#16#...)	Description
0000	No error occurred.
8090	Error at the parameter OB_NR (no time-of-day interrupt OB addressed).
8091	Error at the parameter SDT (invalid date and time-of-day specification).
8092	Incorrect input at the parameter PERIOD.
80A1	The set start time is in the past. This error code only occurs at PERIOD = W#16#0000.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 3392)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

ACT_TINT: Enable time-of-day interrupt (Page 4703)

SET_TINTL: Set time-of-day interrupt

Description

The instruction "SET_TINTL" is used to set the start data and time of the time-of-day interrupt organization blocks from the user program, without having to make settings in the hardware configuration.

- Enter the number of the time-of-day interrupt-OBs at the parameter OB_NR for which you would like to set the start date and the time-of-day.
- Use the parameter SDT and PERIOD to specify when and how often the time-of-day interrupt OB should be called:
  - One-time call: Enter at the parameter SDT the date and the time-of-day and use the value "0" at the parameter PERIOD.
  - Repeated call: Enter the date and the time-of-day of the first call at the parameter SDT. Use the parameter PERIOD to define the time interval in which subsequent calls of the OBs should take place.
- Use the parameter LOCAL to select whether the time specified at the parameter SDT refers to the local time or the system time.
- With the ACTIVATE parameter, you specify whether the settings made for the organization block are to be applied directly (ACTIVATE = true) or only after "ACT_TINT (Page 4703)" for the time-of-day interrupt organization block is called (ACTIVATE = false).

Observe the following when setting the start date and start time:

- The seconds and milliseconds specification is ignored and set to "0" at the start time.
- Only the starting date days 1, 2, ... 28 are possible if you want to specify a monthly time-of-day interrupt OBs. This restriction prevents skipping the monthly call (e.g. 30-day months or in February).
  The setting "end of the month" (W#16#2001) can be used at the parameter PERIOD as an alternative to the 29th 30th and the 31st of the month.

Please observe the following during the use of local time:

- Changeover from daylight saving to standard time: When calling time-of-day interrupt organization blocks with a start time within the second hour during changeover from daylight saving time to standard time, also use a time-delay interrupt during the first hour of the time changeover.
- Changeover from standard to daylight saving time: If you specify the skipped hour as the time-of-day for the daylight saving time changeover day, the error code 16#8091 will be output at a single execution (PERIOD = W#16#0000).

Note

Additional information about time-of-day interrupt OBs

Other special features for the use of time-of-day interrupt OBs can be found in the descriptions of the organization blocks of the respective CPU:

For the S7-1200: Auto-Hotspot

For the S7-1500: Auto-Hotspot

The settings at the parameters SDT and PERIOD coincide with the settings for time-of-day interrupt in the properties of the time-of-day interrupt OBs.

Parameters

The following table shows the parameters of the "SET_TINTL" instruction:

Parameters	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_TOD	I, Q, M, D, L or constant	Number of the time-of-day interrupt OBs • The numbers 10 to 17 are available for the time-of-day interrupt OBs. • An OB number starting with 123 can also be assigned as an alternative. The OB number is displayed in the program block folder and in the system constants.
SDT	Input	DTL	D, L or constant	Start date and start time
LOCAL	Input	BOOL	I, Q, M, D, L or constant	• true: Use local time • false: Use system time

Parameters	Declaration	Data type	Memory area	Description
PERIOD	Input	WORD	I, Q, M, D, L or constant	Execution intervals from the starting point SDT to: • W#16#0000 = single execution • W#16#0201 = Once every minute • W#16#0401 = once hourly • W#16#1001 = once daily • W#16#1201 = once weekly • W#16#1401 = once monthly • W#16#1801 = once yearly • W#16#2001 = at month's end
ACTIVATE	Input	BOOL	I, Q, M, D, L or constant	• true: set and activate the time-of-day interrupt • false: set the time-of-day interrupt and only activate at call of "ACT_TINT (Page 4703)"
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs during execution of the instruction, the actual parameter of RET_VAL contains an error code.

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

RET_VAL parameter

Error code* (W#16#...)	Description
0000	No error occurred.
8090	Error at the parameter OB_NR (no time-of-day interrupt OB addressed).
8091	Error at the parameter SDT (invalid date and time-of-day specification).
8092	Incorrect input at the parameter PERIOD.
80A1	The set start time is in the past. This error code only occurs at PERIOD = W#16#0000.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 3392)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

CAN_TINT: Cancel time-of-day interrupt

Description

The instruction "CAN_TINT" is used to delete the start data and start time of a specified time-of-day interrupt organization block. This deactivates the time-of-day interrupt, and the organization block is no longer called.

If you want to reuse the time-of-day interrupt, the start time must be reset (instruction "SET_TINTL (Page 4699)" or "SET_TINT (Page 4697)").

Thereafter, you must reactivate the time-of-day interrupt:

- If you use the instruction "SET_TINT (Page 4697)" or "SET_TINTL (Page 4699)" with parameter ACTIVE=false to set the time-of-day interrupt, call "ACT_TINT (Page 4703)".
- By using the instruction "SET_TINTL (Page 4699)" you can also activate the time-of-day interrupt directly with the parameter ACTIVE=true.

Parameters

The following table shows the parameters of the instruction "CAN_TINT":

Parameters	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_TOD	I, Q, M, D, L or constant	Number of the time-of-day interrupt OB, whose starting date and time-of-day are to be deleted.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs during execution of the instruction, the actual parameter of RET_VAL contains an error code.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	No error occurred.
8090	Error at the parameter OB_NR.
80A0	No starting date/time-of-day defined for the affected time-of-day interrupt OB.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 3392)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

ACT_TINT: Enable time-of-day interrupt

Description

With the instruction "ACT_TINT" you can activate a time-of-day interrupt organization block from the user program. A prerequisite for the execution of the instruction is that the starting date and time-of-day are already set for the time-of-day interrupt OB.

Parameters

The following table shows the parameters of the instruction "ACT_TINT":

Parameters	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_TOD	I, Q, M, D, L or constant	Number of the time-of-day interrupt OBs • The numbers 10 to 17 are available for the time-of-day interrupt OBs. • An OB number starting with 123 can also be assigned as an alternative. The OB number is displayed in the program block folder and in the system constants.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs during execution of the instruction, the actual parameter of RET_VAL contains an error code.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	No error occurred.
8090	Error at the parameter OB_NR (no time-of-day interrupt OB addressed).
80A0	Start date and time-of day not set for the relevant time-of-day interrupt OB.
80A1	The activated time is in the past. The error only occurs if the time-of-day interrupt is only to be executed once.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 3392)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

QRY_TINT: Query status of time-of-day interrupt

Description

You can use this instruction to display the status of a time-of-day interrupt organization block in the STATUS output parameter.

Parameters

The following table shows the parameters of the "QRY_TINT" instruction:

Parameters	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_TOD	I, Q, M, D, L or constant	Number of the time-of-day interrupt OB whose status is queried The OB number is displayed in the program block folder and in the system constants.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs during execution of the instruction, the actual parameter of RET_VAL contains an error code.
STATUS	Output	WORD	I, Q, M, D, L	Status of the time-of-day interrupt (see below)

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	No error occurred. Exception: Status message "0" on bit 4 (no OB with this number created).
8090	Error at the parameter OB_NR. Possible causes: • The value at the parameter OB_NR is not an OB number (<1 or > 32767) supported by the CPU. • The OB number does not address a time-of-day interrupt OB.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 3392)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

Parameter STATUS

If an error occurs (see RET_VAL parameter), "0" is output in the STATUS parameter.

Bit	Value	Meaning
0	0	In Run.
	1	During startup.

Bit	Value	Meaning
1	0	The time-of-day interrupt is enabled.
	1	The time-of-day interrupt is disabled.
2	0	Time-of-day interrupt is not activated or has elapsed.
	1	The time-of-day interrupt is activated.
4	0	An OB with an OB number as specified at OB_NR parameter does not exist.
	1	An OB with an OB number as specified at OB_NR parameter does exist.
6	0	The time-of-day interrupt is based on the system time
	1	The time-of-day interrupt is based on the local time
Other		Always "0"

See also

Switching display formats in the program status (Page 2980)

Time-delay interrupt

Using time-delay interrupts

Definition

After you have called the "SRT_DINT (Page 4706)" instruction, the operating system generates an interrupt after the specified delay time has elapsed, in other words, the assigned time-delay interrupt OB is called.

Prerequisites for the call

Before a time-delay interrupt can be called by the operating system, the following conditions must be met:

- The time-delay interrupt OB must be started by the "SRT_DINT (Page 4706)" instruction.
- The time-delay interrupt OB must not be deselected during configuration.
- The time-delay interrupt OB must exist in the CPU.

Purpose of the instructions "SRT_DINT", "CAN_DINT" and "QRY_DINT"

You use the instructions to

- Start time-delay interrupts ("SRT_DINT (Page 4706)")
- Cancel time-delay interrupts ("CAN_DINT (Page 4708)")
- Query time-delay interrupts ("QRY_DINT (Page 4708)")

Effects on the time-delay interrupt

The following table lists a number of different situations and explains the effect they have on a time-delay interrupt.

If ...	and ...	Then ...
A time-delay interrupt is started (by calling "SRT_DINT (Page 4706)")	The time-delay interrupt has already started,	The delay time is overwritten; the time-delay interrupt is started again.
	The time-delay interrupt OB does not exist at the time of the call,	The operating system generates a priority class error (calls OB 85). If OB 85 does not exist, the CPU changes to STOP.
	The interrupt is started in a startup OB and the delay time elapses before the CPU changes to RUN,	The call of the time-delay interrupt OB is delayed until the CPU is in RUN mode.
The delay time has elapsed,	A previously started time-delay interrupt OB is still being executed,	The operating system generates a time error (calls OB 80). If OB 80 does not exist, the CPU changes to STOP.

Response to warm restart and cold restart

During a warm restart or a cold restart, all the time-delay interrupt settings made in the user program by means of instructions are cleared.

Starting in a startup OB

A time-delay interrupt can be started in a startup OB. Two conditions must be satisfied to call the time-delay OB:

- The delay time must have elapsed.
- The CPU must be in the RUN mode.

If the delay time has elapsed and the CPU is not yet in the RUN mode, the time-delay interrupt OB call is delayed until the CPU is in RUN mode. The time-delay interrupt OB is then called before the first instruction in OB Main [OB 1] is executed.

SRT_DINT: Start time-delay interrupt

Description

The instruction "SRT_DINT" is used to start a time-delay interrupt, which calls a time-delay interrupt OB after the expiry of the delay time specified at the parameter DTIME. The delay time is started when a negative edge is generated in the EN enable input. The signal state of enable input EN must be "0" during delay time countdown. If the delay time countdown is interrupted, the OB configured in the OB_NR parameter is not executed.

Accuracy

The maximum time between the call of the "SRT_DINT" instruction and the start of the time-delay interrupt OB is one millisecond longer than the configured delay time, provided that no interruption events delay the call.

Parameters

The following table shows the parameters of the instruction "SRT_DINT":

Parameter	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_DELAY (INT)	I, Q, M, D, L or constant	Number of the OB to be executed after a delay time
DTIME	Input	TIME	I, Q, M, D, L or constant	Delay time (1 to 60000 ms) You can realize longer times, for example, by using a counter in a time-delay interrupt OB.
SIGN	Input	WORD	I, Q, M, D, L or constant	Identifier that appears when the time-delay interrupt OB is called in the start event information of the OB.
RET_VAL	Return	INT	I, Q, M, D, L	Status of the instruction

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	No error
8090	Incorrect parameter OB_NR
8091	Incorrect parameter DTIME
General error information	See also: GET_ERR_ID: Get error ID locally (Page 3392)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

CAN_DINT: Cancel time-delay interrupt

Description

You use this instruction to cancel a started time-delay interrupt and, thus, also cancel the call of the time delay interrupt OB that is to be executed after the configured delay time. You specify the number of the organization block whose call is to be canceled in the OB_NR parameter.

Parameters

The following table shows the parameters of the "CAN_DINT" instruction:

Parameter	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_DELAY (INT)	I, Q, M, D, L or constant	Number of the OB whose call will be canceled
RET_VAL	Return	INT	I, Q, M, D, L	Status of the instruction

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	No error
8090	Incorrect parameter OB_NR
80A0	Time-delay interrupt has not started.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

QRY_DINT: Query time-delay interrupt status

Description

The instruction "QRY_DINT" is used to query the status of the time-delay interrupt.

Parameters

The following table shows the parameters of the instruction "QRY_DINT":

Parameter	Declaration	Data type	Memory area	Description
OB_NR	Input	OB_DELAY (INT)	I, Q, M, D, L or constant	Number of the OB whose status is being queried.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs during execution of the instruction, the actual parameter of RET_VAL contains an error code. The value "0" is displayed in the STATUS parameter.
STATUS	Output	WORD	I, Q, M, D, L	Status of the time-delay interrupt, see following table.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter STATUS

Bit	Value	Meaning
0	0	Operating system in RUN
	1	Operating system in startup
1	0	Time-delay interrupt is enabled by the operating system.
	1	Time-delay interrupt is disabled.
2	0	Time-delay interrupt is not activated or has elapsed.
	1	Time-delay interrupt is activated.
3	-	-
4	0	Time-delay interrupt OB with the specified number does not exist.
	1	Time-delay interrupt OB with the specified number exists.
Other bits		Always "0"

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	No error occurred.
8090	Incorrect information in the OB_NR parameter
General error information	See also: GET_ERR_ID: Get error ID locally (Page 3392)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

Synchronous errors

Mask synchronous error events

Introduction

Synchronous errors are programming and access errors. Such errors occur as a result of programming with incorrect operand areas or numbers, or incorrect addresses. **Masking** these synchronous errors means the following:

- Masked synchronous errors do not trigger an error OB call and do not lead to a programmed alternative reaction.
- The CPU "records" the masked errors that have occurred in an error status register.

Masking is carried out by calling the "MSK_FLT (Page 4715)" instruction.

Unmasking errors means canceling a previously set mask and clearing the corresponding bit in the event status register of the current priority class. Masking is canceled as follows:

- By calling the "DMSK_FLT (Page 4716)" instruction.
- Once the current priority class has been completed.

If an error event occurs after it has been unmasked, then the operating system will start the associated error OB.

You can use the "READ_ERR (Page 4717)" instruction to read the masked errors that have occurred.

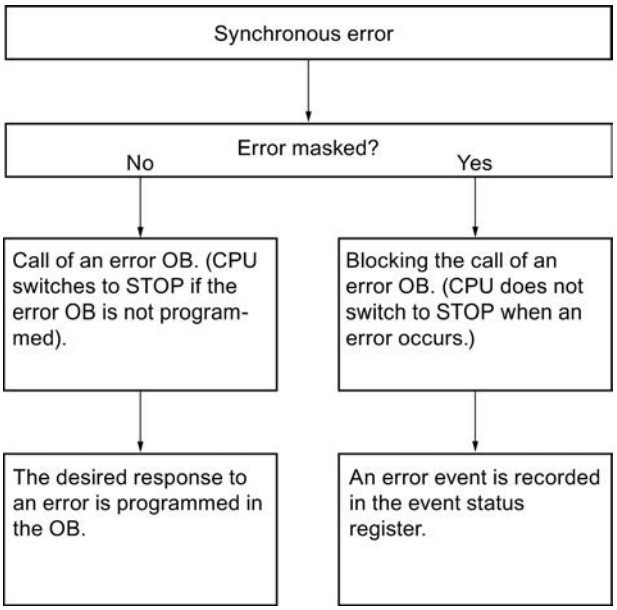
Note

Regardless of masking or unmasking of the error events for the S7-1500, the error event is entered in the diagnostic buffer and the group error LED of the CPU lights up.

Handling errors in general

Different actions are possible if programming and I/O area access errors occur in a user program:

- You can program an error OB that is called by the operating system when the corresponding error occurs.
- You can disable the error OB call individually for each priority class. In this case, the CPU does not change to STOP when an error of this type occurs in the particular priority class. The CPU enters the error in an error register. From this entry, however, you cannot recognize when or how often the error occurred.

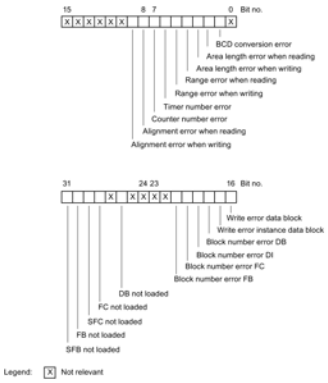


Synchronous errors are assigned to a particular bit pattern known as the **error filter**. You can find this error filter in the input and output parameters of the "MSK_FLT (Page 4715)", "DMSK_FLT (Page 4716)", and "READ_ERR (Page 4717)" instructions.

Synchronous errors are divided into **programming and access errors** which you can mask in two error masks. The error filters are illustrated in the following figures.

Programming error filter

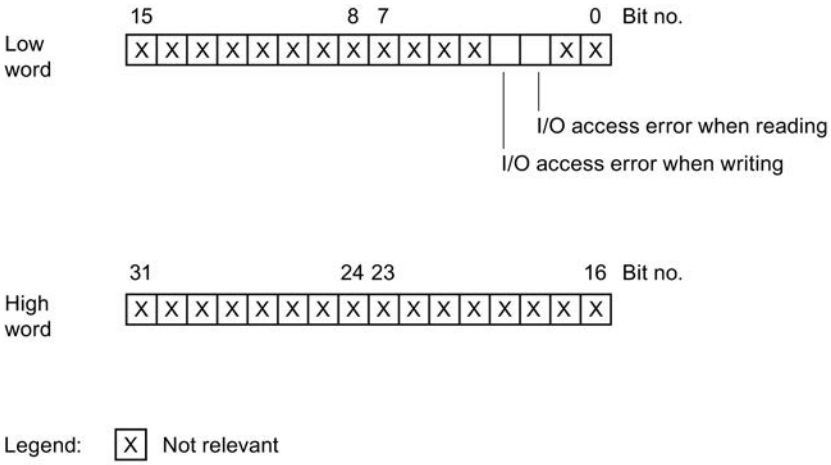
The following figure shows the bit pattern of the error filter for programming errors. The error filter for the programming errors is available in the "PRGFLT_..." parameters (see below "Programming errors, Low-Word" or "Programming errors, High-Word").



The non-relevant bits "x" are set to "0" for the input and output parameters of the instructions "MSK_FLT (Page 4715)", "DMSK_FLT (Page 4716)", "READ_ERR (Page 4717)".

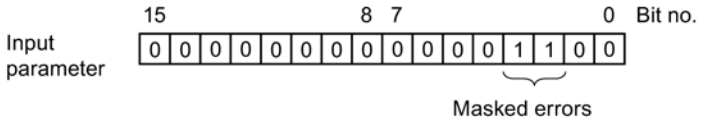
Access error mask

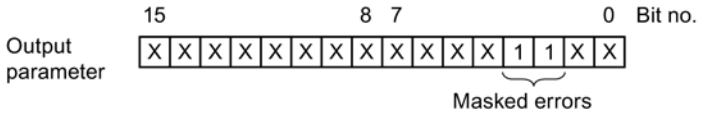
The following figure shows the bit pattern of the error mask for access errors. The error mask for access errors is located in the parameters ACCFLT_...



Example: The following figure shows how the low word of the error mask for access errors appears with all masked errors.

- As input parameters for "MSK_FLT (Page 4715)"
- As output parameters for "MSK_FLT (Page 4715)"





Legend:
- X Not relevant
- 0 Not masked
- 1 Masked

Programming error low word

The following table lists the errors assigned to the low word of the error filter for programming errors. The possible causes of the errors are also listed.

Error	Event ID (W#16#...)	Possible cause of error
BCD conversion error	2521	The value to be converted is not a BCD number (e.g. 5E8).
Area length error when reading	2522	An addressed operand is not located entirely within the possible operand range. Example: MW 320 should be read although the memory area is only 256 bytes long.
Area length error when writing	2523	An addressed operand is not located entirely within the possible operand range. Example: A value should be written to MW 320 although the memory area is only 256 bytes long.
Range error when reading	2524	An incorrect range identifier for the operand was specified during indirect, overlapping range addressing. Example: • Correct: LAR1 P#E 12.0 L W[AR1, P#0.0] • Incorrect: LAR1 P#12.0 L W[AR1, P#0.0] The range error is reported during this operation.

Error	Event ID (W#16#...)	Possible cause of error
Range error when writing	2525	An incorrect range identifier for the operand was specified during indirect, overlapping range addressing. Example: • Correct: LAR1 P#E 12.0 T W[AR1, P#0.0] • Incorrect: LAR1 P#12.0 T W[AR1, P#0.0] The range error is reported during this operation.
Timer number error	2526	A non-existent timer is accessed. Example: • SI T [MW 0] with MW 0 = 129; the timer 129 should be started although there are only 128 timers available.
Counter number error	2527	A non-existent counter is accessed. • Example: ZV Z [MW 0] with MW 0 = 600; the counter 600 is accessed although only 512 counters are available.
Alignment error when reading	2528	A byte, word or double word operand with a bit address ≠ 0 is addressed. Example: • Correct: LAR1 P#M12.0 L B[AR1, P#0.0] • Incorrect: LAR1 P#M12.4 L B[AR1, P#0.0]
Alignment error when writing	2529	A byte, word or double word operand with a bit address ≠ 0 is addressed. Example: • Correct: LAR1 P#M12.0 T B[AR1, P#0.0] • Incorrect: LAR1 P#M12.4 T B[AR1, P#0.0]

Programming error high word

The following table lists the errors assigned to the high word of the error filter for programming errors. The possible causes of the errors are also listed.

Error	Event ID (W#16#...)	Possible cause of error
Write error data block	2530	The data block to be written to is write-protected.
Write error instance data block	2531	The instance data block to be written to is write-protected.
Block number error DB	2532	The number of the data block is greater than the highest permissible number.
Block number error DI	2533	The number of the instance data block is greater than the highest permissible number.
Block number error FC	2534	The number of a called function (FC) is greater than the highest permissible number.
Block number error FB	2535	The number of a called function block (FB) is greater than the highest permissible number.

Error	Event ID (W#16#...)	Possible cause of error
DB not loaded	253A	The data block is not loaded.
Instruction not loaded	253C to 253F	The instruction to be called is not loaded.

Access error

The following table contains the errors that are assigned to the error mask for access errors. The possible causes of the errors are also listed.

Error	Event ID (W#16#...)	Possible cause of error
I/O access error when reading	2942	• The address in the I/O area has no assigned signal module. • Access to this I/O area was not acknowledged within the specified module monitoring time (timeout).
I/O access error when writing	2943	

MSK_FLT: Mask synchronous error events

Description

You use the instruction to control the reaction of the CPU to synchronous errors. This is done by masking the respective synchronous errors (for error filters, see Mask synchronous error events (Page 4710)). When you call "MSK_FLT", you mask the synchronous errors in the current priority class.

If you set individual bits of the synchronous error filter to "1" in the input parameters, other bits that were set previously retain their value of "1". You obtain new error filters that you can read out using the output parameters. The synchronous errors you have masked do not call an OB but are simply entered in an error register. You can read the error register with the "READ_ERR (Page 4717)" instruction.

Parameters

The following table shows the parameters of the instruction "MSK_FLT":

Parameter	Declaration	Data type	Memory area	Description
PRGFLT_SET_MASK	Input	DWORD	I, Q, M, D, L or constant	Programming error to be masked
ACCFLT_SET_MASK	Input	DWORD	I, Q, M, D, L or constant	Access error to be masked
RET_VAL	Return	INT	I, Q, M, D, L	Error information
PRGFLT_MASKED	Output	DWORD	I, Q, M, D, L	Masked programming errors
ACCFLT_MASKED	Output	DWORD	I, Q, M, D, L	Masked access errors

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	None of the errors were already masked.
0001	At least one of the errors was already masked. Nevertheless the other errors will be masked.
-	General error information See also: Getting error ID locally with GetErrorID (Page 4282)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Evaluating errors with output parameter RET_VAL (Page 3152)

Switching display formats in the program status (Page 2980)

DMSK_FLT: Unmask synchronous error events

Description

You use the instruction to unmask the errors masked with "MSK_FLT (Page 4715)". To do this, you must set the corresponding bits of the error filter to "1" in the input parameters. With the "DMSK_FLT" call, you unmask the corresponding synchronous errors of the current priority class. At the same time, the queried entries are cleared in the error register. You can read out the new error filters using the output parameters.

Parameters

The following table shows the parameters of the instruction "DMSK_FLT":

Parameter	Declaration	Data type	Memory area	Description
PRGFLT_RESET_MASK	Input	DWORD	I, Q, M, D, L or constant	Programming errors to be unmasked
ACCFLT_RESET_MASK	Input	DWORD	I, Q, M, D, L or constant	Access errors to be unmasked
RET_VAL	Return	INT	I, Q, M, D, L	Error information
PRGFLT_MASKED	Output	DWORD	I, Q, M, D, L	Still masked programming errors
ACCFLT_MASKED	Output	DWORD	I, Q, M, D, L	Still masked access errors

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	All specified errors were unmasked.
0001	At least one of the errors was not masked. Nevertheless the other errors will be unmasked.
-	General error information See also: GET_ERR_ID: Get error ID locally (Page 4282)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

READ_ERR: Read out event status register

Description

You use the instruction to read the error register. The structure of the error register corresponds to that of the programming and access error filters, which you can program as input parameters with "MSK_FLT (Page 4715)" and "DMSK_FLT (Page 4716)".

In the input parameters, you enter the synchronous errors you want to read from the error register. When you call "READ_ERR", you read the required entries from the error register and at the same time clear these entries.

The error register contains information that tells you which of the masked synchronous errors in the current priority class occurred at least once. If a bit is set, this means that the corresponding masked synchronous error occurred at least once.

Parameters

The following table shows the parameters of the instruction "READ_ERR":

Parameter	Declaration	Data type	Memory area	Description
PRGFLT_QUERY	Input	DWORD	I, Q, M, D, L or constant	Query programming error
ACCFLT_QUERY	Input	DWORD	I, Q, M, D, L or constant	Query access error
RET_VAL	Return	INT	I, Q, M, D, L	Error information
PRGFLT_CLR	Output	DWORD	I, Q, M, D, L	Programming errors that occurred
ACCFLT_CLR	Output	DWORD	I, Q, M, D, L	Access errors that occurred

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	All queried errors are masked.
0001	At least one of the queried errors is not masked.
-	General error information See also: Getting error ID locally with GetErrorID (Page 4282)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Evaluating errors with output parameter RET_VAL (Page 3152)

Switching display formats in the program status (Page 2980)

Asynchronous error event

DIS_IRT: Disable interrupt event

Description

The instruction "DIS_IRT" is used to disable the processing of new interrupts and asynchronous error events. Disabling means that if an interrupting event occurs, the operating system of the CPU reacts as follows:

- It **neither** calls an interrupt OB nor an asynchronous error OB,
- **nor** does it trigger the normal reaction if an interrupt OB or asynchronous error OB is not programmed.

If you disable interrupts and asynchronous error events, this remains in effect for all priority classes. The disable can only be canceled with the "EN_IRT (Page 4720)" instruction or by a warm or a cold restart.

Whether the operating system writes interrupts and asynchronous error events to the diagnostics buffer when they occur depends on the input parameter setting you select for MODE.

Note

Remember that when you program the "DIS_IRT" instruction, all interrupts that occur will be discarded.

Parameters

The following table shows the parameters of the instruction "DIS_IRT":

Parameter	Declaration	Data type	Memory area	Description
MODE	Input	BYTE	I, Q, M, D, L or constant	Specifies which interrupts and asynchronous errors are disabled.
OB_NR	Input	INT	I, Q, M, D, L or constant	OB number
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

Parameter MODE

MODE (B#16#...)	Meaning
00	All newly occurring interrupts and asynchronous error events are disabled. (Synchronous errors are not disabled.) Assign the value "0" to the OB_NR parameter. Entries continue to be made in the diagnostics buffer.
01	All newly occurring events belonging to a specified interrupt class are disabled. Identify the interrupt class by specifying it as follows: • Time-of-day interrupts: 10 • Time-delay interrupts: 20 • Cyclic interrupts: 30 • Process interrupts: 40 • Interrupts for DPV1: 50 • Multicomputing interrupt: 60 • Redundancy error interrupts: 70 • Asynchronous error interrupts: 80 Entries continue to be made in the diagnostics buffer.
02	All new occurrences of a specified interrupt are disabled. You designate the interrupt using the OB number. Entries continue to be made in the diagnostics buffer.

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	No error occurred.
8090	The OB_NR input parameter contains an invalid value.
8091	The MODE input parameter contains an invalid value.

Error code* (W#16#...)	Description
General error information	See also: GET_ERR_ID: Get error ID locally (Page 3392)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

EN_IRT: Enable interrupt event

Description

You use the instruction to enable the processing of new interrupts and asynchronous error events that you have previously disabled with the "DIS_IRT (Page 4718)" instruction. This means that if an interrupting event occurs, the operating system of the CPU reacts in one of the following ways:

- It calls an interrupt OB or asynchronous error OB.
  or
- It triggers the standard reaction if the interrupt OB or asynchronous error OB is not programmed.

Parameters

The following table shows the parameters of the instruction "EN_IRT":

Parameter	Declaration	Data type	Memory area	Description
MODE	Input	BYTE	I, Q, M, D, L or constant	Specifies which interrupts and asynchronous error events will be enabled.
OB_NR	Input	INT	I, Q, M, D, L or constant	OB number
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter MODE

MODE	Meaning
0	All newly occurring interrupts and asynchronous error events are enabled.
1	All newly occurring events belonging to a specified interrupt class are enabled. Identify the interrupt class by specifying it as follows: • Time-of-day interrupts: 10 • Time-delay interrupts: 20 • Cyclic interrupts: 30 • Process interrupts: 40 • Interrupts for DPV1: 50 • Multicomputing interrupt: 60 • Redundancy error interrupts: 70 • Asynchronous error interrupts: 80
2	All newly occurring events of a specified interrupt are enabled. You designate the interrupt using the OB number.

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	No error occurred.
8090	The OB_NR input parameter contains an invalid value.
8091	The MODE input parameter contains an invalid value.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 3392)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

DIS_AIRT: Delay execution of higher priority interrupts and asynchronous error events

Description

You use "DIS_AIRT" to delay the processing of interrupt OBs whose priority are higher than the priority of the current organization block.

You can call "DIS_AIRT" multiple times in an organization block. The "DIS_AIRT" calls are counted by the operating system. Processing is delayed more and more each time "DIS_AIRT" is executed. To cancel a delay, you must execute the "EN_AIRT (Page 4722)" instruction. To cancel all delays, the number of "EN_AIRT (Page 4722)" executions must be equal to the number of "DIS_AIRT" calls.

You can query the number of delays in the RET_VAL parameter of the "DIS_AIRT" instruction. If the value in the RET_VAL parameter is "0", there are no delays.

Parameters

The following table shows the parameters of the "DIS_AIRT" instruction:

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Return	INT	I, Q, M, D, L	Number of delays

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

EN_AIRT: Enable execution of higher priority interrupts and asynchronous error events

Description

You use "EN_AIRT" to enable processing of organization blocks when interrupts occur that have been delayed by the "DIS_AIRT (Page 4721)" instruction.

When "EN_AIRT" is executed, you cancel a processing delay that was registered by the operating system when "DIS_AIRT (Page 4721)" was called. To cancel all delays, the number of "EN_AIRT" executions must be equal to the number of "DIS_AIRT (Page 4721)" calls. If, for example, you have called "DIS_AIRT (Page 4721)" five times and thereby also delayed the processing five times, you must call the "EN_AIRT" instruction five times in order to cancel all five delays.

You can query the number of interrupt delays that have not yet been enabled after the execution of "EN_AIRT" in the RET_VAL parameter of the "EN_AIRT" instruction. The value "0" in the RET_VAL parameter means that all delays enabled by "DIS_AIRT (Page 4721)" have been cancelled.

Parameters

The following table shows the parameters of the "EN_AIRT" instruction:

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Return	INT	I, Q, M, D, L	Number of configured delays

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Alarms

Program_Alarm: Generate program alarm with associated values

Description

The instruction "Program_Alarm" monitors a signal and generates a program alarm when a signal change occurs at the SIG parameter (for definition, see also: Introduction to alarm configuration (Page 3047)). An incoming program message is generated with a signal change

from 0 to 1 and an outgoing program message is generated with a signal change from 1 to 0. The program alarm is triggered synchronously to program execution.

You can append up to ten associated values to the program alarm at the parameters SD_i (0 ≤ i ≤10). The associated values are detected at the time of signal change at parameter SIG and assigned to the program alarm. For more information on configuring associated values, refer to the following sections: Insert associated values in alarms (Page 3065), Examples of associated values (Page 3067).

Each incoming or outgoing alarm is assigned a time stamp:

- The current system time of the PLC when the signal change occurs is used by default (default value at the TIMESTAMP parameter).
- If you wish to specify a different time stamp, you can create one at the TIMESTAMP parameter.
  The time value must always be specified in system time (i.e. UTC) as this is the time used for time synchronization throughout the plant.
- If an alarm is to be time-stamped with a local time, a conversion module, which converts the local time to system time, must be connected in series. This is the only way to make sure that the time stamp is shown properly in the alarm display.

To use the current system time of the CPU, set the TIMESTAMP parameter to its default value (LDT#1970-1-1-0:0:0.0).

Calling the "Program_Alarm" instruction

The instruction "Program_Alarm" can only be called in a function block (FB).

Once the instruction has been inserted in the FB, a multi-instance of the data type "Program_Alarm" is created in the "Static" section of the FB interfaces. You can choose any name for this multi-instance in the dialog which appears. The name of the multi-instance is also the name of the program alarm.

Finally, add the parameters of the "Program_Alarm" instruction in line with your requirements (see the "Parameters" table).

Configuring the program alarm

The program alarm settings are displayed in the "Properties" window when you select the name of the program alarm in the "Static" section or in the network of the FB. In this window, you can select the alarm class, priority, etc. and edit the alarm text.

The settings that are made here can be edited in the project tree. Go to "PLC alarms" and open the "Program alarms" tab. All existing program alarms are then displayed in the "Alarm types" table.

Parameters

The following table shows the parameters of the "Program_Alarm" instruction:

Parameter	Declaration	Data type	Memory area	Description
SIG	Input	BOOL	I, Q, M, D, L, T, C or constant	The signal to be monitored.
TIMESTAMP	Input	LDT	M, D, L or constant	This parameter is used to assign an alarm a time stamp, for example, from an input signal with a distributed stamp. The time value must always be specified in system time (i.e. UTC), as this is the time used for time synchronization throughout the plant. • "Not assigned" means that the system time of the CPU will be used as the interrupt time stamp when a signal change occurs (default). • Any system time input will be used as the interrupt time stamp when a signal change occurs. Note: If an interrupt is to be time-stamped with a local time, a conversion module must be connected in series to convert the local time to system time. This is the only way to make sure that the time stamp is shown correctly in the interrupt display.
SD_i	Input	LREAL, REAL, DINT, INT, SINT, UDINT, UINT, USINT, DWORD, WORD, BYTE, BOOL, WSTRING, STRING, WCHAR, CHAR	I, Q, M, D, L	i-th associated value (0 ≤ i ≤ 10)
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR ERROR = TRUE indicates that an error has occurred during processing. The possible error cause is displayed at the STATUS parameter.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter STATUS Display of error information (see "ERROR and STATUS parameters").

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

ERROR and STATUS parameters

The following table contains all specific error information that can be output via the ERROR and STATUS parameters.

ERROR	STATUS*	Explanation
0	0000	No error
1	0085	"Only information" alarm type.
1	8001	Invalid static alarm information.
1	8002	No valid static alarm information.
1	8004	Maximum size of 512 bytes reached for the associated values of the alarm.
1	8005	Incoming alarm.
1	8007	Outgoing alarm, preceded by no incoming alarm.
1	8087	Deactivate static alarms.
1	8089	Length of the alarm is too large.
1	80Ax	The SD_i parameter has an invalid value.
1	80C1	The CPU cannot generate any alarms at this time because initialization routines are running (this is the case after download in RUN, for example). Try again later.
1	80C2	The maximum number of alarms per time unit has been sent. Try again later.
1	80C3	All dynamic alarm instances are being used. Try again later.
1	80C4	Alarm is being output and cannot be overwritten. Try again later.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

See also

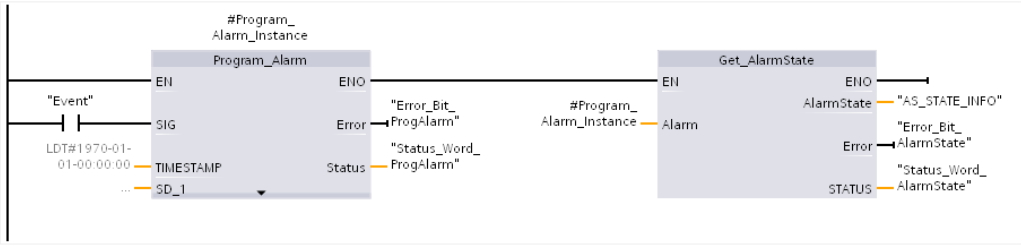
Switching display formats in the program status (Page 2980)

Get_AlarmState: Output alarm status

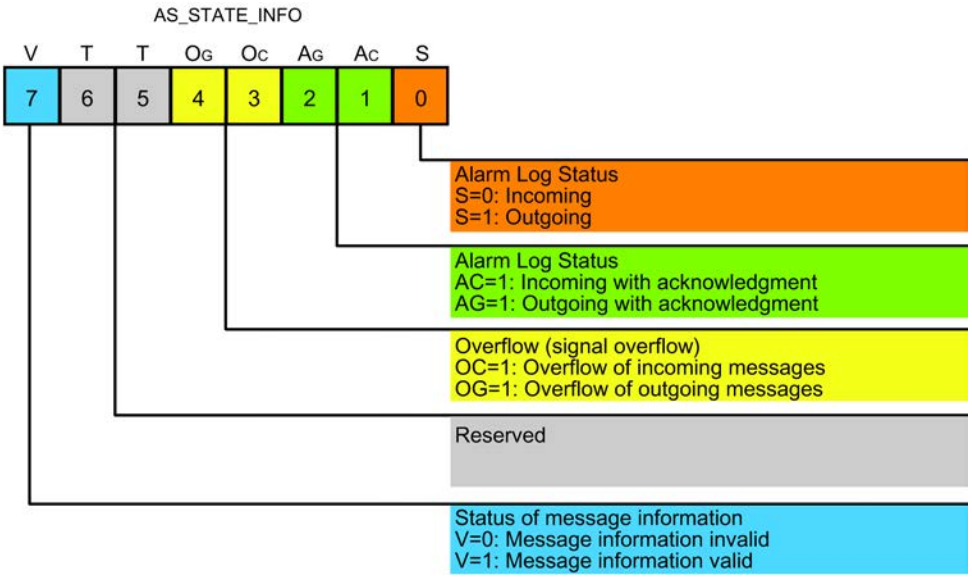
Description

The "Get_AlarmState" instruction outputs the alarm status of a program alarm. The output of the alarm status always refers to a program alarm that was created using the "Program_Alarm (Page 4722)" instruction.

The program alarm is selected with the Alarm input parameter. Specify the instance DB of the "Program_Alarm" instruction at the Alarm parameter.



The alarm status is returned in a byte via the AlarmState output parameter. The meaning of the individual bits is shown in the following figure:



You can find more information about the possible alarm states in the section "Creating and editing alarms (Page 3062)".

The execution status of the instruction is shown with the output parameters ERROR and STATUS.

Parameters

The following table shows the parameters of the "Get_AlarmState" instruction:

Parameter	Declaration	Data type	Memory area	Description
Alarm	Input	ALARM_BASE	D	Instance of the "Program_Alarm (Page 4722)" instruction.
AlarmState	Output	BYTE	I, Q, M, D, L	Status of the alarm as a bit array.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter • 0: No error • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter The parameter is only set for the duration of one call. To display the status, you should therefore copy STATUS to a free data area.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter STATUS

Error code* (W#16#...)	Explanation
8001	Invalid static alarm instance.
8002	ID of the alarm is invalid.
8003	No active alarms within the message class.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

Gen_UsrMsg: Generate user diagnostic alarms

Description

You can use the "Gen_UsrMsg" instruction to create a message for entry into the diagnostic buffer.

You can use the Mode parameter to select whether an incoming or an outgoing message is to be generated:

- With Mode = 0: Create an incoming message.
- With Mode = 1: Create an outgoing message.
- Regardless of whether an incoming or outgoing message is generated, the message always has the attribute "Information only".

The content of the message is defined with a text list:

- Specify the text list you want to use with the parameter TextListID. For this purpose open the dialog "Text lists" in the project navigation. Show the column "ID" in the dialog "Text lists". Apply the ID at the parameter TextListID.
- Use the parameter TextID to select the text list entry you want to write in the diagnostic buffer. For this purpose select an entry from the "Text lists entries" dialog by applying a number from the columns "Range from / range to" at the parameter TextID. The same number must be used from both the "Range from" and "Range to" columns for the text list entry.
- Additional information about text lists can be found here: Text lists (Page 409)

New associated values can be defined in the text entry and then added to the message:

- Associated values are defined by adding the following information in the text list entry: @<Nr. des Begleitwertes><Formatangabe>@
- Use the system data type AssocValues to specify which associated value is added when generating the message.
- Additional information on the structure of associated values can be found here: Structure of associated values (Page 3059)

The entry in the diagnostic buffer is created synchronously. Message transmission is asynchronous.

If an error occurs during the execution of an instruction, it is output via the parameter RET_VAL.

Parameters

The following table shows the parameters of the "Gen_UsrMsg" instruction:

Parameter	Declaration	Data type	Memory area	Description
Mode	Input	UInt	I, Q, M, D, L or constant	Parameters for selecting the status of the message: • 1: incoming message • 2: outgoing message
TextID	Input	UInt	I, Q, M, D, L or constant	ID of the text list entry that should be used for the message text.
TextListID	Input	UInt	I, Q, M, D, L or constant	ID of the text list that contains text list entry.
Ret_Val	Return	Int	I, Q, M, D, L	Error code of the instruction.
AssocValues	InOut	VARIANT	I, Q, M, D, L	Pointer to the system data type AssocValues that allows you to define the associated values .

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

AssocValues parameter

Use the system data type AssocValues to define which associated values will be sent. A maximum of 8 associated values are possible. The structure of the system data type is created by entering the data type "AssocValues" as a data block.

The associated values are selected by entering the numbers of the associated values for the parameters Value[x]. The associated value must be used in the selected text list entry (parameter TextID). Note the following:

- The values for TextID and TextListID are treated by the instruction as the associated values to be sent. This means that the digits "1" and "2" can not be used as a numbers for associated values.
- Therefore, use "3" as the first number at the parameter Value [0] and "4" at the parameter Value [1] etc.

Byte	Parameters	Data type	Start value	Description
0..1	Value[1]	UINT	0	First associated value of the message.
2..3	Value[2]	UINT	0	Second associated value of the message.
4..5	Value[3]	UINT	0	...
6..7	Value[4]	UINT	0	...

Byte	Parameters	Data type	Start value	Description
8..9	Value[5]	UINT	0	...
10..11	Value[6]	UINT	0	...
12..13	Value[7]	UINT	0	...
14..15	Value[8]	UINT	0	Eighth associated value of the message.

RET_VAL parameter

The following table contains all specific error information that can be output via the RET_VAL parameter.

Error code* (W#16#...)	Explanation
0000	No error
8080	Value in the Mode parameter is not supported.
80C1	Resource bottleneck due to too many parallel calls.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

Creating and editing alarms (Page 3062)

Diagnostics

RD_SINFO: Read current OB start information

Description

You use the instruction "RD_SINFO" to read the start information

- of the last OB called that has not yet been completely executed, and
- of the last startup OB started.

There is no time stamp in either case. If the call is in OB 100, OB 101 or OB 102, two identical start information messages will be returned.

Parameter

The following table shows the parameters of the "RD_SINFO" instruction:

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Return	INT	I, Q, M, D, L	Error information
TOP_SI	Output	VARIANT	D, L	Start information of the current OB
START_UP_SI	Output	VARIANT	D, L	Start information of the startup OB last started

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

SDTs of the TOP_SI parameter

The following table shows the possible SDTs for the TOP_SI parameter:

Organization blocks (OB)	System data types (SDT)	System data type numbers
Any	SI_classic	592
	SI_none	593
ProgramCycleOB	SI_ProgramCycle	594
TimeOfDayOB	SI_TimeOfDay	595
TimeDelayOB	SI_Delay	596
CyclicOB	SI_Cyclic	597
ProcessEventOB	SI_HWInterrupt	598
ProfileEventOB StatusEventOB UpdateEventOB	SI_Submodule	601
SynchronousCycleOB	SI_SynchCycle	602
IOredundancyErrorOB	SI_IORedundancyError	604
CPUredundancyErrorOB	SI_CPURedundancyError	605
TimeErrorOB	SI_TimeError	606
DiagnosticErrorOB	SI_DiagnosticInterrupt	607
PullPlugEventOB	SI_PlugPullModule	608
PeripheralAccessErrorOB	SI_AccessError	609
RackStationFailureOB	SI_StationFailure	610
ServoOB	SI_Servo	611
IpoOB	SI_Ipo	612
StartupOB	SI_Startup	613
ProgrammingErrorOB IOaccessErrorOB	SI_ProgIOAccessError	614

SDTs of the START_UP_SI parameter

The following table shows the possible SDTs for the START_UP_SI parameter:

System data types (SDT)	System data type numbers
SI_classic	592
SI_none	593
SI_Startup	613

Structures

The following tables set out the meaning of the structure elements of the individual structures:

Table 9-61 SI_classic structure

Structure element	Data type	Description
EV_CLASS	BYTE	• Bits 0 to 3: Event ID • Bits 4 to 7: Event class
EV_NUM	BYTE	Event number
PRIORITY	BYTE	Priority class number (Meaning of B#16#FE: OB not available or disabled or cannot be started in current operating mode)
NUM	BYTE	OB number
TYP2_3	BYTE	Data ID 2_3: Identifies the information entered in ZI2_3
TYP1	BYTE	Data ID 1: Identifies the information entered in ZI1
ZI1	WORD	Additional information 1
ZI2_3	DWORD	Additional information 2_3

Table 9-62 SI_none structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)

Table 9-63 SI_ProgramCycle structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 1	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
Initial_Call	BOOL	For OB_Class = 1, 30, 52, 61, 65
Remanence	BOOL	For OB_Class = 1

Table 9-64 SI_TimeOfDay structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 10	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
CaughtUp	BOOL	For OB_Class = 10
SecondTime	BOOL	For OB_Class = 10

Table 9-65 SI_Delay structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 20	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
Sign	WORD	For OB_Class = 20

Table 9-66 SI_Cyclic structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 30	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
Initial_Call	BOOL	For OB_Class = 1, 30, 52, 61, 65
Event_Count	INT	For OB_Class = 30, 51, 52, 61, 65, 91, 92

Table 9-67 SI_HWInterrupt structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 40	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
LADDR	HW_IO	For OB_Class = 40, 51, 55, 56, 57, 70, 82, 83, 85, 86, 91, 92
USI	WORD	For OB_Class = 40
IChannel	USINT	For OB_Class = 40
EventType	BYTE	For OB_Class = 40

Table 9-68 SI_Submodule structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
LADDR	HW_IO	For OB_Class = 40, 51, 55, 56, 57, 70, 82, 83, 85, 86, 91, 92
Slot	UINT	For OB_Class = 55, 56, 57
Specifier	WORD	For OB_Class = 55, 56, 57

Table 9-69 SI_SynchCycle structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 61	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
Initial_Call	BOOL	For OB_Class = 1, 30, 52, 61, 65
PIP_Input	BOOL	For OB_Class = 61, 91, 92
PIP_Output	BOOL	For OB_Class = 61, 91, 92
IO_System	USINT	For OB_Class = 61, 91, 92
Event_Count	INT	For OB_Class = 30, 51, 52, 61, 65, 91, 92
SyncCycleTime	LTIME	Calculated cycle time

Table 9-70 SI_IORedundancyError structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 70	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
LADDR	HW_ANY	For OB_Class = 40, 51, 55, 56, 57, 70, 82, 83, 85, 86, 91, 92
Event_Class	BYTE	For OB_Class = 70, 83, 85, 86
Fault_ID	BYTE	For OB_Class = 70, 80, 83, 85, 86

Table 9-71 SI_CPURedundancyError structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 72	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
Switch_Over	BOOL	For OB_Class = 72

Table 9-72 SI_TimeError structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 80	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
Fault_ID	BYTE	For OB_Class = 70, 80, 83, 85, 86
Csg_OBnr	OB_ANY	For OB_Class = 80
Csg_Prio	UINT	For OB_Class = 80

Table 9-73 SI_DiagnosticInterrupt structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 82	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
IO_State	WORD	For OB_Class = 82
LADDR	HW_ANY	For OB_Class = 40, 51, 55, 56, 57, 70, 82, 83, 85, 86, 91, 92
Channel	UINT	For OB_Class = 82
MultiError	BOOL	For OB_Class = 82

Table 9-74 SI_PlugPullModule structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 83	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
LADDR	HW_IO	For OB_Class = 40, 51, 55, 56, 57, 70, 82, 83, 85, 86, 91, 92
Event_Class	BYTE	For OB_Class = 70, 83, 85, 86
Fault_ID	BYTE	For OB_Class = 70, 80, 83, 85, 86

Table 9-75 SI_AccessError structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 85	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
LADDR	HW_IO	For OB_Class = 40, 51, 55, 56, 57, 70, 82, 83, 85, 86, 91, 92
Event_Class	BYTE	For OB_Class = 70, 83, 85, 86
Fault_ID	BYTE	For OB_Class = 70, 80, 83, 85, 86
IO_Addr	UINT	For OB_Class = 85
IO_LEN	UINT	For OB_Class = 85

Table 9-76 SI_StationFailure structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 86	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
LADDR	HW_IO	For OB_Class = 40, 51, 55, 56, 57, 70, 82, 83, 85, 86, 91, 92
Event_Class	BYTE	For OB_Class = 70, 83, 85, 86
Fault_ID	BYTE	For OB_Class = 70, 80, 83, 85, 86

Table 9-77 SI_Servo structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 91	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
Initial_Call	BOOL	For OB_Class = 1, 30, 52, 61, 65
PIP_Input	BOOL	For OB_Class = 61, 91, 92
PIP_Output	BOOL	For OB_Class = 61, 91, 92
IO_System	USINT	For OB_Class = 61, 91, 92
Event_Count	INT	For OB_Class = 30, 51, 52, 61, 65, 91, 92
Synchronous	BOOL	

Table 9-78 SI_Ipo structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 92	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
Initial_Call	BOOL	For OB_Class = 1, 30, 52, 61, 65
PIP_Input	BOOL	For OB_Class = 61, 91, 92
PIP_Output	BOOL	For OB_Class = 61, 91, 92
IO_System	USINT	For OB_Class = 61, 91, 92
Event_Count	INT	For OB_Class = 30, 51, 52, 61, 65, 91, 92
Reduction	UINT	For OB_Class = 92

Table 9-79 SI_Startup structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT := 100	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)

Structure element	Data type	Description
LostRetentive	BOOL	For OB_Class = 100
LostRTC	BOOL	For OB_Class = 100

Table 9-80 SI_ProgIOAcessError structure

Structure element	Data type	Description
SI_Format	USINT	• 16#FF = No information • 16#FE = Optimized start information
OB_Class	USINT	OB class for "No information" or "Optimized start information"
OB_Nr	UINT	OB number (1 ... 32767)
BlockNr	UINT	For OB_Class = 121, 122
Reaction	USINT	For OB_Class = 121, 122
Fault_ID	BYTE	For OB_Class = 121, 122
BlockType	USINT	For OB_Class = 121, 122
Area	USINT	For OB_Class = 121, 122
DBNr	DB_ANY	For OB_Class = 121, 122
Csg_OBNr	OB_ANY	For OB_Class = 121, 122
Csg_Prio	USINT	For OB_Class = 121, 122
Width	USINT	For OB_Class = 121, 122

Note

The structure elements specified for the SI_classic structure are identical in content to the temporary tags of any OB created with the block property "Default".

Please note, however, that temporary tags of the individual OBs can have different names and different data types. Also note that the call interface of each OB includes additional information regarding the date and the time of the OB request.

Bits 4 to 7 of the EV_CLASS structure element contain the event class. The following values are possible here:

- 1: Start events from standard OBs
- 2: Start events from synchronous error OBs
- 3: Start events from asynchronous error OBs

The PRIORITY structure element supplies the priority class belonging to the current OB.

Apart from these two elements, NUM is also relevant. NUM contains the number of the current OB or the startup OB that was started last.

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
8080	Start information of the current OB does not correspond to the specified user-defined data type
8081	Start information of the current OB does not correspond to the specified system data type
8082	Start information of the last startup OB started does not correspond to the specified user-defined data type
8083	Start information of the last startup OB started does not correspond to the specified system data type

Example

OB 80 is the OB that was called last and that has not yet been completely processed and OB 100 is the start-up OB that was started last.

The following table shows the assignment between the structure elements of the TOP_SI parameter of the "RD_SINFO" instruction and the associated local tags of OB 80.

TOP_SI structure element	Data type	OB 80 - Associated local tag	Data type
EV_CLASS	BYTE	OB80_EV_CLASS	BYTE
EV_NUM	BYTE	OB80_FLT_ID	BYTE
PRIORITY	BYTE	OB80_PRIORITY	BYTE
NUM	BYTE	OB80_OB_NUMBR	BYTE
TYP2_3	BYTE	OB80_RESERVED_1	BYTE
TYP1	BYTE	OB80_RESERVED_2	BYTE
ZI1	WORD	OB80_ERROR_INFO	WORD
ZI2_3	DWORD	OB80_ERR_EV_CLASS	BYTE
		OB80_ERR_EV_NUM	BYTE
		OB80_OB_PRIORITY	BYTE
		OB80_OB_NUM	BYTE

The following table shows the assignment between the structure elements of the START_UP_SI parameter of the "RD_SINFO" instruction and the associated local tags of OB 100.

START_UP_SI structure element	Data type	OB 100 - Local tag	Data type
EV_CLASS	BYTE	OB100_EV_CLASS	BYTE
EV_NUM	BYTE	OB100_STRTUP	BYTE
PRIORITY	BYTE	OB100_PRIORITY	BYTE
NUM	BYTE	OB100_OB_NUMBR	BYTE
TYP2_3	BYTE	OB100_RESERVED_1	BYTE
TYP1	BYTE	OB100_RESERVED_2	BYTE
ZI1	WORD	OB100_STOP	WORD
ZI2_3	DWORD	OB100_STRT_INFO	DWORD

See also

Evaluating errors with output parameter RET_VAL (Page 3152)

RT_INFO: Read out runtime statistics

Description

Use the instruction "RT_INFO" to generate statistics on the runtime of specific organization blocks, communication or the user program.

Use the MODE parameter to select which information is going to be output:

- MODE 1 to 3 return the data on the runtime of a specific organization block whose number was specified at the OB parameter.
- MODE 10 and 11 return the proportions of runtime used for communication and the user program.
- MODE 30 to 32 return data on configured settings of the user program.

All measurements are started once again when the CPU transitions from startup to RUN.

Parameters

The following table shows the parameters of the instruction "RT_INFO":

Parameter	Declaration	Data type	Memory area	Description
MODE	Input	UINT	I, Q, M, D, L or constant	Use the MODE parameter to select the information you want to read out (see table "MODE parameter").
OB	Input	OB_ANY	I, Q, M, D, L or constant	Use the OB parameter to select the OB for which you want to read out information.
RET_VAL	Return	INT	I, Q, M, D, L	Error information (see " RET_VAL parameter").
INFO	InOut	VARIANT	I, Q, M, D, L	Pointer to the area to which the read data is to be written. The data type required for INFO depends on the MODE parameter (see table "MODE parameter").

You can find additional information on valid data types in "Overview of the valid data types (Page 2236)".

MODE parameter

The following table shows which information is returned for the values at the MODE parameter.

MODE (decimal)	Description	Note	Value at OB parameter	Data type of INFO	Available as of CPU version
1	Runtime of a specific OB	The runtime information always refers to the specified OB. Processes, such as interruptions by OBs of a higher priority or communication, are not included. You can use the instruction "RUNTIME" to measure the runtime of the entire program.	OB number	LTIME	S7-1500 V1.5
2	Maximum runtime of a specific OB		OB number	LTIME	S7-1500 V1.5
3	Minimum runtime of a specific OB		OB number	LTIME	S7-1500 V1.5
10	Percentage of total runtime used by higher-priority OBs The runtime of all OBs used in the user program with a higher priority than cyclical program OBs is output (ProgramCycle). These usually include all OB types except startup OBs. The CPU determines which OBs can be used and their priority. This information is included in the basic chapters on programming.	The percentages of MODE 10, 11 and 12 always add up to 100 (=total runtime).	Not relevant	UINT	S7-1500 V1.5
11	Percentage of total runtime used by communication The percentage of communication processes in the total runtime of the user program is output.		Not relevant	UINT	S7-1500 V1.5
30	Output of the warning limit from which a time error OB is called. The warning limit is the maximum permitted number of requested but not yet completely processed requirements of an OB.		Not relevant	LTIME	S7-1500 V1.5
31	Output of the configured minimum cycle time for the user program		Not relevant	UINT	S7-1500 V1.5
32	Output of the configured maximum communication load		Not relevant	UINT	S7-1500 V1.5

RET_VAL parameter

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0	No error
8080	Selection for MODE parameter is not supported.
8081	Organization block selected at OB parameter does not exist in the user program.
80C3	Insufficient resources. Try to call the instruction again at a later time.
8452	Data type at INFO parameter is incorrect. Check whether the correct data type was selected depending on the MODE parameter.

See also

Evaluating errors with output parameter RET_VAL (Page 3152)

LED: Read LED status

Description

You can use the "LED" instruction to read out the status (e.g., "On" or "Off") of a particular module LED.

- With the LADDR parameter, you address the CPU or the interface.
- With the LED parameter, you select the module LED whose current status is to be read out using the instruction.
- The RET_VAL parameter outputs the status of the selected LED when the instruction is called. Depending on the LED selected, only certain status information may be displayed. For example, some LEDs have only color information. Refer to the hardware documentation of the respective module for the available status information of a particular LED.

Parameters

The following table shows the parameters of the "LED" instruction:

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_IO	I, Q, M, L or constant	Identification number of the CPU or interface. The number is automatically assigned and stored in the CPU properties or the interface in the hardware configuration.
LED	Input	UINT	I, Q, M, D, L or constant	Identification number of the LED: • 1: STOP/RUN • 2: ERROR • 3: MAINT (maintenance) • 4: Redundant • 5: Link (green) • 6: Rx/Tx (yellow)
RET_VAL	Return	INT	I, Q, M, D, L	Status of LED

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

RET_VAL	Description
0 to 9	LED status: • 0 = LED does not exist • 1 = Permanently switched off • 2 = Color 1 (e.g., for LED STOP/RUN: green) permanently ON • 3 = Color 2 (e.g., for LED STOP/RUN: orange) permanently ON • 4 = Color 1 flashing at 2 Hz • 5 = Color 2 flashing at 2 Hz • 6 = Colors 1 and 2 flashing alternately at 2 Hz • 7 = LED is active, color 1 • 8 = LED is active, color 2 • 9 = LED exists, but status information not available
8091	Module addressed with the LADDR parameter does not exist.
8092	A module that does not support LEDs was addressed with the LADDR parameter.
8093	The identification number specified in the LED parameter is not defined.
80Bx	The CPU specified in the LADDR parameter does not support the "LED" instruction.

Get_IM_Data: Reading identification and maintenance data

Description

The "Get_IM_Data" instruction reads the identification and maintenance data (I&M) data from a device. Use the LADDR parameter to select the device from which you want to read the I&M via the hardware identifier.

You select the data to be read via the instruction using the IM_TYPE parameter:

- IM_TYPE = 0: I&M 0 data
  The I&M 0 data are the device-specific basic information of a device and contain information such as the manufacturer ID, order number, serial number and the hardware and firmware version. Only read access is possible to the I&M 0 data. The information is also displayed in the TIA Portal with the "Online & Diagnostics" view of a device.
- IM_TYPE = 11: I&M 1 data from the parameter assignment data of the CPU
  The I&M 1 data contains a functional description of the device and the location ID, i.e. the information about how the device is designated within the plant.
- IM_TYPE = 12: I&M 2 data from the parameter assignment data of the CPU
  The I&M 2 data includes the date of installation, i.e. the information about when the device was installed in the plant.
- IM_TYPE = 13: I&M 3 data from the parameter assignment data of the CPU
  The I&M 3 data contains additional information about the installed device. The additional information is free text and can be assigned as desired.

The I&M data read is written to the addressed area defined at the DATA parameter.

The execution status of the read job is displayed via the BUSY, DONE, ERROR output parameters and the two middle bytes of the STATUS output parameter.

Definition: Identification and maintenance data (I&M)

Identification and maintenance (I&M) data refers to information stored in a module, which assists you in checking the plant configuration, locating hardware changes in a plant and eliminating errors.

- Identification data (I data) are static information about a device that can only be read.
- Maintenance data (M-data) refers to plant-dependent information, such as the installation location or date. Maintenance data are created during configuration and written to the module.

Parameters

The following table shows the parameters of the "Get_IM_Data" instruction:

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_IO	I, Q, M, D, L or constant	Hardware ID of the device The number is assigned automatically and is stored in the properties of the device or in the hardware configuration.
IM_TYPE	Input	UINT	I, Q, M, D, L or constant	Identification and maintenance data number Possible values: • 0: I&M 0 data • 11: I&M 1 data from the parameter assignment data of the CPU • 12: I&M 2 data from the parameter assignment data of the CPU • 13: I&M 3 data from the parameter assignment data of the CPU
DATA	InOut	VARIANT	I, Q, M, D, L	Area for storing the read identification and maintenance data (see DATA parameter).
DONE	Output	BOOL	I, Q, M, D, L	The instruction was executed successfully. The I&M data were transferred to the DATA parameter.
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Execution of the instruction complete or not yet started. • 1: Execution of the instruction not yet complete.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter • 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter The parameter is only set for the duration of one call. To display it, you should therefore copy STATUS to a free data area.

You can find more information on the data types in "Overview of the valid data types (Page 2236)".

DATA parameter

You can use a string (STRING), an array (ARRAY of CHAR/BYTE) or a data structure (STRUCT) to store the I&M data:

- If you address a string (STRING data type) at the the DATA parameter, the length of the string automatically adjusts the length of the read I&M data (up to 254 characters).
- If you address a data structure (ARRAY of CHAR/BYTE or STRUCT) a the DATA parameter, the I&M data read are written to the individual components of the data type used. If the addressed data structure is longer than the read data, the remaining components are filled with zeros.
- If you crate a data structure with a STRUCT data type at the DATA parameter, use a data block without optimized block access (see category "Attributes" of the block properties).

If a data type other than STRING, ARRAY of BYTE/CHAR or STRUCT is used at the DATA parameter, the STATUS parameter generates error code 8093. There is an exception for I&M 0 data (IM_TYPE parameter = 0). For I&M 0 data, the following "IuM0_data" structure can also be used at the DATA parameter:

Parameter		Data type	Bytes	Description
Manufacturer_ID		UINT	2	Manufacturer ID (e.g. "42" for SIEMENS)
Order_ID		CHAR[20]	20	Order number
Serial_Number		CHAR[16]	16	Serial number
Hardware_Revision		UNIT	2	Hardware revision
Software_Revision		STRUCT	4	Firmware revision
	Type	CHAR	1	-
	Functional	UNIT8	1	-
	Bugfix	UNIT8	1	-
	Internal	UINT8	1	-
Revision_Counter		UINT	2	Revision counter
Profile_ID		UNIT	2	Profile
Profile_Specific_Type		UNIT	2	Device class
IM_Version		UNIT	2	I&M version
I&M_Supported		UNIT	2	I&M data (I&M 0-I&M 4) supported at the device end

STATUS parameter

Error code* (W#16#...)	Description
0	No error
7000	No job in progress.
7001	First call of the asynchronous instruction "Get_IM_Data". Execution of the instruction not yet complete (BUSY = 1, DONE = 0).
7002	Additional call of the asynchronous instruction "Get_IM_Data". Execution of the instruction not yet complete (BUSY = 1, DONE = 0).

Error code* (W#16#...)	Description
807F	Maximum of 10 parallel executions of the instances exceeded: • Avoid more than 10 parallel calls of the instruction. • Repeat the call at a later time, as this is a temporary error.
8091	The device addressed at the LADDR parameter does not exist.
8092	LADDR addresses a device that does not support the output of the I&M data.
8093	Data type at the DATA parameter is not supported.
80B1	The "Get_IM_Data" instruction is not supported by the CPU used.
80B2	Invalid value at the IM_TYPE parameter or the selected IM_TYPE is not supported by the CPU or the addressed device.
8752	The memory area specified at the DATA parameter is too small to store all the I&M data. The read I&M data are stored only up to the maximum length of the specified memory area.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

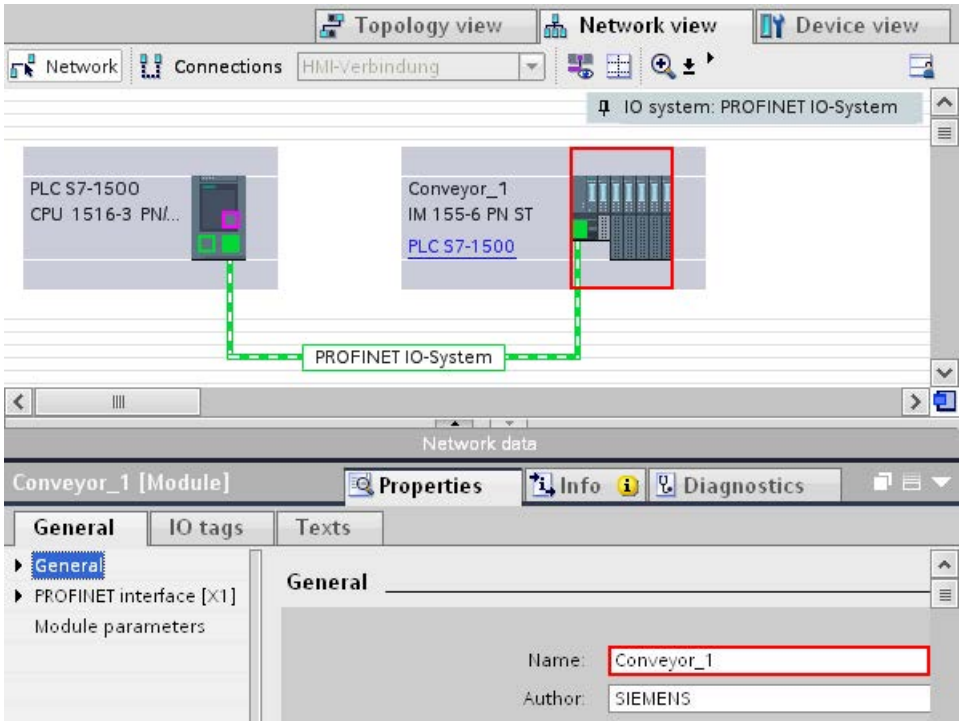
See also

Switching display formats in the program status (Page 2980)

GET_NAME: Reading the name of a module

Description

The instruction "GET_NAME" reads the name of an IO device. The name is displayed in the network view and in the properties of the IO device:



You select the IO device by using the hardware identifier of the PROFINET IO system (at the LADDR parameter) and the device number of the IO device (STATION_NR parameter).

Once the instruction has been executed, the name of the IO device is written to the area addressed with the DATA parameter.

The name that is read depends on the type of IO device:

- For a DP slave or IO device, the name of the head module is output.
- For an I-slave or I-device, the name of the interface module is output.
- The station name is output for an HMI panel.
- The name of the interface module is output for a PC station.
- The name of the Device Access Point (DAP) is displayed (name of the interface or head module) for GSD devices.

The length of the name is output at the LEN parameter. If the name is longer than the area specified at the DATA parameter, only that section which corresponds to the maximum length of the addressed area will be written.

The maximum length for a name is 128 characters.

Note

Name of the CPU readout (Version 1.1)

If you assign a "0" at each of the parameters LADDR and STATION_NR the instruction will output the name of the CPU.

Parameters

The following table shows the parameters of the "GET_NAME" instruction:

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_IOSYSTEM	I, Q, M, D, L or constant	Hardware identifier of the PROFINET IO system. The number can be taken from the system constants or the properties of the IO system.
STATION_NR	Input	UINT	I, Q, M, D, L or constant	Device number of the IO device. The device number can be applied in the Network view from the properties of the IO device under "Ethernet addresses".
DATA	InOut	VARIANT	I, Q, M, D, L	Pointer to the area to which the name of the module is written.
DONE	Output	BOOL	I, Q, M, D, L	The instruction was executed successfully. Name of the module transferred to the area at the DATA parameter.
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Execution of the instruction complete. • 1: Execution of the instruction not yet complete.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter • 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.
LEN	Output	DINT	I, Q, M, D, L	Length of the name of the IO device (number of characters).
STATUS	Output	WORD	I, Q, M, D, L	Status parameter The parameter is only set for the duration of one call. To display the status, you should therefore copy STATUS to a free data area.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter STATUS

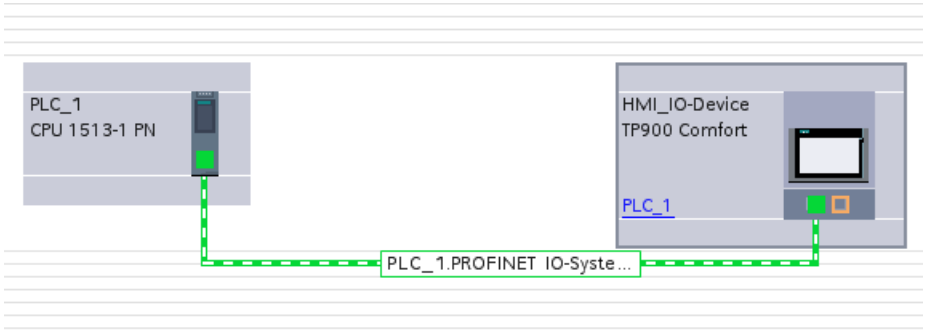
Error code* (W#16#...)	Explanation
0	No error
7000	No job in progress.
7001	First call of the asynchronous instruction "GET_NAME". Execution of the instruction not yet complete (BUSY = 1, DONE = 0).
7002	Additional call of the asynchronous instruction "GET_NAME". Execution of the instruction not yet complete (BUSY = 1, DONE = 0).
8090	The LADDR input parameter contains an invalid value. Possible causes: • The specified number is not a hardware identifier of a PROFINET IO system. • You have not specified a number at the LADDR parameter.
8092	The value at the LADDR parameter does not address a PROFINET IO system.
8093	Instruction does not support data type at the DATA parameter.
8095	Device number (STATION_NR parameter) does not exist in the selected PROFINET IO system.
80B1	The CPU used does not support the instruction.
80C3	Temporary resource error: The CPU is currently processing the maximum possible number of simultaneous block calls. "GET_NAME" cannot be executed unless at least one of the block calls is finished.
8852	The area specified at the DATA parameter is too short for the full name of the IO device. The name will be written up to the maximum possible length. To read the full name, use a longer data area at the DATA parameter. The area must have at least as many characters as there are at the LEN parameter.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

Example

The following example shows how you can read out the station name of a HMI panel.

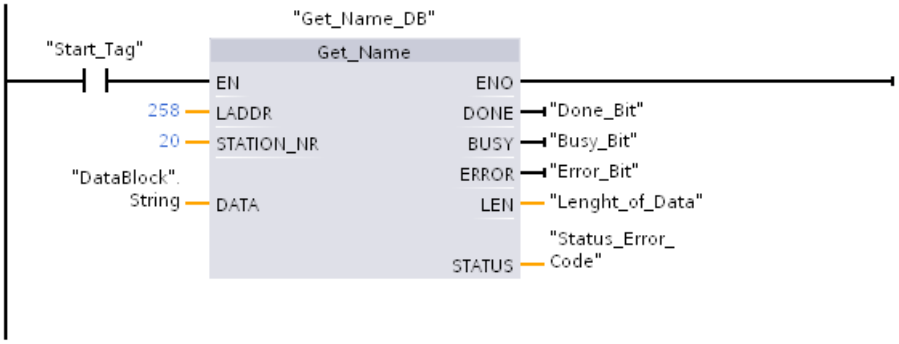
Configuration of the HMI panel:

- The HMI panel with the station name "HMI_IO-Device" is created in the network view and assigned to the same PROFINET IO system as the CPU.
- The operating mode "IO device" was activated in the properties of the hardware configuration for the HMI panel and the CPU was assigned as the IO controller.
- The device number "20" assigned in the properties under "Ethernet addresses".



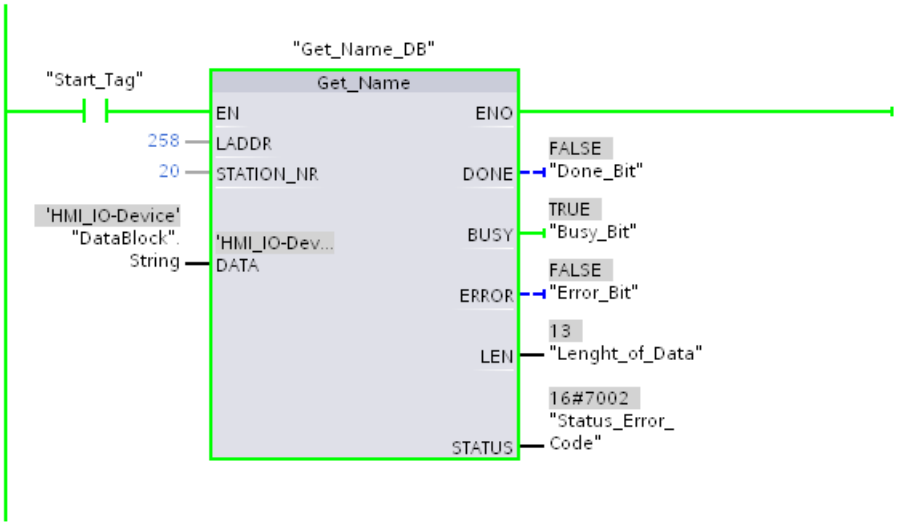
Assigning parameters to the instruction "GET_NAME":

- The hardware ID (258) of the IO system was specified at the parameter LADDR.
- The device number (20) of the HMI panel was specified at the parameter STATION_NR.
- A tag was connected with the data type STRING of a data block at the parameter DATA.
- PLC tags (memory area, flags) were defined for the output parameters of the instruction.



Execution of the instruction:

- After execution of the instruction, the name of the station of the HMI panels (HMI_IO-Device) is written into the data block at the parameter DATA.
- The number of characters of the name is output (13) at the parameter LEN.



See also

Switching display formats in the program status (Page 2980)

GetStationInfo: Read information of an IO device

Description

You can use the "GetStationInfo" instruction to read information of a PROFINET IO device. The instruction also enables you to read the information of an IO device located in a lower-level IO system (connected via CP/CM).

The IO device is addressed via the hardware identifier of the station at the LADDR parameter. The hardware identifier is displayed in the "Devices & Networks" view in the station properties.

Use the MODE parameter to select the information to be read. The instruction supports reading the IP address parameters according IPv4 in TIA V12 SP1 (MODE = 1).

At the DATA parameter, specify the data area to which you want to write the read address data. Use the "IF_CONF_v4" structure to store the IP address.

Enable reading of the address data using the REQ control parameter. This requires the IO device to be accessible.

The execution status of the read job is displayed via the BUSY, DONE, ERROR output parameters and the STATUS output parameter.

Note

Address the IO device using only the hardware identifier of the station

The station, the IO device and PROFINET interface have their own hardware identifier. Use only the hardware identifier of the station for the "GetStationInfo" instruction.

If a PROFINET is addressed via the LADDR parameter, for example, the address data is not read and the error code 8092 is generate.

To read the address data of an integrated PROFINET interface or a CM/CP in the central configuration, use the "RDREC" instruction.

Parameters

The following table shows the parameters of the "GetStationInfo" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter Request Activates the reading of the information with REQ = "1".
LADDR	Input	HW_DEVICE	I, Q, M, D, L or constant	Hardware identifier of the station of the IO device The number can be taken from the properties of the station in the Network view or from the "System constants" tab of the standard tag table.
DETAIL	Input	HW_SUBMODULE	I, Q, M, D, L or constant	The DETAIL parameter is not used. Leave the parameter unconnected.

Parameter	Declaration	Data type	Memory area	Description
MODE	Input	UNIT	I, Q, M, D, L or constant	Selection of address data to be read Connect the MODE parameter with "1" (read address parameter according to IPv4).
DATA	InOut	VARIANT	D, L	Pointer to the area to which the address data of the IO device is written. Use the "IF_CONF_v4" structure for MODE = 1.
DONE	Output	BOOL	I, Q, M, D, L	The instruction was executed successfully. The address data was transferred to the DATA parameter.
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Execution of the instruction complete. • 1: Execution of the instruction not yet complete.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter • 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter The parameter is only set for the duration of one call. To display the status, you should therefore copy STATUS to a free data area.

You can find more information on the data types in "Overview of the valid data types (Page 2236)".

Parameter DATA

Use the "IF_CONF_v4" structure at the DATA parameter to store the address parameter according to IPv4.

Byte	Parameter	Data type	Start value	Description
0 … 1	Id	UINT	30	ID of the "IF_CONF_v4" structure.
2 … 3	Length	UNIT	18	Length of data read in BYTE.
4 ... 5	Mode	UNIT	0	Not relevant for the "GetStationInfo" instruction (left at "0").
6 ... 9	InterfaceAddress	ARRAY [1..4] of BYTE	-	IP address of the IO device in the format IP_V4, e.g. for 192.168.3.10: • addr[1] = 192 • addr[2] = 168 • addr[3] = 3 • addr[4] = 10

Byte	Parameter	Data type	Start value	Description
10 … 13	SubnetMask	ARRAY [1..4] of BYTE	-	Subnet mask of the IO device in the format IP_V4, e.g. for 255.255.255.0: • addr[1] = 255 • addr[2] = 255 • addr[3] = 255 • addr[4] = 0
14 ... 17	DefaultRouter	ARRAY [1..4] of BYTE	-	IP address of the router in the format IP_V4, e.g. for 192.168.3.1: • addr[1] = 192 • addr[2] = 168 • addr[3] = 3 • addr[4] = 1

Parameter STATUS

Error code* (W#16#...)	Explanation
0	No error
7000	No job in progress.
7001	First call of the asynchronous instruction "GetStationInfo". Execution of the instruction not yet complete (BUSY = 1, DONE = 0).
7002	Additional call of the asynchronous instruction "GetStationInfo". Execution of the instruction not yet complete (BUSY = 1, DONE = 0).
8080	Value at the MODE parameter is not supported.
8090	The hardware identifier specified at the LADDR parameter is not configured.
8092	The LADDR parameter does not address a PROFINET IO device.
8093	Invalid data type at the DATA parameter.
80A0	Cannot read requested information.
80C0	Addressed IO device is not reachable.
80C3	The maximum number of simultaneous calls of the "GetStationInfo" instruction (10 instances) has been reached.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

DeviceStates: Read module status information in an IO system

Description

You use the instruction "DeviceStates" to query specific status information for all modules in an IO system, which means:

- Either for all IO devices in a PROFINET IO system
- Or for all DP slaves in a DP master system

The Boolean value that is output indicates the modules to which the selected status applies. You can, for example, read which IO devices are currently disabled in a PROFINET IO system.

Information is also displayed as to whether the status information to be read applies to at least one of the IO devices or DP slaves.

The instruction can be called in a cyclic OB as well as in an interrupt OB (e.g. OB82 - diagnostic interrupt).

Parameters

The following table shows the parameters of the "DeviceStates" instruction:

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_IOSYSTEM	I, Q, M, L or constant	Hardware identifier of the PROFINET IO or DP master system (see description below)
MODE	Input	UINT	I, Q, M, D, L or constant	Selection of status information to be read (see description below)
RET_VAL	Return	INT	I, Q, M, D, L	Status of instruction (see description below)
STATE	InOut	VARIANT	I, Q, M, D, L	Buffer for status of the IO devices or the DP slaves (see description below)

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

Parameter LADDR

You select the PROFINET IO or DP master system at the LADDR parameter by means of the hardware identifier.

The hardware identifier is available:

- Either in the Network view of the properties of the PROFINET IO or DP master system.
- Or in the PLC tag table in the listed system constants with the data type HW_IOSYSTEM.

Parameter MODE

You use the MODE parameter to read out status information. You can read out one of the following status information items for the entire PROFINET IO or DP master system:

- 1: IO devices/DP slaves are configured
- 2: IO devices/DP slaves are faulty
- 3: IO devices/DP slaves are disabled
- 4: IO devices/DP slaves exist
- 5: IO devices/DP slaves for which a problem has occurred. For example:
  - Maintenance demanded or recommended
  - Not accessible
  - Not available
  - Error occurred

Parameter STATE

With the STATE parameter, the status of the IO devices/DP slaves that was selected with the MODE parameter is output.

If the status selected using MODE applies to an IO device/DP slave, the following bits are set to "1" at the STATE parameter:

- Bit 0 = 1: Group display. The bit n of at least one IO device/DP slave was set to "1".
- Bit n = 1: The status selected with MODE applies to the IO device/DP slave.
  - With a PROFINET IO system, bit n corresponds to the device number of the respective IO device (see properties of the PROFINET interface in the device view and network view)
  - With a PROFIBUS DP system, bit n corresponds to the PROFIBUS address of the DP slave (see properties of the DP slave in the device view and network view)

Use "BOOL" or "Array of BOOL" as data type:

- To only output the bit for group display of the status information, you can use the data type BOOL at the STATE parameter.
- To output the status information for all IO devices/DP slaves, use Array of BOOL with the following length:
  - With PROFINET IO system: 1024 bits
  - With DP master system: 128 bits

Parameter RET_VAL

Error code* (W#16#...)	Description
0	No error
8091	Hardware identifier of the LADDR parameter does not exist. Check (for example, in the system constants) whether the value for LADDR exists in the project.
8092	LADDR does not address a PROFINET IO or DP master system.
8093	Invalid data type at the STATE parameter.
80B1	Instruction "DeviceStates" is not supported by the CPU.
80B2	The selected MODE parameter is not supported by the used CPU for the IO system specified in the LADDR parameter.
8452	The complete status information does not fit in the tag configured in the STATE parameter.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

Example - Reading the faulty stations of a PROFINET IO master system

A PROFINET IO system consists of 4 IO devices with the device numbers 1, 2, 3 and 4. The IO device with number 2 is faulty.

The instruction "DeviceStates" is executed for the PROFINET IO system with MODE = 2 (faulty/ not faulty).

The following bits are set in the STATE parameter:

- Bit 0 = 1: A fault exists for at least one of the IO devices.
- Bit 1 = 0: IO device with device number 1 is not faulty.
- Bit 2 = 1: IO device with device number 2 is faulty.
- Bit 3 = 0: IO device with device number 3 is not faulty.
- Bit 4 = 0: IO device with device number 4 is not faulty.
- Bit 5 = 0: irrelevant
- Bit 6 = 0: irrelevant
- ...

Example - Reading the faulty stations of a PROFIBUS DP master system

A DP master system consists of 4 DP slaves with the PROFIBUS addresses 3, 4, 5 and 6. The DP slave with address 4 is faulty.

The instruction "DeviceStates" is executed for the DP master system with MODE = 2 (faulty/ not faulty).

The following bits are set in the STATE parameter:

- Bit 0 = 1: A fault exists for at least one of the DP slaves.
- Bit 1 = 0: irrelevant

- Bit 2 = 0: irrelevant
- Bit 3 = 0: DP slave with address 3 is not faulty.
- Bit 4 = 1: DP slave with address 4 is faulty.
- Bit 5 = 0: DP slave with address 5 is not faulty.
- Bit 6 = 0: DP slave with address 6 is not faulty.
- Bit 7 = 0: irrelevant
- Bit 8 = 0: irrelevant
- ...

See also

Switching display formats in the program status (Page 2980)

ModuleStates: Read module status information of a module

Description

You can use the "ModuleStates" instruction to read the status information of the modules of a PROFINET IO device or PROFIBUS DP slave.

The Boolean value that is output indicates the modules to which the selected status applies. You can, for example, read which modules are currently disabled in a PROFINET IO device.

Information is also displayed as to whether the status information to be read applies to at least one of the modules.

The instruction can be called in a cyclic OB as well as in an interrupt OB (e.g. OB82 - diagnostic interrupt).

Parameters

The following table shows the parameters of the "ModuleStates" instruction:

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_DEVICE	I, Q, M, D, L or constant	Hardware identifier of the station (see description below)
MODE	Input	UINT	I, Q, M, D, L or constant	Selection of module status information to be read (see description below)
RET_VAL	Return	INT	I, Q, M, D, L	Status of instruction (see description below)
STATE	InOut	VARIANT	I, Q, M, D, L	Buffer for the module status (see description below)

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

Parameter LADDR

You select the IO device or the DP slave at the LADDR parameter by means of the hardware identifier of the station.

The hardware identifier is available:

- Either in the network view of the properties of the IO device station or DP slave station.
- Or in the PLC tag table for the listed system constants with the data type HW_DEVICE (for an IO device) or with the data type HW_DPSLAVE (for a DP slave).

Parameter MODE

You use the MODE parameter to read out status information. One of the following status information items can be read for the modules:

- 1: Modules are configured
- 2: Modules are faulty
- 3: Modules are disabled
- 4: Modules exist
- 5: There is a problem in the modules. For example:
  - Maintenance demanded or recommended
  - Not accessible
  - Not available
  - Error occurred

Parameter STATE

The STATE parameter outputs the status of the modules selected with the MODE parameter.

If the status selected using MODE applies to a module, the following bits are set to "1":

- Bit 0 = 1: Group display. The bit n of at least one module was set to "1".
- Bit n = 1: The status selected with MODE applies to the module in slot n-1 (example: bit 3 = slot 2).

Use "BOOL" or "Array of BOOL" as data type:

- To only output the bit for group display of the status information, you can use the data type BOOL at the STATE parameter.
- To output the status information for all modules, use Array of BOOL with a length of 128 bits.

Parameter RET_VAL

Error code* (W#16#...)	Description
0	No error
8091	Hardware identifier of the LADDR parameter does not exist. Check (for example, in the system constants) whether the value for LADDR exists in the project.
8092	LADDR does not address an IO device or DP slave.
8093	Invalid data type at the STATE parameter.
80B1	The instruction "ModuleStates" is not supported by the CPU.
80B2	The selected MODE parameter is not supported by the used CPU for the IO device/the DP slave in the LADDR parameter.
8452	The complete status information does not fit in the tag configured in the STATE parameter.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

Example

An IO device includes 4 modules in slots 1 to 4. The module in slot 2 is faulty.

The instruction "ModuleStates" is executed for the IO device with MODE = 2 (faulty/not faulty).

The following bits are set in the STATE parameter:

- Bit 0 = 1: A fault exists for at least one module.
- Bit 1 = 0: Slot number 0 (used by IO device)
- Bit 2 = 0: Module in slot number 1 is not faulty.
- Bit 3 = 1: Module in slot number 2 is faulty.
- Bit 4 = 0: Module in slot number 3 is not faulty.
- Bit 5 = 0: Module in slot number 4 is not faulty.
- Bit 6 = 0: irrelevant
- Bit 7 = 0: irrelevant

See also

Switching display formats in the program status (Page 2980)

GEN_DIAG: Generate diagnostics information

Description

The "GEN_DIAG" instruction generates diagnostics information for hardware components from other manufacturers for use in TIA Portal diagnostics. The GSD(GSDL/GSDML) file supplied by the manufacturer must first be installed before the instruction can be used.

The instruction generates all diagnostic events (including those for maintenance).

- Use the LADDR parameter to select the hardware component for which you want to generate a diagnostic event.
- Use the MODE parameter to specify whether the event is to be outgoing or incoming.
- Use the DiagEvent parameter to define the diagnostic event in the DiagnosticDetail structure. The structure is created automatically in the local interface of the block when you define a tag at the DiagEvent parameter.

The diagnostics information is provided synchronously. Diagnostics information transfer and alarm output are asynchronous.

NOTICE
Failsafe-specific errors messages are not valid If you define failsafe-specific diagnostics information at the DiagEvent parameter, the instruction will check this information and output the error code 80A1.

Parameters

The following table shows the parameters of the "GEN_DIAG" instruction:

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_ANY	I, Q, M, D, L or constant	Identification number of the hardware component
MODE	Input	UINT	I, Q, M, D, L or constant	Selection of incoming/outgoing information: • 1: The diagnostic event specified is an incoming event • 2: The diagnostic event specified is an outgoing event • 3: All diagnostic events are outgoing. There are therefore no hardware component faults (green diagnostics symbol). The DiagEvent parameter is not evaluated when MODE = 3.
DiagEvent	InOut	DiagnosticDetail	L	Specifies the diagnostic event (see "DiagEvent parameter").
RET_VAL	Return	INT	I, Q, M, D, L	Status of instruction / error message (see "RET_VAL parameter")

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

DiagEvent parameter

The structure DiagnosticDetail is a system data type for specifying the diagnostic event. Its structure is as follows:

Parameter		Data type	Description
DiagnosticDetail		Struct	
	ChannelInfo	WORD	Channel properties (0...7)
	ALID	UINT	Local ID of the alarm. The ID uniquely identifies the alarm.
	TextID	UINT	ID of an alarm text in a text list.
	Channel Number	UINT	Manufacturer-specific channel number (0x0000 — 0x7FFF)
	Addval_0	DWORD	Placeholder for additional information The value / list of values depends on the connection error.
	TextID2	UINT	Texts for CPU response (mode, OB calls, etc.).
	LADDR	HW_ANY	Identical to the LADDR parameter.
	TextListId	UINT	• 0: No text list • ≠0: ID of the text list
	Channel Direction	UINT	• 0000: irrelevant • FFF1: Input • FFF2: Output • FFF3: Input/Output
	Addval_1	DWORD	Placeholder for additional information on the channel error (depends on the GSD file). For channel error types, see also: IEC 61158 (PROFINET IO Type 10 and PROFIBUS DP Type 3).

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0	No error
8080	Value in the MODE parameter is not supported.
8090	Identification for hardware component not available at the LADDR parameter.
8091	Diagnostics information cannot be generated for the hardware component addressed at the LADDR parameter.
80A1	• Content of the DiagnosticsDetail structure at the DiagEvent parameter is invalid or inconsistent. • Failsafe-specific diagnostics information defined at the DiagEvent parameter (not valid).
80A4	Hardware component addressed cannot be accessed.
80C1	Insufficient resources for parallel execution.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

GET_DIAG: Read diagnostic information

Description

You can use the "GET_DIAG" instruction to read out the diagnostic information of a hardware object. The hardware object is selected using the parameter LADDR. With the MODE parameter, you select which diagnostic information is to be read out.

Parameters

The following table shows the parameters of the "GET_DIAG" instruction:

Parameter	Declaration	Data type	Memory area	Description
MODE	Input	UINT	I, Q, M, D, L or constant	Use the MODE parameter to select which diagnostic data is to be output.
LADDR	Input	HW_ANY (WORD)	I, Q, M, L or constant	Hardware identifier of the device.
RET_VAL	Return	INT	I, Q, M, D, L	Status of the instruction
CNT_DIAG	Output	UINT	I, Q, M, D, L	Number of output diagnostic details
DIAG	InOut	VARIANT	I, Q, M, D, L	Diagnostic information corresponds to the selected mode, see table below
DETAIL	InOut	VARIANT	I, Q, M, D, L	Diagnostics details correspond with the selected mode. Parameter is hidden (only used for MODE = 3).

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

MODE parameter

Depending on the value at the MODE parameter, different diagnostic data is output at the DIAG, CNT_DIAG and DETAIL output parameters.

MODE	Description	DIAG	CNT_DIAG	DETAIL
0	Output of all supported diagnostic information for a module as DWORD, where Bit X=1 indicates that mode X is supported.	Bit string of the supported modes as DWORD, where Bit X=1 indicates that mode X is supported.	0	-
1	Output of the inherent status of the addressed hardware object.	Diagnostics status. Output of the inherent status of the addressed hardware object in accordance with the DIS structure.	0	-
2	Output of the status of all subordinate modules of the addressed hardware object.	Diagnostics status. Output in accordance with the DNN structure.	0	-
3 *	Output of the inherent status of the addressed hardware object and detailed information on channel diagnostics.	Output of the inherent status of the addressed hardware object in accordance with the DIS structure.	Number of module status information.	Module status information in accordance with the DiagnosticDetail structure.
* Not supported by CPU S7-1200.				

DIS structure

With parameter MODE = 1 or = 3, the diagnostic information is output in accordance with the DIS structure. In this case, enter the system data type "DIS" as data type for the tag declaration.

The following table shows the meaning of the individual parameter values.

Parameter	Data type	Value	Description
MaintenanceState	DWORD	Enum	
		0	No maintenance required
		1	The module or device is disabled.
		2	-
		3	-
		4	-
		5	Maintenance required
		6	Maintenance demanded
		7	Error
		8	Status unknown / error in subordinate module
		9	-
		10	Inputs/outputs are not available.

Parameter	Data type	Value	Description
ComponentStateDetail	DWORD	Bit array	Status of the module sub-modules: • Bit 0 to 15: Status message of the module • Bit 16 to 31: Status message of the CPU
		0 to 2 (enum)	Additional information: • Bit 0: No additional information • Bit 1: Transfer not permitted
		3	Bit 3 = 1: At least one channel supports qualifiers for diagnostics
		4	Bit 4 = 1: Maintenance required for at least one channel or one component.
		5	Bit 5 = 1: Maintenance demanded for at least one channel or one component.
		6	Bit 6 = 1: Error in at least one channel or one component.
		7 to 10	Reserved (always = 0)
		11 to 14	• Bit 11 = 1: PNIO - sub-module correct • Bit 12 = 1: PNIO - replacement module • Bit 13 = 1: PNIO - incorrect module • Bit 14 = 1: PNIO - module disconnected
		15	Reserved (always = 0)
		16 to 31	Status information for modules generated by the CPU: • Bit 16 = 1: Module disabled • Bit 17 = 1: CiR operation active • Bit 18 = 1: Input not available • Bit 19 = 1: Output not available bit • 20 = 1: Overflow diagnostics buffer bit • 21 = 1: Diagnostics not available bit • 22 - 31: Reserved (always 0)
OwnState	UINT	Enum	The value of the parameter Ownstate describes the maintenance status of the module.
		0	No fault
		1	The module or device is disabled.
		2	Maintenance required
		3	Maintenance demanded
		4	Error
		5	The module or the device cannot be reached from the CPU (valid for modules and devices below a CPU).
		6	Inputs/outputs are not available.
		7	-
IOState	WORD	Bit array	I/O status of the module
		0	Bit 0 = 1: No maintenance required
		1	Bit 1 = 1: The module or device is disabled.
		2	Bit 2 = 1: Maintenance required
		3	Bit 3 = 1: Maintenance demanded
		4	Bit 4 = 1: Error

Parameter	Data type	Value	Description
		5	Bit 5 = 1: The module or the device cannot be reached from the CPU (valid for modules and devices below a CPU).
		6	Inputs/outputs are not available.
		7	Qualifier; bit 7 = 1, if bit 0, 2, or 3 are set
		8 to 15	Reserved (always = 0)
OperatingState	UINT	Enum	
		0	-
		1	In STOP / Firmware update
		2	In STOP / reset memory
		3	In STOP / self start
		4	In STOP
		5	Memory reset
		6	In START
		7	In RUN
		8	-
		9	In HOLD
		10	-
		11	-
		12	Module defective
		13	-
		14	No power
		15	CiR
		16	In STOP / without ODIS
		17	In
		18	
		19	
		20	

DNN structure

With parameter MODE = 2, the diagnostic information details are output in accordance with the DNN structure. In this case, enter the system data type "DNN" as data type for the tag declaration.

The following table shows the meaning of the individual parameter values.

Parameter	Data type	Value	Description
SubordinateState	UINT	Enum	Status of the subordinate module (see parameter OwnState of the DIS structure)
SubordinateIOState	WORD	Bitarray	Status of the inputs and outputs of the subordinate module (see parameter IO State of the DIS structure)
DNNmode	WORD	Bitarray	• Bit 0 = 0: Diagnostics enabled • Bit 0 = 1: Diagnostics disabled • Bit 1 to 15: Reserved

DiagnosticDetail structure

With parameter MODE = 3, the diagnostic information details are output in accordance with the DiagnosticDetail structure. In this case, enter the system data type "DiagnosticDetail" as data type for the tag declaration.

The following table shows the meaning of the individual parameter values.

Parameter	Data type	Description
ChannelInfo	WORD	Channel properties (0...7)
ALID	UInt	Identification ID of alarm
TextID	UNIT	ID of an alarm text in a text list.
ChannelNumber	UINT	Manufacturer-specific channel number (0x0000 — 0x7FFF)
Addval_0	DWORD	Placeholder for additional information The value / list of values depends on the connection error.
TextID2	UInt	Texts for CPU response (mode, OB calls, etc.).
LADDR	HW_ANY	Identical to the LADDR parameter.
TextListId	UInt	• 0: No text list • ≠0: ID of the text list
ChannelDirection	UInt	• 000: Manufacturer-specific • 001: Input • 002: Output • 003: Input/Output • 004 - 007: Reserved
AddVal_1	DWORD	Placeholder for additional information on the channel error (depends on the GSD file). For channel error types, see also: IEC 61158 (PROFINET IO Type 10 and PROFIBUS DP Type 3).

RET_VAL parameter

Error code* (W#16#...)	Description
0	No error
n	The data area at the DETAIL parameter is too small. Not all details of the diagnostic data can be output.
8080	Value in the MODE parameter is not supported.
8081	Type at the DIAG parameter is not supported with the selected mode (parameter MODE).
8082	Type at the DETAIL parameter is not supported by the selected mode (parameter MODE).
8090	LADDR does not exist
80C1	Insufficient resources for parallel execution.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

Pulse

CTRL_PWM: Pulse-width modulation

Description

You can use the "CTRL_PWM" instruction to enable and disable a pulse output supported by the CPU using the software.

- Enter the hardware ID of the pulse generator that you want to control with the instruction at the input PWM.
- The pulse output is enabled when the bit in the ENABLE input of the instruction is set.
  - If ENABLE has a value of TRUE the pulse generator will generate a pulse with the properties defined in the device configuration.
  - When the bit in the ENABLE input is reset or the CPU changes to STOP, the pulse output is disabled and no more pulses are generated.

BUSY always has the value of FALSE for S7-1200 since S7-1200 activates the pulse generator if the instruction "CTRL_PWM" is executed.

The ENO enable output is set only when the EN enable input has signal state "1" and no errors have occurred during execution of the instruction.

Note

Use of the force table for PWM and PTO

Digital inputs and outputs that are used for PWM and PTO cannot be forced. Digital inputs and outputs that were assigned via device configuration cannot be controlled by either the force table or the monitoring table.

Requirement

A prerequisite for the correct execution of the instruction is that the specified pulse generator is activated in the hardware configuration.

Open the properties of the module in the device view. Go to "Pulse generators (PTO/PWM)", open the desired PTO/PWM and activate the function the "Activate this pulse generator" under "General".

Go to "Assigning parameters" and set the Pulse options.

Note

Pulse output parameters are assigned exclusively in the device configuration and not using the "CTRL_PWM" instruction. Any change of parameters that is intended to have an effect on the CPU must therefore be made while the CPU is in STOP mode. Changing the duration of the pulse is an exception.

Changing the pulse duration from the user program

The pulse duration setting made in the dialog "Pulse options" can be changed from the user program.

The value set for "Initial pulse duration" is written in the output bytes of the pulse generator. The start address and the end address are displayed in the properties of the pulse generator under "I/O-addresses".

To change the pulse duration, write the desired values in the output word address specified in the device configuration.

Example:

- A value of 500 (decimal) is used for the "Initial pulse duration". The start address of the PTO/PWM is "1000" and the end address is "1001".
- The binary value of "0000000111110100" (=500 decimal) is written in both of the output bytes.
  - Start address (AB1000): 0000_0001 (BIN)
  - End address (AB1001): 1111_0100 (BIN)

Please note that the pulse duration always depends on the parameters set for the Pulse duration format (hundredths, thousandths, ...).

Parameters

The following table shows the parameters of the "CTRL_PWM" instruction:

Parameters	Declaration	Data type	Memory area	Description
PWM	Input	HW_PWM	I, Q, M, D, L or constant	Hardware ID of the pulse generator The hardware ID can be found in the properties of the pulse generator in the Device view. The hardware IDs of the pulse generators are also listed in the system constants.
ENABLE	Input	BOOL	I, Q, M, D, L or constant	The pulse output is enabled when ENABLE = TRUE and disabled when ENABLE = FALSE.
BUSY	Output	BOOL	I, Q, M, D, L	Processing status
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction (see below).

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter STATUS

Error code* (W#16#...)	Description
0	No errors
80A1	Hardware ID of the pulse generator is invalid.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

Recipes and data logging

Recipe functions

RecipeExport: Exporting recipes

Description

The "RecipeExport" instruction exports the recipe data from a data block to a CSV file on the memory card of the CPU.

The export is triggered by the "REQ" parameter. The BUSY parameter is set to "1" during the export. During the export, the CSV file is created in the "Recipes" folder in the main directory of the memory card. The name of the data block is used as file name of the created CSV file. If a CSV file with an identical name already exists, it must be deleted before this export.

After the execution of the instruction, BUSY is reset to "0" and the completion of the operation is indicated with "1" at the DONE parameter. If an error occurs during execution, this is signaled by the parameters ERROR and STATUS.

Parameters

The following table shows the parameters of the "RecipeExport" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, T, C or constant (T and C are only available in LAD and FBD with S7-1500)	Control parameter REQUEST: Activates the export on a rising edge.
RECIPE_DB	InOut	VARIANT	D	Pointer to the recipe data block. For information on the structure of the data block, refer to: Structure of a recipe DB (Page 4771)

Parameter	Declaration	Data type	Memory area	Description
DONE	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Job not yet started or still executing. • 1: Job executed without errors.
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter • 0: The instruction is not executed. • 1: The instruction is executed.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Neither warning nor error. • 1: An error has occurred. STATUS supplies detailed information on the type of error.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter See the "STATUS" parameter table.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter STATUS

Error code* (W#16#...)	Explanation
0	No error occurred
7000	No job processing active
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8090	File name of the CSV file contains invalid characters. The file name of the CSV file is identical to the name of the data block.
8091	The data structure referenced with RECIPE_DB cannot be processed.
8092	Data structure at RECIPE_DB parameter exceeds 5000 bytes.
80B3	Insufficient memory space on the memory card or the internal load memory.
80B4	The memory card is write-protected.
80C0	CSV files is temporarily locked.
80C1	Data block temporarily locked.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

RecipeImport: Importing recipes

Description

The "RecipeImport" instruction imports the recipe data from a CSV file on the memory card into the data block at the RECIPE_DB parameter. The values in the data block are overwritten in the process.

Note the following when importing the CSV file:

- The CSV file must be located in the "Recipes" directory of the memory card.
- The name of the CSV file must match the name of the data block at the RECIPE_DB parameter.
- The first line (header) of the CSV file contains the name of the recipe components (see also: Structure of a recipe DB (Page 4771)). The first line is ignored during import. The names of the recipe components in the CSV file and the data block are not reconciled during import.
- In each case the first value in each line of the CSV file is the index number of the recipe. The individual recipes are imported in the order of the index. For this, the index in the CSV file has to be in ascending order and may contain no gaps (if this is not the case, the error message 80B0 is output at the STATUS parameter.
- The CSV file may not contain more recipe data records than provided for in the data block. The maximum number of data records is indicated by the array limits in the data block.

The import is triggered by the "REQ" parameter. The BUSY parameter is set to "1" during the import. After the execution of the instruction, BUSY is reset to "0" and the completion of the operation is indicated with "1" at the DONE parameter. If an error occurs during execution, this is signaled by the parameters ERROR and STATUS.

Parameters

The following table shows the parameters of the "RecipeImport" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, T, C or constant (T and C are only available in LAD and FBD with S7-1500)	Control parameter REQUEST: Activates the import on a positive edge.
RECIPE_DB	InOut	VARIANT	D	Pointer to the recipe data block. For information on the structure of the data block, refer to: Structure of a recipe DB (Page 4771)
DONE	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Job not yet started or still executing. • 1: Job executed without errors.
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter • 0: The instruction is not executed. • 1: The instruction is executed.

Parameter	Declaration	Data type	Memory area	Description
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Neither warning nor error. • 1: An error has occurred. STATUS supplies detailed information on the type of error.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter See the "STATUS" parameter table.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter STATUS

Error code* (W#16#...)	Explanation
0	No error occurred
7000	No job processing active
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8090	The file name contains invalid characters.
8092	No matching CSV file found for the import. Possible cause: The name of the CSV file does not match the name of the recipe DB.
80C0	CSV file is temporarily locked.
80C1	Data block is temporarily locked.
80B0	Numbering in the index of the CSV file is not continuous, not ascending or exceeds the maximum number (array limit) in the data block.
80B1	Structure of the recipe data block and the CSV file do not match: The CSV file contains too many fields.
80B2	Structure of the recipe data block and the CSV file do not match: The CSV file contains too few fields.
80D0 +n	Structure of the recipe data block and the CSV file do not match: Data type in field n does not match (n<=46).
80FF	Structure of the recipe data block and the CSV file do not match: Data type in field n does not match (n>46).
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

Structure of a recipe DB

Introduction

The following sections describes the structure of a recipe DB based on a simple example. The recipe DB consists of five data records, three of which are used. The fourth and fifth data

records are left free for later expansions. Each data record contains one recipe, which is made up of the recipe name and eight ingredients.

product name	water	barley	wheat	hops	yeast	waterTmp	mashTmp	mashTime	QTest
Pils	10	9	3	280	39	40	30	100	0
Lager	10	9	3	150	33	50	30	120	0
BlackBeer	10	9	3	410	47	60	30	90	1
Not_used	0	0	0	0	0	0	0	0	0
Not_used	0	0	0	0	0	0	0	0	0

Structure of the recipe data block

The recipe data are implemented as follows in a global data block:

- The template for all recipes is the PLC data type "Beer_Recipe" with the recipe components "procutname", "water", etc. with the corresponding data types.
- In a global data block, the PLC data type is used as Array [1.. 5] of "Beer_Recipe". The array limits (1 to 5 in this case) indicate the maximum number of recipes that the DB may contain.
- The values for the recipe components are added as start value in the data block.
- The global DB is interconnected with the instruction via the InOut parameter RECIPE_DB.

Recipe_DB

	Name	Data type	Offset	Start value
1	Static			
2	Products	Array [1 .. 5] of "Beer_Recipe"	...	
3	Products[1]	"Beer_Recipe"	...	
4	Products[2]	"Beer_Recipe"	...	
5	Products[3]	"Beer_Recipe"	...	
6	productname	String[20]	...	'BlackBeer'
7	water	UInt	...	10
8	barley	UInt	...	9
9	wheat	UInt	...	3
10	hops	UInt	...	410
11	yeast	UInt	...	47
12	waterTmp	UInt	...	60
13	mashTmp	UInt	...	30
14	mashTime	UInt	...	90
15	QTest	UInt	...	1
16	Products[4]	"Beer_Recipe"	...	
17	Products[5]	"Beer_Recipe"	...	

Exporting to CSV file

After the execution of the "RecipeExport (Page 4768)" instruction the data of the DB are written to a CSV file with the following structure:

Recipe_DB.csv

```
index,productname,water,barley,wheat,hops,yeast,waterTmp,mashTmp,mashTime,QTest
1,”Pils”,10,9,3,280,39,40,30,100,0
2,”Lager”,10,9,3,150,33,50,30,120,0
3,”BlackBeer”,10,9,3,410,47,60,30,90,1
4,”Not_used”,0,0,0,0,0,0,0,0,0
5,”Not_used”,0,0,0,0,0,0,0,0,0
```

Display in Excel

The CSV file can be opened in Excel to make reading and editing easier. If the commas are not recognized as separators when you open the file, use the Excel import function to output the data in structured form:

	A	B	C	D	E	F	G	H	I	J	K
1	index	product	water	barley	wheat	hops	yeast	waterTmp	mashTmp	mashTime	QTest
2	1	"Pils"	10	9	3	280	39	40	30	100	0
3	2	"Lager"	10	9	3	150	33	50	30	120	0
4	3	"BlackBeer"	10	9	3	410	47	60	30	90	1
5	4	"Not_used"	0	0	0	0	0	0	0	0	0
6	5	"Not_used"	0	0	0	0	0	0	0	0	0

Editing the CSV file

The CSV file can be uploaded to the PC/programming device by means of the Web server and edited. After editing, the modified file can be reloaded in the CPU. You can use the "RecipeImport (Page 4770)" instruction to re-import the modified data from the CSV file into the data block.

Note that the modified data must still be compatible with the data block. In other words:

- No changes may be made to the structure in the table (for example, by adding ingredients in a new column).
- If you add data records to the file, make sure when importing into the data block that the array limits through which you specify the maximum number of data records correspond to at least the number of data records.
- An index is automatically generated during export to the CSV file. If you create additional data records, add consecutive index numbers accordingly.
- The values in the table cells must correspond to the data types used in the data block with respect to format as well as length.
  - Example 1: If data type INT was used in the data block, you can only use integers in the table.
  - Example 2: If data type SINT was used in the data block, you can only use integers with the values -128 to +127 in the table.

  Always keep in mind the permitted data types and data areas as listed in the table below when you make changes in the table.

Data type		Format	Note
Floating-point numbers	LReal	+9.9999999999999E +999	Always written exponentially
	Real	+9.999999E+99	Always written exponentially
Signed integers	LInt	+9999999999999999999	Value range for signed integers: -9223372036854775808 .. +9223372036854775807
	DInt	+9999999999	Value range of integers: -2147483648 to +2147483647
	Int	+99999	Value range of integers: -32768 to +32767
	SInt	+999	Value range of integers: -128 to +127
Unsigned integers	ULInt	+9999999999999999999 9	Value range of integers: 0 to +18446744073709551615
	UDInt	+9999999999	Value range of integers: 0 to +4294967295
	UInt	+99999	Value range of integers: 0 to + 65535
	USInt	+999	Value range of integers: 0 to +255
Binary numbers	LWord	+9999999999999999999 9	Value range of integers: +00000000000000000000 to +18446744073709551615
	DWord	+9999999999	Value range of integers: +0000000000 to +4294967295
	Word	+99999	Value range of integers: +00000 to + 65535
	Byte	+999	Value range of integers: +000 to +255
	Bool	9	Range of values: 0 or 1
Date and time	LTIME	hhh:mm:ss.999	ISO format with milliseconds
	TIME	hhh:mm:ss.999	ISO format with milliseconds
	S5TIME	hhh:mm:ss.999	ISO format with milliseconds
	LDT	YYYY-MM-DD hh:mm:ss. 999	ISO format with milliseconds
	DTL	YYYY-MM-DD hh:mm:ss. 999	ISO format with milliseconds
	DT	YYYY-MM-DD hh:mm:ss. 999	ISO format with milliseconds
	DATE	YYYY-MM-DD	ISO format
	LTime_Of_ Day	hh:mm:ss.999_999_999	ISO format
	TOD	hh:mm:ss.999	ISO format
Character	String	"abcd"	The string must be in double quotation marks.
	Char	"a"	Individual character in double quotation marks.

Data logging

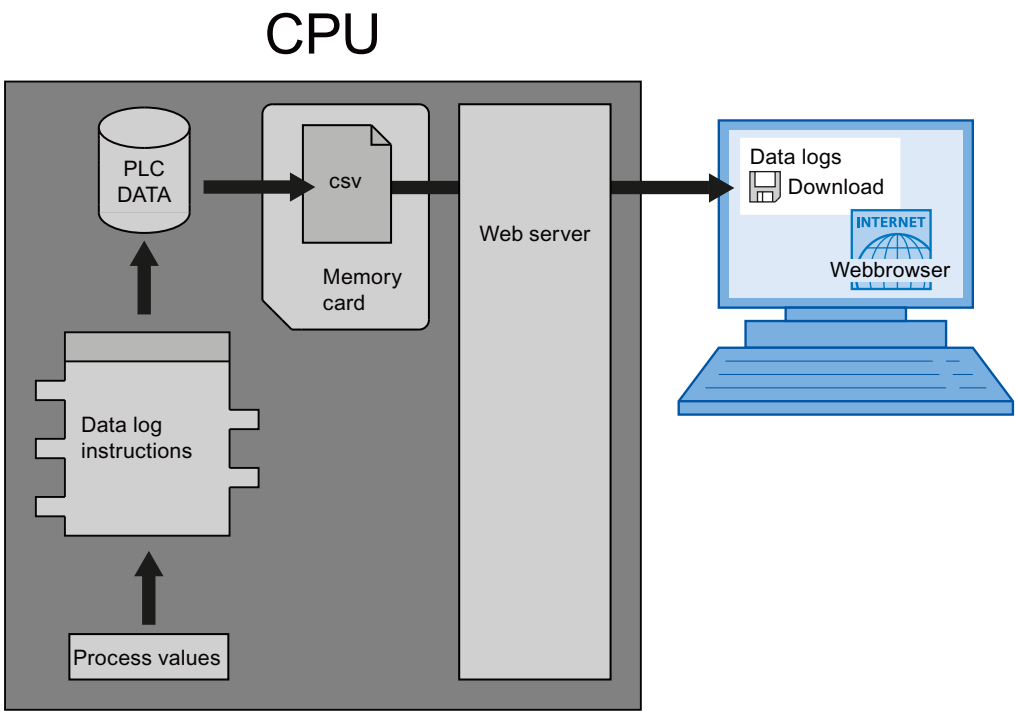
Data logging - Overview

Saving process values

The data logging instructions are used in the user program to save process values to data logs. Data logs can be saved on the Memory Card (MC) or in the internal load memory. The data logs are saved in CSV format (Comma Separated Value).

The data logging instructions are used in your program to create or open a data log, to write an entry and to close the data log file.

You decide by the creation of the data buffer which program values are stored in the data log during the creation of the data buffer. The data buffer is used as a memory for new data log entries. New values have to be written to the buffer before "DataLogWrite (Page 4789)" is called. During execution of the "DataLogWrite (Page 4789)" instruction, data is written from the buffer to a data log record.



Data log files can be copied to the PC as follows:

- If the PROFINET interface is connected to the PC, you use a Web browser to access the data logs via the Web server. The CPU can be in RUN or STOP mode for this. If the CPU is in "RUN" mode, the program continues running while the Web server is transferring data.
- If there is a memory card in the CPU, you can remove this card and insert it into a standard slot for SD (Secure Digital) cards or MMC (MultiMediaCard) cards on a PC or programming device. Use File Manager to transfer the data log files from the memory card to the PC. The CPU goes to "STOP" when you remove the memory card.

Properties of the data log

Writing of the data records of a data log is carried out in accordance with the principle of a ring buffer. New data records are added until the maximum number of data records is reached (RECORD parameter). The next data record then overwrites the "oldest" data record of the data log.

If you want to prevent data records from being overwritten, use the "DataLogNewFile (Page 4795)" instruction to create a new data log file based on the current data log. New data records are then written into the new data log.

Creating data logs

With the "DataLogCreate (Page 4777)" instruction, you can create a new data log file in the ""\DataLogs" directory of the load memory.

- The name assigned at the NAME parameter is the designation for the data log and is also used the file name for the CSV file. The file is stored in the directory "DataLogs".
- The block parameter DATA specifies the data buffer for the new data log object and the columns and data types in the data log. The columns and data types of a data record in the data log are generated by the elements in the structure declaration or the array declaration of this data buffer. Each element of a structure or of an array corresponds to a column in a line in the data log.
- You can use the HEADER block parameter to assign a header text in the header to each column.
- The "DataLogCreate (Page 4777)" instruction returns an ID. This ID is used by the other data logging instructions as a reference for the created data log.

Opening data logs

You use the instruction "DataLogOpen" (S7-1200 and S7-1500) to open an existing data log on the memory card. A data log must be open before you can write new data records to it.

Data log opens automatically when you execute the instructions "DataLogCreate (Page 4777)" and "DataLogNewFile (Page 4795)".

Up to 10 data logs can be open simultaneously. You can select the data log to be opened using the ID or name of the data log.

- If you specify the ID and the name of the data log in the ID and NAME parameters, respectively, the data log will be identified based on the ID. The data log name will not be compared.
- If you select the data log using the NAME parameter and no ID is specified, the ID will be displayed in the ID parameter when the data log is opened.
- If you select the data log using the ID parameter and no name is specified, the name will be displayed in the NAME parameter when the data log is opened.

You use the MODE parameter to specify whether the data records of the data log are to be deleted upon opening.

Writing to data logs

A data log must be open ("DataLogOpen (Page 4784)" instruction) before a data record can be written to a data log. The "DataLogWrite (Page 4789)" instruction writes a data record to the data log.

Closing data logs

You use the "DataLogClose (Page 4791)" instruction to close an open data log. You select the data log using the ID parameter.

The data log is closed automatically when the CPU switches to STOP and upon restart.

Deleting data logs

You use the instruction "DataLogDelete (Page 4793)" (S7-1500) to delete a data log file from the memory card. The data log and the data records it contains can only be deleted if the log was created with the instruction "DataLogCreate (Page 4777)".

Select the data log to be deleted using the NAME and ID parameters. The ID parameter is evaluated first. If there is a data log with the relevant ID, the NAME parameter will not be evaluated. If the value "0" is used at the ID parameter, a value with the data type STRING must be used at the NAME parameter.

Clearing data logs

You use the instruction "DataLogClear (Page 4788)" (S7-1500) to delete all the data records in an existing data log. The instruction does not delete the optional header of the CSV file (see the description of the HEADER parameter of the instruction "DataLogCreate (Page 4777)").

You use the ID parameter to select the data log whose data records are to be deleted. Before you can delete data records, the data log must be open.

New file for data logs

You use the instruction "DataLogNewFile (Page 4795)" (S7-1200) or "DataLogTypedNewFile (Page 4796)" (S7-1500) to create a new data log with the same properties as an existing data log. This allows you to retain the contents of an existing data log.

When called, the instruction creates a new data log on the memory card or in the internal load memory with the name defined at the NAME parameter. You use the ID parameter to specify the ID of the old data log whose properties you want to apply to the new data log. The ID of the new data log is then output via the ID parameter.

You specify the file size of the new data log with the RECORDS parameter of the instruction.

You can run a consistency check for "DataLogTypedNewFile (Page 4796)" (S7-1500).

DataLogCreate: Create data log

Description

With the "DataLogCreate" instruction you create a data log.

The data log is saved on the memory card or in the internal load memory in the directory "\DataLogs". The amount of data that can be stored in a data log is dependent on the available space on the memory card or the storage space in the internal load memory of the CPU.

You specify the maximum number of data records that can be stored in a data log in the RECORDS parameter. If the specified maximum number of data records in the data log is reached, the oldest data record will be overwritten. To avoid overwriting of existing data records, use the "DataLogNewFile (Page 4795)" instruction. The instruction can be used to create a new data log with the same structure when the number specified at the RECORDS parameter is reached (return value 1 at the STATUS parameter of the "DataLogWrite (Page 4789)" instruction). The data records are then saved in the new data log.

You can specify the name for the data log in the NAME parameter. The data log is created in CSV (Comma Separated Value) format. With the HEADER parameter, you can create an (optional) header for the data log.

Once the data log is created, it is opened automatically. This means that data can be written.

Parameters

The following table shows the parameters of the "DataLogCreate" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, L, D, T, C or constant (T and C are only available in LAD and FBD with S7-1500)	Execution of the instruction The data log is created with a rising edge at the parameter REQ.
RECORDS	Input	UDInt	I, Q, M, L, D or constant	Maximum number of data records in the data log If the instruction "DataLogWrite (Page 4789)" writes more records than specified in this parameter, the oldest record is overwritten.
FORMAT	Input	UInt	I, Q, M, L, D or constant	Data format: • 0: Internal (not supported) • 1: CSV (Comma separated values)
TIMESTAMP	Input	UInt	I, Q, M, L, D or constant	Time stamp: • 0: No time stamp • 1: Date and time The extra columns in the header are automatically added if time-stamping is activated.
NAME	Input	VARIANT	L, D	Name of the data log The specified name is also used as a file name for the csv file. The restrictions for Windows file names apply when assigning the name. The following characters must not be used: "\", "/", ":", "*", "?", "<", ">", "|", "space"
ID	InOut	DWORD	I, Q, M, L, D	Object ID of the data log (only output) The ID of the data log is required for further data logging instructions to address the created data log.
HEADER	InOut	VARIANT	L, D	Header of the data log (optional) The parameter is hidden after the instruction is added. The header is written as the first line in the CSV file.
DATA	InOut	VARIANT	L, D	Pointer to the data structure that is to be written as data record when executing the instruction "DataLogWrite (Page 4789)".

Parameter	Declaration	Data type	Memory area	Description
DONE	Output	BOOL	I, Q, M, L, D	Status parameter: • 0: Processing not yet complete. • 1: Processing of instruction finished successfully.
BUSY	Output	BOOL	I, Q, M, L, D	Status parameter: • 0: Processing of the instruction has not started, completed, or canceled. • 1: Processing of the instruction is in progress.
ERROR	Output	BOOL	I, Q, M, L, D	Status parameter: • 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.
STATUS	Output	WORD	I, Q, M, L, D	Detailed status information: Detailed error and status information is output at the parameter STATUS. The parameter is only set for the duration of one call. To display the status, you should therefore copy the STATUS parameter to a free data area.

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

HEADER parameter

The HEADER parameter is a VARIANT pointer to a data block that defines a header for the CSV file (header). The header is always the first line in the CSV file representation.

- When creating a header, remember that the individual columns must be separated by a comma (S7-1200) or a semicolon (S7-1500).
- A STRING, Array of BYTE, or Array of CHAR data type can be used for the individual column names. A longer character string is possible with the Array [...] of type data type than with the STRING data type. When the STRING data type is used, the length is limited to 254 bytes.

If no header is to be created, do not specify a value in the HEADER parameter.

DATA parameter

The DATA parameter is a VARIANT pointer to a structure or an array in a data block. An element of a structure or an array corresponds to a column in the data log with a specific data type.

Note the following when creating the data block:

- The number of columns must correspond to the number of columns defined in the HEADER parameter.
- Each element of the structure or the array corresponds to a column entry in the CSV file. Therefore, no nesting structures (STRUCT in STRUCT) may be used when using the data type STRUCT.
- The data structure can contain up to 256 elements. If there are more than 256 elements, error code 8C52 is output at the STATUS parameter.
- The tags of the data block can be set as retentive or non-retentive tags. However, the retentivity setting must be the same for all tags of the data block.

STATUS (S7-1200) parameter

Error code* (W#16#...)	Description
0	No errors.
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8070	Complete internal instance memory is assigned.
8090	Invalid file name (see description of the NAME parameter).
8093	Data log already exists.
8097	File length exceeds the file system limit.
80B3	Insufficient memory space on the memory card.
80B4	The memory card is write-protected.
80C1	Too many data logs are open.
80C3	Resources are not sufficient. Possible causes: • Multiple calling of instructions with different parameters. • More than 10 data logs open simultaneously. At least one data log must be closed in order to continue because the instruction "DataLogCreate" automatically opens the the log that is being created.
8453	Invalid format selection.
8553	Invalid time stamp.
8B51	Invalid data type at HEADER parameter.
8C20	String with length specification other than 254 used.
8C51	Invalid data type at DATA parameter.
8C52	Structure at the parameter DATA contains more than 256 elements.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

STATUS (S7-1500) parameter

Error code* (W#16#...)	Description
0	No errors.
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
8070	Complete internal instance memory is assigned.
8090	Invalid file name (see description of the NAME parameter).
8091	"NAME" parameter is not a string.
8093	Data log already exists.
8097	File length exceeds the file system limit.
80B3	Insufficient memory space on the memory card.
80B4	The memory card is write-protected.
80C0	Access currently not possible.
80C1	Too many data logs are open.
80C3	Resources are not sufficient. Possible causes: • Multiple calling of instructions with different parameters. • More than 10 data logs open simultaneously. At least one data log must be closed in order to continue because the instruction "DataLogCreate" automatically opens the the log that is being created.
8253	The RECORDS parameter has an invalid value.
8353	Invalid format selection.
8453	Invalid time stamp.
8B51	Invalid data type at HEADER parameter.
8C51	Invalid data type at DATA parameter.
8C52	Structure at the parameter DATA contains more than 256 elements.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

Example

In the following example, a simple data log is created with a time stamp and three process values.

Tags in the global data block

The values for the input parameters of the data log are stored in the global data block "DataLogDB":

- DataLogName (String): The tag contains the name of the data log which is also used as the file name for the CSV file.
- DataLogID (DInt): The ID of the data log is written in this tag when the instruction is called.
  - The ID is automatically assigned by the instruction.
  - Use the tag "DataLogID" in other DataLog instructions in order to address this data log.

- MyHeader (String): This tag contains the header of the DataLog, i.e. the column headings for the process values. A semicolon is used as a delimiter for the columns when using a S7-1500.
- MyData (Struct): This tag contains the three process values that are written in the data log. Each time a record is written (instruction "DataLogWrite (Page 4789)") the current values are written in a new record.

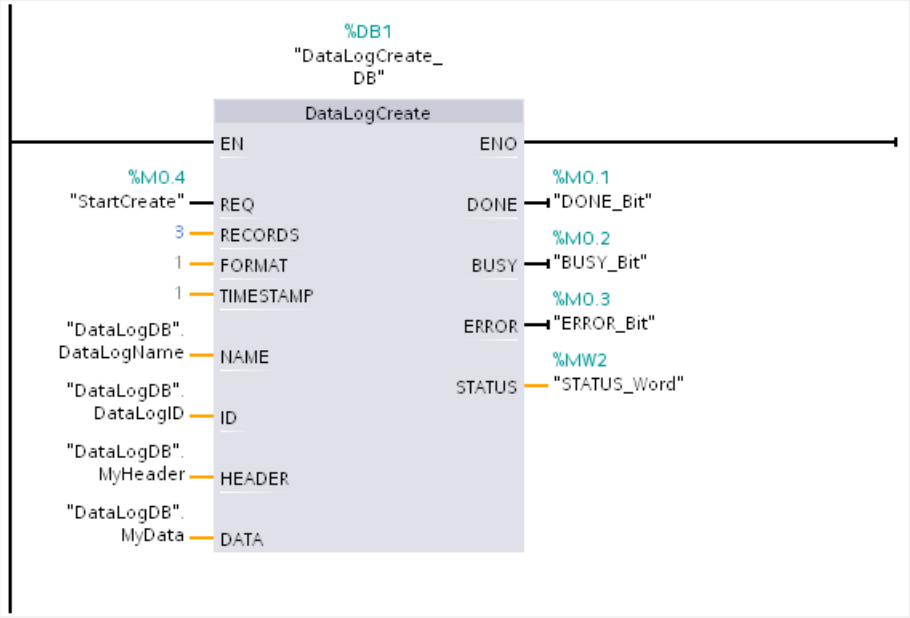
DataLogDB

	Name	Data type	Start value	Retain
1	Static			
2	DataLogName	String	'MyDataLog'	
3	DataLogID	DInt	0	
4	MyHeader	String	'Value1;Value2;Value3'	
5	MyData	Struct		
6	ProcessValue1	Int	2	
7	ProcessValue2	Int	3	
8	ProcessValue3	Int	4	

Calling the "DataLogCreate" instruction

The instruction is called with the following input parameters:

- REQ (BOOL): With REQ = "1" the data log is created.
- RECORD (3): A maximum of three data records can be written in the data log. Thereafter, the oldest data record is overwritten.
- FORMAT (1): The data log is created as a CSV file.
- TIMESTAMP (1): Activated. Two additional columns are automatically created (date and time) for the data log. The current time stamp is written in the data record each time "DataLogWrite (Page 4789)" is executed.
- NAME (VARIANT): Pointer to the tag "DataLogName" in the data block "DataLogDB".
- ID (VARIANT): Pointer to the tag "DataLogID" in the data block "DataLogDB".
- HEADER (VARIANT): Pointer to the tag "MyHeader" in the data block "DataLogDB".
- DATA (VARIANT): Pointer to the tag "MyData" in the data block "DataLogDB".



Readout of the data log via the web server

You can readout the created data log with the web server:

- Activate the web server in the properties of the CPU. The authorization "...to read files" must be activated for the web server in order to readout the data log.
- You can access the web server via the Internet browser by entering the IP address of the CPU as a URL.
- Under "Filebrowser" the directory "\DataLogs" was automatically created which contains the data log.
- The DataLog only contains one entry "//END" if the instruction "DataLogWrite (Page 4789)" has still not been executed. The first data record is written after the initial execution of "DataLogWrite (Page 4789)".

	A	B	C	D	E	F
1	SeqNo	Date	Time	ProcessValue1	ProcessValue2	ProcessValue3
2	1	01.07.2013	12:33:42.917	2	3	4
3						
4						
5						

See also

Switching display formats in the program status (Page 2980)

Data logging - Overview (Page 4774)

DataLogOpen: Open data log

DataLogOpen: Open data log

Description

You use the "DataLogOpen" instruction to open an existing data log on the memory card. A data log must be open before you can write new data records to it.

Data log opens automatically when you execute the instructions "DataLogCreate (Page 4777)" and "DataLogNewFile (Page 4795)".

Up to 10 data logs can be open simultaneously. You can select the data log to be opened using the ID or name of the data log.

- If you specify the ID and the name of the data log in the ID and NAME parameters, respectively, the data log will be identified based on the ID. The data log name will not be compared.
- If you select the data log using the NAME parameter and no ID is specified, the ID will be displayed in the ID parameter when the data log is opened.
- If you select the data log using the ID parameter and no name is specified, the name will be displayed in the NAME parameter when the data log is opened.

You use the MODE parameter to specify whether the data records of the data log are to be deleted upon opening.

Parameters

The following table shows the parameters of the "DataLogOpen" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, L, D or constant	Execution of the instruction upon a rising edge.
MODE	Input	UInt	I, Q, M, L, D or constant	Mode for opening the data log: • MODE= "0" Data records of the data log are retained • MODE= "1" Data records of the data log are deleted, but the header is retained
NAME	Input	VARIANT	L, D	(File) name of the data log.
ID	InOut	DWORD	I, Q, M, L, D	Object ID of the data log.
DONE	Output	BOOL	I, Q, M, L, D	Instruction was executed successfully.
BUSY	Output	BOOL	I, Q, M, L, D	Execution of instruction not yet complete.

Parameter	Declaration	Data type	Memory area	Description
ERROR	Output	BOOL	I, Q, M, L, D	• 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.
STATUS	Output	WORD	I, Q, M, L, D	Status parameter The parameter is only set for the duration of one call. To display the status, you should therefore copy the STATUS parameter to a free data area.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter STATUS

Error code* (W#16#...)	Description
0	No errors.
2	Warning: Data log file has already been opened by this application.
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8070	Complete internal instance memory is assigned.
8090	There are inconsistencies between the data log definition and existing data log data.
8091	A data type other than String was used at the NAME parameter.
8092	Data log does not exist.
80B4	The memory card is write-protected.
80C0	The data log file is locked.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

Data logging - Overview (Page 4774)

DataLogOpen: Open data log

Description

You use the "DataLogOpen" instruction to open an existing data log on the memory card. A data log must be open to write new data records to it.

Data log opens automatically when you execute the instructions "DataLogCreate (Page 4777)" and "DataLogNewFile (Page 4795)".

Up to 10 data logs can be open simultaneously. You can select the data log to be opened using the ID or name of the data log.

- If you specify the ID and the name of the data log in the ID and NAME parameters, respectively, the data log will be identified based on the ID. The data log name will not be compared.
- If you select the data log using the NAME parameter and no ID is specified, the ID will be displayed in the ID parameter when the data log is opened.
- If you select the data log using the ID parameter and no name is specified, the name will be displayed in the NAME parameter when the data log is opened.

You use the MODE parameter to specify whether the data records of the data log are to be deleted upon opening.

The parameter DATA enables a consistency check between the data log to be opened and the definition of the data log of the instruction "DataLogCreate (Page 4777)". The consistency check can only be executed if the data log was created with the instruction "DataLogCreate (Page 4777)":

- If you are using the same pointer at the DATA parameter as at the DATA parameter of the "DataLogCreate (Page 4777)" instruction, a check is performed to determine whether the data types match. If the data types do not match, the error code W#16#80A0 is output at the STATUS parameter.
- If the data log to be opened was not created with "DataLogCreate (Page 4777)", a consistency check is not possible. In this case, enter the value "NULL" in the DATA parameter.

Parameters

The following table shows the parameters of the "DataLogOpen" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, L, D, T, C or constant (T and C are only available in LAD and FBD with S7-1500)	Execution of the instruction upon a rising edge.
MODE	Input	UInt	I, Q, M, L, D or constant	Mode for opening the data log: • MODE= "0" Data records of the data log are retained • MODE= "1" Data records of the data log are deleted, but the header is retained
NAME	Input	VARIANT	L, D	(File) name of the data log.
ID	InOut	DWORD	I, Q, M, L, D	Object ID of the data log.
DATA	InOut	VARIANT	L, D	With consistency check: Pointer to the data area at the DATA parameter of the "DataLogCreate (Page 4777)" instruction.

Parameter	Declaration	Data type	Memory area	Description
DONE	Output	BOOL	I, Q, M, L, D	Instruction was executed successfully.
BUSY	Output	BOOL	I, Q, M, L, D	Execution of instruction not yet complete.
ERROR	Output	BOOL	I, Q, M, L, D	• 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.
STATUS	Output	WORD	I, Q, M, L, D	Status parameter The parameter is only set for the duration of one call. To display the status, you should therefore copy the STATUS parameter to a free data area.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter STATUS

Error code* (W#16#...)	Description
0	No errors.
2	Warning: Data log file has already been opened by this application.
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8070	Complete internal instance memory is assigned.
8090	There are inconsistencies between the data log definition and existing data log data.
8091	A data type other than String was used at the NAME parameter.
8092	Data log does not exist.
80A0	Data types inconsistent. The data log at the ID parameter uses different data types than specified in the DATA parameter.
80B4	The memory card is write-protected.
80C1	Too many files open.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

Data logging - Overview (Page 4774)

DataLogClear: Empty data log

Description

The "DataLogClear" instruction deletes all data records in an existing data log. The instruction does not delete the optional header of the CSV file (see the description of the HEADER parameter of the instruction "DataLogCreate (Page 4777)").

You use the ID parameter to select the data log whose data records are to be deleted.

Requirement

Before you can delete data records, the data log must be open (see "DataLogOpen (Page 4784) instruction").

Parameters

The following table shows the parameters of the "DataLogClear" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, L, D, T, C or constant (T and C are only available in LAD and FBD with S7-1500)	Execution of the instruction upon a rising edge.
ID	InOut	DWORD	I, Q, M, D, L	Object ID of the data log
DONE	Output	BOOL	I, Q, M, D, L	Instruction was executed successfully.
BUSY	Output	BOOL	I, Q, M, D, L	Execution of the instruction not yet complete.
ERROR	Output	BOOL	I, Q, M, D, L	• 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output at the STATUS parameter.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter The parameter is only set for the duration of one call. To display the status, you should therefore copy the STATUS parameter to a free data area.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter STATUS

Error code* (W#16#...)	Explanation
0000	No error.
7000	No job processing active.

Error code* (W#16#...)	Explanation
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8080	The data log file selected with the ID parameter cannot be processed with the "DataLogClear" instruction.
8092	Data log does not exist.
80B0	Data log is not open.
80B4	The memory card is write-protected.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

Data logging - Overview (Page 4774)

DataLogWrite: Write data log

Description

The instruction "DataLogWrite" is used to write a data record to an existing data log. The ID parameter is used to select the data log to which the data record is to be written. To create a new data record, the data log must be open. The instruction creates a new data record in the format that was specified in the DATA parameter when the data log was created.

Before calling the "DataLogWrite" instruction, transfer the data to the tag that you interconnected at the DATA parameter of the "DataLogCreate" instruction. When the "DataLogWrite" instruction is executed, the transferred data is copied to the data log.

NOTICE
Data log data loss when the power supply to the CPU is interrupted If the power supply is interrupted during execution of the "DataLogWrite" instruction, the data record to be transferred is lost.

Parameters

The following table shows the parameters of the "DataLogWrite" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, L, D, T, C or constant (T and C are only available in LAD and FBD with S7-1500)	Execution of the instruction upon a rising edge.
ID	InOut	DWORD	I, Q, M, L, D	Object ID of the data log
DONE	Output	BOOL	I, Q, M, L, D	Instruction was executed successfully.
BUSY	Output	BOOL	I, Q, M, L, D	Execution of the instruction not yet complete.
ERROR	Output	BOOL	I, Q, M, L, D	• 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.
STATUS	Output	WORD	I, Q, M, L, D	Status parameter The parameter is only set for the duration of one call. To display the status, you should therefore copy the STATUS parameter to a free data area.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter STATUS (S7-1200)

Error code* (W#16#...)	Description
0	No errors
0001	Last possible data record created at the end of the file. Creating another data record will overwrite an older data record.
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8070	Complete internal instance memory is assigned.
8092	Data log does not exist.
80B0	Data log is not open.
80B3	Insufficient memory space on the memory card.
80B4	The memory card is write-protected.
80C0	Data log is locked.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

Parameter STATUS (S7-1500)

Error code* (W#16#...)	Description
0	No errors
0001	Last possible data record created at the end of the file. Creating another data record will overwrite an older data record.
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
8070	Complete internal instance memory is assigned.
8092	Data log does not exist.
80B0	Data log is not open.
80B3	Insufficient memory space on the memory card.
80B4	The memory card is write-protected.
80C0	Data log is locked.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

Data logging - Overview (Page 4774)

DataLogClose: Close data log

Description

You use the "DataLogClose" instruction to close an open data log. You select the data log using the ID parameter.

Note

Closing data logs automatically

The data log is closed automatically when the CPU goes to STOP or if there is a restart.

Parameters

The following table shows the parameters of the "DataLogClose" instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, L, D, T, C or constant (T and C are only available in LAD and FBD with S7-1500)	Execute function on a rising edge.
ID	InOut	DWORD	I, Q, M, L, D	Object ID of the data log
DONE	Output	BOOL	I, Q, M, L, D	Instruction was executed successfully.
BUSY	Output	BOOL	I, Q, M, L, D	Execution of the instruction not yet complete.
ERROR	Output	BOOL	I, Q, M, L, D	• 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.
STATUS	Output	WORD	I, Q, M, L, D	Status parameter The parameter is only set for the duration of one call. To display the status, you should therefore copy the STATUS parameter to a free data area.

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

STATUS (S7-1200) parameter

Error code* (W#16#...)	Description
0	No errors
1	Data log is not open
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8092	Data log does not exist.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

STATUS (S7-1500) parameter

Error code* (W#16#...)	Description
0	No errors
1	Data log is not open

Error code* (W#16#...)	Description
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
8070	Complete internal instance memory is assigned.
8092	Data log does not exist.
80B4	The memory card is write-protected.
80C0	Access currently not possible.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

DataLogDelete: Delete data log

Description

You use the "DataLogDelete" instruction to delete a data log file from the memory card. The data log and the data records it contains can only be deleted if it was created with the "DataLogCreate" or "DataLogNewFile" instruction.

Parameters

The following table shows the parameters of the "DataLogDelete" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, L, D, T, C or constant (T and C are only available in LAD and FBD with S7-1500)	Execution of the instruction upon a rising edge.
NAME	Input	VARIANT	L, D	File name of the data log
DELFILE	Input	BOOL	I, Q, M, D, L	• 0: Data log is retained. • 1: Data log is deleted. Detailed information is output at the STATUS parameter.
ID	InOut	DWORD	I, Q, M, D, L	Object ID of the data log
DONE	Output	BOOL	I, Q, M, D, L	Instruction was executed successfully.
BUSY	Output	BOOL	I, Q, M, D, L	Deletion of the data log is not yet complete.

Parameter	Declaration	Data type	Memory area	Description
ERROR	Output	BOOL	I, Q, M, D, L	• 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output at the STATUS parameter.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter The parameter is only set for the duration of one call. To display the status, you should therefore copy the STATUS parameter to a free data area.

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

NAME and ID parameters

Select the data log to be deleted using the NAME and ID parameters. The ID parameter is evaluated first. If there is a data log with the relevant ID, the NAME parameter will not be evaluated. If the value "0" is used at the ID parameter, a value with the data type STRING must be used at the NAME parameter.

RET_VAL parameter

Error code* (W#16#...)	Explanation
0	No error.
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8091	A data type other than STRING is being used at the NAME parameter.
8092	Data log does not exist.
80A2	Write error
80B4	The memory card is write-protected.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

DataLogCreate: Create data log (Page 4777)

DataLogNewFile: Data log in new file

DataLogNewFile: Data log in new file

Description

You use the "DataLogNewFile" instruction to create a new data log with the same properties as an existing data log. This allows the contents of an existing data log to be retained.

When called, the instruction creates a new data log on the memory card or in the internal load memory with the name defined at the NAME parameter. You use the ID parameter to specify the ID of the old data log whose properties you want to apply to the new data log. The ID of the new data log is then output via the ID parameter.

You specify the file size of the new data log with the RECORDS parameter of the instruction.

Once the new data log is created, it is opened automatically. This means that data can be written.

Parameters

The following table shows the parameters of the "DataLogNewFile" instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, L, D or constant	Execution of the instruction upon a rising edge.
RECORDS	Input	UDInt	I, Q, M, L, D or constant	Number of data records in the data log.
NAME	Input	VARIANT	L, D	File name of the new data log.
ID	InOut	DWORD	I, Q, M, L, D	Object ID of the data log • In: ID of the existing data log • Out: ID of the new data log
DONE	Output	BOOL	I, Q, M, L, D	Instruction was executed successfully.
BUSY	Output	BOOL	I, Q, M, L, D	Execution of the instruction not yet complete.
ERROR	Output	BOOL	I, Q, M, L, D	• 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.
STATUS	Output	WORD	I, Q, M, L, D	Status parameter The parameter is only set for the duration of one call. To display the status, you should therefore copy the STATUS parameter to a free data area.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter STATUS

Error code* (W#16#...)	Description
0	No errors.
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8070	Complete internal instance memory is assigned.
8090	Invalid file name.
8091	Path does not exist.
8092	Source data log does not exist.
8093	New data log already exists.
8097	File length exceeds the file system limit.
80B3	Load memory not sufficient.
80B4	The memory card is write-protected.
80C1	Too many files open.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

DataLogNewFile: Data log in new file

Description

You use the "DataLogNewFile" instruction to create a new data log with the same properties as an existing data log. This allows the contents of an existing data log to be retained.

When called, the instruction creates a new data log on the memory card or in the internal load memory with the name defined at the NAME parameter. You use the ID parameter to specify the ID of the old data log whose properties you want to apply to the new data log. The ID of the new data log is then output via the ID parameter.

You specify the file size of the new data log with the RECORDS parameter of the instruction.

The parameter DATA enables a consistency check between the new data log to be created and the definition of the data log of the instruction "DataLogCreate (Page 4777)". The consistency check can only be executed if the data log was created with the instruction "DataLogCreate (Page 4777)":

- If you are using the same pointer at the DATA parameter as at the DATA parameter of the "DataLogCreate (Page 4777)" instruction, a check is performed to determine whether the data types match. If the data types do not match, the error code W#16#80A0 is output at the STATUS parameter.
- If the data log to be opened was not created with "DataLogCreate (Page 4777)", a consistency check is not possible. In this case, enter the value "NULL" in the DATA parameter.

Once the new data log is created, it is opened automatically. This means that data can be written.

Parameters

The following table shows the parameters of the "DataLogNewFile" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, L, D, T, C or constant (T and C are only available in LAD and FBD with S7-1500)	Execution of the instruction upon a rising edge.
RECORDS	Input	UDInt	I, Q, M, L, D or constant	Number of data records in the data log.
NAME	Input	VARIANT	L, D	File name of the new data log.
ID	InOut	DWORD	I, Q, M, L, D	Object ID of the data log • In: ID of the existing data log • Out: ID of the new data log
DATA	InOut	VARIANT	L, D	Data type for consistency check
DONE	Output	BOOL	I, Q, M, L, D	Instruction was executed successfully.
BUSY	Output	BOOL	I, Q, M, L, D	Execution of the instruction not yet complete.
ERROR	Output	BOOL	I, Q, M, L, D	• 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.
STATUS	Output	WORD	I, Q, M, L, D	Status parameter The parameter is only set for the duration of one call. To display the status, you should therefore copy the STATUS parameter to a free data area.

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

STATUS parameter

Error code* (W#16#...)	Description
0	No errors.
7000	No job processing active.
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
8070	Complete internal instance memory is assigned.

Error code* (W#16#...)	Description
8090	• Invalid file name. • Data types inconsistent. The data log at the ID parameter uses different data types than specified in the DATA parameter.
8091	Path does not exist.
8092	Source data log does not exist.
8093	New data log already exists.
8097	File length exceeds the file system limit.
80B3	Load memory not sufficient.
80B4	The memory card is write-protected.
80C0	Access currently not possible.
80C1	Too many files open.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

Data block functions

CREATE_DB: Create data block

Description

The instruction "CREATE_DB" is used to create a new data block in the load and/or work memory.

The instruction "CREATE_DB" does not change the checksum of the user program.

Number of the data block

The data block created is assigned a number from the range defined at the LOW_LIMIT (low limit) and UP_LIMIT (high limit) parameters. "CREATE_DB" assigns the smallest possible number from the specified range to the DB. You cannot assign the numbers of the DBs already contained in the user program.

To create a DB with a specific number, enter the same number for the high and low limit of the range to be specified. If a DB with the same number already exists in the work memory and/ or load memory, or if the DB exists as a copied version, the instruction will be terminated and an error message will be generated at the RET_VAL parameter.

Start values of the data block

Using the SRCBLK parameter, you can define start values for the DB to be created. The SRCBLK parameter is a pointer to a DB or a DB area from which you apply the start values.

The DB addressed at the SRCBLK parameter must have been generated with standard access ("Optimized block access" attribute disabled).

- If the area specified at the SRCBLK parameter is larger than the DB generated, the values up to the length of the DB generated will be applied as start values.
- If the area specified at the SRCBLK parameter is smaller than the DB generated, the remaining values will be filled with "0".

To ensure data consistency, you must not change this data area while "CREATE_DB" is being executed (which means as long as the BUSY parameter has the value TRUE).

Parameters

The following table shows the parameters of the "CREATE_DB" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ= 1: Request to create the DB
LOW_LIMIT	Input	UINT	I, Q, M, D, L or constant	Low limit of the area ### used by "CREATE_DB" to assign a number to your DB 60000)
UP_LIMIT	Input	UINT	I, Q, M, D, L or constant	High limit of the area ### used by "CREATE_DB" to assign a number to your DB (largest possible DB number: 60999)
COUNT	Input	UDINT	I, Q, M, D, L or constant	The count value specifies the number of bytes which you want to reserve for the DB generated. The number of bytes must be an even number. The maximum length is 65534 bytes.
ATTRIB	Input	BYTE	I, Q, M, D, L or constant	The first 4 bits of the byte at the ATTRIB parameter define the properties of the data block:
				• Bit 0 = 0: DB only in the work memory • Bit 0 = 1: DB only in the load memory
				• Bit 1 = 0: DB is not write-protected. • Bit 1 = 1: DB is write-protected
				• Bit 2 = 0: DB is retentive (only for DBs generated in the load memory) • Bit 2 = 1: DB is not retentive
				• Bit 3= 0: DB generated either in load or in work memory • Bit 3= 1: DB generated both in load and in work memory To ensure compatibility with STEP7 V5.x, bits 1 and 3 must be used in combination:

Bit 0	Bit 3	DB generation
0	0	In work memory only
1	0	In load memory only
0	1	Work and load memory
1	1	Work and load memory

• Bit 4 = 0 - No start values specified (input values at the SRCBLK parameter will be ignored). • Bit 4 = 1 - Specify start values (values correspond to the DB addressed by the SRCBLK parameter).

Parameter	Declaration	Data type	Memory area	Description
SRCBLK	Input	VARIANT	D	Pointer to the data block whose values will be used to initialize the data block to be generated.
RET_VAL	Return	INT	I, Q, M, D, L	Error information
BUSY	Output	BOOL	I, Q, M, D, L	BUSY= 1: The process is not yet complete.
DB_NUM	Output	DB_DYN (UINT)	I, Q, M, D, L	Number of the DB created.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	No error
0081	The target range is greater than the source range. The complete source range is written to the target range. The remaining bytes of the target range are filled with 0.
7000	First call with REQ = 0: No data transfer active; BUSY has the value "0".
7001	First call with REQ = 1: Data transfer triggered; BUSY has the value "1".
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".
8081	The source range is larger than the target range. The complete target range is written. The remaining bytes of the source range are ignored.
8092	The "Create data block" function is currently unavailable because • The "Compress user memory" function is currently active. • The maximum number of blocks on your CPU has already been reached.
8093	No data block or a data block that is not in the work memory is specified for the SRCBLK parameter.
8094	A not yet supported attribute was specified for the ATTRIBparameter.
80A1	DB number error: • The number is "0" • Low limit > high limit
80A2	DB length error: • The length is "0" • The length is an odd number • The length is greater than permitted by the CPU
80A3	The data block at the SRCBLK parameter was not created with standard access.
80B1	There is no DB number free.
80B2	Not enough work memory.
80B4	The memory card is write-protected.
80BB	Not enough load memory.
80C0	The destination is currently being processed by another instruction or a communication function.
80C1	A DB with this DB number is currently being deleted.

Error code* (W#16#...)	Description
80C3	The maximum number of simultaneously active "CREATE_DB" instructions has already been reached.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

READ_DBL: Read from data block in the load memory

Description

With the instruction you copy a DB or an area of a DB in load memory (Micro Memory Card) to the data area of a destination DB. The destination DB must be relevant for execution; that is, it must not be created with the attribute UNLINKED. The content of the load memory is not changed during the copy process.

To ensure data consistency, you must not change the target range while "READ_DBL" is being executed (i.e., as long as the BUSY parameter has the value TRUE).

The following restrictions apply to the SRCBLK and DSTBLK parameters (source and destination blocks):

- You must be able to divide the length of the VARIANT pointer by eight.
- For a VARIANT pointer of type STRING, the length must be equal to 1.
- The source and destination block must have been created with the same block access, i.e. both must use either the access type "Optimized" or "Standard".

Note

""READ_DBL" is processed asynchronously. Therefore, it is not suitable for frequent (or cyclical) reading of tags in the load memory.

Once started, a job is always completed. If the maximum number of simultaneously active "READ_DBL" instructions is reached and you call "READ_DBL" once again at this time in a priority class having higher priority, error code W#16#80C3 will be returned. Consequently it does not make sense to restart the high-priority job right away.

Functional description

The "READ_DBL" instruction works asynchronously, that is, its execution extends over multiple calls. You start the job by calling "READ_DBL" with REQ = 1.

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 3150)

Parameters

The following table shows the parameters of the "READ_DBL" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = 1: Read request
SRCBLK	Input	VARIANT	D	Pointer to data block in the load memory that is to be read from
RET_VAL	Return	INT	I, Q, M, D, L	Error information
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The reading process is not yet complete.
DSTBLK	Output	VARIANT	D	Pointer to the data block in the work memory that is to be written to

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	No error
0081	The target range is greater than the source range. The source range is written completely to the target range; the remaining bytes of the target range will not be changed.
7000	First call with REQ = 0: No data transfer active; BUSY has the value "0".
7001	First call with REQ=1: Data transfer triggered; BUSY has the value "1".
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".
8081	The source range is larger than the target range. The complete target range is written. The remaining bytes of the source range are ignored.
8082	Destination DB type different from source DB type (optimized/standard access).
8093	No data block or a data block that is not in the work memory is specified for the DSTBLK parameter.
80B1	No data block is specified for the SRCBLK parameter, or the data block specified there is not a load memory object.
80B4	DB with F-attribute must not be read.
80C3	The maximum number of simultaneously active "READ_DBL" instructions has already been reached.
80C0	The destination DB is currently being processed by another instruction or a communication function.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

GET_ERR_ID: Get error ID locally (Page 4282)

WRIT_DBL: Write to data block in the load memory

Description

The instruction "WRIT_DBL" is used to transfer the contents of a DB or a DB area from the work memory to a DB or a DB area in the load memory (Micro Memory Card). The source DB must be relevant for execution, which means it must not be created with the attribute UNLINKED.

To ensure data consistency, it is not permitted to change the source range while "WRIT_DBL" is being executed (i.e., as long as the BUSY parameter has the value TRUE).

The following restrictions apply to the SRCBLK and DSTBLK parameters (source and destination blocks):

- For a VARIANT pointer of type BOOL, the length must be divisible by 8.
- For a VARIANT pointer of type STRING, the length must be equal to 1.
- The source and destination block must have been created with the same block access, i.e. both must use either the access type "Optimized" or "Standard - compatible with S7-300/400".

The "WRIT_DBL" instruction does not change the checksum of the user program if you write a DB that was created using an instruction. However, when a loaded DB is written, the first entry in this DB changes the checksum of the user program.

Note

"WRIT_DBL" is not suitable for frequent (or cyclical) writing of tags in the load memory. This is because the Micro Memory Card technology limits the number of write accesses that can be made to a Micro Memory Card.

Functional description

The "WRIT_DBL" instruction works asynchronously, that is, its execution extends over multiple calls. You start the job by calling "WRIT_DBL" with REQ=1.

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 3150).

Parameters

The following table shows the parameters of the instruction "WRIT_DBL":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = 1: Write request
SRCBLK	Input	VARIANT	D	Pointer to the DB in the work memory that is to be read from
RET_VAL	Return	INT	I, Q, M, D, L	Error information

Parameter	Declaration	Data type	Memory area	Description
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The writing process is not yet completed.
DSTBLK	Output	VARIANT	D	Pointer to the data block in the load memory that is to be written to

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code* (W#16#...)	Description
0000	No error
0081	The target range is greater than the source range. The source range is written completely to the target range; the remaining bytes of the target range will not be changed.
7000	First call with REQ = 0: No data transfer active; BUSY has the value "0".
7001	First call with REQ = 1: Data transfer triggered; BUSY has the value "1".
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".
8081	The source range is larger than the target range. The complete target range is written. The remaining bytes of the source range are ignored.
8082	Destination DB type different from source DB type (optimized/non-optimized access).
8092	Incorrect operating mode: While "WRIT_DBL" was active, the CPU went into STOP mode. This error code is supplied at the next transition to RUN. Call "WRIT_DBL" again.
8093	No data block or a data block that is not in the work memory is specified for the SRCBLK parameter.
80B1	No data block has been specified for the DSTBLK parameter, or the data block specified there is not a load memory object.
80B4	• DB with F-attribute must not be read. • The memory card is write-protected.
80BB	Insufficient load memory available.
80C3	The maximum number of simultaneously active "WRIT_DBL" instructions has already been reached.
80C0	The destination DB is currently being processed by another instruction or a communication function.
General error codes	See also: GET_ERR_ID: Get error ID locally (Page 3392)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

ATTR_DB: Read data block attribute

Description

You use the instruction "ATTR_DB" to obtain information about a data block (DB) located in the work memory of the CPU. The instruction determines the attributes set at the ATTRIB parameter for the DB selected.

The length cannot be read out for data blocks with optimized access, the DB_LENGH parameter contains the length "0" for DBs with optimized access.

Data blocks for Motion Control cannot be read out with the "ATTR_DB" instruction. The error code 80B2 is output for this.

Parameters

The following table shows the parameters of the "ATTR_DB" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ = 1: Read request for block attributes
DB_NUMBER	Input	DB_ANY (UINT)	I, Q, M, D, L or constant	Number of the DB to be tested
RET_VAL	Output	INT	I, Q, M, D, L	Error information
DB_LENGTH	Output	UDINT	I, Q, M, D, L	Number of data bytes which the selected DB contains.
ATTRIB	Output	BYTE	I, Q, M, D, L	DB properties: • Bit 0*= 0: LINKED - The DB only exists in the work memory • Bit 0*= 1: UNLINKED - The DB only exists in the load memory • Bit 1 = 0: READ_ONLY - The DB is not write-protected. • Bit 1 = 1: READ_ONLY - The DB is write-protected. • Bit 2 = 0: RETAIN - The DB is retentive. • Bit 2 = 1: NON_RETAIN - The DB is not retentive. • Bit 3*= 0: The DB exists either in the load or the work memory • Bit 3*= 1: The DB is generated in both the load and the work memory
* The relationship between bit 0 and bit 3 is explained in the parameters of the instruction "CREATE_DB: Create data block (Page 4798)".				

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0000	No error occurred.
80A1	Error in input parameter DB_NUMBER: the actual parameter selected • Is "0" • Is greater than the maximum permitted DB number for the CPU used.
80B1	The DB with the specified number does not exist on the CPU.
80B2	Data blocks of Motion Control technology objects cannot be read out with the "ATTR_DB" instruction.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

DELETE_DB: Delete data block

Description

You use the instruction "DELETE_DB" to delete a data block (DB) that was created from the user program by calling the instruction "CREATE_DB (Page 4798)".

If the data block was not created with "CREATE_DB", the error code W#16#80B5 is output at the RET_VAL parameter.

The DB to be deleted may not be located in the current or in a lower priority class. Otherwise, the CPU starts OB 121 when the "DELETE_DB" instruction is called. If OB 121 is not present, the CPU switches to STOP mode. The instruction "DELETE_DB" can be interrupted by higher priority classes. If the instruction is called again there, then this second call will be aborted and W#16#8091 will be entered in RET_VAL .

Parameters

The following table shows the parameters of the "DELETE_DB" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	REQ =1: Request to delete the DB with the number in parameter DB_NUMBER
DB_NUMBER	Input	UINT	I, Q, M, D, L or constant	Number of the DB to be deleted
RET_VAL	Output	INT	I, Q, M, D, L	Error information (see "RET_VAL parameter")
BUSY	Output	BOOL	I, Q, M, D, L	BUSY= 1: The process is not yet complete.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0000	No error occurred.
8091	"DELETE_DB" calls were nested and the maximum nesting level of the CPU used was exceeded.
8092	The instruction can currently not be executed because: • The "Compress user memory" function is currently active. • You are copying the DB to be deleted from the CPU to an offline project.
80A1	Error in input parameter DB_NUMBER: • The value at the parameter is "0". • The value at the parameter is greater than the maximum permitted DB number for the CPU used.
80B1	The DB with the specified number does not exist on the CPU.
80B4	The DB cannot be deleted because the memory card of the CPU is write-protected.
80B5	The DB was not created using "CREATE_DB".
80BB	Not enough load memory.
80C1	The "Delete a DB" function cannot be executed at this time due to a temporary resource bottleneck.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 3392)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

Addressing

Instructions for address conversion

Description

There are different options for addressing a module (IO address, hardware identifier, slot).

You can convert the address data with the following instructions:

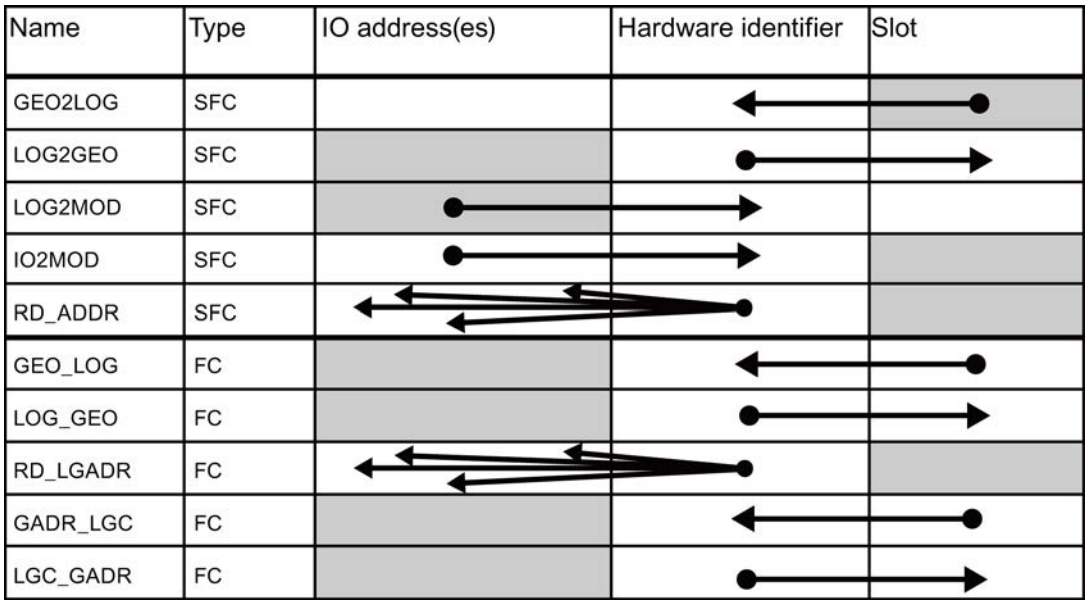
- GEO2LOG: Determine hardware identifier from slot (Page 4808)
- LOG2GEO: Determine slot from hardware identifier (Page 4810)
- LOG2MOD: Determine the hardware identifier from addressing of STEP 7 V5.5 SPx (Page 4811)
- IO2MOD: Determine hardware identifier from an IO address (Page 4812)
- RD_ADDR: Determine IO addresses from the hardware identifier (Page 4813)

The following instructions are supported in addition for migrated projects:

- GEO_LOG: Determine hardware identifier from slot (Page 4816)
- LOG_GEO: Determine slot from hardware identifier (Page 4817)
- RD_LGADR: Determine IO addresses from the hardware identifier (Page 4818)
- GADR_LGC: Determine hardware identifier from slot and offset in the user data address area (Page 4820)
- LGC_GADR: Determine slot from hardware identifier (Page 4821)

Type of address conversion

The figure below shows which instruction runs which address conversion.



GEO2LOG: Determine hardware identifier from slot

Description

You use the "GEO2LOG" instruction to determine hardware identifier based on slot information that you define using the system data type GEOADDR.

Depending on the type of hardware you define at the parameter HWTYPE the following information is evaluated from the other parameters GEOADDR:

- With HWTYPE = 1 (PROFINET IO system):
  - Only IOSYSTEM is evaluated. The other parameters of GEOADDR are not taken into consideration.
  - The hardware identifier of the PROFINET IO system is output.
- With HWTYPE = 2 (PROFINET IO device):
  - IOSYSTEM and STATION are evaluated. The other parameters of GEOADDR are not taken into consideration.
  - The hardware identifier of the PROFINET IO device is output.
- With HWTYPE = 3 (rack):
  - Only IOSYSTEM and STATION are evaluated. The other parameters of GEOADDR are not taken into consideration.
  - The hardware identifier of the rack is output.
- With HWTYPE = 4 (module):
  - IOSYSTEM, STATION and SLOT are evaluated. The SUBSLOT parameter of GEOADDR is not taken into consideration.
  - The hardware identifier of the module is output.
- With HWTYPE = 5 (submodule):
  - All parameters of GEOADDR are evaluated.
  - The hardware identifier of the submodule is output.

The AREA parameter of the GEOADDR system data type is not evaluated.

Parameter

The following table shows the parameters of the "GEO2LOG" instruction:

Parameter	Declaration	Data type	Memory area	Description
GEOADDR	Input	VARIANT	D, L	Pointer to the structure of the GEOADDR system data type. The system data type contains the slot information from which the hardware ID is determined. See also: System data type GEOADDR (Page 4815)
RET_VAL	Return	INT	I, Q, M, D, L	Output of error information.
LADDR	Output	HW_ANY	I, Q, M, D, L	Hardware identifier of the assembly or the module. The number is automatically assigned and is stored in the properties in the hardware configuration.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0	No error occurred.
8091	Invalid value for in GEOADDR for HWTYPE.
8094	Invalid value for in GEOADDR for IOSYSTEM.
8095	Invalid value for in GEOADDR for STATION.
8096	Invalid value for in GEOADDR for SLOT.
8097	Invalid value for in GEOADDR for SUBSLOT.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

Instructions for address conversion (Page 4807)

LOG2GEO: Determine slot from hardware identifier

Description

You use the "LOG2GEO" instruction to determine the module slot belonging to a hardware identifier.

Parameters

The following table shows the parameters of the "LOG2GEO" instruction:

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_ANY	I, Q, M, D, L or constant	Hardware identifier of the module whose slot you want to find. The hardware ID is assigned automatically and is stored in the properties of the module in the hardware configuration and the system constants.
RET_VAL	Output	INT	I, Q, M, D, L	Output of error information.
GEOADDR	InOut	VARIANT	D	Pointer to the GEOADDR system data type. The slot information is written in the system data type GEOADDR. See also: System data type GEOADDR (Page 4815)

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0	No error occurred.
8090	The address specified at the LADDR parameter is invalid.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

LOG2MOD: Determine the hardware identifier from addressing of STEP 7 V5.5 SPx

Description

You use the "LOG2MOD" instruction to determine the hardware identifier for an IO (sub)module from the addressing of STEP 7 5.5 SPx (IO data address or diagnostic address).

The hardware identifier is used at the LADDR input parameter for addressing of various instructions. You can convert the addressing parameters from STEP 7 5.5 SPx by calling "LOG2MOD" beforehand.

Parameter

The following table shows the parameters of the "LOG2MOD" instruction:

Parameter	Declaration	Data type	Memory area	Description
IOID	Input	BYTE	I, Q, M, D, L or constant	Identifier of the address area as in STEP 7 5.5 SPx: • B#16#00: Bit15 of ADDR specifies whether an input (Bit15=0) or output address (Bit15=1) exists. • B#16#54= Peripheral input (PI) • B#16#55= Peripheral output (PQ)
ADDR	Input	WORD	I, Q, M, D, L or constant	Logical address of the IO data of the module as offset (corresponding addressing in STEP 7 5.5 SPx) or diagnostic address.
RET_VAL	Return	INT	I, Q, M, D, L	Error code of the instruction.
HWID	Output	HW_IO	I, Q, M, D, L	Determined hardware identifier (logical address) of the IO (sub)module.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

RET_VAL parameter

Error code* (W#16#...)	Explanation
0	No error occurred.
8093	• Specified address is not used by any hardware components. • Specified value at IOID parameter is invalid.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

IO2MOD: Determine hardware identifier from an IO address

Description

The "IO2MOD" instruction determines the hardware identifier of the module from an IO address (I, Q, PI, PQ) of a (sub)module.

Enter the IO address at the ADDR parameter. If a series of IO addresses is used at this parameter, only the first address is evaluated to determine the hardware identifier. If the first address is correctly specified, the length for the address specification at the ADDR is of no significance. If an address area is used that encompasses several modules or non-used addresses, the hardware identifier of the first module can also be determined.

If no IO address of a sub(module) is specified at the ADDR parameter, the error code 8090 is output at the RET_VAL parameter.

Note

Input of IO address in SCL

You cannot program using the IO access ID "%QWx:P" in SCL. In this case, use the symbolic tag name or the absolute address in the process image.

Parameters

The following table shows the parameters of the "IO2MOD" instruction:

Parameter	Declaration	Data type	Memory area	Description
ADDR	Input	VARIANT	I, Q, M, D, L	IO address (I, Q, PI, PQ) within a (sub)module. Make sure that slice access is not used for the parameter ADDR. If this is the case, incorrect values are output at the LADDR parameter.
RET_VAL	Return	INT	I, Q, M, D, L	Error code of the instruction.
LADDR	Output	HW_IO	I, Q, M, D, L	Determined hardware identifier (logical address) of the IO (sub)module.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

RET_VAL parameter

Error code* (W#16#...)	Explanation
0	No error occurred.
8090	IO address specified at ADDR parameter is not used by any hardware component.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

RD_ADDR: Determine IO addresses from the hardware identifier

Description

The "RD_ADDR" instruction determines the length and the start address of the inputs or outputs based on the hardware identifier of a sub(module).

- Use the LADDR parameter to select the input or output module based on the hardware identifier.
- The following output parameters are used depending on whether it is an input module or output module:
  - In the case of an input module the determined values are output at the PIADDR and PICOUNT parameters.
  - In the case of an output module the determined values are output at the PQADDR and PQCOUNT parameters.

- The PIADDR and PQADDR parameters each contain the start address of the I/O addresses of the module.
- The PICOUNT and PQCOUNT parameters each contain the number of bytes of the inputs our outputs (1 byte for 8 inputs/outputs, 2 bytes for 16 inputs/outputs).

Note

Addressing packed modules

If packed modules (not the first module of the packed module group) are addressed, the displayed data deviates from the hardware configuration. "0" is returned for PIADDR or PQADDR and PICOUNT or PQCOUNT. RET_VAL does not indicate an error (16#0000).

Parameters

The following table shows the parameters of the "RD_ADDR" instruction:

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_IO	I, Q, M, D, L or constant	Hardware identifier of the (sub)module.
RET_VAL	Return	INT	I, Q, M, D, L	Error code of the instruction.
PIADDR	Output	UDINT	I, Q, M, D, L	Start address of the input module.
PICOUNT	Output	UINT	I, Q, M, D, L	Number of bytes of the inputs.
PQADDR	Output	UDINT	I, Q, M, D, L	Start address of the output module.
PQCOUNT	Output	UINT	I, Q, M, D, L	Number of bytes of the outputs.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0	No error occurred.
8090	Hardware identifier of the module at the LADDR parameter is invalid.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

System data type GEOADDR

Geographical address

The system data type GEOADDR contains the geographical address of a module, i.e. the slot information.

- Geographical address for PROFINET IO
  For PROFINET IO the geographical address was composed of the ID of the PROFINET IO system, the device number, the slot number and the submodule (if a sub-module is used).
- Geographical address for PROFIBUS DP
  For PROFIBUS DP, the geographical address consists of the ID of the DP master system, the station number and the slot number.

The slot information of the modules can be found in the hardware configuration of each module.

System data type GEOADDR

The structure GEOADDR is automatically created if you enter "GEOADDR" as the data type in a data block.

The system data type GEOADDR has the following structure:

Parameter name		Data type	Description
GEOADDR		STRUCT	
	HWTYPE	UINT	Hardware type: • 1: IO system (PROFINET/PROFIBUS) • 2: IO device/DP slave • 3: Rack • 4: Module • 5: Submodule If a hardware type is not supported by the instruction, a HWTYPE "0" is output.
	AREA	UINT	Area ID: • 0 = CPU • 1 = PROFINET IO • 2 = PROFIBUS DP • 3 = AS-i
	IOSYSTEM	UINT	PROFINET IO system (0=central unit in the rack)
	STATION	UINT	• Number of the rack if area identifier AREA = 0 (central module). • Station number if area identifier AREA > 0.
	SLOT	UINT	Slot number
	SUBSLOT	UINT	Number of the submodule. This parameter has the value "0" if no submodule is available or can be plugged.

Other

GEO_LOG: Determine hardware identifier from slot

Description

The corresponding module slot of a signal module is known. Use the "GEO_LOG" instruction to determine the corresponding hardware identifier of the module from this.

Parameters

The following table shows the parameters of the instruction "GEO_LOG":

Parameter	Declaration	Data type	Memory area	Description
MASTER	Input	INT	I, Q, M, D, L or constant	Area ID: • 0, if the slot is located in a centralized configuration. • 1 to 32: DP master system ID of the associated field device if the slot is located in a field device on PROFIBUS • 100 to 115: PROFINET IO system ID of the associated field device if the slot is located in a field device on PROFINET
STATION	Input	INT	I, Q, M, D, L or constant	• If MASTER = 0: Number of the rack • If MASTER > 0: Station number of the field device
SLOT	Input	INT	I, Q, M, D, L or constant	Slot number
SUBSLOT	Input	INT	I, Q, M, D, L or constant	The parameter SUBSLOT is not evaluated by the instruction.
RET_VAL	Return	INT	I, Q, M, D, L	Error information
LADDR	Output	HW_IO	I, Q, M, D, L	Hardware identifier of the module

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0000	No error occurred.
8094	No subnet was configured with the specified SUBNETID .
8095	Illegal value for STATION parameter
8096	Illegal value for SLOT parameter

Error code* (W#16#...)	Explanation
General error information	See also: GET_ERR_ID: Get error ID locally (Page 3392)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

Instructions for address conversion (Page 4807)

LOG_GEO: Determine slot from hardware identifier

Description

You use the "LOG_GEO" instruction to determine the module slot belonging to a hardware identifier.

Parameters

The following table shows the parameters of the instruction "LOG_GEO":

Parameter	Declaration	Data type	Memory area	Description
LADDR	Input	HW_IO	I, Q, M, D, L or constant	Hardware identifier of the module for which the slot is to be determined.
RET_VAL	Return	INT	I, Q, M, D, L	Error information
AREA	Output	INT	I, Q, M, D, L	Area ID: indicates how the remaining output parameters are to be interpreted: • 0: central device • 2: PROFIBUS DP / PROFINET IO
MASTER	Output	INT	I, Q, M, D, L	With AREA = 0: • 0: If the slot is located in one of the racks (central device). With AREA = 2: • 1 to 32: DP master system ID of the associated field device if the slot is located in a field device on PROFIBUS • 100 to 115: PROFINET IO system ID of the associated field device if the slot is located in a field device on PROFINET
STATION	Output	INT	I, Q, M, D, L	• With MASTER = 0: Number of the rack • With MASTER > 0: Station number of the field device

Parameter	Declaration	Data type	Memory area	Description
SLOT	Output	INT	I, Q, M, D, L	Slot number
SUBSLOT	Output	INT	I, Q, M, D, L	The SUBSLOT parameter is not output by the instruction (always "0").
OFFSET	Output	INT	I, Q, M, D, L	The OFFSET parameter is not output by the instruction (always "0").

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0000	No error occurred.
8090	Specified logical address invalid
General error information	See also: GET_ERR_ID: Get error ID locally (Page 3392)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

Instructions for address conversion (Page 4807)

RD_LGADR: Determine IO addresses from the hardware identifier

Description

Based on the hardware identifier, you can determine the logical addresses of a module, a central submodule or a submodule for PNIO with the "RD_LGADR" instruction.

- You can specify the hardware ID of the submodule at the LADDR parameter.
- The addresses are written to the PEADDR and PAADDR parameters is ascending order.
  - For an input module, only the PEADDR parameter is written. For an output module, the PAADDR parameter is written.
  - An Array of WORD is used to store the addresses in each case.
- The number of addresses is output at the PECOUNT parameter (for an input module) and PACOUNT parameter (for an output module).

Parameters

The following table shows the parameters of the instruction "RD_LGADR":

Parameter	Declaration	Data type	Memory area	Description
IOID	Input	BYTE	I, Q, M, D, L or constant	Address area identifier: • B#16#54 = Peripheral input (PI) • B#16#55 = Peripheral output (PQ)
LADDR	Input	HW_ANY	I, Q, M, D, L or constant	Hardware identifier of the module or the submodule.
RET_VAL	Return	INT	I, Q, M, D, L	Error information
PEADDR	Output	ANY	I, Q, M, D, L	Field for the PI addresses with the Array of WORD data type
PECOUNT	Output	INT	I, Q, M, D, L	Number of returned PI addresses
PAADDR	Output	ANY	I, Q, M, D, L	Field for the PQ addresses with the Array of WORD data type
PACOUNT	Output	INT	I, Q, M, D, L	Number of returned PQ addresses

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

RET_VAL parameter

Error code (W#16#...)	Explanation
0000	No error occurred.
8090	Specified logical address invalid or illegal value for the IOID parameter.
80A0	Error in the output parameter PEADDR: The data type of the array elements is not WORD.
80A1	Error in the output parameter PAADDR: The data type of the array elements is not WORD.
80A2	Error in the output parameter PEADDR: The specified array could not accommodate all the logical addresses.
80A3	Error in the output parameter PAADDR: The specified array could not accommodate all the logical addresses.
General error information	See also: GET_ERR_ID: Get error ID locally (Page 3392)

See also

Switching display formats in the program status (Page 2980)

Instructions for address conversion (Page 4807)

GADR_LGC: Determine hardware identifier from slot and offset in the user data address area

Description

You use the "GADR_LGC" instruction to determine the hardware identifier of a signal module. The hardware identifier is determined from the module slot and the offset in the user data address area of the module.

If you use the "GADR_LGC" instruction on power modules or modules with packed addresses (ET 200S), then the diagnostics address will be returned.

Parameters

The following table shows the parameters of the instruction "GADR_LGC":

Parameter	Declaration	Data type	Memory area	Description
SUBNETID	Input	BYTE	I, Q, M, D, L or constant	Area ID: • 0: If the slot is located in the central module • 1 to 32: DP master system ID of the corresponding distributed I/O system if the slot is in a distributed I/O device. • 100 to 115: PROFINET IO system ID of the associated field device if the slot is located in a field device on PROFINET
RACK	Input	WORD	I, Q, M, D, L or constant	• Number of the rack, if area identifier is 0 • Device number of the distributed I/O device if the area identifier > 0
SLOT	Input	WORD	I, Q, M, D, L or constant	Slot no.
SUBSLOT	Input	BYTE	I, Q, M, D, L or constant	Sub-module slot (if no sub-module can be inserted, 0 must be entered here)
SUBADDR	Input	WORD	I, Q, M, D, L or constant	Offset in the user data address area of the module
RET_VAL	Return	INT	I, Q, M, D, L	Error information
IOID	Output	BYTE	I, Q, M, D, L	The IOID output parameter is not written (always "0").
LADDR	Output	HW_MODULE	I, Q, M, D, L	Hardware identifier of the module

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0000	No error occurred.
8093	Illegal value for SUBNETID parameter
8094	No subnet was configured with the specified SUBNETID .
8095	Illegal value for RACK parameter.
8096	Illegal value for SLOT parameter.
8097	Illegal value for SUBSLOT parameter.
8098	Illegal value for SUBADDR parameter.
8099	The slot is not configured.
809A	The sub-address of the selected slot is not configured (only possible with central I/O for CPU and IM).
General error information	See also: GET_ERR_ID: Get error ID locally (Page 3392)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

Instructions for address conversion (Page 4807)

LGC_GADR: Determine slot from hardware identifier

Description

You use the "LGC_GADR" instruction to determine the module slot belonging to a hardware identifier.

Note

The "LGC_GADR" instruction cannot be used on a module with packed addresses (ET 200S).

Parameters

The following table shows the parameters of the instruction "LGC_GADR":

Parameter	Declaration	Data type	Memory area	Description
IOID	Input	BYTE	I, Q, M, D, L or constant	Is not evaluated.
LADDR	Input	HW_MODULE	I, Q, M, D, L or constant	Hardware identifier of the module
RET_VAL	Return	INT	I, Q, M, D, L	Error information

Parameter	Declaration	Data type	Memory area	Description
AREA	Output	BYTE	I, Q, M, D, L	Area ID: indicates how the remaining output parameters are to be interpreted: • 0: Central module • 2: PROFIBUS DP
RACK	Output	WORD	I, Q, M, D, L	Rack number: • With central module AREA= 0): – Rack number • For PROFIBUS DP (AREA=2): – Low byte: Station number – High byte: DP master system ID
SLOT	Output	WORD	I, Q, M, D, L	Slot number: • With central module AREA= 0): – Slot number • For PROFIBUS DP (AREA=2): – Slot no. in the station
SUBADDR	Output	WORD	I, Q, M, D, L	Is not output (always "0").

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code* (W#16#...)	Explanation
0000	No error occurred.
8090	Specified logical address invalid or illegal value for the IOID parameter.
8093	This instruction is not permitted for the module selected by the parameters IOID and LADDR .
General error information	See also: GET_ERR_ID: Get error ID locally (Page 3392)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

Instructions for address conversion (Page 4807)

9.8.1.4 Technology

Motion Control

S7-1200 Motion Control

S7-1200 Motion Control V4

MC_Power

MC_Power: Enable, disable axis V4

Description

The Motion Control instruction "MC_Power" enables or disables an axis.

Requirements

- The positioning axis technology object has been configured correctly.
- There is no pending enable-inhibiting error.

Override response

Execution of "MC_Power" cannot be aborted by a Motion Control command.

Disabling the axis (input parameter "Enable" = FALSE) aborts all Motion Control commands for the associated technology object in accordance with the selected "StopMode".

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis	-	Axis technology object	
Enable	INPUT	BOOL	FALSE	TRUE	The axis is enabled.
				FALSE	All current jobs are interrupted in accordance with the "StopMode" configured. The axis is stopped and disabled.
StopMode	INPUT	INT	0	0	Emergency stop If a request to disable the axis is pending, the axis brakes at the configured emergency stop deceleration. The axis is disabled after reaching standstill.

Parameter	Declaration	Data type	Default value	Description	
				1	Immediate stop If a request to disable the axis is pending, this axis is disabled without deceleration. The pulse output is stopped immediately.
				2	Emergency stop with jerk control If a request to disable the axis is pending, the axis brakes at the configured emergency stop deceleration. If the jerk control is activated, the configured jerk is taken into account. The axis is disabled after reaching standstill.
Status	OUTPUT	BOOL	FALSE	Status of axis enable	
				FALSE	The axis is disabled. The axis does not execute Motion Control commands and does not accept any new commands (exception: MC_Reset command). The axis is not homed. Upon disabling, the status does not change to FALSE until the axis reaches a standstill.
				TRUE	The axis is enabled. The axis is ready to execute Motion Control commands. Upon axis enabling, the status does not change to TRUE until the signal "Drive ready" is pending. If the "Drive ready" drive interface was not configured in the axis configuration, the status changes to TRUE immediately.
Busy	OUTPUT	BOOL	FALSE	TRUE	"MC_Power" is active.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred in Motion Control instruction "MC_Power" or in the associated technology object. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 10132) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 10132) for parameter "ErrorID"	

Note

If the axis is switched off due to an error, it will be enabled again automatically after the error has been eliminated and acknowledged. This requires that the input parameter "Enable" has retained the value TRUE during this process.

Enabling an axis with configured drive interface

To enable the axis, follow these steps:

1. Check the requirements indicated above.
2. Initialize input parameter "StopMode" with the desired value. Set the input parameter "Enable" to TRUE.
   The enable output for "Drive enabled" changes to TRUE to enable the power to the drive.
   The CPU waits for the "Drive ready" signal of the drive.
   When the "Drive ready" signal is available at the configured ready input of the CPU, the axis is enabled. Output parameter "Status" and tag of technology object <Axis name>.StatusBits.Enable indicate the value TRUE.

Enabling an axis without configured drive interface

To enable the axis, follow these steps:

1. Check the requirements indicated above.
2. Initialize input parameter "StopMode" with the desired value. Set the input parameter "Enable" to TRUE. The axis is enabled. The output parameter "Status" and the tag of the technology object <Axis name>.StatusBits.Enable indicate the value TRUE.

Disabling an axis

To disable an axis, you can follow the steps described below:

1. Bring the axis to a standstill.
   You can identify when the axis is at a standstill in the tag of the technology object <Axis name>.StatusBits.StandStill.
2. Set input parameter "Enable" to FALSE after standstill is reached.
3. If output parameters "Busy" and "Status" and tag of technology object <Axis name>.StatusBits.Enable indicate the value FALSE, disabling of the axis is complete.

See also

Overview of the Motion Control statements (Page 10096)

List of ErrorIDs and ErrorInfos (technology objects as of V4) (Page 10132)

MC_Power: Function chart V4 (Page 4826)

MC_Reset: Acknowledge fault V4 (Page 4827)

MC_Home: Home axes, set reference point V4 (Page 4829)

MC_Halt: Halt axes V4 (Page 4832)

MC_MoveAbsolute: Absolute positioning of axes V4 (Page 4835)

MC_MoveRelative: Relative positioning of axes V4 (Page 4838)

MC_MoveVelocity: Move axes at preset rotational speed V4 (Page 4842)

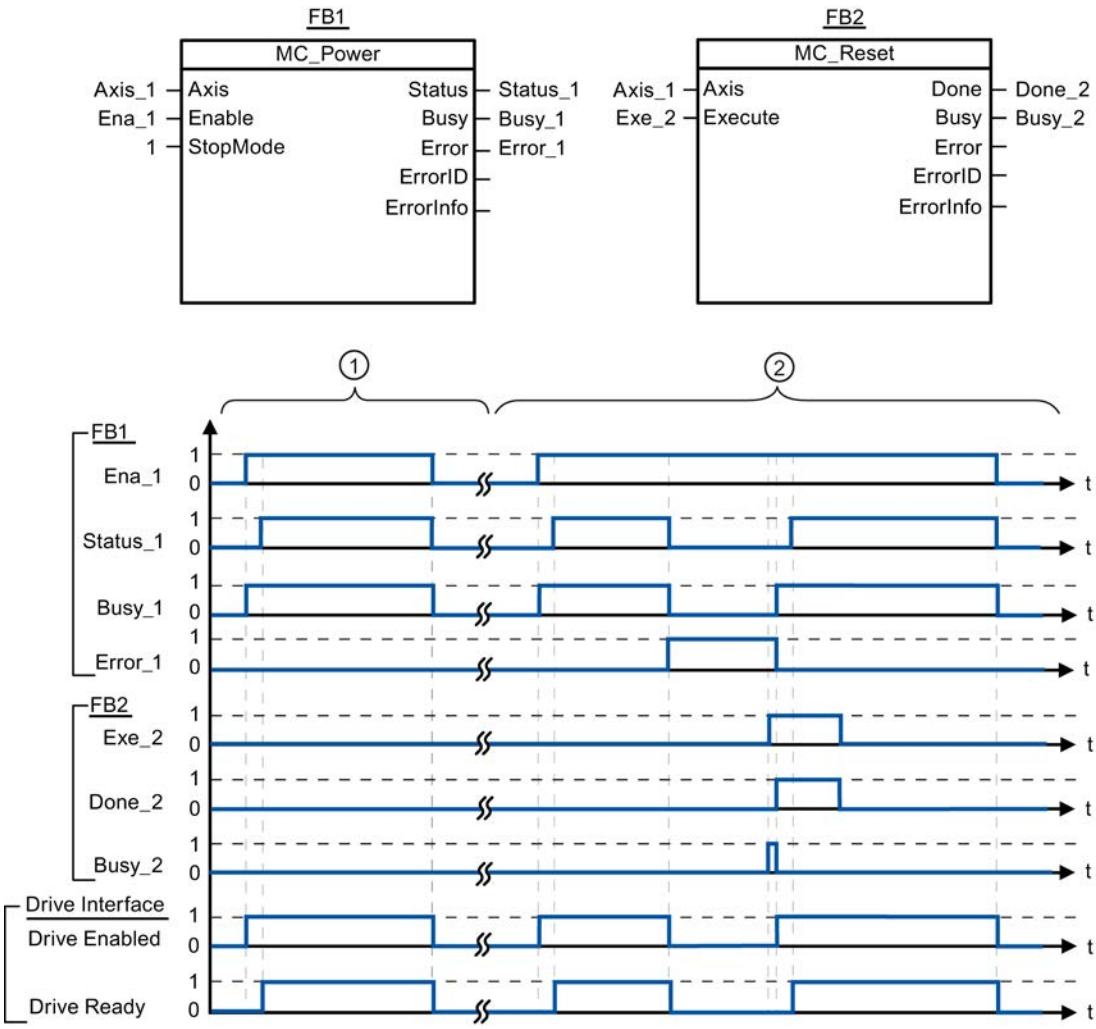
MC_MoveJog: Move axes in jog mode V4 (Page 4846)

MC_ChangeDynamic: Change dynamic settings of axis V4 (Page 4851)

MC_CommandTable: Run axis commands as motion sequence V4 (Page 4849)
MC_ReadParam: Continuously read motion data of a positioning axis V4 (Page 4853)
MC_WriteParam: Write tag of positioning axis V4 (Page 4855)
S7-1200 Motion Control V1...3 (Page 4856)

MC_Power: Function chart V4

Function chart



①	An axis is enabled and then disabled again. When the drive has signaled "Drive ready" back to the CPU, the successful enable can be read out via "Status_1".
②	Following an axis enable, an error has occurred that caused the axis to be disabled. The error is eliminated and acknowledged with "MC_Reset". The axis is then enabled again.

See also

MC_Power: Enable, disable axis V4 (Page 4823)

MC_Reset

MC_Reset: Acknowledge fault V4

Description

Motion Control instruction "MC_Reset" can be used to acknowledge "Operating error with axis stop" and "Configuration error". The errors that require acknowledgement can be found in the "List of ErrorIDs and ErrorInfos" under "Remedy".

The axis configuration can be downloaded to the work memory after a download in RUN operating mode.

Requirements

- The positioning axis technology object has been configured correctly.
- The cause of a pending configuration error requiring acknowledgement has been eliminated (for example, acceleration in positioning axis technology object has been changed to a valid value).

Override response

The MC_Reset command cannot be aborted by any other Motion Control command.

The new MC_Reset command does not abort any other active Motion Control commands.

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Restart	INPUT	BOOL	FALSE	TRUE	Download the axis configuration from the load memory to the work memory. The command can only be executed when the axis is disabled. Refer to the notes on Download to the CPU (Page 10074).
				FALSE	Acknowledges pending errors
Done	OUTPUT	BOOL	FALSE	TRUE	Error has been acknowledged.
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".

Parameter	Declaration	Data type	Default value	Description
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 10132) for parameter "Error"
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 10132) for parameter "ErrorID"

Acknowledging an error requiring acknowledgement with MC_Reset

To acknowledge an error, follow these steps:

1. Check the requirements indicated above.
2. Start the acknowledgement of the error with a rising edge at input parameter "Execute".
3. If output parameter "Done" indicates the value TRUE and tag of technology object <Axis name>.StatusBits.Error the value FALSE, the error has been acknowledged.

See also

Overview of the Motion Control statements (Page 10096)

List of ErrorIDs and ErrorInfos (technology objects as of V4) (Page 10132)

Download to CPU (Page 10074)

MC_Power: Enable, disable axis V4 (Page 4823)

MC_Home: Home axes, set reference point V4 (Page 4829)

MC_Halt: Halt axes V4 (Page 4832)

MC_MoveAbsolute: Absolute positioning of axes V4 (Page 4835)

MC_MoveRelative: Relative positioning of axes V4 (Page 4838)

MC_MoveVelocity: Move axes at preset rotational speed V4 (Page 4842)

MC_MoveJog: Move axes in jog mode V4 (Page 4846)

MC_CommandTable: Run axis commands as motion sequence V4 (Page 4849)

MC_ChangeDynamic: Change dynamic settings of axis V4 (Page 4851)

MC_ReadParam: Continuously read motion data of a positioning axis V4 (Page 4853)

MC_WriteParam: Write tag of positioning axis V4 (Page 4855)

S7-1200 Motion Control V1...3 (Page 4856)

MC_Home

MC_Home: Home axes, set reference point V4

Description

Motion Control instruction "MC_Home" is used to match the axis coordinates to the real, physical drive position. Homing is required for absolute positioning of the axis. The following types of homing can be executed:

- Active homing (Mode = 3)
  The homing procedure is executed automatically.
- Passive homing (Mode = 2)
  During passive homing, the "MC_Home" Motion Control instruction does not carry out any homing motion. The traversing motion required for this must be implemented by the user via other Motion Control instructions. When the homing switch is detected, the axis is homed.
- Direct homing absolute (Mode = 0)
  The current axis position is set to the value of parameter "Position".
- Direct homing relative (Mode = 1)
  The current axis position is offset by the value of parameter "Position".

Requirements

- The positioning axis technology object has been configured correctly.
- The axis is enabled.
- No MC_CommandTable command may be active upon start with Mode = 0, 1, or 2.

Override response

The override response depends on the selected mode:

Mode = 0, 1

The MC_Home command cannot be aborted by any other Motion Control command.

The MC_Home command does not abort any active Motion Control commands. Position-related motion commands are resumed after homing according to the new homing position (value at input parameter: "Position").

Mode = 2

The MC_Home command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 2, 3

The new MC_Home command aborts the following active Motion Control command:

- MC_Home command Mode = 2

Position-related motion commands are resumed after homing according to the new homing position (value at input parameter: "Position").

Mode = 3

The MC_Home command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_Home command aborts the following active Motion Control commands:

- MC_Home command Mode = 2, 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Position	INPUT	REAL	0.0	• Mode = 0, 2, and 3 Absolute position of axis after completion of the homing operation • Mode = 1 Correction value for the current axis position Limit values: $-1.0e^{12} \leq$ Position $\leq 1.0e^{12}$	
Mode	INPUT	INT	0	Homing mode	
				0	Direct homing (absolute) New axis position is the position value of parameter "Position".
				1	Direct homing (relative) New axis position is the current axis position + position value of parameter "Position".

Parameter	Declaration	Data type	Default value	Description	
				2	Passive homing Homing according to the axis configuration. Following homing, the value of parameter "Position" is set as the new axis position.
				3	Active homing Homing procedure in accordance with the axis configuration. Following homing, the value of parameter "Position" is set as the new axis position.
Done	OUTPUT	BOOL	FALSE	TRUE	Command completed
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 10132) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 10132) for parameter "ErrorID"	

Note

Axis homing is lost under the following conditions:

- Disabling of axis by the "MC_Power" Motion Control instruction
- Changeover between automatic mode and manual control
- Upon start of active homing. After successful completion of the homing operation, axis homing is again available.
- After POWER OFF -> POWER ON of the CPU
- After CPU restart (RUN-STOP -> STOP-RUN)

Homing an axis

To home the axis, follow these stops:

1. Check the requirements indicated above.
2. Initialize the necessary input parameters with values, and start the homing operation with a rising edge at input parameter "Execute"
3. If output parameter "Done" and technology object tag <Axis name>.StatusBits.HomingDone indicate the value TRUE, homing is complete.

See also

Overview of the Motion Control statements (Page 10096)

List of ErrorIDs and ErrorInfos (technology objects as of V4) (Page 10132)

MC_Power: Enable, disable axis V4 (Page 4823)
MC_Reset: Acknowledge fault V4 (Page 4827)
MC_Halt: Halt axes V4 (Page 4832)
MC_MoveAbsolute: Absolute positioning of axes V4 (Page 4835)
MC_MoveRelative: Relative positioning of axes V4 (Page 4838)
MC_MoveVelocity: Move axes at preset rotational speed V4 (Page 4842)
MC_MoveJog: Move axes in jog mode V4 (Page 4846)
MC_CommandTable: Run axis commands as motion sequence V4 (Page 4849)
MC_ChangeDynamic: Change dynamic settings of axis V4 (Page 4851)
MC_ReadParam: Continuously read motion data of a positioning axis V4 (Page 4853)
MC_WriteParam: Write tag of positioning axis V4 (Page 4855)
S7-1200 Motion Control V1...3 (Page 4856)

MC_Halt

MC_Halt: Halt axes V4

Description

The "MC_Halt" Motion Control instruction stops all movements and brings the axis to a standstill with the configured deceleration. The standstill position is not defined.

Requirements

- The positioning axis technology object has been configured correctly.
- The axis is enabled.

Override response

The MC_Halt command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_Halt command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_SpeedAxis	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Done	OUTPUT	BOOL	FALSE	TRUE	Zero velocity reached
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 10132) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 10132) for parameter "ErrorID"	

See also

Overview of the Motion Control statements (Page 10096)

List of ErrorIDs and ErrorInfos (technology objects as of V4) (Page 10132)

MC_Halt: Function chart V4 (Page 4834)

MC_Power: Enable, disable axis V4 (Page 4823)

MC_Reset: Acknowledge fault V4 (Page 4827)

MC_Home: Home axes, set reference point V4 (Page 4829)

MC_MoveAbsolute: Absolute positioning of axes V4 (Page 4835)

MC_MoveRelative: Relative positioning of axes V4 (Page 4838)

MC_MoveVelocity: Move axes at preset rotational speed V4 (Page 4842)

MC_MoveJog: Move axes in jog mode V4 (Page 4846)

MC_CommandTable: Run axis commands as motion sequence V4 (Page 4849)

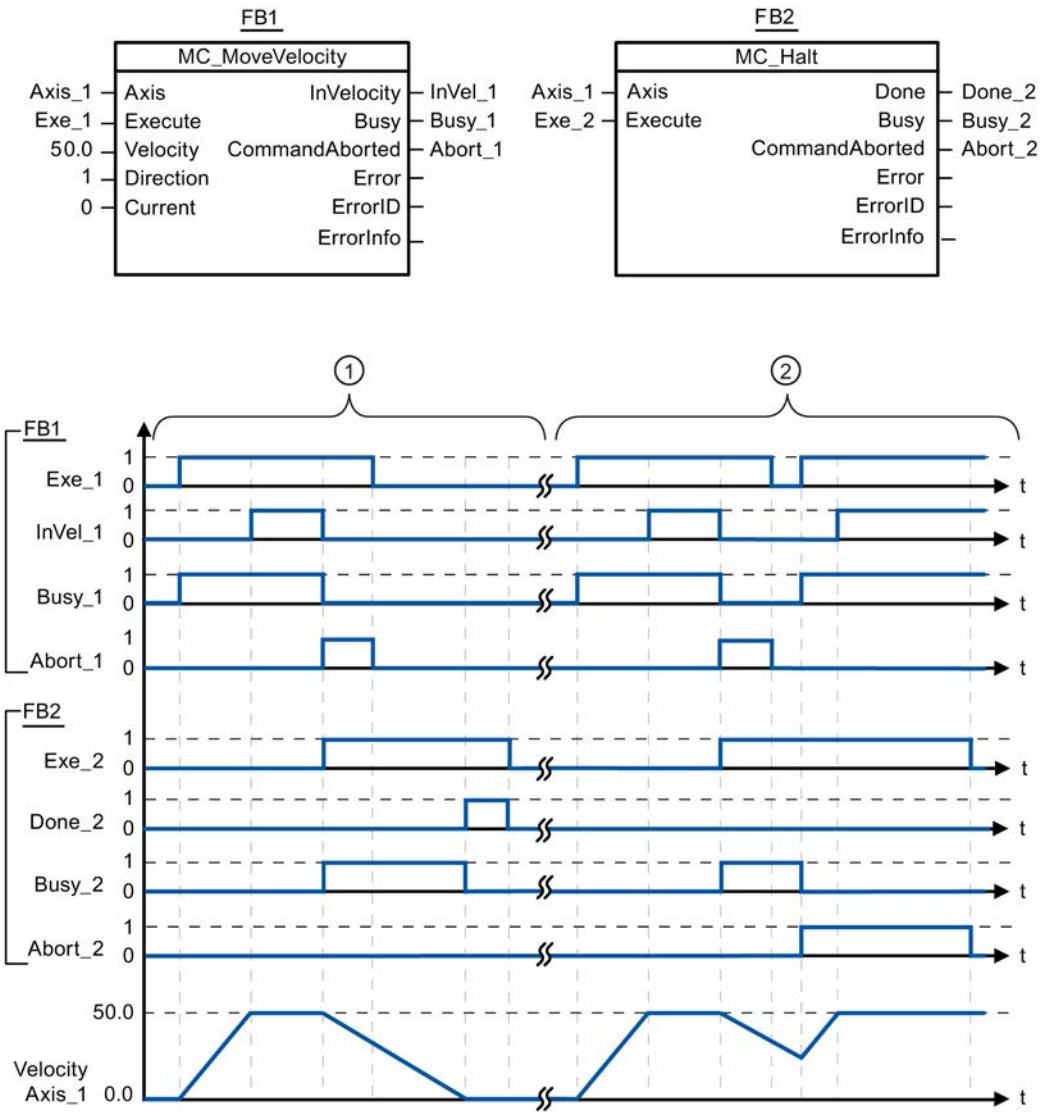
MC_ChangeDynamic: Change dynamic settings of axis V4 (Page 4851)

MC_ReadParam: Continuously read motion data of a positioning axis V4 (Page 4853)

MC_WriteParam: Write tag of positioning axis V4 (Page 4855)
S7-1200 Motion Control V1...3 (Page 4856)

MC_Halt: Function chart V4

Function chart



The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 5.0

①	The axis is braked by an MC_Halt command until it comes to a standstill. The axis standstill is signaled via "Done_2".
②	While an MC_Halt command is braking the axis, this command is aborted by another motion command. The abort is signaled via "Abort_2".

See also

MC_Halt: Halt axes V4 (Page 4832)

MC_MoveAbsolute

MC_MoveAbsolute: Absolute positioning of axes V4

Description

The "MC_MoveAbsolute" Motion Control instruction starts an axis positioning motion to move it to an absolute position.

Requirements

- The positioning axis technology object has been configured correctly.
- The axis is enabled.
- The axis is homed.

Override response

The MC_MoveAbsolute command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_MoveAbsolute command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command

- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Positioning Axis	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Position	INPUT	REAL	0.0	Absolute target position Limit values: $-1.0e^{12} \leq$ Position $\leq 1.0e^{12}$	
Velocity	INPUT	REAL	10.0	Velocity of axis This velocity is not always reached on account of the configured acceleration and deceleration and the target position to be approached. Limit values: Start/stop velocity ≤ Velocity ≤ maximum velocity	
Done	OUTPUT	BOOL	FALSE	TRUE	Absolute target position reached
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 10132) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 10132) for parameter "ErrorID"	

See also

Overview of the Motion Control statements (Page 10096)

List of ErrorIDs and ErrorInfos (technology objects as of V4) (Page 10132)

MC_MoveAbsolute: Function chart V4 (Page 4837)

MC_Power: Enable, disable axis V4 (Page 4823)

MC_Reset: Acknowledge fault V4 (Page 4827)

MC_Home: Home axes, set reference point V4 (Page 4829)

MC_Halt: Halt axes V4 (Page 4832)

MC_MoveRelative: Relative positioning of axes V4 (Page 4838)

MC_MoveVelocity: Move axes at preset rotational speed V4 (Page 4842)

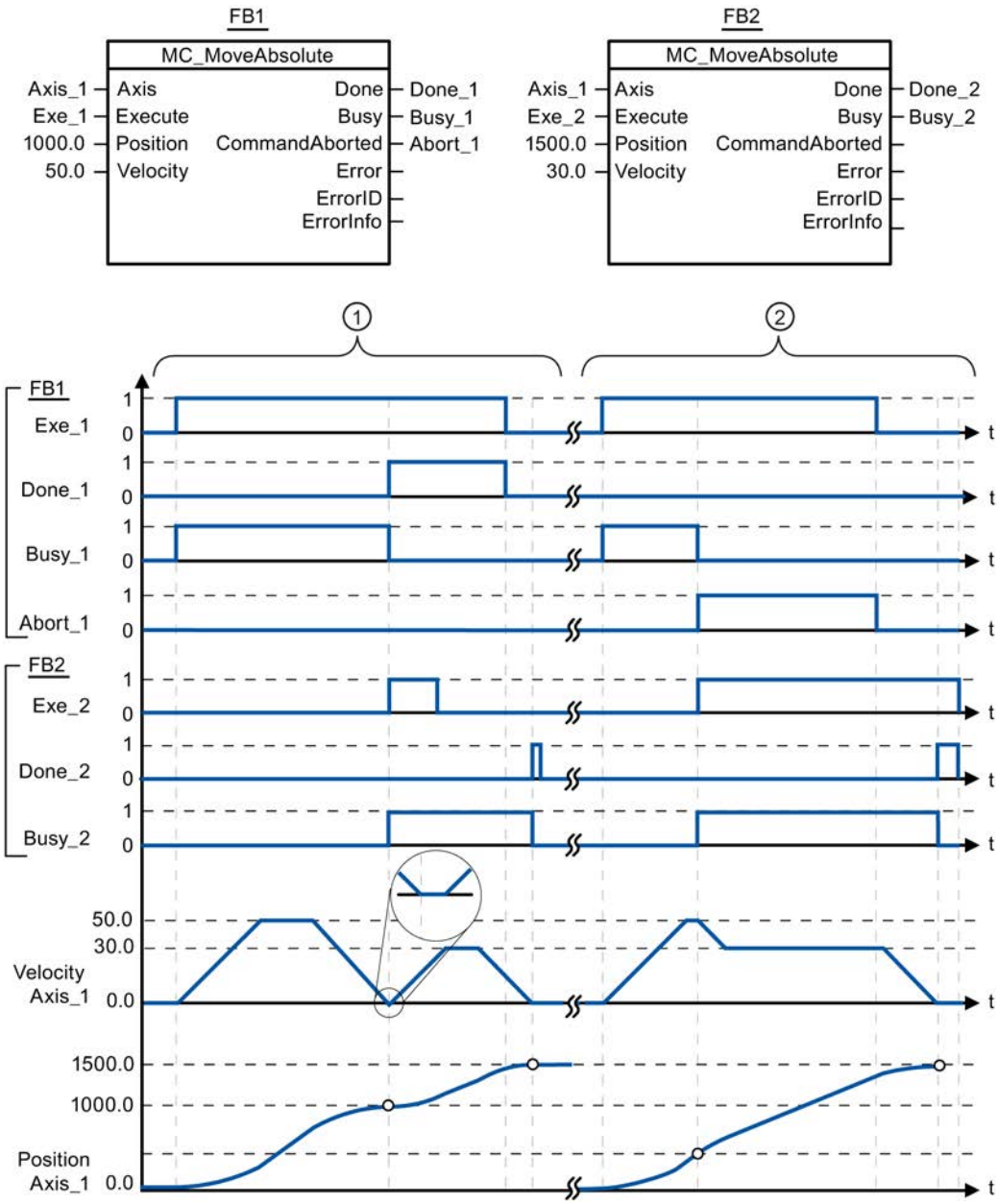
MC_MoveJog: Move axes in jog mode V4 (Page 4846)

MC_CommandTable: Run axis commands as motion sequence V4 (Page 4849)

MC_ChangeDynamic: Change dynamic settings of axis V4 (Page 4851)
MC_ReadParam: Continuously read motion data of a positioning axis V4 (Page 4853)
MC_WriteParam: Write tag of positioning axis V4 (Page 4855)
S7-1200 Motion Control V1...3 (Page 4856)

MC_MoveAbsolute: Function chart V4

Function chart



The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 10.0

①	An axis is moved to absolute position 1000.0 with an MC_MoveAbsolute command. When the axis reaches the target position, this is signaled via "Done_1". When "Done_1" = TRUE, another MC_MoveAbsolute command, with target position 1500.0, is started. Because of the response times (e.g., cycle time of user program, etc.), the axis comes to a standstill briefly (see zoomed-in detail). When the axis reaches the new target position, this is signaled via "Done_2".
②	An active MC_MoveAbsolute command is aborted by another MC_MoveAbsolute command. The abort is signaled via "Abort_1". The axis is then moved at the new velocity to the new target position 1500.0. When the new target position is reached, this is signaled via "Done_2".

See also

MC_MoveAbsolute: Absolute positioning of axes V4 (Page 4835)

MC_MoveRelative

MC_MoveRelative: Relative positioning of axes V4

Description

The "MC_MoveRelative" Motion Control instruction starts a positioning motion relative to the start position.

Requirements

- The positioning axis technology object has been configured correctly.
- The axis is enabled.

Override response

The MC_MoveRelative command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_MoveRelative command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Positioning Axis	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Distance	INPUT	REAL	0.0	Travel distance for the positioning operation Limit values: $-1.0e^{12} \leq$ Distance $\leq 1.0e^{12}$	
Velocity	INPUT	REAL	10.0	Velocity of axis This velocity is not always reached on account of the configured acceleration and deceleration and the distance to be traveled. Limit values: Start/stop velocity ≤ Velocity ≤ maximum velocity	
Done	OUTPUT	BOOL	FALSE	TRUE	Target position reached
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 10132) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 10132) for parameter "ErrorID"	

See also

Overview of the Motion Control statements (Page 10096)

List of ErrorIDs and ErrorInfos (technology objects as of V4) (Page 10132)

MC_MoveRelative: Function chart V4 (Page 4841)

MC_Power: Enable, disable axis V4 (Page 4823)

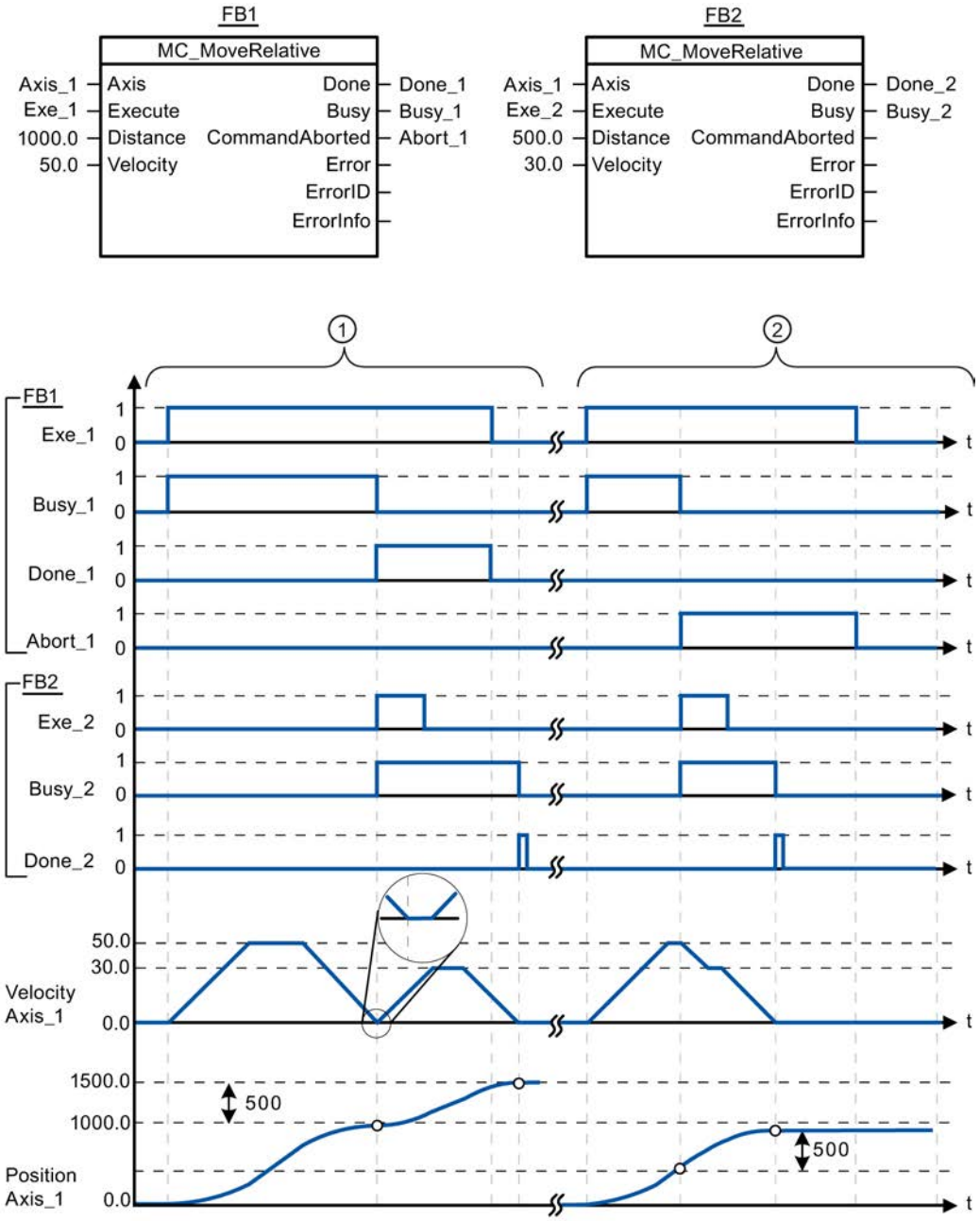
MC_Reset: Acknowledge fault V4 (Page 4827)

MC_Home: Home axes, set reference point V4 (Page 4829)

MC_Halt: Halt axes V4 (Page 4832)
MC_MoveAbsolute: Absolute positioning of axes V4 (Page 4835)
MC_MoveVelocity: Move axes at preset rotational speed V4 (Page 4842)
MC_MoveJog: Move axes in jog mode V4 (Page 4846)
MC_CommandTable: Run axis commands as motion sequence V4 (Page 4849)
MC_ChangeDynamic: Change dynamic settings of axis V4 (Page 4851)
MC_ReadParam: Continuously read motion data of a positioning axis V4 (Page 4853)
MC_WriteParam: Write tag of positioning axis V4 (Page 4855)
S7-1200 Motion Control V1...3 (Page 4856)

MC_MoveRelative: Function chart V4

Function chart



The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 10.0

①	The axis is moved by an MC_MoveRelative command by the distance ("Distance") 1000.0. When the axis reaches the target position, this is signaled via "Done_1". When "Done_1" = TRUE, another MC_MoveRelative command, with travel distance 500.0, is started. Because of the response times (e.g., cycle time of user program, etc.), the axis comes to a standstill briefly (see zoomed-in detail). When the axis reaches the new target position, this is signaled via "Done_2".
②	An active MC_MoveRelative command is aborted by another MC_MoveRelative command. The abort is signaled via "Abort_1". The axis is then moved at the new velocity by the new distance ("Distance") 500.0. When the new target position is reached, this is signaled via "Done_2".

See also

MC_MoveRelative: Relative positioning of axes V4 (Page 4838)

MC_MoveVelocity

MC_MoveVelocity: Move axes at preset rotational speed V4

Description

The Motion Control instruction "MC_MoveVelocity" moves the axis constantly at the specified velocity.

Requirements

- The positioning axis technology object has been configured correctly.
- The axis is enabled.

Override response

MC_MoveVelocity can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_MoveVelocity command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command

- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_SpeedAxis	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Velocity	INPUT	REAL	10.0	Velocity specification for axis motion Limit values: Start/stop velocity ≤ |Velocity ≤ maximum velocity (Velocity = 0.0 is permitted)	
Direction	INPUT	INT	0	Direction specification	
				0	Direction of rotation corresponds to the sign of the value in parameter "Velocity"
				1	Positive direction of rotation (The sign of the value in parameter "Velocity" is ignored)
				2	Negative direction of rotation (The sign of the value in parameter "Velocity" is ignored)
Current	INPUT	BOOL	FALSE	Maintain current velocity	
				FALSE	"Maintain current velocity" is deactivated. The values of parameters "Velocity" and "Direction" are used.
				TRUE	"Maintain current velocity" is activated. The values in parameters "Velocity" and "Direction" are not taken into account. When the axis resumes motion at the current velocity, the "InVelocity" parameter returns the value TRUE.
InVelocity	OUTPUT	BOOL	FALSE	TRUE	• "Current" = FALSE: The velocity specified in parameter "Velocity" was reached. • "Current" = TRUE: The axis travels at the current velocity at the start time.
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 10132) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 10132) for parameter "ErrorID"	

Note

PLCopen Version 2.0

The Motion Control instruction "MC_MoveVelocity" is compatible to PLCopen Version 2.0 as of V4.

The "InVelocity" and "Busy" parameters show their status regardless of the "Execute" parameter until the command is overridden or stopped by an error. For more information, refer to the section " Tracking active commands (Page 10106).

Behavior with zero setpoint velocity (Velocity = 0.0)

An MC_MoveVelocity command with "Velocity" = 0.0 (such as an MC_Halt command) aborts active motion commands and stops the axis with the configured deceleration.

When the axis comes to a standstill, output parameter "InVelocity" indicates TRUE for at least one program cycle.

"Busy" indicates the value TRUE during the deceleration process and changes to FALSE together with "InVelocity". If parameter "Execute" = TRUE is set, "InVelocity" and "Busy" are latched.

When the "MC_MoveVelocity" command is started, status bit "SpeedCommand" is set in the technology object. Status bit "ConstantVelocity" is set upon axis standstill. Both bits are adapted to the new situation when a new motion command is started.

See also

Overview of the Motion Control statements (Page 10096)

List of ErrorIDs and ErrorInfos (technology objects as of V4) (Page 10132)

MC_MoveVelocity: Function chart V4 (Page 4845)

MC_Power: Enable, disable axis V4 (Page 4823)

MC_Reset: Acknowledge fault V4 (Page 4827)

MC_Home: Home axes, set reference point V4 (Page 4829)

MC_Halt: Halt axes V4 (Page 4832)

MC_MoveAbsolute: Absolute positioning of axes V4 (Page 4835)

MC_MoveRelative: Relative positioning of axes V4 (Page 4838)

MC_MoveJog: Move axes in jog mode V4 (Page 4846)

MC_CommandTable: Run axis commands as motion sequence V4 (Page 4849)

MC_ChangeDynamic: Change dynamic settings of axis V4 (Page 4851)

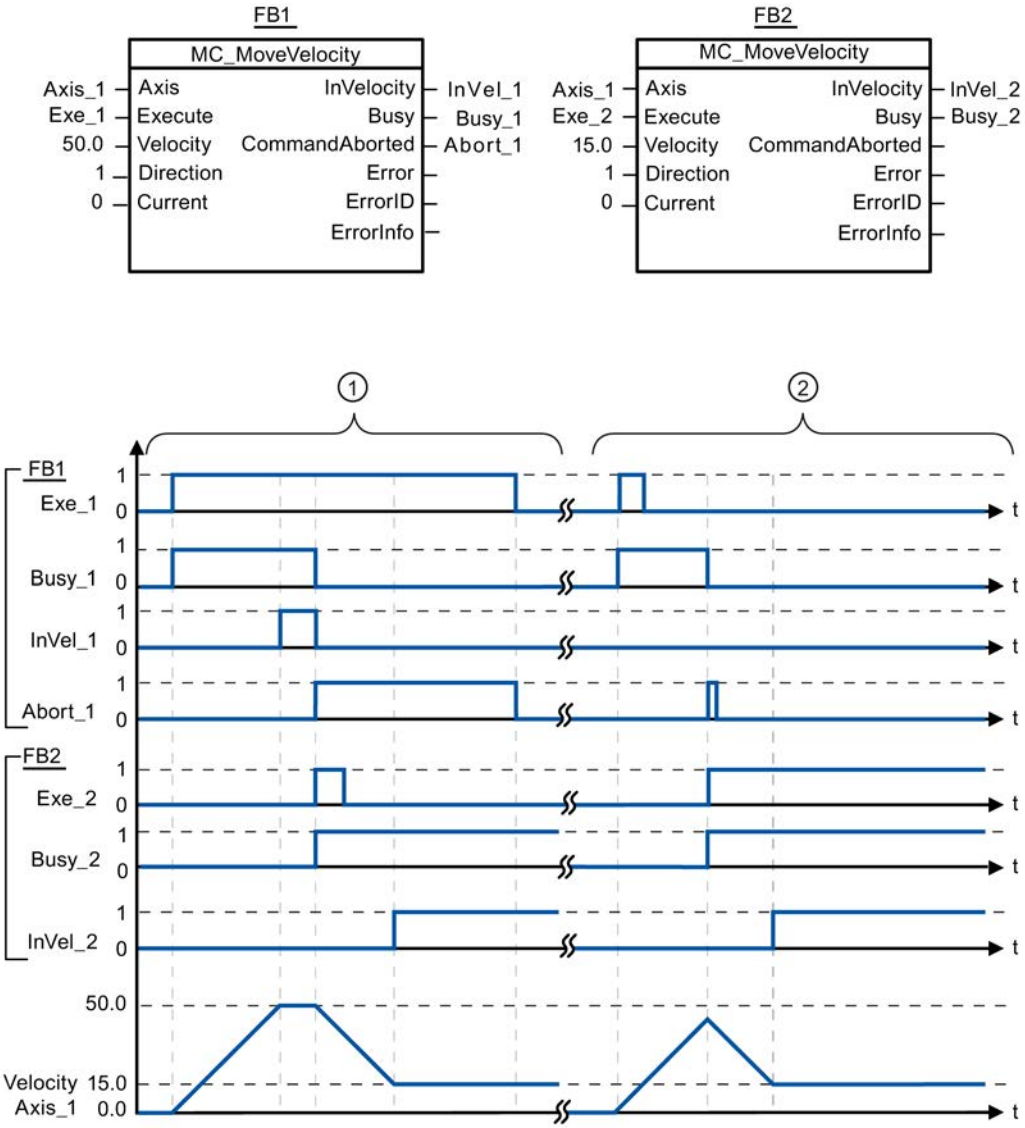
MC_ReadParam: Continuously read motion data of a positioning axis V4 (Page 4853)

MC_WriteParam: Write tag of positioning axis V4 (Page 4855)

S7-1200 Motion Control V1...3 (Page 4856)

MC_MoveVelocity: Function chart V4

Function chart



The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 10.0

①	An active MC_MoveVelocity command signals via "InVel_1" that its target velocity has been reached. It is then aborted by another MC_MoveVelocity command. The abort is signaled via "Abort_1". When the new target velocity 15.0 is reached, this is signaled via "InVel_2". The axis then continues moving at the new constant velocity.
②	An active MC_MoveVelocity command is aborted by another MC_MoveVelocity command prior to reaching its target velocity. The abort is signaled via "Abort_1". When the new target velocity 15.0 is reached, this is signaled via "InVel_2". The axis then continues moving at the new constant velocity.

See also

MC_MoveVelocity: Move axes at preset rotational speed V4 (Page 4842)

MC_MoveJog

MC_MoveJog: Move axes in jog mode V4

Description

Motion control instruction "MC_MoveJog" moves the axis constantly at the specified velocity in jog mode. You use this Motion Control instruction, for example, for testing and commissioning purposes.

Requirements

- The positioning axis technology object has been configured correctly.
- The axis is enabled.

Override response

The MC_MoveJog command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_MoveJog command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_SpeedAxis	-	Axis technology object	
JogForward	INPUT	BOOL	FALSE	As long as the parameter is TRUE, the axis moves in the positive direction at the velocity specified in parameter "Velocity".	
JogBackward	INPUT	BOOL	FALSE	As long as the parameter is TRUE, the axis moves in the negative direction at the velocity specified in parameter "Velocity".	
If both parameters are simultaneously TRUE, the axis stops with the configured deceleration. An error is indicated in parameters "Error", "ErrorID", and "ErrorInfo".					
Velocity	INPUT	REAL	10.0	Preset velocity for jog mode Limit values: Start/stop velocity ≤ velocity ≤ maximum velocity	
InVelocity	OUTPUT	BOOL	FALSE	TRUE	The velocity specified in parameter "Velocity" was reached.
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 10132) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 10132) for parameter "ErrorID"	

See also

Overview of the Motion Control statements (Page 10096)

List of ErrorIDs and ErrorInfos (technology objects as of V4) (Page 10132)

MC_MoveJog: Function chart V4 (Page 4848)

MC_Power: Enable, disable axis V4 (Page 4823)

MC_Reset: Acknowledge fault V4 (Page 4827)

MC_Home: Home axes, set reference point V4 (Page 4829)

MC_Halt: Halt axes V4 (Page 4832)

MC_MoveAbsolute: Absolute positioning of axes V4 (Page 4835)

MC_MoveRelative: Relative positioning of axes V4 (Page 4838)

MC_MoveVelocity: Move axes at preset rotational speed V4 (Page 4842)

MC_CommandTable: Run axis commands as motion sequence V4 (Page 4849)

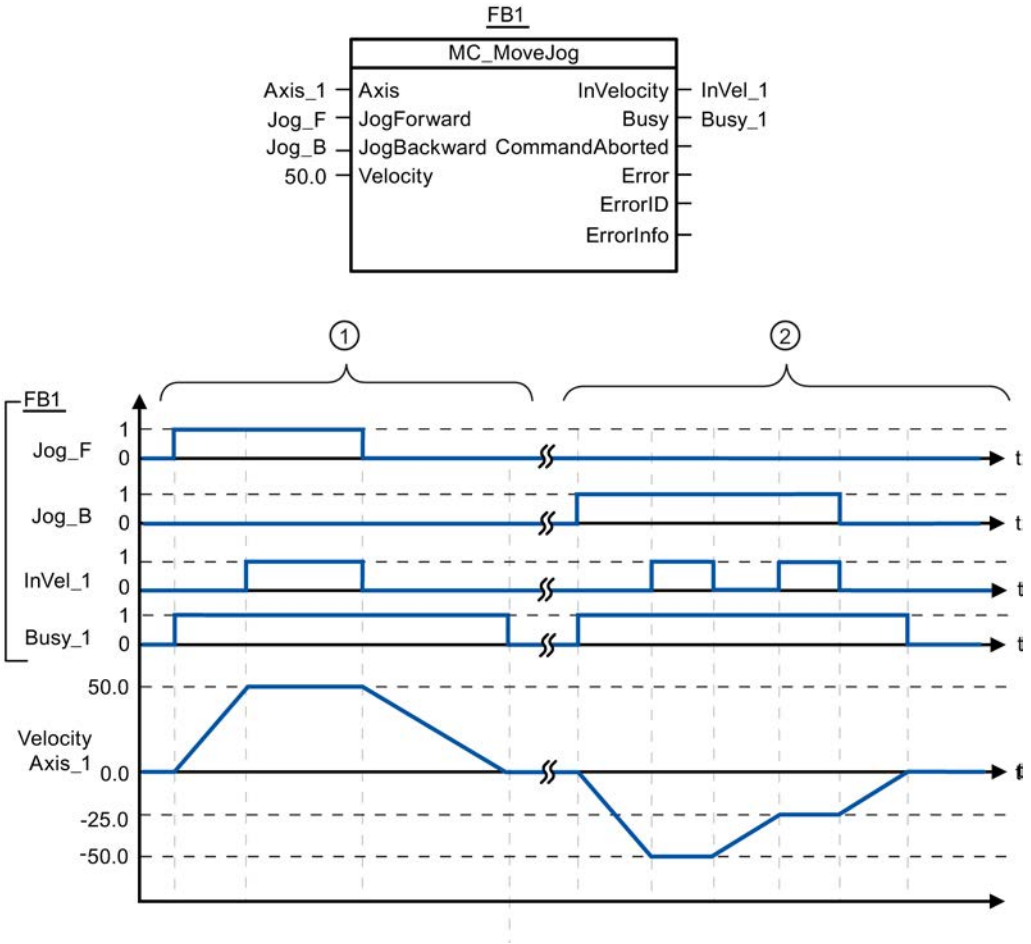
MC_ChangeDynamic: Change dynamic settings of axis V4 (Page 4851)

MC_ReadParam: Continuously read motion data of a positioning axis V4 (Page 4853)

MC_WriteParam: Write tag of positioning axis V4 (Page 4855)
S7-1200 Motion Control V1...3 (Page 4856)

MC_MoveJog: Function chart V4

Function chart



The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 5.0

①	The axis is moved in the positive direction in jog mode via "Jog_F". When the target velocity 50.0 is reached, this is signaled via "InVel_1". After "Jog_F" is reset, the axis is braked to a standstill.
②	The axis is moved in the negative direction in jog mode via "Jog_B". When the target velocity -50.0 is reached, this is signaled via "InVel_1". When "Jog_B" is set, the value at parameter "Velocity" changes to 25.0. "InVel_1" is reset and the axis is braked. When the new target velocity -25.0 is reached, this is signaled via "InVel_1". After "Jog_B" is reset, the axis is braked to a standstill.

See also

MC_MoveJog: Move axes in jog mode V4 (Page 4846)

MC_CommandTable

MC_CommandTable: Run axis commands as motion sequence V4

Description

The Motion Control instruction "MC_CommandTable" combines multiple individual axis control commands in one movement sequence.

Requirements

- The positioning axis technology object has been inserted and correctly configured.
- The command table technology object has been inserted and correctly configured.
- The axis is enabled.

Override response

The MC_CommandTable command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_CommandTable command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The active Motion Control command is cancelled by the start of the first "Positioning Relative", "Positioning Absolute", "Velocity set point" or "Halt" command.

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_SpeedAxis	-	Axis technology object	
CommandTable	INPUT	TO_CommandTable	-	Command table technology object	
Execute	INPUT	BOOL	FALSE	Command table start with positive edge	
StartStep	INPUT	INT	1	Defines the step at which the execution of the command table should begin Limit values: 1 ≤ StartStep ≤ EndStep	
EndStep	INPUT	INT	32	Defines the step up to which the execution of command table should take place Limit values: StartStep ≤ EndStep ≤ 32	
Done	OUTPUT	BOOL	FALSE	TRUE	Command table has been successfully executed
Busy	OUTPU	BOOL	FALSE	TRUE	The command table is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The command table was cancelled by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command table. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 10132) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 10132) for parameter "ErrorID"	
CurrentStep	OUTPUT	INT	0	Step in command table currently being executed	
StepCode	OUTPUT	WORD	16#0000	User-defined numerical value / bit pattern of the step currently being executed	

See also

Overview of the Motion Control statements (Page 10096)

List of ErrorIDs and ErrorInfos (technology objects as of V4) (Page 10132)

MC_Power: Enable, disable axis V4 (Page 4823)

MC_Reset: Acknowledge fault V4 (Page 4827)

MC_Home: Home axes, set reference point V4 (Page 4829)

MC_Halt: Halt axes V4 (Page 4832)

MC_MoveAbsolute: Absolute positioning of axes V4 (Page 4835)

MC_MoveRelative: Relative positioning of axes V4 (Page 4838)

MC_MoveVelocity: Move axes at preset rotational speed V4 (Page 4842)

MC_MoveJog: Move axes in jog mode V4 (Page 4846)

MC_ChangeDynamic: Change dynamic settings of axis V4 (Page 4851)

MC_ReadParam: Continuously read motion data of a positioning axis V4 (Page 4853)

MC_WriteParam: Write tag of positioning axis V4 (Page 4855)

S7-1200 Motion Control V1...3 (Page 4856)

MC_ChangeDynamic

MC_ChangeDynamic: Change dynamic settings of axis V4

Description

Motion Control instruction "MC_ChangeDynamic" allows you to change the following settings of the axis:

- Change the ramp-up time (acceleration) value
- Change the ramp-down time (deceleration) value
- Change the emergency stop ramp-down time (emergency stop deceleration) value
- Change the smoothing time (jerk) value

For the effectiveness of the change, refer to the description of the tag (Page 10142).

Requirements

The positioning axis technology object has been configured correctly.

Override response

A MC_ChangeDynamic command cannot be aborted by any other Motion Control command.

A new MC_ChangeDynamic command does not abort any active Motion Control commands.

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_SpeedAxis	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
ChangeRampUp	INPUT	BOOL	FALSE	TRUE	Change ramp-up time in line with input parameter "RampUpTime"
RampUpTime	INPUT	REAL	5.00	Time (in seconds) to accelerate axis from standstill to configured maximum velocity without jerk limit. The change will influence the tag <Axis name>. Config.DynamicDefaults.Acceleration. For the effectiveness of the change, refer to the description of this tag.	
ChangeRampDown	INPUT	BOOL	FALSE	TRUE	Change ramp-down time to correspond to input parameter "RampDownTime"

Parameter	Declaration	Data type	Default value	Description	
RampDownTime	INPUT	REAL	5.00	Time (in seconds) to decelerate axis from the configured maximum velocity to standstill without jerk limiter. The change will influence the tag <Axis name>. Config.DynamicDefaults.Deceleration . For the effectiveness of the change, refer to the description of this tag.	
ChangeEmergency	INPUT	BOOL	FALSE	TRUE	Change emergency stop ramp-down time in line with input parameter "EmergencyRampTime"
EmergencyRampTime	INPUT	REAL	2.00	Time (in seconds) to decelerate the axis from configured maximum velocity to standstill without jerk limiter in emergency stop mode. The change will influence the tag <Axis name>. Config.DynamicDefaults.EmergencyDeceleration . For the effectiveness of the change, refer to the description of this tag.	
ChangeJerkTime	INPUT	BOOL	FALSE	TRUE	Change smoothing time according to the input parameter "JerkTime"
JerkTime	INPUT	REAL	0.25	Smoothing time (in seconds) used for the axis acceleration and deceleration ramps The change will influence the tag <Axis name>. Config.DynamicDefaults.Jerk . For the effectiveness of the change, refer to the description of this tag.	
Done	OUTPUT	BOOL	FALSE	TRUE	The changed values have been written to the technology data block. The description of the tags will show when the change becomes effective.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 10132) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 10132) for parameter "ErrorID"	

Note

At the input parameters "RampUpTime", "RampDownTime", "EmergencyRampTime" and "JerkTime", values can be entered which exceed the admissible limits of the resulting parameters: "Acceleration", "Deceleration", "Emergency stop deceleration" and "Jerk".

Ensure that your inputs are within the valid range, taking into consideration the equations and limits in section "Dynamic (Page 10041)".

See also

Overview of the Motion Control statements (Page 10096)

List of ErrorIDs and ErrorInfos (technology objects as of V4) (Page 10132)

Changing the configuration of dynamics in the user program (Page 10047)

Changing the homing configuration in the user program (Page 10054)

Tags of the positioning axis technology object as of V4 (Page 10142)

MC_Power: Enable, disable axis V4 (Page 4823)

MC_Reset: Acknowledge fault V4 (Page 4827)
MC_Home: Home axes, set reference point V4 (Page 4829)
MC_Halt: Halt axes V4 (Page 4832)
MC_MoveAbsolute: Absolute positioning of axes V4 (Page 4835)
MC_MoveRelative: Relative positioning of axes V4 (Page 4838)
MC_MoveVelocity: Move axes at preset rotational speed V4 (Page 4842)
MC_MoveJog: Move axes in jog mode V4 (Page 4846)
MC_CommandTable: Run axis commands as motion sequence V4 (Page 4849)
MC_ReadParam: Continuously read motion data of a positioning axis V4 (Page 4853)
MC_WriteParam: Write tag of positioning axis V4 (Page 4855)
S7-1200 Motion Control V1...3 (Page 4856)

MC_ReadParam

MC_ReadParam: Continuously read motion data of a positioning axis V4

Description

Motion Control instruction "MC_ReadParam" enables continuous reading of motion data of an axis. The current value of the corresponding tag is determined at the start of the command.

The following motion data can be read:

- Current position of the axis
- Current velocity of the axis
- Current distance of axis from target position
- Target position of the axis

Requirements

The positioning axis technology object has been configured correctly.

Override response

A MC_ReadParam command cannot be aborted by any other Motion Control command.

A new MC_ReadParam command does not abort any active Motion Control commands.

Parameters

Parameter	Declaration	Data type	Default value	Description	
Enable	INPUT	BOOL	FALSE	TRUE	Read the tag specified with the "Parameter" and store the value in the destination address specified with "Value".
				FALSE	Do not update assigned motion data
Parameter	INPUT	VARIANT (REAL)	-	VARIANT pointer to the value to be read. The following tags are permitted: • <Axis name>.Position • <Axis name>.Velocity • <Axis name>.StatusPositioning.Distance • <Axis name>.StatusPositioning.TargetPosition	
Value	INOUT	VARIANT (REAL)	-	VARIANT pointer to the target tag or destination address to which the read value is to be written.	
Valid	OUTPUT	BOOL	FALSE	TRUE	The read value is valid.
				FALSE	The read value is invalid.
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 10132) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 10132) for parameter "ErrorID"	

See also

Overview of the Motion Control statements (Page 10096)

List of ErrorIDs and ErrorInfos (technology objects as of V4) (Page 10132)

S7-1200 Motion Control V1...3 (Page 4856)

MC_Power: Enable, disable axis V4 (Page 4823)

MC_Reset: Acknowledge fault V4 (Page 4827)

MC_Home: Home axes, set reference point V4 (Page 4829)

MC_Halt: Halt axes V4 (Page 4832)

MC_MoveAbsolute: Absolute positioning of axes V4 (Page 4835)

MC_MoveRelative: Relative positioning of axes V4 (Page 4838)

MC_MoveVelocity: Move axes at preset rotational speed V4 (Page 4842)

MC_MoveJog: Move axes in jog mode V4 (Page 4846)

MC_CommandTable: Run axis commands as motion sequence V4 (Page 4849)

MC_ChangeDynamic: Change dynamic settings of axis V4 (Page 4851)

MC_WriteParam: Write tag of positioning axis V4 (Page 4855)

MC_WriteParam

MC_WriteParam: Write tag of positioning axis V4

Description

Motion Control instruction "MC_WriteParam" enables the writing of tags of the positioning axis technology object in the user program. In contrast to the value assignment of the tags in the user program, "MC_WriteParam" can also change values of read-only tags.

You can learn about the tags, the conditions under which they can be written and the time at which they take effect in the description of the technology object tags (Page 10142).

Requirements

- The positioning axis technology object has been configured correctly.
- To write tags that are read-only in the user program, the axis must be disabled.

Override response

A MC_WriteParam command cannot be aborted by any other Motion Control command.

A new MC_WriteParam command does not abort any active Motion Control commands.

Parameters

Parameter	Declaration	Data type	Default value	Description	
Parameter	INPUT	VARIANT (BOOL, INT, DINT, REAL)	-	VARIANT pointer to the technology object tag (Page 10142) positioning axis (destination address) to be written	
Value	INPUT	VARIANT (BOOL, INT, DINT, REAL)	-	VARIANT pointer to the value to be written (source address)	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Done	OUTPUT	BOOL	FALSE	TRUE	Value was written
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 10132) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 10132) for parameter "ErrorID"	

See also

Overview of the Motion Control statements (Page 10096)
List of ErrorIDs and ErrorInfos (technology objects as of V4) (Page 10132)
Tags of the positioning axis technology object as of V4 (Page 10142)
S7-1200 Motion Control V1...3 (Page 4856)
MC_Power: Enable, disable axis V4 (Page 4823)
MC_Reset: Acknowledge fault V4 (Page 4827)
MC_Home: Home axes, set reference point V4 (Page 4829)
MC_Halt: Halt axes V4 (Page 4832)
MC_MoveAbsolute: Absolute positioning of axes V4 (Page 4835)
MC_MoveRelative: Relative positioning of axes V4 (Page 4838)
MC_MoveVelocity: Move axes at preset rotational speed V4 (Page 4842)
MC_MoveJog: Move axes in jog mode V4 (Page 4846)
MC_CommandTable: Run axis commands as motion sequence V4 (Page 4849)
MC_ChangeDynamic: Change dynamic settings of axis V4 (Page 4851)
MC_ReadParam: Continuously read motion data of a positioning axis V4 (Page 4853)

S7-1200 Motion Control V1...3

MC_Power

MC_Power: Enable, disable axis V1...3

Description

The Motion Control instruction "MC_Power" enables or disables an axis.

Requirements

- The axis technology object has been configured correctly.
- There is no pending enable-inhibiting error.

Override response

Execution of "MC_Power" cannot be aborted by a Motion Control command.

Disabling the axis (input parameter "Enable" = FALSE) aborts all Motion Control commands for the associated technology object in accordance with the selected "StopMode".

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
Enable	INPUT	BOOL	FALSE	TRUE	Motion Control attempts to enable the axis.
				FALSE	All current jobs are interrupted in accordance with the "StopMode" configured. The axis is stopped and disabled.
StopMode	INPUT	INT	0	0	Emergency stop If a request to disable the axis is pending, the axis brakes at the configured emergency stop deceleration. The axis is disabled after reaching standstill.
				1	Immediate stop If a request to disable the axis is pending, this axis is disabled without deceleration. The pulse output is stopped immediately.
				2	Emergency stop with jerk control If a request to disable the axis is pending, the axis brakes at the configured emergency stop deceleration. If the jerk control is activated, the configured jerk is taken into account. The axis is disabled after reaching standstill.
Status	OUTPUT	BOOL	FALSE	Status of axis enable	
				FALSE	The axis is disabled. The axis does not execute Motion Control commands and does not accept any new commands (exception: MC_Reset command). The axis is not homed. Upon disabling, the status does not change to FALSE until the axis reaches a standstill.
				TRUE	The axis is enabled. The axis is ready to execute Motion Control commands. Upon axis enabling, the status does not change to TRUE until the signal "Drive ready" is pending. If the "Drive ready" drive interface was not configured in the axis configuration, the status changes to TRUE immediately.
Busy	OUTPUT	BOOL	FALSE	TRUE	"MC_Power" is active.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred in Motion Control instruction "MC_Power" or in the associated technology object. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 10177) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 10177) for parameter "ErrorID"	

Note

If the axis is switched off due to an error, it will be enabled again automatically after the error has been eliminated and acknowledged. This requires that the input parameter "Enable" has retained the value TRUE during this process.

Enabling an axis with configured drive interface

To enable the axis, follow these steps:

1. Check the requirements indicated above.
2. Initialize input parameter "StopMode" with the desired value. Set the input parameter "Enable" to TRUE.
   The enable output for "Drive enabled" changes to TRUE to enable the power to the drive. The CPU waits for the "Drive ready" signal of the drive.
   When the "Drive ready" signal is available at the configured Ready input of the CPU, the axis is enabled. The output parameter "Status" and the tag of the technology object <Axis name>.StatusBits.Enable indicate the value TRUE.

Enabling an axis without configured drive interface

To enable the axis, follow these steps:

1. Check the requirements indicated above.
2. Initialize input parameter "StopMode" with the desired value. Set the input parameter "Enable" to TRUE. The axis is enabled. The output parameter "Status" and the tag of the technology object <Axis name>.StatusBits.Enable indicate the value TRUE.

Disabling an axis

To disable an axis, you can follow the steps described below:

1. Bring the axis to a standstill.
   You can identify when the axis is at a standstill in the tag of the technology object <Axis name>.StatusBits.StandStill.
2. Set input parameter "Enable" to FALSE after standstill is reached.
3. If output parameters "Busy" and "Status" and the tag of technology object <Axis name>.StatusBits.Enable indicate the value FALSE, disabling of the axis is complete.

See also

Overview of the Motion Control statements (Page 10096)

S7-1200 Motion Control V4 (Page 4823)

ErrorIDs and ErrorInfos (Page 10177)

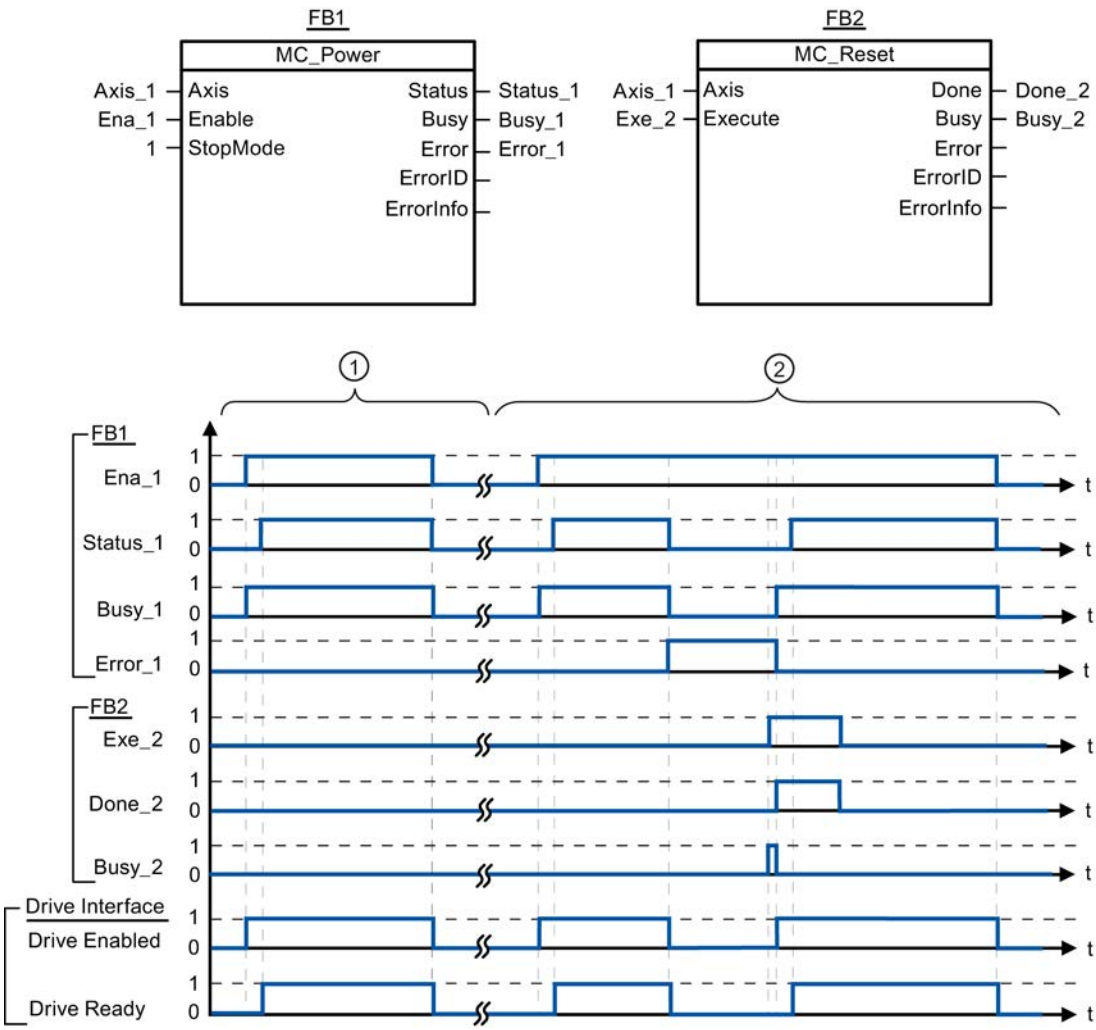
MC_Power: Function chart V1...3 (Page 4859)

MC_Reset: Acknowledge fault V1...3 (Page 4860)

MC_Home: Home axes, set reference point V1...3 (Page 4862)
MC_Halt: Halt axes V1...3 (Page 4865)
MC_MoveAbsolute: Absolute positioning of axes V1...3 (Page 4868)
MC_MoveRelative: Relative positioning of axes V1...3 (Page 4871)
MC_MoveVelocity: Move axes at preset rotational speed V1...3 (Page 4874)
MC_MoveJog: Move axes in jog mode V1...3 (Page 4878)
MC_CommandTable: Run axis commands as motion sequence V2...3 (Page 4881)
MC_ChangeDynamic: Change dynamic settings of axis V2...3 (Page 4883)

MC_Power: Function chart V1...3

Function chart



①	An axis is enabled and then disabled again. When the drive has signaled "Drive ready" back to the CPU, the successful enable can be read out via "Status_1".
②	Following an axis enable, an error has occurred that caused the axis to be disabled. The error is eliminated and acknowledged with "MC_Reset". The axis is then enabled again.

See also

MC_Power: Enable, disable axis V1...3 (Page 4856)

MC_Reset

MC_Reset: Acknowledge fault V1...3

Description

Motion Control instruction "MC_Reset" can be used to acknowledge "Operating error with axis stop" and "Configuration error". The errors that require acknowledgement can be found in the "List of ErrorIDs and ErrorInfos" under "Remedy".

From version V3.0, the axis configuration can be downloaded to the work memory in RUN operating mode.

Requirements

- The axis technology object has been configured correctly.
- The cause of a pending configuration error requiring acknowledgement has been eliminated (for example, acceleration in positioning axis technology object has been changed to a valid value).

Override response

The MC_Reset command cannot be aborted by any other Motion Control command.

The new MC_Reset command does not abort any other active Motion Control commands.

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Restart	INPUT	BOOL	FALSE	(From version V3.0)	
				TRUE	Download the axis configuration from the load memory to the work memory. The command can only be executed when the axis is disabled. Refer to the notes on Download to the CPU (Page 10074).

Parameter	Declaration	Data type	Default value	Description	
				FALSE	Acknowledges pending errors
Done	OUTPUT	BOOL	FALSE	TRUE	Error has been acknowledged.
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 10177) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 10177) for parameter "ErrorID"	

Acknowledging an error requiring acknowledgement with MC_Reset

To acknowledge an error, follow these steps:

1. Check the requirements indicated above.
2. Start the acknowledgement of the error with a rising edge at input parameter "Execute".
3. If output parameter "Done" indicates the value TRUE and tag of technology object <Axis name>.StatusBits.Error the value FALSE, the error has been acknowledged.

See also

Overview of the Motion Control statements (Page 10096)

S7-1200 Motion Control V4 (Page 4823)

ErrorIDs and ErrorInfos (Page 10177)

Download to CPU (Page 10074)

MC_Power: Enable, disable axis V1...3 (Page 4856)

MC_Home: Home axes, set reference point V1...3 (Page 4862)

MC_Halt: Halt axes V1...3 (Page 4865)

MC_MoveAbsolute: Absolute positioning of axes V1...3 (Page 4868)

MC_MoveRelative: Relative positioning of axes V1...3 (Page 4871)

MC_MoveVelocity: Move axes at preset rotational speed V1...3 (Page 4874)

MC_MoveJog: Move axes in jog mode V1...3 (Page 4878)

MC_CommandTable: Run axis commands as motion sequence V2...3 (Page 4881)

MC_ChangeDynamic: Change dynamic settings of axis V2...3 (Page 4883)

MC_Home

MC_Home: Home axes, set reference point V1...3

Description

Motion Control instruction "MC_Home" is used to match the axis coordinates to the real, physical drive position. Homing is required for absolute positioning of the axis. The following types of homing can be executed:

- Active homing (Mode = 3)
  The homing procedure is executed automatically.
- Passive homing (Mode = 2)
  During passive homing, the "MC_Home" Motion Control instruction does not carry out any homing motion. The traversing motion required for this must be implemented by the user via other Motion Control instructions. When the homing switch is detected, the axis is homed.
- Direct homing absolute (Mode = 0)
  The current axis position is set to the value of parameter "Position".
- Direct homing relative (Mode = 1)
  The current axis position is offset by the value of parameter "Position".

Requirements

- The axis technology object has been configured correctly.
- The axis is enabled.
- No MC_CommandTable command may be active upon start with Mode = 0, 1, or 2.

Override response

The override response depends on the selected mode:

Mode = 0, 1

The MC_Home command cannot be aborted by any other Motion Control command.

The MC_Home command does not abort any active Motion Control commands. Position-related motion commands are resumed after homing according to the new homing position (value at input parameter: "Position").

Mode = 2

The MC_Home command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 2, 3

The new MC_Home command aborts the following active Motion Control command:

- MC_Home command Mode = 2

Position-related motion commands are resumed after homing according to the new homing position (value at input parameter: "Position").

Mode = 3

The MC_Home command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_Home command aborts the following active Motion Control commands:

- MC_Home command Mode = 2, 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Position	INPUT	REAL	0.0	• Mode = 0, 2, and 3 Absolute position of axis after completion of the homing operation • Mode = 1 Correction value for the current axis position Limit values: $-1.0e^{12} \leq$ Position $\leq 1.0e^{12}$	
Mode	INPUT	INT	0	Homing mode	
				0	Direct homing absolute New axis position is the position value of parameter "Position".
				1	Direct homing relative New axis position is the current axis position + position value of parameter "Position".

Parameter	Declaration	Data type	Default value	Description	
				2	Passive homing Homing according to the axis configuration. Following homing, the value of parameter "Position" is set as the new axis position.
				3	Active homing Homing procedure in accordance with the axis configuration. Following homing, the value of parameter "Position" is set as the new axis position.
Done	OUTPUT	BOOL	FALSE	TRUE	Command completed
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 10177) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 10177) for parameter "ErrorID"	

Note

Axis homing is lost under the following conditions:

- Disabling of axis by the "MC_Power" Motion Control instruction
- Changeover between automatic mode and manual control
- Upon start of active homing. After successful completion of the homing operation, axis homing is again available.
- After POWER OFF -> POWER ON of the CPU
- After CPU restart (RUN-STOP -> STOP-RUN)

Homing an axis

To home the axis, follow these stops:

1. Check the requirements indicated above.
2. Initialize the necessary input parameters with values, and start the homing operation with a rising edge at input parameter "Execute"
3. If output parameter "Done" and technology object tag <Axis name>.StatusBits.HomingDone indicate the value TRUE, homing is complete.

See also

Overview of the Motion Control statements (Page 10096)

S7-1200 Motion Control V4 (Page 4823)

ErrorIDs and ErrorInfos (Page 10177)
MC_Power: Enable, disable axis V1...3 (Page 4856)
MC_Reset: Acknowledge fault V1...3 (Page 4860)
MC_Halt: Halt axes V1...3 (Page 4865)
MC_MoveAbsolute: Absolute positioning of axes V1...3 (Page 4868)
MC_MoveRelative: Relative positioning of axes V1...3 (Page 4871)
MC_MoveVelocity: Move axes at preset rotational speed V1...3 (Page 4874)
MC_MoveJog: Move axes in jog mode V1...3 (Page 4878)
MC_CommandTable: Run axis commands as motion sequence V2...3 (Page 4881)
MC_ChangeDynamic: Change dynamic settings of axis V2...3 (Page 4883)

MC_Halt

MC_Halt: Halt axes V1...3

Description

The "MC_Halt" Motion Control instruction stops all movements and brings the axis to a standstill with the configured deceleration. The standstill position is not defined.

Requirements

- The axis technology object has been configured correctly.
- The axis is enabled.

Override response

The MC_Halt command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_Halt command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command

- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Parameters

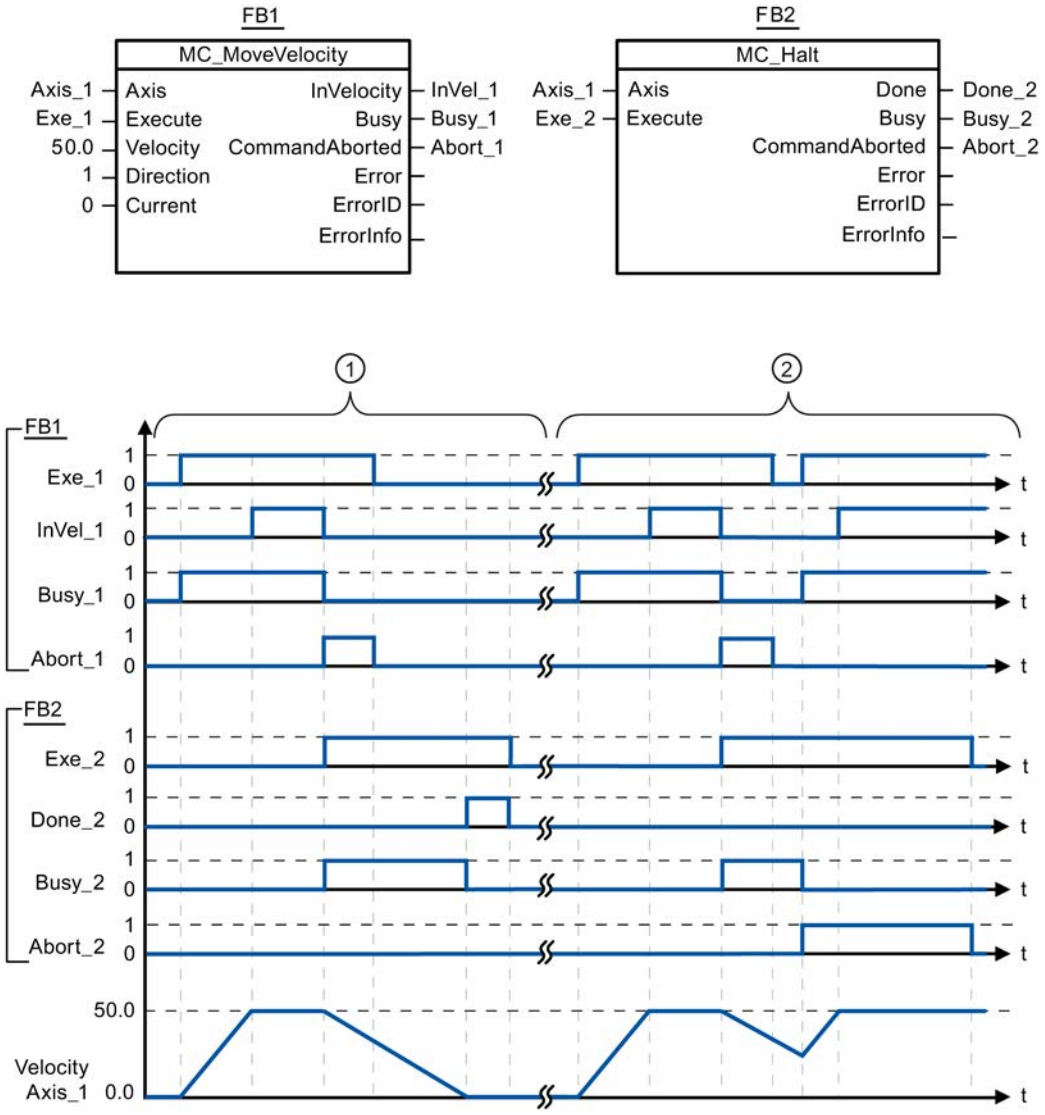
Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Done	OUTPUT	BOOL	FALSE	TRUE	Zero velocity reached
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 10177) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 10177) for parameter "ErrorID"	

See also

Overview of the Motion Control statements (Page 10096)
S7-1200 Motion Control V4 (Page 4823)
ErrorIDs and ErrorInfos (Page 10177)
MC_Halt: Function chart V1...3 (Page 4867)
MC_Power: Enable, disable axis V1...3 (Page 4856)
MC_Reset: Acknowledge fault V1...3 (Page 4860)
MC_Home: Home axes, set reference point V1...3 (Page 4862)
MC_MoveAbsolute: Absolute positioning of axes V1...3 (Page 4868)
MC_MoveRelative: Relative positioning of axes V1...3 (Page 4871)
MC_MoveVelocity: Move axes at preset rotational speed V1...3 (Page 4874)
MC_MoveJog: Move axes in jog mode V1...3 (Page 4878)
MC_CommandTable: Run axis commands as motion sequence V2...3 (Page 4881)
MC_ChangeDynamic: Change dynamic settings of axis V2...3 (Page 4883)

MC_Halt: Function chart V1...3

Function chart



The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 5.0

①	The axis is braked by an MC_Halt command until it comes to a standstill. The axis standstill is signaled via "Done_2".
②	While an MC_Halt command is braking the axis, this command is aborted by another motion command. The abort is signaled via "Abort_2".

See also

MC_Halt: Halt axes V1...3 (Page 4865)

MC_MoveAbsolute

MC_MoveAbsolute: Absolute positioning of axes V1...3

Description

The "MC_MoveAbsolute" Motion Control instruction starts an axis positioning motion to move it to an absolute position.

Requirements

- The axis technology object has been configured correctly.
- The axis is enabled.
- The axis is homed.

Override response

The MC_MoveAbsolute command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_MoveAbsolute command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Parameters

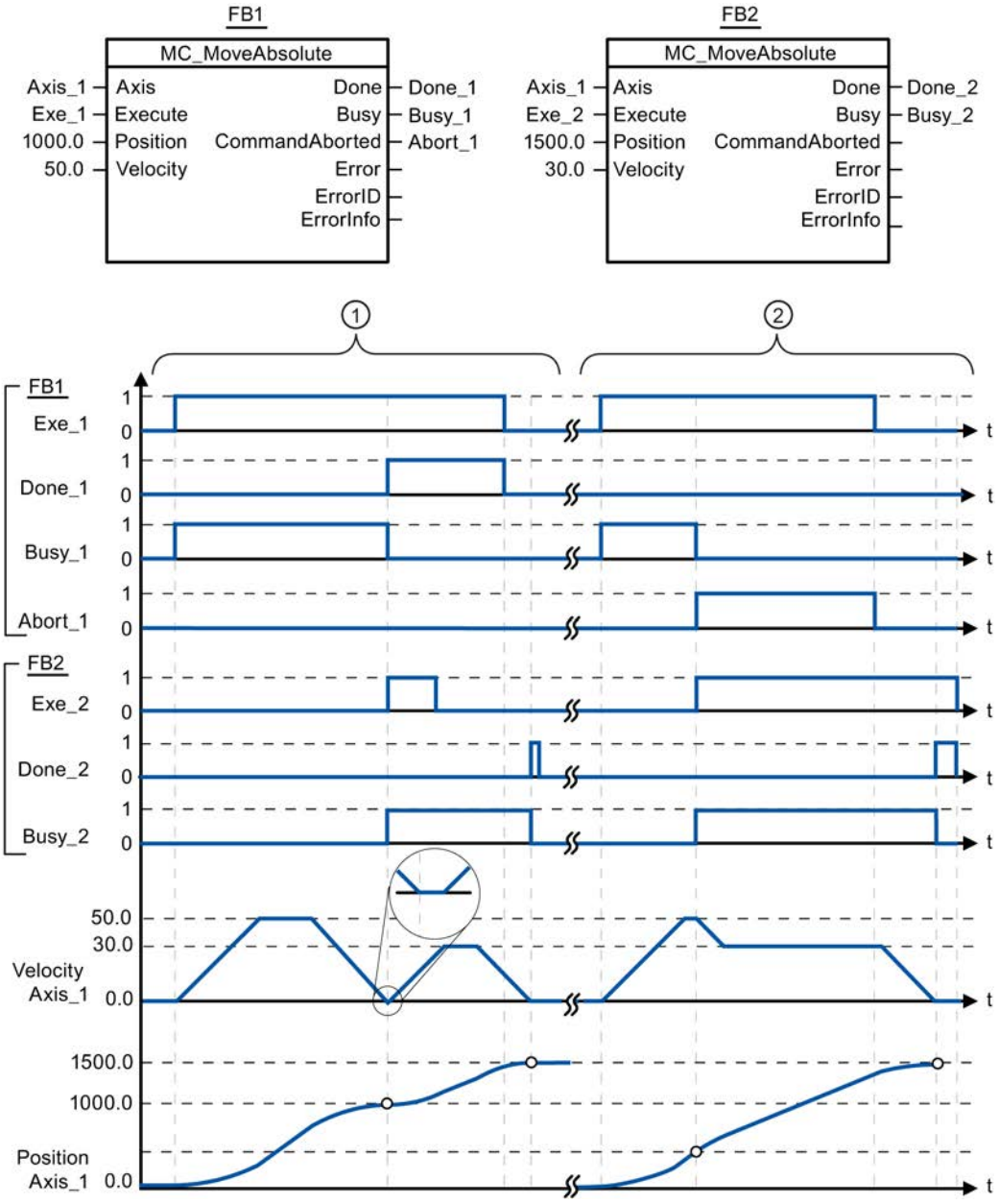
Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Position	INPUT	REAL	0.0	Absolute target position Limit values: $-1.0e^{12} \leq$ Position $\leq 1.0e^{12}$	
Velocity	INPUT	REAL	10.0	Velocity of axis This velocity is not always reached on account of the configured acceleration and deceleration and the target position to be approached. Limit values: Start/stop velocity ≤ Velocity ≤ maximum velocity	
Done	OUTPUT	BOOL	FALSE	TRUE	Absolute target position reached
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 10177) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 10177) for parameter "ErrorID"	

See also

Overview of the Motion Control statements (Page 10096)
S7-1200 Motion Control V4 (Page 4823)
ErrorIDs and ErrorInfos (Page 10177)
MC_MoveAbsolute: Function chart V1...3 (Page 4870)
MC_Power: Enable, disable axis V1...3 (Page 4856)
MC_Reset: Acknowledge fault V1...3 (Page 4860)
MC_Home: Home axes, set reference point V1...3 (Page 4862)
MC_Halt: Halt axes V1...3 (Page 4865)
MC_MoveRelative: Relative positioning of axes V1...3 (Page 4871)
MC_MoveVelocity: Move axes at preset rotational speed V1...3 (Page 4874)
MC_MoveJog: Move axes in jog mode V1...3 (Page 4878)
MC_CommandTable: Run axis commands as motion sequence V2...3 (Page 4881)
MC_ChangeDynamic: Change dynamic settings of axis V2...3 (Page 4883)

MC_MoveAbsolute: Function chart V1...3

Function chart



The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 10.0

①	An axis is moved to absolute position 1000.0 with an MC_MoveAbsolute command. When the axis reaches the target position, this is signaled via "Done_1". When "Done_1" = TRUE, another MC_MoveAbsolute command, with target position 1500.0, is started. Because of the response times (e.g., cycle time of user program, etc.), the axis comes to a standstill briefly (see zoomed-in detail). When the axis reaches the new target position, this is signaled via "Done_2".
②	An active MC_MoveAbsolute command is aborted by another MC_MoveAbsolute command. The abort is signaled via "Abort_1". The axis is then moved at the new velocity to the new target position 1500.0. When the new target position is reached, this is signaled via "Done_2".

See also

MC_MoveAbsolute: Absolute positioning of axes V1...3 (Page 4868)

MC_MoveRelative

MC_MoveRelative: Relative positioning of axes V1...3

Description

The "MC_MoveRelative" Motion Control instruction starts a positioning motion relative to the start position.

Requirements

- The axis technology object has been configured correctly.
- The axis is enabled.

Override response

The MC_MoveRelative command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_MoveRelative command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command

- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Distance	INPUT	REAL	0.0	Travel distance for the positioning operation Limit values: $-1.0e^{12} \leq$ Distance $\leq 1.0e^{12}$	
Velocity	INPUT	REAL	10.0	Velocity of axis This velocity is not always reached on account of the configured acceleration and deceleration and the distance to be traveled. Limit values: Start/stop velocity ≤ Velocity ≤ maximum velocity	
Done	OUTPUT	BOOL	FALSE	TRUE	Target position reached
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 10177) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 10177) for parameter "ErrorID"	

See also

Overview of the Motion Control statements (Page 10096)

S7-1200 Motion Control V4 (Page 4823)

ErrorIDs and ErrorInfos (Page 10177)

MC_MoveRelative: Function chart V1...3 (Page 4873)

MC_Power: Enable, disable axis V1...3 (Page 4856)

MC_Reset: Acknowledge fault V1...3 (Page 4860)

MC_Home: Home axes, set reference point V1...3 (Page 4862)

MC_Halt: Halt axes V1...3 (Page 4865)

MC_MoveAbsolute: Absolute positioning of axes V1...3 (Page 4868)

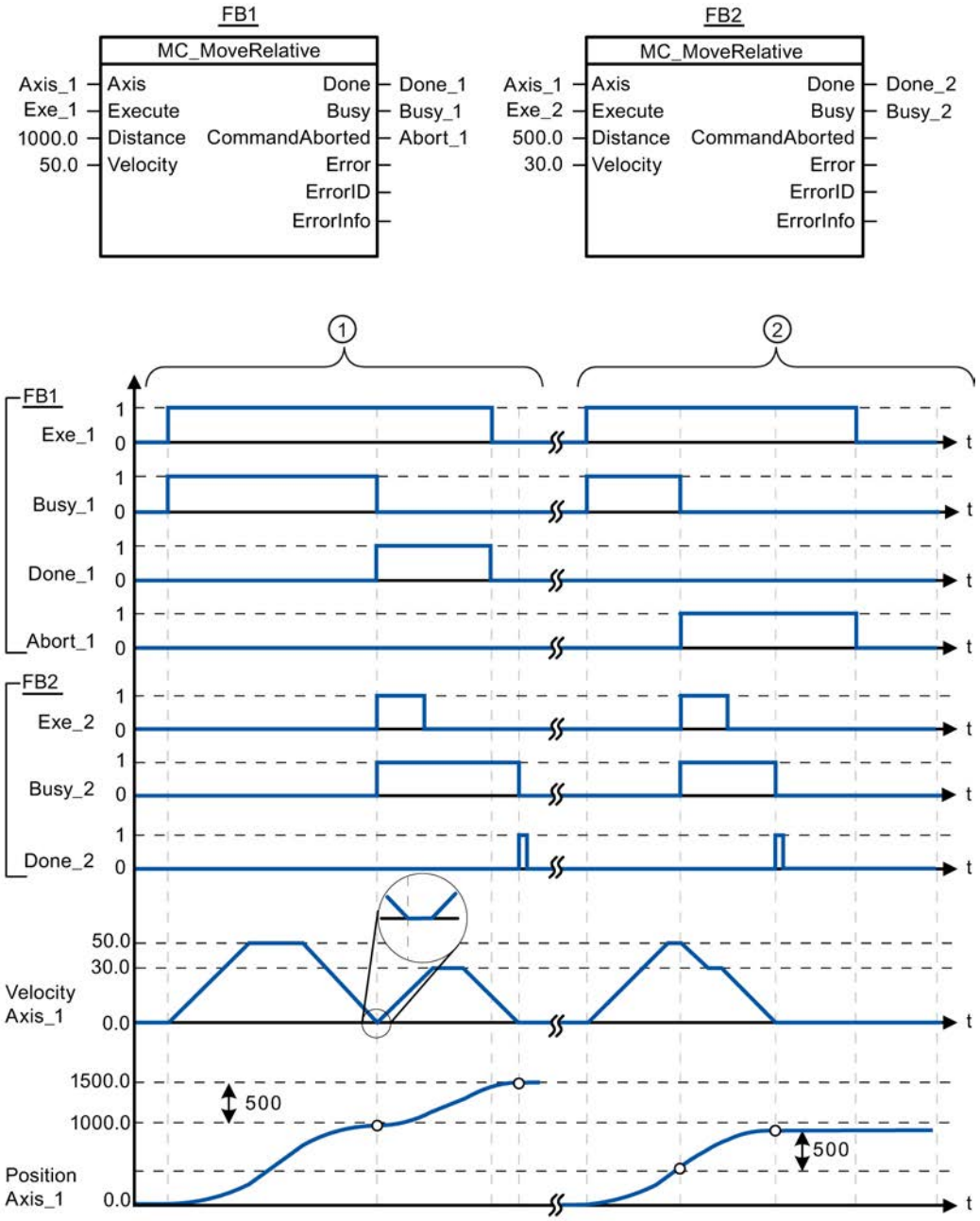
MC_MoveVelocity: Move axes at preset rotational speed V1...3 (Page 4874)

MC_MoveJog: Move axes in jog mode V1...3 (Page 4878)

MC_CommandTable: Run axis commands as motion sequence V2...3 (Page 4881)
MC_ChangeDynamic: Change dynamic settings of axis V2...3 (Page 4883)

MC_MoveRelative: Function chart V1...3

Function chart



The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 10.0

①	The axis is moved by an MC_MoveRelative command by the distance ("Distance") 1000.0. When the axis reaches the target position, this is signaled via "Done_1". When "Done_1" = TRUE, another MC_MoveRelative command, with travel distance 500.0, is started. Because of the response times (e.g., cycle time of user program, etc.), the axis comes to a standstill briefly (see zoomed-in detail). When the axis reaches the new target position, this is signaled via "Done_2".
②	An active MC_MoveRelative command is aborted by another MC_MoveRelative command. The abort is signaled via "Abort_1". The axis is then moved at the new velocity by the new distance ("Distance") 500.0. When the new target position is reached, this is signaled via "Done_2".

See also

MC_MoveRelative: Relative positioning of axes V1...3 (Page 4871)

MC_MoveVelocity

MC_MoveVelocity: Move axes at preset rotational speed V1...3

Description

Motion control instruction "MC_MoveVelocity" moves the axis constantly at the specified velocity.

Requirements

- The axis technology object has been configured correctly.
- The axis is enabled.

Override response

MC_MoveVelocity can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_MoveVelocity command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command

- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
Velocity	INPUT	REAL	10.0	Velocity specification for axis motion Limit values: Start/stop velocity ≤ |Velocity ≤ maximum velocity (Velocity = 0.0 is permitted)	
Direction	INPUT	INT	0	Direction specification	
				0	Direction of rotation corresponds to the sign of the value in parameter "Velocity"
				1	Positive direction of rotation (The sign of the value in parameter "Velocity" is ignored)
				2	Negative direction of rotation (The sign of the value in parameter "Velocity" is ignored)
Current	INPUT	BOOL	FALSE	Maintain current velocity	
				FALSE	"Maintain current velocity" is deactivated. The values of parameters "Velocity" and "Direction" are used.
				TRUE	"Maintain current velocity" is activated. The values in parameters "Velocity" and "Direction" are not taken into account. When the axis resumes motion at the current velocity, the "InVelocity" parameter returns the value TRUE.
InVelocity	OUTPUT	BOOL	FALSE	TRUE	• "Current" = FALSE: The velocity specified in parameter "Velocity" was reached. • "Current" = TRUE: The axis travels at the current velocity at the start time.
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the command was aborted by another command.

Parameter	Declaration	Data type	Default value	Description	
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 10177) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 10177) for parameter "ErrorID"	

Behavior with zero setpoint velocity (Velocity = 0.0)

An MC_MoveVelocity command with "Velocity" = 0.0 (such as an MC_Halt command) aborts active motion commands and stops the axis with the configured deceleration.

When the axis comes to a standstill, output parameter "InVelocity" indicates TRUE for at least one program cycle.

"Busy" indicates the value TRUE during the deceleration process and changes to FALSE together with "InVelocity". If parameter "Execute" = TRUE is set, "InVelocity" and "Busy" are latched.

When the "MC_MoveVelocity" command is started, status bit "SpeedCommand" is set in the technology object. Status bit "ConstantVelocity" is set upon axis standstill. Both bits are adapted to the new situation when a new motion command is started.

See also

Overview of the Motion Control statements (Page 10096)

S7-1200 Motion Control V4 (Page 4823)

ErrorIDs and ErrorInfos (Page 10177)

MC_MoveVelocity: Function chart V1...3 (Page 4877)

MC_Power: Enable, disable axis V1...3 (Page 4856)

MC_Reset: Acknowledge fault V1...3 (Page 4860)

MC_Home: Home axes, set reference point V1...3 (Page 4862)

MC_Halt: Halt axes V1...3 (Page 4865)

MC_MoveAbsolute: Absolute positioning of axes V1...3 (Page 4868)

MC_MoveRelative: Relative positioning of axes V1...3 (Page 4871)

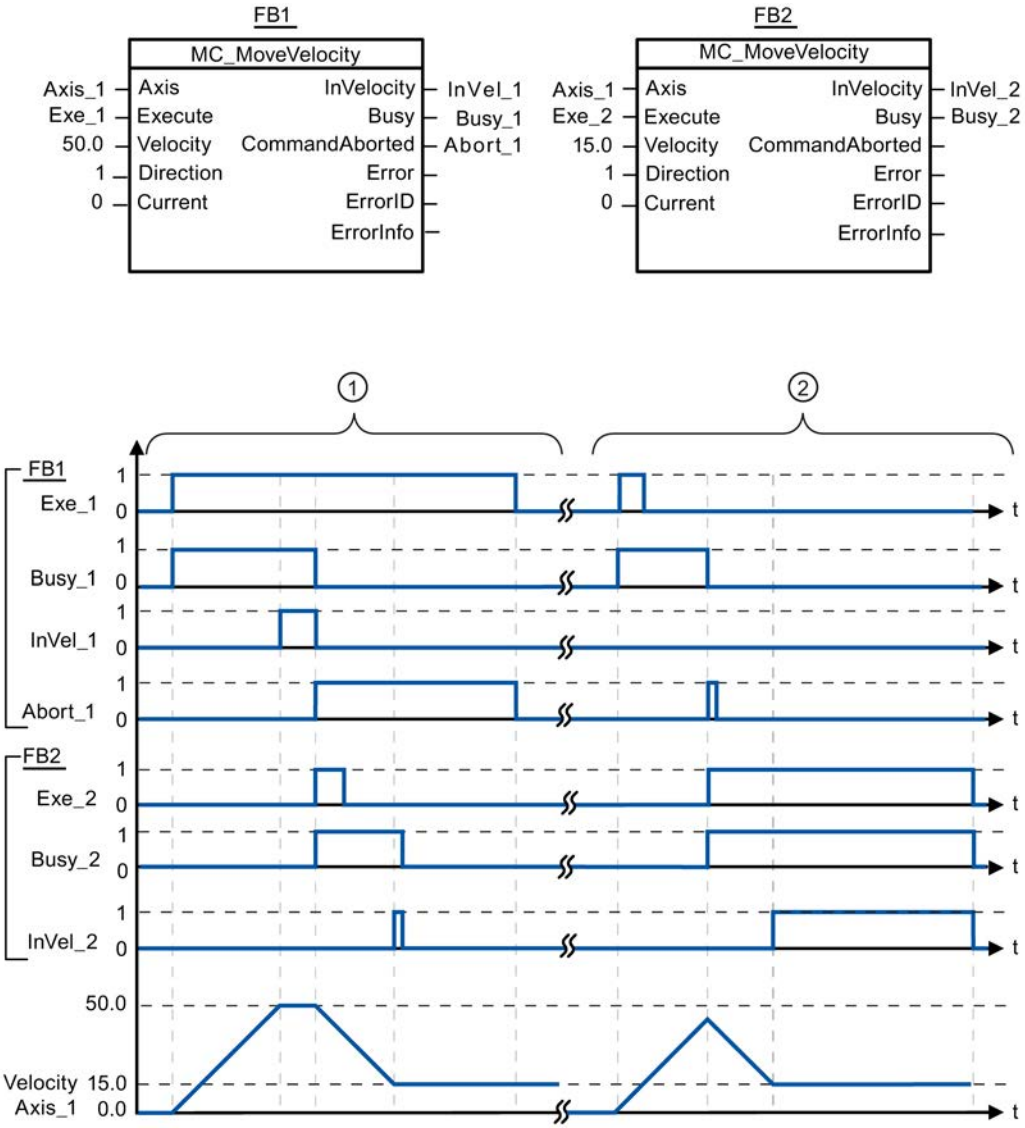
MC_MoveJog: Move axes in jog mode V1...3 (Page 4878)

MC_CommandTable: Run axis commands as motion sequence V2...3 (Page 4881)

MC_ChangeDynamic: Change dynamic settings of axis V2...3 (Page 4883)

MC_MoveVelocity: Function chart V1...3

Function chart



The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 10.0

①	An active MC_MoveVelocity command signals via "InVel_1" that its target velocity has been reached. It is then aborted by another MC_MoveVelocity command. The abort is signaled via "Abort_1". When the new target velocity 15.0 is reached, this is signaled via "InVel_2". The axis then continues moving at the new constant velocity.
②	An active MC_MoveVelocity command is aborted by another MC_MoveVelocity command prior to reaching its target velocity. The abort is signaled via "Abort_1". When the new target velocity 15.0 is reached, this is signaled via "InVel_2". The axis then continues moving at the new constant velocity.

See also

MC_MoveVelocity: Move axes at preset rotational speed V1...3 (Page 4874)

MC_MoveJog

MC_MoveJog: Move axes in jog mode V1...3

Description

Motion control instruction "MC_MoveJog" moves the axis constantly at the specified velocity in jog mode. You use this Motion Control instruction, for example, for testing and commissioning purposes.

Requirements

- The axis technology object has been configured correctly.
- The axis is enabled.

Override response

The MC_MoveJog command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_MoveJog command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
JogForward	INPUT	BOOL	FALSE	As long as the parameter is TRUE, the axis moves in the positive direction at the velocity specified in parameter "Velocity".	
JogBackward	INPUT	BOOL	FALSE	As long as the parameter is TRUE, the axis moves in the negative direction at the velocity specified in parameter "Velocity".	
If both parameters are simultaneously TRUE, the axis stops with the configured deceleration. An error is indicated in parameters "Error", "ErrorID", and "ErrorInfo".					
Velocity	INPUT	REAL	10.0	Preset velocity for jog mode	
				Limit values, instruction version V1.0: Start/stop velocity ≤ |Velocity ≤ maximum velocity	
				Limits, instruction version V2.0: Start/stop velocity ≤ velocity ≤ maximum velocity	
InVelocity	OUTPUT	BOOL	FALSE	TRUE	The velocity specified in parameter "Velocity" was reached.
Busy	OUTPUT	BOOL	FALSE	TRUE	The command is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the command was aborted by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 10177) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 10177) for parameter "ErrorID"	

See also

Overview of the Motion Control statements (Page 10096)

S7-1200 Motion Control V4 (Page 4823)

ErrorIDs and ErrorInfos (Page 10177)

MC_MoveJog: Function chart V1...3 (Page 4880)

MC_Power: Enable, disable axis V1...3 (Page 4856)

MC_Reset: Acknowledge fault V1...3 (Page 4860)

MC_Home: Home axes, set reference point V1...3 (Page 4862)

MC_Halt: Halt axes V1...3 (Page 4865)

MC_MoveAbsolute: Absolute positioning of axes V1...3 (Page 4868)

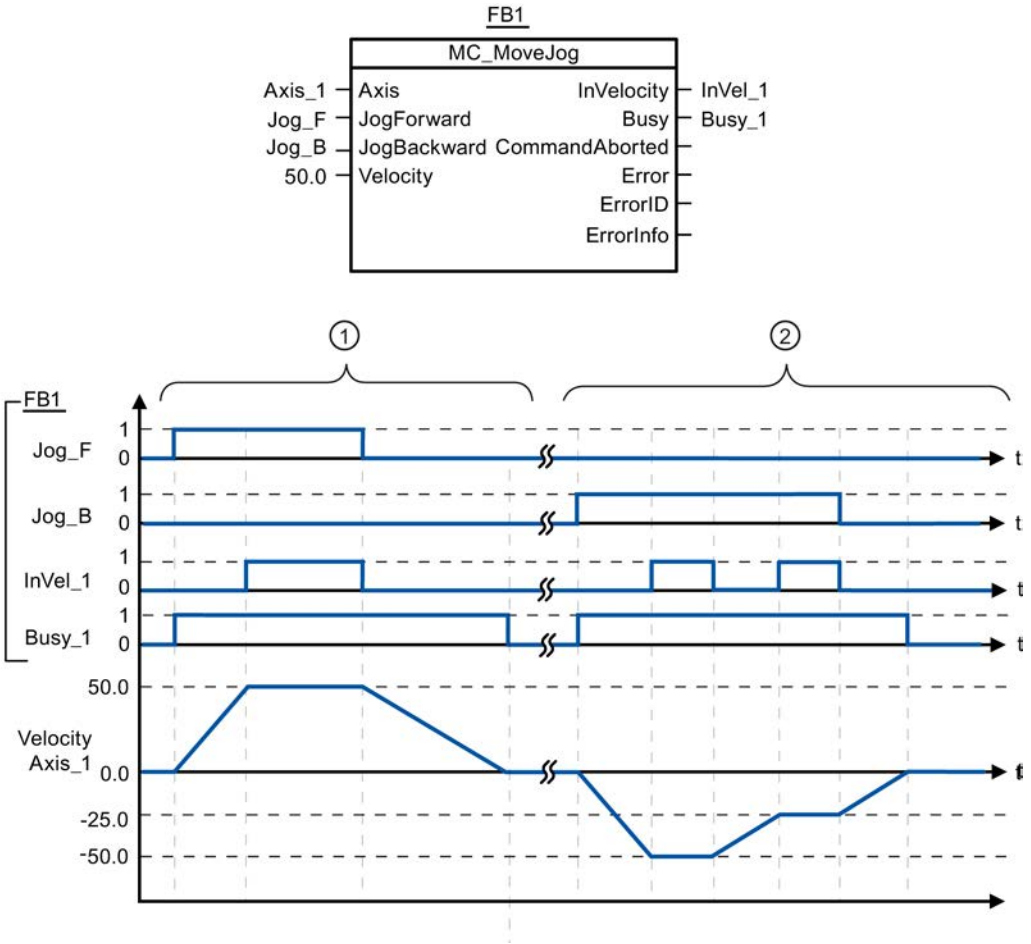
MC_MoveRelative: Relative positioning of axes V1...3 (Page 4871)

MC_MoveVelocity: Move axes at preset rotational speed V1...3 (Page 4874)

MC_CommandTable: Run axis commands as motion sequence V2...3 (Page 4881)
MC_ChangeDynamic: Change dynamic settings of axis V2...3 (Page 4883)

MC_MoveJog: Function chart V1...3

Function chart



The following values were configured in the configuration window **Dynamics > General**:

- Acceleration: 10.0
- Deceleration: 5.0

①	The axis is moved in the positive direction in jog mode via "Jog_F". When the target velocity 50.0 is reached, this is signaled via "InVelo_1". After " Jog_F" is reset, the axis brakes again until it comes to a standstill.
②	The axis is moved in the negative direction in jog mode via "Jog_B". When the target velocity 50.0 is reached, this is signaled via "InVelo_1". After " Jog_B" is reset, the axis brakes again until it comes to a standstill.

See also

MC_MoveJog: Move axes in jog mode V1...3 (Page 4878)

MC_CommandTable

MC_CommandTable: Run axis commands as motion sequence V2...3

Description

The Motion Control instruction "MC_CommandTable" combines multiple individual axis control commands in one movement sequence.

Requirements

- The axis technology object has been inserted in V2 and correctly configured.
- The command table technology object has been inserted and correctly configured.
- The axis is enabled.

Override response

The MC_CommandTable command can be aborted by the following Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The new MC_CommandTable command aborts the following active Motion Control commands:

- MC_Home command Mode = 3
- MC_Halt command
- MC_MoveAbsolute command
- MC_MoveRelative command
- MC_MoveVelocity command
- MC_MoveJog command
- MC_CommandTable command

The active Motion Control command is cancelled by the start of the first "Positioning Relative", "Positioning Absolute", "Velocity set point" or "Halt" command.

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
CommandTable	INPUT	TO_CommandTable_1	-	Command table technology object	
Execute	INPUT	BOOL	FALSE	Command table start with positive edge	
StartStep	INPUT	INT	1	Defines the step at which the execution of the command table should begin Limit values: 1 ≤ StartStep ≤ EndStep	
EndStep	INPUT	INT	32	Defines the step up to which the execution of command table should take place Limit values: StartStep ≤ EndStep ≤ 32	
Done	OUTPUT	BOOL	FALSE	TRUE	Command table has been successfully executed
Busy	OUTPU	BOOL	FALSE	TRUE	The command table is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	The command table was cancelled by another command.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command table. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 10177) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 10177) for parameter "ErrorID"	
CurrentStep	OUTPUT	INT	0	Step in command table currently being executed	
StepCode	OUTPUT	WORD	16#0000	User-defined numerical value / bit pattern of the step currently being executed	

See also

Overview of the Motion Control statements (Page 10096)

S7-1200 Motion Control V4 (Page 4823)

ErrorIDs and ErrorInfos (Page 10177)

MC_Power: Enable, disable axis V1...3 (Page 4856)

MC_Reset: Acknowledge fault V1...3 (Page 4860)

MC_Home: Home axes, set reference point V1...3 (Page 4862)

MC_Halt: Halt axes V1...3 (Page 4865)

MC_MoveAbsolute: Absolute positioning of axes V1...3 (Page 4868)

MC_MoveRelative: Relative positioning of axes V1...3 (Page 4871)

MC_MoveVelocity: Move axes at preset rotational speed V1...3 (Page 4874)

MC_MoveJog: Move axes in jog mode V1...3 (Page 4878)

MC_ChangeDynamic: Change dynamic settings of axis V2...3 (Page 4883)

MC_ChangeDynamic

MC_ChangeDynamic: Change dynamic settings of axis V2...3

Description

Motion Control instruction "MC_ChangeDynamic" allows you to change the following settings of the axis:

- Change the ramp-up time (acceleration) value
- Change the ramp-down time (deceleration) value
- Change the emergency stop ramp-down time (emergency stop deceleration) value
- Change the smoothing time (jerk) value

For the effectiveness of the change, refer to the description of the tag (Page 10190).

Requirements

- The axis technology object has been inserted in Version V2.
- The axis technology object has been configured correctly.

Override response

A MC_ChangeDynamic command cannot be aborted by any other Motion Control command.

A new MC_ChangeDynamic command does not abort any active Motion Control commands.

Parameters

Parameter	Declaration	Data type	Default value	Description	
Axis	INPUT	TO_Axis_1	-	Axis technology object	
Execute	INPUT	BOOL	FALSE	Start of the command with a positive edge	
ChangeRampUp	INPUT	BOOL	FALSE	TRUE	Change ramp-up time in line with input parameter "RampUpTime"
RampUpTime	INPUT	REAL	5.00	Time (in seconds) to accelerate axis from standstill to configured maximum velocity without jerk limit. The change will influence the tag <Axis name>. Config.DynamicDefaults.Acceleration. For the effectiveness of the change, refer to the description of this tag.	
ChangeRampDown	INPUT	BOOL	FALSE	TRUE	Change ramp-down time in line with input parameter "RampDownTime"
RampDownTime	INPUT	REAL	5.00	Time (in seconds) to decelerate axis from the configured maximum velocity to standstill without jerk limiter. The change will influence the tag <Axis name>. Config.DynamicDefaults.Deceleration . For the effectiveness of the change, refer to the description of this tag.	

Parameter	Declaration	Data type	Default value	Description	
ChangeEmergency	INPUT	BOOL	FALSE	TRUE	Change emergency stop ramp-down time in line with input parameter "EmergencyRampTime"
EmergencyRampTime	INPUT	REAL	2.00	Time (in seconds) to decelerate the axis from configured maximum velocity to standstill without jerk limiter in emergency stop mode. The change will influence the tag <Axis name>. Config.DynamicDefaults.EmergencyDeceleration . For the effectiveness of the change, refer to the description of this tag.	
ChangeJerkTime	INPUT	BOOL	FALSE	TRUE	Change smoothing time according to the input parameter "JerkTime"
JerkTime	INPUT	REAL	0.25	Smoothing time (in seconds) used for the axis acceleration and deceleration ramps The change will influence the tag <Axis name>. Config.DynamicDefaults.Jerk . For the effectiveness of the change, refer to the description of this tag.	
Done	OUTPUT	BOOL	FALSE	TRUE	The changed values have been written to the technology data block. The description of the tags will show when the change becomes effective.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 10177) for parameter "Error"	
ErrorInfo	OUTPUT	WORD	16#0000	Error info ID (Page 10177) for parameter "ErrorID"	

Note

At the input parameters "RampUpTime", "RampDownTime", "EmergencyRampTime" and "JerkTime", values can be entered which exceed the admissible limits of the resulting parameters: "Acceleration", "Deceleration", "Emergency stop deceleration" and "Jerk".

Please note the equations and limits in "Axis technology object" -> "Configuring the technology object" -> "Dynamics" and ensure that the values you input are within the valid range.

See also

Overview of the Motion Control statements (Page 10096)

S7-1200 Motion Control V4 (Page 4823)

ErrorIDs and ErrorInfos (Page 10177)

Tag of the axis technology object V1...3 (Page 10190)

MC_Power: Enable, disable axis V1...3 (Page 4856)

MC_Reset: Acknowledge fault V1...3 (Page 4860)

MC_Home: Home axes, set reference point V1...3 (Page 4862)

MC_Halt: Halt axes V1...3 (Page 4865)

MC_MoveAbsolute: Absolute positioning of axes V1...3 (Page 4868)
MC_MoveRelative: Relative positioning of axes V1...3 (Page 4871)
MC_MoveVelocity: Move axes at preset rotational speed V1...3 (Page 4874)
MC_MoveJog: Move axes in jog mode V1...3 (Page 4878)
MC_CommandTable: Run axis commands as motion sequence V2...3 (Page 4881)

S7-1500 Motion Control

MC_Power

MC_Power: Enable, disable technology objects

Description

The Motion Control instruction "MC_Power" is used to enable and disable technology objects.

Applies to

- Positioning axis
- Speed axis
- External encoder

Requirements

- The technology object has been configured correctly.

Override response

- A MC_Power job cannot be aborted by any other Motion Control job.
- A MC_Power job with the parameter Enable = TRUE enables a technology object, but does not thereby abort any other Motion Control instructions.
- Disabling the technology object (parameter "Enable" = FALSE) aborts all Motion Control jobs at the corresponding technology object in accordance with the selected "StopMode". This process cannot be aborted by the user.

Parameters

The following table shows the parameters of the "MC_POWER" Motion Control instruction:

Parameter	Declaration	Data type	Default value	Description	
Axis	InOut	TO_Axis	-	Technology object	
Enable	INPUT	BOOL	FALSE	TRUE	The technology object is enabled.

Parameter	Declaration	Data type	Default value	Description	
				FALSE	The technology object is disabled. All current jobs at the technology object are aborted in accordance with the configured "StopMode".
StopMode	INPUT	INT	0	Not applicable to the external encoder technology object. If you disable a technology object with a falling edge at the Enable parameter, then the axis decelerates in accordance with the selected "StopMode".	
				0	Emergency stop When the technology object is disabled, the axis brakes to a standstill without any jerk limitation, using the emergency stop deceleration configured in Technology object > Configuration > Extended parameters > Emergency stop ramp. Enabling is then removed. (<TO>.DynamicDefaults. EmergencyDeceleration)
				1	Immediate stop When a technology object is disabled, the setpoint zero is output, and the enabling is removed. The axis is braked depending on the configuration in the drive, and is brought to a standstill.
				2	Stop with maximum dynamic values When the technology object is disabled, the axis brakes to a standstill using the maximum deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic limits". The configured maximum jerk is taken into account at the same time. Enabling is then removed. (<TO>.DynamicLimits.MaxDeceleration; <TO>.DynamicLimits.MaxJerk)
Status	OUTPUT	BOOL	FALSE	Technology object enable status	
				FALSE	Disabled - A positioning axis or speed axis does not accept any Motion Control jobs. - Speed control and position control are not active. - The actual values of the technology object are not checked for validity.
				TRUE	Enabled - An enabled positioning axis or speed axis accepts Motion Control jobs. - Speed control and positioning control are active. - The actual values of the technology object are valid.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being executed.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred in Motion Control instruction MC_Power. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 10405) for parameter "Error"	

Enabling technology objects

To enable a technology object, set the Enable parameter to TRUE.

If the Status parameter shows the value TRUE , then the technology object is enabled.

If an axis is in motion when the technology object is enabled (actual velocity is present), then the axis brakes to the setpoint zero, using the maximum deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic limits" (<TO>.DynamicLimits.MaxDeceleration). This braking ramp can be overridden by Motion Control jobs.

Note

Automatic enabling after acknowledgement of a technology alarm

If the technology object is disabled due to a technology alarm, then it will automatically be enabled again after the cause has been eliminated and the alarm has been acknowledged. This requires the Enable parameter to have retained the value TRUE during this process.

Disabling technology objects

To disable a technology object, set the Enable parameter to FALSE.

If an axis is in motion, then it will be braked to a standstill according to the selected "StopMode".

If the "Busy" and "Status" parameters show the value FALSE , then the disabling of the technology object has been completed.

Drive connection by means of PROFIdrive

When coupling a drive with PROFIdrive, the setpoint, enabling, and drive status are transmitted via the PROFIdrive frame.

- **Enable technology object and drive**
  The parameter ""Enable" = TRUE" is used to enable the technology object. The drive is enabled according to the PROFIdrive standard.
  If the tag <TO>.StatusDrive.InOperation shows the value TRUE , then the drive is ready to execute setpoints. The "Status" parameter is set to the value TRUE .
- **Disable technology object and drive**
  With the parameter "Enable" = FALSE, the "Status" parameter is set to the value FALSE, and the axis is braked according to the selected "StopMode". The drive is disabled according to the PROFIdrive standard.

Analog drive connection

The setpoint is output via an analog output. Optionally, you can configure an enabling signal via digital output (<TO>.Actor.Interface.EnableDriveOutput), and a readiness signal via digital input (<TO>.Actor.Interface.DriveReadyInput).

- **Enable technology object and drive**
  With the parameter "Enable" = TRUE, the enabling output ("Enable drive output") is set. If the drive reports back the readiness signal via the ready input ("Drive ready input"), then the "Status" parameter and the technology object's <TO>.StatusDrive.InOperation tag are set to TRUE, and the setpoint is switched at the analog output.
- **Disable technology object and drive**
  With the parameter "Enable" = FALSE, the "Status" parameter is set to the value FALSE, and the axis is braked according to the selected "StopMode". When the setpoint zero is reached, the enabling output is set to FALSE.

Additional information

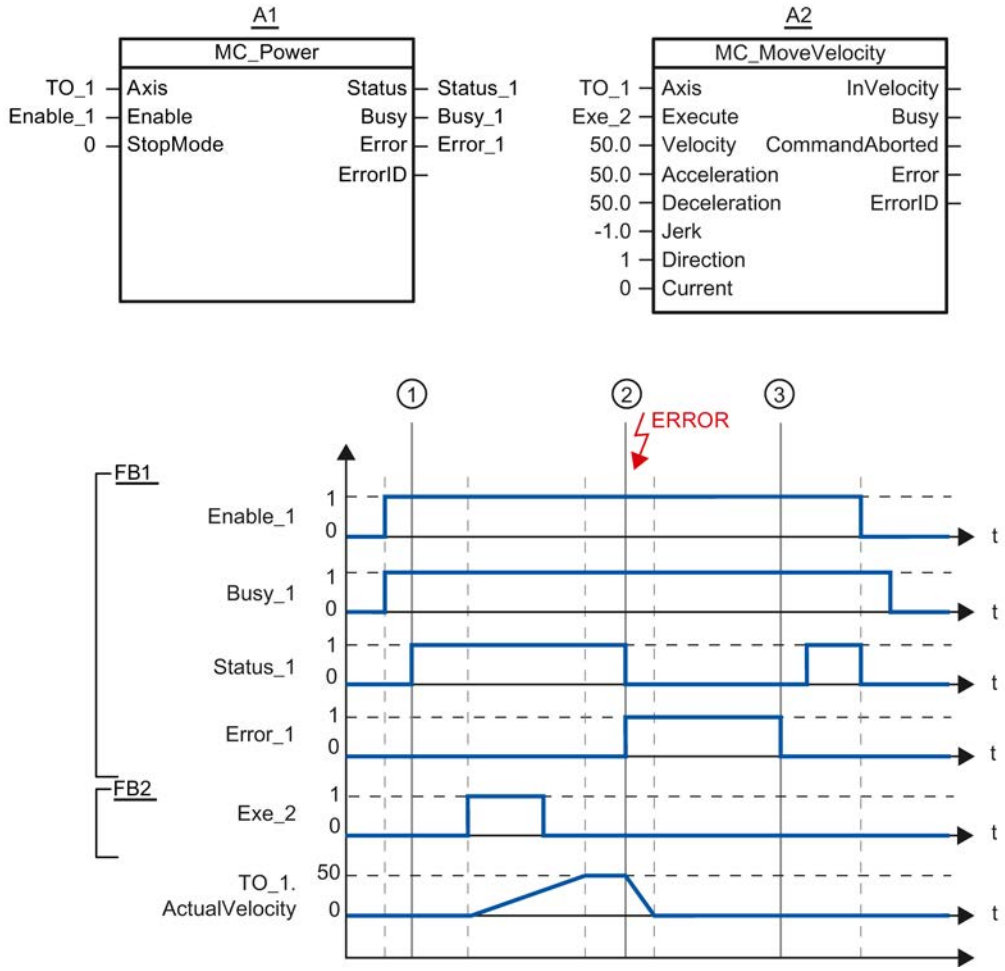
Additional information on enabling and disabling technology objects and drives can be found in the appendix, MC_Power Function Charts (Page 10408).

See also

Error ID (Page 10405)

MC_Power: Function chart

Function chart: Enabling a technology object, and example for alarm response



A technology object is enabled with "Enable_1= TRUE". Successful enabling can be read from "Status_1" at time ①. The axis will then move with an "MC_MoveVelocity" job (A2). The velocity profile of the axis can be read from "Velocity Axis_1".

At time ② an error occurs in the technology object, which results in the disabling of the technology object (alarm response: remove enable). The axis is braked depending on the configuration in the drive, and is brought to a standstill. When the technology object is disabled, Status_1 is reset. Since the axis was not disabled using "Enable_1" = FALSE, the selected "StopMode" does not apply. The cause of the error is corrected and the alarm is acknowledged at time ③.

Since "Enable_1" is still set, the technology object is enabled again. Successful enabling can be read from "Status_1". Next the technology object is disabled with "Enable_1" = FALSE.

MC_Home

MC_Home: Homing technology objects, setting home position

Description

With the Motion Control instruction "MC_Home", you create the relationship between the position in the technology object and the mechanical position. The position value in the technology object is assigned to a homing mark at the same time. This homing mark represents a known mechanical position.

The homing process occurs according to the mode selected with the "Mode" parameter, and the configuration under "Technology object > Configuration > Extended parameters > Homing".

The preset values under "Technology object > Configuration > Extended parameters > Dynamic default values" are used for the dynamic values Acceleration, Deceleration and Jerk.

Applies to

- Positioning axis
- External encoder

The following table shows which modes are possible with each of the technology objects:

Operating mode	Positioning axis with incremental encoder	Positioning axis with absolute encoder	External incremental encoder	External absolute encoder
Active homing (Mode = 4, 5)	X	-	-	-
Passive homing (Mode = 2, 3, 8)	X	-	X	-
Setting a positioning value ("Mode" = 0)	X	X	X	X
Relative shift of the actual value ("Mode" = 1)	X	X	X	X
Homing absolute encoder ("Mode" = 6, 7)	-	X	-	X

Requirements

- The technology object has been configured correctly.
- "Mode" = 2, 3, 4, 5, 8
  The technology object must be enabled.
- "Mode" = 0, 1, 2, 6, 7
  The encoder values must be valid. (<TO>.StatusSensor[n].State = 2)

Override response

- An MC_Home job for passive homing is aborted by:
  - Disabling the technology object by means of "MC_Power.Enable" = FALSE
  - "MC_Home" job with parameter "Mode" = 4, 5, 9
- An "MC_Home" job for passive homing does not abort any other Motion Control jobs.
- An "MC_Home" job for active homing is aborted by:
  - Disabling the technology object with "MC_Power.Enable" = FALSE
  - "MC_Home" job "Mode" = 4, 5
  - "MC_Halt" job
  - "MC_MoveAbsolute" job
  - "MC_MoveRelative" job
  - "MC_MoveVelocity" job
  - "MC_MoveJog" job
- An "MC_Home" job for active homing aborts the following running Motion Control jobs.
  - "MC_Home" job with parameter "Mode" = 2, 3, 4, 5
  - "MC_Halt" job
  - "MC_MoveAbsolute" job
  - "MC_MoveRelative" job
  - "MC_MoveVelocity" job
  - "MC_MoveJog" job

Parameters

The following table shows the parameters of the "MC_Home" Motion Control instruction:

Parameter	Declaration	Data type	Default value	Description	
Axis	InOut	TO_Axis	-	Technology object	
Execute	INPUT	BOOL	FALSE	Start of the job with a positive edge	
Position	INPUT	LREAL	0.0	The specified value is used according to the selected "Mode".	
Mode	INPUT	INT	4	Operating mode	
				0	Direct homing (absolute) The current position of the technology object is set to the value of parameter "Position".
				1	Direct homing (relative) The current position of the technology object is shifted by the value of parameter "Position".

Parameter	Declaration	Data type	Default value	Description	
				2	Passive homing When the homing mark is detected, the actual value is set to the value of the "Position" parameter.
				3	Passive homing ("Position" parameter has no effect) When the homing mark is detected, the actual value is set to the home position configured under "Technology object > Configuration > Extended parameters > Homing > Passive homing". (<TO>.Homing.HomePosition)
				4	Active homing The TO positioning axis performs a homing movement according to the configuration. After the completion of the movement, the axis is positioned at the value of the "Position" parameter.
				5	Active homing ("Position" parameter has no effect) The TO positioning axis performs a homing movement according to the configuration. After completion of the movement, the axis is positioned at the home position configured under "Technology object > Configuration > Extended parameters > Homing > Active homing". (<TO>.Homing.HomePosition)
				6	Absolute encoder adjustment (relative) The current position is shifted by the value of the parameter "Position". The calculated absolute value offset is stored retentively in the CPU. (<TO>.StatusSensor[n].AbsEncoderOffset)
				7	Absolute encoder adjustment (absolute) The current position is set to the value of the parameter "Position". The calculated absolute value offset is stored retentively in the CPU. (<TO>.StatusSensor[n].AbsEncoderOffset)
				8	Passive homing (without reset) Functions like mode 2, with the difference that the "homed" status is **not** reset when the function is enabled.
				9	Canceling passive homing A running job for passive homing is aborted.

Parameter	Declaration	Data type	Default value	Description	
Done	OUTPUT	BOOL	FALSE	TRUE	Job is completed
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the job was aborted by another job.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 10405) for parameter "Error"	

Resetting the "Homed" status

The "Homed" status of a technology object is reset under the following conditions (<TO>.StatusWord.HomingDone):

- **Technology objects with incremental actual values:**
  - Starting an "MC_Home" job with "Mode" = 2, 3, 4, 5
    (After successful completion of the homing operation, the "Homed" status is reset.)
  - Error in the encoder system, or encoder failure
  - Restart of the technology object
  - After POWER OFF -> POWER ON of the CPU
  - Memory reset
  - Modification of the encoder configuration
- **Technology objects with absolute actual values:**
  - Restoration of the CPU factory settings
  - Modification of the encoder configuration
  - Replacement of the CPU

Homing a technology object with "Mode" = 1..8

To home a technology object, follow these steps:

1. Check the requirements indicated above.
2. Specify the desired homing function in the "Mode" parameter.
3. Initialize the necessary parameters with values, and start the homing operation with a positive edge at the "Execute" parameter.

If the "Done" parameter shows the value TRUE, then the "MC_Home" job has been completed according to the selected "Mode". The "Homed" status of the technology object is indicated under "Technology object > Diagnostics > Status and error bits > Motion status > Homed" (<TO>.StatusWord.HomingDone).

Termination of a passive homing process with "Mode" = 9

With "Mode" = 9, the technology object is not homed. If a running "MC_Home" job for passive homing ("Mode" = 2, 3, 8) is overridden by another "MC_Home" job with "Mode" = 9, then the running job is terminated with the parameter "CommandAborted" = TRUE. The overriding job with "Mode" = 9 signals successful execution with the parameter "Done" = TRUE.

Additional information

An option for the evaluation of the individual status bits can be found in the Evaluating StatusWord, ErrorWord and WarningWord (Page 10317) section.

See also

Error ID (Page 10405)

MC_MoveJog

MC_MoveJog: Moving axes in jog mode

Description

With the Motion Control instruction "MC_MoveJog", you can move an axis in jog mode.

Dynamic behavior during movement is defined with the parameters "Velocity", "Jerk", "Acceleration" and "Deceleration".

- Positioning axis:
  A velocity is specified in the "Velocity" parameter.
- Speed axis:
  A speed is specified at the "Velocity" parameter.

Applies to

- Positioning axis
- Speed axis

Requirements

- The technology object has been configured correctly.
- The technology object is enabled.

Override response

The "MC_MoveJog" job is aborted by:

- Disabling the axis with "MC_Power.Enable" = FALSE
- "MC_Home" job "Mode" = 4, 5
- "MC_Halt" job
- "MC_MoveAbsolute" job
- "MC_MoveRelative" job
- "MC_MoveVelocity" job
- "MC_MoveJog" job

Starting a "MC_MoveJog" job aborts the following active Motion Control jobs:

- "MC_Home" job "Mode" = 4, 5
- "MC_Halt" job
- "MC_MoveAbsolute" job
- "MC_MoveRelative" job
- "MC_MoveVelocity" job
- "MC_MoveJog" job

Parameters

The following table shows the parameters of the "MC_MoveJog" Motion Control instruction:

Parameter	Declaration	Data type	Default value	Description
Axis	InOut	TO_SpeedAxis	-	Technology object
JogForward	INPUT	BOOL	FALSE	As long as the parameter is TRUE, the axis moves in the positive direction at the velocity specified in parameter "Velocity".
JogBackward	INPUT	BOOL	FALSE	As long as the parameter is TRUE, the axis moves in the negative direction at the velocity specified in parameter "Velocity".
Velocity	INPUT	LREAL	100.0	Setpoint speed / setpoint speed for the motion process Value > 0.0: The specified value is used. Value < 0.0: The specified value is used. ("Velocity" = 0.0 is permitted)
Acceleration	INPUT	LREAL	-1.0	Acceleration Value > 0.0: The specified value is used. Value = 0.0: not permitted Value < 0.0: The acceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Acceleration)

Parameter	Declaration	Data type	Default value	Description	
Deceleration	INPUT	LREAL	-1.0	Deceleration Value > 0.0: The specified value is used. Value = 0.0: not permitted Value < 0.0: The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)	
Jerk	INPUT	LREAL	-1.0	Jerk Value > 0.0: Constant-acceleration velocity profile; the specified jerk is used. Value = 0.0: Trapezoid velocity profile Value < 0.0: The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)	
InVelocity	OUTPUT	BOOL	FALSE	TRUE	The setpoint velocity / setpoint speed was reached and will be maintained.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the job was aborted by another job.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 10405) for parameter "Error"	

Behavior with setpoint velocity / setpoint speed zero (Velocity" = 0.0)

An "MC_MoveJog" job with "Velocity" = 0.0 stops the axis with the configured deceleration. When the setpoint velocity / setpoint speed zero is reached, the parameter "InVelocity" will indicate the value TRUE.

Under "Technology object > Diagnostics > Status and error bits > Motion status", "constant velocity" and "standstill" will be displayed (<TO>.StatusWord.ConstantVelocity; <TO>.StatusWord.Standstill).

Moving an axis in jog mode

Proceed as follows to move an axis in jog mode:

1. Check the requirements indicated above.
2. Move the axis in the positive direction with "JogForward", or in the negative direction with "JogBackward".

The current motion state is indicated in "Busy", "InVelocity" and "Error".

Note

Behavior when changing the override

If the velocity / speed is influenced during constant motion by a change in the override (<TO>.Override.Velocity), then the "InVelocity" parameter is reset during the acceleration or deceleration. When the newly calculated velocity is reached ("Velocity" × "Override" %), then "InVelocity" is set again.

Additional information

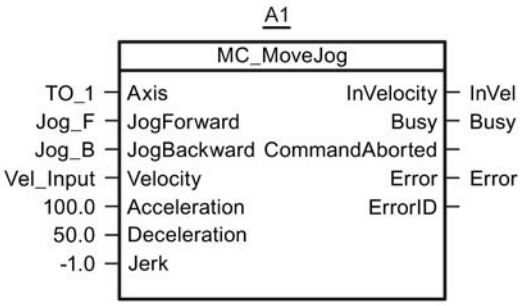
An option for the evaluation of the individual status bits can be found in the Evaluating StatusWord, ErrorWord and WarningWord (Page 10317) section.

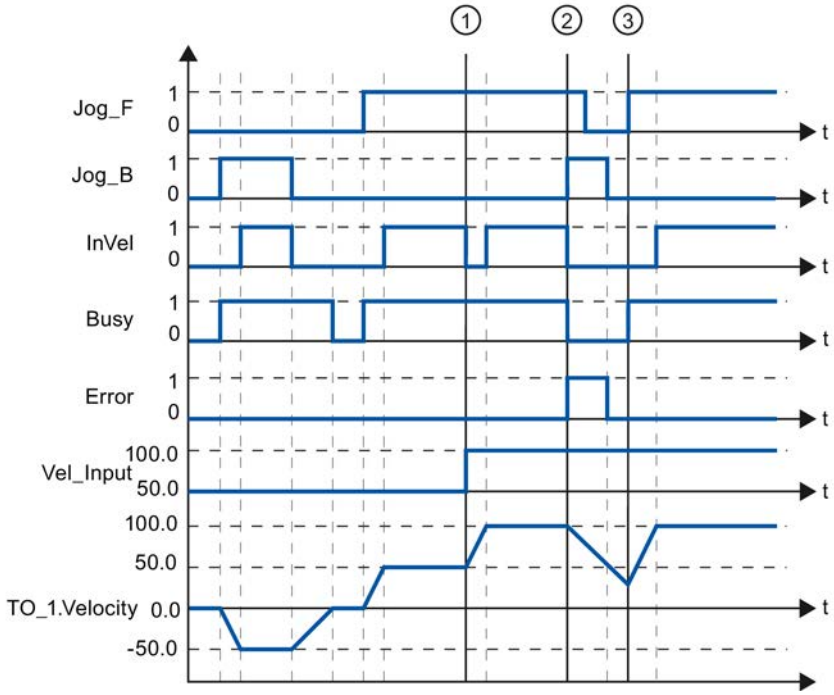
See also

Error ID (Page 10405)

MC_MoveJog: Function chart

Function chart: Moving an axis in jog mode





The axis is moved in the negative direction in jog mode via "Jog_B". When the setpoint velocity -50.0 is reached, this is signaled via "InVel" = TRUE. After "Jog_B" is reset, the axis is braked and brought to a standstill. Then the axis is moved in the positive direction via "Jog_F". When the setpoint velocity 50.0 is reached, this is signaled via "InVel" = TRUE.

At time ①, if "Jog_F" is set, the setpoint velocity is changed to 100.0 by means of "Vel_Input". Alternatively, you can also change the setpoint velocity via the velocity override. "InVel" is reset. The axis is being accelerated. When the new setpoint velocity 100.0 is reached, this is signaled via "InVel" = TRUE.

If "Jog_F" is set, "Jog_B" is likewise set at time ②. If both "Jog_F" and "Jog_B" are set, then the axis is braked with the last applicable deceleration. An error is indicated via "Error", and the "ErrorID" of the error 16#8007 (incorrect direction specification) is output.

This error is resolved by resetting the two inputs "Jog_F" and "Jog_B".

During the braking ramp, "Jog_F" is set at time ③. The axis is accelerated to the last configured velocity. When the setpoint velocity 100.0 is reached, this is signaled via "InVel" = TRUE.

MC_MoveVelocity

MC_MoveVelocity: Moving axes with the specified speed

Description

With the Motion Control instruction "MC_MoveVelocity", you can move an axis with a constant velocity.

Dynamic behavior during movement is defined with the parameters "Velocity", "Jerk", "Acceleration" and "Deceleration".

- Positioning axis:
  A velocity is specified in the "Velocity" parameter.
- Speed axis:
  A speed is specified at the "Velocity" parameter.

Applies to

- Positioning axis
- Speed axis

Requirements

- The technology object has been configured correctly.
- The technology object is enabled.

Override response

The "MC_MoveVelocity" is aborted by:

- Disabling the axis with "MC_Power.Enable" = FALSE
- "MC_Home" job "Mode" = 4, 5
- "MC_Halt" job
- "MC_MoveAbsolute" job
- "MC_MoveRelative" job
- "MC_MoveVelocity" job
- "MC_MoveJog" job

Starting a "MC_MoveVelocity" job aborts the following active Motion Control jobs:

- "MC_Home" job "Mode" = 4, 5
- "MC_Halt" job
- "MC_MoveAbsolute" job
- "MC_MoveRelative" job
- "MC_MoveVelocity" job
- "MC_MoveJog" job

Parameters

The following table shows the parameters of the "MC_MoveVelocity" Motion Control instruction:

Parameter	Declaration	Data type	Default value	Description
Axis	InOut	TO_SpeedAxis	-	Technology object
Execute	INPUT	BOOL	FALSE	Start of the job with a positive edge
Velocity	INPUT	LREAL	100.0	Setpoint speed / setpoint speed for the motion process ("Velocity" = 0.0 is permitted)
Acceleration	INPUT	LREAL	-1.0	Acceleration Value > 0.0: The specified value is used. Value = 0.0: not permitted Value < 0.0: The acceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Acceleration)
Deceleration	INPUT	LREAL	-1.0	Deceleration Value > 0.0: The specified value is used. Value = 0.0: not permitted Value < 0.0: The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)
Jerk	INPUT	LREAL	-1.0	Jerk Value > 0.0: Constant-acceleration velocity profile; the specified jerk is used. Value = 0.0: Trapezoid velocity profile Value < 0.0: The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)
Direction	INPUT	INT	0	Direction of rotation of the axis 0: The sign of the velocity specified at the "Velocity" parameter defines the direction of rotation. 1: Positive direction of rotation Value of "Velocity" is used.

Parameter	Declaration	Data type	Default value	Description	
				2	Negative direction of rotation Value of "Velocity" is used.
Current	INPUT	BOOL	FALSE	Maintain current velocity	
				FALSE	Disabled The values of parameters "Velocity" and "Direction" are taken into account.
				TRUE	Enabled The values at the parameters "Velocity" and "Direction" are **not** taken into account. The current velocity and direction at function start are retained. When the axis resumes motion at the velocity that was current at function start, the "InVelocity" parameter returns the value TRUE.
InVelocity	OUTPUT	BOOL	FALSE	TRUE	The setpoint velocity / setpoint speed was reached and will be maintained.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the job was aborted by another job.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the job. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 10405) for parameter "Error"	

Behavior with setpoint velocity / setpoint speed zero (Velocity" = 0.0)

An "MC_MoveVelocity" job with "Velocity" = 0.0 stops the axis with the configured deceleration. When the setpoint velocity / setpoint speed zero is reached, the parameter "InVelocity" will indicate the value TRUE.

Under "Technology object > Diagnostics > Status and error bits > Motion status", "constant velocity" and "standstill" will be displayed (<TO>.StatusWord.ConstantVelocity; <TO>.StatusWord.Standstill).

The parameters "InVelocity" and "Busy" show the value TRUE, until the "MC_MoveVelocity" job is overridden by another Motion Control job.

Moving an axis with constant velocity / speed

Proceed as follows to move an axis with constant velocity / speed:

1. Check the requirements indicated above.
2. At the parameter "Velocity", specify the velocity / speed, with which the axis should be moved.
3. Start the "MC_MoveVelocity" job with a positive edge at the parameter "Execute".

The current motion state is indicated in "Busy", "InVelocity" and "Error".

If the "InVelocity" parameter shows the value TRUE, then the setpoint velocity / setpoint speed was reached. The axis continues moving at this constant velocity. The parameters "InVelocity" and "Busy" show the value TRUE, until the "MC_MoveVelocity" job is overridden by another Motion Control job.

Note

Behavior when changing the override

If the velocity / speed is influenced during constant motion by a change in the override (<TO>.Override.Velocity), then the "InVelocity" parameter is reset during the acceleration or deceleration. When the newly calculated velocity / speed is reached ("Velocity" × "Override" %), then "InVelocity" is reset.

Additional information

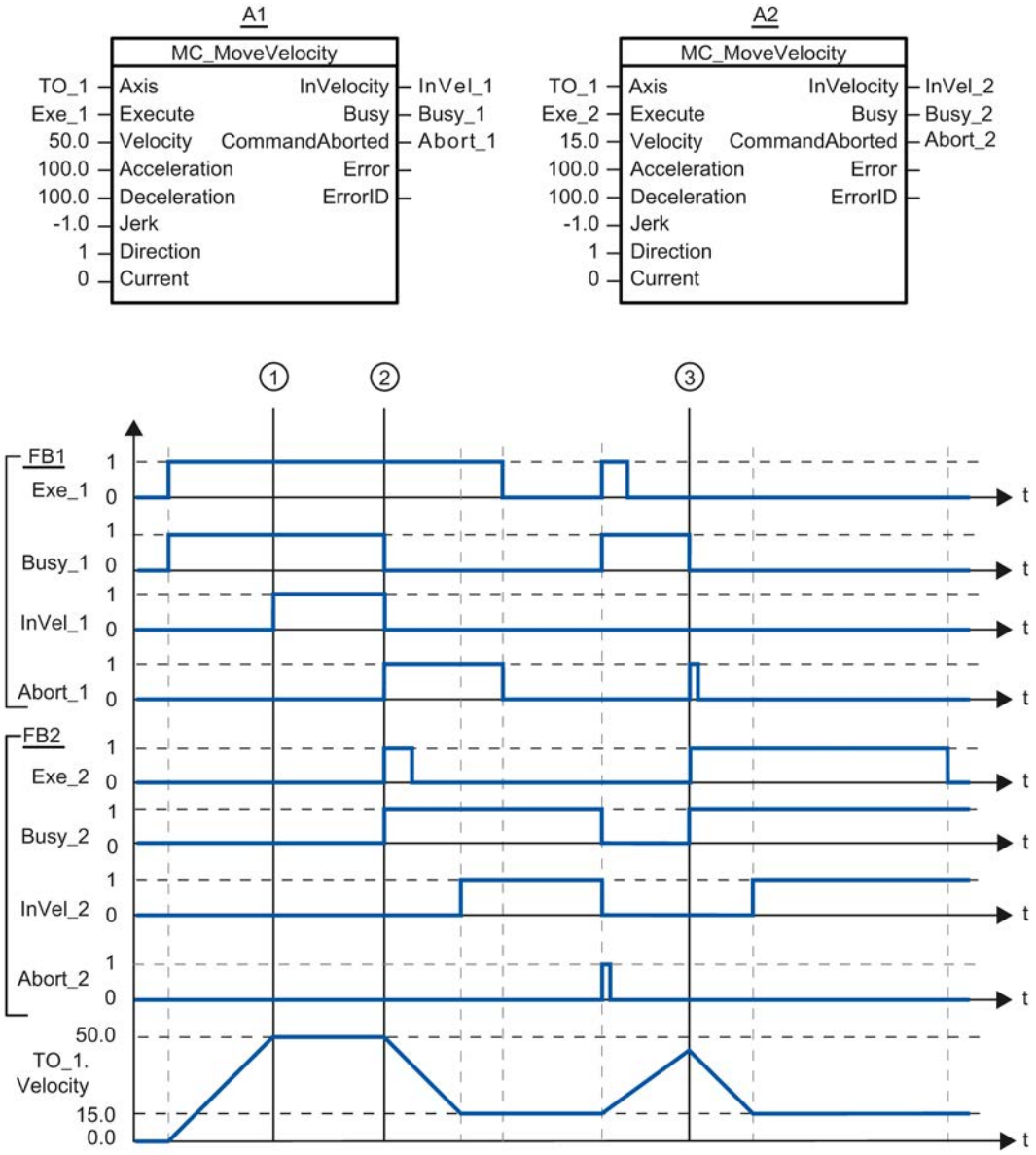
An option for the evaluation of the individual status bits can be found in the Evaluating StatusWord, ErrorWord and WarningWord (Page 10317) section.

See also

Error ID (Page 10405)

MC_MoveVelocity: Function chart

Function chart: Moving an axis with specification of velocity, and the response to an overriding job



A "MC_MoveVelocity" job (A1) initiated via "Exe_1" accelerates the axis and signals at time ① via "InVel_1" that the setpoint velocity 50.0 has been reached.

At time ②, the task is overridden via another "MC_MoveVelocity" job (A2). The abort is signaled via "Abort_1". When the new setpoint velocity 15.0 is reached, this is signaled via "InVel_2". The axis then continues moving at the constant velocity 15.0.

The running "MC_MoveVelocity" job (A2) is overridden by another "MC_MoveVelocity" job (A1). The abort is signaled via "Abort_2". The axis is accelerated to the new setpoint velocity 50.0. Before the setpoint velocity is reached, the current "MC_MoveVelocity" job A1) is

overridden at time ③ by another "MC_MoveVelocity" job A2). The abort is signaled via "Abort_1". When the new setpoint velocity 15.0 is reached, this is signaled via "InVel_2". The axis then continues moving at the constant velocity 15.0.

MC_MoveRelative

MC_MoveRelative: Relative positioning of axes

Description

With the Motion Control instruction "MC_MoveRelative", you can move an axis relative to the position that exists at the beginning of job processing.

Dynamic behavior during movement is defined with the parameters "Velocity", "Jerk", "Acceleration" and "Deceleration".

Applies to

- Positioning axis

Requirements

- The technology object has been configured correctly.
- The technology object is enabled.

Override response

The "MC_MoveRelative" job is aborted by:

- Disabling the axis with "MC_Power.Enable" = FALSE
- "MC_Home" job "Mode" = 4, 5
- "MC_Halt" job
- "MC_MoveAbsolute" job
- "MC_MoveRelative" job
- "MC_MoveVelocity" job
- "MC_MoveJog" job

Starting a "MC_MoveRelative" job aborts the following active Motion Control jobs:

- "MC_Home" job "Mode" = 4, 5
- "MC_Halt" job
- "MC_MoveAbsolute" job
- "MC_MoveRelative" job

- "MC_MoveVelocity" job
- "MC_MoveJog" job

Parameters

The following table shows the parameters of the "MC_MoveRelative" Motion Control instruction:

Parameter	Declaration	Data type	Default value	Description	
Axis	InOut	TO_Positioning Axis	-	Technology object	
Execute	INPUT	BOOL	FALSE	Start of the job with a positive edge	
Distance	INPUT	LREAL	0.0	Distance for the positioning process (negative or positive)	
Velocity	INPUT	LREAL	-1.0	Setpoint velocity for the positioning Value > 0.0: The specified value is used. Value = 0.0: not permitted Value < 0.0: The velocity configured in "Technology object > Configuration > Advanced parameters > Default dynamics" is used. (<TO>.DynamicDefaults.Velocity)	
Acceleration	INPUT	LREAL	-1.0	Acceleration Value > 0.0: The specified value is used. Value = 0.0: not permitted Value < 0.0: The acceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Acceleration)	
Deceleration	INPUT	LREAL	-1.0	Deceleration Value > 0.0: The specified value is used. Value = 0.0: not permitted Value < 0.0: The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)	
Jerk	INPUT	LREAL	-1.0	Jerk Value > 0.0: Constant-acceleration velocity profile; the specified jerk is used Value = 0.0: Trapezoid velocity profile Value < 0.0: The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)	
Done	OUTPUT	BOOL	FALSE	TRUE	Target position reached
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the job was aborted by another job.

Parameter	Declaration	Data type	Default value	Description	
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 10405) for parameter "Error"	

Moving an axis relative to the starting position

Proceed as follows to move an axis relative to the starting position:

1. Check the requirements indicated above.
2. Specify distance to be moved in the "Distance" parameter.
3. Start the "MC_MoveRelative" job with a positive edge at the parameter "Execute".

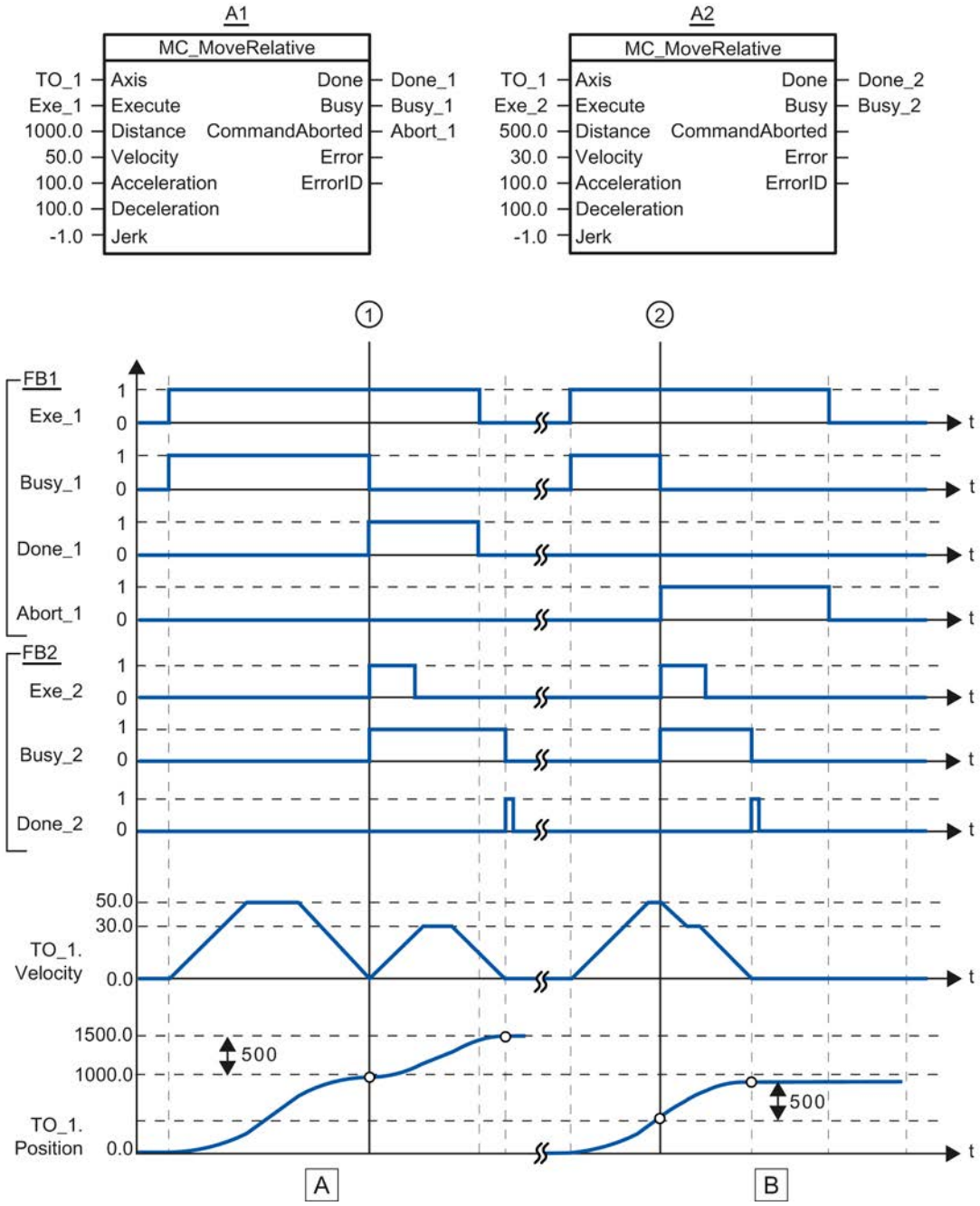
The current motion state is indicated in "Busy", "Done" and "Error".

See also

Error ID (Page 10405)

MC_MoveRelative: Function chart

Function chart: Relative positioning of an axis, and the response to an overriding job



Section A	The axis is moved by an "MC_MoveRelative" job (A1) by the distance ("Distance") 1000.0 (the starting position here is 0.0). When the axis reaches the target position, this is signaled at time ① via "Done_1". At this time ①, another "MC_MoveRelative" job (A2) with distance 500.0 is started. When the axis reaches the new target position, this is signaled via "Done_2". Since "Exe_2" was previously reset, "Done_2" is applied only to one cycle.
Section B	A running "MC_MoveRelative" job (A1) is overridden by another "MC_MoveRelative" job (A2). The abort is signaled at time ② via "Abort_1". The axis is then moved at the new velocity by the distance ("Distance") 500.0. When the new target position is reached, this is signaled via "Done_2".

MC_MoveAbsolute

MC_MoveAbsolute: Absolute positioning of axes

Description

With the Motion Control instruction "MC_MoveAbsolute", you can move an axis to an absolute position.

Dynamic behavior during movement is defined with the parameters "Velocity", "Jerk", "Acceleration" and "Deceleration".

Applies to

- Positioning axis

Requirements

- The technology object has been configured correctly.
- The technology object is enabled.
- The technology object is homed.

Override response

The "MC_MoveAbsolute" job is aborted by:

- Disabling the axis with "MC_Power.Enable" = FALSE
- "MC_Home" job "Mode" = 4, 5
- "MC_Halt" job
- "MC_MoveAbsolute" job
- "MC_MoveRelative" job
- "MC_MoveVelocity" job
- "MC_MoveJog" job

Starting a "MC_MoveAbsolute" job aborts the following active Motion Control jobs:

- "MC_Home" job "Mode" = 4, 5
- "MC_Halt" job

- "MC_MoveAbsolute" job
- "MC_MoveRelative" job
- "MC_MoveVelocity" job
- "MC_MoveJog" job

Parameters

The following table shows the parameters of the "MC_MoveAbsolute" Motion Control instruction:

Parameter	Declaration	Data type	Default value	Description
Axis	InOut	TO_Positioning Axis	-	Technology object
Execute	INPUT	BOOL	FALSE	Start of the job with a positive edge
Position	INPUT	REAL	0.0	Absolute target position
Velocity	INPUT	LREAL	-1.0	Setpoint velocity for the positioning Value > 0.0: The specified value is used. Value = 0.0: not permitted Value < 0.0: The velocity configured in "Technology object > Configuration > Advanced parameters > Default dynamics" is used. (<TO>.DynamicDefaults.Velocity)
Acceleration	INPUT	LREAL	-1.0	Acceleration Value > 0.0: The specified value is used. Value = 0.0: not permitted Value < 0.0: The acceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Acceleration)
Deceleration	INPUT	LREAL	-1.0	Deceleration Value > 0.0: The specified value is used. Value = 0.0: not permitted Value < 0.0: The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)
Jerk	INPUT	LREAL	-1.0	Jerk Value > 0.0: Constant-acceleration velocity profile; the specified jerk is used Value = 0.0: Trapezoid velocity profile Value < 0.0: The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)

Parameter	Declaration	Data type	Default value	Description	
Direction	INPUT	INT	3	Direction of rotation of the axis Is only evaluated with "modulo" enabled. "Technology object > Configuration > Basic parameters > Enable modulo	
				1	Positive direction of rotation
				2	Negative direction of rotation
				3	Shortest path
Done	OUTPUT	BOOL	FALSE	TRUE	Target position reached.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the job was aborted by another job.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 10405) for parameter "Error"	

Moving an axis to an absolute position

Proceed as follows to move an axis to an absolute position:

1. Check the requirements indicated above.
2. Specify the desired target position at the "Position" parameter.
3. Start the "MC_MoveAbsolute" job with a positive edge at the parameter "Execute".

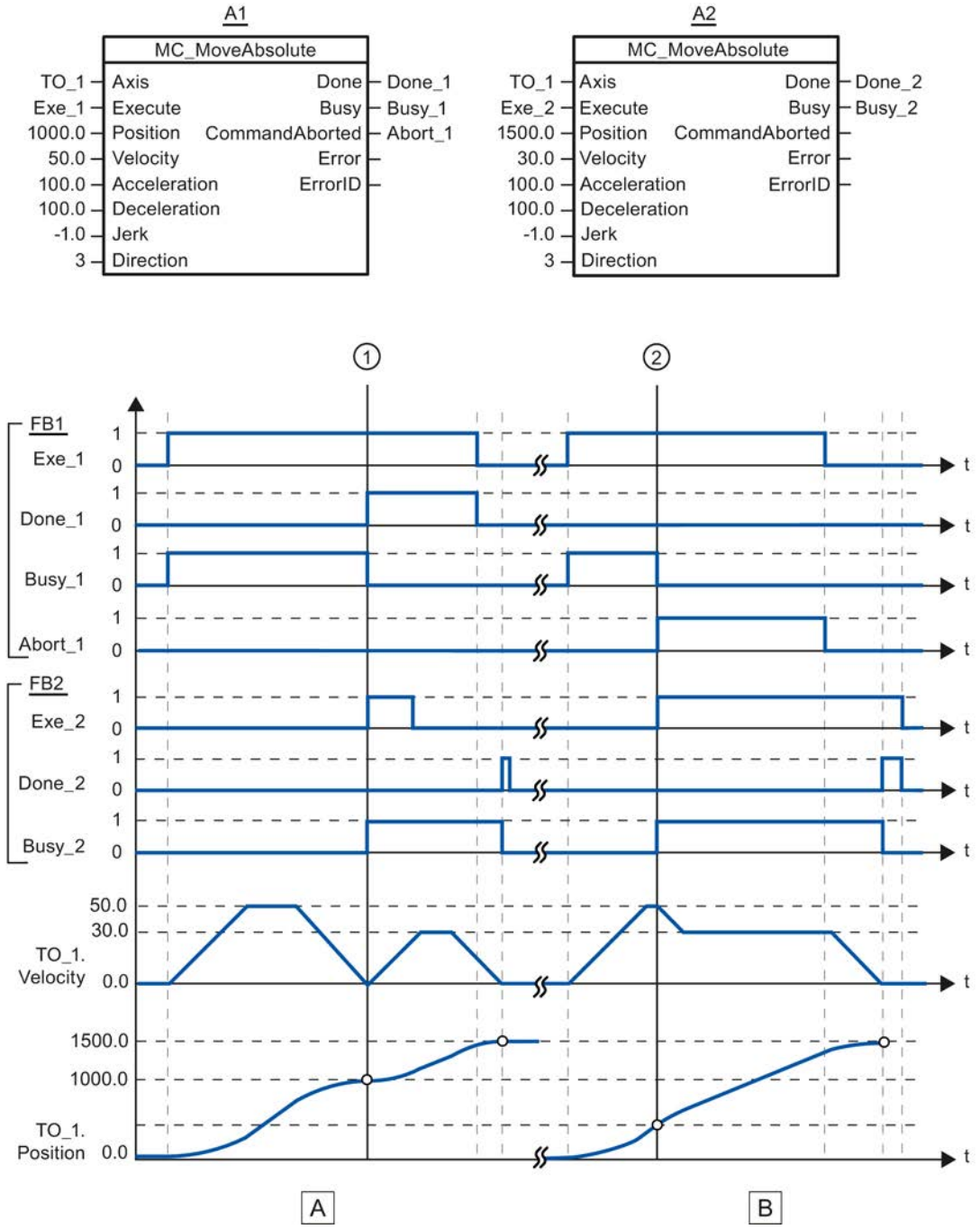
The current motion state is indicated in "Busy", "Done" and "Error".

See also

Error ID (Page 10405)

MC_MoveAbsolute: Function chart

Function chart: Absolute positioning of an axis, and the response to an overriding job



Section A	An axis is moved to absolute position 1000.0 with an "MC_MoveAbsolute" job (A1). When the axis reaches the target position, this is signaled at time ① via "Done_1". At this time ①, another "MC_MoveAbsolute" job (A2) with target position 1500.0 is started. When the axis reaches the target position 1500.0, this is signaled via "Done_2". Since "Exe_2" was previously reset, "Done_2" is applied only to one cycle.
Section B	A running "MC_MoveAbsolute" job (A1) is overridden at time ② by another "MC_MoveAbsolute" job (A2). The abort is signaled via "Abort_1". The axis is braked to the changed velocity and moved to the new target position 1500.0. When the new target position is reached, this is signaled via "Done_2".

MC_Halt

MC_Halt: Halt axis

Description

With the Motion Control instruction "MC_Halt", you can brake an axis to a standstill.

Dynamic behavior during the braking process is defined with the parameters "Jerk" and "Deceleration".

Applies to

- Positioning axis
- Speed axis

Requirements

- The technology object has been configured correctly.
- The technology object is enabled.

Override response

The "MC_Halt" job is aborted by:

- Disabling the axis with "MC_Power.Enable" = FALSE
- "MC_Home" job "Mode" = 4, 5
- "MC_Halt" job
- "MC_MoveAbsolute" job
- "MC_MoveRelative" job
- "MC_MoveVelocity" job
- "MC_MoveJog" job

Starting a "MC_Halt" job aborts the following active Motion Control jobs:

- "MC_Home" job "Mode" = 4, 5
- "MC_Halt" job

- "MC_MoveAbsolute" job
- "MC_MoveRelative" job
- "MC_MoveVelocity" job
- "MC_MoveJog" job

Parameters

The following table shows the parameters of the "MC_Halt" Motion Control instruction:

Parameter	Declaration	Data type	Default value	Description	
Axis	InOut	TO_SpeedAxis	-	Technology object	
Execute	INPUT	BOOL	FALSE	Start of the job with a positive edge	
Deceleration	INPUT	LREAL	-1.0	Deceleration Value > 0.0: The specified value is used. Value = 0.0: not permitted Value < 0.0: The deceleration configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Deceleration)	
Jerk	INPUT	LREAL	-1.0	Jerk Value > 0.0: Constant-acceleration velocity profile; the specified jerk is used Value = 0.0: Trapezoid velocity profile Value < 0.0: The jerk configured in "Technology object > Configuration > Extended parameters > Dynamic defaults" is used. (<TO>.DynamicDefaults.Jerk)	
Done	OUTPUT	BOOL	FALSE	TRUE	Zero velocity reached
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the job was aborted by another job.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 10405) for parameter "Error"	

Braking an axis with "MC_Halt"

Proceed as follows to decelerate an axis to a standstill:

1. Check the requirements indicated above.
2. Supply the parameters "Deceleration" and "Jerk" with the desired values.
3. Start the "MC_Halt" job with a positive edge at the parameter "Execute".

The current motion state is indicated in "Busy", "Done" and "Error". The standstill of the axis is indicated under "Technology object > Diagnostics > Status and error bits > Motion status > Standstill" (<TO>.StatusWord.Standstill).

Additional information

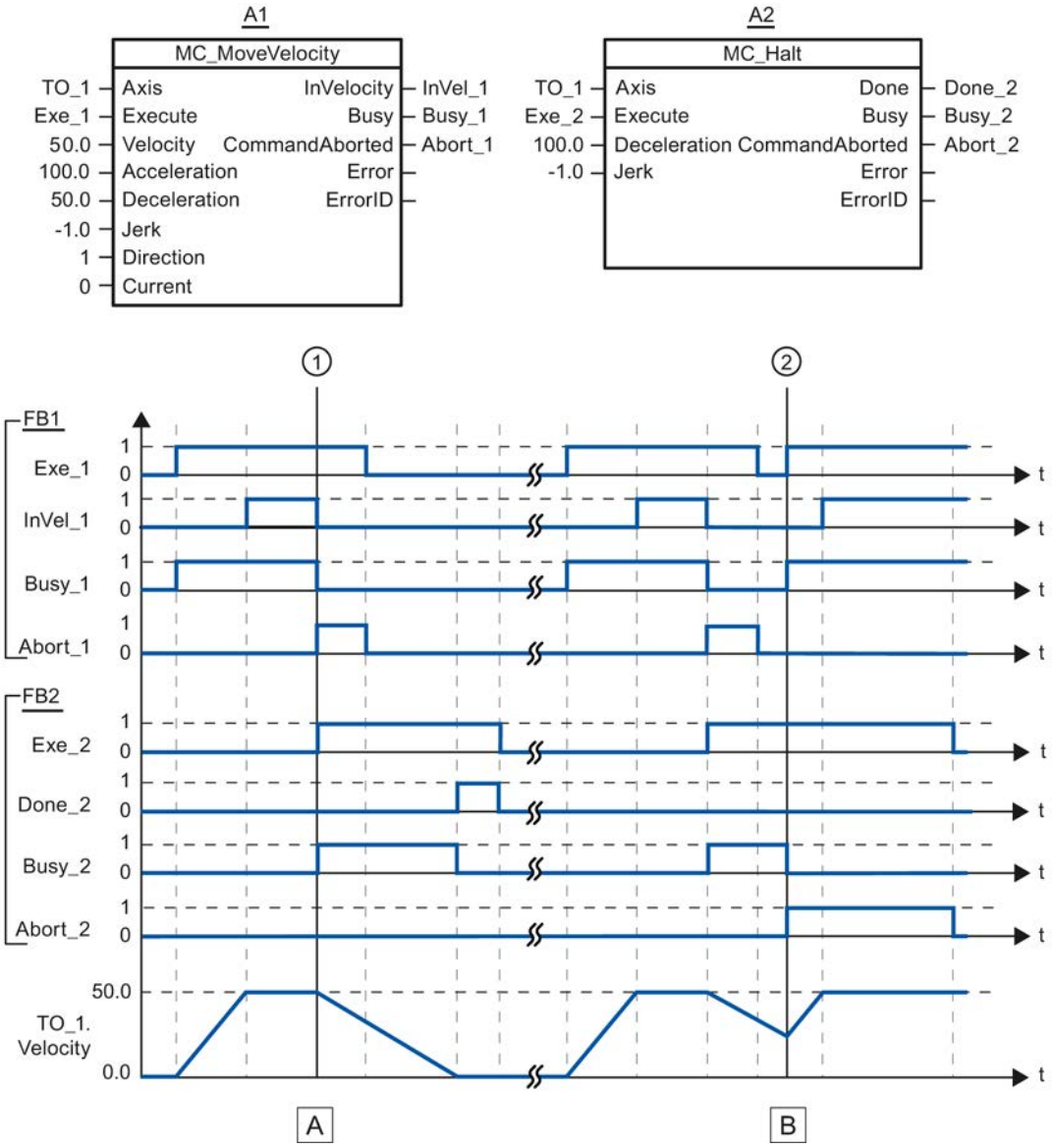
An option for the evaluation of the individual status bits can be found in the Evaluating StatusWord, ErrorWord and WarningWord (Page 10317) section.

See also

Error ID (Page 10405)

MC_Halt: Function chart

Function chart: Stopping an axis, and the response to an overriding job



Section A	An axis is moved with an "MC_MoveVelocity" job (A1). When the setpoint velocity 50.0 is reached, this is signaled via "InVel_1". At time ①, the "MC_MoveVelocity" job is overridden by an "MC_Halt" job (A2). The job termination is signaled via "Abort_1". The axis is braked to a standstill. Successful completion of the "MC_Halt" job is signaled via "Done_2".
Section B	The axis is moved with an "MC_MoveVelocity" job (A1). When the setpoint velocity 50.0 is reached, this is signaled via "InVel_1". Next the "MC_MoveVelocity" job is overridden by an "MC_Halt" job (A2). The job termination is signaled via "Abort_1". During the braking process the "MC_Halt" job is overridden at time ② by an "MC_MoveVelocity" job (A1). The job termination is signaled via "Abort_2". When the setpoint velocity 50.0 is reached, this is signaled via "InVel_1". The axis is then moved with constant velocity.

MC_Reset

MC_Reset: Acknowledging alarms, restarting technology objects

Description

All technology alarms that can be acknowledged in the user program are acknowledged with the Motion Control instruction "MC_Reset". Acknowledgment also resets the "Error" and "Warning" bits in the technology data block.

Technology objects are reinitialized (restarted) using the Motion Control instruction "MC_Reset" with "Restart" = TRUE. Upon restart of the technology object, the new configuration data are applied in the technology data block.

Applies to

- Positioning axis
- Speed axis
- External encoder

Requirements

- The cause of the error for a pending technology alarm has been resolved.
- For a restart, the technology object must be disabled.
  ("MC_Power.Status" = FALSE and "MC_Power.Busy" = FALSE)

Override response

- Parameter "Restart" = FALSE:

Processing of the instruction "MC_Reset" can be aborted by other Motion Control jobs. The MC_Reset job does not abort any running Motion Control jobs.

- Parameter "Restart" = TRUE:

The processing of the instruction "MC_Reset" with parameter "Restart" = TRUE cannot be aborted by any other Motion Control job.

Parameters

The following table shows the parameters of the "MC_Reset" Motion Control instruction:

Parameter	Declaration	Data type	Default value	Description
Axis	InOut	TO_Axis	-	Technology object
Execute	INPUT	BOOL	FALSE	Start of the job with a positive edge

Parameter	Declaration	Data type	Default value	Description	
Restart	INPUT	BOOL	FALSE	TRUE	"Restart" Reinitialization of the technology object and acknowledgment of pending technology alarms. The technology object is reinitialized with the configured initial values.
				FALSE	Acknowledgement of queued technology alarms.
Done	OUTPUT	BOOL	FALSE	TRUE	Error has been acknowledged. Restart has been executed.
Busy	OUTPUT	BOOL	FALSE	TRUE	The job is being executed.
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	During execution the job was aborted by another job.
Error	OUTPUT	BOOL	FALSE	TRUE	An error occurred during execution of the job. The job is rejected. The cause of the error can be found in the "ErrorID" parameter.
ErrorID	OUTPUT	WORD	16#0000	Error ID (Page 10405) for parameter "Error"	

Acknowledging technology alarms

To acknowledge technology alarms, follow these steps:

1. Check the requirements indicated above.
2. Set the parameter "Restart" = FALSE.
3. Start the acknowledgement of the error with a positive edge at parameter "Execute".

If the Done parameter shows the value TRUE, then the error was acknowledged.

Note

Acknowledging with "Restart" = FALSE

If only the technology alarms are to be acknowledged, set "Restart" = FALSE, The technology object cannot be used during a restart.

Restarting a technology object

Proceed as follows to restart a technology object:

1. Check the requirements indicated above.
2. Set the parameter "Restart" = TRUE.
3. Perform the restart with a positive edge at the input parameter "Execute".

If the "Done" parameter shows the value TRUE, then the restart of the technology object has been completed.

Additional information on a restart can be found in the Restarting Technology Objects (Page 10336) section.

See also

Error ID (Page 10405)

High-speed counters

CTRL_HSC: Control high-speed counters

Parameter

Parameter	Declaration	Data type	Memory area	Description
EN	INPUT	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	OUTPUT	BOOL	I, Q, M, D, L	Enable output
HSC	INPUT	HW_HSC	I, Q, M or constant	Hardware address of the high-speed counter (HW-ID)
DIR	INPUT	BOOL	I, Q, M, D, L or constant	Enables the new count direction (see NEW_DIR)
CV	INPUT	BOOL	I, Q, M, D, L or constant	Enables the new count value (see NEW_CV)
RV	INPUT	BOOL	I, Q, M, D, L or constant	Enables the new reference value (see NEW_RV)
PERIOD	INPUT	BOOL	I, Q, M, D, L or constant	Enables the new period of a frequency measurement (see NEW_PERIOD)
NEW_DIR	INPUT	INT	I, Q, M, D, L or constant	Count direction loaded when DIR = TRUE.
NEW_CV	INPUT	DINT	I, Q, M, D, L or constant	Count value loaded when CV = TRUE.
NEW_RV	INPUT	DINT	I, Q, M, D, L or constant	Reference value loaded when RV = TRUE.
NEW_PERIOD	INPUT	INT	I, Q, M, D, L or constant	Period of the frequency measurement loaded when PERIOD = TRUE.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	Processing status
STATUS	OUTPUT	WORD	I, Q, M, D, L	Status of the operation

Description

With the "Control high-speed counter" instruction, you can make parameter settings and control the high-speed counters supported by the CPU by loading new values into the counter. Execution of the instruction requires that the high-speed counter to be controlled is enabled. You cannot execute multiple "Control high-speed counter" instructions simultaneously in the program for a given high-speed counter.

You can load the following parameter values into a high-speed counter using the "Control high-speed counter" instruction:

- Count direction (NEW_DIR): The count direction defines whether a high-speed counter counts up or down. The count direction is defined by the following values at the NEW_DIR input: 1 = up, -1 = down.
  A change to the count direction with the "Control high-speed counter" instruction is only possible when direction control is set in the parameters by the program. The count direction specified at the NEW_DIR input is loaded into a high-speed counter when the bit at the DIR input is set.
- Count value (NEW_CV): The count value is the initial value at which a high-speed counter starts counting. The count value can be in the range -2147483648 to 2147483647.
  The count value specified at the NEW_CV input is loaded into a high-speed counter when the bit at the CV input is set.
- Reference value (NEW_RV): You can compare the reference value with the current counter value to trigger an alarm. Similar to the counter value, the reference value can be in the range -2147483648 to 2147483647.
  The reference value specified at the NEW_RV input is loaded into a high-speed counter when the bit at the RV input is set.
- Period of the frequency measurement (NEW_PERIOD): The period of the frequency measurement is specified by the following values at the NEW_PERIOD input: 10 = 0.01s, 100 = 0.1s, 1000 = 1s.
  The time period can be updated if the "Measure frequency" function for the specified high-speed counter is configured. The time period specified at the NEW_PERIOD input is loaded into a high-speed counter when the bit at the PERIOD input is set.

The "Control high-speed counter" instruction is only executed if the signal state at the EN input is "1". As long as the operation is executing, the bit at the BUSY output is set. Once the operation has executed completely, the bit at the BUSY output is reset.

The ENO enable output is set only when the EN enable input has signal state "1" and no errors occur during execution of the operation.

When inserting the "Control high-speed counter" instruction, an instance data block is created in which the operation data is saved.

Parameter STATUS

At the STATUS output, you can query whether errors occurred during execution of the "Control high-speed counter" instruction. The following table shows the meaning of the values output at the STATUS output:

Error code (hexadecimal)	Description
0	No error
80A1	Hardware identifier of the high-speed counter invalid
80B1	Count direction (NEW_DIR) invalid
80B2	Count value (NEW_CV) invalid
80B3	Reference value (NEW_RV) invalid
80B4	Period of the frequency measurement (NEW_PERIOD) invalid
80C0	Multiple access to the high-speed counter
80D0	The high-speed counter (HSC) is not enabled in the CPU hardware configuration.

Counting and measurement (S7-1500)

High_Speed_Counter

Description High_Speed_Counter

Description

The High_Speed_Counter instruction is used to control the technology module counting and measuring functions via the user program.

Call

The instruction High_Speed_Counter must be called once per counter, either cyclically or in a time-controlled program. The call is not permitted in an event-controlled interrupt program.

Functional description

Counter value: The counter value is available at the output parameter CountValue. The counter value is updated at every call of the High_Speed_Counter instruction.

Measured value: The measured value is available at the output parameter MeasuredValue. The measured value is updated at every call of the High_Speed_Counter instruction.

The measured value and the counter value are available concurrently in the feedback interface.

Capture: The output parameter CaptureStatus = TRUE indicates a valid Capture value at the output parameter CapturedValue.

- A Capture value is captured under the following conditions:
  - A digital input has the parameter assignment "Capture"
  - CaptureEnable = TRUE
  - Edge at digital input with the Capture function
- The output parameter CaptureStatus is reset by a falling edge at input parameter CaptureEnable.

Synchronization: The output parameter SyncStatus = TRUE indicates that a synchronization has occurred.

- The counter value is synchronized under the following conditions:
  - A digital input has the parameter assignment "Synchronization" or the incremental encoder has the parameter assignment "Synchronization at signal N"
  - SyncEnable = TRUE
  - SyncUpDirection (or SyncDownDirection) = TRUE
  - Edge at digital input with the synchronization function or rising edge of the signal N at the encoder input
- The output parameter SyncStatus is reset by a falling edge at
  - the input parameter SyncEnable or
  - the static tag SyncDownDirection or
  - the static tag SyncUpDirection

Parameter changes via the user program

Proceed as follows to modify parameters using the user program:

1. Check the relevant Set tag to establish whether the technology object is ready for the parameter change (Set tag = FALSE) or whether a change command is still running (Set tag = TRUE).
2. If the technology object is ready for the parameter change, modify the relevant static tag.
3. Set the relevant Set tag for execution of the change command.
4. Use the output parameter Error to check whether an error has occurred.
   If no errors have occurred and the Set tag has been automatically reset by the technology object, the parameter change was successful.

Low or high counter limit: The two limits are changed as follows via the user program:

1. Checking the static tag SetUpperLimit (or SetLowerLimit) = FALSE
2. Changing the static tag NewUpperLimit (or NewLowerLimit)
3. SetUpperLimit (or SetLowerLimit) = TRUE
4. Checking the new counter limit in the static tags CurUpperLimit or CurLowerLimit

Note

If the new high counting limit is less than the current count value, the count value is set to the low counting limit or the start value, depending on the configuration. If the new low counting limit is greater than the current count value, the count value is set to the high counting limit or the start value, depending on the configuration.

Comparison values: The two comparison values are changed as follows via the user program:

1. Checking the static tag SetReferenceValue0 (or ReferenceValue1) = FALSE
2. Changing the static tag NewReferenceValue0 (or NewReferenceValue1)
3. SetReferenceValue0 (or SetReferenceValue1) = TRUE
4. Checking the new comparison value in the static tag CurReferenceValue0 (or CurReferenceValue1)

Counter value: You can set the current counter value to a new counter value from the user program. The current counter value is changed as follows:

1. Checking the input parameter SetCountValue = FALSE
2. Changing the static tag NewCountValue
3. SetCountValue = TRUE
4. Checking the new counter value in the output parameter CountValue

Start value: The start value is changed as follows using the user program:

1. Checking the static tag SetStartValue = FALSE
2. Changing the static tag NewStartValue
3. SetStartValue = TRUE
4. Checking the new start value in the static tag CurStartValue

Acknowledgment of events

Acknowledgment of signaled events can be performed via the rising edge of the input parameter EventAck . EventAck must stay set until the technology object has reset the status bits of the following events of the counting channel:

- CompResult0
- CompResult1
- ZeroStatus
- PosOverflow
- NegOverflow

Status of the digital inputs

You can obtain the status of the digital inputs with the static tags StatusDI0, StatusDI1 or StatusDI2.

Use digital outputs by user program

You can set the digital outputs with the High_Speed_Counter instruction,

- if the setting "Use by user program" is configured for "Set output".
- if the setting "After set command from CPU until comparison value" is configured for "Set output" .
- if you set the corresponding static tag ManualCtrlDQm (temporary overwrite).

The static tags SetDQ0 and SetDQ1 are only effective in these cases. In the first and third case, DQm follows the value of SetDQm. In the second case, DQm is set with an edge (rising or falling) by SetDQm. DQm is reset when the counter value equals the comparison value.

Reaction to error

If an error has occurred during the call of the instruction or in the technology module, the output parameter Error is set. More detailed error information can be read at the output parameter ErrorID (Page 4925).

Eliminate the cause of the error and acknowledge the error message by setting the input parameter ErrorAck. When no more errors are pending, the technology object resets the output parameter Error . No new error is signaled until you acknowledge the previous error.

Changing the count direction

The count direction can only be changed by the user program if "Pulse (A)" is configured as the signal type. In all other cases, the count direction is determined by the input signals of the technology module. The count direction is controlled by the static tag NewDirection:

- +1: Upward count direction
- -1: Downward count direction

To execute the change command, you need to set the tag SetNewDirection = TRUE.

High_Speed_Counter input parameters

Parameter	Declaration	Data type	Default	Description
SwGate	INPUT	BOOL	FALSE	Control software gate: • Rising edge: Software gate opens • Falling edge: Software gate closes Together with the hardware gate, the SwGate enables the internal gate.
SetCountValue	INOUT	BOOL	FALSE	A rising edge starts the transmission of the new counter value in the static tag NewCountValue to the technology module. The counter value is effective as soon as transmission is complete.

Parameter	Declaration	Data type	Default	Description
CaptureEnable	INPUT	BOOL	FALSE	Enabling the Capture function A Capture event occurs after enabling at the next configured edge at the relevant digital input. A falling edge at CaptureEnable resets the output parameter CaptureStatus. A falling edge at CaptureEnable resets the enable even if no Capture event has occurred.
SyncEnable	INPUT	BOOL	FALSE	Enable synchronization The direction enabled for synchronization is indicated in the static tags SyncUpDirection and SyncDownDirection. A falling edge at SyncEnable resets the output parameter SyncStatus .
ErrorAck	INPUT	BOOL	FALSE	A rising edge acknowledges the error status reported.
EventAck	INPUT	BOOL	FALSE	A rising edge resets the following output parameters: • CompResult0 • CompResult1 • ZeroStatus • PosOverflow • NegOverflow

High_Speed_Counter output parameters

Parameter	Declaration	Data type	Default	Description
StatusHW	OUTPUT	BOOL	FALSE	Status bit technology module: The module is configured and ready for operation. The module data is valid.
StatusGate	OUTPUT	BOOL	FALSE	Status bit: Internal gate is released if parameter is set
StatusUp	OUTPUT	BOOL	FALSE	Status bit: The last count pulse has incremented counter and took place no more than 0.5 s ago
StatusDown	OUTPUT	BOOL	FALSE	Status bit: The last count pulse has decremented the counter and took place no more than 0.5 s ago
CompResult0	OUTPUT	BOOL	FALSE	Status bit: Compare event for DQ0 has occurred The rising edge of the input parameter EventAck is used to reset CompResult0 .
CompResult1	OUTPUT	BOOL	FALSE	Status bit: Compare event for DQ1 has occurred The rising edge of the input parameter EventAck is used to reset CompResult1 .
SyncStatus	OUTPUT	BOOL	FALSE	Status bit: Synchronization has occurred If the input parameter SyncEnable is set, the configured edge sets the status bit SyncStatus at the respective digital input. SyncStatus is reset with falling edge at • SyncEnable (input parameter) or • SyncUpDirection (static tag) or • SyncDownDirection (static tag)

Parameter	Declaration	Data type	Default	Description
CaptureStatus	OUTPUT	BOOL	FALSE	Status bit: Capture event has occurred If the input parameter CaptureEnable is set, the configured edge sets the status bit CaptureStatus. at the respective digital input. Falling edge of input parameter CaptureEnable resets CaptureStatus .
ZeroStatus	OUTPUT	BOOL	FALSE	Status bit: CountValue has reached "0" value The rising edge of the input parameter EventAck is used to reset ZeroStatus.
PosOverflow	OUTPUT	BOOL	FALSE	Status bit: CountValue has violated high counting limit The rising edge of the input parameter EventAck is used to reset PosOverflow.
NegOverflow	OUTPUT	BOOL	FALSE	Status bit: CountValue has violated low counting limit The rising edge of the input parameter EventAck is used to reset NegOverflow.
Error	OUTPUT	BOOL	FALSE	An error has occurred. Refer to the output parameter ErrorID for the cause of the error.
ErrorID	OUTPUT	WORD	0	The ErrorID (Page 4925) parameter displays the number of the error message. ErrorID = 0000_H: No error pending.
CountValue	OUTPUT	DINT	0	Current counter value
CapturedValue	OUTPUT	DINT	0	Last captured Capture value; CaptureStatus = TRUE, if new Capture event has occurred
MeasuredValue	OUTPUT	REAL	0.0	Current measured value for frequency, period or velocity (depending on configuration)

ErrorID parameter

Error code (W#16#...)	Description
0000	No error
Error messages from technology module	
80A1	POWER_ERROR from feedback interface: Incorrect supply voltage L+
80A2	ENC_ERROR from feedback interface: Incorrect encoder signal
80A3	LD_ERROR from feedback interface: Error when loading via control interface
Error messages of the instruction High_Speed_Counter	
80B1	Invalid count direction
80B4	New low counting limit does not meet the following conditions: • Low counting limit < high counting limit • Low counting limit <= comparison value/counter value/start value
80B5	New high counting limit does not meet the following conditions: • Low counting limit < high counting limit • High counting limit >= comparison value/counter value/start value

Error code (W#16#...)	Description
80B6	New start value does not meet the following condition: • Low counting limit <= start value <= high counting limit
80B7	New counter value does not meet the following condition: • Low counting limit <= counter value <= high counting limit
80B8	New comparison value 0 does not meet the following conditions: • Low counting limit <= comparison value 0 <= high counting limit • Comparison value 0 < comparison value 1
80B9	New comparison value 1 does not meet the following conditions: • Low counting limit <= comparison value 1 <= high counting limit • Comparison value 0 < comparison value 1
80C0	Instruction High_Speed_Counter was called multiple times with the same instance (DB)
80C1	Communication with technology module failed (read data records): Error information of internal instruction RDREC saved in static tag AdditionalErrorID
80C2	Communication with technology module failed (write data records): Error information of internal instruction WRREC saved in static tag AdditionalErrorID
80C3	Access to input data (feedback interface) failed: Error information of internal instruction GETIO_PART saved in static tag AdditionalErrorID
80C4	Access to output data (control interface) failed: Error information of internal instruction SETIO_PART saved in static tag AdditionalErrorID
80C5	Reading of the current start information of the OB failed: Error information of internal instruction RD_SINFO saved in static tag AdditionalErrorID
80C6	Failed to get I/O addresses of the technology module: Error information of internal instruction RD_ADDR saved in static tag AdditionalErrorID

High_Speed_Counter static variables

Tag	Data type	Default	Access	Description
NewCountValue	DINT	L#0	Writing	New counter value
NewReferenceValue0	DINT	L#0	Writing	New comparison value 0
NewReferenceValue1	DINT	L#10	Writing	New comparison value 1
NewUpperLimit	DINT	L#2147483647	Writing	New high counting limit
NewLowerLimit	DINT	L#-2147483648	Writing	New low counting limit
NewStartValue	DINT	L#0	Writing	New start value
CurReferenceValue0	DINT	L#0	Read	Current comparison value 0
CurReferenceValue1	DINT	L#10	Read	Current comparison value 1
CurUpperLimit	DINT	L#2147483647	Read	Current high counting limit
CurLowerLimit	DINT	L#-2147483648	Read	Current low counting limit
CurStartValue	DINT	L#0	Read	Current start value
NewDirection	INT	0	Writing	New count direction: +1: Upward count direction -1: Downward count direction

Tag		Data type	Default	Access	Description
AdditionalErrorID		WORD	W#16#0000	Read	Error information of an internal instruction, e.g.,RDREC
UserCmdFlags		STRUCT	-		
	SetNewDirection	BOOL	FALSE	Writing	Setting a new count direction
	SetUpperLimit	BOOL	FALSE	Writing	Set high counting limit
	SetLowerLimit	BOOL	FALSE	Writing	Set low counting limit
	SetReferenceValue0	BOOL	FALSE	Writing	Set comparison value 0
	SetReferenceValue1	BOOL	FALSE	Writing	Set comparison value 1
	SetStartValue	BOOL	FALSE	Writing	Set start value
	SyncDownDirection	BOOL	TRUE	Writing	Enable synchronization in downward count direction
	SyncUpDirection	BOOL	TRUE	Writing	Enable synchronization in upward count direction
	SetDQ0	BOOL	FALSE	Writing	Set digital output DQ0
	SetDQ1	BOOL	FALSE	Writing	Set digital output DQ1
	ManualCtrlDQ0	BOOL	FALSE	Writing	Enable setting of digital output DQ0: TRUE: • SetDQ0 sets DQ0 • Control bit TM_CTRL_DQ0 = FALSE FALSE: • Setting not enabled • Control bit TM_CTRL_DQ0 = TRUE
	ManualCtrlDQ1	BOOL	FALSE	Writing	Enable setting of digital output DQ1: TRUE: • SetDQ1 sets DQ1 • Control bit TM_CTRL_DQ1 = FALSE FALSE: • Setting not enabled • Control bit TM_CTRL_DQ1 = TRUE
UserStatusFlags		STRUCT	-		
	StatusDI0	BOOL	FALSE	Read	Current status of digital input DI0
	StatusDI1	BOOL	FALSE	Read	Current status of digital input DI1
	StatusDI2	BOOL	FALSE	Read	Current status of digital input DI2
	StatusDQ0	BOOL	FALSE	Read	Current status of digital output DQ0
	StatusDQ1	BOOL	FALSE	Read	Current status of digital output DQ1

PID Control

PID_Compact

New features of PID_Compact

PID_Compact V2.2

- **Use with S7-1200**
  As of PID_Compact V2.2, the instruction with V2 functionality can also be used on S7-1200 with firmware version 4.0 or higher.

PID_Compact V2.0

- **Reaction to error**
  The reaction to error has been completely overhauled. PID_Compact now reacts in a more fault-tolerant manner in the default setting. This reaction is set when copying PID_Compact V1.X from an S7-1200 CPU to an S7-1500 CPU.

NOTICE
Your system may be damaged. If you use the default setting, PID_Compact remains in automatic mode when the process value limits are exceeded. This may damage your system. It is essential to configure how your controlled system reacts in the event of an error to protect your system from damage.

  The Error parameter indicates if an error is pending. When the error is no longer pending, Error = FALSE. The ErrorBits parameter shows which errors have occurred. Use ErrorAck to acknowledge the errors and warnings without restarting the controller or clearing the integral action. Switching operating modes no longer clears errors that are no longer pending.
  You can configure the reaction to error with SetSubstituteOutput and ActivateRecoverMode.
- **Substitute output value**
  You can configure a substitute output value that is to be output if an error occurs.
- **Switching the operating mode**
  You specify the operating mode at the Mode in/out parameter and use a rising edge at ModeActivate to start the operating mode. The sRet.i_Mode tag has been omitted.
- **Multi-instance capability**
  You can call up PID_Compact as multi-instance DB. No technology object is created in this case and no parameter assignment interface or commissioning interface is available. You must assign parameters for PID_Compact directly in the multi-instance DB and commission it via a watch table.
- **Startup characteristics**
  The operating mode specified at the Mode parameter is also started on a falling edge at Reset and during a CPU cold restart, if RunModeByStartup = TRUE.

- **ENO characteristics**
  ENO is set depending on the operating mode.
  If State = 0, then ENO = FALSE.
  If State ≠ 0, then ENO = TRUE.
- **Setpoint value specification during tuning**
  You configure the permitted fluctuation of the setpoint during tuning at the CancelTuningLevel tag.
- **Value range for output value limits**
  The value 0.0 no longer has to fall within the output value limits.
- **Pre-assigning the integral action**
  Using the tags IntegralResetMode and OverwriteInitialOutputValue, you can determine the pre-assignment of the integral action when switching from "Inactive" operating mode to "Automatic mode".
- **Switching a disturbance variable on**
  You can switch a disturbance variable on at the Disturbance parameter.
- **Default value of PID parameters**
  The following default settings have been changed:
  - Proportional action weighting (PWeighting) from 0.0 to 1.0
  - Derivative action weighting (DWeighting) from 0.0 to 1.0
  - Coefficient for derivative delay (TdFiltRatio) from 0.0 to 0.2
- **Renaming tags**
  The static tags have been given new names that are compatible with PID_3Step.

PID_Compact V1.2

- **Manual mode on CPU startup**
  If ManualEnable = TRUE when the CPU starts, PID_Compact starts in manual mode. A rising edge at ManualEnable is not necessary.
- **Pretuning**
  If the CPU is switched off during pretuning, pretuning starts again when the CPU is switched back on.

PID_Compact V1.1

- **Manual mode on CPU startup**
  When the CPU starts up, PID_Compact only switches to manual mode with a rising edge at ManualEnable. Without rising edge, PID_Compact starts in the last operating mode in which ManualEnable was FALSE.
- **Reaction to reset**
  A rising edge at Reset resets the errors and warnings and clears the integral action. A falling edge at Reset triggers a switchover to the most recently active operating mode.
- **Default of process value high limit**
  The default value of r_Pv_Hlm has been changed to 120.0.

- Monitoring the sampling time
  - An error is no longer output when the current sampling time is >= 1.5 x current mean value or when the current sampling time is <= 0.5 x current mean value. The sampling time may deviate much more in automatic mode.
  - PID_Compact is compatible with FW, V2.0 or higher.
- Access to tags
  The following tags can now be used in the user program.
  - i_Event_SUT
  - i_Event_TIR
  - r_Ctrl_Ioutv
- Troubleshooting
  PID_Compact now outputs the correct pulses when the shortest ON time is not equal to the shortest OFF time.

Compatibility with CPU and FW

The following table shows which version of PID_Compact can be used on which CPU.

CPU	FW	PID_Compact
S7-1200	V4.x	V2.2 V1.2
S7-1200	V3.X	V1.2 V1.1
S7-1200	V2.X	V1.2 V1.1
S7-1200	V1.X	V1.0
S7-1500	V1.1	V2.2 V2.1 V2.0
S7-1500	V1.0	V2.0

PID_Compact V2

Description of PID_Compact V2

Description

The PID_Compact instruction provides a PID controller with integrated tuning for actuators with proportional action.

The following operating modes are possible:

- Inactive
- Pretuning
- Fine tuning
- Automatic mode
- Manual mode
- Substitute output value with error monitoring

For a more detailed description of the operating modes, see the State parameter.

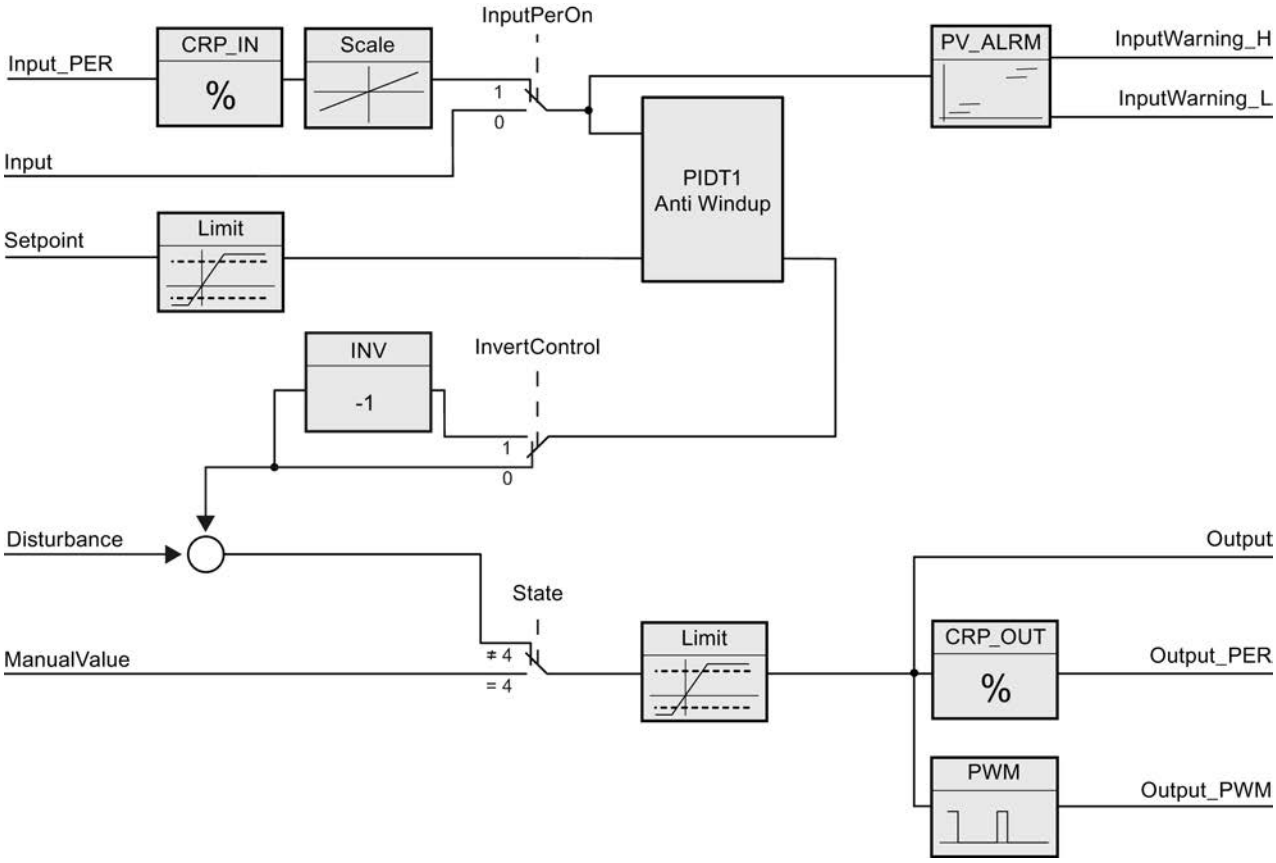
PID algorithm

PID_Compact is a PIDT1 controller with anti-windup and weighting of the proportional and derivative actions. The PID algorithm operates according to the following equation:

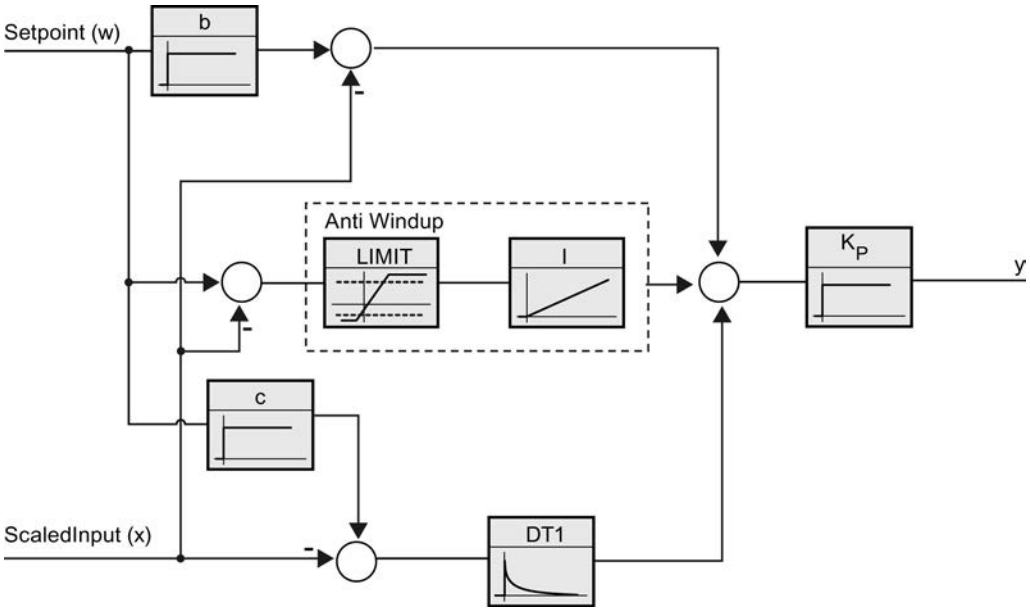
$$y = K_p \left[(b \cdot w - x) + \frac{1}{T_I \cdot s} (w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1} (c \cdot w - x) \right]$$

Symbol	Description
y	Output value of the PID algorithm
K_p	Proportional gain
s	Laplace operator
b	Proportional action weighting
w	Setpoint
x	Process value
T_I	Integral action time
T_D	Derivative action time
a	Derivative delay coefficient (derivative delay T1 = a × T_D)
c	Derivative action weighting

Block diagram of PID_Compact



Block diagram of PIDT1 with anti-windup



Call

PID_Compact is called in the constant time scale of a cycle interrupt OB.

If you call PID_Compact as a multi-instance DB, no technology object is created. No parameter assignment interface or commissioning interface is available. You must assign parameters for PID_Compact directly in the multi-instance DB and commission it via a watch table.

Download to device

The actual values of retentive variables are only updated when you download PID_Compact completely.

Downloading technology objects to device (Page 9536)

Startup

When the CPU starts up, PID_Compact starts in the operating mode that is saved in the Mode in/out parameter. To switch to "Inactive" operating mode during startup, set RunModeByStartup = FALSE.

Reaction to error

In automatic mode and during commissioning, the reaction to error depends on the SetSubstituteOutput and ActivateRecoverMode variables. In manual mode, the reaction is independent of SetSubstituteOutput and ActivateRecoverMode. If ActivateRecoverMode = TRUE, the reaction additionally depends on the error that occurred.

SetSubstituteOutput	ActivateRecoverMode	Configuration editor > output value > Set Output to	Reaction
Not relevant	FALSE	Zero (inactive)	Switch to "Inactive" mode (State = 0) The value 0.0 0 is transferred to the actuator.
FALSE	TRUE	Current output value while error is pending	Switch to "Substitute output value with error monitoring" mode (State = 5) The current output value is transferred to the actuator while the error is pending.
TRUE	TRUE	Substitute output value while error is pending	Switch to "Substitute output value with error monitoring" mode (State = 5) The value at SubstituteOutput is transferred to the actuator while the error is pending.

In manual mode, PID_Compact uses ManualValue as output value, unless ManualValue is invalid. If ManualValue is invalid, SubstituteOutput is used. If ManualValue and SubstituteOutput are invalid, Config.OutputLowerLimit is used.

The Error parameter indicates if an error is pending. When the error is no longer pending, Error = FALSE. The ErrorBits parameter shows which errors have occurred. ErrorBits is reset by a rising edge at Reset or ErrorAck.

PID_Compact V2 mode of operation

Monitoring process value limits

You specify the high limit and low limit of the process value in the Config.InputUpperLimit and Config.InputLowerLimit tags. If the process value is outside these limits, an error occurs (ErrorBits = 0001h).

You specify a high and low warning limit of the process value in the Config.InputUpperWarning and Config.InputLowerWarning tags. If the process value is outside these warning limits, a warning occurs (Warning = 0040h), and the InputWarning_H or InputWarning_L output parameter changes to TRUE.

Limiting the setpoint

You specify a high limit and low limit of the setpoint in the Config.SetpointUpperLimit and Config.SetpointLowerLimit tags. PID_Compact automatically limits the setpoint to the process value limits. You can limit the setpoint to a smaller range. PID_Compact checks whether this range falls within the process value limits. If the setpoint is outside these limits, the high or low limit is used as the setpoint, and output parameter SetpointLimit_H or SetpointLimit_L is set to TRUE.

The setpoint is limited in all operating modes.

Limiting the output value

You specify a high limit and low limit of the output value in the Config.OutputUpperLimit and Config.OutputLowerLimit tags. Output, ManualValue, and SubstituteOutput are limited to these values. The output value limits must match the control logic.

The valid output value limit values depend on the Output used.

Output	-100.0 to 100.0%
Output_PER	-100.0 to 100.0%
Output_PWM	0.0 to 100.0%

Rule:

OutputUpperLimit > OutputLowerLimit

Substitute output value

In the event of an error, PID_Compact can output a substitute output value that you define at the tag SubstituteOutput. The substitute output value must be within the output value limits.

Monitoring signal validity

The values of the following parameters are monitored for validity when used:

- Setpoint
- Input

- Input_PER
- Disturbance
- ManualValue
- SubstituteOutput
- Output
- Output_PER
- Output_PWM

Monitoring of the sampling time PID_Compact

Ideally, the sampling time is equivalent to the cycle time of the calling OB. The PID_Compact instruction measures the time interval between two calls. This is the current sampling time. On every switchover of operating mode and during the initial startup, the mean value is formed from the first 10 sampling times. Too great a difference between the current sampling time and this mean value triggers an error (Error = 0800h).

The error occurs during tuning if:

- New mean value >= 1.1 x old mean value
- New mean value <= 0.9 x old mean value

The error occurs in automatic mode if:

- New mean value >= 1.5 x old mean value
- New mean value <= 0.5 x old mean value

If you deactivate the sampling time monitoring (CycleTime.EnMonitoring = FALSE), you can also call PID_Compact in OB1. You must then accept a lower control quality due to the deviating sampling time.

Sampling time of the PID algorithm

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is calculated during tuning and rounded to a multiple of the cycle time. All other functions of PID_Compact are executed at every call.

If you use Output_PWM, the accuracy of the output signal is determined by the ratio of the PID algorithm sampling time to the cycle time of the OB. The cycle time should be at least 10 times the PID algorithm sampling time.

Control logic

An increase of the output value is generally intended to cause an increase in the process value. This is referred to as a normal control logic. For cooling and discharge control systems, it may be necessary to invert the control logic. PID_Compact does not work with negative proportional gain. If InvertControl = TRUE, an increasing control deviation causes a reduction in the output value. The control logic is also taken into account during pretuning and fine tuning.

Input parameters of PID_Compact V2

Table 9-81

Parameter	Data type	Default	Description
Setpoint	REAL	0.0	Setpoint of the PID controller in automatic mode
Input	REAL	0.0	A tag of the user program is used as the source of the process value. If you are using the Input parameter, then Config.InputPerOn = FALSE must be set.
Input_PER	INT	0	An analog input is used as the source of the process value. If you are using the Input_PER parameter, then Config.InputPerOn = TRUE must be set.
Disturbance	REAL	0.0	Disturbance variable or precontrol value
ManualEnable	BOOL	FALSE	• A FALSE -> TRUE edge activates "manual mode", while State = 4, Mode remain unchanged. As long as ManualEnable = TRUE, you cannot change the operating mode via a rising edge at ModeActivate or use the commissioning dialog. • A TRUE -> FALSE edge activates the operating mode that is specified by Mode. We recommend that you change the operating mode using ModeActivate only.
ManualValue	REAL	0.0	Manual value This value is used as the output value in manual mode. Values from Config.OutputLowerLimit to Config.OutputUpperLimit are permitted.
ErrorAck	BOOL	FALSE	• FALSE -> TRUE edge ErrorBits and Warning are reset.
Reset	BOOL	FALSE	Restarts the controller. • FALSE -> TRUE edge – Switch to "Inactive" mode – ErrorBits and Warnings are reset. – Integral action is cleared (PID parameters are retained) • As long as Reset = TRUE, PID_Compact remains in "Inactive" mode (State = 0). • TRUE -> FALSE edge PID_Compact switches to the operating mode that is saved in the Mode parameter.
ModeActivate	BOOL	FALSE	• FALSE -> TRUE edge PID_Compact switches to the operating mode that is saved in the Mode parameter.

Output parameters of PID_Compact V2

Table 9-82

Parameter	Data type	Default	Description
ScaledInput	REAL	0.0	Scaled process value
The "Output", "Output_PER", and "Output_PWM" outputs can be used concurrently.			
Output	REAL	0.0	Output value in REAL format
Output_PER	INT	0	Analog output value
Output_PWM	BOOL	FALSE	Pulse-width-modulated output value The output value is formed by by variable On and Off times.
SetpointLimit_H	BOOL	FALSE	If SetpointLimit_H = TRUE, the absolute setpoint high limit is reached (Setpoint ≥ Config.SetpointUpperLimit). The setpoint is limited to Config.SetpointUpperLimit .
SetpointLimit_L	BOOL	FALSE	If SetpointLimit_L = TRUE, the absolute setpoint low limit has been reached (Setpoint ≤ Config.SetpointLowerLimit). The setpoint is limited to Config.SetpointLowerLimit .
InputWarning_H	BOOL	FALSE	If InputWarning_H = TRUE, the process value has reached or exceeded the warning high limit.
InputWarning_L	BOOL	FALSE	If InputWarning_L = TRUE, the process value has reached or fallen below the warning low limit.
State	INT	0	The State parameter (Page 4947) shows the current operating mode of the PID controller. You can change the operating mode using the input parameter Mode and a rising edge at ModeActivate. • State = 0: Inactive • State = 1: Pretuning • State = 2: Fine tuning • State = 3: Automatic mode • State = 4: Manual mode • State = 5: Substitute output value with error monitoring
Error	BOOL	FALSE	If Error = TRUE, at least one error message is pending in this cycle.
ErrorBits	DWORD	DW#16#0	The ErrorBits parameter (Page 4951) shows which error messages are pending. ErrorBits is retentive and is reset upon a rising edge at Reset or ErrorAck.

In/out parameters of PID_Compact V2

Table 9-83

Parameter	Data type	Default	Description
Mode	INT	4	At Mode, specify the operating mode to which PID_Compact is to switch. Options are: • Mode = 0: Inactive • Mode = 1: Pretuning • Mode = 2: Fine tuning • Mode = 3: Automatic mode • Mode = 4: Manual mode The operating mode is activated by: • Rising edge at ModeActivate • Falling edge at Reset • Falling edge at ManualEnable • Cold restart of CPU if RunModeByStartup = TRUE Mode is retentive. A detailed description of the operating modes can be found in Parameters State and Mode V2 (Page 4947).

See also

Parameters State and Mode V2 (Page 4947)

Static tags of PID_Compact V2

You must not change variables that are not listed. These are used for internal purposes only.

Table 9-84

Tag	Data type	Default	Description
IntegralResetMode	INT	1	The tag IntegralResetMode determines how PIDCtrl.IntegralSum is pre-assigned when switching from "Inactive" operating mode to "Automatic mode". This setting only works for one cycle. Options are: • IntegralResetMode = 0: Smoothing The value of IntegralSum is pre-assigned so that the switchover is bumpless. • IntegralResetMode = 1: Deleting The value of IntegralSum is deleted. Any control deviation will cause a jump change of the output value. • IntegralResetMode = 2: Holding The value of IntegralSum is not changed. You can define a new value using the user program. • IntegralResetMode = 3: Pre-assigning The value of IntegralSum is automatically pre-assigned so that Output is calculated with reference to the value OverwriteInitialOutputValue. This setting is useful, for example, for an override controller.
OverwriteInitialOutputValue	REAL	0.0	If IntegralResetMode = 3, the value of IntegralSum is automatically pre-assigned so that Output = OverwriteInitialOutputValue in the next cycle.
RunModeByStartup	BOOL	TRUE	Activate operating mode at Mode parameter after CPU restart If RunModeByStartup = TRUE, PID_Compact starts in the operating mode saved in the Mode parameter after CPU startup. If RunModeByStartup = FALSE, PID_Compact remains in "Inactive" mode after CPU startup.
LoadBackUp	BOOL	FALSE	If LoadBackUp = TRUE, the last set of PID parameters is reloaded. The set was saved prior to the last tuning. LoadBackUp is automatically set back to FALSE.
PhysicalUnit	INT	0	Unit of measurement of the process value and setpoint, e.g., ºC, or ºF.
PhysicalQuantity	INT	0	Physical quantity of the process value and setpoint, e.g., temperature.
ActivateRecoverMode	BOOL	TRUE	The Tag ActivateRecoverMode V2 (Page 4953) determines the reaction to error.
Warning	DWORD	0	Tag Warning V2 (Page 4955) shows the warnings since Reset = TRUE or ErrorAck =TRUE. Warning is retentive.
Progress	REAL	0.0	Progress of tuning as a percentage (0.0 - 100.0)

Tag	Data type	Default	Description
CurrentSetpoint	REAL	0.0	CurrentSetpoint always displays the current setpoint. This value is frozen during tuning.
CancelTuningLevel	REAL	10.0	Permissible fluctuation of setpoint during tuning. Tuning is not canceled until: • Setpoint > CurrentSetpoint + CancelTuningLevel or • Setpoint < CurrentSetpoint - CancelTuningLevel
SubstituteOutput	REAL	0.0	Substitute output value When the following conditions are met, the substitute output value is used: • An error has occurred in automatic mode. • SetSubstituteOutput = TRUE • ActivateRecoverMode = TRUE
SetSubstituteOutput	BOOL	TRUE	If SetSubstituteOutput = TRUE and ActivateRecoverMode = TRUE, the substitute output value configured is output as long as an error is pending. If SetSubstituteOutput = FALSE and ActivateRecoverMode = TRUE, the actuator remains at the current output value as long as an error is pending. If ActivateRecoverMode = FALSE, SetSubstituteOutput is not effective. If SubstituteOutput is invalid (ErrorBits = 20000h), the substitute output value cannot be output.
Config.InputPerOn	BOOL	TRUE	If InputPerOn = TRUE, the Input_PER parameter is used. If InputPerOn = FALSE, the Input parameter is used.
Config.InvertControl	BOOL	FALSE	Invert control logic If InvertControl = TRUE, an increasing control deviation causes a reduction in the output value.
Config.InputUpperLimit	REAL	120.0	High limit of the process value Input and Input_PER are monitored to ensure adherence to this limit. At the I/O input, the process value can be a maximum of 18% higher than the standard range (overrange). This pre-assignment ensures that an error is no longer signaled due to a violation of the "Process value high limit". Only a wire-break and a short-circuit are recognized and PID_Compact reacts according to the configured reaction to error. InputUpperLimit > InputLowerLimit
Config.InputLowerLimit	REAL	0.0	Low limit of the process value Input and Input_PER are monitored to ensure adherence to this limit. InputLowerLimit < InputUpperLimit

Tag	Data type	Default	Description
Config.InputUpperWarning	REAL	3.402822e+38	Warning high limit of the process value If you set InputUpperWarning outside the process value limits, the configured absolute process value high limit is used as the warning high limit. If you configure InputUpperWarning within the process value limits, this value is used as the warning high limit. InputUpperWarning > InputLowerWarning InputUpperWarning ≤ InputUpperLimit
Config.InputLowerWarning	REAL	-3.402822e +38	Warning low limit of the process value If you set InputLowerWarning outside the process value limits, the configured absolute process value low limit is used as the warning low limit. If you configure InputLowerWarning within the process value limits, this value is used as the warning low limit. InputLowerWarning < InputUpperWarning InputLowerWarning ≥ InputLowerLimit
Config.OutputUpperLimit	REAL	100.0	High limit of output value For details, see OutputLowerLimit OutputUpperLimit > OutputLowerLimit
Config.OutputLowerLimit	REAL	0.0	Low limit of output value For Output and Output_PER, the range of values from -100.0 to +100.0, including zero, is valid. At -100.0, Output_PER = -27648; at +100.0, Output_PER = 27648. For Output_PWM, the value range 0.0 to +100.0 applies. The output value limits must match the control logic. OutputLowerLimit < OutputUpperLimit
Config.SetpointUpperLimit	REAL	3.402822e+38	High limit of setpoint If you configure SetpointUpperLimit outside the process value limits, the configured absolute process value high limit is used as the setpoint high limit. If you configure SetpointUpperLimit within the process value limits, this value is used as the setpoint high limit.
Config.SetpointLowerLimit	REAL	-3.402822e +38	Low limit of the setpoint If you set SetpointLowerLimit outside the process value limits, the configured process value absolute low limit is used as the setpoint low limit. If you set SetpointLowerLimit within the process value limits, this value is used as the setpoint low limit.
Config.MinimumOnTime	REAL	0.0	The minimum ON time of the pulse width modulation in seconds is rounded to MinimumOnTime = n×CycleTime.Value
Config.MinimumOffTime	REAL	0.0	The minimum OFF time of the pulse width modulation in seconds is rounded to MinimumOffTime = n×CycleTime.Value

Tag	Data type	Default	Description
Config.InputScaling.UpperPointIn	REAL	27648.0	Scaling Input_PER high Input_PER is converted to percent based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn.
Config.InputScaling.LowerPointIn	REAL	0.0	Scaling Input_PER low Input_PER is converted to percent based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn.
Config.InputScaling.UpperPointOut	REAL	100.0	Scaled high process value Input_PER is converted to percent based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn.
Config.InputScaling.LowerPointOut	REAL	0.0	Scaled low process value Input_PER is converted to percent based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn.
CycleTime.StartEstimation	BOOL	TRUE	If CycleTime.StartEstimation = TRUE, the automatic determination of the cycle time is started. CycleTime.StartEstimation = FALSE once measurement is complete.
CycleTime.EnEstimation	BOOL	TRUE	If CycleTime.EnEstimation = TRUE, the PID_Compact sampling time is calculated. If CycleTime.EnEstimation = FALSE, the PID_Compact sampling time is not calculated and you need to correct the configuration of CycleTime.Value manually.
CycleTime.EnMonitoring	BOOL	TRUE	If CycleTime.EnMonitoring = FALSE, the PID_Compact sampling time is not monitored. If it is not possible to execute PID_Compact within the sampling time, no error (ErrorBits=0800h) is output and PID_Compact does not switch to "Inactive" mode.
CycleTime.Value	REAL	0.1	PID_Compact sampling time in seconds CycleTime.Value is determined automatically and is usually equivalent to the cycle time of the calling OB.
CtrlParamsBackUp.Gain	REAL	1.0	Saved proportional gain You can reload values from the CtrlParamsBackUp structure with LoadBackUp = TRUE.
CtrlParamsBackUp.Ti	REAL	20.0	Saved integral action time [s]
CtrlParamsBackUp.Td	REAL	0.0	Saved derivative action time [s]
CtrlParamsBackUp.TdFiltRatio	REAL	0.2	Saved derivative delay coefficient
CtrlParamsBackUp.PWeighting	REAL	1.0	Saved proportional action weighting factor
CtrlParamsBackUp.DWeighting	REAL	1.0	Saved derivative action weighting factor
CtrlParamsBackUp.Cycle	REAL	1.0	Saved sampling time of PID algorithm

Tag	Data type	Default	Description
PIDSelfTune.SUT.CalculateParams	BOOL	FALSE	The properties of the controlled system are saved during tuning. If SUT.CalculateParams = TRUE, the parameters for pretuning are recalculated according to these properties. This enables you to change the parameter calculation method without having to repeat controller tuning. SUT.CalculateParams is set to FALSE after the calculation.
PIDSelfTune.SUT.TuneRule	INT	0	Methods used to calculate parameters during pretuning: • SUT.TuneRule = 0: PID according to Chien, Hrones and Reswick • SUT.TuneRule = 1: PI according to Chien, Hrones and Reswick
PIDSelfTune.SUT.State	INT	0	The SUT.State tag indicates the current phase of pretuning: • State = 0: Initialize pretuning • State = 100: Calculate standard deviation • State = 200: Determine point of inflection • State = 9900: Pretuning successful • State = 1: Pretuning not successful
PIDSelfTune.TIR.RunIn	BOOL	FALSE	With the RunIn tag, you can specify that fine tuning can also be performed without pretuning. • RunIn = FALSE Pretuning is started when fine tuning is started from inactive or manual mode. If the requirements for pretuning are not met, PID_Compact reacts as when RunIn = TRUE. If fine tuning is started from automatic mode, the system uses the existing PID parameters to control to the setpoint. Only then will fine tuning start. If pretuning is not possible, PID_Compact switches to the mode from which tuning was started. • RunIn = TRUE The pretuning is skipped. PID_Compact tries to reach the setpoint with minimum or maximum output value. This can produce increased overshoot. Fine tuning then starts automatically. RunIn is set to FALSE after fine tuning.
PIDSelfTune.TIR.CalculateParams	BOOL	FALSE	The properties of the controlled system are saved during tuning. If TIR.CalculateParams = TRUE, the parameters for fine tuning are recalculated according to these properties. This enables you to change the parameter calculation method without having to repeat controller tuning. TIR.CalculateParams is set to FALSE after the calculation.

Tag	Data type	Default	Description
PIDSelfTune.TIR.TuneRule	INT	0	Methods used to calculate parameters during fine tuning: • TIR.TuneRule = 0: PID automatic • TIR.TuneRule = 1: PID rapid • TIR.TuneRule = 2: PID slow • TIR.TuneRule = 3: Ziegler-Nichols PID • TIR.TuneRule = 4: Ziegler-Nichols PI • TIR.TuneRule = 5: Ziegler-Nichols P
PIDSelfTune.TIR.State	INT	0	The TIR.State tag indicates the current phase of fine tuning: • State = -100 Fine tuning is not possible. Pretuning will be performed first. • State = 0: Initialize fine tuning • State = 200: Calculate standard deviation • State = 300: Attempt to reach the setpoint • State = 400: Attempt to reach the setpoint with existing PID parameters (if pretuning was successful) • State = 500: Determine oscillation and calculate parameters • State = 9900: Fine tuning successful • State = 1: Fine tuning not successful
PIDCtrl.IntegralSum	REAL	0.0	Current integral action
Retain.CtrlParams.Gain	REAL	1.0	Active proportional gain To invert the control logic, use the Config.InvertControl tag. Negative values at Gain also invert the control logic. We recommend you use only InvertControl to set the control logic. The control logic is also inverted if InvertControl = TRUE and Gain < 0.0. Gain is retentive.
Retain.CtrlParams.Ti	REAL	20.0	• CtrlParams.Ti > 0.0: Active integral action time • CtrlParams.Ti = 0.0: Integral action is deactivated Ti is retentive.
Retain.CtrlParams.Td	REAL	0.0	• CtrlParams.Td > 0.0: Active derivative action time • CtrlParams.Td = 0.0: Derivative action is deactivated Td is retentive.

Tag	Data type	Default	Description
Retain.CtrlParams.TdFiltRatio	REAL	0.2	Active derivative delay coefficient The derivative delay coefficient delays the effect of the derivative action. Derivative delay = derivative action time × derivative delay coefficient • 0.0: Derivative action is effective for one cycle only and therefore almost not effective. • 0.5: This value has proved useful in practice for controlled systems with **one** dominant time constant. • > 1.0: The greater the coefficient, the longer the effect of the derivative action is delayed. TdFiltRatio is retentive.
Retain.CtrlParams.PWeighting	REAL	1.0	Active proportional action weighting The proportional action may weaken with changes to the setpoint. Values from 0.0 to 1.0 are applicable. • 1.0: Proportional action for setpoint change is fully effective • 0.0: Proportional action for setpoint change is not effective The proportional action is always fully effective when the process value is changed. PWeighting is retentive.
Retain.CtrlParams.DWeighting	REAL	1.0	Active derivative action weighting The derivative action may weaken with changes to the setpoint. Values from 0.0 to 1.0 are applicable. • 1.0: Derivative action is fully effective upon setpoint change • 0.0: Derivative action is not effective upon setpoint change The derivative action is always fully effective when the process value is changed. DWeighting is retentive.
Retain.CtrlParams.Cycle	REAL	1.0	Active sampling time of the PID algorithm CtrlParams.Cycle is calculated during tuning and rounded to an integer multiple of CycleTime.Value. Cycle is retentive.

Note

Change the tags listed in this table in "Inactive" mode to prevent malfunction of the PID controller.

See also

Tag ActivateRecoverMode V2 (Page 4953)
Tag Warning V2 (Page 4955)
Downloading technology objects to device (Page 9536)

Changing the PID_Compact V2 interface

The following table shows what has changed in the PID_Compact instruction interface.

PID_Compact V1	PID_Compact V2	Change
Input_PER	Input_PER	Data type from Word to Int
	Disturbance	New
	ErrorAck	New
	ModeActivate	New
Output_PER	Output_PER	Data type from Word to Int
Error	ErrorBits	Renamed
	Error	New
	Mode	New
sb_RunModeByStartup	RunModeByStartup	Function
	IntegralResetMode	
	OverwriteInitialOutputValue	New
	SetSubstituteOutput	New
	CancelTuningLevel	New
	SubstituteOutput	New

The following table shows which variables have been renamed.

PID_Compact V1.x	PID_Compact V2
sb_GetCycleTime	CycleTime.StartEstimation
sb_EnCyclEstimation	CycleTime.EnEstimation
sb_EnCyclMonitoring	CycleTime.EnMonitoring
sb_RunModeByStartup	RunModeByStartup
si_Unit	PhysicalUnit
si_Type	PhysicalQuantity
sd_Warning	Warning
sBackUp.r_Gain	CtrlParamsBackUp.Gain
sBackUp.r_Ti	CtrlParamsBackUp.Ti
sBackUp.r_Td	CtrlParamsBackUp.Td
sBackUp.r_A	CtrlParamsBackUp.TdFiltRatio
sBackUp.r_B	CtrlParamsBackUp.PWeighting
sBackUp.r_C	CtrlParamsBackUp.DWeighting
sBackUp.r_Cycle	CtrlParamsBackUp.Cycle

PID_Compact V1.x	PID_Compact V2
sPid_Calc.r_Cycle	CycleTime.Value
sPid_Calc.b_RunIn	PIDSelfTune.TIR.RunIn
sPid_Calc.b_CalcParamSUT	PIDSelfTune.SUT.CalculateParams
sPid_Calc.b_CalcParamTIR	PIDSelfTune.TIR.CalculateParams
sPid_Calc.i_CtrlTypeSUT	PIDSelfTune.SUT.TuneRule
sPid_Calc.i_CtrlTypeTIR	PIDSelfTune.TIR.TuneRule
sPid_Calc.r_Progress	Progress
sPid_Cmpt.r_Sp_Hlm	Config.SetpointUpperLimit
sPid_Cmpt.r_Sp_Llm	Config.SetpointLowerLimit
sPid_Cmpt.r_Pv_Norm_IN_1	Config.InputScaling.LowerPointIn
sPid_Cmpt.r_Pv_Norm_IN_2	Config.InputScaling.UpperPointIn
sPid_Cmpt.r_Pv_Norm_OUT_1	Config.InputScaling.LowerPointOut
sPid_Cmpt.r_Pv_Norm_OUT_2	Config.InputScaling.UpperPointOut
sPid_Cmpt.r_Lmn_Hlm	Config.OutputUpperLimit
sPid_Cmpt.r_Lmn_Llm	Config.OutputLowerLimit
sPid_Cmpt.b_Input_PER_On	Config.InputPerOn
sPid_Cmpt.b_LoadBackUp	LoadBackUp
sPid_Cmpt.b_InvCtrl	Config.InvertControl
sPid_Cmpt.r_Lmn_Pwm_PPTm	Config.MinimumOnTime
sPid_Cmpt.r_Lmn_Pwm_PBTm	Config.MinimumOffTime
sPid_Cmpt.r_Pv_Hlm	Config.InputUpperLimit
sPid_Cmpt.r_Pv_Llm	Config.InputLowerLimit
sPid_Cmpt.r_Pv_HWrn	Config.InputUpperWarning
sPid_Cmpt.r_Pv_LWrn	Config.InputLowerWarning
sParamCalc.i_Event_SUT	PIDSelfTune.SUT.State
sParamCalc.i_Event_TIR	PIDSelfTune.TIR.State
sRet.i_Mode	sRet.i_Mode has been omitted. The operating mode is changed using Mode and ModeActivate.
sRet.r_Ctrl_Gain	Retain.CtrlParams.Gain
sRet.r_Ctrl_Ti	Retain.CtrlParams.Ti
sRet.r_Ctrl_Td	Retain.CtrlParams.Td
sRet.r_Ctrl_A	Retain.CtrlParams.TdFiltRatio
sRet.r_Ctrl_B	Retain.CtrlParams.PWeighting
sRet.r_Ctrl_C	Retain.CtrlParams.DWeighting
sRet.r_Ctrl_Cycle	Retain.CtrlParams.Cycle

Parameters State and Mode V2

Correlation of the parameters

The State parameter shows the current operating mode of the PID controller. You cannot change the State parameter.

With a rising edge at ModeActivate, PID_Compact switches to the operating mode saved in the Mode in-out parameter.

When the CPU is switched on or switches from Stop to RUN mode, PID_Compact starts in the operating mode that is saved in the Mode parameter. To leave PID_Compact in "Inactive" mode, set RunModeByStartup = FALSE.

Meaning of values

State / Mode	Description of operating mode
0	Inactive In "Inactive" operating mode, the output value 0.0 is always output, regardless of Config.OutputUpperLimit and Config.OutputLowerLimit. Pulse width modulation is off.
1	Pretuning The pretuning determines the process response to a jump change of the output value and searches for the point of inflection. The PID parameters are calculated from the maximum rate of rise and dead time of the controlled system. You obtain the best PID parameters when you perform pretuning and fine tuning. Pretuning requirements: • Inactive (State = 0), manual mode (State = 4), or automatic mode (State = 3) • ManualEnable = FALSE • Reset = FALSE • The process value must not be too close to the setpoint. |Setpoint - Input| > 0.3 * | Config.InputUpperLimit - Config.InputLowerLimit| and |Setpoint - Input| > 0.5 * |Setpoint| • The setpoint and the process value lie within the configured limits. The more stable the process value is, the easier it is to calculate the PID parameters and the more precise the result will be. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher compared to the noise. The setpoint is frozen in the CurrentSetpoint tag. Tuning is canceled when: • Setpoint > CurrentSetpoint + CancelTuningLevel or • Setpoint < CurrentSetpoint - CancelTuningLevel Before the PID parameters are recalculated, they are backed up and can be reactivated with LoadBackUp. The controller switches to automatic mode following successful pretuning. If pretuning is unsuccessful, the switchover of the operating mode is dependent on ActivateRecoverMode. The phase of pretuning is indicated with PIDSelfTune.SUT.State.

State / Mode	Description of operating mode
2	Fine tuning Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are recalculated based on the amplitude and frequency of this oscillation. PID parameters from fine tuning usually have better master control and disturbance characteristics than PID parameters from pretuning. You obtain the best PID parameters when you perform pretuning and fine tuning. PID_Compact automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value. The setpoint is frozen in the CurrentSetpoint tag. Tuning is canceled when: • Setpoint > CurrentSetpoint + CancelTuningLevel or • Setpoint < CurrentSetpoint - CancelTuningLevel Before the PID parameters are recalculated, they are backed up and can be reactivated with LoadBackUp. Requirements for fine tuning: • No disturbances are expected. • The setpoint and the process value lie within the configured limits. • ManualEnable = FALSE • Reset = FALSE • Automatic (State = 3), inactive (State = 0) or manual (State = 4) mode Fine tuning proceeds as follows when started from: • Automatic mode (State = 3) Start fine tuning from automatic mode if you wish to improve the existing PID parameters through tuning. PID_Compact controls the system using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start. • Inactive (State = 0) or manual mode (State = 4) If the requirements for pretuning are met, pretuning is started. The determined PID parameters will be used for control until the control loop has stabilized and the requirements for fine tuning have been met. If the process value for pretuning is already too near the setpoint or PIDSelfTune.TIR.RunIn = TRUE, an attempt is made to reach the setpoint with the minimum or maximum output value. This can produce increased overshoot. Only then will fine tuning start. The controller switches to automatic mode following successful fine tuning. If fine tuning is unsuccessful, the switchover of the operating mode is dependent on ActivateRecoverMode. The "Fine tuning" phase is indicated with PIDSelfTune.TIR.State.
3	Automatic mode In automatic mode, PID_Compact corrects the controlled system in accordance with the parameters specified. The controller switches to automatic mode if one the following requirements is fulfilled: • Pretuning successfully completed • Fine tuning successfully completed • Changing of the Mode in-out parameter to the value 3 and a rising edge at ModeActivate. The switchover from automatic mode to manual mode is only bumpless if carried out in the commissioning editor. The ActivateRecoverMode tag is taken into consideration in automatic mode.

State / Mode	Description of operating mode
4	Manual mode In manual mode, you specify a manual output value in the ManualValue parameter. You can also activate this operating mode using ManualEnable = TRUE. We recommend that you change the operating mode using Mode and ModeActivate only. The switchover from manual mode to automatic mode is bumpless. Manual mode is also possible when an error is pending.
5	Substitute output value with error monitoring The control algorithm is deactivated. The SetSubstituteOutput tag determines which output value is output in this operating mode. • SetSubstituteOutput = FALSE: Last valid output value • SetSubstituteOutput = TRUE: Substitute output value You cannot activate this operating mode using Mode = 5. In the event of an error, it is activated instead of "Inactive" operating mode if all the following conditions are met: • Automatic mode (Mode = 3) • ActivateRecoverMode = TRUE • One or more errors have occurred in which ActivateRecoverMode is effective. As soon as the errors are no longer pending, PID_Compact switches back to automatic mode.

ENO characteristics

If State = 0, then ENO = FALSE.

If State ≠ 0, then ENO = TRUE.

Automatic switchover of operating mode during commissioning

Automatic mode is activated following successful pretuning or fine tuning. The following table shows how Mode and State change during successful pretuning.

Cycle no.	Mode	State	Action
0	4	4	Set Mode = 1
1	1	4	Set ModeActivate = TRUE
1	4	1	Value of State is saved in Mode parameter Pretuning is started
n	4	1	Pretuning successfully completed
n	3	3	Automatic mode is started

PID_Compact automatically switches the operating mode in the event of an error. The following table shows how Mode and State change during pretuning with errors.

Cycle no.	Mode	State	Action
0	4	4	Set Mode = 1
1	1	4	Set ModeActivate = TRUE

Cycle no.	Mode	State	Action
1	4	1	Value of State is saved in Mode parameter Pretuning is started
n	4	1	Pretuning canceled
n	4	4	Manual mode is started

If ActivateRecoverMode = TRUE, the operating mode that is saved in the Mode parameter is activated. At the start of pretuning or fine tuning, PID_Compact has saved the value of State in the Mode in/out parameter. PID_Compact therefore switches to the operating mode from which tuning was started.

If ActivateRecoverMode = FALSE, the system switches to "Inactive" operating mode.

See also

Output parameters of PID_Compact V2 (Page 4937)

Parameter ErrorBits V2

If several errors are pending simultaneously, the values of the ErrorBits are displayed with binary addition. The display of ErrorBits = 0003h, for example, indicates that the errors 0001h and 0002h are pending simultaneously.

In manual mode, PID_Compact uses ManualValue as output value. The exception is Errorbits = 10000h.

ErrorBits (DW#16#...)	Description
0000	There is no error.
0001	The "Input" parameter is outside the process value limits. • Input > Config.InputUpperLimit or • Input < Config.InputLowerLimit If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact remains in automatic mode. If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact switches to the operating mode that is saved in the Mode parameter.
0002	Invalid value at "Input_PER" parameter. Check whether an error is pending at the analog input. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact outputs the configured substitute output value. As soon as the error is no longer pending, PID_Compact switches back to automatic mode. If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact switches to the operating mode that is saved in the Mode parameter.
0004	Error during fine tuning. Oscillation of the process value could not be maintained. If ActivateRecoverMode = TRUE before the error occurred, PID_Compact cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0008	Error at start of pretuning. The process value is too close to the setpoint. Start fine tuning. If ActivateRecoverMode = TRUE before the error occurred, PID_Compact cancels the tuning and switches to the operating mode that is saved in the Mode parameter.

ErrorBits (DW#16#...)	Description
0010	The setpoint was changed during tuning. You can set the permitted fluctuation of the setpoint at the CancelTuningLevel tag. If ActivateRecoverMode = TRUE before the error occurred, PID_Compact cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0020	Pretuning is not permitted during fine tuning. If ActivateRecoverMode = TRUE before the error occurred, PID_Compact remains in fine tuning mode.
0080	Error during pretuning. Incorrect configuration of output value limits. Check whether the limits of the output value are configured correctly and match the control logic. If ActivateRecoverMode = TRUE before the error occurred, PID_Compact cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0100	Error during fine tuning resulted in invalid parameters. If ActivateRecoverMode = TRUE before the error occurred, PID_Compact cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0200	Invalid value at "Input" parameter: Value has an invalid number format. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact outputs the configured substitute output value. As soon as the error is no longer pending, PID_Compact switches back to automatic mode. If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact switches to the operating mode that is saved in the Mode parameter.
0400	Calculation of output value failed. Check the PID parameters. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact outputs the configured substitute output value. As soon as the error is no longer pending, PID_Compact switches back to automatic mode. If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact switches to the operating mode that is saved in the Mode parameter.
0800	Sampling time error: PID_Compact is not called within the sampling time of the cyclic interrupt OB. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact remains in automatic mode. If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact switches to the operating mode that is saved in the Mode parameter.
1000	Invalid value at "Setpoint" parameter: Value has an invalid number format. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact outputs the configured substitute output value. As soon as the error is no longer pending, PID_Compact switches back to automatic mode. If pretuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact switches to the operating mode that is saved in the Mode parameter.
10000	Invalid value at ManualValue parameter. Value has an invalid number format. If ActivateRecoverMode = TRUE before an error occurred, PID_Compact uses SubstituteOutput as the output value. As soon as you specify a valid value in ManualValue, PID_Compact uses it as the output value.

ErrorBits (DW#16#...)	Description
20000	Invalid value at SubstituteOutput tag. Value has an invalid number format. PID_Compact uses the output value low limit as the output value. If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_Compact switches back to automatic mode.
40000	Invalid value at Disturbance parameter. Value has an invalid number format. If automatic mode was active and ActivateRecoverMode = TRUE before the error occurred, Disturbance is set to zero. PID_Compact remains in automatic mode. If pretuning or fine tuning mode was active and ActivateRecoverMode = TRUE before the error occurred, PID_Compact switches to the operating mode saved in the Mode parameter. If Disturbance in the current phase has no effect on the output value, tuning is not be canceled.

Tag ActivateRecoverMode V2

The ActivateRecoverMode tag determines the reaction to error. The Error parameter indicates if an error is pending. When the error is no longer pending, Error = FALSE. The ErrorBits parameter shows which errors have occurred.

Automatic mode

NOTICE
Your system may be damaged. If ActivateRecoverMode = TRUE, PID_Compact remains in automatic mode even if there is an error and the process limit values are exceeded. This may damage your system. It is essential to configure how your controlled system reacts in the event of an error to protect your system from damage.

ActivateRecover Mode	Description
FALSE	PID_Compact automatically switches to "Inactive" mode in the event of an error. The controller is only activated by a falling edge at Reset or a rising edge at ModeActivate.
TRUE	If errors occur frequently in automatic mode, this setting has a negative effect on the control response, because PID_Compact switches between the calculated output value and the substitute output value at each error. In this case, check the ErrorBits parameter and eliminate the cause of the error. If one or more of the following errors occur, PID_Compact stays in automatic mode: • 0001h: The "Input" parameter is outside the process value limits. • 0800h: Sampling time error • 40000h: Invalid value at parameter Disturbance. If one or more of the following errors occur, PID_Compact switches to "Substitute output value with error monitoring" mode: • 0002h: Invalid value at Input_PER parameter. • 0200h: Invalid value at Input parameter. • 0400h: Calculation of output value failed. • 1000h: Invalid value at Setpoint parameter. If the following error occurs, PID_Compact switches to "Substitute output value with error monitoring" mode and moves the actuator to Config.OutputLowerLimit: • 20000h: Invalid value at SubstituteOutput tag. Value has an invalid number format. This characteristics are independent of SetSubstituteOutput. As soon as the errors are no longer pending, PID_Compact switches back to automatic mode.

Pretuning and fine tuning

ActivateRecover Mode	Description
FALSE	PID_Compact automatically switches to "Inactive" mode in the event of an error. The controller is only activated by a falling edge at Reset or a rising edge at ModeActivate.
TRUE	If the following error occurs, PID_Compact remains in the active mode: • 0020h: Pretuning is not permitted during fine tuning. The following errors are ignored: • 10000h: Invalid value at ManualValue parameter. • 20000h: Invalid value at SubstituteOutput tag. When any other error occurs, PID_Compact cancels the tuning and switches to the mode from which tuning was started.

Manual mode

ActivateRecoverMode is not effective in manual mode.

Tag Warning V2

If several warnings are pending simultaneously, the values of the Warning tag are displayed with binary addition. The display of warning 0003h, for example, indicates that the warnings 0001h and 0002h are pending simultaneously.

Warning (DW#16#....)	Description
0000	No warning pending.
0001	The point of inflection was not found during pretuning.
0004	The setpoint was limited to the configured limits.
0008	Not all the necessary controlled system properties were defined for the selected method of calculation. Instead, the PID parameters were calculated using the TIR.TuneRule = 3 method.
0010	The operating mode could not be changed because Reset = TRUE or ManualEnable = TRUE.
0020	The cycle time of the calling OB limits the sampling time of the PID algorithm. Improve results by using shorter OB cycle times.
0040	The process value exceeded one of its warning limits.
0080	Invalid value at Mode. The operating mode is not switched.
0100	The manual value was limited to the limits of the controller output.
0200	The specified rule for tuning is not supported. No PID parameters are calculated.
1000	The substitute output value cannot be reached because it is outside the output value limits.

The following warnings are deleted as soon as the cause is eliminated:

- 0001h
- 0004h
- 0008h
- 0040h
- 0100h

All other warnings are cleared with a rising edge at Reset or ErrorAck.

PID_Compact V1

Description of PID_Compact V1

Description

The PID_Compact instruction provides a PID controller with integrated tuning for automatic and manual mode.

Call

PID_Compact is called in the constant interval of the cycle time of the calling OB (preferably in a cyclic interrupt OB).

Download to device

The actual values of retentive tags are only updated when you download PID_Compact completely.

Downloading technology objects to device (Page 9536)

Startup

At the startup of the CPU, PID_Compact starts in the operating mode that was last active. To retain PID_ Compact in "Inactive" mode, set sb_RunModeByStartup = FALSE.

Monitoring of the sampling time PID_Compact

Ideally, the sampling time is equivalent to the cycle time of the calling OB. The PID_Compact instruction measures the time interval between two calls. This is the current sampling time. On every switchover of operating mode and during the initial startup, the mean value is formed from the first 10 sampling times. If the current sampling time deviates too much from this mean value, Error = 0800 hex occurs and PID_Compact switches to "Inactive" mode.

PID_Compact, Version 1.1 or higher is set to "Inactive" mode during controller tuning under the following conditions:

- New mean value >= 1.1 x old mean value
- New mean value <= 0.9 x old mean value

In automatic mode, PID_Compact, Version 1.1 or higher, is set to "Inactive" mode under the following conditions:

- New mean value >= 1.5 x old mean value
- New mean value <= 0.5 x old mean value

During controller tuning and in automatic mode, PID_Compact 1.0 is set to "Inactive" operating mode under the following conditions:

- New mean value >= 1.1 x old mean value
- New mean value <= 0.9 x old mean value
- Current sampling time >= 1.5 x current mean value
- Current sampling time <= 0.5 x current mean value

Sampling time of the PID algorithm

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is calculated during tuning and rounded to a multiple of the cycle time. All other functions of PID_Compact are executed at every call.

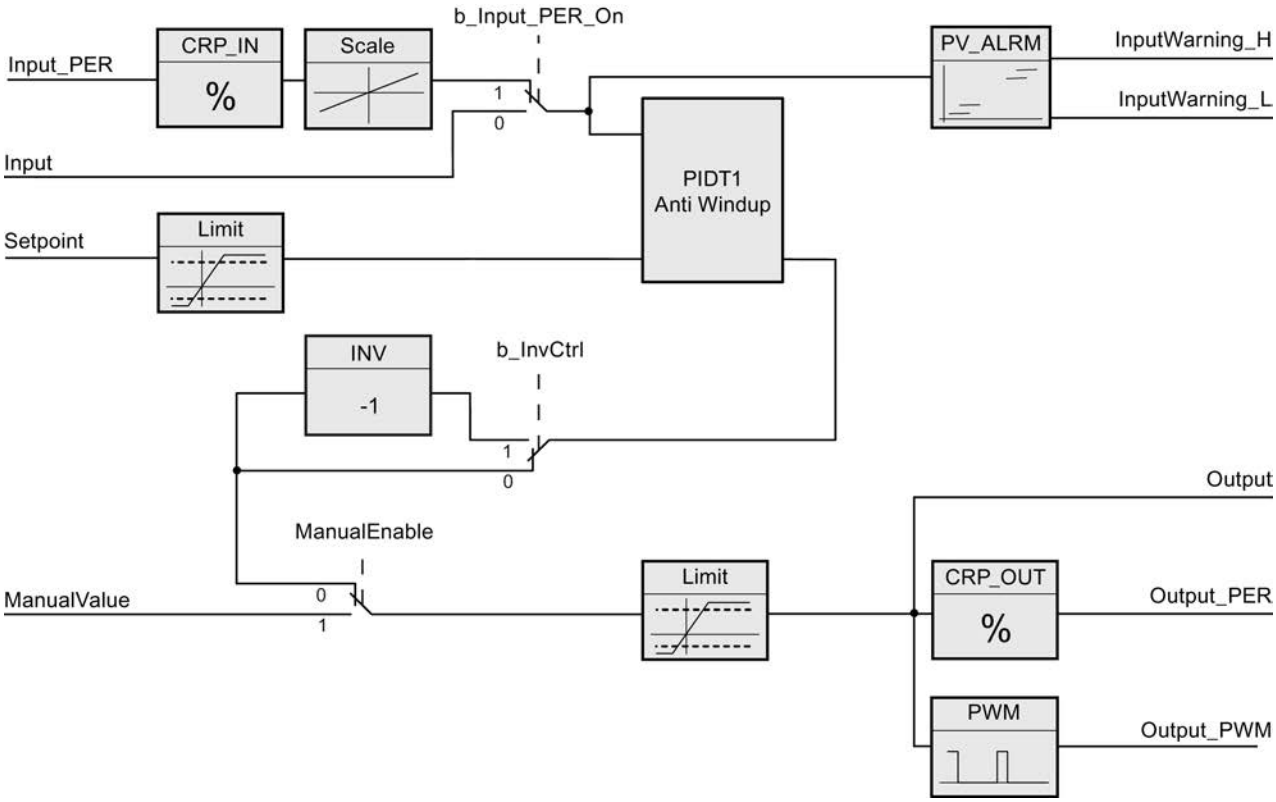
PID algorithm

PID_Compact is a PIDT1 controller with anti-windup and weighting of the proportional and derivative actions. The following equation is used to calculate the output value.

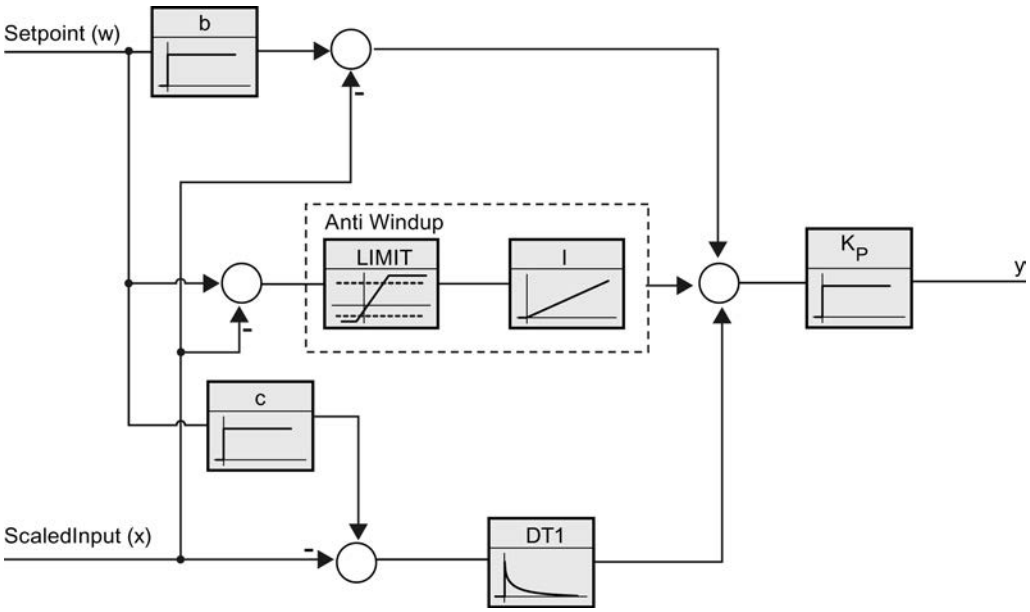
$$y = K_p \left[(b \cdot w - x) + \frac{1}{T_I \cdot s}(w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1}(c \cdot w - x) \right]$$

Symbol	Description
y	Output value
K_p	Proportional gain
s	Laplace operator
b	Proportional action weighting
w	Setpoint
x	Process value
T_I	Integral action time
a	Derivative delay coefficient (T1 = a × T_D)
	Derivative action time
c	Derivative action weighting

Block diagram of PID_Compact



Block diagram of PIDT1 with anti-windup



Reaction to error

If errors occur, they are output in parameter Error, and PID_Compact changes to "Inactive" mode. Reset the errors using the Reset parameter.

Control logic

An increase of the output value is generally intended to cause an increase in the process value. This is referred to as a normal control logic. For cooling and discharge control systems, it may be necessary to invert the control logic. PID_Compact does not work with negative proportional gain. If InvertControl = TRUE, an increasing control deviation causes a reduction in the output value. The control logic is also taken into account during pretuning and fine tuning.

See also

Controller type (Page 9575)

Input parameters of PID_Compact V1

Table 9-85

Parameter	Data type	Default	Description
Setpoint	REAL	0.0	Setpoint of the PID controller in automatic mode
Input	REAL	0.0	A variable of the user program is used as source for the process value. If you are using parameter Input, then sPid_Cmpt.b_Input_PER_On = FALSE must be set.

Parameter	Data type	Default	Description
Input_PER	WORD	W#16#0	Analog input as the source of the process value If you are using parameter Input_PER, then sPid_Cmpt.b_Input_PER_On = TRUE must be set.
ManualEnable	BOOL	FALSE	• A FALSE -> TRUE edge selects "Manual mode", while State = 4, sRet.i_Mode remains unchanged. • A TRUE -> FALSE edge selects the most recently active operating mode, State =sRet.i_Mode A change of sRet.i_Mode will not take effect during ManualEnable = TRUE. The change of sRet.i_Mode will only be considered upon a TRUE -> FALSE edge at ManualEnable . **PID_Compact V1.2 und PID_Compact V1.0** If at start of the CPU ManualEnable = TRUE, PID_Compact starts in manual mode. A rising edge (FALSE > TRUE) at ManualEnable is not necessary. **PID_Compact V1.1** At the start of the CPU, PID_Compact only switches to manual mode with a rising edge (FALSE->TRUE) at ManualEnable . Without rising edge, PID_Compact starts in the last operating mode in which ManualEnable was FALSE.
ManualValue	REAL	0.0	Manual value This value is used as the output value in manual mode.
Reset	BOOL	FALSE	The Reset parameter (Page 4968) restarts the controller.

Output parameters of PID_Compact V1

Table 9-86

Parameter	Data type	Default	Description
ScaledInput	REAL	0.0	Output of the scaled process value
Outputs "Output", "Output_PER", and "Output_PWM" can be used concurrently.			
Output	REAL	0.0	Output value in REAL format
Output_PER	WORD	W#16#0	Analog output value
Output_PWM	BOOL	FALSE	Pulse-width-modulated output value The output value is formed by minimum On and Off times.
SetpointLimit_H	BOOL	FALSE	If SetpointLimit_H = TRUE, the setpoint absolute high limit is reached. The setpoint in the CPU is limited to the configured setpoint absolute high limit. The configured process value absolute high limit is the default for the setpoint high limit. If you set sPid_Cmpt.r_Sp_HIm to a value within the process value limits, this value is used as the setpoint high limit.
SetpointLimit_L	BOOL	FALSE	If SetpointLimit_L = TRUE, the setpoint absolute low limit has been reached. In the CPU, the setpoint is limited to the configured setpoint absolute low limit. The configured process value absolute low limit is the default setting for the setpoint low limit. If you set sPid_Cmpt.r_Sp_LIm to a value within the process value limits, this value is used as the setpoint low limit.

Parameter	Data type	Default	Description
InputWarning_H	BOOL	FALSE	If InputWarning_H = TRUE, the process value has reached or exceeded the warning high limit.
InputWarning_L	BOOL	FALSE	If InputWarning_L = TRUE, the process value has reached or fallen below the warning low limit.
State	INT	0	The State parameter (Page 4965) shows the current operating mode of the PID controller. To change the operating mode, use variable sRet.i_Mode. • State = 0: Inactive • State = 1: pretuning • State = 2: fine tuning • State = 3: Automatic mode • State = 4: Manual mode
Error	DWORD	W#16#0	The Error parameter (Page 4968) indicates the error messages. Error = 0000: No error pending.

Static tags of PID_Compact V1

You must not change tags that are not listed. These are used for internal purposes only.

Table 9-87

Tag	Data type	Default	Description
sb_GetCycleTime	BOOL	TRUE	If sb_GetCycleTime = TRUE, the automatic determination of the cycle time is started. CycleTime.StartEstimation = FALSE once measurement is complete.
sb_EnCyclEstimation	BOOL	TRUE	If sb_EnCyclEstimation = TRUE, the sampling time PID_Compact is calculated.
sb_EnCyclMonitoring	BOOL	TRUE	If sb_EnCyclMonitoring = FALSE, the sampling time PID_Compact is not monitored. If it is not possible to execute PID_Compact within the sampling time, an 0800 error is not output and PID_Compact does not change to "Inactive" mode.
sb_RunModeByStartup	BOOL	TRUE	Activate Mode after CPU restart If sb_RunModeByStartup = FALSE, the controller will remain inactive after a CPU startup. After a CPU startup and if sb_RunModeByStartup = TRUE, the controller will return to the most recently active operating mode.
si_Unit	INT	0	Unit of measurement of the process value and setpoint, e.g., ºC, or ºF.
si_Type	INT	0	Physical quantity of the process value and setpoint, e.g., temperature.
sd_Warning	DWORD	DW#16#0	Variable sd_warning (Page 4970) displays the warnings generated since the reset, or since the last change of the operating mode.

Tag	Data type	Default	Description
sBackUp.r_Gain	REAL	1.0	Saved proportional gain You can reload values from the sBackUp structure with sPid_Cmpt.b_LoadBackUp = TRUE.
sBackUp.r_Ti	REAL	20.0	Saved integral action time [s]
sBackUp.r_Td	REAL	0.0	Saved derivative action time [s]
sBackUp.r_A	REAL	0.0	Saved derivative delay coefficient
sBackUp.r_B	REAL	0.0	Saved proportional action weighting factor
sBackUp.r_C	REAL	0.0	Saved derivative action weighting factor
sBackUp.r_Cycle	REAL	1.0	Saved sampling time of PID algorithm
sPid_Calc.r_Cycle	REAL	0.1	Sampling time of the PID_Compact instruction r_Cycle is determined automatically and usually equivalent to the cycle time of the calling OB.
sPid_Calc.b_RunIn	BOOL	FALSE	• b_RunIn = FALSE Pretuning is started when fine tuning is started from inactive or manual mode. If the requirements for pretuning are not met, PID_Compact reacts like b_RunIn = TRUE. If fine tuning is started from automatic mode, the system uses the existing PID parameters to control to the setpoint. Only then will fine tuning start. If pretuning is not possible, PID_Compact will change to "Inactive" mode. • b_RunIn = TRUE The pretuning is skipped. PID_3Compact tries to reach the setpoint with minimum or maximum output value. This can produce increased overshoot. Fine tuning then starts automatically. b_RunIn is set to FALSE after fine tuning.
sPid_Calc.b_CalcParamSUT	BOOL	FALSE	The parameters for pretuning will be recalculated if b_CalcParamSUT = TRUE. This enables you to change the parameter calculation method without having to repeat controller tuning. b_CalcParamSUT will be set to FALSE after calculation.
sPid_Calc.b_CalcParamTIR	BOOL	FALSE	The parameters for fine tuning will be recalculated if b_CalcParamTIR = TRUE. This enables you to change the parameter calculation method without having to repeat controller tuning. b_CalcParamTIR will be set to FALSE after calculation.
sPid_Calc.i_CtrlTypeSUT	INT	0	Methods used to calculate parameters during pretuning: • i_CtrlTypeSUT = 0: PID according to Chien, Hrones and Reswick • i_CtrlTypeSUT = 1: PI according to Chien, Hrones and Reswick

Tag	Data type	Default	Description
sPid_Calc.i_CtrlTypeTIR	INT	0	Methods used to calculate parameters during fine tuning: • i_CtrlTypeTIR = 0: PID automatic • i_CtrlTypeTIR = 1: PID rapid • i_CtrlTypeTIR = 2: PID slow • i_CtrlTypeTIR = 3: Ziegler-Nichols PID • i_CtrlTypeTIR = 4: Ziegler-Nichols PI • i_CtrlTypeTIR = 5: Ziegler-Nichols P
sPid_Calc.r_Progress	REAL	0.0	Progress of tuning as a percentage (0.0 - 100.0)
sPid_Cmpt.r_Sp_Hlm	REAL	+3.402822e +38	High limit of setpoint If you set sPid_Cmpt.r_Sp_Hlm outside the process value limits, the configured process value absolute high limit is used as the setpoint high limit. If you set sPid_Cmpt.r_Sp_Hlm within the process value limits, this value is used as the setpoint high limit.
sPid_Cmpt.r_Sp_Llm	REAL	-3.402822e +38	Low limit of the setpoint If you set sPid_Cmpt.r_Sp_Llm outside the process value limits, the configured process value absolute low limit is used as the setpoint low limit. If you set sPid_Cmpt.r_Sp_Llm within the process value limits, this value is used as the setpoint low limit.
sPid_Cmpt.r_Pv_Norm_IN_1	REAL	0.0	Scaling Input_PER low Input_PER is converted to percent based on the two value pairs r_Pv_Norm_OUT_1, r_Pv_Norm_IN_1 and r_Pv_Norm_OUT_2, r_Pv_Norm_IN_2 from the sPid_Cmpt structure.
sPid_Cmpt.r_Pv_Norm_IN_2	REAL	27648.0	Scaling Input_PER high Input_PER is converted to percent based on the two value pairs r_Pv_Norm_OUT_1, r_Pv_Norm_IN_1 and r_Pv_Norm_OUT_2, r_Pv_Norm_IN_2 from the sPid_Cmpt structure.
sPid_Cmpt.r_Pv_Norm_OUT_1	REAL	0.0	Scaled low process value Input_PER is converted to percent based on the two value pairs r_Pv_Norm_OUT_1, r_Pv_Norm_IN_1 and r_Pv_Norm_OUT_2, r_Pv_Norm_IN_2 from the sPid_Cmpt structure.
sPid_Cmpt.r_Pv_Norm_OUT_2	REAL	100.0	Scaled high process value Input_PER is converted to percent based on the two value pairs r_Pv_Norm_OUT_1, r_Pv_Norm_IN_1 and r_Pv_Norm_OUT_2, r_Pv_Norm_IN_2 from the sPid_Cmpt structure.
sPid_Cmpt.r_Lmn_Hlm	REAL	100.0	Output value high limit for output parameter "Output"
sPid_Cmpt.r_Lmn_Llm	REAL	0.0	Low output value limit for output parameter "Output"
sPid_Cmpt.b_Input_PER_On	BOOL	TRUE	If b_Input_PER_On = TRUE, then parameter Input_PER is used. If b_Input_PER_On = FALSE, then parameter Input is used.

Tag	Data type	Default	Description
sPid_Cmpt.b_LoadBackUp	BOOL	FALSE	Activate the back-up parameter set. If an optimization has failed, you can reactivate the previous PID parameters by setting this bit.
sPid_Cmpt.b_InvCtrl	BOOL	FALSE	Invert control logic With b_InvCtrl = TRUE, a rising control deviation reduces the output value.
sPid_Cmpt.r_Lmn_Pwm_PPTm	REAL	0.0	The minimum ON time of the pulse width modulation in seconds is rounded to r_Lmn_Pwm_PPTm = r_Cycle or r_Lmn_Pwm_PPTm = n*r_Cycle
sPid_Cmpt.r_Lmn_Pwm_PBTm	REAL	0.0	The minimum OFF time of the pulse width modulation in seconds is rounded to r_Lmn_Pwm_PBTm = r_Cycle or r_Lmn_Pwm_PBTm = n*r_Cycle
sPid_Cmpt.r_Pv_Hlm	REAL	120.0	High limit of the process value At the I/O input, the process value can be a maximum of 18% higher than the standard range (overrange). An error is no longer reported for a violation of the "Process value high limit". Only a wire-break and a short-circuit are recognized and the PID_Compact switches to "Inactive" mode. r_Pv_Hlm > r_Pv_Llm
sPid_Cmpt.r_Pv_Llm	REAL	0.0	Low limit of the process value r_Pv_Llm < r_Pv_Hlm
sPid_Cmpt.r_Pv_HWrn	REAL	+3.402822e +38	Warning high limit of the process value If you set r_Pv_HWrn outside the process value limits, the configured process value absolute high limit is used as the warning high limit. If you set r_Pv_HWrn within the process value limits, this value is used as the warning high limit. r_Pv_HWrn > r_Pv_LWrn r_Pv_HWrn ≤ r_Pv_Hlm
sPid_Cmpt.r_Pv_LWrn	REAL	-3.402822e +38	Warning low limit of the process value If you set r_Pv_LWrn outside the process value limits, the configured process value absolute low limit is used as the warning low limit. If you set r_Pv_LWrn within the process value limits, this value is used as the warning low limit. r_Pv_LWrn < r_Pv_HWrn r_Pv_LWrn ≥ r_Pv_LWrn
sParamCalc.i_Event_SUT	INT	0	Variable i_Event_SUT (Page 4971) indicates the current phase of "pretuning":
sParamCalc.i_Event_TIR	INT	0	Variable i_Event_TIR (Page 4971) indicates the current phase of "fine tuning":

Tag	Data type	Default	Description
sRet.i_Mode	INT	0	The operating mode is changed edge-triggered. The following operating mode is enabled on a change to • i_Mode = 0: "Inactive" (controller stop) • i_Mode = 1: "Pretuning" mode • i_Mode = 2: "Fine tuning" mode • i_Mode = 3: "Automatic mode" • i_Mode = 4: "Manual mode" i_Mode is retentive.
sRet.r_Ctrl_Gain	REAL	1.0	Active proportional gain Gain is retentive.
sRet.r_Ctrl_Ti	REAL	20.0	• r_Ctrl_Ti > 0.0: active integral action time • r_Ctrl_Ti = 0.0: Integral action is disabled r_Ctrl_Ti is retentive.
sRet.r_Ctrl_Td	REAL	0.0	• r_Ctrl_Td > 0.0: Active derivative action time • r_Ctrl_Td = 0.0: Derivative action is disabled r_Ctrl_Td is retentive.
sRet.r_Ctrl_A	REAL	0.0	Active derivative delay coefficient r_Ctrl_A is retentive.
sRet.r_Ctrl_B	REAL	0.0	Active proportional action weighting r_Ctrl_B is retentive.
sRet.r_Ctrl_C	REAL	0.0	Active derivative action weighting r_Ctrl_C is retentive.
sRet.r_Ctrl_Cycle	REAL	1.0	Active sampling time of the PID algorithm r_Ctrl_Cycle is calculated during controller tuning and rounded to an integer multiple of r_Cycle. r_Ctrl_Cycle is retentive.

Note

Change the tags listed in this table in "Inactive" mode to prevent malfunction of the PID controller. The "Inactive" mode is forced by setting variable "sRet.i_Mode" to "0".

See also

Downloading technology objects to device (Page 9536)

Parameters State and sRet.i_Mode V1

Correlation of the parameters

The State parameter indicates the current operating mode of the PID controller. You cannot modify the State parameter.

You need to modify the sRet.i_Mode tag to change the operating mode. This also applies when the value for the new operating mode is already in sRet.i_Mode. First set sRet.i_Mode = 0 and then sRet.i_Mode = 3. Provided the current operating mode of the controller supports this change, State is set to the value of sRet.i_Mode.

When PID_Compact automatically switches the operating mode, the following applies: State ! = sRet.i_Mode.

Examples:

- Successful pretuning
  State = 3 and sRet.i_Mode = 1
- Error
  State = 0 and sRet.i_Mode remains at the same value, e.g sRet.i_Mode = 3
- ManualEnalbe = TRUE
  State = 4 and sRet.i_Mode remain at the previous value, for example, sRet.i_Mode = 3

Note

You wish to repeat successful fine tuning without exiting automatic mode with i_Mode = 0.

Setting sRet.i_Mode to an invalid value such as 9999 for one cycle has no effect on State. Set Mode = 2 in the next cycle. You can generate a change to sRet.i_Mode without first switching to "inactive" mode.

Meaning of values

State / sRet.i_Mode	Description of the operating mode
0	Inactive The controller is switched off. The controller was in "inactive" mode before pretuning was performed. The PID controller will change to "inactive" mode when running if an error occurs or if the "Deactivate controller" icon is clicked in the commissioning window.
1	Pretuning The pretuning determines the process response to a jump of the output value and searches for the point of inflection. The optimized PID parameters are calculated as a function of the maximum rate of rise and dead time of the controlled system. Pretuning requirements: • The controller is in inactive mode or manual mode • ManualEnable = FALSE • The process value must not be too close to the setpoint. |Setpoint - Input| > 0.3 * |sPid_Cmpt.r_Pv_Hlm - sPid_Cmpt.r_Pv_Llm| and |Setpoint - Input| > 0.5 * |Setpoint| • The setpoint may not be changed during pretuning. The higher the stability of the process value, the easier it is to calculate the PID parameters and increase precision of the result. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher compared to the noise. PID parameters are backed up before they are recalculated and can be reactivated with sPid_Cmpt.b_LoadBackUp. There is a change to automatic mode following successful pretuning and to "inactive" mode following unsuccessful pretuning. The phase of pretuning is indicated with Tag i_Event_SUT V1 (Page 4971).

State / sRet.i_Mode	Description of the operating mode
2	Fine tuning Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are optimized based on the amplitude and frequency of this oscillation. The differences between the process response during pretuning and fine tuning are analyzed. All PID parameters are recalculated on the basis of the findings. PID parameters from fine tuning usually have better master control and disturbance behavior than PID parameters from pretuning. PID_Compact automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value. PID parameters are backed up before they are recalculated and can be reactivated with sPid_Cmpt.b_LoadBackUp. Requirements for fine tuning: • No disturbances are expected. • The setpoint and the process value lie within the configured limits. • The setpoint may not be changed during fine tuning. • ManualEnable = FALSE • Automatic (State = 3), inactive (State = 0) or manual (State = 4) mode Fine tuning proceeds as follows when started in: • Automatic mode (State = 3) Start fine tuning in automatic mode if you wish to improve the existing PID parameters using controller tuning. PID_Comact will regulate using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start. • Inactive (State = 0) or manual (State = 4) mode If the requirements for pretuning are met, pretuning is started. The PID parameters established will be used for adjustment until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start. If pretuning is not possible, PID_Compact will change to "Inactive" mode. An attempt is made to reach the setpoint with a minimum or maximum output value if the process value for pretuning is already too near the setpoint or sPid_Calc.b_RunIn = TRUE. This can produce increased overshoot. The controller will change to "automatic mode" after successfully completed "fine tuning" and to "inactive" mode if "fine tuning" has not been successfully completed. The "Fine tuning" phase is indicated with Tag i_Event_TIR V1 (Page 4971).
3	Automatic mode In automatic mode, PID_Compact corrects the controlled system in accordance with the parameters specified. The controller changes to automatic mode if one the following conditions is fulfilled: • Pretuning successfully completed • Fine tuning successfully completed • Change of variable sRet.i_Mode to the value 3. After CPU startup or change from Stop to RUN mode, PID_Compact will start in the most recently active operating mode. To retain PID_Compact in "Inactive" mode, set sb_RunModeByStartup = FALSE.
4	Manual mode In manual mode, you specify a manual output value in the ManualValue parameter. This operating mode is enabled if sRet.i_Mode = 4, or at the rising edge on ManualEnable. If ManualEnable changes to TRUE, only State will change. sRet.i_Mode will retain its current value. PID_Compact will return to the previous operating mode upon a falling edge at ManualEnable. The change to automatic mode is bumpless.

See also

Output parameters of PID_Compact V1 (Page 4959)
Pretuning (Page 9585)
Fine tuning (Page 9587)
"Manual" mode (Page 9588)
Tag i_Event_SUT V1 (Page 4971)
Tag i_Event_TIR V1 (Page 4971)

Parameter Error V1

If several errors are pending simultaneously, the values of the error codes are displayed with binary addition. The display of error code 0003, for example, indicates that the errors 0001 and 0002 are pending simultaneously.

Error (DW#16#...)	Description
0000	There is no error.
0001	The "Input" parameter is outside the process value limits. • Input > sPid_Cmpt.r_Pv_Hlm or • Input < sPid_Cmpt.r_Pv_Llm You cannot move the actuator again until you eliminate the error.
0002	Invalid value at "Input_PER" parameter. Check whether an error is pending at the analog input.
0004	Error during fine tuning. Oscillation of the process value could not be maintained.
0008	Error at start of pretuning. The process value is too close to the setpoint. Start fine tuning.
0010	The setpoint was changed during tuning.
0020	Pretuning is not permitted in automatic mode or during fine tuning.
0080	Incorrect configuration of output value limits. Check whether the limits of the output value are configured correctly and match the control logic.
0100	Error during tuning resulted in invalid parameters.
0200	Invalid value at "Input" parameter: Value has an invalid number format.
0400	Calculation of output value failed. Check the PID parameters.
0800	Sampling time error: PID_Compact is not called within the sampling time of the cyclic interrupt OB.
1000	Invalid value at "Setpoint" parameter: Value has an invalid number format.

See also

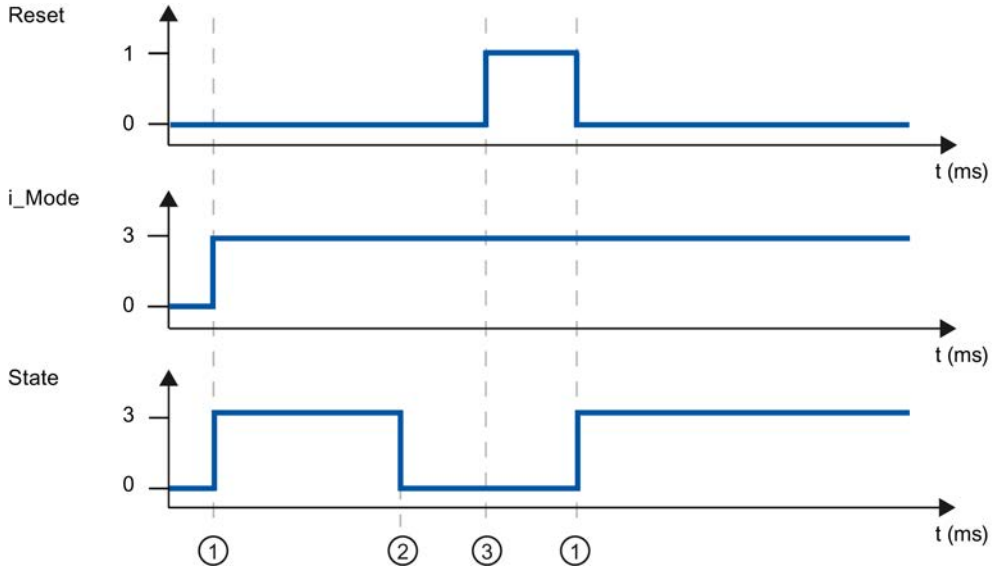
Output parameters of PID_Compact V1 (Page 4959)

Parameter Reset V1

The response to Reset = TRUE depends on the version of the PID_Compact instruction.

Reset response PID_Compact V.1.1 or higher

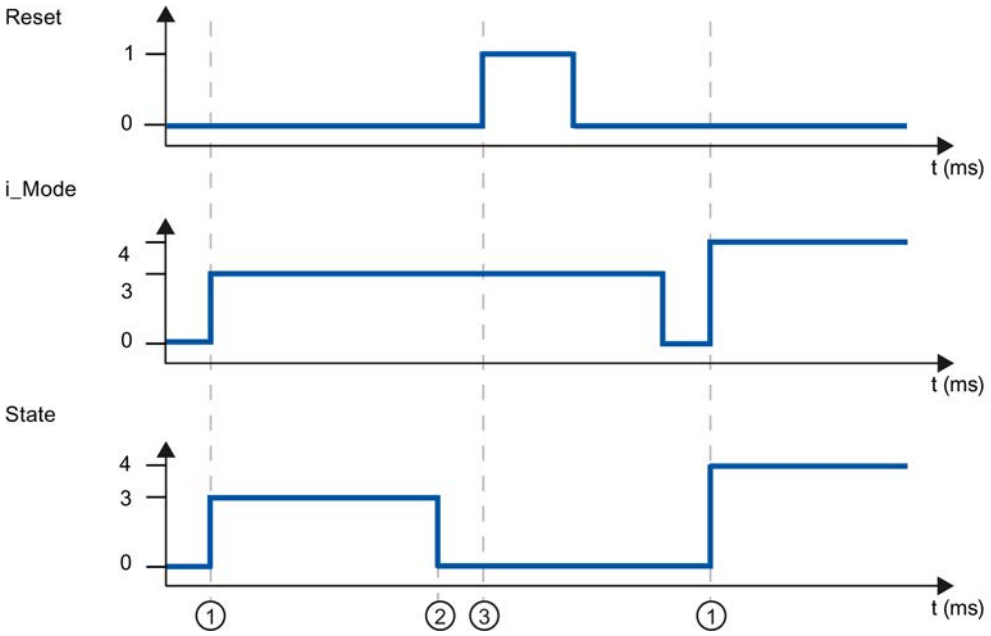
A rising edge at Reset resets the errors and warnings and clears the integral action. A falling edge at Reset triggers a change to the most recently active operating mode.



① Activation
② Error
③ Reset

Reset response PID_Compact V.1.0

A rising edge at Reset resets the errors and warnings and clears the integral action. The controller is not reactivated until the next edge at i_Mode.



① Activation
② Error
③ Reset

Tag sd_warning V1

If several warnings are pending, the values of variable sd_warning are displayed by means of binary addition. The display of warning 0003, for example, indicates that the warnings 0001 and 0002 are also pending.

sd_warning (DW#16#....)	Description
0000	No warning pending.
0001	The point of inflection was not found during pretuning.
0002	Oscillation increased during fine tuning.
0004	The setpoint was outside the set limits.
0008	Not all the necessary controlled system properties were defined for the selected method of calculation. The PID parameters were instead calculated using the "i_CtrlTypeTIR = 3" method.
0010	The operating mode could not be changed because ManualEnable = TRUE.
0020	The cycle time of the calling OB limits the sampling time of the PID algorithm. Improve results by using shorter OB cycle times.
0040	The process value exceeded one of its warning limits.

The following warnings are deleted as soon as the cause is dealt with:

- 0004
- 0020
- 0040

All other warnings are cleared with a rising edge at Reset.

Tag i_Event_SUT V1

i_Event_SUT	Name	Description
0	SUT_INIT	Initialize pretuning
100	SUT_STDABW	Calculate the standard deviation
200	SUT_GET_POI	Find the point of inflection
9900	SUT_IO	Pretuning successful
1	SUT_NIO	Pretuning not successful

See also

Static tags of PID_Compact V1 (Page 4960)

Parameters State and sRet.i_Mode V1 (Page 4965)

Tag i_Event_TIR V1

i_Event_TIR	Name	Description
-100	TIR_FIRST_SUT	Fine tuning is not possible. Pretuning will be executed first.
0	TIR_INIT	Initialize fine tuning
200	TIR_STDABW	Calculate the standard deviation
300	TIR_RUN_IN	Attempt to reach the setpoint
400	TIR_CTRLN	Attempt to reach the setpoint with the existing PID parameters (if pretuning has been successful)
500	TIR_OSZIL	Determine oscillation and calculate parameters
9900	TIR_IO	Fine tuning successful
1	TIR_NIO	Fine tuning not successful

See also

Static tags of PID_Compact V1 (Page 4960)

Parameters State and sRet.i_Mode V1 (Page 4965)

PID_3Step

New features of PID_3Step

PID_3Step V2.2

- Use with S7-1200
  As of PID_3Step V2.2, the instruction with V2 functionality can also be used on S7-1200 with firmware version 4.0 or higher.

PID_3Step V2.0

- Reaction to error
  The reaction to ActivateRecoverMode = TRUE has been completely overhauled.
  PID_3Step reacts in a more fault tolerant manner in the default setting.

NOTICE
Your system may be damaged. If you use the default setting, PID_3Step remains in automatic mode even if the process value limits are exceeded. This may damage your system. It is essential to configure how your controlled system reacts in the event of an error to protect your system from damage.

  You use the ErrorAck input parameter to acknowledge the errors and warnings without restarting the controller or clearing the integral action.
  Switching operating modes does not acknowledge errors that are no longer pending.
- Switching the operating mode
  You specify the operating mode at the Mode in/out parameter and use a rising edge at ModeActivate to start the operating mode. The Retain.Mode tag has been omitted.
  The transition time measurement can no longer be started with GetTransitTime.Start, but only with Mode = 6 and a rising edge at ModeActivate.
- Multi-instance capability
  You can call up PID_3Step as multi-instance DB. No technology object is created in this case and no parameter assignment interface or commissioning interface is available. You must assign parameters for PID_3Step directly in the multi-instance DB and commission it via a watch table.
- Startup characteristics
  The operating mode specified at the Mode parameter is also started on a falling edge at Reset and during a CPU cold restart, if RunModeByStartup = TRUE.
- ENO characteristics
  ENO is set depending on the operating mode.
  If State = 0, then ENO = FALSE.
  If State ≠ 0, then ENO = TRUE.

- **Manual mode**
  The Manual_UP and Manual_DN input parameters no longer function as edge-triggered parameters. Edge-triggered manual mode continues to be possible using the ManualUpInternal and ManualDnInternal tags.
  In "Manual mode without endstop signals" (Mode = 10), the endstop signals Actuator_H and Actuator_L are ignored even though they are activated.
- **Default value of PID parameters**
  The following default settings have been changed:
  - Proportional action weighting (PWeighting) from 0.0 to 1.0
  - Derivative action weighting (DWeighting) from 0.0 to 1.0
  - Coefficient for derivative delay (TdFiltRatio) from 0.0 to 0.2
- **Limiting of motor transition time**
  You configure the maximum percentage of the motor transition time that the actuator will travel in one direction in the Config.VirtualActuatorLimit tag.
- **Setpoint value specification during tuning**
  You configure the permitted fluctuation of the setpoint during tuning at the CancelTuningLevel tag.
- **Switching a disturbance variable on**
  You can switch a disturbance variable on at the Disturbance parameter.
- **Troubleshooting**
  If the endstop signals are not activated (ActuatorEndStopOn = FALSE), ScaledFeedback is determined without Actuator_H or Actuator_L.

PID_3Step V1.1

- **Manual mode on CPU startup**
  If ManualEnable = TRUE when the CPU starts, PID_3Step starts in manual mode. A rising edge at ManualEnable is not necessary.
- **Reaction to error**
  The ActivateRecoverMode tag is no longer effective in manual mode.
- **Troubleshooting**
  The Progress tag is reset following successful tuning or transition time measurement.

Compatibility with CPU and FW

The following table shows which version of PID_3Step can be used on which CPU.

CPU	FW	PID_3Step
S7-1200	V4.X	V2.2 V1.1
S7-1200	V3.X	V1.1 V1.0
S7-1200	V2.X	V1.1 V1.0

CPU	FW	PID_3Step
S7-1200	V1.X	-
S7-1500	V1.1	V2.2 V2.1 V2.0
S7-1500	V1.0	V2.0

PID_3Step V2

Description of PID_3Step V2

Description

You use the PID_3Step instruction to configure a PID controller with self tuning for valves or actuators with integrating behavior.

The following operating modes are possible:

- Inactive
- Pretuning
- Fine tuning
- Automatic mode
- Manual mode
- Approach substitute output value
- Transition time measurement
- Error monitoring
- Approach substitute output value with error monitoring
- Manual mode without endstop signals

For a more detailed description of the operating modes, see the State parameter.

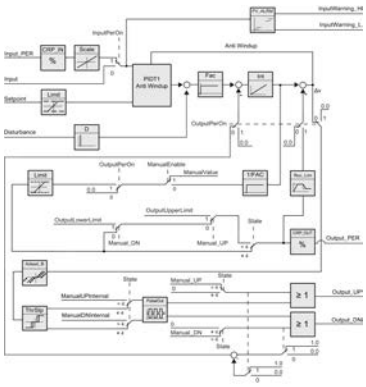
PID algorithm

PID_3Step is a PIDT1 controller with anti-windup and weighting of the proportional and derivative actions. The PID algorithm operates according to the following equation:

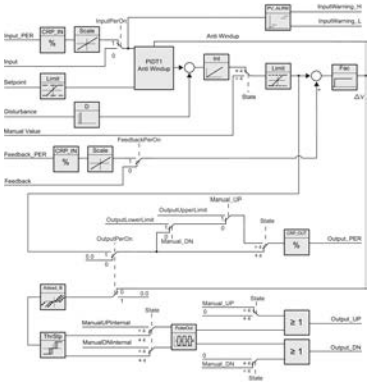
$$\Delta y = K_p \cdot s \cdot \left[(b \cdot w - x) + \frac{1}{T_I \cdot s}(w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1}(c \cdot w - x) \right]$$

Symbol	Description
Δy	Output value of the PID algorithm
K_p	Proportional gain
s	Laplace operator
b	Proportional action weighting
w	Setpoint
x	Process value
T_I	Integral action time
T_D	Derivative action time
a	Derivative delay coefficient (derivative delay T1 = a × T_D)
c	Derivative action weighting

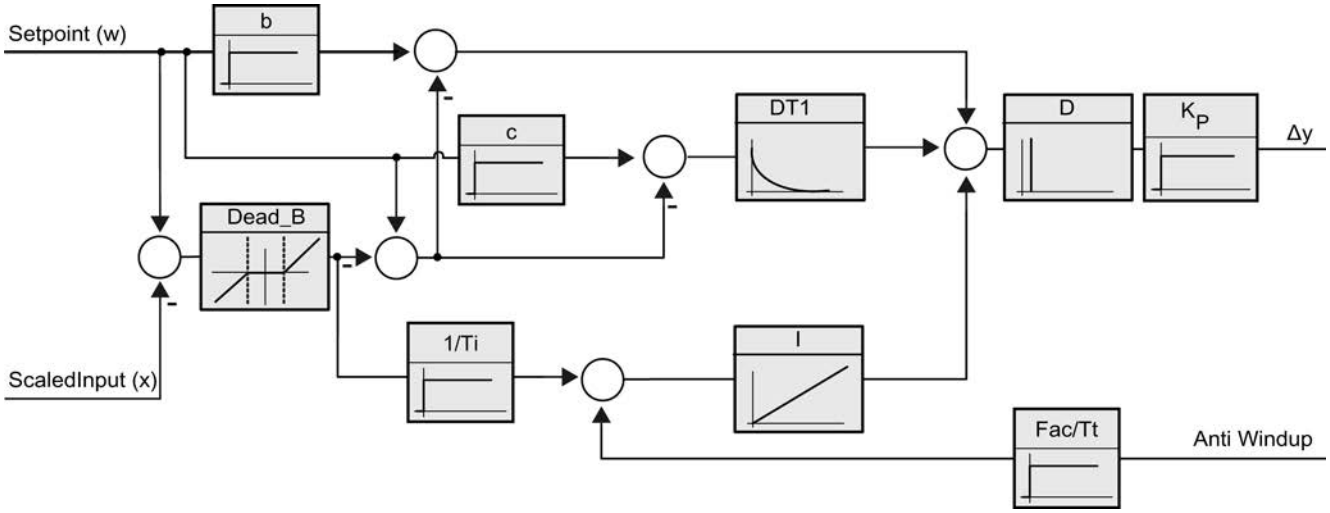
Block diagram without position feedback



Block diagram with position feedback



Block diagram of PIDT1 with anti-windup



Call

PID_3Step is called in the constant time scale of a cycle interrupt OB.

If you call PID_3Step as a multi-instance DB, no technology object is created. No parameter assignment interface or commissioning interface is available. You must assign parameters for PID_3Step directly in the multi-instance DB and commission it via a watch table.

Download to device

The actual values of retentive tags are only updated when you download PID_3Step completely.

Downloading technology objects to device (Page 9536)

Startup

When the CPU starts up, PID_3Step starts in the operating mode that is saved in the Mode in/out parameter. To leave PID_3Step in "Inactive" mode, set RunModeByStartup = FALSE.

Reaction to error

In automatic mode and during commissioning, the reaction to error depends on the ErrorBehaviour and ActivateRecoverMode tags. In manual mode, the reaction is independent of ErrorBehaviour and ActivateRecoverMode. If ActivateRecoverMode = TRUE, the reaction additionally depends on the error that occurred.

ErrorBehaviour	ActivateRecoverMode	Configuration editor > actuator setting > Set Output to	Reaction
FALSE	FALSE	Current output value	Switch to "Inactive" mode (State = 0) The actuator remains in the current position.
FALSE	TRUE	Current output value while error is pending	Switch to "Error monitoring" mode (State = 7) The actuator remains in the current position while the error is pending.
TRUE	FALSE	Substitute output value	Switch to "Approach substitute output value" mode (State = 5) The actuator moves to the configured substitute output value. Switch to "Inactive" mode (State = 0) The actuator remains in the current position.
TRUE	TRUE	Substitute output value while error is pending	Switch to "Approach substitute output value with error monitoring" mode (State = 8) The actuator moves to the configured substitute output value. Switch to "Error monitoring" mode (State = 7)

In manual mode, PID_3Step uses ManualValue as output value, unless the following errors occur:

- 2000h: Invalid value at Feedback_PER parameter.
- 4000h: Invalid value at Feedback parameter.
- 8000h: Error during digital position feedback.

You can only change the position of the actuator with Manual_UP and Manual_DN, not with ManualValue.

The Error parameter indicates whether an error has occurred in this cycle. The ErrorBits parameter shows which errors have occurred. ErrorBits is reset by a rising edge at Reset or ErrorAck.

See also

Parameters State and Mode V2 (Page 4993)

Parameter ErrorBits V2 (Page 4998)

Configuring PID_3Step V2 (Page 9590)

Mode of operation of PID_3Step V2

Monitoring process value limits

You specify the high limit and low limit of the process value in the Config.InputUpperLimit and Config.InputLowerLimit variables. If the process value is outside these limits, an error occurs (ErrorBits = 0001h).

You specify a high and low warning limit of the process value in the Config.InputUpperWarning and Config.InputLowerWarning variables. If the process value is outside these warning limits, a warning occurs (Warning = 0040h), and the InputWarning_H or InputWarning_L output parameter changes to TRUE.

Limiting the setpoint

You specify a high limit and low limit of the setpoint in the Config.SetpointUpperLimit and Config.SetpointLowerLimit variables. PID_3Step automatically limits the setpoint to the process value limits. You can limit the setpoint to a smaller range. PID_3Step checks whether this range falls within the process value limits. If the setpoint is outside these limits, the high or low limit is used as the setpoint, and output parameter SetpointLimit_H or SetpointLimit_L is set to TRUE.

The setpoint is limited in all operating modes.

Limiting the output value

You specify a high limit and low limit of the output value in the Config.OutputUpperLimit and Config.OutputLowerLimit variables. The output value limits must be within "Low endstop" and "High endstop".

- High endstop: Config.FeedbackScaling.UpperPointOut
- Low endstop: Config.FeedbackScaling.LowerPointOut

Rule:

UpperPointOut ≥ OutputUpperLimit > OutputLowerLimit ≥ LowerPointOut

The valid values for "High endstop" and "Low endstop" depend upon:

- FeedbackOn
- FeedbackPerOn
- OutputPerOn

OutputPerOn	FeedbackOn	FeedbackPerOn	LowerPointOut	UpperPointOut
FALSE	FALSE	FALSE	Cannot be set (0.0%)	Cannot be set (100.0%)
FALSE	TRUE	FALSE	-100.0% or 0.0%	0.0% or +100.0%
FALSE	TRUE	TRUE	-100.0% or 0.0%	0.0% or +100.0%
TRUE	FALSE	FALSE	Cannot be set (0.0%)	Cannot be set (100.0%)
TRUE	TRUE	FALSE	-100.0% or 0.0%	0.0% or +100.0%
TRUE	TRUE	TRUE	-100.0% or 0.0%	0.0% or +100.0%

If OutputPerOn = FALSE and FeedbackOn = FALSE, you cannot limit the output value. Output_UP and Output_DN are then reset at Actuator_H = TRUE or Actuator_L = TRUE. If endstop signals are also not present, Output_UP and Output_DN are reset after a travel time of Config.VirtualActuatorLimit × Retain.TransitTime/100.

The output value is 27648 at 100% and -27648 at -100%. PID_3Step must be able to close the valve completely.

Substitute output value

If an error has occurred, PID_3Step can output a substitute output value and move the actuator to a safe position that is specified in the SavePosition tag. The substitute output value must be within the output value limits.

Monitoring signal validity

The values of the following parameters are monitored for validity when used:

- Setpoint
- Input
- Input_PER
- Input_PER
- Feedback
- Feedback_PER
- Disturbance
- ManualValue
- SavePosition
- Output_PER

Monitoring the PID_3Step sampling time

Ideally, the sampling time is equivalent to the cycle time of the calling OB. The PID_3Step instruction measures the time interval between two calls. This is the current sampling time. On every switchover of operating mode and during the initial startup, the mean value is formed from the first 10 sampling times. Too great a difference between the current sampling time and this mean value triggers an error (ErrorBits = 0800h).

The error occurs during tuning if:

- New mean value >= 1.1 x old mean value
- New mean value <= 0.9 x old mean value

The error occurs in automatic mode if:

- New mean value >= 1.5 x old mean value
- New mean value <= 0.5 x old mean value

If you deactivate the sampling time monitoring (CycleTime.EnMonitoring = FALSE), you can also call PID_3Step in OB1. You must then accept a lower control quality due to the deviating sampling time.

Sampling time of the PID algorithm

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is calculated during tuning and rounded to a multiple of the cycle time. All other functions of PID_3Step are executed at every call.

Measuring the motor transition time

The motor transition time is the time in seconds the motor requires to move the actuator from the closed to the opened state. The actuator is moved in one direction for a maximum time of Config.VirtualActuatorLimit × Retain.TransitTime/100. PID_3Step requires the motor transition time to be as accurate as possible for good controller results. The data in the actuator documentation contains average values for this type of actuator. The value for the specific actuator used may differ. You can measure the motor transition time during commissioning. The output value limits are not taken into consideration during the motor transition time measurement. The actuator can travel to the high or the low endstop.

Control logic

An increase of the output value is generally intended to cause an increase in the process value. This is referred to as a normal control logic. For cooling and discharge control systems, it may be necessary to invert the control logic. PID_3Step does not work with negative proportional gain. If InvertControl = TRUE, an increasing control deviation causes a reduction in the output value. The control logic is also taken into account during pretuning and fine tuning.

See also

Configuring PID_3Step V1 (Page 9607)

Changing the PID_3Step V2 interface

The following table shows what has changed in the PID_3Step instruction interface.

PID_3Step V1	PID_3Step V2	Change
Input_PER	Input_PER	Data type from Word to Int
Feedback_PER	Feedback_PER	Data type from Word to Int
	Disturbance	New
Manual_UP	Manual_UP	Function
Manual_DN	Manual_DN	Function
	ErrorAck	New
	ModeActivate	New
Output_PER	Output_PER	Data type from Word to Int
	ManualUPInternal	New
	ManualDNInternal	New
	CancelTuningLevel	New
	VirtualActuatorLImit	New
Config.Loadbackup	Loadbackup	Renamed
Config.TransitTime	Retain.TransitTime	Renamed and retentivity added
GetTransitTime.Start		Replaced by Mode and ModeActivate
SUT.CalculateSUTParams	SUT.CalculateParams	Renamed
SUT.TuneRuleSUT	SUT.TuneRule	Renamed
TIR.CalculateTIRParams	TIR.CalculateParams	Renamed
TIR.TuneRuleTIR	TIR.TuneRule	Renamed
Retain.Mode	Mode	Function Declaration of static for in-out parameters

Input parameters of PID_3Step V2

Table 9-88

Parameter	Data type	Default	Description
Setpoint	REAL	0.0	Setpoint of the PID controller in automatic mode
Input	REAL	0.0	A tag of the user program is used as the source of the process value. If you are using the Input parameter, then Config.InputPerOn = FALSE must be set.
Input_PER	INT	0	An analog input is used as the source of the process value. If you are using the Input_PER parameter, then Config.InputPerOn = TRUE must be set.
Actuator_H	BOOL	FALSE	Digital position feedback of the valve for the high endstop If Actuator_H = TRUE, the valve is at the high endstop and is no longer moved towards this direction.
Actuator_L	BOOL	FALSE	Digital position feedback of the valve for the low endstop If Actuator_L = TRUE, the valve is at the low endstop and is no longer moved towards this direction.

Parameter	Data type	Default	Description
Feedback	REAL	0.0	Position feedback of the valve If you are using the Feedback parameter, then Config.FeedbackPerOn = FALSE must be set.
Feedback_PER	INT	0	Analog position feedback of a valve If you are using the Feedback_PER parameter, then Config.FeedbackPerOn = TRUE must be set. Feedback_PER is scaled based on the tags: • Config.FeedbackScaling.LowerPointIn • Config.FeedbackScaling.UpperPointIn • Config.FeedbackScaling.LowerPointOut • Config.FeedbackScaling.UpperPointOut
Disturbance	REAL	0.0	Disturbance tag or precontrol value
ManualEnable	BOOL	FALSE	• A FALSE -> TRUE edge activates "manual mode", while State = 4, Mode remain unchanged. As long as ManualEnable = TRUE, you cannot change the operating mode via a rising edge at ModeActivate or use the commissioning dialog. • A TRUE -> FALSE edge activates the operating mode that is specified by Mode. We recommend that you change the operating mode using ModeActivate only.
ManualValue	REAL	0.0	In manual mode, the absolute position of the valve is specified. ManualValue is only evaluated if you are using Output_PER, or if position feedback is available.
Manual_UP	BOOL	FALSE	• Manual_UP = TRUE The valve is opened even if you are using Output_PER or a position feedback. The valve is no longer moved if the high endstop has been reached. See also Config.VirtualActuatorLimit • Manual_UP = FALSE If you are using Output_PER or a position feedback, the valve is moved to ManualValue. Otherwise, the valve is no longer moved. If Manual_UP and Manual_DN are set to TRUE simultaneously, the valve is not moved.
Manual_DN	BOOL	FALSE	• Manual_DN = TRUE The valve is closed even if you are using Output_PER or a position feedback. The valve is no longer moved if the low endstop has been reached. See also Config.VirtualActuatorLimit • Manual_DN = FALSE If you are using Output_PER or a position feedback, the valve is moved to ManualValue. Otherwise, the valve is no longer moved.
ErrorAck	BOOL	FALSE	• FALSE -> TRUE edge ErrorBits and Warning are reset.

Parameter	Data type	Default	Description
Reset	BOOL	FALSE	Restarts the controller. • FALSE -> TRUE edge – Switch to "Inactive" mode – ErrorBits and Warning are reset. – Integral action is cleared (PID parameters are retained) • As long as Reset = TRUE, PID_3Step remains in "Inactive" mode (State = 0). • TRUE -> FALSE edge PID_3Step switches to the operating mode that is saved in the Mode parameter.
ModeActivate	BOOL	FALSE	• FALSE -> TRUE edge PID_3Step switches to the operating mode that is saved in the Mode parameter.

Output parameters of PID_3Step V2

Table 9-89

Parameter	Data type	Default	Description
ScaledInput	REAL	0.0	Scaled process value
ScaledFeedback	REAL	0.0	Scaled position feedback For an actuator without position feedback, the position of the actuator indicated by ScaledFeedback is very imprecise. ScaledFeedback may only be used for rough estimation of the current position in this case.
Output_UP	BOOL	FALSE	Digital output value for opening the valve If Config.OutputPerOn = FALSE, the Output_UP parameter is used.
Output_DN	BOOL	FALSE	Digital output value for closing the valve If Config.OutputPerOn = FALSE, the Output_DN parameter is used.
Output_PER	INT	0	Analog output value If Config.OutputPerOn = TRUE, Output_PER is used.
SetpointLimit_H	BOOL	FALSE	If SetpointLimit_H = TRUE, the absolute setpoint high limit is reached (Setpoint ≥ Config.SetpointUpperLimit). The setpoint is limited to Config.SetpointUpperLimit .
SetpointLimit_L	BOOL	FALSE	If SetpointLimit_L = TRUE, the absolute setpoint low limit has been reached (Setpoint ≤ Config.SetpointLowerLimit). The setpoint is limited to Config.SetpointLowerLimit .
InputWarning_H	BOOL	FALSE	If InputWarning_H = TRUE, the process value has reached or exceeded the warning high limit.
InputWarning_L	BOOL	FALSE	If InputWarning_L = TRUE, the process value has reached or fallen below the warning low limit.

Parameter	Data type	Default	Description
State	INT	0	The State parameter (Page 4993) shows the current operating mode of the PID controller. You can change the operating mode using the input parameter Mode and a rising edge at ModeActivate. • State = 0: Inactive • State = 1: Pretuning • State = 2: Fine tuning • State = 3: Automatic mode • State = 4: Manual mode • State = 5: Approach substitute output value • State = 6: Transition time measurement • State = 7: Error monitoring • State = 8: Approach substitute output value with error monitoring • State = 10: Manual mode without end stop signals
Error	BOOL	FALSE	If Error = TRUE, at least one error message is pending in this cycle.
ErrorBits	DWORD	DW#16#0	The ErrorBits parameter (Page 4998) shows which error messages are pending. ErrorBits is retentive and is reset upon a rising edge at Reset or ErrorAck.

See also

Parameters State and Mode V2 (Page 4993)

Parameter ErrorBits V2 (Page 4998)

In-out parameters of PID_3Step V2

Table 9-90

Parameter	Data type	Default	Description
Mode	INT	4	At the Mode parameter, you specify the operating mode to which PID_3Step is to switch. Options are: • Mode = 0: Inactive • Mode = 1: Pretuning • Mode = 2: Fine tuning • Mode = 3: Automatic mode • Mode = 4: Manual mode • Mode = 6: Transition time measurement • Mode = 10: Manual mode without endstop signals The operating mode is activated by: • Rising edge at ModeActivate • Falling edge at Reset • Falling edge at ManualEnable • Cold restart of CPU if RunModeByStartup = TRUE Mode is retentive. A detailed description of the operating modes can be found in Parameters State and Mode V2 (Page 4993).

Static tags of PID_3Step V2

You must not change tags that are not listed. These are used for internal purposes only.

Tag	Data type	Default	Description
ManualUpInternal	BOOL	FALSE	In manual mode, each rising edge opens the valve by 5% of the total control range or for the duration of the minimum motor transition time. ManualUpInternal is only evaluated if you are not using Output_PER or a position feedback. This tag is used in the commissioning dialog.
ManualDnInternal	BOOL	FALSE	In manual mode, every rising edge closes the valve by 5% of the total control range or for the duration of the minimum motor transition time. ManualDnInternal is only evaluated if you are not using Output_PER or position feedback. This tag is used in the commissioning dialog.
ActivateRecoverMode	BOOL	TRUE	The ActivateRecoverMode V2 (Page 5001) tag determines the reaction to error.
RunModeByStartup	BOOL	TRUE	Activate operating mode at Mode parameter after CPU restart If RunModeByStartup = TRUE, PID_3Step starts in the operating mode saved in the Mode parameter after CPU startup. If RunModeByStartup = FALSE, PID_3Step remains in "Inactive" mode after CPU startup.
LoadBackUp	BOOL	FALSE	If LoadBackUp = TRUE, the last set of PID parameters is reloaded. The set was saved prior to the last tuning. LoadBackUp is automatically set back to FALSE.

Tag	Data type	Default	Description
PhysicalUnit	INT	0	Unit of measurement of the process value and setpoint, e.g., ºC, or ºF.
PhysicalQuantity	INT	0	Physical quantity of the process value and setpoint, e.g., temperature
ErrorBehaviour	BOOL	FALSE	If ErrorBehaviour = FALSE and an error has occurred, the valve stays at its current position and the controller switches directly to "Inactive" or "Error monitoring" mode. If ErrorBehaviour = TRUE and an error occurs, the actuator moves to the substitute output value and only then switches to "Inactive" or "Error monitoring" mode. If the following errors occur, you can no longer move the valve to a configured substitute output value. • 2000h: Invalid value at Feedback_PER parameter. • 4000h: Invalid value at Feedback parameter. • 8000h: Error during digital position feedback. • 20000h: Invalid value at SavePosition tag.
Warning	DWORD	DW#16#0	The Warning tag (Page 4993) shows the warnings since Reset = TRUE or ErrorAck =TRUE. Warning is retentive. Cyclic warnings (for example, process value warning) are shown until the cause of the warning is removed. They are automatically deleted once their cause has gone. Non-cyclic warnings (for example, point of inflection not found) remain and are deleted like errors.
SavePosition	REAL	0.0	Substitute output value If ErrorBehaviour = TRUE, the actuator is moved to a position that is safe for the plant when an error occurs. As soon as the substitute output value has been reached, PID_3Step switches the operating mode according to ActivateRecoverMode.
CurrentSetpoint	REAL	0.0	Currently active setpoint. This value is frozen at the start of tuning.
CancelTuningLevel	REAL	10.0	Permissible fluctuation of setpoint during tuning. Tuning is not canceled until: • Setpoint > CurrentSetpoint + CancelTuningLevel or • Setpoint < CurrentSetpoint - CancelTuningLevel
Progress	REAL	0.0	Progress of tuning as a percentage (0.0 - 100.0)
Config.InputPerOn	BOOL	TRUE	If InputPerOn = TRUE, the Input_PER parameter is used. If InputPerOn = FALSE, the Input parameter is used.
Config.OutputPerOn	BOOL	FALSE	If OutputPerOn = TRUE, the Output_PER parameter is used. If OutputPerOn = FALSE, the Ouput_UP and Output_DN parameters are used.
Config.InvertControl	BOOL	FALSE	Invert control logic If InvertControl = TRUE, an increasing control deviation causes a reduction in the output value.
Config.FeedbackOn	BOOL	FALSE	If FeedbackOn = FALSE, a position feedback is simulated. Position feedback is generally activated when FeedbackOn = TRUE.

Tag	Data type	Default	Description
Config.FeedbackPerOn	BOOL	FALSE	FeedbackPerOn is only effective when FeedbackOn = TRUE. If FeedbackPerOn = TRUE, the analog input is used for the position feedback (Feedback_PER parameter). If FeedbackPerOn = FALSE, the Feedback parameter is used for the position feedback.
Config.ActuatorEndStopOn	BOOL	FALSE	If ActuatorEndStopOn = TRUE, the digital position feedback Actuator_L and Actuator_H are taken into consideration.
Config.InputUpperLimit	REAL	120.0	High limit of the process value Input and Input_PER are monitored to ensure adherence to this limit. At the I/O input, the process value can be a maximum of 18% higher than the standard range (overrange). An error is no longer signaled due to a violation of the "Process value high limit". Only a wire-break and a short-circuit are recognized and PID_3Step reacts according to the configured reaction to error. InputUpperLimit > InputLowerLimit
Config.InputLowerLimit	REAL	0.0	Low limit of the process value InputLowerLimit < InputUpperLimit
Config.InputUpperWarning	REAL	+3.40282 2e+38	Warning high limit of the process value If you set InputUpperWarning outside the process value limits, the configured absolute process value high limit is used as the warning high limit. If you configure InputUpperWarning within the process value limits, this value is used as the warning high limit. InputUpperWarning > InputLowerWarning InputUpperWarning ≤ InputUpperLimit
Config.InputLowerWarning	REAL	-3.40282 2e+38	Warning low limit of the process value If you set InputLowerWarning outside the process value limits, the configured absolute process value low limit is used as the warning low limit. If you configure InputLowerWarning within the process value limits, this value is used as the warning low limit. InputLowerWarning < InputUpperWarning InputLowerWarning ≥ InputLowerLimit
Config.OutputUpperLimit	REAL	100.0	High limit of output value For details, see OutputLowerLimit
Config.OutputLowerLimit	REAL	0.0	Low limit of output value If OutputPerOn = TRUE or FeedbackOn = TRUE, the range of values from -100% to +100%, including zero, is valid. At -100%, Output = -27648; at +100% Output = 27648 If OutputPerOn = FALSE, the range of values from 0% to 100% is valid. The valve is completely closed at 0% and completely opened at 100%.
Config.SetpointUpperLimit	REAL	+3.40282 2e+38	High limit of setpoint If you set SetpointUpperLimit outside the process value limits, the configured absolute process value high limit is preassigned as the setpoint high limit. If you configure SetpointUpperLimit within the process value limits, this value is used as the setpoint high limit.

Tag	Data type	Default	Description
Config.SetpointLowerLimit	REAL	-3.402822e+38	Low limit of the setpoint If you set SetpointLowerLimit outside the process value limits, the configured absolute process value low limit is preassigned as the setpoint low limit. If you set SetpointLowerLimit within the process value limits, this value is used as the setpoint low limit.
Config.MinimumOnTime	REAL	0.0	Minimum ON time Minimum time in seconds for which the servo drive must be switched on.
Config.MinimumOffTime	REAL	0.0	Minimum OFF time Minimum time in seconds for which the servo drive must be switched off.
Config.VirtualActuatorLimit	REAL	150.0	If all the following conditions have been satisfied, the actuator is moved in one direction for the maximum period of VirtualActuatorLimit × Retain.TransitTime/100 and the warning 2000h is output: • Config.OutputPerOn = FALSE • Config.ActuatorEndStopOn = FALSE • Config.FeedbackOn = FALSE If Config.OutputPerOn = FALSE and Config.ActuatorEndStopOn = TRUE or Config.FeedbackOn = TRUE, only the warning 2000h is output. If Config.OutputPerOn = TRUE, VirtualActuatorLimit is not taken into consideration.
Config.InputScaling.UpperPointIn	REAL	27648.0	Scaling Input_PER high Input_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the InputScaling structure.
Config.InputScaling.LowerPointIn	REAL	0.0	Scaling Input_PER low Input_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the InputScaling structure.
Config.InputScaling.UpperPointOut	REAL	100.0	Scaled high process value Input_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the InputScaling structure.
Config.InputScaling.LowerPointOut	REAL	0.0	Scaled low process value Input_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the InputScaling structure.
Config.FeedbackScaling.UpperPointIn	REAL	27648.0	Scaling Feedback_PER high Feedback_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the FeedbackScaling structure.
Config.FeedbackScaling.LowerPointIn	REAL	0.0	Scaling Feedback_PER low Feedback_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the FeedbackScaling structure.

Tag	Data type	Default	Description
Config.FeedbackScaling.UpperPointOut	REAL	100.0	High endstop Feedback_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the FeedbackScaling structure.
Config.FeedbackScaling.LowerPointOut	REAL	0.0	Low endstop Feedback_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the FeedbackScaling structure.
GetTransitTime.InvertDirection	BOOL	FALSE	If InvertDirection = FALSE, the valve is fully opened, closed, and then reopened in order to determine the valve transition time. If InvertDirection = TRUE, the valve is fully closed, opened, and then closed again.
GetTransitTime.SelectFeedback	BOOL	FALSE	If SelectFeedback = TRUE, then Feedback_PER, or Feedback is taken into consideration in the transition time measurement. If SelectFeedback = FALSE, then Actuator_H and Actuator_L are taken into consideration in the transition time measurement.
GetTransitTime.State	INT	0	Current phase of the transition time measurement • State = 0: Inactive • State = 1: Open valve completely • State = 2: Close valve completely • State = 3: Move valve to target position (NewOutput) • State = 4: Transition time measurement successfully completed • State = 5: Transition time measurement canceled
GetTransitTime.NewOutput	REAL	0.0	Target position for transition time measurement with position feedback The target position must be between "High endstop" and "Low endstop". The difference between NewOutput and ScaledFeedback must be at least 50% of the permissible control range.
CycleTime.StartEstimation	BOOL	TRUE	If StartEstimation = TRUE, the measurement of the PID_3Step sampling time is started. CycleTime.StartEstimation = FALSE once measurement is complete.
CycleTime.EnEstimation	BOOL	TRUE	If EnEstimation = TRUE, the PID_3Step sampling time is calculated. If CycleTime.EnEstimation = FALSE, the PID_3Step sampling time is not calculated and you need to correct the configuration of CycleTime.Value manually.
CycleTime.EnMonitoring	BOOL	TRUE	If EnMonitoring = TRUE, the PID_3Step sampling time is monitored. If it is not possible to execute PID_3Step within the sampling time, the error 0800h is output and the operating mode is switched. ActivateRecoverMode and ErrorBehaviour determine which operating mode is switched to. If EnMonitoring = FALSE, the PID_3Step sampling time is not monitored, the error 0800h is not output, and the operating mode is not switched.
CycleTime.Value	REAL	0.1	PID_3Step sampling time in seconds CycleTime.Value is determined automatically and is usually equivalent to the cycle time of the calling OB.

Tag	Data type	Default	Description
CtrlParamsBackUp.SetByUser	BOOL	FALSE	Saved value of Retain.CtrlParams.SetByUser You can reload values from the CtrlParamsBackUp structure with LoadBackUp = TRUE.
CtrlParamsBackUp.Gain	REAL	1.0	Saved proportional gain
CtrlParamsBackUp.Ti	REAL	20.0	Saved integral action time in seconds
CtrlParamsBackUp.Td	REAL	0.0	Saved derivative action time in seconds
CtrlParamsBackUp.TdFiltRatio	REAL	0.2	Saved derivative delay coefficient
CtrlParamsBackUp.PWeighting	REAL	1.0	Saved proportional action weighting
CtrlParamsBackUp.DWeighting	REAL	1.0	Saved derivative action weighting
CtrlParamsBackUp.Cycle	REAL	1.0	Saved sampling time of PID algorithm in seconds
CtrlParamsBackUp.InputDeadBand	REAL	0.0	Saved deadband width of the control deviation
PIDSelfTune.SUT.CalculateParams	BOOL	FALSE	The properties of the controlled system are saved during tuning. If CalculateParams = TRUE, the PID parameters are recalculated on the basis of these properties. The PID parameters are calculated using the method set in TuneRule. CalculateParams is set to FALSE following calculation.
PIDSelfTune.SUT.TuneRule	INT	1	Methods used to calculate parameters during pretuning: • SUT.TuneRule = 0: PID rapid I • SUT.TuneRule = 1: PID slow I • SUT.TuneRule = 2: Chien, Hrones and Reswick PID • SUT.TuneRule = 3: Chien, Hrones, Reswick PI • SUT.TuneRule = 4: PID rapid II • SUT.TuneRule = 5: PID slow II
PIDSelfTune.SUT.State	INT	0	The SUT.State tag indicates the current phase of pretuning: • State = 0: Initialize pretuning • State = 50: Determine start position without position feedback • State = 100: Calculate standard deviation • State = 200: Determine point of inflection • State = 300: Determine rise time • State = 9900: Pretuning successful • State = 1: Pretuning not successful

Tag	Data type	Default	Description
PIDSelfTune.TIR.RunIn	BOOL	FALSE	With the RunIn tag, you can specify that fine tuning can also be performed without pretuning. • RunIn = FALSE Pretuning is started when fine tuning is started from inactive or manual mode. If fine tuning is started from automatic mode, the system uses the existing PID parameters to control to the setpoint. Only then will fine tuning start. If pretuning is not possible, PID_3Step switches to the mode from which tuning was started. • RunIn = TRUE The pretuning is skipped. PID_3Step attempts to reach the setpoint with the minimum or maximum output value. This can produce increased overshoot. Only then will fine tuning start. RunIn is set to FALSE after fine tuning.
PIDSelfTune.TIR.CalculateParams	BOOL	FALSE	The properties of the controlled system are saved during tuning. If CalculateParams = TRUE, the PID parameters are recalculated on the basis of these properties. The PID parameters are calculated using the method set in TuneRule. CalculateParams is set to FALSE following calculation.
PIDSelfTune.TIR.TuneRule	INT	0	Methods used to calculate parameters during fine tuning: • TIR.TuneRule = 0: PID automatic • TIR.TuneRule = 1: PID rapid • TIR.TuneRule = 2: PID slow • TIR.TuneRule = 3: Ziegler-Nichols PID • TIR.TuneRule = 4: Ziegler-Nichols PI • TIR.TuneRule = 5: Ziegler-Nichols P
PIDSelfTune.TIR.State	INT	0	The TIR.State tag indicates the current phase of fine tuning: • State = -100 Fine tuning is not possible. Pretuning will be performed first. • State = 0: Initialize fine tuning • State = 200: Calculate standard deviation • State = 300: Attempt to reach the setpoint with the maximum or minimum output value • State = 400: Attempt to reach the setpoint with existing PID parameters (if pretuning was successful) • State = 500: Determine oscillation and calculate parameters • State = 9900: Fine tuning successful • State = 1: Fine tuning not successful
Retain.TransitTime	REAL	30.0	Motor transition time in seconds Time in seconds the actuating drive requires to move the valve from the closed to the opened state. TransitTime is retentive.

Tag	Data type	Default	Description
Retain.CtrlParams.SetByUser	BOOL	FALSE	If SetByUser = FALSE, the PID parameters are determined automatically and PID_3Step operates with a deadband at the output value. The deadband width is calculated during tuning on the basis of the standard deviation of the output value and saved in Retain.CtrlParams.OutputDeadBand. If SetByUser = TRUE, the PID parameters are entered manually and PID_3 Step operates without a deadband at the output value. Retain.CtrlParams.OutputDeadBand = 0.0 SetByUser is retentive.
Retain.CtrlParams.Gain	REAL	1.0	Active proportional gain To invert the control logic, use the Config.InvertControl tag. Negative values at Gain also invert the control logic. We recommend you use only InvertControl to set the control logic. The control logic is also inverted if InvertControl = TRUE and Gain < 0.0. Gain is retentive.
Retain.CtrlParams.Ti	REAL	20.0	• Ti > 0.0: Active integral action time in seconds • Ti = 0.0: Integral action is deactivated Ti is retentive.
Retain.CtrlParams.Td	REAL	0.0	• Td > 0.0: Active derivative action time in seconds • Td = 0.0: Derivative action is deactivated Td is retentive.
Retain.CtrlParams.TdFiltRatio	REAL	0.2	Active derivative delay coefficient The derivative delay coefficient delays the effect of the derivative action. Derivative delay = derivative action time × derivative delay coefficient • 0.0: Derivative action is effective for one cycle only and therefore almost not effective. • 0.5: This value has proved useful in practice for controlled systems with one dominant time constant. • > 1.0: The greater the coefficient, the longer the effect of the derivative action is delayed. TdFiltRatio is retentive.
Retain.CtrlParams.PWeighting	REAL	1.0	Active proportional action weighting The proportional action may weaken with changes to the setpoint. Values from 0.0 to 1.0 are applicable. • 1.0: Proportional action for setpoint change is fully effective • 0.0: Proportional action for setpoint change is not effective The proportional action is always fully effective when the process value is changed. PWeighting is retentive.

Tag	Data type	Default	Description
Retain.CtrlParams.DWeighting	REAL	1.0	Active derivative action weighting The derivative action may weaken with changes to the setpoint. Values from 0.0 to 1.0 are applicable. • 1.0: Derivative action is fully effective upon setpoint change • 0.0: Derivative action is not effective upon setpoint change The derivative action is always fully effective when the process value is changed. DWeighting is retentive.
Retain.CtrlParams.Cycle	REAL	1.0	Active sampling time of PID algorithm in seconds, rounded to an integer multiple of the cycle time of the calling OB. Cycle is retentive.
Retain.CtrlParams.InputDeadBand	REAL	0.0	Deadband width of the control deviation InputDeadBand is retentive.

Note

Change the tags listed in this table in "Inactive" mode to prevent malfunction of the PID controller.

See also

Parameters State and Mode V2 (Page 4993)

Tag ActivateRecoverMode V2 (Page 5001)

Downloading technology objects to device (Page 9536)

Parameters State and Mode V2

Correlation of the parameters

The State parameter shows the current operating mode of the PID controller. You cannot change the State parameter.

With a rising edge at ModeActivate, PID_3Step switches to the operating mode saved in the Mode in-out parameter.

When the CPU is switched on or switches from Stop to RUN mode, PID_3Step starts in the operating mode that is saved in the Mode parameter. To leave PID_3Step in "Inactive" mode, set RunModeByStartup = FALSE.

Meaning of values

State	Description of operating mode
0	Inactive The controller is switched off and no longer changes the valve position.
1	Pretuning The pretuning determines the process response to a pulse of the output value and searches for the point of inflection. The PID parameters are calculated from the maximum rate of rise and dead time of the controlled system. You obtain the best PID parameters when you perform pretuning and fine tuning. Pretuning requirements: • The motor transition time has been configured or measured. • Inactive (State = 0), manual mode (State = 4), or automatic mode (State = 3) • ManualEnable = FALSE • Reset = FALSE • The setpoint and the process value lie within the configured limits. The more stable the process value is, the easier it is to calculate the PID parameters and the more precise the result will be. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher as compared to the noise. This is most likely the case in operating modes "Inactive" and "manual mode". The setpoint is frozen in the CurrentSetpoint tag. Tuning is canceled when: • Setpoint > CurrentSetpoint + CancelTuningLevel or • Setpoint < CurrentSetpoint - CancelTuningLevel Before the PID parameters are recalculated, they are backed up and can be reactivated with LoadBackUp. The controller switches to automatic mode following successful pretuning. If pretuning is unsuccessful, the switchover of operating mode is dependent on ActivateRecoverMode and ErrorBehaviour. The pretuning phase is indicated with the SUT.State tag.

State	Description of operating mode
2	Fine tuning Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are recalculated based on the amplitude and frequency of this oscillation. PID parameters from fine tuning usually have better master control and disturbance characteristics than PID parameters from pretuning. You obtain the best PID parameters when you perform pretuning and fine tuning. PID_3Step automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value. The setpoint is frozen in the CurrentSetpoint tag. Tuning is canceled when: • Setpoint > CurrentSetpoint + CancelTuningLevel or • Setpoint < CurrentSetpoint - CancelTuningLevel The PID parameters are backed up before fine tuning. They can be reactivated with LoadBackUp. Requirements for fine tuning: • The motor transition time has been configured or measured. • The setpoint and the process value lie within the configured limits. • ManualEnable = FALSE • Reset = FALSE • Automatic (State = 3), inactive (State = 0) or manual (State = 4) mode Fine tuning proceeds as follows when started from: • Automatic mode (State = 3) Start fine tuning from automatic mode if you wish to improve the existing PID parameters through tuning. PID_3Step controls the system using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start. • Inactive (State = 0) or manual mode (State = 4) If the requirements for pretuning are met, pretuning is started. The determined PID parameters will be used for control until the control loop has stabilized and the requirements for fine tuning have been met. If PIDSelfTune.TIR.RunIn = TRUE, pretuning is skipped and an attempt is made to reach the setpoint with the minimum or maximum output value. This can produce increased overshoot. Fine tuning then starts automatically. The controller switches to automatic mode following successful fine tuning. If fine tuning is unsuccessful, the switchover of operating mode is dependent on ActivateRecoverMode and ErrorBehaviour. The fine tuning phase is indicated with the TIR.State tag.
3	Automatic mode In automatic mode, PID_3Step controls the controlled system in accordance with the parameters specified. The controller switches to automatic mode if one the following requirements is fulfilled: • Pretuning successfully completed • Fine tuning successfully completed • Changing of the Mode in-out parameter to the value 3 and a rising edge at ModeActivate. The switchover from automatic mode to manual mode is only bumpless if carried out in the commissioning editor. The ActivateRecoverMode tag is taken into consideration in automatic mode.

State	Description of operating mode
4	Manual mode In manual mode, you specify manual output values in the Manual_UP and Manual_DN parameters or ManualValue parameter. Whether or not the actuator can be moved to the output value in the event of an error is described in the ErrorBits parameter. You can also activate this operating mode using ManualEnable = TRUE. We recommend that you change the operating mode using Mode and ModeActivate only. The switchover from manual mode to automatic mode is bumpless. Manual mode is also possible when an error is pending.
5	Approach substitute output value This operating mode is activated in the event of an error, if Errorbehaviour = TRUE and ActivateRecoverMode = FALSE.. PID_3Step moves the actuator to the substitute output value and then switches to "Inactive" mode.
6	Transition time measurement The time that the motor needs to completely open the valve from the closed condition is determined. This operating mode is activated when Mode = 6 and ModeActivate = TRUE is set. If endstop signals are used to measure the transition time, the valve will be opened completely from its current position, closed completely, and opened completely again. If GetTransitTime.InvertDirection = TRUE, this behavior is inverted. If position feedback is used to measure the transition time, the actuator will be moved from its current position to a target position. The output value limits are not taken into consideration during the transition time measurement. The actuator can travel to the high or the low endstop.
7	Error monitoring The control algorithm is switched off and no longer changes the valve position. This operating mode is activated instead of "Inactive" mode in the event of an error. All the following conditions must be met: • Automatic mode (Mode = 3) • Errorbehaviour = FALSE • ActivateRecoverMode = TRUE • One or more errors have occurred in which ActivateRecoverMode (Page 5001) is effective. As soon as the errors are no longer pending, PID_3Step switches back to automatic mode.
8	Approach substitute output value with error monitoring This operating mode is activated instead of "approach substitute output value" mode when an error occurs. PID_3Step moves the actuator to the substitute output value and then switches to "error monitoring" mode. All the following conditions must be met: • Automatic mode (Mode = 3) • Errorbehaviour = TRUE • ActivateRecoverMode = TRUE • One or more errors have occurred in which ActivateRecoverMode (Page 5001) is effective. As soon as the errors are no longer pending, PID_3Step switches back to automatic mode.
10	Manual mode without endstop signals The endstop signals are **not** taken into consideration, even though Config.ActuatorEndStopOn = TRUE. The output value limits are not taken into consideration. Otherwise, PID_3Step behaves the same as in manual mode.

ENO characteristics

If State = 0, then ENO = FALSE.

If State ≠ 0, then ENO = TRUE.

Automatic switchover of operating mode during commissioning

Automatic mode is activated following successful pretuning or fine tuning. The following table shows how Mode and State change during successful pretuning.

Cycle no.	Mode	State	Action
0	4	4	Set Mode = 1
1	1	4	Set ModeActivate = TRUE
1	4	1	Value of State is saved in Mode parameter Pretuning is started
n	4	1	Pretuning successfully completed
n	3	3	Automatic mode is started

PID_3Step automatically switches the operating mode in the event of an error. The following table shows how Mode and State change during pretuning with errors.

Cycle no.	Mode	State	Action
0	4	4	Set Mode = 1
1	1	4	Set ModeActivate = TRUE
1	4	1	Value of State is saved in Mode parameter Pretuning is started
n	4	1	Pretuning canceled
n	4	4	Manual mode is started

If ActivateRecoverMode = TRUE, the operating mode that is saved in the Mode parameter is activated. At the start of transition time measurement, pretuning, or fine tuning, PID_3Step saved the value of State in the Mode in/out parameter. PID_3Step therefore switches to the operating mode from which transition time measurement or tuning was started.

If ActivateRecoverMode = FALSE, "Inactive" or "Approach substitute output value" mode is activated.

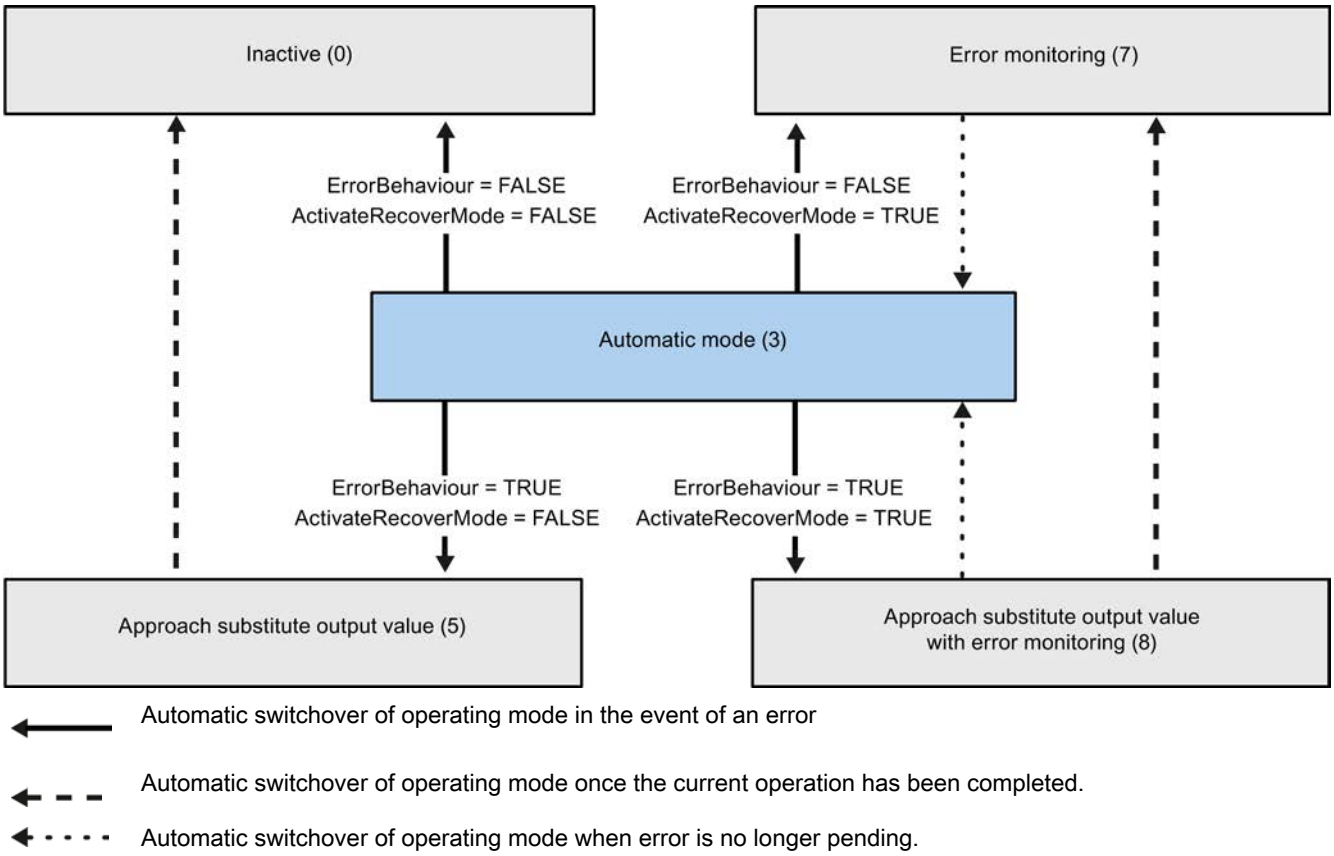
Automatic switchover of operating mode after transition time measurement

If ActivateRecoverMode = TRUE, the operating mode that is saved in the Mode parameter is activated after successful transition time measurement.

If ActivateRecoverMode = FALSE, the system switches to "Inactive" operating mode after successful transition time measurement.

Automatic switchover of operating mode in automatic mode

PID_3Step automatically switches the operating mode in the event of an error. The following diagram illustrates the influence of ErrorBehaviour and ActivateRecoverMode on this switchover of operating mode.



See also

Tag ActivateRecoverMode V2 (Page 5001)

Parameter ErrorBits V2 (Page 4998)

Parameter ErrorBits V2

If several errors are pending simultaneously, the values of the ErrorBits are displayed with binary addition. The display of ErrorBits = 0003h, for example, indicates that the errors 0001h and 0002h are pending simultaneously.

If there is a position feedback, PID_3Step uses ManualValue as output value in manual mode. The exception is Errorbits = 10000h.

ErrorBits (DW#16#...)	Description
0000	There is no error.
0001	The "Input" parameter is outside the process value limits. • Input > Config.InputUpperLimit or • Input < Config.InputLowerLimit If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_3Step remains in automatic mode. If pretuning, fine tuning, or transition time measurement mode and ActivateRecoverMode = TRUE were active before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter.
0002	Invalid value at "Input_PER" parameter. Check whether an error is pending at the analog input. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_3Step switches to "Approach substitute output value with error monitoring" or "Error monitoring" mode. As soon as the error is no longer pending, PID_3Step switches back to automatic mode. If pretuning, fine tuning, or transition time measurement mode and ActivateRecoverMode = TRUE were active before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter.
0004	Error during fine tuning. Oscillation of the process value could not be maintained. If ActivateRecoverMode = TRUE before the error occurred, PID_3Step cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0010	The setpoint was changed during tuning. You can set the permitted fluctuation of the setpoint at the CancelTuningLevel tag. If ActivateRecoverMode = TRUE before the error occurred, PID_3Step cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0020	Pretuning is not permitted during fine tuning. If ActivateRecoverMode = TRUE before the error occurred, PID_3Step remains in fine tuning mode.
0080	Error during pretuning. Incorrect configuration of output value limits. Check whether the limits of the output value are configured correctly and match the control logic. If ActivateRecoverMode = TRUE before the error occurred, PID_3Step cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0100	Error during fine tuning resulted in invalid parameters. If ActivateRecoverMode = TRUE before the error occurred, PID_3Step cancels the tuning and switches to the operating mode that is saved in the Mode parameter.
0200	Invalid value at "Input" parameter: Value has an invalid number format. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_3Step switches to "Approach substitute output value with error monitoring" or "Error monitoring" mode. As soon as the error is no longer pending, PID_3Step switches back to automatic mode. If pretuning, fine tuning, or transition time measurement mode and ActivateRecoverMode = TRUE were active before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter.
0400	Calculation of output value failed. Check the PID parameters. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_3Step switches to "Approach substitute output value with error monitoring" or "Error monitoring" mode. As soon as the error is no longer pending, PID_3Step switches back to automatic mode. If pretuning, fine tuning, or transition time measurement mode and ActivateRecoverMode = TRUE were active before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter.

ErrorBits (DW#16#...)	Description
0800	Sampling time error: PID_3Step is not called within the sampling time of the cyclic interrupt OB. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_3Step remains in automatic mode. If pretuning, fine tuning, or transition time measurement mode and ActivateRecoverMode = TRUE were active before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter.
1000	Invalid value at "Setpoint" parameter: Value has an invalid number format. If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_3Step switches to "Approach substitute output value with error monitoring" or "Error monitoring" mode. As soon as the error is no longer pending, PID_3Step switches back to automatic mode. If pretuning, fine tuning, or transition time measurement mode and ActivateRecoverMode = TRUE were active before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter.
2000	Invalid value at Feedback_PER parameter. Check whether an error is pending at the analog input. The actuator cannot be moved to the substitute output value and remains in its current position. In manual mode, you can change the position of the actuator only with Manual_UP and Manual_DN, and not with ManualValue. If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode. If pretuning, fine tuning, or transition time measurement mode and ActivateRecoverMode = TRUE were active before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter.
4000	Invalid value at Feedback parameter. Value has an invalid number format. The actuator cannot be moved to the substitute output value and remains in its current position. In manual mode, you can change the position of the actuator only with Manual_UP and Manual_DN, and not with ManualValue. If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode. If pretuning, fine tuning, or transition time measurement mode and ActivateRecoverMode = TRUE were active before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter.
8000	Error during digital position feedback. Actuator_H = TRUE and Actuator_L = TRUE. The actuator cannot be moved to the substitute output value and remains in its current position. Manual mode is not possible in this state. In order to move the actuator from this state, you must deactivate the "Actuator endstop" (Config.ActuatorEndStopOn = FALSE) or switch to manual mode without endstop signals (Mode = 10). If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode. If pretuning, fine tuning, or transition time measurement mode and ActivateRecoverMode = TRUE were active before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter.
10000	Invalid value at ManualValue parameter. Value has an invalid number format. The actuator cannot be moved to the manual value and remains in its current position. Specify a valid value in ManualValue or move the actuator in manual mode with Manual_UP and Manual_DN.

ErrorBits (DW#16#...)	Description
20000	Invalid value at SavePosition tag. Value has an invalid number format. The actuator cannot be moved to the substitute output value and remains in its current position.
40000	Invalid value at Disturbance parameter. Value has an invalid number format. If automatic mode was active and ActivateRecoverMode = TRUE before the error occurred, Disturbance is set to zero. PID_3Step remains in automatic mode. If pretuning or fine tuning mode was active and ActivateRecoverMode = TRUE before the error occurred, PID_3Step switches to the operating mode saved in the Mode parameter. If Disturbance in the current phase has no effect on the output value, tuning is not be canceled. The error has no effect during transition time measurement.

Tag ActivateRecoverMode V2

The ActivateRecoverMode tag determines the reaction to error. The Error parameter indicates if an error is pending. When the error is no longer pending, Error = FALSE. The ErrorBits parameter shows which errors have occurred.

NOTICE
Your system may be damaged. If ActivateRecoverMode = TRUE, PID_3Step remains in automatic mode even if the process limit values are exceeded. This may damage your system. It is essential to configure how your controlled system reacts in the event of an error to protect your system from damage.

Automatic mode

ActivateRecoverMode	Description
FALSE	In the event of an error, PID_3Step switches to "Inactive" or "Approach substitute output value" mode. The controller is only activated by a falling edge at Reset or a rising edge at ModeActivate.
TRUE	If errors occur frequently in automatic mode, this setting has a negative effect on the control response, because PID_3Step switches between the calculated output value and the substitute output value at each error. In this case, check the ErrorBits parameter and eliminate the cause of the error. If one or more of the following errors occur, PID_3Step stays in automatic mode: • 0001h: The "Input" parameter is outside the process value limits. • 0800h: Sampling time error • 40000h: Invalid value at Disturbance parameter. If one or more of the following errors occur, PID_3Step switches to "Approach substitute output value with error monitoring" or "Error monitoring" mode: • 0002h: Invalid value at Input_PER parameter. • 0200h: Invalid value at Input parameter. • 0400h: Calculation of output value failed. • 1000h: Invalid value at Setpoint parameter. If one or more of the following errors occur, PID_3Step can no longer move the actuator: • 2000h: Invalid value at Feedback_PER parameter. • 4000h: Invalid value at Feedback parameter. • 8000h: Error during digital position feedback. • 20000h: Invalid value at SavePosition tag. Value has an invalid number format. The characteristics are independent of ErrorBehaviour. As soon as the errors are no longer pending, PID_3Step switches back to automatic mode.

Pretuning, fine tuning, and transition time measurement

ActivateRecoverMode	Description
FALSE	In the event of an error, PID_3Step switches to "Inactive" or "Approach substitute output value" mode. The controller is only activated by a falling edge at Reset or a rising edge at ModeActivate. The controller changes to "Inactive" mode after successful transition time measurement.
TRUE	If the following error occurs, PID_3Step remains in the active mode: • 0020h: Pretuning is not permitted during fine tuning. The following errors are ignored: • 10000h: Invalid value at ManualValue parameter. • 20000h: Invalid value at SavePosition tag. When any other error occurs, PID_3Step cancels the tuning and switches to the mode from which tuning was started.

Manual mode

ActivateRecoverMode is not effective in manual mode.

See also

Static tags of PID_3Step V2 (Page 4985)

Parameters State and Mode V2 (Page 4993)

Tag Warning V2

If several warnings are pending simultaneously, their values are displayed with binary addition. The display of warning 0005h, for example, indicates that the warnings 0001h and 0004h are pending simultaneously.

Warning (DW#16#...)	Description
0000	No warning pending.
0001	The point of inflection was not found during pretuning.
0004	The setpoint was limited to the configured limits.
0008	Not all the necessary controlled system properties were defined for the selected method of calculation. Instead, the PID parameters were calculated using the TIR.TuneRule = 3 method.
0010	The operating mode could not be changed because Reset = TRUE or ManualEnable = TRUE.
0020	The cycle time of the calling OB limits the sampling time of the PID algorithm. Improve results by using shorter OB cycle times.
0040	The process value exceeded one of its warning limits.
0080	Invalid value at Mode. The operating mode is not switched.
0100	The manual value was limited to the limits of the controller output.
0200	The specified rule for tuning is not supported. No PID parameters are calculated.
0400	The transition time cannot be measured because the actuator settings do not match the selected measuring method.
0800	The difference between the current position and the new output value is too small for transition time measurement. This can produce incorrect results. The difference between the current output value and new output value must be at least 50% of the entire control range.
1000	The substitute output value cannot be reached because it is outside the output value limits.
2000	The actuator was moved in one direction for longer than Config.VirtualActuatorLimit × Retain.TransitTime. Check whether the actuator has reached an endstop signal.

The following warnings are deleted as soon as the cause is eliminated:

- 0001h
- 0004h
- 0008h
- 0040h
- 0100h
- 2000h

All other warnings are cleared with a rising edge at Reset or ErrorAck.

PID_3Step V1

Description PID_3Step V1

Description

You use the PID_3Step instruction to configure a PID controller with self tuning for valves or actuators with integrating behavior.

The following operating modes are possible:

- Inactive
- Pretuning
- Fine tuning
- Automatic mode
- Manual mode
- Approach substitute output value
- Transition time measurement
- Approach substitute output value with error monitoring
- Error monitoring

For a more detailed description of the operating modes, see the State parameter.

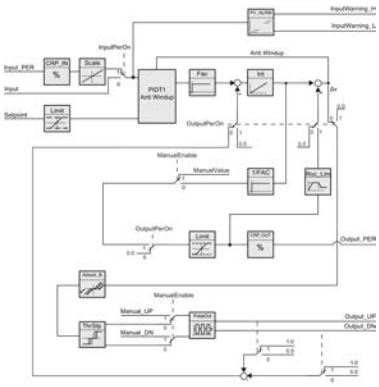
PID algorithm

PID_3Step is a PIDT1 controller with anti-windup and weighting of the proportional and derivative actions. The following equation is used to calculate the output value.

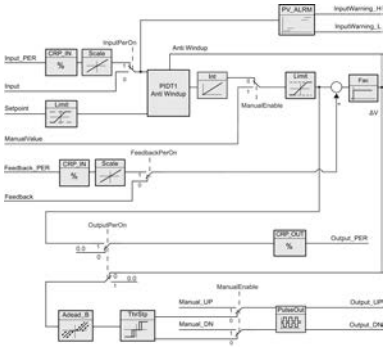
$$\Delta y = K_p \cdot s \cdot \left[(b \cdot w - x) + \frac{1}{T_I \cdot s}(w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1}(c \cdot w - x) \right]$$

Symbol	Description
y	Output value
K_p	Proportional gain
s	Laplace operator
b	Proportional action weighting
w	Setpoint
x	Process value
T_I	Integral action time
a	Derivative delay coefficient (T1 = a × T_D)
T_D	Derivative action time
c	Derivative action weighting

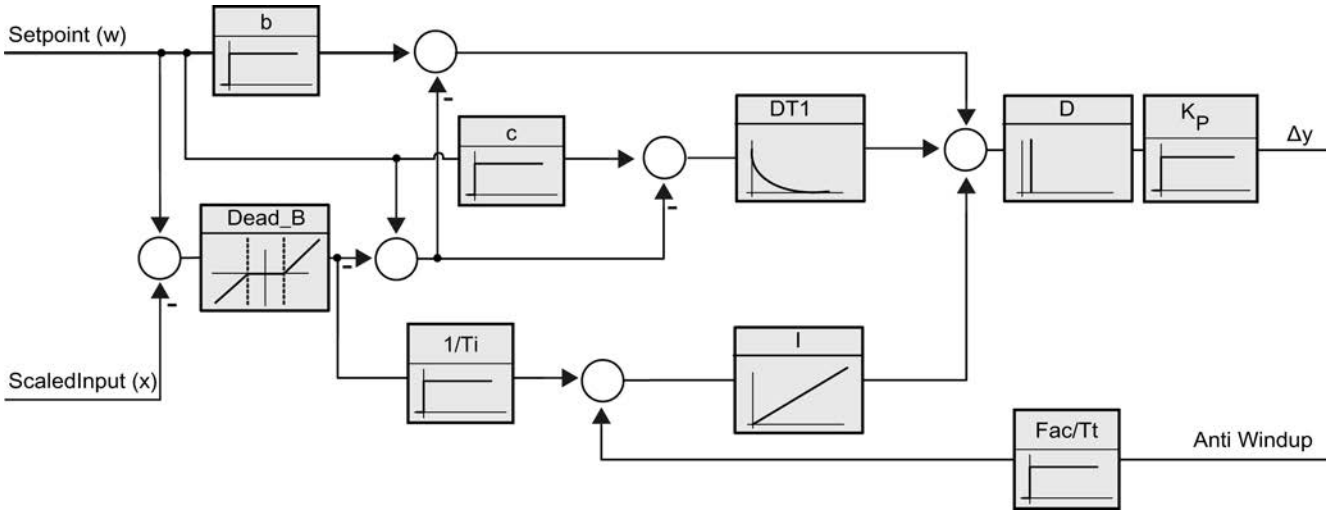
Block diagram without position feedback



Block diagram with position feedback



Block diagram of PIDT1 with anti-windup



Call

PID_3Step is called in a constant time interval of the cycle time of the calling OB (preferably in a cyclic interrupt OB).

Download to device

The actual values of retentive tags are only updated when you download PID_3Step completely.

Downloading technology objects to device (Page 9536)

Startup

At the startup of the CPU, PID_3Step starts in the operating mode that was last active. To leave PID_3Step in "Inactive" mode, set RunModeByStartup = FALSE.

Reaction to error

If errors occur, these are output in the Error parameter. You configure the reaction of PID_3Step using the ErrorBehaviour and ActivateRecoverMode tags.

ErrorBehaviour	ActivateRecoverMode	Actuator setting configuration Set Output to	Reaction
0	FALSE	Current output value	Switch to "Inactive" mode (Mode = 0)
0	TRUE	Current output value while error is pending	Switch to "Error monitoring" mode (Mode = 7)
1	FALSE	Substitute output value	Switch to "Approach substitute output value" mode (Mode = 5) Switch to "Inactive" mode (Mode = 0)
1	TRUE	Substitute output value while error is pending	Switch to "Approach substitute output value with error monitoring" mode (Mode = 8) Switch to "Error monitoring" mode (Mode = 7)

The ErrorBits parameter shows which errors have occurred.

See also

Parameter State and Retain.Mode V1 (Page 5019)

Parameter ErrorBits V1 (Page 5026)

Configuring PID_3Step V1 (Page 9607)

Operating principle PID_3Step V1

Monitoring process value limits

You specify the high limit and low limit of the process value in the Config.InputUpperLimit and Config.InputLowerLimit tags. If the process value is outside these limits, an error occurs (ErrorBits = 0001hex).

You specify a high and low warning limit of the process value in the Config.InputUpperWarning and Config.InputLowerWarning tags. If the process value is outside these warning limits, a warning occurs (Warnings = 0040hex), and the InputWarning_H or InputWarning_L output parameter changes to TRUE.

Limiting the setpoint

You specify a high limit and low limit of the setpoint in the Config.SetpointUpperLimit and Config.SetpointLowerLimit tags. PID_3Step automatically limits the setpoint to the process value limits. You can limit the setpoint to a smaller range. PID_3Step checks whether this range falls within the process value limits. If the setpoint is outside these limits, the high or low limit is used as the setpoint, and output parameter SetpointLimit_H or SetpointLimit_L is set to TRUE.

The setpoint is limited in all operating modes.

Limiting the output value

You specify a high limit and low limit of the output value in the Config.OutputUpperLimit and Config.OutputLowerLimit tags. The output value limits must be within "Low endstop" and "High endstop".

- High endstop: Config.FeedbackScaling.UpperPointOut
- Low endstop: Config.FeedbackScaling.LowerPointOut

Rule:

UpperPointOut ≥ OutputUpperLimit > OutputLowerLimit ≥ LowerPointOut

The valid values for "High endstop" and "Low endstop" depend upon:

- FeedbackOn
- FeedbackPerOn
- OutputPerOn

OutputPerOn	FeedbackOn	FeedbackPerOn	LowerPointOut	UpperPointOut
FALSE	FALSE	FALSE	Cannot be set (0.0%)	Cannot be set (100.0%)
FALSE	TRUE	FALSE	-100.0% or 0.0%	0.0% or +100.0%
FALSE	TRUE	TRUE	-100.0% or 0.0%	0.0% or +100.0%
TRUE	FALSE	FALSE	Cannot be set (100.0%)	Cannot be set (100.0%)
TRUE	TRUE	FALSE	-100.0% or 0.0%	0.0% or +100.0%
TRUE	TRUE	TRUE	-100.0% or 0.0%	0.0% or +100.0%

If OutputPerOn = FALSE and FeedbackOn = FALSE, you cannot limit the output value. The digital outputs are reset with Actuator_H = TRUE or Actuator_L = TRUE, or after a travel time amounting to 110% of the motor transition time.

The output value is 27648 at 100% and -27648 at -100%. PID_3Step must be able to close the valve completely. Therefore, zero must be included in the output value limits.

Substitute output value

If an error has occurred, PID_3Step can output a substitute output value and move the actuator to a safe position that is specified in the SavePosition tag. The substitute output value must be within the output value limits.

Monitoring signal validity

The values of the following parameters are monitored for validity:

- Setpoint
- Input
- Input_PER
- Feedback

- Feedback_PER
- Output

Monitoring the PID_3Step sampling time

Ideally, the sampling time is equivalent to the cycle time of the calling OB. The PID_3Step instruction measures the time interval between two calls. This is the current sampling time. On every switchover of operating mode and during the initial startup, the mean value is formed from the first 10 sampling times. Too great a difference between the current sampling time and this mean value triggers an error (ErrorBits = 0800 hex).

PID_3Step is set to "Inactive" mode during tuning under the following conditions:

- New mean value >= 1.1 x old mean value
- New mean value <= 0.9 x old mean value

In automatic mode, PID_3Step is set to "Inactive" mode under the following conditions:

- New mean value >= 1.5 x old mean value
- New mean value <= 0.5 x old mean value

Sampling time of the PID algorithm

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is calculated during tuning and rounded to a multiple of the cycle time. All other functions of PID_3Step are executed at every call.

Measuring the motor transition time

The motor transition time is the time in seconds the motor requires to move the actuator from the closed to the opened state. The maximum time that the actuator is moved in one direction is 110% of the motor transition time. PID_3Step requires the motor transition time to be as accurate as possible for good controller results. The data in the actuator documentation contains average values for this type of actuator. The value for the specific actuator used may differ. You can measure the motor transition time during commissioning. The output value limits are not taken into consideration during the motor transition time measurement. The actuator can travel to the high or the low endstop.

Control logic

An increase of the output value is generally intended to cause an increase in the process value. This is referred to as a normal control logic. For cooling and discharge control systems, it may be necessary to invert the control logic. PID_3Step does not work with negative proportional gain. If InvertControl = TRUE, an increasing control deviation causes a reduction in the output value. The control logic is also taken into account during pretuning and fine tuning.

See also

Configuring PID_3Step V1 (Page 9607)

PID_3Step V1 input parameters

Table 9-91

Parameters	Data type	Default	Description
Setpoint	REAL	0.0	Setpoint of the PID controller in automatic mode
Input	REAL	0.0	A variable of the user program is used as source for the process value. If you are using parameter Input, then Config.InputPerOn = FALSE must be set.
Input_PER	WORD	W#16#0	An analog input is used as source for the process value. If you are using parameter Input_PER, then Config.InputPerOn = TRUE must be set.
Actuator_H	BOOL	FALSE	Digital position feedback of the valve for the high endstop If Actuator_H = TRUE, the valve is at the high endstop and is no longer moved towards this direction.
Actuator_L	BOOL	FALSE	Digital position feedback of the valve for the low endstop If Actuator_L = TRUE, the valve is at the low endstop and is no longer moved towards this direction.
Feedback	REAL	0.0	Position feedback of the valve If you are using parameter Feedback, then Config.FeedbackPerOn = FALSE must be set.
Feedback_PER	WORD	W#16#0	Analog feedback of the valve position If you are using parameter Feedback_PER, then Config.FeedbackPerOn = TRUE must be set. Feedback_PER is scaled based on the variables: • Config.FeedbackScaling.LowerPointIn • Config.FeedbackScaling.UpperPointIn • Config.FeedbackScaling.LowerPointOut • Config.FeedbackScaling.UpperPointOut
ManualEnable	BOOL	FALSE	• A FALSE -> TRUE edge selects "Manual mode", while State = 4, Retain.Mode remains unchanged. • A TRUE -> FALSE edge selects the most recently active operating mode A change of Retain.Mode will not take effect during ManualEnable = TRUE. The change of Retain.Mode will only be considered upon a TRUE -> FALSE edge at ManualEnable . **PID_3Step V1.1**If at start of the CPU ManualEnable = TRUE, PID_3Step starts in manual mode. A rising edge (FALSE > TRUE) at ManualEnable is not necessary. **PID_3Step V1.0** At the start of the CPU, PID_3Step only switches to manual mode with a rising edge (FALSE->TRUE) at ManualEnable . Without rising edge, PID_3Step starts in the last operating mode in which ManualEnable was FALSE.

Parameters	Data type	Default	Description
ManualValue	REAL	0.0	In manual mode, you specify the absolute position of the valve. ManualValue will only be evaluated if you are using OutputPer, or if position feedback is available.
Manual_UP	BOOL	FALSE	In manual mode, every rising edge opens the valve by 5% of the total control range, or for the duration of the minimum motor transition time. Manual_UP is evaluated only if you are not using Output_PER and there is no position feedback available.
Manual_DN	BOOL	FALSE	In manual mode, every rising edge closes the valve by 5% of the total control range, or for the duration of the minimum motor transition time. Manual_DN is evaluated only if you are not using Output_PER and there is no position feedback available.
Reset	BOOL	FALSE	Restarts the controller. • FALSE -> TRUE edge – Change to "Inactive" mode – Intermediate controller values are reset (PID parameters are retained) • TRUE -> FALSE edge Change in most recent active mode

PID_3Step V1 output parameters

Table 9-92

Parameter	Data type	Default	Description
ScaledInput	REAL	0.0	Scaled process value
ScaledFeedback	REAL	0.0	Scaled position feedback For an actuator without position feedback, the position of the actuator indicated by ScaledFeedback is very imprecise. ScaledFeedback may only be used for rough estimation of the current position in this case.
Output_UP	BOOL	FALSE	Digital output value for opening the valve If Config.OutputPerOn = FALSE, the Output_UP parameter is used.
Output_DN	BOOL	FALSE	Digital output value for closing the valve If Config.OutputPerOn = FALSE, the Output_DN parameter is used.
Output_PER	WORD	W#16#0	Analog output value If Config.OutputPerOn = TRUE, Output_PER is used.
SetpointLimit_H	BOOL	FALSE	If SetpointLimit_H = TRUE, the absolute setpoint high limit is reached. In the CPU, the setpoint is limited to the configured absolute setpoint high limit. The configured absolute process value high limit is the default for the setpoint high limit. If you configure Config.SetpointUpperLimit to a value within the process value limits, this value is used as the setpoint high limit.

Parameter	Data type	Default	Description
SetpointLimit_L	BOOL	FALSE	If SetpointLimit_L = TRUE, the absolute setpoint low limit has been reached. In the CPU, the setpoint is limited to the configured absolute setpoint low limit. The configured absolute process value low limit is the default setting for the setpoint low limit. If you configure Config.SetpointLowerLimit to a value within the process value limits, this value is used as the setpoint low limit.
InputWarning_H	BOOL	FALSE	If InputWarning_H = TRUE, the process value has reached or exceeded the warning high limit.
InputWarning_L	BOOL	FALSE	If InputWarning_L = TRUE, the process value has reached or fallen below the warning low limit.
State	INT	0	The State parameter (Page 5019) shows the current operating mode of the PID controller. You change the operating mode with the Retain.Mode tag. • State = 0: Inactive • State = 1: Pretuning • State = 2: Fine tuning • State = 3: Automatic mode • State = 4: Manual mode • State = 5: Approach substitute output value • State = 6: Transition time measurement • State = 7: Error monitoring • State = 8: Approach substitute output value with error monitoring
Error	BOOL	FALSE	If Error = TRUE, at least one error message is pending.
ErrorBits	DWORD	DW#16#0	The ErrorBits parameter (Page 5026) indicates the error messages.

See also

Parameter State and Retain.Mode V1 (Page 5019)

Parameter ErrorBits V1 (Page 5026)

PID_3Step V1 static tags

You must not change tags that are not listed. These are used for internal purposes only.

Table 9-93

Tag	Data type	Default	Description
ActivateRecoverMode	BOOL	TRUE	The ActivateRecoverMode tag (Page 5027) determines the reaction to error.
RunModeByStartup	BOOL	TRUE	Activate Mode after CPU restart If RunModeByStartup = TRUE, the controller returns to the last active operating mode after a CPU restart. If RunModeByStartup = FALSE, the controller remains inactive after a CPU restart.

Tag	Data type	Default	Description
PhysicalUnit	INT	0	Unit of measurement of the process value and setpoint, e.g., ºC, or ºF.
PhysicalQuantity	INT	0	Physical quantity of the process value and setpoint, e.g., temperature.
ErrorBehaviour	INT	0	If ErrorBehaviour = 0 and an error has occurred, the valve stays at its current position and the controller switches directly to "Inactive" or "Error monitoring" mode. If ErrorBehaviour = 1 and an error occurs, the actuator moves to the substitute output value and only then switches to "Inactive" or "Error monitoring" mode. If the following errors occur, you can no longer move the valve to a configured substitute output value. • 2000h: Invalid value at Feedback_PER parameter. • 4000h: Invalid value at Feedback parameter. • 8000h: Error during digital position feedback.
Warning	DWORD	DW#16#0	The Warning tag (Page 5019) displays the warnings generated since a Reset or since the last switchover of operating mode. Cyclic warnings (for example, process value warning) are shown until the cause of the warning is removed. They are automatically deleted once their cause has gone. Non-cyclic warnings (for example, point of inflection not found) remain and are deleted like errors.
SavePosition	REAL	0.0	Substitute output value If ErrorBehaviour = 1 and an error occurs, the actuator moves to a safe position for the plant and only then switches to "Inactive" mode.
CurrentSetpoint	REAL	0.0	Currently active setpoint. This value is frozen at the start of tuning.
Progress	REAL	0.0	Progress of tuning as a percentage (0.0 - 100.0)
Config.InputPerOn	BOOL	TRUE	If InputPerOn = TRUE, the Input_PER parameter is used. If InputPerOn = FALSE, the Input parameter is used.
Config.OutputPerOn	BOOL	FALSE	If OutputPerOn = TRUE, the Output_PER parameter is used. If OutputPerOn = FALSE, the Ouput_UP and Output_DN parameters are used.
Config.LoadBackUp	BOOL	FALSE	If LoadBackUp = TRUE, the last set of PID parameters is reloaded. This set was saved prior to the last tuning operation. LoadBackUp is automatically reset to FALSE.
Config.InvertControl	BOOL	FALSE	Invert control logic If InvertControl = TRUE, an increasing control deviation causes a reduction in the output value.
Config.FeedbackOn	BOOL	FALSE	If FeedbackOn = FALSE, a position feedback is simulated. Position feedback is generally activated when FeedbackOn = TRUE.
Config.FeedbackPerOn	BOOL	FALSE	FeedbackPerOn is only effective when FeedbackOn = TRUE. If FeedbackPerOn = TRUE, the analog input is used for the position feedback (Feedback_PER parameter). If FeedbackPerOn = FALSE, the Feedback parameter is used for the position feedback.
Config.ActuatorEndStopOn	BOOL	FALSE	If ActuatorEndStopOn = TRUE, the digital position feedback Actuator_L and Actuator_H are taken into consideration.

Tag	Data type	Default	Description
Config.InputUpperLimit	REAL	120.0	High limit of the process value At the I/O input, the process value can be a maximum of 18% higher than the standard range (overrange). An error is no longer signaled due to a violation of the "Process value high limit". Only a wire-break and a short-circuit are recognized and PID_3Step reacts according to the configured reaction to error. InputUpperLimit > InputLowerLimit
Config.InputLowerLimit	REAL	0.0	Low limit of the process value InputLowerLimit < InputUpperLimit
Config.InputUpperWarning	REAL	+3.402822e+38	Warning high limit of the process value If you set InputUpperWarning outside the process value limits, the configured absolute process value high limit is used as the warning high limit. If you configure InputUpperWarning within the process value limits, this value is used as the warning high limit. InputUpperWarning > InputLowerWarning InputUpperWarning ≤ InputUpperLimit
Config.InputLowerWarning	REAL	-3.402822e+38	Warning low limit of the process value If you set InputLowerWarning outside the process value limits, the configured absolute process value low limit is used as the warning low limit. If you configure InputLowerWarning within the process value limits, this value is used as the warning low limit. InputLowerWarning < InputUpperWarning InputLowerWarning ≥ InputLowerLimit
Config.OutputUpperLimit	REAL	100.0	High limit of output value For details, see OutputLowerLimit
Config.OutputLowerLimit	REAL	0.0	Low limit of output value If OutputPerOn = TRUE or FeedbackOn = TRUE, the range of values from -100% to +100%, including zero, is valid. At -100%, Output = -27648; at +100%, Output = 27648 If OutputPerOn = FALSE, the range of values from 0% to 100% is valid. The valve is completely closed at 0% and completely opened at 100%.
Config.SetpointUpperLimit	REAL	+3.402822e+38	High limit of setpoint If you set SetpointUpperLimit outside the process value limits, the configured absolute process value high limit is preassigned as the setpoint high limit. If you configure SetpointUpperLimit within the process value limits, this value is used as the setpoint high limit.
Config.SetpointLowerLimit	REAL	-3.402822e+38	Low limit of the setpoint If you set SetpointLowerLimit outside the process value limits, the configured absolute process value low limit is preassigned as the setpoint low limit. If you set SetpointLowerLimit within the process value limits, this value is used as the setpoint low limit.

Tag	Data type	Default	Description
Config.MinimumOnTime	REAL	0.0	Minimum ON time Minimum time in seconds for which the servo drive must be switched on.
Config.MinimumOffTime	REAL	0.0	Minimum OFF time Minimum time in seconds for which the servo drive must be switched off.
Config.TransitTime	REAL	30.0	Motor transition time Time in seconds the actuating drive requires to move the valve from the closed to the opened state.
Config.InputScaling.UpperPointIn	REAL	27648.0	Scaling Input_PER high Input_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the InputScaling structure.
Config.InputScaling.LowerPointIn	REAL	0.0	Scaling Input_PER low Input_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the InputScaling structure.
Config.InputScaling.UpperPointOut	REAL	100.0	Scaled high process value Input_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the InputScaling structure.
Config.InputScaling.LowerPointOut	REAL	0.0	Scaled low process value Input_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the InputScaling structure.
Config.FeedbackScaling.UpperPointIn	REAL	27648.0	Scaling Feedback_PER high Feedback_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the FeedbackScaling structure.
Config.FeedbackScaling.LowerPointIn	REAL	0.0	Scaling Feedback_PER low Feedback_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the FeedbackScaling structure.
Config.FeedbackScaling.UpperPointOut	REAL	100.0	High endstop Feedback_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the FeedbackScaling structure.
Config.FeedbackScaling.LowerPointOut	REAL	0.0	Low endstop Feedback_PER is converted to a percentage based on the two value pairs UpperPointOut, UpperPointIn and LowerPointOut, LowerPointIn of the FeedbackScaling structure.
GetTransitTime.InvertDirection	BOOL	FALSE	If InvertDirection = FALSE, the valve is fully opened, closed, and then reopened in order to determine the valve transition time. If InvertDirection = TRUE, the valve is fully closed, opened, and then closed again.

Tag	Data type	Default	Description
GetTransitTime.SelectFeedback	BOOL	FALSE	If SelectFeedback = TRUE, then Feedback_PER, or Feedback is taken into consideration in the transition time measurement. If SelectFeedback = FALSE, then Actuator_H and Actuator_L are taken into consideration in the transition time measurement.
GetTransitTime.Start	BOOL	FALSE	If Start = TRUE, the transition time measurement is started.
GetTransitTime.State	INT	0	Current phase of the transition time measurement • State = 0: Inactive • State = 1: Open valve completely • State = 2: Close valve completely • State = 3: Move valve to target position (NewOutput) • State = 4: Transition time measurement successfully completed • State = 5: Transition time measurement canceled
GetTransitTime.NewOutput	REAL	0.0	Target position for transition time measurement with position feedback The target position must be between "High endstop" and "Low endstop". The difference between NewOutput and ScaledFeedback must be at least 50% of the permissible control range.
CycleTime.StartEstimation	BOOL	TRUE	If StartEstimation = TRUE, the measurement of the PID_3Step sampling time is started. CycleTime.StartEstimation = FALSE once measurement is complete.
CycleTime.EnEstimation	BOOL	TRUE	If EnEstimation = TRUE, the PID_3Step sampling time is calculated.
CycleTime.EnMonitoring	BOOL	TRUE	If EnMonitoring = TRUE, the PID_3Step sampling time is monitored. If it is not possible to execute PID_3Step within the sampling time, the error 0800h is output and the operating mode is switched. ActivateRecoverMode and ErrorBehaviour determine which operating mode is switched to. If EnMonitoring = FALSE, the PID_3Step sampling time is not monitored, the error 0800h is not output, and the operating mode is not switched.
CycleTime.Value	REAL	0.1	PID_3Step sampling time in seconds CycleTime.Value is determined automatically and is usually equivalent to the cycle time of the calling OB.
CtrlParamsBackUp.SetByUser	BOOL	FALSE	Saved value of Retain.CtrlParams.SetByUser. You can reload values from the CtrlParamsBackUp structure with Config.LoadBackUp = TRUE.
CtrlParamsBackUp.Gain	REAL	1.0	Saved proportional gain
CtrlParamsBackUp.Ti	REAL	20.0	Saved integral action time
CtrlParamsBackUp.Td	REAL	0.0	Saved derivative action time
CtrlParamsBackUp.TdFiltRatio	REAL	0.0	Saved derivative delay coefficient
CtrlParamsBackUp.PWeighting	REAL	0.0	Saved proportional action weighting
CtrlParamsBackUp.DWeighting	REAL	0.0	Saved derivative action weighting
CtrlParamsBackUp.Cycle	REAL	1.0	Saved sampling time of PID algorithm
CtrlParamsBackUp.InputDeadBand	REAL	0.0	Saved dead band width of the control deviation

Tag	Data type	Default	Description
PIDSelfTune.SUT.CalculateSUTParams	BOOL	FALSE	The properties of the controlled system are saved during tuning. If CalculateSUTParams = TRUE, the PID parameters are recalculated on the basis of these properties. The PID parameters are calculated using the method set in TuneRuleSUT. CalculateSUTParams is set to FALSE following calculation.
PIDSelfTune.SUT.TuneRuleSUT	INT	1	Methods used to calculate parameters during pretuning: • TuneRuleSUT = 0: PID rapid I • TuneRuleSUT = 1: PID slow I • TuneRuleSUT = 2: Chien, Hrones and Reswick PID • TuneRuleSUT = 3TuneRuleSUT = 3: Chien, Hrones, Reswick PI • TuneRuleSUT = 4: PID rapid II • TuneRuleSUT = 5: PID slow II
PIDSelfTune.SUT.State	INT	0	The SUT.State tag indicates the current phase of pretuning:
PIDSelfTune.TIR.RunIn	BOOL	FALSE	• RunIn = FALSE Pretuning is started when fine tuning is started from inactive or manual mode. If fine tuning is started from automatic mode, the system uses the existing PID parameters to control to the setpoint. Only then will fine tuning start. If pretuning is not possible, PID_3Step switches to "Inactive" mode. • RunIn = TRUE The pretuning is skipped. PID_3Step attempts to reach the setpoint with the minimum or maximum output value. This can produce increased overshoot. Only then will fine tuning start. RunIn is set to FALSE after fine tuning.
PIDSelfTune.TIR.CalculateTIRParams	BOOL	FALSE	The properties of the controlled system are saved during tuning. If CalculateTIRParams = TRUE, the PID parameters are recalculated on the basis of these properties. The PID parameters are calculated using the method set in TuneRuleTIR. CalculateTIRParams is set to FALSE following calculation.
PIDSelfTune.TIR.TuneRuleTIR	INT	0	Methods used to calculate parameters during fine tuning: • TuneRuleTIR = 0: PID automatic • TuneRuleTIR = 1: PID rapid • TuneRuleTIR = 2: PID slow • TuneRuleTIR = 3: Ziegler-Nichols PID • TuneRuleTIR = 4: Ziegler-Nichols PI • TuneRuleTIR = 5: Ziegler-Nichols P
PIDSelfTune.TIR.State	INT	0	The TIR.State tag indicates the current phase of "fine tuning":

Tag	Data type	Default	Description
Retain.Mode	INT	0	A change to the value of Retain.Mode initiates a switch to another operating mode. The following operating mode is enabled upon a change of Mode to: • Mode = 0: Inactive • Mode = 1: Pretuning • Mode = 2: Fine tuning • Mode = 3: Automatic mode • Mode = 4: Manual mode • Mode = 5: Approach substitute output value • Mode = 6: Transition time measurement • Mode = 7Mode = 7: Error monitoring • Mode = 8: Approach substitute output value with error monitoring Mode is retentive.
Retain.CtrlParams.SetByUser	BOOL	FALSE	If SetByUser = FALSE, the PID parameters are determined automatically and PID_3Step operates with a dead band at the output value. The dead band width is calculated during tuning on the basis of the standard deviation of the output value and saved in Retain.CtrlParams.OutputDeadBand. If SetByUser = TRUE, the PID parameters are entered manually and PID_3 Step operates without a dead band at the output value. Retain.CtrlParams.OutputDeadBand = 0.0 SetByUser is retentive.
Retain.CtrlParams.Gain	REAL	1.0	Active proportional gain Gain is retentive.
Retain.CtrlParams.Ti	REAL	20.0	• Ti > 0.0: Active integral action time • Ti = 0.0: Integral action is deactivated Ti is retentive.
Retain.CtrlParams.Td	REAL	0.0	• Td > 0.0: Active derivative action time • Td = 0.0: Derivative action is deactivated Td is retentive.
Retain.CtrlParams.TdFiltRatio	REAL	0.0	Active derivative delay coefficient TdFiltRatio is retentive.
Retain.CtrlParams.PWeighting	REAL	0.0	Active proportional action weighting PWeighting is retentive.
Retain.CtrlParams.DWeightin g	REAL	0.0	Active derivative action weighting DWeighting is retentive.
Retain.CtrlParams.Cycle	REAL	1.0	Active sampling time of PID algorithm in seconds, rounded to an integer multiple of the cycle time of the calling OB. Cycle is retentive.
Retain.CtrlParams.InputDead Band	REAL	0.0	Dead band width of the control deviation InputDeadBand is retentive.

Note

Change the tags listed in this table in "Inactive" mode to prevent malfunction of the PID controller. "Inactive" mode is forced by setting the "Retain.Mode" tag to "0".

See also

Parameter State and Retain.Mode V1 (Page 5019)

Tag ActivateRecoverMode V1 (Page 5027)

Downloading technology objects to device (Page 9536)

Parameter State and Retain.Mode V1

Correlation of the parameters

The State parameter shows the current operating mode of the PID controller. You cannot change the State parameter.

To switch from one operating mode to another, you must change the Retain.Mode tag. This also applies when the value for the new operating mode is already in Retain.Mode. For example, set Retain.Mode = 0 first and then Retain.Mode = 3. Provided the current operating mode of the controller permits this switchover, State will be set to the value of Retain.Mode.

When PID_3Step automatically switches from one operating mode to another, the following applies: State != Retain.Mode.

Examples:

- After successful pretuning
  State = 3 and Retain.Mode = 1
- In the event of an error
  State = 0 and Retain.Mode remain at the previous value, for example, Retain.Mode = 3
- ManualEnalbe = TRUE
  State = 4 and Retain.Mode remain at the previous value, e.g., Retain.Mode = 3

Note

You want, for example, to repeat successful fine tuning without exiting automatic mode with Mode = 0.

Setting Retain.Mode to an invalid value such as 9999 for one cycle has no effect on State. Set Mode = 2 in the next cycle. In this way, you can generate a change to Retain.Mode without first switching to "Inactive" mode.

Meaning of values

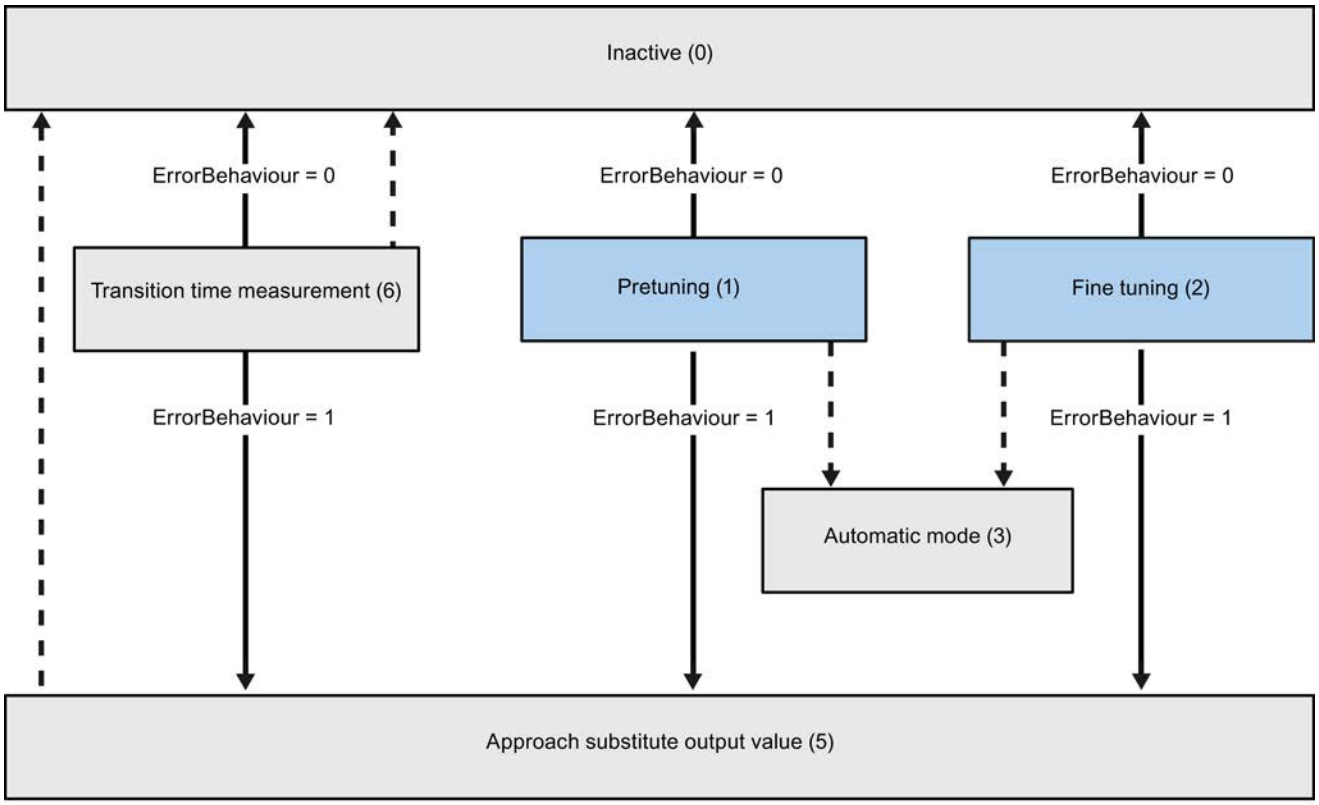
State / Retain.Mode	Description
0	Inactive The controller is switched off and no longer changes the valve position.
1	Pretuning The pretuning determines the process response to a pulse of the output value and searches for the point of inflection. The optimized PID parameters are calculated as a function of the maximum rate of rise and dead time of the controlled system. Pretuning requirements: • State = 0 or State = 4 • ManualEnable = FALSE • The motor transition time has been configured or measured. • The setpoint and the process value lie within the configured limits. The more stable the process value is, the easier it is to calculate the PID parameters and the more precise the result will be. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher as compared to the noise. Before the PID parameters are recalculated, they are backed up and can be reactivated with Config.LoadBackUp. The setpoint is frozen in the CurrentSetpoint tag. The controller switches to automatic mode following successful pretuning and to "Inactive" mode following unsuccessful pretuning. The pretuning phase is indicated with the SUT.State tag.

State / Retain.Mode	Description
2	Fine tuning Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are tuned based on the amplitude and frequency of this oscillation. The differences between the process response during pretuning and fine tuning are analyzed. All PID parameters are recalculated from the results. PID parameters from fine tuning usually have better master control and disturbance characteristics than PID parameters from pretuning. PID_3Step automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value. The PID parameters are backed up before fine tuning. They can be reactivated with Config.LoadBackUp. The setpoint is frozen in the CurrentSetpoint tag. Requirements for fine tuning: • The motor transition time has been configured or measured. • The setpoint and the process value lie within the configured limits. • ManualEnable = FALSE • Automatic (State = 3), inactive (State = 0) or manual (State = 4) mode Fine tuning proceeds as follows when started from: • Automatic mode (State = 3) Start fine tuning from automatic mode if you wish to improve the existing PID parameters through tuning. PID_3Step controls the system using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start. • Inactive (State = 0) or manual mode (State = 4) Pretuning is always started first. The determined PID parameters will be used for control until the control loop has stabilized and the requirements for fine tuning have been met. If PIDSelfTune.TIR.RunIn = TRUE, pretuning is skipped and an attempt is made to reach the setpoint with the minimum or maximum output value. This can produce increased overshoot. Fine tuning then starts automatically. The controller switches to automatic mode following successful fine tuning. If fine tuning was not successful, the controller switches to "Inactive" mode. The fine tuning phase is indicated with the TIR.State tag.
3	Automatic mode In automatic mode, PID_3Step controls the controlled system in accordance with the parameters specified. The controller switches to automatic mode if one the following requirements is fulfilled: • Pretuning successfully completed • Fine tuning successfully completed • Changing the Retain.Mode tag to the value 3. When the CPU is switched on or switches from Stop to RUN mode, PID_3Step starts in the most recently active operating mode. To leave PID_3Step in "Inactive" mode, set RunModeByStartup = FALSE. The ActivateRecoverMode tag is taken into consideration in automatic mode.

State / Retain.Mode	Description
4	Manual mode In manual mode, you specify manual output values in the Manual_UP and Manual_DN parameters or ManualValue parameter. Whether or not the actuator can be moved to the output value in the event of an error is described in the ErrorBits parameter. This operating mode is enabled if Retain.Mode = 4, or on a rising edge at ManualEnable. If ManualEnable changes to TRUE, only State changes. Retain.Mode retains its current value. On a falling edge at ManualEnable, PID_3Step returns to the previous operating mode. The switchover to automatic mode is bumpless. **PID_3Step V1.1** Manual mode is always possible in the event of an error. **PID_3Step V1.0** Manual mode is dependent on the ActivateRecoverMode tag in the event of an error.
5	Approach substitute output value This operating mode is activated in the event of an error or when Reset = TRUE if Errorbehaviour = 1 and ActivateRecoverMode = FALSE.. PID_3Step moves the actuator to the substitute output value and then switches to "Inactive" mode.
6	Transition time measurement The time that the motor needs to completely open the valve from the closed condition is determined. This operating mode is activated when GetTransitTime.Start = TRUE is set. If endstop signals are used to measure the transition time, the valve will be opened completely from its current position, closed completely, and opened completely again. If GetTransitTime.InvertDirection = TRUE, this behavior is inverted. If position feedback is used to measure the transition time, the actuator will be moved from its current position to a target position. The output value limits are not taken into consideration during the transition time measurement. The actuator can travel to the high or the low endstop.
7	Error monitoring The control algorithm is switched off and no longer changes the valve position. This operating mode is activated instead of "Inactive" mode in the event of an error. All the following conditions must be met: • Mode = 3 (automatic mode) • Errorbehaviour = 0 • ActivateRecoverMode = TRUE • One or more errors have occurred in which ActivateRecoverMode (Page 5027) is effective. As soon as the errors are no longer pending, PID_3Step switches back to automatic mode.
8	Approach substitute output value with error monitoring This operating mode is activated instead of "Approach substitute output value" mode in the event of an error. PID_3Step moves the actuator to the substitute output value and then switches to "Error monitoring" mode. All the following conditions must be met: • Mode = 3 (automatic mode) • Errorbehaviour = 1 • ActivateRecoverMode = TRUE • One or more errors have occurred in which ActivateRecoverMode (Page 5027) is effective. As soon as the errors are no longer pending, PID_3Step switches back to automatic mode.

Automatic switchover of operating mode during commissioning

PID_3Step automatically switches the operating mode in the event of an error. The following diagram illustrates the influence of ErrorBehaviour on the switchover of operating mode from transition time measurement, pretuning, and fine tuning modes.

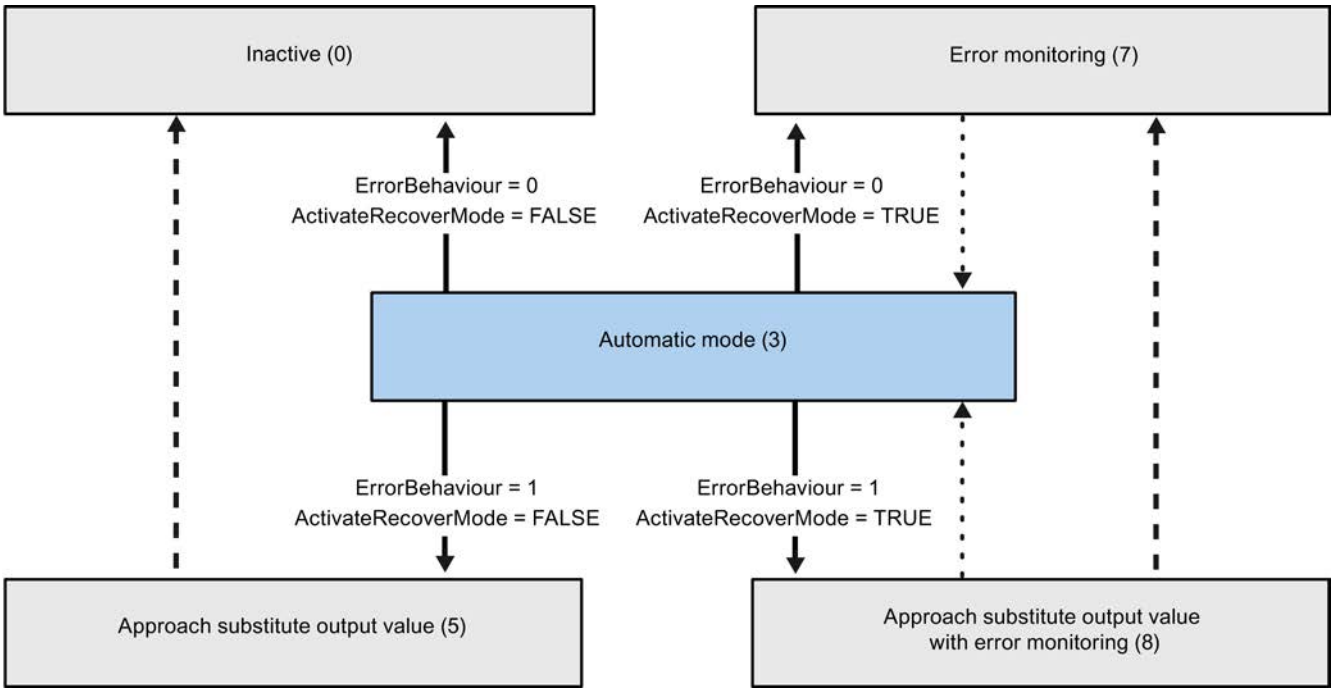


Automatic switchover of operating mode in the event of an error

Automatic switchover of operating mode once the current operation has been completed.

Automatic switchover of operating mode in automatic mode (PID_3Step V1.1)

PID_3Step automatically switches the operating mode in the event of an error. The following diagram illustrates the influence of ErrorBehaviour and ActivateRecoverMode on this switchover of operating mode.

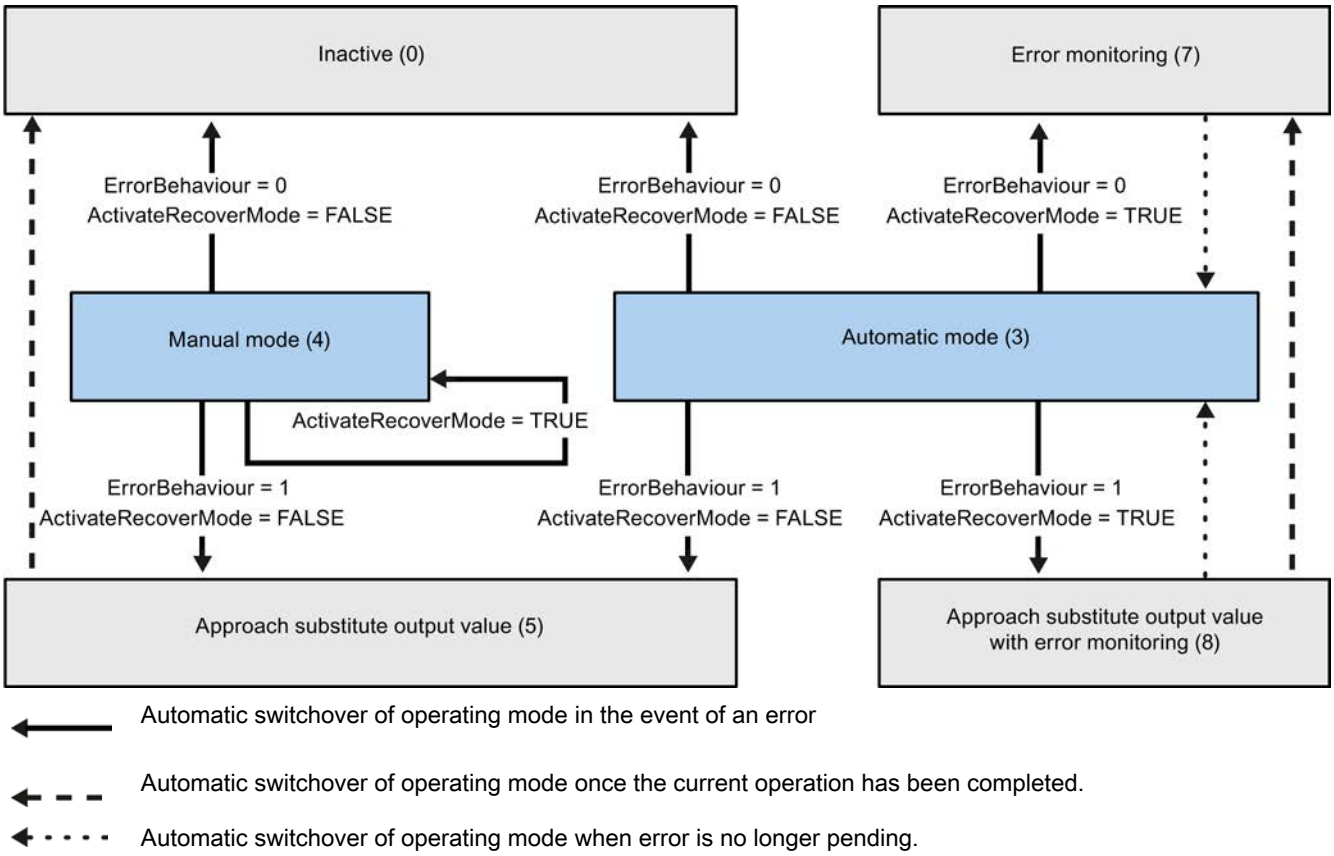


Automatic switchover of operating mode in the event of an error

Automatic switchover of operating mode once the current operation has been completed.

Automatic switchover of operating mode in automatic and manual modes (PID_3Step V1.0)

PID_3Step automatically switches the operating mode in the event of an error. The following diagram illustrates the influence of ErrorBehaviour and ActivateRecoverMode on this switchover of operating mode.



Automatic switchover of operating mode in the event of an error

Automatic switchover of operating mode once the current operation has been completed.

Automatic switchover of operating mode when error is no longer pending.

See also

Tag ActivateRecoverMode V1 (Page 5027)

Parameter ErrorBits V1 (Page 5026)

Parameter ErrorBits V1

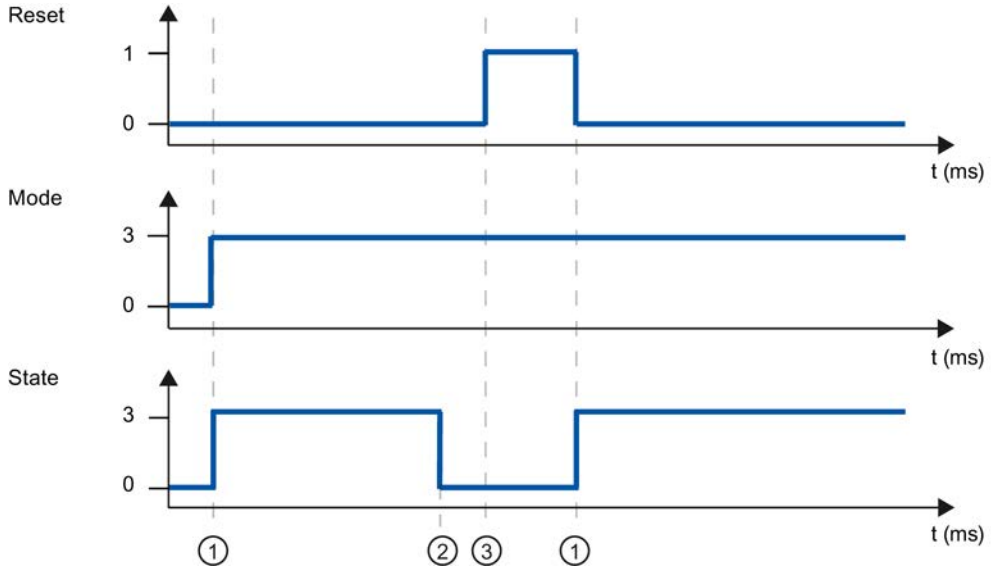
If several errors are pending simultaneously, the values of the error codes are displayed with binary addition. The display of error code 0003, for example, indicates that the errors 0001 and 0002 are pending simultaneously.

ErrorBits (DW#16#...)	Description
0000	There is no error.
0001	The "Input" parameter is outside the process value limits. • Input > Config.InputUpperLimit or • Input < Config.InputLowerLimit If ActivateRecoverMode = TRUE and ErrorBehaviour = 1, the actuator moves to the substitute output value. If ActivateRecoverMode = TRUE and ErrorBehaviour = 0, the actuator stops in its current position. If ActivateRecoverMode = FALSE, the actuator stops in its current position. PID_3Step V1.1 You can move the actuator in manual mode. PID_3Step V1.0 Manual mode is not possible in this state. You cannot move the actuator again until you eliminate the error.
0002	Invalid value at "Input_PER" parameter. Check whether an error is pending at the analog input. If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode.
0004	Error during fine tuning. Oscillation of the process value could not be maintained.
0020	Pretuning is not permitted in automatic mode or during fine tuning.
0080	Error during pretuning. Incorrect configuration of output value limits. Check whether the limits of the output value are configured correctly and match the control logic.
0100	Error during fine tuning resulted in invalid parameters.
0200	Invalid value at "Input" parameter: Value has an invalid number format. If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode.
0400	Calculation of output value failed. Check the PID parameters.
0800	Sampling time error: PID_3Step is not called within the sampling time of the cyclic interrupt OB. If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode.
1000	Invalid value at "Setpoint" parameter: Value has an invalid number format. If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode.
2000	Invalid value at Feedback_PER parameter. Check whether an error is pending at the analog input. The actuator cannot be moved to the substitute output value and remains in its current position. Manual mode is not possible in this state. You must deactivate position feedback (Config. FeedbackOn = FALSE) to move the actuator from this state. If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode.

ErrorBits (DW#16#...)	Description
4000	Invalid value at Feedback parameter. Value has an invalid number format. The actuator cannot be moved to the substitute output value and remains in its current position. Manual mode is not possible in this state. You must deactivate position feedback (Config. FeedbackOn = FALSE) to move the actuator from this state. If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode.
8000	Error during digital position feedback. Actuator_H = TRUE and Actuator_L = TRUE. The actuator cannot be moved to the substitute output value and remains in its current position. Manual mode is not possible in this state. In order to move the actuator from this state, you must deactivate the "Actuator endstop" (Config.ActuatorEndStopOn = FALSE). If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_3Step switches back to automatic mode.

Parameter Reset V1

A rising edge at Reset resets the errors and warnings and clears the integral action. A falling edge at Reset triggers a change to the most recently active operating mode.



① Activation
② Error
③ Reset

Tag ActivateRecoverMode V1

The effect of the ActivateRecoverMode variable depends on the version of the PID_3Step.

Behavior in version 1.1

The ActivateRecoverMode variable determines the behavior in the event of an error in automatic mode. ActivateRecoverMode is not effective during pretuning, fine tuning and transition time measurement.

ActivateRecover Mode	Description
FALSE	In the event of an error, PID_3Step switches to "Inactive" or "Approach substitute output value" operating mode. The controller is activated by a reset or a change in Retain.Mode.
TRUE	If errors occur frequently in automatic mode, this setting has a negative effect on the control response. In this case, check the ErrorBits parameter and eliminate the cause of the error. If one or more errors occur, PID_3Step switches to "Approach substitute output value with error monitoring" or "Error monitoring" mode: • 0002h: Invalid value at parameter Input_PER. • 0200h: Invalid value at parameter Input. • 0800h: Sampling time error • 1000h: Invalid value at parameter Setpoint. • 2000h: Invalid value at parameter Feedback_PER. • 4000h: Invalid value at parameter Feedback. • 8000h: Error in digital position feedback. With errors 2000h, 4000h and 8000h, PID_3Step **cannot** approach the configured substitute output value. As soon as the errors are no longer pending, PID_3Step switches back to automatic mode.

Behavior in version 1.0

The ActivateRecoverMode variable determines the behavior in the event of an error in automatic and manual mode. ActivateRecoverMode is not effective during pretuning, fine tuning and transition time measurement.

ActivateRecover Mode	Description
FALSE	In the event of an error, PID_3Step switches to "Inactive" or "Approach substitute output value" operating mode. The controller is activated by a reset or a change in Retain.Mode.
TRUE	**Errors in automatic mode** If errors occur frequently in automatic mode, this setting has a negative effect on the control response. In this case, check the ErrorBits parameter and eliminate the cause of the error. If one or more errors occur, PID_3Step switches to "Approach substitute output value with error monitoring" or "Error monitoring" mode: • 0002h: Invalid value at parameter Input_PER. • 0200h: Invalid value at parameter Input. • 0800h: Sampling time error • 1000h: Invalid value at parameter Setpoint. • 2000h: Invalid value at parameter Feedback_PER. • 4000h: Invalid value at parameter Feedback. • 8000h: Error in digital position feedback. With errors 2000h, 4000h and 8000h, PID_3Step **cannot** approach the configured substitute output value. As soon as the errors are no longer pending, PID_3Step switches back to automatic mode. **Errors in manual mode** If one or more of the following errors occur, PID_3Step stays in manual mode: • 0002h: Invalid value at parameter Input_PER. • 0200h: Invalid value at parameter Input. • 0800h: Sampling time error • 1000h: Invalid value at parameter Setpoint. • 2000h: Invalid value at parameter Feedback_PER. • 4000h: Invalid value at parameter Feedback. • 8000h: Error in digital position feedback. With errors 2000h, 4000h and 8000h, you **cannot** move the valve to a suitable position.

See also

PID_3Step V1 static tags (Page 5012)

Parameter State and Retain.Mode V1 (Page 5019)

Tag Warning V1

If several warnings are pending simultaneously, their values are displayed with binary addition. The display of warning 0003, for example, indicates that the warnings 0001 and 0002 are pending simultaneously.

Warning (DW#16#...)	Description
0000	No warning pending.
0001	The point of inflection was not found during pretuning.
0002	Oscillation increased during fine tuning.
0004	The setpoint was limited to the configured limits.
0008	Not all the necessary controlled system properties were defined for the selected method of calculation. The PID parameters were instead calculated using the TuneRuleTIR = 3 method.
0010	The operating mode could not be changed because ManualEnable = TRUE.
0020	The cycle time of the calling OB limits the sampling time of the PID algorithm. Improve results by using shorter OB cycle times.
0040	The process value exceeded one of its warning limits.
0080	Invalid value at Retain.Mode. The operating mode is not switched.
0100	The manual value was limited to the limits of the controller output.
0200	The rule used for tuning produces an incorrect result, or is not supported.
0400	Method selected for transition time measurement not suitable for actuator. The transition time cannot be measured because the actuator settings do not match the selected measuring method.
0800	The difference between the current position and the new output value is too small for transition time measurement. This can produce incorrect results. The difference between the current output value and new output value must be at least 50% of the entire control range.
1000	The substitute output value cannot be reached because it is outside the output value limits.

The following warnings are deleted as soon as the cause is eliminated:

- 0004
- 0020
- 0040
- 0100

All other warnings are cleared with a rising edge at Reset.

Tag SUT.State V1

SUT.State	Name	Description
0	SUT_INIT	Initialize pretuning
50	SUT_TPDN	Determine start position without position feedback
100	SUT_STDABW	Calculate the standard deviation
200	SUT_GET_POI	Find the point of inflection

SUT.State	Name	Description
300	SUT_GET_RISETM	Determine the rise time
9900	SUT_IO	Pretuning successful
1	SUT_NIO	Pretuning not successful

Tag TIR.State V1

TIR.State	Name	Description
-100	TIR_FIRST_SUT	Fine tuning is not possible. Pretuning will be executed first.
0	TIR_INIT	Initialize fine tuning
200	TIR_STDABW	Calculate the standard deviation
300	TIR_RUN_IN	Attempt to reach the setpoint with the maximum or minimum output value
400	TIR_CTRLN	Attempt to reach the setpoint with the existing PID parameters (if pretuning has been successful)
500	TIR_OSZIL	Determine oscillation and calculate parameters
9900	TIR_IO	Fine tuning successful
1	TIR_NIO	Fine tuning not successful

PID basic functions

Differences compared to CONT_C S7-300/400

The CONT_C instruction is created with optimized block access.

All parameters are retentive. The retentivity cannot be changed. The actual values of retentive tags are only updated when you download CONT_C completely.

Downloading technology objects to device (Page 9536)

Otherwise, the behavior of CONT_C on an S7-1500 CPU is exactly the same as on an S7-300, S7-400 CPU.

Description CONT_C (Page 6793)

How CONT_C works (Page 6794)

CONT_C block diagram (Page 6796)

Input parameter CONT_C (Page 6796)

Output parameters CONT_C (Page 6798)

Differences compared to CONT_S S7-300/400

The CONT_S instruction is created with optimized block access. S

All parameters are retentive. The retentivity cannot be changed. The actual values of retentive tags are only updated when you download CONT_S completely.

Downloading technology objects to device (Page 9536)

Otherwise, the behavior of CONT_S on an S7-1500 CPU is exactly the same as on an S7-300, S7-400 CPU.

Description CONT_S (Page 6799)

Mode of operation CONT_S (Page 6799)

Block diagram CONT_S (Page 6801)

Input parameters CONT_S (Page 6801)

Output parameters CONT_S (Page 6803)

Differences compared to PULSEGEN S7-300/400

The PULSEGEN instruction is created with optimized block access.

All parameters are retentive. The retentivity cannot be changed. The actual values of retentive tags are only updated when you download PULSEGEN completely.

Downloading technology objects to device (Page 9536)

The behavior of PULSEGEN on an S7-1500 CPU is exactly the same as on an S7-300, S7-400 CPU.

Description PULSEGEN (Page 6803)

Mode of operation PULSEGEN (Page 6804)

Mode of operation PULSEGEN (Page 6807)

Three-step control (Page 6807)

Two-step control (Page 6809)

Input parameters PULSEGEN (Page 6811)

Output parameter PULSEGEN (Page 6812)

Differences compared to TCONT_CP S7-300/400

The TCONT_CP instruction is created with optimized block access.

All parameters are retentive. The retentivity cannot be changed. The actual values of retentive tags are only updated when you download TCONT_CP completely.

Downloading technology objects to device (Page 9536)

Otherwise, the behavior of TCONT_CP on an S7-1500 CPU is exactly the same as on an S7-300, S7-400 CPU.

Description TCONT_CP (Page 6812)

Mode of operation TCONT_CP (Page 6813)

Operating principle of the pulse generator (Page 6822)

Block diagram TCONT_CP (Page 6824)

Input parameters TCONT_CP (Page 6825)

Output parameters TCONT_CP (Page 6826)

In/out parameters TCONT_CP (Page 6826)

Static variables TCONT_CP (Page 6827)

Parameter STATUS_H (Page 6831)

Parameters STATUS_D (Page 6832)

Differences compared to TCONT_S S7-300/400

The TCONT_S instruction is created with optimized block access.

All parameters are retentive. The retentivity cannot be changed. The actual values of retentive tags are only updated when you download TCONT_S completely.

Downloading technology objects to device (Page 9536)

Otherwise, the behavior of TCONT_S on an S7-1500 CPU is exactly the same as on an S7-300, S7-400 CPU.

Description TCONT_S (Page 6833)

Mode of operation TCONT_S (Page 6834)

Block diagram TCONT_S (Page 6837)

Input paramters TCONT_S (Page 6838)

Output parameters TCONT_S (Page 6839)

In/out parameters TCONT_S (Page 6839)

Static variables TCONT_S (Page 6839)

9.8.1.5 Communication

S7 communication

Data consistency

Definition

A data block which cannot be modified by concurrent processes is called consistent data area. A data block which is larger than the consistent data area can thus be falsified as a whole during transmission. This means that a data block which belongs together and which is larger than consistent data area can consist in part of new and of old consistent data at the same time.

Example

An inconsistency can occur when an instruction for communication is interrupted, for example, by a hardware interrupt OB with higher priority. If the user program in this OB now changes

the data that has already been partly processed by the instruction, the transferred data originates:

- In part from the time before the hardware interrupt was processed
- And in part from the time after the hardware interrupt was processed

This means that these data are inconsistent (not coherent).

Ensuring data consistency

If the communication process can be interrupted by an interrupt OB, you must ensure that the data is transferred consistently. Make sure that only an image of the data and not the data to be transferred is not changed directly by the interrupt OB. Copy the image of the data to the transfer area of the communication instruction prior to the next data transfer.

- If the user program contains a communication instruction that accesses common data, access to this data area can be coordinated, for example, by means of the DONE parameter. The data consistency of the communication areas which are transferred locally with a communication instruction can therefore be ensured in the user program.
- In the case of S7 communication instructions "PUT (Page 5040)"/"GET (Page 5038)", the size of the consistent data areas must already be taken into consideration during programming or configuration, because a communication block is not available in the user program of the target device (server) to synchronize communication data to the user program.
- For the S7-300 and C7-300 (exception: CPU 318-2 DP) the communication data are copied consistently into the user memory in blocks of 32 bytes in the cycle control point of the operating system. Data consistency is not guaranteed for larger data areas. If a defined data consistency is required, the communication data in the user program may not exceed 32 bytes (maximum of 8 bytes, depending on the version).
- In the S7-400 and S7-1500, on the other hand, the communication data in 462-byte blocks is not processed at the cycle control point, but in fixed time slices during the program cycle. The consistency of a tag is ensured by the system. These communication areas can then be accessed consistently using the "PUT (Page 5040)" / "GET (Page 5038)" instructions or when reading/writing tags, for example by an OP or an OS.

Note

Additional information on data consistency is provided in the description of the specific instructions.

Effect on interrupt reaction times

The interrupt reaction times of the CPU are slightly extended by copying the data. The greater the quantity of data which must be transferred with absolute consistency, the longer the interrupt reaction time of a system.

Common parameters of instructions for S7 communication

Classification

The parameters of the instructions for S7 communication can be divided into the following five categories according to their functions:

1. Control parameters are used to activate an instruction.
2. Addressing parameters are used to address the remote communication partner.
3. Send parameters point to the data areas that are to be sent to the remote partner.
4. Receive parameters point to the data areas where the data received from remote partners will be entered.
5. Status parameters are used to monitor whether the instruction has completed its task without error or for the analysis of any errors that have occurred.

Control parameter

Data exchange will only be activated if the appropriate control parameters have a defined value (for example, are set) when the instruction is called or when the value has undergone a specific change since the previous call (for example, a positive edge).

Addressing parameters

Parameter	Description
ID	Reference to the local connection description (specified by the configuration of the connection).
R_ID	With the R_ID parameter, you specify that a send and a receive instruction belong together: The R_ID parameter must match in the instruction at the sending end and the instruction at the receiving end. This allows the communication of several instruction pairs via the same logic connection. • R_ID must be specified in the form DW#16#wxyzWXYZ. • The instruction pairs of a logical connection specified in R_ID must be unique for this connection.

Note

Addressing parameters ID and R_ID

You can reassign the addressing parameters ID and R_ID during runtime. The new parameters are validated with each new job after the previous job has been closed.

You can use the following options to reduce the number of instance DBs and therefore the work memory required:

1. With tag IDs, you can use several connections via one data instance block.
2. With tag R_IDs, you can define several pairs of sending and receiving instructions for one job with a single instance.
3. You can combine cases 1 and 2.

Please note that the new parameters are valid after the last job is executed. When you activate the sending operation, the R_ID parameter in the instruction at the sending end must match its counterpart at the receiving end.

Status parameter

With the status parameters, you monitor whether the instruction has completed its task correctly or whether it is still active. The status parameters also indicate errors.

Note

The status parameters are valid for one cycle only, namely from the first command following the call until the next call. As a result, you must evaluate these parameters after each instruction cycle.

Send and receive parameters

For communication instructions configured at both ends

- The number of the SD_i and RD_i parameters used must match at the send and receive end.
- The data types of the SD_i and RD_i parameters that belong together must match at the send and receive end.
- The amount of data to be sent according to the SD_i parameter must not exceed the range made available by the corresponding RD_i (does not apply to "BSEND (Page 5046)" / "BRCV (Page 5048)"). The RD_i parameters must (with exception of "BSEND"/"BRCV") have the identical data size.

If you do not keep to the rules above, this is indicated by ERROR = 1 and STATUS = 4.

Note

Supplying the send and receive parameters

With the VARIANT data type, send and receive parameters must always be supplied any time a communication instruction is called. It is not possible, for example, to supply the send parameters of the communication instructions at startup and to only trigger the send job in cyclic operation.

User data size

With the "USEND (Page 5043)", "URCV (Page 5044)", "GET (Page 5038)" and "PUT (Page 5040)" instructions, the amount of data to be transferred must not exceed a defined user data size. The maximum user data size depends on:

- The instruction used
- The communication partner.

The guaranteed minimum size of the user data for an instruction with 1-4 tags is listed in the following table:

Instruction	Partner: S7-300	Partner: S7-400	Partner: S7-1200	Partner: S7-1500
PUT / GET	160 bytes	400 bytes	160 bytes	880 bytes
USEND / URCV	160 bytes	440 bytes	-	920 bytes
BSEND / BRCV	32768/65534 bytes	65534 bytes	-	• 65534 bytes with standard access • 65535 bytes with optimized access

Further information on the user data size can be found in the technical data of the respective CPU.

Exact user data size

If the user data size specified above is insufficient you can determine the maximum byte length of the user data as follows:

First read the data block size valid for communication from the following table:

Local CPU	Remote CPU	Data block size in bytes
S7-1200	Any	240
S7-1500	S7-300	240
	S7-400	480
	S7-1200	240
	S7-1500	960

Use this value in the following table to read the maximum possible user data length in bytes as total of the parameters used. It applies for even lengths of the areas SD_i, RD_i, and ADDR_i.

For each range of uneven length the maximum possible user data length is reduced by one byte.

		Number of SD_i, RD_i, ADDR_i parameters used			
Data block size	Instruction	1	2	3	4
240 (S7-300)	PUT/GET/ USEND	160	-	-	-
240 (S7-300 via integrated interface)	PUT	212	-	-	-
	GET	222	-	-	-
	USEND	212	-	-	-
240 (S7-400)	PUT	212	196	180	164
	GET	222	218	214	210
	USEND	212	-	-	-
480 (S7-400)	PUT	452	436	420	404
	GET	462	458	454	450
	USEND	452	448	444	440
240 (S7-1200)	PUT	212	196	180	164
	GET	222	218	214	210
960 (S7-1500)	PUT	932	916	900	884
	GET	942	938	934	930
	USEND	932	928	924	920

GET: Read data from a remote CPU

Description

With the instruction "GET", you can read data from a remote CPU.

The instruction is started on a positive edge at control input REQ:

- The relevant pointers to the areas to be read out (ADDR_i) are then sent to the partner CPU. The partner CPU can be in RUN or STOP mode.
- The partner CPU returns the data:
  - If the reply exceeds the maximum user data length, this is displayed with error code "2" at the STATUS parameter.
  - The received data is copied to the configured receive areas (RD_i) at the next call.
- Completion of this action is indicated by the status parameter NDR having the value "1".

Reading can only be activated again after the previous reading process has been completed. Errors and warnings are output via ERROR and STATUS if access problems occurred while the data was being read or if the data type check results in an error.

Changes in the data areas addressed on the partner CPU are not registered by the "GET" instruction.

Requirements for using the instruction

- The "Permit access with PUT/GET communication from remote partner" function was activated in the properties of the partner CPU under "Protection".
- The blocks which you access with the "GET" instruction were created with the access type "standard".
- Make sure that the areas defined with the parameters ADDR_i and SD_i match in terms of number, length and data type.
- The area to be read (ADDR_i parameter) cannot be larger than the area for data storage (RD_i parameter).

Parameters

The following table shows the parameters of the "GET" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Control parameter request, activates the data exchange on a positive edge.
ID	Input	WORD	I, Q, M, D, L or constant	Addressing parameters for specifying the connection to the partner CPU.
NDR	Output	BOOL	I, Q, M, D, L	Status parameter NDR: • 0: Job not yet started or still running. • 1: Job successfully completed.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: – 0000H: Neither warning nor error – <> 0000H: Warning, STATUS supplies detailed information. • ERROR=1 An error has occurred. STATUS supplies detailed information on the type of error.
STATUS	Output	WORD	I, Q, M, D, L	
ADDR_1	InOut	REMOTE	I, Q, M, D	Pointers to the areas on the partner CPU that are to be read. When the REMOTE pointer accesses a DB, the DB must always be specified. Example: P#DB10.DBX5.0 Byte 10.
ADDR_2	InOut	REMOTE		
ADDR_3	InOut	REMOTE		
ADDR_4	InOut	REMOTE		
RD_1	InOut	VARIANT	I, Q, M, D, L	Pointers to the areas on the local CPU in which the read data is entered.
RD_2	InOut	VARIANT		
RD_3	InOut	VARIANT		
RD_4	InOut	VARIANT		

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameters ERROR and STATUS

The following table contains all specific error information for the "GET" instruction that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Warning: New job not active because the previous job is still busy.
0	25	Communication has started. The job is being processed.
1	1	Communication problems, for example • Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode) • Connection to partner not yet established
1	2	• Negative acknowledgment from the partner device. The function cannot be executed. • Response from the remote station exceeds the maximum user data length (see: Common parameters of instructions for S7 communication (Page 5035)). • Access protection is activated on the partner CPU. Deactivate access protection in the CPU settings.
1	4	Error in the pointers to the data storage RD_i: • Data types of the parameters RD_i and ADDR_i are not compatible with each other. • The length of the area RD_i is smaller than the length of the data of the ADDR_i parameter that is to be read.
1	8	Access error on the partner CPU.
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	20	• Maximum number of parallel jobs exceeded. • The job is currently being processed in a priority class with low priority (first call).

Note

Data consistency

Data is received consistently if you read the part of the receive area RD_i currently being used completely before initiating another job.

PUT: Write data to a remote CPU

Description

You can write data to a remote CPU with the instruction "PUT".

The instruction is started on a positive edge at control input REQ:

- The pointers to the areas to be written (ADDR_i) and the data (SD_i) are then sent to the partner CPU. The partner CPU can be in RUN or STOP mode.
- The data to be sent is copied from the configured send areas ((SD_i). The partner CPU saves the sent data under the addresses supplied with the data and returns an execution acknowledgment.
- If no errors occur, this is indicated at the next instruction call with status parameter DONE = "1". The writing process can only be activated again after the last job is complete.

Errors and warnings are output via ERROR and STATUS if access problems occurred while the data was being written or if the execution check results in an error.

Requirements for using the instruction

- The "Permit access with PUT/GET communication from remote partner" function was activated in the properties of the partner CPU under "Protection".
- The blocks which you access with the "PUT" instruction were created with the access type "standard".
- Make sure that the areas defined with the parameters ADDR_i and SD_i match in terms of number, length and data type.
- The area to be written (ADDR_i parameter) must be as large as the send area (SD_i parameter).

Parameters

The following table shows the parameters of the "PUT" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Control parameter request, activates the data exchange on a positive edge.
ID	Input	WORD	I, Q, M, D, L or constant	Addressing parameters for specifying the connection to the partner CPU.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: • 0: Job not yet started or still executing • 1: Job executed without errors.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: – 0000H: Neither warning nor error – <> 0000H: Warning, STATUS supplies detailed information. • ERROR=1 An error has occurred. STATUS supplies detailed information on the type of error.
STATUS	Output	WORD	I, Q, M, D, L	
ADDR_1	InOut	REMOTE	I, Q, M, D	Pointers to the areas on the partner CPU to which the data will be written.
ADDR_2	InOut	REMOTE		
ADDR_3	InOut	REMOTE		

Parameter	Declaration	Data type	Memory area	Description
ADDR_4	InOut	REMOTE		When the REMOTE pointer accesses a DB, the DB must always be specified. Example: P#DB10.DBX5.0 Byte 10. When transferring data structures (e.g., Struct) the data type CHAR must be used at the ADDR_i parameters.
SD_1	InOut	VARIANT	I, Q, M, D, L	Pointers to the areas on the local CPU which contain the data to be sent. Only the data types BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL are permitted. When transferring data structures (e.g., Struct) the data type CHAR must be used at the SD_i parameters.
SD_2	InOut	VARIANT		
SD_3	InOut	VARIANT		
SD_4	InOut	VARIANT		

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameters ERROR and STATUS

The following table contains all specific error information for the "PUT" instruction that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Warning: New job not active because the previous job is still busy.
0	25	Communication has started. The job is being processed.
1	1	Communication problems, for example • Connection description not loaded (local or remote). • Connection interrupted (for example, cable, CPU off, CP in STOP mode). • Connection to partner not yet established.
1	2	• Negative acknowledgment of partner CPU. The function cannot be executed. • Access on the partner CPU was not granted. Activate the access in the CPU settings.
1	4	Error in the pointers to the data storage: • Data types of the parameters SD_i and ADDR_i are not compatible with each other. • The length of the area SD_i is greater than the length of the data of the ADDR_i parameter that is to be written. • Not possible to access SD_i. • Maximum user data size exceeded. • Number of the parameters SD_i and ADDR_i does not match.
1	8	Access error with the partner CPU (e.g. DB not loaded or write-protected).
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	20	• Maximum number of parallel jobs exceeded. • The job is currently being processed in a priority class with low priority (first call).

Data consistency

When a send operation is activated (positive edge at REQ), the data to be sent from the send area SD_i is copied from the user program. After the block call, you can write to these areas without corrupting the current send data.

Note

The send operation is only complete when the DONE status parameter has the value "1".

Other

USEND: Send data uncoordinated

Description

The "USEND" instruction sends data to a remote partner instruction of type "URCV (Page 5044)". The sending process is carried out without coordination with the partner instruction. This means that the data transfer is carried out without acknowledgment by the partner instruction.

When a send operation is activated (positive edge at REQ), the data to be sent from the send areas SD_i are copied from the user program. After the instruction call, you can write to these send areas again without corrupting the current send data.

Successful completion of the send operation is indicated by the value of the DONE status parameter being set to "1".

Parameters

The following table shows the parameters of the "USEND" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Control parameter request, activates the data exchange on a positive edge.
ID	Input	CONN_PRG	I, Q, M, D, L, P or constant	Addressing parameters for specifying the connection to the partner CPU.
R_ID	Input	CONN_R_ID	I, Q, M, D, L or constant	Addressing parameter R_ID for defining the instruction pairs "USEND" and "URCV". See also: Common parameters of instructions for S7 communication (Page 5035)
DONE	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Job not yet started or still executing. • 1: Job executed without errors.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Neither warning nor error. • 1: An error has occurred. STATUS supplies detailed information on the type of error.

Parameter	Declaration	Data type	Memory area	Description
STATUS	Output	WORD	I, Q, M, D, L	Status parameter See "ERROR and STATUS parameters" table.
SD_i (1≤ i ≤4)	InOut	VARIANT	I, Q, M, D	Pointer on the i-th send area. Only the BOOL data types are permitted (not permitted: bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL or STRUCT. The maximum user data size for SD_i parameters depends on the partner CPU ("URCV" instruction) and on the number of parameters used. Additional information is available in: Common parameters of instructions for S7 communication (Page 5035)

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameters ERROR and STATUS

ERROR	STATUS (decimal)	Explanation
0	11	Warning: New job not active because the previous job is still busy.
0	25	Communication has started. The job is being processed.
1	1	Communication problem has occurred. Possible causes: • Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode) • Connection to partner not yet established
1	4	• Errors in the send area pointers SD_i relating to the data length or the data type. • Maximum user data length was exceeded.
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	18	Value at R_ID parameter already exists in the connection specified at the ID parameter (R_ID value must be unique for the connection).
1	20	• Maximum number of parallel jobs exceeded. • The job is currently being processed in a priority class with lower priority (first call).

URCV: Receive data uncoordinated

Description

The "URCV" instruction receives data asynchronously from a remote partner instruction of type "USEND (Page 5043)" and copies this data into the configured receive areas.

The instruction is ready to receive if there is a logical 1 at the EN_R input. An active job can be canceled with EN_R=0.

The receive data areas are referenced by the parameters RD_1 to RD_4. Make sure that the areas defined by the parameters RD_i / RD_1 and SD_i / SD_1 (with the associated partner instruction "USEND (Page 5043)") match in terms of number and length.

Successful completion of the copy operation is indicated when the value of the NDR status parameter is set to logical "1". Once the status parameter NDR has changed to "1", there will be new receive data in your receive areas (RD_i). A new block call may cause these data to be overwritten with new receive data. If you want to prevent this, call "URCV" (for example, during cyclic block processing) with the value 0 at EN_R until you have finished processing the received data.

Parameters

The following table shows the parameters of the "URCV" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L or constant	Control parameter enabled to receive, signals ready to receive if the input is set.
ID	Input	CONN_PRG	I, Q, M, D, L, P or constant	Addressing parameters for specifying the connection to the partner CPU.
R_ID	Input	CONN_R_ID	I, Q, M, D, L or constant	Addressing parameter for defining the instruction pairs "USEND" and "URCV". See also: Common parameters of instructions for S7 communication (Page 5035)
NDR	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Job not yet started or still running. • 1: Job successfully completed.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Neither warning nor error • 1: An error has occurred. STATUS supplies detailed information on the type of error.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter See "ERROR and STATUS parameters" table.
RD_i (1≤ i ≤4)	InOut	VARIANT	I, Q, M, D	Pointer to the i-th receive area: Only the BOOL data types are permitted (not permitted: bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL or STRUCT. Additional information is available in: Common parameters of instructions for S7 communication (Page 5035)

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameters ERROR and STATUS

ERROR	STATUS (decimal)	Explanation
0	9	Warning: older received data were overwritten by newer received data.
0	11	Warning: The received data is already being processed in a priority class with lower priority (error can occur when copying data to the receive area).
0	25	Communication has started. The job is being processed.
1	1	Communication problem has occurred. Possible causes: • Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode) • Connection to partner not yet established
1	4	Errors in the receive area pointers RD_i relating to the data length or the data type.
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	18	Value at R_ID parameter already exists in the connection specified at the ID parameter (R_ID value must be unique for the connection).
1	19	The associated "USEND (Page 5043)" instruction sends data faster than it can be copied to the receive areas by "URCV".
1	20	• Maximum number of parallel jobs exceeded. • The job is currently being processed in a priority class with lower priority (first call).

BSEND: Send data in segments

Description

The "BSEND" instruction sends data to a remote partner instruction of type "BRCV (Page 5048)". With this type of data transfer, more data can be transported between the communication partners than is possible with all other communication instructions for configured S7 connections. In each case, the maximum data volume amounts to 65534 bytes (standard access) or 65535 bytes (optimized access) for the integrated interface and for SIMATIC Net CP.

Functional description

You specify the instruction pair "BSEND" and "BRCV" with the input parameter R_ID. The R_ID parameter must be identical in the related instructions.

The send job is activated after the instruction is called and a positive edge is detected at control input REQ. After the call, "BSEND" is not processed in the background, which means the data can only be read within the user program.

The data area to be transmitted is segmented. Each segment is sent individually to the partner. Each segment is acknowledged by the partner after the application of this segment by "BRCV (Page 5048)". If the data is segmented, "BSEND" must be called multiple times until all segments have been transferred.

The data area of the data to be sent is specified by SD_1. To ensure data consistency, you may only write to the part of the currently used sending area SD_1 after the current send

operation is complete. This is the case when the value of the status parameter DONE changes to "1".

The length of the send data is specified by LEN based on the job. With LEN = "0", all data addressed with the SD_1 parameter is sent.

If there is a positive edge at control input R, the current sending process is canceled.

Due to the asynchronous data transfer, a new data transfer can only be initiated if the previous data has been fetched by a partner instruction call. When the data has been collected, the status parameter "NDR" is set in the partner instruction "BRCV".

Note

Migration of S7-400 user programs

An S7-400 CPU interprets the parameter SD_1 as pointer and not as data area.

With S7-1500, LEN may not exceed the area of SD1. This was permitted with S7-400. Recommendation: Use the maximum size for the LEN parameter (65534 bytes for the integrated interface) as size of the data area at the SD_1 parameter.

Parameters

The following table shows the parameters of the "BSEND" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Control parameter request, activates the data exchange at a positive edge
R	Input	BOOL	I, Q, M, D, L or constant	Control parameter reset, activates an abort of the current data exchange at a positive edge
ID	Input	CONN_PRG	I, Q, M, D, L, P or constant	Addressing parameters for specifying the connection to the partner CPU.
R_ID	Input	CONN_R_ID	I, Q, M, D, L or constant	Addressing parameter for defining the instruction pairs "BSEND" and "BRCV (Page 5048)". See also: Common parameters of instructions for S7 communication (Page 5035)
SD_1	InOut	VARIANT	I, Q, M, D	Pointer to send area
LEN	InOut	WORD	I, Q, M, D, L	Length of the block of data to be sent in bytes. With LEN = "0", all data is sent by SD_1.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Job not yet started or still executing. • 1: Job executed without errors.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Neither warning nor error. • 1: An error has occurred. STATUS supplies detailed information on the type of error.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter See "ERROR and STATUS parameters" table.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameters ERROR and STATUS

The following table contains all specific error information for "BSEND" that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Warning: New job not active because the previous job is still busy.
0	25	Communication has started. The job is being processed.
1	1	Communication problem has occurred. Possible causes: • Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode) • Connection to partner not yet established
1	2	Negative acknowledgment of partner instruction. The instruction cannot be executed.
1	3	R_ID is unknown on the connection specified by the ID or the receive block has not yet been called.
1	4	• Error in the data length or data type in the send area pointer SD_1. • Value for LEN is greater than area SD_1.
1	5	Reset request was executed.
1	6	Partner instruction is in DISABLED status (EN_R has the value "0"). Also check the input parameters of "BRCV (Page 5048)" for consistency with "BSEND".
1	7	"BRCV (Page 5048)" partner instruction not called since the last data transfer.
1	8	Access to remote object in the user memory was rejected: The destination area for the associated "BRCV (Page 5048)" is too small. ERROR = 1, STATUS = 4 or ERROR = 1, STATUS = 10 reported at the output parameters of "BRCV (Page 5048)".
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	18	R_ID already exists in the connection.
1	20	• Maximum number of parallel jobs exceeded. • The job is currently being processed in a priority class with lower priority (first call).

BRCV: Receive data in segments

Description

The "BRCV" instruction receives data from a remote partner instruction of type "BSEND (Page 5046)". The R_ID parameter must be identical in the related instructions.

The instruction is ready to receive data (STATUS = 25) after it is called with the value "1" at control input EN_R. An active job can be canceled with EN_R=0.

The maximum receive area is specified by RD_1. Data will be received consistently if you fully evaluate the part of the RD_1 receive area currently in use before calling the block again with value 1 at control input EN_R.

After each received data segment, an acknowledgment is sent to the partner instruction. If there are multiple segments it will be necessary to call "BRCV" several times until all of the segments have been received. Asynchronous data receipt is indicated by STATUS = 17. The amount of data currently received is indicated at the parameter LEN. The RD_1 parameter must remain constant during the operation.

Value "1" at the NDR status parameter indicates that all data segments have been received without error. The received data remains unchanged until the next call with EN_R=1.

Parameters

The following table shows the parameters of the "BRCV" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L or constant	Control parameter enabled to receive, signals ready to receive if the input is set.
ID	Input	CONN_PRG	I, Q, M, D, L, P or constant	Addressing parameters for specifying the connection to the partner CPU.
R_ID	Input	CONN_R_ID	I, Q, M, D, L or constant	Addressing parameter for defining the instruction pairs "BSEND (Page 5046)" and "BRCV". See also: Common parameters of instructions for S7 communication (Page 5035)
RD_1	InOut	VARIANT	I, Q, M, D	Pointer to the receive area.
LEN	InOut	WORD	I, Q, M, D, L	Length of the data already received in bytes.
NDR	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Job not yet started or still running. • 1: Job successfully completed.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter • 0: Neither warning nor error • 1: An error has occurred. STATUS supplies detailed information on the type of error.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter See "ERROR and STATUS parameters" table.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameters ERROR and STATUS

The following table contains all specific error information for "BRCV" that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	17	Warning: Instruction receives data asynchronously. The LEN parameter shows the amount of data already received in bytes.
0	25	Communication has started. The job is being processed.

ERROR	STATUS (decimal)	Explanation
1	1	Communication problem has occurred. Possible causes: • Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode) • Connection to partner not yet established
1	2	Function cannot be executed (protocol error)
1	4	Error in the receive area pointer RD_1 relating to the data length or data type. The block of data sent is longer than the receive area.
1	5	Reset request received, incomplete transfer.
1	8	Access error in associated "BSEND (Page 5046)": After the last valid data segment has been sent, ERROR = 1 and STATUS = 4 or ERROR = 1 and STATUS = 10 is reported.
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	18	Value at R_ID parameter already exists in the connection specified at the ID parameter (R_ID value must be unique for the connection).
1	20	• Maximum number of parallel jobs exceeded. • The job is currently being processed in a priority class with lower priority (first call).

Open User Communication

TSEND_C: Send data via Ethernet

TSEND_C: Send data via Ethernet

Description

The "TSEND_C" instruction sets up and establishes a TCP or ISO-on-TCP communication connection. Once the connection has been set up and established, it is automatically maintained and monitored by the CPU. The connection description specified at the CONNECT parameter is used to set up the communication connection.

The instruction is executed asynchronously and has the following functions:

- A communication connection is set up and established:
  The communication connection is set up and established with CONT=1. Once the connection is successfully established, the DONE parameter is set to "1" for one cycle. An existing connection is terminated and the connection which has been set up is removed when the CPU goes into STOP mode. To set up and establish the connection again, you must execute "TSEND_C" again. For information on the number of possible communication connections, refer to the technical specifications for your CPU.
- Sending data via an existing communication connection:
  You specify the send area with the DATA parameter. This includes the address and the length of the data to be sent. Do not use a data area with the data type BOOL or Array of BOOL at the DATA parameter. If you use purely symbolic values at the DATA parameter, the LEN parameter must have the value "0".

- The send job is executed when a rising edge is detected at the REQ parameter. With the LEN parameter, you specify the maximum number of bytes sent with a send job. When sending data (rising edge at the REQ parameter), the CONT parameter must have the value "1" in order to establish or maintain a connection. The data to be sent must not be edited until the send job is completed. If the send job executes successfully, the DONE parameter is set to "1". Signal state "1" at the DONE parameter is not confirmation that the data sent has already been read by the communication partner.
- Terminating the communication connection:
  The communication connection is terminated when the CONT parameter is set to "0" even if an ongoing data transmission is not complete yet. This does not apply if you are using an already configured connection for "TSEND_C".

When setting the COM_RST parameter to "1", the currently established connection or a current data transmission can be reset at any time. This terminates the existing communication connection and a new connection is established. If data is being transferred when it executes again, this can lead to a loss of data.

To enable "TSEND_C" again after the execution (DONE = 1), call the instruction once with REQ = 0.

Parameters

The following table shows the parameters of the "TSEND_C" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Starts the send job on a rising edge.
CONT	Input	BOOL	I, Q, M, D, L	Controls the communication connection: • 0: Disconnect the communication connection • 1: Establish and maintain the communication connection When sending data (rising edge at the REQ parameter), the CONT parameter must have the value TRUE in order to establish or maintain a connection.
LEN	Input	UINT	I, Q, M, D, L or constant	Maximum number of bytes to be sent with the job. If you use purely symbolic values at the DATA parameter, the LEN parameter must have the value "0".
CONNECT	InOut	TCON_Param	D	Pointer to the connection description See also: Auto-Hotspot
DATA	InOut	VARIANT	I, Q, M, D, L	Pointer to the send area containing the address and the length of the data to be sent.
COM_RST	InOut	BOOL	I, Q, M, D, L	Restarts the instruction: • 0: Irrelevant • 1: Complete restart of the instruction causing an existing connection to be terminated and a new connection to be established.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or still in progress • 1: Job completed without error

Parameter	Declaration	Data type	Memory area	Description
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or already completed • 1: Job not yet completed. A new job cannot be started.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

BUSY, DONE and ERROR parameters

You can check the status of the job with the BUSY, DONE, ERROR and STATUS parameters. The BUSY parameter indicates the processing status. With the DONE parameter, you can check whether or not a job executed successfully. The ERROR parameter is set when errors occurred during execution of "TSEND_C". The error information is output at the STATUS parameter.

The following table shows the relationship between the BUSY, DONE and ERROR parameters:

BUSY	DONE	ERROR	Description
1	-	-	The job is being processed.
0	1	0	The job was completed successfully.
0	0	1	The job ended with an error. The cause of the error is specified in the STATUS parameter.
0	0	0	No new job was assigned.

ERROR and STATUS parameters

ERROR	STATUS* (W#16#...)	Description
0	0000	Job completed without error.
0	0001	The connection establishment is complete.
0	0003	The connection termination is complete.
0	7000	No job processing active.
0	7001	• Start execution of the job • Establish connection • Wait for connection partner
0	7002	Data was sent.
0	7003	Connection is being terminated.
0	7004	Connection established and monitored, no job processing active.
1	80A0	Group error for error codes 80A1 and 80A2.

ERROR	STATUS* (W#16#...)	Description
1	80A1	• Connection or port already being used by user. • Communication error: – The specified connection has not yet been established. – The specified connection is being terminated. Transfer via this connection is not possible. – The interface is being re-initialized.
1	80A2	Local or remote port is being used by the system.
1	80A3	Attempt being made to terminate a non-existent connection.
1	80A4	IP address of the remote endpoint of the connection is invalid, in other words, it matches the IP address of the local partner.
1	80A7	Communication error: You called the instruction with COM_RST = 1 before the send job was complete.
1	80AA	A connection is currently being established with the same connection ID by another block. Repeat the job with a new rising edge at the REQ parameter.
1	80B2	The CONNECT parameter points to a data block that was generated with the attribute "Only store in load memory".
1	80B3	Inconsistent parameter assignment: Group error for error codes 80A0 to 80A2, 80A4, 80B4 to 80B9.
1	80B4	You have violated one or both of the following conditions for passive connection establishment (active_est = FALSE) when using the ISO-on-TCP protocol variant (connection_type = B#16#12): "local_tsap_id_len >= B#16#02", and/or "local_tsap_id[1] = B#16#E0".
1	80B5	Only passive connection establishment is permitted for connection type 13 = UDP.
1	80B6	Parameter assignment error in the connection_type parameter of the data block for connection description.
1	80B7	Error in one of the following parameters of the data block for connection description: block_length, local_tsap_id_len, rem_subnet_id_len, rem_staddr_len, rem_tsap_id_len, next_staddr_len.
1	8085	The LEN parameter is larger than the highest permitted value.
1	8086	The ID parameter within the CONNECT parameter is outside the permitted range.
1	8087	Maximum number of connections reached; no additional connection possible.
1	8088	The value at the LEN parameter does not correspond to the receive area set at the DATA parameter.
1	8089	The CONNECT parameter does not point to a data block.
1	8091	Maximum nesting depth exceeded.
1	809A	The CONNECT parameter points to a field that does not correspond to the length of the connection description.
1	809B	The ID of the local device in the connection description does not correspond to the CPU.
1	80C3	• All connection resources are in use. • A block with this ID is already being processed in a different priority group.
1	80C4	Temporary communication error: • The connection cannot be established at this time. • The interface is receiving new parameters or the connection is being established. • The configured connection is currently being removed by a "TDISCON" instruction. • The connection used is being terminated by a call with COM_RST= 1.
1	80C6	Remote network error. Remote partner cannot be reached.
1	8722	Parameter CONNECT: The source area is invalid. The area does not exist in the DB.

ERROR	STATUS* (W#16#...)	Description
1	873A	Parameter CONNECT: Access to the connection description is not possible (for example, DB does not exist).
1	877F	Parameter CONNECT: Internal error.
1	8822	Parameter DATA: Invalid source area, the area does not exist in the DB.
1	8824	Parameter DATA: Area error in the VARIANT pointer.
1	8832	Parameter DATA: The DB number is too high.
1	883A	Parameter CONNECT: Access to specified connection data not possible (e.g. because the DB does not exist).
1	887F	Parameter DATA: Internal error, e.g. invalid VARIANT reference.
1	893A	Parameter DATA: Access to send area not possible (e.g. because the DB does not exist).
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

Note

Error messages of the instructions "TCON", "TSEND", "T_DIAG" and "TDISCON"

Internally, the "TSEND_C" instruction uses the instructions "TCON (Page 5083)", "TSEND (Page 5096)", "T_DIAG (Page 5117)", "T_RESET (Page 5115)" and "TDISCON (Page 5091)". The error messages of these instructions can also be output at the STATUS parameter. The meaning of the error codes is described in the corresponding instructions. In the event of identical error codes for internally used instructions with different meanings, the instance data block of "TSEND_C" can be used to determine which instruction output the error.

TSEND_C: Send data via Ethernet

Description

The "TSEND_C" instruction sets up and establishes a communications connection. Once the connection has been set up and established, it is automatically maintained and monitored by the CPU.

The instruction is executed asynchronously and has the following functions:

- Setting up and establishing a communication connection
- Sending data via an existing communication connection
- Terminating or resetting the communication connection

Internally, the instruction "TSEND_C" uses the communication instructions TCON", "TSEND", "T_DIAG", "T_RESET" and "TDISCON".

Setting up and establishing a communication connection

The communication connection is set up and established with CONT=1. For information on the number of possible communication connections, refer to the technical specifications for

your CPU. The connection description specified at the CONNECT parameter is used to set up the communication connection. The following connection types can be used:

- Programmed connections (structure of connection via "TCON"):
  - TCP / UDP: Connection description via the TCON_IP_v4 system data type
  - ISO-on-TCP: Connection description via the TCON_IP_RFC system data type
  - ISO: Connection description via the TCON_ISOnative system data type (for CP1543-1 only)
- Configured connections
  - Specify an existing connection in the TCON_Configured system data type.

An existing connection is terminated and the connection which has been set up is removed when the CPU goes into STOP mode. To set up and establish the connection again, you must execute "TSEND_C" again.

Sending data via an existing communication connection

The send job is executed when a rising edge is detected at the REQ parameter. As described above, the communication connection is established first.

You specify the send area with the DATA parameter. This includes the address and the length of the data to be sent. Do not use a data area with the data type BOOL or Array of BOOL at the DATA parameter. With the LEN parameter, you specify the maximum number of bytes sent with a send job. If you use a send area with optimized access at the DATA parameter, the LEN parameter must have the value "0".

The data to be sent must not be edited until the send job is completed.

Terminating and resetting the communication connection

The communication connection is terminated when the CONT parameter is set to "0" even if an ongoing data transmission is not complete yet. This does not apply if you are using a configured connection for "TSEND_C".

The connection can be reset at any time by setting the parameter COM_RST to "1". This terminates the existing communication connection and a new connection is established. If data is being transferred at this time, this can lead to data loss.

Parameters

The following table shows the parameters of the "TSEND_C" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, T, C or constant	Starts the send job on a rising edge.
CONT	Input	BOOL	I, Q, M, D, L	Controls the communication connection: • 0: Disconnect the communication connection. • 1: Establish and maintain the communication connection.

Parameter	Declaration	Data type	Memory area	Description
LEN	Input	UDINT	I, Q, M, D, L or constant	Optional parameter (hidden) Maximum number of bytes to be sent with the job. If you use a send area with optimized access at the DATA parameter, the value "0" must be used at the LEN parameter.
CONNECT	InOut	VARIANT	D	Pointer to the structure of the connection description: • Programmed connection: – For TCP, use the TCON_IP_v4 system data type For description, refer to:Auto-Hotspot – For ISO-on-TCP, use the TCON_IP_RFC system data type For description, refer to: Auto-Hotspot – For ISO, use the TCON_ISOnative system data type (only for CP1543-1) For description, refer to instruction "TCON (Page 5086)". • Configured connection: – For existing connections, use the TCON_Configured system data type For a description see "System data type for configured connections" below
DATA	InOut	VARIANT	I, Q, M, D, L	Pointer to the send area containing the address and the length of the data to be sent.
ADDR	InOut	VARIANT	D	Optional parameter (hidden) Pointer to the address of the recipient.
COM_RST	InOut	BOOL	I, Q, M, D, L	Optional parameter (hidden) Resets the connection: • 0: Irrelevant • 1: The existing connection is reset. The COM_RST parameter is reset after evaluation by the "TSEND_C" instruction and should not, therefore, be interconnected statically.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Send job not yet started or still in progress. • 1: Send job executed without error. This state is only displayed for one cycle. The output parameter DONE is set if an intermediate step was completed successfully during processing (connection establishment, sending, connection termination) and if the execution of "TSEND_C" was completed successfully.
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Send job not yet started or already completed. • 1: Send job not yet completed. A new send job cannot be started.

Parameter	Declaration	Data type	Memory area	Description
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred during connection establishment, data transmission or connection termination. The output parameter ERROR can be set due to an error in the "TSEND_C" instruction or the communication instructions used internally.
STATUS	Output	WORD	I, Q, M, D, L	Status of instruction (see the "ERROR and STATUS" parameters" description).

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

REQ, CONT and COM_RST parameters

The parameter CONT controls the connection establishment of the "TSEND_C" instruction regardless of the REQ parameter. The behavior of the CONT parameter partially depends on whether a programmed or a configured connection is used:

- With CONT = "0": No data is sent (regardless of whether a programmed or a configured connection is used).
- When changing CONT = "0" to "1":
  - With a programmed connection, it is established with "TCON".
  - With a configured connection, it is checked with "T_DIAG".
- With CONT = "1":
  - As long as no data is sent (REQ="0"), the connection is checked with "T_DIAG".
  - If the internally used communication instructions signal that no connection end point exists, the connection is automatically reestablished with "TCON".
- When changing CONT = "1" to "0":
  - With a programmed connection, it is terminated with "TDISCON".
  - With a configured connection, it is reset with "T_RESET".

The parameter COM_RST resets the connection when changing from "0" to "1":

- If a connection is established, it is reset with "T_RESET" (regardless of whether a programmed or configured connection is used).
- If no connection is established, the setting of the parameter has no effect.

The REQ and COM_RST parameters only have an effect if CONT has been set to "1". The following table shows the relationship between the REQ, CONT and COM_RST parameters:

REQ	CONT	COM_ RST	Status of the instruction	Description
irrelevant	0	irrelevant	Not yet executed	No job active (STATUS = 7000).
irrelevant	0	irrelevant	Initialization	Connection is being terminated. The instruction is being reset.
irrelevant	0 > 1	irrelevant	Connection establishment	Connection is being established. Data is not being transferred yet.
0	1	0	Connection established	The connection is established and is monitored with the instruction "T_DIAG".
irrelevant	1	0 > 1	Connection established	The connection is interrupted by "T_RESET" briefly and reset.
0 > 1	1	0	Connection established	Instruction starts sending.
irrelevant	1	0 > 1	Data is being sent	Data transmission is interrupted. The connection is being reset.

System data type for configured connections

For configured connections at the CONNECT parameter, use the following structure for connection description to TCON_Configured:

Byte	Parameter	Data type	Start value	Description
0 … 1	InterfaceID	HW_ANY	-	Hardware identifier of the local interface (value range: 0 to 65535).
2 … 3	ID	CONN_OUC	-	Reference to the connection (value range: 1 to 4095). Enter the connection ID of the existing connection.
4	ConnectionType	BYTE	-	Connection type Select 254 (decimal) for a configured connection.

BUSY, DONE and ERROR parameters

You can check the status of the job with the BUSY, DONE, ERROR and STATUS parameters. The BUSY parameter indicates the processing status. With the DONE parameter, you can check whether or not a send job executed successfully. The ERROR parameter is set when errors occurred during execution of "TSEND_C". The error information is output at the STATUS parameter.

The following table shows the relationship between the BUSY, DONE and ERROR parameters:

DONE	BUSY	ERROR	Description
0	0	0	The instruction has not been executed yet (no rising edge at REQ parameter).
0	1	0	The instruction is being executed and calls the internally used communication instructions.

DONE	BUSY	ERROR	Description
1	0	0	The send job was completed successfully. "0000" is output at the STATUS parameter. DONE = "1" is only displayed for one cycle.
0	0	1	The execution of the instruction or an intermediate step during processing was terminated with an error. If there is a subsequent error due to an internally used communication instruction, the error that occurred first during processing is displayed. This state is only displayed for one cycle.

ERROR and STATUS parameters

ERROR	STATUS* (W#16#...)	Description
0	0000	Send job was executed without error.
0	0001	Communication connection established.
0	0003	Communication connection closed.
0	7000	No active send job execution; no communication connection established.
0	7001	Initial call for establishing a connection.
0	7002	Second call for establishing a connection
0	7003	Communication connection is being terminated.
0	7004	Communication connection has been established and is being monitored. No send job execution active.
0	7005	Data transmission in progress.
1	80A1	• Connection or port already being used by user. • Communication error: – The specified connection has not yet been established. – The specified connection is being terminated. Transfer via this connection is not possible. – The interface is being re-initialized.
1	80A3	The nested "T_DIAG" instruction has reported that the connection has closed.
1	80A4	IP address of the remote endpoint of the connection is invalid or it matches the IP address of the local partner.
1	80A7	Communication error: You called the instruction with COM_RST = 1 before the send job was complete.
1	80AA	A connection is currently being established with the same connection ID by another block. Repeat the job with a new rising edge at the REQ parameter.
1	80B4	You have violated one or both of the following conditions for passive connection establishment (active_est = FALSE) when using the ISO-on-TCP protocol variant (connection_type = B#16#12): • local_tsap_id_len >= B#16#02 • local_tsap_id[1] = B#16#E0
1	80B5	Only passive connection establishment is permitted for connection type 13 = UDP.
1	80B6	Parameter assignment error in the connection_type parameter of the data block for connection description.
1	80B7	Error in one of the following parameters of the data block for connection description: block_length, local_tsap_id_len, rem_subnet_id_len, rem_staddr_len, rem_tsap_id_len, next_staddr_len.

ERROR	STATUS* (W#16#...)	Description
1	8085	The LEN parameter is larger than the highest permitted value.
1	8086	The ID parameter within the CONNECT parameter is outside the permitted range.
1	8087	Maximum number of connections reached; no additional connection possible.
1	8088	The value at the LEN parameter does not correspond to the receive area set at the DATA parameter.
1	8089	The CONNECT parameter does not point to a data block.
1	8091	Maximum nesting depth exceeded.
1	809A	The CONNECT parameter points to a field that does not correspond to the length of the connection description.
1	809B	InterfaceID is invalid. It is either zero or it does not point to a local CPU interface or a CP.
1	80C3	• All connection resources are in use. • A block with this ID is already being processed in a different priority group.
1	80C4	Temporary communication error: • The connection cannot be established at this time. • The interface is receiving new parameters or the connection is being established. • The configured connection is currently being removed by a "TDISCON (Page 5091)" instruction. • The connection used is being terminated by a call with COM_RST = 1.
1	80C6	Remote network error. Remote partner cannot be reached.
1	8722	Parameter CONNECT: The source area is invalid. The area does not exist in the DB.
1	873A	Parameter CONNECT: Access to the connection description is not possible (for example, because the DB is not available).
1	877F	Parameter CONNECT: Internal error.
1	8822	Parameter DATA: Invalid source area, the area does not exist in the DB.
1	8824	Parameter DATA: Area error in the VARIANT pointer.
1	8832	Parameter DATA: The DB number is too high.
1	883A	Parameter CONNECT: Access to specified connection data not possible (e.g. because the DB does not exist).
1	887F	Parameter DATA: Internal error, e.g. invalid VARIANT reference.
1	893A	Parameter DATA: Access to send area not possible (e.g. because the DB does not exist).
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

Note

Error messages of the instructions "TCON", "TSEND", "T_DIAG", "T_RESET" and "TDISCON"

Internally, the "TSEND_C" instruction uses the instructions "TCON (Page 5086)", "TSEND (Page 5096)", "T_DIAG (Page 5117)", "T_RESET (Page 5115)" and "TDISCON (Page 5091)". The error messages of these instructions can also be output at the STATUS parameter. The meaning of the error codes is described in the corresponding instructions. In the event of identical error codes for internally used instructions with different meanings, the instance data block of "TSEND_C" can be used to determine which instruction output the error.

TRCV_C: Receive data via Ethernet

TRCV_C: Receive data via Ethernet

Description

The "TRCV_C" instruction is executed asynchronously and has the following functions:

1. **Setting up and establishing a communication connection:**
   "TRCV_C" sets up and establishes a TCP or ISO-on-TCP communication connection. Once the connection has been set up and established, it is automatically maintained and monitored by the CPU.
   The connection description specified at the CONNECT parameter is used to set up the communication connection. To establish a connection, the CONT parameter must be set to the value "1". Once the connection is successfully established, the DONE parameter is set to "1".
   An existing connection is terminated and the connection which has been set up is removed when the CPU goes into STOP mode. To set up and establish the connection again, you must execute "TRCV_C" again.
   For information on the number of possible communication connections, refer to the technical specifications for your CPU.
2. **Receiving data via an existing communication connection:**
   If the EN_R parameter is set to the value "1", receipt of data is enabled. When receiving data (rising edge at the EN_R parameter), the CONT parameter must have the value TRUE in order to establish or maintain a connection.
   The received data is entered in a receive area. You specify the length of the receive area either with the LEN parameter (if LEN <> 0) or with the length information of the DATA parameter (if LEN = 0) in accordance with the protocol variant used. If you use purely symbolic values at the DATA parameter, the LEN parameter must have the value "0".
   After data has been received successfully, the signal state at the DONE parameter is "1".
   If errors occur in the data transfer, the DONE parameter is set to "0".
3. **Terminating the communication connection:**
   The communication connection is terminated immediately when the CONT parameter is set to "0".

TRCV_C is executed again when the COM_RST parameter is set. This terminates the existing communication connection and a new connection is established. If data is being received when it executes again, this can lead to a loss of data.

Receive modes of TRCV_C

The following table shows how the received data is entered in the receive area.

Protocol variant	Availability of data in the receive area	connection_type parameter of the connection description	LEN parameter
TCP (Ad-hoc mode)	The data is immediately available.	Hexadecimal value: B#16#11 Integer value: 17	0
TCP (Receipt of data with specified length)	The data is available as soon as the data length specified at the LEN parameter has been fully received.	Hexadecimal value: B#16#11 Integer value: 17	1 to 8192
ISO on TCP (Message-oriented data transfer)	The data is available as soon as the data length specified at the LEN parameter has been fully received.	Hexadecimal value: B#16#12 Integer value: 18	• 1 to 1452, if a CP is used. • 1 to 8192, if no CP is used.

TCP (Ad-hoc mode)

The ad-hoc mode is only available with the TCP protocol variant. You use the ad-hoc mode to receive data with dynamic length with the "TRCV" instruction.

You set ad-hoc mode by assigning the value "0" to the LEN parameter. All data types can be used for data blocks with standard access when you use ad-hoc mode. Only ARRAY of BYTE or data types with a length of 8 bits can be used for data blocks with optimized access (e.g., CHAR, USINT, SINT, etc.). The data length actually received is output at the RCVD_LEN parameter.

TCP (Receipt of data with specified length)

You use the value of the LEN parameter to specify the length for the data receipt. The data receipt is not complete until the length of data specified at the LEN parameter has been completely received. Only then is the data available in the receive area (DATA parameter). The actually received data length in bytes at the RCVD_LEN parameter corresponds to the data length at the LEN parameter after receipt.

ISO on TCP (Message-oriented data transfer)

Complete message blocks are sent via a connection with the protocol variant ISO on TCP; these are recognized as such by the recipient. When using ISO on TCP, "TRCV_C" signals data receipt as soon as the message block has been completely received. The receive area is defined by the LEN and DATA parameters. If the receive buffer (DATA parameter) is too small for the sent data, "TRCV_C" signals an error. The actually received data length in bytes at the RCVD_LEN parameter corresponds to the data length at the LEN parameter after receipt.

Parameters

The following table shows the parameters of the "TRCV_C" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L	Receive enable
CONT	Input	BOOL	I, Q, M, D, L	Controls the communication connection: • 0: Disconnect the communication connection • 1: Establish and maintain the communication connection When receiving data (rising edge at the EN_R parameter), the CONT parameter must have the value TRUE in order to establish or maintain a connection.
LEN	Input	UINT	I, Q, M, D, L or constant	Maximum length of the data to be received. If you use purely symbolic values at the DATA parameter, the LEN parameter must have the value "0".
CONNECT	InOut	TCON_Param	D	Pointer to the connection description See also: Auto-Hotspot
DATA	InOut	VARIANT	I, Q, M, D, L	Pointer to the receive area
COM_RST	InOut	BOOL	I, Q, M, D, L	Restarts the instruction: • 0: Irrelevant • 1: Complete restart of the instruction causing an existing connection to be terminated
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or still in progress • 1: Job completed without error
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or already completed • 1: Job not yet completed. A new job cannot be started
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction
RCVD_LEN	Output	UDINT	I, Q, M, D, L	Amount of data actually received in bytes

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

BUSY, DONE and ERROR parameters

You can check the status of the job with the BUSY, DONE, ERROR and STATUS parameters. The BUSY parameter indicates the processing status. With the DONE parameter, you can check whether or not a job executed successfully. The ERROR parameter is set when errors occurred during execution of "TRCV_C". The error information is output at the STATUS parameter.

The following table shows the relationship between the BUSY, DONE and ERROR parameters:

BUSY	DONE	ERROR	Description
1	-	-	The job is being processed.
0	1	0	The job was completed successfully.
0	0	1	The job ended with an error. The cause of the error is output at the STATUS parameter.
0	0	0	No new job was assigned.

ERROR and STATUS parameters

ERROR	STATUS* (W#16#...)	Description
0	0000	Job completed without error.
0	0001	The connection establishment is complete.
0	0003	The connection termination is complete.
0	7000	No job processing active.
0	7001	• Start execution of the job • Establish connection • Wait for connection partner
0	7002	Data is being received.
0	7003	Connection is being terminated.
0	7004	• Connection established and monitored • No job processing active
0	7006	Data is currently being received.
1	8085	• The LEN parameter is larger than the highest permitted value. • The value at the LEN or DATA parameter was changed after the first call.
1	8086	The ID parameter is outside the permitted range.
1	8087	Maximum number of connections reached; no additional connection possible
1	8088	The value at the LEN parameter does not correspond to the receive area set at the DATA parameter.
1	8089	The CONNECT parameter does not point to a data block.
1	8091	Maximum nesting depth exceeded.
1	809A	The CONNECT parameter points to a field that does not correspond to the length of the connection description.
1	809B	The ID of the local device (local_device_id) in the connection description does not correspond to the CPU.
1	80A0	Group error for error codes W#16#80A1 and W#16#80A2.
1	80A1	• Connection or port already being used by user. • Communication error: – The specified connection has not yet been established. – The specified connection is being terminated. Transfer via this connection is not possible. – The interface is being re-initialized.
1	80A2	Local or remote port is being used by the system.

ERROR	STATUS* (W#16#...)	Description
1	80A3	• Attempt being made to re-establish an existing connection. • Attempt being made to terminate a non-existent connection.
1	80A4	IP address of the remote endpoint of the connection is invalid, in other words, it matches the IP address of the local partner.
1	80A7	Communication error: You called the instruction with COM_RST = 1 before the send job was complete.
1	80B2	The CONNECT parameter points to a data block that was generated with the attribute "Only store in load memory".
1	80B3	Inconsistent parameter assignment: Group error for error codes W#16#80A0 to W#16#80A2, W#16#80A4, W#16#80B4 to W#16#80B9.
1	80B4	You have violated one or both of the following conditions for passive connection establishment (active_est = FALSE) when using the ISO-on-TCP protocol variant (connection_type = B#16#12): "local_tsap_id_len >= B#16#02", and/or "local_tsap_id[1] = B#16#E0".
1	80B5	Only passive connection establishment is permitted for connection type 13 = UDP.
1	80B6	Parameter assignment error in the connection_type parameter of the data block for connection description.
1	80B7	Error in one of the following parameters of the data block for connection description: block_length, local_tsap_id_len, rem_subnet_id_len, rem_staddr_len, rem_tsap_id_len, next_staddr_len.
1	80C3	• All connection resources are in use. • A block with this ID is already being processed in a different priority group.
1	80C4	Temporary communication error: • The connection cannot be established at this time. • The interface is receiving new parameters or the connection is being established. • The configured connection is currently being removed by a "TDISCON" instruction. • The connection used is being terminated by a call with COM_RST= 1.
1	80C6	The remote partner cannot be reached (network error).
1	8722	Error in the CONNECT parameter: Invalid source area (area not declared in data block).
1	873A	Error in the CONNECT parameter: Access to connection description is not possible (no access to data block).
1	877F	Error in the CONNECT parameter: Internal error
1	8922	Parameter DATA: Invalid target area; the area does not exist in the DB.
1	8924	Parameter DATA: Area error in the VARIANT pointer.
1	8932	Parameter DATA: The DB number is too high.
1	893A	Parameter CONNECT: Access to specified connection data not possible (e.g. because the DB does not exist).
1	897F	Parameter DATA: Internal error, e.g. invalid VARIANT reference.
1	8A3A	Parameter DATA: No access to the data area, for example because the data block does not exist.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

Note

Error messages of the instructions "TCON", "TRCV" and "TDISCON"

Internally, the TRV_C instruction uses the "TCON (Page 5083)", "TRCV (Page 5099)" and "TDISCON (Page 5091)" instructions. The error messages of these instructions are contained in the respective descriptions.

TRCV_C: Receive data via Ethernet

Description

The "TRCV_C" instruction is executed asynchronously and implements the following functions in sequence:

- Setting up and establishing a communication connection
- Receiving data via an existing communication connection
- Terminating or resetting the communication connection

Internally, the instruction "TRCV_C" uses the communication instructions TCON", "TRCV", "T_DIAG", "T_RESET" and "TDISCON".

Setting up and establishing a communication connection

The communication connection is set up and established with CONT=1. For information on the number of possible communication connections, refer to the technical specifications for your CPU. The connection description specified at the CONNECT parameter is used to set up the communication connection. The following connection types can be used:

- Programmed connections (structure of connection via "TCON"):
  - TCP / UDP: Connection description via the TCON_IP_v4 system data type
  - ISO-on-TCP: Connection description via the TCON_IP_RFC system data type
  - ISO: Connection description via the TCON_ISOnative system data type (for CP1543-1 only)
- Configured connections
  - Specify an existing connection in the TCON_Configured system data type.

An existing connection is terminated and the connection which has been set up is removed when the CPU goes into STOP mode. To set up and establish the connection again, you must execute "TRCV_C" again.

Receiving data via an existing communication connection

Receipt of data is enabled when the EN_R parameter is set to the value "1". The received data is entered in a receive area. You specify the length of the receive area either with the LEN parameter (if LEN <> 0) or with the length information of the DATA parameter (if LEN = 0),

depending on the protocol variant being used. If you use purely symbolic values at the DATA parameter, the LEN parameter must have the value "0".

Receive modes of TRCV_C:

- TCP (Ad-hoc mode)
  The ad-hoc mode is only available with the TCP protocol variant. You use the ad-hoc mode to receive data with dynamic length with the "TRCV_C" instruction.
  You set ad-hoc mode by assigning the value "1" to the ADHOC parameter. All data types can be used for data blocks with standard access when you use ad-hoc mode. Only ARRAY of BYTE or data types with a length of 8 bits can be used for data blocks with optimized access (e.g., CHAR, USINT, SINT, etc.). The data length actually received is output at the RCVD_LEN parameter.
- TCP (Receipt of data with specified length)
  Assign the value "0" to the ADHOC parameter for receipt of data with specified length. If ad-hoc mode is disabled, the data receipt is not complete until the length of data specified at the LEN parameter has been completely received. Only then is the data available in the receive area (DATA parameter). The actually received data length in bytes at the RCVD_LEN parameter corresponds to the data length at the LEN parameter after receipt.
- ISO on TCP (Message-oriented data transfer)
  Complete message blocks are sent via a connection with the protocol variant ISO on TCP; these are recognized as such by the recipient. The receive area is defined by the LEN and DATA parameters. If the receive buffer (DATA parameter) is too small for the sent data, "TRCV_C" signals an error. The actually received data length in bytes at the RCVD_LEN parameter corresponds to the data length at the LEN parameter after receipt.

The following table shows how the received data is entered in the receive area.

Protocol variant	Availability of data in the receive area	connection_type parameter of the connection description	LEN parameter
TCP (Ad-hoc mode)	The data is immediately available.	Hexadecimal value: B#16#11 Integer value: 17	1 up to maximum length (depending on the CPU)
TCP (Receipt of data with specified length)	The data is available as soon as the data length specified at the LEN parameter has been fully received.	Hexadecimal value: B#16#11 Integer value: 17	1 to 8192
ISO on TCP (Message-oriented data transfer)	The data is available as soon as the data length specified at the LEN parameter has been fully received.	Hexadecimal value: B#16#12 Integer value: 18	1 to 8192

Terminating the communication connection

The communication connection is terminated when the CONT parameter is set to "0" even if an ongoing data transmission is not complete yet. This does not apply if you are using a configured connection, however.

The connection can be reset at any time by setting the parameter COM_RST to "1". This terminates the existing communication connection and a new connection is established. If data is being transferred at this time, this can lead to data loss.

Parameters

The following table shows the parameters of the "TRCV_C" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L, T, C or constant	Receive enable
CONT	Input	BOOL	I, Q, M, D, L	Controls the communication connection: • 0: Disconnect the communication connection. • 1: Establish communication connection and maintain after receipt of data.
LEN	Input	UDINT	I, Q, M, D, L or constant	Maximum length of the data to be received. If you use a receive area with optimized access at the DATA parameter, the value "0" must be used at the LEN parameter.
ADHOC	Input	BOOL	I, Q, M, D, L or constant	Optional parameter (hidden) Use ad-hoc mode for the TCP protocol variant.
CONNECT	InOut	VARIANT	D	Pointer to the connection description • Programmed connection: – For TCP or UDP, use the structure TCON_IP_v4 For description, refer to:Auto-Hotspot – For ISO-on-TCP, use the structure TCON_IP_RFC For description, refer to: Auto-Hotspot – For ISO, use the structure TCON_ISOnative (only for CP1543-1) For description, refer to instruction "TCON (Page 5086)": "Structure of the connection description according to TCON_ISOnative" • Configured connection: – For existing connections, use the TCON_Configured system data type. For a description, see "System data type for configured connections" below.
DATA	InOut	VARIANT	I, Q, M, D, L	Pointer to the receive area.
ADDR	InOut	VARIANT	D	Optional parameter (hidden) Pointer to the address of the sender with connection type UDP.
COM_RST	InOut	BOOL	I, Q, M, D, L	Optional parameter (hidden) Resets the connection: • 0: Irrelevant • 1: The existing connection is reset. The COM_RST parameter is reset after evaluation by the "TRCV_C" instruction and should not, therefore, be interconnected statically.

Parameter	Declaration	Data type	Memory area	Description
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Receipt not yet started or still in progress. • 1: Receipt executed without error. This state is only displayed for one cycle. The output parameter DONE is set if an intermediate step was completed successfully during processing (connection establishment, receipt, connection termination) and if the execution of "TRCV_C" was completed successfully.
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Receipt not yet started or already completed. • 1: Receipt not yet completed. A new send job cannot be started.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred during connection establishment, data receipt or connection termination. The output parameter ERROR can be set due to an error in the "TRCV_C" instruction or the communication instructions used internally.
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction
RCVD_LEN	Output	UDINT	I, Q, M, D, L	Amount of data actually received in bytes

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

EN_R, CONT and COM_RST parameters

The parameter CONT controls the connection establishment of the "TRCV_C" instruction regardless of the EN_R parameter. The behavior of the CONT parameter partially depends on whether a programmed or a configured connection is used:

- With CONT = "0": No data is received (regardless of whether a programmed or a configured connection is used).
- When changing CONT = "0" to "1":
  - With a programmed connection, it is established with "TCON".
  - With a configured connection, it is checked with "T_DIAG".
- With CONT = "1":
  - As long as no data is received (EN_R="0"), the connection is checked with "T_DIAG".
  - If the internally used communication instructions signal that no connection end point exists, the connection is automatically reestablished with "TCON".
- When changing CONT = "1" to "0":
  - With a programmed connection, it is terminated with "TDISCON".
  - With a configured connection, it is reset with "T_RESET".

The parameter COM_RST resets the connection when changing from "0" to "1":

- If a connection is established, it is reset with "T_RESET" (regardless of whether a programmed or configured connection is used).
- If no connection is established, the setting of the parameter has no effect.

The EN_R and COM_RST parameters only have an effect if CONT has been set to "1". The following table shows the relationship between the EN_R, CONT and COM_RST parameters:

EN_R	CONT	COM_ RST	Status of the instruction	Description
irrelevant	0	irrelevant	Not yet executed	No job active (STATUS = 7000).
irrelevant	0	irrelevant	Initialization	Connection is being terminated. The instruction is being reset.
irrelevant	0 > 1	irrelevant	Connection establishment	Connection is being established. Data is not being transferred yet.
0	1	0	Connection established	The connection is established and is monitored with the instruction "T_DIAG".
irrelevant	1	0 > 1	Connection established	The connection is interrupted by "T_RESET" briefly and reset.
0 > 1	1	0	Connection established	Instruction starts receiving.
irrelevant	1	0 > 1	Data is being received	Data transmission is interrupted. The connection is being reset.

System data type for configured connections

For configured connections at the CONNECT parameter, use the following structure for connection description to TCON_Configured:

Byte	Parameter	Data type	Start value	Description
0 … 1	InterfaceID	HW_ANY	-	Hardware identifier of the local interface (value range: 0 to 65535).
2 … 3	ID	CONN_OUC	-	Reference to the connection (value range: 1 to 4095). Enter the connection ID of the existing connection.
4	ConnectionType	BYTE	-	Connection type Select 254 (decimal) for a configured connection.

BUSY, DONE and ERROR parameters

You can check the status of the job with the BUSY, DONE, ERROR and STATUS parameters. The BUSY parameter indicates the processing status. With the DONE parameter, you can check whether or not a job executed successfully. The ERROR parameter is set when errors occurred during execution of "TRCV_C". The error information is output at the STATUS parameter.

The following table shows the relationship between the BUSY, DONE and ERROR parameters:

DONE	BUSY	ERROR	Description
0	0	0	The instruction has not been executed yet (no rising edge at EN_R parameter).
0	1	0	The instruction is being executed and calls the internally used communication instructions.
1	0	0	The receipt has been completed successfully. "0000" is output at the STATUS parameter. DONE = "1" is only displayed for one cycle.
0	0	1	The execution of the instruction or an intermediate step during processing was terminated with an error. If there is a subsequent error due to an internally used communication instruction, the error that occurred first during processing is displayed. This state is only displayed for one cycle.

ERROR and STATUS parameters

ERROR	STATUS (W#16#...)	Description
0	0000	Receive job executed without error.
0	0001	Communication connection established.
0	0003	Communication connection closed.
0	7000	No job processing active.
0	7001	Initial call for establishing a connection.
0	7002	Second call for establishing a connection.
0	7003	Communication connection is being terminated.
0	7004	Communication connection has been established and is being monitored. No receive job processing active.
0	7006	Data is currently being received.
1	8085	• The LEN parameter is larger than the highest permitted value. • The value at the LEN or DATA parameter was changed after the first call.
1	8086	The ID parameter is outside the permitted range.
1	8087	Maximum number of connections reached; no additional connection possible
1	8088	The value at the LEN parameter does not correspond to the receive area set at the DATA parameter.
1	8089	The CONNECT parameter does not point to a data block.
1	8091	Maximum nesting depth exceeded.
1	809A	The CONNECT parameter points to a field that does not correspond to the length of the connection description.
1	809B	InterfaceID is invalid. It is either zero or it does not point to a local CPU interface or a CP.
1	80A1	• Connection or port already being used by user. • Communication error: – The specified connection has not yet been established. – The specified connection is being terminated. Transfer via this connection is not possible. – The interface is being re-initialized.
1	80A3	The nested "T_DIAG" instruction has reported that the connection has closed.
1	80A4	IP address of the remote endpoint of the connection is invalid or it matches the IP address of the local partner.
1	80A7	Communication error: You called the instruction with COM_RST = 1 before the send job was complete.

ERROR	STATUS (W#16#...)	Description
1	80B4	You have violated one or both of the following conditions for passive connection establishment (active_est = FALSE) when using the ISO-on-TCP protocol variant (connection_type = B#16#12): "local_tsap_id_len >= B#16#02", and/or "local_tsap_id[1] = B#16#E0".
1	80B5	Only passive connection establishment is permitted for connection type 13 = UDP.
1	80B6	Parameter assignment error in the connection_type parameter of the data block for connection description.
1	80B7	Error in one of the following parameters of the data block for connection description: block_length, local_tsap_id_len, rem_subnet_id_len, rem_staddr_len, rem_tsap_id_len, next_staddr_len.
1	80C3	• All connection resources are in use. • A block with this ID is already being processed in a different priority group.
1	80C4	Temporary communication error: • The connection cannot be established at this time. • The interface is receiving new parameters or the connection is being established. • The configured connection is currently being removed by a "TDISCON" instruction. • The connection used is being terminated by a call with COM_RST = 1.
1	80C6	The remote partner cannot be reached (network error).
1	8722	Error in the CONNECT parameter: Invalid source area (area not declared in data block).
1	873A	Error in the CONNECT parameter: Access to connection description is not possible (no access to data block).
1	877F	Error in the CONNECT parameter: Internal error
1	8922	Parameter DATA: Invalid target area; the area does not exist in the DB.
1	8924	Parameter DATA: Area error in the VARIANT pointer.
1	8932	Parameter DATA: The DB number is too high.
1	893A	Parameter CONNECT: Access to specified connection data not possible (e.g. because the DB does not exist).
1	897F	Parameter DATA: Internal error, e.g. invalid VARIANT reference.
1	8A3A	Parameter DATA: No access to the data area, for example because the data block does not exist.

Note

Error messages of the instructions "TCON", "TRCV" and "TDISCON"

Internally, the TRV_C instruction uses the "TCON (Page 5086)", "TRCV (Page 5099)" and "TDISCON (Page 5091)" instructions. The error messages of these instructions are contained in the respective descriptions.

See also

TSEND_C: Send data via Ethernet (Page 5054)

TMAIL_C: Transfer email

Description of TMAIL_C

Description

You use the instruction "TMAIL_C" to send an e-mail via the Ethernet interface of the S7-1500 CPU or S7-1200 V4.0, a communication module (CM), or of a communication processor (CP).

The instruction can only be used once the hardware has been configured and if the network infrastructure allows for a communication connection to the mail server.

You define the content of the e-mail, and the connection data, using the following parameters:

- You define the recipient addresses with the parameters TO_S and CC.
- You define the content of the e-mail with the parameters SUBJECT and TEXT.
- You can define an attachment using VARIANT pointers at the ATTACHMENT and ATTACHMENT_NAME parameters.
- The connection data is defined, and addressing and authentication for the mail server executed, using the system data type Tmail_v4, Tmail_v6 or Tmail_FQDN at the MAIL_ADDR_PARAM parameter.
  - If you are using the interface of the S7-1500 CPU, the system data type Tmail_v4 must be used. In this case, the e-mail can only be sent via SMTP.
  - Any of the system data types can be used if you are using the interface of a CM/CP. The e-mail can then also be sent via SMTPS.
- You start the sending of an e-mail with an edge change from "0" to "1" for the REQ parameter.
- The job status is indicated by the output parameters "BUSY", "DONE", "ERROR" and "STATUS".

You cannot send an SMS directly with the "TMAIL_C" instruction. Whether or not the e-mail can be forwarded by the mail server as an SMS depends on your telecommunications provider.

Operation of the instruction

The "TMAIL_C" instruction works asynchronously, which means its execution extends over multiple calls. You must specify an instance when you call the instruction "TMAIL_C".

In the following cases, the connection to the mail server will be lost:

- If the CPU switches to STOP while "TMAIL_C" is active.
- If communication problems occur at the Industrial Ethernet bus.

In this case, the transfer of the e-mail will be interrupted and it will not reach its recipient. The connection is also canceled once the instruction has been successfully executed and the e-mail sent.

NOTICE
Changing user programs You can change the parts of your user program that directly affect calls of "TMAIL_C" only: • The CPU is in "STOP" mode. • No e-mail is being sent (REQ = 0 and BUSY = 0). This relates, in particular, to deleting and replacing program blocks that contain "TMAIL_C" calls or calls for the instance of "TMAIL_C". Ignoring this restriction can tie up connection resources. The automation system can change to an undefined status with the TCP/IP communication functions via Industrial Ethernet. A warm or cold restart of the CPU is required after the changes are transferred.

Data consistency

The TO_S, CC, SUBJECT, TEXT, ATTACHMENT and MAIL_ADDR_PARAM parameters are applied by the "TMAIL_C" instruction while it is running, which means that they may only be changed after the job has been completed (BUSY = 0).

SMTP authentication

Authentication refers here to a procedure for verifying identity, for example, with a password query.

If you are using the S7-1500 CPU interface, the instruction "TMAIL_C" supports the SMTP authentication procedure AUTH-LOGIN which is required by most mail servers. For information about the authentication procedure of your mail server, please refer to your mail server manual or the website of your Internet service provider.

- Before you can use the AUTH-LOGIN authentication procedure, the "TMAIL_C" instruction requires the user name with which it is to log on to the mail server. This user name corresponds to the user name with which you set up a mail account on your mail server. It is transferred via the UserName parameter to the structure at parameter MAIL_ADDR_PARAM.
  If no user name is specified at the MAIL_ADDR_PARAM parameter, the AUTH-LOGIN authentication procedure is not used. The e-mail is then sent without authentication.
- To log on, the "TMAIL_C" instruction also requires the associated password. This password corresponds to the password you specified when you set up your mail account. It is transferred via the PassWord parameter to the structure at parameter MAIL_ADDR_PARAM.

Parameters

The following table shows the parameters of the "TMAIL_C" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, T, C or constant	Control parameter REQUEST: Activates the sending of an e-mail upon a rising edge.
TO_S (Page 5076)	Input	STRING	D, L or constant	Recipient addresses STRING with a maximum length of 240 characters (bytes). For the e-mail address format, see the example in the parameter description.
CC (Page 5076)	Input	STRING	D, L or constant	CC recipient addresses (optional) STRING with a maximum length of 240 characters (bytes). Same e-mail address format as for the TO_S parameter. If an empty string is assigned here, the e-mail is not sent to a CC recipient.
SUBJECT	Input	STRING	D, L or constant	Subject of the e-mail STRING with a maximum length of 240 characters (bytes).
TEXT	Input	STRING	D, L or constant	Text of the e-mail (optional) STRING with a maximum length of 240 characters (bytes). If an empty string is assigned at this parameter, the e-mail is sent without text.
ATTACHMENT	Input	VARIANT	D	E-mail attachment (optional) Reference to a byte/word/double word field (ArrayOfByte, ArrayOfWord or ArrayOfDWord) with a maximum length of 64 bytes. If no value is assigned, the e-mail is sent without an attachment.
ATTACHMENT_NAME	Input	STRING	D, L or constant	E-mail attachment name (optional) Reference to a character string with a maximum length of 50 characters (bytes) to define the file name of the attachment. If an empty string is assigned at this parameter, the e-mail attachment will be sent with the file name "attachment.bin".
MAIL_ADDR_PARAM (Page 5077)	Input	VARIANT	D	Connection parameter and address of the e-mail server To define the connection parameters, use the structure Tmail_v4, Tmail_v6 or Tmail_FQDN (see parameter description).
DONE (Page 5079)	Output	BOOL	I, Q, M, D, L	Status parameter • DONE = 0: Job not yet started or still executing. • DONE = 1: Job was executed without errors.

Parameter	Declaration	Data type	Memory area	Description
BUSY (Page 5079)	Output	BOOL	I, Q, M, D, L	Status parameter • BUSY=0: The processing of "TMAIL_C" was stopped. • BUSY = 1: E-mail transmission is not yet complete.
ERROR (Page 5079)	Output	BOOL	I, Q, M, D, L	Status parameter • ERROR = 0: No error has occurred. • ERROR = 1: An error occurred during processing. STATUS supplies detailed information on the type of error.
STATUS (Page 5080)	Output	WORD	I, Q, M, D, L	Status parameter Return value or error information of the "TMAIL_C" instruction (see parameter description).

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

Note

Optional parameters

The optional parameters CC, TEXT, and ATTACHMENT are only sent with the e-mail if the corresponding parameters contain a string of length > 0.

TO_S and CC parameters

Description

The TO_S and CC parameters are strings, for example, with the following content:

- <wenna@mydomain.com>, <ruby@mydomain.com>
- <admin@mydomain.com>, <judy@mydomain.com>

Note the following rules when entering the parameters:

- A space and an opening pointed bracket "<" must be entered before each address.
- A closing pointed bracket ">" must be entered after each address.
- A comma must be entered between the addresses in TO and CC.

For runtime and memory space reasons, the "TMAIL_C" instruction does not perform a syntax check of parameter TO_S or CC.

MAIL_ADDR_PARAM parameter

Description

At the MAIL_ADDR_PARAM parameter, you define the connection for sending the e-mail in the structure Tmail_v4, Tmail_v6 or Tmail_FQDN, and save the e-mail server address and login details.

The structure you use at the MAIL_ADDR_PARAM parameter depends on the format in which the e-mail server is to be addressed:

- Tmail_v4: Addressing by IP address (IPv4).
- Tmail_v6: Addressing by IP address (IPv6).
- Tmail_FQDN: Addressing by fully qualified domain name (FQDN).

Which structure you can use depends on the interface addressed at the InterfaceId parameter:

- If you want to use the "TMAIL_C" instruction with the internal interface, the structure Tmail_v4 must be used at the MAIL_ADDR_PARAM parameter.
- If you are using a communication processor (CP) or communication module (CM), you can use all three addressing options (IPv4, IPv6 and FQDN).

Table 9-94 **Tmail_v4: Addressing the mail server by IP address (IPv4)**

Parameter	Data type	Description
Tmail_v4	Struct	
InterfaceId	LADDR	Hardware identifier of the interface
ID	CONN_OUC	Connection ID
ConnectionType	BYTE	Connection type. Select 16#20 as the connection type for IPv4.
ActiveEstablished	BOOL	Status bit. Set to "1" once the connection is established.
CertIndex	BYTE	• =0: SMTP used (**S**imple **M**ail **T**ransfer **P**rotocol). SMTP must be used if the e-mail is being sent via the interface of an S7-1500 CPU. • ≠0: SMTPS used to secure the connection before it is established (with CPs/CMs). You use the CertIndex parameter to specify the certificate to be used (see "Project navigation" > "Global security settings" > "Certificate manager").
WatchDogTime	TIME	Execution watchdog. Use this parameter to define the maximum execution time for the send operation. Note: Connection establishment can take longer (approx. one minute) if the connection is slow. When you specify the WATCH_DOG_TIME parameter, remember to allow for the time required to establish the connection. The connection is terminated once the specified time has elapsed.
MailServerAddress	IP_v4	IP address of the mail server. IPv4 in the following format: XXX.XXX.XXX.XXX (decimal). Example: 192.142.131.237.
UserName	STRING[254]	Mail server login name

Parameter	Data type	Description
PassWord	STRING[254]	Mail server password
From	EMAIL_ADDR	E-mail sender address, which is defined using the following two STRING parameters. For example: "myname@mymailserver.com".
LocalPartPlusAtSign	STRING[64]	Local part of sender address, including @ sign. Example: "myname@".
FullQualifiedDomainName	STRING[254]	Fully Qualified Domain Name (FQDN for short) of the mail server. Example: "mymailserver.com".

Table 9-95 **Tmail_v6: Addressing the mail server by IP address (IPv6)**

Parameter	Data type	Description
Tmail_v6	Struct	
InterfaceId	LADDR	Hardware identifier of the interface
ID	CONN_OUC	Connection ID
ConnectionType	BYTE	Connection type. Select 16#21 as the connection type for IPv6.
ActiveEstablished	BOOL	Status bit. Set to "1" once the connection is established.
CertIndex	BYTE	• =0: SMTP used (Simple Mail Transfer Protocol). SMTP must be used if the e-mail is being sent via the interface of an S7-1500 CPU. • ≠0: SMTPS used to secure the connection before it is established (with CPs/CMs). You use the CertIndex parameter to specify the certificate to be used (see "Project navigation" > "Global security settings" > "Certificate manager").
WatchDogTime	TIME	Execution watchdog. Use this parameter to define the maximum execution time for the send operation. Note: Connection establishment can take longer (approx. one minute) if the connection is slow. When you specify the WATCH_DOG_TIME parameter, remember to allow for the time required to establish the connection. The connection is terminated once the specified time has elapsed.
MailServerAddress	IP_v6	IP address of the mail server (IPv6) in the following format: XXXX.XXXX.XXXX.XXXX.XXXX.XXXX.XXXX.XXXX (hexadecimal). The address is divided into 8 blocks of 2 bytes each (16 bytes in total). Example: 2001:db8:1f11:08d3:290:27ff:0370:2093
UserName	STRING[254]	Mail server login name
PassWord	STRING[254]	Mail server password
From	EMAIL_ADDR	E-mail sender address, which is defined using the following two STRING parameters. For example: "myname@mymailserver.com".
LocalPartPlusAtSign	STRING[64]	Local part of sender address, including @ sign. Example: "myname@".
FullQualifiedDomainName	STRING[254]	Fully Qualified Domain Name (FQDN for short) of the mail server. Example: "mymailserver.com".

Table 9-96 Tmail_FQDN: Addressing the mail server by FQDN

Parameter	Data type	Description
Tmail_v6	Struct	
Tmail_FQDN	LADDR	Hardware identifier of the interface
ID	CONN_OUC	Connection ID
ConnectionType	BYTE	Connection type. Select 16#22 as the connection type for FQDN.
ActiveEstablished	BOOL	Status bit. Set to "1" once the connection is established.
CertIndex	BYTE	• =0: SMTP used (Simple Mail Transfer Protocol). SMTP must be used if the e-mail is being sent via the interface of an S7-1500 CPU. • ≠0: SMTPS used to secure the connection before it is established (with CPs/CMs). You use the CertIndex parameter to specify the certificate to be used (see "Project navigation" > "Global security settings" > "Certificate manager").
WatchDogTime	TIME	Execution watchdog. Use this parameter to define the maximum execution time for the send operation. Note: Connection establishment can take longer (approx. one minute) if the connection is slow. When you specify the WATCH_DOG_TIME parameter, remember to allow for the time required to establish the connection. The connection is terminated once the specified time has elapsed.
MailServerAddress	STRING[254]	FQDN (Fully Qualified Domain Name) of the mail server. The mail server is addressed using the fully qualified domain name. Example: "www.mymailserver.com.".
UserName	STRING[254]	Mail server login name
PassWord	STRING[254]	Mail server password
From	Struct	E-mail sender address, which is defined using the following two STRING parameters. For example: "myname@mymailserver.com".
LocalPartPlusAtSign	STRING[64]	Local part of sender address, including @ sign. Example: "myname@".
FullQualifiedDomainName	STRING[254]	Fully Qualified Domain Name (FQDN for short) of the mail server. Example: "mymailserver.com".

DONE, BUSY and ERROR parameters

Description

The output parameters DONE, BUSY and ERROR are each displayed for only one cycle if the status of the BUSY output parameter changes from "1" to "0".

The following table shows the relationship between DONE, BUSY, and ERROR. Using this table, you can determine the current status of the instruction "TMAIL_C and when the sending of the e-mail is complete.

DONE	BUSY	ERROR	Description
0	1	0	The job is being processed.
1	0	0	Job successfully completed.
0	0	1	The job ended with an error. The cause of the error can be found in the STATUS (Page 5080) parameter.
0	0	0	The "TMAIL_C" instruction was not assigned a (new) job.

STATUS parameter

Description

The following table shows the return values of "TMAIL_C" at the STATUS parameter:

Return value STATUS* (W#16#...):	Explanation	Notes
0000	The processing of "TMAIL_C" was completed without errors.	Error-free completion of "TMAIL_C" does not mean that the e-mail sent will necessarily arrive. Incorrectly entering the recipient addresses does not generate a status error of the "TMAIL_C" instruction. In this case, there is no guarantee that the e-mail will be sent to other recipients, even if these were entered correctly.
7001	"TMAIL_C" is active (BUSY = 1).	First call: Job triggered.
7002	"TMAIL_C" is active (BUSY = 1).	Intermediate call: Job already active.
8xxx	The processing of "TMAIL_C" was completed with an error code of the communication instructions called internally.	For detailed information, please refer to the STATUS parameter descriptions for the "TCON (Page 5086)", "TDISCON (Page 5091)", "TSEND (Page 5096)" and "TRCV (Page 5099)" communication instructions.
8010	Error during connection establishment	You will find further information on evaluation in the SFB_STATUS parameter of the instance data block. The error code displayed at the SFB_STATUS parameter is explained in the STATUS parameter description for the "TCON (Page 5086)" instruction.
8011	Error sending the data	You will find further information on evaluation in the SFB_STATUS parameter of the instance data block. The error code displayed at the SFB_STATUS parameter is explained in the STATUS parameter description for the "TSEND (Page 5096)" instruction.

Return value STATUS* (W#16#...):	Explanation	Notes
8012	Error receiving the data	You will find further information on evaluation in the SFB_STATUS parameter of the instance data block. The error code displayed at the SFB_STATUS parameter is explained in the STATUS parameter description for the "TRCV (Page 5099)" instruction.
8013	Error during connection establishment	You will find further information on evaluation in the SFB_STATUS parameter of the instance data block. The error code displayed at the SFB_STATUS parameter is explained in the STATUS parameter description for the TCON (Page 5086)" and "TDISCON (Page 5091)" instructions.
8014	Establishment of a connection is not possible.	You may have entered an incorrect mail server IP address (MailServerAddress (Page 5077)) or too short a time span (WatchDogTime (Page 5077)) for connection establishment. It is also possible that the CPU has no connection to the network or that the CPU configuration is incorrect.
8015	Incorrect data type for MAIL_ADDR_PARAM	The only valid data types are the system data types (structures) Tmail_v4, Tmail_v6 and TMail_FQDN.
8016	Incorrect data type for the ATTACHMENT parameter	The only valid data types are ArrayOfByte, ArrayOfWord and ArrayOfDWord.
8017	Incorrect data length for the ATTACHMENT parameter	Data length must be <= 65534 bytes.
8401	No channels available. Possible cause: There is already an e-mail connection via the CP. A second connection cannot be established in parallel.	Specific error for CP 1543
8403	A TCP/IP connection to the mail server was not possible.	Specific error for CP 1543
8405	Server has denied login request.	Specific error for CP 1543
8406	SMTP client has detected an internal SSL error or a problem with the certificate structure.	Specific error for CP 1543
8407	Request to use SSL denied.	Specific error for CP 1543
8408	Client unable to identify socket for establishing a TCP/IP connection to the mail server.	Specific error for CP 1543
8409	Writing via the connection not possible. Possible cause: The communication partner has reset the connection or the connection has been lost.	Specific error for CP 1543

Return value STATUS* (W#16#...):	Explanation	Notes
8410	Reading via the connection not possible. Possible cause: The communication partner has reset the connection or the connection has been lost.	Specific error for CP 1543
8411	E-mail could not be sent. Cause: Insufficient memory space to execute the send operation.	Specific error for CP 1543
8412	DNS server configured was unable to resolve the specified domain name.	Specific error for CP 1543
8413	An internal error in the DNS subsystem prevented domain name resolution.	Specific error for CP 1543
8414	Empty character string entered as domain name.	Specific error for CP 1543
8415	An internal error has occurred in the cURL module. Execution was stopped.	Specific error for CP 1543
8416	An internal error has occurred in the SMTP module. Execution was stopped.	Specific error for CP 1543
8417	Request to SMTP on a channel already in use or invalid channel ID. Execution was stopped.	Specific error for CP 1543
8418	E-mail send job aborted. Possible causes: Execution time exceeded (WatchDogTime parameter) or CP transition from start to stop.	Specific error for CP 1543
8419	Channel interrupted and cannot be used before the connection is closed.	Specific error for CP 1543
8420	Server certificate string could not be verified with CP root certificate.	Specific error for CP 1543
8421	Internal error occurred. Execution was stopped.	Specific error for CP 1543
82xx, 84xx, or 85xx	The error message originates from the mail server and corresponds, except for the "8", to the error number of the SMTP protocol. The following lines list several error codes that can occur.	You will find more detailed information on the SMTP error code and other error codes in the SMTP protocol on the Internet or in the error documentation of the mail server. You can also view the most recent error message from the mail server in your instance DB in the BUFFER1 parameter. You will find the last data sent by the "TMAIL_C" instruction under DATEN in the instance DB.
8450	Action not executed: Mailbox not available/cannot be reached	Try again later.
8451	Action aborted: Local processing error	Try again later.

Return value STATUS* (W#16#...):	Explanation	Notes
8500	Syntax error: Error not recognized. This also includes the error when a command string is too long. This can also occur when the e-mail server does not support the LOGIN authentication procedure.	Check the parameters of "TMAIL_C". Try to send an e-mail without authentication. To do this, replace the content of the UserName parameter with an empty string. If no user name is specified, the LOGIN authentication procedure is not used.
8501	Syntax error: Incorrect input at a parameter	Possible cause: Incorrect address at the TO_S or CC parameter (see also: TO_S and CC parameters (Page 5076)).
8502	Command unknown or not implemented	Check your entries, in particular the FROM parameter. It may be incomplete and you may have forgotten the "@" or "." (see also: TO_S and CC parameters (Page 5076)).
8504	Command parameters not implemented	Check your entries, in particular the FROM parameter. It may be incomplete and you may have forgotten the "@" or "." (see also: TO_S and CC parameters (Page 5076)).
8535	SMTP authentication incomplete	You have possibly entered an incorrect user name or incorrect password.
8550	Mail server cannot be reached. You have no access rights.	You may have entered an incorrect user name or password, or the mail server may not support your login. Another cause of error could be a mistake in the domain name after the "@" or a missing "." at the TO_S, CC and FROM parameters (see also: TO_S and CC parameters (Page 5076)).
8552	Action aborted: Assigned memory size has been exceeded	Try again later.
8554	Transfer failed	Try again later.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

Other

TCON: Establishing a communication connection

TCON: Establish communication connection

Description

You use the "TCON" instruction to set up and establish a communication connection. Once the connection has been set up and established, it is automatically maintained and monitored by the CPU. "TCON" is executed asynchronously.

The connection data specified for the CONNECT and ID parameters are used to set up the communication connection. To establish the connection, a rising edge must be detected at the

REQ parameter. Once the connection is successfully established, the DONE parameter is set to "1".

Number of possible connections

For information on the number of possible communication connections, refer to the technical specifications for your CPU.

Connection with TCP and ISO-on-TCP

Both communication partners call the "TCON" instruction to set up and establish the communication connection. During parameter assignment, you specify which partner is the active communication end point and which is the passive one.

If the connection aborts, for example due to a line break or due to the remote communication partner, the active partner attempts to reestablish the configured connection. You do not have to call "TCON" again. This applies only if "TCON" has been executed successfully once (DONE = 1).

An existing connection is terminated and the connection set up is removed when the "TDISCON (Page 5091)" instruction is executed or when the CPU changes to STOP mode. To set up and establish the connection again, you will need to execute "TCON" again.

Parameters

The following table shows the parameters of the "TCON" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Starts the job to establish the connection specified in the ID upon a rising edge.
ID	Input	CONN_OUC	I, Q, M, D, L or constant	Reference to the assigned connection. Range of values: W#16#0001 to W#16#0FFF
CONNECT	InOut	TCON_Param	D	Pointer to the connection description See also: Auto-Hotspot
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or still in progress • 1: Job completed without error
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or already completed • 1: Job not yet completed. A new job cannot be started
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

BUSY, DONE and ERROR parameters

You can check the status of the job with the BUSY, DONE, ERROR and STATUS parameters. The BUSY parameter indicates the processing status. You use the DONE parameter to check whether or not a job has been executed successfully. The ERROR parameter is set when errors occurred during execution of "TCON". The error information is output at the STATUS parameter.

The following table shows the relationship between the BUSY, DONE and ERROR parameters:

BUSY	DONE	ERROR	Description
1	0	0	The job is being processed.
0	1	0	The job was completed successfully.
0	0	1	The job ended with an error. The cause of the error is output at the STATUS parameter.
0	0	0	No new job was assigned.

ERROR and STATUS parameters

ERROR	STATUS* (W#16#...)	Explanation
0	0000	Connection successfully established.
0	7000	No job processing active
0	7001	Start job execution, establish connection.
0	7002	Connection is being established (REQ irrelevant).
1	8085	Connection ID (ID parameter) is already being used by a configured connection.
1	8086	The ID parameter is outside the valid range.
1	8087	Maximum number of connections reached; no additional connection possible
1	8089	The CONNECT parameter does not point to a data block.
1	809A	The CONNECT parameter points to a field that does not correspond to the length of the connection description.
1	809B	The element InterfaceId within the TCON_xxx structure does not reference a hardware identifier of a CPU or CM/CP interface or has the value "0".
1	80A0	Group error for error codes W#16#80A1 and W#16#80A2.
1	80A2	Local or remote port is being used by the system.
1	80A3	Attempt being made to re-establish an existing connection.
1	80A4	IP address of the remote endpoint of the connection is invalid, in other words, it matches the IP address of the local partner.
1	80A5	Connection ID is already in use.
1	80A7	Communication error: You executed "TDISCON (Page 5091)" before "TCON" had completed.
1	80B2	The CONNECT parameter points to a data block that was generated with the attribute "Only store in load memory".

ERROR	STATUS* (W#16#...)	Explanation
1	80B4	You have violated one or more of the following conditions for passive connection establishment with the ISO-on-TCP protocol variant (connection_type = B#16#12): • local_tsap_id_len >= B#16#02 • local_tsap_id[1] = B#16#E0 • With local_tsap_id_len >= B#16#03, local_tsap_id[1] is an ASCII character. • local_tsap_id[1] is an ASCII character and local_tsap_id_len >= B#16#03.
1	80B5	Only passive connection establishment is permitted for connection type 13 = UDP.
1	80B6	Parameter assignment error at the connection_type parameter of the SDT TCON_Param.
1	80B7	Error in one of the following parameters of the data block for connection description: block_length, local_tsap_id_len, rem_subnet_id_len, rem_staddr_len, rem_tsap_id_len, next_staddr_len.
1	80B8	Connection description of the structure element ID and the block parameter ID are not identical.
1	80C3	All connection resources are in use.
1	80C4	Temporary communication error: • The connection cannot be established at this time. • The interface is currently receiving new parameters. • The configured connection is currently being removed by a "TDISCON (Page 5091)" instruction.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

TCON: Establish communication connection

Description

You use the "TCON" instruction to set up and establish a communication connection. Once the connection has been set up and established, it is automatically maintained and monitored by the CPU. "TCON" is executed asynchronously.

The connection data specified for the CONNECT and ID parameters are used to set up the communication connection.

- At the CONNECT parameter, if possible, only use the predefined structures as created by the connection parameter assignment in the Inspector window of the program editor.
- If the CONNECT parameter is not interconnected with a structure listed in the parameter table or if the structure includes an error, the error code 8089 is output at the STATUS parameter.

To establish the connection, a rising edge must be detected at the REQ parameter. Once the connection is successfully established, the DONE parameter is set to "1".

The instruction "TCON" V3.0 can also be used with CPU S7-1200, Version 4.0 or later.

Number of possible connections

For information on the number of possible communication connections, refer to the technical specifications for your CPU.

Connection with TCP and ISO-on-TCP

Both communication partners call the "TCON" instruction to set up and establish the communication connection. During parameter assignment, you specify which partner is the active communication end point and which is the passive one.

If the connection aborts, for example due to a line break or due to the remote communication partner, the active partner attempts to reestablish the configured connection. You do not have to call "TCON" again. This applies only if "TCON" has been executed successfully once (DONE = 1).

An existing connection is terminated and the connection set up is removed when the "TDISCON (Page 5091)" instruction is executed or when the CPU changes to STOP mode. To set up and establish the connection again, you will need to execute "TCON" again.

Parameters

The following table shows the parameters of the "TCON" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Starts the job to establish the connection specified on a rising edge.
ID	Input	CONN_OUC	I, Q, M, D, L or constant	Reference to the assigned connection. Range of values: W#16#0001 to W#16#0FFF
CONNECT	InOut	VARIANT	D	Pointer to the connection description • For TCP or UDP, use the structure TCON_IP_v4 For description, refer to: Auto-Hotspot • For ISO-on-TCP, use the structure TCON_IP_RFC For description, refer to: Auto-Hotspot • For ISO, use the structure TCON_ISOnative (only for CP1543-1) For description, refer to: "Structure of the connection description according to TCON_ISOnative"
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or still in progress • 1: Job completed without error
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or already completed • 1: Job not yet completed. A new job cannot be started
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

BUSY, DONE and ERROR parameters

You can check the status of the job with the BUSY, DONE, ERROR and STATUS parameters. The BUSY parameter indicates the processing status. You use the DONE parameter to check whether or not a job has been executed successfully. The ERROR parameter is set when errors occurred during execution of "TCON". The error information is output at the STATUS parameter.

The instruction "TCON" generates an error message in version 3.0 if an active connection establishment to a remote partner fails. Create a rising edge at the REQ parameter to establish a connection once again.

The following table shows the relationship between the BUSY, DONE and ERROR parameters:

BUSY	DONE	ERROR	Description
1	0	0	The job is being processed.
0	1	0	The job was completed successfully.
0	0	1	The job ended with an error. The cause of the error is output at the STATUS parameter.
0	0	0	No new job was assigned.

Structure of the connection description according to TCON_ISOnative

A connection description DB with a structure according to TCON_ISOnative is used to assign the communication connection parameters for ISO. The fixed data structure of the TCON_ISOnative contains all parameters that are required to establish the connection.

Byte	Parameter	Data type	Start value	Description
0 … 1	InterfaceId	HW_ANY	64	Hardware identifier of the local interface (value range: 0 to 65535).
2 … 3	ID	CONN_OUC	1	Reference to this connection (unique ID in the value range: 1 to 4095).
4	ConnectionType	BYTE	16#16	Connection type: ISO
5	ActiveEstablished	BOOL	TRUE	ID for the manner in which the connection is established: • FALSE: Passive connection establishment • TRUE: Active connection establishment
8 … 13	RemoteMacAddress	ARRAY [1..6] of BYTE	-	MAC address of the partner end point, for example, for 00-74-41-FD-AE-84: • MacAddr[1] = 00 • MacAddr[2] = 74 • MacAddr[3] = 41 • MacAddr[4] = FD • MacAddr[5] = AE • MacAddr[6] = 84

Byte	Parameter	Data type	Start value	Description
14 .. . 19	LocalMacAddress	ARRAY [1..6] of BYTE	-	MAC address of the local end point, for example, for 00-74-41-FD-AE-84: • MacAddr[1] = 00 • MacAddr[2] = 74 • MacAddr[3] = 41 • MacAddr[4] = FD • MacAddr[5] = AE • MacAddr[6] = 84
20 .. . 53	RemoteTSelector	Struct	-	TSelector of the remote connection partner: • Bytes 20 to 21 = TSelLength • Bytes 22 to 53: = TSel[1-32]
	TSelLength	UINT	-	Value range from 0 to 32
	TSel	ARRAY [1..32] of BYTE	-	Value range in each case 0 to 255 in bytes
54 .. . 87	LocalTSelector	Struct	-	TSelector of the remote connection partner: • Bytes 20 to 21 = TSelLength • Bytes 22 to 53: = TSel[1-32]
	TSelLength	UINT		Value range from 0 to 32
	TSel	ARRAY [1..32] of BYTE	-	Value range in each case 0 to 255 in bytes
88 … 89	CrRetransmitionTime	UINT	-	Duration until connection attempt is repeated in seconds.
90 .. . 91	DataRetransmitionTime	UINT	100 ms	Duration until data transfer is repeated in milliseconds.
92 .. . 93	MaxRetransmitionCount	UINT	-	Maximum number of repetitions.
94 .. . 95	InactivityTime	UINT	-	in seconds
96 .. . 97	WindowTime	UINT		in seconds

ERROR and STATUS parameters

ERROR	STATUS* (W#16#...)	Explanation
0	0000	Connection successfully established.
0	7000	No job processing active
0	7001	Start job execution, establish connection.
0	7002	Connection is being established (REQ irrelevant).
1	8085	Connection ID (ID parameter) is already being used by a configured connection.
1	8086	The ID parameter is outside the valid range.
1	8087	Maximum number of connections reached; no additional connection possible
1	8089	The CONNECT parameter does not point to a connection description or the connection description was created manually.

ERROR	STATUS* (W#16#...)	Explanation
1	809A	The structure at the CONNECT parameter is not supported or the length is invalid.
1	809B	The element InterfaceId within the TCON_xxx structure does not reference a hardware identifier of a CPU or CM/CP interface or has the value "0".
1	80A2	Local or remote port is being used by the system. The following ports are reserved locally: 20, 21, 80, 102, 135, 161, 162, 443, 34962, 34963, 34964 as well as the area 49152 to 65535.
1	80A3	ID is used by a connection created by the user program, which uses the same connection description at the CONNECT parameter.
1	80A4	IP address of the remote endpoint of the connection is invalid or it corresponds to the IP address of the local partner.
1	80A7	Communication error: You executed "TDISCON (Page 5091)" before "TCON" had completed.
1	80B4	Only with TCON_IP_RFC: The local T selector was not specified or the first byte does not contain the value 0x0E (only with a length of T selector = 2) or the local T selector starts with "SIMATIC-".
1	80B5	Only passive connection establishment is permitted for connection type 13 = UDP (parameter ActiveEstablished of the structure TCON_IP_v4 / TCON_PARAM has the value TRUE).
1	80B6	Parameter assignment error in the ConnectionType parameter of the data block for connection description. • Only valid with TCON_IP_v4: 0x11, 0x0B and 0x13. • Only valid with TCON_IP_RFC: 0x0C and 0x12
1	80B7	With TCON_IP_v4: • TCP (active connection establishment): Remote port is "0". • TCP (passive connection establishment): Local port is "0". • UDP: Local port is "0". With TCON_IP_RFC: • Local (LocalTSelector) or remote (RemoteTSelector) T selector was specified with a length of more than 32 bytes. • For TSelLength of the T selector (local or remote), a length greater than 32 was entered. • Error in the length of the IP address of the specific connection partner.
1	80B8	Parameter ID in the local connection description (structure at CONNECT parameter) and parameter ID of the instruction are different.
1	80C3	All connection resources are in use.
1	80C4	Temporary communication error: • The connection cannot be established at this time. • The interface is currently receiving new parameters. • The configured connection is currently being removed by a "TDISCON (Page 5091)" instruction.
1	80C5	The connection partner refuses to establish the connection, has terminated the connection or actively ended it.
1	80C6	The connection partner cannot be reached (network error).
1	80C7	Execution timeout.
1	80C8	Value at the ID parameter is already being used by a connection that was created using the user program. The connection uses the identical ID, but different connection settings at the parameter CONNECT.
1	80C9	Validation of the connection partner failed. The connection partner that wants to establish the connection does not match the defined partner of the structure at the CONNECT parameter.

ERROR	STATUS* (W#16#...)	Explanation
1	80CE	The IP address of the local interface is 0.0.0.0.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

TDISCON: Terminate communication connection

Description

The "TDISCON" instruction terminates a communications connection from the CPU to a connection partner.

Functional description

"The TDISCON" instruction works asynchronously, which means its job processing extends over multiple calls. You start the job for terminating a connection by calling the "TDISCON" instruction with REQ = 1.

After "TDISCON" has been successfully executed, the ID specified for "TCON" is no longer valid and cannot be used for sending or receiving.

The job status is indicated by the output parameters BUSY and STATUS. Here, STATUS corresponds to the output parameter RET_VAL of the asynchronous instructions (see also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 3150)).

The following table shows the relationship between BUSY, DONE and ERROR. Using this table, you can recognize the current status of "TDISCON" or when the establishment of the connection is completed.

BUSY	DONE	ERROR	Description
1	0	0	The job is being processed.
0	1	0	Job successfully completed.
0	0	1	The job ended with an error. The cause of the error can be found in the STATUS parameter.
0	0	0	The instruction was not assigned a (new) job.

Parameters

The following table shows the parameters of the instruction "TDISCON":

Parameter	Declaration	Data type	Memory area		Description
			S7-1200	S7-1500	
REQ	Input	BOOL	I, Q, M, D, L or constant	I, Q, M, D, L or constant	Control parameter REQUEST starts the job for terminating the connection specified by ID. The job starts on a rising edge.
ID	Input	CONN_OUC (WORD)	D, L or constant	I, Q, M, D, L or constant	Reference to the connection to be terminated (connection ID) Range of values: W#16#0001 to W#16#0FFF
DONE	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Status parameter: • 0: Job not yet started or still executing. • 1: Job executed without errors
BUSY	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Status parameter: • BUSY = 1: The job is not yet completed. • BUSY = 0: The job is completed or has not started yet.
ERROR	Output	BOOL	I, Q, M, D, L	I, Q, M, D, L	Status parameter: • ERROR=0: No error. • ERROR=1: Error occurred during processing. STATUS supplies detailed information on the type of error
STATUS	Output	WORD	I, Q, M, D, L	I, Q, M, D, L	Status parameter: Error information (see "Parameters ERROR and STATUS")

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

Parameters ERROR and STATUS

ERROR	STATUS* (W#16#...)	Explanation
0	0000	Connection terminated successfully.
0	7000	No job processing active.
0	7001	Start of job processing, connection being terminated.
0	7002	Intermediate call (REQ irrelevant), connection being terminated.
1	8086	The ID parameter is not in the permitted value range.
1	80A3	Attempt is being made to terminate a non-existent connection or the connection is already terminated.

ERROR	STATUS* (W#16#...)	Explanation
1	80C4	Temporary communication error: New parameters are being assigned to the interface or the connection is currently being established.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

TSEND: Send data via communication connection

TSEND: Send data via communication connection

Description

The following description of the "TSEND" instruction is valid for the CPU S7-1200 to version 3.0.

You use the "TSEND" instruction to send data over an existing communication connection. "TSEND" is executed asynchronously.

You specify the send area with the DATA parameter. This includes the address and the length of the data to be sent. All data types except BOOL and Array of BOOL can be used for the data to be sent.

The send job is executed when a rising edge is detected at the REQ parameter.

With the LEN parameter, you specify the maximum number of bytes sent with a send job.

- When data is transferred with TCP (streaming protocol), the "TSEND" instruction provides no information about the length of the data sent to "TRCV (Page 5099)".
- When data is transferred with ISO-on-TCP (message-oriented protocol), the length of the data sent is communicated to "TRCV (Page 5099)". The amount of data sent with "TSEND" as packet must also be received again at the receiving end ("TRCV (Page 5099)"):
  - If the receive buffer is too small for the sent data, an error occurs at the receiving end.
  - If the receive buffer is sufficiently large, "TRCV" returns with DONE=1 as soon as the data packet is received.

The data to be sent must not be edited until the send job is completed. If the send job executes successfully, the DONE parameter is set to "1". Signal state "1" at the DONE parameter is not confirmation that the data sent has already been read by the communication partner.

Parameters

The following table shows the parameters of the "TSEND" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Starts the send job on a rising edge.
ID	Input	CONN_OUC	I, Q, M, D, L or constant	Reference to the connection established with "TCON". Range of values: W#16#0001 to W#16#0FFF

Parameter	Declaration	Data type	Memory area	Description
LEN	Input	UINT	I, Q, M, D, L or constant	Maximum number of bytes to be sent with the job.
DATA	InOut	VARIANT	I, Q, M, D	Pointer to the send area containing the address and the length of the data to be sent. The address references: • The process image of the inputs • The process image of the outputs • A bit memory • A data block
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or still in progress • 1: Job completed without error
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or already completed • 1: Job not yet completed. A new job cannot be started.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

LEN and DATA parameters

- With LEN = 0 , all the data specified with the DATA parameter is sent.
- If the number of bytes at the LEN parameter is larger than the length of the data to be sent that was defined with the DATA parameter, the error code 8088 is output at the STATUS parameter (see description of the STATUS parameter below).
- If a structure (Struct) is referenced via the DATA parameter, LEN can be shorter than the structure. In this case, only the data up to the length of the LEN parameter is transferred.
- With data types STRING and WSTRING, all data is transferred if the parameter LEN = 0. When LEN > 0, the length must be at least the maximum number of bytes plus two additional bytes which contain the length information. You can find more detailed information on the structure of the data types in: "Overview of the valid data types (Page 2236)".
- The maximum number of bytes that can be transmitted is 65534.
- When you use structured tags from optimized DBs, the address of the structured tag should be interconnected at the DATA parameter and the parameter LEN = 0. This guarantees type-safe transmission of the entire structure if the same structure is used at the receiving end.

BUSY, DONE and ERROR parameters

You can check the status of the job with the BUSY, DONE, ERROR and STATUS parameters. The BUSY parameter indicates the processing status. With the DONE parameter, you can check whether or not a job executed successfully. The ERROR parameter is set when errors occurred during execution of "TSEND". The error information is output at the STATUS parameter.

The following table shows the relationship between the BUSY, DONE and ERROR parameters:

BUSY	DONE	ERROR	Description
1	0	0	The job is being processed.
0	1	0	The job was completed successfully.
0	0	1	The job ended with an error. The cause of the error is specified in the STATUS parameter.
0	0	0	No new job was assigned.

Note

Because "TSEND" is executed asynchronously, you must keep the data in the send area consistent until the DONE parameter or the ERROR parameter changes to the value "1".

ERROR and STATUS parameters

ERROR	STATUS* (W#16#...)	Description
0	0000	Send job completed without error.
0	7000	No job processing active.
0	7001	Start of job execution, data is being sent. When processing this job, the operating system accesses the data in the DATA send area.
0	7002	Job executing (REQ irrelevant). When processing this job, the operating system accesses the data in the DATA send area.
1	8085	• The LEN parameter is greater than the highest permitted value (65536). • DATA and LEN parameters both have the value "0".
1	8086	The ID parameter is outside the permitted address range (1..0xFFF).
1	8088	The LEN parameter is greater than the area specified in DATA.
1	80A1	Communication error: • The specified connection has not yet been established. • The specified connection is being terminated. Transfer via this connection is not possible. • The interface is being re-initialized.
1	80B3	The protocol variant (ConnectionType parameter in the connection description) is set to UDP. Use the instruction "TUSEND" with a UDP connection.
1	80C3	• A block with this ID is already being processed in a different priority group. • Internal lack of resources.

ERROR	STATUS* (W#16#...)	Description
1	80C4	Temporary communication error: • The connection cannot be established to the partner at this time. • The interface is receiving new parameter settings or the connection is being established.
1	80C5	Connection terminated by the communication partner.
1	80C6	Network error. Communication partner cannot be reached.
1	80C7	Execution timeout.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

TSEND: Send data via communication connection

Description

The following description of the "TSEND" instruction is valid for the CPU S7-1500 and S7-1200 V4.0.

You use the "TSEND" instruction to send data over an existing communication connection. "TSEND" is executed asynchronously.

You specify the send area with the DATA parameter. This includes the address and the length of the data to be sent. All data types except BOOL and Array of BOOL can be used for the data to be sent.

The send job is executed when a rising edge is detected at the REQ parameter.

With the LEN parameter, you specify the maximum number of bytes sent with a send job.

- When data is transferred with TCP (streaming protocol), the "TSEND" instruction provides no information about the length of the data sent to "TRCV (Page 5099)".
- When data is transferred with ISO-on-TCP (message-oriented protocol), the length of the data sent is communicated to "TRCV (Page 5099)". The amount of data sent with "TSEND" as packet must also be received again at the receiving end ("TRCV (Page 5099)"):
  - If the receive buffer is too small for the sent data, an error occurs at the receiving end.
  - If the receive buffer is sufficiently large, "TRCV" returns with DONE=1 as soon as the data packet is received.

The data to be sent must not be edited until the send job is completed. If the send job executes successfully, the DONE parameter is set to "1". Signal state "1" at the DONE parameter is not confirmation that the data sent has already been read by the communication partner.

Parameters

The following table shows the parameters of the "TSEND" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Starts the send job on a rising edge.
ID	Input	CONN_OUC	I, Q, M, D, L or constant	Reference to the connection established with "TCON". Range of values: W#16#0001 to W#16#0FFF
LEN	Input	UDINT	I, Q, M, D, L or constant	Maximum number of bytes to be sent with the job.
DATA	InOut	VARIANT	I, Q, M, D	Pointer to the send area containing the address and the length of the data to be sent. The address references: • The process image of the inputs • The process image of the outputs • A bit memory • A data block
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or still in progress • 1: Job completed without error
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or already completed • 1: Job not yet completed. A new job cannot be started.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

LEN and DATA parameters

- With LEN = 0 , all the data specified with the DATA parameter is sent.
- If the number of bytes at the LEN parameter is larger than the length of the data to be sent that was defined with the DATA parameter, the error code 8088 is output at the STATUS parameter (see description of the STATUS parameter below).
- If a structure (Struct) is referenced via the DATA parameter, LEN can be shorter than the structure. In this case, only the data up to the length of the LEN parameter is transferred.
- With data types STRING and WSTRING, all data is transferred if the parameter LEN = 0. When LEN > 0, the length must be at least the maximum number of bytes plus two additional bytes which contain the length information. You can find more detailed information on the structure of the data types in: "Overview of the valid data types (Page 2236)".

- The maximum number of bytes that can be transmitted depends on the device.
- When you use structured tags from optimized DBs, the address of the structured tag should be interconnected at the DATA parameter and the parameter LEN = 0. This guarantees type-safe transmission of the entire structure if the same structure is used at the receiving end.

BUSY, DONE and ERROR parameters

You can check the status of the job with the BUSY, DONE, ERROR and STATUS parameters. The BUSY parameter indicates the processing status. With the DONE parameter, you can check whether or not a job executed successfully. The ERROR parameter is set when errors occurred during execution of "TSEND". The error information is output at the STATUS parameter.

The following table shows the relationship between the BUSY, DONE and ERROR parameters:

BUSY	DONE	ERROR	Description
1	0	0	The job is being processed.
0	1	0	The job was completed successfully.
0	0	1	The job ended with an error. The cause of the error is specified in the STATUS parameter.
0	0	0	No new job was assigned.

Note

Because "TSEND" is executed asynchronously, you must keep the data in the send area consistent until the DONE parameter or the ERROR parameter changes to the value "1".

ERROR and STATUS parameters

ERROR	STATUS* (W#16#...)	Description
0	0000	Send job completed without error.
0	7000	No job processing active.
0	7001	Start of job execution, data is being sent. When processing this job, the operating system accesses the data in the DATA send area.
0	7002	Job executing (REQ irrelevant). When processing this job, the operating system accesses the data in the DATA send area.
1	8085	• The LEN parameter is greater than the highest permitted value (65536). • DATA and LEN parameters both have the value "0".
1	8086	The ID parameter is outside the permitted address range (1..0xFFF).
1	8088	The LEN parameter is greater than the area specified in DATA.
1	80A1	Communication error: • The specified connection has not yet been established. • The specified connection is being terminated. Transfer via this connection is not possible. • The interface is being re-initialized.

ERROR	STATUS* (W#16#...)	Description
1	80B3	The protocol variant (ConnectionType parameter in the connection description) is set to UDP. Use the instruction "TUSEND" with a UDP connection.
1	80C3	• A block with this ID is already being processed in a different priority group. • Internal lack of resources.
1	80C4	Temporary communication error: • The connection cannot be established to the partner at this time. • The interface is receiving new parameter settings or the connection is being established.
1	80C5	Connection terminated by the communication partner.
1	80C6	Network error. Communication partner cannot be reached.
1	80C7	Execution timeout.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

TRCV: Receive data via communication connection

TRCV: Receive data via communication connection

Description

The following description of the "TRCV" instruction is valid for the CPU S7-1200 up to version 3.0.

You use the "TRCV" instruction to receive data over an existing communication connection. "TRCV" is executed asynchronously.

Receipt of data is enabled when the EN_R parameter is set to the value "1". The received data is entered in a receive area. You specify the length of the receive area either with the LEN parameter (if LEN <> 0) or with the length information of the DATA parameter (if LEN = 0), depending on the protocol variant being used.

While data is being received, you cannot make changes to the DATA parameter or the defined receive area to ensure consistency of the received data.

After successful receipt of data, the NDR parameter is set to the value "1". You can query the amount of data actually received at the RCVD_LEN parameter.

Receive modes of "TRCV"

The following table shows how the received data is entered in the receive area.

Protocol variant	Availability of data in the receive area	connection_type* parameter of the connection description	LEN parameter
TCP (Ad-hoc mode)	The data is immediately available.	Hexadecimal value: B#16#11 Integer value: 17	0
TCP (Receipt of data with specified length)	The data is available as soon as the data length specified at the LEN parameter has been fully received.	Hexadecimal value: B#16#11 Integer value: 17	1 to 8192
ISO on TCP (Message-oriented data transfer)	The data is available as soon as the data length specified at the LEN parameter has been fully received.	Hexadecimal value: B#16#12 Integer value: 18	• 1 to 1452, if a CP is used. • 1 to 8192, if no CP is used.
* See "Auto-Hotspot".			

TCP (Ad-hoc mode)

The ad-hoc mode is only available with the TCP protocol variant. You use the ad-hoc mode to receive data with dynamic length with the "TRCV" instruction.

You set ad-hoc mode by assigning the value "0" to the LEN parameter. All data types can be used for data blocks with standard access when you use ad-hoc mode. Only ARRAY of BYTE or data types with a length of 8 bits can be used for data blocks with optimized access (e.g., CHAR, USINT, SINT, etc.). When ad-hoc mode is active, receipt of data is already signaled after a received byte at the NDR parameter.

TCP (Receipt of data with specified length)

For receipt of data with a specified length, enter the length of the data at the LEN parameter. The data receipt is not complete until the length of data specified at the LEN parameter has been completely received. Only then is the data available in the receive area (DATA parameter). The receipt of data is signaled by the NDR output parameter. The actually received data length in bytes at the RCVD_LEN parameter corresponds to the data length at the LEN parameter after receipt.

ISO on TCP (Message-oriented data transfer)

Complete message blocks are sent via a connection with the protocol variant ISO on TCP; these are recognized as such by the recipient. When using ISO on TCP, "TRCV" signals data receipt as soon as the message block has been completely received. The receive area is defined by the LEN and DATA parameters. If the receive buffer (DATA parameter) is too small for the sent data, "TRCV" signals an error. The successful receipt of data is signaled by the NDR output parameter. The actually received data length in bytes at the RCVD_LEN parameter corresponds to the data length at the LEN parameter after receipt.

Parameters

The following table shows the parameters of the "TRCV" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L or constant	Receive enable
ID	Input	CONN_OUC	I, Q, M, D, L or constant	Reference to the connection established with "TCON (Page 5083)". Range of values: W#16#0001 (1) to W#16#0FFF (4095)
LEN	Input	UDINT	I, Q, M, D, L or constant	Length of the receive area in bytes (hidden). If you use a memory area with optimized access at the DATA parameter, the LEN parameter must have the value "0".
DATA	InOut	VARIANT	I, Q, M, D	Pointer to the receive area
NDR	Output	BOOL	I, Q, M, D, L	Status parameter (New Data Received): • 0: Job not yet started or still in progress • 1: Job completed without error
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter: • 0: Job not yet started or already completed • 1: Job not yet completed. A new job cannot be started
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter: • 0: No error. • 1: An error occurred during execution of the instruction. Detailed information is output via the STATUS parameter.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter: Output of status and error information.
RCVD_LEN	Output	UDINT	I, Q, M, D, L	Amount of data actually received in bytes

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

LEN, DATA and RCVD_LEN parameters

- If LEN = 0, the received data is saved in the receive area specified at the DATA parameter. The number of bytes received is indicated at the RCVD_LEN parameter.
- If the length specified at the LEN parameter is greater than the length of the data received at the DATA parameter, the error code 8088 is output at the STATUS parameter (see description of the STATUS) parameter in the following.
- If a structure (Struct) is referenced via the DATA parameter, LEN can be shorter than the structure. In this case, only the data up to the length of the LEN parameter is transferred.
- If the DATA parameter references a data block with optimized access, the LEN parameter must be set to "0".

- If a STRING data type is referenced via the DATA parameter, the length specified at the LEN parameter must be 0 or >=2 (LEN = 1 is not permitted).
- If a WSTRING data type is referenced via the DATA parameter, the length specified at the LEN parameter must be 0 or >=5.

BUSY, NDR and ERROR parameters

You can check the status of the job with the BUSY, NDR, ERROR and STATUS parameters. The BUSY parameter indicates the processing status. With the NDR parameter, you can check whether or not a job executed successfully. The ERROR parameter is set when errors occurred during execution of TRCV. The error information is output at the STATUS parameter.

The following table shows the relationship between the BUSY, NDR and ERROR parameters:

BUSY	NDR	ERROR	Description
1	-	-	The job is being processed.
0	1	0	The job was completed successfully.
0	0	1	The job ended with an error. The cause of the error is output at the STATUS parameter.
0	0	0	No new job was assigned.

Note

Because "TRCV" is executed asynchronously, the data in the receive area is only consistent when the NDR parameter is set to the value "1".

ERROR and STATUS parameters

ERROR	STATUS* (W#16#...)	Explanation
0	0000	Job completed successfully. The current length of the received data is output at the RCVD_LEN parameter.
0	7000	Block not ready to receive.
0	7001	Block is ready to receive, receive job was activated.
0	7002	Interim call, the receive job is executing. Note: While the job is being processed, data is written to the receive area. Access to the receive area during this time may provide inconsistent data.
1	8085	• The LEN parameter is larger than the highest permitted value. • The value of the LEN or DATA parameter was changed after the first call. • Both LEN parameters and the DATA parameter have the value "0" or LEN is longer than the maximum permitted value (65536).
1	8086	The ID parameter is outside the permitted address range (1 .. 0x0FFF).
1	8088	• Receive area is too small. • The value at the LEN parameter is larger than the receive area set at the DATA parameter.

ERROR	STATUS* (W#16#...)	Explanation
1	80A1	Communication error: • The specified connection has not yet been established. • The specified connection is being terminated. Receive job over this connection is not possible. • The connection is being re-initialized.
1	80B3	The protocol variant (connection_type parameter in the connection description) is set to UDP. Use the instruction "TURCV" with a UDP connection.
1	80C3	• A block with this ID is already being processed in a different priority group. • Internal lack of resources.
1	80C4	Temporary communication error: • The connection cannot be established to the partner at this time. • The interface is receiving new parameter settings or the connection is being established.
1	80C5	The remote partner has terminated the connection.
1	80C6	The remote partner cannot be reached (network error).
1	80C7	Execution timeout.
1	80C9	The length of the receive area is smaller than the length of the sent data.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

TRCV: Receive data via communication connection

Description

The following description of the "TRCV" instruction is valid for the CPU S7-1500 and S7-1200 V4.0.

You use the "TRCV" instruction to receive data over an existing communication connection. "TRCV" is executed asynchronously.

Receipt of data is enabled when the EN_R parameter is set to the value "1". The received data is entered in a receive area. You specify the length of the receive area either with the LEN parameter (if LEN <> 0) or with the length information of the DATA parameter (if LEN = 0), depending on the protocol variant being used.

While data is being received, you cannot make changes to the DATA parameter or the defined receive area to ensure consistency of the received data.

After successful receipt of data, the NDR parameter is set to the value "1". You can query the amount of data actually received at the RCVD_LEN parameter.

Receive modes of "TRCV"

The following table shows how the received data is entered in the receive area.

Protocol variant	ADHOC parameter	Availability of data in the receive area	connection_type parameter of the connection description	LEN parameter
TCP (Ad-hoc mode)	1 (ad-hoc activated)	The data is immediately available.	Hexadecimal value: B#16#11 Integer value: 17	1 up to maximum length (depending on the CPU)
TCP (Receipt of data with specified length)	0 (ad-hoc deactivated)	The data is available as soon as the data length specified at the LEN parameter has been fully received.	Hexadecimal value: B#16#11 Integer value: 17	1 to 8192
ISO on TCP (Message-oriented data transfer)	-	The data is available as soon as the data length specified at the LEN parameter has been fully received.	Hexadecimal value: B#16#12 Integer value: 18	• 1 to 1452, if a CP is used. • 1 to 8192, if no CP is used.

TCP (Ad-hoc mode)

The ad-hoc mode is only available with the TCP protocol variant. You use the ad-hoc mode to receive data with dynamic length with the "TRCV" instruction.

You set ad-hoc mode by assigning the value "1" to the ADHOC parameter. All data types can be used for data blocks with standard access when you use ad-hoc mode. Only ARRAY of BYTE or data types with a length of 8 bits can be used for data blocks with optimized access (e.g., CHAR, USINT, SINT, etc.). When ad-hoc mode is active, receipt of data is already signaled after a received byte at the NDR parameter.

TCP (Receipt of data with specified length)

Assign the value "0" to the ADHOC parameter for receipt of data with specified length. If ad-hoc mode is disabled, the data receipt is not complete until the length of data specified at the LEN parameter has been completely received. Only then is the data available in the receive area (DATA parameter). The successful receipt of data is signaled by the NDR output parameter. The actually received data length in bytes at the RCVD_LEN parameter corresponds to the data length at the LEN parameter after receipt.

ISO on TCP (Message-oriented data transfer)

Complete message blocks are sent via a connection with the protocol variant ISO on TCP; these are recognized as such by the recipient. When using ISO on TCP, "TRCV" signals data receipt as soon as the message block has been completely received. The receive area is defined by the LEN and DATA parameters. If the receive buffer (DATA parameter) is too small for the sent data, "TRCV" signals an error. The successful receipt of data is signaled by the NDR output parameter. The actually received data length in bytes at the RCVD_LEN parameter corresponds to the data length at the LEN parameter after receipt.

Parameters

The following table shows the parameters of the "TRCV" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L or constant	Receive enable
ID	Input	CONN_OUC	I, Q, M, D, L or constant	Reference to the connection established with "TCON". Range of values: W#16#0001 to W#16#0FFF
LEN	Input	UDINT	I, Q, M, D, L or constant	Length of the receive area in bytes (hidden). If you use a receive area with optimized access at the DATA parameter, the LEN parameter must have the value "0".
ADHOC	Input	BOOL	I, Q, M, D, L or constant	Use ad-hoc mode for the TCP protocol variant (hidden).
DATA	InOut	VARIANT	I, Q, M, D	Pointer to the receive area
NDR	Output	BOOL	I, Q, M, D, L	Status parameter (New Data Received): • 0: Job not yet started or still in progress • 1: New data received
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or already completed • 1: Job not yet completed. A new job cannot be started
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction
RCVD_LEN	Output	UDINT	I, Q, M, D, L	Amount of data actually received in bytes

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

LEN, DATA and RCVD_LEN parameters

- If LEN = 0, the received data is saved in the receive area specified at the DATA parameter. The number of bytes received is indicated at the RCVD_LEN parameter.
- If the length specified at the LEN parameter is greater than the length of the data received at the DATA parameter, the error code 8088 is output at the STATUS parameter (see description of the STATUS) parameter in the following.
- If a structure (Struct) is referenced via the DATA parameter, LEN can be shorter than the structure. In this case, only the data up to the length of the LEN parameter is transferred.
- If the DATA parameter references a data block with optimized access, the LEN parameter must be set to "0". If the length of the data does not match for elementary data types, the data will not be received and the error code 8088 is output at the STATUS parameter.

- If a STRING data type is referenced via the DATA parameter, the length specified at the LEN parameter must be 0 or >=2 (LEN = 1 is not permitted).
- If a WSTRING data type is referenced via the DATA parameter, the length specified at the LEN parameter must be 0 or >=5.

BUSY, NDR and ERROR parameters

You can check the status of the job with the BUSY, NDR, ERROR and STATUS parameters. The BUSY parameter indicates the processing status. With the NDR parameter, you can check whether or not a job executed successfully. The ERROR parameter is set when errors occurred during execution of TRCV. The error information is output at the STATUS parameter.

The following table shows the relationship between the BUSY, NDR and ERROR parameters:

BUSY	NDR	ERROR	Description
1	-	-	The job is being processed.
0	1	0	The job was completed successfully.
0	0	1	The job ended with an error. The cause of the error is output at the STATUS parameter.
0	0	0	No new job was assigned.

Note

Because "TRCV" is executed asynchronously, the data in the receive area is only consistent when the NDR parameter is set to the value "1".

ERROR and STATUS parameters

ERROR	STATUS* (W#16#...)	Explanation
0	0000	Job completed successfully. The current length of the received data is output at the RCVD_LEN parameter.
0	7000	Block not ready to receive.
0	7001	Block is ready to receive, receive job was activated.
0	7002	Interim call, the receive job is executing. Note: While the job is being processed, data is written to the receive area. Access to the receive area during this time may provide inconsistent data.
1	8085	• The LEN parameter is larger than the highest permitted value. • The value of the LEN or DATA parameter was changed after the first call. • Both LEN parameters and the DATA parameter have the value "0" or LEN is longer than the maximum permitted value (65536).
1	8086	The ID parameter is outside the permitted value range (1 .. 0x0FFF).
1	8088	• Receive area is too small. • The value at the LEN parameter is larger than the receive area set at the DATA parameter.

ERROR	STATUS* (W#16#...)	Explanation
1	80A1	Communication error: • The specified connection has not yet been established. • The specified connection is being terminated. Receive job over this connection is not possible. • The connection is being re-initialized.
1	80B3	The protocol variant (connection_type parameter in the connection description) is set to UDP. Use the instruction "TURCV" with a UDP connection.
1	80C3	• A block with this ID is already being processed in a different priority group. • Internal lack of resources.
1	80C4	Temporary communication error: • The connection cannot be established to the partner at this time. • The interface is receiving new parameter settings or the connection is being established.
1	80C5	The remote partner has terminated the connection.
1	80C6	The remote partner cannot be reached (network error).
1	80C7	Execution timeout.
1	80C9	The length of the receive area is smaller than the length of the sent data.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

See also

TRCV: Receive data via communication connection (Page 5099)

TCON: Establish communication connection (Page 5086)

Structure of the address information for the remote partner with UDP

Overview

When using a UDP connection, the address information for the remote partner is stored in the system data type TADDR_Param:

- By using the instruction "TUSEND (Page 5109)" the address information of the recipient is assigned to the parameter ADDR via TADDR_Param.
  The stored address data of the remote partner are read from the system data type by the instruction.
- By using the instruction "TURCV (Page 5112)" the address of the sender is received at the parameter ADDR via TADDR_Param.
  The address data is written by the instruction in the system data type.

Structure of the address information according to TADDR_Param

The system data type TADDR_Param contains the address information of the remote partner and consists of the IP address and the port number.

The system data type TADDR_Param has the following structure:

Byte	Parameter	Data type	Start value	Description
0 to 3	rem_ip_addr	ARRAY [1..4] of USINT	B#16#00 ...	IP address of the remote partner, e.g. 192.168.002.003: • rem_ip_addr[1] = B#16#C0 (192) • rem_ip_addr[2] = B#16#A8 (168) • rem_ip_addr[3] = B#16#02 (002) • rem_ip_addr[4] = B#16#03 (003) The IP address can be taken from the Devices & networks view in the interface properties of the remote partner. As an alternative these are also displayed in the properties of the UDP connection under Address details.
4 to 5	rem_port_nr	UINT	B#16#00 ...	Remote port-number (possible values see: Auto-Hotspot): • rem_port_nr[1] = high byte of the port no. in hexadecimal notation • rem_port_nr[2] = low byte of port no. in hexadecimal notation The port number can be taken from the Devices & networks view in the UDP connection properties. The port number is indicated under Address details as a decimal value. Example: Port number = 2000 (decimal) / W#16#07D0 (hexadecimal) • rem_port_nr[1] = 07 (high byte) • rem_port_nr[2] = D0 (low byte)
6 to 7	reserved	WORD	B#16#00 ...	Not used. Leave the value "0" at this parameter.

Creating TADDR_Param in a data block

You have the following options for creating TADDR_Param:

- Create a new data block and select TADDR_Param as type in the dialog "Add new data block".
- Open an existing data block create a new tag and enter in the column Data type TADDR_Param.

A data block can contain several system data types TADDR_Param.

TUSEND: Send data via Ethernet (UDP)

Description

The "TUSEND" instruction sends data to the remote partner addressed by the ADDR parameter using UDP.

WARNING

Data transfer via UDP

Data transferred via UDP to RFC 768 to the remote partner are sent without acknowledgement and are therefore unsecured. This means the data can be lost without any indication at the block.

Note

For sequential send operations to different partners, you only need to adjust the ADDR parameter when calling "TUSEND". You do not need to call the "TCON (Page 5083)" and "TDISCON (Page 5091)" instructions again. The UDP port must be set up at the specific remote partner in order for this partner to receive data.

Functional description

"The TUSEND" instruction works asynchronously, which means its job processing extends over multiple calls. Create a rising edge at the REQ parameter to establish a connection once again.

The output parameters BUSY, DONE, ERROR and STATUS indicate the status of the job.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 3150).

The following table shows the relationship between BUSY, DONE and ERROR. Using this table, you can determine the current status of "TUSEND" or when the send process is concluded.

BUSY	DONE	ERROR	Description
TRUE	FALSE	FALSE	The job is being processed.
FALSE	TRUE	FALSE	Job successfully completed.
FALSE	FALSE	TRUE	The job ended with an error. The cause of the error can be found in the STATUS parameter.
FALSE	FALSE	FALSE	The instruction was not assigned a (new) job.

Note

Due to the asynchronous operation of "TUSEND", make sure the data in the send area remains consistent until the DONE parameter or the ERROR parameter has the value TRUE.

Parameters

The following table shows the parameters of the "TUSEND" instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Control parameter REQUEST starts the send job on a rising edge. The data is transferred from the area specified by DATA and LEN.
ID	Input	CONN_OUC	I, Q, M, D, L or constant	Reference to the associated connection between the user program and the communication layer of the operating system. ID must be identical to the associated parameter ID in the local connection description with the instruction "TCON". Range of values: W#16#0001 to W#16#0FFF
LEN	Input	UINT	I, Q, M, D, L or constant	Number of bytes to be sent with the job Range of values: 1 to 1472
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: • 0: Job not yet started or still executing. • 1: Job executed without errors. This value is only displayed for one cycle.
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY = 1: The job is not yet completed. A new job cannot be triggered. • BUSY = 0: The job is completed.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: • ERROR=1: An error has occurred during processing, STATUS supplies detailed information on the type of error.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter STATUS: Error information
DATA	InOut	VARIANT	I, Q, M, D	Send area, contains address and length The address refers to: • The process image of the inputs • The process image of the outputs • A bit memory • A data block
ADDR	InOut	VARIANT	D	Pointer to the system data type TADDR_Param. Store the address information of the remote partner (IP address and port number) in a data block with the system data type TADDR_Param. See also: Structure of the address information for the remote partner with UDP (Page 5107)

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

ERROR and STATUS parameters

ERROR	STATUS* (W#16#...)	Explanation
0	0000	Send job completed without error
0	7000	No job processing active
0	7001	Start of job processing, data being sent Note: During this processing phase, the operating system accesses the data in the DATA send area.
0	7002	Intermediate call (REQ irrelevant), job is being processed Note: During this processing phase, the operating system accesses the data in the DATA send area.
1	8085	The LEN parameter has the value "0" or is greater than the highest permitted value.
1	8086	The ID parameter is not in the permitted value range.
0	8088	The LEN parameter is greater than the memory area specified in DATA.
1	8089	The ADDR parameter does not point to a data block with the TADDR_Param structure.
1	80A1	Communication error: • The specified connection between user program and communication layer of the operating system has not yet been established. • The specified connection between the user program and the communication layer of the operating system is currently being terminated. Transmission over this connection is not possible. • The interface is being reinitialized.
1	80A4	IP address (at the ADDR parameter) of the remote connection end point is invalid; it may correspond to the local partner's own IP address.
1	80B3	• The protocol variant (connection_type parameter in the connection description) is not set to UDP. Please use "TSEND (Page 5096)". • ADDR parameter: Invalid information for port no.
1	80B7	The length of the structure referenced by the parameter ADDR is not 8 bytes.
1	80C3	• A block with this ID is already being processed in a different priority class. • Internal lack of resources.
1	80C4	Temporary communication error: • The connection between the user program and the communication layer of the operating system cannot be established at this time. • New parameter settings are being assigned to the interface.
1	80C5	The remote partner has terminated the connection.
1	80C6	The remote partner cannot be reached (network error).
1	80C7	Execution timeout.
-	General error informatio n	See also: GET_ERR_ID: Get error ID locally (Page 3392)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

TURCV: Receive data via Ethernet (UDP)

Description

The "TURCV" instruction receives data via UDP. After successful completion of "TURCV", the ADDR parameter will show you the address of the remote partner (the sender).

WARNING

Unsecured data transfer

Data transferred via UDP to RFC 768 to the remote partner are sent without acknowledgement and are therefore unsecured. This means the data can be lost without any indication at the block.

Functional description

"The TURCV" instruction works asynchronously, which means its job processing extends over multiple calls. You start the receive job by calling "TURCV" with EN_R = 1.

The output parameters BUSY, DONE, ERROR and STATUS indicate the status of the job.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 3150).

The following table shows the relationship between BUSY, NDR and ERROR. Using this table, you can determine the current status of TURCV or when the receive process is concluded.

BUSY	NDR	ERROR	Description
TRUE	FALSE	FALSE	The job is being processed.
FALSE	TRUE	FALSE	Job successfully completed. New data was received.
FALSE	FALSE	TRUE	The job ended with an error. The cause of the error can be found in the STATUS parameter.
FALSE	FALSE	FALSE	The instruction was not assigned a (new) job.

Note

Due to the asynchronous operation of "TURCV", the data in the receive area is only consistent when the NDR parameter has the value TRUE.

Parameters

The following table shows the parameters of the "TURCV" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L or constant	Control parameter enabled to receive When EN_R = 1, "TURCV" is ready to receive. The receive job is being processed.
ID	Input	WORD	I, Q, M, D, L or constant	Reference to the associated connection between the user program and the communication layer of the operating system. ID must be identical to the corresponding parameter ID in the local connection description. Value range: W#16#0001 to W#16#0FFF
LEN	Input	UINT	I, Q, M, D, L or constant	Length of the receive area in bytes: 0 (recommended) or 1 to 1472
NDR	Output	BOOL	I, Q, M, D, L	Status parameter NDR: • NDR = 0: Job not yet started or still running • NDR = 1: Job successfully completed
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: • ERROR=1: Error occurred during processing. STATUS supplies detailed information on the type of error
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY = 1: The job is not yet completed. A new job cannot be triggered. • BUSY = 0: The job is completed.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter STATUS: Error information
RCVD_LEN	Output	UINT	I, Q, M, D, L	Amount of data actually received in bytes
DATA	InOut	VARIANT	I, Q, M, D, L	Receive area The address references: • The process image of the inputs • The process image of the outputs • A bit memory • A data block
ADDR	InOut	VARIANT	D	Pointer to the system data type TADDR_Param. The address information of the remote partner (IP address and port number) is written in a data block with the system data type TADDR_Param. See also: Structure of the address information for the remote partner with UDP (Page 5107)

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

ERROR and STATUS parameters

ERROR	STATUS* (W#16#...)	Explanation
0	0000	New data was accepted. The current length of the received data is shown in RCVD_LEN.
0	7000	Block not ready to receive
0	7001	Block is ready to receive, receive job was activated
0	7002	Intermediate call, receive job being processed Note: During this processing phase, "TURCV" writes data to the receive area. For this reason, an error could result in inconsistent data in the receive area.
1	8085	The LEN parameter is greater than the largest permitted value, or you changed the value of the LEN or DATA parameter since the first call
1	8086	The ID parameter is not in the permitted value range
1	8088	• Receive area is too small • Value in LEN is higher than the receive area specified by DATA
1	8089	The ADDR parameter does not point to a data block with the TADDR_Param structure.
1	80A1	Communication error: • The specified connection between user program and communication layer of the operating system has not yet been established. • The specified connection between the user program and the communication layer of the operating system is currently being terminated. A receive job over this connection is not possible. • New parameter settings are being assigned to the interface.
1	80B3	The protocol variant (connection_type parameter in the connection description) is not set to UDP. Please use "TRCV (Page 5099)".
1	80B7	The length at ADDR parameter does not correspond to 8 bytes.
1	80C3	• A block with this ID is already being processed in a different priority class. • Internal lack of resources.
1	80C4	Temporary communication error: New parameter settings are being assigned to the interface.
1	80C5	The remote partner has terminated the connection.
1	80C7	Execution timeout.
1	80C9	With RFC1006 / UDP: The received data is longer than expected (size of receive buffer exceeded).
-	General error information	See also: GET_ERR_ID: Get error ID locally (Page 3392)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

See also

TCON: Establish communication connection (Page 5083)

TDISCON: Terminate communication connection (Page 5091)

T_RESET: Resetting the connection

Description

The "T_RESET" instruction terminates and then reestablishes an existing connection.

The local end points of the connection are retained. They are generated automatically:

- If a connection has been configured and loaded to the CPU.
- If a connection has been generated by the user program, for example, by calling the instruction "TCON (Page 5083)".

The instruction "T_RESET" can be executed for all connection types (TCP, UDP, ISO-on-TCP, etc.). It does not matter whether the local interface of the CPU or the interface of a CM/CP was used for the connection.

Once the instruction "T_RESET" has been called using the REQ parameter, the connection specified with the ID parameter is terminated and, if necessary, the data send and receive buffer cleared. Canceling the connection also cancels any data transfer in progress. There is therefore a risk of losing data if data transfer is in progress. The CPU defined as the active connection partner will then automatically attempt to restore the interrupted communication connection. You therefore do not need to call the "TCON (Page 5083)" instruction to reestablish the communication connection.

The output parameters DONE, BUSY and STATUS indicate the status of the job.

Parameters

The following table shows the parameters of the "T_RESET" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Control parameter REQUEST starts the job for terminating the connection specified by ID. The job starts on a rising edge.
ID	Input	CONN_OUC	I, Q, M, D, L or constant	Reference to the connection to the passive partner ID being interrupted must be identical to the corresponding parameter ID in the local connection description. Range of values: W#16#0001 to W#16#0FFF
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE • 0: Job not yet started or still executing. • 1: Job executed without errors
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter BUSY • 0: Job is complete. • 1: The job is not yet completed.

Parameter	Declaration	Data type	Memory area	Description
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR • 0: No error occurred. • 1: Error occurred during processing. The STATUS parameter supplies detailed information on the type of error
STATUS	Output	WORD	I, Q, M, D, L	Status parameter STATUS Error information (see "STATUS parameter" table).

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

Parameter STATUS

STATUS* (W#16#...)	Explanation
0000	No error.
0001	Connection has not been established.
7001	Connection termination launched.
7002	Connection being terminated.
8081	Unknown connection specified at the ID parameter.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

T_DIAG: Checking the connection

Description

You use the "T_DIAG" instruction to check the status of a connection and read further information on the local end point of this connection.

- The connection is referenced by the ID parameter. You can read both connection end points configured in the connection editor and programmed connection end points (e.g. with the "TCON" instruction).
  Temporary connection end points (for example end points created when you connect to an engineering station) cannot be diagnosed, as no connection ID is generated in this process.
- The connection information read is stored in a structure referenced by the RESULT parameter.
- The output parameter STATUS indicates whether it was possible to read the connection information. The connection information in the structure at the RESULT parameter is only valid if the "T_DIAG" instruction has been completed with STATUS = W#16#0000 and ERROR = FALSE.
  Connection information cannot be evaluated if an error occurs.

Possible connection information

Two different structures can be used to read the connection information at the RESULT parameter:

- The "TDiag_Status" structure only contains the most important information about a connection end point, for example the protocol used, the connection status and the number of data bytes sent or received.
- The structure "TDiag_StatusExt" supplies not just the most important information, but also the number of connection attempts, the reason for a connection abort, etc.

The structure and parameters of the two structures are described below (see the "TDIAG_Status and TDIAG_StatusExt structures" table).

Parameters

The following table shows the parameters of the "T_DIAG" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Starts the instruction to check the connection specified in the ID parameter when there is a positive edge.
ID	Input	CONN_OUC (WORD)	L, D or constant	Reference to the assigned connection. Range of values: W#16#0001 to W#16#0FFF
RESULT	InOut	VARIANT	D	Pointer to the structure in which the connection information is stored. The structure TDiag_Status or TDiag_StatusExt can be used at the RESULT parameter (for a description, see the "TDIAG_Status and TDIAG_StatusExt structures" table).

Parameter	Declaration	Data type	Memory area	Description
DONE	Output	BOOL	I, Q, M, D, L	Status parameter: • 0: Instruction not yet started or still in progress. • 1: Instruction executed without errors.
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter: • 0: Instruction not yet started or already completed. • 1: Instruction not yet completed. A new job cannot be started.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter: • 0: No error. • 1: Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

Parameters BUSY, DONE and ERROR

You can check the status of "T_DIAG" instruction execution with the BUSY, DONE, ERROR and STATUS parameters. The BUSY parameter indicates the processing status. You use the DONE parameter to check whether or not an instruction has been executed successfully. The ERROR parameter is set if errors occur during execution of "T_DIAG".

The following table shows the relationship between the BUSY, DONE and ERROR parameters:

BUSY	DONE	ERROR	Description
1	-	-	The instruction is being processed.
0	1	0	The instruction has been executed successfully. The data in the structure referenced by RESULT are only valid if this is the case.
0	0	1	Instruction completed with an error. The cause of the error is output at the STATUS parameter.
0	0	0	No new instruction has been assigned.

Parameter STATUS

The following table shows the meaning of the values at the STATUS parameter:

STATUS* (W#16#...)	Explanation
0000	The instruction "T_DIAG" has been executed successfully. The data in the structure referenced at the RESULT parameter can be evaluated.
7000	No instruction processing active.
7001	Instruction processing launched.
7002	Connection information is being read (REQ parameter irrelevant).

STATUS* (W#16#...)	Explanation
8086	The value at the ID parameter is outside the valid range (W#16#0001 ... W#16#0FFF).
8089	The RESULT parameter points to an invalid data type (structures TDIAG_Status and TDIAG_StatusExt only).
80A3	The ID parameter references a connection end point which does not exist. With programmed connections, this error can also occur after the "TDISCON" instruction is called.
80C4	Internal error. Access to the connection end point is temporarily unavailable.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

Structures TDIAG_Status and TDIAG_StatusExt

The table below details the form of the TDIAG_Status and TDIAG_StatusExt structures:

- From the InterfaceID to the ReceivedBytes parameter, the "TDIAG_StatusExt" and "TDIAG_Status" structures are identical.
- The structure TDIAG_StatusExt also includes the parameters ConnTrials to LastDisconnTimeStamp.

The value of each element is only valid if the instruction has been executed without errors. If an error occurs, the content of the parameters will not change.

Name	Data type	Description
The following parameters are in both the TDIAG_Status and the TDIAG_StatusExt structure:		
InterfaceID	HW_ANY	Interface ID (LADDR) of the CPU or the CM/CP.
ID	CONN_OUC	ID of the connection diagnosed. Following a successful call, the value of this element is identical to the parameter ID of the "T_DIAG" instruction.

Name	Data type	Description
ConnectionType	BYTE	Protocol type used for connection: • 0x01: Not used. • ... • 0x0B: TCP protocol (IP_v4) • 0x0C: ISO-on-TCP protocol (RFC1006) • 0x0D: TCP protocol (DNS) • 0x0E: Dial-in protocol • 0x0F: WDC protocol • 0x10: SMTP protocol • 0x11: TCP protocol • 0x12: TCP and ISO-on-TCP protocol (RFC1006) • 0x13: UDP protocol • 0x14: Reserved • 0x15: PROFIBUS bus access protocol (FDL) • 0x16: ISO 8073 transport protocol (ISO native) • ... • 0x20: SMTP or SMTPS protocol - based on IPv4 • 0x21: SMTP or SMTPS protocol - based on IPv6 • 0x22: SMTP or SMTPS protocol - based on FQDN (Fully Qualified Domain Name) • ... • 0x70: S7 connection • Other: Reserved
ActiveEstablished	BOOL	• FALSE: Locally, the passive connection end point • TRUE: Locally, the active connection end point
State	BYTE	Current status of the connection end point • 0x00: Not used. • 0x01: Connection terminated. Temporary status, for example, after the "T_RESET" instruction is called. The system then automatically attempts to reestablish the connection. • 0x02: The active connection end point is attempting to establish a connection to the remote communication partner. • 0x03: The passive connection end point is waiting for establishment of the connection to the remote communication partner. • 0x04: Connection established. • 0x05: The connection is being terminated. This may be because the "T_RESET" or "T_DISCON" instruction has been called. Other possible reasons are protocol errors and line breaks. • 0x06..0xFF: Not used.

Name	Data type	Description
Kind	BYTE	Mode of the connection end point: • 0x00: Not used. • 0x01: Configured, static connection which has been configured and loaded to the CPU. • 0x02: Configured, dynamic connection which has been configured and loaded to the CPU (not currently supported). • 0x03: Programmed connection generated in the user program with the instruction "TCON". A call of the instruction "TDISCON" or a transition to CPU STOP status has destroyed the connection end point. • 0x04: Temporary, dynamic connection established by the engineering station (ES) or operator station (OS), for example. (this connection type cannot currently by diagnosed as there is no ID). • 0x05..0xFF: Not used.
SentBytes	UDINT	Number of data bytes sent.
ReceivedBytes	UDINT	Number of data bytes received.
The following parameters only occur in the TDiag_StatusExt structure:		
ConnTrials	UDINT	Number of connection attempts. Once a connection has been established, ConnTrials has the value 0. If this element is not equal to 0, there are connection problems. Note: With a passive connection end point, this value is never greater than 1.
ConnTrialsSuccess	UDINT	Number of successful connection attempts. This element is never reset during the life cycle of a connection end point, and returns to 0 after reaching 0xFFFF FFFF. Note: This parameter is 1 if there has never been a problem in this connection.
LastConnErrReason	UDINT	Error ID output during the last connection attempt with errors (the error messages are identical to those at the LastDisconnReason parameter): • 0x4F01: Remote connection end point cannot be reached (this error usually occurs during connection establishment). • 0x4F02: Connection terminated locally. • 0x4F03: Connection terminated by remote communication partner. • 0x4F04: Connection terminated by a protocol error. • 0x4F05: Connection terminated by a network problem detected locally. • 0x4F06: Connection terminated by a network problem detected remotely. • 0x4F07: Connected terminated due to timeout in protocol. • 0x4F08: Incorrect parameter assignment: Connection to be established to local partner's own address. • 0x4F09: Connection temporarily reset by a call of the instruction "T_RESET". • 0x4F0A: Insufficient connection resources available (quantity exceeded) • 0x4F0B: Internal error: Incorrect addressing parameters • 0x4F0C: Internal CPU communication error • 0x4F0D: Internal AS communication error between CPU and CM/CP • 0x4F0E: The local TCP/UDP port (or RFC1006-T selector) specified is already in use.
LastConnErrTimeStamp	LDT	Time of the last connection attempt with errors.

Name	Data type	Description
LastDisconnReason	UDINT	Error ID which led to the last connection termination (the error messages are identical to those at the LastConnErrReason parameter): • 0x4F01: Remote connection end point cannot be reached (this error usually occurs during connection establishment). • 0x4F02: Connection terminated locally. • 0x4F03: Connection terminated by remote communication partner. • 0x4F04: Connection terminated by a protocol error. • 0x4F05: Connection terminated by a network problem detected locally. • 0x4F06: Connection terminated by a network problem detected remotely. • 0x4F07: Connected terminated due to timeout in protocol. • 0x4F08: Incorrect parameter assignment: Connection to be established to local partner's own address. • 0x4F09: Connection temporarily reset by a call of the instruction "T_RESET". • 0x4F0A: Insufficient connection resources available (quantity exceeded) • 0x4F0B: Internal error: Incorrect addressing parameters • 0x4F0C: Internal CPU communication error • 0x4F0D: Internal AS communication error between CPU and CM/CP • 0x4F0E: The local TCP/UDP port (or RFC1006-T selector) specified is already in use.
LastDisconnTimeStamp	LDT	Time of the last connection termination.

T_CONFIG: Configure interface

Description T_CONFIG

Description

The "T_CONFIG" instruction is used for the program-controlled configuration of the integrated PROFINET interfaces of the CPU or the interface of a CP/CM.

You can change the Ethernet address of the PROFINET device name from the user program via the instruction. The existing configuration data is overwritten.

You can make the following changes via the instruction "T_CONFIG":

- Settings for the IP protocol
  - IP address
  - Subnet mask
  - Router address
- Settings for PROFINET
  - Assignment of the PROFINET device name

The settings correspond with the configuration options under "IP protocol" and "PROFINET" in the dialog "Ethernet addresses". This is displayed in the view "devices & networks" under the properties of the PROFINET interface.

WARNING

Restart of CPU after execution of the "T_CONFIG" instruction

The CPU is restarted after you have executed the instruction to change an IP parameter. The CPU goes to STOP mode, a warm restart is carried out and the CPU starts up again (RUN mode).

Make sure that the control process is in a secure operating mode after the CPU has been restarted following execution of the instruction. Uncontrolled operation can result in serious material damage or personal injury due to malfunctions or programming errors, for example. Non-retentive data could be lost.

Requirement

In order to use the instruction must be explicitly specified in the hardware configuration that the assignment of IP address parameters and device name must be made by the user program.

- For this purpose open the properties of the PROFINET interface in the device view. Activate the following options in the dialog "Ethernet addresses":
  - In order to change the IP address parameters with "T_CONFIG": Select the setting "Set IP address using a different method" under "IP protocol".
  - In order to change the PROFINET device name with "T_CONFIG": Select the setting "Set PROFINET device name using a different method" under "PROFINET".
- The configuration data must be stored in the following system data types and assigned at the parameter CONF_DATA (Page 5126):
  - Store the IP address, subnet mask, and the router address in the system data type IF_CONF_V4.
  - Store the device name in the system type IF_CONF_NOS. Observe the restrictions that exist for assigning the name of the device (see parameter CONF_DATA (Page 5126)).

Functional description

The "T_CONFIG" instruction works asynchronously, which means its execution extends over multiple calls. The configuration process is started by calling "T_CONFIG" with REQ = 1. Only one job can be active at any time.

The block is edge triggered which means that after BUSY= FALSE the block must be re-called with REQ=FALSE to enable the instance.

Parameters

The following table shows the parameters of the "T_CONFIG" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Calling the instruction with REQ = 1 starts processing of the instruction.
INTERFACE	Input	HW_INTERFACE	I, Q, M, D, L or constant	Hardware identifier of the interface The hardware ID is displayed in the properties of the interface in the device view, and in the system constants of the PLC tags.
CONF_DATA (Page 5126)	Input	VARIANT	D, L	Pointer to the parent structure that contains the system data types IF_CONF_HEADER, IF_CONF_V4 and IF_CONF_NOS (see description of the parameters CONF_DATA).
DONE (Page 5130)	Output	BOOL	I, Q, M, D, L	Status parameter: • 0: Processing not yet complete. • 1: Processing of instruction finished successfully.
BUSY (Page 5130)	Output	BOOL	I, Q, M, D, L	Status parameter: • 0: Processing of the instruction has not started, completed or canceled yet. • 1: Processing of instruction is in progress
ERROR (Page 5130)	Output	BOOL	I, Q, M, D, L	Status parameter: • 0: No error • 1: Error
STATUS (Page 5130)	Output	DWORD	I, Q, M, D, L	Detailed status information: Detailed error and status information in the form of an error code are output at the parameter STATUS.
ERR_LOC (Page 5130)	Output	DWORD	I, Q, M, D, L	Error location: • 0: Error during execution of the instruction or when assigning parameters. • > 0: Errors in the structure or content of the configuration data at the parameter CONF_DATA.

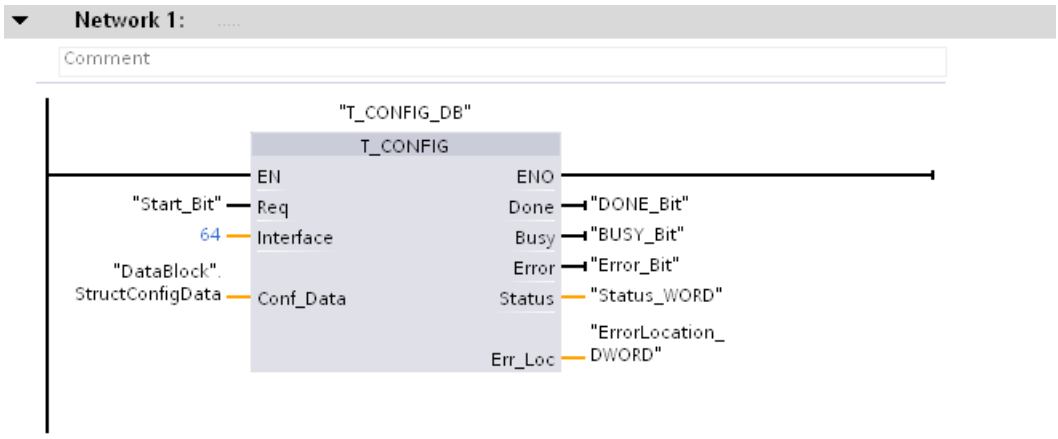
For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Example

In the following example the device name of an IO device is changed with the instruction "T_CONFIG".

Call of instruction

- The execution of the instruction is started with REQ=1.
- The hardware ID of the PROFINET interface is specified at the parameter Interface.
- The structure "StructConfigData" is referenced at the parameter CONF_DATA. The structure was created in the global data block "DataBlock".



Parameters CONF_DATA

The structure "StructConfigData" at the parameter CONF_DATA contains the following information:

- In the Header (IF_CONF_HEADER).
  SubfieldCount = 1: Indicates that only one additional structure (nos) is used below.
- In the structure "nos" (system data type IF_CONF_NOS)
  - Lenght = 11: Information on overall length of the structure NOS (5 bytes for the device name "myplc" + 6 bytes for the parameters Id, Length and Mode)
    Note: Instead of an absolute length, you can use the default start value (Lenght = 0) for a dynamic length.
  - Mode = 1: Permanent change of the device name in "myplc".
  - NOS[1] ... NOS[5]: Device name (1 character / byte)

DataBlock

	Name	Data type	Start value
1	Static		
2	StructConfigData	Struct	
3	Header	IF_CONF_Header	
4	FieldType	UInt	0
5	FieldId	UInt	0
6	SubfieldCount	UInt	1
7	nos	IF_CONF_NOS	
8	Id	UInt	40
9	Length	UInt	11
10	Mode	UInt	1
11	NOS	array [1..240] of Byte	
12	NOS[1]	Byte	'm'
13	NOS[2]	Byte	'y'
14	NOS[3]	Byte	'p'
15	NOS[4]	Byte	'l'
16	NOS[5]	Byte	'c'
17	NOS[6]	Byte	16#0
18	NOS[7]	Byte	16#0
19	NOS[8]	Byte	16#0
20	NOS[9]	Byte	16#0
21	NOS[10]	Byte	16#0
22	NOS[11]	Byte	16#0

See also

Evaluating errors with output parameter RET_VAL (Page 3152)

Parameter CONF_DATA

Structure of the configuration data

The configuration data at the parameter CONF_DATA can be stored in a global data block or in the section "Static" of the block interface.

The configuration data must be stored according to the following structure:

Name	Data type	Description
ConfData	Struct	Parent structure that is assigned at the parameter CONF_DATA.
Header	IF_CONF_HEADER	The header is used to define the number of these system data types. The system data type IF_CONF_HEADER must always be included.
IPData	IF_CONF_V4	The IP address, subnet mask and router address are stored in this system data type. Only create IF_CONF_V4 if you also want to change the Ethernet address with "T_CONFIG".
NoS	IF_CONF_NOS	The PROFINET device names are stored in this system data type. Only create IF_CONF_NOS if you also want to change the device names with "T_CONFIG".

The system data types IF_CONF_HEADER, IF_CONF_V4 and IF_CONF_NOS are created by entering the name of the system data type in the column "data type" of the data block or the block interface. The name for the system data types can be freely assigned.

System data type IF_CONF_Header

Use the system data type IF_CONF_Header to specify the number of system data types IF_CONF_V4 and IF_CONF_NOS that are used during the execution of "T_CONFIG".

Byte	Parameter	Data type	Start value	Description
0 ... 1	FieldType	UINT	0	Field type: Must always have the value "0".
2 ... 3	FieldId	UINT	0	Error ID: Must always have the value "0".
4 ... 5	SubfieldCount	UINT	0	Number of system data types IF_CONF_V4 and IF_CONF_NOS: • 1: Only one of the system data types is used. • 2: Both system data types are used

System data type IF_CONF_V4

The IP address, subnet mask and router address are defined with the system data type IF_CONF_V4.

Byte	Parameter	Data type	Start value	Description
0 ... 1	Id	UINT	30	System data type ID The start value of this parameter may not be changed.
2 ... 3	Length	UINT	18	Length of the system data type IF_CONF_V4 The start value must be used for the length specification since the parameters of IF_CONF_V4 have a fixed length and structure.
4 ... 5	Mode	UINT	0	Validity of addressing: • 1: Permanent validity of the configuration data • 2: Temporary validity of the configuration data including the deletion of existing permanent configuration data
6 ... 9	InterfaceAddress	IP_V4 *	0.0.0.0	IP address
10 ... 12	SubnetMask	IP_V4 *	0.0.0.0	Subnet mask
14 ... 16	DefaultRouter	IP_V4 *	0.0.0.0	Router address
* The data type IP_V4 is a structure of 4 BYTE, which includes the respective address of the respective parameter (e.g. at parameter SubnetMask the four-digit address of the subnet mask of the IP protocol).				

Subfield IF_CONF_NOS

Use the subfield IF_CONF_NOS to specify the station name to be assigned during execution of the instruction "T_CONFIG".

Byte	Parameter	Data type	Start value	Description
0 ... 1	Id	UINT	40	System data type ID The start value of this parameter may not be changed.
2 ... 3	Length	UINT	246	Length of the system data type IF_CONF_NOS in bytes. • For an absolute length, the value for the Lenght parameter consists of: – 6 bytes for the parameter Id, Length and Mode. – up to 240 bytes for the device name (parameter NOS). Example: An overall length of 10 is the result for the device name "plc1" with a length of 4 characters (=4 bytes). • Use the default start value 246 at the Lenght parameter for a dynamic length. Make sure that you enter the value "0" after the name (see description for NOS parameter).

Byte	Parameter	Data type	Start value	Description
4 ... 5	Mode	UINT	0	Validity of the device name change: • 1: Permanent validity of the device name. • 2: Temporary validity of the device name.
6 ... 244	NOS	ARRAY [1...240] of Byte	0	Device name (Name of Station) • You must occupy the ARRAY from the first byte. The station name is deleted if "0" is assigned as the first byte. • The minimum length for the name is 2 bytes. The maximum length for a name is 240 bytes. • If the device name is shorter than specified at the parameter Length then a zero byte (16#0 hex) must be entered after the actual station name (according to IEC 61185-6-10). Otherwise NOS will be rejected and the instruction "T_CONFIG" outputs the error codeDW#16#C0809400 at the parameter STATUS. • If the device name is longer than specified at the parameter Length then the device name will only be written up to to specified length. The following restrictions apply for the device name: • The name must be specified in ASCII code. • Only lowercase letters, numbers, hyphen or dots may be used for the name. – The name may not begin or end with a hyphen. – The name may not begin with numbers. – The name may not have the form n.n.n.n (n = 0, ... 999). – The name may not begin with the string "port-xyz" or "port-xyz-abcde" (a, b, c, d, e, x, y, z = 0, ... 9). • A name component between two points may be up to 63 characters long. • No special characters such as umlauts, brackets, underscore, slash, blank etc. The error code C080_9400 is output at the parameter STATUS if an invalid character is used.

DONE, BUSY and ERROR parameters

Description

The following table shows the relationship between BUSY, DONE and ERROR. Using this table, you can determine the current status of the instruction and when the transfer of configuration data is concluded.

BUSY	DONE	ERROR	Description
TRUE	FALSE	FALSE	The job is being processed.
FALSE	TRUE	FALSE	Job successfully completed.
FALSE	FALSE	TRUE	The job ended with an error. The cause of the error can be found in the parameter STATUS (Page 5130).
FALSE	FALSE	FALSE	The instruction was not assigned a (new) job.

Parameter STATUS and ERR_LOC

Description

The status and error messages of the instruction "T_CONFIG" are output at the parameters STATUS and ERR_LOC.

- The cause of the error is output at the parameter STATUS.
- The location of the error that occurred is output at the parameter ERR_LOC. The following options are available here:
  - 16#0000_0000: Error when calling the instruction (e.g. errors when assigning parameters to the instruction or in communication with the PROFINET interface).
  - 16#0001_0000: Error with the configuration data in the parameters of the system data type IF_CONF_HEADER.
  - 16#0001_0001: Error with the configuration data in the parameters of the system types IF_CONF_V4 or IF_CONF_NOS.

The following table shows the possible values for the parameters STATUS and ERR_LOC:

STATUS*	ERR_LOC*	Explanation
0000_0000	0000_0000	Order processing completed without errors.
0070_0000	0000_0000	No job processing active.
0070_0100	0000_0000	Start of the order processing.
0070_0200	0000_0000	Intermediate call (REQ irrelevant).
C08x_yy00	0000_0000	General error information. See also: GET_ERR_ID: Get error ID locally (Page 3392)
C080_8000	0000_0000	Error at call of the instruction: The hardware ID at the parameter Interface is invalid.
C080_8100	0000_0000	Error at call of the instruction: The hardware ID at the parameter Interface does not address a PROFINET interface.

STATUS*	ERR_LOC*	Explanation
C080_8700	0000_0000	Error at call of the instruction: Incorrect length of the data block at the parameter CONF_DATA.
C080_8800	0001_0000	Error in the system data type IF_CONF_HEADER: The parameter FieldType has an invalid value. Use the value "0" for FieldType.
C080_8900	0001_0000	Error in the system data type IF_CONF_HEADER: The parameter FieldId has an invalid value or was used several times. Use the value "0" for FieldId.
C080_8A00	0001_0000	Error in the system data type IF_CONF_HEADER: Incorrect number at the parameter SubfieldCount. Enter the correct number of system data types IF_CONF_V4 and IF_CONF_NOS in use.
C080_8B00	0001_0001	Error in the system data type IF_CONF_V4 or IF_CONF_NOS: The parameter Id has an invalid value. Use "30" and IF_CONF_NOS "40" for IF_CONF_V4.
C080_8C00	0001_0001	Error in the system data type IF_CONF_V4 or IF_CONF_NOS: Incorrect data type system used, wrong order or multiple use of a system data type.
C080_8D00	0001_0001	Error in the system data type IF_CONF_V4 or IF_CONF_NOS: The parameter Length has an incorrect or invalid value.
C080_8E00	0001_0001	Error in the system data type IF_CONF_V4 or IF_CONF_NOS: The parameter Mode has an incorrect or invalid value. Only the values "1" (permanent) or "2" (temporary) are valid.
C080_9000	0001_0001	Error in the system data type IF_CONF_V4 or IF_CONF_NOS: Configuration data cannot be applied. Possible cause: The setting "Set IP address on device" or "Set PROFINET device name on device" was not selected in the hardware configuration.
C080_9400	0001_0001	Error in the system data type IF_CONF_V4 or IF_CONF_NOS: A parameter value is undefined or invalid.
C080_9500	0001_0001	Error in the system data type IF_CONF_V4 or IF_CONF_NOS: The values of two parameters are inconsistent.
C080_C200	0000_0000	Error at call of the instruction: The configuration data can not be transferred. Possible cause: The PROFINET interface is not accessible.
C080_C300	0000_0000	Error at call of the instruction: Insufficient resources (e.g., multiple calling of "T_CONFIG" with different parameters).
C080_C400	0000_0000	Error at call of the instruction: Temporary communication error. Time indication for change to daylight saving time.
C080_D200	0000_0000	Error at call of the instruction: Call not possible. Instruction is not supported by the selected PROFINET interface.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

See also

Switching display formats in the program status (Page 2980)

Web server

WWW: Synchronizing user-defined web pages

Description

The instruction WWW initializes the web server of the CPU or synchronizes user-defined web pages with the user program in the CPU.

User-defined web pages together with the web server make it possible for the CPU to access freely designed web pages of the CPU with a web browser.

Use script instructions (such as Javascript) and HTML code in user-defined web pages to transfer data via a web browser for further processing to the CPU and to display data from the operand area of the CPU in the web browser. Call the WWW instruction in the user program for synchronization of the user program and the web server as well as initialization.

Initialization

User-defined web pages are "packaged" in data blocks for processing by the CPU. You will have to generate appropriate data blocks from the source files (HTML files, screens, Javascript files, ...) during configuration. The Web Control DB has a special role (default: DB 333). It contains status and control information as well as links to additional data blocks with coded web pages. The data blocks with coded web pages are called fragment DBs.

When the data block is downloaded into the CPU, the CPU does not "know" that user-defined web pages are coded inside it. The instruction "WWW" in the startup OB, for example, will inform the CPU which DB is the Web Control DB. The user-defined web pages can be accessed via a web browser after this initialization.

Synchronization

If you want the user program to interact with the user-defined web pages, then the instruction WWW must be used in the cyclical program part.

Examples of interaction between user program and web page:

- Check received data
- Assemble and send back data to the web browser making the request

In this case it must be possible to evaluate the current status information and the web server must receive control information, such as release of a web page requested by a web browser.

Parameter

The following table shows the parameters of the instruction "WWW":

Parameter	Declaration	Data type	Memory area	Description
CTRL_DB	Input	DB_WWW	D	Data block that describes the user-defined web pages (Web Control DB)
RET_VAL	Output	INT	I, Q, M, D, L	Error information

For additional information on valid data types, refer to Overview of the valid data types (Page 2236).

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error occurred. There are no web page requests that have to be released by the user program.
00xy	x: indicates whether an error has occurred during initialization of the Web Control DB (CTRL_DB): x=0: No errors occurred. x=1: Error occurred. The error is coded in the byte "CTRL_DB.last_error" of the Web Control DB, see description of Web Control DB. y: Number of the pending request. Several requests are possible (e. g. requests "0" and "1" are pending: y="3". y="1": Request "0" y="2": Request "1" y="4": Request "2" y="8": Request "3"
803A	The specified Web Control DB does not exist on the CPU.
8081	Incorrect version or incorrect format of the Web Control DB
80C1	There are no resources to initialize the web application.

See also

Overview of the valid data types (Page 2236)

Communications processor

SIMATIC NET CM/CP

S7-1500 CM/CP

Industrial Ethernet

Instructions for FTP services

FTP_CMD for FTP services

Overview of FTP_CMD

Meaning

Using the FTP_CMD instruction, you can establish FTP connections and transfer files from and to an FTP server.

Data transfer is possible using FTP or FTPS (secure SSL connections).

Note

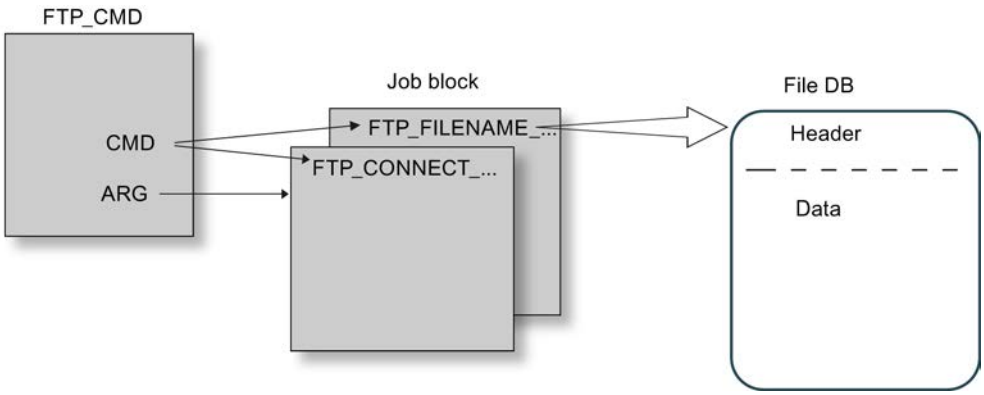
FTPS: Comparing certificates

FTPS requires a comparison of the certificates between FTP server and FTP client. If the FTP server is configured outside the STEP 7 project of the FTP client, the certificate needs to be imported from the FTP server. Import the certificate of the FTP server as a trusted certificate in the certificate manager.

How it works

The FTP_CMD instruction references a job block (ARG) in which the FTP command is specified. Depending on the type of FTP command (CMD), this job block uses different data structures for parameter assignment. Suitable data types (UDTs) are available for these various structures.

The following diagram shows the call structure:



Job blocks

The following data structures are used for the job blocks:

- Connection establishment
  Various data structures are available for the connection establishment using the following types of access:
  - UDT FTP_CONNECT_IPV4: Connection establishment with IP addresses according to IPv4
  - UDT FTP_CONNECT_IPV6: Connection establishment with IP addresses according to IPv6
  - UDT FTP_CONNECT_NAME: Connection establishment with server name (DNS)
- Data transfer
  For the data transfer, two different data structures are available:
  - UDT FTP_FILENAME: Data structure for access to a complete file
  - UDT FTP_FILENAME_PART: Data structure for read access to a data area

Data transfer in the File_DB

The data transfer is achieved using data blocks containing a header for job data and the area for the user data. The data block is specified in the job buffer.

Requirements in the CPU configuration

Use the following settings to allow FTP access:

- In the configuration data of the CPU in "Properties > General > Protection": Disable the "Disable PUT/GET communication" option.
- For all data blocks being used as file DBs, disable the "Optimized block access" attribute

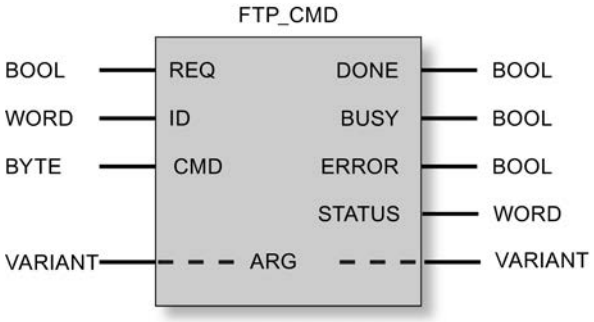
Validity

The FTP_CMD instruction can be used with the following module types:

- CP 1543-1

Call interface

Call interface in FBD representation



See also

Output parameters and status information FTP_CMD (Page 5143)
Input parameter - FTP_CMD (Page 5136)
Structure of the data blocks (file DBs) for FTP services - FTP client mode (Page 5147)

Input parameter - FTP_CMD

Explanation of the input parameters

You supply the FTP_CMD instruction with the following input parameters:

Table 9-97 Formal parameters of the FTP_CMD instruction - input parameters

Parameter	Declaration	Data type	Memory area	Meaning / remarks
REQ	Input	BOOL	I, Q, M, D, L	Starts the send job on a rising edge.
ID *	INPUT	INT	1, 2 ... 64	The FTP jobs are handled on FTP connections. The parameter identifies the connection being used.

Parameter	Declaration	Data type	Memory area	Meaning / remarks
CMD *	INPUT	BYTE	See following table "Commands".	FTP command to be executed when the instruction is called. You will find value ranges for the FTP command types after the table. Note: The FTP command specified here must be specified identically in the job block (ARG parameter). If a command is not supported by the CP firmware, an error message with STATUS = $8F6B_H$ is output.
ARG *	INPUT	VARIANT	See following table "Commands".	Job block References the data area with the execution parameters suitable for the FTP command. Specific data types (UDT) are used depending on the FTP command. The UDTs are shown below. The ANY data type is not permitted for the pointer to be specified here!

* The values of the input parameters "ID" and "CMD" overwrite the value of the input parameter "ARG".

FTP commands in the "CMD" parameter

The following table shows you the significance of the commands of the "CMD" parameter and which UDTs you use to supply the job blocks.

Table 9-98 Command types

CMD (command type)	Relevant job blocks / UDT	Meaning / handling
0 (NOOP)	*	The called FC does not execute any action. The status codes are set as follows when these parameters are supplied: DONE=1; ERROR=0; STATUS=0
1 (CONNECT)	FTP_CONNECT_IPV4 FTP_CONNECT_IPV6 FTP_CONNECT_NAME	FTP connection establishment With this command, the FTP client establishes an FTP connection to an FTP server (port 21). The connection is available under the connection ID specified here for all further FTP commands. Data is then exchanged with the FTP server specified for this user.
2 (STORE)	FTP_FILENAME	This function call transfers a data block (file DB) from the FTP client (S7-CPU) to the FTP server. Caution: If the file (file DB) already exists on the FTP server, it will be overwritten.
3 (RETRIEVE)	FTP_FILENAME	This function call transfers a file from the FTP server to the FTP client (S7-CPU). Caution: If the data block (file DB) on the FTP client already contains a file, it will be overwritten.
4 (DELETE)	FTP_FILENAME	With this function call, you delete a file on the FTP server.
5 (QUIT)	*	With this function call, you establish the FTP connections selected with the ID.

CMD (command type)	Relevant job blocks / UDT	Meaning / handling
6 (APPEND)	FTP_FILENAME	Similar to "STORE", the "APPEND" command saves a file on the FTP server. With "APPEND", the file on the FTP server is, however, not overwritten. The new content is appended to the existing file. If the file (file DB) does not exist on the FTP server, it will be created.
7 (RETR_PART)	FTP_FILENAME_PART	Using the "RETR_PART" command (retrieve part) , you can request a section of a file from the FTP server. If very large files are involved, this allows you to restrict the read to the part you currently require. To do this, you need to know the structure of the file. Enter the required part of the file using the two parameters "OFFSET" and "LEN" in FB40.

* With the command types 0 (NOOP) and 5 (QUIT) a freely selectable job block (UDT) must be specified. This is not evaluated.

See also

Overview of FTP_CMD (Page 5134)

Job blocks for FTP_CMD

Meaning

You supply the FTP_CMD instruction with a job block using the ARG parameter. The structure depends on the FTP command type. By using the default data types (UDT), the instruction recognizes the type of the job block. Below, you will find the relevant data types (UDTs) for the following job blocks:

- FTP connection establishment with IP address according to IPv4
- FTP connection establishment with IP address according to IPv6
- FTP connection establishment with server name
- Write and read access and other FTP commands
- FTP command RETR_PART

Job block for FTP connection establishment with IP address according to IPv4

For FTP connection establishment with IP address according to IPv4, the following data structure is used.

Table 9-99 UDT FTP_CONNECT_IPV4

Parameter	Type	Range of values	Meaning / remarks
LADDR	WORD		Module start address When you call an instruction, you transfer the module start address of the CP in the LADDR parameter. You will find the module start address of the CP in the configuration of the CP under: "Properties>Addresses>Inputs"
ID	INT	1, 2...64	The FTP jobs are handled on FTP connections. The parameter identifies the connection being used.
ConnectionType	BYTE	0	Connection type "FTP"
ActiveEstablished	BOOL	TRUE	
FTPcmd	BYTE	1	FTP command "CONNECT" FTP command that executes when the instruction is called. You will find the ranges of values in Table 9-98 Command types (Page 5137) Note: The FTP command specified here must be specified identically in the CMD input parameter.
CertIndex	BYTE	0 = FTP 1 = FTPS	Here, choose between the protocol types FTP or FTPS. Note on FTPS: If the FTP server is configured outside the STEP 7 project of the FTP client, the certificate needs to be imported from the FTP server.
UserName	STRING[32]	'user'	User name for the login on the FTP server
Password	STRING[32]	'password'	Password for the login on the FTP server
FTPserverIP_addr	IPv4	’142.11.25.135’	IP address of the FTP server

Job block for FTP connection establishment with IP address according to IPv6

For FTP connection establishment with IP address according to IPv6, the following data structure is used.

Table 9-100 UDT FTP_CONNECT_IPV6

Parameter	Type	Range of values	Meaning / remarks
LADDR	WORD		Module start address When you call an instruction, you transfer the module start address of the CP in the LADDR parameter. You will find the module start address of the CP in the configuration of the CP under: "Properties>Addresses>Inputs"
ID	INT	1, 2...64	The FTP jobs are handled on FTP connections. The parameter identifies the connection being used.
ConnectionType	BYTE	0	Connection type "FTP"
ActiveEstablished	BOOL	TRUE	
FTPcmd	BYTE	1	FTP command "CONNECT" FTP command that executes when the instruction is called. You will find the ranges of values in Table 9-98 Command types (Page 5137) Note: The FTP command specified here must be specified identically in the CMD input parameter.
CertIndex	BYTE	0 = FTP 1 = FTPS	Here, choose between the protocol types FTP or FTPS. Note on FTPS: If the FTP server is configured outside the STEP 7 project of the FTP client, the certificate needs to be imported from the FTP server.
UserName	STRING[32]	'user'	User name for the login on the FTP server
Password	STRING[32]	'password'	Password for the login on the FTP server
FTPserverIP_addr	IPv6 array [1..16] of Byte		IP address of the FTP server

Job block for FTP connection establishment with server name

For FTP connection establishment specifying the server name, the following data structure is used. The server name is assigned to an IP address using DNS.

Table 9-101 UDT FTP_CONNECT_NAME

Parameter	Type	Range of values	Meaning / remarks
LADDR	WORD		Module start address When you call an instruction, you transfer the module start address of the CP in the LADDR parameter. You will find the module start address of the CP in the configuration of the CP under: "Properties>Addresses>Inputs"
ID	INT	1, 2...64	The FTP jobs are handled on FTP connections. The parameter identifies the connection being used.
ConnectionType	BYTE	0	Connection type "FTP"
ActiveEstablished	BOOL	TRUE	
FTPcmd	BYTE	1	FTP command "CONNECT" FTP command that executes when the instruction is called. You will find the ranges of values in Table 9-98 Command types (Page 5137) Note: The FTP command specified here must be specified identically in the CMD input parameter.
CertIndex	BYTE	0 = FTP 1 = FTPS	Here, choose between the protocol types FTP or FTPS. Note on FTPS: If the FTP server is configured outside the STEP 7 project of the FTP client, the certificate needs to be imported from the FTP server.
UserName	STRING[32]	'user'	User name for the login on the FTP server
Password	STRING[32]	'password'	Password for the login on the FTP server
FTPserverName	STRING[254]		IP address of the FTP server

Job block for write and read access and other FTP commands

The following data structure is used for the FTP commands store, retrieve, delete and append.

Table 9-102 UDT FTP_FILENAME

Parameter	Type	Range of values	Meaning / remarks
LADDR	WORD		Module start address When you call an instruction, you transfer the module start address of the CP in the LADDR parameter. You will find the module start address of the CP in the configuration of the CP under: "Properties>Addresses>Inputs"
ID	INT	1, 2...64	The FTP jobs are handled on FTP connections. The parameter identifies the connection being used.

Parameter	Type	Range of values	Meaning / remarks
ConnectionType	BYTE	0	Connection type "FTP"
ActiveEstablished	BOOL	TRUE	
FTPcmd	BYTE	2, 3, 4, 6	FTP command "STORE / RETRIEVE / DELETE / APPEND" FTP command that executes when the instruction is called. You will find the ranges of values in Table 9-98 Command types (Page 5137) Note: The FTP command specified here must be specified identically in the CMD input parameter.
CertIndex	BYTE	0 = FTP 1 = FTPS	Here, choose between the protocol types FTP or FTPS. Note on FTPS: If the FTP server is configured outside the STEP 7 project of the FTP client, the certificate needs to be imported from the FTP server.
DataBlockNumber	UINT		The data block specified here contains the file DB to be read / written.
LenFilename	UINT	0...1000	The "LenFilename" parameter for specifying the total length of the file name is not evaluated. Instead the length information in the string of the "Filename" parameter is evaluated.
Filename	ARRAY[0..3] OF STRING[220]		File name of the source or destination file.

Job block for the RETR_PART FTP command

The following data structure is used for the RETR_PART FTP command.

Table 9-103 UDT FTP_FILENAME_PART

Parameter	Type	Range of values	Meaning / remarks
LADDR	WORD		Module start address When you call an instruction, you transfer the module start address of the CP in the LADDR parameter. You will find the module start address of the CP in the configuration of the CP under: "Properties>Addresses>Inputs"
ID	INT	1, 2...64	The FTP jobs are handled on FTP connections. The parameter identifies the connection being used.
ConnectionType	BYTE	0	Connection type "FTP"
ActiveEstablished	BOOL	TRUE	
FTPcmd	BYTE	7	FTP command "RETR_PART" FTP command that is executed when the instruction is called. You will find value ranges in Table 9-98 Command types (Page 5137) Note: The FTP command specified here must be specified identically in the CMD input parameter.

Parameter	Type	Range of values	Meaning / remarks
CertIndex	BYTE	0 = FTP 1 = FTPS	Here, choose between the protocol types FTP or FTPS. Note on FTPS: If the FTP server is configured outside the STEP 7 project of the FTP client, the certificate needs to be imported from the FTP server.
Offset	DWORD		Offset in bytes starting at which the file will be read.
Length	DWORD		Sublength in bytes that is read starting at the value specified in "OFFSET". Special features: • If "DW#16#FFFFFFFF" is specified, the available rest of the file will be read. Result OK (DONE = 1, STATUS = 0) if no other error occurred. • When OFFSET > length of the original file: Length of the destination file (ACT_LENGTH in file DB): 0 bytes on the CPU. Result OK (DONE = 1, STATUS = 0) if no other error occurred. • When OFFSET + LEN > length of the original file (and LEN ≠ 0xFFFFFFFF): Length of the destination file (ACT_LENGTH in file DB): Available bytes starting at "OFFSET". Result OK (DONE = 1, STATUS = 0) if no other error occurred.
DataBlockNumber	UINT		The data block specified here contains the file DB to be read / written.
LenFilename	UINT	0...1000	The "LenFilename" parameter for specifying the total length of the file name is not evaluated. Instead the length information in the string of the "Filename" parameter is evaluated.
Filename	ARRAY[0..3] OF STRING[220]		File name of the source or destination file.

See also

Overview of FTP_CMD (Page 5134)

Input parameter - FTP_CMD (Page 5136)

Output parameters and status information FTP_CMD

Parameters BUSY, DONE and ERROR

You control the execution status with the parameters BUSY, DONE, ERROR and STATUS. The BUSY parameter indicates the processing status. With the DONE parameter, you check whether or not a job was correctly executed. The ERROR parameter is set if errors occur during execution of "FTP_CMD". Error information is output in the STATUS parameter.

The following table shows the relationship between the parameters BUSY, DONE and ERROR:

BUSY	DONE	ERROR	Description
1	-	-	The job is being processed.
0	1	0	The job was completed successfully.
0	0	1	The job was terminated with an error. The cause of the error is specified in the STATUS parameter.
0	0	0	No new job was assigned.

Evaluating status codes

Note

For entries coded with 8FxxH in STATUS, refer to the information in the STEP 7 Standard and System Functions reference manual. The chapter describing error evaluation with the RET_VAL output parameter contains detailed information.

Table 9-104 FTP_CMD: Meaning of the STATUS parameter in conjunction with DONE and ERROR

DONE	ERROR	STATUS	Meaning
0	0	0000_H	No job is being executed.
1	0	0000_H	the job was completed without error.
0	0	7001_H	The job was triggered the first time (BUSY=1)
0	0	7002_H	The job is still being executed (BUSY=1)
0	0	$80C4_H$	Communication error (occurs temporarily, it is usually best to repeat the job in the user program).
0	0	8183_H	The configuration does not match the job parameters.
0	1	8401_H	Unknown error Possible causes: • A timeout was detected on the connection. • The FTP server has aborted the connection. Remedy: • Send the QUIT and CONNECT commands again to re-establish the connection.
0	1	8402_H	The connection has an error status. The timeout of the connection may have been exceeded or the FTP server may have terminated the connection. Remedy: Send the QUIT and CONNECT commands to re-establish the connection.
0	1	8403_H	Login has failed.
0	1	8404_H	FTP server is not obtainable.
0	1	8405_H	Transfer has failed.
0	1	8406_H	Timeout for current action
0	1	8407_H	File was not found on the FTP server.
0	1	8408_H	Transfer not possible.

DONE	ERROR	STATUS	Meaning
0	1	8409_H	File could not be fetched.
0	1	8410_H	Setting the TCP port for the data connection has failed.
0	1	8411_H	Offset information does not match.
0	1	8412_H	Error changing the specified directory
0	1	8413_H	Error receiving data
0	1	8414_H	Error sending data
0	1	8415_H	Specified CMD (command type) was rejected by the client.
0	1	8416_H	Connection was closed by the FTP server.
0	1	8418_H	Error in the user data. Possible causes: • File name is empty. • Data length is "0" • etc.
0	1	8419_H	There is no socket resource for opening a data connection.
0	1	8420_H	There is no socket resource for opening a control connection.
0	1	8421_H	Error opening the file DB to be read.
0	1	8422_H	Error opening the file DB to be written.
0	1	8423_H	Connection establishment to the FTP server has failed.
0	1	8424_H	Internal error
0	1	8425_H	Format error in the domain name
0	1	8426_H	There are too many DNS queries pending.
0	1	8427_H	The specified DNS server could not assign the specified domain name.
0	1	8428_H	There is no connection resource available.
0	1	8429_H	Unknown channel ID
0	1	8430_H	The file DB is too short.
0	1	8431_H	Error when writing to the file DB.
0	1	8432_H	Error when reading from the file DB.
0	1	8433_H	Error when accessing the file DB.
0	1	8434_H	Action was aborted.
0	1	8435_H	Channel will be reset.
0	1	8436_H	Unexpected server reply
0	1	8437_H	Certificate could not be verified.
0	1	8438_H	Unknown error occurred
0	1	8439_H	The FTP command causes an error. The cause must be looked for on the FTP server (REST command).
0	1	8440_H	The FTP server does not support the requested SSL protocol.
0	1	8446_H	After the FTP password was sent to the FTP server, an unexpected code was returned by the FTP server.
0	1	8451_H	An error was signaled when attempting to change the transmission mode from binary to ASCII.
0	1	8455_H	A memory request has failed on the CM/CP.
0	1	8460_H	A problem has occurred handling SSL/TLS.

DONE	ERROR	STATUS	Meaning
0	1	8469_H	Interface error The specified output interface could not be used. Remedy: Set the interface to be used for outgoing connections.
0	1	8475_H	The SSL certificate or the SSH md5 fingerprint was not considered trusted.
0	1	8476_H	Nothing was received from the FTP server. In the current status, an incorrect response must be assumed.
0	1	8477_H	The specified "Crypto engine" (cryptographic module) was not found.
0	1	8478_H	The attempt to set the selected SSL "Crypto engine" as the default failed.
0	1	8480_H	A problem has occurred with the certificate of the FTP client.
0	1	8481_H	The specified number could not be used.
0	1	8482_H	The FTP server uses a coding that is not supported.
0	1	8484_H	The maximum file size was exceeded.
0	1	8485_H	The file DB was modified while being processed to be sent or the file DB is incorrectly structured.
0	1	8489_H	Data could not be sent. There is not enough memory available for the action on the FTP server.
0	1	8492_H	The file already exists. The file will not be overwritten.
0	1	8496_H	A problem occurred reading the SSL CA certificate.
0	1	8497_H	An unexpected error occurred in the SSH session.
0	1	8498_H	It was not possible to terminate the SSL connection.
0	1	8499_H	The socket is not ready for sending/receiving. Wait until it is ready and try again.
0	1	8501_H	The SSL certificate check by the FTP server has failed.
0	1	8507_H	A timeout has occurred establishing the connection during the active FTP session while waiting for the FTP server.
0	1	$8F55_H$	Header status bit: Locked
0	1	$8F56_H$	The NEW bit in the file DB header was not reset
0	1	$8F6B_H$	Possible causes: • Bad value for the CMD parameter Values from 0 to 15 are permitted. • An FB40 command is not supported. Possible cause: Wrong firmware on the CP Remedy: Firmware update (with older CPs, use the functions FC 40...FC 44 instead of FB 40.)
0	1	$8F7F_H$	Internal error, for example illegal ANY reference

See also

Overview of FTP_CMD (Page 5134)

Structure and use of the file DB - FTP client mode

Structure of the data blocks (file DBs) for FTP services - FTP client mode

How it works

To transfer data with FTP, create data blocks (file DBs) on the CPU of your S7 station. These data blocks must have certain structure to allow them to be handled as transferable files by the FTP services. They consist of the following sections:

- Section 1: File DB header (has a fixed length of 20 bytes)
- Section 2: User data (has a variable length and structure)

Requirements in the CPU configuration

Use the following settings to allow FTP access:

- In the configuration data of the CPU in "Properties > General > Protection": Disable the "Disable PUT/GET communication" option.
- For all data blocks being used as file DBs, disable the "Optimized block access" attribute

File DB header for FTP client mode

Note: The file DB header described here is largely identical to the file DB header for server mode. The differences relate to the following parameters:

- WRITE_ACCESS
- FTP_REPLY_CODE

Parameter	Type	Value / meaning	Supply
EXIST	BOOL	The EXIST bit indicates whether the user data area contains valid data. The retrieve FTP command executes the job only when EXIST=1. • 0: The file DB does not contain valid user data ("file does not exist"). • 1: The file DB contains valid user data ("file exists").	The dele FTP command sets EXIST=0; The store FTP command sets EXIST=1;
LOCKED	BOOL	The LOCKED bit is used to restrict access to the file DB. • 0: The file DB can be accessed. • 1: The file DB is locked.	The stor and retr FTP commands set LOCKED=1 when they are executed. The following function is also possible when writing from the user program: The user program on the S7 CPU can set or reset LOCKED during write access to achieve data consistency. Recommended sequence in the user program: 1. Check LOCKED bit; if = 0 2. Set WRITEACCESS bit = 0 3. Check LOCKED bit; if = 0 4. Set LOCKED bit = 1 5. Write data 6. Set LOCKED bit = 0
NEW	BOOL	The NEW bit indicates whether data has been modified since the last read access. • 0: The content of the file DB is unchanged since the last write access. The user program of the S7 CPU has registered the last modification. • 1: The user program of the S7 CPU has not yet registered the last write access.	After execution, the stor FTP command sets NEW=1 After reading the data, the user program in the S7-CPU must set NEW=0 to allow a new retr command.

Parameter	Type	Value / meaning	Supply
WRITE_ACCESS	BOOL	0: The user program (FTP client blocks) has write access rights for the file DBs on the S7 CPU. 1: The user program (FTP client blocks) has no write access rights for the file DBs on the S7 CPU.	During the configuration of the DB, the bit is set to an initialization value. Recommendation: Whenever possible, the bit should remain unchanged! In special situations, adaptation during operation is possible.
ACT_LENGTH	DINT	Current length of the user data area. The content of this field is only valid when EXIST = 1.	The current length is updated following write access.
MAX_LENGTH	DINT	Maximum length of the user data area (length of the entire DB less 20 bytes header).	The maximum length should be specified during configuration of the DB. The value can also be modified by the user program during operation.
FTP_REPLY_CODE	INT	Unsigned integer (16-bit) containing the last reply code from FTP as a binary value. The content of this field is only valid when EXIST = 1.	This is updated by the FTP client when the FTP command is executed.
DATE_TIME	DATE_AND_TIME	Date and time of the last modification to the file. The content of this field is only valid when EXIST = 1.	The current date is updated following a write access. If the function for forwarding the time of day is used, the entry corresponds to the time that was passed on. If the function for forwarding the time of day is not used, a relative time is entered. This time relates to the startup of the CP (the initialization value is 1.1.1994 0.0 (midnight)).

See also

FILE_DB_HEADER data block as template - FTP client mode (Page 5149)

FILE_DB_HEADER data block as template - FTP client mode

Meaning

The data type FILE_DB_HEADER is predefined for creating a file DB header.

How it works

To transfer data with FTP, create data blocks (file DBs) on the CPU of your S7 station. These data blocks must have certain structure to allow them to be handled as transferable files by the FTP services. They consist of the following sections:

- Section 1: File DB header (has a fixed length of 20 bytes)
- Section 2: User data (has a variable length and structure)

Follow the steps outlined below:

1. For the PLC CPU on which you create the user program with the FTP instructions, create a data block of the type "Global DB".
2. Select the line you want to use as the start line for the file DB.
3. From the drop-down list in the "Data type" column, select a structure element of the type FILE_DB_HEADER.
   Result: A data structure with the header structure required for the file DB is created.
4. Select the properties of the newly created data block and disable the "Optimized block access" attribute.

Note

"Add new block" function - type selection

When you create new data blocks, the "FILE_DB_HEADER" block type is also available under the "Type" entry in the drop-down list. Do not use this option! The data block created in this way only contains the header structure and cannot be expanded by the area required for storing user data.

FILE_DB_HEADER data block - example and template for the file DB header

In the declaration view, you will see the following structure:

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	bit08	BOOL	FALSE	reserved
+0.1	bit09	BOOL	FALSE	reserved
+0.2	bit10	BOOL	FALSE	reserved
+0.3	bit11	BOOL	FALSE	reserved
+0.4	bit12	BOOL	FALSE	reserved
+0.5	bit13	BOOL	FALSE	reserved
+0.6	bit14	BOOL	FALSE	reserved
+0.7	bit15	BOOL	FALSE	reserved
+1.0	EXIST	BOOL	FALSE	if TRUE: File DB does not contain valid data

Address	Name	Type	Initial value	Comment
+1.1	LOCKED	BOOL	FALSE	if TRUE: The file DB is blocked due to a change to the content.
+1.2	NEW	BOOL	FALSE	if TRUE: The content of the File DB is new and must not be overwritten.
+1.3	WRITE_ACCESS	BOOL	FALSE	if TRUE: The FTP server has write access.
+1.4	bit04	BOOL	FALSE	reserved
+1.5	bit05	BOOL	FALSE	reserved
+1.6	bit06	BOOL	FALSE	reserved
+1.7	bit07	BOOL	FALSE	reserved
+2.0	ACT_LENGTH	DINT	L#0	Current length of the content in bytes (not including 20 bytes for the header)
+6.0	MAX_LENGTH	DINT	L#0	Current length of the content in bytes (not including 20 bytes for the header)
+10.0	FTP_REPLY_CODE	INT	0	Last replay information from the remote FTP server.
+12.0	DATE_TIME	DATE_AND_TIME	DT#00-1-1-0:0:0.000	Date and time of the last change to the content of the file DB.
=20.0		END_STRUCT		

Differences in the modes

File DB header for FTP client mode

The file DB header described here is largely identical for the FTP client mode and FTP server mode. The differences relate to the following parameters:

- WRITE_ACCESS
- FTP_REPLY_CODE

S7-1200 CM/CP

Telecontrol

Telecontrol instructions

TC_CON: Establish connection via the GSM network

Meaning

The TC_CON instruction allows an S7-1200 with a CP 1242-7 to establish a connection of the following types:

- ISO-ON-TCP
  Connection partner is a CP 1242-7.
  ISO-ON-TCP connections are used only in "GPRS direct" mode.
- UDP
  Any connection partner is possible.
- SMS
  The connection partner is an SMS client.
- Telecontrol connection
  The connection partner is either a telecontrol server or another station that can be reached via the telecontrol server.

A TC_CON establishes exactly one connection. Depending on the mode of the CP 1242-7 and the protocol you are using, a maximum of 3 to 5 simultaneous connections with unique IDs (see below) are supported per CP. You will find the maximum number of simultaneous connections in the performance data of the CP.

The CONNECT parameter uses a data block (DB) with the structure of a system data type (SDT) for the connection description.

The required connection type is defined using a connection-specific SDT "TCON_..." (see below). For each of the connection types listed above, one of the following SDTs must be assigned:

- TCON_IP_RFC for ISO-ON-TCP connections
- TCON_IP_V4 for UDP connections
- TCON_PHONE for SMS connections
- TCON_WDC for telecontrol connections

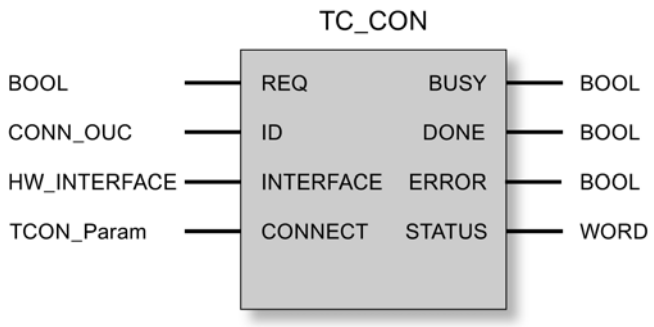
The "ActiveEstablished" parameter of these SDTs also specifies whether or not connection establishment is active or passive.

For parameter settings for these SDTs, see TCON_...: SDTs for the telecontrol connection establishment (Page 5167).

The ID parameter references the GPRS connection. The ID is assigned and must be unique within the CPU.

The INTERFACE parameter references the GPRS interface of the required local CP. This must be taken from STEP 7.

Call interface in FBD representation



Explanation of the formal parameters

The following table explains all the formal parameters for the TC_CON instruction.

Parameter	Declaration	Data type	Range of values	Description
REQ	INPUT	BOOL	0, 1	The instruction is started and the status codes initialized on a rising edge. Updating of the DONE, ERROR and STATUS status codes when there is a positive edge.
ID	INPUT	CONN_OUC (WORD)	1...07FF$_h$	Reference to the relevant connection. The ID is assigned. The value of ID is also required by the system data type (SDT) of the CONNECT parameter.
INTERFACE	INPUT	HW_INTERFACE		Reference to the interface of the local CP 1242-7 (see STEP 7 > CP configuration > Telecontrol interface > "Hardware identifier")
CONNECT	INOUT	TCON_Param	See also "TCON_...: SDTs for telecontrol connection establishment"	Reference to a data block for connection establishment. The SDTs of the type TCON_IP_RFC, TCON_IP_V4, TCON_PHONE or TCON_WDC specify the structure of the data block suitable for the relevant connection. In the SDTs, note the parameter "ActiveEstablished" (active / passive connection establishment).
ENO	OUTPUT	BOOL	0: Error 1: Error-free	Enable output If there is a runtime error with the instruction, ENO = 0 is set.

Parameter	Declaration	Data type	Range of values	Description
BUSY	OUTPUT	BOOL	0: Execution of the instruction not started, completed or aborted 1: The instruction is executing	Display of the processing status of the instruction
DONE	OUTPUT	BOOL	0: - 1: The instruction executed successfully	This parameter indicates whether or not the job was completed without errors. For the meaning in conjunction with the parameters ERROR and STATUS, refer to Codes of the instruction.
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning in conjunction with the parameters DONE and STATUS, refer to Codes of the instruction.
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the parameters DONE and ERROR, refer to Codes of the instruction.

The codes BUSY, DONE and ERROR

The codes of DONE and ERROR are relevant only when BUSY = 0.

BUSY	DONE	ERROR	Meaning
0	0	0	No job being executed

You will find all other code combinations of DONE and ERROR in the following table.

When called, the instruction remains in the BUSY = 1 state for several seconds. In the following situations, the BUSY state = 1 can last for a longer time:

- On active ISO-on-TCP connections if the partner cannot be reached.
- On passive connections when no frame is received.

The codes DONE, ERROR and STATUS

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

DONE	ERROR	STATUS	Meaning
1	0	0000_H	Job executed without errors
0	0	7000_H	No job processing active (first instruction call)
0	0	7001_H	Job processing started (first instruction call)
0	0	7002_H	Job processing already active (renewed instruction call when BUSY = 1)
0	1	8086_H	Illegal value for ID
0	1	8087_H	Maximum number of connections reached, no further connection possible
0	1	$80E3_H$	The ID is already being used by another connection.
0	1	$80E6_H$	No query in progress (instruction call not started)

DONE	ERROR	STATUS	Meaning
0	1	$80E8_H$	Remote partner cannot be reached. Check the connection parameters. In the "GPRS direct" mode, the message is output if the partner can be reached but is not accepting a connection request.
0	1	$80EB_H$	Request temporarily denied (TC_CON has already been called with the same destination address.)
0	1	$80EC_H$	Opening the Listener Port failed: Check the connection parameters.
0	1	$80F2_H$	The CP is in the wrong mode: • Telecontrol connections are permitted only in "Telecontrol" mode. • ISO-ON-TCP connections are permitted only in "GPRS direct" mode.
0	1	$80F3_H$	No free connection endpoint for sending data: • Use less connections or • Use less passive connections or • Turn off NTP. Remember the maximum number of simultaneous connections of the CP 1242-7.
0	1	$80F4_H$	Connection endpoint cannot be generated: Repeat the call. If necessary, check the connection parameters.
0	1	$80F6_H$	Format error of a parameter in the called data block (wrong length, wrong format or invalid value) Check the configuration of the "TC_CON..." SDT.

TC_DISCON: Terminate connection via the GSM network

Meaning

The TC_DISCON instruction on an S7-1200 with CP 1242-7 terminates an ISO-ON-TCP, UDP, SMS or telecontrol connection that was established with the TC_CON instruction.

You will find detailed information on the connection types in the description of the TC_CON instruction.

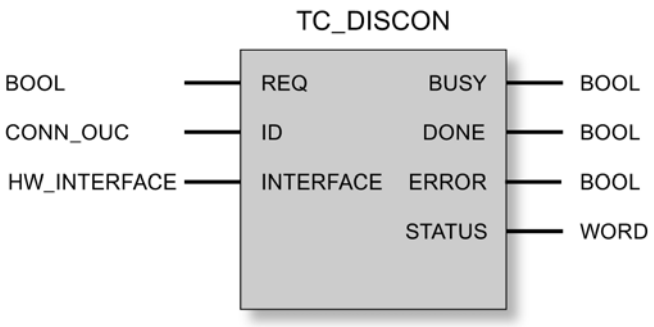
TC_DISCON terminates the connection to the telecontrol server only logically. If you want the connection to the telecontrol server to be terminated physically, configure the connection as a "Temporary connection" in the "telecontrol server" parameter group in STEP 7.

At the TCP/IP level, the connection is retained. Temporary stations terminate the connection automatically after sending the data.

The ID parameter references the GPRS connection. The ID must be unique within the CPU and the same as the ID used with TC_CON.

The INTERFACE parameter references the GPRS interface of the required local CP. The value must be the same as that used by TC_CON for INTERFACE.

Call interface in FBD representation



Explanation of the formal parameters

The following table explains all the formal parameters for the TC_DISCON instruction

Parameter	Declaration	Data type	Range of values	Description
REQ	INPUT	BOOL	0, 1	The instruction is started and the status codes initialized on a rising edge. Updating of the DONE, ERROR and STATUS status codes when there is a positive edge.
ID	INPUT	CONN_OUC (WORD)	1...07FF$_h$	Reference to the relevant connection
INTERFACE	INPUT	HW_INTERFACE		Reference to the interface of the local CP 1242-7 (see STEP 7 > CP configuration > Telecontrol interface > "Hardware identifier")
ENO	OUTPUT	BOOL	0: Error 1: Error-free	Enable output If there is a runtime error with the instruction, ENO = 0 is set.
BUSY	OUTPUT	BOOL	0: Execution of the instruction not started, completed or aborted 1: The instruction is executing	Display of the processing status of the instruction
DONE	OUTPUT	BOOL	0: - 1: The instruction executed successfully	This parameter indicates whether or not the job was completed without errors. For the meaning in conjunction with the parameters ERROR and STATUS, refer to Codes of the instruction.
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning in conjunction with the parameters DONE and STATUS, refer to Codes of the instruction.
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the parameters DONE and ERROR, refer to Codes of the instruction.

The codes BUSY, DONE and ERROR

The codes of DONE and ERROR are relevant only when BUSY = 0.

BUSY	DONE	ERROR	Meaning
0	0	0	The instruction has not yet been called.

You will find all other code combinations of DONE and ERROR in the following table.

Note

When called, the instruction remains in the BUSY = 1 state for several seconds.

The codes DONE, ERROR and STATUS

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

DONE	ERROR	STATUS	Meaning
1	0	0000_H	Job executed without errors
0	0	7000_H	No job processing active (first instruction call)
0	0	7001_H	Job processing started (first instruction call)
0	0	7002_H	Job processing already active (renewed instruction call when BUSY = 1)
0	1	8086_H	Illegal value for ID
0	1	$80E4_H$	Unknown ID: No connection with this ID has been established by TC_CON.
0	1	$80E6_H$	No query in progress (instruction call not started)
0	1	$80F5_H$	Invalid connection endpoint: • Connection establishment by TC_CON failed or • Connection terminated by remote partner.
0	1	$80F6_H$	Format error of a parameter in the called data block (wrong length, wrong format or invalid value) Check the configuration of the "TC_CON..." SDT.

TC_SEND: Send data via the GSM network

Meaning

The TC_SEND instruction allows the sending of data via programmed connections of the following types:

- ISO-ON-TCP connections
- UDP connections

- SMS connections
  The sending of SMS messages is supported only if this was set up in the STEP 7 configuration of the CP.
- Telecontrol connections

Note

Sending SMS messages to multiple recipients

If you want to send an identical SMS message to several recipients, you need to establish a connection to each recipient.

You will find more detailed information on the connection types in the description of the TC_CON instruction.

The ID parameter references the GPRS connection. The value of ID must correspond to the value used for ID by TC_CON.

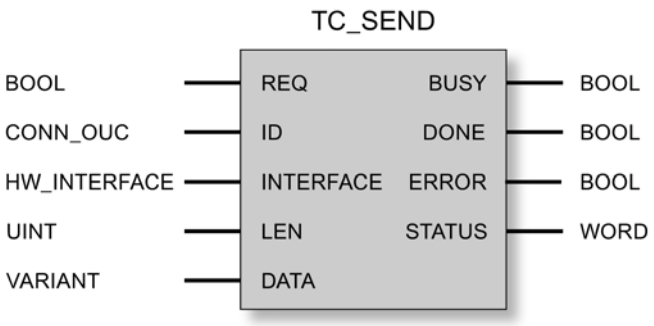
The INTERFACE parameter references the GPRS interface of the required local CP. The value must be the same as that used by TC_CON for INTERFACE.

The amount of data to be sent is specified with the LEN parameter.

The size of the data area specified in DATA must be at least as large as the number of bytes configured for LEN. Permitted data types in the data area specified in DATA are all except BOOL and ARRAY of BOOL.

The destination address (connection partner) for the data to be sent is configured in the TC_CON instruction.

Call interface in FBD representation



Explanation of the formal parameters

The following table explains all the formal parameters for the TC_SEND instruction.

Parameter	Declaration	Data type	Range of values	Description
REQ	INPUT	BOOL	0, 1	The instruction is started and the status codes initialized on a rising edge. Updating of the DONE, ERROR and STATUS status codes when there is a positive edge.
ID	INPUT	CONN_OUC (WORD)	1...07FF$_h$	Reference to the relevant connection
INTERFACE	INPUT	HW_INTERFACE		Reference to the interface of the local CP 1242-7 (see STEP 7 > CP configuration > Telecontrol interface > "Hardware identifier")
LEN	INPUT	UINT	1...2048	Number of bytes of data to be sent, maximum 2048. The value should match the size of the range of DATA.
DATA	INOUT	VARIANT		Address reference to the send data area of the CPU *
ENO	OUTPUT	BOOL	0: Error 1: Error-free	Enable output If there is a runtime error with the instruction, ENO = 0 is set.
BUSY	OUTPUT	BOOL	0: Execution of the instruction not started, completed or aborted 1: The instruction is executing	Display of the processing status of the instruction
DONE	OUTPUT	BOOL	0: - 1: The instruction executed successfully	This parameter indicates whether or not the job was completed without errors. ** For the meaning in conjunction with the parameters ERROR and STATUS, refer to Codes of the instruction.
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning in conjunction with the parameters DONE and STATUS, refer to Codes of the instruction.
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the parameters DONE and ERROR, refer to Codes of the instruction.

* For special features of the DATA parameter for SMS texts, refer to the next section.

** After sending a frame, TC_SEND sets DONE = 1. Note the following response:
The loss of an ISO-on-TCP connection is only recognized by the sender after 1 to 2 minutes. The transferred data may be lost although TC_SEND has set DONE = 1 at the sender.
If an ISO-on-TCP connection is aborted after receiving a frame before TC_RECV was started, the transferred data may be lost even if TC_SEND sets DONE = 1 at the sender.

Configuring SMS texts with the DATA parameter

The instruction sends the data referenced by the pointer of the type VARIANT of the DATA parameter as an SMS text.

If an operand of the data type STRING is referenced by DATA for SMS texts, the first two bytes are transferred with length information of the string.

One option for the correct text representation of SMS messages to be sent is to convert the text string into an Array of BYTE or Array of CHAR using the conversion function Strg_TO_Chars. Strg_TO_Chars at the EN parameter is linked to the output parameter ENO by TC_SEND.

For SMS texts, the CP does not support all special characters, for example umlauts (ü, ä etc.). The specification GSM 03.38 applies. There may be additional restrictions imposed by the GSM network provider.

The codes BUSY, DONE and ERROR

The codes of DONE and ERROR are relevant only when BUSY = 0.

BUSY	DONE	ERROR	Meaning
0	0	0	No job being executed

You will find all other code combinations of DONE and ERROR in the following table.

The codes DONE, ERROR and STATUS

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

DONE	ERROR	STATUS *	Meaning
1	0	0000_H	Job executed without errors
0	0	7000_H	No job processing active (first instruction call)
0	0	7001_H	Job processing started (first instruction call)
0	0	7002_H	Job processing already active (renewed instruction call when BUSY = 1)
0	1	8086_H	Illegal value for ID
0	1	$80E0_H$	• The length information under LEN is greater than the range of data to be transferred under DATA. or • Internal error If you send the frames directly to the telecontrol server (mode "Telecontrol"), make sure that the send cycle time is ≥ 1 second.
0	1	$80E1_H$	Timeout: • Increase the value of the "Connection monitoring time" in the configuration of the CP 1242-7 or • Check the connection partner.
0	1	$80E4_H$	Unknown ID: First call TC_CON.
0	1	$80E6_H$	No query in progress (instruction call not started)

DONE	ERROR	STATUS *	Meaning
0	1	$80E7_H$	Data to be sent not completely transferred: Repeat the job.
0	1	$80E8_H$	Remote partner cannot be reached. Check the connection parameters. In the "GPRS direct" mode, the message is output if the partner can be reached but is not accepting a connection request.
0	1	$80E9_H$	Connection establishment by remote partner: Check the connection partner. If necessary, terminate the connection with TC_DISCON and establish it again with TC_CON.
0	1	$80EA_H$	Error message from remote partner: • Check the connection partner. Enable the "TC_RECV" instruction on the communications partner. • If necessary, terminate the connection with TC_DISCON and establish it again with TC_CON.
0	1	$80EF_H$	SMS could not be sent: • Check whether the destination address (telephone number of the destination subscriber) exists. • Check whether the inserted SIM card allows sending of SMS. • Make sure that when the data block TCON_PHONE was created, the "Standard" option was selected for block access.
0	1	$80F1_H$	Sending of SMS messages is not enabled in the STEP 7 configuration of the CP: Enalble the "Allow SMS" option in the configuration of the CP.
0	1	$80F4_H$	Connection endpoint cannot be generated: Check the connection partner.
0	1	$80F5_H$	Invalid connection endpoint: • Connection establishment by TC_CON failed. or • Connection terminated by remote partner: First call TC_DISCON.
0	1	$80F6_H$	Format error of a parameter in the called data block (wrong length, wrong format or invalid value): Check the configuration of the "TC_CON..." SDT.

* Further statuses that are not listed here can be found in the status display is of the "RDREC" or "WRREC" instructions in the two middle status bytes (STATUS[2], STATUS[3]).

TC_RECV: Receive data via the GSM network

Meaning

The TC_RECV instruction allows the reception of data via programmed connections of the following types:

- ISO-ON-TCP connections
- SMS connections
  To receive SMS messages, the phone number of the sender must be configured in the STEP 7 configuration of the receiving CP (authorized phone numbers). The sender must support the CLIP function.
  The phone number of the connection partner must be entered in the "TCON_PHONE" SDT.
  Wake-up SMS messages are filtered out.
- Telecontrol connections

Note

Receiving SMS messages from different senders

If you want to receive SMS messages from different senders, you have two alternatives:

- You configure several connections (TC_CON, TC_RECV, TC_DISCON).
  or
- You may only enter no telephone number for only one configured connection in the required data block "TCON_PHONE" in the "PhoneNumber" parameter. When receiving messages, this is then interpreted as a placeholder for all authorized connection partners.

You will find more detailed information on the connection types in the description of the TC_CON instruction.

The ID parameter references the GPRS connection. The value of ID must correspond to the value used for ID by TC_CON.

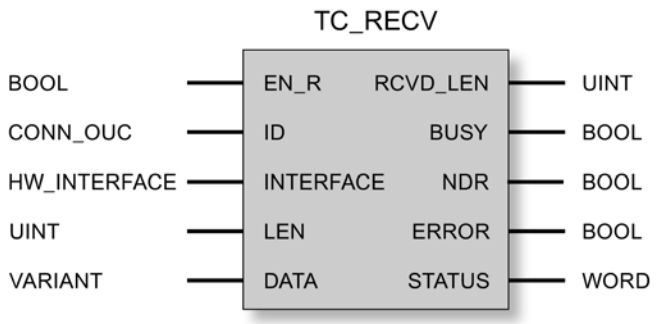
The INTERFACE parameter references the GPRS interface of the required local CP. The value must be the same as that used by TC_CON for INTERFACE.

The maximum amount of data to be received is specified with the LEN parameter.

The size of the data area specified in DATA must be at least as large as the number of bytes configured for LEN. Permitted data types in the data area specified in DATA are all except BOOL and ARRAY of BOOL. The received data is interpreted as if the sending partner had used the same data types.

The DB (system data type) used for the connection description of TC_RECV must differ from a DB used for TC_SEND.

Call interface in FBD representation



Explanation of the formal parameters

The following table explains all the formal parameters for the TC_RECV instruction

Parameter	Declaration	Data type	Range of values	Description
EN_R	INPUT	BOOL	0: Data reception locked 1: Data reception enabled	Enables / locks the reception of data. After setting 1 to 0, the program block receives data again.
ID	INPUT	CONN_OUC (WORD)	1...07FF$_h$	Reference to the relevant connection
INTERFACE	INPUT	HW_INTERFACE		Reference to the interface of the local CP 1242-7 (see STEP 7 > CP configuration > Telecontrol interface > "Hardware identifier")
LEN	INPUT	UINT	1...2048	(minimum) number of bytes of data to be received, maximum 2048
DATA	INOUT	VARIANT		Address reference to the receive data area of the CPU *
ENO	OUTPUT	BOOL	0: Error 1: Error-free	Enable output If there is a runtime error with the instruction, ENO = 0 is set.
RCVD_LEN	OUTPUT	UINT		Number of bytes of received data
BUSY	OUTPUT	BOOL	0: Execution of the instruction not started, completed or aborted 1: The instruction is executing	Display of the processing status of the instruction
DONE	OUTPUT	BOOL	0: - 1: The instruction executed successfully	This parameter indicates whether or not the job was completed without errors. For the meaning in conjunction with the parameters ERROR and STATUS, refer to Codes of the instruction.

Parameter	Declaration	Data type	Range of values	Description
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning in conjunction with the parameters DONE and STATUS, refer to Codes of the instruction.
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the parameters DONE and ERROR, refer to Codes of the instruction.

* For special features of the DATA parameter for SMS texts, refer to the next section.

Configuring SMS texts with the DATA parameter

The instruction references the received SMS text with the pointer of the type VARIANT of the DATA parameter to the data area of the CPU.

If DATA references an operand of the data type STRING for the SMS text, the first two bytes of the SMS text will be interpreted as length information of the data type STRING and not as SMS text.

One option for the correct text representation of SMS messages to be received is to convert an Array of BYTE or Array of CHAR to a text string using the conversion function Chars_TO_Strg. Chars_TO_Strg at the EN parameter is linked to the output parameter ENO of TC_RECV.

For SMS texts, the CP does not support all special characters, for example umlauts (ü, ä etc.). The specification GSM 03.38 applies. There may be additional restrictions imposed by the GSM network provider.

The codes BUSY, DONE and ERROR

The codes of DONE and ERROR are relevant only when BUSY = 0.

BUSY	DONE	ERROR	Meaning
0	0	0	No job being executed

You will find all other code combinations of DONE and ERROR in the following table.

The codes DONE, ERROR and STATUS

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

DONE	ERROR	STATUS *	Meaning
1	0	0000_H	Job executed without errors
0	0	7000_H	No job processing active (first instruction call)
0	0	7001_H	Job processing started (first instruction call)
0	0	7002_H	Job processing already active (renewed instruction call when BUSY = 1)

DONE	ERROR	STATUS *	Meaning
0	1	$80A3_H$	• An attempt is made to re-establish an existing connection. • An attempt is made to terminate a non-existent connection.
0	1	$80E0_H$	• The size of the data received for the range specified by DATA is greater than the length information in LEN. or • Internal error
0	1	8086_H	Illegal value for ID
0	1	$80E4_H$	Unknown ID: First call TC_CON.
0	1	$80E6_H$	No query in progress (instruction call not started)
0	1	$80F5_H$	Invalid connection endpoint: • Connection establishment by TC_CON failed. or • Connection terminated by remote partner: First call TC_DISCON.
0	1	$80F6_H$	Format error of a parameter in the called data block (wrong length, wrong format or invalid value) Check the configuration of the "TC_CON..." SDT.

* Further statuses that are not listed here can be found in the status display is of the "RDREC" or "WRREC" instructions in the two middle status bytes (STATUS[2], STATUS[3]).

TC_CONFIG: Transferring configuration data to a CP

Meaning

With the TC_CONFIG instruction, parameters of a the CP 1242-7 configured in STEP 7 can be modified. The configured values are not overwritten retentively. The overwritten values remain valid until TC_CONFIG is called again or until the station starts up again (cold restart after cycling power).

If the STEP 7 configuration data of the CP needs to be changed permanently, the instruction needs to be called again each time the station restarts (cold restart) or a modified project must be downloaded to the station.

The CONFIG parameter points to the memory area with the configuration data. The configuration data is stored in a data block (DB). The structure of the DB is specified by the system data type (SDT) IF_CONF.

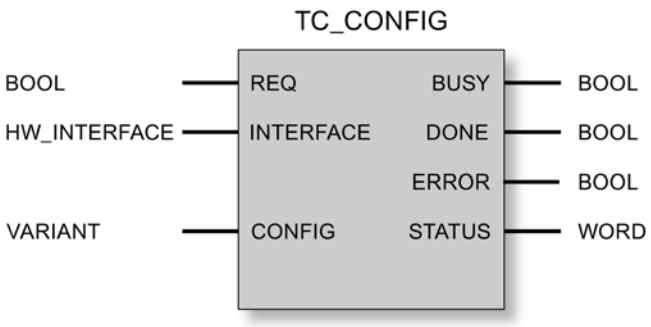
The configuration data to be modified on the CP is put together as necessary in blocks in IF_CONF "IF_CONF_..." for the individual parameters.

Parameters that are not intended to change as a result of the instruction are not entered in IF_CONF. They retain the value configured in STEP 7.

For detail information on assigning value to IF_CONF, refer to the section IF_CONF: SDT for telecontrol configuration data (Page 5172).

The INTERFACE parameter references the GPRS interface of the required local CP.

Call interface in FBD representation



Explanation of the formal parameters

The following table explains all the formal parameters for the TC_CONFIG instruction

Parameter	Declaration	Data type	Range of values	Description
REQ	INPUT	BOOL	0, 1	The instruction is started and the status codes initialized on a rising edge. Updating of the DONE, ERROR and STATUS status codes when there is a positive edge.
INTERFACE	INPUT	HW_INTERFACE (WORD)		Reference to the interface of the local CP 1242-7
CONFIG	INOUT	VARIANT	See also "IF_CONF: SDT for telecontrol configuration data	Reference to the memory area with the collected configuration data to be modified
ENO	OUTPUT	BOOL	0: Error 1: Error-free	Enable output If there is a runtime error with the instruction, ENO = 0 is set.
BUSY	OUTPUT	BOOL	0: Execution of the instruction not started, completed or aborted 1: The instruction is executing	Display of the processing status of the instruction
DONE	OUTPUT	BOOL	0: - 1: The instruction executed successfully	This parameter indicates whether or not the job was completed without errors. For the meaning in conjunction with the parameters ERROR and STATUS, refer to Codes of the instruction.
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning in conjunction with the parameters DONE and STATUS, refer to Codes of the instruction.
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the parameters DONE and ERROR, refer to Codes of the instruction.

The codes BUSY, DONE and ERROR

The codes of DONE and ERROR are relevant only when BUSY = 0.

BUSY	DONE	ERROR	Meaning
0	0	0	No job being executed

You will find all other code combinations of DONE and ERROR in the following table.

The codes DONE, ERROR and STATUS

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

DONE	ERROR	STATUS	Meaning
1	0	0000_H	Job executed without errors
0	0	7000_H	No job processing active (first instruction call)
0	0	7001_H	Job processing started (first instruction call)
0	0	7002_H	Job processing already active (renewed instruction call when BUSY = 1)
0	1	$80E6_H$	No query in progress (instruction call not started)
0	1	$80EB_H$	Query temporarily rejected (the CP is currently being configured by STEP 7).
0	1	$80F6_H$	Format error of a parameter in the called data block (wrong length, wrong format or invalid value) Check the "IF_CONF" SDT.
0	1	$80F7_H$	Wrong ID in the parameter fields of the configuration data: Check the "IF_CONF" SDT.

Other error messages

Other error messages

The following error messages are used for diagnostics purposes. You can obtain more information from the Siemens hotline.

DONE	ERROR	STATUS	Meaning
0	1	$80E0_H$	Internal error You should also note the possible meaning with the TC_SEND and TC_RECV instructions.

TCON_...: SDTs for the telecontrol connection establishment

System data types TCON_... for the TC_CON instruction

To configure a telecontrol connection using the TC_CON instruction, the CONNECT parameter of the instruction is used for the connection description.

The connection description is specified by the structure of the system data type (SDT). The structure of the relevant SDT contains the parameters necessary to establish the connection with the remote communications partner.

For different connection types that depend on the remote communications partner, the following SDTs are used:

- TCON_IP_RFC for ISO-on-TCP connections to IPv4 stations with CP 1242-7
- TCON_IP_V4 for UDP connections to IPv4 stations (sending only)
- TCON_PHONE for connections to SMS clients
- TCON_WDC for connections to telecontrol servers or stations that can be reached via the telecontrol server.

The parameter assignment of the connection description is made in a data block of the same type as the SDT.

Creating a DB of the type TCON_...

You will need to type in the data types of the relevant DBs with the keyboard. They are not displayed in the selection list. The data types are not case-sensitive.

To create a TCON_... DB, follow the steps outlined below:

1. Create a data block of the type "global DB" with block access "Standard".
2. Create an SDT in the table of the parameter configuration of the DB by assigning the name and typing in the required type (for example "TCON_IP_RFC") in the cell of the data type. The SDT and its parameters are created (see below).
3. Configure the parameters that are described below for each SDT type.

Reserved bits are not displayed.

System data type TCON_IP_RFC for connections to IPv4 stations

This connection type is supported only on ISO-on-TCP connections to communications partners with a fixed IP address. The CP must be configured for the "GPRS direct" mode.

Table 9-105 Parameters of TCON_IP_RFC

Byte	Parameter	Data type	Initial value	Description
0 ... 1	InterfaceID	HW_ANY		Reference to the interface of the local CP 1242-7 (see STEP 7 > CP configuration > Telecontrol interface > "Hardware identifier")
2 ... 3	ID	CONN_OUC	1...07FF$_h$	Reference to the GPRS connection. The ID is assigned and must be unique within the CPU. Here, the same value as that of the ID parameter of the TC_CON instruction must be used.
4	ConnectionType	BYTE	W#16#0C	Protocol variant 12 (C$_h$): ISO-on-TCP connection
5	ActiveEstablished	BOOL		Identifier for the type of connection establishment: • 0: Passive connection establishment • 1: Active connection establishment

Byte	Parameter	Data type	Initial value	Description
6 ... 7	-	-	-	- reserved -
8 ... 11	RemoteAddress	IP_V4		IP address of the connection partner
	ADDR	Array [1...4] of Byte		IP address of the relevant connection partner
12 ... 13	RemoteTSelector	TSelector		Remote T selector
	TSelLen	UINT		Length of the remote T selector "RemoteTSelector"
14 ... 45	TSel	Array [1...32] of Byte	any	Remote transport selector of the connection • When "ActiveEstablished" = 1: With active connection establishment, the T selector of the local partner must be the same as the T selector of the connection partner (passive connection establishment on the remote partner). • When "ActiveEstablished" = 0 correspondingly (passive connection establishment local, active connection establishment remote)
46 ... 47	LocalTSelector	TSelector		Local T selector
	TSelLen	UINT		Length of the local T selector "LOCAL_TSel"
48 ... 79	TSel	Array [1...32] of Byte	any	Local transport selector of the connection • When "ActiveEstablished" = 1: With active connection establishment, the T selector of the local partner must be the same as the T selector of the connection partner (passive connection establishment on the remote partner). • When "ActiveEstablished" = 0 correspondingly (passive connection establishment local, active connection establishment remote)

System data type TCON_IP_V4 for connections to IPv4 stations

This connection type is supported only for sending on UDP connections to communications partners with a fixed IP address.

To receive, ActiveEstablished = 0 must be set.

Table 9-106 Parameters of TCON_IP_V4

Byte	Parameter	Data type	Initial value	Description
0 ... 1	InterfaceID	HW_ANY		Reference to the interface of the local CP 1242-7 (see STEP 7 > CP configuration > Telecontrol interface > "Hardware identifier")
2 ... 3	ID	CONN_OUC	$1...07FF_h$	Reference to the GPRS connection. The ID is assigned and must be unique within the CPU. Here, the same value as that of the ID parameter of the TC_CON instruction must be used.
4	ConnectionType	BYTE	W#16#0B	Protocol variant 11 (B_h): UDP connection

Byte	Parameter	Data type	Initial value	Description
5	ActiveEstablished	BOOL		Identifier for the type of connection establishment: • 0: Passive connection establishment Setting for sending and receiving data. • 1: Active connection establishment Setting for sending data only.
6 ... 7	-	-	-	- reserved -
8 ... 11	RemoteAddress	IP_V4		IP address of the connection partner
	ADDR	Array [1...4] of Byte		IP address of the relevant connection partner
12 ... 13	RemotePort	UINT	1...65535	IP port of the connection partner Not relevant if ActiveEstablished = 0
14 ... 15	LocalPort	UINT	1...65535	Local IP port ("0" is not permitted) Not relevant if ActiveEstablished = 1

System data type TCON_PHONE for SMS connections

Note

Authorized phone numbers

The CP only accepts an SMS if the sending communication partner is authorized based on its phone number. These numbers are in configured for the CP in STEP 7 in the "authorized phone numbers" list.

SMS text

- Programmed SMS texts for SMS messages to be sent are accessed using the DATA parameter of the TC_SEND instruction.
- The text of a received SMS message is assigned to the address area of the CPU by the DATA parameter of the TC_RECV instruction.

Table 9-107 Parameters of TCON_PHONE

Byte	Parameter	Data type	Initial value	Description
0 ... 1	InterfaceID	HW_ANY		Reference to the interface of the local CP 1242-7 (see STEP 7 > CP configuration > Telecontrol interface > "Hardware identifier")
2 ... 3	ID	CONN_OUC	1...07FF$_h$	Reference to the GPRS connection. The ID is assigned and must be unique within the CPU. Here, the same value as that of the ID parameter of the TC_CON instruction must be used.
4	ConnectionType	BYTE	W#16#0E	Protocol variant 14 (E$_h$): SMS connection
5	ActiveEstablished	BOOL		Identifier for the type of connection establishment (not relevant for the CP 1242-7): • 0: Passive connection establishment (not relevant here) • 1: Active connection establishment

Byte	Parameter	Data type	Initial value	Description
6...7	-	-	-	- reserviert -
8 ... 31	PhoneNumber	STRING[22]		Call number of the connection partner Permitted values: Plus character (+) and numbers Note the exact notation of the international dialing code of the relevant phone number assigned by the network provider ("+" character or zeros). Without an entry for the PhoneNumber parameter, no connection partner is specified and SMS messages can be received reception from all authorized connection partners. Note the following during startup: Without an entry, TC_RECV first delivers the oldest received SMS message.

System data type TCON_WDC for connections to telecontrol servers or remote stations

You can configure the connection to the telecontrol server assigned to the S7-1200 or to a remote station that can be reached via the telecontrol server with TCON_WDC. The address data of the telecontrol server assigned to the CP can be found in STEP 7 in the "Telecontrol interface > Mode" tab of the CP. The telecontrol server or a remote station is addressed using the host name or the IP address.

The "RemoteWdcAddress" parameter of TCON_WDC specifies the Access ID of the connection partner.

Table 9-108 Parameters of TCON_WDC

Byte	Parameter	Data type	Initial value	Description
0 ... 1	InterfaceID	HW_ANY		Reference to the interface of the local CP 1242-7 (see STEP 7 > CP configuration > Telecontrol interface > "Hardware identifier")
2 ... 3	ID	CONN_OUC	$1...07FF_h$	Reference to the GPRS connection. The ID is assigned and must be unique within the CPU. Here, the same value as that of the ID parameter of the TC_CON instruction must be used.
4	ConnectionType	BYTE	W#16#0F	Protocol variant 15 (F_h): Telecontrol connection using an IP address
5	ActiveEstablished	BOOL		Identifier for the type of connection establishment: • 0: Passive connection establishment • 1: Active connection establishment

Byte	Parameter	Data type	Initial value	Description
6 ... 7	-	-	-	- reserved -
8 ... 11	RemoteWdcAddress	DWORD		Specifies the Access ID (hex). The access ID depends on the connection partner. • Connection to a remote CP: The access ID is made up of the following: – STEP 7 project number – Station number – Slot If the remote station has more than one GPRS-CP and you do not want to specify the path, the last byte for the slot must be set to 0. You will find the access ID in the STEP 7 project in the "CP authentication of the CP" parameter group. • Connection to the telecontrol server: Access ID = 0 • To only write to the process image of the CP: Access ID = DW#16#FEEDDADA

IF_CONF: SDT for telecontrol configuration data

Structure of the system data type IF_CONF for the TC_CONFIG instruction

The CONFIG parameter of the TC_CONFIG instruction references the memory area with the configuration data of the CP 1242-7 to be modified. The configuration data stored in a data block is described as a structure of the system data type (SDT) IF_CONF.

IF_CONF is made up of a header followed by fields that correspond to the parameters or parameter areas of the CP in the device properties of the STEP 7 project.

The CP configuration data to be modified is collected together as IF_CONF fields. Parameters that will not be modified are ignored in the IF_CONF structure and remain as they were configured in the STEP 7 project.

Creating the DB and the IF_CONF structures

You can create the parameters of the CP within the IF_CONF DB in one or more structures each with one or more fields.

You will need to type in the data types of the fields using the keyboard. They are not displayed in the selection list. The data types are not case-sensitive.

Follow the steps below to create IF_CONF:

1. Create a data block of the type "global DB" with block access "Standard".
2. Create a structure (data type "Struct") in the table of the parameter configuration of the DB. You can specify any name.
3. Under this structure add a header by assigning the name of the header and typing it in in the cell of the data type "IF_CONF_Header".
   The header of the structure and its three parameters (see below) is created.

4. Create a field for the first parameter to be changed by typing in the required data type (for example "IF_CONF_APN") in the cell of the data type.
5. Repeat the last step for all parameters you want to change on the CP using the TC_CONFIG instruction.
6. Finally, update the number of fields in the header in the "subfieldCnt" parameter.

Header of IF_CONF

Table 9-109 IF_CONF_Header

Byte	Parameter	Data type	Initial value	Description
0 ... 1	fieldType	UINT		Field type: Must always be 0.
2 ... 3	fieldId	UINT		Field ID: Must always be 0.
4 ... 5	subfieldCnt	UINT		Total number of fields contained in the structure

General parameters of the parameter fields

Each field has the following general parameters:

- Id
  This parameter identifies the field and must not be modified.
- Length
  This parameter indicates the length of the field. The value serves as information.
  Fields with strings and / or arrays have a variable length. Due to hidden bytes, the actual length of fields can be greater than the sum of the displayed parameters.
- Mode
  The following values are permitted to these parameters:

Table 9-110 Values of "Mode"

Value	Meaning
1	Permanent validity of the configuration data Not relevant for the CP 1242-7
2	Temporary validity of the configuration data, including deleting of existing permanent configuration data The permanent configuration data is replaced by the parameter fields of IF_CONF.

Field for the parameter area "GPRS access"

Table 9-111 IF_CONF_APN

Parameter	Data type	Initial value	Description
Id	UINT	4	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 174
Mode	UINT		Validity (1: permanent, 2: temporary)

Parameter	Data type	Initial value	Description
AccesspointGPRS	STRING [98]		APN: Name of the access point of the GSM network provider to the Internet
AccesspointUser	STRING [42]		APN user name
AccesspointPassword	STRING [22]		APN password

Field for the parameter area "CP identification"

Table 9-112 IF_CONF_Login

Parameter	Data type	Initial value	Description
Id	UINT	5	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 54
Mode	UINT		Validity (1: permanent, 2: temporary)
ModemName	STRING [22]		Access ID The value cannot be set.
ModemPassword	STRING [22]		Telecontrol password (max. 20 characters)

Field for the parameter area "Telecontrol server access"

This field is only used when the telecontrol server is addressed with a name that can be resolved by DNS. If the telecontrol server is addressed with its IP address, the "IF_CONF_TCS_IP_V4" field is used.

In STEP 7, the corresponding data is located in the "Mode" parameter area.

If there is more than one telecontrol server, use the field once per server.

Table 9-113 IF_CONF_TCS_Name

Parameter	Data type	Initial value	Description
Id	UINT	6	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 266
Mode	UINT		Validity (1: permanent, 2: temporary)
TcsName	-	-	- reserved -
	STRING [254]		Name of the telecontrol server that can be resolved by DNS
RemotePort	UINT		Port of the telecontrol server
Rank	UINT		Priority of the server [1, 2] 1 = main telecontrol server, 2 = substitute telecontrol server

Field for the parameter area "Telecontrol server access"

This field is only used when the telecontrol server is addressed by its IP address. If the telecontrol server is addressed by its DNS name, the "IF_CONF_TCS_Name" field is used.

In STEP 7, the corresponding data is located in the "Mode" parameter area.

If there is more than one telecontrol server, use the field once per server.

Table 9-114 IF_CONF_TCS_IP_v4

Parameter	Data type	Initial value	Description
Id	UINT	7	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 14
Mode	UINT		Validity (1: permanent, 2: temporary)
RemoteAddress	IP_V4		IP address of the telecontrol server
RemotePort	UINT		Port of the telecontrol server
Rank	UINT		Priority of the server [1, 2] 1 = main telecontrol server, 2 = substitute telecontrol server

Field for the "Mode" parameter area

In STEP 7, the corresponding data is located in the parameter areas "Mode" and Modem settings".

Table 9-115 IF_CONF_GPRS_Mode

Parameter	Data type	Initial value	Description
Id	UINT	8	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 10
Mode	UINT		Validity (1: permanent, 2: temporary)
GPRSmode	UINT		Mode of the CP: • 0 = Telecontrol • 1 = GPRS direct
TemporaryStation	BOOL		Bit 0: Temporary connection If this option is selected, the CP only establishes a temporary connection to send data. Once the frames have been transferred, the CP terminates the connection again. • 1: activated (temporary connection) • 0: deactivated (permanent connection)
SMS_Enabled	BOOL		Bit 1: Allow SMS Selecting the option allows the S7 station to send SMS messages. • 1: activated (SMS allowed) • 0: deactivated (no SMS)

Field for the "SMSC" parameter

In STEP 7, the corresponding data is located in the parameter area "Modem settings".

Table 9-116 IF_CONF_SMS_Provider

Parameter	Data type	Initial value	Description
Id	UINT	10	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 28

Parameter	Data type	Initial value	Description
Mode	UINT		Validity (1: permanent, 2: temporary)
SMSProvider	STRING [20]		Node number of the SMS center (SMSC) of the GSM network provider with which the contract was signed for this station.

Field for the "PIN" parameter

In STEP 7, the corresponding data is located in the parameter area "Modem settings".

Table 9-117 IF_CONF_PIN

Parameter	Data type	Initial value	Description
Id	UINT	11	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 16
Mode	UINT		Validity (1: permanent, 2: temporary)
Pin	STRING [8]		PIN of the SIM card inserted in the SIM card The parameter is not relevant if the PIN was correctly configured. If the PIN was incorrectly configured, the correct PIN can be entered.

Field for monitoring times

In STEP 7, the corresponding data is located in the parameter areas "Keepalive timeout" and Operating mode".

Table 9-118 IF_CONF_TC_Timeouts

Parameter	Data type	Initial value	Description
Id	UINT	12	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 12
Mode	UINT		Validity (1: permanent, 2: temporary)
KeepAliveTimeout	-	-	- Reserved - (cannot be set)
SendTimeout	UINT		Connection monitoring time: Monitoring time of the connection to the communications partner (seconds) Relevant in the modes "Telecontrol" and "GPRS direct"
RedialTimeout	UINT		Dialing repetition delay: Basic value for the wait time until the next attempt to establish a connection following an unsuccessful connection establishment. After every 3 attempts, the basic value is doubled up to a maximum of 900 s. Range of values: 10 to 600 s. If a substitute telecontrol server is configured, the CP attempts to connect to it at the 4th dialin attempt. Example: Basic value 20 results in the following dialing intervals: three times 20 s, three times 40 s, three times 80 s etc. up to a maximum of 900 s. Not relevant for SMS connections

Field for the "Wake up right" parameter area

Table 9-119 IF_CONF_WakeupList

Parameter	Data type	Initial value	Description
Id	UINT	13	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 246
Mode	UINT		Validity (1: permanent, 2: temporary)
WakeupPhone [1...10]	ARRAY [1...10] of STRING [22]		Phone number subscriber authorized to wake up The asterisk (*) at the end of a call number is used a placeholder for direct dialing numbers.

Field for the "Preferred GSM networks" parameter area

Table 9-120 IF_CONF_PrefProvider

Parameter	Data type	Initial value	Description
Id	UINT	14	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 46
Mode	UINT		Validity (1: permanent, 2: temporary)
Provider [1...5]	ARRAY [1...5] of STRING [6]		Alternative GSM networks with priority 1 to 5 into which the CP dials. Up to 5 networks can be configured. No. 1 with highest priority, no. 5 with lowest priority. Entry of the Public Land Mobile Network (PLMN) of the network provider consisting of Mobile Country Code (MCC) and Mobile Network Code (MNC). Example (test network of Siemens AG): 26276

Field for the "DNS configuration" parameter area

Table 9-121 IF_CONF_DNS

Parameter	Data type	Initial value	Description
Id	UINT	16	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 14
Mode	UINT		Validity (1: permanent, 2: temporary)
DNS_IP [1]	IP_V4		IP address of the 1st domain name system server
DNS_IP [2]	IP_V4		IP address of the 2nd domain name system server

Field for the "Time-of-day synchronization" parameter area

Table 9-122 IF_CONF_NTP

Parameter	Data type	Initial value	Description
Id	UINT	17	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 24
Mode	UINT		Validity (1: permanent, 2: temporary)
NTP_IP [1]	ARRAY [1...4] of IP_V4		IP address of NTP server 1
...	...		(IP address of NTP server 2...3)
NTP_IP [4]	ARRAY [1...4] of IP_V4		IP address of NTP server 4

Block for activating / deactivating TeleService users

SDT for activating or deactivating TeleService users already configured in the STEP 7 project of the CP. In STEP 7, the corresponding data can be found in the parameter area "TeleService settings" > "TeleService user management".

Table 9-123 IF_CONF_GPRS_UserList

Parameter	Data type	Initial value	Description
Id	UINT	19	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 506
Mode	UINT		Validity (1: permanent, 2: temporary)
GPRS_User [1...10]	ARRAY [1...10] of GPRS_User		TeleService user no. 1 to max. no. 10

The array is formed from the parameter records for the TeleService users ("GPRS_User" [1...n]).

Table 9-124 GPRS_User [n] (parameter for TeleService user)

Parameter	Data type	Initial value	Description
UserName [n]	STRING [22]		TeleService user name
Password [n]	STRING [22]		- The string must be empty! -
Diag_Allowed [n]	BOOL		- Reserved - (cannot be set)
Teleserv_Allowed [n]	BOOL		Activation of the TeleService user • 0 = user is deactivated • 1 = user is activated
FW_Load_Allowed [n]	BOOL		- Reserved - (cannot be set)

Field for setting the parameters for TeleService access (DNS name of the server)

Access data of the TeleService server (switching station).

In STEP 7, the corresponding data is located in the parameter area "TeleService settings".

If there is more than one TeleService server, use the field once per server.

Table 9-125 IF_CONF_TS_Name

Parameter	Data type	Initial value	Description
Id	UINT	20	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 266
Mode	UINT		Validity (1: permanent, 2: temporary)
ts_name	String [254]		Name of the TeleService server that can be resolved by DNS
RemotePort	UINT		Port of the engineering station
Rank	UINT		Priority of the server [1] or [2] 1 = server 1, 2 = server 2

Field for setting the parameters for TeleService access (IP address of the server)

Access data of the TeleService server (switching station).

In STEP 7, the corresponding data is located in the parameter area "TeleService settings".

If there is more than one TeleService server, use the field once per server.

Table 9-126 IF_CONF_TS_IF_V4

Parameter	Data type	Initial value	Description
Id	UINT	21	ID of the parameter field
Length	UINT		Length of the parameter field in bytes: 14
Mode	UINT		Validity (1: permanent, 2: temporary)
RemoteAddress	IP_V4		IP address of the TeleService server
RemotePort	UINT		Port of the TeleService server
Rank	UINT		Priority of the server [1] or [2] 1 = server 1, 2 = server 2

Point-to-point

Overview of Freeport communication

STEP 7 offers extended instructions that can be used for Freeport communication with a protocol specified in the user program. These instructions can be divided into two categories:

- Configuration instructions
- Communication instructions

Note

CPU capacity

The Freeport instructions communicate with the communication modules by reading or writing data records.
When using the instructions, you should therefore check the capacity of the CPU for reading and writing data records.
If several instructions need to read or write data records simultaneously on a CPU, the user program may need to create a time offset between the calls of each instruction.

Configuration instructions

Before the user program can start the Freeport communication, the communication interface and the parameters for sending and receiving of data must be configured.

The interface configuration and the data configuration can be set for each CM in the device configuration or with the following instructions of your user program:

- Port_Config (Page 5184)
- Send_Config (Page 5187)
- Receive_Config (Page 5188)
- P3964_Config (Page 5193)

NOTICE

Device configuration <-> Configuration instructions

The device configuration parameters are transferred to the CM upon each Power On of the CPU (return of voltage).

The parameters of the configuration instructions are transferred to the CM as defined in your user program.

The parameters of the device configuration are not synchronized with the parameters of the configuration instructions, which means the parameters of the configuration instructions are not applied to the CPU device configuration.
With your user program, you determine the parameters that apply in the CM and when they apply.

Communication instructions

The user program uses the instructions for Freeport communication to send data to and receive data from the communication interfaces. The CMs send data to and receive data from the communication stations.

- Send_P2P (Page 5194)
- Receive_P2P (Page 5197)

Note

Data consistency

- If the data to be sent is transmitted consistently, it cannot be changed after the positive edge at the REQ parameter until DONE has been set by the Send_P2P instruction.
- If the receive data is to be read consistently, it may only be evaluated in the cycle in which NDR = TRUE.

The receive buffer can be reset with additional instructions and special RS232 signals can be queried and set.

- Receive_Reset (Page 5199)
- Signal_Get (Page 5199)
- Signal_Set (Page 5200)

The following instructions let you read or write extended functions, as long as these are supported by the module.

- Get_Features (Page 5201)
- Set_Features (Page 5202)

All Freeport instructions work asynchronously. The instructions must therefore be called until the DONE output parameter indicates that execution is complete.

The user program can determine the send and receive status with the help of the query architecture. Send_P2P and Receive_P2P can be run at the same time. The communication modules buffer the send and receive data as required until a module-specific maximum buffer size has been reached.

Note

Resolution of bit times

The number of bit times is specified with the configured data transmission rate for different parameters. Specifying the parameter in bit times makes it independent of the data transmission rate. All parameters with unit of bit times can be specified with a maximum number of 65535.

Using the instructions

The Freeport instructions must be called cyclically to query received data or the end of transmission for a send process.

Depending on the data volume, data transmission may take place over several calls (program cycles).

If a command is completed with DONE = TRUE or NDR = TRUE, it has been executed without errors.

Note

Backing up STATUS

The DONE, NDR, ERROR and STATUS parameters are only available for one block cycle. To display the STATUS, you should therefore copy it to a free data area.

Master

Typical sequence for a master:

1. The Send_P2P instruction triggers transmission to the CM.
   Data transmission is initiated by a positive edge at the REQ input.
2. The Send_P2P instruction is executed in subsequent cycles to query the status of the transmission process.
3. When the Send_P2P instruction signals that transmission is complete at the DONE output, the user code can prepare the receipt of the answer.
4. The Receive_P2P instruction is run repeatedly to query an answer. If the CM has acquired response data, the Receive_P2P instruction copies the response to the CPU and signals that new data has been received at the NDR output.
5. The user program can process the response.
6. Back to step 1 and repetition of the sequence.

Slave

Typical sequence for a slave:

1. The user program runs the Receive_P2P instruction in each cycle.
2. If the CM has received a request, the Receive_P2P signals that new data is available at the NDR output and the request is copied to the CPU.
3. The user program processes the request and creates a response.
4. The response is returned to the master with the Send_P2P instruction.
5. The Send_P2P instruction must be run repeatedly to ensure that the send process is actually taking place.
6. Back to step 1 and repetition of the sequence.

The slave must ensure that Receive_P2P is called up often enough so that a transmission can be received by the master before it cancels the process due to a timeout while waiting for the response. To do so, the user program Receive_P2P can be called from within a cycle OB whose cycle time is sufficiently short so that the master can receive a transmission before the timeout setting expires. If the OB cycle time is set so that two runs can take place during the timeout setting of the master, the user program can receive all transmissions without any losses.

General parameters for Freeport operations

Table 9-127 General input parameters of the Freeport instructions

Parameter	Description
REQ	Data transmission is initiated by a positive edge at the REQ input. Another edge at REQ may only be created after the command has been completed (DONE or ERROR). Data transmission can take several calls (program cycles), depending on the data volume. When you add a Freeport instruction to your program, STEP 7 prompts you to specify the instance DB (or to have STEP 7 create a corresponding instance DB). Use a unique DB for each PtP instruction call.
PORT	A port address is assigned during configuration of the communication module. The PORT parameter communicates assignment to a specific communication module to the instruction. You can select a symbolic name for the standard port after the configuration. The assigned CM port value is the "Hardware ID" property of the device configuration with S7-1500 and the "Input address" with S7-300/400. The symbolic port name is assigned in the symbol table.

The output parameters DONE, NDR, ERROR and STATUS of the Freeport instructions indicate the execution status of the Freeport functions.

Table 9-128 Output parameters DONE, NDR, ERROR and STATUS

Parameter	Data type	Standard	Description
DONE	Bool	FALSE	Set to TRUE for one cycle to indicate that the last request was completed with errors; otherwise FALSE.
NDR	Bool	FALSE	Set to TRUE for one cycle to indicate that new data has been received; otherwise FALSE.
ERROR	Bool	FALSE	Set to TRUE for one cycle to indicate that the last request was completed with errors; the corresponding error code can be found in STATUS; otherwise FALSE.
STATUS	Word	16#0000 or 16#7000	Result status: • If the bit DONE or NDR is set, STATUS is set to 0/16#7000 or to a specific error code. • If the bit ERROR is set, STATUS displays an error code. • If none of the bits listed above is set, the instruction can return status results that describe the current status of the function. The value in STATUS is valid until you call this instruction again (with one and the same port address).

Table 9-129 COM_RST in/out parameter

Parameter	Data type	Standard	Description
COM_RST	Bool	FALSE	Initialization of the instruction You initialize the instruction with TRUE. COM_RST is then reset to FALSE. Note: You must set COM_RST to TRUE during startup and should not subsequently change the parameter (that is, do not assign a value when you call the instruction). COM_RST is reset by the instruction following initialization of the instance DB.

Note

Please note that the parameters DONE, NDR, ERROR and STATUS are only set for one cycle.

Table 9-130 Shared error codes

Error code	Description
16#0000	No error
16#7000	Function not active
16#7001	Initial call after request started.
16#7002	Subsequent call after request started.
16#8x3A	Invalid pointer in parameter x

Table 9-131 Shared error classes of the STATUS parameter

Description of the class	Error classes	Description
Port configuration	16#81Ax	For the description of frequent errors in the interface configuration
Send configuration	16#81Bx	For the description of errors in the send configuration
Receive configuration	16#81Cx	For the description of errors in the receive configuration
Sending	16#81Dx	For the description of runtime errors during sending
Receiving	16#81Ex	For the description of runtime errors during receiving
RS232 accompanying signals	16#81Fx	For the description of errors in connection with signal processing

Port_Config: Configure PtP communication port

Description

The Port_Config instruction (port configuration) allows you to change parameters such as the data transmission rate in runtime using your program. The data pending in the CM is deleted with the execution of Port_Config.

Configuration changes of Port_Config are saved on the CM and not in the CPU. When the voltage returns, the CM is configured with the data saved in the device configuration.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the transmission of data to the CM upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	The CM port value can be found in the "Hardware ID" property of the device configuration (S7-1500) (input address; S7-300/400) once you have configured the CM. The symbolic port name is assigned in the "System constants" tab of the PLC tag table. In S7-300/400/WinAC systems, the input address assigned in HWCN must be assigned to the CM port.
PROTOCOL	IN	UInt	Word	0	Protocol • 0 = Freeport protocol • 1 = Protocol 3964(R)
BAUD	IN	UInt	Word	6	Data transmission rate of the port: • 1 = 300 Bit/s • 2 = 600 Bit/s • 3 = 1200 Bit/s • 4 = 2400 Bit/s • 5 = 4800 Bit/s • 6 = 9600 Bit/s • 7 = 19200 Bit/s • 8 = 38400 Bit/s • 9 = 57600 Bit/s • 10 = 76800 Bit/s • 11 = 115200 Bit/s
PARITY	IN	UInt	Word	1	Parity of the port: • 1 = no parity • 2 = even parity • 3 = odd parity • 4 = mark parity • 5 = space parity • 6 = any
DATABITS	IN	UInt	Word	1	Bits per character: • 1 = 8 data bits • 2 = 7 data bits
STOPBITS	IN	UInt	Word	1	Stop bits: • 1 = 1 stop bit • 2 = 2 stop bits

Parameter	Declaration	Data type		Default	Description
		S7-1500	S7-300/400/WinAC		
FLOWCTRL	IN	UInt	Word	1	Flow control: • 1 = no flow control • 2 = XON/XOFF • 3 = Hardware RTS always ON • 4 = Hardware RTS switched • 5 = Hardware RTS always ON, ignore DTR/DSR
XONCHAR	IN	Char		16#0011	Specifies the character that serves as XON character. It is typically a DC1 character (11H). This parameter is only evaluated when software flow control is active.
XOFFCHAR	IN	Char		16#0013	Specifies the character that serves as XOFF character. It is typically a DC3 character (13H). This parameter is only evaluated when software flow control is active.
WAITIME	IN	UInt	Word	2000	Specifies how long to wait for an XOFF character after receipt of an XON character or how long to wait for a CTS = ON signal after CTS = OFF (0 to 65535 ms). This parameter is only evaluated when flow control is active.
MODE	IN	USInt	Byte	0	Operating mode Valid operating modes are: • 0 = Full duplex (RS232) • 1 = Full duplex (RS422) four-wire operation (point-to-point) • 2 = Full duplex (RS 422) four-wire mode (multipoint master; CM PtP (ET 200SP)) • 3 = Full duplex (RS 422) four-wire mode (multipoint slave; CM PtP (ET 200SP)) • 4 = Half duplex (RS485) two-wire mode
LINE_PRE	IN	USInt	Byte	0	Receive line initial state Valid initial states are: • 0 = "no" initial state • 1 = signal R(A)=5 V, signal R(B)=0 V (break detection): Break detection is possible with this initial state. Can only be selected with: "Full duplex (RS422) four-wire operation (point-to-point connection)" and "Full duplex (RS422) four-wire operation (multipoint slave)". • 2 = signal R(A)=0 V, signal R(B)=5 V: This default setting corresponds to the idle state (no active send operation). No break detection is possible with this initial state.
CABLE_BRK	IN	USInt	Byte	0	Break detection The following settings are valid: • 0 = break detection deactivated • 1 = break detection activated

Parameter	Declaration	Data type		Default	Description
		S7-1500	S7-300/400/WinAC		
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Port_Config instruction The instruction is initialized with 1. The instruction then resets COM_RST to 0.
DONE	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed without errors
ERROR	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed with errors
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 5203))

Send_Config: Configure PtP sender

Description

The Send_Config instruction (send configuration) allows you to change send parameters in runtime using your program (conditions that identify the start and the end of the data to be sent). Any data pending in a CM is deleted when Send_Config is executed.

Configuration changes of Send_Config are saved on the CM and not in the CPU. The parameters saved in the device configuration are restored once the voltage returns to the CPU or the communication module.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Activates the configuration change upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	The CM port value can be found in the "Hardware ID" property of the device configuration (S7-1500) (input address; S7-300/400) once you have configured the CM. The symbolic port name is assigned in the "System constants" tab of the PLC tag table. In S7-300/400/WinAC systems, the input address assigned in HWCN must be assigned to the CM port.
RTSONDLY	IN	UInt	Word	0	Number of milliseconds to wait after activation of RTS before a transmission of send data is started. This parameter is only valid if the hardware flow control is active. The valid range is 0 to 65535 ms. The value 0 deactivates the function.
RTSOFFDLY	IN	UInt	Word	0	Number of milliseconds to wait after transmission of send data before RTS is deactivated: This parameter is only valid if the hardware flow control is active. The valid range is 0 to 65535 ms. The value 0 deactivates the function.

Parameter	Declaration	Data type		Default	Description
		S7-1500	S7-300/400/WinAC		
BREAK	IN	UInt	Word	0	This parameter specifies that a BREAK is to be sent at the start of each frame for the specified number of bit times. The maximum is 65535 bit times. The value 0 deactivates the function.
IDLELINE	IN	UInt	Word	0	This parameter specifies that the line is to remain in idle for the specified number of bit times prior to the start of each frame. The maximum is 65535 bit times. The value 0 deactivates the function.
USR_END	IN	STRING[2]		0	Input of end delimiters. No more than 2 end delimiters can be configured. All data including the end delimiter(s) is sent, independent of the configured frame length.
APP_END	IN	STRING[5]		0	Input of characters to be appended. You can append up to 5 characters.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Send_Config instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
DONE	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed without errors
ERROR	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed with errors
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 5203))

Receive_Config: Configure PtP recipient

Description

The Receive_Config instruction (receive configuration) allows you to change receive parameters in runtime using your program. This instruction configures the conditions that mark the start and the end of received data. Any data pending in a CM is deleted when Receive_Config is executed.

Configuration changes of Port_Config are saved on the CM and not in the CPU. The parameters saved in the device configuration are restored once the voltage returns to the CPU or the communication module.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Activates the configuration change upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	The CM port value can be found in the "Hardware ID" property of the device configuration (S7-1500) (input address; S7-300/400) once you have configured the CM. The symbolic port name is assigned in the "System constants" tab of the PLC tag table. In S7-300/400/WinAC systems, the input address assigned in HWCN must be assigned to the CM port.
Receive_Conditions	IN	Variant	Any	-	The data structure of Receive_Conditions specifies the start and end conditions used to identify the start and end of a frame.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Receive_Config instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
DONE	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed without errors
ERROR	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed with errors
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 5203))

.

Start conditions for the Receive_P2P instruction

The Receive_P2P instruction uses the configuration specified in the device configuration or by the Receive_Config instruction to determine the start and end of Freeport communication frames. The start of the frame is defined by the start conditions. The start of the frame can be determined with one or several start conditions.
If Break as well as Idle Line is activated, Break must be met first and then Idle Line as well. After that, one of the other conditions (start character or start sequence) is sufficient to start data transmission.
The start condition "Random character" cannot be combined with other start conditions.

Data type structure of the Receive_Conditions parameter, part 1 (start conditions)

Table 9-132 Structure of Receive_Conditions for start conditions

Parameter	Declaration	Data type	Default	Description
START.START COND	IN	Word	16#0002	Specifying the start condition • 01H - detection of the start character • 02H - Any character • 04H - detection of a line break • 08H - detection of an idle line • 10H - detection of start sequence 1 • 20H - detection of start sequence 2 • 40H - detection of start sequence 3 • 80H - detection of start sequence 4 The start conditions can be combined by adding the values together.
START.IDLETI ME	IN	Word	16#0028	The number of bit times required in the idle state for a new frame start to be detected (default value: W#16#28). Only in connection with the condition "Detection of an idle line". 0 to FFFF
START.START CHAR	IN	Byte	16#0002	The start character for the condition "Start character". (default value: B#16#2)
START.SEQ[1]. CTL	IN	Byte	0	Start sequence 1, deactivate/activate comparison for each character: (default value: B#16#0) These are the activation bits for each character of the start character string. • 01H - character 1 • 02H - character 2 • 04H - character 3 • 08H - character 4 • 10H - character 5 When a bit is deactivated for a specific character, this means that each character in this position in the character string represents a valid start character string (e.g. 1FH = all 5 characters interpreted).
START.SEQ[1]. STR[1] .. START.SEQ[1]. STR.[5]	IN	Char[5]	0	Start sequence 1, start character (5 characters).
START.SEQ[2]. CTL	IN	Byte	0	Start sequence 2, deactivate/activate comparison for each character. Default value: B#16#0)
START.SEQ[2]. STR[1] .. START.SEQ[2]. STR.[5]	IN	Char[5]	0	Start sequence 2, start character (5 characters).
START.SEQ[3]. CTL	IN	Byte	0	Start sequence 3, deactivate/activate comparison for each character. Default value: B#16#0

Parameter	Declaration	Data type	Default	Description
START.SEQ[3]. STR[1] .. START.SEQ[3]. STR.[5]	IN	Char[5]	0	Start sequence 3, start character (5 characters).
START.SEQ[4]. CTL	IN	Byte	0	Start sequence 4, deactivate/activate comparison for each character. Default value: B#16#0
START.SEQ[4]. STR[1] .. START.SEQ[4]. STR.[5]	IN	Char[5]	0	Start sequence 4, start character (5 characters),

Example

Have a look at the following received data in hexadecimal coding: "68 10 aa 68 bb 10 aa 16". The configured start character strings are available in the following table. Start character strings are evaluated once the first character 68H has been successfully received. After the fourth character has been successfully received (the second 68H), start condition 1 has been met. Once the start conditions have been met, the evaluation of the end conditions starts.

Processing of the start character string can be canceled due to different errors in parity, framing or time intervals between characters. These errors prevent receipt of the data because the start condition has not been met (an error message is output).

Table 9-133 Start conditions

Start condition	First character	First character +1	First character +2	First character +3	First character +4
1	68H	xx	xx	68H	xx
2	10H	aaH	xx	xx	xx
3	dcH	aaH	xx	xx	xx
4	e5H	xx	xx	xx	xx

End conditions for the Receive_P2P instruction

The end of a frame is defined by the first occurrence of one or more configured end conditions.

You can configure the end conditions either in the properties of the communication interface in the device configuration, or with the Receive_Config instruction. The receive parameters (start and end conditions) are reset to the settings in the device configuration each time the voltage returns to the CPU or the communication module. When the STEP 7 user program executes Receive_Config, the settings are set to the parameters of Receive_Config .

Data type structure of the Receive_Conditions parameter, part 2 (end conditions)

Table 9-134 Structure of Receive_Conditions for end conditions

Parameter	Declaration	Data type	Default	Description
END.ENDCOND	IN	Word	0	This parameter specifies the condition for the frame end: • 01H - response timeout • 02H - message timeout • 04H - character delay time • 08H - maximum frame length • 10H - read message length from message (N+LEN+M) • 20H - end sequence • 40H - fixed frame length
END.FIXLEN	IN	Word	1	Fixed frame length: Only used if the end condition "Fixed length" has been selected. 1 to 4000 bytes (up to 4 KB depending on the module)
END.MAXLEN	IN	Word	1	Maximum frame length: Only used if the end condition "Maximum length" has been selected. 1 to 4000 bytes (up to 4 KB depending on the module)
END.N	IN	Word	0	Byte position of the length field in the frame. Only used with end condition N+LEN+M. 1 to 4000 bytes (up to 4 KB depending on the module)
END.LENGTHSIZE	IN	Word	0	Size of the length field (1, 2, or 4 bytes). Only used with end condition N+LEN+M.
END.LENGTHM	IN	Word	0	Number of characters after the length field that are not included in the value of the length field. This entry is only used with end condition N+LEN+M. 0 to 255 bytes
END.RCVTIME	IN	Word	200	Specify the wait time for the first received character after a frame has been sent. The receive instruction is terminated with an error message if a character is not received within the specified time. This information is used only with the condition "Response timeout". (0 to 65535 ms). Note: This parameter cannot be used as sole end criterion but only in connection with at least one other end condition.
END.MSGTIME	IN	Word	200	Specify how long to wait for receipt of the complete frame after receipt of the first character. This parameter is used only if the condition "Message timeout" is selected. (0 to 65535 ms)
END.CHARGAP	IN	Word	12	Enter the maximum number of bit times between characters. If the number of bit times between characters exceeds the specified value, the end condition has been met. This information is used only with the condition "Character delay time". (0 to 65535 bit times)

Parameter	Declaration	Data type	Default	Description
END.SEQ.CTL	IN	Byte	0	Character sequence 1, deactivate/activate comparison for each character: These are the activation bits for each character of the end character string. Character 1 is bit 0, character 2 is bit 1, ..., character 5 is bit 4. If a bit is deactivated for a specific character, this means that each character represents a congruence at this position of the character string.
END.SEQ.STR[1] .. END.SEQ.STR[5]	IN	Char[5]	0	Character string 1, start character (5 characters)

Table 9-135 General parameters of the Receive_P2P instruction

Parameter	Declaration	Data type	Default	Description
GENERAL.MBUF_SIZE	IN	Byte	255	Input number of frames that are to be buffered in the receive buffer of the CM. If no other conditions are active that influence the reaction of the receive buffer (prevent timeout, data flow control), additional frames are discarded once the limit has been reached. (1 to 255 frames)
GENERAL.OW_PROT	IN	Byte	0	Activates the no overwriting function of the buffered frame if the CM receives a new frame and the receive buffer of the CM was not yet read. This step prevents already buffered received frames from being lost. • 0 - not activated • 1 - activated
GENERAL.CLR_MBUF	IN	Byte	0	Activates deletion of the receive buffer during CPU startup. The receive buffer is automatically deleted when the CPU switches from STOP to RUN. The receive buffer only contains frames received after CPU startup. • 0 - not activated • 1 - activated

P3964_Config: Configuring the 3964(R) protocol

Description

The P3964_Config instruction (protocol configuration) allows you to change protocol parameters for 3964(R), such as character delay time, priority and block check, in runtime using your program.

Configuration changes of P3964_Config are saved on the CM and not in the CPU. The parameters saved in the device configuration are restored once the voltage returns to the CPU or the communication module.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the instruction upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	The CM port value can be found in the "Hardware ID" property of the device configuration (S7-1500) (input address; S7-300/400) once you have configured the CM. The symbolic port name is assigned in the "System constants" tab of the PLC tag table. In S7-300/400/WinAC systems, the input address assigned in HWCN must be assigned to the CM port.
BCC	IN	SInt	Byte	1	Activates/deactivates the use of the block check • 0 = without block check • 1 = with block check
Priority	IN	SInt	Byte	1	Selection of the priority • 0 = low priority • 1 = high priority
CharacterDelayTime	IN	UInt	Word	16#00DC	Setting the character delay time (depending on the set data transmission rate) (default value: 220 ms) 1 ms to 65535 ms
AcknDelayTime	IN	UInt	Word	16#07D0	Setting the acknowledgment delay time (depending on the set data transmission rate) (default value: 2000 ms) 1 ms to 65535 ms
BuildupAttempts	IN	SInt	Byte	16#0006	Setting the number of connection attempts (default value: 6 connection attempts) 1 to 255
RepetitionAttempts	IN	SInt	Byte	16#0006	Setting the number of transmission attempts (default value: 6 connection attempts) 1 to 255
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the P3964_Config instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
DONE	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed without errors
ERROR	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed with errors
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 5203))

Send_P2P: Sending data

Description

The Send_P2P instruction (send point-to-point data) starts the transmission of data and transmits the contents of the assigned buffer to the communication module. The CPU program is still being executed while the CM sends the data with the data transmission rate. Only one

send instruction per communication module may be pending at any time. The CM signals an error if a second Send_P2P instruction is executed while the CM is already sending a frame.

Parameters

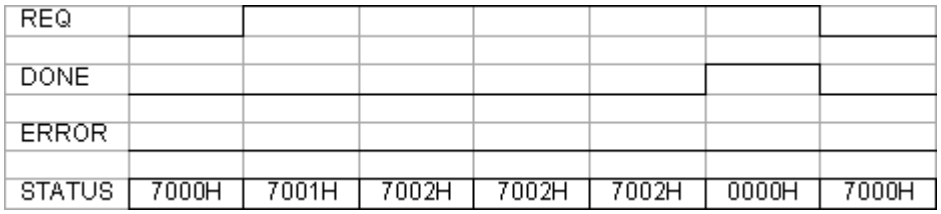
Parameter	Declaration	Data type		Default	Description
		S7-1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the transmission of data to the CM upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	The CM port value can be found in the "Hardware ID" property of the device configuration (S7-1500) (input address; S7-300/400) once you have configured the CM. The symbolic port name is assigned in the "System constants" tab of the PLC tag table. In S7-300/400/WinAC systems, the input address assigned in HWCN must be assigned to the CM port.
BUFFER	IN	Variant	Any	0	This parameter points to the memory area of the send buffer. Notes: • Boolean data and Boolean fields are not supported. • If the send buffer is in the optimized memory area, the maximum permitted length of the sent data is 1024 bytes. Exception: Arrays of Byte, Word or DWord are supported up to a length of 4096 bytes. • If the send buffer is a String or WString, the content of the string is transferred without the current and maximum length.
LENGTH	IN	UInt	Word	0	Length in bytes of the data to be transferred. The memory area addressed in the BUFFER parameter is completely transmitted with LENGTH = 0.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Send_P2P instruction The instruction is initialized with 1. The instruction then resets COM_RST to 0.
DONE	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed without errors
ERROR	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed with errors
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 5203))

Parameters

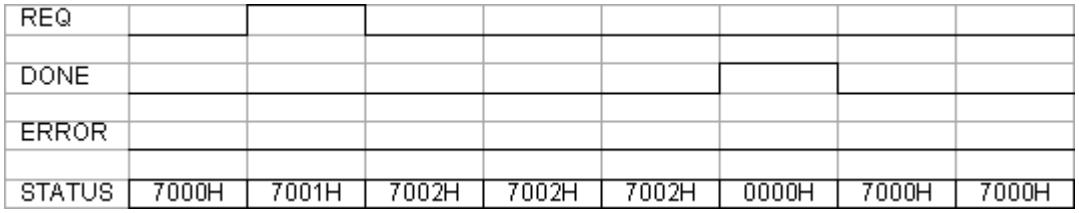
The DONE and ERROR outputs are in FALSE status when a send instruction is being processed. At the end of the send instruction, one of the DONE or ERROR outputs is set to TRUE for one cycle to signal the status of the send instruction. The error code at the STATUS output can be evaluated when the status of ERROR is TRUE.

The instruction outputs the status 16#7001 when the communication interface accepts the send data. Subsequent executions of Send_P2P output the value 16#7002 if the CM is still sending. At the end of the send instruction, the CM outputs the status 16#0000 for the send instruction (if no error has occurred). Subsequent executions of Send_P2P with REQ = 0 output the status 16#7000 (free).

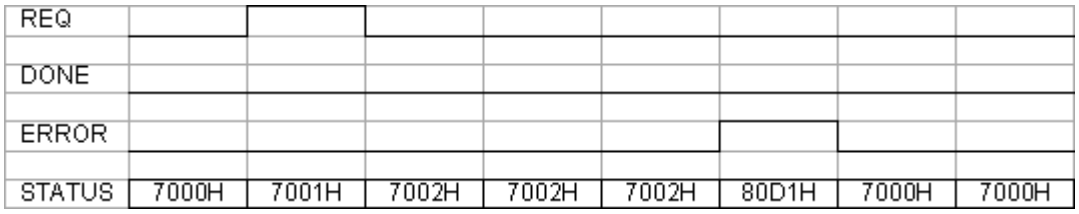
The diagram below shows the relationship between the output values and REQ. It is based on the assumption that the instruction is called cyclically to check the status of the send process (indicated by the STATUS values).



The figure below shows how the DONE and STATUS parameters are only valid for one cycle if a pulse is pending at the REQ line (for one cycle) to trigger the send instruction.



The figure below shows the relationship of the DONE, ERROR and STATUS parameters in case of an error.



The DONE, ERROR and STATUS values are only valid until Send_P2P is executed again with the same instance DB.

Using LENGTH and BUFFER parameters in communication operations

Interaction of LENGTH and BUFFER parameters for Send_P2P

The minimum data size sent by the Send_P2P instruction is one byte. The BUFFER parameter specifies the size of the data to be sent if a "0" is passed at the LENGTH parameter during

call. You cannot use the data type Bool or arrays of the Bool type for the BUFFER parameter.

Table 9-136 LENGTH parameter

LENGTH	Description
> 0	Transfers the configured number of bytes.
= 0	Transfers the entire contents of the send buffer addressed with BUFFER. If BUFFER points to a string, the entire string contents are transferred except for the bytes with the maximum and actual length.

Table 9-137 BUFFER parameter

BUFFER	Description
Elementary data type	When sending: The LENGTH value must include the byte size of this data type. Example: For a Word value, the LENGTH must be two. For a DWord value or Real value, the LENGTH must be four.
Structure	For optimized memory: The maximum permitted length of the BUFFER is 1024 Byte; otherwise 4 KB are permitted depending on the module. For sending: The LENGTH value can include a byte size smaller than the complete byte length of the structure; in this case, only the first LENGTH bytes of the structure from BUFFER are sent.
Array	For optimized memory: If the array data type is not equal to Byte, Word or DWord, the maximum permitted buffer length is 1024 Byte. Depending on the module, up to 4 KB can be transmitted if the memory is not optimized, independent of the data structure. For sending: The LENGTH value can include a byte size smaller than the complete byte length of the array, whereby, this byte size is a multiple of the byte size of the data element. Example: The LENGTH parameter of an array of the Word type must be a multiple of two and a multiple of four for an array of the Real type. If LENGTH is specified, the number of array elements is transmitted. If BUFFER includes an array with 15 DWord elements (a total of 60 Bytes), for example, and you specify LENGTH = 20, the first five DWord elements from the array are transmitted.
String	The LENGTH parameter includes the number or characters to be sent. Only the characters of the String are transmitted. The bytes with the maximum and actual length of the String are not sent.

Receive_P2P: Receiving data

Description

The Receive_P2P instruction (receive data using point-to-point communication) checks the frames received in the CM. If a frame is available, it is transmitted from the CM to the CPU. A receive error is indicated at the STATUS parameter.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1500	S7-300/400/WinAC		
PORT	IN	PORT (UInt)	Word	0	The CM port value can be found in the "Hardware ID" property of the device configuration (S7-1500) (input address; S7-300/400) once you have configured the CM. The symbolic port name is assigned in the "System constants" tab of the PLC tag table. In S7-300/400/WinAC systems, the input address assigned in HWCN must be assigned to the CM port.
BUFFER	IN	Variant	Any	0	This parameter points to the start address of the receive buffer. This buffer must be large enough to receive the maximum frame length. Note: • Boolean data or Boolean fields are not supported. • If the receive buffer is in the optimized memory area, the maximum permitted length of the received data is 1024 bytes. Exception: Arrays of Byte, Word or DWord are supported up to a length of 4096 bytes. • If the receive buffer is a String or WString, the received data is written to the content of the string and the current length of the string is set accordingly.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Receive_P2P instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
LENGTH	OUT	UInt	Word	0	Length of the frame received in bytes
NDR	OUT	Bool		FALSE	TRUE for one cycle if new data is available and the instruction has been completed without errors.
ERROR	OUT	Bool		FALSE	TRUE for one cycle once the instruction has been completed with an error.
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 5203))

The error code at the STATUS output can be evaluated if the status of ERROR is TRUE. The STATUS value provides the reason for terminating the receive operation in the CM.
This is usually a positive value which indicates that the receive operation has been successful and the frame criterion that has been detected.
If the STATUS value is negative (the most significant bit of the hexadecimal value is set), the receive operation was terminated due to an error condition, such as a parity, framing or overflow error.

Each communication module can buffer a module-specific number for frames. If several frames are available in the CM, the Receive_P2P instruction outputs the oldest available frame (FIFO).

Receive_Reset: Clear receive buffer

Description

The Receive_Reset instruction (reset receiver) clears the receive buffer in the CM.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the transmission of data to the CM upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	The CM port value can be found in the "Hardware ID" property of the device configuration (S7-1500) (input address; S7-300/400) once you have configured the CM. The symbolic port name is assigned in the "System constants" tab of the PLC tag table. In S7-300/400/WinAC systems, the input address assigned in HWCN must be assigned to the CM port.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Receive_Reset instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
DONE	OUT	Bool		FALSE	TRUE for one cycle means that the last request was completed without errors.
ERROR	OUT	Bool		FALSE	TRUE means that the last request was completed with errors. If this output is TRUE, the STATUS output contains the corresponding error codes.
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 5203))

Signal_Get: Read status

Description

The Signal_Get instruction (get RS232 signals) reads the current states of the RS232 accompanying signals and displays them at the corresponding instruction outputs.

Note

Restriction

This function can only be used with CMs RS232 BA and RS232 HF.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the transmission of data to the CM upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	The CM port value can be found in the "Hardware ID" property of the device configuration (S7-1500) (input address; S7-300/400) once you have configured the CM. The symbolic port name is assigned in the "System constants" tab of the PLC tag table. In S7-300/400/WinAC systems, the input address assigned in HWCN must be assigned to the CM port.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Signal_Get instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
NDR	OUT	Bool		FALSE	TRUE for one cycle if the RS232 accompanying signals have been read and the instruction has been completed without errors.
ERROR	OUT	Bool		FALSE	TRUE for one cycle once the instruction has been completed with an error
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 5203))
DTR	OUT	Bool		FALSE	Data device ready, module ready (output)
DSR	OUT	Bool		FALSE	Data device ready, communication station ready (input)
RTS	OUT	Bool		FALSE	Send request, module ready to send (output)
CTS	OUT	Bool		FALSE	Ready to send, communication station can receive data (input)
DCD	OUT	Bool		FALSE	Data carrier signal detected, signal level received
RING	OUT	Bool		FALSE	Call display, signaling incoming call

Signal_Set: Set accompanying signals

Description

The Signal_Set instruction (set RS232 signals) allows you to set the RS232 communication signals.

This function is only valid for RS232 CM.

Note

Restrictions

This function can only be used with CMs RS232 BA and RS232 HF.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the instruction upon a positive edge of this input.
PORT	IN	PORT (UInt)	Word	0	The CM port value can be found in the "Hardware ID" property of the device configuration (S7-1500) (input address; S7-300/400) once you have configured the CM. The symbolic port name is assigned in the "System constants" tab of the PLC tag table. In S7-300/400/WinAC systems, the input address assigned in HWCN must be assigned to the CM port.
SIGNAL	IN	Byte		0	Selection of the signal to be set (more than one possible): • 01H = RTS • 02H = DTR • 04H = DSR (for interface type DCE only)
RTS	IN	Bool		FALSE	Send request, module ready to send Set this value at the output (TRUE or FALSE), default value: FALSE
DTR	IN	Bool		FALSE	Data terminal ready, module ready Set this value at the output (TRUE or FALSE), default value: FALSE
DSR	IN	Bool		FALSE	Data terminal ready (for DCE interface type only), not used.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Signal_Set instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
DONE	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed without errors
ERROR	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed with errors
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 5203))

.

Get_Features: Get extended functions

Description

If supported by the module, you can use the Get_Features instruction (get extended functions) to get information on the ability of the module to support CRC and to generate diagnostic messages.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the instruction upon a positive edge of this input.
PORT	IN	PORT	Word	0	The CM port value can be found in the "Hardware ID" property of the device configuration (S7-1500) (input address; S7-300/400) once you have configured the CM. The symbolic port name is assigned in the "System constants" tab of the PLC tag table. In S7-300/400/WinAC systems, the input address assigned in HWCN must be assigned to the CM port.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Get_Features instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
NDR	OUT	Bool		FALSE	TRUE for one cycle if new data is available and the instruction has been completed without errors
MODBUS_CRC	OUT	Bool		FALSE	Modbus CRC support
DIAG_ALARM	OUT	Bool		FALSE	Generation of diagnostic messages
SUPPLY_VOLT	OUT	Bool		FALSE	Diagnostics for missing supply voltage L+ is available
ERROR	OUT	Bool		FALSE	TRUE for one cycle once the instruction has been completed with an error
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 5203))

Set_Features: Set extended functions

Description

If supported by the module, you can use the Set_Features instruction (select extended functions) to activate CRC support and the generation of diagnostic messages.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	The instruction to set extended functions is started upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	The CM port value can be found in the "Hardware ID" property of the device configuration (S7-1500) (input address; S7-300/400) once you have configured the CM. The symbolic port name is assigned in the "System constants" tab of the PLC tag table. In S7-300/400/WinAC systems, the input address assigned in HWCN must be assigned to the CM port.
EN_MODBUS_CRC	IN	Bool		FALSE	Activate Modbus CRC support
EN_DIAG_ALARM	IN	Bool		FALSE	Activate generation of diagnostic messages
EN_SUPPLY_VOLT	IN	Bool		FALSE	Enable diagnostics for missing supply voltage L+
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Set_Features instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
DONE	OUT	Bool		FALSE	TRUE for one execution once the last request is completed without errors
ERROR	OUT	Bool		FALSE	TRUE for one cycle once the instruction has been completed with an error
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 5203))

Error messages

Overview of error messages - PtP

The error messages are provided at the STATUS output of an instruction and can be evaluated there or processed in the user program.

Error code	Description	Remedy
16#0000	No error	-
RECEIVE status and error codes		
16#0094	frame end identified based on the "Receipt of fixed/maximum frame length"	-
16#0095	frame end identified based on "Message timeout"	-
16#0096	frame end identified based on expiration of the "Character delay time"	-
16#0097	The frame was aborted because the maximum response time was reached.	-

Error code	Description	Remedy
16#0098	frame end identified based on the fulfillment of the "Read message length from message" conditions	-
16#0099	frame end identified based on the receipt of the "End sequence"	-
SEND status and error codes		
16#7000	Block idle	-
16#7001	Initial call for a new frame: Data transmission initiated	-
16#7002	Interim call: Data transmission running	-
16#8085	Invalid length	Select a suitable frame length. The following are valid (depending on the module): 1-1024/2048/4096 (Byte)
16#8088	The specified length exceeds the range set in the receive buffer. Note: If the data type STRING has been specified at the BUFFER parameter, this error code also appears if the current string is shorter than the length specified at the LENGTH parameter.	Change the range in the receive buffer or select a frame length which corresponds to the range set in the receive buffer. The following are valid (depending on the module): 1-1024/2048/4096 (Byte)
16#8090	Configuration error: Odd number of bytes for WString	Select an even number of bytes.
RECEIVE status and error codes		
16#8088	The number of characters received exceeds the number specified at the BUFFER parameter.	Select a suitable frame length. The following are valid (depending on the module): 1-1024/2048/4096 (Byte)
16#8090	Configuration error: Odd number of bytes for WString	Select an even number of bytes.
Error message codes of the special functions		
16#818F	Incorrect parameter number setting (with USS only)	Select a suitable parameter number (PARAM). The following numbers are valid: 0-2047
16#8190	Incorrect setting of the CRC calculation	Select a suitable value for the CRC calculation. The following are valid: deactivated or activated. Check whether the module addressed supports CRC calculation.
16#8191	Incorrect setting of the diagnostic error interrupt	Select a suitable value for "Diagnostic error interrupt". The following are valid: Diagnostic error interrupt deactivated or diagnostic error interrupt activated. Check whether the module addressed supports the generation of diagnostic interrupts.
Error message codes of the "Port configuration"		
16#81A0	The module does not support this protocol.	Select a valid protocol for the module (PROTOCOL).
16#81A1	The module does not support this data transmission rate.	Select a valid data transmission rate for the module (BAUD).

Error code	Description	Remedy
16#81A2	The module does not support this parity setting.	Select a suitable value for "Parity" (PARITY). The following are valid: • None (1) • Even (2) • Odd (3) • Mark (4) • Space (5) • Any (6)
16#81A3	The module does not support this number of data bits.	Select a suitable value for "Number of data bits" (DATABITS). The following are valid: • 7 (2) • 8 (1)
16#81A4	The module does not support this number of stop bits.	Select a suitable value for "Number of stop bits" (STOPBITS). The following are valid: • 1 (1) • 2 (2)
16#81A5	The module does not support this type of data flow control.	Select a valid data flow control for the module (FLOWCTRL).
16#81A7	Invalid value for XON or XOFF	Select suitable values for XON (XONCHAR) and XOFF (XOFFCHAR). Valid range of values: 0...255
16#81AA	Invalid operating mode	Valid operating modes are: • Full duplex (RS232) (0) • Full duplex (RS422) four-wire mode (point-to-point) (1) / (CM PtP (ET 200SP)) • Full duplex (RS422) four-wire mode (multipoint master) (2)/ (CM PtP (ET 200SP)) • Full duplex (RS 422) four-wire mode (multipoint slave) (3) • Half duplex (RS485) two-wire mode. (4)
16#81AB	Invalid receive line initial state	Valid initial states are: • "No" default setting (0) • Signal R(A)=5 V, signal R(B)=0 V (break detection) (1): Can only be selected with: "Full duplex (RS422) four-wire operation (point-to-point connection)" and "Full duplex (RS422) four-wire operation (multipoint slave)". • Signal R(A)=0 V, signal R(B)=5 V (2): This default setting corresponds to the idle state (no active send operation).
16#81AC	Invalid value for "Break detection"	Select a suitable value for "Break detection". The following are valid: • Break detection deactivated (0) • Break detection activated (1).

Error code	Description	Remedy
16#81AF	The module does not support this protocol.	Select a valid protocol for the module.
Error codes of the "Send configuration"		
16#81B5	More than two end delimiters or end sequence > 5 characters	Select suitable values for "End delimiter" and "End sequence". The following are valid: • deactivated (0), • 1 (1) or 2 (2) end delimiters or • deactivated (0), • 1 (1) up to 5 (5) characters for the end sequence.
16#81B6	Send configuration rejected because the 3964(R) protocol was selected	Make sure that no send configuration is transmitted if the 3964(R) protocol is set.
Error codes of the "Receive configuration"		
16#81C0	Invalid start condition	Select a suitable start condition. The following are valid: • Send break before frame start • Send Idle Line.
16#81C1	Invalid end condition or no end condition selected	Select a suitable end condition (see Sending data with Freeport (Page 990)).
16#81C3	Invalid value for "Maximum message length"	Select a suitable value for "Maximum message length" (MAXLEN). Valid range of values (depending on the module): 1-1024/2048/4096 (Byte)
16#81C4	Invalid value for "Offset of the length specification in the message"	Select a suitable value for "Offset of the length specification in the message". Valid range of values (depending on the module): 1-1024/2048/4096 (Byte)
16#81C5	Invalid value for "Size of length field"	Select a suitable value for "Size of length field" (LENGTHSIZE). Valid range of values in bytes: • 1 (1) • 2 (2) • 4 (4)
16#81C6	Invalid value for "Number of characters not counted in length specification"	Select a suitable value for "Number of characters not counted in length specification" (LENGTHM). Valid range of values: 0 to 255 (bytes)
16#81C7	The total of "Offset in the message + size of length field + number of characters not counted" is greater than the maximum frame length	Select a suitable value for "Offset in message", "Size of length field" and "Number of characters not counted". Valid range of values: • Offset in the message (depending on the module): 0-1024/2048/4096 (Byte) • Size of length field: 1, 2, or 4 (bytes) • Number of characters not counted: 0 to 255 (bytes)

Error code	Description	Remedy
16#81C8	Invalid value for "Response timeout"	Select a suitable value for "Response timeout". Valid range of values: 1-65535 (ms)
16#81C9	Invalid value for "Character delay time"	Select a suitable value for "Character delay time". Valid range of values: 1 to 65535 (bit times)
16#81CB	frame end sequence is activated, but no character is activated for the check	Activate one or several characters for the check.
16#81CC	frame start sequence is activated, but no character is activated for the check	Activate one or several characters for the check.
16#81CD	Invalid value for "Prevent overwriting"	Select a suitable value for "Prevent overwriting". The following are valid: • Prevent overwriting is deactivated (0) or • Prevent overwriting is activated (1)
16#81CE	Invalid value for "Clear receive buffer on startup"	Select a suitable value for "Clear receive buffer on startup". The following are valid: • Clear receive buffer at startup is deactivated (0) • Clear receive buffer at startup is activated (1)
SEND status and error codes		
16#81D0	Receiving send request during runtime of a send command	Make sure that you do not receive an additional send request during runtime of a send command.
16#81D1	The waiting time for XON or CTS = ON has expired.	The communication partner has a fault, is too slow or is offline. Check the communication partner or change the parameters, if necessary.
16#81D2	"Hardware RTS always ON": Send job canceled due to change from DSR = ON to OFF	Check the communication partner. Make sure that DSR is ON for the entire duration of transmission.
16#81D3	Send buffer overflow / send frame too long	Select a shorter frame length. The following are valid (depending on the module): 1 to 1024/2048/4096 (bytes)
16#81D5	Transmission canceled due to parameter changes, detected wire break, or CPU in STOP	Check the parameter assignment, wire break, and CPU status.
16#81D6	Transmission canceled because end identifier was not received	Check the parameter assignment of the end characters and the frame of the communication partner.
16#81D7	Communication error between the user program and module	Check the communication (e.g., matching the sequence number).
16#81D8	Transmission attempt rejected because module is not configured	Configure the module.
Error codes of the receive configuration		
16#81E0	Frame aborted: Send buffer overflow / send frame too long	Increase the call rate for the receive function in the user program or configure communication with data flow control.
16#81E1	Frame aborted: Parity error	Check the connection line of the communication partners, or verify that both devices are configured for the same data transmission rate, parity and stop bit number.
16#81E2	Frame aborted: Character frame error	Check the settings for start bit, data bits, parity bit, data transmission rate, and stop bit(s).

Error code	Description	Remedy
16#81E3	Frame aborted: Character overflow error	Firmware error: Please contact Customer Support.
16#81E4	Frame aborted: The total length of "Offset in the message + size of the length field + number of characters not counted" is greater than the receive buffer	Select a suitable value for offset in message, size of the length field, and number of characters not counted.
16#81E5	Frame aborted: Break	Receive line to partner is interrupted. Reconnect or switch on partner.
16#81E6	Maximum number of "Buffered received frames" exceeded	In the user program call the instruction more often or configure a communication with data flow control or increase the number of buffered frames.
16#81E7	Synchronization error module and Receive_P2P	Make sure that different instances of the Receive_P2P do not access the same module.
16#81E8	Frame aborted: The character delay time has expired before the message end criterion was detected	Partner device faulty or too slow. Check this, if required, using an interface tester that is interconnected in the transmission line.
16#81E9	Modbus CRC error (only communication modules which support Modbus)	Checksum error of the Modbus frame. Check the communication partner.
16#81EA	Modbus frame too short (only communication modules which support Modbus)	Minimum length of Modbus frame not met. Check the communication partner.
16#81EB	Frame aborted: Maximum frame length reached	Select a shorter frame length at the communication partner. The following are valid (depending on the module): 1-1024/2048/4096 (Byte) Check the parameters for end of frame detection.
Error codes V24 accompanying signals		
16#81F0	The module does not support V24 accompanying signals	You have tried to set accompanying signals for a module that does not support V24 accompanying signals. Make sure that this is an RS232 module or that RS232 mode (ET 200SP) is set.
16#81F1	No operation of the V24 accompanying signals	The V24 accompanying signals cannot be operated manually if hardware data flow control is active.
Error codes of the receive configuration		
16#8201 [1)]	Receive_Conditions is a pointer to an invalid data type	Enter a pointer to one of the following data types: DB, BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TIME_OF_DAY, TIME, S5TIME, DATE_AND_TIME, STRING
16#8225	Receive_Conditionspoints to an optimized memory area greater than 1 kB or Receive_Conditions points to an optimized memory area and the receive length is greater than the area addressed by Receive_Conditions.	Enter a pointer to an area with a maximum length of: • Optimized memory area: 1 kByte • Non-optimized area: 4 kByte Note: If the pointer points to an optimized memory area, do not send more than 1 kByte.
16#8229 [1)]	Receive_Conditions is a pointer to BOOL with a number of bits not equal to n * 8	If you are using a pointer to BOOL, the number of bits must be a multiple of 8.
Error codes, general		

Error code	Description	Remedy
16#8280	Negative acknowledgment when reading module	You can find more detailed information on error causes in theRDREC.STATUS static parameters and in the description of the SFB RDREC. • Check the input at the PORT parameter • Set the COM_RST parameter prior to the initial call.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter You can find more detailed information on error causes in the WRREC.STATUS static parameters and in the description of the SFB WRREC.
16#8282	Module not available	Check the input at the PORT parameter and ensure that the module can be reached.
Error codes of the receive configuration		
16#82C1	Invalid value for "Buffered received frames".	Select a suitable value for "Buffered received frames". Valid range of values: 1-255
16#82C2	Receive configuration rejected because the 3964(R) protocol was selected	Make sure that no receive configuration is sent if the 3964(R) protocol is set.
16#8301 1)	Receive_Conditions is a pointer to an invalid data type	Select a valid data type. The following are valid: DB, BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TIME_OF_DAY, TIME, S5TIME, DATE_AND_TIME, STRING
16#8322	Range length error when reading a parameter	Check the input at the Receive_Conditions parameter
16#8324	Range error when reading a parameter	Check the input at the Receive_Conditions parameter
16#8328	Setting error when reading a parameter	Check the input at the Receive_Conditions parameter
SEND status and error codes		
16#8328 1)	BUFFER is a pointer to BOOL with a number of bits not equal to n * 8	If you are using a pointer to BOOL, the number of bits must be a multiple of 8.
Error codes of the receive configuration		
16#8332	Invalid data block at the Receive_Conditions parameter	Check the input at the Receive_Conditions parameter
16#833A	The designation of the data block at the Receive_Conditions parameter refers to a data block which is not loaded.	Check the input at the Receive_Conditions parameter
16#8351	Invalid data type	Check the input at the Receive_Conditions parameter
16#8352 1)	Receive_Conditions does not point to a data block	Check the pointer to Receive_Conditions
16#8353 1)	Receive_Conditions does not point to a structure of the type Receive_Conditions	Check the pointer to Receive_Conditions
Error codes 3964(R) protocol		
16#8380	Parameter assignment error: Invalid value for "Character delay time".	Select a suitable value for "Character delay time" (CharacterDelayTime). Valid range of values: 1 to 65535 (ms)
16#8381	Parameter assignment error: Invalid value for "Response timeout".	Select a suitable value for "Response timeout" (AcknDelayTime). Valid range of values: 1 to 65535 (ms)

Error code	Description	Remedy
16#8382	Parameter assignment error: Invalid value for "Priority".	Select a suitable value for "Priority" (Priority). The following are valid: • High (1) • Low (0)
16#8383	Parameter assignment error: Invalid value for "Block check"	Select a suitable value for "Block check" (BCC). The following are valid: • With block check (1) • Without block check (0)
16#8384	Parameter assignment error: Invalid value for "Connection attempts".	Select a suitable value for "Connection attempts" (BuildupAttempts). Valid range of values: 1-255
16#8385	Parameter assignment error: Invalid value for "Transmission attempts".	Select a suitable value for "Transmission attempts" (RepetitionAttempts). Valid range of values: 1-255
16#8386	Runtime error: Number of connection attempts exceeded	Check the interface cable and the transmission parameters. Also check whether the receive function is configured correctly at the partner device.
16#8387	Runtime error: Number of transmission attempts exceeded	Check the interface cable, the transmission parameters and the configuration of the communication partner.
16#8388	Runtime error: Error at the "Block check character" The internally calculated value of the block check character does not correspond to the block check character received by the partner at the connection end.	Check whether the connection is seriously disrupted; in this case you may also occasionally see error codes. Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#8389	Runtime error: Invalid character received while waiting for free receive buffer	The send request of the communication partner (STX, 02H) is only answered with DLE when the receive buffer is empty. No additional character may be received before (except STX again). Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#838A	Runtime error: Logical error during receiving. After DLE was received, a further random character (other than DLE or ETX) was received.	Check if partner DLE in frame header and in data string is always in duplicate or the connection is released with DLE ETX. Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#838B	Runtime error: Character delay time exceeded	Partner device too slow or faulty. Verify by using an interface test device that is switched into the transmission line, if necessary.
16#838C	Runtime error: Wait time for free receive buffer has started	In the user program call the instruction more often or configure a communication with data flow control.
16#838D	Runtime error: frame repetition does not start within 4 s after NAK	Check the communication partner. A received frame that is possibly corrupted must be repeated by the partner within 4 seconds.

Error code	Description	Remedy
16#838E	Runtime error: In idle mode, one or several characters (other than NAK or STX) were received.	Check for proper function of the partner device, possibly using an interface test device that is switched into the transmission line.
16#838F	Runtime error: Initialization conflict - Both partners have set high priority	Set the "Low" priority at one of the partners
16#8391	Parameter assignment error: 3964 configuration data rejected because Freeport is set	If the Freeport protocol is set, make sure that no 3964 parameter assignment data is sent.

[1] Only with instructions for S7-300/400 CPUs

Overview of error messages - Modbus

Error code	Description	Remedy
16#0000	No error	-
Configuration error of the interface - Modbus_Comm_Load		
16#8181	The module does not support this data transmission rate.	Select a valid data transmission rate for the module at the BAUD parameter.
16#8182	The module does not support this parity setting.	Select a suitable value for "Parity" at the PARITY parameter. The following are valid: • None (1) • Even (2) • Odd (3) • Mark (4) • Space (5) • Any (6)
16#8183	The module does not support this type of data flow control.	Select a valid data flow control for the module at the FLOW_CTRL parameter.
16#8184	Invalid value for "Response timeout".	Select a suitable value for "Response timeout" at the RESP_TO parameter. Valid range of values: 1 to 65535 (ms)
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the Send_Config.RDREC.STATUS or Receive_Config.RDREC.STATUS static parameters or RDREC.STATUS and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the Send_Config.WRREC.STATUS or Receive_Config.WRREC.STATUS static parameters or WRREC.STATUS and in the description of the SFB WRREC.
16#8282	Module not available	Check the input at the PORT parameter and ensure that the module can be reached.
Configuration error - Modbus_Slave		

Error code	Description	Remedy
16#8186	Invalid slave address	Select a suitable slave address at the MB_ADDR parameter. The following are valid: 1-247 at standard address area; 1-65535 at extended address area (0 is reserved for Broadcast)
16#8187	Invalid value at MB_HOLD_REG parameter	Select a suitable value for the hold register at the MB_HOLD_REG parameter.
16#8188	Invalid operating mode or broadcast (MB_ADDR = 0) and MODE parameter ≠ 1	Select the value 1 for MODE in Broadcast mode or select a different operating mode.
16#818C	The pointer to a MB_HOLD_REG area must be a data block or a bit memory address area.	Select a suitable value for the pointer to the MB_HOLD_REG area.
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.WRREC.STATUS or Receive_P2P.WRREC.STATUS, and in the description of the SFB WRREC.
16#8452 [1)]	MB_HOLD_REG is not a pointer to a DB or a bit memory area	Check the MB_HOLD_REG pointer
16#8453 [1)]	MB_HOLD_REG is not a pointer of type BOOL or WORD	Check the MB_HOLD_REG pointer
16#8454 [1)]	The area addressed by MB_HOLD_REG is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the MB_HOLD_REG pointer
16#8455 [1)]	MB_HOLD_REG points to a write-protected DB	Check the MB_HOLD_REG pointer
16#8456 [1)]	Error during instruction execution. The cause of the error is shown in the STATUS parameter.	Determine the value of the SFCSTATUS parameter. Check what this means in the description for SFC51, STATUS parameter.
Configuration error - Modbus_Master		
16#8180	Invalid value for MB_DB parameter	The value configured for MB_DB (instance data DB) at the Modbus_Comm_Load instruction is not valid. Check the interconnection of the Modbus_Comm_Load instruction and its error messages.
16#8186	Invalid station address	Select a suitable station address at the MB_ADDR parameter. The following are valid: 1-247 at standard address area; 1-65535 at extended address area (0 is reserved for Broadcast)
16#8188	Invalid operating mode or broadcast (MB_ADDR = 0) and MODE parameter ≠ 1	Select the value 1 for MODE in Broadcast mode or select a different operating mode.

Error code	Description	Remedy
16#8189	Invalid data address	Select a suitable value for the data address at the DATA_ADDR parameter. See description Modbus_Master (Page 5249) in the Info system
16#818A	Invalid length	Select a suitable data length at the DATA_LEN parameter. See description Modbus_Master (Page 5249) in the Info system
16#818B	Invalid value for DATA_PTR	Select a suitable value for the data pointer at the DATA_PTR parameter (M or DB address). See description Modbus_Master (Page 5249) in the Info system
16#818C	Interconnection error of the DATA_PTR parameter	Check the interconnection of the instruction.
16#818D	The area addressed by DATA_PTR is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the DATA_PTR pointer
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.WRREC.STATUS or Receive_P2P.WRREC.STATUS or Receive_Reset and in the description of the SFB WRREC.
Communication errors - Modbus_Master and Modbus_Slave		
16#80 D1	The waiting time for XON or CTS = ON has expired.	The communication partner has a fault, is too slow or is offline. Check the communication partner or change the parameters, if necessary.
16#80D2	"Hardware RTS always ON": Send job canceled due to change from DSR = ON to OFF	Check the communication partner. Make sure that DSR is ON for the entire duration of transmission.
16#80E0	Frame aborted: Send buffer overflow / send frame too long	In the user program call the instruction more often or configure a communication with data flow control.
16#80E1	Frame aborted: Parity error	Check the connection line of the communication partners, or verify that both devices are configured for the same data transmission rate, parity and stop bit number.
16#80E2	Frame aborted: Character frame error	Check the settings for start bit, data bits, parity bit, data transmission rate, and stop bit(s).
16#80E3	Frame aborted: Character overflow error	Check the number of data in the frame of the communication partner.
16#80E4	Frame aborted: Maximum frame length reached	Select a shorter frame length at the communication partner. The following are valid (depending on the module): 1 to 1024/2048/4096 (bytes)

Error code	Description	Remedy
Communication error - Modbus_Master		
16#80C8	The slave does not respond within the set time	Check the data transmission rate, parity and wiring of the slave.
16#8200	The interface is busy with an ongoing request.	Repeat the command later. Make sure that there are no commands still running before you start a new one.
Protocol error - Modbus_Slave (only communication modules that support Modbus)		
16#8380	CRC error	Checksum error of the Modbus frame. Check the communication partner.
16#8381	The function code is not supported or is not supported for broadcast.	Check the communication partner and make sure that a valid function code is sent.
16#8382	Invalid length information in the request frame	Select a suitable data length at the DATA_LEN parameter.
16#8383	Invalid data address in the request frame	Select a suitable value for the data address at the DATA_ADDR parameter.
16#8384	Invalid data value error in the request frame	Check the data value in the request frame of the Modbus master
16#8385	The diagnostic value is not supported by the Modbus slave (function code 08)	The Modbus slave only supports the diagnostic values 16#0000 and 16#000A.
Protocol error - Modbus_Master (only communication modules that support Modbus)		
16#8380	CRC error	Checksum error of the Modbus frame. Check the communication partner.
16#8381	Response frame from Modbus Slave with the following error message: The function code is not supported.	Check the communication partner and make sure that a valid function code is sent.
16#8382	Response frame from Modbus Slave with the following error message: Invalid length	Select a suitable data length.
16#8383	Response frame from Modbus Slave with the following error message: Invalid data address in the request frame	Select a suitable value for the data address at the DATA_ADDR parameter.
16#8384	Response frame from Modbus Slave with the following error message: Data value error	Check the request frame to the Modbus slave.
16#8385	Response frame from Modbus Slave with the following error message: The diagnostic value is not supported by the Modbus slave	The Modbus slave only supports the diagnostic values 16#0000 and 16#000A.
16#8386	The returned function code does not match the requested function code.	Check the response frame and the addressing of the slave.
16#8387	A slave that was not requested answers	Check the response frame of the slave. Check the address settings of the slave.
16#8388	Error in the response of the slave to a write request.	Check the response frame of the slave.
16#8828 [1)]	DATA_PTR points to a bit address that is not equal to n * 8	Check the DATA_PTR pointer
16#8852 [1)]	DATA_PTR is not a pointer to a DB or a bit memory area	Check the DATA_PTR pointer
16#8853 [1)]	DATA_PTR is not a pointer of type BOOL or WORD	Check the DATA_PTR pointer
16#8855 [1)]	DATA_PTR points to a write-protected DB	Check the DATA_PTR pointer
16#8856 [1)]	Error during call of SFC51	Call the Modbus_Master instruction again
Error - Modbus_Slave (only communication modules that support Modbus)		

Error code	Description	Remedy
16#8428 [1]	MB_HOLD_REG points to a bit address that is not equal to n * 8	Check the MB_HOLD_REG pointer
16#8452 [1]	MB_HOLD_REG is not a pointer to a DB or a bit memory area	Check the MB_HOLD_REG pointer
16#8453 [1]	MB_HOLD_REG is not a pointer of type BOOL or WORD	Check the MB_HOLD_REG pointer
16#8454 [1]	The area addressed by MB_HOLD_REG is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the MB_HOLD_REG pointer
16#8455 [1]	MB_HOLD_REG points to a write-protected DB	Check the MB_HOLD_REG pointer
16#8456 [1]	Error during call of SFC51	Call the Modbus_Slave instruction again

[1] Only with instructions for S7-300/400 CPUs

Overview of error messages - USS

Error code	Description	Remedy
16#0000	No error	-
16#8180	Length error in response of drive	Check the response frame of the drive.
16#8181	Data type error	Select a suitable data type. The following are valid: • Real • Word • Double word
16#8182	Data type error: "Double word" or "Real" may not be returned for the "Word" request.	Check the response frame of the drive.
16#8183	Data type error: "Word" may not be returned for the "Double word" or "Real" request.	Check the response frame of the drive.
16#8184	Checksum error in response of drive	Check the drive and the communication connection.
16#8185	Addressing error	Valid drive address range: 1 to 16
16#8186	Setpoint error	Valid setpoint range: -200 % to +200 %
16#8187	Incorrect drive number returned	Check the response frame of the drive.
16#8188	Invalid PZD length	Permitted PZD lengths: 2, 4, 6, 8 words
16#8189	The module does not support this data transmission rate.	Select a valid data transmission rate for the module.
16#818A	A different request for this drive is currently active.	Repeat the parameter read or write command later.
16#818B	The drive does not respond.	Check the drive.
16#818C	The drive responds with an error message to a parameter request.	Check the response frame of the drive. Check the parameter request.
16#818D	The drive responds with an access error message to a parameter request.	Check the response frame of the drive. Check the parameter request.
16#818E	The drive was not initialized.	Check the user program and make sure that the USS_Drive_Control instruction is called for this drive.

Error code	Description	Remedy
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Port_Config.RDREC.STATUS, Send_Config.RDREC.STATUS , Receive_Config.RDREC.STATUS , Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Port_Config.WRREC.STATUS, Send_Config.WRREC.STATUS , Receive_Config.WRREC.STATUS , Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB WRREC.

[1] Only with instructions for S7-300/400 CPUs

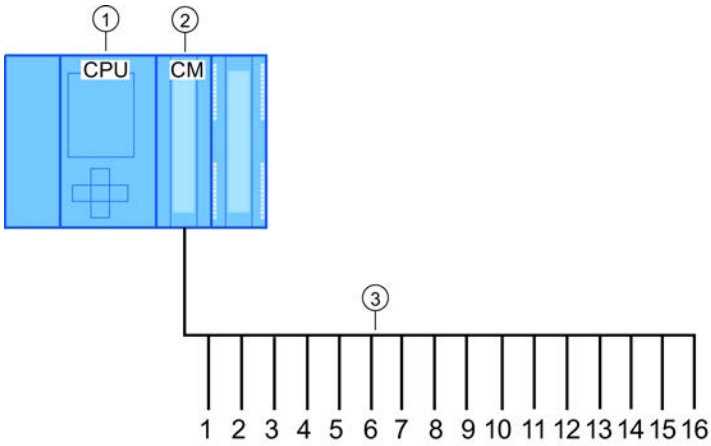
USS

Overview of USS communication

USS communication

The USS instructions control the operation of drives which support the protocol of the universal serial interface (USS). You can communicate with several drives by means of RS485 connections and the USS instructions with the PtP communication modules. Each RS485 port can operate up to 16 drives.

The USS protocol uses a master-slave network for communication via a serial bus. The master uses an address parameter to send data to a selected slave. A slave cannot send without having first received a send request. Communication between individual slaves is not possible. USS communication takes place in half-duplex mode. The figure below shows a network diagram for a sample application with 16 drives.



① CPU
② CM
③ USS drives in a USS network

Figure 9-1 Wiring example with S7-1500 communication module

Note

Communicating with a drive via RS232

You can basically also use CM PtP RS232 BA and CM PtP RS232 HF for communication with a drive. However, only **one** drive can be connected to an RS232 port.

Communicating with a drive via RS422

You can basically also use the RS422 interface of the CM PtP RS422/485 BA and CM PtP RS422/485 HF for communication with a drive. However, only **one** drive can be connected to an RS422 port.

USS instructions in your program

- USS_Port_Scan: The instruction USS_Port_Scan allows you to communicate via a communication module with up to 16 drives using a USS network (must be called cyclically). There is only one USS_Port_Scan instruction per PtP communication port in the program, and it controls transmission to all drives.
- USS_Drive_Control: The instruction USS_Drive_Control allows you to prepare the send data from USS_Port_Scan for a drive and display its receive data. USS_Drive_Control configures the data to be sent and evaluates the data received in a previous request from USS_Port_Scan.
- USS_Read_Param: The instruction USS_Read_Param allows you to read parameters from a drive.
- USS_Write_Param: The instruction USS_Write_Param allows you to change parameters in a drive.

Requirements for using the USS protocol

The four USS instructions use 2 FBs and 2 FCs to support the USS protocol. For each USS network, one instance data block (DB) is used for USS_Port_Scan and one instance data block for all calls of USS_Drive_Conrol.

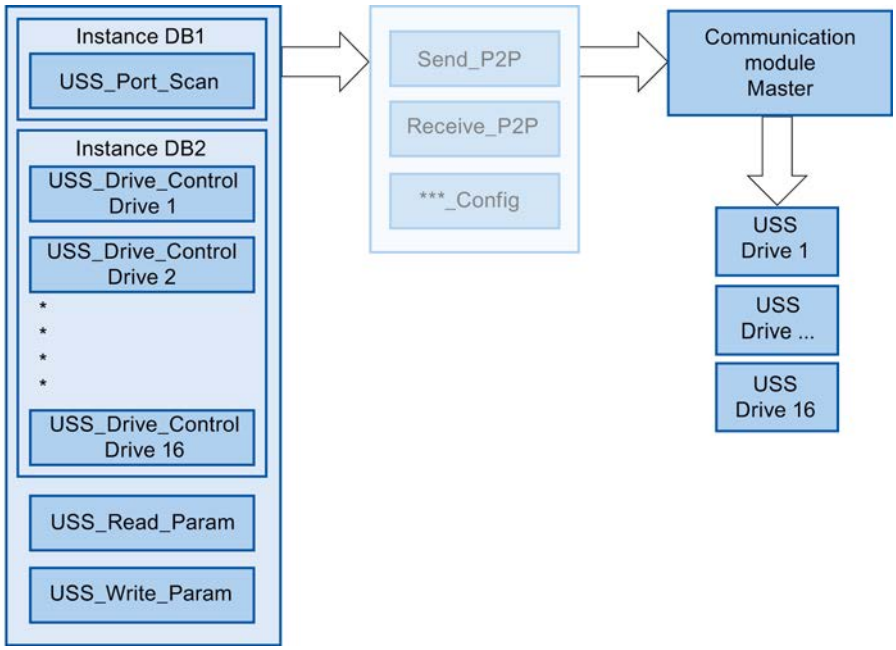


Figure 9-2 USS program sequence

All drives (max. 16) that are connected to one RS485 port are part of the same USS network. All drives that are connected to another RS485 port are part of a different USS network. Each USS network is managed using a unique instance data block for all USS_Drive_Control instructions and one further instance data block for the USS_Port_Scan instruction. All instructions that are part of a USS network must share this instance data block for USS_Drive_Conrol. The USS_Port_Scan, USS_Read_Param and USS_Write_Param instructions have the USS_DB parameter for this function. This parameter must be connected to the (static) USS_DB parameter of the instance DB of the USS_Drive_Control instruction.

- The instructions USS_Drive_Control and USS_Port_Scan are function blocks (FB). If you add the USS_Drive_Control or USS_Port_Scan instruction to the programming editor, you are prompted to assign a DB for this FB in the "Call options" dialog. If it is the first USS_Drive_Control instruction in this program for this USS network, you can apply the DB standard assignment (or change the name, if necessary) and the new DB is created for you. If, however, it is not the first USS_Drive_Control instruction for this drive, you need to select the DB which has already been assigned to this USS network from the drop-down list in the "Call options" dialog.
- The USS_Port_Scan and USS_Read_Param instructions are functions (FCs). No DB is assigned when you add these FCs in the editor. If you add these FCs or the USS_Port_Scan instruction in the editor, you need to assign the USS_DB parameter of the corresponding USS_Drive_Control instance DB to the USS_DB input of these instructions. Double-click the parameter field and click the symbol to display the available DBs. Enter a period "." and select the USS_DB parameter from the drop-down list.

- The USS_Port_Scan function controls communication between the CPU and the drives via the point-to-point RS485 communication port. A communication with the drive is processed every time you call this function. Your program must call this function fast enough so that the drives do not signal a timeout. To ensure a constant time response for frame communication, you should call this instruction in a cyclic interrupt OB.
- The USS_Drive_Control instruction gives your program access to a specified drive in the USS network. Its inputs and outputs correspond to the states and operating functions of the drive. If there are 16 drives in your network, you must call USS_Drive_Control at least 16 times in your program, i.e. once for each drive.
  You should only call the USS_Drive_Control instruction from a cyclic OB.
- The operating parameters of the drive are read and written with the USS_Read_Param and USS_Write_Param functions. These parameters control the internal operation of the drive. See the drive manual for a definition of these parameters. Your program may include any number of these functions, but only one read or write request may be active for a drive at any particular time. You may only call the USS_Read_Param and USS_Write_Param functions from the cycle OB of a main program.

NOTICE
USS instruction calls Only ever call USS_Drive_Control, USS_Read_Param or USS_Write_Param from a cycle OB of the main program. The USS_Port_Scan instruction function can be called from any OB, but it is usually called from a cyclic interrupt OB. Do not use the USS_Drive_Control, USS_Read_Param or USS_Write_Param instruction in an OB with a higher priority than the corresponding USS_Port_Scan instruction. For example, do not add USS_Port_Scan to the main program or USS_Read_Param to a cyclic interrupt OB. You may encounter unexpected errors if the execution of USS_Port_Scan is interrupted by another instruction.

Calculating time for communication with the drive

Communication with the drive takes place asynchronously to the cycle of the CPU. The CPU usually runs through several cycles before communication with a drive is complete.

To ensure that the watchdog set for the drive is not triggered, the send frames must be sent to the drive within the watchdog time. You must allow for the number of any retries which may be needed to complete the transaction if communication errors occur. By default, up to 2 retries are made for each transaction with the USS protocol.
The time interval between two send frames is calculated as follows:

(N * 5 * cycle time + 2 * frame runtime) * number of retries + (received frame timeout) * (number of retries - 1)

N	Number of drives in this network
Factor 5	5 cycles are needed for sending and receiving frames.
Cycle time	Max. cycle time of the cyclic interrupt OB in which the USS_Port_Scan instruction is called.
2 * frame time	Once each for sending and receiving

Frame runtime	Frame time = (number of characters per frame) * (11 Bit per character) / (data transmission rate in Bit/s)
Number of retries	3

The following times apply for "Received frame timeout" (ms)

Bit/s	115200	57600	38400	19200	9600	4800	2400	1200
Receive_Conditions.END.RCVTIME	25	29	33	56	72	100	100	100
Receive_Conditions.END.MSGTIME	25	29	33	56	72	124	240	460

Example:
5 drives
data transmission rate = 9600 Bit/s
28 characters per frame
cycle time = 0.020 sec
time interval = (5 * 5 * 0.02 sec + 2 * 28 * 11 / 1200 sec) * 3 = 1.69 sec
The time monitoring of the drive must be set to approx. 2 seconds in this case.

USS_Port_Scan: Communication by means of a USS network

Description

The USS_Port_Scan instruction processes the communication by means of a USS network. STEP 7 automatically creates the instance DB when you add the instruction.

Parameters

Parameter	Declaration	Data type		Standard	Description
		S7-1500	S7-300/400/ WinAC		
PORT	IN	Port	Word	0	The CM port value can be found in the "Hardware ID" property of the device configuration (S7-1500) (input address; S7-300/400) once you have configured the CM. The symbolic port name is assigned in the "System constants" tab of the PLC tag table. In S7-300/400/WinAC systems, the input address assigned in HWCN must be assigned to the CM port.
BAUD	IN	DInt		9600	Data transmission rate for USS communication The following are valid: • 1200 Bit/s • 2400 Bit/s • 4800 Bit/s • 9600 Bit/s • 19200 Bit/s • 38400 Bit/s • 57600 Bit/s • 115200 Bit/s
USS_DB	INOUT	USS_BASE		–	The USS_DB parameter must be connected to the (static) USS_DB parameter of the instance DB which is generated and initialized when you add a USS_Drive_Control instruction to your program.
COM_RST	INOUT	---	Bool	FALSE	Initialization of the USS_Port_Scan instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
ERROR	OUT	Bool		FALSE	If TRUE, this output indicates that an error has occurred and that the STATUS output is valid.
STATUS	OUT	Word		0	Error code (see Error messages (Page 5231)).

There is only one USS_Port_Scan instruction per PtP communication port in the program and each call of this instruction controls a transmission to or from all drives in this network. All USS functions assigned to one USS network and one PtP communication port must use the same instance DB.

Your program must execute the USS_Port_Scan instruction often enough to prevent a timeout in the drive (see Requirements for using the USS protocol (Page 5218) "Calculating time for communication with the drive").

The USS_Port_Scan instruction is usually called from a cyclic interrupt OB to prevent drive timeouts and to have the last USS data updates available for calls of USS_Drive_Control .

USS_Port_Scan data block tags

The table below shows the public static tags in the instance DB of USS_Port_Scan that you can use in your program.

Table 9-138 Static tags in the instance DB

Tag	Data type	Standard	Description
MODE	USInt	4	Operating mode Valid operating modes are: • 0 = Full duplex (RS232) • 1 = Full duplex (RS422) four-wire operation (point-to-point) • 2 = Full duplex (RS 422) four-wire mode (multipoint master; CM PtP (ET 200SP)) • 3 = Full duplex (RS 422) four-wire mode (multipoint slave; CM PtP (ET 200SP)) • 4 = Half duplex (RS485) two-wire mode
LINE_PRE	USInt	2	Receive line initial state Valid initial states are: • 0 = "no" initial state • 1 = signal R(A)=5 V, signal R(B)=0 V (break detection): Break detection is possible with this initial state. Can only be selected with: "Full duplex (RS422) four-wire operation (point-to-point connection)" and "Full duplex (RS422) four-wire operation (multipoint slave)". • 2 = signal R(A)=0 V, signal R(B)=5 V: This default setting corresponds to the idle state (no active send operation). No break detection is possible with this initial state.
BRK_DET	USInt	0	Activate diagnostics interrupt: • 0 - not activated • 1 - activated
RETRIES_MAX	SInt/Byte	2	Number of retries when communication errors occur. You can use this parameter to set the number of send retries for a request frame if the response frame is not received within the set time.
EN_DIAG_ALARM	Bool	0	Activate diagnostics interrupt: • 0 - not activated • 1 - activated
EN_SUPPLY_VOLT	Bool	0	Enable diagnostics for missing supply voltage L+ • 0 - not activated • 1 - activated

USS_Drive_Control: Preparing and displaying data for the drive

Description

The instruction USS_Drive_Control prepares send data for the drive and evaluates the drive response data. You need to use a separate instance of the instruction for each drive, and all USS functions assigned to one USS network and one PtP communication port must use the

same instance DB. You must enter the DB name when you add the first USS_Drive_Control instruction. You then refer to this DB that was created when the first instruction was added.

STEP 7 automatically creates the DB when you add the instruction.

Parameters

Parameters	Declaration	Data type		Standard	Description
		S7-1500	S7-300/400/WinAC		
RUN	IN	Bool		FALSE	Start bit of the drive: If this parameter is TRUE, this input allows you to operate the drive with the preset speed. If RUN changes to FALSE during operation of the drive, the motor coasts to a standstill. This behavior differs from disconnection of the power supply (OFF2) and braking of the motor (OFF3).
OFF2	IN	Bool		FALSE	"Coast to standstill" bit: If this parameter is FALSE, this bit causes the drive to coast to a standstill without braking.
OFF3	IN	Bool		FALSE	Fast stop bit: If this parameter is FALSE, this bit causes a fast stop by braking the drive.
F_ACK	IN	Bool		FALSE	Error acknowledgment bit: This bit resets the error bit of a drive. The bit is set after clearing the error and the drive detects this way that the previous error no longer has to be reported.
DIR	IN	Bool		FALSE	Direction control of the drive: This bit is set if the drive is to run forward (if SPEED_SP is positive; see table "Interaction of SPEED_SP and DIR parameters").
DRIVE	IN	USInt	Byte	1	Address of the drive: This input is the address of the USS drive. The valid range is between drive 1 and drive 16.
PZD_LEN	IN	USInt	Byte	2	Word length: This is the number of PZD data words. Valid values are 2, 4, 6 or 8 words.
SPEED_SP	IN	Real		0.0	Speed setpoint: This is the speed of the drive as a percentage of the configured frequency. A positive value means that the drive runs forward (if DIR is true). The valid range is 200.00 to -200.00.
CTRL3	IN	Word		0	Control word 3: Value that is written to a user-defined parameter of the drive. You have to configure it in the drive (optional parameter).
CTRL4	IN	Word		0	Control word 4: Value that is written to a user-defined parameter of the drive. You have to configure it in the drive (optional parameter).
CTRL5	IN	Word		0	Control word 5: Value that is written to a user-defined parameter of the drive. You have to configure it in the drive (optional parameter).
CTRL6	IN	Word		0	Control word 6: Value that is written to a user-defined parameter of the drive. You have to configure it in the drive (optional parameter).
CTRL7	IN	Word		0	Control word 7: Value that is written to a user-defined parameter of the drive. You have to configure it in the drive (optional parameter).

Parameters	Declaration	Data type		Standard	Description
		S7-1500	S7-300/400/ WinAC		
CTRL8	IN	Word		0	Control word 8: Value that is written to a user-defined parameter of the drive. You have to configure it in the drive (optional parameter).
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the USS_Drive_Control instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
NDR	OUT	Bool		FALSE	New data available: If this parameter is TRUE, the bit signals that data of a new communication request is available at the output.
ERROR	OUT	Bool		FALSE	Error occurred: If TRUE, this indicates that an error has occurred and that the STATUS output is valid. All other outputs are set to zero in the event of an error. Communication errors are only signaled at the ERROR and STATUS outputs of the USS_Port_Scan instruction.
STATUS	OUT	Word		0	Error code (see Error messages (Page 5231)).
RUN_EN	OUT	Bool		FALSE	Operation enabled: This bit signals if the drive is running.
D_DIR	OUT	Bool		FALSE	Drive direction: This bit signals if the drive is running forward. • FALSE – forward • TRUE – backward
INHIBIT	OUT	Bool		FALSE	Drive blocked: This bit signals the status of the block bit for the drive. • FALSE – not blocked • TRUE – blocked
FAULT	OUT	Bool		FALSE	Drive error: This bit signals that an error occurred in the drive. You must remedy the fault and set the F_ACK bit to clear this bit.
SPEED	OUT	Real		0.0	Actual value drive speed (scaled value of STATUS 2 of the drive): This is the speed of the drive as a percentage of the configured speed.
STATUS1	OUT	Word		0	STATUS 1 of the drive This value includes fixed status bits of a drive.
STATUS3	OUT	Word		0	STATUS 3 of the drive This value includes a user-definable status word of the drive.
STATUS4	OUT	Word		0	STATUS 4 of the drive This value includes a user-definable status word of the drive.
STATUS5	OUT	Word		0	STATUS 5 of the drive This value includes a user-definable status word of the drive.
STATUS6	OUT	Word		0	STATUS 6 of the drive This value includes a user-definable status word of the drive.
STATUS7	OUT	Word		0	STATUS 7 of the drive This value includes a user-definable status word of the drive.
STATUS8	OUT	Word		0	STATUS 8 of the drive This value includes a user-definable status word of the drive.

When the initial execution of USS_Drive_Control takes place, the drive specified by the USS address (DRIVE parameter) is initialized in the instance DB. After initialization, subsequent USS_Port_Scan instructions can start communication with the drive from this drive number.

If you change the drive number, you must first place the CPU in STOP and then back in RUN to initialize the instance DB. The input parameters are configured in the USS send buffer and any outputs are read from a "previous" valid response buffer. USS_Drive_Control only configures the data to be sent and evaluates data received in a previous request.

You can control the drive's direction of rotation by using the D_IR input (Bool) or the sign (positive or negative) at the SPEED_SP input (Real). The table below explains how these inputs work together to determine the direction of rotation of the drive, provided the motor rotates forward.

Table 9-139 Interaction of SPEED_SP and DIR parameters

SPEED_SP	DIR	Direction of rotation of the drive
Value > 0	0	Backward
Value > 0	1	Forward
Value < 0	0	Forward
Value < 0	1	Backward

USS_Read_Param: Read data from drive

Description

The USS_Read_Param instruction reads a parameter from a drive. All USS functions assigned to one USS network and one PtP communication port must use the instance data block of the USS_Drive_Control instruction. USS_Read_Param must be called from a cycle OB of the main program.

Parameters

Parameters	Declaration	Data type		Standard	Description
		S7-1500	S7-300/400/ WinAC		
REQ	IN	Bool		–	A positive edge at REQ creates a new read request.
DRIVE	IN	USInt	Byte	–	Address of the drive: DRIVE is the address of the USS drive. The valid range is between drive 1 and drive 16.
PARAM	IN	UInt		–	Parameter number: PARAM specifies the drive parameter to write. The range for this parameter is located between 0 and 2047; for some drives the most significant byte of the INDEX parameter can access parameter values greater than 2047. Additional information on access to an extended range is available in your drive manual.

Parameters	Declaration	Data type		Standard	Description
		S7-1500	S7-300/400/ WinAC		
INDEX	IN	UInt		–	Parameter index: INDEX specifies the drive parameter index to which to write. It is a 16-bit value in which the least significant bit is the actual index value with a range of (0 to 255). The drive can also use the most significant byte, which is drive-specific. Additional information is available in your drive manual.
USS_DB	INOUT	USS_BASE		–	The USS_DB parameter must be connected to the (static) USS_DB parameter of the instance DB which is generated and initialized when you add a USS_Drive_Control instruction to your program.
DONE[1]	OUT	Bool		FALSE	If this parameter is TRUE, the previously requested value of the read parameter is available at the VALUE output. This bit is set when the USS_Drive_Control instruction recognizes the read response of the drive. This bit is reset the next time USS_Read_Param is called.
ERROR	OUT	Bool		FALSE	ERROR = TRUE: An error has occurred and the STATUS output is valid. All other outputs are set to zero in the event of an error. Communication errors are only signaled at the ERROR and STATUS outputs of the USS_Port_Scan instruction.
STATUS	OUT	Word		0	Error code (see Error messages (Page 5231)).
VALUE	OUT	Variant (Word, Int, UInt, DWord, DInt, UDInt, Real)	Any (Word, Int, UInt, DWord, DInt, UDInt, Real)	–	This is the value of the parameter that was read and is only valid if the DONE bit is true.

[1] The DONE bit indicates that valid data was read out of the referenced motor drive and transmitted to the CPU. It does not indicate that the instruction is cabable of immediately read an additional parameter. An empty read request must be sent to the motor drive and acknowledged by the instruction before the parameter channel is freed up for use by the respective drive. The immediate call of USS_Read_Param or USS_Write_Param for the specific motor drive results in error 0x818A.

USS_Write_Param: Change data in drive

Note

For EEPROM write instructions (EEPROM in a USS drive):

Keep the number of EEPROM write operations to a minimum in order to maximize the EEPROM service life.

Description

The USS_Write_Param instruction changes a parameter in the drive. All USS functions assigned to one USS network and one PtP communication port must use the instance data block of the USS_Drive_Control.

USS_Write_Param must be called from the cycle OB of a main program.

Parameters

Table 9-140 Data types for the parameters

Parameters	Declaration	Data type		Standard	Description
		S7-1500	S7-300/400/ WinAC		
REQ	IN	Bool		–	A positive edge at REQ creates a new write request.
DRIVE	IN	USInt	Byte	–	Address of the drive: DRIVE is the address of the USS drive. The valid range is between drive 1 and drive 16.
PARAM	IN	UInt		–	Parameter number: PARAM specifies the drive parameter to write. The range for this parameter is located between 0 and 2047; for some drives the most significant byte of the INDEX parameter can access parameter values greater than 2047. Additional information on access to an extended range is available in your drive manual.
INDEX	IN	UInt		–	Parameter index: INDEX specifies the drive parameter index to which to write. It is a 16-bit value in which the least significant bit is the actual index value with a range of (0 to 255). The drive can also use the most significant byte, which is drive-specific. Additional information is available in your drive manual.
EEPROM	IN	Bool		–	Save in EEPROM of the drive: If TRUE, the transaction of a parameter for writing to the drive is saved in the EEPROM of the drive. If FALSE, the written value is saved only temporarily and is lost the next time you switch on the drive.
VALUE	IN	Variant (Word, Int, UInt, DWord, DInt, UDInt, Real)	Any (Word, Int, UInt, DWord, DInt, UDInt, Real)	–	Value of the parameter in which you want to write. It must be valid with a positive edge of REQ.
USS_DB	INOUT	USS_BASE		–	The USS_DB parameter must be connected to the (static) USS_DB parameter of the instance DB which is generated and initialized when you add a USS_Drive_Control instruction to your program.
DONE[1]	OUT	Bool		FALSE	If TRUE, the VALUE input is written to the drive. This bit is set when the USS_Drive_Control instruction recognizes the write response of the drive. This bit is reset the next time USS_Write_Paramis called.

Parameters	Declaration	Data type		Standard	Description
		S7-1500	S7-300/400/ WinAC		
ERROR	OUT	Bool		FALSE	If TRUE, an error has occurred and the STATUS output is valid. All other outputs are set to zero in the event of an error. Communication errors are only signaled at the ERROR and STATUS outputs of the USS_Port_Scan instruction.
STATUS	OUT	Word		0	Error code (see Error messages (Page 5231)).

[1] The DONE bit indicates that valid data was read out of the referenced motor drive and transmitted to the CPU. It does not indicate that the USS library is able to read out an additional parameter immediately. An empty write request must be sent to the motor drive and acknowledged by the instruction before the parameter channel is freed up for use by the respective drive. The immediate call of USS_Read_Param or USS_Write_Param FC for the specific motor drive results in error 0x818A.

General information on drive setup

Requirements for the drive setup

- You need to configure the use of 4 PIV words (**P**arameter**I**D**V**alue) for the drives.
- The drives can be configured for 2, 4, 6 or 8 PZD words (**P**ro**c**ess **d**ata area).
- The number of PZD words in the drive must correspond to the PZD_LEN input of the USS_Drive_Control instruction of the drive.
- Make sure that the data transmission rate of all drives corresponds to the BAUD input of the USS_Port_Scan instruction.
- Make sure that the drive is set up for USS communication.
- Make sure that it is specified in the drive that the frequency setpoint is provided by the USS interface.
- Make sure that the drive address is specified (areas: 1-16).
  This address must correspond to the DRIVE input at the USS_Drive_Control block of the drive.
- Make sure that the RS485 network is terminated correctly.

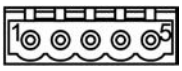
Connection and setup of SINAMICS V20 drive

An application example for operation of a SINAMICS V20 at an S7-1200 is available on the Internet. (http://support.automation.siemens.com/WW/view/en/63696870)

Connecting the SINAMICS V20 drive

Connection example of a SIEMENS G120(C) drive to a USS network. Connection examples for other drives are available in the manual of the respective drive.

The connection of a SINAMICS G120(C) drive to the USS network takes place via plug-in connection. The connection is short-circuit proof and isolated.



1 0 V reference potential
2 RS485N, receiving and sending (-)
3 RS485N, receiving and sending (+)
4 Cable shield
5 Not used

Figure 9-3 USS connection

NOTICE
Different reference voltages If you connect devices that do not have the same reference voltage, you may create unwanted currents in the connection cable. These unwanted currents may cause communication errors or damages in the devices. Make sure that all devices you connect with a communication cable either have the same reference conductor in the circuit or are electrically disconnected to prevent the occurrence of unwanted currents. Make sure that the shield is connected to the ground or pin 1 of the bus connector on the drive. Make sure that wiring terminal 2 (GND) of the G120(C) is connected to the ground.

If the RS485 master (e.g., S7-1200 CPU with CM1241 communication module) is connected by means of a PROFIBUS connector, wire the bus cables as follows:

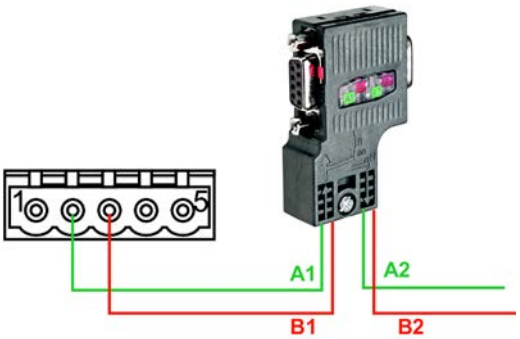


Figure 9-4 Connection of communication module

If the RS485 master is a terminating station in the network or a point-to-point connection, you must use terminals A1 and B1 (not A2 and B2) of the PROFIBUS connector, as these terminals provide termination settings (for example, with the DP plug connector 6ES7972-0BB52-0XA0).

If the G120(C) is configured as terminating station in the network, you must set the switch for the bus terminating resistor to "ON".

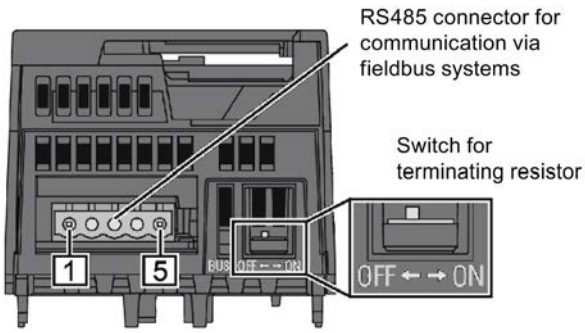


Figure 9-5 Connection of terminating stations

Setup of a G120(C) drive

Before you connect a drive to the S7-1500 or ET 200SP, make sure that the drive has the following system parameters.

Step	Instruction	Operating Instructions	
		G120 [1]	G120C [2]
1	Perform a basic commissioning of the drive with the Operator Panel BOP-2. The converter offers different defaults (macros) for its inputs and outputs and the fieldbus interface. In the ninth step of the basic commissioning (MAC PAR p15), select macro 21 for USS communication. This determines the default for the following parameters: • Data transmission rate (p2020): 38400 bps • Number of PZD (p2022): 2 • Number of PIV (p2023): variable Note: You can also conduct basic commissioning with the STARTER commissioning software or with SINAMICS Startdrive.	Chapter 4.4.3	Chapter 6.4.1
2	Specify the USS address of the converter with the address switch on the control unit of the G120 or on the G120(C). • Valid address range: 1 … 30 Note: Yon can also set the USS address with the parameter p2021 or with STARTER or SINAMICS Startdrive.	Chapter 6.2.2.1	Chapter 8.4.2.1
	With the following steps you are accessing the parameters directly with the BOP-2 by entering their numbers and modifying their values.	Chapter 4.4.2	Chapter 6.4.2

Step	Instruction	Operating Instructions	
		G120 [1]	G120C [2]
3	Adapt the following communication-related converter parameters of your application: • Data transmission rate (p2020), of ≠ 38400 bps (Make sure that setting is identical with the BAUD parameter of the USS_Port_Scan communication instruction.) • Number of PZD (p2022), if ≠ 2 (Make sure that the setting is identical with the PZD_LEN parameter of the USS_Drive_Control communication instruction.) • Number of PIV (p2023) = 4 (Change the value set to "variable" (127) with macro 21 by default to "4" (required by instructions USS_Read_Param and USS_Write_Param.) • Fieldbus SS monitoring time [ms] (p2040)	Chapter 6.2.2.2	Chapter 8.4.2.1
4	Specify the source for the speed setpoint. • n_set Eval (p1000[0]) = 6 (The speed setpoint is provided by the USS bus.)		
5	Set the reference value for speed and frequency. • n_reference f_reference (p2000) = (6,00 min^{-1} to 210000,00 min^{-1}) (all relative speeds or frequencies refer to this reference value. The reference value corresponds to 100% or 4000_{hex} (word) or $4000\ 0000_{hex}$ (double word). The following applies: $f_{reference_value}$ (in Hz) = $n_{reference_value}$ (in ((min^{-1}) / 60) x number of pole pairs))		
6	Transfer the parameters to the non-volatile memory. • Save par (p0971) = 1		

1) G120 (http://support.automation.siemens.com/WW/view/en/62089662)

2) G120(C) (http://support.automation.siemens.com/WW/view/en/61462568)

Error messages

Overview of error messages - PtP

The error messages are provided at the STATUS output of an instruction and can be evaluated there or processed in the user program.

Error code	Description	Remedy
16#0000	No error	-
RECEIVE status and error codes		
16#0094	frame end identified based on the "Receipt of fixed/ maximum frame length"	-
16#0095	frame end identified based on "Message timeout"	-
16#0096	frame end identified based on expiration of the "Character delay time"	-
16#0097	The frame was aborted because the maximum response time was reached.	-

Error code	Description	Remedy
16#0098	frame end identified based on the fulfillment of the "Read message length from message" conditions	-
16#0099	frame end identified based on the receipt of the "End sequence"	-
SEND status and error codes		
16#7000	Block idle	-
16#7001	Initial call for a new frame: Data transmission initiated	-
16#7002	Interim call: Data transmission running	-
16#8085	Invalid length	Select a suitable frame length. The following are valid (depending on the module): 1-1024/2048/4096 (Byte)
16#8088	The specified length exceeds the range set in the receive buffer. Note: If the data type STRING has been specified at the BUFFER parameter, this error code also appears if the current string is shorter than the length specified at the LENGTH parameter.	Change the range in the receive buffer or select a frame length which corresponds to the range set in the receive buffer. The following are valid (depending on the module): 1-1024/2048/4096 (Byte)
16#8090	Configuration error: Odd number of bytes for WString	Select an even number of bytes.
RECEIVE status and error codes		
16#8088	The number of characters received exceeds the number specified at the BUFFER parameter.	Select a suitable frame length. The following are valid (depending on the module): 1-1024/2048/4096 (Byte)
16#8090	Configuration error: Odd number of bytes for WString	Select an even number of bytes.
Error message codes of the special functions		
16#818F	Incorrect parameter number setting (with USS only)	Select a suitable parameter number (PARAM). The following numbers are valid: 0-2047
16#8190	Incorrect setting of the CRC calculation	Select a suitable value for the CRC calculation. The following are valid: deactivated or activated. Check whether the module addressed supports CRC calculation.
16#8191	Incorrect setting of the diagnostic error interrupt	Select a suitable value for "Diagnostic error interrupt". The following are valid: Diagnostic error interrupt deactivated or diagnostic error interrupt activated. Check whether the module addressed supports the generation of diagnostic interrupts.
Error message codes of the "Port configuration"		
16#81A0	The module does not support this protocol.	Select a valid protocol for the module (PROTOCOL).
16#81A1	The module does not support this data transmission rate.	Select a valid data transmission rate for the module (BAUD).

Error code	Description	Remedy
16#81A2	The module does not support this parity setting.	Select a suitable value for "Parity" (PARITY). The following are valid: • None (1) • Even (2) • Odd (3) • Mark (4) • Space (5) • Any (6)
16#81A3	The module does not support this number of data bits.	Select a suitable value for "Number of data bits" (DATABITS). The following are valid: • 7 (2) • 8 (1)
16#81A4	The module does not support this number of stop bits.	Select a suitable value for "Number of stop bits" (STOPBITS). The following are valid: • 1 (1) • 2 (2)
16#81A5	The module does not support this type of data flow control.	Select a valid data flow control for the module (FLOWCTRL).
16#81A7	Invalid value for XON or XOFF	Select suitable values for XON (XONCHAR) and XOFF (XOFFCHAR). Valid range of values: 0...255
16#81AA	Invalid operating mode	Valid operating modes are: • Full duplex (RS232) (0) • Full duplex (RS422) four-wire mode (point-to-point) (1) / (CM PtP (ET 200SP)) • Full duplex (RS422) four-wire mode (multipoint master) (2)/ (CM PtP (ET 200SP)) • Full duplex (RS 422) four-wire mode (multipoint slave) (3) • Half duplex (RS485) two-wire mode. (4)
16#81AB	Invalid receive line initial state	Valid initial states are: • "No" default setting (0) • Signal R(A)=5 V, signal R(B)=0 V (break detection) (1): Can only be selected with: "Full duplex (RS422) four-wire operation (point-to-point connection)" and "Full duplex (RS422) four-wire operation (multipoint slave)". • Signal R(A)=0 V, signal R(B)=5 V (2): This default setting corresponds to the idle state (no active send operation).
16#81AC	Invalid value for "Break detection"	Select a suitable value for "Break detection". The following are valid: • Break detection deactivated (0) • Break detection activated (1).

Error code	Description	Remedy
16#81AF	The module does not support this protocol.	Select a valid protocol for the module.
Error codes of the "Send configuration"		
16#81B5	More than two end delimiters or end sequence > 5 characters	Select suitable values for "End delimiter" and "End sequence". The following are valid: • deactivated (0), • 1 (1) or 2 (2) end delimiters or • deactivated (0), • 1 (1) up to 5 (5) characters for the end sequence.
16#81B6	Send configuration rejected because the 3964(R) protocol was selected	Make sure that no send configuration is transmitted if the 3964(R) protocol is set.
Error codes of the "Receive configuration"		
16#81C0	Invalid start condition	Select a suitable start condition. The following are valid: • Send break before frame start • Send Idle Line.
16#81C1	Invalid end condition or no end condition selected	Select a suitable end condition (see Receiving data with Freeport (Page 991)).
16#81C3	Invalid value for "Maximum message length"	Select a suitable value for "Maximum message length" (MAXLEN). Valid range of values (depending on the module): 1-1024/2048/4096 (Byte)
16#81C4	Invalid value for "Offset of the length specification in the message"	Select a suitable value for "Offset of the length specification in the message". Valid range of values (depending on the module): 1-1024/2048/4096 (Byte)
16#81C5	Invalid value for "Size of length field"	Select a suitable value for "Size of length field" (LENGTHSIZE). Valid range of values in bytes: • 1 (1) • 2 (2) • 4 (4)
16#81C6	Invalid value for "Number of characters not counted in length specification"	Select a suitable value for "Number of characters not counted in length specification" (LENGTHM). Valid range of values: 0 to 255 (bytes)
16#81C7	The total of "Offset in the message + size of length field + number of characters not counted" is greater than the maximum frame length	Select a suitable value for "Offset in message", "Size of length field" and "Number of characters not counted". Valid range of values: • Offset in the message (depending on the module): 0-1024/2048/4096 (Byte) • Size of length field: 1, 2, or 4 (bytes) • Number of characters not counted: 0 to 255 (bytes)

Error code	Description	Remedy
16#81C8	Invalid value for "Response timeout"	Select a suitable value for "Response timeout". Valid range of values: 1-65535 (ms)
16#81C9	Invalid value for "Character delay time"	Select a suitable value for "Character delay time". Valid range of values: 1 to 65535 (bit times)
16#81CB	frame end sequence is activated, but no character is activated for the check	Activate one or several characters for the check.
16#81CC	frame start sequence is activated, but no character is activated for the check	Activate one or several characters for the check.
16#81CD	Invalid value for "Prevent overwriting"	Select a suitable value for "Prevent overwriting". The following are valid: • Prevent overwriting is deactivated (0) or • Prevent overwriting is activated (1)
16#81CE	Invalid value for "Clear receive buffer on startup"	Select a suitable value for "Clear receive buffer on startup". The following are valid: • Clear receive buffer at startup is deactivated (0) • Clear receive buffer at startup is activated (1)
SEND status and error codes		
16#81D0	Receiving send request during runtime of a send command	Make sure that you do not receive an additional send request during runtime of a send command.
16#81D1	The waiting time for XON or CTS = ON has expired.	The communication partner has a fault, is too slow or is offline. Check the communication partner or change the parameters, if necessary.
16#81D2	"Hardware RTS always ON": Send job canceled due to change from DSR = ON to OFF	Check the communication partner. Make sure that DSR is ON for the entire duration of transmission.
16#81D3	Send buffer overflow / send frame too long	Select a shorter frame length. The following are valid (depending on the module): 1 to 1024/2048/4096 (bytes)
16#81D5	Transmission canceled due to parameter changes, detected wire break, or CPU in STOP	Check the parameter assignment, wire break, and CPU status.
16#81D6	Transmission canceled because end identifier was not received	Check the parameter assignment of the end characters and the frame of the communication partner.
16#81D7	Communication error between the user program and module	Check the communication (e.g., matching the sequence number).
16#81D8	Transmission attempt rejected because module is not configured	Configure the module.
Error codes of the receive configuration		
16#81E0	Frame aborted: Send buffer overflow / send frame too long	Increase the call rate for the receive function in the user program or configure communication with data flow control.
16#81E1	Frame aborted: Parity error	Check the connection line of the communication partners, or verify that both devices are configured for the same data transmission rate, parity and stop bit number.
16#81E2	Frame aborted: Character frame error	Check the settings for start bit, data bits, parity bit, data transmission rate, and stop bit(s).

Error code	Description	Remedy
16#81E3	Frame aborted: Character overflow error	Firmware error: Please contact Customer Support.
16#81E4	Frame aborted: The total length of "Offset in the message + size of the length field + number of characters not counted" is greater than the receive buffer	Select a suitable value for offset in message, size of the length field, and number of characters not counted.
16#81E5	Frame aborted: Break	Receive line to partner is interrupted. Reconnect or switch on partner.
16#81E6	Maximum number of "Buffered received frames" exceeded	In the user program call the instruction more often or configure a communication with data flow control or increase the number of buffered frames.
16#81E7	Synchronization error module and Receive_P2P	Make sure that different instances of the Receive_P2P do not access the same module.
16#81E8	Frame aborted: The character delay time has expired before the message end criterion was detected	Partner device faulty or too slow. Check this, if required, using an interface tester that is interconnected in the transmission line.
16#81E9	Modbus CRC error (only communication modules which support Modbus)	Checksum error of the Modbus frame. Check the communication partner.
16#81EA	Modbus frame too short (only communication modules which support Modbus)	Minimum length of Modbus frame not met. Check the communication partner.
16#81EB	Frame aborted: Maximum frame length reached	Select a shorter frame length at the communication partner. The following are valid (depending on the module): 1-1024/2048/4096 (Byte) Check the parameters for end of frame detection.
Error codes V24 accompanying signals		
16#81F0	The module does not support V24 accompanying signals	You have tried to set accompanying signals for a module that does not support V24 accompanying signals. Make sure that this is an RS232 module or that RS232 mode (ET 200SP) is set.
16#81F1	No operation of the V24 accompanying signals	The V24 accompanying signals cannot be operated manually if hardware data flow control is active.
Error codes of the receive configuration		
16#8201 [1)]	Receive_Conditions is a pointer to an invalid data type	Enter a pointer to one of the following data types: DB, BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TIME_OF_DAY, TIME, S5TIME, DATE_AND_TIME, STRING
16#8225	Receive_Conditionspoints to an optimized memory area greater than 1 kB or Receive_Conditions points to an optimized memory area and the receive length is greater than the area addressed by Receive_Conditions.	Enter a pointer to an area with a maximum length of: • Optimized memory area: 1 kByte • Non-optimized area: 4 kByte Note: If the pointer points to an optimized memory area, do not send more than 1 kByte.
16#8229 [1)]	Receive_Conditions is a pointer to BOOL with a number of bits not equal to n * 8	If you are using a pointer to BOOL, the number of bits must be a multiple of 8.
Error codes, general		

Error code	Description	Remedy
16#8280	Negative acknowledgment when reading module	You can find more detailed information on error causes in theRDREC.STATUS static parameters and in the description of the SFB RDREC. • Check the input at the PORT parameter • Set the COM_RST parameter prior to the initial call.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter You can find more detailed information on error causes in the WRREC.STATUS static parameters and in the description of the SFB WRREC.
16#8282	Module not available	Check the input at the PORT parameter and ensure that the module can be reached.
Error codes of the receive configuration		
16#82C1	Invalid value for "Buffered received frames".	Select a suitable value for "Buffered received frames". Valid range of values: 1-255
16#82C2	Receive configuration rejected because the 3964(R) protocol was selected	Make sure that no receive configuration is sent if the 3964(R) protocol is set.
16#8301 1)	Receive_Conditions is a pointer to an invalid data type	Select a valid data type. The following are valid: DB, BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TIME_OF_DAY, TIME, S5TIME, DATE_AND_TIME, STRING
16#8322	Range length error when reading a parameter	Check the input at the Receive_Conditions parameter
16#8324	Range error when reading a parameter	Check the input at the Receive_Conditions parameter
16#8328	Setting error when reading a parameter	Check the input at the Receive_Conditions parameter
SEND status and error codes		
16#8328 1)	BUFFER is a pointer to BOOL with a number of bits not equal to n * 8	If you are using a pointer to BOOL, the number of bits must be a multiple of 8.
Error codes of the receive configuration		
16#8332	Invalid data block at the Receive_Conditions parameter	Check the input at the Receive_Conditions parameter
16#833A	The designation of the data block at the Receive_Conditions parameter refers to a data block which is not loaded.	Check the input at the Receive_Conditions parameter
16#8351	Invalid data type	Check the input at the Receive_Conditions parameter
16#8352 1)	Receive_Conditions does not point to a data block	Check the pointer to Receive_Conditions
16#8353 1)	Receive_Conditions does not point to a structure of the type Receive_Conditions	Check the pointer to Receive_Conditions
Error codes 3964(R) protocol		
16#8380	Parameter assignment error: Invalid value for "Character delay time".	Select a suitable value for "Character delay time" (CharacterDelayTime). Valid range of values: 1 to 65535 (ms)
16#8381	Parameter assignment error: Invalid value for "Response timeout".	Select a suitable value for "Response timeout" (AcknDelayTime). Valid range of values: 1 to 65535 (ms)

Error code	Description	Remedy
16#8382	Parameter assignment error: Invalid value for "Priority".	Select a suitable value for "Priority" (Priority). The following are valid: • High (1) • Low (0)
16#8383	Parameter assignment error: Invalid value for "Block check"	Select a suitable value for "Block check" (BCC). The following are valid: • With block check (1) • Without block check (0)
16#8384	Parameter assignment error: Invalid value for "Connection attempts".	Select a suitable value for "Connection attempts" (BuildupAttempts). Valid range of values: 1-255
16#8385	Parameter assignment error: Invalid value for "Transmission attempts".	Select a suitable value for "Transmission attempts" (RepetitionAttempts). Valid range of values: 1-255
16#8386	Runtime error: Number of connection attempts exceeded	Check the interface cable and the transmission parameters. Also check whether the receive function is configured correctly at the partner device.
16#8387	Runtime error: Number of transmission attempts exceeded	Check the interface cable, the transmission parameters and the configuration of the communication partner.
16#8388	Runtime error: Error at the "Block check character" The internally calculated value of the block check character does not correspond to the block check character received by the partner at the connection end.	Check whether the connection is seriously disrupted; in this case you may also occasionally see error codes. Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#8389	Runtime error: Invalid character received while waiting for free receive buffer	The send request of the communication partner (STX, 02H) is only answered with DLE when the receive buffer is empty. No additional character may be received before (except STX again). Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#838A	Runtime error: Logical error during receiving. After DLE was received, a further random character (other than DLE or ETX) was received.	Check if partner DLE in frame header and in data string is always in duplicate or the connection is released with DLE ETX. Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#838B	Runtime error: Character delay time exceeded	Partner device too slow or faulty. Verify by using an interface test device that is switched into the transmission line, if necessary.
16#838C	Runtime error: Wait time for free receive buffer has started	In the user program call the instruction more often or configure a communication with data flow control.
16#838D	Runtime error: frame repetition does not start within 4 s after NAK	Check the communication partner. A received frame that is possibly corrupted must be repeated by the partner within 4 seconds.

Error code	Description	Remedy
16#838E	Runtime error: In idle mode, one or several characters (other than NAK or STX) were received.	Check for proper function of the partner device, possibly using an interface test device that is switched into the transmission line.
16#838F	Runtime error: Initialization conflict - Both partners have set high priority	Set the "Low" priority at one of the partners
16#8391	Parameter assignment error: 3964 configuration data rejected because Freeport is set	If the Freeport protocol is set, make sure that no 3964 parameter assignment data is sent.

[1] Only with instructions for S7-300/400 CPUs

Overview of error messages - Modbus

Error code	Description	Remedy
16#0000	No error	-
Configuration error of the interface - Modbus_Comm_Load		
16#8181	The module does not support this data transmission rate.	Select a valid data transmission rate for the module at the BAUD parameter.
16#8182	The module does not support this parity setting.	Select a suitable value for "Parity" at the PARITY parameter. The following are valid: • None (1) • Even (2) • Odd (3) • Mark (4) • Space (5) • Any (6)
16#8183	The module does not support this type of data flow control.	Select a valid data flow control for the module at the FLOW_CTRL parameter.
16#8184	Invalid value for "Response timeout".	Select a suitable value for "Response timeout" at the RESP_TO parameter. Valid range of values: 1 to 65535 (ms)
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the Send_Config.RDREC.STATUS or Receive_Config.RDREC.STATUS static parameters or RDREC.STATUS and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the Send_Config.WRREC.STATUS or Receive_Config.WRREC.STATUS static parameters or WRREC.STATUS and in the description of the SFB WRREC.
16#8282	Module not available	Check the input at the PORT parameter and ensure that the module can be reached.
Configuration error - Modbus_Slave		

Error code	Description	Remedy
16#8186	Invalid slave address	Select a suitable slave address at the MB_ADDR parameter. The following are valid: 1-247 at standard address area; 1-65535 at extended address area (0 is reserved for Broadcast)
16#8187	Invalid value at MB_HOLD_REG parameter	Select a suitable value for the hold register at the MB_HOLD_REG parameter.
16#8188	Invalid operating mode or broadcast (MB_ADDR = 0) and MODE parameter ≠ 1	Select the value 1 for MODE in Broadcast mode or select a different operating mode.
16#818C	The pointer to a MB_HOLD_REG area must be a data block or a bit memory address area.	Select a suitable value for the pointer to the MB_HOLD_REG area.
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.WRREC.STATUS or Receive_P2P.WRREC.STATUS, and in the description of the SFB WRREC.
16#8452 [1)]	MB_HOLD_REG is not a pointer to a DB or a bit memory area	Check the MB_HOLD_REG pointer
16#8453 [1)]	MB_HOLD_REG is not a pointer of type BOOL or WORD	Check the MB_HOLD_REG pointer
16#8454 [1)]	The area addressed by MB_HOLD_REG is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the MB_HOLD_REG pointer
16#8455 [1)]	MB_HOLD_REG points to a write-protected DB	Check the MB_HOLD_REG pointer
16#8456 [1)]	Error during instruction execution. The cause of the error is shown in the STATUS parameter.	Determine the value of the SFCSTATUS parameter. Check what this means in the description for SFC51, STATUS parameter.
Configuration error - Modbus_Master		
16#8180	Invalid value for MB_DB parameter	The value configured for MB_DB (instance data DB) at the Modbus_Comm_Load instruction is not valid. Check the interconnection of the Modbus_Comm_Load instruction and its error messages.
16#8186	Invalid station address	Select a suitable station address at the MB_ADDR parameter. The following are valid: 1-247 at standard address area; 1-65535 at extended address area (0 is reserved for Broadcast)
16#8188	Invalid operating mode or broadcast (MB_ADDR = 0) and MODE parameter ≠ 1	Select the value 1 for MODE in Broadcast mode or select a different operating mode.

Error code	Description	Remedy
16#8189	Invalid data address	Select a suitable value for the data address at the DATA_ADDR parameter. See description Modbus_Master (Page 5249) in the Info system
16#818A	Invalid length	Select a suitable data length at the DATA_LEN parameter. See description Modbus_Master (Page 5249) in the Info system
16#818B	Invalid value for DATA_PTR	Select a suitable value for the data pointer at the DATA_PTR parameter (M or DB address). See description Modbus_Master (Page 5249) in the Info system
16#818C	Interconnection error of the DATA_PTR parameter	Check the interconnection of the instruction.
16#818D	The area addressed by DATA_PTR is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the DATA_PTR pointer
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.WRREC.STATUS or Receive_P2P.WRREC.STATUS or Receive_Reset and in the description of the SFB WRREC.
Communication errors - Modbus_Master and Modbus_Slave		
16#80 D1	The waiting time for XON or CTS = ON has expired.	The communication partner has a fault, is too slow or is offline. Check the communication partner or change the parameters, if necessary.
16#80D2	"Hardware RTS always ON": Send job canceled due to change from DSR = ON to OFF	Check the communication partner. Make sure that DSR is ON for the entire duration of transmission.
16#80E0	Frame aborted: Send buffer overflow / send frame too long	In the user program call the instruction more often or configure a communication with data flow control.
16#80E1	Frame aborted: Parity error	Check the connection line of the communication partners, or verify that both devices are configured for the same data transmission rate, parity and stop bit number.
16#80E2	Frame aborted: Character frame error	Check the settings for start bit, data bits, parity bit, data transmission rate, and stop bit(s).
16#80E3	Frame aborted: Character overflow error	Check the number of data in the frame of the communication partner.
16#80E4	Frame aborted: Maximum frame length reached	Select a shorter frame length at the communication partner. The following are valid (depending on the module): 1 to 1024/2048/4096 (bytes)

Error code	Description	Remedy
Communication error - Modbus_Master		
16#80C8	The slave does not respond within the set time	Check the data transmission rate, parity and wiring of the slave.
16#8200	The interface is busy with an ongoing request.	Repeat the command later. Make sure that there are no commands still running before you start a new one.
Protocol error - Modbus_Slave (only communication modules that support Modbus)		
16#8380	CRC error	Checksum error of the Modbus frame. Check the communication partner.
16#8381	The function code is not supported or is not supported for broadcast.	Check the communication partner and make sure that a valid function code is sent.
16#8382	Invalid length information in the request frame	Select a suitable data length at the DATA_LEN parameter.
16#8383	Invalid data address in the request frame	Select a suitable value for the data address at the DATA_ADDR parameter.
16#8384	Invalid data value error in the request frame	Check the data value in the request frame of the Modbus master
16#8385	The diagnostic value is not supported by the Modbus slave (function code 08)	The Modbus slave only supports the diagnostic values 16#0000 and 16#000A.
Protocol error - Modbus_Master (only communication modules that support Modbus)		
16#8380	CRC error	Checksum error of the Modbus frame. Check the communication partner.
16#8381	Response frame from Modbus Slave with the following error message: The function code is not supported.	Check the communication partner and make sure that a valid function code is sent.
16#8382	Response frame from Modbus Slave with the following error message: Invalid length	Select a suitable data length.
16#8383	Response frame from Modbus Slave with the following error message: Invalid data address in the request frame	Select a suitable value for the data address at the DATA_ADDR parameter.
16#8384	Response frame from Modbus Slave with the following error message: Data value error	Check the request frame to the Modbus slave.
16#8385	Response frame from Modbus Slave with the following error message: The diagnostic value is not supported by the Modbus slave	The Modbus slave only supports the diagnostic values 16#0000 and 16#000A.
16#8386	The returned function code does not match the requested function code.	Check the response frame and the addressing of the slave.
16#8387	A slave that was not requested answers	Check the response frame of the slave. Check the address settings of the slave.
16#8388	Error in the response of the slave to a write request.	Check the response frame of the slave.
16#8828 [1)]	DATA_PTR points to a bit address that is not equal to n * 8	Check the DATA_PTR pointer
16#8852 [1)]	DATA_PTR is not a pointer to a DB or a bit memory area	Check the DATA_PTR pointer
16#8853 [1)]	DATA_PTR is not a pointer of type BOOL or WORD	Check the DATA_PTR pointer
16#8855 [1)]	DATA_PTR points to a write-protected DB	Check the DATA_PTR pointer
16#8856 [1)]	Error during call of SFC51	Call the Modbus_Master instruction again
Error - Modbus_Slave (only communication modules that support Modbus)		

Error code	Description	Remedy
16#8428 [1]	MB_HOLD_REG points to a bit address that is not equal to n * 8	Check the MB_HOLD_REG pointer
16#8452 [1]	MB_HOLD_REG is not a pointer to a DB or a bit memory area	Check the MB_HOLD_REG pointer
16#8453 [1]	MB_HOLD_REG is not a pointer of type BOOL or WORD	Check the MB_HOLD_REG pointer
16#8454 [1]	The area addressed by MB_HOLD_REG is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the MB_HOLD_REG pointer
16#8455 [1]	MB_HOLD_REG points to a write-protected DB	Check the MB_HOLD_REG pointer
16#8456 [1]	Error during call of SFC51	Call the Modbus_Slave instruction again

[1] Only with instructions for S7-300/400 CPUs

Overview of error messages - USS

Error code	Description	Remedy
16#0000	No error	-
16#8180	Length error in response of drive	Check the response frame of the drive.
16#8181	Data type error	Select a suitable data type. The following are valid: • Real • Word • Double word
16#8182	Data type error: "Double word" or "Real" may not be returned for the "Word" request.	Check the response frame of the drive.
16#8183	Data type error: "Word" may not be returned for the "Double word" or "Real" request.	Check the response frame of the drive.
16#8184	Checksum error in response of drive	Check the drive and the communication connection.
16#8185	Addressing error	Valid drive address range: 1 to 16
16#8186	Setpoint error	Valid setpoint range: -200 % to +200 %
16#8187	Incorrect drive number returned	Check the response frame of the drive.
16#8188	Invalid PZD length	Permitted PZD lengths: 2, 4, 6, 8 words
16#8189	The module does not support this data transmission rate.	Select a valid data transmission rate for the module.
16#818A	A different request for this drive is currently active.	Repeat the parameter read or write command later.
16#818B	The drive does not respond.	Check the drive.
16#818C	The drive responds with an error message to a parameter request.	Check the response frame of the drive. Check the parameter request.
16#818D	The drive responds with an access error message to a parameter request.	Check the response frame of the drive. Check the parameter request.
16#818E	The drive was not initialized.	Check the user program and make sure that the USS_Drive_Control instruction is called for this drive.

Error code	Description	Remedy
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Port_Config.RDREC.STATUS, Send_Config.RDREC.STATUS , Receive_Config.RDREC.STATUS , Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Port_Config.WRREC.STATUS, Send_Config.WRREC.STATUS , Receive_Config.WRREC.STATUS , Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB WRREC.

[1] Only with instructions for S7-300/400 CPUs

MODBUS (RTU)

Overview of the Modbus RTU communication

Modbus RTU communication

Modbus RTU (Remote Terminal Unit) is a standard protocol for communication in the network and uses the RS232 or RS422/485 connection for serial data transmission between Modbus devices in the network.

Modbus RTU uses a master/slave network in which all communication is triggered by a single master device while the slaves can only respond to the request of the master. The master sends a request to a slave address and only the slave with this slave address responds to the command.

Exception: Modbus slave address 0 sends a broadcast frame to all slaves (without slave response).

Modbus function codes

- A CPU that is operated as a Modbus RTU master can read and write data and I/O states in a Modbus RTU slave connected by means of a communication connection.
- A CPU operated as a Modbus RTU slave allows a Modbus RTU master connected over a communication connection to read and write data and I/O states in its own CPU.

Table 9-141 Functions for reading data: Reading distributed I/O and program data

Modbus function code	Functions for reading data from the slave (server) - standard addressing
01	Read output bits: 1 to 2000/1992[1)] bits per request
02	Read input bits: 1 to 2000/1992[1)] bits per request
03	Read hold register: 1 to 125/124[1)] words per request
04	Read input words: 1 to 125/124[1)] words per request

1) for extended addressing

Table 9-142 Functions for writing data: Changing distributed I/O and program data

Modbus function code	Functions for writing of data in the slave (server) - standard addressing
05	Write one output bit: 1 bit per request
06	Write one hold register: 1 word per request
15	Write one or several output bits: 1 to 1960 bits per request
16	Write one or several hold registers: 1 to 122 words per request

- The Modbus function codes 08 and 11 offer diagnostic options for communication with the slave device.
- Modbus slave address 0 sends a broadcast frame to all slaves (without slave response; for function codes 5, 6, 15, 16).

Table 9-143 Station addresses in the Modbus network

Station		Address
RTU station	Standard station address	1 to 247 and 0 for broadcast
	Extended station address	1 to 65535 and 0 for broadcast

Modbus memory addresses

The number of Modbus memory addresses (input/output addresses) that is actually available depends on the CPU version and the available work memory.

Modbus RTU instructions in your program

- Modbus_Comm_Load: You need to run Modbus_Comm_Load to set up PtP parameters such as data transmission rate, parity and data flow control. Once you have configured the communication module for the Modbus RTU protocol, it can only be used by the Modbus_Master instruction or the Modbus_Slave instruction.
- Modbus_Master: The CPU can be used as Modbus RTU master device with the Modbus master instruction for communication with one or more Modbus slave devices.
- Modbus_Slave: The CPU can be used as Modbus RTU slave device with the Modbus slave instruction for communication with one Modbus master device.

Modbus_Comm_Load: Configure communication module for Modbus

Description

The Modbus_Comm_Load instruction configures a communication module for communication by means of the Modbus RTU protocol. An instance data block is automatically assigned when you add the Modbus_Comm_Load instruction in your program.

Parameters

Parameters	Declaration	Data type		Standard	Description
		S7-1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the instruction upon a positive edge of this input.
PORT	IN	Port	Laddr	0	The CM port value can be found in the "Hardware ID" property of the device configuration (S7-1500) (input address; S7-300/400) once you have configured the CM. The symbolic port name is assigned in the "System constants" tab of the PLC tag table. In S7-300/400/WinAC systems, the input address assigned in HWCN must be assigned to the CM port.
BAUD	IN	UDInt	DWord	9600	Selection of the data transmission rate Valid values are: 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 76800, 115200 bit/s.
PARITY	IN	UInt	Word	1	Selection of parity: • 0 – None • 1 – Odd • 2 – Even
FLOW_CTRL	IN	UInt	Word	0	Selection of flow control: • 0 – (default) no flow control • 1 – Hardware flow control with RTS always ON (not with RS422/485 CMs) • 2 – Hardware flow control with RTS switched (not with RS422/485 CMs)
RTS_ON_DLY	IN	UInt	Word	0	Selection RTS ON delay: • 0 – No delay from "RTS active" until the first character of the frame is sent. • 1 to 65535 – Delay in milliseconds from "RTS active" until the first character of the frame is sent (not with RS422/485 CMs). RTS delays must be used independent of the selection FLOW_CTRL.

Parameters	Declaration	Data type		Standard	Description
		S7-1500	S7-300/400/WinAC		
RTS_OFF_DLY	IN	UInt	Word	0	Selection RTS OFF delay: • 0 – No delay after transmission of last character until "RTS inactive" • 1 to 65535 – Delay in milliseconds after transmission of last character until "RTS inactive" (not with RS422/485 ports). RTS delays must be used independent of the selection FLOW_CTRL.
RESP_TO	IN	UInt	Word	1000	Response timeout: 5 ms to 65535 ms - Time in milliseconds that Modbus_Master waits for a response from the slave. If the slave does not respond within this period, Modbus_Master repeats the request or terminates the request with an error if the specified number of repetitions (see below, RETRIES parameter) has been sent.
MB_DB	IN/OUT	MB_BASE		-	A reference to the instance data block of the Modbus_Master or Modbus_Slave instructions. The MB_DB parameter must be connected with the (static and therefore not visible in the instruction) MB_DB parameter of the Modbus_Master or Modbus_Slave instruction.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Modbus_Comm_Load instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
DONE	OUT	Bool		FALSE	The DONE bit is TRUE for one cycle after the last request has been completed without errors.
ERROR	OUT	Bool		FALSE	The ERROR bit is TRUE for one cycle after the last request has been completed with errors. The error code in the STATUS parameter is only valid in the cycle in which ERROR = TRUE.
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 5264))

Modbus_Comm_Load is executed to configure a port for the Modbus RTU protocol. Once you have configured the port for the Modbus RTU protocol, it can only be used by the Modbus_Master or Modbus_Slave instructions.

You have to run Modbus_Comm_Load for the configuration of each communication port that is to be used for Modbus communication. You must assign a unique Modbus_Comm_Load instance DB to each port that you use. Only run Modbus_Comm_Load again if you need to change communication parameters, such as data transmission rate or parity, or in case the network has returned.

For example, an instance data block is assigned to the instruction if you add Modbus_Master or Modbus_Slave to your program. You need to connect the MB_DB parameter of the Modbus_Comm_Load instruction to the MB_DB parameter of the Modbus_Master or Modbus_Slave instruction.

Modbus_Comm_Load data block tags

The table below shows the public static tags in the instance DB of Modbus_Comm_Load that you can use in your program.

Table 9-144 Static tags in the instance DB

Tag	Data type		Standard	Description
	S7-1500	S7-300/400/WinAC		
ICHAR_GAP	Word		0	Maximum character delay time between characters. This parameter is specified in milliseconds and increases the anticipated period between the received characters. The corresponding number of bit times for this parameter is added to the Modbus default value of 35 bit times (3.5 character times).
RETRIES	Word		2	Number of retries that the master executes before the error code 0x80C8 for "No response" is returned.
EN_SUPPLY_VOLT	Bool		0	Enable diagnostics for missing supply voltage L+
MODE	USInt	Byte	0	Operating mode Valid operating modes are: • 0 = Full duplex (RS232) • 1 = Full duplex (RS422) four-wire operation (point-to-point) • 2 = Full duplex (RS 422) four-wire mode (multipoint master, CM PtP (ET 200SP)) • 3 = Full duplex (RS 422) four-wire mode (multipoint slave, CM PtP (ET 200SP)) • 4 = Half duplex (RS485) two-wire mode
LINE_PRE	USInt	Byte	0	Receive line initial state Valid initial states are: • 0 = "no" initial state • 1 = signal R(A)=5 V, signal R(B)=0 V (break detection): Break detection is possible with this initial state. Can only be selected with: "Full duplex (RS422) four-wire operation (point-to-point connection)" and "Full duplex (RS422) four-wire operation (multipoint slave)". • 2 = signal R(A)=0 V, signal R(B)=5 V: This default setting corresponds to the idle state (no active send operation). No break detection is possible with this initial state.
CABLE_BRK	USInt	Byte	0	Break detection The following are valid: • 0 = break detection deactivated • 1 = break detection activated
EN_DIAG_ALARM	Bool		0	Activate diagnostics interrupt: • 0 - not activated • 1 - activated

Modbus_Master: Communicate as Modbus master

Description

The Modbus_Master instruction communicates as Modbus master via a port configured by the Modbus_Comm_Load instruction. An instance data block is automatically assigned when you add the Modbus_Master instruction in your program. The MB_DB parameter of the Modbus_Comm_Load instruction must be connected to the (static) MB_DB parameter of the Modbus_Master instruction.

Parameters

Parameters	Declaration	Data type		Standard	Description
		S7-1500	S7-300/400/ WinAC		
REQ	IN	Bool		FALSE	FALSE = no request TRUE = request to send data to the Modbus slave
MB_ADDR	IN	UInt	Word	-	Modbus RTU station address: Standard addressing range (1 to 247 as well as 0 for Broadcast) Extended addressing range (1 to 65535 as well as 0 for Broadcast) The value 0 is reserved for the broadcast of a frame to all Modbus slaves. Only the Modbus function codes 05, 06, 15 and 16 are supported for the broadcast.
MODE	IN	USInt	Byte	0	Mode selection: Specifies the type of request (read, write or diagnostics). Additional information is available in the table of Modbus functions below.
DATA_ADDR	IN	UDInt	DWord	0	Start address in the slave: Specifies the start address of the data that is accessed in the Modbus slave. The valid addresses are listed in the table of Modbus functions below.
DATA_LEN	IN	UInt	Word	0	Data length: Specifies the number of bits or words this instruction is to access. The valid lengths are listed in the table of Modbus functions below.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Modbus_Master instruction You initialize the instruction with TRUE. The instruction then resets COM_RST to FALSE.
DATA_PTR	IN/OUT	Variant	Any	-	Data pointer: Points to the flag or DB address for the data to be written or read.
DONE	OUT	Bool		FALSE	The DONE bit is TRUE for one cycle after the last request has been completed without errors.
BUSY	OUT	Bool		-	• FALSE – no command active for Modbus_Master • TRUE – command for Modbus_Master in progress
ERROR	OUT	Bool		FALSE	The ERROR bit is TRUE for one cycle after the last request has been completed with errors. The error code in the STATUS parameter is only valid in the cycle in which ERROR = TRUE.
STATUS	OUT	Word		0	Error code (see Error messages (Page 5264))

Tags in the data block of the Modbus master

The table below shows the public static tags in the instance DB of Modbus_Master that you can use in your program.

Table 9-145 Static tags in the instance DB

Tag	Data type	Standard	Description
Blocked_Proc_Timeout	Real	0.5	Duration (in seconds) for which to wait for a blocked Modbus master instance before this instance is removed as ACTIVE. This may happen, for example, if a master request was output and the program then stops to call the master function before it has completely finished the request. The time value must be greater than 0 and less than 55 seconds to avoid an error to occur. See also "Rules for communication by the Modbus-Master" and "Calling theModbus_Master instruction with different parameter settings".
Extended_Addressing	Bool	FALSE	Configures the slave station address as single or double byte. • FALSE = One-byte address; 0 to 247 • TRUE = Two-byte address (corresponds to extended addressing); 0 to 65535
Compatibility_Mode [1]	Bool	FALSE	Compatibility mode with CP 341 and CP 441-2 with driver for Modbus RTU and with ET 200SP 1SI for Modbus. The default value is 0. • FALSE = as per Modbus specification, not compatible • TRUE = compatible – For FC1 and FC2: The data read from the received frame is written word for word to the addressed CPU memory and exchanged byte by byte. If the number of bits to be transmitted is not a multiple of 16, the bits which are not relevant are set to null in the last word. – For FC15: The words to be transmitted are read word by word from the addressed memory and written byte by byte to the send frame. If the number of bits to be transmitted is not a multiple of 8, the bits in the last byte which are not relevant are read unchanged from the addressed memory and entered in the send frame.
MB_DB	MB_BASE	-	The MB_DB parameter of the Modbus_Comm_Load instruction must be connected to this MB_DB parameter of the Modbus_Master instruction.

[1] The PtP communication modules respond as defined in the Modbus specification. To retain a response as with CP 341, CP 441-2 and ET 200SP 1SI for Modbus, use the "Compatibility_Mode" parameter.

You program can write values to the Blocked_Proc_Timeout and Extended_Addressing variables to control the Modbus master operations.

Rules for communication by the Modbus-Master

- Modbus_Comm_Load must be run to configure a port so that the Modbus_Master instruction can communicate with this port.
- A port which is to be used as Modbus master must not be used by Modbus _Slave . You can use one or several instances of Modbus_Master [1] with this port. But all versions of the Modbus_Master must use the same instance DB for the port.
- The Modbus instructions do not use communication alarm events to control the communication process. Your program must query the Modbus_Master instruction for completed commands (DONE, ERROR).
- We recommend to call all executions of Modbus_Master for a specific port from a program cycle OB. Modbus master instructions can only be executed in one program cycle or in one cyclical/time-controlled processing level. They may not be processed in different processing levels. The priority interruption of a Modbus master instruction by another Modbus master instruction in a processing level with higher priority results in improper operation. Modbus master instructions may mot be processed in startup, diagnostic or time error levels.

[1] "Instance of Modbus master" here means a call of the Modbus_Master instruction with the same interconnection to a Modbus_Comm_Load instruction and the same setting for the MB_ADDR, MODE, DATA_ADDR and DATA_LEN parameters.

Example

Modbus_Master is called with MODE=0 and DATA_ADDR=10

This job is now active until it is completed with DONE=1 or ERROR=1 or until the time monitoring configured at the Blocked_Proc_Timeout parameter has expired. If a new command is started after the watchdog time expires and before the previous command has been completed, the previous command is aborted without an error message.

If, while this command is running, the instruction is now called a second time with the same instance data but different MODE and DATA_ADDR parameter settings, this second call is terminated with ERROR=1 and STATUS=8200.

Calling the Modbus_Master instruction with different parameter settings

If multiple calls of the Modbus_Master instruction with different settings for MB_ADDR, MODE, DATA_ADDR or DATA_LEN are placed in your program, you must ensure that only one of these calls is active at any given time. Otherwise, the error message 0x8200 is output (interface is busy with an ongoing request).

If a call cannot be processed in full, the watchdog is activated by the Blocked_Proc_Timeout parameter and terminates the ongoing command.

REQ parameter

FALSE = no request; TRUE = request to send data to the Modbus slave

Enable the requested transmission. This transmits the contents of the buffer to the point-to-point communication interface.

You use the DATA_ADDR and MODE parameters to select the Modbus function code.

DATA_ADDR (Modbus start address in the slave): Specifies the start address of the data that is accessed in the Modbus slave.

The Modbus_Master instruction uses the MODE input instead of a function code input. The combination of MODE and DATA_ADDR specifies the function code that is used in the actual Modbus frame. The table below shows how the MODE parameter, the Modbus function code and the Modbus address range in DATA_ADDR are related.

Table 9-146 Modbus functions

MODE	DATA_ADDR (Modbus address)			DATA_LEN (data length)			Modbus function code	Operation and data		
0				Bits per request			01	Read output bits:		
	1	to	9999	1	to	2000/1992 [1]		0	to	9998
0				Bits per request			02	Read input bits:		
	10001	to	19999	1	to	2000/1992 [1]		0	to	9998
0				Words per request			03	Read hold register:		
	40001	to	49999	1	to	125/124 [1]		0	to	9998
	400001	to	465535	1	to	125/124 [1]		0	to	65534
0				Words per request			04	Read input words:		
	30001	to	39999	1	to	125/124 [1]		0	to	9998
1				Bits per request			05	Write one output bit:		
	1	to	9999	1				0	to	9998
1				1 word per request			06	Write one hold register:		
	40001	to	49999	1				0	to	9998
	400001	to	465535	1				0	to	65524
1				Bits per request			15	Write multiple output bits:		
	1	to	9999	2	to	1968/1960 [1]		0	to	9998
1				Words per request			16	Write multiple hold registers:		
	40001	to	49999	2	to	123/122		0	to	9998
	400001	to	465534	2	to	123/122 [1]		0	to	65534
2 [2]				Bits per request			15	Write one or several output bits:		
	1	to	9999	2	to	1968/1960 [1]		0	to	9998
2 [2]				Words per request			16	Write one or several hold registers:		
	40001	to	49999	1	to	123		0	to	9998
	400001	to	465535	1	to	122 [1]		0	to	65534
11	Both DATA_ADDR and DATA_LEN operands of the Modbus_Master are ignored with this function.						11	Read status word and event counter of the slave communication. The status word indicates busy (0 – not busy, 0xFFFF - busy). The event counter is incremented for each successful processing of a frame.		

MODE	DATA_ADDR (Modbus address)			DATA_LEN (data length)			Modbus function code	Operation and data		
80				1 word per request			08	Check slave status with data diagnostic code 0x0000 (loopback test – slave returns an echo of the request)		
	-			1				-		
81				1 word per request			08	Reset slave event counter using data diagnostic code 0x000A		
	-			1				-		
3 to 10, 12 to 79, 82 to 255	-			-				Reserved		

[1] In extended addressing, see the Extended_Adressing parameter, the maximum data length is shorter by 1 byte or 1 word depending on the data type of the function.

[2] MODE 2 allows you to write one or more output bits and one or more hhold registers using the Modbus functions 15 and 16. MODE 1 uses the Modbus functions 5 and 6 to write 1 output bit and 1 hold register, and Modbus functions 15 and 16 to write multiple output bits and multiple hold registers.

DATA_PTR parameter

The DATA_PTR parameter points to the DB or bit memory address in which reading or writing is performed. If you use a data block, you must create a global data block that provides the data memory for read and write processes on Modbus slaves.

Note

S7-1500 - The data block addressed using DATA_PTR must support direct addressing

The data block must permit direct (absolute) and symbolic addressing.

Data block structures for the DATA_PTR parameter

- These data types are valid for **reading words** of Modbus addresses (DATA_PTR) 30001 to 39999, 40001 to 49999 and 400001 to 465535 as well as for **writing words** to Modbus addresses (DATA_PTR parameter) 40001 to 49999 and 400001 to 465535.
  - Standard array of data types WORD, UINT or INT
  - Named structure of the WORD, UINT or INT type in which each element has a unique name and a 16-bit data type.
  - Named complex structure in which each element has a unique name and a 16-bit or 32-bit data type.
- For **reading** and writing bits for the Modbus addresses (DATA_PTR parameter) 00001 to 09999 and for reading of bits from 10001 to 19999.
  - Standard field from Boolean data types.
  - Named Boolean structure from uniquely named Boolean variables.
- It is not essential, but nevertheless advisable, to allocate each Modbus_Master instruction its own separate memory area. The reason for this is that data destruction is far more likely if multiple Modbus_Master instructions are reading and writing in the same memory area.
- It is not necessary for the data areas for DATA_PTR to be located in the same global data block. You can create a data block with several areas for Modbus read processes, a data block for Modbus write processes or a data block for each slave station.

See also

Modbus_Slave: Communicate as Modbus slave (Page 5254)

Modbus_Slave: Communicate as Modbus slave

Description

Your program can use the Modbus_Slave instruction to communicate as a Modbus slave by using a CM (RS422/485 or RS232). STEP 7 automatically creates an instance DB when you add the instruction. The MB_DB parameter of the Modbus_Comm_Load instruction must be connected to the (static) MB_DB parameter of the Modbus_Slave instruction.

Parameters

Parameters	Declaration	Data type		Standard	Description
		S7-1500	S7-300/400/ WinAC		
MB_ADDR	IN	UInt	Word	-	Standard address of the Modbus slave: Standard addressing range (1 to 247) Extended addressing range (0 to 65535) Note: 0 is the broadcast address
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Modbus_Slave instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
MB_HOLD_REG	IN/OUT	Variant	Any	-	Pointer to the Modbus hold register DB: The Modbus hold register may be the memory area of the flags or a data block.
NDR	OUT	Bool		FALSE	New data available: • FALSE – No new data • TRUE – Indicates that new data was written by the Modbus master The NDR bit is TRUE for one cycle after the last request has been completed without errors.
DR	OUT	Bool		FALSE	Read data: • FALSE – No data read • TRUE – Indicates that the instruction has stored the data received by the Modbus master in the target area. The DR bit is TRUE for one cycle after the last request has been completed without errors.
ERROR	OUT	Bool		FALSE	The ERROR bit is TRUE for one cycle after the last request has been completed with errors. If the execution was terminated with an error, the error code in the STATUS parameter is only valid in the cycle in which ERROR = TRUE.
STATUS	OUT	Word		0	Error code (see Error messages (Page 5264))

The function codes of the Modbus communication (1, 2, 4, 5 and 15) can read and write bits and words directly in the process image of the inputs and in the process image of the outputs of the CPU. The MB_HOLD_REG parameter must be defined as data type greater than one byte for these function codes. The table below shows the sample assignment of Modbus addresses to the process image in the CPU.

Table 9-147 Assignment of Modbus addresses to the process image

Modbus functions						S7-1500 / S7-300 / S7-400			
Function code	Function	Data area	Address area			Data area	CPU address		
01	Read bits	Output	0	to	9998	Process image of outputs	O0.0	to	O1248.6
02	Read bits	Input	0	to	9998	Process image of inputs	I0.0	to	I1248.6
04	Read words	Input	0	to	9998	Process image of inputs	IW0	to	IW19996

Modbus functions						S7-1500 / S7-300 / S7-400			
Function code	Function	Data area	Address area			Data area	CPU address		
05	Write bit	Output	0	to	9998	Process image of outputs	O0.0	to	O1248.6
15	Write bits	Output	0	to	9998	Process image of outputs	O0.0	to	O1248.6

Note

The available address area may be smaller, depending on the memory configuration of the CPU.

The function codes of the Modbus communication (3, 6, 16) use a Modbus hold register which is an address area in the memory area of the flags or a data block. The type of hold register is specified by the MB_HOLD_REG parameter of the Modbus_Slave instruction.

Note

S7-1500 - Type of MB_HOLD_REG data block

The data block with Modbus hold register must permit direct (absolute) and symbolic addressing.

Variables of the Modbus slave

This table below shows the public static tags in the instance data block of Modbus_Slave that you can use in your program.

Table 9-148 Variables of the Modbus slave

Tag	Data type	Standard	Description
HR_Start_Offset	Word	0	Specifies the start address of the Modbus hold register (default = 0)
Extended_Addressing	Bool	FALSE	Extended addressing, configures slave addressing as single or double byte (FALSE = single byte address, TRUE = double byte address)
Request_Count	Word	0	The number of all requests received by this slave
Slave_Message_Count	Word	0	The number of requests received for this specific slave
Bad_CRC_Count	Word	0	The number of received requests that have a CRC error
Broadcast_Count	Word	0	The number of received broadcast requests
Exception_Count	Word	0	Modbus-specific errors that are acknowledged with an exception to the master
Success_Count	Word	0	The number of received requests without protocol errors for this specific slave
MB_DB	MB_BASE	-	The MB_DB parameter of the Modbus_Comm_Load instruction must be connected to this MB_DB parameter of the Modbus_Master instruction.

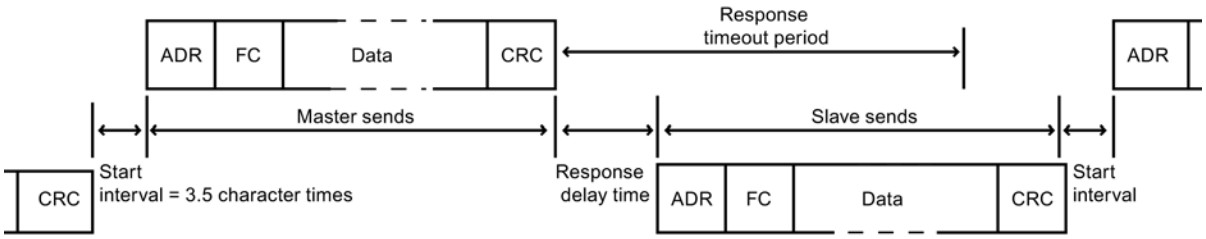
You program can write values to the HR_Start_Offset and Extended_Addressing variables and control the Modbus slave operations. The other variables can be read to monitor the Modbus status.

Rules for Modbus slave communication

- Modbus_Comm_Load must be run to configure a port so that the Modbus_Slave instruction can communicate by means of this port.
- If a port is to respond as slave to a Modbus master, this port may not be programmed with the Modbus_Master instruction.
- Only one instance of Modbus_Slave can be used with a specific port; otherwise you may encounter unexpected behavior.
- The Modbus instructions do not use communication alarm events to control the communication process. Your program must control the communication process by querying the Modbus_Slave instruction for completed send and receive processes.
- The Modbus_Slave instruction must be executed regularly with a frequency that allows a timely response to incoming requests of a Modbus master. We recommend executing Modbus_Slave in each cycle from a program cycle OB. Modbus_Slave can be executed from a cyclic interrupt OB but we do not recommend it, because excessive time delays in the interrupt program can temporarily block the execution of other interrupt programs.

Time control of the Modbus signal

Modbus_Slave must be executed regularly to receive each request of the Modbus master and respond accordingly. The frequency with which Modbus_Slave is executed depends on the timeout value specified for the response by the Modbus master. This can be seen in the figure below.



The timeout period of the (RESP_TO) response is the duration that a Modbus master waits for the beginning of an answer from a Modbus slave. This period is not defined by the Modbus protocol, but by a parameter of the Modbus_Comm_Load instruction. As both receiving and sending a frame requires multiple calls of the Modbus_Slave instruction (at least three), you should execute Modbus_Slave at least twelve times during the timeout period for the response of the Modbus master so that the Modbus slave receives and sends data twice as many times as specified by the timeout period.

HR_Start_Offset

The addresses of the Modbus hold register start with 40001 or 400001; these addresses correspond to the start address of the hold register in the target system memory. But you can

configure the HR_Start_Offset tag to configure a start address different than 40001 or 400001 for the Modbus hold register.

The address 0 in the received frame correspond to the start address of the hold register in the target system memory. Use the tag HR_Start_Offset to configure a start address other than 0 for the Modbus hold register.

You can, for example, configure a hold register with start at MW100 and a length of 100 words. With HR_Start_Offset = 20 the address 20 in the received frame corresponds to the start address of the hold register in the target memory (MW100). Each address in the received frame below 20 and above 119 results in an addressing error.

Table 9-149 Example for addressing the Modbus hold register when DATA_PTR is a pointer to MW100 with a length of 100 words

HR_Start_Offset	Address	Minimum	Maximum
0	Modbus address (word)	0	99
	S7-1500 address	MW100	MW298
20	Modbus address (word)	20	119
	S7-1500 address	MW100	MW298

HR_Start_Offset is a word value which specifies the start address of the Modbus hold register and is saved in the Modbus_Slave instance data block. You select this public static tag by means of the parameter drop-down list once you have added Modbus_Slave in your program.

If you have added Modbus_Slave to an LAD network, for example, you can go to a previous network and assign the value HR_Start_Offset using the move command. The value must be assigned prior to execution of Modbus_Slave.

Enter Modbus slave tag using the standard DB name:

1. Position the cursor in the OUT1 parameter field and enter the character m.
2. Select the required instance DB of the Modbus_Slave instruction from the drop-down list.
3. Position the cursor to the right of the DB name (after the quotation mark) and enter a point.
4. Select "Modbus_Slave_DB.HR_Start_Offset" in the drop-down list.

Frame structure

Extended_Addressing

You access the Extended_Addressing tag as described for the HR_Start_Offset reference, except that the Extended_Addressing tag is a Boolean value.

You can configure a single byte (Modbus standard) or two bytes (Extended_Adressing = TRUE) with Extended_Adressing = FALSE for addressing the Modbus slave. Extended addressing is used to address more than 247 devices in a single network. With Extended_Adressing = TRUE you can address up to 65535 addresses. The following example shows a Modbus frame.

Table 9-150 Slave address with one byte (Byte 0)

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	
Request	Slave address	Function code	Start address		Data		
Valid response	Slave address	Function code	Length	Data...			
Error message	Slave address	0xxx	Exception code				

Table 9-151 Slave address with two bytes (Byte 0 and Byte 1)

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Request	Slave address		Function code	Start address		Data	
Valid response	Slave address		Function code	Length	Data...		
Error message	Slave address		0xxx	Exception code			

Frame description

Data traffic between master and slave / slave and master starts with the slave address, following by the function code. The data is then transferred. The structure of the data field depends on the function code used. The checksum (CRC) is transmitted at the end of the frame.

Function code 1 - This function allows individual output bits to be read

Table 9-152 FC 1 - Read output bits

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 1	Start address		Number of outputs	
Valid response	Slave address	Function code 1	Length [1]	Output data [3)]		
Error message	Slave address	0x81	Exception code [2]	---		

[1] Length: If there is a remainder when the number of outputs is divided by 8, the number of bytes must be increased by 1.

[2] E code: 01 or 02 or 03 or 04

[3] The output data can comprise multiple bytes

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 1	Start address		Number of outputs	
Valid response	Slave address		Function code 1	Length [1]	Output data		
Error message	Slave address		0x81	Exception code [2]	---		

[1] Length: If there is a remainder when the number of outputs is divided by 8, the number of bytes must be increased by 1.

[2] E code: 01 or 02 or 03 or 04

[3] The output data can comprise multiple bytes

Function code 2 - This function allows individual input bits to be read

Table 9-153 FC 2 - Read input bits

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 2	Start address		Number of inputs	
Valid response	Slave address	Function code 2	Length [1]	Input data		
Error message	Slave address	0x82	Exception code [2]	---		

[1] Length: If there is a remainder when the number of inputs is divided by 8, the number of bytes must be increased by 1.

[2] E code: 01 or 02 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 2	Start address		Number of inputs	
Valid response	Slave address		Function code 2	Length [1]	Input data		
Error message	Slave address		0x82	Exception code [2]	---		

[1] Length: If there is a remainder when the number of inputs is divided by 8, the number of bytes must be increased by 1.

[2] E code: 01 or 02 or 03 or 04

Function code 3 - This function allows individual registers to be read

Table 9-154 FC 3 - Read hold register

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 3	Start address		Number of registers	
Valid response	Slave address	Function code 3	Length [1]	Register data		
Error message	Slave address	0x83	Exception code [2]	---		

[1] Length: Number of bytes

[2] E code: 01 or 02 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 3	Start address		Number of registers	
Valid response	Slave address		Function code 3	Length [1]	Register data		
Error message	Slave address		0x83	Exception code [2]	---		

[1] Length: Number of bytes

[2] E code: 01 or 02 or 03 or 04

Function code 4 - This function allows individual registers to be read

Table 9-155 FC 4 - Read input words

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 4	Start address		Number of input words	
Valid response	Slave address	Function code 4	Length [1]	Input data		
Error message	Slave address	0x84	Exception code [2]	---		

[1] Length: 2 * number of input words

[2] E code: 01 or 02 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 4	Start address		Number of input words	
Valid response	Slave address		Function code 4	Length [1]	Input data		
Error message	Slave address		0x84	Exception code [2]	---		

[1] Length: 2 * number of input words

[2] E code: 01 or 02 or 03 or 04

Function code 5 - This function can set or delete individual bits

Table 9-156 FC 5 - Write an output bit

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 5	Start address		Value	
Valid response	Slave address	Function code 5	Length	Value		
Error message	Slave address	0x85	Exception code [1]	---		

[1] E code: 01 or 02 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 5	Start address		Value	
Valid response	Slave address		Function code 5	Length	Value		
Error message	Slave address		0x85	Exception code [1]	---		

[1] E code: 01 or 02 or 03 or 04

Function code 6 - This function allows individual registers to be written

Table 9-157 FC 6 - Write hold register

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 6	Address		Register	
Valid response	Slave address	Function code 6	Address		Register	
Error message	Slave address	0x86	Exception code [1]	---		

[1] E code: 01 or 02 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 6	Address		Register	
Valid response	Slave address		Function code 6	Address		Register	
Error message	Slave address		0x86	Exception code [1]	---		

[1] E code: 01 or 02 or 03 or 04

Function code 8 - This function is used to check the communication connection

Table 9-158 FC 8 - Slave status

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 8	Diagnostic code		Test value	
Valid response	Slave address	Function code 8	Diagnostic code		Test value	
Error message	Slave address	0x88	Exception code [1]	---		

[1] E code: 01 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 8	Diagnostic code		Test value	
Valid response	Slave address		Function code 8	Diagnostic code		Test value	
Error message	Slave address		0x88	Exception code [1]	---		

[1] E code: 01 or 03 or 04

Function code 11 - This function can read 2 bytes of "Status word" and 2 bytes of "Event counter"

Table 9-159 FC 11 - Event counter for slave communication

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 11	---			
Valid response	Slave address	Function code 11	Status		Event counter	
Error message	Slave address	0x8B	Exception code [1]	---		

[1] E code: 01 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 11	---			
Valid response	Slave address		Function code 11	Status		Event counter	
Error message	Slave address		0x8B	Exception code [1]	---		

[1] E code: 01 or 04

Function code 15 - This function allows multiple bits to be written

Table 9-160 FC 15 - Write one/multiple output bits

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte n
Query	Slave address	Function code 15	Start address		Number of output words		Byte counter [1]	Value	
Valid response	Slave address	Function code 15	Start address		Number of output words		---		
Error message	Slave address	0x8F	Exception code [2]	---					

[1] Byte counter: If there is a remainder when the number of bytes is divided by 8, the number of bytes must be increased by 1.

[2] E code: 01 or 02 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte n
Query	Slave address		Function code 15	Start address		Number of output words		Byte counter [1]	Value	
Valid response	Slave address		Function code 15	Start address		Number of output words		---		
Error message	Slave address		0x8F	Exception code [2]		---				

[1] Byte counter: If there is a remainder when the number of bytes is divided by 8, the number of bytes must be increased by 1.

[2] E code: 01 or 02 or 03 or 04

Function code 16 - This function allows individual or multiple registers to be written

Table 9-161 FC 16 - Write one/multiple hold registers

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte n
Query	Slave address	Function code 16	Start address		Number of registers		Byte counter [1]	Value	
Valid response	Slave address	Function code 16	Start address		Number of registers		---		
Error message	Slave address	0x90	Exception code [2]	---					

[1] Byte counter: Number of registers * 2

[2] E code: 01 or 02 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte n
Query	Slave address		Function code 16	Start address		Number of registers		Byte counter [1]	Value	
Valid response	Slave address		Function code 16	Start address		Number of registers		---		
Error message	Slave address		0x90	Exception code [2]		---				

[1] Byte counter: Number of registers * 2

[2] E code: 01 or 02 or 03 or 04

Error messages

Overview of error messages - PtP

The error messages are provided at the STATUS output of an instruction and can be evaluated there or processed in the user program.

Error code	Description	Remedy
16#0000	No error	-
RECEIVE status and error codes		

Error code	Description	Remedy
16#0094	frame end identified based on the "Receipt of fixed/ maximum frame length"	-
16#0095	frame end identified based on "Message timeout"	-
16#0096	frame end identified based on expiration of the "Character delay time"	-
16#0097	The frame was aborted because the maximum response time was reached.	-
16#0098	frame end identified based on the fulfillment of the "Read message length from message" conditions	-
16#0099	frame end identified based on the receipt of the "End sequence"	-
SEND status and error codes		
16#7000	Block idle	-
16#7001	Initial call for a new frame: Data transmission initiated	-
16#7002	Interim call: Data transmission running	-
16#8085	Invalid length	Select a suitable frame length. The following are valid (depending on the module): 1-1024/2048/4096 (Byte)
16#8088	The specified length exceeds the range set in the receive buffer. Note: If the data type STRING has been specified at the BUFFER parameter, this error code also appears if the current string is shorter than the length specified at the LENGTH parameter.	Change the range in the receive buffer or select a frame length which corresponds to the range set in the receive buffer. The following are valid (depending on the module): 1-1024/2048/4096 (Byte)
16#8090	Configuration error: Odd number of bytes for WString	Select an even number of bytes.
RECEIVE status and error codes		
16#8088	The number of characters received exceeds the number specified at the BUFFER parameter.	Select a suitable frame length. The following are valid (depending on the module): 1-1024/2048/4096 (Byte)
16#8090	Configuration error: Odd number of bytes for WString	Select an even number of bytes.
Error message codes of the special functions		
16#818F	Incorrect parameter number setting (with USS only)	Select a suitable parameter number (PARAM). The following numbers are valid: 0-2047
16#8190	Incorrect setting of the CRC calculation	Select a suitable value for the CRC calculation. The following are valid: deactivated or activated. Check whether the module addressed supports CRC calculation.
16#8191	Incorrect setting of the diagnostic error interrupt	Select a suitable value for "Diagnostic error interrupt". The following are valid: Diagnostic error interrupt deactivated or diagnostic error interrupt activated. Check whether the module addressed supports the generation of diagnostic interrupts.
Error message codes of the "Port configuration"		
16#81A0	The module does not support this protocol.	Select a valid protocol for the module (PROTOCOL).
16#81A1	The module does not support this data transmission rate.	Select a valid data transmission rate for the module (BAUD).

Error code	Description	Remedy
16#81A2	The module does not support this parity setting.	Select a suitable value for "Parity" (PARITY). The following are valid: • None (1) • Even (2) • Odd (3) • Mark (4) • Space (5) • Any (6)
16#81A3	The module does not support this number of data bits.	Select a suitable value for "Number of data bits" (DATABITS). The following are valid: • 7 (2) • 8 (1)
16#81A4	The module does not support this number of stop bits.	Select a suitable value for "Number of stop bits" (STOPBITS). The following are valid: • 1 (1) • 2 (2)
16#81A5	The module does not support this type of data flow control.	Select a valid data flow control for the module (FLOWCTRL).
16#81A7	Invalid value for XON or XOFF	Select suitable values for XON (XONCHAR) and XOFF (XOFFCHAR). Valid range of values: 0...255
16#81AA	Invalid operating mode	Valid operating modes are: • Full duplex (RS232) (0) • Full duplex (RS422) four-wire mode (point-to-point) (1) / (CM PtP (ET 200SP)) • Full duplex (RS422) four-wire mode (multipoint master) (2)/ (CM PtP (ET 200SP)) • Full duplex (RS 422) four-wire mode (multipoint slave) (3) • Half duplex (RS485) two-wire mode. (4)
16#81AB	Invalid receive line initial state	Valid initial states are: • "No" default setting (0) • Signal R(A)=5 V, signal R(B)=0 V (break detection) (1): Can only be selected with: "Full duplex (RS422) four-wire operation (point-to-point connection)" and "Full duplex (RS422) four-wire operation (multipoint slave)". • Signal R(A)=0 V, signal R(B)=5 V (2): This default setting corresponds to the idle state (no active send operation).
16#81AC	Invalid value for "Break detection"	Select a suitable value for "Break detection". The following are valid: • Break detection deactivated (0) • Break detection activated (1).

Error code	Description	Remedy
16#81AF	The module does not support this protocol.	Select a valid protocol for the module.
Error codes of the "Send configuration"		
16#81B5	More than two end delimiters or end sequence > 5 characters	Select suitable values for "End delimiter" and "End sequence". The following are valid: • deactivated (0), • 1 (1) or 2 (2) end delimiters or • deactivated (0), • 1 (1) up to 5 (5) characters for the end sequence.
16#81B6	Send configuration rejected because the 3964(R) protocol was selected	Make sure that no send configuration is transmitted if the 3964(R) protocol is set.
Error codes of the "Receive configuration"		
16#81C0	Invalid start condition	Select a suitable start condition. The following are valid: • Send break before frame start • Send Idle Line.
16#81C1	Invalid end condition or no end condition selected	Select a suitable end condition (see Receiving data with Freeport (Page 991)).
16#81C3	Invalid value for "Maximum message length"	Select a suitable value for "Maximum message length" (MAXLEN). Valid range of values (depending on the module): 1-1024/2048/4096 (Byte)
16#81C4	Invalid value for "Offset of the length specification in the message"	Select a suitable value for "Offset of the length specification in the message". Valid range of values (depending on the module): 1-1024/2048/4096 (Byte)
16#81C5	Invalid value for "Size of length field"	Select a suitable value for "Size of length field" (LENGTHSIZE). Valid range of values in bytes: • 1 (1) • 2 (2) • 4 (4)
16#81C6	Invalid value for "Number of characters not counted in length specification"	Select a suitable value for "Number of characters not counted in length specification" (LENGTHM). Valid range of values: 0 to 255 (bytes)
16#81C7	The total of "Offset in the message + size of length field + number of characters not counted" is greater than the maximum frame length	Select a suitable value for "Offset in message", "Size of length field" and "Number of characters not counted". Valid range of values: • Offset in the message (depending on the module): 0-1024/2048/4096 (Byte) • Size of length field: 1, 2, or 4 (bytes) • Number of characters not counted: 0 to 255 (bytes)

Error code	Description	Remedy
16#81C8	Invalid value for "Response timeout"	Select a suitable value for "Response timeout". Valid range of values: 1-65535 (ms)
16#81C9	Invalid value for "Character delay time"	Select a suitable value for "Character delay time". Valid range of values: 1 to 65535 (bit times)
16#81CB	frame end sequence is activated, but no character is activated for the check	Activate one or several characters for the check.
16#81CC	frame start sequence is activated, but no character is activated for the check	Activate one or several characters for the check.
16#81CD	Invalid value for "Prevent overwriting"	Select a suitable value for "Prevent overwriting". The following are valid: • Prevent overwriting is deactivated (0) or • Prevent overwriting is activated (1)
16#81CE	Invalid value for "Clear receive buffer on startup"	Select a suitable value for "Clear receive buffer on startup". The following are valid: • Clear receive buffer at startup is deactivated (0) • Clear receive buffer at startup is activated (1)
SEND status and error codes		
16#81D0	Receiving send request during runtime of a send command	Make sure that you do not receive an additional send request during runtime of a send command.
16#81D1	The waiting time for XON or CTS = ON has expired.	The communication partner has a fault, is too slow or is offline. Check the communication partner or change the parameters, if necessary.
16#81D2	"Hardware RTS always ON": Send job canceled due to change from DSR = ON to OFF	Check the communication partner. Make sure that DSR is ON for the entire duration of transmission.
16#81D3	Send buffer overflow / send frame too long	Select a shorter frame length. The following are valid (depending on the module): 1 to 1024/2048/4096 (bytes)
16#81D5	Transmission canceled due to parameter changes, detected wire break, or CPU in STOP	Check the parameter assignment, wire break, and CPU status.
16#81D6	Transmission canceled because end identifier was not received	Check the parameter assignment of the end characters and the frame of the communication partner.
16#81D7	Communication error between the user program and module	Check the communication (e.g., matching the sequence number).
16#81D8	Transmission attempt rejected because module is not configured	Configure the module.
Error codes of the receive configuration		
16#81E0	Frame aborted: Send buffer overflow / send frame too long	Increase the call rate for the receive function in the user program or configure communication with data flow control.
16#81E1	Frame aborted: Parity error	Check the connection line of the communication partners, or verify that both devices are configured for the same data transmission rate, parity and stop bit number.
16#81E2	Frame aborted: Character frame error	Check the settings for start bit, data bits, parity bit, data transmission rate, and stop bit(s).

Error code	Description	Remedy
16#81E3	Frame aborted: Character overflow error	Firmware error: Please contact Customer Support.
16#81E4	Frame aborted: The total length of "Offset in the message + size of the length field + number of characters not counted" is greater than the receive buffer	Select a suitable value for offset in message, size of the length field, and number of characters not counted.
16#81E5	Frame aborted: Break	Receive line to partner is interrupted. Reconnect or switch on partner.
16#81E6	Maximum number of "Buffered received frames" exceeded	In the user program call the instruction more often or configure a communication with data flow control or increase the number of buffered frames.
16#81E7	Synchronization error module and Receive_P2P	Make sure that different instances of the Receive_P2P do not access the same module.
16#81E8	Frame aborted: The character delay time has expired before the message end criterion was detected	Partner device faulty or too slow. Check this, if required, using an interface tester that is interconnected in the transmission line.
16#81E9	Modbus CRC error (only communication modules which support Modbus)	Checksum error of the Modbus frame. Check the communication partner.
16#81EA	Modbus frame too short (only communication modules which support Modbus)	Minimum length of Modbus frame not met. Check the communication partner.
16#81EB	Frame aborted: Maximum frame length reached	Select a shorter frame length at the communication partner. The following are valid (depending on the module): 1-1024/2048/4096 (Byte) Check the parameters for end of frame detection.
Error codes V24 accompanying signals		
16#81F0	The module does not support V24 accompanying signals	You have tried to set accompanying signals for a module that does not support V24 accompanying signals. Make sure that this is an RS232 module or that RS232 mode (ET 200SP) is set.
16#81F1	No operation of the V24 accompanying signals	The V24 accompanying signals cannot be operated manually if hardware data flow control is active.
Error codes of the receive configuration		
16#8201 [1)]	Receive_Conditions is a pointer to an invalid data type	Enter a pointer to one of the following data types: DB, BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TIME_OF_DAY, TIME, S5TIME, DATE_AND_TIME, STRING
16#8225	Receive_Conditionspoints to an optimized memory area greater than 1 kB or Receive_Conditions points to an optimized memory area and the receive length is greater than the area addressed by Receive_Conditions.	Enter a pointer to an area with a maximum length of: • Optimized memory area: 1 kByte • Non-optimized area: 4 kByte Note: If the pointer points to an optimized memory area, do not send more than 1 kByte.
16#8229 [1)]	Receive_Conditions is a pointer to BOOL with a number of bits not equal to n * 8	If you are using a pointer to BOOL, the number of bits must be a multiple of 8.
Error codes, general		

Error code	Description	Remedy
16#8280	Negative acknowledgment when reading module	You can find more detailed information on error causes in theRDREC.STATUS static parameters and in the description of the SFB RDREC. • Check the input at the PORT parameter • Set the COM_RST parameter prior to the initial call.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter You can find more detailed information on error causes in the WRREC.STATUS static parameters and in the description of the SFB WRREC.
16#8282	Module not available	Check the input at the PORT parameter and ensure that the module can be reached.
Error codes of the receive configuration		
16#82C1	Invalid value for "Buffered received frames".	Select a suitable value for "Buffered received frames". Valid range of values: 1-255
16#82C2	Receive configuration rejected because the 3964(R) protocol was selected	Make sure that no receive configuration is sent if the 3964(R) protocol is set.
16#8301 1)	Receive_Conditions is a pointer to an invalid data type	Select a valid data type. The following are valid: DB, BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TIME_OF_DAY, TIME, S5TIME, DATE_AND_TIME, STRING
16#8322	Range length error when reading a parameter	Check the input at the Receive_Conditions parameter
16#8324	Range error when reading a parameter	Check the input at the Receive_Conditions parameter
16#8328	Setting error when reading a parameter	Check the input at the Receive_Conditions parameter
SEND status and error codes		
16#8328 1)	BUFFER is a pointer to BOOL with a number of bits not equal to n * 8	If you are using a pointer to BOOL, the number of bits must be a multiple of 8.
Error codes of the receive configuration		
16#8332	Invalid data block at the Receive_Conditions parameter	Check the input at the Receive_Conditions parameter
16#833A	The designation of the data block at the Receive_Conditions parameter refers to a data block which is not loaded.	Check the input at the Receive_Conditions parameter
16#8351	Invalid data type	Check the input at the Receive_Conditions parameter
16#8352 1)	Receive_Conditions does not point to a data block	Check the pointer to Receive_Conditions
16#8353 1)	Receive_Conditions does not point to a structure of the type Receive_Conditions	Check the pointer to Receive_Conditions
Error codes 3964(R) protocol		
16#8380	Parameter assignment error: Invalid value for "Character delay time".	Select a suitable value for "Character delay time" (CharacterDelayTime). Valid range of values: 1 to 65535 (ms)
16#8381	Parameter assignment error: Invalid value for "Response timeout".	Select a suitable value for "Response timeout" (AcknDelayTime). Valid range of values: 1 to 65535 (ms)

Error code	Description	Remedy
16#8382	Parameter assignment error: Invalid value for "Priority".	Select a suitable value for "Priority" (Priority). The following are valid: • High (1) • Low (0)
16#8383	Parameter assignment error: Invalid value for "Block check"	Select a suitable value for "Block check" (BCC). The following are valid: • With block check (1) • Without block check (0)
16#8384	Parameter assignment error: Invalid value for "Connection attempts".	Select a suitable value for "Connection attempts" (BuildupAttempts). Valid range of values: 1-255
16#8385	Parameter assignment error: Invalid value for "Transmission attempts".	Select a suitable value for "Transmission attempts" (RepetitionAttempts). Valid range of values: 1-255
16#8386	Runtime error: Number of connection attempts exceeded	Check the interface cable and the transmission parameters. Also check whether the receive function is configured correctly at the partner device.
16#8387	Runtime error: Number of transmission attempts exceeded	Check the interface cable, the transmission parameters and the configuration of the communication partner.
16#8388	Runtime error: Error at the "Block check character" The internally calculated value of the block check character does not correspond to the block check character received by the partner at the connection end.	Check whether the connection is seriously disrupted; in this case you may also occasionally see error codes. Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#8389	Runtime error: Invalid character received while waiting for free receive buffer	The send request of the communication partner (STX, 02H) is only answered with DLE when the receive buffer is empty. No additional character may be received before (except STX again). Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#838A	Runtime error: Logical error during receiving. After DLE was received, a further random character (other than DLE or ETX) was received.	Check if partner DLE in frame header and in data string is always in duplicate or the connection is released with DLE ETX. Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#838B	Runtime error: Character delay time exceeded	Partner device too slow or faulty. Verify by using an interface test device that is switched into the transmission line, if necessary.
16#838C	Runtime error: Wait time for free receive buffer has started	In the user program call the instruction more often or configure a communication with data flow control.
16#838D	Runtime error: frame repetition does not start within 4 s after NAK	Check the communication partner. A received frame that is possibly corrupted must be repeated by the partner within 4 seconds.

Error code	Description	Remedy
16#838E	Runtime error: In idle mode, one or several characters (other than NAK or STX) were received.	Check for proper function of the partner device, possibly using an interface test device that is switched into the transmission line.
16#838F	Runtime error: Initialization conflict - Both partners have set high priority	Set the "Low" priority at one of the partners
16#8391	Parameter assignment error: 3964 configuration data rejected because Freeport is set	If the Freeport protocol is set, make sure that no 3964 parameter assignment data is sent.

1) Only with instructions for S7-300/400 CPUs

Overview of error messages - Modbus

Error code	Description	Remedy
16#0000	No error	-
Configuration error of the interface - Modbus_Comm_Load		
16#8181	The module does not support this data transmission rate.	Select a valid data transmission rate for the module at the BAUD parameter.
16#8182	The module does not support this parity setting.	Select a suitable value for "Parity" at the PARITY parameter. The following are valid: • None (1) • Even (2) • Odd (3) • Mark (4) • Space (5) • Any (6)
16#8183	The module does not support this type of data flow control.	Select a valid data flow control for the module at the FLOW_CTRL parameter.
16#8184	Invalid value for "Response timeout".	Select a suitable value for "Response timeout" at the RESP_TO parameter. Valid range of values: 1 to 65535 (ms)
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the Send_Config.RDREC.STATUS or Receive_Config.RDREC.STATUS static parameters or RDREC.STATUS and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the Send_Config.WRREC.STATUS or Receive_Config.WRREC.STATUS static parameters or WRREC.STATUS and in the description of the SFB WRREC.
16#8282	Module not available	Check the input at the PORT parameter and ensure that the module can be reached.
Configuration error - Modbus_Slave		

Error code	Description	Remedy
16#8186	Invalid slave address	Select a suitable slave address at the MB_ADDR parameter. The following are valid: 1-247 at standard address area; 1-65535 at extended address area (0 is reserved for Broadcast)
16#8187	Invalid value at MB_HOLD_REG parameter	Select a suitable value for the hold register at the MB_HOLD_REG parameter.
16#8188	Invalid operating mode or broadcast (MB_ADDR = 0) and MODE parameter ≠ 1	Select the value 1 for MODE in Broadcast mode or select a different operating mode.
16#818C	The pointer to a MB_HOLD_REG area must be a data block or a bit memory address area.	Select a suitable value for the pointer to the MB_HOLD_REG area.
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.WRREC.STATUS or Receive_P2P.WRREC.STATUS, and in the description of the SFB WRREC.
16#8452 [1)]	MB_HOLD_REG is not a pointer to a DB or a bit memory area	Check the MB_HOLD_REG pointer
16#8453 [1)]	MB_HOLD_REG is not a pointer of type BOOL or WORD	Check the MB_HOLD_REG pointer
16#8454 [1)]	The area addressed by MB_HOLD_REG is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the MB_HOLD_REG pointer
16#8455 [1)]	MB_HOLD_REG points to a write-protected DB	Check the MB_HOLD_REG pointer
16#8456 [1)]	Error during instruction execution. The cause of the error is shown in the STATUS parameter.	Determine the value of the SFCSTATUS parameter. Check what this means in the description for SFC51, STATUS parameter.
Configuration error - Modbus_Master		
16#8180	Invalid value for MB_DB parameter	The value configured for MB_DB (instance data DB) at the Modbus_Comm_Load instruction is not valid. Check the interconnection of the Modbus_Comm_Load instruction and its error messages.
16#8186	Invalid station address	Select a suitable station address at the MB_ADDR parameter. The following are valid: 1-247 at standard address area; 1-65535 at extended address area (0 is reserved for Broadcast)
16#8188	Invalid operating mode or broadcast (MB_ADDR = 0) and MODE parameter ≠ 1	Select the value 1 for MODE in Broadcast mode or select a different operating mode.

Error code	Description	Remedy
16#8189	Invalid data address	Select a suitable value for the data address at the DATA_ADDR parameter. See description Modbus_Master (Page 5249) in the Info system
16#818A	Invalid length	Select a suitable data length at the DATA_LEN parameter. See description Modbus_Master (Page 5249) in the Info system
16#818B	Invalid value for DATA_PTR	Select a suitable value for the data pointer at the DATA_PTR parameter (M or DB address). See description Modbus_Master (Page 5249) in the Info system
16#818C	Interconnection error of the DATA_PTR parameter	Check the interconnection of the instruction.
16#818D	The area addressed by DATA_PTR is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the DATA_PTR pointer
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.WRREC.STATUS or Receive_P2P.WRREC.STATUS or Receive_Reset and in the description of the SFB WRREC.
Communication errors - Modbus_Master and Modbus_Slave		
16#80 D1	The waiting time for XON or CTS = ON has expired.	The communication partner has a fault, is too slow or is offline. Check the communication partner or change the parameters, if necessary.
16#80D2	"Hardware RTS always ON": Send job canceled due to change from DSR = ON to OFF	Check the communication partner. Make sure that DSR is ON for the entire duration of transmission.
16#80E0	Frame aborted: Send buffer overflow / send frame too long	In the user program call the instruction more often or configure a communication with data flow control.
16#80E1	Frame aborted: Parity error	Check the connection line of the communication partners, or verify that both devices are configured for the same data transmission rate, parity and stop bit number.
16#80E2	Frame aborted: Character frame error	Check the settings for start bit, data bits, parity bit, data transmission rate, and stop bit(s).
16#80E3	Frame aborted: Character overflow error	Check the number of data in the frame of the communication partner.
16#80E4	Frame aborted: Maximum frame length reached	Select a shorter frame length at the communication partner. The following are valid (depending on the module): 1 to 1024/2048/4096 (bytes)

Error code	Description	Remedy
Communication error - Modbus_Master		
16#80C8	The slave does not respond within the set time	Check the data transmission rate, parity and wiring of the slave.
16#8200	The interface is busy with an ongoing request.	Repeat the command later. Make sure that there are no commands still running before you start a new one.
Protocol error - Modbus_Slave (only communication modules that support Modbus)		
16#8380	CRC error	Checksum error of the Modbus frame. Check the communication partner.
16#8381	The function code is not supported or is not supported for broadcast.	Check the communication partner and make sure that a valid function code is sent.
16#8382	Invalid length information in the request frame	Select a suitable data length at the DATA_LEN parameter.
16#8383	Invalid data address in the request frame	Select a suitable value for the data address at the DATA_ADDR parameter.
16#8384	Invalid data value error in the request frame	Check the data value in the request frame of the Modbus master
16#8385	The diagnostic value is not supported by the Modbus slave (function code 08)	The Modbus slave only supports the diagnostic values 16#0000 and 16#000A.
Protocol error - Modbus_Master (only communication modules that support Modbus)		
16#8380	CRC error	Checksum error of the Modbus frame. Check the communication partner.
16#8381	Response frame from Modbus Slave with the following error message: The function code is not supported.	Check the communication partner and make sure that a valid function code is sent.
16#8382	Response frame from Modbus Slave with the following error message: Invalid length	Select a suitable data length.
16#8383	Response frame from Modbus Slave with the following error message: Invalid data address in the request frame	Select a suitable value for the data address at the DATA_ADDR parameter.
16#8384	Response frame from Modbus Slave with the following error message: Data value error	Check the request frame to the Modbus slave.
16#8385	Response frame from Modbus Slave with the following error message: The diagnostic value is not supported by the Modbus slave	The Modbus slave only supports the diagnostic values 16#0000 and 16#000A.
16#8386	The returned function code does not match the requested function code.	Check the response frame and the addressing of the slave.
16#8387	A slave that was not requested answers	Check the response frame of the slave. Check the address settings of the slave.
16#8388	Error in the response of the slave to a write request.	Check the response frame of the slave.
16#8828 [1)]	DATA_PTR points to a bit address that is not equal to n * 8	Check the DATA_PTR pointer
16#8852 [1)]	DATA_PTR is not a pointer to a DB or a bit memory area	Check the DATA_PTR pointer
16#8853 [1)]	DATA_PTR is not a pointer of type BOOL or WORD	Check the DATA_PTR pointer
16#8855 [1)]	DATA_PTR points to a write-protected DB	Check the DATA_PTR pointer
16#8856 [1)]	Error during call of SFC51	Call the Modbus_Master instruction again
Error - Modbus_Slave (only communication modules that support Modbus)		

Error code	Description	Remedy
16#8428 [1]	MB_HOLD_REG points to a bit address that is not equal to n * 8	Check the MB_HOLD_REG pointer
16#8452 [1]	MB_HOLD_REG is not a pointer to a DB or a bit memory area	Check the MB_HOLD_REG pointer
16#8453 [1]	MB_HOLD_REG is not a pointer of type BOOL or WORD	Check the MB_HOLD_REG pointer
16#8454 [1]	The area addressed by MB_HOLD_REG is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the MB_HOLD_REG pointer
16#8455 [1]	MB_HOLD_REG points to a write-protected DB	Check the MB_HOLD_REG pointer
16#8456 [1]	Error during call of SFC51	Call the Modbus_Slave instruction again

[1] Only with instructions for S7-300/400 CPUs

Overview of error messages - USS

Error code	Description	Remedy
16#0000	No error	-
16#8180	Length error in response of drive	Check the response frame of the drive.
16#8181	Data type error	Select a suitable data type. The following are valid: • Real • Word • Double word
16#8182	Data type error: "Double word" or "Real" may not be returned for the "Word" request.	Check the response frame of the drive.
16#8183	Data type error: "Word" may not be returned for the "Double word" or "Real" request.	Check the response frame of the drive.
16#8184	Checksum error in response of drive	Check the drive and the communication connection.
16#8185	Addressing error	Valid drive address range: 1 to 16
16#8186	Setpoint error	Valid setpoint range: -200 % to +200 %
16#8187	Incorrect drive number returned	Check the response frame of the drive.
16#8188	Invalid PZD length	Permitted PZD lengths: 2, 4, 6, 8 words
16#8189	The module does not support this data transmission rate.	Select a valid data transmission rate for the module.
16#818A	A different request for this drive is currently active.	Repeat the parameter read or write command later.
16#818B	The drive does not respond.	Check the drive.
16#818C	The drive responds with an error message to a parameter request.	Check the response frame of the drive. Check the parameter request.
16#818D	The drive responds with an access error message to a parameter request.	Check the response frame of the drive. Check the parameter request.
16#818E	The drive was not initialized.	Check the user program and make sure that the USS_Drive_Control instruction is called for this drive.

Error code	Description	Remedy
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Port_Config.RDREC.STATUS, Send_Config.RDREC.STATUS , Receive_Config.RDREC.STATUS , Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Port_Config.WRREC.STATUS, Send_Config.WRREC.STATUS , Receive_Config.WRREC.STATUS , Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB WRREC.

[1] Only with instructions for S7-300/400 CPUs

Point-to-point

PORT_CFG: Configure communication parameters dynamically

Description

The instruction "PORT_CFG" allows dynamic configuration of communications parameters for a point-to-point communications port.

You set up the original static configuration of the port in the hardware configuration. You can change this configuration by executing the "PORT_CFG" instruction. You can also use this function to save created blocks in libraries and to avoid configuration in the hardware configuration when you reuse it.

With "PORT_CFG" you can influence the following communications parameter settings:

- Parity
- Baud rate
- Number of bits per character
- Number of stop bits
- Type and properties of flow control

The changes made by the "PORT_CFG" instruction are not stored permanently on the target system.

You can transfer serial data via the electrical connections RS-232 (half and full duplex) and RS-485 (half duplex).

Parameters

The following table shows the parameters of the "PORT_CFG" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Activates the configuration change on a rising edge
PORT	Input	PORT	I, Q, M, D, L or constant	Identification of the communication port (HW-ID)
PROTOCOL	Input	UINT	I, Q, M, D, L or constant	Transmission protocol: • 0: Point-to-point communication protocol • 1..n: Future definition for specific transmission protocols
BAUD	Input	UINT	I, Q, M, D, L or constant	Baud rate of the port: • 1: 300 baud • 2: 600 baud • 3: 1200 baud • 4: 2400 baud • 5: 4800 baud • 6: 9600 baud (default) • 7: 19200 baud • 8: 38400 baud • 9: 57600 baud • 10: 76800 baud • 11: 115200 baud
PARITY	Input	UINT	I, Q, M, D, L or constant	Parity of the port: • 1: No parity (default) • 2: Even parity • 3: Odd parity • 4: Mark parity • 5: Space parity
DATABITS	Input	UINT	I, Q, M, D, L or constant	Bits per character: • 1: 8 bits per character (default) • 2: 7 bits per character
STOPBITS	Input	UINT	I, Q, M, D, L or constant	Number of stop bits: • 1: 1 stop bit (default) • 2: 2 stop bits
FLOWCTRL	Input	UINT	I, Q, M, D, L or constant	Data flow control: • 1: None (default) • 2: XON/XOFF • 3: Hardware flow control (RTS always activated) • 4: Hardware flow control (RTS can be deactivated during transmission)
XONCHAR	Input	CHAR	I, Q, M, D, L or constant	Indicates the character used as XON character. The character DC1 (11H) is set as default.

Parameter	Declaration	Data type	Memory area	Description
XOFFCHAR	Input	CHAR	I, Q, M, D, L or constant	Indicates the character used as XOFF character. The character DC3 (13H) is set as default.
WAITTIME	Input	UINT	I, Q, M, D, L or constant	Specifies the wait time for XON or CTS after the start of the transmission. The specified value must be greater than 0. 2000 milliseconds are set as default.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or is still executing • 1: Job executed without errors
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

STATUS parameter

Error code* (W#16#...)	Description
80A0	The specified protocol is invalid.
80A1	The specified baud rate is invalid.
80A2	The specified parity rate is invalid.
80A3	The specified number of bits per character is invalid.
80A4	The specified number of stop bits is invalid.
80A5	The specified type of flow control is invalid.
80A6	Incorrect value at the WAITTIME parameter When the data flow control is enabled, the value at the WAITTIME parameter must be greater than zero.
80A7	Invalid values at XONCHAR and XOFFCHAR parameters.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

You will find more detailed information on general error codes of the communication instructions in: "General status information of the communications blocks (Page 5295)".

See also

Switching display formats in the program status (Page 2980)

SEND_CFG: Configure serial transmission parameters dynamically

Description

The instruction "SEND_CFG" allows dynamic configuration of serial transmission parameters for a point-to-point communications port. All the messages waiting for transfer are discarded after execution of SEND_CFG.

You set up the original static configuration of the port in the hardware configuration. You can change this configuration by executing the "SEND_CFG" instruction. You can also use this function to save created blocks in libraries and to avoid configuration in the hardware configuration when you reuse it. With "SEND_CFG" you can influence the following transmission parameter settings:

- Time between the activation of RTS (Request to Send) and the start of the transmission
- Time between the end of transmission and the deactivation of RTS
- Define bit times for breaks

The changes made by the "SEND_CFG" instruction are not stored permanently on the target system.

You can transfer serial data via the electrical connections RS-232 (half and full duplex) and RS-485 (half duplex).

Parameters

The following table shows the parameters of the "SEND_CFG" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Activates the configuration change on a rising edge
PORT	Input	PORT	I, Q, M, D, L or constant	Identification of the communication port (HW-ID)
RTSONDLY	Input	UINT	I, Q, M, D, L or constant	The time that should elapse after activating RTS until the start of transmission. Valid values for this parameter are as follows: • 0 (default) • 0 to 65535 ms in steps of 1 ms This parameter does not apply to RS-485 modules.
RTSOFFDLY	Input	UINT	I, Q, M, D, L or constant	Time that should elapse after the end of transmission until deactivation of RTS. Valid values for this parameter are as follows: • 0 (default) • 0 to 65535 ms in steps of 1 ms This parameter does not apply to RS-485 modules.
BREAK	Input	UINT	I, Q, M, D, L or constant	Specifies the bit times for a break, which are sent at the start of the message. 12 bit times are set as default. A maximum of 25000 bit times can be specified.

Parameter	Declaration	Data type	Memory area	Description
IDLELINE	Input	UINT	I, Q, M, D, L or constant	Specifies the bit times for idle line after the break at the start of the message. 12 bit times are set as default. A maximum of 25000 bit times can be specified.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or is still executing • 1: Job executed without errors
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

STATUS parameter

Error code* (W#16#...)	Description
80B0	The configuration of a transmission interruption is not permitted.
80B1	The specified break time exceeds the permitted maximum of 25000 bit times.
80B2	The specified time for idle line exceeds the permitted maximum of 25000 bit times.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

You will find more detailed information on general error codes of the communication instructions in: "General status information of the communications blocks (Page 5295)".

See also

Switching display formats in the program status (Page 2980)

RCV_CFG: Configure serial receive parameters dynamically

Description

The instruction "RCV_CFG" allows dynamic configuration of serial receive parameters for a point-to-point communications port. You can use this instruction to configure the conditions that specify the start and end of the message to be transmitted. The receipt of messages that correspond to these conditions can be enabled by the "RCV_PTP (Page 5290)" instruction.

You set up the original static configuration of the port in the properties of the hardware configuration. Execute the "RCV_CFG" instruction in your program to change the configuration. You can also use this function to save created blocks in libraries and to avoid configuration in the hardware configuration when you reuse it. The changes made by the "RCV_CFG" instruction are not stored permanently on the target system.

All the messages waiting for transfer are discarded after execution of the "RCV_CFG" instruction.

Parameters

The following table shows the parameters of the "RCV_CFG" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Activates the configuration change on a rising edge.
PORT	Input	PORT	I, Q, M, D, L or constant	Identification of the communication port (HW-ID)
CONDITIONS	Input	CONDITIONS	D, L	Data structure defining the conditions for start and end of data transmission.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or is still executing • 1: Job executed without errors
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

Data type CONDITIONS

You can use the CONDITIONS structure to define the start and end conditions for the message transmission. The structure CONDITIONS is included in the instance DB of the "RCV_CFG" instruction. Use the structure CONDITIONS to define the start and end conditions when the transmission of a message is complete and when the next message transfer is to start:

- You define the start conditions for the data transfer in the START structure.
- You define the end conditions for the data transfer in the END structure.

You can define one or more start and end conditions for this. If you specify multiple start or end conditions these are linked by an OR logic instruction.

The following table shows the "CONDITIONS" structure:

Parameters		Data type	Description
START		STRUCT	Start conditions
	STARTCOND	UINT	Specifies the start condition (details, see below). The start condition can be specified as a 16-bit hexadecimal value. Possible values for the start condition are: • 1: Start character • 2: Any character (default) • 4: Line break • 8: Idle line • 16: Character string 1 • 32: Character string 2 • 64: Character string 3 • 128: Character string 4 Multiple start conditions can also be defined at the STARTCOND parameter. The total from the values of the individual conditions is specified for this. If, for example, you want to define "Idle line" OR "Character string 1" OR "Character string 4" as start condition, the value "152" must be specified.
	IDLETIME	UINT	Specifies the maximum idle time of the line before receipt is started. Valid values for this parameter are as follows: • 40 bit times (default) • 0 to 2500 bit times
	STARTCHAR	BYTE	Specifies the start character. This setting is only enabled when the configured start condition is "Start character". Valid values for this parameter are as follows: • 02 (STX): Default setting • B#16#00 to B#16#FF
	SEQ[1].CTL	BYTE	Character string 1: Control sequence for each character You can use the bit position of the character to define which characters of the character string will be considered or ignored. To evaluate the characters, the corresponding bits have to be set. • Bit 0: 1 character • Bit 1: 2 characters • Bit 2: 3 characters • Bit 3: 4 characters • Bit 4: 5 characters A character is ignored when the corresponding bit is reset.
	SEQ[1].STR	CHAR[5]	Character string 1: Start character (5 characters)
	SEQ[2].CTL	BYTE	Character string 2: Ignore/compare control sequence for each character
	SEQ[2].STR	CHAR[5]	Character string 2: Start character (5 characters)
	SEQ[3].CTL	BYTE	Character string 3: Ignore/compare control sequence for each character
	SEQ[3].STR	CHAR[5]	Character string 3: Start character (5 characters)
	SEQ[4].CTL	BYTE	Character string 4: Ignore/compare control sequence for each character
	SEQ[4].STR	CHAR[5]	Character string 4: Start character (5 characters)
END		STRUCT	End conditions

Parameters		Data type	Description
	ENDCOND	UINT	Specifies the end condition (details, see below). The end condition can be specified as a 16-bit hexadecimal value. Possible values for the end condition are: • 1: Reply timeout • 2: Message timeout • 4: Timeout within the character string • 8: Maximum length • 16: N+LEN+M; the information on the message length in integrated in the message and will be evaluated. • 32: Character string 1 Multiple end conditions can also be defined at the ENDCOND parameter. The total from the values of the individual end conditions is specified for this. If, for example, you want to define the end condition "Maximum length" OR "Sequence 1", the value "40" must be specified.
	MAXLEN	UINT	Specifies the maximum number of characters in a message. Valid values* for this parameter are as follows: • 1 character (default) • 0 to 1024 characters This setting is only enabled if the "Maximum length" end condition is set at the ENDCOND parameter.
	N	UINT	Offset of the length field in the message Valid values for this parameter are as follows: • 0 characters (default) • 0 to 1024 characters This setting is only enabled if the "N+LEN+M" end condition is set at the ENDCOND parameter.
	LENGTHSIZE	UINT	Size of the length field in bytes Valid values* for this parameter are as follows: • 0 bytes (default) • 1 byte • 2 bytes • 4 bytes This setting is only enabled if the "N+LEN+M" end condition is set at the ENDCOND parameter.
	LENGTHM	UINT	Specifies the number of end characters that follow the length field but are not contained in the length of the message. Valid values for this parameter are as follows: • 0 characters (default) • 0 to 255 characters This setting is only enabled if the "N+LEN+M" end condition is set at the ENDCOND parameter.

Parameters		Data type	Description
	RCVTIME	UINT	Specifies the maximum duration for the receipt of the first character of a message. Valid values for this parameter are as follows: • 200 ms (default) • 0 to 65535 ms in steps of 1 ms This setting is only enabled if the "Reply timeout" end condition is set at the ENDCOND parameter.
	MSGTIME	UINT	Specifies the maximum duration of the receipt of a message. Valid values for this parameter are as follows: • 200 ms (default) • 0 to 65535 ms in steps of 1 ms This setting is only enabled if the "Message timeout" end condition is set at the ENDCOND parameter.
	CHARGAP	UINT	Specifies the time interval between received consecutive characters. Valid values for this parameter are as follows: • 12 bit times (default) • 0 to 2500 bit times This setting is only enabled if the "Timeout within the character string" end condition is set at the ENDCOND parameter.
	SEQ.CTL	BYTE	Character string: Control sequence for each character You can use the bit position of the character to define which characters of the character string will be considered or ignored. To evaluate the characters, the corresponding bits have to be set. • Bit 0: 1 character • Bit 1: 2 characters • Bit 2: 3 characters • Bit 3: 4 characters • Bit 4: 5 characters A character is ignored when the corresponding bit is reset.
	SEQ.STR	CHAR[5]	Character string: Start character (5 characters)
	* These value ranges also apply to the corresponding hardware settings for specifying the end of message.		

Start conditions for the message receipt (STARTCOND parameter)

The start of the message is recognized by the receiver if a configured start condition applies. The following conditions can be defined as start conditions for message receipt:

- Start character: The start of a message is recognized when a certain character occurs. This character is stored as first character of the message. All characters received before the start character are rejected.
- Any character: Any character can define the start of a message. This character is stored as first character of the message.
- Line break: The start of a message is recognized if the received data stream is interrupted for longer than one character length.

- Idle line: The start of a message is recognized when the send transmission line is in the idle state for a certain time (specified in bit times) followed by renewed transmission of characters.
- Character string (sequence): The start of a message is recognized when a specified character sequence occurs in the data stream. You can specify up to four character sequences with up to five characters each.
  Example: A received hexadecimal message includes the following characters: "68 10 aa 68 bb 10 aa 16". The configured start character sequences are listed in the following table. Start character sequences will be evaluated once the first character 68H has been received successfully. After the fourth character has been received successfully (the second 68H), the start condition "1" has been met. Once the start conditions have been met, evaluation of the end conditions will start.
  Processing of the start character sequence can end due to different errors in parity, framing or time intervals between characters. These errors will prevent reception of the message, because the start condition has not been met.

Start condition	First character	First character +1	First character +2	First character +3	First character +4
1	68H	xx	xx	68H	xx
2	10H	aaH	xx	xx	xx
3	dcH	aaH	xx	xx	xx
4	e5H	xx	xx	xx	xx

End conditions for the message receipt (ENDCOND parameter)

The start of a message is recognized by the receiver if a configured end condition applies. The following conditions can be defined as end conditions for message receipt:

- Reply timeout: The receipt of messages will end when the specified maximum duration for the receipt of a character is exceeded. The maximum duration is defined at the RCVTIME parameter. The defined time starts to run down as soon at the last transmission is completed and the RCV_PTP instruction enables the receipt of the message. If no character was received within the defined time (RCVTIME), the RCV_PTP instruction reports an error.
- Message timeout: The receipt of messages will end when the specified maximum duration for the receipt of a message is exceeded. The maximum duration is defined at the MSGTIME parameter. The defined time starts to run down as soon as the first character of the message is received.
- Timeout within the character string: The receipt of messages will end when the time interval between the receipt of two consecutive characters is longer than the value at the CHARGAP parameter.
- Maximum length: The receipt of messages will end when the length of the message defined at the MAXLEN parameter is exceeded.

- Reading message length (N+LEN+M): The receipt of messages will end when a certain message length is reached. This length is calculated by the values of the following parameters:
  - N: Position of the character in the message from which the length field begins.
  - LENGTHSIZE: Size of the length field in bytes
  - LENGTHM: Number of end characters that follow the length field. These characters are not taken into account in the evaluation of the message length.
- Character string: The receipt of messages will end when a defined character sequence is received. The character string can contain a maximum of five characters. For each character of the character string, you can use the bit position to define if this will be considered or ignored in the evaluation.

STATUS parameter

Error code* (W#16#...)	Description
80C0	Error in start condition
80C1	• Error in end condition • No end condition defined
80C2	Receive interrupt enabled
80C3	A value that is equal to 0 or greater than 4132 was entered at the MAXLEN parameter while the "Maximum length" end condition was set.
80C4	A value that is greater than 4131 was entered at the N parameter while the "N+LEN+M" end condition was set.
80C5	A value that is equal to 0 or invalid was entered at the LENGTHSIZE parameter while the "N+LEN+M" end condition was set.
80C6	A value that is greater than 255 was entered at the LENGTHM parameter while the "N+LEN+M" end condition was set.
80C7	A message length greater than 4132 was calculated while the "N+LEN+M" end condition was set.
80C8	A value that is equal to 0 was entered at the RCVTIME parameter while the "Reply timeout" end condition was set.
80C9	A value that is equal to 0 or greater than 2500 was entered at the CHARGAP parameter while the "Timeout within a character string" end condition was set.
80CA	A value that is equal to 0 or greater than 2500 was entered at the IDLETIME parameter while the "Idle line" start condition was set.
80CB	All characters of the character string are marked as "Don't care" even though "Character string" is set as the end condition.
80CC	All characters of the character string are marked as "Don't care" even though "Character string" is set as the start condition.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

You will find more detailed information on general error codes of the communication instructions in: "General status information of the communications blocks (Page 5295)".

See also

Switching display formats in the program status (Page 2980)

SEND_PTP: Transmit send buffer data

Description

You use the "SEND_PTP" instruction to start the transmission of data. The "SEND_PTP" instruction does not execute the actual transmission of the data. The data of the send buffer is transmitted to the relevant point-to-point communication module (CM). The CM executes the actual transmission.

Parameters

The following table shows the parameters of the "SEND_PTP" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Enables the requested transmission on a rising edge of this enable input. The content of buffer is transmitted to the point-to-point communication module (CM).
PORT	Input	PORT	I, Q, M, D, L or constant	Identification of the communication port (HW-ID)
BUFFER	Input	VARIANT	I, Q, M, D, L or constant	Pointer to the start address of the send buffer. Boolean values or Array of BOOL are not supported.
LENGTH	Input	UINT	I, Q, M, D, L or constant	Length of the send buffer
PTRCL	Input	BOOL	I, Q, M, D, L or constant	This parameter selects the buffer for normal point-to-point communication or for specific Siemens protocols implemented in the connected CM. FALSE = point-to-point operations controlled by the user program (only valid option)
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or is still executing • 1: Job executed without errors
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

STATUS parameter

Error code* (W#16#...)	Description
7000	The send operation is not active.
7001	The send operation is processing the first call.
7002	The send operation is processing subsequent calls (queries following the first call).
8080	The identifier entered for the communications port number is invalid.
8088	The length of the LENGHT parameter does not correspond to the length of data to be sent. See also: Parameters LENGHT and BUFFER.
80D0	A new send request was received while a transmission was taking place.
80D1	The transmission was interrupted because the CTS signal was not confirmed within the specified wait time.
80D2	The send request was interrupted because the communications partner (DCE) signaled that it was not willing to receive (DSR).
80D3	The send request was interrupted because the maximum size of the waiting loop was exceeded (more than 1024 Byte).
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

You will find more detailed information on general error codes of the communication instructions in: "General status information of the communications blocks (Page 5295)".

Parameters LENGTH and BUFFER

The minimum data size that can be sent by the "PTP_SEND" instruction is one byte. The parameter BUFFER defines the size of the data to be sent. You can use neither the BOOL nor the Array of BOOL data type for the BUFFER parameter.

LENGTH parameter	BUFFER parameter	Description
LENGTH = 0	Not used	The complete data is sent as defined by the BUFFER parameter. If LENGTH = 0, you do not need to specify the number of bytes transferred.
LENGTH > 0	Elementary data type	The LENGTH value must contain the byte count of this data type. Otherwise, data is not transferred and error 8088 is output.
	STRUCT	The LENGTH value can contain a byte count that is smaller than the complete byte length of the structure. In this case, only the first LENGTH bytes are transferred.

LENGTH parameter	BUFFER parameter	Description
	ARRAY	The LENGTH value can contain a byte count that is smaller than the complete byte length of the field. In this case, only the field elements that fit completely in the LENGTH bytes are transferred. The LENGTH value must be a multiple of the byte count of the data elements. Otherwise, STATUS = 8088, ERROR = 1, and no data is transferred.
	STRING	The complete memory arrangement of the character sequence format will be transmitted as well as the information about maximum length of the character string and the actual length of the character string. The LENGTH value must contain bytes for maximum length, actual length, and the characters of the character string. With the data type STRING, all lengths and characters have the size of one byte. If a character string is used for the BUFFER parameter, the LENGTH value must also contain two bytes for the two length fields.

See also

Switching display formats in the program status (Page 2980)

RCV_PTP: Enable receive messages

Description

With the RCV_PTP instruction you enable receipt of a sent message. Each message must be enabled individually. The sent data is only available in the receive area when the message has been acknowledged by the relevant communications partner.

Parameters

The following table shows the parameters of the "RCV_PTP" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L or constant	Enables receipt on a rising edge
PORT	Input	PORT	I, Q, M, D, L or constant	Identification of the communication port (HW-ID)
BUFFER	Input	VARIANT	I, Q, M, D, L or constant	Points to the start address of the receive buffer. Do not use a tag of the type STRING in the receive buffer.
NDR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or is still executing • 1: Job executed without errors
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred

Parameter	Declaration	Data type	Memory area	Description
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction
LENGTH	Output	UINT	I, Q, M, D, L	Length of the message in the receive buffer

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

STATUS parameter

Error code* (W#16#....)	Description
80E0	Receipt of messages was terminated because the receive buffer is full.
80E1	Receipt of messages was terminated as a result of a parity error.
80E2	Receipt of messages was terminated as a result of a framing error.
80E3	Receipt of messages was terminated as a result of an overflow error.
80E4	Receipt of messages was terminated because the calculated message length (N+LEN+M) exceeds the size of the receive buffer.
8080	The identifier entered for the communications port number is invalid.
8088	A data type STRING is referenced via the BUFFER parameter.
0094	Receipt of messages was terminated because the maximum character length was received.
0095	Receipt of messages was terminated as a result of a timeout.
0096	Receipt of messages was terminated because of a timeout within the character string.
0097	Receipt of messages was terminated as a result of a reply timeout.
0098	Receipt of messages was terminated because the "N+LEN+M" length condition has been satisfied.
0099	Receipt of messages was terminated because the character string defined as the end condition was received.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

You will find more detailed information on general error codes of the communication instructions in: "General status information of the communications blocks (Page 5295)".

See also

Switching display formats in the program status (Page 2980)

RCV_RST: Delete receive buffer

Description

With the "RCV_RST" instruction, you delete the receive buffer of a communications partner.

Parameters

The following table shows the parameters of the "RCV_RST" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Enables deleting of the receive buffer on a rising edge
PORT	Input	PORT	I, Q, M, D, L or constant	Identification of the communication port (HW-ID)
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or is still executing • 1: Job executed without errors
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction You will find more detailed information on general error codes of the communication instructions in: "General status information of the communications blocks (Page 5295)".

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

SGN_GET: Query RS-232 signals

Description

With the "SGN_GET" instruction, you query the current state of several signals of an RS-232 communications module.

Parameters

The following table shows the parameters of the "SGN_GET" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Enables the query on a rising edge
PORT	Input	PORT	I, Q, M, D, L or constant	Identification of the communication port (HW-ID)
NDR	Output	BOOL	I, Q, M, D, L	Is set for one cycle if new data are ready for sending and the instruction was executed error-free.
DTR	Output	BOOL	I, Q, M, D, L	Data terminal ready, module ready
DSR	Output	BOOL	I, Q, M, D, L	Data set ready, communications partner ready
RTS	Output	BOOL	I, Q, M, D, L	Send request, module ready to send

Parameter	Declaration	Data type	Memory area	Description
CTS	Output	BOOL	I, Q, M, D, L	Clear to send, communications partner can receive data (reaction to RTS = ON of the module).
DCD	Output	BOOL	I, Q, M, D, L	Data carrier detect, received signal level
RING	Output	BOOL	I, Q, M, D, L	Ring display, display of an incoming call
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

STATUS parameter

Error code* (W#16#....)	Description
80F0	The communication module is an RS-485 module and no signals are available.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

You will find more detailed information on general error codes of the communication instructions in: "General status information of the communications blocks (Page 5295)".

See also

Switching display formats in the program status (Page 2980)

SGN_SET: Set RS-232 signals

Description

With the "SGN_SET" instruction, you set the status of the output signals of an RS-232 communications module.

Parameters

The following table shows the parameters of the "SGN_SET" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Activates the action on a rising edge Initial value: FALSE
PORT	Input	PORT	I, Q, M, D, L or constant	Identification of the communication port (HW-ID) Initial value: FALSE

Parameter	Declaration	Data type	Memory area	Description
SIGNAL	Input	BYTE	I, Q, M, D, L or constant	Specifies the signals to be set: • Set 01H = RTS • Set 02H = DTR • Set 04H = DSR Initial value: FALSE
RTS	Input	BOOL	I, Q, M, D, L or constant	Send request, module ready to send Initial value: FALSE
DTR	Input	BOOL	I, Q, M, D, L or constant	Data terminal ready, module ready Initial value: FALSE
DSR	Input	BOOL	I, Q, M, D, L or constant	Data set ready (applies only to interfaces of the DCE type) Initial value: FALSE
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or is still executing • 1: Job executed without errors Initial value: FALSE
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: No error • 1: Error occurred Initial value: FALSE
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction Initial value: 0

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

STATUS parameter

Error code* (W#16#...)	Description
80F0	The communication module is an RS-485 module and no signals are available.
80F1	No signals are settable because H/W flow control is enabled.
80F2	The DSR signal cannot be set because the module is a DTE device.
80F3	The DTR signal cannot be set because the module is a DCE device.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

You will find more detailed information on general error codes of the communication instructions in: "General status information of the communications blocks (Page 5295)".

See also

Switching display formats in the program status (Page 2980)

General status information of the communications blocks

General information on execution status of the communications blocks

The following table shows which general information can be output at the STATUS parameter of the communications blocks:

Error code* (W#16#....)	Description
8070	All internal instance memory is in use
8080	The identifier entered for the communications port is invalid
8081	Timeout, module error, internal error
8085	Error specifying the length at the LENGHT parameter. The specified length is "0" or greater than the maximum permitted value.
8090	Message length invalid, module invalid, message invalid
8091	Incorrect version in parameterization message
8092	Invalid record length in parameterization message
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Switching display formats in the program status (Page 2980)

USS

Overview of USS instructions

Introduction

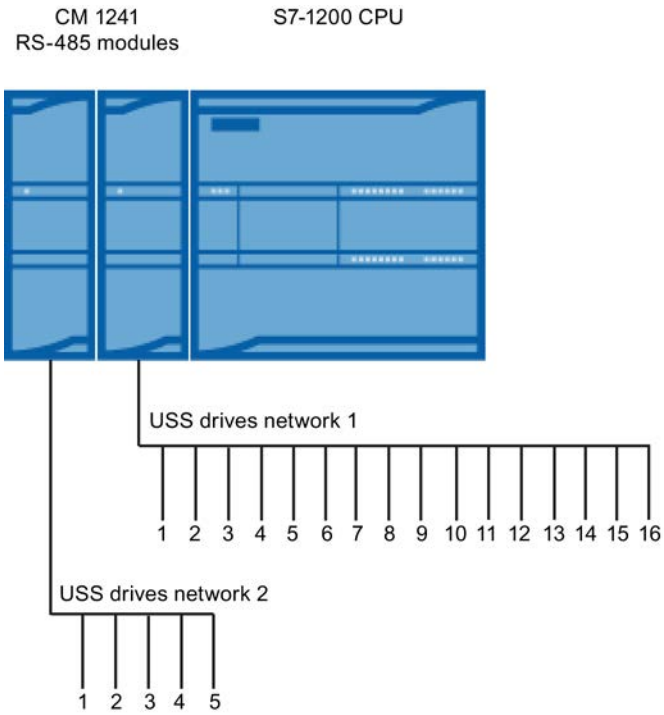
The USS instructions control the operation of drives that support the universal serial interface (USS). With the USS instructions, you can communicate with more than one drive via an RS-485 connection.

To do this, you require a CM 1241 RS-485 communications module or a CB 1241 RS-485 communications board. Up to three CM 1241 RS-485 modules and one CB 1241 RS-485 board can be installed in an S7-1200 CPU.

Each RS-485 port can operate up to sixteen drives.

The USS protocol uses a master/slave network for communication via a serial bus. The master uses an address parameter to send a message to a selected slave. A slave itself can never send without previously receiving a request. Direct exchange of messages between slaves is not possible. USS communication works in half duplex mode.

The following figure shows an example of a USS network diagram:



Requirements for using the USS protocol

General requirements for setting up a drive

- The use of 4 PKW words must be set up for the drives.
- The drives can be configured for 2, 4, 6 or 8 PZD words.
- The number of PZD words in the drive must correspond to the PZD_LEN input of the "USS_DRIVE (Page 5300)" instruction of the drive.
- The baud rate of all drives must match the baud rate of the BAUD input parameter of the "USS_PORT (Page 5299)" instruction.
- The drive must be set up for remote control.
- USS must be specified for the desired frequency value on the COM connection of the drive.
- 1 to 16 must be set as the drive address. This address must correspond to the address of the DRIVE input parameter of the "USS_DRIVE (Page 5300)" instruction.
- To control the direction of the drive, the polarity of the drive desired value must be set up.
- The RS-485 network must be connected correctly.

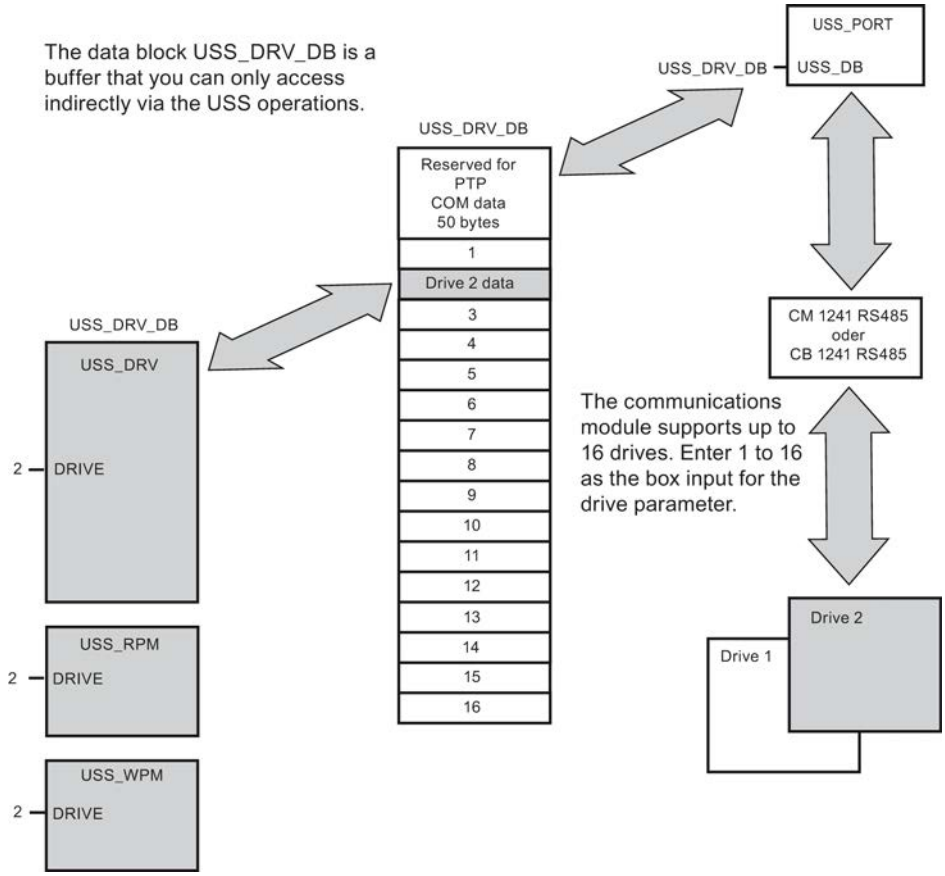
Definition: PKW / PZD area

- The PKW area relates to the handling of the parameter-identifier-value interface.
  The PKW interface is not a physical interface but describes a mechanism that controls the exchange of parameters between two communications partners, in other words, reading and writing parameter values, parameter descriptions and corresponding texts and the handling of parameter changes due to spontaneous messages. All the tasks handled via the PKW interface are essentially tasks for operator control and monitoring, service and diagnostics.
- The PZD area includes the signals required for automation:
  - Control word(s), and setpoint(s) from the master to the slave
  - Status word(s), and actual value(s) from the slave to the master.

Both areas together result in the user data field. This is transferred by the master to the slave as a job frame or by the slave to the master as a reply frame.

Description

Each communication module CM 1241 RS485 supports a maximum of 16 drives. A single instance data block contains temporary memory and buffer functions for all drives in the USS network that are connected to a PtP communications module you installed. The USS instructions for these drives have shared access to the information in this data block.



- All drives (max. 16) that are connected to an RS485 port are part of the same USS network. All drives that are connected to a different RS485 port are part of a different USS network. As the S7-1200 supports up to three CM 1241 RS485 modules, you can set up up to three USS networks, each having a maximum of 16 drives, thus a total of 48 USS drives are supported.
- Each USS network is managed via a unique data block (three data blocks are required for three USS networks with three CM 1241 RS485 modules). All instructions that belong to a USS network must share this data block. This includes all "USS_DRIVE (Page 5300)", "USS_PORT (Page 5299)", "USS_RPM (Page 5303)", and "USS_WPM (Page 5304)" instructions for controlling all drives in a USS network.
- The instruction "USS_DRIVE (Page 5300)" is a function block (FB). When you insert the instruction "USS_DRIVE" in the editor, the "Call options" dialog will ask you to assign a DB to the instruction.
  - If this is the first "USS_DRIVE" instruction in this program for this USS network, you can accept the default DB assignment (or, if necessary, change the name) and the new DB will be created.
  - If, however, this is not the first "USS_DRIVE" instruction for this network, you must select the DB that was previously assigned to this USS network in the drop-down list in the "Call options" dialog.
- All "USS_PORT (Page 5299), USS_RPM (Page 5303)", and "USS_WPM (Page 5304)" instructions are functions (FCs). When you insert these FCs in the editor, no DB is assigned. Instead, you have to assign the DB in question to the USS_DB input of these instructions (double-click on the parameter field and then on the icon to display the available DBs).
- The instruction "USS_PORT (Page 5299)" controls communication between the CPU and the drives via the PtP communications module. Whenever this instruction is called, communication with a drive is processed. Your program must call this function quickly enough so that the drives do not report a timeout. The instruction can be called from the main program or any interrupt OB.
- The function block "USS_DRIVE (Page 5300)" gives your program access to a specified drive in the USS network. Its inputs and outputs correspond to the states and operating functions of the drive. If there are 16 drives in the network, "USS_DRIVE" must be called at least 16 times in your program, i.e., once for each drive. How quickly these blocks are called depends on the required speed for controlling drive functions.
  You can only call the instruction "USS_DRIVE" from the main program OB.

⚠ CAUTION
Call "USS_DRIVE", "USS_RPM", "USS_WPM" only from the main program OB. The instruction "USS_PORT" can be called from any OB, it is usually called from a time-delay interrupt OB. If the instruction "USS_PORT" is interrupted during execution, this may result in unexpected errors.

The "USS_RPM" and "USS_WPM" instructions are used to read and write the operating parameters of the drive. These parameters control the internal mode of operation of the drive. A definition of these parameters can be found in the drive manual.

Your program may also contain any number of these instructions; however only one read or write request can be active for a drive. You may only call the instructions "USS_RPM" and "USS_WPM" from the main program OB.

Calculating the time for communication with the drive

Communication with the drive is runs asynchronous to the cycle of the S7-1200. The S7-1200 runs through several cycles before communication with a drive is completed.

The interval of "USS_PORT" is the time required for a drive transaction. The following table shows the minimum intervals for "USS_PORT" for each baud rate. Calling the "USS_PORT" more frequently than the "USS_PORT" interval will not increase the number of transactions. The timeout interval of the drive is the period of time available to a transaction if 3 attempts are required to complete the transaction due to communications errors. By default, up to 2 further attempts are made for each transaction with the USS protocol.

Baud rate	Calculated minimum interval for calling USS_PORT (ms)	Drive message interval timeout per drive (ms)
1200	790	2370
2400	405	1215
4800	212.5	638
9600	116.3	349
19200	68.2	205
38400	44.1	133
57600	36.1	109
115200	28.1	85

USS_PORT: Edit communication via USS network

Description

The "USS_PORT" instruction handles communication over the USS network. In the program, use one "USS_PORT" instruction per PtP communications port to control the transmission to or from one drive.

All USS instructions that are assigned to one USS network and one PtP communications port must use the same instance data block.

Call

Your program must execute the "USS_PORT" instruction often enough to prevent timeouts in the drive. You should therefore call the "USS_PORT" instruction from a cyclic interrupt OB to prevent drive timeouts and keep the most recent USS data updates available for "USS_DRIVE (Page 5300)" calls.

Parameters

The following table shows the parameters of the "USS_PORT" instruction:

Parameter	Declaration	Data type	Memory area	Description
PORT	Input	PORT	D, L or constant	PtP communications port identifier Constant that can be referenced within the "Constants" tab of the default tag table.
BAUD	Input	DINT	I, Q, M, D, L or constant	Baud rate for USS communication.
USS_DB	InOut	USS_BASE	D	Reference to the instance DB of the "USS_DRIVE (Page 5300)" instruction.
ERROR	Output	BOOL	I, Q, M, D, L	ERROR is set to TRUE if an error occurs. A corresponding error code will be output at the STATUS output.
STATUS (Page 5305)	Output	WORD	I, Q, M, D, L	Status value of the request. It indicates the result of the cycle or initialization. Additional information is available in the "USS_Extended_Error (Page 5305)" tag for some status codes.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

USS_DRIVE: Swap data with drive

Description

The "USS_DRIVE" instruction exchanges data with the drive by creating request messages and interpreting the drive response messages. A separate instruction must be used for each drive, but all USS instructions assigned to one USS network and one PtP communications module must use the same instance data block. You must create the DB name when you place the first "USS_DRIVE" instruction. Then reuse this DB that was created when the initial instruction was inserted.

When the "USS_DRIVE" instruction is executed the first time, the drive indicated by the USS address (parameter DRIVE) is initialized in the instance DB. After this initialization, subsequent "USS_PORT (Page 5299)" instructions can start communication with the drive at this drive number.

Changing the drive number requires a PLC STOP to RUN mode transition that initializes the instance DB. Input parameters are configured in the USS send buffer, and outputs are read from a "previous" valid response buffer if any exists. There is no data transmission during execution of the "USS_DRIVE" instruction. Communication with the drives takes place when "USS_PORT (Page 5299)" is executed. "USS_DRIVE" only configures the messages to be sent and interprets data received in a previous request.

You can control the drive direction of rotation using either the DIR (BOOL) input or using the sign (positive or negative) at the SPEED_SP (REAL) input. The following table explains how these inputs work together to determine the drive direction, assuming the motor is wired for forward rotation.

SPEED_SP	DIR	Direction of rotation of drive
Value > 0	0	Reverse
Value > 0	1	Forward
Value < 0	0	Forward
Value < 0	1	Reverse

Parameters

Expand the box to display all the parameters by clicking the bottom of the box. The parameter connections that are grayed are optional and do not need to be assigned.

The following table shows the parameters of the "USS_DRIVE" instruction:

Parameter	Declaration	Data type	Memory area	Description
RUN	Input	BOOL	I, Q, M, D, L or constant	Drive start bit: If this parameter has the value TRUE, this input enables the drive to run at the preset speed.
OFF2	Input	BOOL	I, Q, M, D, L or constant	"Electrical stop" bit: If this parameter has the value FALSE, this bit cause the drive to coast to a stop without braking.
OFF3	Input	BOOL	I, Q, M, D, L or constant	Fast stop bit - If this parameter has the value FALSE, this bit causes a fast stop by braking the drive.
F_ACK	Input	BOOL	I, Q, M, D, L or constant	Fault acknowledge bit - This bit resets the fault bit on a drive. This bit is set after the fault is cleared to indicate to the drive that it no longer needs to indicate the previous fault.
DIR	Input	BOOL	I, Q, M, D, L or constant	Drive direction control - This bit is set to indicate that the direction is forward (when SPEED_SP is positive).
DRIVE	Input	USINT	I, Q, M, D, L or constant	Drive address: This input is the address of the USS drive. The valid range is drive 1 to drive 16.
PZD_LEN	Input	USINT	I, Q, M, D, L or constant	Word length - This is the number of words of PZD data. Valid values are 2, 4, 6, or 8 words. The default is 2.
SPEED_SP	Input	REAL	I, Q, M, D, L or constant	Speed setpoint - This is the speed of the drive as a percentage of configured frequency. A positive value specifies forward direction (when DIR has the value TRUE).
CTRL3	Input	WORD	I, Q, M, D, L or constant	Control word 3 - A value written to a user-configurable parameter on the drive. The user must configure this on the drive. Optional parameter
CTRL4	Input	WORD	I, Q, M, D, L or constant	Control word 4 - A value written to a user-configurable parameter on the drive. The user must configure this on the drive. Optional parameter
CTRL5	Input	WORD	I, Q, M, D, L or constant	Control word 5 - A value written to a user-configurable parameter on the drive. The user must configure this on the drive. Optional parameter
CTRL6	Input	WORD	I, Q, M, D, L or constant	Control word 6 - A value written to a user-configurable parameter on the drive. The user must configure this on the drive.

Parameter	Declaration	Data type	Memory area	Description
CTRL7	Input	WORD	I, Q, M, D, L or constant	Control word 7 - A value written to a user-configurable parameter on the drive. The user must configure this on the drive. Optional parameter
CTRL8	Input	WORD	I, Q, M, D, L or constant	Control word 8 - A value written to a user-configurable parameter on the drive. The user must configure this on the drive. Optional parameter
NDR	Output	BOOL	I, Q, M, D, L	New data ready - If this parameter has the value TRUE, the bit indicates that the output contains data from a new communication request.
ERROR	Output	BOOL	I, Q, M, D, L	Error occurred - If this parameter has the value TRUE, this indicates that an error has occurred and the STATUS output is valid. All other outputs are set to zero on an error. Communication errors are only reported at the ERROR and STATUS outputs of the "USS_PORT" instruction.
STATUS (Page 5305)	Output	WORD	I, Q, M, D, L	Status value of the request. It indicates the result of the cycle. This is not a status word returned from the drive.
RUN_EN	Output	BOOL	I, Q, M, D, L	Run enabled - This bit indicates whether the drive is running.
D_DIR	Output	BOOL	I, Q, M, D, L	Drive direction - This bit indicates whether the drive is running forward.
INHIBIT	Output	BOOL	I, Q, M, D, L	Drive inhibited - This bit indicates the state of the inhibit bit on the drive.
FAULT	Output	BOOL	I, Q, M, D, L	Drive fault - This bit indicates that the drive has registered a fault. The user must eliminate the fault and then set the F_ACK bit to clear this bit.
SPEED	Output	REAL	I, Q, M, D, L	Drive current speed (scaled value of drive status word 2) - The value of the speed of the drive as a percentage of configured speed.
STATUS1	Output	WORD	I, Q, M, D, L	Drive status word 1 - This value contains fixed status bits of a drive.
STATUS3	Output	WORD	I, Q, M, D, L	Drive status word 3 - This value contains a user-configurable status word on the drive.
STATUS4	Output	WORD	I, Q, M, D, L	Drive status word 4 - This value contains a user-configurable status word on the drive.
STATUS5	Output	WORD	I, Q, M, D, L	Drive status word 5 - This value contains a user-configurable status word on the drive.
STATUS6	Output	WORD	I, Q, M, D, L	Drive status word 6 - This value contains a user-configurable status word on the drive.
STATUS7	Output	WORD	I, Q, M, D, L	Drive status word 7 - This value contains a user-configurable status word on the drive.
STATUS8	Output	WORD	I, Q, M, D, L	Drive status word 8 - This value contains a user-configurable status word on the drive.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

USS_RPM: Readout parameters from the drive

Description

The "USS_RPM" instruction reads a parameter from the drive. All USS functions that are assigned to one USS network and one PtP communications module must use the same instance data block. "USS_RPM" must be called from the main program OB.

Parameter

The following table shows the parameters of the "USS_RPM" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Send request: If this parameter has the value TRUE, it indicates that a new read request is desired. This is ignored if the request for this parameter is already pending.
DRIVE	Input	USINT	I, Q, M, D, L or constant	Drive address: This input is the address of the USS drive. The valid range is drive 1 to drive 16.
PARAM	Input	UINT	I, Q, M, D, L or constant	Parameter number: This input specifies which drive parameter is written. The range of this parameter is 0 to 2047. See your drive manual for details on how to access any parameters above this range.
INDEX	Input	UINT	I, Q, M, D, L or constant	Parameter index: This input specifies which drive parameter index is to be written. This is a 16-bit value where the least significant byte is the actual index value with a range of (0 to 255). The most significant byte may also be used by the drive and is drive-specific. See your drive manual for additional information.
USS_DB	InOut	USS_BASE	D	Reference to the instance DB that is created and initialized when a "USS_DRIVE" instruction is inserted in your program.
DONE	Output	BOOL	I, Q, M, D, L	If this parameter has the value TRUE, it indicates that the VALUE output holds the previously requested read parameter value. This bit is set when the "USS_DRIVE" instruction recognizes the read response from the drive. This bit is reset when either: • you request the response data via another "USS_RPM" poll or • the second of the next two calls of "USS_DRIVE (Page 5300)" is executed
ERROR	Output	BOOL	I, Q, M, D, L	Error occurred - If this parameter has the value TRUE, this indicates that an error has occurred and the STATUS output is valid. All other outputs are set to zero on an error. Communication errors are only reported at the ERROR and STATUS outputs of the "USS_PORT (Page 5299)" instruction.

Parameter	Declaration	Data type	Memory area	Description
STATUS (Page 5305)	Output	WORD	I, Q, M, D, L	This is the status value of the request. It indicates the result of the read request. Additional information is available in the "USS_Extended_Error (Page 5305)" tag for some status codes.
VALUE	Output	VARIANT	I, Q, M, D, L	This is the value of the parameter that was read and is valid only when the DONE bit has the value TRUE.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

USS_WPM: Change parameters in the drive

Description

The "USS_WPM" instruction modifies a parameter in the drive. All USS functions that are assigned to one USS network and one PtP communications module must use the same instance data block. "USS_WPM" must be called from the main program OB.

Note

EEPROM write operations

Beware of overusing the EEPROM write operation. Minimize the number of EEPROM write operations to extend the EEPROM life.

Parameter

The following table shows the parameters of the "USS_WPM" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Send request: If this parameter has the value TRUE, it indicates that a new write request is desired. This is ignored if the request for this parameter is already pending.
DRIVE	Input	USINT	I, Q, M, D, L or constant	Drive address: This input is the address of the USS drive. The valid range is drive 1 to drive 16.
PARAM	Input	UINT	I, Q, M, D, L or constant	Parameter number: This input specifies which drive parameter is written. The range of this parameter is 0 to 2047. See your drive manual for details on how to access any parameters above this range.
INDEX	Input	UINT	I, Q, M, D, L or constant	Parameter index: This input specifies which drive parameter index is to be written. This is a 16-bit value where the least significant byte is the actual index value with a range of (0 to 255). The most significant byte may also be used by the drive and is drive-specific. See your drive manual for additional information.

Parameter	Declaration	Data type	Memory area	Description
EEPROM	Input	BOOL	I, Q, M, D, L or constant	Store to drive EEPROM: If this parameter has the value TRUE, values written to the drive parameter will be stored in the drive EEPROM. If this parameter has the value FALSE, the value written is only temporarily saved and will be lost the next time the drive is switched on.
VALUE	Input	VARIANT	I, Q, M, D, L or constant	The value of the parameter that is to be written. It must be valid on the transition of REQ.
USS_DB	InOut	USS_BASE	D	This is a reference to the instance DB that is created and initialized when a "USS_DRIVE (Page 5300)" instruction is inserted in your program.
DONE	Output	BOOL	I, Q, M, D, L	If this parameter has the value TRUE, the VALUE input was written to the drive. This bit is set when the "USS_DRIVE (Page 5300)" instruction recognizes the write response from the drive. This bit is reset when either: You request the drive's confirmation that the write operation has been completed via another "USS_WPM" query or when the second of the next two calls of "USS_DRIVE (Page 5300)" is executed.
ERROR	Output	BOOL	I, Q, M, D, L	Error occurred: If this parameter has the value TRUE, this indicates that an error has occurred and the STATUS output is valid. All other outputs are set to zero on an error. Communication errors are only reported at the ERROR and STATUS outputs of the "USS_PORT (Page 5299)" instruction.
STATUS (Page 5305)	Output	WORD	I, Q, M, D, L	This is the status value of the request. It indicates the result of the write request. Additional information is available in the "USS_Extended_Error (Page 5305)" tag for some status codes.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

Parameter STATUS of USS instructions

STATUS parameter

The following table contains the status codes of the USS operation that are output at the STATUS output of the USS instructions:

STATUS* (W#16#....)	Description
0000	No error
8180	The length of the drive response did not match the characters received from the drive. The drive number where the error occurred is returned in the "USS_Extended_Error" tag. See the extended error description below this table.
8181	The parameter VALUE is not of the data type WORD, REAL, or DWORD
8182	User supplied a parameter value of the type word and received a DWORD or REAL from the drive in the response

STATUS* (W#16#....)	Description
8183	User supplied a parameter value of the type DWORD or REAL and received a word from the drive in the response
8184	Response telegram from drive had a bad checksum. The drive number where the error occurred is returned in the "USS_Extended_Error" tag. See the extended error description below this table.
8185	Illegal drive address (valid drive address range: 1-16)
8186	Speed set point out of valid range (valid speed SP range: -200% to 200%)
8187	Wrong drive number responded to the request sent. The drive number where the error occurred is returned in the "USS_Extended_Error" tag. See the extended error description below this table.
8188	Illegal PZD word length specified (valid range = 2, 4, 6 or 8 words)
8189	Illegal baud rate was specified
818A	Parameter request channel is in use by another request for this drive
818B	Drive has not responded to requests and retries. The drive number where the error occurred is returned in the "USS_Extended_Error" tag. See the extended error description below this table.
818C	Drive returned an extended error on a parameter request operation. See the extended error description below this table.
818D	Drive returned an invalid access error on a parameter request operation. See your drive manual for information of why parameter access may be limited
818E	Drive has not been initialized: This error code is output at "USS_RPM (Page 5303)" or "USS_WPM (Page 5304)" if the "USS_DRIVE (Page 5300)" instruction has not been called at least once for this drive. This prevents the initialization of the first cycle of "USS_DRIVE (Page 5300)" from overwriting a pending parameter read or write request since it initializes the drive as a new entry. To eliminate this error, call the "USS_DRIVE (Page 5300)" instruction for this drive.
80Ax-80Fx	Specific errors returned from PtP (Point-to-Point) communication instructions called by the USS library: These error code values are not modified by the USS library and are defined in the PtP instruction descriptions.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

USS_Extended_Error - USS drive extended error codes

USS drives support read and write access to a drive's internal parameters. This feature allows distributed control and configuration of the drive. Drive parameter access operations can fail due to errors such as values out of range or invalid requests in a drive's current mode. The drive generates an error code that is output in the "USS_Extended_Error" variable in the instance DB of the "USS_DRIVE (Page 5300)" instruction. This error code value is only valid for the last execution of the "USS_RPM (Page 5303)" or "USS_WPM (Page 5304)" instruction. The drive error code is put into the "USS_Extended_Error" tag when the value of STATUS is hexadecimal 818C. The error code of "USS_Extended_Error" depends on the drive variant. See the drive's manual for a description of the extended error codes for read and write parameter operations.

See also

Switching display formats in the program status (Page 2980)

MODBUS (RTU)

MB_COMM_LOAD: Configure port on the PtP module for Modbus RTU

Description

The "MB_COMM_LOAD" instruction configures a port for communication using the Modbus RTU protocol. The following hardware can be used for this:

- Up to three point-to-point modules (PtP) CM 1241 RS485 or CM 1241 RS232
- A communications board CB 1241 RS485 in addition to this

After configuration of the port, you communicate over Modbus by executing the "MB_SLAVE" or "MB_MASTER" instruction.

Call

"MB_COMM_LOAD" must be called once to configure the port for the Modbus RTU protocol. On completion of the configuration, the port can be used by the "MB_MASTER (Page 5310)" and "MB_SLAVE (Page 5318)" instructions.

"MB_COMM_LOAD" only needs to be called again if one of the communication parameters has to be modified. Each "MB_COMM_LOAD" call deletes the communications buffer. To avoid data loss during communication, you should not call the instruction unnecessarily.

One instance of "MB_COMM_LOAD" must be used to configure the port of each communication module that is used for Modbus communication. You assign a unique "MB_COMM_LOAD" instance data block for each port that you use. The S7-1200 CPU is limited to three communication modules.

An instance data block is assigned when you insert the "MB_MASTER (Page 5310)" or "MB_SLAVE (Page 5318)" instruction. This instance data block is referenced when you specify the MB_DB parameter on the "MB_COMM_LOAD" instruction.

Parameters

The following table shows the parameters of the instruction "MB_COMM_LOAD":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Execution of the instruction upon a rising edge.
PORT	Input	PORT	I, Q, M, D, L or constant	ID of the communications port: After you have inserted the communications module in the device configuration, the port ID appears in the drop-down list at the PORT box connection. This constant can also be referenced within the "Constants" tab of the tag table.
BAUD	Input	UDINT	I, Q, M, D, L or constant	Baud rate selection: 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 76800, 115200 All other values are invalid.

Parameter	Declaration	Data type	Memory area	Description
PARITY	Input	UINT	I, Q, M, D, L or constant	Parity selection: • 0 – None • 1 – Odd • 2 – Even
FLOW_CTRL	Input	UINT	I, Q, M, D, L or constant	Flow control selection: • 0 – (default) No flow control • 1 – Hardware flow control with RTS always ON (does not apply to RS485 ports) • 2 - Hardware flow control with RTS switched
RTS_ON_DLY	Input	UINT	I, Q, M, D, L or constant	RTS on-delay selection: • 0 – (default) No delay of RTS active until the first character of the message is transmitted. • 1 to 65535 – Delay in milliseconds of "RTS active" until the first character of the message is transmitted (does not apply to RS-485 ports). RTS delays must be applied independent of the FLOW_CTRL selection.
RTS_OFF_DLY	Input	UINT	I, Q, M, D, L or constant	RTS off-delay selection: • 0 – (default) No delay after the last character transmitted until "RTS inactive" • 1 to 65535 – Delay in milliseconds after the last character transmitted until "RTS inactive" (does not apply to RS-485 ports). RTS delays must be applied independent of the FLOW_CTRL selection.
RESP_TO	Input	UINT	I, Q, M, D, L or constant	Response timeout: Time in milliseconds allowed by "MB_MASTER (Page 5310)" for the slave to respond. If the slave does not respond in this time, "MB_MASTER (Page 5310)" repeats the request or terminates the request with an error if the specified number of retries has been sent. 5 ms to 65535 ms (default = 1000 ms).
MB_DB	Input	MB_BASE	D	A reference to the instance data block of the "MB_MASTER (Page 5310)" or "MB_SLAVE (Page 5318)" instructions. After you insert "MB_SLAVE (Page 5318)" or "MB_MASTER (Page 5310)" in your program, the DB identifier appears in the drop-down list at the MB_DB box connection.
DONE	Output	BOOL	I, Q, M, D, L	Execution of instruction completed without error.
ERROR	Output	BOOL	I, Q, M, D, L	Error: • 0 – No error detected • 1 – Indicates that an error was detected. An error code is output in the STATUS parameter.
STATUS	Output	WORD	I, Q, M, D, L	Port configuration error code

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter STATUS

Error code* (W#16#...)	Description
0000	No error
8180	Invalid value for the port ID (wrong address for the communications module).
8181	Invalid baud rate value.
8182	Invalid parity value.
8183	Invalid flow control value.
8184	Invalid value for the timeout of the response (the time before which a timeout is reported must be at least 25 ms).
8185	Incorrect pointer in the MB_DB parameter to the instance DB of the "MB_MASTER (Page 5310)" or "MB_SLAVE (Page 5318)" instruction.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

MB_COMM_LOAD data block tags

The table below shows the public static tags in the instance DB of MB_COMM_LOAD that you can use in your program.

Table 9-162 Static tags in the instance DB

Tag	Data type	Default	Description
ICHAR_GAP	WORD	0	Delay for the character spacing between characters. This parameter is specified in milliseconds and is used to increase the anticipated period between the received characters. The corresponding number of bit times for this parameter is added to the Modbus standard value of 35 bit times (3.5 character times).
RETRIES	WORD	2	The number of repeated attempts by the master before the error code 0x80C8 is returned for "No response".
MODE	USINT	0	Mode Permitted modes are: • 0 = Full duplex (RS232) • 1 = Full duplex (RS422) four-wire mode (point-to-point) • 2 = Full duplex (RS422) four-wire mode (multipoint master) • 3 = Full duplex (RS422) four-wire mode (multipoint slave) • 4 = Half duplex (RS485) two-wire mode

Tag	Data type	Default	Description
LINE_PRE	USINT	0	Receive line initial state Permitted defaults are: • 0 = "No" default • 1 = Signal R(A)=5V, signal R(B)=0 V (break detection): No break detection is possible with this default. Can only be selected with: "Full duplex (RS422) four-wire mode (point-to-point connection)" and "Full duplex (RS422) four-wire mode (multipoint slave)". • 2 = Signal R(A)=0 V, signal R(B)=5 V: This default corresponds to the at-rest state (no send process is active). No break detection is possible with this default.
CABLE-BREAK	USINT	0	Activate cable break detection: • 0 - not activated • 1 - activated

See also

Switching display formats in the program status (Page 2980)

MB_MASTER: Communicate via the PtP port as Modbus master

Description of MB_MASTER

Description

The "MB_MASTER" instruction allows your program to communicate as a Modbus master using a port on a point-to-point module (CM) or a communications board (CB). You can access data in one or more Modbus slave devices.

Before the "MB_MASTER" instruction can communicate with a port, "MB_COMM_LOAD (Page 5307)" must first execute.

An instance DB is created when you insert the "MB_MASTER" instruction in your program. You specify this instance DB in the MB_DB input parameter of the "MB_COMM_LOAD (Page 5307)" instruction.

Rules for Modbus master communication

- A port used for Modbus master requests cannot be used for "MB_SLAVE".
- A port can be used for one or more "MB_MASTER" calls if the same instance DB is used.
- The Modbus instructions do not use communication interrupt events to control the communication process. Your program must poll the "MB_MASTER" instruction for completed send and receive operations.

- Calling the instruction:
  - Call the "MB_MASTER" instruction if possible in a cyclic program OB. The instruction can only be called in a time delay or cyclic interrupt OB.
  - Do not call more than one "MB_MASTER" instruction in organization blocks with different priority classes. If a "MB_MASTER" instruction executes "preemptively" from a higher priority class, the instruction may execute incorrectly.
  - Do not call the "MB_MASTER" instruction in a startup, diagnostics or time error OB.
- After a transfer has started, the EN parameter (LAD/FBD) must remain set to the value "1" until the DONE or ERROR output parameter is set to "1" by the instruction. A renewed call by the REQ parameter while the instruction is executing causes an error. After the instruction executes, the bit in the REQ parameter remains set for the time specified by the BLOCKED_PROC_TIMEOUT parameter in the instance DB.
- If "MB_MASTER" sends a request to a slave, make sure that "MB_MASTER" continues to execute until the response from the slave arrives.

Parameters

The following table shows the parameters of the "MB_MASTER" instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ (Page 5313)	Input	BOOL	I, Q, M, D, L	Request input: • 0 – No request • 1 – Request to transmit data to Modbus slave(s)
MB_ADDR	Input	UINT	I, Q, M, D, L or constant	Modbus RTU station address: • Default address range: 0 to 247 • Extended address range: 0 to 65535 The value "0" is reserved for broadcasting a message to all Modbus slaves. Modbus function codes 05, 06, 15, and 16 are the only function codes supported for broadcast.
MODE (Page 5313)	Input	USINT	I, Q, M, D, L or constant	Mode selection: Specifies the type of request: Read, write, or diagnostics: Refer to the Modbus functions table for details.
DATA_ADDR (Page 5313)	Input	UDINT	I, Q, M, D, L or constant	Starting address in the slave: Specifies the starting address of the data to be accessed in the Modbus slave. You will find the valid addresses in the Modbus functions table.
DATA_LEN	Input	UINT	I, Q, M, D, L or constant	Data length: Specifies the number of bits or words to be accessed in this request. You will find the valid lengths in the Modbus functions table.
DATA_PTR (Page 5315)	Input	VARIANT	M, D	Points to the DB or bit memory address of the CPU for the data to be written or read. For a DB, this must be created with the "Standard - compatible with S7-300/400" access type.
DONE	Output	BOOL	I, Q, M, D, L	• 0: Transaction not completed • 1: Transaction completed without error

Parameters	Declaration	Data type	Memory area	Description
BUSY	Output	BOOL	I, Q, M, D, L	• 0: No "MB_MASTER" transaction in progress • 1: "MB_MASTER" transaction in progress
ERROR	Output	BOOL	I, Q, M, D, L	• 0: No error • 1: Error, the error code is indicated by the STATUS parameter
STATUS	Output	WORD	I, Q, M, D, L	Execution condition code

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

STATUS parameter

Table 9-163 Communications and configuration error messages of the instruction

Error code* (W#16#....)	Description
0000	No error
80C8	Slave timeout. Check the baud rate, parity and the connectors on the slave.
80D1	The receiver issued a flow control request to suspend an active transmission and never re-enabled the transmission within the wait time. This error is also generated during hardware flow control if the recipient does not detect CTS within the wait time.
80D2	The send request was aborted because no DSR signal is received from the DCE.
80E0	The message was terminated because the receive buffer is full.
80E1	The message was terminated as a result of a parity error.
80E2	The message was terminated as a result of a framing error.
80E3	The message was terminated as a result of an overrun error.
80E4	The message was terminated as a result of the specified length exceeding the total buffer size.
8180	Invalid value for the port ID.
8186	Invalid Modbus station address
8188	The MODE parameter has an invalid value for a broadcast call.
8189	Invalid data address value.
818A	Invalid data length value.
818B	Invalid pointer to the local data source/destination: Size not correct
818C	The DATA_PTR parameter has an invalid pointer. Use a pointer to a bit memory area or a DB with the "Standard - compatible with S7-300/400" access type.
8200	Port is busy processing a send request
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

Table 9-164 Error messages of the Modbus protocol

Error code* (W#16#....)	Response code of slave	Description
8380	-	CRC error
8381	01	Function code not supported
8382	03	Data length error
8383	02	Error in the data address or address outside the valid range of DATA_PTR
8384	> 03	Data value error
8385	03	Data diagnostic code value not supported (function code 08)
8386	-	Function code of the response does not match the function code of the query.
8387	-	Response from wrong slave
8388	-	The response of the slave to a write call is not correct. The data sent by the slave does not match the query from the master.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

See also

Switching display formats in the program status (Page 2980)

Parameter REQ

Description

- REQ = FALSE: No request
- REQ = TRUE: Request to transmit data to Modbus slave(s)

You can control this input by means of a level-controlled or edge-controlled contact.

A state machine is started every time this input is activated to ensure that another instruction "MB_MASTER" that uses the same instance DB can only issue a request when the current request has been processed. All other input states are recorded and saved internally for the current request until the reply has been received or an error has been detected.

If the same instance of "MB_MASTER" is executed again with REQ input = 1 before the current request is completely processed, no subsequent transfers are made. However, if the request has been processed, a new request is issued at the time when "MB_MASTER" is executed again with REQ input = 1.

DATA_ADDR and MODE parameters

Description

You specify the start address for data access to the Modbus slave using the DATA_ADDR parameter.

With the MODE parameter and the Modbus address, you specify the function code to be transferred to the Modbus slave. The following table shows the relationship between the MODE parameter, the function code and Modbus address range.

MODE	Modbus function	Data length	Operation and data	Modbus address
0	01	1 to 2000 1 to 1992 (1)	Read output bits: 1 to (1992 or 2000) bits per query	1 to 9999
0	02	1 to 2000 1 to 1992 (1)	Read input bits: 1 to (1992 or 2000) bits per query	10001 to 19999
0	03	1 to 125 1 to 124 (1)	Read holding register: 1 to (124 or 125) WORD per query	40001 to 49999 or 400001 to 465535
0	04	1 to 125 1 to 124 (1)	Read input WORD: 1 to (124 or 125) WORD per query	30001 to 39999
1	05	1	Writing an output bit: One bit per query	1 to 9999
1	06	1	Writing a holding register: 1 WORD per query	40001 to 49999 or 400001 to 465535
1	15	2 to 1968 2 to 1960 (1)	Writing multiple output bits: 2 to (1960 or 1968) bits per query	1 to 9999
1	16	2 to 123 2 to 122 (1)	Writing multiple holding registers: 2 to (122 or 123) WORD per query	40001 to 49999 or 400001 to 465535
2	15	1 to 1968 2 to 1960 (1)	Writing one or more output bits: 1 to (1960 or 1968) bits per query	1 to 9999
2	16	1 to 123 2 to 122 (1)	Writing one or more holding registers: 1 to (122 or 123) WORD per query	40001 to 49999 or 400001 to 465535
11	11	0	Reading out the communications status word of the slaves and the event counter: The status word indicates execution of the instruction (0: is not executing; 0xFFFF: is executing). The event counter is incremented each time a message is transferred successfully. The DATA_ADDR and DATA_LEN parameters of the "MB_MASTER" instruction are ignored with this function.	-
80	08	1	Checking the slave status by reading the error code (0x0000): 1 WORD per query	-
81	08	1	Resetting the event counter of the slave with the diagnostics code 0x000A: 1 WORD per query	-
3 to 10, 12 to 79, 82 to 2555			Reserved	-

(1) For the "Extended address range", the maximum data length is reduced by one byte or one WORD depending on which data type is used for the function.

Parameter DATA_PTR

Description

The DATA_PTR parameter is a pointer to a data block or bit memory from which the data should be written or read. If you use a data block, create a global data block with the "Standard - compatible with S7-300/400" access type.

Data block structures for the DATA_PTR parameter

- These data types are valid for **reading of words** of Modbus addresses 30001 to 39999, 40001 to 49999, and 400001 to 465536 and also for **writing of words** to Modbus addresses 40001 to 49999 and 400001 to 465536.
  - Standard array of WORD, UINT, or INT data types (see below).
  - Named WORD, UINT, or INT structure where each element has a unique name and 16 bit data type.
  - Named complex structure where each element has a unique name and 16 bit or 32 bit data type.
- For reading and writing of bits of Modbus addresses 00001 to 09999 and 10001 to 19999.
  - Standard array of Boolean data types.
  - Named Boolean structure of uniquely named Boolean variables.
- Although not required, it is recommended that each "MB_MASTER" instruction has its own separate memory area in a global data block. The reason for this recommendation is that there is a greater possibility of data corruption if multiple "MB_MASTER" instructions are reading and writing the same area of a global data block.
- The memory areas for DATA_PTR do not need to be in the same global data block. You can create one data block with multiple areas for Modbus read operations, one data block for Modbus write operations, or one data block for each slave station.

Instance DB of the "MB_MASTER" instruction

Static variables of the instance DB

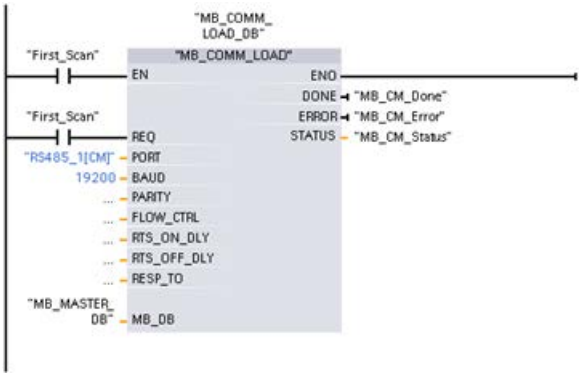
The following table describes the static variables of the instance DB of the instruction that you can use in the user program.

Variable	Data type	Description
MB_STATE	UINT	Internal status of the Modbus instruction
BLOCKED_ PROC_TIMEOUT	REAL	Time between completion of the instruction call and resetting the ACTIVE bit in the instance DB. The time buffer is used to avoid execution of the instruction being terminated before a job has been sent completely. The default time is 500 ms.
EXTENDED_ ADDRESSING	BOOL	Configuring addressing: • 0: Default address area (1 byte) • 1: Extended address area (2 bytes) For additional information, refer to the section EXTENDED_ADDRESSING: Instance DB of the "MB_SLAVE" instruction (Page 5321)

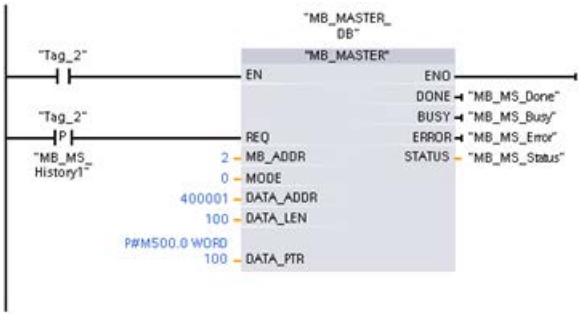
Sample program for a Modbus master

Networks (LAD)

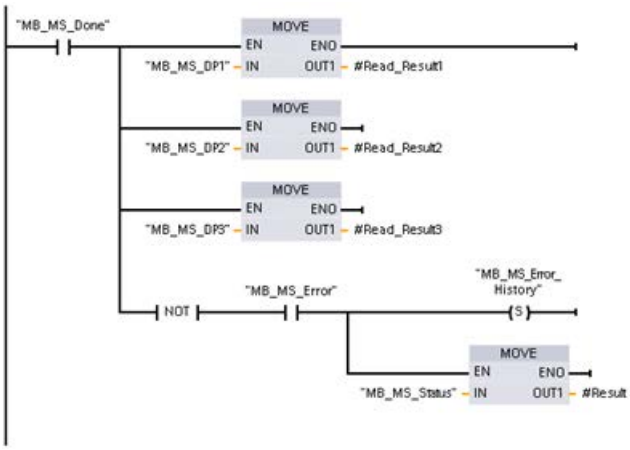
Network 1: Initialize parameters of the RS-485 module only once during the first cycle.



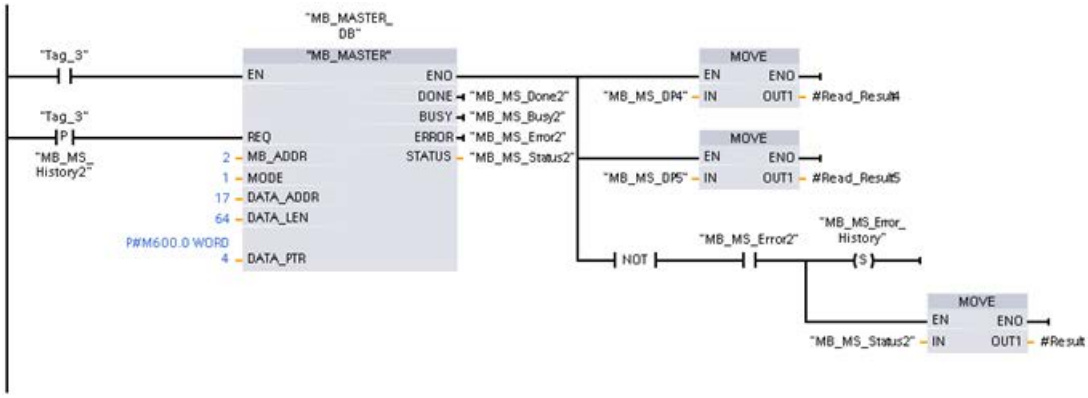
Network 2: Read 100 words from the holding register of the slave.



Network 3: This is an optional network that displays the values of the first 3 words as soon as the read operation has executed.



Network 4: Write 64 bits to the process image output, starting at slave address Q2.0.



MB_SLAVE: Communicate via the PtP port as Modbus slave

Description of MB_SLAVE

Description

The "MB_SLAVE" instruction allows your program to communicate as a Modbus slave using a port on a point-to-point module (PtP) or a communications board (CB). A Modbus RTU master can issue a request and then your program responds via "MB_SLAVE" execution.

You must assign a unique instance data block when you insert the "MB_SLAVE" instruction in your program. This instance data block is used when you specify it at the MB_DB parameter of the "MB_COMM_LOAD (Page 5307)" instruction.

Modbus communication function codes (1, 2, 4, 5, and 15) can read and write bits and words directly in the process image input and process image output in the target system. The following table shows the mapping of Modbus addresses to the process image in the CPU.

Modbus functions of "MB_SLAVE"						S7-1200	
Codes	Function	Data area	Address range			Data area	CPU address
01	Read bits	Output	1	to	8192	Process image output	Q0.0 to Q1023.7
02	Read bits	Input	10001	to	18192	Process image input	I0.0 to I1023.7
04	Read words	Input	30001	to	30512	Process image input	IW0 to IW1022
05	Write bit	Output	1	to	8192	Process image output	Q0.0 to Q1023.7
15	Write bits	Output	1	to	8192	Process image output	Q0.0 to Q1023.7

Modbus communication function codes (function codes 3, 6, 16) use a separate holding register. To do this, you can use bit memory or a data block with the "Standard - compatible with S7-300/400" access type.

You specify the type of the holding register using the MB_HOLD_REG parameter of the MB_SLAVE instruction. The following table shows the mapping of the Modbus holding register to the DB address of MB_HOLD_REG in the target system.

Modbus functions of "MB_SLAVE"				S7-1200	
Codes	Function	Data area	Address range (WORD number)	Address in the DB (BYTE number)	Bit memory address (BYTE number)
03	Read words	Holding register	40001 to 49999 or	DW0 to DW19998 or	MW0 to CPU limit
			400001 to 465535	DW0 to DW131068	
06	Write word	Holding register	40001 to 49999 or	DW0 to DW19998 or	
			400001 to 465535	DW0 to DW131068	
16	Write words	Holding register	40001 to 49999 or	DW0 to DW19998 or	
			400001 to 465535	DW0 to DW131068	

The table below shows the supported Modbus diagnostic functions.

S7-1200 "MB_SLAVE" Modbus diagnostic functions		
Codes	Subfunction	Description
08	0000H	Return query data echo test: The "MB_SLAVE" instruction returns the echo of a received data word to a Modbus master.
08	000AH	Clear communication event counter: The "MB_SLAVE" instruction clears the communication event counter that is used for Modbus function 11.
11	-	Get communication event counter: The "MB_SLAVE" instruction uses an internal communication event counter for recording the number of successful Modbus read and write requests that are sent to the Modbus slave. The counter is not incremented on any Function 8, Function 11, or broadcast requests. It is also not incremented on any requests that result in a communication error (for example, parity or CRC errors).

The "MB_SLAVE" instruction supports broadcast write requests from Modbus masters as long as the requests include access to valid addresses.

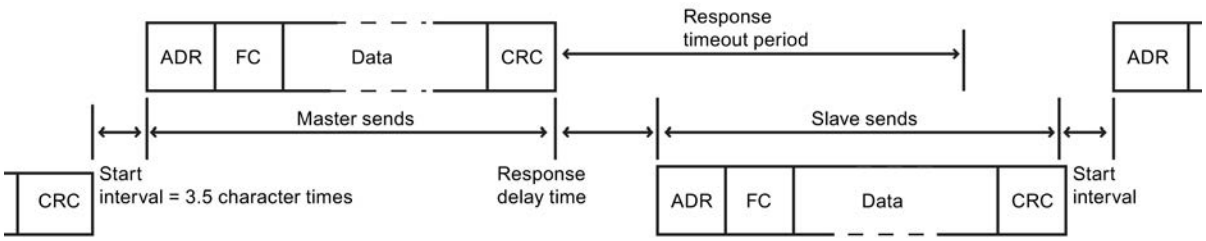
Regardless of the validity of a request, "MB_SLAVE" gives no response to a Modbus master as the result of a broadcast request.

Rules of Modbus slave communication

- "MB_COMM_LOAD" must be executed to configure a port, before the "MB_SLAVE" instruction can communicate with this port.
- If a port is to respond as a slave to a Modbus master, then that port cannot be used by "MB_MASTER (Page 5310)". Only one instance of "MB_SLAVE" can be used with a given port.
- The Modbus instructions do not use communication interrupt events to control the communication process. Your program must control the communication process by polling the "MB_SLAVE" instruction for completed send and receive operations.
- The "MB_SLAVE" instruction must be executed periodically at a rate that allows it to make a timely response to incoming requests from a Modbus master. It is therefore advisable to call the instruction in a cyclic program OB. Calling the "MB_SLAVE" instruction in an interrupt OB is possible but not advisable since it can lead to long delays in execution.

Frequency of execution of "MB_SLAVE"

The "MB_SLAVE" instruction must be executed periodically to receive each request from the Modbus master and to respond as required. The frequency of execution of "MB_SLAVE" is dependent upon the specified response timeout period of the Modbus master. This is illustrated in the following diagram.



The response timeout period is the amount of time a Modbus master waits for the start of a response from a Modbus slave. This time period is not defined by the Modbus protocol, but rather by a parameter of each Modbus master. The frequency of execution (time between one execution and the next execution) of "MB_SLAVE" must be based on the particular parameters of your Modbus master. As a minimum, you should execute "MB_SLAVE" twice within the response timeout period of the Modbus master.

Parameters

The following table shows the parameters of the "MB_SLAVE" instruction:

Parameter	Declaration	Data type	Memory area	Description
MB_ADDR	Input	USINT	I, Q, M, D, L or constant	Station address of the Modbus slave (address range: 0 to 255)
MB_HOLD_REG	Input	VARIANT	D	Pointer to the Modbus holding register DB. The DB must be created with the "Standard - compatible with S7-300/400" access type.
NDR	Output	BOOL	I, Q, M, D, L	New data ready: • 0: No new data • 1: Indicates that new data has been written by the Modbus master
DR	Output	BOOL	I, Q, M, D, L	Read data: • 0: No data read • 1: Indicates that data has been read by the Modbus master
ERROR	Output	BOOL	I, Q, M, D, L	• 0: No error detected • 1: Error, a corresponding error code is output in the STATUS
STATUS	Output	WORD	I, Q, M, D, L	Error code

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

STATUS parameter

STATUS* (W#16#....)	Description
80C8	The specified response timeout (refer to RCVTIME or MSGTIME) is "0".
80D1	The receiver issued a flow control request to suspend an active transmission and never re-enabled the transmission within the wait time. This error is also generated during hardware flow control if the recipient does not detect CTS within the wait time.
80D2	The send request was aborted because no DSR signal is received from the DCE.
80E0	The message was terminated because the receive buffer is full
80E1	The message was terminated as a result of a parity error
80E2	The message was terminated as a result of a message frame error
80E3	The message was terminated as a result of a overrun error

STATUS* (W#16#....)	Description	
80E4	The message was terminated as a result of the specified length exceeding the total buffer size	
8180	Invalid value for the port ID.	
8186	Invalid Modbus station address	
8187	Invalid pointer to MB_HOLD_REG-DB	
818C	Pointer to a type safe DB type MB_HOLD_REG (must be a Classic DB type)	
Response code sent to Modbus master (B#16#...)		
8380	No response	CRC error
8381	01	Function code not supported or not supported in a broadcast
8382	03	Data length error
8383	02	Error in the data address or address outside the valid range of MB_HOLD_REG
8384	03	Data value error
8385	03	Data diagnostic code value not supported (function code 08)
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

See also

Switching display formats in the program status (Page 2980)

Instance DB of the "MB_SLAVE" instruction

Static variables of the instance DB

The following table describes the static variables of the instance DB of the instruction that you can use in the user program. Your program can write values to the HR_Start_Offset and Extended_Addressing variables and control the Modbus slave operations.

The other variables can be read to monitor the Modbus status.

Variable	Data type	Description
HR_Start_Offset	WORD	Start address of the Modbus holding register (default = " 0")
Extended_ Addressing	BOOL	Configuring addressing: • 0: Default address area (1 byte) • 1: Extended address area (2 bytes)
Request_Count	WORD	Total number of queries received by the slave
Slave_Message_ Count	WORD	Number of queries sent to this specific slave
Bad_CRC_Count	WORD	Number of received queries with CRC errors
Broadcast_Count	WORD	Number of received broadcast queries

Variable	Data type	Description
Exception_Count	WORD	Number of Modbus-specific errors that require the return of an exception
Success_Count	WORD	The number of requests received for this specific slave without protocol errors

HR_Start_Offset

The addresses of the Modbus holding register start at 40001 or 400001. These addresses correspond to the start address of the holding register in the target system memory. Using the HR_Start_Offset variable, you can set the offset to a different start address.

Example: A holding register starts at MW 100 and has a length of 100 WORD. With an offset of 20 in the HR_Start_Offset parameter, the holding register begins at address 40021 instead of 40001. Each address below 40021 and above 400119 causes an addressing error.

	HR_Start_Offset = 0			HR_Start_Offset = 20	
	Modbus word address	S7-1200 byte address		Modbus word address	S7-1200 byte address
Minimum	40001	MW100		40021	MW100
Maximum	40099	MW198		40119	MW198

Extended_Addressing

To address the Modbus slave, a single byte (default address range) or a double byte (extended address range) can be configured. Extended addressing is used to address more than 247 devices in a single network. If you decide on extended addressing, you can address a maximum of 64,000 addresses. Below, you will see a frame of Modbus function 1 as an example.

Table 9-165 Slave address with one byte (byte 0)

Function 1	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	
Request	Slave address	F code	Start address		Length of the coils		
Valid response	Slave address	F code	Length	Coil data			
Error response	Slave address	0x81	E code				

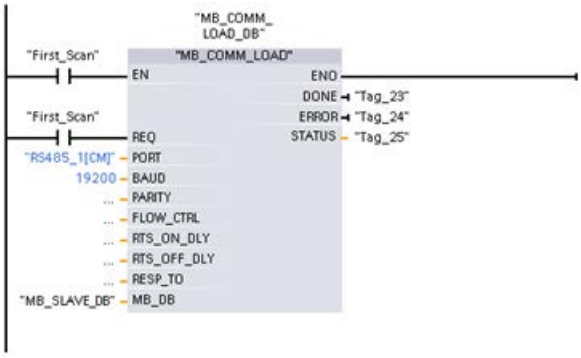
Table 9-166 Slave address with two bytes (byte 0 and byte 1)

Function 1	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Request	Slave address		F code	Start address		Length of the coils	
Valid response	Slave address		F code	Length	Coil data		
Error response	Slave address		0x81	F code			

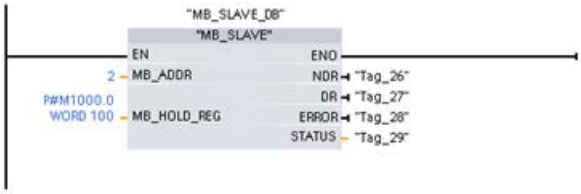
Sample program for a Modbus slave

Networks (LAD)

Network 1: Initialize parameters of the RS-485 module only once during the first cycle.



Network 2: Check the requests of the Modbus master in every cycle. 100 words starting at MW1000 are configured for the Modbus holding register.



MODBUS (TCP)

MODBUS (TCP)

MB_CLIENT: Communicating via PROFINET as a Modbus TCP client

Description of MB_CLIENT

Description

The "MB_CLIENT" instruction communicates as a Modbus TCP client via the PROFINET connection of the S7-1200 CPU. To use the instruction, you do not require any additional hardware module. With the "MB_CLIENT" instruction, you establish a connection between the client and the server, send requests and receive responses and control connection termination of the Modbus TCP server.

Parameters

The following table shows the parameters of the "MB_CLIENT" instruction:

Parameter	Declaration	Data type	Description
REQ (Page 5326)	Input	BOOL	Communication request with the Modbus TCP server The REQ parameter is level-controlled. This means that as long as the input is set (REQ=true), the instruction sends communication requests. • The instance DB for other clients is blocked with the communications request. • Changes to the input parameters will not become effective until the server has responded or an error message has been output. • If the parameter REQ is set again during an ongoing Modbus request, no additional transmission takes place afterwards.
DISCONNECT (Page 5326)	Input	BOOL	With this parameter, you control the establishment and termination of the connection to the Modbus server: • 0: Establish a communications connection to the specified IP address and port number. • 1: Disconnect the communication connection. No other function is executed during connection termination. The value 7003 is output at the STATUS parameter after successful connection termination. The request is sent immediately if the REQ parameter is set during connection establishment.
CONNECT_ID	Input	UINT	Unique ID to identify the connection. Each instance of the instructions "MB_CLIENT" and "MB_SERVER (Page 5331)" must be assigned a unique connection ID.
IP_OCTET_1	Input	USINT	1. Octet of the IP address* of the Modbus TCP server.
IP_OCTET_2	Input	USINT	2. Octet of the IP address* of the Modbus TCP server.
IP_OCTET_3	Input	USINT	3. Octet of the IP address* of the Modbus TCP server.
IP_OCTET_4	Input	USINT	4. Octet of the IP address* of the Modbus TCP server.
IP_PORT	Input	UINT	IP port number of the server to which the client establishes the connection and communicates using the TCP/IP protocol (default value: 502).
MB_MODE (Page 5327)	Input	USINT	Selects the mode of the request (read, write or diagnostics).
MB_DATA_ADDR (Page 5327)	Input	UDINT	Start address of the data accessed by the "MB_CLIENT" instruction.
DATA_LEN	Input	UINT	Data length: Number of bits or words for the data access (see "MB_MODE and MB_DATA_ADDR parameters" - data length).
MB_DATA_PTR (Page 5328)	InOut	VARIANT	Pointer to the Modbus data register: The register is a buffer for the data received from the Modbus server or to be sent to the Modbus server. The pointer must reference a global data block with standard access. The number of bits addressed must be divisible by 8.
DONE	Out	BOOL	The bit at output parameter DONE is set to "1" as soon as the last job is completed without errors.
BUSY	Out	BOOL	• 0: No "MB_CLIENT " job in progress • 1: "MB_ CLIENT " job in progress

Parameter	Declaration	Data type	Description
ERROR	Out	BOOL	• 0: No error • 1: Error occurred. The cause of error is indicated by the STATUS parameter.
STATUS (Page 5329)	Out	WORD	Error code of the instruction.
* 8-bit long component of the 32-bit IPv4 IP address of the Modbus TCP server.			

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Note

Consistent input data during an "MB_CLIENT" call

When a Modbus client calls a Modbus instruction, the status of the input parameters is stored internally and then compared at the next call. The comparison is used to determine whether this particular call initialized the current request. Several "MB_CLIENT" calls can be executed if you use a common instance DB. The values of the input parameters must not be changed while an "MB_CLIENT" instance is being executed. If the input parameters are changed during execution, "MB_CLIENT" cannot be used to check whether or not the instance is currently being executed.

Multiple client connections

A Modbus TCP client can support several TCP connections and the maximum number of connections depends on the CPU being used. The total number of connections of one CPU, including those of the Modbus TCP clients and server must not exceed the maximum number of supported connections. Modbus TCP connections can also be shared by client and/or server connections.

With individual client connections, remember the following rules:

- Each "MB_CLIENT" connection must use a unique instance DB.
- For each "MB_CLIENT" connection, a unique server IP address must be specified.
- Each "MB_CLIENT" connection requires a unique connection ID.
  The relevant individual connection ID must be used for each individual instance DB of the instruction. The connection ID and instance DB belong together in pairs and must be unique for each connection.
- Unique numbers for the IP port may or may not be required depending on the server configuration.

Static tags of the instruction

The following table describes the editable static tags of the instance data block of the "MB_CLIENT" instruction.

Tag	Data type	Start value	Description
Blocked_Proc_Timeout	REAL	3.0	Wait time in seconds before the static tag ACTIVE is reset if there is a blocked Modbus instance. This can, for example, occur if a client request is output and the execution of the client function aborts before the request was fully executed. The maximum wait time is 55 seconds.
MB_Transaction_ID	WORD	1	Transaction ID of the Modbus TCP protocol. The start value of "1" should only be changed if the Modbus TCP server requires a different value.
MB_Unit_ID	BYTE	255	Unit ID of the Modbus protocol. The tag corresponds to the slave address of the Modbus RTU protocol. Change this value only if the Modbus TCP server can be used as a gateway and is controlled by the application program within the Modbus server.
RCV_TIMEOUT	REAL	2.0	Time interval in seconds in which the "MB_CLIENT" instruction waits for a response from the server.
Connected	BOOL	0	Indicates whether the connection to the assigned client has been established or not: 1 = connected, 0 = not connected.

See also

MB_CLIENT example 1: Send several requests via a TCP connection (Page 5337)

MB_CLIENT example 2: Send multiple requests via several TCP connections (Page 5338)

MB_CLIENT example 3: Coordinate several requests (Page 5339)

REQ and DISCONNECT parameters

Description

If no instance of the "MB_CLIENT" instruction is executing and if the value of the DISCONNECT parameter is "0", a new job executes if REQ=1. If there is not yet a connection, this is established during execution.

If the same instance of the "MB_CLIENT" instruction executes again (DISCONNECT = 0 and REQ = 1), before the active job was executed, this is not executed on completion of the active job. A new job can only be started on completion of the active job (REQ = 1).

You can monitor the status of execution with the DONE parameter. You can use this to monitor the status of execution if the "MB_CLIENT" instruction is executed sequentially.

See also

Description of MB_CLIENT (Page 5323)

MB_MODE and MB_DATA_ADDR parameters

Description

Instead of a function code, the "MB_CLIENT" instruction uses the MB_MODE parameter. The MB_DATA_ADDR parameter is used to specify the Modbus start address of the data you want to access. The combination of the parameters MB_MODE and MB_DATA_ADDRdefines the function code used in the current Modbus message.

The following table shows the relationship between the MB_MODE parameter, the Modbus function and the address space.

MB_MODE	Modbus function	Data length	Function and data type	MB_DATA_ADDR
0	01	1 to 2000	Read output bits: 1 to 2000 bits per call	1 to 9999
0	02	1 to 2000	Read input bits: 1 to 2000 bits per call	10001 to 19999
0	03	1 to 125	Read holding register: 1 to 125 WORD per call	40001 to 49999
0	04	1 to 125	Read input words: 1 to 125 WORD per call	30001 to 39999
1	05	1	Write an output bit: One bit per call	1 to 9999
1	06	1	Write a holding register: 1 WORD per call	40001 to 49999
1	15	2 to 1968	Write multiple output bits: 2 to 1968 bits per call	1 to 9999
1	16	2 to 123	Write several holding registers: 2 to 123 WORD per call	40001 to 49999
2	15	1 to 1968	Write one or more output bits: 1 to 1968 bits per call	1 to 9999
2	16	1 to 123	Write one or more holding registers: 1 to 123 WORD per call	40001 to 49999
11	11	0	Read status word and event counter of server communication: • The status word reflects the the processing status (0 - not processing, 0xFFFF - processing). • The event counter is incremented each time a message is sent successfully. The MB_DATA_ADDR and MB_DATA_LEN parameters of the "MB_CLIENT" instruction are not evaluated when this function executes.	-
80	08	1	Check the server status with the error code 0x0000 (return loop test - the server sends the request back): 1 WORD per call	-

MB_MODE	Modbus function	Data length	Function and data type	MB_DATA_ADDR
81	08	1	Reset the event counter of the server with the error code 0x000A: 1 WORD per call	
3 to 10, 12 to 79, 82 to 255			Reserved	

See also

Description of MB_CLIENT (Page 5323)

MB_DATA_PTR parameter

Description

The MB_DATA_PTR parameter is a pointer to a data buffer for storing the data read from or written to the Modbus server. As the data buffer, you can use a global data block or a memory area (M).

For a buffer in the memory area (M), use a pointer in the ANY format as follows: "P#bit address" "data type" "length" (example: P#M1000.0 WORD 500).

The MB_DATA_PTR parameter uses a communications buffer:

- For the communication functions of the "MB_CLIENT" instruction:
  - Reading and writing of 1 bit of data of the Modbus server addresses 00001 to 09999 and 10001 to 19999.
  - Reading of 16-bit WORD data of the Modbus server addresses 30001 to 39999 and 40001 to 49999.
  - Writing 16-bit WORD data of the Modbus server addresses 40001 to 49999.
- During data transmission (length: bit or WORD) from or to the global DB or the memory area (M) that you assigned with the MB_DATA_PTR parameter.

If you use a data block for the buffer in the MB_DATA_PTR parameter, you will need to assign data types to the DB elements.

- Use the 1-bit data type BOOL for a Modbus bit address
- Use a 16-bit data type such as WORD, UINT, INT or REAL for a Modbus WORD address.
- Use a 32-bit data type (double word) such as DWORD, DINT or REAL for two Modbus WORD addresses.

- With MB_DATA_PTR, you can also access complex DB elements such as:
  - Standard arrays
  - Structures with unique element names
  - Complex structures with unique naming of the elements and data type lengths of 16 or 32 bits.
- The data areas for the MB_DATA_PTR parameter can also be in different global data blocks (or in different memory areas). You can, for example, use a data block for the the read jobs and another one for the write jobs or a separate data block for each "MB_CLIENT" station.

See also

Description of MB_CLIENT (Page 5323)

Parameter STATUS

Parameter STATUS (general status information)

STATUS* (W#16#)	Description
0000	Instruction executed without errors.
0001	Connection established.
0003	Connection terminated.
7000	No call active (REQ=0).
7001	First call with REQ=1: Processing started; BUSY has the value 1.
7002	Intermediate call (REQ irrelevant). Processing already active; BUSY has the value 1.
7003	Connection is being terminated.
7004	Connection established and monitored. No job processing active.
7005	Data was sent.
7006	Data was received.
80BB	Invalid value at ACTIVE_EST parameter (identifier for the type of connection establishment, see T_CON_PARAM): • Only passive connection establishment permitted for server (ACTIVE_EST = FALSE). • Only active connection establishment permitted for client (ACTIVE_EST = TRUE).
8380	Received Modbus frame has incorrect format or too few bytes were received.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

Parameter STATUS (protocol error)

STATUS* (W#16#)	Code of the response to the Modbus client (B#16#)	Description
8381	01	Function code is not supported.
8382	03	Error in data length.
8383	02	Error in the data address or access outside the address area of MB_DATA_PTR (Page 5328).
8384	03	Error in data value.
8385	03	Error codes of diagnostics not supported (function code 08).
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

Parameter STATUS (parameter error)

In addition to the errors listed in the following table, errors are also possible with the "MB_CLIENT" instruction caused by the communications instructions ("TCON", "TDISCON", "TSEND" and "TRCV").

STATUS* (W#16#)	Description
80C8	No response of the server in the defined period. Check the connection to the Modbus server. This error is only reported on completion of the configured repeated attempts. If the "MB_CLIENT" instruction does not receive an answer with the originally transferred transaction ID (MB_TRANSACTION_ID tag) within the defined period, this error code is output.
8188	The MB_MODE parameter has an invalid value.
8189	Invalid addressing of data at the MB_DATA_ADDR parameter.
818A	Invalid data length at the MB_DATA_LEN parameter.
818B	The MB_DATA_PTR parameter has an invalid pointer. You should also check the values of the MB_DATA_ADDR (Page 5327) and MB_DATA_LEN parameters.
818C	• The MB_DATA_PTR (Page 5328) pointer references an an optimized data block. Either use a data block with standard access or a memory area • Timeout at parameter BLOCKED_PROC_TIMEOUT (see static tags of instruction). The limit of 55 seconds has been exceeded.
818D	The transaction ID (MB_TRANSACTION_ID tag) does not correspond to the one sent originally (see static tags of instruction).
8200	• A further Modbus request is currently being processed via the port. • Another instance of MB_CLIENT with the same connection parameters is processing an existing Modbus request.
8380	Received block of transferred Modbus data is not well-formed or too few bytes were received.
8386	Received function code does not match the one sent originally.
8387	• The assigned connection ID is different from that used for previous requests. Only one connection ID can be used for each instance DB of the "MB_CLIENT" instruction. • The error code is also output when the ID of the Modbus TCP protocol received by the server is not "0".

STATUS* (W#16#)	Description
8388	The Modbus server sent a different data length than was requested. This error occurs only when using the Modbus functions 15 or 16.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

Note

Error codes of internally used communications instructions.

With the "MB_CLIENT" instruction, in addition to the errors listed in the tables, errors caused by the communication instructions ("TCON", "TDISCON", "TSEND" and "TRCV") used by the instruction can occur.

The error codes are assigned via the instance data block of the "MB_CLIENT" instruction. The error codes are displayed for the respective instruction under STATUS in the Static section.

The meaning of the error codes is available in the documentation of the corresponding communications instruction.

See also

Description of MB_CLIENT (Page 5323)

MB_HOLD_REG parameter (Page 5335)

Switching display formats in the program status (Page 2980)

MB_SERVER: Communicating via PROFINET as a Modbus TCP server

Description of MB_SERVER

Description

The "MB_SERVER" instruction communicates as a Modbus TCP server via the PROFINET connection of the S7-1200 CPU. To use the instruction, you do not require any additional hardware module. The "MB_SERVER" instruction processes connection requests of a Modbus TCP client, receives requests from Modbus functions and sends responses.

NOTICE

Safety instruction

Note that each client on the network has access to the data of the Modbus instructions.

Although the data block for the Modbus holding register cannot be addressed directly by the client, it is possible to indirectly access the data via the offset addresses. The option is available to restrict access to an IP address to prevent unauthorized read and write operations. Note, however, that the shared address can also be used for unauthorized access.

Parameters

The following table shows the parameters of the "MB_SERVER" instruction:

Parameter	Declaration	Data type	Description
DISCONNECT	Input	BOOL	The instruction "MB_SERVER" enters into a passive connection with a partner module, which means the server responds to a TCP connection request from each requesting IP address. You can use this parameter to control when a connection request is accepted: • 0: A passive connection is established when there is no communications connection. • 1: Initialization of the connection termination. If the input is set, no other operations are executed. The value 7003 is output at the STATUS parameter after successful connection termination.
CONNECT_ID	Input	UINT	The parameter uniquely identifies a connection within the CPU. Each individual instance of the instructions "MB_CLIENT (Page 5323)" and "MB_SERVER" must have a unique CONNECT_ID parameter.
IP_PORT	Input	UINT	Start value = 502. The number of IP ports defines which IP port is monitored for connection requests of the Modbus client. The following TCP port numbers must not be used for the passive connection of the "MB_SERVER" instruction: 20, 21, 25, 80, 102, 123, 5001, 34962, 34963 and 34964.
MB_HOLD_REG (Page 5335)	InOut	VARIANT	Pointer to the Modbus holding register of the "MB_SERVER" instruction: Use a global data block with standard access as holding register. The holding register contains the values that may be accessed by a Modbus client using the Modbus functions 3 (read), 6 (write) and 16 (read).
NDR	Output	BOOL	"New Data Ready": • 0: No new data • 1: New data written by the Modbus client written
DR	Output	BOOL	"Data Read": • 0: No data read • 1: Data read by the Modbus client
ERROR	Output	BOOL	If an error occurs during the call of the "MB_SERVER" instruction, the output of the ERROR parameter is set to TRUE. Detailed information about the cause of the problem is indicated by the STATUS parameter.
STATUS (Page 5336)	Output	WORD	Error code of the instruction.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Mapping of Modbus addresses to the process image

The "MB_SERVER" instruction allows incoming Modbus functions (1, 2, 4, 5 and 15) direct read and write access to the process image input and output of the S7-1200 CPU (use of the data types BOOL and WORD).

For the data transfer of the function codes 3, 6 and 16, the holding register (MB_HOLD_REG parameter) must be defined longer than one byte. The following table shows the mapping of the Modbus addresses to the process image of the CPU.

Modbus function						S7-1200	
Code	Function	Data area	Address space			Data area	CPU address
01	Read: Bits	Output	1	to	8192	Process image output	Q0.0 to Q1023.7
02	Read: Bits	Input	10001	to	18192	Process image input	I0.0 to I1023.7
04	Read: WORD	Input	30001	to	30512	Process image input	IW0 to IW1022
05	Write: Bits	Output	1	to	8192	Process image output	Q0.0 to Q1023.7
15	Write: Bits	Output	1	to	8192	Process image output	Q0.0 to Q1023.7

Incoming Modbus messages with the function codes 3, 6 and 16 write to or read from the Modbus holding registers (you specify the holding registers with the MB_HOLD_REG parameter).

Multiple server connections

You can create multiple Server connections. This allows a single CPU to establish connections to more than one Modbus TCP client at the same time.

A Modbus TCP server can support several TCP connections and the maximum number of connections depends on the CPU being used.

The total number of connections of one CPU, including those of the Modbus TCP clients and server must not exceed the maximum number of supported connections.

Modbus TCP connections can also be shared by client and/or server connections.

In the case of Server connections, remember the following rules:

- Each "MB_SERVER" connection must use a unique instance DB.
- Each "MB_SERVER" connection must be created with a unique IP port number. Only one connection is supported for each port.
- Each "MB_SERVER" connection must use a unique connection ID.
  The relevant individual connection ID must be used for each individual instance DB of the instruction. The connection ID and instance DB belong together in pairs and must be unique for each connection.
- For each connection, the "MB_SERVER" instruction must be called individually.

Modbus diagnostics functions

The table below contains a description of the Modbus diagnostics functions.

Code	Subfunction	Description
08	0x0000	Echo test: The "MB_SERVER" instruction receives a data word and returns this unchanged to the Modbus master.
08	0x000A	Reset event counter: The "MB_SERVER" instruction resets the event counter for communication that is used for Modbus function 11.
11	-	Fetch event counter of the communication: The "MB_SERVER" instruction uses an internal event counter for communication to record the number of successfully executed read and write requests sent to the Modbus server. The event counter is not incremented by the functions 8, 11 or broadcast requests. The same applies to requests that result in a communications error (for example parity errors or CRC errors). The broadcast function is not available for Modbus TCP because only one client/server connection can exist at any one time.

Static tags of the instruction

The following table describes the static tags of the instance data block of the "MB_SERVER" instruction used in the program. You can write the HR_Start_Offset tag. You can read the other tags to monitor the Modbus status.

Tag	Data type	Start value	Description
HR_Start_Offset	WORD	0	Assign the start address of the Modbus holding register.
Request_Count	WORD	0	Total number of requests received by the server.
Server_Message_Count	WORD	0	Total number of received messages for the relevant Server.
Xmt_Rcv_Count	WORD	0	Counter for detecting the number of transfers during which an error occurred. The counter is also incremented if an invalid Modbus message is received.
Exception_Count	WORD	0	Counter for detecting the number of errors specifically for Modbus cause an exception.
Success_Count	WORD	0	Counter for detecting the number of requests that contain no error in the transferred protocol.
Connected	BOOL	0	Indicates whether the connection to the assigned client has been established or not: 1 = connected, 0 = not connected.

Example: Addressing via static tag HR_Start_Offset

The addresses of the Modbus holding register start at 40001. These addresses correspond to the address space of the CPU memory area for the holding register. You can also define the HR_Start_Offset tag so that the Modbus holding register has a start address other than 40001.

Example: The holding register begins at MW100, and has a length of 100 WORD. An offset value in the HR_Start_Offset parameter means that the start address of the holding register is moved from 40001 to 40021. Whenever the holding register is addressed below the address 40021 and above the address 40119, this causes an error.

HR_Start_Offset	Address	Minimum	Maximum
0	Modbus address (WORD)	40001	40099
	S7-1200 address	MW100	MW298
20	Modbus address (WORD)	40021	40119
	S7-1200 address	MW100	MW298

See also

MB_SERVER example: Multiple TCP connections (Page 5340)

MB_HOLD_REG parameter

Description

The MB_HOLD_REG parameter is a pointer to a data buffer for storing the data read from or written to the Modbus server. As the data buffer, you can use a global data block or a memory area (M).

As pointer to a buffer in the memory area (M), use the ANY format as follows: "P#bit address" "data type" "length" (example: P#M1000.0 WORD 500).

The following table shows examples of mapping Modbus addresses to the holding register for the Modbus functions 3 (read WORD), 6 (write WORD) and 16 (write several WORD). The upper limit for the number of addresses in the data block is decided by the maximum work memory of the CPU. If you use a memory area, the upper limit for the addresses is decided by the size of the memory area of the CPU.

Modbus addresses	MB_HOLD_REG parameter - examples		
P#M100.0 WORD 5	P#DB10.DBx0.0 WORD 5	"Recipe".ingredient	
40001	MW100	DB10.DBW0	"Recipe".ingredient[1]
40002	MW102	DB10.DBW2	"Recipe".ingredient[2]
40003	MW104	DB10.DBW4	"Recipe".ingredient[3]
40004	MW106	DB10.DBW6	"Recipe".ingredient[4]
40005	MW108	DB10.DBW8	"Recipe".ingredient[5]

See also

Description of MB_SERVER (Page 5331)

Parameter STATUS

Parameter STATUS (general status information)

STATUS* (W#16#)	Description
0000	Instruction executed without errors.
0001	Connection established.
0003	Connection terminated.
7000	No call active (REQ=0).
7001	First call with REQ=1: Processing started; BUSY has the value 1.
7002	Intermediate call (REQ irrelevant). Processing already active; BUSY has the value 1.
7003	Connection is being terminated.
7004	Connection established and monitored. No job processing active.
7005	Data was sent.
7006	Data was received.
80BB	Invalid value at ACTIVE_EST parameter (identifier for the type of connection establishment, see T_CON_PARAM): • Only passive connection establishment permitted for server (ACTIVE_EST = FALSE). • Only active connection establishment permitted for client (ACTIVE_EST = TRUE).
8380	Received Modbus frame has incorrect format or too few bytes were received.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

Parameter STATUS (parameter error)

STATUS* (W#16#)	Code of the response to the Modbus server (B#16#)	Description
8187	No response	The MB_HOLD_REG parameter has an invalid pointer. Data area is too small.
818C	No response	• The MB_HOLD_REG parameter references an an optimized data block. Either use a data block with standard access or a memory area • Error due to timeout of execution (more than 55 seconds).
8381	01	Function code is not supported.
8382	03	Error in data length
8383	02	Error in data address or access outside the address area of the holding register (MB_HOLD_REG (Page 5335) parameter).
8384	03	Error in data value
8385	03	Value of the diagnostic code is not supported (only with function code 08).
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

Note

Error codes of internally used communications instructions.

With the "MB_SERVER" instruction, in addition to the errors listed in the tables, errors caused by the communication instructions ("TCON", "TDISCON", "TSEND" and "TRCV") used by the instruction can occur.

The error codes are assigned via the instance data block of the "MB_SERVER" instruction. The error codes are displayed for the respective instruction under STATUS in the Static section.

The meaning of the error codes is available in the documentation of the corresponding communications instruction.

See also

Description of MB_SERVER (Page 5331)

Switching display formats in the program status (Page 2980)

Examples

MB_CLIENT example 1: Send several requests via a TCP connection

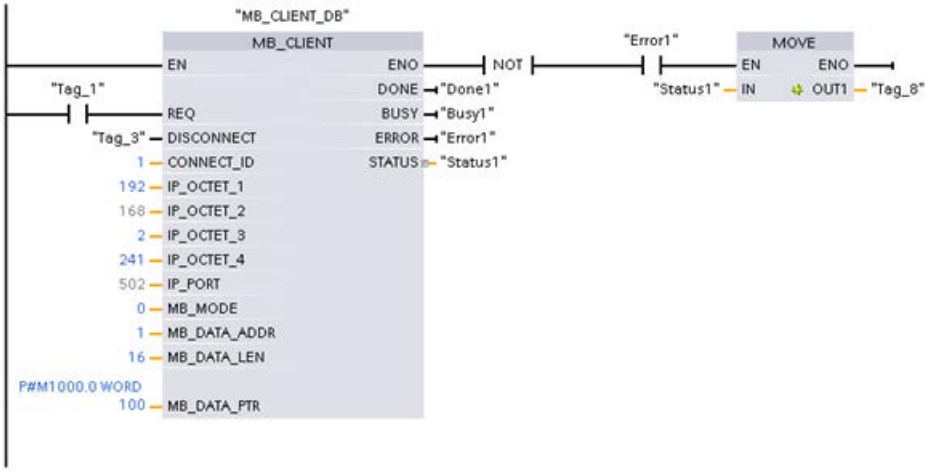
Description

Several requests of the Modbus Client can be sent via a TCP connection. To do this, use the same instance DB, the same connection ID and the same port number.

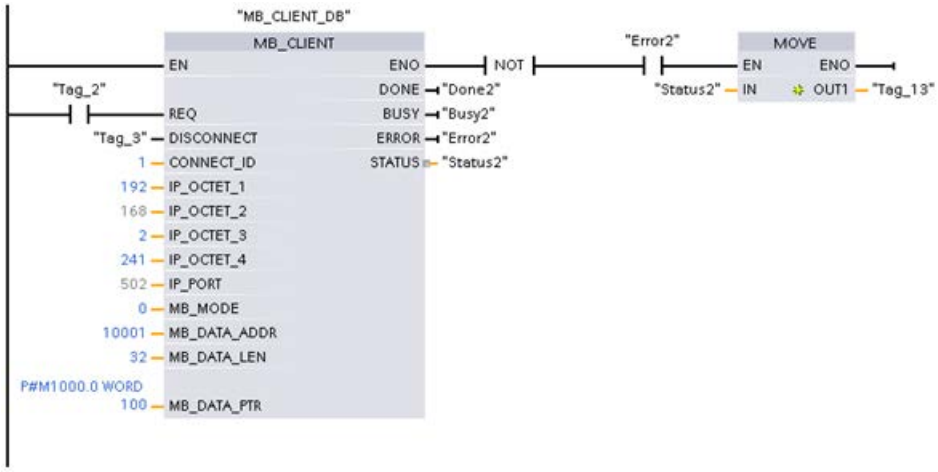
Only one client can be active at any one time. After processing of a client has been completed, the next client is processed. The order of execution must be defined in the program.

In the following sample program, the value of the STATUS output parameter is also copied.

Network 1: Modbus function 1 - 16 read output bits



Network 2: Modbus function 2 - 32 read input bits



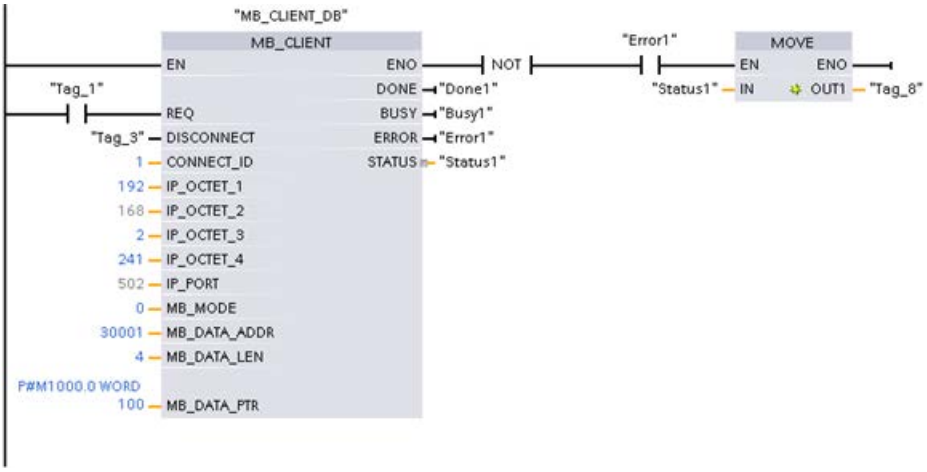
MB_CLIENT example 2: Send multiple requests via several TCP connections

Description

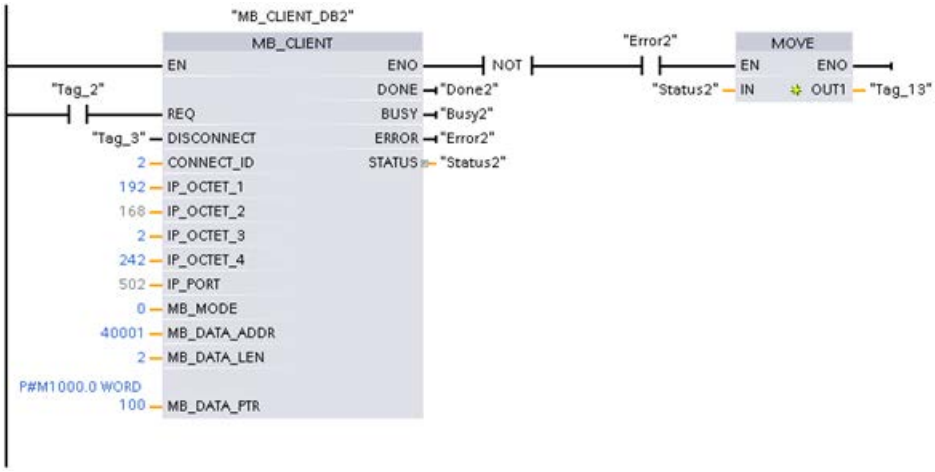
Requests from the Modbus client can be sent via different TCP connections. If you require this, use a different instance DB and a different connection ID.

Use a different port number, if the connections are to the same Modbus server. If the connections are to different Modbus servers, you can freely assign the port number.

Network 1: Modbus function 4 - read input (WORD)



Network 2: Modbus function 3 - read holding register (WORD)

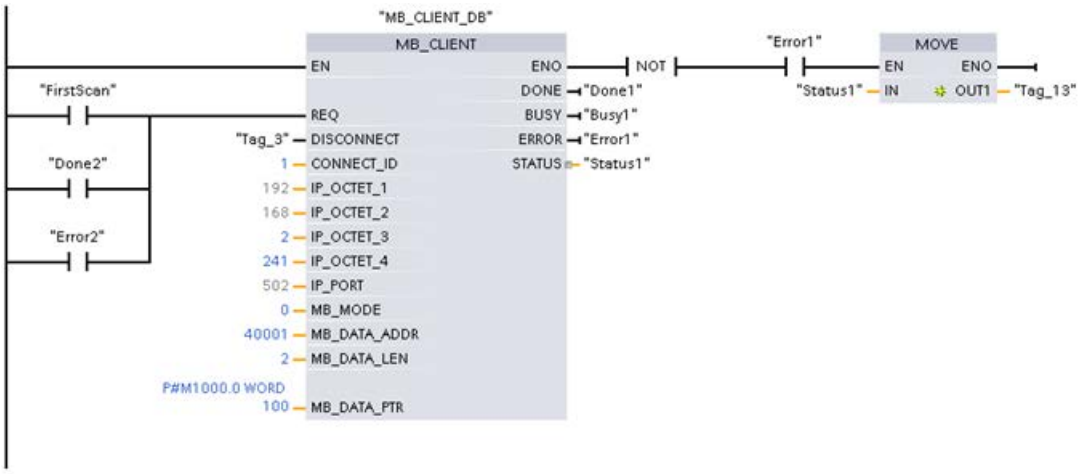


MB_CLIENT example 3: Coordinate several requests

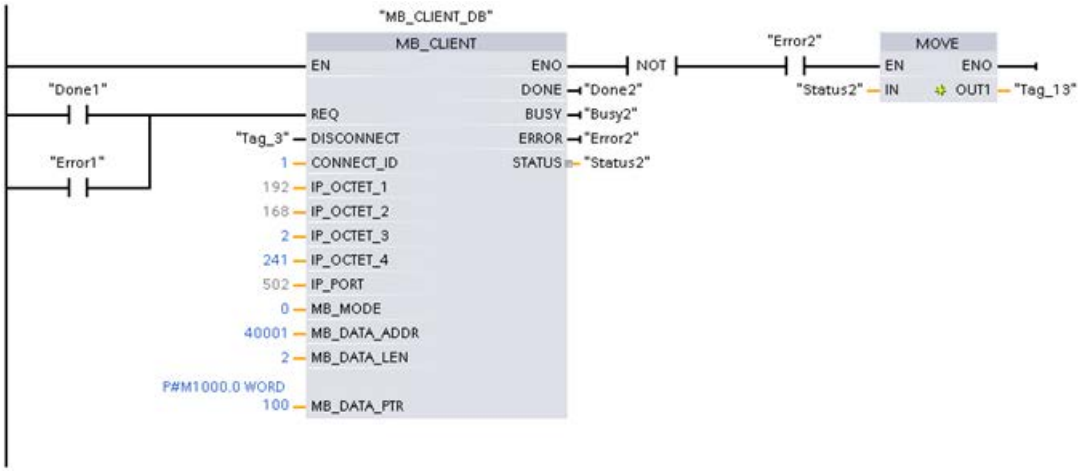
Description

Make sure that the individual Modbus requests are executed. You control coordination of requests with the program. The following example demonstrates how the output parameters of the first and second client request can be used to coordinate execution of the instructions.

Network 1: Modbus function 3 - read holding register (WORD)



Network 2: Modbus function 3 - read holding register (WORD)



MB_SERVER example: Multiple TCP connections

Description

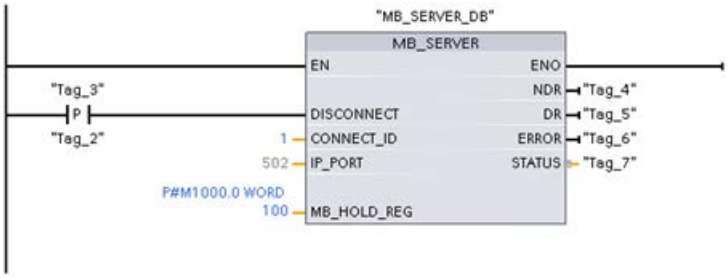
You can use several Modbus TCP server connections. To do this, the "MB_SERVER" instruction must be called independently for each connection.

Every connection requires the following:

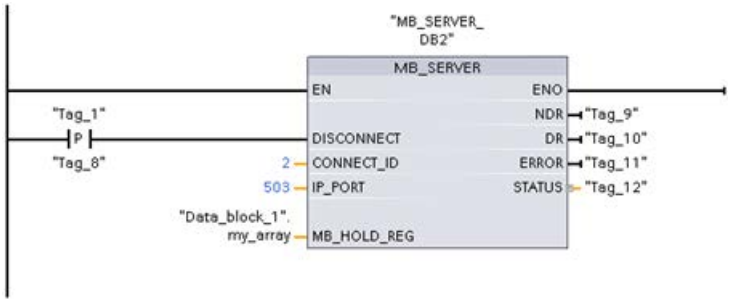
- An independent instance data block of the instruction
- A unique connection ID
- A separate IP port (on the S7-1200, only one connection is permitted per IP port)

To optimize the performance "MB_SERVER" should be executed once per program cycle for each connection.

Network 1: Connection #1 with associated IP port connection ID and instance DB



Network 2: Connection #1 with associated IP port connection ID and instance DB



MODBUS (TCP)

MB_CLIENT: Communicating via PROFINET as a Modbus TCP client

Description of MB_CLIENT

Description

The "MB_CLIENT" instruction communicates as a Modbus TCP client via the PROFINET connection. With the "MB_CLIENT" instruction, you establish a connection between the client and the server, send Modbus requests and receive responses and control connection termination of the Modbus TCP client.

The "MB_CLIENT" instruction V3.0 can be used for the S7-1500 as well as for the S7-1200 version 4.0 or higher. The connection can take place via the local interface of the CPU or CM/CP.

To use the instruction, you do not require an additional hardware module.

Multiple client connections

A Modbus TCP client can support several TCP connections and the maximum number of connections depends on the CPU being used. The total number of connections of one CPU, including those of the Modbus TCP clients and server must not exceed the maximum number of supported connections. Modbus TCP connections can also be shared by "MB_CLIENT" and/or "MB_SERVER" instances.

With individual client connections, remember the following rules:

- Each "MB_CLIENT" connection must use a unique instance DB.
- For each "MB_CLIENT" connection, a unique server IP address must be specified.
- Each "MB_CLIENT" connection requires a unique connection ID.
  The relevant individual connection ID must be used for each individual instance DB of the instruction. The connection ID and instance DB belong together in pairs and must be unique for each connection.
- Unique numbers for the IP port may or may not be required depending on the server configuration.

Parameters

The following table shows the parameters of the "MB_CLIENT" instruction:

Parameter	Declaration	Data type	Description
REQ (Page 5344)	Input	BOOL	Communication request with the Modbus TCP server The REQ parameter is level-controlled. This means that as long as the input is set (REQ=true), the instruction sends communication requests. • The instance DB for other clients is blocked with the communications request. • Changes to the input parameters will not become effective until the server has responded or an error message has been output. • If the parameter REQ is set again during an ongoing Modbus request, no additional transmission takes place afterwards.
DISCONNECT (Page 5344)	Input	BOOL	With this parameter, you control the establishment and termination of the connection to the Modbus server: • 0: Establish communications connection to the connection partner configured at the CONNECT parameter (see CONNECT parameter). • 1: Disconnect the communication connection. No other function is executed during connection termination. The value 0003 is output at the STATUS parameter after successful connection termination. The Modbus request is sent immediately if the REQ parameter is set during connection establishment.
MB_MODE (Page 5344)	Input	USINT	Selects the mode of the Modbus request (read, write or diagnostics).
MB_DATA_ADDR (Page 5344)	Input	UDINT	Start address of the data accessed by the "MB_CLIENT" instruction.

Parameter	Declaration	Data type	Description
MB_DATA_LEN	Input	UINT	Data length: Number of bits or words for the data access (see "MB_MODE and MB_DATA_ADDR parameters" - data length).
MB_DATA_PTR (Page 5346)	InOut	VARIANT	Pointer to the Modbus data register: The register is a buffer for the data received from the Modbus server or to be sent to the Modbus server. The pointer must reference a global data block with optimized access. The number of bits addressed must be divisible by 8.
CONNECT (Page 5347)	InOut	VARIANT	Pointer to the structure of the connection description The following structures (system data types) can be used: • TCON_IP_v4: Includes all address parameters required for establishing a programmed connection. When using TCON_IP_v4, the connection is established when calling the instruction "MB_SERVER". • TCON_Configured: Includes the address parameters of a configured connection. When using TCON_Configured, an existing connection is used that was created by the CPU after download of the hardware configuration.
DONE	Out	BOOL	The bit at output parameter DONE is set to "1" as soon as the last job is completed without errors.
BUSY	Out	BOOL	• 0: No Modbus request in progress • 1: Modbus request being processed The output parameter BUSY is not set during connection establishment and termination.
ERROR	Out	BOOL	• 0: No error • 1: Error occurred. The cause of error is indicated by the STATUS parameter.
STATUS (Page 5349)	Out	WORD	Detailed status information of the instruction.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Note

Consistent input data during an "MB_CLIENT" call

When a Modbus client instruction is called, the status of the input parameters is stored internally and then compared at the next call. The comparison is used to determine whether this particular call initialized the current request. Several "MB_CLIENT" calls can be executed if you use a common instance DB. The values of the input parameters must not be changed while an "MB_CLIENT" instance is being executed. If the input parameters are changed during execution, "MB_CLIENT" cannot be used to check whether or not the instance is currently being executed.

Static tags of the instruction

The following table describes the editable static tags of the instance data block of the "MB_CLIENT" instruction.

Tag	Data type	Start value	Description
Blocked_Proc_Timeout	REAL	3.0	Wait time in seconds before the static tag ACTIVE is reset if there is a blocked Modbus instance. This can, for example, occur if a client request is output and the execution of the client function aborts before the request was fully executed. The maximum wait time is 55 seconds.
MB_Transaction_ID	WORD	1	Transaction ID of the Modbus TCP protocol. The start value of "1" should only be changed if the Modbus TCP server requires a different value.
MB_Unit_ID	BYTE	255	Unit ID of the Modbus protocol. The tag corresponds to the slave address of the Modbus RTU protocol. Change this value only if the Modbus TCP server can be used as a gateway and is controlled by the application program within the Modbus server.
RCV_TIMEOUT	REAL	2.0	Time interval in seconds in which the "MB_CLIENT" instruction waits for a response from the server.
Connected	BOOL	0	Indicates whether the connection to the assigned server has been established or not: 1 = connected, 0 = not connected.

REQ and DISCONNECT parameters

Description

If no instance of the "MB_CLIENT" instruction is executing and if the value of the DISCONNECT parameter is "0", a new job executes if REQ=1. If there is not yet a connection, this is established during execution.

If the same instance of the "MB_CLIENT" instruction executes again (DISCONNECT = 0 and REQ = 1), before the active job was executed, this is not executed on completion of the active job. A new job can only be started on completion of the active job (REQ = 1).

The status of the execution is output by the output parameters. You can use it to monitor the execution status when the "MB_CLIENT" instruction is executed sequentially.

MB_MODE and MB_DATA_ADDR parameters

Description

Instead of a function code, the "MB_CLIENT" instruction uses the MB_MODE parameter. The MB_DATA_ADDR parameter is used to specify the Modbus start address of the data you want to access.

The combination of the parameters MB_MODE, MB_DATA_ADDR and MB_DATA_LEN defines the function code used in the current Modbus message. For example:

- Function code 5
  - MB_MODE=1
  - MB_DATA_ADDR=1
  - MB_DATA_LEN=1
- Function code 15
  - MB_MODE=1
  - MB_DATA_ADDR=1
  - MB_DATA_LEN=2

The following table shows the relationship between the input parameters of the "MB_CLIENT" instruction and the Modbus function.

MB_MODE	MB_DATA_ADDR	MB_DATA_LEN	Modbus function	Function and data type
0	Start address: • 1 to 9999	Data length (bits) per call: • 1 to 2000	01	Read input bits: • 1 to 2000
0	Start address: • 10001 to 19999	Data length (bits) per call: • 1 to 2000	02	Read input bits: • 1 to 2000
0	Start address: • 40001 to 49999 • 400001 to 465535	Data length (WORD) per call: • 1 to 125 • 1 to 125	03	Read holding register: • 0 to 9998 • 0 to 65534
0	Start address: • 30001 to 39999	Data length (WORD) per call: • 1 to 125	04	Read input words: • 0 to 9998
1	Start address: • 1 to 9999	Data length (bits) per call: • 1	05	Writing an output bit: • 0 to 9998
1	Start address: • 40001 to 49999 • 400001 to 465535	Data length (WORD) per call: • 1 • 1	06	Write a holding register: • 0 to 9998 • 0 to 65524
1	Start address: • 1 to 9999	Data length (bits) per call: • 2 to 1968	15	Write multiple output bits: • 0 to 9998
1	Start address: • 40001 to 49999 • 400001 to 465534	Data length (WORD) per call: • 2 to 123 • 2 to 123	16	Write multiple holding registers: • 0 to 9998 • 0 to 65534

MB_MODE	MB_DATA_ADDR	MB_DATA_LEN	Modbus function	Function and data type
2	Start address: • 1 to 9999	Data length (bits) per call: • 1 to 1968	15	Write one or more output bits: • 0 to 9998
2	Start address: • 40001 to 49999 • 400001 to 465535	Data length (WORD) per call: • 1 to 123 • 1 to 123	16	Write one or more holding registers: • 0 to 9998 • 0 to 65534
11	The MB_DATA_ADDR and MB_DATA_LEN parameters are not evaluated when this function is executed.		11	Read status word and event counter of the server: • The status word reflects the the processing status (0 - not processing, 0xFFFF - processing). • The event counter is incremented when the Modbus request was executed successfully. If an error occurred during execution of a Modbus function, a message is sent by the server but the event counter is not incremented.
80	-	Data length (WORD) per call: • 1	08	Check the server status with the diagnostic code 0x0000 (return loop test - the server sends the request back): • 1 WORD per call
81	-	Data length (WORD) per call: • 1	08	Reset the event counter of the server with the diagnostic code 0x000A: • 1 WORD per call
3 to 10, 12 to 79, 82 to 255				Reserved

MB_DATA_PTR parameter

Description

The MB_DATA_PTR parameter is a pointer to a data buffer for storing the data read from or written to the Modbus server. As the data buffer, you can use a global data block or a memory area (M).

For a buffer in the memory area (M), use a pointer in the ANY format as follows: "P#bit address" "data type" "length" (example: P#M1000.0 WORD 500).

The MB_DATA_PTR parameter uses a communications buffer:

The communications buffer is used:

- For the communication functions of the "MB_CLIENT" instruction:
  - Reading and writing of 1 bit of data of the Modbus server addresses 00001 to 09999 and 10001 to 19999.
  - Reading of 16-bit WORD data of the Modbus server addresses 30001 to 39999 and 40001 to 49999.
  - Writing of 16-bit WORD data of the Modbus server addresses 40001 to 49999.
- During data transmission (length: bit or WORD) from or to the global DB or the memory area (M) that you assigned with the MB_DATA_PTR parameter.

If you use a data block for the buffer in the MB_DATA_PTR parameter, you will need to assign data types to the DB elements.

- Use the 1-bit data type BOOL for a Modbus bit address
- Use a 16-bit data type such as WORD, UINT, INT or REAL for a Modbus WORD address.
- Use a 32-bit data type (double word) such as DWORD, DINT or REAL for two Modbus WORD addresses.
- With MB_DATA_PTR, you can also access complex DB elements such as:
  - Standard arrays
  - Structures with unique element names
  - Complex structures with unique naming of the elements and data type lengths of 16 or 32 bits.
- The data areas for the MB_DATA_PTR parameter can also be in different global data blocks (or in different memory areas). You can, for example, use a data block for the the read jobs and another one for the write jobs or a separate data block for each "MB_CLIENT" station.

CONNECT parameter

Connection descriptions at CONNECT parameter

Two different connection descriptions can be used for the "MB_CLIENT" instruction:

- Programmed connections with the structure TCON_IP_v4
  The connection parameters are stored in the TCON_IP_v4 structure and the connection is set up with the call of the instruction "MB_CLIENT".
- Configured connections with the structure TCON_Configured
  The configured connection has already been established by the CPU. Use the structure TCON_Configured to specify which existing connection is to be used for the instruction.

Each instance of the instruction "MB_CLIENT" requires a unique connection. Create a separate structure TCON_IP_v4 or TCON_Configured for each instance of the instruction to describe the connection.

Connection description for programmed connections

Use the following structure for connection description to TCON_IP_v4 for programmed connections at the CONNECT parameter:

- Make sure that only connections of the type TCP are specified in the TCON_IP_v4 structure.
- The connection may not use the following TCP port numbers: 20, 21, 25, 80, 102, 123, 5001, 34962, 34963 and 34964.

Byte	Parameter	Data type	Start value	Description
0 … 1	InterfaceID	HW_ANY	-	Hardware identifier of the local interface (value range: 0 to 65535).
2 … 3	ID	CONN_OUC	-	Reference to this connection (value range: 1 to 4095). The parameter uniquely identifies a connection within the CPU. Each individual instance of the instruction "MB_CLIENT" must use a unique ID.
4	ConnectionType	BYTE	11	Connection type Select 11 (decimal) for TCP. Other connection types are not permitted. If another connection type (e.g. UDP) is used, a corresponding error message is output at the STATUS parameter of the instruction.
5	ActiveEstablished	BOOL	TRUE	ID for the manner in which the connection is established Select TRUE for active connection establishment.
6 … 9	RemoteAddress	ARRAY [1..4] of BYTE	-	IP address of the connection partner (Modbus server), for example, for 192.168.0.1: • addr[1] = 192 • addr[2] = 168 • addr[3] = 0 • addr[4] = 1
10 … 11	RemotePort	UINT	502	Port number of the remote connection partner (value range: 1 to 49151). Use the IP port number of the server to which the client establishes the connection and communicates using the TCP/IP protocol (default value: 502).
12 … 13	LocalPort	UINT	0	Port number of the local connection partner: • Port numbers: 1 to 49151 • Any port: "0"

Note

Migration of the instruction "MB_CLIENT" version 2.1

The parameters CONNECT_ID, IP_PORT and IP_OCTET_x are mapped in version 3.0 of the "MB_CLIENT" instruction in the TCON_IP_v4 structure:

- The parameter CONNECT_ID of the "MB_CLIENT" V2.1 instruction corresponds to the ID parameter of TCON_IP_v4.
- The parameter IP_PORT of the "MB_CLIENT" V2.1 instruction corresponds to the RemotePort parameter of TCON_IP_v4.
- The four parameters IP_OCTET_x of the "MB_CLIENT" V2.1 instruction correspond to the array of the RemoteAddress parameter of TCON_IP_v4.

Connection description for configured connections

For programmed connections at the CONNECT parameter, use the following structure for connection description to TCON_Configured.

- Make sure that only connections of the type TCP are specified in the TCON_Configured structure.
- The connection may not use the following TCP port numbers: 20, 21, 25, 80, 102, 123, 5001, 34962, 34963 and 34964.

Byte	Parameter	Data type	Start value	Description
0 … 1	InterfaceId	HW_ANY	-	Hardware identifier of the local interface (value range: 0 to 65535).
2 … 3	ID	CONN_OUC	-	Reference to this connection (value range: 1 to 4095). Enter the connection ID of the existing connection.
4	ConnectionType	BYTE	0	Connection type Select 254 (decimal) for a configured connection.

Parameter STATUS

Parameter STATUS (general status information)

STATUS* (W#16#)	Description
0000	Instruction executed without errors.
0001	Connection established.
0003	Connection terminated.
7000	No call active and no connection established (REQ=0).
7001	Connection establishment triggered.
7002	Intermediate call. Connection is being established.
7003	Connection is being terminated.

STATUS* (W#16#)	Description
7004	Connection established and monitored. No job processing active.
7005	Data is being sent.
7006	Data is being received.
80BB	Invalid value at ActiveEstablished parameter (identifier for the type of connection establishment, see CONNECT parameter (Page 5347)): • Only passive connection establishment permitted for server (ActiveEstablished = FALSE). • Only active connection establishment permitted for client (ActiveEstablished = TRUE).
8380	Received Modbus frame has incorrect format or too few bytes were received.
* The status codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

Parameter STATUS (protocol error)

STATUS* (W#16#)	Error code in the error message from MB_SERVER (B#16#)	Description
8381	01	Function code is not supported.
8382	03	Error in data length: • Invalid length specification in received Modbus frame. • The length of the Modbus frame in the frame header does not match the number of received bytes. • The number of bytes does not match the number of actually transmitted bytes (only functions 1-4). • The received start address does not match the one sent originally (functions 5, 6, 15, 16). • The number of words does not match the number of actually transmitted words (only functions 15 and 16).
8383	02	Error reading or writing data or access outside the address area of MB_DATA_PTR (Page 5346). The error can occur locally as well as with the instruction "MB_SERVER".
8384	03	Error in data value: • Error in data value for function 5 (error on server). • A different data value was received than was originally sent by the client (functions 5 and 6) (local error). • Invalid exception code received.
8385	03	Diagnostic code is not supported (function code 08). The error can occur locally as well as on the server.
8386	-	Received function code does not match the one sent originally.
8387	-	• The assigned connection ID is different from that used for previous requests. Only one connection ID can be used for each instance DB of the "MB_CLIENT" instruction. • The error code is also output when the protocol ID of the Modbus TCP frame received by the server is not "0".

STATUS* (W#16#)	Error code in the error message from MB_SERVER (B#16#)	Description
8388	15 or 16	The Modbus server sent a different data length than was requested. This error occurs only when using the Modbus functions 15 or 16.
* The status codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

Parameter STATUS (parameter error)

STATUS* (W#16#)	Description
80B6	Invalid connection type, only TCP connections are supported.
80C8	No response of the server in the defined period. Check the connection to the Modbus server. This error is only reported on completion of the configured repeated attempts. If the "MB_CLIENT" instruction does not receive an answer with the originally transferred transaction ID (see static MB_TRANSACTION_ID tag) within the defined period, this error code is output.
8188	The MB_MODE parameter has an invalid value.
8189	Invalid addressing of data at the MB_DATA_ADDR parameter.
818A	Invalid data length at the MB_DATA_LEN parameter.
818B	The MB_DATA_PTR parameter has an invalid pointer. You should also check the values of the MB_DATA_ADDR (Page 5346) and MB_DATA_LEN parameters.
818C	Timeout at parameter BLOCKED_PROC_TIMEOUT or RCV_TIMEOUT (see static tags of instruction). The limit of 55 seconds has been exceeded.
818D	The transaction ID (MB_TRANSACTION_ID tag) does not correspond to the one sent originally (see static tags of instruction).
8200	• A different Modbus request is currently being processed via the port. • Another instance of MB_CLIENT with the same connection parameters is processing an existing Modbus request.
* The status codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

Note

Error codes of internally used communications instructions

With the "MB_CLIENT" instruction, in addition to the errors listed in the tables, errors caused by the communication instructions ("TCON", "TDISCON", "TSEND", "TRCV", "T_DIAG" and "TRESET") used by the instruction can occur.

The error codes are assigned via the instance data block of the "MB_CLIENT" instruction. The error codes are displayed for the respective instruction under STATUS in the "Static" section.

The meaning of the error codes is available in the documentation of the corresponding communications instruction.

See also

Switching display formats in the program status (Page 2980)

GET_ERR_ID: Get error ID locally (Page 3392)

MB_SERVER: Communicating via PROFINET as a Modbus TCP server

Description of MB_SERVER

Description

The "MB_SERVER" instruction communicates as Modbus TCP server via a PROFINET connection. The "MB_SERVER" instruction processes connection requests of a Modbus TCP client, receives and processes Modbus requests and sends responses.

The "MB_SERVER" instruction V3.0 can be used for the S7-1500 as well as for the S7-1200 version 4.0 or higher. The connection can take place via the local interface of the CPU or CM/CP.

To use the instruction, you do not require an additional hardware module.

NOTICE
Safety instruction Note that each client on the network has access to the data of the Modbus instructions. Although the data block for the Modbus holding register cannot be addressed directly by the client, it is possible to indirectly access the data via the offset addresses. The option is available to restrict access to an IP address to prevent unauthorized read and write operations. Note, however, that the shared address can also be used for unauthorized access.

Multiple server connections

You can create multiple Server connections. This allows a single CPU to accept connections from multiple Modbus TCP clients at the same time.

A Modbus TCP server can support several TCP connections and the maximum number of connections depends on the CPU being used.

The total number of connections of one CPU, including those of the Modbus TCP clients and server must not exceed the maximum number of supported connections.

Modbus TCP connections can also be shared by "MB_CLIENT" and/or "MB_SERVER" instances.

In the case of Server connections, remember the following rules:

- Each "MB_SERVER" connection must use a unique instance DB.
- Each "MB_SERVER" connection must use a unique connection ID.
  The relevant individual connection ID must be used for each individual instance DB of the instruction. The connection ID and instance DB belong together in pairs and must be unique for each connection.
- For each connection, the "MB_SERVER" instruction must be called individually.

Parameters

The following table shows the parameters of the "MB_SERVER" instruction:

Parameter	Declaration	Data type	Description
DISCONNECT	Input	BOOL	The "MB_SERVER" instruction is used to enter into a passive connection with a partner module. The server responds to a connection request from the IP address which is entered in the SDT "TCON_IP_v4" in the CONNECT parameter. You can use this parameter to control when a connection request is accepted: • 0: A passive connection is established when there is no communications connection. • 1: Initialization of the connection termination. If the input is set, no other operations are executed. The value 0003 is output at the STATUS parameter after successful connection termination.
MB_HOLD_REG (Page 5356)	InOut	VARIANT	Pointer to the Modbus holding register of the "MB_SERVER" instruction The holding register contains the values that may be accessed by a Modbus client using the Modbus functions 3 (read), 6 (write) and 16 (read). As the holding register, use either a global data block with optimized access or the memory area of the bit memories.
CONNECT (Page 5356)	InOut	VARIANT	Pointer to the structure of the connection description The following structures (SDTs) can be used: • TCON_IP_v4: Includes all address parameters required for establishing a programmed connection. The default address is 0.0.0.0 (any IP address), but you can enter a specific IP address so that the server only responds to requests from this address. When using TCON_IP_v4, the connection is established when calling the instruction "MB_SERVER". • TCON_Configured: Includes the address parameters of a configured connection. When using TCON_Configured, the connection is established by the CPU after download of the hardware configuration.
NDR	Output	BOOL	"New Data Ready": • 0: No new data • 1: New data written by the Modbus client written
DR	Output	BOOL	"Data Read": • 0: No data read • 1: Data read by the Modbus client

Parameter	Declaration	Data type	Description
ERROR	Output	BOOL	If an error occurs during the call of the "MB_SERVER" instruction, the output of the ERROR parameter is set to "1". Detailed information about the cause of the problem is indicated by the STATUS parameter.
STATUS (Page 5358)	Output	WORD	Detailed status information of the instruction.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

Static tags of the instruction

The following table describes the static tags of the instance data block of the "MB_SERVER" instruction used in the program. You can write the HR_Start_Offset tag. You can read the other tags to monitor the Modbus status.

Tag	Data type	Start value	Description
HR_Start_Offset	WORD	0	Assign the start address of the Modbus holding register.
Request_Count	WORD	0	Total number of requests received by the server.
Server_Message_Count	WORD	0	Total number of received messages for the relevant Server.
Xmt_Rcv_Count	WORD	0	Counter for detecting the number of transfers during which an error occurred. The counter is only incremented when an invalid Modbus request is received.
Exception_Count	WORD	0	Counters for detecting the number of errors specifically for Modbus which cause an error message to "MB_CLIENT".
Success_Count	WORD	0	Event counter for detecting the number of requests that were successfully executed by the server.
Connected	BOOL	0	Indicates whether the connection to the assigned client has been established or not: 1 = connected, 0 = not connected.

Mapping of Modbus addresses to the process image

The "MB_SERVER" instruction allows incoming Modbus functions (1, 2, 4, 5 and 15) direct read and write access to the process image inputs and outputs of the CPU (use of the data types BOOL and WORD).

For the data transfer of the function codes 3, 6 and 16, the holding register (MB_HOLD_REG parameter) must be defined longer than one byte. The following table shows the mapping of the Modbus addresses to the process image of the CPU.

Modbus function						S7-1500, S7-1200 V4.0	
Function code	Function	Data area	Address space			Data area	CPU address
01	Read: Bits	Output	1	to	9999	Process image output	Q0.0 to Q1249.6
02	Read: Bits	Input	1	to	19999	Process image input	I0.0 to I1249.6
04	Read: WORD	Input	1	to	39999	Process image input	IW0 to IW19996

Modbus function						S7-1500, S7-1200 V4.0	
05	Write: Bits	Output	1	to	9999	Process image output	Q0.0 to Q1249.6
15	Write: Bits	Output	1	to	9999	Process image output	Q0.0 to Q1249.6

Incoming Modbus requests with the function codes 3, 6, 16 and 23 write to or read from the Modbus holding register (you specify the holding register with the MB_HOLD_REG parameter).

Example: Addressing via static tag HR_Start_Offset

The addresses of the Modbus holding register start at 40001. These addresses correspond to the address space of the CPU memory area for the holding register. You can also define the HR_Start_Offset tag so that the Modbus holding register has a start address other than 40001.

Example: The holding register begins at MW100, and has a length of 100 WORD. An offset value in the HR_Start_Offset parameter means that the start address of the holding register is moved from 40001 to 40021. Whenever the holding register is addressed below the address 40021 and above the address 40120, this causes an error.

HR_Start_Offset	Address	Minimum	Maximum
0	Modbus address (WORD)	40001	40100
	CPU address	MW100	MW298
20	Modbus address (WORD)	40021	40120
	CPU address	MW100	MW298

Modbus diagnostics functions

The table below contains a description of the Modbus diagnostics functions.

Function code	Diagnostic code	Description
08	0x0000	Echo test: The "MB_SERVER" instruction receives a data word and returns this unchanged to the Modbus client.
08	0x000A	Reset event counter: The "MB_SERVER" instruction resets the following event counters: "Success_Count", "Xmt_Rcv_Count", "Exception_Count", "Server_Message_Co" and "Request_Count".
11	-	Fetch event counter of the communication: The "MB_SERVER" instruction uses an internal event counter for communication to record the number of successfully executed read and write requests sent to the Modbus server. The event counter is not incremented with the functions 8 or 11. The same holds true for requests that cause a communications error, for example, if a protocol error has occurred (e.g., the function code in the received Modbus request is not supported).

MB_HOLD_REG parameter

Description

The MB_HOLD_REG parameter is a pointer to a data buffer for storing the data read from or written to the Modbus server. You can use a global data block or a bit memory (M) as memory area.

- The high limit for the number of addresses in the data block (D) is determined by the maximum work memory of the CPU.
- The high limit for the number of bit memories (M) is determined by the size of the CPU memory area.

The following table shows examples of mapping Modbus addresses to the holding register for the Modbus functions 3 (read WORD), 6 (write WORD), 16 (write several WORD) and 23 (write and read several words).

Modbus addresses	MB_HOLD_REG parameter - examples		
40001	MW100	DB10.DBW0	"Recipe".ingredient[1]
40002	MW102	DB10.DBW2	"Recipe".ingredient[2]
40003	MW104	DB10.DBW4	"Recipe".ingredient[3]
40004	MW106	DB10.DBW6	"Recipe".ingredient[4]
40005	MW108	DB10.DBW8	"Recipe".ingredient[5]

CONNECT parameter

Connection descriptions at CONNECT parameter

Two different connection descriptions can be used for the "MB_SERVER" instruction:

- Programmed connections with the structure TCON_IP_v4
  The connection parameters are stored in the TCON_IP_v4 structure and the connection is set up with the call of the instruction "MB_SERVER".
- Configured connections with the structure TCON_Configured
  The configured connection has already been established by the CPU. Use the structure TCON_Configured to specify which existing connection is to be used for the instruction.

Each instance of the instruction "MB_SERVER" requires a unique connection. Create a separate structure TCON_IP_v4 or TCON_Configured for each instance of the instruction to describe the connection.

Connection description for programmed connections

For programmed connections at the CONNECT parameter, use the following structure for connection description to TCON_IP_v4.

- Make sure that only connections of the type TCP are specified in the TCON_IP_v4 structure.
- The connection may not use the following TCP port numbers: 20, 21, 25, 80, 102, 123, 5001, 34962, 34963 and 34964.

Byte	Parameter	Data type	Start value	Description
0 … 1	InterfaceID	HW_ANY	-	Hardware identifier of the local interface (value range: 0 to 65535).
2 … 3	ID	CONN_OUC	-	Reference to this connection (value range: 1 to 4095). The parameter uniquely identifies a connection within the CPU. Each individual instance of the instruction "MB_SERVER" must use a unique ID.
4	ConnectionType	BYTE	11	Connection type Select 11 (decimal) for TCP. Other connection types are not permitted. If another connection type (e.g. UDP) is used, a corresponding error message is output at the STATUS parameter of the instruction.
5	ActiveEstablished	BOOL	FALSE	ID for the manner in which the connection is established Select FALSE for passive connection establishment.
6 … 9	RemoteAddress	ARRAY [1..4] of BYTE	0.0.0.0	IP address of the connection partner, for example, for 192.168.0.1: • addr[1] = 192 • addr[2] = 168 • addr[3] = 0 • addr[4] = 1 If the instruction "MB_SERVER" is to accept connection requests from any connection partner, use "0.0.0.0" as IP address.
10 … 11	RemotePort	UINT	0	Port number of the remote connection partner (value range: 1 to 49151). If the instruction "MB_SERVER" is to accept connection requests from any remote partner, use "0" as port number.
12 … 13	LocalPort	UINT	502	Port number of the local connection partner (value range: 1 to 49151). The number of the IP port defines which IP port is monitored for connection requests of the Modbus client. The following TCP port numbers must not be used for the passive connection of the "MB_SERVER" instruction: 20, 21, 25, 80, 102, 123, 5001, 34962, 34963 and 34964.

Note

Migration of the instruction "MB_SERVER" version 2.1

The parameters CONNECT_ID and IP_PORT are mapped in version 3.0 of the "MB_SERVER" instruction in the TCON_IP_v4 structure:

- The parameter CONNECT_ID of the "MB_SERVER" V2.1 instruction corresponds to the ID parameter of TCON_IP_v4.
- The parameter IP_PORT of the "MB_SERVER" V2.1 instruction corresponds to the LocalPort parameter of TCON_IP_v4.

Connection description for configured connections

For configured connections at the CONNECT parameter, use the following structure for connection description to TCON_Configured.

- Make sure that only connections of the type TCP are specified in the TCON_Configured structure.
- The connection may not use the following TCP port numbers: 20, 21, 25, 80, 102, 123, 5001, 34962, 34963 and 34964.

Byte	Parameter	Data type	Start value	Description
0 … 1	InterfaceID	HW_ANY	-	Hardware identifier of the local interface (value range: 0 to 65535).
2 … 3	ID	CONN_OUC	-	Reference to this connection (value range: 1 to 4095). Enter the connection ID of the existing connection.
4	ConnectionType	BYTE	-	Connection type Select 254 (decimal) for a configured connection.

Parameter STATUS

Parameter STATUS (general status information)

STATUS* (W#16#)	Description
0000	Instruction executed without errors.
0001	Connection established.
0003	Connection terminated.
7000	No call active (REQ=0).
7001	First call. Connection establishment triggered.
7002	Intermediate call. Connection is being established.
7003	Connection is being terminated.
7005	Data is being sent.
7006	Data is being received.

STATUS* (W#16#)	Description
80BB	Invalid value at ActiveEstablished parameter (identifier for the type of connection establishment, see CONNECT parameter (Page 5356)): • Only passive connection establishment permitted for server (active_established = FALSE). • Only active connection establishment permitted for client (active_established = TRUE).
8380	Received Modbus frame has incorrect format or too few bytes were received.
* The status codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".	

Parameter STATUS (parameter error)

STATUS* (W#16#)	Error code in the error message from "MB_SERVER" (B#16#)	Description
8187	No response	The MB_HOLD_REG parameter has an invalid pointer. Data area is too small.
8381	01	Function code is not supported.
8382	03	Error in data length: • Invalid length specification in received Modbus frame • The frame length entered in the header of the Modbus frame does not match the number of actually received bytes. • The number of bytes entered in the header of the Modbus frame does not match the number of actually received bytes (functions 15 and 16).
8383	02	Error in data address or access outside the address area of the holding register (MB_HOLD_REG (Page 5356) parameter).
8384	03	Invalid data value (function 5).
8385	03	Diagnostic code is not supported (only with function code 08).
* The status codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".		

Note

Error codes of internally used communications instructions

With the "MB_SERVER" instruction, in addition to the errors listed in the tables, errors caused by the communication instructions ("TCON", "TDISCON", "TSEND", "TRCV", "T_DIAG" and "T_RESET") used by the instruction can occur.

The error codes are assigned via the instance data block of the "MB_SERVER" instruction. The error codes are displayed for the respective instruction under STATUS in the "Static" section.

The meaning of the error codes is available in the documentation of the corresponding communications instruction.

See also

Switching display formats in the program status (Page 2980)

GET_ERR_ID: Get error ID locally (Page 3392)

TeleService

TM_MAIL: Transfer email

Description of TM_MAIL

Description

The "TM_MAIL" instruction works asynchronously, in other words, its execution extends over multiple calls. You must specify an instance when you call the instruction "TM_MAIL". The attribute "retentive" may not be set in the instance. This attribute ensures that the instance is initialized when the CPU switches from STOP to RUN and that a new e-mail send job can be triggered afterwards.

You start the sending of an e-mail with an edge change from "0" to "1" for the REQ parameter. The job status is indicated by the output parameters BUSY , "DONE", "ERROR", "STATUS" and "SFC_STATUS". "SFC_STATUS" corresponds in this case to the "STATUS" output parameter of the called communication blocks.

The output parameters DONE, ERROR, STATUS, and SFC_STATUS are each displayed for only one cycle if the status of the BUSY output parameter changes from "1" to "0". The following table shows the relationship between BUSY, DONE, and ERROR. Using this table, you can determine the current status of the instruction "TM_MAILand when the sending of the e-mail is complete.

DONE	BUSY	ERROR	Description
0	1	0	The job is being processed.
1	0	0	Job successfully completed.
0	0	1	The job ended with an error. The cause of the error can be found in the STATUS and SFC_STATUS parameters.
0	0	0	The "TM_MAIL" instruction was not assigned a (new) job.

If the CPU changes to STOP mode while "TM_MAIL" is active, the communication connection to the mail server aborts. The communication connection to the mail server will also be lost if

communication problems occur on the Industrial Ethernet bus. In this case, the transfer of the e-mail will be interrupted and it will not reach its recipient.

NOTICE
Changing user programs You can change the parts of your user program that directly affect calls of "TM_MAIL" only: • when the CPU is in "STOP" mode or • when no mail is being sent (REQ = 0 and BUSY = 0). This relates, in particular, to deleting and replacing program blocks that contain "TM_MAIL" calls or calls for the instance of "TM_MAIL". Ignoring this restriction can tie up connection resources. The automation system can change to an undefined status for the TPC/IP communication functions via Industrial Ethernet. A warm or cold restart of the CPU is required after the changes are transferred.

Data consistency

The ADDR_MAIL_SERVER input parameter of the instruction is taken from the "TM_MAIL" instruction again each time the sending of an e-mail is triggered. If a change is made during operation, the "new" value only becomes effective once an e-mail is triggered again.

In contrast, the WATCH_DOG_TIME, TO_S, CC, FROM, SUBJECT, TEXT, ATTACHMENT, and, if applicable, USERNAME and PASSWORD parameters are taken from the "TM_MAIL" instruction while it is running, which means that they may only be changed after the job is complete (BUSY = 0)

Setting the parameters of the TS Adapter IE

On the TS Adapter IE, you need to specify parameters for outgoing calls in such a way as to enable the TS Adapter IE to establish a connection to the dial-up server of your service provider.

If you set "When required" for connection establishment, the connection will only be established when an e-mail needs to be sent.

Connection establishment can take longer (approx. one minute) for an analog modem connection. When you specify the WATCH_DOG_TIME parameter, remember to allow for the time required to establish the connection.

Parameters

The following table shows the parameters of the "TM_MAIL" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Control parameter REQUEST: Activates the sending of an e-mail on a rising edge.
ID	Input	CONN_OUC (Word)	D, L or constant	Reference to the connection to be established. See ID parameter of the TCON (Page 5083), TDISCON (Page 5091), TSEND (Page 5096), and TRCV (Page 5099) instructions. A number that is not used for any additional instances of this instruction in the user program must be entered here.
TO_S (Page 5364)	Input	STRING	D	Input parameter for receiver addresses: STRING with a maximum length of 240 characters (see example call).
CC (Page 5364)	Input	STRING	D	Input parameter for CC recipient addresses (optional): STRING with a maximum length of 240 characters (see example call). If an empty string is assigned here, the e-mail is not sent to a CC recipient.
SUBJECT	Input	STRING	D	Input parameter for subject of the e-mail: STRING with a maximum length of 240 characters.
TEXT	Input	STRING	D	Input parameter for text of the e-mail (optional): Reference to a data string with a maximum length of 240 characters. If an empty string is assigned at this parameter, the e-mail is sent without text.
ATTACHMENT	Input	VARIANT	I, Q, M, D, L	Input parameter for e-mail attachment (optional): Reference to a byte/word/double word field with a maximum length of 65534 bytes. If no value is assigned, the e-mail is sent without an attachment.
DONE	Output	BOOL	I, Q, M, D, L	• DONE = 0: Job not yet started or still executing. • DONE = 1: Job was executed without errors.
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY = 1: E-mail transmission is not yet complete. • BUSY=0: The processing of "TM_MAIL" was stopped.
ERROR	Output	BOOL	I, Q, M, D, L	ERROR=1: An error occurred during processing. STATUS and SFC_STATUS supply detailed information about the type of error.
STATUS (Page 5365)	Output	WORD	I, Q, M, D, L	Output/status parameter STATUS: Return value or error information of the "TM_MAIL" instruction.

Parameter	Declaration	Data type	Memory area	Description
ADDR_MAIL_SERVER	Static*	DWORD	I, Q, M, D, L	IP address input parameter of the mail server: Specify as a data word in HEX format, for example: IP address = 192.168.0.200. ADDR_MAIL_SERVER = DW#16#C0A800C8, where: • 192 = 16#C0, • 168 =16#A8 • 0 = 16#00 and • 200 = 16#C8.
WATCH_DOG_TIME	Static*	TIME	I, Q, M, D, L	Input parameter for max. period of time: The "TM_MAIL" instruction should establish a connection within the period specified by WATCH_DOG_TIME. The block is exited with an error if this period is exceeded. The time before the block is exited and the error is output can exceed the WATCH_DOG_TIME because connection termination also takes a certain amount of time. To begin with, you should set a period of 2 minutes. If you have an ISDN telephone connection, a significantly shorter period can be selected.
USERNAME	Static*	STRING	D	Input parameter for user name: STRING with a maximum length of 180 characters. A user name is an absolute requirement for authentication.
PASSWORD	Static*	STRING	D	Input parameter for password: STRING with a maximum length of 180 characters. A password is an absolute requirement for authentication.
FROM (Page 5364)	Static*	STRING	D	Input parameter for sender address: STRING with a maximum length of 240 characters (see example call).
SFC_STATUS (Page 5365)	Static*	WORD	I, Q, M, D, L	Output/status parameter "SFC_STATUS": Error information of the called communication blocks.
*The values of the parameter are not modified at every call of the instruction "TM_MAIL". The values are in the static parameters of the instance and are only written once at the first call of the instruction.				

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

Note

Optional parameters

The optional parameters CC, TEXT, and ATTACHMENT are only sent with the e-mail if the corresponding parameters contain a string of length > 0.

SMTP authentication

Authentication refers here to a procedure for ensuring an identity, for example, by a password query.

The "TM_MAIL" instruction supports the SMTP authentication procedure AUTH-LOGIN that is requested by most mail servers. For information about the authentication procedure of your mail server, please refer to your mail server manual or the website of your Internet service provider.

To use the AUTH-LOGIN authentication procedure, the "TM_MAIL" instruction requires the user name with which it can log onto the mail server. This user name corresponds to the user name with which you set up a mail account on your mail server. It is made available to the "TM_MAIL" instruction in the USERNAME parameter.

To log on, the "TM_MAIL" instruction also requires the associated password. This password corresponds to the password you specified when you set up your mail account. It is made available to the "TM_MAIL" instruction in the PASSWORD parameter.

The user name and the password are each transferred to the mail server unencrypted (BASE64-coded).

If no user name is specified in the DB, the AUTH-LOGIN authentication procedure is not used. The e-mail is then sent without authentication.

Parameters TO_S, CC, and FROM

Description

The TO_S, CC, and FROM parameters are strings with, for example, the following content:

- TO: <wenna@mydomain.com>, <ruby@mydomain.com>,
- CC: <admin@mydomain.com>, <judy@mydomain.com>,
- FROM: <admin@mydomain.com>

Note the following rules when entering the parameters:

- The "TO:", "CC:", and "FROM:" characters must be entered.
- A space and an opening pointed bracket "<" must be entered before each address.
- A closing pointed bracket ">" must be entered after each address.
- A comma must be entered after each address in TO and CC.
- Only one e-mail may be entered in FROM, and there must not be a comma at the end of this address

Because of runtime and memory space, the "TM_MAIL" instruction does not perform any syntax check of the parameters TO_S, CC and FROM.

STATUS and SFB_STATUS parameters

Description

The return values of the "TM_MAIL" instruction can be classified as follows:

- W#16#0000: "TM_MAIL" was completed successfully
- W#16#7xxx: Status of "TM_MAIL"
- W#16#8xxx: An error was reported during the internal call of a communication block or from the mail server.

The following table shows the return values of "TM_MAIL" except for error codes of the internally called communication blocks.

Return value STATUS* (W#16#...):	Return value SFB_STATUS (W#16#...):	Explanation	Notes
0000	-	The processing of "TM_MAIL" was completed without error.	A without error completion of "TM_MAIL" does not mean that the sent e-mail will necessarily arrive (see below - note on point 1)
7001		"TM_MAIL" is active (BUSY = 1).	Initial call; job has started
7002	7002	"TM_MAIL" is active (BUSY = 1).	Intermediate call; job already active
8xxx	xxxx	The processing of "TM_MAIL" was completed with an error code of the internally called communication instructions.	For detailed information on the evaluation of the SFB_STATUS parameter, refer to the descriptions of the STATUS parameter of the communication instructions.
8010	xxxx	Error during connection establishment.	For further information on the evaluation of the SFB_STATUS parameter, refer to the descriptions of the STATUS parameter of the "TCON (Page 5083)" instruction.
8011	xxxx	Error sending the data.	For further information on the evaluation of SFB_STATUS, refer to the descriptions of the STATUS parameter of the "TSEND (Page 5096)" instruction.
8012	xxxx	Error receiving the data.	For further information on the evaluation of SFB_STATUS, refer to the descriptions of the STATUS parameter of the "TRCV (Page 5099)" instruction.
8013	xxxx	Error during connection establishment.	For further information on the evaluation of SFB_STATUS, refer to the descriptions of the STATUS parameter of the "TCON (Page 5083)" and "TDISCON (Page 5091)" instructions.

Return value STATUS* (W#16#...):	Return value SFB_STATUS (W#16#...):	Explanation	Notes
8014	-	Establishment of a connection is not possible.	You have possibly entered an incorrect mail server IP address (ADDR_MAIL_SERVER) or a time span that is too short (WATCH_DOG_TIME) to establish the connection. It is also possible that the CPU has no connection to the network or that the CPU configuration is incorrect.
8016	-	Error copying the attachment	-
82xx, 84xx, or 85xx	-	The error message originates from the mail server and corresponds, except for the "8", to the error number of the SMTP protocol. The following columns list several error codes that can occur:	See point 2 in the note.
8450	-	Action not executed: Mailbox not available/ not reachable.	Try again later.
8451	-	Action aborted: Local processing error	Try again later.
8500	-	Syntax error: Error not recognized. This also includes the error when a command string is too long. This can also be occur when the e-mail server does not support the LOGIN authentication procedure.	Check the parameters of "TM_MAIL". Try to send an e-mail without authentication. To do this, replace the USERNAME parameter with an empty string.
8501	-	Syntax error: Parameter or argument incorrect	You have possibly entered an incorrect address in TO_S or CC.
8502	-	Command unknown or not implemented.	Check your entries, in particular the FROM parameter. This is possibly incomplete and you have forgotten "@" or ".".
8535	-	SMTP authentication incomplete.	You have possibly entered an incorrect user name or incorrect password.
8550	-	Mail server cannot be reached, you have no access rights.	You have possibly entered an incorrect user name or password, or the mail server does not support your LOGIN. Another cause of error could be an incorrect entry of the domain name after the "@" in TO_S or CC.
8552	-	Action aborted: Assigned memory size has been exceeded	Try again later.
8554	-	Transmission failed.	Try again later.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For additional information on toggling display formats, refer to "See also".			

Note

Status error

1. An incorrect entry of the recipient addresses does not generate a status error of the "TM_MAIL" instruction. In this case, there is no guarantee that the e-mail will be sent to other recipients, even if these were entered correctly.
2. You can find more detailed information on the SMTP error code and other error codes in SMTP protocol on the Internet or in the error documentation of the mail server. You can also view the most recent message from the mail server in your instance DB in the BUFFER1 parameter. If you look under "Data", you will find the data most recently sent by the "TM_MAIL" instruction.

See also

Switching display formats in the program status (Page 2980)

9.8.2 References (S7-300, S7-400)

9.8.2.1 General parameters of the instructions

Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions

Asynchronous instructions

For instructions that work asynchronously, the function is executed with several calls.

Identification of the job

If you trigger a process interrupt, output control commands to DP slaves, start a data transfer, or abort a non-configured connection with one of the instructions listed above and then call the same instruction again before the current job is completed, the reaction of the instruction will depend on whether the second call involves the same job.
The following table gives an explanation for each instruction whose input parameters identify a job. If the parameters indicated there match a job that is not yet complete, the call is regarded as a follow-on call.

Instruction	Job is identified by ...
"DP_PRAL (Page 6515)"	IOID, LADDR
"DPSYC_FR (Page 6517)"	LADDR, GROUP, MODE
"D_ACT_DP (Page 6486)"	LADDR

Instruction	Job is identified by ...
"DPNRM_DG (Page 6523)"	LADDR
"RDSYSST (Page 6681)"	SSL_ID, INDEX
"WR_PARM (Page 6600)"	IOID, LADDR, RECNUM
"WR_DPARM (Page 6601)"	IOID, LADDR, RECNUM
"PARM_MOD (Page 6596)"	IOID, LADDR
"WR_REC (Page 6496)"	IOID, LADDR, RECNUM
"RD_REC (Page 6491)"	IOID, LADDR, RECNUM
"X_SEND (Page 7923)"	DEST_ID, REQ_ID
"X_GET (Page 7928)"	DEST_ID, VAR_ADDR
"X_PUT (Page 7930)"	DEST_ID, VAR_ADDR
"X_ABORT (Page 7931)"	DEST_ID
"I_GET (Page 7913)"	IOID, LADDR, VAR_ADDR
"I_PUT (Page 7915)"	IOID, LADDR, VAR_ADDR
"I_ABORT (Page 7916)"	IOID, LADDR
"CREA_DBL (Page 6756)"	LOW_LIMIT, UP_LIMIT, COUNT, ATTRIB, SRCBLK
"READ_DBL (Page 6747)"	SRCBLK, DSTBLK
"WRIT_DBL (Page 6749)"	SRCBLK, DSTBLK
"RD_DPARA (Page 6595)"	LADDR, RECNUM
"DP_TOPOL (Page 6526)"	DP_ID
"PN_DP (Page 7922)"	-

Parameter REQ

The input parameter REQ (request) is used solely to start the job:

- If you call the instruction for a job that is not currently active, the job is started with REQ = 1 (case 1).
- If a particular job has been started and not yet completed and you call the instruction again to perform the same job (for example, in a cyclic interrupt OB), then REQ is not evaluated by the instruction (case 2).

Parameter RET_VAL and BUSY

The output parameters RET_VAL and BUSY indicate the status of the job.

Pay attention to the note in section: Evaluating errors with output parameter RET_VAL (Page 5370)

- In case 1 (first call with REQ=1), RET_VALW#16#7001 will be entered if system resources are available and the input parameters are correct. BUSY will be set.
  If the required system resources are currently being used or the input parameters have errors, the corresponding error code is entered in RET_VAL and BUSY has the value 0.
- In case 2 (interim call) W#16#7002 will be entered in RET_VAL (this corresponds to a warning: Job still being processed!), and BUSY will be set.
- The following applies to the last call for a job:
  - For instructions "DPNRM_DG (Page 6523)", "X_GET (Page 7928)" and "I_GET (Page 7913)" in case of fault-free data transmission, the number of supplied data will be entered in RET_VAL as a positive number in bytes. BUSY has the value "0" in this case.
    If there is an error, then the error information will be entered in RET_VAL and you should not evaluate BUSY in this case.
  - With "RD_REC" the size of the data record in bytes or the value 0 is entered in RET_VAL if no error has occurred during data transmission (see also: RD_REC: Read data record from I/O (Page 6491)). In this case BUSY is written with "0". If there is an error, the error code is entered in RET_VAL and BUSY has the value "0" in this case.
  - For all other instructions, "0" will be entered in RET_VAL if the job was executed without errors and BUSY has the value "0" in this case. If there is an error, the error code is entered in RET_VAL and BUSY has the value "0" in this case.

Note

If the first and last call coincide, the reaction is the same for RET_VAL and BUSY as described for the last call.

Overview

The following table provides you with an overview of the relationships explained above. In particular, it shows the possible values of the output parameters if the execution of the job is not completed after an instruction call has been completed.

Note

Following every call, you must evaluate the relevant output parameters in your program.

Relationship between call, REQ, RET_VAL and BUSY during execution of a "running" job.

Number of the call	Type of call	REQ	RET_VAL	BUSY
1	First call	1	W#16#7001	1
			Error code	0
2 to (n - 1)	Intermediate call	irrelevant	W#16#7002	1

Number of the call	Type of call	REQ	RET_VAL	BUSY
n	Last call	irrelevant	W#16#0000 (exceptions: "RD_REC (Page 6491)" if the destination area is larger than the data record transferred; "DPNRM_DG (Page 6523)", "X_GET (Page 7928)" and "I_GET (Page 7913)"), if no errors have occurred.	0
			Error code if errors occurred	0

Evaluating errors with output parameter RET_VAL

Types of error information

An executed instruction indicates in the user program whether or not the CPU was able to execute the function of the instruction successfully.

You can obtain information about any errors that occurred in two ways:

- In the BR bit of the status word
- in the output parameter RET_VAL (return value).

Note

Before evaluating the output parameters specific to an instruction, you should always follow the steps below:

- First, evaluate the BR bit of the status word.
- Then check the output parameter RET_VAL.

If the BR bit indicates that an error has occurred or if RET_VAL contains a general error code, you should not evaluate the instruction-specific output parameters.

Error information in the return value

An instruction indicates that an error occurred during its execution by entering the value "0" in the binary result bit (BR) of the status word. Some instructions provide an additional error code at an output parameter known as the return value (RET_VAL). If a general error is entered in the output parameter RET_VAL (see below for explanation), this is only indicated by the value "0" in the BR bit of the status word.

The return value is of the data type integer (INT). The relationship of the return value to the value "0" indicates whether or not an error occurred during execution of the function.

CPU execution of the instruction	BR	Return value	Sign of the integer
With error(s)	0	less than "0"	negative (sign bit is "1")
Without error	1	greater than or equal to "0"	positive (sign bit is "0")

Reacting to error information

There are two types of error codes in RET_VAL:

- A general error code that all instructions can output and
- A specific error code that an instruction can output and which relates to its specific function.

You can write your program so that it reacts to the errors that occur during execution of an instruction. This way you prevent further errors occurring as a result of the first error.

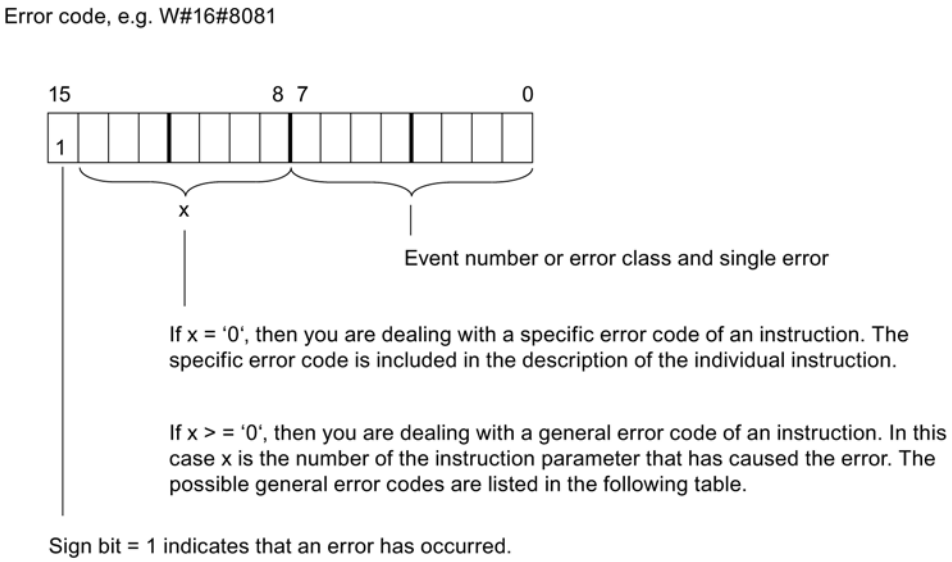
General and specific error information

The return value (RET_VAL) of an instruction provides one of the two following types of error codes:

- A general error code that relates to errors that can occur in any instruction.
- A specific error code that relates only to the particular instruction.

Even though the data type of the output parameter RET_VAL is an integer (INT), the error codes for the instruction are grouped according to hexadecimal values. If you want to examine a return value and compare the value with the error codes listed in this documentation, then display the error code in hexadecimal format.

The figure below shows the structure of a system function error code in hexadecimal format.

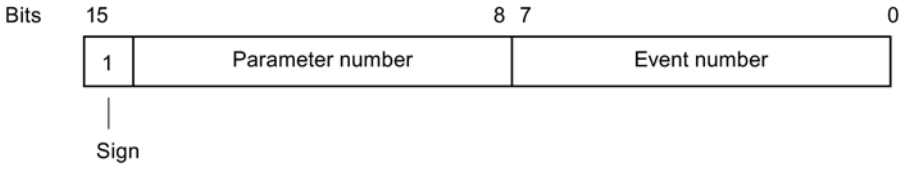


General error information

The general error code indicates errors that can occur in all instructions. A general error code consists of the following two numbers:

- A parameter number from 1 to 111, where 1 indicates the first parameter of the called instruction, 2 the second parameter, and so forth.
- An event number from 0 to 127. The event number indicates that a synchronous error occurred.

The following table lists the codes for general errors and an explanation of each error.



Note

If a general error code was entered in RET_VAL, then the following situations are possible:

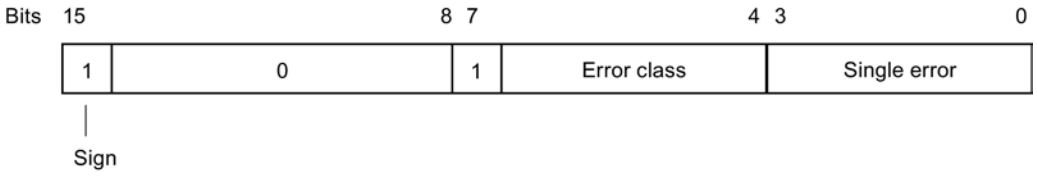
- The action associated with the instruction may have been started or already completed.
- A specific instruction error may have occurred when the action was performed. As a result of a general error that occurred later, the specific error could, however, no longer be indicated.

Specific error information

Some instructions have a return value that provides an error code specific for the instruction. A specific error code indicates errors that can occur only in specific instructions.

A specific error code consists of the following two numbers:

- An error class from 0 to 7.
- An error number from 0 to 15.



General error codes

The following table explains the general error codes of a return value. The error code is shown in hexadecimal format. The letter x in each code number is simply a place holder and represents the number of the system function parameter that caused the error.

General error codes

Error code (W#16#...)	Explanation
8x7F	Internal error This error code indicates an internal error at parameter x.
8x01	Illegal syntax ID at an ANY parameter
8x22	Range length error when reading a parameter.

Error code (W#16#...)	Explanation
8x23	Range length error when writing a parameter. This error code indicates that the parameter x is located either entirely or partly outside the range of an address, or that the length of a bit range is not a multiple of 8 with an ANY parameter.
8x24	Range error when reading a parameter.
8x25	Range error when writing a parameter. This error code indicates that the parameter x is located in a range that is illegal for the system function. Refer to the descriptions of the individual functions for information about the illegal ranges.
8x26	The parameter contains a timer cell number that is too high. This error code indicates that the timer cell specified in parameter x does not exist.
8x27	The parameter contains a counter cell number that is too high (counter number error). This error code indicates that the counter cell specified in parameter x does not exist.
8x28	Alignment error when reading a parameter.
8x29	Alignment error when writing a parameter. This error code indicates that the reference to parameter x is an operand with bit address that is not equal to 0.
8x30	The parameter is located in a read-only global DB.
8x31	The parameter is located in a read-only instance DB. This error code indicates that parameter x is located in a read-only data block. If the data block was opened by the system function itself, the system function always returns the value W#16#8x30.
8x32	The parameter contains a DB number that is too high (DB number error).
8x34	The parameter contains an FC number that is too high (FC number error).
8x35	The parameter contains an FB number that is too high (FB number error). This error code indicates that parameter x contains a block number higher than the highest permitted number.
8x3A	The parameter contains the number of a DB that is not loaded.
8x3C	The parameter contains the number of an FC that is not loaded.
8x3E	The parameter contains the number of an FB that is not loaded.
8x42	An access error occurred while the system was attempting to read a parameter from the peripheral input area.
8x43	An access error occurred while the system was attempting to write a parameter to the peripheral output area.
8x44	Error in the nth (n > 1) read access after an error occurred.
8x45	Error in the nth (n > 1) write access after an error occurred. This error code indicates that access to the required parameter is denied.

9.8.2.2 Basic instructions

LAD

Bit logic operations

---| |---: Normally open contact

Description

The activation of the normally open contact depends on the signal state of the associated operand. When the operand has signal state "1", the normally open contact closes and the signal state at the output is set to the signl state of the input.

When the operand has signal state "0", the normally open contact is not activated and the signal state at the output of the instruction is reset to "0".

Two or more normally open contacts are linked bit-by-bit by AND when connected in series. With a series connection, power flows when all contacts are closed.

The normally open contacts are linked by OR when connected in parallel. With a parallel connection, power flows when one of the contacts is closed.

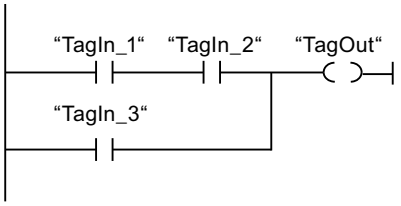
Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, D, L, T, C	Operand whose signal state is queried.

Example

The following example shows how the instruction works:



The "TagOut" operand is set when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The operand "TagIn_3" has the signal state "1".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Example of controlling a conveyor belt (Page 3115)

Example of detecting the fill level of a storage area (Page 3118)

Example of controlling room temperature (Page 3122)

Setting the status bit for bit logic operations (Page 2455)

Basic information on the status word (Page 2453)

Example of detecting the direction of a conveyor belt (Page 3117)

---| / |---: Normally closed contact

Description

The activation of the normally closed contact depends on the signal state of the associated operand. When the operand has signal state "1", the normally closed contact opens and the signal state at the output of the instruction is reset to "0".

When the operand has signal state "0", the normally closed contact is not enabled and the signal state of the input is transferred to the output.

Two or more normally closed contacts are linked bit-by-bit by AND when connected in series. With a series connection, power flows when all contacts are closed.

The normally closed contacts are linked by OR when connected in parallel. With a parallel connection, power flows when one of the contacts is closed.

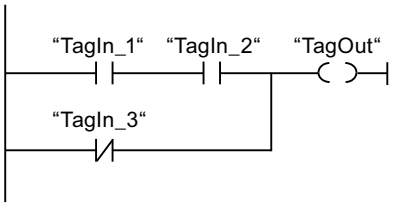
Parameter

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, D, L, T, C	Operand whose signal state is queried.

Example

The following example shows how the instruction works:



The "TagOut" operand is set when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The operand "TagIn_3" has the signal state "0".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Example of detecting the direction of a conveyor belt (Page 3117)

Example of controlling room temperature (Page 3122)

Setting the status bit for bit logic operations (Page 2455)

Basic information on the status word (Page 2453)

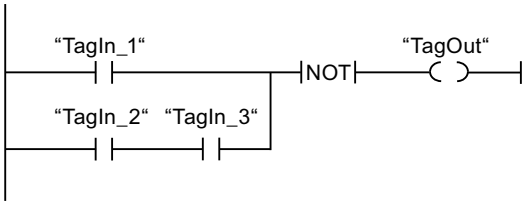
--|NOT|--: Invert RLO

Description

You use the "Invert RLO" instruction to invert the signal state of the result of logic operation (RLO). If the signal state is "1" at the input of the instruction, the output of the instruction has signal state "0". If the signal state is "0" at the input of the instruction, the output has the signal state "1".

Example

The following example shows how the instruction works:



Operand "TagOut" is reset when one of the following conditions is fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The signal state of the operands "TagIn_2" and "TagIn_3" is "1".

See also

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Setting the status bit for bit logic operations (Page 2455)

Basic information on the status word (Page 2453)

---( )---: Assignment

Description

You can use the "Assignment" instruction to set the bit of a specified operand. If the result of logic operation (RLO) at the input of the coil has signal state "1", the specified operand is set to signal state "1". If the signal state is "0" at the input of the coil, the bit of the specified operand is reset to "0".

The instruction does not influence the RLO. The RLO at the input of the coil is sent directly to the output.

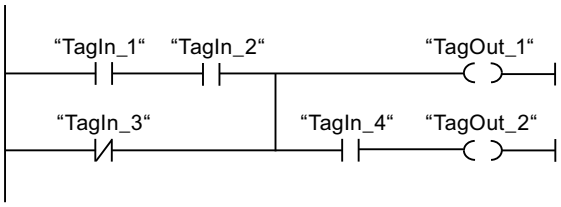
Parameter

The following table shows the parameters of the "Assignment" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand to which the RLO is assigned.

Example

The following example shows how the instruction works:



The "TagOut_1" operand is set when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The signal state of the operand "TagIn_3" is "0".

The "TagOut_2" operand is set when one of the following conditions is fulfilled:

- Operands "TagIn_1", "TagIn_2", and "TagIn_4" have signal state "1".
- The signal state of the "TagIn_3" operand is "0" and the signal state of the "TagIn_4" operand is "1".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Example of detecting the fill level of a storage area (Page 3118)

Example of controlling room temperature (Page 3122)

Master control relay (Page 2471)

Important notes on using MCR functions (Page 2472)

Setting the status bit for bit logic operations (Page 2455)

Basic information on the status word (Page 2453)

---(MCR<): Open MCR ranges (Page 5529)

---(R)---: Reset output

Description

You can use the "Reset output" instruction to reset the signal state of a specified operand to "0".

The instruction is only executed if the result of logic operation (RLO) at the input of the coil is "1". If power flows to the coil (RLO = "1"), the specified operand is reset to "0". If the RLO at the input of the coil is "0" (no signal flow to the coil), the signal state of the specified operand remains unchanged.

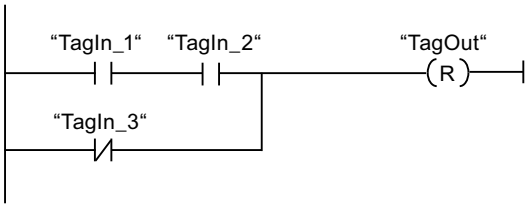
Parameter

The following table shows the parameters of the "Reset output" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L, T, C	Operand that is reset when RLO = "1".

Example

The following example shows how the instruction works:



Operand "TagOut" is reset when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The signal state of the operand "TagIn_3" is "0".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Example of controlling a conveyor belt (Page 3115)

Example of detecting the direction of a conveyor belt (Page 3117)

Master control relay (Page 2471)

Important notes on using MCR functions (Page 2472)

Setting the status bit for bit logic operations (Page 2455)

Basic information on the status word (Page 2453)

---(MCR<): Open MCR ranges (Page 5529)

---(S)---: Set output

Description

You can use the "Set output" instruction to set the signal state of a specified operand to "1".

The instruction is only executed if the result of logic operation (RLO) at the input of the coil is "1". If power flows to the coil (RLO = "1"), the specified operand is set to "1". If the RLO at the input of the coil is "0" (no signal flow to the coil), the signal state of the specified operand remains unchanged.

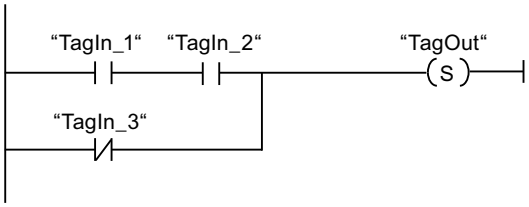
Parameter

The following table shows the parameters of the "Set output" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand which is set with RLO = "1".

Example

The following example shows how the instruction works:



The "TagOut" operand is set when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The signal state of the operand "TagIn_3" is "0".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Example of controlling a conveyor belt (Page 3115)

Example of detecting the direction of a conveyor belt (Page 3117)

Master control relay (Page 2471)

Important notes on using MCR functions (Page 2472)

Setting the status bit for bit logic operations (Page 2455)

Basic information on the status word (Page 2453)

---(MCR<): Open MCR ranges (Page 5529)

SR: Set/reset flip-flop

Description

Use the instruction "Set/reset flip-flop" to set or reset the bit of the specified operand, depending on the signal state of the inputs S and R1. If the signal state at input S is "1" and the input R1

is "0", the specified operand is set to "1". If the signal state is "0" at input S and "1" at input R1, the specified operand will be reset to "0".

Input R1 takes priority over input S. When the signal state is "1" on both inputs S and R1, the signal state of the specified operand is reset to "0".

The instruction is not executed if the signal state at the two inputs S and R1 is "0". The signal state of the operand then remains unchanged.

The current signal state of the operand is transferred to output Q and can be queried there.

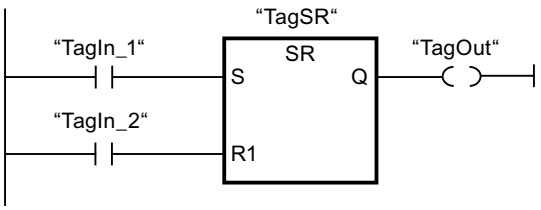
Parameter

The following table shows the parameters of the "Set/reset flip-flop" instruction:

Parameter	Declaration	Data type	Memory area	Description
S	Input	BOOL	I, Q, M, D, L	Enable setting
R1	Input	BOOL	I, Q, M, D, L, T, C	Enable resetting
<Operand>	InOut	BOOL	I, Q, M, D, L	Operand that is set or reset.
Q	Output	BOOL	I, Q, M, D, L	Signal state of the operand

Example

The following example shows how the instruction works:



The operands "TagSR" and "TagOut" are set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The operand "TagIn_2" has the signal state "0".

The operands "TagSR" and "TagOut" are reset when one of the following conditions is fulfilled:

- The operand "TagIn_1" has signal state "0" and the operand "TagIn_2" has signal state "1".
- The operands "TagIn_1" and "TagIn_2" have signal state "1".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Master control relay (Page 2471)

Important notes on using MCR functions (Page 2472)

Setting the status bit for bit logic operations (Page 2455)

Basic information on the status word (Page 2453)

---(MCR<): Open MCR ranges (Page 5529)

RS: Reset/set flip-flop

Description

You can use the "Reset/set flip-flop" instruction to reset or set the bit of a specified operand based on the signal state of the inputs R and S1. If the signal state is "1" at input R and "0" at input S1, the specified operand will be reset to "0". If the signal state is "0" at input R and "1" at input S1, the specified operand is set to "1".

Input S1 takes priority over input R. When the signal state is "1" at both inputs R and S1, the signal state of the specified operand is set to "1".

The instruction is not executed if the signal state at the two inputs R and S1 is "0". The signal state of the operand then remains unchanged.

The current signal state of the operand is transferred to output Q and can be queried there.

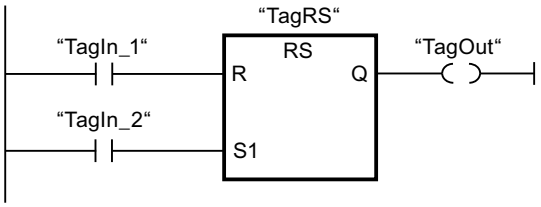
Parameter

The following table shows the parameters of the "Reset/set flip-flop" instruction:

Parameter	Declaration	Data type	Memory area	Description
R	Input	BOOL	I, Q, M, D, L	Enable resetting
S1	Input	BOOL	I, Q, M, D, L, T, C	Enable setting
<Operand>	InOut	BOOL	I, Q, M, D, L	Operand that is reset or set.
Q	Output	BOOL	I, Q, M, D, L	Signal state of the operand

Example

The following example shows how the instruction works:



The operands "TagRS" and "TagOut" are reset when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The operand "TagIn_2" has the signal state "0".

The operands "TagRS" and "TagOut" are set when one of the following conditions is fulfilled:

- The operand "TagIn_1" has signal state "0" and the operand "TagIn_2" has signal state "1".
- The operands "TagIn_1" and "TagIn_2" have signal state "1".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Master control relay (Page 2471)

Important notes on using MCR functions (Page 2472)

Setting the status bit for bit logic operations (Page 2455)

Basic information on the status word (Page 2453)

---(MCR<): Open MCR ranges (Page 5529)

--|P|--: Scan operand for positive signal edge

Description

You can use the "Scan operand for positive signal edge" instruction to determine if there is a "0" to "1" change in the signal state of a specified operand (<Operand1>). The instruction compares the current signal state of <Operand1> with the signal state of the previous scan that is saved in an edge memory bit (<Operand2>). If the instruction detects a change in the result of logic operation (RLO) from "0" to "1", there is a positive, rising edge.

If a positive edge is detected, the output of the instruction has the signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Specify the operand to be queried (<Operand1>) in the operand placeholder above the instruction. Specify the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This step influences the edge evaluation and the result is therefore no longer unique. The memory area of the edge memory bit must be located in a DB (static area for FB) or in the bit memory area.

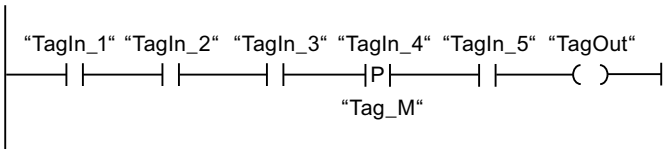
Parameter

The following table shows the parameters of the "Scan operand for positive signal edge" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	BOOL	I, Q, M, D, L, T, C	Signal to be scanned
<Operand2>	InOut	BOOL	I, Q, M, D, L	Edge memory bit in which the signal state of the previous query is saved.

Example

The following example shows how the instruction works:



Operand "TagOut" is set when the following conditions are fulfilled:

- The operands "TagIn_1", "TagIn_2", and "TagIn_3" have signal state "1".
- There is a rising edge at operand "TagIn_4". The signal state of the previous scan is stored in the edge memory bit "Tag_M".
- The signal state of the operand "TagIn_5" is "1".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Example of detecting the direction of a conveyor belt (Page 3117)

Setting the status bit for bit logic operations (Page 2455)

Basic information on the status word (Page 2453)

--|N|--: Scan operand for negative signal edge

Description

You can use the "Scan operand for negative signal edge" instruction to determine if there is a "1" to "0" change in the signal state of a specified operand (<Operand1>). The instruction compares the current signal state of <Operand1> with the signal state of the previous scan

that is saved in an edge memory bit <Operand2>. If the instruction detects a change in the result of logic operation (RLO) from "1" to "0", there is a negative, falling edge.

If a negative signal edge is detected, the output of the instruction has the signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Specify the operand to be queried (<Operand1>) in the operand placeholder above the instruction. Specify the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This step influences the edge evaluation and the result is therefore no longer unique. The memory area of the edge memory bit must be located in a DB (static area for FB) or in the bit memory area.

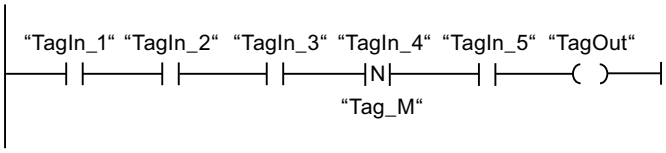
Parameter

The following table shows the parameters of the "Scan operand for negative signal edge" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	BOOL	I, Q, M, D, L, T, C	Signal to be scanned
<Operand2>	InOut	BOOL	I, Q, M, D, L	Edge memory bit in which the signal state of the previous query is saved.

Example

The following example shows how the instruction works:



Operand "TagOut" is set when the following conditions are fulfilled:

- The operands "TagIn_1", "TagIn_2", and "TagIn_3" have signal state "1".
- There is a negative signal edge at operand "TagIn_4". The signal state of the previous scan is stored in the edge memory bit "Tag_M".
- The signal state of the operand "TagIn_5" is "1".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Setting the status bit for bit logic operations (Page 2455)

Basic information on the status word (Page 2453)

P_TRIG: Scan RLO for positive signal edge

Description

Use the "Scan RLO for positive signal edge" instruction to query a "0" to "1" change in the signal state of the result of logic operation (RLO). The instruction compares the current signal state of the RLO with the signal state of the previous query, which is saved in an edge memory bit (<operand>). If the instruction detects a change in the RLO from "0" to "1", there is a positive signal edge.

If a positive edge is detected, the output of the instruction has the signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This step influences the edge evaluation and the result is therefore no longer unique. The memory area of the edge memory bit must be located in a DB (static area for FB) or in the bit memory area.

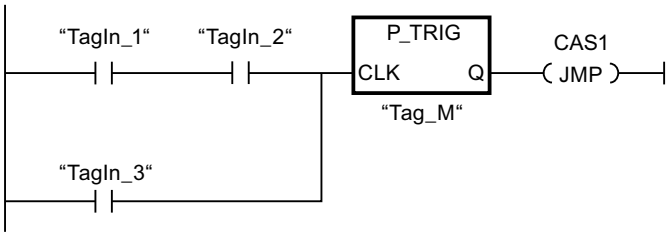
Parameter

The following table shows the parameters of the "Scan RLO for positive signal edge" instruction:

Parameter	Declaration	Data type	Memory area	Description
CLK	Input	BOOL	I, Q, M, D, L	Current RLO
<Operand>	InOut	BOOL	M, D	Edge memory bit in which the RLO of the previous query is saved.
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:



The RLO of the previous query is saved in the edge memory bit "Tag_M". If a "0" to "1" change is detected in the signal state of the RLO, the program jumps to jump label CAS1.

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Setting the status bit for bit logic operations (Page 2455)

Basic information on the status word (Page 2453)

N_TRIG: Scan RLO for negative signal edge

Description

Use the "Scan RLO for negative signal edge" instruction to query a "1" to "0" change in the signal state of the result of logic operation (RLO). The instruction compares the current signal state of the RLO with the signal state of the previous query, which is saved in an edge memory bit (<operand>). If the instruction detects a change in the RLO from "1" to "0", there is a negative edge.

If a negative signal edge is detected, the output of the instruction has the signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the bit memory is overwritten. This step influences the edge evaluation and the result is therefore no longer unique. The memory area of the edge memory bit must be located in a DB (static area for FB) or in the bit memory area.

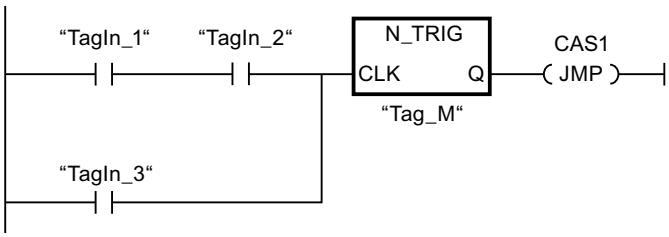
Parameter

The following table shows the parameters of the "Scan RLO for negative signal edge" instruction:

Parameters	Declaration	Data type	Memory area	Description
CLK	Input	BOOL	I, Q, M, D, L	Current RLO
<Operand>	InOut	BOOL	M, D	Edge memory bit in which the RLO of the previous query is saved.
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:



The RLO of the previous query is saved in the edge memory bit "Tag_M". If a "1" to "0" change is detected in the signal state of the RLO, the program jumps to jump label CAS1.

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Setting the status bit for bit logic operations (Page 2455)

Basic information on the status word (Page 2453)

Timer operations

IEC Timers

TP: Generate pulse

Description

You can use the "Generate pulse" instruction to set the output Q for a programmed duration. The instruction is started when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. Output Q is set for the duration PT, regardless of the subsequent course of the input signal. Even if a new positive signal edge is detected, the signal state at the output Q is not affected as long as the PT time duration is running.

You can query the current time value at the ET output. The timer value starts at T#0s and ends when the value of duration PT is reached. When the duration PT is reached and the signal state at input IN is "0", the ET output is reset.

Note

If the timer is not called in the program because it is skipped, for example, the ET output returns a constant value as soon as the timer has expired.

Each call of the "Generate pulse" instruction must be assigned an IEC timer in which the instruction data is stored. An IEC timer can be declared as follows:

- Declaration of a data block of the type TP (for example, "TP_DB")
- Declaration as a local tag of the type TP in the "Static" section of a block (for example, #MyTP_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens automatically, in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when the instruction is called and also each time the outputs Q or ET are accessed.

The operating system resets the instances of the "Generate pulse" instruction during a cold restart. If you want instances of the instruction to be in the initialized state after a warm restart, call these in a startup OB with the value "0" set for the PT parameter. If instances of the instruction "Generate pulse" are located within another block, you can reset these instances, for example, by initializing the higher-level block.

The execution of the "Generate pulse" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameter

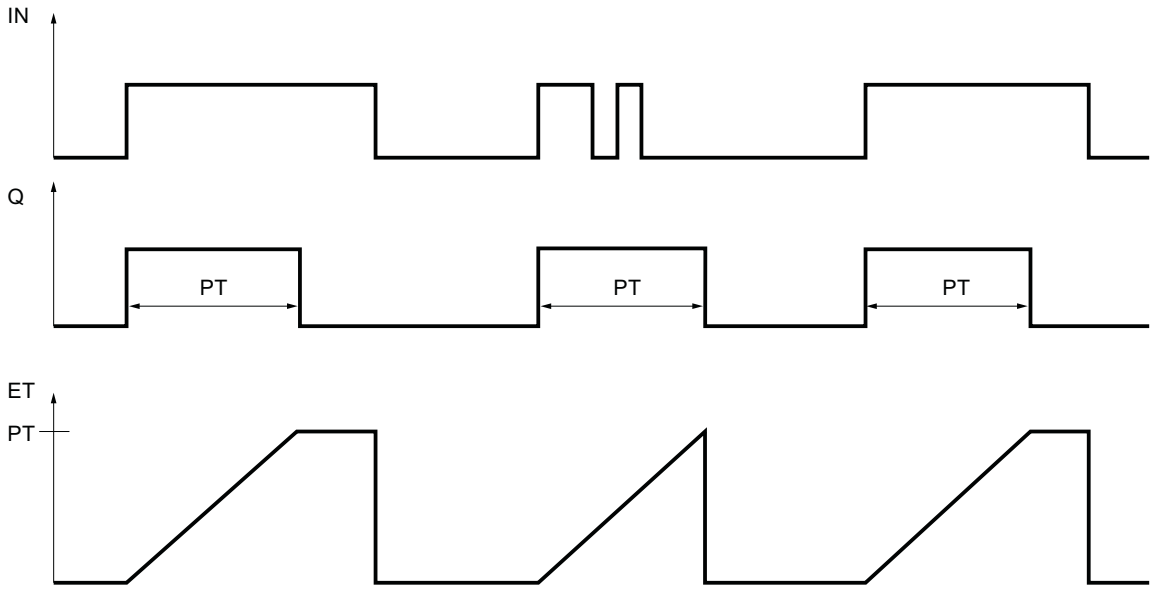
The following table shows the parameters of the "Generate pulse" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L	Start input
PT	Input	TIME	I, Q, M, D, L, P or constant	Duration of the pulse. The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L	Pulse output
ET	Output	TIME	I, Q, M, D, L	Current timer value

For additional information on valid data types, refer to "See also".

Pulse diagram

The following figure shows the pulse diagram of the "Generate pulse" instruction:



See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Example of controlling room temperature (Page 3122)

Call function blocks as single or multi-instances (Page 2182)

TON: Generate on-delay

Description

You can use the "Generate on-delay" instruction to delay setting of the Q output by the programmed duration PT. The instruction is started when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. When the duration PT expires, the output Q has the signal state "1". Output Q remains set as long as the start input is still "1". When the signal state at the start input changes from "1" to "0", the Q output is reset. The timer function is started again when a new positive signal edge is detected at the start input.

You can query the current time value at the ET output. The timer value starts at T#0s and ends when the value of duration PT is reached. The ET output is reset as soon as the signal state at the IN input changes to "0".

Note

If the timer is not called in the program because it is skipped, for example, the ET output returns a constant value as soon as the timer has expired.

Each call of the "Generate on-delay" instruction must be assigned an IEC timer in which the instruction data is stored. An IEC timer can be declared as follows:

- Declaration of a data block of the type TON (for example, "TON_DB")
- Declaration as a local tag of the type TON in the "Static" section of a block (for example, #MyTON_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens automatically, in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when the instruction is called and also each time the outputs Q or ET are accessed.

The operating system resets the instances of the "Generate on-delay" instruction during a cold restart. If you want instances of the instruction to be in the initialized state after a warm restart, call these in a startup OB with the value "0" set for the PT parameter. If instances of the "Generate on-delay" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

The execution of the "Generate on-delay" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameter

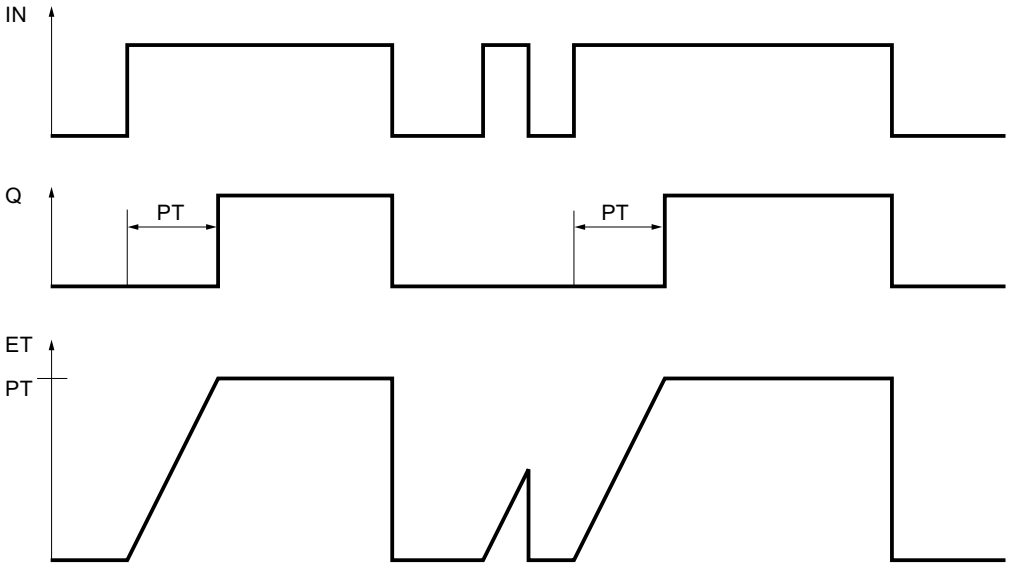
The following table shows the parameters of the "Generate on-delay" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L	Start input
PT	Input	TIME	I, Q, M, D, L, P or constant	Duration of the on-delay The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L	Output that is set when the PT time expires.
ET	Output	TIME	I, Q, M, D, L	Current timer value

For additional information on valid data types, refer to "See also".

Pulse diagram

The following figure shows the pulse diagram of the "Generate on-delay" instruction:



See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Call function blocks as single or multi-instances (Page 2182)

TOF: Generate off-delay

Description

You can use the "Generate off-delay" instruction to delay resetting of the Q output by the programmed duration PT. The Q output is set when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive signal edge). When the signal state at input IN changes back to "0", the programmed time PT starts. Output Q remains set as long as the duration PT is running. When duration PT expires, the Q output is reset. If the signal state at input IN changes to "1" before the PT time duration expires, the timer is reset. The signal state at the output Q continues to be "1".

You can query the current time value at the ET output. The current timer value starts at T#0s and ends when the value of the time duration PT is reached. When the time duration PT expires, the ET output remains set to the current value until the IN input changes back to "1". If input IN switches to "1" before the duration PT has expired, the ET output is reset to the value T#0s.

Note

If the timer is not called in the program because it is skipped, for example, the ET output returns a constant value as soon as the timer has expired.

Each call of the "Generate off-delay" instruction must be assigned an IEC timer in which the instruction data is stored. An IEC timer can be declared as follows:

- Declaration of a data block of the type TOF (for example, "TOF_DB")
- Declaration as a local tag of the type TOF in the "Static" section of a block (for example, #MyTOF_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens automatically, in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when the instruction is called and also each time the outputs Q or ET are accessed.

The operating system resets the instances of the "Generate off-delay" instruction during a cold restart. If you want instances of the instruction to be in the initialized state after a warm restart, call these in a startup OB with the value "0" set for the PT parameter. If instances of the "Generate off-delay" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

The execution of the "Generate off-delay" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameter

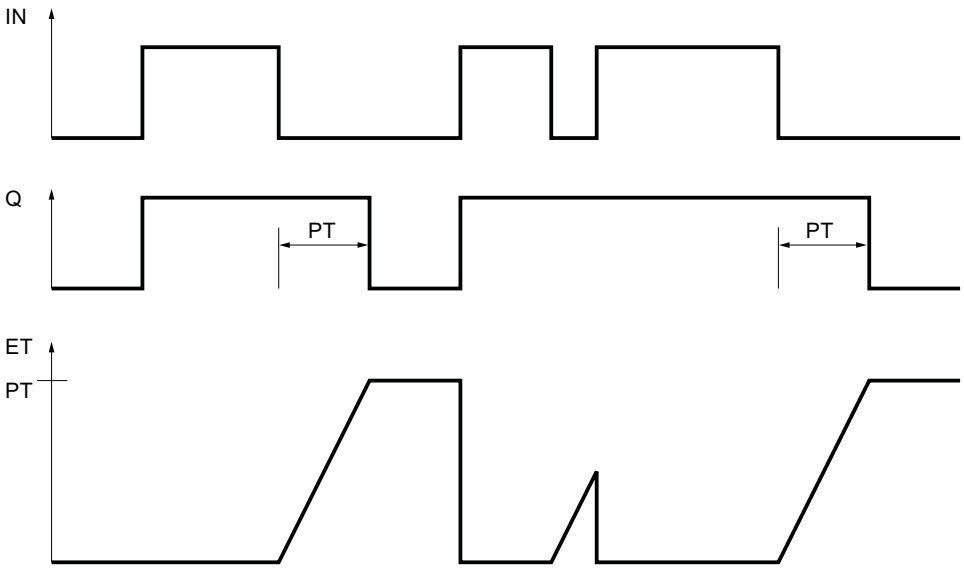
The following table shows the parameters of the "Generate off-delay" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L	Start input
PT	Input	TIME	I, Q, M, D, L, P or constant	Duration of the off delay The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L	Output that is reset when the time PT expires.
ET	Output	TIME	I, Q, M, D, L	Current timer value

For additional information on valid data types, refer to "See also".

Pulse diagram

The following figure shows the pulse diagram of the "Generate off-delay" instruction:



See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Call function blocks as single or multi-instances (Page 2182)

SIMATIC Timers

S_PULSE: Assign pulse timer parameters and start

Description

The "Assign pulse timer parameters and start" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV) as soon as the signal state at input S is "1". If the signal state at input S changes to "0" before the programmed duration expires, the timer is stopped. In this case, the signal state at output Q is "0".

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed. The current time value is output BI-coded at output BI and BCD-coded at output BCD.

If the timer is running and the signal state at input R changes to "1" then the current time value and the time base are also set to zero. If the timer is not running, the signal state "1" at the R input has no effect.

The "Assign pulse timer parameters and start" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

The instruction data is updated at every access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameter

The following table shows the parameters of the instruction "Assign pulse timer parameters and start":

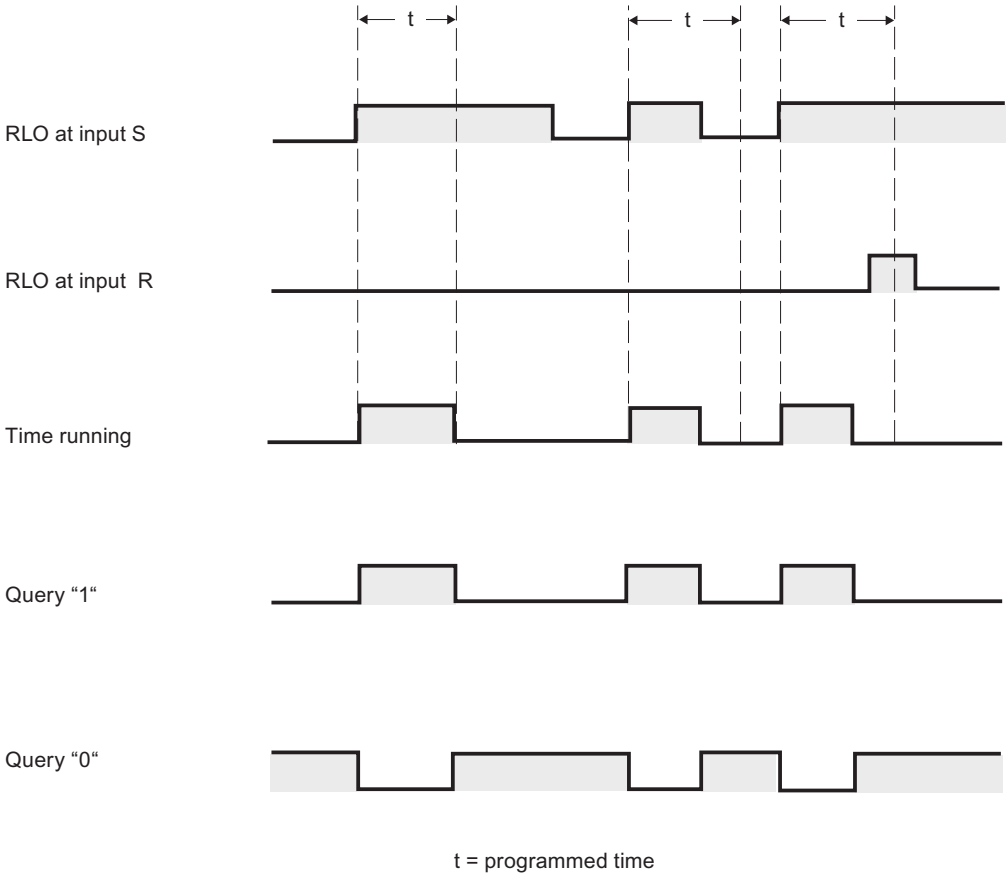
Parameter	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Preset time value

Parameter	Declaration	Data type	Memory area	Description
R	Input	BOOL	I, Q, M, D, L, T, C, P	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (BI-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the timer

For additional information on valid data types, refer to "See also".

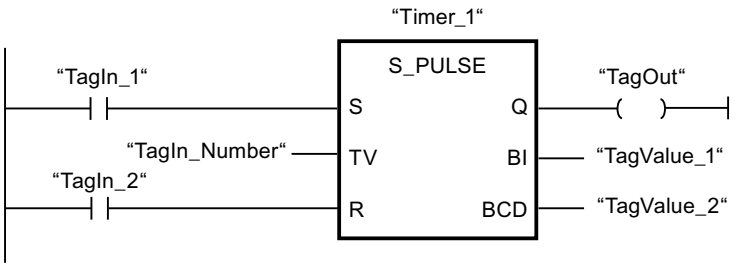
Pulse timing diagram

The following figure shows the pulse timing diagram of the instruction "Assign pulse timer parameters and start":



Example

The following example shows how the instruction works:



The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "0" to "1". The timer expires with the time value of the operand "TagIn_Number" as long as the operand "TagIn_1" has the signal state "1". If the signal state of the operand "TagIn_1" changes from "1" to "0" before the timer expires, the timer "Timer_1" is stopped. The operand "TagOut" is reset to "0" in this case.

The operand "TagOut" has the signal state "1" as long as the timer is running and the operand "TagIn_1" has the signal state "1". When the timer expires or is reset, the operand "TagOut" is reset to "0".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

L: Load timer value (Page 5934)

S_PEXT: Assign extended pulse timer parameters and start

Description

The "Assign extended pulse timer parameters and start" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV), even if the signal state at input S changes to "0". As long as the timer is running, the output Q has the signal state "1". When the timer expires, the output Q is reset to "0". If the signal state at input S changes from "0" to "1" while the timer is running, the timer is restarted with the duration programmed at input TV.

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed. The current time value is output BI-coded at output BI and BCD-coded at output BCD.

If the timer is running and the signal state at input R changes to "1" then the current time value and the time base are also set to zero. If the timer is not running, the signal state "1" at the R input has no effect.

The "Assign extended pulse timer parameters and start" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

The instruction data is updated at every access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameter

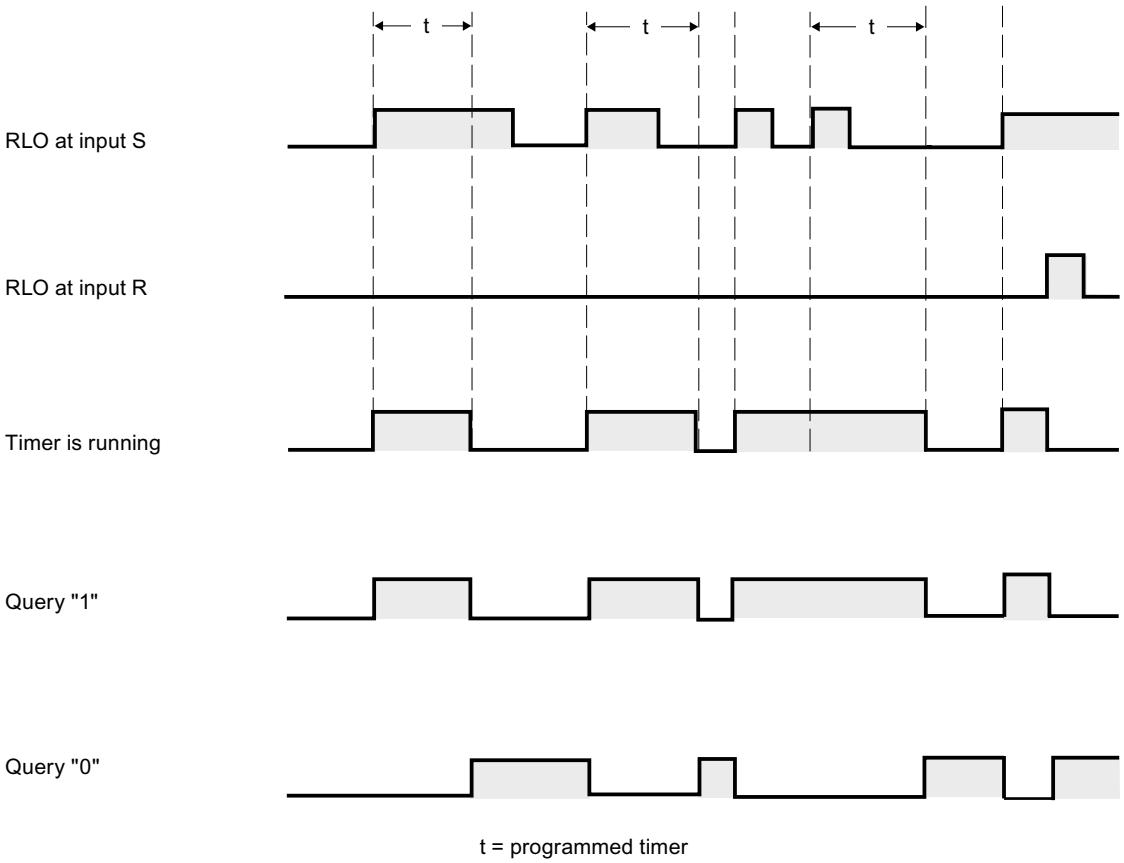
The following table shows the parameters of the instruction "Assign extended pulse timer parameters and start":

Parameter	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Preset time value
R	Input	BOOL	I, Q, M, D, L, T, C, P	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (BI-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the timer

For additional information on valid data types, refer to "See also".

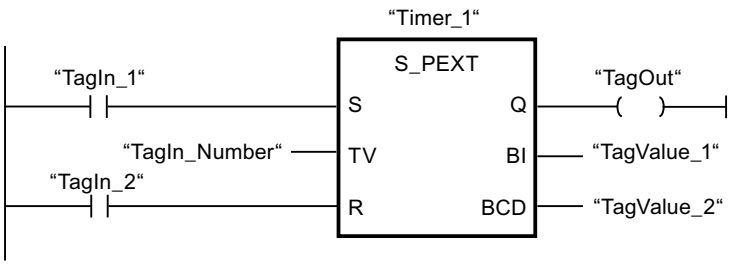
Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign extended pulse timer parameters and start" instruction:



Example

The following example shows how the instruction works:



The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "0" to "1". The timer expires with the time value of the operand "TagIn_Number" without being affected by a negative edge at input S. If the signal state of the operand "TagIn_1" changes from "0" to "1" before the timer expires, the timer is restarted.

The operand "TagOut" has the signal state "1" as long as the timer is running. When the timer expires or is reset, the operand "TagOut" is reset to "0".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

L: Load timer value (Page 5934)

S_ODT: Assign on-delay timer parameters and start

Description

The "Assign on-delay timer parameters and start" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV) as soon as the signal state at input S is "1". If the timer expires correctly and input S still has signal state "1" then output Q returns signal state "1". If the signal state at input S changes from "1" to "0" while the timer is running, the timer is stopped. In this case, output Q is reset to signal state "0".

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed. The current time value is output BI-coded at output BI and BCD-coded at output BCD.

If the time is running and the signal state at input R changes from "0" to "1" then the current time value and the time base are also set to zero. In this case, the signal state at output Q is "0". The timer is reset if the signal state is "1" at the R input even if the timer is not running and the RLO at input S is "1".

Specify the timer of the instruction in the placeholder above the box. The timer must be declared with the data type TIMER.

The "Assign on-delay timer parameters and start" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

The instruction data is updated at every access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

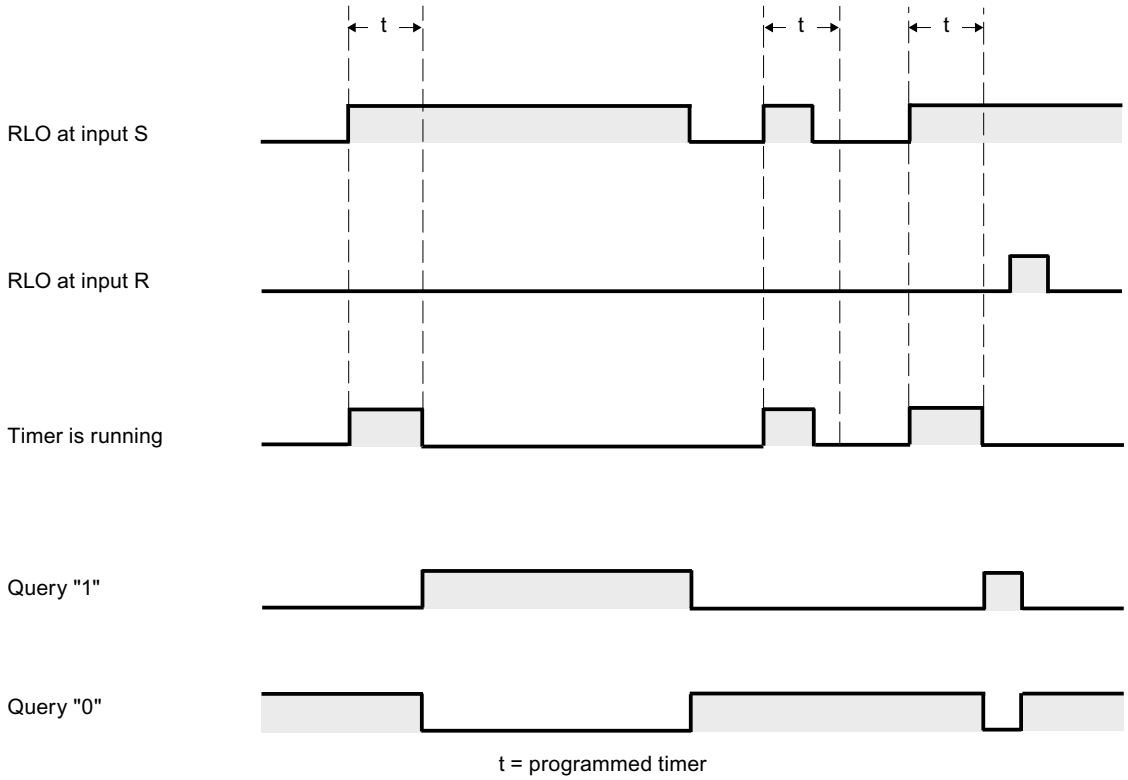
The following table shows the parameters of the "Assign on-delay timer parameters and start" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Preset time value
R	Input	BOOL	I, Q, M, D, L, T, C, P	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (BI-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the timer

For additional information on valid data types, refer to "See also".

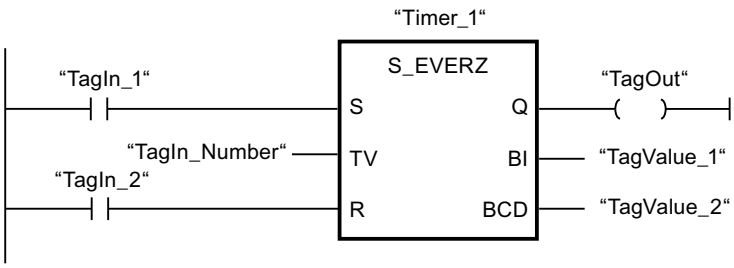
Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign on-delay timer parameters and start" instruction:



Example

The following example shows how the instruction works:



The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "0" to "1". The timer expires with the time value of the operand "TagIn_Number". If the timer expires and the operand has the signal state "1", the operand "TagOut" is set to "1". If the signal state of the operand "TagIn_1" changes from "1" to "0" before the timer expires, the timer is stopped. The operand "TagOut" has the signal state "0" in this case.

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

L: Load timer value (Page 5934)

S_ODTS: Assign retentive on-delay timer parameters and start

Description

The "Assign retentive on-delay timer parameters and start" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV), even if the signal state at input S changes to "0". If the timer expires, the "Q" output returns signal state "1" regardless of the signal state at input "S". If the signal state at input S changes from "0" to "1" while the timer is running, the timer is restarted with the duration programmed at input (TV).

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed. The current time value is output BI-coded at output BI and BCD-coded at output BCD.

Signal state "1" at input R resets the current time value and time base to "0" regardless of the signal state at start input S. In this case, the signal state at output Q is "0".

The "Assign retentive on-delay timer parameters and start" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

The instruction data is updated at every access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

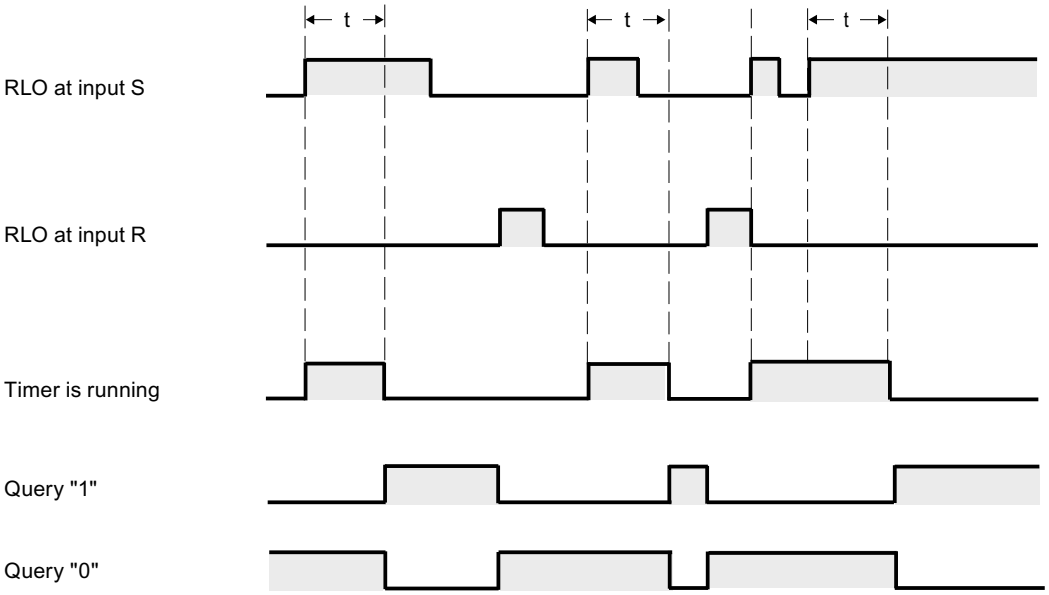
The following table shows the parameters of the "Assign retentive on-delay timer parameters and start" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Preset time value
R	Input	BOOL	I, Q, M, D, L, T, C, P	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (BI-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the timer

For additional information on valid data types, refer to "See also".

Pulse timing diagram

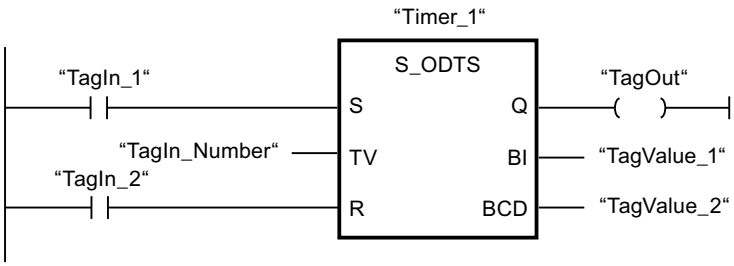
The following figure shows the pulse timing diagram of the "Assign retentive on-delay timer parameters and start" instruction:



t = programmed timer

Example

The following example shows how the instruction works:



The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "0" to "1". The timer expires with the time value of the operand "TagIn_Number", even if the signal state of the operand "TagIn_1" changes to "0". When the timer expires, the operand "TagOut" is set to "1". If the signal state of the operand "TagIn_1" changes from "0" to "1" while the timer is running, the timer is restarted.

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

L: Load timer value (Page 5934)

S_OFFDT: Assign off-delay timer parameters and start

Description

The "Assign off-delay timer parameters and start" instruction starts a programmed timer when a change from "1" to "0" (negative signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV). As long as the timer is running or input S returns signal state "1", then output Q has signal state "1". When the timer expires and the signal state at input S is "0", output Q is reset to the signal state "0". If the signal state at input S changes from "0" to "1" while the timer is running, the timer is stopped. The timer is only restarted after a falling signal edge is detected at input S.

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed. The current time value is output BI-coded at output BI and BCD-coded at output BCD.

Signal state "1" at input R resets the current time value and time base to "0". In this case, the signal state at output Q is "0".

The "Assign off-delay timer parameters and start" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

The instruction data is updated at every access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameters

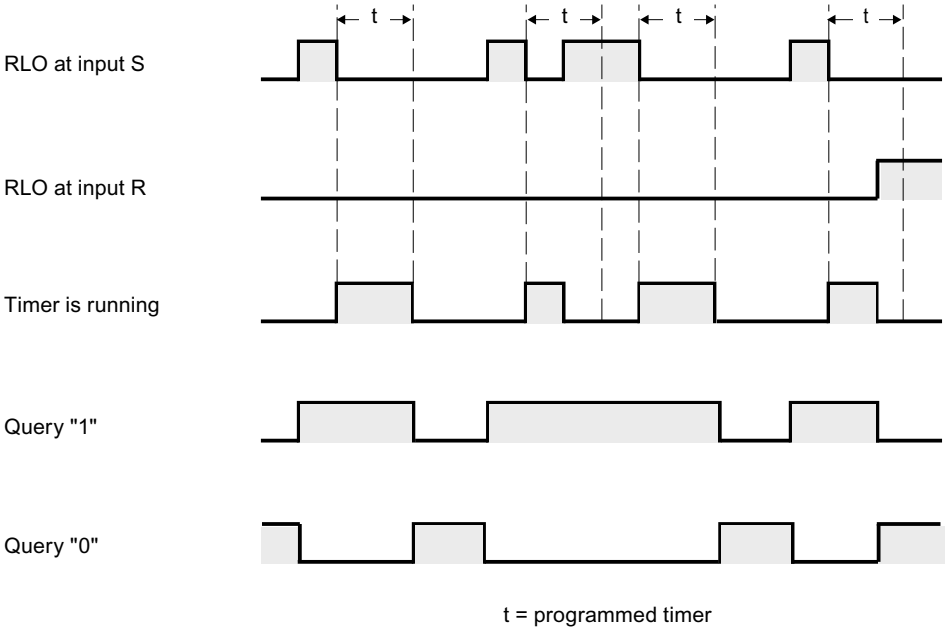
The following table shows the parameters of the "Assign off-delay timer parameters and start" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, D, L	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Preset time value
R	Input	BOOL	I, Q, M, D, L, T, C, P	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (BI-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the timer

For additional information on valid data types, refer to "See also".

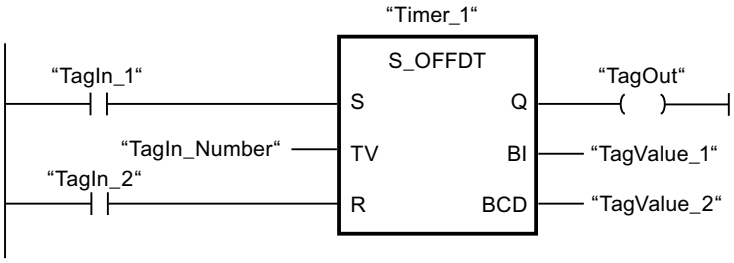
Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign off-delay timer parameters and start" instruction:



Example

The following example shows how the instruction works:



The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "1" to "0". The timer expires with the time value of the operand "TagIn_Number". The operand "TagOut" is set to "1" when the timer is running and when the operand "TagIn_1" has the signal state "0". If the signal state of the operand "TagIn_1" changes from "0" to "1" while the timer is running, the timer is reset.

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

L: Load timer value (Page 5934)

---(SP): Start pulse timer

Description

The "Start pulse timer" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO). The time runs with the specified duration as long as the RLO has the signal state "1". As long as the timer is running, the querying of the timer status for "1" returns the signal state "1". If there is a change from "1" to "0" in the RLO before the time value has elapsed, the timer stops. In this case, the querying of the timer status for "1" returns the signal state "0".

The duration is made up internally of a time value and a time base. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed.

The "Start pulse timer" instruction requires a preceding logic operation for edge evaluation and can only be placed at the right side of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameter

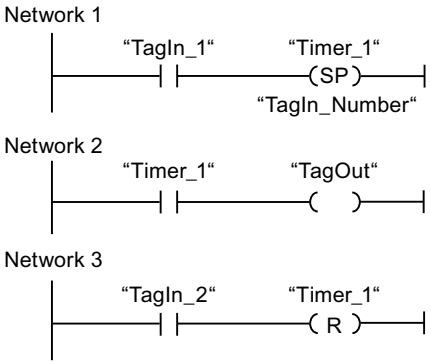
The following table shows the parameters of the "Start pulse timer" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Time duration>	Input	S5TIME, WORD	I, Q, M, D, L or constant	Duration with which the timer expires.
<Timer>	Output	TIMER	T	Timer which is started. The number of timers depends on the CPU.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "0" to "1". The timer expires with the time value of the operand "TagIn_Number" as long as the signal state of the operand "TagIn_1" is "1". If the signal state of the operand "TagIn_1" changes from "1" to "0" before the timer expires, the timer is stopped. As long as the timer is running, the operand "TagOut" has the signal state "1". A signal state change of the operand "TagIn_1" from "0" to "1" resets the timer, which stops the timer and sets the current time value to "0".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

L: Load timer value (Page 5934)

---(SE): Start extended pulse timer

Description

The "Start extended pulse timer" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO). The timer expires with the specified duration, even if the RLO changes to the signal state "0". As long as the timer is running, the querying of the timer status for "1" returns the signal state "1". If the RLO changes from "0" to "1" while the timer is running, the timer is restarted with the programmed duration. The querying of the timer status for "1" returns the signal state "0" when the timer expires.

The duration is made up internally of a time value and a time base. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed.

The "Start extended pulse timer" instruction requires a preceding logic operation for edge evaluation and can only be placed at the right side of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameter

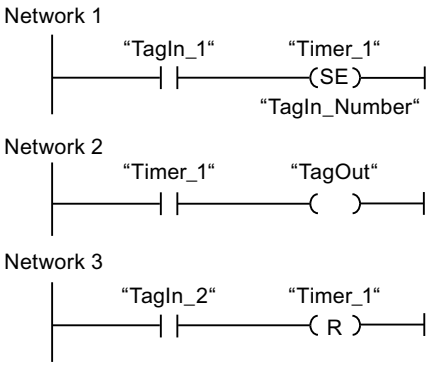
The following table shows the parameters of the "Start extended pulse timer" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Time duration>	Input	S5TIME, WORD	I, Q, M, D, L or constant	Duration with which the timer expires.
<Timer>	Output	TIMER	T	Timer which is started. The number of timers depends on the CPU.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "0" to "1". The timer expires with the time value of the operand "TagIn_Number" without being affected by a negative edge in the RLO. As long as the timer is running, the operand "TagOut" has the signal state "1". If the signal state of the operand "TagIn_1" changes from "0" to "1" before the timer expires, the timer is restarted.

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

L: Load timer value (Page 5934)

---(SD): Start on-delay timer

Description

The "Start on-delay timer" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO). The timer runs for the specified duration as long as the RLO is "1". If the timer expires and the RLO still has the signal state "1", the query of the timer status for "1" returns the signal state "1". If the RLO changes from "1" to "0" while the timer is running, the timer is stopped. In this case, the querying of the timer status for "1" returns the signal state "0".

The duration is made up internally of a time value and a time base. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed.

The "Start on-delay timer" instruction requires a preceding logic operation for edge evaluation and can only be placed at the right side of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

Parameter

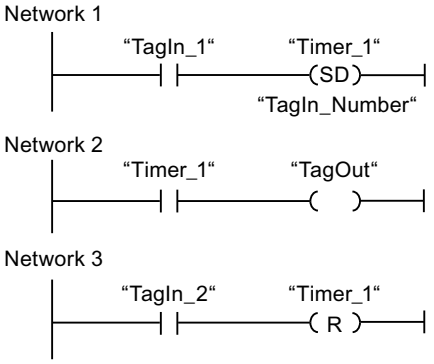
The following table shows the parameters of the "Start on-delay timer" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Time duration>	Input	S5TIME, WORD	I, Q, M, D, L or constant	Duration with which the timer expires.
<Timer>	Output	TIMER	T	Timer which is started. The number of timers depends on the CPU.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "0" to "1". The timer expires with the time value of the operand "TagIn_Number". If the timer expires and the RLO has the signal state "1", the operand "TagOut" is set to "1". If the signal state of the operand "TagIn_1" changes from "1" to "0" before the timer expires, the timer is stopped. If the signal state of the operand "TagIn_2" is "1", the timer "Timer_1" is reset, which stops the timer and sets the current time value to "0".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

L: Load timer value (Page 5934)

---(SS): Start retentive on-delay timer

Description

The "Start retentive on-delay timer" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO). The timer expires with the specified duration, even if the RLO changes to the signal state "0". When the timer expires, the querying of the timer status for "1" returns the signal state "1". After expiry of the timer, the timer can only be restarted if it is explicitly reset.

The duration is made up internally of a time value and a time base. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed.

The "Start retentive on-delay timer" instruction requires a preceding logic operation for edge evaluation and can only be placed at the right side of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameter

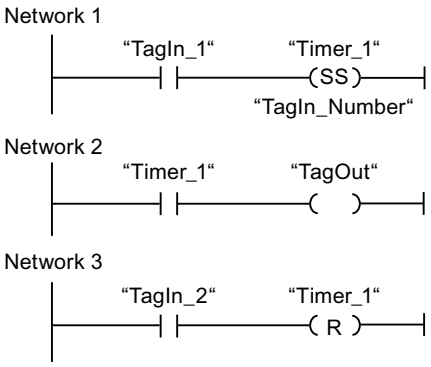
The following table shows the parameters of the "Start retentive on-delay timer" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Time duration>	Input	S5TIME, WORD	I, Q, M, D, L or constant	Duration with which the timer expires.
<Timer>	Output	TIMER	T	Timer which is started. The number of timers depends on the CPU.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "0" to "1". The timer expires with the time value of the operand "TagIn_Number". When the timer expires, the operand "TagOut" is set to "1". If the signal state of the operand "TagIn_1" changes from "0" to "1" while the timer is running, the timer is restarted. If the signal state of

the operand "TagIn_2" is "1", the timer "Timer_1" is reset, which stops the timer and sets the current time value to "0".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

L: Load timer value (Page 5934)

---(SF): Start off-delay timer

Description

The "Start off-delay timer" instruction starts a programmed timer when a change from "1" to "0" (negative signal edge) is detected in the result of logic operation (RLO). The timer expires with the specified duration. As long as the timer is running, the querying of the timer status for "1" returns the signal state "1". If the RLO changes from "0" to "1" while the timer is running, the timer is reset. The timer is always restarted when the RLO changes from "1" to "0".

The duration is made up internally of a time value and a time base. When the instruction is started, the programmed time value is counted down towards zero. The time base indicates the time period with which the time value is changed.

The "Start off-delay timer" instruction requires a preceding logic operation for edge evaluation and can only be placed at the right side of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Parameter

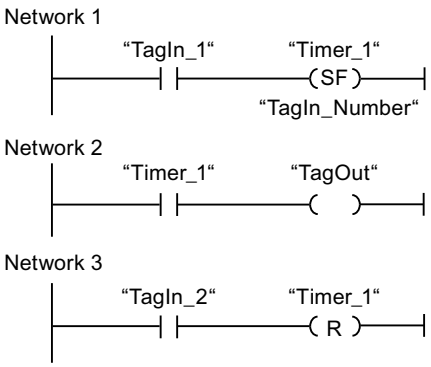
The following table shows the parameters of the "Start off-delay timer" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Time duration>	Input	S5TIME, WORD	I, Q, M, D, L or constant	Duration with which the timer expires.
<Timer>	Output	TIMER	T	Timer which is started. The number of timers depends on the CPU.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The timer "Timer_1" is started when the signal state of the operand "TagIn_1" changes from "1" to "0". The timer expires with the time value of the operand "TagIn_Number". As long as the timer is running, the operand "TagOut" is set to "1". If the signal state of the operand "TagIn_1" changes from "1" to "0" while the timer is running, the timer is restarted. If the signal state of the operand "TagIn_2" is "1", the timer "Timer_1" is reset, which stops the timer and sets the current time value to "0".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

L: Load timer value (Page 5934)

Counter operations

IEC Counters

CTU: Count up

Description

You can use the "Count up" instruction to increment the value at output CV. When the signal state at the CU input changes from "0" to "1" (positive signal edge), the instruction executes and the current counter value at the CV output is incremented by one. When the instruction executes for the first time, the current counter value at the CV output is set to zero. The counter value is incremented each time a positive signal edge is detected, until it reaches the high limit for the data type INT specified at the CV output. When the high limit is reached, the signal state at the CU input no longer has an effect on the instruction.

You can scan the counter status at the Q output. The signal state at the Q output is determined by the PV parameter. If the current counter value is greater than or equal to the value of the PV parameter, the Q output is set to signal state "1". In all other cases, the Q output has signal state "0". You can also specify a constant for the PV parameter.

The value at the CV output is reset to zero when the signal state at input R changes to "1". As long as the R input has signal state "1", the signal state at the CU input has no effect on the instruction.

Each call of the "Count up" instruction must be assigned an IEC counter in which the instruction data is stored. You can declare an IEC counter as follows:

- Declaration of a data block of the type CTU (for example, "CTU_DB")
- Declaration as a local tag of the type CTU in the "Static" section of a block (for example, #MyCTU_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of "Count up" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized with the value "1" for the R parameter of the instruction in a startup OB. If instances of the "Count up" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Note

Only use a counter at a single location in the program to avoid risk of counting errors.

The execution of the "Count up" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameter

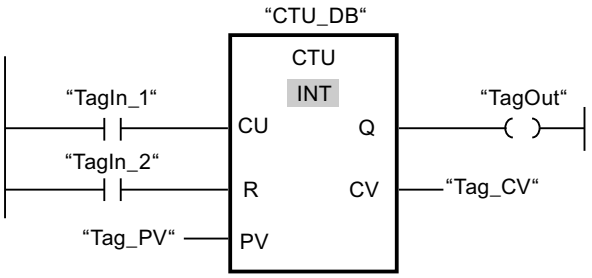
The following table shows the parameters of the "Count up" instruction:

Parameter	Declaration	Data type	Memory area	Description
CU	Input	BOOL	I, Q, M, D, L	Count input
R	Input	BOOL	I, Q, M, D, L, T, C, P	Reset input
PV	Input	INT	I, Q, M, D, L, P or constant	Value at which the output Q is set.
Q	Output	BOOL	I, Q, M, D, L	Counter status
CV	Output	INT	I, Q, M, D, L, P	Current counter value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



When the signal state of the "TagIn_1" operand changes from "0" to "1", the "Count up" instruction executes and the current counter value of the operand "Tag_CV" is incremented by one. With each additional positive signal edge, the counter value is incremented until the high limit value of the data type (INT = 32767) is reached.

The value of the PV parameter is adopted as the limit for determining the "TagOut" output. The "TagOut" output has signal state "1" as long as the current counter value is greater than or equal to the value of the "Tag_PV" operand. In all other cases, the "TagOut" output has signal state "0".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Call function blocks as single or multi-instances (Page 2182)

CTD: Count down

Description

You can use the "Count down" instruction to decrement the value at output CV. When the signal state at the CD input changes from "0" to "1" (positive signal edge), the instruction executes and the current counter value at the CV output is decremented by one. When the instruction executes the first time, the counter value of the CV parameter is set to the value of the PV parameter. Each time a positive signal edge is detected, the counter value is decremented until it reaches the low limit value of the specified data type (INT). When the low limit is reached, the signal state at the CD input no longer has an effect on the instruction.

You can scan the counter status at the Q output. If the current counter value is less than or equal to zero, the Q output is set to signal state "1". In all other cases, the Q output has signal state "0". You can also specify a constant for the parameter PV.

The value at the CV output is set to the value of the PV parameter when the signal state at the LD input changes to "1". As long as the LD input has signal state "1", the signal state at the CD input has no effect on the instruction.

Each call of the "Count down" instruction must be assigned an IEC counter in which the instruction data is stored. You can declare an IEC counter as follows:

- Declaration of a data block of the type CTD (for example, "CTD_DB")
- Declaration as a local tag of the type CTD in the "Static" section of a block (for example, #MyCTD_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (mult-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of "Count down" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, you need to call the instances to be initialized with the value "1" for the LD parameter of the instruction in a startup OB. In this case, the desired initial value for the CV parameter is specified in the PV parameter.

Note

Only use a counter at a single location in the program to avoid risk of counting errors.

The execution of the "Count down" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

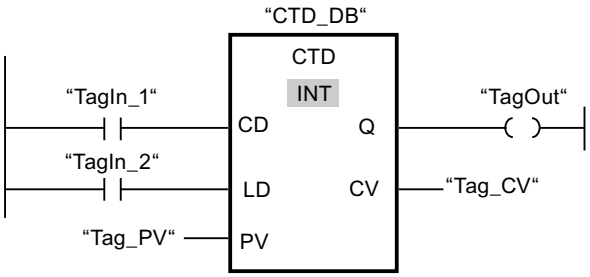
The following table shows the parameters of the "Count down" instruction:

Parameter	Declaration	Data type	Memory area	Description
CD	Input	BOOL	I, Q, M, D, L	Count input
LD	Input	BOOL	I, Q, M, D, L, P	Load input
PV	Input	INT	I, Q, M, D, L, P or constant	Value at which the Q output is set
Q	Output	BOOL	I, Q, M, D, L	Counter status
CV	Output	INT	I, Q, M, D, L, P	Current counter value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



When the signal state of the "TagIn_1" operand changes from "0" to "1", the "Count down" instruction is executed and the value at the "Tag_CV" output is decremented by one. With each additional positive signal edge, the counter value is decremented until it reaches the low limit value of the data type (INT = -32768).

The "TagOut" output has signal state "1" as long as the current counter value is less than or equal to zero. In all other cases, the "TagOut" output has signal state "0".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Call function blocks as single or multi-instances (Page 2182)

CTUD: Count up and down

Description

You can use the "Count up and down" instruction to increment and decrement the counter value at the CV output. If the signal state at the CU input changes from "0" to "1" (positive signal edge), the current counter value is incremented by one and stored at the CV output. If the signal state at the CD input changes from "0" to "1" (positive signal edge), the counter value at the CV output is decremented by one. If there is a positive signal edge at the CU and CD inputs in one program cycle, the current counter value at the CV output remains unchanged.

The counter value can be incremented until it reaches the high limit of the data type INT specified at the CV output. When the high limit value is reached, the counter value is no longer incremented on a positive signal edge. When the low limit of the specified data type (INT) is reached, the counter value is not decremented any further.

When the signal state at the LD input changes to "1", the counter value at the CV output is set to the value of the PV parameter. As long as the LD input has the signal state "1", the signal state at the CU and CD inputs has no effect on the instruction.

The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has signal state "1", a change in the the signal state of the CU, CD and LD inputs has no effect on the "Count up and down" instruction.

You can scan the current status of the up counter at the QU output. If the current counter value is greater than or equal to the value of the PV parameter, the QU output is set to signal state "1". In all other cases, the QU output has signal state "0".

You can scan the current status of the down counter at the QD output. If the current counter value is less than or equal to zero, the QD output is set to signal state "1". In all other cases, the QD output has signal state "0".

Each call of the "Count up and down" instruction must be assigned an IEC counter in which the instruction data is stored. You can declare an IEC counter as follows:

- Declaration of a data block of the type CTUD (for example, "CTUD_DB")
- Declaration as a local tag of the type CTUD in the "Static" section of a block (for example, #MyCTUD_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of "Count up and down" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, you must call the instances to be initialized with the following parameter values in a startup OB:

- When used as counter up, the value at the R parameter must be set to "1".
- When used as down counter, the value at the LD parameter must be set to "1". In this case you specify the desired initial value for the CV parameter in the PV parameter.

If instances of the "Count up and down" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Note

Only use a counter at a single location in the program to avoid risk of counting errors.

The execution of the "Count up and down" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

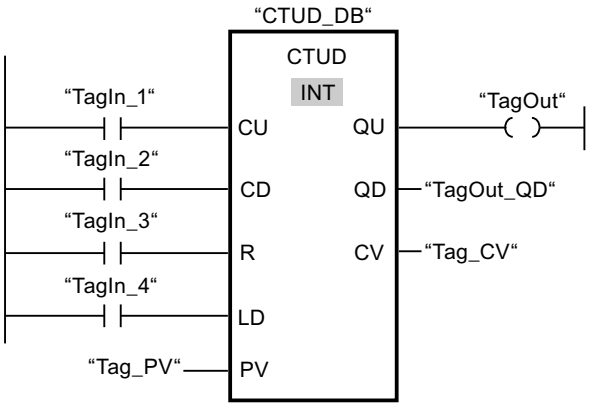
The following table shows the parameters of the "Count up and down" instruction:

Parameter	Declaration	Data type	Memory area	Description
CU	Input	BOOL	I, Q, M, D, L	Count up input
CD	Input	BOOL	I, Q, M, D, L	Count down input
R	Input	BOOL	I, Q, M, D, L, T, C, P	Reset input
LD	Input	BOOL	I, Q, M, D, L, P	Load input
PV	Input	INT	I, Q, M, D, L, P or constant	Value at which the QU output is set
QU	Output	BOOL	I, Q, M, D, L	Status of the incremental counter
QD	Output	BOOL	I, Q, M, D, L	Status of the down counter
CV	Output	INT	I, Q, M, D, L, P	Current counter value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the signal state at the "TagIn_1" or "TagIn_2" input changes from "0" to "1" (positive signal edge), the "Count up and down" instruction is executed. When there is a positive signal edge at the "TagIn_1" input, the current counter value is incremented by one and stored at the "Tag_CV" output. When there is a positive signal edge at the "TagIn_2" input, the counter value is decremented by one and stored at the "Tag_CV" output. When there is a positive signal edge at the CU input, the counter value is incremented until it reaches the high limit of 32767. If input CD has a positive signal edge, the counter value is decremented until it reaches the low limit value of INT = -32768.

The "TagOut" output has signal state "1" as long as the current counter value is greater than or equal to the value at the "Tag_PV" input. In all other cases, the "TagOut" output has signal state "0".

The "TagOut_QD" output has signal state "1" as long as the current counter value is less than or equal to zero. In all other cases, the "TagOut_QD" output has signal state "0".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Call function blocks as single or multi-instances (Page 2182)

SIMATIC Counters

S_CUD: Assign parameters and count up / down

Description

You can use the "Assign parameters and count up / down" instruction to increment or decrement the value of a counter. If the signal state at the CU input changes from "0" to "1" (positive signal edge), the current counter value is incremented by one. If the signal state at the CD input changes from "0" to "1" (positive signal edge), the counter value is decremented by one. The current counter value is output as a hexadecimal value at output CV and BCD-coded at output CV_BCD. If there is a positive signal edge at the CU and CD inputs in one program cycle, the counter value remains unchanged.

The counter value can be incremented until the high limit of "999" is reached. When the high limit value is reached, the counter value is no longer incremented on a positive signal edge. When the low limit "0" is reached, the counter value is not decremented any further.

When the signal state at input S changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if RLO is "1" at the inputs CU and CD, the counter counts accordingly in the next scan cycle, even if no change in the signal edge was detected.

The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has the signal state "1", processing of the signal state of the CU, CD and S inputs has no effect on the counter value.

The signal state at output Q is "1" if the counter value is greater than zero. If the counter value is equal to zero, output Q has the signal state "0".

Note

Only use a counter at a single location in the program to avoid risk of counting errors.

The "Assign parameters and count up / down" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

Parameters

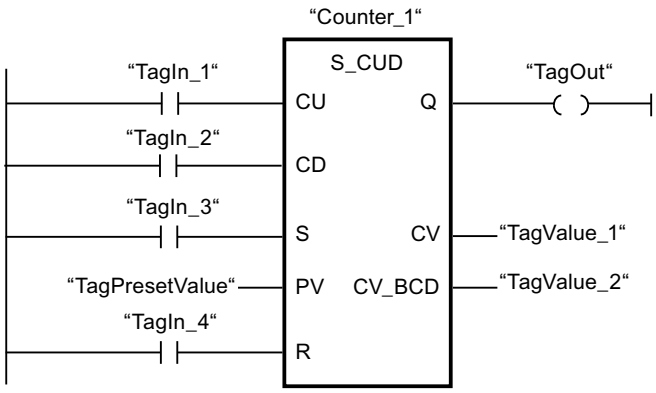
The following table shows the parameters of the "Assign parameters and count up / down" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Counter>	InOut	COUNTER	C	Counter of the instruction The number of counters depends on the CPU.
CU	Input	BOOL	I, Q, M, D, L	Count up input
CD	Input	BOOL	I, Q, M, D, L, T, C	Count down input
S	Input	BOOL	I, Q, M, D, L, T, C	Input for presetting counter
PV	Input	WORD	I, Q, M, D, L, P	Preset counter value (C#0 to C#999)
R	Input	BOOL	I, Q, M, D, L, T, C	Reset input
CV	Output	WORD, S5TIME, DATE	I, Q, M, D, L, P	Current counter value (hexadecimal)
CV_BCD	Output	WORD, S5TIME, DATE	I, Q, M, D, L, P	Current counter value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the counter

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the signal state at the input "TagIn_1" or at the input "TagIn_2" changes from "0" to "1" (positive edge), the instruction "Assign parameters and count up / down is executed. When there is a positive signal edge at the "TagIn_1" input and the current counter value is less than "999", the counter value is incremented by one. When there is a positive signal edge at the "TagIn_2" input and the current counter value is greater than "0", the counter value is decremented by one.

When the signal state at input "TagIn_3" changes from "0" to "1", the counter value is set to the value of the operand "TagPresetValue". The counter value is reset to "0" when the "TagIn_4" operand has signal state "1".

The current counter value is saved as a hexadecimal value in the operand "TagValue_1" and BCD-coded in the operand "TagValue_2".

The "TagOut" output has the signal state "1" as long as the current counter value is not equal to "0".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

S_CU: Assign parameters and count up

Description

You can use the "Assign parameters and count up" instruction to increment the value of a counter . If the signal state at the CU input changes from "0" to "1" (positive signal edge), the current counter value is incremented by one. The current counter value is output as a hexadecimal value at output CV and BCD-coded at output CV_BCD. The counter value can be incremented until the limit of "999" is reached. When the limit is reached, the counter value is no longer incremented on a positive signal edge.

When the signal state at input S changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if RLO at input CU is "1", the counter will count accordingly in the next scan cycle, even when no change has been detected in the signal edge.

The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has the signal state "1", processing of the signal state of the CU and S inputs has no effect on the counter value.

The signal state at output Q is "1" if the counter value is greater than zero. If the counter value is equal to zero, output Q has the signal state "0".

Note

Only use a counter at a single location in the program to avoid risk of counting errors.

The "Assign parameters and count up" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

Parameters

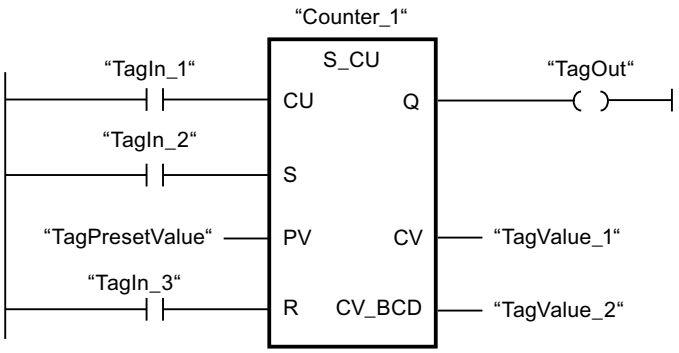
The following table shows the parameters of the "Assign parameters and count up" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Counter>	InOut	COUNTER	C	Counter of the instruction The number of counters depends on the CPU.
CU	Input	BOOL	I, Q, M, D, L	Count up input
S	Input	BOOL	I, Q, M, D, L, T, C	Input for presetting counter
PV	Input	WORD	I, Q, M, D, L, P	Preset counter value (C#0 to C#999)
R	Input	BOOL	I, Q, M, D, L, T, C	Reset input
CV	Output	WORD, S5TIME, DATE	I, Q, M, D, L, P	Current counter value (hexadecimal)
BCD	Output	WORD, S5TIME, DATE	I, Q, M, D, L, P	Current counter value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the counter

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



When the signal state at the "TagIn_1" input changes from "0" to "1" (positive signal edge) and the current counter value is less than "999", the counter value is incremented by one. When the signal state at input "TagIn_2" changes from "0" to "1", the counter value is set to the value of the operand "TagPresetValue". The counter value is reset to "0" when the "TagIn_3" operand has signal state "1".

The current counter value is saved as a hexadecimal value in the operand "TagValue_1" and BCD-coded in the operand "TagValue_2".

The "TagOut" output has the signal state "1" as long as the current counter value is not equal to "0".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

S_CD: Assign parameters and count down

Description

You can use the "Assign parameters and count down" instruction to decrement the value of a counter . If the signal state at the CD input changes from "0" to "1" (positive signal edge), the counter value is decremented by one. The current counter value is output as a hexadecimal value at output CV and BCD-coded at output CV_BCD. The counter value can be decremented until the low limit of "0" is reached. When the low limit is reached, the counter value is no longer decremented on a positive signal edge.

When the signal state at input S changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if RLO at input CD is "1", the counter will count accordingly in the next scan cycle, even when no change has been detected in the signal edge.

The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has the signal state "1", processing of the signal state of the CD and S inputs has no effect on the counter value.

The signal state at output Q is "1" if the counter value is greater than zero. If the counter value is equal to zero, output Q has the signal state "0".

Note

Only use a counter at a single location in the program to avoid risk of counting errors.

The "Assign parameters and count down" instruction requires a preceding logic operation for edge evaluation and can be placed within or at the end of the network.

Parameters

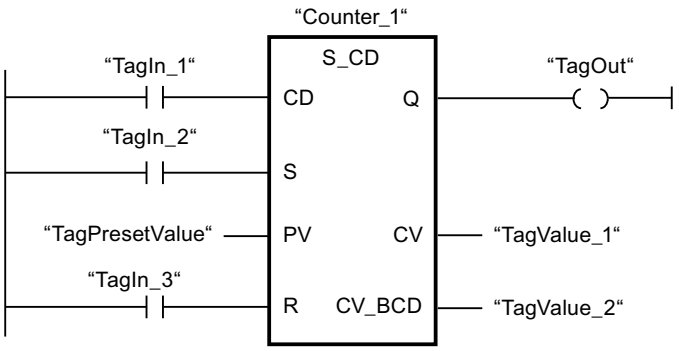
The following table shows the parameters of the "Assign parameters and count down" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Counter>	InOut	COUNTER	C	Counter of the instruction The number of counters depends on the CPU.
CD	Input	BOOL	I, Q, M, D, L	Count down input
S	Input	BOOL	I, Q, M, D, L, T, C	Input for presetting counter
PV	Input	WORD	I, Q, M, D, L, P	Preset counter value (C#0 to C#999)
R	Input	BOOL	I, Q, M, D, L, T, C	Reset input
CV	Output	WORD, S5TIME, DATE	I, Q, M, D, L, P	Current counter value (hexadecimal)
BCD	Output	WORD, S5TIME, DATE	I, Q, M, D, L, P	Current counter value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the counter

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



When the signal state at the "TagIn_1" input changes from "0" to "1" (positive signal edge) and the current counter value is greater than "0", the counter value is decremented by one. When the signal state at input "TagIn_2" changes from "0" to "1", the counter value is set to the value of the operand "TagPresetValue". The counter value is reset to "0" when the "TagIn_3" operand has signal state "1".

The current counter value is saved as a hexadecimal value in the operand "TagValue_1" and BCD-coded in the operand "TagValue_2".

The "TagOut" output has the signal state "1" as long as the current counter value is not equal to "0".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

---(SC): Set counter value

Description

You can use the "Set counter value" instruction to set the value of a counter. The instruction is executed when the result of logic operation (RLO) at the input changes from "0" to "1". When the instruction is executed, the counter is set to the specified counter value.

The "Set counter value" instruction requires a preceding logic operation for edge evaluation and can only be placed at the right side of the network.

Parameter

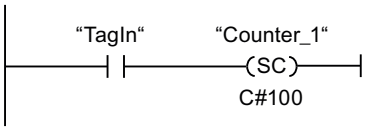
The following table shows the parameters of the "Set counter value" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Count value>	Input	WORD	I, Q, M, D, L or constant	Value with which the counter is preset in the BCD format. (C#0 to C#999)
<Counter>	InOut/Input	COUNTER	C	Counter that is preset

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The counter "Counter_1" is preset with the value "100" when the signal state of the operand "TagIn" changes from "0" to "1".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

---(CU): Count up

Description

With the "Count up" instruction you can increment the value of the specified counter by one if there is a positive signal edge in the result of logic operation (RLO). The counter value can be incremented until the limit of "999" is reached. When the limit is reached, the counter value is no longer incremented on a positive signal edge.

The "Count up" instruction requires a preceding logic operation for edge evaluation and can only be placed at the right side of the network.

Parameters

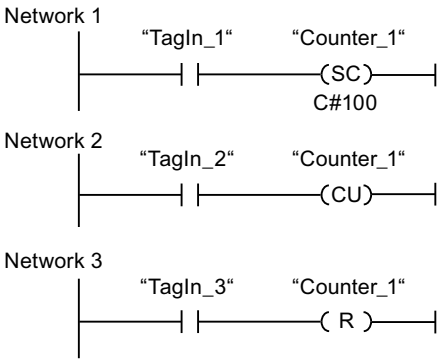
The following table shows the parameters of the "Count up" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Counter>	InOut/Input	COUNTER	C	Counter whose value is incremented.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



When the signal state of the "TagIn_1" operand changes from "0" to "1" (positive signal edge), the counter "Counter_1" is preset with the value "100".

The value of the counter "Counter_1" is incremented by one when the signal state of the operand "TagIn_2" changes from "0" to "1".

When the operand "TagIn_3" has the signal state "1", the value of the counter "Counter_1" is reset to "0".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

---(CD): Count down

Description

With the "Count down" instruction you can increment the value of the specified counter by one if there is a positive signal edge in the result of logic operation (RLO). The counter value can

be decremented until the limit of "0" is reached. When the limit is reached, the counter value is no longer changed on a positive signal edge.

The "Count down" instruction requires a preceding logic operation for edge evaluation and can only be placed at the right side of the network.

Parameter

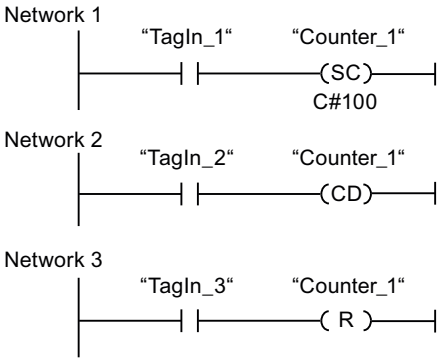
The following table shows the parameters of the "Count down" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Counter>	InOut/Input	COUNTER	C	Counter whose value is decremented.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



When the signal state of the "TagIn_1" operand changes from "0" to "1" (positive signal edge), the counter "Counter_1" is preset with the value "100".

The value of the counter "Counter_1" is decremented by one when the signal state of the operand "TagIn_2" changes from "0" to "1".

When the operand "TagIn_3" has the signal state "1", the value of the counter "Counter_1" is reset to "0".

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Comparator operations

CMP ==: Equal

Description

You can use the "Equal" instruction to determine if a first comparison value (<Operand1>) is equal to a second comparison value (<Operand2>). Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify the first comparison value (<Operand1>) at the operand placeholder above the instruction. Specify the second comparison value (<Operand2>) at the operand placeholder below the instruction.

Note

Comparison of floating-point numbers

If you want to compare the data types REAL or LREAL, instead of the instruction "CMP ==: Equal", use the instruction "IN_RANGE: Value within range".

Parameters

The following table shows the parameters of the "Equal" instruction:

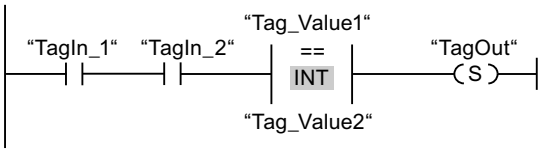
Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P or constant	First comparison value
<Operand2>	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also":

Example

The following example shows how the instruction works:



The output "TagOut" is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled if "Tag_Value1" = "Tag_Value2".

See also

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Overview of the valid data types (Page 2236)

Setting the status bit for comparison instructions (Page 2460)

Basic information on the status word (Page 2453)

CMP <>: Not equal

Description

You can use the "Not equal" instruction to determine if a first comparison value (<Operand1>) is not equal to a second comparison value (<Operand2>). Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify the first comparison value (<Operand1>) at the operand placeholder above the instruction. Specify the second comparison value (<Operand2>) at the operand placeholder below the instruction.

Parameter

The following table shows the parameters of the "Not equal" instruction:

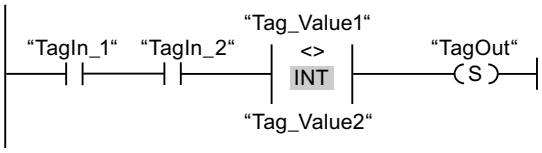
Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P or constant	First comparison value
<Operand2>	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled if ("Tag_Value1" <> "Tag_Value2").

See also

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Overview of the valid data types (Page 2236)

Setting the status bit for comparison instructions (Page 2460)

Basic information on the status word (Page 2453)

CMP >=: Greater or equal

Description

You can use the "Greater or equal" instruction to determine if a first comparison value (<Operand1>) is greater than or equal to a second comparison value (<Operand2>). Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify the first comparison value (<Operand1>) at the operand placeholder above the instruction. Specify the second comparison value (<Operand2>) in the operand placeholder below the instruction.

Parameter

The following table shows the parameters of the "Greater or equal" instruction:

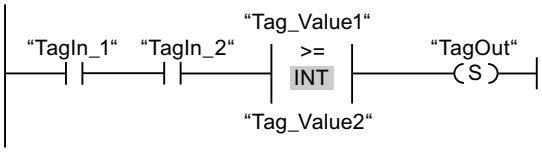
Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	First comparison value
<Operand2>	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled if ("Tag_Value1" >= "Tag_Value2").

See also

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Overview of the valid data types (Page 2236)

Example of detecting the fill level of a storage area (Page 3118)

Setting the status bit for comparison instructions (Page 2460)

Basic information on the status word (Page 2453)

CMP <=: Less or equal

Description

You can use the "Less or equal" instruction to determine if a first comparison value (<Operand1>) is less than or equal to a second comparison value (<Operand2>). Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify the first comparison value (<Operand1>) at the operand placeholder above the instruction. Specify the second comparison value (<Operand2>) at the operand placeholder below the instruction.

Parameter

The following table shows the parameters of the "Less or equal" instruction:

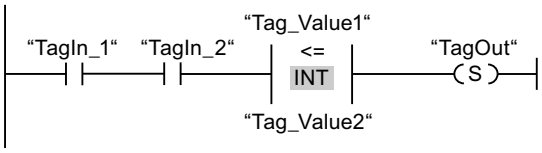
Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	First comparison value
<Operand2>	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled if ("Tag_Value1" <= "Tag_Value2").

See also

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Overview of the valid data types (Page 2236)

Setting the status bit for comparison instructions (Page 2460)

Basic information on the status word (Page 2453)

CMP >: Greater than

Description

You can use the "Greater than" instruction to determine if a first comparison value (<Operand1>) is greater than a second comparison value (<Operand2>). Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify the first comparison value (<Operand1>) at the operand placeholder above the instruction. Specify the second comparison value (<Operand2>) at the operand placeholder below the instruction.

Parameter

The following table shows the parameters of the "Greater than" instruction:

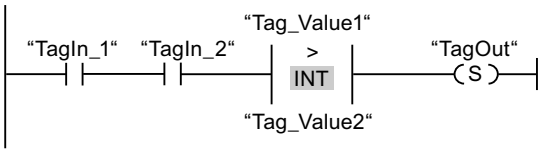
Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	First comparison value
<Operand2>	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled if ("Tag_Value1" > "Tag_Value2").

See also

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Overview of the valid data types (Page 2236)

Setting the status bit for comparison instructions (Page 2460)

Basic information on the status word (Page 2453)

CMP <: Less than

Description

You can use the "Less than" instruction to determine if a first comparison value (<Operand1>) is less than a second comparison value (<Operand2>). Both values to be compared must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0". The RLO of the instruction is logically combined with the RLO of the entire rung as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Specify the first comparison value (<Operand1>) at the operand placeholder above the instruction. Specify the second comparison value (<Operand2>) at the operand placeholder below the instruction.

Parameter

The following table shows the parameters of the "Less than" instruction:

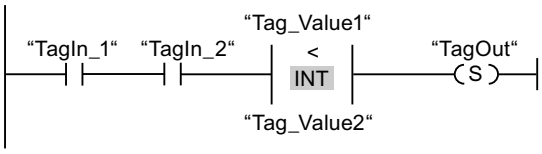
Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	First comparison value
<Operand2>	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "TagOut" output is set when the following conditions are fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled if ("Tag_Value1" < "Tag_Value2").

See also

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Overview of the valid data types (Page 2236)

Example of detecting the fill level of a storage area (Page 3118)

Setting the status bit for comparison instructions (Page 2460)

Basic information on the status word (Page 2453)

Math functions

ADD: Add

Description

You can use the "Add" instruction to add the value at input IN1 and the value at input IN2 and query the sum at output OUT (OUT := IN1+IN2). A maximum of 2 inputs can be specified.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at the OUT output.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Add" instruction:

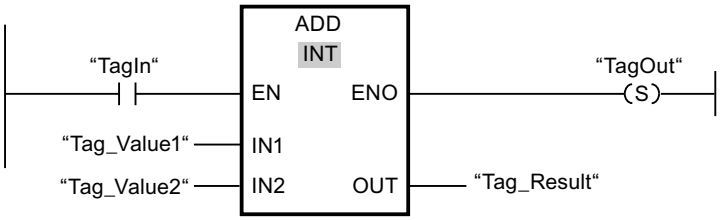
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	First number to be added
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Second number to be added
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Sum

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also":

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state "1", the "Add" instruction is executed. The value of operand "Tag_Value1" is added to the value of operand "Tag_Value2". The result of the addition is stored in operand "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Example of calculating an equation (Page 3121)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

SUB: Subtract

Description

You can use the "Subtract" instruction to subtract the value at input IN2 from the value at input IN1 and query the difference at output OUT (OUT := IN1-IN2).

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at the OUT output.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Subtract" instruction:

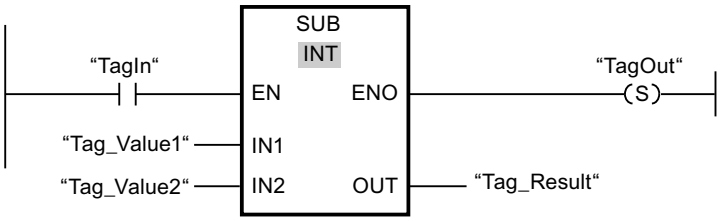
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Minuend
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Subtracting
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Difference

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state "1", the "Subtract" instruction is executed. The value of operand "Tag_Value2" is subtracted from the value of operand "Tag_Value1". The result of the subtraction is stored in operand "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

MUL: Multiply

Description

You can use the "Multiply" instruction to multiply the value at input IN1 with the value at input IN2 and query the product at output OUT (OUT := IN1*IN2). A maximum of 2 inputs can be specified.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The result is outside the range permitted for the data type specified at output OUT.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Multiply" instruction:

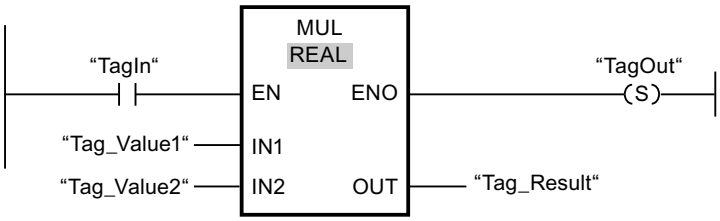
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Multiplier
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Number being multiplied
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Product

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state "1", the "Multiply" instruction is executed. The value of operand "Tag_Value1" is multiplied by the value of operand "Tag_Value2". The result of the multiplication is stored in operand "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)
Basics of the EN/ENO mechanism (Page 2447)
Inserting LAD elements (Page 2579)
Selecting the data type of a LAD element (Page 2583)
Editing LAD elements (Page 2594)
Inserting operands into LAD instructions (Page 2601)
Example of calculating an equation (Page 3121)
Basic information on the status word (Page 2453)
Setting the status bits in instructions with integers (Page 2458)
Setting the status bits in instructions with floating-point numbers (Page 2459)

DIV: Divide

Description

You can use the "Divide" instruction to divide the value at input IN1 by the value at input IN2 and query the quotient at output OUT (OUT := IN1/IN2).

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at the OUT output.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the "Divide" instruction:

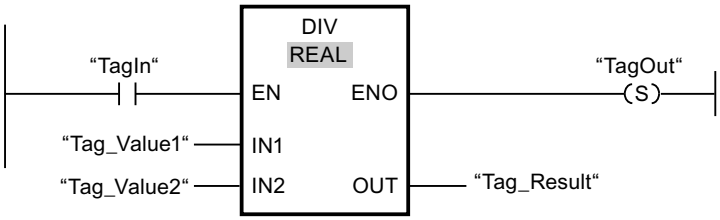
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Dividend
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Divisor
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Quotient value

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state "1", the "Divide" instruction is executed. The value of operand "Tag_Value1" is divided by the value of operand "Tag_Value2". The division result is stored in operand "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Example of calculating an equation (Page 3121)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

MOD: Return remainder of division

Description

You can use the "Return remainder of division" instruction to divide the value at input IN1 by the value at input IN2 and query the remainder of division at output OUT.

Parameters

The following table shows the parameters of the "Return remainder of division" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

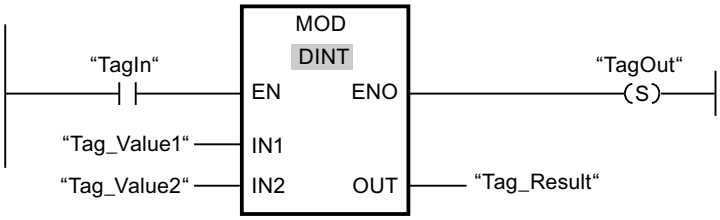
Parameter	Declaration	Data type	Memory area	Description
IN1	Input	Integers	I, Q, M, D, L, P or constant	Dividend
IN2	Input	Integers	I, Q, M, D, L, P or constant	Divisor
OUT	Output	Integers	I, Q, M, D, L, P	Remainder of division

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state "1", the "Return remainder of division" instruction is executed. The value of operand "Tag_Value1" is divided by the value of operand "Tag_Value2". The remainder is stored in operand "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

NEG: Create twos complement

Description

You can use the "Create twos complement" instruction to change the sign of the value at the IN input and query the result at the OUT output. If there is a positive value at input IN, for example, the negative equivalent of this value is sent to output OUT.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at the OUT output (only valid for integers).

Parameter

The following table shows the parameters of the "Create twos complement" instruction:

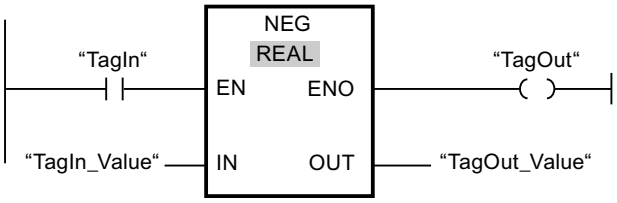
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Input value
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Twos complement of the input value

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state "1", the "Create twos complement" instruction is executed. The sign of the value at input "TagIn_Value" is changed and the result is provided at output "TagOut_Value". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)
Basics of the EN/ENO mechanism (Page 2447)
Inserting LAD elements (Page 2579)
Selecting the data type of a LAD element (Page 2583)
Editing LAD elements (Page 2594)
Inserting operands into LAD instructions (Page 2601)
Basic information on the status word (Page 2453)
Setting the status bits in instructions with integers (Page 2458)
Setting the status bits in instructions with floating-point numbers (Page 2459)

ABS: Form absolute value

Description

You can use the "Form absolute value" instruction to calculate the absolute value of the value specified at input IN. The result of the instruction is sent to the OUT output and can be queried there.

Parameters

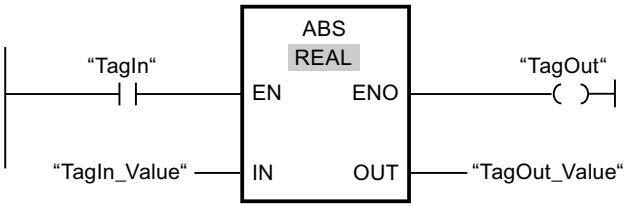
The following table shows the parameters of the "Form absolute value" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	REAL	I, Q, M, D, L, P or constant	Input value
OUT	Output	REAL	I, Q, M, D, L, P	Absolute value of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	TagIn_Value	-6.234
OUT	TagOut_Value	6.234

If operand "TagIn" has the signal state "1", the "Form absolute value" instruction is executed. The instruction calculates the absolute value of the value at input "TagIn_Value" and sends the result to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

MIN: Get minimum

Description

The "Get minimum" instruction compares the values at the inputs IN1, IN2 and IN3 and writes the lowest value to the OUT output. The instruction is only executed if the tags of all inputs are of the same data type.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- Enable input EN has the signal state "0".
- The specified tags are not of the same data type.
- A floating-point number has an invalid value.

Note

Only the data types INT, DINT and REAL are permitted at the input parameters, even when the IEC check is disabled. If you use the data types WORD or DWORD at the input parameters, the instruction will return an undefined result.

Parameters

The following table shows the parameters of the "Get minimum" instruction:

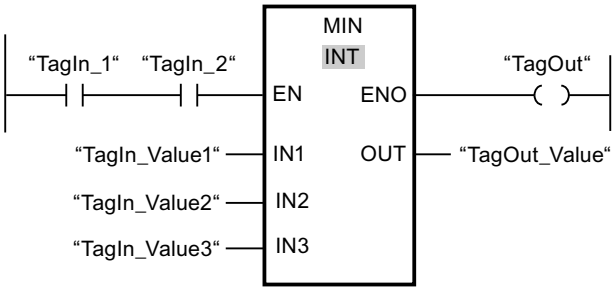
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	INT, DINT, REAL	I, Q, M, D, L, P	First input value
IN2	Input	INT, DINT, REAL	I, Q, M, D, L, P	Second input value
IN3	Input	INT, DINT, REAL	I, Q, M, D, L, P	Third input value
OUT	Output	INT, DINT, REAL	I, Q, M, D, L, P	Result

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
IN3	TagIn_Value3	13333
OUT	TagOut_Value	12222

If the "TagIn_1" and "TagIn_2" operands have signal state "1", the "Get minimum" instruction is executed. The instruction compares the values of the specified operands and copies the lowest value ("TagIn_Value1") to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

MAX: Get maximum

Description

The "Get maximum" instruction compares the values at the inputs IN1, IN2 and IN3 and writes the highest value to the OUT output. The instruction is only executed if the tags of all inputs are of the same data type.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- Enable input EN has the signal state "0".
- The specified tags are not of the same data type.
- A floating-point number has an invalid value.

Note

Only the data types INT, DINT and REAL are permitted at the input parameters, even when the IEC check is disabled. If you use the data types WORD or DWORD at the input parameters, the instruction will return an undefined result.

Parameters

The following table shows the parameters of the "Get maximum" instruction:

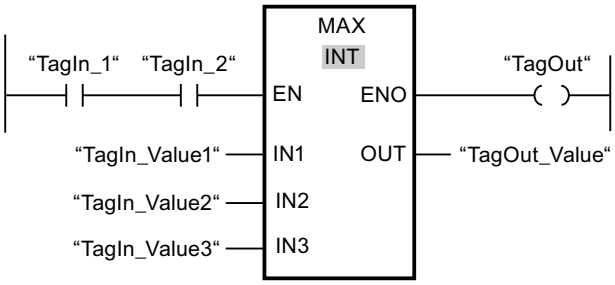
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	INT, DINT, REAL	I, Q, M, D, L, P	First input value
IN2	Input	INT, DINT, REAL	I, Q, M, D, L, P	Second input value
IN3	Input	INT, DINT, REAL	I, Q, M, D, L, P	Third input value
OUT	Output	INT, DINT, REAL	I, Q, M, D, L, P	Result

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	TagIn_Value1	12666
IN2	TagIn_Value2	14444
IN3	TagIn_Value3	13333
OUT	TagOut_Value	14444

If the "TagIn_1" and "TagIn_2" operands have signal state "1", the "Get maximum" instruction is executed. The instruction compares the values of the specified operands and copies the highest value ("TagIn_Value2") to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

LIMIT: Set limit value

Description

You use the "Set limit value" instruction to limit the value at input IN to the values at the inputs MN and MX. If the value at the IN input meets the condition MN <= IN <= MX, it is copied to the OUT output. If the condition is not fulfilled and the input value IN is below the low limit MN,

the output OUT is set to the value of the MN input. If the MX high limit is exceeded, the OUT output is set to the value of the MX input.

If the value at the MN input is greater than at the MX input, the result is undefined and the enable output ENO is "0".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- Enable input EN has the signal state "0".
- The specified tags are not of the same data type.
- An operand has an invalid value.
- The value at the MN input is greater than the value at the MX input.

Note

Only the data types INT, DINT and REAL are permitted at the input parameters, even when the IEC check is disabled. If you use the data types WORD or DWORD at the input parameters, the instruction will return an undefined result.

Parameters

The following table shows the parameters of the "Set limit value" instruction:

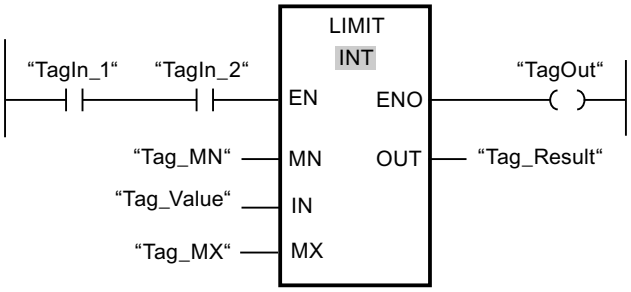
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
MN	Input	INT, DINT, REAL	I, Q, M, D, L, P	Low limit
IN	Input	INT, DINT, REAL	I, Q, M, D, L, P	Input value
MX	Input	INT, DINT, REAL	I, Q, M, D, L, P	High limit
OUT	Output	INT, DINT, REAL	I, Q, M, D, L, P	Result

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
MN	Tag_MN	12000
IN	Tag_Value	8000
MX	Tag_MX	16000
OUT	Tag_Result	12000

If operands "TagIn_1" and "TagIn_2" have the signal state "1", the "Set limit value" instruction is executed. The value of operand "Tag_Value" is compared with the values of operands "Tag_MN" and "Tag_MX". Since the value of operand "Tag_Value" is less than the low limit, the value of operand "Tag_MN" is copied to the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

SQR: Form square

Description

You can use the "Form square" instruction to square the value at the IN input and query the result at the OUT output.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The value at the IN input is not a valid floating-point number.

Parameters

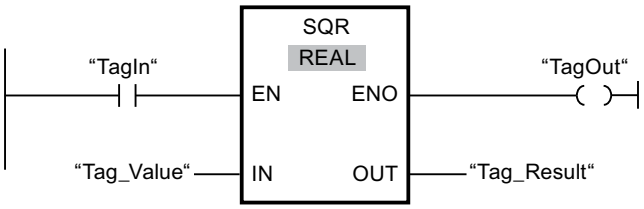
The following table shows the parameters of the "Form square" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Square of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_Value	5.0
OUT	Tag_Result	25.0

If operand "TagIn" has the signal state "1", the "Form square" instruction is executed. The instruction squares the value of operand "Tag_Value" and sends the result to the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

SQRT: Form square root

Description

You can use the "Form square root" instruction to find the square root of the value at the IN input and query the result at the OUT output. The instruction outputs a positive result if the input value is greater than zero. If input values are less than zero, the OUT output returns an invalid floating-point number. If the value at input IN is "0", then the result is also "0".

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The value at the IN input is not a valid floating-point number.
- The value at the IN input is negative.

Parameters

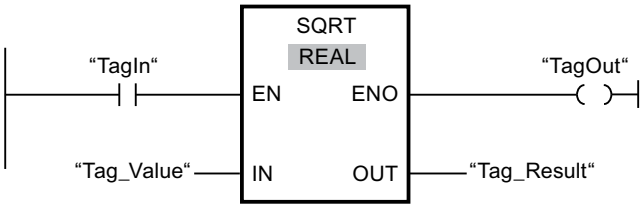
The following table shows the parameters of the "Form square root" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L	Square root of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_Value	25.0
OUT	Tag_Result	5.0

If operand "TagIn" has the signal state "1", the "Form square root" instruction is executed. The instruction finds the square root of the value of operand "Tag_Value" and stores the result in

the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

LN: Form natural logarithm

Description

You can use the "Form natural logarithm" instruction to calculate the natural logarithm to base e (e = 2.718282) of the value at input IN. The result is sent to the OUT output and can be queried there. The instruction outputs a positive result if the input value is greater than zero. If input values are less than zero, the OUT output returns an invalid floating-point number.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The enable input EN has the signal state "0".
- The value at the IN input is not a valid floating-point number.
- The value at the IN input is negative.

Parameter

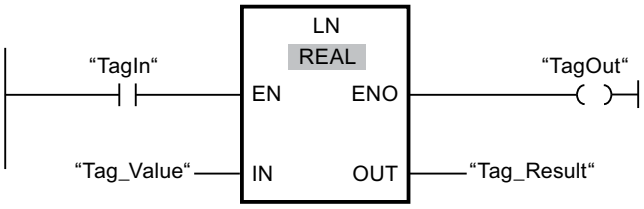
The following table shows the parameters of the "Form natural logarithm" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Natural logarithm of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state "1", the "Form natural logarithm" instruction is executed. The instruction forms the natural logarithm of the value at the "Tag_Value" input and stores the result in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

EXP: Form exponential value

Description

You can use the "Form exponential value" instruction to calculate the exponent from the base e (e = 2.718282) and the value specified at input IN. The result is provided at output OUT and can be queried there (OUT = e^{IN}).

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The enable input EN has the signal state "0".
- The value at the IN input is not a valid floating-point number.

Parameter

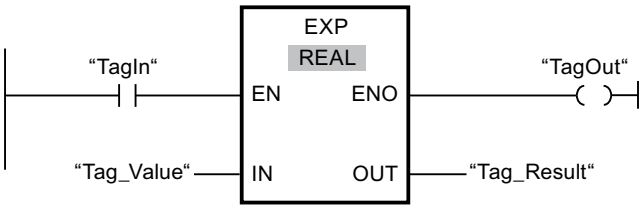
The following table shows the parameters of the "Form exponential value" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Exponential value of input value IN

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state "1", the "Form exponential value" instruction is executed. The instruction calculates the exponent from base e and the value of operand "Tag_Value" and sends the result to the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

SIN: Form sine value

Description

Use the "Form sine value" instruction to calculate the sine of the angle. The size of the angle is specified in radians at the IN input. The result of the instruction is sent to the OUT output and can be queried there.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The value at the IN input is not a valid floating-point number.

Parameters

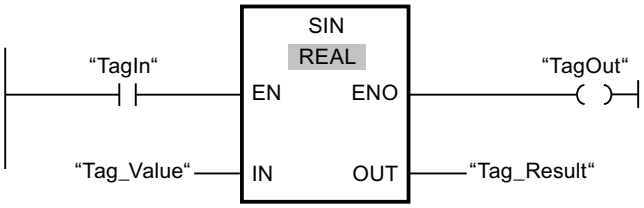
The following table shows the parameters of the "Form sine value" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Size of angle in radians
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Sine of the specified angle

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_Value	+1.570796 (π/2)
OUT	Tag_Result	1.0

If operand "TagIn" has the signal state "1", the "Form sine value" instruction is executed. The instruction calculates the sine of the angle specified at the "Tag_Value" input and stores the result in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)
Basics of the EN/ENO mechanism (Page 2447)
Inserting LAD elements (Page 2579)
Selecting the data type of a LAD element (Page 2583)
Editing LAD elements (Page 2594)
Inserting operands into LAD instructions (Page 2601)
Basic information on the status word (Page 2453)
Setting the status bits in instructions with integers (Page 2458)
Setting the status bits in instructions with floating-point numbers (Page 2459)

COS: Form cosine value

Description

Use the "Form cosine value" instruction to calculate the cosine of the angle. The size of the angle is specified in radians at the IN input. The result of the instruction is sent to the OUT output and can be queried there.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The value at the IN input is not a valid floating-point number.

Parameters

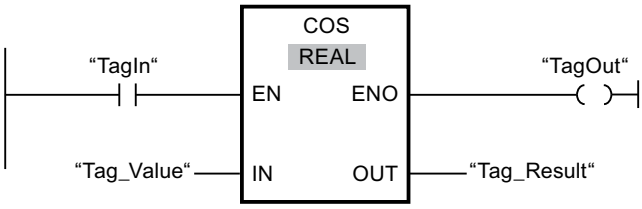
The following table shows the parameters of the "Form cosine value" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Size of angle in radians
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Cosine of the specified angle

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_Value	+1.570796 (π/2)
OUT	Tag_Result	0

If operand "TagIn" has the signal state "1", the "Form cosine value" instruction is executed. The instruction calculates the cosine of the angle specified at the "Tag_Value" input and stores the result in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

TAN: Form tangent value

Description

Use the "Form tangent value" instruction to calculate the tangent of the angle. The size of the angle is specified in radians at the IN input. The result of the instruction is sent to the OUT output and can be queried there.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The value at the IN input is not a valid floating-point number.

Parameter

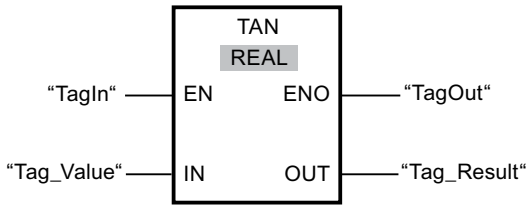
The following table shows the parameters of the "Form tangent value" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Size of angle in radians
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Tangent of the specified angle

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_Value	+3.141593 (π)
OUT	Tag_Result	0

If operand "TagIn" has the signal state "1", the "Form tangent value" instruction is executed. The instruction calculates the tangent of the angle specified at the "Tag_Value" input and stores the result in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

ASIN: Form arcsine value

Description

You can use the "Form arcsine value" instruction to calculate the size of the angle from the arcsine value specified at input IN, which corresponds to this value. Only valid floating-point numbers within the range -1 to +1 can be specified at the IN input. The calculated angle size is output in radians at the OUT output and can range in value from -π/2 to +π/2.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The value at the IN input is not a valid floating-point number.
- The value at the IN input is outside the permitted value range (-1 to +1).

Parameter

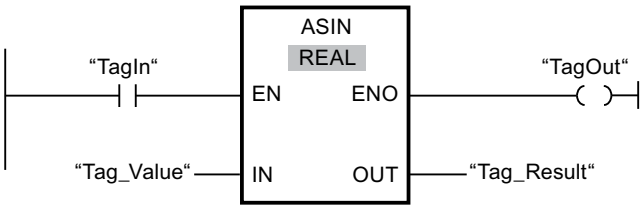
The following table shows the parameters of the "Form arcsine value" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Sine value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Size of angle in radians

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_Value	1.0
OUT	Tag_Result	+1.570796 (π/2)

If operand "TagIn" has the signal state "1", the "Form arcsine value" instruction is executed. The instruction calculates the size of the angle corresponding to the sine value at the "Tag_Value" input. The result of the instruction is stored in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

ACOS: Form arccosine value

Description

You can use the "Form arccosine value" instruction to calculate the size of the angle from the cosine value specified at input IN, which corresponds to this value. Only valid floating-point numbers within the range -1 to +1 can be specified at the IN input. The calculated angle size is output in radians at the OUT output and can range in value from 0 to +π.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The value at the IN input is not a valid floating-point number.
- The value at the IN input is outside the permitted value range (-1 to +1).

Parameter

The following table shows the parameters of the "Form arccosine value" instruction:

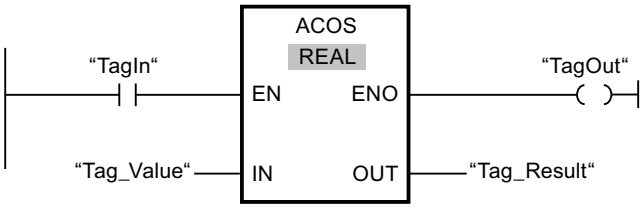
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

Parameter	Declaration	Data type	Memory area	Description
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Cosine value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Size of angle in radians

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_Value	0
OUT	Tag_Result	+1.570796 (π/2)

If operand "TagIn" has the signal state "1", the "Form arccosine value" instruction is executed. The instruction calculates the size of the angle corresponding to the cosine value at the "Tag_Value" input. The result of the instruction is stored in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

ATAN: Form arctangent value

Description

You can use the "Form arctangent value" instruction to calculate the size of the angle from the tangent value specified at input IN, which corresponds to this value. Only valid floating-point numbers may be specified at the IN input. The calculated angle size is output in radians at the OUT output and can range in value from -π/2 to +π/2.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN enable input has the signal state "0".
- The value at the IN input is not a valid floating-point number.

Parameter

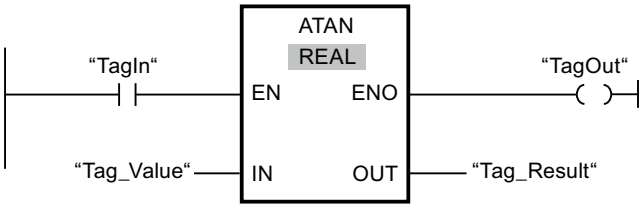
The following table shows the parameters of the "Form arctangent value" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	REAL	I, Q, M, D, L, P or constant	Tangent value
OUT	Output	REAL	I, Q, M, D, L, P	Size of angle in radians

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_Value	1.0
OUT	Tag_Result	+0.785398 (π/4)

If operand "TagIn" has the signal state "1", the "Form arctangent value" instruction is executed. The instruction calculates the size of the angle corresponding to the tangent value at the

"Tag_Value" input. The result of the instruction is stored in the "Tag_Result" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

Move operations

MOVE: Move value

Description

You use the "Move value" instruction to transfer the contents of the operand at the IN input to the operand at the OUT1 output. The transfer is always made in the direction of the ascending address.

The following table shows the possible transfers:

Source (IN)	Destination (OUT1)	
	With IEC check	Without IEC check
BYTE	BYTE, WORD, DWORD	BYTE, WORD, DWORD, INT, DINT, TIME, DATE, TOD, CHAR
WORD	WORD; DWORD	BYTE, WORD, DWORD, INT, DINT, TIME, S5TIME, DATE, TOD, CHAR
DWORD	DWORD	BYTE, WORD, DWORD, INT, DINT, REAL, TIME, DATE, TOD, CHAR
INT	INT	BYTE, WORD, DWORD, INT, DINT, TIME, DATE, TOD
DINT	DINT	BYTE, WORD, DWORD, INT, DINT, TIME, DATE, TOD
REAL	REAL	DWORD, REAL
TIME	TIME	BYTE, WORD, DWORD, INT, DINT, TIME
S5TIME	S5TIME	WORD, S5TIME
DATE	DATE	BYTE, WORD, DWORD, INT, DINT, DATE

Source (IN)	Destination (OUT1)	
	With IEC check	Without IEC check
TOD	TOD	BYTE, WORD, DWORD, INT, DINT, TOD
CHAR	CHAR	BYTE, WORD, DWORD, CHAR
COUNTER	INT, WORD, COUNTER	WORD, DWORD, INT, UINT, DINT, UDINT
TIMER	INT, WORD, TIMER	WORD, DWORD, INT, UINT, DINT, UDINT

If the bit length of the data type at input IN exceeds the bit length of the data type at output OUT1, the higher-order bits of the source value are lost. If the bit length of the data type at the IN input is less than the bit length of the data type at the OUT1 output, then the more significant bits of the destination value will be filled with zeros.

You can also use the "Move block" (BLKMOV) and "Move block uninterruptible" (UBLKMOV) instructions to move fields and structures.

Parameters

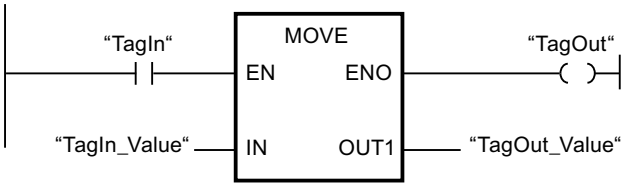
The following table shows the parameters of the "Move value" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers, floating-point numbers, timers, DATE, TOD, CHAR, TIMER, COUNTER	I, Q, M, D, L or constant	Source value
OUT1	Output	Bit strings, integers, floating-point numbers, timers, DATE, TOD, CHAR, TIMER, COUNTER	I, Q, M, D, L	Destination address

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
OUT1	TagOut_Value	0011 1111 1010 1111

If operand "TagIn" has the signal state "1", the "Move value" instruction is executed. The instruction copies the contents of operand "TagIn_Value" to operand "TagOut_Value" and sets output "TagOut" to signal state "1".

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Master control relay (Page 2471)

Important notes on using MCR functions (Page 2472)

---(MCR<): Open MCR ranges (Page 5529)

BLKMOV: Move block

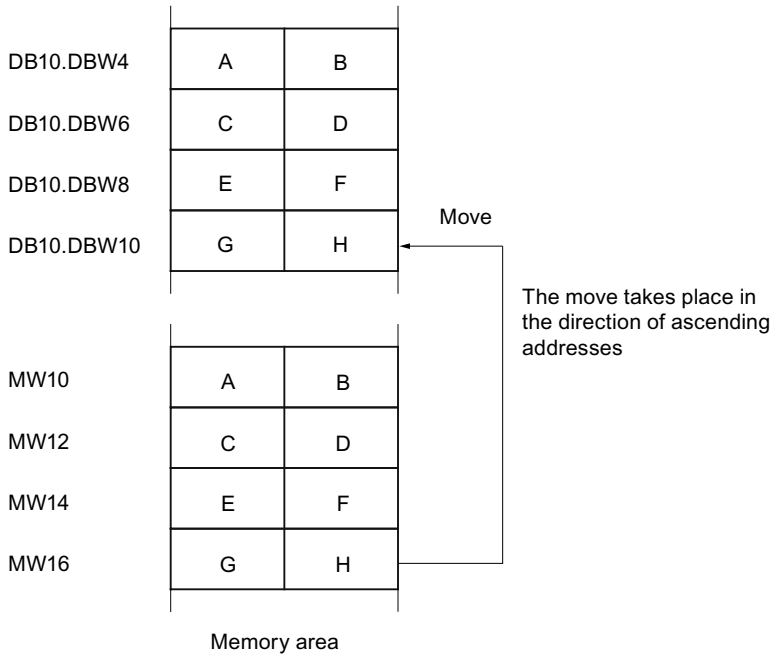
Description

You can use the "Move block" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use ANY pointer to define the source and destination area.

Note

In this instruction, you can only use data types which are valid on a CPU of the S7-300/400 series.

The following figure shows the principle of the move operation:



Consistency of the source data and the target data

Make sure that the source data remain unchanged during execution of the Move block instruction. Otherwise, consistency of the destination data cannot be guaranteed.

Interruptibility

As long as the source area is not part of a data block that only exists in the load memory, there is no limit to the nesting depth.

If, however, BLKMOV is interrupted while copying from a DB that is not runtime-relevant, the execution of BLKMOV can no longer be nested.

Memory areas

You can use the "Move block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output
- Data blocks not runtime-relevant

General rules for moving

The source and destination area must not overlap. If the source and destination area have different lengths, only the length of the smaller area will be moved.

If the source area is less than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is less than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a block of data type BOOL is moved, the specified length of the area must be dividable by 8, otherwise it will not be possible to execute the instruction.

Note

If an actual source or destination area is smaller than a destination or source area specified on the SRCBLK or DSTBLK parameter, the system behavior depends on which CPU type is being used:

- The following applies to S7-300 CPUs:
  No data are transferred. Error code W#16#837F is output at the RET_VAL parameter. This error code has to be evaluated in your user program.
- The following applies to the S7-400 V4-CPUs (V4, V4 H-CPU and V4.5. H-CPU):
  No data are transferred. Error code W#16#8122 or W#16#8323 is output at the RET_VAL parameter. This error code has to be evaluated in your user program.
- For all other S7-400-CPUs:
  Data are transferred. Error code W#16#8122 or W#16#8323 is output at the RET_VAL parameter. This error code has to be evaluated in your user program.

Rules for moving character strings

You can use the "Move block" instruction to also move source and destination areas of the STRING data type. If only the source area is STRING data type, the characters will be moved that are actually contained in the character string. Information on the actual and maximum length is also written to the destination area. If the source and destination area are each STRING data type, the current length of the character string in the destination area is set to the number of actually moved characters.

If you want to move the information on the maximum and actual length of a character string, specify the areas in bytes to the SRCBLK and DSTBLK parameters.

Rules for moving data blocks that are not runtime-relevant

The source area is in a data block in load memory that is not runtime-relevant. Data blocks that are not runtime-relevant are indicated by the keyword UNLINKED.

If a non-runtime-relevant data block is copied to the work memory with the "Move block" instruction and loaded at the same time, for example, by the programming device, the execution of the instruction can be delayed for several milliseconds. This results in a longer OB cycle and may trip the cycle monitoring.

If a non-runtime-relevant data block is moved with the "Move block" instruction and the move operation is interrupted, the execution of the instruction can no longer be continued.

If the Read from data block in the load memory instruction is available on your CPU, you must use this instruction to read data blocks that are not runtime-relevant from load memory. If you use the Move block instruction, error W#16#8092 is output.

Parameters

The following table shows the parameters of the "Move block" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRCBLK	Input	ANY	I, Q, M, D, L, P	Specifies the memory area to be moved (source area).
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.
DSTBLK	Output	ANY	I, Q, M, D, L, P	Specifies the memory area to which the block is to be moved (destination area).

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8092	The source or target area is only available in the load memory.
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Evaluating errors with output parameter RET_VAL (Page 5370)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Switching display formats in the program status (Page 2980)

UBLKMOV: Move block uninterruptible

Description

You can use the "Move block uninterruptible" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use ANY pointer to define the source and destination area.

The copy operation cannot be interrupted by other operating system activities. As a result the alarm reaction time of the CPU can increase during the execution of the "Move block uninterruptible" instruction.

Note

In this instruction, you can only use data types which are valid on a CPU of the S7-300/400 series.

Memory areas

You can use the "Move block uninterruptible" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input table
- Process image output table

General rules for moving

The source and destination area must not overlap during the execution of the "Move block uninterruptible" instruction. If the source area is less than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is less than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a source or destination area defined as a formal parameter is less than a destination or source area specified on the SRCBLK or DSTBLK parameter, no data will be transferred.

If a block of data type BOOL is moved, the specified length of the area must be dividable by 8, otherwise it will not be possible to execute the instruction.

You can use the "Move block uninterruptible" instruction to move a maximum of 512 bytes. Note the CPU specific restrictions for this.

Note

If an actual source or destination area is smaller than a destination or source area specified on the SRCBLK or DSTBLK parameter, the system behavior depends on which CPU type is being used:

- The following applies to S7-300 CPUs:
  No data are transferred. The error code W#16#837F is output on the RET_VAL parameter. This error code has to be evaluated in your user program.
- The following applies to the S7-400 V4-CPUs (V4, V4 H-CPU and V4.5. H-CPU):
  No data are transferred. The error code W#16#8122 or W#16#8323 is output on the RET_VAL parameter. This error code has to be evaluated in your user program.
- For all other S7-400-CPUs:
  Data are transferred. The error code W#16#8122 or W#16#8323 is output on the RET_VAL parameter. This error code has to be evaluated in your user program.

Rules for moving character strings

You can use the "Move block uninterruptible" instruction to also move source and destination areas of the STRING data type. If only the source area is STRING data type, the characters will be moved that are actually contained in the character string. Information on the actual and maximum length are not written in the destination area. If the source and destination area are each STRING data type, the current length of the character string in the destination area is set to the number of actually moved characters. If areas of the data type STRING are moved, you have to specify "1" as area length.

Parameters

The following table shows the parameters of the "Move block uninterruptible" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRCBLK	Input	ANY	I, Q, M, D, L, P	Specifies the memory area to be moved (source area).
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output on the RET_VAL parameter.
DSTBLK	Output	ANY	I, Q, M, D, L, P	Specifies the memory area to which the block is to be moved (destination area).

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8091	The source area is in a data block that is not relevant for program execution.
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Evaluating errors with output parameter RET_VAL (Page 5370)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Switching display formats in the program status (Page 2980)

FILL: Fill block

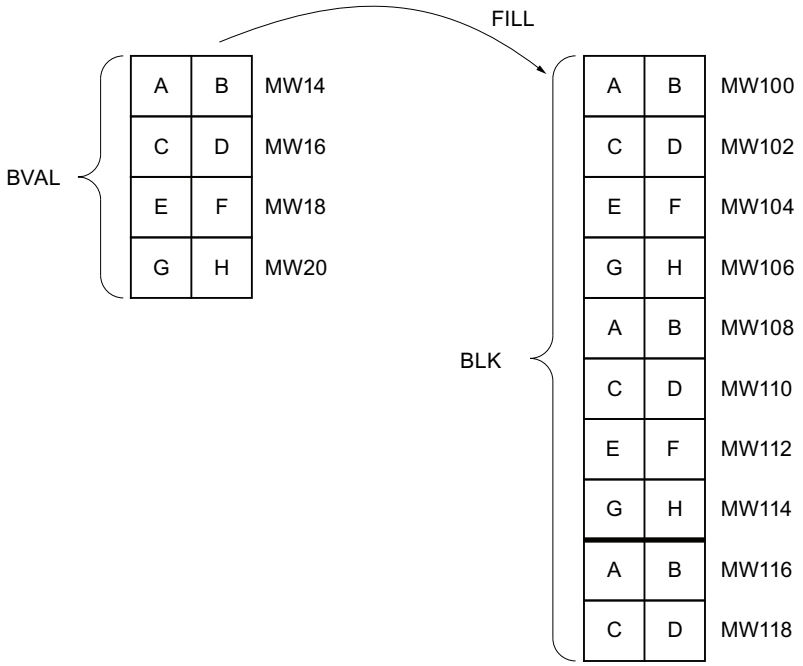
Description

You can use the "Fill block" instruction to fill a memory area (destination area) with the content of another memory area (source area). The "Fill block" instruction moves the content of the source area to the destination area until the destination area is completely written. The move operation takes place in the direction of ascending addresses. You use ANY pointer to define the source and destination area.

Note

In this instruction, you can only use data types which are valid on a CPU of the S7-300/400 series.

The following figure shows the principle of the move operation:



Example: The contents of the range MW100 to MW118 are to be preassigned with the contents of the memory words MW14 to MW20.

Consistency of the source data and the target data

Please note that during execution of the instruction "Fill block" that the source data remain unchanged, otherwise the consistency of the target data is not ensured.

Memory areas

You can use the "Fill block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input table
- Process image output table
- Data blocks not relevant for program execution

General rules for moving

The source and destination area must not overlap. If the destination block to be preset is not an integer multiple of the length of the input parameter BVAL, the destination block is nevertheless written up to the last byte.

If the destination area to be preset is smaller than the source area, the function only copies as much data as can be written to the destination area.

If the destination or source block actually present is smaller than the assigned memory area for the source or destination block (BVAL BLK parameters), no data will be transferred.

If the ANY pointer (source or destination) is of the data type BOOL, the length specified must be divisible by 8; otherwise the instruction will not be executed.

If the destination block is of data type STRING, the instruction describes the entire string including the administration information.

Note

If an actual source or destination area is smaller than a destination or source area specified on the BVAL or BLK parameter, the system behavior depends on which CPU type is being used:

- The following applies to S7-300 CPUs:
  No data are transferred. The error code W#16#837F is output on the RET_VAL parameter. This error code has to be evaluated in your user program.
- The following applies to the S7-400 V4-CPUs (V4, V4 H-CPU and V4.5. H-CPU):
  No data are transferred. The error code W#16#8122 or W#16#8323 is output on the RET_VAL parameter. This error code has to be evaluated in your user program.
- For all other S7-400-CPUs:
  Data are transferred. The error code W#16#8122 or W#16#8323 is output on the RET_VAL parameter. This error code has to be evaluated in your user program.

Parameters

The following table shows the parameters of the "Fill block" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
BVAL	Input	ANY	I, Q, M, D, L	Specification of the memory area (source area) with whose content the destination area on the BLK parameter will be filled.
RET_VAL	Output	INT	I, Q, M, D, L	Error information: If an error occurs during execution of the instruction, an error code is output on the RET_VAL parameter.
BLK	Output	ANY	I, Q, M, D, L	Specification of the memory area that will be filled with the content of the source area.

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Evaluating errors with output parameter RET_VAL (Page 5370)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Switching display formats in the program status (Page 2980)

Conversion operations

CONVERT: Convert value

Description

The "Convert value" instruction reads the content of the IN parameter and converts it according to the data types selected in the instruction box. The converted value is provided at output OUT.

For information on possible conversions, refer to the "Explicit conversion" section at "See also".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the "Convert value" instruction:

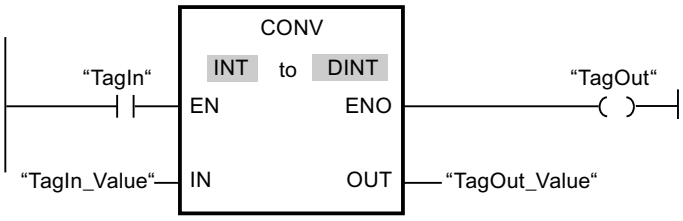
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Integers, floating-point numbers, BCD16, BCD32	I, Q, M, D, L, P or constant	Value to be converted.
OUT	Output	Integers, floating-point numbers, BCD16, BCD32	I, Q, M, D, L, P	Result of the conversion

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state "1", the content of the operand "TagIn_Value" is read and converted to an integer (32 bits). The result is stored in the operand "TagOut_Value". The output "TagOut" is set to "1" if the instruction was executed without errors.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

Explicit conversion (Page 2430)

ROUND: Round numerical value

Description

You can use the "Round numerical value" instruction to round the value at input IN to the nearest integer. The instruction interprets the value at input IN as a floating-point number and converts this to an integer of data type DINT. If the input value is exactly between an even and odd number, the even number is selected. The result of the instruction is sent to the OUT output and can be queried there.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The enable input EN has the signal state "0".
- Errors such as an overflow occur during execution.

Parameters

The following table shows the parameters of the "Round numerical value" instruction:

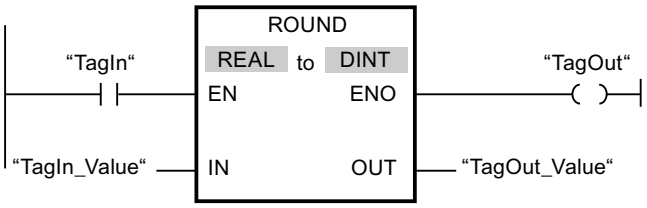
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	REAL	I, Q, M, D, L, P or constant	Input value to be rounded.
OUT	Output	DINT	I, Q, M, D, L, P	Result of rounding

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
IN	TagIn_Value	1.50000000	-1.50000000
OUT	TagOut_Value	2	-2

If operand "TagIn" has the signal state "1", the "Round numerical value" instruction is executed. The floating-point number at input "TagIn_Value" is rounded to the nearest even integer and

sent to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Setting the status bits in instructions with floating-point numbers (Page 2459)

Basic information on the status word (Page 2453)

CEIL: Generate next higher integer from floating-point number

Description

You can use the "Generate next higher integer from floating-point number" instruction to round the value at input IN to the next higher integer. The instruction interprets the value at the IN input as a floating-point number and converts this number to the next higher integer. The result of the instruction is sent to the OUT output and can be queried there. The output value can be greater than or equal to the input value.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The enable input EN has the signal state "0".
- Errors such as an overflow occur during execution.

Parameters

The following table shows the parameters of the "Generate next higher integer from floating-point number" instruction:

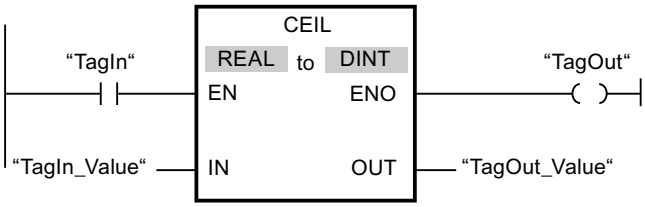
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result with next higher integer

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
IN	TagIn_Value	0.50000000	-0.50000000
OUT	TagOut_Value	1	0

If operand "TagIn" has the signal state "1", the "Generate next higher integer from floating-point number" instruction is executed. The floating-point number at the "TagIn_Value" input is rounded to the next higher integer and sent to the "TagOut_Value" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Setting the status bits in instructions with floating-point numbers (Page 2459)

Basic information on the status word (Page 2453)

FLOOR: Generate next lower integer from floating-point number

Description

You can use the "Generate next lower integer from floating-point number" instruction to round the value at input IN to the next lower integer. The instruction interprets the value at input IN as a floating-point number and converts this to the next lower integer. The result of the instruction is sent to the OUT output and can be queried there. The output value can be less than or equal to the input value.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The enable input EN has the signal state "0".
- Errors such as an overflow occur during execution.

Parameters

The following table shows the parameters of the "Generate next lower integer from floating-point number" instruction:

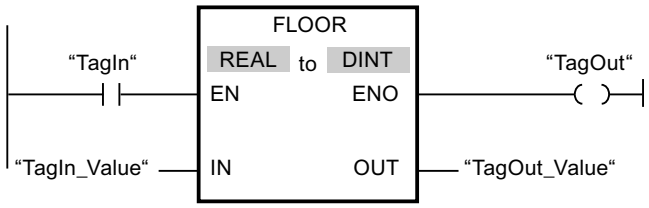
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result with next lower integer

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
IN	TagIn_Value	0.50000000	-0.50000000
OUT	TagOut_Value	0	-1

If operand "TagIn" has the signal state "1", the "Generate next lower integer from floating-point number" instruction is executed. The floating-point number at input "TagIn_Value" is rounded to the next lower integer and sent to output "TagOut_Value". If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Setting the status bits in instructions with floating-point numbers (Page 2459)

Basic information on the status word (Page 2453)

TRUNC: Truncate numerical value

Description

You can use the "Truncate numerical value" instruction to form an integer from the value at the IN input. The value at the IN input is interpreted as a floating-point number. The instruction selects only the integer part of the floating-point number and sends this to the OUT output without decimal places.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- The EN input has the signal state "0".
- Errors such as an overflow occur during execution.

Parameters

The following table shows the parameters of the "Truncate numerical value" instruction:

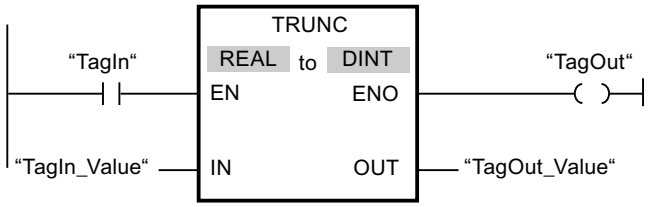
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	REAL	I, Q, M, D, L, P or constant	Input value
OUT	Output	Integers	I, Q, M, D, L	Integer part of the input value

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
IN	TagIn_Value	1.50000000	-1.50000000
OUT	TagOut_Value	1	-1

If operand "TagIn" has the signal state "1", the "Truncate numerical value" instruction is executed. The integer part of the floating-point number at the "TagIn_Value" input is converted to an integer and sent to the "TagOut_Value" output. If the instruction is executed without errors, the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Setting the status bits in instructions with floating-point numbers (Page 2459)

Basic information on the status word (Page 2453)

SCALE: Scale

Description

Use the "Scale" instruction to convert the integer at the IN parameter into a floating-point number, which can be scaled in physical units between a low limit value and a high limit value. You use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is scaled. The result of the instruction is output at the OUT parameter.

The "Scale" instruction works with the following equation:

OUT = [((FLOAT (IN) – K1)/(K2–K1)) * (HI_LIM–LO_LIM)] + LO_LIM

The values of the constants "K1" and "K2" are determined by the signal state of the BIPOLAR parameter. The following signal states are possible at the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case the "K1" constant has the value "-27648.0" and the "K2" constant the value "+27648.0".
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case the "K1" constant has the value "0.0" and the "K2" constant the value "+27648.0".

When the value at the IN parameter is greater than the value of the constant "K2", the result of the instruction is set to the value of the high limit (HI_LIM) and an error is output.

When the value at the IN parameter is less than the value of the constant "K1", the result of the instruction is set to the value of the low limit value (LO_LIM) and an error is output.

When the indicated low limit value is greater than the high limit value (LO_LIM > HI_LIM), the result is scaled in reverse proportion to the input value.

Parameters

The following table shows the parameters of the "Scale" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	INT	I, Q, M, D, L, P or constant	Input value to be scaled
HI_LIM	Input	REAL	I, Q, M, D, L, P or constant	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P or constant	Low limit
BIPOLAR	Input	BOOL	I, Q, M, D, L or constant	Indicates if the value at the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
OUT	Output	REAL	I, Q, M, D, L, P	Result of the instruction
RET_VAL	Output	WORD	I, Q, M, D, L, P	Error information

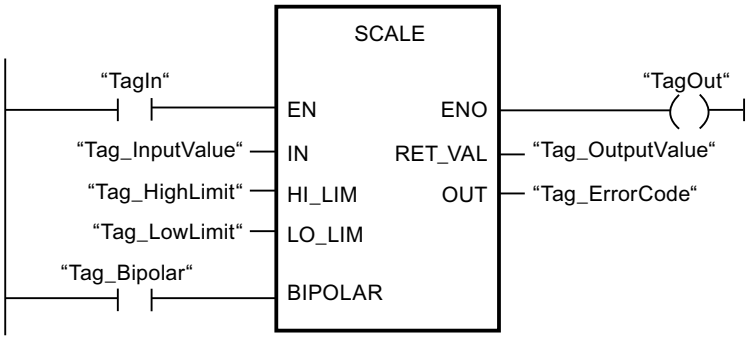
Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the constant "K2" or less than the value of the constant "K1"
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_InputValue	22
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
OUT	Tag_OutputValue	50.03978588
RET_VAL	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 2236)

Evaluating errors with output parameter RET_VAL (Page 5370)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Switching display formats in the program status (Page 2980)

UNSCALE: Unscale

Description

The "Unscale" instruction is used to unscale the floating-point number on the IN parameter into physical units between a low limit and a high limit and convert it into an integer. You use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is unscaled. The result of the instruction is output at the OUT parameter.

The "Unscale" instruction works with the following equation:

OUT = [((IN–LO_LIM)/(HI_LIM–LO_LIM)) * (K2–K1)] + K1

The values of the constants "K1" and "K2" are determined by the signal state of the BIPOLAR parameter. The following signal states are possible at the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case the "K1" constant has the value "-27648.0" and the "K2" constant the value "+27648.0".
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case the "K1" constant has the value "0.0" and the "K2" constant the value "+27648.0".

When the value at the IN parameter is greater than the value of the constant "HI_LIM", the result of the instruction is set to the value of the constant (K2) and an error is output.

When the value at the IN parameter is less than the value of the constant of the low limit (LO_LIM), the result of the instruction is set to the value of the constant (K1) and an error is output.

Parameters

The following table shows the parameters of the "Unscale" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Input	BOOL	I, Q, M, D, L	Enable output
IN	Input	REAL	I, Q, M, D, L, P or constant	Input value to be unscaled to an integer value
HI_LIM	Input	REAL	I, Q, M, D, L, P or constant	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P or constant	Low limit

Parameter	Declaration	Data type	Memory area	Description
BIPOLAR	Input	BOOL	I, Q, M, D, L or constant	Indicates if the value at the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
OUT	Output	INT	I, Q, M, D, L, P	Result of the instruction
RET_VAL	Output	WORD	I, Q, M, D, L, P	Error information

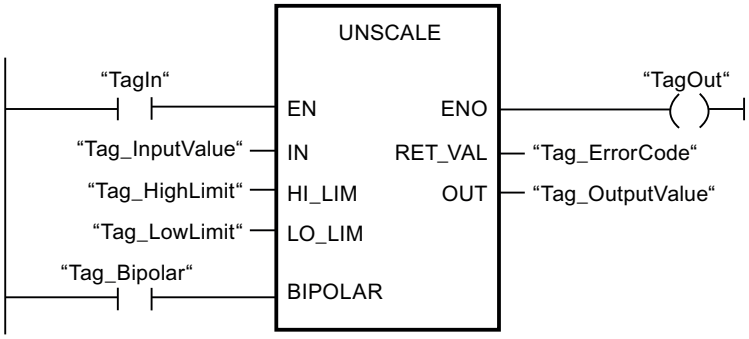
Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the high limit (HI_LIM) or less than the value of the low limit (LO_LIM).
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0

Parameter	Operand	Value
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
OUT	Tag_OutputValue	22
RET_VAL	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 2236)

Evaluating errors with output parameter RET_VAL (Page 5370)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Switching display formats in the program status (Page 2980)

Program control operations

---(JMP): Jump if RLO = 1

Description

You can use the "Jump if RLO = 1" instruction to interrupt the linear execution of the program and resume it in another network. The destination network must be identified by a jump label (LABEL). The name of this jump label is specified in the placeholder above the instruction.

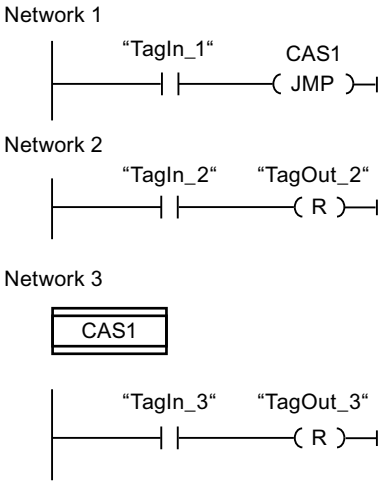
The specified jump label must be in the same block in which the instruction is executed. The name you specify can only occur once in the block. Only one jumping coil is permitted within a network.

If the result of logic operation (RLO) at the input of the instruction is "1", the jump to the network identified by the specified jump label is executed. The jump direction can be towards higher or lower network numbers.

If the condition at the input of the instruction is not fulfilled (RLO = 0), execution of the program continues in the next network.

Example

The following example shows how the instruction works:



If operand "TagIn_1" has the signal state "1", the "Jump if RLO = 1" instruction is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. If the "TagIn_3" input has the signal state "1", the "TagOut_3" output is set.

See also

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

---(JMPN): Jump if RLO = 0

Description

You can use the instruction "Jump if RLO = 0" to interrupt the linear execution of the program and resume it in another network, when the result of logic operation at the input of the instruction is "0". The destination network must be identified by a jump label (LABEL). The name of this jump label is specified in the placeholder above the instruction.

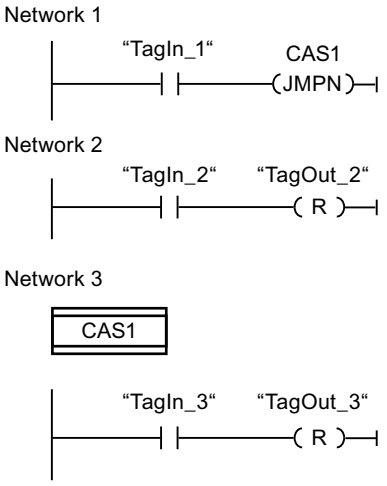
The specified jump label must be in the same block in which the instruction is executed. The name you specify can only occur once in the block. Only one jumping coil is permitted within a network.

If the result of logic operation (RLO) at the input of the instruction is "0", the jump to the network identified by the specified jump label is executed. The jump direction can be towards higher or lower network numbers.

If the result of logic operation at the input of the instruction is "1", execution of the program continues in the next network.

Example

The following example shows how the instruction works:



If operand "TagIn_1" has the signal state "0", the "Jump if RLO = 0" instruction is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. If the "TagIn_3" input has the signal state "1", the "TagOut_3" output is set.

See also

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

LABEL: Jump label

Description

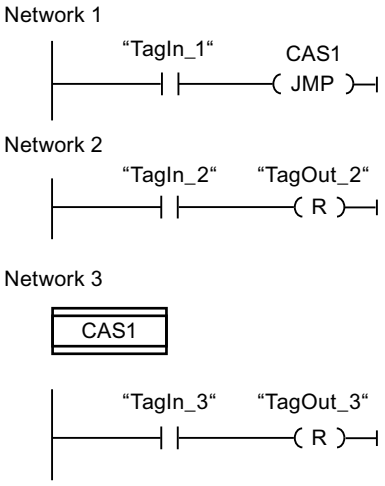
You can use a jump label to identify a destination network, in which the program execution should resume when a jump is executed.

The jump label and the instruction in which the jump label is specified must be located in the same block. The name of a jump label can only be assigned once in a block.

Only one jump label can be placed in a network. Each jump label can jump to several locations.

Example

The following example shows how the instruction works:



If operand "TagIn_1" has the signal state "1", the "Jump if RLO = 1" instruction is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. If the "TagIn_3" input has the signal state "1", the "TagOut_3" output is set.

See also

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

--(RET): Return

Description

You can use the "Return" instruction to stop the execution of a block.

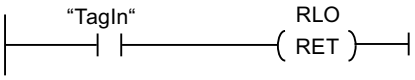
If the result of logic operation (RLO) at the input of the "Return" instruction is "1", program execution is terminated in the currently called block and resumed after the call function in the calling block (for example, in the calling OB). The status (ENO) of the call function is determined by the RLO. The RLO is displayed as a fixed parameter value above the instruction. When the RLO at the input of the "Return" instruction is "0", the instruction is not executed. In this case, the execution of the program continues in the next network of the called block.

If the "Return" instruction is programmed in the last network of the called block, the ENO output of the call function is set to the current signal state of the RLO.

If an organization block is terminated by the "Return" instruction, the CPU goes on working in the system program.

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state "1", the "Return" instruction is executed. Program execution is terminated in the called block and continues in the calling block. The ENO enable output of the call function is set to signal state "1".

See also

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

---(OPN): Open global data block

Description

You can use the "Open global data block" instruction to open a global data block (DB). The number of the data block is transferred to the DB register. Subsequent DB commands access the relevant blocks depending on the content of the register.

Parameters

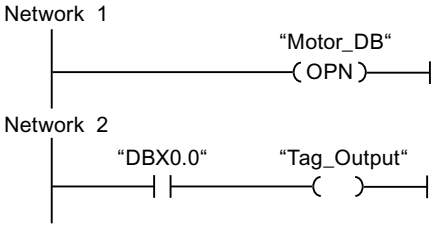
The following table shows the parameters of the "Open global data block" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Data block>	Input	DB_ANY	DB	Data block to be opened.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "Motor_DB" data block is called in network 1. The number of the data block is transferred to the DB register. The "DBX0.0" operand is queried in network 2. The signal state of the DBX0.0 operand is assigned to the Tag_Output operand.

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

---(OPNI): Open instance data block

Description

You can use the "Open instance data block" instruction to open a data block (DB). The number of the data block is transferred to the DI register. Subsequent DI commands access the relevant blocks depending on the content of the register.

Parameter

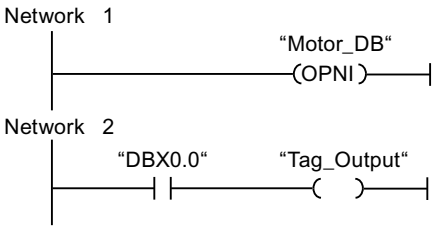
The following table shows the parameters of the "Open instance data block" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Data block>	Input	DB_ANY	DB	Data block to be opened

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The "Motor_DB" data block is called in network 1. The number of the data block is transferred to the DI register. The "DBX0.0" operand is queried in network 2. The signal state of the DBX0.0 operand is assigned to the Tag_Output operand.

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Runtime control

COMPRESS: Compress CPU memory

Description

The "Compress CPU memory" instruction is used to compress the working memory of the CPU or the RAM work memory. During this compression the gaps caused by deleting or loading blocks are filled by pushing together the existing blocks.

The compression operation is started by calling the "Compress CPU memory" instruction. The compression operation is divided into several program cycles. The compression operation is monitored by the values of the individual parameters. If value "1" is on the BUSY parameter of the instruction, the compression operation is still being executed. The value "1" on the DONE parameter reports that the compression operation has been completed. Error information can be queried on the RET_VAL parameter.

If an externally started compression operation has already been executed, the calling of the "Compress CPU memory" instruction causes an error to be displayed.

Parameter

The following table shows the parameters of the "Compress CPU memory" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
RET_VAL	Output	INT	I, Q, M, D, L	Error information: If an error occurs during execution of the instruction, an error code is output on the RET_VAL parameter.
BUSY	Output	BOOL	I, Q, M, D, L	Information whether compression operation is active.
DONE	Output	BOOL	I, Q, M, D, L	Information whether compression operation was completed.

For additional information on valid data types, refer to "See also".

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error The status of the compression operation can be queried on the BUSY and DONE parameters.
8091	An externally started compression operation is already being executed.
8092	The operation cannot be executed, because one of the following conditions applies: • The "Delete block" function is active. • A test and commissioning function affects a block (for example, status). • The "copy blocks" function was triggered externally and is active. • The H-CPU is running a link-up or update function.
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Evaluating errors with output parameter RET_VAL (Page 5370)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Basics of the EN/ENO mechanism (Page 2447)

Switching display formats in the program status (Page 2980)

RE_TRIGR: Restart cycle monitoring time

Description

You can use the "Restart cycle monitoring time" instruction to restart the cycle monitoring of the CPU. The maximum cycle time then starts over again for the duration you have set in the CPU configuration. By restarting the maximum cycle time, you can prevent errors being triggered or the CPU changing to STOP.

The "Restart cycle monitoring time" instruction can be used in blocks of priority class 1 (the cyclic OB) and in the blocks called within it.

If the instruction is called in a block with a higher priority, such as a hardware interrupt, diagnostic error interrupt, or cyclic interrupt, the instruction is not executed and the ENO enable output is set to signal state "0".

Note

Make sure that you do not create an infinite loop in the cyclical program processing, i.e. in OB1, when you use the instruction "Restart cycle monitoring time". Otherwise the CPU will not reach the cycle control point. As a result, it may not be possible to execute certain CPU functions (e.g. process image update).

Parameter

The "Restart cycle monitoring time" instruction has no parameters.

See also

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

STP: Exit program

Description

You use the "Exit program" instruction to set the CPU to STOP mode and therefore to terminate the execution of the program. The effects of changing from RUN to STOP depend on the CPU configuration.

When the result of logic operation (RLO) at the input of the instruction is "1", the CPU changes to STOP mode and program execution is terminated. The signal state at the output of the instruction is not evaluated.

When the RLO is "0" at the input of the instruction, then the instruction will not be executed.

Parameters

The "Exit program" instruction has no parameters.

See also

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

WAIT: Configure time delay

Description

The "Configure time delay" instruction is used to halt the execution of the program for a set period of time. You indicate the period of time in microseconds on the WT parameter of the instruction.

You can configure time delays of up to 32767 microseconds (µs). The smallest possible delay time depends on the respective CPU and corresponds to the execution time of the "Configure time delay" instruction.

The execution of the instruction can be interrupted by higher priority events.

The "Configure time delay" instruction supplies no error information.

Parameter

The following table shows the parameters of the "Configure time delay" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
WT	Input	INT	I, Q, M, D, L, P or constant	Time delay in microseconds (µs)

See also

Overview of the valid data types (Page 2236)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Basics of the EN/ENO mechanism (Page 2447)

PROTECT: Change protection level

Description

The instruction "Change protection level" is used to change the configured protection level "1" of the CPU.

The configured protection level "1" with selected option "Removable with password", means that you have to configure a password. This password enables you to write-access during operation the CPU which was read-only protected with the instruction "Change Protection Level" (e. g. Load).

You specify the protection level on the MODE parameter of the instruction. The current protection level of your CPU is displayed in the inspector window under "Properties > Protection".

The following table shows the meaning of the protection levels.

Protection level	Meaning
1	All programming device functions are permitted. Any existing blocking of the password authentication is canceled by the instruction "Change protection level".
2	• The program and the configuration of the CPU cannot be changed. The program in the CPU can be read. • The functions for process control, process monitoring, and process communication are permitted. • All informational functions are permitted. • If you know the valid password you can cancel the set write protection. • Any existing blocking of the password authentication is canceled by the instruction "Change protection level".
3	• You cannot cancel the set read and write protection, even if you know the valid password. • If a valid connection exists at the time when the instruction "Change protection level" is called, the call of the instruction does not affect this connection.

If you have configured protection level "2" or "3" for your CPU, calling the "Change protection level" instruction has no effect.

You can use the SSL parts list with the SSL ID W#16#0232 and index W#16#0004 to check whether protection level "2" or "3" is still in effect after the execution of the "Change protection level" instruction.

Note

You can only use the "Change protection level" instruction to set a protection level that is equal to or higher than the protection level you have configured with STEP 7 "Configure hardware. You have, for example, configured protection level 3 and then called the instruction "Change protection level" with the MODE parameter = 12. This blocks the password authentication. If the instruction is called again with the parameter MODE = 0 or 1, protection level 3 is retained and the read and write protection canceled by entry of the valid password.

Effects of the operating state transitions on the protection level

The following table shows what effects the operating state transitions have on the protection level set with the "Change protection level" instruction.

Action	Effects on the protection level
• Operating mode switch to "STOP" • Unbuffered POWER ON • Memory reset with operating mode switch • Configuration and/or programming changes in the "STOP" mode of the S7-400 CPU	The protection level is set to "1". All programming device functions are permitted.
• Cold restart • Warm restart	The protection level is set by the operating system to "1". All programming device functions are permitted. If required, you can set the protection level in your program to "2" or "3" by calling the "Change protection level" instruction.
Buffered POWER ON	Remains unchanged
Operating mode change RUN/STARTUP/HOLD -> STOP (by calling the instruction "Exit program" (STP), due to a program error without corresponding error OB or by an operator action).	Remains unchanged
S7-400: Restart by user action or automatically following POWER ON	Remains unchanged

Parameter

The following table shows the parameters of the "Change protection level" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
MODE	Input	WORD	I, Q, M, D, L or constant	Protection level Possible values: • W#16#0000: Setting protection level "1" • W#16#0001: Setting protection level "2" with password authentication • W#16#000C: Setting protection level "3" with password authentication Constants can also be specified on the MODE parameter.
RET_VAL	Output	INT	I, Q, M, D, L	If an error occurs during processing, information on the error is output.

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	Job executed without error
8090	Invalid value at parameter MODE.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Evaluating errors with output parameter RET_VAL (Page 5370)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Basics of the EN/ENO mechanism (Page 2447)

Switching display formats in the program status (Page 2980)

CiR: Control CiR process

Description

The "Control CiR process" instruction is used to influence the CiR process in the user program as follows:

- Disable CiR process. In this case, the download of a modified configuration from the programming device to the CPU is always rejected. The disable is maintained until you cancel it again with the "Control CiR process" instruction.
- You can disable the CiR process conditionally by specifying a high limit for the CiR synchronization time on the FRZ_TIME parameter. In this case, the download of a modified configuration from the programming device to the CPU is only accepted if the CiR synchronization time calculated in the CPU does not exceed the specified high limit.
- Disable CiR process. The high limit of the CiR synchronization time is set to the default value 1000 ms.
- Determine whether the CiR process is enabled or not. If the CiR process is enabled or conditionally enabled, you will get the current high limit for the CiR synchronization time in the A_FT parameter of the instruction.

Note

During the CiR synchronization period the outputs are frozen and the inputs are not evaluated.

Parameters

The following table shows the parameters of the "Control CiR process" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
MODE	Input	BYTE	I, Q, M, D, L or constant	Job identifier Valid values on the MODE parameter are: • 0: Information function • 1: Enable CiR process • 2: Disable CiR process • 3: Disable CiR conditionally. You specify the high limit of the CiR synchronization time in FRZ_TIME.
FRZ_TIME	Input	TIME	I, Q, M, D, L or constant	High limit of the CiR synchronization time in milliseconds. Permitted range of values: 60 ... 2500 ms (default: 1000 ms) The FRZ_TIME parameter is only relevant when the value 3 is set on the MODE parameter.
RET_VAL	Output	INT	I, Q, M, D, L	Error information When the MODE parameter is "0", the RET_VAL parameter supplies information on the enabling of the CiR process.
A_FT	Output	TIME	I, Q, M, D, L	Currently valid high limit of the CiR synchronization time

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	Job executed without error This error code can be output only if the value "1", "2" or "3" is set on the MODE parameter.
0001	CiR is enabled. This error code can be output only if the value "0" is set on the MODE parameter.
0002	The CiR process is disabled. This error code can be output only if the value "0" is set on the MODE parameter.
0003	CiR is disabled conditionally. This error code can be output only if the value "0" is set on the MODE parameter.

Error code* (W#16#...)	Explanation
8001	The CPU is not ready for a CiR operation. You are using an H CPU in an H-system (stand-alone), or you are working with a standard CPU in multicomputing mode.
8002	The value at the MODE parameter is invalid.
8003	The value at the FRZ_TIME parameter is invalid.
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

Example

The "Control CiR process" instruction can be used, for example, to ensure that no CiR process is initiated during periods in which maximum CPU performance is required for the execution of the process.

To use the "Control CiR process" for this, the instruction must be called before the start of the period with increased process activity and executed with the value "2" (disable CiR process) on the MODE parameter. The CiR process is then disabled.

To enable the CiR process again after the completion of the period with increased process activity, you have to again call the "Control CiR process" instruction and execute it with the value "1" on the MODE parameter. Alternatively, you can also set the value 3 on the MODE parameter to disable the CiR process conditionally.

See also

Overview of the valid data types (Page 2236)

Evaluating errors with output parameter RET_VAL (Page 5370)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Basics of the EN/ENO mechanism (Page 2447)

Switching display formats in the program status (Page 2980)

Word logic operations

AND: AND logic operation

Description

You can use the "AND logic operation" instruction to combine the value at the IN1 input and the value at the IN2 input bit-by-bit by AND logic and query the result at the OUT output.

When the instruction is executed, bit 0 of the value at the IN1 input and bit 0 of the value at the IN2 input are logically ANDed. The result is stored in bit 0 of output OUT. The same logic operation is executed for all other bits of the specified values.

The result bit has signal state "1" only when both of the bits in the logic operation also have signal state "1". If one of the two bits of the logic operation has signal state "0", the corresponding result bit is reset.

Parameters

The following table shows the parameters of the "AND logic operation" instruction:

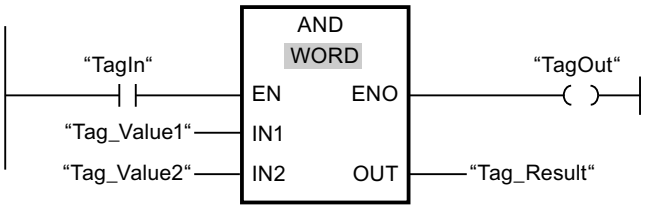
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	WORD, DWORD	I, Q, M, D, L, P or constant	First value for logic operation
IN2	Input	WORD, DWORD	I, Q, M, D, L, P or constant	Second value for logic operation
OUT	Output	WORD, DWORD	I, Q, M, D, L, P	Result of the instruction

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	Tag_Value1	0101 0101 0101 0101
IN2	Tag_Value2	0000 0000 0000 1111
OUT	Tag_Result	0000 0000 0000 0101

If operand "TagIn" has the signal state "1", the "AND logic operation" instruction is executed. The value of operand "Tag_Value1" and the value of operand "Tag_Value2" are ANDed. The result is mapped bit-for-bit and output in operand "Tag_Result". The ENO enable output and the "TagOut" output are set to signal state "1".

See also

Overview of the valid data types (Page 2236)
Basics of the EN/ENO mechanism (Page 2447)
Inserting LAD elements (Page 2579)
Selecting the data type of a LAD element (Page 2583)
Editing LAD elements (Page 2594)
Inserting operands into LAD instructions (Page 2601)
Setting the status bit for word logic operations (Page 2462)
Basic information on the status word (Page 2453)

OR: OR logic operation

Description

You can use the "OR logic operation" instruction to combine the value at the IN1 input and the value at the IN2 input bit-by-bit by OR logic and query the result at the OUT output.

When the instruction is executed, bit 0 of the value at the IN1 input and bit 0 of the value at the IN2 input are combined by OR logic operation. The result is stored in bit 0 of output OUT. The same logic operation is executed for all bits of the specified tags.

The result bit has signal state "1" when at least one of the two bits in the logic operation has signal state "1". If both of the bits of the logic operation have signal state "0", the corresponding result bit is reset.

Parameters

The following table shows the parameters of the "OR logic operation" instruction:

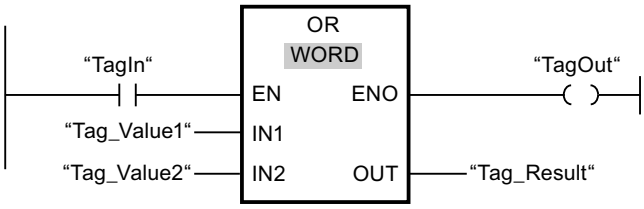
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	WORD, DWORD	I, Q, M, D, L, P or constant	First value for logic operation
IN2	Input	WORD, DWORD	I, Q, M, D, L, P or constant	Second value for logic operation
OUT	Output	WORD, DWORD	I, Q, M, D, L, P	Result of the instruction

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	Tag_Value1	0101 0101 0101 0101
IN2	Tag_Value2	0000 0000 0000 1111
OUT	Tag_Result	0101 0101 0101 1111

If operand "TagIn" has the signal state "1", the "OR logic operation" instruction is executed. The value of operand "Tag_Value1" and the value of operand "Tag_Value2" are ORed. The result is mapped bit-for-bit and output in operand "Tag_Result". The ENO enable output and the "TagOut" output are set to signal state "1".

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Setting the status bit for word logic operations (Page 2462)

Basic information on the status word (Page 2453)

XOR: EXCLUSIVE OR logic operation

Description

You can use the "EXCLUSIVE OR logic operation" instruction to combine the value at the IN1 input and the value at the IN2 input bit-by-bit by EXCLUSIVE OR logic and query the result at the OUT output.

When the instruction is executed, bit 0 of the value at the IN1 input and bit 0 of the value at the IN2 input are combined by EXCLUSIVE OR logic operation. The result is stored in bit 0 of output OUT. The same logic operation is executed for all other bits of the specified value.

The result bit has signal state "1" when one of the two bits in the logic operation has signal state "1". If both of the bits of the logic operation have signal state "1" or "0", the corresponding result bit is reset.

Parameters

The following table shows the parameters of the "EXCLUSIVE OR logic operation" instruction:

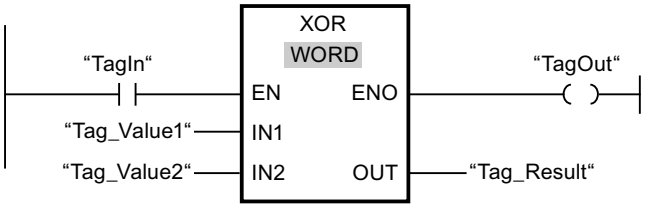
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	WORD, DWORD	I, Q, M, D, L, P or constant	First value for logic operation
IN2	Input	WORD, DWORD	I, Q, M, D, L, P or constant	Second value for logic operation
OUT	Output	WORD, DWORD	I, Q, M, D, L, P	Result of the instruction

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	Tag_Value1	0101 0101 0101 0101
IN2	Tag_Value2	0000 0000 0000 1111
OUT	Tag_Result	0101 0101 0101 1010

If operand "TagIn" has the signal state "1", the "EXCLUSIVE OR logic operation" instruction is executed. The value of operand "Tag_Value1" and the value of operand "Tag_Value2" are combined by EXCLUSIVE OR logic. The result is mapped bit-for-bit and output in operand "Tag_Result". The ENO enable output and the "TagOut" output are set to signal state "1".

See also

Overview of the valid data types (Page 2236)
Basics of the EN/ENO mechanism (Page 2447)
Inserting LAD elements (Page 2579)
Selecting the data type of a LAD element (Page 2583)
Editing LAD elements (Page 2594)
Inserting operands into LAD instructions (Page 2601)
Setting the status bit for word logic operations (Page 2462)
Basic information on the status word (Page 2453)

INV: Create ones complement

Description

You can use the "Create ones complement" instruction to invert the signal state of the bits at the IN input. When the instruction is processed, the value at the IN input and a hexadecimal template (W#16#FFFF for 16-bit numbers or DW#16#FFFFFFFF for 32-bit numbers) are combined by EXCLUSIVE OR logic. This inverts the signal state of the individual bits that are then stored at output OUT.

Parameter

The following table shows the parameters of the "Create ones complement" instruction:

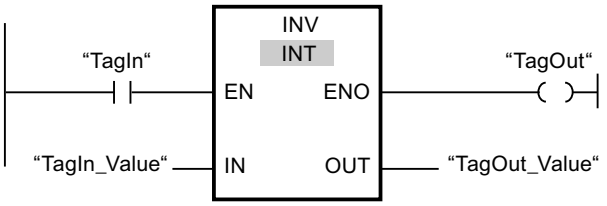
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Integers	I, Q, M, D, L, P or constant	Input value
OUT	Output	Integers	I, Q, M, D, L, P	Ones complement of the value at input IN

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	TagIn_Value	W#16#000F
OUT	TagOut_Value	W#16#FFF0

If operand "TagIn" has the signal state "1", the "Create ones complement" instruction is executed. The instruction inverts the signal state of the individual bits at input "TagIn_Value" and writes the result to output "TagOut_Value". The ENO enable output and the "TagOut" output are set to signal state "1".

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

DECO: Decode

Description

You use the "Decode" instruction to set a bit in the output value that is specified by the input value.

The "Decode" instruction reads the value at the IN input and sets the bit in the output value whose bit position corresponds to the read value. The other bits in the output value are filled with zeroes. When the value at input IN is greater than 31, a modulo 32 instruction is executed.

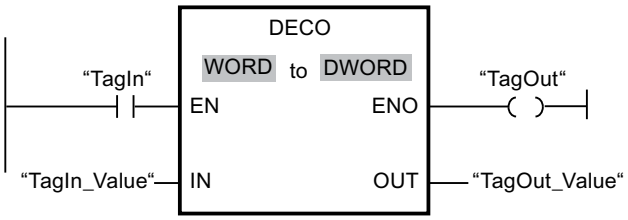
Parameters

The following table shows the parameters of the "Decode" instruction:

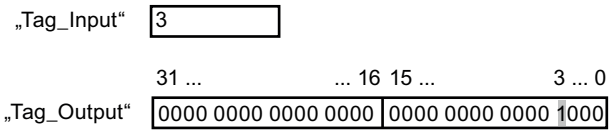
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	WORD	I, Q, M, D, L, P or constant	Position of the bit in the output value which is set.
OUT	Output	DWORD	I, Q, M, D, L, P	Output value

Example

The following example shows how the instruction works:



The following figure shows how the instruction works using specific operand values:



If operand "TagIn" has the signal state "1", the "Decode" instruction is executed. The instruction reads bit number "3" from the value of the operand "TagIn_Value" at the input and sets the third bit to the value of the operand "TagOut_Value" at the output.

If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

ENCO: Encode

Description

Use the "Encode" instruction to read the bit number of the least significant bit in the input value and to send it to the output OUT.

The "Encode" instruction selects the least significant bit of the value at the IN input and writes its bit number to the tag in the OUT output.

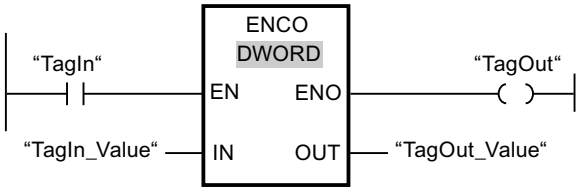
Parameters

The following table shows the parameters of the "Encode" instruction:

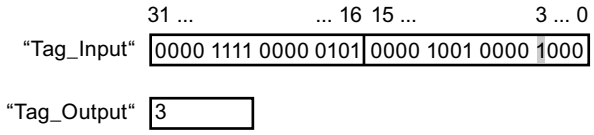
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	DWORD	I, Q, M, D, L, P or constant	Input value
OUT	Output	INT	I, Q, M, D, L, P	Output value

Example

The following example shows how the instruction works:



The following figure shows how the instruction works using specific operand values:



If operand "TagIn" has the signal state "1", the "Encode" instruction is executed. The instruction selects the least significant bit at the "TagIn_Value" input and writes bit position "3" to the tag in the "TagOut_Value" output.

If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

SEL: Select

Description

Depending on a switch (G input), the "Select" instruction selects one of the inputs, IN0 or IN1, and copies its content to the OUT output. When the G input has the signal state "0", the value at the IN0 input is moved. When the G input has the signal state "1", the value at the IN1 input is moved to the OUT output.

The instruction is only executed if the tags of all parameters are of the same data type.

Parameters

The following table shows the parameters of the "Select" instruction:

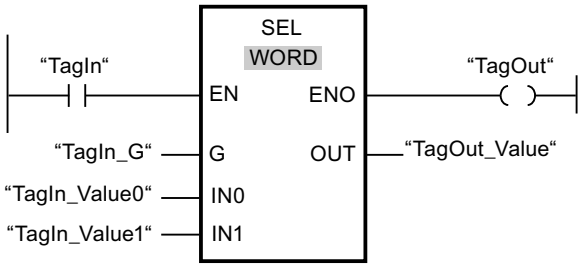
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
G	Input	BOOL	I, Q, M, D, L, T, C	Switch
IN0	Input	Bit strings, integers, floating-point numbers, timers, CHAR, TOD, DATE	I, Q, M, D, L, P	First input value
IN1	Input	Bit strings, integers, floating-point numbers, timers, CHAR, TOD, DATE	I, Q, M, D, L, P	Second input value
OUT	Output	Bit strings, integers, floating-point numbers, timers, CHAR, TOD, DATE	I, Q, M, D, L, P	Result

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
G	TagIn_G	0	1
IN0	TagIn_Value0	W#16#0000	W#16#4C
IN1	TagIn_Value1	W#16#FFFF	W#16#5E
OUT	TagOut_Value	W#16#0000	W#16#5E

If operand "TagIn" has the signal state "1", the "Select" instruction is executed. Based on the signal state at the "TagIn_G" input, the value at the "TagIn_Value0" or "TagIn_Value1" input is selected and copied to the "TagOut_Value" output. If the instruction is executed without errors, enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Shift and rotate

SHR: Shift right

Description

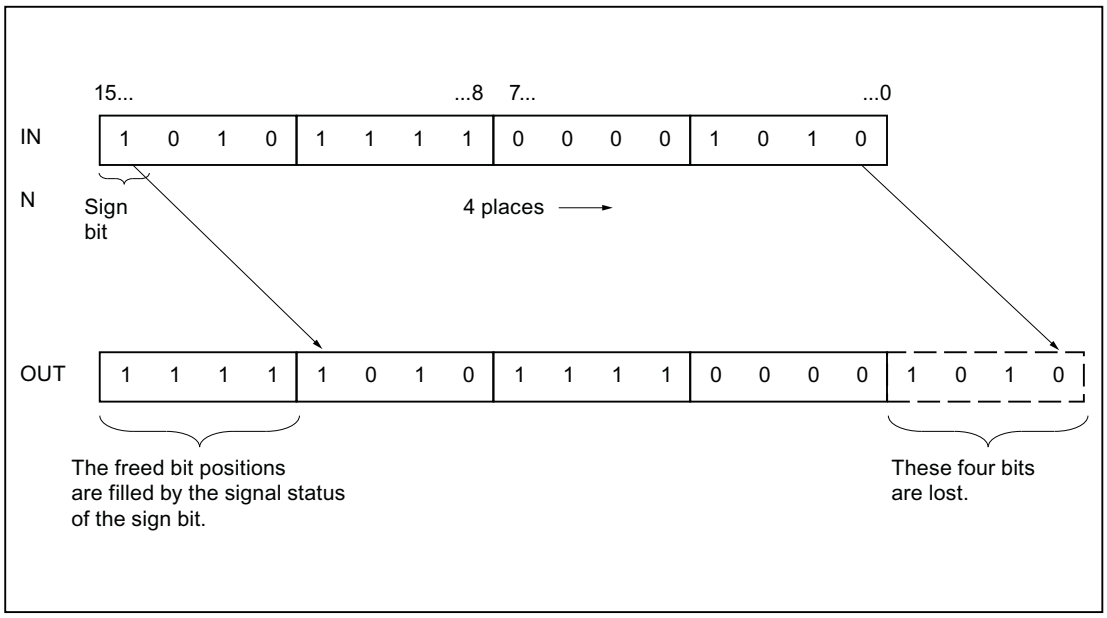
You can use the "Shift right" instruction to shift the content of the operand at the IN input bit-by-bit to the right and query the result at the OUT output. You use the N parameter to specify the number of bit positions by which the specified value is to be shifted.

If the value at the parameter N is "0", the value at input IN is copied to the operand at output OUT.

If the value at the parameter N is greater than the number of bit positions, the operand value at input IN is shifted to the right by the available number of bit positions.

The freed bit positions in the left area of the operand are filled by zeroes when values without signs are shifted. If the specified value has a sign, the free bit positions are filled with the signal state of the sign bit.

The following figure shows how the content of an operand of the INT data type is shifted by four bit positions to the right:



Parameters

The following table shows the parameters of the "Shift right" instruction:

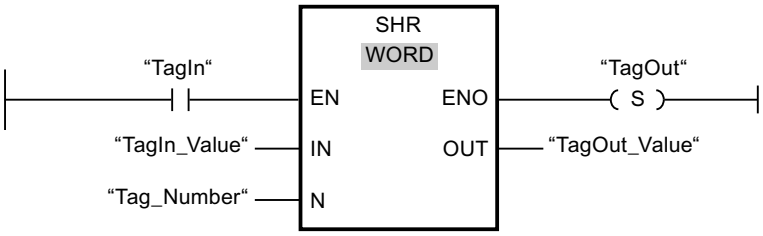
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	BYTE, WORD, DWORD, INT, DINT	I, Q, M, D, L, P or constant	Value to be shifted.
N	Input	BYTE, WORD, INT	I, Q, M, D, L, P or constant	Number of bit positions by which the value is shifted.
OUT	Output	BYTE, WORD, DWORD, INT, DINT	I, Q, M, D, L	Result of the instruction

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
N	Tag_Number	3
OUT	TagOut_Value	0000 0111 1111 0101

If operand "TagIn" has the signal state "1", the "Shift right" instruction is executed. The content of the operand "TagIn_Value" is shifted three bit positions to the right. The result is sent at the "TagOut_Value" output. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Setting the status bits for shift and rotate instructions (Page 2462)

SHL: Shift left

Description

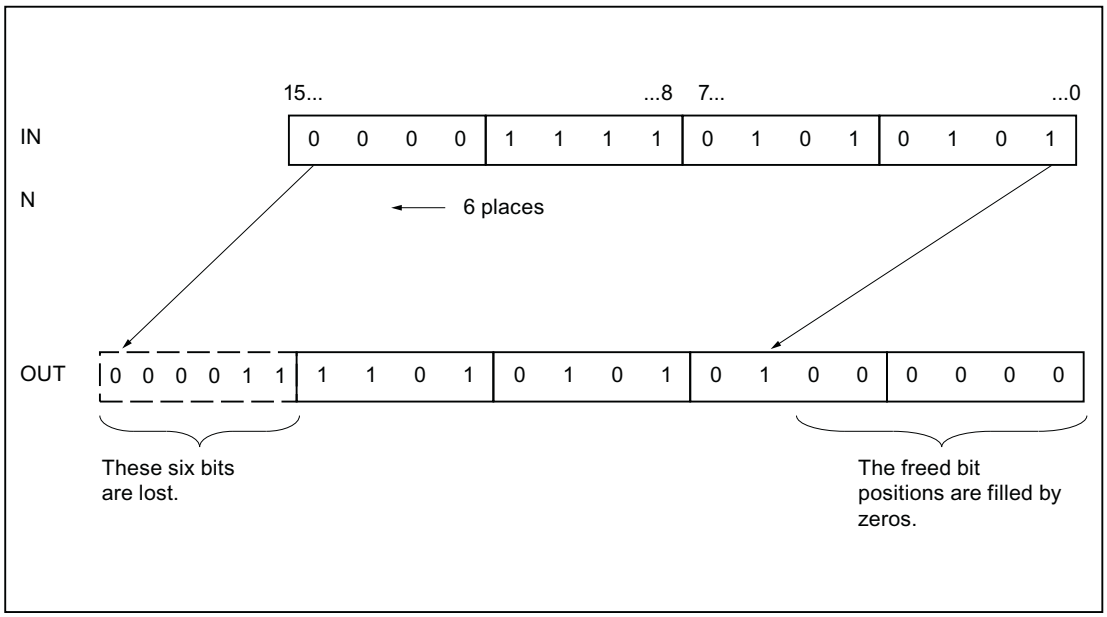
You can use the "Shift left" instruction to shift the content of the operand at the IN input bit-by-bit to the left and query the result at the OUT output. You use the N parameter to specify the number of bit positions by which the specified value is to be shifted.

If the value at the parameter N is "0", the value at input IN is copied to the operand at output OUT.

If the value at the parameter N is greater than the number of bit positions, the operand value at input IN is shifted to the left by the available number of bit positions.

The bit positions in the right part of the operand freed by shifting are filled with zeros.

The following figure shows how the content of an operand of the WORD data type is shifted by six bit positions to the left:



Parameters

The following table shows the parameters of the "Shift left" instruction:

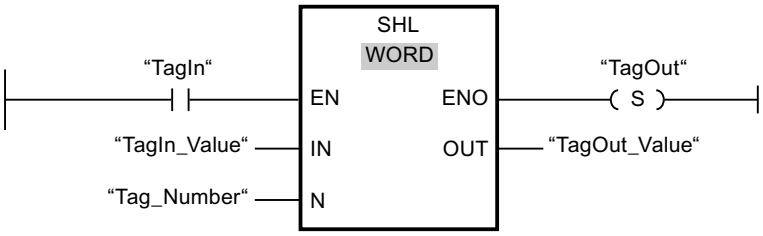
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	BYTE, WORD, DWORD, INT, DINT	I, Q, M, D, L, P or constant	Value to be shifted
N	Input	BYTE, WORD, INT	I, Q, M, D, L, P or constant	Number of bit positions by which the value is shifted
OUT	Output	BYTE, WORD, DWORD, INT, DINT	I, Q, M, D, L	Result of the instruction

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
N	Tag_Number	4
OUT	TagOut_Value	1111 1010 1111 0000

If operand "TagIn" has the signal state "1", the "Shift left" instruction is executed. The content of the operand "TagIn_Value" is shifted four bit positions to the left. The result is sent at the "TagOut_Value" output. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Setting the status bits for shift and rotate instructions (Page 2462)

ROR: Rotate right

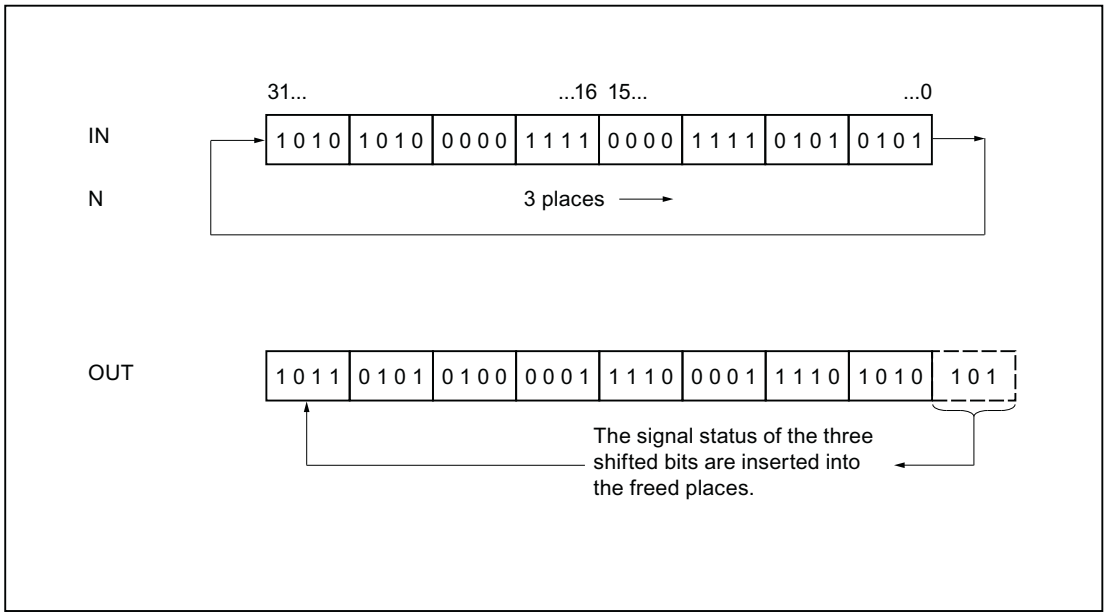
Description

You can use the "Rotate right" instruction to rotate the content of the operand at the IN input bit-by-bit to the right and query the result at the OUT output. You use the N parameter to specify the number of bit positions by which the specified value is to be rotated. The bit positions freed by rotating are filled with the bit positions that are pushed out.

If the value at the parameter N is "0", the value at input IN is copied to the operand at output OUT.

If the value at the N parameter is greater than the number of available bit positions, the operand value at the IN input is still rotated by the specified number of bit positions.

The following figure shows how the content of an operand of the DWORD data type is rotated three bit positions to the right:



Parameters

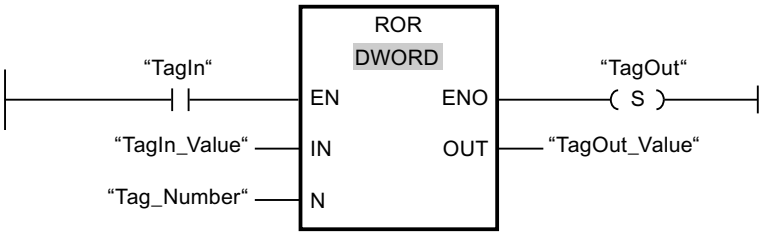
The following table shows the parameters of the "Rotate right" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	BYTE, WORD, DWORD, DINT	I, Q, M, D, L, P or constant	Value to be rotated
N	Input	BYTE, WORD, INT	I, Q, M, D, L, P or constant	Number of bit positions by which the value is rotated
OUT	Output	BYTE, WORD, DWORD, DINT	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	TagIn_Value	1010 1010 0000 1111 0000 1111 0101 0101
N	Tag_Number	5
OUT	TagOut_Value	1010 1101 0101 0000 0111 1000 0111 1010

If operand "TagIn" has the signal state "1", the "Rotate right" instruction is executed. The content of operand "TagIn_Value" is rotated five bit positions to the right. The result is sent at the "TagOut_Value" output. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Setting the status bits for shift and rotate instructions (Page 2462)

ROL: Rotate left

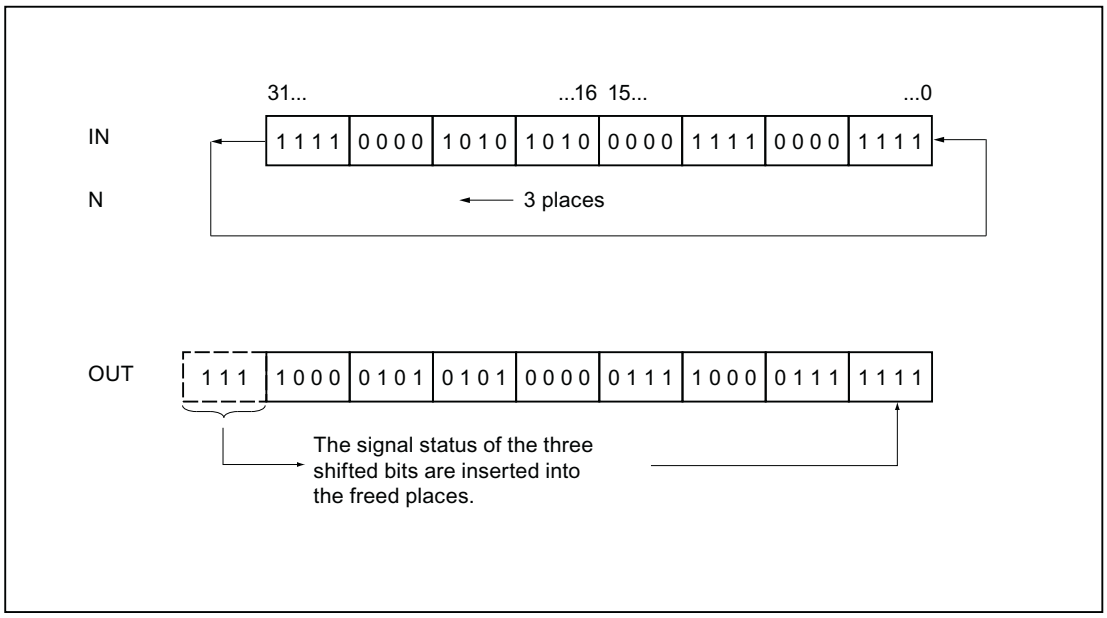
Description

You can use the "Rotate left" instruction to rotate the content of the operand at the IN input bit-by-bit to the left and query the result at the OUT output. You use the N parameter to specify the number of bit positions by which the specified value is to be rotated. The bit positions freed by rotating are filled with the bit positions that are pushed out.

When the value at the N parameter is "0", the value at the IN input is copied to the operand at the OUT output.

When the value at the N parameter is greater than the number of available bit positions, the operand value at the IN input is nevertheless rotated by the specified number of bit positions.

The following figure shows how the content of an operand of the DWORD data type is rotated three bit positions to the left:



Parameters

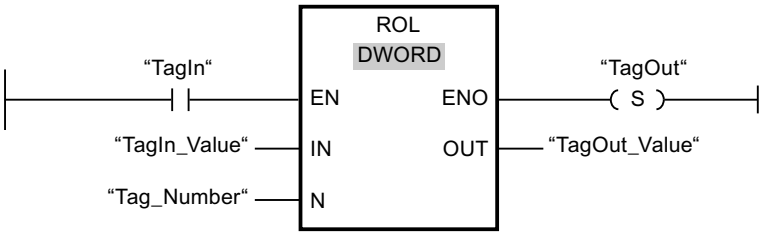
The following table shows the parameters of the "Rotate left" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	BYTE, WORD, DWORD, DINT	I, Q, M, D, L, P or constant	Value to be rotated
N	Input	BYTE, WORD, INT	I, Q, M, D, L, P or constant	Number of bit positions by which the value is rotated
OUT	Output	BYTE, WORD, DWORD, DINT	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	TagIn_Value	1111 0000 1010 1010 0000 1111 0000 1111
N	Tag_Number	5
OUT	TagOut_Value	0001 0101 0100 0001 1110 0001 1111 1110

If the "TagIn" input has the signal state "1", the "Rotate left" instruction is executed. The content of operand "TagIn_Value" is rotated five bit positions to the left. The result is sent at the "TagOut_Value" output. If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Selecting the data type of a LAD element (Page 2583)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Setting the status bits for shift and rotate instructions (Page 2462)

Additional instructions

---| |---: Get status bit

Description

With the "Get status bit" instruction you can query the signal state of the following status bits:

- BR: Binary result
- OV: Overflow

- OS: Exception Bit Overflow Stored
- CC 0 and CC 1: Conditional codes

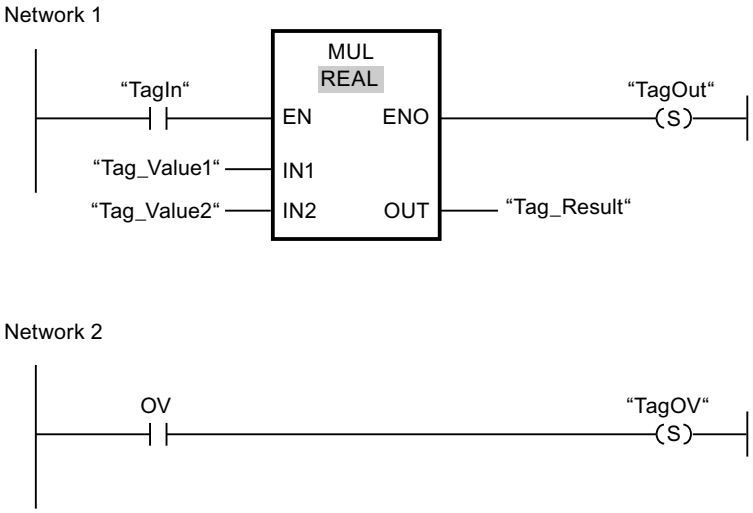
You define which status bits are queried by selecting the corresponding symbol from the drop-down list of the instruction. The following table shows the possible ways of querying status bits:

Symbol	Query	Explanation
==0	CC 0 = 0 and CC 1 = 0	The instruction asks whether the result of a mathematical function is equal to 0.
<>0	CC 0 = 0 and CC 1 = 1 or CC 0 = 0 and CC 1 = 0	The instruction asks whether the result of a mathematical function is not equal to 0.
>=0	CC 0 = 0	The instruction asks whether the result of a mathematical function is greater than or equal to 0.
<=0	CC 1 = 0	The instruction asks whether the result of a mathematical function is less than or equal to 0.
>0	CC 0 = 0 and CC 1 = 1	The instruction asks whether the result of a mathematical function is greater than 0.
<0	CC 0 = 1 and CC 1 = 0	The instruction asks whether the result of a mathematical function is less than 0.
UO	CC 0 = 1 and CC 1 = 1	The instruction asks whether the result of a mathematical function is invalid. The cause of an invalid result can be a floating-point number or division by zero.
OV	OV = 1	The instruction queries whether a number range overflow has occurred.
OS	OS = 1	The instruction queries whether a saving number range overflow has occurred.
BR	BR= 1	The instruction asks the signal state of the status bit BR.

The "Get status bit" instruction functions like a normally open contact. When the query is satisfied, the instruction has the signal state "1". If the query is not satisfied, the instruction has the signal state "0".

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state "1", the "Multiply" (MUL) instruction is executed. In the event of an overflow during the execution of the "Multiply" instruction, the status bit OV is set. Following the query of the status bit OV, the "Set output" instruction (S) is executed in network 2 and the operand "TagOV" is set.

See also

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Basic information on the status word (Page 2453)

Setting status bits (Page 2455)

---| / |---: Get negated status bit

Description

With the "Get negated status bit" instruction you can query the signal state of the following status bits and negate the query result:

- BR: Binary result
- OV: Overflow
- OS: Exception Bit Overflow Stored
- CC 0 and CC 1: Conditional codes

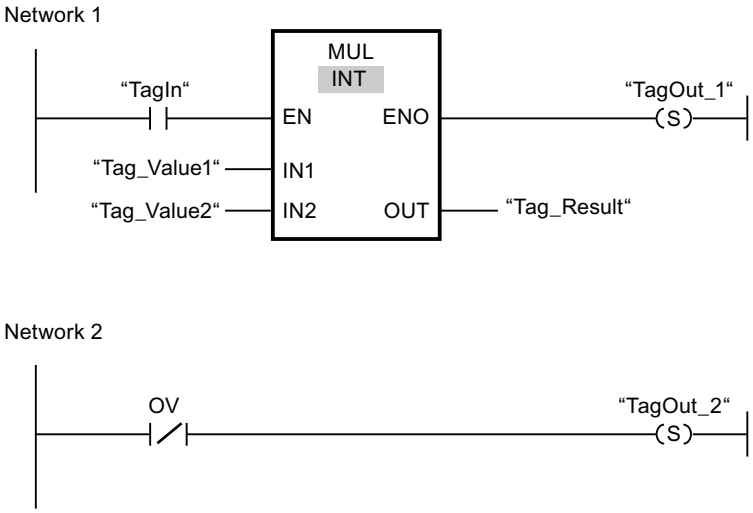
You define which status bits are queried by selecting the corresponding symbol from the drop-down list of the instruction. The following table shows the possible ways of querying status bits:

Symbol	Query	Explanation
==0	CC 0 = 0 and CC 1 = 0	The instruction asks whether the result of a mathematical function is equal to 0.
<>0	CC 0 = 0 and CC 1 = 1 or CC 0 = 0 and CC 1 = 0	The instruction asks whether the result of a mathematical function is not equal to 0.
>=0	CC 0 = 0	The instruction asks whether the result of a mathematical function is greater than or equal to 0.
<=0	CC 1 = 0	The instruction asks whether the result of a mathematical function is less than or equal to 0.
>0	CC 0 = 0 and CC 1 = 1	The instruction asks whether the result of a mathematical function is greater than 0.
<0	CC 0 = 1 and CC 1 = 0	The instruction asks whether the result of a mathematical function is less than 0.
UO	CC 0 = 1 and CC 1 = 1	The instruction asks whether the result of a mathematical function is invalid. The cause of an invalid result can be a floating-point number or division by zero.
OV	OV = 1	The instruction asks whether a number range overflow has occurred.
OS	OS = 1	The instruction asks whether a stored number range overflow has occurred in an instruction.
BR	BR= 1	The instruction asks the signal state of the status bit BR.

The "Get negated status bit" instruction functions like a normally closed contact. When the query is satisfied, the instruction has the signal state "0". If the query is not satisfied, the instruction has the signal state "1".

Example

The following example shows how the instruction works:



If operand "TagIn" has the signal state "1", the "Multiply" (MUL) instruction is executed. If an overflow does not occur during the execution of the instruction, the status bit OV is reset to "0". Following the query of the status bit OV, the instruction "Set output" (S) is executed in network 2 and the operand "TagOut_2" is set.

See also

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Basic information on the status word (Page 2453)

Setting status bits (Page 2455)

---(CALL): Call block

Description

The "Call block" instruction calls a function (FC) without parameter. The call is conditional or unconditional depending on the preceding logic operation.

In the statement part of a function (FC), you cannot specify any parameter of the BLOCK_FC data type as the operand for a conditional call. You can, however, specify a parameter of the type BLOCK_FC as the operand in a function block (FB).

A conditional call is then only executed if the result of logic operation (RLO) is "1". If a conditional call is not executed the RLO is "0" after the call operation. If the operation is executed, it works as follows:

- It stores the return address of the calling block.
- It stores the two data block registers (data block and instance data block).

- It updates the local data for the called function.
- It moves the MA bit (active MCR bit) to the block stack (B stack).

Parameters

The following table shows the parameters of the "Call block" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Block>	Output	BLOCK_FC	-	Called function

Example

The following example shows how the instruction works:

The function FC3 is called if the operand "TagIn_1" has the signal state "1".

See also

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Master control relay (Page 2471)

Important notes on using MCR functions (Page 2472)

---(SAVE): Save RLO in BR bit

Description

Use the instruction "Save RLO in BR bit" to save the result of logic operation (RLO) in the binary result (BR). During processing, the instruction transfers the signal state of the current RLO to the status bit BR. The instruction operates independent of the conditions and does not affect any other status bits.

Note

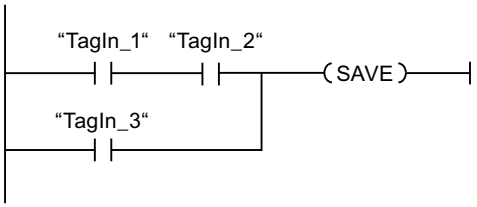
After application of the instruction "Save RLO in BR bit", the BR bit can be modified again by the following instructions in the same or lower-level blocks.

Use the instruction "Save RLO in BR bit" to check the execution status of a block. If you program the instruction "Save RLO in BR bit" at the end of a block, for example, the BR bit is set to the signal state of the current result of logic operation of the block. If the block is executed without errors, the BR bit and the enable output ENO in the calling block are set to signal state "1".

Signal state "0" is assigned to the BR bit if an error occurs during processing of the block. You can program a troubleshooting routine by querying the BR bit.

Example

The following example shows how the instruction works:



The BR bit is set to the signal state "1" if one of the following conditions is met:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The operand "TagIn_3" has the signal state "1".

See also

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Basic information on the status word (Page 2453)

Setting status bits (Page 2455)

Example of the use of the EN/ENO mechanism in LAD (Page 2450)

---(MCR<): Open MCR ranges

Description

The "Open MCR ranges" instruction saves the result of logic operation (RLO) in the MCR stack and enables MCR dependency within an MCR range (MCR = Master Control Relay).

The MCR stack can contain up to a maximum of 8 entries and functions like a LIFO buffer (Last In, First Out). When the stack is full, the instruction generates an error message (MCRF).

Note

Read the information on using MCR functions in the "Important notes on using MCR functionality" at "See also".

Use the instructions "Enable MCR range" (MCRA) and "Disable MCR range" (MCRD) to specify the start and end of an MCR range. Within the defined MCR range you can enable and disable MCR dependency for one or more program sections.

MCR dependency is enabled when the RLO is "1" before processing of the instruction. If the RLO is "0" before the instruction, MCR dependency is disabled. The end of the MCR dependent

program section is programmed using the instruction "Close MCR ranges". The program section between the two instructions is affected by the MCR. Only the following MCR-dependent instructions are affected by this:

- Assignment
- Set output (S)
- Reset output (R)
- Reset/set flip-flop (RS)
- Set/reset flip-flop (SR)
- Move value (MOVE)

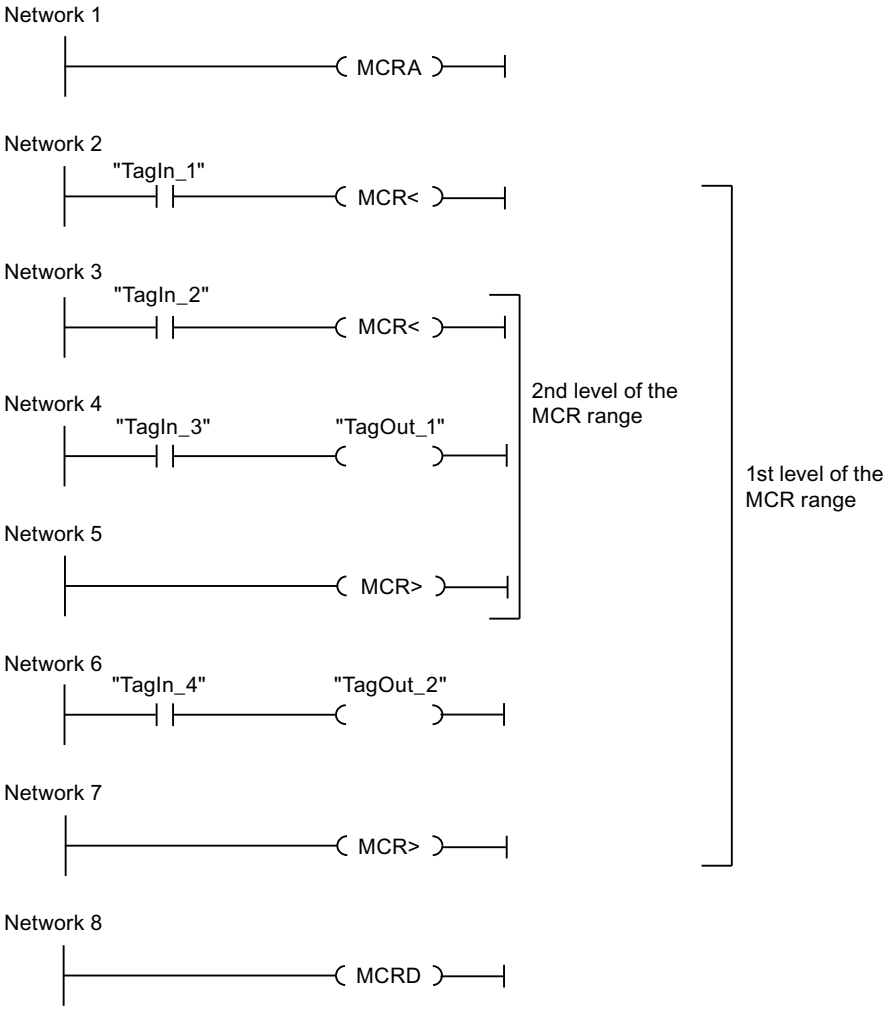
You can also nest the enabling and disabling of MCR dependency by programming the instruction "Open MCR ranges" in an MCR dependent program section. You can nest the instruction up to 8 levels deep. The MCR dependency of the higher-level program section hereby controls the MCR dependencies in all lower-level program sections. If MCR dependency is disabled in the higher-level program section, this MCR dependency cannot be switched on in any of the lower-level MCR dependent program sections.

If you call a block within an MCR range, MCR dependency is disabled in the called block. When the block is exited, MCR dependency is set so that it is the same as before the block call.

The instructions "Open MCR ranges" and "Close MCR ranges" must always be programmed in pairs. If one of the two instructions is not programmed, an MCR stack error is triggered.

Example

The following example shows how the instruction works:



The start of the MCR range is programmed in network 1. If the "TagIn_1" operand returns signal state "1", MCR dependency is enabled in the first level of the MCR range. In this case, the signal state of the "TagIn_4" operand is assigned to the "TagOut_2" operand (network 6). If the "TagIn_1" operand returns signal state "0", MCR dependency is disabled and the "TagOut_2" operand is reset to "0". MCR dependency of the MCR range is disabled in network 7.

In network 3, MCR dependency is enabled in the second level of the MCR range, when the "TagIn_2" operand has signal state "1" and the first level of the MCR range is enabled. In this case, the signal state of the "TagIn_3" operand is assigned to the "TagOut_1" operand. If the "TagIn_2" operand returns signal state "0", MCR dependency is disabled and the "TagOut_1" operand is reset to "0". MCR dependency in the second level of the MCR range is disabled in network 5. The end of the MCR range is programmed in network 8.

See also

Important notes on using MCR functions (Page 2472)

Master control relay (Page 2471)

---(MCR>): Close MCR ranges (Page 5532)

---(MCRA): Enable MCR range (Page 5534)

---(MCRD): Disable MCR range (Page 5535)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

---(MCR>): Close MCR ranges

Description

The instruction "Close MCR ranges" completes an MCR dependent (MCR = Master Control Relay) program section. In this case the instruction deletes the result of logic operation (RLO) from the last entry in the MCR stack.

The MCR stack can contain up to a maximum of 8 entries and functions like a LIFO buffer (Last In, First Out). If the stack is already empty, the instruction generates an error message (MCRF).

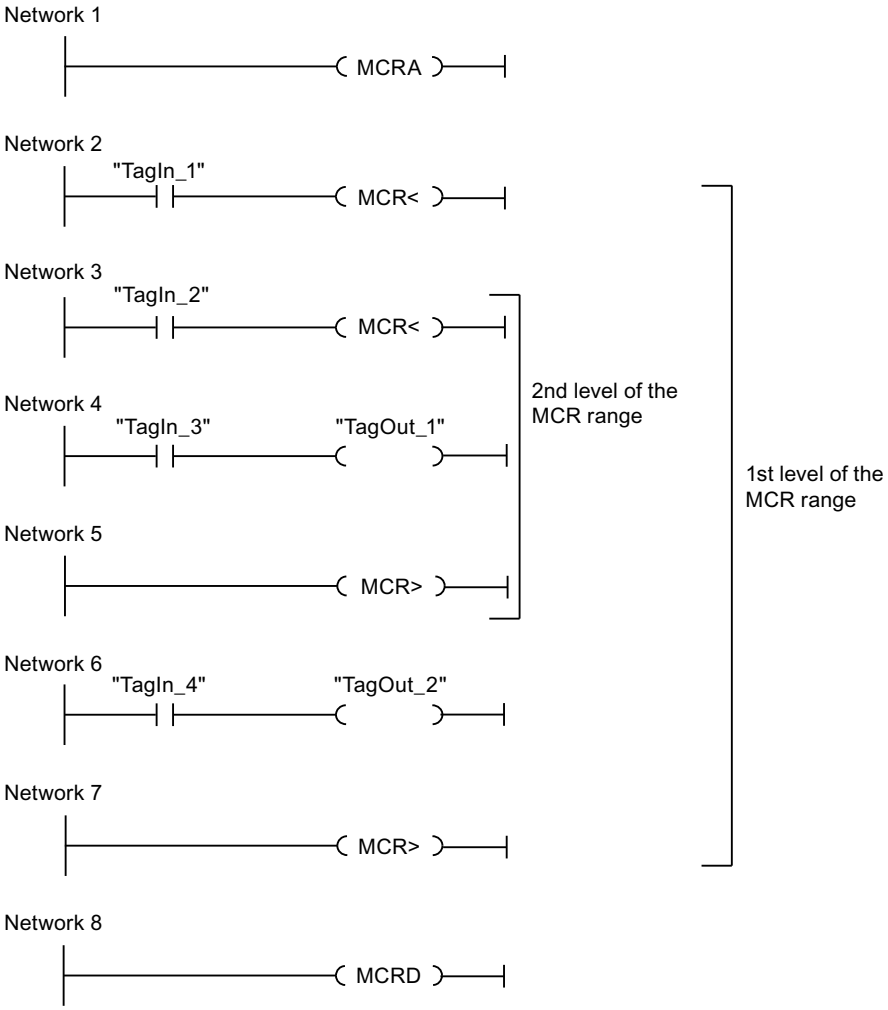
Note

Read the information on using MCR functions in the "Important notes on using MCR functionality" at "See also".

Use the instruction "Open MCR ranges" to program MCR dependency. The instructions "Open MCR ranges" and "Close MCR ranges" must always be programmed in pairs. If one of the two instructions is not programmed, an MCR stack error is triggered.

Example

The following example shows how the instruction works:



The start of the MCR range is programmed in network 1. If the "TagIn_1" operand returns signal state "1", MCR dependency is enabled in the first level of the MCR range. In this case, the signal state of the "TagIn_4" operand is assigned to the "TagOut_2" operand (network 6). If the "TagIn_1" operand returns signal state "0", MCR dependency is disabled and the "TagOut_2" operand is reset to "0". MCR dependency of the MCR range is disabled in network 7.

In network 3, the MCR-dependency is enabled in the second level of the MCR range, when the "TagIn_2" operand has signal state "1". In this case, the signal state of the "TagIn_3" operand is assigned to the "TagOut_1" operand. If the "TagIn_2" operand returns signal state "0", the MCR dependency is disabled and the "TagOut_1" operand is reset to "0" regardless of the signal state of the "TagIn_3" operand. MCR dependency in the second level of the MCR range is disabled in network 5. The end of the MCR range is programmed in network 8.

See also

Master control relay (Page 2471)

Important notes on using MCR functions (Page 2472)

---(MCR<): Open MCR ranges (Page 5529)

---(MCRA): Enable MCR range (Page 5534)

---(MCRD): Disable MCR range (Page 5535)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

---(MCRA): Enable MCR range

Description

Use the instruction "Enable MCR range" to specify the start of an MCR range. Within this range, you can switch MCR dependency on and off using the following instructions:

- ---(MCR<): Open MCR ranges
- ---(MCR>): Close MCR ranges

You specify the end of the MCR range with the instruction "Disable MCR range" (MCRD). The instructions for defining the start and the end of an MCR range must always be used in pairs.

Note

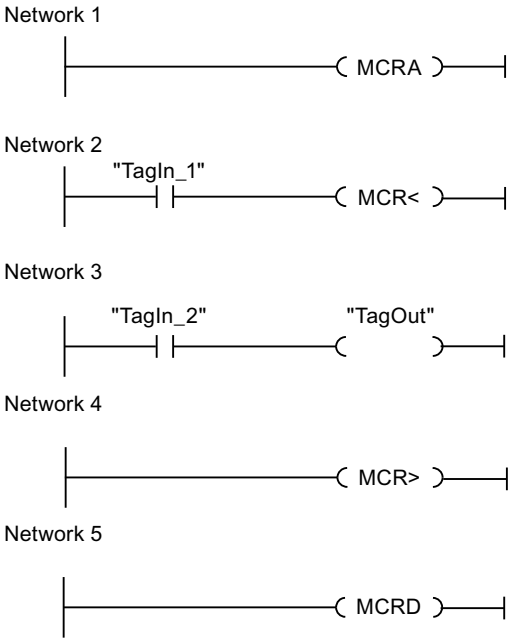
Read the information on using MCR functions in the "Important notes on using MCR functionality" at "See also".

If you call a block within an MCR range, MCR dependency is disabled in the called block. When a block is exited, the MCR dependency is set so that it is the same as before the block call.

The instruction is executed regardless of the status bits and without affecting these bits.

Example

The following example shows how the instruction works:



The start of the MCR range is programmed in network 1. If the "TagIn_1" operand returns the signal state "1", the MCR dependency is enabled. In this case, the signal state of the "TagIn_2" operand is assigned to the "TagOut" operand (network 3). If the "TagIn_1" operand returns signal state "0", the MCR dependency is disabled and the "TagOut" operand is reset to "0" regardless of the signal state of the "TagIn_2" operand. MCR dependency of the MCR range is disabled in network 4. The end of the MCR range is programmed in network 8.

See also

Master control relay (Page 2471)

Important notes on using MCR functions (Page 2472)

---(MCR<): Open MCR ranges (Page 5529)

---(MCR>): Close MCR ranges (Page 5532)

---(MCRD): Disable MCR range (Page 5535)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

---(MCRD): Disable MCR range

Description

Use the instruction "Disable MCR range" to specify the start of an MCR range.

You specify the start of the MCR range with the instruction "Enable MCR range" (MCRA). The instructions for defining the start and the end of an MCR range must always be used in pairs.

Note

Read the information on using MCR functions in the "Important notes on using MCR functionality" at "See also".

Within an MCR range you can program the MCR dependency with the help of the following instructions:

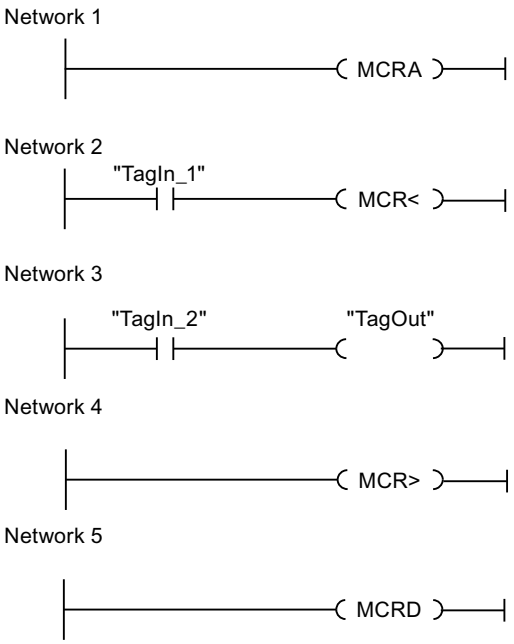
- ---(MCR<): Open MCR ranges
- ---(MCR>): Close MCR ranges

If you call a block within an MCR range, MCR dependency is disabled in the called block. When a block is exited, the MCR dependency is set so that it is the same as before the block call.

The instruction is executed regardless of the status bits and without affecting these bits.

Example

The following example shows how the instruction works:



The start of the MCR range is programmed in network 1. If the "TagIn_1" operand returns the signal state "1", the MCR dependency is enabled. In this case, the signal state of the "TagIn_2" operand is assigned to the "TagOut" operand (network 3). If the "TagIn_1" operand returns signal state "0", the MCR dependency is disabled and the "TagOut" operand is reset to "0" regardless of the signal state of the "TagIn_2" operand. MCR dependency of the MCR range is disabled in network 4. The end of the MCR range is programmed in network 8.

See also

Master control relay (Page 2471)

Important notes on using MCR functions (Page 2472)

---(MCR<): Open MCR ranges (Page 5529)

---(MCR>): Close MCR ranges (Page 5532)

---(MCRA): Enable MCR range (Page 5534)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

SET: Set bit array

Description

The "Set bit array" instruction sets the signal state of the bit in a specified area to "1". You use the pointer on the S_BIT parameter to define the starting point of the range. If the pointer points to the memory area of the external I/O, the instruction is not executed. At the N parameter, you specify the number of bits to be set in the specified area. When the N parameter has the value "0", calling the instruction has no effect.

The instruction is only executed when the MCR bit is "1". When the MCR bit has the signal state "0", the bits in the specified range are not changed.

Parameter

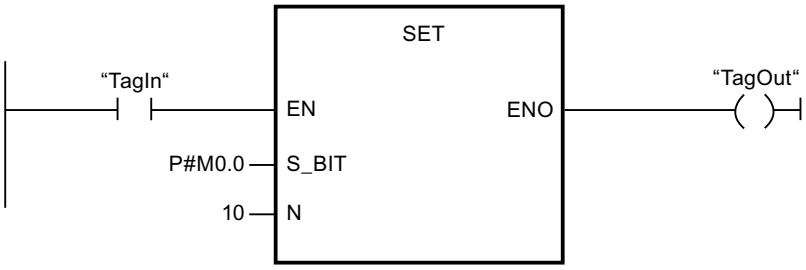
The following table shows the parameters of the "Set bit array" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
S_BIT	Input	POINTER	I, Q, M, D	Pointer to the first bit of the range whose bits are set. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	I, Q, M, D, L, P or constant	Number of bits to be set

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the TagIn operand has the signal state "1", the instruction sets 10 bits in the area from M0.0 to M1.2.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Master control relay (Page 2471)

Important notes on using MCR functions (Page 2472)

SETP: Set a bit array in the I/O area

Description

The instruction "Set a bit array in the I/O area" sets the bits of an I/O area to the signal state "1". At the same time the corresponding bits in the process image of the outputs are also set parallel to the I/O bits.

You use the N and SA parameters of the instruction to define the bit array that will be set. The bits of the array must be located in the part of the I/O area to which a process image is assigned.

If no I/O devices are plugged in for part of the selected bit array, the instruction attempts to set the entire bit array. In this case an error information is output on the RET_VAL parameter.

The I/O area is set byte-by-byte. If the bit array defined via the N and SA parameters does not begin or end at a byte boundary, calling the instruction has the following effect:

- The bits that are in the first or last byte to be transferred and that do not belong to the defined bit array, get the value of the corresponding bits in the process image output.

> **WARNING**
>
> **Unintended consequences**
>
> The setting of the bits to the value of the associated bits in the process image output can lead to unintended reactions such as starting a motor or turning off a cooling system.

- The bits belonging to the defined bit array are set to the signal state "1".

The instruction "Set a bit array in the I/O area" must be called in an MCR area.

Calling the "Set a bit array in the I/O area" instruction has no effect and the enable output ENO is set to "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The N parameter has the value "0"
- The MCR bit has the signal state "1".

Parameters

The following table shows the parameters of the "Set a bit array in the I/O area" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
N	Input	INT	I, Q, M, D, L or constant	Number of bits to be set
RET_VAL	Output	INT	I, Q, M, D, L	Error information
SA	Output	POINTER	P	Pointer to the first bit to be set

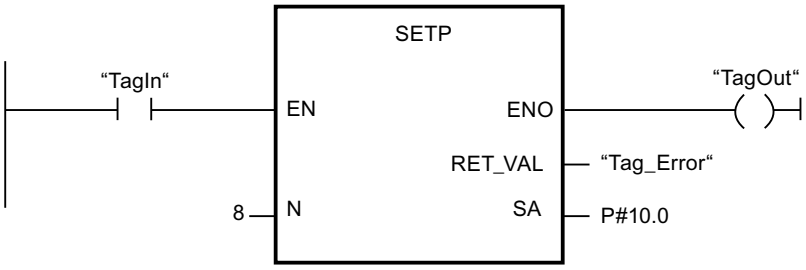
Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:



If the operand "TagIn" has the signal state "1", the instruction sets 8 bits in the I/O range corresponding to the outputs Q10.0 to Q10.7.

See also

Evaluating errors with output parameter RET_VAL (Page 5370)

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Switching display formats in the program status (Page 2980)

SETI: Set byte array

Description

The "Set byte array" instruction sets the signal state of the bit in a specified area of bytes to "1". You define the start point of the array on the S_BYTE. You define the size of the array by specifying the number of bits to be set on the N parameter of the instruction.

Note

The value of the N parameter must be a multiple of the number eight (for example, 8, 16, 24, etc.).

The pointer specified on the S_BYTE parameter must reference the external I/O memory area (P). Since the "I/0" memory area is accessed as bytes, words, or double words, the pointer on

the S_BYTE parameter must reference an operand that corresponds to a byte limit value, in other words the bit number of the pointer must be "0".

Note

The signal state of the corresponding bits in the process-image output is reset to "0".

Calling the "Set byte array" instruction has no effect and the ENO enable output is set to "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The pointer on the S_BYTE parameter does not reference the external I/O memory area.
- The pointer on the S_BYTE parameter references an operand that is not byte-aligned.
- The value of the N parameter is not a multiple of the number 8.
- The MCR bit has the signal state "0".

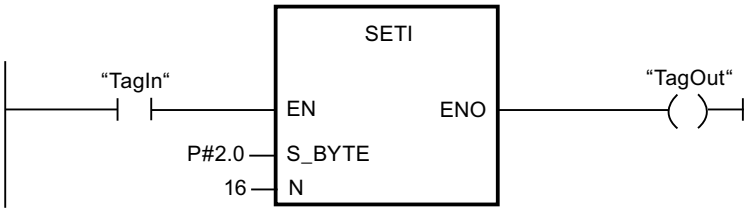
Parameters

The following table shows the parameters of the "Set byte array" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
S_BYTE	Input	POINTER	P or constant	Pointer to the first byte of the array whose bits are to be set. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	I, Q, M, D, L, P or constant	Size of the array to be set The array is specified by the number of bits as a multiple of eight (for example 8, 16, 24 etc.).

Example

The following example shows how the instruction works:



If the operand "TagIn" has the signal state "1", the instruction sets 2 bytes (16 bits) in the range from P2.0 to P3.7.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Master control relay (Page 2471)

Important notes on using MCR functions (Page 2472)

RESET: Reset bit array

Description

You use"Reset bit array" instruction to reset the signal state of the bits in a specified area to "0". You use the pointer on the S_BIT parameter to define the starting point of the range. If the pointer points to the memory area of the external I/O, the instruction is not executed. You specify the number of bits to be reset in the specified range at the N parameter of the instruction. When the N parameter has the value "0", calling the instruction has no effect.

The instruction is only executed when the MCR bit is "1". When the MCR bit has the signal state "0", the bits in the specified range are not changed.

Parameters

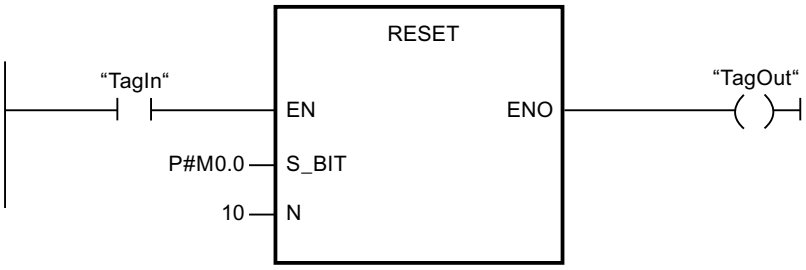
The following table shows the parameters of the "Reset bit array" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

Parameter	Declaration	Data type	Memory area	Description
S_BIT	Input	POINTER	I, Q, M, D	Pointer to the first bit of the range whose bits are to be reset For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	I, Q, M, D, L, P or constant	Number of bits to be reset.

Example

The following example shows how the instruction works:



If the operand "TagIn" has the signal state "1", the instruction resets 10 bits in the range from M0.0 to M1.2.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Master control relay (Page 2471)

Important notes on using MCR functions (Page 2472)

RESETP: Reset a bit array in the I/O area

Description

The instruction "Reset a bit array in the I/O area" sets the bits of an I/O area to the signal state "0". At the same time the corresponding bits in the process image of the outputs are also reset parallel to the I/O bits.

You use the N and SA parameters to define the bit array that will be reset. The bits of the array must be located in the part of the I/O area to which a process image is assigned.

If no I/O devices are plugged in for part of the selected bit array, the instruction attempts to set the entire bit array. In this case an error information is output on the RET_VAL parameter.

The I/O area is reset byte-by-byte. If the bit array defined via the N and SA parameters does not begin or end at a byte boundary, calling the instruction has the following effect:

- The bits that are in the first and last bytes to be transferred and that do not belong to the defined bit array, get the value of the corresponding bits in the process image output table.

  WARNING

  The setting of the bits at the value of the relevant bit in the process image output table can lead, for example, to unintended reactions such as starting a motor or turning off a cooling system.

- The bits belonging to the defined bit array are reset to the signal state "0".

The "Reset a bit array in the I/O area" instruction must be called in an MCR range.

Calling the "Reset a bit array in the I/O area" instruction has no effect and the enable output ENO is set to "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The N parameter has the value "0"
- The MCR bit has the signal state "1".

Parameters

The following table shows the parameters of the "Reset a bit array in the I/O area" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
N	Input	INT	I, Q, M, D, L or constant	Number of bits to be reset.
RET_VAL	Output	INT	I, Q, M, D, L	Error information
SA	Output	POINTER	P	Pointer to the first bit to be reset.

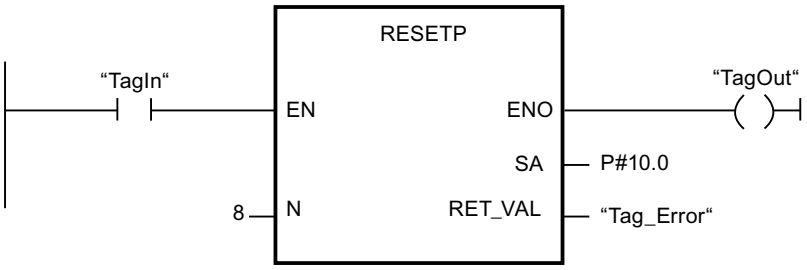
Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:



If the operand "TagIn" has the signal state "1", the instruction sets 8 bits in the I/O range corresponding to the outputs Q10.0 to Q10.7.

See also

Evaluating errors with output parameter RET_VAL (Page 5370)

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Switching display formats in the program status (Page 2980)

RESETI: Reset byte array

Description

The "Reset byte array" instruction is used to reset the signal state of the bits in a specified range of bytes to "0". You use the pointer on the S_BYTE parameter to define the starting point

of the range. You define the size of the range by specifying the number of bits to be reset on the N parameter of the instruction.

Note

The value of the N parameter must be a multiple of the number eight (for example, 8, 16, 24, etc.).

The pointer specified on the S_BYTE parameter must reference the external I/O memory area. Since the "I/0" memory area is accessed as bytes, words, or double words, the pointer on the S_BYTE parameter must reference an operand that corresponds to a byte limit value, in other words the bit number of the pointer must be "0".

Note

The signal state of the corresponding bits in the process-image output is reset to "0".

Calling the "Reset byte array" instruction has no effect and the ENO enable output is set to "0" if one of the following conditions applies:

- The EN enable input has the signal state "0".
- The pointer on the S_BYTE parameter does not reference the external I/O memory area.
- The pointer on the S_BYTE parameter references an operand that is not byte-aligned.
- The value of the N parameter is not a multiple of the number 8.
- The MCR bit has the signal state "0".

Parameters

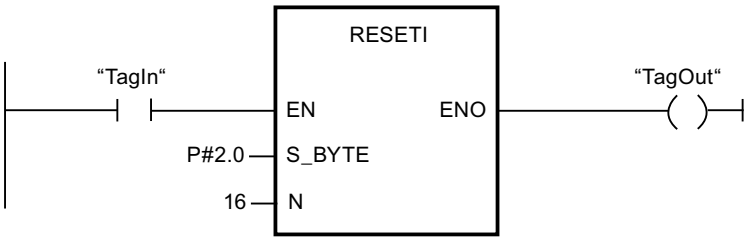
The following table shows the parameters of the "Reset byte array" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

Parameters	Declaration	Data type	Memory area	Description
S_BYTE	Input	POINTER	P or constant	Pointer to the first byte of the range whose bits are to be reset. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	I, Q, M, D, L, P or constant	Size of the array whose bits are to be reset. The array is specified by the number of bits as a multiple of eight (for example 8, 16, 24 etc.).

Example

The following example shows how the instruction works:



If the operand "TagIn" has the signal state "1", the instruction resets 2 bytes (16 bits) in the range from P2.0 to P3.7.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Master control relay (Page 2471)

Important notes on using MCR functions (Page 2472)

REPL_VAL: Enter substitute value

Description

The "Enter substitute value" instruction is used to enter a substitute value in the accumulator 1 (ACCU 1) from a synchronous error organization block.

Note

You can only call the "Enter substitute value" instruction in a synchronous error organization block (OB 121, OB 122).

You use the "Enter substitute value" instruction when you can no longer read any value from a module. The OB 122 is started after each access to this module. By calling the "Enter substitute value" instruction you can cause the substitute value to be loaded in accumulator 1. In this case the program is executed with this substitute value. You can take the required information for the selection of the substitute value from the local tags of the OB 122.

Parameters

The following table shows the parameters of the "Enter substitute value" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
VAL	Input	DWORD	I, Q, M, D, L or constant	Substitute value
RET_VAL	Output	INT	I, Q, M, D, L	Error information

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Description
0000	No error
8080	The instruction was not called from a synchronous error OB.
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Evaluating errors with output parameter RET_VAL (Page 5370)

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Switching display formats in the program status (Page 2980)

DRUM: Implement sequencer

Description

The "Implement sequencer" instruction is used to assign the programmed values of the OUT_VAL parameter of the corresponding step to the programmed output bits (OUT1 to OUT16) and the output word OUT_WORD. The specific step must thereby satisfy the conditions of the programmed enable mask on the S_MASK parameter while the instruction remains at this step. The instruction advances to the next step if the event for the step is true and the programmed time for the current step elapses, or if the value at the JOG parameter changes from "0" to "1". The instruction is reset if the signal state on the RESET parameter changes to "1". The current step is hereby equated to the preset step (DSP).

The amount of time spent on a step is determined by the product of the preset timebase (DTBP) and the preset counter value (S_PRESET) for each step. At the start of a new step, this calculated value is loaded into the DCC parameter, which contains the time remaining for the current step. If, for example the value at the DTBP parameter is "2" and the preset value for the first step is "100" (100 ms), the DCC parameter has the value "200" (200 ms).

A step can be programmed with a timer value, an event, or both. Steps that have an event bit and the timer value "0" advance to the next step as soon as the signal state of the event bit is "1". Steps that are programmed only with a timer value start the time immediately. Steps that are programmed with an event bit and a timer value greater than "0" start the time when the signal state of the event bit is "1". The event bits are initialized with a signal state of "1".

When the sequencer is on the last programmed step (LST_STEP) and the time for this step has expired, the signal state on the Q parameter is set to "1"; otherwise it is set to "0". When the parameter Q is set, the instruction remains on the step until it is reset.

In the configurable mask (S_MASK) you can selected the separate bits in the output word (OUT_WORD) and set or reset the output bits (OUT1 to OUT16) by means of the output values (OUT_VAL). If a bit of the configurable mask is in the signal state "1", the value OUT_VAL sets or resets the corresponding bit. If the signal state of a bit of the configurable mask is "0", the corresponding bit is left unchanged. All the bits of the configurable mask for all 16 steps are initialized with a signal state of "1".

The output bit on the OUT1 parameter corresponds to the least significant bit of the output word (OUT_WORD). The output bit on the OUT16 parameter corresponds to the most significant bit of the output word (OUT_WORD).

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the "Implement sequencer" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
RESET	Input	BOOL	I, Q, M, D, L	A signal state of "1" indicates a reset condition.
JOG	Input	BOOL	I, Q, M, D, L	When the signal state changes from "0" to "1", the instruction advances to the next step.
DRUM_EN	Input	BOOL	I, Q, M, D, L	A signal state of "1" allows the sequencer to advance based on the event and time criteria.
LST_STEP	Input	BYTE	I, Q, M, D, L or constant	Step number of the last step programmed.
EVENT(i), 1 ≤ i ≤ 16	Input	BOOL	I, Q, M, D, L	Event bit (i); Initial signal state is "1".
OUT(j), 1 ≤ j ≤ 16	Output	BOOL	I, Q, M, D, L	Output bit (j)
Q	Output	BOOL	I, Q, M, D, L	A signal state of "1" indicates that the time for the last step has elapsed.
OUT_WORD	Output	WORD	I, Q, M, D, L, P	Word address to which the sequencer writes the output values.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
JOG_HIS	Static	BOOL	I, Q, M, D, L	JOG parameter history bit

Parameter	Declaration	Data type	Memory area	Description
EOD	Static	BOOL	I, Q, M, D, L	A signal state of "1" indicates that the time for the last step has elapsed.
DSP	Static	BYTE	I, Q, M, D, L, P	Preset step of the sequencer
DSC	Static	BYTE	I, Q, M, D, L, P	Current step of the sequencer
DCC	Static	DWORD	I, Q, M, D, L, P	Current numerical value of the sequencer
DTBP	Static	WORD	I, Q, M, D, L, P	Preset timebase of the sequencer
PREV_TIME	Static	DWORD	I, Q, M, D, L or constant	Previous system time
S_PRESET	Static	ARRAY of WORD	I, Q, M, D, L	Count preset for each step [1 to 16] where 1 count = 1 ms.
OUT_VAL	Static	ARRAY of BOOL	I, Q, M, D, L	Output values for each step [1 to 16, 0 to 15].
S_MASK	Static	ARRAY of BOOL	I, Q, M, D, L	Configurable mask for each step [1 to 16, 0 to 15]. Initial signal states are "1".

For additional information on valid data types, refer to "See also".

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

ERR_CODE*	Explanation
W#16#0000	No error
W#16#000B	The value at the LST_STEP parameter is less than 1 or greater than 16.
W#16#000C	The value at the DSC parameter is less than 1 or greater than the value at the LST_STEP parameter.
W#16#000D	The value at the DSP parameter is less than 1 or greater than the value at the LST_STEP parameter.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Switching display formats in the program status (Page 2980)

DRUM_X: Implement sequencer

Description

The "Implement sequencer" instruction is used to Implement sequencer with maximum 16 steps. You specify the number of the first step on the DSP parameter. You specify the number of the last step on the LST_STEP parameter.

During each step, all 16 output bits OUT0 to OUT15 and the output parameter OUT_WORD (in which the output bits are collected together) are written. An output bit is assigned either the corresponding bit of the OUT_VAL array that you specified or the output bit of the preceding step. The value assigned depends on the value of the mask bit on the S_MASK parameter.

Note

The default setting for the mask bit is 0. If you want to change the current setting of one or more mask bits, make this change in the instance data block.

The "Implement sequencer" instruction advances to the next step when there is a rising edge on the JOG parameter. If the instruction is already in the last step, a positive edge on the parameter JOG sets the tags Qand EOD. In this case the DCC parameter is set to the value "0". The instruction remains in the last step until the RESET parameter is set to the value "1".

You can also assign parameters so that switching to the next step is time dependent. To do this, the DRUM_EN parameter has to be set to the value "1". The transition to the next step is made when the event bit (EVENTi) belonging to the current step is set and the time specified for the current step has expired. The time for a step is the product of the (DTBP) time base and the time factor valid for the current step (S_PRESET).

Note

The execution time (DCC) remaining in the current step is reduced only when the corresponding event bit (EVENTi) is set.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

If the value "1" is set on the RESET parameter when the instruction is called, the sequencer goes to the step whose number you assigned on the DSP parameter.

A step can be programmed with a timer value, an event, or both. Steps that have an event bit and the timer value "0" advance to the next step as soon as the signal state of the event bit is "1". Steps that are programmed only with a timer value start the time immediately. Steps that are programmed with an event bit and a timer value greater than "0" start the time when the signal state of the event bit is "1".

You can also move the next step at any time via the value at the JOG parameter.

When the instruction is called for the first time, you must set the value "1" on the RESET parameter.

When the sequencer is in the last step and the execution time specified for this step has expired, the parameters Q and EOD are set. The instruction remains in the last step until the RESET parameter changes to the value "1".

The instruction is executed only in the STARTUP and RUN modes.

The operating system resets the "Implement sequencer" instruction during a cold restart. During a warm restart, the instruction is not reset. If you want the "Implement sequencer" instruction to be initialized after a warm restart, you have to call the OB 100 with the value "1" on the RESET parameter.

Parameters

The following table shows the parameters of the "Implement sequencer" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
RESET	Input	BOOL	I, Q, M, D, L	The signal state "1" resets the sequencer. When the instruction is called for the first time, you have to set the value "1" on the RESET parameter.
JOG	Input	BOOL	I, Q, M, D, L	A positive signal edge (compared to the last call) switches the sequencer to the next step if it is not yet in the last step. The next step is enabled depending on the value you assigned to the DRUM_EN parameter.
DRUM_EN	Input	BOOL	I, Q, M, D, L	Control parameter that specifies whether time-dependent switching to the next step is possible. If the value is "1" a time-dependent switching to the next step is possible.

Parameter	Declaration	Data type	Memory area	Description
LST_STEP	Input	BYTE	I, Q, M, D, L or constant	Number of the last step Values from 1 to 16 are permitted.
EVENTi, 1 ≤ i ≤ 16	Input	BOOL	I, Q, M, D, L	Event bit number i (belongs to step i)
OUTj, 0 ≤ j ≤ 15	Output	BOOL	I, Q, M, D, L	Output bit number j (identical to the bit number j j of OUT_WORD)
Q	Output	BOOL	I, Q, M, D, L	Status parameter that indicates whether the specified execution time of the last step has expired.
OUT_WORD	Output	WORD	I, Q, M, D, L, P	Output bits collected together in a tag
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
JOG_HIS	Static	BOOL	I, Q, M, D, L	Not relevant to user (input parameter JOG of the preceding call)
EOD	Static	BOOL	I, Q, M, D, L	Identical to the output parameter Q
DSP	Static	BYTE	I, Q, M, D, L, P or constant	Number of the first step Values from 1 to 16 are permitted.
DSC	Static	BYTE	I, Q, M, D, L, P or constant	Number of the current step
DCC	Static	DWORD	I, Q, M, D, L, P	The execution time still remaining in the current step in milliseconds (ms) (only relevant if DRUM_EN = 1 and the corresponding event bit = 1)
DTBP	Static	WORD	I, Q, M, D, L, P or constant	The time base valid for all steps in milliseconds (ms)
PREV_TIME	Static	DWORD	I, Q, M, D, L or constant	(Not relevant for the user: system time of the previous call)
S_PRESET	Static	ARRAY of WORD	I, Q, M, D, L or constant	One-dimensional array with the time factor for each step. Recommended selection of the indices: [1 to 16]. In this case, S_PRESET [x] has the time factor of step x.

Parameter	Declaration	Data type	Memory area	Description
OUT_VAL	Static	ARRAY of BOOL	I, Q, M, D, L	Two-dimensional array with the values to be output in each step if they have not been masked out using S_MASK. Recommended selection of the indices: [1 to 16, 0 to 15]. In this case, OUT_VAL [x, y] has the value assigned to the output bit OUTy in step x.
S_MASK	Static	ARRAY of BOOL	I, Q, M, D, L	Two-dimensional array with the mask bits for each step. Recommended selection of the indices: [1 to 16, 0 to 15]. In this case, S_MASK [x, y] contains the mask bit for the "y"-th value output in step x. Meaning of the mask bits: • 0: The corresponding value of the previous step is assigned to the relevant output bit. • 1: The corresponding value from OUT_VAL is assigned to the relevant output bit.

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

ERR_CODE* (W#16#...)	Explanation
0000	No error
8081	Invalid value at parameter LST_STEP.
8082	Invalid value on parameter DSC.
8083	Invalid value on parameter DSP.
8084	The product DCC = DTBP * S_PRESET[DSC] exceeds the value 2**31-1 (approximately 24.86 days).
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Switching display formats in the program status (Page 2980)

DCAT: Discrete control-timer alarm

Description

You can use the "Discrete control-timer alarm" instruction to accumulate the time from the point at which theCMD parameter issued the command to open or close. The time is accumulated until the preset time (PT) is exceeded or the information is received that the device was opened or closed (O_FB or C_FB) within the specified time. If the preset time is exceeded before the information on the opening or closing of the device is received, the corresponding alarm is activated. If the signal state on the command input changes state before the preset time, the time is restarted.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The "Discrete control-timer alarm" instruction has the following reactions to the input conditions:

- When the signal state of the CMD parameter changes from "0" to "1", the signal states of the parameters Q, CMD_HIS, ET, OA and CA are influenced as follows:
  - The parameters Q and CMD_HIS are set to "1".
  - The parameters ET, OA and CA are reset to "0".
- When the signal state on the parameter CMD changes from "1" to "0", the parameters Q, ET, OA, CA and CMD_HIS are reset to "0".
- When the signal state of the parameters CMD and CMD_HIS is "1" and the parameter O_FB is set to "0", the time difference (ms) since the last execution of the instruction is added to the value at the parameter ET. If the value of the parameter ET exceeds the value of the parameter PT, the signal state on the parameter OA is set to "1". If the value of the parameter ET does not exceed the value of the parameter PT, the signal state on the parameter OA is reset to "0". The value at the parameter CMD_HIS is reset to the value of the parameter CMD.
- If the signal state of the parameters CMD, CMD_HIS and O_FB are set to "1" and the parameter C_FB has the value "0", the signal state of the parameter OA is set to "0". The value of the parameter ET is set to the value of the parameter PT. If the signal state of the parameter O_FB changes to "0", the alarm is set the next time the instruction is executed. The value of the parameter CMD_HIS is set to the value of the parameter CMD.
- If the parameters CMD, CMD_HIS and C_FB have the value "0", the time difference (ms) since the last execution of the instruction is added to the value of the parameter ET. If the value of the parameter ET exceeds the value of the parameter PT, the signal state of the parameter CA is set to "1". If the value at the parameter PT is not exceeded, the parameter CA has the signal state "0". The value of the parameter CMD_HIS is set to the value of the parameter CMD.

- If the parameters CMD, CMD_HIS and O_FB have the signal state "0" and the parameter C_FB is set to "1", the parameter CA is set to "0". The value of the parameter ET is set to the value of the parameter PT. If the signal state of the parameter C_FB changes to "0", the alarm is set the next time the instruction is executed. The value of the parameter CMD_HIS is set to the value of the parameter CMD.
- If the parameters O_FB and C_FB simultaneously have the signal state "1", the signal states of both alarm outputs are set to "1".

The "Discrete control-timer alarm" instruction has no error information.

Parameters

The following table shows the parameters of the "Discrete control-timer alarm" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
CMD	Input	BOOL	I, Q, M, D, L	A signal state of "0" indicates a "close" command. A signal state of "1" indicates an "open" command.
O_FB	Input	BOOL	I, Q, M, D, L	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L	Feedback input when closing
Q	Output	BOOL	I, Q, M, D, L	Shows the status of the parameter CMD
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
ET	Static	DINT	I, Q, M, D, L	Currently elapsed time, where 1 count = 1 ms
PT	Static	DINT	I, Q, M, D, L	Preset timer value, where 1 count = 1 ms
PREV_TIME	Static	DWORD	I, Q, M, D, L	Previous system time
CMD_HIS	Static	BOOL	I, Q, M, D, L	CMD history bit

For additional information on valid data types, refer to "See also".

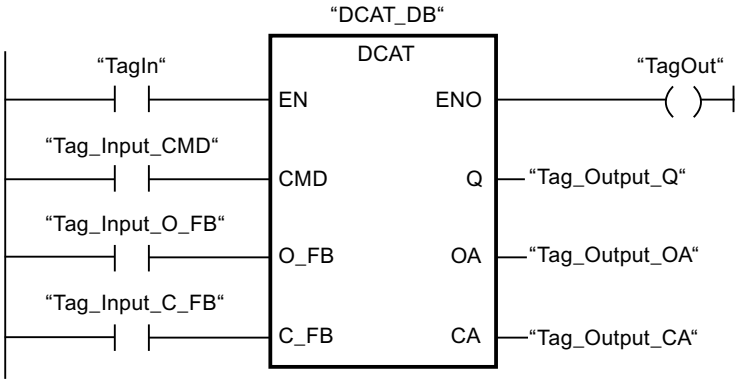
Example

In the following example the parameter CMD changes from "0" to "1". After the execution of the instruction the parameter Q is set to "1" and the two alarm outputs OA and CA have the

signal state "0". The parameter CMD_HIS of the instance data block is set to the signal state "1" and the parameter ET is reset to "0".

Note

You can initialize static parameters in the data block.



The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameter	Operand	Value
CMD	Tag_Input_CMD	TRUE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
Q	Tag_Output_Q	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the instance data block "DCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#12
PT	DBD8	L#222
CMD_HIS	DBX16.0	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
Q	Tag_Output_Q	TRUE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the instance data block "DCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#0
CMD_HIS	DBX16.0	TRUE

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

MCAT: Motor control-timer alarm

Description

The instruction "Motor control-timer alarm" is used to accumulate the time from the point of time from which the one of the command inputs (opening or closing) is switched on. The time is accumulated until the preset time is exceeded or the relevant feedback input indicates that the device has executed the requested operation within the specified time. If the preset time is exceeded before the feedback is received, the corresponding alarm is triggered.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Execution of the "Motor control-timer alarm" instruction

The following table shows the reactions of the instruction "Motor control-timer alarm" to the various input conditions:

Input parameters								Output parameters								
ET	O_HIS	C_HIS	O_CMD	C_CMD	S_CMD	O_FB	C_FB	OO	CO	OA	CA	ET	O_HIS	C_HIS	Q	Status
X	1	1	X	X	X	X	X	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	X	1	1	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	1	X	X	0	0	0	0	X	0	0	1	Stop
X	X	X	1	1	X	X	X	0	0	0	0	X	0	0	1	Stop
X	0	X	1	0	0	X	X	1	0	0	0	0	1	0	1	Start opening
<PT	1	0	X	0	0	0	X	1	0	0	0	INC	1	0	1	Open

Input parameters								Output parameters								
X	1	0	X	0	0	1	0	0	0	0	0	PT	1	0	1	Opened
>=PT	1	0	X	0	0	0	X	0	0	1	0	PT	1	0	0	Opening alarm
X	X	0	0	1	0	X	X	0	1	0	0	0	0	1	1	Start closing
<PT	0	1	0	X	0	X	0	0	1	0	0	INC	0	1	1	Close
X	0	1	0	X	0	0	1	0	0	0	0	PT	0	1	1	Closed
>=PT	0	1	0	X	0	X	0	0	0	0	1	PT	0	1	0	Closing alarm
X	0	0	0	0	0	X	X	0	0	0	0	X	0	0	1	Stopped
Legend:																
INC	Add the time difference (ms) since the last processing of the FB to ET															
PT	PT is set to the same value as ET															
X	Cannot be used															
<PT	ET < PT															
>=PT	ET >= PT															
If the input parameters O_HIS and C_HIS both have the signal state "1", they are immediately set to signal state "0". In this case, the last row in the table (X) mentioned above is valid. Because it is therefore no longer possible to check whether the input parameters O_HIS and C_HIS have the signal state "1", the output parameters are set as follows in this case: OO = FALSE CO = FALSE OA = FALSE CA = FALSE ET = PT Q = TRUE																

The "Motor control-timer alarm" instruction has no error information.

Parameters

The following table shows the reactions of the instruction "Motor control-timer alarm":

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
O_CMD	Input	BOOL	I, Q, M, D, L	"Open" command input
C_CMD	Input	BOOL	I, Q, M, D, L	"Close" command input
S_CMD	Input	BOOL	I, Q, M, D, L	"Stop" command input
O_FB	Input	BOOL	I, Q, M, D, L	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L	Feedback input when closing
OO	Output	BOOL	I, Q, M, D, L	"Open" output

Parameter	Declaration	Data type	Memory area	Description
CO	Output	BOOL	I, Q, M, D, L	"Close" output
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
Q	Output	BOOL	I, Q, M, D, L	A signal state of "0" indicates an error condition.
ET	Static	DINT	I, Q, M, D, L	Currently elapsed time, where 1 count = 1 ms
PT	Static	DINT	I, Q, M, D, L	Preset time value, where 1 count = 1 ms
PREV_TIME	Static	DWORD	I, Q, M, D, L	Previous system time
O_HIS	Static	BOOL	I, Q, M, D, L	"Open" history bit
C_HIS	Static	BOOL	I, Q, M, D, L	"Close" history bit

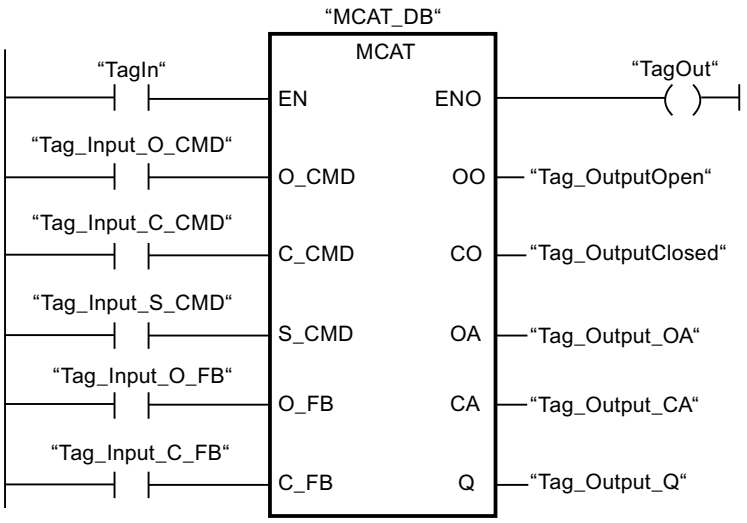
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.



The following table shows how the instruction works using specific operand values:

Before processing

In this example the following values are used for the input and output parameters:

Parameter	Operand	Value
O_CMD	Tag_Input_O_CMD	TRUE
C_CMD	Tag_Input_C_CMD	FALSE
S_CMD	Tag_Input_S_CMD	FALSE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
OO	Tag_OutputOpen	FALSE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	FALSE

The following values are saved in the instance data block "MCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#2
PT	DBD8	L#22
O_HIS	DBX16.0	TRUE
C_HIS	DBX16.1	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OO	Tag_OutputOpen	TRUE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	TRUE

The following values are saved in the instance data block "MCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#0
O_HIS	DBX16.0	TRUE
CMD_HIS	DBX16.1	FALSE

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

IMC: Compare input bits with the bits of a mask

Description

The instruction "Compare input bits with the bits of a mask" is used to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bits of a mask. Up to 16 steps with masks can be programmed. The value of the IN_BIT0 parameter is compared with the value of the mask CMP_VAL[x,0], with "x" indicating the step number. On the CMP_STEP parameter, you specify the step number of the mask that is used for the comparison. All programmed values are compared in the same manner. Unprogrammed input bits or unprogrammed bits of the mask have a default signal state FALSE.

If a match is found in the comparison, the signal state of the OUT parameter is set to "1". Otherwise the OUT parameter is set to "0".

If the value of CMP_STEP parameter is greater than 15, the instruction is not executed. An error message is output at the ERR_CODE parameter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the "Compare input bits with the bits of a mask" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN_BIT0	Input	BOOL	I, Q, M, D, L	Input bit 0 to be compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L	Input bit 1 to be compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L	Input bit 2 to be compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L	Input bit 3 to be compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L	Input bit 4 to be compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L	Input bit 5 to be compared with bit 5 of the mask.

Parameter	Declaration	Data type	Memory area	Description
IN_BIT6	Input	BOOL	I, Q, M, D, L	Input bit 6 to be compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L	Input bit 7 to be compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L	Input bit 8 to be compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L	Input bit 9 to be compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L	Input bit 10 to be compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L	Input bit 11 to be compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L	Input bit 12 to be compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L	Input bit 13 to be compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L	Input bit 14 to be compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L	Input bit 15 to be compared with bit 15 of the mask.
CMP_STEP	Input	BYTE	I, Q, M, D, L, P or constant	The step number of the mask used for the comparison.
OUT	Output	BOOL	I, Q, M, D, L	A signal state of "1" indicates that a match was found. A signal state of "0" indicates that a match was found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
CMP_VAL	Static	ARRAY OF BOOL	I, Q, M, D, L	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

For additional information on valid data types, refer to "See also".

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000A	The value at the CMP_STEP parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)
Basics of the EN/ENO mechanism (Page 2447)
Inserting LAD elements (Page 2579)
Editing LAD elements (Page 2594)
Inserting operands into LAD instructions (Page 2601)
Switching display formats in the program status (Page 2980)

SMC: Compare scan matrix

Description

The "Compare scan matrix" instruction is used to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bits of the comparison masks for each step. Processing starts at step 1 and is continued until the last programmed step (LAST) or until a match is found. The input bit of the IN_BIT0 parameter is compared with the value of the mask CMP_VAL[x,0], with "x" indicating the step number. All programmed values are compared in the same manner. If a match is found the signal state of the OUT parameter is set to "1" and the step number with the matching mask is written in the OUT_STEP parameter. Unprogrammed input bits or unprogrammed bits of the mask have the default signal state FALSE.. If several steps have a corresponding mask, only the the first match is specified at the OUT_STEP parameter. If no match is found, the signal state of the OUT parameter is set to "0". In this case the value at the OUT_STEP parameter is greater by "1" than the value at the LAST parameter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the "Compare scan matrix" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN_BIT0	Input	BOOL	I, Q, M, D, L	Input bit 0 to be compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L	Input bit 1 to be compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L	Input bit 2 to be compared with bit 2 of the mask.

Parameter	Declaration	Data type	Memory area	Description
IN_BIT3	Input	BOOL	I, Q, M, D, L	Input bit 3 to be compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L	Input bit 4 to be compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L	Input bit 5 to be compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L	Input bit 6 to be compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L	Input bit 7 to be compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L	Input bit 8 to be compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L	Input bit 9 to be compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L	Input bit 10 to be compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L	Input bit 11 to be compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L	Input bit 12 to be compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L	Input bit 13 to be compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L	Input bit 14 to be compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L	Input bit 15 to be compared with bit 15 of the mask.
OUT	Output	BOOL	I, Q, M, D, L	A signal state of "1" indicates that a match was found. A signal state of "0" indicates that no match was found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
OUT_STEP	Output	BYTE	I, Q, M, D, L, P	Contains the step number with the matching mask, or the step number which is greater by "1" than the value at the LAST parameter, provided no match is found.
LAST	Static	BYTE	I, Q, M, D, L, P	Specifies the step number of the last step to be scanned for a matching mask.
CMP_VAL	Static	ARRAY of WORD	I, Q, M, D, L	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

For additional information on valid data types, refer to "See also".

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000E	The value at the LAST parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Switching display formats in the program status (Page 2980)

LEAD_LAG: Lead and lag algorithm

Description

The "Lead and lag algorithm" instruction is used to process signals with an analog tag. The gain value at the GAIN parameter must be greater than zero. The result of the instruction "Lead and lag algorithm" is calculated using the following equation:

$$OUT = \left[\frac{LG_TIME}{LG_TIME + SAMPLE_T}\right] PREV_OUT + GAIN \left[\frac{LD_TIME + SAMPLE_T}{LG_TIME + SAMPLE_T}\right] IN - GAIN \left[\frac{LD_TIME}{LG_TIME + SAMPLE_T}\right] * PREV_IN$$

The "Lead and lag algorithm" instruction supplies plausible results only when processing is in fixed program cycles. The same units must be specified for the LD_TIME, LG_TIME, and SAMPLE_T parameters. At LG_TIME > 4 + SAMPLE_T, the instruction approaches the following function:

OUT = GAIN * ((1 + LD_TIME * s) / (1 + LG_TIME * s)) * IN

When the value of the GAIN parameter is less than or equal to zero, the calculation is not performed and an error information is output on the ERR_CODE parameter.

You can use the "Lead and lag algorithm" instruction in conjunction with loops as a compensator in dynamic feed-forward control. The instruction consists of two operations. The "Lead" operation shifts the phase of the OUT output so that the output leads the input. The "Lag" operation, on the other hand, shifts the output so that the output lags behind the input. Because the "Lag" operation is equivalent to an integration, it can be used as a noise suppressor or as a low-pass filter. The "Lead" operation is equivalent to a differentiation and can therefore be used as a high-pass filter. The two operations together (Lead and Lag) result in the output phase lagging behind the the input at lower frequencies and leading it at higher

frequencies. This means that the "Lead and lag algorithm" instruction can be used as a band pass filter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the "Lead and lag algorithm" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	REAL	I, Q, M, D, L, P or constant	The input value of the current sample time (cycle time) to be processed. Constants can also be specified on the IN parameter.
SAMPLE_T	Input	INT	I, Q, M, D, L, P or constant	Sample time Constants can also be specified on the SAMPLE_T parameter.
OUT	Output	REAL	I, Q, M, D, L, P	Result of the instruction
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
LD_TIME	Static	REAL	I, Q, M, D, L, P or constant	Lead time in the same unit as sample time.
LG_TIME	Static	REAL	I, Q, M, D, L, P or constant	Lag time in in the same unit as sample time.
GAIN	Static	REAL	I, Q, M, D, L, P or constant	Gain as % / % (the ratio of the change in output to a change in input as a steady state).
PREV_IN	Static	REAL	I, Q, M, D, L, P or constant	Previous input
PREV_OUT	Static	REAL	I, Q, M, D, L, P or constant	Previous output

For additional information on valid data types, refer to "See also".

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

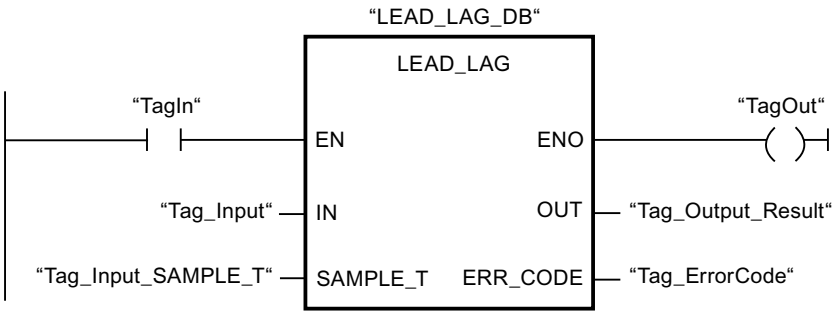
Error code* (W#16#...)	Explanation
0000	No error
0009	The value at the GAIN parameter is less than or equal to zero.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.



The following table shows how the instruction works using specific operand values:

Before processing

In this example the following values are used for the input parameters:

Parameter	Operand	Value
IN	Tag_Input	2.0
SAMPLE_T	Tag_InputSampleTime	10

The following values are saved in the instance data block "LEAD_LAG_DB" of the instruction:

Parameter	Address	Value
LD_TIME	DBD12	2.0
LG_TIME	DBD16	2.0
GAIN	DBD20	1.0
PREV_IN	DBD24	6.0
PREV_OUT	DBD28	6.0

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OUT	Tag_Output_Result	2.0

The following values are saved in the instance data block "LEAD_LAD_DB" of the instruction:

Parameter	Operand	Value
PREV_IN	DBD24	2.0
PREV_OUT	DBD28	2.0

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Switching display formats in the program status (Page 2980)

TONR_X: Time accumulator

Description

The "Time accumulator" instruction is used to accumulate time values within a period set by the PT parameter. When the TMR_EN parameter changes to signal state "1", the instruction is executed and the time set at PT starts. The "Time accumulator" instruction adds the value of the DELTA_T parameter and the value of the ET parameter as long as the following conditions are satisfied:

- The TMR_EN parameter has the signal state "1".
- The timer value at the ET parameter is less than the timer value at the PT parameter.
- The RESET parameter has the signal state "0".

When the signal state on the TMR_EN parameter changes to "0", the time on the DELTA_T parameter is no longer added to the timer value at the ET parameter.

When the timer value at the ET parameter reaches the timer value at the PT parameter the Q parameter has the signal state "1". As long as the Q parameter has the signal state "1", the value at the ET parameter remains unchanged. The value at the ET parameter is reset to "0" when the value "1" is set on the RESET parameter.

Since the "Time accumulator" instruction uses the execution time (DELTA_T) of the last cycle of the organization block (OB), the instruction may only be called in cyclic organization blocks.

Note

You have to move the cycle time of the organization block from the startup information to the global tag of the DELTA_T parameters.

Parameters

The following table shows the parameters of the "Time accumulator" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
TMR_EN	Input	BOOL	I, Q, M, D, L	Activates the time for accumulating the timer value.
RESET	Input	BOOL	I, Q, M, D, L	Reset input
PV	Input	DINT	I, Q, M, D, L, P or constant	Maximum duration of time recording
DELTA_T	Input	INT	I, Q, M, D, L or constant	Cycle time of OB from previous cycle.
Q	Output	BOOL	Q, M, D, L	Output that is set when the PT time expires.
ET	InOut	DINT	I, Q, M, D, L	Current timer value of the elapsed time

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

WSR: Save data to shift register

Description

The "Save data to shift register" instruction is used to write a bit from a specified source to a shift register. Each element of the register is shifted into the next address. The data contained in the last address of the shift register is lost after the shift.

Each time the instruction is executed with the value "0" on the RESET parameter, new data are read from the source (S_DATA) and shifted to the start address (START) of the shift register. The LENGTH parameter specifies the number of elements to be shifted. You specify the data type of the elements on the E_TYPE parameter. The instruction is not executed if an invalid data type is specified on the E_TYPE parameter.

The register is deleted if the RESET parameter is set to "1" when the instruction is executed. The Q parameter is set to the signal state "1" when the shift register is filled with zeros or deleted.

Parameters

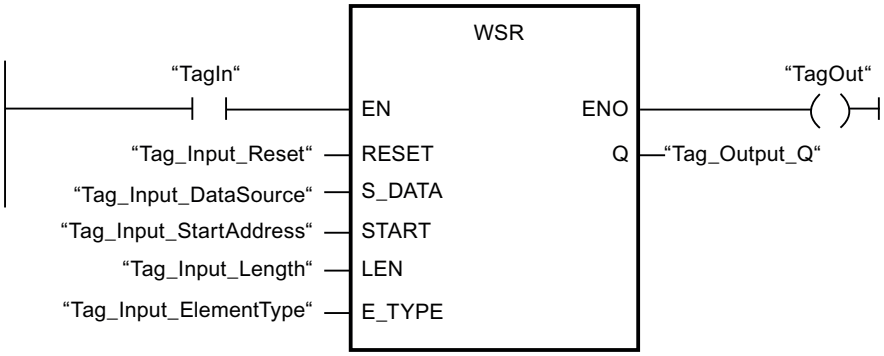
The following table shows the parameters of the "Save data to shift register" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
RESET	Input	BOOL	I, Q, M, D, L	Reset input
S_DATA	Input	POINTER	I, Q, M, D	Points to the source data element to be shifted to the start address (START). For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
START	Input	POINTER	I, Q, M, D	Points to the start address of the register. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
LEN	Input	WORD	I, Q, M, D, L, P	Number of elements to be shifted.

Parameter	Declaration	Data type	Memory area	Description
E_TYPE	Input	BYTE	I, Q, M, D, L, P	Specifies the data type of the elements to be shifted. The following data types are permitted: • W#16#04 = WORD • B#16#05 = INT • B#16#06 = DWORD • B#16#07 = DINT • B#16#08 = REAL
Q	Output	BOOL	Q, M, D, L	Is set to "0" if RESET parameter is active (1) or all elements to be shifted have the value "0".

Example

In the following example four elements of the data type WORD are shifted starting from the address specified on the START parameter. When the first element of the register has been shifted to the next address, the first address is filled with the data that the pointer references on the S_DATA parameter. The value of the last element in the register is lost in the process.



The following tables show how the instruction works using specific values.

Before processing

In this example, the following values are used for the input and output parameters:

Parameter	Operand	Value
RESET	Tag_Input_Reset	0
S_DATA	Tag_Input_DataSource	P#DB2.DBX6.0 DW0 = W#16#1111 DW2 = W#16#2222 DW4 = W#16#3333 DW6 = W#16#4444
START	Tag_Input_StartAddress	P#DB1.DBX0.0 DW6 = W#16#1234
LEN	Tag_Input_Length	W#16#04
E_TYPE	Tag_Input_ElementType	B#16#04
Q	Tag_Output_Q	1

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
RESET	Tag_Input_Reset	0
S_DATA	Tag_Input_DataSource	P#DB2.DBX6.0 DW0 = W#16#1234 DW2 = W#16#1111 DW4 = W#16#2222 DW6 = W#16#3333
START	Tag_Input_StartAddress	P#DB1.DBX0.0 DW6 = W#16#1234
LEN	Tag_Input_Length	W#16#04
E_TYPE	Tag_Input_ElementType	B#16#04
Q	Tag_Output_Q	1

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

SHRB: Shift bit to shift register

Description

The "Shift bit to shift register" instruction is used to write a bit from a specified source (DATA) to a shift register. Each time the instruction is executed with a value "0" on the RESET parameter, new data are read from the source and shifted to the start address (S_BIT) of the shift register. All following bits are shifted by one. The bit contained at the last address (S_BIT + N) is lost after the shift. If the RESET parameter is set to "1" during the processing of the instruction, the addresses in the shift register are reset to "0" and not shifted.

Parameters

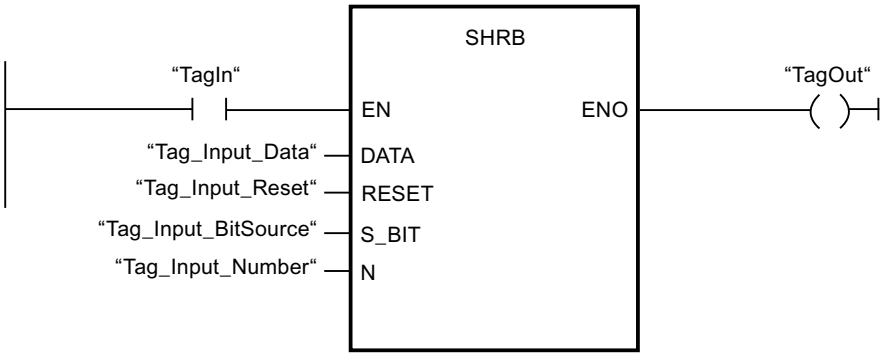
The following table shows the parameters of the "Shift bit to shift register" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
DATA	Input	BOOL	I, Q, M, D, L	Source bit
RESET	Input	BOOL	I, Q, M, D, L	Reset input
S_BIT	Input	POINTER	I, Q, M, D	Points to the start bit of the shift register For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	WORD	I, Q, M, D, L, P	Number of bits to be shifted.

Example

In the following example the N parameter is set to the value 14 ("E" in hexadecimal). As a result 14 bits starting from the address specified on the S_BIT parameter are shifted. After the

bits have been shifted, the first address is filled with the data from the source on the DATA parameter. The last bit value is lost.



The following table shows how the instruction works using specific operand values:

Before processing

In this example the following values are used for the input and output parameters:

Parameter	Operand	Value
RESET	Tag_Input_Reset	0
DATA	Tag_Input_Data	0
S_BIT	Tag_Input_BitSource	P#DB1.DBX1.4
N	Tag_Input_Number	B#16#0E

Before processing the instruction the following signal states are stored in the 14 bits of the register:

1.7	...		S_BIT			...	1.0
0	0	1	1	1	0	1	0

2.7	...					...	2.0
0	0	1	1	1	0	1	0

3.7	...					...	3.0
0	0	1	1	1	0	1	0

After processing

After the processing of the instruction the following signal states are stored in the 14 bits of the register:

1.7	...		S_BIT			...	1.0
0	1	1	0	1	0	1	0

2.7	...					...	2.0
0	1	1	1	0	1	0	0

3.7	...					...	3.0
0	0	1	1	1	0	0	0

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

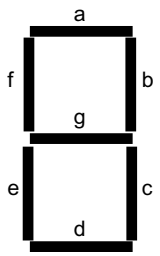
SEG: Create bit pattern for seven-segment display

Description

The instruction "Create bit pattern for seven-segment display" is used to convert each of the four hexadecimal digits of the specified source word (IN) into an equivalent bit pattern for a seven-segment display. The result of the instruction is output in the double word on the OUT parameter.

The following relation exists between the hexadecimal digits and the assignment of the 7 segments (a, b, c, d, e, f, g):

Input digit (Binary)	Assignment of the segments - g f e d c b a	Display (Hexadecimal)	Seven-segment display
0000	00111111	0	
0001	00000110	1	
0010	01011011	2	
0011	01001111	3	
0100	01100110	4	
0101	01101101	5	
0110	01111101	6	



0111	00000111	7
1000	01111111	8
1001	01100111	9
1010	01110111	A
1011	01111100	B
1100	00111001	C
1101	01011110	D
1110	01111001	E
1111	01110001	F

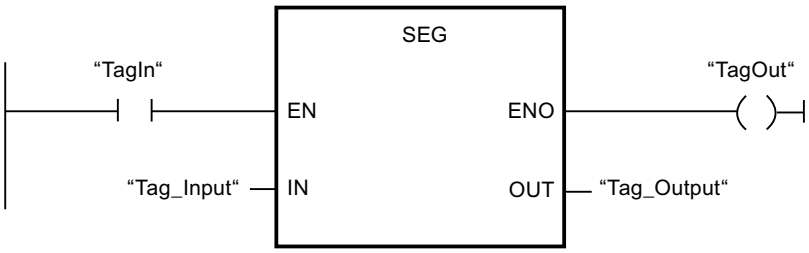
Parameter

The following table shows the parameters of the "Create bit pattern for seven-segment display" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	WORD	I, Q, M, D, L, P or constant	Source word with four hexadecimal digits
OUT	Output	DWORD	I, Q, M, D, L, P	Bit pattern for the seven-segment display

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
Hexadecimal	Binary		
IN	Tag_Input	W#16#1234	0001 0010 0011 0100
OUT	Tag_Output	DW#16065B4F66	000 00110 0101 1011 0100 1111 0110 0110 Display: 1234

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

BCDCPL: Create tens complement

Description

The "Create tens complement" instruction is used to create the tens complement of a seven-digit BCD number specified by the IN parameter. This instruction uses the following mathematical formula to calculate:

10000000 (as BCD)

– 7-digit BCD value

Tens complement (as BCD)

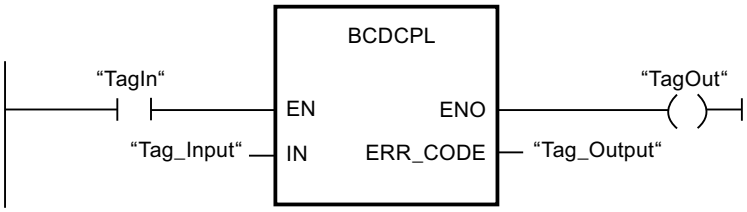
Parameters

The following table shows the parameters of the "Create tens complement" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	DWORD	I, Q, M, D, L, P or constant	7-digit BCD number
ERR_CODE	Output	DWORD	I, Q, M, D, L, P	Result of the instruction

Example

The following example shows how the instruction works:



The following table shows how the instruction functions using specific values:

Parameter	Operand	Value*
IN	Tag_Input	DW#16#01234567
ERR_CODE	Tag_Output	DW#16#08765433
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".		

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Switching display formats in the program status (Page 2980)

BITSUM: Count number of set bits

Description

The "Count number of set bits" instruction is used to count the number of bits of an operand that is set to the signal state "1". The operand whose bits are to be counted is specified on the IN parameter. The result of the instruction is output on the RET_VAL parameter.

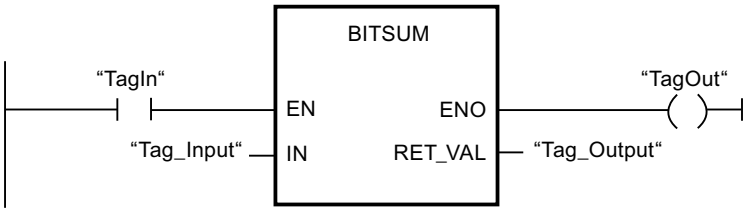
Parameters

The following table shows the parameters of the "Count number of set bits" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	DWORD	I, Q, M, D, L, P or constant	Operand whose set bits are counted
RET_VAL	Output	INT	I, Q, M, D, L, P	Number of set bits

Example

The following example shows how the instruction works:



The following table shows how the instruction functions using specific values:

Parameter	Operand	Value*
IN	Tag_Input	DW#16#12345678
RET_VAL	Tag_Output	W#16#000D (13 Bits)
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".		

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting LAD elements (Page 2579)

Editing LAD elements (Page 2594)

Inserting operands into LAD instructions (Page 2601)

Switching display formats in the program status (Page 2980)

FBD

Bit logic operations

&: AND logic operation

Description

You can use the instruction "AND logic operation" to query the signal states of two or more specified operands and evaluate them according to the AND truth table.

If the signal state of all the operands is "1", then the condition is fulfilled and the instruction returns the result "1". If the signal state of one of the operands is "0", then the condition is not fulfilled and the instruction generates the result "0".

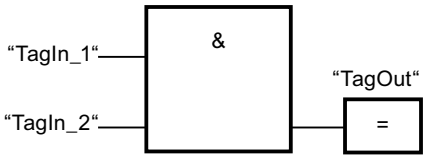
Parameters

The following table shows the parameters of the instruction "AND logic operation":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L	The operand indicates the bit whose signal state will be queried.

Example

The following example shows how the instruction works:



Output "TagOut" is set, when the signal state of the operands "TagIn_1" and "TagIn_2" is "1" and reset when the state of the operands "TagIn_1" and "TagIn_2" is "0".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Example of controlling room temperature (Page 3132)

Example of detecting the fill level of a storage area (Page 3127)

Example of detecting the direction of a conveyor belt (Page 3126)

Setting the status bit for bit logic operations (Page 2455)

Basic information on the status word (Page 2453)

AND truth table (Page 5583)

Insert input (Page 5586)

AND truth table

Results of the logic operation

The following table shows the results that arise from the AND logic operation of two operands:

Signal state of the first operand	Signal state of the second operand	Result of the logic operation
1	1	1
0	1	0
1	0	0
0	0	0

See also

&: AND logic operation (Page 5581)

>=1: OR logic operation

Description

You can use the instruction "OR logic operation" to query the signal states of two or more specified operands and evaluate them according to the OR truth table.

If the signal state of one of the operands is "1", then the condition is fulfilled and the instruction returns the result "1". If the signal state of all the operands is "0", then the condition is not fulfilled and the instruction generates the result "0".

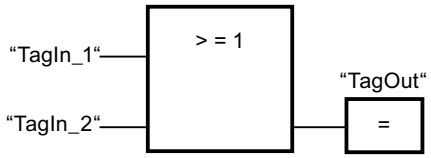
Parameters

The following table shows the parameters of the instruction "OR logic operation":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L	The operand indicates the bit whose signal state will be queried.

Example

The following example shows how the instruction works:



Output "TagOut" is set, when the signal state of the operands "TagIn_1" or "TagIn_2" is "1".

See also

Overview of the valid data types (Page 2236)
Inserting FBD elements (Page 2621)
Editing FBD elements (Page 2635)
Inserting operands in FBD instructions (Page 2643)
Setting the status bit for bit logic operations (Page 2455)
Basic information on the status word (Page 2453)
OR truth table (Page 5584)
Insert input (Page 5586)

OR truth table

Results of the logic operation

The following table shows the results that arise from the OR logic operation of two operands:

Signal state of the first operand	Signal state of the second operand	Result of the logic operation
1	0	1
0	1	1
1	1	1
0	0	0

See also

>=1: OR logic operation (Page 5583)

X: EXCLUSIVE OR logic operation

Description

You can use the "EXCLUSIVE OR logic operation" instruction to query the result of a signal state query according to the EXCLUSIVE OR truth table.

With an "EXCLUSIVE OR logic operation" instruction, the signal state is "1" when the signal state of one of the two specified operands is "1". When more than two operands are queried, the common RLO is "1" if an odd number of the queried operands returns the result "1".

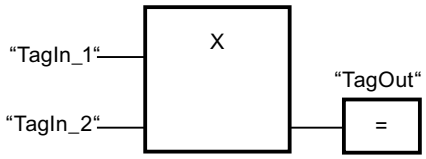
Parameter

The following table shows the parameters of the instruction "EXCLUSIVE OR logic operation":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L	The operand indicates the bit whose signal state will be queried.

Example

The following example shows how the instruction works:



Output "TagOut" is set when the signal state of the operands "TagIn_1" and "TagIn_2" is "1". When both operands return the signal state "1" or "0", the output "TagOut" is reset.

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Setting the status bit for bit logic operations (Page 2455)

Basic information on the status word (Page 2453)

EXCLUSIVE OR truth table (Page 5586)

Insert input (Page 5586)

EXCLUSIVE OR truth table

Results of the logic operation

The following table shows the results that arise from the EXCLUSIVE OR logic operation of two operands:

Signal state of the first operand	Signal state of the second operand	Result of the logic operation
1	0	1
0	1	1
1	1	0
0	0	0

The following table shows the results that arise from the EXCLUSIVE OR logic operation of three operands:

Signal state of the first operand	Signal state of the second operand	Signal state of the third operand	Result of the logic operation
1	0	0	1
0	1	1	0
0	1	0	1
1	0	1	0
0	0	1	1
1	1	0	0
1	1	1	1
0	0	0	0

See also

X: EXCLUSIVE OR logic operation (Page 5585)

Insert input

Description

The "Insert input" instruction is used to add an input to the box of one of the following instructions:

- "AND logic operation"
- "OR logic operation"
- "EXCLUSIVE OR logic operation"

You can query the signal state of several operands by the extension of an instruction box.

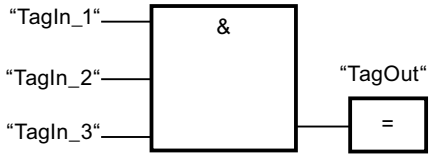
Parameter

The following table shows the parameters of the "Insert input" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L	The operand indicates the bit whose signal state will be queried.

Example

The following example shows how the instruction works:



The box of the "AND logic operation" instruction was extended by an additional input at which the signal state of the operand "TagIn_3" is queried. The "TagOut" output is set, when the operands "TagIn_1" and "TagIn_2" and "TagIn_3" return signal state "1".

See also

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Setting the status bit for bit logic operations (Page 2455)

Basic information on the status word (Page 2453)

&: AND logic operation (Page 5581)

>=1: OR logic operation (Page 5583)

X: EXCLUSIVE OR logic operation (Page 5585)

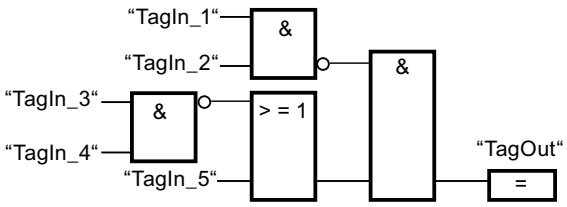
Invert RLO

Description

You use the "Invert RLO" instruction to invert the signal state of the result of logic operation (RLO).

Example

The following example shows how the instruction works:



The output "TagOut" is set when the following conditions are fulfilled:

- The input "TagIn_1" and/or "TagIn_2" has signal state "0".
- The input "TagIn_3" and/or "TagIn_4" has signal state "0" or the input "TagIn_5" has signal state "1".

See also

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Example of controlling room temperature (Page 3132)

Example of detecting the fill level of a storage area (Page 3127)

Example of detecting the direction of a conveyor belt (Page 3126)

Setting the status bit for bit logic operations (Page 2455)

Basic information on the status word (Page 2453)

=: Assignment

Description

You can use the instruction "Assignment" to set the bit of a specified operand. If the result of logic operation (RLO) at the box input has the signal state "1", the specified operand is set to signal state "1". If the signal state at the box input is "0", the bit of the specified operand is reset to "0".

The instruction does not influence the RLO. The RLO at the box input is assigned directly to the operand above the assignment box.

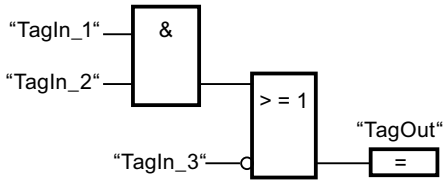
Parameters

The following table shows the parameters of the instruction "Assignment":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand to which the RLO is assigned.

Example

The following example shows how the instruction works:



The operand "TagOut" is set at the output of the "Assignment" instruction when one of the following conditions is fulfilled:

- The inputs "TagIn_1" and "TagIn_2" have the signal state "1".
- The signal state at the input "TagIn_3" is "0".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Example of controlling room temperature (Page 3132)

Example of detecting the fill level of a storage area (Page 3127)

Master control relay (Page 2471)

Setting the status bit for bit logic operations (Page 2455)

Basic information on the status word (Page 2453)

MCR<: Open MCR ranges (Page 5739)

R: Reset output

Description

You can use the "Reset output" instruction to reset the signal state of a specified operand to "0".

The instruction is only executed if the result of logic operation (RLO) at the box input is "1". If the box input has the signal state "1", the specified operand is reset to "0". If there is an RLO of "0" at the box input, the signal state of the specified operand remains unchanged.

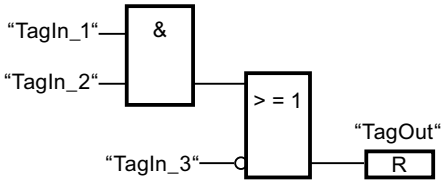
Parameters

The following table shows the parameters of the "Reset output" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, T, C, D, L	Operand that is reset if RLO = "1".

Example

The following example shows how the instruction works:



Operand "TagOut" is reset when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The signal state of the operand "TagIn_3" is "0".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Example of detecting the direction of a conveyor belt (Page 3126)

Master control relay (Page 2471)

Setting the status bit for bit logic operations (Page 2455)

Basic information on the status word (Page 2453)

MCR<: Open MCR ranges (Page 5739)

S: Set output

Description

You can use the instruction "Set output" to set the signal state of a specified operand to "1".

The instruction is only executed if the result of logic operation (RLO) at the box input is "1". If the box input has the signal state "1", the specified operand is set to "1". If there is an RLO of "0" at the box input, the signal state of the specified operand remains unchanged.

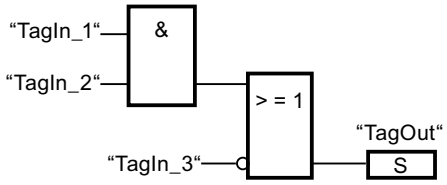
Parameters

The following table shows the parameters of the instruction "Set output":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand which is set with RLO = "1".

Example

The following example shows how the instruction works:



The "TagOut" operand is set when one of the following conditions is fulfilled:

- The operands "TagIn_1" and "TagIn_2" have signal state "1".
- The operand "TagIn_3" has the signal state "0".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Example of detecting the direction of a conveyor belt (Page 3126)

Master control relay (Page 2471)

Setting the status bit for bit logic operations (Page 2455)

Basic information on the status word (Page 2453)

MCR<: Open MCR ranges (Page 5739)

SR: Set/reset flip-flop

Description

You can use the "Set/reset flip-flop" instruction to set or reset the bit of a specified operand based on the signal state of the inputs S and R1. If the signal state is "1" at input S and "0" at input R1, the specified operand is set to "1". If the signal state is "0" at input S and "1" at input R1, the specified operand will be reset to "0".

Input R1 takes priority over input S. When the signal state is "1" on both inputs S and R1, the signal state of the specified operand is reset to "0".

The instruction is not executed if the signal state at the two inputs S and R1 is "0". The signal state of the operand then remains unchanged.

The current signal state of the operand is transferred to output Q and can be queried there.

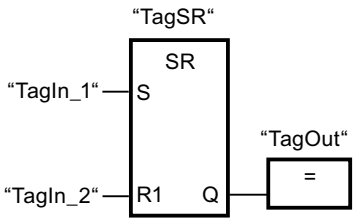
Parameter

The following table shows the parameters of the "Set/reset flip-flop" instruction:

Parameter	Declaration	Data type	Memory area	Description
S	Input	BOOL	I, Q, M, D, L, T, C	Enable setting
R1	Input	BOOL	I, Q, M, D, L, T, C	Enable resetting
<Operand>	InOut	BOOL	I, Q, M, D, L	Operand that is set or reset
Q	Output	BOOL	I, Q, M, D, L	Signal state of the operand

Example

The following example shows how the instruction works:



The operands "TagSR" and "TagOut" are set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The operand "TagIn_2" has the signal state "0".

The operands "TagSR" and "TagOut" are reset when one of the following conditions is fulfilled:

- The operand "TagIn_1" has signal state "0" and the operand "TagIn_2" has signal state "1".
- The operands "TagIn_1" and "TagIn_2" have signal state "1".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Master control relay (Page 2471)

Setting the status bit for bit logic operations (Page 2455)

Basic information on the status word (Page 2453)

MCR<: Open MCR ranges (Page 5739)

RS: Reset/set flip-flop

Description

You can use the "Reset/set flip-flop" instruction to reset or set the bit of a specified operand based on the signal state of the inputs R and S1. If the signal state is "1" at input R and "0" at input S1, the specified operand will be reset to "0". If the signal state is "0" at input R and "1" at input S1, the specified operand is set to "1".

Input S1 takes priority over input R. When the signal state is "1" at both inputs R and S1, the signal state of the specified operand is set to "1".

The instruction is not executed if the signal state at the two inputs R and S1 is "0". The signal state of the operand then remains unchanged.

The current signal state of the operand is transferred to output Q and can be queried there.

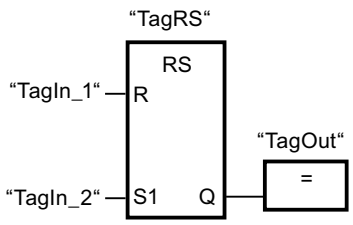
Parameter

The following table shows the parameters of the "Reset/set flip-flop" instruction:

Parameter	Declaration	Data type	Memory area	Description
R	Input	BOOL	I, Q, M, T, C, D, L	Enable resetting
S1	Input	BOOL	I, Q, M, T, C, D, L	Enable setting
<Operand>	InOut	BOOL	I, Q, M, D, L	Operand that is reset or set
Q	Output	BOOL	I, Q, M, D, L	Signal state of the operand

Example

The following example shows how the instruction works:



The operands "TagRS" and "TagOut" are reset when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The operand "TagIn_2" has the signal state "0".

The operands "TagRS" and "TagOut" are set when the following conditions are fulfilled:

- The operand "TagIn_1" has signal state "0" and the operand "TagIn_2" has signal state "1".
- The operands "TagIn_1" and "TagIn_2" have signal state "1".

See also

Overview of the valid data types (Page 2236)

Master control relay (Page 2471)

Setting the status bit for bit logic operations (Page 2455)

Basic information on the status word (Page 2453)

MCR<: Open MCR ranges (Page 5739)

P: Scan operand for positive signal edge

Description

You can use the "Scan operand for positive signal edge" instruction to determine whether there is a "0" to "1" change in the signal state of a specified operand (<Operand1>). The instruction compares the current signal state of <Operand1> with the signal state of the previous scan, which is saved in an edge memory bit (<Operand2>). If the instruction detects a change in the result of logic operation (RLO) from "0" to "1", there is a positive, rising edge.

If a rising edge is detected, the output of the instruction has the signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Specify the operand to be queried (<Operand1>) in the operand placeholder above the instruction. Specify the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the memory bit is overwritten. This would influence edge evaluation and the result would no longer be unequivocal. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

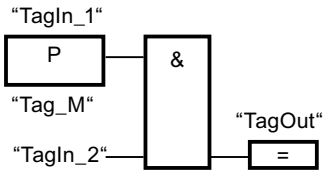
Parameters

The following table shows the parameters of the "Scan operand for positive signal edge" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	BOOL	I, Q, M, T, C, D, L	Signal to be scanned
<Operand2>	InOut	BOOL	I, Q, M, D, L	Edge memory bit in which the signal state of the previous query is saved.

Example

The following example shows how the instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- There is a rising edge at input "TagIn_1".
- The signal state of the operand "TagIn_2" is "1".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Example of detecting the direction of a conveyor belt (Page 3126)

Setting the status bit for bit logic operations (Page 2455)

Basic information on the status word (Page 2453)

N: Scan operand for negative signal edge

Description

You can use the "Scan operand for negative signal edge" instruction to determine whether there is a "1" to "0" change in the signal state of a specified operand (<Operand1>). The instruction compares the current signal state of <Operand1> with the signal state of the

previous scan, which is saved in an edge memory bit (<Operand2>). If the instruction detects a change in the result of logic operation (RLO) from "1" to "0", there is a negative, falling edge.

If a falling edge is detected, the output of the instruction has the signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Specify the operand to be queried (<Operand1>) in the operand placeholder above the instruction. Specify the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the memory bit is overwritten. This would influence edge evaluation and the result would no longer be unequivocal. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

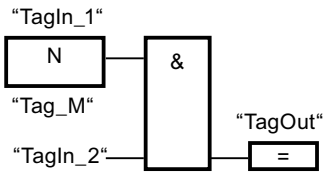
Parameters

The following table shows the parameters of the "Scan operand for negative signal edge" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	BOOL	I, Q, M, T, C, D, L	Signal to be scanned
<Operand2>	InOut	BOOL	I, Q, M, D, L	Edge memory bit in which the signal state of the previous scan is saved.

Example

The following example shows how the instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- There is a falling edge at input "TagIn_1".
- The signal state of the operand "TagIn_2" is "1".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Setting the status bit for bit logic operations (Page 2455)

Basic information on the status word (Page 2453)

P_TRIG: Scan RLO for positive signal edge

Description

Use the instruction "Scan RLO for positive signal edge" to query a "0" to "1" change in the signal state of the result of logic operation (RLO). The instruction compares the current signal state of the RLO with the signal state of the previous query, which is saved in an edge memory bit (<Operand>). If the instruction detects a change in the result of logic operation (RLO) from "0" to "1", there is a positive, rising edge.

If a rising edge is detected, the output of the instruction has the signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the memory bit is overwritten. This would influence edge evaluation and the result would no longer be unequivocal. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

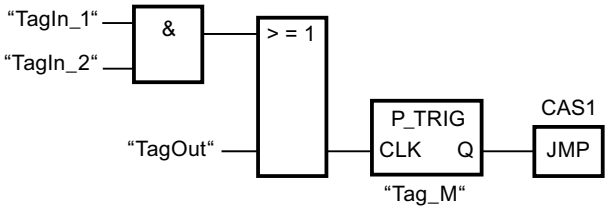
Parameters

The following table shows the parameters of the instruction "Scan RLO for positive signal edge":

Parameter	Declaration	Data type	Memory area	Description
CLK	Input	BOOL	I, Q, M, D, L	Current RLO
<Operand>	InOut	BOOL	M, D	Edge memory bit in which the RLO of the previous query is saved.
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:



The RLO of the preceding bit logic operation is saved in the edge memory bit "Tag_M". If a "0" to "1" change is detected in the signal state of the RLO, the program jumps to jump label CAS1.

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Setting the status bit for bit logic operations (Page 2455)

Basic information on the status word (Page 2453)

N_TRIG: Scan RLO for negative signal edge

Description

Use the instruction "Scan RLO for negative signal edge" to query a "1" to "0" change in the signal state of the result of logic operation (RLO). The instruction compares the current signal state of the RLO with the signal state of the previous query saved in the edge memory bit (<Operand>). If the instruction detects a change in the result of logic operation (RLO) from "1" to "0", there is a negative, falling edge.

If a falling edge is detected, the output of the instruction has the signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the memory bit is overwritten. This would influence edge evaluation and the result would no longer be unequivocal. The memory area of the edge memory bit has to be located in a DB (static area for FB) or in the bit memory area.

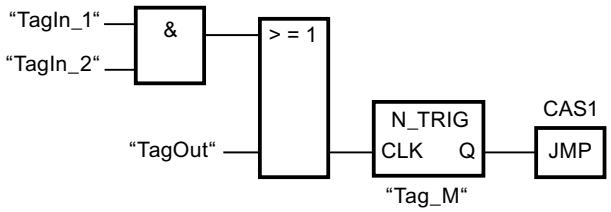
Parameters

The following table shows the parameters of the instruction "Scan RLO for negative signal edge":

Parameter	Declaration	Data type	Memory area	Description
CLK	Input	BOOL	I, Q, M, D, L	Current RLO
<Operand>	InOut	BOOL	M, D	Edge memory bit in which the RLO of the previous query is saved.
Q	Output	BOOL	I, Q, M, D, L	Result of edge evaluation

Example

The following example shows how the instruction works:



The RLO of the preceding bit logic operation is saved in the edge memory bit "Tag_M". If a "1" to "0" change is detected in the signal state of the RLO, the program jumps to jump label CAS1.

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Setting the status bit for bit logic operations (Page 2455)

Basic information on the status word (Page 2453)

Timer operations

IEC Timers

TP: Generate pulse

Description

You can use the "Generate pulse" instruction to set output Q for a programmed duration. The instruction is started when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. Output Q is set for the duration PT, regardless of the subsequent course of the input signal. Even when a new positive signal edge is detected, the signal state of the Q output is not affected as long as the PT duration is running.

You can query the current time value at the ET output. The timer value starts at T#0s and ends when the value of the time duration PT is reached. If the time duration PT is reached and the signal state at input IN is "0", the ET output is reset.

Note

If the timer is not called in the program because it is skipped, for example, output ET returns a constant value as soon as the timer has expired.

Each call of the "Generate pulse" instruction must be assigned an IEC Timer in which the instruction data is stored. You can declare an IEC Timer as follows:

- Declaration of a data block of the type TP (for example, "TP_DB")
- Declaration as a local tag of the type TP in the "Static" section of a block (for example, #MyTP_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens automatically, in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when the instruction is called and also each time the outputs Q or ET are accessed.

The operating system resets the instances of the "Generate pulse" instruction during a cold restart. If you want instances of the instruction to be in the initialized state after a warm restart, call these in a startup OB with the value "0" set for the PT parameter. If instances of the instruction "Generate pulse" are located within another block, you can reset these instances, for example, by initializing the higher-level block.

The execution of the "Generate pulse" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

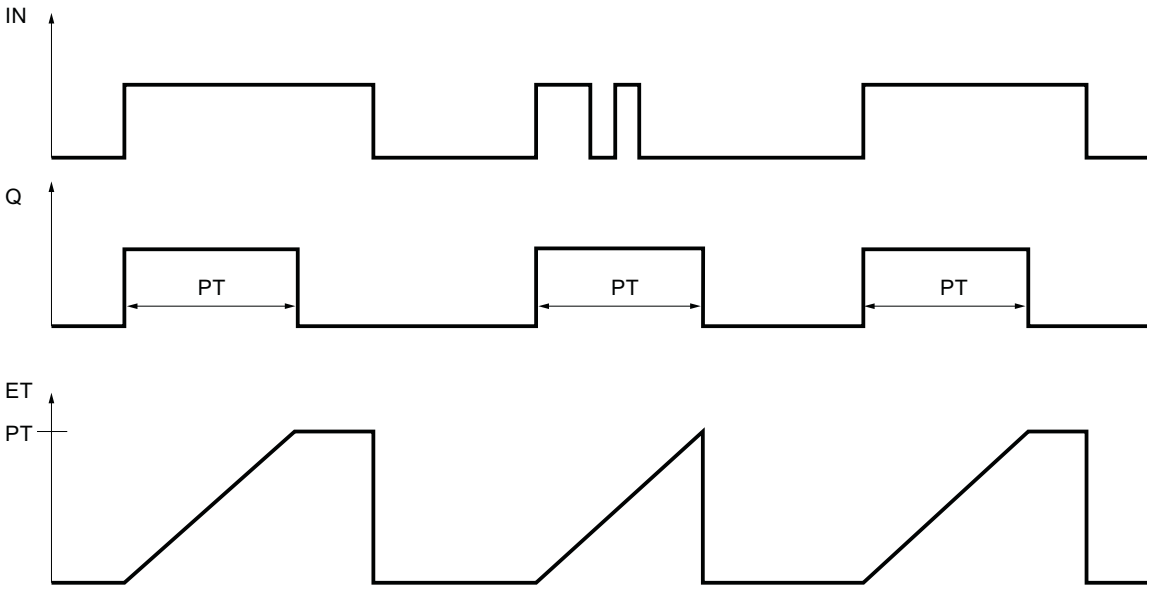
The following table shows the parameters of the "Generate pulse" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L, T, C	Start input
PT	Input	TIME	I, Q, M, D, L, P or constant	Duration of the pulse. The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L	Pulse output
ET	Output	TIME	I, Q, M, D, L	Current timer value

For additional information on valid data types, refer to "See also".

Pulse diagram

The following figure shows the pulse diagram of the "Generate pulse" instruction:



See also

Inserting FBD elements (Page 2621)
Editing FBD elements (Page 2635)
Inserting operands in FBD instructions (Page 2643)
Example of controlling room temperature (Page 3132)
Overview of the valid data types (Page 2236)
Call function blocks as single or multi-instances (Page 2182)

TON: Generate on-delay

Description

You can use the "Generate on-delay" instruction to delay setting of the Q output by the time configured with the PT time. The instruction is started when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. When the duration PT expires, output Q has the signal state "1". Output Q remains set as long as the start input is still "1". When the signal state at the start input changes from "1" to "0", the Q output is reset. The timer function is started again when a new positive signal edge is detected at the start input.

You can query the current time value at the ET output. The timer value starts at T#0s and ends when the value of the time duration PT is reached. The ET output is reset as soon as the signal state at the IN input changes to "0".

Note

If the timer is not called in the program because it is skipped, for example, output ET returns a constant value as soon as the timer has expired.

Each call of the "Generate on-delay" instruction must be assigned an IEC Timer in which the instruction data is stored. You can declare an IEC Timer as follows:

- Declaration of a data block of the type TON (for example, "TON_DB")
- Declaration as a local tag of the type TON in the "Static" section of a block (for example, #MyTON_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens automatically, in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when the instruction is called and also each time the outputs Q or ET are accessed.

The operating system resets the instances of the "Generate on-delay" instruction during a cold restart. If you want instances of the instruction to be in the initialized state after a warm restart, call these in a startup OB with the value "0" set for the PT parameter. If instances of the "Generate on-delay" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

The execution of the "Generate on-delay" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

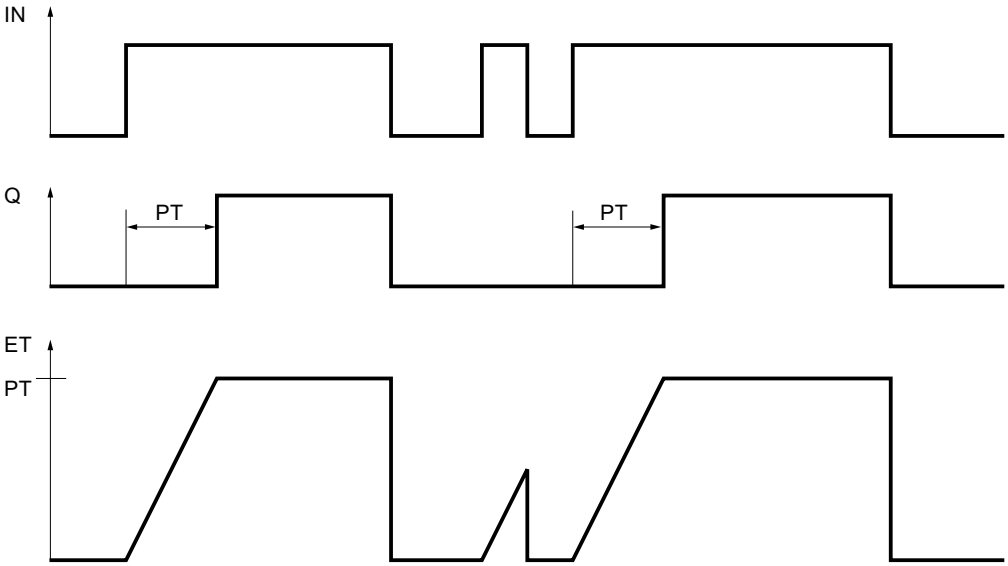
The following table shows the parameters of the "Generate on-delay" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L, T, C	Start input
PT	Input	TIME	I, Q, M, D, L, P or constant	Duration of the on-delay The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L	Output that is set when the time PT expires.
ET	Output	TIME	I, Q, M, D, L	Current timer value

For additional information on valid data types, refer to "See also".

Pulse diagram

The following figure shows the pulse diagram of the "Generate on-delay" instruction:



See also

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Overview of the valid data types (Page 2236)

Call function blocks as single or multi-instances (Page 2182)

TOF: Generate off-delay

Description

You can use the "Generate off-delay" instruction to delay resetting of the Q output by the time configured with the PT time. The output Q is set when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive signal edge). When the signal state at input IN changes back to "0", the programmed time PT starts. Output Q remains set as long as the time duration PT is running. When the PT time duration expires, the Q output is reset. If the signal state at input IN changes to "1" before the PT time duration expires, the timer is reset. The signal state at the output Q will continue to be "1".

You can query the current time value at the ET output. The current timer value starts at T#0s and ends when the value of the time duration PT is reached. When the time duration PT expires, the ET output remains set to the current value until input IN changes back to "1". If input IN changes to "1" before the time duration PT has expired, the ET output is reset to the value T#0s.

Note

If the timer is not called in the program because it is skipped, for example, output ET returns a constant value as soon as the timer has expired.

Each call of the "Generate off-delay" instruction must be assigned an IEC Timer in which the instruction data is stored. You can declare an IEC Timer as follows:

- Declaration of a data block of the type TOF (for example, "TOF_DB")
- Declaration as a local tag of the type TOF in the "Static" section of a block (for example, #MyTOF_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens automatically, in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when the instruction is called and also each time the outputs Q or ET are accessed.

The operating system resets the instances of the "Generate off-delay" instruction during a cold restart. If you want instances of the instruction to be in the initialized state after a warm restart, call these in a startup OB with the value "0" set for the PT parameter. If instances of the "Generate off-delay" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

The execution of the "Generate off-delay" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

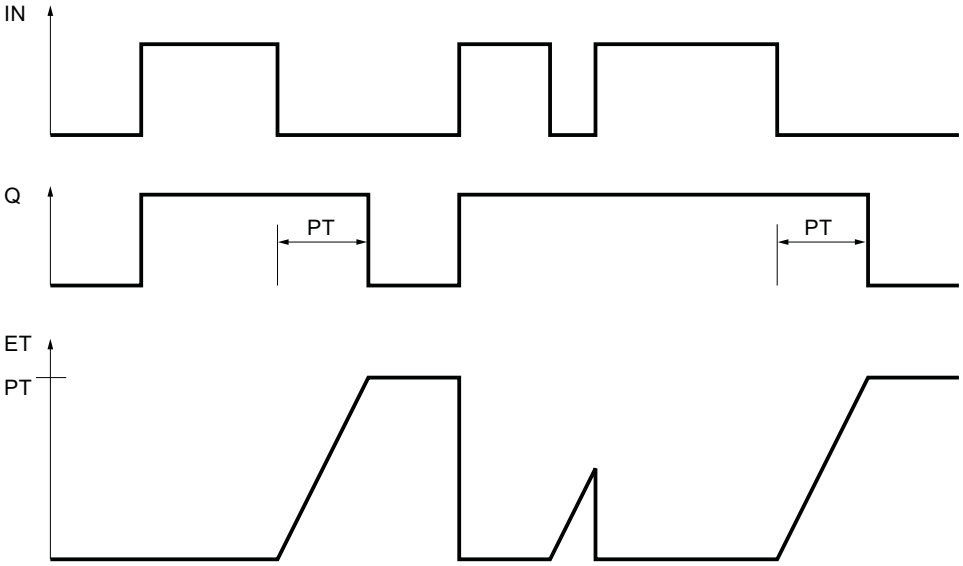
The following table shows the parameters of the "Generate off-delay" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L, T, C	Start input
PT	Input	TIME	I, Q, M, D, L, P or constant	Duration of the off delay The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L	Output that is reset when the time PT expires.
ET	Output	TIME	I, Q, M, D, L	Current timer value

For additional information on valid data types, refer to "See also".

Pulse diagram

The following figure shows the pulse diagram of the "Generate off-delay" instruction:



See also

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Overview of the valid data types (Page 2236)

Call function blocks as single or multi-instances (Page 2182)

SIMATIC Timers

S_PULSE: Assign pulse timer parameters and start

Description

The "Assign pulse timer parameters and start" instruction starts a programmed timer when a transition from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV) as long as the signal state at input S is "1". If the signal state at input S changes to "0" before the programmed duration expires, the timer is stopped. In this case, the signal state at output Q is "0".

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value. The current time value is output binary-coded at output BI and BCD-coded at output BCD.

If the timer is running and the signal state at input R changes to "1", the current time value and the time base are also set to zero. If the timer is not running, the signal state "1" at the R input has no effect.

The "Assign pulse timer parameters and start" instruction needs a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

The instruction data is updated with each access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be structured under: See also "L: Load timer value".

Parameters

The following table shows the parameters of the "Assign pulse timer parameters and start" instruction:

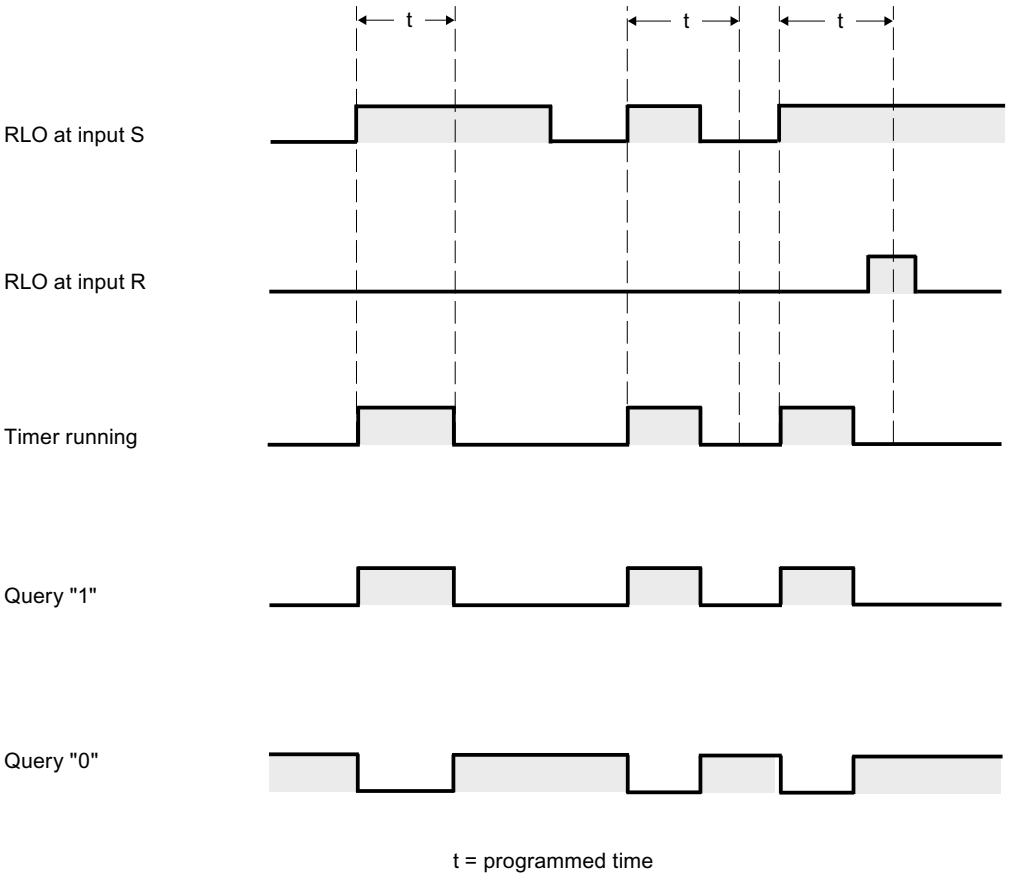
Parameter	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, T, C, D, L, P	Start input

Parameter	Declaration	Data type	Memory area	Description
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
R	Input	BOOL	I, Q, M, T, C, D, L, P	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (binary-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer

For additional information on valid data types, refer to "See also".

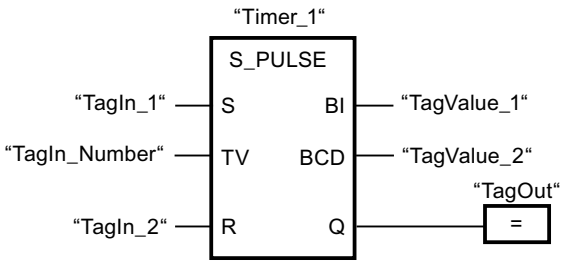
Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign pulse timer parameters and start" instruction:



Example

The following example shows how the instruction works:



"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer runs for the time value of the "TagIn_Number" operand as long as the "TagIn_1" operand has the signal state "1". If the signal state of the operand "TagIn_1" changes from "1" to "0" before the timer expires, the timer "Timer_1" is stopped. The "TagOut" operand is reset to signal state "0".

The "TagOut" operand has the signal state "1" as long as the timer is running and the "TagIn_1" operand has the signal state "1". When the time has expired or is reset, the "TagOut" operand is reset to "0".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

L: Load timer value (Page 5934)

S_PEXT: Assign extended pulse timer parameters and start

Description

The "Assign extended pulse timer parameters and start" instruction starts a programmed timer when a transition from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV), even if the signal state at input S changes to "0". As long as the timer is running, output Q has the signal state "1". When the timer has expired, output Q is reset to "0". If the signal state at input S changes from "0" to "1" while the timer is running, the timer is restarted with the duration programmed at input TV.

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value. The current time value is output binary-coded at output BI and BCD-coded at output BCD.

If the timer is running and the signal state at input R changes to "1", the current time value and the time base are also set to zero. If the timer is not running, the signal state "1" at the R input has no effect.

The "Assign extended pulse timer parameters and start" instruction needs a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

The instruction data is updated with each access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be structured under: See also "L: Load timer value".

Parameters

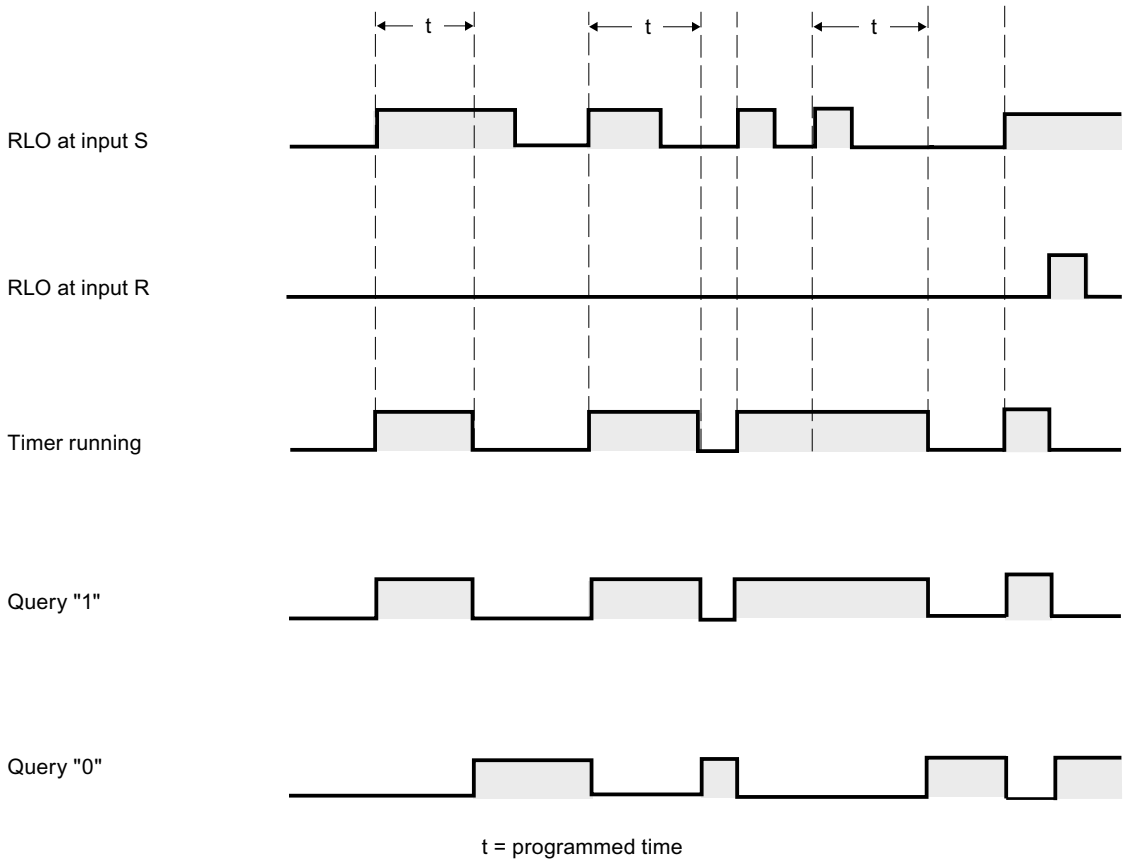
The following table shows the parameters of the "Assign extended pulse timer parameters and start" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
R	Input	BOOL	I, Q, M, T, C, D, L, P	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (binary-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer

For additional information on valid data types, refer to "See also".

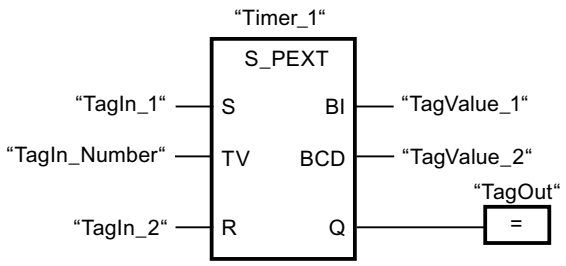
Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign extended pulse timer parameters and start" instruction:



Example

The following example shows how the instruction works:



"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer runs for the time value of the "TagIn_Number" operand without being affected by a negative edge at the S input. If the signal state at the "TagIn_1" operand changes from "0" to "1" before the timer expires, the timer is restarted.

The "TagOut" operand has the signal state "1" as long as the timer is running. When the time has expired or is reset, the "TagOut" operand is reset to "0".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

L: Load timer value (Page 5934)

S_ODT: Assign on-delay timer parameters and start

Description

The "Assign on-delay timer parameters and start" instruction starts a programmed timer when a transition from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV) as long as the signal state at input S is "1". If the timer expires correctly and input S still has signal state "1", output Q returns signal state "1". If the signal state at input S changes from "1" to "0" while the timer is running, the timer is stopped. In this case, output Q is reset to signal state "0".

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value. The current time value is output binary-coded at output BI and BCD-coded at output BCD.

If the time is running and the signal state at input R changes from "0" to "1", the current time value and the time base are also set to zero. In this case, the signal state at output Q is "0". The timer is reset if the signal state is "1" at the R input even if the timer is not running and the RLO at input S is "1".

The "Assign on-delay timer parameters and start" instruction needs a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

The instruction data is updated with each access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be structured under: See also "L: Load timer value".

Parameters

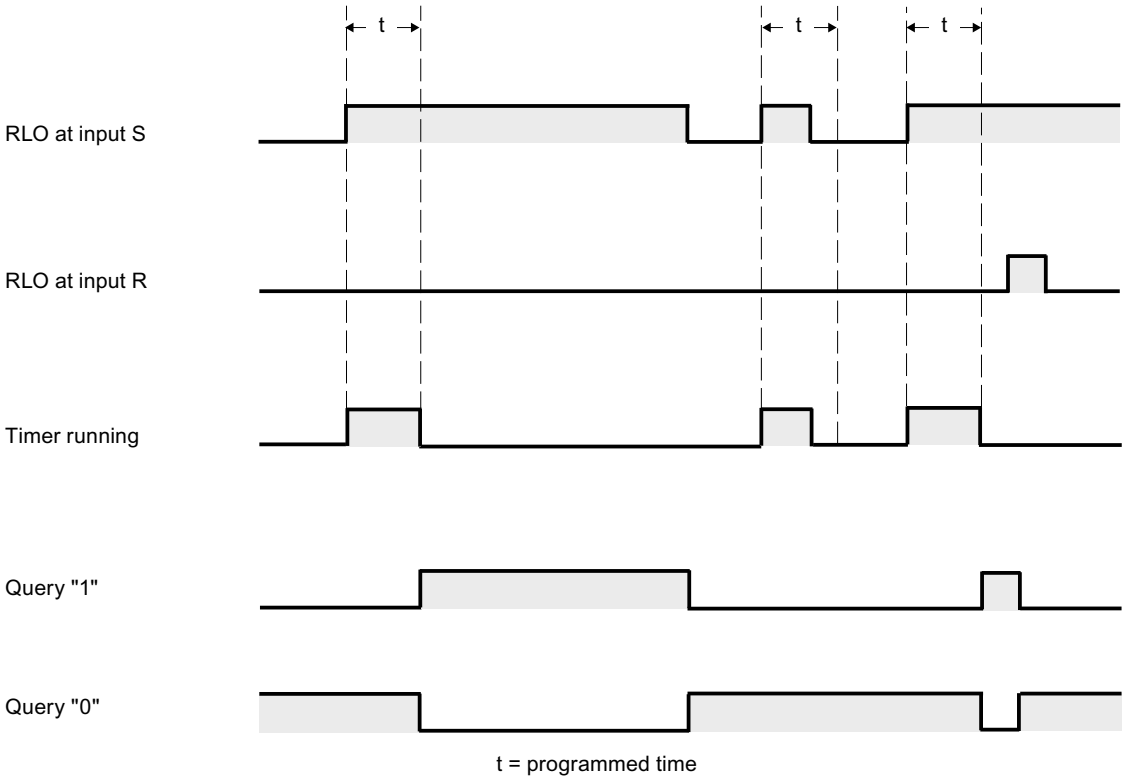
The following table shows the parameters of the "Assign on-delay timer parameters and start" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
R	Input	BOOL	I, Q, M, T, C, D, L, P	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (binary-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer

For additional information on valid data types, refer to "See also".

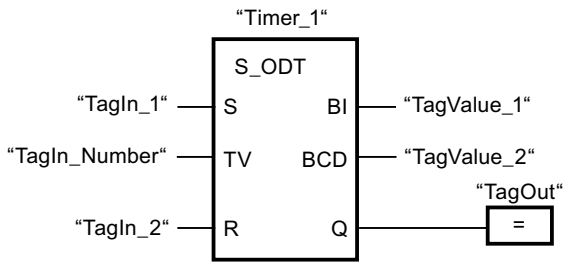
Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign on-delay timer parameters and start" instruction:



Example

The following example shows how the instruction works:



"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer expires with the value of operand "TagIn_Number". If the timer expires and the signal state of the operand is "1", the "TagOut" operand is set to "1". If the signal state at the "TagIn_1" operand changes from "1" to "0" before the timer expires, the timer is stopped. The "TagOut" operand has the signal state "0".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

L: Load timer value (Page 5934)

S_ODTS: Assign retentive on-delay timer parameters and start

Description

The "Assign retentive on-delay timer parameters and start" instruction starts a programmed timer when a transition from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV), even if the signal state at input S changes to "0". If the timer expires, the "Q" output returns signal state "1" regardless of the signal state at input "S". If the signal state at input S changes from "0" to "1" while the timer is running, the timer is restarted with the duration programmed at input (TV).

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value. The current time value is output binary-coded at output BI and BCD-coded at output BCD.

Signal state "1" at input R resets the current time value and time base to "0" regardless of the signal state at start input S. In this case, the signal state at output Q is "0".

The "Assign retentive on-delay timer parameters and start" instruction needs a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

The instruction data is updated with each access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be structured under: See also "L: Load timer value".

Parameters

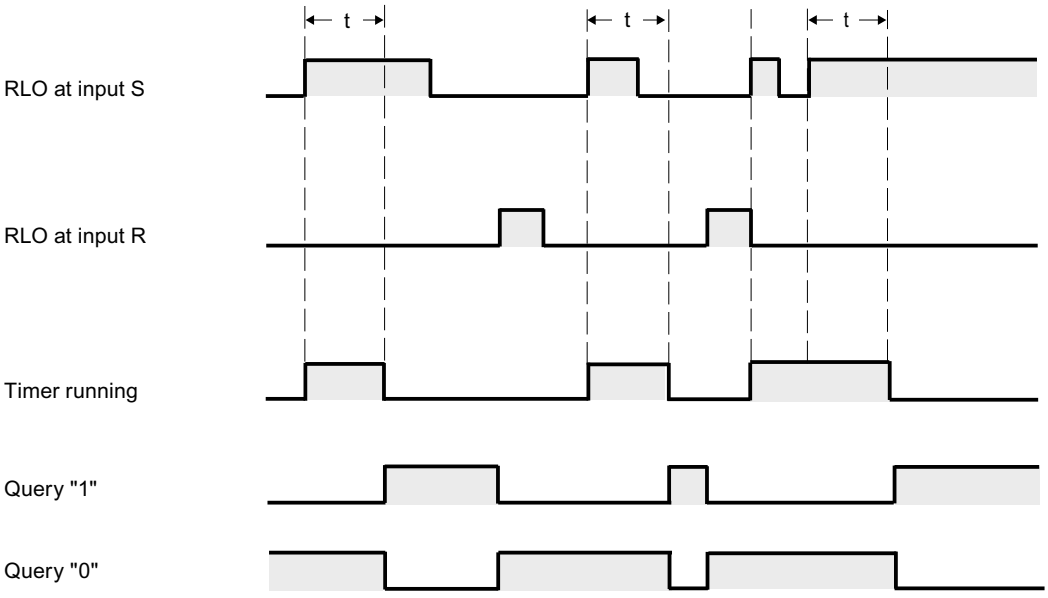
The following table shows the parameters of the "Assign retentive on-delay timer parameters and start" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
R	Input	BOOL	I, Q, M, T, C, D, L, P	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (binary-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer

For additional information on valid data types, refer to "See also".

Pulse timing diagram

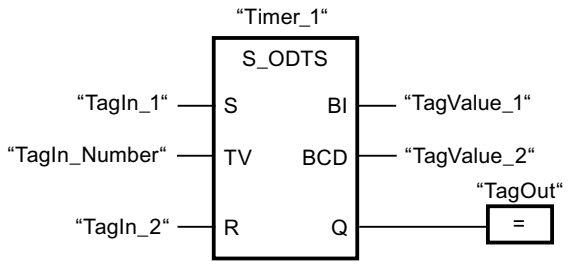
The following figure shows the pulse timing diagram of the "Assign retentive on-delay timer parameters and start" instruction:



t = programmed time

Example

The following example shows how the instruction works:



"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer expires with the time value of the "TagIn_Number" operand, even when the signal state of the "TagIn_1" operand changes to "0". When the time expires, the "TagOut" operand is reset to "1". If the signal state at the "TagIn_1" operand changes from "0" to "1" while the timer is running, the timer is restarted.

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

L: Load timer value (Page 5934)

S_OFFDT: Assign off-delay timer parameters and start

Description

The "Assign off-delay timer parameters and start" instruction starts a programmed timer when a transition from "1" to "0" (negative signal edge) is detected in the result of logic operation (RLO) at input S. The timer expires with the programmed duration (TV). As long as the timer is running or input S returns signal state "1", output Q has signal state "1". If the timer expires and the signal state is "0", output Q is reset to signal state "0". If the signal state at input S changes from "0" to "1" while the timer is running, the timer is reset. The timer is only restarted after a falling signal edge is detected at input S.

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value. The current time value is output binary-coded at output BI and BCD-coded at output BCD.

Signal state "1" at input R resets the current time value and time base to "0". In this case, the signal state at output Q is "0".

The "Assign off-delay timer parameters and start" instruction needs a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

The instruction data is updated with each access. It can therefore happen that the query of the data at the start of the cycle returns different values than at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be structured under: See also "L: Load timer value".

Parameters

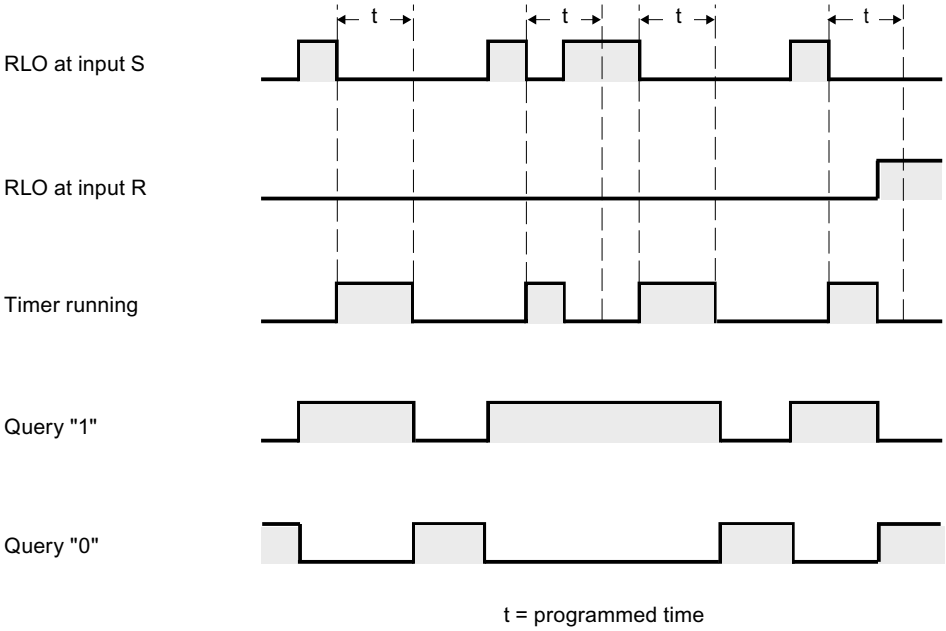
The following table shows the parameters of the "Assign off-delay timer parameters and start" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Timer>	InOut/Input	TIMER	T	Time of the instruction The number of timers depends on the CPU.
S	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
R	Input	BOOL	I, Q, M, T, C, D, L, P	Reset input
BI	Output	WORD	I, Q, M, D, L, P	Current time value (binary-coded)
BCD	Output	WORD	I, Q, M, D, L, P	Current time value (BCD format)
Q	Output	BOOL	I, Q, M, D, L, P	Status of the timer

For additional information on valid data types, refer to "See also".

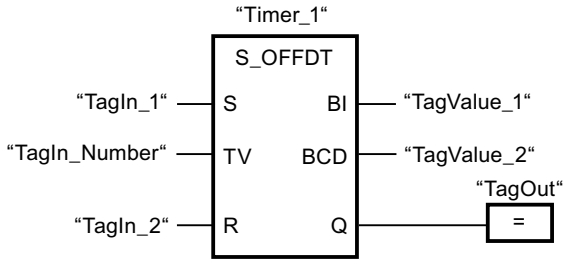
Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign off-delay timer parameters and start" instruction:



Example

The following example shows how the instruction works:



"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "1" to "0". The timer expires with the value of operand "TagIn_Number". The "TagOut" operand is set to "1" if the timer is running or the "TagIn_1" operand has the signal state "0". If the signal state at the "TagIn_1" operand changes from "0" to "1" while the timer is running, the timer is reset.

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

L: Load timer value (Page 5934)

SP: Start pulse timer

Description

The instruction "Start pulse timer" starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at the start input. The timer runs with the specified duration as long as the RLO has the signal state "1". As long as the timer is running, the query of timer status "1" returns the signal state "1". If there is a change from "1" to "0" in the RLO before the time value has elapsed, the timer stops. In this case, the query for timer status for "1" returns the signal state "0".

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value.

The "Start pulse timer" instruction needs a preceding logic operation for the edge evaluation and can only be placed on the right edge of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be structured under: See also "L: Load timer value".

Parameters

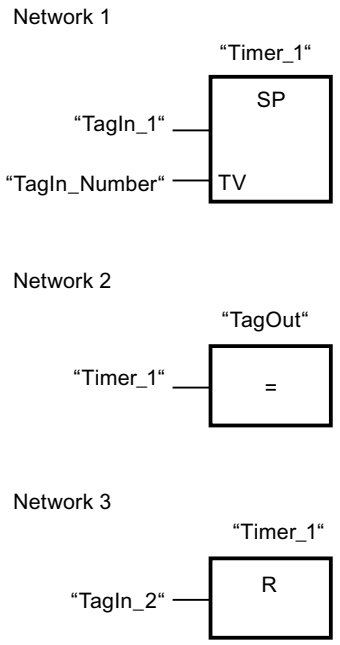
The following table shows the parameters of the instruction "Start pulse timer":

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
<Timer>	Output	TIMER	T	Timer which is started. The number of timers depends on the CPU.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer expires with the time value of the "TagIn_Number" operand as long as the signal state of the "TagIn_1" operand is "1". If the signal state at the "TagIn_1" operand changes from "1" to "0" before the timer expires, the timer is stopped. As long as the timer is running, the "TagOut" operand returns signal state "1". If the signal state of the "TagIn_1" operand changes from "0" to "1", the timer is reset, i.e. the timer is stopped and the current time value is set to "0".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

L: Load timer value (Page 5934)

SE: Start extended pulse timer

Description

The "Start extended pulse timer" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at the start input. The timer runs for the specified time period even when the RLO changes to signal state "0". As long as the timer is running, the query of timer status "1" returns the signal state "1". If the RLO changes from "0" to "1" while the timer is running, the timer is restarted with the

programmed time period. When the timer expires, the query for timer status for "1" returns the signal state "0".

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value.

The "Start extended pulse timer" instruction needs a preceding logic operation for the edge evaluation and can only be placed on the right edge of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be structured under: See also "L: Load timer value".

Parameters

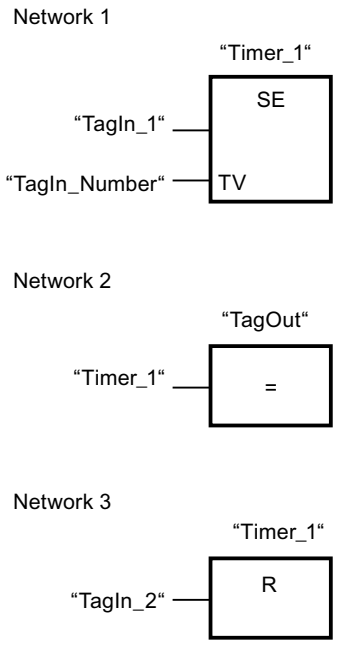
The following table shows the parameters of the instruction "Start extended pulse timer":

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
<Timer>	Output	TIMER	T	Timer which is started. The number of timers depends on the CPU.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer expires with the time value of the "TagIn_Number" operand without being affected by a negative edge at the RLO. As long as the timer is running, the "TagOut" operand returns signal state "1". If the signal state at the "TagIn_1" operand changes from "0" to "1" before the timer expires, the timer is restarted.

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

L: Load timer value (Page 5934)

SD: Start on-delay timer

Description

The "Start on-delay timer" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at the start input. The timer runs for the specified period of time as long as the RLO is "1". If the timer expires and the RLO has the signal state "1", the query timer status "1" returns the signal state "1". If the RLO changes from "1" to "0" while the timer is running, the timer is stopped. In this case, querying the timer status for "1" returns the signal state "0".

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value.

The "Start on-delay timer" instruction needs a preceding logic operation for the edge evaluation and can only be placed on the right edge of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be structured under: See also "L: Load timer value".

Parameters

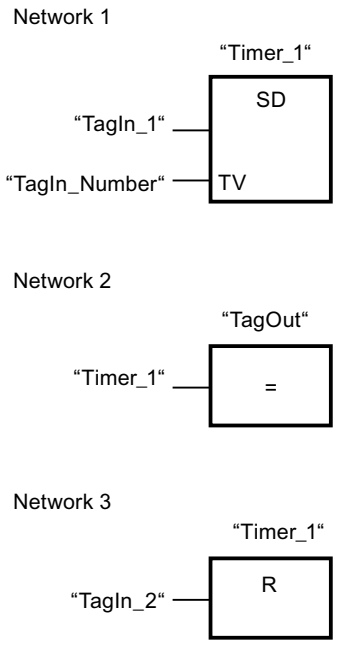
The following table shows the parameters of the instruction "Start on-delay timer":

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
<Timer>	Output	TIMER	T	Timer which is started. The number of timers depends on the CPU.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer expires with the value of operand "TagIn_Number". If the timer expires and the RLO has the signal state "1", the "TagOut" operand is set to "1". If the signal state at the "TagIn_1" operand changes from "1" to "0" before the timer expires, the timer is stopped. If the signal state of the "TagIn_2" operand changes to "1", "Timer_1" is reset, i.e. the timer is stopped and the current time value is set to "0".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

L: Load timer value (Page 5934)

SS: Start retentive on-delay timer

Description

The "Start retentive on-delay timer" instruction starts a programmed timer when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) at the start input. The timer runs for the specified time period even when the RLO changes to signal state "0". When the timer expires, the query for timer status for "1" returns the signal state "1". When the timer expires, the timer can only be restarted if it is explicitly reset.

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value.

The "Start retentive on-delay timer" instruction needs a preceding logic operation for the edge evaluation and can only be placed on the right edge of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be structured under: See also "L: Load timer value".

Parameters

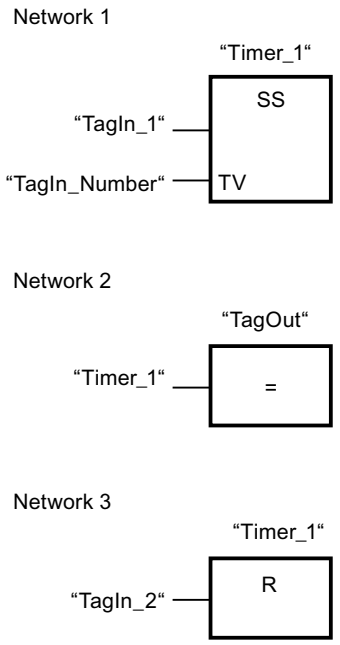
The following table shows the parameters of the instruction "Start retentive on-delay timer":

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
<Timer>	Output	TIMER	T	Timer which is started. The number of timers depends on the CPU.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "0" to "1". The timer expires with the value of operand "TagIn_Number". When the time expires, the "TagOut" operand is reset to "1". If the signal state at the "TagIn_1" operand changes from "0" to "1" while the timer is running, the timer is restarted. If the signal state of the "TagIn_2" operand changes to "1", "Timer_1" is reset, i.e. the timer is stopped and the current time value is set to "0".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

L: Load timer value (Page 5934)

SF: Start off-delay timer

Description

The "Start off-delay timer" instruction starts a programmed timer when a change from "1" to "0" (negative signal edge) is detected in the result of logic operation (RLO) at the start input. The timer runs for the specified time period. As long as the timer is running, the query of timer status "1" returns the signal state "1". If the RLO changes from "0" to "1" while the timer is

running, the timer is reset. The timer is always restarted when the RLO changes from "1" to "0".

The duration is made up internally of a time value and a time base and is programmed at parameter TV. When the instruction is started, the programmed time value is counted down to zero. The time base determines the time period of the time value.

The "Start off-delay timer" instruction needs a preceding logic operation for the edge evaluation and can only be placed on the right edge of the network.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be structured under: See also "L: Load timer value".

Parameters

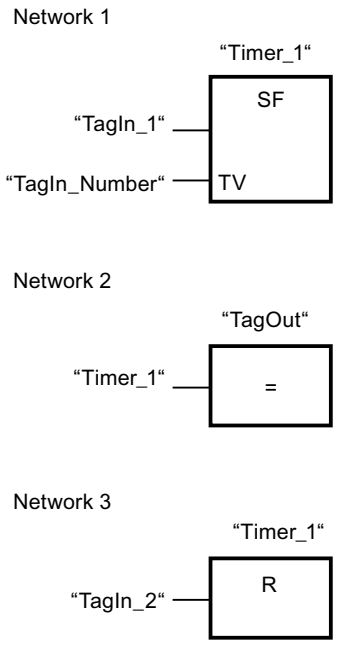
The following table shows the parameters of the instruction "Start off-delay timer":

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L, P	Start input
TV	Input	S5TIME, WORD	I, Q, M, D, L or constant	Time duration
<Timer>	Output	TIMER	T	Timer which is started. The number of timers depends on the CPU.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



"Timer_1" starts when the signal state of the "TagIn_1" operand changes from "1" to "0". The timer expires with the value of operand "TagIn_Number". As long as the timer is running, the "TagOut" operand is set to "1". If the signal state at the "TagIn_1" operand changes from "1" to "0" while the timer is running, the timer is restarted. If the signal state of the "TagIn_2" operand changes to "1", "Timer_1" is reset, i.e. the timer is stopped and the current time value is set to "0".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

L: Load timer value (Page 5934)

Counter operations

IEC Counters

CTU: Count up

Description

You can use the "Count up" instruction to increment the value at output CV. When the signal state at the CU input changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value at the CV output is incremented by one. When the instruction executes for the first time, the current counter value at the CV output is set to zero. The counter value is increased each time a positive signal edge is detected, until it reaches the high limit of the data type (INT). When the high limit is reached, the signal state at the CU input no longer has an effect on the instruction.

You can scan the counter status at the Q output. The signal state at the Q output is determined by the parameter PV. If the current counter value is greater than or equal to the value of the PV parameter, the Q output is set to signal state "1". In all other cases, the Q output has signal state "0".

The value at the CV output is reset to zero when the signal state at the R input changes to "1". As long as the R input has signal state "1", the signal state at the CU input has no effect on the instruction.

Each call of the "Count up" instruction must be assigned to an IEC Counter in which the instruction data is stored. You can declare an IEC Counter as follows:

- Declaration of a data block of the type CTU (for example, "CTU_DB")
- Declaration as a local tag of the type CTU in the "Static" section of a block (for example, #MyCTU_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC Counter is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Count up" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized with the value "1" for the R parameter of the instruction in a startup OB. If instances of the "Count up" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Note

Avoid using a counter at more than one point in the program (risk of counting errors).

The execution of the "Count up" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

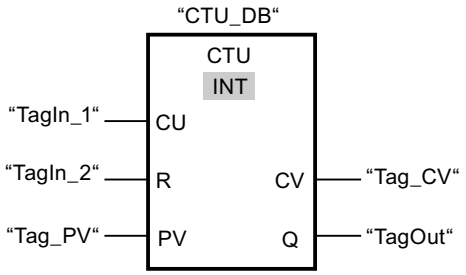
The following table shows the parameters of the "Count up" instruction:

Parameter	Declaration	Data type	Memory area	Description
CU	Input	BOOL	I, Q, M, D, L	Count input
R	Input	BOOL	I, Q, M, D, L, T, C, P	Reset input
PV	Input	INT	I, Q, M, D, L, P or constant	Value at which the output Q is set.
Q	Output	BOOL	I, Q, M, D, L	Counter status
CV	Output	INT	I, Q, M, D, L, P	Current counter value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



When the signal state of the "TagIn_1" operand changes from "0" to "1", the "Count up" instruction executes and the current counter value of the "Tag_CV" operand is incremented by one. With each additional positive signal edge, the counter value is incremented until it reaches the high limit value of the data type (INT = 32767).

The value of the PV parameter is adopted as the limit for determining the "TagOut" output. The "TagOut" output has signal state "1" as long as the current counter value is greater than or equal to the value of the "Tag_PV" operand. In all other cases, the "TagOut" output has signal state "0".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Call function blocks as single or multi-instances (Page 2182)

CTD: Count down

Description

You can use the "Count down" instruction to decrement the value at output CV. When the signal state at the CD input changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value at the CV output is decremented by one. When the instruction executes the first time, the counter value of the CV parameter will be set to the value of the PV parameter. Each time a positive signal edge is detected, the counter is decremented until it reaches the low limit value of the specified data type (INT). When the low limit is reached, the signal state at the CD input no longer has an effect on the instruction.

You can scan the counter status at the Q output. If the current counter value is less than or equal to zero, the Q output is set to signal state "1". In all other cases, the Q output has signal state "0".

The value at the CV output is set to the value of the PV parameter when the signal state at the LD input changes to "1". As long as the LD input has signal state "1", the signal state at the CD input has no effect on the instruction.

Each call of the "Count down" instruction must be assigned to an IEC Counter in which the instruction data is stored. You can declare an IEC Counter as follows:

- Declaration of a data block of the type CTD (for example, "CTD_DB")
- Declaration as a local tag of the type CTD in the "Static" section of a block (for example, #MyCTD_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC Counter is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Count down" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized with the value "1" for the LD parameter of the instruction in a startup OB. In this case the desired initial value for the CV parameter is specified in the PV parameter.

Note

Avoid using a counter at more than one point in the program (risk of counting errors).

The execution of the "Count down" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

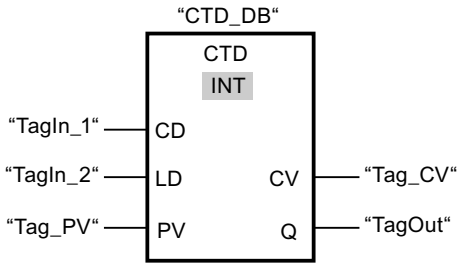
The following table shows the parameters of the "Count down" instruction:

Parameter	Declaration	Data type	Memory area	Description
CD	Input	BOOL	I, Q, M, D, L	Count input
LD	Input	BOOL	I, Q, M, D, L, T, C, P	Load input
PV	Input	INT	I, Q, M, D, L, P or constant	Value at which the output Q is set.
Q	Output	BOOL	I, Q, M, D, L	Counter status
CV	Output	INT	I, Q, M, D, L, P	Current counter value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



When the signal state of the "TagIn_1" operand changes from "0" to "1", the "Count down" instruction executes and the value at the "Tag_CV" output is decremented by one. With each additional positive signal edge, the counter value is decremented until it reaches the low limit value of the data type (INT = -32768).

The "TagOut" output has signal state "1" as long as the current counter value is less than or equal to zero. In all other cases, the "TagOut" output has signal state "0".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Call function blocks as single or multi-instances (Page 2182)

CTUD: Count up and down

Description

You can use the "Count up and down" instruction to increment and decrement the counter value at the CV output. If the signal state at the CU input changes from "0" to "1" (positive signal edge), the current counter value is incremented by one and stored at the CV output. If the signal state at the CD input changes from "0" to "1" (positive signal edge), the counter value at the CV output is decremented by one. If there is a positive signal edge at the CU and CD inputs in one program cycle, the current counter value at the CV output remains unchanged.

The counter value can be incremented until it reaches the high limit value of the specified data type (INT). When the high limit value is reached, the counter value is no longer incremented on a positive signal edge. When the low limit value of the specified data type (INT) is reached, the counter value is not decremented any further.

When the signal state at the LD input changes to "1", the counter value at the CV output is set to the value of the PV parameter. As long as the LD input has signal state "1", the signal state at the CU and CD inputs has no effect on the instruction.

The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has signal state "1", a change in the signal state of the CU, CD and LD inputs has no effect on the "Count up and down" instruction.

You can scan the current status of the up counter at the QU output. If the current counter value is greater than or equal to the value of the PV parameter, the QU output is set to signal state "1". In all other cases, the QU output has signal state "0". You can also specify a constant for the PV parameter.

You can scan the current status of the down counter at the QD output. If the current counter value is less than or equal to zero, the QD output is set to signal state "1". In all other cases, the QD output has signal state "0".

Each call of the "Count up and down" instruction must be assigned an IEC Counter in which the instruction data is stored. You can declare an IEC Counter as follows:

- Declaration of a data block of the type CTUD (for example, "CTUD_DB")
- Declaration as a local tag of the type CTUD in the "Static" section of a block (for example, #MyCTUD_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC Counter is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find this in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Count up and down" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, you must call the instances to be initialized with the following parameter values in a restart OB:

- When used as counter up, the value at the R parameter must be set to "1".
- When used as counter down, the value at the LD parameter must be set to "1". In this case you specify the desired initial value for the CV parameter in the PV parameter.

If instances of the "Count up and down" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Note

Avoid using a counter at more than one point in the program (risk of counting errors).

The execution of the "Count up and down" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Parameters

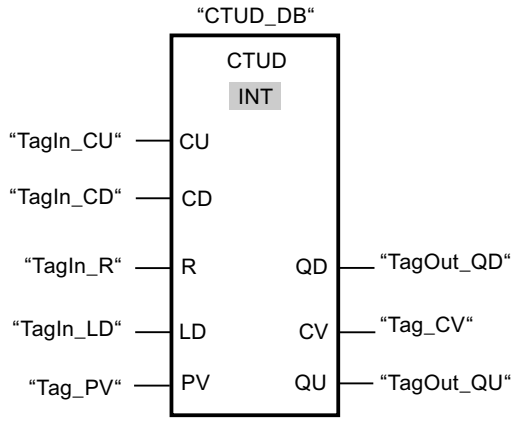
The following table shows the parameters of the "Count up and down" instruction:

Parameter	Declaration	Data type	Memory area	Description
CU	Input	BOOL	I, Q, M, D, L	Count up input
CD	Input	BOOL	I, Q, M, D, L	Count down input
R	Input	BOOL	I, Q, M, D, L, T, C, P	Reset input
LD	Input	BOOL	I, Q, M, D, L, T, C, P	Load input
PV	Input	INT	I, Q, M, D, L, P or constant	Value at which the output QU is set.
QU	Output	BOOL	I, Q, M, D, L	Status of the counter up
QD	Output	BOOL	I, Q, M, D, L	Status of the counter down
CV	Output	INT	I, Q, M, D, L, P	Current counter value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the signal state at the "TagIn_CU" or "TagIn_CD" input changes from "0" to "1" (positive signal edge), the "Count up and down" instruction is executed. When there is a positive signal edge at the "TagIn_CU" input, the current counter value is incremented by one and stored at the "Tag_CV" output. When there is a positive signal edge at the "TagIn_CD" input, the counter value is decremented by one and stored at the "Tag_CV" output. When there is a positive signal edge at the CU input, the counter value is incremented until it reaches the high limit of INT = 32767. If input CD has a positive signal edge, the counter is decremented until it reaches the low limit value of INT = -32768.

The "TagOut_QU" output has signal state "1" as long as the current counter value is greater than or equal to the value at the "Tag_PV" input. In all other cases, the "TagOut_QU" output has signal state "0".

The "TagOut_QD" output has signal state "1" as long as the current counter value is less than or equal to zero. In all other cases, the "TagOut_QD" output has signal state "0".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Call function blocks as single or multi-instances (Page 2182)

SIMATIC Counters

S_CUD: Assign parameters and count up / down

Description

You can use the "Assign parameters and count up / down" instruction to increment or decrement the value of a counter. If the signal state at the CU input changes from "0" to "1" (positive signal edge), the current counter value is incremented by one. If the signal state at the CD input changes from "0" to "1" (positive signal edge), the counter value is decremented by one. The current counter value is output as a hexadecimal value at output CV and BCD-coded at output CV_BCD. If there is a positive signal edge at the CU and CD inputs in one program cycle, the counter value remains unchanged.

The counter value can be incremented until the high limit of "999" is reached. When the high limit value is reached, the counter value is no longer incremented on a positive signal edge. When the low limit of "0" is reached, the counter value is no longer decremented.

When the signal state at input S changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if RLO is "1" at the inputs CU and CD, the counter counts accordingly in the next scan cycle, even if no change in the signal edge is detected.

The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has the signal state "1", processing of the signal state of the CU, CD and S inputs has no effect on the counter value.

The signal state at output Q is "1" if the counter value is greater than zero. If the counter value is equal to zero, output Q has the signal state "0".

Note

Avoid using a counter at more than one point in the program (risk of counting errors).

The "Assign parameters and count up / down" instruction requires a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

Parameters

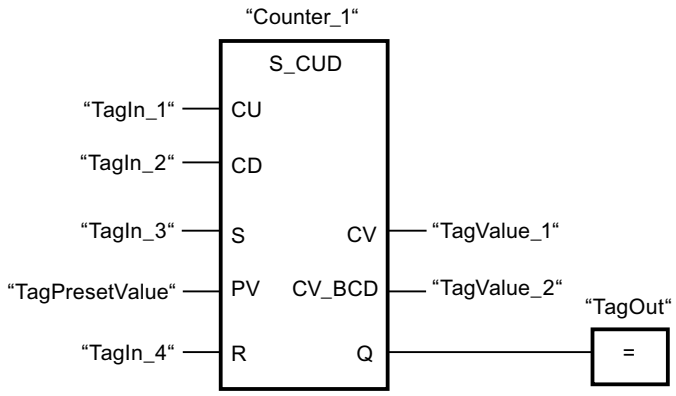
The following table shows the parameters of the "Assign parameters and count up / down" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Counter>	InOut	COUNTER	C	Counter of the instruction The number of counters depends on the CPU.
CU	Input	BOOL	I, Q, M, D, L, T, C	Count up input
CD	Input	BOOL	I, Q, M, D, L, T, C	Count down input
S	Input	BOOL	I, Q, M, D, L, T, C	Input for presetting counter
PV	Input	WORD	I, Q, M, D, L, P	Preset counter value (C#0 to C#999)
R	Input	BOOL	I, Q, M, D, L, T, C	Reset input
CV	Output	WORD, S5TIME, DATE	I, Q, M, D, L, P	Current counter value (hexadecimal)
BCD	Output	WORD, S5TIME, DATE	I, Q, M, D, L, P	Current counter value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the counter

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the signal state at the "TagIn_1" or "TagIn_2" input changes from "0" to "1" (positive signal edge), the "Assign parameters and count up / down" instruction is executed. When there is a positive signal edge at the "TagIn_1" input and the current counter value is less than "999", the counter value is incremented by one. When there is a positive signal edge at the "TagIn_2" input and the current counter value is greater than "0", the counter value is decremented by one.

When the signal state at the "TagIn_3" input changes from "0" to "1", the counter value is set to the value of the "TagPresetValue" operand. The counter value is reset to "0" when the "TagIn_4" operand has signal state "1".

The current counter value is hexadecimal in the "TagValue_1" operand and BCD-coded in the "TagValue_2" operand.

The "TagOut" output has the signal state "1" as long as the current counter value is not equal to "0".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

S_CU: Assign parameters and count up

Description

You can use the "Assign parameters and count up" instruction to increment the value of a counter. If the signal state at the CU input changes from "0" to "1" (positive signal edge), the current counter value is incremented by one. The current counter value is output as a hexadecimal value at output CV and BCD-coded at output CV_BCD. The counter value can be incremented until the limit of "999" is reached. When the limit is reached, the counter value is no longer incremented on a positive signal edge.

When the signal state at input S changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if RLO at input CU is "1", the counter will count accordingly in the next scan cycle, even when no change has been detected in the signal edge.

The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has the signal state "1", processing of the signal state of the CU and S inputs has no effect on the counter value.

The signal state at output Q is "1" if the counter value is greater than zero. If the counter value is equal to zero, output Q has the signal state "0".

Note

Avoid using a counter at more than one point in the program (risk of counting errors).

The "Assign parameters and count up" instruction requires a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

Parameters

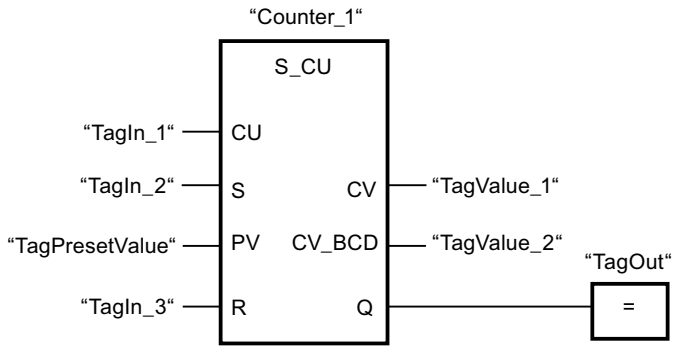
The following table shows the parameters of the "Assign parameters and count up" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Counter>	InOut	COUNTER	C	Counter of the instruction The number of counters depends on the CPU.
CU	Input	BOOL	I, Q, M, D, L, T, C	Count up input
S	Input	BOOL	I, Q, M, D, L, T, C	Input for presetting counter
PV	Input	WORD	I, Q, M, D, L, P	Preset counter value (C#0 to C#999)
R	Input	BOOL	I, Q, M, D, L, T, C	Reset input
CV	Output	WORD, S5TIME, DATE	I, Q, M, D, L, P	Current counter value (hexadecimal)
BCD	Output	WORD, S5TIME, DATE	I, Q, M, D, L, P	Current counter value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the counter

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



When the signal state at the "TagIn_1" input changes from "0" to "1" (positive signal edge) and the current counter value is less than "999", the counter value is incremented by one. When the signal state at the "TagIn_2" input changes from "0" to "1", the counter value is set to the value of the "TagPresetValue" operand. The counter value is reset to "0" when the "TagIn_3" operand has signal state "1".

The current counter value is hexadecimal in the "TagValue_1" operand and BCD-coded in the "TagValue_2" operand.

The "TagOut" output has the signal state "1" as long as the current counter value is not equal to "0".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

S_CD: Assign parameters and count down

Description

You can use the "Assign parameters and count down" instruction to decrement the value of a counter. If the signal state at the CD input changes from "0" to "1" (positive signal edge), the counter value is decremented by one. The current counter value is output as a hexadecimal value at output CV and BCD-coded at output CV_BCD. The counter value can be decremented until the low limit of "0" is reached. When the low limit is reached, the counter value is no longer decremented on a positive signal edge.

When the signal state at input S changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if RLO at input CD is "1", the counter will count accordingly in the next scan cycle, even when no change has been detected in the signal edge.

The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has the signal state "1", processing of the signal state of the CD and S inputs has no effect on the counter value.

The signal state at output Q is "1" if the counter value is greater than zero. If the counter value is equal to zero, output Q has the signal state "0".

Note

Avoid using a counter at more than one point in the program (risk of counting errors).

The "Assign parameters and count down" instruction requires a preceding logic operation for the edge evaluation and can be placed within or at the end of the network.

Parameters

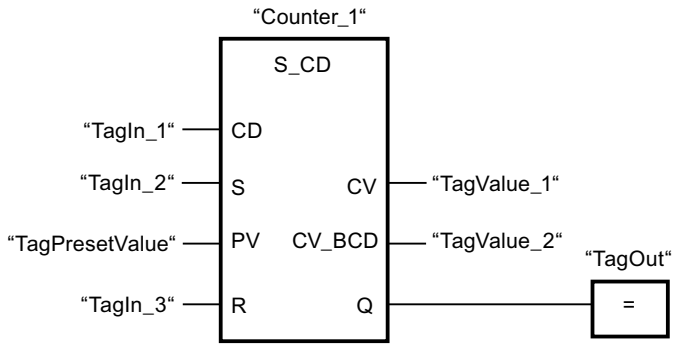
The following table shows the parameters of the "Assign parameters and count down" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Counter>	InOut	COUNTER	C	Counter of the instruction The number of counters depends on the CPU.
CD	Input	BOOL	I, Q, M, D, L	Count down input
S	Input	BOOL	I, Q, M, D, L, T, C	Input for presetting counter
PV	Input	WORD	I, Q, M, D, L, P	Preset counter value (C#0 to C#999)
R	Input	BOOL	I, Q, M, D, L, T, C	Reset input
CV	Output	WORD, S5TIME, DATE	I, Q, M, D, L, P	Current counter value (hexadecimal)
BCD	Output	WORD, S5TIME, DATE	I, Q, M, D, L, P	Current counter value (BCD format)
Q	Output	BOOL	I, Q, M, D, L	Status of the counter

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



When the signal state at the "TagIn_1" input changes from "0" to "1" (positive signal edge) and the current counter value is greater than "0", the counter value is decremented by one. When the signal state at the "TagIn_2" input changes from "0" to "1", the counter value is set to the value of the "TagPresetValue" operand. The counter value is reset to "0" when the "TagIn_3" operand has signal state "1".

The current counter value is hexadecimal in the "TagValue_1" operand and BCD-coded in the "TagValue_2" operand.

The "TagOut" output has the signal state "1" as long as the current counter value is not equal to "0".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

SC: Set counter value

Description

You can use the "Set counter value" instruction to increment the value of a counter. The instruction is executed when the result of logic operation (RLO) at the start input of the instruction changes from "0" to "1". If the instruction is executed, the counter is set to the specified counter value.

The "Set counter value" instruction needs a preceding logic operation for the edge evaluation and can only be placed on the right edge of the network.

Parameters

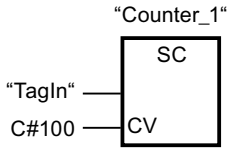
The following table shows the parameters of the "Set counter value" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L	Start input
PV	Input	WORD	I, Q, M, D, L or constant	Value with which the counter is preset in the BCD format. (C#0 to C#999)
<Counter>	InOut/Input	COUNTER	C	Counter which is preset.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



"Counter_1" starts with the value "100" when the signal state of the "TagIn" operand changes from "0" to "1".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

CU: Count up

Description

You use the "Count up" instruction to increment the value of the specified counter by the count of one on a rising edge at the start input. The count is incremented until the limit of "999" is reached. When the limit is reached, the counter value is no longer incremented on a positive signal edge.

The "Count up" instruction needs a preceding logic operation for the edge evaluation and can only be placed on the right edge of the network.

Parameters

The following table shows the parameters of the "Count up" instruction:

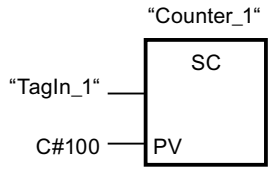
Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L	Start input
<Counter>	InOut/Input	COUNTER	C	Counter whose value is incremented.

For additional information on valid data types, refer to "See also".

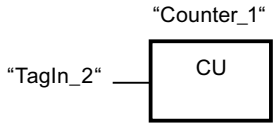
Example

The following example shows how the instruction works:

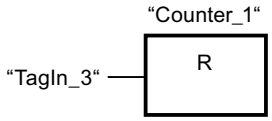
Network 1



Network 2



Network 3



When the signal state of the operand "TagIn_1" changes from "0" to "1" (positive signal edge), the counter "Counter_1" is preset with the value "100".

The value of "Counter_1" is incremented by the count of one when the signal state of the "TagIn_2" operand changes from "0" to "1".

If the "TagIn_3" operand returns signal state "1", the value of "Counter_1" is reset to "0".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

CD: Count down

Description

You use the "Count down" instruction to decrement the value of the specified counter by the count of one on a rising edge at the start input. The count is decremented until the limit of "0" is reached. When the limit is reached, the counter value is no longer changed on a positive signal edge.

The "Count down" instruction needs a preceding logic operation for the edge evaluation and can only be placed on the right edge of the network.

Parameters

The following table shows the parameters of the "Count down" instruction:

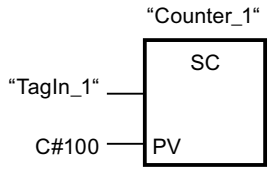
Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, T, C, D, L	Start input
<Counter>	InOut/Input	COUNTER	C	Counter whose value is decremented.

For additional information on valid data types, refer to "See also".

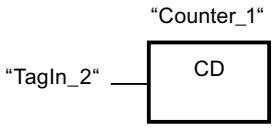
Example

The following example shows how the instruction works:

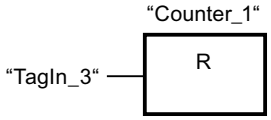
Network 1



Network 2



Network 3



When the signal state of the operand "TagIn_1" changes from "0" to "1" (positive signal edge), the counter "Counter_1" is preset with the value "100".

The value of "Counter_1" is incremented by the count of one when the signal state of the "TagIn_2" operand changes from "0" to "1".

If the "TagIn_3" operand returns signal state "1", the value of "Counter_1" is reset to "0".

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Comparator operations

CMP ==: Equal

Description

You can use the "Equal" instruction to query whether the value at input IN1 is equal to the value at input IN2.

If the condition of the comparison is satisfied, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Note

Comparison of floating-point numbers

If you want to compare the data types REAL or LREAL, instead of the instruction "CMP ==: Equal", use the instruction "IN_RANGE: Value within range".

Parameters

The following table shows the parameters of the instruction "Equal":

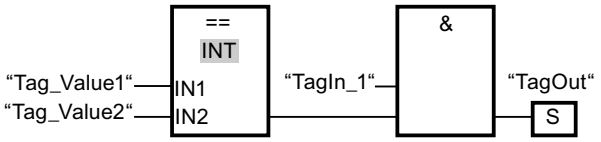
Parameter	Declaration	Data type	Memory area	Description
IN1	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P or constant	First comparison value
IN2	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also":

Example

The following example shows how the instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The condition of the comparison instruction is fulfilled ("Tag_Value1" = "Tag_Value2").

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Setting the status bit for comparison instructions (Page 2460)

Basic information on the status word (Page 2453)

CMP <>: Not equal

Description

You can use the "Not equal" instruction to query whether the value at input IN1 is not equal to the value at input IN2.

If the condition of the comparison is satisfied, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Parameters

The following table shows the parameters of the instruction "Not equal":

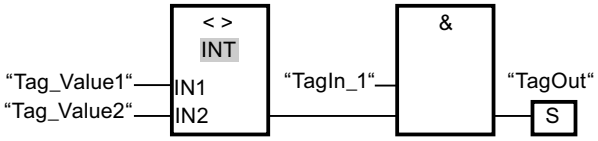
Parameters	Declaration	Data type	Memory area	Description
IN1	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P or constant	First value to compare
IN2	Input	Bit strings, integers, floating-point numbers	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The condition of the comparison instruction is fulfilled ("Tag_Value1" <> "Tag_Value2").

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Setting the status bit for comparison instructions (Page 2460)

Basic information on the status word (Page 2453)

CMP >=: Greater or equal

Description

You can use the "Greater or equal" instruction to query whether the value at input IN1 is greater or equal to the value at input IN2. Both values to be compared must be of the same data type.

If the condition of the comparison is satisfied, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Parameters

The following table shows the parameters of the instruction "Greater or equal":

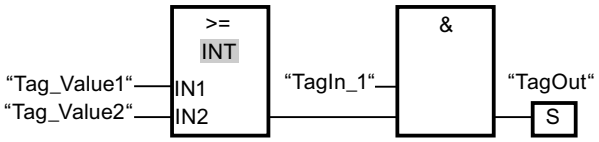
Parameters	Declaration	Data type	Memory area	Description
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	First value to compare
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The condition of the comparison instruction is fulfilled ("Tag_Value1" >= "Tag_Value2").

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Example of detecting the fill level of a storage area (Page 3127)

Setting the status bit for comparison instructions (Page 2460)

Basic information on the status word (Page 2453)

CMP <=: Less or equal

Description

You can use the "Less or equal" instruction to query whether the value at input IN1 is less or equal to the value at input IN2. Both values to be compared must be of the same data type.

If the condition of the comparison is satisfied, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Parameters

The following table shows the parameters of the instruction "Less or equal":

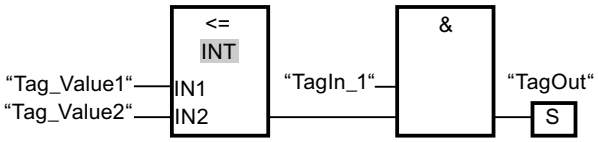
Parameters	Declaration	Data type	Memory area	Description
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	First value to compare
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The condition of the comparison instruction is fulfilled ("Tag_Value1" <= "Tag_Value2").

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Setting the status bit for comparison instructions (Page 2460)

Basic information on the status word (Page 2453)

CMP >: Greater than

Description

You can use the "Greater than" instruction to query whether the value at input IN1 is greater than the value at input IN2. Both values to be compared must be of the same data type.

If the condition of the comparison is satisfied, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Parameters

The following table shows the parameters of the instruction "Greater than":

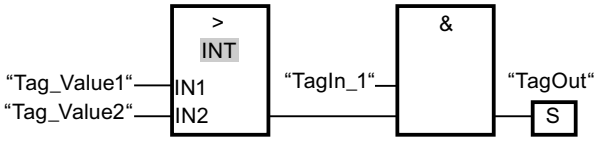
Parameters	Declaration	Data type	Memory area	Description
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	First value to compare
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The condition of the comparison instruction is fulfilled ("Tag_Value1" > "Tag_Value2").

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Setting the status bit for comparison instructions (Page 2460)

Basic information on the status word (Page 2453)

CMP <: Less than

Description

You can use the "Less than" instruction to query whether the value at input IN1 is less than the value at input IN2. Both values to be compared must be of the same data type.

If the condition of the comparison is satisfied, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

Parameters

The following table shows the parameters of the instruction "Less than":

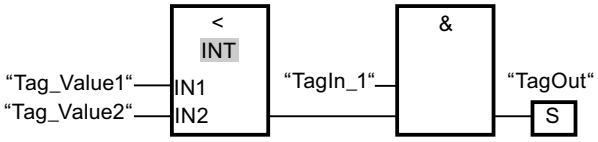
Parameters	Declaration	Data type	Memory area	Description
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	First value to compare
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Second value to compare

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- The operand "TagIn_1" has the signal state "1".
- The condition of the comparison instruction is fulfilled ("Tag_Value1" < "Tag_Value2").

See also

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Example of detecting the fill level of a storage area (Page 3127)

Setting the status bit for comparison instructions (Page 2460)

Basic information on the status word (Page 2453)

Math functions

ADD: Add

Description

You can use the "Add" instruction to add the value at input IN1 to the value at input IN2 and query the sum at output OUT (OUT := IN1+IN2).

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at output OUT.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the instruction "Add":

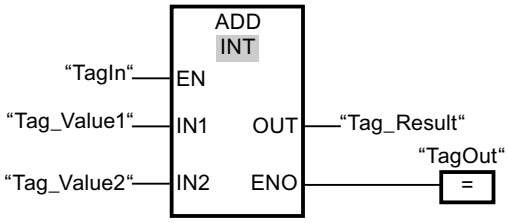
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	First number to be added
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Second number to be added
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Sum

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also":

Example

The following example shows how the instruction works:



If the operand "TagIn" has the signal state "1", the "Add" instruction is executed. The value of operand "Tag_Value1" is added to the value of operand "Tag_Value2". The result of the addition is stored in the operand "Tag_Result". If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Example of calculating an equation (Page 3131)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

SUB: Subtract

Description

You can use the "Subtract" instruction to subtract the value at input IN2 from the value at input IN1and query the difference at output OUT (OUT := IN1-IN2).

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at output OUT.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the instruction "Subtract":

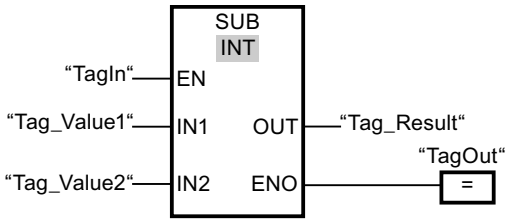
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Minuend
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Subtrahend
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Difference

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the operand "TagIn" has the signal state "1", the "Subtract" instruction is executed. The value of operand "Tag_Value2" is subtracted from the value of operand "Tag_Value1". The result of the subtraction is stored in the operand "Tag_Result". If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

MUL: Multiply

Description

You can use the "Multiply" instruction to multiply the value at input IN1 with the value at input IN2and query the total at output OUT (OUT := IN1*IN2).

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The result is outside the range permitted for the data type specified at output OUT.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the instruction "Multiply":

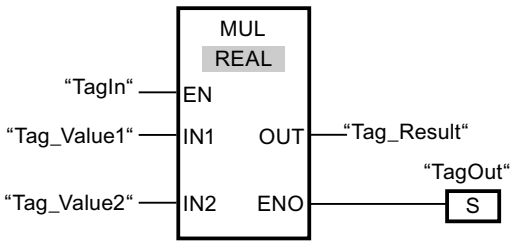
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Multiplier
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Multiplicand
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Product

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the operand "TagIn" has the signal state "1", the "Multiply" instruction is executed. The value of operand "Tag_Value1" is multiplied with the value of operand "Tag_Value2". The result of the multiplication is stored in the operand "Tag_Result". If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

Overview of the valid data types (Page 2236)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Selecting the data type of an FBD element (Page 2624)

Adding additional inputs and outputs to FBD elements (Page 2639)

Inserting operands in FBD instructions (Page 2643)

DIV: Divide

Description

You can use the "Divide" instruction to divide the value at input IN1 by the value at input IN2 and query the quotient at output OUT (OUT := IN1/IN2).

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The result of the instruction is outside the range permitted for the data type specified at output OUT.
- A floating-point number has an invalid value.

Parameters

The following table shows the parameters of the instruction "Divide":

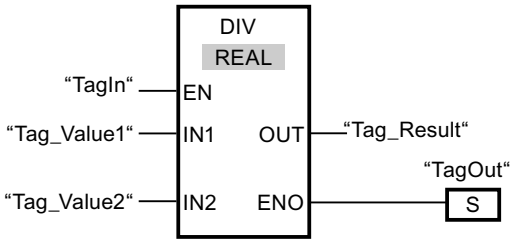
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Dividend
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Divisor
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Quotient value

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the operand "TagIn" has the signal state "1", the "Divide" instruction is executed. The value of operand "Tag_Value1" is divided by the value of operand "Tag_Value2". The result of the division is stored in the operand "Tag_Result". If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Example of calculating an equation (Page 3131)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

MOD: Return remainder of division

Description

You can use the "Return remainder of division" instruction to divide the value at input IN1 by the value at input IN2 and query the quotient at output OUT.

Parameters

The following table shows the parameters of the instruction "Return remainder of division":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

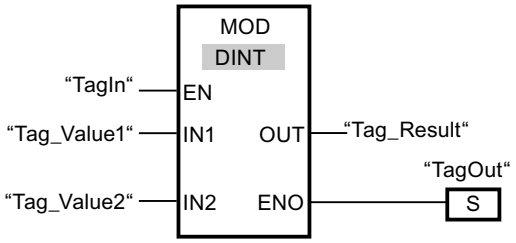
Parameters	Declaration	Data type	Memory area	Description
IN1	Input	Integers	I, Q, M, D, L, P or constant	Dividend
IN2	Input	Integers	I, Q, M, D, L, P or constant	Divisor
OUT	Output	Integers	I, Q, M, D, L, P	Remainder of division

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the operand "TagIn" has the signal state "1", the "Return remainder of division" instruction is executed. The value of operand "Tag_Value1" is divided by the value of operand "Tag_Value2". The remainder of division is stored in operand "Tag_Result". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

NEG: Create twos complement

Description

You can use the "Create twos complement" instruction to change the sign of the value at input IN and query the result at output OUT. If there is a positive value at input IN, for example, the negative equivalent of this value is sent to output OUT.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The result of the instruction is outside the valid range for the data type specified at the OUT output (only applies to integers).

Parameters

The following table shows the parameters of the "Create twos complement" instruction:

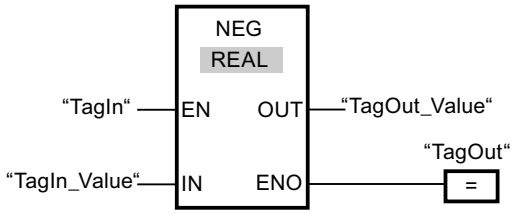
Parameters	Declaration	Data type		Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Input value
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Twos complement of the input value

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the operand "TagIn" has the signal state "1", the "Create twos complement" instruction is executed. The sign of the value at input "TagIn_Value" is changed and the result is provided at output "TagOut_Value". If the instruction is executed without errors, the enable output ENO has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)
Basics of the EN/ENO mechanism (Page 2447)
Inserting FBD elements (Page 2621)
Selecting the data type of an FBD element (Page 2624)
Editing FBD elements (Page 2635)
Inserting operands in FBD instructions (Page 2643)
Basic information on the status word (Page 2453)
Setting the status bits in instructions with integers (Page 2458)
Setting the status bits in instructions with floating-point numbers (Page 2459)

ABS: Form absolute value

Description

You can use the "Form absolute value" instruction to calculate the absolute value of the value specified at input IN. The result of the instruction is provided at output OUT and can be queried there.

Parameters

The following table shows the parameters of the instruction "Form absolute value":

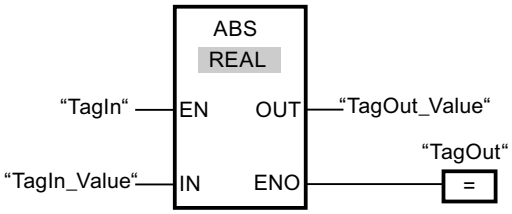
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	REAL	I, Q, M, D, L, P or constant	Input value
OUT	Output	REAL	I, Q, M, D, L, P	Absolute value of the input value

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	TagIn_Value	-6.234
OUT	TagOut_Value	6.234

If the operand "TagIn" has the signal state "1", the "Form absolute value" instruction is executed. The instruction calculates the absolute value of the value at input "TagIn_Value" and sends the result to output "TagOut_Value". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

MIN: Get minimum

Description

The "Get minimum" instruction compares the values at the inputs IN1, IN2 and IN3 and writes the lowest value to the OUT output. The instruction can only be executed if the tags of all inputs are of the same data type.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- Enable input EN has the signal state "0".
- The specified tags are not of the same data type.
- A floating-point number has an invalid value.

Note

Only the data types INT, DINT and REAL are permitted at the input parameters, even when the IEC check is disabled. If you use the data types WORD or DWORD at the input parameters, the instruction will return an undefined result.

Parameters

The following table shows the parameters of the "Get minimum" instruction:

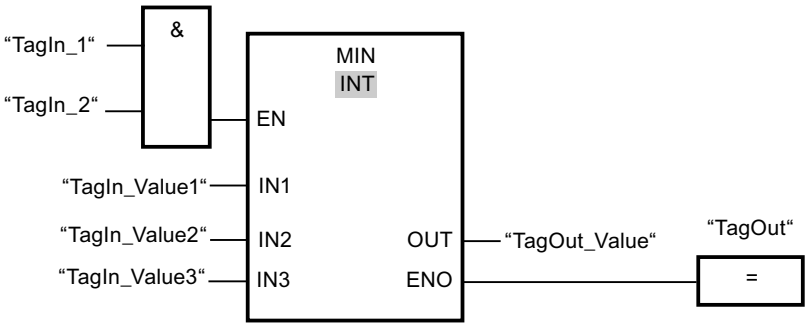
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	INT, DINT, REAL	I, Q, M, D, L, P	First input value
IN2	Input	INT, DINT, REAL	I, Q, M, D, L, P	Second input value
IN3	Input	INT, DINT, REAL	I, Q, M, D, L, P	Third input value
OUT	Output	INT, DINT, REAL	I, Q, M, D, L, P	Result

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
IN3	TagIn_Value3	13333
OUT	TagOut_Value	12222

If the "TagIn_1" and "TagIn_2" operands have signal state "1", the "Get minimum" instruction is executed. The instruction compares the values of the specified operands and copies the lowest value ("TagIn_Value1") to output "TagOut_Value". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Removing inputs and outputs (Page 2599)

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

MAX: Get maximum

Description

The "Get maximum" instruction compares the values at the inputs IN1, IN2 and IN3 and writes the highest value to the OUT output. The instruction can only be executed if the tags of all inputs are of the same data type.

The ENO enable output has the signal state "0" if one of the following conditions is fulfilled:

- Enable input EN has the signal state "0".
- The specified tags are not of the same data type.
- A floating-point number has an invalid value.

Note

Only the data types INT, DINT and REAL are permitted at the input parameters, even when the IEC check is disabled. If you use the data types WORD or DWORD at the input parameters, the instruction will return an undefined result.

Parameters

The following table shows the parameters of the "Get maximum" instruction:

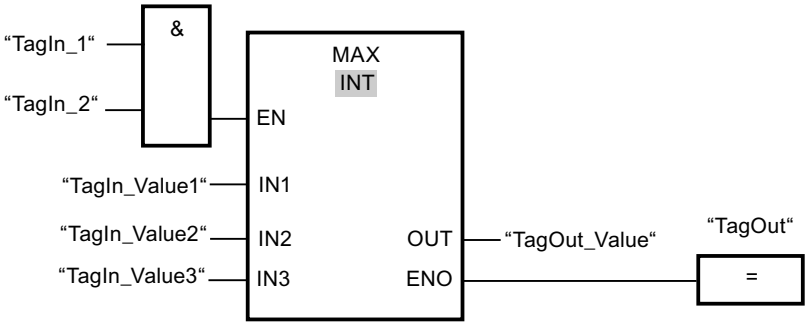
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	INT, DINT, REAL	I, Q, M, D, L, P	First input value
IN2	Input	INT, DINT, REAL	I, Q, M, D, L, P	Second input value
IN3	Input	INT, DINT, REAL	I, Q, M, D, L, P	Third input value
OUT	Output	INT, DINT, REAL	I, Q, M, D, L, P	Result

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
IN3	TagIn_Value3	13333
OUT	TagOut_Value	14444

If the "TagIn_1" and "TagIn_2" operands have signal state "1", the "Get maximum" instruction is executed. The instruction compares the values of the specified operands and copies the highest value ("TagIn_Value2") to output "TagOut_Value". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

LIMIT: Set limit value

Description

You use the "Set limit value" instruction to limit the value at input IN to the values at the inputs MN and MX. If the value at the IN input meets the MN condition <= IN <= MX, it is copied to the OUT output. If the condition is not fulfilled and the input value IN is below the low limit MN, output OUT is set to the value of the input MN. If the high limit MX is exceeded, output OUT is set to the value of the input MX.

If the value at input MN is greater than that at input MX, the result is undefined and the enable output ENO is "0".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The specified tags are not of the same data type.
- An operand has an invalid value.
- The value at input MN is greater than the value at input MX.

Note

Only the data types INT, DINT and REAL are permitted at the input parameters, even when the IEC check is disabled. If you use the data types WORD or DWORD at the input parameters, the instruction will return an undefined result.

Parameters

The following table shows the parameters of the "Set limit value" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
MN	Input	INT, DINT, REAL	I, Q, M, D, L, P	Low limit
IN	Input	INT, DINT, REAL	I, Q, M, D, L, P	Input value

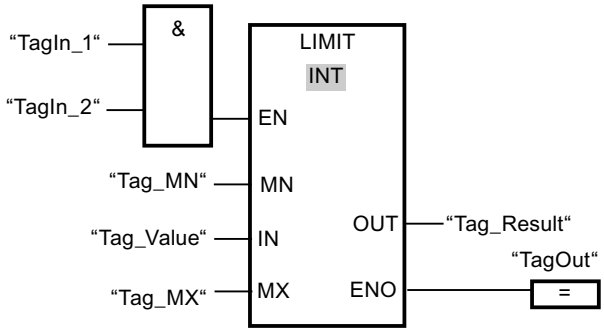
Parameter	Declaration	Data type	Memory area	Description
MX	Input	INT, DINT, REAL	I, Q, M, D, L, P	High limit
OUT	Output	INT, DINT, REAL	I, Q, M, D, L, P	Result

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
MN	Tag_MN	12000
IN	Tag_Value	8000
MX	Tag_MX	16000
OUT	Tag_Result	12000

If the "TagIn_1" and "TagIn_2" operands have the signal state "1", the "Set limit value" instruction is executed. The value of operand "Tag_Value" is compared with the values of operands "Tag_MN" and "Tag_MX". Since the value at the operand "Tag_Value" is less than the low limit, the value of the operand "Tag_MN" is copied to output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

SQR: Form square

Description

You can use the "Form square" instruction to square the value at input IN and query the result at output OUT.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

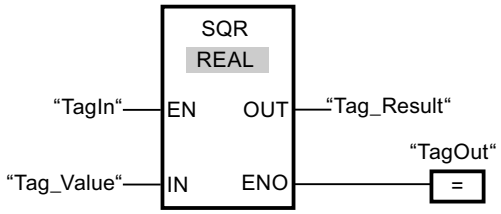
The following table shows the parameters of the instruction "Form square":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Square of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	5.0
OUT	Tag_Result	25.0

If the operand "TagIn" has the signal state "1", the "Form square" instruction is executed. The instruction squares the value of the operand "Tag_Value" and sends the result to output

"Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

SQRT: Form square root

Description

You can use the "Form square root" instruction to form the square root of the value at input IN and query the result at output OUT. The instruction has a positive result if the input value is greater than zero. If input values are less than zero, output OUT returns an invalid floating-point number. If the value at input IN is "0", the result is also "0".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.
- The value at input IN is negative.

Parameters

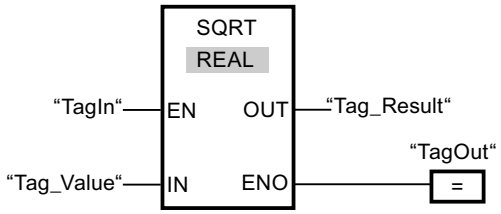
The following table shows the parameters of the instruction "Form square root":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Square root of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	25.0
OUT	Tag_Result	5.0

If the operand "TagIn" has the signal state "1", the "Form square root" instruction is executed. The instruction calculates the square root of the operand "Tag_Value" and sends the result to output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

LN: Form natural logarithm

Description

You can use the "Form natural logarithm" instruction to calculate the natural logarithm to base e (e = 2.718282) of the value at input IN. The result is sent to output OUT and can be queried there. The instruction has a positive result if the input value is greater than zero. If input values are less than zero, output OUT returns an invalid floating-point number.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.
- The value at input IN is negative.

Parameters

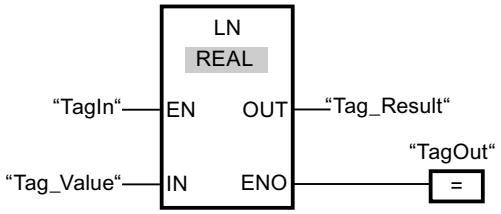
The following table shows the parameters of the instruction "Form natural logarithm":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Natural logarithm of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the operand "TagIn" has the signal state "1", the "Form natural logarithm" instruction is executed. The instruction forms the natural logarithm of the value at input "Tag_Value" and sends the result to output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

EXP: Form exponential value

Description

You can use the "Form exponential value" instruction to calculate the exponent from the base e (e = 2.718282) and the value specified at input IN. The result is provided at output OUTand can be queried there (OUT = e^{IN}).

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

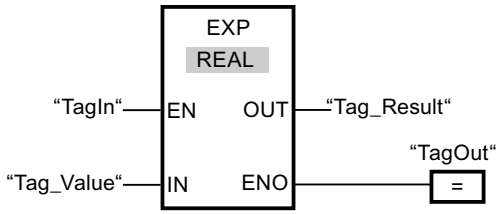
The following table shows the parameters of the instruction "Form exponential value":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Exponential value of the input value IN

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the operand "TagIn" has the signal state "1", the "Form exponential value" instruction is executed. The instruction forms the exponent from base e and the value of the operand "Tag_Value" and sends the result to output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

SIN: Form sine value

Description

You can use the "Form sine value" instruction to calculate the sine of an angle. The size of the angle is specified in the radian measure at input IN. The result of the instruction is provided at output OUT and can be queried there.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

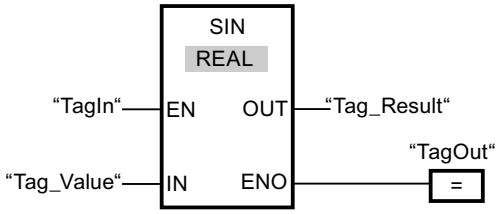
The following table shows the parameters of the instruction "Form sine value":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Size of angle in the radian measure
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Sine of the specified angle

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	+1.570796 (π/2)
OUT	Tag_Result	1.0

If the operand "TagIn" has the signal state "1", the "Form sine value" instruction is executed. The instruction calculates the sine of the angle specified at input "Tag_Value" and sends the result to output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

COS: Form cosine value

Description

You can use the "Form cosine value" instruction to calculate the cosine of an angle. The size of the angle is specified in the radian measure at input IN. The result of the instruction is provided at output OUT and can be queried there.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

The following table shows the parameters of the instruction "Form cosine value":

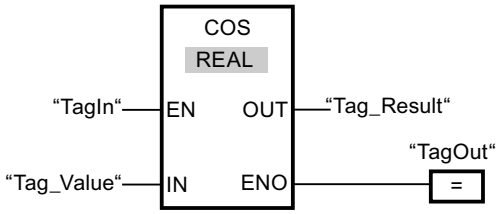
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

Parameters	Declaration	Data type	Memory area	Description
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Size of angle in the radian measure
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Cosine of the specified angle

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	+1.570796 (π/2)
OUT	Tag_Result	0

If the operand "TagIn" has the signal state "1", the "Form cosine value" instruction is executed. The instruction calculates the cosine of the angle specified at input "Tag_Value" and sends the result to output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

TAN: Form tangent value

Description

You can use the "Form tangent value" instruction to calculate the tangent of an angle. The size of the angle is specified in the radian measure at input IN. The result of the instruction is provided at output OUT and can be queried there.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

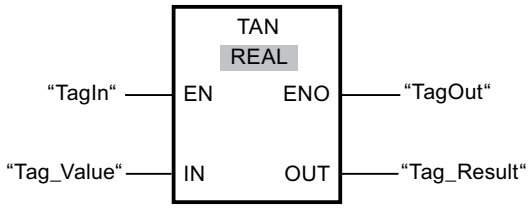
The following table shows the parameters of the instruction "Form tangent value":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Size of angle in the radian measure
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Tangent of the specified angle

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	+3.141593 (π)
OUT	Tag_Result	0

If the operand "TagIn" has the signal state "1", the "Form tangent value" instruction is executed. The instruction calculates the tangent of the angle specified at input "Tag_Value" and stores the result at output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 2236)
Basics of the EN/ENO mechanism (Page 2447)
Inserting FBD elements (Page 2621)
Selecting the data type of an FBD element (Page 2624)
Editing FBD elements (Page 2635)
Inserting operands in FBD instructions (Page 2643)
Basic information on the status word (Page 2453)
Setting the status bits in instructions with integers (Page 2458)
Setting the status bits in instructions with floating-point numbers (Page 2459)

ASIN: Form arcsine value

Description

You can use the "Form arcsine value" instruction to calculate the size of the angle from the sine value specified at input IN, which corresponds to this value. Only valid floating-point numbers within the range -1 to +1 can be specified at input IN. The calculated angle size is given in the radian measure at output OUT and can range in value from -π/2 to +π/2.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.
- The value at input IN is outside the permitted range (-1 to +1).

Parameters

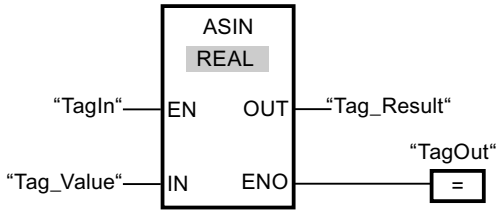
The following table shows the parameters of the instruction "Form arcsine value":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Sine value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Size of angle in the radian measure

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	1.0
OUT	Tag_Result	+1.570796 (π/2)

If the operand "TagIn" has the signal state "1", the "Form arcsine value" instruction is executed. The instruction calculates the size of the angle corresponding to the sine value at input "Tag_Value". The result of the instruction is stored at output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

ACOS: Form arccosine value

Description

You can use the "Form arccosine value" instruction to calculate the size of the angle from the cosine value specified at input IN, which corresponds to this value. Only valid floating-point numbers within the range -1 to +1 can be specified at input IN. The calculated angle size is given in the radian measure at output OUT and can range in value from 0 to +π.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.
- The value at input IN is outside the permitted range (-1 to +1).

Parameters

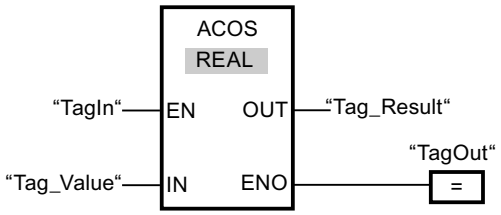
The following table shows the parameters of the instruction "Form arccosine value":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Cosine value
OUT	Output	Floating-point numbers	I, Q, M, D, L, P	Size of angle in the radian measure

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	0
OUT	Tag_Result	+1.570796 (π/2)

If the operand "TagIn" has the signal state "1", the "Form arccosine value" instruction is executed. The instruction calculates the size of the angle corresponding to the cosine value at input "Tag_Value". The result of the instruction is stored at output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 2236)
Basics of the EN/ENO mechanism (Page 2447)
Inserting FBD elements (Page 2621)
Selecting the data type of an FBD element (Page 2624)
Editing FBD elements (Page 2635)
Inserting operands in FBD instructions (Page 2643)
Basic information on the status word (Page 2453)
Setting the status bits in instructions with integers (Page 2458)
Setting the status bits in instructions with floating-point numbers (Page 2459)

ATAN: Form arctangent value

Description

You can use the "Form arctangent value" instruction to calculate the size of the angle from the tangent value specified at input IN, which corresponds to this value. Only valid floating-point numbers may be specified at input IN. The calculated angle size is output in radians at the OUT output and can range in value from -π/2 to +π/2.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The value at input IN is not a valid floating-point number.

Parameters

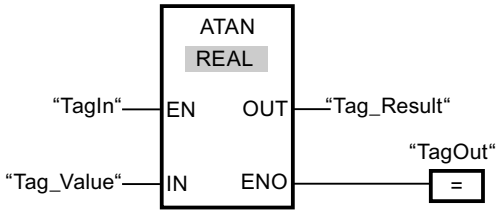
The following table shows the parameters of the instruction "Form arctangent value":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	REAL	I, Q, M, D, L, P or constant	Tangent value
OUT	Output	REAL	I, Q, M, D, L, P	Size of angle in the radian measure

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	Tag_Value	1.0
OUT	Tag_Result	+0.785398 (π/4)

If the operand "TagIn" has the signal state "1", the "Form arctangent value" instruction is executed. The instruction calculates the size of the angle corresponding to the tangent value at input "Tag_Value". The result of the instruction is stored at output "Tag_Result". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

Move operations

MOVE: Move value

Description

You use the "Move value" instruction to transfer the content of the operand at the IN input to the operand at the OUT1 output. The transfer is always made in the direction of ascending addresses.

The following table shows the possible transfers:

Source (IN)	Destination (OUT1)	
	With IEC check	Without IEC check
BYTE	BYTE, WORD, DWORD	BYTE, WORD, DWORD, INT, DINT, TIME, DATE, TOD, CHAR
WORD	WORD; DWORD	BYTE, WORD, DWORD, INT, DINT, TIME, S5TIME, DATE, TOD, CHAR
DWORD	DWORD	BYTE, WORD, DWORD, INT, DINT, REAL, TIME, DATE, TOD, CHAR
INT	INT	BYTE, WORD, DWORD, INT, DINT, TIME, DATE, TOD
DINT	DINT	BYTE, WORD, DWORD, INT, DINT, TIME, DATE, TOD
REAL	REAL	DWORD, REAL
TIME	TIME	BYTE, WORD, DWORD, INT, DINT, TIME
S5TIME	S5TIME	WORD, S5TIME
DATE	DATE	BYTE, WORD, DWORD, INT, DINT, DATE
TOD	TOD	BYTE, WORD, DWORD, INT, DINT, TOD
CHAR	CHAR	BYTE, WORD, DWORD, CHAR
COUNTER	INT, WORD, COUNTER	WORD, DWORD, INT, UINT, DINT, UDINT
TIMER	INT, WORD, TIMER	WORD, DWORD, INT, UINT, DINT, UDINT

If the bit length of the data type at input IN exceeds the bit length of the data type at output OUT1, the higher-order bits of the source value are lost. If the bit length of the data type at input IN is less than the bit length of the data type at output OUT1, the higher order bits of the destination value are overwritten with zeros.

You can also use the "Move block" (BLKMOV) and "Move block uninterruptible" (UBLKMOV) instructions to copy fields and structures.

Parameters

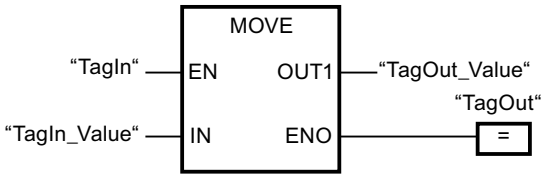
The following table shows the parameters of the "Move value" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers, floating-point numbers, timers, DATE, TOD, CHAR, TIMER, COUNTER	I, Q, M, D, L or constant	Source value
OUT1	Output	Bit strings, integers, floating-point numbers, timers, DATE, TOD, CHAR, TIMER, COUNTER	I, Q, M, D, L	Destination address

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
OUT1	TagOut_Value	0011 1111 1010 1111

If the operand "TagIn" has the signal state "1", the "Move value" instruction is executed. The instruction copies the content of the operand "TagIn_Value" to the operand "TagOut_Value" and set output "TagOut" to signal state "1".

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Master control relay (Page 2471)

MCR<: Open MCR ranges (Page 5739)

BLKMOV: Move block

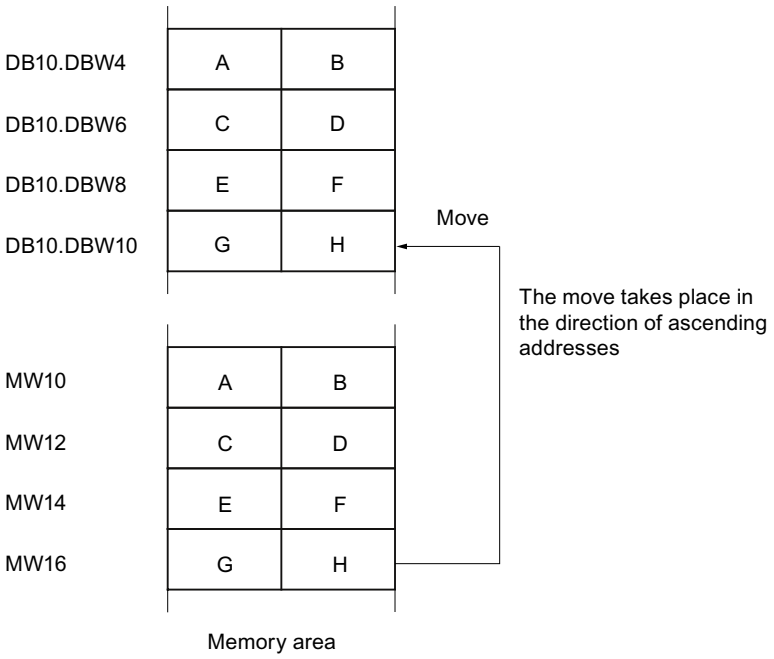
Description

You can use the "Move block" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use ANY pointer to define the source and destination area.

Note

In this instruction, you can only use data types which are valid on a CPU of the S7-300/400 series.

The following figure shows the principle of the move operation:



Consistency of the source data and the target data

Make sure that the source data remain unchanged during execution of the instruction "Move block". Otherwise, consistency of the destination data cannot be guaranteed.

Interruptibility

As long as the source area is not part of a data block that only exists in the load memory, there is no limit to the nesting depth.

If, however, BLKMOV is interrupted while copying from a DB that is not runtime-relevant, the execution of BLKMOV can no longer be nested.

Memory areas

You can use the "Move block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output
- Data blocks not runtime-relevant

General rules for moving

The source and destination area must not overlap. If the source and destination area have different lengths, only the length of the smaller area will be moved.

If the source area is less than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is less than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a block of data type BOOL is moved, the specified length of the area must be dividable by 8, otherwise it will not be possible to execute the instruction.

Note

If an actual source or destination area is smaller than a destination or source area specified on the SRCBLK or DSTBLK parameter, the system behavior depends on which CPU type is being used:

- The following applies to S7-300 CPUs:
  No data are transferred. Error code W#16#837F is output at the RET_VAL parameter. This error code has to be evaluated in your user program.
- The following applies to the S7-400 V4-CPUs (V4, V4 H-CPU and V4.5. H-CPU):
  No data are transferred. Error code W#16#8122 or W#16#8323 is output at the RET_VAL parameter. This error code has to be evaluated in your user program.
- For all other S7-400-CPUs:
  Data are transferred. Error code W#16#8122 or W#16#8323 is output at the RET_VAL parameter. This error code has to be evaluated in your user program.

Rules for moving character strings

You can use the "Move block" instruction to also move source and destination areas of the STRING data type. If only the source area is STRING data type, the characters will be moved that are actually contained in the character string. Information on the actual and maximum length is also written to the destination area. If the source and destination area are each STRING data type, the current length of the character string in the destination area is set to the number of actually moved characters.

If you want to move the information on the maximum and actual length of a character string, specify the areas in bytes to the SRCBLK and DSTBLK parameters.

Rules for moving data blocks that are not runtime-relevant

The source area is in a data block in load memory that is not runtime-relevant. Data blocks that are not runtime-relevant are indicated by the keyword UNLINKED.

If a non-runtime-relevant data block is copied to the work memory with the "Move block" instruction and loaded at the same time, for example, by the programming device, the execution of the instruction can be delayed for several milliseconds. This results in a longer OB cycle and may trip the cycle monitoring.

If a non-runtime-relevant data block is moved with the "Move block" instruction and the move operation is interrupted, the execution of the instruction can no longer be continued.

If the instruction Read from data block in the load memory is available on your CPU, you must use this instruction to read data blocks that are not runtime-relevant from load memory. If you use the "Move block" instruction, error W#16#8092 is output.

Parameters

The following table shows the parameters of the "Move block" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRCBLK	Input	ANY	I, Q, M, D, L, P	Specifies the memory area to be moved (source area).
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.
DSTBLK	Output	ANY	I, Q, M, D, L, P	Specifies the memory area to which the block is to be moved (destination area).

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8092	The source or target area is only available in the load memory.
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Evaluating errors with output parameter RET_VAL (Page 5370)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Switching display formats in the program status (Page 2980)

UBLKMOV: Move block uninterruptible

Description

You can use the "Move block uninterruptible" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use ANY pointer to define the source and destination area.

The move operation cannot be interrupted by other operating system activities. As a result the alarm reaction time of the CPU can increase during the execution of the "Move block uninterruptible" instruction.

Note

In this instruction, you can only use data types which are valid on a CPU of the S7-300/400 series.

Memory areas

You can use the "Move block uninterruptible" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input table
- Process image output table

General rules for moving

The source and destination area must not overlap during the execution of the "Move block uninterruptible" instruction. If the source area is less than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is less than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a source or destination area defined as a formal parameter is less than a destination or source area specified on the SRCBLK or DSTBLK parameter, no data will be transferred.

If a block of data type BOOL is moved, the specified length of the area must be dividable by 8, otherwise it will not be possible to execute the instruction.

You can use the "Move block uninterruptible" instruction to move a maximum of 512 bytes. Note the CPU specific restrictions for this.

Note

If an actual source or destination area is smaller than a destination or source area specified on the SRCBLK or DSTBLK parameter, the system behavior depends on which CPU type is being used:

- The following applies to S7-300 CPUs:
  No data are transferred. The error code W#16#837F is output on the RET_VAL parameter. This error code has to be evaluated in your user program.
- The following applies to the S7-400 V4-CPUs (V4, V4 H-CPU and V4.5. H-CPU):
  No data are transferred. The error code W#16#8122 or W#16#8323 is output on the RET_VAL parameter. This error code has to be evaluated in your user program.
- For all other S7-400-CPUs:
  Data are transferred. The error code W#16#8122 or W#16#8323 is output on the RET_VAL parameter. This error code has to be evaluated in your user program.

Rules for moving character strings

You can use the "Move block uninterruptible" instruction to also move source and destination areas of the STRING data type. If only the source area is of data type STRING, the characters will be moved that are actually contained in the character string. Information on the actual and maximum length are not written in the destination area. If the source and destination area are each STRING data type, the current length of the character string in the destination area is

set to the number of actually moved characters. If areas of the data type STRING are moved, you have to specify "1" as area length.

Parameters

The following table shows the parameters of the "Move block uninterruptible" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
SRCBLK	Input	ANY	I, Q, M, D, L, P	Specifies the memory area to be moved (source area).
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during the execution of the instruction, an error code is output on the RET_VAL parameter.
DSTBLK	Output	ANY	I, Q, M, D, L, P	Specifies the memory area to which the block is to be moved (destination area).

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8091	The source area is in a data block that is not relevant for program execution.
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Evaluating errors with output parameter RET_VAL (Page 5370)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Switching display formats in the program status (Page 2980)

FILL: Fill block

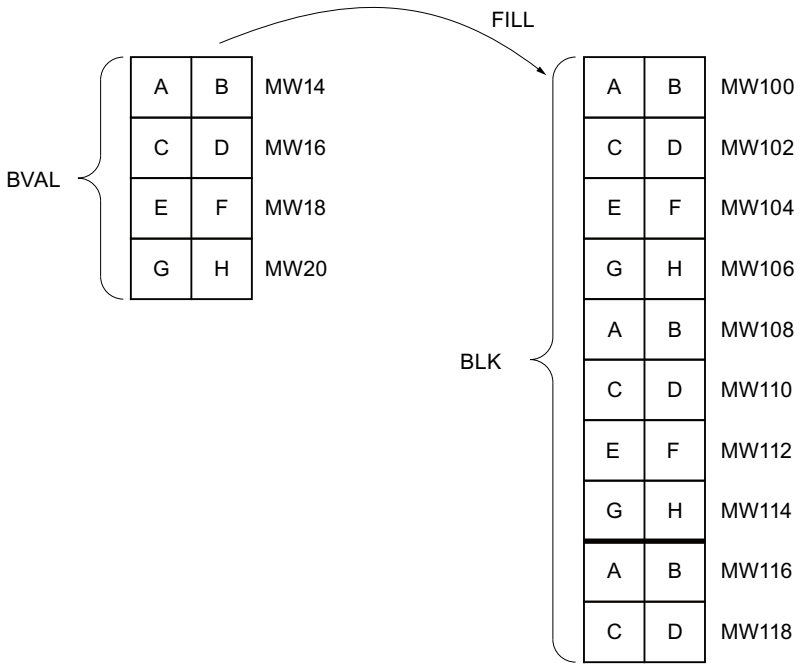
Description

You can use the "Fill block" instruction to fill a memory area (destination area) with the content of another memory area (source area). The "Fill block" instruction moves the content of the source area to the destination area until the destination area is completely written. The move operation takes place in the direction of ascending addresses. You use ANY pointer to define the source and destination area.

Note

In this instruction, you can only use data types which are valid on a CPU of the S7-300/400 series.

The following figure shows the principle of the move operation:



Example: The contents of the range MW100 to MW118 are to be preassigned with the contents of the memory words MW14 to MW20.

Consistency of the source data and the target data

Please note that during execution of the instruction "Fill block" that the source data remain unchanged, otherwise the consistency of the target data is not ensured.

Memory areas

You can use the "Fill block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input table
- Process image output table
- Data blocks not relevant for program execution

General rules for moving

The source and destination area must not overlap. If the destination block to be preset is not an integer multiple of the length of the input parameter BVAL, the destination block is nevertheless written up to the last byte.

If the destination area to be preset is smaller than the source area, the function only copies as much data as can be written to the destination area.

If the destination or source block actually present is smaller than the assigned memory area for the source or destination block (BVAL, BLK parameters), no data is transferred.

If the ANY pointer (source or destination) is of the data type BOOL, the length specified must be divisible by 8; otherwise the instruction is not executed.

If the destination block is of data type STRING, the instruction describes the entire string including the administration information.

Note

If an actual source or destination area is smaller than a destination or source area specified on the BVAL or BLK parameter, the system behavior depends on which CPU type is being used:

- The following applies to S7-300 CPUs:
  No data are transferred. The error code W#16#837F is output on the RET_VAL parameter. This error code has to be evaluated in your user program.
- The following applies to the S7-400 V4-CPUs (V4, V4 H-CPU and V4.5. H-CPU):
  No data are transferred. The error code W#16#8122 or W#16#8323 is output on the RET_VAL parameter. This error code has to be evaluated in your user program.
- For all other S7-400-CPUs:
  Data are transferred. The error code W#16#8122 or W#16#8323 is output on the RET_VAL parameter. This error code has to be evaluated in your user program.

Parameters

The following table shows the parameters of the "Fill block" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
BVAL	Input	ANY	I, Q, M, D, L	Specification of the memory area (source area), the content of which is used to fill the destination area at the BLK parameter.
RET_VAL	Output	INT	I, Q, M, D, L	Error information: If an error occurs during the execution of the instruction, an error code is output on the RET_VAL parameter.
BLK	Output	ANY	I, Q, M, D, L	Specification of the memory area that is filled with the content of the source area.

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Evaluating errors with output parameter RET_VAL (Page 5370)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Switching display formats in the program status (Page 2980)

Conversion operations

CONVERT: Convert value

Description

The "Convert value" instruction reads the content of the IN parameter and converts it according to the data types selected in the instruction box. The converted value is provided at output OUT.

For information on possible conversions, refer to the "Explicit conversion" section at "See also".

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- Errors, such as an overflow, occur during execution.

Parameter

The following table shows the parameters of the "Convert value" instruction:

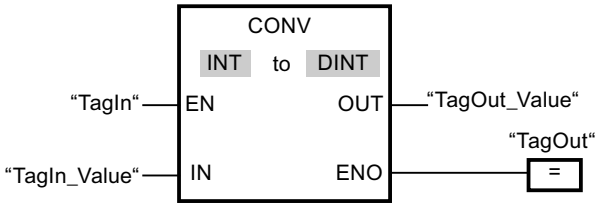
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Integers, floating-point numbers, BCD16, BCD32	I, Q, M, D, L, P or constant	Value to be converted.
OUT	Output	Integers, floating-point numbers, BCD16, BCD32	I, Q, M, D, L, P	Result of the conversion

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the operand "TagIn" has the signal state "1", the content of the operand "TagIn_Value" is read and converted to an integer (16 bits). The result is stored in the operand "TagOut_Value". The output "TagOut" is set to "1" if the instruction was executed without errors.

See also

Overview of the valid data types (Page 2236)
Basics of the EN/ENO mechanism (Page 2447)
Inserting FBD elements (Page 2621)
Selecting the data type of an FBD element (Page 2624)
Editing FBD elements (Page 2635)
Inserting operands in FBD instructions (Page 2643)
Basic information on the status word (Page 2453)
Setting the status bits in instructions with floating-point numbers (Page 2459)
Setting the status bits in instructions with integers (Page 2458)
Explicit conversion (Page 2430)

ROUND: Round numerical value

Description

You can use the "Round numerical value" instruction to round the value at input IN to the nearest integer. The instruction interprets the value at input IN as a floating-point number and converts this to the next higher integer of the data type DINT. If the input value is exactly between an even and odd number, the even number is selected. The result of the instruction is provided at output OUT and can be queried there.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the "Round numerical value" instruction:

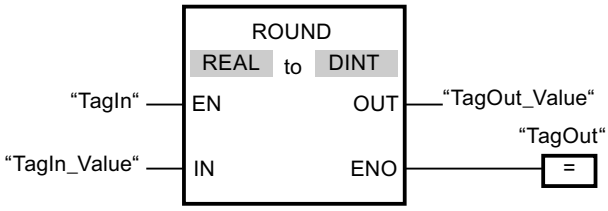
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	REAL	I, Q, M, D, L, P or constant	Input value to be rounded.
OUT	Output	DINT	I, Q, M, D, L, P	Result of the rounding

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value	
IN	TagIn_Value	1.50000000	-1.50000000
OUT	TagOut_Value	2	-2

If the operand "TagIn" has the signal state "1", the "Round numerical value" instruction is executed. The floating-point number at input "TagIn_Value" is rounded to the nearest even integer and sent to output "TagOut_Value". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with floating-point numbers (Page 2459)

CEIL: Generate next higher integer from floating-point number

Description

You can use the "Generate next higher integer from floating-point number" instruction to round the value at input IN to the next higher integer. The instruction interprets the value at input IN as a floating-point number and converts this to the next higher integer. The result of the instruction is provided at output OUT and can be queried there. The output value can be greater than or equal to the input value.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the instruction "Generate next higher integer from floating-point number":

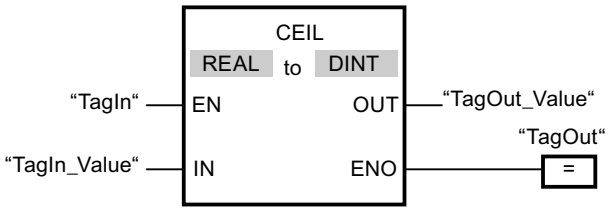
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result with the next higher integer

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value	
IN	TagIn_Value	0.50000000	-0.50000000
OUT	TagOut_Value	1	0

If the operand "TagIn" has the signal state "1", the instruction "Generate next higher integer from floating-point number" is executed. The floating-point number at input "TagIn_Value" is rounded to the next higher integer and sent to output "TagOut_Value". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with floating-point numbers (Page 2459)

FLOOR: Generate next lower integer from floating-point number

Description

You can use the "Generate next lower integer from floating-point number" instruction to round the value at input IN to the next lower integer. The instruction interprets the value at input IN as a floating-point number and converts this to the next lower integer. The result of the instruction is provided at output OUT and can be queried there. The output value can be less than or equal to the input value.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the instruction "Generate next lower integer from floating-point number":

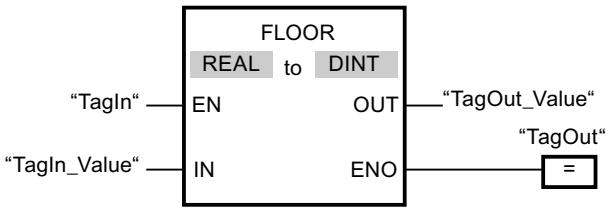
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Floating-point numbers	I, Q, M, D, L, P or constant	Input value
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Result with the next lower integer

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value	
IN	TagIn_Value	0.50000000	-0.50000000
OUT	TagOut_Value	0	-1

If the operand "TagIn" has the signal state "1", the instruction "Generate next lower integer from floating-point number" is executed. The floating-point number at input "TagIn_Value" is rounded to the next lower integer and displayed at output "TagOut_Value". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with floating-point numbers (Page 2459)

TRUNC: Truncate numerical value

Description

You can use the "Truncate numerical value" instruction to form an integer from the value at input IN. The value at input IN is interpreted as a floating-point number. The instruction selects only the integer part of the floating-point number and sends this to output OUT without decimal places.

Enable output ENO has the signal state "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- Errors, such as an overflow, occur during execution.

Parameters

The following table shows the parameters of the instruction "Truncate numerical value":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

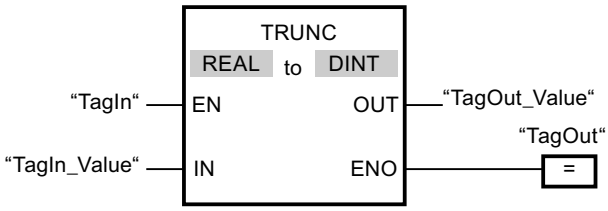
Parameters	Declaration	Data type	Memory area	Description
IN	Input	REAL	I, Q, M, D, L or constant	Input value
OUT	Output	Integers	I, Q, M, D, L	Integer component of the input value

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value	
IN	TagIn_Value	1.50000000	-1.50000000
OUT	TagOut_Value	1	-1

If the operand "TagIn" has the signal state "1", the "Truncate numerical value" instruction is executed. The integer part of the floating-point number at input "TagIn_Value" is converted to an integer and sent to output "TagOut_Value". If no errors occur during the execution of the instruction, the output "TagOut" is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

SCALE: Scale

Description

Use the "Scale" instruction to convert the integer at the IN parameter into a floating-point number, which can be scaled in physical units between a low limit value and a high limit value. You use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is scaled. The result of the instruction is output at the OUT parameter.

The "Scale" instruction works with the following equation:

OUT = [((FLOAT (IN) – K1)/(K2–K1)) * (HI_LIM–LO_LIM)] + LO_LIM

The values of the constants "K1" and "K2" are determined by the signal state of the BIPOLAR parameter. The following signal states are possible at the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case the "K1" constant has the value "-27648.0" and the "K2" constant the value "+27648.0".
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case the "K1" constant has the value "0.0" and the "K2" constant the value "+27648.0".

When the value at the IN parameter is greater than the value of the constant "K2", the result of the instruction is set to the value of the high limit (HI_LIM) and an error is output.

When the value at the IN parameter is less than the value of the constant "K1", the result of the instruction is set to the value of the low limit value (LO_LIM) and an error is output.

When the indicated low limit value is greater than the high limit value (LO_LIM > HI_LIM), the result is scaled in reverse proportion to the input value.

Parameters

The following table shows the parameters of the "Scale" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	INT	I, Q, M, D, L, P or constant	Input value to be scaled
HI_LIM	Input	REAL	I, Q, M, D, L, P or constant	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P or constant	Low limit

Parameter	Declaration	Data type	Memory area	Description
BIPOLAR	Input	BOOL	I, Q, M, D, L or constant	Indicates if the value at the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
OUT	Output	REAL	I, Q, M, D, L, P	Result of the instruction
RET_VAL	Output	WORD	I, Q, M, D, L, P	Error information

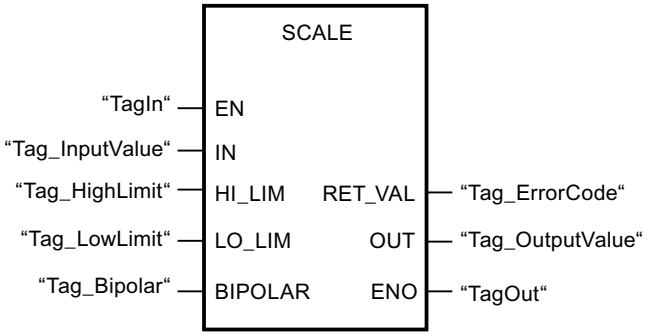
Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the constant "K2" or less than the value of the constant "K1".
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_InputValue	22
HI_LIM	Tag_HighLimit	100.0

Parameter	Operand	Value
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
OUT	Tag_OutputValue	50.03978588
RET_VAL	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 2236)

Evaluating errors with output parameter RET_VAL (Page 5370)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Switching display formats in the program status (Page 2980)

UNSCALE: Unscale

Description

The "Unscale" instruction unscales the floating-point number at the IN parameter into physical units between a low limit value and a high limit and converts them into integers. You use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is unscaled. The result of the instruction is output at the OUT parameter.

The "Unscale" instruction works with the following equation:

OUT = [((IN–LO_LIM)/(HI_LIM–LO_LIM)) * (K2–K1)] + K1

The values of the constants "K1" and "K2" are determined by the signal state of the BIPOLAR parameter. The following signal states are possible at the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case the "K1" constant has the value "-27648.0" and the "K2" constant the value "+27648.0".
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case the "K1" constant has the value "0.0" and the "K2" constant the value "+27648.0".

When the value at the IN parameter is greater than the value of the constant "HI_LIM", the result of the instruction is set to the value of the constant (K2) and an error is output.

When the value at the IN parameter is less than the value of the constant of the low limit (LO_LIM), the result of the instruction is set to the value of the constant (K1) and an error is output.

Parameters

The following table shows the parameters of the "Unscale" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Input	BOOL	I, Q, M, D, L	Enable output
IN	Input	REAL	I, Q, M, D, L, P or constant	Input value to be unscaled to an integer value
HI_LIM	Input	REAL	I, Q, M, D, L, P or constant	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P or constant	Low limit
BIPOLAR	Input	BOOL	I, Q, M, D, L or constant	Indicates if the value at the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
OUT	Output	INT	I, Q, M, D, L, P	Result of the instruction
RET_VAL	Output	WORD	I, Q, M, D, L, P	Error information

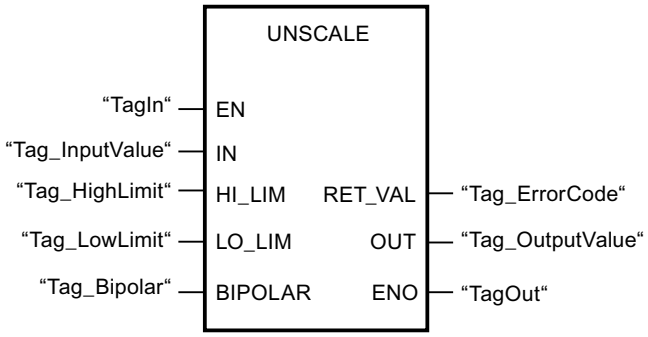
Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the high limit (HI_LIM) or less than the value of the low limit (LO_LIM).
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
OUT	Tag_OutputValue	22
RET_VAL	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 2236)

Evaluating errors with output parameter RET_VAL (Page 5370)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Switching display formats in the program status (Page 2980)

Program control operations

JMP: Jump if RLO = 1

Description

You can use the "Jump if RLO = 1" instruction to interrupt the linear execution of the program and resume it in another network. The target network must be identified by a jump label (LABEL). The jump label description is entered in the placeholder above the instruction box.

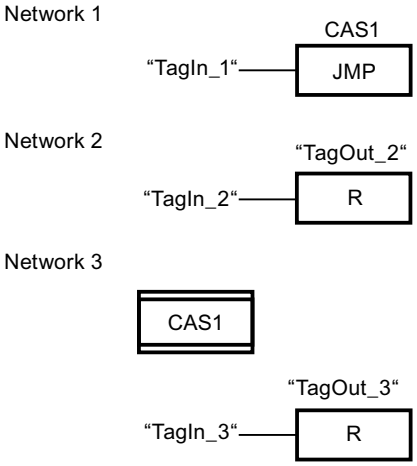
The specified jump label must be in the same block in which the instruction is executed. The name you specify can only occur once in the block. Only one jumping coil can occur in a network.

If the result of logic operation (RLO) at the input of the instruction is "1", the jump to the network identified by the jump label is executed. The jump direction can be towards higher or lower network numbers.

If the condition at the input of the instruction is not fulfilled (RLO = 0), execution of the program continues in the next network.

Example

The following example shows how the instruction works:



If the operand "TagIn_1" has the signal state "1", the instruction "Jump if RLO = 1" is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. If input "TagIn_3" has the signal state "1", output "TagOut_3" is reset.

See also

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

JMPN: Jump if RLO = 0

Description

You can use the instruction "Jump if RLO = 0" to interrupt the linear execution of the program and resume it in another network, when the result of logic operation at the input of the instruction is "0". The target network must be identified by a jump label (LABEL). The jump label description is entered in the placeholder above the instruction box.

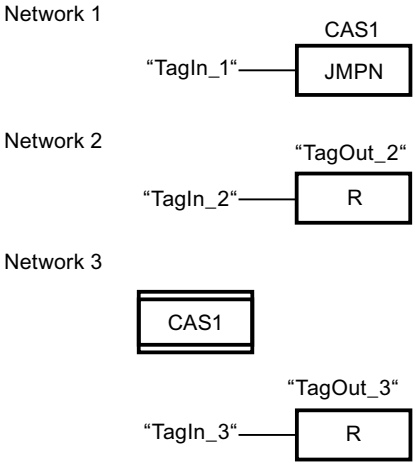
The specified jump label must be in the same block in which the instruction is executed. The name you specify can only occur once in the block. Only one jumping coil can occur in a network.

If the result of logic operation (RLO) at the input of the instruction is "0", the jump to the network identified by the jump label is executed. The jump direction can be towards higher or lower network numbers.

If the result of the logic operation RLO at the input of the instruction is "1", execution of the program continues in the next network.

Example

The following example shows how the instruction works:



If the operand "TagIn_1" has the signal state "0", the instruction "Jump if RLO = 0" is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. If input "TagIn_3" has the signal state "1", output "TagOut_3" is reset.

See also

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

LABEL: Jump label

Description

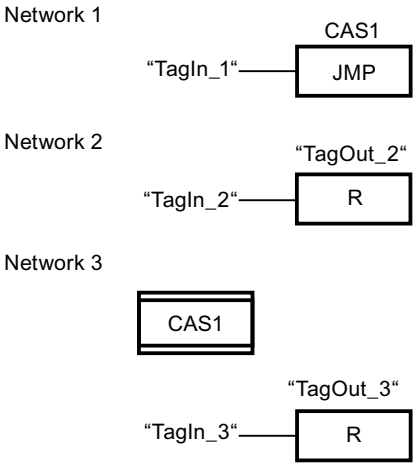
The jump label identifies a destination network in which the execution of the program can be resumed after the execution of a jump instruction.

The jump label and the instruction in which the jump label is specified must be located in the same block. The name of a jump label can only be assigned once in a block.

Only one jump label can be placed in a network. Each jump label can jump to several locations.

Example

The following example shows how the instruction works:



If the operand "TagIn_1" has the signal state "1", the instruction "Jump if RLO = 1" is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. If input "TagIn_3" has the signal state "1", output "TagOut_3" is reset.

See also

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

RET: Return

Description

You can use the instruction "Return" to stop the execution of a block.

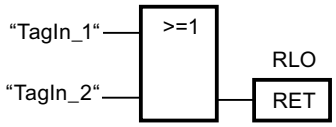
If the result of logic operation (RLO) at the input of the instruction "Return" is "1", the execution of the program is terminated in the currently called block and continued after the call function in the calling block (for example, in the calling OB). The status (ENO) of the call function is determined by the RLO. The RLO is shown via the instruction as a fixed parameter value. If the RLO at the input of the "Return" instruction is "0", the instruction is not executed. In this case, the execution of the program continues in the next network of the called block.

If the "Return" instruction in the last network of the called block is programmed, the ENO enable output of the call function is set to the current signal state of the RLO.

If an organization block is terminated by the "Return" instruction, the CPU goes on working in the system program.

Example

The following example shows how the instruction works:



If the operands "TagIn_1" or "TagIn_2" have signal state "1", the instruction "Return" is executed. Program execution in the called block is terminated and continues in the calling block. The enable output ENO of the call function is reset to signal state "1".

See also

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

OPN: Open global data block

Description

You can use the "Open global data block" instruction to open a global data block (DB). The number of the data block is transferred to the DB register. Subsequent DB commands access the relevant blocks depending on the content of the registers.

Parameters

The following table shows the parameters of the "Open global data block" instruction:

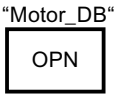
Parameter	Declaration	Data type	Memory area	Description
<data block>	Input	DB_ANY	-	Data block to be opened

For additional information on valid data types, refer to "See also".

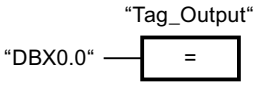
Example

The following example shows how the instruction works:

Network 1



Network 2



The "Motor_DB" data block is called in network 1. The number of the data block is transferred to the DB register. The "DBX0.0" operand is queried in network 2. The signal state of the "DBX0.0" operand is assigned to the "Tag_Output" operand.

See also

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

OPNI: Open instance data block

Description

You can use the "Open instance data block" instruction to open a data block (DB). The number of the data block is transferred to the DI register. Subsequent DI commands access the relevant blocks depending on the content of the registers.

Parameters

The following table shows the parameters of the "Open instance data block" instruction:

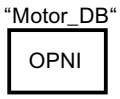
Parameter	Declaration	Data type	Memory area	Description
<data block>	Input	DB_ANY	-	Data block to be opened

For additional information on valid data types, refer to "See also".

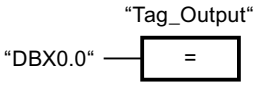
Example

The following example shows how the instruction works:

Network 1



Network 2



The "Motor_DB" data block is called in network 1. The number of the data block is transferred to the DI register. The "DBX0.0" operand is queried in network 2. The signal state of the "DBX0.0" operand is assigned to the "Tag_Output" operand.

See also

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Runtime control

COMPRESS: Compress CPU memory

Description

The "Compress CPU memory" instruction is used to compress the working memory of the CPU or the RAM load memory. During this compression the gaps caused by deleting or loading blocks are filled by pushing together the existing blocks.

The compression operation is started by calling the "Compress CPU memory" instruction. The compression operation is divided into several program cycles. The compression operation is monitored by the values of the individual parameters. If value "1" is on the BUSY parameter of the instruction, the compression operation is still being executed. The value "1" on the DONE parameter reports that the compression operation has been completed. Error information can be queried on the RET_VAL parameter.

If an externally started compression operation has already been executed, the calling of the "Compress CPU memory" instruction will cause an error to be displayed.

Parameters

The following table shows the parameters of the "Compress CPU memory" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
RET_VAL	Output	INT	I, Q, M, D, L	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.
BUSY	Output	BOOL	I, Q, M, D, L	Information whether compression operation is active.
DONE	Output	BOOL	I, Q, M, D, L	Information whether compression operation was completed.

For additional information on valid data types, refer to "See also".

Parameters RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error The status of the compression operation can be queried on the BUSY and DONE parameters.
8091	An externally started compression operation is already being executed.
8092	The instruction cannot be executed, because one of the following conditions applies: • The "Delete block" function is active. • A test and commissioning function affects a block (for example, status). • The "Copy blocks" function is enabled by an external trigger. • The H-CPU is running a linking or update routine.
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Evaluating errors with output parameter RET_VAL (Page 5370)

Switching display formats in the program status (Page 2980)

RE_TRIGR: Restart cycle monitoring time

Description

You can use the "Restart cycle monitoring time" instruction to restart the cycle monitoring of the CPU. The cycle time monitoring then starts over again for the duration you have set in the CPU configuration. By restarting the maximum cycle time, you can prevent errors being triggered or the CPU changing to STOP.

The "Restart cycle monitoring time" instruction can be used in blocks of the priority class 1 (cyclic OBs) and in the blocks called within it.

If the instruction is called in a block with a higher priority, for example a hardware interrupt, diagnostics interrupt or a cyclic interrupt, the instruction is not executed and the ENO enable output is set to signal state "0".

Note

Make sure that you do not create an infinite loop in the cyclical program execution, i.e. in OB1, when you use the "Restart cycle monitoring time" instruction. Otherwise, the CPU will not reach the cycle check point. As a result, it may not be possible to execute certain CPU functions (e.g. process image update).

Parameters

The instruction "Restart cycle monitoring time" has no parameters.

See also

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

STP: Exit program

Description

The instruction "Exit program" instruction is used to set the CPU to STOP mode and therefore to terminate the execution of the program. The effects of changing from RUN to STOP depend on the CPU configuration.

When the (RLO) at the input of the instruction is "1", the CPU changes to STOP mode and the execution of the program is terminated. The signal state at the output of the instruction is not evaluated.

If the RLO at the input of the instruction is "0", then the instruction will not be executed.

Parameters

The instruction "Exit program" has no parameters.

See also

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

WAIT: Configure time delay

Description

The "Configure time delay" instruction is used to halt the execution of the program for a set period of time. You indicate the period of time in microseconds on the WT parameter of the instruction.

You can configure time delays of up to 32767 microseconds (µs). The smallest possible delay time depends on the respective CPU and corresponds to the execution time of the "Configure time delay" instruction.

The execution of the instruction can be interrupted by higher priority events.

The "Configure time delay" instruction supplies no error information.

Parameters

The following table shows the parameters of the "Configure time delay" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
WT	Input	INT	I, Q, M, D, L, P or constant	Time delay in microseconds (µs)

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

PROTECT: Change protection level

Description

The instruction "Change protection level" is used to change the configured protection level "1" of the CPU.

The configured protection level "1" with selected option "Removable with password", means that you have to configure a password. This password enables you to write-access during operation the CPU which was read-only protected with the instruction "Change Protection Level" (e. g. Load).

You specify the protection level on the MODE parameter of the instruction. The current protection level of your CPU is displayed in the inspector window under "Properties > Protection".

The following table shows the meaning of the protection levels.

Protection level	Meaning
1	All programming device functions are permitted. Any existing blocking of the password authentication is canceled by the instruction "Change protection level".
2	• The program and the configuration of the CPU cannot be changed. The program in the CPU can be read. • The functions for process control, process monitoring, and process communication are permitted. • All informational functions are permitted. • If you know the valid password you can cancel the set write protection. • Any existing blocking of the password authentication is canceled by the instruction "Change protection level".
3	• You cannot cancel the set read and write protection, even if you know the valid password. • If a valid connection exists at the time when the instruction "Change protection level" is called, the call of the instruction does not affect this connection.

If you have configured protection level "2" or "3" for your CPU, calling the "Change protection level" instruction has no effect.

You can use the SSL parts list with the SSL ID W#16#0232 and index W#16#0004 to check whether protection level "2" or "3" is still in effect after the execution of the "Change protection level" instruction.

Note

You can only use the "Change protection level" instruction to set a protection level that is equal to or higher than the protection level you have configured with STEP 7 "Configure hardware. You have, for example, configured protection level 3 and then called the instruction "Change protection level" with the MODE parameter = 12. This blocks the password authentication. If the instruction is called again with the parameter MODE = 0 or 1, protection level 3 is retained and the read and write protection canceled by entry of the valid password.

Effects of the operating state transitions on the protection level

The following table shows what effects the operating state transitions have on the protection level set with the "Change protection level" instruction.

Action	Effects on the protection level
• Operating mode switch to "STOP" • Unbuffered POWER ON • Memory reset with operating mode switch • Configuration and/or programming changes in "STOP" mode of the S7-400 CPU	The protection level is set to "1". All programming device functions are permitted.
• Cold restart • Warm restart	The operating system sets the protection level to "1". All programming device functions are permitted. If required, you can set the protection level in your program to "2" or "3" by calling the "Change protection level" instruction.
Buffered POWER ON	Remains unchanged
Operating mode transition RUN/STARTUP/HOLD -> STOP (by calling the instruction "Exit program" (STP), due to a program error without corresponding error OB or by an operator action).	Remains unchanged
S7-400: Restart by user action or automatically following POWER ON	Remains unchanged

Parameters

The following table shows the parameters of the "Change protection level" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
MODE	Input	WORD	I, Q, M, D, L or constant	Protection level Possible values: • W#16#0000: Setting protection level "1" • W#16#0001: Setting protection level "2" with password authentication • W#16#000C: Setting protection level "3" with password authentication Constants can also be specified at the MODE parameter.
RET_VAL	Output	INT	I, Q, M, D, L	If an error occurs during processing, information on the error is output.

Parameters RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	Job executed without error
8090	Invalid value at parameter MODE.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Switching display formats in the program status (Page 2980)

CiR: Control CiR process

Description

The "Control CiR process" instruction is used to influence the CiR process in the user program as follows:

- Disable CiR process In this case, the download of a modified configuration from the programming device to the CPU is always rejected. The disable is maintained until you cancel it again with the "Control CiR process" instruction.
- Disable the CiR process conditionally by specifying a high limit for the CiR synchronization time on the FRZ_TIME parameter. In this case, the download of a modified configuration from the programming device to the CPU is only accepted if the CiR synchronization time calculated in the CPU does not exceed the specified high limit.
- Enable CiR process The high limit of the CiR synchronization time is set to the default value 1 000 ms.
- Determine whether the CiR process is enabled. If the CiR process is enabled or conditionally enabled, you will get the current high limit for the CiR synchronization time in the A_FT parameter of the instruction.

Note

During the CiR synchronization period the outputs are frozen and the inputs are not evaluated.

Parameters

The following table shows the parameters of the "Control CiR process" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
MODE	Input	BYTE	I, Q, M, D, L or constant	Job identifier Valid values on the MODE parameter are: • 0: Information function • 1: Enable CiR process • 2: Disable CiR process • 3: Disable CiR conditionally. You specify the high limit of the CiR synchronization time in FRZ_TIME.
FRZ_TIME	Input	TIME	I, Q, M, D, L or constant	High limit of the CiR synchronization time in milliseconds. Permitted range of values: 60 ... 2500 ms (default: 1000 ms) The FRZ_TIME parameter is only relevant when the value "3" is set on the MODE parameter.
RET_VAL	Output	INT	I, Q, M, D, L	Error information When the MODE parameter is "0", the RET_VAL parameter supplies information on the enabling of the CiR process.
A_FT	Output	TIME	I, Q, M, D, L	Currently valid high limit of the CiR synchronization time

Parameters RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	Job executed without error This error code can be output only if the value "1", "2", or "3" is set at the MODE parameter.
0001	CiR is enabled. This error code can be output only if the value "0" is set on the MODE parameter.
0002	The CiR process is disabled. This error code can be output only if the value "0" is set on the MODE parameter.
0003	CiR is disabled conditionally. This error code can be output only if the value "0" is set on the MODE parameter.

Error code* (W#16#...)	Explanation
8001	The CPU is not ready for a CiR operation. You are using an H CPU in an H system (stand-alone), or you are working with a standard CPU in multicomputing mode.
8002	The value at the MODE parameter is invalid.
8003	The value at the FRZ_TIME parameter is invalid.
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

Example

The "Control CiR process" instruction can be used, for example, to ensure that no CiR process is initiated during periods in which maximum CPU performance is required for the execution of the process.

To use the "Control CiR process" for this, the instruction must be called before the start of the period with increased process activity and executed with the value "2" (disable CiR process) on the MODE parameter. The CiR process is then disabled.

To enable the CiR process again after the completion of the period with increased process activity, you have to again call the "Control CiR process" instruction and execute it with the value "1" on the MODE parameter. Alternatively, you can also set the value "3" on the MODE parameter to disable the CiR process conditionally.

See also

Overview of the valid data types (Page 2236)

Evaluating errors with output parameter RET_VAL (Page 5370)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Switching display formats in the program status (Page 2980)

Word logic operations

AND: AND logic operation

Description

You can use the instruction "AND logic operation" to link the value at input IN1 to the value at input IN2 bit-by-bit by AND logic and query the result at the output OUT.

When the instruction is executed, bit 0 of the value at input IN1 is linked by AND logic to bit 0 of the value at input IN2. The result is stored in bit 0 of output OUT. The same logic operation is executed for all other bits of the specified values.

The result bit has the signal state "1" only when both of the bits in the logic operation also have signal state "1". If one of the two bits of the logic operation has signal state "0", the corresponding result bit is reset.

Parameters

The following table shows the parameters of the instruction "AND logic operation":

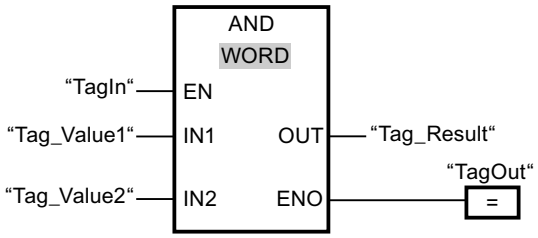
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	WORD, DWORD	I, Q, M, D, L, P or constant	First value for logic operation
IN2	Input	WORD, DWORD	I, Q, M, D, L, P or constant	Second value for logic operation
OUT	Output	WORD, DWORD	I, Q, M, D, L, P	Result of the instruction

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN1	Tag_Value1	01010101 01010101
IN2	Tag_Value2	00000000 00001111
OUT	Tag_Result	00000000 00000101

If the operand "TagIn" has the signal state "1", the instruction "AND logic operation" is executed. The value of operand "Tag_Value1" is linked by AND to the value of the operand "Tag_Value2". The result is mapped bit-for-bit and sent to the operand "Tag_Result". The enable output ENO and the output "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 2236)
Basics of the EN/ENO mechanism (Page 2447)
Inserting FBD elements (Page 2621)
Selecting the data type of an FBD element (Page 2624)
Editing FBD elements (Page 2635)
Inserting operands in FBD instructions (Page 2643)
Setting the status bit for word logic operations (Page 2462)
Basic information on the status word (Page 2453)

OR: OR logic operation

Description

You can use the instruction "OR logic operation" to link the value at input IN1 to the value at input IN2 bit-by-bit by OR logic and query the result at the output OUT.

When the instruction is executed, bit 0 of the value at input IN1 is linked by OR logic to bit 0 of the value at input IN2. The result is stored in bit 0 of output OUT. The same logic operation is executed for all bits of the specified tags.

The result bit has the signal state "1" when at least one of the two bits in the logic operation has signal state "1". If both of the bits of the logic operation have signal state "0", the corresponding result bit is reset.

Parameters

The following table shows the parameters of the instruction "OR logic operation":

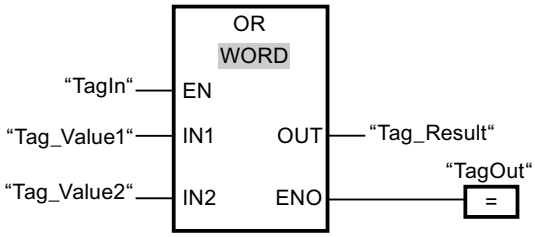
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	WORD, DWORD	I, Q, M, D, L, P or constant	First value for logic operation
IN2	Input	WORD, DWORD	I, Q, M, D, L, P or constant	Second value for logic operation
OUT	Output	WORD, DWORD	I, Q, M, D, L, P	Result of the instruction

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN1	Tag_Value1	01010101 01010101
IN2	Tag_Value2	00000000 00001111
OUT	Tag_Result	01010101 01011111

If the operand "TagIn" has the signal state "1", the instruction "OR logic operation" is executed. The value of operand "Tag_Value1" is linked by OR to the value of the operand "Tag_Value2". The result is mapped bit-for-bit and sent to the operand "Tag_Result". The enable output ENO and the output "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Setting the status bit for word logic operations (Page 2462)

Basic information on the status word (Page 2453)

XOR: EXCLUSIVE OR logic operation

Description

You can use the instruction "EXCLUSIVE OR logic operation" to link the value at input IN1 to the value at input IN2 bit-by-bit by EXCLUSIVE OR logic and query the result at the output OUT.

When the instruction is executed, bit 0 of the value at input IN1 is linked by EXCLUSIVE OR logic to bit 0 of the value at input IN2. The result is stored in bit 0 of output OUT. The same logic operation is executed for all other bits of the specified value.

The result bit has the signal state "1" when one of the two bits in the logic operation has signal state "1". If both of the bits of the logic operation have signal state "1" or "0", the corresponding result bit is reset.

Parameters

The following table shows the parameters of the instruction "EXCLUSIVE OR logic operation":

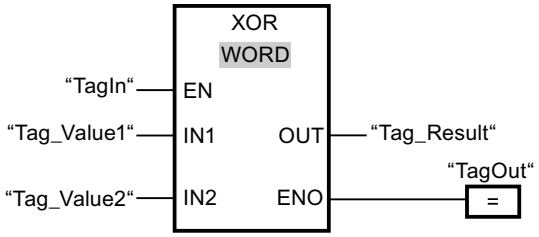
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	WORD, DWORD	I, Q, M, D, L, P or constant	First value for logic operation
IN2	Input	WORD, DWORD	I, Q, M, D, L, P or constant	Second value for logic operation
OUT	Output	WORD, DWORD	I, Q, M, D, L, P	Result of the instruction

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN1	Tag_Value1	01010101 01010101
IN2	Tag_Value2	00000000 00001111
OUT	Tag_Result	01010101 01011010

If the operand "TagIn" has the signal state "1", the instruction "EXCLUSIVE OR logic operation" is executed. The value of operand "Tag_Value1" is linked by EXCLUSIVE OR to the value of the operand "Tag_Value2". The result is mapped bit-for-bit and sent to the operand "Tag_Result". The enable output ENO and the output "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 2236)
Basics of the EN/ENO mechanism (Page 2447)
Inserting FBD elements (Page 2621)
Selecting the data type of an FBD element (Page 2624)
Editing FBD elements (Page 2635)
Inserting operands in FBD instructions (Page 2643)
Setting the status bit for word logic operations (Page 2462)
Basic information on the status word (Page 2453)

INV: Create ones complement

Description

You can use the instruction "Create ones complement" to invert the signal status of the bits at input IN. When the instruction is processed, the value at input IN is linked to EXCLUSIVE OR by a hexadecimal mask (W#16#FFFF for 16-bit numbers or DW#16#FFFF FFFF for 32-bit number). This inverts the signal state of the individual bits that are then stored at output OUT.

Parameters

The following table shows the parameters of the instruction "Create ones complement":

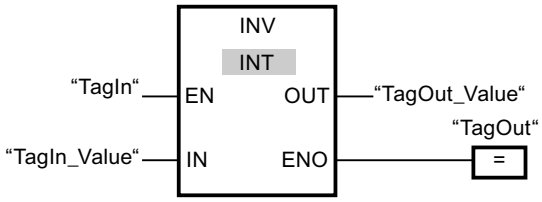
Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Integers	I, Q, M, D, L, P or constant	Input value
OUT	Output	Integers	I, Q, M, D, L, P	Ones complement of the value at input IN

You can select the data type of the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
IN	TagIn_Value	W#16#000F
OUT	TagOut_Value	W#16#FFF0

If the operand "TagIn" has the signal state "1", the "Create ones complement" instruction is executed. The instruction inverts the signal state of the individual bits at input "TagIn_Value" and writes the result to output "TagOut_Value". The enable output ENO and the output "TagOut" are set to signal state "1".

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

DECO: Decode

Description

You can use the "Decode" instruction to set a bit in the output value specified by the input value.

The "Decode" instruction reads the value at the IN input and sets the bit in the output value whose bit position corresponds to the read value. The other bits in the output value are filled with zeroes. If the value at input IN is greater than 31, a modulo 32 instruction is executed.

Parameters

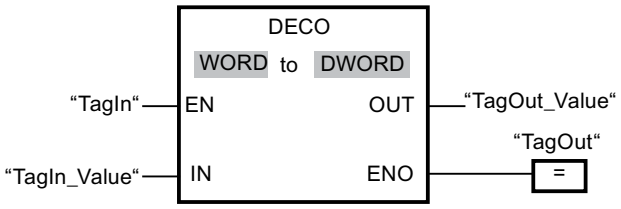
The following table shows the parameters of the "Decode" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	WORD	I, Q, M, D, L, P or constant	Position of the bit in the output value which is set.
OUT	Output	DWORD	I, Q, M, D, L, P	Output value

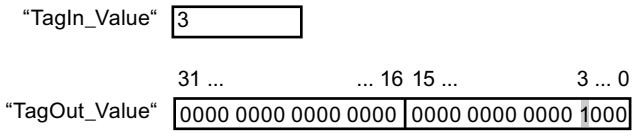
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following figure shows how the instruction works using specific operand values:



If the operand "TagIn" has the signal state "1", the "Decode" instruction is executed. The instruction reads bit number "3" from the value of the operand "TagIn_Value" at the input and sets the third bit to the value of the operand "TagOut_Value" at the output.

If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

ENCO: Encode

Description

The instruction "Encode" is used to read the bit number of the lowest bit in the input value and output it to the output OUT.

The "Encode" instruction selects the least significant bit of the value at the IN input and writes its bit number to the tag in the OUT output.

Parameters

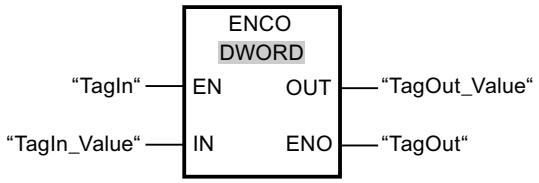
The following table shows the parameters of the "Encode" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	DWORD	I, Q, M, D, L, P or constant	Input value
OUT	Output	INT	I, Q, M, D, L, P	Output value

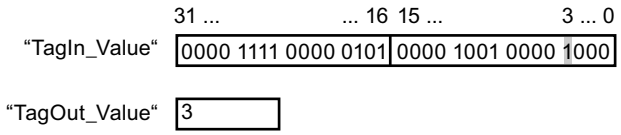
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following figure shows how the instruction works using specific operand values:



If the operand "TagIn" has the signal state "1", the "Encode" instruction is executed. The instruction selects the least significant bit at the "TagIn_Value" input and writes bit position "3" to the tag in the "TagOut_Value" output.

If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

SEL: Select

Description

Depending on a switch (input G), the "Select" instruction selects one of the inputs IN0 or IN1 and moves its content to the output OUT. When the input G has the signal state "0", the value at the input IN0 is moved. When the input G has the signal state "1", the value at the input IN1 is moved to the output OUT.

The instruction can only be executed if the enable input EN has the signal state "1" and the tags at all parameters are of the same data type.

Parameters

The following table shows the parameters of the "Select" instruction:

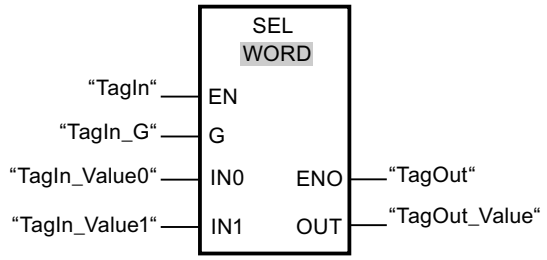
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
G	Input	BOOL	I, Q, M, D, L, T, C	Switch
IN0	Input	Bit strings, integers, floating-point numbers, timers, CHAR, TOD, DATE	I, Q, M, D, L, P	First input value
IN1	Input	Bit strings, integers, floating-point numbers, timers, CHAR, TOD, DATE	I, Q, M, D, L, P	Second input value
OUT	Output	Bit strings, integers, floating-point numbers, timers, CHAR, TOD, DATE	I, Q, M, D, L, P	Result

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
G	TagIn_G	0	1
IN0	TagIn_Value0	W#16#0000	W#16#4C
IN1	TagIn_Value1	W#16#FFFF	W#16#5E
OUT	TagOut_Value	W#16#0000	W#16#5E

If the operand "TagIn" has the signal state "1", the instruction "Select" is executed. Based on the signal state at the input "TagIn_G", the value at input "TagIn_Value0" or "TagIn_Value1" is selected and copied to output "TagOut_Value". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Shift and rotate

SHR: Shift right

Description

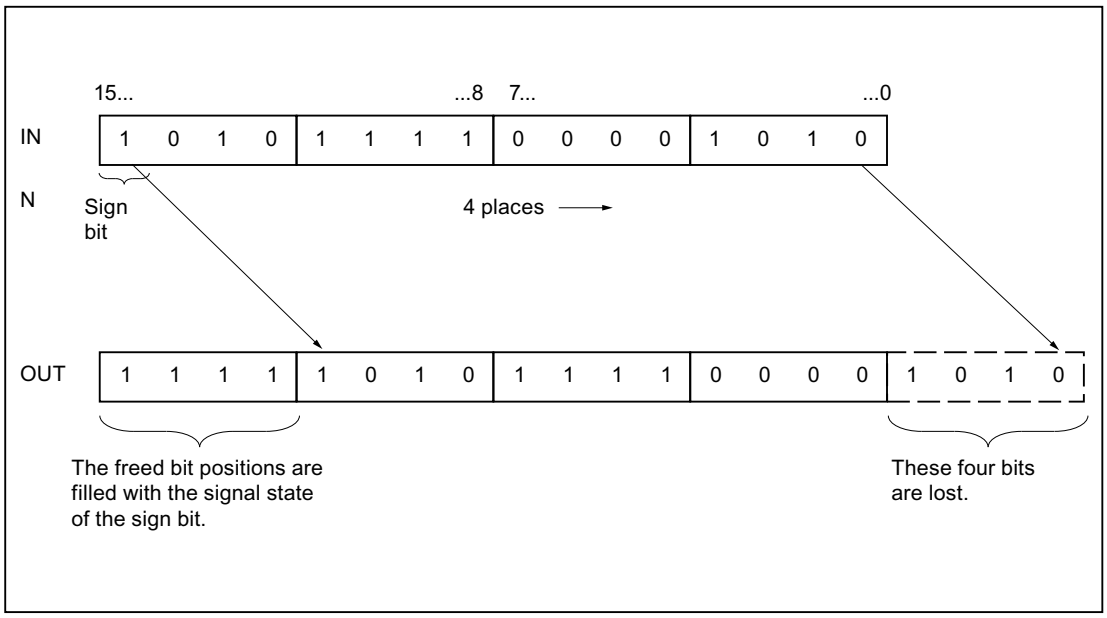
You can use the "Shift right" instruction to shift the content of the operand at the input IN bit-by-bit to the right and query the result at the OUT output. The parameter N is used to specify the number of bit positions by which the specified value should be shifted.

If the value at the parameter N is "0", the value at input IN is copied to the operand at output OUT.

If the value at the parameter N is greater than the number of bit positions, the operand value at input IN is shifted to the right by the available number of bit positions.

The freed bit positions in the left area of the operand are filled by zeroes when values without signs are shifted. If the specified value has a sign, the free bit positions are filled with the signal state of the sign bit.

The following figure shows how the content of an operand of the INT data type is shifted by four bit positions to the right:



Parameters

The following table shows the parameters of the instruction "Shift right":

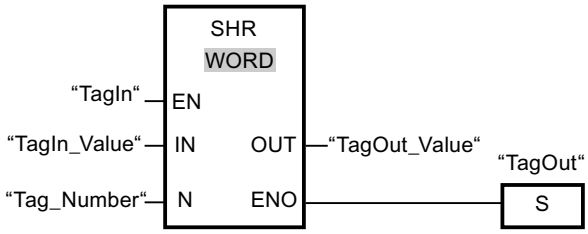
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	BYTE, WORD, DWORD, INT, DINT	I, Q, M, D, L, P or constant	Value to be shifted
N	Input	BYTE, WORD, INT	I, Q, M, D, L, P or constant	Number of bit positions by which the value is shifted
OUT	Output	BYTE, WORD, DWORD, INT, DINT	I, Q, M, D, L	Result of the instruction

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
N	Tag_Number	3
OUT	TagOut_Value	0000 0111 1111 0101

If the operand "TagIn" has the signal state "1", the instruction "Shift right" is executed. The content of the operand "TagIn_Value" is shifted three bit positions to the right. The result is sent at the output "TagOut_Value". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Setting the status bits for shift and rotate instructions (Page 2462)

Basic information on the status word (Page 2453)

SHL: Shift left

Description

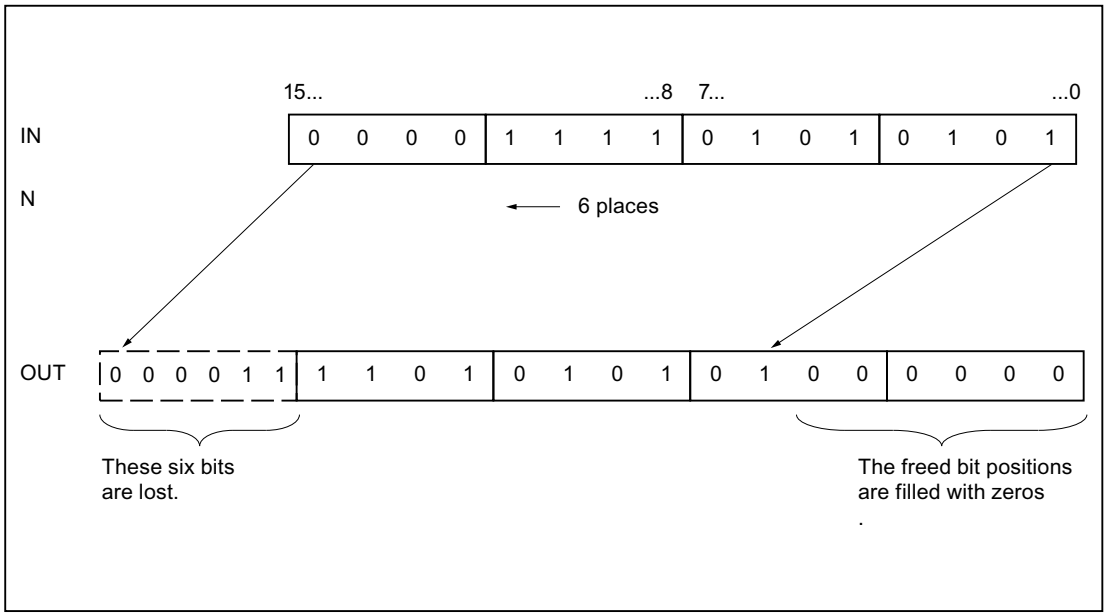
You can use the "Shift left" instruction to shift the content of the operand at the input IN bit-by-bit to the left and query the result at the OUT output. The parameter N is used to specify the number of bit positions by which the specified value should be shifted.

If the value at the parameter N is "0", the value at input IN is copied to the operand at output OUT.

If the value at the parameter N is greater than the number of bit positions, the operand value at input IN is shifted to the left by the available number of bit positions.

The bit positions in the right part of the operand freed by shifting are filled with zeros.

The following figure shows how the content of an operand of the WORD data type is shifted by six bit positions to the left:



Parameters

The following table shows the parameters of the instruction "Shift left":

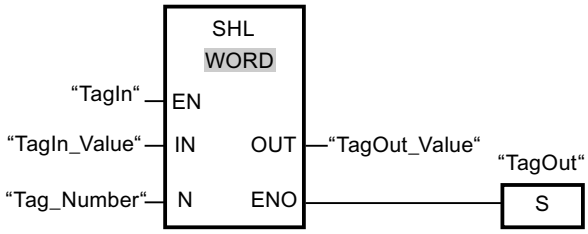
Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	BYTE, WORD, DWORD, INT, DINT	I, Q, M, D, L, P or constant	Value to be shifted
N	Input	BYTE, WORD, INT	I, Q, M, D, L, P or constant	Number of bit positions by which the value is shifted
OUT	Output	BYTE, WORD, DWORD, INT, DINT	I, Q, M, D, L	Result of the instruction

You can select the data type for the instruction from the "<???>" drop-down list of the instruction box.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
N	Tag_Number	4
OUT	TagOut_Value	1111 1010 1111 0000

If the operand "TagIn" has the signal state "1", the instruction "Shift left" is executed. The content of the operand "TagIn_Value" is shifted four bit positions to the left. The result is sent at the output "TagOut_Value". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Setting the status bits for shift and rotate instructions (Page 2462)

Basic information on the status word (Page 2453)

ROR: Rotate right

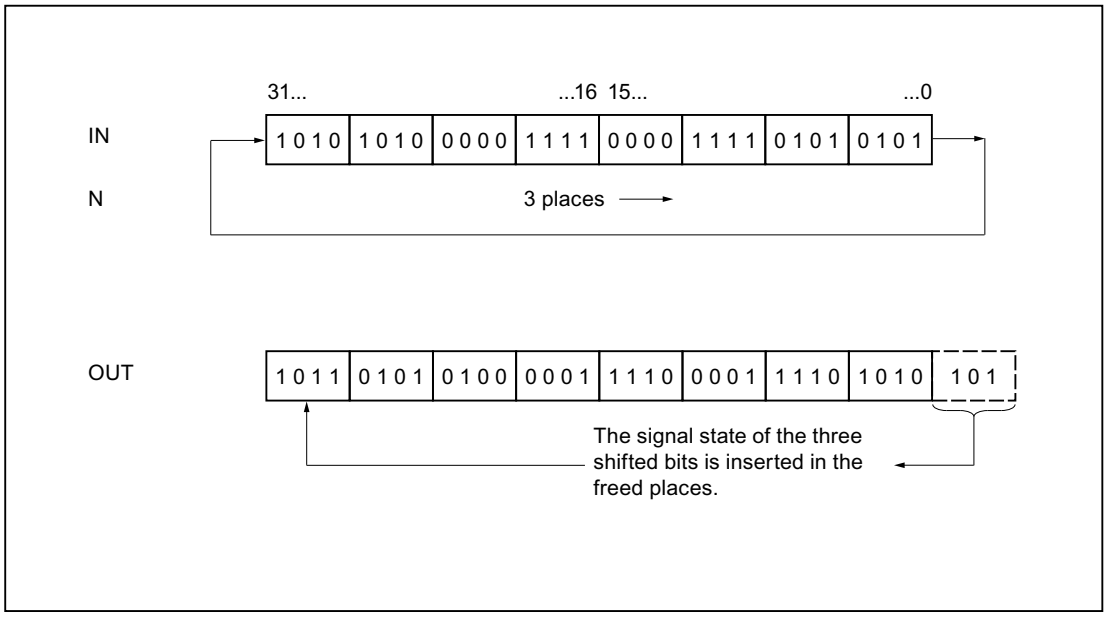
Description

You can use the "Rotate right" instruction to rotate the content of the operand at the input IN bit-by-bit to the right and query the result at the OUT output. The parameter N is used to specify the number of bit positions by which the specified value should be rotated. The bit positions freed by rotating are filled with the bit positions that are pushed out.

If the value at the parameter N is "0", the value at input IN is copied to the operand at output OUT.

If the value at the N parameter is greater than the number of available bit positions, the operand value at IN input is nevertheless rotated by the specified number of bit positions.

The following figure shows how the content of an operand of the DWORD data type is rotated three bit positions to the right:



Parameters

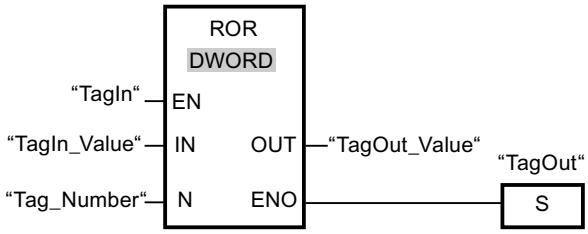
The following table shows the parameters of the instruction "Rotate right":

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	BYTE, WORD, DWORD, DINT	I, Q, M, D, L, P or constant	Value to be rotated
N	Input	BYTE, WORD, INT	I, Q, M, D, L, P or constant	Number of bit positions by which the value is rotated
OUT	Output	BYTE, WORD, DWORD, DINT	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	TagIn_Value	1010 1010 0000 1111 0000 1111 0101 0101
N	Tag_Number	5
OUT	TagOut_Value	1010 1101 0101 0000 0111 1000 0111 1010

If the operand "TagIn" has the signal state "1", the instruction "Rotate right" is executed. The content of the operand "TagIn_Value" is rotated five bit positions to the right. The result is sent at the output "TagOut_Value". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Setting the status bits for shift and rotate instructions (Page 2462)

Basic information on the status word (Page 2453)

ROL: Rotate left

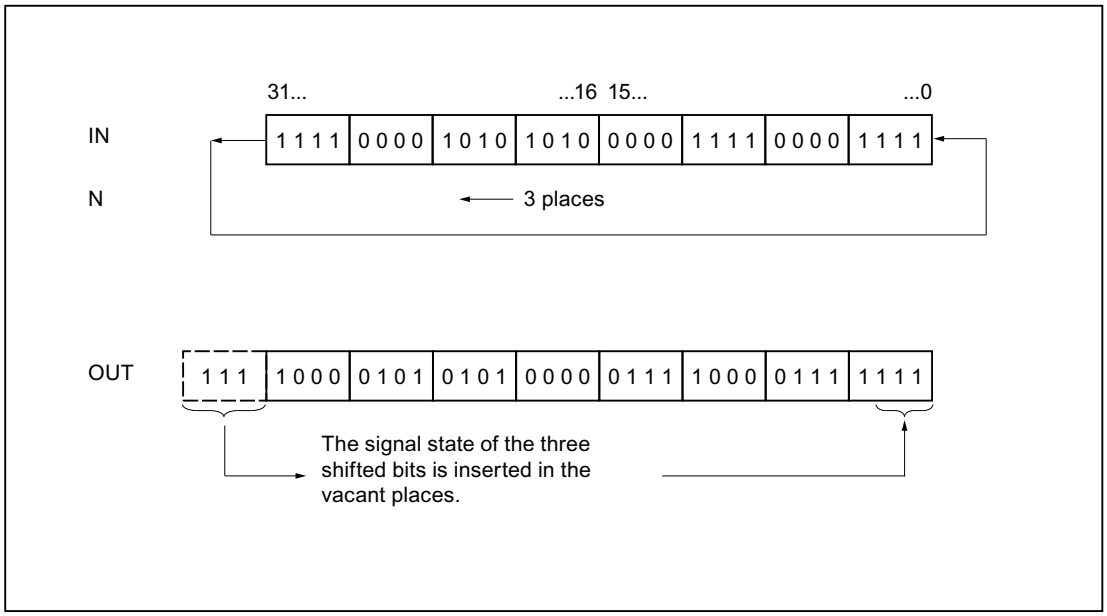
Description

You can use the "Rotate left" instruction to rotate the content of the operand at the input IN bit-by-bit to the left and query the result at the OUT output. The parameter N is used to specify the number of bit positions by which the specified value should be rotated. The bit positions freed by rotating are filled with the bit positions that are pushed out.

If the value at the parameter N is "0", the value at input IN is copied to the operand at output OUT.

If the value at the N parameter is greater than the number of available bit positions, the operand value at IN input is nevertheless rotated by the specified number of bit positions.

The following figure shows how the content of an operand of the DWORD data type is rotated three bit positions to the left:



Parameters

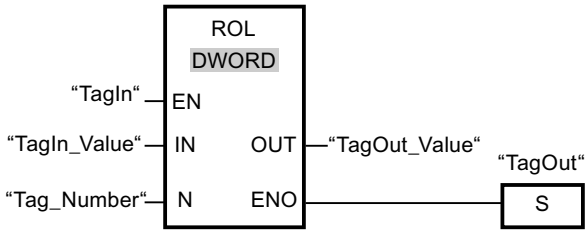
The following table shows the parameters of the instruction "Rotate left":

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, T, C, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	BYTE, WORD, DWORD, DINT	I, Q, M, D, L, P or constant	Value to be rotated.
N	Input	BYTE, WORD, INT	I, Q, M, D, L, P or constant	Number of bit positions by which the value is rotated.
OUT	Output	BYTE, WORD, DWORD, DINT	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	TagIn_Value	1111 0000 1010 1010 0000 1111 0000 1111
N	Tag_Number	5
OUT	TagOut_Value	0001 0101 0100 0001 1110 0001 1111 1110

If input "TagIn" has the signal state "1", the instruction "Rotate left" is executed. The content of the operand "TagIn_Value" is rotated five bit positions to the left. The result is sent at the output "TagOut_Value". If the instruction is executed without errors, the ENO enable output has the signal state "1" and the "TagOut" output is set.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Selecting the data type of an FBD element (Page 2624)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Setting the status bits for shift and rotate instructions (Page 2462)

Basic information on the status word (Page 2453)

Additional instructions

---| |---: Get status bit

Description

With the "Get status bit" instruction you can query the signal state of the following status bits:

- BR: Binary result
- OV: Overflow
- OS: Latching overflow
- CC 0 and CC 1: Display bits

You determine the bits to be queried by selecting the appropriate symbols from the drop-down list of the instruction. The following table shows the options for querying the status bits:

Symbol	Query	Explanation
==0	CC 0 = 0 and CC 1 = 0	The instruction queries whether the result of a mathematical function is equal to 0.
<>0	CC 0 = 0 and CC 1 = 1 or CC 0 = 0 and CC 1 = 0	The instruction queries whether the result of a mathematical function is not equal to 0.
>=0	CC 0 = 0	The instruction queries whether the result of a mathematical function is greater than or equal to 0.
<=0	CC 1 = 0	The instruction queries whether the result of a mathematical function is less than or equal to 0.
>0	CC 0 = 0 and CC 1 = 1	The instruction queries whether the result of a mathematical function is greater than 0.
<0	CC 0 = 1 and CC 1 = 0	The instruction queries whether the result of a mathematical function is less than 0.
OV	CC 0 = 1 and CC 1 = 1	The instruction queries whether the result of a mathematical function is invalid. An invalid result may be caused by an invalid floating-point number or a division by zero.
OV	OV = 1	The instruction queries whether a number range overflow has occurred in an instruction.

Symbol	Query	Explanation
OS	OS = 1	The instruction queries whether a saving number range overflow has occurred in an instruction.
BR	BR = 1	The instruction queries the signal state of the BR status bits.

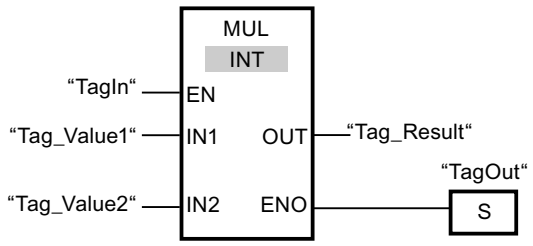
If the query is met, the instruction returns signal state "1". If the query is not met, the instruction returns signal state "0".

You can program a query for status bits "0" using the "Invert RLO" instruction.

Example

The following example shows how the instruction works:

Network 1



Network 2

If the operand "TagIn" has the signal state "1", the instruction "Multiply" (MUL) is executed. If an overflow occurs during the execution of the "Multiply" instruction, the OV status bit is set. After the query of the OV status bit, the "Set output" (S) instruction is executed in network 2 and the "TagOV" operand is set.

See also

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Basic information on the status word (Page 2453)

Setting status bits (Page 2455)

CALL: Call block

Description

The "Call block" instruction calls a function (FC) without parameter. The call is conditional or unconditional, depending on the preceding logic operation.

In the code section of a function (FC), you cannot specify any parameter of the data type BLOCK_FC as the operand for a conditional call. You can, however, specify a parameter of the type BLOCK_FC as the operand in a function block (FB).

A conditional call is then only executed if the result of logic operation (RLO) is "1". If a conditional call is not executed the RLO is "0" after the call instruction. If the instruction is executed, it works as follows:

- It stores the return address of the calling block.
- It stores the two data block registers (data block and instance data block).
- It updates the local data for the called function.
- It shifts the MA bit (active MCR bit) to the block stack (B stack).

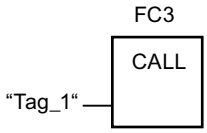
Parameters

The following table shows the parameters of the "Call block" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Block>	Output	BLOCK_FC	-	Called function

Example

The following example shows how the instruction works:



The function FC3 is called if the operand "Tag_1" has the signal state "1".

See also

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Master control relay (Page 2471)

SAVE: Save RLO in BR bit

Description

Use the instruction "Save RLO in BR bit" to save the result of logic operation (RLO) in the binary result (BR). During processing, the instruction transfers the signal state of the current result of logic operation to the status bit BR. The instruction operates independent of the conditions and does not affect any other status bits.

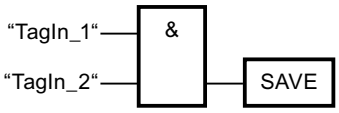
Note

After application of the instruction "Save RLO in BR bit", the BR bit can be modified again by the following instructions in the same or lower-level blocks.

Use the instruction "Save RLO in BR bit" to check the execution status of a block. If you program the "Save RLO in BR bit" instruction at the end of a block, for example, the BR bit is set to the signal state of the current result of logic operation of the block. If the block is executed without errors, the BR bit and the ENO enable output in the calling block are set to signal state "1". Signal state "0" is assigned to the BR bit if an error occurs during processing of the block. You can program a troubleshooting routine by querying the BR bit.

Example

The following example shows how the instruction works:



The BR is set to signal state "1" when the "TagIn_1" and "TagIn_2" operands return signal state "1".

See also

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Basic information on the status word (Page 2453)

Example of the use of the EN/ENO mechanism in FBD (Page 2450)

MCR<: Open MCR ranges

Description

The "Open MCR ranges" instruction saves the result of logic operation (RLO) in the MCR stack and enables MCR dependency within an MCR range (MCR = Master Control Relay).

The MCR stack can contain up to a maximum of 8 entries and functions like a LIFO buffer (Last In, First Out). When the stack is full, the instruction generates an error message (MCRF).

Note

Read the information on using MCR functions in the "Important notes on using MCR functionality" at "See also".

Use the instructions "Enable MCR range" (MCRA) and "Disable MCR range" (MCRD) to specify the start and end of an MCR range. Within the defined MCR range you can enable and disable MCR dependency for one or more program sections.

MCR dependency is enabled when the RLO is "1" before processing of the instruction. If the RLO is "0" before the instruction, MCR dependency is disabled. The end of the MCR dependent program section is programmed using the instruction "Close MCR ranges". The program section between the two instructions is affected by the MCR. Only the following MCR-dependent instructions are affected by this:

- Assignment
- Set output (S)
- Reset output (R)
- Reset/set flip-flop (RS)
- Set/reset flip-flop (SR)
- Move value (MOVE)

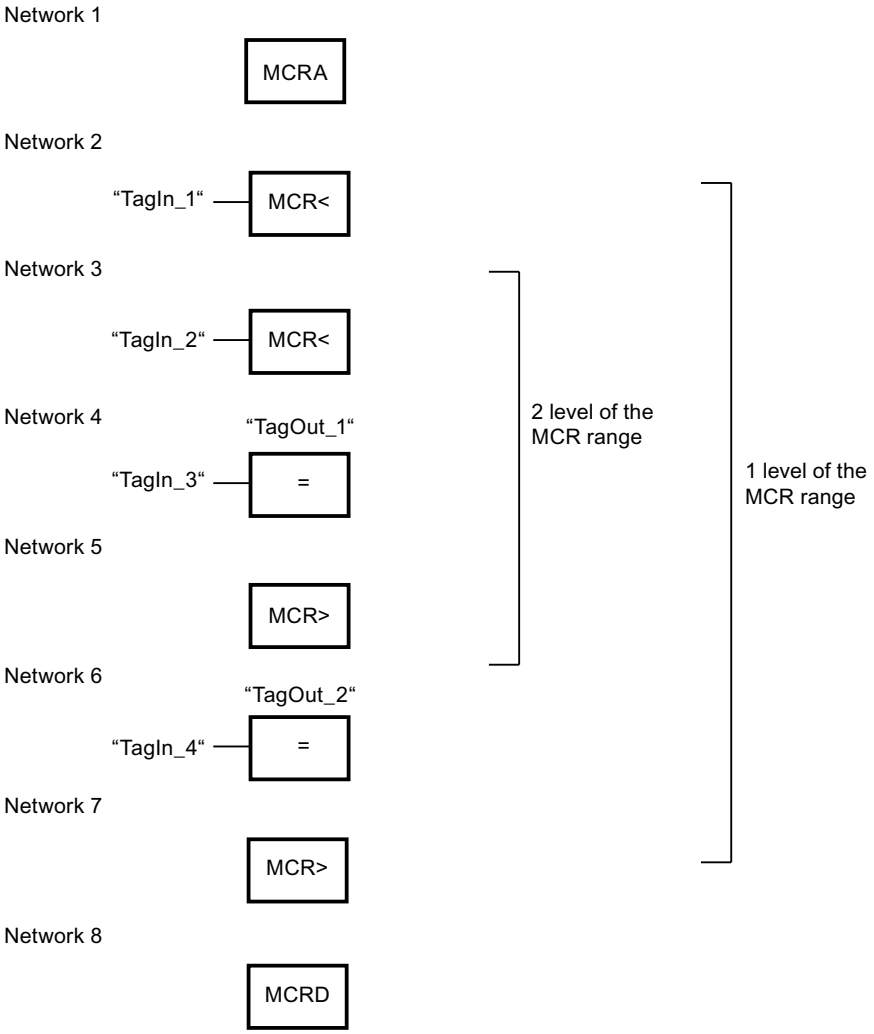
You can also nest the enabling and disabling of MCR dependency by programming the instruction "Open MCR ranges" in an MCR dependent program section. You can nest the instruction up to 8 levels deep. The MCR dependency of the higher-level program section hereby controls the MCR dependencies in all lower-level program sections. If MCR dependency is disabled in the higher-level program section, this MCR dependency cannot be switched on in any of the lower-level MCR dependent program sections.

If you call a block within an MCR range, MCR dependency is disabled in the called block. When the block is exited, MCR dependency is set so that it is the same as before the block call.

The instructions "Open MCR ranges" and "Close MCR ranges" must always be programmed in pairs. If one of the two instructions is not programmed, an MCR stack error is triggered.

Example

The following example shows how the instruction works:



The start of the MCR range is programmed in network 1. If the "TagIn_1" operand returns signal state "1", MCR dependency is enabled in the first level of the MCR range. In this case, the signal state of the "TagIn_4" operand is assigned to the "TagOut_2" operand (network 6). If the "TagIn_1" operand returns signal state "0", MCR dependency is disabled and the "TagOut_2" operand is reset to "0". MCR dependency of the MCR range is disabled in network 7.

In network 3, MCR dependency is enabled in the second level of the MCR range, when the "TagIn_2" operand has signal state "1" and the first level of the MCR range is enabled. In this case, the signal state of the "TagIn_3" operand is assigned to the "TagOut_1" operand. If the "TagIn_2" operand returns signal state "0", MCR dependency is disabled and the "TagOut_1" operand is reset to "0". MCR dependency in the second level of the MCR range is disabled in network 5. The end of the MCR range is programmed in network 8.

See also

Master control relay (Page 2471)

Important notes on using MCR functions (Page 2472)

MCR>: Close MCR ranges (Page 5742)

MCRA: Enable MCR range (Page 5744)

MCRD: Disable MCR range (Page 5745)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

MCR>: Close MCR ranges

Description

The instruction "Close MCR ranges" closes an MCR dependent program section (MCR = Master Control Relay). In this case the instruction deletes the result of logic operation (RLO) from the last entry in the MCR stack.

The MCR stack can contain up to a maximum of 8 entries and functions like a LIFO buffer (Last In, First Out). If the stack is already empty, the instruction generates an error message (MCRF).

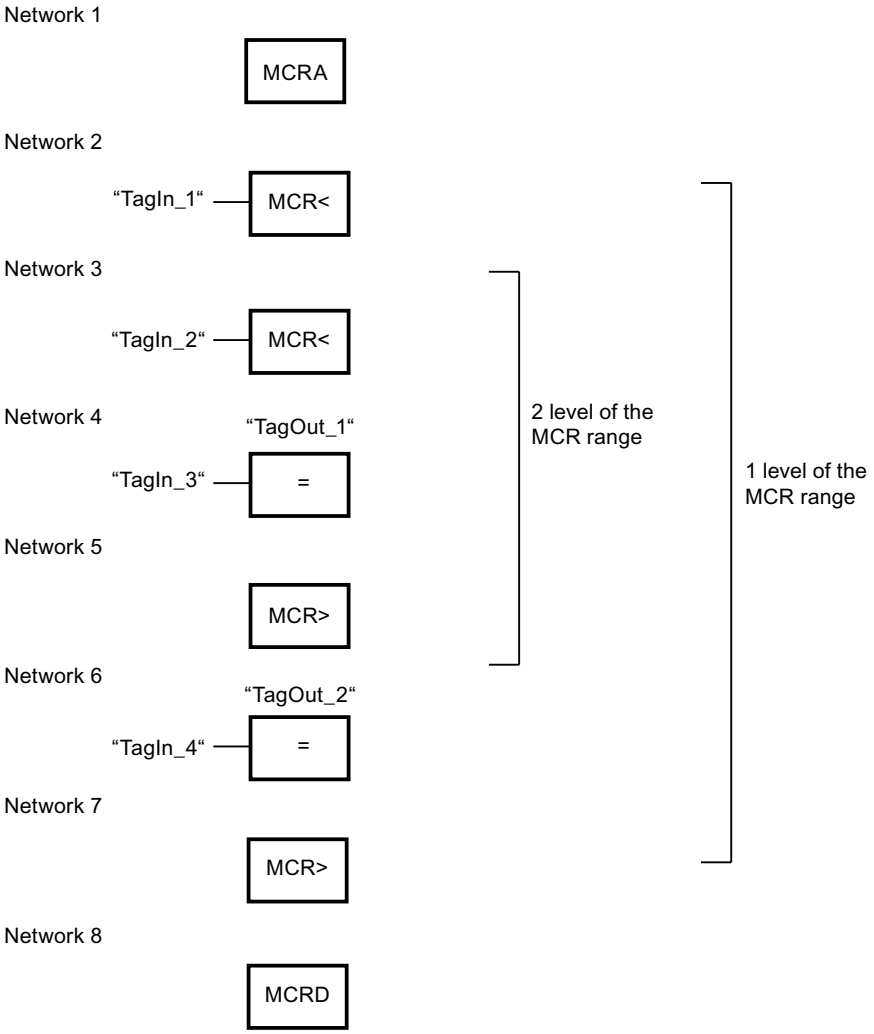
Note

Read the information on using MCR functions in the "Important notes on using MCR functionality" at "See also".

Use the instruction "Open MCR ranges" to program MCR dependency. The instructions "Open MCR ranges" and "Close MCR ranges" must always be programmed in pairs. If one of the two instructions is not programmed, an MCR stack error is triggered.

Example

The following example shows how the instruction works:



The start of the MCR range is programmed in network 1. If the "TagIn_1" operand returns signal state "1", MCR dependency is enabled in the first level of the MCR range. In this case, the signal state of the "TagIn_4" operand is assigned to the "TagOut_2" operand (network 6). If the "TagIn_1" operand returns signal state "0", MCR dependency is disabled and the "TagOut_2" operand is reset to "0". MCR dependency of the MCR range is disabled in network 7.

In network 3, the MCR-dependency is enabled in the second level of the MCR range, when the "TagIn_2" operand has signal state "1". In this case, the signal state of the "TagIn_3" operand is assigned to the "TagOut_1" operand. If the "TagIn_2" operand returns signal state "0", the MCR dependency is disabled and the "TagOut_1" operand is reset to "0" regardless of the signal state of the "TagIn_3" operand. MCR dependency in the second level of the MCR range is disabled in network 5. The end of the MCR range is programmed in network 8.

See also

Master control relay (Page 2471)

Important notes on using MCR functions (Page 2472)

MCR<: Open MCR ranges (Page 5739)

MCRA: Enable MCR range (Page 5744)

MCRD: Disable MCR range (Page 5745)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

MCRA: Enable MCR range

Description

Use the instruction "Enable MCR range" to specify the start of an MCR range (MCR = Master Control Relay). Within this range, you can switch MCR dependency on and off using the following instructions:

- MCR<: Open MCR ranges
- MCR>: Close MCR ranges

You specify the end of the MCR range with the instruction "Disable MCR range" (MCRD). The instructions for defining the start and the end of an MCR range must always be used in pairs.

Note

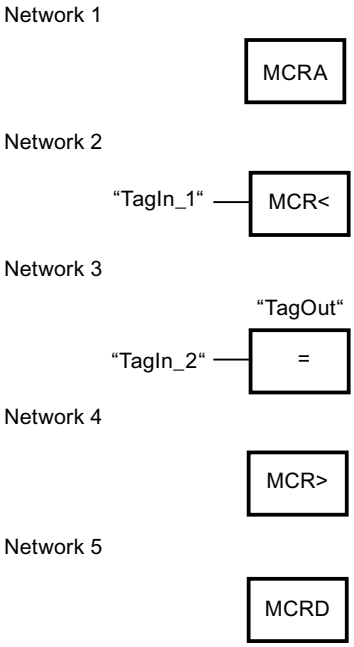
Read the information on using MCR functions in the "Important notes on using MCR functionality" at "See also".

If you call a block within an MCR range, MCR dependency is disabled in the called block. When a block is exited, the MCR dependency is set so that it is the same as before the block call.

The instruction is executed regardless of the status bits and without affecting these bits.

Example

The following example shows how the instruction works:



The start of the MCR range is programmed in network 1. If the "TagIn_1" operand returns the signal state "1", the MCR dependency is enabled. In this case, the signal state of the "TagIn_2" operand is assigned to the "TagOut" operand (network 3). If the "TagIn_1" operand returns signal state "0", the MCR dependency is disabled and the "TagOut" operand is reset to "0" regardless of the signal state of the "TagIn_2" operand. MCR dependency of the MCR range is disabled in network 4. The end of the MCR range is programmed in network 8.

See also

Master control relay (Page 2471)

Important notes on using MCR functions (Page 2472)

MCR<: Open MCR ranges (Page 5739)

MCR>: Close MCR ranges (Page 5742)

MCRD: Disable MCR range (Page 5745)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

MCRD: Disable MCR range

Description

Use the instruction "Disable MCR range" to specify the start of an MCR range (MCR = Master Control Relay).

You specify the start of the MCR range with the instruction "Enable MCR range" (MCRA). The instructions for defining the start and the end of an MCR range must always be used in pairs.

Note

Read the information on using MCR functions in the "Important notes on using MCR functionality" at "See also".

Within an MCR range you can program the MCR dependency with the help of the following instructions:

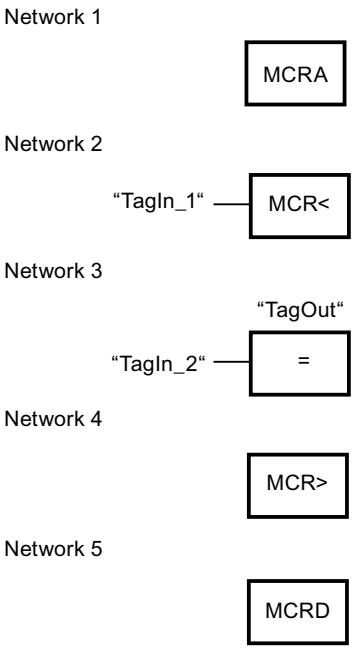
- MCR<: Open MCR ranges
- MCR>: Close MCR ranges

If you call a block within an MCR range, MCR dependency is disabled in the called block. When a block is exited, the MCR dependency is set so that it is the same as before the block call.

The instruction is executed regardless of the status bits and without affecting these bits.

Example

The following example shows how the instruction works:



The start of the MCR range is programmed in network 1. If the "TagIn_1" operand returns the signal state "1", the MCR dependency is enabled. In this case, the signal state of the "TagIn_2" operand is assigned to the "TagOut" operand (network 3). If the "TagIn_1" operand returns signal state "0", MCR dependency is disabled and the "TagOut" operand is reset to "0". MCR dependency of the MCR range is disabled in network 4. The end of the MCR range is programmed in network 8.

See also

Master control relay (Page 2471)

Important notes on using MCR functions (Page 2472)

MCR<: Open MCR ranges (Page 5739)

MCR>: Close MCR ranges (Page 5742)

MCRA: Enable MCR range (Page 5744)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

SET: Set bit array

Description

The "Set bit array" instruction sets the signal state of the bit in a specified area to "1". You use the pointer on the S_BIT parameter to define the starting point of the range. If the pointer points to the memory area of the external I/O, the instruction is not executed. At the N parameter, you specify the number of bits to be set in the specified area. When the N parameter has the value "0", calling the instruction has no effect.

The instruction is only executed when the MCR bit is "1". When the MCR bit has the signal state "0", the bits in the specified range are not changed.

Parameters

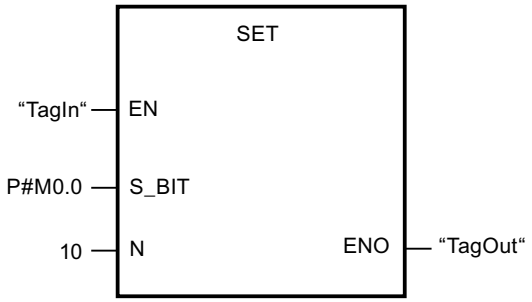
The following table shows the parameters of the "Set bit array" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
S_BIT	Input	POINTER	I, Q, M, D	Pointer to the first bit of the range whose bits are set For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	I, Q, M, D, L, P or constant	Number of bits to be set

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:



If the "TagIn" operand has the signal state "1", the instruction sets 10 bits in the area from M0.0 to M1.2.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Master control relay (Page 2471)

SETP: Set a bit array in the I/O area

Description

The instruction "Set a bit array in the I/O area" sets the bits of an I/O area to the signal state "1". At the same time the corresponding bits in the process image of the outputs are also set parallel to the I/O bits.

You use the N and SA parameters to define the bit array that will be set. The bits of the array must be located in the part of the I/O area to which a process image is assigned.

If no I/O devices are plugged in for part of the selected bit array, the instruction attempts to set the entire bit array. In this case an error information is output on the RET_VAL parameter.

The I/O area is set byte-by-byte. If the bit array defined via the N and SA parameters does not begin or end at a byte boundary, calling the instruction has the following effect:

- The bits that are in the first and last bytes to be transferred and that do not belong to the defined bit array, get the value of the corresponding bits in the process image output table.

  WARNING

  The setting of the bits at the value of the relevant bit in the process image output table can lead, for example, to lead to unintended reactions such as starting a motor or turning off a cooling system.

- The bits belonging to the defined bit array are set to the signal state "1".

The instruction "Set a bit array in the I/O area" must be called in an MCR area.

Calling the "Set a bit array in the I/O area" instruction has no effect and the ENO enable output is set to "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The N parameter has the value "0".
- The MCR bit has the signal state "1".

Parameters

The following table shows the parameters of the "Set a bit array in the I/O area" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
N	Input	INT	I, Q, M, D, L or constant	Number of bits to be set
RET_VAL	Output	INT	I, Q, M, D, L	Error information
SA	Output	POINTER	P	Pointer to the first bit to be set

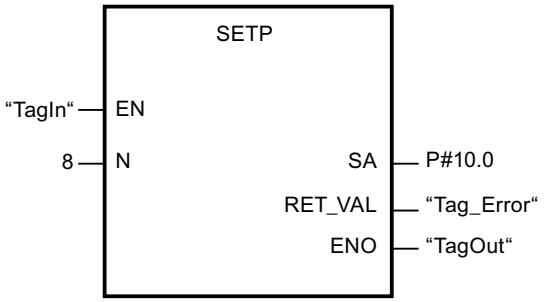
Parameters RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:



If the "TagIn" operand returns the signal state "1", the instruction sets 8 bits in the I/O area based on outputs Q10.0 to Q10.7.

See also

Evaluating errors with output parameter RET_VAL (Page 5370)

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Switching display formats in the program status (Page 2980)

SETI: Set byte array

Description

The "Set byte array" instruction sets the signal state of the bits in a specified area to "1". You define the start point of the array on the S_BYTE. You define the size of the array by specifying the number of bits to be set on the N parameter of the instruction.

Note

The value of the N parameter must be a multiple of the number 8 (for example, 8, 16, 24, etc.).

The pointer specified on the S_BYTE parameter must reference the external I/O memory area (P). Since the "P" memory area is accessed as bytes, words, or double words, the pointer on

the S_BYTE parameter must reference an operand that corresponds to a byte limit value, in other words the bit number of the pointer must be "0".

Note

The signal state of the corresponding bits in the process-image output table (Q) is reset to "0".

Calling the "Set byte array" instruction has no effect and the ENO enable output is set to "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The pointer on the S_BYTE parameter does not reference the external I/O memory area (P).
- The pointer on the S_BYTE parameter references an operand that is not byte-aligned.
- The value of the N parameter is not a multiple of the number 8.
- The MCR bit has the signal state "0".

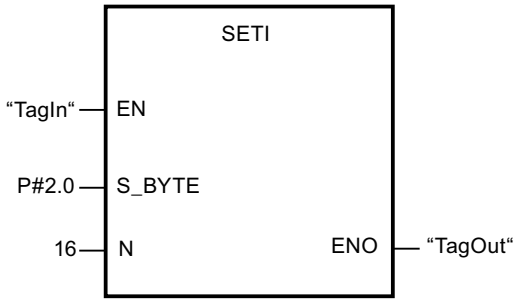
Parameters

The following table shows the parameters of the "Set byte array" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
S_BYTE	Input	POINTER	P or constant	Pointer to the first byte of the array whose bits are to be set For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	I, Q, M, D, L, P or constant	Size of the array to be set The array is specified by the number of bits as a multiple of 8 (for example 8, 16, 24 etc.).

Example

The following example shows how the instruction works:



If the "TagIn" operand returns the signal state "1", the instruction sets 2 bytes (16 bits) in the range from P2.0 to P3.7

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Master control relay (Page 2471)

RESET: Reset bit array

Description

The "Reset bit array" instruction sets the signal state of the bit in a specified area to "0". You use the pointer on the S_BIT parameter to define the starting point of the range. If the pointer points to the memory area of the external I/O, the instruction is not executed. At the N parameter, you specify the number of bits to be reset in the specified range. When the N parameter has the value "0", calling the instruction has no effect.

The instruction is only executed when the MCR bit is "1". When the MCR bit has the signal state "0", the bits in the specified range are not changed.

Parameters

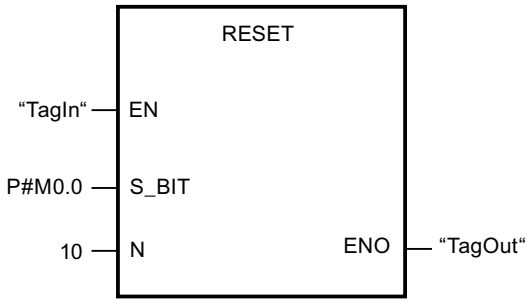
The following table shows the parameters of the "Reset bit array" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

Parameters	Declaration	Data type	Memory area	Description
S_BIT	Input	POINTER	I, Q, M, D	Pointer to the first bit of the range whose bits are to be reset For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	I, Q, M, D, L, P or constant	Number of bits to be reset

Example

The following example shows how the instruction works:



If the "TagIn" operand has the signal state "1", the instruction resets 10 bits in the area from M0.0 to M1.2.

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Master control relay (Page 2471)

RESETP: Reset a bit array in the I/O area

Description

The instruction "Reset a bit array in the I/O area" resets the bits of an I/O area to the signal state "0". At the same time the corresponding bits in the process image of the outputs are also reset parallel to the I/O bits.

You use the N and SA parameters to define the bit array that will be reset. The bits of the array must be located in the part of the I/O area to which a process image is assigned.

If no I/O devices are plugged in for part of the selected bit array, the instruction attempts to set the entire bit array. In this case an error information is output on the RET_VAL parameter.

The I/O area is reset byte-by-byte. If the bit array defined via the N and SA parameters does not begin or end at a byte boundary, calling the instruction has the following effect:

- The bits that are in the first and last bytes to be transferred and that do not belong to the defined bit array, get the value of the corresponding bits in the process image output table.

  WARNING

  The setting of the bits at the value of the relevant bit in the process image output table can lead, for example, to lead to unintended reactions such as starting a motor or turning off a cooling system.

- The bits belonging to the defined bit array are reset to the signal state "0".

The instruction "Reset a bit array in the I/O area" must be called in an MCR range.

Calling the instruction "Reset a bit array in the I/O area" has no effect and enable output ENO is set to "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The N parameter has the value "0".
- The MCR bit has the signal state "1".

Parameters

The following table shows the parameters of the instruction "Reset a bit array in the I/O area":

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
N	Input	INT	I, Q, M, D, L or constant	Number of bits to be reset.
RET_VAL	Output	INT	I, Q, M, D, L	Error information
SA	Output	POINTER	P	Pointer to the first bit to be reset.

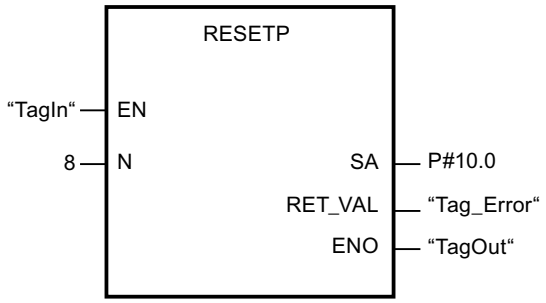
Parameters RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:



If the "TagIn" operand returns the signal state "1", the instruction sets 8 bits in the I/O area based on outputs Q10.0 to Q10.7.

See also

Evaluating errors with output parameter RET_VAL (Page 5370)

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Switching display formats in the program status (Page 2980)

RESETI: Reset byte array

Description

The "Reset byte array" instruction sets the signal state of the bits in a specified area to "0". You use the pointer on the S_BYTE parameter to define the starting point of the range. You

define the size of the range by specifying the number of bits to be reset on the N parameter of the instruction.

Note

The value of the N parameter must be a multiple of the number 8 (for example, 8, 16, 24, etc.).

The pointer specified on the S_BYTE parameter must reference the external I/O memory area (P). Since the "P" memory area is accessed as bytes, words, or double words, the pointer on the S_BYTE parameter must reference an operand that corresponds to a byte limit value, in other words the bit number of the pointer must be "0".

Note

The signal state of the corresponding bits in the process-image output table (Q) is reset to "0".

Calling the "Reset byte array" instruction has no effect and the ENO enable output is set to "0" if one of the following conditions applies:

- Enable input EN has the signal state "0".
- The pointer on the S_BYTE parameter does not reference the external I/O memory area (P).
- The pointer on the S_BYTE parameter references an operand that is not byte-aligned.
- The value of the N parameter is not a multiple of the number 8.
- The MCR bit has the signal state "0".

Parameters

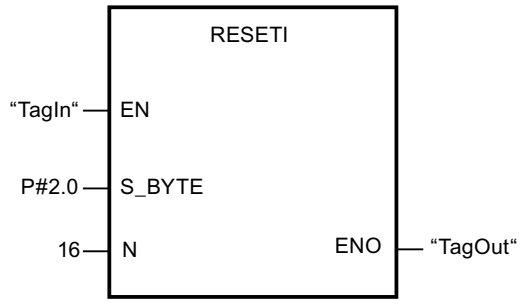
The following table shows the parameters of the "Reset byte array" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output

Parameters	Declaration	Data type	Memory area	Description
S_BYTE	Input	POINTER	P or constant	Pointer to the first byte of the range whose bits are to be reset For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	I, Q, M, D, L, P or constant	Size of the array whose bits are to be reset. The array is specified by the number of bits as a multiple of 8 (for example 8, 16, 24 etc.).

Example

The following example shows how the instruction works:



If the "TagIn" operand returns the signal state "1", the instruction sets 2 bytes (16 bits) in the range from P2.0 to P3.7

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Master control relay (Page 2471)

REPL_VAL: Enter substitute value

Description

The "Enter substitute value" instruction is used to enter a substitute value in the accumulator 1 (ACCU 1) from a synchronous error organization block.

Note

You can only call the "Enter substitute value" instruction in a synchronous error OB (OB 121, OB 122).

You use the "Enter substitute value" instruction when you can no longer read any value from a module. The OB 122 is started after each access to this module. By calling the "Enter substitute value" instruction you can cause the substitute value to be loaded in accumulator 1. In this case the program will then be processed with this substitute value. You can take the required information for the selection of the substitute value from the local tags of the OB 122.

Parameters

The following table shows the parameters of the "Enter substitute value" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
VAL	Input	DWORD	I, Q, M, D, L or constant	Substitute value
RET_VAL	Output	INT	I, Q, M, D, L	Error information

Parameters RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Description
0000	No error
8080	The instruction was not called from a synchronous error OB.
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Evaluating errors with output parameter RET_VAL (Page 5370)

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Switching display formats in the program status (Page 2980)

DRUM: Implement sequencer

Description

You can use the "Implement sequencer" instruction to assign the programmed values of the OUT_VAL parameter of the corresponding step to the programmed output bits (OUT1 to OUT16) and the output word (OUT_WORD). The specific step must thereby satisfy the conditions of the programmed enable mask on the S_MASK parameter while the instruction remains at this step. The instruction advances to the next step if the event for the step is true and the programmed time for the current step elapses, or if the value at the JOG parameter changes from "0" to "1". The instruction is reset if the signal state on the RESET parameter changes to "1". The current step is hereby equated to the preset step (DSP).

The amount of time spent on a step is determined by the product of the preset timebase (DTBP) and the preset counter value (S_PRESET) for each step. At the start of a new step, this calculated value is loaded into the DCC parameter, which contains the time remaining for the current step. If, for example the value at the DTBP parameter is "2" and the preset value for the first step is "100" (100 ms), the DCC parameter has the value "200" (200 ms).

A step can be programmed with a timer value, an event, or both. Steps that have an event bit and the timer value "0" advance to the next step as soon as the signal state of the event bit is "1". Steps that are programmed only with a timer value start the time immediately. Steps that are programmed with an event bit and a timer value greater than "0" start the time when the signal state of the event bit is "1". The event bits are initialized with a signal state of "1".

When the sequencer is on the last programmed step (LST_STEP) and the time for this step has expired, the signal state on the Q parameter is set to "1"; otherwise it is set to "0". When the parameter Q is set, the instruction remains on the step until it is reset.

In the configurable mask (S_MASK) you can select the separate bits in the output word (OUT_WORD) and set or reset the output bits (OUT1 to OUT16) by means of the output values (OUT_VAL). If a bit of the configurable mask is in the signal state "1", the value OUT_VAL sets or resets the corresponding bit. If the signal state of a bit of the configurable mask is "0", the corresponding bit is left unchanged. All the bits of the configurable mask for all 16 steps are initialized with a signal state of "1".

The output bit on the OUT1 parameter corresponds to the least significant bit of the output word (OUT_WORD). The output bit on the OUT16 parameter corresponds to the most significant bit of the output word (OUT_WORD).

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the "Implement sequencer" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
RESET	Input	BOOL	I, Q, M, D, L	A signal state of "1" indicates a reset condition.
JOG	Input	BOOL	I, Q, M, D, L	When the signal state changes from "0" to "1", the instruction advances to the next step.
DRUM_EN	Input	BOOL	I, Q, M, D, L	A signal state of "1" allows the sequencer to advance based on the event and time criteria.
LST_STEP	Input	BYTE	I, Q, M, D, L or constant	Number of the last programmed step
EVENT(i), 1 ≤ i ≤ 16	Input	BOOL	I, Q, M, D, L	Event bit (i); Initial signal state is "1".
OUT(j), 1 ≤ j ≤ 16	Output	BOOL	I, Q, M, D, L	Output bit (j)
Q	Output	BOOL	I, Q, M, D, L	A signal state of "1" indicates that the time for the last step has elapsed.
OUT_WORD	Output	WORD	I, Q, M, D, L, P	Word address to which the sequencer writes the output values.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
JOG_HIS	Static	BOOL	I, Q, M, D, L	JOG parameter history bit

Parameter	Declaration	Data type	Memory area	Description
EOD	Static	BOOL	I, Q, M, D, L	A signal state of "1" indicates that the time for the last step has elapsed.
DSP	Static	BYTE	I, Q, M, D, L, P	Preset step of the sequencer
DSC	Static	BYTE	I, Q, M, D, L, P	Current step of the sequencer
DCC	Static	DWORD	I, Q, M, D, L, P	Current numerical value of the sequencer
DTBP	Static	WORD	I, Q, M, D, L, P	Preset timebase of the sequencer
PREV_TIME	Static	DWORD	I, Q, M, D, L or constant	Previous system time
S_PRESET	Static	ARRAY of WORD	I, Q, M, D, L	Count preset for each step [1 to 16]; whereby 1 count = 1 ms.
OUT_VAL	Static	ARRAY of BOOL	I, Q, M, D, L	Output values for each step [1 to 16, 0 to 15].
S_MASK	Static	ARRAY of BOOL	I, Q, M, D, L	Configurable mask for each step [1 to 16, 0 to 15]. Initial signal states are "1".

For additional information on valid data types, refer to "See also".

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

ERR_CODE*	Explanation
W#16#0000	No error
W#16#000B	The value at the LST_STEP parameter is less than 1 or greater than 16.
W#16#000C	The value at the DSC parameter is less than 1 or greater than the value at the LST_STEP parameter.
W#16#000D	The value at the DSP parameter is less than 1 or greater than the value at the LST_STEP parameter.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Switching display formats in the program status (Page 2980)

DRUM_X: Implement sequencer

Description

The "Implement sequencer" instruction is used to implement a sequencer with maximum 16 steps. You specify the number of the first step on the DSP parameter. You specify the number of the last step on the LST_STEP parameter.

During each step, all 16 output bits OUT0 to OUT15 and the output parameter OUT_WORD (in which the output bits are collected together) are written. An output bit is assigned either the corresponding bit of the OUT_VAL array that you specified or the output bit of the preceding step. The value assigned depends on the value of the mask bit on the S_MASK parameter.

Note

The default setting for the mask bit is 0. If you want to change the current setting of one or more mask bits, make this change in the instance data block.

The "Implement sequencer" instruction advances to the next step when there is a rising edge on the JOG parameter. If the instruction is already in the last step, the tags Q and EOD are set when the JOG parameter has a rising edge. In this case the DCC parameter is set to the value "0". The instruction remains in the last step until the RESET parameter is set to the value "1".

You can also assign parameters so that switching to the next step is time dependent. To do this, the DRUM_EN parameter has to be set to the value "1". The transition to the next step is made when the event bit (EVENTi) belonging to the current step is set and the time specified for the current step has expired. The time for a step is the product of the (DTBP) time base and the time factor valid for the current step (S_PRESET).

Note

The execution time (DCC) remaining in the current step is reduced only when the corresponding event bit (EVENTi) is set.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

If the value "1" is set on the RESET parameter when the instruction is called, the sequencer goes to the step whose number you assigned on the DSP parameter.

A step can be programmed with a timer value, an event, or both. Steps that have an event bit and the timer value "0" advance to the next step as soon as the signal state of the event bit is "1". Steps that are programmed only with a timer value start the time immediately. Steps that are programmed with an event bit and a timer value greater than "0" start the time when the signal state of the event bit is "1".

You can also move the next step at any time via the value at the JOG parameter.

When the instruction is called for the first time, you must set the value "1" on the RESET parameter.

When the sequencer is in the last step and the execution time specified for this step has expired, the parameters Q and EOD are set. The instruction remains in the last step until the RESET parameter changes to the value "1".

The instruction is executed only in the STARTUP and RUN modes.

The operating system resets the "Implement sequencer" instruction during a cold restart. During a warm restart, the instruction is not reset. If you want the "Implement sequencer" instruction to be initialized after a warm restart, you have to call the OB 100 with the value "1" on the RESET parameter.

Parameters

The following table shows the parameters of the "Implement sequencer" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
RESET	Input	BOOL	I, Q, M, D, L	The signal state "1" resets the sequencer. When the instruction is called for the first time, you have to set the value "1" on the RESET parameter. Constants can also be specified on the RESET parameter.
JOG	Input	BOOL	I, Q, M, D, L	A rising edge (compared to the last call) switches the sequencer to the next step if it is not yet in the last step. The next step is enabled depending on the value you assigned to the DRUM_EN parameter.
DRUM_EN	Input	BOOL	I, Q, M, D, L	Control parameter that specifies whether time-dependent switching to the next step is possible. If the value is "1" a time-dependent switching to the next step is possible.

Parameter	Declaration	Data type	Memory area	Description
LST_STEP	Input	BYTE	I, Q, M, D, L or constant	Number of the last step Values from 1 to 16 are permitted.
EVENTi, 1 ≤ i ≤ 16	Input	BOOL	I, Q, M, D, L	Event bit number i (belongs to step i)
OUTj, 0 ≤ j ≤ 15	Output	BOOL	I, Q, M, D, L	Output bit j
Q	Output	BOOL	I, Q, M, D, L	Status parameter that indicates whether the specified execution time of the last step has expired.
OUT_WORD	Output	WORD	I, Q, M, D, L, P	Output bits collected together in a tag
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
JOG_HIS	Static	BOOL	I, Q, M, D, L	Not relevant to user (input parameter JOG of the preceding call)
EOD	Static	BOOL	I, Q, M, D, L	Identical to the output parameter Q
DSP	Static	BYTE	I, Q, M, D, L, P or constant	Number of the first step Values from 1 to 16 are permitted.
DSC	Static	BYTE	I, Q, M, D, L, P or constant	Number of the current step
DCC	Static	DWORD	I, Q, M, D, L, P	The execution time still remaining in the current step in milliseconds (ms) (only relevant if DRUM_EN = 1 and the corresponding event bit = 1)
DTBP	Static	WORD	I, Q, M, D, L, P or constant	The time base valid for all steps in milliseconds (ms)
PREV_TIME	Static	DWORD	I, Q, M, D, L or constant	(Not relevant for the user: system time of the previous call)
S_PRESET	Static	ARRAY of WORD	I, Q, M, D, L or constant	One-dimensional array with the time factor for each step. Recommended selection of the indices: [1 to 16]. In this case, S_PRESET [x] has the time factor of step x.

Parameter	Declaration	Data type	Memory area	Description
OUT_VAL	Static	ARRAY of BOOL	I, Q, M, D, L	Two-dimensional array with the values to be output in each step if they have not been masked out using S_MASK. Recommended selection of the indices: [1 to 16, 0 to 15]. In this case, OUT_VAL [x, y] has the value assigned to the output bit OUTy in step x.
S_MASK	Static	ARRAY of BOOL	I, Q, M, D, L	Two-dimensional array with the mask bits for each step. Recommended selection of the indices: [1 to 16, 0 to 15]. In this case, S_MASK [x, y] contains the mask bit for the "y"the value to be output in step x. Meaning of the mask bits: • 0: The corresponding value of the previous step is assigned to the relevant output bit. • 1: The corresponding value from OUT_VAL is assigned to the relevant output bit.

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

ERR_CODE* (W#16#...)	Explanation
0000	No error
8081	Invalid value at parameter LST_STEP.
8082	Invalid value on parameter DSC.
8083	Invalid value on parameter DSP.
8084	The product DCC = DTBP * S_PRESET[DSC] exceeds the value 2**31-1 (approximately 24.86 days).
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Switching display formats in the program status (Page 2980)

DCAT: Discrete control-timer alarm

Description

The instruction "Discrete control-timer alarm" is used to accumulate the time from the point at which the parameter CMD issues the command to open or close. The time is accumulated until the preset time (PT) is exceeded or the information is received that the device was opened or closed (O_FB or C_FB) within the specified time. If the preset time is exceeded before the information on the opening or closing of the device is received, the corresponding alarm is activated. If the signal state on the command input changes state before the preset time, the time is restarted.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The "Discrete control-timer alarm" instruction has the following reactions to the input conditions:

- When the signal state of the CMD parameter changes from "0" to "1", the signal states of the parameters Q, CMD_HIS, ET, OA and CA are influenced as follows:
  - The parameters Q and CMD_HIS are set to "1".
  - The parameters ET, OA and CA are reset to "0".
- When the signal state on the parameter CMD changes from "1" to "0", the parameters Q, ET, OA, CA and CMD_HIS are reset to "0".
- When the signal state of the CMD and CMD_HIS parameters is "1" and the parameter O_FB is set to "0", the time difference (ms) since the last execution of the instruction is added to the value at the parameter ET. If the value of the parameter ET exceeds the value of the parameter PT, the signal state on the parameter OA is set to "1". If the value of the parameter ET does not exceed the value of the parameter PT, the signal state on the parameter OA is reset to "0". The value at the parameter CMD_HIS is reset to the value of the parameter CMD.
- If the signal state of the parameters CMD, CMD_HIS and O_FB are set to "1" and the parameter C_FB has the value "0", the signal state of the parameter OA is set to "0". The value of the parameter ET is set to the value of the parameter PT. If the signal state of the parameter O_FB changes to "0", the alarm is set the next time the instruction is executed. The value of the parameter CMD_HIS is set to the value of the parameter CMD.
- If the CMD, CMD_HIS and C_FB parameters return signal state "0", the time difference (ms) since the last execution of the instruction is added to the value of the ET parameter. If the value of the parameter ET exceeds the value of the parameter PT, the signal state of the parameter CA is reset to "1". If the value at the parameter PT is not exceeded, the parameter CA has the signal state "0". The value of the parameter CMD_HIS is set to the value of the parameter CMD.

- If the parameters CMD, CMD_HIS and O_FB have the signal state "0" and the parameter C_FB is set to "1", the parameter CA is set to "0". The value of the parameter ET is set to the value of the parameter PT. If the signal state of the parameter C_FB changes to "0", the alarm is set the next time the instruction is executed. The value of the parameter CMD_HIS is set to the value of the parameter CMD.
- If the parameters O_FB and C_FB simultaneously have the signal state "1", the signal states of both alarm outputs are set to "1".

The "Discrete control-timer alarm" instruction has no error information.

Parameters

The following table shows the parameters of the "Discrete control-timer alarm" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
CMD	Input	BOOL	I, Q, M, D, L	A signal state of "0" indicates a "close" command. A signal state of "1" indicates an "open" command.
O_FB	Input	BOOL	I, Q, M, D, L	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L	Feedback input when closing
Q	Output	BOOL	I, Q, M, D, L	Shows the status of the parameter CMD
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
ET	Static	DINT	I, Q, M, D, L	Currently elapsed time, where one count = 1 ms.
PT	Static	DINT	I, Q, M, D, L	Preset timer value, where one count = 1 ms.
PREV_TIME	Static	DWORD	I, Q, M, D, L	Previous system time
CMD_HIS	Static	BOOL	I, Q, M, D, L	CMD history bit

For additional information on valid data types, refer to "See also".

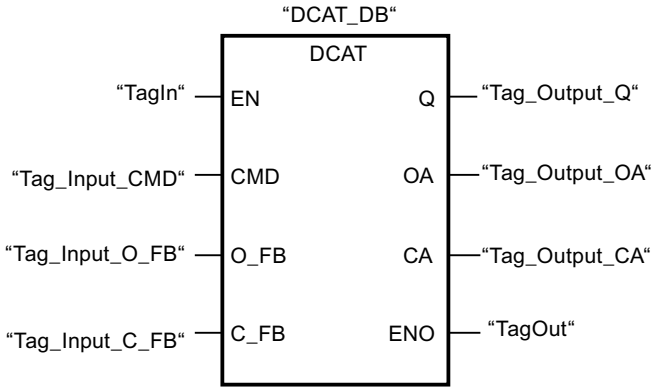
Example

In the following example the parameter CMD changes from "0" to "1". After the execution of the instruction the parameter Q is set to "1" and the two alarm outputs OA and CA have the

signal state "0". The parameter CMD_HIS of the instance data block is set to the signal state "1" and the parameter ET is reset to "0".

Note

You can initialize static parameters in the data block.



The following table shows how the instruction works using specific operand values:

Before processing

In this example the following values are used for the input and output parameters:

Parameter	Operand	Value
CMD	Tag_Input_CMD	TRUE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
Q	Tag_Output_Q	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the instance data block "DCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#12
PT	DBD8	L#222
CMD_HIS	DBX16.0	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
Q	Tag_Output_Q	TRUE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the instance data block "DCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#0
CMD_HIS	DBX16.0	TRUE

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

MCAT: Motor control-timer alarm

Description

You can use the "Motor control-timer alarm" instruction to accumulate the time from the point in time as of which one of the command inputs (open or close) is activated. The time is accumulated until the preset time is exceeded or the relevant feedback input indicates that the device has executed the requested operation within the specified time. If the preset time is exceeded before the feedback is received, the corresponding alarm is triggered.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction "Motor control-timer alarm" returns no error information.

Execution of the "Motor control-timer alarm" instruction

The following table shows the reactions of the instruction "Motor control-timer alarm" to the various input conditions:

Input parameters								Output parameters								
ET	O_HIS	C_HIS	O_CMD	C_CMD	S_CMD	O_FB	C_FB	OO	CO	OA	CA	ET	O_HIS	C_HIS	Q	Status
X	1	1	X	X	X	X	X	0	0	1	1	PT	0	0	0	Alarm

Input parameters								Output parameters								
X	X	X	X	X	X	1	1	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	1	X	X	0	0	0	0	X	0	0	1	Stop
X	X	X	1	1	X	X	X	0	0	0	0	X	0	0	1	Stop
X	0	X	1	0	0	X	X	1	0	0	0	0	1	0	1	Start opening
<PT	1	0	X	0	0	0	X	1	0	0	0	INC	1	0	1	Open
X	1	0	X	0	0	1	0	0	0	0	0	PT	1	0	1	Opened
>=PT	1	0	X	0	0	0	X	0	0	1	0	PT	1	0	0	Opening alarm
X	X	0	0	1	0	X	X	0	1	0	0	0	0	1	1	Start closing
<PT	0	1	0	X	0	X	0	0	1	0	0	INC	0	1	1	Close
X	0	1	0	X	0	0	1	0	0	0	0	PT	0	1	1	Closed
>=PT	0	1	0	X	0	X	0	0	0	0	1	PT	0	1	0	Closing alarm
X	0	0	0	0	0	X	X	0	0	0	0	X	0	0	1	Stopped

Legend:	
INC	Add the time difference (ms) since the last processing of the FB to ET
PT	PT is set to the same value as ET
X	Cannot be used
<PT	ET < PT
>=PT	ET >= PT

If the input parameters O_HIS and C_HIS both have the signal state "1", they are immediately set to signal state "0". In this case, the last row in the table (X) mentioned above is valid. Because it is therefore no longer possible to check whether the input parameters O_HIS and C_HIS have the signal state "1", the output parameters are set as follows in this case:

OO = FALSE

CO = FALSE

OA = FALSE

CA = FALSE

ET = PT

Q = TRUE

Parameters

The following table shows the parameters of the "Motor control-timer alarm" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
O_CMD	Input	BOOL	I, Q, M, D, L	"Open" command input
C_CMD	Input	BOOL	I, Q, M, D, L	"Close" command input
S_CMD	Input	BOOL	I, Q, M, D, L	"Stop" command input
O_FB	Input	BOOL	I, Q, M, D, L	Feedback input when opening

Parameter	Declaration	Data type	Memory area	Description
C_FB	Input	BOOL	I, Q, M, D, L	Feedback input when closing
OO	Output	BOOL	I, Q, M, D, L	"Open" output
CO	Output	BOOL	I, Q, M, D, L	"Close" output
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
Q	Output	BOOL	I, Q, M, D, L	A signal state of "0" indicates an error condition.
ET	Static	DINT	I, Q, M, D, L	Currently elapsed time, where one count = 1 ms
PT	Static	DINT	I, Q, M, D, L	Preset time value, where one count = 1 ms
PREV_TIME	Static	DWORD	I, Q, M, D, L	Previous system time
O_HIS	Static	BOOL	I, Q, M, D, L	"Open" history bit
C_HIS	Static	BOOL	I, Q, M, D, L	"Close" history bit

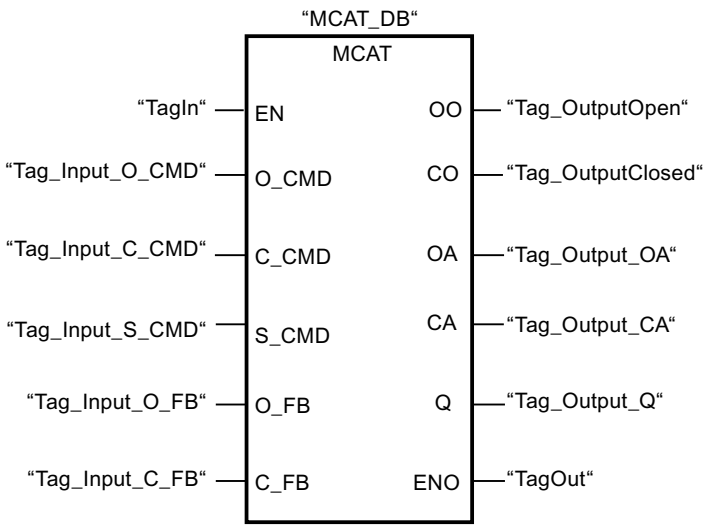
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.



The following table shows how the instruction works using specific operand values:

Before processing

In this example the following values are used for the input and output parameters:

Parameter	Operand	Value
O_CMD	Tag_Input_O_CMD	TRUE
C_CMD	Tag_Input_C_CMD	FALSE
S_CMD	Tag_Input_S_CMD	FALSE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
OO	Tag_OutputOpen	FALSE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	FALSE

The following values are saved in the instance data block "MCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#2
PT	DBD8	L#22
O_HIS	DBX16.0	TRUE
C_HIS	DBX16.1	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OO	Tag_OutputOpen	TRUE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	TRUE

The following values are saved in the instance data block "MCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#0
O_HIS	DBX16.0	TRUE
CMD_HIS	DBX16.1	FALSE

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

IMC: Compare input bits with the bits of a mask

Description

The instruction "Compare input bits with the bits of a mask" is used to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bits of a mask. Up to 16 steps with masks can be programmed. The value of the IN_BIT0 parameter is compared with the value of the mask CMP_VAL[x,0], with "x" indicating the step number. At the CMP_STEP parameter, specify the step number of the mask that is used for the comparison. All programmed values are compared in the same manner. Unprogrammed input bits or unprogrammed bits of the mask have the default signal state FALSE.

If a match is found in the comparison, the signal state of the OUT parameter is set to "1". Otherwise the OUT parameter is set to "0".

If the value of CMP_STEP parameter is greater than 15, the instruction is not executed. An error message is output at the ERR_CODE parameter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the "Compare input bits with the bits of a mask" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN_BIT0	Input	BOOL	I, Q, M, D, L	Input bit 0 to be compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L	Input bit 1 to be compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L	Input bit 2 to be compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L	Input bit 3 to be compared with bit 3 of the mask.

Parameter	Declaration	Data type	Memory area	Description
IN_BIT4	Input	BOOL	I, Q, M, D, L	Input bit 4 to be compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L	Input bit 5 to be compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L	Input bit 6 to be compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L	Input bit 7 to be compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L	Input bit 8 to be compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L	Input bit 9 to be compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L	Input bit 10 to be compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L	Input bit 11 to be compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L	Input bit 12 to be compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L	Input bit 13 to be compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L	Input bit 14 to be compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L	Input bit 15 to be compared with bit 15 of the mask.
CMP_STEP	Input	BYTE	I, Q, M, D, L, P or constant	The step number of the mask used for the comparison.
OUT	Output	BOOL	I, Q, M, D, L	A signal state of "1" indicates that a match was found. A signal state of "0" indicates that no match was found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
CMP_VAL	Static	ARRAY of BOOL	I, Q, M, D, L	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

For additional information on valid data types, refer to "See also".

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000A	The value at the CMP_STEP parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Switching display formats in the program status (Page 2980)

SMC: Compare scan matrix

Description

The "Compare scan matrix" instruction is used to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bits of the comparison masks for each step. Processing starts at step 1 and is continued until the last programmed step (LAST), or until a match is found. The input bit of the IN_BIT0 parameter is compared with the value of the mask CMP_VAL[x,0], with "x" indicating the step number. All programmed values are compared in the same manner. If a match is found, the signal state of the OUT parameter is set to "1" and the step number with the matching mask is written to the OUT_STEP parameter. Unprogrammed input bits or unprogrammed bits of the mask have the default signal state FALSE. If more than one step has a matching mask, only the first one found is indicated in the OUT_STEP parameter. If no match is found, the signal state of the OUT parameter is set to "0". In this case the value at the OUT_STEP parameter is greater by "1" than the value at the LAST parameter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the "Compare scan matrix" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN_BIT0	Input	BOOL	I, Q, M, D, L	Input bit 0 to be compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L	Input bit 1 to be compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L	Input bit 2 to be compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L	Input bit 3 to be compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L	Input bit 4 to be compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L	Input bit 5 to be compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L	Input bit 6 to be compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L	Input bit 7 to be compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L	Input bit 8 to be compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L	Input bit 9 to be compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L	Input bit 10 to be compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L	Input bit 11 to be compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L	Input bit 12 to be compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L	Input bit 13 to be compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L	Input bit 14 to be compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L	Input bit 15 to be compared with bit 15 of the mask.
OUT	Output	BOOL	I, Q, M, D, L	A signal state of "1" indicates that a match was found. A signal state of "0" indicates that no match was found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information

Parameter	Declaration	Data type	Memory area	Description
OUT_STEP	Output	BYTE	I, Q, M, D, L, P	Contains the step number with the matching mask, or the step number which is greater by "1" than the value at the LAST parameter, provided no match is found.
LAST	Static	BYTE	I, Q, M, D, L, P	Specifies the step number of the last step to be scanned for a matching mask.
CMP_VAL	Static	ARRAY of WORD	I, Q, M, D, L	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

For additional information on valid data types, refer to "See also".

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000E	The value at the LAST parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Switching display formats in the program status (Page 2980)

LEAD_LAG: Lead and lag algorithm

Description

The "Lead and lag algorithm" instruction is used to process signals with an analog tag. The gain value at the GAIN parameter must be greater than zero. The result of the instruction "Lead and lag algorithm" is calculated using the following equation:

$$OUT = \left[\frac{LG_TIME}{LG_TIME + SAMPLE_T}\right] PREV_OUT + GAIN \left[\frac{LD_TIME + SAMPLE_T}{LG_TIME + SAMPLE_T}\right] IN - GAIN \left[\frac{LD_TIME}{LG_TIME + SAMPLE_T}\right] * PREV_IN$$

The "Lead and lag algorithm" instruction supplies plausible results only when processing is in fixed program cycles. The same units must be specified for the LD_TIME, LG_TIME, and SAMPLE_T parameters. At LG_TIME > 4 + SAMPLE_T, the instruction approaches the following function:

OUT = GAIN * ((1 + LD_TIME * s) / (1 + LG_TIME * s)) * IN

When the value of the GAIN parameter is less than or equal to zero, the calculation is not performed and an error information is output on the ERR_CODE parameter.

You can use the instruction "Lead and lag algorithm" in conjunction with loops as a compensator in dynamic feed-forward control. The instruction consists of two operations. The "Lead" operation shifts the phase of the OUT output so that the output leads the input. The "Lag" operation, on the other hand, shifts the output so that the output lags behind the input. Because the "Lag" operation is equivalent to an integration, it can be used as a noise suppressor or as a low-pass filter. The "Lead" operation is equivalent to a differentiation and can therefore be used as a high-pass filter. The two instructions together (Lead and Lag) result in the output phase lagging behind the the input at lower frequencies and leading it at higher frequencies. This means that the "Lead and lag algorithm" instruction can be used as a band pass filter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Parameters

The following table shows the parameters of the "Lead and lag algorithm" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	REAL	I, Q, M, D, L, P or constant	The input value of the current sample time (cycle time) to be processed.
SAMPLE_T	Input	INT	I, Q, M, D, L, P or constant	Sample time

Parameter	Declaration	Data type	Memory area	Description
OUT	Output	REAL	I, Q, M, D, L, P	Result of the instruction
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
LD_TIME	Static	REAL	I, Q, M, D, L, P or constant	Lead time in the same unit as sample time.
LG_TIME	Static	REAL	I, Q, M, D, L, P or constant	Lag time in the same unit as sample time
GAIN	Static	REAL	I, Q, M, D, L, P or constant	Gain as % / % (the ratio of the change in output to a change in input as a steady state).
PREV_IN	Static	REAL	I, Q, M, D, L, P or constant	Previous input
PREV_OUT	Static	REAL	I, Q, M, D, L, P or constant	Previous output

For additional information on valid data types, refer to "See also".

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

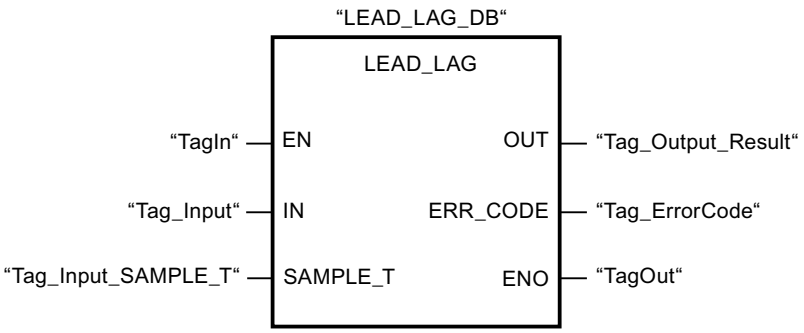
Error code* (W#16#...)	Explanation
0000	No error
0009	The value at the GAIN parameter is less than or equal to zero.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.



The following table shows how the instruction works using specific operand values:

Before processing

In this example the following values are used for the input parameters:

Parameter	Operand	Value
IN	Tag_Input	2.0
SAMPLE_T	Tag_InputSampleTime	10

The following values are saved in the instance data block "LEAD_LAG_DB" of the instruction:

Parameter	Address	Value
LD_TIME	DBD12	2.0
LG_TIME	DBD16	2.0
GAIN	DBD20	1.0
PREV_IN	DBD24	6.0
PREV_OUT	DBD28	6.0

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OUT	Tag_Output_Result	2.0

The following values are saved in the instance data block "LEAD_LAD_DB" of the instruction:

Parameter	Operand	Value
PREV_IN	DBD24	2.0
PREV_OUT	DBD28	2.0

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Switching display formats in the program status (Page 2980)

TONR_X: Time accumulator

Description

The instruction "Time accumulator" is used to accumulate timer values within a period set by the PT parameter. When the TMR_EN parameter changes to signal state "1", the instruction is executed and the time set at PT starts. The "Time accumulator" instruction adds the value of the DELTA_T parameter to the value of the ET parameter as long as the following conditions are satisfied:

- The TMR_EN parameter has the signal state "1".
- The timer value at the ET parameter is less than the timer value at the PT parameter.
- The RESET parameter has the signal state "0".

When the signal state on the TMR_EN parameter changes to "0", the time on the DELTA_T parameter is no longer added to the timer value at the ET parameter.

When the timer value at the ET parameter reaches the timer value at the PT parameter the Q parameter has the signal state "1". As long as the Q parameter has the signal state "1", the value at the ET parameter remains unchanged. The value at the ET parameter is reset to "0" when the value "1" is set on the RESET parameter.

Since the "Time accumulator" instruction uses the execution time (DELTA_T) of the last cycle of the organization block (OB), the instruction may only be called in cyclic organization blocks.

Note

You have to move the cycle time of the organization block from the startup information to the global tag of the DELTA_T parameters.

Parameters

The following table shows the parameters of the "Time accumulator" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
TMR_EN	Input	BOOL	I, Q, M, D, L	Activates the time for accumulating the timer value.
RESET	Input	BOOL	I, Q, M, D, L	Reset input
PV	Input	DINT	I, Q, M, D, L, P or constant	Maximum duration of time recording

Parameters	Declaration	Data type	Memory area	Description
DELTA_T	Input	INT	I, Q, M, D, L or constant	Cycle time of OB from previous cycle.
Q	Output	BOOL	Q, M, D, L	Output that is set when the PT time expires.
ET	InOut	DINT	I, Q, M, D, L	Current timer value of the elapsed time

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

WSR: Save data to shift register

Description

The "Save data to shift register" instruction is used to write a bit from a specified source to a shift register. Each element of the register is shifted into the next address. The data contained in the last address of the shift register is lost after the shift.

Each time the instruction is executed with the value "0" on the RESET parameter, new data are read from the source (S_DATA) and shifted to the start address (START) of the shift register. The LENGTH parameter specifies the number of elements to be shifted. You specify the data type of the elements on the E_TYPE parameter. The instruction is not executed if an invalid data type is specified on the E_TYPE parameter.

The register is deleted if the RESET parameter is set to "1" when the instruction is executed. The Q parameter is set to the signal state "1" when the shift register is filled with zeros or deleted.

Parameters

The following table shows the parameters of the "Save data to shift register" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
RESET	Input	BOOL	I, Q, M, D, L	Reset input

Parameter	Declaration	Data type	Memory area	Description
S_DATA	Input	POINTER	I, Q, M, D	Points to the source data element to be shifted to the start address (START). For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
START	Input	POINTER	I, Q, M, D	Points to the start address of the register. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
LEN	Input	WORD	I, Q, M, D, L, P	Number of elements to be shifted.
E_TYPE	Input	BYTE	I, Q, M, D, L, P	Specifies the data type of the elements to be shifted. The following data types are permitted: • W#16#04 = WORD • B#16#05 = INT • B#16#06 = DWORD • B#16#07 = DINT • B#16#08 = REAL
Q	Output	BOOL	Q, M, D, L	Is set to "0" if RESET parameter is active (1) or all elements to be shifted have the value "0".

Example

In the following example four elements of the data type WORD are shifted starting from the address specified on the START parameter. When the first element of the register has been

shifted to the next address, the first address is filled with the data that the pointer references on the S_DATA parameter. The value of the last element in the register is lost in the process.

The following table shows how the instruction works using specific operand values:

Before processing

In this example the following values are used for the input and output parameters:

Parameter	Operand	Value
RESET	Tag_Input_Reset	0
S_DATA	Tag_Input_DataSource	P#DB2.DBX6.0 DW0 = W#16#1111 DW2 = W#16#2222 DW4 = W#16#3333 DW6 = W#16#4444
START	Tag_Input_StartAddress	P#DB1.DBX0.0 DW6 = W#16#1234
LEN	Tag_Input_Length	W#16#04
E_TYPE	Tag_Input_ElementType	B#16#04
Q	Tag_Output_Q	1

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
RESET	Tag_Input_Reset	0
S_DATA	Tag_Input_DataSource	P#DB2.DBX6.0 DW0 = W#16#1234 DW2 = W#16#1111 DW4 = W#16#2222 DW6 = W#16#3333
START	Tag_Input_StartAddress	P#DB1.DBX0.0 DW6 = W#16#1234
LEN	Tag_Input_Length	W#16#04
E_TYPE	Tag_Input_ElementType	B#16#04
Q	Tag_Output_Q	1

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

SHRB: Shift bit to shift register

Description

The "Shift bit to shift register" instruction is used to write a bit from a specified source (DATA) to a shift register. Each time the instruction is executed with a value "0" on the RESET parameter, new data are read from the source and shifted to the start address (S_BIT) of the shift register. All following bits are shifted by one. The bit contained at the last address (S_BIT + N) is lost after the shift. If the RESET parameter is set to "1" during the processing of the instruction, the addresses in the shift register are reset to "0" and not shifted.

Parameters

The following table shows the parameters of the "Shift bit to shift register" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
DATA	Input	BOOL	I, Q, M, D, L	Source bit
RESET	Input	BOOL	I, Q, M, D, L	Reset input
S_BIT	Input	POINTER	I, Q, M, D	Points to the start bit of the shift register For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	WORD	I, Q, M, D, L, P	Number of bits to be shifted.

Example

In the following example the N parameter is set to the value 14 ("E" in hexadecimal). As a result 14 bits starting from the address specified on the S_BIT parameter are shifted. After the

bits have been shifted, the first address is filled with the data from the source on the DATA parameter. The last bit value is lost.

The following table shows how the instruction works using specific operand values:

Before processing

In this example the following values are used for the input and output parameters:

Parameters	Operand	Value
RESET	Tag_Input_Reset	0
DATA	Tag_Input_Data	0
S_BIT	Tag_Input_BitSource	P#DB1.DBX1.4
N	Tag_Input_Number	B#16#0E

Before processing the instruction the following signal states are stored in the 14 bits of the register:

1.7	...		S_BIT			...	1.0
0	0	1	1	1	0	1	0

2.7	...					...	2.0
0	0	1	1	1	0	1	0

3.7	...					...	3.0
0	0	1	1	1	0	1	0

After processing

After the processing of the instruction the following signal states are stored in the 14 bits of the register:

1.7	...		S_BIT			...	1.0
0	1	1	0	1	0	1	0

2.7	...					...	2.0
0	1	1	1	0	1	0	0

3.7	...					...	3.0
0	0	1	1	1	0	0	0

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

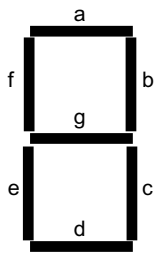
Inserting operands in FBD instructions (Page 2643)

SEG: Create bit pattern for seven-segment display

Description

The instruction "Create bit pattern for seven-segment display" is used to convert each of the four hexadecimal digits of the specified source word (IN) into an equivalent bit pattern for a seven-segment display. The result of the instruction is output in the double word on the OUT parameter.

The following relation exists between the hexadecimal digits and the assignment of the 7 segments (a, b, c, d, e, f, g):

Input digit (Binary)	Assignment of the segments - g f e d c b a	Display (Hexadecimal)	Seven-segment display
0000	00111111	0	
0001	00000110	1	
0010	01011011	2	
0011	01001111	3	
0100	01100110	4	
0101	01101101	5	
0110	01111101	6	
0111	00000111	7	
1000	01111111	8	
1001	01100111	9	
1010	01110111	A	
1011	01111100	B	
1100	00111001	C	
1101	01011110	D	
1110	01111001	E	
1111	01110001	F	

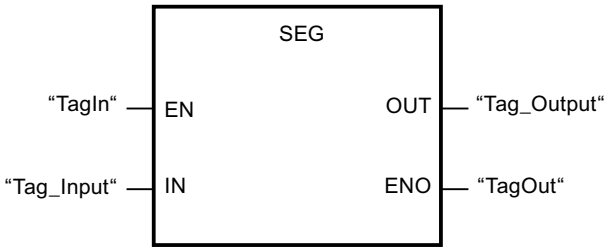
Parameters

The following table shows the parameters of the "Create bit pattern for seven-segment display" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	WORD	I, Q, M, D, L, P or constant	Source word with four hexadecimal digits
OUT	Output	DWORD	I, Q, M, D, L, P	Bit pattern for the seven-segment display

Example

The following example shows how the instruction works:



The following table shows how the instruction functions using specific values:

Parameters	Operand	Value	
Hexadecimal	Binary		
IN	Tag_Input	W#16#1234	0001 0010 0011 0100
OUT	Tag_Output	DW#16065B4F66	00000110 01011011 01001111 01100110 Display: 1234

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

BCDCPL: Create tens complement

Description

The "Create tens complement" instruction is used to create the tens complement of a seven-digit BCD number specified by the IN parameter. This instruction uses the following mathematical formula to calculate:

10000000 (as BCD)

– 7-digit BCD value

Tens complement (as BCD)

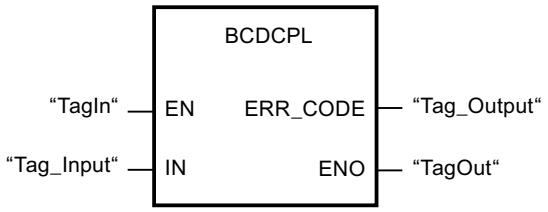
Parameters

The following table shows the parameters of the "Create tens complement" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	DWORD	I, Q, M, D, L, P or constant	7-digit BCD number
ERR_CODE	Output	DWORD	I, Q, M, D, L, P	Result of the instruction

Example

The following example shows how the instruction works:



The following table shows how the instruction functions using specific values:

Parameters	Operand	Value*
IN	Tag_Input	DW#16#01234567
ERR_CODE	Tag_Output	DW#16#08765433
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".		

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Switching display formats in the program status (Page 2980)

BITSUM: Count number of set bits

Description

The "Count number of set bits" instruction is used to count the number of bits of an operand that is set to the signal state "1". The operand whose bits are to be counted is specified on the IN parameter. The result of the instruction is output at the RET_VAL parameter.

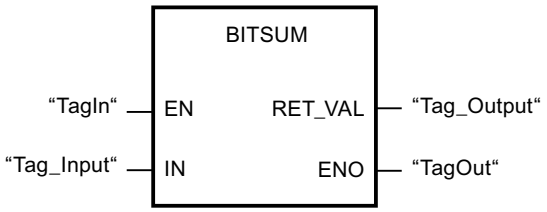
Parameters

The following table shows the parameters of the "Count number of set bits" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L, T, C	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	DWORD	I, Q, M, D, L, P or constant	Operand whose set bits are counted
RET_VAL	Output	INT	I, Q, M, D, L, P	Number of set bits

Example

The following example shows how the instruction works:



The following table shows how the instruction functions using specific values:

Parameter	Operand	Value*
IN	Tag_Input	DW#16#12345678
RET_VAL	Tag_Output	W#16#000D (13 Bits)
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".		

See also

Overview of the valid data types (Page 2236)

Basics of the EN/ENO mechanism (Page 2447)

Inserting FBD elements (Page 2621)

Editing FBD elements (Page 2635)

Inserting operands in FBD instructions (Page 2643)

Switching display formats in the program status (Page 2980)

STL

Timer operations

TP: Generate pulse

Description

You can use the "Generate pulse" instruction to set an operation for a programmed duration. The instruction is started when the result of logic operation (RLO) of the IN parameter changes from "0" to "1" (positive signal edge). The time programmed in the (PT) parameter begins when the instruction starts. The Q parameter is set for the period of time PT, regardless of the subsequent changes in the signal of the IN parameter. Even when a new positive signal edge is detected, the signal state of the Q parameter is not affected as long as the time duration PT is active.

You can query the current time value at the ET output. The timer value starts at T#0s and ends when the value of the time PT is reached. If the time duration PT is reached and the signal state at input IN is "0", the ET output is reset.

Note

If the timer is not called in the program because it is skipped, for example, output ET returns a constant value as soon as the timer has expired.

Use the "Call block" (CALL) instruction in the program code to call the "Generate pulse" instruction.

Each call of the "Generate pulse" instruction must be assigned an IEC timer in which the instruction data is stored. You can declare an IEC timer as follows:

- Declaration of a data block of the type TP (for example, "TP_DB")
- Declaration as a local tag of the type TP in the "Static" section of a block (for example, #MyTP_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens automatically, in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For more information on this topic, refer to "See also".

The instruction data is updated both when the instruction is called and also each time the outputs Q or ET are accessed.

The operating system resets the instances of the "Generate pulse" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized in a startup OB with the value "0" set for the PT parameter. If

instances of the "Generate pulse" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Syntax

Use the following syntax for the "Generate pulse" instruction:

```
CALL TP, "IEC timer"
TIME
IN := <Operand>
PT := := <Operand>
Q := := <Operand>
ET := <Operand>
```

Parameters

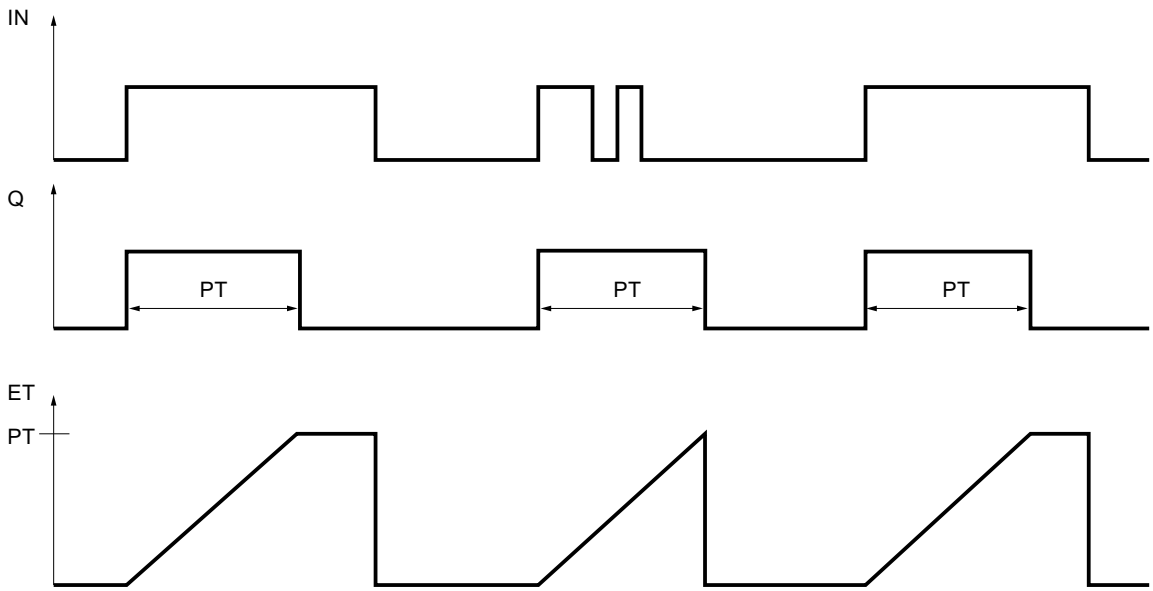
The following table shows the parameters of the "Generate pulse" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L	Start input
PT	Input	TIME	I, Q, M, D, L, P or constant	Duration of the pulse. The value specified at the PT parameter must be positive
Q	Output	BOOL	I, Q, M, D, L	Pulse output
ET	Output	TIME	I, Q, M, D, L	Current timer value

For more information on valid data types, refer to "See also".

Pulse diagram

The following figure shows the behavior of the "Generate pulse" instruction after the start:



Example

The following example shows how the instruction works:

STL	Explanation
CALL TP, "TP_DB"	// The "Generate pulse" instruction is called. The "TP_DB" data block is assigned to the instruction.
TIME	// Data type of the instruction
IN := "Tag_Start"	// The instruction executes on a rising signal edge of the operand.
PT := "Tag_PresetTIME"	// Duration of the pulse
Q := "Tag_Status"	// The operand is set for the duration that was specified by the "Tag_PresetTIME" operand.
ET := "Tag_Elapsed"	// Current timer value

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Call function blocks as single or multi-instances (Page 2182)

Setting status bits in timer and counter instructions (Page 2457)

TON: Generate on-delay

Description

You can use the "Generate on-delay" instruction to delay a positive signal edge by the time set for PT. The instruction is started when the result of logic operation (RLO) of the IN parameter changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. When the time PT has expired, the Q parameter has signal state "1". The operand of the Q parameter remains set as long as the IN start input is still "1". When the signal state of the start input changes from "1" to "0", the Q parameter is reset. The timer function is started again when a new positive signal edge is detected at the start input.

You can query the current time value at the ET output. The timer value starts at T#0s and ends when the value of the time PT is reached. If the time duration PT is reached and the signal state at input IN is "0", the ET output is reset.

Note

If the timer is not called in the program because it is skipped, for example, output ET returns a constant value as soon as the timer has expired.

In the program code, you call the "Generate on-delay" instruction with the "Call block" (CALL) instruction.

Each call of the "Generate on-delay" instruction must be assigned an IEC timer in which the instruction data is stored. You can declare an IEC timer as follows:

- Declaration of a data block of the type TON (for example, "TON_DB")
- Declaration as a local tag of the type TON in the "Static" section of a block (for example, #MyTON_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens automatically, in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For more information on this topic, refer to "See also".

The instruction data is updated both when the instruction is called and also each time the outputs Q or ET are accessed.

The operating system resets the instances of the "Generate on-delay" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized in a startup OB with the value "0" set for the PT parameter. If instances of the "Generate on-delay" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Syntax

Use the following syntax for the "Generate on-delay" instruction:

```
CALL TON, "IEC timer"
```

```
TIME
IN := <Operand>
PT := := <Operand>
Q := := <Operand>
ET := <Operand>
```

Parameters

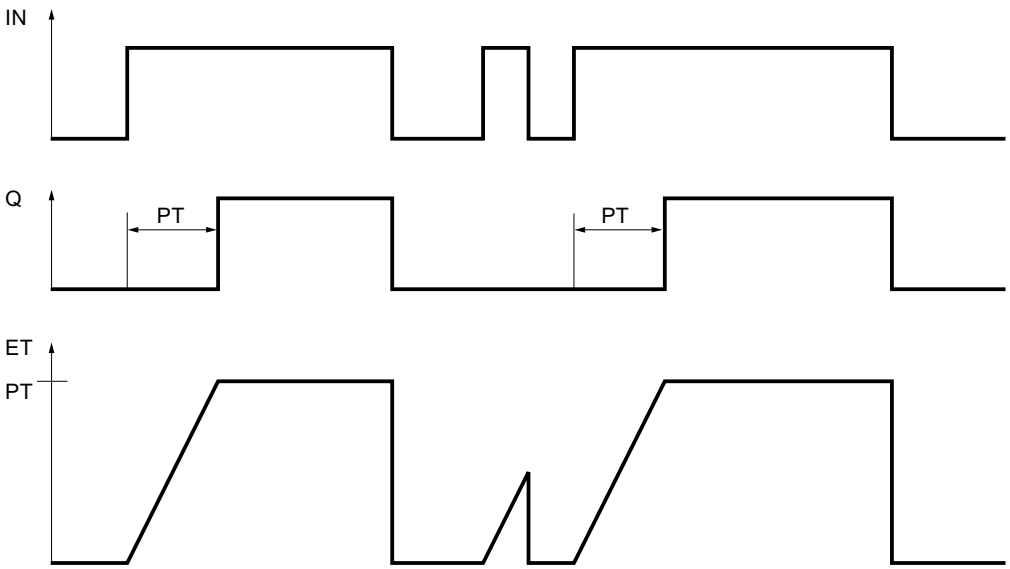
The following table shows the parameters of the "Generate on-delay" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L	Start input
PT	Input	TIME	I, Q, M, D, L, P or constant	Duration of the on-delay The value specified at the PT parameter must be positive
Q	Output	BOOL	I, Q, M, D, L	Signal state that is delayed by the time PT
ET	Output	TIME	I, Q, M, D, L	Current timer value

For more information on valid data types, refer to "See also".

Pulse diagram

The following figure shows the behavior of the "Generate on-delay" instruction after the start:



Example

The following example shows how the instruction works:

STL	Explanation
CALL TON, "TON_DB"	// The "Generate on-delay" instruction is called. The "TON_DB" data block is assigned to the instruction.
TIME	// Data type of the instruction
IN := "Tag_Start"	// The instruction executes on a rising signal edge of the operand.
PT := "Tag_PresetTIME"	// Specifies the time by which the rising signal edge of the IN parameter is delayed.
Q := "Tag_Status"	// The operand is set if the duration PT that was specified by the "Tag_PresetTIME" tag has expired.
	// The Q parameter remains set as long as the "Tag_Start" variable is "1".
	// When the signal state of the start input changes from "1" to "0", the operand of the Q parameter is reset.
ET := "Tag_ElapsedTIME"	// Current timer value

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Call function blocks as single or multi-instances (Page 2182)

Setting status bits in timer and counter instructions (Page 2457)

TOF: Generate off-delay

Description

You can use the "Generate off-delay" instruction to delay a negative signal edge by the time set for PT. The operand of the Q parameter is set when the result of logic operation (RLO) of the IN parameter changes from "0" to "1" (positive signal edge). When the signal state of the IN parameter changes back to "0", the programmed time PT starts. The operand of the Q parameter remains set as long the time duration PT is running. When the time PT expires, the Q parameter is reset. If the signal state of the IN parameter changes to "1" before the time PT expires, the timer is reset. The signal state of the Q parameter remains set to "1".

You can query the current time value set at the ET parameter. The current timer value starts at T#0s and ends when the value of the time duration PT is reached. When the time PT expires, the ET parameter remains set to the current value until the IN parameter changes back to "1".

Note

If the timer is not called in the program because it is skipped, for example, output ET returns a constant value as soon as the timer has expired.

In the program code, you call the "Generate off-delay" instruction with the "Call block" (CALL) instruction.

Each call of the "Generate off-delay" instruction must be assigned an IEC timer in which the instruction data is stored. You can declare an IEC timer as follows:

- Declaration of a data block of the type TOF (for example, "TOF_DB")
- Declaration as a local tag of the type TOF in the "Static" section of a block (for example, #MyTOF_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens automatically, in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For more information on this topic, refer to "See also".

The instruction data is updated both when the instruction is called and also each time the outputs Q or ET are accessed.

The operating system resets the instances of the "Generate off-delay" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized in a startup OB with the value "0" set for the PT parameter. If instances of the "Generate off-delay" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Syntax

Use the following syntax for the "Generate off-delay" instruction:

```
CALL TOF, "IEC timer"
TIME
IN := <Operand>
PT := := <Operand>
Q := := <Operand>
ET := <Operand>
```

Parameters

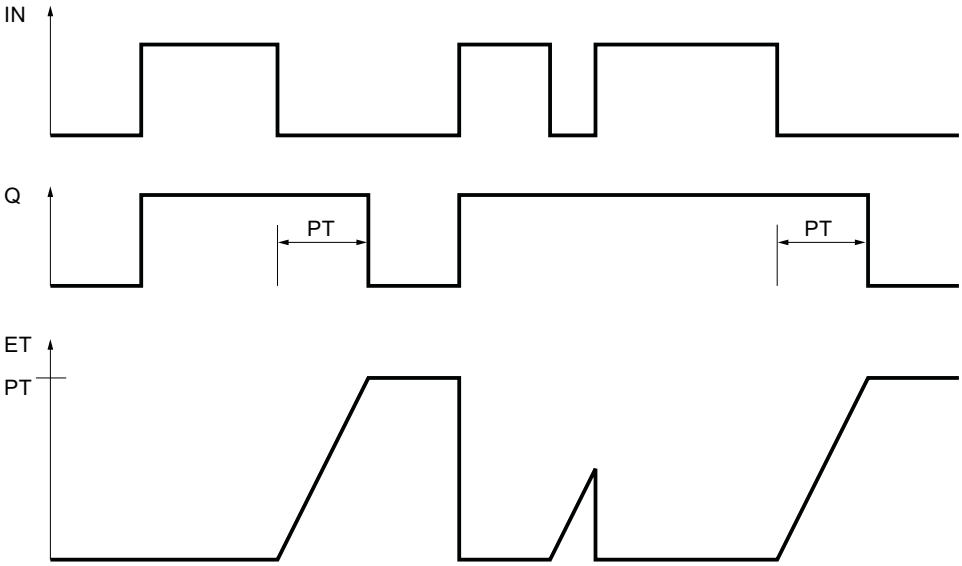
The following table shows the parameters of the "Generate off-delay" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L	Start input
PT	Input	TIME	I, Q, M, D, L, P or constant	Duration of the off delay The value specified at the PT parameter must be positive
Q	Output	BOOL	I, Q, M, D, L	Signal state that is delayed by the time PT
ET	Output	TIME	I, Q, M, D, L	Current timer value

For more information on valid data types, refer to "See also".

Pulse diagram

The following figure shows the behavior of the "Generate off-delay" instruction after the start:



Example

The following example shows how the instruction works:

STL	Explanation
`CALL TOF, "TOF_DB"`	`// The "Generate off-delay" instruction is called. The "TOF_DB" data block is assigned to the instruction.`

STL	Explanation
TIME	// Data type of the instruction
IN := "Tag_Start"	// The instruction executes on a rising signal edge of the operand.
PT := "Tag_PresetTIME"	// Specifies the time by which the falling signal edge of the IN parameter is delayed.
Q := "Tag_Status"	// The operand is set when the instruction is started by a rising signal edge of the IN parameter.
	// When the value of the IN parameter changes from "1" to "0", the "Tag_Status" operand remains set as long as the time specified by the "Tag_PresetTIME" tag is running.
	// When the time of the PT parameter expires, the operand is reset.
ET := "Tag_ElapsedTIME"	// Current timer value

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Call function blocks as single or multi-instances (Page 2182)

Setting status bits in timer and counter instructions (Page 2457)

Counter operations

CTU: Count up

Description

You can use the "Count up" instruction to increment the count value at the CV parameter. When the signal state of the CU parameter changes from "0" to "1" (positive signal edge), the instruction is executed and the current count value of the CV parameter is incremented by one. When the instruction is executed for the first time the current count of the CV parameter is set to zero. The count value is increased each time a positive signal edge is detected, until it reaches the high limit of the data type INT. When the high limit is reached, the signal state of the CU parameter no longer has an effect on the instruction.

You can query the count status of the Q parameter. The signal state of the Q parameter is determined by the PV parameter. When the current count value is greater than or equal to the value of the PV parameter, the Q parameter is set to signal state "1". In all other cases, the signal state of the Q parameter is "0". You can also specify a constant for the PV parameter.

The value of the CV parameter is reset to zero when the signal state at the R parameter changes to "1". As long as the R parameter has signal state "1", the signal state of the CU parameter has no effect on the instruction.

In the program code, you call the "Count up" instruction with the "Call block" (CALL) instruction.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count up" instruction must be assigned an IEC counter in which the instruction data is stored. You can declare an IEC counter as follows:

- Declaration of a data block of the type CTU (for example, "CTU_DB")
- Declaration as a local tag of the type CTU in the "Static" section of a block (for example, #MyCTU_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For more information on this topic, refer to "See also".

The operating system resets the instances of the "Count up" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized in a startup OB with the value "1" set for the R parameter of the instruction. If instances of the "Count up" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Syntax

Use the following syntax for the "Count up" instruction:

```
CALL CTU, "IEC counters"
<Data type>
CU := <Operand>
R := <Operand>
PV := <Operand>
Q := <Operand>
CV := <Operand>
```

Parameters

The following table shows the parameters of the "Count up" instruction:

Parameters	Declaration	Data type	Memory area	Description
CU	Input	BOOL	I, Q, M, D, L	Count input
R	Input	BOOL	I, Q, M, D, L, P	Reset input
PV	Input	INT	I, Q, M, D, L, P or constant	Value to which the output Q is set

Parameters	Declaration	Data type	Memory area	Description
Q	Output	BOOL	I, Q, M, D, L	Counter status
CV	Output	INT	I, Q, M, D, L, P	Current count value

For more information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
CALL CTU, "CTU_DB"	// The "Count up" instruction is called. The "CTU_DB" data block is assigned to the instruction.
INT	// Data type of the instruction
CU := "Tag_StartCTU"	// When the signal state of the "Tag_StartCTU" operand changes from "0" to "1", the instruction is executed and the current count value of the "Tag_CounterValue" operand is incremented by one.
	// The count value can be incremented until the high limit of INT = 32767 is reached.
R := "Tag_ResetCounter"	// When the signal state of the "Tag_ResetCounter" operand changes to "1", the "Tag_CounterValue" operand is reset to "0".
PV := "Tag_PresetValue"	// The operand determines the value at which the operand of the Q parameter is set.
Q := "Tag_CounterStatus"	// The operand is set as long as the current count value is greater than or equal to the value of the PV parameter.
CV := "Tag_CounterValue"	// Current count value

See also

Editing STL instructions (Page 2677)

Inserting STL instructions (Page 2663)

Overview of the valid data types (Page 2236)

Call function blocks as single or multi-instances (Page 2182)

Setting status bits in timer and counter instructions (Page 2457)

CTD: Count down

Description

You can use the "Count down" instruction to decrement the count value of the CV parameter. When the signal state of the CD parameter changes from "0" to "1" (positive signal edge), the instruction is executed and the current count value of the CV parameter is decremented by one. When the instruction executes the first time, the count value of the CV parameter will be

set to the value of the PV parameter. Each time a positive signal edge is detected, the counter is decremented until it reaches the low limit value of the specified data type INT. When the low limit is reached, the signal state of the CD parameter no longer has an effect on the instruction.

You can query the count status of the Q parameter. If the current count value is less than or equal to zero, the Q parameter is set to signal state "1". In all other cases, the signal state of the Q parameter is "0".

The value of the CV parameter is set to the value of the PV parameter when the signal state of the LD parameter changes to "1". As long as the LD parameter has signal state "1", the signal state of the CD parameter has no effect on the instruction.

In the program code, you call the "Count down" instruction with the "Call block" (CALL) instruction.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count down" instruction must be assigned an IEC counter in which the instruction data is stored. You can declare an IEC counter as follows:

- Declaration of a data block of the type CTD (for example, "CTD_DB")
- Declaration as a local tag of the type CTD in the "Static" section of a block (for example, #MyCTD_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For more information on this topic, refer to "See also".

The operating system resets the instances of "Count down" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized with the value "1" for the LD parameter of the instruction in a startup OB. In this case the desired initial value for the CV parameter is specified in the PV parameter. If instances of the "Count down" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Syntax

The following syntax is used for the "Count down" instruction:

```
CALL CTD, "IEC counters"
<Data type>
CD := <Operand>
LD := <Operand>
PV := <Operand>
Q := <Operand>
CV := <Operand>
```

Parameters

The following table shows the parameters of the "Count down" instruction:

Parameters	Declaration	Data type	Memory area	Description
CD	Input	BOOL	I, Q, M, D, L	Count input
LD	Input	BOOL	I, Q, M, D, L, P	Load input
PV	Input	INT	I, Q, M, D, L, P or constant	Value at which the Q output is set
Q	Output	BOOL	I, Q, M, D, L	Counter status
CV	Output	INT	I, Q, M, D, L, P	Current count value

For more information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL CTD, "CTD_DB"           // The "Count down" instruction is called. The
                             "CTD_DB" data block is assigned to the instruction.
INT                          // Data type of the instruction
CD := "Tag_StartCTD"         // When the signal state of the "Tag_StartCTD" operand
                             changes from "0" to "1", the instruction is executed
                             and the current count value of the "Tag_CounterValue"
                             operand is decremented by one.
                             // The count value of the CV parameter is decremented
                             until the low limit value of INT = -32768 is reached.
LD := "Tag_LoadPV"           // When the signal state of the "Tag_LoadPV" operand
                             changes to "1", the "Tag_CounterValue" operand is set
                             to the value of the "Tag_PresetValue" operand.
PV := "Tag_PresetValue"      // Specifies the value to which the counter is set
                             when the signal state is "1" in the LD parameter.
Q := "Tag_CounterStatus"     // The operand is set when the current count value is
                             less than or equal to zero.
CV := "Tag_CounterValue"     // Current count value
```

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Call function blocks as single or multi-instances (Page 2182)

Setting status bits in timer and counter instructions (Page 2457)

CTUD: Count up and down

Description

You can use the "Count up and down" instruction to increment and decrement the count value of the CV parameter. When the signal state of the CU parameter changes from "0" to "1" (positive signal edge), the current count value of the CV parameter is incremented by one. When the signal state of the CD parameter changes from "0" to "1" (positive signal edge), the current count value of the CV parameter is decremented by one. If there is a positive signal edge at the CU and CD inputs in one program cycle, the current count value of the CV parameter remains unchanged.

The count value can be incremented until it reaches the high limit value of the data type specified for the INT parameter. When the high limit value is reached, the count value is no longer incremented on a positive signal edge. When the low limit of the specified INT data type is reached, the count value is not decremented any further.

When the signal state of the LD parameter changes to "1", the count value of the CV parameter is set to the value of the PV parameter. As long as theLD parameter has signal state "1", the signal state of the CU and CD inputs has no effect on the instruction.

The count value is set to zero when the signal state of the R parameter changes to "1". As long as the R parameter has signal state "1", a change in the signal state of the CU, CD and LD parameters has no effect on the "Count up and down" instruction.

You can scan the current status of the up counter based on the value of the QU parameter. When the current count value is greater than or equal to the value of the PV parameter, the QU parameter is set to signal state "1". In all other cases, the signal state of the QU parameter is "0". You can also specify a constant for the PV parameter.

You can scan the current status of the down counter based on the value of the QD parameter. If the current count value is less than or equal to zero, the QD parameter is set to signal state "1". In all other cases, the signal state of the QD parameter is "0".

In the program code, you call the "Count up and down" instruction with the "Call block" (CALL) instruction.

Note

Only use a counter at a single point in the program to avoid the risk of counting errors.

Each call of the "Count up and down" instruction must be assigned an IEC counter in which the instruction data is stored. You can declare an IEC counter as follows:

- Declaration of a data block of the type CTUD (for example, "CTUD_DB")
- Declaration as a local tag of the type CTUD in the "Static" section of a block (for example, #MyCTUD_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multi-instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For more information on this topic, refer to "See also".

The operating system resets the instances of "Count up and down" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, you must call the instances to be initialized with the following parameter values in a restart OB:

- When used as counter up, the value at the R parameter must be set to "1".
- When used as counter down, the value at the LD parameter must be set to "1". In this case you specify the desired initial value for the CV parameter in the PV parameter.

If instances of the "Count up and down" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Syntax

The following syntax is used for the "Count up and down" instruction:

```
CALL CTUD, "IEC counters"
<Data type>
CU := <Operand>
CD := <Operand>
R := <Operand>
LD := <Operand>
PV := <Operand>
QU := <Operand>
QD := <Operand>
CV := <Operand>
```

Parameters

The following table shows the parameters of the "Count up and down" instruction:

Parameters	Declaration	Data type	Memory area	Description
CU	Input	BOOL	I, Q, M, D, L	Count up input
CD	Input	BOOL	I, Q, M, D, L	Count down input
R	Input	BOOL	I, Q, M, D, L, P	Reset input
LD	Input	BOOL	I, Q, M, D, L, P	Load input
PV	Input	INT	I, Q, M, D, L, P or constant	Value at which the QU output is set
QU	Output	BOOL	I, Q, M, D, L	Status of the counter up
QD	Output	BOOL	I, Q, M, D, L	Status of the counter down
CV	Output	INT	I, Q, M, D, L, P	Current count value

For more information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
CALL CTUD, "CTUD_DB"	// The "Count up and down" instruction is called. The "CTUD_DB" data block is assigned to the instruction.
INT	// Data type of the instruction
CU := "Tag_StartCTU"	// When the signal state of the "Tag_StartCTU" operand changes from "0" to "1", the instruction is executed and the current count value of the "Tag_CounterValue" operand is incremented by one.
	// The count value is incremented on a rising edge of the CU parameter, until it reaches the high limit value of INT = 32767.
CD := "Tag_StartCTD"	// When the signal state of the "Tag_StartCTD" operand changes from "0" to "1", the instruction is executed and the current count value of the "Tag_CounterValue" operand is decremented by one.
	// The count value of the CV parameter is decremented until the low limit value of -32768 is reached.
R := "Tag_ResetCounter"	// When the signal state of the "Tag_ResetCounter" operand changes to "1", the "Tag_CounterValue" operand is reset to "0".
LD := "Tag_LoadPV"	// When the signal state of the "Tag_LoadPV" operand changes to "1", the "Tag_CounterValue" operand is set to the value of the "Tag_PresetValue" operand.
PV := "Tag_PresetValue"	// Specifies the value to which the counter is set when the signal state is "1" in the LD parameter.
QU := "Tag_CounterStatus"	// The operand is set as long as the current count value is greater than or equal to the value of the PV parameter.
QD := "Tag_CounterStatus"	// The operand is set when the current count value is less than or equal to zero.
CV := "Tag_CounterValue"	// Current count value

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Call function blocks as single or multi-instances (Page 2182)

Setting status bits in timer and counter instructions (Page 2457)

Math functions

MIN: Get minimum

Description

The "Get minimum" instruction compares the values at the inputs IN1, IN2 and IN3 and writes the lowest value to the OUT output. The instruction is only executed if the tags of all inputs are of the same data type.

The value of the OUT parameter is invalid if one of the following conditions is met:

- The specified operands are not of the same data type.
- A floating-point number has an invalid value.

Note

Only the data types INT, DINT and REAL are permitted at the input parameters even with deactivated IEC check. If you use the data types WORD or DWORD at the input parameters, the instruction returns an undefined result.

Syntax

The following syntax is used for the "Get minimum" instruction:

```
CALL MIN
<Data type>
IN1 := <Operand>
IN2 := <Operand>
IN3 := <Operand>
OUT := <Operand>
```

You can select the data type of the instruction from the "<data type>" drop-down list.

Parameters

The following table shows the parameters of the "Get minimum" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	INT, DINT, REAL	I, Q, M, D, L, P	First input value
IN2	Input	INT, DINT, REAL	I, Q, M, D, L, P	Second input value
IN3	Input	INT, DINT, REAL	I, Q, M, D, L, P	Third input value
OUT	Output	INT, DINT, REAL	I, Q, M, D, L, P	Result

For more information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL MIN                     // The "Get minimum" instruction is called.
INT                          // Data type of the instruction
IN1 := "TagIn_Value1"        // First input value to be compared.
IN2 := "TagIn_Value2"        // Second input value to be compared.
IN3 := "TagIn_Value3"        // Third input value to be compared.
OUT := "Tag_Minimum"         // Lowest value of the comparison
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
IN3	TagIn_Value3	13333
OUT	Tag_Minimum	12222

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

MAX: Get maximum

Description

The "Get maximum" instruction compares the values at the inputs IN1, IN2 and IN3 and writes the highest value to the OUT output. The instruction is only executed if the tags of all inputs are of the same data type.

The value of the OUT output is invalid if one of the following conditions is fulfilled:

- The specified operands are not of the same data type.
- An operand has an invalid value.

Note

Only the data types INT, DINT and REAL are permitted at the input parameters even with deactivated IEC check. If you use the data types WORD or DWORD at the input parameters, the instruction returns an undefined result.

Syntax

The following syntax is used for the "Get maximum" instruction:

```
CALL MAX
<Data type>
IN1 := <Operand>
IN2 := <Operand>
IN3 := <Operand>
OUT := <Operand>
```

You can select the data type of the instruction from the "<data type>" drop-down list.

Parameters

The following table shows the parameters of the "Get maximum" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	INT, DINT, REAL	I, Q, M, D, L, P	First input value
IN2	Input	INT, DINT, REAL	I, Q, M, D, L, P	Second input value
IN3	Input	INT, DINT, REAL	I, Q, M, D, L, P	Third input value
OUT	Output	INT, DINT, REAL	I, Q, M, D, L, P	Result

For more information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL MAX                     // The "Get maximum" instruction is called.
INT                          // Data type of the instruction
IN1 := "TagIn_Value1"        // First input value to be compared.
IN2 := "TagIn_Value2"        // Second input value to be compared.
IN3 := "TagIn_Value3"        // Third input value to be compared.
OUT := "Tag_Maximum"         // Highest value of the comparison
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
IN3	TagIn_Value3	13333
OUT	Tag_Maximum	14444

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

LIMIT: Set limit value

Description

You can use the "Set limit value" instruction to limit the value at the IN input to the values at the inputs MN and MX. If the value at the IN input meets the MN condition <= IN <= MX, it is copied to the OUT output. If the condition is not fulfilled and the IN input value is below the low limit MN, the OUT output is set to the value of the MN input. If the high limit MX is exceeded, the OUT output is set to the value of the MX input.

The instruction is only executed if all the tags at the inputs are of the same data type.

The value of the OUT output is invalid if one of the following conditions is fulfilled:

- The specified tags are not of the same data type.
- An operand has an invalid value.
- The value at the MN parameter is greater than the value at the MX parameter.

Note

Only the data types INT, DINT and REAL are permitted at the input parameters even with deactivated IEC check. If you use the data types WORD or DWORD at the input parameters, the instruction returns an undefined result.

Syntax

The following syntax is used for the "Set limit value" instruction:

```
CALL LIMIT
<Data type>
MN := <Operand>
```

```
IN := <Operand>
MX := <Operand>
OUT := <Operand>
```

You can select the data type of the instruction from the "<data type>" drop-down list.

Parameters

The following table shows the parameters of the "Set limit value" instruction:

Parameters	Declaration	Data type	Memory area	Description
MN	Input	INT, DINT, REAL	I, Q, M, D, L, P	Low limit
IN	Input	INT, DINT, REAL	I, Q, M, D, L, P	Input value
MX	Input	INT, DINT, REAL	I, Q, M, D, L, P	High limit
OUT	Output	INT, DINT, REAL	I, Q, M, D, L, P	Result

For more information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL LIMIT                   // The "Set limit value" instruction is called.
INT                          // Data type of the instruction
MN := "Tag_LowLimit"         // Low limit
IN := "Tag_InputValue"       // Input value
MX := "Tag_HighLimit"        // High limit
OUT := "Tag_Result"          // Result
```

The following table shows how the instruction works using specific operand values:

Parameters	Operand	Value
MN	Tag_LowLimit	12000
IN	Tag_InputValue	8000
MX	Tag_HighLimit	16000
OUT	Tag_Result	12000

The "Tag_Result" operand is written with the value of the "Tag_LowLimit" operand because the value of the "Tag_InputValue" operand is outside the defined limit range.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

Move operations

BLKMOV: Move block

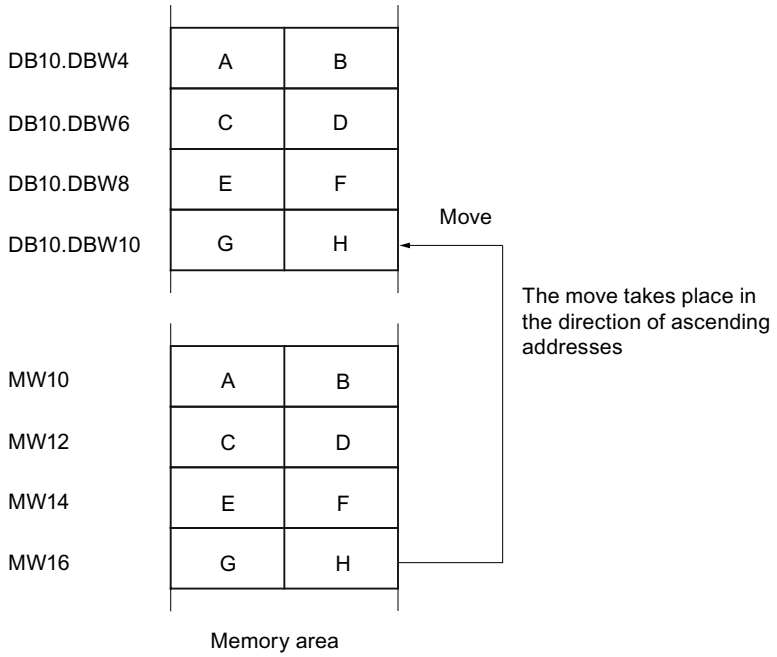
Description

You can use the "Move block" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use ANY pointer to define the source and destination area.

Note

In this instruction, you can only use data types which are valid on a CPU of the S7-300/400 series.

The following figure shows the principle of the move operation:



Consistency of source and destination data

Make sure that the source data remain unchanged during execution of the "Move block" instruction. Otherwise, consistency of the destination data cannot be guaranteed.

Interruptibility

As long as the source area is not part of a data block that only exists in the load memory, there is no limit to the nesting depth.

If, however, BLKMOV is interrupted while copying from a DB that is not runtime-relevant, the execution of BLKMOV can no longer be nested.

Memory areas

You can use the "Move block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output
- Data blocks not runtime-relevant

General rules for moving

The source and destination areas must not overlap. If the source and destination area have different lengths, only the length of the smaller area will be moved.

If the source area is less than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is less than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If an area of data type BOOL is copied, the specified length of the area must be divisible by 8, otherwise it will not be possible to execute the instruction.

Note

If the actual source or destination area is smaller than a destination or source area specified on the SRCBLK or DSTBLK parameter, behavior will depend on the CPU type.

- The following applies to S7-300 CPUs:
  No data are transferred. Error code W#16#837F is output at the RET_VAL parameter. This error code has to be evaluated in your user program.
- The following applies to S7-400 V4-CPUs (V4, V4 H-CPU and V4.5. H-CPU):
  No data are transferred. Error code W#16#8122 or W#16#8323 is output at the RET_VAL parameter. This error code has to be evaluated in your user program.
- For all other S7-400-CPUs:
  Data are transferred. Error code W#16#8122 or W#16#8323 is output at the RET_VAL parameter. This error code has to be evaluated in your user program.

Rules for moving character strings

You can also use the "Move block" instruction to move source and destination areas of the STRING data type. If only the source area is data type STRING, the characters are moved

that are actually contained in the character string. Information on the actual and maximum length is also written to the destination area. If the source and destination area are each STRING data type, the current length of the character string in the destination area is set to the number of actually moved characters.

If you want to move the information on the maximum and actual length of a character string, specify the areas in bytes to the SRCBLK and DSTBLK parameters.

Rules for moving data blocks that are not runtime-relevant

The source area is in a data block in load memory that is not runtime-relevant. Data blocks that are not runtime-relevant are indicated by the keyword UNLINKED.

If a data block that is not runtime-relevant is copied to work memory with the "Move block" instruction and loaded at the same time, for example, by the programming device, the execution of the instruction can be delayed for several milliseconds. This results in a longer OB cycle and may trip the cycle monitoring.

If a data block that is not runtime-relevant is moved with the "Move block" instruction and the move operation is interrupted, the execution of the instruction can no longer be continued.

If the "Read from data block in the load memory" instruction is available on your CPU, you will need to use this instruction to read data blocks that are not runtime-relevant from load memory. If you use the "Move block" instruction, error W#16#8092 is output.

Syntax

The following syntax is used for the "Move block" instruction:

```
CALL BLKMOV
ANY
SRCBLK := <Operand>
RET_VAL := <Operand>
DSTBLK := <Operand>
```

Parameters

The following table shows the parameters of the "Move block" instruction:

Parameters	Declaration	Data type	Memory area	Description
SRCBLK	Input	ANY	I, Q, M, D, L, P	Specifies the memory area to be moved (source area).
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.
DSTBLK	Output	ANY	I, Q, M, D, L, P	Specifies the memory area to which the block is to be moved (destination area).

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8092	The source or target area is only available in the load memory.
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For more information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL BLKMOV                  // The "Move block" instruction is called.
ANY                          // Data type of the instruction
SRCBLK := P#M100.0 BYTE 10   // 10 Bytes starting from MB100 are moved.
RET_VAL := "Tag_ErrorCode"   // If an error occurs during the move operation, its
                             error code is output in the variable "Tag_ErrorCode".
DSTBLK := P#DB1.DBX0.0 BYTE  // The moved bytes are transferred to DB1.
10
```

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Setting the status bits in instructions with integers (Page 2458)

Switching display formats in the program status (Page 2980)

UBLKMOV: Move block uninterruptible

Description

You can use the "Move block uninterruptible" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use ANY pointer to define the source and destination area.

The copy operation cannot be interrupted by other operating system activities. As a result, the interrupt reaction time of the CPU can increase during execution of the "Move block uninterruptible" instruction.

Note

In this instruction, you can only use data types which are valid on a CPU of the S7-300/400 series.

Memory areas

You can use the "Move block uninterruptible" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output

General rules for moving

The source and destination area must not overlap during execution of the "Move block uninterruptible" instruction. If the source area is less than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is less than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a source or destination area defined as a formal parameter is less than a destination or source area specified on the SRCBLK or DSTBLK parameter, No data are transferred.

If an area of data type BOOL is copied, the specified length of the area must be divisible by 8, otherwise it will not be possible to execute the instruction.

You can use the "Move block uninterruptible" instruction to move a maximum of 512 bytes. Note the CPU specific restrictions for this.

Note

If the actual source or destination area is smaller than a destination or source area specified on the SRCBLK or DSTBLK parameter, behavior will depend on the CPU type.

- The following applies to S7-300 CPUs:
  No data are transferred. Error code W#16#837F is output at the RET_VAL parameter. This error code has to be evaluated in your user program.
- The following applies to S7-400 V4-CPUs (V4, V4 H-CPU and V4.5. H-CPU):
  No data are transferred. Error code W#16#8122 or W#16#8323 is output at the RET_VAL parameter. This error code has to be evaluated in your user program.
- For all other S7-400-CPUs:
  Data are transferred. Error code W#16#8122 or W#16#8323 is output at the RET_VAL parameter. This error code has to be evaluated in your user program.

Rules for moving character strings

You can also use the "Move block uninterruptible" instruction to move source and destination areas of the STRING data type. If only the source area is data type STRING, the characters will be moved that are actually contained in the character string. Information on the actual and maximum length are not written in the destination area. If the source and destination area are each STRING data type, the current length of the character string in the destination area is set to the number of actually moved characters. If blocks of the data type STRING are moved, you have to specify "1" as the block length.

Syntax

The following syntax is used for the "Move block uninterruptible" instruction:

```
CALL UBLKMOV
SRCBLK := <Operand>
RET_VAL := <Operand>
DSTBLK := <Operand>
```

Parameters

The following table shows the parameters of the "Move block uninterruptible" instruction:

Parameters	Declaration	Data type	Memory area	Description
SRCBLK	Input	ANY	I, Q, M, D, L, P	Specifies the memory area to be moved (source area).
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.
DSTBLK	Output	ANY	I, Q, M, D, L, P	Specifies the memory area to which the block is to be moved (destination area).

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8091	The source area is in a data block that is not runtime-relevant.
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For more information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

STL	Explanation
CALL UBLKMOV	// The "Move block uninterruptible" instruction is called.
SRCBLK := P#M100.0 BYTE 10	// 10 Bytes starting from MB100 are moved.
RET_VAL := "Tag_ErrorCode"	// If an error occurs during the move operation, its error code is output in the variable "Tag_ErrorCode".
DSTBLK := P#M200.0 BYTE 10	// The moved bytes are written to the 10 bytes starting from MB200

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Setting the status bits in instructions with integers (Page 2458)

Switching display formats in the program status (Page 2980)

FILL: Fill block

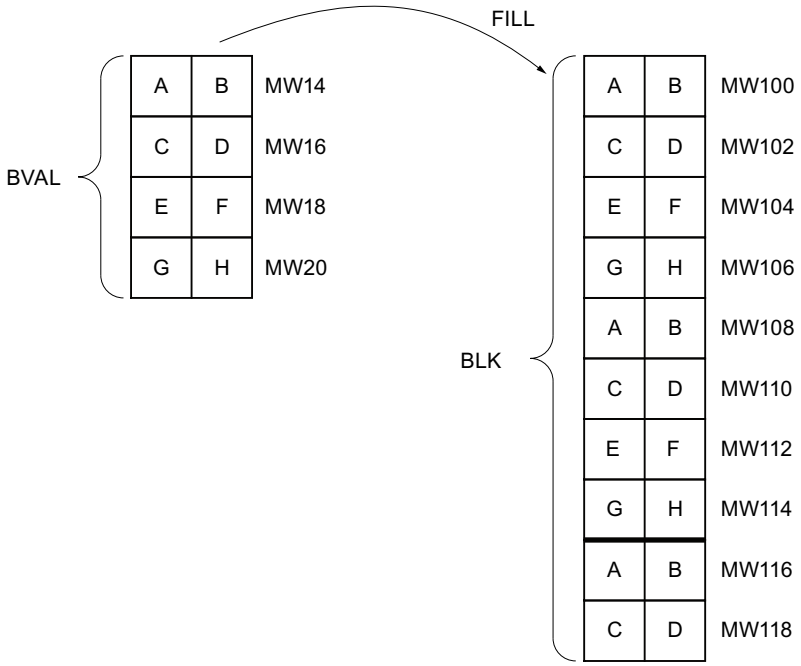
Description

You can use the "Fill block" instruction to fill a memory area (destination area) with the content of another memory area (source area). The "Fill block" instruction moves the content of the source area to the destination area until the destination area is completely written. The move operation takes place in the direction of ascending addresses. You use ANY pointer to define the source and destination area.

Note

In this instruction, you can only use data types which are valid on a CPU of the S7-300/400 series.

The following figure shows the principle of the move operation:



Example: The contents of the range MW100 to MW118 are to be preassigned with the contents of the memory words MW14 to MW20.

Consistency of source and destination data

Make sure that the source data remains unchanged during execution of the instruction "Fill block", since otherwise consistency of the destination data cannot be guaranteed.

Memory areas

You can use the "Fill block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image of the inputs
- Process image of the outputs
- Data blocks not relevant for program execution

General rules for moving

The source and destination areas must not overlap. If the destination area to be preset is not an integer multiple of the length of the input parameter BVAL, the destination block is nevertheless written up to the last byte.

If the destination area to be preset is smaller than the source block, the function only copies as much data as can be written to the destination block.

If the destination or source block actually present is smaller than the assigned memory area for the source or destination block (BVAL, BLK parameters), No data is transferred.

If the ANY pointer (source or destination) is of the data type BOOL, the specified length of the area must be divisible by 8; otherwise the instruction will not be executed.

If the destination block is of data type STRING, then the instruction describes the entire string including the administration information.

Note

If the actual source or destination area is smaller than a destination or source area specified on the BVAL or BLK parameter, behavior will depend on the CPU type.

- The following applies to S7-300 CPUs:
  No data is transferred. Error code message W#16#837F is output on the RET_VAL parameter. This error code has to be evaluated in your user program.
- The following applies to S7-400 V4-CPUs (V4, V4 H-CPU and V4.5. H-CPU):
  No data is transferred. Error code message W#16#8122 or W#16#8323 is output on the RET_VAL parameter. This error code has to be evaluated in your user program.
- For all other S7-400-CPUs:
  No data is transferred. Error code message W#16#8122 or W#16#8323 is output on the RET_VAL parameter. This error code has to be evaluated in your user program.

Rules for moving structures

When you transfer a structure as an input parameter you must bear in mind that the length of a structure is always oriented to an even number of bytes. The structure will need one byte of additional memory space if you declare a structure with an odd number of bytes.

Syntax

The following syntax is used for the "Fill block" instruction:

```
CALL FILL
ANY
BVAL := <Operand>
RET_VAL := <Operand>
BLK := <Operand>
```

Parameters

The following table shows the parameters of the "Fill block" instruction:

Parameters	Declaration	Data type	Memory area	Description
BVAL	Input	ANY	I, Q, M, D, L	Specification of the memory area (source area) with whose content the destination area on the BLK parameter will be filled.
RET_VAL	Output	INT	I, Q, M, D, L	Error information: If an error occurs during the execution of the instruction, an error code is output on the RET_VAL parameter.
BLK	Output	ANY	I, Q, M, D, L	Specification of the memory area that will be filled with the content of the source area.

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For more information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL FILL                    // The "Fill block" instruction is called.
ANY                          // Data type of the instruction
BVAL := P#M14.0 WORD 4       // The area from MW14 to MW20 is moved
RET_VAL := "Tag_ErrorCode"   // Error information
BLK := P#M100.0 WORD 10      // The memory area from MW100 to MW118 is filled with
                             the content of 4 words that are contained in the
                             memory area on the BVAL parameter
```

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Setting the status bits in instructions with integers (Page 2458)

Switching display formats in the program status (Page 2980)

Conversion operations

SCALE: Scale

Description

Use the "Scale" instruction to convert the integer at the IN parameter into a floating-point number that can be scaled in physical units between a low limit value and a high limit value. You use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is scaled. The result of the instruction is output at the OUT parameter.

The "Scale" instruction works with the following equation:

OUT = [((FLOAT (IN) – K1)/(K2–K1)) * (HI_LIM–LO_LIM)] + LO_LIM

The values of the constants "K1" and "K2" are determined by the signal state of the BIPOLAR parameter. The following signal states are possible at the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case the "K1" constant has the value "-27648.0" and the "K2" constant the value "+27648.0".
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case the "K1" constant has the value "0.0" and the "K2" constant the value "+27648.0".

When the value at the IN parameter is greater than the value of the constant "K2", the result of the instruction is set to the value of the high limit (HI_LIM) and an error is output.

When the value at the IN parameter is less than the value of the constant "K1", the result of the instruction is set to the value of the low limit value (LO_LIM) and an error is output.

When the indicated low limit value is greater than the high limit value (LO_LIM > HI_LIM), the result is scaled inversely proportionate to the input value.

Syntax

The following syntax is used for the "Scale" instruction:

```
CALL SCALE
IN := <Operand>
HI_LIM := <Operand>
LO_LIM := <Operand>
```

```
BIPOLAR := <Operand>
RET_VAL := <Operand>
OUT := <Operand>
```

Parameters

The following table shows the parameters of the "Scale" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	INT	I, Q, M, D, L, P or constant	Input value to be scaled
HI_LIM	Input	REAL	I, Q, M, D, L, P or constant	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P or constant	Low limit
BIPOLAR	Input	BOOL	I, Q, M, D, L or constant	Indicates whether the value at the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
RET_VAL	Output	WORD	I, Q, M, D, L, P	Error information
OUT	Output	REAL	I, Q, M, D, L, P	Result of the instruction

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the constant "K2" or less than the value of the constant "K1"
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For more information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL SCALE                   // The "Scale" instruction is called
IN := "Tag_InputValue"       // Specifies the value to be converted and scaled
HI_LIM := "Tag_HighLimit"    // High limit
LO_LIM := "Tag_LowLimit"     // Low limit
BIPOLAR := "Tag_Bipolar"     // Indicates whether the value at the IN parameter is
                             to be interpreted as bipolar or unipolar
RET_VAL := "Tag_ErrorCode"   // Error information
OUT := "Tag_OutputValue"     // Result of the instruction
```

The following table shows the values of the various operands before execution of the instruction:

Parameters	Operand	Value
IN	Tag_InputValue	22
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	0.0

The following table shows the values of the various operands after execution of the instruction:

Parameter	Operand	Value
IN	Tag_InputValue	22
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	50.03978588

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Setting the status bits in instructions with floating-point numbers (Page 2459)

Setting the status bits in instructions with integers (Page 2458)

Switching display formats in the program status (Page 2980)

UNSCALE: Unscale

Description

The "Unscale" instruction unscales the floating-point number at the IN parameter in physical units between a low limit value and a high limit value and converts it to an integer. You use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is unscaled. The result of the instruction is output at the OUT parameter.

The "Unscale" instruction works with the following equation:

OUT = [((IN–LO_LIM)/(HI_LIM–LO_LIM)) * (K2–K1)] + K1

The values of the constants "K1" and "K2" are determined by the signal state of the BIPOLAR parameter. The following signal states are possible at the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case the "K1" constant has the value "-27648.0" and the "K2" constant the value "+27648.0".
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case the "K1" constant has the value "0.0" and the "K2" constant the value "+27648.0".

When the value at the IN parameter is greater than the value of the constant "HI_LIM", the result of the instruction is set to the value of the constant (K2) and an error is output.

When the value at the IN parameter is less than the value of the constant of the low limit (LO_LIM), the result of the instruction is set to the value of the constant (K1) and an error is output.

Syntax

The following syntax is used for the "Unscale" instruction:

```
CALL UNSCALE
IN := <Operand>
HI_LIM := <Operand>
LO_LIM := <Operand>
BIPOLAR := <Operand>
RET_VAL := <Operand>
OUT := <Operand>
```

Parameters

The following table shows the parameters of the "Unscale" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	REAL	I, Q, M, D, L, P or constant	Input value to be unscaled to an integer value
HI_LIM	Input	REAL	I, Q, M, D, L, P or constant	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P or constant	Low limit
BIPOLAR	Input	BOOL	I, Q, M, D, L or constant	Indicates whether the value at the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
RET_VAL	Output	WORD	I, Q, M, D, L, P	Error information
OUT	Output	INT	I, Q, M, D, L, P	Result of the instruction

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the high limit (HI_LIM) or less than the value of the low limit (LO_LIM).
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For more information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL UNSCALE                 // The "Unscale" instruction is called
IN := "Tag_InputValue"       // Specifies the value to be unscaled
HI_LIM := "Tag_HighLimit"    // High limit
LO_LIM := "Tag_LowLimit"     // Low limit
```

```
STL                              Explanation
BIPOLAR := "Tag_Bipolar"         // Indicates whether the value at the IN parameter is
                                 to be interpreted as bipolar or unipolar
RET_VAL := "Tag_ErrorCode"       // Error information
OUT := "Tag_OutputValue"         // Result of the instruction
```

The following table shows the values of the various operands before execution of the instruction:

Parameters	Operand	Value
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	0.0

The following table shows the values of the various operands after execution of the instruction:

Parameters	Operand	Value
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	22

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

Switching display formats in the program status (Page 2980)

Program control operations

Runtime control

COMPRESS: Compress CPU memory

Description

The "Compress CPU memory" instruction is used to compress the working memory of the CPU or the RAM load memory. During this compression the gaps caused by deleting or loading blocks are filled by pushing together the existing blocks.

The compression operation is started by calling the "Compress CPU memory" instruction. The compression operation is divided into several program cycles. The compression operation is monitored by the values of the individual parameters. If value "1" is on the BUSY parameter of the instruction, the compression operation is still being executed. The value "1" on the DONE parameter reports that the compression operation has been completed. Error information can be queried on the RET_VAL parameter.

If an externally started compression operation has already been executed, calling the "Compress CPU memory" instruction will cause an error display.

Checking the compression function

Calling the instruction "Compress CPU memory" once starts the compression function. However, you have no control over whether the compression is performed successfully.

If you want to check the compression function, follow the steps outlined below:

The "Compress CPU memory" instruction must be called in the cycle. First evaluate the RET_VAL parameter after every call. If it has value "0", the parameters BUSY and DONE has to be evaluated. If BUSY = "1" and DONE = "0", this indicates that the compression function is still active. The compression has not been completed successfully until BUSY changes to value "0" and DONE to the value "1". Compression is repeated if the "Compress CPU memory" instruction is called again afterwards.

Syntax

The following syntax is used for the "Compress CPU memory" instruction:

```
CALL COMPRESS
RET_VAL := <Operand>
BUSY := <Operand>
DONE := <Operand>
```

Parameters

The following table shows the parameters of the "Compress CPU memory" instruction:

Parameters	Declaration	Data type	Memory area	Description
RET_VAL	Output	INT	I, Q, M, D, L	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.
BUSY	Output	BOOL	I, Q, M, D, L	Information on whether compression operation is active
DONE	Output	BOOL	I, Q, M, D, L	Information on whether compression operation is complete

For more information on valid data types, refer to "See also".

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error The status of the compression operation can be queried on the BUSY and DONE parameters.
8091	An externally started compression operation is already being executed.
8092	The instruction cannot be executed, because one of the following conditions applies: • The "Delete block" function is active. • A programming device function is accessing one of the blocks to be moved. • An externally started compression operation is already being executed. • The H-CPU is running a link-up or update function.
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For more information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Setting status bits in program control operation instructions (Page 2460)

Switching display formats in the program status (Page 2980)

RE_TRIGR: Restart cycle monitoring time

Description

You can use the "Restart cycle monitoring time" instruction to restart the cycle monitoring of the CPU. The cycle time monitoring then restarts with the time you have set in the CPU configuration. By restarting the cycle monitoring time, you can prevent errors being triggered or the CPU changing to STOP.

The "Restart cycle monitoring time" instruction can be used in blocks of priority class 1 (the cyclic OB) and in the blocks called within it.

If the instruction is called in a block with a higher priority, for example, a hardware interrupt, diagnostic interrupt or cyclic interrupt, the instruction is not be executed.

Note

Make sure that you do not create an infinite loop in the cyclical program processing, i.e. in OB1, when you use the instruction "Restart cycle monitoring time". Otherwise the CPU will not reach the cycle control point. As a result, it may not be possible to execute certain CPU functions (e.g. process image update).

Syntax

The following syntax is used for the "Restart cycle monitoring time" instruction:

```
CALL RE_TRIGR
```

Parameters

The "Restart cycle monitoring time" instruction has no parameters and returns no error information.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits in program control operation instructions (Page 2460)

STP: Exit program

Description

You use the "Exit program" instruction to set the CPU to STOP mode and therefore to terminate the execution of the program. The effects of changing from RUN to STOP depend on the CPU configuration.

Syntax

The following syntax is used for the "Exit program" instruction:

```
CALL STP
```

Parameters

The "Exit program" instruction has no parameters and returns no error information.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits in program control operation instructions (Page 2460)

WAIT: Configure time delay

Description

The "Configure time delay" instruction is used to halt the execution of the program for a set period of time. You indicate the period of time in microseconds on the WT parameter of the instruction.

You can configure time delays of up to 32767 microseconds (µs). The smallest possible delay time depends on the particular CPU and corresponds to the execution time of the "Configure time delay" instruction.

The execution of the instruction can be interrupted by higher priority events.

The "Configure time delay" instruction returns no error information.

Syntax

The following syntax is used for the "Configure time delay" instruction:

```
CALL WAIT
WT := <Operand>
```

Parameters

The following table shows the parameters of the "Configure time delay" instruction:

Parameters	Declaration	Data type	Memory area	Description
WT	Input	INT	I, Q, M, D, L, P or constant	Time delay in microseconds (µs)

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits in program control operation instructions (Page 2460)

PROTECT: Change protection level

Description

The instruction "Change protection level" is used to change the configured protection level "1" of the CPU.

The configured protection level "1" with selected option "Removable with password", means that you have to configure a password. This password enables you to write-access during operation the CPU which was read-only protected with the instruction "Change Protection Level" (e. g. Load).

You specify the protection level on the MODE parameter of the instruction. The current protection level of your CPU is displayed in the inspector window under "Properties > Protection".

The following table shows the meaning of the protection levels.

Protection level	Meaning
1	All programming device functions are permitted. Any existing blocking of the password authentication is canceled by the instruction "Change protection level".
2	• The program and the configuration of the CPU cannot be changed. The program in the CPU can be read. • The functions for process control, process monitoring, and process communication are permitted. • All informational functions are permitted. • If you know the valid password you can cancel the set write protection. • Any existing blocking of the password authentication is canceled by the instruction "Change protection level".
3	• You cannot cancel the set read and write protection, even if you know the valid password. • If a valid connection exists at the time when the instruction "Change protection level" is called, the call of the instruction does not affect this connection.

If you have configured protection level "2" or "3" for your CPU, calling the "Change protection level" instruction has no effect.

You can use the SSL parts list with the SSL ID W#16#0232 and index W#16#0004 to check whether protection level "2" or "3" is still in effect after the execution of the "Change protection level" instruction.

Note

You can only use the "Change protection level" instruction to set a protection level that is equal to or higher than the protection level you have configured with STEP 7 "Configure hardware. You have, for example, configured protection level 3 and then called the instruction "Change protection level" with the MODE parameter = 12. This blocks the password authentication. If the instruction is called again with the parameter MODE = 0 or 1, protection level 3 is retained and the read and write protection canceled by entry of the valid password.

Effects of the operating state transitions on the protection level

The following table shows what effects the operating state transitions have on the protection level set with the "Change protection level" instruction.

Action	Effects on the protection level
• Operating mode switch to STOP • Unbuffered POWER ON • Memory reset with operating mode switch • Configuration and/or programming changes in STOP mode of the S7-400 CPU	The protection level is set to "1". All programming device functions are permitted.
• Cold restart • Warm restart	The protection level is set by the operating system to "1". All programming device functions are permitted. If required, you can set the protection level in your program to "2" or "3" by calling the "Change protection level" instruction.
Buffered POWER ON	Remains unchanged

Action	Effects on the protection level
Operating mode transition RUN/STARTUP/HOLD -> STOP (by calling the "Exit program" instruction (STP) due to program error without corresponding error OB, or by operator action)	Remains unchanged
S7-400: Restart by user action or automatically following POWER ON	Remains unchanged

Syntax

The following syntax is used for the "Change protection level" instruction:

```
CALL PROTECT
MODE := <Operand>
RET_VAL := <Operand>
```

Parameters

The following table shows the parameters of the "Change protection level" instruction:

Parameters	Declaration	Data type	Memory area	Description
MODE	Input	WORD	I, Q, M, D, L or constant	Protection level Possible values: • W#16#0000: Setting protection level "1" • W#16#0001: Setting protection level "2" with password authentication • W#16#000C: Setting protection level "3" without password authentication Constants can also be specified at the MODE parameter.
RET_VAL	Output	INT	I, Q, M, D, L	If an error occurs during processing, information on the error is output.

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	Job executed without error
8090	Invalid value at MODE parameter.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For more information on toggling display formats, refer to "See also".	

Example

The CPU should always be protected against access in RUN mode. The protection should only be rescinded when the operating mode switch is at STOP. You can reach this functionality by calling the "Change protection level" instruction in a startup OB and executing the instruction with the value W#16#0001 (protection level "2") on the MODE parameter. This way, protection level "2" remains set even after a warm restart.

STL	Explanation
CALL PROTECT	// The "Change protection level" instruction is called.
MODE := "Tag_ProtectionLevel"	// Set protection level
RET_VAL := "Tag_ErrorCode"	// Error information

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits in program control operation instructions (Page 2460)

Switching display formats in the program status (Page 2980)

CiR: Control CiR process

Description

The "Control CiR process" instruction is used to influence the CiR process in the user program as follows:

- Disable CiR process. In this case, the download of a modified configuration from the programming device to the CPU is always rejected. The disable is maintained until you cancel it again with the "Control CiR process" instruction.
- You can disable the CiR process conditionally by specifying a high limit for the CiR synchronization time on the FRZ_TIME parameter. In this case, the download of a modified configuration from the programming device to the CPU is only accepted if the CiR synchronization time calculated in the CPU does not exceed the specified high limit.
- Enable CiR process. The high limit of the CiR synchronization time is set to the default value 1000 ms.
- Determine whether the CiR process is enabled or not. If the CiR process is enabled or conditionally enabled, you will get the current high limit for the CiR synchronization time in the A_FT parameter of the instruction.

Note

During the CiR synchronization period the outputs are frozen and the inputs are not evaluated.

Syntax

The following syntax is used for the "Control CiR process" instruction:

```
CALL CIR
MODE := <Operand>
FRZ_TIME := <Operand>
RET_VAL := <Operand>
A_FT := <Operand>
```

Parameters

The following table shows the parameters of the "Control CiR process" instruction:

Parameters	Declaration	Data type	Memory area	Description
MODE	Input	BYTE	I, Q, M, D, L or constant	Job identifier Valid values on the MODE parameter are: • 0: Information function • 1: Enable CiR process • 2: Disable CiR process • 3: Disable CiR conditionally. You specify the high limit of the CiR synchronization time in FRZ_TIME.
FRZ_TIME	Input	TIME	I, Q, M, D, L or constant	High limit of the CiR synchronization time in milliseconds. Permitted range of values: 60 ... 2500 ms (default: 1000 ms) The FRZ_TIME parameter is only relevant when the value 3 is set on the MODE parameter.
RET_VAL	Output	INT	I, Q, M, D, L	Error information When the MODE parameter is "0", the RET_VAL parameter supplies information on the enabling of the CiR process.
A_FT	Output	TIME	I, Q, M, D, L	Currently valid high limit of the CiR synchronization time

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	Job executed without error This error code can be output only if the value "1", "2" or "3" is set on the MODE parameter.
0001	CiR is enabled. This error code can be output only if the value "0" is set on the MODE parameter.
0002	The CiR process is disabled. This error code can be output only if the value "0" is set on the MODE parameter.
0003	CiR is disabled conditionally. This error code can be output only if the value "0" is set on the MODE parameter.
8001	The CPU is not ready for a CiR operation. You are using an H CPU in an H-system (stand-alone), or you are working with a standard CPU in multicomputing mode.
8002	The value at the MODE parameter is invalid.
8003	The value at the FRZ_TIME parameter is invalid.
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For more information on toggling display formats, refer to "See also".	

Example

The "Control CiR process" instruction can be used, for example, to ensure that no CiR process is initiated during periods in which maximum CPU performance is required to control the process.

To use the "Control CiR process" for this, the instruction must be called in the program before the start of the period with increased process activity and executed with the value "2" (disable CiR process) for the MODE parameter. The CiR process is then disabled.

To enable the CiR process again after the completion of the period with increased process activity, call the "Control CiR process" instruction again and execute it with the value "1" for the MODE parameter. Alternatively, you can also set the value 3 on the MODE parameter to disable the CiR process conditionally.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Setting status bits in program control operation instructions (Page 2460)

Switching display formats in the program status (Page 2980)

Word logic operations

DECO: Decode

Description

You can use the "Decode" instruction to set a bit in the output value specified by the input value.

The "Decode" instruction reads the value of the IN parameter and sets the bit in the OUT parameter whose bit position corresponds to the value read. The other bits in the output value are filled with zeroes. When the value at the IN parameter is greater than 31, a modulo-32 operation is executed.

Syntax

The following syntax is used for the "Decode" instruction:

```
CALL DECO
<Data type> DWORD
IN := <Operand>
OUT := <Operand>
```

Parameter

The following table shows the parameters of the "Decode" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	WORD, INT	I, Q, M, D, L, P or constant	Position of the bit in the output value which is set
OUT	Output	DWORD	I, Q, M, D, L, P	Output value

You can select the data type of the instruction from the "<???>" drop-down list.

For more information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
CALL DECO	// The "Decode" instruction is called
WORD DWORD	// Data types of the instruction
IN := "Tag_Input"	// Input value that specifies the bit to be set in the output value

STL	Explanation
OUT := "Tag_Output"	// Result

The following table shows how the instruction functions using specific values:

Tag_Input	3	
	31 16	15 ... 3 ... 0
Tag_Output	0000 0000 0000 0000	0000 0000 0000 1000

The instruction reads bit number "3" from the value of the operand "Tag_Input" and sets the third bit to the value of the operand "Tag_Output".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bit for word logic operations (Page 2462)

ENCO: Encode

Description

The instruction "Encode" is used to read the bit number of the least significant set bit in the input value and to output it at the parameter OUT.

The "Encode" instruction selects the least significant bit of the value at the IN parameter and writes this bit number to the operand at the OUT parameter. If the IN parameter has the value DW#16#00000001 or DW#16#00000000, the value "0" is output at the OUT parameter.

Syntax

The following syntax is used for the "Encode" instruction:

```
CALL ENCO
DWORD
IN := <Operand>
OUT := <Operand>
```

Parameters

The following table shows the parameters of the "Encode" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	DWORD	I, Q, M, D, L, P or constant	Input value
OUT	Output	INT	I, Q, M, D, L, P	Output value

For more information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL ENCO                    // The "Encode" instruction is called
DWORD                        // Data type of the instruction
IN := "Tag_Input"            // Input value the least significant bit of which is
                             read.
OUT := "Tag_Output"          // Bit number of least significant bit set in the
                             input value
```

The following table shows how the instruction functions using specific values:

	31 16	15 ... 3 ... 0
"Tag_Input"	0000 1111 0000 0101	0000 1001 0000 1000
"Tag_Output"	3	

The instruction selects the least significant set bit of the variable "Tag_Input" and writes its bit position "3" to the variable "Tag_Output".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bit for word logic operations (Page 2462)

SEL: Select

Description

Depending on a switch (G input), the "Select" instruction selects one of the inputs, IN0 or IN1 and copies its content to the OUT output. When the G input has the signal state "0", the value at the IN0 input is moved. If input G has the signal state "1", the value at input IN1 is copied to output OUT.

The instruction is only executed if the tags of all parameters are of the same data type.

Syntax

The following syntax is used for the "Select" instruction:

```
CALL SEL
<Data type>
G := <Operand>
IN0 := <Operand>
IN1 := <Operand>
OUT := <Operand>
```

Parameters

The following table shows the parameters of the "Select" instruction:

Parameter	Declaration	Data type	Memory area	Description
G	Input	BOOL	I, Q, M, D, L	Switch
IN0	Input	Bit strings, integers, floating-point numbers, timers, CHAR, TOD, DATE, DT	I, Q, M, D, L, P	First input value
IN1	Input	Bit strings, integers, floating-point numbers, timers, CHAR, TOD, DATE, DT	I, Q, M, D, L, P	Second input value
OUT	Output	Bit strings, integers, floating-point numbers, timers, CHAR, TOD, DATE, DT	I, Q, M, D, L, P	Result

You can select the data type of the instruction from the "<data type>" drop-down list.

For more information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL SEL                     // The "Select" instruction is called
INT                          // Data type of the instruction
G := "Tag_Input_G"           // Signal state of the switch
```

```
STL                          Explanation
IN0 := "Tag_Input0"          // First input value
IN1 := "Tag_Input1"          // Second input value
OUT := "Tag_Output"          // Value of the selected input
```

The following table shows how the instruction functions using specific values:

Parameters	Operand	Value
G	Tag_Input_G	1
IN0	Tag_Input0	W#16#0000
IN1	Tag_Input1	W#16#FFFF
OUT	Tag_Output	W#16#FFFF

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bit for word logic operations (Page 2462)

Additional instructions

SET: Set bit array

Description

You use the "Set bit array" instruction to set the signal state of the bits in a specified area to "1". You use the pointer on the S_BIT parameter to define the starting point of the range. If the pointer points to the memory area of the external I/O, the instruction is not executed. You specify the number of bits to be set in the specified area in the N parameter. When the N parameter has the value "0", calling the instruction has no effect.

The instruction is only executed when the MCR bit is "1". When the MCR bit has the signal state "0", the bits in the specified range are not changed.

Syntax

The following syntax is used for the "Set bit array" instruction:

```
CALL SET
S_BIT := <Operand>
N := <Operand>
```

Parameters

The following table shows the parameters of the "Set bit array" instruction:

Parameters	Declaration	Data type	Memory area	Description
S_BIT	Input	POINTER	I, Q, M, D	Pointer to the first bit of the range whose bits are set. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	I, Q, M, D, L, P or constant	Number of bits to be set

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL SET                     // The "Set bit array" instruction is called
S_BIT := P#M0.0              // Pointer to the first bit to be set (M0.0)
N := 10                      // Number of bits to be set
```

In the example the instruction sets 10 bits in the range from M0.0 to M1.2.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Master control relay (Page 2471)

Important notes on using MCR functions (Page 2472)

Setting status bits (Page 2455)

SETP: Set a bit array in the I/O area

Description

The instruction "Set a bit array in the I/O area" sets the bits of an I/O area to the signal state "1". At the same time the corresponding bits in the process image outputs are also set parallel to the I/O bits.

You use the N and SA parameters to define the bit array that will be set. The bits of the array must be located in the part of the I/O area to which a process image is assigned.

If no I/O devices are plugged in for part of the selected bit array, the instruction attempts to set the entire bit array. In this case an error information is output at the RET_VAL parameter.

The I/O area is set byte-by-byte. If the bit array defined via the N and SA parameters does not begin or end at a byte boundary, calling the instruction has the following effect:

- The bits that are in the first and last bytes to be transferred and that do not belong to the defined bit array, get the value of the corresponding bits in the process image output.

NOTICE
Setting bits The setting of the bits at the value of the relevant bit in the process image output can lead, for example, to lead to unintended reactions such as starting a motor or turning off a cooling system.

- The bits belonging to the defined bit array are set to the signal state "1".

The instruction "Set a bit array in the I/O area" must be called in an MCR area.

Calling the "Set a bit array in the I/O area" instruction has no effect if one of the following conditions applies:

- The N parameter has the value "0".
- The MCR bit has the signal state "0".

Syntax

The following syntax is used for the "Set a bit array in the I/O area" instruction:

```
CALL SETP
N := <Operand>
RET_VAL := <Operand>
SA := <Operand>
```

Parameters

The following table shows the parameters of the "Set a bit array in the I/O area" instruction:

Parameters	Declaration	Data type	Memory area	Description
N	Input	INT	I, Q, M, D, L or constant	Number of bits to be set
RET_VAL	Output	INT	I, Q, M, D, L	Error information
SA	Output	POINTER	P	Pointer to the first bit to be set

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For more information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL SETP                    // The "Set a bit array in the I/O area" instruction
                             is called.
N := 8                       // Number of bits to be set
RET_VAL := "Tag_ErrorCode"   // Error information
SA := P#10.0                 // Pointer to the first bit to be set (Q10.0)
```

In the example the instruction sets 8 bits in the I/O range corresponding to the outputs Q10.0 to Q10.7.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits (Page 2455)

Switching display formats in the program status (Page 2980)

SETI: Set byte array

Description

The "Set byte array" instruction sets the signal state of the bits in a specified area of bytes to "1". You define the start point of the array on the S_BYTE. You define the size of the array by specifying the number of bits to be set on the N parameter of the instruction.

Note

The value of the N parameter must be a multiple of the number eight (for example, 8, 16, 24, etc.).

The pointer specified on the S_BYTE parameter must reference the external I/O memory area (P). Since the "P" memory area is accessed as bytes, words, or double words, the pointer of the S_BYTE parameter must point to an operand that corresponds to a byte limit value, i.e., the bit number of the pointer must be "0".

Note

The signal state of the corresponding bits in the process-image output (Q) is reset to "0".

Calling the "Set byte array" instruction has no effect if one of the following conditions applies:

- The pointer on the S_BYTE parameter does not reference the external I/O memory area (P).
- The pointer on the S_BYTE parameter references an operand that is not byte-aligned.
- The value of the N parameter is not a multiple of the number 8.
- The MCR bit has the signal state "0".

Syntax

The following syntax is used for the "Set byte array" instruction:

```
CALL SETI
S_BYTE := <Operand>
N := <Operand>
```

Parameters

The following table shows the parameters of the "Set byte array" instruction:

Parameters	Declaration	Data type	Memory area	Description
S_BYTE	Input	POINTER	P or constant	Points to the first byte of the array whose bits will be set For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	I, Q, M, D, L, P or constant	Size of the array to be set The array is specified by the number of bits as a multiple of eight (for example 8, 16, 24 etc.).

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL SETI                    // The "Set byte array" instruction is called.
S_BYTE := P#2.0              // Pointer to the first byte of the range (P2.0)
N := 16                      // Number of bits to be set
```

In the example the instruction sets 2 bytes (16 bits) in the range from P2.0 to P3.7 (I/O outputs).

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Master control relay (Page 2471)

Important notes on using MCR functions (Page 2472)

Setting status bits (Page 2455)

RESET: Reset bit array

Description

The "Reset bit array" instruction is used to reset the signal state of the bits in a specified range to "0". You use the pointer on the S_BIT parameter to define the starting point of the range. If the pointer points to the memory area of the external I/O, the instruction is not executed. You specify the number of bits to be reset in the specified range at the N parameter of the instruction. When the N parameter has the value "0", calling the instruction has no effect.

The instruction is only executed when the MCR bit is "1". When the MCR bit has the signal state "0", the bits in the specified range are not changed.

Syntax

The following syntax is used for the "Reset bit array" instruction:

```
CALL RESET
S_BIT := <Operand>
N := <Operand>
```

Parameters

The following table shows the parameters of the "Reset bit array" instruction:

Parameters	Declaration	Data type	Memory area	Description
S_BIT	Input	POINTER	I, Q, M, D	Points to the first bit of the range whose bits will be reset. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	I, Q, M, D, L, P or constant	Number of bits to be reset.

Example

The following example shows how the instruction works:

STL	Explanation
CALL RESET	// The "Reset bit array" instruction is called.
S_BIT := P#M0.0	// Pointer to the first bit (M0.0) to be reset

STL	Explanation
N := 10	// Number of bits which are reset

In the example the instruction resets 10 bits in the range from M0.0 to M1.2.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Master control relay (Page 2471)

Important notes on using MCR functions (Page 2472)

Setting status bits (Page 2455)

RESETP: Reset a bit array in the I/O area

Description

The "Reset a bit array in the I/O area" instruction is used to reset the bits of an I/O area to signal state "0". At the same time the corresponding bits in the process image outputs are also reset parallel to the I/O bits.

You use the N and SA parameters to define the bit array that will be reset. The bits of the array must be located in the part of the I/O area to which a process image is assigned.

If no I/O devices are plugged in for part of the selected bit array, the instruction attempts to set the entire bit array. In this case an error information is output at the RET_VAL parameter.

The I/O area is reset byte-by-byte. If the bit array defined via the N and SA parameters does not begin or end at a byte boundary, calling the instruction has the following effect:

- The bits that are in the first and last bytes to be transferred and that do not belong to the defined bit array, get the value of the corresponding bits in the process image output.

NOTICE
Setting bits The setting of the bits at the value of the relevant bit in the process image output can lead, for example, to lead to unintended reactions such as starting a motor or turning off a cooling system.

- The bits belonging to the defined bit array are reset to the signal state "0".

The instruction "Reset a bit array in the I/O area" must be called in an MCR range.

Calling the "Reset a bit array in the I/O area" instruction has no effect if one of the following conditions applies:

- The N parameter has the value "0".
- The MCR bit has the signal state "0".

Syntax

The following syntax is used for the "Reset a bit array in the I/O area" instruction:

```
CALL RESETP
N := <Operand>
RET_VAL := <Operand>
SA := <Operand>
```

Parameters

The following table shows the parameters of the "Reset a bit array in the I/O area" instruction:

Parameters	Declaration	Data type	Memory area	Description
N	Input	INT	I, Q, M, D, L or constant	Number of bits to be reset.
RET_VAL	Output	INT	I, Q, M, D, L	Error information
SA	Output	POINTER	P	Pointer to the first bit to be reset.

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For more information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

STL	Explanation
CALL RESETP	// The "Reset a bit array in the I/O area" instruction is called.
N := 8	// Number of bits to be reset
RET_VAL := "Tag_ErrorCode"	// Error information
SA := P#10.0	// Pointer to the first bit (Q10.0) to be reset

In the example the instruction sets 8 bits in the I/O area corresponding to the outputs Q10.0 to Q10.7.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits (Page 2455)

Switching display formats in the program status (Page 2980)

RESETI: Reset byte array

Description

The "Reset byte array" instruction is used to reset the signal state of the bits in a specified range of bytes to "0". You use the pointer on the S_BYTE parameter to define the starting point of the range. You define the size of the range by specifying the number of bits to be reset on the N parameter of the instruction.

Note

The value of the N parameter must be a multiple of the number eight (for example, 8, 16, 24, etc.).

The pointer specified on the S_BYTE parameter must reference the external I/O memory area (P). Since the "P" memory area is accessed as bytes, words, or double words, the pointer of the S_BYTE parameter must point to an operand that corresponds to a byte limit value, i.e., the bit number of the pointer must be "0".

Note

The signal state of the corresponding bits in the process-image output (Q) is reset to "0".

Calling the "Reset byte array" instruction has no effect if one of the following conditions applies:

- The pointer on the S_BYTE parameter does not reference the external I/O memory area (P).
- The pointer on the S_BYTE parameter references an operand that is not byte-aligned.
- The value of the N parameter is not a multiple of the number 8.
- The MCR bit has the signal state "0".

Syntax

The following syntax is used for the "Reset byte array" instruction:

```
CALL RESETI
S_BYTE := <Operand>
```

```
N := <Operand>
```

Parameters

The following table shows the parameters of the "Reset byte array" instruction:

Parameters	Declaration	Data type	Memory area	Description
S_BYTE	Input	POINTER	P or constant	Points to the first byte of the range whose bits will be reset. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	I, Q, M, D, L, P or constant	Size of the array whose bits are to be reset. The array is specified by the number of bits as a multiple of eight (for example 8, 16, 24 etc.).

Example

The following example shows how the instruction works:

STL	Explanation
CALL RESETI	// The "Reset byte array" instruction is called.
S_BYTE := P#2.0	// Pointer to the first byte of the range (P2.0)
N := 16	// Number of bits to be reset

In the example the instruction resets 2 Bytes (16 bits) in the range from P2.0 to P3.7 (I/O outputs).

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Master control relay (Page 2471)

Important notes on using MCR functions (Page 2472)

Setting status bits (Page 2455)

REPL_VAL: Enter substitute value

Description

Use the "Enter substitute value" instruction to enter a substitute value in accumulator 1 (ACCU 1) from a synchronous error organization block.

Note

You can only call the "Enter substitute value" instruction in a synchronous error OB (OB 121, OB 122).

Use the "Enter substitute value" instruction when you can no longer read any value from a module. The OB 122 is started after each access to this module. By calling the "Enter substitute value" instruction you can have the substitute value loaded in accumulator 1. In this case the program will then be processed with this substitute value. You can take the required information for the selection of the substitute value from the local variable of the OB 122.

Syntax

The following syntax is used for the "Enter substitute value" instruction:

```
CALL REPL_VAL
VAL := <Operand>
RET_VAL := <Operand>
```

Parameters

The following table shows the parameters of the "Enter substitute value" instruction:

Parameters	Declaration	Data type	Memory area	Description
VAL	Input	DWORD	I, Q, M, D, L or constant	Substitute value
RET_VAL	Output	INT	I, Q, M, D, L	Error information

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Description
0000	No error
8080	The instruction was not called from a synchronous error OB.
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For more information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Evaluating errors with output parameter RET_VAL (Page 5370)

Setting status bits (Page 2455)

Switching display formats in the program status (Page 2980)

DRUM: Implement sequencer

Description

The "Implement sequencer" instruction is used to set the programmed output bits (OUT1 to OUT16) and the output word (OUT_WORD) with the programmed values of the OUT_VAL parameter of the relevant step. The specific step must thereby satisfy the conditions of the programmed enable mask on the S_MASK parameter while the instruction remains at this step. The instruction advances to the next step if the event for the step is true and the programmed time for the current step elapses, or if the value at the JOG parameter changes from "0" to "1". The instruction is reset if the signal state on the RESET parameter changes to "1". The current step is hereby equated to the preset step (DSP).

The amount of time spent on a step is determined by the product of the preset timebase (DTBP) and the preset count value (S_PRESET) for each step. At the start of a new step, this calculated value is loaded into the DCC parameter, which contains the time remaining for the current step. If, for example the value at the DTBP parameter is "2" and the preset value for the first step is "100" (100 ms), the DCC parameter has the value "200" (200 ms).

A step can be programmed with a time value, an event, or both. Steps that have an event bit and the time value "0" advance to the next step as soon as the signal state of the event bit is "1". Steps that are programmed only with a time value start the time immediately. Steps that are programmed with an event bit and a time value greater than "0" start the time when the signal state of the event bit is "1". The event bits are initialized with a signal state of "1".

When the sequencer is on the last programmed step (LST_STEP) and the time for this step has expired, the signal state on the Q parameter is set to "1"; otherwise it is set to "0". When the parameter Q is set, the instruction remains on the step until it is reset.

In the configurable mask (S_MASK) you can selected the separate bits in the output word (OUT_WORD) and set or reset the output bits (OUT1 to OUT16) by means of the output values (OUT_VAL). If a bit of the configurable mask is in the signal state "1", the value OUT_VAL sets or resets the corresponding bit. If the signal state of a bit of the configurable mask is "0", the corresponding bit is left unchanged. All the bits of the configurable mask for all 16 steps are initialized with a signal state of "1".

The output bit on the OUT1 parameter corresponds to the least significant bit of the output word (OUT_WORD). The output bit on the OUT16 parameter corresponds to the most significant bit of the output word (OUT_WORD).

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For more information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Implement sequencer" instruction:

```
CALL DRUM, "Instance DB"
RESET := <Operand>
JOG := <Operand>
DRUM_EN := <Operand>
LST_STEP := <Operand>
EVENT1 - 16 := <Operand>
OUT1 - 16 := <Operand>
Q := <Operand>
OUT_WORD := <Operand>
ERR_CODE := <Operand>
```

Parameters

The following table shows the parameters of the "Implement sequencer" instruction:

Parameters	Declaration	Data type	Memory area	Description
RESET	Input	BOOL	I, Q, M, D, L	A signal state of "1" indicates a reset condition
JOG	Input	BOOL	I, Q, M, D, L	When the signal state changes from "0" to "1", the instruction advances to the next step.
DRUM_EN	Input	BOOL	I, Q, M, D, L	A signal state of "1" allows the sequencer to advance based on the event and time criteria.
LST_STEP	Input	BYTE	I, Q, M, D, L or constant	Step number of the last step programmed.
EVENT(i), 1<= i <=16	Input	BOOL	I, Q, M, D, L	Event bit (i); Initial signal state is "1".
OUT(j), 1 ≤ j ≤ 16	Output	BOOL	I, Q, M, D, L	Output bit (j)
Q	Output	BOOL	I, Q, M, D, L	A signal state of "1" indicates that the time for the last step has elapsed.
OUT_WORD	Output	WORD	I, Q, M, D, L, P	Word address to which the sequencer writes the output values.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
JOG_HIS	Static	BOOL	I, Q, M, D, L	JOG parameter history bit
EOD	Static	BOOL	I, Q, M, D, L	A signal state of "1" indicates that the time for the last step has elapsed.
DSP	Static	BYTE	I, Q, M, D, L, P	Preset first step of the sequencer (1 to 16)
DSC	Static	BYTE	I, Q, M, D, L, P	Current step of the sequencer

Parameters	Declaration	Data type	Memory area	Description
DCC	Static	DWORD	I, Q, M, D, L, P	Remaining processing time for the current step
DTBP	Static	WORD	I, Q, M, D, L, P	Preset timebase of the sequencer
PREV_TIME	Static	DWORD	I, Q, M, D, L or constant	System time of the previous call
S_PRESET	Static	ARRAY of WORD	I, Q, M, D, L	Count preset for each step [1 to 16] where 1 count = 1 ms.
OUT_VAL	Static	ARRAY of BOOL	I, Q, M, D, L	Output values for each step [1 to 16, 0 to 15].
S_MASK	Static	ARRAY of BOOL	I, Q, M, D, L	Configurable mask for each step [1 to 16, 0 to 15]. Initial signal states are "1".

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

ERR_CODE*	Explanation
W#16#0000	No error
W#16#000B	The value at the LST_STEP parameter is less than 1 or greater than 16.
W#16#000C	The value at the DSC parameter is less than 1 or greater than the value at the LST_STEP parameter.
W#16#000D	The value at the DSP parameter is less than 1 or greater than the value at the LST_STEP parameter.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For more information on toggling display formats, refer to "See also".	

Example

In the following example the instruction advances from step 1 to step 2. The output bit (OUT1 to OUT16) and the output word (OUT_WORD) are set according to the mask configured for step 2 and the values of the OUT_VAL parameter.

Note

You can initialize static parameters in the data block.

STL	Explanation
CALL DRUM, "DRUM_DB"	// The "Implement sequencer" instruction is called and the instance data block "DRUM_DB" is created.
RESET := "Tag_Reset"	// Reset input
JOG := "Tag_Input_Jog"	// On a rising edge, the instruction advances to the next step
DRUM_EN := "Tag_Input_DrumEN"	// A signal state of "1" allows the sequencer to advance based on the event and time criteria.
LST_STEP := "Tag_Number_LastStep"	// Step number of the last step programmed
EVENT1 := "MyTag_Event_1"	// Event bit 1
EVENT2 := "MyTag_Event_2"	// Event bit 2
EVENT3 := "MyTag_Event_3"	// Event bit 3
EVENT4 := "MyTag_Event_4"	// Event bit 4
EVENT5 := "MyTag_Event_5"	// Event bit 5
EVENT6 := "MyTag_Event_6"	// Event bit 6
EVENT7 := "MyTag_Event_7"	// Event bit 7
EVENT8 := "MyTag_Event_8"	// Event bit 8
EVENT9 := "MyTag_Event_9"	// Event bit 9
EVENT10 := "MyTag_Event_10"	// Event bit 10
EVENT11 := "MyTag_Event_11"	// Event bit 11
EVENT12 := "MyTag_Event_12"	// Event bit 12
EVENT13 := "MyTag_Event_13"	// Event bit 13
EVENT14 := "MyTag_Event_14"	// Event bit 14
EVENT15 := "MyTag_Event_15"	// Event bit 15
EVENT16 := "MyTag_Event_16"	// Event bit 16
OUT1 := "MyTag_Output_1"	// Output bit 1
OUT2 := "MyTag_Output_2"	// Output bit 2
OUT3 := "MyTag_Output_3"	// Output bit 3
OUT4 := "MyTag_Output_4"	// Output bit 4
OUT5 := "MyTag_Output_5"	// Output bit 5
OUT6 := "MyTag_Output_6"	// Output bit 6
OUT7 := "MyTag_Output_7"	// Output bit 7
OUT8 := "MyTag_Output_8"	// Output bit 8
OUT9 := "MyTag_Output_9"	// Output bit 9
OUT10 := "MyTag_Output_10"	// Output bit 10
OUT11 := "MyTag_Output_11"	// Output bit 11
OUT12 := "MyTag_Output_12"	// Output bit 12
OUT13 := "MyTag_Output_13"	// Output bit 13
OUT14 := "MyTag_Output_14"	// Output bit 14
OUT15 := "MyTag_Output_15"	// Output bit 15
OUT16 := "MyTag_Output_16"	// Output bit 16
Q := "Tag_Output_Q"	// A signal state of "1" indicates that the time for the last step has elapsed
OUT_WORD := "Tag_OutputWord"	// Word address to which the sequencer writes the output values

```
STL                              Explanation
ERR_CODE := "Tag_ErrorCode"    // Error information
```

The following tables show how the instruction works using specific values.

Before processing

In this example the following values are used for the input parameters:

Parameters	Operand	Address	Value
RESET	Tag_Reset	M0.0	FALSE
JOG	Tag_Input_JOG	M0.1	FALSE
DRUM_EN	Tag_Input_DrumEN	M0.2	TRUE
LST_STEP	Tag_Number_LastStep	MB1	B#16#08
EVENT2	MyTag_Event_2	M20.0	FALSE
EVENT4	MyTag_Event_4	M20.1	FALSE
EVENT6	MyTag_Event_6	M20.2	FALSE
EVENT8	MyTag_Event_8	M20.3	FALSE
EVENT10	MyTag_Event_10	M20.4	FALSE
EVENT12	MyTag_Event_12	M20.5	FALSE
EVENT14	MyTag_Event_14	M20.6	FALSE
EVENT16	MyTag_Event_16	M20.7	FALSE

The following values are saved in the instance data block "DRUM_DB" of the instruction:

Parameters	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSP	DBB13	W#16#0001
DSC	DBB14	W#16#0001
DCC	DBD16	DW#16#0000000A
DTBP	DBW20	W#16#0001
S_PRESET[1]	DBW26	W#16#0064
S_PRESET[2]	DBW28	W#16#00C8
OUT_VAL[1,0]	DBX58.0	TRUE
OUT_VAL[1,1]	DBX58.1	TRUE
OUT_VAL[1,2]	DBX58.2	TRUE
OUT_VAL[1,3]	DBX58.3	TRUE
OUT_VAL[1,4]	DBX58.4	TRUE
OUT_VAL[1,5]	DBX58.5	TRUE
OUT_VAL[1,6]	DBX58.6	TRUE
OUT_VAL[1,7]	DBX58.7	TRUE
OUT_VAL[1,8]	DBX59.0	TRUE
OUT_VAL[1,9]	DBX59.1	TRUE
OUT_VAL[1,10]	DBX59.2	TRUE

Parameters	Address	Value
OUT_VAL[1,11]	DBX59.3	TRUE
OUT_VAL[1,12]	DBX59.4	TRUE
OUT_VAL[1,13]	DBX59.5	TRUE
OUT_VAL[1,14]	DBX59.6	TRUE
OUT_VAL[1,15]	DBX59.7	TRUE
OUT_VAL[2,0]	DBX60.0	FALSE
OUT_VAL[2,1]	DBX60.1	FALSE
OUT_VAL[2,2]	DBX60.2	FALSE
OUT_VAL[2,3]	DBX60.3	FALSE
OUT_VAL[2,4]	DBX60.4	FALSE
OUT_VAL[2,5]	DBX60.5	FALSE
OUT_VAL[2,6]	DBX60.6	FALSE
OUT_VAL[2,7]	DBX60.7	FALSE
OUT_VAL[2,8]	DBX61.0	FALSE
OUT_VAL[2,9]	DBX61.1	FALSE
OUT_VAL[2,10]	DBX61.2	FALSE
OUT_VAL[2,11]	DBX61.3	FALSE
OUT_VAL[2,12]	DBX61.4	FALSE
OUT_VAL[2,13]	DBX61.5	FALSE
OUT_VAL[2,14]	DBX61.6	FALSE
OUT_VAL[2,15]	DBX61.7	FALSE
S_MASK[2,0]	DBX92.0	FALSE
S_MASK[2,1]	DBX92.1	TRUE
S_MASK[2,2]	DBX92.2	TRUE
S_MASK[2,3]	DBX92.3	TRUE
S_MASK[2,4]	DBX92.4	TRUE
S_MASK[2,5]	DBX92.5	FALSE
S_MASK[2,6]	DBX92.6	TRUE
S_MASK[2,7]	DBX92.7	TRUE
S_MASK[2,8]	DBX93.0	FALSE
S_MASK[2,9]	DBX93.1	FALSE
S_MASK[2,10]	DBX93.2	TRUE
S_MASK[2,11]	DBX93.3	TRUE
S_MASK[2,12]	DBX93.4	TRUE
S_MASK[2,13]	DBX93.5	TRUE
S_MASK[2,14]	DBX93.6	FALSE
S_MASK[2,15]	DBX93.7	TRUE

The output parameters are set to the following values before the execution of the instruction:

Parameters	Operand	Address	Value
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#FFFF

Parameters	Operand	Address	Value
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	TRUE
OUT3	MyTag_Output_3	M4.2	TRUE
OUT4	MyTag_Output_4	M4.3	TRUE
OUT5	MyTag_Output_5	M4.4	TRUE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	TRUE
OUT8	MyTag_Output_8	M4.7	TRUE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	TRUE
OUT12	MyTag_Output_12	M5.3	TRUE
OUT13	MyTag_Output_13	M5.4	TRUE
OUT14	MyTag_Output_14	M5.5	TRUE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	TRUE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameters	Operand	Address	Value
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	FALSE
OUT3	MyTag_Output_3	M4.2	FALSE
OUT4	MyTag_Output_4	M4.3	FALSE
OUT5	MyTag_Output_5	M4.4	FALSE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	FALSE
OUT8	MyTag_Output_8	M4.7	FALSE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	FALSE
OUT12	MyTag_Output_12	M5.3	FALSE
OUT13	MyTag_Output_13	M5.4	FALSE
OUT14	MyTag_Output_14	M5.5	FALSE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	FALSE
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#4321
ERR_CODE	Tag_ErrorCode	MW10	W#16#0000

The following values are changed in the instance data block "DRUM_DB" of the instruction after the execution of the instruction:

Parameters	Address	Value
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSC	DBB14	W#16#0002
DCC	DBD16	DW#16#000000C8

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits (Page 2455)

Switching display formats in the program status (Page 2980)

DRUM_X: Implement sequencer

Description

Use the "Implement sequencer" instruction to Implement a sequencer with a maximum of 16 steps. You specify the number of the first step on the DSP parameter. You specify the number of the last step on the LST_STEP parameter.

During each step, all 16 output bits OUT0 to OUT15 and the output parameter OUT_WORD (in which the output bits are collected together) are written. An output bit is assigned either the corresponding bit of the OUT_VAL array that you specified or the output bit of the preceding step. The value assigned depends on the value of the mask bit on the S_MASK parameter.

Note

The default setting for the mask bit is 0. If you want to change the current setting of one or more mask bits, make this change in the instance data block.

The "Implement sequencer" instruction advances to the next step when there is a rising signal edge on the JOG parameter. If the instruction is already in the last step, the Q und EOD tags are set when the JOG parameter has a positive signal edge. In this case the DCC parameter is set to the value "0". The instruction remains in the last step until the RESET parameter is set to the value "1".

You can also assign parameters so that switching to the next step is time dependent. To do this, the DRUM_EN parameter has to be set to the value "1". The transition to the next step is made when the event bit (EVENTi) belonging to the current step is set and the time specified

for the current step has expired. The time for a step is the product of the (DTBP) time base and the time factor valid for the current step (S_PRESET).

Note

The execution time (DCC) remaining in the current step is reduced only when the corresponding event bit (EVENTi) is set.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For more information on this topic, refer to "See also".

If the value "1" is set on the RESET parameter when the instruction is called, the sequencer goes to the step whose number you assigned on the DSP parameter.

A step can be programmed with a time value, an event, or both. Steps that have an event bit and the time value "0" advance to the next step as soon as the signal state of the event bit is "1". Steps that are programmed only with a time value start the time immediately. Steps that are programmed with an event bit and a time value greater than "0" start the time when the signal state of the event bit is "1".

You can also move the next step at any time via the value at the JOG parameter.

When the instruction is called for the first time, you must set the value "1" on the RESET parameter.

When the sequencer is in the last step and the execution time specified for this step has expired, the parameters Q and EOD are set. The instruction remains in the last step until the RESET parameter changes to the value "1".

The instruction is executed only in the STARTUP and RUN modes.

The operating system resets the "Implement sequencer" instruction during a cold restart. During a warm restart, the instruction is not reset. If you want the "Implement sequencer instruction" to be initialized after a warm restart, call OB 100 with the value "1" for the RESET parameter.

Syntax

The following syntax is used for the "Implement sequencer" instruction:

```
CALL DRUM_X, "Instance DB"
RESET := <Operand>
JOG := <Operand>
DRUM_EN := <Operand>
LST_STEP := <Operand>
EVENT1 - 16 := <Operand>
OUT0 - 15 := <Operand>
Q := <Operand>
OUT_WORD := <Operand>
```

```
ERR_CODE := <Operand>
```

Parameters

The following table shows the parameters of the "Implement sequencer" instruction:

Parameters	Declaration	Data type	Memory area	Description
RESET	Input	BOOL	I, Q, M, D, L	The signal state "1" resets the sequencer. When the instruction is called for the first time, you have to set the value "1" on the RESET parameter.
JOG	Input	BOOL	I, Q, M, D, L	A rising edge (compared to the last call) switches the sequencer to the next step if it is not yet in the last step. The next step is enabled depending on the value you assigned to the DRUM_EN parameter.
DRUM_EN	Input	BOOL	I, Q, M, D, L	Control parameter that specifies whether time-dependent switching to the next step is possible. If the value is "1" a time-dependent switching to the next step is possible.
LST_STEP	Input	BYTE	I, Q, M, D, L or constant	Number of the last step Values from 1 to 16 are permitted.
EVENT(i), 1 ≤ i ≤ 16	Input	BOOL	I, Q, M, D, L	Event bit (i) (belongs to step i)
OUTj, 0 ≤ j ≤ 15	Output	BOOL	I, Q, M, D, L	Output bit (j) (identical with bit no. j of OUT_WORD)
Q	Output	BOOL	I, Q, M, D, L	Status parameter that indicates whether the specified execution time of the last step has expired.
OUT_WORD	Output	WORD	I, Q, M, D, L, P	Output bits collected together in a tag
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
JOG_HIS	Static	BOOL	I, Q, M, D, L	Not relevant to user (input parameter JOG of the preceding call)
EOD	Static	BOOL	I, Q, M, D, L	Identical with output parameter Q

Parameters	Declaration	Data type	Memory area	Description
DSP	Static	BYTE	I, Q, M, D, L, P or constant	Number of the first step Values from 1 to 16 are permitted.
DSC	Static	BYTE	I, Q, M, D, L, P or constant	Number of the current step
DCC	Static	DWORD	I, Q, M, D, L, P	The execution time still remaining in the current step in milliseconds (ms) (only relevant if DRUM_EN = 1 and the corresponding event bit = 1)
DTBP	Static	WORD	I, Q, M, D, L, P or constant	The time base valid for all steps in milliseconds (ms)
PREV_TIME	Static	DWORD	I, Q, M, D, L or constant	(Not relevant for the user: system time of the previous call)
S_PRESET	Static	ARRAY of WORD	I, Q, M, D, L or constant	One-dimensional array with the time factor for each step. Recommended selection of the indices: [1 to 16]. In this case, S_PRESET [x] has the time factor of step x.
OUT_VAL	Static	ARRAY of BOOL	I, Q, M, D, L	Two-dimensional array with the values to be output in each step if they have not been masked out using S_MASK. Recommended selection of the indices: [1 to 16, 0 to 15]. In this case, OUT_VAL [x, y] has the value assigned to the output bit OUTy in step x.
S_MASK	Static	ARRAY of BOOL	I, Q, M, D, L	Two-dimensional array with the mask bits for each step. Recommended selection of the indices: [1 to 16, 0 to 15]. In this case, S_MASK [x, y] contains the mask bit for the "y"the value to be output in step x. Meaning of the mask bits: • 0: The corresponding value of the previous step is assigned to the relevant output bit. • 1: The corresponding value from OUT_VAL is assigned to the relevant output bit.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits (Page 2455)

Switching display formats in the program status (Page 2980)

DCAT: Discrete control-timer alarm

Description

The "Discrete control-timer alarm" instruction is used to accumulate the time from the point at which the CMD parameter issues the command to open or close. The time is accumulated until the preset time (PT) is exceeded or the information is received that the device was opened or closed (O_FB or C_FB) within the specified time. If the preset time is exceeded before the information on the opening or closing of the device is received, the corresponding alarm is activated. If the signal state on the command input changes state before the preset time, the time is restarted.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For more information on this topic, refer to "See also".

The "Discrete control-timer alarm" instruction has the following effects on the input conditions:

- When the signal state of the CMD parameter changes from "0" to "1", the signal states of the parameters Q, CMD_HIS, ET, OA and CA are influenced as follows:
  - The parameters Q and CMD_HIS are set to "1".
  - The parameters ET, OA and CA are reset to "0".
- When the signal state on the parameter CMD changes from "1" to "0", the parameters Q, ET, OA, CA and CMD_HIS are reset to "0".
- When the signal state of the CMD and CMD_HIS parameters is "1" and the parameter O_FB is set to "0", the time difference (ms) since the last execution of the instruction is added to the value at the parameter ET. If the value of the parameter ET exceeds the value of the parameter PT, the signal state on the parameter OA is set to "1". If the value of the parameter ET does not exceed the value of the parameter PT, the signal state on the parameter OA is reset to "0". The value at the parameter CMD_HIS is reset to the value of the parameter CMD.
- If the signal state of the parameters CMD, CMD_HIS and O_FB are set to "1" and the parameter C_FB has the value "0", the signal state of the parameter OA is set to "0". The value of parameter ET is set to the value of parameter PT. If the signal state of the parameter O_FB changes to "0", the alarm is set the next time the instruction is executed. The value of the parameter CMD_HIS is set to the value of the parameter CMD.
- If the parameters CMD, CMD_HIS and C_FB have the value "0", the time difference (ms) since the last execution of the instruction is added to the value of the parameter ET. If the value of the parameter ET exceeds the value of the parameter PT, the signal state of the parameter CA is set to "1". If the value at the parameter PT is not exceeded, the parameter CA has the signal state "0". The value of the parameter CMD_HIS is set to the value of the parameter CMD.

- If the parameters CMD, CMD_HIS and O_FB have the signal state "0" and the parameter C_FB is set to "1", the parameter CA is set to "0". The value of the parameter ET is set to the value of the parameter PT. If the signal state of the parameter C_FB changes to "0", the alarm is set the next time the instruction is executed. The value of parameter CMD_HIS is set to the value of parameter CMD.
- If the parameters O_FB and C_FB simultaneously have the signal state "1", the signal states of both alarm outputs are set to "1".

The "Discrete control-timer alarm" instruction has no error information.

Syntax

The following syntax is used for the "Discrete control-timer alarm" instruction:

```
CALL DCAT, "Instance DB"
CMD := <Operand>
O_FB := <Operand>
C_FB := <Operand>
Q := <Operand>
OA := <Operand>
CA := <Operand>
```

Parameters

The following table shows the parameters of the "Discrete control-timer alarm" instruction:

Parameters	Declaration	Data type	Memory area	Description
CMD	Input	BOOL	I, Q, M, D, L	A signal state of "0" indicates a "close" command. A signal state of "1" indicates an "open" command.
O_FB	Input	BOOL	I, Q, M, D, L	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L	Feedback input when closing
Q	Output	BOOL	I, Q, M, D, L	Shows the status of the parameter CMD
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
ET	Static	DINT	I, Q, M, D, L	Currently elapsed time, where one count = 1 ms.
PT	Static	DINT	I, Q, M, D, L	Preset time value, where one count = 1 ms.

Parameters	Declaration	Data type	Memory area	Description
PREV_TIME	Static	DWORD	I, Q, M, D, L	Previous system time
CMD_HIS	Static	BOOL	I, Q, M, D, L	CMD history bit

For more information on valid data types, refer to "See also".

Example

In the following example the parameter CMD changes from "0" to "1". After the execution of the instruction the parameter Q is set to "1" and the two alarm outputs OA and CA have the signal state "0". The parameter CMD_HIS of the instance data block is set to the signal state "1" and the parameter ET is reset to "0".

Note

You can initialize static parameters in the data block.

```
STL                          Explanation
CALL DCAT, "DCAT_DB"         // The "Discrete control-timer alarm" instruction is
                             called and the instance data block "DCAT_DB" is
                             created.
CMD := "Tag_Input_CMD"       // Opening or closing of device
O_FB := "Tag_Input_O_FB"     // Feedback when opening
C_FB := "Tag_Input_C_FB"     // Feedback when closing
Q := "Tag_Output_Q"          // Shows the status of the parameter CMD
OA := "Tag_Output_OA"        // Alarm output when opening
CA := "Tag_Output_CA"        // Alarm output when closing
```

The following tables show how the instruction works using specific values.

Before processing

In this example the following values are used for the input and output parameters:

Parameters	Operand	Value
CMD	Tag_Input_CMD	TRUE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
Q	Tag_Output_Q	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the instance data block "DCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#12
PT	DBD8	L#222
CMD_HIS	DBX16.0	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
Q	Tag_Output_Q	TRUE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the instance data block "DCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#0
CMD_HIS	DBX16.0	TRUE

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits (Page 2455)

MCAT: Motor control-timer alarm

Description

The "Motor control-timer alarm" instruction is used to accumulate the time from the point in time as of which one of the command inputs (open or close) is activated. The time is accumulated until the preset time is exceeded or the relevant feedback input indicates that the device has executed the requested operation within the specified time. If the preset time is exceeded before the feedback is received, the corresponding alarm is triggered.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Execution of the "Motor control-timer alarm" instruction

The following table shows the reactions of the "Motor control-timer alarm" to the various input conditions:

Input parameters								Output parameters								
ET	O_HIS	C_HIS	O_CMD	C_CMD	S_CMD	O_FB	C_FB	OO	CO	OA	CA	ET	O_HIS	C_HIS	Q	State
X	1	1	X	X	X	X	X	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	X	1	1	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	1	X	X	0	0	0	0	X	0	0	1	Stop
X	X	X	1	1	X	X	X	0	0	0	0	X	0	0	1	Stop
X	0	X	1	0	0	X	X	1	0	0	0	0	1	0	1	Start opening
<PT	1	0	X	0	0	0	X	1	0	0	0	INC	1	0	1	Open
X	1	0	X	0	0	1	0	0	0	0	0	PT	1	0	1	Opened
>=PT	1	0	X	0	0	0	X	0	0	1	0	PT	1	0	0	Opening alarm
X	X	0	0	1	0	X	X	0	1	0	0	0	0	1	1	Start closing
<PT	0	1	0	X	0	X	0	0	1	0	0	INC	0	1	1	Close
X	0	1	0	X	0	0	1	0	0	0	0	PT	0	1	1	Closed
>=PT	0	1	0	X	0	X	0	0	0	0	1	PT	0	1	0	Closing alarm
X	0	0	0	0	0	X	X	0	0	0	0	X	0	0	1	Stopped
Legend:																
INC	Add the time difference (ms) since the last processing of the FB to ET															
PT	PT is set to the same value as ET															
X	Cannot be used															
<PT	ET < PT															
>=PT	ET >= PT															
If the input parameters O_HIS and C_HIS both have the signal state "1", they are immediately set to signal state "0". In this case, the last row in the table (X) mentioned above is valid. Because it is therefore no longer possible to check whether the input parameters O_HIS and C_HIS have the signal state "1", the output parameters are set as follows in this case: OO = FALSE CO = FALSE OA = FALSE CA = FALSE ET = PT Q = TRUE																

Syntax

The following syntax is used for the "Motor control-timer alarm" instruction:

```
CALL MCAT, "Instance DB"
```

```
O_CMD := <Operand>
C_CMD := <Operand>
S_CMD := <Operand>
O_FB := <Operand>
C_FB := <Operand>
OO := <Operand>
CO := <Operand>
OA := <Operand>
CA := <Operand>
Q := <Operand>
```

Parameters

The following table shows the parameters of the "Motor control-timer alarm" instruction:

Parameter	Declaration	Data type	Memory area	Description
O_CMD	Input	BOOL	I, Q, M, D, L	"Open" command input
C_CMD	Input	BOOL	I, Q, M, D, L	"Close" command input
S_CMD	Input	BOOL	I, Q, M, D, L	"Stop" command input
O_FB	Input	BOOL	I, Q, M, D, L	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L	Feedback input when closing
OO	Output	BOOL	I, Q, M, D, L	"Open" output
CO	Output	BOOL	I, Q, M, D, L	"Close" output
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
Q	Output	BOOL	I, Q, M, D, L	A signal state of "0" indicates an error condition
ET	Static	DINT	I, Q, M, D, L	Currently elapsed time, where one count = 1 ms
PT	Static	DINT	I, Q, M, D, L	Preset time value, where one count = 1 ms
PREV_TIME	Static	DWORD	I, Q, M, D, L	Previous system time
O_HIS	Static	BOOL	I, Q, M, D, L	"Open" history bit
C_HIS	Static	BOOL	I, Q, M, D, L	"Close" history bit

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.

```
STL                              Explanation
CALL MCAT, "MCAT_DB"             // The "Motor control-timer alarm" instruction is
                                 called and the instance data block "MCAT_DB" is
                                 created.
O_CMD := "Tag_Input_O_CMD"       // "Open" command input
C_CMD := "Tag_Input_C_CMD"       // "Close" command input
S_CMD := "Tag_Input_S_CMD"       // "Stop" command input
O_FB := "Tag_Input_O_FB"         // Feedback when opening
C_FB := "Tag_Input_C_FB"         // Feedback when closing
OO := "Tag_OutputOpen"           // "Open" output
CO := "Tag_OutputClosed"         // "Close" output
OA := "Tag_Output_OA"            // Alarm output when opening
CA := "Tag_Output_CA"            // Alarm output when closing
Q := "Tag_Output_Q"              // Error information
```

The following tables show how the instruction works using specific values.

Before processing

In this example the following values are used for the input and output parameters:

Parameter	Operand	Value
O_CMD	Tag_Input_O_CMD	TRUE
C_CMD	Tag_Input_C_CMD	FALSE
S_CMD	Tag_Input_S_CMD	FALSE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
OO	Tag_OutputOpen	FALSE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	FALSE

The following values are saved in the instance data block "MCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#2
PT	DBD8	L#22

Parameter	Address	Value
O_HIS	DBX16.0	TRUE
C_HIS	DBX16.1	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OO	Tag_OutputOpen	TRUE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	TRUE

The following values are saved in the instance data block "MCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#0
O_HIS	DBX16.0	TRUE
CMD_HIS	DBX16.1	FALSE

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits (Page 2455)

IMC: Compare input bits with the bits of a mask

Description

You can use the "Compare input bits with the bits of a mask" to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bits of a mask. Up to 16 steps with masks can be programmed. The value of the IN_BIT0 parameter is compared with the value of the mask CMP_VAL[x,0], with "x" indicating the step number. In the CMP_STEP parameter, you specify the step number of the mask that is used for the comparison. All programmed values are compared in the same manner. Unprogrammed input bits or unprogrammed bits of the mask have a default signal state FALSE.

If a match is found in the comparison, the signal state of the OUT parameter is set to "1". Otherwise the OUT parameter is set to "0".

If the value of CMP_STEP parameter is greater than 15, the instruction is not executed. An error message is output at the ERR_CODE parameter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For more information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Compare input bits with the bits of a mask" instruction:

```
CALL IMC, "Instance DB"
IN_BIT0 - 15 := <Operand>
CMP_STEP := <Operand>
OUT := <Operand>
ERR_CODE := <Operand>
```

Parameters

The following table shows the parameters of the "Compare input bits with the bits of a mask" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN_BIT0	Input	BOOL	I, Q, M, D, L	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L	Input bit 2 is compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L	Input bit 6 is compared with bit 6 of the mask.

Parameter	Declaration	Data type	Memory area	Description
IN_BIT7	Input	BOOL	I, Q, M, D, L	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L	Input bit 12 is compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L	Input bit 15 is compared with bit 15 of the mask.
CMP_STEP	Input	BYTE	I, Q, M, D, L, P or constant	The step number of the mask used for the comparison.
OUT	Output	BOOL	I, Q, M, D, L	A signal state of "1" indicates that a match was found. A signal state of "0" indicates that no match was found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
CMP_VAL	Static	ARRAY of BOOL	I, Q, M, D, L	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

For more information on valid data types, refer to "See also".

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000A	The value at the CMP_STEP parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For more information on toggling display formats, refer to "See also".	

Example

In the following example all 16 input bits are compared with the mask for step 2. The signal state of the OUT parameter is set to TRUE, since the input bits match the mask for step 2.

Note

You can initialize static parameters in the data block.

```
STL                              Explanation
CALL IMC, "IMC_DB"               // The instruction "Compare input bits with the bits
                                 of a mask" is called and the instance data block
                                 "IMC_DB" is created.
IN_BIT0 := "Tag_Input_BIT0"      // Input bit 0
IN_BIT1 := "Tag_Input_BIT1"      // Input bit 1
IN_BIT2 := "Tag_Input_BIT2"      // Input bit 2
IN_BIT3 := "Tag_Input_BIT3"      // Input bit 3
IN_BIT4 := "Tag_Input_BIT4"      // Input bit 4
IN_BIT5 := "Tag_Input_BIT5"      // Input bit 5
IN_BIT6 := "Tag_Input_BIT6"      // Input bit 6
IN_BIT7 := "Tag_Input_BIT7"      // Input bit 7
IN_BIT8 := "Tag_Input_BIT8"      // Input bit 8
IN_BIT9 := "Tag_Input_BIT9"      // Input bit 9
IN_BIT10 := "Tag_Input_BIT10"    // Input bit 10
IN_BIT11 := "Tag_Input_BIT11"    // Input bit 11
IN_BIT12 := "Tag_Input_BIT12"    // Input bit 12
IN_BIT13 := "Tag_Input_BIT13"    // Input bit 13
IN_BIT14 := "Tag_Input_BIT14"    // Input bit 14
IN_BIT15 := "Tag_Input_BIT15"    // Input bit 15
CMP_STEP := "Tag_CMP_STEP"       // Step number of the mask
OUT := "Tag_Output"              // Indicates whether or not a match was found
ERR_CODE := "Tag_ErrorCode"      // Error information
```

The following tables show how the instruction works using specific values.

Before processing

In this example the following values are used for the input and output parameters:

Parameters	Operand	Value
IN_BIT0	Tag_Input_BIT0	TRUE
IN_BIT1	Tag_Input_BIT1	TRUE
IN_BIT2	Tag_Input_BIT2	FALSE
IN_BIT3	Tag_Input_BIT3	TRUE
IN_BIT4	Tag_Input_BIT4	TRUE
IN_BIT5	Tag_Input_BIT5	FALSE
IN_BIT6	Tag_Input_BIT6	TRUE
IN_BIT7	Tag_Input_BIT7	TRUE
IN_BIT8	Tag_Input_BIT8	FALSE
IN_BIT9	Tag_Input_BIT9	TRUE
IN_BIT10	Tag_Input_BIT10	TRUE
IN_BIT11	Tag_Input_BIT11	FALSE
IN_BIT12	Tag_Input_BIT12	TRUE
IN_BIT13	Tag_Input_BIT13	TRUE
IN_BIT14	Tag_Input_BIT14	FALSE
IN_BIT15	Tag_Input_BIT15	TRUE
CMP_STEP	Tag_CMP_STEP	B#16#02
OUT	Tag_Output	FALSE
ERR_CODE	Tag_ErrorCode	W#16#0000

The following values for the mask of step 2 are saved in the instance data block "IMC_DB" of the instruction:

Parameter	Address	Value
CMP_VAL [2,0]	DBX12.0	TRUE
CMP_VAL [2,1]	DBX12.1	TRUE
CMP_VAL [2,2]	DBX12.2	FALSE
CMP_VAL [2,3]	DBX12.3	TRUE
CMP_VAL [2,4]	DBX12.4	TRUE
CMP_VAL [2,5]	DBX12.5	FALSE
CMP_VAL [2,6]	DBX12.6	TRUE
CMP_VAL [2,7]	DBX12.7	TRUE
CMP_VAL [2,8]	DBX13.0	FALSE
CMP_VAL [2,0]	DBX13.1	TRUE
CMP_VAL [2,10]	DBX13.2	TRUE
CMP_VAL [2,11]	DBX13.3	FALSE
CMP_VAL [2,12]	DBX13.4	TRUE
CMP_VAL [2,13]	DBX13.5	TRUE
CMP_VAL [2,14]	DBX13.6	FALSE
CMP_VAL [2,15]	DBX13.7	TRUE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OUT	Tag_Output	TRUE
ERR_CODE	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits (Page 2455)

Switching display formats in the program status (Page 2980)

SMC: Compare scan matrix

Description

You can use the "Compare scan matrix" instruction to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bit of the comparison masks for each step. Processing starts at step 1 and is continued until the last programmed step (LAST) or until a match is found. The input bit of the IN_BIT0 parameter is compared with the value of the mask CMP_VAL[x,0], with "x" indicating the step number. All programmed values are compared in the same manner. If a match is found the signal state of the OUT parameter is set to "1" and the step number with the matching mask is written in the OUT_STEP parameter. Unprogrammed input bits or unprogrammed bits of the mask have a default signal state FALSE. If more than one step has a matching mask, only the first one found is indicated in the OUT_STEP parameter. If no match is found, the signal state of the OUT parameter is set to "0". In this case the value at the OUT_STEP parameter is greater by "1" than the value at the LAST parameter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For more information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Compare scan matrix" instruction:

```
CALL SMC, "Instance DB"
IN_BIT0 - 15 := <Operand>
```

```
OUT := <Operand>
OUT_STEP := <Operand>
ERR_CODE := <Operand>
```

Parameters

The following table shows the parameters of the "Compare scan matrix" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN_BIT0	Input	BOOL	I, Q, M, D, L	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L	Input bit 2 is compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L	Input bit 12 is compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L	Input bit 15 is compared with bit 15 of the mask.
OUT	Output	BOOL	I, Q, M, D, L	A signal state of "1" indicates that a match was found. A signal state of "0" indicates that no match was found.

Parameter	Declaration	Data type	Memory area	Description
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
OUT_STEP	Output	BYTE	I, Q, M, D, L, P	Contains the step number with the matching mask, or the step number which is greater by "1" than the value at the LAST parameter, provided no match is found.
LAST	Static	BYTE	I, Q, M, D, L, P	Specifies the step number of the last step to be scanned for a matching mask.
CMP_VAL	Static	ARRAY of WORD	I, Q, M, D, L	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

For more information on valid data types, refer to "See also".

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000E	The value at the LAST parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For more information on toggling display formats, refer to "See also".	

Example

In this example, all 16 input bits are compared with the mask for steps 0 through 5 until a match is found. Only the masks for steps 0 through 2 are scanned, since the mask for step 2 matches the input bit.

Note

You can initialize static parameters in the data block.

```
STL                              Explanation
CALL SMC, "SMC_DB"               // The "Compare scan matrix" instruction is called and
                                 the instance data block "SMC_DB" is created.
IN_BIT0 := "Tag_Input_BIT0"      // Input bit 0
IN_BIT1 := "Tag_Input_BIT1"      // Input bit 1
IN_BIT2 := "Tag_Input_BIT2"      // Input bit 2
```

```
STL                              Explanation
IN_BIT3 := "Tag_Input_BIT3"    // Input bit 3
IN_BIT4 := "Tag_Input_BIT4"    // Input bit 4
IN_BIT5 := "Tag_Input_BIT5"    // Input bit 5
IN_BIT6 := "Tag_Input_BIT6"    // Input bit 6
IN_BIT7 := "Tag_Input_BIT7"    // Input bit 7
IN_BIT8 := "Tag_Input_BIT8"    // Input bit 8
IN_BIT9 := "Tag_Input_BIT9"    // Input bit 9
IN_BIT10 := "Tag_Input_BIT10"  // Input bit 10
IN_BIT11 := "Tag_Input_BIT11"  // Input bit 11
IN_BIT12 := "Tag_Input_BIT12"  // Input bit 12
IN_BIT13 := "Tag_Input_BIT13"  // Input bit 13
IN_BIT14 := "Tag_Input_BIT14"  // Input bit 14
IN_BIT15 := "Tag_Input_BIT15"  // Input bit 15
OUT := "Tag_Output"            // Indicates whether or not a match was found.
OUT_STEP := "Tag_Output_STEP"  // Contains the step number with the relevant mask.
ERR_CODE := "Tag_ErrorCode"    // Error information
```

The following tables show how the instruction works using specific values.

Before processing

In this example the following values are used for the input and output parameters:

Parameters	Operand	Value
IN_BIT0	Tag_Input_BIT0	TRUE
IN_BIT1	Tag_Input_BIT1	TRUE
IN_BIT2	Tag_Input_BIT2	FALSE
IN_BIT3	Tag_Input_BIT3	TRUE
IN_BIT4	Tag_Input_BIT4	TRUE
IN_BIT5	Tag_Input_BIT5	FALSE
IN_BIT6	Tag_Input_BIT6	TRUE
IN_BIT7	Tag_Input_BIT7	TRUE
IN_BIT8	Tag_Input_BIT8	FALSE
IN_BIT9	Tag_Input_BIT9	TRUE
IN_BIT10	Tag_Input_BIT10	TRUE
IN_BIT11	Tag_Input_BIT11	FALSE
IN_BIT12	Tag_Input_BIT12	TRUE
IN_BIT13	Tag_Input_BIT13	TRUE
IN_BIT14	Tag_Input_BIT14	FALSE
IN_BIT15	Tag_Input_BIT15	TRUE
OUT	Tag_Output	FALSE
OUT_STEP	Tag_Output_STEP	B#16#00
ERR_CODE	Tag_ErrorCode	W#16#0000

The following values for the mask of step 2 are saved in the instance data block "SMC_DB" of the instruction:

Parameters	Address	Value
CMP_VAL [2,0]	DBX12.0	TRUE
CMP_VAL [2,1]	DBX12.1	TRUE
CMP_VAL [2,2]	DBX12.2	FALSE
CMP_VAL [2,3]	DBX12.3	TRUE
CMP_VAL [2,4]	DBX12.4	TRUE
CMP_VAL [2,5]	DBX12.5	FALSE
CMP_VAL [2,6]	DBX12.6	TRUE
CMP_VAL [2,7]	DBX12.7	TRUE
CMP_VAL [2,8]	DBX13.0	FALSE
CMP_VAL [2,0]	DBX13.1	TRUE
CMP_VAL [2,10]	DBX13.2	TRUE
CMP_VAL [2,11]	DBX13.3	FALSE
CMP_VAL [2,12]	DBX13.4	TRUE
CMP_VAL [2,13]	DBX13.5	TRUE
CMP_VAL [2,14]	DBX13.6	FALSE
CMP_VAL [2,15]	DBX13.7	TRUE
LAST	DB84	B#16#05

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OUT	Tag_Output	TRUE
OUT_STEP	Tag_Output_STEP	B#16#02
ERR_CODE	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits (Page 2455)

Switching display formats in the program status (Page 2980)

LEAD_LAG: Lead and lag algorithm

Description

Use the "Lead and lag algorithm" instruction to process signals with an analog tag. The gain value at the GAIN parameter must be greater than zero. The result of the "Lead and lag algorithm" instruction is calculated using the following equation:

$$OUT = \left[\frac{LG_TIME}{LG_TIME + SAMPLE_T}\right] PREV_OUT + GAIN \left[\frac{LD_TIME + SAMPLE_T}{LG_TIME + SAMPLE_T}\right] IN - GAIN \left[\frac{LD_TIME}{LG_TIME + SAMPLE_T}\right] * PREV_IN$$

The instruction "Lead and lag algorithm" supplies plausible results only when processing is in fixed program cycles. The same units must be specified at the parameters LD_TIME, LG_TIME and SAMPLE_T. At LG_TIME > 4 + SAMPLE_T, the instruction approaches the following function:

OUT = GAIN * ((1 + LD_TIME * s) / (1 + LG_TIME * s)) * IN

When the value of the GAIN parameter is less than or equal to zero, the calculation is not performed and an error information is output at the ERR_CODE parameter.

You can use the "Lead and lag algorithm" instruction in conjunction with loops as a compensator in dynamic feed-forward control. The instruction consists of two operations. The "Lead" operation shifts the phase of the OUT output so that the output leads the input. The "Lag" operation, on the other hand, shifts the output so that the output lags behind the input. Because the "Lag" operation is equivalent to an integration, it can be used as a noise suppressor or as a low-pass filter. The "Lead" operation is equivalent to a differentiation and can therefore be used as a high-pass filter. The two instructions together (Lead and Lag) result in the output phase lagging behind the input at lower frequencies and leading it at higher frequencies. This means that the "Lead and lag algorithm" instruction can be used as a band pass filter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For more information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Lead and lag algorithm" instruction:

```
CALL LEAD_LAG, "Instance DB"
IN := <Operand>
SAMPLE_T := <Operand>
OUT := <Operand>
ERR_CODE := <Operand>
```

Parameters

The following table shows the parameters of the "Lead and lag algorithm" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	REAL	I, Q, M, D, L, P or constant	The input value of the current sample time (cycle time) to be processed.
SAMPLE_T	Input	INT	I, Q, M, D, L, P or constant	Sample time
OUT	Output	REAL	I, Q, M, D, L, P	Result of the instruction
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
LD_TIME	Static	REAL	I, Q, M, D, L, P or constant	Lead time in the same unit as sample time.
LG_TIME	Static	REAL	I, Q, M, D, L, P or constant	Lag time in the same unit as sample time
GAIN	Static	REAL	I, Q, M, D, L, P or constant	Gain as % / % (the ratio of the change in output to a change in input as a steady state).
PREV_IN	Static	REAL	I, Q, M, D, L, P or constant	Previous input
PREV_OUT	Static	REAL	I, Q, M, D, L, P or constant	Previous output

For more information on valid data types, refer to "See also".

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
0009	The value at the GAIN parameter is less than or equal to zero.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For more information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.

```
STL                              Explanation
CALL LEAD_LAG, "LEAD_LAG_DB"     // The "Lead and lag algorithm" instruction is called
                                 and the instance data block "LEAD_LAG_DB" is created.
IN := "Tag_Input"                // The input value of the current sample time (cycle
                                 time) to be processed.
SAMPLE_T :=                      // Sample time
"Tag_Input_SAMPE_T"
OUT := "Tag_Output_Result"       // Result of the instruction
ERR_CODE := "Tag_ErrorCode"      // Error information
```

The following tables show how the instruction works using specific values.

Before processing

In this example the following values are used for the input parameters:

Parameters	Operand	Value
IN	Tag_Input	2.0
SAMPLE_T	Tag_InputSampleTime	10

The following values are saved in the instance data block "LEAD_LAG_DB" of the instruction:

Parameters	Address	Value
LD_TIME	DBD12	2.0
LG_TIME	DBD16	2.0
GAIN	DBD20	1.0
PREV_IN	DBD24	6.0
PREV_OUT	DBD28	6.0

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameters	Operand	Value
OUT	Tag_Output_Result	2.0

The following values are saved in the instance data block "LEAD_LAD_DB" of the instruction:

Parameters	Operand	Value
PREV_IN	DBD24	2.0
PREV_OUT	DBD28	2.0

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits (Page 2455)

Switching display formats in the program status (Page 2980)

TONR_X: Time accumulator

Description

The "Time accumulator" instruction is used to accumulate time values within a period set by the PV parameter. When the TMR_EN parameter changes to signal state "1", the instruction is executed and the time set at PV starts. The "Time accumulator" instruction adds the value of the DELTA_T parameter to the value of the ET parameter as long as the following conditions are satisfied:

- The TMR_EN parameter has the signal state "1".
- The time value of the ET parameter is less than the time value of the PV parameter.
- The RESET parameter returns signal state "0"

When the signal state on the TMR_EN parameter changes to "0", the time on the DELTA_T parameter is no longer added to the time value at the ET parameter.

When the time value of the ET parameter reaches the time value of the PV parameter, the Q parameter has signal state "1". As long as the Q parameter has the signal state "1", the value of the ET parameter remains unchanged. The value of the ET parameter is reset to "0" when the value "1" is set in the RESET parameter.

Since the "Time accumulator" instruction uses the execution time (DELTA_T) of the last cycle of the organization block (OB), the instruction may only be called in cyclic organization blocks.

Note

You have to move the cycle time of the organization block from the startup information to the global tag of the DELTA_T parameter.

Syntax

Use the following syntax for the "Time accumulator" instruction:

```
CALL TONR_X
TMR_EN := <Operand>
RESET := <Operand>
PV := <Operand>
DELTA_T := <Operand>
Q := <Operand>
ET := <Operand>
```

Parameters

The following table shows the parameters of the "Time accumulator" instruction:

Parameters	Declaration	Data type	Memory area	Description
TMR_EN	Input	BOOL	I, Q, M, D, L	Activates the time for accumulating the time value.
RESET	Input	BOOL	I, Q, M, D, L	Reset input
PV	Input	DINT	I, Q, M, D, L, P or constant	Maximum duration of time recording
DELTA_T	Input	INT	I, Q, M, D, L or constant	Cycle time of OB from previous cycle
Q	Output	BOOL	Q, M, D, L	Output that is set when the value of the ET parameter is greater than or equal to the value of the PV parameter.
ET	InOut	DINT	I, Q, M, D, L	Current timer value of the elapsed time

For more information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL TONR_X                  // The "Time accumulator" instruction is called.
TMR_EN := "Tag_TMR_EN"       // Start input
RESET := "Tag_ResetTimer"    // Reset input
```

STL	Explanation
PV := "Tag_Input_PV"	// Maximum duration of time recording
DELTA_T := "Tag_DELTA_T"	// Cycle time of OB from previous cycle
ET := "Tag_Input_ET"	// Accumulated time
Q := "Tag_Ouput_Q"	// Indicates whether the time set for the PV parameter has been reached.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits (Page 2455)

WSR: Save data to shift register

Description

The "Save data to shift register" instruction is used to write a bit from a specified source to a shift register. Each element of the register is shifted into the next address. The data contained in the last address of the shift register is lost after the shift.

Each time the instruction is executed with the value "0" in the RESET parameter, new data are read from the source (S_DATA) and shifted to the start address (START) of the shift register. The LEN parameter specifies the number of elements to be shifted. You specify the data type of the elements on the E_TYPE parameter. The instruction is not executed if an invalid data type is specified on the E_TYPE parameter.

The register is deleted if the RESET parameter is set to "1" when the instruction is executed. The Q parameter is set to the signal state "1" when the shift register is filled with zeros or deleted.

Syntax

The following syntax is used for the "Save data to shift register" instruction:

```
CALL WSR
RESET := <Operand>
S_DATA := <Operand>
START := <Operand>
LEN := <Operand>
E_TYPE := <Operand>
Q := <Operand>
```

Parameter

The following table shows the parameters of the "Save data to shift register" instruction:

Parameters	Declaration	Data type	Memory area	Description
RESET	Input	BOOL	I, Q, M, D, L	Reset input
S_DATA	Input	POINTER	I, Q, M, D	Points to the source data element to be shifted to the start address (START). For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
START	Input	POINTER	I, Q, M, D	Points to the start address of the register. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
LEN	Input	WORD	I, Q, M, D, L, P	Number of elements to be shifted.
E_TYPE	Input	BYTE	I, Q, M, D, L, P	Specifies the data type of the elements to be shifted. The following data types are permitted: • W#16#04 = WORD • B#16#05 = INT • B#16#06 = DWORD • B#16#07 = DINT • B#16#08 = REAL
Q	Output	BOOL	Q, M, D, L	Is set to "1" if RESET parameter is active (1) or all elements to be shifted have the value "0".

For more information on valid data types, refer to "See also".

Example

In the following example four elements of the data type WORD are shifted starting from the address specified on the START parameter. When the first element of the register has been shifted to the next address, the first address is filled with the data that the pointer references on the S_DATA parameter. The value of the last element in the register is lost in the process.

```
STL                                Explanation
CALL WSR                           // The "Save data to shift register" instruction is
                                   called.
RESET := "Tag_Input_Reset"         // Reset input
S_DATA :=                          // Source element that is shifted to the start address.
"Tag_Input_DataSource"
START :=                           // Start address
"Tag_Input_StartAddress"
LEN := "Tag_Input_LEN"             // Number of elements to be shifted
E_TYPE :=                          // Data type of the elements
"Tag_Input_ElementType"
Q := "Tag_Output_Q"                // Status of the register
```

The following tables show how the instruction works using specific values.

Before processing

In this example the following values are used for the input and output parameters:

Parameter	Operand	Value
RESET	Tag_Input_Reset	0
S_DATA	Tag_Input_SourceData	P#DB2.DBX6.0 DW6 = W#16#1234
START	Tag_Input_StartAddress	P#DB1.DBX0.0 DW0 = W#16#1111 DW2 = W#16#2222 DW4 = W#16#3333 DW6 = W#16#4444
LEN	Tag_Input_LEN	W#16#04
E_TYPE	Tag_Input_ElementType	W#16#04
Q	Tag_Output_Q	1

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
RESET	Tag_Input_Reset	0
S_DATA	Tag_Input_SourceData	P#DB2.DBX6.0 DW6 = W#16#1234
START	Tag_Input_StartAddress	P#DB1.DBX0.0 DW0 = W#16#1234 DW2 = W#16#1111 DW4 = W#16#2222 DW6 = W#16#3333
LEN	Tag_Input_LEN	W#16#04
E_TYPE	Tag_Input_ElementType	B#16#04
Q	Tag_Output_Q	1

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits (Page 2455)

SHRB: Shift bit to shift register

Description

The "Shift bit to shift register" instruction is used to write a bit from a specified source (DATA) to a shift register. Each time the instruction is executed with a value "0" on the RESET parameter, new data are read from the source and shifted to the start address (S_BIT) of the shift register. All following bits are shifted by one. The bit contained at the last address (S_BIT + N) is lost after the shift. If the RESET parameter is set to "1" during the processing of the instruction, the addresses in the shift register are reset to "0" and not shifted.

Syntax

The following syntax is used for the "Save data to shift register" instruction:

```
CALL SHRB
DATA := <Operand>
RESET := <Operand>
S_BIT := <Operand>
```

```
N := <Operand>
```

Parameters

The following table shows the parameters of the "Shift bit to shift register" instruction:

Parameters	Declaration	Data type	Memory area	Description
DATA	Input	BOOL	I, Q, M, D, L	Source bit
RESET	Input	BOOL	I, Q, M, D, L	Reset input
S_BIT	Input	POINTER	I, Q, M, D	Points to the start bit of the shift register. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	WORD	I, Q, M, D, L, P	Number of bits to be shifted.

For more information on valid data types, refer to "See also".

Example

In the following example the N parameter is set to the value 14 ("E" in hexadecimal). As a result 14 bits starting from the address specified on the S_BIT parameter are shifted. After the bits have been shifted, the first address is filled with the data from the source on the DATA parameter. The last bit value is lost.

```
STL                          Explanation
CALL SHRB                    // The "Shift bit to shift register" instruction is
                             called.
DATA := "Tag_Input_Data"     // Source bit that is shifted to start address
RESET := "Tag_Input_Reset"   // Reset input
S_BIT :=                     // Start address
"Tag_Input_BitSource"
N := "Tag_Input_Number"      // Number of bits to be shifted
```

The following tables show how the instruction works using specific values.

Before processing

In this example the following values are used for the input and output parameters:

Parameter	Operand	Value
RESET	Tag_Input_Reset	0
DATA	Tag_Input_Data	0
S_BIT	Tag_Input_BitSource	P#DB1.DBX1.4
N	Tag_Input_Number	B#16#0E

Before processing the instruction the following signal states are stored in the 14 bits of the register:

1.7	...		S_BIT			...	1.0
0	0	1	1	1	0	1	0

2.7	...					...	2.0
0	0	1	1	1	0	1	0

3.7	...					...	3.0
0	0	1	1	1	0	1	0

After processing

After the processing of the instruction the following signal states are stored in the 14 bits of the register:

1.7	...		S_BIT			...	1.0
0	1	1	0	1	0	1	0

2.7	...					...	2.0
0	1	1	1	0	1	0	0

3.7	...					...	3.0
0	0	1	1	1	0	0	0

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits (Page 2455)

SEG: Create bit pattern for seven-segment display

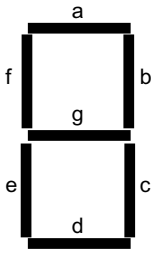
Description

The instruction "Create bit pattern for seven-segment display" is used to convert each of the four hexadecimal digits of the specified source word (IN) into an equivalent bit pattern for a seven-segment display. The result of the instruction is output in the double word on the OUT parameter.

The following relation exists between the hexadecimal digits and the assignment of the 7 segments (a, b, c, d, e, f, g):

Input digit (Binary)	Assignment of the segments - g f e d c b a	Display (Hexadecimal)
0000	00111111	0
0001	00000110	1
0010	01011011	2
0011	01001111	3
0100	01100110	4
0101	01101101	5
0110	01111101	6
0111	00000111	7
1000	01111111	8
1001	01100111	9
1010	01110111	A
1011	01111100	B
1100	00111001	C
1101	01011110	D
1110	01111001	E
1111	01110001	F

Seven-segment display



Syntax

Use the following syntax for the "Create bit pattern for seven-segment display" instruction:

```
CALL SEG
IN := <Operand>
OUT := <Operand>
```

Parameters

The following table shows the parameters of the "Create bit pattern for seven-segment display" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	WORD	I, Q, M, D, L, P or constant	Source word with four hexadecimal digits
OUT	Output	DWORD	I, Q, M, D, L, P	Bit pattern for the seven-segment display

For more information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
CALL SEG                     // The "Create bit pattern for seven-segment display"
                             instruction is called.
IN := "Tag_Input"            // Source word
OUT := "Tag_Output"          // Bit pattern for the seven-segment display
```

The following table shows how the instruction functions using specific values:

Parameters	Operand	Value	
Hexadecimal	Binary		
IN	Tag_Input	W#16#1234	0001 0010 0011 0100
OUT	Tag_Output	DW#16#065B4F66	00000110 01011011 01001111 01100110 Display: 1234

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits (Page 2455)

BCDCPL: Create tens complement

Description

The "Create tens complement" instruction is used to create the tens complement of a seven-digit BCD number specified by the IN parameter. This instruction uses the following mathematical formula to calculate:

10000000 (as BCD)

– 7-digit BCD value

Tens complement (as BCD)

Syntax

The following syntax is used for the "Create tens complement" instruction:

```
CALL BCDCPL
IN := <Operand>
ERR_CODE := <Operand>
```

Parameters

The following table shows the parameters of the "Create tens complement" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	DWORD	I, Q, M, D, L, P or constant	7-digit BCD number
ERR_CODE	Output	DWORD	I, Q, M, D, L, P	Result of the instruction

For more information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
CALL BCDCPL	// The "Create tens complement" instruction is called.
IN := "Tag_Input"	// 7-digit BCD number
ERR_CODE := "Tag_Output"	// Tens complement of the specified BCD number

The following table shows how the instruction functions using specific values:

Parameter	Operand	Value*
IN	Tag_Input	DW#16#01234567
ERR_CODE	Tag_Output	DW#16#08765433
*The error codes can be displayed as integer or hexadecimal value in the program editor. For more information on toggling display formats, refer to "See also".		

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits (Page 2455)

Switching display formats in the program status (Page 2980)

BITSUM: Count number of set bits

Description

The "Count number of set bits" instruction is used to count the number of bits of an operand that is set to the signal state "1". The operand whose bits are to be counted is specified on the IN parameter. The result of the instruction is output at the RET_VAL parameter.

Syntax

The following syntax is used for the "Count number of set bits" instruction:

```
CALL BITSUM
IN := <Operand>
RET_VAL := <Operand>
```

Parameters

The following table shows the parameters of the "Count number of set bits" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	DWORD	I, Q, M, D, L, P or constant	Operand whose set bits are counted
RET_VAL	Output	INT	I, Q, M, D, L, P	Number of set bits

For more information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
`CALL BITSUM`	`// The "Count number of set bits" instruction is called.`
`IN := "Tag_Input"`	`// Operand whose set bits are counted`
`RET_VAL := "Tag_Output"`	`// Number of set bits of the operand "Tag_Input"`

The following table shows how the instruction functions using specific values:

Parameter	Operand	Value*
IN	Tag_Input	DW#16#12345678
RET_VAL	Tag_Output	W#16#000D (13 bits)
*The error codes can be displayed as integer or hexadecimal value in the program editor. For more information on toggling display formats, refer to "See also".		

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits (Page 2455)

Switching display formats in the program status (Page 2980)

Basic instructions

Bit logic operations

A: AND logic operation

Description

Use the instruction "AND logic operation" to query whether the signal state of a binary operand is "1" and to AND the query result with the signal state of the result of logic operation (RLO). The query result is hereby identical with the signal state of the checked operand.

After execution of the instruction the RLO will be "1" if both linked signal states are also "1". If one of the linked signal states is "0", the RLO is "0" after executing the instruction.

If you use the instruction multiple times consecutively, all query results must be "1" for the common result of the logic operation to also be "1".

You can also use the instruction "AND logic operation" to query the signal state of the status word. To do this, specify the relevant status bit (==0, <>0, >0, <0, >=0, <=0, OV, OS, UO, BR) as parameter of the instruction.

Syntax

Use the following syntax for the "AND logic operation" instruction:

```
A <operand>
```

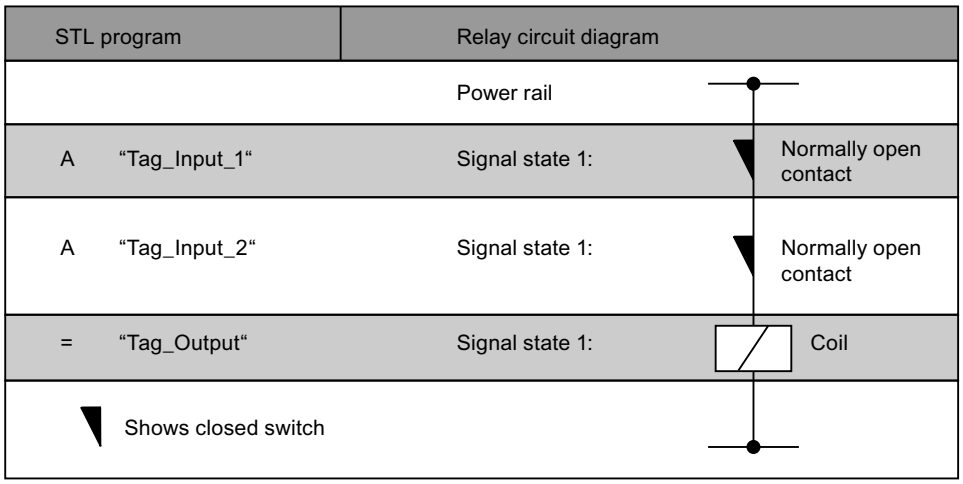
Parameters

The following table shows the parameters of the "AND logic operation" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, D, L, T, C	Operand whose signal state is queried.

Example

The following example shows how the instruction works:



See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Example of detecting the direction of a conveyor belt (Page 3135)

Example of detecting the fill level of a storage area (Page 3136)

Example: Word logic instructions (Page 3140)

Example of a step sequence (Page 3142)

Basic information on the status word (Page 2453)

Setting the status bit for bit logic operations (Page 2455)

Querying status bits with bit logic operation instructions (Page 2463)

AN: Negated AND logic operation

Description

Use the instruction "Negated AND logic operation" to query whether the signal state of a binary operand is "0" and to AND the query result with the signal state of the result of logic operation (RLO). The result of the query is hereby the negated signal state of the checked operand. If the operand has signal state "0," the result of the query is "1". If the operand has signal state "1," the result of the query is "0".

After the instruction executes, the RLO is "1" if the result of the query and the RLO before the execution of the instruction have signal state "1".

After the instruction executes, the RLO is "0" if the result of the query or the RLO before the execution of the instruction have signal state "0".

You can also use the instruction "Negated AND logic operation" to query the signal state of the status word. To do this, specify the relevant status bit (==0, <>0, >0, <0, >=0, <=0, OV, OS, UO, BR) as parameter of the instruction.

Syntax

Use the following syntax for the "Negated AND logic operation" instruction:

```
AN <operand>
```

Parameters

The following table shows the parameters of the "Negated AND logic operation" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, D, L, T, C	Operand whose signal state is queried.

Example

The following example shows how the instruction works:

STL program		Relay circuit diagram	
		Power rail	
A	“Tag_Input_1“	Signal state 0:	Normally open contact
AN	“Tag_Input_2“	Signal state 1:	Normally closed contact
=	“Tag_Output“	Signal state 0:	Coil

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Example of detecting the direction of a conveyor belt (Page 3135)

Example of detecting the fill level of a storage area (Page 3136)

Basic information on the status word (Page 2453)

Setting the status bit for bit logic operations (Page 2455)

Querying status bits with bit logic operation instructions (Page 2463)

O: OR logic operation

Description

Use the instruction "OR logic operation" to query whether the signal state of a binary operand is "1" and to OR the query result with the signal state of the result of logic operation (RLO). The query result is hereby identical with the signal state of the checked operand.

After the instruction executes, the RLO is "1" if one of the linked signal states is "1". If both linked signal states are "0", the RLO is "0" after the instruction executes.

If you use the instruction multiple times consecutively, it is sufficient that one of the results of the query is "1" for the common result of the logic operation to also be "1".

You can also use the instruction "OR logic operation" to query the signal state of the status word. To do this, specify the relevant status bit (==0, <>0, >0, <0, >=0, <=0, OV, OS, UO, BR) as parameter of the instruction.

Syntax

Use the following syntax for the "OR logic operation" instruction:

```
O <Operand>
```

Parameters

The following table shows the parameters of the "OR logic operation" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, D, L, T, C	Operand whose signal state is queried.

Example

The following example shows how the instruction works:

STL program	Relay circuit diagram
	Power rail
O “Tag_Input_1“	Signal state 1 normally open contact (“Tag_Input_1“) / Signal state 0 normally open contact (“Tag_Input_2“)
O “Tag_Input_2“	
= “Tag_Output“	Signal state 1: Coil
	Shows closed switch

See also

Overview of the valid data types (Page 2236)
Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)
Example: Bit logic instructions (Page 3134)
Basic information on the status word (Page 2453)
Setting the status bit for bit logic operations (Page 2455)
Querying status bits with bit logic operation instructions (Page 2463)

ON: Negated OR logic operation

Description

Use the instruction "Negated OR logic operation" to query whether the signal state of a binary operand is "0" and to OR the query result with the signal state of the result of logic operation (RLO). The result of the query is hereby the negated signal state of the checked operand. If the operand has signal state "0," the result of the query is "1". If the operand has signal state "1," the result of the query is "0".

After the instruction executes, the RLO is "1" if the result of the query or the RLO before the execution of the instruction have signal state "1". If both linked signal states are "0", the RLO is also "0" after the instruction executes.

If you use the instruction multiple times consecutively, it is sufficient that one of the results of the query is "1" for the common result of the logic operation to also be "1".

You can also use the instruction "Negated OR logic operation" to query the signal state of the status word. To do this, specify the relevant status bit (==0, <>0, >0, <0, >=0, <=0, OV, OS, UO, BR) as parameter of the instruction.

Syntax

Use the following syntax for the "Negated OR logic operation" instruction:

```
ON <Operand>
```

Parameters

The following table shows the parameters of the "Negated OR logic operation" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, D, L, T, C	Operand whose signal state is checked

Example

The following example shows how the instruction works:

STL program		Relay circuit diagram	
		Power rail	
O	“Tag_Input_1“	Signal state 0	Normally open contact
ON	“Tag_Input_2“	Signal state 1	Normally closed contact
=	“Tag_Output“	Signal state 1	Coil

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting the status bit for bit logic operations (Page 2455)

Querying status bits with bit logic operation instructions (Page 2463)

X: EXCLUSIVE OR logic operation

Description

Use the instruction "EXCLUSIVE OR logic operation" to query whether the signal state of a binary operand is "1" and to XOR the query result with the signal state of the result of logic operation (RLO). The query result is hereby identical with the signal state of the checked operand. After the instruction EXECUTES, the RLO is "1" if the linked signal states are not equal. The RLO is "0" if both signal states are the same.

If, for example, the RLO before execution of the instruction is "1" and the queried operand is "1", the RLO is reset to "0". If the RLO is "1" and the operand has signal state "0", the RLO is set to "1" after the instruction executes.

If you use the instruction several times consecutively, it is sufficient that one of the query results is "1" for the common RLO to be "1" as well. If all query results are "1" or "0" the RLO is reset to "0".

You can also use the instruction "EXCLUSIVE OR logic operation" to query the signal state of the status word. To do this, specify the relevant status bit (==0, <>0, >0, <0, >=0, <=0, OV, OS, UO, BR) as parameter of the instruction.

Syntax

Use the following syntax for the "EXCLUSIVE OR logic operation" instruction:

```
X <Operand>
```

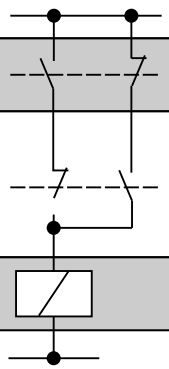
Parameters

The following table shows the parameters of the "EXCLUSIVE OR logic operation" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, D, L, T, C	Operand whose signal state is queried.

Example

The following example shows how the instruction works:

STL program	Relay circuit diagram	
	Power rail	
X "Tag_Input_1"	Contact	
X "Tag_Input_2"	Contact	
= "Tag_Output"	Coil	

See also

Overview of the valid data types (Page 2236)
Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)
Basic information on the status word (Page 2453)
Setting the status bit for bit logic operations (Page 2455)
Querying status bits with bit logic operation instructions (Page 2463)

XN: Negated EXCLUSIVE OR logic operation

Description

Use the "Negated EXCLUSIVE OR logic operation" instruction to query whether the signal state of a binary operand is "0" and to OR the query result with the signal state of the result of logic operation (RLO). The result of the query is hereby the negated signal state of the checked operand. After the instruction EXECUTES, the RLO is "1" if the linked signal states are not equal. The RLO is "0" if both signal states are the same.

If, for example, the queried operand has signal state "0," the result of the query is "1". XORing the query result with an RLO "1" results in signal state "0". The logic operation with an RLO "0" results in signal state "1".

You can also use the "Negated EXCLUSIVE OR logic operation" instruction to query the signal state of the status word. To do this, specify the relevant status bit (==0, <>0, >0, <0, >=0, <=0, OV, OS, UO, BR) as parameter of the instruction.

Syntax

Use the following syntax for the "Negated EXCLUSIVE OR logic operation" instruction:

```
XN <Operand>
```

Parameters

The following table shows the parameters of the "Negated EXCLUSIVE OR logic operation" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	BOOL	I, Q, M, D, L, T, C	Operand whose signal state is queried.

Example

The following example shows how the instruction works:

STL program		Relay circuit diagram
		Power rail
X	“Tag_Input_1“	Contact
XN	“Tag_Input_2“	Contact
=	“Tag_Output“	Coil

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting the status bit for bit logic operations (Page 2455)

Querying status bits with bit logic operation instructions (Page 2463)

O: OR logic operation of AND functions

Description

Use the instruction "OR logic operation of AND functions" to execute several AND logic operations before an OR logic operation. The overall result of the AND logic operations is stored and ORed with the current result of logic operation (RLO) and to any other signal queries.

Syntax

Use the following syntax for the "OR logic operation of AND functions" instruction:

```
O
```

No brackets are needed for the syntax of this instruction.

Example

The following example shows how the instruction works:

STL program	Relay circuit diagram
	Power rail
A "Tag_Input_1" A "Tag_Input_2" O A "Tag_Input_3" A "Tag_Input_4" O "Tag_Input_5"	"Tag_Input_1" "Tag_Input_3" "Tag_Input_5" "Tag_Input_2" "Tag_Input_4"
= "Tag_Output"	"Tag_Output"

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting the status bit for bit logic operations (Page 2455)

Querying status bits with bit logic operation instructions (Page 2463)

A(: AND logic operation with branch

Description

Use the instruction "AND logic operation with branch" to execute the instructions of a parenthesized expression before an AND logic operation.

When the instruction executes, the CPU saves the current result of logic operation (RLO), as well as the binary result BR and the status bit OR. The saved RLO is ANDed with the result of logic operation of the complete nested expression. If additional signal queries follow after the nested expression, these are also ANDed.

Parenthesized expressions can also be nested by programming an instruction within the parenthesized expression that also contains a parenthesized expression. The nesting depth is hereby limited to 7 levels.

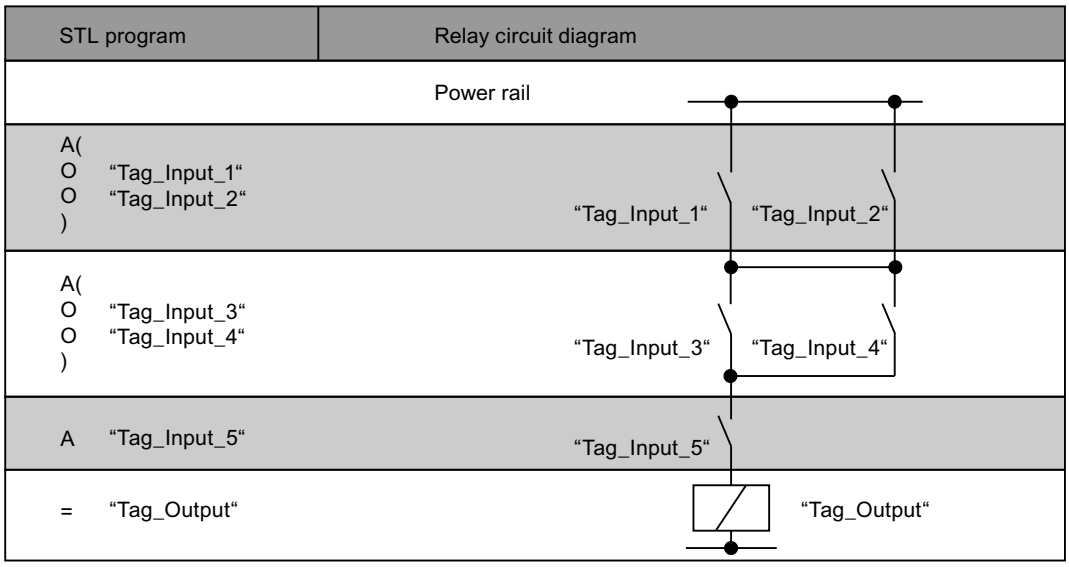
Syntax

Use the following syntax for the instruction "AND logic operation with branch":

```
A(
```

Example

The following example shows how the instruction works:



See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting the status bit for bit logic operations (Page 2455)

Querying status bits with bit logic operation instructions (Page 2463)

AN(: Negated AND logic operation with branch

Description

Use the "Negated AND logic operation with branch" instruction to execute the instructions of a parenthesized expression before an AND logic operation and to negate the complete result of the parenthesized expression.

When the instruction executes, the CPU saves the current result of logic operation (RLO), as well as the binary result BR and the status bit OR. After the instructions in the parenthesized

expression have executed, the RLO of the complete parenthesized expression is negated and ANDed with the saved RLO. If additional signal queries follow after the parenthesized expression, these are also ANDed.

Parenthesized expressions can also be nested by programming an instruction within the parenthesized expression that also contains a parenthesized expression. The nesting depth is hereby limited to 7 levels.

Syntax

Use the following syntax for the instruction "Negated AND logic operation with branch":

```
AN(
```

Example

The following example shows how the instruction works:

STL	Explanation
AN(	// Start of the nested expression
	// AND the RLO of the parenthesized expression and negate the complete result
O "Tag_Input_1"	// Query the signal state of the operand for "1" and OR with the current RLO.
O "Tag_Input_2"	// Query the signal state of the operand for "1" and OR with the current RLO.
)	// End of the nested expression
AN "Tag_Input_3"	// Query the signal state of the operand for "0" and AND with the current RLO.
= "Tag_Output"	// Assign signal state of the RLO to the operand.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting the status bit for bit logic operations (Page 2455)

Querying status bits with bit logic operation instructions (Page 2463)

O(: OR logic operation with branch

Description

Use the instruction "OR logic operation with branch" to execute the instructions of a parenthesized expression before an OR logic operation.

When the instruction executes, the CPU saves the current result of logic operation (RLO), as well as the binary result BR and the status bit OR. After the instructions contained in the parenthesized expression have executed, the saved RLO is ORed with the RLO of the complete parenthesized expression. If additional signal queries follow after the parenthesized expression, these are also ORed.

Parenthesized expressions can also be nested by programming an instruction within the parenthesized expression that also contains a parenthesized expression. The nesting depth is hereby limited to 7 levels.

Syntax

Use the following syntax for the "OR logic operation with branch" instruction:

```
O(
```

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input_1"	// Query the signal state of the operand for "1" and AND with the current RLO.
O(	// Start of the nested expression
	// OR the RLO of the parenthesized expression.
A "Tag_Input_2"	// Query the signal state of the operand for "1" and AND with the current RLO.
A "Tag_Input_3"	// Query the signal state of the operand for "1" and AND with the current RLO.
)	// End of the nested expression
= "Tag_Output"	// Assign signal state of the RLO to the operand.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting the status bit for bit logic operations (Page 2455)

Querying status bits with bit logic operation instructions (Page 2463)

ON(: Negated OR logic operation with branch

Description

Use the "Negated OR logic operation with branch" instruction to execute the instructions of a parenthesized expression before an OR logic operation and to negate the complete result of the parenthesized expression.

When the instruction executes, the CPU saves the current result of logic operation (RLO), as well as the binary result BR and the status bit OR. After the instructions in the parenthesized expression have executed, the RLO of the complete parenthesized expression is negated and ORed with the saved RLO. If additional signal queries follow after the parenthesized expression, these are also ORed.

Parenthesized expressions can also be nested by programming an instruction within the parenthesized expression that also contains a parenthesized expression. The nesting depth is hereby limited to 7 levels.

Syntax

Use the following syntax for the instruction "Negated OR logic operation with branch":

```
ON(
```

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input_1"	// Query the signal state of the operand for "1" and AND with the current RLO.
ON(	// Start of the nested expression
	// OR the RLO of the parenthesized expression and negate the complete result
A "Tag_Input_2"	// Query the signal state of the operand for "1" and AND with the current RLO.
A "Tag_Input_3"	// Query the signal state of the operand for "1" and AND with the current RLO.
)	// End of the nested expression
= "Tag_Output"	// Assign signal state of the RLO to the operand.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting the status bit for bit logic operations (Page 2455)

Querying status bits with bit logic operation instructions (Page 2463)

X(: EXCLUSIVE OR logic operation with branch

Description

Use the instruction "EXCLUSIVE OR logic operation with branch" to execute the instructions of a parenthesized expression before an EXCLUSIVE OR logic operation.

When the instruction executes, the CPU saves the current result of logic operation (RLO), as well as the binary result BR and the status bit OR. After the instructions in the parenthesized expression have executed, the saved RLO is XORed with the RLO of the complete parenthesized expression. If additional signal queries follow after the parenthesized expression, these are also XORed.

Parenthesized expressions can also be nested by programming an instruction within the parenthesized expression that also contains a parenthesized expression. The nesting depth is hereby limited to 7 levels.

Syntax

Use the following syntax for the "EXCLUSIVE OR logic operation with branch" instruction:

```
X(
```

Example

The following example shows how the instruction works:

STL	Explanation
X(	// Start of the nested expression
	// EXCLUSIVE OR the RLO of the parenthesized expression.
A "Tag_Input_1"	// Query the signal state of the operand for "1" and AND with the current RLO.
A "Tag_Input_2"	// Query the signal state of the operand for "1" and AND with the current RLO.
)	// End of the nested expression
X "Tag_Input_3"	// Query the signal state of the operand for "1" and AND with the current RLO.

STL	Explanation
= "Tag_Output"	// Assign signal state of the RLO to the operand.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting the status bit for bit logic operations (Page 2455)

Querying status bits with bit logic operation instructions (Page 2463)

XN(: Negated EXCLUSIVE OR logic operation with branch

Description

Use the "Negated EXCLUSIVE OR logic operation with branch" instruction to execute the instructions of a parenthesized expression before an EXCLUSIVE OR logic operation and to negate the complete result of the parenthesized expression.

When the instruction executes, the CPU saves the current result of logic operation (RLO), as well as the binary result BR and the status bit OR. After the instructions contained in the parenthesized expression have been executed, the RLO of the complete parenthesized expression will be negated and EXCLUSIVE ORed with the saved RLO. If additional signal queries follow after the parenthesized expression, these are also XORed.

Parenthesized expressions can also be nested by programming an instruction within the parenthesized expression that also contains a parenthesized expression. The nesting depth is hereby limited to 7 levels.

Syntax

Use the following syntax for the instruction "Negated EXCLUSIVE OR logic operation with branch":

```
XN(
```

Example

The following example shows how the instruction works:

STL	Explanation
XN(	// Start of the nested expression
	// EXCLUSIVE OR the RLO of the parenthesized expression and negate the complete result

STL	Explanation
A "Tag_Input_1"	// Query the signal state of the operand for "1" and AND with the current RLO.
A "Tag_Input_2"	// Query the signal state of the operand for "1" and AND with the current RLO.
)	// End of the nested expression
XN "Tag_Input_3"	// Query the signal state of the operand for "1" and AND with the current RLO.
= "Tag_Output"	// Assign signal state of the RLO to the operand.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting the status bit for bit logic operations (Page 2455)

Querying status bits with bit logic operation instructions (Page 2463)

): Close branch

Description

Use the "Close branch" instruction to specify the end of a parenthesized expression. After processing of the instruction the complete result of the nested expression is linked according to the instruction that was used to open the nested expression.

Use the following instructions to open a nested expression:

- A(: AND logic operation with branch
- AN(: Negated AND logic operation with branch
- O(: OR logic operation with branch
- ON(: Negated OR logic operation with branch
- X(: EXCLUSIVE OR logic operation with branch
- XN(: Negated EXCLUSIVE OR logic operation with branch

After the "Close branch" instruction, the CPU sets the binary result BR to the signal state that it had before the parenthesized expression.

Syntax

Use the following syntax for the "Close branch" instruction:

```
)
```

Example

The following example shows how the instruction works:

STL program	Relay circuit diagram
	Power rail
A(O “Tag_Input_1“ O “Tag_Input_2“)	“Tag_Input_1“ “Tag_Input_2“
A(O “Tag_Input_3“ O “Tag_Input_4“)	“Tag_Input_3“ “Tag_Input_4“
A “Tag_Input_5“	“Tag_Input_5“
= “Tag_Output“	“Tag_Output“

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bit for bit logic operations (Page 2455)

Basic information on the status word (Page 2453)

Querying status bits in STL (Page 2463)

=: Assignment

Description

Use the "Assignment" instruction to assign the signal state of a result of logic operation (RLO) saved in the CPU to a specified operand. If the RLO has signal state "1", the operand is set. If the signal is "0", the operand is reset to "0".

The execution of the "Assignment" instruction is influenced by the Master Control Relay (MCR). When the MCR bit is set to "0", the specified operand is set to signal state "0".

Syntax

Use the following syntax for the "Assignment" instruction:

```
= <Operand>
```

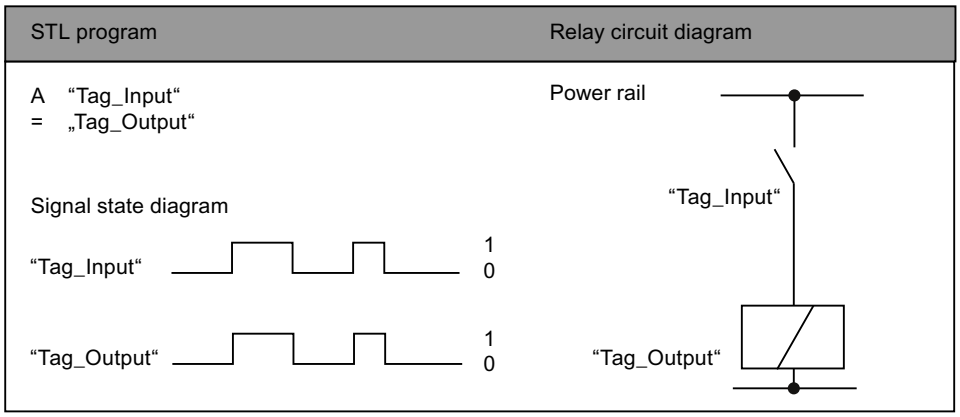
Parameters

The following table shows the parameters of the "Assignment" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand to which the current RLO is assigned.

Example

The following example shows how the instruction works:



See also

Overview of the valid data types (Page 2236)
Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)
Example of detecting the fill level of a storage area (Page 3136)
Example: Word logic instructions (Page 3140)
Example of a step sequence (Page 3142)
Master control relay (Page 2471)
Important notes on using MCR functions (Page 2472)
Basic information on the status word (Page 2453)
Setting the status bit for bit logic operations (Page 2455)

R: Reset

Description

Use the "Reset" instruction to reset the signal state of a specified operand to "0".

The instruction is only executed when the current result of logic operation (RLO) is "1". After execution of the instruction the specified operand is reset to "0". If there is a current RLO of "0", the signal state of the specified operand remains unchanged.

The execution of the "Reset" instruction is influenced by the Master Control Relay (MCR). If the MCR bit is set to "0", the signal state of the specified operand remains unchanged.

Syntax

Use the following syntax for the "Reset" instruction:

```
R <Operand>
```

Parameters

The following table shows the parameters of the "Reset" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand that is reset when RLO = "1".

Example

The following example shows how the instruction works:

```
STL                          Explanation
A "Tag_Input_1"              // Query the signal state of the operand for "1" and
                             AND with the current RLO.
A "Tag_Input_2"              // Query the signal state of the operand for "1" and
                             AND with the current RLO.
R "Tag_Output"               // Reset the operand to "0" if RLO is "1"
```

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Example: Bit logic instructions (Page 3134)

Example of detecting the direction of a conveyor belt (Page 3135)

Master control relay (Page 2471)

Important notes on using MCR functions (Page 2472)

Basic information on the status word (Page 2453)

Setting the status bit for bit logic operations (Page 2455)

S: Set

Description

Use the "Set" instruction to set the signal state of a specified operand to "1".

The instruction is only executed when the current result of logic operation (RLO) is "1". After the instruction executes, the specified operand is set to "1". If there is a current RLO of "0", the signal state of the specified operand remains unchanged.

The execution of the "Set" instruction is influenced by the Master Control Relay (MCR). If the MCR bit is set to "0", the signal state of the specified operand remains unchanged.

Syntax

Use the following syntax for the "Set" instruction:

```
S <Operand>
```

Parameters

The following table shows the parameters of the "Set" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Operand that is set when RLO = "1".

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input_1"	// Query the signal state of the operand for "1" and AND with the current RLO.
A "Tag_Input_2"	// Query the signal state of the operand for "1" and AND with the current RLO.
S "Tag_Output"	// Sets the operand to "1" if RLO = "1".

See also

Overview of the valid data types (Page 2236)
Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)
Example: Bit logic instructions (Page 3134)
Example of detecting the direction of a conveyor belt (Page 3135)
Master control relay (Page 2471)
Important notes on using MCR functions (Page 2472)
Basic information on the status word (Page 2453)
Setting the status bit for bit logic operations (Page 2455)

NOT: Invert RLO

Description

Use the "Invert RLO" instruction to invert the signal state of the result of logic operation (RLO). You can use the "Invert RLO" instruction anywhere, even within a logic operation.

Syntax

Use the following syntax for the "Invert RLO" instruction:

```
NOT
```

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input_1"	// Query the signal state of the "Tag_Input_1" operand for "1" and AND with the current RLO.
A "Tag_Input_2"	// Query the signal state of the "Tag_Input_2" operand for "1" and AND with the current RLO.
A "Tag_Input_3"	// Query the signal state of the "Tag_Input_3" operand for "1" and AND with the current RLO.
NOT	// Invert signal state of the RLO
= "Tag_Output"	// Assign signal state of the RLO to the "Tag_Output" operand.

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Input_1	1	1	1	0	0	0	1	0
Tag_Input_2	0	1	1	0	1	1	0	1
Tag_Input_3	0	0	1	0	0	1	1	1
Tag_Output	1	1	0	1	1	1	1	1

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting the status bit for bit logic operations (Page 2455)

SET: Set RLO to 1

Description

Use the "Set RLO to 1" instruction to set the current result of logic operation (RLO) to signal state "1".

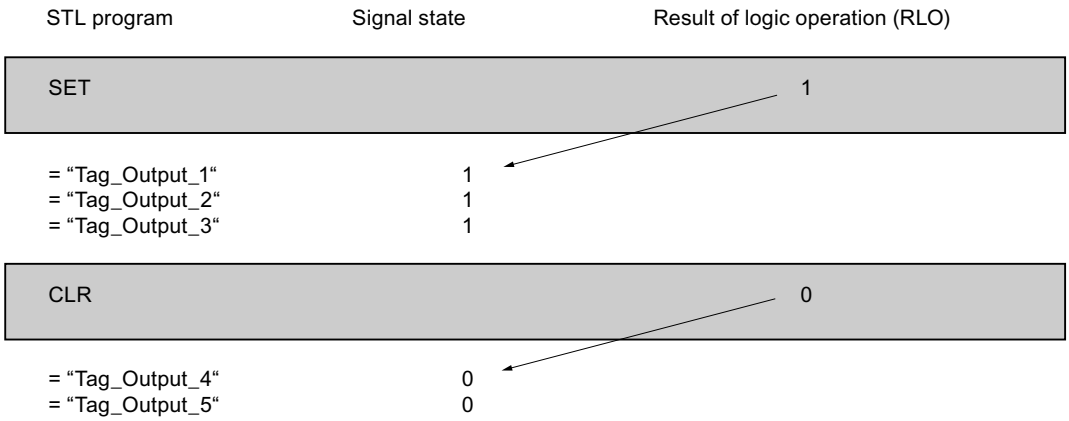
Syntax

Use the following syntax for the "Set RLO to 1" instruction:

```
SET
```

Example

The following example shows how the instruction works:



See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Setting the status bit for bit logic operations (Page 2455)

CLR: Reset RLO to 0

Description

Use the "Reset RLO to 0" instruction to set the current result of logic operation (RLO) to signal state "0".

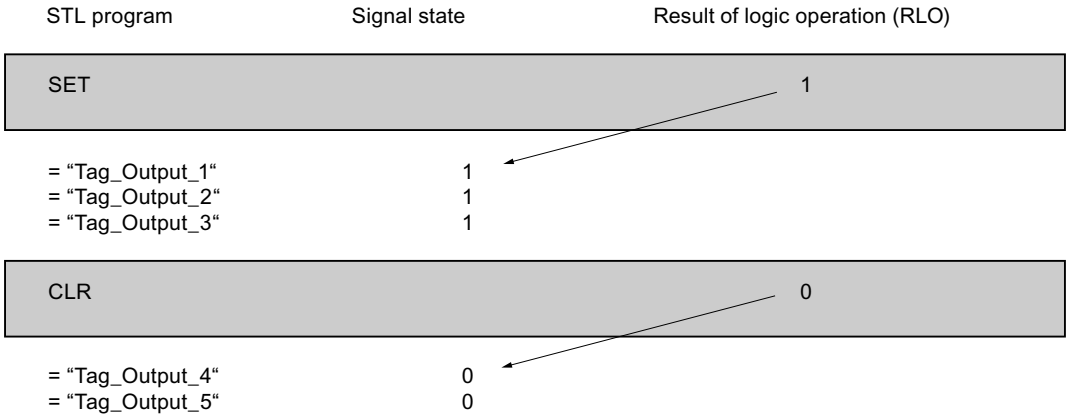
Syntax

Use the following syntax for the "Reset RLO to 0" instruction:

```
CLR
```

Example

The following example shows how the instruction works:



See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bit for bit logic operations (Page 2455)

Basic information on the status word (Page 2453)

SAVE: Save RLO in BR bit

Description

Use the "Save RLO in BR bit" instruction to save the result of logic operation (RLO) in the binary result (BR). When it executes, the instruction transfers the signal state of the current result of logic operation to the status bit BR. The instruction operates independent of the conditions and does not affect any other status bits.

Note

After executing the "Save RLO in BR bit" instruction, the BR bit can be modified again by the following instructions in the same or lower-level blocks.

Use the "Save RLO in BR bit" instruction to check the execution status of a block. If, for example, you use the instruction "Save RLO in BR bit" at the end of a block, the BR bit is set to the signal state of the current result of logic operation of the block.

The BR bit helps to implement the EN/ENO mechanism for block calls. For more information on this topic, refer to "See also".

Syntax

Use the following syntax for the "Save RLO in BR bit" instruction:

```
SAVE
```

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input"	// Query signal state of the Tag_Input operand for "1".
SAVE	// Transfer signal state of the RLO to the BR bit.
BEC	// End of block if RLO = "1".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting the status bit for bit logic operations (Page 2455)

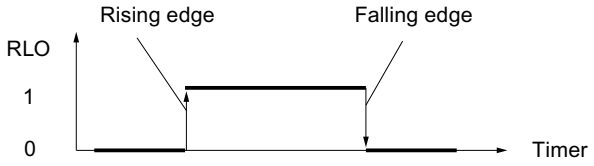
Example of the simulation of the EN/ENO mechanism in STL (Page 2452)

FN: Scan RLO for negative signal edge

Description

Use the "Scan RLO for negative signal edge" instruction to query a "1" to "0" change in the signal state of the result of logic operation (RLO). The instruction compares the current signal state of the result of logic operation (RLO) with the signal state of the previous query that is saved in the operand of the instruction. If the instruction detects a change in the RLO from "1" to "0", there is a negative, falling edge.

The following figure shows the change in the RLO in case of a falling and a rising edge:



The negative edge is queried each time the instruction executes. When the instruction detects a falling edge, the RLO is set to signal state "1" after edge evaluation. If no falling edge is detected, the signal state of the result of logic operation is "0".

Note

The operand for the edge evaluation must be located in bit memory or in a data block.

Syntax

Use the following syntax for the "Scan RLO for negative signal edge" instruction:

```
FN <Operand>
```

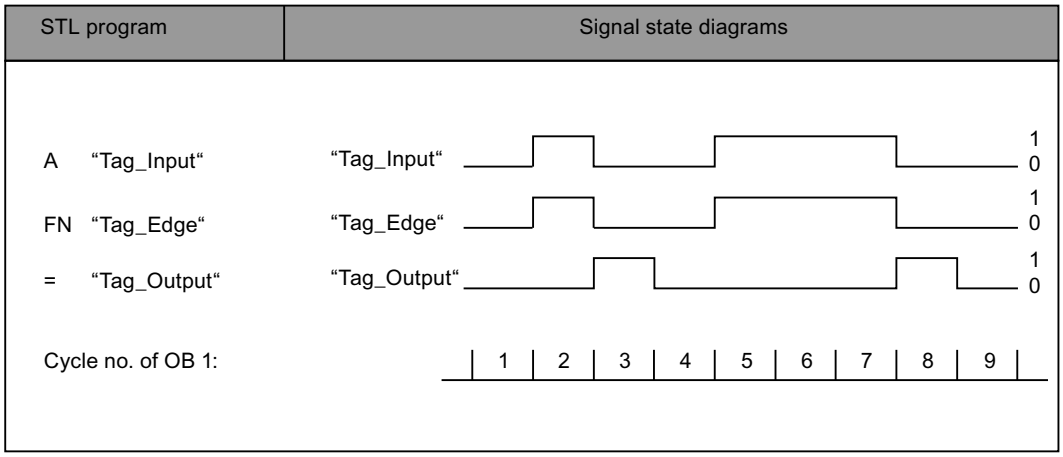
Parameters

The following table shows the parameters of the "Scan RLO for negative signal edge" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Edge memory bit in which the RLO of the last query is saved.

Example

The following example shows how the instruction works:



See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

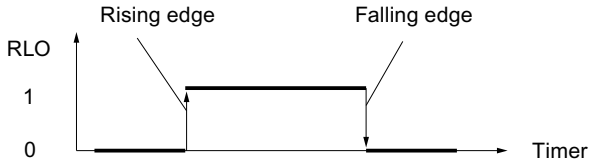
Setting the status bit for bit logic operations (Page 2455)

FP: Scan RLO for positive signal edge

Description

Use the "Scan RLO for positive signal edge" instruction to query a "0" to "1" change in the signal state of the result of logic operation (RLO). The instruction compares the current signal state of the result of logic operation (RLO) with the signal state of the previous query that is saved in the operand of the instruction. If the instruction detects a change in the RLO from "0" to "1", there is a positive, rising edge.

The following figure shows the change in the RLO in case of a falling and a rising edge:



The positive edge is queried each time the instruction executes. When the instruction detects a rising edge, the RLO after the edge evaluation is set to signal state "1". If no rising edge is detected, the signal state of the result of logic operation is "0".

Note

The operand for the edge evaluation must be located in bit memory or in a data block.

Syntax

Use the following syntax for the "Scan RLO for positive signal edge" instruction:

```
FP <Operand>
```

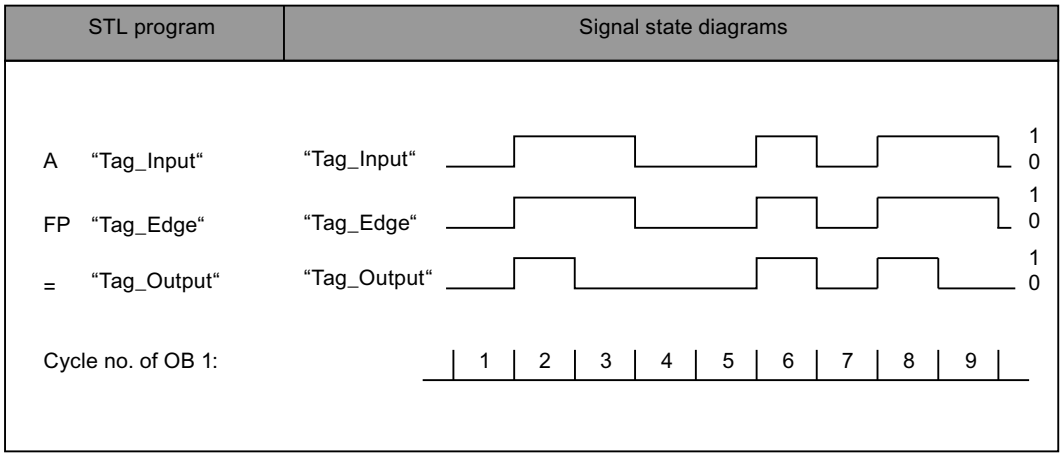
Parameters

The following table shows the parameters of the "Scan RLO for positive signal edge" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	BOOL	I, Q, M, D, L	Edge memory bit in which the RLO of the last query is saved.

Example

The following example shows how the instruction works:



See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Example of detecting the direction of a conveyor belt (Page 3135)

Basic information on the status word (Page 2453)

Setting the status bit for bit logic operations (Page 2455)

Timer operations

FR: Enable timer

Description

Use the instruction "Enable timer" to restart a timer. The instruction is executed with a positive edge and resets the internal edge trigger flag for the starting of a timer. During the processing of an instruction for starting the timer with a result of logic operation "1", the timer restarts even if no positive signal edge is present before the start instruction. If the result of logic operation is "0" before a start instruction, the resetting of the internal edge trigger flag does not affect the timer.

A change of the result of logic operation to "0" before the "Enable timer" instruction has no affect after the restart of the timer.

A valid time value in S5TIME format must be present in accumulator 1, even if the timer is not started during execution of a start function.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Syntax

Use the following syntax for the "Enable timer" instruction:

```
FR <Timer>
```

Parameters

The following table shows the parameters of the "Enable timer" instruction:

Operand	Declaration	Data type	Memory area	Description
<timer>	Input	TIMER	T	Enabled timer

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_EnableInput"	// Query operand for "1" and link with the current RLO according to AND.
FR "MyTimer"	// If the signal edge is positive, enable timer.
A "Tag_StartInput"	// Query operand for "1" and link with the current RLO according to AND.
L S5T#10s	// Load period of time (10s)
SP "MyTimer"	// Start pulse timer if there is a positive signal edge at the "Tag_StartInput" operand or at the "Tag_EnableInput" operand and the "Tag_StartInput" operand has signal state "1".
A "Tag_Reset"	// Query operand for "1" and link with the current RLO according to AND.
R "MyTimer"	// If RLO = "1", reset timer.
A "MyTimer"	// Query status of timer for "1".
= "Tag_TimerStatus"	// Assign timer status to the operand.
L "MyTimer"	// Load current time value of timer in accumulator 1.
T "Tag_TimerValue"	// Transfer current time value to operand "Tag_TimerValue".

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits in timer and counter instructions (Page 2457)

Basic information on the status word (Page 2453)

L: Load timer value (Page 5934)

L: Load timer value

Description

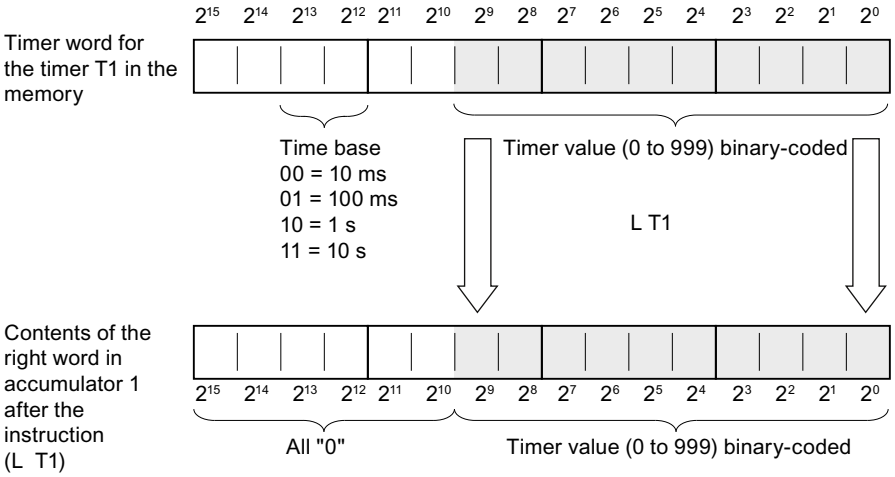
The "Load timer value" instruction loads the time value of a specified timer in accumulator 1. The binary-coded timer value is loaded without the timebase. Zeros are written to accumulator 1 instead of the timebase.

After the loading, the value in accumulator 1 corresponds to a positive number of the data type INT.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

The following figure is an example of how the time value is loaded in the accumulator:



Syntax

Use the following syntax for the "Load timer value" instruction:

```
L <Timer>
```

Parameters

The following table shows the parameters of the "Load timer value" instruction:

Operand	Declaration	Data type	Memory area	Description
<timer>	Input	TIMER	T	Timer whose current time value is loaded

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_EnableInput"	// Query operand for "1" and link with the current RLO according to AND.
FR "MyTimer"	// If the signal edge is positive, enable timer.
A "Tag_StartInput"	// Query operand for "1" and link with the current RLO according to AND.
L S5T#10s	// Load period of time (10s)
SP "MyTimer"	// Start pulse timer if there is a positive signal edge at the "Tag_StartInput" operand or at the "Tag_EnableInput" operand and the "Tag_StartInput" operand has signal state "1".
A "Tag_Reset"	// Query operand for "1" and link with the current RLO according to AND.
R "MyTimer"	// If RLO = "1", reset timer.
A "MyTimer"	// Query status of timer for "1".
= "Tag_TimerStatus"	// Assign timer status to the operand.
L "MyTimer"	// Load current time value of timer in accumulator 1.
T "Tag_TimerValue"	// Transfer current time value to operand "Tag_TimerValue".

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits in timer and counter instructions (Page 2457)

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

LC: Load BCD-coded timer value

Description

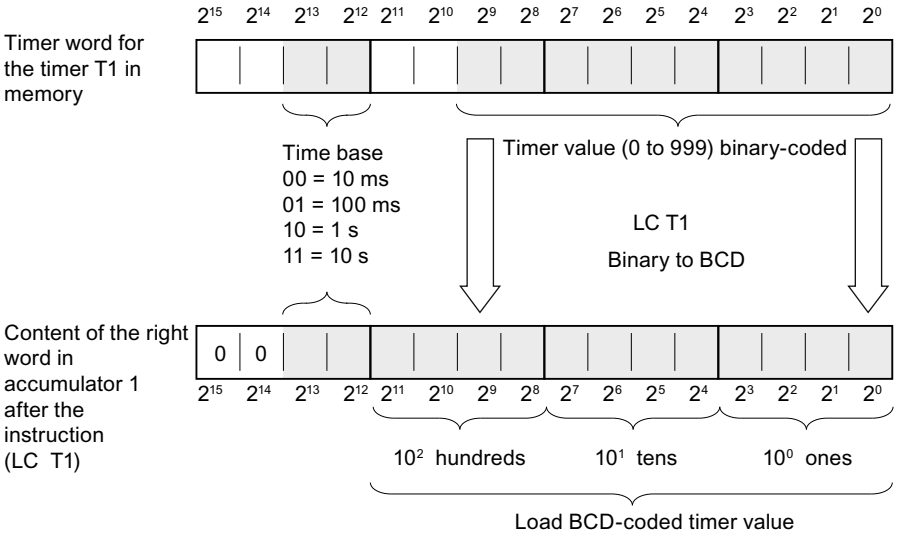
The "Load BCD-coded timer value" instruction loads a binary-coded timer value of a specified timer in BCD format in accumulator 1. The timebase is transferred to accumulator 1 during loading.

After loading, the value in accumulator 1 corresponds to a time duration in S5TIME format. The left word in accumulator 1 is padded with zeros.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

The following figure shows how the time value is loaded in the accumulator:



Syntax

Use the following syntax for the "Load BCD-coded timer value" instruction:

```
LC <Timer>
```

Parameters

The following table shows how the parameters of the "Load BCD-coded timer value" instruction:

Operand	Declaration	Data type	Memory area	Description
<timer>	Input	TIMER	T	Timer whose current BCD-coded timer value is loaded

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
A "Tag_EnableInput"          // Query operand for "1" and link with the current RLO
                             according to AND.
FR "MyTimer"                 // If the signal edge is positive, enable timer.
A "Tag_StartInput"           // Query operand for "1" and link with the current RLO
                             according to AND.
L S5T#10s                    // Load period of time (10s)
SP "MyTimer"                 // Start pulse timer if there is a positive signal
                             edge at the "Tag_StartInput" operand or at the
                             "Tag_EnableInput" operand and the "Tag_StartInput"
                             operand has signal state "1".
A "Tag_Reset"                // Query operand for "1" and link with the current RLO
                             according to AND.
R "MyTimer"                  // If RLO = "1", reset timer.
A "MyTimer"                  // Query status of timer for "1".
= "Tag_TimerStatus"          // Assign timer status to the operand.
LC "MyTimer"                 // Load current BCD-coded timer value of timer in
                             accumulator 1.
T "Tag_TimerValue"           // Transfer current time value to operand
                             "Tag_TimerValue".
```

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting status bits in timer and counter instructions (Page 2457)

Overview of the valid data types (Page 2236)

L: Load timer value (Page 5934)

R: Reset timer

Description

Use the "Reset timer" instruction to reset a specified timer to "0". The instruction is executed when the current result of logic operation (RLO) is "1". As long as the RLO is "1" before the instruction, the specified timer returns the value "0". The instruction also resets the time value and the time basis of the programmed period of time to "0".

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

Note

The "Reset timer" instruction does not reset the internal edge memory bit. To reset the internal edge memory bit, program the "Enable timer" instruction or execute an instruction to start the time with the signal state "0".

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Syntax

Use the following syntax for the "Reset timer" instruction:

```
R <Timer>
```

Parameters

The following table shows the parameters of the "Reset timer" instruction:

Operand	Declaration	Data type	Memory area	Description
<timer>	Input	TIMER	T	Timer, which is reset

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Reset"	// Query operand for "1" and link with the current RLO according to AND.
R "MyTimer"	// If RLO = "1", reset timer.
A "MyTimer"	// Query status of timer for "1".
= "Tag_TimerStatus"	// Assign timer status to the operand.
L "MyTimer"	// Load current time value of timer in accumulator 1.
T "Tag_TimerValue"	// Transfer current time value to operand "Tag_TimerValue".

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits in timer and counter instructions (Page 2457)

Basic information on the status word (Page 2453)

Overview of the valid data types (Page 2236)

L: Load timer value (Page 5934)

SP: Start pulse timer

Description

Use the "Start pulse timer" instruction to start a specified timer on a positive signal edge. If a positive signal edge is detected, the instruction is executed and the timer started. The timer then runs down with the period of time specified in accumulator 1, as long as the result of logic operation (RLO) before the instruction remains "1". If the RLO changes to "0" before the end of the period of time, the timer is stopped. The queries of the timer status for "1" return the query result "1", as long as the period of time is running.

The timer can be restarted with a positive signal edge before the "Start pulse timer" instruction or using the "Enable timer" instruction. This restart can be effected provided the timer is not reset.

The period of time in accumulator 1 is made up of a time value and a time base. If a specified timer is started with the "Start pulse timer" instruction, the time value is counted down according to the timebase. The timer runs until the count value is counted down to zero.

A valid period of time in S5TIME format must be present in accumulator 1, even if the timer is not started during execution of the instruction. If accumulator 1 does not contain a valid BCD

value, the synchronous error handling (STOP or OB122) is performed. The status bit OV is not influenced.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Syntax

Use the following syntax for the "Start pulse timer" instruction:

```
SP <Timer>
```

Parameters

The following table shows the parameters of the "Start pulse timer" instruction:

Operand	Declaration	Data type	Memory area	Description
<timer>	Input	TIMER	T	Timer that is started as pulse

For additional information on valid data types, refer to "See also".

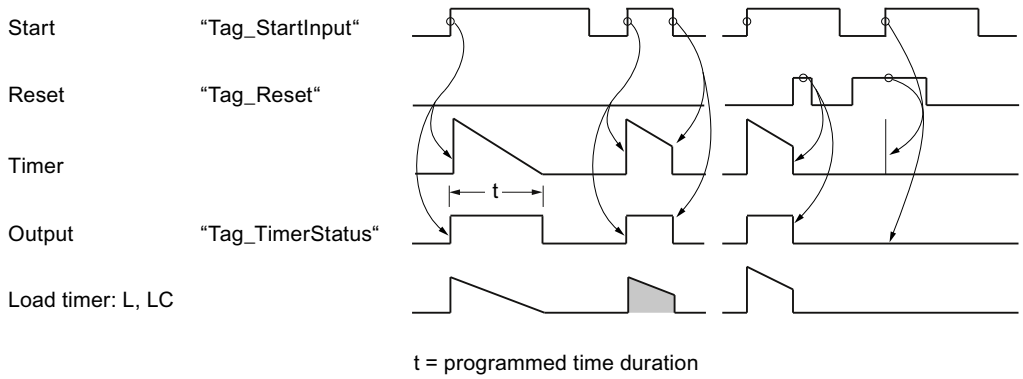
Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_StartInput"	// Query operand for "1" and link with the current RLO according to AND.
L S5T#10s	// Load period of time (10s)
SP "MyTimer"	// Start pulse timer if there is a positive signal edge at the "Tag_StartInput" operand or at the "Tag_EnableInput" operand and the "Tag_StartInput" operand has signal state "1".
A "Tag_Reset"	// Query operand for "1" and link with the current RLO according to AND.
R "MyTimer"	// If RLO = "1", reset timer and stop timing
A "MyTimer"	// Query status of timer for "1".
= "Tag_TimerStatus"	// Assign timer status to the operand.
L "MyTimer"	// Load current time value of timer in accumulator 1.

STL	Explanation
`T "Tag_TimerValue"`	`// Transfer current time value to operand "Tag_TimerValue".`
`LC "MyTimer"`	`// Load current time value in BCD format in accumulator 1.`
`T "Tag_TimerValue_BCD"`	`// Transfer time value in BCD format to operand "Tag_TimerValue_BCD"`

The figure below shows the timer diagram of the example:



See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits in timer and counter instructions (Page 2457)

Basic information on the status word (Page 2453)

Overview of the valid data types (Page 2236)

L: Load timer value (Page 5934)

SE: Start extended pulse timer

Description

Use the "Start extended pulse timer" to start a specified timer on a positive signal edge. If a positive signal edge is detected, the instruction is executed and the timer started. The timer then runs down with the period of time specified in accumulator 1, as long as the result of logic operation changes to "0" before the instruction. The queries of the timer status for "1" return the query result "1", as long as the period of time is running.

With each positive signal edge, the instruction restarts the timer with the programmed time duration, even if the timer has still not run down.

The period of time in accumulator 1 is made up of a time value and a time base. If a specified timer is started with the "Start extended pulse timer" instruction, the time value is counted down according to the timebase. The timer runs until the count value is counted down to zero.

A valid period of time in S5TIME format must be present in accumulator 1, even if the timer is not started during execution of the instruction. If accumulator 1 does not contain a valid BCD value, the synchronous error handling (STOP or OB 122) is performed. The status bit OV is not influenced.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Syntax

Use the following syntax for the "Start extended pulse timer" instruction:

```
SE <Timer>
```

Parameters

The following table shows the parameters of the "Start extended pulse timer" instruction:

Operand	Declaration	Data type	Memory area	Description
<timer>	Input	TIMER	T	Timer, which is started as extended pulse

For additional information on valid data types, refer to "See also".

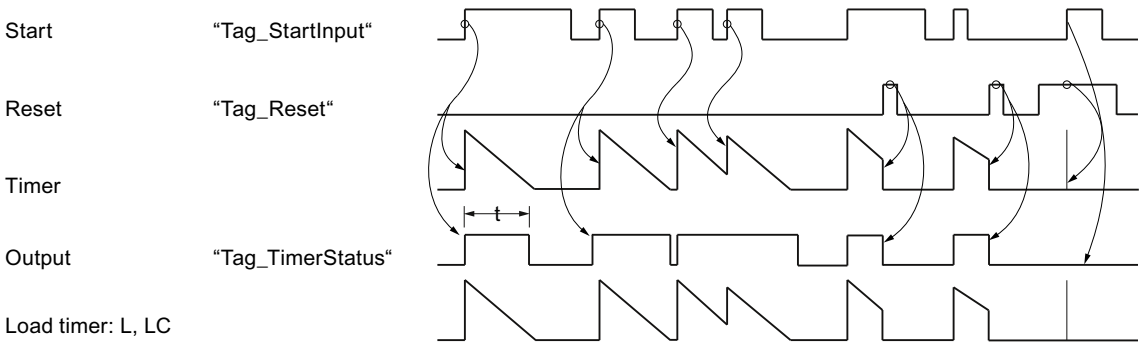
Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_StartInput"	// Query operand for "1" and link with the current RLO according to AND.
L S5T#10s	// Load period of time (10s)
SE "MyTimer"	// Start timer as extended pulse if there is a positive signal edge at the "Tag_StartInput" operand or at the "Tag_EnableInput" operand and the "Tag_StartInput" operand has signal state "1".
A "Tag_Reset"	// Query operand for "1" and link with the current RLO according to AND.
R "MyTimer"	// If RLO = "1", reset timer and stop timing

STL	Explanation
A "MyTimer"	// Query status of timer for "1".
= "Tag_TimerStatus"	// Assign timer status to the operand.
L "MyTimer"	// Load current time value of timer in accumulator 1.
T "Tag_TimerValue"	// Transfer current time value to operand "Tag_TimerValue".
LC "MyTimer"	// Load current time value in BCD format in accumulator 1.
T "Tag_TimerValue_BCD"	// Transfer time value in BCD format to operand "Tag_TimerValue_BCD"

The figure below shows the timer diagram of the example:



t = programmed time duration

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Example: Word logic instructions (Page 3140)

Setting status bits in timer and counter instructions (Page 2457)

Basic information on the status word (Page 2453)

Overview of the valid data types (Page 2236)

L: Load timer value (Page 5934)

SD: Start on-delay timer

Description

Use the "Start on-delay timer" instruction to start a specified timer on a positive signal edge. If a positive signal edge is detected, the instruction is executed and the timer started. The timer then runs down with the period of time specified in accumulator 1, as long as the result of logic operation (RLO) before the instruction remains "1". If the result of logic operation changes to "0" before the end of the period of time, the timer is stopped.

The queries of the timer status for "1" return the query result "1" if the time expired correctly and the current RLO before the instruction is "1". If the time has not expired or if the RLO changes to "0", the queries of the timer status for "1" return the query result "0".

The period of time in accumulator 1 is made up of a time value and a time base. If a specified timer is started with the "Start on-delay timer" instruction, the time value is counted down according to the timebase. The timer runs until the count value is counted down to zero.

A valid period of time in S5TIME format must be present in accumulator 1, even if the timer is not started during execution of the instruction. If accumulator 1 does not contain a valid BCD value, the synchronous error handling (STOP or OB 122) is performed. The status bit OV is not influenced.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Syntax

Use the following syntax for the "Start on-delay timer" instruction:

```
SD <Timer>
```

Parameters

The following table shows the parameters of the "Start on-delay timer" instruction:

Operand	Declaration	Data type	Memory area	Description
<timer>	Input	TIMER	T	Timer, which is started as on delay

For additional information on valid data types, refer to "See also".

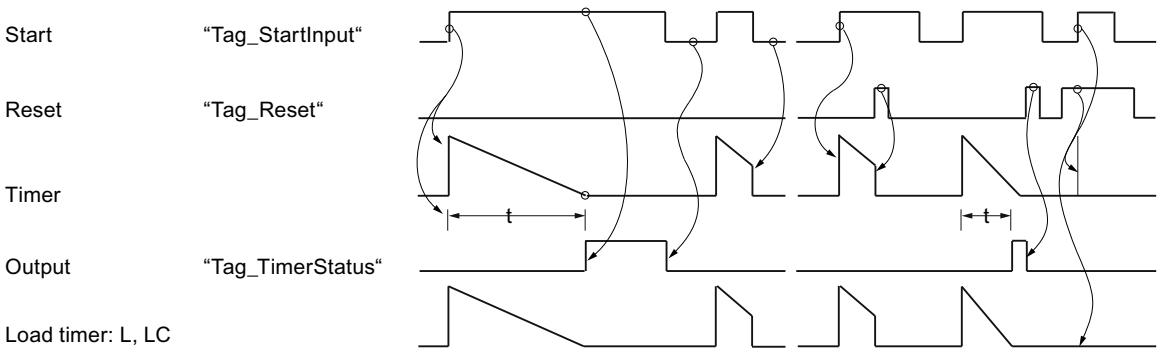
Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_StartInput"	// Query operand for "1" and link with the current RLO according to AND.
L S5T#10s	// Load period of time (10s)

STL	Explanation
SD "MyTimer"	// Start on-delay timer if there is a positive signal edge at the "Tag_StartInput" operand or at the "Tag_EnableInput" operand and the "Tag_StartInput" operand has signal state "1".
A "Tag_Reset"	// Query operand for "1" and link with the current RLO according to AND.
R "MyTimer"	// If RLO = "1", reset timer and stop timing
A "MyTimer"	// Query status of timer for "1".
= "Tag_TimerStatus"	// Assign timer status to the operand.
L "MyTimer"	// Load current time value of timer in accumulator 1.
T "Tag_TimerValue"	// Transfer current time value to operand "Tag_TimerValue".
LC "MyTimer"	// Load current time value in BCD format in accumulator 1.
T "Tag_TimerValue_BCD"	// Transfer time value in BCD format to the operand "Tag_TimerValue_BCD"

The figure below shows the timer diagram of the example:



t = programmed time duration

See also

Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)
Setting status bits in timer and counter instructions (Page 2457)
Basic information on the status word (Page 2453)
Overview of the valid data types (Page 2236)
L: Load timer value (Page 5934)

SS: Start retentive on-delay timer

Description

Use the "Start retentive on-delay timer" instruction to start a specified timer on a positive signal edge. If a positive signal edge is detected, the instruction is executed and the timer started. The timer then runs with the period of time specified in accumulator 1, even if the result of logic operation (RLO) changes to "0" before the instruction. When the time has run out, queries of the timer status for "1" return the query result "1". The query result is not hereby affected by the signal state of the current result of logic operation before the instruction.

With each positive signal edge, the instruction restarts the timer with the programmed time duration, even if the timer has still not run down.

The period of time in accumulator 1 is made up of a time value and a time base. If a specified timer is started with the "Start retentive on-delay timer" instruction, the time value is counted down according to the time base. The timer runs until the count value is counted down to zero.

A valid period of time in S5TIME format must be present in accumulator 1, even if the timer is not started during execution of the instruction. If accumulator 1 does not contain a valid BCD value, the synchronous error handling (STOP or OB 122) is performed. The status bit OV is not influenced.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Syntax

Use the following syntax for the "Start retentive on-delay timer" instruction:

```
SS <Timer>
```

Parameters

The following table shows the parameters of the "Start retentive on-delay timer" instruction:

Operand	Declaration	Data type	Memory area	Description
<timer>	Input	TIMER	T	Timer that is started as retentive on-delay

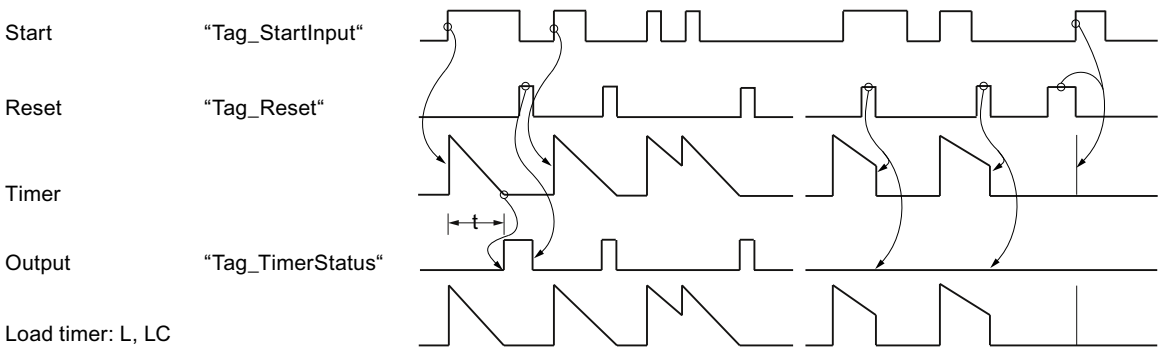
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_StartInput"	// Query operand for "1" and link with the current RLO according to AND.
L S5T#10s	// Load period of time (10s)
SS "MyTimer"	// Start retentive on-delay timer if there is a positive signal edge at the "Tag_StartInput" operand or at the "Tag_EnableInput" operand and the "Tag_StartInput" operand has signal state "1".
A "Tag_Reset"	// Query operand for "1" and link with the current RLO according to AND.
R "MyTimer"	// If RLO = "1", reset timer and stop timing
A "MyTimer"	// Query status of timer for "1".
= "Tag_TimerStatus"	// Assign timer status to the operand.
L "MyTimer"	// Load current time value of timer in accumulator 1.
T "Tag_TimerValue"	// Transfer current time value to operand "Tag_TimerValue".
LC "MyTimer"	// Load current time value in BCD format in accumulator 1.
T "Tag_TimerValue_BCD"	// Transfer time value in BCD format to the operand "Tag_TimerValue_BCD"

The figure below shows the timer diagram of the example:



t = programmed time duration

See also

Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)
Setting status bits in timer and counter instructions (Page 2457)
Basic information on the status word (Page 2453)
Overview of the valid data types (Page 2236)
L: Load timer value (Page 5934)

SF: Start off-delay timer

Description

Use the "Start off-delay timer" instruction to start a specified timer on a positive signal edge. If a negative signal edge is detected, the instruction is executed and the timer started. The timer then runs down with the period of time specified in accumulator 1. If the result of logic operation (RLO) changes to "1" before the end of the period of time, the timer is reset. The timer is only restarted if a negative signal edge is detected during the processing of the instruction.

The queries of the timer status for "1" return the query result "1" if the signal state of the result of logic operation is "1" at the execution of the instruction or if the programmed time is running. If the time does not elapse or the RLO is "0", queries of the timer status for "1" return the query result "0".

The period of time in accumulator 1 is made up of a time value and a time base. If a specified timer is started with the "Start off-delay timer" instruction, the time value is counted down according to the timebase. The timer runs until the count value is counted down to zero.

A valid period of time in S5TIME format must be present in accumulator 1, even if the timer is not started during execution of the instruction. If accumulator 1 does not contain a valid BCD value, the synchronous error handling (STOP or OB 122) is performed. The status bit OV is not influenced.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

Syntax

Use the following syntax for the "Start off-delay timer" instruction:

```
SF <Timer>
```

Parameters

The following table shows the parameters of the "Start off-delay timer" instruction:

Operand	Declaration	Data type	Memory area	Description
<timer>	Input	TIMER	T	Timer, which is started as off delay

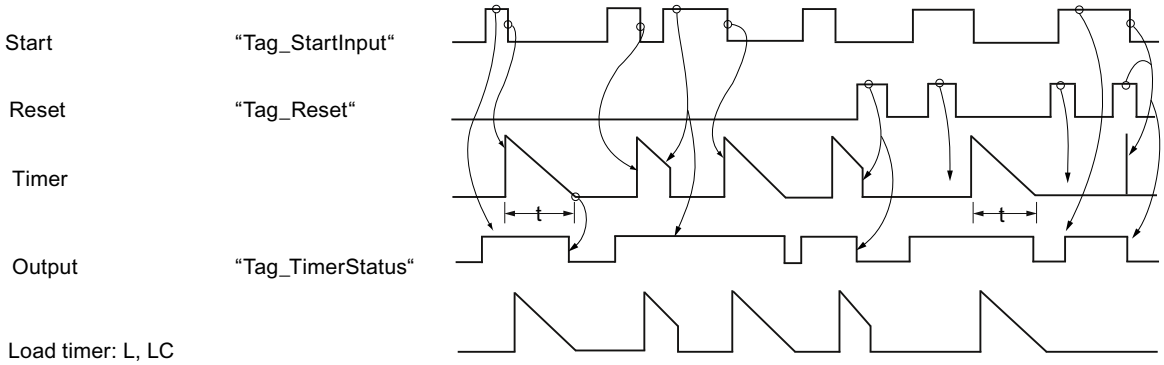
For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_StartInput"	// Query operand for "1" and link with the current RLO according to AND.
L S5T#10s	// Load period of time (10s)
SF "MyTimer"	// Start off-delay timer if there is a negative signal edge at the "Tag_StartInput" operand or at the "Tag_EnableInput" operand and the "Tag_StartInput" operand has signal state "1".
A "Tag_Reset"	// Query operand for "1" and link with the current RLO according to AND.
R "MyTimer"	// If RLO = "1", reset timer and stop timing
A "MyTimer"	// Query status of timer for "1".
= "Tag_TimerStatus"	// Assign timer status to the operand.
L "MyTimer"	// Load current time value of timer in accumulator 1.
T "Tag_TimerValue"	// Transfer current time value to operand "Tag_TimerValue".
LC "MyTimer"	// Load current time value in BCD format in accumulator 1.
T "Tag_TimerValue_BCD"	// Transfer time value in BCD format to operand "Tag_TimerValue_BCD"

The figure below shows the timer diagram of the example.



t = programmed time duration

See also

Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)
Setting status bits in timer and counter instructions (Page 2457)
Basic information on the status word (Page 2453)

Overview of the valid data types (Page 2236)

L: Load timer value (Page 5934)

Counter operations

FR: Enable counter

Description

Use the "Enable counter" instruction to execute the instructions "Set counter", "Count up" and "Count down" without the presence of a positive signal edge. During processing, the instruction resets the internal edge flag to the signal state "0". The query before the corresponding count instruction must continue to have signal state "1".

The "Enable counter" instruction is executed when the result of logic operation (RLO) before the instruction changes from "0" to "1".

A counter does not have to be enabled for the count instruction to be executed.

Syntax

Use the following syntax for the "Enable counter" instruction:

```
FR <Operand>
```

Parameters

The following table shows the parameters of the "Enable counter" instruction:

Operand	Declaration	Data type	Memory area	Description
<Operand>	Input	COUNTER	C	Counter that is enabled.

For more information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input_1"	// Query signal state for "1" and AND with the current "RLO.
FR "MyCounter"	// Enable counter
A "Tag_Input_2"	// Query signal state for "1" and AND with the current "RLO.
CD "MyCounter"	// Count down

STL	Explanation
A "MyCounter"	// Query counter status
= "Tag_Output"	// Assign query result to the operand.

In the example the counter ("MyCounter") is enabled when the signal state of the operand "Tag_Input_1" changes from "0" to "1". When the operand "Tag_Input_2" has signal state "1", the count value is decremented by one when the counter is enabled. When the signal state of the address "Tag_Input_2" is "0", the count value remains unchanged. The count value is also decremented by one regardless of the counter enabling, when the signal state of the address "Tag_Input_2" changes from "0" to "1".

The query of the counter status has the result "1" when the counter reading is greater than zero. If the counter reading is equal to zero, the result of the query is "0".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits in timer and counter instructions (Page 2457)

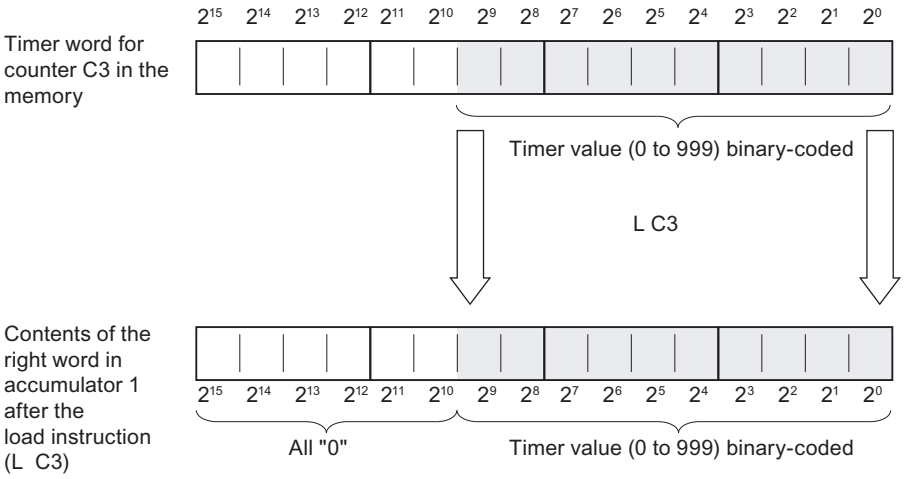
Basic information on the status word (Page 2453)

L: Load counter

Description

The "Load counter" instruction transfers the current value of a specified counter binary-coded to the right word of accumulator 1. The loaded count value is available as a 16-bit integer for further processing.

The following figure shows how the instruction works:



Syntax

Use the following syntax for the "Load counter" instruction:

```
L <Operand>
```

Parameters

The following table shows the parameters of the "Load counter" instruction:

Operand	Declaration	Data type	Memory area	Description
<Operand>	Input	COUNTER	C	Counter whose current value is loaded.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
L "MyCounter"	// Load current count value binary-coded in accumulator 1
L "Tag_Value"	// Move the counter value to accumulator 2.
	// Load value of the operand "Tag_Value" in accumulator 1
>=I	// Query whether the count value is greater than or equal to the value of the address "Tag_Value"
= "Tag_Output"	// Assign query result to the operand.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

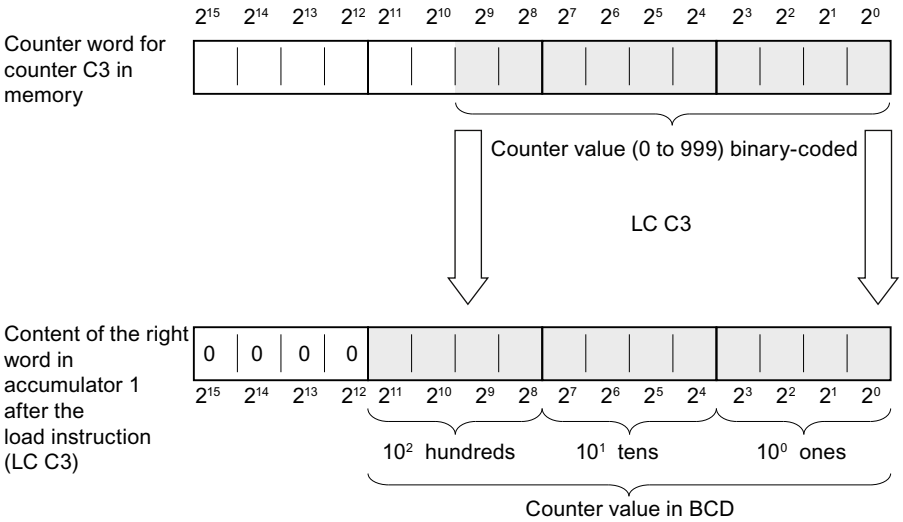
Setting status bits in timer and counter instructions (Page 2457)

LC: Load BCD-coded counter value

Description

The "Load BCD-coded counter value" instruction transfers the current value of a specified counter as BCD to the right-hand word of the accumulator 1. The loaded counter value is available in BCD form for further processing.

The following figure shows how the "Load BCD-coded counter value" instruction works:



Syntax

Use the following syntax for the "Load BCD-coded counter value" instruction:

```
LC <Operand>
```

Parameters

The following table shows the parameters of the "Load BCD-coded counter value" instruction:

Operand	Declaration	Data type	Memory area	Description
<Operand>	Input	COUNTER	C	Counter whose current value is loaded.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
LC "MyCounter"	// Load current BCD-coded counter value in accumulator 1.
T "Tag_Output"	// Transfer current counter value to the address "Tag_Output" and save.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting status bits in timer and counter instructions (Page 2457)

R: Reset counter

Description

Use the "Reset counter" instruction to reset the current count value of a specified counter to "0".

The instruction is only executed when the result of logic operation (RLO) before the instruction has signal state "1". As long as the RLO before the "Reset counter" instruction is "1", the queries of the counter status for "1" return the result "0".

When a counter is reset, the internal edge memory bit for the instructions "Set counter", "Count up" and "Count down" is not reset.

Syntax

Use the following syntax for the "Reset counter" instruction:

```
R <Operand>
```

Parameters

The following table shows the parameters of the "Reset counter" instruction:

Operand	Declaration	Data type	Memory area	Description
<Operand>	Input	COUNTER	C	Counter whose count value is reset to "0".

For more information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input"	// Query signal state for "1" and AND with the current "RLO.
R "MyCounter"	// Reset counter

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits in timer and counter instructions (Page 2457)

Basic information on the status word (Page 2453)

S: Set counter

Description

Use the "Set counter" instruction to set a specified counter to the value contained in accumulator 1 at the time of the query.

The "Set counter" instruction is executed when the result of logic operation before the instruction changes from "0" to "1". The positive signal edge can also be achieved with the "Enable counter" instruction that resets the internal edge memory bit to "0".

To set a counter, accumulator 1 must contain a valid count value. Valid count values are values in BCD format with three decades. Only positive count values are permitted; the count instruction cannot process negative count values. If accumulator 1 does not contain a valid BCD value, synchronous error handling (STOP or OB122) is executed. The status bit OV is not influenced.

Syntax

Use the following syntax for the "Set counter" instruction:

```
S <Operand>
```

Parameters

The following table shows the parameters of the "Set counter" instruction:

Operand	Declaration	Data type	Memory area	Description
<Operand>	Input	COUNTER	C	Counter that is set.

For more information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
STL                          Explanation
A "Tag_Input"                // Query signal state for "1" and AND with the current
                             "RLO.
L "Tag_CountValue"           // Load count value in accumulator 1.
S "MyCounter"                // Set counter to the value that is in accumulator 1
```

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits in timer and counter instructions (Page 2457)

Basic information on the status word (Page 2453)

CU: Count up

Description

Use the "Count up" instruction to increment the current count value by one when the result of logic operation (RLO) before the instruction changes from "0" to "1". The counter value continues to be incremented for each positive signal edge until the high limit of "999" is reached. When the limit is reached the RLO before the instruction no longer has any affect on the count value.

A positive signal edge is always required to execute the "Count up" instruction. The positive signal edge can also be achieved with the "Enable counter" instruction that resets the internal edge memory bit to "0".

Syntax

Use the following syntax for the "Count up" instruction:

```
CU <operand>
```

Parameters

The following table shows the parameters of the "Count up" instruction:

Operand	Declaration	Data type	Memory area	Description
<Operand>	Input	COUNTER	C	Counter, whose count value is incremented by one on a positive edge.

For more information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
`A "Tag_Input"`	`// Query signal state for "1" and AND with the current "RLO.`
`CU "MyCounter"`	`// Increment count value by one on a positive edge.`

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Example of detecting the fill level of a storage area (Page 3136)

Setting status bits in timer and counter instructions (Page 2457)

Basic information on the status word (Page 2453)

CD: Count down

Description

Use the "Count down" instruction to decrement the current count value by one when the result of logic operation (RLO) before the instruction changes from "0" to "1". The count value continues to be decremented for each positive signal edge until the low limit of "0" is reached. When the limit is reached the RLO before the instruction no longer has any affect on the count value. Counting does not occur when count value is negative.

A positive signal edge is always required to execute the "Count down" instruction. The positive signal edge can also be achieved with the "Enable counter" instruction that resets the internal edge memory bit to "0".

Syntax

The following syntax is used for the "Count down" instruction:

```
CD <operand>
```

Parameters

The following table shows the parameters of the "Count down" instruction:

Operand	Declaration	Data type	Memory area	Description
<Operand>	Input	COUNTER	C	Counter, whose count value is decremented by one on a positive edge.

For more information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input"	// Query signal state for "1" and AND with the current "RLO.
CD "MyCounter"	// Decrement counter value by one on a positive edge.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Example of detecting the fill level of a storage area (Page 3136)

Setting status bits in timer and counter instructions (Page 2457)

Basic information on the status word (Page 2453)

Comparator operations

? I: Compare 16-bit integers

Description

Use the "Compare 16-bit integers" instruction to compare the contents of the right word in accumulator 2 with the contents of the right word in accumulator 1. The contents of accumulators 1 and 2 are interpreted as 16-bit integers. The instruction performs the comparison depending on the specified comparison function.

The following table shows the available comparison functions and their meaning:

Comparison function	Meaning
==I	Compare 16-bit integers for "equivalence"
<>I	Compare 16-bit integers for "non-equivalence"
>I	Compare 16-bit integers for "greater than"
<I	Compare 16-bit integers for "less than"
>=I	Compare 16-bit integers for "greater than or equal to"
<=I	Compare 16-bit integers for "less than or equal to"

The result of the comparison influences the result of logic operation (RLO) and the status bits CC 1 and CC 0. If the condition of the comparison is satisfied, the RLO is set to signal state "1". When the condition of the comparison is not met, the RLO is set to signal state "0".

The following table shows how the various comparison functions affect the RLO:

Comparison function	RLO if ACCU 2 > ACCU 1	RLO if ACCU 2 = ACCU 1	RLO if ACCU 2 < ACCU 1
==I	0	1	0
<>I	1	0	1
>I	1	0	0
<I	0	0	1
>=I	1	1	0
<=I	0	1	1

The setting of status bits CC 1 and CC 0 depends on the relationship between the two values involved in the comparison.

The contents of accumulators 1 and 2 are not changed by execution of the comparison functions. After the processing of a comparison function you can evaluate the result of the logic operation with assignment and jump functions.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Input_1"              // Load value of the operand "Tag_Input_1" in
                             accumulator 1
L "Tag_Input_2"              // Shift the contents of accumulator 1 (Tag_Input_1)
                             to accumulator 2.
                             // Load value of the operand "Tag_Input_2" in
                             accumulator 1
>I                           // Compare whether the value of the operand
                             "Tag_Input_1" is greater than the value of the operand
                             "Tag_Input_2".
= "Tag_Output"               // Assign result of the comparison to the operand
                             "Tag_Output".
```

The following table shows how the instruction works using specific operand values:

Operand	Values		
Tag_Input_1	10	10	5
Tag_Input_2	5	10	10
Tag_Output	1	0	0

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Example of a step sequence (Page 3142)

Example of detecting the fill level of a storage area (Page 3136)

Setting the status bit for comparison instructions (Page 2460)

Basic information on the status word (Page 2453)

? D: Compare 32-bit integers

Description

Use the instruction "Compare 32-bit integers" to compare the contents of accumulator 2 (ACCU 2) with the contents of accumulator 1 (ACCU 1). The contents of accumulators 1 and 2 are interpreted as 32-bit integers. The instruction performs the comparison depending on the specified comparison function.

The following table shows the available comparison functions and their meaning:

Comparison function	Meaning
==D	Compare 32-bit integers for "equivalence"
<>D	Compare 32-bit integers for "non-equivalence"
>D	Compare 32-bit integers for "greater than"
<D	Compare 32-bit integers for "less than"
>=D	Compare 32-bit integers for "greater than or equal to"
<=D	Compare 32-bit integers for "less than or equal to"

The result of the comparison influences the result of logic operation (RLO) and the status bits CC 1 and CC 0. If the condition of the comparison is satisfied, the RLO is set to signal state "1". If the condition of the comparison is not met, the RLO is reset to signal state "0".

The following table shows how the various comparison functions affect the RLO:

Comparison function	RLO if ACCU 2 > ACCU 1	RLO if ACCU 2 = ACCU 1	RLO if ACCU 2 < ACCU 1
==D	0	1	0
<>D	1	0	1
>D	1	0	0
<D	0	0	1
>=D	1	1	0
<=D	0	1	1

The setting of status bits CC 1 and CC 0 depends on the relationship between the two values involved in the comparison.

The contents of accumulators 1 and 2 are not changed by execution of the comparison functions. After the processing of a comparison function, you can evaluate the result of the logic operation with assignment and jump functions.

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Input_1"`	`// Load value of the operand "Tag_Input_1" in accumulator 1`
`L "Tag_Input_2"`	`// Contents of accumulator 1 (Tag_Input_1) shifted to accumulator 2.`
	`// Load value of the operand "Tag_Input_2" in accumulator 1`
`>D`	`// Compare whether the value of the operand "Tag_Input_1" is greater than the value of the operand "Tag_Input_2".`
`= "Tag_Output"`	`// Assign result of the comparison to the operand "Tag_Output".`

The following table shows how the instruction works using specific operand values:

Operand	Values		
Tag_Input_1	150 000	150 000	100 000
Tag_Input_2	100 000	150 000	150 000
Tag_Output	1	0	0

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bit for comparison instructions (Page 2460)

Basic information on the status word (Page 2453)

? R: Compare floating-point numbers

Description

Use the instruction "Compare floating-point numbers" to compare the contents of accumulator 2 (ACCU 2) with the contents of accumulator 1 (ACCU 1). The contents of accumulators 1 and 2 are interpreted as floating-point numbers. For the comparison, accumulators 1 and 2 must contain valid floating-point numbers. The instruction performs the comparison depending on the specified comparison function.

The following table shows the available comparison functions and their meaning:

Comparison function	Meaning
==R	Compare floating-point numbers for "equivalence"
<>R	Compare floating-point numbers for "non-equivalence"
>R	Compare floating-point numbers for "greater than"
<R	Compare floating-point numbers for "less than"
>=R	Compare floating-point numbers for "greater than or equal to"
<=R	Compare floating-point numbers for "less than or equal to"

The result of the comparison influences the result of logic operation (RLO) and the status bits CC 1 and CC 0. If the condition of the comparison is satisfied, the RLO is set to signal state "1". If the condition of the comparison is not met, the RLO is reset to signal state "0".

The following table shows how the various comparison functions affect the RLO:

Comparison function	RLO if ACCU 2 > ACCU 1	RLO if ACCU 2 = ACCU 1	RLO if ACCU 2 < ACCU 1
==R	0	1	0
<>R	1	0	1
>R	1	0	0
<R	0	0	1

Comparison function	RLO if ACCU 2 > ACCU 1	RLO if ACCU 2 = ACCU 1	RLO if ACCU 2 < ACCU 1
>=R	1	1	0
<=R	0	1	1

The setting of status bits CC 1 and CC 0 depends on the relationship between the two values involved in the comparison.

Note

During a comparison of invalid floating-point numbers, the CPU sets the result of logic operation to "0" and the status bits CC 0, CC 1, OVand OS to "1".

The contents of accumulators 1 and 2 are not changed by execution of the comparison functions. After the processing of a comparison function you can evaluate the result of the logic operation with assignment and jump functions.

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Input_1"`	`// Load value of the operand "Tag_Input_1" in accumulator 1`
`L "Tag_Input_2"`	`// Shift the contents of accumulator 1 (Tag_Input_1) to accumulator 2.`
	`// Load value of the operand "Tag_Input_2" in accumulator 1`
`>R`	`// Compare whether the value of the operand "Tag_Input_1" is greater than the value of the operand "Tag_Input_2".`
`= "Tag_Output"`	`// Assign result of the comparison to the operand "Tag_Output".`

The following table shows how the instruction works using specific operand values:

Operand	Values		
Tag_Input_1	10 515	10 515	5 232
Tag_Input_2	5 232	10 515	10 515
Tag_Output	1	0	0

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bit for comparison instructions (Page 2460)

Basic information on the status word (Page 2453)

Math functions

Integers

+I: Add integers (16 bit)

Description

Use the instruction "Add integers (16 bit)" to add the values in the right words of accumulators 1 and 2. The instruction interprets these values as 16-bit integers.

The instruction saves the sum in the right word of accumulator 1. The left word of accumulator 1 remains unchanged.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the sum is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

In case of controllers with two accumulators the content of accumulator 2 remains unchanged after processing of the instruction.

In case of controllers with four accumulators the contents of accumulator 3 are copied to accumulator 2 and the contents of accumulator 4 are copied to accumulator 3. The contents of accumulator 4 remain unchanged after the instruction has executed.

Syntax

The following syntax is used for the "Add integers (16 bit)" instruction:

```
+I
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load first value for addition.
L "Tag_Value_2"	// Load second value for addition.
+I	// Add values
T "Tag_Result"	// Transfer sum to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Example of calculating an equation (Page 3139)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

-I: Subtract integers (16 bit)

Description

Use the "Subtract integers (16 bit)" instruction to subtract the value of the right word of accumulator 1 from the value of the right word of accumulator 2. The instruction interprets the values of accumulators 1 and 2 as 16-bit integers.

The instruction saves the difference in the right word of accumulator 1. The left word of accumulator 1 remains unchanged.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the difference is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

In case of controllers with two accumulators the content of accumulator 2 remains unchanged after processing of the instruction.

In case of controllers with four accumulators the contents of accumulator 3 are copied to accumulator 2 and the contents of accumulator 4 are copied to accumulator 3. The contents of accumulator 4 remain unchanged after the instruction has executed.

Syntax

The following syntax is used for the "Subtract integers (16 bit)" instruction:

```
-I
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load first value of subtraction into accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load second value for addition in accumulator 1.
-I	// Subtract value of accumulator 1 from value of accumulator 2.

STL	Explanation
T "Tag_Result"	// Transfer difference to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

*I: Multiply integers (16 bit)

Description

Use the instruction "Multiply integers (16 bit)" to multiply the values in the right words of accumulators 1 and 2. The instruction interprets these values as 16-bit integers.

The instruction saves the product in accumulator 1 as 32-bit integer (DINT).

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the product is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

In case of controllers with two accumulators the content of accumulator 2 remains unchanged after processing of the instruction.

In case of controllers with four accumulators the contents of accumulator 3 are copied to accumulator 2 and the contents of accumulator 4 are copied to accumulator 3. The contents of accumulator 4 remain unchanged after the instruction has executed.

Syntax

The following syntax is used for the "Multiply integers (16 bit)" instruction:

```
*I
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load first value for multiplication.
L "Tag_Value_2"	// Load second value for multiplication.
*I	// Multiply values
T "Tag_Result"	// Transfer product to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Example of calculating an equation (Page 3139)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

/I: Divide integers (16 bit)

Description

Use the "Divide integers (16 bit)" instruction to divide the value of the right word of accumulator 2 by the value of the right word of accumulator 1. The instruction interprets the values of the accumulators as 16-bit integers.

The instruction provides two results: the quotients and the remainder of the division. The quotient is the whole number result of the division. The instruction stores the whole number in the right word of accumulator 1. The instruction stores the remainder of the division in the left word of accumulator 1. If the dividend is negative, the remainder of the division is also negative.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the quotient is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

In the case of a division by zero, the instruction reports the value as quotient and in each case the value zero as remainder. In this case, status bits CC 0, CC 1, OV and OS are set to signal state "1".

In case of controllers with two accumulators the content of accumulator 2 remains unchanged after processing of the instruction.

In case of controllers with four accumulators the contents of accumulator 3 are copied to accumulator 2 and the contents of accumulator 4 are copied to accumulator 3. The contents of accumulator 4 remain unchanged after the instruction has executed.

Syntax

The following syntax is used for the "Divide integers (16 bit)" instruction:

```
/I
```

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Value_1"`	`// Load dividend in accumulator 1.`

STL	Explanation
L "Tag_Value_2"	// Move dividend to accumulator 2.
	// Load divisor in accumulator 1.
/I	// Divide the value of accumulator 2 by the value of accumulator 1.
T "Tag_Result"	// Transfer the result to operand "Tag_Result".

Before processing:

The following table shows the contents of the right words in accumulators 1 and 2 before executing the "Divide integers (16 bit)" instruction:

Accumulator	Value
Accumulator 2	13
Accumulator 1	4

After processing

The following table shows the contents of accumulator 1 after executing the "Divide integers (16 bit)" instruction:

Accumulator 1	Value
Bit 0 to 15 (right word)	3
Bit 16 to 31 (left word)	1

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Example of calculating an equation (Page 3139)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

+D: Add double integers (32 bit)

Description

Use the "Add double integers (32 bit)" instruction to add the values in accumulators 1 and 2. The instruction interprets these values as 32-bit integers. The instruction saves the sum in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the sum is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

In case of controllers with two accumulators the content of accumulator 2 remains unchanged after processing of the instruction.

In case of controllers with four accumulators the contents of accumulator 3 are copied to accumulator 2 and the contents of accumulator 4 are copied to accumulator 3. The contents of accumulator 4 remain unchanged after the instruction has executed.

Syntax

The following syntax is used for the "Add double integers (32 bit)" instruction:

```
+D
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load first value for addition.
L "Tag_Value_2"	// Load second value for addition.
+D	// Add values
T "Tag_Result"	// Transfer sum to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

-D: Subtract double integers (32 bit)

Description

Use the "Subtract double integers (32 bit)" instruction to subtract the value of the right word of accumulator 1 from the value of the right word of accumulator 2. The instruction interprets the values of accumulators 1 and 2 as 32-bit integers. The instruction saves the result of the subtraction in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the difference is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

In case of controllers with two accumulators the content of accumulator 2 remains unchanged after processing of the instruction.

In case of controllers with four accumulators the contents of accumulator 3 are copied to accumulator 2 and the contents of accumulator 4 are copied to accumulator 3. The contents of accumulator 4 remain unchanged after the instruction has executed.

Syntax

The following syntax is used for the "Subtract double integers (32 bit)" instruction:

```
-D
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load first value of subtraction into accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load the second value of subtraction into accumulator 1.
-D	// Subtract value of accumulator 1 from value of accumulator 2.
T "Tag_Result"	// Transfer difference to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

*D: Multiply double integers (32 bit)

Description

Use the "Multiply double integers (32 bit)" instruction to multiply the values in accumulators 1 and 2. The instruction interprets these values as 32-bit integers. The instruction saves the product in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the product is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

In case of controllers with two accumulators the content of accumulator 2 remains unchanged after processing of the instruction.

In case of controllers with four accumulators the contents of accumulator 3 are copied to accumulator 2 and the contents of accumulator 4 are copied to accumulator 3. The contents of accumulator 4 remain unchanged after the instruction has executed.

Syntax

The following syntax is used for the "Multiply double integers (32 bit)" instruction:

```
*D
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load first value for multiplication.
L "Tag_Value_2"	// Load second value for multiplication.
*D	// Multiply values
T "Tag_Result"	// Transfer product to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

/D: Divide double integers (32 bit)

Description

Use the "Divide double integers (32 bit)" instruction to divide the value of accumulator 2 by the value of accumulator 1. The instruction interprets the values of accumulators 1 and 2 as as 32-bit integers. The instruction saves the result of the division (quotient) in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the quotient is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

In the case of a division by zero, the instruction reports the value zero as quotient. In this case, status bits CC 0, CC 1, OV and OS are set to signal state "1".

In case of controllers with two accumulators the content of accumulator 2 remains unchanged after processing of the instruction.

In case of controllers with four accumulators the contents of accumulator 3 are copied to accumulator 2 and the contents of accumulator 4 are copied to accumulator 3. The contents of accumulator 4 remain unchanged after the instruction has executed.

Syntax

The following syntax is used for the "Divide double integers (32 bit)" instruction:

```
/D
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load dividend in accumulator 1.
L "Tag_Value_2"	// Move dividend to accumulator 2.
	// Load divisor in accumulator 1.
/D	// Divide the value of accumulator 2 by the value of accumulator 1.
T "Tag_Result"	// Transfer quotients to the operand "Tag_Result".

Before processing:

The following table shows the contents of accumulators 1 and 2 before executing the "Divide double integers (32 bit)" instruction:

Accumulator	Value
Accumulator 2	13
Accumulator 1	4

After processing

The following table shows the contents of accumulator 1 after executing the "Divide double integers (32 bit)" instruction:

Accumulator	Value
Accumulator 1	3

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

+: Add constants

Description

Use the instruction "Add constants" to add the value of a specified constant to the contents of accumulator 1. You can specify the constant value as a 16-bit or 32-bit integer. The instruction stores the result in accumulator 1. In the case of addition of 16-bit integers, the instruction influences only the right word of accumulator 1.

To execute the addition of a 16-bit integer as DINT math function, enter "L#" before the constant. The "Add constants" instruction permits the use of negative numbers.

The instruction does not affect any status bits.

Syntax

Use the following syntax for the "Add constants" instruction:

```
+ <constant>
```

Parameters

The following table shows the parameters of the "Add constants" instruction:

Parameters	Data type	Description
<constant>	INT, DINT	Constants to be added

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1".
+ 25	// Increment the value of accumulator 1 by 25.
T "Tag_Result_1"	// Transfer contents of accumulator 1 to the operand "Tag_Result_1".
L "Tag_Value_2"	// Load value of the operand "Tag_Value_2" in accumulator 1

STL	Explanation
L "Tag_Value_3"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_3" in accumulator 1
+ -100	// Decrement the value of accumulator 1 by 100.
<I	// Check whether the value of accumulator 2 is less than the value of accumulator 1
JC NEXT	// If the result of the query is "1", jump to jump label NEXT .
	// If the result of the query is "0", execute the next instruction.
T "Tag_Result_2"	Transfer contents of accumulator 1 to the operand "Tag_Result_2".
L "Tag_Value_4"	// Load value of the operand "Tag_Value_4".
+ L#20	// Increment content of accumulator 1 by 20.
	// The addition is done as a DINT calculation
T "Tag_Result_3"	// Transfer contents of accumulator 1 to the operand "Tag_Result_3".
NEXT: L "Tag_Value_5"	// Jump label "NEXT"
	// Load value of the operand "Tag_Value_5".
+D	// Add values of accumulators 1 and 2.
T "Tag_Result_4"	// Transfer sum to the operand "Tag_Result_4".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

Setting the status bits in instructions with floating-point numbers (Page 2459)

INC: Increment

Description

Use the "Increment" instruction to increment the value in accumulator 1 by the value specified as parameter of the instruction. The specified value can be in a value range from 0 to 255. The value is incremented only in the right byte of accumulator 1. No transfer takes place to be byte located to the left.

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of accumulator 2, 3 and 4 remain unchanged.

If the parameter value is outside the permitted range of value, the result will be corrupted without generating an error message.

Syntax

Use the following syntax for the "Increment" instruction:

```
INC <constant>
```

Parameters

The following table shows the parameters of the "Increment" instruction:

Parameters	Format	Description
<constant>	8-bit integer	Value by which the value is incremented.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value"                // Load value of the operand in accumulator 1.
INC 5                        // Increment value of accumulator 1 by 5.
T "Tag_Result"               // Transfer contents of accumulator 1 to the operand.
```

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

DEC: Decrement

Description

Use the "Decrement" instruction to decrement the value in accumulator 1 by the value specified as parameter of the instruction. The specified value can be in a value range from 0 to 255. The value is decremented only in the right byte of accumulator 1. No transfer takes place to be byte located on the left side.

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of accumulator 2, 3 and 4 remain unchanged.

If the parameter value is outside the permitted range of value, the result will be corrupted without generating an error message.

Syntax

Use the following syntax for the "Decrement" instruction:

```
DEC <constant>
```

Parameters

The following table shows the parameters of the "Decrement" instruction:

Parameters	Format	Description
<constant>	8-bit integer	Value by which the value is decremented.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value"                // Load value of the operand in accumulator 1.
DEC 5                        // Decrement the value of accumulator 1 by 5.
T "Tag_Result"               // Transfer contents of accumulator 1 to the operand.
```

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

MOD: Return remainder of division

Description

Use the "Return remainder of division" instruction to divide the value of accumulator 2 by the value of accumulator 1 and save the remainder of the division.

The instruction interprets the values of accumulators 1 and 2 as 32-bit integers. The instruction stores the remainder of the division in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the remainder of division is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

In the case of a division by zero, the instruction returns the value zero as result. In this case, status bits CC 0, CC 1, OV and OS are set to signal state "1".

In case of controllers with two accumulators the content of accumulator 2 remains unchanged after processing of the instruction.

In case of controllers with four accumulators the contents of accumulator 3 are copied to accumulator 2 and the contents of accumulator 4 are copied to accumulator 3. The contents of accumulator 4 remain unchanged after the instruction has executed.

Syntax

Use the following syntax for the "Return remainder of division" instruction:

```
MOD
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load dividend in accumulator 1.
L "Tag_Value_2"	// Move dividend to accumulator 2.
	// Load divisor in accumulator 1.
MOD	// Divide the value of accumulator 2 by the value of accumulator 1.
T "Tag_Result"	// Transfer remainder of division to the operand "Tag_Result".

Before processing:

The following table shows the contents of accumulators 1 and 2 before executing the instruction "Return remainder of division":

Accumulator	Value
Accumulator 2	13
Accumulator 1	4

After processing

The following table shows the contents of accumulator 1 after executing the instruction "Return remainder of division":

Accumulator	Value
Accumulator 1	1

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

Floating-point numbers

+R: Add floating-point numbers

Description

Use the "Add floating-point numbers" instruction to add the values contained in accumulators 1 and 2. The instruction interprets these values as floating-point numbers. The instruction saves the sum in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the sum is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If an invalid floating-point number or the sign for infinity (+/- ∞) is entered, the instruction writes an invalid value to accumulator 1 and sets the status bits CC 0, CC 1, OV and OS to "1".

In case of controllers with two accumulators the content of accumulator 2 remains unchanged after processing of the instruction.

In case of controllers with four accumulators the contents of accumulator 3 are copied to accumulator 2 and the contents of accumulator 4 are copied to accumulator 3. The contents of accumulator 4 remain unchanged after the instruction has executed.

Syntax

The following syntax is used for the instruction "Add floating-point numbers":

```
+R
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load first value for addition.
L "Tag_Value_2"	// Load second value for addition.
+R	// Add values
T "Tag_Result"	// Transfer sum to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bits in instructions with floating-point numbers (Page 2459)

Basic information on the status word (Page 2453)

-R: Subtract floating-point numbers

Description

Use the "Subtract floating-point numbers" instruction to subtract the value of accumulator 1 from the value of accumulator 2. The instruction interprets the values of accumulators 1 and 2 as floating-point numbers. The instruction stores the difference in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the difference is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If an invalid floating-point number or the sign for infinity (+/- ∞) is entered, the instruction writes an invalid value to accumulator 1 and sets the status bits CC 0, CC 1, OV and OS to "1".

In case of controllers with two accumulators the content of accumulator 2 remains unchanged after processing of the instruction.

In case of controllers with four accumulators the contents of accumulator 3 are copied to accumulator 2 and the contents of accumulator 4 are copied to accumulator 3. The contents of accumulator 4 remain unchanged after the instruction has executed.

Syntax

Use the following syntax for the instruction "Subtract floating-point numbers":

```
-R
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load first value of subtraction into accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load the second value of subtraction into accumulator 1.
-R	// Subtract value of accumulator 1 from value of accumulator 2.
T "Tag_Result"	// Transfer difference to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bits in instructions with floating-point numbers (Page 2459)

Basic information on the status word (Page 2453)

*R: Multiply floating-point numbers

Description

Use the "Multiply floating-point numbers" instruction to multiply the values contained in accumulators 1 and 2. The instruction interprets these values as floating-point numbers. The instruction saves the product in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the product is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If an invalid floating-point number or the sign for infinity (+/- ∞) is entered, the instruction writes an invalid value to accumulator 1 and sets the status bits CC 0, CC 1, OV and OS to "1".

In case of controllers with two accumulators the content of accumulator 2 remains unchanged after processing of the instruction.

In case of controllers with four accumulators the contents of accumulator 3 are copied to accumulator 2 and the contents of accumulator 4 are copied to accumulator 3. The contents of accumulator 4 remain unchanged after the instruction has executed.

Syntax

Use the following syntax for the "Multiply floating-point numbers" instruction:

```
*R
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load first value for multiplication.
L "Tag_Value_2"	// Load second value for multiplication.
*R	// Multiply values
T "Tag_Result"	// Transfer product to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bits in instructions with floating-point numbers (Page 2459)

Basic information on the status word (Page 2453)

/R: Divide floating-point numbers

Description

Use the "Divide floating-point numbers" instruction to divide the value of accumulator 2 by the value of accumulator 1. The instruction interprets the values of accumulators 1 and 2 as floating-point numbers. The instruction saves the result of the division (quotient) in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the quotient is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

In the case of a division by zero, the instruction reports the value zero as quotient. In this case, status bits CC 0, CC 1, OV and OS are set to signal state "1".

If an invalid floating-point number or the sign for infinity (+/- ∞) is entered, the instruction writes an invalid value to accumulator 1 and sets the status bits CC 0, CC 1, OV and OS to "1".

In case of controllers with two accumulators the content of accumulator 2 remains unchanged after processing of the instruction.

In case of controllers with four accumulators the contents of accumulator 3 are copied to accumulator 2 and the contents of accumulator 4 are copied to accumulator 3. The contents of accumulator 4 remain unchanged after the instruction has executed.

Syntax

Use the following syntax for the "Divide floating-point numbers" instruction:

```
/R
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load dividend in accumulator 1.
L "Tag_Value_2"	// Move dividend to accumulator 2.
	// Load divisor in accumulator 1.
/R	// Divide the value of accumulator 2 by the value of accumulator 1.
T "Tag_Result"	// Transfer quotients to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bits in instructions with floating-point numbers (Page 2459)

Basic information on the status word (Page 2453)

ABS: Form absolute value

Description

Use the "Form absolute value" instruction to form the absolute value of the value in accumulator 1. The instruction interprets the contents of accumulator 1 as a floating-point number and sets the sign of the mantissa to "0". The result is stored in accumulator 1.

The "Form absolute value" instruction does not affect any status bits.

Syntax

Use the following syntax for the "Form absolute value" instruction:

```
ABS
```

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value"                // Load floating-point number into accumulator 1
ABS                          // Form absolute value
T "Tag_Result"               // Transfer the result to operand "Tag_Result".
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	-1.5E+02
Tag_Result	1.5E+02

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Advanced floating-point numbers

SQR: Form square

Description

Use the "Form square" instruction to calculate the square of a floating-point number located in accumulator 1. The result is stored in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If the floating-point numbers are invalid, the instruction sets the status bits CC 0, CC 1, OV and OS to "1".

The instruction influences only the contents of accumulator 1. The contents of the other accumulators remain unchanged.

Syntax

Use the following syntax for the "Form square" instruction:

```
SQR
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
SQR	// Form square
	// Store result in accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bits in instructions with floating-point numbers (Page 2459)

Basic information on the status word (Page 2453)

SQRT: Form square root

Description

Use the "Form square root" instruction to form the square root of the floating-point number located in accumulator 1. The result is stored in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If accumulator 1 has a value less than zero or an invalid floating-point number, the instruction writes an invalid value to accumulator 1 and sets the status bits CC 0, CC 1, OV and OS to "1".

If accumulator 1 has the value "0", the instruction returns the value "0".

The instruction influences only the contents of accumulator 1. The contents of the other accumulators remain unchanged.

Syntax

Use the following syntax for the "Form square root" instruction:

```
SQRT
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
+R	// Add the values of accumulator 1 and 2.
	// Store sum in accumulator 1.
SQRT	// Get square root
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bits in instructions with floating-point numbers (Page 2459)

Basic information on the status word (Page 2453)

EXP: Form exponential value

Description

Use the "Form exponential value" to form the exponent from the base e (e = 2.718282) and the floating-point number in accumulator 1. The instruction stores the result in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If the floating-point numbers are invalid, the instruction sets the status bits CC 0, CC 1, OV and OS to "1".

The instruction influences only the contents of accumulator 1. The contents of the other accumulators remain unchanged.

Syntax

Use the following syntax for the "Form exponential value" instruction:

```
EXP
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
EXP	// Form exponential value
	// Store result in accumulator 1.
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bits in instructions with floating-point numbers (Page 2459)

Basic information on the status word (Page 2453)

LN: Form natural logarithm

Description

Use the "Form natural logarithm" instruction to calculate the natural logarithm of the value in accumulator 1 to base e (e = 2.718282).

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If accumulator 1 has a value less than or equal to zero, the instruction writes an invalid value to accumulator 1 and sets the status bits CC 0, CC 1, OV and OS to "1". If the floating-point numbers are invalid, the instruction also sets the status bits CC 0, CC 1, OV and OS to "1".

The instruction influences only the contents of accumulator 1. The contents of the other accumulators remain unchanged.

Syntax

Use the following syntax for the "Form natural logarithm" instruction:

```
LN
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
LN	// Form natural logarithm
	// Store result in accumulator 1.
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bits in instructions with floating-point numbers (Page 2459)

Basic information on the status word (Page 2453)

SIN: Form sine value

Description

Use the "Form sine value" instruction to calculate the sine of the angle in accumulator 1. The angle must be specified in radian measure and exist as floating-point number in accumulator 1. The instruction stores the result in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1". If the floating-point numbers are invalid, the instruction sets the status bits CC 0, CC 1, OV and OS to "1".

The instruction influences only the contents of accumulator 1. The contents of the other accumulators remain unchanged.

Syntax

Use the following syntax for the "Form sine value" instruction:

```
SIN
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
SIN	// Form sine value
	// Store result in accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bits in instructions with floating-point numbers (Page 2459)

Basic information on the status word (Page 2453)

ASIN: Form arcsine value

Description

Use the "Form arcsine value" instruction to calculate the size of the angle from the value in accumulator 1 that corresponds to this value. The value of the accumulator 1 must exist as floating-point number in a value range from "-1" to "1".

The result of the instruction is returned in radian measure and stored in accumulator 1. Depending on the value of accumulator 1, the result can be in a value range from $-\pi/2$ to $+\pi/2$ (π= 3.14159).

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If accumulator 1 has a value that exceeds the permitted value range, the instruction writes an invalid floating-point number to accumulator 1 and sets the status bits CC 0, CC 1, OV and OS to "1". If the floating-point numbers are invalid, the instruction also sets the status bits CC 0, CC 1, OV and OS to "1".

The instruction influences only the contents of accumulator 1. The contents of the other accumulators remain unchanged.

Syntax

Use the following syntax for the "Form arcsine value" instruction:

```
ASIN
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
ASIN	// Form arcsine value
	// Store result in accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bits in instructions with floating-point numbers (Page 2459)

Basic information on the status word (Page 2453)

COS: Form cosine value

Description

Use the "Form cosine value" instruction to calculate the cosine of the angle in accumulator 1. The size of the angle must be specified in radian measure and exist as floating-point number in accumulator 1. The instruction stores the result in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1". If the floating-point numbers are invalid, the instruction sets the status bits CC 0, CC 1, OV and OS to "1".

The instruction influences only the contents of accumulator 1. The contents of the other accumulators remain unchanged.

Syntax

Use the following syntax for the "Form cosine value" instruction:

```
COS
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
COS	// Form cosine value
	// Store result in accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bits in instructions with floating-point numbers (Page 2459)

Basic information on the status word (Page 2453)

ACOS: Form arccosine value

Description

Use the "Form arccosine value" instruction to calculate the size of the angle from the value in accumulator 1 that corresponds to this value. The value of the accumulator 1 must exist as floating-point number in a value range from "-1" to "1".

The result of the instruction is returned in radian measure and stored in accumulator 1. Depending on the value of accumulator 1, the result can be in a value range from "0" to "π" (π= 3.14159).

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If accumulator 1 has a value that exceeds the permitted value range, the instruction writes an invalid floating-point number to accumulator 1 and sets the status bits CC 0, CC 1, OV and OS to "1". If the floating-point numbers are invalid, the instruction also sets the status bits CC 0, CC 1, OV and OS to "1".

The instruction influences only the contents of accumulator 1. The contents of the other accumulators remain unchanged.

Syntax

Use the following syntax for the "Form arccosine value" instruction:

```
ACOS
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
ACOS	// Form arccosine value
	// Store result in accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bits in instructions with floating-point numbers (Page 2459)

Basic information on the status word (Page 2453)

TAN: Form tangent value

Description

Use the "Form tangent value" instruction to calculate the tangent of the angle in accumulator 1. The size of the angle must be specified in radian measure and exist as floating-point number in accumulator 1. The instruction stores the result in accumulator 1.

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1". If the floating-point numbers are invalid, the instruction sets the status bits CC 0, CC 1, OV and OS to "1".

The instruction influences only the contents of accumulator 1. The contents of the other accumulators remain unchanged.

Syntax

Use the following syntax for the "Form tangent value" instruction:

```
TAN
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
TAN	// Form tangent value
	// Store result in accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bits in instructions with floating-point numbers (Page 2459)

Basic information on the status word (Page 2453)

ATAN: Form arctangent value

Description

Use the "Form arctangent value" instruction to calculate the size of the angle from the value in accumulator 1 that corresponds to this value. The value of the accumulator 1 must exist as floating-point number.

The result of the instruction is returned in radian measure and stored in accumulator 1. Depending on the value of accumulator 1, the result can be in a value range from $-\pi/2$ to $+\pi/2$ (π= 3.14159).

After the instruction has executed, the status bits CC 0 and CC 1 indicate whether the result is negative, zero or positive. If the number is outside the permitted number range, the status bits OV and OS are set to "1".

If accumulator 1 has a value that exceeds the permitted value range, the instruction writes an invalid floating-point number to accumulator 1 and sets the status bits CC 0, CC 1, OV and OS to "1". If the floating-point numbers are invalid, the instruction also sets the status bits CC 0, CC 1, OV and OS to "1".

The instruction influences only the contents of accumulator 1. The contents of the other accumulators remain unchanged.

Syntax

Use the following syntax for the instruction "Form arctangent value":

```
ATAN
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
ATAN	// Form arctangent value
	// Store result in accumulator 1.

STL	Explanation
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bits in instructions with floating-point numbers (Page 2459)

Basic information on the status word (Page 2453)

Load and transfer

Load

L: Load

Description

The "Load" instruction loads the content of a specified operand in accumulator 1. The operand to be loaded can be addressed as a byte, word or double word in one of the following memory areas:

- Process image of the inputs and outputs (I, Q)
- Bit memory (M)
- Temporary local data (L)
- Data blocks (DB, DI)
- Pointer
- I/O (PI)
- Timer (T)
- Counter (C)

The memory area of accumulator 1 is organized in bytes and is 32 bits wide.

The instruction writes the contents of an operand that is to be loaded in byte format right aligned into accumulator 1. The remaining bytes in accumulator 1 are filled with "0".

The instruction writes the contents of an operand that is to be loaded in word format into the right word of accumulator 1. The higher addressed byte is hereby transferred into the right byte (bits 0 to 7) of accumulator 1. The lower addressed byte is written to the left of it. The remaining bytes in the left word of accumulator 1 are filled with "0".

The instruction writes the contents of an operand that is to be loaded in double word format into the 32 bits of accumulator 1. The highest addressed byte is hereby transferred to the right byte (bits 0 to 7) of accumulator 1. The lowest addressed byte is written in bits 24 to 31.

The following table shows, as an example, how the contents of accumulator 1 are changed by the loading of operands in byte, word and double word format:

Instruction	Accumulator 1							
	31 . . .	. . . 24	23 . . .	. . . 16	15 . . .	. . . 8	7 . . .	. . . 0
Loading of a byte: L MB10	0000	0000	0000	0000	0000	0000	<MB10>	
Loading of a word: L MW10	0000	0000	0000	0000	<MB10>		<MB11>	
Load a double word: L MD10	<MB10>		<MB11>		<MB12>		<MB13>	

During the loading of the specified operand the previous contents of accumulator 1 are moved to accumulator 2. The instruction "Load" hereby transfers the complete contents of accumulator 1 to accumulator 2. The previous contents of accumulator 2 are lost in the process.

In case of controllers with four accumulators, the contents of accumulators 3 and 4 are not influenced by the instruction.

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

Syntax

Use the following syntax for the "Load" instruction:

```
L <Operand>
```

Parameters

The following table shows the parameters of the "Load" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	Bit strings, integers, timers, REAL, DATE, TOD, CHAR	I, Q, PI, M, L, DB, DI, T, C, pointer, parameter	Operand whose contents are loaded.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.
L "Tag_Value_3"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_3" in accumulator 1
+R	// Add the values of accumulator 1 and 2.
	// Store sum in accumulator 1.
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Example of detecting the fill level of a storage area (Page 3136)

Example of calculating an equation (Page 3139)

Example: Word logic instructions (Page 3140)

Example of a step sequence (Page 3142)

Basic information on the status word (Page 2453)

Setting status bits (Page 2455)

L STW: Load status word in accumulator 1

Description

Use the "Load status word in accumulator 1" instruction to load the status word in accumulator 1. When it executes, the instruction writes the individual status bits to the right word of accumulator 1. The remaining bits of accumulator 1 are written with "0".

The following table shows the contents of the accumulator after executing the "Load status word in accumulator 1" instruction:

	Accumulator 1									
Bit number	31-9	8	7	6	5	4	3	2	1	0
Content	0	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC

The instruction is always executed independently of the signal state of the status bit.

Note

The controllers of the S7-300 series (except S7-318) do not load the status bits /FC, STA and OR into accumulator 1. These places are written with "0".

Syntax

Use the following syntax for the "Load status word in accumulator 1" instruction:

```
L STW
```

Example

The following example shows how the instruction works:

STL	Explanation
L STW	// Load status word in accumulator 1.
T "Tag_STW"	// Transfer the contents of accumulator 1 (status bits) to the operand "Tag_STW".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting status bits (Page 2455)

LAR1: Load AR1 with contents of accumulator 1

Description

Use the "Load AR1 with contents of accumulator 1" instruction to load address register 1 (AR1) with the contents of accumulator 1. The value of accumulator 1 must match the format of an area pointer (POINTER).

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Syntax

Use the following syntax for the "Load AR1 with contents of accumulator 1" instruction:

```
LAR1
```

Example

The following example shows how the instruction works:

STL	Explanation
L P#10.0	// Load pointer (P#10.0) in accumulator 1.
LAR1	// Load address register 1 with contents of accumulator 1.
L MW [AR1,P#4.0]	// Load MW14 in accumulator 1.
L "Tag_Value"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load content of the operand "Tag_Value" in accumulator 1.
>I	// Compare whether the value of accumulator 2 is greater than the value of accumulator 1.
A I [AR1,P#2.1]	// Check whether signal state of bit I12.1 is "1" and link AND with current RLO.
= "Tag_Output"	// If the condition is satisfied (RLO = "1"), set the operand "Tag_Output" to "1".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting status bits (Page 2455)

LAR1 <D>: Load AR1 with double word or area pointer

Description

Use the "Load AR1 with double word or area pointer" instruction to load address register 1 (AR1) with the contents of a double word or an area internal or area crossing pointer. The contents of the double word must match the format of an area pointer.

The pointer and the double word can address one of the following memory areas:

- Bit memory (M)
- Temporary local data (L)
- Data blocks (DB, DI)

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Syntax

Use the following syntax for the instruction "Load AR1 with double word or area pointer":

```
LAR1 <D>
```

Parameters

The following table shows the parameters of the "Load AR1 with double word or area pointer" instruction:

Parameters	Declaration	Data type	Memory area	Description
<D>	Input	DWORD, POINTER	D, M, L	Operand whose contents are loaded.

Example

The following example shows how the instruction works:

STL	Explanation
LAR1 P#10.0	// Load address register 1 with contents of the area internal pointer P#10.0.
L MW [AR1,P#4.0]	// Load MW14 in accumulator 1.
L "Tag_Value"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load content of the operand "Tag_Value" in accumulator 1.
>I	// Compare whether the value of accumulator 2 is greater than the value of accumulator 1.
A I [AR1,P#2.1]	// Check whether signal state of bit I12.1 is "1" and link AND with current RLO.
= "Tag_Output"	// If the condition is satisfied (RLO = "1"), set the operand "Tag_Output" to "1".
LAR1 MD20	// Load address register 1 with contents of the MD20 (MD20 = P#30.0)
A I [AR1,P#2.1]	// Check whether signal state of bit I32.1 is "1" and link AND with current RLO.

See also

Overview of the valid data types (Page 2236)
Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)
Basic information on the status word (Page 2453)
Setting status bits (Page 2455)

LAR1 AR2: Load AR1 with contents of AR2

Description

Use the "Load AR1 with contents of AR2" instruction to load the contents of address register 2 (AR2) into address register 1 (AR1).

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Syntax

Use the following syntax for the "Load AR1 with contents of AR2" instruction:

```
LAR1 AR2
```

Example

The following example shows how the instruction works:

STL	Explanation
LAR2 P#20.0	// Load address register 2 with contents of the area internal pointer P#20.0.
LAR1 AR2	// Copy contents of address register 2 into address register 1.
L MW [AR1,P#4.0]	// Load MW24 in accumulator 1.
T "Tag_2"	// Transfer accumulator 1 to "Tag_2".

See also

Overview of the valid data types (Page 2236)
Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)
Basic information on the status word (Page 2453)
Setting status bits (Page 2455)

LAR2: Load AR2 with contents of accumulator 1

Description

Use the "Load AR2 with contents of accumulator 1" instruction to load address register 2 (AR2) with the contents of accumulator 1. The value of accumulator 1 must match the format of an area pointer (POINTER).

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Syntax

Use the following syntax for the "Load AR2 with contents of accumulator 1" instruction:

```
LAR2
```

Example

The following example shows how the instruction works:

STL	Explanation
L P#10.0	// Load pointer (P#10.0) in accumulator 1.
LAR2	// Load address register 2 with contents of accumulator 1.
L MW [AR2,P#4.0]	// Load MW14 in accumulator 1.
L "Tag_Value"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load content of the operand "Tag_Value" in accumulator 1.
>I	// Compare whether the value of accumulator 2 is greater than the value of accumulator 1.
A I [AR2,P#2.1]	// Check whether signal state of bit I12.1 is "1" and link AND with current RLO.
= "Tag_Output"	// If the condition is satisfied (RLO = "1"), set the operand "Tag_Output" to "1".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting status bits (Page 2455)

LAR2 <D>: Load AR2 with double word or area pointer

Description

Use the "Load AR2 with double word or area pointer" instruction to load address register 2 (AR2) with the contents of a double word or an area-internal or cross-area pointer. The contents of the double word must match the format of an area pointer (POINTER).

The pointer and the double word can address one of the following memory areas:

- Bit memory (M)
- Temporary local data (L)
- Data blocks (DB, DI)

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Syntax

Use the following syntax for the "Load AR2 with double word or area pointer" instruction:

```
LAR2 <D>
```

Parameters

The following table shows the parameters of the "Load AR2 with double word or area pointer" instruction:

Parameters	Declaration	Data type	Memory area	Description
<D>	Input	DWORD, POINTER	D, M, L	Operand whose contents are loaded.

Example

The following example shows how the instruction works:

```
STL                          Explanation
LAR2 P#10.0                  // Load address register 2 with contents of the area
                             internal pointer P#10.0.
L MW [AR2,P#4.0]             // Load MW14 in accumulator 1.
L "Tag_Value"                // Shift the contents of accumulator 1 to accumulator
                             2.
                             // Load content of the operand "Tag_Value" in
                             accumulator 1.
>I                           // Compare whether the value of accumulator 2 is
                             greater than the value of accumulator 1.
```

STL	Explanation
A I [AR2,P#2.1]	// Check whether signal state of bit I12.1 is "1" and link AND with current RLO.
= "Tag_Output"	// If the condition is satisfied (RLO = "1"), set the operand "Tag_Output" to "1".
LAR2 MD20	// Load address register 2 with contents of the MD20 (MD20 = P#30.0)
A I [AR2,P#2.1]	// Check whether signal state of bit I32.1 is "1" and link AND with current RLO.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting status bits (Page 2455)

Transfer

T: Transfer

Description

Use the "Transfer" instruction to transfer the contents of accumulator 1 in byte, word or double word format to a specified operand. The contents of accumulator 1 are not changed by this instruction. The specified operand must be addressed in one of the following memory areas:

- Process image of the inputs and outputs (I, Q)
- Bit memory (M)
- Temporary local data (L)
- Data blocks (DB, DI)
- I/O (PQ)

The number of bytes to be transferred depends on the format of the specified operand. If the operand exists in byte format, the instruction transfers the contents of the bits 0 to 7 of accumulator 1.

If the operand exists in word format, the instruction transfers the contents of the word that is right-aligned in accumulator 1. The right byte of accumulator 1 is transferred to the higher addressed byte of the operand. The bits 8 to 15 of the accumulator 1 are transferred to the lower byte of the operand.

If the specified operand is available in double word format, the instruction transfers the entire contents of accumulator 1. The right byte of accumulator 1 is transferred to the highest addressed byte of the operand. The bits 24 to 31 of accumulator 1 are transferred to the lowest addressed byte of the operand.

The following table shows an example of how the contents of accumulator 1 are transferred in byte, word and double word format:

Instruction	Accumulator 1			
Bits	31 24	23 16	15 8	7 0
Contents of accumulator 1	byte (n)	byte (n+1)	byte (n+2)	byte (n+3)
Transfer of a byte				byte (n+3)
Transfer of a word			byte (n+2)	byte (n+3)
Transferring a double word	byte (n)	byte (n+1)	byte (n+2)	byte (n+3)

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The execution of the "Transfer" instruction is influenced by the Master Control Relay (MCR). When the MCR bit is set to "0", the value "0" is transferred to the specified operand.

Syntax

Use the following syntax for the "Transfer" instruction:

```
T <Operand>
```

Parameters

The following table shows the parameters of the "Transfer" instruction:

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Output	Bit strings, integers, timers, REAL, DATE, TOD, CHAR	I, Q, PQ, M, L, DB, DI	Operand to which the contents of accumulator 1 are transferred.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.

STL	Explanation
T "Tag_Result_1"	// Transfer product (contents of Accumulator 1) to the operand "Tag_Result_1".
L "Tag_Value_3"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_3" in accumulator 1
+R	// Add the values of accumulator 1 and 2.
	// Store sum in accumulator 1.
T "Tag_Result_2"	// Transfer sum (contents of Accumulator 1) to the operand "Tag_Result_2"

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Example of calculating an equation (Page 3139)

Example of a step sequence (Page 3142)

Master control relay (Page 2471)

Important notes on using MCR functions (Page 2472)

Basic information on the status word (Page 2453)

Setting status bits (Page 2455)

T STW: Transfer accumulator 1 to status word

Description

Use the "Transfer accumulator 1 to status word" instruction to transfer bits 0 to 8 of accumulator 1 to the status word.

The following table shows the bits in accumulator 1 that will overwrite status bits with processing of the instruction:

	Accumulator 1								
Bit number	8	7	6	5	4	3	2	1	0
Status bits	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC

The instruction is always executed independently of the signal state of the status bit.

Note

The bits of the status word /FC, STA and OR are not loaded by the "Transfer accumulator 1 to status word" instruction. Only bits 1, 4, 5, 6, 7 and 8 are loaded to the corresponding bit positions of the lower value word of accumulator 1.

Syntax

Use the following syntax for the "Transfer accumulator 1 to status word" instruction:

```
T STW
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_STW"	// Load value of the operand "Tag_STW" in accumulator 1.
T STW	// Transfer bits 0 to 8 of accumulator 1 to status word

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting status bits (Page 2455)

CAR: Switch AR1 and AR2

Description

Use the instruction "Switch AR1 and AR2" to switch the contents of address registers 1 (AR1) and 2 (AR2).

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Syntax

Use the following syntax for the "Switch AR1 and AR2" instruction:

```
CAR
```

Example

The following example shows how the instruction works:

STL	Explanation
LAR1 P#10.0	// Load address register 1 with contents of the area internal pointer P#10.0.
LAR2 P#20.0	// Load address register 2 with contents of the area internal pointer P#20.0.
CAR	// Switch contents of the address registers.
L MD [AR1,P#2.0]	// Load MD22 in accumulator 1.
T MD [AR2,P#2.0]"	// Transfer MD12 too accumulator 2.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting status bits (Page 2455)

TAR1: Transfer AR1 to accumulator 1

Description

Use the "Transfer AR1 to accumulator 1" instruction to transfer the contents of address register 1 (AR1) to accumulator 1. The contents of accumulator 1 are shifted to accumulator 2.

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

Syntax

Use the following syntax for the "Transfer AR1 to accumulator 1" instruction:

```
TAR1
```

Example

The following example shows how the instruction works:

STL	Explanation
LAR1 P#15.0	// Load contents of the area-internal pointer P#15.0 into address register 1.
TAR1	// Transfer contents of address register 1 (P#15.0) to accumulator 1.
T "Tag_Pointer"	// Transfer contents of accumulator 1 to the operand "Tag_Pointer".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting status bits (Page 2455)

TAR1 <D>: Transfer AR1 to double word

Description

Use the "Transfer AR1 to double word" instruction to transfer the contents of address register 1 (AR1) to a double word. The double word must hereby be addressed in one of the following memory areas:

- Bit memory (M)
- Temporary local data (L)
- Data blocks (DB, DI)

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Syntax

Use the following syntax for the "Transfer AR1 to double word" instruction:

```
TAR1 <D>
```

Parameters

The following table shows the parameters of the "Transfer AR1 to double word" instruction:

Parameters	Declaration	Data type	Memory area	Description
<D>	Output	DWORD	D, M, L	Operand to which the contents of accumulator 1 are transferred.

Example

The following example shows how the instruction works:

STL	Explanation
TAR1 %DBD20	// Transfer contents of address register 1 to data double word DBD20.
TAR1 %DID30	// Transfer contents of address register 1 to instance double word DID30.
TAR1 %LD18	// Transfer contents of address register 1 to local data double word LD18.
TAR1 %MD24	// Transfer contents of address register 1 to memory double word MD24.

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting status bits (Page 2455)

TAR1 AR2: Transfer AR1 to AR2

Description

Use the "Transfer AR1 to AR2" instruction to copy the contents of address register 1 (AR1) to address register 2 (AR2).

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Syntax

Use the following syntax for the "Transfer AR1 to AR2" instruction:

```
TAR1 AR2
```

Example

The following example shows how the instruction works:

STL	Explanation
LAR1 P#10.4	// Load contents of the area-internal pointer P#10.4 into address register 1.
TAR1 AR2	// Transfer contents of address register 1 to address register 2.
A I [AR2,P#3.2]	// Check whether signal state of bit I13.6 is "1" and AND with current RLO.
= "Tag1"	// Set operand "Tag_1" to "1"

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting status bits (Page 2455)

TAR2: Transfer AR2 to accumulator 1

Description

Use the "Transfer AR2 to accumulator 1" instruction to transfer the contents of address register 2 (AR2) to accumulator 1. The contents of accumulator 1 are shifted to accumulator 2.

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

Syntax

Use the following syntax for the "Transfer AR2 to accumulator 1" instruction:

```
TAR2
```

Example

The following example shows how the instruction works:

STL	Explanation
LAR2 P#20.0	// Load address register 2 with contents of the area internal pointer P#20.0.
TAR2	// Transfer contents of address register 2 (P#20.0) to accumulator 1.
T "Tag_Pointer"	// Transfer contents of accumulator 1 to the operand "Tag_Pointer".

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting status bits (Page 2455)

TAR2 <D>: Transfer AR2 to double word

Description

Use the "Transfer AR2 to double word" instruction to transfer the contents of address register 2 (AR2) to a double word. The double word must hereby be addressed in one of the following memory areas:

- Bit memory (M)
- Temporary local data (L)
- Data blocks (DB, DI)

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Syntax

Use the following syntax for the "Transfer AR2 to double word" instruction:

```
TAR2 <D>
```

Parameters

The following table shows the parameters of the "Transfer AR2 to double word" instruction:

Parameters	Declaration	Data type	Memory area	Description
<D>	Output	DWORD	D, M, L	Operand to which the contents of accumulator 2 are transferred.

Example

The following example shows how the instruction works:

```
STL                          Explanation
TAR2 %DBD20                  // Transfer contents of address register 2 to data
                             double word DBD20.
TAR2 %DID30                  // Transfer contents of address register 2 to instance
                             double word DID30.
TAR2 %LD18                   // Transfer contents of address register 2 to local
                             data double word LD18.
TAR2 %MD24                   // Transfer contents of address register 2 to memory
                             double word MD24.
```

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting status bits (Page 2455)

Conversion operations

BTI: Convert BCD to integer (16 bit)

Description

Use the "Convert BCD to integer (16 bit)" instruction to convert the value in the right word of accumulator 1 to a 16-bit integer. The instruction interprets the word that is to be converted as a three-digit binary coded decimal (BCD).

The bits 1 to 11 of accumulator 1 indicate the value that will be converted. Values in the range from "-999" to "+999" are permitted.

The sign of the result value is read from bit 15 of accumulator 1. If the signal state of the bit is "0", the sign is positive. If the signal state of the bit is "1," the sign is negative. Bits 12 through 14 are not used in the conversion.

If the BCD code contains a pseudo tetrad, the CPU reports a programming error and opens the organization block "OB121". If the organization block OB121 is not present, the CPU changes to stop mode.

The result of the conversion is saved in the right word of accumulator 1. The contents of the left word in accumulator 1 are not affected by the instruction.

Syntax

Use the following syntax for the "Convert BCD to integer (16 bit)" instruction:

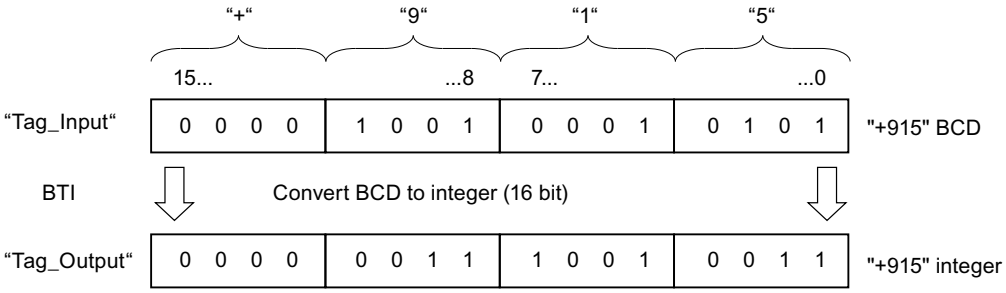
```
BTI
```

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Input"`	`// Load BCD in accumulator 1.`
`BTI`	`// Convert BCD to 16-bit integer.`
`T "Tag_Output"`	`// Transfer the result to operand "Tag_Output".`

The following figure shows how the instruction works using specific operand values:



See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting status bits (Page 2455)

ITB: Convert integer (16 bit) to BCD

Description

Use the "Convert integer (16 bit) to BCD" instruction to convert the value in the right word of accumulator 1 to a three-digit binary-coded decimal (BCD). The instruction interprets the value that is to be converted as a 16-bit integer.

The bits 0 to 11 of accumulator 1 indicate the value that will be converted. Values in the range from "-999" to "+999" are permitted. If the value to be converted is outside this range, the bits OV and OS are set to signal state "1". In this case the conversion is not executed.

The sign of the result value is read from bits 12 through 15 of accumulator 1. If the signal state of the bits is "0", the sign is positive. If the signal state of all four bits is "1", the sign is negative.

The result of the conversion is saved in the right word of accumulator 1. The contents of the left word in accumulator 1 are not affected by the instruction.

Syntax

Use the following syntax for the "Convert integer (16 bit) to BCD" instruction:

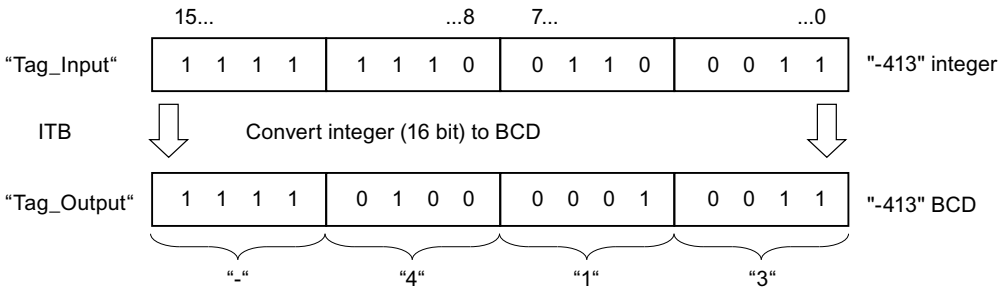
```
ITB
```

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Input"`	`// Load integer (16 bit) in accumulator 1.`
`ITB`	`// Convert integer to BCD.`
`T "Tag_Output"`	`// Transfer the result to operand "Tag_Output".`

The following figure shows how the instruction works using specific operand values:



See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bits in instructions with integers (Page 2458)

Basic information on the status word (Page 2453)

BTD: Convert BCD to integer (32 bit)

Description

Use the "Convert BCD to integer (32 bit)" to convert the value in accumulator 1 to a 32-bit integer. The instruction interprets the value that is to be converted as a seven-digit binary coded binary coded decimal (BCD).

The bits 0 to 27 of accumulator 1 indicate the value that will be converted. Values in the range from "-9999999" through "+9999999" are permitted.

The sign of the result value is read from bit 31 of accumulator 1. If the signal state of the bit is "0", the sign is positive. If the signal state of the bit is "1," the sign is negative. Bits 28 through 30 are not used in the conversion.

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

If the BCD code contains a pseudo tetrad, the CPU reports a programming error and opens the organization block "OB121". If the organization block OB121 is not present, the CPU changes to stop mode.

Syntax

Use the following syntax for the "Convert BCD to integer (32 bit)" instruction:

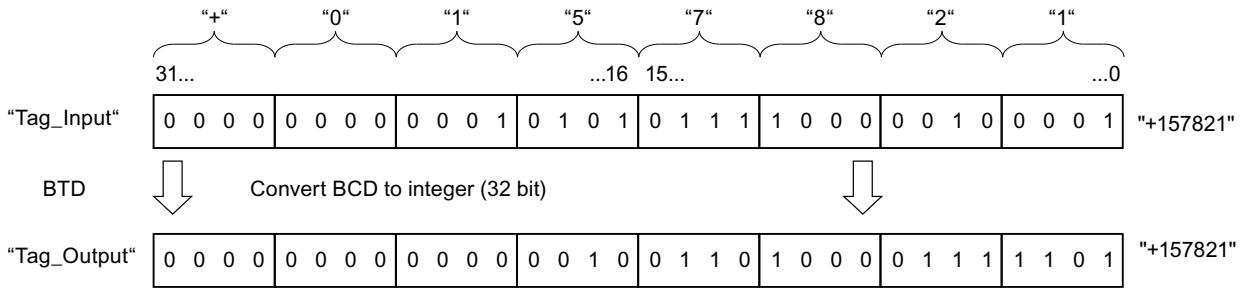
```
BTD
```

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Input"`	`// Load BCD in accumulator 1.`
`BTD`	`// BCD is converted to 32-bit integer.`
`T "Tag_Output"`	`// Transfer the result to operand "Tag_Output".`

The following figure shows how the instruction works using specific operand values:



See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting status bits (Page 2455)

ITD: Convert integer (16 bit) to integer (32 bit)

Description

Use the "Convert integer (16 bit) to integer (32 bit)" instruction to convert the value in the right word of accumulator 1 to a 32-bit integer. The instruction interprets the value that is to be converted as a 16-bit integer.

Bits 0 through 15 are not used in the conversion. The bits 16 through 31 of the result value are filled with the signal state of the bit 15.

The result value is saved in accumulator 1. The content of accumulator 2 remains unchanged.

Syntax

Use the following syntax for the "Convert integer (16 bit) to integer (32 bit)" instruction:

```
ITD
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Input"	// Load integer (16 bit) in accumulator 1.
ITD	// Convert integer (16 bit) to integer (32 bit).
T "Tag_Output"	// Transfer the result to operand "Tag_Output".

The following table shows the contents of accumulator 1 before and after processing of the instruction:

Status	Accumulator 1							
	31 . . .	. . . .	. . . .	. . . 16	15 . . .	. . . .	. . . .	. . . 0
Before processing:	XXXX	XXXX	XXXX	XXXX	1111	1111	1111	0110
After processing	1111	1111	1111	1111	1111	1111	1111	0110
	X: The signal state of the bits is irrelevant for the conversion							

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting status bits (Page 2455)

DTB: Convert integer (32 bit) to BCD

Description

Use the "Convert integer (32 bit) to BCD" instruction to convert the value in accumulator 1 to a seven-digit binary coded decimal (BCD). The instruction interprets the value that is to be converted as a 32-bit integer.

The bits 0 to 27 of accumulator 1 indicate the value that will be converted. Values in the range from "-9999999" through "+9999999" are permitted. If the value to be converted is outside this range, the bits OV and OS are set to signal state "1". In this case the conversion is not executed.

The sign of the result value is read from bits 28 through 31 of accumulator 1. If the signal state of the bits is "0", the sign is positive. If the signal state of all four bits is "1", the sign is negative.

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

Syntax

Use the following syntax for the "Convert integer (32 bit) to BCD" instruction:

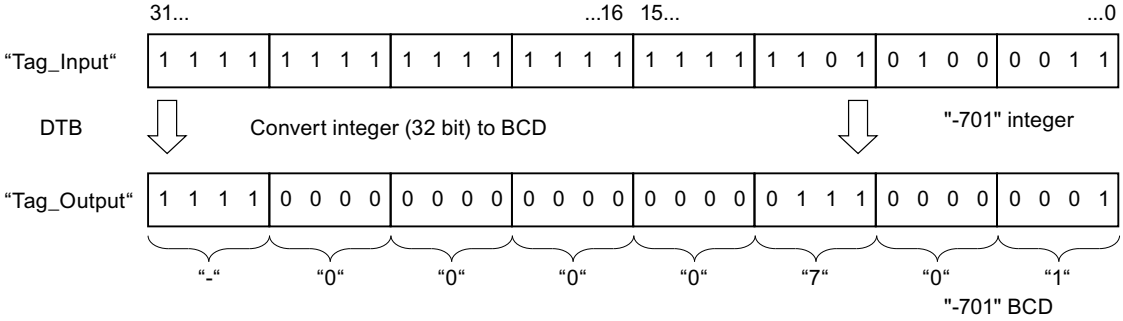
```
DTB
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Input"	// Load integer (32 bit) to accumulator 1
DTB	// Convert integer (32 bit) to BCD
T "Tag_Output"	// Transfer the result to operand "Tag_Output".

The following figure shows how the instruction works using specific operand values:



See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bits in instructions with integers (Page 2458)

Basic information on the status word (Page 2453)

DTR: Convert integer (32 bit) to floating-point number

Description

Use the "Convert integer (32 bit) to floating-point number" instruction to convert the value in accumulator 1 to a floating-point number. The instruction interprets the value that is to be converted as a 32-bit integer.

Because a 32-bit integer has a higher accuracy than a floating-point number, the result of the conversion is rounded off to the next integer.

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

Syntax

Use the following syntax for the "Convert integer (32 bit) to floating-point number" instruction:

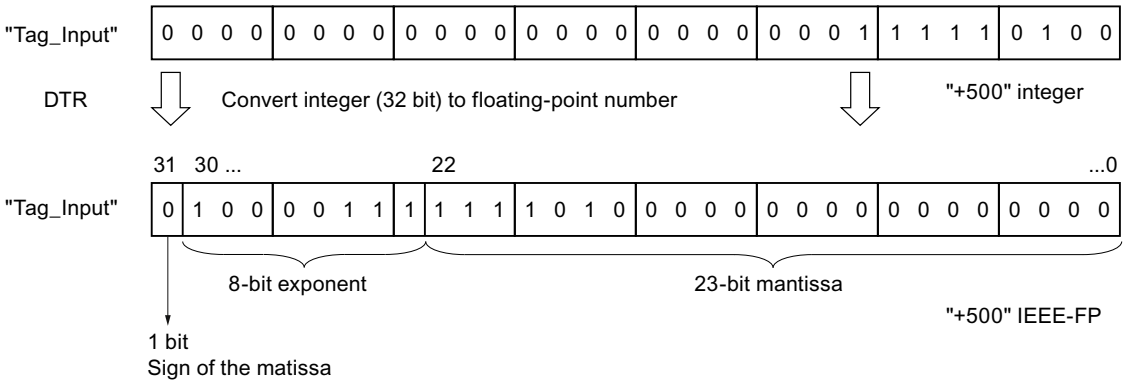
```
DTR
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Input"	// Load integer (32 bit) to accumulator 1
DTR	// Convert integer (32 bit) to floating-point number.
T "Tag_Output"	// Transfer the result to operand "Tag_Output".

The following figure shows how the instruction works using specific operand values:



See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting status bits (Page 2455)

INVI: Create ones complement integer (16 bit)

Description

Use the "Create ones complement integer (16 bit)" instruction to negate the value in the right word of accumulator 1 bit for bit.

During processing, the instruction inverts the signal state of each separate bit in the right word of accumulator 1. The ones are replaced with zeros and the zeros with ones.

The result is saved in the right word of accumulator 1. The contents of the left word in accumulator 1 are not affected by the instruction.

Syntax

Use the following syntax for the "Create ones complement integer (16 bit)" instruction:

```
INVI
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Input"	// Load value in accumulator 1
INVI	// Create ones complement
T "Tag_Output"	// Transfer the result to operand "Tag_Output".

The following table shows how the instruction works using specific operand values:

Operand	Value			
	15 . . .	. . . .	. . . .	. . . 0
Tag_Input	0110	0011	1010	1110
Tag_Output	1001	1100	0101	0001

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting status bits (Page 2455)

INVD: Create ones complement double integer (32 bit)

Description

Use the "Create ones complement double integer (32 bit)" to negate the value in accumulator 1 bit for bit.

During processing, the instruction inverts the signal state of each separate bit in accumulator 1. The ones are replaced with zeros and the zeros with ones.

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

Syntax

Use the following syntax for the "Create ones complement double integer (32 bit)" instruction:

```
INVD
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Input"	// Load value in accumulator 1
INVD	// Create ones complement
T "Tag_Output"	// Transfer the result to operand "Tag_Output".

The following table shows how the instruction works using specific operand values:

Operand	Value							
	31 . . .	. . . .	. . . .	. . . 16	15 . . .	. . . .	. . . .	. . . 0
Tag_Input	0110	1111	1000	1100	0110	0011	1010	1110
Tag_Output	1001	0000	0111	0011	1001	1100	0101	0001

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting status bits (Page 2455)

NEGI: Negate integer (16 bit)

Description

Use the "Negate integer (16 bit)" instruction to change the sign of the value in the right word of accumulator 1 by forming the twos complement. The instruction interprets the value of accumulator 1 as a 16-bit integer.

The processing of the instruction is equal to a multiplication with "-1". The result of the instruction is saved in the right word of accumulator 1. The contents of the left word in accumulator 1 are not affected by the instruction.

The "Negate integer (16 bit)" instruction affects the status bits CC 0, CC 1, OV and OS. The following table shows how the instruction influences the status bit depending on the result:

Result:	CC 1	CC 0	OV	OS
+1 to +32 767	1	0	0	-
0	0	0	0	-
-1 to -32 767	0	1	0	-
(-) 32 768	0	1	1	1

Syntax

The following syntax is used for the "Negate integer (16 bit)" instruction:

```
NEGI
```

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Input"                // Load value in accumulator 1
NEGI                         // Negate integer (16 bit)
T "Tag_Output"               // Transfer the result to operand "Tag_Output".
```

The following table shows how the instruction works using specific operand values:

Operand	Value			
	15 . . .	. . . .	. . . .	. . . 0
Tag_Input	0101	1101	0011	1000
Tag_Output	1010	0010	1100	1000

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

NEGD: Negate integer (32 bit)

Description

Use the "Negate integer (32 bit)" instruction to change the sign of the value in accumulator 1 by forming the twos complement. The instruction interprets the value of accumulator 1 as a 32-bit integer.

The processing of the instruction is equal to a multiplication with "-1". The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

The "Negate integer (32 bit)" instruction affects the status bits CC 0, CC 1, OV and OS. The following table shows how the instruction influences the status bit depending on the result:

Result	CC 1	CC 0	OV	OS
+1 to +2 147 483 647	1	0	0	-
0	0	0	0	-
-1 to -2 147 483 647	0	1	0	-
(-) 2 147 483 648	0	1	1	1

Syntax

The following syntax is used for the "Negate integer (32 bit)" instruction:

```
NEGD
```

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Input"                // Load value in accumulator 1
NEGD                         // Negate integer (32 bit).
T "Tag_Output"               // Transfer the result to operand "Tag_Output".
```

The following table shows how the instruction works using specific operand values:

Status	Value							
	31 . . .	. . . .	. . . .	. . . 16	15 . . .	. . . .	. . . .	. . . 0
Tag_Input	0101	1111	0110	0100	0101	1101	0011	1000
Tag_Output	1010	0000	1001	1011	1010	0010	1100	1000

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting the status bits in instructions with integers (Page 2458)

NEGR: Negate floating-point number

Description

Use the "Negate floating-point number" instruction to invert the signal state of bit 31 in accumulator 1. The instruction interprets the value of accumulator 1 as a floating-point number. The sign of the mantissa is in bit 31 of the floating-point number.

The processing of the instruction is equal to a multiplication with "-1". After the processing of the instruction, the sign of the mantissa is inverted.

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

The instruction does not affect any status bits.

Syntax

Use the following syntax for the "Negate floating-point number" instruction:

```
NEGR
```

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Input"`	`// Load floating-point number into accumulator 1`
`NEGR`	`// Negate floating-point number.`
`T "Tag_Output"`	`// Transfer the result to operand "Tag_Output".`

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Input	1.5E+02
Tag_Output	-1.5E+02

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

CAW: Switch bytes in the right word of accumulator 1

Description

Use the "Switch bytes in the right word of accumulator 1" instruction to switch the sequence of the two right bytes in the right word of accumulator 1.

The following table shows the contents of accumulator 1 before and after processing of the instruction:

Status	Bytes in accumulator 1			
Before processing:	Value A	Value B	Value C	Value D
After processing	Value A	Value B	Value D	Value C

The result of the instruction is saved in the right word of accumulator 1. The bytes in the left word of accumulator 1 are not influenced by the instruction and remain unchanged.

Syntax

Use the following syntax for the "Switch bytes in the right word of accumulator 1" instruction:

```
CAW
```

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Input"                // Load value in accumulator.
CAW                          // Switch bytes in the right word of accumulator 1.
T "Tag_Output"               // Transfer the result to operand "Tag_Output".
```

The following table shows how the instruction works using specific operand values:

Operand	Value			
Tag_Input	0000	0000	1111	1111
Tag_Output	1111	1111	0000	0000

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting status bits (Page 2455)

CAD: Switch all bytes in accumulator 1

Description

Use the "Switch all bytes in accumulator 1" instruction to switch the sequence of the bytes in accumulator 1.

The following table shows the contents of accumulator 1 before and after processing of the instruction:

Status	Bytes in accumulator 1			
Before processing:	Value A	Value B	Value C	Value D
After processing	Value D	Value C	Value B	Value A

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

Syntax

Use the following syntax for the "Switch all bytes in accumulator 1" instruction:

```
CAD
```

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Input"                // Load value of the operand in accumulator 1.
CAD                          // Switch bytes of accumulator 1.
T "Tag_Output"               // Transfer the result to operand "Tag_Output".
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Input	1111	0000	0000	1111	0000	0000	1111	1111
Tag_Output	1111	1111	0000	0000	0000	1111	1111	0000

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting status bits (Page 2455)

RND: Round numerical value

Description

Use the "Round numerical value" instruction to convert the value in accumulator 1 to a 32-bit integer. The instruction interprets the value that is to be converted as a floating-point number and round this to the next integer.

If the floating-point number is exactly between an even and odd number, the even number is selected as result.

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

The conversion is not executed and the status bits OV and OS are set, if one of the following conditions applies:

- The value in accumulator 1 is not a valid floating-point number.
- The result is outside the range permitted for an integer of the data type DINT.

Syntax

Use the following syntax for the "Round numerical value" instruction:

```
RND
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Input"	// Load floating-point number into accumulator 1
RND	// Round floating-point number to an integer.
T "Tag_Output"	// Transfer the result to operand "Tag_Output".

The following table shows how the instruction works using specific operand values:

Operand	Values	
Tag_Input	101.5	-101.5
Tag_Output	102	-102

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bits in instructions with floating-point numbers (Page 2459)

Basic information on the status word (Page 2453)

TRUNC: Truncate numerical value

Description

Use the "Truncate numerical value" instruction to convert the value in accumulator 1 to a 32-bit integer. The instruction interprets the value to be converted as a floating-point number and returns its whole number as result.

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

The conversion is not executed and the status bits OV and OS are set, if one of the following conditions applies:

- The value in accumulator 1 is not a valid floating-point number.
- The result is outside the range permitted for an integer of the data type DINT.

Syntax

Use the following syntax for the "Truncate numerical value" instruction:

```
TRUNC
```

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Input"`	`// Load floating-point number into accumulator 1`
`TRUNC`	`// Convert floating-point number to an integer of data type DINT.`
`T "Tag_Output"`	`// Transfer the result to operand "Tag_Output".`

The following table shows how the instruction works using specific operand values:

Operand	Values	
Tag_Input	101.5	-101.5
Tag_Output	101	-101

See also

Overview of the valid data types (Page 2236)

Setting the status bits in instructions with floating-point numbers (Page 2459)

Basic information on the status word (Page 2453)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

RND+: Generate next higher integer from floating-point number

Description

Use the "Generate next higher integer from floating-point number" instruction to convert the value in accumulator 1 to a 32-bit integer. The instruction interprets the value that is to be converted as a floating-point number and rounds this number to the next higher integer. The result of the instruction is a number of the data type DINT that is greater than or equal to the floating-point number to be converted.

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

The conversion is not executed and the status bits OV and OS are set, if one of the following conditions applies:

- The value in accumulator 1 is not a valid floating-point number.
- The result is outside the range permitted for an integer of the data type DINT.

Syntax

Use the following syntax for the "Generate next higher integer from floating-point number" instruction:

```
RND+
```

Example

The following example shows how the instruction works:

STL	Explanation
`L "Tag_Input"`	`// Load floating-point number into accumulator 1`
`RND+`	`// Round floating-point number to next higher integer.`
`T "Tag_Output"`	`// Transfer the result to operand "Tag_Output".`

The following table shows how the instruction works using specific operand values:

Operand	Values	
Tag_Input	100.5	-100.5
Tag_Output	101	-100

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bits in instructions with floating-point numbers (Page 2459)

Basic information on the status word (Page 2453)

RND-: Generate next lower integer from floating-point number

Description

Use the "Generate next lower integer from floating-point number" instruction to convert the value in accumulator 1 to a 32-bit integer. The instruction interprets the value that is to be converted as a floating-point number and rounds this number to the next lower integer. The result of the instruction is a number of the data type DINT that is less than or equal to the floating-point number to be converted.

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

The conversion is not executed and the status bits OV and OS are set, if one of the following conditions applies:

- The value in accumulator 1 is not a valid floating-point number.
- The result is outside the range permitted for an integer of the data type DINT.

Syntax

Use the following syntax for the "Generate next lower integer from floating-point number" instruction:

```
RND-
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Input"	// Load floating-point number into accumulator 1
RND-	// Round floating-point number to next lower integer.
T "Tag_Output"	// Transfer the result to operand "Tag_Output".

The following table shows how the instruction works using specific operand values:

Operand	Values	
Tag_Input	100.5	-100.5
Tag_Output	100	-101

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bits in instructions with floating-point numbers (Page 2459)

Basic information on the status word (Page 2453)

Program control operations

Jumps

Jump label

Description

Use a jump label to specify the place in the program at which the program execution should resume after a jump. The name of the jump label can consist of up to 128 letters, numbers or underscores.

The jump label and the instruction in which the jump label is specified must be located in the same block. The name of a jump label can only be assigned once in a block. Each jump label can jump to several locations.

Both forward and backward jumps are possible. The maximum jump distance is -32 768 or +32 767 words of program code. The actual maximum number of instructions you can jump over depends on the mix of the statements used in your program (one, two, or three-word instructions).

If you use the MCR functions, the jump label must be located in the same MCR range as the jump instruction.

Syntax

Use the following syntax for the jump label:

<jump label>:

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Input_1"	// Load first comparison value.
L "Tag_Input_2"	// Load second comparison value.
>I	// Check whether the value of the operand "Tag_Input_1" is greater than the value of the operand "Tag_Input_2"
JC MyLABEL	// With a scan result "1" jump to jump label "MyLABEL" and continue with program processing
	// If the result of the query is "0", execute the next instruction.
L "Tag_Input_3"	// Load content of the operand "Tag_Input_3" in accumulator 1.
T "Tag_Output"	// Load contents of accumulator 1 in operand "Tag_Output".
MyLABEL: A "Tag_Input_4"	// If there was a jump, continue program execution at this point.
	// Query whether operand "Tag_Input_4" is "1"

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting status bits (Page 2455)

Overview of the valid data types (Page 2236)

JU: Unconditional jump

Description

Use the "Unconditional jump" instruction to interrupt the linear program execution and continue at the point that is marked by the specified jump label. The instruction is always executed independently of conditions.

The "Unconditional jump" instruction does not affect the status bits.

Syntax

Use the following syntax for the "Unconditional jump" instruction:

```
JU <jump label>
```

Parameters

The following table shows the parameters of the "Unconditional jump" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Input_1"              // Load first comparison value.
L "Tag_Input_2"              // Load second comparison value.
>I                           // Check whether the value of the operand
                             "Tag_Input_1" is greater than the value of the operand
                             "Tag_Input_2"
JC MyLABEL_1                 // With a scan result "1" jump to jump label
                             "MyLABEL_1" and continue with program processing
                             // If the result of the query is "0", continue linear
                             program execution.
L "Tag_Input_3"              // Load content of the operand "Tag_Input_3" in
                             accumulator 1.
T "Tag_Output_1"             // Load contents of accumulator 1 in operand
                             "Tag_Output_1".
JU MyLABEL_2                 // Jump to jump label "MyLABEL_2" and resume execution
                             of the program there.
MyLABEL_1: L "Tag_Input_4"   // Jump label
                             // Load content of the operand "Tag_Input_4" in
                             accumulator 1.
T "Tag_Output_2"             // Load contents of accumulator 1 in operand
                             "Tag_Output_2".
MyLABEL_2: A "Tag_Input_5"   // Jump label "MyLABEL_2"
                             // Query whether operand "Tag_Input_5" is "1"
```

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Example of a step sequence (Page 3142)

Setting status bits in program control operation instructions (Page 2460)

Querying status bits with jump instructions (Page 2464)

Basic information on the status word (Page 2453)

JC: Jump if RLO = 1

Description

Use the "Jump if RLO = 1" instruction to interrupt linear program execution depending on the result of logic operation and resume it at the point marked by the specified jump label.

The jump to the specified jump label is only executed when the current RLO is "1". If the current RLO before the instruction is "0", the jump is not executed and the program is resumed with the next instruction.

The "Jump if RLO = 1" instruction sets the RLO and the status bit STA to "1" and the status bits OR and /FC to "0" both when the condition is met and when it is not met.

Syntax

Use the following syntax for the "Jump if RLO = 1" instruction:

```
JCB <jump label>
```

Parameters

The following table shows the parameters of the "Jump if RLO = 1" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Input_1"	// Load first comparison value.
L "Tag_Input_2"	// Load second comparison value.
>I	// Check whether the value of the operand "Tag_Input_1" is greater than the value of the operand "Tag_Input_2"
JC MyLABEL	// If RLO = "1", jump to jump label "MyLABEL" and continue program execution there.
	// If the RLO = "0", execute the next instruction.
L "Tag_Input_3"	// Load content of the operand "Tag_Input_3" in accumulator 1.
T "Tag_Output_1"	// Load contents of accumulator 1 in operand "Tag_Output_1".
MyLABEL: L "Tag_Input_4"	// If there was a jump, continue program execution at this point.
	// Load content of the operand "Tag_Input_4" in accumulator 1.

STL	Explanation
`T "Tag_Output_2"`	`// Transfer contents of accumulator 1 to the operand "Tag_Output_2".`

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Example of a step sequence (Page 3142)

Setting status bits in program control operation instructions (Page 2460)

Querying status bits with jump instructions (Page 2464)

Basic information on the status word (Page 2453)

JCN: Jump if RLO = 0

Description

Use the "Jump if RLO = 0" instruction to interrupt the linear program execution depending on the result of logic operation (RLO) and resume it at the point marked by the specified jump label.

The jump to the specified jump label is only executed when the current RLO is "0". If the current RLO before the instruction is "1", the jump is not executed and the program is resumed with the next instruction.

The "Jump if RLO = 0" instruction sets the RLO and the status bit STA to "1" and the status bits OR and /FC to "0" both when the condition is met and when it is not met.

Syntax

Use the following syntax for the "Jump if RLO = 0" instruction:

```
JCN <jump label>
```

Parameters

The following table shows the parameters of the "Jump if RLO = 0" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input_1"	// Query whether the value of operand "Tag_Input_1" is "1" and AND.
A "Tag_Input_2"	// Query whether the value of operand "Tag_Input_2" is "1" and AND.
JCN MyLABEL	// If RLO = "0", jump to jump label "MyLABEL" and continue program execution there.
	// If the RLO = "1", execute the next instruction.
L "Tag_Input_3"	// Load content of the operand "Tag_Input_3" in accumulator 1.
T "Tag_Output"	// Load contents of accumulator 1 in operand "Tag_Output".
MyLABEL: A "Tag_Input_4"	// If there was a jump, continue program execution at this point.
	// Query whether the value of operand "Tag_Input_4" is "1" and AND with the RLO

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Example of a step sequence (Page 3142)

Setting status bits in program control operation instructions (Page 2460)

Querying status bits with jump instructions (Page 2464)

JCB: Jump if RLO = 1 and save RLO

Description

Use the "Jump if RLO = 1 and save RLO" instruction to interrupt the linear program execution depending on the result of logic operation (RLO) and resume it at the point marked by the specified jump label. At the same time, the signal state of the current RLO is copied to the binary result (BR).

The jump to the specified jump label is only executed when the current RLO is "1". In this case the instruction set the binary result to "1".

If the current RLO before the instruction is "0", the jump is not executed and the program is resumed with the next instruction. In this case the instruction assigns the signal state "0" to the binary result.

The "Jump if RLO = 1 and save RLO" instruction sets the RLO and the status bit STA to "1" and the status bits OR and /FC to "0" both when the condition is met and when it is not met.

Syntax

Use the following syntax for the "Jump if RLO = 1 and save RLO" instruction:

```
JCB <jump label>
```

Parameters

The following table shows the parameters of the "Jump if RLO = 1 and save RLO" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Input_1"	// Load first comparison value.
L "Tag_Input_2"	// Load second comparison value.
>I	// Check whether the value of the operand "Tag_Input_1" is greater than the value of the operand "Tag_Input_2"
JCB MyLABEL	// Copy current RLO to BR.
	// If RLO = "1", jump to jump label "MyLABEL" and continue program execution there.
	// If the RLO = "0", execute the next instruction.
L "Tag_Input_3"	// Load content of the operand "Tag_Input_3" in accumulator 1.
T "Tag_Output_1"	// Load contents of accumulator 1 in operand "Tag_Output_1".
MyLABEL: L "Tag_Input_4"	// If there was a jump, continue program execution at this point.
	// Load content of the operand "Tag_Input_4" in accumulator 1.
T "Tag_Output_2"	// Load contents of accumulator 1 in operand "Tag_Output_2".

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits in program control operation instructions (Page 2460)

Querying status bits with jump instructions (Page 2464)

Overview of the valid data types (Page 2236)

JNB: Jump if RLO = 0 and save RLO

Description

Use the "Jump if RLO = 0 and save RLO" instruction to interrupt the linear program execution depending on the result of logic operation (RLO) and resume it at the point marked by the specified jump label. At the same time, the signal state of the current result of logic operation is copied to the binary result (BR).

The jump to the specified jump label is only executed when the current RLO is "0". In this case the instruction sets the binary result to "0".

If the current RLO before the instruction is "1", the jump is not executed and the program is resumed with the next instruction. In this case the instruction assigns the signal state "1" to the binary result.

The "Jump if RLO = 0 and save RLO" instruction sets the RLO and the status bit STA to "1" and the status bits OR and /FC to "0" both when the condition is met and when it is not met.

Syntax

Use the following syntax for the "Jump if RLO = 0 and save RLO" instruction:

```
JNB <jump label>
```

Parameters

The following table shows the parameters of the "Jump if RLO = 0 and save RLO" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input_1"	// Query whether the value of operand "Tag_Input_1" is "1" and AND.
A "Tag_Input_2"	// Query whether the value of operand "Tag_Input_2" is "1" and AND.
JNB MyLABEL	// Copy current RLO to BR.
	// If RLO = "0", jump to jump label "MyLABEL" and continue program execution there.
	// If the RLO = "1", execute the next instruction.
L "Tag_Input_3"	// Load content of the operand "Tag_Input_3" in accumulator 1.
T "Tag_Output"	// Transfer contents of accumulator 1 to the operand "Tag_Output".

STL	Explanation
MyLABEL: A "Tag_Input_4"	// If there was a jump, continue program execution at this point. // Query whether the value of operand "Tag_Input_4" is "1" and AND with the RLO

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits in program control operation instructions (Page 2460)

Querying status bits with jump instructions (Page 2464)

Overview of the valid data types (Page 2236)

JBI: Jump if BR = 1

Description

Use the "Jump if BR = 1" instruction to interrupt linear program execution depending on the binary result and resume it at the point marked by the specified jump label.

The jump to the specified jump label is only executed if the status bit BR is "1". If the status bit BR is "0", the jump is not executed and the program resumes with the next instruction.

The "Jump if BR = 1" instruction sets the status bit STA to "1" and the status bits OR and /FC to "0" both when the condition is met and when it is not met.

Syntax

Use the following syntax for the "Jump if BR = 1" instruction:

```
JBI <jump label>
```

Parameters

The following table shows the parameters of the "Jump if BR = 1" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

STL	Explanation
SET	// Set RLO to "1" at the start of the block.
SAVE	// Transfer signal state of the RLO to the BR bit.
....	// Any program
AN OV	// Query OV bit at end of block.
SAVE	// If there is an overflow, set BR bit to "0".
	// Set BR to "1" if there is no overflow.
JBI END	// If BR = "1", jump to jump label END.
	// If BR = "0", execute the next instruction.
R "Tag_Output_1"	// Reset operand "Tag_Output_1" to "0".
END: S "Tag_Output_2"	// Set operand "Tag_Output_2" to "1"

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting status bits (Page 2455)

Setting status bits in program control operation instructions (Page 2460)

Querying status bits with jump instructions (Page 2464)

Overview of the valid data types (Page 2236)

JNBI: Jump if BR = 0

Description

Use the "Jump if BR = 0" instruction to interrupt the linear program execution depending on the binary result (BR) and resume it at the point marked by the specified jump label.

The jump to the specified jump label is only executed if the status bit BR is "0". If the status bit BR before the instruction is "1", the jump is not executed and the program resumes with the next instruction.

The "Jump if BR = 0" instruction sets the status bit STA to "1" and the status bits OR and /FC to "0" both when the condition is met and when it is not met.

Syntax

Use the following syntax for the "Jump if BR = 0" instruction:

```
JNBI <jump label>
```

Parameters

The following table shows the parameters of the "Jump if BR = 0" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

```
STL                          Explanation
SET                          // Set RLO to "1" at the start of the block.
SAVE                         // Transfer signal state of the RLO to the BR bit.
....                         // Any program
AN OV                        // Query OV bit at end of block.
SAVE                         // If there is an overflow, set BR bit to "0".
                             // Set BR to "1" if there is no overflow.
JNBI END                     // If BR = "0", jump to jump label END.
                             // If BR = "1", execute the next instruction.
S "Tag_Output"               // Set operand "Tag_Output" to "1"
END: BE                      // End block
```

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting status bits (Page 2455)

Setting status bits in program control operation instructions (Page 2460)

Querying status bits with jump instructions (Page 2464)

JO: Jump if OV = 1

Description

Use the "Jump if OV = 1" instruction to interrupt linear program execution depending on the status bit OV and resume it at the point marked by the specified jump label.

The jump to the specified jump label is only executed if the status bit OV is "1". The status bit OV is set if, for example, the result of a calculation is outside the permitted range or an invalid value is returned during the comparison of floating-point numbers.

If the status bit OV is "0", the jump is not executed and the program continues with the next instruction.

Syntax

Use the following syntax for the "Jump if OV = 1" instruction:

```
JO <jump label>
```

Parameters

The following table shows the parameters of the "Jump if OV = 1" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value
L "Tag_Value_2"	// Load value
*I	// Multiply values
JO OVER	// If the status bit OV has signal state "1", jump to jump label "OVER" and continue program execution there.
	// If the signal state of the status bit OV is "0", execute the next instruction.
T "Tag_Result"	// Transfer the result of the multiplication to the operand "Tag_Result".
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
OVER: SET	// Continue program execution at this point if the condition (OV = "1") is satisfied.
	// Set RLO to signal state "1".
R "Tag_Output"	// Reset operand "Tag_Output" to "0".
NEXT: A "MyTag_1"	// Jump label "NEXT"
	// Query whether the value of operand "MyTag_1" is "1" and AND.
A "MyTag_2"	// Query whether the value of operand "MyTag_2" is "1" and AND.
S "Tag_Output_2"	// If RLO = 1, set the "Tag_Output_2" operand to "1".

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting status bits (Page 2455)

Setting status bits in program control operation instructions (Page 2460)

Querying status bits with jump instructions (Page 2464)

JOS: Jump if OS = 1

Description

Use the "Jump if OS = 1" instruction to interrupt linear program execution depending on the status bit OS and resume it at the point marked by the specified jump label.

The jump to the specified jump label is only executed if the status bit OS is "1". The status bit OS is always set when a number range overflow sets the status bit OV to "1". In contrast to status bit OV, the status bit OS remains set even if a result is subsequently in the permitted number range.

If the status bit OS is "0", the jump is not executed and the program continues with the next instruction.

Syntax

Use the following syntax for the "Jump if OS = 1" instruction:

```
JOS <jump label>
```

Parameters

The following table shows the parameters of the "Jump if OS = 1" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value
L "Tag_Value_2"              // Load value
*I                           // Multiply loaded values.
                             // Store product in accumulator 1.
L "Tag_Value_3"              // Load value
+I                           // Add loaded value to the product.
                             // Store product in accumulator 1.
L "Tag_Value_4"              // Load value
```

STL	Explanation
-I	// Subtract loaded value from the calculated sum.
	// Store product in accumulator 1.
JOS OVER	// If there is an overflow in one of the three preceding instructions, the status bit OS is set to "1"
	// If the signal state of the status bit OS is "1", the program resumes at the jump label "OVER".
	// If the signal state of the status bit OS is "0", the next instruction is executed
T "Tag_Result"	// Transfer the result of the entire calculation to the operand "Tag_Result"
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
OVER: SET	// Continue program execution at this point if the condition (OV = "1") is satisfied.
	// Set RLO to signal state "1".
R "Tag_Output"	// Reset operand "Tag_Output" to "0".
NEXT: A "MyTag_1"	// Jump label "NEXT"
	// Query whether the value of operand "MyTag_1" is "1" and AND.
A "MyTag_2"	// Query whether the value of operand "MyTag_2" is "1" and AND.
S "Tag_Output_2"	// If RLO = 1, set the "Tag_Output_2" operand to "1".

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting status bits (Page 2455)

Setting status bits in program control operation instructions (Page 2460)

Querying status bits with jump instructions (Page 2464)

JZ: Jump if the result is zero

Description

Use the "Jump if the result is zero" instruction to interrupt linear program execution depending on the status bits CC 0 and CC 1 and continue at the point marked by the specified jump label.

The jump to the specified jump label is only executed if the signal state of the status bits CC 0 and CC 1 is "0". The situation occurs when one of the following conditions applies:

- After processing an arithmetic instruction without overflow, accumulator 1 has the value "0".
- A negative overflow occurs when executing the "Add integers (16 bit)" (+I) or "Add double integers (32 bit)" (+D) instruction.

- The value gradually drops below the permitted range when processing an arithmetic instruction (floating-point arithmetic) with overflow.
- The content of accumulator 2 is equal to the content of accumulator 1 after the execution of a comparison instruction.
- The content of accumulator 1 is zero after the execution of a word logic operation.
- The value of the last shifted out bit is "0" after the execution of a shift instruction.

In all other cases the jump is not executed and the processing of the program is resumed with the next instruction.

Syntax

Use the following syntax for the "Jump if the result is zero" instruction:

```
JZ <jump label>
```

Parameters

The following table shows the parameters of the "Jump if the result is zero" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value"	// Load value of the operand "Tag_Value".
SRW 1	// Shift the contents of the right word in accumulator 1 one place to the right.
JZ ZERO	// If the signal state of the last bit shifted is "0", jump to jump label "OVER" and continue with program processing from there
	// If the signal state of the last bit shifted is "1", execute the next instruction.
L "MyTag_1"	// Load value of the operand "MyTag_1" in accumulator 1
INC 1	// Add right-hand byte of accumulator 1 with the value "1".
T "Tag_Result_1"	// Transfer the result to operand "Tag_Result_1".
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
ZERO: L "MyTag_2"	// Jump label "ZERO"
	// Load value of the operand "MyTag_2" in accumulator 1

STL	Explanation
INC 1	// Add right-hand byte of accumulator 1 with the value "1".
T "Tag_Result_2"	// Transfer the result to operand "Tag_Result_2".
NEXT: A "MyTag_3"	// Jump label "NEXT"
	// Query whether the value of operand "MyTag_3" is "1" and AND.
A "MyTag_4"	// Query whether the value of operand "MyTag_4" is "1" and AND.

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting status bits (Page 2455)

Setting status bits in program control operation instructions (Page 2460)

Querying status bits with jump instructions (Page 2464)

JN: Jump if the result is not zero

Description

Use the "Jump if the result is not zero" instruction to interrupt linear program execution depending on the status bits CC 0 and CC 1 and continue at the point marked by the specified jump label.

The jump to the specified jump label is only executed if the signal states of the status bits CC 0 and CC 1 are different. The situation occurs when one of the following conditions applies:

- After processing an arithmetic instruction without overflow, the value of accumulator 1 is not equal to zero.
- A negative overflow occurs when processing an arithmetic instruction with integers (+I, -I, *I, +D, -D, *D).
- A positive overflow occurs when executing a math instruction with integers (+I, -I, *I, /I, +D, -D, *D, /D, NEGI, NEGD).
- A positive or negative overflow occurs when processing an arithmetic instruction with floating-point numbers.
- The content of accumulator 2 is not equal to the content of accumulator 1 after the execution of a comparison instruction.
- The content of accumulator 1 is not zero after the execution of a word logic operation.
- The value of the last shifted out bit is "1" after the execution of a shift instruction.

In all other cases the jump is not executed and the processing of the program is resumed with the next instruction.

Syntax

Use the following syntax for the "Jump if the result is not zero" instruction:

```
JN <jump label>
```

Parameters

The following table shows the parameters of the "Jump if the result is not zero" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Valu_1".
L "Tag_Value_2"	// Load value of the operand "Tag_Value_2".
XOW	// Apply EXCLUSIVE OR logic operation
JN NOZERO	// If the value in accumulator 1 is not equal to zero, the jump to the "NOZERO" jump label is performed and program processing continues from there.
	// If the value in accumulator 1 is equal to zero, the next instruction will be processed.
AN "MyTag_1"	// Query whether the value of operand "MyTag_1" is "0" and AND.
S "Tag_Output_1"	// If RLO is "1", set the operand "Tag_Output_1" to "1".
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
NOZERO: AN "MyTag_2"	// Jump label "NOZERO"
	// Query whether the value of operand "MyTag_2" is "0" and AND.
S "Tag_Output_2"	// If RLO is "1", set the operand "Tag_Output_2" to "1"
NEXT: A "MyTag_3"	// Jump label "NEXT"
	// Query whether the value of operand "MyTag_3" is "1" and AND.
A "MyTag_4"	// Query whether the value of operand "MyTag_4" is "1" and AND.

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting status bits (Page 2455)

Setting status bits in program control operation instructions (Page 2460)

Querying status bits with jump instructions (Page 2464)

JP: Jump if the result is greater than zero

Description

Use the "Jump if the result is greater than zero" instruction to interrupt linear program execution depending on the status bits CC 0 and CC 1 and continue at the point marked by the specified jump label.

The jump to the specified jump label is executed only if the signal state of status bit CC 0 is "0" and CC 1 is "1". The situation occurs when one of the following conditions applies:

- After processing an arithmetic instruction without overflow, the value of accumulator 1 is greater than zero.
- A negative overflow occurs when processing an arithmetic instruction with integers (+I, -I, *I, +D, -D, *D).
- A positive overflow occurs when processing an arithmetic instruction with integers (*I, /I, *D, /D).
- A positive overflow occurs when processing an arithmetic instruction with floating-point numbers.
- After the execution of a comparison instruction, the content of accumulator 2 is greater than the content of accumulator 1.
- The contents of accumulator 1 are not zero after the execution of a word logic operation.
- The value of the last shifted out bit is "1" after the execution of a shift instruction.

In all other cases the jump is not executed and the processing of the program is resumed with the next instruction.

Syntax

Use the following syntax for the "Jump if the result is greater than zero" instruction:

```
JP <jump label>
```

Parameters

The following table shows the parameters of the "Jump if the result is greater than zero" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1".
L "Tag_Value_2"	// Load value of the operand "Tag_Value_2".
-I	// Subtract value of the operand "Tag_Value_2" from the value of the operand "Tag_Value_1".
JP POSITIVE	// At a positive result value in accumulator "1" jump to jump label "POSITIVE" and continue program processing from there
	// If the result in accumulator 1 is negative, execute the next instruction.
AN "MyTag_1"	// Query whether the value of operand "MyTag_1" is "0" and AND.
S "Tag_Output_1"	// If RLO is "1", set the operand "Tag_Output_1" to "1".
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
POSITIVE: AN "MyTag_2"	// Jump label "POSITIVE"
	// Query whether the value of operand "MyTag_2" is "0" and AND.
S "Tag_Output_2"	// If RLO is "1", set the operand "Tag_Output_2" to "1".
NEXT: A "MyTag_3"	// Jump label "NEXT"
	// Query whether the value of operand "MyTag_3" is "1" and AND.
A "MyTag_4"	// Query whether the value of operand "MyTag_4" is "1" and AND.

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting status bits (Page 2455)

Setting status bits in program control operation instructions (Page 2460)

Querying status bits with jump instructions (Page 2464)

JM: Jump if the result is less than zero

Description

Use the "Jump if the result is less than zero" instruction to interrupt linear program execution depending on the status bits CC 0 and CC 1 and continue at the point marked by the specified jump label.

The jump to the specified jump label is executed only if the signal state of status bit CC 0 is "1" and CC 1 is "0". The situation occurs when one of the following conditions applies:

- After processing an arithmetic instruction without overflow, the value of accumulator 1 is less than zero.
- A negative overflow occurs when processing an arithmetic instruction with integers (*I, *D).
- A positive overflow occurs when processing an arithmetic instruction with integers (+I, -I, +D, -D, NEGI, NEGD).
- A negative overflow occurs when processing an arithmetic instruction with floating-point numbers.
- After the execution of a comparison instruction, the content of accumulator 2 is less than the content of accumulator 1.

In all other cases the jump is not executed and the processing of the program is resumed with the next instruction.

Syntax

Use the following syntax for the "Jump if the result is less than zero" instruction:

```
JM <jump label>
```

Parameters

The following table shows the parameters of the "Jump if the result is less than zero" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand "Tag_Value_1".
L "Tag_Value_2"              // Load value of the operand "Tag_Value_2".
-I                           // Subtract value of the operand "Tag_Value_2" from
                             the value of the operand "Tag_Value_1".
```

STL	Explanation
JM NEGATIVE	// At a negative result value in accumulator "1" jump to jump label "NEGATIVE" and continue program processing from there
	// If the result in accumulator 1 is positive, execute the next instruction.
AN "MyTag_1"	// Query whether the value of operand "MyTag_1" is "0" and AND.
S "Tag_Output_1"	// If RLO is "1", set the operand "Tag_Output_1" to "1".
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
NEGATIVE: AN "MyTag_2"	// Jump label "NEGATIVE"
	// Query whether the value of operand "MyTag_2" is "0" and AND.
S "Tag_Output_2"	// If RLO is "1", set the operand "Tag_Output_2" to "1".
NEXT: A "MyTag_3"	// Jump label "NEXT"
	// Query whether the value of operand "MyTag_3" is "1" and AND.
A "MyTag_4"	// Query whether the value of operand "MyTag_4" is "1" and AND.

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting status bits (Page 2455)

Setting status bits in program control operation instructions (Page 2460)

Querying status bits with jump instructions (Page 2464)

JPZ: Jump if the result is greater than or equal to zero

Description

Use the "Jump if the result is greater than or equal to zero" instruction to interrupt linear program execution depending on the status bit CC 0 and continue it at the point marked by the specified jump label.

The jump to the specified jump label is only executed if the signal state of the status bit CC 0 is "0". The situation occurs when one of the following conditions applies:

- After processing an arithmetic instruction without overflow, the value of accumulator 1 is greater than or equal to zero.
- A negative overflow occurs when processing an arithmetic instruction with integers (+I, -I, *I, +D, -D, *D).

- A positive overflow occurs when processing an arithmetic instruction with integers (*I, /I, *D, /D).
- A positive overflow occurs when processing an arithmetic instruction with floating-point numbers.
- The value gradually drops below the permitted range when processing an arithmetic instruction (floating-point arithmetic) with overflow.
- After the execution of a comparison instruction, the content of accumulator 2 is greater than or equal to the content of accumulator 1.
- An instruction for word logic operation was executed.
- A shift instruction was executed.

In all other cases the jump is not executed and the processing of the program is resumed with the next instruction.

Syntax

Use the following syntax for the "Jump if the result is greater than or equal to zero" instruction:

```
JPZ <jump label>
```

Parameters

The following table shows the parameters of the "Jump if the result is greater than or equal to zero" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

```
STL                         Explanation
L "Tag_Value_1"             // Load value of the operand "Tag_Value_1".
L "Tag_Value_2"             // Load value of the operand "Tag_Value_2".
-I                          // Subtract value of the operand "Tag_Value_2" from
                            the value of the operand "Tag_Value_1".
JPZ REGULAR                 // At a positive result value or null in accumulator
                            "1" jump to jump label "REGULAR" and continue program
                            processing from there
                            // If the result in accumulator 1 is negative, execute
                            the next instruction.
AN "MyTag_1"                // Query whether the value of operand "MyTag_1" is "0"
                            and AND.
S "Tag_Output_1"            // If RLO is "1", set the operand "Tag_Output_1" to
                            "1".
```

STL	Explanation
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
REGULAR: AN "MyTag_2"	// Jump label "REGULAR"
	// Query whether the value of operand "MyTag_2" is "0" and AND.
S "Tag_Output_2"	// If RLO is "1", set the operand "Tag_Output_2" to "1".
NEXT: A "MyTag_3"	// Jump label "NEXT"
	// Query whether the value of operand "MyTag_3" is "1" and AND.
A "MyTag_4"	// Query whether the value of operand "MyTag_4" is "1" and AND.

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting status bits (Page 2455)

Setting status bits in program control operation instructions (Page 2460)

Querying status bits with jump instructions (Page 2464)

JMZ: Jump if the result is less than or equal to zero

Description

Use the "Jump if the result is less than or equal to zero" instruction to interrupt linear program execution depending on the status bit CC 1 and continue it at the point marked by the specified jump label.

The jump to the specified jump label is only executed if the signal state of the status bit CC 1 is "0". The situation occurs when one of the following conditions applies:

- After processing an arithmetic instruction without overflow, the value of accumulator 1 is less than or equal to zero.
- A negative overflow occurs when processing an arithmetic instruction with integers (+I, *I, +D, *D).
- A positive overflow occurs when processing an arithmetic instruction with integers (+I, -I, +D, -D, NEGI, NEGD).
- A negative overflow occurs when processing an arithmetic instruction with floating-point numbers.
- The value gradually drops below the permitted range when processing an arithmetic instruction (floating-point arithmetic) with overflow.

- After the execution of a comparison instruction, the content of accumulator 2 is less than or equal to the content of accumulator 1.
- The value of accumulator 1 is equal to zero after processing a word logic.
- The value of the last shifted bit is equal to zero after processing a write instruction.

In all other cases the jump is not executed and the processing of the program is resumed with the next instruction.

Syntax

Use the following syntax for the "Jump if the result is less than or equal to zero" instruction:

```
JMZ <jump label>
```

Parameters

The following table shows the parameters of the "Jump if the result is less than or equal to zero" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1".
L "Tag_Value_2"	// Load value of the operand "Tag_Value_2".
-I	// Subtract value of the operand "Tag_Value_2" from the value of the operand "Tag_Value_1".
JMZ MyLABEL	// At a negative result value or null in accumulator "1" jump to jump label "MyLABEL" and continue program processing from there
	// If the result in accumulator 1 is positive, execute the next instruction.
AN "MyTag_1"	// Query whether the value of operand "MyTag_1" is "0" and AND.
S "Tag_Output_1"	// If RLO = "1", set the operand "Tag_Output_1" to "1"
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
MyLABEL: AN "MyTag_2"	// Jump label "MyLABEL"
	// Query whether the value of operand "MyTag_2" is "0" and AND.
S "Tag_Output_2"	// If RLO="1" set the operand "Tag_Output_2" to "1"
NEXT: A "MyTag_3"	// Jump label "NEXT"

STL	Explanation
	// Query whether the value of operand "MyTag_3" is "1" and AND.
A "MyTag_4"	// Query whether the value of operand "MyTag_4" is "1" and AND.

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Basic information on the status word (Page 2453)

Setting status bits (Page 2455)

Setting status bits in program control operation instructions (Page 2460)

Querying status bits with jump instructions (Page 2464)

JUO: Jump if the result is invalid

Description

Use the "Jump if the result is invalid" instruction to interrupt linear program execution depending on the status bits CC 0 and CC 1 and continue at the point marked by the specified jump label.

The jump to the specified jump label is only executed if the signal state of the status bits CC 0 and CC 1 is "1". The situation occurs when one of the following conditions applies:

- It is divided by zero with a mathematical instruction (/I, /D, MOD).
- When processing an arithmetic instruction with floating-point numbers an overflow occurs and the result value is an invalid floating-point number.
- When processing a comparative instruction with floating-point numbers an invalid floating-point number was used or occurred as a result value.

In all other cases the jump is not executed and the processing of the program is resumed with the next instruction.

Syntax

Use the following syntax for the instruction "Jump if the result is invalid":

```
JUO <jump label>
```

Parameters

The following table shows the parameters of the "Jump if the result is invalid" instruction:

Parameters	Description
<jump label>	Symbolic name of the jump destination

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand "Tag_Value_1".
L "Tag_Value_2"              // Load value of the operand "Tag_Value_2".
/I                           // Divide value of the operand "Tag_Value_1" by value
                             of the operand "Tag_Value_2".
JUO ERROR                    // At a division by zero, jump to jump label "ERROR"
                             and continue with program processing from there
                             // Otherwise execute next instruction.
T "Tag_Result"               // Transfer contents of accumulator 1 to the operand
                             "Tag_Result".
A "MyTag_1"                  // Query whether the value of operand "MyTag_1" is "1"
                             and AND.
R "MyTag_1"                  // If RLO = "1", reset the operand "MyTag_1" to "0"
JU NEXT                      // Jump to jump label "NEXT" and resume execution of
                             the program there.
ERROR: AN "MyTag_1"          // Jump label "ERROR"
                             // Query whether the value of operand "MyTag_1" is "0"
                             and AND.
S "MyTag_1"                  // If RLO is "1", set the operand "MyTag_1" to "1".
NEXT: A "MyTag_3"            // Jump label "NEXT"
                             // Query whether the value of operand "MyTag_3" is "1"
                             and AND.
A "MyTag_4"                  // Query whether the value of operand "MyTag_4" is "1"
                             and AND.
```

See also

Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)
Basic information on the status word (Page 2453)
Setting status bits (Page 2455)
Setting status bits in program control operation instructions (Page 2460)
Querying status bits with jump instructions (Page 2464)

JL: Define jump list

Description

Use the instruction "Define jump list" to program a list that consists of several entries of the instruction "Unconditional jump" (JU). The list begins directly after the instruction "Define jump list" and may contain up to 255 entries. The numbering of the jump functions in the list begins at zero. The jump list must be programmed without any gaps. The end of the jump list is marked by a jump label that is specified for the "Define jump list" instruction.

Which jump function of the list executes, depends on the value in the right byte of accumulator 1. If, for example, the accumulator 1 has the value "0", the first jump function executes. If the value in accumulator 1 is "1", the second jump function executes. If the value in accumulator 1 is greater than the number of list entries, the "Define jump list" instruction references the end of the list.

The "Jump distributor" instruction is processed independently of conditions and does not affect the status bits.

Syntax

You program the instruction "Define jump list" according to the following general outline:

```
L <jump number>
JL <jumplabel_end>
JU <jump label>
JU <jump label>
.... <jump label>
<jumplabel_end>:
```

The operand "jump number" contains the number of the instruction "Unconditional jump" (JU) to be executed. This number must be loaded in accumulator 1. The end of the jump list is marked by the jump label "jumplabel_end".

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value"	// Load jump number in accumulator 1.
JL END	// Start of the jump list
JU MyLABEL_1	// If the value is "0", execute in accumulator 1 and jump to jump label "MyLABEL_1".
JU MyLABEL_2	// If the value is "1", execute in accumulator 1 and jump to jump label "MyLABEL_2"
JU MyLABEL_3	// If the value is "1", execute in accumulator 2 and jump to jump label "MyLABEL_3"
END: L "MyTag_1"	// End of the jump list

STL	Explanation
	// Load content of the operand "MyTag_1" in accumulator 1.
L "MyTag_2"	// Load content of the operand "MyTag_2" in accumulator 1.
+I	// Add values
T "Tag_Output_2"	// Load contents of accumulator 1 in operand "Tag_Output_2".
MyLABEL_1: ...	// Jump label "MyLABEL_1"
....	// Any program
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
MyLABEL_2: ...	// Jump label "MyLABEL_2"
....	// Any program
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
MyLABEL_3: ...	// Jump label "MyLABEL_3"
....	// Any program
JU NEXT	// Jump to jump label "NEXT" and resume execution of the program there.
NEXT: ...	// Jump label "NEXT"

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Example of a step sequence (Page 3142)

Setting status bits in program control operation instructions (Page 2460)

Querying status bits with jump instructions (Page 2464)

Basic information on the status word (Page 2453)

LOOP: Loop

Description

Use the "Loop" instruction to program loops in the program.

The instruction interprets the right word of accumulator 1 as an unsigned 16-bit integer in the range from 0 to 65 535. During execution the instruction first decrements the content of accumulator 1 by one. If the value in accumulator 1 after the decrement is not zero, the jump is executed at the specified jump label. If the value is zero, the jump is not executed and the processing of the program is resumed with the next instruction. The value in accumulator 1 therefore specifies the number of program loops that have to be executed. This number has to be saved in a loop counter.

The "Loop" instruction does not affect any status bits.

> **WARNING**
>
> If the loop counter is not preset or is preset with the value "0", the loop counter continues to decrement into negative numbers so that an endless loop results that can change the CPU to STOP mode.

Syntax

The "Loop" instruction is programmed according to the following general outline:

```
L <number>
<jump label>:
T <loop counter>
... Any program
L <loop counter>
LOOP <jump label>
```

The operand "number" contains the number of the loop cycles. The operand "Loop counter" contains the number of the loop cycles still to be executed. At the end of the program loop, the content of the loop counter is loaded in accumulator 1 and decremented by one by the instruction. The jump to the jump label at the start of the program loop is executed if the value in the accumulator is not equal to zero after the decrement.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value"	// Load number of time loop will execute into the right word of accumulator 1.
START: T "Tag_Counter"	// Start of the jump list
	// Transfer contents of accumulator 1 to the loop counter.
L "MyTag_1"	// Load value of the operand "MyTag_1".
L "MyTag_2"	// Load value of the operand "MyTag_2".
*D	// Multiply values
T "MyTag_1"	// Transfer the result of the multiplication to the operand "MyTag_1".
L "Tag_Counter"	// Load value of the loop counter into accumulator 1.
LOOP START	// Decrement value in accumulator 1 by one.
	// Jump to the start of the program loop if the value is not equal to zero.
	// If the value is zero, execute the next instruction.
L "MyTag_2"	// Load value of the operand "MyTag_2".

STL	Explanation
L 100	// Load the value 100.
>I	// Compare whether the value of the operand "MyTag_2" is greater than 200.
= "MyTag_3"	// Write comparison result to the operand "MyTag_3".

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits in program control operation instructions (Page 2460)

Querying status bits with jump instructions (Page 2464)

Basic information on the status word (Page 2453)

Data blocks

OPN: Open global data block

Description

You can open a global data block (DB) with the "Open global data block" instruction. The number of the data block is transferred to the DB register. Subsequent DB commands access the relevant blocks depending on the content of the register.

The "Open global data block" instruction is processed without any conditions and does not affect either the result of logic operation or the accumulator contents.

Syntax

Use the following syntax for the "Open global data block" instruction:

```
OPN <data block>
```

Parameters

The following table shows the parameters of the "Open global data block" instruction:

Operand	Declaration	Data block type	Description
<data block>	-	DB	Data block that will be opened.

Example

The following example shows how the instruction works:

STL	Explanation
OPN "GlobalDataBlock"	// Open global data block.
L %DBW0	// Load data word DBW0 of the opened global data block into accumulator 1.
T "MyTag"	// Transfer contents of accumulator 1 to the operand "MyTag_1".

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits in program control operation instructions (Page 2460)

Basic information on the status word (Page 2453)

OPNDI: Open instance data block

Description

With the "Open instance data block" instruction, you open an instance data block (DI). The number of the data block is transferred to the DI register. Subsequent DI commands access the relevant blocks depending on the content of the register.

The "Open instance data block" instruction is processed without any conditions and does not affect either the result of logic operation or the accumulator contents.

Syntax

Use the following syntax for the "Open instance data block" instruction:

```
OPNDI <data block>
```

Parameters

The following table shows the parameters of the "Open instance data block" instruction:

Operand	Declaration	Data block type	Description
<data block>	-	DI	Data block that will be opened.

Example

The following example shows how the instruction works:

STL	Explanation
OPNDI "InstanceDataBlock"	// Open instance data block
L %DIW0	// Load data word DIW0 of the opened instance data block into accumulator 1.
T "MyTag"	// Transfer contents of accumulator 1 to the operand "MyTag".

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits in program control operation instructions (Page 2460)

Basic information on the status word (Page 2453)

CDB: Swap data block register

Description

Use the "Swap data block register" instruction to swap the contents of the data block register. The instruction executes regardless of conditions and does not affect the status bits.

Syntax

Use the following syntax for the "Swap data block register" instruction:

```
CDB
```

Example

The following example shows how the instruction works:

STL	Explanation
OPN "GlobalDataBlock"	// Open global data block
OPNDI "InstanceDataBlock"	// Open instance data block
CDB	// Swap data block register
	// The DB register points to "InstanceDataBlock" and the DI register points to "GlobalDataBlock".
L %DIW0	// Load DW0 from "GlobalDataBlock" into accumulator 1.

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

L DBLG: Load the length of a global data block into accumulator 1

Description

Use the "Load the length of a global data block into accumulator 1" instruction to load the length of a global data block opened using the data block register into accumulator 1. The previous contents of accumulator 1 are shifted to accumulator 2.

If no global data block was opened via the data block register prior to the processing of the instruction, the value "0" is loaded into accumulator 1. The length of the data block is equal to the number of the data bytes.

The status bits are not influenced by the instruction.

Syntax

Use the following syntax for the "Load the length of a global data block into accumulator 1" instruction:

```
L DBLG
```

Example

The following example shows how the instruction works:

STL	Explanation
OPN "GlobalDataBlock"	// Open global data block
L DBLG	// Load the length of the opened data block into accumulator 1.
L "MyTag_1"	// Load comparison value
<D	// Compare whether the length of the data block is less than the value of the operand "MyTag_1".
= "MyTag_2"	// Write query result to the "MyTag_2" operand.

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

L DBNO: Load the number of a global data block into accumulator 1

Description

Use the "Load the number of a global data block into accumulator 1" instruction to load the number of a global data block opened using the data block register into accumulator 1. The previous contents of accumulator 1 are shifted to accumulator 2.

If no global data block was opened via the data block register prior to the processing of the instruction, the value "0" is loaded into accumulator 1.

The status bits are not influenced by the instruction.

Syntax

Use the following syntax for the "Load the number of a global data block into accumulator 1" instruction:

```
L DBNO
```

Example

The following example shows how the instruction works:

STL	Explanation
OPN "GlobalDataBlock"	// Open global data block
L DBNO	// Load the number of the opened data block into accumulator 1.
L "MyTag_1"	// Load comparison value
==I	// Compare whether the number of the data block is equal to the value of the operand "MyTag_1".
= "MyTag_2"	// Write query result to the "MyTag_2" operand.

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

L DILG: Load the length of an instance data block into accumulator 1

Description

Use the "Load the length of an instance data block into accumulator 1" instruction to load the length of an instance data block opened using the data block register into accumulator 1. The previous contents of accumulator 1 are shifted to accumulator 2.

If no instance data block was opened via the data block register prior to the instruction executing, the value "0" is loaded into accumulator 1. The length of the data block is equal to the number of the data bytes.

The status bits are not influenced by the instruction.

Syntax

Use the following syntax for the "Load the length of an instance data block into accumulator 1" instruction:

```
L DILG
```

Example

The following example shows how the instruction works:

STL	Explanation
OPNDI "InstanceDataBlock"	// Open instance data block
L DILG	// Load the length of the opened data block into accumulator 1.
L "MyTag_1"	// Load comparison value
<I	// Compare whether the length of the data block is less than the value of the operand "MyTag_1".
= "MyTag_2"	// Write query result to the "MyTag_2" operand.

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

L DINO: Load the number of an instance data block into accumulator 1

Description

Use the "Load the number of an instance data block into accumulator 1" instruction to load the number of an instance data block opened using the data block register into accumulator 1. The previous contents of accumulator 1 are shifted to accumulator 2.

If no instance data block was opened via the data block register prior to the instruction executing, the value "0" is loaded into accumulator 1.

The status bits are not influenced by the instruction.

Syntax

Use the following syntax for the "Load the number of an instance data block into accumulator 1" instruction:

```
L DINO
```

Example

The following example shows how the instruction works:

STL	Explanation
OPNDI "InstanceDataBlock"	// Open instance data block
L DINO	// Load the number of the opened data block into accumulator 1.
L "MyTag_1"	// Load comparison value
==I	// Compare whether the number of the data block is equal to the value of the operand "MyTag_1".
= "MyTag_2"	// Assign query result to the "MyTag_2" operand.

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Code blocks

BE: Block end

Description

Use the "Block end" instruction to end the block currently being processed and go to the point in the program where the block is called. The program processing is then resumed with the instruction that is located directly after the block call.

When the block ends the current result of logic operation is transferred to the calling block.

The "Block end" instruction is executed independently of conditions. If the processing of the instruction is skipped by a jump instruction, the current program execution does not end but instead resumes at the jump destination within the block.

Syntax

Use the following syntax for the "Block end" instruction:

```
BE
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
SIN	// Form sine value
	// Store result in accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.
L "Tag_Value_3"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_3" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product in accumulator 1.
T "Tag_Result"	// Transfer contents of accumulator 1 to the operand "Tag_Result".
BE	// End block

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits in program control operation instructions (Page 2460)

Basic information on the status word (Page 2453)

BEC: Conditional block end

Description

Use the "Conditional block end" instruction to end the block currently being processed depending on the result of logic operation (RLO) and to go to point in the program where the block is called.

If the RLO is "1", the instruction is executed. The block currently being processed is terminated and the program processing is resumed in the calling block. The program processing is then resumed with the instruction that is located directly after the block call.

When the block ends the current result of logic operation is transferred to the calling block. The MCR dependency establishes itself in the calling block in the same way as it was before the block call. The data blocks that were open when the block was called are re-opened.

If the RLO is "0" when the instruction is processed, the instruction is not executed. In this case the CPU sets the RLO to "1" and executes the next instruction.

If the processing of the instruction is skipped by a jump instruction, the current program execution does not end but instead resumes at the jump destination within the block.

Syntax

Use the following syntax for the "Conditional block end" instruction:

```
BEC
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
>I	// Compare whether the value of accumulator 2 is greater than the value of accumulator 1.
A "Tag_Input"	// Query the value of operand "Tag_Input" for "1" and AND with the RLO.
BEC	// End block if condition satisfied (RLO = "1").
	// If the condition is not satisfied(RLO = "0"), execute the next instruction.
T "Tag_Result"	// Transfer contents of accumulator 1 to the operand "Tag_Result".

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Example: Word logic instructions (Page 3140)

Setting status bits in program control operation instructions (Page 2460)

Basic information on the status word (Page 2453)

BEU: Unconditional block end

Description

Use the "Unconditional block end" instruction to end the block currently being processed and go to the point in the program where the block is called. The program processing is then resumed with the instruction that is located directly after the block call.

When the block ends the current result of logic operation is transferred to the calling block. The MCR dependency establishes itself in the calling block in the same way as it was before the block call. The data blocks that were open when the block was called are re-opened.

The "Unconditional block end" instruction is executed independently of conditions. In contrast to the "Block end" instruction, you can program the "Unconditional block end" instruction more than once in a block. If the processing of the instruction is skipped by a jump instruction, the current program execution does not end but instead resumes at the jump destination within the block.

Syntax

Use the following syntax for the "Unconditional block end" instruction:

```
BEU
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
>I	// Compare whether the value of accumulator 2 is greater than the value of accumulator 1.
A "Tag_Input"	// Query the value of operand "Tag_Input" for "1" and AND with the RLO.
JC NEXT	// If the condition is satisfied (RLO = "1"), continue program execution at jump label"NEXT".
	// If the condition is not satisfied(RLO = "0"), execute the next instruction.
BEU	// End block
NEXT: T "Tag_Result"	// Jump label "NEXT"
	// Transfer contents of accumulator 1 to the operand "Tag_Result".

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Example of a step sequence (Page 3142)

Setting status bits in program control operation instructions (Page 2460)

Basic information on the status word (Page 2453)

CALL: Call block

Description

Use the "Call block" instruction to call the following block types in the program:

- Functions
- Function blocks

The "Call block" instruction executes without any conditions. After processing of the instruction, program execution is resumed in the called block.

You can feed a called block with data. The data is transferred via the block parameters. The parameters of the called block are listed according to the call instruction in the called block. You can assign the required current parameters to these parameters. During execution of the "Call block" instruction, the data is transferred to the called block. By transferring the data, the content of the status word and the address and data block registers are changed.

Those parameters of a function block that are not fed with parameters, retain their current value. All parameters must be fed when functions are called. If a called block has no parameters, no parameter list is displayed.

If the called block requires an instance data block, give it to the block, separated by a comma when called. The specified data block must be created before the call.

When the called block is processed, the CPU switches back to the calling block and resumes the processing of this block after the call instruction.

The MCR dependency establishes itself in the calling block in the same way as it was before the block call. The data blocks that were open when the block was called are re-opened.

Syntax

When you call a function, use the following syntax for the "Call block" instruction:

```
CALL <FC>
```

When you call a function block, use the following syntax for the "Call block" instruction:

```
CALL <FB>, <FB_DB>
```

Example

The following example shows how the instruction works:

```
STL                                       Explanation
CALL "MyFunction"                         // Call function "MyFunction"
      Input_1 := "Tag_Input_1",           // Assign actual parameters
      Input_2 := "Tag_Input_2"
      Output_1 := "Tag_Output_1"
      Output_2 := "Tag_Output_1"
CALL "MyFunctionBlock", "MyFB_DB"         // Call function block "MyFunctionBlock"
      Value_1 := "Tag_Value_1"            // Assign actual parameters
      Value_2 := "Tag_Value_2"
      Output := "Tag_Output"
CALL "LIMIT"                              // Call "Set limit value" instruction
      MN := "Tag_LowLimit"                // Assign actual parameters
      IN := "Tag_InputValue"
      MX := "Tag_HighLimit"
      OUT := "Tag_Output"
CALL "CTU", "CTU_DB"                      // Call "Count up" counter
      CU := "Tag_StartCTU"                // Assign actual parameters
      R := "Tag_ResetCounter"
      PV := "Tag_PresetValue"
      Q := "Tag_CounterStatus"
      CV := "Tag_CounterValue"
```

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits in program control operation instructions (Page 2460)

Basic information on the status word (Page 2453)

CC: Conditional block call

Description

Use the "Conditional block call" instruction to call functions (FCs) and function blocks (FBs) depending on the result of logic operation (RLO).

The instruction is only executed if the current result of logic operation (RLO) before execution of the instruction is "1". After processing of the instruction, program execution is resumed in the called block. When the called block is processed, the CPU switches back to the calling block and resumes the processing of this block after the call instruction.

The status bits OS and /FC are reset to "0" when there is a block change. The status bits CC 0, CC 1 and OV are not affected.

The MCR dependency establishes itself in the calling block in the same way as it was before the block call. The data blocks that were open when the block was called are re-opened.

The contents of the accumulators and the address registers are not changed by the "Conditional block call" instruction.

If the current RLO is "0", the instruction as well as the block call are not executed and the RLO is set to "1".

Syntax

When you call a function, use the following syntax for the "Conditional block call" instruction:

```
CC <FC>
```

When you call a function block, use the following syntax for the "Conditional block call" instruction:

```
CC <FB>
```

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input_1"	// Query the value of operand "Tag_Input_1" for "1" and AND with the current RLO.
CC "MyFunction"	// If condition is satisfied (RLO = "1"), call the function MyFunction.
	// If the condition is not satisfied(RLO = "0"), execute the next instruction.
A "Tag_Input_2"	// Query the value of operand "Tag_Input_2" for "1" and AND with the current RLO.
CC "MyFunctionBlock"	// If condition is satisfied (RLO = "1"), call the function block MyFunctionBlock.
	// If the condition is not satisfied(RLO = "0"), execute the next instruction.

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits in program control operation instructions (Page 2460)

Basic information on the status word (Page 2453)

UC: Unconditional block call

Description

You can use the "Unconditional block call" instruction to call functions (FCs) and function blocks (FBs) that have no parameters and no instance data blocks.

The instruction is executed regardless of conditions. After processing of the instruction, program execution is resumed in the called block. When the called block is processed, the CPU switches back to the calling block and resumes the processing of this block after the call instruction.

The status bits OS and /FC are reset to "0" when there is a block change. The status bits CC 0, CC 1 and OV are not affected.

The MCR dependency establishes itself in the calling block in the same way as it was before the block call. The data blocks that were open when the block was called are re-opened.

The contents of the accumulators and the address registers are not changed by the "Unconditional block call" instruction.

Syntax

When you call a function, use the following syntax for the "Unconditional block call" instruction:

```
UC <FC>
```

When you call a function block, use the following syntax for the "Unconditional block call" instruction:

```
UC <FB>
```

Example

The following example shows how the instruction works:

STL	Explanation
A "Tag_Input_1"	// Query the value of operand "Tag_Input_1" for "1" and AND with the current RLO.
UC "MyFunction"	// Call function "MyFunction".
A "Tag_Input_2"	// Query the value of operand "Tag_Input_2" for "1" and AND with the current RLO.
UC "MyFunctionBlock"	// Call function block "MyFunctionBlock".

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits in program control operation instructions (Page 2460)

Basic information on the status word (Page 2453)

Word logic operations

AW: AND logic operation word by word

Description

Use the instruction "AND logic operation word by word" to AND the value of the right word in accumulator 1 with the value of the right word in accumulator 2 or a specified constant bit by bit. The result is saved in the right word of accumulator 1. The contents of the left word in accumulator 1 remain unchanged.

The instruction links bit 0 of accumulator 1 with bit 0 of accumulator 2 or a constant and saves the result to bit 0 of accumulator 1. Bits 1 to 15 are linked in the same manner.

The result bit has signal state "1" if both of the bits in the logic operation also have signal state "1". If one of the two bits of the logic operation has signal state "0", the corresponding result bit is reset.

The following table shows how the result is formed by an AND word logic operation:

Accumulator 2 / constant	0	0	1	1
Accumulator 1	0	1	0	1
Result	0	0	0	1

The instruction is executed independent of conditions and does not affect the result of logic operation.

The instruction affects the status bits CC 0, CC 1 and OV.

Syntax

If the operation is used with a constant, use the following syntax for the "AND logic operation word by word" instruction:

```
AW <constant>
```

If the operation is used with the value of the right word in accumulator 2, use the following syntax for the "AND logic operation word by word" instruction:

```
AW
```

Parameters

The following table shows the parameters of the "AND logic operation word by word" instruction:

Parameters	Data type	Description
<constant>	WORD	Value that is ANDed with the value in the right word of accumulator 1.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand into the right word of
                             accumulator 1.
AW W#16#F6B5                 // AND the value of the right word in accumulator 1
                             with the constant (W#16#F6B5).
                             // Save the result in the right word of accumulator 1.
T "Tag_Result_1"             // Transfer the result to operand "Tag_Result_1".
L "Tag_Value_2"              // Load value of the operand into the right word of
                             accumulator 1.
L "Tag_Value_3"              // Shift the contents of accumulator 1 to accumulator
                             2.
                             // Load value of the operand "Tag_Value_3" into the
                             right word of accumulator 1.
AW                           // AND the value of the right word in accumulator 2
                             ("Tag_Value_2") with the value of the right word in
                             accumulator 1 ("Tag_Value_3")
                             // Save the result in the right word of accumulator 1.
T "Tag_Result_2"             // Transfer the result to operand "Tag_Result_2".
```

The following table shows how the instruction works using specific operand values:

Operand	Value			
Tag_Value_1	0101	1001	0011	1011
Constant (W#16#F6B5)	1111	0110	1011	0101
Tag_Result_1	0101	0000	0011	0001
Tag_Value_2	0110	1100	0010	1010
Tag_Value_3	1101	1010	1001	0011
Tag_Result_2	0100	1000	0000	0010

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Example: Word logic instructions (Page 3140)

Setting the status bit for word logic operations (Page 2462)

Basic information on the status word (Page 2453)

OW: OR logic operation word by word

Description

Use the instruction "OR logic operation word by word" to OR the value of the right word in accumulator 1 with the value of the right word in accumulator 2 or a specified constant bit by bit. The result is saved in the right word of accumulator 1. The contents of the left word in accumulator 1 remain unchanged.

The instruction links bit 0 of accumulator 1 with bit 0 of accumulator 2 or a constant and saves the result to bit 0 of accumulator 1. Bits 1 to 15 are linked in the same manner.

The result bit has signal state "1" when at least one of the two bits in the logic operation has signal state "1". If both bits of the logic operation have signal state "0", the corresponding result bit is reset to "0".

The following table shows how the result is formed by an OR word logic operation:

Accumulator 2 / constant	0	0	1	1
Accumulator 1	0	1	0	1
Result	0	1	1	1

The instruction is executed independent of conditions and does not affect the result of logic operation.

The instruction affects the status bits CC 0, CC 1 and OV.

Syntax

If the operation is used with a constant, use the following syntax for the "OR logic operation word by word" instruction:

```
OW <constant>
```

If the operation is used with the value of the right word in accumulator 2, use the following syntax for the "OR logic operation word by word" instruction:

```
OW
```

Parameters

The following table shows the parameters of the "OR logic operation word by word" instruction:

Parameters	Data type	Description
<constant>	WORD	Value that is ORed with the value in the right word of accumulator 1.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand into the right word of
                             accumulator 1.
OW W#16#F6B5                 // OR the value of the right word in accumulator 1
                             with the constant (W#16#F6B5).
                             // Save the result in the right word of accumulator 1.
T "Tag_Result_1"             // Transfer the result to operand "Tag_Result_1".
L "Tag_Value_2"              // Load value of the operand into the right word of
                             accumulator 1.
L "Tag_Value_3"              // Shift the contents of accumulator 1 to accumulator
                             2.
                             // Load value of the operand "Tag_Value_3" into the
                             right word of accumulator 1.
OW                           // OR the value of the right word in accumulator 2
                             ("Tag_Value_2") with the value of the right word in
                             accumulator 1 ("Tag_Value_3")
                             // Save the result in the right word of accumulator 1.
T "Tag_Result_2"             // Transfer the result to operand "Tag_Result_2".
```

The following table shows how the instruction works using specific operand values:

Operand	Value			
Tag_Value_1	0101	1001	0011	1011
Constant (W#16#F6B5)	1111	0110	1011	0101
Tag_Result_1	1111	1111	1011	1111
Tag_Value_2	0110	1100	0010	1010
Tag_Value_3	1101	1010	1001	0011
Tag_Result_2	1111	1110	1011	1011

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Example: Word logic instructions (Page 3140)

Setting the status bit for word logic operations (Page 2462)

Basic information on the status word (Page 2453)

XOW: EXCLUSIVE OR logic operation word by word

Description

Use the instruction "EXCLUSIVE OR logic operation word by word" to EXCLUSIVE OR the value of the right word in accumulator 1 with the value of the right word in accumulator 2 or a specified constant bit by bit. The result is saved in the right word of accumulator 1. The contents of the left word in accumulator 1 remain unchanged.

The instruction links bit 0 of accumulator 1 with bit 0 of accumulator 2 or a constant and saves the result to bit 0 of accumulator 1. Bits 1 to 15 are linked in the same manner.

The result bit has signal state "1" if the signal state of the two bits to be linked is different. If both of the bits of the logic operation have the same signal state, the corresponding result bit is reset to "0".

The following table shows how the result is formed by an EXCLUSIVE OR word logic operation:

Accumulator 2 / constant	0	0	1	1
Accumulator 1	0	1	0	1
Result	0	1	1	0

The instruction is executed independent of conditions and does not affect the result of logic operation.

The instruction affects the status bits CC 0, CC 1 and OV.

Syntax

If the operation is used with a constant, use the following syntax for the "EXCLUSIVE OR logic operation word by word" instruction:

```
XOW <constant>
```

If the operation is used with the value of the right word in accumulator 2, use the following syntax for the "EXCLUSIVE OR logic operation word by word" instruction:

```
XOW
```

Parameters

The following table shows the parameters of the "EXCLUSIVE OR logic operation word by word" instruction:

Parameters	Data type	Description
<constant>	WORD	Value that is XORed with the value in the right word of accumulator 1.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand into the right word of
                             accumulator 1.
XOW W#16#F6B5                // XOR the value of the right word in accumulator 1
                             with the constant (W#16#F6B5).
                             // Save the result in the right word of accumulator 1.
T "Tag_Result_1"             // Transfer the result to operand "Tag_Result_1".
L "Tag_Value_2"              // Load value of the operand into the right word of
                             accumulator 1.
L "Tag_Value_3"              // Shift the contents of accumulator 1 to accumulator
                             2.
                             // Load value of the operand "Tag_Value_3" into the
                             right word of accumulator 1.
XOW                          // XOR the value of the right word in accumulator 2
                             ("Tag_Value_2") with the value of the right word in
                             accumulator 1 ("Tag_Value_3")
                             // Save the result in the right word of accumulator 1.
T "Tag_Result_2"             // Transfer the result to operand "Tag_Result_2".
```

The following table shows how the instruction works using specific operand values:

Operand	Value			
Tag_Value_1	0101	1001	0011	1011
Constant (W#16#F6B5)	1111	0110	1011	0101
Tag_Result_1	1010	1111	1000	1110
Tag_Value_2	0110	1100	0010	1010
Tag_Value_3	1101	1010	1001	0011
Tag_Result_2	1011	0110	1011	1001

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bit for word logic operations (Page 2462)

Basic information on the status word (Page 2453)

AD: AND logic operation double word by double word

Description

Use the instruction "AND logic operation double word by double word" to AND the contents of accumulator 1 with the contents of accumulator 2 bit by bit. The result is stored in accumulator 1.

The instruction links bit 0 of accumulator 1 with bit 0 of accumulator 2 or a constant and saves the result to bit 0 of accumulator 1. Bits 1 to 31 are linked in the same manner.

The result bit has signal state "1" if both of the bits in the logic operation also have signal state "1". If one of the two bits of the logic operation has signal state "0", the corresponding result bit is reset.

The following table shows how the result is formed by an AND word logic operation:

Accumulator 2 / constant	0	0	1	1
Accumulator 1	0	1	0	1
Result	0	0	0	1

The instruction is executed independent of conditions and does not affect the result of logic operation.

The instruction affects the status bits CC 0, CC 1 and OV.

Syntax

If the operation is used with a constant, use the following syntax for the "AND logic operation double word by double word" instruction:

```
AD <constant>
```

When using the operation with the value of accumulator 2, use the following syntax for the "AND logic operation double word by double word" instruction:

```
AD
```

Parameters

The following table shows the parameters of the instruction "AND logic operation double word by double word":

Parameters	Data type	Description
<constant>	DWORD	Value that is ANDed with the value in accumulator 1.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand into accumulator 1.
AD DW#16#39C657AC            // Link the value in accumulator 1 with the constant
                             (DW#16#39C657AC) according to AND.
                             // Store result in accumulator 1.
T "Tag_Result_1"             // Transfer the result to operand "Tag_Result_1".
L "Tag_Value_2"              // Load value of the operand into accumulator 1.
L "Tag_Value_3"              // Shift the contents of accumulator 1 to accumulator
                             2.
                             // Load value of the operand "Tag_Value_3" in
                             accumulator 1.
AD                           // AND the value of accumulator 2 Tag_Value_2 with the
                             value of accumulator 1 ("Tag_Value_3")
                             // Store result in accumulator 1.
T "Tag_Result_2"             // Transfer the result to operand "Tag_Result_2".
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	0101	1111	0110	0100	1001	1101	0011	1011
Constant (DW#16#39C657AC)	0011	1001	1100	0110	0101	0111	1010	1100
Tag_Result_1	0001	1001	0100	0100	0001	0101	0010	1000
Tag_Value_2	0110	0101	0100	0011	0101	1101	0010	1011
Tag_Value_3	0011	1001	1100	0110	0101	0111	1010	1100
Tag_Result_2	0010	0001	0100	0010	0101	0101	0010	1010

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bit for word logic operations (Page 2462)

Basic information on the status word (Page 2453)

OD: OR logic operation double word by double word

Description

Use the instruction "OR logic operation double word by double word" to OR the contents of accumulator 1 with the contents of accumulator 2 bit by bit. The result is stored in accumulator 1.

The instruction links bit 0 of accumulator 1 with bit 0 of accumulator 2 or a constant and saves the result to bit 0 of accumulator 1. Bits 1 to 31 are linked in the same manner.

The result bit has signal state "1" when at least one of the two bits in the logic operation has signal state "1". If both bits of the logic operation have signal state "0", the corresponding result bit is reset to "0".

The following table shows how the result is formed by an OR word logic operation:

Accumulator 2 / constant	0	0	1	1
Accumulator 1	0	1	0	1
Result	0	1	1	1

The instruction is executed independent of conditions and does not affect the result of logic operation.

The instruction affects the status bits CC 0, CC 1 and OV.

Syntax

If the operation is used with a constant, use the following syntax for the "OR logic operation double word by double word" instruction:

```
OD <constant>
```

When using the operation with the value of accumulator 2, use the following syntax for the "OR logic operation double word by double word" instruction:

```
OD
```

Parameters

The following table shows the parameters of the "OR logic operation double word by double word" instruction:

Parameters	Data type	Description
<constant>	DWORD	Value that is ORed with the value in accumulator 1.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand into accumulator 1.
OD DW#16#39C657AC            // Link the value in accumulator 1 with the constant
                             (DW#16#39C657AC) according to OR.
                             // Store result in accumulator 1.
T "Tag_Result_1"             // Transfer the result to operand "Tag_Result_1".
L "Tag_Value_2"              // Load value of the operand into accumulator 1.
L "Tag_Value_3"              // Shift the contents of accumulator 1 to accumulator
                             2.
                             // Load value of the operand "Tag_Value_3" in
                             accumulator 1.
OD                           // OR the value of accumulator 2 Tag_Value_2 with the
                             value of accumulator 1 ("Tag_Value_3")
                             // Store result in accumulator 1
T "Tag_Result_2"             // Transfer the result to operand "Tag_Result_2".
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	0101	1111	0110	0100	1001	1101	0011	1011
Constant (DW#16#39C657AC)	0011	1001	1100	0110	0101	0111	1010	1100
Tag_Result_1	0111	1111	1110	0110	1101	1111	1011	1111
Tag_Value_2	0110	0101	0100	0011	0101	1101	0010	1011
Tag_Value_3	0011	1001	1100	0110	0101	0111	1010	1100
Tag_Result_2	0111	1101	1100	0111	0101	1111	1110	1111

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bit for word logic operations (Page 2462)

Basic information on the status word (Page 2453)

XOD: EXCLUSIVE OR logic operation double word by double word

Description

Use the instruction "EXCLUSIVE OR logic operation double word by double word" to XOR the contents of accumulator 1 with the contents of accumulator 2 bit by bit. The result is stored in accumulator 1.

The instruction links bit 0 of accumulator 1 with bit 0 of accumulator 2 or a constant and saves the result to bit 0 of accumulator 1. Bits 1 to 31 are linked in the same manner.

The result bit has signal state "1" if the signal state of the two bits to be linked is different. If both of the bits of the logic operation have the same signal state, the corresponding result bit is reset to "0".

The following table shows how the result is formed by an EXCLUSIVE OR word logic operation:

Accumulator 2 / constant	0	0	1	1
Accumulator 1	0	1	0	1
Result	0	1	1	0

The instruction is executed independent of conditions and does not affect the result of logic operation.

The instruction affects the status bits CC 0, CC 1 and OV.

Syntax

If the operation is used with a constant, use the following syntax for the "EXCLUSIVE OR logic operation double word by double word" instruction:

```
XOD <constant>
```

When using the operation with the value of accumulator 2, use the following syntax for the "EXCLUSIVE OR logic operation double word by double word" instruction:

```
XOD
```

Parameters

The following table shows the parameters of the "EXCLUSIVE OR logic operation double word by double word" instruction:

Parameters	Data type	Description
<constant>	DWORD	Value that is XORed with the value in accumulator 1

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand into accumulator 1.
XOD DW#16#39C657AC           // Link the value in accumulator 1 with the constant
                             (DW#16#39C657AC) according to XOR.
                             // Store result in accumulator 1.
T "Tag_Result_1"             // Transfer the result to operand "Tag_Result_1".
L "Tag_Value_2"              // Load value of the operand into accumulator 1.
L "Tag_Value_3"              // Shift the contents of accumulator 1 to accumulator
                             2.
                             // Load value of the operand "Tag_Value_3" in
                             accumulator 1.
XOD                          // XOR the value of accumulator 2 Tag_Value_2 with the
                             value of accumulator 1 ("Tag_Value_3")
                             // Store result in accumulator 1.
T "Tag_Result_2"             // Transfer the result to operand "Tag_Result_2".
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	0101	1111	0110	0100	1001	1101	0011	1011
Constant (DW#16#39C657AC)	0011	1001	1100	0110	0101	0111	1010	1100
Tag_Result_1	0110	0110	1010	0010	1100	1010	1001	0111
Tag_Value_2	0110	0101	0100	0011	0101	1101	0010	1011
Tag_Value_3	0011	1001	1100	0110	0101	0111	1010	1100
Tag_Result_2	0101	1100	1000	0101	0000	1010	1000	0111

See also

Overview of the valid data types (Page 2236)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bit for word logic operations (Page 2462)

Basic information on the status word (Page 2453)

Shift and rotate

Shift

SSI: Shift word by word with sign

Description

Use the instruction "Shift word by word with sign" to shift the right word (bits 0 to 15) of accumulator 1 bit by bit to the right. During the shifting, positions that become free are padded with the signal state of bit 15 (sign bit for INT numbers). The bits 16 to 31 of accumulator 1 remain unchanged.

The following options are available for specifying the number of bit positions by which the bits are to be shifted:

- Specification of a positive integer as parameter of the instruction. The specified number must be in a value range from 0 to 15.
- Specification of the value of the right byte in accumulator 2. The byte is interpreted as a positive integer.

The instruction is executed regardless of the RLO. It resets the status bit CC 0 to "0" and the status bit CC 1 to the signal state of the last bit that was shifted out.

If the specified number of places shifted is greater than 15, all bits in the right word of accumulator 1 are padded with the signal state of bit 15.

If the specified number of places shifted is equal to zero, the instruction is not executed. The status bits are not changed.

Syntax

If the number of places shifted is specified as a parameter for the "Shift word by word with sign" instruction, the following syntax is used:

```
SSI <number of places shifted>
```

If the number of places shifted is specified using the right byte of accumulator 2, the following syntax is used for the instruction "Shift word by word with sign":

```
SSI
```

Parameters

The following table shows the parameters of the "Shift word by word with sign" instruction:

Parameters	Format	Description
<number of places shifted>	Positive integers	Number of bit positions by which the value will be shifted

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand in accumulator 1.
SSI 6                        // Shift bits 0 to 15 of accumulator 1 six positions
                             to the right.
                             // Fill the now empty bit places with the state of bit
                             15.
T "Tag_Result_1"             // Transfer contents of accumulator 1 to the operand.
L 3                          // Load number of places shifted into accumulator 1
L "Tag_Value_2"              // Move number of places shifted to the right byte of
                             accumulator 2.
                             // Load value of the operand in accumulator 1.
SSI                          // Shift bits 0 to 15 of accumulator 1 three positions
                             to the right.
                             // Fill the now empty bit places with the state of bit
                             15.
T "Tag_Result_2"             // Transfer contents of accumulator 1 to the operand.
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	0101	1111	0110	0100	1001	1101	0011	1011
Tag_Result_1	0101	1111	0110	0100	1111	1110	0111	0100
Tag_Value_2	0101	1111	0110	0100	0101	1101	0010	1011
Tag_Result_2	0101	1111	0110	0100	0000	1011	1010	0101

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bits for shift and rotate instructions (Page 2462)

Basic information on the status word (Page 2453)

Overview of the valid data types (Page 2236)

SSD: Shift double word by double word with sign

Description

Use the instruction "Shift double word by double word with sign" to shift the complete contents of accumulator 1 bit by bit to the right. During the shifting, positions that become free are padded with the signal state of bit 31 (sign bit for DINT numbers).

The following options are available for specifying the number of bit positions by which the bits are to be shifted:

- Specification of a positive integer as parameter of the instruction. The specified number must be in a value range from 0 to 31.
- Specification of the value of the right byte in accumulator 2. The byte is interpreted as a positive integer.

The instruction is executed regardless of the RLO. It resets the status bit CC 0 to "0" and the status bit CC 1 to the signal state of the last bit that was shifted out.

If the specified number of places shifted is greater than 31, all bits of accumulator 1 are padded with the signal state of bit 31.

If the specified number of places shifted is equal to zero, the instruction is not executed. The status bits are not changed.

Syntax

If the number of places shifted is specified as a parameter for the "Shift double word by double word with sign" instruction, the following syntax is used:

```
SSD <number of places shifted>
```

If the number of places shifted is specified using the right byte of accumulator 2, the following syntax is used for the "Shift double word by double word with sign" instruction:

```
SSD
```

Parameters

The following table shows the parameters of the instruction "Shift double word by double word with sign":

Parameters	Format	Description
<number of places shifted>	Positive integers	Number of bit positions by which the value will be shifted

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand in accumulator 1.
SSD 7                        // Shift bits 0 to 31 of accumulator 1 seven positions
                             to the right.
                             // Fill the now empty bit places with the state of bit
                             31.
T "Tag_Result_1"             // Transfer contents of accumulator 1 to the operand.
L 4                          // Load number of places shifted into accumulator 1
L "Tag_Value_2"              // Move number of places shifted to the right byte of
                             accumulator 2.
                             // Load value of the operand in accumulator 1.
SSD                          // Shift bits 0 to 31 of accumulator 1 four positions
                             to the right.
                             // Fill the now empty bit places with the state of bit
                             31.
T "Tag_Result_2"             // Transfer contents of accumulator 1 to the operand.
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	1000	1111	0110	0100	0101	1101	0011	1011
Tag_Result_1	1111	1111	0001	1110	1100	1000	1011	1010
Tag_Value_2	0010	1000	1010	0010	1001	1011	1100	1101
Tag_Result_2	0000	0010	1000	1010	0010	1001	1011	1100

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bits for shift and rotate instructions (Page 2462)

Basic information on the status word (Page 2453)

Overview of the valid data types (Page 2236)

SLW: Shift left word by word

Description

Use the "Shift left word by word" instruction to shift the right word (bits 0 to 15) of accumulator 1 bit by bit to the left. The bit positions that become free are padded with zeros. The bits 16 to 31 of accumulator 1 remain unchanged.

The following options are available for specifying the number of bit positions by which the bits are to be shifted:

- Specification of a positive integer as parameter of the instruction. The specified number must be in a value range from 0 to 15.
- Specification of the value of the right byte in accumulator 2. The byte is interpreted as a positive integer.

The instruction is executed regardless of the RLO. It resets the status bit CC 0 to "0" and the status bit CC 1 to the signal state of the last bit that was shifted out.

If the specified number of places shifted is greater than 15, all bits in the right word of accumulator 1 are padded with zeros.

If the specified number of places shifted is equal to zero, the instruction is not executed. The status bits are not changed.

Syntax

If the number of places shifted is specified as a parameter, the following syntax is used for the "Shift left word by word" instruction:

```
SLW <number of places shifted>
```

If the number of places shifted is specified using the right byte of accumulator 2, the following syntax is used for the "Shift left word by word" instruction:

```
SLW
```

Parameters

The following table shows the parameters of the "Shift left word by word" instruction:

Parameters	Format	Description
<number of places shifted>	Positive integers	Number of bit positions by which the value will be shifted

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand in accumulator 1.
SLW 5	// Shift bits 0 to 15 of accumulator 1 five positions to the left.
	// Fill the now empty bit places with zeros.
T "Tag_Result_1"	// Transfer contents of accumulator 1 to the operand.
L 4	// Load number of places shifted into accumulator 1

```
STL                          Explanation
L "Tag_Value_2"              // Move number of places shifted to the right byte of
                             accumulator 2.
                             // Load value of the operand in accumulator 1.
SLW                          // Shift bits 0 to 15 of accumulator 1 four positions
                             to the left.
                             // Fill the now empty bit places with zeros.
T "Tag_Result_2"             // Transfer contents of accumulator 1 to the operand.
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	0101	1111	0110	0100	0101	1101	0011	1011
Tag_Result_1	0101	1111	0110	0100	1010	0111	0110	0000
Tag_Value_2	0101	1111	0110	0100	0101	1101	0010	1011
Tag_Result_2	0101	1111	0110	0100	1101	0010	1011	0000

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bits for shift and rotate instructions (Page 2462)

Basic information on the status word (Page 2453)

Overview of the valid data types (Page 2236)

SRW: Shift right word by word

Description

Use the "Shift right word by word" instruction to shift the right word (bits 0 to 15) of accumulator 1 bit by bit to the right. The bit positions that become free are padded with zeros. The bits 16 to 31 of accumulator 1 remain unchanged.

The following options are available for specifying the number of bit positions by which the bits are to be shifted:

- Specification of a positive integer as parameter of the instruction. The specified number must be in a value range from 0 to 15.
- Specification of the value of the right byte in accumulator 2. The byte is interpreted as a positive integer.

The instruction is executed regardless of the RLO. It resets the status bit CC 0 to "0" and the status bit CC 1 to the signal state of the last bit that was shifted out.

If the specified number of places shifted is greater than 15, all bits in the right word of accumulator 1 are padded with zeros.

If the specified number of places shifted is equal to zero, the instruction is not executed. The status bits are not changed.

Syntax

If the number of places shifted is specified as a parameter, the following syntax is used for the "Shift right word by word" instruction:

```
SRW <number of places shifted>
```

If the number of places shifted is specified using the right word of accumulator 2, the following syntax is used for the "Shift right word by word" instruction:

```
SRW
```

Parameters

The following table shows the parameters of the "Shift right word by word" instruction:

Parameters	Format	Description
<number of places shifted>	Positive integers	Number of bit positions by which the value will be shifted

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand in accumulator 1.
SRW 6                        // Shift bits 0 to 15 of accumulator 1 six positions
                             to the right.
                             // Fill the now empty bit places with zeros.
T "Tag_Result_1"             // Transfer contents of accumulator 1 to the operand.
L 4                          // Load number of places shifted into accumulator 1
L "Tag_Value_2"              // Move number of places shifted to the right byte of
                             accumulator 2.
                             // Load value of the operand in accumulator 1.
SRW                          // Shift bits 0 to 15 of accumulator 1 four positions
                             to the right.
                             // Fill the now empty bit places with zeros.
T "Tag_Result_2"             // Transfer contents of accumulator 1 to the operand.
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	0101	1111	0110	0100	0101	1101	0011	1011
Tag_Result_1	0101	1111	0110	0100	0000	0001	0111	0100
Tag_Value_2	0101	1111	0110	0100	0101	1101	0010	1011
Tag_Result_2	0101	1111	0110	0100	0000	0101	1101	0010

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bits for shift and rotate instructions (Page 2462)

Basic information on the status word (Page 2453)

Overview of the valid data types (Page 2236)

SLD: Shift left double word by double word

Description

Use the "Shift left double word by double word" instruction to shift the complete contents of accumulator 1 bit by bit to the left. The bit positions that become free are padded with zeros.

The following options are available for specifying the number of bit positions by which the bits are to be shifted:

- Specification of a positive integer as parameter of the instruction. The specified number must be in a value range from 0 to 31.
- Specification of the value of the right byte in accumulator 2. The byte is interpreted as a positive integer.

The instruction is executed regardless of the RLO. It resets the status bit CC 0 to "0" and the status bit CC 1 to the signal state of the last bit that was shifted out.

If the specified number of places shifted is greater than 31, all bits of accumulator 1 are padded with zeros.

If the specified number of places shifted is equal to zero, the instruction is not executed. The status bits are not changed.

Syntax

If the number of places shifted is specified as a parameter, the following syntax is used for the "Shift left double word by double word" instruction:

```
SLD <number of places shifted>
```

If the number of places shifted is specified using the right byte of accumulator 2, the following syntax is used for the "Shift left double word by double word" instruction:

```
SLD
```

Parameters

The following table shows the parameters of the "Shift left double word by double word" instruction:

Parameters	Format	Description
<number of places shifted>	Positive integers	Number of bit positions by which the value will be shifted

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand in accumulator 1.
SLD 5                        // Shift bits 0 to 31 of accumulator 1 five positions
                             to the left.
                             // Fill the now empty bit places with zeros.
T "Tag_Result_1"             // Transfer contents of accumulator 1 to the operand.
L 4                          // Load number of places shifted into accumulator 1
L "Tag_Value_2"              // Move number of places shifted to the right byte of
                             accumulator 2.
                             // Load value of the operand in accumulator 1.
SLD                          // Shift bits 0 to 31 of accumulator 1 four positions
                             to the left.
                             // Fill the now empty bit places with zeros.
T "Tag_Result_2"             // Transfer contents of accumulator 1 to the operand.
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	0101	1111	0110	0100	0101	1101	0011	1011
Tag_Result_1	1110	1100	1000	1011	1010	0111	0110	0000
Tag_Value_2	1010	1000	1010	0010	1001	1011	1100	1101
Tag_Result_2	1000	1010	0010	1001	1011	1100	1101	0000

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bits for shift and rotate instructions (Page 2462)

Basic information on the status word (Page 2453)

Overview of the valid data types (Page 2236)

SRD: Shift right double word by double word

Description

Use the instruction "Shift right double word by double word" to shift the complete contents of accumulator 1 bit by bit to the right. The bit positions that become free are padded with zeros.

The following options are available for specifying the number of bit positions by which the bits are to be shifted:

- Specification of a positive integer as parameter of the instruction. The specified number must be in a value range from 0 to 31.
- Specification of the value of the right byte in accumulator 2. The byte is interpreted as a positive integer.

The instruction is executed regardless of the RLO. It resets the status bit CC 0 to "0" and the status bit CC 1 to the signal state of the last bit that was shifted out.

If the specified number of places shifted is greater than 31, all bits of accumulator 1 are padded with zeros.

If the specified number of places shifted is equal to zero, the instruction is not executed. The status bits are not changed.

Syntax

If the number of places shifted is specified as a parameter, the following syntax is used for the "Shift right double word by double word" instruction:

```
SRD <number of places shifted>
```

If the number of places shifted is specified using the right byte of accumulator 2, the following syntax is used for the "Shift right double word by double word" instruction:

```
SRD
```

Parameters

The following table shows the parameters of the "Shift right double word by double word" instruction:

Parameters	Format	Description
<number of places shifted>	Positive integers	Number of bit positions by which the value will be shifted

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand in accumulator 1.
SRD 7                        // Shift bits 0 to 31 of accumulator 1 seven positions
                             to the right.
                             // Fill the now empty bit places with zeros.
T "Tag_Result_1"             // Transfer contents of accumulator 1 to the operand.
L 4                          // Load number of places shifted into accumulator 1
L "Tag_Value_2"              // Move number of places shifted to the right byte of
                             accumulator 2.
                             // Load value of the operand in accumulator 1.
SRD                          // Shift bits 0 to 31 of accumulator 1 four positions
                             to the right.
                             // Fill the now empty bit places with zeros.
T "Tag_Result_2"             // Transfer contents of accumulator 1 to the operand.
```

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	0101	1111	0110	0100	0101	1101	0011	1011
Tag_Result_1	0000	0000	1011	1110	1100	1000	1011	1010
Tag_Value_2	1010	1000	1010	0010	1001	1011	1100	1101
Tag_Result_2	0000	1010	1000	1010	0010	1001	1011	1100

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bits for shift and rotate instructions (Page 2462)

Basic information on the status word (Page 2453)

Overview of the valid data types (Page 2236)

Rotate

RLD: Rotate left double word by double word

Description

Use the "Rotate left double word by double word" instruction to rotate the complete contents of accumulator 1 bit by bit to the left. During processing the instruction shifts bit 0 to 31 of

accumulator 1 bit by bit to the left and pads the positions that become free during writing with the pushed out bit positions.

The following options are available for specifying the number of bit positions by which the bits are to be rotated:

- Specification of a positive integer as parameter of the instruction.
- Specification of the value of the right byte in accumulator 2. The byte is interpreted as a positive integer.

The instruction is executed regardless of the RLO. It resets the status bit CC 0 to "0" and the status bit CC 1 to the signal state of the last bit that was shifted out.

If the specified rotate number is 32, the contents of accumulator 1 remain unchanged. For values greater than 32, the rotation number is calculated with a modulo division by 32. If, for example, the rotate number is 34, the bits of accumulator 1 are rotated by 2 bit positions.

If the specified rotate number is equal to zero, the instruction is not executed. The status bits are not changed.

Syntax

If the number of places rotated is specified as a parameter, the following syntax is used for the "Rotate left double word by double word" instruction:

```
RLD <rotate number>
```

If the number of places rotated is specified using the right byte of accumulator 2, the following syntax is used for the "Rotate left double word by double word" instruction:

```
RLD
```

Parameters

The following table shows the parameters of the "Rotate left double word by double word" instruction:

Parameters	Format	Description
<rotate number>	Positive integers	Number of bit positions to be rotated.

Example

The following example shows how the instruction works:

```
STL                          Explanation
L "Tag_Value_1"              // Load value of the operand in accumulator 1.
RLD 4                        // Rotate bits 0 to 31 of accumulator 1 four positions
                             to the left
T "Tag_Result_1"             // Transfer contents of accumulator 1 to the operand.
```

STL	Explanation
L 6	// Load number of places to be rotated in accumulator 1.
L "Tag_Value_2"	// Move number of places to be rotated to right byte of accumulator 2
	// Load value of the operand in accumulator 1.
RLD	// Rotate bits 0 to 31 of accumulator 1 six positions to the left
T "Tag_Result_2"	// Transfer contents of accumulator 1 to the operand.

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	**0101**	1111	0110	0100	0101	1101	0011	1011
Tag_Result_1	1111	0110	0100	0101	1101	0011	1011	**0101**
Tag_Value_2	**1010**	**10**00	1010	0010	1001	1011	1100	1101
Tag_Result_2	0010	1000	1010	0110	1111	0011	01**10**	**1010**

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bits for shift and rotate instructions (Page 2462)

Basic information on the status word (Page 2453)

Overview of the valid data types (Page 2236)

RRD: Rotate right double word by double word

Description

Use the "Rotate right double word by double word" instruction to rotate the complete contents of accumulator 1 bit by bit to the right. During processing the instruction shifts bits 0 to 31 of accumulator 1 bit by bit to the right and pads the positions that become free during writing with the pushed out bit positions.

The following options are available for specifying the number of bit positions by which the bits are to be rotated:

- Specification of a positive integer as parameter of the instruction.
- Specification of the value of the right byte in accumulator 2. The byte is interpreted as a positive integer.

The instruction is executed regardless of the RLO. It resets the status bit CC 0 to "0" and the status bit CC 1 to the signal state of the last bit that was shifted out.

If the specified rotate number is 32, the contents of accumulator 1 remain unchanged. For values greater than 32, the rotation number is calculated with a modulo division by 32. If, for example, the rotate number is 34, the bits of accumulator 1 are rotated by 2 bit positions.

If the specified rotate number is equal to zero, the instruction is not executed. The status bits are not changed.

Syntax

If the number of places rotated is specified as a parameter, the following syntax is used for the "Rotate right double word by double word" instruction:

```
RRD <rotate number>
```

If the number of places rotated is specified using the right byte of accumulator 2, the following syntax is used for the "Rotate right double word by double word" instruction:

```
RRD
```

Parameters

The following table shows the parameters of the instruction "Rotate right double word by double word":

Parameters	Format	Description
<rotate number>	Positive integers	Number of bit positions to be rotated.

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand in accumulator 1.
RRD 4	// Rotate bits 0 to 31 of accumulator 1 four places to the right.
T "Tag_Result_1"	// Transfer contents of accumulator 1 to the operand.
L 6	// Load number of places to be rotated in accumulator 1.
L "Tag_Value_2"	// Move number of places to be rotated to right byte of accumulator 2
	// Load value of the operand in accumulator 1.
RRD	// Rotate bits 0 to 31 of accumulator 1 six places to the right.
T "Tag_Result_2"	// Transfer contents of accumulator 1 to the operand.

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_Value_1	0101	1111	0110	0100	0101	1101	0011	**1011**
Tag_Result_1	**1011**	0101	1111	0110	0100	0101	1101	0011
Tag_Value_2	1010	1000	1010	0010	1001	1011	11**00**	**1101**
Tag_Result_2	**0011**	**01**10	1010	0010	1000	1010	0110	1111

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bits for shift and rotate instructions (Page 2462)

Basic information on the status word (Page 2453)

Overview of the valid data types (Page 2236)

RLDA: Rotate left by status bit CC 1

Description

Use the "Rotate left by status bit CC 1" instruction to shift the contents of accumulator 1 one bit to the left. The bit positions (0) that become free during shifting are padded with the signal state of status bit CC 1. Status bit CC 1 receives the signal state of the shifted out bit (31).

The instruction is executed regardless of the RLO. It resets the status bit CC 0 to "0".

Syntax

The following syntax is used for the "Rotate left by status bit CC 1" instruction:

```
RLDA
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value"	// Load value of the operand in accumulator 1.
RLDA	// Rotate bits 0 to 31 of accumulator 1 one place to the left.
	// Fill bit 1 with the signal state of status bit CC 1.
	// Write the shifted-out bit to the status bit CC 1.
T "Tag_Result"	// Transfer contents of accumulator 1 to the operand.

The following table shows how the instruction works using specific operand values:

Operand	CC 1	Value							
Tag_Value	1	0101	1111	0110	0100	0101	1101	0011	1011
Tag_Result	0	1011	1110	1100	1000	1011	1010	0111	0111

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bits for shift and rotate instructions (Page 2462)

Basic information on the status word (Page 2453)

RRDA: Rotate right by status bit CC 1

Description

Use the "Rotate right by status bit CC 1" instruction to shift the contents of accumulator 1 one bit to the right. The bit positions (31) that become free during shifting are padded with the signal state of status bit CC 1. Status bit CC 1 receives the signal state of the shifted out bit (0).

The instruction is executed regardless of the RLO. It resets the status bit CC 0 to "0".

Syntax

Use the following syntax for the "Rotate right by status bit CC 1" instruction:

```
RRDA
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value"	// Load value of the operand in accumulator 1.
RRDA	// Rotate bits 0 to 31 of accumulator 1 one place to the right
	// Fill bit 31 with the signal state of status bit CC 1.
	// Write the shifted-out bit to the status bit CC 1.
T "Tag_Result"	// Transfer contents of accumulator 1 to the operand.

The following table shows how the instruction works using specific operand values:

Operand	CC 1	Value							
Tag_Value	**1**	**0**101	1111	0110	0100	0101	1101	0011	1011
Tag_Result	**0**	**1**010	1111	1011	0010	0010	1110	1001	1101

Status bit CC 1 has the value "1" because the bit 0 of the "Tag_Value" operand also has the value "1".

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting the status bits for shift and rotate instructions (Page 2462)

Basic information on the status word (Page 2453)

Additional instructions

Accumulators

TAK: Swap content of accumulators 1 and 2

Description

Use the instruction "Swap content of accumulators 1 and 2" to switch the contents accumulator 1 with the contents of accumulator 2. On controllers with four accumulators, the contents of accumulators 3 and 4 are not affected by the instruction.

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

Syntax

Use the following syntax for the "Swap content of accumulators 1 and 2" instruction:

```
TAK
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.

STL	Explanation
	// Load value of the operand "Tag_Value_2" in accumulator 1
>I	// Compare whether the value of accumulator 2 is greater than the value of accumulator 1.
JC NEXT	// If condition is satisfied (RLO = "1"), jump to jump label NEXT.
	// If the condition is not satisfied(RLO = "0"), execute the next instruction.
TAK	// Swap the contents of accumulators 1 and 2
NEXT: -I	// Jump label "NEXT"
	// Subtract value of accumulator 1 from value of accumulator 2.
T "Tag_Output"	// Transfer contents of accumulator 1 to the operand "Tag_Output".

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

PUSH: Shift content to the next higher accumulator

Description

Use the "Shift content to the next higher accumulator" instruction to move the contents of the accumulators 1 to 3 to the next higher accumulator.

In case of controllers with 2 accumulators, only the contents of accumulator 1 are copied to accumulator 2.

In controls with 4 accumulators, the content of accumulator 1 is moved to accumulator 2; the content of accumulator 2 is moved to accumulator 3 and the content of accumulator 3 to accumulator 4. The content of accumulator 4 will be lost.

The content of accumulator 1 is not influenced by the instruction and remains unchanged after processing the instruction.

The following table shows the contents of accumulators 1 to 4 before and after the processing of the instruction:

Status	Accumulator			
	1	2	3	4
Before processing:	Value A	Value B	Value C	Value D
After processing	Value A	Value A	Value B	Value C

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

Syntax

Use the following syntax for the instruction "Shift content to the next higher accumulator":

```
PUSH
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand in accumulator 1.
PUSH	// Shift contents of accumulators 1 to 3 to the next higher accumulator.
L "Tag_Value_2"	// Load value of the operand in accumulator 1.

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

POP: Shift content to the next lower accumulator

Description

Use the "Shift content to the next lower accumulator" instruction to move the contents of the accumulators 4 to 2 to the next lower accumulator.

In controls with 2 accumulators the content of accumulator 2 will be moved to accumulator 1.

In controls with 4 accumulators, the content of accumulator 4 is moved to accumulator 3; the content of accumulator 3 is moved to accumulator 2 and the content of accumulator 2 to accumulator 1. The content of accumulator 1 will be lost.

The content of accumulator 4 is not influenced by the instruction and remains unchanged after processing the instruction.

The following table shows the contents of accumulators 1 to 4 before and after the processing of the instruction:

Status	Accumulator			
	1	2	3	4
Before processing:	Value A	Value B	Value C	Value D
After processing	Value B	Value C	Value D	Value D

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

Syntax

Use the following syntax for the instruction "Shift content to the next lower accumulator":

```
POP
```

Example

The following example shows how the instruction works:

STL	Explanation
`T "Tag_Value_1"`	`// Transfer contents of accumulator 1 to the operand "Tag_Value_1".`
`POP`	`// Shift content of accumulators 4 to 2 to the next lower accumulator`
`T "Tag_Value_2"`	`// Transfer contents of accumulator 1 to the operand "Tag_Value_2".`

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

ENT: Shift accumulators 2 and 3 to the next higher accumulator

Description

Use the instruction "Shift accumulators 2 and 3 to the next higher accumulator" to shift the contents of accumulator 3 to accumulator 4 and the contents of accumulator 2 to accumulator 3. The contents of accumulators 1 and 2 remain unchanged.

The following table shows the contents of accumulators 1 to 4 before and after the processing of the instruction:

Status	Accumulator			
	1	2	3	4
Before processing:	Value A	Value B	Value C	Value D
After processing	Value A	Value B	Value B	Value C

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

Syntax

Use the following syntax for the "Shift accumulators 2 and 3 to the next higher accumulator" instruction:

```
ENT
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
*R	// Multiply the values of accumulator 1 and 2.
	// Store product (intermediate result) in accumulator 1.
L "Tag_Value_3"	// Shift the content of accumulator 1 (intermediate result) to accumulator 2.
	// Load value of the operand "Tag_Value_3" in accumulator 1
ENT	// Copy the content of accumulator 2 (intermediate result) to accumulator 3.
	// Copy the content of accumulator 3 to accumulator 4.
L "Tag_Value_4"	// Move the contents of accumulator 1 ("Tag_Value_3") to accumulator 2.
	// Load value of the operand "Tag_Value_4" in accumulator 1
+R	// Add the values of accumulator 1 and 2.
	// Store sum in accumulator 1.
	// Transfer the value of accumulator 3 (intermediate result) to accumulator 2.
LEAVE	// Shift the content of accumulator 3 (intermediate result) to accumulator 2.
	// Shift the contents of accumulator 4 to accumulator 3.
/R	// Divide the values of accumulator 1 and 2.
	// Store result in accumulator 1.
T "Tag_Result"	// Transfer result (contents of accumulator 1) to the operand "Tag_Result".

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

LEAVE: Shift accumulators 3 and 4 to the next lower accumulator

Description

Use the instruction "Shift accumulators 3 and 4 to the next lower accumulator" to shift the contents of accumulator 3 to accumulator 2 and the contents of accumulator 4 to accumulator 3. The contents of accumulators 1 and 4 remain unchanged.

Instructions for arithmetic calculations include the functionality of the "Shift accumulators 3 and 4 to the next lower accumulator" instruction.

The following table shows the contents of accumulators 1 to 4 before and after the processing of the instruction:

Status	Accumulator			
	1	2	3	4
Before processing:	Value A	Value B	Value C	Value D
After processing	Value A	Value C	Value D	Value D

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

Syntax

Use the following syntax for the "Shift accumulators 3 and 4 to the next lower accumulator" instruction:

```
LEAVE
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
AD	// AND the value of accumulator 2 Tag_Value_1 with the value of accumulator 1 ("Tag_Value_2")
	// Store result in accumulator 1.

STL	Explanation
L "Tag_Value_3"	// Shift the contents of accumulator 1 (intermediate result) to accumulator 2.
	// Load value of the operand "Tag_Value_3" in accumulator 1
ENT	// Copy the content of accumulator 2 (intermediate result) to accumulator 3.
	// Copy the content of accumulator 3 to accumulator 4.
L "Tag_Value_4"	// Move the contents of accumulator 1 ("Tag_Value_3") to accumulator 2.
	// Load value of the operand "Tag_Value_4" in accumulator 1
AD	// AND the value of accumulator 2 Tag_Value_3 with the value of accumulator 1 ("Tag_Value_4")
	// Store result in accumulator 1.
LEAVE	// Shift the contents of accumulator 3 (intermediate result) to accumulator 2.
	// Shift the contents of accumulator 4 to accumulator 3.
OD	// OR the value of accumulator 2 with the value of accumulator 1
	// Store result in accumulator 1.
T "Tag_Result"	// Transfer the result to operand "Tag_Result".

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Address register

+AR1: Add accumulator 1 to AR1

Description

Use the "Add accumulator 1 to AR1" instruction to add a value to the content of address register 1. The type of pointer in address register 1 and the operand range are retained.

The following options are available for specifying the value to be added:

- Specification via a constant: The instruction adds the value of the constant to address register 1. The value of the constant must correspond to the format of the range-internal pointer (POINTER).
- Specification via the value of the right word in the accumulator 1: The instruction interprets the value of the right word in accumulator 1 as a 16-bit integer and extends it with the correct sign to 24 bits. The value of accumulator 1 then adds the instruction to address register 1. Values in the range from -32 768 to +32 767 are permitted.

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Syntax

If the addition value is specified using a constant, use the following syntax for the "Add accumulator 1 to AR1" instruction:

```
+AR1 <constant>
```

If the addition value is specified using the right word of accumulator 1, use the following syntax for the "Add accumulator 1 to AR1" instruction:

```
+AR1
```

Parameters

The following table shows the parameters of the "Add accumulator 1 to AR1" instruction:

Parameters	Data type	Description
<constant>	POINTER	Value that is added to address register 1.

Example

The following example shows how the instruction works:

STL	Explanation
+AR1 P#10.0	// Add pointer to address register 1.
L "Tag_Value"	// Load value of the operand "Tag_Value" into the right word of accumulator 1.
+AR1	// Add value of accumulator 1 to address register 1.
TAR1 %MD24	// Transfer contents of address register 1 to double word MD24.

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

+AR2: Add accumulator 1 to AR2

Description

Use the "Add accumulator 1 to AR2" instruction to add a value to the content of address register 2. The type of pointer in address register 2 and the operand range are retained.

The following options are available for specifying the value to be added:

- Specification via a constant: The instruction adds the value of the constant to address register 2. The value of the constant must correspond to the format of the range-internal pointer (POINTER).
- Specification via the value of the right word in the accumulator 1: The instruction interprets the value of the right word in accumulator 1 as a 16-bit integer and extends it with the correct sign to 24 bits. The value of accumulator 1 then adds the instruction to address register 2. Values in the range from -32 768 to +32 767 are permitted.

The CPU executes the instruction regardless of the result of logic operation and the status bit. The instruction does not affect either the result of logic operation or the status bit.

The contents of the accumulators are not changed by the instruction.

Note

Address register AR2 is used when processing multiple instances. If you program the "Add accumulator 1 to AR2" instruction, you must first "save" the contents of address register 2 and reload it later.

Syntax

If the addition value is specified using a constant, use the following syntax for the "Add accumulator 1 to AR2" instruction:

```
+AR2 <constant>
```

If the addition value is specified using the right word of accumulator 1, use the following syntax for the "Add accumulator 1 to AR2" instruction:

```
+AR2
```

Parameters

The following table shows the parameters of the "Add accumulator 1 to AR2" instruction:

Parameters	Data type	Description
<constant>	POINTER	Value that is added to address register 1.

Example

The following example shows how the instruction works:

STL	Explanation
+AR2 P#10.0	// Add pointer to address register 2.
L "Tag_Value"	// Load value of the operand "Tag_Value" into the right word of accumulator 1.
+AR2	// Add value of accumulator 1 to address register 2.
TAR2 %MD24	// Transfer contents of address register 1 to double word MD24.

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Master Control Relay

MCR(: Open MCR ranges

Description

Use the "Open MCR ranges" instruction to switch the MCR dependency of an MCR range on and off depending on the result of logic operation (RLO).

Use the instructions "Enable MCR range" (MCRA) and "Disable MCR range" (MCRD) to specify the start and end of an MCR range. Within the defined MCR range you can switch the MCR dependency on and off for one or more program sections.

Note

Read the information in the section "Important notes on using the MCR functionality" in "See also".

The MCR dependency is switched on when the RLO is "1" before processing of the instruction. If the RLO is "0" before the instruction, the MCR dependency is switched off. The end of the MCR dependent program section is programmed using the instruction "Close MCR range". The program section between the two instructions is affected by the MCR. Only the MCR-dependent instructions (for example, the "Transfer" instruction (T)) are affected.

You can also nest the switching on and off of MCR dependency by programming the "Open MCR ranges" instruction in an MCR-dependent program section. You can nest the instruction up to 8 levels deep. The MCR dependency of the higher-level program section hereby controls the MCR dependencies in all lower-level program sections. If the MCR dependency is switched off in the higher-level program section, this MCR dependency cannot be switched on in any of the lower-level MCR dependent program sections.

If you call a block within an MCR range, the MCR dependency is disabled in the called block. When the block is exited, the MCR dependency is set so that it is the same as before the block call.

The instructions "Open MCR ranges" and "Close MCR ranges" must always be programmed in pairs. If one of the two instructions is not programmed, an MCR stack error is triggered.

Syntax

The "Open MCR ranges" instruction is programmed according to the following general outline:

```
MCRA
MCR(
....
)MCR
MCRD
```

Example

The following example shows how the instruction works:

STL	Explanation
MCRA	// Start of the MCR range
A "Tag_Input_1"	// Query operand for "1" and link with the current RLO according to AND.
MCR(	// Start of the MCR dependency
	// If signal state of RLO is "1", switch on MCR dependency.
	// If signal state of RLO is "0", switch off MCR dependency.
A "Tag_Input_2"	// Query operand for "1" and link with the current RLO according to AND.
= "Tag_Output_1"	// If MCR dependency is switched off, reset operand to "0".
	// If MCR dependency is switched on, set or reset operand depending on RLO
A "Tag_Input_3"	// Query operand for "1" and link with the current RLO according to AND.
MCR(	// Nesting of MCR dependency
A "Tag_Input_4"	// Query operand for "1" and link with the current RLO according to AND.
A "Tag_Input_5"	// Query operand for "1" and link with the current RLO according to AND.
= "Tag_Output_2"	// If MCR dependency is switched off, reset operand to "0".
	// If MCR dependency is switched on, set or reset operand depending on RLO

STL	Explanation
)MCR	// End of the nesting
L "Tag_InputValue_1"	// Load value of the operand in accumulator 1.
T "Tag_OutputValue_1"	// If MCR dependency is switched off, write the value "0" to the operand.
	// If MCR dependency is switched on, transfer the contents of accumulator 1 to the operand.
)MCR	// End of the MCR dependency
A "Tag_Input_6"	// Query operand for "1" and link with the current RLO according to AND.
MCR(	// Start of the MCR dependency
	// If signal state of RLO is "1", switch on MCR dependency.
	// If signal state of RLO is "0", switch off MCR dependency.
L "Tag_InputValue_2"	// Load value of the operand in accumulator 1.
T "Tag_OutputValue_2"	// If MCR dependency is switched off, write the value "0" to the operand.
	// If MCR dependency is switched on, transfer the contents of accumulator 1 to the operand.
)MCR	// End of the MCR dependency
MCRD	// End of the MCR range
A "Tag_Input_7"	// Query operand for "1" and link with the current RLO according to AND.
= "Tag_Output_3"	// Set or reset operand depending on RLO

See also

Important notes on using MCR functions (Page 2472)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits in program control operation instructions (Page 2460)

Basic information on the status word (Page 2453)

)MCR: Close MCR ranges

Description

The "Close MCR ranges" instruction completes an MCR dependent program section. In this case the instruction deletes the RLO from the last entry in the MCR stack.

Note

Read the information in the section "Important notes on using the MCR functionality" in "See also".

Use the instruction "Open MCR ranges" to program the MCR dependency. The instructions "Open MCR ranges" and "Close MCR ranges" must always be programmed in pairs. If one of the two instructions is not programmed, an MCR stack error is triggered.

Syntax

The instruction "Close MCR ranges" is programmed according to the following general outline:

```
MCRA          // Start of the MCR range
MCR(          // Start of the MCR dependency
.....         // Any program
)MCR          // End of the MCR dependency
MCRD          // End of the MCR range
```

Example

The following example shows how the instruction works:

STL	Explanation
MCRA	// Start of the MCR range
A "Tag_Input_1"	// Query operand for "1" and link with the current RLO according to AND.
MCR(	// Start of the MCR dependency
	// If signal state of RLO is "1", switch on MCR dependency.
	// If signal state of RLO is "0", switch off MCR dependency.
A "Tag_Input_2"	// Query operand for "1" and link with the current RLO according to AND.
= "Tag_Output_1"	// If MCR dependency is switched off, reset operand to "0".
	// If MCR dependency is switched on, set or reset operand depending on RLO
L "Tag_InputValue"	// Load value of the operand in accumulator 1.
T "Tag_OutputValue"	// If MCR dependency is switched off, write the value "0" to the operand.
	// If MCR dependency is switched on, transfer the contents of accumulator 1 to the operand.
)MCR	// End of the MCR dependency
MCRD	// End of the MCR range
A "Tag_Input_3"	// Query operand for "1" and link with the current RLO according to AND.
= "Tag_Output_2"	// Set or reset operand depending on RLO

See also

Important notes on using MCR functions (Page 2472)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits in program control operation instructions (Page 2460)

Basic information on the status word (Page 2453)

MCRA: Enable MCR range

Description

Use the instruction "Enable MCR range" to specify the start of an MCR range. Within this range you can switch MCR dependency on and off using the following instructions:

- MCR(: Open MCR ranges
-)MCR: Close MCR ranges

You specify the end of the MCR range with the instruction "Disable MCR range" (MCRD). The instructions for defining the start and the end of an MCR range must always be used in pairs.

Note

Read the information in the section "Important notes on using the MCR functionality" in "See also".

If you call a block within an MCR range, the MCR dependency is disabled in the called block. When a block is exited, the MCR dependency is set so that it is the same as before the block call.

The instruction is executed regardless of the status bits and without affecting these bits.

Syntax

Program the instruction "Enable MCR range" according to the following general outline:

```
MCRA          // Start of the MCR range
MCR(          // Start of the MCR dependency
.....         // Any program
)MCR          // End of the MCR dependency
MCRD          // End of the MCR range
```

Example

The following example shows how the instruction works:

STL	Explanation
MCRA	// Start of the MCR range
A "Tag_Input_1"	// Query operand for "1" and link with the current RLO according to AND.
MCR(	// Start of the MCR dependency
	// If signal state of RLO is "1", switch on MCR dependency.
	// If signal state of RLO is "0", switch off MCR dependency.
A "Tag_Input_2"	// Query operand for "1" and link with the current RLO according to AND.
= "Tag_Output_1"	// If MCR dependency is switched off, reset operand to "0".
	// If MCR dependency is switched on, set or reset operand depending on RLO
L "Tag_InputValue"	// Load value of the operand in accumulator 1.
T "Tag_OutputValue"	// If MCR dependency is switched off, write the value "0" to the operand.
	// If MCR dependency is switched on, transfer the contents of accumulator 1 to the operand.
)MCR	// End of the MCR dependency
MCRD	// End of the MCR range
A "Tag_Input_3"	// Query operand for "1" and link with the current RLO according to AND.
= "Tag_Output_2"	// Set or reset operand depending on RLO

See also

Important notes on using MCR functions (Page 2472)
Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)
Setting status bits in program control operation instructions (Page 2460)
Basic information on the status word (Page 2453)

MCRD: Disable MCR range

Description

Use the instruction "Disable MCR range" to specify the end of an MCR range.

You specify the start of the MCR range with the instruction "Enable MCR range" (MCRA). The instructions for defining the start and the end of an MCR range must always be used in pairs.

Note

Read the information in the section "Important notes on using the MCR functionality" in "See also".

Within an MCR range you can program the MCR dependency with the help of the following instructions:

- MCR(: Open MCR ranges
-)MCR: Close MCR ranges

If you call a block within an MCR range, the MCR dependency is disabled in the called block. When a block is exited, the MCR dependency is set so that it is the same as before the block call.

The instruction is executed regardless of the status bits and without affecting these bits.

Syntax

The instruction "Disable MCR range" is programmed according to the following general outline:

```
MCRA          // Start of the MCR range
MCR(          // Start of the MCR dependency
.....         // Any program
)MCR          // End of the MCR dependency
MCRD          // End of the MCR range
```

Example

The following example shows how the instruction works:

STL	Explanation
MCRA	// Start of the MCR range
A "Tag_Input_1"	// Query operand for "1" and link with the current RLO according to AND.
MCR(	// Start of the MCR dependency
	// If signal state of RLO is "1", switch on MCR dependency.
	// If signal state of RLO is "0", switch off MCR dependency.
A "Tag_Input_2"	// Query operand for "1" and link with the current RLO according to AND.
= "Tag_Output_1"	// If MCR dependency is switched off, reset operand to "0".
	// If MCR dependency is switched on, set or reset operand depending on RLO

STL	Explanation
L "Tag_InputValue"	// Load value of the operand in accumulator 1.
T "Tag_OutputValue"	// If MCR dependency is switched off, write the value "0" to the operand.
	// If MCR dependency is switched on, transfer the contents of accumulator 1 to the operand.
)MCR	// End of the MCR dependency
MCRD	// End of the MCR range
A "Tag_Input_3"	// Query operand for "1" and link with the current RLO according to AND.
= "Tag_Output_2".	// Set or reset operand depending on RLO.

See also

Important notes on using MCR functions (Page 2472)

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

Setting status bits in program control operation instructions (Page 2460)

Basic information on the status word (Page 2453)

Null instructions

BLD: Program display (null instruction)

Description

The instruction "Program display (null instruction)" does not execute any function and does not affect the status bits. The instruction is used to recognize code sequences during a parameter transfer or with LAD/FBD networks. It is created automatically when a LAD or FBD program is displayed in STL. The parameter value is the number of the instruction and is generated by the programming device.

Syntax

Use the following syntax for the "Program display (null instruction)" instruction:

```
BLD <number>
```

Parameters

The following table shows the parameters of the "Program display (null instruction)" instruction:

Parameters	Data type	Description
<number>	WORD	Number of the instruction

See also

Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)

NOP 0: Null instruction

Description

If "Null instruction" has a parameter 0, it does not execute any function and does not affect the status bits. The instruction code contains a bit pattern with 16 zeros. The instruction is only relevant for the programming device when a program is displayed.

Syntax

Use the following syntax for "Null instruction" with a parameter 0:

```
NOP 0
```

Example

The following example shows how the instruction works:

STL	Explanation
L "Tag_Value_1"	// Load value of the operand "Tag_Value_1" in accumulator 1.
L "Tag_Value_2"	// Shift the contents of accumulator 1 to accumulator 2.
	// Load value of the operand "Tag_Value_2" in accumulator 1
>I	// Compare whether the value of accumulator 2 is greater than the value of accumulator 1.
A	// Query the signal state of the operand for "1" and AND with the current RLO.
JC NEXT	// If condition is satisfied (RLO = "1"), jump to jump label NEXT.
	// If the condition is not satisfied(RLO = "0"), execute the next instruction.
TAK	// Swap the contents of accumulators 1 and 2
NEXT: NOP 0	// Jump label "NEXT"

See also

Inserting STL instructions (Page 2663)
Editing STL instructions (Page 2677)

NOP 1: Null instruction

Description

If "Null instruction" has a parameter 1, it does not execute any function and does not affect the status bits. The instruction code contains a bit pattern with 16 ones. The instruction is only relevant for the programming device when a program is displayed.

Syntax

Use the following syntax for "Null instruction" with a parameter 1:

```
NOP 1
```

See also

Inserting STL instructions (Page 2663)

Editing STL instructions (Page 2677)

SCL

Timer operations

IEC Timers

TP: Generate pulse

Description

You can use the "Generate pulse" instruction to set the Q parameter for the duration PT. The instruction starts when the result of logic operation (RLO) of the IN parameter changes from "0" to "1" (positive signal edge). The time programmed for PT begins when the instruction starts. The Q parameter is set for the period of time PT, regardless of the subsequent changes in the signal of the IN parameter. Even when a new positive signal edge is detected, the signal state of the Q parameter is not affected as long as the time duration PT is active.

You can scan the current time value set in the ET parameter. The time value starts at T#0s and ends when the value of the time duration PT is reached. When the time duration PT is reached and the signal state at the parameter IN is "0", the parameter ET is reset.

Note

If the timer is not called in the program because it is skipped, for example, output ET returns a constant value as soon as the timer has expired.

Each call of the "Generate pulse" instruction must be assigned to an IEC timer in which the instruction data is stored. An IEC timer can be declared as follows:

- Declaration of a data block of the type TP (for example, "TP_DB")
- Declaration as a local tag of the TP type in the "Static" section of a block (for example, #MyTP_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when an instruction is called and also at each access to the Q or ET outputs.

The operating system resets the instances of the "Generate pulse" instruction during a cold restart. If you want instances of the instruction to be in the initialized state after a warm restart, call these in a startup OB with the value "0" set for the PT parameter. If instances of the instruction "Generate pulse" are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Syntax

Use the following syntax for the instruction "Generate pulse":

- Data block of system data type IEC_Timer (global DB):

SCL

```
<IEC_Timer_DB> TP(IN  := <Operand>,
                  PT := <Operand>,
                  Q => <Operand>,
                  ET => <Operand>)
```

- Local tag:

SCL

```
#myLocal_timer(IN := <Operand>,
               PT := <Operand>,
               Q => <Operand>,
               ET => <Operand>)
```

The syntax of the instruction consists of the following parts:

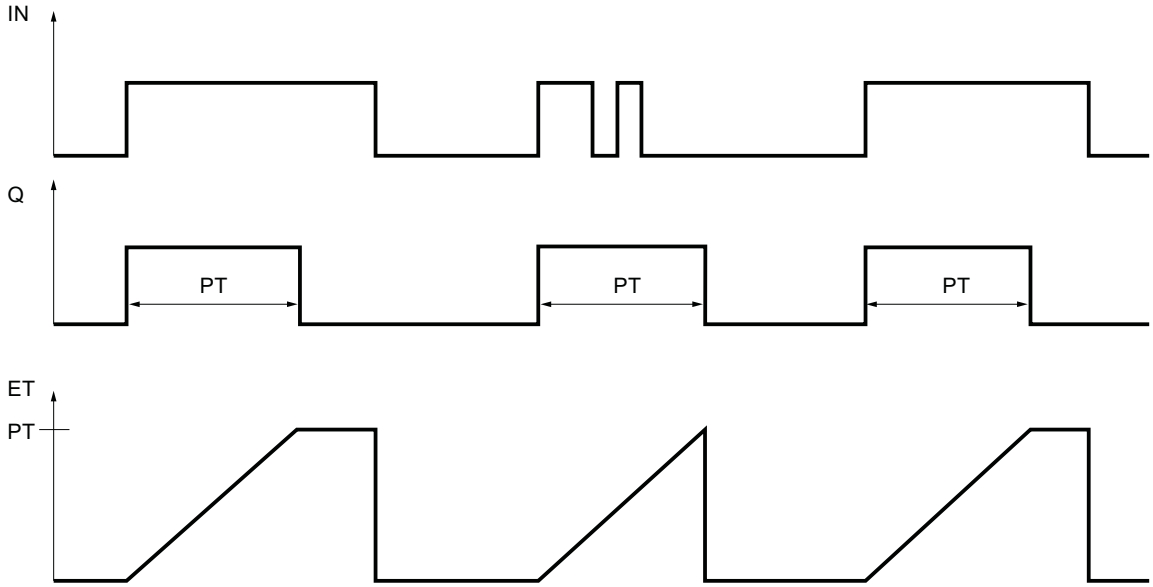
Parameter	Declaration	Data type	Description
IN	Input	BOOL	Start input
PT	Input	TIME	Duration of the pulse. The value of the PT parameter must be positive.

Parameter	Declaration	Data type	Description
Q	Output	BOOL	Operand that is set for the PT duration.
ET	Output	TIME	Current time value

For additional information on valid data types, refer to "See also".

Pulse diagram

The following figure shows the pulse diagram of the "Generate pulse" instruction:



Example

The following example shows how the instruction works:

```
SCL
"TP_DB".TP(IN := "Tag_Start",
           PT := "Tag_PresetTime",
           Q => "Tag_Status",
           ET => "Tag_ElapsedTime");
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the time programmed for the PT parameter is started and the "Tag_Status" operand is set to "1". The current time value is stored in the "Tag_ElapsedTime" operand.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Call function blocks as single or multi-instances (Page 2182)

TON: Generate on-delay

Description

You can use the "Generate on-delay" instruction to delay the setting of the Q parameter for the programmed duration PT. The instruction starts when the result of logic operation (RLO) of the IN parameter changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. When the PT duration has expired, the Q parameter returns signal state "1". The parameter Q remains set as long as the start input is still "1". If the signal state of the IN parameter changes from "1" to "0", the parameter Q will be reset. The timer function is started again when a new rising edge is detected at the parameter IN.

You can scan the current time value set in the ET parameter. The time value starts at T#0s and ends when the value of the time duration PT is reached. The ET parameter is reset as soon as the signal state of the parameter IN changes to "0".

Note

If the timer is not called in the program because it is skipped, for example, output ET returns a constant value as soon as the timer has expired.

Each call of the "Generate on-delay" instruction must be assigned to an IEC timer in which the instruction data is stored. An IEC timer can be declared as follows:

- Declaration of a data block of the type TON (for example, "TON_DB")
- Declaration as a local tag of the TON type in the "Static" section of a block (for example, #MyTON_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when an instruction is called and also at each access to the Q or ET outputs.

The operating system resets the instances of the "Generate on-delay" instruction during a cold restart. If you want instances of the instruction to be in the initialized state after a warm restart, call these in a startup OB with the value "0" set for the PT parameter. If instances of the "Generate on-delay" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Syntax

Use the following syntax for the "Generate on-delay" instruction:

- Data block of system data type IEC_Timer (global DB):

```
SCL
<IEC_Timer_DB> TON(IN  := <Operand>,
                   PT := <Operand>,
                   Q => <Operand>,
                   ET => <Operand>)
```

- Local tag:

```
SCL
#myLocal_timer(IN := <Operand>,
               PT := <Operand>,
               Q => <Operand>,
               ET => <Operand>)
```

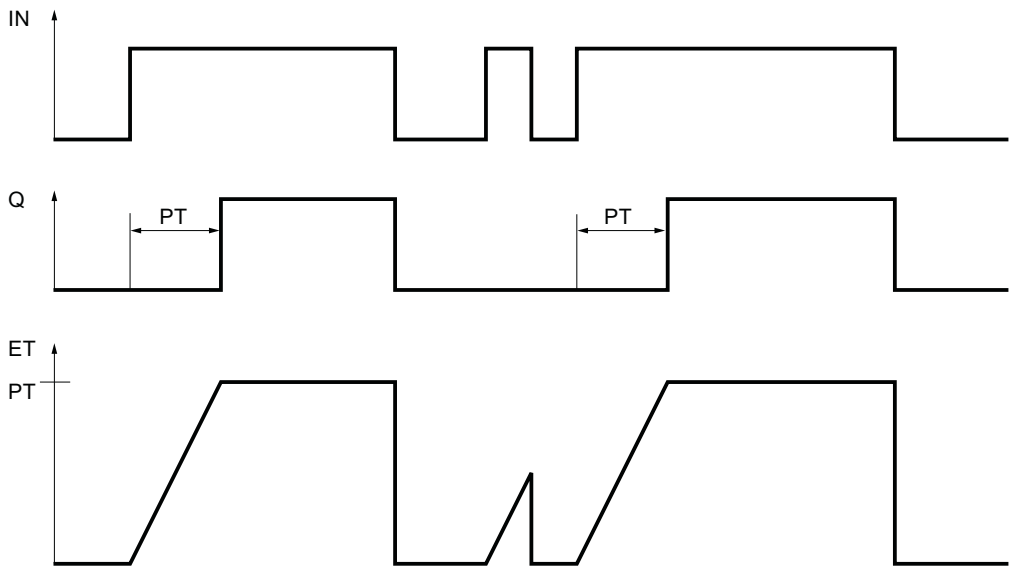
The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
IN	Input	BOOL	Start input
PT	Input	TIME	Duration of the on delay. The value of the PT parameter must be positive.
Q	Output	BOOL	Operand that is set when the timer PT expires.
ET	Output	TIME	Current time value

For additional information on valid data types, refer to "See also".

Pulse diagram

The following figure shows the pulse diagram of the "Generate on-delay" instruction:



Example

The following example shows how the instruction works:

```
SCL
"TON_DB".TON(IN := "Tag_Start",
             PT := "Tag_PresetTime",
             Q => "Tag_Status",
             ET => "Tag_ElapsedTime");
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the time programmed for the PT parameter is started. At the end of the time, the "Tag_Start" operand is set to signal state "1" if the "Tag_Status" operand has signal state "1". At the end of the time the "Tag_Status" operand is set to signal state "1" if the "Tag_Start" operand has signal state "1". The current time value is stored in the "Tag_ElapsedTime" operand.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Call function blocks as single or multi-instances (Page 2182)

TOF: Generate off-delay

Description

You can use the "Generate off-delay" instruction to delay the resetting of the Q parameter for the programmed duration PT. The Q parameter is set when the result of logic operation (RLO) of the parameter IN changes from "0" to "1" (positive signal edge). When the signal state of the IN parameter changes back to "0", the programmed time PT starts. The parameter Q remains set as long the time duration PT is running. When the time duration PT expires, the Q parameter is reset. If the signal state of the IN parameter changes to "1" before the time duration PT has expired, the timer is reset. The signal state of the Q parameter remains set to "1".

You can scan the current time value set in the ET parameter. The current time value starts at T#0s and ends when the value of the time duration PT is reached. When the time duration PT expires, the ET parameter remains set to the current value until the IN parameter changes back to "1". If the IN parameter changes to "1" before the time duration PT has expired, the ET parameter is reset to the value T#0s.

Note

If the timer is not called in the program because it is skipped, for example, output ET returns a constant value as soon as the timer has expired.

Each call of the "Generate off-delay" instruction must be assigned to an IEC timer in which the instruction data is stored. An IEC timer can be declared as follows:

- Declaration of a data block of the type TOF (for example, "TOF_DB")
- Declaration as a local tag of the TOF type in the "Static" section of a block (for example, #MyTOF_TIMER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC timer is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when an instruction is called and also at each access to the Q or ET outputs.

The operating system resets the instances of the "Generate off-delay" instruction during a cold restart. If you want instances of the instruction to be in the initialized state after a warm restart, call these in a startup OB with the value "0" set for the PT parameter. If instances of the

"Generate off-delay" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

Syntax

Use the following syntax for the "Generate off-delay" instruction:

- Data block of system data type IEC_Timer (global DB):

```
SCL
<IEC_Timer_DB> TOF(IN  := <Operand>,
                   PT := <Operand>,
                   Q => <Operand>,
                   ET => <Operand>)
```

- Local tag:

```
SCL
#myLocal_timer(IN := <Operand>,
               PT := <Operand>,
               Q => <Operand>,
               ET => <Operand>)
```

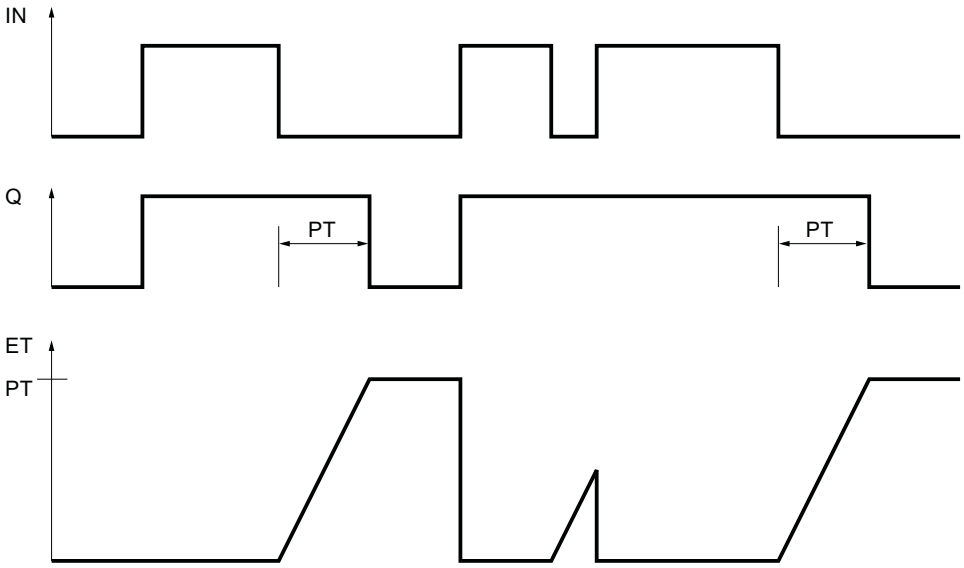
The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
IN	Input	BOOL	Start input
PT	Input	TIME	Duration of the off delay The value of the PT parameter must be positive.
Q	Output	BOOL	Operand that is reset when the time PT expires.
ET	Output	TIME	Current time value

For additional information on valid data types, refer to "See also".

Pulse diagram

The following figure shows the pulse diagram of the "Generate off-delay" instruction:



Example

The following example shows how the instruction works:

```
  SCL
 "TOF_DB".TOF(IN := "Tag_Start",
              PT := "Tag_PresetTime",
              Q => "Tag_Status",
              ET => "Tag_ElapsedTime");
```

With a change in the signal state of the "Tag_Start" operand from "0" to "1", the "Tag_Status" operand is set. When the signal state of the "Tag_Start" operand changes from "1" to "0", the time programmed for the PT parameter is started. As long as the time is running, the "Tag_Status" operand remains set. When the time has expired, the Tag_Status operand is reset. The current time value is stored in the "Tag_ElapsedTime" operand.

See also

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Overview of the valid data types (Page 2236)

Call function blocks as single or multi-instances (Page 2182)

SIMATIC Timers

S_PULSE: Assign pulse timer parameters and start

Description

The "Assign pulse timer parameters and start" instruction starts the time programmed in the T_NO parameter when a change from "0" to "1" (positive signal edge) is detected in the result of logic operation (RLO) of the S parameter. The timer runs for the programmed time (TV) as long as the signal state of the S parameter is "1".

When the signal state of the parameter S changes to "0" before the programmed time has expired, the timer is stopped and the parameter "Q" is reset to "0".

Internally, the time is made up of a time value and a time base and is programmed in the TV parameter. When the instruction starts, the programmed time value counts down to zero. The time base specifies the time increment by which the time value changes. The current time value is provided at the parameter BI.

If the timer is running and the signal state at input R changes to "1" then the current time value and the time base are also set to zero. If the timer is not running, the signal state "1" at the R input has no effect.

Parameter Q returns signal state "1" as long as the timer is running and the signal state at parameter S is "1". When the signal state of the parameter S changes to "0" before the programmed time has expired, the parameter Q returns signal state "0". If the timer is reset by parameter R or if the timer has expired then parameter Q also returns signal state "0".

The instruction data is updated with each access. It is therefore possible that the query of the data at the start of the cycle returns different values from those at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Syntax

Use the following syntax for the "Assign pulse timer parameters and start" instruction:

SCL

```
S_PULSE(T_NO := <Operand>,
        S := <Operand>,
        TV := <Operand>,
        R := <Operand>,
        Q => <Operand>,
        BI => <Operand>)
```

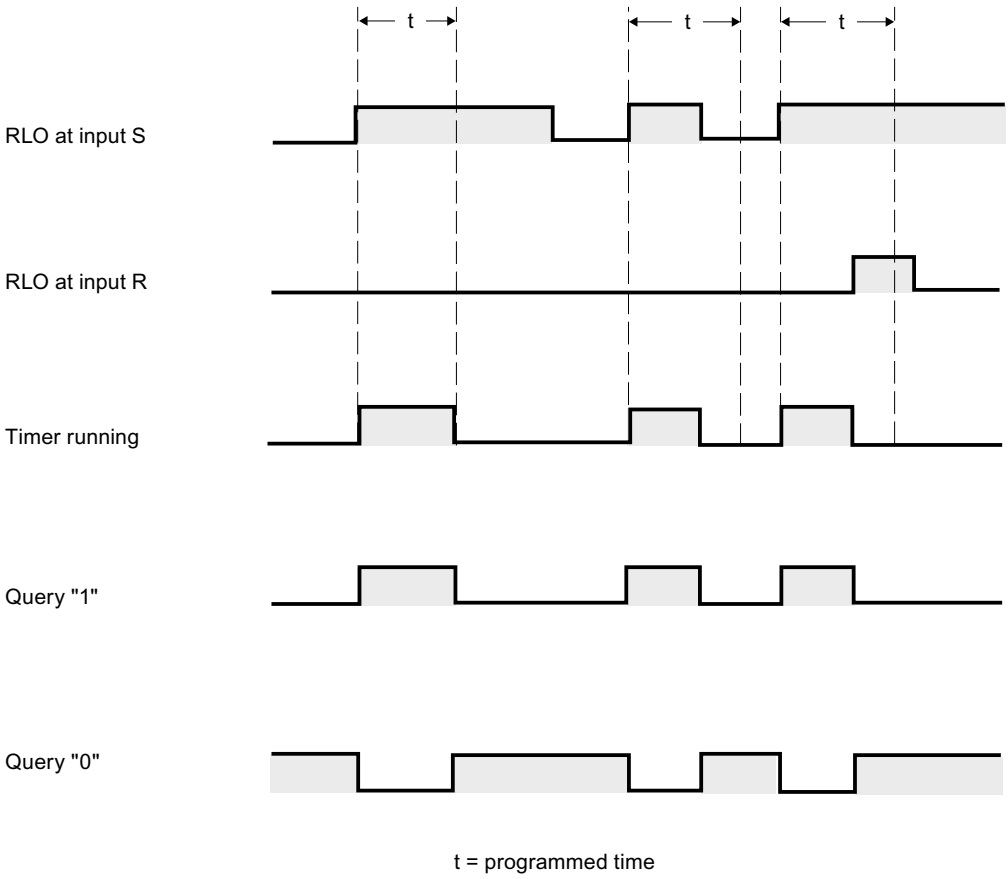
The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
T_NO	Input	TIMER, INT	The timer that is started. The number of timers depends on the CPU.
S	Input	BOOL	Start input
TV	Input	S5TIME, WORD	Preset time value
R	Input	BOOL	Reset input
Q	Output	BOOL	Status of the timer
BI	Output	WORD	Current dual-coded timer value
Function value		S5TIME	Current time value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign pulse timer parameters and start" instruction:



Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := S_PULSE(T_NO := "Timer_1",
                        S := "Tag_1",
                        TV := "Tag_Number",
                        R := "Tag_Reset",
                        Q := "Tag_Status",
                        BI := "Tag_Value");
```

"Timer_1" starts when the signal state of the "Tag_1" operand changes from "0" to "1". The timer counts down with the time value of the operand "Tag_Number" until operand "Tag_1" returns signal state "1".

If the signal state at the S parameter changes to "0" before the programmed time has elapsed, the "Tag_Status" operand is reset to "0". If the timer is reset by the R parameter or if the timer has expired, the "Tag_Status" operand also returns signal state "0".

The current time value is stored both dual-coded at the "Tag_Value" operand and returned as a function value.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

L: Load timer value (Page 5934)

S_PEXT: Assign extended pulse timer parameters and start

Description

The "Assign extended pulse timer parameters and start" instruction starts a programmed timer when a positive signal edge is detected at the S parameter. The timer runs for the programmed time (TV) even if the signal state at the S parameter changes to "0". As long as the timer runs, parameter Q returns the signal state "1".

When the timer has expired, parameter Q is reset to "0". If the signal state at the S parameter changes from "0" to "1" while the timer is running, the timer is restarted with the time programmed in the TV parameter.

Internally, the time is made up of a time value and a time base and is programmed in the TV parameter. When the instruction starts, the programmed time value counts down to zero. The time base specifies the time increment by which the time value changes. The current time value is provided at the parameter BI.

If the timer is running and the signal state at parameter R changes to "1" then the current time value and the time base are also set to zero. If the timer is not running then signal state "1" at parameter R has no effect.

The instruction data is updated with each access. It is therefore possible that the query of the data at the start of the cycle returns different values from those at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Syntax

Use the following syntax for the "Assign extended pulse timer parameters and start" instruction:

SCL

```
S_PEXT(T_NO := <Operand>,
       S := <Operand>,
       TV := <Operand>,
       R := <Operand>,
       Q => <Operand>,
       BI =><Operand>)
```

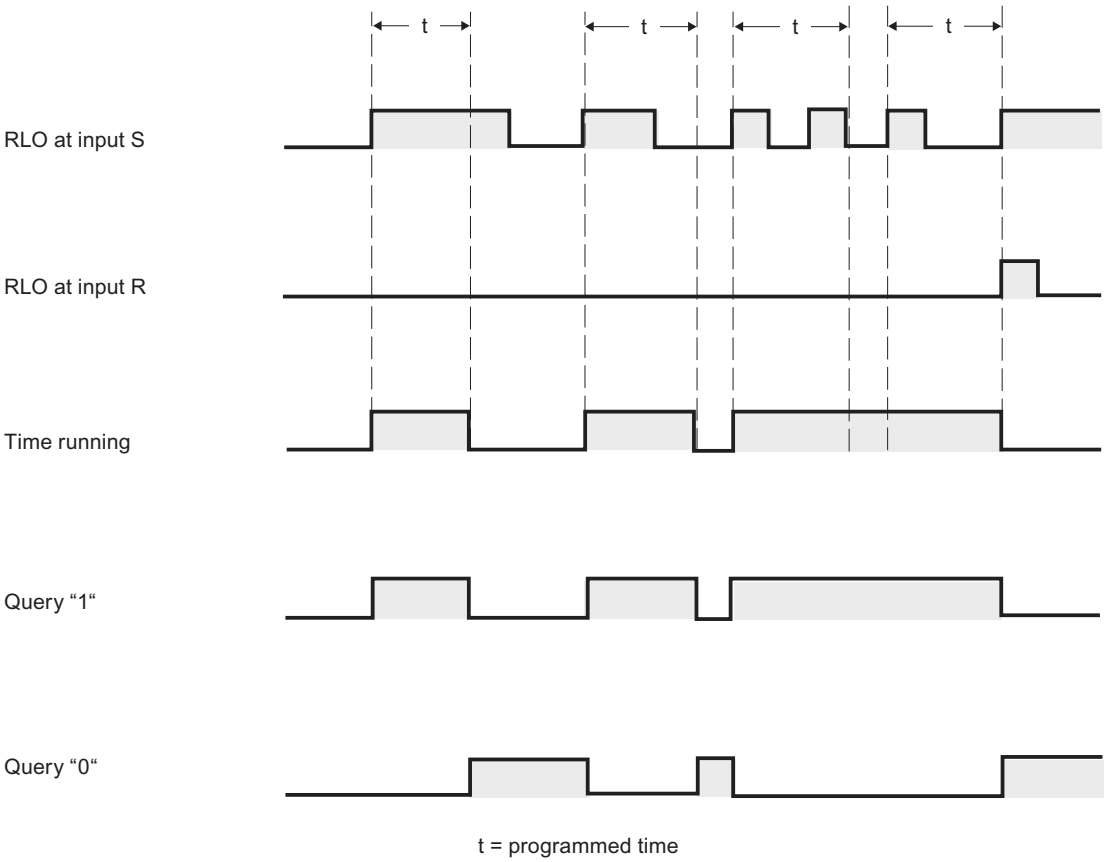
The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
T_NO	Input	TIMER, INT	The timer that is started. The number of timers depends on the CPU.
S	Input	BOOL	Start input
TV	Input	S5TIME, WORD	Preset time value
R	Input	BOOL	Reset input
Q	Output	BOOL	Status of the timer
BI	Output	WORD	Current dual-coded timer value
Function value		S5TIME	Current time value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign extended pulse timer parameters and start" instruction:



Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := S_PEXT(T_NO := "Timer_1",
                       S := "Tag_1",
                       TV := "Tag_Number",
                       R := "Tag_Reset",
                       Q := "Tag_Status",
                       BI := "Tag_Value");
```

"Timer_1" starts when the signal state of the "Tag_1" operand changes from "0" to "1". As long as the timer runs, operand "Tag_Status" returns the signal state "1". When the timer has expired, operand "Tag_Status" is reset to "0". If the signal state at the S input changes from "0" to "1" while the timer is running, the timer is restarted with the time "Tag_Number".

The current time value is stored both dual-coded at the "Tag_Value" operand and returned as a function value.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

L: Load timer value (Page 5934)

S_ODT: Assign on-delay timer parameters and start

Description

The "Assign on-delay timer parameters and start" instruction starts a programmed timer as on-delay when a positive signal edge is detected at the S parameter. The timer runs for the programmed time (TV) as long as the signal state of the S parameter is "1".

If the timer expires correctly and parameter S still has signal state "1" then parameter Q returns signal state "1". If the signal state at the S parameter changes from "1" to "0" while the timer is running, the timer is stopped. In this case, output Q is reset to signal state "0".

Internally, the time is made up of a time value and a time base and is programmed in the TV parameter. When the instruction starts, the programmed time value counts down to zero. The time base specifies the time increment by which the time value changes. The current time value is provided at the parameter BI.

If the time is running and the signal state at input R changes from "0" to "1" then the current time value and the time base are also set to zero. In this case, the signal state at parameter Q is "0". The timer is reset if the signal state at the R parameter is "1", even if the timer is not running and the result of logic operation (RLO) at the S parameter is "1".

The instruction data is updated with each access. It is therefore possible that the query of the data at the start of the cycle returns different values from those at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Syntax

Use the following syntax for the "Assign on-delay timer parameters and start" instruction:

SCL

```
S_ODT(T_NO := <Operand>,
      S := <Operand>,
      TV := <Operand>,
```

SCL

```
      R := <Operand>,
      Q => <Operand>,
      BI =><Operand>)
```

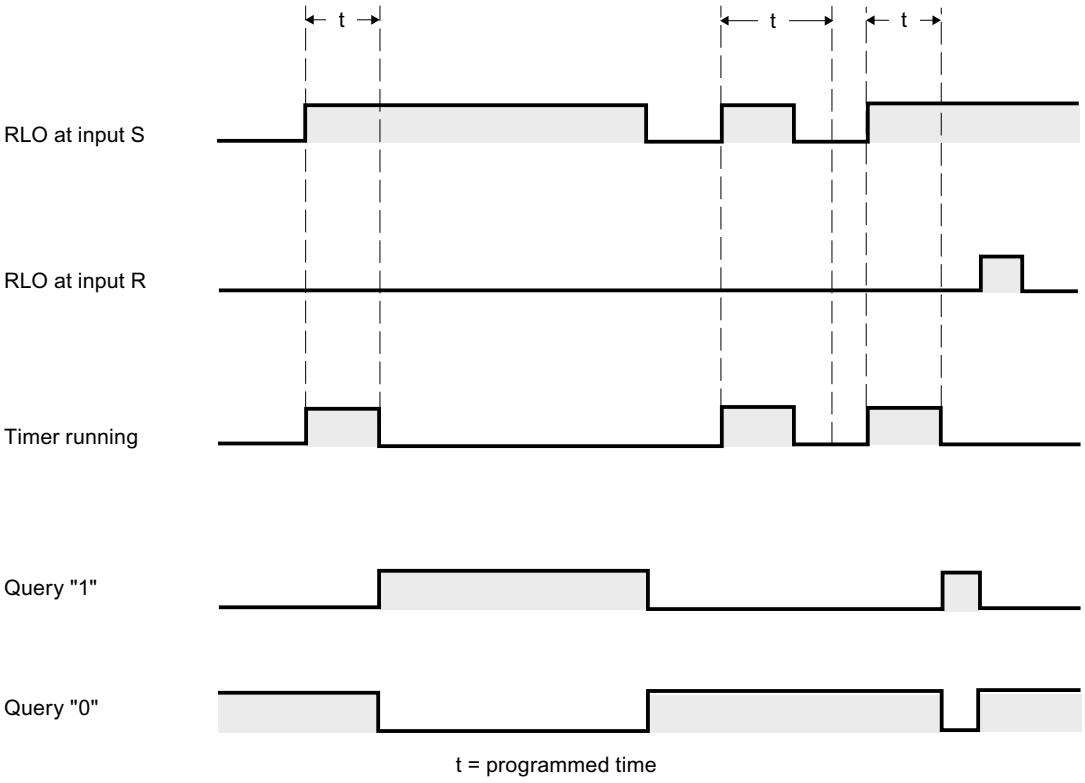
The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
T_NO	Input	TIMER, INT	The timer that is started. The number of timers depends on the CPU.
S	Input	BOOL	Start input
TV	Input	S5TIME, WORD	Preset time value
R	Input	BOOL	Reset input
Q	Output	BOOL	Status of the timer
BI	Output	WORD	Current dual-coded timer value
Function value		S5TIME	Current time value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign on-delay timer parameters and start" instruction:



Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := S_ODT(T_NO := "Timer_1",
                      S := "Tag_1",
                      TV := "Tag_Number",
                      R := "Tag_Reset",
                      Q := "Tag_Status",
                      BI := "Tag_Value");
```

"Timer_1" starts when the signal state of the "Tag_1" operand changes from "0" to "1". The timer runs for the duration "Tag_Number" as long as the signal state of operand "Tag_1" is "1".

If the timer expires correctly and operand "Tag_Status" has signal state "1" then operand "Tag_Status" is reset to "1". If the signal state of the "Tag_1" operand changes from "1" to "0" while the timer is running, the timer is stopped. In this case, operand "Tag_Status" returns signal state "0".

The current time value is stored both dual-coded at the "Tag_Value" operand and returned as a function value.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

L: Load timer value (Page 5934)

S_ODTS: Assign retentive on-delay timer parameters and start

Description

The "Assign retentive on-delay timer parameters and start" instruction starts a programmed timer when a positive signal edge is detected at the S parameter. The timer runs for the programmed time (TV) even if the signal state at the S parameter changes to "0".

If the timer expires, the "Q" parameter returns signal state "1" regardless of the signal state of the "S" parameter. If the signal state at the S parameter changes from "0" to "1" while the timer is running, the timer is restarted with the programmed time TV.

Internally, the time is made up of a time value and a time base and is programmed in the TV parameter. When the instruction starts, the programmed time value counts down to zero. The time base specifies the time increment by which the time value changes. The current time value is provided at the parameter BI.

Signal state "1" at parameterR resets the current time value and time base to "0", independent of the signal state at parameter S. In this case, the signal state at parameter Q is "0".

The instruction data is updated with each access. It is therefore possible that the query of the data at the start of the cycle returns different values from those at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Syntax

Use the following syntax for the "Assign retentive on-delay timer parameters and start" instruction:

SCL

```
S_ODTS(T_NO := <Operand>,
       S := <Operand>,
       TV := <Operand>,
       R := <Operand>,
       Q => <Operand>,
       BI =><Operand>)
```

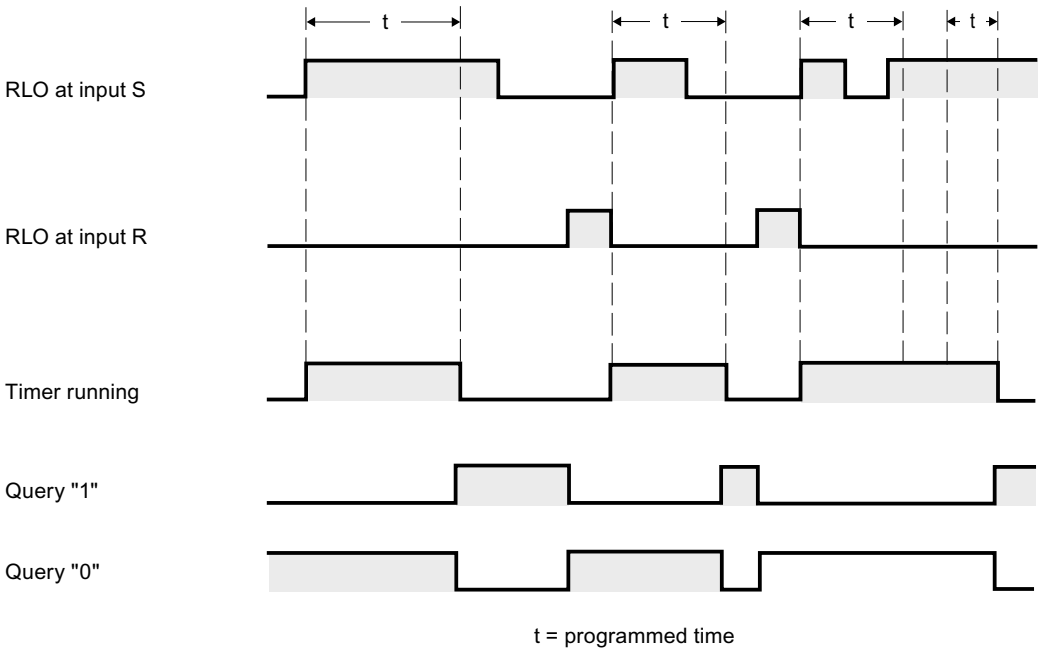
The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
T_NO	Input	TIMER, INT	The timer that is started. The number of timers depends on the CPU.
S	Input	BOOL	Start input
TV	Input	S5TIME, WORD	Preset time value
R	Input	BOOL	Reset input
Q	Output	BOOL	Status of the timer
BI	Output	WORD	Current dual-coded timer value
Function value		S5TIME	Current time value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign retentive on-delay timer parameters and start" instruction:



Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := S_ODTS(T_NO := "Timer_1",
                       S := "Tag_1",
                       TV := "Tag_Number",
                       R := "Tag_Reset",
                       Q := "Tag_Status",
                       BI := "Tag_Value");
```

"Timer_1" starts when the signal state of the "Tag_1" operand changes from "0" to "1". The timer runs for the duration "Tag_Number".

If the timer expires, operand "Tag_Status" returns signal state "1" independent of the signal state of operand "Tag_1". If the signal state of the "Tag_1" operand changes from "0" to "1" while the timer is running, the timer is restarted with the time "Tag_Number".

The current time value is stored both dual-coded at the "Tag_Value" operand and returned as a function value.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

L: Load timer value (Page 5934)

S_OFFDT: Assign off-delay timer parameters and start

Description

The "Assign off-delay timer parameters and start" instruction starts a programmed timer when a negative signal edge is detected at the S parameter. The timer runs for the programmed time (TV). As long as the timer is running or parameter S returns signal state "1", then parameter Q has signal state "1".

If the timer expires and the signal state is "0" then parameter Q is reset to signal state "0". If the signal state at parameter S changes from "0" to "1" while the timer is running, the timer is stopped. The timer is only restarted after a falling signal edge is detected at parameter S.

Internally, the time is made up of a time value and a time base and is programmed in the TV parameter. When the instruction starts, the programmed time value counts down to zero. The time base specifies the time increment by which the time value changes. The current time value is provided at the parameter BI.

Signal state "1" at parameter R resets the current time value and time base to "0". In this case, the signal state at parameter Q is "0".

The instruction data is updated with each access. It is therefore possible that the query of the data at the start of the cycle returns different values from those at the end of the cycle.

Note

In the time cell, the operating system reduces the time value in an interval specified by the time base by one unit until the value equals "0". The decrementation is performed asynchronously to the user program. The resulting timer is therefore at maximum up to one time interval shorter than the desired time base.

You can find an example of how a time cell can be formed under: See also "L: Load timer value".

Syntax

Use the following syntax for the "Assign off-delay timer parameters and start" instruction:

SCL

```
S_OFFDT(T_NO := <Operand>,
        S := <Operand>,
        TV := <Operand>,
        R := <Operand>,
        Q => <Operand>,
        BI =><Operand>)
```

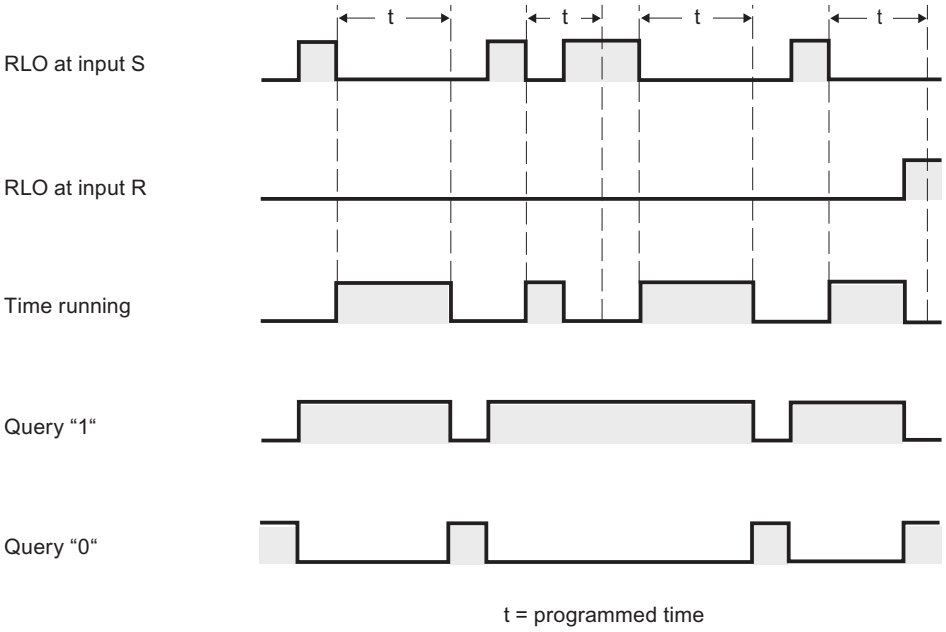
The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
T_NO	Input	TIMER, INT	The timer that is started. The number of timers depends on the CPU.
S	Input	BOOL	Start input
TV	Input	S5TIME, WORD	Preset time value
R	Input	BOOL	Reset input
Q	Output	BOOL	Status of the timer
BI	Output	WORD	Current dual-coded timer value
Function value		S5TIME	Current time value

For additional information on valid data types, refer to "See also".

Pulse timing diagram

The following figure shows the pulse timing diagram of the "Assign off-delay timer parameters and start" instruction:



Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := S_OFFDT(T_NO := "Timer_1",
                        S := "Tag_1",
                        TV := "Tag_Number",
                        R := "Tag_Reset",
                        Q := "Tag_Status",
```

SCL

```
                                BI := "Tag_Value");
```

"Timer_1" starts when the signal state of the "Tag_1" operand changes from "0" to "1". The timer runs for the duration "Tag_Number". As long as the timer is running or operand "Tag_1" returns signal state "1", then operand "Tag_Status" has signal state "1".

If the timer expires and the signal state of operand "Tag_1" is "0" then operand "Tag_Status" is reset to signal state "0". If the signal state of the "Tag_1" operand changes from "0" to "1" while the timer is running, the timer is reset. The timer is only restarted after a falling edge is detected at parameter S.

The current time value is stored both dual-coded at the "Tag_Value" operand and returned as a function value.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

L: Load timer value (Page 5934)

Counter operations

IEC Counters

CTU: Count up

Description

You can use the "Count up" instruction to increment the value at the CV parameter. When the signal state of the parameter CU changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value of the parameter CV is incremented by one. When the instruction is executed for the first time the the current count of the CV parameter is set to zero. The counter value is increased each time a positive signal edge is detected, until it reaches the high limit of the data type (INT). When the high limit is reached, the signal state of the CU parameter no longer has an effect on the instruction.

You can query the count status of the Q parameter. The signal state of the Q parameter is determined by the PV parameter. When the current counter value is greater than or equal to the value of the PV parameter, the Q parameter is set to signal state "1". In all other cases, the signal state of the Q parameter is "0". You can also specify a constant for the PV parameter.

The value of the CV parameter is reset to zero when the signal state at the R parameter changes to "1". As long as the signal state of the R parameter is "1", the signal state of the CU parameter has no effect on the instruction.

Note

Only use a counter at a single location in the program to avoid risk of counting errors.

Each call of the "Count up" instruction must be assigned an IEC counter in which the instruction data is stored. You can declare an IEC counter as follows:

- Declaration of a data block of the type CTU (for example, "CTU_DB")
- Declaration as a local tag of the CTU type in the "Static" section of a block (for example, #MyCTU_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of "Count up" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized with the value "1" for the R parameter of the instruction in a startup OB. Instances of the "Count up" instruction in another block can be reset by initializing the higher-level block, for example.

Syntax

The following syntax is used for the "Count up" instruction:

Table 9-167 Data block of system data type IEC_counter (global DB)

SCL

```
<IEC_Counter_DB>.CTU(CU := <Operand>,
                     R := <Operand>,
                     PV := <Operand>,
                     Q => <Operand>,
                     CV => <Operand>)
```

Table 9-168 Local tag

SCL

```
#myLocal_counter(CU := <Operand>,
                 R := <Operand>,
                 PV := <Operand>,
                 Q => <Operand>,
                 CV => <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
CU	Input	BOOL	Count input
R	Input	BOOL	Reset input
PV	Input	INT	Value at which the output Q is set.
Q	Output	BOOL	Counter status
CV	Output	INT	Current counter value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"CTU_DB".CTU(CU := "Tag_Start",
             R := "Tag_Reset",
             PV := "Tag_PresetValue",
             Q => "Tag_Status",
             CV => "Tag_CounterValue");
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the "Count up" instruction executes and the current counter value of the "Tag_CounterValue" operand is incremented by one. With each additional positive signal edge, the counter value is incremented until it reaches the high limit value of the (INT =32767) data type.

The "Tag_Status" output has signal state "1" as long as the current counter value is greater than or equal to the value of the "Tag_PresetValue" operand. In all other cases, the "Tag_Status" output has signal state "0". The current counter value is stored in the "Tag_CounterValue" operand.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Call function blocks as single or multi-instances (Page 2182)

CTD: Count down

Description

The "Count down" instruction is used to decrement the value at the parameter CV. When the signal state of the CD parameter changes from "0" to "1" (positive signal edge), the instruction is executed and the current counter value of the CV parameter is decremented by one. When the instruction is executed the first time, the counter value of the CV parameter will be set to the value of the PV parameter. Each time a positive signal edge is detected, the counter is

decremented until it reaches the low limit value of the specified data type (INT). When the low limit is reached, the signal state of the CD parameter no longer has an effect on the instruction.

You can query the count status of the Q parameter. If the current counter value is less than or equal to zero, the Q parameter is set to signal state "1". In all other cases, the signal state of the Q parameter is "0".

The value of the CV parameter is set to the value of the PV parameter when the signal state of the LD parameter changes to "1". As long as the signal state of the LD parameter is "1", the signal state of the CD parameter has no effect on the instruction.

Note

Only use a counter at a single location in the program to avoid risk of counting errors.

Each call of the "Count down" instruction must be assigned an IEC counter in which the instruction data is stored. You can declare an IEC counter as follows:

- Declaration of a data block of the type CTD (for example, "CTD_DB")
- Declaration as a local tag of the CTD type in the "Static" section of a block (for example, #MyCTD_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Count down" instruction during cold restarts. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized with the value "1" for the LD parameter of the instruction in a startup OB. In this case the desired initial value for the CV parameter is specified in the PV parameter. Instances of the "Count down" instruction in another block can be reset by initializing the higher-level block, for example.

Syntax

The following syntax is used for the "Count down" instruction:

Table 9-169 Data block of system data type IEC_counter (global DB)

SCL

```
<IEC_Counter_DB>.CTD(CD := <Operand>,
                     LD := <Operand>,
                     PV := <Operand>,
                     Q => <Operand>,
                     CV => <Operand>)
```

Table 9-170 Local tag

SCL

```
#myLocal_counter(CD := <Operand>,
                 LD := <Operand>,
```

SCL

```
                PV := <Operand>,
                Q => <Operand>,
                CV => <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
CD	Input	BOOL	Count input
LD	Input	BOOL	Load input
PV	Input	INT	Value at which the output Q is set.
Q	Output	BOOL	Counter status
CV	Output	INT	Current counter value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"CTD_DB".CTD(CD := "Tag_Start",
             LD := "Tag_Load",
             PV := "Tag_PresetValue",
             Q => "Tag_Status",
             CV => "Tag_CounterValue");
```

When the signal state of the "Tag_Start" changes from "0" to "1", the "Count down" instruction executes and the value of the "Tag_CounterValue" operand is decremented by one. With each additional positive signal edge, the counter value will be decremented until it reaches the low limit value of the (INT = -32768) data type.

The operand "Tag_Status" has the signal state "1" as long as the current counter value is less than or equal to zero. In all other cases, the "Tag_Status" output has signal state "0". The current counter value is stored in the "Tag_CounterValue" operand.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Call function blocks as single or multi-instances (Page 2182)

CTUD: Count up and down

Description

Use the "Count up and down" instruction to increment or decrement the counter value at the CV parameter. When the signal state of the CU parameter changes from "0" to "1" (positive signal edge), the current counter value of the CV parameter is incremented by one. When the signal state of the CD parameter changes from "0" to "1" (positive signal edge), the counter value of the CV parameter is decremented by one. If there is a positive signal edge at the CU and CD inputs in one program cycle, the current counter value of the CV parameter remains unchanged.

The counter value can be incremented until it reaches the high limit value of the specified data type (INT). When the high limit is reached, the counter value is no longer incremented on a positive signal edge. When the low limit of the specified data type (INT) is reached, the counter value is not decremented any further.

When the signal state of the LD parameter changes to "1", the counter value of the CV parameter is set to the value of the PV parameter. As long as the LD parameter has the signal state "1", the signal state of the CU and CD parameters has no effect on the instruction.

The counter value is set to zero when the signal state of the R parameter changes to "1". As long as the R parameter has signal state "1", a change in the signal state at the CU, CD, and LD parameters will not affect the "Count up and down" instruction.

You can scan the current status of the up counter based on the value of the QU parameter. When the current counter value is greater than or equal to the value of the PV parameter, the QU parameter is set to signal state "1". In all other cases, the signal state of the QU parameter is "0". You can also specify a constant for the PV parameter.

You can scan the current status of the down counter based on the value of the QD parameter. If the current counter value is less than or equal to zero, the QD parameter is set to signal state "1". In all other cases, the signal state of the QD parameter is "0".

Note

Only use a counter at a single location in the program to avoid risk of counting errors.

Each call of the "Count up and down" instruction must be assigned an IEC counter in which the instruction data is stored. You can declare an IEC counter as follows:

- Declaration of a data block of the type CTUD (for example, "CTUD_DB")
- Declaration as a local tag of the CTUD type in the "Static" section of a block (for example, #MyCTUD_COUNTER)

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the IEC counter is stored in its own data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Count up and down" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, you must call the instances to be initialized with the following parameter values in a restart OB:

- When used as counter up, the value at the R parameter must be set to "1".
- When used as down counter, the value at the LD parameter must be set to "1". In this case you specify the desired initial value for the CV parameter in the PV parameter.

If instances of the "Count up and down" instruction are located within another block, you can reset these instances by initializing the higher-level block, for example.

Syntax

The following syntax is used for the "Count up and down" instruction:

Table 9-171 Data block of system data type IEC_counter (global DB)

SCL

```
<IEC_Counter_DB>.CTUD(CU := <Operand>,
                      CD := <Operand>,
                      R := <Operand>,
                      LD := <Operand>,
                      PV := <Operand>,
                      QU=> <Operand>,
                      QD := <Operand>,
                      CV=> <Operand>)
```

Table 9-172 Local tag

SCL

```
myLocal_counter(CU := <Operand>,
                CD := <Operand>,
                R := <Operand>,
                LD := <Operand>,
                PV := <Operand>,
                QU=> <Operand>,
                QD := <Operand>,
                CV=> <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
CU	Input	BOOL	Count up input
CD	Input	BOOL	Count down input
R	Input	BOOL	Reset input
LD	Input	BOOL	Load input
PV	Input	INT	Value at which the output QU is set.
QU	Output	BOOL	Status of the counter up
QD	Output	BOOL	Status of the counter down
CV	Output	INT	Current counter value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"CTUD_DB".CTUD(CU := "Tag_Start1",
               CD := "Tag_Start2",
               LD := "Tag_Load",
               R := "Tag_Reset",
               PV := "Tag_PresetValue",
               QU => "Tag_CU_Status",
               QD => "Tag_CD_Status",
               CV => "Tag_CounterValue");
```

If the "Tag_Start1" operand has a positive signal edge in the signal state, the current counter value is incremented by one and stored in the "Tag_CounterValue" operand. If the "Tag_Start2" operand has a positive signal edge in the signal state, the counter value is decremented by one and is also stored in the "Tag_CounterValue" operand. The counter value is incremented on a positive signal edge of the CU parameter until it reaches the high limit value of 32 767. The counter value is decremented on a positive signal edge of the CD parameter until it reaches the low limit value of INT = -32 768.

The operand "Tag_CU_Status" has the signal state "1" as long as the current counter value is greater than or equal to the value of the operand "Tag_PresetValue". In all other cases, the "Tag_CU_Status" output has signal state "0".

The operand "Tag_CD_Status" has the signal state "1" as long as the current counter value is less than or equal to zero. In all other cases, the "Tag_CD_Status" output has signal state "0".

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Call function blocks as single or multi-instances (Page 2182)

SIMATIC Counters

S_CU: Assign parameters and count up

Description

You can use the "Assign parameters and count up" instruction to increment the value of a counter. When the signal state of the CU parameter changes from "0" to "1" (positive signal edge), the current counter value is incremented by one. The current counter value is provided at the parameter CV. The counter value is incremented until the limit of "999" is reached. When the limit value is reached, the counter value is no longer incremented on a positive signal edge.

When the signal state of the parameter S changes from "0" to "1", the counter value is set to the value of the parameter PV. If the counter is set and if the result of logic operation (RLO)

at the CU input is "1", the counter counts once in the next cycle, even when no signal edge change is detected.

The counter value is set to zero when the signal state of the R parameter changes to "1". As long as the R parameter has the signal state "1", a change in the signal state of the parameters CU and S has no effect on the counter value.

The signal state at parameter Q is "1" if the counter value is greater than zero. When the counter value equals zero, parameter Q returns signal state "0".

Note

Only use a counter at a single location in the program to avoid risk of counting errors.

Syntax

The following syntax is used for the "Assign parameters and count up" instruction:

SCL

```
S_CU(C_NO := <Operand>,
     CU := <Operand>,
     S := <Operand>,
     PV := <Operand>,
     R := <Operand>,
     Q => <Operand>,
     CV=> <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
C_NO	Input	COUNTER, INT	Counters The number of counters depends on the CPU.
CU	Input	BOOL	Count up input
S	Input	BOOL	Input for presetting the counter
PV	Input	WORD	Preset counter value (C#0 to C#999) in BCD format
R	Input	BOOL	Reset input
Q	Output	BOOL	Status of the counter
CV	Output	WORD, S5TIME, DATE	Current counter value
Function value		WORD, S5TIME, DATE	Current counter value in BCD format

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := S_CU(C_NO := "Counter_1",
                     CU := "Tag_Start",
                     S := "Tag_1",
                     PV := "Tag_PresetValue",
                     R := "Tag_Reset",
                     Q => "Tag_Status",
                     CV => "Tag_Value");
```

If the signal state of the "Tag_Start" parameter changes from "0" to "1" (positive signal edge) and the current counter value is less than "999", the counter value is incremented by one. If the signal state of the "Tag_1" input changes from "0" to "1", the counter value in BCD format is set to the value of the "Tag_PresetValue" operand. The counter value is reset to "0" when the "Tag_Reset" operand has signal state "1".

The current counter value is stored in hexadecimal form in the operand "Tag_Value".

The "Tag_Status" output has the signal state "1" as long as the current counter value is not equal to "0". The current counter value is returned in the "Tag_Value" operand and as a function value.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

S_CD: Assign parameters and count down

Description

You can use the "Assign parameters and count down" instruction to decrement the value of a counter . When the signal state of the CD parameter changes from "0" to "1" (positive signal edge), the current counter value is decreased by one. The current counter value is provided at the parameter CV. The counter value is decreased until the low limit of "0" is reached. When the low limit value is reached, the counter value is no longer decreased on a positive signal edge.

When the signal state of the S parameter changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if the result of logic operation (RLO) at the CD parameter is "1", the counter counts once in the next cycle, even when no signal edge change is detected.

The counter value is set to zero when the signal state of the R parameter changes to "1". As long as the R parameter has the signal state "1", a change in the signal state of the parameters CD and S has no effect on the counter value.

The signal state at parameter Q is "1" if the counter value is greater than zero. When the counter value equals zero, parameter Q returns signal state "0".

Note

Only use a counter at a single location in the program to avoid risk of counting errors.

Syntax

The following syntax is used for the "Assign parameters and count down" instruction:

SCL

```
S_CD(C_NO := <Operand>,
     CD := <Operand>,
     S := <Operand>,
     PV := <Operand>,
     R := <Operand>,
     Q => <Operand>,
     CV=> <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
C_NO	Input	COUNTER, INT	Counters The number of counters depends on the CPU.
CD	Input	BOOL	Count down input
S	Input	BOOL	Input for presetting the counter
PV	Input	WORD	Preset counter value (C#0 to C#999) in BCD format
R	Input	BOOL	Reset input
Q	Output	BOOL	Status of the counter
CV	Output	WORD, S5TIME, DATE	Current counter value
Function value		WORD, S5TIME, DATE	Current counter value in BCD format

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := S_CD(C_NO := "Counter_1",
```

SCL

```
                    CD := "Tag_Start",
                    S := "Tag_1",
                    PV := "Tag_PresetValue",
                    R := "Tag_Reset",
                    Q => "Tag_Status",
                    CV => "Tag_Value");
```

When the signal state of the "Tag_Start" operand changes from "0" to "1" (positive signal edge) and the current counter value is greater than "0", the counter value is decreased by one. If the signal state of the "Tag_1" operand changes from "0" to "1", the counter value is set to the value of the "Tag_PresetValue" operand. The counter value is reset to "0" when the "Tag_Reset" operand has signal state "1".

The current counter value is stored in the "Tag_Value" operand.

The "Tag_Status" operand returns signal state "1" as long as the current counter value is not equal to "0". The current counter value is returned in the "Tag_Value" operand and as a function value.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

S_CUD: Assign parameters and count up / down

Description

You can use the "Assign parameters and count up / down" instruction to increment and decrement the value of a counter. When the signal state of the CU parameter changes from "0" to "1" (positive signal edge), the current counter value is incremented by one. When the signal state of the CD parameter changes from "0" to "1" (positive signal edge), the counter value is decreased by one. The current counter value is provided at the parameter CV. If there is a positive signal edge at the parameters CU and CD in one program cycle, the counter value remains unchanged.

The counter value is incremented until the high limit value of "999" is reached. When the high limit value is reached, the counter value is no longer incremented on a positive signal edge. When the low limit value "0" is reached, the counter value is not decremented any further.

When the signal state of the S parameter changes from "0" to "1", the counter value is set to the value of the PV parameter. If the counter is set and if the result of logic operation (RLO) of the CU and CD parameters is "1", the counter will count once in the next cycle, even if no signal edge change was detected.

The counter value is set to zero when the signal state of the R parameter changes to "1". As long as the R parameter has the signal state "1", processing of the signal state of the parameters CU, CD and S has no effect on the counter value.

The signal state at parameter Q is "1" if the counter value is greater than zero. When the counter value equals zero, parameter Q returns signal state "0".

Note

Only use a counter at a single location in the program to avoid risk of counting errors.

Syntax

The following syntax is used for the "Assign parameters and count up / down" instruction:

SCL

```
S_CUD(C_NO := <Operand>,
      CU := <Operand>,
      CD := <Operand>,
      S := <Operand>,
      PV := <Operand>,
      R := <Operand>,
      Q => <Operand>,
      CV=> <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
C_NO	Input	COUNTER, INT	Counters The number of counters depends on the CPU.
CU	Input	BOOL	Count up input
CD	Input	BOOL	Count down input
S	Input	BOOL	Input for presetting the counter
PV	Input	WORD	Preset counter value (C#0 to C#999) in BCD format
R	Input	BOOL	Reset input
Q	Output	BOOL	Status of the counter
CV	Output	WORD, S5TIME, DATE	Current counter value (hexadecimal)
Function value		WORD, S5TIME, DATE	Current counter value in BCD format

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := S_CD(C_NO := "Counter_1",
                     CU := "Tag_CU",
                     CD := "Tag_CD",
                     S := "Tag_1",
                     PV := "Tag_PresetValue",
                     R := "Tag_Reset",
                     Q => "Tag_Status",
                     CV => "Tag_Value");
```

When a positive signal edge is detected in the signal state of the "Tag_CU" operand and the current counter value is less than "999", the counter value is incremented by one. When a positive signal edge is detected in the signal state of the "Tag_CD" operand and the current counter value is greater than "0", the counter value is decremented by one.

If the signal state of the "Tag_1" operand changes from "0" to "1", the counter value is set to the value of the "Tag_PresetValue" operand. The counter value is reset to "0" when the "Tag_Reset" operand has signal state "1".

The current counter value is stored in the "Tag_Value" operand.

The "Tag_Status" operand returns signal state "1" as long as the current counter value is not equal to "0". The current counter value is returned in the "Tag_Value" operand and as a function value.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Math functions

ABS: Form absolute value

Description

Use the "Form absolute value" instruction to calculate the absolute value of an input value and to save the result in the specified operands.

Syntax

Use the following syntax for the "Form absolute value" instruction:

```
SCL
ABS(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
<Expression>	Input	INT, DINT, REAL	Input value
Function value		INT, DINT, REAL	Absolute value of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result1" := ABS("Tag_Value");
"Tag_Result2" := ABS("Tag_Value1"*"Tag_Value2");
```

The absolute value of the input value is returned as function value in the format of the input value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	-2
Tag_Result1	2
Tag_Value1	4
Tag_Value2	-1
Tag_Result2	4

See also

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Overview of the valid data types (Page 2236)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

MIN: Get minimum

Description

The instruction "Get minimum" compares the values of the specified inputs and returns the lowest value as the result. The instruction is only executed if the tags of all inputs are of the same data type.

A minimum of two and a maximum of 32 inputs can be specified for execution of the instruction.

Syntax

The following syntax is used for the instruction "Get minimum":

SCL

```
MIN(IN1:= <operand>,
    IN2 := <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
IN1	Input	INT, DINT, REAL, TIME, DATE, DT	First input value
IN2	Input	INT, DINT, REAL, TIME, DATE, DT	Second input value
INn	Input	INT, DINT, REAL, TIME, DATE, DT	Additionally inserted inputs whose values are to be compared
Function value		INT, DINT, REAL, TIME, DATE, DT	Result of the instruction

For more information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := MIN(IN1 := "Tag_Value1",
                    IN2 := "Tag_Value2",
                    IN3 := "Tag_Value3");
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	Tag_Value1	12 222
IN2	Tag_Value2	14 444
IN3	Tag_Value3	13 333
Function value	Tag_Result	12 222

The instruction compares the values of the specified operands and copies the lowest value ("Tag_Value1") to the "Tag_Result" operand.

See also

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Overview of the valid data types (Page 2236)

MAX: Get maximum

Description

The instruction "Get maximum" compares the values of the specified inputs and returns the greatest value as the result. The instruction is only executed if the tags of all inputs are of the same data type.

A minimum of two and a maximum of 32 inputs can be specified for execution of the instruction.

Syntax

The following syntax is used for the instruction "Get maximum":

SCL

```
MAX(IN1:= <Operand>,
    IN2 := <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
IN1	Input	INT, DINT, REAL, TIME, DATE, DT	First input value
IN2	Input	INT, DINT, REAL, TIME, DATE, DT	Second input value
INn	Input	INT, DINT, REAL, TIME, DATE, DT	Additionally inserted inputs whose values are to be compared
Function value		INT, DINT, REAL, TIME, DATE, DT	Result of the instruction

For more information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := MAX(IN1 := "Tag_Value1",
                    IN2 := "Tag_Value2",
                    IN3 := "Tag_Value3");
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	Tag_Value1	12 222
IN2	Tag_Value2	14 444
IN3	Tag_Value3	13 333
Function value	Tag_Result	14 444

The instruction compares the values of the specified operands and copies the greatest value ("Tag_Value2") to the "Tag_Result" operand.

See also

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Overview of the valid data types (Page 2236)

LIMIT: Set limit value

Description

The "Set limit value" instruction limits the value of the parameter IN to the values of the parameters MN and MX. The value of the MN parameter may not be greater than the value of the MX parameter.

If the value of the IN parameter fulfills the MN condition <=IN <= MX, it is returned as the result of the instruction. If the condition is not fulfilled and the IN input value is less than the MN low limit, the value of the MN parameter is returned as the result. If the high limit MX is exceeded, the value of the MX parameter is returned as the result.

If the value of the MN input is greater than the value of the MX input, the result is undefined.

The instruction is only executed if the operands of all parameters are of the same data type.

Syntax

The following syntax is used for the "Set limit value" instruction:

```
SCL
LIMIT(MN := <Operand>,
```

SCL

```
      IN := <Operand>,
      MX := <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
MN	Input	INT, DINT, REAL, TIME, DATE, DT	Low limit
IN	Input	INT, DINT, REAL, TIME, DATE, DT	Input value
MX	Input	INT, DINT, REAL, TIME, DATE, DT	High limit
Function value		INT, DINT, REAL, TIME, DATE, DT	Result of the instruction

For more information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := LIMIT(MN := "Tag_Minimum",
                      IN := "Tag_Value",
                      MX := "Tag_Maximum");
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
MN	Tag_Minimum	12 000
IN	Tag_Value	8 000
MX	Tag_Maximum	16 000
Function value	Tag_Result	12 000

The value of operand "Tag_Value" is compared with the values of operands "Tag_Minimum" and "Tag_Maximum". Because the value of the operand "Tag_Value" is less than the low limit value, the value of operand "Tag_Minimum" will be copied to operand "Tag_Result".

See also

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Overview of the valid data types (Page 2236)

SQR: Form square

Description

Use the "Form square" instruction to square the input value and save the result in the specified operand.

Syntax

Use the following syntax for the instruction "Form square":

```
SCL
SQR(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
<Expression>	Input	Floating-point numbers	Input value
Function value		Floating-point numbers	Square of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result1" := SQR("Tag_Value");
"Tag_Result2" := SQR((SQR("Tag_Value1"))*"Tag_Value2);
```

The square of the input value is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	2.5
Tag_Result1	6.25
Tag_Value1	6.0
Tag_Value2	2.0
Tag_Result2	5184.0

See also

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Overview of the valid data types (Page 2236)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

SQRT: Form square root

Description

Use the "Form square root" instruction to calculate the square root of the the input value and save the result in the specified operand. The instruction has a positive result if the input value is greater than zero. If input values are less than zero, the instruction returns an invalid floating-point number. If the input value is "-0", the result is also "-0".

Syntax

Use the following syntax for the "Form square root" instruction:

SCL

```
SQRT(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
<Expression>	Input	Floating-point numbers	Input value
Function value		Floating-point numbers	Square root of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result1" := SQRT("Tag_Value");
"Tag_Result2" := SQRT((SQR("Tag_Value1"))+"Tag_Value2");
```

The square root of the input value is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	4.0
Tag_Result1	2.0
Tag_Value1	3.0
Tag_Value2	16.0
Tag_Result2	5.0

See also

Entering SCL instructions (Page 2697)
Editing SCL instructions (Page 2716)
Overview of the valid data types (Page 2236)
Expressions (Page 2681)
Operators and operator precedence (Page 2687)

LN: Form natural logarithm

Description

You can use the "Form natural logarithm" instruction to calculate the natural logarithm to base e (e=2.718282) of the input value. The instruction has a positive result if the input value is greater than zero. If input values are less than zero, the instruction returns an invalid floating-point number.

Syntax

Use the following syntax for the "Form natural logarithm" instruction:

SCL
```
LN(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
<Expression>	Input	Floating-point numbers	Input value
Function value		Floating-point numbers	Natural logarithm of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL
```
"Tag_Result1" := LN("Tag_Value");
"Tag_Result2" := LN("Tag_Value1"+"Tag_Value2");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	2.5
Tag_Result1	0.916
Tag_Value1	1.5
Tag_Value2	3.2
Tag_Result2	1.548

See also

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Overview of the valid data types (Page 2236)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

EXP: Form exponential value

Description

The "Form exponential value" instruction calculates the exponent from the base e (e = 2.718282) and the input value and saves the result in the specified operand.

Syntax

The following syntax is used for the "Form exponential value" instruction:

```
SCL
EXP(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
<Expression>	Input	Floating-point numbers	Input value
Function value		Floating-point numbers	Exponential value of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result1" := EXP("Tag_Value");
```

```
SCL
"Tag_Result2" := EXP("Tag_Value1"/"Tag_Value2");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	20.5
Tag_Result1	799 902 200
Tag_Value1	15.5
Tag_Value2	30.2
Tag_Result2	1.671

See also

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Overview of the valid data types (Page 2236)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

SIN: Form sine value

Description

Use the "Form sine value" instruction to calculate the sine of the input value. The input value must be given in the radian measure.

Syntax

Use the following syntax for the "Form sine value" instruction:

```
SCL
SIN(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
<Expression>	Input	Floating-point numbers	Input value (size of an angle in the radian measure)
Function value		Floating-point numbers	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := SIN("Tag_Value");
```

The result of the instruction is returned in the "Tag_Result" operand as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	+1.570796 (π/2)
Tag_Result	1.0

See also

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Overview of the valid data types (Page 2236)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

COS: Form cosine value

Description

Use the "Form cosine value" instruction to calculate the cosine of the input value. The input value must be given in the radian measure.

Syntax

Use the following syntax for the "Form cosine value" instruction:

```
SCL
COS(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
<Expression>	Input	Floating-point numbers	Input value (size of an angle in the radian measure)
Function value		Floating-point numbers	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := COS("Tag_Value");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	+1.570796 (π/2)
Tag_Result	0

See also

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Overview of the valid data types (Page 2236)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

TAN: Form tangent value

Description

Use the "Form tangent value" instruction to calculate the sine of the input value. The input value must be given in the radian measure.

Syntax

Use the following syntax for the "Form tangent value" instruction:

```
SCL
TAN(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
<Expression>	Input	Floating-point numbers	Input value (size of an angle in the radian measure)
Function value		Floating-point numbers	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := TAN("Tag_Value");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	+3.141593 (π)
Tag_Result	0

See also

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Overview of the valid data types (Page 2236)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

ASIN: Form arcsine value

Description

You can use the "Form arcsine value" instruction to calculate the size of the angle from a sine value, which corresponds to this value. Only valid floating-point numbers within the range -1 to +1 can be specified as input values. The calculated angle size is given in the radian measure and can range in value from -π/2 to +π/2.

Syntax

Use the following syntax for the "Form arcsine value" instruction:

```
SCL
ASIN(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
<Expression>	Input	Floating-point numbers	Sine value
Function value		Floating-point numbers	Size of angle in the radian measure

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := ASIN("Tag_Value");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	1.0
Tag_Result	+1.570796 (π/2)

See also

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Overview of the valid data types (Page 2236)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

ACOS: Form arccosine value

Description

You can use the "Form arccosine value" instruction to calculate the size of the angle from a cosine value, which corresponds to this value. Only valid floating-point numbers within the range -1 to +1 can be specified as input values. The calculated angle size is given in the radian measure and can range in value from 0 to +π.

Syntax

Use the following syntax for the "Form arccosine value" instruction:

```
SCL
ACOS(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
<Expression>	Input	Floating-point numbers	Cosine value
Function value		Floating-point numbers	Size of angle in the radian measure

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := ACOS("Tag_Value");
```

The result of the instruction is returned in the "Tag_Result" operand as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	0
Tag_Result	+1.570796 (π/2)

See also

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Overview of the valid data types (Page 2236)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

ATAN: Form arctangent value

Description

You can use the "Form arctangent value" instruction to calculate the size of the angle from a tangent value, which corresponds to this value. Only valid floating-point numbers may be specified as input values. The calculated angle size is given in the radian measure and can range in value from -π/2 to +π/2.

Syntax

Use the following syntax for the "Form arctangent value" instruction:

```
SCL
ATAN(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
<Expression>	Input	REAL	Tangent value
Function value		REAL	Size of angle in the radian measure

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := ATAN("Tag_Value");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value	1.0
Tag_Result	+0.785398 (π/4)

See also

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Overview of the valid data types (Page 2236)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

Move operations

BLKMOV: Move block

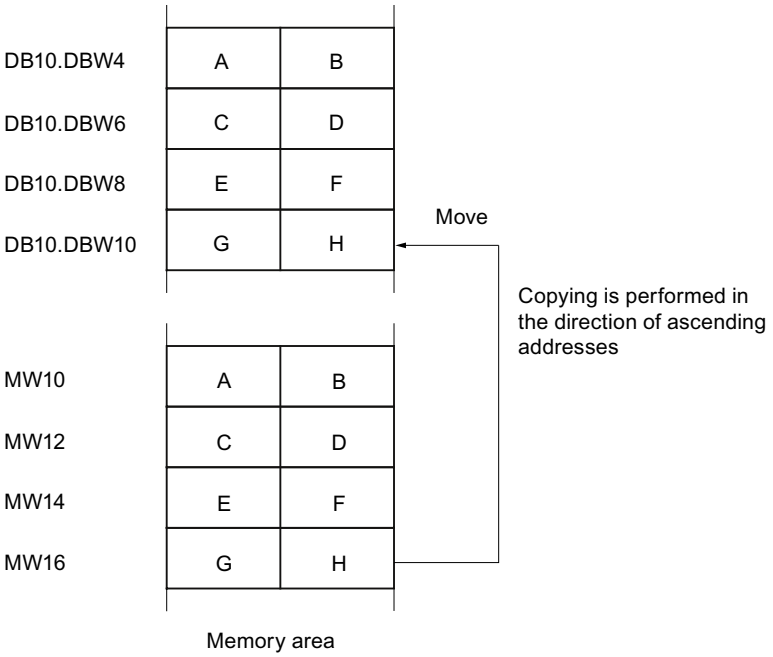
Description

The instruction "Move block" copies the contents of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use ANY pointer to define the source and destination area.

Note

In this instruction, you can only use data types which are valid on a CPU of the S7-300/400 series.

The following figure shows the principle of the move operation:



Consistency of source and destination data

Note that while the instruction "Move block" is being executed, the source data remains unchanged; otherwise the consistency of the destination data cannot be guaranteed.

Interruptibility

As long as the source area is not part of a data block that only exists in the load memory, there is no limit to the nesting depth.

If, however, BLKMOV is interrupted while copying from a DB that is not runtime-relevant, the execution of BLKMOV can no longer be nested.

Memory areas

The instruction "Move block" moves the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output
- Data blocks not runtime-relevant

General rules for moving

The source and destination areas must not overlap. If the source and destination areas have different lengths, only the length of the smaller area will be moved.

If the source area is smaller than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is smaller than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If an area of the BOOL data type is copied, the specified length of the area must be divisible by 8, otherwise the instruction will not execute.

Note

If the actual source or destination area is smaller than a destination or source area specified by the parameter SRCBLK or DSTBLK, the response depends on the CPU type:

- The following applies to S7-300 CPUs:
  No data are transferred. Error code W#16#837F is output at the parameter RET_VAL. This error code has to be evaluated in your user program.
- The following applies to the S7-400 V4 CPUs (V4, V4 H-CPU and V4.5. H-CPU):
  No data are transferred. Error code W#16#8122 or W#16#8323 is output at the parameter RET_VAL. This error code has to be evaluated in your user program.
- For all other S7-400-CPUs:
  Data are transferred. Error code W#16#8122 or W#16#8323 is output at the parameter RET_VAL. This error code has to be evaluated in your user program.

Rules for moving character strings

You can use the "Move block" instruction to also move source and destination areas of the STRING data type. If only the source area is data type STRING, the characters that are actually contained in the character string are moved. Information on the actual and maximum length is also written to the destination area. If the source and destination area are both STRING data type, the current length of the character string in the destination area is set to the number of characters actually moved.

If you want to move the information on the maximum and actual length of a character string, specify the areas in bytes in the SRCBLK and DSTBLK parameters.

Rules for moving data blocks that are not runtime-relevant

The source area can also be in a data block in load memory that is not runtime-relevant. Data blocks that are not runtime-relevant are indicated by the key word UNLINKED.

If a non-runtime-relevant data block is copied to the work memory with the "Move block" instruction and loaded at the same time, for example, by the programming device, the execution of the instruction can be delayed for several milliseconds. This results in a longer OB cycle and may trip the cycle monitoring.

If a non-runtime-relevant data block is moved with the "Move block" instruction and the move operation is interrupted, the execution of the instruction can no longer be continued.

If the instruction Read from data block in the load memory is available on your CPU, you must use this instruction to read data blocks that are not runtime-relevant from load memory. If you use the "Move block" instruction, error W#16#8092 is output.

Syntax

The following syntax is used for the "Move block" instruction:

SCL

```
BLKMOV(SRCBLK := <Operand>,
       DSTBLK=> <Operand>)
```

Parameter

The following table shows the parameters of the "Move block" instruction:

Parameter	Declaration	Data type	Description
SRCBLK	Input	ANY	Specifies the memory area to be moved (source area).
DSTBLK	Output	ANY	Specifies the memory area to which the data will be moved (destination area).
Function value (RET_VAL)		INT	Error information

For more information on valid data types, refer to "See also".

Parameter RET_VAL

The following table shows the meaning of the values of the parameter RET_VAL:

Error code* (W#16#...)	Explanation
0000	No error
8092	The source or target area is only available in the load memory.
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For more information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Evaluating errors with output parameter RET_VAL (Page 5370)

Switching display formats in the program status (Page 2980)

UBLKMOV: Move block uninterruptible

Description

The instruction "Move block uninterruptible" copies the contents of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use ANY pointer to define the source and destination area.

The copy operation cannot be interrupted by other operating system activities. As a result the alarm reaction time of the CPU can increase during the execution of the instruction "Move block uninterruptible".

Note

In this instruction, you can only use data types which are valid on a CPU of the S7-300/400 series.

Memory areas

The instruction "Move block uninterruptible" copies the following memory areas:

- Areas of a data block
- Bit memory

- Process image input
- Process image output

General rules when copying

The source and destination area must not overlap during the execution of the instruction "Move block uninterruptible". If the source area is smaller than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is smaller than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a source or destination area defined as a formal parameter is less than a destination or source area specified on the SRCBLK or DSTBLK parameter, No data is transferred.

If an area of the BOOL data type is copied, the specified length of the area must be divisible by 8, otherwise the instruction will not execute.

The instruction "Move block uninterruptible" copies a maximum of 512 bytes. Note the CPU specific restrictions for this.

Rules for moving character strings

The instruction "Move block uninterruptible" copies source and destination areas of the data type STRING. If only the source area is data type STRING, the characters that are actually contained in the character string are moved. Information on the actual and maximum length are not written in the destination area. If the source and destination area are both of the data type STRING, the current length of the character string in the destination area is set to the number of characters actually moved. If areas of the data type STRING are moved, you must specify "1" as the area length.

Note

If the actual source or destination area is smaller than a destination or source area specified by the parameter BVAL or BLK, the response depends on the CPU type:

- The following applies to S7-300 CPUs:
  No data is transferred. The error code W#16#837F is issued on the parameter RET_VAL. This error code has to be evaluated in your user program.
- The following applies to the S7-400 V4 CPUs (V4, V4 H-CPU and V4.5. H-CPU):
  No data is transferred. The error code W#16#8122 or W#16#8323 is issued on the parameter RET_VAL. This error code has to be evaluated in your user program.
- For all other S7-400-CPUs:
  Data is transferred. The error code W#16#8122 or W#16#8323 is issued on the parameter RET_VAL. This error code has to be evaluated in your user program.

Rules for moving character strings

The instruction "Move block uninterruptible" copies source and destination areas of the data type STRING. If only the source area is data type STRING, the characters that are actually

contained in the character string are moved. Information on the actual and maximum length is not written to the destination area. If the source and destination area are both STRING data type, the current length of the character string in the destination area is set to the number of characters actually moved. If areas of the STRING data type are moved, specify "1" as the area length.

Syntax

The following syntax is used for the instruction "Move block uninterruptible":

```
SCL
UBLKMOV(SRCBLK := <Operand>,
        DSTBLK=> <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
SRCBLK	Input	ANY	Specifies the memory area to be moved (source area).
DSTBLK	Output	ANY	Specifies the memory area to which the data will be moved (destination area).
Function value (RET_VAL)		INT	Error information

For more information on valid data types, refer to "See also".

Parameter RET_VAL

The following table shows the meaning of the values of the parameter RET_VAL:

Error code* (W#16#...)	Explanation
0000	No error
8091	The source area is in a data block that is not relevant for program execution.
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For more information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Evaluating errors with output parameter RET_VAL (Page 5370)

Switching display formats in the program status (Page 2980)

FILL: Fill block

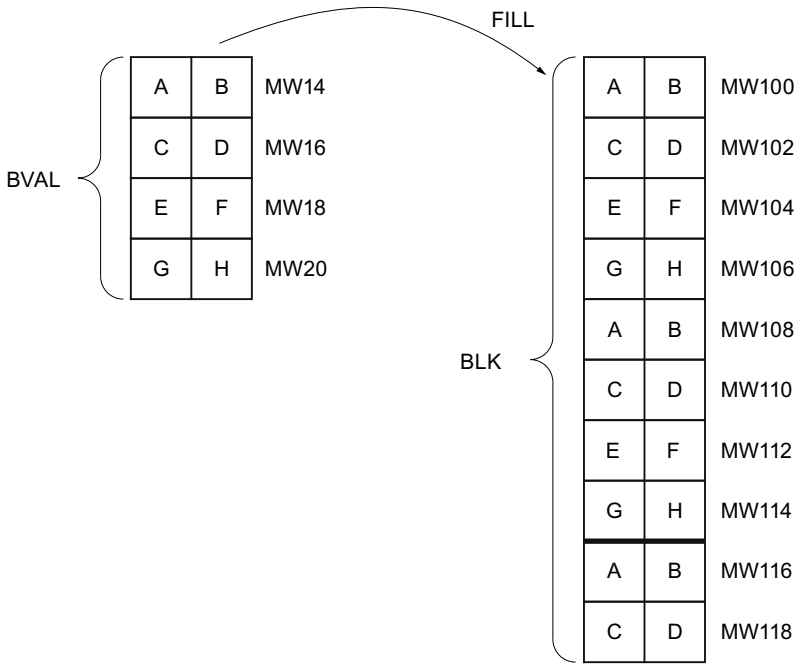
Description

The instruction "Fill block" fills a memory area (destination area) with the content of another memory area (source area). The instruction "Fill block" copies the content of the source area to the destination area until the destination area is completely written. The move operation takes place in the direction of ascending addresses. You use ANY pointer to define the source and destination area.

Note

In this instruction, you can only use data types which are valid on a CPU of the S7-300/400 series.

The following figure shows the principle of the move operation:



Example: The contents of the range MW100 to MW118 are to be preassigned with the contents of the memory words MW14 to MW20.

Consistency of source and destination data

Note that while the instruction "Fill block" is being executed, the source data remains unchanged; otherwise the consistency of the destination data cannot be guaranteed.

Memory areas

The instruction "Fill block" copies the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output
- Data blocks not relevant for program execution

The instruction "Fill block" copies the following memory areas:

- Instructions, system data blocks
- Counters
- Timers
- Memory areas of the I/O area

General rules when copying

The source and destination areas must not overlap. If the destination area to be preset is not an integer multiple of the length of the BVAL input parameter, the destination area is nevertheless written up to the last byte.

If the destination area to be preset is smaller than the source area, the function only copies as much data as can be written to the destination area.

If the actually present destination or source area is smaller than the assigned memory area for the source or destination area (BVAL, BLK parameters), No data is transferred.

If the ANY pointer (source or destination) is the BOOL type, the length specified must be divisible by 8; otherwise the instruction will not execute.

If the destination area is STRING data type, the instruction writes the entire string including the administration information.

Note

If the actual source or destination area is smaller than a destination or source area specified by the parameter BVAL or BLK, the response depends on the CPU type:

- The following applies to S7-300 CPUs:
  No data is transferred. The error code W#16#837F is issued on the parameter RET_VAL. This error code has to be evaluated in your user program.
- The following applies to the S7-400 V4 CPUs (V4, V4 H-CPU and V4.5. H-CPU):
  No data is transferred. The error code W#16#8122 or W#16#8323 is issued on the parameter RET_VAL. This error code has to be evaluated in your user program.
- For all other S7-400-CPUs:
  Data is transferred. The error code W#16#8122 or W#16#8323 is issued on the parameter RET_VAL. This error code has to be evaluated in your user program.

Syntax

The following syntax is used for the instruction "Fill block":

```
SCL
FILL(BVAL := <Operand>,
     BLK=> <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
BVAL	Input	ANY	Specification of the memory area (source area) with whose content the destination area on the BLK parameter will be filled.
BLK	Output	ANY	Specification of the memory area that will be filled with the content of the source area.
Function value (RET_VAL)		INT	Error information

For more information on valid data types, refer to "See also".

Parameter BVAL

When you transfer a structure as an input parameter, remember that the length of a structure is always based on an even number of bytes. The structure will need one byte of additional memory space if you declare a structure with an odd number of bytes.

Parameter RET_VAL

The following table shows the meaning of the values of the parameter RET_VAL:

Error code* (W#16#...)	Explanation
0000	No error
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For more information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Evaluating errors with output parameter RET_VAL (Page 5370)

Switching display formats in the program status (Page 2980)

Conversion operations

CONVERT: Convert value

Description

Use the "Convert value" instruction to program explicit conversions. You determine the data types to be converted in a dialog box with opens automatically when you insert the instruction. When executed, the instruction reads the source value and converts it into the specified target value.

For information on possible conversions, refer to the "Explicit conversion" section at "See also".

Syntax

Use the following syntax for the "Convert value" instruction:

SCL

```
<Target_value> := <Conversion_function>(<Source_value>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
<Source_value>	Input, constant	Bit strings, integers, floating-point numbers, times, date and time, character strings	Value to be converted.
<Conversion_function>	-	-	Function that specifies the data type to be converted .
<Target_value>	Output	Bit strings, integers, floating-point numbers, times, date and time, character strings	Result of the conversion

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_INT" := REAL_TO_INT("Tag_REAL");
```

The following table shows how the instruction works using specific operand values:

Operand	Data type	Value
Tag_REAL	REAL	20.56
Tag_INT	INT	21

During the conversion, the value of the "Tag_REAL" operand will be rounded to the nearest integer and saved in the "Tag_INT" operand.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Explicit conversion (Page 2430)

ROUND: Round numerical value

Description

The "Round numerical value" instruction is used to round the value at input IN to the nearest integer. The instruction interprets the value at input IN as a floating-point number and converts it into an integer or floating-point number. If the input value is exactly between an even and odd number, the even number is selected.

Syntax

The following syntax is used for the "Round numerical value" instruction:

SCL

```
ROUND(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
<Expression>	Input	REAL	Input value to be rounded.
Function value		DINT	Result of the rounding

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := ROUND("Tag_Value");
```

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_Value	1.50000000	-1.50000000
Tag_Result	2	-2

The result of the instruction is returned in the operand "Tag_Result" as a function value.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

CEIL: Generate next higher integer from floating-point number

Description

Use the "Generate next higher integer from floating-point number" instruction to round the value of a floating point number to the next higher integer. The instruction interprets the input value as floating-point number and converts it to the next higher integer. The function value can be greater than or equal to the input value.

Syntax

The following syntax is used for the "Generate next higher integer from floating-point number" instruction:

```
SCL
CEIL(<expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
<expression>	Input	REAL	Input value
Function value		DINT	Input value rounded up

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := CEIL("Tag_Value");
```

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_Value	0.5	-0.5
Tag_Result	1	0

The result of the instruction is returned in the operand "Tag_Result" as a function value.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

FLOOR: Generate next lower integer from floating-point number

Description

Use the "Generate next lower integer from floating-point number" instruction to round the value of a floating point number to the next lower integer. The instruction interprets the input value as floating-point number and converts it to the next lower integer. The function value can be equal or less than the input value.

Syntax

Use the following syntax for the "Generate next lower integer from floating-point number" instruction:

```
SCL
FLOOR(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
<Expression>	Input	REAL	Input value
Function value		DINT	Input value rounded

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := FLOOR("Tag_Value");
```

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_Value	0.5	-0.5
Tag_Result	0	-1

The result of the instruction is returned in the operand "Tag_Result" as a function value.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

TRUNC: Truncate numerical value

Description

The "Truncate numerical value" instruction is used to generate an integer from the input value without rounding. The instruction selects only the integer part of the input value and returns this part without decimal places as the function value.

Syntax

The following syntax is used for the "Truncate numerical value" instruction:

```
SCL
TRUNC(<Expression>)
TRUNC_<Data type>(<Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
<Expression>	Input	REAL	Input value
_<Data type>		Integers default: DINT	Data type of the function value: • You do not need to specify the data type if using the default. • Any other valid data type you may use must be declared explicitly.
Function value		Integers	Integer component of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result1" := TRUNC("Tag_Value1");
"Tag_Result2" := TRUNC("Tag_Value2"+"Tag_Value3");
"Tag_Result3" := TRUNC_REAL("Tag_Value4");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_Value1	-1.5
Tag_Result1	-1
Tag_Value2	2.1
Tag_Value3	3.2
Tag_Result2	5
Tag_Result3	2.0
Tag_Value4	2.4

See also

Operators and operator precedence (Page 2687)

Expressions (Page 2681)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Overview of the valid data types (Page 2236)

SCALE: Scale

Description

The instruction "Scale" converts the integer on the parameter IN into a floating-point number, which can be scaled in physical units between a low and a high limit. You use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is scaled. The result of the instruction is output on the OUT parameter.

The instruction "Scale" works with the following equation:

OUT = [((FLOAT (IN) – K1)/(K2–K1)) * (HI_LIM–LO_LIM)] + LO_LIM

The values of the "K1" and "K2" constants are determined by the signal state on the BIPOLAR parameter. The following signal states are possible on the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case the "K1" constant has the value "-27648.0" and the "K2" constant the value "+27648.0".
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case the "K1" constant has the value "0.0" and the "K2" constant the value "+27648.0".

When the value at the IN parameter is greater than the value of the "K2" constant, the result of the instruction is set to the value of the high limit (HI_LIM) and an error is output.

When the value at the IN parameter is less than the value of the "K1" constant, the result of the instruction is set to the value of the low limit value (LO_LIM) and an error is output.

When the indicated low limit is greater than the high limit (LO_LIM >HI_LIM), the result is scaled in reverse proportion to the input value.

Syntax

The following syntax is used for the "Scale" instruction:

SCL

```
SCALE(IN := <Expression>,
      HI_LIM := <Operand>,
      LO_LIM := <Operand>,
      BIPOLAR := <Operand>,
      OUT=> <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
IN	Input	INT	Input value to be scaled.
HI_LIM	Input	REAL	High limit
LO_LIM	Input	REAL	Low limit

Parameter	Declaration	Data type	Description
BIPOLAR	Input	BOOL	Indicates if the value at the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
OUT	Output	REAL	Result of the instruction
Function value (RET_VAL)		WORD	Error information

For additional information on valid data types, refer to "See also".

Parameter RET_VAL

The following table shows the meaning of the values of the parameter RET_VAL:

Error code* (W#16#...)	Explanation
0000	No error
0008	The value of the parameter IN is greater than 27 648 or is less than 0 (unipolar) or -27 648 (bipolar).
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

```
"Tag_ErrorCode" := SCALE(IN := "Tag_InputValue",
                         HI_LIM := "Tag_HighLimit"
                         LO_LIM := "Tag_LowLimit"
                         BIPOLAR := "Tag_Bipolar",
                         OUT => "Tag_Result");
```

The error information of the instruction is returned in the operand "Tag_ErrorCode" as a function value.

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_InputValue	22
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1

Parameter	Operand	Value
OUT	Tag_Result	50.03978588
RET_VAL	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

Evaluating errors with output parameter RET_VAL (Page 5370)

Switching display formats in the program status (Page 2980)

UNSCALE: Unscale

Description

The instruction "Unscale" unscales the floating-point number on the IN parameter in physical units between a low and a high limit and converts them into an integer. You use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is unscaled. The result of the instruction is output on the OUT parameter.

The instruction "Unscale" works with the following equation:

OUT = [((IN–LO_LIM)/(HI_LIM–LO_LIM)) * (K2–K1)] + K1

The values of the "K1" and "K2" constants are determined by the signal state on the BIPOLAR parameter. The following signal states are possible on the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case the "K1" constant has the value "-27648.0" and the "K2" constant the value "+27648.0".
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case the "K1" constant has the value "0.0" and the "K2" constant the value "+27648.0".

When the value at the IN parameter is greater than the value of the "HI_LIM" constant, the result of the instruction is set to the value of the constant (K2) and an error is output.

When the value at the IN parameter is less than the value of the constant of the low limit (LO_LIM), the result of the instruction is set to the value of the constant (K1) and an error is output.

Syntax

The following syntax is used for the instruction "Unscale":

```
SCL
UNSCALE(IN := <Expression>,
        HI_LIM := <Operand>,
        LO_LIM := <Operand>,
        BIPOLAR := <Operand>,
        OUT=> <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
IN	Input	REAL	Input value to be unscaled to an integer value.
HI_LIM	Input	REAL	High limit
LO_LIM	Input	REAL	Low limit
BIPOLAR	Input	BOOL	Indicates if the value at the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
OUT	Output	INT	Result of the instruction
Function value (RET_VAL)		WORD	Error information

For additional information on valid data types, refer to "See also".

Parameter RET_VAL

The following table shows the meaning of the values of the parameter RET_VAL:

Error code* (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the high limit (HI_LIM) or less than the value of the low limit (LO_LIM).
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

```
SCL
"Tag_ErrorCode" := UNSCALE(IN := "Tag_InputValue",
                           HI_LIM := "Tag_HighLimit"
                           LO_LIM := "Tag_LowLimit"
                           BIPOLAR := "Tag_Bipolar",
                           OUT => "Tag_Result");
```

The error information of the instruction is returned in the operand "Tag_ErrorCode" as a function value.

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
OUT	Tag_Result	22
RET_VAL	Tag_ErrorCode	W#16#0000

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

Evaluating errors with output parameter RET_VAL (Page 5370)

Switching display formats in the program status (Page 2980)

Program control operations

IF: Run conditionally

Description

The instruction "Run conditionally" branches the program flow depending on a condition. The condition is an expression with Boolean value (TRUE or FALSE). Logical expression or comparative expressions can be stated as conditions.

When the instruction is executed, the stated expressions are evaluated. If the value of an expression is TRUE, the condition is fulfilled; if the value is FALSE, it is not fulfilled.

Syntax

Depending on the type of branch, you can program the following forms of the instruction:

- Branch through IF:

SCL

```
IF <Condition> THEN <Instructions>
END_IF;
```

If the condition is satisfied, the instructions programmed after the THEN are executed. If the condition is not satisfied, the execution of the program continues with the next instruction after the END_IF.

- Branch through IF and ELSE:

SCL

```
IF <Condition> THEN <Instructions1>
ELSE <Instructions0>;
END_IF;
```

If the condition is satisfied, the instructions programmed after the THEN are executed. If the condition is not satisfied, the instructions programmed after the ELSE are executed. Then the execution of the program continues with the next instruction after the END_IF.

- Branch through IF, ELSIF and ELSE:

SCL

```
IF <Condition1> THEN <Instructions1>
ELSIF <Condition2> THEN <Instruction2>
ELSE <Instructions0>;
END_IF;
```

If the first condition (<Condition1>) is satisfied, the instructions (<Instructions1>) after the THEN are executed. After execution of the instructions, the execution of the program continues after the END_IF.

If the first condition is not satisfied, the second condition (<Condition2>) is checked. If the second condition (<Condition2>) is fulfilled, the instructions (<Instructions2>) after the THEN are executed. After execution of the instructions, the execution of the program continues after the END_IF.

If none of the conditions are fulfilled, the instructions (<Instructions0>) after ELSE are executed followed by the execution of the program after END_IF.

You can nest as many combinations of ELSIF and THEN as you like within the IF instruction. The programming of an ELSE branch is optional.

The syntax of the IF instruction consists of the following parts:

Parameter	Data type	Description
<Condition>	BOOL	Expression to be evaluated
<Instructions>	-	Instructions to be executed with satisfied condition. An exception are instructions programmed after the ELSE. These are executed if no condition within the program loop is satisfied.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
IF "Tag_1" = 1
THEN "Tag_Value" := 10;
ELSIF "Tag_2" = 1
THEN "Tag_Value" := 20;
ELSIF "Tag_3" = 1
THEN "Tag_Value" := 30;
ELSE "Tag_Value" := 0;
END_IF;
```

The following table shows how the instruction works using specific operand values:

Operand	Value			
Tag_1	1	0	0	0
Tag_2	0	1	0	0
Tag_3	0	0	1	0
Tag_Value	10	20	30	0

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

CASE: Create multiway branch

Description

The instruction "Create multiway branch" executes one of several instruction sequences depending on the value of a numerical expression.

The value of the expression must be an integer. When the instruction is executed, the value of the expression is compared with the values of several constants. If the value of the expression agrees with the value of a constant, the instructions programmed directly after this constant are executed. The constants can assume the following values:

- An integer (for example, 5)
- A range of integers (for example, 15..20)
- An enumeration consisting of integers and ranges (for example, 10, 11, 15..20)

Syntax

The following syntax is used for the "Create multiway branch" instruction:

SCL

```
CASE <expression> OF
<Constant1>: <Instructions1>
<Constant2>: <Instructions2>
<ConstantX>: <InstructionsX>; // X >=3
ELSE <Instructions0>;
END_CASE;
```

The syntax of the instruction consists of the following parts:

Parameter	Data type	Description
<expression>	Integers	Value which is compared to the programmed constant values.
<Constant>	Integers	Constant values which form the condition for the execution of an instruction sequence. The constants can assume the following values: • An integer (for example, 5) • A range of integers (for example, 15..20) • An enumeration consisting of integers and ranges (for example, 10, 11, 15..20)
<Instruction>	-	Any instructions which are executed if the value of the expression agrees with the value of a constant. An exception are instructions programmed after the ELSE. These instructions are executed if the values do not agree.

For additional information on valid data types, refer to "See also".

If the value of the expression agrees with the value of the first constant (<Constant1>), the instructions (<Instructions1>) which are programmed directly after the first constant are executed. Program execution subsequently resumes after the END_CASE.

If the value of the expression does not agree with the value of the first constant (<Constant1>), this value is compared to the value of the constant which is programmed next. In this way, the CASE instruction is executed until the values agree. If the value of the expression does not correspond to any of the programmed constant values, the instructions (<Instructions0>) which are programmed after the ELSE are executed. ELSE is an optional part of the syntax and can be omitted.

The CASE instruction can also be nested by replacing an instruction block with CASE. END_CASE represents the end of the CASE instruction.

Example

The following example shows how the instruction works:

```
SCL
CASE "Tag_Value" OF
     0 :
     "Tag_1" := 1;
     1,3,5 :
     "Tag_2" :=1;
     6..10 :
     "Tag_3" := 1;
     16,17,20..25 :
     "Tag_4" := 1;
ELSE "Tag_5" := 1;
END_CASE;
```

The following table shows how the instruction works using specific operand values:

Operand	Values				
Tag_Value	0	1, 3 , 5	6, 7, 8, 9, 10	16,17, 20, 21, 22, 23, 24, 25	2
Tag_1	1	-	-	-	-
Tag_2	-	1	-	-	-
Tag_3	-	-	1	-	-
Tag_4	-	-	-	1	-
Tag_5	-	-	-	-	1
1: The operand is set to the signal state "1". -: The signal state of the operand remains unaltered.					

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

EXIT: Exit loop immediately (Page 6200)

CONTINUE: Recheck loop condition (Page 6199)

FOR: Run in counting loop

Description

The instruction "Run in counting loop" causes repeated execution of a program loop until a loop variable lies within a specified value range.

Program loops can also be nested. Within a program loop, you can program additional program loops with other loop variables.

The current continuous run of a program loop can be ended by the instruction "Recheck loop condition" (CONTINUE). The instruction "Exit loop immediately" (EXIT) ends the entire loop execution. For additional information on this topic, refer to "See also".

Limit values for FOR statements

To program "safe" FOR statements that do not run endlessly, observe the following rules and limit values:

Rule

```
FOR ii := Start TO End BY Step DO
```

If ...	... then	Note
start < end	end < (PMAX step)	Run tag ii runs in a positive direction
start > end AND step < O	end > (NMAX step)	Run tag ii runs in a negative direction

Limit values

Different limit values apply to the possible data types:

Data type	PMAX	NMAX
ii of type INT	32767	-32768
ii of type DINT	2147483647	-2147483648

Syntax

The following syntax is used for the "Run in counting loop" instruction:

SCL

```
FOR<Run_tag> := <Start_value> TO<End_value> BY<Increment> DO<Instructions>
END_FOR;
```

The syntax of the FOR instruction consists of the following parts:

Parameter	Data type	Description
<Run tag>	INT, DINT	Operand whose value is evaluated with the loop execution. The data type of the loop variable determines the data type of the other parameters.
<Start value>	INT, DINT	Expression whose value is allocated at the start of the loop execution of the loop variables.

Parameter	Data type	Description
<End value>	INT, DINT	Expression whose value defines the last run of the program loop. The value of the loop variable is checked before each loop: • End value not reached: The instructions according to DO are executed • End value is reached: The FOR loop is executed one last time • End value exceeded: The FOR loop is completed An alteration to the end value is not permitted during execution of the instruction.
<Increment>	INT, DINT	Expression by whose value the loop variable is increased (positive increment) or decreased (negative increment) after each loop. Specification of the increment is optional. If no increment is given, the value of the loop variable is increased by 1 after each loop. An alteration of the increment is not permitted during execution of the instruction.
<Instructions>	-	Instructions which are carried out with each loop, as long as the value of the loop variable lies within the value range. The value range is defined by the start and end values.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
FOR i := 2 TO 8 BY 2
DO "a_array[i] := "Tag_Value"*"b_array[i]";
END_FOR;
```

The operand "Tag_Value" is multiplied with the elements (2, 4, 6, 8) of the ARRAY tag "b_array". The result is read in to the elements (2, 4, 6, 8) of the ARRAY tag "a_array".

See also

CONTINUE: Recheck loop condition (Page 4267)

EXIT: Exit loop immediately (Page 4268)

Overview of the valid data types (Page 2236)

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

WHILE: Run if condition is met

Description

The instruction "Run if condition is met" causes a program loop to be repeatedly executed until the implementation condition is satisfied. The condition is an expression with Boolean value (TRUE or FALSE). Logical expression or comparative expressions can be stated as conditions.

When the instruction is executed, the stated expressions are evaluated. If the value of an expression is TRUE, the condition is fulfilled; if the value is FALSE, it is not fulfilled.

Program loops can also be nested. Within a program loop, you can program additional program loops with other loop variables.

The current continuous run of a program loop can be ended by the instruction "Recheck loop condition" (CONTINUE). The instruction "Exit loop immediately" (EXIT) ends the entire loop execution. For additional information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Run if condition is met" instruction:

SCL

```
WHILE <Condition> DO <Instructions>
END_WHILE;
```

The syntax of the WHILE instruction consists of the following parts:

Parameter	Data type	Description
<Condition>	BOOL	Expression which is evaluated before each loop.
<Instructions>	-	Instructions to be executed with satisfied condition. If the condition has not been satisfied, program execution continues after END_WHILE.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
WHILE
      "Tag_Value1" <> "Tag_Value2"
      DO "Tag_Result"
      := "Tag_Input";
END_WHILE;
```

As long as the values of the operands "Tag_Value1" and "Tag_Value2" do not match, the value of the operand "Tag_Input" is allocated to the operand "Tag_Result".

See also

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Overview of the valid data types (Page 2236)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

CONTINUE: Recheck loop condition (Page 6199)

EXIT: Exit loop immediately (Page 6200)

REPEAT: Run if condition is not met

Description

The instruction "Run if condition is not met" causes a program loop to be repeatedly executed until a termination condition is met. The condition is an expression with Boolean value (TRUE or FALSE). Logical expression or comparative expressions can be stated as conditions.

When the instruction is executed, the stated expressions are evaluated. If the value of an expression is TRUE, the condition is fulfilled; if the value is FALSE, it is not fulfilled.

The instructions are executed once, even if the termination condition is fulfilled.

Program loops can also be nested. Within a program loop, you can program additional program loops with other loop variables.

The current continuous run of a program loop can be ended by the instruction "Recheck loop condition" (CONTINUE). The instruction "Exit loop immediately" (EXIT) ends the entire loop execution. For additional information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Run if condition is not met" instruction:

SCL

```
REPEAT <Instructions>
UNTIL <Condition>END_REPEAT;
```

The syntax of the REPEAT instruction consists of the following parts:

Parameter	Data type	Description
<Instructions>	-	Instructions that are executed as long as the programmed condition has the value FALSE. The instructions are executed once, even if the termination condition is fulfilled.
<Condition>	BOOL	Expression which is evaluated after each loop. If the expression has the value FALSE, the program loop is executed once again. If the expression has the value TRUE, the program loop continues after END_REPEAT.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
REPEAT "Tag_Result"
        := "Tag_Value";
UNTIL "Tag_Error"
END_REPEAT;
```

As long as the value of the operand "Tag_Error" has the signal state "0", the value of the operand "Tag_Value" is allocated to the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

CONTINUE: Recheck loop condition

Description

The "Recheck loop condition" instruction ends the current program run of a FOR, WHILE or REPEAT loop.

After execution of the instruction, the conditions for the continuation of the program loop are evaluated again. The instruction affects the program loop which directly contains the instruction.

Syntax

The following syntax is used for the "Recheck loop condition" instruction:

```
SCL
CONTINUE;
```

Example

The following example shows how the instruction works:

```
SCL
FOR i
```

```
SCL
     := 1 TO 15 BY 2 DO
     IF (i < 5) THEN
     CONTINUE;
END_IF;
      "DB10".Test[i] := 1;
END_FOR;
```

For additional information on valid data types, refer to "See also".

If the condition i < 5 is satisfied, then the subsequent value allocation ("DB10".Test[i] := 1) will not be executed. The run variable (i) is increased by the increment of "2" and checked to see whether its current value lies in the programmed value range. If the run variable lies in the value range, the IF condition is evaluated again.

If the condition i < 5 is not satisfied, then the subsequent value allocation ("DB10".Test[i] := 1) will be executed and a new loop started. In this case, the run variable is also increased by the increment "2" and checked.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

EXIT: Exit loop immediately (Page 6200)

EXIT: Exit loop immediately

Description

The instruction "Exit loop immediately" cancels the execution of a FOR, WHILE or REPEAT loop at any point regardless of conditions. The execution of the program is continued after the end of the loop (END_FOR, END_WHILE, END_REPEAT).

The instruction affects the program loop which directly contains the instruction.

Syntax

The following syntax is used for the "Exit loop immediately" instruction:

```
SCL
EXIT;
```

Example

The following example shows how the instruction works:

```
SCL
FOR i := 15 TO 1 BY -2 DO
IF (i < 5)
THEN EXIT;
END_IF;
"DB10".Test[i] := 1;
END_FOR;
```

For additional information on valid data types, refer to "See also".

If the condition i < 5 is satisfied, then the execution of the loop will be cancelled. Program execution resumes after the END_FOR.

If the condition i < 5 is not satisfied, then the subsequent value allocation ("DB10".Test[i] := 1) will be executed and a new loop started. The run tag (i) is decreased by the increment of "-2" and it is checked whether its current value lies in the programmed value range. If the (i) run variable lies within the value range, the IF condition is evaluated again.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

CONTINUE: Recheck loop condition (Page 6199)

GOTO: Jump

Description

Use the instruction "Jump" to resume the execution of a program at a given point marked with a jump label.

The jump labels and the instruction "Jump" must be in the same block. The name of a jump label can only be assigned once within a block. Each jump label can be the target of several jump instructions.

A jump from the "outside" into a program loop is not permitted, but a jump from a loop to the "outside" is possible.

Syntax

Use the following syntax for the "Jump" instruction:

```
SCL
GOTO <Jump label>
```

```
SCL
...
<Jump label>: <Instructions>
```

The syntax of the GOTO instruction consists of the following parts:

Parameter	Data type	Description
<jump label>	-	Jump label to be jumped to
<Instructions>	-	Instructions which are executed after the jump.

Example

The following example shows how the instruction works:

```
SCL
CASE "Tag_Value" OF
1 : GOTO MyLABEL1;
2 : GOTO MyLABEL2;
3 : GOTO MyLABEL3;
ELSE GOTO MyLABEL4;
END_CASE;
MyLABEL1: "Tag_1" := 1;
MyLABEL2: "Tag_2" := 1;
MyLABEL3: "Tag_3" := 1;
MyLABEL4: "Tag_4" := 1;
```

Depending on the value of the "Tag_Value" operand, the execution of the program will resume at the point identified by the corresponding jump label. If the operand "Tag_Value" has the value 2, for example, program execution will resume at the jump label "MyLABEL2". The program line identified by the jump label "MyLABEL1" will be skipped in this case.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

RETURN: Exit block

Description

The instruction "Exit block" exits the program execution in the currently edited block and continues in the calling block.

The instruction can be omitted at the end of the block.

Syntax

The following syntax is used for the "Exit block" instruction:

```
SCL
RETURN;
```

Example

The following example shows how the instruction works:

```
SCL
IF "Tag_Error" <>0 THEN RETURN;
END_IF;
```

If the signal state of the "Tag_Error" operand is not zero, execution of the program ends in the block currently being processed.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

(*...*): Inserting a comment section

Description

You can use the "Insert comment section" instruction to add a comment section. The text within the parenthesis "(*...*)" is treated as a comment.

Syntax

The following syntax is used for the "Insert comment section" instruction:

```
SCL
(*...*)
```

Example

The following example shows how the instruction works:

```
SCL
(*This is a comment section.*)
```

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Runtime control

COMPRESS: Compress CPU memory

Description

The instruction "Compress CPU memory" compresses the working memory of the CPU or the RAM load memory. During this compression the gaps caused by deleting or loading blocks are filled by pushing together the existing blocks.

The compression operation is started by calling the instruction "Compress CPU memory". The compression operation is divided into several program cycles. The compression operation is monitored by the values of the individual parameters. If value "1" is on the BUSY parameter of the instruction, the compression operation is still being executed. The value "1" on the DONE parameter reports that the compression operation has been completed. Error information can be obtained from the RET_VAL parameter.

If an externally started compression operation has already been executed, calling the instruction "Compress CPU memory" will cause an error to be displayed.

Syntax

The following syntax is used for the instruction "Compress CPU memory":

SCL

```
COMPRESS(BUSY => <Operand>, DONE => <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
BUSY	Output	BOOL	Information whether compression operation is active.
DONE	Output	BOOL	Information whether compression operation was completed.
Function value (RET_VAL)		INT	Error information

For additional information on valid data types, refer to "See also".

Parameter RET_VAL

The following table shows the meaning of the values of the parameter RET_VAL:

Error code* (W#16#...)	Explanation
0000	No error The status of the compression operation can be obtained from the BUSY and DONE parameters.
8091	An externally started compression operation is already being executed.
8092	The operation cannot be executed, because one of the following conditions applies: • The "Delete block" function is active. • A programming device function is accessing one of the blocks to be moved. • An externally started compression operation is already being executed. • The H-CPU is running a link-up or update function.
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Evaluating errors with output parameter RET_VAL (Page 5370)

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Switching display formats in the program status (Page 2980)

RE_TRIGR: Restart cycle monitoring time

Description

The instruction "Restart cycle monitoring time" restarts the cycle monitoring of the CPU. The cycle time monitoring then restarts with the time you have set in the CPU configuration. By restarting the cycle monitoring time, you can prevent errors being triggered or the CPU changing to STOP.

The "Restart cycle monitoring time" instruction can be used in blocks of priority class 1 (the cyclic OB) and in the blocks called within it.

If the instruction is called in a block with a higher priority, such as a hardware interrupt, diagnostic interrupt, or cyclic interrupt, the instruction is not executed and the ENO enable output is set to signal state "0".

Note

Make sure that you do not create an infinite loop in the cyclical program processing, i.e. in OB1, when you use the instruction "Restart cycle monitoring time". Otherwise the CPU will not reach the cycle control point. As a result, it may not be possible to execute certain CPU functions (e.g. process image update).

Note

If the instruction "Restart cycle monitoring time" is started more than 30 times within a LOOP instruction, the CPU switches to STOP because of a runtime error.

Syntax

The following syntax is used for the "Restart cycle monitoring time" instruction:

SCL

```
RE_TRIGR()
```

See also

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

WAIT: Configure time delay

Description

The instruction "Configure time delay" pauses the program execution for a specific period of time. You indicate the period of time in microseconds on the WT parameter of the instruction.

You can configure time delays of up to 32 767 microseconds (µs). The shortest possible time delay depends on the respective CPU and corresponds to the execution time of the instruction "Configure time delay".

The execution of the instruction can be interrupted by higher priority events.

The instruction "Configure time delay" supplies no error information.

Note

For the S7-300 CPUs (except S7-318), the delay time programmed by the instruction is a minimum time. It is extended by the execution time of the nested priority classes and by system utilizations.

Syntax

The following syntax is used for the instruction "Configure time delay":

```
SCL
WAIT(WT := <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
WT	Input	INT	Time delay in microseconds (µs)

See also

Evaluating errors with output parameter RET_VAL (Page 5370)

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

STP: Exit program

Description

The "Exit program" instruction is used to set the CPU to STOP mode and therefore to terminate the execution of the program. The effects of changing from RUN to STOP depend on the CPU configuration.

Syntax

The following syntax is used for the instruction "Exit program":

```
SCL
STP()
```

See also

Evaluating errors with output parameter RET_VAL (Page 5370)

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

PROTECT: Change protection level

Description

The instruction "Change protection level" is used to change the configured protection level "1" of the CPU.

The configured protection level "1" with selected option "Removable with password", means that you have to configure a password. This password enables you to write-access during operation the CPU which was read-only protected with the instruction "Change Protection Level" (e. g. Load).

You specify the protection level on the MODE parameter. The current protection level of your CPU is displayed in the inspector window under "Properties > Protection".

The following table shows the meaning of the protection levels.

Protection level	Meaning
1	All programming device functions are permitted. Any existing blocking of the password authentication is canceled by the instruction "Change protection level".
2	• The program and the configuration of the CPU cannot be changed. The program in the CPU can be read. • The functions for process control, process monitoring, and process communication are permitted. • All informational functions are permitted. • If you know the valid password you can cancel the set write protection. • Any existing blocking of the password authentication is canceled by the instruction "Change protection level".
3	• You cannot cancel the set read and write protection, even if you know the valid password. • If a valid connection exists at the time when the instruction "Change protection level" is called, the call of the instruction does not affect this connection.

If you have configured protection level "2" or "3" for your CPU, using the instruction "Change protection level" has no effect.

You can use the SSL parts list with the SSL ID W#16#0232 and index W#16#0004 to check whether protection level "2" or "3" is still in effect after the execution of the "Change protection level" instruction.

Note

You can only use the "Change protection level" instruction to set a protection level that is equal to or higher than the protection level you have configured with STEP 7 "Configure hardware. You have, for example, configured protection level 3 and then called the instruction "Change protection level" with the MODE parameter = 12. This blocks the password authentication. If the instruction is called again with the parameter MODE = 0 or 1, protection level 3 is retained and the read and write protection canceled by entry of the valid password.

Effects of the operating state transitions on the protection level

The following table shows what effects the operating state transitions have on the protection level set with the instruction "Change protection level".

Action	Effects on the protection level
• Operating mode switch to "STOP" • Unbuffered POWER ON • Memory reset with operating mode switch • Configuration and/or programming changes in the "STOP" mode of the S7-400 CPU	The protection level is set to "1". All programming device functions are permitted.
• Cold restart • Warm restart	The protection level is set to "1" by the operating system. All programming device functions are permitted. If required, you can set the protection level in your program to "2" or "3" by calling the "Change protection level" instruction.
Buffered POWER ON	Remains unchanged
Operating mode transition RUN/STARTUP/HOLD -> STOP (by calling the "Exit program" instruction (STP), by program error without corresponding error OB, or by operator action).	Remains unchanged
S7-400: Restart by user action or automatically following POWER ON	Remains unchanged

Syntax

The following syntax is used for the instruction "Change protection level":

```
SCL
PROTECT(MODE:= <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
MODE	Input	WORD	Protection level Possible values: • W#16#0000: Setting protection level "1" • W#16#0001: Setting protection level "2" with password authentication • W#16#000C: Setting protection level "3" with password authentication
Function value (RET_VAL)		INT	If an error occurs during processing, information on the error is output.

Parameter RET_VAL

The following table shows the meaning of the values of the parameter RET_VAL:

Error code* (W#16#...)	Explanation
0000	Job executed without error
8090	Invalid value at MODE parameter.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Evaluating errors with output parameter RET_VAL (Page 5370)

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Switching display formats in the program status (Page 2980)

CiR: Control CiR process

Description

The instruction "Control CiR process" influences the CiR process in the user program as follows:

- Disable CiR process. In this case, the download of a modified configuration from the programming device to the CPU is always rejected. The disable is maintained until you cancel it again with the instruction "Control CiR process".
- You can disable the CiR process conditionally by specifying a high limit for the CiR synchronization time on the FRZ_TIME parameter. In this case, the download of a modified configuration from the programming device to the CPU is only accepted if the CiR synchronization time calculated in the CPU does not exceed the specified high limit.
- Enable CiR process. The high limit of the CiR synchronization time is set to the default value 1000 ms.
- Determine whether the CiR process is enabled or not. If the CiR process is enabled or conditionally enabled, you will get the current high limit for the CiR synchronization time in the A_FT parameter of the instruction.

Note

During the CiR synchronization period the outputs are frozen and the inputs are not evaluated.

Syntax

The following syntax is used for the instruction "Control CiR process":

```
SCL
CIR(MODE := <Operand>,
    FRZ_TIME := <Operand>,
    A_FT => <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
MODE	Input	BYTE	Job identifier Valid values on the MODE parameter are: • 0: Information function • 1: Enable CiR process • 2: Disable CiR process • 3: Disable CiR conditionally. You specify the high limit of the CiR synchronization time in FRZ_TIME.
FRZ_TIME	Input	TIME	High limit of the CiR synchronization time in milliseconds. Permitted range of values: 60 ... 2500 ms (default: 1000 ms) The FRZ_TIME parameter is only relevant when the value 3 is set on the MODE parameter.
A_FT	Output	TIME	Currently valid high limit of the CiR synchronization time
Function value (RET_VAL)		INT	Error information When the MODE parameter is "0", the RET_VAL parameter supplies information on the enabling of the CiR process.

Parameter RET_VAL

The following table shows the meaning of the values of the parameter RET_VAL:

Error code* (W#16#...)	Explanation
0000	Job executed without error This error code can be output only if the value "1", "2" or "3" is set on the MODE parameter.
0001	CiR is enabled. This error code can be output only if the value "0" is set on the MODE parameter.
0002	The CiR process is disabled. This error code can be output only if the value "0" is set on the MODE parameter.
0003	CiR is disabled conditionally. This error code can be output only if the value "0" is set on the MODE parameter.
8001	The CPU is not ready for a CiR operation.

Error code* (W#16#...)	Explanation
8002	The value at the MODE parameter is invalid.
8003	The value at the FRZ_TIME parameter is invalid.
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

Example

The instruction "Control CiR process" ensures, for example, that no CiR process is initiated during periods in which maximum CPU performance is required for execution of the process.

To do this the "Control CiR process" instruction must be called before the start of the period with increased process activity in the program and executed with the value "2" (Disable CiR process) on the MODE parameter . The CiR process is then disabled.

To enable the CiR process again after the completion of the period with increased process activity, you have to again call the "Control CiR process" instruction and execute it with the value 1 on the MODE parameter. Alternatively, you can also set the value 3 on the MODE parameter to disable the CiR process conditionally.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Evaluating errors with output parameter RET_VAL (Page 3152)

Switching display formats in the program status (Page 2980)

Word logic operations

DECO: Decode

Description

The instruction "Decode" sets a bit specified by the input value in the output value.

The instruction "Decode" reads the value of the IN parameter and sets the bit in the output value, whose bit position corresponds to the read value. The other bits in the output value are filled with zeroes. If the value of the IN parameter is greater than 31, a modulo 32 instruction is executed.

Syntax

The following syntax is used for the instruction "Decode":

SCL

```
DECO(IN := <Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
IN	Input	WORD	Position of the bit in the output value which is set.
Function value		DWORD	Current output value

For additional information on valid data types, refer to "See also".

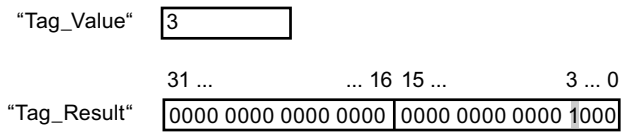
Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := DECO(IN := "Tag_Value");
```

The following figure shows how the instruction works using specific operand values:



The instruction reads the number "3" from the value of the operand "Tag_Value" and sets the third bit to the value of the operand "Tag_Result".

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Operators and operator precedence (Page 2687)

Expressions (Page 2681)

ENCO: Encode

Description

The instruction "Encode" reads the bit number of the lowest-value bit set in the input value and issues this as a result.

Syntax

The following syntax is used for the instruction "Encode":

SCL

```
ENCO(IN := <expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
IN	Input	DWORD	Input value
Function value		INT	Bit number of the bit in the input value that is read out.

For additional information on valid data types, refer to "See also".

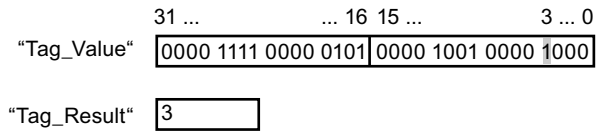
Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := ENCO(IN := "Tag_Value");
```

The following figure shows how the instruction works using specific operand values:



The instruction reads the lowest-value set bit of the operand "Tag_Value" and writes the bit position "3" in the operand "Tag_Result". The result of the instruction is returned in the operand "Tag_Result" as a function value.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Operators and operator precedence (Page 2687)

Expressions (Page 2681)

SEL: Select

Description

The instruction "Select" selects one of the parameters IN0 or IN1 depending on a switch (parameter G) and issues its content as a result. When the parameter G has the signal status "0", the value at parameter IN0 is moved. When the parameter G has the signal status "1", the value at parameter IN1 is moved and returned as a function value.

The instruction is only executed if the tags of all parameters are of the same data type class.

Syntax

The following syntax is used for the instruction "Select":

SCL

```
SEL(G:= <Expression>,
    IN0 := <Expression>,
    IN1 := <Expression>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
G	Input	BOOL	Switch
IN0	Input	Bit strings, integers, floating-point numbers, timers, CHAR, TOD, DATE, DT	First input value
IN1	Input	Bit strings, integers, floating-point numbers, timers, CHAR, TOD, DATE, DT	Second input value
Function value		Bit strings, integers, floating-point numbers, timers, CHAR, TOD, DATE, DT	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := SEL(G := "Tag_Value",
                    IN0 := "Tag_0",
                    IN1 := "Tag_1");
```

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_Value	0	1
Tag_0	W#16#0000	W#16#4C
Tag_1	W#16#FFFF	D#16#5E
Tag_Result	W#16#0000	D#16#5E

The result of the instruction is returned in the operand "Tag_Result" as a function value.

See also

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Expressions (Page 2681)

Operators and operator precedence (Page 2687)

Overview of the valid data types (Page 2236)

MUX: Multiplex

Description

The "Multiplex" instruction copies the value of a selected input parameter and issues it. Use the parameter K to determine the number of the input parameter whose value will be moved. Numbering starts at IN0 and is incremented continuously with each new input. You can declare a maximum of 32 input parameters.

If the value of the K parameter is greater than the number of inputs and the INELSE parameter is not set, the instruction function value is invalid and the ENO enabling output is set to 0.

All tags with assigned parameters must be of the same data type. The function value assumes the data type having the highest value.

Syntax

The following syntax is used for the instruction "Multiplex":

SCL

```
MUX(K:= <Expression>,
    IN0 := <Expression>,
    IN1 := <Expression>,
    INELSE := <Expression>)
```

Parameter

The following table shows the parameters of the "Multiplex" instruction:

Parameter	Declaration	Data type	Description
K	Input	Integers	Specifies the parameter whose content is to be transferred. • If K = 0 => parameter IN0 • If K = 1 => parameter IN1, etc.
IN0	Input	Binary numbers, integers, floating-point numbers, timers, strings, TOD, DATE, DT	First input value
IN1	Input	Binary numbers, integers, floating-point numbers, timers, strings, TOD, DATE, DT	Second input value
INn	Input	Binary numbers, integers, floating-point numbers, timers, strings, TOD, DATE, DT	Optional input values
INELSE	Input	Binary numbers, integers, floating-point numbers, timers, strings, TOD, DATE, DT	Specifies the value to be copied when K <> n.
Function value		Binary numbers, integers, floating-point numbers, timers, strings, TOD, DATE, DT	Result of the instruction

For more information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
"Tag_Result" := MUX(K := "Tag_Number",
                    IN0 := "Tag_1",
                    IN1 := "Tag_2",
                    INELSE := "Tag_3");
```

The result of the instruction is returned in the operand "Tag_Result" as a function value.

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_Number	1	4
Tag_1	DW#16#00000000	DW#16#00000000
Tag_2	DW#16#003E4A7D	DW#16#003E4A7D
Tag_3	DW#16#FFFF0000	DW#16#FFFF0000
Tag_Result	DW#16#003E4A7D	DW#16#FFFF0000

See also

Overview of the valid data types (Page 2236)

Operators and operator precedence (Page 2687)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Shift and rotate

SHR: Shift right

Description

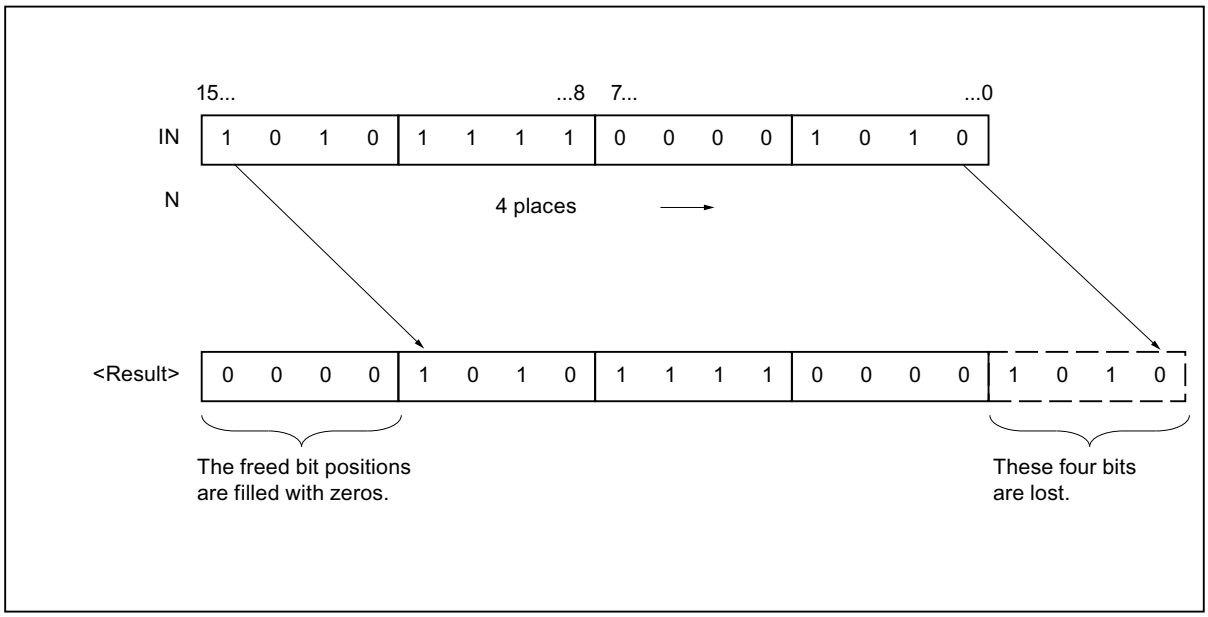
The instruction "Shift right" shifts the content of the parameter IN bit-by-bit to the right and returns it as a function value. The parameter N is used to specify the number of bit positions by which the specified value should be shifted.

If the value of the N parameter is "0", the value of the IN parameter is given as a result.

If the value of the N parameter is greater than the number of available bit positions, the value of the IN parameter is shifted to the right by the available number of bit positions.

The bit positions that are freed by shifting in the left operand area are filled with zeros.

The following figure shows how the content of an operand of the INT data type is shifted four bit positions to the right:



Syntax

The following syntax is used for the instruction "Shift right":

SCL

```
SHR(IN := <Operand>,
    N := <Operand>)
```

Parameter

The following table shows the parameters of the "Shift right" instruction:

Parameter	Declaration	Data type	Description
IN	Input	Bit strings	Value to be shifted.
N	Input	Integers	Number of bits by which the value (IN) is shifted. The value of the N parameter must be positive.
Function value		Bit strings	Result of the instruction

For more information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := SHR(IN := "Tag_Value",
                    N := "Tag_Number");
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_Value	0011 1111 1010 1111
N	Tag_Number	3
Function value	Tag_Result	0000 0111 1111 010 1

The content of the "Tag_Value" operand is shifted by three bit positions to the right. The result of the instruction is returned in the operand "Tag_Result" as a function value.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

SHL: Shift left

Description

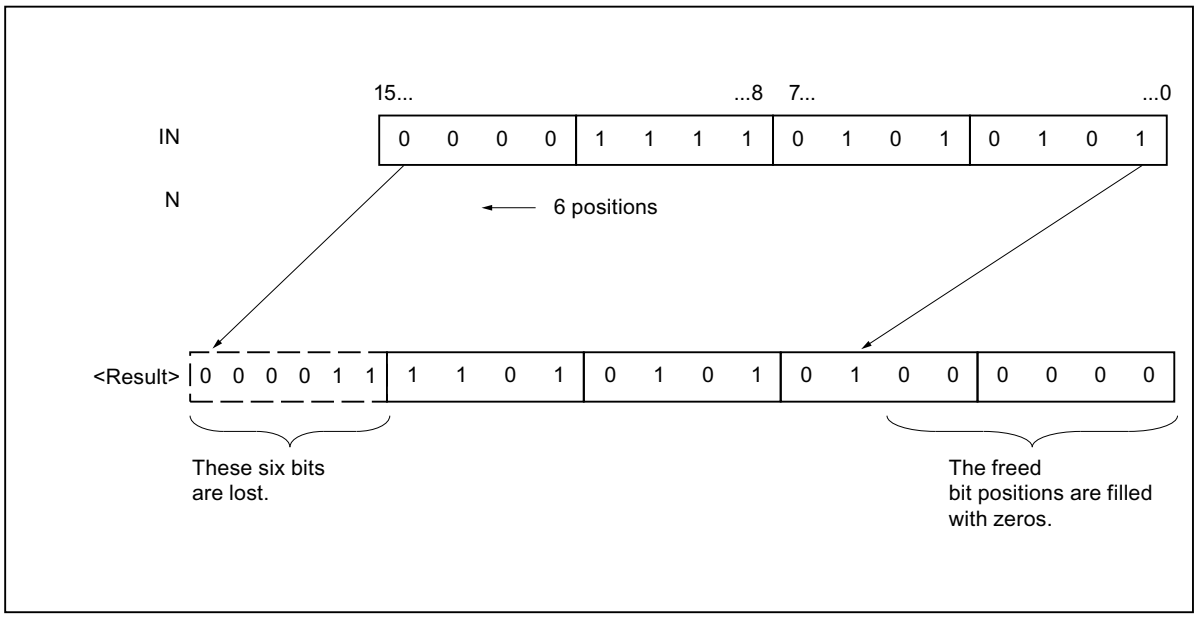
The "Shift left" instruction shifts the contents of the IN parameter bit-by-bit to the left and returns it as a function value. The parameter N is used to specify the number of bit positions by which the specified value should be shifted.

If the value of the N parameter is "0", then the value of the IN parameter will be given as a result.

If the value of the N parameter is greater than the number of bit positions, then the value of the IN parameter will be shifted to the left by the available number of bit positions.

The bit positions freed by the shift are filled with zeros in the result value.

The following figure shows how the content of an operand of the WORD data type is shifted six bit positions to the left:



Syntax

The following syntax is used for the instruction "Shift left":

SCL

```
SHL(IN := <Operand>,
    N := <Operand>)
```

Parameter

The following table shows the parameters of the "Shift left" instruction:

Parameter	Declaration	Data type	Description
IN	Input	Bit strings	Value to be shifted
N	Input	Integers	Number of bits by which the value IN is shifted. The value of the N parameter must be positive.
Function value		Bit strings	Result of the instruction

For more information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := SHL(IN := "Tag_Value",
                    N := "Tag_Number");
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_Value	0011 1111 1010 1111
N	Tag_Number	4
Function value	Tag_Result	1111 1010 1111 0000

The value of the "Tag_Value" operand will be shifted by four bit positions to the left. The result of the instruction is returned in the operand "Tag_Result" as a function value.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

ROR: Rotate right

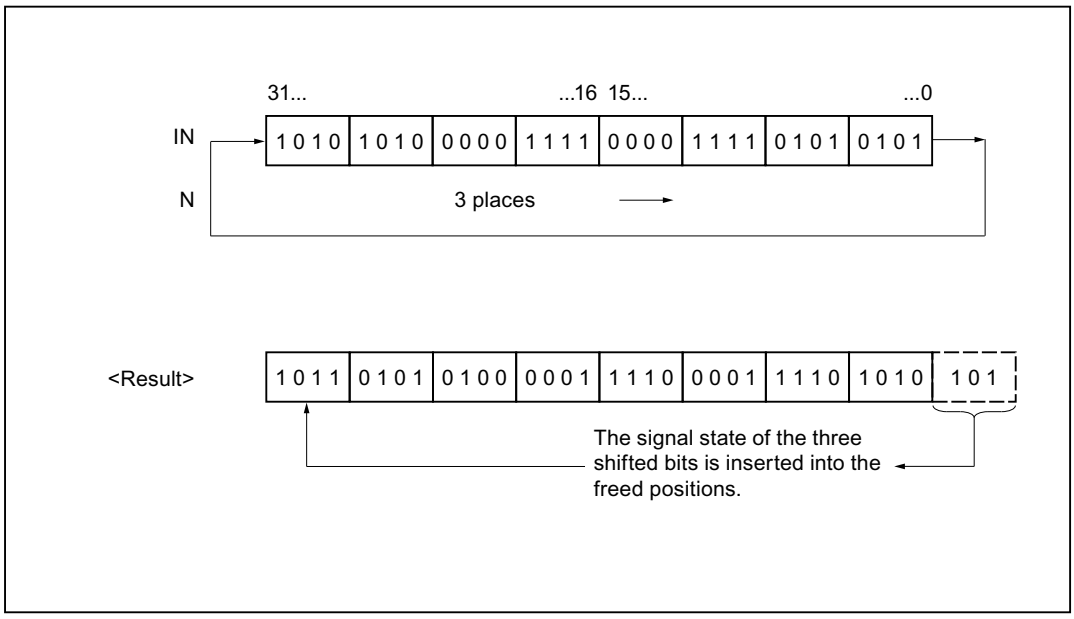
Description

The "Rotate right" instruction rotates the content of the parameter IN bit-by-bit and returns it as a function value. The parameter N is used to specify the number of bit positions by which the specified value should be rotated. The bit positions freed by rotating are filled with the bit positions that are pushed out.

If the value of the N parameter is "0", the value at input IN is given as a result.

If the value at the N parameter is greater than the number of available bit positions, the operand value at the IN input is still rotated by the specified number of bit positions.

The following figure shows how the content of an operand of the DWORD data type is rotated three bit positions to the right:



Syntax

The following syntax is used for the instruction "Rotate right":

SCL

```
ROR(IN := <Operand>,
    N := <Operand>)
```

Parameter

The following table shows the parameters of the "Rotate right" instruction:

Parameter	Declaration	Data type	Description
IN	Input	Bit strings	Value to be rotated
N	Input	Integers	Number of bit positions by which the value IN is rotated. The value of the N parameter must be positive.
Function value		Bit strings	Result of the instruction

For more information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := ROR(IN := "Tag_Value",
                    N := "Tag_Number");
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_Value	1010 1010 0000 1111 0000 1111 0101 0101
N	Tag_Number	5
Function value	Tag_Result	1010 1101 0101 0000 0111 1000 0111 1010

The content of the "Tag_Value" operand is rotated by five bit positions to the right. The result of the instruction is returned in the operand "Tag_Result" as a function value.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

ROL: Rotate left

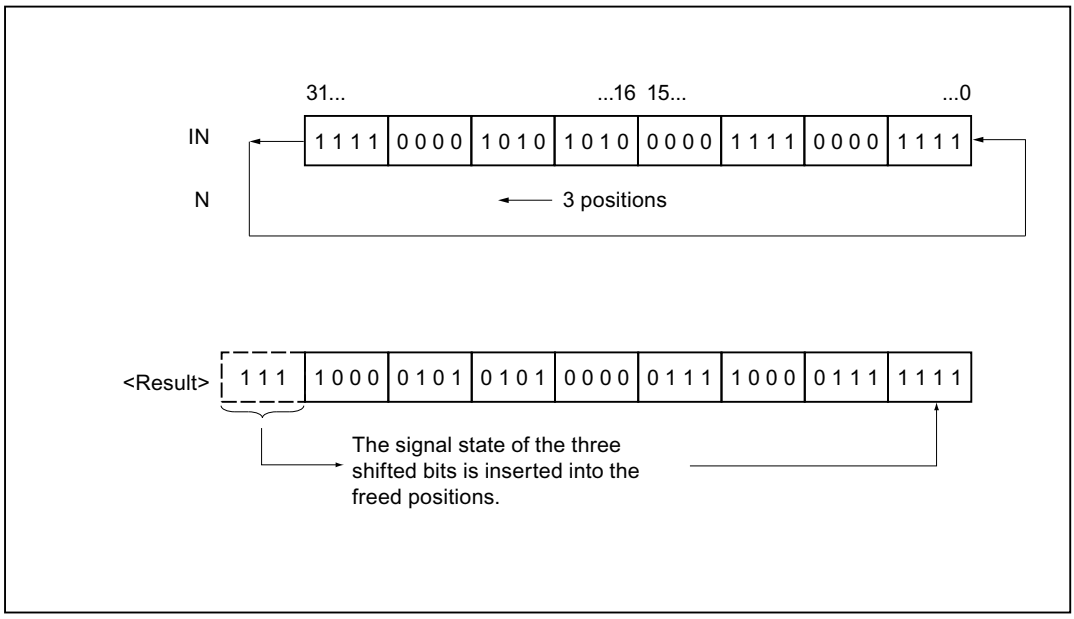
Description

The "Rotate left" instruction rotates the contents of the parameter IN bit-by-bit to the left and returns it as a function value. The N parameter is used to specify the number of bit positions by which the specified value should be rotated. The bit positions freed by rotating are filled with the bit positions that are pushed out.

If the value of the N parameter is "0", then the value at input IN will be given as a result.

If the value at the N parameter is greater than the number of available bit positions, the operand value at the IN input is still rotated by the specified number of bit positions.

The following figure shows how the content of an operand of the DWORD data type is rotated three bit positions to the left:



Syntax

The following syntax is used for the "Rotate left" instruction:

SCL

```
ROL(IN:= <Operand>,
    N := <Operand>)
```

Parameter

The following table shows the parameters of the "Rotate left" instruction:

Parameter	Declaration	Data type	Description
IN	Input	Bit strings	Value to be rotated
N	Input	Integers	Number of bit positions by which the value IN is rotated. The value of the N parameter must be positive.
Function value		Bit strings	Result of the instruction

For more information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := ROL(IN := "Tag_Value",
                    N := "Tag_Number");
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	Tag_Value	1111 0000 1010 1010 0000 1111 0000 1111
N	Tag_Number	5
Function value	Tag_Result	0001 0101 0100 0001 1110 0001 1111 1110

The content of operand "Tag_Value" is rotated five bit positions to the left. The result of the instruction is returned in the operand "Tag_Result" as a function value.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Additional instructions

SET: Set bit array

Description

The "Set bit array" instruction sets the signal state of the bit in a specified area to "1". You use the pointer on the S_BIT parameter to define the starting point of the range. If the pointer points to the memory area of the external I/O, the instruction is not executed. At the N parameter, you specify the number of bits to be set in the specified area. When the N parameter has the value "0", calling the instruction has no effect.

Syntax

The following syntax is used for the "Set bit array" instruction:

```
SCL
SET(S_BIT := <operand>,
    N := <operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
S_BIT	Input	POINTER	Pointer to the first bit of the range whose bits are set. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	Number of bits to be set

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

SCL

```
SET(S_BIT := M0.0,
    N := 10);
```

In the example, the instruction sets 10 bits in the range from M0.0 to M1.1.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

SETP: Set a bit array in the I/O area

Description

The "Set a bit array in the I/O area" instruction sets the bits of an I/O area to the signal status "1". At the same time the corresponding bits in the process image output are also set parallel to the I/O bits.

You use the N and SA parameters to define the bit array that will be set. The bits of the array must be located in the part of the I/O area to which a process image is assigned.

If no I/O devices are plugged in for part of the selected bit array, the instruction attempts to set the entire bit array. In this case an error information is output on the RET_VAL parameter.

The I/O area is set byte-by-byte. If the bit array defined via the N and SA parameters does not begin or end at a byte boundary, calling the instruction has the following effect:

- The bits that are in the first and last bytes to be transferred and that do not belong to the defined bit array, get the value of the corresponding bits in the process image output table. This can lead to unintended reactions such as starting a motor or turning off a cooling system.
- The bits belonging to the defined bit array are set to the signal state "1".

Calling the "Set a bit array in the I/O area instruction has no effect if the N parameter returns the value "0".

Syntax

The following syntax is used for the "Set a bit array in the I/O area" instruction:

SCL

```
SETP(N := <Operand>,
     SA => <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
N	Input	INT	Number of bits to be set
SA	Output	POINTER	Pointer to the first bit to be set
Function value (RET_VAL)		INT	Error information

Parameter RET_VAL

The following table shows the meaning of the values of the parameter RET_VAL:

Error code* (W#16#...)	Explanation
0000	No error
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

SCL

```
"Tag_Error" := SETP(N := 8,
                    SA => P#10.0);
```

In the example, the instruction sets 8 bits in the I/O range corresponding to the outputs Q10.0 to Q10.7.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Evaluating errors with output parameter RET_VAL (Page 3152)

Switching display formats in the program status (Page 2980)

SETI: Set byte array

Description

The "Set byte array" instruction sets the signal state of the bit in a specified area to "1". You define the start point of the array on the S_BYTE. You define the size of the array by specifying the number of bits to be set on the N parameter of the instruction.

Note

The value of the N parameter must be a multiple of the number eight (for example, 8, 16, 24, etc.).

The pointer specified on the S_BYTE parameter must reference the external I/O memory area (P). Since the "P" memory area is accessed as bytes, words, or double words, the pointer on the S_BYTE parameter must reference an operand that corresponds to a byte limit value, in other words the bit number of the pointer must be "0".

Note

The signal state of the corresponding bits in the process-image output table (Q) is reset to "0".

Calling the "Set byte array" instruction has no effect if one of the following conditions applies:

- The pointer on the S_BYTE parameter does not reference the external I/O memory area (P).
- The pointer on the S_BYTE parameter references an operand that is not byte-aligned.
- The value of the N parameter is not a multiple of the number 8.

Syntax

The following syntax is used for the "Set byte array" instruction:

SCL

```
SETI(S_BYTE := <operand>,
     N := <operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
S_BYTE	Input	POINTER	Pointer to the first byte of the array whose bits are to be set. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	Size of the array to be set The array is specified by the number of bits as a multiple of eight (for example 8, 16, 24 etc.).

Example

The following example shows how the instruction works:

SCL

```
SETI(S_BYTE := byte_array[100],
     N := 16);
```

In the example, the instruction sets 2 bytes (16 bits) in the range from 100.0 to 101.7 of the byte_array.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

RESET: Reset bit array

Description

The "Reset bit array" instruction resets the signal state of the bit in a specified area to "0". You use the pointer on the S_BIT parameter to define the starting point of the range. If the pointer points to the memory area of the external I/O, the instruction is not executed. At the N parameter, you specify the number of bits to be reset in the specified range. When the N parameter has the value "0", calling the instruction has no effect.

Syntax

The following syntax is used for the "Reset bit array" instruction:

SCL

```
RESET(S_BIT := <operand>,
      N := <operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
S_BIT	Input	POINTER	Pointer to the first bit of the range whose bits are to be reset For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	Number of bits to be reset.

Example

The following example shows how the instruction works:

SCL

```
RESET(S_BIT := M0.0,
      N := 10);
```

In the example the instruction resets 10 bits in the range from M0.0 to M1.1.

See also

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Overview of the valid data types (Page 2236)

RESETP: Reset a bit array in the I/O area

Description

The "Reset a bit array in the I/O area" instruction resets the bits of an I/O area to the signal status "0". At the same time the corresponding bits in the process image of the outputs are also reset parallel to the I/O bits.

You use the N and SA parameters to define the bit array that will be reset. The bits of the array must be located in the part of the I/O area to which a process image is assigned.

If no I/O devices are plugged in for part of the selected bit array, the instruction attempts to set the entire bit array. In this case an error information is output on the RET_VAL parameter.

The I/O area is reset byte-by-byte. If the bit array defined via the N and SA parameters does not begin or end at a byte boundary, calling the instruction has the following effect:

- The bits that are in the first and last bytes to be transferred and that do not belong to the defined bit array, get the value of the corresponding bits in the process image output table.

  WARNING

  Set bits

  The setting of the bits at the value of the relevant bit in the process image output table can lead, for example, to lead to unintended reactions such as starting a motor or turning off a cooling system.

- The bits belonging to the defined bit array are reset to the signal state "0".

Calling the "Reset a bit array in the I/O area" instruction has no effect if the N parameter returns the value "0".

Syntax

The following syntax is used for the "Reset a bit array in the I/O area" instruction:

```
SCL
RESETP(N := <Operand>,
       SA => <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
N	Input	INT	Number of bits to be reset.
SA	Output	POINTER	Pointer to the first bit to be reset.
Function value (RET_VAL)		INT	Error information

Parameter RET_VAL

The following table shows the meaning of the values of the parameter RET_VAL:

Error code* (W#16#...)	Explanation
0000	No error
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

```
SCL
"Tag_Error" := RESETP(N := 8,
                      SA => P#10.0);
```

In the example, the instruction resets 8 bits in the I/O range corresponding to the outputs Q10.0 to Q10.7.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Evaluating errors with output parameter RET_VAL (Page 3152)

Switching display formats in the program status (Page 2980)

RESETI: Reset byte array

Description

The "Reset byte array" instruction resets the signal state of the bit in a specified area of bytes to "0". You use the pointer on the S_BYTE parameter to define the starting point of the range. You define the size of the range by specifying the number of bits to be reset on the N parameter of the instruction.

Note

The value of the N parameter must be a multiple of the number eight (for example, 8, 16, 24, etc.).

The pointer specified on the S_BYTE parameter must reference the external I/O memory area (P). Since the "P" memory area is accessed as bytes, words, or double words, the pointer on the S_BYTE parameter must reference an operand that corresponds to a byte limit value, in other words the bit number of the pointer must be "0".

Note

The signal state of the corresponding bits in the process-image output table (Q) is reset to "0".

Calling the "Reset byte array" instruction has no effect if one of the following conditions applies:

- The pointer on the S_BYTE parameter does not reference the external I/O memory area (P).
- The pointer on the S_BYTE parameter references an operand that is not byte-aligned.
- The value of the N parameter is not a multiple of the number 8.

Syntax

The following syntax is used for the "Reset byte array" instruction:

SCL

```
RESETI(S_BYTE := <operand>,
       N := <operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
S_BYTE	Input	POINTER	Pointer to the first byte of the range whose bits are to be reset For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	Size of the array whose bits are to be reset. The array is specified by the number of bits as a multiple of eight (for example 8, 16, 24 etc.).

Example

The following example shows how the instruction works:

SCL

```
RESETI(S_BYTE := byte_array[100],
       N := 16);
```

In the example, the instruction resets 2 bytes (16 bits) in the range from 100.0 to 101.7 of the byte_array.

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

REPL_VAL: Enter substitute value

Description

The "Enter substitute value" instruction enters a substitute value in the accumulator 1 (ACCU 1) from a synchronous error organization block.

Note

The "Enter substitute value" instruction may only be called in a synchronous error organization block (OB 121, OB 122).

The "Enter substitute value" instruction is used when you can no longer read any value from a module. The OB 122 is started after each access to this module. By calling the "Enter substitute value" instruction you can cause the substitute value to be downloaded to accumulator 1. In this case the program will then be processed with this substitute value. You can take the required information for the selection of the substitute value from the local tags of the OB 122.

Syntax

The following syntax is used for the "Enter substitute value" instruction:

SCL

```
REPL_VAL(VAL := <Operand>)
```

The syntax of the instruction consists of the following parts:

Parameter	Declaration	Data type	Description
VAL	Input	DWORD	Substitute value
Function value (RET_VAL)		INT	Error information

Parameter RET_VAL

The following table shows the meaning of the values of the parameter RET_VAL:

Error code* (W#16#...)	Description
0000	No error
8080	The instruction was not called from a synchronous error OB.
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Evaluating errors with output parameter RET_VAL (Page 5370)

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Switching display formats in the program status (Page 2980)

DRUM: Implement sequencer

Description

The "Implement sequencer" instruction is used to assign the programmed values of the OUT_VAL parameter of the corresponding step to the programmed output bits (OUT1 to OUT16) and the output word (OUT_WORD). The specific step must thereby satisfy the conditions of the programmed enable mask on the S_MASK parameter while the instruction remains at this step. The instruction advances to the next step if the event for the step is true and the programmed time for the current step elapses, or if the value at the JOG parameter changes from "0" to "1". The instruction is reset if the signal state on the RESET parameter changes to "1". The current step is hereby equated to the preset step (DSP).

The amount of time spent on a step is determined by the product of the preset timebase (DTBP) and the preset counter value (S_PRESET) for each step. At the start of a new step, this calculated value is loaded into the DCC parameter, which contains the time remaining for the current step. If, for example the value at the DTBP parameter is "2" and the preset value for the first step is "100" (100 ms), the DCC parameter has the value "200" (200 ms).

A step can be programmed with a time value, an event, or both. Steps that have an event bit and the time value "0" advance to the next step as soon as the signal state of the event bit is "1". Steps that are programmed only with a time value start the time immediately. Steps that are programmed with an event bit and a time value greater than "0" start the time when the signal state of the event bit is "1". The event bits are initialized with a signal state of "1".

When the sequencer is on the last programmed step (LST_STEP) and the time for this step has expired, the signal state on the Q parameter is set to "1"; otherwise it is set to "0". When the parameter Q is set, the instruction remains on the step until it is reset.

In the configurable mask (S_MASK) you can selected the separate bits in the output word (OUT_WORD) and set or reset the output bits (OUT1 to OUT16) by means of the output values (OUT_VAL). If a bit of the configurable mask is in the signal state "1", the value OUT_VAL sets or resets the corresponding bit. If the signal state of a bit of the configurable mask is "0", the corresponding bit is left unchanged. All the bits of the configurable mask for all 16 steps are initialized with a signal state of "1".

The output bit on the OUT1 parameter corresponds to the least significant bit of the output word (OUT_WORD). The output bit on the OUT16 parameter corresponds to the most significant bit of the output word (OUT_WORD).

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate

data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For more information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Implement sequencer" instruction:

SCL

```
<Instance>RESET:= <Operand>,
          JOG := <Operand>,
          DRUM_EN := <Operand>,
          LST_STEP := <Operand>,
          EVENT1 - 16 := <Operand>,
          OUT1 - 16 => <Operand>,
          Q => <Operand>,
          OUT_WORD => <Operand>,
          ERR_CODE => <Operand>)
```

Parameter

The following table shows the parameters of the "Implement sequencer" instruction:

Parameter	Declaration	Data type	Description
RESET	Input	BOOL	A signal state of "1" indicates a reset condition.
JOG	Input	BOOL	When the signal state changes from "0" to "1", the instruction advances to the next step.
DRUM_EN	Input	BOOL	A signal state of "1" allows the sequencer to advance based on the event and time criteria.
LST_STEP	Input	BYTE	Step number of the last step programmed.
EVENT(i), 1 ≤ i ≤ 16	Input	BOOL	Event bit (i); Initial signal state is "1".
OUT(j), 1 ≤ j ≤ 16	Output	BOOL	Output bit (j)
Q	Output	BOOL	A signal state of "1" indicates that the time for the last step has elapsed.
OUT_WORD	Output	WORD	Word address to which the sequencer writes the output values.
ERR_CODE	Output	WORD	Error information

Parameter	Declaration	Data type	Description
JOG_HIS	Static	BOOL	JOG parameter history bit
EOD	Static	BOOL	A signal state of "1" indicates that the time for the last step has elapsed.
DSP	Static	BYTE	Preset step of the sequencer
DSC	Static	BYTE	Current step of the sequencer
DCC	Static	DWORD	Current numerical value of the sequencer
DTBP	Static	WORD	Preset timebase of the sequencer
PREV_TIME	Static	DWORD	Previous system time
S_PRESET	Static	ARRAY of WORD	Count preset for each step [1 to 16] where 1 count = 1 ms.
OUT_VAL	Static	ARRAY of BOOL	Output values for each step [1 to 16, 0 to 15].
S_MASK	Static	ARRAY of BOOL	Configurable mask for each step [1 to 16, 0 to 15]. Initial signal states are "1".

For more information on valid data types, refer to "See also".

The static parameters are not visible when calling the instruction in the program. These are saved in the instance of the instruction.

Parameter ERR_CODE

The following table shows the meaning of the values of the parameter ERR_CODE:

ERR_CODE*	Explanation
W#16#0000	No error
W#16#000B	The value at the LST_STEP parameter is less than 1 or greater than 16.
W#16#000C	The value at the DSC parameter is less than 1 or greater than the value at the LST_STEP parameter.
W#16#000D	The value at the DSP parameter is less than 1 or greater than the value at the LST_STEP parameter.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For more information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Switching display formats in the program status (Page 2980)

DRUM_X: Implement sequencer

Description

The "Implement sequencer" instruction is used to implement a sequencer with maximum 16 steps. You specify the number of the first step on the DSP parameter. You specify the number of the last step on the LST_STEP parameter.

During each step, all 16 output bits OUT0 to OUT15 and the output parameter OUT_WORD (in which the output bits are collected together) are written. An output bit is assigned either the corresponding bit of the OUT_VAL array that you specified or the output bit of the preceding step. The value assigned depends on the value of the mask bit on the S_MASK parameter.

Note

The default setting for the mask bit is 0. If you want to change the current setting of one or more mask bits, make this change in the instance data block.

The "Implement sequencer" instruction advances to the next step when there is a rising signal edge on the JOG parameter. If the instruction is already in the last step, the tags Q and EOD are set when the JOG parameter has a rising edge. In this case the DCC parameter is set to the value "0". The instruction remains in the last step until the RESET parameter is set to the value "1".

You can also assign parameters so that switching to the next step is time dependent. To do this, the DRUM_EN parameter has to be set to the value "1". The transition to the next step is made when the event bit (EVENTi) belonging to the current step is set and the time specified for the current step has expired. The time for a step is the product of the (DTBP) time base and the time factor valid for the current step (S_PRESET).

Note

The execution time (DCC) remaining in the current step is reduced only when the corresponding event bit (EVENTi) is set.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For more information on this topic, refer to "See also".

If the value "1" is set on the RESET parameter when the instruction is called, the sequencer goes to the step whose number you assigned on the DSP parameter.

A step can be programmed with a time value, an event, or both. Steps that have an event bit and the time value "0" advance to the next step as soon as the signal state of the event bit is "1". Steps that are programmed only with a time value start the time immediately. Steps that

are programmed with an event bit and a time value greater than "0" start the time when the signal state of the event bit is "1".

You can also move the next step at any time via the value at the JOG parameter.

When the instruction is called for the first time, you must set the value "1" on the RESET parameter.

When the sequencer is in the last step and the execution time specified for this step has expired, the parameters Q and EOD are set. The instruction remains in the last step until the RESET parameter changes to the value "1".

The instruction is executed only in the STARTUP and RUN modes.

The operating system resets the "Implement sequencer" instruction during a cold restart. During a warm restart, the instruction is not reset. If you want the "Implement sequencer" instruction to be initialized after a warm restart, you have to call the OB 100 with the value "1" on the RESET parameter.

Syntax

The following syntax is used for the "Implement sequencer" instruction:

```
SCL
<Instance>RESET:= <Operand>,
          JOG := <Operand>,
          DRUM_EN := <Operand>,
          LST_STEP := <Operand>,
          EVENT1 - 16 := <Operand>,
          OUT0 - 15 => <Operand>,
          Q => <Operand>,
          OUT_WORD => <Operand>,
          ERR_CODE => <Operand>)
```

Parameter

The following table shows the parameters of the "Implement sequencer" instruction:

Parameter	Declaration	Data type	Description
RESET	Input	BOOL	The signal state "1" resets the sequencer. When the instruction is called for the first time, you have to set the value "1" on the RESET parameter.
JOG	Input	BOOL	A positive signal edge (compared to the last call) switches the sequencer to the next step if it is not yet in the last step. The next step is enabled depending on the value you assigned to the DRUM_EN parameter.

Parameter	Declaration	Data type	Description
DRUM_EN	Input	BOOL	Control parameter that specifies whether time-dependent switching to the next step is possible. If the value is "1" a time-dependent switching to the next step is possible.
LST_STEP	Input	BYTE	Number of the last step Values from 1 to 16 are permitted.
EVENTi, 1 ≤ i ≤ 16	Input	BOOL	Event bit number i (belongs to step i)
OUTj, 0 ≤ j ≤ 15	Output	BOOL	Output bit number j (identical to the bit number j of OUT_WORD)
Q	Output	BOOL	Status parameter that indicates whether the specified execution time of the last step has expired.
OUT_WORD	Output	WORD	Output bits collected together in a tag
ERR_CODE	Output	WORD	Error information
JOG_HIS	Static	BOOL	Not relevant to user (input parameter JOG of the preceding call)
EOD	Static	BOOL	Identical with output parameter Q
DSP	Static	BYTE	Number of the first step Values from 1 to 16 are permitted.
DSC	Static	BYTE	Number of the current step
DCC	Static	DWORD	The execution time still remaining in the current step in milliseconds (ms) (only relevant if DRUM_EN = 1 and the corresponding event bit = 1)
DTBP	Static	WORD	The time base valid for all steps in milliseconds (ms)
PREV_TIME	Static	DWORD	(Not relevant for the user: system time of the previous call)
S_PRESET	Static	ARRAY of WORD	One-dimensional array with the time factor for each step. Recommended selection of the indices: [1 to 16]. In this case, S_PRESET [x] has the time factor of step x.

Parameter	Declaration	Data type	Description
OUT_VAL	Static	ARRAY of BOOL	Two-dimensional array with the values to be output in each step if they have not been masked out using S_MASK. Recommended selection of the indices: [1 to 16, 0 to 15]. In this case, OUT_VAL [x, y] has the value assigned to output bit OUTy in step x.
S_MASK	Static	ARRAY of BOOL	Two-dimensional array with the mask bits for each step. Recommended selection of the indices: [1 to 16, 0 to 15]. In this case, S_MASK [x, y] contains the mask bit for the "y"the value to be output in step x. Meaning of the mask bits: • 0: The corresponding value of the previous step is assigned to the relevant output bit. • 1: The corresponding value from OUT_VAL is assigned to the relevant output bit.

The static parameters are not visible when calling the instruction in the program. These are saved in the instance of the instruction.

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

ERR_CODE* (W#16#...)	Explanation
0000	No error
8081	Invalid value at parameter LST_STEP.
8082	Invalid value on parameter DSC
8083	Invalid value on parameter DSP.
8084	The product DCC = DTBP * S_PRESET[DSC] exceeds the value 2**31-1 (approximately 24.86 days).
*The error codes can be displayed as integer or hexadecimal value in the program editor. For more information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Switching display formats in the program status (Page 2980)

DCAT: Discrete control-timer alarm

Description

The "Discrete control-timer alarm" instruction is used to accumulate the time from the point at which the CMD parameter issues the command to open or close. The time is accumulated until the preset time (PT) is exceeded or the information is received that the device was opened or closed (O_FB or C_FB) within the specified time. If the preset time is exceeded before the information on the opening or closing of the device is received, the corresponding alarm is activated. If the signal state on the command input changes state before the preset time, the time is restarted.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For more information on this topic, refer to "See also".

The "Discrete control-timer alarm" instruction has the following reactions to the input conditions:

- When the signal state of the CMD parameter changes from "0" to "1", the signal states of the parameters Q, CMD_HIS, ET, OA and CA are influenced as follows:
  - The parameters Q and CMD_HIS are set to "1".
  - The parameters ET, OA and CA are reset to "0".
- When the signal state on the parameter CMD changes from "1" to "0", the parameters Q, ET, OA, CA and CMD_HIS are reset to "0".
- When the signal state of the parameters CMD and CMD_HIS is "1" and the parameter O_FB is set to "0", the time difference (ms) since the last execution of the instruction is added to the value at the parameter ET. If the value of the parameter ET exceeds the value of the parameter PT, the signal state on the parameter OA is set to "1". If the value of the parameter ET does not exceed the value of the parameter PT, the signal state on the parameter OA is reset to "0". The value at the parameter CMD_HIS is set to the value of the parameter CMD.
- If the signal state of the parameters CMD, CMD_HIS and O_FB are set to "1" and the parameter C_FB has the value "0", the signal state of the parameter OA is set to "0". The value of parameter ET is set to the value of parameter PT. If the signal state of the parameter O_FB changes to "0", the alarm is set the next time the instruction is executed. The value of parameter CMD_HIS is set to the value of parameter CMD.
- If the parameters CMD, CMD_HIS and C_FB have the value "0", the time difference (ms) since the last execution of the instruction is added to the value of the parameter ET. If the value of the parameter ET exceeds the value of the parameter PT, the signal state of the parameter CA is set to "1". If the value at the parameter PT is not exceeded, the parameter CA has the signal state "0". The value of parameter CMD_HIS is set to the value of parameter CMD.

- If the parameters CMD, CMD_HIS and O_FB have the signal state "0" and the parameter C_FB is set to "1", the parameter CA is set to "0". The value of parameter ET is set to the value of parameter PT. If the signal state of the parameter C_FB changes to "0", the alarm is set the next time the instruction is executed. The value of parameter CMD_HIS is set to the value of parameter CMD.
- If the parameters O_FB and C_FB simultaneously have the signal state "1", the signal states of both alarm outputs are set to "1".

The "Discrete control-timer alarm" instruction has no error information.

Syntax

The following syntax is used for the "Discrete control-timer alarm" instruction:

SCL

```
<Instance>CMD:= <Operand>,
          O_FB := <Operand>,
          C_FB := <Operand>,
          Q => <Operand>,
          OA => <Operand>,
          CA => <Operand>)
```

Parameter

The following table shows the parameters of the "Discrete control-timer alarm" instruction:

Parameter	Declaration	Data type	Description
CMD	Input	BOOL	A signal state of "0" indicates a "close" command. A signal state of "1" indicates an "open" command.
O_FB	Input	BOOL	Feedback input when opening
C_FB	Input	BOOL	Feedback input when closing
Q	Output	BOOL	Shows the status of the parameter CMD
OA	Output	BOOL	Alarm output when opening
CA	Output	BOOL	Alarm output when closing
ET	Static	DINT	Currently elapsed time, where one count = 1 ms.
PT	Static	DINT	Preset time value, where one count = 1 ms.
PREV_TIME	Static	DWORD	Previous system time
CMD_HIS	Static	BOOL	CMD history bit

For more information on valid data types, refer to "See also".

The static parameters are not visible when calling the instruction in the program. These are saved in the instance of the instruction.

Example

In the following example the parameter CMD changes from "0" to "1". After the execution of the instruction the parameter Q is set to "1" and the two alarm outputs OA and CA have the signal state "0". The parameter CMD_HIS of the instance data block is set to the signal state "1" and the parameter ET is reset to "0".

Note

You can initialize static parameters in the data block.

SCL

```
"DCAT_DB"(CMD := "Tag_Input_CMD",
          O_FB := "Tag_Input_O_FB",
          C_FB := "Tag_Input_C_FB",
          Q => "Tag_Output_Q",
          OA => "Tag_Output_OA",
          CA => "Tag_Output_CA");
```

The following tables show how the instruction works using specific values:

Before processing

In this example the following values are used for the input and output parameters:

Parameter	Operand	Value
CMD	Tag_Input_CMD	TRUE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
Q	Tag_Output_Q	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the instance data block "DCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#12
PT	DBD8	L#222
CMD_HIS	DBX16.0	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
Q	Tag_Output_Q	TRUE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the instance data block "DCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#0
CMD_HIS	DBX16.0	TRUE

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

MCAT: Motor control-timer alarm

Description

The "Motor control-timer alarm" instruction is used to accumulate the time from the point of time from which one of the command inputs (opening or closing) is switched on. The time is accumulated until the preset time is exceeded or the relevant feedback input indicates that the device has executed the requested operation within the specified time. If the preset time is exceeded before the feedback is received, the corresponding alarm is triggered.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Execution of the "Motor control-timer alarm" instruction

The following table shows the reactions of the "Motor control-timer alarm" instruction to the various input conditions:

Input parameters								Output parameters								
ET	O_HIS	C_HIS	O_CMD	C_CMD	S_CMD	O_FB	C_FB	OO	CO	OA	CA	ET	O_HIS	C_HIS	Q	State
X	1	1	X	X	X	X	X	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	X	1	1	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	1	X	X	0	0	0	0	X	0	0	1	Stop
X	X	X	1	1	X	X	X	0	0	0	0	X	0	0	1	Stop
X	0	X	1	0	0	X	X	1	0	0	0	0	1	0	1	Start opening
<PT	1	0	X	0	0	0	X	1	0	0	0	INC	1	0	1	Open
X	1	0	X	0	0	1	0	0	0	0	0	PT	1	0	1	Opened
>=PT	1	0	X	0	0	0	X	0	0	1	0	PT	1	0	0	Opening alarm

Input parameters								Output parameters								
X	X	0	0	1	0	X	X	0	1	0	0	0	0	1	1	Start closing
<PT	0	1	0	X	0	X	0	0	1	0	0	INC	0	1	1	Close
X	0	1	0	X	0	0	1	0	0	0	0	PT	0	1	1	Closed
>=PT	0	1	0	X	0	X	0	0	0	0	1	PT	0	1	0	Closing alarm
X	0	0	0	0	0	X	X	0	0	0	0	X	0	0	1	Stopped

Legend:	
INC	Add the time difference (ms) since the last processing of the FB to ET
PT	PT is set to the same value as ET
X	Cannot be used
<PT	ET < PT
>=PT	ET >= PT

If the input parameters O_HIS and C_HIS both have the signal state "1", they are immediately set to signal state "0". In this case, the last row in the table (X) mentioned above is valid. Because it is therefore no longer possible to check whether the input parameters O_HIS and C_HIS have the signal state "1", the output parameters are set as follows in this case:

OO = FALSE

CO = FALSE

OA = FALSE

CA = FALSE

ET = PT

Q = TRUE

Syntax

The following syntax is used for the "Motor control-timer alarm" instruction:

SCL

```
<Instance>O_CMD:= <Operand>,
          C_CMD := <Operand>,
          S_CMD := <Operand>,
          O_FB := <Operand>,
          C_FB := <Operand>,
          OO => <Operand>,
          CO => <Operand>,
          OA => <Operand>,
          CA => <Operand>,
          Q => <Operand>)
```

Parameter

The following table shows the parameters of the "Motor control-timer alarm" instruction:

Parameter	Declaration	Data type	Description
O_CMD	Input	BOOL	"Open" command input
C_CMD	Input	BOOL	"Close" command input

Parameter	Declaration	Data type	Description
S_CMD	Input	BOOL	"Stop" command input
O_FB	Input	BOOL	Feedback input when opening
C_FB	Input	BOOL	Feedback input when closing
OO	Output	BOOL	"Open" output
CO	Output	BOOL	"Close" output
OA	Output	BOOL	Alarm output when opening
CA	Output	BOOL	Alarm output when closing
Q	Output	BOOL	A signal state of "0" indicates an error condition.
ET	Static	DINT	Currently elapsed time, where one count = 1 ms
PT	Static	DINT	Preset time value, where one count = 1 ms
PREV_TIME	Static	DWORD	Previous system time
O_HIS	Static	BOOL	"Open" history bit
C_HIS	Static	BOOL	"Close" history bit

For additional information on valid data types, refer to "See also".

The static parameters are not visible when calling the instruction in the program. These are saved in the instance of the instruction.

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.

SCL

```
"MCAT_DB"(O_CMD := "Tag_Iput_O_CMD",
          C_CMD := "Tag_Input_C_CMD",
          S_CMD := "Tag_Input_S_CMD",
          O_FB := "Tag_Input_O_FB",
          C_FB := "Tag_Input_C_FB",
          OO => "Tag_OutputOpen",
          CO => "Tag_OutputClosed",
          OA => "Tag_Output_OA",
          CA => "Tag_Output_CA",
          Q => "Tag_Output_Q");
```

The following tables show how the instruction works using specific values:

Before processing

In this example the following values are used for the input and output parameters:

Parameter	Operand	Value
O_CMD	Tag_Input_O_CMD	TRUE
C_CMD	Tag_Input_C_CMD	FALSE
S_CMD	Tag_Input_S_CMD	FALSE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
OO	Tag_OutputOpen	FALSE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	FALSE

The following values are saved in the instance data block "MCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#2
PT	DBD8	L#22
O_HIS	DBX16.0	TRUE
C_HIS	DBX16.1	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OO	Tag_OutputOpen	TRUE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	TRUE

The following values are saved in the instance data block "MCAT_DB" of the instruction:

Parameter	Address	Value
ET	DBD4	L#0
O_HIS	DBX16.0	TRUE
CMD_HIS	DBX16.1	FALSE

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

IMC: Compare input bits with the bits of a mask

Description

The "Compare input bits with the bits of a mask" instruction is used to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bit of a mask. Up to 16 steps with masks can be programmed. The value of the IN_BIT0 parameter is compared with the value of the mask CMP_VAL[x,0], with "x" indicating the step number. On the CMP_STEP parameter, you specify the step number of the mask that is used for the comparison. All programmed values are compared in the same manner. Unprogrammed input bits or unprogrammed bits of the mask have a default signal state FALSE.

If a match is found in the comparison, the signal state of the OUT parameter is set to "1". Otherwise the OUT parameter is set to "0".

If the value of CMP_STEP parameter is greater than 15, the instruction is not executed. An error message is output at the ERR_CODE parameter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For more information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Compare input bits with the bits of a mask" instruction:

SCL

```
<Instance>(IN_BIT0 - 15 := <Operand>,
          CMP_STEP := <Operand>,
          OUT => <Operand>,
          ERR_CODE => <Operand>)
```

Parameter

The following table shows the parameters of the "Compare input bits with the bits of a mask" instruction:

Parameter	Declaration	Data type	Description
IN_BIT0	Input	BOOL	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	Input bit 2 is compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	Input bit 8 is compared with bit 8 of the mask.

Parameter	Declaration	Data type	Description
IN_BIT9	Input	BOOL	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	Input bit 12 is compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	Input bit 15 is compared with bit 15 of the mask.
CMP_STEP	Input	BYTE	The step number of the mask used for the comparison.
OUT	Output	BOOL	A signal state of "1" indicates that a match was found. A signal state of "0" indicates that no match was found.
ERR_CODE	Output	WORD	Error information
CMP_VAL	Static	ARRAY OF BOOL	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

For more information on valid data types, refer to "See also".

The static parameters are not visible when calling the instruction in the program. These are saved in the instance of the instruction.

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000A	The value at the CMP_STEP parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For more information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Switching display formats in the program status (Page 2980)

SMC: Compare scan matrix

Description

You can use the "Compare scan matrix" instruction to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bit of the comparison masks for each step. Processing starts at step 1 and is continued until the last programmed step (LAST) or until a match is found. The input bit of the IN_BIT0 parameter is compared with the value of the mask CMP_VAL[x,0], with "x" indicating the step number. All programmed values are compared in the same manner. If a match is found the signal state of the OUT parameter is set to "1" and the step number with the matching mask is written in the OUT_STEP parameter. Unprogrammed input bits or unprogrammed bits of the mask have a default signal state FALSE. If more than one step has a matching mask, only the first one found is indicated in the OUT_STEP parameter. If no match is found, the signal state of the OUT parameter is set to "0". In this case the value at the OUT_STEP parameter is greater by "1" than the value at the LAST parameter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For more information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Compare scan matrix" instruction:

SCL

```
<Instance>(IN_BIT0 - 15 := <Operand>,
          OUT => <Operand>,
          OUT_STEP => <Operand>,
          ERR_CODE => <Operand>)
```

Syntax

The following table shows the parameters of the "Compare scan matrix" instruction:

Parameter	Declaration	Data type	Description
IN_BIT0	Input	BOOL	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	Input bit 2 is compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	Input bit 9 is compared with bit 9 of the mask.

Parameter	Declaration	Data type	Description
IN_BIT10	Input	BOOL	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	Input bit 12 is compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	Input bit 15 is compared with bit 15 of the mask.
OUT	Output	BOOL	A signal state of "1" indicates that a match was found. A signal state of "0" indicates that no match was found.
OUT_STEP	Output	BYTE	Contains the step number with the matching mask, or the step number which is greater by "1" than the value at the LAST parameter, provided no match is found.
ERR_CODE	Output	WORD	Error information
LAST	Static	BYTE	Specifies the step number of the last step to be scanned for a matching mask.
CMP_VAL	Static	ARRAY OF WORD	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

For more information on valid data types, refer to "See also".

The static parameters are not visible when calling the instruction in the program. These are saved in the instance of the instruction.

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000E	The value at the LAST parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For more information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Switching display formats in the program status (Page 2980)

LEAD_LAG: Lead and lag algorithm

Description

Use the "Lead and lag algorithm" instruction to process signals with an analog tag. The gain value at the GAIN parameter must be greater than zero. The result of the "Lead and lag algorithm" instruction is calculated using the following equation:

$$OUT = \left[\frac{LG_TIME}{LG_TIME + SAMPLE_T}\right] PREV_OUT + GAIN \left[\frac{LD_TIME + SAMPLE_T}{LG_TIME + SAMPLE_T}\right] IN - GAIN \left[\frac{LD_TIME}{LG_TIME + SAMPLE_T}\right] * PREV_IN$$

The instruction "Lead and lag algorithm" supplies plausible results only when processing is in fixed program cycles. The same units must be specified at the parameters LD_TIME, LG_TIME and SAMPLE_T. At LG_TIME > 4 + SAMPLE_T, the instruction approaches the following function:

OUT = GAIN * ((1 + LD_TIME * s) / (1 + LG_TIME * s)) * IN

When the value of the GAIN parameter is less than or equal to zero, the calculation is not performed and an error information is output at the ERR_CODE parameter.

You can use the "Lead and lag algorithm" instruction in conjunction with loops as a compensator in dynamic feed-forward control. The instruction consists of two operations. The "Lead" operation shifts the phase of output OUT so that the output leads the input. The "Lag" operation, on the other hand, shifts the output so that the output lags behind the input. Because the "Lag" operation is equivalent to an integration, it can be used as a noise suppressor or as a low-pass filter. The "Lead" operation is equivalent to a differentiation and can therefore be used as a high-pass filter. The two instructions together (Lead and Lag) result in the output phase lagging behind the input at lower frequencies and leading it at higher frequencies. This means that the "Lead and lag algorithm" instruction can be used as a band pass filter.

When you insert the instruction in the program, the "Call options" dialog opens in which you can specify whether the block parameters will be stored in a separate data block (single instance) or as a local tag (multiple instance) in the block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder under "Program blocks > System blocks". For more information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Lead and lag algorithm" instruction:

SCL

```
<Instance>IN:= <Operand>,
          SAMPLE_T := <Operand>,
          OUT => <Operand>,
          ERR_CODE => <Operand>)
```

Parameter

The following table shows the parameters of the "Lead and lag algorithm" instruction:

Parameter	Declaration	Data type	Description
IN	Input	REAL	The input value of the current sample time (cycle time) to be processed. Constants can also be specified at the IN parameter.
SAMPLE_T	Input	INT	Sample time Constants can also be specified at the SAMPLE_T parameter.
OUT	Output	REAL	Result of the instruction
ERR_CODE	Output	WORD	Error information
LD_TIME	Static	REAL	Lead time in the same unit as sample time.
LG_TIME	Static	REAL	Lag time in the same unit as sample time
GAIN	Static	REAL	Gain as % / % (the ratio of the change in output to a change in input as a steady state).
PREV_IN	Static	REAL	Previous input
PREV_OUT	Static	REAL	Previous output

For more information on valid data types, refer to "See also".

The static parameters are not visible when calling the instruction in the program. These are saved in the instance of the instruction.

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
0009	The value at the GAIN parameter is less than or equal to zero.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For more information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.

```
SCL
"LEAD_LAG_DB"(IN := "Tag_Input",
              SAMPLE_T := "Tag_Input_SAMPLE_T",
              OUT => "Tag_Output_Result",
              ERR_CODE => "Tag_ErrorCode");
```

The following tables show how the instruction works using specific values.

Before processing

In this example the following values are used for the input parameters:

Parameter	Operand	Value
IN	Tag_Input	2.0
SAMPLE_T	Tag_Input_SAMPLE_T	10

The following values are saved in the instance data block "LEAD_LAG_DB" of the instruction:

Parameter	Address	Value
LD_TIME	DBD12	2.0
LG_TIME	DBD16	2.0
GAIN	DBD20	1.0
PREV_IN	DBD24	6.0
PREV_OUT	DBD28	6.0

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OUT	Tag_Output_Result	2.0

The following values are saved in the instance data block "LEAD_LAD_DB" of the instruction:

Parameter	Operand	Value
PREV_IN	DBD24	2.0
PREV_OUT	DBD28	2.0

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Switching display formats in the program status (Page 2980)

TONR_X: Time accumulator

Description

The "Time accumulator" instruction is used to accumulate time values within a period set by the PV parameter. When the TMR_EN parameter changes to signal state "1", the instruction is executed and the time set at PV starts. The "Time accumulator" instruction adds the value of the DELTA_T parameter to the value of the ET parameter as long as the following conditions are satisfied:

- The TMR_EN parameter has the signal state "1".
- The time value at the ET parameter is less than the time value at the PV parameter.
- The RESET parameter has the signal state "0".

When the signal state on the TMR_EN parameter changes to "0", the time on the DELTA_T parameter is no longer added to the time value at the ET parameter.

When the time value at the ET parameter reaches the time value at the PV parameter the Q parameter has the signal state "1". As long as the Q parameter has the signal state "1", the value at the ET parameter remains unchanged. The value at the ET parameter is reset to "0" when the value "1" is set on the RESET parameter.

Since the "Time accumulator" instruction uses the execution time (DELTA_T) of the last cycle of the organization block (OB), the instruction may only be called in cyclic organization blocks.

Note

You have to move the cycle time of the organization block from the startup information to the global tag of the DELTA_T parameters.

Syntax

Use the following syntax for the "Time accumulator" instruction:

SCL

```
TONR_X(TMR_EN := <Operand>,
       RESET := <Operand>,
       PV := <Operand>,
       DELTA_T := <Operand>,
       ET := <Operand>,
       Q => <Operand>)
```

Syntax

The following table shows the parameters of the "Time accumulator" instruction:

Parameter	Declaration	Data type	Description
TMR_EN	Input	BOOL	Activates the time for accumulating the time value.
RESET	Input	BOOL	Reset input
PV	Input	DINT	Maximum duration of time recording
DELTA_T	Input	INT	Cycle time of OB from previous cycle.
ET	InOut	DINT	Current time value of the elapsed time
Q	Output	BOOL	Output that is set when the time PV expires.
Function value		VOID	Empty function value

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

WSR: Save data to shift register

Description

The "Save data to shift register" instruction is used to write a bit from a specified source to a shift register. Each element of the register is shifted into the next address. The data contained in the last address of the shift register is lost after the shift.

Each time the instruction is executed with the value "0" on the RESET parameter, new data are read from the source (S_DATA) and shifted to the start address (START) of the shift register. The LEN parameter specifies the number of elements to be shifted. You specify the data type of the elements on the E_TYPE parameter. The instruction is not executed if an invalid data type is specified on the E_TYPE parameter.

The register is deleted if the RESET parameter is set to "1" when the instruction is executed. The Q parameter is set to the signal state "1" when the shift register is filled with zeros or deleted.

Syntax

The following syntax is used for the "Save data to shift register" instruction:

SCL

```
WSR(RESET := <Operand>,
    S_DATA := <Operand>,
    START := <Operand>,
    LEN := <Operand>,
    E_TYPE := <Operand>
    Q => <Operand>)
```

Parameter

The following table shows the parameters of the "Save data to shift register" instruction:

Parameter	Declaration	Data type	Description
RESET	Input	BOOL	Reset input
S_DATA	Input	POINTER	Points to the source data element to be shifted to the start address (START). For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
START	Input	POINTER	Points to the start address of the register. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
LEN	Input	WORD	Number of elements to be shifted.
E_TYPE	Input	BYTE	Specifies the data type of the elements to be shifted. The following data types are permitted: • W#16#04 = WORD • B#16#05 = INT • B#16#06 = DWORD • B#16#07 = DINT • B#16#08 = REAL

Parameter	Declaration	Data type	Description
Q	Output	BOOL	Is set to "0" if RESET parameter is active (1) or all elements to be shifted have the value "0".
Function value		VOID	Empty function value

Example

In the following example four elements of the data type WORD are shifted, starting from the address specified at the START parameter. When the first element of the register has been shifted to the next address, the first address is filled with the data that the pointer references on the S_DATA parameter. The value of the last element in the register is lost in the process.

SCL

```
WSR(RESET := "Tag_Input_Reset",
    S_DATA := "Tag_Input_DataSource",
    START := "Tag_Input_StartAddress",
    LEN := "Tag_Input_Length",
    E_TYPE := "Tag_Input_ElementType",
    Q => "Tag_Output_Q");
```

No function value is transferred.

The following tables show how the instruction works using specific values:

Before processing

In this example the following values are used for the input and output parameters:

Parameter	Operand	Value
RESET	Tag_Input_Reset	0
S_DATA	Tag_Input_DataSource	DB2.DBX6.0 DW0 = W#16#1111 DW2 = W#16#2222 DW4 = W#16#3333 DW6 = W#16#4444
START	Tag_Input_StartAddress	DB1.DBX0.0 DW6 = W#16#1234
LEN	Tag_Input_Length	W#16#04
E_TYPE	Tag_Input_ElementType	W#16#04
Q	Tag_Output_Q	1

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
RESET	Tag_Input_Reset	0
S_DATA	Tag_Input_DataSource	DB2.DBX6.0 DW0 = W#16#1234 DW2 = W#16#1111 DW4 = W#16#2222 DW6 = W#16#3333
START	Tag_Input_StartAddress	DB1.DBX0.0 DW6 = W#16#1234
LEN	Tag_Input_Length	W#16#04
E_TYPE	Tag_Input_ElementType	B#16#04
Q	Tag_Output_Q	1

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

SHRB: Shift bit to shift register

Description

The "Shift bit to shift register" instruction is used to write a bit from a specified source (DATA) to a shift register. Each time the instruction is executed with a value "0" on the RESET parameter, new data are read from the source and shifted to the start address (S_BIT) of the shift register. All following bits are shifted by one. The bit contained at the last address (S_BIT + N) is lost after the shift. If the RESET parameter is set to "1" during the processing of the instruction, the addresses in the shift register are reset to "0" and not shifted.

Syntax

The following syntax is used for the "Shift bit to shift register" instruction:

SCL

```
SHRB(DATA := <Operand>,
     RESET := <Operand>,
     S_BIT := <Operand>,
     N := <Operand>)
```

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
DATA	Input	BOOL	Source bit
RESET	Input	BOOL	Reset input

Parameter	Declaration	Data type	Description
S_BIT	Input	POINTER	Points to the start bit of the shift register. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	WORD	Number of bits to be shifted.
Function value		VOID	Empty function value

Example

In the following example the N parameter is set to the value 14 ("E" in hexadecimal). As a result 14 bits starting from the address specified on the S_BIT parameter are shifted. After the bits have been shifted, the first address is filled with the data from the source on the DATA parameter. The last bit value is lost.

SCL

```
SHRB(DATA := "Tag_Input_Data",
     RESET := "Tag_Input_Reset",
     S_BIT := "Tag_Input_BitSource",
     N := "Tag_Input_Number");
```

No function value is transferred.

The following tables show how the instruction works using specific values:

Before processing

In this example the following values are used for the input and output parameters:

Parameter	Operand	Value
RESET	Tag_Input_Reset	0
DATA	Tag_Input_Data	0
S_BIT	Tag_Input_BitSource	DB1.DBX1.4
N	Tag_Input_Number	B#16#0E

Before processing the instruction the following signal states are stored in the 14 bits of the register:

1.7	...		S_BIT			...	1.0
0	0	1	1	1	0	1	0

2.7	...					...	2.0
0	0	1	1	1	0	1	0

3.7	...					...	3.0
0	0	1	1	1	0	1	0

After processing

After the processing of the instruction the following signal states are stored in the 14 bits of the register:

1.7	...		S_BIT			...	1.0
0	1	1	0	1	0	1	0

2.7	...					...	2.0
0	1	1	1	0	1	0	0

3.7	...					...	3.0
0	0	1	1	1	0	0	0

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

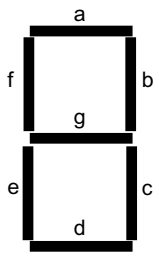
Editing SCL instructions (Page 2716)

SEG: Create bit pattern for seven-segment display

Description

The "Create bit pattern for seven-segment display" instruction is used to convert each of the four hexadecimal digits of the specified source word (IN) into an equivalent bit pattern for a 7-segment display. The result of the instruction is output in the double word on the OUT parameter.

The following relation exists between the hexadecimal digits and the assignment of the 7 segments (a, b, c, d, e, f, g):

Input digit (Binary)	Assignment of the segments - g f e d c b a	Display (Hexadecimal)	Seven-segment display
0000	00111111	0	
0001	00000110	1	
0010	01011011	2	
0011	01001111	3	
0100	01100110	4	
0101	01101101	5	
0110	01111101	6	
0111	00000111	7	
1000	01111111	8	
1001	01100111	9	
1010	01110111	A	
1011	01111100	B	
1100	00111001	C	
1101	01011110	D	
1110	01111001	E	
1111	01110001	F	

Syntax

Use the following syntax for the "Create bit pattern for seven-segment display" instruction:

SCL

```
SEG(IN := <Operand>,
    OUT => <Operand>)
```

Parameter

The following table shows the parameters of the "Create bit pattern for seven-segment display" instruction:

Parameter	Declaration	Data type	Description
IN	Input	WORD	Source word with four hexadecimal digits
OUT	Output	DWORD	Bit pattern for the seven-segment display
Function value		VOID	Empty function value

Example

The following example shows how the instruction works:

```
SCL
SEG(IN := "Tag_Input",
    OUT => "Tag_Output");
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
Hexadecimal	Binary		
IN	Tag_Input	W#16#1234	0001 0010 0011 0100
OUT	Tag_Output	DW16#065B4F66	00000110 01011011 01001111 01100110 Display: 1234

No function value is transferred.

See also

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Overview of the valid data types (Page 2236)

BCDCPL: Create tens complement

Description

The "Create tens complement" instruction is used to create the tens complement of a seven-digit BCD number specified by the operand. This instruction uses the following mathematical formula to calculate:

10000000 (as BCD)

– 7-digit BCD value

Tens complement (as BCD)

Syntax

The following syntax is used for the "Create tens complement" instruction:

```
SCL
BCDCPL(<Operand>)
```

Parameter

The following table shows the parameters of the "Create tens complement" instruction:

Parameter	Declaration	Data type	Description
<Operand>	Input	DWORD	7-digit BCD number
Function value		DWORD	Result of the instruction

For more information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := BCDCPL("Tag_Input");
```

The following table shows how the instruction functions using specific values:

Operand	Value*
Tag_Input	DW#16#01234567
Tag_Result	DW#16#08765433
*The error codes can be displayed as integer or hexadecimal value in the program editor. For more information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Switching display formats in the program status (Page 2980)

BITSUM: Count number of set bits

Description

The "Count number of set bits" instruction is used to count the number of bits of an operand that is set to the signal state "1".

Syntax

The following syntax is used for the "Count number of set bits" instruction:

```
SCL
BITSUM(<Operand>)
```

Parameter

The following table shows the parameters of the "Count number of set bits" instruction:

Parameter	Declaration	Data type	Description
<Operand>	Input	DWORD	Operand whose set bits are counted
Function value		INT	Result of the instruction

For more information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

```
SCL
"Tag_Result" := BITSUM("Tag_Input");
```

The following table shows how the instruction functions using specific values:

Operand	Value*
Tag_Input	DW#16#12345678
Tag_Result	W#16#000D (13 bits)
*The error codes can be displayed as integer or hexadecimal value in the program editor. For more information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Entering SCL instructions (Page 2697)

Editing SCL instructions (Page 2716)

Switching display formats in the program status (Page 2980)

GRAPH

GRAPH structure

Step and transition

Description

You can use the "Step and transition" structure element to insert a step and transition simultaneously in your sequencer.

See also:

Step (Page 6268)

Transition (Page 6269)

See also

Steps and transitions (Page 2724)

Step (Page 6268)

Transition (Page 6269)

Sequence end (Page 6269)

Jump to step (Page 6270)

Alternative branch (Page 6270)

Simultaneous branch (Page 6271)

Close branch (Page 6271)

Step

Description

Steps are used to subdivide complex automation tasks into straightforward subtasks that can then be accomplished using actions. The individual steps are organized in sequencers so that every step can be executed in the specified order in the program execution. Each step must be assigned a unique name and number.

For the execution to actually be performed, the step must be activated by one of the following conditions:

- The step has been defined as an initial step.
- The transition of the previous step has been fulfilled.
- The step has been called by an event-dependent action.

Once all actions have been executed, the step becomes inactive again.

Steps in which no actions are programmed are called empty steps. Such an empty step reacts like an active step and the subsequent transition is always fulfilled.

See also

Basics on steps (Page 2724)

Step and transition (Page 6267)

Transition (Page 6269)

Sequence end (Page 6269)

Jump to step (Page 6270)

Alternative branch (Page 6270)

Simultaneous branch (Page 6271)

Close branch (Page 6271)

Transition

Description

Transitions are located between the steps and contain the conditions for advancing from one step to the next (step-enabling). When the step-enabling conditions of a transition are fulfilled, the next step becomes active and its actions are executed. You can program the conditions of a transition either in LAD or FBD.

See also

Basics on transitions (Page 2726)

Step and transition (Page 6267)

Step (Page 6268)

Sequence end (Page 6269)

Jump to step (Page 6270)

Alternative branch (Page 6270)

Simultaneous branch (Page 6271)

Close branch (Page 6271)

Sequence end

Description

You can use the "Sequence end" structure element to stop a sequencer or a branch. In a simultaneous branch, a transition must precede the sequence end.

Note

If all branches of a sequencer are completed with a sequence end, the sequencer can be restarted by the "INIT_SQ" parameter or using the "Initialize" button in the "Sequence control" pane of the "Testing" task card.

See also

Elements of a sequencer (Page 2721)

Step and transition (Page 6267)

Step (Page 6268)

Transition (Page 6269)

Jump to step (Page 6270)
Alternative branch (Page 6270)
Simultaneous branch (Page 6271)
Close branch (Page 6271)

Jump to step

Description

You can use a jump to continue the program execution at any step within the GRAPH function block. Jumps can be inserted at the end of the main branch or an alternative branch to enable cyclic processing of the sequencer. The jump and jump destination are represented in the sequencer as arrows, in which case the return transition is specified for the jump destination.

See also

Elements of a sequencer (Page 2721)
Step and transition (Page 6267)
Step (Page 6268)
Transition (Page 6269)
Sequence end (Page 6269)
Alternative branch (Page 6270)
Simultaneous branch (Page 6271)
Close branch (Page 6271)

Alternative branch

Description

You can use alternative branches to program OR branches. This means that you insert branches that start with a transition after a step. The corresponding branch is executed depending on which transition is satisfied first. If several transitions are satisfied at the same time, the transition farthest to the left has the highest priority and the corresponding branch is executed. Alternative branches end once again with a transition.

You can program up to 125 alternative branches in a sequencer.

See also

Elements of a sequencer (Page 2721)
Step and transition (Page 6267)
Step (Page 6268)

Transition (Page 6269)

Sequence end (Page 6269)

Jump to step (Page 6270)

Simultaneous branch (Page 6271)

Close branch (Page 6271)

Simultaneous branch

Description

You can use simultaneous branches to program AND branches. This means that you use one transition to activate multiple steps, the actions of which are then executed. Simultaneous branches always start with a step.

The subsequent transitions of the simultaneous branches are located on the main branch, in which case the various simultaneous branches can be connected to various points of the main branch. Note that branches that you join together in a transition only advance to the next step when all branches are completely executed.

You can program up to 249 simultaneous branches in a sequencer.

See also

Elements of a sequencer (Page 2721)

Step and transition (Page 6267)

Step (Page 6268)

Transition (Page 6269)

Sequence end (Page 6269)

Jump to step (Page 6270)

Alternative branch (Page 6270)

Close branch (Page 6271)

Close branch

Description

You can use the "Close branch" element to close simultaneous and alternative branches to their parent branch. This is necessary if you do not want to complete the branch with a jump or a sequence end. In a simultaneous branch, you can insert "Close branch" only after a step.

See also

Elements of a sequencer (Page 2721)
Step and transition (Page 6267)
Step (Page 6268)
Transition (Page 6269)
Sequence end (Page 6269)
Jump to step (Page 6270)
Alternative branch (Page 6270)
Simultaneous branch (Page 6271)

GRAPH actions

Timer operations

TP: Generate pulse

Description

You use the "Generate pulse" instruction to set the Q output for a programmed time. The instruction is started when the result of logic operation (RLO) in the IN parameter changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. The Q parameter is set for the time PT, regardless of the subsequent changes in the input signal. Even when a new positive signal edge is detected, the signal state of the Q parameter is not affected as long as time PT is running.

You can query the current time value set at the ET parameter. The timer value starts at T#0s and ends when the value of time PT is reached. When time PT is reached and the signal state of the IN parameter is "0", the ET parameter is reset.

Note

If the time is not called in the program because it is skipped, for example, the ET parameter returns a constant value as soon as the time has expired.

Each call of the "Generate pulse" instruction must be assigned an IEC timer in which the instruction data is stored. An IEC timer can be declared as follows:

- Declaration of a data block of the type TP (for example, "TP_DB")

The "Call options" dialog opens automatically when the instruction is inserted in the program. You will find the created data block in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when the instruction is called and also each time the Q or ET parameter is accessed.

The operating system resets the instances of the "Generate pulse" instruction during a cold restart. If you want instances of the instruction to be in the initialized state after a warm restart, call these in a startup OB with the value "0" set for the PT parameter. If instances of the "Generate pulse" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

The execution of the "Generate pulse" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Syntax

Use the following syntax for the "Generate pulse" instruction:

GRAPH

```
CALL TP <Data type>, <IEC timer>
     (IN := <Operand>
      PT := <Operand>
      Q => <Operand>
      ET => <Operand>
      )
```

Parameters

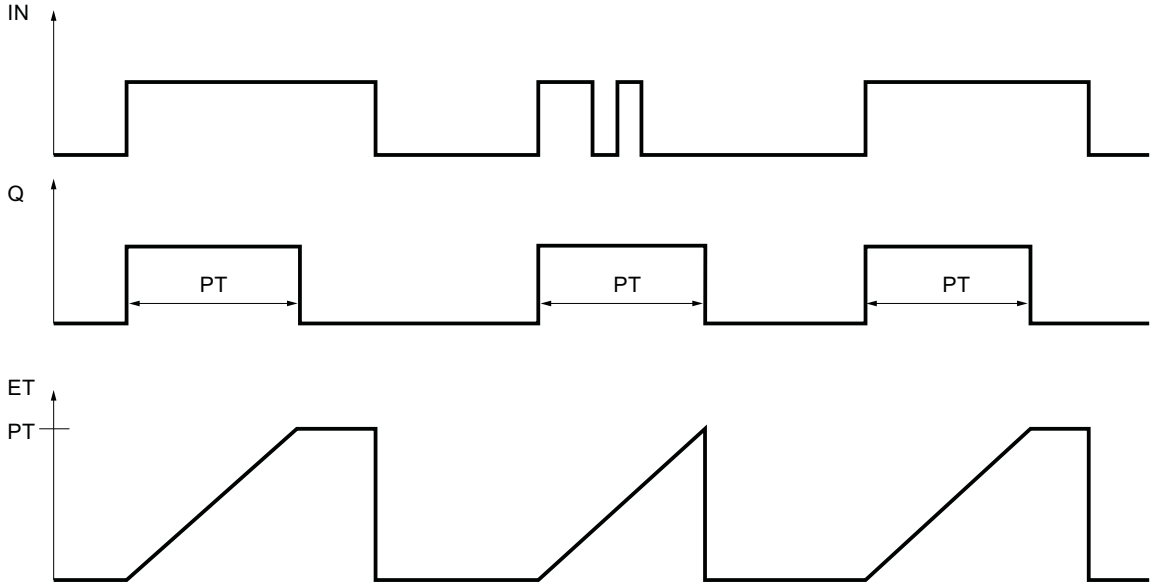
The following table shows the parameters of the "Generate pulse" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L	Start input
PT	Input	TIME	I, Q, M, D, L, P or constant	Duration of the pulse. The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L	Pulse output
ET	Output	TIME	I, Q, M, D, L	Current timer value

For additional information on valid data types, refer to "See also".

Pulse diagram

The following figure shows the pulse diagram of the "Generate pulse" instruction:



Example

The following example shows how the instruction works:

GRAPH

```
CALL TP TIME, "IEC_TP_DB"
     (IN := "Tag_Start"
      PT := "Tag_PresetTIME"
      Q => "Tag_Status"
      ET => "Tag_ElapsedTIME"
      )
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the time programmed for the PT parameter is started and the "Tag_Status" operand is set to "1". The current time value is stored in the "Tag_ElapsedTIME" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

TON: Generate on-delay

Description

You use the "Generate on-delay" instruction to delay setting of the Q output by the programmed time PT. The instruction is started when the result of logic operation (RLO) in the IN input changes from "0" to "1" (positive signal edge). The programmed time PT begins when the instruction starts. When time PT expires, the Q output has the signal state "1". The Q output remains set as long as the start input IN is still "1". When the signal state of the start input changes from "1" to "0", the Q output is reset. The timer function is started again when a new positive signal edge is detected at the start input.

You can query the current time value at the ET output. The timer value starts at T#0s and ends when the value of time PT is reached. The ET output is reset as soon as the signal state in the IN input changes to "0".

Note

If the time is not called in the program because it is skipped, for example, the ET output returns a constant value as soon as the time has expired.

Each call of the "Generate on-delay" instruction must be assigned an IEC timer in which the instruction data is stored. An IEC timer can be declared as follows:

- Declaration of a data block of the type TON (for example, "TON_DB")

The "Call options" dialog opens automatically when the instruction is inserted in the program. You will find the created data block in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when the instruction is called and also each time the Q or ET output is accessed.

The operating system resets the instances of the "Generate on-delay" instruction during a cold restart. If you want instances of the instruction to be in the initialized state after a warm restart, call these in a startup OB with the value "0" set for the PT parameter. If instances of the "Generate on-delay" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

The execution of the "Generate on-delay" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Syntax

The following syntax is used for the "Generate on-delay" instruction:

GRAPH

```
CALL TON <Data type>, <IEC timer>
     (IN := <Operand>
      PT := <Operand>
      Q => <Operand>
      ET => <Operand>
```

GRAPH
)

Parameters

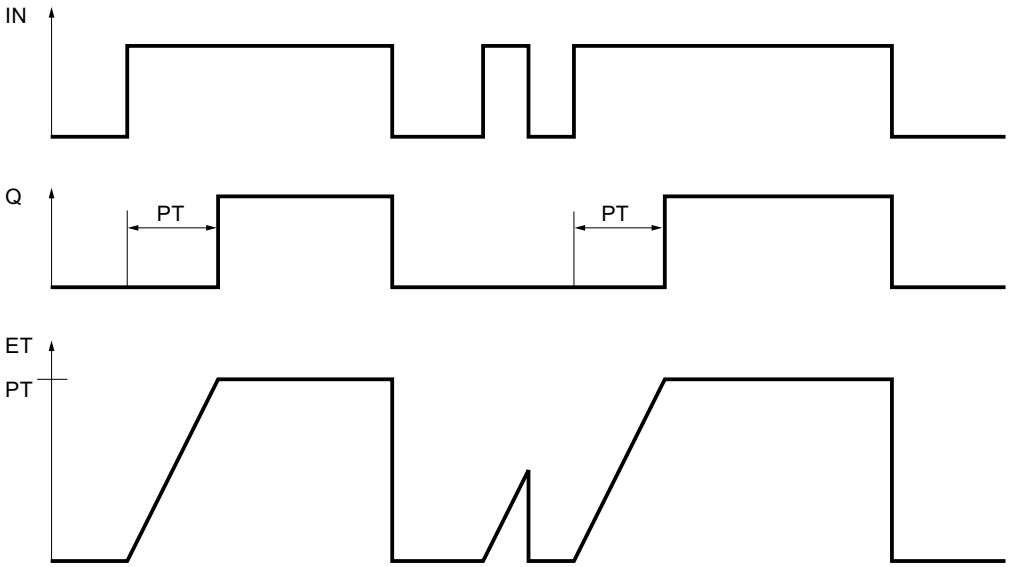
The following table shows the parameters of the "Generate on-delay" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L	Start input
PT	Input	TIME	I, Q, M, D, L, P or constant	Duration of the on-delay The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L	Output that is set when time PT expires.
ET	Output	TIME	I, Q, M, D, L	Current timer value

For additional information on valid data types, refer to "See also".

Pulse diagram

The following figure shows the pulse diagram of the "Generate on-delay" instruction:



Example

The following example shows how the instruction works:

GRAPH

```
CALL TON TIME, "IEC_TON_DB"
     (IN := "Tag_Start"
      PT := "Tag_PresetTIME"
      Q => "Tag_Status"
      ET => "Tag_ElapsedTIME"
      )
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the time programmed for the PT parameter is started. When the time expires, the "Tag_Status" operand is set to the signal state "1". The Tag_Status operand remains set to "1" as long as the Tag_Start operand has signal state "1". The current time value is stored in the "Tag_ElapsedTime" operand. When the signal state of the "Tag_Start" operand changes from "1" to "0", the "Tag_Status" operand is reset.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

TOF: Generate off-delay

Description

You use the "Generate off-delay" instruction to delay resetting of the Q output by the programmed time PT. The Q output is set when the result of logic operation (RLO) in the IN input changes from "0" to "1" (positive signal edge). When the signal state of the IN input changes back to "0", the programmed time PT starts. The Q output remains set as long as time PT is running. When time PT expires, the Q output is reset. If the signal state of the IN input changes to "1" before time PT expires, the timer is reset. The signal state of the Q output continues to be "1".

The current time value can be queried in the ET output. The timer value starts at T#0s and ends when the value of time PT is reached. When time PT expires, the ET output remains set to the current value until the IN input changes back to "1". If the IN input changes to "1" before time PT has expired, the ET output is reset to the value T#0s.

Note

If the time is not called in the program because it is skipped, for example, the ET output returns a constant value as soon as the time has expired.

Each call of the "Generate off-delay" instruction must be assigned an IEC timer in which the instruction data is stored. An IEC timer can be declared as follows:

- Declaration of a data block of the type TOF (for example, "TOF_DB")

The "Call options" dialog opens automatically when the instruction is inserted in the program. You will find the created data block in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The instruction data is updated both when the instruction is called and also each time the Q or ET output is accessed.

The operating system resets the instances of the "Generate off-delay" instruction during a cold restart. If you want instances of the instruction to be in the initialized state after a warm restart, call these in a startup OB with the value "0" set for the PT parameter. If instances of the "Generate off-delay" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

The execution of the "Generate off-delay" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Syntax

The following syntax is used for the "Generate off-delay" instruction:

GRAPH

```
CALL TOF <Data type>, <IEC timer>
     (IN := <Operand>
      PT := <Operand>
      Q => <Operand>
      ET => <Operand>
      )
```

Parameters

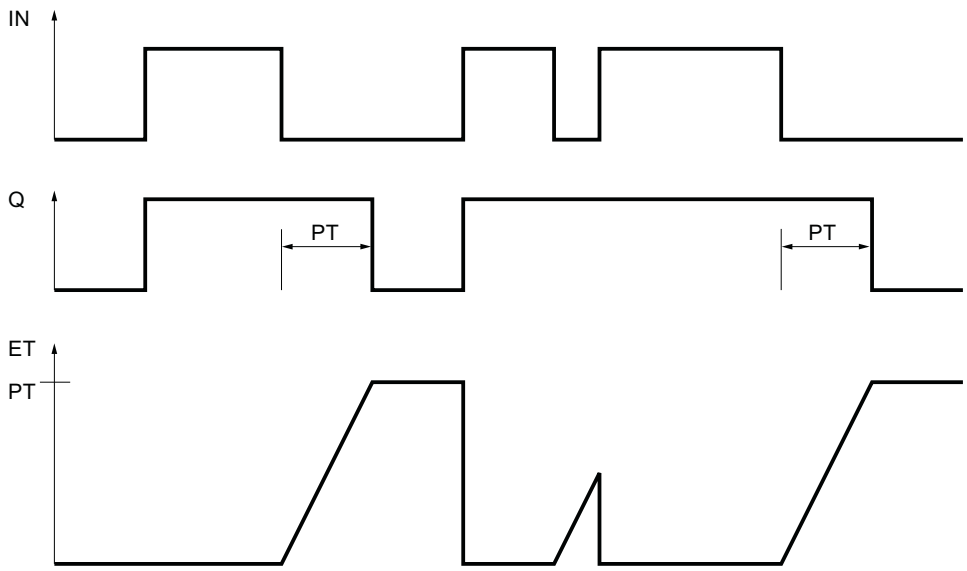
The following table shows the parameters of the "Generate off-delay" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L	Start input
PT	Input	TIME	I, Q, M, D, L, P or constant	Duration of the off delay The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L	Output that is reset when time PT expires.
ET	Output	TIME	I, Q, M, D, L	Current timer value

For additional information on valid data types, refer to "See also".

Pulse diagram

The following figure shows the pulse diagram of the "Generate off-delay" instruction:



Example

The following example shows how the instruction works:

GRAPH

```
CALL TOF TIME, "IEC_TOF_DB"
     (IN := "Tag_Start"
      PT := "Tag_PresetTIME"
      Q => "Tag_Status"
      ET => "Tag_ElapsedTIME"
      )
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the "Tag_Status" operand is set. When the signal state of the "Tag_Start" operand changes from "1" to "0", the time programmed for the PT parameter is started. As long as the time is running, the "Tag_Status" operand remains set. When the time has expired, the Tag_Status operand is reset. The current time value is stored in the "Tag_ElapsedTIME" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

Counter operations

CTU: Count up

Description

You use the "Count up" instruction to increment the value in the CV output. When the signal state of the CU input changes from "0" to "1" (positive signal edge), the instruction is executed and the current count value in the CV output is incremented by one. When the instruction is executed for the first time, the current count value in the CV output is set to zero. The count value is incremented each time a positive signal edge is detected, until it reaches the high limit for the data type specified in the CV output. When the high limit is reached, the signal state of the CU input no longer has an effect on the instruction.

You can query the counter status in the Q output. The signal state of the Q output is determined by the PV parameter. If the current count value is greater than or equal to the value of the PV parameter, the Q output is set to signal state "1". In all other cases, the Q output has signal state "0".

The value of the CV output is reset to zero when the signal state of the R input changes to "1". As long as the R input has signal state "1", the signal state of the CU input has no effect on the instruction.

Note

Only use a counter at a single location in the program to avoid counting errors.

Each call of the "Count up" instruction must be assigned an IEC counter in which the instruction data is stored. You can declare an IEC counter as follows:

- Declaration of a data block of the type CTU (for example, "CTU_DB")

The "Call options" dialog opens automatically when the instruction is inserted in the program. You will find the created data block in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Count up" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, call the instances to be initialized with the value "1" for the R parameter of the instruction in a startup OB. If instances of the "Count up" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

The execution of the "Count up" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Syntax

The following syntax is used for the "Count up" instruction:

GRAPH

```
CALL CTU <Data type>, <IEC counter>
     (CU := <Operand>
```

GRAPH

```
      R := <Operand>
      PV := <Operand>
      Q => <Operand>
      CV => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Count up" instruction:

Parameter	Declaration	Data type	Memory area	Description
CU	Input	BOOL	I, Q, M, D, L	Count input
R	Input	BOOL	I, Q, M, D, L, T, C, P	Reset input
PV	Input	INT	I, Q, M, D, L, P, or constant	Value at which the Q output is set.
Q	Output	BOOL	I, Q, M, D, L	Counter status
CV	Output	INT	I, Q, M, D, L, P	Current count value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL CTU INT,"IEC_CTU_DB"
     (CU := "Tag_Start"
      R := "Tag_ResetCOUNTER"
      PV := "Tag_PresetValue"
      Q => "Tag_Status"
      CV => "Tag_CounterValue"
      )
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the "Count up" instruction is executed and the current count value of the "Tag_CounterValue" operand is incremented by one. With each additional positive signal edge, the count value is incremented until the high limit of the data type (INT = 32767) is reached.

The value of the PV parameter is adopted as the limit for determining the "Tag_Status" output. The "Tag_Status" output has signal state "1" as long as the current count value is greater than or equal to the value of the "Tag_PresetValue" operand. In all other cases, the "Tag_Status" output has signal state "0".

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

CTD: Count down

Description

You use the "Count down" instruction to decrement the value of the CV output. When the signal state of the CD input changes from "0" to "1" (positive signal edge), the instruction is executed and the current count value of the CV output is decremented by one. When the instruction is executed the first time, the count value of the CV parameter is set to the value of the PV parameter. Each time a positive signal edge is detected, the counter is decremented until it reaches the low limit of the specified data type. When the low limit is reached, the signal state of the CD input no longer has an effect on the instruction.

You can query the counter status in the Q output. If the current count value is less than or equal to zero, the Q output is set to signal state "1". In all other cases, the Q output has signal state "0".

The value of the CV output is set to the value of the PV parameter when the signal state of the LD input changes to "1". As long as the LD input has signal state "1", the signal state of the CD input has no effect on the instruction.

Note

Only use a counter at a single location in the program to avoid counting errors.

Each call of the "Count down" instruction must be assigned an IEC counter in which the instruction data is stored. You can declare an IEC counter as follows:

- Declaration of a data block of the type CTD (for example, "CTD_DB")

The "Call options" dialog opens automatically when the instruction is inserted in the program. You will find the created data block in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Count down" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, you must call the instances to be initialized with the value "1" for the LD parameter of the instruction in a startup OB. In this case, the desired initial value for the CV parameter is specified in the PV parameter.

The execution of the "Count down" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Syntax

The following syntax is used for the "Count down" instruction:

GRAPH

```
CALL CTD <Data type>, <IEC counter>
     (CD := <Operand>
      LD := <Operand>
      PV := <Operand>
      Q => <Operand>
      CV => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Count down" instruction:

Parameter	Declaration	Data type	Memory area	Description
CD	Input	BOOL	I, Q, M, D, L	Count input
LD	Input	BOOL	I, Q, M, D, L, P	Load input
PV	Input	INT	I, Q, M, D, L, P, or constant	Value at which the Q output is set
Q	Output	BOOL	I, Q, M, D, L	Counter status
CV	Output	INT	I, Q, M, D, L, P	Current count value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL CTD INT, "IEC_CTD_DB"
     (CD := "Tag_Start"
      LD := "Tag_LoadPV"
      PV := "Tag_PresetValue"
      Q => "Tag_Status"
      CV => "Tag_CounterValue"
      )
```

When the signal state of the "Tag_Start" operand changes from "0" to "1", the "Count down" instruction is executed and the value of the "Tag_CounterValue" output is decremented by one. With each additional positive signal edge, the count value is decremented until it reaches the low limit of the specified data type (INT = -32768).

The "Tag_Status" output has signal state "1" as long as the current count value is less than or equal to zero. In all other cases, the "Tag_Status" output has signal state "0".

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

CTUD: Count up and down

Description

You use the "Count up and down" instruction to increment and decrement the value of the CV output. When the signal state of the CU input changes from "0" to "1" (positive signal edge), the current count value of the CV output is incremented by one. When the signal state of the CD input changes from "0" to "1" (positive signal edge), the current count value of the CV output is decremented by one. If there is a positive signal edge at the CU and CD inputs in one program cycle, the current count value of the CV output remains unchanged.

The count value can be incremented until it reaches the high limit of the data type specified for the CV output. When the high limit is reached, the count value is no longer incremented on a positive signal edge. The count value is no longer decremented once the low limit of the specified data type has been reached.

When the signal state of the LD input changes to "1", the count value of the CV output is set to the value of the PV parameter. As long as the LD input has the signal state "1", the signal state of the CU and CD inputs has no effect on the instruction.

The count value is set to zero when the signal state of the R input changes to "1". As long as the R input has signal state "1", a change in the the signal state of the CU, CD and LD inputs has no effect on the "Count up and down" instruction.

You can query the status of the up counter in the QU output. If the current count value is greater than or equal to the value of the PV parameter, the QU output is set to signal state "1". In all other cases, the QU output has signal state "0".

You can scan the status of the down counter in the QD output. If the current count value is less than or equal to zero, the QD output is set to signal state "1". In all other cases, the QD output has signal state "0".

Note

Only use a counter at a single location in the program to avoid counting errors.

Each call of the "Count up and down" instruction must be assigned an IEC counter in which the instruction data is stored. You can declare an IEC counter as follows:

- Declaration of a data block of the type CTUD (for example, "CTUD_DB")

The "Call options" dialog opens automatically when the instruction is inserted in the program. You will find the created data block in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The operating system resets the instances of the "Count up and down" instruction during a cold restart. If you want instances of the instruction to be initialized after a warm restart, you must call the instances to be initialized with the following parameter values in a startup OB:

- When used as up counter, the value of the R parameter must be set to "1".
- When used as down counter, the value of the LD parameter must be set to "1". In this case you specify the desired initial value for the CV parameter in the PV parameter.

If instances of the "Count up and down" instruction are located within another block, you can reset these instances, for example, by initializing the higher-level block.

The execution of the "Count up and down" instruction requires a preceding logic operation. It can be placed within or at the end of the network.

Syntax

The following syntax is used for the "Count up and down" instruction:

GRAPH

```
CALL CTUD <Data type>, <IEC counter>
     (CU := <Operand>
      CD := <Operand>
      R := <Operand>
      LD := <Operand>
      PV := <Operand>
      QU => <Operand>
      QD => <Operand>
      CV => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Count up and down" instruction:

Parameter	Declaration	Data type	Memory area	Description
CU	Input	BOOL	I, Q, M, D, L	Count up input
CD	Input	BOOL	I, Q, M, D, L	Count down input
R	Input	BOOL	I, Q, M, D, L, T, C, P	Reset input
LD	Input	BOOL	I, Q, M, D, L, P	Load input
PV	Input	INT	I, Q, M, D, L, P, or constant	Value at which the QU output is set
QU	Output	BOOL	I, Q, M, D, L	Status of the up counter
QD	Output	BOOL	I, Q, M, D, L	Status of the down counter
CV	Output	INT	I, Q, M, D, L, P	Current count value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL CTUD INT, "IEC_CTUD_DB"
      (CU := "Tag_StartCTU"
       CD := "Tag_StartCTD"
       R := "Tag_ResetCOUNTER"
       LD := "Tag_LoadPV"
       PV := "Tag_PresetValue"
       QU => "Tag_CounterStatusUP"
       QD => "Tag_CounterStatusDOWN"
       CV => "Tag_CounterValue"
       )
```

When the signal state of the "Tag_StartCTU" or "Tag_StartCTD" input changes from "0" to "1" (positive signal edge), the "Count up and down" instruction is executed. When there is a positive signal edge in the "Tag_StartCTU" input, the current count value is incremented by one and stored in the "Tag_CounterValue" output. When there is a positive signal edge in the "Tag_StartCTD" input, the count value is decremented by one and stored in the "Tag_CounterValue" output. When there is a positive signal edge in the CU input, the count value is incremented until it reaches the high limit of 32767. When there is a positive signal edge in the CD input, the count value is decremented until it reaches the low limit of INT = -32768.

The "Tag_CounterStatusUP" output has signal state "1" as long as the current count value is greater than or equal to the value of the "Tag_PresetValue" input. In all other cases, the "Tag_CounterStatusUP" output has signal state "0".

The "Tag_CounterStatusDOWN" output has signal state "1" as long as the current count value is less than or equal to zero. In all other cases, the "Tag_CounterStatusDOWN" output has signal state "0".

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

Math functions

NEG: Create twos complement

Description

You use the "Create twos complement" instruction to change the sign of the operand value. For example, if the value is positive, the negative equivalent of this value is output.

The value of the OUT output is invalid if the following condition is met:

- The result of the instruction is outside the range permitted for the data type specified for the OUT output (only valid for integers).

Syntax

The following syntax is used for the "Create twos complement" instruction:

GRAPH

```
<Result> := NEG_<Data type>(<Operand>)
```

Parameters

The following table shows the parameters of the "Create twos complement" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	INT, DINT, REAL	I, Q, M, D, L, P, or constant	Input value
<Result>	Output	INT, DINT, REAL	I, Q, M, D, L, P	Twos complement of the input value

You can select the data type of the instruction from the "<???>" drop-down list.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := NEG_REAL("Tag_InValue")
```

The instruction changes the sign of the value of the "TagIn_Value" operand and outputs the result in the "TagOut_Value" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

ABS: Form absolute value

Description

You use the "Form absolute value" instruction to calculate the absolute value of the operand value.

The value of the result is invalid if the following condition is met:

- A floating-point number has an invalid value.

Syntax

The following syntax is used for the "Form absolute value" instruction:

GRAPH

```
<Result> := ABS(<Operand>)
```

Parameters

The following table shows the parameters of the "Form absolute value" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	REAL	I, Q, M, D, L, P, or constant	Input value
<Result>	Output	REAL	I, Q, M, D, L, P	Absolute value of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := ABS("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
TagIn_Value	-6.234
TagOut_Value	6.234

The instruction calculates the absolute value of the value of the "Tag_InValue" operand and outputs the result in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

MIN: Get minimum

Description

You use the "Get minimum" instruction to compare the values of the IN1, IN2, and IN3 inputs and write the lowest value to the OUT output. The instruction is only executed if the tags of all inputs are of the same data type.

The value of the OUT output is invalid if one of the following conditions is met:

- The specified tags are not of the same data type.
- A floating-point number has an invalid value.

Note

Only the INT, DINT, and REAL data types are permitted for the input parameters, even when the IEC check is disabled. If you use the WORD or DWORD data type for the input parameters, the instruction will return an undefined result.

Syntax

The following syntax is used for the "Get minimum" instruction:

GRAPH

```
CALL MIN <Data type>
     (IN1 := <Operand>
      IN2 := <Operand>
      IN3 := <Operand>
      OUT => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Get minimum" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	INT, DINT, REAL	I, Q, M, D, L, P	First input value
IN2	Input	INT, DINT, REAL	I, Q, M, D, L, P	Second input value

Parameter	Declaration	Data type	Memory area	Description
IN3	Input	INT, DINT, REAL	I, Q, M, D, L, P	Third input value
OUT	Output	INT, DINT, REAL	I, Q, M, D, L, P	Result

You can select the data type of the instruction from the "<data type>" drop-down list.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL MIN INT
     (IN1 := "TagIn_Value1"
      IN2 := "TagIn_Value2"
      IN3 := "TagIn_Value3"
      OUT => "Tag_Minimum"
      )
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
IN3	TagIn_Value3	13333
OUT	Tag_Minimum	12222

The instruction compares the values of the specified operands and copies the lowest value ("TagIn_Value1") to the "Tag_Minimum" output.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

MAX: Get maximum

Description

You use the "Get maximum" instruction to compare the values of the IN1, IN2, and IN3 inputs and write the highest value to the OUT output. The instruction is only executed if the tags of all inputs are of the same data type.

The value of the OUT output is invalid if one of the following conditions is met:

- The specified tags are not of the same data type.
- A floating-point number has an invalid value.

Note

Only the INT, DINT, and REAL data types are permitted for the input parameters, even when the IEC check is disabled. If you use the WORD or DWORD data type for the input parameters, the instruction will return an undefined result.

Syntax

The following syntax is used for the "Get maximum" instruction:

GRAPH

```
CALL MAX <Data type>
     (IN1 := <Operand>
      IN2 := <Operand>
      IN3 := <Operand>
      OUT => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Get maximum" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	INT, DINT, REAL	I, Q, M, D, L, P	First input value
IN2	Input	INT, DINT, REAL	I, Q, M, D, L, P	Second input value
IN3	Input	INT, DINT, REAL	I, Q, M, D, L, P	Third input value
OUT	Output	INT, DINT, REAL	I, Q, M, D, L, P	Result

You can select the data type of the instruction from the "<data type>" drop-down list.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL MAX INT
     (IN1 := "TagIn_Value1"
      IN2 := "TagIn_Value2"
      IN3 := "TagIn_Value3"
```

GRAPH

```
OUT => "Tag_Maximum"
)
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN1	TagIn_Value1	12666
IN2	TagIn_Value2	14444
IN3	TagIn_Value3	13333
OUT	TagOut_Value	14444

The instruction compares the values of the specified operands and copies the highest value ("TagIn_Value2") to the "Tag_Maximum" output.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

LIMIT: Set limit value

Description

You use the "Set limit value" instruction to limit the value of the IN input to the values of the MN and MX inputs. If the value of the IN input meets the condition MN <= IN <= MX, it is copied to the OUT output. If the condition is not met and the IN input value is below the low limit MN, the OUT output is set to the value of the MN input. If the high limit MX is exceeded, the OUT output is set to the value of the MX input.

If the value of the MN input is greater than the value of the MX input, the result is undefined. The instruction is only executed if the tags of all inputs are of the same data type.

The value of the OUT output is invalid if one of the following conditions is met:

- The specified tags are not of the same data type.
- An operand has an invalid value.
- The value of the MN input is greater than the value of the MX input.

Note

Only the INT, DINT, and REAL data types are permitted for the input parameters, even when the IEC check is disabled. If you use the WORD or DWORD data type for the input parameters, the instruction will return an undefined result.

Syntax

The following syntax is used for the "Set limit value" instruction:

GRAPH

```
CALL LIMIT <Data type>
     (MN := <Operand>
      IN := <Operand>
      MX := <Operand>
      OUT => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Set limit value" instruction:

Parameter	Declaration	Data type	Memory area	Description
MN	Input	INT, DINT, REAL	I, Q, M, D, L, P	Low limit
IN	Input	INT, DINT, REAL	I, Q, M, D, L, P	Input value
MX	Input	INT, DINT, REAL	I, Q, M, D, L, P	High limit
OUT	Output	INT, DINT, REAL	I, Q, M, D, L, P	Result

You can select the data type of the instruction from the "<data type>" drop-down list.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL LIMIT INT
     (MN := "Tag_LowLimit"
      IN := "Tag_InputValue"
      MX := "Tag_HighLimit"
      OUT => "Tag_Result"
      )
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
MN	Tag_LowLimit	12000
IN	Tag_InputValue	8000
MX	Tag_HighLimit	16000
OUT	Tag_Result	12000

The value of the "Tag_InputValue" operand is compared with the values of the "Tag_LowLimit" and "Tag_HighLimit" operands. Since the value of the "Tag_InputValue" operand is less than the low limit, the value of the "Tag_LowLimit" operand is copied to the "Tag_Result" output.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

SQR: Form square

Description

You use the "Form square" instruction to square the value of the operand.

The value of the result is invalid if the following condition is met:

- The value of the operand is not a valid floating-point number.

Syntax

The following syntax is used for the "Form square" instruction:

GRAPH

```
<Result> := SQR(<Operand>)
```

Parameters

The following table shows the parameters of the "Form square" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	REAL	I, Q, M, D, L, P, or constant	Input value
<Result>	Output	REAL	I, Q, M, D, L, P	Square of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := SQR("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	5.0
Tag_OutValue	25.0

The instruction squares the value of the "Tag_InValue" operand and outputs the result in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

SQRT: Form square root

Description

You use the "Form square root" instruction to calculate the square root of the value of the operand. The instruction outputs a positive result if the input value is greater than zero. If input values are less than zero, the instruction returns an invalid floating-point number. If the value of the operand is "0", then the result is also "0".

The value of the result is invalid if any of the following conditions are met:

- The value of the operand is not a valid floating-point number.
- The value of the operand is negative.

Syntax

The following syntax is used for the "Form square root" instruction:

GRAPH

```
<Result> := SQRT(<Operand>)
```

Parameters

The following table shows the parameters of the "Form square root" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	REAL	I, Q, M, D, L, P, or constant	Input value
<Result>	Output	REAL	I, Q, M, D, L	Square root of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := SQRT("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	25.0
Tag_OutValue	5.0

The instruction calculates the square root of the value of the "Tag_InValue" operand and outputs the result in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

LN: Form natural logarithm

Description

You use the "Form natural logarithm" instruction to calculate the natural logarithm to base e (e = 2.718282) from the value of the operand. The instruction outputs a positive result if the input value is greater than zero. If input values are less than zero, the instruction returns an invalid floating-point number.

The value of the result is invalid if any of the following conditions are met:

- The value of the operand is not a valid floating-point number.
- The value of the operand is negative.

Syntax

The following syntax is used for the "Form natural logarithm" instruction:

GRAPH

```
<Result> := LN(<Operand>)
```

Parameters

The following table shows the parameters of the "Form natural logarithm" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	REAL	I, Q, M, D, L, P, or constant	Input value
<Result>	Output	REAL	I, Q, M, D, L, P	Natural logarithm of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := LN("Tag_InValue")
```

The instruction calculates the natural logarithm of the value in the "Tag_InValue" operand and outputs the result in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

EXP: Form exponential value

Description

You use the "Form exponential value" instruction to calculate the exponent from the base e (e = 2.718282) and the value of the operand.

The value of the result is invalid if the following condition is met:

- The value of the operand is not a valid floating-point number.

Syntax

The following syntax is used for the "Form exponential value" instruction:

GRAPH

```
<Result> := EXP(<Operand>)
```

Parameters

The following table shows the parameters of the "Form exponential value" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	REAL	I, Q, M, D, L, P, or constant	Input value
<Result>	Output	REAL	I, Q, M, D, L, P	Exponential value of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := EXP("Tag_InValue")
```

The instruction calculates the exponent from base e and the value of the "Tag_InValue" operand and outputs the result e^{IN}) in the Tag_OutValue operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

SIN: Form sine value

Description

Use the "Form sine value" instruction to calculate the sine of the angle. The size of the angle is specified in radians in the operand.

The value of the result is invalid if the following condition is met:

- The value of the operand is not a valid floating-point number.

Syntax

The following syntax is used for the "Form sine value" instruction:

GRAPH

```
<Result> := SIN(<Operand>)
```

Parameters

The following table shows the parameters of the "Form sine value" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	REAL	I, Q, M, D, L, P, or constant	Size of angle in radians
<Result>	Output	REAL	I, Q, M, D, L, P	Sine of the specified angle

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := SIN("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	+1.570796 (π/2)
Tag_OutValue	1.0

The instruction calculates the sine of the angle specified in the "Tag_InValue" operand and outputs the result in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

COS: Form cosine value

Description

You use the "Form cosine value" instruction to calculate the cosine of an angle. The size of the angle is specified in radians in the operand.

The value of the result is invalid if the following condition is met:

- The value of the operand is not a valid floating-point number.

Syntax

The following syntax is used for the "Form cosine value" instruction:

GRAPH

```
<Result> := COS(<Operand>)
```

Parameters

The following table shows the parameters of the "Form cosine value" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	REAL	I, Q, M, D, L, P, or constant	Size of angle in radians
<Result>	Output	REAL	I, Q, M, D, L, P	Cosine of the specified angle

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := COS("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	+1.570796 (π/2)
Tag_OutValue	0

The instruction calculates the cosine of the angle specified in the "Tag_InValue" operand and outputs the result in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

TAN: Form tangent value

Description

You use the "Form tangent value" instruction to calculate the tangent of an angle. The size of the angle is specified in radians in the operand.

The value of the result is invalid if the following condition is met:

- The value of the operand is not a valid floating-point number.

Syntax

Use the following syntax for the "Form tangent value" instruction:

GRAPH

```
<Result> := TAN(<Operand>)
```

Parameters

The following table shows the parameters of the "Form tangent value" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	REAL	I, Q, M, D, L, P, or constant	Size of angle in radians
<Result>	Output	REAL	I, Q, M, D, L, P	Tangent of the specified angle

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := TAN("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	+3.141593 (π)
Tag_OutValue	0

The instruction calculates the tangent of the angle specified in the "Tag_InValue" operand and outputs the result in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

ASIN: Form arcsine value

Description

You use the "Form arcsine value" instruction to calculate the size of the angle that corresponds to the sine value specified in the operand. Only valid floating-point numbers within the range -1 to +1 can be specified for the operand. The calculated angle size can range in value from -π/2 to +π/2.

The value of the result is invalid if any of the following conditions are met:

- The value of the operand is not a valid floating-point number.
- The value of the operand is outside the permitted value range (-1 to +1).

Syntax

Use the following syntax for the "Form arcsine value" instruction:

GRAPH

```
<Result> := ASIN(<Operand>)
```

Parameters

The following table shows the parameters of the "Form arcsine value" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	REAL	I, Q, M, D, L, P, or constant	Sine value
<Result>	Output	REAL	I, Q, M, D, L, P	Size of angle in radians

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := ASIN("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	1.0
Tag_OutValue	+1.570796 (π/2)

The instruction calculates the size of the angle corresponding to the sine value in the "Tag_InValue" operand. The result of the instruction is stored in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

ACOS: Form arccosine value

Description

You use the "Form arccosine value" instruction to calculate the size of the angle that corresponds to the cosine value specified in the operand. Only valid floating-point numbers within the range -1 to +1 can be specified for the operand. The calculated angle size can range in value from 0 to +π.

The value of the result is invalid if any of the following conditions are met:

- The value of the operand is not a valid floating-point number.
- The value of the operand is outside the permitted value range (-1 to +1).

Syntax

Use the following syntax for the "Form arccosine value" instruction:

GRAPH

```
<Result> := ACOS(<Operand>)
```

Parameters

The following table shows the parameters of the "Form arccosine value" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	REAL	I, Q, M, D, L, P, or constant	Cosine value
<Result>	Output	REAL	I, Q, M, D, L, P	Size of angle in radians

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH
```
"Tag_OutValue" := ACOS("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	0
Tag_OutValue	+1.570796 (π/2)

The instruction calculates the size of the angle corresponding to the cosine value in the "Tag_InValue" operand. The result of the instruction is stored in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

ATAN: Form arctangent value

Description

You use the "Form arctangent value" instruction to calculate the size of the angle that corresponds to the tangent value specified in the operand. Only valid floating-point numbers may be specified for the operand. The calculated angle size can range in value from -π/2 to +π/2.

The value of the result is invalid if the following condition is met:

- The value of the operand is not a valid floating-point number.

Syntax

The following syntax is used for the "Form arctangent value" instruction:

GRAPH
```
<Result> := ATAN(<Operand>)
```

Parameters

The following table shows the parameters of the "Form arctangent value" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	REAL	I, Q, M, D, L, P, or constant	Tangent value
<Result>	Output	REAL	I, Q, M, D, L, P	Size of angle in radians

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := ATAN("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	1.0
Tag_OutValue	+0.785398 (π/4)

The instruction calculates the size of the angle corresponding to the tangent value in the "Tag_InValue" operand. The result of the instruction is stored in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

Move operations

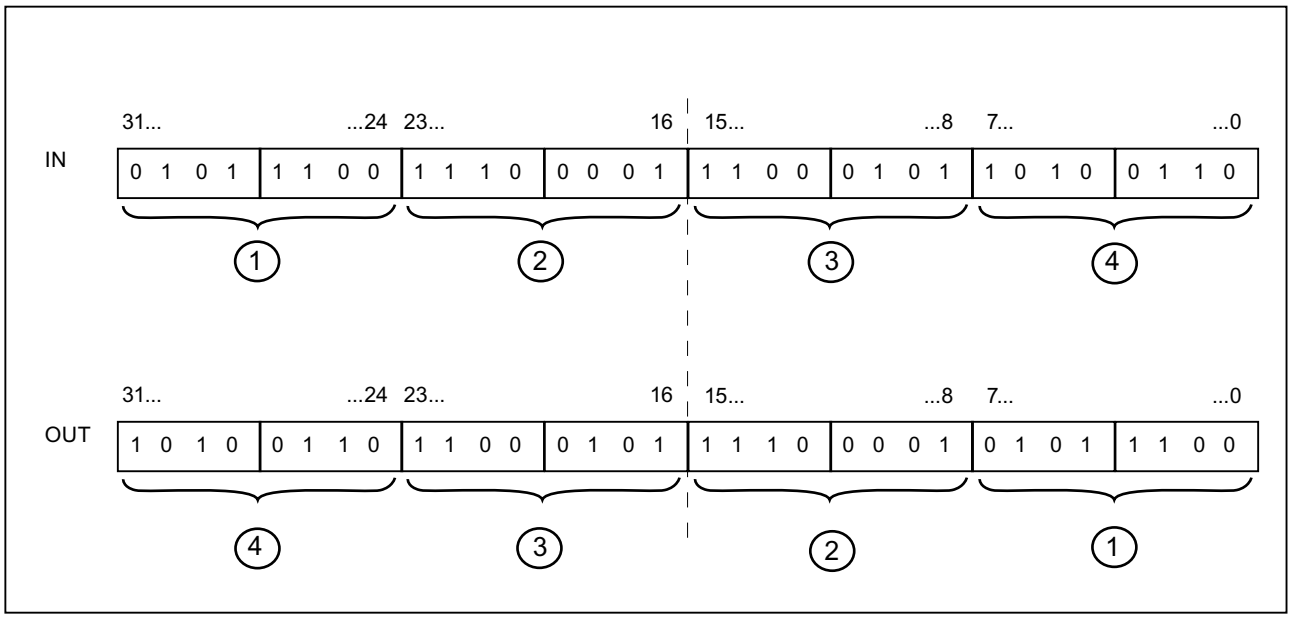
SWAP: Swap

Description

You use the "Swap" instruction to change the order of the bytes of the operand.

Depending on the data type used, you can swap all bytes in accumulator 1 or only the bytes in the right word of accumulator 1.

The following figure shows how the bytes of the operand of data type DWORD are swapped:



Order of bytes in the right word of accumulator 1

For data type WORD, you swap the bytes in the right word of accumulator 1.

The following table shows the contents of accumulator 1 before and after execution of the instruction:

Status	Bytes in accumulator 1			
Before execution	Value A	Value B	Value C	Value D
After execution	Value A	Value B	Value D	Value C

The result of the instruction is saved in the right word of accumulator 1. The bytes in the left word of accumulator 1 are not influenced by the instruction and remain unchanged.

Syntax

The following syntax is used for the "Swap" instruction:

GRAPH

```
<Result> := SWAP(<Operand>)
```

Parameters

The following table shows the parameters of the "Swap" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	WORD, DWORD	I, Q, M, D, L, or constant	Operand whose bytes are swapped
<Result>	Output	WORD, DWORD	I, Q, M, D, L	Result

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := SWAP("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value			
Tag_InValue	0000	1111	0000	1111
Tag_OutValue	0000	1111	1111	0000

Swapping all bytes in accumulator 1:

For data type DWORD, you swap all bytes in accumulator 1.

The following table shows the contents of accumulator 1 before and after execution of the instruction:

Status	Bytes in accumulator 1			
Before execution	Value A	Value B	Value C	Value D
After execution	Value D	Value C	Value B	Value A

The result of the instruction is saved in accumulator 1. The content of accumulator 2 remains unchanged.

The following table shows how the instruction works using specific operand values:

Operand	Value							
Tag_InValue	1111	0000	0000	1111	0000	0000	1111	1111
Tag_OutValue	1111	1111	0000	0000	0000	1111	1111	0000

The bytes in the "Tag_InValue" operand are swapped and stored in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

BLKMOV: Move block

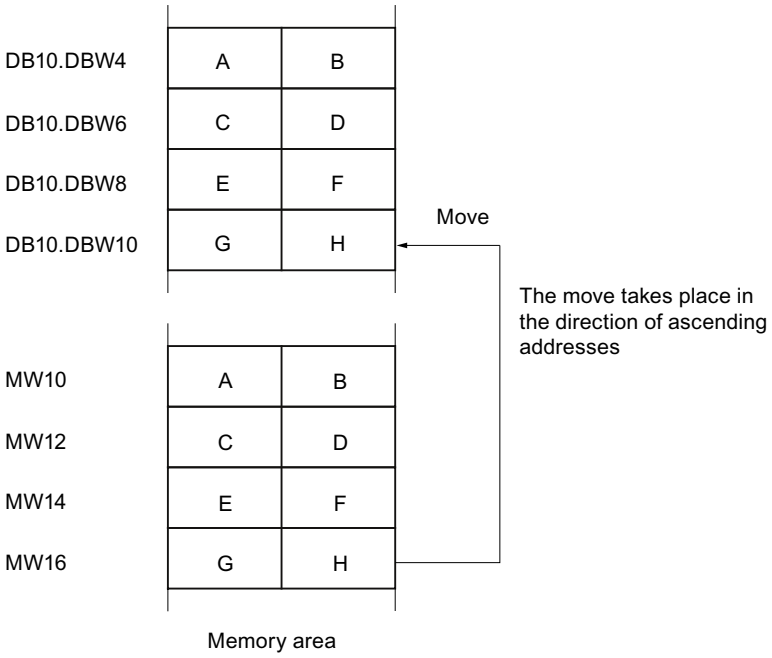
Description

You use the "Move block" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use ANY pointer to define the source and destination area.

Note

In this instruction, you can only use data types which are valid on a CPU of the S7-300/400 series.

The following figure shows the principle of the move operation:



Consistency of source and destination data

Make sure that the source data remain unchanged during execution of the "Move block" instruction. Otherwise, consistency of the destination data cannot be guaranteed.

Interruptibility

As long as the source area is not part of a data block that only exists in the load memory, there is no limit to the nesting depth.

If, however, BLKMOV is interrupted while copying from a DB that is not runtime-relevant, the execution of BLKMOV can no longer be nested.

Memory areas

You can use the "Move block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output
- Data blocks that are not runtime-relevant

General rules for moving

The source and destination area must not overlap. If the source and destination area have different lengths, only the length of the smaller area will be moved.

If the source area is smaller than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is smaller than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a block of data type BOOL is moved, the specified length of the block must be divisible by 8, otherwise it will not be possible to execute the instruction.

Note

If an actual source or destination area is smaller than a destination or source area specified in the SRCBLK or DSTBLK parameter, the system behavior depends on which CPU type is being used:

- The following applies to S7-300 CPUs:
  No data are transferred. Error code W#16#837F is output at the RET_VAL parameter. This error code has to be evaluated in your user program.
- The following applies to the S7-400 V4-CPUs (V4, V4 H-CPU and V4.5. H-CPU):
  No data are transferred. Error code W#16#8122 or W#16#8323 is output at the RET_VAL parameter. This error code has to be evaluated in your user program.
- For all other S7-400-CPUs:
  Data are transferred. Error code W#16#8122 or W#16#8323 is output at the RET_VAL parameter. This error code has to be evaluated in your user program.

Rules for moving character strings

You can also use the "Move block" instruction to move source and destination areas of data type STRING. If only the source area is of data type STRING, the characters that are actually contained in the character string will be moved. Information on the actual and maximum length is also written to the destination area. If both the source and destination area are of data type STRING, the current length of the character string in the destination area is set to the number of actually moved characters.

If you want to move the information on the maximum and actual length of a character string, specify the areas in bytes in the SRCBLK and DSTBLK parameters.

Rules for moving data blocks that are not runtime-relevant

The source area can also be in a non-runtime-relevant data block in the load memory. Data blocks that are not runtime-relevant are indicated by the keyword UNLINKED.

If a non-runtime-relevant data block is copied to the work memory with the "Move block" instruction and loaded at the same time, for example, by the programming device, the execution of the instruction can be delayed for several milliseconds. This results in a longer OB cycle and may trip the cycle monitoring.

If a non-runtime-relevant data block is moved with the "Move block" instruction and the move operation is interrupted, the execution of the instruction can no longer be continued.

If the Read from data block in the load memory instruction is available on your CPU, you must use this instruction to read non-runtime-relevant data blocks in the load memory. If you use the Move block instruction, error W#16#8092 is output.

Syntax

The following syntax is used for the "Move block" instruction:

GRAPH

```
CALL BLKMOV ANY
      (SRCBLK := <Operand>
       RET_VAL => <Operand>
       DSTBLK => <Operand>
       )
```

Parameters

The following table shows the parameters of the "Move block" instruction:

Parameter	Declaration	Data type	Memory area	Description
SRCBLK	Input	ANY	I, Q, M, D, L, P	Specifies the memory area to be moved (source area).
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output at the RET_VAL parameter.
DSTBLK	Output	ANY	I, Q, M, D, L, P	Specifies the memory area to which the block is moved (destination area).

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8092	The source or target area is only available in the load memory.
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

GRAPH

```
CALL BLKMOV ANY
     (SRCBLK := P#M100.0 BYTE 10
      RET_VAL => "Tag_ErrorCode"
      DSTBLK => P#DB1.DBX0.0 BYTE 10
      )
```

The instruction copies 10 bytes starting from MB100 and writes them to DB1. If an error occurs during the move operation, its error code is output at the "Tag_ErrorCode" tag.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

UBLKMOV: Move block uninterruptible

Description

You use the "Move block uninterruptible" instruction to move the content of a memory area (source area) to another memory area (destination area). The move operation takes place in the direction of ascending addresses. You use ANY pointer to define the source and destination area.

The move operation cannot be interrupted by other operating system activities. As a result, the interrupt reaction time of the CPU can increase during execution of the "Move block uninterruptible" instruction.

Note

In this instruction, you can only use data types which are valid on a CPU of the S7-300/400 series.

Memory areas

You can use the "Move block uninterruptible" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output

General rules for moving

The source and destination area must not overlap during execution of the "Move block uninterruptible" instruction. If the source area is smaller than the destination area, the entire source area will be written to the destination area. The remaining bytes of the destination area remain unchanged.

If the destination area is smaller than the source area, the entire destination area will be written. The remaining bytes of the source area are ignored.

If a source or destination area defined as a formal parameter is smaller than a destination or source area specified in the SRCBLK or DSTBLK parameter, no data will be transferred.

If a block of data type BOOL is moved, the specified length of the block must be divisible by 8, otherwise it will not be possible to execute the instruction.

You can use the "Move block uninterruptible" instruction to move a maximum of 512 bytes. Note the CPU specific restrictions for this.

Note

If an actual source or destination area is smaller than a destination or source area specified in the SRCBLK or DSTBLK parameter, the system behavior depends on which CPU type is being used:

- The following applies to S7-300 CPUs:
  No data are transferred. The error code W#16#837F is output in the RET_VAL parameter. This error code must be evaluated in your user program.
- The following applies to the S7-400 V4-CPUs (V4, V4 H-CPU and V4.5. H-CPU):
  No data are transferred. The error code W#16#8122 or W#16#8323 is output in the RET_VAL parameter. This error code must be evaluated in your user program.
- For all other S7-400-CPUs:
  Data are transferred. The error code W#16#8122 or W#16#8323 is output in the RET_VAL parameter. This error code must be evaluated in your user program.

Rules for moving character strings

You can also use the "Move block uninterruptible" instruction to also move source and destination areas of data type STRING. If only the source area is of data type STRING, the characters that are actually contained in the character string will be moved. Information on the actual and maximum length are not written in the destination area. If both the source and destination area are of data type STRING, the current length of the character string in the destination area is set to the number of actually moved characters. If blocks of data type STRING are moved, you must specify "1" as the block length.

Syntax

The following syntax is used for the "Move block uninterruptible" instruction:

GRAPH

```
CALL UBLKMOV ANY
     (SRCBLK := <Operand>
      RET_VAL => <Operand>
      DSTBLK => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Move block uninterruptible" instruction:

Parameter	Declaration	Data type	Memory area	Description
SRCBLK	Input	ANY	I, Q, M, D, L, P	Specifies the memory area to be moved (source area).
RET_VAL	Output	INT	I, Q, M, D, L, P	Error information: If an error occurs during execution of the instruction, an error code is output in the RET_VAL parameter.
DSTBLK	Output	ANY	I, Q, M, D, L, P	Specifies the memory area to which the block is moved (destination area).

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8091	The source area is in a data block that is not runtime-relevant.
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

GRAPH

```
CALL UBLKMOV ANY
     (SRCBLK := P#M100.0 BYTE 10
      RET_VAL => "Tag_ErrorCode"
      DSTBLK => P#DB1.DBX0.0 BYTE 10
      )
```

The instruction copies 10 bytes starting from MB100 and writes them to DB1. If an error occurs during the move operation, its error code is output in the "Tag_ErrorCode" tag.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

FILL: Fill block

Description

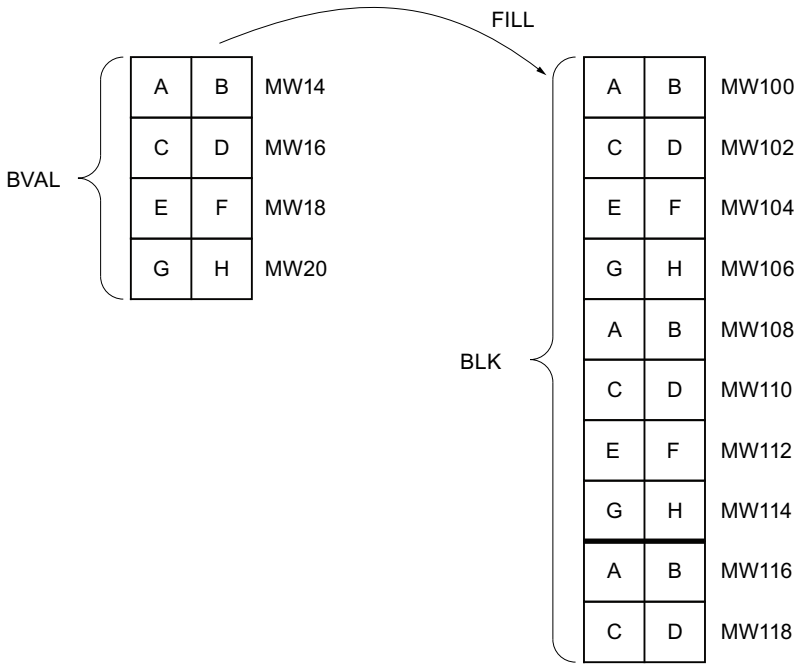
You use the "Fill block" instruction to fill a memory area (destination area) with the content of another memory area (source area). The "Fill block" instruction moves the content of the source area to the destination area until the destination area is completely written. The move operation takes place in the direction of ascending addresses.

You use ANY pointer to define the source and destination area.

Note

In this instruction, you can only use data types which are valid on a CPU of the S7-300/400 series.

The following figure shows the principle of the move operation:



Example: The contents of the range MW100 to MW118 are to be preassigned with the contents of the memory words MW14 to MW20.

Consistency of source and destination data

Make sure that the source data remain unchanged during execution of the "Fill block" instruction, since otherwise consistency of the destination data cannot be guaranteed.

Memory areas

You can use the "Fill block" instruction to move the following memory areas:

- Areas of a data block
- Bit memory
- Process image input
- Process image output
- Data blocks that are not runtime-relevant

General rules for moving

The source and destination area must not overlap. If the destination area to be preassigned is not an integer multiple of the length of the input parameter BVAL, the destination area is nevertheless written up to the last byte.

If the destination area to be preassigned is smaller than the source area, the function only copies as much data as can be written to the destination area.

If the destination or source area actually present is smaller than the assigned memory area for the source or destination area (BVAL, BLK parameters), no data will be transferred.

If the ANY pointer (source or destination) is of data type BOOL, the specified length of the area must be divisible by 8; otherwise the instruction will not be executed.

If the destination area is of data type STRING, the instruction writes the entire string including the administration information.

Note

If an actual source or destination area is smaller than a destination or source area specified in the BVAL or BLK parameter, the system behavior depends on which CPU type is being used:

- The following applies to S7-300 CPUs:
  No data are transferred. The error code W#16#837F is output in the RET_VAL parameter. This error code must be evaluated in your user program.
- The following applies to the S7-400 V4-CPUs (V4, V4 H-CPU and V4.5. H-CPU):
  No data are transferred. The error code W#16#8122 or W#16#8323 is output in the RET_VAL parameter. This error code must be evaluated in your user program.
- For all other S7-400-CPUs:
  Data are transferred. The error code W#16#8122 or W#16#8323 is output in the RET_VAL parameter. This error code must be evaluated in your user program.

Syntax

The following syntax is used for the "Fill block" instruction:

GRAPH

```
CALL FILL ANY
     (BVAL := <Operand>
```

GRAPH

```
      RET_VAL => <Operand>
      BLK => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Fill block" instruction:

Parameter	Declaration	Data type	Memory area	Description
BVAL	Input	ANY	I, Q, M, D, L	Specification of the memory area (source area) whose content is used to fill the destination area of the BLK parameter.
RET_VAL	Output	INT	I, Q, M, D, L	Error information: If an error occurs during execution of the instruction, an error code is output in the RET_VAL parameter.
BLK	Output	ANY	I, Q, M, D, L	Specification of the memory area that will be filled with the content of the source area.

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

GRAPH

```
CALL FILL ANY
      (BVAL := P#M14.0 WORD 4
       RET_VAL => "Tag_ErrorCode"
       BLK => P#M100.0 WORD 10
       )
```

The instruction copies the source area from MW14 to MW20 and fills the destination area from MW100 to MW118 with the content of the 4 words contained in the memory area in the BVAL parameter.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

Conversion operations

CONVERT: Convert value

Description

You use the "Convert value" instruction to read the content of the operand and convert it according to the assigned data types.

For information on possible conversions, refer to section "Explicit conversion" under "See also".

The value of the result is invalid if the following condition is met:

- Errors such as an overflow occur during execution.

Syntax

The following syntax is used for the "Convert value" instruction:

GRAPH

```
<Result> := ???_TO_??? (<Operand>)
```

Parameters

The following table shows the parameters of the "Convert value" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	Integers, floating-point numbers, BCD16, BCD32	I, Q, M, D, L, P, or constant	Value to be converted
<Result>	Output	Integers, floating-point numbers, BCD16, BCD32	I, Q, M, D, L, P	Result of the conversion

You can select the data type of the instruction from the "<???>" drop-down lists.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := INT_TO_DINT("Tag_InValue")
```

The content of the "Tag_InValue" operand is read and converted to an integer (32-bit). The result is stored in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

ROUND: Round numerical value

Description

You can use the "Round numerical value" instruction to round the value of the operand to the nearest integer. The instruction interprets the value as a floating-point number and converts this to an integer of data type DINT. If the input value is exactly between an even and odd number, the even number is selected.

The value of the result is invalid if the following condition is met:

- Errors such as an overflow occur during execution.

Syntax

The following syntax is used for the "Round numerical value" instruction:

GRAPH

```
<Result> := ROUND(<Operand>)
```

Parameters

The following table shows the parameters of the "Round numerical value" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	REAL	I, Q, M, D, L, P or constant	Input value to be rounded
<Result>	Output	DINT	I, Q, M, D, L, P	Result of rounding

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH
```
"Tag_OutValue" := ROUND("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_InValue	1.50000000	-1.50000000
Tag_OutValue	2	-2

The floating-point number in the "Tag_InValue" operand is rounded to the nearest even integer and output in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

CEIL: Generate next higher integer from floating-point number

Description

You use the "Generate next higher integer from floating-point number" instruction to round the value of the operand to the next higher integer. The instruction interprets the value as a floating-point number and converts this number to the next higher integer. The result of the instruction can be greater than or equal to the input value.

The value of the result is invalid if the following condition is met:

- Errors such as an overflow occur during execution.

Syntax

The following syntax is used for the "Generate next higher integer from floating-point number" instruction:

GRAPH
```
<Result> := CEIL(<Operand>)
```

Parameters

The following table shows the parameters of the "Generate next higher integer from floating-point number" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	REAL	I, Q, M, D, L, P, or constant	Input value
<Result>	Output	INT, DINT, REAL	I, Q, M, D, L, P	Result with next higher integer

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := CEIL("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_InValue	0.50000000	-0.50000000
Tag_OutValue	1	0

The floating-point number in the "Tag_InValue" operand is rounded to the next higher integer and output in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

FLOOR: Generate next lower integer from floating-point number

Description

You use the "Generate next lower integer from floating-point number" instruction to round the value of the operand to the next lower integer. The instruction interprets the value as a floating-point number and converts this number to the next lower integer. The result of the instruction can be less than or equal to the input value.

The value of the result is invalid if the following condition is met:

- Errors such as an overflow occur during execution.

Syntax

The following syntax is used for the "Generate next lower integer from floating-point number" instruction:

GRAPH

```
<Result> := FLOOR(<Operand>)
```

Parameters

The following table shows the parameters of the "Generate next lower integer from floating-point number" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	REAL	I, Q, M, D, L, P, or constant	Input value
<Result>	Output	INT, DINT, REAL	I, Q, M, D, L, P	Result with next lower integer

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := FLOOR("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_InValue	0.50000000	-0.50000000
Tag_OutValue	0	-1

The floating-point number in the "Tag_InValue" operand is rounded to the next lower integer and output in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

TRUNC: Truncate numerical value

Description

You use the "Truncate numerical value" instruction to form an integer from the value of the operand. The value is interpreted as a floating-point number. The instruction selects only the integer part of the floating-point number and outputs it without decimal places as the result.

The value of the result is invalid if the following condition is met:

- Errors such as an overflow occur during execution.

Syntax

The following syntax is used for the "Truncate numerical value" instruction:

GRAPH

```
<Result> := TRUNC(<Operand>)
```

Parameters

The following table shows the parameters of the "Truncate numerical value" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand>	Input	REAL	I, Q, M, D, L, or constant	Input value
<Result>	Output	INT, DINT	I, Q, M, D, L	Integer part of the input value

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := TRUNC("Tag_InValue")
```

The following table shows how the instruction works using specific operand values:

Operand	Value	
Tag_InValue	1.50000000	-1.50000000
Tag_OutValue	1	-1

The integer part of the floating-point number in the "Tag_InValue" operand is converted to an integer and output in the "Tag_OutValue" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

SCALE: Scale

Description

You use the "Scale" instruction to convert the integer of the IN parameter to a floating-point number that can be scaled in physical units between a low limit and a high limit. You use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is scaled. The result of the instruction is output at the OUT parameter.

The "Scale" instruction works with the following equation:

OUT = [((FLOAT (IN) – K1)/(K2–K1)) * (HI_LIM–LO_LIM)] + LO_LIM

The values of the constants "K1" and "K2" are determined by the signal state of the BIPOLAR parameter. The following signal states are possible at the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case the "K1" constant has the value "-27648.0" and the "K2" constant the value "+27648.0".
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case the "K1" constant has the value "0.0" and the "K2" constant the value "+27648.0".

When the value of the IN parameter is greater than the value of the constant "K2", the result of the instruction is set to the value of the high limit (HI_LIM) and an error is output.

When the value of the IN parameter is less than the value of the constant "K1", the result of the instruction is set to the value of the low limit (LO_LIM) and an error is output.

When the specified low limit is greater than the high limit (LO_LIM > HI_LIM), the result is scaled inversely proportional to the input value.

Syntax

The following syntax is used for the "Scale" instruction:

GRAPH

```
CALL SCALE
      (IN := <Operand>
       HI_LIM := <Operand>
       LO_LIM := <Operand>
       BIPOLAR := <Operand>
       RET_VAL => <Operand>
       OUT => <Operand>
       )
```

Parameters

The following table shows the parameters of the "Scale" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	INT	I, Q, M, D, L, P, or constant	Input value to be scaled
HI_LIM	Input	REAL	I, Q, M, D, L, P, or constant	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P, or constant	Low limit
BIPOLAR	Input	BOOL	I, Q, M, D, L, or constant	Indicates whether the value of the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
OUT	Output	REAL	I, Q, M, D, L, P	Result of the instruction
RET_VAL	Output	WORD	I, Q, M, D, L, P	Error information

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the constant "K2" or less than the value of the constant "K1"
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

GRAPH

```
CALL SCALE
     (IN := "Tag_InputValue"
      HI_LIM := "Tag_HighLimit"
      LO_LIM := "Tag_LowLimit"
      BIPOLAR := "Tag_Bipolar"
```

GRAPH

```
RET_VAL => "Tag_ErrorCode"
OUT => "Tag_OutputValue"
)
```

The following table shows the values of the various operands before execution of the instruction:

Parameter	Operand	Value
IN	Tag_InputValue	22
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	0.0

The following table shows the values of the various operands after execution of the instruction:

Parameter	Operand	Value
IN	Tag_InputValue	22
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	50.03978588

The "Tag_InputValue" operand indicates the value that is to be converted and scaled. The high and low limits are defined by the "Tag_HighLimit" and "Tag_LowLimit" operands. You use operand "Tag_Bipolar" = TRUE to specify that the value of the IN parameter is to be interpreted as bipolar. The result of the instruction is output at the "Tag_OutputValue" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

UNSCALE: Unscale

Description

You use the "Unscale" instruction to unscale the floating-point number in the IN parameter into physical units between a low limit and a high limit and convert it into an integer. You use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is unscaled. The result of the instruction is output at the OUT parameter.

The "Unscale" instruction works with the following equation:

OUT = [((IN–LO_LIM)/(HI_LIM–LO_LIM)) * (K2–K1)] + K1

The values of the constants "K1" and "K2" are determined by the signal state of the BIPOLAR parameter. The following signal states are possible at the BIPOLAR parameter:

- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case the "K1" constant has the value "-27648.0" and the "K2" constant the value "+27648.0".
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case the "K1" constant has the value "0.0" and the "K2" constant the value "+27648.0".

When the value of the IN parameter is greater than the value of the constant "HI_LIM", the result of the instruction is set to the value of the constant (K2) and an error is output.

When the value of the IN parameter is less than the value of the constant of the low limit (LO_LIM), the result of the instruction is set to the value of the constant (K1) and an error is output.

Syntax

The following syntax is used for the "Unscale" instruction:

GRAPH

```
CALL UNSCALE
      (IN := <Operand>
       HI_LIM := <Operand>
       LO_LIM := <Operand>
       BIPOLAR := <Operand>
       RET_VAL => <Operand>
       OUT => <Operand>
       )
```

Parameters

The following table shows the parameters of the "Unscale" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	REAL	I, Q, M, D, L, P, or constant	Input value to be unscaled to an integer value
HI_LIM	Input	REAL	I, Q, M, D, L, P, or constant	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P, or constant	Low limit

Parameter	Declaration	Data type	Memory area	Description
BIPOLAR	Input	BOOL	I, Q, M, D, L or constant	Indicates whether the value of the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
OUT	Output	INT	I, Q, M, D, L, P	Result of the instruction
RET_VAL	Output	WORD	I, Q, M, D, L, P	Error information

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
0008	The value of the IN parameter is greater than the value of the high limit (HI_LIM) or less than the value of the low limit (LO_LIM).
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

GRAPH

```
CALL UNSCALE
      (IN := "Tag_InputValue"
       HI_LIM := "Tag_HighLimit"
       LO_LIM := "Tag_LowLimit"
       BIPOLAR := "Tag_Bipolar"
       RET_VAL => "Tag_ErrorCode"
       OUT => "Tag_OutputValue"
       )
```

The following table shows the values of the various operands before execution of the instruction:

Parameter	Operand	Value
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	0.0

The following table shows the values of the various operands after execution of the instruction:

Parameter	Operand	Value
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	22

The "Tag_InputValue" operand indicates the value that is to be converted and unscaled. The high and low limits are defined by the "Tag_HighLimit" and "Tag_LowLimit" operands. You use operand "Tag_Bipolar" = TRUE to specify that the value of the IN parameter is to be interpreted as bipolar. The result of the instruction is output at the "Tag_OutputValue" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

Program control operations

Runtime control

COMPRESS: Compress CPU memory

Description

You use the "Compress CPU memory" instruction to compress the work memory of the CPU or the RAM load memory. During this compression the gaps caused by deleting or loading blocks are filled by pushing together the existing blocks.

The compression operation is started by calling the "Compress CPU memory" instruction. The compression operation is divided into several program cycles. The compression operation is monitored using the values of the individual parameters. If the BUSY parameter has the value "1", the compression operation is still being executed. The value "1" in the DONE parameter

indicates that the compression operation is complete. Error information can be queried in the RET_VAL parameter.

If an externally started compression operation is already being executed, the calling of the "Compress CPU memory" instruction causes an error to be displayed.

Syntax

The following syntax is used for the "Compress CPU memory" instruction:

GRAPH

```
CALL COMPRESS
     (RET_VAL => <Operand>
      BUSY => <Operand>
      DONE => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Compress CPU memory" instruction:

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Output	INT	I, Q, M, D, L	Error information: If an error occurs during execution of the instruction, an error code is output in the RET_VAL parameter.
BUSY	Output	BOOL	I, Q, M, D, L	Information on whether compression operation is active
DONE	Output	BOOL	I, Q, M, D, L	Information on whether compression operation is complete

For additional information on valid data types, refer to "See also".

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error The status of the compression operation can be queried in the BUSY and DONE parameters.
8091	An externally started compression operation is already being executed.

Error code* (W#16#...)	Explanation
8092	The instruction cannot be executed, because one of the following conditions applies: • The "Delete block" function is active. • A test and commissioning function is acting on a block (for example, status). • The "copy blocks" function was triggered externally and is active. • The H-CPU is running a link-up or update function.
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

RE_TRIGR: Restart cycle monitoring time

Description

You use the "Restart cycle monitoring time" instruction to restart the cycle monitoring time of the CPU. The maximum cycle time then starts over again for the duration you have set in the CPU configuration. By restarting the cycle monitoring time, you can prevent errors being triggered or the CPU changing to STOP.

The "Restart cycle monitoring time" instruction can be used in blocks of priority class 1 (the cyclic OB) and in the blocks called within it.

If the instruction is called in a block with a higher priority, such as a hardware interrupt, diagnostic error interrupt, or cyclic interrupt, the instruction is not executed and the ENO enable output is set to signal state "0".

Note

Make sure that you do not create an infinite loop in the cyclical program processing, i.e., in OB1, when you use the "Restart cycle monitoring time" instruction. Otherwise the CPU will not reach the cycle control point. As a result, it may not be possible to execute certain CPU functions (e.g., process image update).

Syntax

The following syntax is used for the "Restart cycle monitoring time" instruction:

GRAPH

```
CALL RE_TRIGR()
```

Parameters

The "Restart cycle monitoring time" instruction has no parameters and supplies no error information.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

STP: Exit program

Description

You use the "Exit program" instruction to set the CPU to STOP mode and therefore to terminate the execution of the program. The effects of changing from RUN to STOP depend on the CPU configuration.

Syntax

The following syntax is used for the "Exit program" instruction:

GRAPH

```
CALL STP()
```

Parameters

The "Exit program" instruction has no parameters and supplies no error information.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

WAIT: Configure time delay

Description

You use the "Configure time delay" instruction to halt the execution of the program for a set period of time. You indicate the period of time in microseconds in the WT parameter.

You can configure time delays of up to 32767 microseconds (µs). The smallest possible delay time depends on the particular CPU and corresponds to the execution time of the "Configure time delay" instruction.

The execution of the instruction can be interrupted by higher priority events.

The "Configure time delay" instruction supplies no error information.

Syntax

The following syntax is used for the "Configure time delay" instruction:

GRAPH

```
CALL WAIT
     (WT := <Operand>
     )
```

Parameters

The following table shows the parameters of the "Configure time delay" instruction:

Parameter	Declaration	Data type	Memory area	Description
WT	Input	INT	I, Q, M, D, L, P, or constant	Time delay in microseconds (µs)

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

PROTECT: Change protection level

Description

You use the "Change protection level" instruction to change the configured protection level "1" of the CPU.

The assigned protection level "1" with selected option "Removable with password" means that you have to assign a password. This password enables you to write-access the CPU that was write-protected with the "Change Protection Level" instruction during operation (e.g., for loading).

You specify the protection level in the MODE parameter of the instruction. The current protection level of your CPU is displayed in the inspector window under "Properties > Protection".

The following table shows the meaning of the protection levels.

Protection level	Meaning
1	All programming device functions are permitted. Any existing blocking of the password authentication is canceled by calling the "Change protection level" instruction.
2	• The program and the configuration of the CPU cannot be changed. The program in the CPU can be read. • The functions for process control, process monitoring, and process communication are permitted. • All informational functions are permitted. • If you know the valid password you can cancel the set write protection. • Any existing blocking of the password authentication is canceled by calling the "Change protection level" instruction.
3	• You also cannot cancel the set read and write protection, even if you know the valid password. • If a valid connection exists at the time when the "Change protection level" instruction is called, the call of the instruction does not affect this connection.

If you have configured protection level "2" or "3" for your CPU, calling the "Change protection level" instruction has no effect.

You can use the SSL parts list with the SSL ID W#16#0232 and index W#16#0004 to check whether protection level "2" or "3" is still in effect after execution of the "Change protection level" instruction.

Note

You can only use the "Change protection level" instruction to set a protection level that is equal to or higher than the protection level you have configured with STEP 7 "Configure hardware". You have, for example, configured protection level 3 and then called the "Change protection level" instruction with parameter MODE = 12. This blocks the password authentication. If the instruction is called again with parameter MODE = 0 or 1, protection level 3 is retained and the read and write protection is canceled by entry of the valid password.

Effects of the operating state transitions on the protection level

The following table shows what effects the operating state transitions have on the protection level set with the "Change protection level" instruction.

Action	Effects on the protection level
• Operating mode switch to "STOP" • Unbuffered POWER ON • Memory reset with operating mode switch • Configuration and/or programming changes in "STOP" mode of the S7-400 CPU	The protection level is set to "1". All programming device functions are permitted.
• Cold restart • Warm restart	The protection level is set by the operating system to "1". All programming device functions are permitted. If required, you can set the protection level in your program to "2" or "3" by calling the "Change protection level" instruction.
Buffered POWER ON	Remains unchanged

Action	Effects on the protection level
Operating mode transition RUN/STARTUP/HOLD -> STOP (by calling the "Exit program" instruction (STP) due to program error without corresponding error OB, or by operator action)	Remains unchanged
S7-400: Restart by user action or automatically following POWER ON	Remains unchanged

Syntax

The following syntax is used for the "Change protection level" instruction:

GRAPH

```
CALL PROTECT
      (MODE := <Operand>
       RET_VAL => <Operand>
       )
```

Parameters

The following table shows the parameters of the "Change protection level" instruction:

Parameter	Declaration	Data type	Memory area	Description
MODE	Input	WORD	I, Q, M, D, L, or constant	Protection level Possible values: • W#16#0000: Setting protection level "1" • W#16#0001: Setting protection level "2" with password authentication • W#16#000C: Setting protection level "3" without password authentication Constants can also be specified for the MODE parameter.
RET_VAL	Output	INT	I, Q, M, D, L	If an error occurs during processing, information on the error is output.

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	Job executed without error
8090	Invalid value at parameter MODE.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

CiR: Control CiR process

Description

You use the "Control CiR process" instruction to influence the CiR process in the user program as follows:

- Disable CiR process. In this case, the download of a modified configuration from the programming device to the CPU is always rejected. The disable is maintained until you cancel it again with the "Control CiR process" instruction.
- You can disable the CiR process conditionally by specifying a high limit for the CiR synchronization time in the FRZ_TIME parameter. In this case, the download of a modified configuration from the programming device to the CPU is only accepted if the CiR synchronization time calculated in the CPU does not exceed the specified high limit.
- Enable CiR process. The high limit of the CiR synchronization time is set to the default value 1000 ms.
- Determine whether the CiR process is enabled or not. If the CiR process is enabled or conditionally enabled, you will get the current high limit for the CiR synchronization time in the A_FT parameter of the instruction.

Note

During the CiR synchronization period the outputs are frozen and the inputs are not evaluated.

Syntax

The following syntax is used for the "Control CiR process" instruction:

GRAPH

```
CALL CIR
     (MODE := <Operand>
      FRZ_TIME := <Operand>
      RET_VAL => <Operand>
      A_FT => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Control CiR process" instruction:

Parameter	Declaration	Data type	Memory area	Description
MODE	Input	BYTE	I, Q, M, D, L, or constant	Job identifier Valid values for the MODE parameter are: • 0: Information function • 1: Enable CiR process • 2: Disable CiR process • 3: Disable CiR conditionally. You specify the high limit of the CiR synchronization time in FRZ_TIME.
FRZ_TIME	Input	TIME	I, Q, M, D, L, or constant	High limit of the CiR synchronization time in milliseconds. Permitted range of values: 60 ... 2500 ms (default: 1000 ms) The FRZ_TIME parameter is only relevant when the value "3" is set in the MODE parameter.
RET_VAL	Output	INT	I, Q, M, D, L	Error information When the MODE parameter has the value "0", the RET_VAL parameter supplies information on the enabling of the CiR process.
A_FT	Output	TIME	I, Q, M, D, L	Currently valid high limit of the CiR synchronization time

For additional information on valid data types, refer to "See also".

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	Job executed without error This error code can be output only if the value "1", "2" or "3" is set in the MODE parameter.
0001	CiR is enabled. This error code can be output only if the value "0" is set in the MODE parameter.
0002	The CiR process is disabled. This error code can be output only if the value "0" is set in the MODE parameter.
0003	CiR is disabled conditionally. This error code can be output only if the value "0" is set in the MODE parameter.
8001	The CPU is not ready for a CiR operation. You are using an H CPU in an H-system (stand-alone), or you are working with a standard CPU in multicomputing mode.
8002	The value of the MODE parameter is invalid.
8003	The value of the FRZ_TIME parameter is invalid.
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

Example

You can use the "Control CiR process" instruction, for example, to ensure that no CiR process is initiated during periods in which maximum CPU performance is required for the execution of the process.

To use the "Control CiR process" instruction for this, the instruction must be called in the program before the start of the period with increased process activity and executed with the value "2" (disable CiR process) in the MODE parameter. The CiR process is then disabled.

To enable the CiR process again after the completion of the period with increased process activity, you must call the "Control CiR process" instruction again and execute it with the value "1" in the MODE parameter. Alternatively, you can also set the value "3" in the MODE parameter to disable the CiR process conditionally.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

Word logic operations

NOT (INV): Inverting

Description

You can use the instruction "Create ones complement" to invert the signal status of the bits of the operand. When the instruction is executed, the value of the operand is linked by EXCLUSIVE OR to a hexadecimal mask (W#16#FFFF for 16-bit numbers or DW#16#FFFF FFFF for 32-bit numbers). This inverts the signal state of the individual bits that are then output as the result.

Syntax

The following syntax is used for the "Invert" instruction:

GRAPH

```
<Result> := NOT(<Operand>)
```

Parameters

The following table shows the parameters of the instruction "Create ones complement":

Parameters	Declaration	Data type	Memory area	Description
<Operand>	Input	Bit strings, integers	I, Q, M, D, L, P or constant	Input value
<Result>	Output	Bit strings, integers	I, Q, M, D, L, P	Ones complement of the operand value

For more information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"TagOut_Value" := NOT("TagIn_Value")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
TagIn_Value	W#16#000F
TagOut_Value	W#16#FFF0

The instruction inverts the signal state of the individual bits at "TagIn_Value" operand and writes the result to the "TagOut_Value" operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

DECO: Decode

Description

You use the "Decode" instruction to set a bit in the output value that is specified by the input value .

The "Decode" instruction reads the value of the IN input and sets the bit in the OUT parameter whose bit position corresponds to the value read. The other bits in the output value are filled with zeroes. If the value of the IN input is greater than 31, a modulo 32 instruction is executed.

Syntax

The following syntax is used for the "Decode" instruction:

GRAPH

```
CALL DECO <DataType>_DWORD
     (IN := <Operand>
      OUT => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Decode" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	WORD	I, Q, M, D, L, P, or constant	Position of the bit in the output value that is set.
OUT	Output	DWORD	I, Q, M, D, L, P	Output value

For additional information on valid data types, refer to "See also".

You can select the data type of the instruction from the "<data type>" drop-down list.

Example

The following example shows how the instruction works:

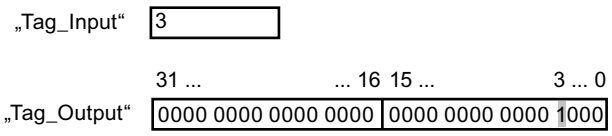
GRAPH

```
CALL DECO INT_DWORD
```

GRAPH

```
(IN := "Tag_Input"
 OUT => "Tag_Output"
 )
```

The following figure shows how the instruction works using specific operand values:



The instruction reads bit number "3" from the value of the "Tag_Input" operand of the input and sets the third bit in the value of the "Tag_Output" operand of the output.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

ENCO: Encode

Description

You use the "Encode" instruction to read the bit number of the least significant set bit in the input value and output it in the OUT output.

The "Encode" instruction selects the least significant bit of the value in the IN input and writes its bit number to the tag in the OUT output. If the IN input has the value DW#16#00000001 or DW#16#00000000, the value "0" is output in the OUT parameter.

Syntax

The following syntax is used for the "Encode" instruction:

GRAPH

```
CALL ENCO DWORD
     (IN := <Operand>
      OUT => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Encode" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	DWORD	I, Q, M, D, L, P, or constant	Input value
OUT	Output	INT	I, Q, M, D, L, P	Output value

For additional information on valid data types, refer to "See also".

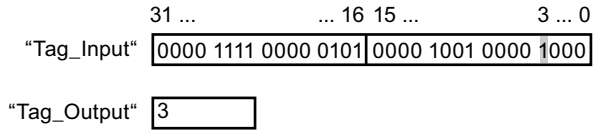
Example

The following example shows how the instruction works:

GRAPH

```
CALL ENCO DWORD
      (IN := "Tag_Input"
       OUT => "Tag_Output"
       )
```

The following figure shows how the instruction works using specific operand values:



The instruction selects the least significant set bit in the "TagInput" input and writes bit position "3" to the tag in the "Tag_Output" output.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

SEL: Select

Description

You use the "Select" instruction to select, dependent on a switch (G input) one of two inputs, IN0 or IN1, and move its content to the OUT output. If the G input has the signal state "0", the value of the IN0 input is moved. If the G input has the signal state "1", the value of the IN1 input is moved to the OUT output.

The instruction is only executed if the tags of all parameters are of the same data type.

Syntax

The following syntax is used for the "Select" instruction:

GRAPH

```
CALL SEL <DataType>
     (G := <Operand>
      IN0 := <Operand>
      IN1 := <Operand>
      OUT => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Select" instruction:

Parameter	Declaration	Data type	Memory area	Description
G	Input	BOOL	I, Q, M, D, L, T, C	Switch
IN0	Input	Bit strings, integers, floating-point numbers, timers, CHAR, TOD, DATE	I, Q, M, D, L, P	First input value
IN1	Input	Bit strings, integers, floating-point numbers, timers, CHAR, TOD, DATE	I, Q, M, D, L, P	Second input value
OUT	Output	Bit strings, integers, floating-point numbers, timers, CHAR, TOD, DATE	I, Q, M, D, L, P	Result

You can select the data type of the instruction from the "<data type>" drop-down list.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL SEL INT
     (G := "Tag_Input_G"
      IN0 := "Tag_Input0"
      IN1 := "Tag_Input1"
      OUT => "Tag_Output"
      )
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
G	Tag_Input_G	0	1
IN0	Tag_Input0	W#16#0000	W#16#4C
IN1	Tag_Input1	W#16#FFFF	W#16#5E
OUT	Tag_Output	W#16#0000	W#16#5E

Based on the signal state of the "Tag_Input_G" input, the value of the "Tag_Input0" or "Tag_Input1" input is selected and moved to the "Tag_Output" output.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

Shift and rotate

SHR: Shift right

Description

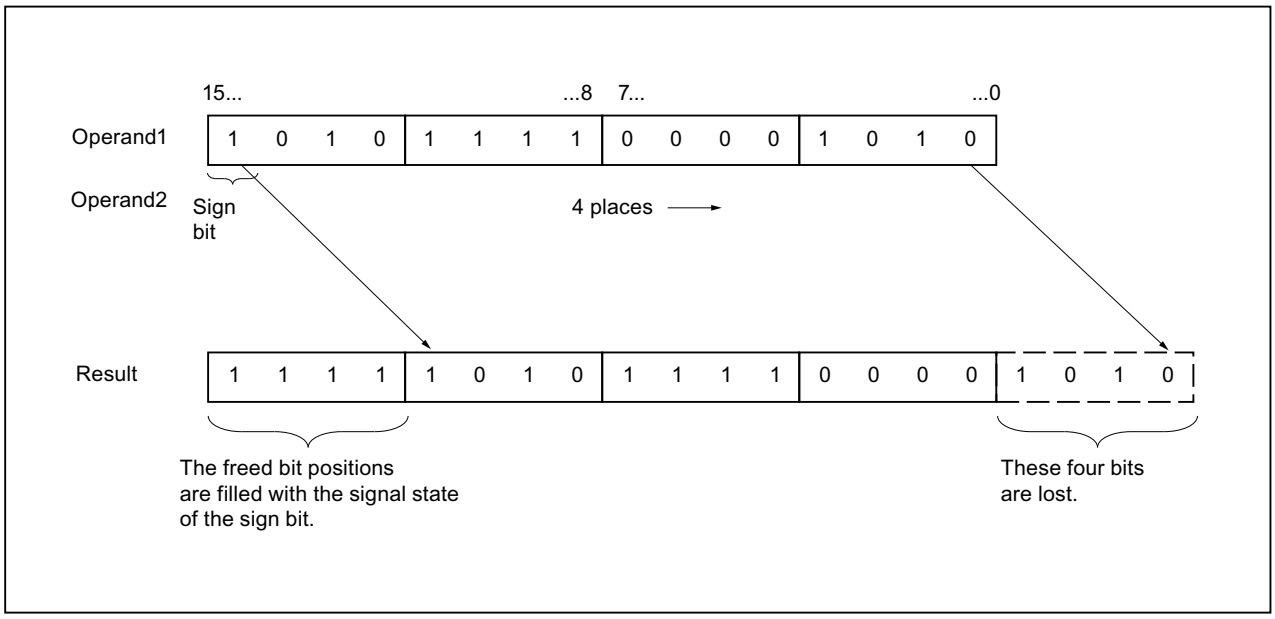
You use the "Shift right" instruction to shift the content of Operand1 to the right bitwise. You use Operand2 to specify the number of bit positions the specified value will be shifted.

If the value of Operand2 is "0", the value of Operand1 is copied to the result.

If the value of Operand2 is greater than the number of available bit positions, the value of Operand1 is shifted to the right by the available number of bit positions.

When unsigned values are shifted, the free bit positions in the left area of Operand1 are filled by zeroes. If the specified value has a sign, the free bit positions are filled with the signal state of the sign bit.

The following figure shows how the content of Operand1 of data type INT is shifted by four bit positions to the right:



Syntax

The following syntax is used for the "Shift right" instruction:

GRAPH

```
<Result> := SHR_<DataType>(<Operand1>,<Operand2>)
```

Parameters

The following table shows the parameters of the "Shift right" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	Bit strings, integers	I, Q, M, D, L, P or constant	Value to be shifted
<Operand2>	Input	BYTE, WORD, INT	I, Q, M, D, L, P or constant	Number of bit positions by which the value is shifted
<Result>	Output	Bit strings, integers	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "<data type>" drop-down list.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := SHR_INT("Tag_InValue","Tag_Number")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	0011 1111 1010 1111
Tag_Number	3
Tag_OutValue	0000 0111 1111 0101

The content of the "Tag_InValue" operand is shifted three bit positions to the right and output as the result in the Tag_OutValue operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

SHL: Shift left

Description

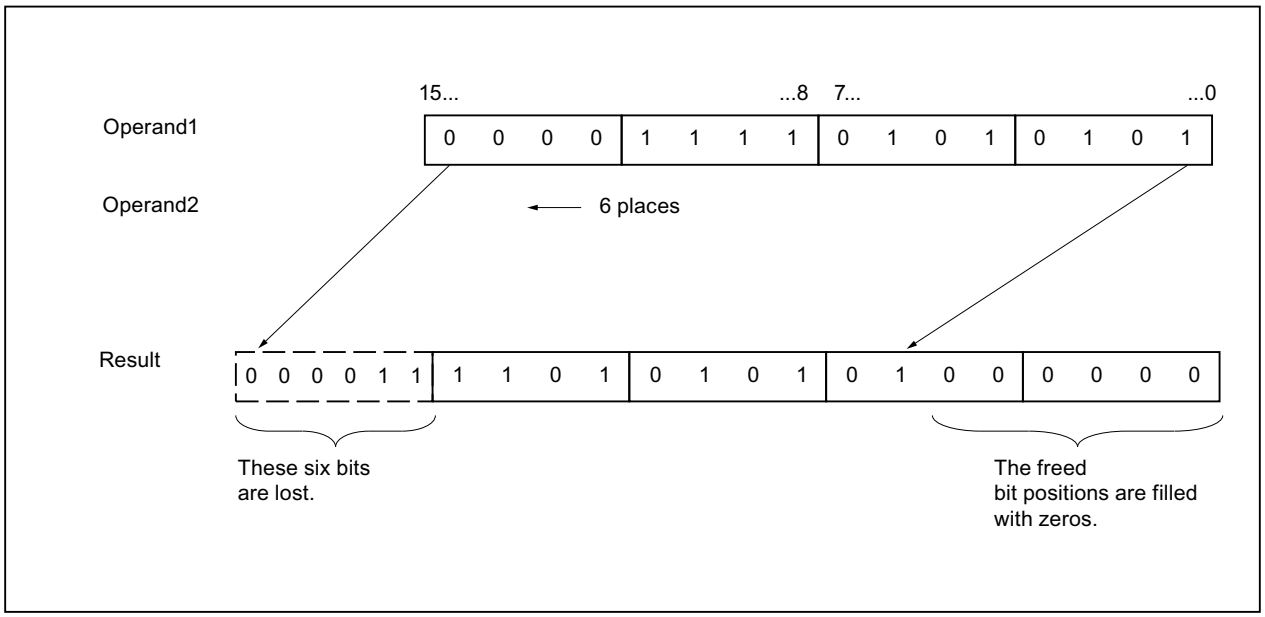
You use the "Shift left" instruction to shift the content of Operand1 to the left bitwise. You use Operand2 to specify the number of bit positions the specified value will be shifted.

If the value of Operand2 is "0", the value of Operand1 is copied to the result.

If the value of Operand2 is greater than the number of available bit positions, the value of Operand1 is shifted to the left by the available number of bit positions.

The bit positions in the right part of Operand1 freed by shifting are filled with zeros.

The following figure shows how the content of Operand1 of data type WORD is shifted by six bit positions to the left:



Syntax

The following syntax is used for the "Shift left" instruction:

GRAPH

```
<Result> := SHL_<DataType>(<Operand1>,<Operand2>)
```

Parameters

The following table shows the parameters of the "Shift left" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	Bit strings, integers	I, Q, M, D, L, P or constant	Value to be shifted
<Operand2>	Input	BYTE, WORD, INT	I, Q, M, D, L, P or constant	Number of bit positions by which the value is shifted
<Result>	Output	Bit strings, integers	I, Q, M, D, L	Result of the instruction

You can select the data type of the instruction from the "<data type>" drop-down list.

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := SHL_WORD("Tag_InValue","Tag_Number")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	0011 1111 1010 1111
Tag_Number	4
Tag_OutValue	1111 1010 1111 0000

The content of the "Tag_InValue" operand is shifted four bit positions to the left and output as the result in the Tag_OutValue operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

ROR: Rotate right

Description

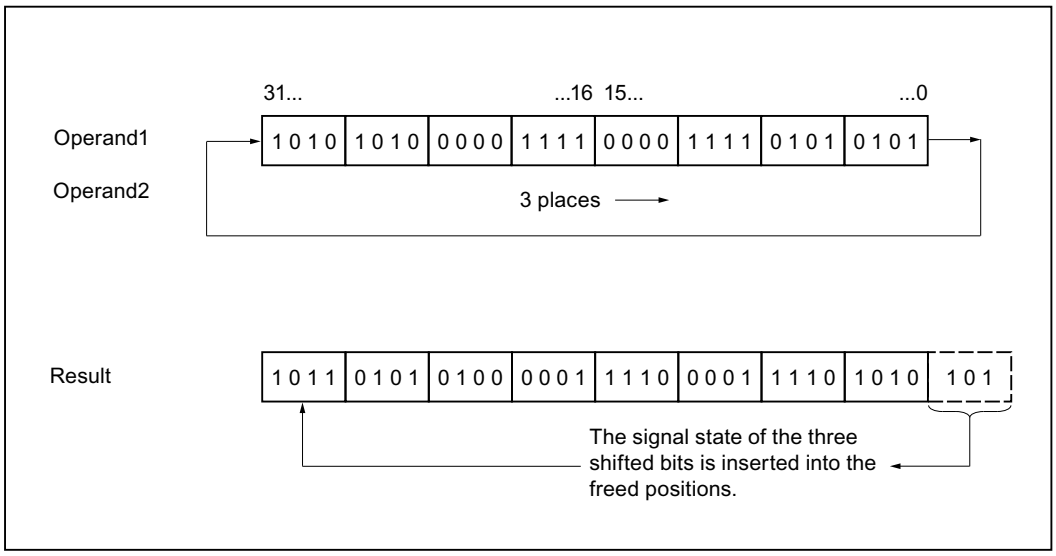
You use the "Rotate right" instruction to rotate the content of Operand1 to the right bitwise. You use Operand2 to specify the number of bit positions the specified value will be rotated.

If the value of Operand2 is "0", the value of Operand1 is copied to the result.

If the value of Operand2 is greater than the number of available bit positions, the value of Operand1 is rotated by the specified number of bit positions.

The bit positions freed by rotating are filled with the bit positions that are pushed out.

The following figure shows how the content of an operand of data type DWORD is rotated three bit positions to the right:



Syntax

The following syntax is used for the "Rotate right" instruction:

GRAPH

```
<Result> := ROR(<Operand1>,<Operand2>)
```

Parameters

The following table shows the parameters of the "Rotate right" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	BYTE, WORD, DWORD, DINT	I, Q, M, D, L, P or constant	Value to be rotated
<Operand2>	Input	BYTE, WORD, INT	I, Q, M, D, L, P or constant	Number of bit positions by which the value is rotated
<Result>	Output	BYTE, WORD, DWORD, DINT	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
"Tag_OutValue" := ROR("Tag_InValue", "Tag_Number")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	1010 1010 0000 1111 0000 1111 0101 0101
Tag_Number	5
Tag_OutValue	1010 1101 0101 0000 0111 1000 0111 1010

The content of the "Tag_InValue" operand is rotated five bit positions to the right and output as the result in the Tag_OutValue operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

ROL: Rotate left

Description

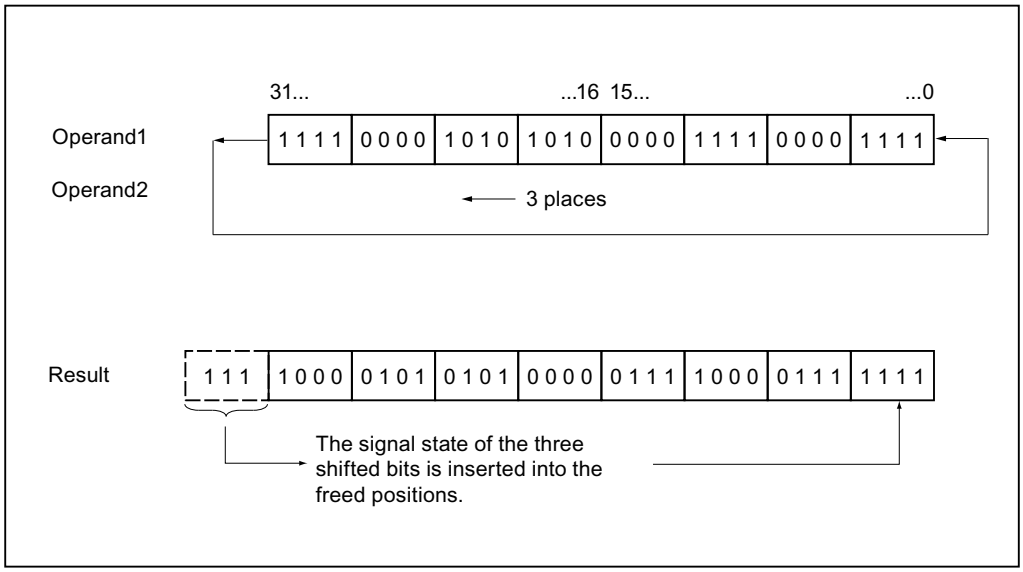
You use the "Rotate left" instruction to rotate the content of Operand1 to the left bitwise. You use Operand2 to specify the number of bit positions the specified value will be rotated.

If the value of Operand2 is "0", the value of Operand1 is copied to the result.

If the value of Operand2 is greater than the number of available bit positions, the value of Operand1 is rotated by the specified number of bit positions.

The bit positions freed by rotating are filled with the bit positions that are pushed out.

The following figure shows how the content of an operand of data type DWORD is rotated three bit positions to the left:



Syntax

The following syntax is used for the "Rotate left" instruction:

GRAPH
```
<Result> := ROL(<Operand1>,<Operand2>)
```

Parameters

The following table shows the parameters of the "Rotate left" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	BYTE, WORD, DWORD, DINT	I, Q, M, D, L, P or constant	Value to be rotated
<Operand2>	Input	BYTE, WORD, INT	I, Q, M, D, L, P or constant	Number of bit positions by which the value is rotated
<Result>	Output	BYTE, WORD, DWORD, DINT	I, Q, M, D, L	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH
```
"Tag_OutValue" := ROL("Tag_InValue","Tag_Number")
```

The following table shows how the instruction works using specific operand values:

Operand	Value
Tag_InValue	1111 0000 1010 1010 0000 1111 0000 1111
Tag_Number	5
Tag_OutValue	0001 0101 0100 0001 1110 0001 1111 1110

The content of the "Tag_InValue" operand is rotated five bit positions to the left and output as the result in the Tag_OutValue operand.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

Additional instructions

SET: Set bit array

Description

You use the "Set bit array" instruction to set the signal state of the bits in a specified area to "1". You use the pointer of the S_BIT parameter to define the starting point of the area. If the pointer points to the memory area of the external I/O, the instruction is not executed. You specify the number of bits to be set in the specified area in the N parameter. When the N parameter has the value "0", calling the instruction has no effect.

The instruction is only executed when the MCR bit is "1". When the MCR bit has the signal state "0", the bits in the specified area are not changed.

Syntax

The following syntax is used for the "Set bit array" instruction:

GRAPH

```
CALL SET
      (S_BIT := <Operand>
       N := <Operand>
       )
```

Parameters

The following table shows the parameters of the "Set bit array" instruction:

Parameter	Declaration	Data type	Memory area	Description
S_BIT	Input	POINTER	I, Q, M, D	Points to the first bit of the area whose bits are set. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	I, Q, M, D, L, P, or constant	Number of bits to be set

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL SET
     (S_BIT := P#M0.0
      N := 10
      )
```

The instruction sets 10 bits in the area from M0.0 to M1.2.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

SETP: Set a bit array in the I/O area

Description

You use the "Set a bit array in the I/O area" instruction to set the bits of an I/O area to signal state "1". In parallel with the I/O bits, the corresponding bits in the process image output are set simultaneously.

You use the N and SA parameters to define the bit array that will be set. The bits of the array must be located in the part of the I/O area to which a process image is assigned.

If no I/O devices are plugged in for part of the selected bit array, the instruction attempts to set the entire bit array. In this case error information is output in the RET_VAL parameter.

The I/O area is set byte-by-byte. If the bit array defined via the N and SA parameters does not begin or end at a byte boundary, calling the instruction has the following effect:

- The bits that are in the first or last byte to be transferred and that do not belong to the defined bit array, get the value of the corresponding bits in the process image output.

  WARNING

  Unintended consequences

  The setting of the bits to the value of the associated bits in the process image output can lead to unintended reactions such as starting a motor or turning off a cooling system.

- The bits belonging to the defined bit array are set to the signal state "1".

The "Set a bit array in the I/O area" instruction must be called in an MCR area.

Calling the "Set a bit array in the I/O area" instruction has no effect if one of the following conditions applies:

- The N parameter has the value "0"
- The MCR bit has the signal state "1".

Syntax

The following syntax is used for the "Set a bit array in the I/O area" instruction:

GRAPH

```
CALL SETP
      (N := <Operand>
       RET_VAL => <Operand>
       SA => <Operand>
       )
```

Parameters

The following table shows the parameters of the "Set a bit array in the I/O area" instruction:

Parameter	Declaration	Data type	Memory area	Description
N	Input	INT	I, Q, M, D, L, or constant	Number of bits to be set
RET_VAL	Output	INT	I, Q, M, D, L	Error information
SA	Output	POINTER	P	Pointer to the first bit to be set

For additional information on valid data types, refer to "See also".

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

GRAPH

```
CALL SETP
      (N := 8
       RET_VAL => "Tag_Error"
```

GRAPH

```
SA => P#10.0
)
```

The instruction sets 8 bits in the I/O area corresponding to outputs Q10.0 to Q10.7.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

SETI: Set byte array

Description

You use the "Set byte array" instruction to set the signal state of the bit in a specified area of bytes to "1". You define the start point of the area in the S_BYTE. You define the size of the area by specifying the number of bits to be set in the N parameter.

Note

The value of the N parameter must be a multiple of the number eight (for example, 8, 16, 24, etc.).

The pointer specified in the S_BYTE parameter must reference the external I/O memory area (P). Since the "P" memory area is accessed as bytes, words, or double words, the pointer of the S_BYTE parameter must point to an operand that corresponds to a byte limit value, i.e., the bit number of the pointer must be "0".

Note

The signal state of the corresponding bits in the process image output is reset to "0".

Calling the "Set byte array" instruction has no effect if one of the following conditions applies:

- The pointer of the S_BYTE parameter does not point to the external I/O memory area.
- The pointer of the S_BYTE parameter points to an operand that is not byte-aligned.
- The value of the N parameter is not a multiple of the number 8.
- The MCR bit has the signal state "0".

Syntax

The following syntax is used for the "Set byte array" instruction:

GRAPH

```
CALL SETI
      (S_BYTE := <Operand>
       N := <Operand>
       )
```

Parameters

The following table shows the parameters of the "Set byte array" instruction:

Parameter	Declaration	Data type	Memory area	Description
S_BYTE	Input	POINTER	P or constant	Points to the first byte of the area whose bits are to be set. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	I, Q, M, D, L, P, or constant	Size of the area to be set The area is specified by the number of bits as a multiple of eight (for example, 8, 16, 24, etc.).

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL SETI
      (S_BYTE := P#2.0
       N := 16
       )
```

The instruction sets 2 bytes (16 bits) in the area from P2.0 to P3.7.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

RESET: Reset bit array

Description

You use"Reset bit array" instruction to reset the signal state of the bits in a specified area to "0". You use the pointer of the S_BIT parameter to define the starting point of the area. If the pointer points to the memory area of the external I/O, the instruction is not executed. You specify the number of bits to be reset in the specified area in the N parameter. When the N parameter has the value "0", calling the instruction has no effect.

The instruction is only executed when the MCR bit is "1". When the MCR bit has the signal state "0", the bits in the specified area are not changed.

Syntax

The following syntax is used for the "Reset bit array" instruction:

GRAPH

```
CALL RESET
      (S_BIT := <Operand>
       N := <Operand>
       )
```

Parameters

The following table shows the parameters of the "Reset bit array" instruction:

Parameter	Declaration	Data type	Memory area	Description
S_BIT	Input	POINTER	I, Q, M, D	Points to the first bit of the area whose bits are to be reset For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	I, Q, M, D, L, P, or constant	Number of bits to be reset

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL RESET
     (S_BIT := P#M0.0
      N := 10
      )
```

The instruction resets 10 bits in the area from M0.0 to M1.2.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

RESETP: Reset a bit array in the I/O area

Description

You use the "Reset a bit array in the I/O area" instruction to reset the bits of an I/O area to the signal state "0". In parallel with the I/O bits, the corresponding bits in the process image output are also reset simultaneously.

You use the N and SA parameters to define the bit array that will be reset. The bits of the array must be located in the part of the I/O area to which a process image is assigned.

If no I/O devices are plugged in for part of the selected bit array, the instruction attempts to reset the entire bit array. In this case error information is output in the RET_VAL parameter.

The I/O area is reset byte-by-byte. If the bit array defined via the N and SA parameters does not begin or end at a byte boundary, calling the instruction has the following effect:

- The bits that are in the first or last byte to be transferred and that do not belong to the defined bit array, get the value of the corresponding bits in the process image output.

  WARNING

  Unintended consequences

  The setting of the bits to the value of the associated bits in the process image output can lead to unintended reactions such as starting a motor or turning off a cooling system.

- The bits belonging to the defined bit array are reset to the signal state "0".

The "Reset a bit array in the I/O area" instruction must be called in an MCR range.

Calling the "Reset a bit array in the I/O area" instruction has no effect if one of the following conditions applies:

- The N parameter has the value "0"
- The MCR bit has the signal state "1".

Syntax

The following syntax is used for the "Reset a bit array in the I/O area" instruction:

GRAPH

```
CALL RESETP
      (N := <Operand>
       RET_VAL => <Operand>
       SA => <Operand>
       )
```

Parameters

The following table shows the parameters of the "Reset a bit array in the I/O area" instruction:

Parameter	Declaration	Data type	Memory area	Description
N	Input	INT	I, Q, M, D, L, or constant	Number of bits to be reset
RET_VAL	Output	INT	I, Q, M, D, L	Error information
SA	Output	POINTER	P	Pointer to the first bit to be reset

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Explanation
0000	No error
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

GRAPH

```
CALL RESETP
      (N := 8
       RET_VAL => "Tag_Error"
```

GRAPH

```
SA => P#10.0
)
```

The instruction resets 8 bits in the I/O area corresponding to outputs Q10.0 to Q10.7.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

RESETI: Reset byte array

Description

You use the "Reset byte array" instruction to reset the signal state of the bits in a specified area of bytes to "0". You use the pointer of the S_BYTE parameter to define the starting point of the area. You define the size of the area by specifying the number of bits to be reset in the N parameter.

Note

The value of the N parameter must be a multiple of the number eight (for example, 8, 16, 24, etc.).

The pointer specified in the S_BYTE parameter must point to the external I/O memory area. Since the "P" memory area is accessed as bytes, words, or double words, the pointer of the S_BYTE parameter must point to an operand that corresponds to a byte limit value, i.e., the bit number of the pointer must be "0".

Note

The signal state of the corresponding bits in the process image output is reset to "0".

Calling the "Reset byte array" instruction has no effect if one of the following conditions applies:

- The pointer of the S_BYTE parameter does not point to the external I/O memory area.
- The pointer of the S_BYTE parameter points to an operand that is not byte-aligned.
- The value of the N parameter is not a multiple of the number 8.
- The MCR bit has the signal state "0".

Syntax

The following syntax is used for the "Reset byte array" instruction:

GRAPH

```
CALL RESETI
     (S_BYTE := <Operand>
      N := <Operand>
      )
```

Parameters

The following table shows the parameters of the "Reset byte array" instruction:

Parameter	Declaration	Data type	Memory area	Description
S_BYTE	Input	POINTER	P or constant	Points to the first byte of the area whose bits are to be reset. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	INT	I, Q, M, D, L, P, or constant	Size of the area whose bits are to be reset. The area is specified by the number of bits as a multiple of eight (for example, 8, 16, 24, etc.).

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL RESETI
     (S_BYTE := P#2.0
      N := 16
      )
```

The instruction resets 2 bytes (16 bits) in the area from P2.0 to P3.7.

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

REPL_VAL: Enter substitute value

Description

You use the "Enter substitute value" instruction to enter a substitute value in accumulator 1 (ACCU 1) from a synchronous error organization block.

Note

You can only call the "Enter substitute value" instruction in a synchronous error organization block (OB 121, OB 122).

You use the "Enter substitute value" instruction when you can no longer read any value from a module. The OB 122 is started after each access to this module. By calling the "Enter substitute value" instruction you can cause the substitute value to be loaded in accumulator 1. The program execution is then continued with this substitute value. You can take the required information for the selection of the substitute value from the local tags of OB 122.

Syntax

The following syntax is used for the "Enter substitute value" instruction:

GRAPH

```
CALL REPL_VAL
     (VAL := <Operand>
      RET_VAL => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Enter substitute value" instruction:

Parameter	Declaration	Data type	Memory area	Description
VAL	Input	DWORD	I, Q, M, D, L, or constant	Substitute value
RET_VAL	Output	INT	I, Q, M, D, L	Error information

For additional information on valid data types, refer to "See also".

Parameter RET_VAL

The following table shows the meaning of the values of the RET_VAL parameter:

Error code* (W#16#...)	Description
0000	No error
8080	The instruction was not called from a synchronous error OB.
8xyy	For more information on errors, refer to "See also".
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

DRUM: Implement sequencer

Description

You use the "Implement sequencer" instruction to assign the programmed output bits (OUT1 to OUT16) and the output word (OUT_WORD) with the programmed values of the OUT_VAL parameter of the relevant step. The specific step must thereby satisfy the conditions of the programmed enable mask in the S_MASK parameter while the instruction remains at this step. The instruction advances to the next step when the event for the step is true and the programmed time for the current step elapses, or when the value of the JOG parameter changes from "0" to "1". The instruction is reset when the signal state of the RESET parameter changes to "1". The current step is hereby equated to the preset step (DSP).

The amount of time spent on a step is determined by the product of the preset timebase (DTBP) and the preset count value (S_PRESET) for each step. At the start of a new step, this calculated value is loaded into the DCC parameter, which contains the time remaining for the current step. If, for example the value of the DTBP parameter is "2" and the preset value for the first step is "100" (100 ms), the DCC parameter has the value "200" (200 ms).

A step can be programmed with a time value, an event, or both. Steps that have an event bit and the time value "0" advance to the next step as soon as the signal state of the event bit is "1". Steps that are programmed only with a time value start the time immediately. Steps that are programmed with an event bit and a time value greater than "0" start the time when the signal state of the event bit is "1". The event bits are initialized with a signal state of "1".

When the sequencer is on the last programmed step (LST_STEP) and the time for this step has expired, the signal state of the Q parameter is set to "1"; otherwise it is set to "0". When the Q parameter is set, the instruction remains on the step until it is reset.

In the configurable mask (S_MASK), you can select the individual bits in the output word (OUT_WORD) and set or reset the output bits (OUT1 to OUT16) using the output values

(OUT_VAL). If a bit of the configurable mask is in signal state "1", the OUT_VAL value sets or resets the corresponding bit. If the signal state of a bit of the configurable mask is "0", the corresponding bit is left unchanged. All the bits of the configurable mask for all 16 steps are initialized with a signal state of "1".

The output bit of the OUT1 parameter corresponds to the least significant bit of the output word (OUT_WORD). The output bit of the OUT16 parameter corresponds to the most significant bit of the output word (OUT_WORD).

The "Call options" dialog opens automatically when the instruction is inserted in the program. You will find the created data block in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Implement sequencer" instruction:

GRAPH

```
CALL DRUM, <InstanceDB>
     (RESET := <Operand>
      JOG := <Operand>
      DRUM_EN := <Operand>
      LST_STEP := <Operand>
      EVENT1 - 16 := <Operand>
      OUT1 - 16 => <Operand>
      Q => <Operand>
      OUT_WORD => <Operand>
      ERR_CODE => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Implement sequencer" instruction:

Parameter	Declaration	Data type	Memory area	Description
RESET	Input	BOOL	I, Q, M, D, L	The signal state "1" indicates a reset condition.
JOG	Input	BOOL	I, Q, M, D, L	When the signal state changes from "0" to "1", the instruction advances to the next step.

Parameter	Declaration	Data type	Memory area	Description
DRUM_EN	Input	BOOL	I, Q, M, D, L	The signal state "1" causes the sequencer to continue counting based on the event and time criteria.
LST_STEP	Input	BYTE	I, Q, M, D, L, or constant	Number of the last step programmed.
EVENT(i), 1 ≤ i ≤ 16	Input	BOOL	I, Q, M, D, L	Event bit (i); Initial signal state is "1".
OUT(j), 1 ≤ j ≤ 16	Output	BOOL	I, Q, M, D, L	Output bit (j)
Q	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that the time for the last step has elapsed.
OUT_WORD	Output	WORD	I, Q, M, D, L, P	Word address to which the sequencer writes the output values.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
JOG_HIS	Static	BOOL	I, Q, M, D, L	JOG parameter history bit
EOD	Static	BOOL	I, Q, M, D, L	The signal state "1" indicates that the time for the last step has elapsed.
DSP	Static	BYTE	I, Q, M, D, L, P	Preset step of the sequencer
DSC	Static	BYTE	I, Q, M, D, L, P	Current step of the sequencer
DCC	Static	DWORD	I, Q, M, D, L, P	Current count value of the sequencer
DTBP	Static	WORD	I, Q, M, D, L, P	Preset timebase of the sequencer
PREV_TIME	Static	DWORD	I, Q, M, D, L, or constant	Previous system time
S_PRESET	Static	ARRAY of WORD	I, Q, M, D, L	Preset count value for each step [1 to 16] where one clock pulse = 1 ms.

Parameter	Declaration	Data type	Memory area	Description
OUT_VAL	Static	ARRAY of BOOL	I, Q, M, D, L	Output values for each step [1 to 16, 0 to 15].
S_MASK	Static	ARRAY of BOOL	I, Q, M, D, L	Configurable mask for each step [1 to 16, 0 to 15]. Initial signal states are "1".

For additional information on valid data types, refer to "See also".

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

ERR_CODE*	Explanation
W#16#0000	No error
W#16#000B	The value of the LST_STEP parameter is less than 1 or greater than 16.
W#16#000C	The value of the DSC parameter is less than 1 or greater than the value of the LST_STEP parameter.
W#16#000D	The value of the DSP parameter is less than 1 or greater than the value of the LST_STEP parameter.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

DRUM_X: Implement sequencer

Description

You use the "Implement sequencer" instruction to Implement a sequencer with a maximum of 16 steps. You specify the number of the first step in the DSP parameter. You specify the number of the last step in the LST_STEP parameter.

During each step, all 16 output bits OUT0 to OUT15 and the output parameter OUT_WORD (in which the output bits are collected together) are written. An output bit is assigned either the

corresponding bit of the OUT_VAL array that you specified or the output bit of the preceding step. The value assigned depends on the value of the mask bit in the S_MASK parameter.

Note

The initial setting for the mask bit is 0. If you want to change the current setting of one or more mask bits, you make this change in the instance data block.

The "Implement sequencer" instruction advances to the next step when there is a rising edge in the JOG parameter. If the instruction is already in the last step, a positive edge in the JOG parameter sets the Q and EOD tags. In this case the DCC parameter is set to the value "0". The instruction remains in the last step until the RESET parameter is set to the value "1".

You can also permit a time-dependent advancement to the next step. To do this, the DRUM_EN parameter must be set to the value "1". The transition to the next step is made when the event bit (EVENTi) belonging to the current step is set and the time specified for the current step has expired. The time for a step is the product of the time base (DTBP) and the time factor valid for the current step (S_PRESET).

Note

The execution time (DCC) remaining in the current step is reduced only when the corresponding event bit (EVENTi) is set.

The "Call options" dialog opens automatically when the instruction is inserted in the program. You will find the created data block in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

If the value "1" is set in the RESET parameter when the instruction is called, the sequencer goes to the step whose number is specified in the DSP parameter.

A step can be programmed with a time value, an event, or both. Steps that have an event bit and the time value "0" advance to the next step as soon as the signal state of the event bit is "1". Steps that are programmed only with a time value start the time immediately. Steps that are programmed with an event bit and a time value greater than "0" start the time when the signal state of the event bit is "1".

You can also move the next step at any time via the value in the JOG parameter.

When the instruction is called for the first time, you must set the value "1" in the RESET parameter.

When the sequencer is in the last step and the execution time specified for this step has expired, the Q and EOD parameters are set. The instruction remains in the last step until the RESET parameter changes to the value "1".

The instruction is executed only in the STARTUP and RUN modes.

The operating system resets the "Implement sequencer" instruction during a cold restart. During a warm restart, the instruction is not reset. If you want the "Implement sequencer" instruction to be initialized after a warm restart, you must call the instruction in OB 100 with the value "1" in the RESET parameter.

Syntax

The following syntax is used for the "Implement sequencer" instruction:

GRAPH

```
CALL DRUM_X, <InstanceDB>
     (RESET := <Operand>
      JOG := <Operand>
      DRUM_EN := <Operand>
      LST_STEP := <Operand>
      EVENT1 - 16 := <Operand>
      OUT0 - 15 => <Operand>
      Q => <Operand>
      OUT_WORD => <Operand>
      ERR_CODE => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Implement sequencer" instruction:

Parameter	Declaration	Data type	Memory area	Description
RESET	Input	BOOL	I, Q, M, D, L	The signal state "1" resets the sequencer. When the instruction is called for the first time, the RESET parameter must be set to "1".
JOG	Input	BOOL	I, Q, M, D, L	A positive signal edge (compared to the last call) switches the sequencer to the next step if it is not yet in the last step. The next step is enabled depending on the value you specified in the DRUM_EN parameter.
DRUM_EN	Input	BOOL	I, Q, M, D, L	Control parameter that specifies whether time-dependent switching to the next step is possible. If the value is "1" a time-dependent switching to the next step is possible.
LST_STEP	Input	BYTE	I, Q, M, D, L, or constant	Number of the last step Values from 1 to 16 are permitted.
EVENTi, 1 ≤ i ≤ 16	Input	BOOL	I, Q, M, D, L	Event bit number i (belongs to step i)
OUTj, 0 ≤ j ≤ 15	Output	BOOL	I, Q, M, D, L	Output bit number j (identical to the bit number j j of OUT_WORD)

Parameter	Declaration	Data type	Memory area	Description
Q	Output	BOOL	I, Q, M, D, L	Status parameter that indicates whether the specified execution time of the last step has expired.
OUT_WORD	Output	WORD	I, Q, M, D, L, P	Output bits collected together in a tag
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
JOG_HIS	Static	BOOL	I, Q, M, D, L	Not relevant to user (input parameter JOG of the preceding call)
EOD	Static	BOOL	I, Q, M, D, L	Identical to the output parameter Q
DSP	Static	BYTE	I, Q, M, D, L, P, or constant	Number of the first step Values from 1 to 16 are permitted.
DSC	Static	BYTE	I, Q, M, D, L, P, or constant	Number of the current step
DCC	Static	DWORD	I, Q, M, D, L, P	The execution time still remaining in the current step in milliseconds (ms) (only relevant if DRUM_EN = 1 and the corresponding event bit = 1)
DTBP	Static	WORD	I, Q, M, D, L, P, or constant	The time base valid for all steps in milliseconds (ms)
PREV_TIME	Static	DWORD	I, Q, M, D, L, or constant	(Not relevant for the user: system time of the previous call)
S_PRESET	Static	ARRAY of WORD	I, Q, M, D, L, or constant	One-dimensional array with the time factors for each step. Recommended selection of the indices: [1 to 16]. In this case, S_PRESET [x] has the time factor of step x.

Parameter	Declaration	Data type	Memory area	Description
OUT_VAL	Static	ARRAY of BOOL	I, Q, M, D, L	Two-dimensional array with the values to be output in each step if they have not been masked out using S_MASK. Recommended selection of the indices: [1 to 16, 0 to 15]. In this case, OUT_VAL [x, y] has the value assigned to the output bit OUTy in step x.
S_MASK	Static	ARRAY of BOOL	I, Q, M, D, L	Two-dimensional array with the mask bits for each step. Recommended selection of the indices: [1 to 16, 0 to 15]. In this case, S_MASK [x, y] contains the mask bit for the "y"-th value output in step x. Meaning of the mask bits: • 0: The corresponding value of the previous step is assigned to the relevant output bit. • 1: The corresponding value from OUT_VAL is assigned to the relevant output bit.

For additional information on valid data types, refer to "See also".

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

ERR_CODE* (W#16#...)	Explanation
0000	No error
8081	Invalid value in LST_STEP parameter.
8082	Invalid value in DSC parameter.
8083	Invalid value in DSP parameter.
8084	The product DCC = DTBP * S_PRESET[DSC] exceeds the value 2**31-1 (approximately 24.86 days).
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

DCAT: Discrete control-timer alarm

Description

You use the "Discrete control-timer alarm" to count the time from the point at which the CMD parameter issues the command to open or close. The time is accumulated until the preset time (PT) is exceeded or the information is received that the device was opened or closed (O_FB or C_FB) within the specified time. If the preset time is exceeded before the information on the opening or closing of the device is received, the corresponding alarm is activated. If the signal state of the command input changes before the preset time, the time is restarted.

The "Call options" dialog opens automatically when the instruction is inserted in the program. You will find the created data block in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

The "Discrete control-timer alarm" instruction has the following reactions to the input conditions:

- When the signal state of the CMD parameter changes from "0" to "1", the signal states of the Q, CMD_HIS, ET, OA, and CA parameters are influenced as follows:
  - The Q and CMD_HIS parameters are set to "1".
  - The ET, OA and CA parameters are reset to "0".
- When the signal state of the CMD parameter changes from "1" to "0", the Q, ET, OA, CA, and CMD_HIS parameters are reset to "0".
- When the signal state of the CMD and CMD_HIS parameters is "1" and the O_FB parameter is set to "0", the time difference (ms) since the last execution of the instruction is added to the value of the parameter ET. If the value of the ET parameter exceeds the value of the PT parameter, the signal state of the OA parameter is set to "1". If the value of the ET parameter does not exceed the value of the PT parameter, the signal state of the OA parameter is reset to "0". The value of the CMD_HIS parameter is reset to the value of the CMD parameter.
- If the signal state of the CMD, CMD_HIS, and O_FB parameters are set to "1" and the C_FB parameter has the value "0", the signal state of the OA parameter is set to "0". The value of the ET parameter is set to the value of the PT parameter. If the signal state of the O_FB parameter changes to "0", the alarm is set the next time the instruction is executed. The value of the CMD_HIS parameter is set to the value of the CMD parameter.
- If the CMD, CMD_HIS, and C_FB parameters have the value "0", the time difference (ms) since the last execution of the instruction is added to the value of the ET parameter. If the value of the ET parameter exceeds the value of the PT parameter, the signal state of the CA parameter is reset to "1". If the value of the PT parameter is not exceeded, the CA parameter has the signal state "0". The value of the CMD_HIS parameter is set to the value of the CMD parameter.
- If the CMD, CMD_HIS, and O_FB parameters have the signal state "0" and the C_FB parameter is set to "1", the CA parameter is set to "0". The value of the ET parameter is set to the value of the PT parameter. If the signal state of the C_FB parameter changes to "0", the alarm is set the next time the instruction is executed. The value of the CMD_HIS parameter is set to the value of the CMD parameter.
- If the O_FB and C_FB parameters simultaneously have the signal state "1", the signal states of both alarm outputs are set to "1".

The "Discrete control-timer alarm" instruction supplies no error information.

Syntax

The following syntax is used for the "Discrete control-timer alarm" instruction:

GRAPH

```
CALL DCAT, <InstanceDB>
     (CMD := <Operand>
      O_FB := <Operand>
      C_FB := <Operand>
      Q => <Operand>
      OA => <Operand>
      CA => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Discrete control-timer alarm" instruction:

Parameter	Declaration	Data type	Memory area	Description
CMD	Input	BOOL	I, Q, M, D, L	The signal state "0" indicates the "close" command. The signal state "1" indicates the "open" command.
O_FB	Input	BOOL	I, Q, M, D, L	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L	Feedback input when closing
Q	Output	BOOL	I, Q, M, D, L	Shows the status of the CMD parameter
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
ET	Static	DINT	I, Q, M, D, L	Currently elapsed time, where one clock pulse = 1 ms
PT	Static	DINT	I, Q, M, D, L	Preset time value, where one clock pulse = 1 ms
PREV_TIME	Static	DWORD	I, Q, M, D, L	Previous system time
CMD_HIS	Static	BOOL	I, Q, M, D, L	CMD history bit

For additional information on valid data types, refer to "See also".

Example

In the following example the CMD parameter changes from "0" to "1". After the execution of the instruction, the Q parameter is set to "1" and the two alarm outputs OA and CA have the

signal state "0". The CMD_HIS parameter of the instance data block is set to the signal state "1" and the ET parameter is reset to "0".

Note

You can initialize static parameters in the data block.

GRAPH

```
CALL DCAT, "DCAT_DB"
      (CMD := "Tag_Input_CMD"
       O_FB := "Tag_Input_O_FB"
       C_FB := "Tag_Input_C_FB"
       Q => "Tag_Output_Q"
       OA => "Tag_Output_OA"
       CA => "Tag_Output_CA"
       )
```

The following tables show how the instruction works using specific values.

Before processing

In this example the following values are used for the input and output parameters:

Parameter	Operand	Value
CMD	Tag_Input_CMD	TRUE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
Q	Tag_Output_Q	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the "DCAT_DB" instance data block of the instruction:

Parameter	Address	Value
ET	DBD4	L#12
PT	DBD8	L#222
CMD_HIS	DBX16.0	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
Q	Tag_Output_Q	TRUE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

The following values are saved in the "DCAT_DB" instance data block of the instruction:

Parameter	Address	Value
ET	DBD4	L#0
CMD_HIS	DBX16.0	TRUE

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

MCAT: Motor control-timer alarm

Description

You can use the "Motor control-timer alarm" instruction to accumulate the time from the point in time as of which one of the command inputs (open or close) is activated. The time is accumulated until the preset time is exceeded or the relevant feedback input indicates that the device has executed the requested operation within the specified time. If the preset time is exceeded before the feedback is received, the corresponding alarm is triggered.

The "Call options" dialog opens automatically when the instruction is inserted in the program. You will find the created data block in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Execution of the "Motor control-timer alarm" instruction

The following table shows the reactions of the "Motor control-timer alarm" instruction to the various input conditions:

Input parameters								Output parameters								
ET	O_HIS	C_HIS	O_CMD	C_CMD	S_CMD	O_FB	C_FB	OO	CO	OA	CA	ET	O_HIS	C_HIS	Q	Status
X	1	1	X	X	X	X	X	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	X	1	1	0	0	1	1	PT	0	0	0	Alarm
X	X	X	X	X	1	X	X	0	0	0	0	X	0	0	1	Stop
X	X	X	1	1	X	X	X	0	0	0	0	X	0	0	1	Stop
X	0	X	1	0	0	X	X	1	0	0	0	0	1	0	1	Start opening
<PT	1	0	X	0	0	0	X	1	0	0	0	INC	1	0	1	Open
X	1	0	X	0	0	1	0	0	0	0	0	PT	1	0	1	Opened
>=PT	1	0	X	0	0	0	X	0	0	1	0	PT	1	0	0	Opening alarm
X	X	0	0	1	0	X	X	0	1	0	0	0	0	1	1	Start closing

Input parameters								Output parameters								
<PT	0	1	0	X	0	X	0	0	1	0	0	INC	0	1	1	Close
X	0	1	0	X	0	0	1	0	0	0	0	PT	0	1	1	Closed
>=PT	0	1	0	X	0	X	0	0	0	0	1	PT	0	1	0	Closing alarm
X	0	0	0	0	0	X	X	0	0	0	0	X	0	0	1	Stopped

Legend:	
INC	Add the time difference (ms) since the last processing of the FB to ET
PT	PT is set to the same value as ET
X	Cannot be used
<PT	ET < PT
>=PT	ET >= PT

If the input parameters O_HIS and C_HIS both have the signal state "1", they are immediately set to signal state "0". In this case, the last row in the table (X) mentioned above is valid. Because it is therefore no longer possible to check whether the input parameters O_HIS and C_HIS have the signal state "1", the output parameters are set as follows in this case:

OO = FALSE
CO = FALSE
OA = FALSE
CA = FALSE
ET = PT
Q = TRUE

The "Motor control-timer alarm" instruction supplies no error information.

Syntax

The following syntax is used for the "Motor control-timer alarm" instruction:

GRAPH

```
CALL MCAT, <InstanceDB>
      (O_CMD := <Operand>
       C_CMD := <Operand>
       S_CMD := <Operand>
       O_FB := <Operand>
       C_FB := <Operand>
       OO => <Operand>
       CO => <Operand>
       OA => <Operand>
       CA => <Operand>
       Q => <Operand>
       )
```

Parameters

The following table shows the parameters of the "Motor control-timer alarm" instruction:

Parameter	Declaration	Data type	Memory area	Description
O_CMD	Input	BOOL	I, Q, M, D, L	"Open" command input
C_CMD	Input	BOOL	I, Q, M, D, L	"Close" command input
S_CMD	Input	BOOL	I, Q, M, D, L	"Stop" command input
O_FB	Input	BOOL	I, Q, M, D, L	Feedback input when opening
C_FB	Input	BOOL	I, Q, M, D, L	Feedback input when closing
OO	Output	BOOL	I, Q, M, D, L	"Open" output
CO	Output	BOOL	I, Q, M, D, L	"Close" output
OA	Output	BOOL	I, Q, M, D, L	Alarm output when opening
CA	Output	BOOL	I, Q, M, D, L	Alarm output when closing
Q	Output	BOOL	I, Q, M, D, L	The signal state "0" indicates an error condition.
ET	Static	DINT	I, Q, M, D, L	Currently elapsed time, where one clock pulse = 1 ms
PT	Static	DINT	I, Q, M, D, L	Preset time value, where one clock pulse = 1 ms
PREV_TIME	Static	DWORD	I, Q, M, D, L	Previous system time
O_HIS	Static	BOOL	I, Q, M, D, L	"Open" history bit
C_HIS	Static	BOOL	I, Q, M, D, L	"Close" history bit

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.

GRAPH

```
CALL MCAT, "MCAT_DB"
     (O_CMD := "Tag_Input_O_CMD"
      C_CMD := "Tag_Input_C_CMD"
      S_CMD := "Tag_Input_S_CMD"
      O_FB := "Tag_Input_O_FB"
```

GRAPH

```
C_FB := "Tag_Input_C_FB"
OO => "Tag_OutputOpen"
CO => "Tag_OutputClosed"
OA => "Tag_Output_OA"
CA => "Tag_Output_CA"
Q => "Tag_Output_Q"
)
```

The following tables show how the instruction works using specific values.

Before processing

In this example the following values are used for the input and output parameters:

Parameter	Operand	Value
O_CMD	Tag_Input_O_CMD	TRUE
C_CMD	Tag_Input_C_CMD	FALSE
S_CMD	Tag_Input_S_CMD	FALSE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
OO	Tag_OutputOpen	FALSE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	FALSE

The following values are saved in the "MCAT_DB" instance data block of the instruction:

Parameter	Address	Value
ET	DBD4	L#2
PT	DBD8	L#22
O_HIS	DBX16.0	TRUE
C_HIS	DBX16.1	FALSE

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OO	Tag_OutputOpen	TRUE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	TRUE

The following values are saved in the "MCAT_DB" instance data block of the instruction:

Parameter	Address	Value
ET	DBD4	L#0
O_HIS	DBX16.0	TRUE
CMD_HIS	DBX16.1	FALSE

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

IMC: Compare input bits with the bits of a mask

Description

You use the "Compare input bits with the bits of a mask" instruction to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bits of a mask. Up to 16 steps with masks can be programmed. The value of the IN_BIT0 parameter is compared with the value of the CMP_VAL[x,0] mask, where "x" indicates the step number. In the CMP_STEP parameter, you specify the step number of the mask that is used for the comparison. All programmed values are compared in the same manner. Unprogrammed input bits or unprogrammed bits of the mask have the default signal state FALSE.

If a match is found in the comparison, the signal state of the OUT parameter is set to "1". Otherwise the OUT parameter is set to "0".

If the value of the CMP_STEP parameter is greater than 15, the instruction is not executed. An error message is output in the ERR_CODE parameter.

The "Call options" dialog opens automatically when the instruction is inserted in the program. You will find the created data block in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Compare input bits with the bits of a mask" instruction:

GRAPH

```
CALL IMC, <InstanceDB>
     (IN_BIT0 - 15 := <Operand>
      CMP_STEP := <Operand>
      OUT => <Operand>
      ERR_CODE => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Compare input bits with the bits of a mask" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN_BIT0	Input	BOOL	I, Q, M, D, L	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L	Input bit 2 is compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L	Input bit 12 is compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L	Input bit 15 is compared with bit 15 of the mask.
CMP_STEP	Input	BYTE	I, Q, M, D, L, P, or constant	The step number of the mask used for the comparison.
OUT	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that a match was found. The signal state "0" indicates that no match was found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information
CMP_VAL	Static	ARRAY OF BOOL	I, Q, M, D, L	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

For additional information on valid data types, refer to "See also".

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000A	The value of the CMP_STEP parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

SMC: Compare scan matrix

Description

You use the "Compare scan matrix" instruction to compare the signal state of up to 16 programmed input bits (IN_BIT0 to IN_BIT15) with the corresponding bit of the comparison masks for each step. Processing starts at step 1 and is continued until the last programmed step (LAST) or until a match is found. The input bit of the IN_BIT0 parameter is compared with the value of the CMP_VAL[x,0] mask, where "x" indicates the step number. All programmed values are compared in the same manner. If a match is found the signal state of the OUT parameter is set to "1" and the step number with the matching mask is written in the OUT_STEP parameter. Unprogrammed input bits or unprogrammed bits of the mask have the default signal state FALSE.. If more than one step has a matching mask, only the first one found is indicated in the OUT_STEP parameter. If no match is found, the signal state of the OUT parameter is set to "0". In this case the value of the OUT_STEP parameter is greater by "1" than the value of the LAST parameter.

The "Call options" dialog opens automatically when the instruction is inserted in the program. You will find the created data block in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Compare scan matrix" instruction:

GRAPH

```
CALL SMC, <InstanceDB>
     (IN_BIT0 - 15 := <Operand>
```

GRAPH

```
OUT => <Operand>
OUT_STEP => <Operand>
ERR_CODE => <Operand>
)
```

Parameters

The following table shows the parameters of the "Compare scan matrix" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN_BIT0	Input	BOOL	I, Q, M, D, L	Input bit 0 is compared with bit 0 of the mask.
IN_BIT1	Input	BOOL	I, Q, M, D, L	Input bit 1 is compared with bit 1 of the mask.
IN_BIT2	Input	BOOL	I, Q, M, D, L	Input bit 2 is compared with bit 2 of the mask.
IN_BIT3	Input	BOOL	I, Q, M, D, L	Input bit 3 is compared with bit 3 of the mask.
IN_BIT4	Input	BOOL	I, Q, M, D, L	Input bit 4 is compared with bit 4 of the mask.
IN_BIT5	Input	BOOL	I, Q, M, D, L	Input bit 5 is compared with bit 5 of the mask.
IN_BIT6	Input	BOOL	I, Q, M, D, L	Input bit 6 is compared with bit 6 of the mask.
IN_BIT7	Input	BOOL	I, Q, M, D, L	Input bit 7 is compared with bit 7 of the mask.
IN_BIT8	Input	BOOL	I, Q, M, D, L	Input bit 8 is compared with bit 8 of the mask.
IN_BIT9	Input	BOOL	I, Q, M, D, L	Input bit 9 is compared with bit 9 of the mask.
IN_BIT10	Input	BOOL	I, Q, M, D, L	Input bit 10 is compared with bit 10 of the mask.
IN_BIT11	Input	BOOL	I, Q, M, D, L	Input bit 11 is compared with bit 11 of the mask.
IN_BIT12	Input	BOOL	I, Q, M, D, L	Input bit 12 is compared with bit 12 of the mask.
IN_BIT13	Input	BOOL	I, Q, M, D, L	Input bit 13 is compared with bit 13 of the mask.
IN_BIT14	Input	BOOL	I, Q, M, D, L	Input bit 14 is compared with bit 14 of the mask.
IN_BIT15	Input	BOOL	I, Q, M, D, L	Input bit 15 is compared with bit 15 of the mask.
OUT	Output	BOOL	I, Q, M, D, L	The signal state "1" indicates that a match was found. The signal state "0" indicates that no match was found.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information

Parameter	Declaration	Data type	Memory area	Description
OUT_STEP	Output	BYTE	I, Q, M, D, L, P	Contains the step number with the matching mask, or the step number which is greater by "1" than the value of the LAST parameter, provided no match is found.
LAST	Static	BYTE	I, Q, M, D, L, P	Specifies the step number of the last step to be scanned for a matching mask.
CMP_VAL	Static	ARRAY of WORD	I, Q, M, D, L	Comparison masks [0 to 15, 0 to 15]: The first number of the index is the step number and the second number is the bit number of the mask.

For additional information on valid data types, refer to "See also".

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
000E	The value of the LAST parameter is greater than 15.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

LEAD_LAG: Lead and lag algorithm

Description

You use the "Lead and lag algorithm" instruction to process signals with an analog tag. The gain value in the GAIN parameter must be greater than zero. The result of the "Lead and lag algorithm" instruction is calculated using the following equation:

$$OUT = \left[\frac{LG_TIME}{LG_TIME + SAMPLE_T}\right] PREV_OUT + GAIN \left[\frac{LD_TIME + SAMPLE_T}{LG_TIME + SAMPLE_T}\right] IN - GAIN \left[\frac{LD_TIME}{LG_TIME + SAMPLE_T}\right] * PREV_IN$$

The "Lead and lag algorithm" instruction supplies plausible results only when processing is in fixed program cycles. The same units must be specified for the LD_TIME, LG_TIME, and

SAMPLE_T parameters. At LG_TIME > 4 + SAMPLE_T, the instruction approaches the following function:

OUT = GAIN * ((1 + LD_TIME * s) / (1 + LG_TIME * s)) * IN

When the value of the GAIN parameter is less than or equal to zero, the calculation is not performed and an error information is output in the ERR_CODE parameter.

You can use the "Lead and lag algorithm" instruction in conjunction with loops as a compensator in dynamic feed-forward control. The instruction consists of two operations. The "Lead" operation shifts the phase of the OUT output so that the output leads the input. The "Lag" operation, on the other hand, shifts the output so that the output lags behind the input. Because the "Lag" operation is equivalent to an integration, it can be used as a noise suppressor or as a low-pass filter. The "Lead" operation is equivalent to a differentiation and can therefore be used as a high-pass filter. The two operations together (Lead and Lag) result in the output phase lagging behind the input at lower frequencies and leading it at higher frequencies. This means that the "Lead and lag algorithm" instruction can be used as a band pass filter.

The "Call options" dialog opens automatically when the instruction is inserted in the program. You will find the created data block in the project tree in the "Program resources" folder under "Program blocks > System blocks". For additional information on this topic, refer to "See also".

Syntax

The following syntax is used for the "Lead and lag algorithm" instruction:

GRAPH

```
CALL LEAD_LAG, <InstanceDB>
     (IN := <Operand>
      SAMPLE_T := <Operand>
      OUT => <Operand>
      ERR_CODE => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Lead and lag algorithm" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	REAL	I, Q, M, D, L, P, or constant	The input value of the current sample time (cycle time) to be processed.
SAMPLE_T	Input	INT	I, Q, M, D, L, P, or constant	Sample time
OUT	Output	REAL	I, Q, M, D, L, P	Result of the instruction
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Error information

Parameter	Declaration	Data type	Memory area	Description
LD_TIME	Static	REAL	I, Q, M, D, L, P, or constant	Lead time in the same unit as sample time
LG_TIME	Static	REAL	I, Q, M, D, L, P, or constant	Lag time in the same unit as sample time
GAIN	Static	REAL	I, Q, M, D, L, P, or constant	Gain as % / % (the ratio of the change in output to a change in input as a steady state).
PREV_IN	Static	REAL	I, Q, M, D, L, P, or constant	Previous input
PREV_OUT	Static	REAL	I, Q, M, D, L, P, or constant	Previous output

For additional information on valid data types, refer to "See also".

Parameter ERR_CODE

The following table shows the meaning of the values of the ERR_CODE parameter:

Error code* (W#16#...)	Explanation
0000	No error
0009	The value of the GAIN parameter is less than or equal to zero.
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".	

Example

The following example shows how the instruction works:

Note

You can initialize static parameters in the data block.

GRAPH

```
CALL LEAD_LAG, "LEAD_LAG_DB"
      (IN := "Tag_Input"
       SAMPLE_T := "Tag_Input_SAMPLE_T"
       OUT => "Tag_Output_Result"
       ERR_CODE => "Tag_ErrorCode"
       )
```

The following tables show how the instruction works using specific values.

Before processing

In this example the following values are used for the input parameters:

Parameter	Operand	Value
IN	Tag_Input	2.0
SAMPLE_T	Tag_Input_SAMPLE_T	10

The following values are saved in the "LEAD_LAG_DB" instance data block of the instruction:

Parameter	Address	Value
LD_TIME	DBD12	2.0
LG_TIME	DBD16	2.0
GAIN	DBD20	1.0
PREV_IN	DBD24	6.0
PREV_OUT	DBD28	6.0

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
OUT	Tag_Output_Result	2.0

The following values are saved in the "LEAD_LAD_DB" instance data block of the instruction:

Parameter	Operand	Value
PREV_IN	DBD24	2.0
PREV_OUT	DBD28	2.0

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

TONR_X: Time accumulator

Description

You use the "Time accumulator" instruction to accumulate time values within a period set by the PT parameter. When the TMR_EN parameter changes to signal state "1", the instruction is executed and the time set at PT starts. The "Time accumulator" instruction adds the value

of the DELTA_T parameter and the value of the ET parameter as long as the following conditions are satisfied:

- The TMR_EN parameter has the signal state "1".
- The time value in the ET parameter is less than the time value in the PT parameter.
- The RESET parameter has the signal state "0".

When the signal state of the TMR_EN parameter changes to "0", the time in the DELTA_T parameter is no longer added to the time value in the ET parameter.

When the time value in the ET parameter reaches the time value in the PT parameter the Q parameter has the signal state "1". As long as the Q parameter has the signal state "1", the value of the ET parameter remains unchanged. The value of the ET parameter is reset to "0" when the RESET parameter is set to the value "1".

Since the "Time accumulator" instruction uses the execution time (DELTA_T) of the last cycle of the organization block (OB), the instruction may only be called in cyclic organization blocks.

Note

You must move the cycle time of the organization block from the startup information to the global tag of the DELTA_T parameter.

Syntax

The following syntax is used for the "Time accumulator" instruction:

GRAPH

```
CALL TONR_X
     (TMR_EN := <Operand>
      RESET := <Operand>
      PV := <Operand>
      DELTA_T := <Operand>
      Q => <Operand>
      ET := <Operand>
      )
```

Parameters

The following table shows the parameters of the "Time accumulator" instruction:

Parameter	Declaration	Data type	Memory area	Description
TMR_EN	Input	BOOL	I, Q, M, D, L	Activates the time for accumulating the time value.
RESET	Input	BOOL	I, Q, M, D, L	Reset input

Parameter	Declaration	Data type	Memory area	Description
PV	Input	DINT	I, Q, M, D, L, P, or constant	Maximum duration of time recording
DELTA_T	Input	INT	I, Q, M, D, L, or constant	Cycle time of OB from previous cycle.
Q	Output	BOOL	Q, M, D, L	Output that is set when time PT expires
ET	InOut	DINT	I, Q, M, D, L	Current time value of the elapsed time

For additional information on valid data types, refer to "See also".

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

WSR: Save data to shift register

Description

You use the "Save data to shift register" instruction to write data from a specified source to a shift register. Each element of the register is shifted into the next address. The data contained in the last address of the shift register is lost after the shift.

Each time the instruction is executed with the value "0" in the RESET parameter, new data are read from the source (S_DATA) and shifted to the start address (START) of the shift register. The LENGTH parameter specifies the number of elements to be shifted. You specify the data type of the elements in the E_TYPE parameter. The instruction is not executed if an invalid data type is specified for the E_TYPE parameter.

The register is deleted if the RESET parameter is set to "1" when the instruction is executed. The Q parameter is set to the signal state "1" when the shift register is filled with zeros or deleted.

Syntax

The following syntax is used for the "Save data to shift register" instruction:

GRAPH

```
CALL WSR
     (RESET := <Operand>
      S_DATA := <Operand>
      START := <Operand>
```

GRAPH

```
LEN := <Operand>
E_TYPE := <Operand>
Q => <Operand>
)
```

Parameters

The following table shows the parameters of the "Save data to shift register" instruction:

Parameter	Declaration	Data type	Memory area	Description
RESET	Input	BOOL	I, Q, M, D, L	Reset input
S_DATA	Input	POINTER	I, Q, M, D	Points to the source data element to be shifted to the start address (START). For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
START	Input	POINTER	I, Q, M, D	Points to the start address of the register. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
LEN	Input	WORD	I, Q, M, D, L, P	Number of elements to be shifted.

Parameter	Declaration	Data type	Memory area	Description
E_TYPE	Input	BYTE	I, Q, M, D, L, P	Specifies the data type of the elements to be shifted. The following data types are permitted: • W#16#04 = WORD • B#16#05 = INT • B#16#06 = DWORD • B#16#07 = DINT • B#16#08 = REAL
Q	Output	BOOL	Q, M, D, L	Is set to "0" if the RESET parameter is active (1) or all elements to be shifted have the value "0".

For additional information on valid data types, refer to "See also".

Example

In the following example four elements of the data type WORD are shifted starting from the address specified in the START parameter. When the first element of the register has been shifted to the next address, the first address is filled with the data that the pointer in the S_DATA parameter points to. The value of the last element in the register is lost in the process.

GRAPH

```
CALL WSR
      (RESET := "Tag_Input_Reset"
       S_DATA := "Tag_Input_DataSource"
       START := "Tag_Input_StartAddress"
       LEN := "Tag_Input_Length"
       E_TYPE := "Tag_Input_ElementType"
       Q => "Tag_Output_Q"
       )
```

The following tables show how the instruction works using specific values.

Before processing

In this example the following values are used for the input and output parameters:

Parameter	Operand	Value
RESET	Tag_Input_Reset	0
S_DATA	Tag_Input_DataSource	P#DB2.DBX6.0 DW0 = W#16#1111 DW2 = W#16#2222 DW4 = W#16#3333 DW6 = W#16#4444
START	Tag_Input_StartAddress	P#DB1.DBX0.0 DW6 = W#16#1234
LEN	Tag_Input_Length	W#16#04
E_TYPE	Tag_Input_ElementType	B#16#04
Q	Tag_Output_Q	1

After processing

The following values are written to the output parameters after the instruction has been executed:

Parameter	Operand	Value
RESET	Tag_Input_Reset	0
S_DATA	Tag_Input_DataSource	P#DB2.DBX6.0 DW0 = W#16#1234 DW2 = W#16#1111 DW4 = W#16#2222 DW6 = W#16#3333
START	Tag_Input_StartAddress	P#DB1.DBX0.0 DW6 = W#16#1234
LEN	Tag_Input_Length	W#16#04
E_TYPE	Tag_Input_ElementType	B#16#04
Q	Tag_Output_Q	1

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

SHRB: Shift bit to shift register

Description

You use the "Shift bit to shift register" instruction to write a bit from a specified source (DATA) to a shift register. Each time the instruction is executed with a value "0" on the RESET parameter, new data are read from the source and shifted to the start address (S_BIT) of the shift register. All following bits are shifted by one. The bit contained at the last address (S_BIT

+ N) is lost after the shift. If the RESET parameter is set to "1" during the processing of the instruction, the addresses in the shift register are reset to "0" and not shifted.

Syntax

The following syntax is used for the "Shift bit to shift register" instruction:

GRAPH

```
CALL SHRB
      (DATA := <Operand>
       RESET := <Operand>
       S_BIT := <Operand>
       N := <Operand>
       )
```

Parameters

The following table shows the parameters of the "Shift bit to shift register" instruction:

Parameter	Declaration	Data type	Memory area	Description
DATA	Input	BOOL	I, Q, M, D, L	Source bit
RESET	Input	BOOL	I, Q, M, D, L	Reset input
S_BIT	Input	POINTER	I, Q, M, D	Points to the start bit of the shift register. For area-crossing, indirect register addressing, the pointer must be specified in double-word format.
N	Input	WORD	I, Q, M, D, L, P	Number of bits to be shifted.

Example

In the following example the N parameter is set to the value 14 ("E" in hexadecimal). As a result 14 bits starting from the address specified in the S_BIT parameter are shifted. After the bits have been shifted, the first address is filled with the data from the source in the DATA parameter. The last bit value is lost.

GRAPH

```
CALL SHRB
      (DATA := "Tag_Input_Data"
       RESET := "Tag_Input_Reset"
       S_BIT := "Tag_Input_BitSource"
```

GRAPH

```
      N := "Tag_Input_Number"
      )
```

The following tables show how the instruction works using specific values.

Before processing

In this example the following values are used for the input and output parameters:

Parameter	Operand	Value
RESET	Tag_Input_Reset	0
DATA	Tag_Input_Data	0
S_BIT	Tag_Input_BitSource	P#DB1.DBX1.4
N	Tag_Input_Number	B#16#0E

Before processing the instruction the following signal states are stored in the 14 bits of the register:

1.7	...		S_BIT			...	1.0
0	0	1	1	1	0	1	0

2.7	...					...	2.0
0	0	1	1	1	0	1	0

3.7	...					...	3.0
0	0	1	1	1	0	1	0

After processing

After the processing of the instruction the following signal states are stored in the 14 bits of the register:

1.7	...		S_BIT			...	1.0
0	1	1	0	1	0	1	0

2.7	...					...	2.0
0	1	1	1	0	1	0	0

3.7	...					...	3.0
0	0	1	1	1	0	0	0

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

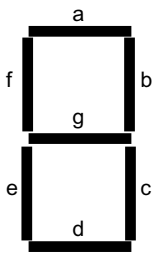
Basic information on GRAPH (Page 2718)

SEG: Create bit pattern for seven-segment display

Description

You use the "Create bit pattern for seven-segment display" instruction to convert each of the four hexadecimal digits of the specified source word (IN) into an equivalent bit pattern for a seven-segment display. The result of the instruction is output in the double word of the OUT parameter.

The following relation exists between the hexadecimal digits and the assignment of the 7 segments (a, b, c, d, e, f, g):

Input digit (Binary)	Assignment of the segments - g f e d c b a	Display (Hexadecimal)	Seven-segment display
0000	00111111	0	
0001	00000110	1	
0010	01011011	2	
0011	01001111	3	
0100	01100110	4	
0101	01101101	5	
0110	01111101	6	
0111	00000111	7	
1000	01111111	8	
1001	01100111	9	
1010	01110111	A	
1011	01111100	B	
1100	00111001	C	
1101	01011110	D	
1110	01111001	E	
1111	01110001	F	

Syntax

The following syntax is used for the "Create bit pattern for seven-segment display" instruction:

GRAPH

```
CALL SEG
     (IN := <Operand>
```

GRAPH

```
      OUT => <Operand>
      )
```

Parameters

The following table shows the parameters of the "Create bit pattern for seven-segment display" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	WORD	I, Q, M, D, L, P, or constant	Source word with four hexadecimal digits
OUT	Output	DWORD	I, Q, M, D, L, P	Bit pattern for the seven-segment display

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL SEG
      (IN := "Tag_Input"
       OUT => "Tag_Output"
       )
```

The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value	
Hexadecimal	Binary		
IN	Tag_Input	W#16#1234	0001 0010 0011 0100
OUT	Tag_Output	DW#16065B4F66	000 00110 0101 1011 0100 1111 0110 0110 Display: 1234

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

BCDCPL: Create tens complement

Description

You use the "Create tens complement" instruction to create the tens complement of a seven-digit BCD number specified in the IN parameter. This instruction uses the following mathematical formula to calculate:

10000000 (as BCD)

– 7-digit BCD value

Tens complement (as BCD)

Syntax

The following syntax is used for the "Create tens complement" instruction:

GRAPH

```
CALL BCDCPL
      (IN := <Operand>
       ERR_CODE => <Operand>
       )
```

Parameters

The following table shows the parameters of the "Create tens complement" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	DWORD	I, Q, M, D, L, P, or constant	7-digit BCD number
ERR_CODE	Output	DWORD	I, Q, M, D, L, P	Result of the instruction

For additional information on valid data types, refer to "See also".

Example

The following example shows how the instruction works:

GRAPH

```
CALL BCDCPL
      (IN := "Tag_Input"
       ERR_CODE => "Tag_Output"
       )
```

The following table shows how the instruction functions using specific values:

Parameter	Operand	Value*
IN	Tag_Input	DW#16#01234567
ERR_CODE	Tag_Output	DW#16#08765433
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".		

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

BITSUM: Count number of set bits

Description

You use the "Count number of set bits" instruction to count the number of bits of an operand that is set to the signal state "1". The operand whose bits are to be counted is specified in the IN parameter. The result of the instruction is output in the RET_VAL parameter.

Syntax

The following syntax is used for the "Count number of set bits" instruction:

GRAPH

```
CALL BITSUM
      (IN := <Operand>
       RET_VAL => <Operand>
       )
```

Parameters

The following table shows the parameters of the "Count number of set bits" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN	Input	DWORD	I, Q, M, D, L, P, or constant	Operand whose set bits are counted
RET_VAL	Output	INT	I, Q, M, D, L, P	Number of set bits

Example

The following example shows how the instruction works:

GRAPH

```
CALL BITSUM
      (IN := "Tag_Input"
       RET_VAL => "Tag_Output"
       )
```

The following table shows how the instruction functions using specific values:

Parameter	Operand	Value*
IN	Tag_Input	DW#16#12345678
RET_VAL	Tag_Output	W#16#000D (13 Bits)
*The error codes can be displayed as integer or hexadecimal value in the program editor. For additional information on toggling display formats, refer to "See also".		

See also

Overview of the valid data types (Page 2236)

Basic information on the status word (Page 2453)

Basic information on GRAPH (Page 2718)

LAD instructions Comparator operations

General LAD instructions

Description

Various LAD instructions are available to you when programming in GRAPH.

The following table shows an overview of the instructions and their availability:

LAD instruction	Permanent instruction	Transition/ Supervision/ Interlock
General		
Insert network	x	
Empty box	x	x
Open branch	x	x
Close branch	x	x
Bit logic operations		
--| |--: Normally open contact (Page 5374)	x	x
--| / |--: Normally closed contact (Page 5375)	x	x

LAD instruction	Permanent instruction	Transition/ Supervision/ Interlock
--|NOT|--: Invert RLO (Page 5376)	x	x
---( )---: Assignment (Page 5377)	x	
---(R)---: Reset output (Page 5378)	x	
---(S)---: Set output (Page 5379)	x	
SR: Set/reset flip-flop (Page 5380)	x	
RS: Reset/set flip-flop (Page 5382)	x	
--|P|--: Scan operand for positive signal edge (Page 5383)	x	
--|N|--: Scan operand for negative signal edge (Page 5384)	x	
P_TRIG: Scan RLO for positive signal edge (Page 5386)	x	
N_TRIG: Scan RLO for negative signal edge (Page 5387)	x	
Timer operations		
IEC timers		
TP: Generate pulse (Page 5389)	x	
TON: Generate on-delay (Page 5391)	x	
TOF: Generate off-delay (Page 5393)	x	
SIMATIC timers		
S_PULSE: Assign pulse timer parameters and start (Page 5395)	x	
S_PEXT: Assign extended pulse timer parameters and start (Page 5397)	x	
S_ODT: Assign on-delay timer parameters and start (Page 5400)	x	
S_ODTS: Assign retentive on-delay timer parameters and start (Page 5403)	x	
S_OFFDT: Assign off-delay timer parameters and start (Page 5405)	x	
Counter operations		
IEC counters		
CTU: Count up (Page 5416)	x	
CTD: Count down (Page 5418)	x	
CTUD: Count up and down (Page 5420)	x	
SIMATIC counters		
S_CU: Assign parameters and count up (Page 5424)	x	
S_CD: Assign parameters and count down (Page 5426)	x	
COUNTERS: Assign parameters and count up / down (Page 5422)	x	
Comparator operations		
CMP ==: Equal (Page 5432)	x	x
CMP <>: Not equal (Page 5433)	x	x
CMP >=: Greater or equal (Page 5435)	x	x
CMP <=: Less or equal (Page 5436)	x	x
CMP >: Greater than (Page 5437)	x	x
CMP <: Less than (Page 5438)	x	x
CMP>T: Greater than step activation time (Page 6400)		x
CMP>U: Greater than uninterrupted step activation time (Page 6401)		x
Math functions		
ADD: Add (Page 5440)	x	

LAD instruction	Permanent instruction	Transition/ Supervision/ Interlock
SUB: Subtract (Page 5441)	x	
MUL: Multiply (Page 5443)	x	
DIV: Divide (Page 5444)	x	
MOD: Return remainder of division (Page 5445)	x	
NEG: Create twos complement (Page 5447)	x	
ABS: Form absolute value (Page 5448)	x	
MIN: Get minimum (Page 5449)	x	
MAX: Get maximum (Page 5451)	x	
LIMIT: Set limit value (Page 5452)	x	
SQR: Form square (Page 5454)	x	
SQRT: Form square root (Page 5456)	x	
LN: Form natural logarithm (Page 5457)	x	
EXP: Form exponential value (Page 5458)	x	
SIN: Form sine value (Page 5460)	x	
COS: Form cosine value (Page 5461)	x	
TAN: Form tangent value (Page 5462)	x	
ASIN: Form arcsine value (Page 5464)	x	
ACOS: Form arccosine value (Page 5465)	x	
ATAN: Form arctangent value (Page 6169)	x	
Move operations		
MOVE: Move value (Page 5468)	x	
BLKMOV: Move block (Page 5470)	x	
UBLKMOV: Move block uninterruptible (Page 5474)	x	
FILL: Fill block (Page 5476)	x	
Conversion operations		
CONVERT: Convert value (Page 5479)	x	
ROUND: Round numerical value (Page 5481)	x	
CEIL: Generate next higher integer from floating-point number (Page 5482)	x	
FLOOR: Generate next lower integer from floating-point number (Page 5483)	x	
TRUNC: Truncate numerical value (Page 5485)	x	
SCALE: Scale (Page 5486)	x	
UNSCALE: Unscale (Page 5489)	x	
Program control operations		
Runtime control		
RE_TRIGR: Restart cycle monitoring time (Page 5498)	x	
STP: Exit program (Page 5499)	x	
WAIT: Configure time delay (Page 5500)	x	
PROTECT: Change protection level (Page 5500)	x	
COMPRESS: Compress CPU memory (S7-400) (Page 5497)	x	
CiR: Control CiR process (S7-400) (Page 5503)	x	
Word logic operations		

LAD instruction	Permanent instruction	Transition/ Supervision/ Interlock
AND: AND logic operation (Page 5505)	x	
OR: OR logic operation (Page 5507)	x	
XOR: EXCLUSIVE OR logic operation (Page 5508)	x	
INVERT: Create ones complement (Page 5510)	x	
DECO: Decode (Page 5511)	x	
ENCO: Encode (Page 5513)	x	
SEL: Select (Page 5514)	x	
Shift and rotate		
SHR: Shift right (Page 5515)	x	
SHL: Shift left (Page 5517)	x	
ROR: Rotate right (Page 5519)	x	
ROL: Rotate left (Page 5521)	x	
Additional instructions		
SET: Set bit array (Page 5537)	x	
SETP: Set a bit array in the I/O area (S7-400) (Page 5538)	x	
SETI: Set byte array (Page 5540)	x	
RESET: Reset bit array (Page 5542)	x	
RESETP: Reset a bit array in the I/O area (S7-400) (Page 5544)	x	
RESETI: Reset byte array (Page 5545)	x	
REPL_VAL: Enter substitute value (Page 5548)	x	
DRUM: Implement sequencer (Page 5549)	x	
DRUM_X: Implement sequencer (Page 5552)	x	
DCAT: Discrete control-timer alarm (Page 5556)	x	
MCAT: Motor control-timer alarm (Page 5559)	x	
IMC: Compare input bits with the bits of the mask (Page 5563)	x	
SMC: Compare scan matrix (Page 5565)	x	
LEAD_LAG: Lead and lag algorithm (Page 5567)	x	
TONR_X: Time accumulator (Page 5570)	x	
WSR: Save data to shift register (Page 5572)	x	
SHRB: Shift bit to shift register (Page 5575)	x	
SEG: Create bit pattern for seven-segment display (Page 5577)	x	
BCDCPL: Create tens complement (Page 5579)	x	
BITSUM: Count number of set bits (Page 5580)	x	

CMP>T: Greater than step activation time

Description

You use the "Greater than step activation time" instruction to monitor the total duration of a step in supervision conditions. The times of any events or errors are also recorded in the process.

In the programmed condition, the current or the last activation time of the step (Operand1) is compared with a defined duration (Operand2) in milliseconds. If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

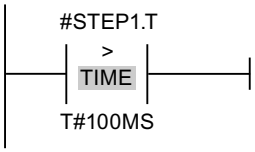
Parameters

The following table shows the parameters of the "Greater than step activation time" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	Integers	I, Q, M, D, L, or constant	Current or last activation time of the step
<Operand2>	Input	DINT, TIME, S5TIME	I, Q, M, D, L, or constant	Duration used for comparison

Example

The following example shows the instruction in the network:



As long as the activation time of #STEP1.T is less than 100 ms, the RLO is "0". As soon as the activation time exceeds 100 ms, the RLO changes to "1".

CMP>U: Greater than uninterrupted step activation time

Description

You use the "Greater than uninterrupted step activation time" instruction to monitor the duration of a step, minus any faults, in supervision conditions. The times of any events or faults are not recorded and only the pure step duration is monitored.

In the programmed condition, the total activation time of the step (Operand1) is compared with a defined duration (Operand2) in milliseconds. If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

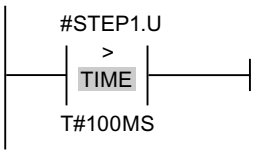
Parameters

The following table shows the parameters of the "Greater than uninterrupted step activation time" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	Integers	I, Q, M, D, L, or constant	Current or last activation time of the step minus the faults
<Operand2>	Input	DINT, TIME, S5TIME	I, Q, M, D, L, or constant	Duration used for comparison

Example

The following example shows the instruction in the network:



As long as the activation time of #STEP1.U, minus any faults, is less than 100 ms, the RLO is "0". As soon as the activation time exceeds 100 ms, the RLO changes to "1".

FBD instructions Comparator operations

General FBD instructions

Description

Different FBD instructions are available to you when programming in GRAPH.

The following table shows an overview of the instructions and their availability:

FBD instruction	Permanent operation	Transition/ Supervision/ Interlock
General		
New network: Insert network	x	
Empty box	x	x
Branch: Open branch	x	x
-|: Insert input (Page 5586)	x	x
-o|: Invert RLO (Page 5587)	x	x
Bit logic operations		
&: AND logic operation (Page 5581)	x	x

FBD instruction	Permanent operation	Transition/ Supervision/ Interlock
>=1: OR logic operation (Page 5583)	x	x
x: EXCLUSIVE OR logic operation (Page 5585)	x	
=: Assignment (Page 5588)	x	
-(R): Reset output (Page 5589)	x	
-(S): Set output (Page 5590)	x	
SR: Set/reset flip-flop (Page 5591)	x	
RS: Reset/set flip-flop (Page 5593)	x	
-|P|-: Scan operand for positive signal edge (Page 5594)	x	
-|N|-: Scan operand for negative signal edge (Page 5595)	x	
P_TRIG: Scan RLO for positive signal edge (Page 5597)	x	
N_TRIG: Scan RLO for negative signal edge (Page 5598)	x	
Timer operations		
IEC timers		
TP: Generate pulse (Page 5600)	x	
TON: Generate on-delay (Page 5602)	x	
TOF: Generate off-delay (Page 5604)	x	
SIMATIC timers		
S_PULSE: Assign pulse timer parameters and start (Page 5606)	x	
S_PEXT: Assign extended pulse timer parameters and start (Page 5608)	x	
S_ODT: Assign on-delay timer parameters and start (Page 5611)	x	
S_ODTS: Assign retentive on-delay timer parameters and start (Page 5614)	x	
S_OFFDT: Assign off-delay timer parameters and start (Page 5616)	x	
Counter operations		
IEC counters		
CTU: Count up (Page 5629)	x	
CTD: Count down (Page 5631)	x	
CTUD: Count up and down (Page 5633)	x	
SIMATIC counters		
S_CU: Assign parameters and count up (Page 5637)	x	
S_CD: Assign parameters and count down (Page 5639)	x	
S_CUD: Assign parameters and count up / down (Page 5635)	x	
Comparator operations		
CMP==: Equal (Page 5645)	x	x
CMP<>: Not equal (Page 5646)	x	x
CMP>=: Greater or equal (Page 5647)	x	x
CMP<=: Less or equal (Page 5648)	x	x
CMP>: Greater than (Page 5650)	x	x
CMP<: Less than (Page 5651)	x	x
CMP>T: Greater than step activation time (Page 6406)		x
CMP>U: Greater than uninterrupted step activation time (Page 6406)		x
Math functions		

FBD instruction	Permanent operation	Transition/ Supervision/ Interlock
ADD: Add (Page 5652)	x	
SUB: Subtract (Page 5653)	x	
MUL: Multiply (Page 5655)	x	
DIV: Divide (Page 5656)	x	
MOD: Return remainder of division (Page 5657)	x	
NEG: Create twos complement (Page 5659)	x	
ABS: Form absolute value (Page 5660)	x	
MIN: Get minimum (Page 5661)	x	
MAX: Get maximum (Page 5663)	x	
LIMIT: Set limit value (Page 5665)	x	
SQR: Form square (Page 5667)	x	
SQRT: Form square root (Page 5668)	x	
LN: Form natural logarithm (Page 5669)	x	
EXP: Form exponential value (Page 5671)	x	
SIN: Form sine value (Page 5672)	x	
COS: Form cosine value (Page 5673)	x	
TAN: Form tangent value (Page 5675)	x	
ASIN: Form arcsine value (Page 5676)	x	
ACOS: Form arccosine value (Page 5677)	x	
ATAN: Form arctangent value (Page 5679)	x	
Move operations		
MOVE: Move value (Page 5681)	x	
BLKMOV: Move block (Page 5683)	x	
UBLKMOV: Move block uninterruptible (Page 5686)	x	
FILL: Fill block (Page 5689)	x	
Conversion operations		
CONVERT: Convert value (Page 5692)	x	
ROUND: Round numerical value (Page 5693)	x	
CEIL: Generate next higher integer from floating-point number (Page 5694)	x	
FLOOR: Generate next lower integer from floating-point number (Page 5696)	x	
TRUNC: Truncate numerical value (Page 5697)	x	
SCALE: Scale (Page 5699)	x	
UNSCALE: Unscale (Page 5701)	x	
Program control operations		
Runtime control		
RE_TRIGR: Restart cycle monitoring time (Page 5710)	x	
STP: Exit program (Page 5711)	x	
PROTECT: Change protection level (Page 5712)	x	
WAIT: Configure time delay (Page 5712)	x	
COMPRESS: Compress CPU memory (S7-400) (Page 5709)	x	
CiR: Control CiR process (S7-400) (Page 5715)	x	

FBD instruction	Permanent operation	Transition/ Supervision/ Interlock
Word logic operations		
AND: AND logic operation (Page 5717)	x	
OR: OR logic operation (Page 5719)	x	
XOR: EXCLUSIVE OR logic operation (Page 5720)	x	
INVERT: Create ones complement (Page 5722)	x	
DECO: Decode (Page 5723)	x	
ENCO: Encode (Page 5725)	x	
SEL: Select (Page 5726)	x	
Shift and rotate		
SHR: Shift right (Page 5727)	x	
SHL: Shift left (Page 5729)	x	
ROR: Rotate right (Page 5731)	x	
ROL: Rotate left (Page 5733)	x	
Additional instructions		
SET: Set bit array (Page 5747)	x	
SETP: Set a bit array in the I/O area (S7-400) (Page 5748)	x	
SETI: Set byte array (Page 5750)	x	
RESET: Reset bit array (Page 5752)	x	
RESETP: Reset a bit array in the I/O area (S7-400) (Page 5754)	x	
RESETI: Reset byte array (Page 5755)	x	
REPL_VAL: Enter substitute value (Page 5758)	x	
DRUM: Implement sequencer (Page 5759)	x	
DRUM_X: Implement sequencer (Page 5762)	x	
DCAT: Discrete control-timer alarm (Page 5766)	x	
MCAT: Motor control-timer alarm (Page 5769)	x	
IMC: Compare input bits with the bits of the mask (Page 5773)	x	
SMC: Compare scan matrix (Page 5775)	x	
LEAD_LAG: Lead and lag algorithm (Page 5778)	x	
TONR_X: Time accumulator (Page 5781)	x	
WSR: Save data to shift register (Page 5782)	x	
SHRB: Shift bit to shift register (Page 5788)	x	
SEG: Create bit pattern for seven-segment display (Page 5792)	x	
BCDCPL: Create tens complement (Page 5793)	x	
BITSUM: Count number of set bits (Page 5795)	x	

See also

JMP: Jump if RLO = 1 (Page 5703)

JMPN: Jump if RLO = 0 (Page 5704)

LABEL: Jump label (Page 5705)

CMP>T: Greater than step activation time

Description

You use the "Greater than step activation time" instruction to monitor the total duration of a step in supervision conditions. The times of any events or errors are also recorded in the process.

In the programmed condition, the current or the last activation time of the step (Operand1) is compared with a defined duration (Operand2) in milliseconds. If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

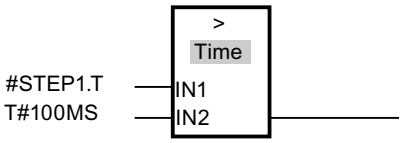
Parameters

The following table shows the parameters of the "Greater than step activation time" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	Integers	I, Q, M, D, L, or constant	Current or last activation time of the step
<Operand2>	Input	DINT, TIME, S5TIME	I, Q, M, D, L, or constant	Duration used for comparison

Example

The following example shows the instruction in the network:



As long as the activation time of #STEP1.T is less than 100 ms, the RLO is "0". As soon as the activation time exceeds 100 ms, the RLO changes to "1".

CMP>U: Greater than uninterrupted step activation time

Description

You use the "Greater than uninterrupted step activation time" instruction to monitor the duration of a step, minus any faults, in supervision conditions. The times of any events or faults are not recorded and only the pure step duration is monitored.

In the programmed condition, the total activation time of a step (Operand1) is compared with a defined duration (Operand2) in milliseconds. If the condition of the comparison is fulfilled, the instruction returns the result of logic operation (RLO) "1". If the comparison condition is not fulfilled, the instruction returns RLO "0".

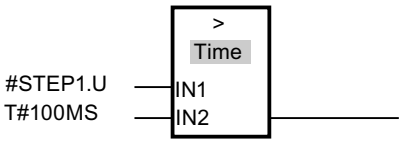
Parameters

The following table shows the parameters of the "Greater than uninterrupted step activation time" instruction:

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	Integers	I, Q, M, D, L, or constant	Current or last activation time of the step minus the faults
<Operand2>	Input	DINT, TIME, S5TIME	I, Q, M, D, L, or constant	Duration used for comparison

Example

The following example shows the instruction in the network:



As long as the activation time of #STEP1.U, minus any faults, is less than 100 ms, the RLO is "0". As soon as the activation time exceeds 100 ms, the RLO changes to "1".

9.8.2.3 Extended instructions

Date and time

T_COMP: Compare time tags

Description

You use the instruction to compare the contents of two tags with the data type DT, and output the result of the comparison as return value at parameter OUT. The parameter OUT is set to "1", when the applied comparison condition is satisfied.

The following comparison options can be used:

Symbol	Description
EQ	The return value has the signal state “1" if the time is the same at the parameter IN1 and IN2.
NE	The return value has the signal state “1" if the time at parameters IN1 and IN2 is not identical.
GE	The return value has the signal state "1" if the time at parameter IN1 is greater (more recent) than or equal to the time at parameter IN2 .
LE	The return value has the signal state "1" if the time at parameter IN1 is less (less recent) than or equal to the time at parameter IN2.

Symbol	Description
GT	The return value has the signal state “1" if the time at parameter IN1 is greater (more recent) than the time at parameter IN2.
LT	The return value has the signal state "1" if the time at parameter IN1 is less (less recent) than the time at parameter IN2 .

Parameter

The following table shows the parameters of the "T_COMP" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	DT	D, L or constant	Value to be compared
IN2	Input	DT	D, L or constant	Value to be compared
OUT	Output	BOOL	I, Q, M, D, L	Return value

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

T_CONV: Convert times and extract

Description

The "T_CONV" instruction is used to convert a DT (DATE_AND_TIME), S5TIME and TIME data types to a DATE, INT, TOD (TIME_OF_DAY), TIME or S5TIME data type.

Parameter

You decide the type of conversion by selecting the data types in the instruction boxes of the input and output of the instruction. You can query the result of the conversion at the OUT parameter.

The following table shows the parameters of the "T_CONV" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN	Input	DT, S5TIME, TIME	I, Q, M, D, L or constant	Value to be converted. The data format of the tag for parameter IN must match the data type selected using the instruction box.
OUT	Return	DATE, INT, TOD, TIME, S5TIME	I, Q, M, D, L	Result of the conversion. The data format of the tag for parameter OUT must match the data type selected using the instruction box.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Examples

Table 9-173 Converting data format TIME to data format S5TIME

Parameter	Declaration	Data type	Description
IN	Output	TIME	Input variable in the TIME data format.
OUT	Input	S5TIME	The function converts the TIME data format to the S5TIME format. The value is rounded down during conversion. If the input parameter is greater than the S5TIME format permits (greater than TIME#02:46:30.000), the result S5TIME#999.3 will be output and the binary result (BR) is set to "0".

Table 9-174 Extraction of the weekday from the DATE_AND_TIME format

Parameter	Declaration	Data type	Description
IN	Output	DT	Input variable in the DATE_AND_TIME data format.
OUT	Input	INT	The function extracts the weekday from the DATE_AND_TIME (DT) format. The weekday is output in INTEGER format. 1: Sunday 2: Monday 3: Tuesday 4: Wednesday 5: Thursday 6: Friday 7: Saturday

T_ADD: Add times

Description

You use the instruction "T_ADD" to add a duration (format TIME) to a new time (format DT) and output the result as a new time (format DT).

The time DT for the parameter IN1 must be in the range between DT#1990-01-01-00:00:00.000 and DT#2089-12-31-23:59:59.999. The instruction does not run an input check. If the result of the addition is not within the valid range, the result is limited to the corresponding value and the binary result (BR) bit of the status word is set to “0".

Parameters

The following table shows the parameters of the "T_ADD" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	DT	D, L or constant	Time in the DT (DATE_AND_TIME) format.
IN2	Input	TIME	I, Q, M, D, L or constant	Duration to be added in the TIME format.
OUT	Return	DT	D, L	Result of the addition in the DT (DATE_AND_TIME) format.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

T_SUB: Subtract times

Description

You use the instruction "T_SUB" to subtract a duration (format TIME) from a time (format DT)and output the result as a new time (format DT).

The time DT for the parameter IN1 must be in the range between DT#1990-01-01-00:00:00.000 and DT#2089-12-31-23:59:59.999. The instruction does not run an input check. If the result of the subtraction is not within the valid range, the result is limited to the corresponding value and the binary result (BR) bit of the status word is set to “0".

Parameters

The following table shows the parameters of the "T_SUB" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	DT	D, L or constant	Time in the DT (DATE_AND_TIME) format.
IN2	Input	TIME	I, Q, M, D, L or constant	Duration to be subtracted in the TIME format.
OUT	Return	DT	D, L	Result of the subtraction in the DT (DATE_AND_TIME) format.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

T_DIFF: Time difference

Description

You use the instruction "T_DIFF" to determine the difference between two times by subtracting two times of the DT format. The time at the IN2 input is subtracted form the time at the IN1 input. The result of the subtraction is output at the OUT output in the TIME format.

- If the first time at the IN1 input is greater (younger) than the second time at the IN2 input, the result at the OUT output is positive.
- If the first time at the IN1 input is less (older) than the second time at the IN2 input, the result at the OUT output is negative.

If the result of the subtraction is outside the TIME range, the result is limited to the corresponding value and the binary result (BR) is set to "0".

Parameter

The following table shows the parameters of the "T_DIFF" instruction:

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	DT	D, L or constant	Minuend
IN2	Input	DT	D, L or constant	Subtrahend
OUT	Return	TIME	I, Q, M, D, L	Difference in the TIME format.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

T_COMBINE: Combine times

Description

The "T_COMBINE" instruction combines the DATE and TIME_OF_DAY (TOD) data formats and converts these formats into the DATE_AND_TIME (DT) data format. The instruction does not report any errors.

- The DATE input value must be between DATE#1990-01-01 and DATE#2089-12-31 (will not be checked).
- At the IN2 input value, the TIME_OF_DAY data type is used.
- At the OUT output value, the DATE_AND_TIME data type is output.

Parameter

The following table shows the parameters of the "T_COMBINE" instruction:

Parameter	Declaration	Data type	Memory area	Description
IN1	Input	DATE	D, L or constant	Input tag in the DATE format.
IN2	Input	TOD	D, L or constant	Input tag in the TOD format.
OUT	Return	DT	D, L	Return value in the DT format.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Time-of-day functions

WR_SYS_T: Set time-of-day

Description

You set the time and the date of the CPU clock with the "WR_SYS_T" instruction. The clock then runs starting from the set time and date. You specify the date and time in the DT data format at the IN input of the instruction. You can use the "T_COMBINE (Page 6411)" instruction to convert the DATE and TOD formats to the required DT data format.

If the clock is a master clock, the CPU also starts the time-of-day synchronization when the instruction is called. You set the synchronization intervals per configuration. You cannot use the "WR_SYS_T" instruction to send information about the local time zone or daylight saving time.

At the RET_VAL output, you can query whether errors have occurred during execution of the instruction.

Parameter

The following table shows the parameters of the instruction "WR_SYS_T":

Parameters	Declaration	Data type	Memory area	Description
IN	Input	DT	D, L	Date and time
RET_VAL	Return	INT	I, Q, M, D, L	Status of the instruction

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code (W#16#...)	Description
0000	No error
8081	Year invalid
8082	Month invalid
8083	Day invalid
8084	Hour information invalid
8085	Minute information invalid
8086	Second information invalid
8087	Nanosecond information invalid
80B0	The realtime clock has failed.

Example

You enter the date and time as DT data type. As an example: for January 15, 2010, 10:30 a.m. and 30 seconds you would enter: DT#2010-01-15-10:30:30. The time-of-day can only be input down to seconds. The day of the week is calculated by "WR_SYS_T" based on the date.

RD_SYS_T: Read time-of-day

Description

You use the "RD_SYS_T" instruction to read the current date and current time-of-day of the CPU clock. The read data is provided in DT format at the OUT output of the instruction. The provided value does not include information about the local time zone or daylight saving time. At the RET_VAL output, you can query whether errors have occurred during execution of the instruction.

Parameter

The following table shows the parameters of the instruction "RD_SYS_T":

Parameters	Declaration	Data type	Memory area	Description
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
OUT	Output	DT	D,L	Date and time of CPU

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code (W#16#...)	Description
0000	No error
8222	The result is outside the permitted range of values.
8223	The result cannot be saved in the specified data type.

TIME_TCK: Read system time

Description

With the "TIME_TCK" instruction, you read the system time of the CPU. The system time is a time counter that counts from 0 to a maximum of 2147483647 ms. In case of an overflow, the system time is counted again starting with "0". The time scale and the accuracy of the system time is 1 ms. The system time is only influenced by the operating modes of the CPU. You can use the system time, for example, to measure the duration of processes by comparing the results of two "TIME_TCK" calls. The instruction does not provide any error information.

The following table provides an overview of how the system time changes depending on the operating modes of the CPU.

Mode	System time ...
Startup	... is constantly updated
RUN	
STOP	... is stopped and retains the current value
Restart (not with the S7-300)	... continues with the value saved at the change to STOP mode
Warm restart	... is deleted and restarts with "0"
Cold restart	

Parameter

The following table shows the parameters of the instruction "TIME_TCK":

Parameters	Declaration	Data type	Memory area	Description
RET_VAL	Return	TIME	I, Q, M, D, L	The RET_VAL parameter contains the read system time in the range from 0 to 2^{31} -1 ms.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

SET_CLKS: Set time-of-day and time-of-day status

Description

You use the "SET_CLKS" instruction to set the time-of-day of your CPU and the time-of-day status.

Note

Use "SET_CLKS" only if the time-of-day for your CPU is not going to be synchronized. Otherwise, with every synchronization the master clock time-of-day status would be applied. This step would overwrite the value specified with "SET_CLKS".

Note

You can determine the current time-of-day status of your CPU by reading SZL-ID W#16#0132 Index W#16#0008 (Page 6711) using the "RDSYSST (Page 6681)" instruction.

Parameters

The following table shows the parameters of the "SET_CLKS" instruction:

Parameter	Declaration	Data type	Memory area	Description
MODE	Input	BYTE	I, Q, M, D, L or constant	Operating mode: • B#16#01: Set time-of-day • B#16#02: Set time-of-day status • B#16#03: Set time-of-day and time-of-day status
PDT	Input	DT	D, L	Specified time-of-day
CORR	Input	INT	I, Q, M, D, L or constant	Correction value (in 0.5 h pattern) Possible values: –24 to +26
SUMMER	Input	BOOL	I, Q, M, D, L	Daylight saving time/standard time ID: • 0 = Standard time • 1 = Daylight saving time
ANN_1	Input	BOOL	I, Q, M, D, L	Announcement hour 1: At the next hourly change daylight saving time is switched over to standard time or vice versa.
RET_VAL	Return	INT	I, Q, M, D, L	Error code

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter MODE

Use the MODE parameter to specify whether to change only the time-of-day, only the time-of-day status, or both. This is explained in the table below:

MODE	Meaning
B#16#01	Set time-of-day The call corresponds to the call of the "SET_CLK" instruction. The input parameters CORR, SUMMER and ANN_1 are not evaluated.
B#16#02	Set time-of-day status The input parameter PDT is not evaluated. The remaining input parameters form the following time-of-day status elements: • Correction value including the sign • Announcement hour • Daylight saving time/standard time indicator The time-of-day resolution is matched to that of your CPU. The bit for synchronization failure of the time-of-day status is written with FALSE. The time-of-day remains unchanged.
B#16#03	Set time-of-day and time-of-day status

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error
8080	MODE outside the permitted value range
8081	CORR outside the permitted value range (only for MODE = B#16#02 or for MODE = B#16#03)
8082	PDT outside the permitted value range Illegal date and/or time-of-day
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

SNC_RTCB: Synchronize slave clocks

Definition: Synchronization of slave clocks

Synchronize slave clocks refers to the transmission of the date and time from the master clock of a bus segment (for example, the S7-400 communication bus, MPI, or S7 backplane bus) to all clock slaves of the bus segment.

Description

To ensure that the time-of-day is consistent in all modules on the network, the time-of-day slaves are updated by the system program at regular assignable intervals.

A synchronization can be performed independently from the configured synchronization interval by calling the "SNC_RTCB" instruction. This involves transmitting the date and time from the master clock to the slave clock for all time slaves on an existing bus segment.

Successful synchronization is only possible if "SNC_RTCB" is called on a CPU whose real-time clock was assigned the master clock function for at least one bus segment.

Parameter

The following table shows the parameters of the instruction "SNC_RTCB":

Parameters	Declaration	Data type	Memory area	Description
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error occurred during synchronization.
0001	The existing clock was not assigned the master clock function for any of the bus segments.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

Additional standard libraries

Handling of the runtime meter

Runtime meters

Introduction

The CPUs have a specific quantity of runtime meters (refer to the technical specifications of your CPUs).

- If your CPU is equipped with 16-bit runtime meters, you can set, start, stop, and read them with the "SET_RTM (Page 6420)", "CTRL_RTM (Page 6421)" and "READ_RTM (Page 6422)" instructions.
- If your CPU is equipped with 32-bit runtime meters, you can set, start, stop or read them with the "RTM (Page 6419)" instruction.

Note

You can also use the "SET_RTM (Page 6420)", "CTRL_RTM (Page 6421)" and "READ_RTM (Page 6422)" instructions for the 32-bit runtime meters. In this case, however, the runtime meters operate in the same way as 16-bit runtime meters (range of values: 0 to 32767 hours).

See also: Data record of the partial list extract with SZL-ID W#16#0132 index W#16#000B (Page 6713).

Application

You can use a runtime meter for a variety of applications:

- For measuring the runtime of the CPU
- For measuring the runtime of controlled equipment or connected devices.

Characteristics of the runtime meter

When it is started, the runtime meter begins to count starting at the last recorded value. If you want it to start at a different initial value, you must explicitly specify this value ("SET_RTM (Page 6420)" or "RTM (Page 6419)" with MODE=4). If the CPU changes to STOP mode, or you stop the runtime meter, the CPU records the current value of the runtime meter. When a warm restart or a cold restart of the CPU is executed, the runtime meter must be restarted ("CTRL_RTM (Page 6421)" or "RTM (Page 6419)" with MODE=1).

The runtime meters are reset to "0" following an update of the operating system or following a CPU reset to the factory state.

Range of values

- CPU with 16-bit runtime meters: 0 to 32 767 hours
- CPU with 32-bit runtime meters: 0 to (2E31) -1 hours = 2,147,483,647 hours

RTM: Runtime meters

Description

With the "RTM" instruction, you can set, start, stop, and read a 32-bit runtime meter of your CPU.

If you want to read all 32-bit runtime meters of your CPU, use the "RDSYSST (Page 6681)" instruction with the SZL_ID=W#16#0132 parameter and:

- INDEX=W#16#000B (for the runtime meters 0 to 7) or
- INDEX=W#16#000C (for the runtime meters 8 to 15).

See also: Data record of the partial list extract with SZL-ID W#16#0132 index W#16#000B (Page 6713).

Parameter

The following table shows the parameters of the instruction "RTM":

Parameters	Declaration	Data type	Memory area	Description
NR	Input	BYTE	I, Q, M, D, L or constant	Number of the runtime meter Numbering starts with 0. For information on the number of runtime meters of your CPU, refer to the technical data.
MODE	Input	BYTE	I, Q, M, D, L or constant	Job ID: • 0: Read out (the status is then written to CQ and the current value to CV). After the runtime meter has reached (2E31) -1 hours, it stops at the highest value that can be displayed and outputs an "Overflow" error message. • 1: start (at the last counter value) • 2: stop • 4: set (to the value specified in PV) • 5: set (to the value specified in PV) and then start • 6: set (to the value specified in PV) and then stop
PV	Input	DINT	I, Q, M, D, L or constant	New value for the runtime meter
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.

Parameters	Declaration	Data type	Memory area	Description
CQ	Output	BOOL	I, Q, M, D, L	Status of the runtime meter (1: running)
CV	Output	DINT	I, Q, M, D, L	Current value of the runtime meter

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Compatibility to programs developed for a CPU with 16-bit runtime meters

You can also use the 32-bit runtime meters with the "SET_RTM (Page 6420)", "CTRL_RTM (Page 6421)" and "READ_RTM (Page 6422)" instructions. In this case, however, the 32-bit runtime meters operate in the same way as 16-bit meters (range of values: 0 to 32767 hours).

The partial list extract with SZL-ID W#16#0132 and the index W#16#0008 displays the 32-bit runtime meters 0 to 7 in 16-bit mode. This means you can continue to use programs developed for a CPU with 16-bit runtime meters that use the partial list extract with SZL-ID W#16#0132 and the index W#16#0008 .

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error
8080	Wrong number for the runtime meter
8081	A negative value was transferred to the PV parameter.
8082	Overflow of the runtime meter
8091	The MODE input parameter contains an illegal value.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

SET_RTM: Set runtime meters

Description

You use the instruction to set a CPU runtime meter to a specified value. The number of runtime meters you can set depends on the particular CPU you are using.

Parameters

The following table shows the parameters of the instruction "SET_RTM":

Parameters	Declaration	Data type	Memory area	Description
NR	Input	BYTE	I, Q, M, D, L or constant	Input NR contains the number of the runtime meter you want to set. The numbering starts with "0". For information on the number of runtime meters of your CPU, refer to the technical data.
PV	Input	INT	I, Q, M, D, L or constant	Input PV contains the setting for the runtime meter.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error
8080	Wrong number for the runtime meter
8081	A negative value was transferred to the PV parameter.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

CTRL_RTM: Start and stop runtime meters

Description

You use the instruction to start or stop a runtime meter.

Parameters

The following table shows the parameters of the "CTRL_RTM" instruction:

Parameter	Declaration	Data type	Memory area	Description
NR	Input	BYTE	I, Q, M, D, L or constant	Input NR contains the number of the runtime meter you want to start or stop. The numbering starts with "0". For information on the number of runtime meters of your CPU, refer to the technical data.
S	Input	BOOL	I, Q, M, D, L	Input S starts or stops the runtime meter. • Set the signal state to "0" when you want to stop the runtime meter. • Set the signal state to "1" when you want to start the runtime meter.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error
8080	Wrong number for the runtime meter
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

READ_RTM: Read runtime meters

Description

You use the instruction to read a runtime meter. The instruction provides the current number of operating hours as output data and the status of the counter, that is, "stopped" or "counting".

If the runtime meter runs for longer than 32767 hours, it stops at the count 32767 and outputs the "Overflow" error message.

Parameter

The following table shows the parameters of the "READ_RTM" instruction:

Parameters	Declaration	Data type	Memory area	Description
NR	Input	BYTE	I, Q, M, D, L, P or constant	Input NR contains the number of the runtime meter you want to start or stop. The numbering starts with "0". For information on the number of runtime meters of your CPU, refer to the technical data.
RET_VAL	Return	INT	I, Q, M, D, L, P	If an error occurs while the instruction is being executed, the return value contains an error code.
CQ	Output	BOOL	I, Q, M, D, L	Output CQ indicates whether the runtime meter is running or stopped. The signal state "0" shows that the runtime meter is stopped. Signal state "1" shows that the runtime meter is running.
CV	Output	INT	I, Q, M, D, L, P	Output CV indicates the current value of the runtime meter.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error
8080	Wrong number for the runtime meter
8081	Overflow of the runtime meter
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

Local time

LOC_TIME: Calculate local time

Description

The instruction reads the time-of-day status and time-of-day of the CPU and calculates the local time from this information. The instruction is only of practical use on CPUs with a time-of-day status. The instruction can be called in OBs of every priority class.

Functional description

"Internally LOC_TIME" uses the " RDSYSST (Page 6681) " instruction to read out the module time, the correction value with sign and the daylight-saving time ID from the current time-of-day status of the CPU. "RDSYSST (Page 6681)" errors terminate the block. The error codes are passed to RET_VAL .

The correction value is the number of half-hours between the basic time and local time. It is also identified as daylight-saving or standard time.

If the CPU does not have a valid time status, "LOC_TIME" is terminated with an error.

After successfully reading out the time-of-day status, the time difference based on the correction value is added to the module time and applied to output LT .

The SUMMER output adopts the value of the daylight saving/standard time bit from the time-of-day status.

Parameters

The following table shows the parameters of the "LOC_TIME" instruction:

Parameters	Declaration	Data type	Description
RET_VAL	Return	INT	Error code
LT	Output	DATE_AND_TIME	Local time (Local Time)
SUMMER	Output	BOOL	Daylight saving time identification • 0: Winter • 1: Summer

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameters RET_VAL, LT and SUMMER

The following table shows the output values and the errors of "LOC_TIME":

RET_VAL	LT	SUMMER	Description
0	Local time	TRUE/FALSE	Instruction executed without errors
1	Local time	TRUE/FALSE	No error, but jump in date
8xyy	DT#90-01-01-0:0:0	FALSE	General error codes of "RDSYSST (Page 6681)"
8091	DT#90-01-01-0:0:0	FALSE	No time status

BT_LT: Calculate local time from base time

Description

The instruction calculates the local time for the base time set at the input.

The base time entered at the BT input is converted to the local time based on data stored in a data block (DB) and output at the LT output.

The data block (DB) contains the number of 30 minute units by which the base and local time differ and the difference between daylight saving and standard time also in units of 30 minutes.

If there is a date overflow in the conversion, this is indicated by a special return value.

The "BT_LT" instruction can be called in OBs of every priority class.

Parameter

The following table shows the parameters of the instruction "BT_LT":

Parameter	Declaration	Data type	Description
BT	Input	DATE_AND_TIME	Base time
WS_DAT	Input	BLOCK_DB	Information on the time zone and daylight saving/standard time adjustment (rule DB) At parameter WS_DAT, use a pointer to a data block of type "WS_RULES (Page 6439)".
RET_VAL	Return	INT	Error code
LT	Output	DATE_AND_TIME	Local time

For more information about the valid data types, refer to "Overview of the valid data types (Page 2236)".

Note

Parameters of data type BLOCK_DB can only be transferred to FB calls, but not to FC calls.

Parameters RET_VAL and LT

The following table shows the output values of "BT_LT":

RET_VAL	LT	Description
0	Local time	Instruction executed without errors
1	Local time	No error, but jump in date
8082	DT#90-01-01-0:0:0	Invalid data in the rule data block

LT_BT: Calculate base time from local time

Description

The instruction calculates the base time for the local time set at the input.

The local time entered at input LT is converted to the base time with the help of data stored in a data block (DB) and the result returned at output BT.

The data block (DB) contains the number of 30 minute units by which the base and local time differ and the difference between daylight saving and standard time also in units of 30 minutes. If there is a date overflow in the conversion, this is indicated by a special return value.

Special considerations when switching between daylight saving and standard time:

- At the changeover from standard to daylight saving time, the local time is put forward one hour. This means that the hour in between does not exist.
  At an instant LT within this hour, "LT_BT" "thinks" in daylight saving time. The result is indicated by the return value 4 or 5.
- At the changeover from daylight saving to standard time, the local time is put back one hour. This means that this one hour is passed through twice ("duplicated hour"). (For CE(S)T, the identifiers 2A and 2B are used).
  This prevents unambiguous mapping to a base time at an instant LT of this hour.
  "LT_BT" receives an LT as input parameter and must determine whether the value is a daylight saving or standard time before it converts it to BT. If LT is within the duplicated hour, the LT is interpreted as standard time. This result is indicated by the return value 2 or 3.

The "LT_BT" instruction can be called in OBs of every priority class.

Parameter

The following table shows the parameters of the instruction "LT_BT":

Parameter	Declaration	Data type	Description
LT	Input	DATE_AND_TIME	Local time
WS_DAT	Input	BLOCK_DB	Information on the time zone and daylight saving/standard time adjustment (rule DB) At parameter WS_DAT, use a pointer to a data block of type "WS_RULES (Page 6439)".
RET_VAL	Return	INT	Error code
BT	Output	DATE_AND_TIME	Base time

For more information about the valid data types, refer to "Overview of the valid data types (Page 2236)".

Note

Parameters of data type BLOCK_DB can only be transferred to FB calls, but not to FC calls.

Parameters RET_VAL and BT

The following table shows the output values of "LT_BT":

RET_VAL	BT	Description
0	Base time	Instruction executed without errors
1	Base time	No error, but jump in date
2	Base time	The LT at the input is within the "duplicated" hour.

RET_VAL	BT	Description
3	Base time	As for 2, but with additional date jump
4	Base time	The LT at the input is within the "prohibited" hour.
5	Base time	As for 4, but with additional date jump
8082	DT#90-01-01-0:0:0	Invalid data in the rule data block

S_LTINT: Set time-of-day interrupt using local time

Description

The instruction generates the required time-of-day interrupt at the set instant. This time is set in local time.

The local time entered at input LT is converted to the base time using the rule that is stored in a data block (DB).

The data block (DB) contains the number of 30 minute units by which the base and local time differ and the difference between daylight saving and standard time also in units of 30 minutes. The parameters of the specified time-of-day interrupt OB are set and the OB activated with the calculated base time. If there is a date overflow in the conversion, this is indicated by a special return value.

Special considerations when switching between daylight saving and standard time:

- At the changeover from standard to daylight saving time, the local time is put forward one hour. This means that the hour in between does not exist.
  At an instant LT within this hour, "S_LTINT" calculates the time in daylight saving format. This process is indicated by the return value (RET_VAL) 4 or 5.
- At the changeover from daylight saving to standard time, the local time is put back one hour. This means that this one hour is passed through twice ("duplicated hour"). (For CE(S)T, the identifiers 2A and 2B are used).
  This prevents unambiguous mapping to a base time at an instant LT of this hour. "S_LTINT" receives an LT as input parameter and must determine whether the value is a daylight saving or standard time before it converts it to BT. If LT is within the duplicated hour, the LT is interpreted as standard time. This process is indicated by the return value (RET_VAL) 2 or 3.

The "S_LTINT" instruction can be called in OBs of every priority class.

Parameter

The following table shows the parameters of the instruction "S_LTINT":

Parameter	Declaration	Data type	Description
OB_NR	Input	INT	Number of the OB to be started (permitted 10 – 17)
SDT	Input	DATE_AND_TIME	Start date and time-of-day in local time See also: "SET_TINT (Page 6605)"

Parameter	Declaration	Data type	Description
PERIOD	Input	WORD	Periods from start point SDT onwards: • W#16#0000 = once • W#16#0201 = once every minute • W#16#0401 = once hourly • W#16#1001 = once daily • W#16#1201 = once weekly • W#16#1401 = once monthly • W#16#1801 = once yearly • W#16#2001 = at month's end
WS_DAT	Input	BLOCK_DB	Information on the time zone and daylight saving/standard time adjustment (rule DB) At parameter WS_DAT, use a pointer to a data block of type "WS_RULES (Page 6439)".
RET_VAL	Return	INT	Error code

For more information about the valid data types, refer to "Overview of the valid data types (Page 2236)".

Note

Parameters of data type BLOCK_DB can only be transferred to FB calls, but not to FC calls.

Parameters RET_VAL

RET_VAL	Description
0	Instruction executed without errors
1	No error, but jump in date
2	The LT at the input was within the "duplicated" hour.
3	As for 2, but with additional date jump
4	The LT at the input is within the "prohibited" hour.
5	As for 4, but with additional date jump
8082	Invalid data in the rule data block
8090	Incorrect parameter OB_NR
8091	Incorrect parameter SDT
8092	Incorrect parameter PERIOD
80A1	The set start time is in the past.
80A2	OB is not loaded
80A3	OB cannot be started

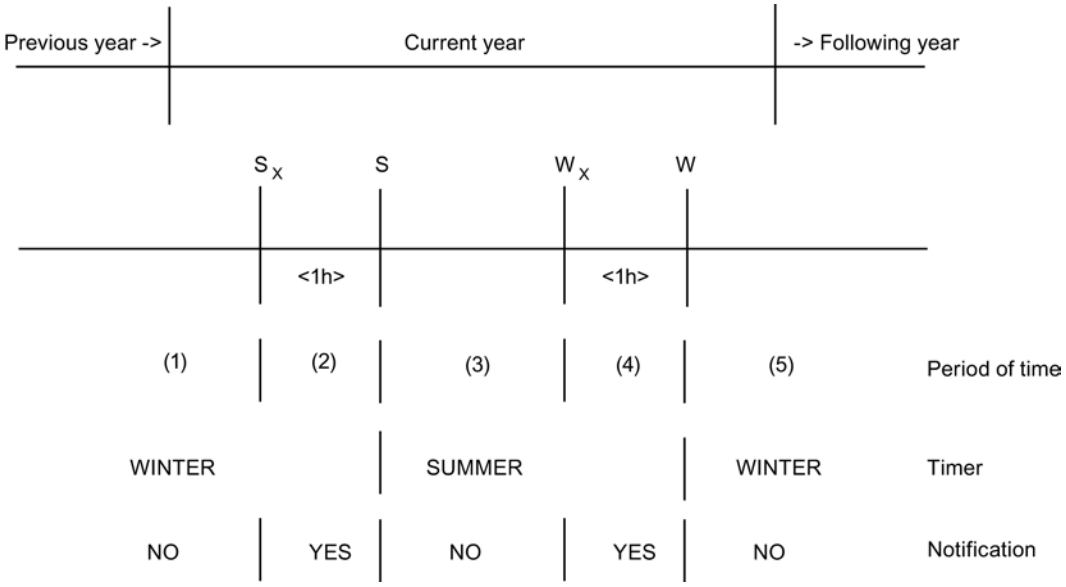
SET_SW: Set daylight saving time/standard time without time-of-day status

Description

The instruction supports the switch from daylight-saving time to standard time in CPUs that are not equipped with time-of-day status. The instruction "SET_SW" adjusts the CPU clock according to the current time and the adjustment rules in the rule DB.

"SET_SW" is called in the organization block OB1 and in a time-of-day interrupt OB once with the same instance. In the time-of-day interrupt OB, the parameters of the "SET_SW" instruction are not connected. The data block (DB) that contains the rule for the start and end of daylight-saving time is specified at input WS_DAT.

The following figure shows the time chart on which the function of "SET_SW" is based:



There are four adjustment times (S_A, S, W_A, W) defined. Their precise position is derived from the rules in the rule DB. The notification hours are one hour before the adjustment times and signal the pending adjustment.

A positive edge at the input REQ triggers initialization and activation of the daylight-saving/ standard time adjustment. This setting remains valid as long as REQ = 1.

For the time adjustment, "SET_SW" sets the parameters and activates the specified time-of-day interrupt OB with the next changeover time in the mode "once". The time-of-day interrupt OB then takes over control. When it becomes due, it changes the time on the CPU according to the current call time and sets parameters and activates itself for the next changeover time.

If the current time is in segment (3), the time-of-day interrupt OB is set to the start time W_A ("standard time notification").

On a falling edge at REQ, the daylight-saving/standard time adjustment is disabled.

Parameters

The following table shows the parameters of the instruction "SET_SW":

Parameters	Declaration	Data type	Description
REQ	Input	BOOL	Control of the changeover
WS_DAT	Input	BLOCK_DB	Information on the time zone and daylight-saving/standard time adjustment (rule DB) At parameter WS_DAT, use a pointer to a data block of type "WS_RULES (Page 6439)".
OB_NR	Input	INT	Number of the time-of-day interrupt to be used
STATUS	Output	INT	Error code
ISSUMMER	Output	BOOL	1 = Daylight saving time
ANN_1	Output	BOOL	1 = notification of a changeover

For more information about the valid data types, refer to "Overview of the valid data types (Page 2236)".

Note

Parameters of data type BLOCK_DB can only be transferred to FB calls, but not to FC calls.

Parameters STATUS

STATUS	Meaning
0	No error
8xyy	General error codes of "RDSYSST (Page 6681)"
8082	Invalid data in WS_DAT
8083	The month number for the change to standard time is higher than the month number for the change to daylight-saving time and notification hour for standard time is earlier than the hour for daylight-saving or the month number for the change to standard time is lower than the month number for the change to daylight-saving time and the notification hour for daylight-saving is earlier than the hour for standard time.
8090	Illegal OB_NR. (Only time-of-day interrupts permitted: OB10 – OB17, depending on the CPU)
80A2	Illegal OB_NR: The OB is not loaded on the CPU
80A3	The OB cannot be started. See also: "QRY_TINT (Page 6608)"

Application

"SET_SW" can be used in CPUs that are not equipped with time-of-day status. CPUs with time-of-day status use "SET_SW_S (Page 6431)".

Calling OBs

"SET_SW" must be called with the same instance in the following organization blocks (OBs):

- **OB1 (cyclic program):**
  "SET_SW" evaluates the REQ input. Based on the current time, the OB it sets the start of the specified time-of-day interrupt to the next of four possible changeover times and activates the time-of-day interrupt with the 'once' identifier at the positive edge at REQ. The time-of-day interrupt is triggered when this time is reached.
- **OB1x (time-of-day interrupt):**
  OB1x became active because a changeover time was reached. "SET_SW" sets the time on the CPU according to the current time (daylight-saving time!), sets its parameters to the next changeover time and activates it. In segment (5), it sets itself to the daylight-saving time notification (S_A) of the next year.

"Multiple instances of SET_SW" are not permitted.

You can use the following instructions to query, set and activate the time-of-day interrupt: "QRY_TINT (Page 6608)", "SET_TINT (Page 6605)", "ACT_TINT (Page 6607)", "CAN_TINT (Page 6607)"

Note

Adjusting the CPU time, for example from a PG, when "SET_SW" (REQ = 1) is activated is only permitted without special measures if this does not result in the time being adjusted past one of the changeover points.

If you missed out on one of the changeover points while adjusting the clock, you have to disable daylight-saving/standard time adjustment with REQ = 0 and then restart it with REQ = 1 after having completed the adjustment.

When you set the CPU clock, you must, of course, make sure that you set the daylight-saving time hour in summer.

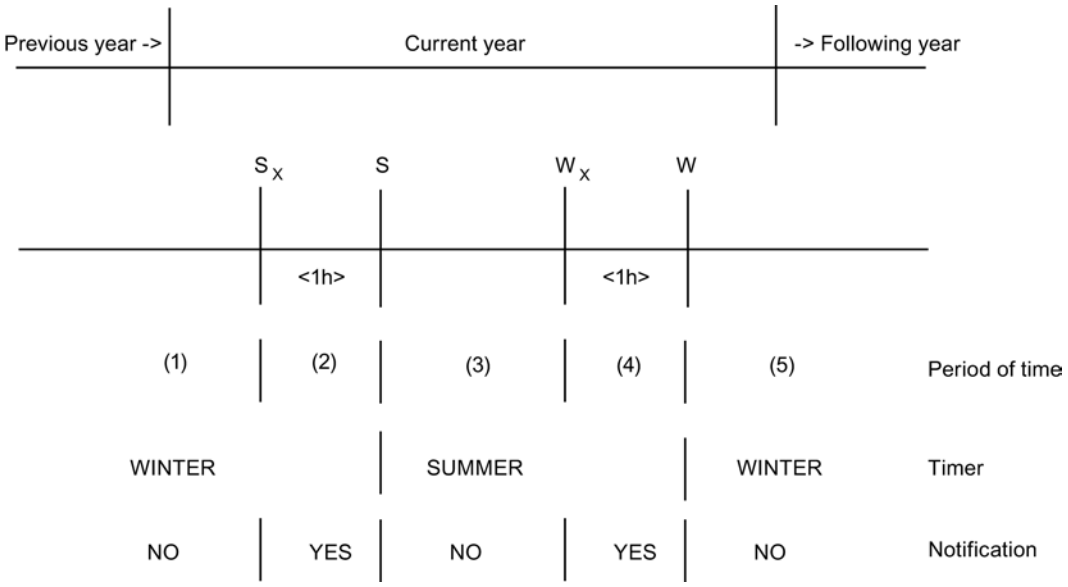
SET_SW_S: Set daylight saving time/standard time with time-of-day status

Description

The instruction supports the switch from daylight-saving time to standard time in CPUs that are equipped with time-of-day status. The instruction "SET_SW_S" adjusts the CPU clock according to the current time and the adjustment rules in the rule DB.

"SET_SW_S" is called in the organization block OB1 and in a time-of-day interrupt OB 1x with the same instance. In the time-of-day interrupt OB, the parameters of the "SET_SW_S" instruction are not connected. The data block (DB) that contains the rule for the start and end of daylight-saving time is specified at input WS_DAT.

The following figure shows the time chart on which the function of "SET_SW_S" is based:



There are four adjustment times (S_A, S, W_A, W) defined. Their precise position is derived from the rules in the rule DB. The notification hours are 1 hour before the adjustment times and signal the pending adjustment.

A positive edge at the input REQ triggers initialization and activation of the daylight-saving/ standard time adjustment. This setting remains valid as long as REQ = 1.

For the time adjustment, "SET_SW_S" sets the parameters and activates the specified time-of-day interrupt OB with the next changeover time in the mode "once". The time-of-day interrupt OB then takes over control. When it becomes due, it changes the time and the time status on the CPU according to the current call time and sets its parameters and activates itself for the next changeover time.

Example: If the current time is in segment (3), the time-of-day interrupt OB is set to the start time WA ("standard time notification").

On a falling edge at REQ, the daylight-saving/standard time adjustment is disabled.

Parameters

The following table shows the parameters of the instruction "SET_SW_S":

Parameters	Declaration	Data type	Description
REQ	Input	BOOL	Control of the changeover
WS_DAT	Input	BLOCK_DB	Information on the time zone and daylight-saving/standard time adjustment (rule DB) At parameter WS_DAT, use a pointer to a data block of type "WS_RULES (Page 6439)".
OB_NR	Input	INT	Number of the time-of-day interrupt to be used
STATUS	Output	INT	Error code
ISSUMMER	Output	BOOL	It is summer (DST)
ANN_1	Output	BOOL	Notification of the adjustment

For more information about the valid data types, refer to "Overview of the valid data types (Page 2236)".

Note

Parameters of data type BLOCK_DB can only be transferred to FB calls, but not to FC calls.

Parameter STATUS

STATUS	Meaning
0	No error
8xyy	General error codes of "RDSYSST (Page 6681)"
8082	Invalid data in WS_DAT
8083	The month number for the change to standard time is higher than the month number for the change to daylight-saving time and notification hour for standard time is earlier than the hour for daylight-saving or the month number for the change to standard time is lower than the month number for the change to daylight-saving time and the notification hour for daylight-saving is earlier than the hour for standard time.
8090	Illegal OB_NR. (Only time-of-day interrupts permitted: OB10 – OB17, depending on the CPU)
8091	Time status invalid ('status_valid' in time status set to FALSE)
80A2	Illegal OB_NR: The OB is not loaded on the CPU
80A3	The OB cannot be started. (see "QRY_TINT (Page 6608)")

Application

"SET_SW_S" can be used in CPUs that are equipped with time-of-day status.

Calling OBs

"SET_SW_S" must be called with the same instance in the following organization blocks (OBs):

- **OB1 (cyclic program):**
  "SET_SW_S" evaluates the REQ input. Based on the current time, the OB it sets the start of the specified time-of-day interrupt to the next of four possible changeover times and activates the time-of-day interrupt with the 'once' identifier at the positive edge at REQ. The time-of-day interrupt is therefore triggered when this time is reached.
- **OB1x (time-of-day interrupt):**
  OB1x became active because a changeover time was reached. "SET_SW_S" sets the time-of-day and the time-of-day status on the CPU according to the current time, sets its parameters to the next changeover time and activates it. In segment (5), it sets itself to the daylight-saving notification of the next year.

"Multiple instances of SET_SW_S" are not permitted.

You can use the following instructions to query, set and activate the time-of-day interrupt: "QRY_TINT (Page 6608)", "SET_TINT (Page 6605)", "ACT_TINT (Page 6607)", "CAN_TINT (Page 6607)"

Note

Adjusting the CPU time, for example from a PG, when "SET_SW_S" (REQ = 1) is activated is only permitted without special measures if this does not result in the time being adjusted past one of the changeover points.

If you missed out on one of the changeover points while adjusting the clock, you have to disable daylight-saving/standard time adjustment with REQ = 0 and then restart it with REQ = 1 after having completed the adjustment.

When setting the CPU clock and the time status, make sure that you set the daylight-saving time identifier or the notification hour identifier depending on whether you are setting daylight-saving or standard time.

TIMESTMP: Transfer time-stamped alarms

Description

The instruction transmits messages with time stamp of an IM153-2 to its instance DB. The data is available there for further processing by the user program.

- IM153-2
  The IM153-2 detects changes in binary signals, adds a time stamp to them and stores this information in data records with a maximum of 20 messages. With certain events, it outputs special messages. When a data record is ready to be fetched, the IM153-2 triggers a process interrupt.
- Call in the hardware interrupt OB
  "TIMESTMP" reads the start information of the process interrupt OB and stores the information relating to the time stamping in a circulating buffer for further processing. "TIMESTMP" can save the data of up to 17 process interrupts.
- Call in the cyclic OB:
  During cyclic processing, "TIMESTMP" reads the data record to be fetched using the instruction "RD_REC (Page 6491)" (read data record) and stores it in the message buffer MESSAGE. If there are several data records to be fetched, the oldest is read first. If the TIMECONV bit is set at the input, "TIMESTMP" converts all time stamps from the ISP format to the DATE_AND_TIME (DT) format.
  If the parameter is BUFRDY = TRUE, the messages in the message buffer can be further processed by the user program and, for example, transferred to a recipient (HMI device or printer). The number of existing messages is indicated by the MSG_QTY parameter. After processing, the user program must reset BUFRDY to release the message buffer for further messages.
  The BUFNOTREAD parameter indicates the number of received process interrupts for which "TIMESTMP" has not yet read data records from the IM153-2 . With BUFNOTREAD = 15 all data records of the IM153-2 are filled. The IM153-2 will then recognize no additional signal changes. Additional data records that are generated by the IM153-2 at a change of binary signals are lost. The MSGLOST parameter (message loss) is set to "1".

Parameter

The following table shows the parameters of the instruction "TIMESTMP":

Parameter	Declaration	Data type	Description
LADDR	Input	INT	Logical address (diagnostic address) of the IM153-2
LADDR2	Input	INT	Logical address (diagnostic address) of slot 2 of the IM153-2
TIMECONV	Input	BOOL	1 = convert ISP time stamp to S7-DT format
MSG_QTY	Output	INT	Number of valid message in the message buffer
BUFNOTREAD	Output	INT	Number of data records of the IM153-2 containing interrupts but not yet read
READERR	Output	BOOL	1 = error reading from the IM153-2
READSTATUS	Output	INT	Return value of instruction "RD_REC (Page 6491)"
BUFRDY	InOut	BOOL	1 = message buffer ready
MSGLOST	InOut	BOOL	1 = messages lost
MESSAGE	Static	ARRAY[1..20] OF STRUCT	Message buffer for up to 20 messages of the IM153-2
SLOT_NO	./.	BYTE	Slot number / 2 = special message
CH_NO	./.	BYTE	Channel number / ID of the special message
SIGNAL	./.	BYTE	Signal state / Characteristic of the special message
TIME1	./.	DWORD	Seconds starting from 1900.0 (ISP)/Year, Month, Day, Hour ((DT)
TIME2	./.	DWORD	Fractions of seconds (ISP) / Minute, Second, Millisecond, Day of week (DT)

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Hardware configuration

You can only use "TIMESTMP" in configurations that support forwarding of the time-of-day and time stamping of binary signals. These are:

- CPUs
  - CPU 4xx with CP443-5 for forwarding time frames (6GK7443-5DX02-0XE0 and higher), or
  - CPU 4xx with forwarding of time frames (for example, CPU416-2 DP as of firmware version 3.0)
- Distributed I/O
  - Interface module IM153-2 with redundancy capability with time stamping (6ES7153-2AA02 and higher, firmware version V1.2.3 (version 10))
    Or:
    IM153-2 FO with redundancy capability with integrated FO interface (6ES7153-2AB01 and higher, firmware version V1.2.3 and higher (version 9))
  - Digital input SM321 (6ES7321-7BH00-0AB0) isolated 16 DI, 24 V DC, hardware interrupt, diagnostics
    Or:
    Digital input SM321 (6ES7321-7TH00-0AB0) isolated 16 DI, 24 V DC, NAMUR, diagnostic capability, with process control functions
    You should only use time stamping for selected binary signals that are important in your application. It is advisable to distribute these signals over several IM153-2 or over several stations due to the load on PROFIBUS-DP and the IM153-2.

Software configuration

- Calling OBs
  "TIMESTMP" must be called with the same instance in the following organization blocks (OBs) and the block parameters must be written and read only in the cyclic program:
  - OB1 (cyclic program) or alternatively a cyclic interrupt OB (OB30 to OB38)
  - OB86 (rack failure)
  - OB100 (startup)
    If you use other startup OBs (OB101, OB102), "TIMESTMP" must be called there as well.
  - OB40 or, if selectable in the hardware configuration, also in other process interrupt OBs (OB41 to OB47)
- Addressing
  - Use a separate instance of "TIMESTMP" for each IM153-2. At the LADDR block input, enter the logical address (diagnostic address) set in the hardware configuration of the IM153-2.
  - If you use the DP master in the "S7-compatible" mode, enter the same value at input LADDR2 as for LADDR. If you use the "DPV1" mode, enter the diagnostic address of slot 2 of the IM153-2 for LADDR2 .
  - If you change addresses during operation, "TIMESTMP" deletes the previously stored process interrupt data.

Error handling

If "TIMESTMP" detects unrecoverable errors when reading a data record, "TIMESTMP" sets the READERR parameter to TRUE. The return value of the WR_USMSG instruction is available for further analysis in the READSTATUS parameter. Because "TIMESTMP" possibly reads another data record in the next cycle, READERR and READSTATUS are valid for only one cycle. You should therefore include suitable evaluation in the user program.

If data records are read while the DP master is out of service, "TIMESTMP" will output READERR = TRUE and READSTATUS =80B2h ("The configured slot is not in use").

Startup characteristics IM153-2

During startup/warm restart/cold restart, the IM153-2 once again outputs process interrupts for the data records; these interrupts had entries before the startup but these have not been fetched yet.

The IM153-2 enters the following messages in the first free data record:

- Special message "Begin startup data"
- Signal changes that occurred immediately prior to a CPU STOP ("incoming" or "outgoing" according to the parameter setting for the edge)
- Current signal status for all binary signals to be time stamped ("incoming" or "outgoing" according to the parameter setting for the edge)
- Special message "End startup data"

Startup characteristics of "TIMESTMP"

If "TIMESTMP" is called in a startup OB (OB100, OB101, OB102), "TIMESTMP" deletes all stored process interrupt data and resets BUFRDY . If data records with interrupts are known but not yet read, "TIMESTMP" sets MSGLOST (message loss) to TRUE. The user program must reset MSGLOST .

Response of "TIMESTMP" in OB86 (module failure)

If "TIMESTMP" is called in OB86 and if the event comes from the corresponding IM153-2, "TIMESTMP" reacts as during startup. This is, for example, the case when the IM153-2 fails and returns to operation and when the DP master returns to operation.

Redundancy

In H systems with two IM153-2 modules, time stamping is redundant in the following situations:

- There is communication between the two IM153-2 modules via the K bus.
- The updating of the active and passive IM153-2 modules has been completed without error.

Time stamping is interrupted during a failover between active and passive IM153-2 modules. The time for which time stamping is interrupted is indicated with the special message "Redundancy failover start/end".

Normally, the active IM153-2 module informs the passive one of the current I/O status. If this communication is disrupted, the special message "Loss of redundancy information incoming"

is output. As soon as communication is possible again between the active and passive IM153-2 modules, the special message "Loss of redundancy information outgoing" is output. Later resynchronization is not possible, in other words, a failover in the case of the IM153-2 causes a loss of messages.

Structure of the message buffer

The message buffer contains a data record of the IM153-2. The interpretation of the time stamp depends on the setting of the TIMECONV parameter.

Time formats of "TIMESTMP"

If the input is TIMECONV = TRUE , all time stamps are stored in the data type DATE_AND_TIME (BCD format):

The following table shows the structure of the data type DATE_AND_TIME:

Bytes	Contents	Range
0	Year	90 … 89 Corresponding to 1990 ... 2089
1	Month	01 ... 12
2	Day	01 ... 31
3	Hour	00 ... 23
4	Minute	00 ... 59
5	Second	00 ... 59
6	2 MSD (most significant decade) of ms	00 ... 99
7 (4 MSB)	LSD (least significant decade) of ms	0 ... 9
7 (4 LSB)	Day of week	1 ... 7 (1 = Sunday)

If the input is TIMECONV = FALSE , all time stamps are stored in ISP format (2 double words).

The following table shows the distribution of the bytes:

Bytes	Contents	Range
0 … 3	Seconds starting from 1.1.1900 0:00:00.000	Corresponding to 1.1.1900 ... 6.2.2036
4 … 7	Fractions of seconds in multiples of $1/2^{32}$ s	0 … <1

The time conversion is valid in the date range covered by both time formats 1.1.1990 up to and including 6.2.2036.

WS_RULES: Rule DB for converting the base time to local time

Data block of "WS_RULES" type

A data block of the "WS_RULES" type has the following structure.

Name		Data type	Start value	Description
// Conversion base time<->local time and "Set interrupt according to local time"				
B2L		Struct		
	S	INT	2	// Offset base->local time [30 min] in winter, permitted -24 ... +24.
	T	INT	2	// Difference standard and daylight-saving time [30 min], permitted: 2
// Rule for: Standard time -> daylight-saving time. Default: Last Sunday in March; 2:00 a.m.				
W2S		Struct		// in STANDARD TIME
	M	BYTE	B#16#3	// Month of conversion
	W	BYTE	B#16#9	// n-th occurrence of the day of the week (1=first, 2=second,..,9=last)
	D	BYTE	B#16#1	// Day of the week (Sunday = 1)
	H	BYTE	B#16#2	// Hour
// Rule for: Daylight-saving time -> standard time. Default: Last Sunday in October; 3:00 a.m.				
S2W				// in DAYLIGHT-SAVING TIME
	M	BYTE	B#16#10	// Month of conversion
	W	BYTE	B#16#9	// n-th occurrence of the day of the week (9 = last)
	D	BYTE	B#16#1	// Day of the week (Sunday = 1)
	H	BYTE	B#16#3	// Hour

Note

Create data block of "WS_RULES" type

You can create a data block of type "WS_RULES", by selecting type "WS_RULES" during the creation of a new DB.

Note

All parameters that have the BYTE format are interpreted as BCD values (binary-coded decimal value). W2S is output in standard time; S2W in daylight-saving time. The defining of the daylight-saving/standard time changeover times by a rule has been prescribed in the EU since 2002.

String + Char

S_COMP: Compare character strings

Description

The instruction compares the contents of two tags in the STRING format and outputs the result of the comparison as a return value. The tags that are to be compared will be interconnected at the IN1 and IN2 inputs. You can only assign a symbolically defined tag for the input parameters.

Use the instruction box to select the comparison condition. If the comparison condition (for example, greater or equal) is satisfied, the signal state is set to "1" on the output parameter OUT .

The following comparison options can be used:

Symbol	Description
EQ	The return value has the signal state "1" if the string at the IN1 parameter is the same as the string at the IN2 parameter.
NE	The return value has the signal state "1" if the string at the IN1 parameter is not equal to the string at the IN2 parameter.
GT (1)	The return value has the signal state “1" if the string at the IN1 parameter is greater than the string at the IN2 parameter.
LT (1)	The return value has the signal state "1" if the string at the IN1 parameter is less than the string at the IN2 parameter.
GE (1)	The return value has the signal state "1" if the string at the IN1 parameter is greater than or equal to the string at the IN2 parameter.
LE (1)	The return value has the signal state "1" if the string at the IN1 parameter is less than or equal to the string at the IN2 parameter.
(1) The characters are compared by their ASCII code (for example, 'a' is greater than 'A'), starting from the left. The first character to be different decides the result of the comparison. If the first characters are the same, the longer string is greater.	

Parameters

The following table shows the parameters of the instruction "S_COMP":

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	STRING*	D, L	Input tag in the STRING format.
IN2	Input	STRING*	D, L	Input tag in the STRING format.
OUT	Output	BOOL	I, Q, M, D, L	Result of comparison
* Define the maximum length of the character string if you use the data type STRING in the interface declaration for a temporary variable (you will find further information in the description of the data type).				

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

S_CONV: Convert character string

Description

You use the instruction to convert the value at the IN input to the data format you have specified at the OUT output. You can use the instruction to convert character strings to integers and floating-point numbers or to convert integers and floating-point numbers to character strings.

Converting a character string to integers and floating-point numbers

Following conversions of a character string to integers and floating-point numbers are possible:

- Conversion of a character string to a tag in INT format. The first character in the string may be a sign or a number, the characters which then follow must be numbers. If the length of the string is equal to zero or greater than 6, or if invalid characters are found in the string, no conversion takes place and the binary result (BR) bit of the status word is set to "0". If the result of the conversion is outside the INT range, the result is limited to the corresponding value and the binary result (BR) bit of the status word is set to "0".
- Conversion of a character string to a tag in DINT format. The first character in the string may be a sign or a number, the characters which then follow must be numbers. If the length of the string is equal to zero or greater than 11, or if invalid characters are found in the string, no conversion takes place and the binary result (BR) bit of the status word is set to "0". If the result of the conversion is outside the DINT range, the result is limited to the corresponding value and the binary result (BR) bit of the status word is set to "0".
- Conversion of a character string to a tag in REAL format. The string must have the following format: ±v.nnnnnnnE±xx
  - ± = sign
  - v = 1 digit before the decimal point
  - n = 7 digits after the decimal point
  - x = 2 exponential digits

  If the length of the string is less than 14, or if it is not structured as shown above, no conversion takes place and the binary result (BR) bit of the status word is set to "0". If the result of the conversion is outside the REAL range, the result is limited to the corresponding value and the binary result (BR) bit of the status word is set to "0".

Converting an integer and floating-point number to a character string

The following conversions of integers and floating-point numbers to a character string are possible:

- Conversion of a tag in INT format to a character string. The character string is shown preceded by a sign. If the tag specified at the return parameter is too short, no conversion will take place and the binary result (BR) bit of the status word is set to "0". You can only assign a symbolically defined tag for the output parameter.
- Conversion of a tag in DINT format to a character string. The character string is shown preceded by a sign. If the tag specified at the return parameter is too short, no conversion will take place and the binary result (BR) bit of the status word is set to "0". You can only assign a symbolically defined tag for the output parameter.
- Conversion of a tag in REAL format to a character string. The string is shown with 14 digits: ±v.nnnnnnnE±xx
  - ± = sign
  - v = 1 digit before the decimal point
  - n = 7 digits after the decimal point
  - x = 2 exponential digits

  If the tag specified at the return parameter is too short or if no valid floating-point number is specified at the INparameter, no conversion will take place and the binary result (BR) bit of the status word is set to "0". You can only assign a symbolically defined tag for the output parameter.

Parameters

The following table shows the parameters of the "S_CONV" instruction:

Table 9-175 Parameters for converting a character string to integers and floating-point numbers

Parameters	Declaration	Data type	Memory area	Description
IN	Input	STRING*	D, L	Value to be converted
OUT	Output	INT, DINT, REAL	I, Q, M, D, L	Result of the conversion.
* Define the maximum length of the character string if you use the data type STRING in the interface declaration for a temporary variable (you will find further information in the description of the data type).				

Table 9-176 Parameters for converting an integer and floating-point number to a character string

Parameters	Declaration	Data type	Memory area	Description
IN	Input	INT, DINT, REAL	I, Q, M, D, L or constant	Value to be converted
OUT	Output	STRING*	D, L	Result of the conversion
* Define the maximum length of the character string if you use the data type STRING in the interface declaration for a temporary variable (you will find further information in the description of the data type).				

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

ATH: Convert ASCII string to hexadecimal number

Description

You use the instruction to convert the ASCII string specified at the IN input to a hexadecimal number. The result of the conversion is stored at the address specified by the OUT parameter.

- You specify the number of ASCII characters to be converted with the N parameter.
- A maximum of 65 535 valid ASCII characters can be converted.
- Only the numbers 0 to 9 and upper case letters A to F can be interpreted. All other characters are converted to zeros.

Since 8 bits are required for the ASCII character and only 4 bits for the hexadecimal digit, the output word length is only half of the input word length. The ASCII characters are converted and positioned in the output in the same order as they are read in. If there is an odd number of ASCII characters, the hexadecimal number is padded with zeros in the nibble to the right of the last converted hexadecimal number.

If an invalid ASCII character is detected, it is converted as "0" and an error message is output to the RET_VAL parameter.

Parameters

The following table shows the parameters of the instruction "ATH":

Parameters	Declaration	Data type	Memory area	Description
IN	Input	POINTER	I, Q, M, D, L	Pointer to the start address of an ASCII string
N	Input	WORD	I, Q, M, D, L, P or constant	Number of ASCII characters to be converted
RET_VAL	Output	WORD	I, Q, M, D, L, P	Status of the instruction
OUT	Output	POINTER	I, Q, M, D, L	Pointer to the address of the result
* Pointer in double-word format for area-crossing, indirect register addressing				

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code (W#16#....)	Description
0000	No error
0007	Invalid character

Example

The instruction is executed if the signal state at input I 0.0 is 1 (activated). In this example, the N input parameter is 5 indicating that five ASCII characters are to be converted. The ASCII characters are stored in data block 1 beginning at the address specified by the IN pointer: DB1.DBX10.0. The character string is output at the address specified by the OUT pointer: Start at DB2.DBX0.0 (data block 2). Since there is an odd number of ASCII characters, the last hexadecimal digit contains only zeros in the right nibble, resulting in the hexadecimal value 0xC0.

If the instruction was executed without errors, RET_VAL will be set to the value W#16#0000.

The following table shows examples of the conversion of ASCII character strings to hexadecimal numbers.

IN	N	OUT	BR status
'0123'	4	16#0123	1
'123AFx1a23'	10	16#123AF01023	0

Table 9-177 The following table shows the ASCII characters and the corresponding hexadecimal values.

ASCII character	ASCII-coded hexadecimal value	Hexadecimal digit
"0"	30	0
"1"	31	1
"2"	32	2
"3"	33	3
"4"	34	4
"5"	35	5
"6"	36	6
"7"	37	7
"8"	38	8
"9"	39	9
"A"	41	A
"B"	42	B
"C"	43	C
D	44	D
E	45	E
F	46	F

HTA: Convert hexadecimal number to ASCII string

Description

You use the instruction to convert the hexadecimal number specified at the IN input to an ASCII string. The result of the conversion is stored at the address specified by the OUT parameter.

You specify the number of hexadecimal bytes to be converted with the N parameter. Since 8 bits are required for the ASCII character and only 4 bits for the hexadecimal digit, the output value is twice as long as the input value. Each nibble of the hexadecimal number is converted to a character while maintaining the original order.

A maximum of 65 635 characters can be written to the ASCII character string. The result of the conversion is represented by the digits 0 to 9 and upper-case letters A to F.

The instruction does not detect any error conditions.

Parameter

The following table shows the parameters of the instruction "HTA":

Parameters	Declaration	Data type	Memory area	Description
IN	Input	POINTER*	I, Q, M, D	Start address of the hexadecimal digits
N	Input	WORD	I, Q, M, D, L, P or constant	Number of hexadecimal bytes to be converted
OUT	Output	POINTER*	I, Q, M, D, L	Address at which the result is stored.
* Pointer in double-word format for area-crossing, indirect register addressing				

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Example

The instruction is executed if the signal state at input I 0.0 is 1 (activated). In this example, the N input parameter is set to 3 indicating that three hexadecimal characters are to be converted. The hexadecimal bytes are stored in data block 1 beginning at the address specified by the IN pointer: DB1.DBX10.0. The character string is output at the address specified by the OUT pointer: Start at DB2.DBX0.0 (data block 2).

The following table shows examples of the conversion of hexadecimal numbers to ASCII character strings:

IN	N	OUT	BR status
W#16#0123	2	'0123'	1
16#123AF01023	4	'123AF010'	0

The following table shows the ASCII characters and the corresponding hexadecimal values:

Hexadecimal digit	ASCII-coded hexadecimal value	ASCII character
0	30	"0"
1	31	"1"
2	32	"2"
3	33	"3"
4	34	"4"
5	35	"5"
6	36	"6"
7	37	"7"
8	38	"8"
9	39	"9"
A	41	"A"
B	42	"B"
C	43	"C"
D	44	"D"
E	45	"E"
F	46	"F"

Other instructions

LEN: Determine the length of a character string

Description

The instruction outputs the current length of a character string (number of valid characters) as return value.

A STRING tag contains two lengths:

- The maximum length (which is specified in square brackets when the tag is defined)
- The current length (this is the number of currently valid characters)

The current length must be smaller than or equal to the maximum length. The number of bytes occupied by a string is 2 greater than the maximum length. A blank string (' ') has the length zero. The maximum length is 254. The instruction does not report any errors.

Parameters

The following table shows the parameters of the instruction "LEN":

Parameters	Declaration	Data type	Memory area	Description
IN	Input	STRING*	D, L* *	Input tag in the format STRING; you can only assign a symbolically defined tag for the input parameter.
OUT	Return	INT	I, Q, M, D, L	Number of valid characters
* Define the maximum length of the character string if you use the data type STRING in the interface declaration for a temporary variable (you will find further information in the description of the data type). ** With SCL: Constants may also be used.				

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

CONCAT: Combine character strings

Description

The instruction combines two STRING tags into one character string. If the result string is longer than the tag given at the output parameter, the result string is limited to the maximum set length and the binary result (BR) bit of the status word set to "0".

Parameters

The following table shows the parameters of the instruction "CONCAT": You can assign only a symbolically defined tag for the parameters.

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	STRING*	D, L	Input tag in the STRING format.
IN2	Input	STRING*	D, L	Input tag in the STRING format.
OUT	Return	STRING*	D, L	Combined character string
* Define the maximum length of the character string if you use the data type STRING in the interface declaration for a temporary variable (you will find further information in the description of the data type).				

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

LEFT: Read the left character of a character string

Description

The instruction provides the first L characters of a string.

- If L is greater than the current length of the STRING tag, the input value is returned.
- With L = 0 and with a blank string as the input value, a blank string is returned.
- If L is negative, a blank string is output and the binary result (BR) bit of the status word is set to "0".

Parameters

The following table shows the parameters of the instruction "LEFT": You can only assign a symbolically defined tag for the parameter IN and the return value RET_VAL.

Parameters	Declaration	Data type	Memory area	Description
IN	Input	STRING*	D, L	Input tag in the STRING format.
L	Input	INT	I, Q, M, D, L or constant	Length of the left character string
OUT	Return	STRING*	D, L	Output tag in format STRING
* Define the maximum length of the character string if you use the data type STRING in the interface declaration for a temporary variable (you will find further information in the description of the data type).				

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

RIGHT: Read the right characters of a character string

Description

The instruction provides the last L characters of a string.

- If L is greater than the current length of the STRING tag, the input value is returned.
- If L is negative, a blank string is returned and the binary result (BR) bit of the status word is set to "0".
- With L = 0 and with a blank string as the input value, a blank string is returned.

Parameters

The following table shows the parameters of the instruction "RIGHT": You can only assign a symbolically defined tag for the parameter IN and the return value RET_VAL.

Parameters	Declaration	Data type	Memory area	Description
IN	Input	STRING*	D, L	Input tag in the STRING format.
L	Input	INT	I, Q, M, D, L or constant	Length of the right character string

Parameters	Declaration	Data type	Memory area	Description
OUT	Return	STRING*	D, L	Output tag in format STRING
* Define the maximum length of the character string if you use the data type STRING in the interface declaration for a temporary variable (you will find further information in the description of the data type).				

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

MID: Read middle characters of a character string

Description

The instruction provides the middle part of a character string (L characters from and including the P character).

- If the total of L and (P-1) exceeds the current length of the STRING tag, a string from the P character to the end of the input value is provided.
- In all other cases (P is outside the current length, P and/or L are equal to zero or negative), a blank string is returned and the binary result (BR) bit of the status word is set to "0".

Parameters

The following table shows the parameters of the instruction "MID": You can only assign a symbolically defined tag for the parameter IN and the return value RET_VAL.

Parameters	Declaration	Data type	Memory area	Description
IN	Input	STRING*	D, L	Input tag in the STRING format.
L	Input	INT	I, Q, M, D, L or constant	Length of the middle character string
P	Input	INT	I, Q, M, D, L or constant	Position of first character
OUT	Return	STRING*	D, L	Output tag in format STRING
* Define the maximum length of the character string if you use the data type STRING in the interface declaration for a temporary variable (you will find further information in the description of the data type).				

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

DELETE: Delete characters in a character string

Description

The instruction deletes (L characters in a character string from and including the P character).

- If L and/or P are equal to zero or if P is greater than the current length of the input string, the input string is returned.
- If the sum of L and P is greater than the input string, the string is deleted up to the end.
- If L and/or P is negative, a blank string is returned and the binary result (BR) bit of the status word is set to "0".

Parameters

The following table shows the parameters of the instruction "DELETE": You can only assign a symbolically defined tag for the parameter IN and the output parameter RET_VAL.

Parameters	Declaration	Data type	Memory area	Description
IN	Input	STRING*	D, L	STRING tag in which the delete is made
L	Input	INT	I, Q, M, D, L or constant	Number of characters to be deleted
P	Input	INT	I, Q, M, D, L or constant	Position of first character to be deleted
OUT	Return	STRING*	D, L	Result string
* Define the maximum length of the character string if you use the data type STRING in the interface declaration for a temporary variable (you will find further information in the description of the data type).				

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

INSERT: Insert characters in a character string

Description

The instruction adds the character string at the IN2 parameter to the character string at the IN1 parameter after the P character.

- If P equals zero, then the second character string will be inserted before the first character string.
- If P is greater than the actual length of the first character string, then the second character string is appended to the first character string.
- If P is negative, then a blank string will be output and the binary result (BR) of the status word is set to "0".

The binary result (BR) bit is also set to "0" if the result string is longer than the tag specified at the output parameter; in this case the result string is limited to the maximum set length.

Parameters

The following table shows the parameters of the instruction "INSERT": You can only assign a symbolically defined tag for the input parameters IN1 and IN2 and the output parameter.

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	STRING*	D, L	STRING tag in which the insertion is made
IN2	Input	STRING*	D, L	STRING tag that is to be inserted
P	Input	INT	I, Q, M, D, L or constant	Insert position
OUT	Return	STRING*	D, L	Result string
* Define the maximum length of the character string if you use the data type STRING in the interface declaration for a temporary variable (you will find further information in the description of the data type).				

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

REPLACE: Replace characters in a character string

Description

The instruction replaces the number of L characters of the first character string (IN1) starting from and including the P character by the complete second character string (IN2).

- If L equals zero and P does not equal zero, then the first string is returned.
- If L equals zero and P equals zero, then the second string is positioned in front of the first string.
- If L does not equal zero and P equals zero or one, it is replaced starting from and including the first character.
- If P is outside the first string, then the second string is appended to the first string.
- If L and/or P are/is negative, then a blank string will be returned and the binary result (BR) bit of the status word is set to "0". The binary result (BR) bit is also set to "0" if the result string is longer than the tag specified at the output parameter; in this case the result string is limited to the maximum set length.

Parameters

The following table shows the parameters of the instruction "REPLACE": You can only assign a symbolically defined tag for the input parameters IN1 and IN2 and the output parameter.

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	STRING*	D, L	STRING tag in which the insertion is made
IN2	Input	STRING*	D, L	STRING tag that is to be inserted
L	Input	INT	I, Q, M, D, L or constant	Number of characters to be replaced

Parameters	Declaration	Data type	Memory area	Description
P	Input	INT	I, Q, M, D, L or constant	Position of first replaced character
OUT	Return	STRING*	D, L	Result string
* Define the maximum length of the character string if you use the data type STRING in the interface declaration for a temporary variable (you will find further information in the description of the data type).				

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

FIND: Find characters in a character string

IND description

The instruction provides the position of the second string (IN2) within the first string (IN1). The search begins left. The first occurrence of the string is reported.

If the second string is not found in the first, zero is returned. The instruction does not report any errors.

Parameters

The following table shows the parameters of the instruction "FIND": You can only assign a symbolically defined tag for the input parameters IN1 and IN2 .

Parameters	Declaration	Data type	Memory area	Description
IN1	Input	STRING*	D, L	STRING tag in which the search is made
IN2	Input	STRING*	D, L	STRING tag that is searched for
OUT	Return	INT	I, Q, M, D, L	Position of the string found
* Define the maximum length of the character string if you use the data type STRING in the interface declaration for a temporary variable (you will find further information in the description of the data type).				

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Process image

UPDAT_PI: Update the process image inputs

Description

With the instruction, you update the OB 1 process image (=process image partition 0) of the inputs or a process image partition input of the inputs defined by configuration.

If you have configured repeated signaling of I/O access errors for the system-side process image update, then the selected process image will be updated constantly.

Otherwise, this update will only be performed if the selected process image partition is not updated by the system, in other words:

- If you have not assigned this process image partition to an interrupt OB,

  or

- If you selected process image partition 0 and have disabled updating of the OB 1 process image partition during configuration.

Note

Each logical address that you have assigned to a process image partition input table per configuration no longer belongs to the OB 1 process image input table.

When you update a process image partition with "UPDAT_PI", you must not perform a simultaneous update with the "SYNC_PI (Page 6455)" instruction.

System-side updating of the OB 1 process image input table and the process image partitions of inputs that you have assigned to an interrupt OB takes place independently of "UPDAT_PI" calls.

Parameter

The following table shows the parameters of the instruction "UPDAT_PI":

Parameters	Declaration	Data type	Memory area	Description
PART	Input	BYTE	I, Q, M, D, L or constant	Number of the process image partition input table to be updated. Maximum value range (depending on the CPU): 0 to 15 (0 means OB 1 process image, n where 1 < n < 15 means process image partition n)
RET_VAL	Return	INT	I, Q, M, D, L	Error information
FLADDR	Output	WORD	I, Q, M, D, L	Address of the first byte to cause an error if an access error has occurred.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error occurred.
8090	Illegal value for PART parameter
8091	The specified process image partition was not yet defined or is not located in the permitted process image area on the CPU.
8092	The process image partition is being updated by the system with an OB and you have not configured repeated signaling of all I/O access errors for this purpose. An update with "UPDAT_PI" was not performed.
80A0	An access error was detected when accessing the I/O.
8xyy	For general error information, refer to Evaluating errors with output parameter RET_VAL (Page 5370)

Note

If you use the instruction for process image partitions of DP standard slaves for which you have defined a consistency area greater than 32 bytes, the error codes of the "DPRD_DAT (Page 6498)" instruction are also possible.

UPDAT_PO: Update the process image outputs

Description

You use the instruction to transfer the signal states of the OB 1 process image (= process image partition 0) of the outputs or a process image partition of the outputs defined per configuration to the output modules.

If you have specified a consistency range for the selected process image partition, corresponding data is transferred as consistent data to the respective I/O module.

Note

Each logical address you have assigned to a process image partition output table during configuration no longer belongs to the OB 1 process image output table.

Outputs that you update with "UPDAT_PO" must not be updated simultaneously with the "SYNC_PO (Page 6457)" instruction.

The OB 1 process image output table and the process image partitions of the outputs that you have assigned to an interrupt OB are transferred by the system to the output modules independently of "UPDAT_PO" calls.

Parameter

The following table shows the parameters of the instruction "UPDAT_PO":

Parameters	Declaration	Data type	Memory area	Description
PART	Input	BYTE	I, Q, M, D, L or constant	Number of the process image partition output table to be transferred. Maximum value range (depending on the CPU): 0 to 15. (0 means OB 1 process image, n where 1 < n < 15 means process image partition n)
RET_VAL	Return	INT	I, Q, M, D, L	Error information
FLADDR	Output	WORD	I, Q, M, D, L	Address of the first byte to cause an error if an access error has occurred.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error occurred.
8090	Illegal value for PART parameter
8091	The specified process image partition was not yet defined or is not located in the permitted process image area on the CPU.
80A0	An access error was detected when accessing the I/O.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

Note

If you use the instruction for process image partitions of DP standard slaves for which you have defined a consistency area greater than 32 bytes, the error codes of the "DPWR_DAT (Page 6500)" instruction are also possible.

SYNC_PI: Synchronize the process image inputs

Description

The "SYNC_PI" is used to update the process image partition of inputs in isochronous mode. A user program linked to a DP cycle or PN send cyle can use this instruction to update input data acquired in a process image partition input table isochronously and consistently.

Call

"SYNC_PI" is interruptible and can only be called in OBs 61, 62, 63, and 64.

Note

A call of the "SYNC_PI" instruction in OBs 61 to 64 not allowed unless you have assigned the process image partition concerned to the associated OB in the hardware configuration.

Process image partitions updated with "SYNC_PI" may not be updated by means of "UPDAT_PI (Page 6452)" as well.

Parameter

The following table shows the parameters of the instruction "SYNC_PI":

Parameter	Declaration	Data type	Memory area	Range of values	Description
PART	Input	BYTE	I, Q, M, D, L or constant	1 to 30	Number of the process image partition input table to be updated isochronously.
RET_VAL	Return	INT	I, Q, M, D, L	-	Error information
FLADDR	Output	WORD	I, Q, M, D, L	-	Address of the first byte to cause an error if an access error has occurred.

For more information about the valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Event class error code	Explanation
W#16#0001	Consistency warning. The update of the process image partition was distributed over two DP and PN cycles. However, the data in one slave or IO device were consistently transferred.
W#16#8090	Illegal value on PART parameter or updating the specified process image partition input table is not permitted in this OB. The process image partition input table was not updated.
W#16#8091	The specified process image partition was not yet defined or is not located in the permitted process image area on the CPU. The process image partition input table was not updated.
W#16#80A0	During updating an access error was detected. The affected inputs were set to "0".
W#16#80A1	The update time is after the permitted access window. The process image partition input table was not updated. The DP or PN cycle is too short to ensure enough time for processing the instruction. You must therefore increase the TDP (also known as T_DC), Ti, and To timers.
W#16#80A2	Access error with consistency warning In the case of update of the specified process image partition an access error with simultaneous consistency warning was detected. • The data of the incorrect inputs was not read by the I/O. In the process image partition of the inputs the involved inputs are set to zero. • The update of the process image partition of the not involved by an access error input data was distributed over two DP and PN cycles.

Event class error code	Explanation
W#16#80C1	The update time is before the permitted access window. The process image partition input table was not updated.
W#16#8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

Note

If you use the "SYNC_PI" instruction for process image partitions of DP standard slaves for which you have defined a consistency area greater than 32 bytes, the error codes of the "DPRD_DAT (Page 6498)" instruction are also possible.

SYNC_PO: Synchronize the process image outputs

Description

Instruction "SYNC_PO" is used to update a process image partition of outputs in isochronous mode A user program linked to a DP cycle can use this instruction to transfer the calculated output data of a process image partition outputs to the I/O isochronously and consistently.

Call

"SYNC_PO" is interruptible and can only be called in OBs 61, 62, 63, and 64.

Note

A call of the "SYNC_PO" instruction in OBs 61 to 64 is only permitted if you have assigned the affected process image partition to the associated OB in the HW configuration.

A process image partition that you update with "SYNC_PO" must not be updated simultaneously with the "UPDAT_PO (Page 6454)" instruction.

Parameter

The following table shows the parameters of the instruction "SYNC_PO":

Parameter	Declaration	Data type	Memory area	Range of values	Description
PART	Input	BYTE	I, Q, M, D, L or constant	1 to 30	Number of the process image partition output table to be updated isochronously.
RET_VAL	Return	INT	I, Q, M, D, L	-	If an error occurs while the instruction is being executed, the return value contains an error code.
FLADDR	Output	WORD	I, Q, M, D, L	-	Address of the first byte that has caused the error.

For more information about the valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Event class error code	Explanation
W#16#0001	Consistency warning. The update of the process image partition was distributed over two DP and PN cycles. However, the data in one slave or IO device were consistently transferred.
W#16#8090	Illegal value in PART parameter, or updating the specified process image partition output table is not permitted in this OB. Outputs were not transferred to the I/O. The process image partition output table remains unchanged.
W#16#8091	The specified process image partition was not yet defined or is not located in the permitted process image area on the CPU. Outputs were not transferred to the I/O. The process image partition output table remains unchanged.
W#16#80A0	In the case of update of the specified process image partition of the outputs an access error was detected. Incorrect outputs were not transferred to the I/O. During process image partition of the outputs, these outputs remain unchanged.
W#16#80A1	Access error with consistency warning In the case of update of the specified process image partition of the outputs an access error with simultaneous consistency warning was detected. • The data of the incorrect outputs were not transferred to the I/O. During process image partition of the outputs, the involved outputs remain unchanged. • The update of the process image partition of the not involved by an access error input data was distributed over two DP and PN cycles.
W#16#80A2	The update time is after the permitted access window. Outputs were not transferred to the I/O. The process image partition output table remains unchanged.
W#16#80C1	The update time is before the permitted access window. Outputs were not transferred to the I/O. The process image partition output table remains unchanged.
W#16#8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

Note

If you use the "SYNC_PO" instruction for process image partitions of DP standard slaves for which you have defined a consistency area greater than 32 bytes, the error codes of the "DPWR_DAT (Page 6500)" instruction are also possible.

Distributed I/O

RDREC: Read data record

Description

You use the instruction to read the data record with the number INDEX from the component addressed using the ID . This may be a module in a central rack or a distributed component (PROFIBUS DP or PROFINET IO).

With MLEN , you specify the maximum number of bytes you want to read. The selected length of the destination area RECORD should have at least the length of MLEN bytes.

The value TRUE for the output parameter VALID indicates that the data record was successfully transferred to the destination area RECORD . In this case, the LEN output parameter contains the length of the read data in bytes.

If an error has occurred during transfer of the data record, this is indicated by the output parameter ERROR . In this case, the output parameter STATUS contains the error information.

Note

If a DPV1 slave is configured via GSD file (GSD rev. 3 and higher) and the DP interface of the DP master is set to "S7 compatible", then you may not read any data records from the I/O modules in the user program with "RDREC". In this case, the DP master addresses the wrong slot (configured slot + 3).

Remedy: set the interface of the DP master to "DPV1".

Note

The interface of the "RDREC" instruction is identical to the "RDREC" FB defined in "PROFIBUS Guideline PROFIBUS Communication and Proxy Function Blocks according to IEC 61131-3".

Functional description

"RDREC" works asynchronously, that is, its execution extends over multiple calls. You start the data record transfer by calling RDREC with REQ = 1.

The job status is displayed via the output parameter BUSY and the two central bytes of output parameter STATUS . The two central bytes of STATUS match the output parameter RET_VAL of the instructions that operate asynchronously.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 5367).

The transfer of the data record is complete when the output parameter BUSY has the value FALSE .

Parameter

The following table shows the parameters of the instruction "RDREC":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = 1: Transfer data record
ID	Input	DWORD	I, Q, M, D, L or constant	logical address of the DP slave/PROFINET IO component (module or sub-module) • For an output module, bit 15 has to be set (for example, for address 5: ID:=DW#16#8005). ID:=DW#16#8005). • For a mixed module, the lower of the two addresses must be specified.
INDEX	Input	INT	I, Q, M, D, L or constant	Data record number
MLEN	Input	INT	I, Q, M, D, L or constant	maximum length in bytes of the data record information to be read
VALID	Output	BOOL	I, Q, M, D, L	New data record was received and is valid
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The reading process is not yet complete.
ERROR	Output	BOOL	I, Q, M, D, L	ERROR = 1: An error occurred during the reading process.
STATUS	Output	DWORD	I, Q, M, D, L	Block status or error information For interpretation of the parameter STATUS , refer to parameter STATUS (Page 6471).
LEN	Output	INT	I, Q, M, D, L	Length of the read data record information
RECORD	InOut	ANY	I, Q, M, D, L	Destination area for the read data record Note: Note that for S7-300 CPUs the parameter RECORD always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and leads to an error message in the user program.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Note

If you use the instruction to read a data record with PROFINET IO, negative values in the INDEX, MLEN and LEN parameters are interpreted as an unsigned 16-bit integer.

WRREC: Write data record

Description

You use the instruction to transfer the data record RECORD to the component addressed using ID. This may be a module in a central rack or a distributed component (PROFIBUS DP or PROFINET IO).

With LEN (hidden), you specify in bytes the length of the data record to be transmitted. The selected length of the source area RECORD should have at least the length of LEN bytes.

The value TRUE at output parameter DONE indicates that the data record has been successfully transferred.

If an error has occurred during transfer of the data record, this is indicated by the output parameter ERROR . In this case, the output parameter STATUS contains the error information.

Note

If a DPV1 slave is configured via GSD file (GSD rev. 3 and higher) and the DP interface of the DP master is set to "S7 compatible", then you may not write any data records from the I/O modules to the user program with "WRREC". In this case, the DP master addresses the wrong slot (configured slot + 3).

Remedy: set the interface of the DP master to "DPV1".

Note

The interface of the "WRREC" instruction is identical to the "WRREC" instruction defined in "PROFIBUS Guideline PROFIBUS Communication and Proxy Function Blocks according to IEC 61131-3".

Functional description

"WRREC" works asynchronously, that is, its execution extends over multiple calls. You start the data record transfer by calling WRREC with REQ = 1.

The job status is displayed via the output parameter BUSY and the two central bytes of output parameter STATUS . The two central bytes of STATUS match the output parameter RET_VAL of the instructions that operate asynchronously.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 5367).

Note that you must assign the same value to the actual parameter of RECORD for all "WRREC"calls that belong to one and the same job. The same applies to the actual parameters of LEN.

The transfer of the data record is complete when the output parameter BUSY has the value FALSE .

Parameter

The following table shows the parameters of the instruction "WRREC":

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = 1: Transfer data record
ID	Input	DWORD	I, Q, M, D, L or constant	Logical address of the DP slave/ PROFINET IO components (assembly or module) 15 bits must be set with an output module (e.g. for address 5: ID:=DW#16#8005). For a mixed module, the lower of the two addresses must be specified.
INDEX	Input	INT	I, Q, M, D, L or constant	Data record number
LEN	Input	INT	I, Q, M, D, L or constant	(hidden) Maximum length of the data record to be transferred in bytes
DONE	Output	BOOL	I, Q, M, D, L	Data record was transferred
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The writing process is not yet completed.
ERROR	Output	BOOL	I, Q, M, D, L	ERROR = 1: An error occurred during the writing process.
STATUS	Output	DWORD	I, Q, M, D, L	Block status or error information For interpretation of the parameter STATUS , refer to parameter STATUS (Page 6471).
RECORD	InOut	ANY	I, Q, M, D, L	Data record Note: Note that for S7-300 CPUs the parameter RECORD always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and leads to an error message in the user program.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Note

If you use the instruction to read a data record with PROFINET IO, negative values in the INDEX and LEN parameters are interpreted as an unsigned 16-bit integer.

GETIO: Read process image

Description

You use the instruction to read out all inputs of a DP standard slave/PROFINET IO device. The instruction calls the "DPRD_DAT (Page 6498)" instruction. If there was no error during the data transmission, the data that have been read are entered in the destination area indicated by INPUTS .

The destination area must have the same length that you configured for the selected component.

If you read from a DP standard slave with a modular configuration or with several DP identifiers, you can only access the data of one component/DP identifier at the configured start address per "GETIO" call.

Parameter

The following table shows the parameters of the instruction "GETIO":

Parameters	Declaration	Data type	Memory area	Description
ID	Input	DWORD	I, Q, M, D, L or constant	• Low word: logical address of the DP slave/PROFINET IO component (module or sub-module) • High word: irrelevant
STATUS	Output	DWORD	I, Q, M, D, L	Contains the error information of "DPRD_DAT (Page 6498)" in the form DW#16#40xxxx00
LEN	Output	INT	I, Q, M, D, L	Amount of data read in bytes
INPUTS	InOut	ANY	I, Q, M, D	Destination area for the read data. It must have the same length as the area that you configured for the selected DP slave/ PROFINET IO component. Only the BYTE data type is permitted.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter STATUS

See also: DPRD_DAT: Read consistent data of a DP standard slave (Page 6498).

SETIO: Transfer process image

Description

You use the "SETIO" instruction to consistently transfer the data from the source area spanned by OUTPUTS to the addressed DP standard slave/PROFINET IO device, and, if necessary,

to the process image (if you have configured the relevant address area of the DP standard slave as a consistent range in a process image). "SETIO" calls the instruction "DPWR_DAT (Page 6500)".

The source area must have the same length that you configured for the selected component.

In the case of a DP standard slave with a modular configuration or with several DP identifiers, you can only access one component/DP identifier of the DP slave per "SETIO" call.

Parameter

The following table shows the parameters of the instruction "SETIO":

Parameters	Declaration	Data type	Memory area	Description
ID	Input	DWORD	I, Q, M, D, L or constant	• Low word: logical address of the DP slave/PROFINET IO component (module or sub-module) • High word: irrelevant
LEN	Input	INT	I, Q, M, D, L	irrelevant
STATUS	Output	DWORD	I, Q, M, D, L	Contains the error information of "DPWR_DAT (Page 6500)" in the form DW#16#40xxxx00
OUTPUTS	InOut	ANY	I, Q, M, D	Source area for the data to be written. It must have the same length as the area that you configured for the selected DP slave/PROFINET IO component. Only the BYTE data type is permitted.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter STATUS

See also: DPWR_DAT: Write consistent data of a DP standard slave (Page 6500).

GETIO_PART: Read process image area

Description

You use the "GETIO_PART" instruction to consistently read part of the process image area belonging to a DP standard slave/PROFINET IO device. "GETIO_PART" calls the instruction "UBLKMOV.

Note

You must assign a process image partition input table to the OB in which "GETIO_PART" is called. Before calling "GETIO_PART" you must add the associated DP standard slave or the associated PROFINET IO device to this process image partition input table. If your CPU does not recognize any process image partitions or you want to call "GETIO_PART" in OB 1, you must add the associated DP standard slave or the associated PROFINET IO device to this process image input table before calling "GETIO_PART".

You use the OFFSET and LEN parameters to specify the portion of the process image area to be read for the components addressed by means of their ID.

- If there was no error during the data transmission, ERROR receives the value FALSE, and the data that have been read are entered in the destination area indicated by INPUTS.
- If there was an error during the data transmission, ERROR receives the value TRUE, and STATUS receives the error information of "UBLKMOV".
- If the destination area (parameter INPUTS) is less than LEN, then as many bytes as INPUTS can accept will be transferred. ERROR receives the value FALSE. If the destination area is greater than LEN, then the first LEN bytes of the destination area will be written. ERROR receives the value FALSE.

Note

"GETIO_PART" does not check the process image input table for delimiters between data belonging to different PROFIBUS DP or PROFINET IO components. This means you have to make sure that the process image area specified by means of OFFSET and LEN belongs to one component. Reading of data for more than one component cannot be guaranteed for future systems and compromises the transferability to systems from other manufacturers.

Parameter

The following table shows the parameters of the instruction "GETIO_PART":

Parameters	Declaration	Data type	Memory area	Description
ID	Input	DWORD	I, Q, M, D, L or constant	• Low word: logical address of the DP slave/PROFINET IO component (module or sub-module) • High word: irrelevant
OFFSET	Input	INT	I, Q, M, D, L or constant	Number of the first byte to be read in the process image for the component (smallest possible value: 0)
LEN	Input	INT	I, Q, M, D or constant	Amount of bytes to be read
STATUS	Output	DWORD	I, Q, M, D, L	Contains the error information of "UBLKMOV" in the form DW#16#40xxxx00, if ERROR = TRUE
ERROR	Output	BOOL	I, Q, M, D, L	Error display: ERROR = TRUE if an error occurs when "UBLKMOV" is called.
INPUTS	InOut	ANY	I, Q, M, D	Destination area for read data: • If the destination area is less than LEN, then as many bytes as INPUTS can accept will be transferred. ERROR receives the value FALSE. • If the destination area is greater than LEN, then the first LEN bytes of the destination area will be written. ERROR receives the value FALSE.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameters STATUS and ERROR

See instruction "UBLKMOV".

SETIO_PART: Transfer process image area

Description

You use the "SETIO_PART" instruction to consistently transfer data from the source area spanned by OUTPUTS into a part of the process image area belonging to a DP standard slave/ PROFINET IO device. "SETIO_PART" calls the instruction "UBLKMOV".

Note

You must assign a process image partition input table to the OB in which "SETIO_PART" is called. Before calling "SETIO_PART", you must add the associated DP standard slave or the associated PROFINET IO device to this process image partition output table. If your CPU does not recognize any process image partitions or you want to call "SETIO_PART" in OB 1, you must add the associated DP standard slave or the associated PROFINET IO device to this process image output table before calling "SETIO_PART".

You use the OFFSET and LEN parameters to specify the portion of the process image area to be written for the components addressed by means of their ID.

- If there was no error during the data transmission, ERROR receives the value FALSE.
- If there was an error during the data transmission, ERROR receives the value TRUE, and STATUS receives the error information of "UBLKMOV".
- If the source area (OUTPUTS parameter) is less than LEN, then as many bytes as OUTPUTS can accept will be transferred. ERROR receives the value FALSE. If the source area is greater than LEN, then the first LEN bytes from OUTPUTS will be transferred. ERROR receives the value FALSE.

Note

"SETIO_PART" does not check the process image input table for delimiters between data belonging to different PROFIBUS DP or PROFINET IO components. This means you have to make sure that the process image area specified by means of OFFSET and LEN belongs to one component. Writing of data for more than one component cannot be guaranteed for future systems and compromises the transferability to systems from other manufacturers.

Parameter

The following table shows the parameters of the instruction "SETIO_PART":

Parameters	Declaration	Data type	Memory area	Description
ID	Input	DWORD	I, Q, M, D, L or constant	• Low word: logical address of the DP slave/PROFINET IO component (module or sub-module) • High word: irrelevant
OFFSET	Input	INT	I, Q, M, D, L or constant	Number of the first byte to be written in the process image for the component (smallest possible value: 0)

Parameters	Declaration	Data type	Memory area	Description
LEN	Input	INT	I, Q, M, D, L or constant	Amount of bytes to be written
STATUS	Output	DWORD	I, Q, M, D, L	Contains the error information of "UBLKMOV" in the form DW#16#40xxxx00, if ERROR = TRUE
ERROR	Output	BOOL	I, Q, M, D, L	Error display: ERROR = TRUE if an error occurs when "UBLKMOV" is called.
OUTPUTS	InOut	ANY	I, Q, M, D	Source area for the data to be written: • If the source area is less than LEN, then as many bytes as OUTPUTS can accept will be transferred. ERROR receives the value FALSE. • If the source area is greater than LEN, then the first LEN bytes from OUTPUTS will be transferred. ERROR receives the value FALSE.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameters STATUS and ERROR

See instruction "UBLKMOV".

RALRM: Receive interrupt

Description RALRM

Description

The instruction receives an interrupt with all corresponding information from an I/O module (centralized structure) or from a DP slave or PROFINET IO device component; it supplies this information to its output parameters.

The information in the output parameters contains the start information of the called OB as well as information of the interrupt source.

Call the instruction only within the interrupt OB started by the CPU operating system as a result of the I/O interrupt that is to be examined.

Note

If you call the instruction in an OB whose start event is not an I/O interrupt, the instruction will provide less information to your outputs.

Note that you use different instance DBs when you call the instruction in different OBs. If you evaluate data resulting from an "RALRM" call outside of the associated interrupt OB, you should moreover use a separate instance DB per OB start event.

Note

The interface of the "RALRM" instruction is identical to the "RALRM" FB defined in "PROFIBUS Guideline PROFIBUS Communication and Proxy Function Blocks according to IEC 61131-3".

Calling "RALRM"

You can call the instruction in three operating modes (MODE). These modes are explained in the following table.

MODE	RALRM ...
0	... shows the component that triggered the interrupt in the output parameter ID and sets the output parameter NEW to TRUE.
1	... writes all output parameters, independent of the interrupt triggering component.
2	... checks whether the component specified in input parameter F_ID has triggered the interrupt. • If not, NEW = FALSE • If yes, NEW = TRUE and all other output parameters are written.

Parameters

The following table shows the parameters of the instruction "RALRM":

Parameters	Declaration	Data type	Memory area	Description
MODE	Input	INT	I, Q, M, D, L or constant	Mode
F_ID	Input	DWORD	I, Q, M, D, L or constant	Logical start address of the component (module) from which interrupts will be received
MLEN	Input	INT	I, Q, M, D, L or constant	Maximum length of the interrupt information to be received, in bytes
NEW	Output	BOOL	I, Q, M, D, L	A new interrupt was received.
STATUS (Page 6471)	Output	DWORD	I, Q, M, D, L	Error code of the instruction or DP master

Parameters	Declaration	Data type	Memory area	Description
ID	Output	DWORD	I, Q, M, D, L	Logical start address of the component (module) from which an interrupt was received. Bit 15 contains the I/O ID: 0 = input address, 1 = output address.
LEN	Output	INT	I, Q, M, D, L	Length of the received interrupt information
TINFO (Page 6475)	InOut	ANY	I, Q, M, D, L	(task information) Destination area for OB start and management information Note: Note that for S7-300 CPUs, the parameter TINFO always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and leads to an error message in the user program.
AINFO (Page 6478)	InOut	ANY	I, Q, M, D, L	(alarm information) Destination area for header information and additional interrupt information For AINFO you should provide a length of at least MLEN bytes. Note: Note that for S7-300 CPUs, the parameter AINFO always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and leads to an error message in the user program.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Note

If you select a destination area TINFO or AINFO that is too short, the instruction cannot enter the full information.

See also: Destination area TINFO and AINFO (Page 6485).

Parameter STATUS

Parameter STATUS structure

The STATUS output parameter contains error information. If it is interpreted as ARRAY[1...4] OF BYTE , the error information has the following structure:

Field element	Name	Meaning
STATUS[1]	Function_Num	• B#16#00, if no error • Function ID from DPV1-PDU: If an error occurs, B#16#80 will be output (for read data record: B#16#DE, for write data record: B#16#DF). If no DPV1 protocol element is used: B#16#C0.
STATUS[2]	Error_Decode	Location of the error ID
STATUS[3]	Error_Code_1	Error ID
STATUS[4]	Error_Code_2	Manufacturer-specific error ID extension

Field element STATUS[2]

STATUS[2] can have the following values:

Error_Decode (B#16#...)	Source	Meaning
00 to 7F	CPU	No error or no warning
80	DPV1	Error according to IEC 61158-6
81 to 8F	CPU	B#16#8x shows an error in the xth call parameter of the instruction.
FE, FF	DP profile	Profile-specific error

Field element STATUS[3]

STATUS[3] can have the following values:

Error_Decode (B#16#...)	Error_Code_1 (B#16#...)	Explanation according to DVP1	Meaning
00	00		No error, no warning
70	00	reserved, reject	Initial call; no active data record transfer
	01	reserved, reject	Initial call; data record transfer has started
	02	reserved, reject	Intermediate call; data record transfer already active
80	90	reserved, pass	Invalid logical start address

Error_Decode (B#16#...)	Error_Code_1 (B#16#...)	Explanation according to DVP1	Meaning
	92	reserved, pass	Illegal type for ANY pointer
	93	reserved, pass	The DP component addressed via ID or F_ID is not configured.
	95		Error when reading additional interrupt information (when reading out additional interrupt information for central or distributed I/O via an external DP interface, this error will be output as a "group error".) Note: During link-up and update, the additional interrupt information may not be available temporarily.
	96		The master CPU is in STOP mode. At that time, an OB was being processed. The instruction "RALRM (Page 6468)" cannot supply the OB start information, the management information, the header information, or the additional interrupt information. The OB start information can be read with the "RD_SINFO (Page 6679)" instruction. In addition, you can use the "DPNRM_DG (Page 6523)" instruction to asynchronously read the current diagnostics frame of the affected DP slave for OBs 4x, 55, 56, 57, 82, and 83 (address information from the OB start information).
	A0	read error	Negative acknowledgment while reading the module.
	A1	write error	Negative acknowledgement when writing to the module
	A2	module failure	DP protocol error at layer 2 (e.g., slave failure or bus problems)
	A3	reserved, pass	• PROFIBUS DP: DP protocol error with Direct-Data-Link-Mapper or User-Interface/User • PROFINET IO: General CM error
	A4	reserved, pass	Communication on the communication bus disrupted
	A5	reserved, pass	–
	A7	reserved, pass	DP slave or module is occupied (temporary error
	A8	version conflict	DP slave or module reports non-compatible versions
	A9	feature not supported	Function is not supported by DP slave or module
	AA to AF	user specific	DP slave or module reports a manufacturer-specific error in its application. Please check the documentation from the manufacturer of the DP slave or module.

Error_Decode (B#16#...)	Error_Code_1 (B#16#...)	Explanation according to DVP1	Meaning
	B0	invalid index	Data record not known in module Illegal data record number ≥ 256
	B1	write length error	The length specified in the RECORD parameter is incorrect; • With "RALRM (Page 6468)": length error in AINFO, • With "RDREC (Page 6459)" and "WRREC (Page 6461)": length error in MLEN
	B2	invalid slot	The configured slot is not occupied.
	B3	type conflict	Actual module type does not match specified module type
	B4	invalid area	DP slave or module reports access to an invalid area
	B5	state conflict	DP slave or module not ready
	B6	access denied	DP slave or module denies access
	B7	invalid range	DP slave or module reports an invalid range for a parameter or value
	B8	invalid parameter	DP slave or module reports an invalid parameter
	B9	invalid type	DP slave or module reports an invalid type • With "RDREC (Page 6459)": buffer too small (subsets cannot be read) • With "WRREC (Page 6461)": buffer too small (subsets cannot be written)
	BA to BF	user specific	DP slave or module reports a manufacturer-specific error when accessing. Please check the documentation from the manufacturer of the DP slave or module. Note on value B#16#BA: The following applies for PROFINET in the H system: If a data record job with the return value W#16#80BA is rejected, the job must be repeated.
	C0	read constrain conflict	With "WRREC (Page 6461)": the data can only be written when the CPU is in STOP mode. Note: this means that writing by the user program is not possible. You can only write the data online with PG/PC. With "RDREC (Page 6459)": the module routes the data record, but either no data is present or the data can only be read when the CPU is in STOP mode. Note: if data can only be read when the CPU is in STOP mode, then an evaluation by the user program is not possible. In this case, you can only read the data online with PG/PC.

Error_Decode (B#16#...)	Error_Code_1 (B#16#...)	Explanation according to DVP1	Meaning
	C1	write constrain conflict	The data of the previous write job on the module for the same data record have not yet been processed by the module.
	C2	resource busy	The module is currently processing the maximum possible number of jobs for a CPU.
	C3	resource unavailable	The required operating resources are currently occupied.
	C4		Internal temporary error. Job could not be carried out. Repeat the job. If this error occurs often, check your installation for sources of electrical interference.
	C5		DP slave or module not available.
	C6		Data record transfer was canceled due to priority class cancellation
	C7		Job aborted due to warm or cold restart on the DP master
	C8 bis CF		DP slave or module reports a manufacturer-specific resource error. Please check the documentation from the manufacturer of the DP slave or module.
	Dx	user specific	DP slave specific. Refer to the description of the DP slave.
8x (x = 1, ... 9, A, B, C, D, E, F)	00 to FF		Error in y-th call parameter (y = 1, ...15) • Error_Code_1 = 00: Illegal operating mode • For all other values of Error_Code_1 refer also to: Evaluating errors with output parameter RET_VAL (Page 5370)
FE, FF	00 bis FF		Profile-specific error

Field element STATUS[4]

With DPV1 errors, the DP master passes on STATUS[4] to the CPU and the instruction. Without DPV1 error, this value is set to 0, with the following exceptions for the "RDREC (Page 6459)" and "WRREC (Page 6461)" instructions:

- STATUS[4] contains the target range length from RECORD, if LEN > the target range length from RECORD
- STATUS[4]=LEN, if the actual data record length < LEN < the target range length from RECORD
- STATUS[4]=0, if STATUS[4] > 255 has to be set.

In PROFINET IO, STATUS[4] has the value 0.

Parameter TINFO

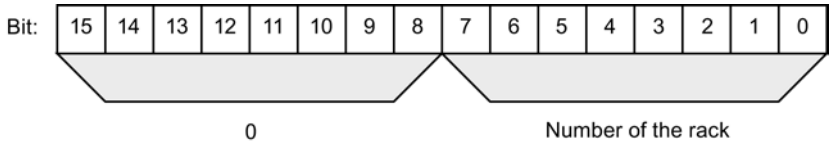
Data structure of the destination area TINFO

Byte	Meaning
0 to 19	Start information of the OB in which "RALRM" was currently called
20 and 21	Address, for exact description, see below
22 to 31	Management information, for exact description, see below

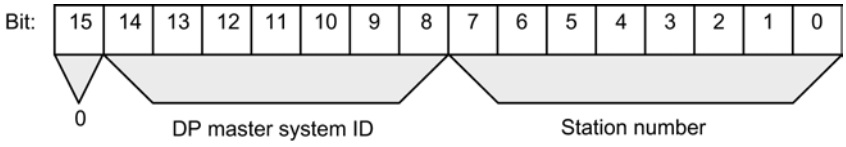
Structure of the address (bytes 20 and 21)

The address contains:

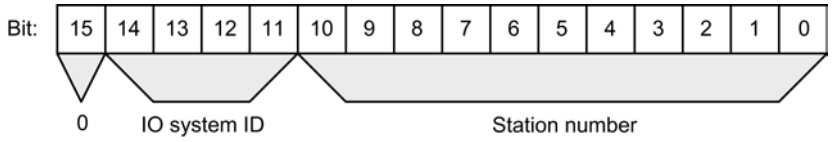
- In a central configuration, the rack number (0-31).



- In a distributed configuration with PROFIBUS DP
  - The DP master system ID (1-32)
  - The station number (0-127).



- In a distributed configuration with PROFINET IO:
  - The last two positions in the PROFINET IO system ID (0-15). To obtain the complete PROFINET IO system ID, you must add 100 (decimal) to it.
  - The station number (0-2047).



Structure of the management information in bytes 20 to 25

Byte no. for TINFO	Data type	Meaning				
20	BYTE	central:	0			
		distributed:	PROFIBUS DP: DP master ID: 1 to 32)			
			PROFINET IO: see above			
21	BYTE	central:	Module rack number (possible values: 0 to 31)			
		distributed:	Number of the DP station (possible values: 0 to 127)			
			PROFINET IO: see above			
22	BYTE	central:	• 0: Data record 0 or data record 1			
		distributed:	Bit 0 to 3:	Slave type	0000: 0001: 0010: 0011: 0100 – 0111: 1000: 1001 and higher:	DP (data record 0 structure) DPS7 (data record 0 or data record 1 structure) DPS7 V1 (data record 0 or data record 1 structure) DPV1 (structure acc. to PROFIBUS DP standard) reserved PROFINET IO (structure acc. to PROFINET IO Standard) reserved
			• Bit 4 to 7:	Profile type		Reserved
23	BYTE	central:	• 0			

Byte no. for TINFO	Data type	Meaning				
		distributed:	• Bit 0 to 3:	Interrupt info type	0000:	Transparent, which is always the case for PROFINET IO (interrupt originates from a configured distributed module)
					0001:	Representative (interrupt originates from a non-DPV1 slave/non IO device or a slot that is not configured)
					0010:	Generated (interrupt generated in the CPU)
					0011 and higher:	Reserved
			• Bit 4 to 7:	Structure version	0000: 0001 and higher:	Initial Reserved
24	BYTE	central:	• 0			
		distributed:	**Flags of the PROFIBUS DP master interface module/PROFINET IO controller master interface module**			
			Bit 0 = 0:	Interrupt originating from an integrated interface module (PROFINET IO or PROFIBUS DP)		
			Bit 0 = 1:	Interrupt originating from an external interface module (PROFINET IO or PROFIBUS DP)		
			Bit 1 to 7:	Reserved		
25	BYTE	central:	• 0			
		distributed:	**Flags of the PROFIBUS DP slave interface module**			
			• Bit 0:	EXT_DIAG_FLAG from the diagnostics message frame, or 0 if this bit does not exist in the interrupt The bit is 1 if the DP slave is faulty.		
			• Bit 1 to 7:	Reserved		
			Flags of the PROFINET IO controller interface module			
			• Bit 0:	ARDiagnosisState or 0 if there is no information in the interrupt. The bit is 1 if the IO device is faulty.		
			• Bit 1 to 7:	Reserved		

Structure of the management information in bytes 26 to 27 with PROFIBUS and a central configuration

Byte no. for TINFO	Data type	Meaning	
26 and 27	WORD	central:	0
	WORD	distributed:	PROFIBUS ID number as unique identifier of the PROFIBUS DP slave
28 and 29	WORD	0	(Bytes 28 and 29 can be omitted)
30 and 31	WORD	0	(Bytes 30 and 31 can be omitted)

Structure of the management information in bytes 26 to 31 with PROFINET IO

Byte no. for TINFO	Data type	Meaning	
26 and 27	WORD	distributed:	PROFINET IO device ID number as unique identifier of the PROFINET IO device
28 and 29	WORD	distributed:	Manufacturer ID
30 and 31	WORD	distributed:	ID number of the instance

Parameter AINFO

Data structure of the destination area AINFO with interrupts from PROFIBUS DP or central I/O devices

The information for PROFINET IO is provided below.

Byte	Meaning	
0 to 3	Header information, for exact description, see below	
4 to 199	Additional interrupt information: data for the respective interrupt:	
	central:	ARRAY[0] to ARRAY[195]
	distributed:	ARRAY[0] to ARRAY[59]

Structure of the header information with interrupts from PROFIBUS DP or central IO devices

Byte	Data type	Meaning	
0	BYTE	Length of the received interrupt information in bytes	
		central: distributed:	4 to 224 4 to 63
1	BYTE	central:	Reserved
		distributed:	ID for the interrupt type

Byte	Data type	Meaning		
			1: 2: 3: 4: 5: 6: 31	Diagnostics interrupt Process interrupt Removal interrupt Insertion interrupt Status interrupt Update Interrupt Failure of an expansion device, DP master system, or DP station
			32 to 126:	Manufacturer-specific interrupt
2	BYTE	Slot number of the component that triggered the interrupt		
3	BYTE	central:	Reserved	
		distributed:	Specifier	
			Bits 0 and 1	0: no further information; 1: Incoming event, faulty slot 2: Outgoing event, slot not faulty anymore 3: Outgoing event, slot still faulty
			Bit 2:	Add_Ack
			Bits 3 to 7:	Sequence number

Data structure of the destination area AINFO with interrupts from PROFINET IO

Byte	Meaning
0 to 25	Header information, for exact description, see below
26 to 1431	Additional interrupt information: Standardized diagnostics data for the respective interrupt: ARRAY[0] to ARRAY[1405] Note: The additional interrupt information may also be omitted.

Structure of the header information with interrupts from PROFINET IO

Byte	Data type	Meaning
0 and 1	WORD	• Bits 0 to 7: Block type • Bits 8 to 15: Reserved
2 and 3	WORD	Block length
4 and 5	WORD	Version: • Bits 0 to 7: low byte • Bits 8 to 15: high byte

Byte	Data type	Meaning
6 and 7	WORD	ID for interrupt type: • 1: Diagnostics interrupt (incoming) • 2: Process interrupt • 3: Remove module interrupt • 4: Insert module interrupt • 5: Status interrupt • 6: Update interrupt • 7: Redundancy interrupt • 8: Controlled by supervisor • 9: Released by supervisor • 10: Configured module not inserted • 11: Return of the sub-module • 12: Diagnostics interrupt (outgoing) • 13: Slave-to-slave connection alarm • 14: Neighborhood change alarm • 15: Clock synchronization message (bus end) • 16: Clock synchronization alarm (device end) • 17: Network component alarm • 18: Time synchronization alarm (bus end) • 19 to 31: Reserved • 32 to 127: Manufacturer-specific interrupt • 128 to 65535: Reserved
8 to 11	DWORD	API (Application Process Identifier)
12 to 13	WORD	Slot number of the component triggering the interrupt (range of values 0 to 65535)
14 to 15	WORD	Submodule slot number of the component triggering the interrupt (range of values 0 to 65535)
16 to 19	DWORD	Module identification; specific information on the source of the interrupt

Byte	Data type	Meaning
20 to 23	DWORD	Submodule identification; specific information on the source of the interrupt
24 to 25	WORD	Interrupt specifier: • Bits 0 to 10: Sequence number (range of values 0 to 2047) • Bit 11: Channel diagnostics: 0: No channel diagnostics available 1: Channel diagnostics information exists • Bit 12: Status of manufacturer-specific diagnostics: 0: No manufacturer-specific status information available 1: Manufacturer-specific status information available • Bit 13: Status of diagnostics for sub-module: 0: No status information available, all errors have been corrected 1: At least one item of channel diagnostics and/or status information is available • Bit 14: Reserved • Bit 15: Application relationship diagnosis state: – 0: None of the modules configured within this application relationship reports diagnostics information – 1: At least one of the modules configured in this AR is reporting diagnostics information

Structure of additional interrupt information with interrupts from PROFINET IO

The additional interrupt information for PROFINET IO depends on the format identifier. It can comprise multiple data blocks with the same or different format identifier. The following format identifiers are available:

- W#16#0000 to W#16#7FFF: Manufacturer-specific diagnostics

Byte	Data type	Meaning
0 to 1	WORD	Format identifier for the structure of the following data serving as additional interrupt information W#16#0000 to W#16#7FFF: Manufacturer-specific diagnostics
2 to n	BYTE	See manufacturer's manual.

- W#16#8000: Channel diagnostics
  Channel diagnostics is output in blocks of 6 bytes each. The additional interrupt information (without format identifier) is only output for disrupted channels.

Byte	Data type	Meaning	
0 to 1	WORD	Format identifier for the structure of the following data serving as additional interrupt information W#16#8000: Channel diagnostics	
2 to 3	WORD	Channel number of the component triggering the interrupt (range of values: 0 to 65535): • W#16#0000 to W#16#7FFF: Channel number of the interface module/sub-module • W#16#8000: The generic substitute for the entire sub-module • W#16#8001 to W#16#FFFF: Reserved	
4	BYTE	Bits 0 to 2:	Reserved
		Bits 3 to 4:	Type of error: • 0: Reserved • 1: Incoming error • 2: Outgoing error • 3: Outgoing error, other errors present
		Bits 5 to 7:	Type of channel: • 0: Reserved • 1: Input channel • 2: Output channel • 3: Input/output channel

Byte	Data type	Meaning
5	BYTE	Data format: • B#16#00: Free data format • B#16#01: Bit • B#16#02: 2 bits • B#16#03: 4 bits • B#16#04: Byte • B#16#05: Word • B#16#06: Double word • B#16#07: 2 double words • B#16#08 to B#16#FF: Reserved
6 to 7	WORD	Type of error: • W#16#0000: Reserved • W#16#0001: Short circuit • W#16#0002: Undervoltage • W#16#0003: Overvoltage • W#16#0004: Overload • W#16#0005: Overtemperature • W#16#0006: Wire break • W#16#0007: High limit exceeded • W#16#0008: Low limit exceeded • W#16#0009: Error • W#16#000A to W#16#000F: Reserved • W#16#0010 to W#16#001F: Manufacturer-specific • W#16#0020 to W#16#00FF: Reserved • W#16#0100 to W#16#7FFF: Manufacturer-specific • W#16#8000: Device diagnostics available • W#16#8001 to W#16#FFFF: Reserved Not all channels support every error type. For detailed information, refer to the description of the diagnostics data for the specific device.

Note

The section from "channel number" to "type of error" can occur from 0 to n times.

W#16#8001

W#16#8001: MULTIPLE (different types of diagnostics information are transmitted).

In this case, the additional interrupt information is transmitted as blocks of variable length.

Byte	Data type	Meaning
0 to 1	WORD	Format identifier for the structure of the following data serving as additional interrupt information W#16#8001: Manufacturer-specific diagnostics and/or channel diagnostics
2 to 3	WORD	Block type
4 to 5	WORD	Block length
6	BYTE	Version: high byte
7	BYTE	Version: low byte
8 to 11	DWORD	API (only if low byte of version = 1)
12 to 13	WORD	Slot number
14 to 15	WORD	Subslot number
16 to 17	WORD	Channel number:
18 to 19	WORD	Channel properties
20 to 21	WORD	Format identifier: • W#16#0000 to W#16#7FFF: Manufacturer-specific diagnostics • W#16#8000: Channel diagnostics • W#16#8002: Extended channel diagnostics • W#16#8003: Stepped extended channel diagnostics • W#16#8004 to W#16#80FF: Reserved
22 to n	BYTE	Data depend on the format identifier

Note

The section starting from "block type" can occur from 1 to n times.

W#16#8002

W#16#8002: Extended channel diagnostics

Byte	Meaning
0 to 1	Format identifier W#16#8002
2 to 3	Channel number:
4 to 5	Channel properties
6 to 7	Error type
8 to 9	Additional error value
10 to 13	Additional error information

W#16#8003

W#16#8003: Stepped extended channel diagnostics

Byte	Meaning
0 to 1	Format identifier W#16#8003
2 to 3	Channel number:

Byte	Meaning
4 to 5	Channel properties
6 to 7	Error type
8 to 9	Additional error value
10 to 13	Additional error information
14 to 17	Qualified channel qualifier

W#16#8100

W#16#8100: Maintenance information

Byte	Meaning
0 to 1	Format identifier W#16#8100
2 to 3	Block type
4 to 5	Block length
6 to 7	Block version
8 to 9	Reserved
10 to 13	Maintenance status

Note

You can find more detailed information about the structure of the additional alarm information in the *Programming Manual SIMATIC PROFINET IO from PROFIBUS DP to PROFINET IO* and the current version of IEC 61158-6-10-1.

Destination area TINFO and AINFO

Destination area TINFO and AINFO

Depending on the respective OB in which the instruction "RALRM" is called, the destination areas TINFO and AINFO are only partially written. Refer to the table below to find out which information is entered respectively.

Interrupt type	OB	TINFO OB status information	TINFO Management information	AINFO Header information	AINFO Additional interrupt information	
Process interrupt	4x	Yes	Yes	Yes	central:	No
					distributed:	as supplied by PROFIBUS DP slave/ PROFINET IO device
Status interrupt	55	Yes	Yes	Yes	Yes	Yes
Update interrupt	56	Yes	Yes	Yes	Yes	Yes

Interrupt type	OB	TINFO OB status information	TINFO Management information	AINFO Header information	AINFO Additional interrupt information	
Manufacturer-specific interrupt	57	Yes	Yes	Yes	Yes	Yes
I/O redundancy error	70	Yes	Yes	No	No	No
Diagnostics interrupt	82	Yes	Yes	Yes	central:	Data record 1
					distributed:	as supplied by PROFIBUS DP slave/ PROFINET IO device
Insert/remove interrupt	83	Yes	Yes	Yes	central:	No
					distributed:	as supplied by PROFIBUS DP slave/ PROFINET IO device
Special form of the remove module interrupt: Controlled by supervisor	83	Yes	Yes	Yes	PROFINET IO only	
Special form of the insert module interrupt: Enabled by supervisor	83	Yes	Yes	Yes	PROFINET IO only	
Unconfigured module inserted	83	Yes	Yes	Yes	PROFINET IO only	
Rack failure/ station failure	86	Yes	Yes	No	No	
... all other OBs		Yes	No	No	No	

D_ACT_DP: Enable/disable DP slaves

Description

Use the "D_ACT_DP" instruction to specifically deactivate and reactivate configured DP slaves/ PROFINET IO devices. In addition, you can determine whether each assigned DP slave or PROFINET IO device is currently activated or deactivated.

If you use the instruction to deactivate an IE/PB Link PN IO type of gateway, then all connected PROFIBUS DP slaves will also cease to function. These failures will be reported.

The instruction cannot be used on PROFIBUS PA field devices that are connected by a DP/PA link to a DP master system.

Note

As long as one or more "D_ACT_DP" jobs are active, you cannot load a changed configuration from the programming device to the CPU (within the scope of CiR).
When a changed configuration is loaded from the programming device to the CPU during ongoing operation (CiR), the CPU rejects activation of a "D_ACT_DP" job.

Several runs through the cycle control point are required to perform the disabling or enabling job. Therefore, you cannot wait for the end of such a job in a programmed loop.

Parameter

The following table shows the parameters of the instruction "D_ACT_DP":

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Level-triggered control parameter REQ=1: Execute activation or deactivation
MODE	Input	BYTE	I, Q, M, D, L or constant	Job identifier Possible values: • 0: Request information on whether the addressed component is activated or deactivated • 1: Activate the DP slave/PROFINET IO device • 2: Deactivate the DP slave/PROFINET IO device • 3: Activate DP slave/PROFINET IO device and call OB 86 after the change of the activation state has been completed • 4: Deactivate DP slave/PROFINET IO device and call OB 86 after the change of the activation state has been completed
LADDR	Input	WORD	I, Q, M, D, L or constant	Any logical address of the DP slave/PROFINET IO device
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
BUSY	Output	BOOL	I, Q, M, D, L	Active code: • BUSY=1: The job is still active. • BUSY=0: The job was terminated.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	Job was executed without errors.
0001	The DP slave/PROFINET IO device is active (this error code is possible only with MODE = 0.)
0002	The DP slave/PROFINET IO device is deactivated (this error code is possible only with MODE = 0.)
7000	First call with REQ=0. The job specified with LADDR is not active; BUSY has the value "0".
7001	First call with REQ=1. The job specified with LADDR was initiated. BUSY has the value "1".
7002	Intermediate call (REQ irrelevant). The activated job is still active; BUSY has the value "1".
8090	• You have not configured a module with the address specified in LADDR. • You operate your CPU as I-Slave and you have specified in LADDR an address of this I-Slave.
8092	The deactivation of the currently addressed DP slave/PROFINET IO device (MODE=2) cannot be canceled by being activated (MODE=1). Activate the component at a later time.
8093	No DP slave/PROFINET IO device is assigned to the address specified in LADDR (no configuration available) or the MODE parameter is not known.
8094	You have attempted to activate a device which is potential partner for a tool change port. However, another device is already activated on this tool change port at this time. The activated device will remain activated.
80A1	Parameters could not be assigned for the addressed component. (This error code is only available when MODE = 1.) Note: "D_ACT_DP" returns this error information only if the activated slave/device fails again during parameter assignment. If the parameter assignment of a single module was unsuccessful, "D_ACT_DP" returns the error information W#16#0000.
80A2	The addressed DP slave does not return an acknowledgment. (This error information is not available with PROFINET IO devices. The process is not time-monitored by PROFINET.)
80A3	The DP master/PROFINET IO controller concerned does not support this function.
80A4	The CPU does not support this function for external DP masters/PROFINET IO controller.
80A6	Slot error in the DP slave/PROFINET IO device; not all user data can be accessed (this error code is only available when MODE=1). Note: "D_ACT_DP" returns this error information only if the activated component fails again after parameter assignment and before the end of "D_ACT_DP". If only a single module is unavailable, "D_ACT_DP" returns the error information W#16#0000.
80C1	"D_ACT_DP" was started and will continue with another logical address (this error code is only available when MODE=1 and MODE=2).
80C3	• Temporary resource error: The CPU is currently processing the maximum possible activation and deactivation jobs. (This error code is available only when MODE = 1 and MODE = 2.) • The CPU is busy receiving a modified configuration. Currently you can not enable/disable DP slaves/ PROFINET IO devices.
80C5	DP: Jobs not collected by the user are discarded at restart.
80C6	PROFINET: Jobs not collected by the user are discarded at restart.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

Application

If you configure DP slaves/PROFINET IO devices in a CPU which are not actually present or not currently required, the CPU will nevertheless continue to access these DP slaves/ PROFINET IO devices at regular intervals. After the slaves are deactivated, further CPU accessing will stop. With PROFIBUS DP, the fastest possible DP bus cycle can be achieved in this manner and the corresponding error events no longer occur.

Examples

From a machine OEM's point of view, there are numerous device options possible in series production of machines. However, each delivered machine includes only one combination of selected options.

Every one of these possible machine options is configured as a DP slave/PROFINET IO device by the manufacturer in order to create and maintain a common user program having all possible options. Use "D_ACT_DP" to deactivate all DP slaves/PROFINET IO devices not present at machine startup.

A similar situation exists for machine tools having numerous tooling options available but actually using only a few of them at any given time. These tools are implemented as DP slaves/ PROFINET IO devices. With "D_ACT_DP", the user program activates the tools currently needed and deactivates those required later.

Functional description

"D_ACT_DP" works asynchronously, that is, its execution extends over multiple calls. You start the job by calling D_ACT_DP with REQ = 1.

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 5367).

Identification of a job

If you have started a deactivation or activation job and you call "D_ACT_DP" again before the job is complete, the behavior of the instruction depends on whether or not the new call involves the same job: If the input parameter LADDR matches, then the call will be interpreted as a follow-on call.

Deactivating DP slaves/PROFINET IO devices

When you deactivate a DP slave/PROFINET IO device with "D_ACT_DP", its process outputs are set to the configured substitute values or to 0 (safe state). The assigned DP master/ PROFINET IO controller does not continue to address this component. Deactivated DP slaves/ PROFINET IO devices are not identified as faulty or missing by the error LEDs on the DP master/PROFINET IO controller or CPU.

The process image of the inputs of deactivated DP slaves/PROFINET IO devices is updated with 0, that is, it is handled just as it is for failed DP slaves/PROFINET IO devices.

If you are using your program to directly access the user data of a previously deactivated DP slave/PROFINET IO device, the I/O access error OB (OB 122) is called, and the corresponding

start event is entered in the diagnostics buffer. If you attempt to access a deactivated DP slave/PROFINET IO device via an instruction (such as "RD_REC (Page 6491)"), you will receive the same error information in RET_VAL as for an unavailable DP slave/PROFINET IO device.

Deactivating a DP slave/PROFINET IO device does not start the program error OB (OB 85), even if its inputs or outputs belong to the system-side process image to be updated. Also there is no entry in the diagnostics buffer.

Whether the deactivation of a DP slave/PROFINET IO device leads to a start of the rack failure OB (OB 86) and an entry in the diagnostics buffer, depends on the MODE parameter:

- MODE = 2: No OB 86 start and no diagnostics buffer entry
- MODE = 4: OB 86 start and diagnostics buffer entry

If a DP station/PNIO station fails after you have deactivated it with "D_ACT_DP", the operating system does not detect the failure. As a result, there is no subsequent start of OB 86 or diagnostics buffer entry.

Applies to PROFIBUS DP: If you wish to deactivate DP slaves functioning as transmitters in slave-to-slave communication, we recommend that you first deactivate the receivers (listeners) that detect which input data the transmitter is transferring to its DP master. Deactivate the transmitter only after you have performed this step.

Activating DP slaves/PROFINET IO devices

When you reactivate a DP slave/PROFINET IO device with "D_ACT_DP", this component is configured and assigned parameters by the associated DP master/PROFINET IO controller (as with the return of a failed DP station/PROFINET IO station). This activation is complete when the component is able to transfer user data.

Activating a DP slave/PROFINET IO device does not start the program execution error OB (OB 85), even if its inputs or outputs belong to the system-side process image to be updated. Also there is no entry in the diagnostics buffer.

Whether the deactivation of a DP slave/PROFINET IO device leads to a start of the rack failure OB (OB 86) and an entry in the diagnostics buffer, depends on the MODE parameter:

- MODE = 1: No OB 86 start and no diagnostics buffer entry
- MODE = 3: OB 86 start and diagnostics buffer entry

If you attempt to use "D_ACT_DP" to activate a DP slave that has been deactivated and is physically separated from the DP bus, the instruction will return the error code W#16#80A2 after approximately one minute and the DP slave will remain deactivated. If the slave is reconnected to the DP bus at a later time, it must be reactivated with "D_ACT_DP".

If you attempt to activate a PROFINET IO device that is physically separated from the PROFINET bus, "D_ACT_DP" will remain active. There is no automatic cancellation after a specific period as with DP slaves. You need to manually cancel the running job.

Note

Activating a DP slave/PROFINET IO device may be time-consuming. If you wish to cancel a current activation job, start "D_ACT_DP" again with the same value for LADDR and MODE = 2. Repeat the call of "D_ACT_DP" with MODE = 2 until successful cancellation of the activation job is indicated by RET_VAL = 0.

If you wish to activate DP slaves which take part in the slave-to-slave communication, we recommend that you first activate the transmitters and then the receivers (listeners).

CPU startup

Depending on the startup mode, the CPU operating system behaves as follows regarding DP slaves/PROFINET IO devices:

- During cold and warm restarts, slaves/devices are activated automatically.
  - With S7-400: Activation of the DP slaves/PROFINET IO devices may take some time. In this case, the CPU continues to run and I/O access errors occur until the activation is complete. The reaction of the CPU depends on the configuration (OB 85 call for I/O access errors) and the program in OB 85. You cannot suppress these I/O access errors.
  - For S7-300: Activation of the DP slaves/PROFINET IO devices may take some time. In this case, the CPU waits until the DP slaves / PROFINET IO devices are activated. The maximum waiting period for activation of the DP slaves / PROFINET IO devices is approx. 1 minute. No I/O access errors occur during this time.
- During a hot restart, the activation status of slaves/devices remains unchanged: Activated slaves/devices remain activated, deactivated slaves/devices remain deactivated.

After the CPU start-up, the CPU cyclically attempts to contact all configured and not deactivated slaves/devices that are either not present or not responding.

Note

A call of "D_ACT_DP" in startup OBs is not supported.

Additional instructions

RD_REC: Read data record from I/O

Description

Use the instruction to read the data record with the number RECNUM from the addressed module. You start the read process by assigning the value "1" to the input parameter REQ during the call. If the read process could be executed immediately, the instruction returns the value "0" at the output parameter BUSY . If BUSY has the value "1", reading is not yet complete.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 5367). The data record read is entered in the destination area spanned by the RECORD parameter, providing the data transfer was free of errors.

Note

When you read out a data record with a number greater than one from an FM or a CP you have purchased prior to February 1997 (below referred to as "old modules"), "RD_REC" responds differently than it does in new modules. This special situation is covered in the section "Using old S7-300 FMs and CPs with data record numbers >1" (see below).

If a DPV1 slave is configured via GSD file (GSD rev. 3 and higher) and the DP interface of the DP master is set to "S7 compatible", then you may not read any data records from the I/O modules in the user program with "RD_REC". In this case, the DP master addresses the wrong slot (configured slot + 3).

Remedy: Set the interface of the DP master to "DPV1".

Parameter

The following table shows the parameters of the instruction "RD_REC":

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = 1: Read request
IOID	Input	BYTE	I, Q, M, D, L or constant	Address area identifier: • B#16#54 = Peripheral input (PI) • B#16#55 = Peripheral output (PQ) If the module is a mixed module, you will have to specify the area identifier of the lower address. If the addresses are identical, you will have to specify B#16#54.
LADDR	Input	WORD	I, Q, M, D, L or constant	Logical address of the module. For a mixed module, the lower of the two addresses must be specified.
RECNUM	Input	BYTE	I, Q, M, D, L or constant	Data record number (permitted values: 0 to 240)
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code. In addition: the length of the data record actually transferred in bytes (possible values: +1 to +240), if the destination area is greater than the transferred data record and if no error occurred during the transfer.

Parameters	Declaration	Data type	Memory area	Description
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The reading process is not yet complete.
RECORD	Output	ANY	I, Q, M, D, L	Destination area for the data record read. With asynchronous execution of "RD_REC", make sure that the actual parameters of RECORD have the same length information for all calls. Only the BYTE data type is permitted. Note: Note that for S7-300 CPUs the parameter RECORD always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and leads to an error message in the user program.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RECORD

Note

If you want to ensure that the entire data record is always read, select a destination area with a length of 241 bytes. If the data transfer is error-free, RET_VAL contains the actual data record length.

Using old S7-300 FMs and CPs with data record numbers > 1

If you want to use the instruction "RD_REC" to read out a data record with a number greater than one from an old S7-300 FM or old S7-300 CP, remember the following points:

- If the destination area is greater than the actual length of the required data record, no data are entered in RECORD . RET_VAL is written with W#16#80B1.
- If the destination area is smaller than the actual length of the required data record, the CPU reads as many bytes beginning at the start of the record as are specified in the length information of RECORD and enters this number of bytes inRECORD . RET_VAL has the value "0".
- If the length specified in RECORD is the same as the actual length of the required data record, the CPU reads the data record and enters it in RECORD . RET_VAL is written with "0".

Parameter RET_VAL

- If an error has occurred while the function was being executed, the return value contains an error code.
- If no error occurred during the transfer, RET_VAL contains the following:
  - 0, if the entire destination area was filled with data from the selected data record (the data record can also be incomplete).
  - The length of the data record actually transferred in bytes (possible values: +1 to +240) if the destination area is greater than the transferred data record.

Note

If the general error W#16#8745 occurs, this only indicates that access to at least one byte of the process image was blocked. The data record was read by the module correctly and written to the I/O memory area.

When looking at the "real" error information (error codes W#16#8xyz) in the following table, we distinguish between two cases:

- Temporary errors (error codes W#16#80A2 to 80A4, 80Cx):
  This type of error can possibly be eliminated without user action, meaning it is helpful to call the instruction again (multiple calls, if necessary).
  Example of a temporary error: Resources required are currently in use (W#16#80C3).
- Permanent errors (error codes W#16#809x, 80A0, 80A1, 80Bx):
  This type of error does not correct itself. A new call of the instruction will not be successful until you have eliminated the error. Example of a permanent error: Wrong length specification in RECORD (W#16#80B1).

Note

If you transfer data records to a DPV1 slave with "WR_REC (Page 6496)" or if you read data records from a DPV1 slave with RD_REC and if this DPV1 slave operates in DPV1 mode, the DP master evaluates the error information it received from the slave as follows:

If the error information lies within the range from W#16#8000 to W#16#80FF or W#16#F000 to W#16#FFFF, the DP master passes the error information to the instruction. If the error information is outside this range, the DP master passes the value W#16#80A2 to the instruction and suspends the slave.

For a description of the error information originating from DPV1 slaves, refer to STATUS[3] Parameter STATUS (Page 6471).

Parameter RET_VAL for WR_REC and RD_REC

Error code (W#16#...)	Explanation	Restriction
0000	No error	-
7000	First call with REQ=0: No data transfer active; BUSY has the value 0.	-

Error code (W#16#...)	Explanation	Restriction
7001	First call with REQ=1: Data transfer started; BUSY has the value 1.	Distributed I/O
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value 1.	Distributed I/O
8090	Specified logical base address invalid: There is no assignment in SDB1/SDB2x or there is no base address.	-
8092	A type other than BYTE is specified in the ANY reference.	For S7-400 only
8093	This instruction is not permitted for the module specified by LADDR and IOID . (permitted are S7-300 modules for an S7-300, S7-400 modules for an S7-400, S7-DP modules for an S7-300 and S7-400).	-
80A0	Negative acknowledgement when reading from the module: Module was removed during the reading process or is defective	With "RD_REC" only
80A1	Negative acknowledgement when writing to the module: Module was removed during the writing process or is defective	With "WR_REC (Page 6496)" only
80A2	• DP protocol error at layer 2 (for example, slave failure or bus problems) • For ET200S, data record cannot be read in DPV0 mode.	Distributed I/O
80A3	DP protocol error with user interface/user	Distributed I/O
80A4	Communication on the communication bus disrupted	Error occurs between the CPU and the external DP interface module.
80B0	• Instruction not possible for module type. • The module does not recognize the data record. • Data record number 241 not permitted. • With "WR_REC (Page 6496)", data records 0 and 1 are not permitted.	-
80B1	The length specified in parameter RECORD is incorrect.	• With "WR_REC (Page 6496)": Length incorrect • With "RD_REC" (only when using old S7-300 FMs and S7-300 CPs): specified length > data record length • With DPNRM_DG: specified length > record length
80B2	The configured slot is not occupied.	-
80B3	Actual module type does not match the specified module type in SDB1.	-
80B7	DP slave or module reports an invalid range for a parameter or value.	With "RD_REC" only

Error code (W#16#...)	Explanation	Restriction
80C0	With "WR_REC (Page 6496)": the data can only be written when the CPU is in STOP mode. Note: this means that writing by the user program is not possible. You can only write the data online with PG/PC. With "RD_REC": the module routes the data record, but either no data is present or the data can only be read when the CPU is in STOP mode. Note: if data can only be read when the CPU is in STOP mode, then an evaluation by the user program is not possible. In this case, you can only read the data online with PG/PC. With "DPNRM_DG (Page 6523)": There are no diagnostics data available.	With "WR_REC (Page 6496)" or "RD_REC" or "DPNRM_DG (Page 6523)"
80C1	The data of the previous write job on the module for the same data record have not yet been processed by the module.	-
80C2	The module is currently processing the maximum possible number of jobs for a CPU.	-
80C3	The required resources (memory, etc.) are currently occupied.	-
80C4	Internal temporary error. Job could not be carried out. Repeat the job. If this error occurs often, check your installation for sources of electrical interference.	-
80C5	Distributed I/O not available.	Distributed I/O
80C6	Data record transfer stopped due to priority class abort (restart or background)	Distributed I/O
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)	-

WR_REC: Write data record to I/O

Description

Use the instruction "WR_REC" to transfer the data record RECORD to the addressed module.

You start the writing process by assigning the value "1" to the input parameter REQ during the call. If the writing process could be executed immediately, the instruction returns the value "0" at the output parameter BUSY. If BUSY has the value "1", writing is not yet complete.

Note

If a DPV1 slave is configured via GSD file (GSD rev. 3 and higher) and the DP interface of the DP master is set to "S7 compatible", then you may not write any data records from the I/O modules to the user program with "WR_REC". In this case, the DP master addresses the wrong slot (configured slot + 3).

Remedy: set the interface of the DP master to "DPV1".

Parameters

The following table shows the parameters of the instruction "WR_REC":

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = 1: Write request
IOID	Input	BYTE	I, Q, M, D, L or constant	Address area identifier: • B#16#54 = Peripheral input (PI) • B#16#55 = Peripheral output (PQ) If the module is a mixed module, you will have to specify the area identifier of the lower address. If the addresses are identical, you will have to specify B#16#54.
LADDR	Input	WORD	I, Q, M, D, L or constant	Logical address of the module. For a mixed module, the lower of the two addresses must be specified.
RECNUM	Input	BYTE	I, Q, M, D, L or constant	Data record number (permitted values: 2 to 240)
RECORD	Input	ANY	I, Q, M, D, L	Data record. Only the BYTE data type is permitted. Note: Note that for S7-300 CPUs the parameter RECORD always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and leads to an error message in the user program.
RET_VAL	Output	Return	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY= 1: The writing process is not yet complete.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RECORD

The data to be transferred are read from the RECORD parameter during the first call. If the transfer of the data record takes longer than the duration of one call, the contents of the RECORD parameter are no longer relevant for the subsequent instruction calls (for the same job).

Parameter RET_VAL

See also: RD_REC: Read data record from I/O (Page 6491)

Note

If the general error W#16#8544 occurs, it only indicates that access to at least one byte of the I/O memory area containing the data record was denied. The data transfer was continued.

DPRD_DAT: Read consistent data of a DP standard slave

Description

You use the instruction "DPRD_DAT" to consistently read data of a DP standard slave/ PROFINET IO device; the maximum length is as follows:

- See the documentation supplied with your CPU for the maximum length.
- If there was no error during the data transmission, the data that have been read are entered in the destination area indicated by RECORD.

The destination area must have the same length that you configured for the selected module. If you read from a DP standard slave with a modular configuration or with several DP identifiers, you can only access the data of one module/DP identifier at the configured start address per "DPRD_DAT" call.

Note

CPUs of the S7-300/400 product range support up to 64 bytes of consistent data. Use instruction "DPRD_DAT" to handle consistent data ranges that exceed 4 bytes. However, you may also use this instruction for a data range of 1 byte. Access errors trigger the output of error code W#16#8090.

Parameter

The following table shows the parameters of the instruction "DPRD_DAT":

Parameters	Declaration	Data type	Memory area	Description
LADDR	Input	WORD	I, Q, M, D, L or constant	Configured start address from the I-area of the module from which the data will be read. Note: Always set the address in hexadecimal format, e.g. start address 100 means: LADDR:=W#16#64.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
RECORD	Output	ANY	I, Q, M, D, L	Destination area for the user data that were read. It must be precisely as long as you have configured for the selected module. Only the BYTE data type is permitted. Note: Note that for S7-300 CPUs the parameter RECORD always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and leads to an error message in the user program.

For more information about the valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Note

If you access DPV1 slaves, error information of these slaves can be forwarded from the DP master to the instruction. For a description of this error information, refer to STATUS[3] Parameter STATUS (Page 6471).

Error code (W#16#...)	Explanation
0000	No error occurred.
8090	• You have not configured a module for the specified logical base address or • You have ignored the restriction concerning the length of consistent data, or • You have not entered the start address in the LADDR parameter in hexadecimal format.
8092	A type other than BYTE is specified in the ANY reference.
8093	No DP module/PROFINET IO device from which you can read consistent data exists at the logical address specified in LADDR.
80A0	Access error detected while I/O devices were being accessed.

Error code (W#16#...)	Explanation
80B0	Slave failure on external DP interface module.
80B1	The length of the specified destination area is not identical to the configured user data length.
80B2	System error with external DP interface module.
80B3	System error with external DP interface module.
80C0	The data haven't yet been read by the module.
80C2	System error with external DP interface module.
80Fx	System error with external DP interface module.
87xy	System error with external DP interface module.
808x	System error with external DP interface module.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

Application

You require "DPRD_DAT" because you can only read out a maximum of four continuous bytes using the load commands that access the I/O or the process image input table.

Note

If required, you can also read consistent data via the process image of the inputs.

- See the associated documentation to determine whether your S7-300 CPU handles this functionality.
- All S7-400 CPUs handle this functionality.

CAUTION

I/O access

When using "DPRD_DAT", avoid accessing I/O areas that have process image partitions with OB6x connections (isochronous mode interrupts) assigned to them.

Data consistency

See also: Section Data consistency (Page 7358).

DPWR_DAT: Write consistent data of a DP standard slave

Description

Use the instruction to consistently transfer the data in RECORD to the addressed DP standard slave/PROFINET IO device, and, if necessary, to the process image (if you have configured the relevant address area of the DP standard slave as a consistent range in a process image).

The following applies for the maximum length of the data that will be transferred: See the documentation supplied with your CPU for the maximum length. The data is transferred synchronously, that is, the write process is completed when the instruction is completed.

The source area must have the same length that you have configured for the selected module. If the DP standard slave has a modular design, you can only access one module of the DP slave.

Note

CPUs of the S7-1200/1500 product range support up to 64 bytes of consistent data. Use instruction "DPWR_DAT" to write consistent data ranges that exceed 4 bytes to a submodule or device. However, you may also use this instruction for a data range of 1 byte. Access errors trigger the output of error code W#16#8090.

Parameter

The following table shows the parameters of the instruction "DPWR_DAT":

Parameters	Declaration	Data type	Memory area	Description
LADDR	Input	WORD	I, Q, M, D, L or constant	Configured start address from the PIQ area of the module to which the data is written. Note: Address must be specified in hexadecimal form, for example, start address 100 means: LADDR:=W#16#64.
RECORD	Input	ANY	I, Q, M, D, L	Source area for the user data to be written. It must be precisely as long as you have configured for the selected module. Only the BYTE data type is permitted. Note: Note that for S7-300 CPUs the parameter RECORD always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and leads to an error message in the user program.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.

For more information about the valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Note

If you access DPV1 slaves, error information of these slaves can be forwarded from the DP master to the instruction. A description of this error information is available in STATUS[3] Parameter STATUS (Page 6471).

Error code (W#16#...)	Explanation
0000	No error occurred.
808x	System error with external DP interface module.
8090	• You have not configured a module for the specified logical base address or • You have ignored the restriction concerning the length of consistent data, or • You have not entered the start address in the LADDR parameter in hexadecimal format.
8092	A type other than BYTE is specified in the ANY reference.
8093	No DP module/PROFINET IO device to which you can write consistent data exists at the logical address specified in LADDR.
80A1	Access error detected while I/O devices were being accessed.
80B0	Slave failure on external DP interface module.
80B1	The length of the specified destination area is not identical to the configured user data length.
80B2	System error with external DP interface module.
80B3	System error with external DP interface module.
80C1	The data of the previous write job on the module have not yet been processed by the module.
80C2	System error with external DP interface module.
80Fx	System error with external DP interface module.
85xy	System error with external DP interface module.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

Application

You require the instruction "DPWR_DAT" because you can only write a maximum of four continuous bytes using the transfer commands that access the I/O or the process image output table.

Note

If required, you can also read consistent data via the process image of the inputs.
See the associated documentation to determine whether your S7-300 CPU handles this functionality,

- All S7-400 CPUs handle this functionality.
- Do not use both possibilities concurrently when writing consistent data: Either use "DPWR_DAT" or write via the process image output table.

CAUTION

I/O access

When using "DPWR_DAT", avoid accessing I/O areas that have process image partitions with OB6x connections (isochronous mode interrupts) assigned to them.

Data consistency

See also: Data consistency (Page 7358).

I-device / I-slave

RCVREC: Receive data record

Description

An I-device can receive a data record from a superordinate controller. The receipt takes place in the user program with the instruction "RCVREC" (receive record).

The instruction has the following operating modes:

- Check whether the I-device has a request for a data record receipt.
- Make the data record available to the output parameters.
- Send an answer to the superordinate controller.

You can determine the operating mode executed by the instruction using the input parameter MODE (see below).

The I-device must be in the operating mode RUN or STARTUP.

With MLEN, you specify the maximum number of bytes you want to receive. The selected length of the destination area RECORD should have at least the length of MLEN bytes.

If a data record was received (MODE=1 or MODE=2), the output parameter NEW will indicate that the data record was stored in RECORD. Note that RECORD has a sufficient length. The output parameter LEN contains the actual length of the data record received in bytes.

Set CODE1 and CODE2 to zero for the positive answer to the superordinate controller. If the received data record is to be rejected, enter the negative answer to the superordinate controller in Error Code 1 of the CODE1 and in Error Code 2 of the CODE2.

Note

If the I-device has received a request for a data record receipt, you must recognize the delivery of this request within a certain duration. After recognition, you must send an answer to the superordinate controller within this time period. Otherwise the I-device will experience a timeout error which causes the operating system of the I-device to send a negative answer to the superordinate controller. For information on the value for the time period, please refer to the specifications of your CPU.

The output parameter STATUS receives the error information after the occurrence of an error.

Operating modes

You can determine the operating mode of the instruction "RCVREC" with the input parameter MODE. This step will be explained in the following table.

MODE	Meaning
0	Check whether a request for a data record receipt exists If a data record from a superordinate controller exists on the I-device, the instruction will only write the output parameters NEW, SLOT, INDEX and LEN. If you call the instruction several times with MODE=0, then the output parameter will only refer to one and the same request.
1	Receiving a data record for any subslot of the I-device If a data record from a superordinate controller exists on the I-device for any subslot of the I-device, the instruction will write the output parameter and transfer the data record to the parameter RECORD.
2	Receiving a data record for a specific subslot of the I-device If a data record from a superordinate controller exists on the I-device for a specific subslot of the I-device, the instruction will write the output parameter and transfer the data record to the parameter RECORD.
3	Sending a positive answer to the superordinate controller The instruction checks the request of the superordinate controller to receive a data record, accepts the existing data record and sends a positive acknowledgement to the superordinate controller.
4	Sending a negative answer to the superordinate controller The instruction checks the request of the superordinate controller to receive a data record, rejects the existing data record and sends a negative acknowledgement to the superordinate controller. Enter the reason for the rejection in the input parameters CODE1 and CODE2.

Note

After the receipt of a data record (NEW=1) you must call "RCVREC" twice to ensure complete processing. You must do this in the following order:

- First call with MODE=1 or MODE=2
- Second call with MODE=3 or MODE=4

Parameter

The following table shows the parameters of the instruction "RCVREC":

Parameters	Declaration	Data type	Memory area	Description
MODE	Input	INT	I, Q, M, D, L or constant	Mode
F_ID	Input	DWORD	I, Q, M, D, L or constant	Subslot in the transfer area of the I-device for the data record to be received (only relevant for MODE=2). The high word is always set to zero.
MLEN	Input	INT	I, Q, M, D, L or constant	Maximum length of the data record to be received in bytes
CODE1	Input	BYTE	I, Q, M, D, L or constant	Zero (for MODE=3) and/or Error Code 1 (for MODE=4)
CODE2	Input	BYTE	I, Q, M, D, L or constant	Zero (for MODE=3) and/or Error Code 2 (forMODE=4)
NEW	Output	BOOL	I, Q, M, D, L	• MODE=0: New data record was received • MODE=1 or 2: Data record was transferred to RECORD
STATUS	Output	DWORD	I, Q, M, D, L	Error information
SLOT	Output	INT	I, Q, M, D, L	Identical to F_ID
SUBSLOT	Output	INT	I, Q, M, D, L	Identical to F_ID
INDEX	Output	INT	I, Q, M, D, L	Number of the data record received
LEN	Output	INT	I, Q, M, D, L	Length of the data record received
RECORD	InOut	ANY	I, Q, M, D, L	Destination area for the data record received. Note: Note that for S7-300 CPUs the parameter RECORD always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 Byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and results in an error message in the user program.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter STATUS

For information about the interpretation of parameter STATUS, refer to section: Parameter STATUS (Page 6471)

PRVREC: Make data record available

Description

An I-device can receive a request from a superordinate controller to make a data record available. The I-device makes the data record available in the user program with the instruction "PRVREC" (provide record).

The instruction has the following operating modes:

- Check whether the I-device has a request for making a data record available.
- Transfer the requested data record to the superordinate controller.
- Sending an answer to the superordinate controller.

You can determine the operating mode executed by the instruction using the input parameter MODE(see below).

The I-device must be in the operating mode RUN or STARTUP.

Enter the maximum number of bytes the data record to be sent should have with LEN. The selected length of the destination area RECORD should have at least the length of LEN bytes.

If a request to make a data record available exists, (MODE=0), the output parameter NEW will be set to TRUE.

If the request for making a data record available is accepted, write RECORD for the positive answer to the superordinate controller with the requested data record and write zero for CODE1 and CODE2. If the request for making a data record available is to be rejected, enter the negative answer to the superordinate controller in Error Code 1 of the CODE1 and in Error Code 2 of theCODE2.

Note

If the I-device has received a request for making a data record available, you must recognize the delivery of this request within a certain time period. After recognition, you must send an answer to the superordinate controller within this time period. Otherwise the I-device will experience a timeout error which causes the operating system of the I-device to send a negative answer to the superordinate controller. For information on the value for the time period, please refer to the specifications of your CPU.

The output parameter STATUS receives the error information after the occurrence of an error.

Operating modes

You can determine the operating mode of the instruction "PRVREC" with the input parameter MODE. This step will be explained in the following table.

MODE	Meaning
0	Check whether a request for making a data record available exists If a request from a superordinate controller for making a data record available exists on the I-device, the instruction will only write the output parameters NEW, SLOT, INDEX and RLEN. If you call the instruction several times with MODE=0, then the output parameter will only refer to one and the same request.
1	Receiving a request for making a data record available for any subslot of the I-device If such a request from a superordinate controller for any subslot of the I-device exists on the I-device, the instruction will write the output parameter.
2	Receiving a request for making a data record available for a specific subslot of the I-device If such a request from a superordinate controller for a specific subslot of the I-device exists on the I-device, the instruction will write the output parameter.
3	Make the data record available and send a positive answer to the superordinate controller The instruction checks the request of the superordinate controller to make a data record available, makes the request data record available to RECORD and sends a positive acknowledgement to the superordinate controller.
4	Sending a negative answer to the superordinate controller The instruction checks the request of the superordinate controller to make a data record available, rejects this request and sends a negative acknowledgement to the superordinate controller. Enter the reason for the rejection in the input parameters CODE1 and CODE2.

Note

After the receipt of a request (NEW=1) you must call the instruction twice to ensure complete processing. You must do this in the following order:

- First call with MODE=1 or MODE=2
- Second call with MODE=3 or MODE=4

Parameter

The following table shows the parameters of the instruction "PRVREC":

Parameters	Declaration	Data type	Memory area	Description
MODE	Input	INT	I, Q, M, D, L or constant	Mode
F_ID	Input	DWORD	I, Q, M, D, L or constant	Subslot in the transfer area of the I-device for the data record to be sent (only relevant for MODE=2). The high word is always set to zero.
CODE1	Input	BYTE	I, Q, M, D, L or constant	Zero (for MODE=3) and/or Error Code 1 (forMODE=4)
CODE2	Input	BYTE	I, Q, M, D, L or constant	Zero (for MODE=3) and/or Error Code 2 (forMODE=4)

Parameters	Declaration	Data type	Memory area	Description
LEN	Input	INT	I, Q, M, D, L or constant	Maximum length of the data record to be sent in bytes
NEW	Output	BOOL	I, Q, M, D, L	The new data record was requested by the superordinate controller.
STATUS	Output	DWORD	I, Q, M, D, L	Error information
SLOT	Output	INT	I, Q, M, D, L	Identical to F_ID
SUBSLOT	Output	INT	I, Q, M, D, L	Identical to F_ID
INDEX	Output	INT	I, Q, M, D, L	Number of the data record to be sent
RLEN	Output	INT	I, Q, M, D, L	Length of the data record to be sent
RECORD	InOut	ANY	I, Q, M, D, L	Data record made available Note: Note that for S7-300 CPUs the parameter RECORD always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 Byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and results in an error message in the user program.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter STATUS

For information about the interpretation of parameter STATUS, refer to section: Parameter STATUS (Page 6471)

SALRM: Send interrupt

Description SALRM

Description

You use the instruction to send an interrupt of a slot in the transition area (virtual slot) from the user program of an intelligent slave to the associated DP master. This starts the associated OB in the DP master.

You can send additional interrupt-specific information along with the interrupt. You can use the instruction to read all additional information in the DP master.

The instruction can only be used in S7-compatible mode.

- DP: The master uses a GSD file to integrate the I-slave.
- S7-compatible: The I-slave is connected to a master during configuration.

Note

The interface of the "SALRM" instruction is identical to the "SALRM" FB defined in "PROFIBUS Guideline PROFIBUS Communication and Proxy Function Blocks according to IEC 61131-3".

Functional description

The "SALRM" instruction works asynchronously, that is, its execution extends over multiple calls. You start the interrupt transfer by calling the instruction with REQ = 1.

The sending process remains active until interrupt processing is either acknowledged or canceled by the DP master.

The output parameter BUSY and bytes 2 and 3 of the output parameter STATUS show the status of the job. Bytes 2 and 3 of STATUS match the output parameter RET_VAL of the instructions that operate asynchronously.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 5367).

The transfer of the interrupt is complete when the output parameter BUSY has the value FALSE .

Identification of the job

When you initiate the transfer of an interrupt to the DP master with the instruction "SALRM" and then call this instruction once again before the current job has been completed, the further response of this instruction will depend on whether or not the new call involves the same job.

If the ID and ATYPE parameters match a still unfinished job, the new call is regarded as a follow-on call.

Parameters

The following table shows the parameters of the instruction "SALRM":

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = 1: Transfer the interrupt
ID	Input	DWORD	I, Q, M, D, L or constant	Any logical address of the transition area to the DP master (virtual slots), as viewed from the DP slave, except the diagnostics address of the station and the logical address of slot 2. The relevant information is found in the low word. Enter zero in the high word. Bit 15 contains the I/O ID: 0 = input address, 1 = output address.

Parameters	Declaration	Data type	Memory area	Description
ATYPE (Page 6512)	Input	INT	I, Q, M, D, L or constant	Interrupt type ID for the interrupt type. Possible values: • 1: Diagnostics interrupt • 2: Process interrupt
ASPEC (Page 6512)	Input	INT	I, Q, M, D, L or constant	Interrupt specifier: • 0: No further information • 1: Incoming event, faulty slot • 2: Outgoing event, slot no longer faulty • 3: Outgoing event, slot still faulty
LEN (Page 6513)	Input	INT	I, Q, M, D, L or constant	Length of additional interrupt information to be sent, in bytes Highest value: 16
DONE	Output	BOOL	I, Q, M, D, L	DONE=1: Interrupt was transmitted.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: Interrupt transmission is not yet complete.
ERROR (Page 6513)	Output	BOOL	I, Q, M, D, L	ERROR = 1: An error has occurred.
STATUS (Page 6513)	Output	DWORD	I, Q, M, D, L	Error information
AINFO (Page 6514)	InOut	ANY	I, Q, M, D, L	Interrupt info Source area for additional interrupt information Note: Note that for S7-300 CPUs, the parameter AINFO always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and leads to an error message in the user program.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Effect of the "SALRM" call on the module status information and the group error LED (SF)

Like every CPU, an intelligent slave has stored the properties of its slots in the module status information.

See also: SZL-ID W#16#xy91 - module status information (Page 6718).

When you use "SALRM" to send a diagnostic interrupt, the operating system at the intelligent slave influences the slave-local module status information and the SF LED based on bit 0 in byte 0 in AINFO (this bit is added to the status information as "module faulty"). However, there is no diagnostics buffer entry made in the I-slave, and no diagnostics interrupt OB is started.

Consistency of module status information between DP master and I-slave

The following section presents different scenarios and discusses their effects on module status information:

- Station return (results in start of OB 86 in the DP master and in the I-slave)
  The module status information will be affected with I-slave and with S7 master ("module fault" will be reset). If, after a station return, there are faults at the I-slave from the point of view of the user, these must be reported to the DP master by means of an "SALRM" call.
- STOP-RUN transition of the DP master (results in start of OB 82 in the I-slave)
  The module status information on the I-slave remains unchanged. The DP master resets the "module fault" bit in the relevant module status information.
  To ensure the consistency of module status information between the DP master and I-slave in S7-compatible mode, you must respond to the I-slave as follows:
  - For each error-free virtual slot, you use "SALRM" to send an outgoing diagnostics interrupt to the DP master.
  - For each faulty virtual slot, you use "SALRM" to send an incoming diagnostics interrupt to the DP master.
- STOP-RUN transition of the I-slave (results in start of OB 82 in the DP master)
  The module status information on the DP master remains unchanged and is reset on the I-slave ("module fault" will be reset).
  To ensure the consistency of module status information between the DP master and I-slave, you must react to the I-slave as follows:
  - For each error-free virtual slot, you use "SALRM" to send an outgoing diagnostics interrupt to the DP master.
  - For each faulty virtual slot, you use "SALRM" to send an incoming diagnostics interrupt to the DP master.

Note

Because "SALRM" functions asynchronously, the "SALRM" calls cannot be completed in the startup OBs. In other words, they must be run in the cyclical program until they are finished.

All the above-mentioned differences between the module status information in the master and the I-slave can only occur at those slots that receive diagnostics interrupts by means of "SALRM". This means that the remedies discussed above only apply to such slots.

Parameter ATYPE

Parameter ATYPE

For all permitted values of ATYPE , the following table shows which OB is started in the associated DP master and in which DP mode the given interrupt type is permitted.

ATYPE	Meaning in DPV1 standard	Associated OB in the S7 DP master	DP mode	
			DP	S7-compatible
1	Diagnostics interrupt	Diagnostics interrupt OB (OB82)	–	Yes
2	Process interrupt	Process interrupt OBs (OBs 40 to 47)	–	Yes

Note

In addition to the information given in the table above, the extent to which interrupt types can be used can also be restricted by the DP master.

Dependency of the interrupt type on the operating mode of the S7 master

For a slave in S7-compatible mode (operated on an S7 master), hardware and diagnostics interrupts can only be sent when the associated DP master is in RUN mode (DP: Operate). If the DP master is in STOP mode (DP: Clear), the interrupt is not sent and "SALRM" provides the error information W#16#80C8. In this case, the user is responsible for sending the interrupt at a later time.

Parameter ASPEC

Parameter ASPEC

According to the applicable standard, this parameter shows the diagnostics status of the virtual slot. For this reason, you can assign a value other than zero to ASPEC only when sending a diagnostics interrupt.

Because the additional S7 interrupt information for a diagnostics interrupt (data record 0) contains incoming/outgoing information (see Diagnostics data>Main byte 0 bit 0), you must write bit 0 (module fault) in byte 0 of the additional interrupt information as follows:

ASPEC	"Module fault" bit in AINFO (Page 6514)
0	0
1	1
2	0
3	1

Parameter LEN

Parameter LEN

In LEN, you specify in bytes the length of the additional interrupt information to be sent. The maximum permitted range of values is 0 to 16.

The following table shows the values LEN can have in the individual modes of an intelligent slave for all possible interrupt types.

Interrupt type	DP	S7-compatible
Diagnostics interrupt	–	4 to 16
Process interrupt	–	4

The following table shows the response of "SALRM" when you assign LEN a value other than the length of AINFO (Page 6514) in BYTE.

Value of LEN	Characteristics of "SALRM"
<= length specification of AINFO	"SALRM" sends an interrupt to the DP master. The number of bytes of additional interrupt information transmitted is as specified in LEN.
Outside the permitted range of values (< 0 or > 16)	"SALRM" does not send an interrupt. Error information: W#16#80B1, STATUS[4]=B#16#FF
> Length specification of AINFO	"SALRM" sends an interrupt to the DP master. The amount of bytes of additional interrupt information transmitted is as specified in the length information of AINFO. Error information: W#16#00B1, STATUS[4]= length specification of AINFO

Parameters STATUS and ERROR

Parameters STATUS and ERROR

The STATUS output parameter contains error information. If it is interpreted as ARRAY[1 ... 4] OF BYTE, the error information has the following structure:

Array element	Meaning
STATUS[1]	• B#16#00: No error • B#16#C0: Error detected by I-slave
STATUS[2], STATUS[3]	Corresponds to the output parameter RET_VAL of instructions
STATUS[4]	B#16#00 with the exception of some length conflicts between LEN and the length of AINFO (Page 6514). These exceptions are shown in the next table.

The following table indicates all specific error information of "SALRM".

ERROR	STATUS[2,3] (W#16#...)	Explanation
0	0000	Job was executed without errors. If LEN is < length of AINFO, only LEN bytes of additional interrupt information were transmitted.
0	00B1	LEN > length of AINFO. The job was completed. The additional interrupt information specified in AINFO was transmitted. STATUS[4] contains the length of AINFO.
0	7000	Initial call with REQ=0 (empty cycle). No interrupt was sent. BUSY has the value "0".
0	7001	Initial call with REQ=1. The job was initiated. BUSY has the value "1".
0	7002	Intermediate call (REQ irrelevant). The interrupt sent was not yet acknowledged by the DP master. BUSY has the value "1".
1	8090	The address specified in the ID is outside the permitted address range or was not configured.
1	8091	• You have disabled the interrupt in the configuration. • The interrupt is not permitted for this type of slave.
1	8092	Illegal data type in AINFO (BYTE and BLOCK-DB are permitted)
1	8093	ID belongs to a virtual slot from which an interrupt cannot be requested.
1	80B0	ASPEC • Does not match bit 0 in byte 0 of AINFO • Must have a value of 0 for the interrupt type used • Is outside the permitted range of values
1	80B1	LEN Is outside the permitted range of values. STATUS[4] contains B#16#FF.
1	80B5	Call of "SALRM" in the DP master is not permitted.
1	80C3	The required resources (memory, etc.) are occupied at this time.
1	80C5	Distributed I/O device is not available at this time (e.g., station failure)
1	80C8	The function is not permitted in the current DP master operating mode (the DP master is an S7 master and is in STOP mode).

Parameter AINFO

Parameter AINFO

The AINFO parameter is the source area for additional interrupt information. As far as the intelligent slave is concerned, you can write any values to this area. However, if you are using a DP master of the S7 family, the additional information sent along with the interrupt must conform to S7 conventions.

If you send a **diagnostics interrupt** (ATYPE (Page 6512) =1), then you are responsible for appropriate assignment of data record 0 and, if necessary, data record 1.

The following table shows you a recommendation for an S7-compatible assignment. In this case, the "module fault" bit (see above) was already set. Except for the indicated bit, this recommendation corresponds to the default assignment (the one existing after a POWER UP, after a STOP-RUN transition of the intelligent slave, or a station return).

Data record no.	Assignment
0	B#16#01, 0B, 00, 00
1	For S7-compatible mode: data record 0 + 12 bytes with zero

Additional information

See also: Structure of diagnostic data (Page 6631).

PROFIBUS

DP_PRAL: Trigger hardware interrupt from DP standard slave

Description

Use the instruction to trigger a process interrupt on the associated DP master from the user program of an intelligent slave. This interrupt starts OB 40 on the DP master.

Using the input parameter AL_INFO , you can identify the cause of the process interrupt. This interrupt identifier is transferred to the DP master and you can evaluate the identifier in OB 40 (tag OB40_POINT_ADDR).

The requested process interrupt is uniquely specified by the input parameters IOID and LADDR . For each configured address area in the transfer memory, you can trigger exactly one process interrupt at any time.

Functional description

"DP_PRAL" works asynchronously, that is, its execution extends over multiple calls. You start the process interrupt request by calling "DP_PRAL" with REQ = 1.

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 5367).

The job is complete when execution of OB 40 is complete on the DP master.

Note

If you operate the DP slave as a standard slave, the job is complete as soon as the diagnostics frame is obtained by the DP master.

Identification of a job

The input parameters IOID and LADDR uniquely specify the job.

If you have called the "DP_PRAL" instruction on a DP slave and you call this instruction again before the DP master has acknowledged the requested process interrupt, the way in which the instruction reacts depends on whether the new call involves the same job: If the parameters IOID and LADDR match a job that is not yet complete, the call is interpreted as a follow-on

call regardless of the value of the parameter AL_INFO and the value W#16#7002 is entered in RET_VAL .

Parameter

The following table shows the parameters of the instruction "DP_PRAL":

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ=1: Trigger process interrupt on the associated DP master
IOID	Input	BYTE	I, Q, M, D, L or constant	Identifier of the address range in the transfer memory (from the point of view of the DP slave): • B#16#00: Bit15 of LADDR specifies whether an input (Bit15=0) or output address (Bit15=1) will be present. • B#16#54 = Peripheral input (PI) • B#16#55 = Peripheral output (PQ) If the module is a mixed module, you will have to specify the area identifier of the lower address. If the addresses are identical, you will have to specify B#16#54.
LADDR	Input	WORD	I, Q, M, D, L or constant	Start address of the address range in the transfer memory (from the point of view of the DP slave). If this is a range belonging to a mixed module, the lower of the two addresses must be specified.
AL_INFO	Input	DWORD	I, Q, M, D, L or constant	Interrupt ID This is transferred to OB 40 that is to be started on the associated DP master (tag OB40_POINT_ADDR). If you operate the intelligent slave with a remote master, you must evaluate the diagnostics frame on the master.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY=1: The triggered process interrupt has not yet been acknowledged by the DP master.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	Job was executed without errors.
7000	First call with REQ=0. No process interrupt request is active; BUSY has the value "0".
7001	First call with REQ=1. A process interrupt request has already been sent to the DP master; BUSY has the value "1".
7002	Intermediate call (REQ irrelevant): The triggered process interrupt has not yet been acknowledged by the DP master; BUSY has the value "1".
8090	Start address of the address range in the transfer memory is incorrect.
8091	Interrupt is blocked (block configured by user)
8093	The parameters IOID and LADDR address a module that is not capable of a process interrupt request.
80B5	Call in the DP master not permitted.
80C3	The required resources (memory, etc.) are occupied at this time.
80C5	Distributed I/O device is not available at this time (i.e. station failure).
80C8	The function is not permitted in the current DP master operating mode.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

DPSYC_FR: Synchronize DP slaves / Freeze inputs

Description

You use the instruction to synchronize one or more groups of DP slaves.

The function involves sending one of the control commands below, or a combination of them, to the relevant groups:

- SYNC (simultaneous output and freezing of output states on the DP slaves)
- UNSYNC (cancels the SYNC control command)
- FREEZE (freeze the input states on the DP slaves and read in the frozen inputs)
- UNFREEZE (cancels the FREEZE control command)

Before you send the control commands listed above, you must assign the DP slaves to groups per configuration. You must know which DP slave is assigned to which group, with which number, and know the reactions of the various groups to SYNC/FREEZE.

Note

Note that the control commands SYNC and FREEZE also remain valid when you perform a warm/cold restart.

Please note also that you may initiate only one SYNC/UNSYNC job or only one FREEZE/UNFREEZE job at a time.

Functional description

"DPSYC_FR" works asynchronously, that is, its execution extends over multiple calls. You start the job by calling "DPSYC_FR" with REQ=1.

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 5367).

Identification of a job

If you have triggered a SYNC/FREEZE job and called "DPSYC_FR" again before the first job was completed, the response of the instruction depends on whether the new call is for the same job. If the input parameters LADDR, GROUP and MODE match, the call is interpreted as a follow-on call.

Writing outputs of DP modules

The writing of outputs of DP modules is triggered as follows:

- By transfer commands to the DP I/O
- By writing the process image output table to the modules (by the operating system at the end of OB 1 or by calling the instruction "UPDAT_PO (Page 6454)")
- Calling the "DPWR_DAT (Page 6500)" instruction.

In normal operation, the DP master transfers the output bytes cyclically (within the cycle of the PROFIBUS DP bus) to the outputs of the DP slaves.

If you want to have certain output data (possibly distributed on several slaves) applied to the outputs to the process at exactly the same time, you can send the SYNC control command to the relevant DP master using the "DPSYC_FR" instruction.

What are the effects of SYNC?

With the SYNC control command, the DP slaves of the selected groups are switched to Sync mode. In other words, the DP master transfers the current output data and instructs the DP slaves involved to freeze their outputs. With the following output message frames, the DP

slaves enter the output data in an internal buffer and the state of the outputs remains unchanged.

Following each SYNC control command, the DP slaves of the selected groups apply the output data of their internal buffer to the outputs on the process.

The outputs are only updated cyclically again when you send the UNSYNC control command using the "DPSYC_FR" instruction.

Note

If the DP slaves of the selected group(s) are not currently connected to the network or have failed when the control command was sent, they will not be switched to SYNC mode. This information will not be communicated in the return value of the instruction.

Reading input data of DP modules

The input data of the DP modules are read as follows:

- Using load commands to the DP I/O
- When the process image input table is updated (by the operating system at the start of OB 1 or by calling the "UPDAT_PI (Page 6452)" instruction)
- By calling the "DPRD_DAT (Page 6498)" instruction.

In normal operation, the DP master receives this input data cyclically (within the cycle of the PROFIBUS DP bus) from its DP slaves and makes them available to the CPU.

If you want to have certain input data (possibly distributed on several slaves) to be read from the process at exactly the same time, send the FREEZE control command to the relevant DP master using the "DPSYC_FR" instruction.

What are the effects of FREEZE?

With the FREEZE control command, the DP slaves involved are switched to Freeze mode, in other words the DP master instructs the DP slaves to freeze the current state of the inputs. It then transfers the frozen data to the input area of the CPU.

Following each FREEZE control command, the DP slaves freeze the state of their inputs again.

The DP master only receives the current state of the inputs cyclically again after you have sent the UNFREEZE control command with the "DPSYC_FR" instruction.

Note

If the DP slaves of the selected group(s) are not currently connected to the network or have failed when the control command has been sent, they will not be switched to FREEZE mode. This information will not be communicated in the return value of the instruction.

Data consistency

Because the DPSYC_FR functions are asynchronous and can be interrupted by higher priority classes, you should make sure that the process images are consistent with the actual inputs and outputs of the I/O when using the "DPSYC_FR" instruction.

This is guaranteed if you comply with the following consistency rules:

- Define suitable process image partitions for the "SYNC outputs" and the "FREEZE inputs" (only possible on the S7-400). Call the "UPDAT_PO (Page 6454)" instruction immediately before each first call of a SYNC job. Call the "UPDAT_PI (Page 6452)" instruction immediately after the respective last call of a FREEZE job.
- As an alternative: Use only direct I/O access for outputs involved in a SYNC job and for inputs involved in a FREEZE job. Do not write to these outputs when a SYNC job is active, and do not read in these inputs when a FREEZE job is active.

Use of DPWR_DAT and DPRD_DAT

If you use the "DPWR_DAT (Page 6500)" instruction, it must be complete before you send a SYNC job for the outputs involved.

If you use the "DPRD_DAT (Page 6498)" instruction, it must be complete before you send a FREEZE job for the inputs involved.

Startup and "DPSYC_FR"

The user alone must take responsibility for sending the SYNC and FREEZE control commands in the startup OBs.

If you want the outputs of one or more groups to be in SYNC mode when the user program starts, you must initialize these outputs during startup and completely execute the "DPSYC_FR" instruction with the SYNC control command.

If you want the inputs of one or more groups to be in FREEZE mode when the user program starts, you must execute the "DPSYC_FR" instruction with the FREEZE control command completely for these inputs during startup.

Parameter

The following table shows the parameters of the instruction "DPSYC_FR":

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Level-triggered control parameter REQ=1: Triggering of the SYNC/FREEZE job
LADDR	Input	WORD	I, Q, M, D, L or constant	Logical address of the DP master

Parameters	Declaration	Data type	Memory area	Description
GROUP	Input	BYTE	I, Q, M, D, L or constant	Group selection Bit 0 = 1: group 1 selected Bit 1 = 1: group 2 selected : Bit 7 = 1: group 8 selected You can select several groups per job. The value B#16#0 is invalid.
MODE	Input	BYTE	I, Q, M, D, L or constant	Job ID (coding complies with EN 50 170 Volume 2, PROFIBUS) Bit 0: reserved (value 0) Bit 1: reserved (value 0) Bit 2: • = 1: UNFREEZE will be executed • = 0: no meaning Bit 3: • = 1: FREEZE will be executed • = 0: no meaning Bit 4: • = 1: UNSYNC will be executed • = 0: no meaning Bit 5: • = 1: SYNC will be executed • = 0: no meaning Bit 6: reserved (value 0) Bit 7: reserved (value 0) Possible values: • with exactly one ID per job: – B#16#04 (UNFREEZE) – B#16#08 (FREEZE) – B#16#10 (UNSYNC) – B#16#20 (SYNC) • with more than one ID per job: – B#16#14 (UNSYNC, UNFREEZE) – B#16#18 (UNSYNC, FREEZE) – B#16#24 (SYNC, UNFREEZE) – B#16#28 (SYNC, FREEZE)
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code. Make sure that you evaluate RET_VAL each time the block has been executed.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY=1: The SYNC/FREEZE job is not yet complete.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Note

If you access DPV1 slaves, error information of these slaves can be forwarded from the DP master to the instruction. For a description of this error information, refer to STATUS[3], STATUS (Page 6471) parameter.

Error code (W#16#...)	Explanation
0000	Job was executed without errors.
7000	First call with REQ=0. The job specified with LADDR, GROUP and MODE is not active; BUSY has the value 0.
7001	First call with REQ=1. The job specified with LADDR, GROUP and MODE was triggered; BUSY has the value 1.
7002	Intermediate call (REQ irrelevant). The activated SYNC/FREEZE job is still running; BUSY has the value 1.
8090	The module selected with LADDR is not a DP master.
8093	This instruction is not permitted for the module selected with LADDR (configuration or version of the DP master).
8094	GROUP parameter is incorrect
8095	MODE parameter is incorrect
80B0	The group selected with GROUP is not configured.
80B1	The group selected with GROUP is not assigned to this CPU.
80B2	The SYNC job specified with MODE is not permitted on the group selected with GROUP.
80B3	The FREEZE job specified with MODE is not permitted on the group selected with GROUP .
80C2	Temporary shortage of resources on the DP master: The DP master is currently processing the maximum number of jobs for a CPU.
80C3	This SYNC/UNSYNC job cannot be activated at present because only one SYNC/UNSYNC job can be triggered at a time. Check your user program.
80C4	This FREEZE/UNFREEZE job cannot be activated at present because only one FREEZE/UNFREEZE job can be triggered at a time. Check your user program.
80C5	Short circuit directly at DP interface
80C6	Job aborted due to I/O disconnection by CPU

Error code (W#16#...)	Explanation
80C7	Job aborted due to warm or cold restart on the DP master
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

DPNRM_DG: Read diagnostics data from a DP slave

Description

You use the "DPNRM_DG" instruction to read the current diagnostics data of a DP slave in the form specified in EN 50 170 Volume 2, PROFIBUS. The data that has been read is entered in the target range indicated by RECORD following error-free data transfer.

Refer to the following table for the basic structure of the slave diagnostics data and to the manuals of the DP slaves for further information.

Byte	Meaning
0	Station status 1
1	Station status 2
2	Station status 3
3	Master station number
4	Vendor ID (high byte)
5	Vendor ID (low byte)
6 ...	Additional slave-specific diagnostic information

Functional description

The reading process is executed asynchronously, in other words, it can extend over several calls. The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 5367).

Call

You start the read process by assigning the value "1" to the input parameter REQ when the instruction is called.

Parameter

The following table shows the parameters of the instruction "DPNRM_DG":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = 1: Read request
LADDR	Input	WORD	I, Q, M, D, L or constant	Configured diagnostic address of the DP slave Note: Address must be specified in hexadecimal form, for example, diagnostic address 1022 means: LADDR:=W#16#3FE.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code. If no error has occurred, the length of the data actually transferred is entered in RET_VAL .
RECORD	Output	ANY	I, Q, M, D, L	Destination area for the diagnostics data that were read. Only the BYTE data type is permitted. The minimum length of the data record to be read or the target range is 6. The maximum length of the data record to be sent is 240. Standard slaves can provide more than 240 bytes of diagnostics data up to a maximum of 244 bytes. In this case, the first 240 bytes are transferred to the target range and the overflow bit is set in the data. Note: Note that for S7-300 CPUs the parameter RECORD always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and leads to an error message in the user program.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY=1: The reading process is not yet complete.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

You will find information on data type conversion in the different programming languages under "Overview of data type conversion (Page 2286)".

Parameter RECORD

The CPU evaluates the actual length of the read diagnostics data:

If the length specified for RECORD

- is less than the number of data bytes supplied, the data are discarded and a corresponding error code is entered in RET_VAL.
- is greater than or equal to the number of data bytes supplied, the data are accepted in the destination area and the actual length is entered in RET_VAL as a positive value.

Note

You must ensure that the actual parameters of RECORD correspond in all calls belonging to a job.

A job is uniquely identified by the LADDR input parameter.

RET_VAL parameter

- If an error occurs while the function is being executed, the return value contains an error code.
- If no errors have occurred during data transfer, RET_VAL contains the length of the data read in bytes as a positive number.

Note

The amount of data read in a DP slave depends on its diagnostics status.

For the evaluation of the error information of the RET_VAL parameter, refer to Evaluating errors with output parameter RET_VAL (Page 5370). This chapter also contains the general error information for the instructions. The specific error information of "DPNRM_DG" is a subset of the error information of "RD_REC (Page 6491)".

Standard slaves with more than 240 bytes of diagnostics data

With standard slaves on which the number of standard diagnostics data is between 241 and 244 bytes, note the following points:

If the length specified for RECORD

- is less than 240 bytes, the data are discarded and a corresponding error code is entered in RET_VAL.
- If the length specified for RECORD is greater than or equal to 240 bytes, the first 240 bytes of the standard diagnostics data are transferred to the target range and the overflow bit is set in the data.

System resources for S7-400

When "DPNRM_DG" is called for a job that is not currently processed, CPU resources (memory space) will be occupied for S7-400. You can call "DPNRM_DG" for multiple DP slaves briefly one after the other if you do not exceed the maximum number of "concurrently" active "DPNRM_DG" jobs for your CPU.

If you activate several jobs "simultaneously," all the jobs will be executed without interfering with each other.

If you reach the limits of the system resources, this is indicated in RET_VAL. In this case, repeat the job.

See also

Overview of data type conversion (Page 2286)

DP_TOPOL: Determine topology for DP master system

Description

You use the instruction to trigger the topology determination for a selected DP master system. Calling the instruction will address all diagnostics repeaters on a DP master system.

Note

The topology can only be determined for one DP master system at a time.

Topology determination is the prerequisite for the detailed display of the error location if line errors occur. After configuration and after every change to the physical configuration of a DP master system, you must repeat the topology determination with "DP_TOPOL".

Changes to the physical configuration are:

- Changes in line lengths
- Adding or removing stations or components with repeater function
- Changing station addresses

If an error is reported by a diagnostics repeater, "DP_TOPOL" writes the DPR and DPRI outputs for the duration of one "DP_TOPOL pass. If errors are reported by multiple diagnostics repeaters of the selected DP master system, "DP_TOPOL" writes DPR and DPRI with information regarding the first diagnostics repeater reporting the error. You can read out the entire diagnostic information on the programming device or using the "DPNRM_DG (Page 6523)" instruction. If no diagnostics repeater reports an error, the DPR and DPRI outputs have the value NULL.

If you want to repeat a topology determination after an error occurs, you must first reset "DP_TOPOL". This is done by calling "DP_TOPOL" with REQ=0 and R=1.

Functional description

"DP_TOPOL" works asynchronously, that is, its execution extends over multiple calls. You start determining the bus topology by calling "DP_TOPOL" with REQ=1 . If you want to cancel the process, call "DP_TOPOL" with R=1 .

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 5367).

Note

Determining the topology may take several minutes.

Identification of a job

The input parameter DP_ID uniquely specifies a job.

If you have called "DP_TOPOL" and you call this instruction again before the topology is re-determined, the manner in which the instruction reacts depends on whether the new call involves the same job: If the DP_ID parameter matches a job that has not yet been completed, then the call is interpreted as a follow-up call and the value W#16#7002 will be entered in RET_VAL. On the other hand, if another job is involved, the CPU rejects it.

Parameters

The following table shows the parameters of the "DP_TOPOL" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ=1: Trigger topology determination
R	Input	BOOL	I, Q, M, D, L	R=1: Cancel topology determination
DP_ID	Input	INT	I, Q, M, D, L or constant	DP master system ID of those master systems for which the topology will be determined
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY=1: Topology determination is not yet complete.
DPR	Output	BYTE	I, Q, M, D, L	PROFIBUS address of the diagnostics repeater reporting the error.
DPRI	Output	BYTE	I, Q, M, D, L	Measuring segment diagnostics repeater reporting the error: • Bit 0 = 1: Temporary faults on segment DP2 • Bit 1 = 1: Permanent faults on segment DP2 • Bit 4 = 1: Temporary faults on segment DP3 • Bit 5 = 1: Permanent faults on segment DP3

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

When looking at the "real" error information (error codes W#16#8xyz) in the following table, we distinguish between two cases:

- Temporary errors (error codes W#16#80A2 to 80A4, 80C3, 80C5):
  It is possible to eliminate this type of error without user action; in other words, it is advisable to call "DP_TOPOL" again (multiple calls, if necessary).
  Example of a temporary error: Resources required are currently in use (W#16#80C3).
- Permanent errors (error codes W#16#8082, 80B0, 80B2):
  This type of error does not correct itself. A new call of "DP_TOPOL" is only advisable after you have eliminated the error. Example of a permanent error: The DP master/CPU does not support this service. (W#16#80B0).

Error code (W#16#...)	Explanation
0000	Job completed without error.
7000	First call with REQ=0. Topology determination is not triggered. BUSY has the value "0".
7001	First call with REQ=1. The request to execute topology determination was sent. BUSY has the value "1".
7002	Intermediate call (REQ irrelevant): Topology determination is not yet complete. BUSY has the value "1".
7010	You have attempted to cancel topology identification. But there is no running job with the specified DP_ID. BUSY has the value "0".
7011	First call with R=1. The cancellation of the topology determination was triggered. BUSY has the value "1".
7012	Intermediate call: The cancellation of the topology determination is not yet complete. BUSY has the value "1".
7013	Final call: The topology determination was cancelled. BUSY has the value "0".
8082	No DP master system is configured with the specified DP_ID.
80A2	Error during topology determination; for more detailed information, refer to output parameters DPR and DPRI.
80A3	Error during topology determination: Monitoring time has elapsed (timeout).
80A4	Communication on the communication bus disrupted
80B0	The DP master/CPU does not support this service.
80B2	Error during topology determination: No diagnostics repeater was found at the selected DP master system.
80C3	Resources required are currently in use. Possible cause: You have initiated a second topology determination (only one topology determination is permitted at one time).
80C5	The DP master system is currently not available.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

ASi

ASi_3422: Control AS-i master behavior

Description

The instruction "ASi_3422" is necessary to use the operating mode "Extended Operation" with CP 342-2. The extended operation allows complete control of the master behavior using the user program. The access to the inputs and outputs takes place as before in the standard operation of the CP 342-2.

Command calls to the CP342-2 occur from the user program via the instruction. You will specify a command call in a send buffer and start the job.

The instruction "ASi_3422" transfers the command call to the CP342-2. The job status will be transferred with the conclusion of the job and possible answer data will be provided in a receive buffer.

Call

The instruction must be cyclically called for each CP342-2. Only one job can be processed at a time per CP342-2. A current job cannot be interrupted and the instruction will not monitor the time. The instruction "ASi_3422" is executed uninterruptedly. Calls can thus not be programmed in program priority classes which interrupt each other (for example, with calls in OB 1 and in OB 35).

Configure the command processing in the user program as follows:

1. When restarting the S7 user program, call "ASi_3422" once with the parameter value STARTUP = TRUE.
2. Specify the command call in a send buffer of the user program. Transfer this send buffer with the call parameter SEND.
   You can find more information about the command interfaces and the ASi slave command in the hardware documentation of the communication processor.
3. You may need an answer buffer depending on the command type. Transfer this answer buffer with the call parameter RECV. The answer buffer is not necessary for status information with this interface.
4. Activate the job using parameter ACT=1.
5. Then query the parameters DONE, ERROR and STATUS.

Parameters

The following table shows the parameters of the "ASi_3422" instruction:

Parameter	Declaration	Data type	Memory area	Description
ACT	Input	BOOL	I, Q, M, D, L	The command processing by the function occurs level-triggered, meaning that a command processing will be started if a call is not already being processed as long as ACT = 1 is active.
STARTUP	Input	BOOL	I, Q, M, D, L	The instruction will indicate a CPU startup with STARTUP = 1. After the first run, the function STARTUP must be reset by the user.
LADDR	Input	WORD	I, Q, M, D, L or constant	Module start address The module start address is to be determined using the information regarding the slot-oriented issuing of addresses to signal modules.
SEND	Input	ANY	I, Q, M, D, L	Send buffer The parameter refers to a memory area in which the command is to be specified by the user, for example: P#DB20.DBX 20.0 Byte 10.
RECV	Input	ANY	I, Q, M, D, L	Receive buffer This buffer is only relevant for commands which deliver answer data. The parameter refers to a memory area in which a command answer is stored. The length of the data area parameterized here is irrelevant, for example: P#DB30.DBX 20.0 Byte 1
DONE	Output	BOOL	Q, M, D, L	DONE = 1 signals ‘job complete without error'.
ERROR	Output	BOOL	Q, M, D, L	ERROR = 1 signals ‘job complete with errors'.
STATUS	InOut	DWORD	M, D	1. Word: Job status / error code (see following table); an error code will be generated for 'job complete with error' for more specific error description. 2. Word: Is required by the instruction for internal purposes and may not be changed. For instruction calls to different logical addresses (LADDR) different double words must be assigned for the parameter STATUS.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter LADDR

The start address of the address area is determined by the slot of CP 342-2.

The CP 342-2 occupies 16 input bytes and 16 output bytes in I/O address area of the S7 automation device (analogous area of the AS). 31 x 4 bits of the 16 byte address area of the CP 342-2 are occupied by the AS-i slave data. The remaining 4 bits are reserved for later applications.

Parameters DONE, ERROR and STATUS

If an error occurs during the processing of the function, a "0" will be in the BR bit in addition to the information in ERROR and STATUS mentioned above. The querying of the BR bit is different for LAD and STL user programs:

- LAD: Query using output parameter ENO
- STL: Direct query of the BR bit

The following table contains the possible display in the first word of STATUS dependent upon DONE and ERROR.

DONE	ERROR	STATUS	Meaning
0	0	8181	Job running.
1	0	0000	Job complete with errors.
0	1	8090	Address at parameter LADDR invalid.
0	1	80A0	Negative acknowledgement when reading from the module.
0	1	80A1	Negative acknowledgement when writing to the module.
0	1	80B0	The module does not recognize the data record.
0	1	80B1	Data record length entered is incorrect.
0	1	80C0	Cannot read data record.
0	1	80C1	The requested data record is still being processed.
0	1	80C2	Jobs are backed up.
0	1	80C3	Resources (memory) occupied.
0	1	80C4	Communication error
0	1	8182	Meaning: Identification after restart with STARTUP=TRUE (no errors).
0	1	8184	Data type of formal operand RECV illegal.
0	1	8381	Slave address incorrect
0	1	8382	Slave is not activated (not in LAS).
0	1	8383	Error in AS interface
0	1	8384	Command (in status of CP) illegal.
0	1	8385	Slave 0 exists.
0	1	83A1	Slave with address to be changed not found at AS interface.
0	1	83A2	Slave 0 present.
0	1	83A3	Slave with new address already present at AS interface.
0	1	83A4	Slave address cannot be deleted.
0	1	83A5	Slave address cannot be set.
0	1	83A6	Slave address cannot be permanently set.
0	1	83F8	Job number unknown.
0	1	83F9	EEPROM error
0	1	8F22	Range length error when reading a parameter.
		8F23	Range length error when writing a parameter. This error code indicates that a parameter is located either entirely or partly outside the range of an address, or that the length of a bit range is not a multiple of 8 with an ANY parameter.
0	1	8F24	Range error when reading a parameter.

DONE	ERROR	STATUS	Meaning
		8F25	Range error when writing a parameter. This error code indicates that a parameter is located in a range that is illegal for a system function.
0	1	8F28	Alignment error when reading a parameter.
		8F29	Alignment error when writing a parameter. This error code indicates that the reference to parameter is an operand with bit address that is not equal to "0".
0	1	8F30	Parameter is in the read-only global DB.
		8F31	Parameter is in the read-only instance DB. This error code indicates that a parameter is located in a read-only data block.
0	1	8F32	Parameter contains a DB number which is too large
0	1	8F3A	The parameter contains the number of a DB that is not loaded.
0	1	8F42	An access error occurred while the system was attempting to read a parameter from the peripheral input area.
		8F43	An access error occurred while the system was attempting to write a parameter to the peripheral output area.
0	1	8F44	This error code indicates that the read access to the parameter has been denied.
		8F45	This error code indicates that the write access to the parameter has been denied.
0	1	8F7F	Internal error

ASI_CTRL: Control AS-i master behavior

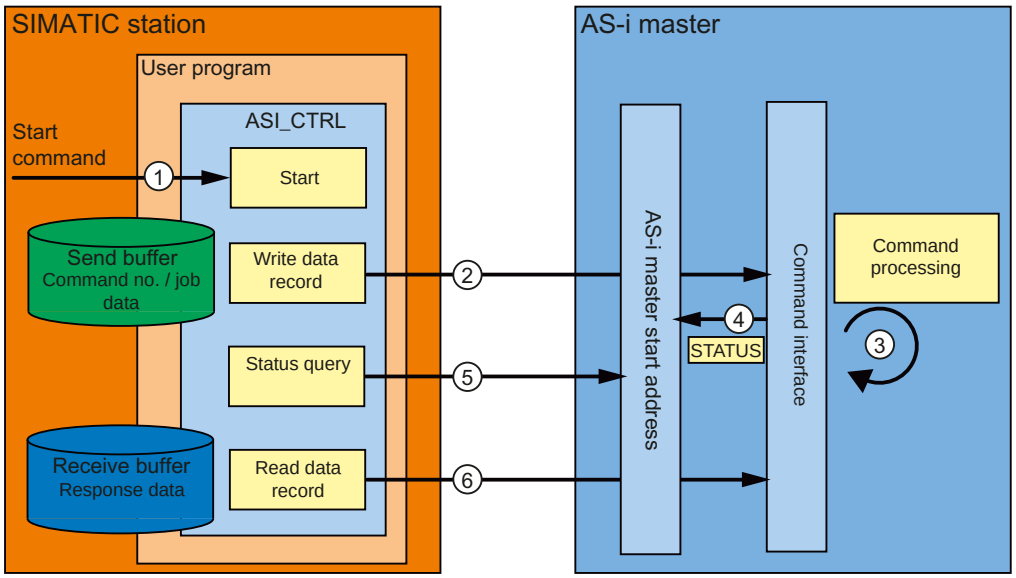
Description of ASI_CTRL

Description

Using the instruction "ASI_CTRL", you can control the behavior of the AS-i master from the user program of the PLC. The instruction processes the command protocol automatically. It also enables parameter assignment on SIMATIC AS-i masters and reading out information data. The functions and operation of the command interface are detailed in the manual for the AS-i master.

Both centrally inserted AS-i masters and distributed AS-i masters via PROFIBUS DP are supported. Combinations with PROFINET IO (e.g. IE/PB Link PN IO) are also possible.

The schematic diagram below shows the functions of the "ASI_CTRL" instruction:



① Start of processing at the REQ parameter.

② The program sends the required command to the AS-i master via the instruction "RDREC".

③ The AS-i master processes the command.

④ The current status of the AS-i master is stored in the input area of the binary data (logical start address).

⑤ The instruction "ASI_CTRL" cyclically queries and evaluates the 4 status bits.

⑥ Once command processing is complete, the command job is completed with "RDREC". Depending on the command, the data field of "RDREC" may contain the response data for the command or additional status information.

Differences in command calls between IE/ AS-i LINK and CPs / DP/AS-i LINK

There are significant differences in how a controller exchanges commands with an AS-i master.

- **IE/AS-i Link** (PROFINET) used the data record interface. The different commands are called with either "Write data record" ("WRREC" instruction) or "Read data record" ("RDREC" instruction) by various data record numbers.
- **S7-300 CPs** and **DP/AS-i Links** (PROFIBUS) use the command interface. All commands are called by data record number 2 with "Write data record" ("WRREC" instruction) plus "Read data record" ("RDREC" instruction). The type of command is defined by the data content in the write job.

Changes compared to the instruction "ASi_3422".

The instruction "ASI_CTRL" is a revised version of the instruction "ASi_3422 (Page 6529)" and provides improved functionality and compatibility. The specific changes are as follows:

- For writing and reading diagnostic data records, the instructions "WR_REC (Page 6496)" and "RD_REC (Page 6491)" have been replaced by the instructions "RDREC (Page 6459)" and "WRREC (Page 6461)". Their function is identical; however, they support data transfer via PROFINET IO.
- The block type of the instruction has been changed from a function (FC) to a function block (FB). "ASI_CTRL" has an instance data block and is multi-instance-capable.
- The designation of the formal parameters of "ASI_CTRL" complies with the SIMATIC system blocks. There is no STARTUP input parameter. The definition of the STATUS parameter is based on the instructions "RDREC (Page 6459)" and "WRREC (Page 6461)". The status identifiers for the parameter DONE and the new parameter BUSY have also been adjusted.

Operation of the "ASI_CTRL" instruction

The instruction "ASI_CTRL" is an asynchronous function block; in other words, its processing extends over multiple calls.

- A job is started with REQ = TRUE.
- The job status is displayed via the BUSY output parameters and the two central bytes of the output parameter STATUS.
- The BUSY parameter is set during job processing. Upon the first call, STATUS is assigned the value 00700100_H. Upon all subsequent calls for this job, it is assigned the value 00700200_H. When the job is complete, the result is output at the parameters DONE or ERROR.
  - If no errors have occurred, DONE is set. For jobs with response data from the AS-i master, this data is provided in the specified receive buffer. In such cases, the STATUS parameter also displays the volume of data supplied in bytes. STATUS is set to 00000000_H for jobs without response data.
  - If an error occurs during job processing, ERROR is set. The content of the receive buffer is invalid in such a case. An error code is entered in the STATUS parameter for a more detailed description of the error which has occurred.

Number of command calls

If you use the instruction "ASI_CTRL" to send commands, you may not simultaneously send other commands to the same AS-i master with "RDREC (Page 6459)" or "WRREC (Page 6461)". This also applies to multiple calls of the instruction for the same AS-i master.

The instruction "ASI_CTRL" cannot be run with interruptions. Calls therefore cannot be programmed in program priority classes which interrupt each other (for example with calls in OB 1 and in OB 35).

Parameters

The following table shows the parameters of the "ASI_CTRL" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = TRUE starts a new job unless a job is already in progress. No edge evaluation takes place.
LADDR	Input	WORD	I, Q, M, D, L or constant	Start address of the AS-i master in the S7 address space (logical base address). The start address is defined in the hardware configuration when the master is configured.
SD	Input	ANY	I, Q, M, D, L	Send buffer The parameter refers to a memory area in which the command is to be specified (see "ASi commands (Page 6536)"). Example: P#DB101.DBX 0.0 BYTE 223
RD	Input	ANY	I, Q, M, D, L	Receive buffer This buffer is only relevant for commands which deliver answer data. The parameter refers to a memory area in which a command response is stored (see "ASi commands (Page 6536)"). Example: P#DB102.DBX 224.0 BYTE 221
DONE	Output	BOOL	Q, M, D, L	DONE = TRUE: Job successfully completed.
BUSY	Output	BOOL	Q, M, D, L	BUSY = TRUE: Job in progress.
ERROR	Output	BOOL	Q, M, D, L	ERROR = TRUE: Job aborted with error.
STATUS	Output	DWORD	M, D	Job status / error code See the "STATUS parameter" description.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Note

Parameters LADDR, SD and RD

The parameters LADDR, SD and RD must not be changed in any block cycle when a job is in progress: they must remain constant.

Parameter STATUS

The following table shows the possible STATUS displays dependent upon DONE and ERROR.

DONE	ERROR	STATUS	Meaning
0	0	00700000_H	First call with REQ = FALSE; no active job.
0	0	00700100_H	First call with REQ = TRUE; job has started.
0	0	00700200_H	Subsequent call (REQ irrelevant); job is still in progress.
1	0	00000000_H	Job successfully completed. No response data.

DONE	ERROR	STATUS	Meaning
1	0	$0000xx00_H$	Job successfully completed. Number of xx bytes of response data.
0	1	$C0818400_H$	Data type of formal operand RD invalid.
0	1	$C0818500_H$	Error in communication with AS-i master (incorrect address configured at the LADDR parameter).
0	1	$C0838100_H$	Incorrect AS-i slave address.
0	1	$C0838200_H$	AS-i slave is not enabled (not in LAS).
0	1	$C0838300_H$	Error at the AS interface (the SD parameter may have been set too small).
0	1	$C0838400_H$	The command is not valid with the current AS-i master status.
0	1	$C0838500_H$	There is no AS-i slave with the address "0".
0	1	$C0838600_H$	The AS-i slave has invalid configuration data (I/O or ID codes).
0	1	$C083A100_H$	The AS-i slave addressed has not been found at the AS interface.
0	1	$C083A200_H$	There is no AS-i slave with the address "0".
0	1	$C083A300_H$	There is already an AS-i slave with the new address at the AS interface.
0	1	$C083A400_H$	The AS-i slave address cannot be deleted.
0	1	$C083A500_H$	The AS-i slave address cannot be set.
0	1	$C083A600_H$	The AS-i slave address cannot be permanently saved.
0	1	$C083A700_H$	Error during reading of the extended ID1 code.
0	1	$C083A800_H$	The target address is not plausible (e.g. a B slave address has been used for a standard slave).
0	1	$C083B100_H$	A length error has occurred during string transfer.
0	1	$C083B200_H$	A protocol error has occurred during string transfer.
0	1	$C083F800_H$	Unknown job number or job parameter.
0	1	$C083F900_H$	The AS-i master has detected an EEPROM error.

ASi commands

Description

The command interface allows the controller and AS-i master to exchange parameter assignment and information data.

These commands

- provide the complete functionality of the M4 master profile of the AS-i master specifications.
- enable the AS-i master to be completely configured from the controller.

Note

AS-i commands supported

Please see the manual of the relevant AS-i master for the AS-i commands supported and a detailed description.

General structure of the send buffer

The general structure of the send buffer for commands and job data is set out in the table below. The area for the command number must always be filled. The number of bytes for the job data can be found in the description of the command (see AS-i master documentation). "q" is the start address of the send buffer.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	Meaning							
q + 0	Command number							
q + 1	Job data							
q + 2	Job data							
q + ...	Job data							

General structure of the receive buffer

The general structure of the receive buffer for the command response data is set out in the table below. The number of bytes for the response data depends on the command. Some commands do not return response data, and therefore only require the definition of a virtual receive buffer which is not filled with data. "n" is the start address of the receive buffer.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	Meaning							
n + 0	Command number (echo)							
n + 1	Response data							
n + 2	Response data							
n + ...	Response data							

CAUTION

Memory areas can be overwritten

If the receive buffer of the "ASI_CTRL" instruction is too short, neighboring memory areas may be overwritten. The length specified in the ANY pointer of the RD parameter when the instruction "ASI_CTRL" is called is irrelevant. The required length for the receive buffer can be found in the description of the command.

The following applies for command numbers 39_H, 41_H, 42_H, 43_H and 44_H:

The receive buffer must be 221 bytes long (bytes 0 to 220), even if the command returns less data. Depending on the command, the highest bytes in the receive buffer may be overwritten with zero values by the AS-i master.

AS-i commands

A selection of possible AS-i commands is set out in the table below.

Name	Parameter	Return	Coding
Set_permanent_parameter (Set_Permanent_Parameter)	Slave address, parameter		00_H
Get_permanent_parameter (Get_Permanent_Parameter)	Slave address	Parameter	01_H
Write_parameter (Write_Parameter)	Slave address, parameter	Parameter echo	02_H
Read_parameter (Read_Parameter)	Slave address	Parameter value	03_H
Store_actual_parameters (Store_Actual_Parameters)			04_H
Set_configuration_data	Slave address, configuration		25_H
Get_configuration_data	Slave address	set configuration data	26_H
Store_actual_configuration (Store_Actual_Configuration)			07_H
Get_actual_configuration	Slave address	Actual configuration data	28_H
Configure_LPS	LPS		29_H
Set_offline_mode	Mode		$0A_H$
Select_auto-program	Mode		$0B_H$
Set_mode	Mode		$0C_H$
Change_AS–iSlave_address (Change_AS–iSlave_Address)	Address1, Address2		$0D_H$
Get_AS-iSlave_status	Slave address	Error record of the AS-i slave	$0F_H$
Read_lists_and_flags		LDS, LAS, LPS, flags	30_H
Get_overall_configuration		Actual configuration data, current parameters, LAS, flags	39_H
Set_overall_configuration	Overall configuration		$3A_H$
Write_parameter_list	Parameter list		$3C_H$
Read_parameter_echo_list		Parameter echo list	33_H
Write_CTT2_request	Slave address CTT2 string	CTT2 string	44_H
Read_version_identifier		Version string	14_H
Read_AS-i_slave	Slave address	ID code	17_H
Read_AS-i_slave_extended_ID1	Slave address	Extended ID1 code	37_H
Write_AS-iSlave_extended_ID1	Extended ID1 code		$3F_H$
Read_AS-iSlave_extended_ID2	Slave address	Extended ID2 code	38_H
Read_AS-iSlave_IO	Slave address	I/O configuration	18_H
Read_I/O_error_list		LPF	$3E_H$

Name	Parameter	Return	Coding
Write_AS–i-slave_parameter_string	Slave address, parameter string		40 $_H$
Read_AS–iSlave_parameter_string	Slave address	Parameter string	41 $_H$
Read_AS-iSlave_ID_string	Slave address	ID string	42 $_H$
Read_AS-iSlave_diagnostic_string	Slave address	Diagnostic string	43 $_H$
Read_AS-i_line_error_counter			4A $_H$
Read_and_clear_AS-i_line_error_counter			4B $_H$
Read_AS-iSlave_error_counter	Slave address		4C $_H$
Read_and_clear_AS-iSlave_error_counter	Slave address		4D $_H$
Additional command for DP/ AS-i F-Link:			
AS–i_status/diag_of_F_slaves		Status/diagnostics of all AS-i safe slaves	51 $_H$

Note

Re-initializing the AS-i master command interface

Another command which is not mentioned in the table is command 77 $_H$. This call re-initializes the command interface of the AS-i master. Any command which the AS-i master specified is currently processing will be terminated.

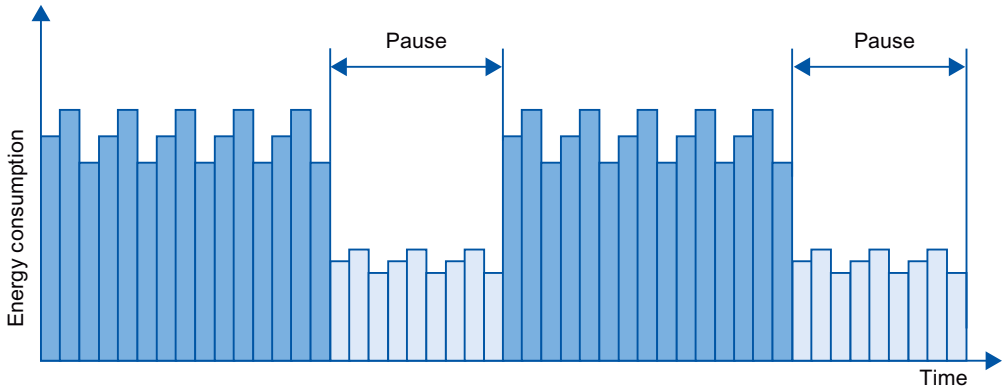
As of version V2.1.20 of DP/AS-i LINK Advanced, command 0E $_H$ is also available. This call enables you to release and block the ground fault monitoring function for a line.

PROFIenergy

Description of PROFIenergy

PROFIenergy

PROFIenergy is a manufacturer- and device-neutral profile for energy management with PROFINET. PROFIenergy enables central, coordinated device shutdown to reduce electricity consumption during breaks in production and unplanned interruptions.



The PROFINET devices/power modules are switched off via special commands in the user program of the PROFINET IO controller. You do not need additional hardware. The PROFIenergy commands are interpreted directly by the PROFINET devices.

PROFIenergy controller (PE controller)

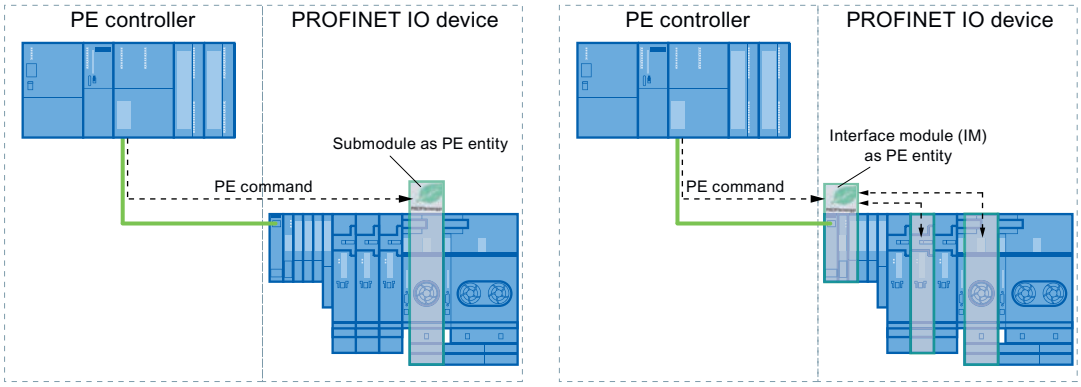
The PE controller is a PLC that activates or deactivates the idle state of slave devices. Specific production components, or complete production lines are deactivated and reactivated via the user program. Commands (e.g. "Start_Pause" and "End_Pause") are sent to the lower-level device using corresponding instructions (function blocks). The commands are sent via the PROFINET communication protocol.

PROFIenergy entity (PE entity)

The PE entity receives the PROFIenergy commands of the PE controller and converts these accordingly (e.g. by returning a measured value, or by activating the energy-saving mode). Integration of the PE entity in a device that supports PROFIenergy is based on specific device and manufacturer data.

The PE entity can be implemented, for example:

- Within the proxy of a submodule: The PE commands are valid for the addressed submodule and any existing lower-level submodules.
- Within the proxy of a module: The PE commands are valid for different submodules within the module.



- For networked submodules without a proxy function: The PE commands in this case are only valid for the respective submodule.

PROFIenergy instructions

- Instructions for IO controller
  - The "PE_START_END (Page 6542)" instruction represents the simplest way to activate or deactivate the idle state of the PROFINET devices (PROFIenergy commands "Start_Pause" and "End_Pause"). This is done using a positive and negative signal edge in the instruction.
  - The "PE_CMD (Page 6546)" instruction allows you to transmit all PROFIenergy commands, including "Start_Pause" and "End_Pause". With other commands, you can query the current status of the PROFINET devices or the behavior during breaks, for example.
  - The instruction "PE_DS3_Write_ET200S (Page 6551)" is used to define the switching characteristic of up to 8 slots of ET 200S. The instruction is not a PROFIenergy instruction; however, it supplements the PROFIenergy functions for an ET 200S.
- Instruction for iDevices
  The "PE_I_DEV (Page 6574)" instruction allows you to implement PROFIenergy on iDevices as well. The instruction receives PROFIenergy commands on the iDevice and forwards these to the user program for processing. After processing the command, the user program calls the "PE_I_DEV (Page 6574)" instruction again in order to send the acknowledgement to the IO controller. For these replies, each command offers you a corresponding auxiliary block that supplies the response data to the instruction "PE_I_DEV (Page 6574)".

PROFIenergy commands (PE commands)

The PE controller transmits PE commands to the PE entity. The PE command can be either a control command to switch a PE entity to the energy-saving mode, or a command to read a status or measured value:

- PI commands for control
  PROFIenergy supports two control commands that can be executed using either the "PE_Start_End (Page 6542)" instruction or the "PE_CMD (Page 6546)" instruction:
  - Start_Pause: Starting a suitable energy-saving mode (PE Energy-saving mode)
  - End_Pause: Exiting the energy-saving mode (switch to PE_ready_to_operate mode)
- PI commands for reading a status or measured value
  Certain condition information can be read by the control system using the following status commands with the instruction "PE_CMD (Page 6546)":
  - PE_Identify: Reading the list of PE commands supported by the PE entity.
  - PEM_Status: Reading the currently active mode of a PE entity (e.g. PE_ready_to_operate).
  - Query_Modes: Output an overview of all supported energy-saving modes, including the time and energy information
  - Query_Measurement: Output the measured values of a PE entity

Example applications

Examples for use of PROFIenergy instructions can be found in the Industry Online Support in the entry "PROFIenergy - Saving Energy with SIMATIC S7 (http://support.automation.siemens.com/WW/view/en/41986454)".

See also

Service and Support (http://support.automation.siemens.com/)

IO controller

PE_START_END: Start and exit energy-saving mode

Description

The instruction "PE_START_END" is used to start and exit the energy-saving mode of a specified PE entity (e.g. ET 200S).

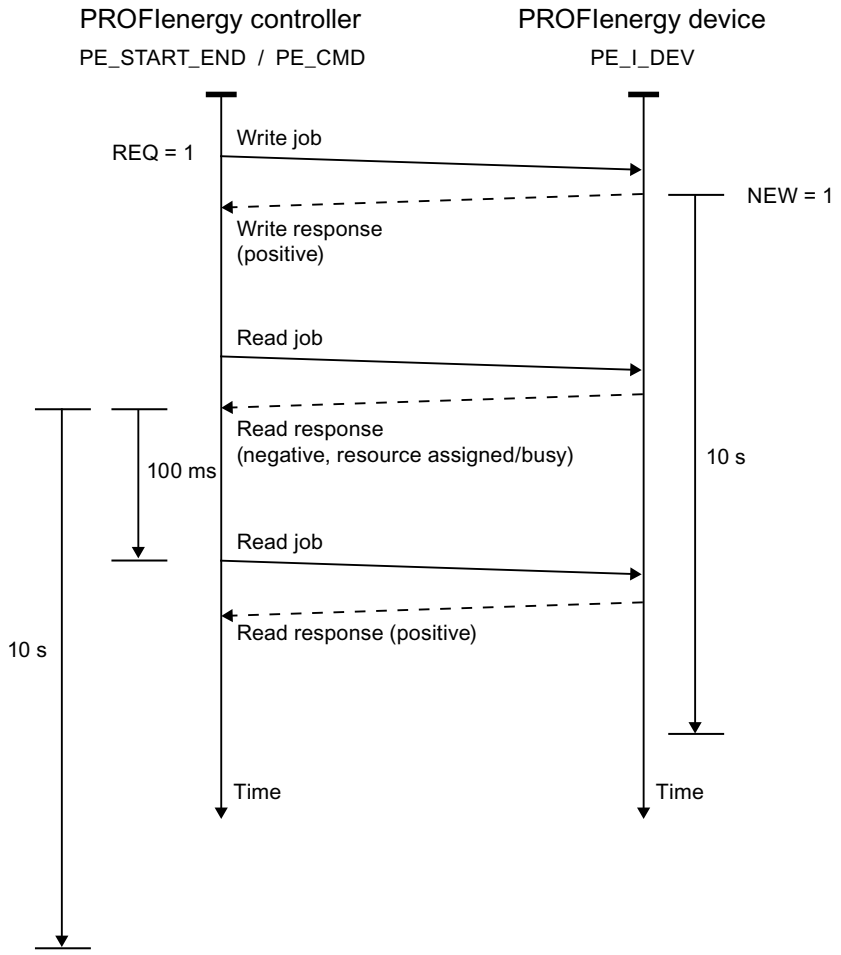
The instruction "PE_START_END" is used in the PE controller, preferably when only field devices from which energy data does not need to be read out are connected to the corresponding PE devices. Alternatively, the instruction "PE_CMD (Page 6546)" can be used to read energy data.

The energy-saving modes are configured in the user program of the PE controller. On completion of the "PE_START_END" instruction, the PE entity reports its current energy-saving mode and outputs this data at parameter PE_MODE_ID.

Write and read jobs of the "PE_START_END" instruction.

The instruction "PE_START_END" uses "WRREC (Page 6461)" internally to transmit a PROFIenergy command as write job to the PE entity. "PE_START_END" then waits for the acknowledgment frame from the PE entity. The acknowledgement data record is read with the instruction "RDREC (Page 6459)" every 100 milliseconds. Until acknowledgment is received from the PE entity, the function repeats the read job for 10 seconds at intervals of 100 milliseconds. The response data returned by the PE entity is also read with the instruction "RDREC (Page 6459)".

Below is a flowchart of the write and read jobs:



Parameters

The following table shows the parameters of the "PE_START_END" instruction:

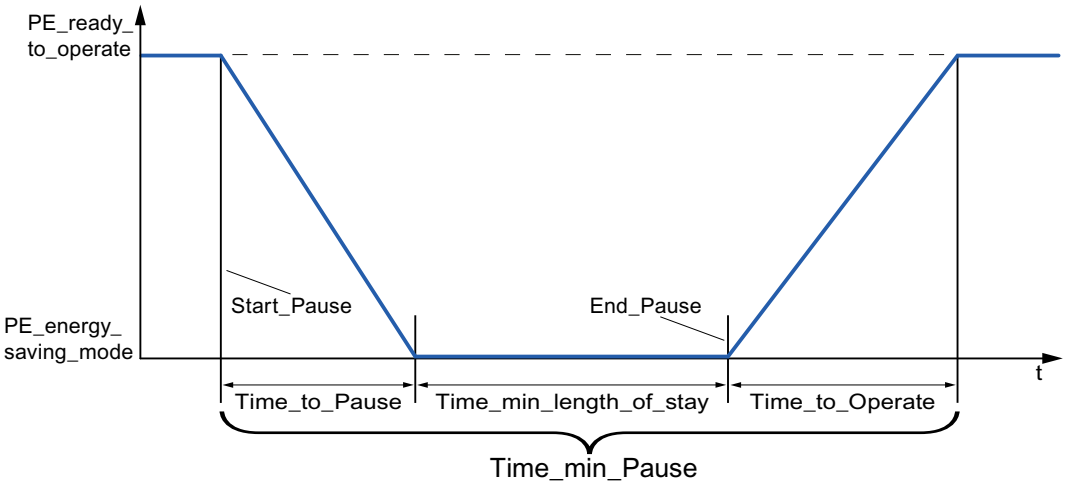
Parameter	Declaration	Data type	Memory area	Description
START	Input	BOOL	I, Q, M, D, L	Transmitting the PE command "Start_Pause" to the PE entity with the address set at parameter ID.
END	Input	BOOL	I, Q, M, D, L	Transmitting the PE command "End_Pause" to the PE entity with the address set at parameter ID.
ID	Input	DWORD	I, Q, M, D, L or constant	Address of the PE entity (e.g. ET 200S). The address may be taken from the hardware configuration.
PAUSE_TIME	Input	TIME	I, Q, M, D, L or constant	Planned pause duration. • Range: T#1MS to T#24D20H31M23S647MS • Start value: T#0MS
VALID	Output	BOOL	I, Q, M, D, L	PE command successfully sent.
BUSY	Output	BOOL	I, Q, M, D, L	PE command still being processed.
ERROR	Output	BOOL	I, Q, M, D, L	Error occurred during processing. The error message is output at the STATUS parameter.
STATUS	Output	DWORD	I, Q, M, D, L	Block status / error number (see "STATUS parameter")
PE_MODE_ID	Output	BYTE	I, Q, M, D, L	Identification number of energy-saving mode (energy-saving level for the duration of the pause).

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

PAUSE_TIME parameter

The parameter PAUSE_TIME is used to set the default duration of the energy-saving period at the PE entity. The PE entity checks whether the pause period is of sufficient length and whether it can be implemented. The minimum pause duration (Time_min_Pause) must be

greater than the sum of the times the device needs to switch to energy-saving mode (Time_to_Pause) and to switch back to operating mode (Time_to_Operate).



ET 200S checks whether the scheduled pause duration is greater than or equal to the minimum pause duration (PM-E_Pause_Min) saved on the ET 200S. This is fixed at 10 seconds. If a shorter pause is used, the power modules (PM-E) of the ET 200S will remain on.

The module does not switch back on automatically at the end of the pause; it remains in OFF mode until the "END" command is sent. This prevents uncoordinated switch-on, which can cause undesired peak loads.

Parameter STATUS

Error information is output at the STATUS output parameter. If it is interpreted as ARRAY[1...4] of BYTE, the error information has the following structure:

Array element	Name	Meaning
STATUS[1]	Function_Num	Cause of the error • B#16#**00**: No error • B#16#**DE**: Error during reading of data record • B#16#**DF**: Error during writing of data record • B#16#**C0**: Error message from the instruction or from the communication instructions "RDREC (Page 6459)" and "WRREC (Page 6461)" used internally.
STATUS[2]	Error_Decode	Location of the error ID • **80**: DPV1 error as defined in IEC 61158-6 or application-specific • **FE**: DP/PNIO profiles - PROFIenergy-specific error

Array element	Name	Meaning
STATUS[3]	Error_Code_1	Error ID • When Error_Decode = **80**: – **80**: Simultaneous rising edge at the input parameters START and END. – **81**: Length conflict at the CMD_PARAM and CMD_PARAM_LEN parameters. – **82-8F**: Other error messages (reserved) • When Error_Decode = **FE**: – **01**: "Service Request ID" invalid – **02**: Incorrect "Request_Reference" – **03**: "Modifier" invalid – **04**: "Data Structure Identifier RQ" invalid – **05**: "Data Structure Identifier RS" invalid – **06**: "PE energy-saving modes" are not supported – **07**: "Response" is too long (maximum transmissible length exceeded) – **08**: "Count" invalid – **50**: No suitable "energy mode" is available. – **51**: Time value specified is not supported. – **52**: "PE_Mode_ID" invalid
STATUS[4]	Error_Code_2	Manufacturer-specific error ID extension

Note

Error messages of the instructions RDREC and WRREC

The instruction "PE_START_END" uses the instructions "WRREC (Page 6461)" and "RDREC (Page 6459)" for communication. Error messages for these instructions are output in the field elements STATUS[1] to STATUS[4].

Please see the description of the corresponding STATUS (Page 6471) parameter for an explanation of the error codes of the "WRREC (Page 6461)" and "RDREC (Page 6459)" instructions.

PE_CMD: Start and exit energy-saving mode / Read out status information

Description

The instruction "PE_CMD" is used in the PE controller to start or terminate a pause in the energy-saving mode in the PE entity. "PE_CMD" can be used to read additional information and energy measurement values from a PE entity.

The instruction is best used with PE controllers to which assigned PE devices are connected from which energy measurements are to be read out. If this is not the case, the instruction "PE_START_END (Page 6542)" can also be used to start and end the pauses.

Transfer of the PROFIenergy commands (PE commands)

The instruction "PE_CMD" transmits a PROFIenergy command to a PE entity.

The instruction can also be used if additional commands are to be added to the PROFIenergy profile in future. The commands which can be used with the current PROFIenergy profile are listed in the description of the CMD and CMD_MODIFIER parameters (see the "CMD and CMD_MODIFIER parameters" table).

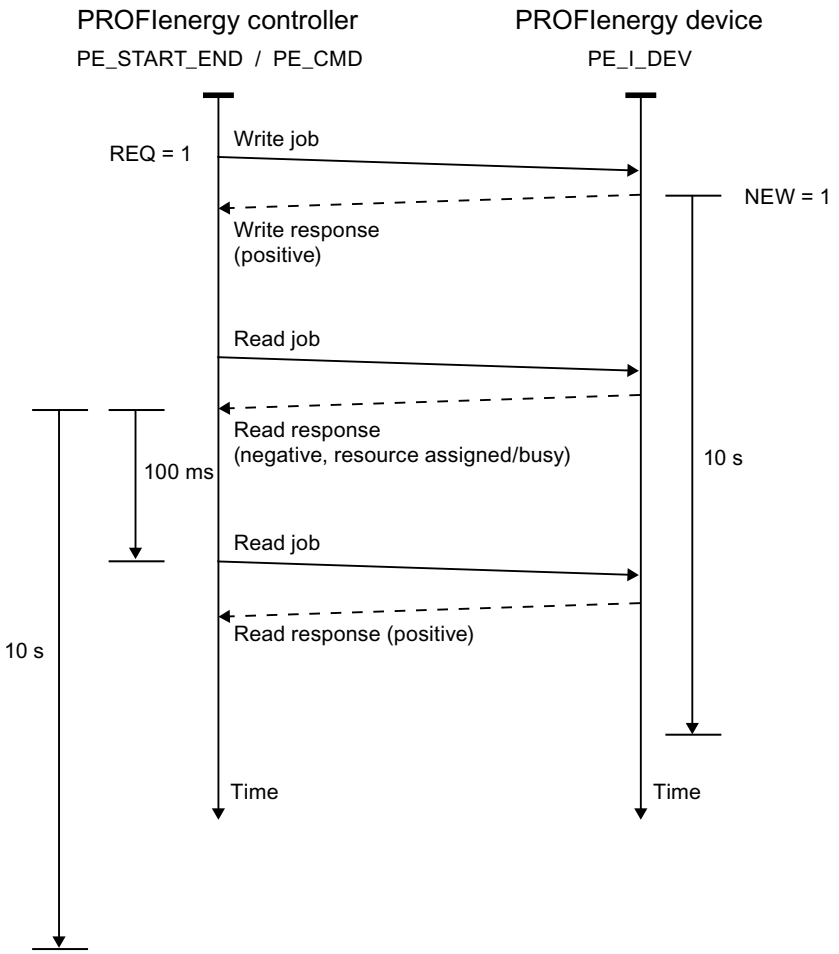
- The individual PE commands which are transferred to the PE entity using the instruction are assigned defined "Service_Request_IDs". The Service_Request_IDs 01 to 05 and 16 are assigned at the CMD parameter.
- The two PE commands 04 (Query_Modes) and 16 (Query_Measurement) are defined in more detail with the CMD_MODIFIER parameter.
- Additional values are transferred at the CMD_PARA parameter for individual PE commands (see the description of the individual PE commands). The parameter CMD_PARA_LEN defines the length of the data at the parameter CMD_PARA.

Commands are transmitted without a plausibility check. The response data of the PE entity is saved to the RESPONSE_DATA data area that is addressed by ANY pointer (for information about the response frame contents, refer to the description of the respective PE command).

Write and read jobs of the "PE_CMD" instruction.

The instruction "PE_CMD" uses "WRREC (Page 6461)" internally to transmit a PROFIenergy command as a write job to the PE entity. "PE_CMD" then waits for the acknowledgment from the PE entity. The acknowledgement data record is read with the instruction "RDREC (Page 6459)" every 100 milliseconds. Until acknowledgment is received from the PE entity, the function repeats the read job for 10 seconds at intervals of 100 milliseconds. The response data returned by the PE entity is also read with the instruction "RDREC (Page 6459)".

Below is a flowchart of the write and read jobs:



Parameters

The following table shows the parameters of the "PE_CMD" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Starts transferring the PE command upon a rising edge.
ID	Input	DWORD	I, Q, M, D, L or constant	Address of the PE entity For PROFINET IO devices, you can transfer the address from the hardware configuration.
CMD	Input	BYTE	I, Q, M, D, L, P or constant	Service-Request-ID of the PROFIenergy command in accordance with the PROFIenergy profile (see "CMD and CMD_MODIFIER parameters"). Further service request IDs are possible following extensions of the PROFIenergy profile.

Parameter	Declaration	Data type	Memory area	Description
CMD_MODIFIER	Input	BYTE	I, Q, M, D, L, P or constant	PROFIenergy sub-command (only when CMD=3 or CMD=16, see "CMD and CMD_MODIFIER parameters") Further sub-commands are possible following extensions of the PROFIenergy profile.
CMD_PARA	Input	ANY	I, Q, M, D, L	Parameters for the PE commands: • Get mode: PE_mode_ID • Get measurement values: List of Measurement_Ids The complete Service Data Request is entered.
CMD_PARA_LEN	Input	INT	I, Q, M, D, L, P or constant	The actual length of the command parameters (<= length defined in CMD_PARA; is verified by the instruction).
RESPONSE_DATA	InOut	ANY	I, Q, M, D, L	PROFIenergy information May be complete response frame including block header, depending on the command. Note: If the buffer is too small, only the number of bytes which is specified in the ANY pointer will be entered.
VALID	Output	BOOL	I, Q, M, D, L	Command successfully sent.
BUSY	Output	BOOL	I, Q, M, D, L	Command still being processed.
ERROR	Output	BOOL	I, Q, M, D, L	Error occurred during processing.
STATUS	Output	DWORD	I, Q, M, D, L	Block status / error number (see "STATUS parameter"):

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameters CMD and CMD_MODIFIER

CMD	CMD_ MODIFIER	PROFIenergy command	Description
01	0	Start_Pause (Page 6564)	Start energy-saving mode or switch to a different energy-saving mode.
02	0	End_Pause (Page 6565)	End energy-saving mode.
03	1	Query_Modes - List of energy-saving modes (Page 6565)	Output the energy-saving modes supported.
	2	Query_Modes - Get mode (Page 6566)	Output attributes of the energy-saving mode currently enabled.
04	0	PEM_Status (Page 6568)	Query status of energy-saving mode.
05	0	PE_Identify (Page 6569)	Read out the number and description of PE commands supported.

CMD	CMD_ MODIFIER	PROFIenergy command	Description
16	1	Query_Measurement - Get_Measurement_List (Page 6570)	List of measured values supported by the PE entity.
	2	Query_Measurement - Get_Measurement_Values (Page 6572)	Output the measured values of the PE entity

Parameter STATUS

Error information is output at the STATUS output parameter. If it is interpreted as ARRAY[1...4] of BYTE, the error information has the following structure:

Array element	Name	Meaning
STATUS[1]	Function_Num	Cause of the error • B#16#00: No error • B#16#DE: Error during reading of data record • B#16#DF: Error during writing of data record • B#16#C0: Error message from the internal communication instructions "RDREC (Page 6459)" and "WRREC (Page 6461)".
STATUS[2]	Error_Decode	Location of the error ID • 80: DPV1 error as defined in IEC 61158-6 or application-specific • FE: DP/PNIO profiles - PROFIenergy-specific error
STATUS[3]	Error_Code_1	Error ID • When Error_Decode = 80: – 81: Length conflict at the CMD_PARA and CMD_PARA_LEN parameters, or maximum data record length (4095 bytes) exceeded. – 82-8F: Other error messages (reserved) • When Error_Decode = FE: – 01: "Service Request ID" invalid – 02: Incorrect "Request_Reference" – 03: "Modifier" invalid – 04: "Data Structure Identifier RQ" invalid – 05: "Data Structure Identifier RS" invalid – 06: "PE energy-saving modes" are not supported – 07: "Response" is too long (maximum transmissible length exceeded) – 08: "Count" invalid – 50: No suitable energy-saving mode (energy mode) is available. – 51: Time value specified is not supported. – 52: "PE_Mode_ID" invalid
STATUS[4]	Error_Code_2	Manufacturer-specific error ID extension

Note

Error messages of the instructions RDREC and WRREC

The "PE_CMD" instructions use the "WRREC (Page 6461)" and "RDREC (Page 6459)" instructions for communication. Error messages for these instructions are output in the field elements STATUS[1] to STATUS[4].

Please see the description of the corresponding STATUS (Page 6471) parameter for an explanation of the error codes of the "WRREC (Page 6461)" and "RDREC (Page 6459)" instructions.

PE_DS3_Write_ET200S: Set power module switching behavior

Description

The instruction "PE_DS3_Write_ET200S" sends basic settings for power module switching behavior to the ET 200S. The switching behavior of up to 8 slots in the ET 200S (e.g. for power modules) can be defined with the instruction "PE_DS3_Write_ET200S".

Note

This instruction is not part of the PROIenergy profile but instead supplements SIMATIC-specific functions.

Parameters

The following table shows the parameters of the "PE_DS3_Write_ET200S" instruction:

Parameter	Declaration	Data type	Memory area	Description
ENABLE	Input	BOOL	I, Q, M, D, L	A positive edge triggers the transfer of the data record. The data record must be transferred again after voltage OFF/ON.
ID	Input	DWORD	I, Q, M, D, L or constant	Address of the ET 200S The address may be taken from the hardware configuration.
SLOT_NO_X	Input	INT	I, Q, M, D, L, P or constant	Slot number of switchable power module X.

Parameter	Declaration	Data type	Memory area	Description
FUNC_X	Input	INT	I, Q, M, D, L, P or constant	Function of the module in this slot. Use the parameter FUNC_X to define the switching behavior of the PM-E (power module of the ET 200S): • FALSE: – With "PAUSE_START": -No effect on PM-E -PM-E remains on – With "PAUSE_STOP": -Switches PM_E back on • TRUE: – With "PAUSE_START": -Switches PM_E off – With "PAUSE_STOP": -Switches PM-E back on
BUSY	Output	BOOL	I, Q, M, D, L	Transfer not yet complete.
DONE	Output	BOOL	I, Q, M, D, L	Transfer completed without errors.
ERROR	Output	BOOL	I, Q, M, D, L	Transfer completed with error.
STATUS	Output	DWORD	I, Q, M, D, L, P	Error number (see STATUS parameter of the instruction "PE_Start_End (Page 6542)")

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

PE_WOL: Starting and stopping energy-saving mode via WakeOnLan

Description PE_WOL

Description

The "PE_WOL" instruction sends the PROFIenergy commands "Start_Pause" and "End_Pause" to multiple PROFIenergy-enabled devices in the PROFINET I/O systems.

Multiple PE devices can be coordinated via the instruction provided that the PE devices support the "Wake on LAN" function over a UDP connection.

The "PE_WOL" instruction can only be executed on a CPU with and integrated Ethernet interface. This CPU should be able to load blocks with a size of about 40 KB. You cannot use the block with PROFINET I/O systems that are connected via an Ethernet CP.

The instruction "PE_WOL" is executed asynchronously.

Definition: Wake on LAN

The Wake on LAN function allows data processing equipment to resume working from a state near total shutdown when a special Ethernet packet is received.

For this procedure to work, the data processing equipment must have a network controller that is capable of receiving such a packet.

This packet (Magic Packet™) has a special format. It contains the MAC address of the network adapter 15 times.

Selection of devices

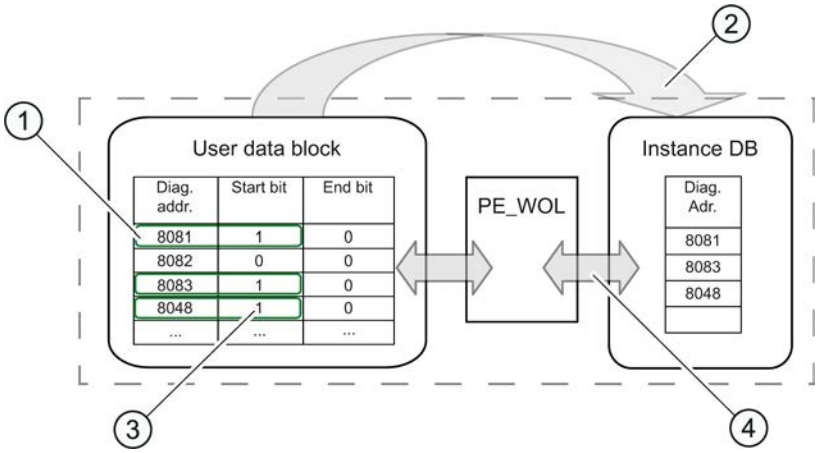
The devices are selected using the user data block at the parameter PENERGY. The user DB represents the database for processing multiple devices.

Prior to initialization of "PE_WOL", you must store at least the following information in the user DB:

- ID of the PROFINET I/O system
- Data of the connection which is used for "Wake On LAN".
- Port number which is used for "Wake On LAN".
- For each device
  - Pause time (PauseTime)
  - Switching of the device to the PE_SLEEP_MODE (EnableSleep)

You initialize the "PE_WOL" instruction with the COM_RST parameter. The jobs stored in the user DB are processed one after the other after initialization.

The graphic below illustrates how the PE command "Start_Pause" is transmitted to several devices:



(1) Step 1: The bit "CmdStartPause" of the devices to be shut down is set to "1" by the user.

(2) Step 2: The diagnostic addresses of the devices to be shut down (CmdStartPause = "1") are linked to the queue.

(3) Step 3: The bit "CmdStartPause" is reset automatically once the jobs have been linked.

(4) Step 4: The instruction "PE_WOL" processes the jobs as soon as they are linked.

A PROFIenergy command "CmdStartPause" or "CmdEndPause" can be sent for all recognized devices in the PROFINET IO system via the START and END parameters.

The status of the job processing as well as possible errors during processing are output by the STATUS parameter.

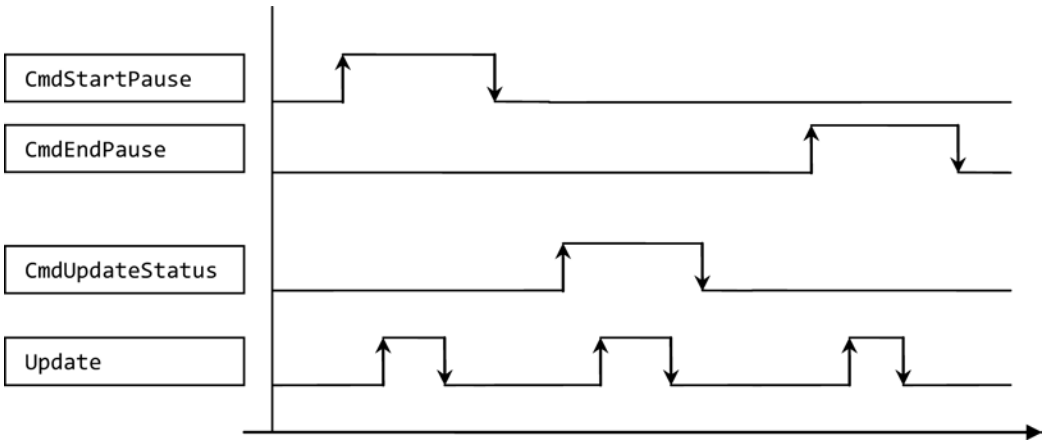
Operation of the instruction using the user DB

The operation of the instruction "PE_WOL" can only take place via the user DB. A basic procedure applies in this case:

1. Selection of the command to be executed for a device:
   - START_PAUSE ("CmdStartPause" in user DB)
   - ENDE_PAUSE ("CmdEndPause" in user DB)
   - UPDATE_STATUS ("CmdUpdateStatus" in user DB)
2. Setting the bit for updating ("Update" in the header of the user DB)
   At least one CPU cycle should pass with an "Update" = False between two updates; otherwise, edge detection cannot be guaranteed.

Prioritizing the PE commands

The graphic below shows the chronological sequence of three possible commands.



These are processed one after the other, regardless of whether the previous command call was successful or completed with an error.

Only one command is executed if you set two commands, such as "CmdEndPause" and "CmdUpdateStatus" simultaneously. A prioritization is present within the block:

- The command "CmdStartPause" has the highest priority and is therefore always executed if selected.
- The command "CmdEndPause" has the second highest priority.
- The command "CmdUpdateStatus" has the lowest priority.

If all three commands are set at the same time, the commands that are not executed remain preselected. In this case, the next rising edge executes the next command.

Parameters

The following table shows the parameters of the "PE_WOL" instruction:

Parameter	Declaration	Data type	Memory area	Description
COM_RST (Page 6555)	Input	BOOL	I, Q, M, D, L	Resets the block and performs a re-initialization. As long as True is set here, the initialization is started but not yet fully completed. Only the falling edge continues the initialization and switches to the normal operating mode after initialization.
START (Page 6557)	Input	BOOL	I, Q, M, D, L	A rising edge performs a "CmdStartPause" PROFIenergy command for all detected devices that support this function.
END (Page 6558)	Input	BOOL	I, Q, M, D, L	A rising edge performs a "CmdEndPause" PROFIenergy command for all detected devices that support this function.
PENERGY (Page 6558)	InOut	ANY	D	Pointer to the user DB that contains the database for processing multiple devices.
STATUS (Page 6562)	Output	DWORD	I, Q, M, D, L	Status/error number for the current status of the instruction (see "STATUS parameter").

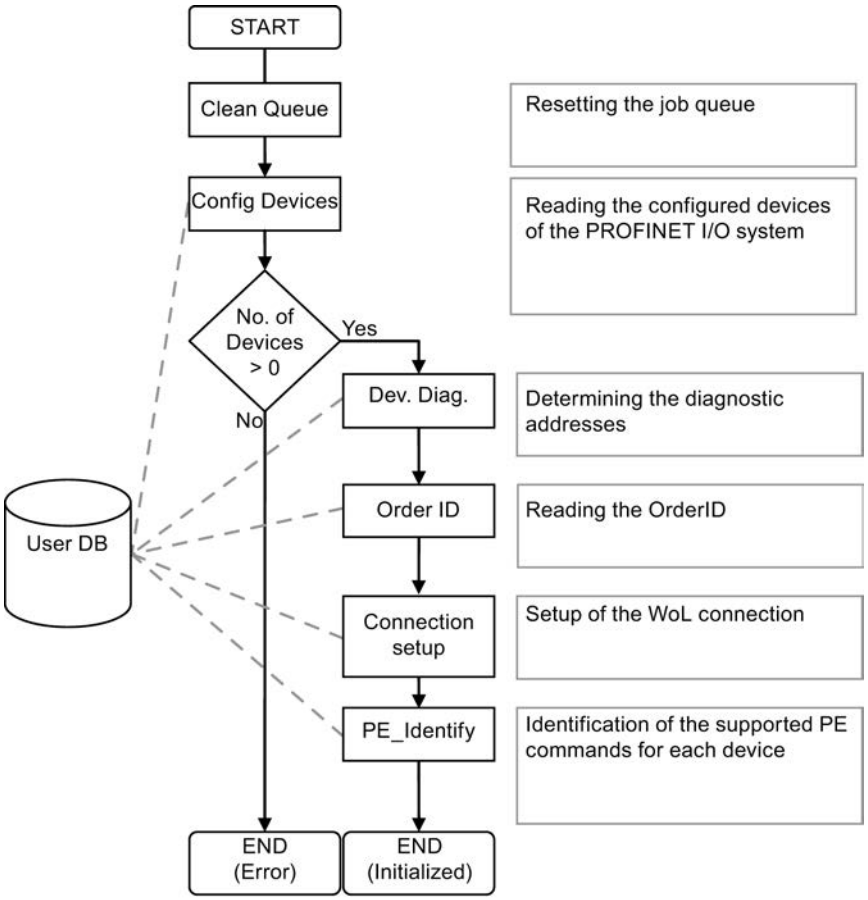
For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter COM_RST

Sequence of the initialization routine

You start the initialization of the "PE_WOL" instruction with the COM_RST parameter.

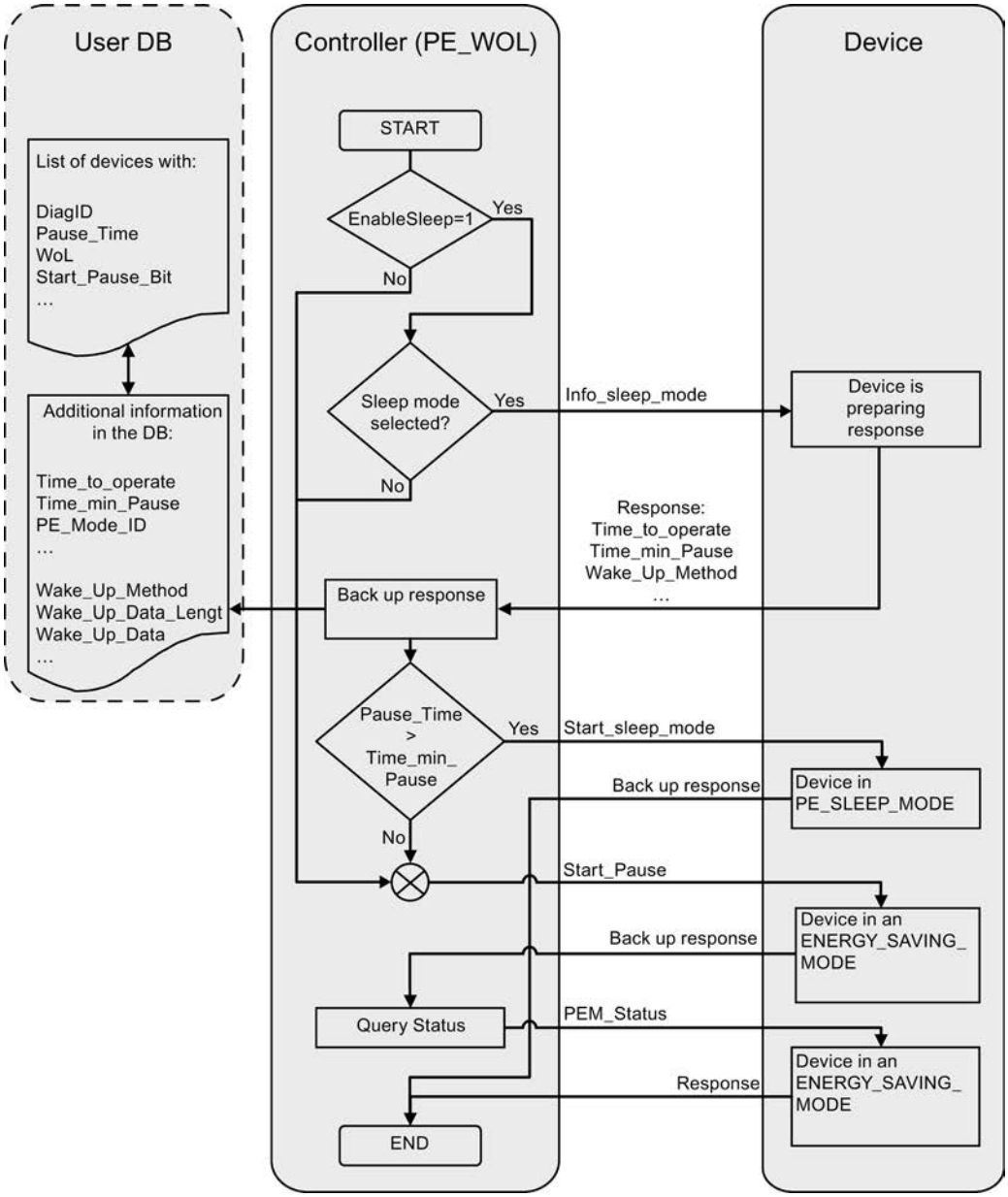
The following flow diagram shows the initialization routine.


START
Clean Queue
Resetting the job queue
Config Devices
Reading the configured devices of the PROFINET I/O system
No. of Devices > 0
Yes
No
Dev. Diag.
Determining the diagnostic addresses
Order ID
Reading the OrderID
User DB
Connection setup
Setup of the WoL connection
PE_Identify
Identification of the supported PE commands for each device
END (Error)
END (Initialized)

Parameter START

Sequence of the CmdStartPause command

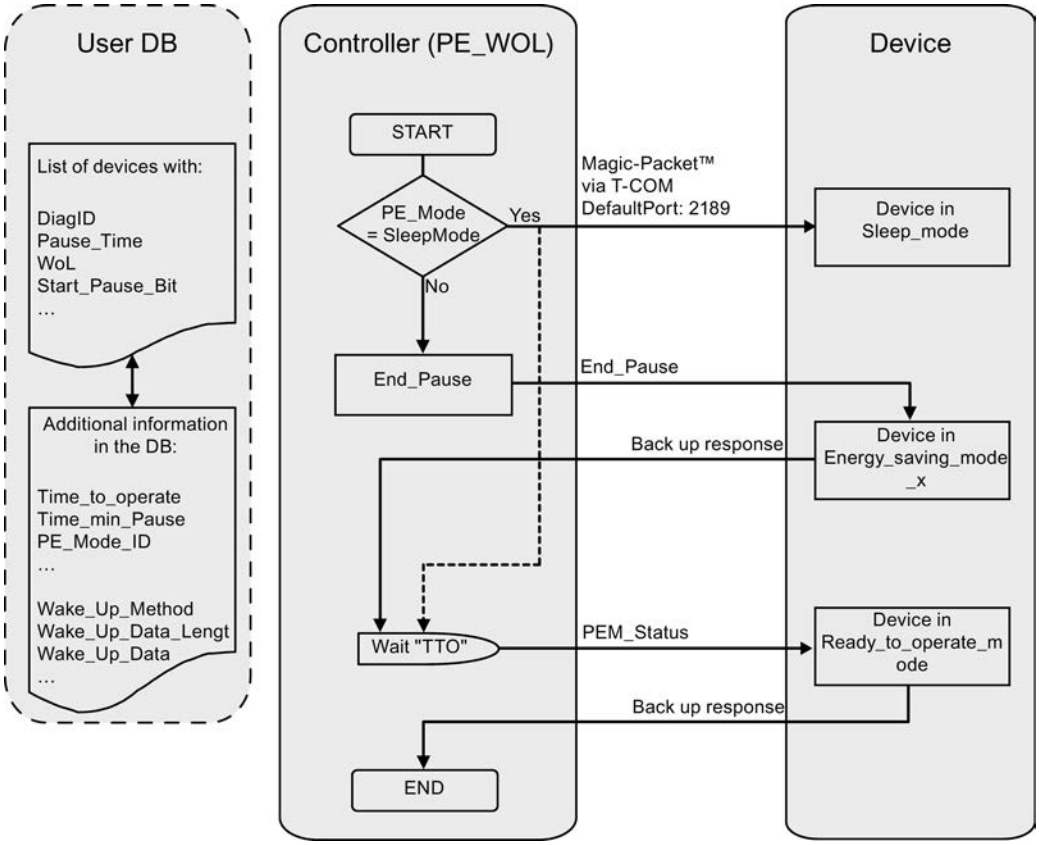
The flow diagram shows the internally used functions and interaction with a device when executing the CmdStartPause command.



Parameter END

Sequence of the CmdEndPause command

The flow diagram shows the internally used functions and interaction with a device when executing the CmdEndPause command.



PENERGY parameter

Data block at the PENERGY parameter

The user DB for the "PE_WOL" PROFIenergy instruction represents a data base for processing multiple devices.

The data block is usually divided into two parts. These are:

- Header 110 bytes
- Device section for a maximum of 256 devices each with 100 bytes (Device). These include:
  - Device-specific data (Device)
  - PROFIenergy-specific data (PE)
  - Data for the job processing (Task)
  - User data (UserData)

The individual sections have a certain size, which is constant. Overall, these result in a constant size of 25,746 bytes for the data block.

Connection parameter "Connection"

The "PE_WOL" instruction reserves a connection resource in the area of "Open User Communication". This is used as a UDP connection. The following parameters must be defined in the data block for this:

- Connection ID ("Connection.id" parameter)
  The connection ID is an integer within the range of 1 to 32. The default setting is 31. It is used to identify the communication resources allocated by the firmware, for example the send and receive buffers.
  The connection ID must be unique throughout the CPU.
- Port number that is used for the "Wake on LAN" function ("Header.PortNo" parameter)
  Number of the UDP port via which a "Wake On LAN" packet is sent. These port numbers are part of the communication resources, which are identified and provided by the firmware through the connection ID. The default setting for port 2189 used here is currently not assigned by IANA. The port number is transferred to the connection configuration. This affects the "Connection.local_tsap_id[1]" and "Connection.rem_tsap_id[1]" parameters.
- Interface ID ("Connection.local_device_id" parameter)
  The interface ID is also part of the connection description. This ID identifies the CPU interface to use for this connection. There are several possible valid values.
  However, these need to be adapted to the employed CPU and interface:
  - B#16#01 for ET200S CPU or WinAC RTX with Ethernet interface in subslot IF1
  - B#16#02 for CPU 315(F)-2PN/DP or CPU 317(F)-2PN/DP
  - B#16#03 for CPU 319(F)-2PN/DP
  - B#16#05 for CPU 41x(F)-3PN/DP
  - B#16#06 for WinAC RTX with Ethernet interface in subslot IF2
  - B#16#0B for WinAC RTX with Ethernet interface in subslot IF3
  - B#16#0F for WinAC RTX with Ethernet interface in subslot IF4

Structure of the data block

The data block has the following structure:

Name	Data type	Offset	Comment
Header	STRUCT	-	Header information
Update (1)	BOOL	0.0	Signal for indicating a change in the data area. • True = indicates a change by the user. • False = indicates the application of changes.
Initialized	BOOL	0.1	Signal indicating the completed initialization. • True = initialization completed. • False = indicates that the block is not initialized.
LinkUp	BOOL	0.2	Indicates successful configuration of the Ethernet interface. • True = interface ready-to-use. • False = interface not yet configured.
LinkDown	BOOL	0.3	Indicates an unconfigured interface. • True = interface is not configured. • False = interface is currently being configured or is now configured.
PROFINET_ID (1)	INT	2.0	ID of the PROFINET I/O system
Reserved	ARRAY [1..37] OF BYTE	4.0	Reserved
LastDeviceID	INT	42.0	Contains the highest device ID in the PROFINET I/O system.
PortNo (1)	INT	44.0	Port number that is used for the "Wake on LAN" function (default= 2189).
Connection	STRUCT	-	Contains the connection configuration of the "Wake on LAN" connection.
block_length	WORD	46.0	Length of the structure (always B#16#40).
id (1)	WORD	48.0	Connection ID
connection_type (1)	BYTE	50.0	Connection type (UDP = B#16#13)
active_est (1)	BOOL	51.0	Active connection establishment (always passive for UDP)
local_device_id (1)	BYTE	52.0	Contains the interface ID (CPU dependent)
local_tsap_id_len (1)	BYTE	53.0	Contains the length in bytes of the own/local UDP port
rem_subnet_id_len (1)	BYTE	54.0	Unused (always B#16#00)
rem_staddr_len (1)	BYTE	55.0	Contains the length of the remote IP address or B#16#00.
rem_tsap_id_len (1)	BYTE	56.0	Contains the length in bytes of the remote UDP port.
next_staddr_len (1)	BYTE	57.0	Contains the length of the default router address (not relevant).
local_tsap_id (1)	ARRAY[1..8] OF INT	58.0	Contains the own/local port number.
rem_subnet_id (1)	ARRAY[1..6] OF BYTE	74.0	Unused (always B#16#00)
rem_staddr (1)	ARRAY[1..6] OF BYTE	80.0	Contains the remote IP address.
rem_tsap_id (1)	ARRAY[1..8] OF INT	86.0	Contains the remote UDP port number.

Name	Data type	Offset	Comment
next_staddr (1)	ARRAY[1..6] OF BYTE	102.0	irrelevant
spare (1)	WORD	108.0	
Device	ARRAY[1..256] OF STRUCT	110.0	Array of devices
Device	STRUCT	110.0	Contains data for each device
DiagID	WORD	110.0	Diagnostic address of the device. Assigned by the hardware configuration.
InterfaceID	WORD	112.0	Diagnostic address of the device interface. Assigned by the hardware configuration.
MACAdr	ARRAY[1..6] OF BYTE	114.0	Contains the MAC address of the device.
IPAdr	ARRAY[1..4] OF BYTE	120.0	Contains the IP address of the device.
OrderID	STRUCT	124.0	Contains the OrderID of the device.
MxLen	BYTE		Contains the maximum length of the array.
ActLen	BYTE		Contains the current number of data in the array.
Data	ARRAY[1..20] CHAR		Contains the data.
PE	STRUCT	146.0	PROFIenergy-specific data.
ModeID	BYTE	146.0	PE_MODE_ID according to Profienergy specification.
Result	BYTE	147.0	PEErrorCode according to Profienergy specification.
PauseTime (1)	TIME	148.0	Contains the idle time in ms.
TimeToPause	TIME	152.0	Contains the time it takes for the device to go into Pause mode.
TimeToOperate	TIME	156.0	Contains the time it takes for the device to go into operating mode.
MinSleepTime	TIME	160.0	Contains the minimum time of the device in PE_SLEEP_MODE.
SleepToOperate	TIME	164.0	Contains the time it takes for the device to go from PE_SLEEP_MODE into operating mode.
StatusOperate	BOOL	168.0	Indicates the operating mode of the device.
StatusPause	BOOL	168.1	Indicates the device is in Pause mode.
StatusSleep	BOOL	168.2	Indicates the PE_SLEEP_MODE of the device.
StatusTransitOK	BOOL	168.3	Indicates the transition from one energy state to another.
StatusInTransit	BOOL	168.4	Indicates a current state transition.
StatusTransitNOK	BOOL	168.5	Indicates that the state change was not successful.
StatusError	BOOL	168.6	Indicates an error with the device.
StatusRetryEx	BOOL	168.7	Indicates an unsuccessful execution of a command. No more attempts will be made to execute this command.
CmdStartPause (1)	BOOL	169.0	Places a START_PAUSE command for this device in the queue.
CmdEndPause (1)	BOOL	169.1	Places a END_PAUSE command for this device in the queue.
CmdUpdateStatus (1)	BOOL	169.2	Places a PEM_STATUS command for this device in the queue.

Name	Data type	Offset	Comment
EnableSleep (1)	BOOL	169.3	Allows PE_SLEEP_MODE for this device. • True = device should go to PE_SLEEP_MODE when idle time is too long • False = device may not go into PE_SLEEP_MODE.
Services	WORD	170.0	Shows all supported PROFIenergy services.
UserData (2)	ARRAY[1..24] OF BYTE	172.0	User-defined data
Task	STRUCT	196.0	Job processing
Cmd	BYTE	196.0	Internal bits for job processing
CmdJ	BYTE	197.0	Internal bits for job processing
TimeStart	BOOL	198.0	Starts a delay time.
TimeStarted	BOOL	198.1	The delay time has just started.
TimeDone	BOOL	198.2	Shows the expiration of the delay time.
Done	BOOL	198.3	Indicates that this job has been completed.
DelayedCmd	BOOL	198.4	Indicates that a delayed command is still pending.
IsV1_0	BOOL	198.5	Indicates that this device is a Spec. V1.0 device.
IsWakeOnLAN	BOOL	198.6	Indicates that this device is woken up by "Wake On LAN".
RetryCount	BYTE	199.0	Repetition counter for PE_COMMANDS
Duration	TIME	200.0	Contains the value for the delay in ms.
StartTime	TIME	204.0	Contains the start time for the delay time.
MachineState	INT	208.0	Contains the internal status of the job.

(1) To be filled in by the user.
(2) Available for use by the user.

Parameter STATUS

Parameter STATUS

The output value at the STATUS parameter is divided into three areas:

- Bits 31 to 24: MESSAGE
- Bit 23 to 16: LOCATION
- Bit 15 to 0: INFORMATION

You can learn the meaning of each error code for the three areas in the following tables:

Table 9-178 Possible values for MESSAGE

Error code (W#16#...)	Description
00	No error.
50	Instruction initialized.
51	Determining configuration of the PROFINET I/O system.
52	The instruction could not determine any configured devices in the PROFINET I/O system.

Error code (W#16#...)	Description
53	Determining the logical addresses of configured devices.
54	Reading interface information out of the devices.
55	Determining I&M data (data record 0 only) of the configured devices.
56	Configuring PROFINET interface for sending the "Wake on LAN" MagicPaket™ via UDP.
57	Determining PROFIenergy capabilities of the connected devices.
62	Invalid PROFINET I/O system ID detected. The causative number is displayed in the INFORMATION field.
70	The instruction is initiated and processing jobs. The value in the INFORMATION field contains the number of currently active jobs.
80	The instruction has lost the initialization during processing of jobs. This generally occurs when either the instance DB or the user DB has been reloaded.
FF	Unknown error has occurred.

Table 9-179 Possible values for LOCATION

Error code (W#16#...)	Description
00	The instruction is either not initialized or it is idle.
70	The instruction is waiting for jobs.
71	The instruction has appended a job to the job list.
72	The instruction is preparing to send a job.
73	The instruction is sending a job to a device.
74	The instruction is waiting for a response from the device.
75	The instruction is evaluating the response of the device.
76	The instruction is removing the job from the job list.
FF	Unknown error has occurred.

Table 9-180 Possible values for INFORMATION

Error code (W#16#...)	Description
0000	No additional information available and no jobs active.
0001 -00FF	1 – 255 jobs are currently being processed.
8001	Error at parameter 1
8002	Error at parameter 2
8003	Error at parameter 3
8004	Error at parameter 4

Error code (W#16#...)	Description
8005	Error at parameter 5. This error is reported if there is no interconnection or an invalid interconnection to the user DB. Possible causes: • The user DB is too small. • The user DB is write-protected. • There is no user DB in the RAM. • The user DB is invalid for the employed CPU.
8100	An attempt has been made to place more than the maximum of 256 possible jobs. This is a temporary error, which is resolved by completing a few jobs. The placed jobs are not accepted and must be placed again.
8200	An attempt has been made to send an invalid or an unsupported PROFIenergy command (PE_COMMAND).
84xx	A communication error has occurred. The number of the device that generated the error is output to "xx".
85xx	An error has been returned from device xx. The number of the device that generated the error is output to "xx".
8600	The requested wake-up method is currently not supported.
FFFF	Unknown error has occurred.

PROFIenergy commands

PI Command "Start_Pause"

Description

Use the PE command "Start_Pause" to start the energy-saving mode. The command Start_Pause can be used to:

- Switch the PE from "ready" state (PE_ready_to_operate) to an energy-saving mode (PE_energy_saving_mode).
- The PE entity can automatically switch between the energy-saving modes.
  Energy consumption can increase or decrease when you switch energy-saving mode.

Calling the PI command "Start_Pause"

The command "Start_Pause" is called with the instruction "PE_CMD (Page 6546)" with the following parameters:

Parameter	Value	Description
CMD	1	Call of PE command "Start_Pause".
CMD_MODIFIER	0	There are no additional specifications of the command call for command "Start_Pause".
CMD_PARA_LEN	4	Length of the CMD_PARA parameter of 4 bytes.
CMD_PARA	ANY	ANY pointer to the value for "Pause_Time" (TIME).

Response frame (Service Data Response)

The following response frame data of the PE entity is written to the data block that is referenced at at parameter RESPONSE_DATA (see instruction "PE_CMD (Page 6546)"):

Attribute	Value	Data type	Description
PE_Mode_ID	1 to 255	BYTE	Identification number of the energy-saving mode
Reserved	0	BYTE	-

PI Command "End_Pause"

Description

Use the PE command "End_Pause" to terminate the energy-saving mode of the PE entity.

Calling the PI command "End_Pause"

The command "End_Pause" is called with the instruction "PE_CMD (Page 6546)" with the following parameters:

Parameter	Value	Description
CMD	2	Call of PE command "End_Pause".
CMD_MODIFIER	0	There are no additional specifications of the command call for command "End_Pause".
CMD_PARA_LEN	0	Length of the CMD_PARA parameter of 0 bytes.
CMD_PARA	irrelevant	-

Response frame (Service Data Response)

The following response frame data of the PE entity is written to the data block that is referenced at parameter RESPONSE_DATA (see "PE_CMD (Page 6546)"):

Attribute	Value	Data type	Description
Time_to_operate	-	DWORD	Expected time it takes to switch to the "PE_ready_to_operate" mode.

PI command "Query_modes" - "List_Energy_Saving_Modes"

Description

Use PE command "Query_modes" and sub-command (modifier) "List_Energy_Saving_Modes" to output all energy-saving modes (PE_Mode_ID) supported by the PE entity.

The query result is written in the form of a response frame to the data block referenced by theRESPONSE_DATA parameter.

Calling the PI command "Query_modes" - "List_Energy_Saving_Modes"

The command "List_Energy_Saving_Modes" is called with the instruction "PE_CMD (Page 6546)" with the following parameters:

Parameter	Value	Description
CMD	3	Call of PE command "Query_modes".
CMD_MODIFIER	1	Specification of the command call: Select the sub-command "List_Energy_Saving_Modes" to output the number and types of energy-saving modes supported.
CMD_PARA_LEN	0	Length of the CMD_PARA parameter of 0 bytes.
CMD_PARA	irrelevant	-

Response frame (Service Data Response)

The following response frame data of the PE entity is written to the data block that is referenced at parameter RESPONSE_DATA (see "PE_CMD (Page 6546)"):

Attribute	Value	Data type	Description
Number_of_PE_ Mode_IDs	1	BYTE	Number of energy-saving modes.
PE_Mode_IDs	-	Array [...] of BYTE	Array with the IDs of the supported energy-saving modes. The significance of the IDs depends on the PE entity.

PI command "Query_modes" - "Get_Mode"

Description

You use the PE command "Query_modes" and the sub-command (modifier) "Get_Mode" to output the attributes of the energy-saving mode which is currently enabled.

Calling the PI command "Query_modes" - "Get_Mode"

The call of the command with the instruction "PE_CMD" occurs with the following parameters:

Parameter	Value	Description
CMD	3	Call of the PE command "Query_modes"
CMD_MODIFIER	2	Specification of the command call: Select the sub-command "Get_Mode" to output the status of the mode which is currently enabled.

Parameter	Value	Description
CMD_PARA_LEN	1	Length of the CMD_PARA parameter of 1 byte.
CMD_PARA	ANY	ANY pointer to the value for PE_MODE_ID.

Response frame (Service Data Response)

The following response frame data of the PE entity is written to the data block that is referenced at parameter RESPONSE_DATA (see "PE_CMD (Page 6546)"):

Attribute	Value	Data type	Description
PE_Mode_ID	• 0 "PE_power_off" Mode • 1...254 Energy-saving mode of the PE entity (manufacturer-specific) • 255 "PE_ready_to_operate" Mode	BYTE	ID of the energy-saving mode currently enabled.
PE_Mode_Attributes	Bit 0: • = 0: Only static energy consumer and time values are available. • = 1: Dynamic energy consumer and time values are available. Bit 1 to 7: • Reserved	BYTE	
Time_min_Pause [1]	Time difference without date	DWORD	Minimum pause time for the PI mode. The minimum pause time is the sum of the values of the following attributes: • Time_to_Pause • Time_to_operate • Time_min_length_of_stay See the description of the "PAUSE_TIME parameter" of the instruction "PE_START_END: Start and exit energy-saving mode (Page 6542)".
Time_to_Pause [1]	Time difference without date	DWORD	Switch off time: Duration from call of energy-saving mode to start of energy-saving mode (transition duration of PE_ready_to_operate to PE_energy_saving_mode). The switch off time depends on the PE entity.
Time_to_operate [1]	Time difference without date	DWORD	Switch on time: Duration of the transition from energy-saving mode (PE_energy_saving_mode) to ready to operate mode (PE_ready_to_operate). The PE entity calculates the duration dynamically in the output operation.
Time_min_length_of_stay [1]		DWORD	Minimum active period of the energy-saving mode on the PE entity.

Attribute	Value	Data type	Description
Time_max_length_ of_stay [1]		DWORD	Maximum active period of the energy-saving mode on the PE entity.
Mode_Power_ Consumption [2]		REAL	Power consumption of the PE entity in active energy-saving mode. Unit: kW
Energy_ Consumption_ to_pause [2]		REAL	Energy consumption of the PE entity at the transition from the ready to operate mode (PE_ready_to_operate) to the energy-saving mode (PE_energy_saving_mode) Unit: kWh
Energy_ Consumption_ to_operate [2]		REAL	Energy consumption of the PE entity at the transition from the energy-saving mode (PE_energy_saving_mode) to the ready to operate mode (PE_ready_to_operate) Unit: kWh
[1] If the duration is infinite, 0xFFFFFFFF will be output. If the duration is zero, "0" will be output. [2] If the PE entity has no defined energy and power consumption, the "0.0" is output.			

PI command "PEM_Status"

Description

Using PE command "PEM_Status", you can query the currently active energy-saving mode of a PE entity.

Calling the PI command "PEM_Status"

The command "PEM_Status" is called with the instruction "PE_CMD" with the following parameters:

Parameter	Value	Description
CMD	4	Call of PE command "PEM_Status".
CMD_MODIFIER	0	There are no other specifications of the command call for command "PEM_Status".
CMD_PARA_LEN	0	Length of the CMD_PARA parameter of 0 bytes.
CMD_PARA	irrelevant	-

Response frame (Service Data Response)

The following response frame data of the PE entity is written to the data block that is referenced at parameter RESPONSE_DATA (see "PE_CMD (Page 6546)"):

Attribute	Value	Data type	Description
PE_Mode_ID_ Source	• 0 "PE_power_off" mode • 1 to 254 Energy-saving mode of the PE entity (manufacturer-specific) • 255 "PE_ready_to_operate" mode	BYTE	Mode of the PE entity before the PE command is sent.
PE_Mode_ID_ Destination	• 0 "PE_power_off" mode • 1 to 254 Energy-saving mode of the PE entity (manufacturer-specific) • 255 "PE_ready_to_operate" mode	BYTE	Mode of the PE entity after a PE command is executed.
Time_to_operate	Time difference without date	DWORD	Switch on time: Duration of the transition from energy-saving mode (PE_energy_saving_mode) to ready to operate mode (PE_ready_to_operate). The PE entity calculates the duration dynamically in the output operation.
Remaining_time_to_ destination	Time difference without date	DWORD	Remaining time to switch to another mode.
Mode_Power_ Consumption		REAL	Power consumption of the PE entity in active energy-saving mode. Unit: kW
Energy_ Consumption_ to_Destination		REAL	Energy consumption for the current PI transition Unit: kWh
Energy_ Consumption_ to_operate		REAL	Energy consumption of the PE entity at the transition from the energy-saving mode (PE_energy-saving mode) to the ready to operate mode (PE_ready_to_operate) Unit: kWh

PI command "PE_Identify"

Description

Use PE command "PE_Identify" to read the number and description of PE commands supported by the PE entity. The number of specific commands supported depends on the PE

entity. As PE_Identify is itself a PE command, at least three supported PE commands will be output upon a positive response: Start_Pause, End_Pause and PE_Identify.

Calling the PI command "PE_Identify"

The command "PE_Identify" is called with the instruction "PE_CMD (Page 6546)" with the following parameters:

Parameter	Value	Description
CMD	5	Call of the command "PE_Identify".
CMD_MODIFIER	0	There are no other specifications of the command call for command "PE_Identify".
CMD_PARA_LEN	0	Length of the CMD_PARA parameter of 0 bytes.
CMD_PARA	irrelevant	-

Response frame (Service Data Response)

The following response frame data of the PE entity is written to the data block that is referenced at parameter RESPONSE_DATA (see "PE_CMD (Page 6546)"):

Attribute	Value	Data type	Description
Count [1]	6	BYTE	Number of supported PROFIenergy commands
Start_Pause	1	BYTE	First PE command supported (Service_Request_ID)
End_Pause	2	BYTE	...
Query_Modes	3	BYTE	...
PEM_Status	4	BYTE	...
PE_Identify	5	BYTE	...
Query_Measurement	16	BYTE	Last PE command supported (Service_Request_ID)
[1] The number of commands supported is manufacturer-specific and depends on the PE entity used. The values given are an example of a response frame where all 6 PE commands are supported.			

PI Command "Query_Measurement" - "Get_Measurement_list"

Description

Use PE command "Query_Measurement" and sub-command (modifier) "Get_measurement_list" to query the specific measured values supported by the PE entity. The supported measured values are output as a list in the data block referenced by the RESPONSE_DATA parameter.

Calling the PI command "Query_Measurement" - "Get_Measurement_list"

The command is called with the instruction "PE_CMD (Page 6546)" with the following parameters:

Parameter	Value	Description
CMD	16	Call of the command "Query_Measurement".
CMD_MODIFIER	1	Specification of the command call: Select the sub-command "Get_Measurement_List" to output a list of supported measured values.
CMD_PARA_LEN	0	Length of the CMD_PARA parameter of 0 bytes.
CMD_PARA	irrelevant	-

Response frame (Service Data Response)

The following response frame data of the PE entity is written to the data block that is referenced at at parameter RESPONSE_DATA (see "PE_CMD (Page 6546)"):

Attribute	Value	Data type	Description
Count	-	BYTE	Number of measurement IDs
reserved	-	BYTE	
...			
Measurement_ID	-	WORD	First supported Measurement_ID The Measurement_ID is manufacturer-specific. For more information, refer to the manual of the respective PE entity.
Accuracy_Domain	-	BYTE	See"accuracy domain" table.
Accuracy_Class	-	BYTE	See "accuracy classes" table.
Range	-	REAL	Specifies the scale end value of the measured value (only for accuracy domain 1). The attribute range uses the same unit as defined using attribute Measurement_ID (only one unit is used for each Measurement_ID).
...			
Measurement_ID	-	WORD	Last supported Measurement_ID
Accuracy_Domain	-	BYTE	See"accuracy domain" table.
Accuracy_Class	-	BYTE	See "accuracy classes" table.
Range	-	REAL	Specifies the scale end value of the measurement value (only for accuracy domain 1). The attribute range uses the same unit as defined using attribute Measurement_ID (only one unit is used for each Measurement_ID).

Accuracy domains

Accuracy domain	Description
0	Reserved
1	The accuracy deviation is output in percentages of the scale end value. The percentage of the possible deviation is divided into accuracy classes (see table: accuracy classes of the accuracy domains 1 and 2).

Accuracy domain	Description
2	The accuracy deviation is output in percentages of the current measurement value. The percentage of the possible deviation is divided into accuracy classes (see table: accuracy classes of the accuracy domains 1 and 2).
3	The measuring accuracy adheres to the standard IEC 61557-12. The function performance classes for performance measurement and monitoring devices (PMD) without external sensors, and the system performance classes for PMD with external sensors, are coded as set out in the "Accuracy classes of accuracy domain 3" table.
4	The entry of the accuracy adheres to the standard EN 50470-3, chapter 8 (see also table: accuracy classes of accuracy domain 4).

Accuracy classes

Table 9-181 Accuracy classes of the accuracy domains 1 and 2

Accuracy class	0	1	2	3	4	5	6	7	8
Meaning	Reserved	0.01%	0.02%	0.05%	0.1%	0.2%	0.5%	1%	1.5%

Accuracy class	9	10	11	12	13	14	15	>15
Meaning	2%	2.5%	3%	5%	10%	20%	>20%	Not defined

Table 9-182 Accuracy classes of accuracy domain 3

Accuracy class	0	1	2	3	4	5	6	7	8
Meaning	Reserved	0.02	0.05	0.1	0.2	0.5	1	1.5	2

Accuracy class	9	10	11	12	13	14	>13
Meaning	2.5	3	5	10	20	20%	Not defined

Table 9-183 Accuracy classes of accuracy domain 4

Accuracy class	0	1	2	3	4	5	6	>7
Meaning	Reserved	0.5	1.0	1.5	2.0	2.5	3.0	Not defined

PI Command "Query_Measurement" - "Get_Measurement_values"

Description

Use PE command "Query_Measurement" and sub-command (modifier) "Get_measurement_values" to output a list of measured values supported by the PE entity. The measured values are output as a list in the data block referenced by the RESPONSE_DATA parameter.

Calling the PI command "Query_Measurement" - "Get_Measurement_values"

The command is called with the instruction "PE_CMD (Page 6546)" with the following parameters:

Parameter	Value	Description
CMD	16	Call of the command "Query_Measurement".
CMD_MODIFIER	2	Specification of the command call: Select the command "Get_Measurement_Values" to output a list of supported measurement values.
CMD_PARA_LEN	0	Depends on the number of measurement values. The length of the parameters results from the attribute count and the sum of the lengths of the attributes for the transferred measurement values.
CMD_PARA	ANY	ANY pointer to data structure with list of measured values to be queried (see "CMD_PARA parameter").

Parameter CMD_PARA

The structure specified with the ANY pointer at the CMD_PARA parameter must have the following setup:

Attribute	Value	Data type	Description
Count	-	BYTE	Number of measured values (Measurement-IDs)
reserved	0	BYTE	Not used
Measurement_ID	-	WORD	First measured value queried
...			
Measurement_ID	-	WORD	Last measured value queried

Response frame (Service Data Response)

The following response frame data of the PE entity is written to the data block that is referenced at parameter RESPONSE_DATA (see "PE_CMD (Page 6546)"):

Attribute	Value	Data type	Description
Count [1]	-	BYTE	Number of measured values (Measurement-IDs)
reserved	0	BYTE	Not used
Length_of_Structure	2 to 65535	WORD	Length of the structure in bytes
Measurement_Data_ Structure_ID	1 = simple value	BYTE	Defines the following structure.
Measurement_ID	0 to 65535	WORD	ID of the supported measurement value.
Status_of_ Measurement_Value	1 to 3	BYTE	Status of the measurement value: • 1: Valid • 2: Not supported • 3: Invalid
Transmission_Data_ Type	-	REAL	

Attribute	Value	Data type	Description
End_of_demand	-	TOD	Optional time stamping with data type TimeOfDay.
...			
Length_of_Structure	-	WORD	Length of the structure in bytes
Measurement_Data_Structure_ID	-	BYTE	Defines the following structure.
Measurement_ID	-	WORD	ID of the supported measurement value.
Status_of_Measurement_Value	-	BYTE	Status of the measurement value: • 1: Valid • 2: Not supported • 3: Invalid
Transmission_Data_Type	-	REAL	
End_of_demand	-	TOD	Optional time stamping with data type TimeOfDay.
[1] If the data length of the measurement values queried exceeds the size of the PDU (Protocol Data Unit) of the protocol layer, the data will not be completely transferred and only the supported measurement values will be output.			

iDevice / iSlave

PE_I_DEV: Control PROFIenergy commands in the I-Device

Description

The instruction "PE_I_DEV" is used to process the PROFIenergy profile in the intelligent IO device (iDevice). The functions which are executed by the firmware in standard PROFIenergy-compatible IO devices, such as the ET 200S, are implemented in the iDevice by the instruction "PE_I_DEV" and the corresponding auxiliary blocks:

- The instruction "PE_I_DEV" is called cyclically by the user program of the iDevice and receives all PROFIenergy commands.
- The PROFIenergy response is generated by configuring an auxiliary block. The response in the pause is fully programmable. The response data must be provided within 10 seconds; otherwise, "State conflict 0x80B5" will occur at the STATUS parameter of the instruction in the IO controller.

No specific knowledge of the PROFIenergy standard is required to use the instruction.

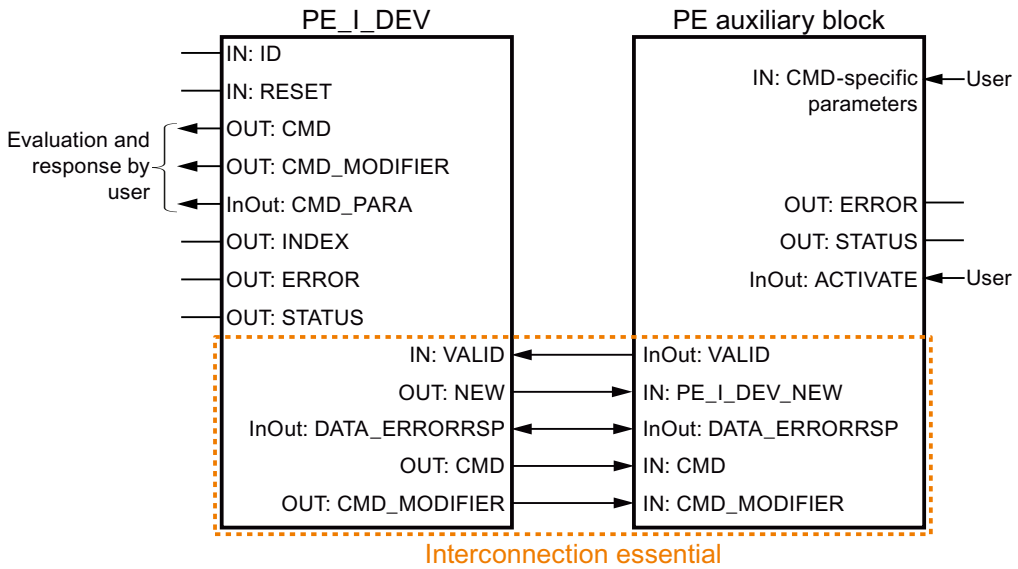
PROFIenergy auxiliary blocks (PE auxiliary blocks)

The PE auxiliary blocks are used to generate the response frame. Enter the response data (in plain text) at the input parameters of the relevant block.

- For each PROFIenergy command, there is a corresponding auxiliary block for a positive response:
  - PE command "Start_Pause": PE_Start_RSP (Page 6579)
  - PE command "End_Pause": PE_End_RSP (Page 6581)
  - PE command "Query_modes" - "List_Energy_Saving_Modes": PE_List_Modes_RSP (Page 6582)
  - PE command "Query_modes" - "Get_Mode": PE_Get_Mode_RSP (Page 6583)
  - PE command "PEM_Status": PE_PEM_Status_RSP (Page 6585)
  - PE command "PE_Identify": PE_Identify_RSP (Page 6587)
  - PE command "Query_Measurement" - "Get_Measurement_list": PE_Measurement_List_RSP (Page 6588)
  - PE command "Query_Measurement" - "Get_Measurement_values": PE_Measurement_Value_RSP (Page 6590)
- Irrespective of the PROFIenergy command, there is an additional shared auxiliary block for a negative response (see PE_Error_RSP (Page 6578)).

Interconnection of the auxiliary blocks

The instruction "PE_I_DEV" and the auxiliary blocks are coordinated. Some of the parameters are simply interconnected. The diagram below shows which of the parameters need to be interconnected.



Parameters

The following table shows the parameters of the instruction "PE_I_DEV":

Parameter	Declaration	Data type	Memory area	Description
RESET	Input	BOOL	I, Q, M, D, L	Resets the instruction.
ID	Input	DWORD	I, Q, M, D, L or constant	Address of the iDevice The address may be taken from the hardware configuration.
VALID	Input	BOOL	I, Q, M, D, L	The response data for the PROFIenergy controller is ready and can be sent.
CMD_PARA	Output	ANY	I, Q, M, D, L	Parameters for: • Get mode: PE_mode_ID • Get measurement values: List of Measurement_IDs (list of IDs of the tags to be read; either one tag or multiple tags may be read at any given time). Maximum length: 234 bytes
DATA_ERRORRSP	InOut	ANY	I, Q, M, D, L	Pointer to the data area containing the acknowledgment data for the PROFIenergy controller. This must correspond to the pointer which is also used with the auxiliary blocks.
INDEX	Output	INT	I, Q, M, D, L	Data record number of the PROFIenergy record (set at 0x80A0)
CMD	Output	INT	I, Q, M, D, L	Service-Request-ID of the PROFIenergy command in accordance with the PROFIenergy profile (see "CMD and CMD_MODIFIER parameters"). Further PE commands (Service-Request-IDs) are possible following extensions of the PROFIenergy profile.
CMD_MODIFIER	Output	INT	I, Q, M, D, L	PROFIenergy sub-command: • Only when CMD=3 or CMD=16; see "CMD and CMD_MODIFIER parameters". • For all other commands: "0". Further sub-commands are possible following extensions of the PROFIenergy profile.
NEW	Output	BOOL	I, Q, M, D, L	New data available from the PROFIenergy controller.
ERROR	Output	BOOL	I, Q, M, D, L	Command completed with error.
STATUS	Output	DWORD	I, Q, M, D, L	Error information (see "STATUS parameter").

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameters CMD and CMD_MODIFIER

CMD	CMD_ MODIFIER	PROFIenergy command	Description
01	0	Start_Pause	Start energy-saving mode or switch to a different energy-saving mode.
02	0	End_Pause	End energy-saving mode.
03	1	Query_Modes - List energy saving Modes	Output the energy-saving modes supported.
	2	Query_Modes - Get Mode	Output attributes of the energy-saving mode currently enabled.
04	0	PEM_Status	Query status of energy-saving mode.
05	0	PE_Identify	Read out the number and description of PE commands supported.
16	1	Query_Measurement - Get_Measurement_List	List of measured values supported by the PE entity.
	2	Query_Measurement - Get_Measurement_Values	Output of the measured values of the PE entity.

STATUS parameter

Error information is output at the STATUS output parameter. If it is interpreted as ARRAY[1...4] of BYTE, the error information has the following structure:

Field element	Name	Meaning
STATUS[1]	Function_Num	Cause of the error • B#16#00: No error • B#16#DE: Error during reading of data record • B#16#DF: Error during writing of data record • B#16#C0: Error message from the instruction "PE_I_DEV" or from the internal communication instructions "RDREC (Page 6459)" and "WRREC (Page 6461)".
STATUS[2]	Error_Decode	Location of the error ID • 80: DPV1 error as defined in IEC 61158-6 or application-specific
STATUS[3]	Error_Code_1	Error ID (when Error_Decode = 80): • B1: Write length error (error in write length or insufficient length information with the data type ANY).
STATUS[4]	Error_Code_2	For PROFINET errors: Output of IO controller error message If no PROFINET error has occurred, the value in STATUS[4] = "0".

Note

Error messages of the instructions RDREC and WRREC

The instruction "PE_I_DEV" uses the instructions "WRREC (Page 6461)" and "RDREC (Page 6459)" for communication. Error messages for these instructions are output in the field elements STATUS[1] to STATUS[4].

Please see the description of the corresponding STATUS (Page 6471) parameter for an explanation of the error codes of the "WRREC (Page 6461)" and "RDREC (Page 6459)" instructions.

Auxiliary blocks of the instruction PE_I_DEV

PE_Error_RSP: Generate negative answer to command

Description

The auxiliary block "PE_Error_RSP" (Response with failure) generates a negative response if the command requested is not supported (either in general or temporarily). Response generation does not depend on the command requested.

Parameter

The following table shows the parameters of the "PE_Error_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 6574)" instruction. The auxiliary block is only processed if the parameter value is set to "1".
ERROR_CODE	Input	BYTE	I, Q, M, D, L or constant	Error number
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 6574)" instruction.

Parameter	Declaration	Data type	Memory area	Description
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 6574)" instruction. The auxiliary block sets the parameter as soon as the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ ERRORRSP	InOut	ANY	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 6574)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in ANY setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

PE_Start_RSP: Generate answer to command at start of pause

Description

The auxiliary block "PE_Start_RSP" (Start Pause) generates the response to the PE command Start_Pause (Page 6564). The instruction returns the energy-saving mode to which the device has switched (PE_MODE_ID parameter).

If the response differs depending on the length of the pause, you can return this fact in the feedback on the energy-saving mode (PE_Mode_ID = 1 for a brief pause, PE_Mode_ID = 2 for a longer pause, etc.).

Parameter

The following table shows the parameters of the "PE_Start_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 6574)" instruction. The auxiliary block is only processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 6574)" instruction.
PE_MODE_ID	Input	BYTE	I, Q, M, D, L or constant	PE mode that the process assumes
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 6574)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 6574)" instruction. The auxiliary block sets the parameter as soon as the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ERRORRSP	InOut	ANY	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 6574)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in ANY setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

PE_End_RSP: Generate answer to command at end of pause

Description

The auxiliary block "PE_End_RSP" generates the response to the PE command End_Pause (Page 6565). The response returned is the time required to switch from the current mode to "Ready_To_Operate" mode.

Parameter

The following table shows the parameters of the "PE_End_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 6574)" instruction. The auxiliary block is only processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 6574)" instruction.
Time_to_Operate	Input	DWORD	I, Q, M, D, L or constant	Time required to switch from the current mode to "Ready_To_Operate".
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 6574)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 6574)" instruction. The auxiliary block sets the parameter as soon as the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ ERRORRSP	InOut	ANY	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 6574)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes

Parameter	Declaration	Data type	Memory area	Description
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in ANY setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

PE_List_Modes_RSP: Generate queried energy savings modes as answer

Description

The auxiliary block "PE_List_Modes_RSP" generates the response to the PE command List_Energy_Saving_Modes (Page 6565). The response generated includes the number and the IDs of energy-saving modes supported.

Parameter

The following table shows the parameters of the "PE_List_Modes_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 6574)" instruction. The auxiliary block is only processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 6574)" instruction.
CMD_MODIFIER	Input	INT	I, Q, M, D, L or constant	PROFIenergy sub-command (evaluated only if CMD=3, or CMD=16). The parameter must be interconnected with the CMD_MODIFIER output parameter of the "PE_I_DEV" instruction.
Number_of_PE_Mode_IDs	Input	BYTE	I, Q, M, D, L or constant	Number of energy-saving modes supported. Permitted values: 1 to 254
PE_MODE_ID	Input	ANY	I, Q, M, D, L	Points to the area in which the IDs of the energy-saving modes supported (PE_Mode_ID) are stored. Valid range: 1 to 254.

Parameter	Declaration	Data type	Memory area	Description
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 6574)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 6574)" instruction. The auxiliary block sets the parameter as soon as the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ ERRORRSP	InOut	ANY	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 6574)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in ANY setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

PE_Get_Mode_RSP: Generate queried energy data as answer

Description

The auxiliary block "PE_Get_Mode_RSP" generates the response to the command Get_Mode (Page 6566). The response frame contains the time, performance, or energy data of an energy-saving mode.

Parameter

The following table shows the parameters of the "PE_Get_Mode_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 6574)" instruction. The auxiliary block is only processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 6574)" instruction.
CMD_MODIFIER	Input	INT	I, Q, M, D, L or constant	PROFIenergy sub-command (evaluated only if CMD=3, or CMD=16). The parameter must be interconnected with the CMD_MODIFIER output parameter of the "PE_I_DEV" instruction.
PE_Mode_ID	Input	BYTE	I, Q, M, D, L or constant	ID of the energy-saving mode currently enabled.
Time_min_Pause	Input	DWORD	I, Q, M, D, L or constant	Minimum pause duration for this PE energy-saving mode. This is the sum of the following three parameters: • Time_to_Pause • Time_to_operate • Time_min_length_of_stay
Time_to_Pause	Input	DWORD	I, Q, M, D, L or constant	Time between the edge at the START parameter (see "PE_I_DEV (Page 6574)") and the requested PE energy-saving mode being reached.
Time_to_Operate	Input	DWORD	I, Q, M, D, L or constant	Maximum switch-on time to "PE_ready_to_operate". The "Time_to_operate" parameter can be used directly for the relevant calculations. The value may be a static maximum, or be calculated dynamically by the PE entity.
Time_min_Lenght_ of_stay	Input	DWORD	I, Q, M, D, L or constant	Minimum dwell time of the PE entity in this PE mode.
Time_max_Lenght_o f_stay	Input	DWORD	I, Q, M, D, L or constant	Maximum dwell time of the PE entity in this PE mode.
Mode_Power_ Consumption	Input	DWORD	I, Q, M, D, L or constant	Energy consumption in the current PE mode in [kW].
Energy_Consum_ to_Pause	Input	DWORD	I, Q, M, D, L or constant	Energy consumption from "PE_ready_to_operate" to the current PE mode in [kWh].
Energy_Consum_ to_operate	Input	DWORD	I, Q, M, D, L or constant	Energy consumption from the current PE mode to "PE_ready_to_operate" in [kWh].

Parameter	Declaration	Data type	Memory area	Description
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 6574)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 6574)" instruction. The auxiliary block sets the parameter as soon as the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ ERRORRSP	InOut	ANY	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 6574)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in ANY setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

PE_PEM_Status_RSP: Generate PEM status as answer

Description

The auxiliary block "PE_PEM_Status_RSP" generates the response to the command PEM_Status (Page 6568).

Parameter

The following table shows the parameters of the "PE_PEM_Status_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 6574)" instruction. The auxiliary block is only processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 6574)" instruction.
PE_MODE_ID_ Source	Input	BYTE	I, Q, M, D, L or constant	Source and Destination for PEM_STATUS. Values: • 0x00: PE_POWER_OFF • 0x01 – 0xFE: manufacturer-specific • 0xFF: PE_READY_TO_OPERATE
PE_MODE_ID_ Destination	Input	BYTE	I, Q, M, D, L or constant	
Time_to_Operate[1]	Input	DWORD	I, Q, M, D, L or constant	Maximum switch-on time to "PE_ready_to_operate". "Time_to_operate" can be used directly for the relevant calculations. The value may be a static maximum, or be calculated dynamically by the PE entity.
Remaining_time_ to_destination[1]	Input	DWORD	I, Q, M, D, L or constant	Optional: Time remaining until the requested PE mode. Dynamic value or static maximum value
Mode_Power_ Consumption[2]	Input	DWORD	I, Q, M, D, L or constant	Energy consumption in the current PE mode in [kW].
Energy_ Consumption_ to_Destination[2]	Input	DWORD	I, Q, M, D, L or constant	Energy consumption in the time until the requested PE mode in [kW].
Energy_ Consumption_ to_operate[2]	Input	DWORD	I, Q, M, D, L or constant	Energy consumption from the current PE mode to "PE_ready_to_operate" in [kWh].
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 6574)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 6574)" instruction. The auxiliary block sets the parameter as soon as the response data for the PROFIenergy controller is available and ready for transmission.

Parameter	Declaration	Data type	Memory area	Description
DATA_ ERRORRSP	InOut	ANY	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 6574)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in ANY setting, e.g. incorrect range
[1] If the time period is unlimited, the maximum value "0xFFFFFFFF" can be specified. If the time period is "Zero", "0x00" can be used. [2] If no energy consumption value has been defined, "0.0" can be specified.				

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

PE_Identify_RSP: Generate supported PROFIenergy commands as answer

Description

The auxiliary block "PE_Identify_RSP" generates the response to the command PE_Identify (Page 6569). In the response to the command, specify which PROFIenergy commands are supported. Please note that PE_IDENTIFY is itself a PE command and has to be included in the response.

Parameter

The following table shows the parameters of the "PE_Identify_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NE W	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 6574)" instruction. The auxiliary block is only processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 6574)" instruction.
Start_Pause	Input	BOOL	I, Q, M, D, L or constant	One parameter for each of the relevant PROFIenergy commands: • 0: PE command is not supported • 1: PE command is supported
End_Pause	Input	BOOL	I, Q, M, D, L or constant	

Parameter	Declaration	Data type	Memory area	Description
Query_Modes	Input	BOOL	I, Q, M, D, L or constant	
PEM_Status	Input	BOOL	I, Q, M, D, L or constant	
PE_Identify	Input	BOOL	I, Q, M, D, L or constant	
Query_ Measurement	Input	BOOL	I, Q, M, D, L or constant	
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 6574)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 6574)" instruction. The auxiliary block sets the parameter as soon as the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ ERRORRSP	InOut	ANY	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 6574)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in ANY setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

PE_Measurement_List_RSP: Generate list of supported measured values as answer

Description

The auxiliary block "PE_Measurement_List_RSP" generates the response to the command Get_measurement_list (Page 6570). In the response, specify which measured values (Measurement-IDs) are supported.

Parameter

The following table shows the parameters of the "PE_Measurement_List_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 6574)" instruction. The auxiliary block is only processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 6574)" instruction.
CMD_MODIFIER	Input	INT	I, Q, M, D, L or constant	PROFIenergy sub-command (evaluated only if CMD=3, or CMD=16). The parameter must be interconnected with the CMD_MODIFIER output parameter of the "PE_I_DEV" instruction.
Count	Input	BYTE	I, Q, M, D, L or constant	Number of measured values supported (measurement IDs)
Measurement_List	Input	ANY	D	Pointer to the array with the Measurement_IDs supported. For information on the structure of the array in accordance with the PROFIenergy profile, see: PI Command "Query_Measurement" - "Get_Measurement_list" (Page 6570)
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 6574)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 6574)" instruction. The auxiliary block sets the parameter as soon as the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ERRORRSP	InOut	ANY	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 6574)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes

Parameter	Declaration	Data type	Memory area	Description
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in ANY setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

PE_Measurement_Value_RSP: Generate queried measured values as answer

Description

The auxiliary block "PE_Measurement_Value_RSP" generates the response to the command Get_measurement_values (Page 6572). In the response, return the values of the requested measurements.

Parameter

The following table shows the parameters of the "PE_Measurement_Value_RSP" auxiliary block:

Parameter	Declaration	Data type	Memory area	Description
PE_I_DEV_NEW	Input	BOOL	I, Q, M, D, L or constant	The parameter must be interconnected with the NEW output parameter of the "PE_I_DEV (Page 6574)" instruction. The auxiliary block is only processed if the parameter value is set to "1".
CMD	Input	INT	I, Q, M, D, L or constant	Service-Request-ID of the PROFIenergy command The parameter must be interconnected with the CMD output parameter of the "PE_I_DEV (Page 6574)" instruction.
CMD_MODIFIER	Input	INT	I, Q, M, D, L or constant	PROFIenergy sub-command (evaluated only if CMD=3, or CMD=16). The parameter must be interconnected with the CMD_MODIFIER output parameter of the "PE_I_DEV" instruction.
Count	Input	BYTE	I, Q, M, D, L or constant	Number of measured values (Measurement_Values).
Measurement_Values	Input	ANY	D	Pointer to the array with the measured values (Measurement_IDs). For information on the structure of the array in accordance with the PROFIenergy profile, see PI Command "Query_Measurement" - "Get_Measurement_values" (Page 6572)

Parameter	Declaration	Data type	Memory area	Description
ACTIVATE	InOut	BOOL	I, Q, M, D, L	The instruction copies the input parameters to the DATA_ERROR_RSP data area at a rising edge at input ACTIVATE. The parameter is then reset by the instruction. The parameter must be set within 10 seconds after a rising edge is detected at parameter NEW of the "PE_I_DEV (Page 6574)" instruction.
VALID	InOut	BOOL	I, Q, M, D, L	The parameter must be interconnected with the VALID input of the "PE_I_DEV (Page 6574)" instruction. The auxiliary block sets the parameter as soon as the response data for the PROFIenergy controller is available and ready for transmission.
DATA_ ERRORRSP	InOut	ANY	D	Pointer on the data area in which the response data is stored. The parameter is identical to the pointer for DATA_ERRORRSP of the "PE_I_DEV (Page 6574)" instruction. The addressed data area contains the complete PROFIenergy frame. Minimum length: 244 bytes
ERROR	Output	BOOL	I, Q, M, D, L	• "0": No error • "1": Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	• "0": No error • "0x80B1": Error in ANY setting, e.g. incorrect range

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

Module parameter assignment

Writing and reading data records

Principle

Some modules have a write-only system data area to which your program can transfer data records. This area contains data records with numbers from 0 to a maximum of 240. Not every module contains all of the data records (see following table).

Modules can also have a read-only system data area in which your program can read data records. This area contains data records with numbers from 0 to a maximum of 240. Not every module contains all of the data records (see following table).

Note

There are modules that have both system data areas. These are physically separate areas and the only thing they have in common is their logical structure.

Write-only system data area

The following table shows the structure of the write-only system data area. This table also shows how the permitted size of the individual data records and which instructions can be used to write them.

Data record number	Contents	Size	Restriction	Can be written with instruction
0	Parameter	For S7-300: 2 to 14 Bytes	Can only be written by an S7-400	WR_DPARM (Page 6601) PARM_MOD (Page 6596)
1	Parameter	For S7-300: 2 to 14 Bytes (Data records 0 and 1 together have a total of exactly 16 bytes.)	-	WR_PARM (Page 6600) WR_DPARM (Page 6601) PARM_MOD (Page 6596)
2 to 127	User data	Each ≤ 240 bytes	-	WR_PARM (Page 6600) WR_DPARM (Page 6601) PARM_MOD (Page 6596) WR_REC (Page 6496)
128 to 240	Parameter	Each ≤ 240 bytes	-	WR_PARM (Page 6600) WR_DPARM (Page 6601) PARM_MOD (Page 6596) WR_REC (Page 6496)

Read-only system data area

The following table shows the structure of the read-only system data area. This table also shows the permitted size of the individual data records and which instructions can be used to read them.

Data record number	Contents	Size	Can be read with instruction
0	Module-specific diagnostic data (set as standard for the entire system)	4 bytes	RDSYSST (Page 6681) (SZL_ID00B1H) RD_REC (Page 6491)
1	Channel-specific diagnostic data (incl. data record 0)	• For S7-300: 16 bytes • with S7-400: 4 to 220 Bytes	RDSYSST (Page 6681) (SZL_ID00B2H and 00B3H) RD_REC (Page 6491)
2 to 127	User data	Each ≤ 240 bytes	RD_REC (Page 6491)
128 to 240	Diagnostic data	Each ≤ 240 bytes	RD_REC (Page 6491)

System resources

If you start several asynchronous data record transfers one after the other with only short intervals between them, the allocation of system resources by the operating system ensures that all the jobs are executed and that they do not interfere with each other.

If the limits of the system resources are reached, this is indicated in RET_VAL. You can remedy this temporary error situation by simply repeating the job.

The maximum number of "simultaneously" active jobs of a single instruction type depends on the CPU.

RD_DPAR: Read module data record

Description

You use the instruction to read the data record with the number INDEX of the addressed component from the configured system data. This may be a module in a central rack or a distributed component (PROFIBUS DP or PROFINET IO).

The value TRUE for the output parameter VALID indicates that the data record was successfully transferred to the destination area RECORD. In this case, the LEN output parameter contains the length of the read data in bytes.

If an error has occurred during transfer of the data record, this is indicated by the output parameter ERROR . In this case, the output parameter STATUS contains the error information.

Functional description

The "RD_DPAR" instruction works asynchronously, that is, its execution extends over multiple calls. You start the data record transfer by calling "RD_DPAR" with REQ = 1.

The output parameter BUSY and bytes 2 and 3 of the output parameter STATUS show the status of the job. Bytes 2 and 3 of STATUS match the output parameter RET_VAL of the instructions that operate asynchronously.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 5367).

The transfer of the data record is complete when the output parameter BUSY has the value FALSE.

Parameter

The following table shows the parameters of the instruction "RD_DPAR":

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = 1: Read request
LADDR	Input	WORD	I, Q, M, D, L or constant	Any logical address of the module In bit 15 you indicate whether an input address (bit 15 = 0) or an output address (bit 15 = 1) is present.
INDEX	Input	INT	I, Q, M, D, L or constant	Data record number
VALID	Output	BOOL	I, Q, M, D, L	New data record was received and is valid
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The job is not yet complete.
ERROR	Output	BOOL	I, Q, M, D, L	ERROR = 1: An error occurred during the reading process.
STATUS	Output	DWORD	I, Q, M, D, L	• Call ID (bytes 2 and 3) or error code • Byte 1: B#16#00, if no error. Otherwise function ID from DPV1-PDU: In the case of error for data record reading B#16#DE, in the case of error for data record writing B#16#DF. If no DPV1 protocol element is used: B#16#C0. • Byte 4: Manufacturer-specific error ID expansion
LEN	Output	INT	I, Q, M, D, L	Length of the read data record information
RECORD	InOut	ANY	I, Q, M, D, L	Destination area for the read data record Note: Note that for S7-300 CPUs the parameter RECORD always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and leads to an error message in the user program.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter ERROR and STATUS

See also: PARM_MOD: Transfer module data records (Page 6596).

RD_DPARA: Read module data record asynchronously

Description

You use the instruction to read the data record with the number RECNUM of a selected module from the configured system data. The read data record is entered into the destination area spanned by the RECORD parameter.

Functional description

The "RD_DPARA" instruction works asynchronously, that is, its execution extends over multiple calls. You start the job by calling the instruction with REQ = 1.

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 5367).

Parameter

The following table shows the parameters of the instruction "RD_DPARA":

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = 1: Read request
LADDR	Input	WORD	I, Q, M, D, L or constant	Any address of the module. For an output address, the most significant bit must be set.
RECNUM	Input	BYTE	I, Q, M, D, L or constant	Data record number (permitted values: 0 to 240)
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code. If no error occurred during the transmission, the following two cases are distinguished: • RET_VAL contains the length of the actually read data record in bytes if the destination area is larger than the read data record. • RET_VAL contains "0" if the length of the read data record is equal to the length of the destination area.

Parameters	Declaration	Data type	Memory area	Description
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The job is not yet complete.
RECORD	Output	ANY	I, Q, M, D, L	Destination area for the data record read. Only the BYTE data type is permitted. Note: Note that for S7-300 CPUs the parameter RECORD always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and leads to an error message in the user program.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

See also: PARM_MOD: Transfer module data records (Page 6596).

PARM_MOD: Transfer module data records

Description

You use the instruction "PARM_MOD" to transfer all configured data records of a module to the module. With this instruction, it is irrelevant whether the data records are static or dynamic.

Parameters

The following table shows the parameters of the instruction "PARM_MOD":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ= 1: Write request
IOID	Input	BYTE	I, Q, M, D, L or constant	Address area identifier: • B#16#54 = Peripheral input (PI) • B#16#55 = Peripheral output (PQ) If the module is a mixed module, the area identifier of the lower address must be specified. If the addresses are identical, specify B#16#54.
LADDR	Input	WORD	I, Q, M, D, L or constant	Logical base address of the module. For a mixed module, the lower of the two addresses must be specified.

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The writing process is not yet complete.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Error information

The "real" error information (error codes W#16#8xyz) can be divided into two classes:

- Temporary errors (error codes W#16#80A2 to 80A4, 80Cx):
  This type of error can possibly be eliminated without user action, meaning it is helpful to call the instruction again (multiple calls, if necessary).
  Example of a temporary error: Resources required are currently in use (W#16#80C3).
- Permanent errors (error codes W#16#809x, 80A1, 80Bx, 80Dx):
  This type of error does not correct itself. Retries to call the instruction will only be successful after the error has been eliminated.
  Example of a permanent error: Incorrect length of the data record to be transferred (W#16#80B1).

Note

If you transfer data records to a DPV1 slave with the WR_PARM, WR_DPARM or PARM_MOD instruction and if this slave operates in DPV1 mode, the DP master evaluates the error information it has received from this slave as follows:

If the error information lies within the range from W#16#8000 to W#16#80FF or W#16#F000 to W#16#FFFF, the DP master passes the error information to the instruction. If the error information is outside this range, the DP master passes the value W#16#80A2 to the instruction and suspends the slave.

Displays for the "RD_DPARM (Page 6599)", "WR_PARM (Page 6600)", "WR_DPARM (Page 6601)" and "PARM_MOD" instructions

Parameter RET_VAL

Error code (W#16#...)	Explanation	Restriction
0000	No error	-
7000	First call with REQ=0: No data transfer active; BUSY has the value 0.	-
7001	First call with REQ=1: Data transfer triggered; BUSY has the value "1".	Distributed I/O
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".	Distributed I/O

Error code (W#16#...)	Explanation	Restriction
8090	Specified logical base address invalid: There is no assignment in SDB1/SDB2x or there is no base address.	-
8092	A type other than BYTE is specified in the ANY reference.	Only with S7-400 for RD_DPARM (Page 6599) and WR_PARM (Page 6600)
8093	This instruction is not permitted for the module specified by LADDR and IOID. S7-300 modules for an S7-300, S7-400 modules for an S7-400, S7-DP modules for an S7-300 and S7-400).	-
80A1	Negative acknowledgment when sending the data record to the module (the module was removed or became defective during transfer).	1)
80A2	DP protocol error at layer 2, possibly hardware/interface fault in DP slave.	Distributed I/O 1)
80A3	DP protocol error with user interface/user.	Distributed I/O 1)
80A4	Communication on the communication bus disrupted	Error occurred between the CPU and external DP interface module 1)
80B0	Instruction for module type not possible, or module does not recognize the data record.	1)
80B1	The length of the data record to be sent is incorrect. For RD_DPARM (Page 6599): The length of the destination area spanned by RECORD is too short.	-
80B2	The configured slot is not occupied.	1)
80B3	Actual module type does not match the specified module type in SDB1.	1)
80C1	The data of the previous write job on the module for the same data record have not yet been processed by the module.	1)
80C2	The module is currently processing the maximum possible number of jobs for a CPU.	1)
80C3	The required resources (memory, etc.) are currently occupied.	
80C4	Internal temporary error. Job could not be carried out. Repeat the job. If this error occurs often, check your installation for sources of electrical interference.	1)
80C5	Distributed I/O not available or deactivated.	Distributed I/O 1)
80C6	Data record transfer stopped due to priority class abort (restart or background)	Distributed I/O 1)
80D0	No entry for the module in the corresponding SDB.	-
80D1	The data record number is not configured in the corresponding SDB for the module (data record numbers > 241 are rejected).	-
80D2	The module cannot be assigned parameters according to its type identifier.	-

Error code (W#16#...)	Explanation	Restriction
80D3	The SDB cannot be accessed since it does not exist.	-
80D4	SDB structure error: The SDB internal pointer points to a value outside the SDB.	For S7-300 only
80D5	The data record is static.	With "WR_PARM (Page 6600)" only
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)	-
[1] does not occur in "RD_DPARM"		

RD_DPARM: Read data record from configured system data

Description

You use instruction to read the data record with the number RECNUM of the addressed module from the configured system data. The read data record is entered into the destination area spanned by the RECORD parameter.

Parameter

The following table shows the parameters of the instruction "RD_DPARM":

Parameters	Declaration	Data type	Memory area	Description
IOID	Input	BYTE	I, Q, M, D, L or constant	Address area identifier: • B#16#54 = Peripheral input (PI) • B#16#55 = Peripheral output (PQ) If the module is a mixed module, the area identifier of the lower address must be specified. If the addresses are identical, specify B#16#54.
LADDR	Input	WORD	I, Q, M, D, L or constant	Logical base address of the module. For a mixed module, the lower of the two addresses must be specified.
RECNUM	Input	BYTE	I, Q, M, D, L or constant	Data record number (permitted values: 0 to 240)
RET_VAL	Return	INT	I, Q, M, D, L	Length of the data record read in bytes if the read data record fits in the destination area and no error occurred in the transfer. If an error occurs while the instruction is being executed, the return value contains an error code.
RECORD	Output	ANY	I, Q, M, D, L	Destination area for the data record read. Only the BYTE data type is permitted.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

See also: PARM_MOD: Transfer module data records (Page 6596).

WR_PARM: Write module data record

Description

You use the instruction to transfer the data record RECORD to the addressed module. Parameters transferred to the module do not overwrite parameters created during configuration.

Requirements

The data record to be transferred must not be static. The information concerning which data records of a module are static is provided in the module description.

Parameters

The following table shows the parameters of the instruction "WR_PARM":

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = 1: Write request
IOID	Input	BYTE	I, Q, M, D, L or constant	Address area identifier: • B#16#54 = Peripheral input (PI) • B#16#55 = Peripheral output (PQ) If the module is a mixed module, the area identifier of the lower address must be specified. If the addresses are identical, specify B#16#54.
LADDR	Input	WORD	I, Q, M, D, L or constant	Logical base address of the module. For a mixed module, the lower of the two addresses must be specified.
RECNUM	Input	BYTE	I, Q, M, D, L or constant	Data record number
RECORD	Input	ANY	I, Q, M, D, L	Data record Note: Note that for S7-300 CPUs the parameter RECORD always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and leads to an error message in the user program.

Parameters	Declaration	Data type	Memory area	Description
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The writing process is not yet complete.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RECORD

The data to be transferred are read from the RECORD parameter during the first call. If the transfer of the data record takes longer than the duration of one call, the contents of the RECORD parameter are no longer relevant for the subsequent instruction calls (for the same job).

Parameter RET_VAL

See also: PARM_MOD: Transfer module data records (Page 6596).

Note

For S7-400 only

If the general error W#16#8544 occurs, it only indicates that access to at least one byte of the I/O memory area containing the data record was denied. The data transfer was continued.

WR_DPARM: Transfer data record

Description

You use the instruction to transfer the data record with the number RECNUM from the configuration data to the addressed module. With this instruction, it is irrelevant whether the data record is static or dynamic.

Parameter

The following table shows the parameters of the instruction "WR_DPARM":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = 1: Write request
IOID	Input	BYTE	I, Q, M, D, L or constant	Address area identifier: • B#16#54 = Peripheral input (PI) • B#16#55 = Peripheral output (PQ) If the module is a mixed module, you will have to specify the area identifier of the lower address. If the addresses are identical, you will have to specify B#16#54.
LADDR	Input	WORD	I, Q, M, D, L or constant	Logical base address of the module. For a mixed module, the lower of the two addresses must be specified.
RECNUM	Input	BYTE	I, Q, M, D, L or constant	Data record number
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The writing process is not yet complete.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

See also: PARM_MOD: Transfer module data records (Page 6596).

Interrupts

Time-of-day interrupt

Using the instructions SET_TINT, CAN_TINT, ACT_TINT, QRY_TINT

Definition

A time-of-day interrupt causes the time-triggered call of a time-of-day interrupt OB.

Prerequisites for the call

Before a time-of-day interrupt OB can be called by the operating system, the following conditions must be met:

- The time-of-day interrupt OB parameters must be assigned (start date and time, execution)
  - Per configuration, or
  - With the "SET_TINT (Page 6605)" instruction in the user program.
- The time-of-day interrupt OB must be activated
  - Per configuration, or
  - with the "ACT_TINT (Page 6607)" instruction in the user program.
- The time-of-day interrupt OB must not be deselected during configuration.
- The time-of-day interrupt OB must exist in the CPU.
- If you use the "ACT_TINT (Page 6607)" instruction for setting, and if you have specified the execution of the OB as **once only**, the start date and time must not yet have elapsed; for **periodic execution**, the time-of-day interrupt OB will be started at the next expired period (start time + multiple of the period duration).

Note

You can assign parameters to the time-of-day interrupt using the configuration tool and then activate the interrupt in your user program ("ACT_TINT (Page 6607)").

Purpose of the instructions "SET_TINT", "CAN_TINT", "ACT_TINT" and "QRY_TINT"

You use the instructions to

- set time-of-day interrupts ("SET_TINT (Page 6605)")
- delete time-of-day interrupts ("CAN_TINT (Page 6607)")
- enable time-of-day interrupts ("ACT_TINT (Page 6607)")
- query time-of-day interrupts ("QRY_TINT (Page 6608)")

Effects on the time-of-day interrupt

The following table lists a number of different situations and explains the effect they have on a time-of-day interrupt.

If ...	Then ...
a new time-of-day is set (Calling of "SET_TINT (Page 6605)"),	The current time-of-day interrupt is canceled automatically.
The time-of-day interrupt will be cancelled (Calling of "CAN_TINT (Page 6607)"),	The start date and time are cleared. The time-of-day interrupt must then be set again before it can be activated.

If ...	Then ...
The time-of-day interrupt OB does not exist when it is called,	The priority class error is generated automatically, which means that the operating system calls OB 85. If OB 85 does not exist, the CPU changes to STOP.
The clock is synchronized or the time of day set forward,	If the start date/time is skipped because the clock is moved forward: • The operating system then calls OB 80.[1] • Following OB 80, every skipped time-of-day interrupt OB is called (once, regardless of the number of periods that were skipped) provided that it was not manipulated in OB 80.[2] If OB 80 does not exist, the CPU changes to STOP.
The real-time clock is synchronized or the clock is set back	S7-400 CPUs and CPU 318: If the time-of-day interrupt OBs had already been called during the time by which the clock has been set back, they are not called again the second time around. S7-300 CPUs: The affected time-of-day interrupt OBs are all executed.
[1] OB 80 contains encoded start event information, indicating which time-of-day interrupt OBs could not be called due to moving the clock forward. The time of day in the start event information corresponds to the time adjusted forward. [2] The time of day in the start event information of the time-of-day interrupt OB activated later after being skipped corresponds to the start time of the first skipped time-of-day interrupt.	

Response to warm restart or cold restart

During a warm restart or a cold restart, all the time-of-day interrupt settings made in the user program by means of instructions are cleared. In this case the parameters set per configuration are effective.

Execution of the time-of-day interrupt OB

The following table shows the different effects of the "execution" parameter. This parameter must be set per configuration or with the "SET_TINT (Page 6605)" instruction (input parameter PERIOD).

Execution of the time-of-day interrupt OB	Reaction
None (can only be set per configuration)	The time-of-day interrupt OB is not executed even when it exists in the CPU. Parameter reassignment, which means you can set the time-of-day interrupt ("SET_TINT (Page 6605)") in the user program.
Once	The time-of-day interrupt is canceled after the time-of-day interrupt OB has been called. It can then be set and activated again.
Periodic (every minute, hour, day, week, month, year)	If the start date and time have already passed when the interrupt is activated, the time-of-day interrupt OB interrupts the cyclic program at the next possible point "start date/time + multiple of the selected period". In extremely rare situations, processing of the time-of-day interrupt OB may not yet be completed when it is called again. Result: • Time error, (the operating system calls OB 80; if OB 80 does not exist, the CPU changes to STOP). • The time-of-day interrupt OB is executed later.

SET_TINT: Set time-of-day interrupt

Description

With the instruction, you can set the start date and time of the time-of-day interrupt organization blocks. The seconds and milliseconds of the specified start time are ignored and set to "0".

Parameter

The following table shows the parameters of the instruction "SET_TINT":

Parameters	Declaration	Data type	Memory area	Description
OB_NR	Input	INT	I, Q, M, D, L or constant	Number of the OB started at the time SDT + multiple of PERIOD (OB 10 to OB 17).
SDT	Input	DT	D, L	Start date and time: the seconds and milliseconds of the specified start time are ignored and set to 0. If you want to set a monthly start of a time-of-day interrupt OB, you can only use the days 1, 2, ... 28 as a start date.
PERIOD	Input	WORD	I, Q, M, D, L or constant	Periods from start point SDT onwards: • W#16#0000 = once • W#16#0201 = once every minute • W#16#0401 = once hourly • W#16#1001 = once daily • W#16#1201 = once weekly • W#16#1401 = once monthly • W#16#1801 = once yearly • W#16#2001 = at month's end
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs during execution of the instruction, the actual parameter of RET_VAL contains an error code.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code (W#16#...)	Description
0000	No error occurred.
8090	Incorrect parameter OB_NR
8091	Incorrect parameter SDT
8092	Incorrect parameter PERIOD
80A1	The set start time is in the past. (This error code occurs only when PERIOD = W#16#0000.)
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

CAN_TINT: Cancel time-of-day interrupt

Description

You use the instruction to set the start date and time of the time-of-day interrupt organization block. This disables the time-of-day interrupt and the organization block is no longer called.

If you would you like to use the time-of-day interrupt again, you must first set the start time ("SET_TINT (Page 6605)" instruction) and then enable the time-of-day interrupt ("ACT_TINT (Page 6607)" instruction).

Parameter

The following table shows the parameters of the instruction "CAN_TINT":

Parameters	Declaration	Data type	Memory area	Description
OB_NR	Input	INT	I, Q, M, D, L or constant	Number of the OB, in which the start date and time will be canceled (OB 10 to OB 17).
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs during execution of the instruction, the actual parameter of RET_VAL contains an error code.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code (W#16#...)	Description
0000	No error occurred.
8090	Incorrect parameter OB_NR
80A0	No start date/time specified for the time-of-day interrupt OB
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

ACT_TINT: Enable time-of-day interrupt

Description

You use the instruction to activate a time-of-day interrupt organization block.

Parameter

The following table shows the parameters of the instruction "ACT_TINT":

Parameters	Declaration	Data type	Memory area	Description
OB_NR	Input	INT	I, Q, M, D, L or constant	Number of the OB to be activated (OB 10 to OB 17).
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs during execution of the instruction, the actual parameter of RET_VAL contains an error code.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code (W#16#...)	Description
0000	No error occurred.
8090	Incorrect parameter OB_NR
80A0	Start date/time-of day not set for the respective time-of-day interrupt OB.
80A1	The activated time is in the past. This error only occurs if execution = once is selected.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

QRY_TINT: Query status of time-of-day interrupt

Description

You use the instruction to display the status of a time-of-day interrupt organization block at the output parameter STATUS .

Parameter

The following table shows the parameters of the instruction "QRY_TINT":

Parameters	Declaration	Data type	Memory area	Description
OB_NR	Input	INT	I, Q, M, D, L or constant	Number of the OB that will be queried for status (OB 10 to OB 17).
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs during execution of the instruction, the actual parameter of RET_VAL contains an error code.
STATUS	Output	WORD	I, Q, M, D, L	Status of the time-of-day interrupt; see following table.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter STATUS

Note

If RET_VAL contains a value different than zero, you may not evaluate STATUS.

Bit	Value	Meaning
0	0	In Run.
	1	During startup.
1	0	The time-of-day interrupt is enabled.
	1	The time-of-time interrupt is disabled by "DIS_IRT (Page 6624)".
2	0	Time-of-day interrupt is not activated or has elapsed.
	1	The time-of-day interrupt is activated.
4	0	An OB with an OB number as specified at OB_NR parameter does not exist.
	1	An OB with an OB number as specified at OB_NR parameter does exist.
Other		Always "0"

Parameter RET_VAL

Error code (W#16#...)	Description
0000	No error occurred.
8090	Incorrect parameter OB_NR
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

Time-delay interrupt

Using time-delay interrupts

Definition

After you have called the "SRT_DINT (Page 6611)" instruction, the operating system generates an interrupt after the specified delay time has elapsed, in other words, the assigned time-delay interrupt OB is called.

Prerequisites for the call

Before a time-delay interrupt can be called by the operating system, the following conditions must be met:

- The time-delay interrupt OB must be started by the "SRT_DINT (Page 6611)" instruction.
- The time-delay interrupt OB must not be deselected during configuration.
- The time-delay interrupt OB must exist in the CPU.

Purpose of the instructions "SRT_DINT", "CAN_DINT" and "QRY_DINT"

You use the instructions to

- Start time-delay interrupts ("SRT_DINT (Page 6611)")
- Cancel time-delay interrupts ("CAN_DINT (Page 6612)")
- Query time-delay interrupts ("QRY_DINT (Page 6613)")

Effects on the time-delay interrupt

The following table lists a number of different situations and explains the effect they have on a time-delay interrupt.

If ...	and ...	Then ...
A time-delay interrupt is started (by calling "SRT_DINT (Page 6611)")	The time-delay interrupt has already started,	The delay time is overwritten; the time-delay interrupt is started again.
	The time-delay interrupt OB does not exist at the time of the call,	The operating system generates a priority class error (calls OB 85). If OB 85 does not exist, the CPU changes to STOP.

If ...	and ...	Then ...
	The interrupt is started in a startup OB and the delay time elapses before the CPU changes to RUN,	The call of the time-delay interrupt OB is delayed until the CPU is in RUN mode.
The delay time has elapsed,	A previously started time-delay interrupt OB is still being executed,	The operating system generates a time error (calls OB 80). If OB 80 does not exist, the CPU changes to STOP.

Response to warm restart and cold restart

During a warm restart or a cold restart, all the time-delay interrupt settings made in the user program by means of instructions are cleared.

Starting in a startup OB

A time-delay interrupt can be started in a startup OB. Two conditions must be satisfied to call the time-delay OB:

- The delay time must have elapsed.
- The CPU must be in the RUN mode.

If the delay time has elapsed and the CPU is not yet in the RUN mode, the time-delay interrupt OB call is delayed until the CPU is in RUN mode. The time-delay interrupt OB is then called before the first instruction in OB Main [OB 1] is executed.

SRT_DINT: Start time-delay interrupt

Description

You use the instruction to start a time-delay interrupt that calls a time-delay interrupt OB once the assigned delay time has elapsed (DTIME parameter).

With the SIGN parameter, you can enter an identifier that identifies the start of the time-delay interrupt. The values of DTIME and SIGN appear again in the start event information of the specified OB when it is executed.

The maximum time between the "SRT_DINT" instruction call and the start of the time-delay interrupt OB is **one millisecond** less than the assigned time, provided that no interruption events delay the call.

Parameter

The following table shows the parameters of the instruction "SRT_DINT":

Parameters	Declaration	Data type	Memory area	Description
OB_NR	Input	INT	I, Q, M, D, L or constant	Number of the OB to be started after a time delay (OB 20 to OB 23).
DTIME	Input	TIME	I, Q, M, D, L or constant	Time delay value (1 to 60000 ms) You can realize longer times, for example, by using a counter in a time-delay interrupt OB.
SIGN	Input	WORD	I, Q, M, D, L or constant	Identifier that appears in the start event information of the OB when the time-delay interrupt OB is called.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs during execution of the instruction, the actual parameter of RET_VAL contains an error code.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code (W#16#...)	Description
0000	No error occurred.
8090	Incorrect parameter OB_NR
8091	Incorrect parameter DTIME
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

CAN_DINT: Cancel time-delay interrupt

Description

You use the instruction to cancel a started time-delay interrupt (see "SRT_DINT (Page 6611)"). The time-delay interrupt OB is not called in this case.

Parameters

The following table shows the parameters of the instruction "CAN_DINT":

Parameters	Declaration	Data type	Memory area	Description
OB_NR	Input	INT	I, Q, M, D, L or constant	Number of the OB to be canceled (OB 20 to OB 23).
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs during execution of the instruction, the actual parameter of RET_VAL contains an error code.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code (W#16#...)	Description
0000	No error occurred.
8090	Incorrect parameter OB_NR
80A0	Time-delay interrupt has not started.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

QRY_DINT: Query time-delay interrupt status

Description

You use the instruction to query the status of a time-delay interrupt. Time-delay interrupts are managed by organization blocks OB 20 to OB 23.

Parameter

The following table shows the parameters of the instruction "QRY_DINT":

Parameters	Declaration	Data type	Memory area	Description
OB_NR	Input	INT	I, Q, M, D, L or constant	Number of the OB that will be queried for status (OB 20 to OB 23).
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs during execution of the instruction, the actual parameter of RET_VAL contains an error code.
STATUS	Output	WORD	I, Q, M, D, L	Status of the time-delay interrupt, see following table.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter STATUS

Note

If RET_VAL contains a value different than zero, you may not evaluate STATUS.

Bit	Value	Meaning
0	0	In Run.
	1	During startup.
1	0	The time-delay interrupt is enabled.
	1	The time-delay interrupt is disabled by "DIS_IRT (Page 6624)".
2	0	Time-delay interrupt is not activated or has elapsed.
	1	The time-delay interrupt is activated.
4	0	An OB with an OB number as specified at OB_NR parameter does not exist.
	1	An OB with an OB number as specified at OB_NR parameter is loaded.
Other		Always "0"

Parameter RET_VAL

Error code (W#16#...)	Description
0000	No error occurred.
8090	Incorrect parameter OB_NR
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

Synchronous error events

Mask synchronous error events

Introduction

Synchronous errors are programming and access errors. Such errors occur as a result of programming with incorrect operand areas or numbers, or incorrect addresses. **Masking** these synchronous errors means the following:

- Masked synchronous errors do not trigger an error OB call and do not lead to a programmed alternative reaction.
- The CPU "records" the masked errors that have occurred in an error status register.

Masking is carried out by calling the "MSK_FLT (Page 6620)" instruction.

Unmasking errors means canceling a previously set mask and clearing the corresponding bit in the event status register of the current priority class. Masking is canceled as follows:

- By calling the "DMSK_FLT (Page 6621)" instruction.
- When the current priority class has been completed (for S7-400 only).

If an error event occurs after it has been unmasked, then the operating system will start the associated error OB. You program OB 121 for the reaction to programming errors and OB 122 for the reaction to access errors.

You can use the "READ_ERR (Page 6622)" instruction to read the masked errors that have occurred.

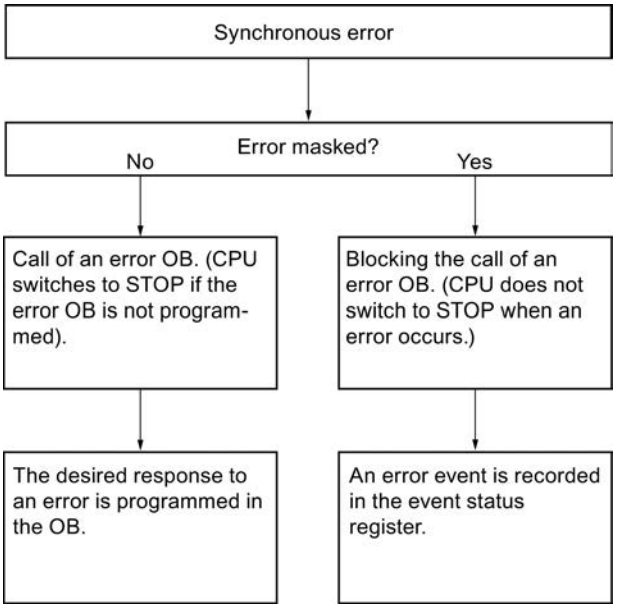
Note

With the S7-300 (except CPU 318), regardless of whether an error is masked or unmasked, the error is entered in the diagnostic buffer and the group error LED of the CPU is illuminated.

Handling errors in general

If programming and access errors occur in a user program, you can react to them in different ways:

- You can program an error OB that is called by the operating system when the corresponding error occurs.
- You can disable the error OB call individually for each priority class. In this case, the CPU does not change to STOP when an error of this type occurs in the particular priority class. The CPU enters the error in an error register. From this entry, however, you cannot recognize when or how often the error occurred.



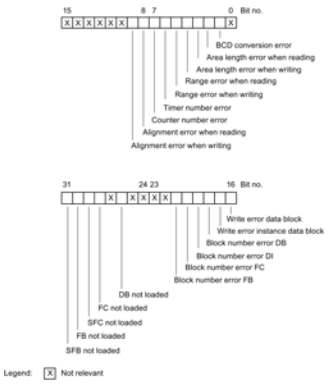
Error filter

Synchronous errors are assigned to a particular bit pattern known as the **error filter**. You can find this error filter in the input and output parameters of the "MSK_FLT (Page 6620)", "DMSK_FLT (Page 6621)", and "READ_ERR (Page 6622)" instructions.

Synchronous errors are divided into programming and access errors that you can mask using two error filters. The error filters are illustrated in the following figures.

Programming error filter

The following figure shows the bit pattern of the error filter for programming errors. The error filter for the programming errors is available in the "PRGFLT_..." parameters (see below "Programming errors, Low-Word" or "Programming errors, High-Word").



Non-relevant bits

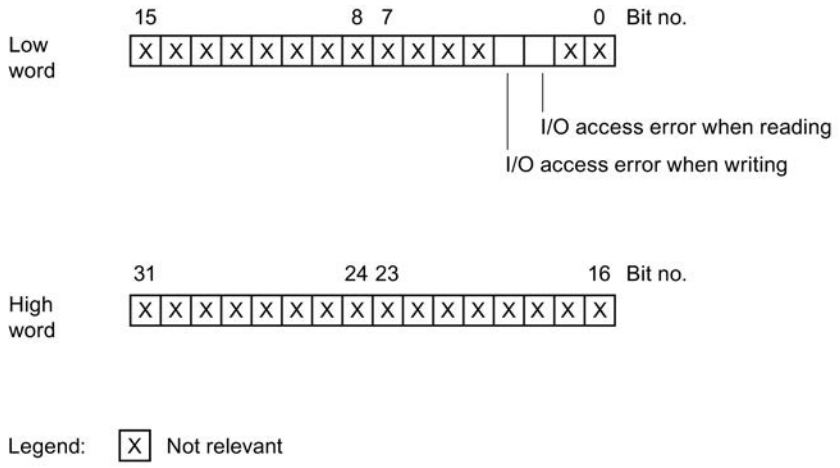
In the figure above, **x** means ...

• ... Input parameters	For "MSK_FLT (Page 6620)", "DMSK_FLT (Page 6621)", "READ_ERR (Page 6622)"	= **"0"**
• ... Output parameters	For "MSK_FLT (Page 6620)", "DMSK_FLT (Page 6621)"	= **"1"** for S7-300 = **"0"** for S7-400
	For "READ_ERR (Page 6622)"	= **"0"**

Access error filter for all CPUs

The following figure shows you the bit pattern of the error filter for access errors. The error mask for access errors is located in the parameters ACCFLT_...

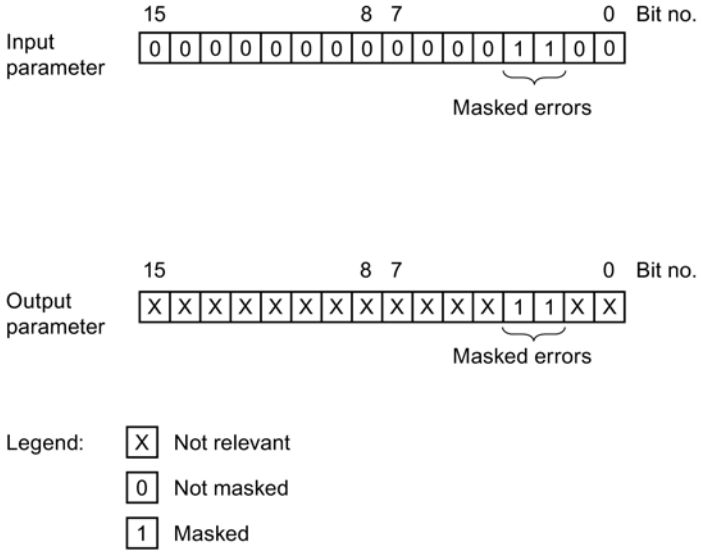
An explanation of the access errors can be found in the following tables (see below).



Example

The following figure shows the low word of the error filter for access errors with all masked errors for all CPUs.

- As input parameters for "MSK_FLT (Page 6620)"
- As output parameters for "MSK_FLT (Page 6620)"



Programming error low word

The following table lists the errors assigned to the low word of the error filter for programming errors. The possible causes of the errors are also listed.

Possible causes of programming errors, low word

Error	Event ID (W#16#...)	Error occurs ...
BCD conversion error	2521	... when the value to be converted is not a BCD number (for example, 5E8).
Area length error when reading	2522	... when an addressed operand is being used that is not completely within the possible operand area. Example: MW 320 should be read although the memory area is only 256 bytes long.
Area length error when writing	2523	... when an addressed operand is being used that is not completely within the possible operand area. Example: A value should be written to MW 320 although the memory area is only 256 bytes long.
Range error when reading	2524	... when an incorrect area identifier is specified for the operand when using indirect, area-overlapping addressing. Example: Correct: LAR1 P#E 12.0 L W[AR1, P#0.0] Incorrect: LAR1 P#12.0 L W[AR1, P#0.0] The range error is reported during this operation.
Range error when writing	2525	... when an incorrect area identifier is specified for the operand when using indirect, area-overlapping addressing. Example: Correct: LAR1 P#E 12.0 T W[AR1, P#0.0] Incorrect: LAR1 P#12.0 T W[AR1, P#0.0] The range error is reported during this operation.
Timer number error	2526	... when a non-existent timer is accessed. Example: SP T [MW 0] where MW 0 = 129; timer 129 should be started although there are only 128 timers available.
Counter number error	2527	... when a non-existent counter is accessed. Example: CU C [MW 0] where MW 0 = 600; counter 600 must be accessed although there are only 512 counters available (CPU 416-1).

Error	Event ID (W#16#...)	Error occurs ...
Alignment error when reading	2528	... when a byte, word, or double word operand is addressed with a bit address ≠ 0. Example: Correct: LAR1 P#M12.0 L B[AR1, P#0.0] Incorrect: LAR1 P#M12.4 L B[AR1, P#0.0]
Alignment error when writing	2529	... when a byte, word, or double word operand is addressed with a bit address ≠ 0. Example: Correct: LAR1 P#M12.0 T B[AR1, P#0.0] Incorrect: LAR1 P#M12.4 T B[AR1, P#0.0]

Programming error high word

The following table lists the errors assigned to the high word of the error filter for programming errors. The possible causes of the errors are also listed.

Possible causes of programming errors, high word

Error	Event ID (W#16#...)	Error occurs ...
Write error data block	2530	... when the data block to be written to is read-only.
Write error instance data block	2531	... when the instance data block to be written to is read-only.
Block number error DB	2532	... when a data block must be opened whose number is higher than the highest permitted number.
Block number error DI	2533	... when an instance data block must be opened whose number is higher than the highest permitted number.
Block number error FC	2534	... when a function is called whose number is higher than the highest permitted number.
Block number error FB	2535	... when a function block is called whose number is higher than the highest permitted number.
DB not loaded	253A	... when the data block to be opened is not loaded.
Instruction not loaded	253C to 253F	... when the called instruction is not loaded.

Access error

The following table lists the errors assigned to the error filter for access errors for all CPUs. The possible causes of the errors are also listed.

Error	Event ID (W#16#...)	Error occurs ...
I/O access error when reading	2942	... when a signal module is not assigned to the address in the I/O area. Or ... when access to this I/O area is not acknowledged within the specified module monitoring time (timeout).
I/O access error when writing	2943	... when a signal module is not assigned to the address in the I/O area. Or ... when access to this I/O area is not acknowledged within the specified module monitoring time (timeout).

MSK_FLT: Mask synchronous error events

Description

You use the instruction to control the reaction of the CPU to synchronous errors. To do this, you mask the respective synchronous errors (for error filters see Mask synchronous error events (Page 6614)). When you call "MSK_FLT", you mask the synchronous errors in the current priority class.

If you set individual bits of the synchronous error filter to "1" in the input parameters, other bits that were set previously retain their value of "1", You obtain new error filters that you can read out using the output parameters. The synchronous errors you have masked do not call an OB but are simply entered in an error register. You can read the error register with the "READ_ERR (Page 6622)" instruction.

Parameter

The following table shows the parameters of the instruction "MSK_FLT":

Parameters	Declaration	Data type	Memory area	Description
PRGFLT_SET_MASK	Input	DWORD	I, Q, M, D, L or constant	Programming error to be masked
ACCFLT_SET_MASK	Input	DWORD	I, Q, M, D, L or constant	Access error to be masked
RET_VAL	Return	INT	I, Q, M, D, L	Error information
PRGFLT_MASKED	Output	DWORD	I, Q, M, D, L	Masked programming errors
ACCFLT_MASKED	Output	DWORD	I, Q, M, D, L	Masked access errors

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code (W#16#...)	Description
0000	None of the errors were already masked.
0001	At least one of the errors was already masked. Nevertheless the other errors will be masked.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

DMSK_FLT: Unmask synchronous error events

Description

You use the instruction to unmask the errors masked with "MSK_FLT (Page 6620)". To do this, you must set the corresponding bits of the error filter to "1" in the input parameters. With the "DMSK_FLT" call, you unmask the corresponding synchronous errors of the current priority class. At the same time, the queried entries are cleared in the error register. You can read out the new error filters using the output parameters.

Parameters

The following table shows the parameters of the instruction "DMSK_FLT":

Parameters	Declaration	Data type	Memory area	Description
PRGFLT_ RESET_MASK	Input	DWORD	I, Q, M, D, L or constant	Programming errors to be unmasked
ACCFLT_ RESET_MASK	Input	DWORD	I, Q, M, D, L or constant	Access errors to be unmasked
RET_VAL	Return	INT	I, Q, M, D, L	Error information
PRGFLT_ MASKED	Output	DWORD	I, Q, M, D, L	Still masked programming errors
ACCFLT_ MASKED	Output	DWORD	I, Q, M, D, L	Still masked access errors

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code (W#16#...)	Description
0000	All specified errors were unmasked.
0001	At least one of the errors was not masked. Nevertheless the other errors will be unmasked.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

READ_ERR: Read out event status register

Description

You use the instruction to read the error register. The structure of the error register corresponds to that of the programming and access error filters, which you can program as input parameters with "MSK_FLT (Page 6620)" and "DMSK_FLT (Page 6621)".

In the input parameters, you enter the synchronous errors you want to read from the error register. When you call "READ_ERR", you read the required entries from the error register and at the same time clear these entries.

The error register contains information that tells you which of the masked synchronous errors in the current priority class occurred at least once. If a bit is set, this means that the corresponding masked synchronous error occurred at least once.

Parameter

The following table shows the parameters of the instruction "READ_ERR":

Parameters	Declaration	Data type	Memory area	Description
PRGFLT_QUERY	Input	DWORD	I, Q, M, D, L or constant	Query programming error
ACCFLT_QUERY	Input	DWORD	I, Q, M, D, L or constant	Query access error
RET_VAL	Return	INT	I, Q, M, D, L	Error information
PRGFLT_CLR	Output	DWORD	I, Q, M, D, L	Programming errors that occurred
ACCFLT_CLR	Output	DWORD	I, Q, M, D, L	Access errors that occurred

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code (W#16#...)	Description
0000	All queried errors are masked.
0001	At least one of the queried errors is not masked.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

Asynchronous error event

Delaying and disabling interrupt and asynchronous error events

Purpose of the instructions "DIS_IRT", "EN_IRT", "DIS_AIRT", "EN_AIRT"

You use these instructions to

- Disable processing of interrupts and asynchronous error events with "DIS_IRT (Page 6624)" for all subsequent CPU cycles.
- Delay higher priority classes with "DIS_AIRT (Page 6624)" until the end of the OB.
- To re-enable processing of interrupts and asynchronous error events with "EN_IRT (Page 6626)" or "EN_AIRT (Page 6628)".

You program the processing of interrupts and asynchronous error events in the user program. You must also program the corresponding OBs.

Purpose of the instructions "DIS_AIRT" and "EN_AIRT"

Delaying higher priority interrupts and asynchronous error events with the "DIS_AIRT (Page 6628)" instruction and then enabling them again with the "EN_AIRT (Page 6628)" instruction has the following advantage:

The number of interrupt delays is counted by the CPU. If you have programmed a delay of interrupts and asynchronous error events, the delay cannot be canceled by standard FC calls if the interrupts and asynchronous error events are also disabled and then enabled again in the standard FCs themselves.

Interrupt classes

The interrupts are divided into various classes. The following table lists all the interrupt classes and the corresponding OBs.

Interrupt class	OB
Time-of-day interrupts	OB 10 to OB 17
Time-delay interrupts	OB 20 to OB 23
Cyclic interrupts	OB 30 to OB 38

Interrupt class	OB
Process interrupts	OB 40 to OB 47
Interrupts for DPV1	OB 55 to OB 57
Multicomputing interrupt	OB 60
Redundancy error interrupts	OB 70, OB 72
Asynchronous error interrupts	OB 80 to OB 87 (se below)
Synchronous error interrupts	OB 121, OB 122 (You mask or unmask the execution of synchronous error interrupts with the "MSK_FLT (Page 6620)", "DMSK_FLT (Page 6621)" and "READ_ERR (Page 6622)" instructions)

Asynchronous error events

The following table lists all the asynchronous error events to which you can react with an OB call in the user program.

Asynchronous error events	OB
Time error (for example, cycle time exceeded)	OB 80
Power supply error (for example, battery fault)	OB 81
Diagnostics interrupt (for example, defective fuse on a signal module)	OB 82
Insert/remove module interrupt	OB 83
CPU hardware fault (for example, interface error)	OB 84
Program execution error	OB 85
Rack failure	OB 86
Communication error	OB 87

DIS_IRT: Disable interrupt event

Description

With the instruction you disable the processing of new interrupts and asynchronous error evetns. Disabling means that if an interrupting event occurs, the operating system of the CPU reacts as follows:

- It **neither** calls an interrupt OB nor an asynchronous error OB,
- **nor** does it trigger the normal reaction if an interrupt OB or asynchronous error OB is not programmed.

If you disable interrupts and asynchronous error events, this remains in effect for all priority classes. The disable can only be canceled with the "EN_IRT (Page 6626)" instruction or by a warm or a cold restart.

Whether the operating system writes interrupts and asynchronous error events to the diagnostics buffer when they occur depends on the input parameter setting you select for MODE.

Note

Remember that when you program the "DIS_IRT" instruction, all interrupts that occur will be discarded.

Parameter

The following table shows the parameters of the instruction "DIS_IRT":

Parameters	Declaration	Data type	Memory area	Description
MODE	Input	BYTE	I, Q, M, D, L or constant	Specifies which interrupts and asynchronous errors are disabled.
OB_NR	Input	INT	I, Q, M, D, L or constant	OB number
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter MODE

MODE (B#16#...)	Meaning
00	All newly occurring interrupts and asynchronous error events are disabled. (Synchronous errors are not disabled.) Assign the value "0" to the OB_NR parameter. Entries continue to be made in the diagnostics buffer.
01	All newly occurring events belonging to a specified interrupt class are disabled. Identify the interrupt class by specifying it as follows: • Time-of-day interrupts: 10 • Time-delay interrupts: 20 • Cyclic interrupts: 30 • Process interrupts: 40 • Interrupts for DPV1: 50 • Multicomputing interrupt: 60 • Redundancy error interrupts: 70 • Asynchronous error interrupts: 80 Entries continue to be made in the diagnostics buffer.
02	All new occurrences of a specified interrupt are disabled. You designate the interrupt using the OB number. Entries continue to be made in the diagnostics buffer.

MODE (B#16#...)	Meaning
80	All newly occurring interrupts and asynchronous error events are disabled and are no longer entered in the diagnostics buffer. Assign the value "0" to the OB_NR parameter. The operating system enters event W#16#5380 in the diagnostics buffer.
81	All newly occurring events belonging to a specified interrupt class are disabled and are no longer entered in the diagnostics buffer. The operating system enters event W#16#5380 in the diagnostics buffer.
82	All newly occurring events belonging to a specified interrupt are disabled and are no longer entered in the diagnostics buffer. The operating system enters event W#16#5380 in the diagnostics buffer.

Parameter RET_VAL

Error code (W#16#...)	Description
0000	No error occurred.
8090	The OB_NR input parameter contains an illegal value.
8091	The MODE input parameter contains an illegal value.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

EN_IRT: Enable interrupt event

Description

You use the instruction to enable the processing of new interrupts and asynchronous error events that you have previously disabled with the "DIS_IRT (Page 6624)" instruction. This means that if an interrupting event occurs, the operating system of the CPU reacts in one of the following ways:

- It calls an interrupt OB or asynchronous error OB.
  Or
- It triggers the standard reaction if the interrupt OB or asynchronous error OB is not programmed.

Parameters

The following table shows the parameters of the instruction "EN_IRT":

Parameters	Declaration	Data type	Memory area	Description
MODE	Input	BYTE	I, Q, M, D, L or constant	Specifies which interrupts and asynchronous error events will be enabled.
OB_NR	Input	INT	I, Q, M, D, L or constant	OB number
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter MODE

MODE	Meaning
0	All newly occurring interrupts and asynchronous error events are enabled.
1	All newly occurring events belonging to a specified interrupt class are enabled. Identify the interrupt class by specifying it as follows: • Time-of-day interrupts: 10 • Time-delay interrupts: 20 • Cyclic interrupts: 30 • Process interrupts: 40 • Interrupts for DPV1: 50 • Multicomputing interrupt: 60 • Redundancy error interrupts: 70 • Asynchronous error interrupts: 80
2	All newly occurring events of a specified interrupt are enabled. You designate the interrupt using the OB number.

Parameter RET_VAL

Error code (W#16#...)	Description
0000	No error occurred.
8090	The OB_NR input parameter contains an illegal value.
8091	The MODE input parameter contains an illegal value.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

DIS_AIRT: Delay execution of higher priority interrupts and asynchronous error events

Description

With the instruction you delay the processing of interrupt OBs and asynchronous OBs with priority higher than that of the current OB. You can call the "DIS_AIRT" instruction more than once in an OB. The instruction calls are counted by the operating system. The processing delay is in force until you use the "EN_AIRT (Page 6628)" instruction to cancel each delayed processing of interrupt OBs and asynchronous error OBs with "DIS_AIRT", **or** the current OB has been executed.

The pending interrupts or asynchronous error events are processed as soon as the processing delay has been canceled with the "EN_AIRT (Page 6628)" instruction or the current OB processing is complete.

Parameter

The following table shows the parameters of the instruction "DIS_AIRT":

Parameters	Declaration	Data type	Memory area	Description
RET_VAL	Return	INT	I, Q, M, D, L	Number of delays (= number of DIS_AIRT calls)

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

The following table shows the return value of "DIS_AIRT" that is output with the RET_VAL parameter.

Return value	Description
n	"n" shows the number of processing delays, which is the number of "DIS_AIRT calls, after the instruction is complete (interrupt processing is only enabled again when n = 0; see "EN_AIRT (Page 6628)").

EN_AIRT: Enable execution of higher priority interrupts and asynchronous error events

Description

With the instruction you re-release the execution of high-priority interrupt events or asynchronous error events delayed with "DIS_AIRT (Page 6628)". Each individual processing delay must be canceled with the "EN_AIRT" instruction.

Example

If, for example, you have delayed interrupts five times with five "DIS_AIRT (Page 6628)" calls, you must cancel each of the interrupt delays using five "EN_AIRT" calls.

Parameter

The following table shows the parameters of the instruction "EN_AIRT":

Parameters	Declaration	Data type	Memory area	Description
RET_VAL	Return	INT	I, Q, M, D, L	Number of delays still programmed on completion of "EN_AIRT" or error message.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

How you evaluate the error information of the RET_VAL parameter is explained in the chapter Evaluating errors with output parameter RET_VAL (Page 5370). This chapter also contains the general error information for the instructions. The following table contains the return value or error information specific to "EN_AIRT" that can be output with the RET_VAL parameter.

Return value and error information	Description
n	"n" shows the number of processing delays, which is the number of "EN_AIRT calls, after the instruction is complete (interrupt processing is only enabled again when n = 0).
W#16#8080	"EN_AIRT" was called even though interrupt processing has already been enabled.

Multicomputing interrupt

MP_ALM: Multicomputing interrupt

Description

Calling the instruction during multiprocessing will trigger the multicomputing interrupt. This leads to a synchronized start of OB 60 on all CPUs involved. In single processor mode and when operating with a segmented rack, OB 60 is only started on the CPU on which you called MP_ALM.

Use the input parameter JOB to identify the cause of the multicomputing interrupt. This job identifier is transferred to all associated CPUs, where you can evaluate it in the multicomputing interrupt OB (OB 60).

You can call the instruction at any point in your program. Since the call would be pointless in any mode other than RUN, if it is called in STARTUP mode, the multicomputing interrupt is suppressed. The function value informs you of this.

Parameters

The following table shows the parameters of the instruction "MP_ALM":

Parameters	Declaration	Data type	Memory area	Description
JOB	Input	BYTE	I, Q, M, D, L or constant	Job identifier Possible values: 1 to 15
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error occurred.
8090	The JOB input parameter contains an illegal value.
80A0	Execution of OB 60 following the last multicomputing interrupt is not completed either on the local CPU or on another CPU.
80A1	Incorrect mode (STARTUP instead of RUN).
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

Alarms

Structure of diagnostic data

Data record 0 and 1 of the system data

The diagnostic data of a module are located in the data records 0 and 1 of the system data area, see Writing and reading data records (Page 6591):

- Data record 0 contains 4 bytes of diagnostic data that describe the current status of a signal module.
- Data record 1 contains
  - The 4 bytes of diagnostic data, also located in data record 0 and
  - The diagnostic data specific to the module.

Structure and contents of the diagnostic data:

This section below describes the structure and contents of the individual bytes of the diagnostic data. The general principle: Whenever an error occurs, the corresponding bit is set to "1."

Byte	Bit	Meaning	Remark	Data record
0	0	Module is defective		0 and 1
	1	Internal error		
	2	External error		
	3	Channel fault		
	4	External voltage failed		
	5	Front panel connector not plugged in		
	6	No parameter assignment		
	7	Wrong parameters in the module		
1	0 to 3	Module class	0101: Analog module 0000: CPU 1000: Function module 1100: CP 1111: Digital module 0011: DP standard slave 1011: I-slave 0100: IM	0 and 1
	4	Channel information exists		
	5	User information exists		
	6	Diagnostic interrupt from substitute		
	7	Maintenance requirement (PROFINET IO only)		
2	0	Memory card missing or wrong		0 and 1
	1	Communication problem		

Byte	Bit	Meaning	Remark	Data record
	2	Operating mode	0: RUN 1: STOP	
	3	Cycle monitoring tripped		
	4	Internal power supply failed		
	5	Battery exhausted		
	6	Entire backup failed		
	7	Maintenance request (PROFINET IO only)		
3	0	Expansion rack failure		0 and 1
	1	Processor failure		
	2	EPROM fault		
	3	RAM fault		
	4	ADC/DAC error		
	5	Fuse tripped		
	6	Hardware interrupt lost		
	7	Reserved		
4	0 to 6	Channel type	B#16#70: Digital input B#16#72: Digital output B#16#71: Analog input B#16#73: Analog output B#16#74: FM-POS B#16#75: FM-REG B#16#76: FM-COUNT B#16#77: FM-TECHNO B#16#78: FM-NCU B#16#79: To B#16#7D: Reserved B#16#7E: US300 B#16#7F: Reserved	1
	7	Additional channel type exists?	0: No 1: Yes	
5	0 to 7	Number of diagnostic bits output per channel by a module.	The number of diagnostic bits per channel is rounded up to byte boundaries	1
6	0 to 7	Number of same type channels on a module	If different channel types exist on a module, the structure is repeated in data record 1 from byte 4 onwards for each channel type.	1
7	0	Channel error channel 0/ channel group 0	First byte of the channel error vector (the length of the channel error vector depends on the number of channels and is rounded up to a byte boundary).	1
	1	Channel error channel 1/ channel group 1		
	2	Channel error channel 2/ channel group 2		
	3	Channel error channel 3/ channel group 3		
	4	Channel error channel 4/ channel group 4		
	5	Channel error channel 5/ channel group 5		
	6	Channel error channel 6/ channel group 6		

Byte	Bit	Meaning	Remark	Data record
	7	Channel error channel 7/ channel group 7		
...	-	Channel-specific errors (see Structure of channel-specific diagnostic data (Page 6633))		1

Structure of channel-specific diagnostic data

Channel-specific errors

Starting at the byte immediately following the channel error vector, the channel-specific errors are indicated for each channel of the module. The tables below show the structure of channel-specific diagnostics data for the different channel types. The bits have the following meaning:

- 1 = Error
- 0 = No error

Analog input channel

Diagnostics byte for an analog input channel:

Bit	Meaning	Remark
0	Configuration/parameter assignment error	Can be signaled with WR_USMSG (Page 6665) and EVENTN= W#16#8x50
1	Common mode error	Can be signaled with WR_USMSG (Page 6665) and EVENTN = W#16#8x51
2	P short circuit	Can be signaled with WR_USMSG (Page 6665) and EVENTN = W#16#8x52
3	M short circuit	Can be signaled with WR_USMSG (Page 6665) and EVENTN = W#16#8x53
4	Wire break	Can be signaled with WR_USMSG (Page 6665) and EVENTN = W#16#8x54
5	Reference channel error	Can be signaled with WR_USMSG (Page 6665) and EVENTN = W#16#8x55
6	Measuring range undershoot	Can be signaled with WR_USMSG (Page 6665) and EVENTN = W#16#8x56
7	Measuring range exceeded	Can be signaled with WR_USMSG (Page 6665) and EVENTN = W#16#8x57

Analog output channel

Diagnostics byte for an analog output channel

Bit	Meaning	Remark
0	Configuration/parameter assignment error	Can be signaled with WR_USMSG (Page 6665) and EVENTN = W#16#8x60
1	Common mode error	Can be signaled with WR_USMSG (Page 6665) and EVENTN = W#16#8x61
2	P short circuit	Can be signaled with WR_USMSG (Page 6665) and EVENTN = W#16#8x62
3	M short circuit	Can be signaled with WR_USMSG (Page 6665) and EVENTN = W#16#8x63
4	Wire break	Can be signaled with WR_USMSG (Page 6665) and EVENTN = W#16#8x64
5	0	Reserved
6	No load voltage	Can be signaled with WR_USMSG (Page 6665) and EVENTN = W#16#8x66
7	0	Reserved

Digital input channel

diagnostics byte for a digital input channel:

Bit	Meaning	Remark
0	Configuration/parameter assignment error	Can be signaled with WR_USMSG (Page 6665) and EVENTN = W#16#8x70
1	Ground error	Can be signaled with WR_USMSG (Page 6665) and EVENTN = W#16#8x71
2	I/O short circuit (sensor)	Can be signaled with WR_USMSG (Page 6665) and EVENTN = W#16#8x72
3	M short circuit	Can be signaled with WR_USMSG (Page 6665) and EVENTN = W#16#8x73
4	Wire break	Can be signaled with WR_USMSG (Page 6665) and EVENTN = W#16#8x74
5	No sensor power supply	Can be signaled with WR_USMSG (Page 6665) and EVENTN = W#16#8x75
6	0	Reserved
7	0	Reserved

Digital output channel

Diagnostics byte for a digital output channel:

Bit	Meaning	Remark
0	Configuration/parameter assignment error	Can be signaled with WR_USMSG (Page 6665) and EVENTN = W#16#8x80
1	Ground error	Can be signaled with WR_USMSG (Page 6665) and EVENTN = W#16#8x81
2	P short circuit	Can be signaled with WR_USMSG (Page 6665) and EVENTN = W#16#8x82
3	M short circuit	Can be signaled with WR_USMSG (Page 6665) and EVENTN = W#16#8x83
4	Wire break	Can be signaled with WR_USMSG (Page 6665) and EVENTN = W#16#8x84
5	Fuse tripped	Can be signaled with WR_USMSG (Page 6665) and EVENTN = W#16#8x85
6	No load voltage	Can be signaled with WR_USMSG (Page 6665) and EVENTN = W#16#8x86
7	Overtemperature	Can be signaled with WR_USMSG (Page 6665) and EVENTN = W#16#8x87

Instructions for generating PLC alarms with instance DB

Introduction to generating PLC alarms with instructions

Instructions for generating block-related alarms

You generate a PLC alarm by calling one of the following instructions in your program:

- "NOTIFY (Page 6646)"
- "NOTIFY_8P (Page 6639)"
- "ALARM (Page 6649)"
- "ALARM_8 (Page 6641)"
- "ALARM_8P (Page 6643)"

These instructions have the following properties:

- With "NOTIFY (Page 6646)" and "NOTIFY_8P (Page 6639)", all 0 -> 1 or 1 -> 0 signal changes detected during the call will cause an alarm to be sent.
- Also for "ALARM (Page 6649)", "ALARM_8 (Page 6641)" and "ALARM_8P (Page 6643)", every signal change detected for a call causes an alarm to be sent, provided the alarm process is set to default (acknowledgement-triggered reporting inactive).
  If, on the other hand, you have activated the acknowledgement-triggered reporting, then not every detected signal change will cause an alarm to be sent (see below for more information).

- After the block processing is complete, all of the associated values (SD_i inputs) are detected and the alarm is assigned (refer to "Send and receive parameters" in Common parameters of instructions for S7 communication (Page 7359).)
  The following applies with regard to the consistency of the associated values compared to higher priority classes: Every associated value SD_i is inherently consistent.
- With status parameters DONE, ERROR and STATUS, you monitor the processing status of the block (refer to "Status parameters" in Common parameters of instructions for S7 communication (Page 7359)).

Note

The parameters ID and EV_ID are only evaluated at the initial call of the block (the actual parameters or the defined values of the instance).

Logging on display devices

Before the instructions for generating PLC alarms can send an alarm when a signal change is detected, at least one display device must be logged on for PLC alarms. STATUS parameter = "1" if no login exists.

Signal change detection

One alarm memory with two memory locations is available for each instance of an alarm instruction.
This alarm memory is initially empty. As soon as the instruction detects a signal change at the SIG input or at one of the inputs SIG_1, ... SIG_8, it is entered in the first memory location. It remains occupied until the associated alarm is sent.
The next detected signal at the SIG or at one of the SIG_1, ... SIG_8 inputs is then entered in the second memory block. If the first memory block is still occupied and additional signals follow, then the second memory block in the alarm memory is always overwritten.
The loss of this alarm is indicated via the output parameters ERROR and STATUS (ERROR=0, STATUS=11). In addition the logged on display devices get an alarm in this regard with the next alarm that can be sent.
If he first memory block is free then the second memory block will be transferred into the first. Thus, the second memory block is cleared again.

Acknowledgement-triggered reporting

You can reduce alarm traffic on your system by using acknowledgement-triggered alarm generation with the "ALARM (Page 6649)", "ALARM_8 (Page 6641)" and "ALARM_8P (Page 6643)" instructions.

That is, after an incoming alarm has been generated (Signal transition 0 > 1) initially for a signal, subsequent alarms will not be generated until you have acknowledged the first alarm on a display device. The next alarm displayed on the display unit after your acknowledgement is the outgoing alarm (signal transition 1 to 0). Subsequently the alarm cycle restarts with an incoming alarm (signal change from 0 to 1) that must be acknowledged. In this manner you can control the reports of signal change (until the outgoing alarm) via the display device.

You specify the alarm generation method (enable or disable acknowledgement-triggered alarm generation) for the "ALARM (Page 6649)", "ALARM_8 (Page 6641)" and "ALARM_8P

(Page 6643)" instructions globally for the CPU during configuration. Acknowledgement-triggered reporting is disabled by default.

To ensure consistent alarm evaluation within your system, you should verify that all display devices can handle acknowledgement-triggered reporting.

Note

(Note on display devices that cannot handle acknowledgement-triggered reporting.)

A CPU with enabled acknowledgement-triggered reporting will distribute the alarms only to display devices capable of handling this reporting method. The CPU will not send any alarms if none of the display devices can handle acknowledgement triggered reporting. This situation is indicated once with ERROR=1 and STATUS=1.

Alarm acknowledgement for the instructions "ALARM", "ALARM_8" and "ALARM_8P"

A centralized acknowledgement concept is used. When you have acknowledged the alarm at a display device, the acknowledgement information is first sent to the CPU that generated the alarm. This then distributes the acknowledgement information to all stations logged on for this purpose.

You always acknowledge a signal and not an individual alarm. If, for example, several rising edges of a signal were indicated and you acknowledge the event entering the state, all previous events with the same alarm number count as having been acknowledged.

Acknowledgement display

The "NOTIFY (Page 6646)" and "NOTIFY_8P (Page 6639)" instructions have no acknowledgement display. With "ALARM (Page 6649)", you can obtain the acknowledgement status from the ACK_UP and ACK_DN output parameters. With "ALARM_8P (Page 6643)" and "ALARM_8 (Page 6641)", the acknowledgement status is available in the ACK_STATE output parameter. These outputs are updated when the block is called, providing the control parameter EN_R has the value "1".

Disabling and enabling alarms with the instruction or display device

In some situations, it may be useful to suppress alarms (e.g. when you reconfigure your system). You can therefore disable and enable alarms at the display device or in your program. Disabling/enabling applies to all stations that are logged on for the particular alarm. A disabled alarm remains disabled until it is enabled again.

You are informed of disabled alarms with the ERROR and STATUS output parameters (ERROR = 1, STATUS = 21).

Work memory required by the instructions for generating block-related alarms

For their proper operation, the instructions for generating PLC alarms require a communications data buffer in the work memory of the CPU (code area) that is generally

dependent on the associated value. Refer to the table below for information on the size of used memory.

Instruction	Required space (in bytes) in the work memory of the CPU
NOTIFY	200 + 2 * length of the associated values specified at SD_1,...SD_10 at the first call
NOTIFY_8P	200 + 2 * length of the associated values specified at SD_1,...SD_10 at the first call
ALARM	200 + 2 * length of the associated values specified at SD_1,...SD_10 at the first call
ALARM_8	100
ALARM_8P	200 + 2 * length of the associated values specified at SD_1,...SD_10 at the first call
AR_SEND	54

Amount of transferable data

The amount of data that can be transferred using the associated values SD_i of the "NOTIFY (Page 6646)", "NOTIFY_8P (Page 6639)", "ALARM (Page 6649)" and "ALARM_8P (Page 6643)" instructions must not exceed a maximum length. This maximum data length is calculated as follows:

maxleng =

min (pdu_local, pdu_remote) - diff - 4 * number of SD_i parameters used

Where:

- min (pdu_local, pdu_remote) is the lowest value of the numbers pdu_lokal and pdu_remote
- pdu_local is the maximum length of the fields of the local CPU (see the technical data of your CPU)
- pdu_remote: the maximum length for display device data blocks
- diff = 48, if acknowledgement-triggered reporting is enabled, and diff = 44 if disabled.

Example

A CPU 414-2 is sending alarms via Industrial Ethernet to HMI. Acknowledgement-triggered reporting is disabled.

The associated values SD_1, SD_2 and SD_3 are used.

pdu_local = 480 bytes, pdu_remote = 480 bytes,

Number of SD_i parameters used: 3

Result:

maxleng = min (480, 480) - 44 - 4 * 3 = 480 - 44 - 12 = 424

The maximum length of data that can be transferred per instruction is 424 bytes.

NOTIFY_8P: Report up to eight signal changes

Description

The "NOTIFY_8P" instruction extends the "NOTIFY (Page 6646)" instruction to include eight signals.

An alarm is generated if at least one signal transition has been detected. An alarm is always generated at the first call of "NOTIFY_8P". All eight signals have a common alarm ID that is split into eight individual alarms on the display device.

One alarm memory with two memory locations is available for each instance of "NOTIFY_8P". For more information on the caching of signal changes, refer to Introduction to generating PLC alarms with instructions (Page 6635).

Note

The display device shows the last two signal transitions, irrespective of alarm loss.

WARNING

Calling "NOTIFY_8P"

Before you call "NOTIFY_8P" in an automation system, you must ensure that all connected display devices are familiar with this block. If you fall to heed this instruction the communication between the automation system and the connected display devices will be aborted. In this case you can no longer access your system with its connected display devices.

Parameters

The following table shows the parameters of the instruction "NOTIFY_8P":

Parameters	Declaration	Data type	Memory area	Description
SIG_i, 1 ≤ i ≤ 8	Input	BOOL	I, Q, M, D, L	i-(nth) signal to be monitored
ID	Input	WORD	I, Q, M, D, L or constant	Data channel for alarms: W#16#EEEE ID is only evaluated at the first call.
EV_ID	Input	C_NOTIFY_8P	I, Q, M, D, L	Alarm number (not allowed: 0) EV_ID is only evaluated at the first call. Subsequently, the alarm number used for the first call applies to every call of "NOTIFY_8P" with the corresponding instance DB. The alarm number is assigned automatically. This ensures the consistency of the alarm numbers. The alarm number must be unique within your user program.

Parameters	Declaration	Data type	Memory area	Description
SEVERITY	Input	WORD	I, Q, M, D, L or constant	Weighting of the event Possible values: 0 to 127 (value 0 means highest weighting); default value: 64 This parameter is irrelevant for processing the alarm.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: Generation of alarm is complete.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: ERROR=TRUE indicates that an error has occurred during processing. For details refer to parameter STATUS.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter STATUS: Displays error information
SD_i, 1 ≤ i ≤ 10	InOut	ANY	I, Q, M, D, T, C	i-th associated value Only the BOOL data types are permitted (not permitted: bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME **Note:** When the ANY pointer accesses a DB, the DB must always be specified. (e.g.: P# DB10.DBX5.0 Byte 10)

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

ERROR and STATUS parameters

The following table contains all specific error information for "NOTIFY_8P" that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Alarm loss: The previous signal change or the previous alarm could not be sent and is replaced by the current alarm.
0	22	• Error in the pointer to the associated values SD_i: – relating to the data length or the data type – No access to associated values in user memory, for example, due to deleted DB or area length error – The activated alarm is transferred without or, if required, with the maximum possible number of associated values. • The actual parameter you have selected for SEVERITY is higher than the permitted range. The activated alarm is sent with SEVERITY=127.
0	25	Communication has started. The alarm is being processed.
1	1	Communication problems: connection termination or no logon

ERROR	STATUS (decimal)	Explanation
1	4	At the first call: • The specified EV_ID is outside the permitted range or • the ANY pointer SD_i has a formal error. • The maximum memory area that can be sent for the CPU per "NOTIFY_8P" has been exceeded.
1	10	Access to local user memory not possible (for example, access to a deleted DB).
1	12	When the instruction was called • An instance DB was specified that does not belong to "NOTIFY_8P" • A global DB was specified instead of an instance DB.
1	18	EV_ID was already used by the "NOTIFY (Page 6646)", "NOTIFY_8P", "ALARM (Page 6649)", "ALARM_8 (Page 6641)" or "ALARM_8P (Page 6643)" instructions.
1	20	Not enough work memory.
1	21	The alarm with the specified EV_ID is disabled.

See also

Creating program alarms (Page 3054)

ALARM_8: Create PLC alarms without associated values for eight signals

Description

Except for the non-existing, associated values SD_1, ..., SD_10, the instruction is identical to the "NOTIFY_8P (Page 6639)" instruction.

Parameters

The following table shows the parameters of the "ALARM_8" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L	Control parameter enabled to receive determines whether the output ACK_STATE is updated (EN_R=1) or not (EN_R=0) when the block is called.
SIG_i, 1≤i ≤8	Input	BOOL	I, Q, M, D, L	i-(nth) signal to be monitored
ID	Input	WORD	I, Q, M, D, L or constant	Data channel for alarms: W#16#EEEE ID is only evaluated at the first call.

Parameter	Declaration	Data type	Memory area	Description
EV_ID	Input	C_ALARM_8	I, Q, M, D, L	Alarm number (not allowed: 0) EV_ID is only evaluated at the first call. Subsequently, the alarm number used for the first call applies to every call of "ALARM_8" with the corresponding instance DB. The alarm number is assigned automatically. This ensures the consistency of the alarm numbers. The alarm number must be unique within your user program.
SEVERITY	Input	WORD	I, Q, M, D, L or constant	Weighting of the event Possible values: 0 to 127 (value 0 means highest weighting) This parameter is irrelevant for processing this alarm.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: Generation of alarm is complete.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: ERROR=TRUE indicates that an error has occurred during processing. For details refer to parameter STATUS.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter STATUS: Displays error information
ACK_STATE	Output	WORD	I, Q, M, D, L	Bit array with the current acknowledgement status of all eight alarms (1: Event acknowledged, 0: Event not acknowledged): • Bits 0 to 7 are mapped to the incoming event of SIG_1 to SIG_8 • Bits 8 to 15 are mapped to the outgoing event of SIG_1 to SIG_8 • Initialization status: W#16#FFFF, that is, all incoming and outgoing events have been acknowledged

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

ERROR and STATUS parameters

The following table contains all specific error information for the instruction "ALARM_8" that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Alarm loss: The previous signal change or the previous alarm could not be sent and is replaced by the current alarm.
0	22	The actual parameter you have selected for SEVERITY is higher than the permitted range. The activated alarm is sent with SEVERITY=127.
0	25	Communication has started. The alarm is being processed.

ERROR	STATUS (decimal)	Explanation
1	1	Communication problems: connection termination or no logon With acknowledgement-triggered reporting active: Temporary display, if display devices do not support acknowledgement-triggered reporting
1	4	At the first call, the specified EV_ID is outside the permitted range.
1	10	Access to local user memory not possible (for example, access to a deleted DB).
1	12	When the instruction was called • An instance DB was specified that does not belong to "ALARM_8" • A global DB was specified instead of an instance DB.
1	18	EV_ID was already used by the "NOTIFY (Page 6646)", "NOTIFY_8P (Page 6639)", "ALARM (Page 6649)", "ALARM_8" or "ALARM_8P (Page 6643)" instructions.
1	20	Not enough work memory.
1	21	The alarm with the specified EV_ID is disabled.

Note

After the first call, all the bits of the ACK_STATE output are set and it is assumed that the previous values of SIG_i, 1 ≤ i ≤ 8 inputs were 0.

See also

Creating program alarms (Page 3054)

ALARM_8P: Create PLC alarms without associated values for eight signals

Description

The instruction is the extension of "ALARM (Page 6649)" to include eight signals.

As long as you have not enabled acknowledgement-triggered reporting, an alarm will always be generated when a signal transition is detected at one or more signals (exception: a alarm is always sent at the first block call). All eight signals have a common alarm ID that is split into eight separate alarms on the display device. You can acknowledge each individual alarm separately or a group of alarms at once.

Use the output parameter ACK_STATE to further process the acknowledgement status of the specific alarms in your program. If you enable or release an alarm of an "ALARM_8P" instruction, this action always affects the entire "ALARM_8P" instruction. Disabling and enabling of individual signals is not possible.

One alarm memory with two memory locations is available for each instance of "ALARM_8P". For more information on the caching of signal changes, refer to Introduction to generating PLC alarms with instructions (Page 6635).

Parameters

The following table shows the parameters of the "ALARM_8P" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L	Control parameter enabled to receive determines whether the output ACK_STATE is updated (EN_R=1) or not (EN_R=0) when the block is called.
SIG_i, 1≤i ≤8	Input	BOOL	I, Q, M, D, L	i-(nth) signal to be monitored
ID	Input	WORD	I, Q, M, D, L or constant	Data channel for alarms: W#16#EEEE ID is only evaluated at the first call.
EV_ID	Input	C_ALARM_8P	I, Q, M, D, L	Alarm number (not allowed: 0); EV_ID is only evaluated at the first call. Subsequently, the alarm number used for the first call applies to every call of "ALARM_8P" with the corresponding instance DB. The alarm number is assigned automatically. This ensures the consistency of the alarm numbers. The alarm number must be unique within your user program.
SEVERITY	Input	WORD	I, Q, M, D, L or constant	Weighting of the event Possible values: 0 to 127 (value 0 means highest weighting) This parameter is irrelevant for processing this alarm.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: Generation of alarm is complete.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR ERROR=TRUE indicates that an error has occurred during processing. For details refer to parameter STATUS.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter STATUS: Displays error information

Parameter	Declaration	Data type	Memory area	Description
ACK_STATE	Output	WORD	I, Q, M, D, L	Bit array with the current acknowledgement status of all eight alarms (1: Event acknowledged, 0: Event not acknowledged): • Bits 0 to 7 are mapped to the incoming event of SIG_1 to SIG_7 • Bits 8 to 15 are mapped to the outgoing event of SIG_1 to SIG_7 • Initialization status: W#16#FFFF, that is, all incoming and outgoing events have been acknowledged
SD_j, 1≤ j ≤10	InOut	ANY	I, Q, M, D, T, C	j-th associated value The associated values apply for all alarms. Only the BOOL data types are permitted (not permitted: bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME. **Note:** When the ANY pointer accesses a DB, the DB must always be specified (for example: P# DB10.DBX5.0 byte 10).

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

ERROR and STATUS parameters

The following table contains all specific error information for the "ALARM_8P" instruction that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Alarm loss: The previous signal change or the previous alarm could not be sent and is replaced by the current alarm.
0	22	• Error in the pointer to the associated values SD_i: – relating to the data length or the data type – No access to associated values in user memory, for example, due to deleted DB or area length error – The activated alarm is sent without associated values • The actual parameter you have selected for SEVERITY is higher than the permitted range. The activated alarm is sent with SEVERITY=127 .
0	25	Communication has started. The alarm is being processed.
1	1	Communication problems: Connection termination or no logon. With acknowledgement-triggered reporting active: Temporary display, if display devices do not support acknowledgement-triggered reporting.
1	4	At the first call: • The specified EV_ID is outside the permitted range or • the ANY pointer SD_i has a formal error. • The maximum memory area that can be sent for the CPU per "ALARM_8P" has been exceeded.

ERROR	STATUS (decimal)	Explanation
1	10	Access to local user memory not possible (for example, access to a deleted DB).
1	12	When the instruction was called: • An instance DB was specified that does not belong to "ALARM_8P" • A global DB was specified instead of an instance DB.
1	18	EV_ID was already used by the "NOTIFY (Page 6646)", "NOTIFY_8P (Page 6639)", "ALARM (Page 6649)", "ALARM_8 (Page 6641)" or "ALARM_8P" instructions.
1	20	Not enough work memory.
1	21	The alarm with the specified EV_ID is disabled.

Note

After the first call, all the bits of the ACK_STATE output are set and it is assumed that the previous values of SIG_i, 1< i < 8 inputs were "0".

See also

Creating program alarms (Page 3054)

NOTIFY: Report a signal change

Description

The instruction monitors a signal. It generates an alarm both on a rising edge (incoming event) and on a falling edge (outgoing event). You can have up to ten associated values sent with the alarm. The alarm is sent to all stations logged on for this purpose. When the SFB is first called, an alarm with the current signal state is sent.

The associated values are queried when the edge is detected and assigned to the alarm.

One alarm memory with two memory locations is available for each instance of the instruction. For more information on the caching of signal changes, refer to Introduction to generating PLC alarms with instructions (Page 6635).

The "NOTIFY" instruction conforms to IEC 1131-5.

Parameter

The following table shows the parameters of the instruction "NOTIFY":

Parameters	Declaration	Data type	Memory area	Description
SIG	Input	BOOL	I, Q, M, D, L	The signal to be monitored
ID	Input	WORD	I, Q, M, D, L or constant	Data channel for alarms: W#16#EEEE ID is only evaluated at the first call.

Parameters	Declaration	Data type	Memory area	Description
EV_ID	Input	C_NOTIFY	I, Q, M, D, L	Alarm number (not allowed: 0) EV_ID is only evaluated at the first call. Subsequently, the alarm number used for the first call applies to every call of "NOTIFY" with the corresponding instance DB. The alarm number is assigned automatically. This ensures the consistency of the alarm numbers. The alarm number must be unique within your user program.
SEVERITY	Input	WORD	I, Q, M, D, L or constant	Weighting of the event Possible values: 0 to 127 (value 0 means highest weighting) This parameter is irrelevant for processing this alarm.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: Generation of alarm is complete.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR ERROR=TRUE indicates that an error has occurred during processing. For details refer to parameter STATUS.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter STATUS: Displays error information
SD_i, 1≤i ≤10	InOut	ANY	I, Q, M, D, T, C	i-th associated value Only the BOOL data types are permitted (not permitted: bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME. **Note:** When the ANY pointer accesses a DB, the DB must always be specified. (e.g.: P# DB10.DBX5.0 Byte 10)

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

ERROR and STATUS parameters

The following table contains all specific error information that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Alarm loss: The previous signal change or the previous alarm could not be sent and is replaced by the current alarm.
0	22	• Error in the pointer to the associated values SD_i: – relating to the data length or the data type – No access to associated values in user memory, for example, due to deleted DB or area length error – The activated alarm is transferred without or, if required, with the maximum possible number of associated values. • The actual parameter you have selected for SEVERITY is higher than the permitted range. The activated alarm is sent with SEVERITY=127 .
0	25	Communication has started. The alarm is being processed.
1	1	Communication problems: connection termination or no logon
1	4	At the first call: • The specified EV_ID is outside the permitted range or • the ANY pointer SD_i has a formal error. • The maximum memory area that can be sent for the CPU per "NOTIFY" has been exceeded.
1	10	Access to local user memory not possible (for example, access to a deleted DB).
1	12	When the instruction was called: • An instance DB was specified that does not belong to "NOTIFY" • A global DB was specified instead of an instance DB.
1	18	EV_ID was already used by the "NOTIFY", "NOTIFY_8P (Page 6639)", "ALARM (Page 6649)", "ALARM_8 (Page 6641)" or "ALARM_8P (Page 6643)" instructions.
1	20	Not enough work memory.
1	21	The alarm with the specified EV_ID is disabled.

See also

Creating program alarms (Page 3054)

ALARM: Create PLC alarms with acknowledgement display

Description

The instruction monitors a signal.

- Default mode (that is, acknowledgement triggered reporting is disabled): The block generates an alarm both on a rising edge (incoming event) and on a falling edge (outgoing event). You can have up to ten associated values sent with the alarm.
- Acknowledgement-triggered reporting is enabled: After an incoming alarm is generated, the block will no longer create alarms for the signal until you have acknowledged this incoming alarm on a display device.
  See also: Introduction to generating PLC alarms with instructions (Page 6635).

The alarm is sent to all stations logged on for this purpose.

When the SFB is first called, an alarm with the current signal state is sent.

The ACK_UP output is reset at the rising edge. It is set when your acknowledgement of the incoming event has arrived from a logged on display device.

The situation for the ACK_DN output is analogous: this is reset at the falling edge. It is set when your acknowledgement of the outgoing event is received from a logged on display device. Once your acknowledgement has been received from a logged on display device, the acknowledgement information is passed on to all other stations logged on for this purpose.

One alarm memory with two memory locations is available for each instance of the "ALARM" instruction. For more detailed information on buffering of signal changes, refer to Section "Signal change detection" in Introduction to generating PLC alarms with instructions (Page 6635).

The "ALARM" instruction conforms to IEC 1131-5.

Parameters

The following table shows the parameters of the "ALARM" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L	Control parameter enabled to receive decides whether the outputs ACK_UP and ACK_DN are updated at the first block call (EN_R=1) or not (EN_R=0). If EN_R=0, then the output parameters ACK_UP and ACK_DN remain unchanged.
SIG	Input	BOOL	I, Q, M, D, L	The signal to be monitored
ID	Input	WORD	I, Q, M, D, L or constant	Data channel for alarms: W#16#EEEE ID is only evaluated at the first call.

Parameter	Declaration	Data type	Memory area	Description
EV_ID	Input	C_ALARM	I, Q, M, D, L	Alarm number (not allowed: 0) EV_ID is only evaluated at the first call. Subsequently, the alarm number used for the first call applies to every call of "ALARM" with the corresponding instance DB. The alarm number will be assigned automatically. This ensures the consistency of the alarm numbers. The alarm number must be unique within your user program.
SEVERITY	Input	WORD	I, Q, M, D, L or constant	Weighting of the event Possible values: 0 to 127 (value 0 means highest weighting) This parameter is irrelevant for processing this alarm.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: Generation of alarm is complete.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR ERROR=TRUE indicates that an error has occurred during processing. For details refer to parameter STATUS.
STATUS	Output	WORD	I, Q, M, D, L	Status parameter STATUS: Displays error information
ACK_DN	Output	BOOL	I, Q, M, D, L	Event leaving state was acknowledged on a display device Initialization status: 1
ACK_UP	Output	BOOL	I, Q, M, D, L	Incoming event was acknowledged on a display device Initialization status: 1
SD_i, 1≤i ≤10	InOut	ANY	I, Q, M, D, T, C	i-th associated value Only the BOOL data types are permitted (not permitted: bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME. **Note:** When the ANY pointer accesses a DB, the DB must always be specified. (e.g.: P# DB10.DBX5.0 Byte 10)

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

ERROR and STATUS parameters

The following table contains all specific error information for the instruction "ALARM" that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Alarm loss: The previous signal change or the previous alarm could not be sent and is replaced by the current alarm.
0	22	• Error in the pointer to the associated values SD_i: – Involving the data length or the data type – No access to associated values in user memory, for example, due to deleted DB or area length error – The activated alarm is sent without associated values • The actual parameter you have selected for SEVERITY is higher than the permitted range. The activated alarm is sent with SEVERITY=127.
0	25	Communication has started. The alarm is being processed.
1	1	Communication problems: connection termination or no logon With acknowledgement-triggered reporting active: Temporary display, if display devices do not support acknowledgement-triggered reporting
1	4	At the first call: • The specified EV_ID is outside the permitted range or • the ANY pointer SD_i has a formal error. • The maximum memory area that can be sent for the CPU per "ALARM " has been exceeded.
1	10	Access to local user memory not possible (for example, access to a deleted DB).
1	12	When the instruction was called • An instance DB was specified that does not belong to ALARM • A global DB was specified instead of an instance DB.
1	18	EV_ID was already used by the "NOTIFY (Page 6646)", "NOTIFY_8P (Page 6639)", "ALARM", "ALARM_8 (Page 6641)" or "ALARM_8P (Page 6643)" instructions.
1	20	Not enough work memory.
1	21	The alarm with the specified EV_ID is disabled.

Note

After the first call, the ACK_UP and ACK_DN outputs have the value "1" and it is assumed that the previous value of the SIG input was "0".

See also

Creating program alarms (Page 3054)

Instructions for generating PLC alarms without instance DB

Startup characteristics of the instructions for generating PLC alarms

Warm restart behavior

During a warm restart, the instance DBs of the instructions for generating PLC alarms are set to the not initialized status. The actual parameters stored in the instance DBs are unchanged. The parameters ID and EV_ID will be re-evaluated at the next block call.

Cold restart behavior

During a cold restart, the contents of the instance DBs of the instructions for generating PLC alarms are set to their initial values.

Hot restart

During a hot restart, the instructions for generating PLC alarms behave like user function blocks that are capable of resuming execution. They continue from the point of interruption.

Memory reset

A memory reset always causes the termination of all connections so that no station is logged on for alarms. The user program is deleted. If you have inserted a FLASH card, the program sections relevant to execution are loaded on the CPU again from the card and the CPU executes a warm or cold restart (implicitly this is always a cold restart, since all user data are initialized after clearing memory).

Error behavior of instructions for generating PLC alarms

Connection termination

The connections assigned to the instances are monitored for termination. If a connection is terminated, the stations involved are removed from the internal CPU list of stations logged on for PLC alarms. Any alarms pending for these stations are deleted.

If other stations are still logged on following a connection termination, they continue to receive alarms. The instructions only stop sending alarms when there are no more connections to any logged on stations. The ERROR and STATUS output parameters indicate this situation (ERROR = 1, STATUS = 1).

Error interface to the user program

If an error occurs during the execution of an instruction for generating PLC alarms, the ERROR output parameter is set to "1" and the STATUS output parameter has the corresponding error identifier. You can evaluate this error information in your program.

Examples of possible errors:

- Sending not possible due to lack of resources
- Error accessing one of the signals to be monitored.

Introduction to generating PLC alarms with instructions

Instructions for generating PLC alarms

You can generate a PLC alarm with the following instructions:

- "ALARM_SQ (Page 6657)"
- "ALARM_S (Page 6655)"
- "ALARM_DQ (Page 6662)"
- "ALARM_D (Page 6660)"

These instructions have the following properties:

- The PLC alarms of "ALARM_SQ (Page 6657)" and "ALARM_DQ (Page 6662)" sent with signal state "1" can be acknowledged from a logged-on display device. The PLC alarms of "ALARM_S" (Page 6655) and "ALARM_D (Page 6660)" are always acknowledged implicitly.
- It is not a detected edge change, but rather every call that generates a PLC alarm.
- After the block processing the associated value SD is completely entered and allocated to the PLC alarm.
  The following applies for the associated value regarding the consistency relative to higher priority classes: The following are consistent:
  - The simple data types (bit, byte, word, and double word)
  - an array of the data type byte up to a maximum length specific for a CPU

ALARM_SC

With the "ALARM_SC (Page 6664)" instruction, you can:

- Determine the acknowledgement status of the last "incoming alarm" and the signal status at the last call of "ALARM_SQ (Page 6657)" / "ALARM_DQ (Page 6662)", or
- Determine the signal status at the last call of "ALARM_S (Page 6655)" / "ALARM_D (Page 6660)".

Logging on display devices

The instructions for generating PLC alarms only send a PLC alarm when they are called if at least one display device has logged on for PLC alarms.

Alarm storage

To avoid alarms being lost when there is a lot of traffic on the communications system, the "ALARM_SQ (Page 6657)", "ALARM_S (Page 6655)", "ALARM_DQ (Page 6662)" and "ALARM_D (Page 6660)" instructions can each buffer two alarms.

If PLC alarms are still lost, you will be informed in RET_VAL. The logged on display devices are informed of this the next time a PLC alarm can be sent.

Alarm acknowledgement for the instructions "ALARM_SQ" and "ALARM_DQ"

If you have acknowledged an "incoming alarm" at a display device, this acknowledgement information is first sent to the CPU where the alarm originated. This then distributes the acknowledgement information to all stations logged on for this purpose.

Disabling and enabling alarms

You cannot disable and then re-enable PLC alarms that you have generated with "ALARM_SQ (Page 6657)" or "ALARM_S (Page 6655)" or "ALARM_DQ (Page 6662)" or "ALARM_D (Page 6660)".

Changes in your program containing "ALARM_SQ" / "ALARM_S" calls

Note

When you download a block with "ALARM_SQ (Page 6657)" / "ALARM_S (Page 6655)" calls that is already on the CPU, it is possible that the previous block has sent an incoming PLC alarm but that the new block has not sent a corresponding outgoing PLC alarm. This means that the alarm remains in the internal alarm memory of the CPU. This status can also occur if you delete blocks with "ALARM_SQ (Page 6657)" / "ALARM_S (Page 6655)" calls.

You can remove such PLC alarms from the internal alarm memory of the CPU by changing the CPU to STOP mode and then executing a warm restart or cold restart.

Changes in your program containing "ALARM_DQ" / "ALARM_D" calls

Even though your program might contain "ALARM_DQ (Page 6662)" and/or "ALARM_D (Page 6660)" calls, the program modifications described above may cause the PLC alarms to become resident in the internal alarm memory and thus permanently occupy system resources.

In contrast to system resources that were allocated by "ALARM_SQ (Page 6657)" / "ALARM_S (Page 6655)" calls, you can again release system resources allocated by "ALARM_DQ (Page 6662)" / "ALARM_D (Page 6660)" calls without having to switch your CPU to STOP mode. The "DEL_SI (Page 6677)" instruction is used for this purpose. Before you enable dynamically-allocated system resources by calling "DEL_SI", it is advisable to use the "READ_SI" instruction to output information regarding the system resources of your CPU that are currently dynamically allocated (see instruction "READ_SI (Page 6674)").

Amount of transferable data

The amount of data that can be transferred using the associated values SD of the "ALARM_S (Page 6655)", "ALARM_SQ (Page 6657)", "ALARM_D (Page 6660)" and "ALARM_DQ (Page 6662)" instructions must not exceed a maximum length. This maximum data length is calculated as follows:

maxleng = min (pdu_local, pdu_remote) - 48

Where:

- pdu_local: the maximum length of CPU data blocks (SZL_ID W#16#0131, INDEX 1, tag pdu)
- pdu_remote: the maximum length for display device data blocks

Example

A CPU 414-2 sends an alarm to a programming device PG 760 (via MPI).

pdu_local = 480 bytes, pdu_remote = 480 bytes,

Result:

maxleng = min (480, 480) - 48 = 480 - 48 = 432

The maximum length of data that can be transferred per instruction is 432 bytes.

ALARM_S: Generate alarm message

Description

With every call, the instruction generates an alarm to which you can append an associated value. The alarm is sent to all stations logged on for this purpose. The instruction provides you with a simple alarm mechanism. You must make sure that you call the instruction only when the value of the alarm-triggering signal SIG is inverted compared to the last call. If this is not the case, then this situation will be indicated in RET_VAL and no alarm will be sent. With the very first call of the instruction you must ensure that the value "1" is applied at the SIG input. Otherwise you will get error information via RET_VAL and no alarm will be sent.

Note

In newly created programs you should use only the "ALARM_D (Page 6660)" instruction (provided your CPU supports this instruction) because they provide improved options for managing system resources.

Use of system resources

When alarms are generated the operating system allocates a system resource for the duration of one signal cycle.

The signal cycle lasts from the call with SIG=1 until the next call with SIG=0. If, during the signal cycle, the alarm-generating block is overloaded or deleted, the associated system resource remains assigned until the next restart (warm restart).

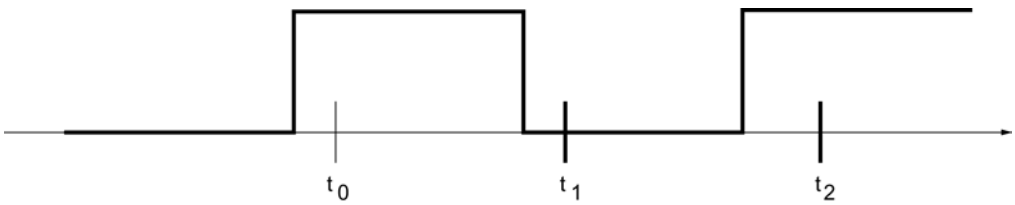
Acknowledging alarms

Alarms you have sent with "ALARM_S" are always implicitly acknowledged. You can determine the signal state at the last "ALARM_S" call using "ALARM_SC (Page 6664)".

Caching signal states

The ALARM_S" instruction allocates system resources Here among other things the last two signal states, including time stamp and associated value are placed in the buffer. If "ALARM_S" is called at a time when the signal states of the two last "valid" calls have not yet been sent (signal overflow), the current signal state and the last signal state are discarded and an overflow ID is set in the buffer. At the next possible opportunity, the next to last signal and the overflow identifier are sent.

Example:



If t_0, t_1 and t_2 are the call times of "ALARM_S". If the signal states of t_0 and t_1 have not yet been sent at the time t_2, the signal states of t_1 and t_2 will be discarded and the overflow identifier is set at the signal state of t_0.

Instance overflow

If the number of "ALARM_S" calls is greater than the maximum number of CPU system resources, this may result in a lack of resources (instance overflow). This situation is indicated both by the information in RET_VAL as well as by indications at the logged on display devices.

The maximum number of "ALARM_S" calls is dependent on the CPU.

Parameter

The following table shows the parameters of the instruction "ALARM_S":

Parameters	Declaration	Data type	Memory area	Description
SIG	Input	BOOL	I, Q, M, D, L	The alarm triggering signal
ID	Input	WORD	I, Q, M, D, L or constant	Data channel for alarms: W#16#EEEE
EV_ID	Input	C_ALARM_S	I, Q, M, D, L	Alarm number (not allowed: "0")
SD	Input	ANY	I, Q, M, D, T, C	Associated value Maximum length: 12 bytes Only the BOOL data types are permitted (not permitted: bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME.
RET_VAL	Return	INT	I, Q, M, D, L	Error information

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

RET_VAL parameter

Error code (W#16#...)	Explanation
0000	No error occurred.
0001	• The length of the associated value exceeds the maximum permitted length. • Access to user memory not possible (for example, access to deleted DB). The alarm is sent. • The associated value points to a value in the local data area. The alarm is sent. (S7-400 only)
0002	Warning: The last free alarm acknowledge memory was assigned. (S7-400 only)
8081	The specified EV_ID is outside the permitted range or
8082	Alarm loss because your CPU has no more resources for generating PLC alarms with instructions.
8083	Alarm loss, because the same signal transition is already present but could not be sent yet (signal overflow).
8084	With the current and the previous "ALARM_S", the alarm-triggering signal SIG has the same value.
8085	There is no logon for the specified EV_ID.
8086	A call for the specified EV_ID is already being processed in a lower priority class.
8087	During the first call of "ALARM_S", the alarm-triggering signal had the value "0".
8088	The specified EV_ID is already in use by another system resource ("ALARM_S", ALARM_DQ (Page 6662), ALARM_D (Page 6660)).
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

See also

Creating program alarms (Page 3054)

ALARM_SQ: Generate alarm message with acknowledgement

Description

With every call, the instruction generates an alarm to which you can append an associated value. The alarm is sent to all stations logged on for this purpose. "ALARM_SQ" also provides you with a simple alarm mechanism. You must make sure that you call "ALARM_SQ" only when the value of the alarm-triggering signal SIG is inverted to the last call. If this is not the case, then this situation will be indicated in RET_VAL and no alarm will be sent. Thus with the very first call of "ALARM_SQ", you must ensure that the value "1" is applied at the SIG input. Otherwise you will get error information via RET_VAL and no alarm will be sent.

Note

In newly created programs you should use only the "ALARM_DQ (Page 6662)" instruction (provided your CPU supports this instruction) because they provide improved options for managing system resources.

Use of system resources

When alarms are generated with "ALARM_SQ", the operating system allocates a system resource for the duration of one signal cycle.

With "ALARM_SQ", the signal cycle lasts from the call with SIG=1 until the next call with SIG=0. Added to this time interval is possibly also the time until the acknowledgement of the incoming signal by a logged on display device.

If, during the signal cycle, the alarm-generating block is overloaded or deleted, the associated system resource remains assigned until the next restart (warm restart).

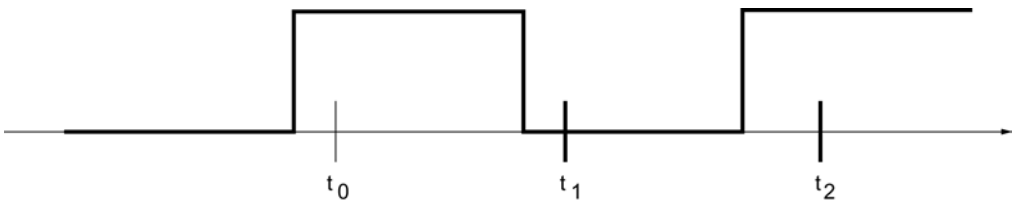
Acknowledging alarms

On a logged-on display device, you can acknowledge the alarms with signal status "1" sent by "ALARM_SQ". You can determine the acknowledgement status of the last "incoming alarm" and the signal status at the last "ALARM_SQ" call using "ALARM_SC (Page 6664)".

Caching signal states

"ALARM_SQ" allocates system resources Here among other things the last two signal states, including time stamp and associated value are placed in the buffer. If "ALARM_SQ" is called at a time when the signal states of the two last "valid" calls have not yet been sent (signal overflow), the current signal state and the last signal state are discarded and an overflow ID is set in the buffer. At the next possible opportunity, the next to last signal and the overflow identifier are sent.

Example:



If t_0, t_1 and t_2 are the call times of "ALARM_SQ". If the signal states of t_0 and t_1 have not yet been sent at the time t_2, the signal states of t_1 and t_2 will be discarded and the overflow identifier is set at the signal state of t_0.

Instance overflow

If the number of "ALARM_SQ" calls is greater than the maximum number of CPU system resources, this may result in a lack of resources (instance overflow). This situation is indicated both by the information in RET_VAL as well as by indications at the logged on display devices.

The maximum number of "ALARM_SQ" calls is dependent on the CPU.

Parameter

The following table shows the parameters of the instruction "ALARM_SQ":

Parameter	Declaration	Data type	Memory area	Description
SIG	Input	BOOL	I, Q, M, D, L	The alarm triggering signal
ID	Input	WORD	I, Q, M, D, L or constant	Data channel for alarms: W#16#EEEE
EV_ID	Input	C_ALARM_S	I, Q, M, D, L	Alarm number (not allowed: 0)
SD	Input	ANY	I, Q, M, D, T, C	Associated value Maximum length: 12 bytes Only the BOOL data types are permitted (not permitted: bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME.
RET_VAL	Return	INT	I, Q, M, D, L	Error information

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

RET_VAL parameter

Error code (W#16#...)	Explanation
0000	No error occurred.
0001	• The length of the associated value exceeds the maximum permitted length. • Access to user memory not possible (for example, access to deleted DB). The alarm is sent. • The associated value points to a value in the local data area. The alarm is sent. (S7-400 only)
0002	Warning: The last free alarm acknowledge memory was assigned. (S7-400 only)
8081	The specified EV_ID is outside the permitted range or
8082	Alarm loss because your CPU has no more resources for generating PLC alarms with instructions.
8083	Alarm loss, because the same signal transition is already present but could not be sent yet (signal overflow).
8084	With the current and the previous "ALARM_SQ", the alarm-triggering signal SIG has the same value.
8085	There is no logon for the specified EV_ID.
8086	A call for the specified EV_ID is already being processed in a lower priority class.
8087	During the first call of "ALARM_SQ", the alarm-triggering signal had the value "0".
8088	The specified EV_ID is already in use by another system resource (ALARM_S (Page 6655), ALARM_DQ (Page 6662), ALARM_D (Page 6660)).
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

See also

Creating program alarms (Page 3054)

ALARM_D: Create permanently acknowledged PLC alarms

Description

With every call, the instruction generates a PLC alarm to which you can append an associated value. It conforms to the "ALARM_S (Page 6655)" instruction in this regard.

When alarms are generated with "ALARM_D", the operating system allocates a system resource for the duration of one signal cycle.

With "ALARM_D", the signal cycle lasts from the call with SIG=1 until the next call with SIG=0. If, during the signal cycle, the alarm-generating block is overloaded or deleted, the associated system resource remains assigned until the next restart (warm restart).

Compared to the "ALARM_S (Page 6655)" instruction, "ALARM_D" has an additional functionality that allows it to manage allocated system resources.

- With the help of READ_SI, you can read information regarding allocated system resources.
- With DEL_SI, you can re-enable allocated system resources. This is of special significance for permanently assigned system resources. A currently allocated system resource, for example, stays allocated until the next restart (warm restart) if you delete an FB call that contains "ALARM_D" calls in the course of a program change. When you change the program and reload an FB with "ALARM_D" calls, it may be the case that "ALARM_D" does not generate any more PLC alarms.

The instruction "ALARM_D" contains one parameter more than the "ALARM_S (Page 6655)" instruction, namely the input CMP_ID. You use this input to assign the PLC alarms generated by the "ALARM_D" instruction to logical areas, for example, to parts of the system. If you call "ALARM_D" in an FB, the obvious thing to do is to assign the number of the corresponding instance DB to "CMP_ID".

Parameter

The following table shows the parameters of the instruction "ALARM_D":

Parameter	Declaration	Data type	Memory area	Description
SIG	Input	BOOL	I, Q, M, D, L	The alarm triggering signal
ID	Input	WORD	I, Q, M, D, L or constant	Data channel for alarms: W#16#EEEE
EV_ID	Input	C_ALARM_S	I, Q, M, D, L	Alarm number (not allowed: 0)
CMP_ID	Input	DWORD	I, Q, M, D, L or constant	component identifier (not permitted: 0) ID for the subsystem to which the corresponding alarm is assigned Recommended values: • Low word: 1 to 65535 • High word: 0 You will not have any problems with the SIEMENS program package if you comply with these recommendations.

Parameter	Declaration	Data type	Memory area	Description
SD	Input	ANY	I, Q, M, D, T, C	Associated value Maximum length: 12 bytes Only the BOOL data types are permitted (not permitted: bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME
RET_VAL	Return	INT	I, Q, M, D, L	Error information

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

RET_VAL parameter

Error code (W#16#...)	Explanation
0000	No error occurred.
0001	• The length of the associated value exceeds the maximum permitted length, or • Access to user memory not possible (for example, access to deleted DB). The alarm is sent. • The associated value points to a value in the local data area. The alarm is sent. (S7-400 only)
0002	Warning: The last free alarm acknowledge memory was assigned. (S7-400 only)
8081	The specified EV_ID is outside the permitted range or
8082	Alarm loss because your CPU has no more resources for generating PLC alarms with instructions.
8083	Alarm loss, because the same signal transition is already present but could not be sent yet (signal overflow).
8084	With the current and the previous "ALARM_D", the alarm-triggering signal SIG has the same value.
8085	There is no logon for the specified EV_ID.
8086	A call for the specified EV_ID is already being processed in a lower priority class.
8087	During the first call of "ALARM_D", the alarm-triggering signal had the value "0".
8088	The specified EV_ID is already in use by another system resource ("ALARM_SQ (Page 6657)", "ALARM_S (Page 6655)", "ALARM_DQ" (Page 6662)).
8089	You have assigned the value "0" to CMP_ID.
808A	CMP_ID does not match EV_ID
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

See also

Creating program alarms (Page 3054)

ALARM_DQ: Create acknowledgeable PLC alarms

Description

With every call, the instruction "ALARM_DQ" generates an alarm to which you can append an associated value. They conform to the "ALARM_SQ (Page 6657)" instruction in this regard.

When alarms are generated with "ALARM_DQ", the operating system allocates a system resource for the duration of one signal cycle.

With "ALARM_DQ", the signal cycle lasts from the call with SIG=1 until the next call with SIG=0. Added to this time interval is possibly also the time until the acknowledgement of the incoming signal by a logged on display device.

If, during the signal cycle, the alarm-generating block is overloaded or deleted, the associated system resource remains assigned until the next restart (warm restart).

Compared to the "ALARM_DQ" instruction, the "ALARM_SQ (Page 6657)" has an additional functionality that allows it to manage allocated system resources.

- With the help of READ_SI, you can read information regarding allocated system resources.
- With DEL_SI, you can re-enable allocated system resources. This is of special significance for permanently assigned system resources. A currently allocated system resource, for example, stays allocated until the next restart (warm restart) if you delete an FB call that contains "ALARM_DQ" calls in the course of a program change. When you change the program and reload an FB with "ALARM_DQ" calls, it may be the case that "ALARM_DQ" does not generate any more alarms.

The "ALARM_DQ" instruction contains one parameter more than the "ALARM_SQ (Page 6657)" instruction, namely the input CMP_ID. You use this input to assign the alarms generated by the "ALARM_DQ" instruction to logical areas, for example, to parts of the system. If you call "ALARM_DQ" in an FB, the obvious thing to do is to assign the number of the corresponding instance DB to CMP_ID.

Parameter

The following table shows the parameters of the instruction "ALARM_DQ":

Parameter	Declaration	Data type	Memory area	Description
SIG	Input	BOOL	I, Q, M, D, L	The alarm triggering signal
ID	Input	WORD	I, Q, M, D, L or constant	Data channel for alarms: W#16#EEEE
EV_ID	Input	C_ALARM_S	I, Q, M, D, L	Alarm number (not allowed: 0)
CMP_ID	Input	DWORD	I, Q, M, D, L or constant	component identifier (not permitted: 0); ID for the subsystem to which the corresponding alarm is assigned Recommended values: • Low word: 1 to 65535 • High word: 0 You will not have any problems with the SIEMENS program package if you comply with these recommendations.

Parameter	Declaration	Data type	Memory area	Description
SD	Input	ANY	I, Q, M, D, T, C	Associated value Maximum length: 12 bytes Only the BOOL data types are permitted (not permitted: bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME
RET_VAL	Return	INT	I, Q, M, D, L	Display

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

RET_VAL parameter

Error code (W#16#...)	Explanation
0000	No error occurred.
0001	• The length of the associated value exceeds the maximum permitted length, or • Access to user memory not possible (for example, access to deleted DB). The alarm is sent. • The associated value points to a value in the local data area. The alarm is sent. (S7-400 only)
0002	Warning: The last free alarm acknowledge memory was assigned. (S7-400 only)
8081	The specified EV_ID is outside the permitted range or
8082	Alarm loss because your CPU has no more resources for generating PLC alarms with instructions.
8083	Alarm loss, because the same signal transition is already present but could not be sent yet (signal overflow).
8084	With the current and the previous "ALARM_DQ", the alarm-triggering signal SIG has the same value.
8085	There is no logon for the specified EV_ID.
8086	A call for the specified EV_ID is already being processed in a lower priority class.
8087	During the first call of "ALARM_DQ", the alarm-triggering signal had the value "0".
8088	The specified EV_ID is already in use by another system resource (ALARM_SQ (Page 6657) , ALARM_S (Page 6655) , ALARM_D (Page 6660)).
8089	You have assigned the value "0" to CMP_ID .
808A	CMP_ID does not match EV_ID
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

See also

Creating program alarms (Page 3054)

ALARM_SC: Determine acknowledgement status of the last ALARM_SQ incoming alarm

Description

You use the instruction to do the following:

- Determine the acknowledgement status of the last "ALARM_SQ (Page 6657)" / "ALARM_DQ (Page 6662)" incoming alarm and the status of the alarm-triggering signal when the last "ALARM_SQ (Page 6657)" / "ALARM_DQ (Page 6662)" was called, or
- Determine the status of the alarm-triggering signal at the last "ALARM_S (Page 6655)" / "ALARM_D (Page 6660)" call.

The alarm or the signal is uniquely referenced via the alarm number that you assign here, if you have assigned alarms numbers via the alarm configuration.

The instruction accesses the temporarily allocated memory of the "ALARM_SQ (Page 6657)", "ALARM_S (Page 6655)", "ALARM_DQ (Page 6662)" and "ALARM_D (Page 6660)" instructions.

Parameter

The following table shows the parameters of the instruction "ALARM_SC":

Parameter	Declaration	Data type	Memory area	Description
EV_ID	Input	C_ALARM_S	I, Q, M, D, L	Alarm number for which you want to determine the signal state at the last call or the acknowledgement status of the last incoming alarm (only with ALARM_SQ and with ALARM_DQ).
RET_VAL	Return	INT	I, Q, M, D, L	Error information
STATE	Output	BOOL	I, Q, M, D, L	State of the alarm-triggering signal at the last call
Q_STATE	Output	BOOL	I, Q, M, D, L	• If the specified EV_ID parameter belongs to an "ALARM_S" / "ALARM_D" call: "1" • If the specified EV_ID parameter belongs to an "ALARM_SQ" / "ALARM_DQ" call: Acknowledgement status of the last incoming alarm: – 0: Not acknowledged – 1: Acknowledged

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

RET_VAL parameter

Error code (W#16#...)	Explanation
0000	No error occurred.
8081	The specified EV_ID is outside the permitted range or
8082	No memory location is currently assigned to this EV_ID (possible cause: The corresponding signal state was never "1" or its signal state has already returned to "0").
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

See also

Creating program alarms (Page 3054)

WR_USMSG: Write a user diagnostics event to the diagnostic buffer

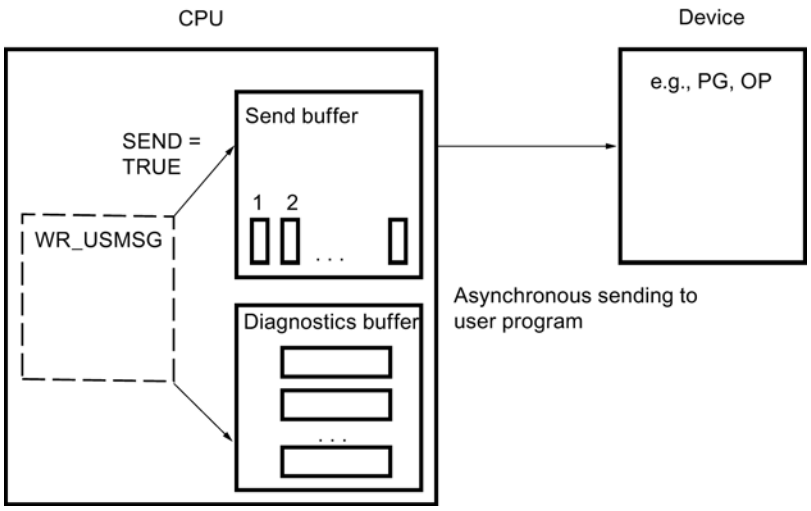
Description

You use the instruction to write a user diagnostics result in the diagnostics buffer. You can also send the corresponding user diagnostics alarm to all stations logged on for this purpose (by setting the input parameter SEND = TRUE). If an error occurs, the output parameter RET_VAL provides the error information.

Sending user diagnostics alarm

The instruction writes a user diagnostics event to the diagnostics buffer. You can also send the corresponding user diagnostics alarm to all stations logged on for this purpose (by setting the input parameter SEND = TRUE). The user diagnostics alarm is then written to the send buffer and automatically sent to the stations logged on for this purpose.

You can check whether the sending of user diagnostics alarms is currently possible. To do this, call the "RDSYSST (Page 6681)" instruction with the SZL_ID = W#16#0132 and INDEX = W#16#0005 parameters. The fourth word of the data record supplied in this manner indicates whether sending is currently possible (1) or not (0).



Send buffer full

The user diagnostics alarm can only be entered in the send buffer if the send buffer is not full. The number of entries that can be made in the send buffer depends on the type of CPU you are using.

If the send buffer is full, then:

- The diagnostics event is nevertheless entered in the diagnostics buffer.
- The RET_VAL parameter indicates that the send buffer is full. (RET_VAL = W#16#8092).

Station not logged on

If a user diagnostics alarm is to be sent (SEND = TRUE) and no station is logged on,

- the user diagnostics event will still be entered in the diagnostics buffer.
- the RET_VAL parameter indicates that no station is logged on (W#16#0091 or W#16#8091. The value W#16#8091 appears only with older versions of the CPU.)

Structure of an entry

The internal structure of an entry in the diagnostics buffer is as follows:

Byte	Contents
1 and 2	Event ID
3	Priority class
4	OB number
5 and 6	Reserved
7 and 8	Additional information 1
9, 10, 11 and 12	Additional information 2
13 to 20	Time stamp

Event ID

An event ID is assigned to every event.

Additional information

This is additional information about the event. The additional information can differ for each event. When you create a user diagnostics event, you can decide on the content of these entries yourself.

When you send a user diagnostics alarm, you can integrate the additional information as associated values in the (event ID-specific) alarm text.

Time stamp

The time stamp is of the type Date_and_Time.

Parameters

The following table shows the parameters of the "WR_USMSG" instruction:

Parameter	Declaration	Data type	Memory area	Description
SEND	Input	BOOL	I, Q, M, D, L	Enable the sending of the user diagnostics alarm to all logged-on stations
EVENTN	Input	WORD	I, Q, M, D, L or constant	Event ID. You assign the event ID. The assignment is not executed by the alarm server.
INFO1	Input	ANY	I, Q, M, D, L	Additional information: 1 word long
INFO2	Input	ANY	I, Q, M, D, L	Additional information: 2 words long
RET_VAL	Return	INT	I, Q, M, D, L	Error information

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter SEND

If SEND = TRUE , the user diagnostics alarm will be sent to all logged-on stations. The alarm is only sent if the station is logged on and if the send buffer is not full. The sending of the element is asynchronous to the user program.

Parameter EVENTN

The EVENTN parameter contains the event ID of the user event. You can enter event IDs of the form W#16#8xyz , W#16#9xyz, W#16#Axyz, W#16#Bxyz.

IDs in the form W#16#8xyz and W#16#9xyz belong to predefined events, IDs in the form W#16#Axyz and W#16#Bxyz belong to freely defined events.

An incoming event is indicated by x = 1, an outgoing event by x = 0. With events in class A and B: yz is the number assigned in hexadecimal presentation in the alarm configuration for the corresponding alarm.

Parameter INFO1

The INFO1 parameter contains information that is one word long. The following data types are permitted for INFO1 :

- WORD
- INT
- ARRAY [0...1] OF CHAR

You can integrate the INFO1 parameter as an associated value in the alarm text and use it to add up-to-date information to the alarm.

Parameter INFO2

The INFO2 parameter contains information that is two words long. The following data types are permitted for INFO2 :

- DWORD
- DINT
- REAL
- TIME
- ARRAY [0...3] OF CHAR

You can integrate the INFO2 parameter as an associated value in the alarm text and use it to add up-to-date information to the alarm.

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error
0091	No station logged on (diagnostics event entered in the diagnostics buffer).
8083	INFO1 data type is not permitted
8084	INFO2 data type is not permitted
8085	EVENTN data type is not permitted
8086	Length of INFO1 is not permitted
8087	Length of INFO2 is not permitted
8091	(This error code appears only with older versions of the CPU.) No station logged on (diagnostics event entered in the diagnostics buffer).

Error code (W#16#...)	Explanation
8092	Sending not possible at present, send buffer full (diagnostics event entered in the diagnostics buffer).
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

AR_SEND: Send archive data

Description

The instruction sends archive data to operator control and operating systems logged on for this purpose. These systems inform the CPU of the relevant archive number in the logon message frame. Depending on the memory available on the CPU and the operand area used, the archive data can be up to 65 534 bytes long. The defaults of the operator interface system you are using must be taken into consideration in the structure of the archive data.

The sending process is activated after the block is called and a rising edge is detected at the control input REQ. The start address of the archive data that is to be sent is specified by SD_1, the length of the data field by LEN. Data transfer is asynchronous to the execution of the user program. Successful completion of the send operation is indicated by the value of the DONE status parameter being set to "1". If there is a rising edge at control input R, the current sending process is canceled.

Parameters

The following table shows the parameters of the "AR_SEND" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter request
R	Input	BOOL	I, Q, M, D, L	Control parameter reset: Current job aborted
ID	Input	WORD	I, Q, M, D or constant	Data channel for alarms: W#16#EEEE ID is only evaluated at the first call.
AR_ID	Input	C_AR_SEND	I, Q, M, D, L or constant	Archive number (not permitted: 0); AR_ID is only evaluated at the first call. Subsequently, the archive number used for the first call applies to every call of AR_SEND with the corresponding instance DB. The archive number is assigned automatically. This ensures the consistency of the archive numbers.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: Sending complete
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: ERROR=TRUE indicates that an error has occurred during processing. For details refer to parameter STATUS.

Parameter	Declaration	Data type	Memory area	Description
STATUS	Output	WORD	I, Q, M, D, L	Status parameter STATUS: Displays error information
SD_1	InOut	ANY	I, Q, M, D, T, C	Pointer to archive data. The length specification is not evaluated. Only the BOOL data types are permitted (not permitted: Bit array, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME. The archive data must have a destination-system specific structure. **Note:** When the ANY pointer accesses a DB, the DB must always be specified (for example: P# DB10.DBX5.0 byte 10).
LEN	InOut	WORD	I, Q, M, D, L	Length of the data field to be sent in bytes

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameters ERROR and STATUS

The following table contains all specific error information that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Warning: New job not active because the previous job is still busy.
0	25	Communication has started. The job is being processed.
1	1	Communication problems
1	2	Negative acknowledgement, function cannot be executed
1	3	There is no logon for the specified AR_ID.
1	4	• Error in the archive data pointer SD_1 involving data length or data type. • At the first call, the specified AR_ID is outside the permitted range.
1	5	Requested reset was executed.
1	7	RESET job irrelevant because the current function was completed or not activated (block in incorrect status).
1	10	Access to local user memory not possible (e.g. access to deleted DB)
1	12	When the instruction was called • An instance DB was specified that does not belong to AR_SEND • A global DB was specified instead of an instance DB.
1	18	AR_ID has already been used by an AR_SEND.
1	20	Not enough work memory.

Data consistency

To ensure data consistency, you may only write to the part of the currently used sending area SD_1 after the current send operation is complete. This is the case when the DONE status parameter assumes the value "1".

Others

EN_MSG: Enable PLC alarms

Description

You use this instruction to enable PLC alarms you have disabled. You may have disabled the alarms either on a display device or using "DIS_MSG (Page 6673)".

You specify the alarms to be enabled using the MODE and MESGN input parameters. Successful enabling of alarms with the instruction is only possible if the instruction is not already actively enabling alarms.

You start the enable process by assigning the value "1" to the input parameter REQ during the call.

Functional description

The enable process is executed asynchronously, in other words, it can extend over several calls from the instruction (see also Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 5367)):

- When the instruction is first called (REQ =1), it checks the input parameters and attempts to assign the required system resources. In the good case, RET_VAL W#16#7001 is entered, BUSY is set, and enabling is triggered.
  Otherwise the associated error information is entered in RET_VAL and the job is concluded. In this case, BUSY should not be evaluated.
- If there are further calls in the meantime, the value RET_VAL W#16#7002 is entered (job still being executed by the CPU) and BUSY is set. An intermediate call does not affect the current job.
- The last time the instruction is called, the value RET_VALW#16#0000 is entered if no error has occurred. In this case, BUSY is written with 0. If there is an error then the error information is entered in RET_VAL and BUSY should not be evaluated.

Parameter

The following table shows the parameters of the instruction "EN_MSG":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = 1: Trigger enable
MODE	Input	BYTE	I, Q, M, D, L or constant	See the table below for parameters for selecting the alarms to be enabled;

Parameter	Declaration	Data type	Memory area	Description
MESGN	Input	DWORD	I, Q, M, D, L or constant	Alarm number Only relevant when MODE = 6. This allows a single alarm to be enabled.
RET_VAL	Return	INT	I, Q, M, D, L	Error information (see below);
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The enabling process is not yet complete.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter MODE

The following table shows the permitted values for the MODE input parameter.

Value	Meaning
0	All PLC alarms of the CPU generated with instructions
1	All CPU PLC alarms generated with instructions, which means all alarms of "NOTIFY (Page 6646)", "NOTIFY_8P (Page 6639)", "ALARM (Page 6649)", "ALARM_8P (Page 6643)" and "ALARM_8 (Page 6641)".
6	Single alarm of the "PLC alarms generated with instructions" class

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	Enabling was terminated without an error.
7000	REQ = 0 at first call: enabling was not activated.
7001	REQ = 1 at first call: enabling was triggered.
7002	Intermediate call: enabling is already active.
8081	Error accessing a parameter
8082	MODE has an illegal value.
8083	The alarm number is outside the permitted range of values.
8084	There is no logon for the alarm(s) specified with MODE and possibly MESGN .
80C3	Enabling of alarm(s) specified via MODE and possibly MESGN , cannot be initiated because another enabling process of "EN_MSG " is currently active.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

DIS_MSG: Disable PLC alarms

Description

You use this instruction to disable PLC alarms you have created with instructions. You select alarms to be disabled using the input parameters MODE and MESGN. Calling the "DIS_MSG" instruction and successfully disabling an alarm is only possible if disabling of an alarm is not already active with "DIS_MSG".

Alarms that are ready to be sent when "DIS_MSG" is called but that are still in an internal buffer can no longer be disabled and are sent.

A disabled alarm transmission is indicated at the ERROR and STATUS outputs of the "NOTIFY (Page 6646)", "NOTIFY_8P (Page 6643)", "ALARM (Page 6649)", "ALARM_8P (Page 6643)" and "ALARM_8 (Page 6641)" instructions.

You start the process by assigning the value "1" to the input parameter REQ when "DIS_MSG" is called.

Functional description

The reading process is executed asynchronously, in other words, it can extend over several calls (see also Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 5367)):

- When it is first called (REQ =1), "DIS_MSG" checks the input parameters and attempts to occupy the required system resources. In the good case, RET_VALW#16#7001 is entered, BUSY is set and disabling is triggered.
  Otherwise the associated error information is entered in RET_VAL and the job is concluded. In this case, BUSYmust not be evaluated.
- If there are further calls in the meantime, the value RET_VALW#16#7002 is entered (job still being executed by the CPU) and BUSY is set. An intermediate call does not affect the current job.
- The last time the instruction is called, the value RET_VALW#16#0000 is entered if no error has occurred. In this case, BUSY is written with "0". If there is an error then the error information is entered in RET_VAL and BUSY should not be evaluated.

Parameter

The following table shows the parameters of the instruction "DIS_MSG":

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = 1: Trigger disable
MODE	Input	BYTE	I, Q, M, D, L or constant	Parameter for selecting the alarms to be disabled, see following table
MESGN	Input	DWORD	I, Q, M, D, L or constant	Alarm number Alarm number only relevant when MODE = 5, 6, 7. This allows a single alarm to be disabled.

Parameters	Declaration	Data type	Memory area	Description
RET_VAL	Return	INT	I, Q, M, D, L	Error code (see below)
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The disabling process is not yet complete.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter MODE

The following table shows the permitted values for the MODE input parameter.

Value	Meaning
0	All PLC alarms of the CPU generated with instructions
1	All CPU PLC alarms generated with instructions, which means all alarms of "NOTIFY (Page 6646)", "NOTIFY_8P (Page 6639)", "ALARM (Page 6649)", "ALARM_8P (Page 6643)" and "ALARM_8 (Page 6641)".
6	Single alarm of the "PLC alarms generated with instructions" class

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	Disabling was terminated without an error.
7000	REQ = 0 at first call: disabling was not activated.
7001	REQ = 1 at first call: disabling was triggered.
7002	Intermediate call: disabling is already active.
8081	Error accessing a parameter
8082	MODE has an illegal value.
8083	The alarm number is outside the permitted range of values.
8084	There is no logon for the alarm(s) specified with MODE and possibly MESGN .
80C3	Disabling of alarm(s) specified via MODE and possibly MESGN , cannot be initiated because another disabling process of "DIS_MSG" is currently active.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

READ_SI: Read out dynamically assigned system resources

Origin of dynamically allocated system resources when generating alarms with the "ALARM_DQ" and "ALARM_D" instructions

For alarm generation with "ALARM_DQ (Page 6662)" and "ALARM_D (Page 6660)", the operating system temporarily allocates memory space in the system memory.

For example, if you delete an FB with "ALARM_DQ (Page 6662)" or "ALARM_D (Page 6660)" calls that was present in the CPU, it may be the case that corresponding system resources remain permanently allocated. If you reload the FB with "ALARM_DQ (Page 6662)" or "ALARM_D (Page 6660)" calls, it may be that the "ALARM_DQ (Page 6662)" and "ALARM_D (Page 6660)" instructions are no longer processed properly.

Description

Use the "READ_SI" instruction to read out the currently used system resources that were allocated for alarm generation with "ALARM_DQ (Page 6662)" and "ALARM_D (Page 6660)". This is done via the values of EV_ID and CMP_ID used there. These values are passed on to the "READ_SI" instruction in the SI_ID parameter.

The "READ_SI" instruction has four possible operating modes, which are explained in the following table. You set the requested operating mode via the MODE parameter.

MODE	Which of the system resources allocated to "ALARM_DQ" / "ALARM_D" are read out?
1	All (calling of "READ_SI" using SI_ID:=0)
2	The system resources that were allocated to the "ALARM_DQ" / "ALARM_D" call with EV_ID:=ev_id calling of READ_SI using SI_ID:=ev_id)
3	All system resources that were allocated to the "ALARM_DQ" / "ALARM_D" call with CMP_ID:=cmp_id (calling of using READ_SI using SI_ID:=cmp_id)
0	Additional system resources that could not be read with the previous call in MODE=1 or MODE=3 because you have selected a destination field SYS_INST that is too small

Functional description

If you have selected a sufficiently large SYS_INST destination area when you called READ_SI with MODE=1 or MODE=3, it will contain the content of all currently occupied system resources selected via the MODE parameter after the call.

If the amount of system resources currently occupied is very large, the runtime will be correspondingly high. That is, a high load on CPU performance may cause the the maximum configurable cycle monitoring time to be exceeded.

You can work around this runtime problem as follows: Select a relatively small SYS_INST destination area. If the instruction cannot enter all of the system resources that will be read out in SYS_INST, this fact will be communicated to you via RET_VAL=W#16#0001. Then call READ_SI with MODE=0 and the same SI_ID as for the previous call until RET_VAL takes on the value W#16#0000.

Note

Because the operating system does not coordinate the READ_SI calls that belong to a read job, you should execute all READ_SI calls in the same priority class.

Parameter

The following table shows the parameters of the instruction "READ_SI":

Parameters	Declaration	Data type	Memory area	Description
MODE	Input	INT	I, Q, M, D, L or constant	Job identifier Permitted values: • 1: Read all system resources • 2: Read the system resources to which "ALARM_DQ"-/ "ALARM_D" = ev_id was allocated when EV_ID = ev_id was called. • 3: Read the system resources to which "ALARM_DQ"-/ "ALARM_D" = ev_id was allocated when CMP_ID = cmp_id was called. • 0: subsequent call
SI_ID	Input	DWORD	I, Q, M, D, L or constant	ID for the system resource(s) to be read Permitted values: • 0, if MODE=1 • Alarm number ev_id, if MODE=2 • ID cmp_id for identification of the subsystem, if MODE=3
RET_VAL	Return	INT	I, Q, M, D, L	Return value (error information or job status)
N_SI	Output	INT	I, Q, M, D, L	Number of system resources output in SYS_INT
SYS_INST	Output	ANY	D	Destination area for the read system resources.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error occurred.
0001	Not all system resources could be read because the SYS_INST destination range you have selected is too short.
8081	(only with MODE=2 or 3). You have assigned the value 0 to SI_ID .
8082	(only with MODE=1). You have assigned one of 0 different values to SI_ID .
8083	(only with MODE=0). You have assigned SI_ID a value other than the value assigned for the preceding call with MODE=1 or 3.
8084	You have assigned an illegal value to MODE .
8085	READ_SI is already being processed in another OB.
8086	The destination area SYS_INST is too small for a system resource.

Error code (W#16#...)	Explanation
8087 Or 8092	The destination area SYS_INST does not exist in a DB or error in the ANY pointer.
8xyy	General error information See also: ALARM_D: Create permanently acknowledged PLC alarms (Page 6660)

Destination area parameter SYS_INST - Structure

The destination area for the read assigned system resources must be within a DB. You should appropriately define the destination area as a field of structures; a structure is comprised of the following:

Structure element	Data type	Description
SFC_NO	WORD	Number of the instruction allocated by the system resource
LEN	BYTE	Length of the structure in bytes, including SFC_NO and LEN: B#16#0C
SIG_STAT	BOOL	Signal state
ACK_STAT	BOOL	Acknowledgement status of the incoming event (positive edge)
EV_ID	DWORD	Alarm number
CMP_ID	DWORD	Subsystem ID

DEL_SI: Delete dynamically assigned system resources

Origin of dynamically allocated system resources when generating alarms with the "ALARM_DQ" and "ALARM_D" instructions

For alarm generation with "ALARM_DQ (Page 6662)" and "ALARM_D (Page 6660)", the operating system temporarily allocates memory space in the system memory.

For example, if you delete an FB with "ALARM_DQ (Page 6662)" or "ALARM_D (Page 6660)" calls that was present in the CPU, it may be the case that corresponding system resources remain permanently allocated. If you reload the FB with "ALARM_DQ (Page 6662)" / "ALARM_D (Page 6660)" calls, it may be that the "ALARM_DQ (Page 6662)" and "ALARM_D (Page 6660)" instructions will no longer be processed properly.

Description

You use the "DEL_SI" instruction to delete system resources that are currently in use.

"DEL_SI" has three possible operating modes, which are explained in the following table. You set the requested operating mode via the MODE parameter.

MODE	Which of the system resources allocated to "ALARM_DQ"/ "ALARM_D" will be deleted?
1	All (calling of "DEL_SI" using SI_ID:=0)
2	The system resources that were allocated to the EV_ID:=ev_id when "ALARM_DQ" (Page 6662) / ALARM_D (Page 6660) was called (DEL_SI called using SI_ID:=ev_id)
3	All system resources to which CMP_ID:=cmp_id was allocated when "ALARM_DQ" (Page 6662) / ALARM_D (Page 6660) was called (DEL_SI called with SI_ID:=cmp_id)

Parameter

The following table shows the parameters of the instruction "DEL_SI":

Parameters	Declaration	Data type	Memory area	Description
MODE	Input	INT	I, Q, M, D, L or constant	Job identifier Permitted values: • 1: delete all system resources • 2: Delete the system resources that were allocated to EV_ID = ev_id when "ALARM_DQ" (Page 6662) / "ALARM_D (Page 6660)" was called • 3: Delete the system resources that were allocated to CMP_ID = cmp_id when "ALARM_DQ" (Page 6662) / "ALARM_D (Page 6660)" was called
SI_ID	Input	DWORD	I, Q, M, D, L or constant	ID of the system resource(s) to be deleted Permitted values: • 0, if MODE=1 • Alarm number ev_id, if MODE=2 • ID cmp_id for identification of the subsystem, if MODE=3
RET_VAL	Return	INT	I, Q, M, D, L	Error information

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error occurred.
8081	(only with MODE=2 or 3). You have assigned the value "0" to SI_ID .
8082	(only with MODE=1). You have assigned one of "0" different values to SI_ID.
8084	You have assigned an illegal value to MODE .
8085	"DEL_SI" is currently being processed.
8086	Not all selected system resources could be deleted because at least one of them was being processed when "DEL_SI" was called.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

Diagnostics

RD_SINFO: Read current OB start information

Description

You read the start information with the instruction:

- of the OB last called that has not yet been completely executed, and
- of the startup OB started last.

There is no time stamp in either case. If the call is in OB 100 or OB 101 or OB 102, two identical start information messages will be returned.

Parameters

The following table shows the parameters of the instruction "RD_SINFO":

Parameters	Declaration	Data type	Memory area	Description
RET_VAL	Return	INT	I, Q, M, D, L	Error information
TOP_SI	Output	STRUCT	D, L	Start information of the current OB
START_UP_SI	Output	STRUCT	D, L	Start information of the startup OB last started

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

The instruction provides only general and not specific error information. The general error information and its evaluation are written to the "RET_VAL" output parameter.

See also: Evaluating errors with output parameter RET_VAL (Page 5370).

Parameter TOP_SI and START_UP_SI

The output parameters TOP_SI and START_UP_SI are two identically set up structures. Their structure is provided in the following table.

Structure element	Data type	Description
EV_CLASS	BYTE	• Bits 0 to 3: Event ID • Bits 4 to 7: Event class
EV_NUM	BYTE	Event number
PRIORITY	BYTE	Number of the priority class (meaning of B#16#FE: OB not available or disabled or cannot be started in current operating mode)
NUM	BYTE	OB number
TYP2_3	BYTE	Data ID 2_3: Identifies the information entered in ZI2_3

Structure element	Data type	Description
TYP1	BYTE	Data ID 1: Identifies the information entered in ZI1
ZI1	WORD	Additional information 1
ZI2_3	DWORD	Additional information 2_3

Note

The structure elements listed in the above table are identical in content with the temporary tags of an OB.

Please note, however, that temporary tags of the individual OBs can have different names and different data types. Also note that the call interface of each OB includes additional information regarding the date and the time of the OB request.

Bits 4 to 7 of the EV_CLASS structure element contain the event class. The following values are possible here:

- 1: Start events from standard OBs
- 2: Start events from synchronous error OBs
- 3: Start events from asynchronous error OBs

The PRIORITY structure element supplies the priority class belonging to the current OB.

Apart from these two elements, NUM is also relevant. NUM contains the number of the current OB or the startup OB that was started last.

Example

OB 80 is the OB that was called last and that has not yet been completely processed and OB 100 is the start-up OB that was started last.

The following table shows the assignment between the structure elements of the TOP_SI parameter of the "RD_SINFO" instruction and the associated local tags of OB 80.

TOP_SI Structure element	Data type	OB 80 - Associated local tag	Data type
EV_CLASS	BYTE	OB80_EV_CLASS	BYTE
EV_NUM	BYTE	OB80_FLT_ID	BYTE
PRIORITY	BYTE	OB80_PRIORITY	BYTE
NUM	BYTE	OB80_OB_NUMBR	BYTE
TYP2_3	BYTE	OB80_RESERVED_1	BYTE
TYP1	BYTE	OB80_RESERVED_2	BYTE
ZI1	WORD	OB80_ERROR_INFO	WORD
ZI2_3	DWORD	OB80_ERR_EV_CLASS	BYTE
		OB80_ERR_EV_NUM	BYTE
		OB80_OB_PRIORITY	BYTE
		OB80_OB_NUM	BYTE

The following table shows the assignment between the structure elements of the START_UP_SI parameter of the "RD_SINFO" instruction and the associated local tags of OB 100.

START_UP_SI Structure element	Data type	OB 100 - Local tag	Data type
EV_CLASS	BYTE	OB100_EV_CLASS	BYTE
EV_NUM	BYTE	OB100_STRTUP	BYTE
PRIORITY	BYTE	OB100_PRIORITY	BYTE
NUM	BYTE	OB100_OB_NUMBR	BYTE
TYP2_3	BYTE	OB100_RESERVED_1	BYTE
TYP1	BYTE	OB100_RESERVED_2	BYTE
ZI1	WORD	OB100_STOP	WORD
ZI2_3	DWORD	OB100_STRT_INFO	DWORD

RDSYSST: Read system status list

Description

With the instruction you read a partial list of the system status lists (SZL) or a SZL partial list extract.

You start the read process by assigning the value "1" to the input parameter REQ when "RDSYSST" is called. If the read process could be executed immediately, the instruction returns the value "0" at the output parameter BUSY. If BUSY has the value "1", reading is not yet complete.

Note

If you call the "RDSYSST" instruction in the diagnostic interrupt OB with the SZL-IDW#16#00B1 or W#16#00B2 or W#16#00B3 and access the module that initiated the diagnostic interrupt, the system status will be read immediately.

With "RDSYSST", only complete data records will be transferred.

System resources

If you start several asynchronous reading processes (the jobs with parameter SZL_IDW#16#00B4 and W#16#4C91 and W#16#4092 and W#16#4292 and W#16#4692 and possibly W#16#00B1 and W#16#00B3) one after the other at brief intervals, the operating system ensures that all the read jobs will be executed and that they do not interfere with each other. If the limits of the system resources are reached, this is indicated in RET_VAL. You can remedy this temporary error situation by repeating the job.

The maximum number of "simultaneously" active jobs of the instruction depends on the CPU.

Note

With S7-400 CPUs, the "RDSYSST" can be used to transfer a maximum of 432 bytes to the destination area.

Parameters

The following table shows the parameters of the "RDSYSST" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = 1: Trigger processing
SZL_ID	Input	WORD	I, Q, M, D, L or constant	SZL-ID of the partial list or partial list extract
INDEX	Input	WORD	I, Q, M, D, L or constant	Type or number of an object in a partial list.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs during execution of the instruction, the RET_VAL parameter contains an error code.
BUSY	Output	BOOL	I, Q, M, D, L	TRUE: Reading not yet complete.
SZL_HEADER	Output	STRUCT	D, L	See below.
DR	Output	ANY	I, Q, M, L, D	Destination area of the read SZL partial list or the read SZL partial list extract: • If you have only read out the header information of an SZL partial list, you should not evaluate DR but only SZL_HEADER. • Otherwise, the product of LENTHDR and N_DR indicates how many bytes were entered in DR.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameters SZL_HEADER

The SZL_HEADER parameter is a structure defined as follows:

```
SZL_HEADER:   STRUCT
    LENTHDR:  WORD
    N_DR:     WORD
END_STRUCT
```

LENTHDR is the length of a data record of the SZL partial list or the SZL partial list extract.

- If you have only read out the header information of an SZL partial list, N_DR contains the number of data records belonging to it.
- Otherwise, N_DR contains the number of data records transferred to the destination area.

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error
0081	Result field too short. (Nevertheless as many data records as possible are supplied. The SZL-Header indicates this number.)
7000	First call with REQ = 0: No data transfer active; BUSY has the value "0".
7001	First call with REQ=1: Data transfer triggered; BUSY has the value "1".
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".
8081	Result field too short (not enough space for one data record).
8082	SZL_ID is incorrect or is unknown in the CPU or instruction.
8083	INDEX incorrect or not permitted
8085	Due to a problem in the system, information is not currently available (for example, due to a lack of resources).
8086	The data record cannot be read due to a system error (bus, modules, operating system).
8087	Data record cannot be read because the module does not exist or does not acknowledge.
8088	Data record cannot be read because the actual module identifier is different from the expected module identifier.
8089	Data record cannot be read because the module is not capable of diagnostics or the data record is not supported.
80A2	DP protocol error (layer 2 error) (temporary error)
80A3	DP protocol error with user interface/user (temporary error)
80A4	Communication problem on communication bus (error occurs between the CPU and the external DP interface module)
80C5	Distributed I/O not available (temporary error).
80C6	Data record transfer stopped due to priority class abort (restart or background)
80D2	Data record cannot be read because the module is not capable of diagnostics.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

Parameters SZL_ID

Note

Partial lists of S7-CPUs that can be read out

- The partial lists for S7-300 that can be read out via "RDSYSST" are provided in the corresponding operation list.
- The partial lists for S7-400 that can be read out via "RDSYSST" are provided in the following table.

SZL_ID (W#16#...)	Partial list	INDEX (W#16#...)
Module ID		
0111 (Page 6692)	An identification data record	
	ID of the module	0001
	Identification of basic hardware	0006
	Identification of the basic firmware	0007
CPU characteristics		
0012 (Page 6693)	All characteristics	irrelevant
0112 (Page 6693)	Characteristics of one group	
	MC7 processing unit	0000
	Time system	0100
	System behavior	0200
	MC7 language description	0300
	Availability of the "C_DIAG (Page 6743)" instruction	0400
0F12 (Page 6693)	Only SZL partial header information	irrelevant
User memory areas		
0113 (Page 6695)	A data record for the memory area specified	
	Work memory	0001
System areas		
0014 (Page 6696)	Data records of all system areas	irrelevant
0F14 (Page 6696)	Only SZL partial header information	irrelevant
Block types		
0015 (Page 6697)	Data records of all block types	irrelevant
Identification of one component		
001C (Page 6698)	Identification of all components	irrelevant
011C (Page 6698)	Identification of one component	
	Name of the automation system	0001
	Name of the module	0002

SZL_ID (W#16#...)	Partial list	INDEX (W#16#...)
	System ID of the module	0003
	Copyright entry	0004
	Serial number of the module	0005
	Module type name	0007
	Serial number of the memory card	0008
	Manufacturer and profile of a CPU module	0009
	Location designation of a module	000B
0F1C (Page 6698)	Only SZL partial header information	irrelevant
Interrupt status		
0222 (Page 6704)	Data record for indicated interrupt	OB number
Assignment of process images to CPUs		
0025 (Page 6707)	Assignment of all process images to OBs	irrelevant
0125 (Page 6707)	Assignment of a process image partition to the corresponding OB	Process image partition no.
0225 (Page 6707)	Assignment of an OB to the corresponding process image partitions	OB no.
0F25 (Page 6707)	Only SZL partial header information	irrelevant
Communication status data		
0132 (Page 6709)	Status data for one communication unit	
	Diagnostics	0005
	Time system	0008
0232 (Page 6709)	Status data for one communication unit	
	CPU protection level and operator control settings	0004
Status of the module LEDs (cannot be read out for all CPUs).		
0174 (Page 6715)	Status of an LED	LED ID
DP master system information		
0090 (Page 6717)	Information regarding all DP master systems known to the CPU	0000
0190 (Page 6717)	Information regarding one DP master system	DP master system ID
0F90 (Page 6717)	Only SZL partial header information	0000
Module status information (a maximum of 27 data records is supplied)		
0091 (Page 6718)	Status information of all modules/sub-modules inserted	irrelevant
0191 (Page 6718)	Status information of all non-deactivated modules/racks with incorrect module ID	irrelevant

SZL_ID (W#16#...)	Partial list	INDEX (W#16#...)
0291 (Page 6718)	Module status information of all faulty and non-deactivated modules	irrelevant
0391 (Page 6718)	Module status information of all unavailable modules	irrelevant
0591 (Page 6718)	Module status information of all sub-modules of the host module	irrelevant
0991 (Page 6718)	Module status information of one DP master system	DP master system ID
0C91 (Page 6718)	Module status information of a module in a central configuration or connected to an integrated DP interface module or a PROFINET interface module (integrated or external)	Logical base address
4C91 (Page 6718)	Module status information of a module connected to an external DP interface module	Logical base address
0D91 (Page 6718)	Module status information of all modules in the rack/ station specified (DP or PROFINET)	Rack or DP master system ID and station number or station number and the last two digits of the PNIO subsystem ID
0E91 (Page 6718)	Module status information of all assigned modules	irrelevant
0F91 (Page 6718)	Only SZL partial header information	irrelevant
Rack/station status information		
0092 (Page 6723)	Expected status of the rack in the central configuration/ stations of a DP master system.	0 / DP master system ID
4092 (Page 6723)	Expected status of the stations of a DP master system connected via an external DP interface module.	DP master system ID
0192 (Page 6723)	Activation status of the stations of a DP master system that is connected via an integrated DP interface module	DP master system ID
0292 (Page 6723)	Current status of the rack in the central configuration / stations of a DP master system	0 / DP master system ID
4292 (Page 6723)	Current status of the stations of a DP master system that is connected via an external DP interface module	DP master system ID
0392 (Page 6723)	Status of the backup batteries in a rack/module rack of a CPU after at least one battery has failed.	0
0492 (Page 6723)	Status of the overall battery backup of all racks/module racks of a CPU.	0
0592 (Page 6723)	Status of the 24 V power supply to all racks/module racks of a CPU	0
0692 (Page 6723)	Diagnostics status of the expansion racks in the central configuration/ of the stations of a DP master system connected via an integrated DP interface module	0 / DP master system ID
4692 (Page 6723)	Diagnostics status of the stations of a DP master system connected via an external DP interface module	DP master system ID
Rack/station status information		

SZL_ID (W#16#...)	Partial list	INDEX (W#16#...)
0094 (Page 6726)	Expected status of the rack in the central configuration / stations of an IO controller system.	0 / PNIO subsystem ID
0194 (Page 6726)	Activation status of a station of an IO controller system that is configured and deactivated	PNIO subsystem ID
0294 (Page 6726)	Actual status of the rack in the central configuration / stations of an IO controller system	0 / PNIO subsystem ID
0694 (Page 6726)	Diagnostic status of the expansion units in the central configuration / stations of an IO controller system.	0 / PNIO subsystem ID
0794 (Page 6726)	Maintenance status of the central rack / stations of an IO controller system.	0 / PNIO subsystem ID
0F94 (Page 6726)	Only header information	-
Extended DP master system information		
0195 (Page 6728)	Extended information about a DP master system	DP master system ID
0F95 (Page 6728)	Only SZL partial header information	0000
Module status information PROFINET IO and PROFIBUS DP		
0696 (Page 6729)	Module status information of all sub-modules of a specified module (only with PROFINET IO on an integrated interface module).	Address with I/O ID
0C96 (Page 6729)	Module status information of a module/sub-module in a central configuration or on an integrated PROFIBUS DP or PROFINET interface module (integrated or external).	Start address with I/O ID
Diagnostics buffer (a maximum of 21 data records is supplied)		
00A0 (Page 6735)	All entries that can be supplied in the currently active operating mode	irrelevant
01A0 (Page 6735)	The most recent entries, the number is specified in the index	Number
0FA0 (Page 6735)	Only SZL partial header information	irrelevant
Diagnostic data on modules		
00B1 (Page 6736)	The first four diagnostics bytes of one module (data record 0)	Logical base address
00B2 (Page 6737)	All diagnostic data of one module (220 bytes, data record 1) (no DP module)	Rack, slot
00B3 (Page 6738)	All diagnostic data of one module (220 bytes, data record 1)	Logical base address
00B4 (Page 6738)	Diagnostic data of a DP slave	Configured diagnostics address

System status list

Overview of the system status list (SZL)

Definition: System status list

The system status list (SZL) describes the current status of an automation system. The contents of the SZL (system status list) can only be read using information functions, contents cannot be modified. The partial lists are virtual lists, in other words, they are only created by the operating system of the CPUs when specifically requested.

You can only read out one partial list using "RDSYSST (Page 6681)". This chapter describes the partial lists of the system status list that can be read with "RDSYSST (Page 6681)" in the user program, and that have information relating to

- CPUs or to
- modules whose partial lists are not module-specific (for example, SZL IDs W#16#00B1, W#16#00B2, W#16#00B3).

Module-specific partial lists, for example, for CPs and FMs are included in the descriptions of the particular modules.

Contents

The system status lists contain information about the following:

- System data
- Module status data in the CPU
- Diagnostics data on modules
- Diagnostics buffer

System data

System data are fixed or parameter- assigned characteristic data of a CPU. They provide information about the following:

- The configuration of the CPU
- The status of the priority classes
- Communication

Module status data

Module status information describes the current status of the components monitored by the system diagnostics.

Diagnostics data on modules

The modules with diagnostics capabilities assigned to a CPU have diagnostics data that are stored directly on the module.

Diagnostics buffer

The diagnostics buffer contains diagnostics entries in the order in which they occur.

Structure of a partial SZL list

Basics

You can read a partial list or a partial list extract using "RDSYSST (Page 6681)". You specify what you want to read using the parameters SZL_ID (System status list ID) and INDEX.

Structure

A partial list consists of the following:

- A header and
- the data records.

Header

The header of a partial list consists of the following:

- SZL-ID
- INDEX
- Length of a data record of the partial list in bytes
- Number of data records contained in the partial list.

Index

With certain partial lists or partial list extracts an object type ID or an object number must be specified. The index is used for this purpose. If it is not required for the information, its contents are irrelevant.

Data records

A data record in a partial list has a specific length. This depends on the information in the partial list. How the data words in a data record are used also depends on the particular partial list.

SZL-ID

SZL-ID (System status list ID)

Every partial list within the system status list has a number. You can output a complete partial list or an extract from it. The possible partial list extracts are predefined and are identified by a number. The SZL-ID consists of the number of the partial list, the number of the partial list extract and the module class.

Structure

The SZL-ID is one word long. The meaning of the bits in the SZL ID is as follows:

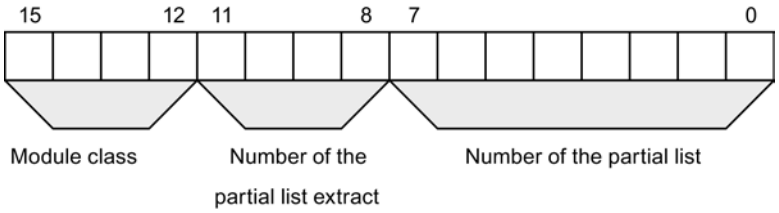


Figure 9-6 Structure of SZL-ID

Module class

Examples of module classes:

Module class	Coding (binary)
CPU	0000
IM	0100
FM	1000
CP	1100

Number of the partial list extract

The numbers of the partial list extracts and their meaning depend on the particular system status list to which they belong. With the number of the partial list extract, you specify which subset of a partial list you want to read.

Number of the partial List

Using the number of the partial list, you specify which partial list of the system status list you want to read out.

Possible SZL partial lists

Subset

Any one module only has a subset of all the possible partial lists. Which partial lists are available depends on the particular module.

Possible SZL partial lists

The following table lists all possible partial lists with the number contained in the SZL-ID (system status list ID).

Partial list	SZL-ID
Module ID	W#16#xy11 (Page 6692)
CPU characteristics	W#16#xy12 (Page 6693)
User memory areas	W#16#xy13 (Page 6695)
System areas	W#16#xy14 (Page 6696)
Block types	W#16#xy15 (Page 6697)
Identification of one component	W#16#xy1C (Page 6698)
Interrupt status	W#16#xy22 (Page 6704)
Assignment of partial process images and OBs	W#16#xy25 (Page 6707)
Communication: Status data	W#16#xy32 (Page 6709)
Status of the module LEDs	W#16#xy74 (Page 6715)
DP master system information	W#16#xy90 (Page 6717)
Module status information	W#16#xy91 (Page 6718)
Rack /station status information	W#16#xy92 (Page 6723)
Rack /station status information	W#16#xy94 (Page 6726)
Extended DP master system information	W#16#xy95 (Page 6728)
Module status information PROFINET IO and PROFIBUS DP	W#16#xy96 (Page 6729)
Diagnostics buffer	W#16#xyA0 (Page 6735)

Partial list	SZL-ID
Module diagnostic information (data record 0)	W#16#00B1 (Page 6736)
Module diagnostic information (data record 1), physical address	W#16#00B2 (Page 6737)
Module diagnostic information (data record 1), logical address	W#16#00B3 (Page 6738)
Diagnostic data of a DP slave	W#16#00B4 (Page 6738)

SZL-ID W#16#xy11 - module identification

Purpose

You obtain the module identification of this module via the partial list with the SZL-ID (system status list ID) W#16#xy11.

Header

The header of the SZL with the SZL-IDW#16#xy11 is structured as follows:

Contents	Meaning
SZL-ID	The SZL-ID of the partial list extract W#16#0111: A single identification data record
INDEX	Number of a particular data record W#16#0001: ID of the module W#16#0006: Identification of the basic hardware W#16#0007: Identification of the basic firmware
LENTHDR	W#16#001C: One data record is 14 words long (28 bytes)
N_DR	Number of data records

Data record

A data record of the SZL with the SZL-IDW#16#xy11 has the following structure:

Name	Length	Meaning
Index	1 word	Number of an identification data record
MRPC	20 bytes	Machine-Readable Product Code With INDEX W#16#0007: Reserved With INDEXW#16#0001 and W#16#0006: Machine readable product code of the module; string consists of 19 characters and a blank (20H); such as for CPU 314: "6ES7 314-0AE01-0AB0"
ModuleType	1 word	Reserved

Name	Length	Meaning
OutMod1	1 word	With INDEXW#16#0001: Output status of the module With INDEXW#16#0006 and W#16#0007: "V" and first number of the version ID
OutMod2	1 word	With INDEXW#16#0001: Reserved With INDEXW#16#0006 and W#16#0007: remaining numbers of the version ID

SZL-ID W#16#xy12 - CPU characteristics

Purpose

CPU modules have different characteristics depending on the hardware being used. An ID is assigned to each of these characteristics. You obtain the characteristics of this module via the partial list with the SZL-ID (system status list ID) W#16#xy12.

Header

The header of the partial list with the SZL-IDW#16#xy12 is structured as follows:

Contents	Meaning
SZL-ID	The SZL-ID of the partial list extract W#16#0012: All characteristics W#16#0112: Characteristics of a group; you specify the group in the INDEX parameter. W#16#0F12: Only SZL partial header information
INDEX	Group W#16#0000: MC7 processing unit W#16#0100: Time system W#16#0200: System behavior W#16#0300: MC7 language description of the CPU W#16#0400: Availability of the "C_DIAG (Page 6743)" instruction
LENTHDR	W#16#0002: One data record is one word long (2 bytes)
N_DR	Number of data records

Data record

A data record of the partial list with the SZL-IDW#16#xy12 is one word long. An identifier is entered for each characteristic. A characteristic identifier is one word long.

Note

All data records relevant to your CPU will be output. They follow each other in sequence with no gaps.

Characteristic identifier

The following table lists all the characteristic identifiers.

Identifier	Meaning
W#16#0000 - 00FF	MC7 processing unit (group with index 0000)
W#16#0001	MC7 processing generating code
W#16#0002	MC7 interpreter
W#16#0100 - 01FF	Time system (group with index 0100)
W#16#0101	1 ms resolution
W#16#0102	10 ms resolution
W#16#0103	No real time clock
W#16#0104	BCD time-of-day format
W#16#0105	All time-of-day functions (set time-of-day, set and read time-of-day, time-of-day synchronization: time-of-day slave and time-of-day master)
W#16#0106	OB_RT is available
W#16#0200 - 02FF	System response (group with index 0200)
W#16#0201	Capable of multiprocessor mode
W#16#0202	Cold restart, warm restart and hot restart possible
W#16#0203	Cold restart and hot restart possible
W#16#0204	Warm restart and hot restart possible
W#16#0205	Only warm restart possible
W#16#0206	New distributed I/O configuration is possible during RUN by using predefined resources
W#16#0208	For taking motion control functionality into account
W#16#0300 - 03FF	MC7 language description of the CPU (group with index 0300)
W#16#0301	Reserved
W#16#0302	All 32 bit fixed-point instructions
W#16#0303	All floating-point instructions
W#16#0304	sin, asin, cos, acos, tan, atan, sqr, sqrt, ln, exp
W#16#0305	Accumulator 3/accumulator 4 with associated commands (ENT, PUSH, POP, LEAVE)
W#16#0306	Master control relay instructions
W#16#0307	Address register 1 exists with corresponding instructions
W#16#0308	Address register 2 exists with corresponding instructions
W#16#0309	Operations for area-overlapping addressing
W#16#030A	Operations for area-internal addressing
W#16#030B	All memory-indirect addressing instructions for bit memory (M)
W#16#030C	All memory-indirect addressing instructions for data blocks (DB)
W#16#030D	All memory-indirect addressing instructions for DI
W#16#030E	All memory-indirect addressing instructions for local data
W#16#030F	All instructions for parameter transfer in instructions
W#16#0310	Memory bit edge instructions for I
W#16#0311	Memory bit edge instructions for Q
W#16#0312	Memory bit edge instructions for bit memory

Identifier	Meaning
W#16#0313	Memory bit edge instructions for data blocks (DB)
W#16#0314	Memory bit edge instructions for DI
W#16#0315	Memory bit edge instructions for local data (LD)
W#16#0316	Dynamic evaluation of the ERAB bit
W#16#0317	Dynamic local data area with the corresponding instructions
W#16#0318	Reserved
W#16#0319	Reserved
W#16#0401	Reserved
W#16#0402	Reserved
W#16#0403	"C_DIAG (Page 6743)" is available

SZL-ID W#16#xy13 - user memory areas

Purpose

You obtain information on the memory areas of the module via the partial list with the SZL-ID (system status list ID) W#16#xy13.

Header

The header of the partial list with the SZL-IDW#16#xy13 is structured as follows:

Contents	Meaning
SZL-ID	The SZL-ID of the partial list extract W#16#0113: Data record for a memory area You use the INDEX parameter to specify the memory area.
INDEX	Specification of a memory area (only with SZL-IDW#16#0113) W#16#0001: Work memory
LENTHDR	W#16#0024: One data record is 18 words long (36 bytes)
N_DR	Number of data records

Data record

A data record of the partial list with the SZL-IDW#16#xy13 has the following structure:

Name	Length	Meaning
Index	1 word	Index of a memory area W#16#0001: Work memory
Code	1 word	Memory type: • W#16#0001: Volatile memory (RAM) • W#16#0002: Non-volatile memory (FEPROM) • W#16#0003: Mixed memory (RAM and FEPROM)

Name	Length	Meaning
Size	2 words	Total size of the selected memory (total of Area1 and Area2)
Mode	1 word	Logical mode of the memory • Bit 0: volatile memory area • Bit 1: non-volatile memory area • Bit 2: mixed memory area for work memory • Bit 3: code and data separated • Bit 4: code and data together
Granularity	1 word	Always has the value "0"
Area1	2 words	Size of the volatile memory area in bytes.
Reserved1	2 words	Size of the volatile memory area being used
Block1	2 words	Largest free block in the volatile memory area If "0": no information available or cannot be ascertained.
Area2	2 words	Size of the non-volatile memory area in bytes
Reserved2	2 words	Size of the non-volatile memory area being used
Block2	2 words	Largest free block in the volatile memory area If "0": no information available or cannot be ascertained.

SZL-ID W#16#xy14 - system areas

Purpose

You obtain information on the system areas of the module via the partial list with the SZL-ID (system status list ID) W#16#xy14.

Header

The header of the partial list with the SZL-IDW#16#xy14 is structured as follows:

Contents	Meaning
SZL-ID	The SZL-ID of the partial list extract W#16#0014: All system areas of a module W#16#0F14: Only SZL partial header information
INDEX	irrelevant
LENTHDR	W#16#0008: One data record is 4 words long (8 bytes)
N_DR	Number of data records You must at least assign a number of 9 data records. If you select a destination area that is too small, "RDSYSST (Page 6681)" does not provide a data record.

Data record

A data record of the partial list with the SZL-IDW#16#xy14 has the following structure:

Name	Length	Meaning
Index	1 word	Index of the system area • W#16#0001: Process image input (number in bytes) • W#16#0002: Process output image (number in bytes) • W#16#0003: Bit memory (number in bits) **Note:** This index is only provided by the CPU, where the number of flags can be shown in one word. If your CPU does not provide this value, you must evaluate the index W#16#0008 . • W#16#0004: Times (number) • W#16#0005: Counters (number) • W#16#0006: Number of bytes in the logical address area • W#16#0007: Local data (total local data area of the CPU in bytes) **Note:** This index is only provided by CPUs, where the length of the entire local data area can be shown in one word. If your CPU does not provide this value, you must evaluate the index W#16#0009. • W#16#0008: Bit memory (number in bytes) • W#16#0009: Local data (total local data area of the CPU in kilobytes)
code	1 word	Memory type • W#16#0001: Volatile memory (RAM) • W#16#0002: Non-volatile memory (FEPROM) • W#16#0003: Mixed memory (RAM and FEPROM)
number	1 word	Number of elements of the system area
retentive	1 word	Number of retentive elements

SZL-ID W#16#xy15 - block types

Purpose

If you read the partial list with the SZL-ID (system status list ID) W#16#xy15, you obtain the block types that exist on the module.

Header

The header of the partial list with the SZL-IDW#16#xy15 is structured as follows:

Contents	Meaning
SZL-ID	The SZL-ID of the partial list extract W#16#0015: Data records of all block types of a module
INDEX	irrelevant

Contents	Meaning
LENTHDR	W#16#000A: One data record is 5 words long (10 bytes)
N_DR	Number of data records

Data record

A data record of the partial list with the SZL-IDW#16#xy15 has the following structure:

Name	Length	Meaning
Index	1 word	Block type number • W#16#0800: OB • W#16#0A00: DB • W#16#0B00: SDB • W#16#0C00: FC • W#16#0E00: FB
MaxNumber	1 word	Maximum number of blocks of the type • Foi OBs: max. possible number of OBs for a CPU • For DBs: max. possible number of DBs including DB0 • For SDBs: max. possible number of SDBs including SDB2 • For FCs and FBs: max. possible number of loadable blocks
MaxLenght	1 word	Maximum total size of the object to be loaded in kilobytes
MaxWorkMem	2 words	Maximum length of the work memory part of a block in bytes

SZL-ID W#16#xy1C - component identification

Purpose

If you read the partial list with the SZL-ID (system status list ID) W#16#xy1C, you can identify the CPU or the automation system.

Header

The header of the partial list with the SZL-IDW#16#xy1C is structured as follows:

Contents	Meaning
SZL-ID	The SZL-ID of the partial list extract
	• W#16#001C: Identification of all components • W#16#011C: Identification of one component • W#16#0F1C: Only SZL partial header information

Contents	Meaning
INDEX	• Identification of components for a partial list extract with the SZL-IDW#16#011C and W#16#031C – W#16#0001: Name of the automation system – W#16#0002: Name of the module – W#16#0003: System ID of the module – W#16#0004: Copyright entry – W#16#0005: Serial number of the module – W#16#0007: Module type name – W#16#0008: Serial number of the memory card No data record is supplied for modules in which a memory card cannot be inserted. – W#16#0009: Manufacturer and profile of a CPU module – W#16#000A: OEM ID of a module (S7-300 only) – W#16#000B: Location designation of a module – W#16#000C: Serial number of Sync module 1 – W#16#000D: Serial number of Sync module 2 • Rack no. for the partial list extract with the SZL-IDW#16#021C (Byte0: Rack no., byte1: B#16#00)
LENTHDR	W#16#0022: One data record is 17 words long (34 bytes)
N_DR	Number of data records

Data record

A data record of the partial list with the SZL-IDW#16#xy1C has the following structure:

INDEX = W#16#0001

Name	Length	Meaning
index	1 word	• For a standard CPU and the partial list extract W#16#011C: component ID: W#16#0001 • For the partial list extracts W#16#021C and W#16#031C: Byte 0: – Bits 0 to 2: Rack no.; – Bit3: 0 = Reserve CPU, 1 = Master CPU; – Bits 4 to 7: 1111 Byte 1: component ID: B#16#01
name	12 words	Name of the automation system (max. 24 characters; when using shorter names, the gaps are filled with B#16#00)
res	4 words	Reserved

INDEX = W#16#0002

Name	Length	Meaning
index	1 word	• For a standard CPU and the partial list extract W#16#011C: component ID: W#16#0002 • For the partial list extracts W#16#021C and W#16#031C: Byte 0: – Bits 0 to 2: Rack no.; – Bit3: 0 = Reserve CPU, 1 = Master CPU; – Bits 4 to 7: 1111 Byte 1: component ID: B#16#02
name	12 words	Name of the module (max. 24 characters; when using shorter names, the gaps are filled with B#16#00)
res	4 words	Reserved

INDEX = W#16#0003

Name	Length	Meaning
index	1 word	• For a standard CPU and the partial list extract W#16#011C: component ID: W#16#0003 • For the partial list extracts W#16#021C and W#16#031C: Byte 0: – Bits 0 to 2: Rack no.; – Bit 3: 0 = Reserve CPU, 1 = Master CPU; – Bits 4 to 7: 1111 Byte 1: component ID: B#16#03
tag	16 words	Plant identification of the module (max. 32 characters; when using shorter system IDs the gaps will be filled with B#16#00)

INDEX = W#16#0004

Name	Length	Meaning
index	1 word	• For a standard CPU and the partial list extract W#16#011C: component ID: W#16#0004 • For the partial list extracts W#16#021C and W#16#031C: Byte 0: – Bits 0 to 2: Rack no.; – Bit 3: 0 = Reserve CPU, 1 = Master CPU; – Bits 4 to 7: 1111 Byte 1: component ID: B#16#04
copyright	13 words	Constant character sequence "Original Siemens Equipment"
res	3 words	Reserved

INDEX = W#16#0005

Name	Length	Meaning
index	1 word	• For a standard CPU and the partial list extract W#16#011C: component ID: W#16#0005 • For the partial list extracts W#16#021C and W#16#031C: Byte 0: – Bits 0 to 2: Rack no.; – Bit 3: 0 = Reserve CPU, 1 = Master CPU; – Bits 4 to 7: 1111 Byte 1: component ID: B#16#05
serialn	12 words	Serial number of the module; character string with max. length of 24 characters. Shorter names are filled with B#16#00. Note: The serial numbers for SIMATIC components are unique worldwide. It is permanently affixed to the CPU hardware, in other words it remains unchanged when the firmware is updated.
res	4 words	Reserved

INDEX = W#16#0007

Name	Length	Meaning
index	1 word	• For a standard CPU and the partial list extract W#16#011C: component ID: W#16#0007 • For the partial list extracts W#16#021C and W#16#031C: Byte 0: – Bits 0 to 2: Rack no.; – Bit 3: 0 = Reserve CPU, 1 = Master CPU; – Bits 4 to 7: 1111 Byte 1: component ID: B#16#07
cputypname	16 words	Module type name; character string with a max. length of 32 characters. Shorter names are filled with B#16#00.

INDEX = W#16#0008

Name	Length	Meaning
index	1 word	• For a standard CPU and the partial list extract W#16#011C: component ID: W#16#0008 • For the partial list extracts W#16#011C, W#16#021C and W#16#031C: Byte 0: – Bits 0 to 2: Rack no.; – Bit 3: 0 = Reserve CPU, 1 = Master CPU; – Bits 4 to 7: 1111 Byte 1: component ID: B#16#08
sn_mc/mmc	16 words	Serial number of the Memory Card/Micro Memory Card; character string with a max. length of 32 characters. Shorter names are filled with B#16#00. • Siemens serial number: only serial number, no index • Product serial number (PSN) of an S7 Micro Memory Card: "MMC" plus serial number (PSN) • Serial number of an S7 Memory Card: "MC" plus serial number The character string ends immediately after "MMC" or "MC" if no Memory Card is installed.

INDEX = W#16#0009

Name	Length	Meaning
index	1 word	• For a standard CPU and the partial list extract W#16#011C: component ID: W#16#0009 • For the partial list extracts W#16#021C and W#16#031C: Byte 0: – Bits 0 to 2: Rack no.; – Bit 3: 0 = Reserve CPU, 1 = Master CPU; – Bits 4 to 7: 1111 Byte 1: component ID: B#16#09
manufacturer_id	1 word	See PROFIBUS Profile Guidelines Part 1, Identification & Maintenance Functions
profile_id	1 word	See PROFIBUS Profile Guidelines Part 1, Identification & Maintenance Functions
profile_specific_typ	1 word	See PROFIBUS Profile Guidelines Part 1, Identification & Maintenance Functions
res	13 words	Reserved

INDEX = W#16#000A

Name	Length	Meaning
index	1 word	• For a standard CPU and the partial list extract W#16#011C: component ID: W#16#000A • For the partial list extracts: W#16#021C and W#16#031C: Byte0: – Bits 0 to 2: Rack no.; – Bit 3: 0 = Reserve CPU, 1 = Master CPU; – Bits 4 to 7: 1111 Byte 1: component ID: B#16#0A
oem_copyright_string	13 words	OEM copyright ID; character string with a max. length of 20 characters. Shorter names are filled with B#16#00.
oem_id	1 word	OEM-ID. This is assigned by Siemens.
oem_add_id	2 words	OEM-ID supplemental ID. This can be assigned by the user.

INDEX = W#16#000B

Name	Length	Meaning
index	1 word	• For a standard CPU and the partial list extract W#16#011C: component ID: W#16#000B • For the partial list extracts: W#16#021C and W#16#031C: Byte0: – Bits 0 to 2: Rack no.; – Bit 3: 0 = Reserve CPU, 1 = Master CPU; – Bits 4 to 7: 1111 Byte 1: component ID: B#16#0B
loc_id	16 words	Location designation; character string with a max. length of 32 characters. Shorter names are filled with B#16#00.

INDEX = W#16#000C

Name	Length	Meaning
index	1 word	• Byte0: – Bits 0 to 2: Rack no.; – Bit 3: 0 = Reserve CPU, 1 = Master CPU; – Bits 4 to 7: 1111 • Byte 1: Component ID: B#16#0C
mlfb	5 words	10th to 19th character of the order number of Sync module 1
fill	1 word	Four spaces (ASCII)
es	1 word	Product version (ASCII), e.g. "01"
fill	1 byte	Two spaces (ASCII)
vendor_sn	17 bytes	Serial number

Note

If Sync module 1 is not available or cannot be identified, the data record will be filled in accordance with the "index" tag with B#16#00.

INDEX = W#16#000D

Name	Length	Meaning
index	1 word	• Byte0: – Bits 0 to 2: Rack no.; – Bit 3: 0 = Reserve CPU, 1 = Master CPU; – Bits 4 to 7: 1111 • Byte 1: Component ID: B#16#0D
mlfb	5 words	10th to 19th character of the order number of Sync module 2
fill	1 word	Four spaces (ASCII)
es	1 word	Product version (ASCII), e.g. "01"
fill	1 byte	Two spaces (ASCII)
vendor_sn	17 bytes	Serial number

Note

If Sync module 2 is not available or cannot be identified, the data record will be filled in accordance with the "index" tag with B#16#00.

SZL-ID W#16#xy22 - interrupt status

Purpose

The partial list with the SZL-ID (system status list ID) W#16#xy22 contains information about the current status of interrupt processing and interrupt generation in the module.

Header

The header of the partial list with the SZL-IDW#16#xy22 is structured as follows:

Contents	Meaning
SZL-ID	The SZL-ID of the partial list extract W#16#0222 Data record for specified interrupt The interrupt (OB no.) is specified in the INDEX parameter.
INDEX	OB no. or interrupt class (for SZL-IDW#16#0222) W#16#0000: Free cycle W#16#000A: Time-of-day interrupt W#16#0014: Time-delay interrupt W#16#001E: Cyclic interrupt W#16#0028: Hardware interrupt W#16#0032: DP interrupt W#16#003C: Synchronous cycle interrupt W#16#0050: Asynchronous error interrupt W#16#005A: Background W#16#0064: Startup W#16#0078: Synchronous error interrupt
LENTHDR	W#16#001C: One data record is 14 words long (28 bytes)
N_DR	Number of data records

Data record

A data record of the partial list with the SZL-IDW#16#xy22 has the following structure:

Name	Length	Meaning
Info	10 words	Start info for the given OB, with following exceptions: • OB 1 provides the current minimum (in bytes 8 and 9) and maximum cycle time (in bytes 10 and 11) (time base: ms, byte count begins at 0). • When a job is active for a time-delay interrupt, bytes 8 and 11 (byte count begins at 0) get the remaining time in ms left of the delay time set as a parameter. • OB 80 contains the configured minimum (in bytes 8 and 9) and maximum cycle time (in bytes 10 and 11) (time base: ms, byte count begins at 0). • Error interrupts without the current information • Interrupts contain the status info from the current parameter settings of the interrupt source. • In the case of synchronous errors, the priority class B#16#7F is entered if the OBs have not been processed yet; otherwise, the priority class of the last call will be entered. • If an OB has several start events and these have not yet occurred at the information time, then event no. W#16#xyzz returned with: – x: Event class – zz: Smallest defined number of the group – y: undefined Otherwise, the number of the last start event that occurred is used.
AlarmStatus1	1 word	Processing identifiers: • Bit 0: Interrupt event is caused by parameters – = 0: Enabled – = 1: disabled • Bit 1: Interrupt event as per "DIS_IRT (Page 6624)" – = 0: Not disabled – = 1: disabled • Bit 2 = 1: Interrupt source is active (generate job is present for time interrupts (time-of-day interrupt OB started, time-delay interrupt OB started, cyclic interrupt OB was configured) • Bit 4: Interrupt OB – = 0: Is not loaded – = 1: Is loaded • Bit 5: Interrupt OB is by TIS – = 1: disabled • Bit 6: Entry in diagnostic buffer – = 1: disabled

Name	Length	Meaning
AlarmStatus2	1 word	Reaction with non-loaded/disabled OB • Bit 0 = 1: Disable interrupt source • Bit 1 = 1: Generate interrupt event error • Bit 2 = 1: CPU goes into STOP mode • Bit 3 = 1: Only discard interrupt
AlarmStatus3	2 words	Discarded by TIS functions: Bit no. x set means: the event number that is greater by x than the smallest event number of the affected OB is discarded by the TIS function.

SZL ID W#16#xy25 - assignment of process image partitions to OBs

Purpose

The partial list with the SZL-ID (system status list ID) W#16#xy25 shows you how process image partitions are assigned to the OBs.

This list provides information about

- Process image partitions you have assigned to specific OBs for update by the system
- Process image partitions you have assigned to specific clock synchronization interrupt OBs (OBs 61 to 64). The partial process image is updated here by calling the "SYNC_PI (Page 6455)" and "SYNC_PO (Page 6457)".
  The assignment of DP master systems to the synchronous cycle interrupt OBs is obtained via SZL W#16#xy95 (Page 6728).

Header

The header of the partial list with the SZL-IDW#16#xy25 is structured as follows:

Contents	Meaning
SZL-ID	The SZL-ID of the partial list extract • W#16#0025: Assignment of all process image partitions to the OBs in the CPU • W#16#0125: Assignment of a process image partition to the corresponding OB Specify the process image partition ID in the INDEX parameter. • W#16#0225: Assignment of the OB to the corresponding process image partitions Specify the OB number in the INDEX parameter. Note: The clock synchronization interrupt OBs (OBs 61 to 64) are the only ones you can assign to multiple process image partitions. • W#16#0F25: Only SZL partial header information
INDEX	• For SZL-ID W#16#0025: irrelevant • For SZL-ID W#16#0125: Process image partition no. • For SZL-ID W#16#0225: OB no. • For SZL-ID W#16#0F25: irrelevant

Contents	Meaning
LENTHDR	W#16#0004: One data record is 2 words long (4 bytes)
N_DR	Number of data records

Data record

A data record of the partial list extract with the SZL-IDW#16#xy25 has the following structure:

Name	Length	Meaning
PPI*_nr	1 byte	Process image partition no.
PPI*_use	1 byte	Type of assignment between process image partitions and the OB: • Bit 0 = 1: The process image partition of the inputs is assigned to the specified OB for update by the system. • Bit 1 = 1: The process image partition of the outputs is assigned to the specified OB for update by the system. • Bit 2 = 1: The process image partition input table is assigned to the specified synchronous cycle interrupt OB. It can be updated in this OB by calling "SYNC_PI (Page 6455)". • Bit 3 = 1: The process image partition output table is assigned to the specified synchronous cycle interrupt OB. It can be updated in this OB by calling "SYNC_PO (Page 6457)". • Bits 4 to 7: 0
ob_nr	1 byte	OB no.
res	1 byte	Reserved
*PPI: Parial Process Image		

Partial list extracts

- Partial list extract with SZL-ID = W#16#0025:
  The data records of all partial process images you have assigned to an OB in your configuration are returned in ascending order. For partial process images without OB assignment, ob_nr has the value zero. No data record is returned for partial process images.
- Partial list extract with SZL-ID = W#16#0125:
  A data record is returned if you have assigned the addressed process image partition to an OB in your configuration. No data record is returned if you have not assigned an OB.

Note

OB 1 is permanently assigned to partial process image 0. You always get a data record for the information about partial process image 0.

- Partial list extract with SZL-ID = W#16#0225:
  A data record is returned for each partial process image that is assigned to the addressed OB. No data record is returned if you have not assigned the addressed process image partition to an OB in your configuration.

  Note

  Clock synchronization interrupt OBs can be assigned multiple process image partitions. If this is the case, several data records will be returned.

- Partial list extract with SZL-ID = W#16#0F25:
  The maximum possible number of data records is returned as number.

Example of the meaning of data records

Call parameters of "RDSYSST (Page 6681)"	Returned variables	Explanation
SZL_ID = W#16#0125, INDEX = W#16#0008	PPI*_nr = B#16#08, PPI*_use = B#16#03, ob_nr = B#16#15	A data record is returned. The input and output process image 8 are assigned to OB 21 for system-side process image update.
SZL_ID = W#16#0125, INDEX = W#16#0009	–	No data record is returned. Thus: process image partition 9 is not assigned to any OB.
SZL_ID = W#16#0225, INDEX = W#16#003D	PPI*_nr = B#16#0A, PPI*_use = B#16#C0, ob_nr = B#16#3D PPI*_nr = B#16#10, PPI*_use = B#16#C0, ob_nr = B#16#3D	Two data records are returned. The input and output process images 10 and 16 are assigned to OB 61. These can be updated in OB 61 by calling "SYNC_PI (Page 6455)" and "SYNC_PO (Page 6457)".
SZL_ID = W#16#0225, INDEX = W#16#0001	PPI*_nr = B#16#00, PPI*_use = B#16#03, ob_nr = B#16#01	A data record is returned. The input and output process image 0 are assigned to OB 1. They are updated by the system.
*PPI: Parial Process Image		

SSL ID W#16#xy32 - Communication status data

SZL-ID W#16#xy32 - communication status data

Purpose

If you read the partial list with SZL-IDW#16#xy32 you obtain the communication status data of the module.

Header

The header of the partial list with the SZL-IDW#16#xy32 is structured as follows:

Contents	Meaning
SZL-ID	The SZL-ID of the partial list extract • W#16#0132: Status data for one communication section (always only one data record). Specify the communication section with the INDEX parameter. • W#16#0232: status data for one communication unit. Specify the communication section with the INDEX parameter.
INDEX	Communication section • For SZL-IDW#16#0132: – W#16#0005 Diagnostics – W#16#0008 Time system – W#16#000B Time system – W#16#000C Time system • For SZL-IDW#16#0232: W#16#0004 CPU protection level, operator switch settings and version IDs / checksums
LENTHDR	W#16#0028: One data record is 20 words long (40 bytes)
N_DR	Number of data records

Data record

A data record of the partial list with the SZL-IDW#16#0132 is always 20 words long. The data records have different contents. The contents depend on the INDEX parameter, in other words, on the communication section to which the data record belongs.

Data record of the partial list extract with SZL-ID W#16#0132 index W#16#0005

Contents

The partial list extract with the SZL-IDW#16#0132 and the IndexW#16#0005 contains information about the diagnostics status of the module.

Data record

A data record of the partial list extract with the SZL-IDW#16#0132 and the IndexW#16#0005 has the following structure:

Name	Length	Meaning
Index	1 word	W#16#0005: Diagnostics
ext	1 word	Extended functions • 0: No • 1: Yes

Name	Length	Meaning
send	1 word	Automatic send • 0: No • 1: Yes
possible	1 word	Sending user-defined diagnostic messages currently possible • 0: No • 1: Yes
res	16 words	Reserved

Data record of the partial list extract with SZL-ID W#16#0132 index W#16#0008

Contents

The partial list extract with the SZL-IDW#16#0132 and the IndexW#16#0008 contains information about the status of the module time system.

Data record

A data record of the partial list extract with the SZL-IDW#16#0132 and the IndexW#16#0008 has the following structure:

Name	Length	Meaning
Index	1 word	W#16#0008: Time system status
cycle	1 word	Reserved
corr	1 word	Correction factor for the time
clock 0	1 word	Runtime meter 0: time in hours
clock 1	1 word	Runtime meter 1: time in hours
clock 2	1 word	Runtime meter 2: time in hours
clock 3	1 word	Runtime meter 3: time in hours
clock 4	1 word	Runtime meter 4: time in hours
clock 5	1 word	Runtime meter 5: time in hours
clock 6	1 word	Runtime meter 6: time in hours
clock 7	1 word	Runtime meter 7: time in hours
time	4 words	Current date and time (format: date_and_time)
rtm _0	1 byte	Runtime meter Bit x: Overflow of runtime meter x, 0 ≤ x ≤ 7 (Bit = 1: run-time meter is busy)
res	1 byte	Reserved
rtmOV _0	1 byte	Runtime meter Overflow Bit x: Overflow of runtime meter x, 0 ≤ x ≤ 7 (Bit = 1: Overflow)
res	1 byte	Reserved
status	1 word	Time status (for bit assignment, see below)
res	3 bytes	Reserved
status_valid	1 byte	Validity of variable status: B#16#01: status valid

Time-of-day status (status)

Bit	Default value	Description
15	0	Sign for the correction value (0: positive, 1: negative)
14 to 10	00000	Correction value This parameter allows the basic time in the frame to be corrected to local time: `Local time = basic time ± correction value * 0.5 h` This correction takes into account the time zone and the time difference in summer time (daylight savings time) and winter time (standard time).
9	0	Reserved
8	0	Reserved
7	0	Notification hour This parameter indicates whether the next time adjustment also includes a switchover from summer (daylight savings time) to winter time (standard time) or vice versa. (0: no adjustment made, 1: adjustment made).
6	0	Summer time (daylight savings time)/winter time (standard time) indicator This parameter indicates whether the local time calculated using the correction value is summer or winter time.)0Ö winter time, 1: summer time)
5	0	Parameter not used by S7.
4 to 3	00	Time resolution This parameter indicates the resolution of the transmitted clock time. (00: 0.001 s, 01: 0.01 s, 10: 0.1 s, 11: 1 s)
2	0	Parameter not used by S7.
1	0	Parameter not used by S7.
0	0	Synchronization failure This parameter indicates whether the time transmitted in the message frame from an external time master (e.g. SICLOCK) is synchronized. (0: synchronization failed, 1: synchronization occurred) **Note:** Evaluation of this bit in a CPU is only meaningful if there is continuous external time synchronization.

Data record of the partial list extract with SZL-ID W#16#0132 index W#16#000B

Contents

The partial list extract with the SZL-IDW#16#0132 and the IndexW#16#000B contains information about the status of 32-bit runtime meter 0 to 7 of the module.

Note

In the partial list extract with the SZL-IDW#16#0132 and the IndexW#16#0008 this runtime meter is displayed as 16-bit runtime meter.
With it you can continue to implement programs that were developed for a CPU with 16-bit runtime meters, and which continue to use the partial list extract with the SZL-IDW#16#0132 and the IndexW#16#0008.

Data record

A data record of the partial list extract with the SZL-IDW#16#0132 and the index W#16#000B has the following structure:

Name	Length	Meaning
Index	1 word	W#16#000B: Time system status
rtm _0	1 byte	Runtime meter Bit x: Overflow of runtime meter x, 0 ≤ x ≤ 7 (Bit = 1: run-time meter is busy)
res	1 byte	Reserved
rtmOV _0	1 byte	Runtime meter overflow Bit x: Overflow of runtime meter x, 0 ≤ x ≤ 7 (Bit = 1: Overflow)
res	1 byte	Reserved
clock 0	2 words	Runtime meter 0: time in hours
clock 1	2 words	Runtime meter 1: time in hours
clock 2	2 words	Runtime meter 2: time in hours
clock 3	2 words	Runtime meter 3: time in hours
clock 4	2 words	Runtime meter 4: time in hours
clock 5	2 words	Runtime meter 5: time in hours
clock 6	2 words	Runtime meter 6: time in hours
clock 7	2 words	Runtime meter 7: time in hours
res	1 word	Reserved

Data record of the partial list extract with SZL-ID W#16#0132 index W#16#000C

Contents

The partial list extract with the SZL-IDW#16#0132 and the INDEXW#16#000C contains information about the status of 32-bit runtime meter 8 to 15 of the module.

Data record

A data record of the partial list extract with the SZL-IDW#16#0132 and the INDEXW#16#000C has the following structure:

Name	Length	Meaning
Index	1 word	W#16#000C: Time system status
rtm_0	1 byte	Runtime meter Bit x: Status of run-time meter (8+x) , 0 ≤ x ≤ 7 (Bit = 1: run-time meter is busy)
res	1 byte	Reserved
rtmOV _0	1 byte	Runtime meter overflow Bit x: Overflow of run-time meter (8+x) , 0 ≤ x ≤ 7 (Bit = 1: Overflow)
res	1 byte	Reserved
clock 8	2 words	Runtime meter 8: time in hours
clock 9	2 words	Runtime meter 9: time in hours
clock 10	2 words	Runtime meter 10: time in hours
clock 11	2 words	Runtime meter 11: time in hours
clock 12	2 words	Runtime meter 12: time in hours
clock 13	2 words	Runtime meter 13: time in hours
clock 14	2 words	Runtime meter 14: time in hours
clock 15	2 words	Runtime meter 15: time in hours
res	1 word	Reserved

Data record of the partial list extract with SZL-ID W#16#0232 index W#16#0004

Contents

The partial list extract with the SZL-IDW#16#0232 and the Index W#16#0004 contains information about the CPU protection level and the settings of the operator mode switch and checksums of the hardware configuration and the user program.

Data record

A data record of the partial list extract with the SZL-IDW#16#0232 and the Index W#16#0004 has the following structure:

Name	Length	Meaning
Index	1 word	• Byte 1: B#16#04: CPU protection level, operator control settings and checksums • Byte 0: Standard CPU: B#16#00 Bit 3: 0 = Reserve CPU, 1 = Master CPU Bits 4 to 7: 1111
ProtLevel_ModeSwitch	1 word	Protection level set with the mode selector (1, 2, 3)

Name	Length	Meaning
ProtLevel_Par	1 word	Protection level set in parameters (0, 1, 2, 3; 0: no password, protection level invalid).
ProtLevel_Valid	1 word	Valid protection level of the CPU
ModeSwitch	1 word	Mode selector setting (1:RUN, 2:RUN-P, 3:STOP, 4:MRES, 0:undefined or cannot be determined)
Startup_ModeSwitch	1 word	Startup switch setting (1:CRST, 2:WRST, 0:undefined, does not exist or cannot be determined)
res	1 word	Reserved
ID_Val_CheckSum	1 word	ID for the validity of the four following checksums (0: invalid)
ID_CheckSum1_HWConf	1 word	Checksum 1 of the hardware configuration (Intel format): Exclusive OR link over the lengths of all system data blocks.
ID_CheckSum2_HWConf	1 word	Checksum 2 of the hardware configuration (Intel format): Exclusive OR link over the checksums of all system data blocks.
ID_CheckSum1_UserProg	1 word	Checksum 1 of the user program (Intel format): Exclusive-or link over the lengths of the following blocks: OBs, DBs, FBs, FCs
ID_CheckSum2_UserProg	1 word	Checksum 2 of the user program (Intel format): Exclusive-or link over the checksums of the following blocks: OBs, DBs, FBs, FCs
res	2 words	Reserved
sfc_req	1 word	Protection level 2 or 3 request by the instruction "PROTECT" (1: request has been made)
sfc_act	1 word	Protection level 2 or 3 activation by the instruction "PROTECT" (1: activated)
res	4 words	Reserved

SZL-ID W#16#xy74 - status of the module LEDs

Purpose

If you read the partial list with SZL-ID W#16#xy74 with standard CPUs (if present) you obtain the status of the module LEDs.

Header

The header of the partial list with the SZL-ID W#16#xy74 is structured as follows:

Contents	Meaning
SZL-ID	The SZL-ID of the partial list extract W#16#0174 Status of an LED. You use the INDEX parameter to select the LED.
INDEX	LED ID (only relevant for SZL-IDW#16#0174) • W#16#0001: SF (group error) • W#16#0002: INTF (internal fault) • W#16#0003: EXTF (external fault) • W#16#0004: RUN • W#16#0005: STOP • W#16#0006: FRCE (Force) • W#16#0007: CRST (cold restart) • W#16#0008: BAF (battery fault/overload, short circuit of battery voltage on bus) • W#16#0009: USR (user-defined) • W#16#000A: USR1 (user-defined) • W#16#000B: BUS1F (bus fault interface 1) • W#16#000C: BUS2F (Bus error interface 2) and BUS5F (Bus error interface 5) for the CPUs 414-3 PN/DP,416-3 PN/DP and 416F-3 PN/DP • W#16#0012: IFM1F (interface fault interface module 1) • W#16#0013: IFM2F (interface fault interface module 2) • W#16#0014: BUS3F (bus fault interface 3) • W#16#0015: MAINT (maintenance request) • W#16#0016: DC24V • W#16#0017: BUS5F (bus fault interface 5) • W#16#0080: IF (init failure) • W#16#0081: UF (user failure) • W#16#0082: MF (monitoring failure) • W#16#0083: CF (communication failure) • W#16#0084: TF (task failure) • W#16#00EC: APPL_STATE_RED • W#16#00ED: APPL_STATE_GREEN
LENTHDR	W#16#0004: One data record is 2 words long (4 bytes)
N_DR	Number of data records

Data record

A data record of the partial list with the SZL-ID W#16#xy74 has the following structure:

Name	Length	Meaning
CPU_LED_ID	1 word	• Byte 0: – Standard CPU: B#16#00 – Bit 3: 0=standby CPU, 1=master CPU – Bits 4 to 7: 1111 • Byte 1: LED ID
LED_on	1 byte	Status of the LED: • 0: off • 1: on
LED_flash	1 byte	Flashing status of the LED: • 0: not flashing • 1: flashing normally (2 Hz) • 2: flashing slowly (0.5 Hz))

SZL-ID W#16#xy90 - DP master system information

Purpose

If you read the partial list SZL-IDW#16#xy90, you obtain the status information of all DP master systems known to the CPU.

Header

The header of the partial list with the SZL-IDW#16#xy90 is structured as follows:

Contents	Meaning
SZL-ID	The SZL-ID of the partial list extract W#16#0090: Information regarding all DP master systems known to the CPU W#16#0190: Information regarding one DP master system W#16#0F90: Only SZL partial header information
INDEX	• For the partial list extract with the SZL-IDW#16#0190: – Low byte: B#16#00 – High byte: DP master system ID • For the partial list extract with the SZL-IDW#16#0090 and W#16#0F90: – W#16#0000
LENTHDR	W#16#000E: One data record is 7 words long (14 bytes)
N_DR	Number of data records For the partial list extract with the SZL-IDW#16#0190: 0 to 1 For the partial list extract with the SZL-IDW#16#0090: • For a standard CPU: 0 to 14

Data record

A data record of the partial list with the IDW#16#xy90 has the following structure:

Name	Length	Meaning	
dp_m_id	1 byte	DP master system ID	
dp_m_rack	1 byte	Rack number of the DP master • For a standard CPU: 0	
dp_m_slot	1 byte	Slot of the DP master or slot of the CPU (with integrated DP interface)	
dp_m_SubModule	1 byte	• with integrated DP interface: interface number of the DP master: – 1: X2 – 2: X1 – 3: IF1 – 4: IF2 • with external DP interface: 0	
logadr	1 word	logic start address of the DP master	
res	1 word	Reserved	
res	1 word	Reserved	
dp_m_state	1 byte	additional properties of the DP master system	
		Bit 0:	DP mode • 0: S7-compatible • 1: DPV1
		Bit 1:	DP cycle • 0: not constant bus cycle • 1: constant bus cycle
		Bit 2 to 6:	• Reserved
		Bit 7:	DP master type • 0: integrated DP master • 1: external DP master
res	3 bytes	Reserved	

SZL-ID W#16#xy91 - module status information

Purpose

If you read the partial list with SZL-ID W#16#xy91, you obtain the status information of modules assigned to the CPU.

Note

An evaluation of the SZL-ID W#16#xy91 is not possible for a module with packed addresses (ET 200S).

Header

The header of the partial list with the SZL-ID W#16#xy91 is structured as follows:

Contents	Meaning
SZL-ID	The SZL-ID of the partial list extract • W#16#0091 Module status information of all inserted and configured modules/sub-modules of the CPU (S7-400 only) • W#16#0191 Status information of all modules / racks with incorrect module ID that are not disabled (S7-400 only) • W#16#0291 Module status information of all modules that are disrupted and not disabled (S7-400 only) • W#16#0391 Module status information of all modules that are not available (S7-400 only) • W#16#0591 Module status information of all sub-modules of the host module • W#16#0991 Module status information of one DP master system • W#16#0A91 Status information of all DP sub and master systems (S7-300 only without CPU 318-2 DP) or PROFINET IO systems • W#16#0C91 Module status information of a module in the central rack or on an integrated DP interface module or on a PROFINET interface module over the logical base address • W#16#4C91 Module status information of a module on an external DP interface module using the logical base address If you use more than four external DP interface modules,RET_VAL W#16#80A4 may occur incorrectly. • W#16#0D91 Module status information of all modules in the specified rack/ station (DP or PROFINET) • W#16#0E91 Module status information of all assigned modules • W#16#0F91 Only SZL partial header information
INDEX	• For the partial list extract with the SZL-ID W#16#0C91: – S7-400: Bits 0 to 14: Logical base address of the module, Bit 15: 0 = input, 1 = output – S7-300: Bits 0 to 14: Any logical address of the module, Bit 15: 0 = input, 1 = output • For the partial list extract with the SZL-ID W#16#4C91 (S7-400 only): – Bits 0 to 14: Logical base address of the module, – Bit 15: 0 = input / 1 = output • For the partial list extract with the SZL IDs W#16#0091, W#16#0191,W#16#0291, W#16#0391, W#16#0491, W#16#0591, W#16#0A91, W#16#0E91, W#16#0F91: INDEX is irrelevant, all modules (in rack and in distributed I/O devices) • For the partial list extract with the SZL-ID W#16#0991 (S7-400 only): W#16#xx00: All modules of a DP master system (xx contains the DP master system ID) • For the partial list extract with the SZL-ID W#16#0D91 – W#16#00xx: All modules and interface modules of a rack (xx contains the number of the rack) – W#16#xxyy: All modules of a DP station or all IO devices of a PROFINET IO station (PROFIBUS DP: xx contains the DP master system ID, yy the station number; PROFINET IO: Bit 0 to 10: station number, bits 11 to 14: the last two places in the PNIO subsystem ID, bit 15: 1 (see third illustration below for adr1)

Contents	Meaning
LENTHDR	W#16#0010: One data record is 8 words long (16 bytes)
N_DR	Number of data records; depending on the product the number of records transferred in the "RDSYSST (Page 6681)" can be lower

In the case of W#16#0091, W#16#0191 and W#16#0F91 two additional data records are supplied per rack:

- A record for the power supply in as far as it exists and has been planned and
- a record for the rack.

The sequence of the records in case of a centralized structure is: PS, Slot 1, Slot 2, ..., Slot 18, rack.

A data record of the partial list with the ID W#16#xy91 has the following structure:

Name	Length	Meaning
adr1	1 word	• For a central configuration: number of the rack • For a distributed configuration with PROFIBUS DP: DP master system ID, station number • For a distributed configuration with PROFINET IO: bit 15 = 1 (PROFINET IO identifier) the last two places in the PROFINET IO system ID, station number Note: A PROFINET interface is always handled as a "sub-module in the central configuration", regardless of the use for PROFINET IO.
adr2	1 word	• For a central configuration and a distributed configuration with PROFIBUS DP: slot number and sub-module slot number • For a distributed configuration with PROFINET IO: slot number Note: A PROFINET interface is always handled as a "sub-module in the central configuration", regardless of the use for PROFINET IO.
LogicalAddr	1 word	First assigned logical I/O address (base address)
SetType	1 word	PROFINET IO: expected (configured) type (see below) otherwise reserved
ActualType	1 word	PROFINET IO: actual type (see below) otherwise reserved
res	1 word	00xx=CPU no.1-4 (only S7-400) for PROFINET IO: • SZL-ID=W#16#0C91: Number of actually existing sub-modules (without sub-module 0) • SZL-ID=W#16#0D91: Number of sub-modules (without sub-module 0) • SZL-ID=W#16#4C91: Number of actually existing sub-modules (without sub-module 0) • SZL-ID=W#16#4D91: Number of actually existing sub-modules (without sub-module 0)

Name	Length	Meaning
IOStat	1 word	I/O status • Bit 0 = 1: Module faulty (detected by diagnostic interrupt) • Bit 1 = 1: module exists • Bit 2 = 1: module does not exist • Bit 3 = 1: module disabled • Bit 4 = 1: station error (only representative slot) • Bit 5 = 1: A CiR event at this module / station is busy or not yet completed • Bit 6 = 1: reserved for S7-400 • Bit 7 = 1: module in local bus segment • Bit 8 to 15: Data ID for logical address (input: B#16#B4, output: B#16#B5, integrated DP interface: B#16#FE, external DP interface: B#16#FF)
AreaID_ModuleWidth	1 word	Area identification/module width • Bit 0 to 2: Module width • Bit 3: Reserved • Bit 4 to 6: Area ID – 0 = S7-400 – 1 = S7-300 – 2 = ET area – 3 = P-area – 4 = Q-area – 5 = IM3-area – 6 = IM4-area • Bit 7: Reserved

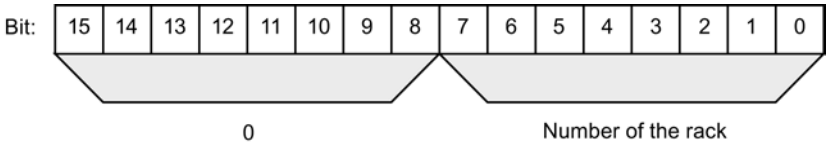
At certain modules the following values are indicated in the record:

Name	PS (S7-400 only)	CPU	IFM-CPU (S7-300)	Rack (only S7-400)
adr1	Number of the rack	Standard information as described above	Standard information as described above	Number of the rack
adr2	W#16#01FF	W#16#0200 or W#16#0200 to W#16#1800	W#16#0200	W#16#00FF
LogicalAddr	W#16#0000	W#16#7FFF	W#16#007C	W#16#0000
SetType	Standard information as described above	W#16#00C0 or W#16#0081 or W#16#0082	W#16#00C0	Standard information as described above
IOStat	W#16#0000	Standard information as described above	Standard information as described above	W#16#0000
AreaID_Module Width	W#16#0000	W#16#0011 or W#16#0001 or W#16#0002	W#16#0011	W#16#0000

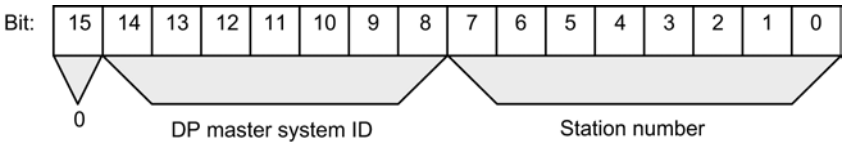
Parameter adr1

The adr1 parameter contains:

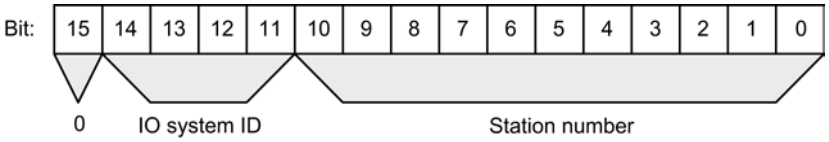
- In a central configuration, the rack number (0-31).



- In a distributed configuration with PROFIBUS DP
  - The DP master system ID (1-32)
  - The station number (0-127).



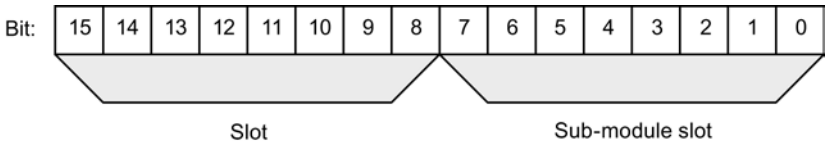
- In a distributed configuration with PROFINET IO:
  - Identifier bit for PROFINET IO (bit 15)
  - The last two positions in the PROFINET IO system ID (0-15). To obtain the complete PROFINET IO system ID, you must add 100 (decimal) to it.
  - The station number (0-2047).



Parameter adr2

The adr2 parameter contains:

- For a central configuration and for distributed configuration with PROFIBUS DP, the slot number and the sub-module slot number.



- For a distributed configuration with PROFINET IO, the slot number.

Set type parameter (SetType) and actual type parameter (ActualType) for PROFINET IO

Type identifier (W#16#...)	Meaning
8100	Entered for expected (configured) type and actual type if type check is not possible
8101	Entered for expected (configured) type if a type check is possible
8101	Entered as actual type if expected = actual
8102	Entered as actual type if expected <> actual

SZL-ID W#16#xy92 - rack / station status information

Purpose

If you read the partial list with the SZL-IDW#16#xy92, you obtain information about the expected and the actual hardware configuration of centrally installed racks and stations of a DP master system.

Reading out the SZL using "RDSYSST" for an S7-400 CPU

If you read out the partial list with "RDSYSST (Page 6681)" you must strictly ensure that the parameters SZL_ID and INDEX of "RDSYSST (Page 6681)" match each other.

SZL_ID	INDEX
W#16#0092 or W#16#0192 or W#16#0292 or W#16#0392 or W#16#0492 or W#16#0592 or W#16#0692	DP master system ID of a DP master system which is connected via an **integrated** DP connection.
:	:
W#16#4092 or W#16#4292 or W#16#4692	DP master system ID of a DP master system which is connected via an **external** DP connection.

Header

The header of the partial list with the SZL-IDW#16#xy92 is structured as follows:

Contents	Meaning
SZL-ID	The SZL-ID of the partial list extract • W#16#0092: Set status of the central racks / stations of a DP master system connected via an integrated DP interface module. • W#16#4092: Set status of the stations of a DP master system connected via an external DP interface module • W#16#0192: Activation status of the stations of a DP master system that is connected via an integrated DP interface module • W#16#0292: Set status of the central racks / stations of a DP master system connected via an integrated DP interface module. • W#16#0392: Status of the backup batteries in a rack / module rack of a CPU after at least one battery has failed • W#16#0492: Status of the overall battery backup of all racks / module racks of a CPU • W#16#0592: Status of the 24 V power supply to all racks / module racks of a CPU • W#16#4292: Actual status of the stations of a DP master system that is connected via an external DP interface module • W#16#0692: Diagnostics status of the expansion racks in the central configuration / stations of a DP master system connected via an integrated DP interface • W#16#4692: Diagnositics status of the stations of a DP master system connected via an external DP interface
INDEX	0 / DP master system ID
LENTHDR	W#16#0010: Data record is eight words long (16 bytes)
N_DR	Number of data records

Data record

A data record of the partial list with the IDW#16#xy92 has the following structure:

Contents	Length	Meaning
status_0 to status_15	16 bytes	Rack status/ station status or backup status. (The backup status is only relevant for DP modules) • W#16#0092: – Bit=0: rack/station not configured – Bit=1: rack/station not configured • W#16#4092: – Bit=0: station not configured – Bit=1: Station configured • W#16#0192: – Bit=0: Station is not configured or configured and activated – Bit=1: station is configured and activated • W#16#0292: – Bit=0: Rack/station failure, deactivated or not configured – Bit=1: Rack/station exists, activated and has not failed • W#16#4292: – Bit=0: Station failure, deactivated or not configured – Bit=1: Station exists, activated and has not failed • W#16#0692: – Bit=0: All modules of the expansion rack / station exist, are available with no problems and the station is activated – Bit=1: At least one module of the expansion rack / station is not OK or the station is deactivated • W#16#4692: – Bit=0: All modules of a station exist, are available with no problems and the station is activated – Bit=1: At least one module of a station is not OK or the station is deactivated
status_0	1 byte	Bit 0: Central rack (INDEX = 0) or station 1 (INDEX <> 0) Bit 1: 1. Expansion rack or station 2 : : Bit 7: 7. Expansion rack or station 8
status_1	1 byte	Bit 0: 8. Expansion rack or station 9 : : Bit 7: 15. Expansion rack or station 16

Contents	Length	Meaning
status_2	1 byte	Bit 0: 16. Expansion rack or station 17 : : Bit 5: 21. Expansion rack or station 22 Bit 6: 0 or station 23 Bit 7: 0 or station 24
status_3	1 byte	Bit 0: 0 or station 25 : : Bit 5: 0 or station 30 Bit 6: Expansion rack SIMATIC S5 area or station 31 Bit 7: 0 or station 32
status_4	1 byte	Bit 0: 0 or station 33 : : Bit 7: 0 or station 40
: :		
status_15	1 byte	Bit 0: 0 or station 121 : : Bit 7: 0 or station 128

SZL-ID W#16#0x94 - status information for rack/station

Purpose

The partial list with the SZL-IDW#16#0x94 contains information about the set and actual configuration of module racks in central configurations and stations of a PROFIBUS DP master system/PROFINET IO controller system.

Header

The header of the partial list with the SZL-IDW#16#0y94 is structured as follows:

Contents	Meaning
SZL-ID	The SZL-ID of the partial list extract • W#16#0094: Set status of the rack in the central rack / stations of an IO controller system (Status bit = 1: rack/station configured) • W#16#0194: Activation status of a station of an IO controller system that is configured and deactivated (Status bit =1) • W#16#0294: Set status of the rack in the central rack / stations of an IO controller system (Status bit = 1: rack/station exists, activated and not failed) • W#16#0694: Diagnostic status of the expansion units in the central rack / stations of an IO controller system (Status bit = 1: At least one module of the rack/station is disrupted or deactivated) • W#16#0794: Diagnostic / maintenance status of the central rack / stations of an IO controller system (Status bit = 0: No fault and no maintenance necessary, Status bit = 1: Rack / station has malfunction or / and maintenance required or / and maintenance demanded) • W#16#0F94: Only header information
INDEX	• 0: Central module • 1-32: Distributed module on PROFIBUS DP • 100-115: distributed module on PROFINET IO
LENTHDR	Length of the following data record
N_DR	Number of data records

Data record

A data record of the partial list with the IDW#16#0y94 has the following structure:

Contents	Length	Meaning
index	1 word	• 0: Central module • 1-32: Distributed module on PROFIBUS DP • 100-115: distributed module on PROFINET IO
status_0	BOOL	Group information • 1: At least one of the following status bits has the value 1 • 0: all the following status bits have the value 0
status_1	BOOL	Status, station 1
status_2	BOOL	Status, station 2
..		
status_2047	BOOL	Status, station 2047

A status bit of non-configured racks/stations/devices has the value "0".

Note

Important difference to the previous SZL-IDW#16#xy92

Compared to the previous SZL-IDW#16#xy92, the data have been shifted by one bit since bit status_0 is used for group information.

SZL-ID W#16#xy95 - extended DP master system / PROFINET IO system information

Purpose

The partial list with the SZL-IDW#16#xy95 supplies you with extended status information on all DP master systems/PROFINET IO systems known to the CPU. Compared to the partial list with the SZL-IDW#16#xy90, this list contains information about PROFINET IO systems and additional information about the isochronous mode of a DP master system.

Header

The header of the partial list with the SZL-IDW#16#xy95 is structured as follows:

Contents	Meaning	
SZL-ID	The SZL-ID of the partial list extract	
	W#16#0195:	Extended information on a DP master system/PROFINET IO system
	W#16#0F95:	Only SZL partial header information
INDEX	• For the partial list extract with the SZL-IDW#16#0195: – Low byte: B#16#00 – High byte: DP master system ID/PROFINET IO system ID • For the partial list extract with the SZL-IDW#16#0F95: – W#16#0000	
LENTHDR	W#16#0028:	one data record is 20 words long (40 bytes)
N_DR	Number of data records: For the partial list extract with the SZL-IDW#16#0195: 0 to 1	

Data record

A data record of the partial list with the IDW#16#xy95 has the following structure:

Name	Length	Meaning
dp_m_id	1 byte	DP master system ID/PROFINET IO system ID
dp_m_rack	1 byte	Rack number of the DP master / IO controller • For a standard CPU: 0

Name	Length	Meaning	
dp_m_slot	1 byte	Slot of the DP master / IO controller or slot of the CPU (with integrated DP interface)	
dp_m_SubModule	1 byte	• with integrated DP interface: Interface ID of the DP master / IO controller: – 1: X2 – 2: X1 – 3: IF1 – 4: IF2 • with external interface: 0	
LogicalAddr	2 bytes	Logical start address of the DP master / IO controller	
res	2 bytes	Reserved	
res	2 bytes	Reserved	
dp_m_state	1 byte	• Additional properties of the DP master system / PROFINET IO system	
		Bit 0:	DP mode (PROFIBUS DP only) • 0: S7-compatible • 1: DPV1
		Bit 1:	DP and PN cycle • 0: not constant bus cycle • 1: constant bus cycle
		Bit 2 to 6:	• Reserved
		Bit 7:	Type of DP master / IO controller • 0: integrated DP master / IO controller • 1: external DP master / IO controller
dp_address	1 byte	DP node number (PROFIBUS address)	
res	2 bytes	Reserved	
SyncCyclInter_OB	1 byte	Assigned synchronous cycle interrupt OB (only relevant if the DP cycle is equidistant)	
res	1 byte	Reserved	
baudrate	4 bytes	Baud rate of the DP master system and PNIO system as hex value	
IsoDPCycle	4 bytes	Period of the equidistant DP and PN cycle in µs	
res	16 bytes	Reserved	

SZL-ID W#16#xy96 - PROFINET IO and PROFIBUS DP module status information

Purpose

If you read the partial list with SZL-IDW#16#xy96, you obtain the status information of modules assigned to the CPU.

The information in the partial list with the SZL-IDW#16#xy96 supplements SZL-IDW#16#xy91 (Page 6718) and provides additional status data on modules and sub-modules.

It provides information specific to PROFINET IO as well as information on PROFIBUS DP modules and central modules.

Header

The header of the partial list with the SZL-IDW#16#xy96 is structured as follows:

Contents	Meaning
SZL-ID	The SZL-ID of the partial list extract • W#16#0696 Module status information of all sub-modules of a specified module (with PROFIBUS DP and central modules, the sub-module level does not exist). • W#16#0C96 Module status information of a module/sub-module located centrally or on a PROFIBUS DP/ PROFINET interface via the start address.
INDEX	• Bits 0 to 14: Address of the module • Bit 15: 0 = input, 1 = output
LENTHDR	Length of the following data record
N_DR	Number of data records

Data record

A data record of the partial list with the IDW#16#xy96 has the following structure:

Contents	Length	Meaning
LogicalAddr	1 word	• Bits 0 to 14: Address of the module • Bit 15: 0 = input, 1 = output
System	1 word	Identifier for the central module/DP master system ID /PROFINET IO system ID: • 0: Central module • 1-32: Distributed module on PROFIBUS DP • 100-115: distributed module on PROFINET IO
API*	2 words	Configured user profile for a distributed PROFINET device. Profiles are sector-specific or technology-specific specifications that go beyond the PROFINET standard. Profile 0 means that the data correspond to the specification in the PROFINET standard.
Station	1 word	Rack no./station number/device number
Slot	1 word	Slot no.
Subslot	1 word	Sub-module slot (if no sub-module can be inserted, "0" must be entered here)
Offset	1 word	Offset in user data address area of the associated module
SetType	7 words	Expected type The expected type is has a hierarchical structure in PROFINET IO

Word	PROFINET IO	PROFIBUS DP
1:	Manufacturer number or profile ID (e.g.: W#16#FF00 for PROFIBUS)	0000
2:	Device	0000
3:	Sequential number or profile index	0000
4:	1. word of the double word for sub-module identification	Type identifier

Contents	Length	Meaning		
		5:	2. word of the double word for sub-module identification	0000
		6:	1. word of the double word for sub-module identification	0000
		7:	2. word of the double word for sub-module identification	0000
SetType_UnEqual_ActualType	1 word	Expected/actual identifier • Bit 0 = 0: Set matches actual • Bit 0 = 1: Set unequal to actual • Bit 1 to 15: Reserved		
res	1 word	Reserved		
IOStat	1 word	I/O status • Bit 0 = 1: Module faulty (detected by diagnostic interrupt) • Bit 1 = 1: Module exists Bit 2 = 1: Module does not exist • Bit 3 = 1: Module disabled Bit 4 = 1: station error (only representative slot) • Bit 5 = 1: M7: Module can be host module for sub-modules : S7: A CiR event at this module /station is busy or not yet completed • Bit 6 = 1: reserved for S7-400 • Bit 7 = 1: Module in local bus segment (S7-300 only) • Bit 8 = 1: Module maintenance demanded ("green") • Bit 9 = 1: Module maintenance request ("yellow") • Bit 10 to 15: Reserved		
AreaID_ModuleWidth	1 word	Area identification/module width • Bit 0 to 2: Module width • Bit 3: Reserved • Bit 4 to 6: Area ID – 0 = S7-400 – 1 = S7-300 – 2 = PROFINET IO (distributed) – 3 = P-area – 4 = Q-area – 5 = IM3-area – 6 = IM4-area • Bit 7: Reserved • Bit 7: Reserved		
res	5 words	Reserved		
* Application Process Instance				

Partial list with the SZL-IDW#16#0696 for modules on PROFIBUS DP

This results in the error message "sub-module level not present".

SZL-ID W#16#xy9C - Tool changeover information (PROFINET IO)

Purpose

You obtain information on the configured tool changeover and their tools via the partial lists with the SZL-ID W#16#xy9C.

Tool changeover contacts are IO devices that manage tools. A tool can consist of one or several IO devices. Each tool is uniquely assigned to a tool changeover contact port.

In the case of a tool changeover all IO devices that belong to the active tool at the time are deactivated and the IO devices that belong to the new tool are activated. Deactivating and activating take place via the "D_ACT_DP (Page 6486)" instruction.

Header

The header of the partial list with the SZL-ID W#16#xy9C is structured as follows:

Contents	Meaning
SZL-ID	The SZL-ID of the partial list extract • W#16#009C Information about all tool changeover contacts and their tools on a PROFINET IO system • W#16#019C Information about all tool changeover contacts on a PROFINET IO system • W#16#029C Information about a tool changeover contact and its tools • W#16#039C Information about a tool changeover contact and its IO devices • W#16#0F9C Only SZL partial header information
INDEX	For the partial list extract with the SZL-ID • W#16#009C: PROFINET IO system ID • W#16#019C: PROFINET IO system ID • W#16#029C: logic address of a tool changeover contact • W#16#039C: logic address of any IO device of a tool • W#16#0F9C: PROFINET IO system ID
LENTHDR	W#16#000C (Length of the following data record in byte: 12)
N_DR	Number of data records

Data record

A data record of the partial list with the SZL-ID W#16#xy9C has the following structure:

Contents	Length	Meaning
StationW	1 WORD	Station number (of a data record, could be a tool changeover contact or a tool station)
LogAdrW	1 WORD	• Bits 0 to 14: Address of the module • Bit 15: 0 = input, 1 = output
StationWZK	1 WORD	Station number of a tool head (The tool head is the tool IO device directly connected to the tool changeover contact port.)
StationWZW	1 WORD	Station number of a tool changeover contact
SlotWZW	1 WORD	Slot of a tool changeover contact
SubslotWZW	1 WORD	Submodule slot of a tool changeover contact

Note

If a tool changeover contact has several ports that manage tools, a data record is supplied for each port.

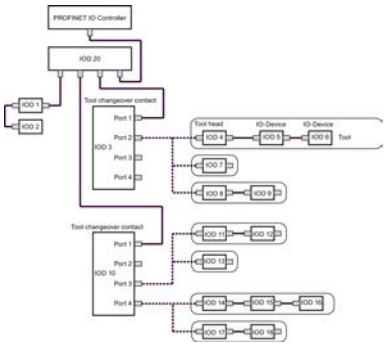
Example

The usage of the individual partial lists is shown once again in the following example.

On the phase of a PROFINET IO system, there are two tool changeover contacts (IOD 3 and IOD 10), whose tools have the following structure:

- Tool changeover contact IOD 3 (IO-Device 3) with the following three tools on the tool changeover contact port 2:
  - Tool 1 (IOD 4, IOD 5 and IOD 6)
  - Tool 2 (IOD 7)
  - Tool 3 (IOD 8 and IOD 9)
- Tool changeover contact IOD 10 with two tool changeover contact ports of which each has two tools
  - Tool changeover contact port 3: Tool 1 (IOD 11 and IOD 12) tool 2 (IOD 13)
  - Tool changeover contact port 4: Tool 1 (IOD 14, IOD 15 and IOD 16), tool 2 (IOD 17 and IOD 18)

The result is the following configuration:



The SZL partial lists return the data records for the following IO-Devices:

- Partial list extract with SZL-ID W#16#009C (Index: PROFINET IO system ID): Returns 17 data records for the following IO-Devices:
  - Tool changeover contacts: IOD 3, IOD 10 (for port 3) and IOD 10 (for port 4)
  - Tools: IOD 4, IOD 5, IOD 6, IOD 7, IOD 8, IOD 9, IOD 11, IOD 12, IOD 13, IOD 14, IOD 15, IOD 16, IOD 17 and IOD 18
- Partial list extract with SZL-ID W#16#019C (Index: PROFINET IO system ID): Returns 3 data records for the following IO-Devices:
  - Tool changeover contacts: IOD 3, IOD 10 (for port 3) and IOD 10 (for port 4)
  - Tools: None
- Partial list extract with SZL-ID W#16#029C (Index: Address of IOD 3): Returns 3 data records for the following IO-Devices:
  - Tool changeover contacts: IOD 3
  - Tools: IOD 4, IOD 5, IOD 6, IOD 7, IOD 8 and IOD 9
- Partial list extract with SZL-ID W#16#029C (Index: Address of IOD 10): Returns 10 data records for the following IO-Devices:
  - Tool changeover contacts: IOD 10 (for port 3) and IOD 10 (for port 4)
  - Tools: IOD 11, IOD 12, IOD 13, IOD 14, IOD 15, IOD 16, IOD 17 and IOD 18
- Partial list extract with SZL-ID W#16#039C (Index: Logic address of IOD 4): Returns 3 data records for the following IO-Devices: (This is analog for the logic address of IOD 5 and IOD 6)
  - Tool changeover contacts: None
  - Tools: IOD 4, IOD 5 and IOD 6
- Partial list extract with SZL-ID W#16#039C (Index: Logic address of IOD 13): Returns data record for the following IO-Devices:
  - Tool changeover contacts: None
  - Tools: IOD 13

SZL-ID W#16#xyA0 - diagnostic buffer

Purpose

If you read the partial list with SZL-IDW#16#xyA0 you obtain the entries in the diagnostics buffer of the module.

Note

The S7-300-CPUs return maximum as many data records as the displayed diagnostic buffer entries number in operating mode (default value: 10). The S7-400-CPUs return maximum 21 data records.

Header

The header of the partial list with the SZL-IDW#16#xyA0 is structured as follows:

Contents	Meaning
SZL-ID	The SZL-ID of the partial list extract • W#16#00A0: All entries that can be supplied in the currently active operating mode • W#16#01A0: The most recent entries; you specify the number of most recent entries with the INDEX parameter. If the number of alarms in the diagnostics buffer is less than the configured maximum number of alarms, the "RDSYSST (Page 6681)" instruction may provide invalid values in this partial list extract. You therefore should avoid a non-buffered POWER OFF! • W#16#0FA0: Only SZL partial header information
INDEX	Only for SZL-IDW#16#01A0: Number of most recent entries
LENTHDR	W#16#0014: One data record is 10 words long (20 bytes)
N_DR	Number of data records

Data record

A data record of the partial list with the SZL-IDW#16#xyA0 has the following structure:

Name	Length in words	Meaning
ID	1 word	Event ID
info	5 words	Information about the event and its consequences
time	4 words	Time stamp of the event

Diagnostics buffer

Additional information on events in the diagnostics buffer is available via context-sensitive help for the displayed event.

SZL-ID W#16#00B1 - module diagnostic information

Purpose

If you read the partial list SZL-IDW#16#00B1, you obtain the first 4 diagnostic bytes of a module with diagnostic capability.

Header

The header of the partial list with the SZL-IDW#16#00B1 is structured as follows:

Contents	Meaning
SZL-ID	W#16#00B1
INDEX	• Bit 0 to 14: Logical base address • Bit 15: 0 = input, 1 = output
LENTHDR	W#16#0004: One data record is 2 words long (4 bytes)
N_DR	1

Data record

A data record of the partial list with the SZL-IDW#16#00B1 has the following structure:

Name	Length	Meaning
byte0	1 byte	• Bit 0: Module fault/OK (group fault ID) • Bit 1: Internal error • Bit 2: External error • Bit 3: Channel fault • Bit 4: External voltage failed • Bit 5: Front panel connector not plugged in • Bit 6: Module is not configured • Bit 7: Incorrect parameters on module
byte1	1 byte	• Bit 0 to bit 3: Module class (CPU, FM, CP, IM, SM, ...) • Bit 4: Channel information exists • Bit 5: User information exists • Bit 6: Diagnostic interrupt from substitute • Bit 7: Maintenance requirement (PROFINET IO only)

Name	Length	Meaning
byte2	1 byte	• Bit 0: User module is missing or has an error • Bit 1: Communication problem • Bit 2: Operating mode RUN/STOP (0 = RUN, 1 = STOP) • Bit 3: Time monitoring responded (watchdog) • Bit 4: Module internal supply voltage failure • Bit 5: Battery exhausted (BFS) • Bit 6: Entire backup failed • Bit 7: Maintenance request (PROFINET IO only)
byte3	1 byte	• Bit 0: Expansion unit failure (detected by IM) • Bit 1: Processor failure • Bit 2: EPROM fault • Bit 3: RAM fault • Bit 4: ADC/DAC error • Bit 5: Fuse tripped • Bit 6: Hardware interrupt lost • Bit 7: Reserve (initialized with 0)

SZL-ID W#16#00B2 - diagnostic data record 1 with physical address

Purpose

If you read the partial list SZL-IDW#16#00B2, you obtain diagnostic data record 1 of a module in a central rack (in other words, not for DP or sub-modules). You specify the module by using the rack and slot number.

Header

The header of the partial list with the SZL-IDW#16#00B2 is structured as follows:

Contents	Meaning
SZL-ID	W#16#00B2
INDEX	W#16#xxyy: • xx contains the number of the rack • yy contains the slot number
LENTHDR	The length of the data record depends on the module.
N_DR	1

Data record

The size of a data record of the partial list SZL-IDW#16#00B2 and its contents depend on the particular module. More detailed information in this regard is provided by the respective manual for the module.

SZL-ID W#16#00B3 - module diagnostic data with logical base address

Purpose

If you read the partial list SZL-IDW#16#00B3, you will obtain all diagnostic data of a module. You can also obtain this information for DP and sub-modules. You select the module using its logical base address.

Header

The header of the partial list with the SZL-IDW#16#00B3 is structured as follows:

Contents	Meaning
SZL-ID	W#16#00B3
INDEX	• Bit 0 to 14: Logical base address • Bit 15: 0 = input, 1 = output
LENTHDR	The length of the data record depends on the module.
N_DR	1

Data record

The size of a data record of a partial list SZL-IDW#16#00B3 and its contents depends on the particular module. More detailed information in this regard is provided by the respective manual for the module.

SZL-ID W#16#00B4 - diagnostic data of a DP slave

Purpose

If you read the partial list SZL-IDW#16#00B4, you will obtain the diagnostic data of a DP slave. This diagnostic data is structured in compliance with EN 50 170 Volume 2, PROFIBUS. You select the module using the diagnostic address you configured.

Header

The header of the partial list with the SZL-IDW#16#00B4 is structured as follows:

Contents	Meaning
SZL-ID	W#16#00B4
INDEX	Configured diagnostic address of the DP slave.
LENTHDR	Length of a data record. The maximum length is 240 bytes; for standard slaves, which have a diagnostic data length of more than 240 bytes up to a maximum of 244 bytes, the first 240 bytes are read and the overflow bit is set in the data.
N_DR	1

Data record

A data record of the partial list with the SZL-IDW#16#00B4 has the following structure:

Name	Length	Meaning
status1	1 byte	Station status1
status2	1 byte	Station status2
status3	1 byte	Station status3
Station_No	1 byte	Master station number
Vendor_ID_highByte	1 byte	Vendor ID (high byte)
Vendor_ID_lowByte	1 byte	Vendor ID (low byte)
....	....	Additional slave-specific diagnostic information

OB_RT: Determine OB program runtime

Description

With the instruction, you can determine the runtime of individual OBs over different time periods.

Note

The instruction returns the last recorded time values for the required OB, regardless of whether or not this OB is currently being loaded. The data are not reset by deleting or overwriting them but rather only after a warm restart.

Parameter

The following table shows the parameters of the instruction "OB_RT":

Parameters	Declaration	Data type	Memory area	Meaning
OB_NR	Input	INT	I, Q, M, D, L	OB whose last evaluated times are to be queried. Valid OB numbers are all those in the OB configuration of your CPU, except for OB 121 and OB 122. Synchronous error processing time is included in the time required for processing the OB that has caused the error. Specification of OBs 121 and 122 or OBs that are not implemented in the CPU causes an error message. For OB_NR=0, the data of the OB in whose context the instruction was called will be transferred. When "OB_RT" is called in OB 121 or OB 122 with OB_NR=0, all time data of the interrupt triggering OB are output, including the time data in OB 12x.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code. Otherwise, RET_VAL contains the number of the OB for which these data were called.
PRIO	Output	INT	I, Q, M, D, L	The priority class of the queried OB is output in PRIO .
LAST_RT	Output	DINT	I, Q, M, D, L	Runtime (in microseconds) of the last completed execution of the specified OB. If the OB for which you request the runtimes is currently being processed, the following applies: For the first call of the "OB_RT" instruction during the current execution of the required OB, LAST_RT indicates the run time of the last completed OB execution. With each subsequent call of the "OB_RT" instruction during the current execution of the required OB, LAST_RT • DW#16#FFFF FFFF indicates, if the required OB has already called an "OB_RT" with OB_NR=0. • The runtime of the last completed OB, if an "OB_RT" call with OB_NR=0 has not occurred in the required OB. Note: Interruptions caused by OBs with higher priority are not included in LAST_RT . The OB-specific operating system services (such as generation and provision of the OB start information, updating of the process image, updating of the process image partition) are contained in LAST_RT .

Parameters	Declaration	Data type	Memory area	Meaning
LAST_ET	Output	DINT	I, Q, M, D, L	Time interval (in microseconds) between the OB call and the end of execution of the specified OB. If the OB for which you want to determine runtimes is currently in process the following applies: For the first call of the "OB_RT" instruction during the current processing of the required OB, the time interval between the last completely executed OB request and the end of processing of the specified OB is specified in LAST_ET . For each subsequent call of the "OB_RT" instruction during current processing of the required OB in LAST_ET • DW#16#FFFF FFFF indicates, if the required OB has already called an "OB_RT" with OB_NR=0. • The time interval between the last completely executed OB request and the end of execution of the required OB, if an "OB_RT" call with OB_NR=0 has not occurred in the required OB. Note: Interruptions caused by OBs with higher priority are included in LAST_ET .
CUR_T	Output	DINT	I, Q, M, D, L	Timing of the OB request of the specified OB that is currently being processed, as a relative time in microseconds. If the specified OB is currently not being processed, CUR_T will contain the value "0". Note: The system time is a counter that counts from "0" to "2 147 483 647" microseconds. If an overflow occurs, the counter restarts at "0".
CUR_RT	Output	DINT	I, Q, M, D, L	Previous runtime of the current execution of the specified OB in microseconds.CUR_RT is "0", if the OB is not being processed or has not yet been processed. After processing, the runtime is transferred to LAST_RT and CUR_RT is set to zero. Note: Interruptions caused by OBs with higher priority are not included in CUR_RT . The OB-specific operating system services performed until "OB_RT" is called are contained in CUR_RT .

Parameters	Declaration	Data type	Memory area	Meaning
CUR_ET	Output	DINT	I, Q, M, D, L	Time elapsed since the request of the specified OB that is currently being processed, in microseconds. CUR_ET is "0", if the specified OB is currently not being processed. After processing, the runtime is transferred to LAST_ET and CUR_ET is set to "0". Note: Interruptions caused by OBs with higher priority are included in CUR_ET .
NEXT_ET	Output	DINT	I, Q, M, D, L	If additional calls of the specified OB are queued before the current request has been completed, NEXT_ET indicates the time-to-go between the current time and the time of execution of the next request in microseconds.. NEXT_ET is "0", if no other start event exists besides the start events for this OB that are currently pending or undergoing processing. WinLC RTX and the S7-400 CPUs do not use this parameter. In these cases, NEXT_ET has the value"FFFF FFFF". Note: Interruptions caused by OBs with higher priority are included in NEXT_ET .

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

The times also include the runtimes for any nested processing of synchronous error interrupts (OB 121, OB 122).

Note

When you declare an OB number in OB_NR that exists in the dynamic project data on your CPU without the OS having called the corresponding OB, or you have not yet downloaded it to the CPU, RET_VAL contains the specified OB number and PRIO contains the configured (default, if required) priority of the specified OB. The runtime parameter (CUR_RT, CUR_ET, LAST_RT, LAST_ET, NEXT_ET) returns the initial value DW#16#FFFF FFFF. For the following restart methods or operating state changes, the runtime parameters are set to their initial value: Warm restart, cold restart, hot restart.

Parameter RET_VAL

Event class error code	Explanation
1 to 102	Number of the OB to which information is being transferred.
W#16#8080	OB_NR parameter contains an illegal value.
W#16#8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

C_DIAG: Determine current connection status

Description

You use the instruction to determine the current status of all S7 connections and of all fault-tolerant S7 connections (or their partial connections).

Suitable evaluation of these connection data lets you recognize failures of S7 connections as well as of fault-tolerant S7 connections and report these, if necessary, to an operator control and monitoring system. Monitored connections can be a connection between automation systems as well as the connection of an automation system to an operator control and monitoring system.

Note

A change in the operating state of the CPU: RUN -> STOP ->RUN, does not affect the state of the configured connections. Exception: When an H-station changes from the redundant system state to the Stop system state, the partial connections of all fault-tolerant connections to the standby CPU will be disconnected.

After a power failure, on the other hand, all configured connections will be reestablished and this changes the connection status.

When the instruction is called for the first time or after startup, the connection information will differ, depending on whether the last operating mode of the CPU was STOP or POWER OFF.

Functional description

The "C_DIAG" instruction functions asynchronously, that is, its execution extends over multiple calls.

You start the job by calling "C_DIAG" with REQ=1.

If the job could be executed immediately, the instruction will return the value "0" at the output parameter BUSY . If BUSY = "1", the job is still active.

Call

To recognize the failure of S7 connections and fault-tolerant S7 connections, you call "C_DIAG" in a cyclic interrupt OB that is started, for example, every 10 seconds by the operating system.

Because the status of a connection normally does not change, it is appropriate to copy the connection data to the user program with these cyclic calls only if they have changed since their last call (call with MODE=B#16#02, see below).

The "C_DIAG" instruction has four possible operating modes, which are explained in the following table.

MODE (B#16#...)	"C_DIAG" copies connection data to the user program	"C_DIAG" transfers acknowledgement information to the operating system
00	No	Yes
01	Yes	Yes

MODE (B#16#...)	"C_DIAG" copies connection data to the user program	"C_DIAG" transfers acknowledgement information to the operating system
02	• Yes, if connection data have changed • No, if connection data have not changed	Yes
03	Yes	No

The status changes of the connection data since the last call of "C_DIAG" (with MODE=B#16#00, 01 or 02) are confirmed by transferring the acknowledgement information to the operating system.

Note

If you operate "C_DIAG" in a cyclic interrupt OB in "Conditional copying" mode (MODE=B#16#02), you must ensure that no initial values are contained in the target range after a cold restart of the CPU. You can achieve this in OB 102 with a single call of "C_DIAG" with "Unconditional copying with acknowledgment" mode (MODE=B#16#01).

Parameter

The following table shows the parameters of the instruction "C_DIAG":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter request to activate REQ=1: Initialize the job, if not already started
MODE	Input	BYTE	I, Q, M, D, L or constant	Job identifier Possible values: • B#16#00: "C_DIAG" does not copy connection data, but merely transfers acknowledgement information to the operating system. • B#16#01: Regardless of the status change, "C_DIAG" copies all connection data to the user program and transfers an acknowledgement information to the operating system. • B#16#02: If connection data have changed, "C_DIAG" will copy them to the user program. If not changed, they are not copied. "In both cases, C_DIAG" transfers an acknowledgement information to the operating system. • B#16#03: "C_DIAG" copies the connection data to the user program, independent of its change status. It does not transfer acknowledgement information to the operating system.
RET_VAL	Return	INT	I, Q, M, D, L	Return value (error code or job status)
BUSY	Output	BOOL	I, Q, M, D, L	BUSY=1: The job is not yet complete.

Parameter	Declaration	Data type	Memory area	Description
N_CON	Output	INT	I, Q, M, D, L	Index of the last structure in CON_ARR, with .DIS_PCON or .DIS_CON = TRUE . This means only the first N_CON elements of CON_ARR need to be checked in the user program. Note: The first structure in the CON_ARR array has the index "1".
CON_ARR	Output	ANY	I, Q, M, D, L	Destination area for the read connection data. Only the BYTE data type is permitted. A structure is assigned to each connection. Choose a target range size that can receive all structures even at the maximum number of possible connections for your CPU. This is particularly important after a CPU replacement when the new CPU which has greater connection resources than the previous one.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter CON_ARR

The target range for the read connection data is an array of structures. A structure is assigned to each connection.

Initially, the array does not need to be occupied with valid entries and it may contain invalid entries between two valid entries.

The connections are not sorted by connection reference.

Note

Data consistency of a connection is ensured if you copy connection data from the operating system to the selected target range.

Structure organization

Parameters	Data type	Description
CON_ID	WORD	Connection reference that you have assigned to this connection during configuration W#16#FFFF: Invalid ID, which means the connection has not been configured. If CON_ARR[i].DIS_PCON or CON_ARR[i].DIS_CON (see below) is also set, this connection has been reconfigured or deleted since the last call of "C_DIAG".
STAT_CON	BYTE	The current status of the S7 connection or fault-tolerant S7 connection Possible values: • B#16#00: S7 connection not established • B#16#10: Fault-tolerant S7 connection not established • B#16#01: S7 connection is currently being established • B#16#11: Fault-tolerant S7 connection is currently being established • B#16#02: S7 connection is established • B#16#12: Fault-tolerant S7 connection is established, but is not fault tolerant • B#16#13: Fault-tolerant S7 connection has been established and is fault tolerant
PROD_CON	BYTE	Partial connection number of the productive connection Possible values: 0, 1, 2, 3
STBY_CON	BYTE	Partial connection number of the standby connection (B#16#FF: no standby connection) Possible values: 0, 1, 2, 3 Note: Only a fault-tolerant S7 connection can have a standby connection.
DIS_PCON	BOOL	The transitions W#16#12 -> W#16#13 and W#16#13 -> W#16#12 from CON_ARR[i].STAT_CON since the last instruction call set CON_ARR[i].DIS_PCON to "1". All other status changes of the i connection leave CON_ARR[i].DIS_PCON unchanged. Note: • With MODE=B#16#01 and 02 , the operating system bit that corresponds to DIS_PCON is reset when connection data are copied to the target range. • With MODE=B#16#03 , the operating system bit that corresponds to DIS_PCON remains unchanged.
DIS_CON	BOOL	Each modification of CON_ARR[i].STAT_CON since the last "C_DIAG" call, with the exception of the transitions W#16#12 -> W#16#13 and W#16#13 -> W#16#12 sets CON_ARR[i].DIS_CON to 1. Note: • With MODE=B#16#01 and 02 , the operating system bit that corresponds to "DIS_CON" is reset when connection data are copied to the target range. • With MODE=B#16#03 , the operating system bit that corresponds to DIS_CON remains unchanged.
RES0	BYTE	Reserved (B#16#00)
RES1	BYTE	Reserved (B#16#00)

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	• MODE=B#16#00, 01 or 02: No connection status change (structure element STAT_CON) since the last call. Job was executed without errors. • MODE=B#16#03: The copy procedure was carried out without errors.
0001	• MODE=B#16#00, 01 or 02: Connection status change (structure element STAT_CON) with at least one connection since the last call. Job was executed without errors. • MODE=B#16#03: RET_VAL W#16#0001 is not possible:
7000	First call with REQ=0. The job specified with MODE is not active. BUSY has the value "0".
7001	First call with REQ=1. The job specified with MODE is not triggered. BUSY has the value "1".
7002	Intermediate call (REQ irrelevant). The activated job is still running. BUSY has the value "1".
8080	The MODE parameter contains an invalid value.
8081	The CON_ARR parameter contains an illegal data type.
8082	The length specification in the CON_ARR parameter is too small. The instruction copies no data to the target range.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

Data block functions

READ_DBL: Read from data block in the load memory

Description

With the instruction you copy a DB or an area of a DB in load memory (Micro Memory Card) to the data area of a destination DB. The destination DB must be relevant for execution; that is, it must not be created with the attribute UNLINKED. The content of the load memory is not changed during the copy process.

To ensure data consistency, you must not change the destination area while "READ_DBL" is being executed (which means as long as the BUSY parameter has the value TRUE).

The following restrictions apply to the parameters SRCBLK and DSTBLK:

- For an ANY pointer of the type BOOL, the length must be divisible by 8.
- For an ANY pointer of the type STRING, the length must be equal to 1.

If required, you can determine the length of the source DB with the "TEST_DB (Page 6758)" instruction.

Note

READ_DBL is processed asynchronously. Therefore, it is not suitable for frequent (or cyclical) reading of tags in the load memory.

Once started, a job is always completed. If the maximum number of simultaneously active "READ_DBL" instructions is reached and you call "READ_DBL" once again at this time in a priority class having higher priority, error code W#16#80C3 will be returned. This means it does not make sense to restart the high-priority job right away.

Functional description

The "READ_DBL" instruction works asynchronously, that is, its execution extends over multiple calls. You start the job by calling "READ_DBL" with REQ =1.

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 5367)

Parameter

The following table shows the parameters of the instruction "READ_DBL":

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = 1: Read request
SRCBLK	Input	ANY	D	Pointer to data block in the load memory that is to be read from
RET_VAL	Return	INT	I, Q, M, D, L	Error information
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The reading process is not yet complete.
DSTBLK	Output	ANY	D	Pointer to the data block in the work memory that is to be written to

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code (W#16#...)	Description
0000	No error
0081	The destination area is larger than the source area. The source area is written completely to the destination area; the remaining bytes of the destination area will not be changed.
7000	First call with REQ=0: No data transfer active; BUSY has the value "0".
7001	First call with REQ=1: Data transfer triggered; BUSY has the value "1".
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".
8081	The source area is larger than the destination area. The destination area is fully written. The remaining bytes of the source area are ignored.
8093	No data block or a data block that is not in the work memory is specified for the DSTBLK parameter.
80B1	No data block is specified for the SRCBLK parameter, or the data block specified there is not an object in the load memory (for example, a DB created using "CREAT_DB (Page 6753)").
80B4	DB with F-attribute must not be read.
80C0	The destination DB is currently being processed by another instruction or a communication function.
80C3	The maximum number of simultaneously active "READ_DBL" instructions has already been reached.
8xyy	General error codes See also: Evaluating errors with output parameter RET_VAL (Page 5370)

WRIT_DBL: Write to data block in the load memory

Description

You use the instruction to transfer the contents of a DB or a DB area from the work memory to a DB or a DB area in the load memory (Micro Memory Card). The source DB must be relevant for execution, which means it must not be created with the attribute UNLINKED. It can, however, have been created with the "CREAT_DB" instruction.

To ensure data consistency, you must not change the source area while "WRIT_DBL" is being executed (which means as long as the BUSY parameter has the value TRUE).

The following restrictions apply to the parameters SRCBLK and DSTBLK:

- For an ANY pointer of the type BOOL , the length must be divisible by 8.
- For an ANY pointer of the type STRING , the length must be equal to 1.

If required, you can determine the length of the destination DB with the "TEST_DB (Page 6758)" instruction.

"WRIT_DBL" does not change the checksum of the user program if you write a DB that was created using an instruction. However, when a loaded DB is written, the first entry in this DB changes the checksum of the user program.

Note

"WRIT_DBL" is not suitable for frequent (or cyclical) writing of tags in the load memory. This is because the Micro Memory Card technology limits the number of write accesses that can be made to a Micro Memory Card.

Functional description

The "WRIT_DBL" instruction works asynchronously, that is, its execution extends over multiple calls. You start the job by calling "WRIT_DBL" with REQ =1.

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 5367)

Parameter

The following table shows the parameters of the instruction "WRIT_DBL":

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = 1: Write request
SRCBLK	Input	ANY	D	Pointer to the DB in the work memory that is to be read from
RET_VAL	Return	INT	I, Q, M, D, L	Error information
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The writing process is not yet complete.
DSTBLK	Output	ANY	D	Pointer to the data block in the load memory that is to be written to.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code (W#16#...)	Description
0000	No error
0081	The destination area is larger than the source area. The source area is written completely to the destination area. The remaining bytes of the destination area will not be changed.
7000	First call with REQ=0: No data transfer active; BUSY has the value "0".

Error code (W#16#...)	Description
7001	First call with REQ=1: Data transfer triggered; BUSY has the value "1".
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".
8081	The source area is larger than the destination area. **The destination area is fully written. The remaining bytes of the source area are ignored.**
8092	Incorrect operating mode: While "WRIT_DBL" was active, the CPU went into STOP mode. This error code is supplied at the next transition to RUN. Call "WRIT_DBL" again.
8093	No data block or a data block that is not in the work memory is specified for the SRCBLK parameter.
80B1	No data block is specified for the DSTBLK parameter, or the data block specified there is not an object in the load memory (for example, a DB created using "CREAT_DB (Page 6753)").
80B4	DB with F-attribute must not be changed
80C0	The destination is currently being processed by another instruction or a communication function. Example: You upload a DB from the CPU to the programming device (PG). You want to change the contents of this DB using "WRIT_DBL".
80C3	The maximum number of simultaneously active "WRIT_DBL" instructions has already been reached.
8xyy	General error codes See also: Evaluating errors with output parameter RET_VAL (Page 5370)

DEL_DB: Delete data block

Description

You use the instruction to delete a data block located in the work memory and, if present, in the load memory of the CPU. The DB to be deleted must not be open in the current or in any lower priority class, in other words, it must not be entered in either of the two DB registers or in the B stack.

Otherwise, the CPU starts OB 121 when the instruction is called. If OB 121 is not present, the CPU switches to STOP mode. For S7-300 (exception: CPU 318), the DB is deleted without calling OB 121.

Note

It is not advisable to delete instance DBs with "DEL_DB" as this always leads to program errors. Therefore, avoid deleting instance DBs with "DEL_DB".

The following table explains when a DB can be deleted with "DEL_DB".

If ...	then use "DEL_DB" ...
the DB was created by calling the "CREAT_DB" instruction,	it can be deleted.
was transferred to the CPU and was not created with the attribute "Only store in load memory"	it can be deleted.
was created with the attribute "Only store in load memory"	• For S7-300 can be deleted: • For S7-400 cannot be deleted:

If ...	then use "DEL_DB" ...
was created by calling the "CREA_DBL" instruction	can be deleted
The DB was stored on the flash card,	it cannot be deleted.

Interrupt ability

The instruction can be interrupted by priority classes of a higher priority. If the instruction is called again there, then this second call will be aborted and W#16#8091 will be entered in RET_VAL .

Parameter

The following table shows the parameters of the instruction "DEL_DB":

Parameters	Declaration	Data type	Memory area	Description
DB_NUMBER	Input	WORD	I, Q, M, D, L or constant	Number of the DB to be deleted
RET_VAL	Return	INT	I, Q, M, D, L	Error information

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error occurred.
8091	"DEL_DB" calls were nested and the maximum nesting level of the CPU used was exceeded.
8092	The "Delete a DB" function cannot be executed currently because • The "Compress user memory" function is currently active. • You are copying the DB to be deleted from the CPU to an offline project. • The WinAC software CPU has detected an error in the OS of the computer on which WinAC is installed.
80A1	Error in input parameter DB_NUMBER: the actual parameter selected • Is "0" • Is greater than the maximum permitted DB number for the CPU used.
80B1	The DB with the specified number does not exist in the work memory of the CPU.
80B3	The DB is on a flash card.
80B4	The DB could not be deleted. Possible causes: • It belongs to an F-program. • It is an instance DB of a block for S7 communication (for S7-400 only). • It is a technology DB.

Error code (W#16#...)	Explanation
80C1	The "Delete a DB" function cannot be executed at this time due to a temporary resource bottleneck.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

CREAT_DB: Create data block

Description

You use the instruction to create a data block with no predefined values in the user program. Instead, the DB contains random data. The instruction creates a data block with a number from a specified area and with a predefined size. The instruction assigns the smallest possible number from the specified area to the DB. To create a DB with a specific number, enter the same number for the upper and lower limits of the specified area. You cannot assign the numbers of the DBs already contained in the user program. The length of the DB must be specified with an even number.

Interrupt ability

The instruction can be interrupted by higher-priority OBs. If a "CREAT_DB" instruction is called in a higher-priority OB, this call will be rejected with the error code W#16#8091.

Parameter

The following table shows the parameters of the instruction "CREAT_DB":

Parameters	Declaration	Data type	Memory area	Description
LOW_LIMIT	Input	WORD	I, Q, M, D, L or constant	The lower limit value is the lowest number in the range of numbers that you can assign to your data block.
UP_LIMIT	Input	WORD	I, Q, M, D, L or constant	The upper limit value is the highest number in the range of numbers that you can assign to your data block.
COUNT	Input	WORD	I, Q, M, D, L or constant	The count value indicates the number of data bytes that you want to reserve for your data block. You must specify an even number of bytes (maximum 65534).
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
DB_NUMBER	Output	WORD	I, Q, M, D, L	The data block number is the number of the created data block. In case of an error (bit 15 of RET_VAL was set), the value "0" will be entered in DB_NUMBER .

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error
8091	You have called CREAT_DB as a nested instruction.
8092	The "Create a DB" function is currently unavailable because • The "Compress user memory" function is currently active. • The number of DBs in the CPU has already reached the maximum possible number. • The H-CPU is running link-up or update functions. • The WinAC software CPU has detected an error in the OS of the computer on which WinAC is installed. • The previous Delete is not completed yet
80A1	Error in the DB number: • The number is "0" • The number exceeds the CPU-specific DB quantity • Lower limit > upper limit
80A2	Error in length of DB: • The length is "0" • The length was specified with an uneven number • The length is greater than the CPU allows
80B1	There is no DB number free.
80B2	There is not enough free memory available.
80B3	There is not enough continuous memory space available (remedy: compress memory!)
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

CREA_DB: Create data block / retentive

Description

You use the instruction to create a data block with no predefined values in the user program. Instead, the DB contains random data. The instruction creates a data block with a number from a specified area and with a predefined size. The instruction assigns the smallest possible number from the specified area to the DB. To create a DB with a specific number, enter the same number for the upper and lower limits of the specified area. You cannot assign the numbers of the DBs already contained in the user program. The length of the DB must be specified with an even number.

Depending on the selection you have made for the ATTRIB parameter, the DB created has the property RETAIN or NON_RETAIN:

- RETAIN (=retentive) means that the DB is created in the retentive part of the work memory. This means the current values of the DB are retained after each power OFF/power ON transition and every restart (warm restart).
- NON_RETAIN (=non-retentive) means that the DB is created in the non-retentive part of the work memory. This means the current values of the DB are undefined after each power OFF/power ON transition and every restart (warm restart).

If no distinction has been made between retentive and non-retentive work memory, the ATTRIB parameter will be ignored. This means the values of the DB will be retained after each power OFF/power ON transition and every restart (warm restart).

Interrupt ability

The "CREA_DB" instruction can be interrupted by higher-priority OBs. If a "CREA_DB" instruction is called in a higher-priority OB, this call will be rejected with the error code W#16#8091 .

Parameter

The following table shows the parameters of the instruction "CREA_DB":

Parameter	Declaration	Data type	Memory area	Description
LOW_LIMIT	Input	WORD	I, Q, M, D, L or constant	The lower limit value is the lowest number in the range of numbers that you can assign to your data block.
UP_LIMIT	Input	WORD	I, Q, M, D, L or constant	The upper limit value is the highest number in the range of numbers that you can assign to your data block.
COUNT	Input	WORD	I, Q, M, D, L or constant	The count value indicates the number of data bytes that you want to reserve for your data block. You must specify an even number of bytes (maximum 65534).
ATTRIB	Input	BYTE	I, Q, M, D, L or constant	DB attributes: • B#16#00: RETAIN • B#16#04: NON_RETAIN
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
DB_NUMBER	Output	WORD	I, Q, M, D, L	The data block number is the number of the created data block. In case of an error (bit 15 of RET_VAL was set), the value "0" will be entered in DB_NUMBER .

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error
8091	You have called CREA_DB as a nested instruction.
8092	The "Create a DB" function is currently unavailable because • The "Compress user memory" function is currently active. • The WinAC software CPU has detected an error in the OS of the computer on which WinAC is installed.
8094	Invalid value in parameter ATTRIB
80A1	Error in the DB number: • The number is "0" • The number exceeds the CPU-specific DB quantity • Lower limit > upper limit
80A2	Error in length of DB: • The length is "0" • The length was specified with an uneven number • The length is greater than the CPU allows
80B1	There is no DB number free.
80B2	There is not enough free memory available.
80B3	There is not enough continuous memory space available (remedy: compress memory!)
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

CREA_DBL: Create data block in the load memory

Description

You use the instruction to create a new data block in the load memory (Micro Memory Card). The instruction creates a data block with a number from a specified area and with a predefined size. The instruction assigns the smallest possible number from the specified area to the DB. To create a DB with a specific number, enter the same number for the upper and lower limits of the range to be specified. You cannot assign the numbers of the DBs already contained in the user program. If a DB with the same number already exists in work memory and/or load memory, or if the DB exists as a copied version, the instruction is terminated and error information is generated.

Note

With the "TEST_DB (Page 6758)" instruction, you can determine whether a DB with the same number already exists.

The contents of the data area to which the parameter SRCBLK(source block) points are written to the DB. This data area must be a DB or an area from a DB. To ensure data consistency,

you must not change this data area while "CREA_DBL" is being executed (which means as long as the BUSY parameter has the value TRUE).

A DB with the READ_ONLY attribute can only be created and initialized using "CREA_DBL".

"CREA_DBL" does not change the checksum of the user program.

Functional description

The "CREA_DBL" instruction works asynchronously, that is, its execution extends over multiple calls. You start the job by calling "CREA_DBL" with REQ=1.

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 5367)

Parameter

The following table shows the parameters of the instruction "CREA_DBL":

Parameters	Declaration	Data type	Memory area	Description	
REQ	Input	BOOL	I, Q, M, D, L	REQ = 1: Request to create the DB	
LOW_LIMIT	Input	WORD	I, Q, M, D, L	Low limit of the range used by "CREA_DBL" to assign a number to your DB	
UP_LIMIT	Input	WORD	I, Q, M, D, L	Upper limit of the range used by "CREA_DBL" to assign a number to your DB	
COUNT	Input	WORD	I, Q, M, D, L	The count value specifies the quantity of data bytes you want to reserve for your DB. Here you must specify an even number of bytes.	
ATTRIB	Input	BYTE	I, Q, M, D, L	DB properties:	
				Bit 0 = 1:	UNLINKED: The DB exists only in load memory.
				Bit 1 = 1:	READ_ONLY: The DB is read-only.
				Bit 2 = 1:	NON_RETAIN: The DB is not retentive.
				Bit 3 to 7:	Reserved
SRCBLK	Input	ANY	D	Pointer to the data block whose values will be used to initialize the created DB	
RET_VAL	Return	INT	I, Q, M, D, L	Error information	
BUSY	Output	BOOL	I, Q, M, D, L	BUSY = 1: The process is not yet complete.	
DB_NUM	Output	WORD	I, Q, M, D, L	Number of the created DB	

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code (W#16#...)	Description
0000	No error
0081	The target range is larger than the source range. The complete source range is written to the target range. The remaining bytes of the target range are filled with 0.
7000	First call with REQ=0: No data transfer active; BUSY has the value "0".
7001	First call with REQ=1: Data transfer triggered; BUSY has the value "1".
7002	Intermediate call (REQ irrelevant): Data transfer already active; BUSY has the value "1".
8081	The source range is larger than the target range. The complete target range is written. The remaining bytes of the source range are ignored.
8091	You have called "CREA_DBL" as a nested instruction.
8092	The "Create data block" function is currently unavailable because • The "Compress user memory" function is currently active. • The maximum number of blocks on your CPU has already been reached.
8093	No data block or a data block that is not in the work memory is specified for the SRCBLK parameter.
8094	A not yet supported attribute was specified for the ATTRIB parameter.
80A1	DB number error: • The number is 0 or is not within the valid range for the CPU (CPU15xx: 60001 to 60999) • The number exceeds the CPU-specific DB quantity • Low limit > high limit
80A2	DB length error: • The length is "0" • The length is an odd number • The length is greater than permitted by the CPU
80B1	No free DB number
80B2	Insufficient work memory
80BB	Insufficient load memory
80C0	The destination is currently being processed by another instruction or a communication function.
80C3	The maximum number of simultaneously active "CREA_DBL" instructions has already been reached.
8xyy	General error codes, for example: • Source DB does not exist or it is only available as copied version • Source range in DB does not exist See also: Evaluating errors with output parameter RET_VAL (Page 5370)

TEST_DB: Test data block

Description

With the instruction you receive on S7-300 information about a data block that is in the CPU work or load memory, on S7-400 information about a data block that is in the CPU work

memory. The instruction determines the number of data bytes in the selected DB and checks whether or not the DB is read only.

Parameters

The following table shows the parameters of the instruction "TEST_DB":

Parameters	Declaration	Data type	Memory area	Description
DB_NUMBER	Input	WORD	I, Q, M, D, L or constant	Number of the DB to be tested
RET_VAL	Return	INT	I, Q, M, D, L	Error information
DB_LENGTH	Output	WORD	I, Q, M, D, L	Number of data bytes the selected DB contains.
WRITE_PROT	Output	BOOL	I, Q, M, D, L	Information about the write-protect identifier of the DB ("1" means read only).

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error occurred.
80A1	Error in input parameter DB_NUMBER: the actual parameter selected • Is "0" • Is greater than the maximum permitted DB number for the CPU used.
80B1	The DB with the specified number does not exist on the CPU.
80B2	The DB was created using the keyword UNLINKED.
8xyy	General error information, See also: Evaluating errors with output parameter RET_VAL (Page 5370)

Table functions

ATT: Add value to table

Description

The instruction supplements the DATA parameter as the next entry in a table and increments the number of entries by one. The table consists of words. You use the instruction to create table entries that are used by "FIFO (Page 6761)" and "LIFO (Page 6765)".

- The first entry in a "FIFO (Page 6761)" or "LIFO (Page 6765)" table specifies the maximum length of the table.
- The second entry in the table specifies the number of existing entries.
- The third entry in the table contains the first word of data.

Note

When you create the table, initialize the first two entries.

Parameter

The following table shows the parameters of the instruction "ATT":

Parameters	Declaration	Data type	Memory area	Description
DATA	Input	WORD	I, Q, M, D, L, P or constant	Data to be entered in the table.
TABLE	Input	*Pointer	I, Q, M, D	Points to the start address of the "FIFO" or "LIFO" table.
* Pointer in double-word format for area-crossing, indirect register addressing				

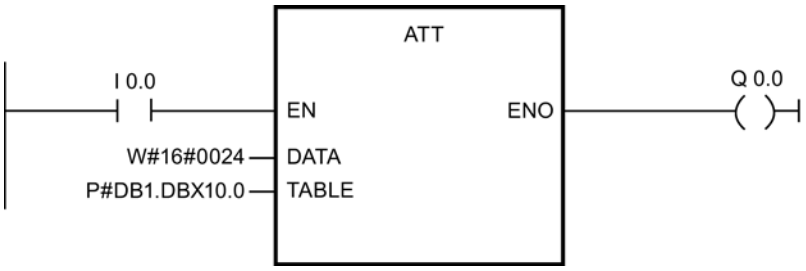
For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

If the number of entries is equal to or greater than the table length, the data will not be added to the table and the signal state of "BR" is set to "0".

Example (LAD)

The "ATT" instruction is executed if the signal state at input I 0.0 is 1 (activated). In this example, DATA is added as the fifth entry in the table and the number of entries is increased from 4 to 5.

If the "ATT" instruction is executed without errors, the signal states of "ENO" and Q 0.0 will be set to "1".



Before processing:

TABLE (table length)	DBW10 = W#16#0006
Number of entries	DBW12 = W#16#0004 DBW14 = W#16#0012 DBW16 = W#16#0029 DBW18 = W#16#0090 DBW20 = W#16#0002 DBW22 = W#16#0000 DBW24 = W#16#0000

After processing:

TABLE (table length)	DBW10 = W#16#0006
Number of entries	DBW12 = W#16#0005 DBW14 = W#16#0012 DBW16 = W#16#0029 DBW18 = W#16#0090 DBW20 = W#16#0002 DBW22 = W#16#0024 DBW24 = W#16#0000

FIFO: Output first value of the table

Description

The instruction outputs the oldest entry of the "FIFO" table as function value. The number of entries is decremented by one. If there are still entries in the table, these are shifted down. The "FIFO" table consists of words. Use ATT (Page 6760) to enter values in the "FIFO" table.

- The first entry in the table specifies the maximum length of the table.
- The second entry in the table specifies the number of existing entries.
- The third entry in the table contains the first word of data.

Parameter

The following table shows the parameters of the instruction "FIFO":

Parameters	Declaration	Data type	Memory area	Description
TABLE	Input	*Pointer	I, Q, M, D	Points to the start address of the "FIFO" table.
RET_VAL	Return	WORD	I, Q, M, D, L, P	The oldest entry from the "FIFO" table.
* Pointer in double-word format for area-crossing, indirect register addressing				

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

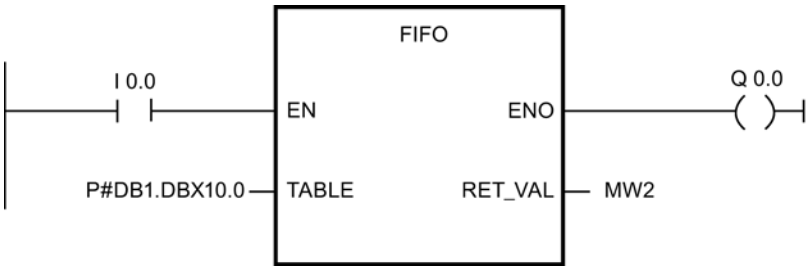
Parameter RET_VAL

If the "FIFO" table is empty (number of entries = 0), then RET_VAL will remain the same and the signal state of "BR / ENO" will be set to "0".

Example (LAD)

"FIFO" is executed if the signal state at input I 0.0 is 1 (activated). In this example, the oldest entry in the table is returned as the function value (MW 2.0). The number of entries is decremented from 5 to 4, and the remaining entries are shifted down in the table.

If "FIFO" is executed without errors, then the signal states of "BR / ENO" and Q 0.0 will be set to "1".



Before processing:

TABLE (table length)	DBW10 = W#16#0006
Number of entries	DBW12 = W#16#0005 DBW14 = W#16#0012 DBW16 = W#16#0029 DBW18 = W#16#0090 DBW20 = W#16#0002 DBW22 = W#16#0024 DBW24 = W#16#0000
RET_VAL	MW2 = W#16#0000

After processing:

TABLE (table length)	DBW10 = W#16#0006
Number of entries	DBW12 = W#16#0004 DBW14 = W#16#0029 DBW16 = W#16#0090 DBW18 = W#16#0002 DBW20 = W#16#0024 DBW22 = W#16#0024 DBW24 = W#16#0000
RET_VAL	MW2 = W#16#0012

TBL_FIND: Find value in table

Description

You use the instruction to search the memory for patterns. The "TBL_FIND" instruction performs a comparison (CMD parameter) between the source pattern (PATRN parameter) and the entries in the source table (SRC parameter). Starting at the entry specified by the INDX parameter, the instruction searches for the next entry in the table that matches the comparison criteria. The number of the entry is stored in the INDX parameter. If no match is found, the "INDX parameter will point beyond the end of the table and the output of the "TBL_FIND" instruction will be deactivated.

- If CMD = 1, then the "TBL_FIND" instruction searches for the first value that corresponds to the value at the PATRN parameter.
- If CMD = 2, then the "TBL_FIND" instruction searches for the first value that does not correspond to the value at the PATRN parameter.
- The first entry in the table contains the maximum length of the table.
- The second entry in the table contains the first table value.

Note

Make sure that you initialize the first entry (table length) of the table.

Parameter

The following table shows the parameters of the instruction "TBL_FIND":

Parameters	Declaration	Data type	Memory area	Description
SRC	Input	*Pointer	I, Q, M, D	Points to the start address of the table.
PATRN	Input	*Pointer	I, Q, M, D	Points to the pattern to be searched for.

Parameters	Declaration	Data type	Memory area	Description
CMD	Input	BYTE	I, Q, M, D, L, P	Specifies the command: • B#16#01 = equal to • B#16#02 = unequal to
E_TYPE	Input	BYTE	I, Q, M, D, L, P	Specifies the data type of the table entries: B#16#02 = BYTE B#16#04= WORD B#16#05=INT B#16#06=DWORD B#16#07=DINT B#16#08=REAL
RET_VAL	Return	WORD	I, Q, M, D, L, P	Returns the value W#16#0000 if the operation executes without errors.
INDX	InOut	WORD	I, Q, M, D, L	Table index that provides the following information: • Input: Number of the entry at which the search starts. • Output: Number of the entry that matches the specified pattern.
* Pointer in double-word format for area-crossing, indirect register addressing				

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

If any of the following conditions occur, the table values are not changed. The signal state of "BR / ENO" will be set to "0" and the return value will be set accordingly:

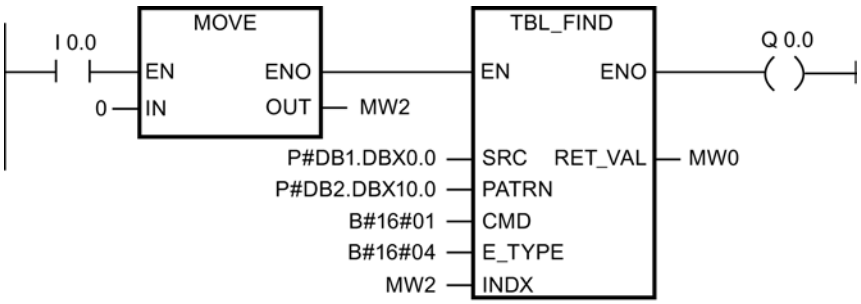
RET_VAL	Explanation
W#16#0007	The value of the INDX parameter is higher than the value of the table length.
W#16#0008	No match was found.
W#16#0009	The E_TYPE and/or CMD parameter(s) is/are invalid.

Example (LAD)

The "TBL_FIND" instruction is executed if the signal state at input I 0.0 is 1 (activated). In this example, because the E_TYPE parameter = "4", data in the table is stored in words, starting at the address pointed to by the SRC parameter. These words are compared with the pattern "5555", which is stored at the address specified by the PATRN parameter. Because CMD = 1, the instruction searches for the first value in the SRC parameter that matches the pattern. The INDX parameter points to the address where the search should begin. After the instruction executes, the INDX parameter specifies the entry number in the table where a match for the pattern was found.

If the "TBL_FIND" instruction is executed without errors, then the signal states of ENO and Q 0.0 will be set to "1", and RET_VAL is set to the value W#16#0000.

The "MOVE" instruction is simply used to reset the value of "MW2" in each cycle.



Before processing:

SRC (table length)	DBW0 = W#16#0004 DBW2 = W#16#1111 DBW4 = W#16#3333 DBW6 = W#16#5555 DBW8 = W#16#7777	The first table entry specifies the length of the find instruction. The value 0004 specifies that the next four entries will be searched.
INDX	MW2 = W#16#0000	INDX is transferred from DW0 (before execution) to DW6 (after execution) because the pattern of DW6 matches the pattern 5555 of PATRN .
PATRN	DBW10 = W#16#5555	

After processing:

INDX	MW2 = W#16#0003	

LIFO: Output last value in the table

Description

The instruction outputs the most recent entry of the "LIFO" table as function value. The number of entries is decremented by one. The "LIFO" table consists of words. You can use the ATT (Page 6760) instruction to enter values in the "LIFO" table.

- The first entry in the table specifies the maximum number of entries in the table (table length).
- The second entry in the table specifies the number of existing entries.
- The third entry in the table contains the first word of data.

Parameter

The following table shows the parameters of the instruction "LIFO":

Parameters	Declaration	Data type	Memory area	Description
TABLE	Input	*Pointer	I, Q, M, D	Points to the start address of the "LIFO" table.
RET_VAL	Return	WORD	I, Q, M, D, L, P	The most recent entry from the "LIFO" table.
* Pointer in double-word format for area-crossing, indirect register addressing				

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

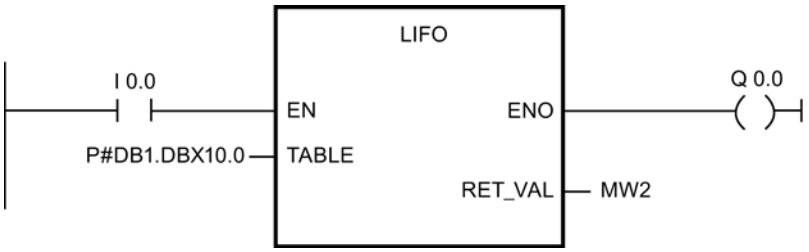
Parameter RET_VAL

If the "LIFO" table is empty (number of entries = 0), then RET_VAL will remain the same and the signal state of "BR / ENO" will be set to "0".

Example (LAD)

The "LIFO" instruction is executed if the signal state at input I 0.0 is 1 (activated). In this example, the most recent entry in the "LIFO" table is returned as the function value (MW 2.0). The number of entries is decremented from 5 to 4.

If "LIFO" is executed without errors, then the signal states of "ENO" and Q 0.0 will be set to "1".



Before processing:

TABLE (table length)	DBW10 = W#16#0006
Number of entries	DBW12 = W#16#0005 DBW14 = W#16#0012 DBW16 = W#16#0029 DBW18 = W#16#0090 DBW20 = W#16#0002 DBW22 = W#16#0024 DBW24 = W#16#0000
RET_VAL	MW2 = W#16#0000

After processing:

TABLE (table length)	DBW10 = W#16#0006
Number of entries	DBW12 = W#16#0004 DBW14 = W#16#0012 DBW16 = W#16#0029 DBW18 = W#16#0090 DBW20 = W#16#0002 DBW22 = W#16#0024 DBW24 = W#16#0000
RET_VAL	MW2 = W#16#0024

TBL: Execute table instruction

Description

The instruction executes the specified instruction (CMD parameter) with the source table and writes the result to the same entry in the table.

- The first entry in the table specifies the maximum number of entries in the table (table length).
- The second entry in the table contains the first table value.
- If the E_TYPE parameter is set to REAL , the CMD value for ones complement is invalid.

Note

Initialize the first entry when you create the table.

Parameter

The following table shows the parameters of the instruction "TBL":

Parameters	Declaration	Data type	Memory area	Description
SRC	Input	*Pointer	I, Q, M, D	Points to the start address of the table.
CMD	Input	BYTE	I, Q, M, D, L, P	Specifies the instruction to be executed. The following instructions are valid: B#16#03 = Ones complement B#16#04 = Delete B#16#05 = Negate B#16#06 = Square root
E_TYPE	Input	BYTE	I, Q, M, D, L, P	Specifies the data type of the table entries. The following data types are valid: B#16#04 = WORD B#16#05 = INT B#16#06 = DWORD B#16#07 = DINT B#16#08 = REAL
RET_VAL	Return	WORD	I, Q, M, D, L, P	Returns the value W#16#0000 if the operation executes without errors.
* Pointer in double-word format for area-crossing, indirect register addressing				

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

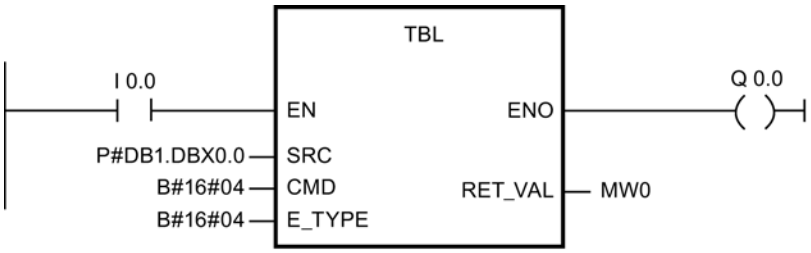
Parameter RET_VAL

If the CMD parameter or the E_TYPE parameter is invalid, and/or if the CMD parameter and the E_TYPE parameter do not match, then the values in the table will remain the same. The signal state of BR / ENO is set to "0" and RET_VAL will be set to the value W#16#0008 .

Example (LAD)

The "TBL" instruction is executed if the signal state at input I 0.0 is 1 (activated). In this example, the SRC parameter points to the address in the data block that will be processed by the instruction. Because the E_TYPE parameter is 4, data in the table will be stored in words starting at the address specified by the SRC parameter. Because CMD is 4 (clear), all the words in the table will be cleared (set to "0") when the "TBL" instruction executes. The table length value of "5" in the first table entry causes the next five table entries to be cleared.

If the "TBL" instruction is executed without error, the signal states of ENO and Q 0.0 will be set to "1", and RET_VAL will be set to the value W#16#0000.



Before processing:

SRC (table length)	DBW0 = W#16#0005 DBW2 = W#16#2000 DBW4 = W#16#3000 DBW6 = W#16#4000 DBW8 = W#16#5000 DBW10 = W#16#6000

After processing:

SRC (table length)	DBW0 = W#16#0005 DBW2 = W#16#0000 DBW4 = W#16#0000 DBW6 = W#16#0000 DBW8 = W#16#0000 DBW10 = W#16#0000

TBL_WRD: Copy value from table

Description

The instruction "TBL_WRD" (copy value from table) copies the entry specified by the INDX parameter from the SRC table to the DEST output. The value of INDX will be incremented if the value is less than the maximum length specified in the first word of the (SRC[0]) table. If the INDX parameter specifies the last table entry, then the output bit will be set to "0" after execution.

- The first entry in the table specifies the maximum number of entries in the table (table length).
- The second entry in the table contains the first table value.

Note

Initialize the first entry when you create the table.

Parameter

The following table shows the parameters of the instruction "TBL_WRD":

Parameter	Declaration	Data type	Memory area	Description
SRC	Input	*Pointer	I, Q, M, D	Points to the start address of the table.
DEST	Input	*Pointer	I, Q, M, D	Points to the start address of the destination.
E_TYPE	Input	BYTE	I, Q, M, D, L, P	Specifies the data type of the table entries. The following data types are valid: • B#16#04 = WORD • B#16#05 = INT • B#16#06 = DWORD • B#16#07 = DINT • B#16#08 = REAL
RET_VAL	Return	WORD	I, Q, M, D, L, P	Returns the value W#16#0000 if the operation executes without errors.
Q	Output	BOOL	Q, M, D, L	Returns the value "0" if the INDX variable contains the last value of the table while calling "TBL_WRD".
INDX	InOut	WORD	I, Q, M, L	Number of the entry to be copied.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

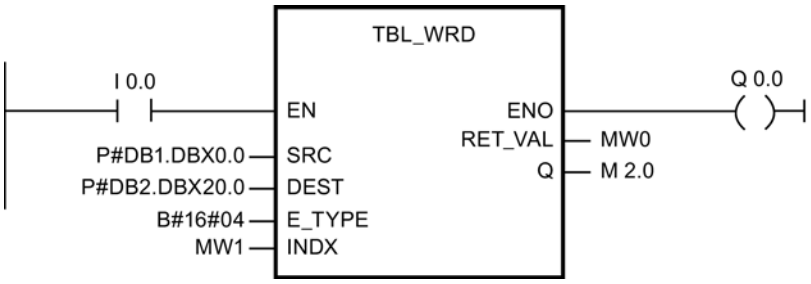
If any of the following conditions occur, "TBL_WRD" will not be executed. The signal state of ENO is set to "0" and the return value will be set accordingly:

RET_VAL	Explanation
W#16#0007	The index is "0".
W#16#0008	E_TYPE is invalid.
W#16#0009	The index is outside the range of the table.

Example

"TBL_WRD" is executed if the signal state at input I 0.0 is 1 (activated). In this example, because E_TYPE = 4 the data word stored in the table entry referenced by SRC is copied to the entry referenced by DEST. The value of INDX points to the table entry to be copied. After "TBL_WRD" executes, the value of INDX is automatically incremented by one entry. If the content of INDX is not the last table entry when "TBL_WRD" is called, output ENO is set to "1" after execution.

If "TBL_WRD" is executed without errors, then the signal states of ENO will be set to "1" and RET_VAL will be set to W#16#0000.



Before processing:

SRC(table length)	DBW0 = W#16#0004 DBW2 = W#16#2000 DBW4 = W#16#3000 DBW6 = W#16#4000 DBW8 = W#16#5000
INDX	MW1 = W#16#0001
DEST	DBW20 = W#16#0000

Before processing:

INDX	MW1 = W#16#0002
DEST	DBW20 = W#16#2000

WRD_TBL: Link value logically with table element and save

Description

The instruction executes the specified instruction (CMD parameter) with the source item and the entry in the table referenced by the INDX parameter. The instruction executes only as long as the INDX parameter is less than the length of the table stored in the first word of the table.

- The first entry in the table specifies the maximum number of entries in the table (table length).
- The second entry in the table contains the first table value.
- If the data type specified by the E_TYPE parameter is REAL, then the CMD parameter can only be "Transfer".

Note

Initialize the first entry when you create the table.

Parameter

The following table shows the parameters of the instruction "WRD_TBL":

Parameters	Declaration	Data type	Memory area	Description
SRC	Input	*Pointer	I, Q, M, D	Points to the source data
TABLE	Input	*Pointer	I, Q, M, D	Points to the start of the table.
CMD	Input	BYTE	I, Q, M, D, L, P	Indicates the instruction to be executed. The following instructions are valid: B#16#0E = Transfer B#16#07 = AND logic operation B#16#08 = OR logic operation B#16#09 = EXCLUSIVE OR logic operation
E_TYPE	Input	BYTE	I, Q, M, D, L, P	Specifies the data type of the table entries. The following data types are valid: B#16#04 = WORD B#16#05 = INT B#16#06 = DWORD B#16#07 = DINT B#16#08 = REAL
RET_VAL	Return	WORD	I, Q, M, D, L, P	Returns a value W#16#0000 if the instruction executes without errors.
Q	Output	BOOL	Q, M, D, L	Is set to "0" if the tag at the INDX parameter contains the number of the last entry in the table.
INDX	InOut	WORD	I, Q, M, D, L	Number of the entry on which the instruction will be executed.
* Pointer in double-word format for area-crossing, indirect register addressing				

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

If any of the following conditions occur, then the "WRD_TBL" instruction will not be executed. The signal state of ENO is set to "0" and the return value will be set accordingly:

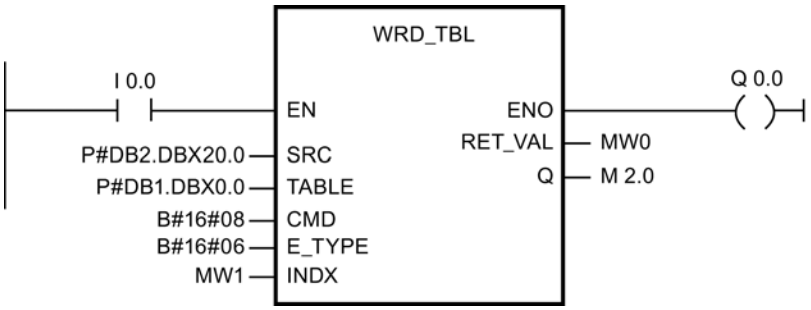
RET_VAL	Explanation
W#16#0007	The index is 0.
W#16#0008	CMD parameter or E_TYPE parameter is invalid or CMD parameter and E_TYPE parameter do not match.
W#16#0009	The index is outside the range of the table.

Example (LAD)

The "WRD_TBL" instruction will be executed if the signal state at input I 0.0 is 1 (activated). Because the E_TYPE parameter in this example is "6", the double word of the data in the table

is saved at the address specified by TABLE . The first word in the table indicates that the table contains three double words. The INDX value points to the table entry that is to be modified. Because CMD is 8, the instruction performs an OR operation with the value to which the INDX parameter points. Because INDX is 2, the second double word (66665544) is ORed with the value to which the SRC parameter (11111111) points. After the instruction executes, the result of the OR operation (77775555) is written back to the table, and the value of INDX will be automatically incremented by one entry. If the INDX parameter points to the last table entry, the "Q" output bit is set to "0" after execution of the instruction. In this example, the value of INDX is not set to the last entry in this table and the "Q" output is therefore set to "1" after execution.

If the "WRD_TBL" instruction is executed without errors, then the signal states of ENO and Q 0.0 will be set to "1", and RET_VAL is set to the value W#16#0000.



Before processing:

SRC	DBD20 = DW#16#11111111
TABLE (table length)	DBW0 = W#16#0003 DBD2 = DW#16#99998877 DBD6 = DW#16#66665544 DBD10 = DW#16#33332222
INDX	MW1 = W#16#0002

After processing:

TABLE (table length)	DBW0 = W#16#0003 DBD2 = DW#16#99998877 DBD6 = DW#16#77775555 DBD10 = DW#16#33332222
INDX	MW1 = W#16#0003

DEV: Calculate standard deviation

Description

The instruction calculates the standard deviation of a group of values stored in a table ("TBL (Page 6767)" instruction). The result will be stored in OUT . The standard deviation is calculated according to the following formula:

$$\text{Standard deviation} = \sqrt{\frac{(N * Sqsum) - Sum^2}{N * (N - 1)}}$$

Where:

- Sum = the sum of the values in the "TBL (Page 6767)" instruction
- N = the number of the values in the "TBL (Page 6767)" instruction
- Sqsum = the sum of all values in the "TBL (Page 6767)" instruction squared

All calculations use IEEE floating point values, with any necessary type conversions being done automatically by the "DEV" instruction.

- The first entry in the table contains the number of entries in the table (table length).
- The second entry in the table contains the first table value.
- The size of the table entries and the calculated value (OUT parameter) are specified by the E_TYPE parameter.

Parameter

The following table shows the parameters of the instruction "DEV":

Parameters	Declaration	Data type	Memory area	Description
TBL	Input	*Pointer	I, Q, M, D	**Points to the start address of a table of values.
OUT	Input	*Pointer	I, Q, M, D	**Points to the address of the computed standard deviation value.
E_TYPE	Input	BYTE	I, Q, M, D, L, P	Specifies the data type of the table entries. The following data types are valid: B#16#05 = INT B#16#07 = DINT B#16#08 = REAL
RET_VAL	Return	WORD	I, Q, M, D, L, P	Returns the value W#16#0000 if the operation executes without errors.
* Double word pointer format for area-crossing register indirect addressing ** Source and destination block must be identical				

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

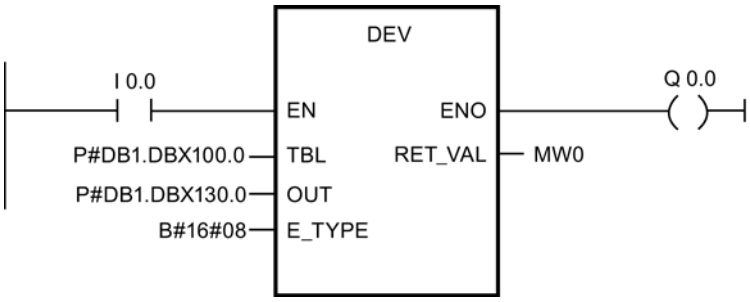
If any of the following conditions occur, then the "DEV" instruction will not be executed. The signal state of BR / ENO will be set to "0" and the return value will be set accordingly:

RET_VAL	Explanation
W#16#0001	Invalid memory type for a parameter of "DEV".
W#16#0002	The E_TYPE parameter is invalid.
W#16#0004	The table length is zero.

Example (LAD)

The "DEV" instruction is executed if the signal state at input I 0.0 is 1 (activated). In this example, there are five table values. This specifies the first word in the table. The table values are of the REAL data type. This fact is indicated by the E_TYPE parameter.

If "DEV" is executed without errors, then the signal states of ENO and Q 0.0 will be set to "1", and RET_VAL will be set to "W#16#0000".



Before processing:

TBL (table length)	DBW100 = W#16#0005 DBD102 = 2.0 DBD106 = 4.0 DBD110 = 8.0 DBD114 = 16.0 DBD118 = 32.0
OUT	DBD130 = 0.0

After processing:

OUT	DBD130 = 12.19836055

CDT: Correlated data tables

Description

The instruction compares an input value (IN parameter) with an existing table of input values (IN_TBL parameter) and locates the first value that is greater than or equal to the input value. Based on its index, the located value is then used to copy the corresponding output value (OUT parameter) to the table of output values (OUT_TBL parameter).

- The values in the input table must be sorted in ascending order. This means that the lowest value is in the first table entry, the highest value is in the last table entry.
- The size of the input value, the table values and output values are defined by the E_TYPE parameter.
- The first entry in the table contains the number of entries of the table (table length).
- The second entry in the table contains the first table value.
- The number of elements in both tables must be identical and be greater than zero.

Note

Initialize the first entry when you create each table.

Parameter

The following table shows the parameters of the instruction "CDT":

Parameters	Declaration	Data type	Memory area	Description
IN_TBL	Input	*Pointer	I, Q, M, D	Points to the start of the input table.
OUT_TBL	Input	*Pointer	I, Q, M, D	Points to the start of the output table.
IN	Input	*Pointer	I, Q, M, D	Points to the input table.
OUT	Input	*Pointer	I, Q, M, D	Points to the output table.
E_TYPE	Input	BYTE	I, Q, M, D, L, P	Specifies the data type of the table entries. The following data types are valid: B#16#05 = INT B#16#07 = DINT B#16#08 = REAL
RET_VAL	Return	WORD	I, Q, M, D, L, P	Returns the value W#16#0000 if the operation executes without errors.
* Pointer in double-word format for area-crossing, indirect register addressing				

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

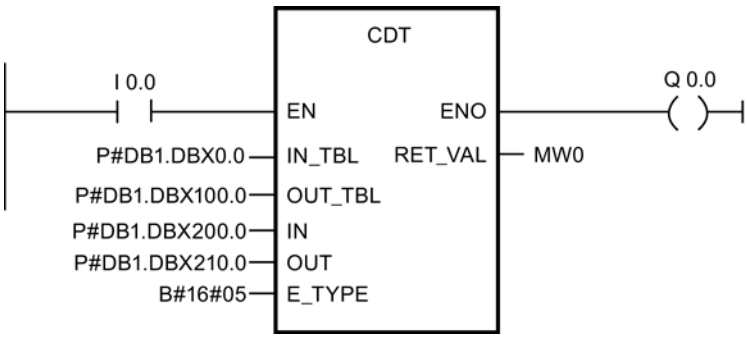
If any of the following conditions occur, CDT will not be executed. The signal state of BR / ENO will be set to "0" and the return value will be set accordingly:

RET_VAL	Explanation
W#16#0001	Invalid memory type specified for a parameter of CDT .
W#16#0002	The E_TYPE parameter is invalid.
W#16#0003	The input and output table lengths do not match.
W#16#0004	The table length is zero.
W#16#0007	No value in the IN_TBL parameter is greater than or equal to the input value.

Example (LAD)

The "CDT" instruction is executed if the signal state at input I 0.0 is 1 (activated). In this example, the IN_TBL parameter and the OUT_TBL parameter each have five table entries. This is specified by the first word of the respective table. The data type of the table values is INTEGER. This fact is indicated by the E_TYPE parameter. The value of the IN parameter is 22. The value of the IN_TBL parameter that is greater than or equal to "22" is "64" which has an index of "5". The corresponding value in OUT_TBL is "25". The value "25" is therefore written in the OUT parameter.

If the "CDT" instruction is executed without errors, then the signal states of ENO and Q 0.0 will be set to "1" and RET_VAL will be set to W#16#0000 .



Before processing:

IN_TBL (table length)	DBW0 = W#16#0005 DBW2 = 2 DBW4 = 4 DBW6 = 8 DBW8 = 16 DBW10 = 64
OUT_TBL (table length)	DBW100 = W#16#0005 DBW102 = 5 DBW104 = 10 DBW106 = 15 DBW108 = 20 DBW110 = 25
IN	DBW200 = 22
OUT	DBW210 = 0

After processing:

OUT DBW210 = 25

TBL_TBL: Link tables

Description

The instruction executes the command specified at the CMD parameter with the corresponding entries of the two source tables (parameters TBL1 and TBL2) and writes the result in the corresponding entries of the destination table (DEST_TBL).

- The data types INT, DINT and REAL are only valid for math instructions.
- The first entry in the table contains the number of entries of the table (table length).
- The number of entries in all of the tables must be the same and be greater than zero.

Note

Initialize the first entry when you create each table.

Parameter

The following table shows the parameters of the instruction "TBL_TBL":

Parameters	Declaration	Data type	Memory area	Description
TBL1	Input	*Pointer	I, Q, M, D	Points to the start of the first source table.
TBL2	Input	*Pointer	I, Q, M, D	Points to the start of the second source table.
DEST_TBL	Input	*Pointer	I, Q, M, D	Points to the start of the destination table.

Parameters	Declaration	Data type	Memory area	Description
CMD	Input	BYTE	I, Q, M, D, L, P	Specifies the command to be performed. The following commands are valid: B#16#07 = AND logic operation B#16#08= OR logic operation B#16#09 = EXCLUSIVE OR logic operation B#16#0a = Add B#16#0b = Subtract B#16#0c = Multiply B#16#0d = Divide
E_TYPE	Input	BYTE	I, Q, M, D, L, P	Specifies the data type of the table entries. The following data types are valid: B#16#04 = WORD B#16#05 = INT B#16#06 = DWORD B#16#07 = DINT B#16#08 = REAL
RET_VAL	Return	WORD	I, Q, M, D, L, P	Returns the value W#16#0000 if the operation executes without errors.
* Pointer in double-word format for area-crossing, indirect register addressing				

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

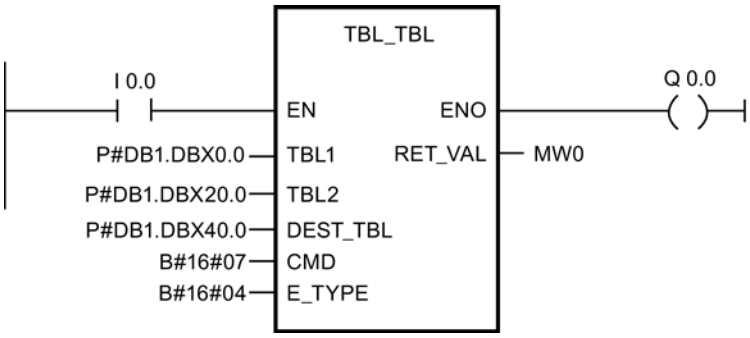
If any of the following conditions occur, then the "TBL_TBL" instruction will not be executed. The signal state of BR / ENO will be set to "0" and the return value will be set accordingly:

RET_VAL	Explanation
W#16#0001	An invalid memory area was specified for a parameter of the "TBL_TBL" instruction.
W#16#0002	The E_TYPE parameter is invalid.
W#16#0003	The input and output table lengths do not match.
W#16#0004	The table length is zero.
W#16#0005	The parameters E_TYPE and CMD do not match.
W#16#0006	The CMD parameter is invalid.

Example (LAD)

The "TBL_TBL" instruction is executed if the signal state at input I 0.0 is 1 (activated). In this example, each of the tables has three table entries. This is specified by the first word of the respective table. The data type of the table values is WORD. E_TYPE specifies this. CMD specifies the command with which TBL1 and TBL2 are to be linked.

If the "TBL_TBL" instruction is executed without errors, then the signal states of ENO and Q 0.0 are set to "1", and RET_VAL is set to the value W#16#0000.



Before processing:

TBL1 (table length)	DBW0 = W#16#0003 DBW2 = W#16#00FF DBW4 = W#16#FF00 DBW6 = W#16#FFFF
TBL2 (table length)	DBW20 = W#16#0003 DBW22 = W#16#1111 DBW24 = W#16#2222 DBW26 = W#16#3333
DEST_TBL (table length)	DBW40 = W#16#0003 DBW42 = W#16#0000 DBW44 = W#16#0000 DBW46 = W#16#0000

After processing:

DST_TBL (table length)	DBW40 = W#16#0003 DBW42 = W#16#0011 DBW44 = W#16#2200 DBW46 = W#16#3333

PACK: Collect/distribute table data

Description

The instruction transfers data located between any addresses and a table. The direction of transfer is specified by the DIR parameter. Each operation of the PACK instruction processes up to five blocks with data (P_DATA1 to P_DATA5). If the DIR parameter has the value "to", the PACK instruction transfers data from the addresses to the specified table. If the DIR parameter has the value "from", data is distributed from the table to the specified addresses.

The rules for transferring data "to" a table are as follows:

- Single bits (BOOL) are transferred to the next available bit in the table.
- 8-bit data types are transferred to the next available byte in the table. When a byte is written to the table, unused bits in the previous byte are filled with zeros.
- 16- and 32-bit data types are transferred to the next available word in the table. When a word is written to the table, unused bits in the previous word are filled with zeros.

The rules for transferring data "from" a table are as follows:

- Sections of a table cannot be skipped.
- All specified data of the BOOL type will be transferred from the table.
- 8-bit data types are transferred to the first available byte in the table. Unused bits in the previous byte of the table are not included in the byte that is transferred from the table.
- 16- and 32-bit data types are transferred from the first available word in the table. Unused bits in the previous word of the table are not included in the a word that is transferred from the table.

The following data types are valid for the ANY pointer and are supported by "PACK": BOOL, WORD, INT, BYTE, DINT, REAL, CHAR, DWORD.

Parameter

The following table shows the parameters of the instruction "PACK":

Parameters	Declaration	Data type	Memory area	Description
TABLE	Input	*Pointer	I, Q, M, D	Points to the start of the table.
P_DATA1	Input	ANY	I, Q, M, D	Points to the start of the block whose data will be transferred.
P_DATA2	Input	ANY	I, Q, M, D	Points to the start of the block whose data will be transferred.
P_DATA3	Input	ANY	I, Q, M, D	Points to the start of the block whose data will be transferred.
P_DATA4	Input	ANY	I, Q, M, D	Points to the start of the block whose data will be transferred.
P_DATA5	Input	ANY	I, Q, M, D	Points to the start of the block whose data will be transferred.
ERR_CODE	Output	WORD	I, Q, M, D, L, P	Returns the value W#16#0000 if the operation executes without errors.
DIR	Static	BOOL	I, Q, M, D, L	Direction of transfer. The following signal states are possible: 0=to, 1=from.
* Pointer in double-word format for area-crossing, indirect register addressing				

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

If any of the following error conditions occurs, then the "PACK" instruction will not be executed. The signal state of BR / ENO will be set to "0" and ERR_CODE will be set accordingly:

ERR_CODE	Explanation
W#16#0001	An invalid memory area was specified for a parameter of the "PACK" instruction.
W#16#0002	The "E_TYPE" instruction is invalid.

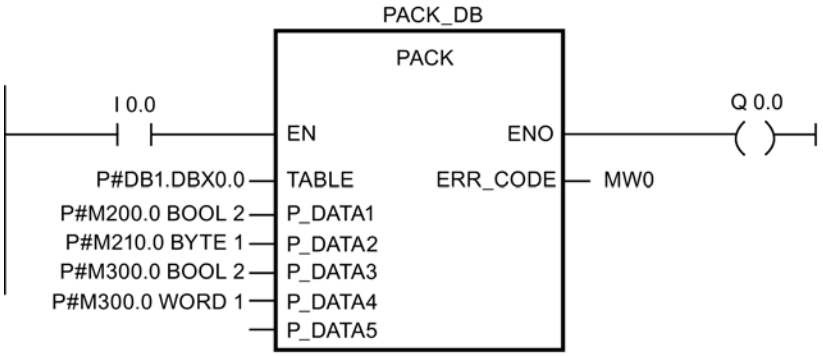
Example (LAD)

"PACK" is executed if the signal state at input I 0.0 is 1 (activated). In this example, there are four blocks with data being transferred "to" the table.

If "PACK" is executed without errors, then the signal states of ENO and Q 0.0 will be set to "1", and ERR_CODE will be set to "W#16#0000".

Note

You can initialize static parameters with the Data Block Editor.



Before processing:

TABLE	DBB0 = B#16#00 DBB1 = B#16#00 DBB2 = B#16#00 DBB3 = B#16#00 DBB4 = B#16#00 DBB5 = B#16#00
P_DATA1	M 200.0 = TRUE M 200.1 = TRUE
P_DATA2	MB210 = B#16#FF
P_DATA3	M 300.0 = TRUE M 300.1 = TRUE

P_DATA4	MW330 = W#16#FFFF
Instance DB PACK_DB	
DIR	DBX58.0 = FALSE

After processing:

TABLE	DBB0 = B#16#03 DBB1 = B#16#FF DBB2 = B#16#03 DBB3 = B#16#00 DBB4 = B#16#FF DBB5 = B#16#FF

Addressing

GEO_LOG: Determine start address of a module

Description

The associated module slot of the module is known from the channel of a signal module. With the "GEO_LOG" you determine the associated start address of the module, which means the smallest I or O address.

If you use the "GEO_LOG" instruction on power modules or modules with packed addresses (ET 200S), then the diagnostics address will be returned.

Parameter

The following table shows the parameters of the instruction "GEO_LOG":

Parameters	Declaration	Data type	Memory area	Description
MASTER	Input	INT	I, Q, M, D, L or constant	Area ID: • 0, if the slot in a centralized configuration is located in: Rack 0 to 3 (for S7-300) or 0 to 21 (for S7-400) • 1 to 32: DP master system ID of the associated field device if the slot is located in a field device on PROFIBUS • 100 to 115: PROFINET IO system ID of the associated field device if the slot is located in a field device on PROFINET
STATION	Input	INT	I, Q, M, D, L or constant	• Number of the rack if the area identifier = 0. • Device number of field device if area ID> 0

Parameters	Declaration	Data type	Memory area	Description
SLOT	Input	INT	I, Q, M, D, L or constant	Slot no.
SUBSLOT	Input	INT	I, Q, M, D, L or constant	Sub-module slot (if no sub-module can be inserted, 0 must be entered here)
RET_VAL	Return	INT	I, Q, M, D, L	Error information
LADDR	Output	WORD	I, Q, M, D, L	Start address of the module Bit 15 of LADDR indicates whether an input address (bit 15 = 0) or an output address (bit 15 = 1) will be present.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error occurred.
8094	No subnet was configured with the specified SUBNETID .
8095	Illegal value for STATION parameter
8096	Illegal value for SLOT parameter
8097	Illegal value for SUBSLOT parameter
8099	The slot is not configured.
809A	The interface module address is not configured for the selected slot.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

LOG_GEO: Determine the slot belonging to a logical address

Description

With the instruction you determine the module slot as well as the offset in the user data address space belonging to a logical address.

Parameter

The following table shows the parameters of the instruction "LOG_GEO":

Parameters	Declaration	Data type	Memory area	Description
LADDR	Input	WORD	I, Q, M, D, L or constant	Any logical address of the module In bit 15 you indicate whether an input address (bit 15 = 0) or an output address (bit 15 = 1) is present.
RET_VAL	Return	INT	I, Q, M, D, L	Error information
AREA	Output	INT	I, Q, M, D, L	Area ID: indicates how the remaining output parameters are to be interpreted.
MASTER	Output	INT	I, Q, M, D, L	Area ID: • 0, if the slot is located in a centralized configuration: Rack 0 to 3 (for S7-300) or 0 to 21 (for S7-400) • 1 to 32: DP master system ID of the associated field device if the slot is located in a field device on PROFIBUS • 100 to 115: PROFINET IO system ID of the associated field device if the slot is located in a field device on PROFINET
STATION	Output	INT	I, Q, M, D, L	No. of the rack when: • Area ID = 0 • Device number of field device if area ID> 0
SLOT	Output	INT	I, Q, M, D, L	Slot no.
SUBSLOT	Output	INT	I, Q, M, D, L	Interface module number
OFFSET	Output	INT	I, Q, M, D, L	Offset in user data address area of the associated module

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter AREA

Value of AREA	System	Meaning of MASTER, STATION, SLOT, SUBSLOT and OFFSET
0	S7-400	MASTER: 0 STATION: Rack no. SLOT: Slot no. SUBSLOT: 0 OFFSET: Difference between the logical address and the logical base address.
1	S7-300	MASTER: 0 STATION: Rack no. SLOT: Slot no. SUBSLOT: 0 OFFSET: Difference between the logical address and the logical base address.
2	PROFIBUS DP	MASTER: DP master ID STATION: Device number SLOT: Slot no. in the station SUBSLOT: 0 OFFSET: Offset in user data address area of the associated module
	PROFINET IO	MASTER: PROFINET IO system ID STATION: Device number SLOT: Slot no. in the station SUBSLOT: Sub-module number OFFSET: Offset in user data address area of the associated module
3	S5-P area	MASTER: 0 STATION: Rack no. SLOT: Slot number of the adaptation capsule SUBSLOT: 0 OFFSET: Address in the S5 x area
4	S5-Q area	MASTER: 0 STATION: Rack no. SLOT: Slot number of the adaptation capsule SUBSLOT: 0 OFFSET: Address in the S5 x area
5	S5-IM3 area	MASTER: 0 STATION: Rack no. SLOT: Slot number of the adaptation capsule OFFSET: Address in the S5 x area
6	S5-IM4 area	MASTER: 0 STATION: Rack no. SLOT: Slot number of the adaptation capsule SUBSLOT: 0 OFFSET: Address in the S5 x area

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error occurred.
8090	Specified logical address invalid
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

RD_LGADR: Determine all logical addresses of a module

Description

You use a logic address of a data block, central submodule or a submodule for PNIO. You use the instruction to determine all the declared logical addresses of this module and the submodule. You have previously configured the assignment of logical addresses to modules and submodules. The "RD_LGADR" instruction enters the determined logical addresses in the PEADDR or PAADDR parameter in ascending order.

Parameter

The following table shows the parameters of the instruction "RD_LGADR":

Parameters	Declaration	Data type	Memory area	Description
IOID	Input	BYTE	I, Q, M, D, L or constant	Address area identifier: • B#16#00: Bit15 of LADDR specifies whether an input (Bit 15=0) or output address (Bit 15=1) is present. • B#16#54 = Peripheral input (PI) • B#16#55 = Peripheral output (PQ)
LADDR	Input	WORD	I, Q, M, D, L or constant	One logical address
RET_VAL	Return	INT	I, Q, M, D, L	Error information
PEADDR	Output	ANY	I, Q, M, D, L	Array for the PI addresses, array elements must be of the WORD data type.
PECOUNT	Output	INT	I, Q, M, D, L	Number of returned PI addresses
PAADDR	Output	ANY	I, Q, M, D, L	Array for the PQ addresses, array elements must be of the WORD data type.
PACOUNT	Output	INT	I, Q, M, D, L	Number of returned PQ addresses

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error occurred.
8090	Specified logical address invalid or illegal value for the IOID parameter.
80A0	Error in the output parameter PEADDR: The data type of the array elements is not WORD. (This error code only exists for S7-400 and CPU 318.)
80A1	Error in the output parameter PAADDR: The data type of the array elements is not WORD. (This error code only exists for S7-400 and CPU 318.)
80A2	Error in the output parameter PEADDR: The specified array could not accommodate all the logical addresses.
80A3	Error in the output parameter PAADDR: The specified array could not accommodate all the logical addresses.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

GADR_LGC: Query logical start address of a module

Description

Based on the channel of a signal module, the corresponding module slot and the offset in the user data address area of the module are known. With the "GADR_LGC" instruction you determine the associated logical base address of the module, which means the smallest I or Q address.

If you use the "GADR_LGC" instruction on power modules or modules with packed addresses (ET 200S), then the diagnostics address will be returned.

Parameter

The following table shows the parameters of the instruction "GADR_LGC":

Parameters	Declaration	Data type	Memory area	Description
SUBNETID	Input	BYTE	I, Q, M, D, L or constant	Area ID: • 0, if the slot is in one of the racks 0 (central rack) or 1 to 21 (expansion racks). • DP master system ID of the corresponding distributed I/O system if the slot is in a distributed I/O device.
RACK	Input	WORD	I, Q, M, D, L or constant	• Number of the rack if the area identifier is 0. • Device number of the distributed I/O device if the area identifier > 0.
SLOT	Input	WORD	I, Q, M, D, L or constant	Slot no.

Parameters	Declaration	Data type	Memory area	Description
SUBSLOT	Input	BYTE	I, Q, M, D, L or constant	Sub-module slot (if no sub-module can be inserted, 0 must be entered here)
SUBADDR	Input	WORD	I, Q, M, D, L or constant	Offset in the user data address area of the module
RET_VAL	Return	INT	I, Q, M, D, L	Error information
IOID	Output	BYTE	I, Q, M, D, L	Address area identifier: • B#16#54 = Peripheral input (PI) • B#16#55 = Peripheral output (PQ) In case of a mixed module, the instruction supplies the area identifier of the lower address. If the addresses are equal, then the instruction will supply the identifier B#16#54.
LADDR	Output	WORD	I, Q, M, D, L	Logical base address of the module

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error occurred.
8093	Invalid value at parameter SUBNETID (the "GADR_LGC" instruction is not valid for PROFINET IO).
8094	No subnet was configured with the specified SUBNETID .
8095	Illegal value for RACK parameter
8096	Illegal value for SLOT parameter
8097	Illegal value for SUBSLOT parameter
8098	Illegal value for SUBADDR parameter
8099	The slot is not configured.
809A	The sub-address of the selected slot is not configured (only possible with central IO devices for CPU and IM).
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

LGC_GADR: Determine the slot belonging to a logical address

Description

With the instruction you determine the module slot as well as the offset in the user data address space belonging to a logical address.

Note

The "LGC_GADR" instruction cannot be used on a module with packed addresses (ET 200S).

Parameters

The following table shows the parameters of the instruction "LGC_GADR":

Parameters	Declaration	Data type	Memory area	Description
IOID	Input	BYTE	I, Q, M, D, L or constant	Address area identifier: • B#16#00: Bit15 of LADDR specifies whether an input (Bit15=0) or output address (Bit15=1) will be present. • B#16#54 = Peripheral input (PI) • B#16#55 = Peripheral output (PQ) If the module is a mixed module, you will have to specify the area identifier of the lower address. If the addresses are identical, you will have to specify B#16#54 .
LADDR	Input	WORD	I, Q, M, D, L or constant	Logical address For a mixed module, the lower of the two addresses must be specified.
RET_VAL	Return	INT	I, Q, M, D, L	Error information
AREA	Output	BYTE	I, Q, M, D, L	Area ID: indicates how the remaining output parameters are to be interpreted.
RACK	Output	WORD	I, Q, M, D, L	Rack no.
SLOT	Output	WORD	I, Q, M, D, L	Slot no.
SUBADDR	Output	WORD	I, Q, M, D, L	Offset in user data address area of the associated module.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Note

If you are using an S7-400H automation system in redundant mode and specify the logical address of a module in a connected DP slave when calling the "LGC_GADR" instruction in the LADDR parameter, then the DP master system ID of the active channel will be supplied in the high byte of the RACK parameter. If no active channel exists, the DP master system ID for the associated DP master system is output to the master CPU.

Parameter AREA

The output parameter AREA specifies how the output parameters RACK, SLOT and SUBADDR are to be interpreted. The following table explains this dependency.

Value of AREA	System	Meaning of RACK, SLOT and SUBADDR
0	S7-400	RACK : Rack no. SLOT : Slot no. SUBADDR : Difference between the logical address and the logical base address.
1	S7-300	RACK : Rack no. SLOT : Slot no. SUBADDR : Difference between the logical address and the logical base address.
2	DP	RACK : (low byte): Device number RACK : (high byte): DP master ID SLOT : Slot no. in the station SUBADDR : Offset in user data address area of the associated module
3	S5-P area	RACK : Rack no. SLOT : Slot number of the adaptation capsule SUBADDR : Address in the S5 x area
4	S5-Q area	RACK : Rack no. SLOT : Slot number of the adaptation capsule SUBADDR : Address in the S5 x area
5	S5-IM3 area	RACK : Rack no. SLOT : Slot number of the adaptation capsule SUBADDR : Address in the S5 x area
6	S5-IM4 area	RACK : Rack no. SLOT : Slot number of the adaptation capsule SUBADDR : Address in the S5 x area

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error occurred.
8090	Specified logical address invalid or illegal value for the IOID parameter.
8093	This instruction is not permitted for the module selected by the parameters IOID and LADDR :
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

Additional functions

iSlave

SET_ADDR: Setting network address as own iSlave

Description

You use this instruction to modify the MAC address of an iSlave DP interface. The addressing is performed via the diagnostics address of the DP interface. For the DP master, the DP slave with the oldest address is discarded from the bus and a DP slave with the new address is included in the bus.

- If the address is already assigned by a station in the DP network, both DP slaves will be discarded from the network.
- If the address is also used as MPI address for the programming device connection, access to the device is possibly disabled. The information about whether the devices are at all accessible is displayed in the "Accessible Stations" dialog.

The network address is not written in the SDB (system data block) and will not be loaded in the work memory. The network address is maintained when the CPU goes to STOP and a warm restart is effected. After a reset or a cold restart, the originally configured address will be used.

The instruction cannot be used for PROFINET IO, because the Ethernet address can be defined as disjoint. If the instruction is used with PROFINET IO, then the error code 16#809B will be output.

Functional description

The "SET_ADDR" instruction works asynchronously, which means its execution extends over multiple calls. You start the job by calling "READ_DBL" with REQ =1.

The output parameters RET_VAL and BUSY indicate the status of the job.

Parameter

The following table shows the parameters of the "SET_ADDR" instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ=1: Function call
LADDR	Input	WORD	I, Q, M, D, L or constant	Logical base address
ADDR	Input	BYTE	I, Q, M, D, L or constant	New device address
RET_VAL	Return	INT	I, Q, M, D, L	Error information
BUSY	Output	BOOL	I, Q, M, D, L	BUSY=1: The job is not yet complete.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error occurred.
700x	The job is not yet complete. Parameter BUSY=1
8091	New device address (ADDR) is invalid.
8092	Station is not an individual slave address of the I-slave
8093	LADDR is invalid or no interface.
809B	Instruction cannot be executed (for example, the interface is not a DP slave or is not active).
80C3	Insufficient resources (for example, multiple calling of instructions with different parameters).
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

9.8.2.4 Technology

PID Control

PID basic functions

CONT_C

Description CONT_C

The CONT_C instruction is used on SIMATIC S7 automation systems to control technical processes with continuous input and output variables. You can assign parameters to enable or disable sub-functions of the PID controller and adapt it to the process. In addition to the functions in the setpoint and process value branches, the instruction implements a complete

PID controller with continuous output value output and the option of manually influencing the value of the output value.

Application

You can use the controller as a PID fixed setpoint controller, or in multi-loop control systems, also as a cascade, blending or ratio controller. The functions of the controller are based on the PID control algorithm of the sampling controller with an analog signal, if necessary extended by including a pulse shaper stage to generate pulse-width modulated output signals for two or three step controllers with proportional actuators.

Call

The CONT_C instruction has an initialization routine that is run through when input parameter COM_RST = TRUE is set. During initialization, the integral action is set to the initialization value I_ITVAL. All the signal outputs are set to zero. COM_RST = FALSE has to be set after the initialization routine has been completed.

The calculation of the values in the control blocks is only correct if the block is called at regular intervals. You should therefore call the control blocks in a cyclic interrupt OB (OB 30 to OB 38). Enter the sampling time in the CYCLE parameter.

If you call the instruction CONT_C as a multiple instance DB, no technology object is created. No parameter assignment interface or commissioning interface is available. You must assign parameters for CONT_C directly in the multiple instance DB and commission it via a watch table.

Error information

The error message word RET_VAL is not evaluated by the block.

See also

Differences compared to CONT_C S7-300/400 (Page 5031)

How CONT_C works

Setpoint branch

The setpoint is entered in floating-point format at the SP_INT input.

Process value branch

The process value can be input in I/O or floating-point format. The function CRP_IN converts the I/O value PV_PER to a floating-point format -100 to +100 % in accordance with the following rule:

Output of CRP_IN = PV_PER * 100 / 27648

The PV_NORM function scales the output of CRP_IN according to the following rule:

Output of PV_NORM = (output of CRP_IN) *PV_FAC + PV_OFF

PV_FAC has a default of 1 and PV_OFF a default of 0.

Forming the error signal

The difference between the setpoint and process value is the error signal. To suppress a minor sustained oscillation due to manipulated variable quantization (e.g. with a pulse width modulation with PULSEGEN), the error signal is applied to a dead band (DEADBAND). With DEADB_W = 0, the dead band is switched off.

PID Algorithm

The PID algorithm operates as a position algorithm. The proportional, integral (INT), and differential (DIF) actions are connected in parallel and can be activated or deactivated individually. This allows P, PI, PD, and PID controllers to be configured. Pure I controllers are also possible.

Manual value processing

It is possible to switch over between manual and automatic mode. In manual mode, the manipulated variable is corrected to a manually selected value.

The integral action (INT) is set internally to LMN - LMN_P - DISV and the derivative action (DIF) is set to 0 and synchronized internally. Changeover to automatic mode is therefore bumpless.

Manipulated value processing

You can use the LMNLIMIT function to limit the manipulated value to selected values. Alarm bits indicate when a limit is exceeded by the input variable.

The LMN_NORM function normalizes the output of LMNLIMIT according to the following rule:

LMN = (output of LMNLIMIT) * LMN_FAC + LMN_OFF

LMN_FAC has a default of 1 and LMN_OFF a default of 0.

The manipulated value is also available in I/O format. The CRP_OUT function converts the LMN floating-point value to a I/O value according to the following rule:

LMN_PER = LMN * 27648 / 100

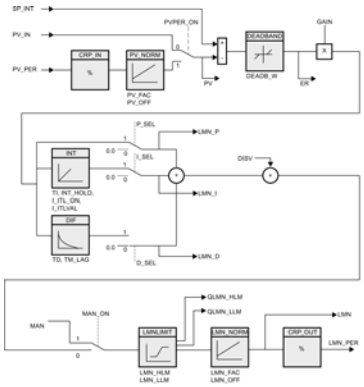
Feedforward control

A disturbance variable can be added at the DISV input.

See also

Differences compared to CONT_C S7-300/400 (Page 5031)

CONT_C block diagram



See also

Differences compared to CONT_C S7-300/400 (Page 5031)

Input parameter CONT_C

Table 9-184

Parameters	Data type	Default	Description
COM_RST	BOOL	FALSE	The instruction has an initialization routine that is processed when the "Restart" input is set.
MAN_ON	BOOL	TRUE	If the input "Enable manual mode" is set then the control loop is interrupted. A manual value is set as the manipulated value.
PVPER_ON	BOOL	FALSE	If the process value is to be read in from the I/Os, the PV_PER input must be interconnected with the I/Os and the "Enable process value I/Os" input must be set.
P_SEL	BOOL	TRUE	The PID actions can be switched on and off individually in the PID algorithm. P-action is on when the "Enable P-action" input is set.
I_SEL	BOOL	TRUE	The PID actions can be switched on and off individually in the PID algorithm. I action is on when the input "I-action on" is set.
INT_HOLD	BOOL	FALSE	The output of the integral action can be frozen. For this the input "I-action hold" must be set.
I_ITL_ON	BOOL	FALSE	The output of the integral action can be set at the I_ITLVAL input. For this the input "Set I-action" must be set.
D_SEL	BOOL	FALSE	The PID actions can be switched on and off individually in the PID algorithm. D-action is on when the input "Enable D-action" is set.
CYCLE	TIME	T#1s	The time between block calls must be constant. The "Sampling time" input specifies the time between block calls. CYCLE >= 1ms

Parameters	Data type	Default	Description
SP_INT	REAL	0.0	The input "Internal setpoint" is used to specify a setpoint. Permissible are values from -100 to 100 % or a physical variable 1).
PV_IN	REAL	0.0	At the "Process value input" you can assign parameters to a commissioning value or you can interconnect an external process value in floating-point format. Permissible are values from -100 to 100 % or a physical variable 1).
PV_PER	WORD	W#16#0000	The process value in I/O format is interconnected with the controller at the "Process value I/0" input.
MAN	REAL	0.0	The "Manual value" input is used to set a manual value using the operator interface functions. Permissible are values from -100 to 100 % or a physical variable 2).
GAIN	REAL	2.0	The "Proportional gain" input specifies controller amplification.
TI	TIME	T#20s	The "Integration time" input determines the time response of the integral action. TI >= CYCLE
TD	TIME	T#10s	The "Derivative action time" input determines the time response of the derivative action. TD >= CYCLE
TM_LAG	TIME	T#2s	Time lag of the D-action The algorithm of the D-action contains a delay for which parameters can be assigned at the input "Time lag of the D-action". TM_LAG >= CYCLE/2
DEADB_W	REAL	0.0	A dead band is applied to the system deviation. The "Dead band width" input determines the size of the dead band. DEADB_W >= 0.0 (%) or a physical variable 1)
LMN_HLM	REAL	100.0	The manipulated value is always restricted to a high limit and low limit. The "High limit of manipulated value" input specifies the high limit. Permissible are real values starting at LMN_LLM or a physical variable 2).
LMN_LLM	REAL	0.0	The manipulated value is always restricted to a high limit and low limit. The "Low limit of manipulated value" input specifies the low limit. Permissible are real values up to LMN_HLM or a physical variable 2).
PV_FAC	REAL	1.0	The "Process value factor" input is multiplied by the process value. The input is used to scale the process value range.
PV_OFF	REAL	0.0	The input "Process value offset" is added to the process value. The input is used to scale the process value range.
LMN_FAC	REAL	1.0	The "Manipulated value factor" input is multiplied with the manipulated value. The input is used to scale the manipulated value range.
LMN_OFF	REAL	0.0	The input "Manipulated value offset" is added to the process value. The input is used to scale the manipulated value range.

Parameters	Data type	Default	Description
I_ITLVAL	REAL	0.0	The output of the integral action can be set at the I_ITL_ON input. The initialization value is applied to the input "Initialization value of the I-action." Permissible are values of -100.0 to 100.0 (%) or a physical variable 2).
DISV	REAL	0.0	For feedforward control, the disturbance variable is interconnected to the "Disturbance variable" input. Permissible are values of -100.0 to 100.0 (%) or a physical variable 2).

1) Parameters in the setpoint and process value branches with the same unit

2) Parameters in the manipulated value branch with the same unit

See also

Differences compared to CONT_C S7-300/400 (Page 5031)

Output parameters CONT_C

Table 9-185

Parameter	Data type	Default	Description
LMN	REAL	0.0	The effective "Manipulated value" is output in floating point format at the "Manipulated value" output.
LMN_PER	WORD	W#16#0000	The manipulated value in I/O format is interconnected on the input "Manipulated value I/O" with the controller.
QLMN_HLM	BOOL	FALSE	The manipulated value is always restricted to a high limit and low limit. The output "High limit of manipulated value reached" indicates that the high limit has been reached.
QLMN_LLM	BOOL	FALSE	The manipulated value is always restricted to a high limit and low limit. The output "Low limit of manipulated value reached" indicates that the low limit has been reached.
LMN_P	REAL	0.0	The "P-action" output contains the proportional action of the manipulated variable.
LMN_I	REAL	0.0	The "I-action" output contains the integral action of the manipulated variable.
LMN_D	REAL	0.0	The "D-action" output contains the derivative action of the manipulated variable.
PV	REAL	0.0	The effective process value is output at the "Process value" output.
ER	REAL	0.0	The effective system deviation is output at the "Error signal" output.

See also

Differences compared to CONT_C S7-300/400 (Page 5031)

CONT_S

Description CONT_S

The CONT_S instruction is used on SIMATIC S7 automation systems to control technical processes with binary output value output signals for actuators with integrating behavior. During parameter assignment, you can activate or deactivate sub-functions of the PI step controller to adapt the controller to the controlled system. In addition to the functions in the process value branch, the instruction implements a complete proportional-plus-integral-action controller with binary output value output and the option of manually influencing the value of the output value. The step controller operates without a position feedback signal.

Application

You can use the controller as a PI fixed setpoint controller or in secondary control loops in cascade, blending or ratio controllers, however you cannot use it as the primary controller. The functions of the controller are based on the PI control algorithm of the sampling controller supplemented by the functions for generating the binary output signal from the analog actuating signal.

Call

The CONT_S instruction has an initialization routine that is run through when input parameter COM_RST = TRUE is set. All the signal outputs are set to zero. COM_RST = FALSE has to be set after the initialization routine has been completed.

The calculation of the values in the control blocks is only correct if the block is called at regular intervals. You should therefore call the control blocks in a cyclic interrupt OB (OB 30 to OB 38). Enter the sampling time in the CYCLE parameter.

If you call the instruction CONT_S as a multiple instance DB, no technology object is created. No parameter assignment interface or commissioning interface is available. You must assign parameters for CONT_S directly in the multiple instance DB and commission it via a watch table.

Error information

The error message word RET_VAL is not evaluated by the block.

See also

Differences compared to CONT_S S7-300/400 (Page 5031)

Mode of operation CONT_S

Setpoint branch

The setpoint is entered in floating-point format at the SP_INT input.

Process value branch

The process value can be input in I/O or floating-point format. The function CRP_IN converts the I/O value PV_PER to a floating-point format -100 to +100 % in accordance with the following rule:

Output of CRP_IN = PV_PER * 100 / 27648

The PV_NORM function normalizes the output of CRP_IN according to the following rule:

Output of PV_NORM = (output of CRP_IN) * PV_FAC + PV_OFF

PV_FAC has a default of 1 and PV_OFF a default of 0.

Forming the error signal

The difference between the setpoint and process value is the error signal. To suppress a small constant oscillation due to the manipulated variable quantization (for example, due to a limited resolution of the manipulated value by the control valve), a dead band is applied to the error signal (DEADBAND). With DEADB_W = 0, the dead band is switched off.

PI step algorithm

The instruction operates without position feedback. The I-action of the PI algorithm and the assumed position feedback signal are calculated in **one** integral action (INT) and compared with the remaining P-action as a feedback value. The difference is applied to a three-step element (THREE_ST) and a pulse shaper (PULSEOUT) that generates the pulses for the control valve. The switching frequency of the controller can be reduced by adapting the response threshold of the three-step element.

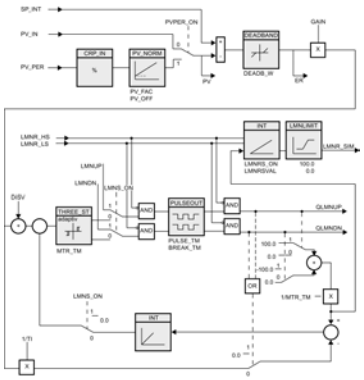
Feedforward control

A disturbance variable can be added at the DISV input.

See also

Differences compared to CONT_S S7-300/400 (Page 5031)

Block diagram CONT_S



See also

Differences compared to CONT_S S7-300/400 (Page 5031)

Input parameters CONT_S

Table 9-186

Parameters	Data type	Default	Description
COM_RST	BOOL	FALSE	The block has an initialization routine that is processed when the "Restart" input is set.
LMNR_HS	BOOL	FALSE	The signal "Control valve at high endstop" is interconnected at the input "High endstop signal of position feedback". LMNR_HS=TRUE means: The control valve is at high endstop.
LMNR_LS	BOOL	FALSE	The signal "Control valve at low endstop" is interconnected on the input "Low endstop signal of position feedback". LMNR_LS=TRUE means The control valve is at low endstop.
LMNS_ON	BOOL	FALSE	Manipulated value signal processing is switched to manual mode at the "Enable manual mode of manipulated signal".
LMNUP	BOOL	FALSE	The output signal QLMNUP is operated in manual mode of the manipulated value signals at the input "Manipulated value signal up".
LMNDN	BOOL	FALSE	The output signal QLMNDN is operated in manual mode of the manipulated value signals at the input "Manipulated value signal down"
PVPER_ON	BOOL	FALSE	If the process value is to be read from the I/O then the input PV_PER must be interconnected with the I/O and the input "Enable process value I/O" must be set.
CYCLE	TIME	T#1s	The time between block calls must be constant. The "Sampling time" input specifies the time between block calls. CYCLE >= 1ms
SP_INT	REAL	0.0	The input "Internal setpoint" is used to specify a setpoint. Permissible are values from -100 to 100 % or a physical variable [1].

Parameters	Data type	Default	Description
PV_IN	REAL	0.0	At the "Process value input" you can assign parameters to a commissioning value or you can interconnect an external process value in floating-point format. Permissible are values from -100 to 100 % or a physical variable [1].
PV_PER	WORD	W#16#0000	The process value in I/O format is interconnected with the controller at the "Process value I/O" input.
GAIN	REAL	2.0	The "Proportional gain" input specifies controller amplification.
TI	TIME	T#20s	The "Integration time" input determines the time response of the integral action. TI >= CYCLE
DEADB_W	REAL	1.0	A dead band is applied to the system deviation. The "Dead band width" input determines the size of the dead band. Permissible are values from 0 to 100 % or a physical variable [1].
PV_FAC	REAL	1.0	The "Process value factor" input is multiplied by the process value. The input is used to scale the process value range.
PV_OFF	REAL	0.0	The input "Process value offset" is added to the process value. The input is used to scale the process value range.
PULSE_TM	TIME	T#3s	You can assign a minimum pulse time at the parameter "Minimum pulse time". PULSE_TM >= CYCLE
BREAK_TM	TIME	T#3s	You can assign a minimum break time at the parameter "Minimum break time". BREAK_TM >= CYCLE
MTR_TM	TIME	T#30s	The time required by the actuator to move from limit stop to limit stop is entered at the "Motor actuating time" parameter. MTR_TM >= CYCLE
DISV	REAL	0.0	For feedforward control, the disturbance variable is interconnected to the "Disturbance variable" input. Permissible are values from -100 to 100 % or a physical variable [2].

[1] Parameters in setpoint and process value branches with identical unit

[2] Parameters in the manipulated value branch with same unit

See also

Differences compared to CONT_S S7-300/400 (Page 5031)

Output parameters CONT_S

Table 9-187

Parameters	Data type	Default	Description
QLMNUP	BOOL	FALSE	If the output "Manipulated value signal up" is set then the control valve should be open.
QLMNDN	BOOL	FALSE	If the output "Manipulated value signal down" is set then the control valve should be closed.
PV	REAL	0.0	The effective process value is output at the "Process value" output.
ER	REAL	0.0	The effective system deviation is output at the "Error signal" output.

See also

Differences compared to CONT_S S7-300/400 (Page 5031)

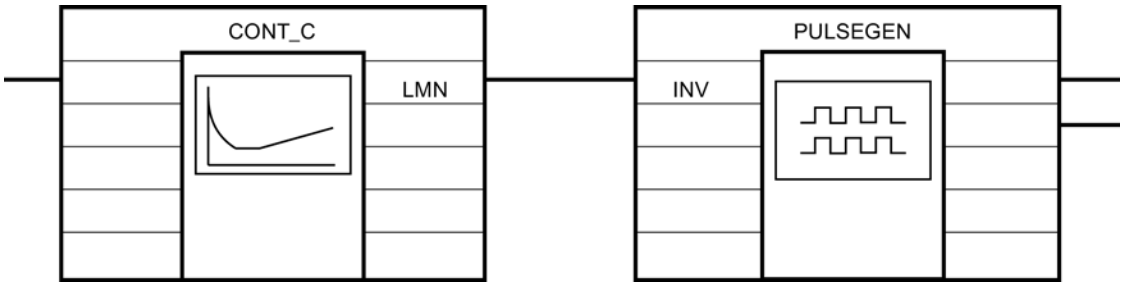
PULSEGEN

Description PULSEGEN

The instruction PULSEGEN serves as the structure of a PID controller with impulse output for proportional actuators. PULSEGEN transforms the input value INV (= LMN of the PID controller) through modulation of the impulse width in an impulse sequence with a constant period duration, which corresponds with the cycle time with which the input value is updated.

Application

You can use the PULSEGEN instruction to configure two- or three-step PID controllers with pulse width modulation. The function is normally used in conjunction with the continuous controller CONT_C.



Call

The PULSEGEN instruction has an initialization routine that is run through when input parameter COM_RST = TRUE is set. All the signal outputs are set to zero. COM_RST = FALSE has to be set after the initialization routine has been completed.

The calculation of the values in the control blocks is only correct if the block is called at regular intervals. You should therefore call the control blocks in a cyclic interrupt OB (OB 30 to OB 38). Enter the sampling time in the CYCLE parameter.

Responses in the event of an error

The error message word RET_VAL is not evaluated by the block.

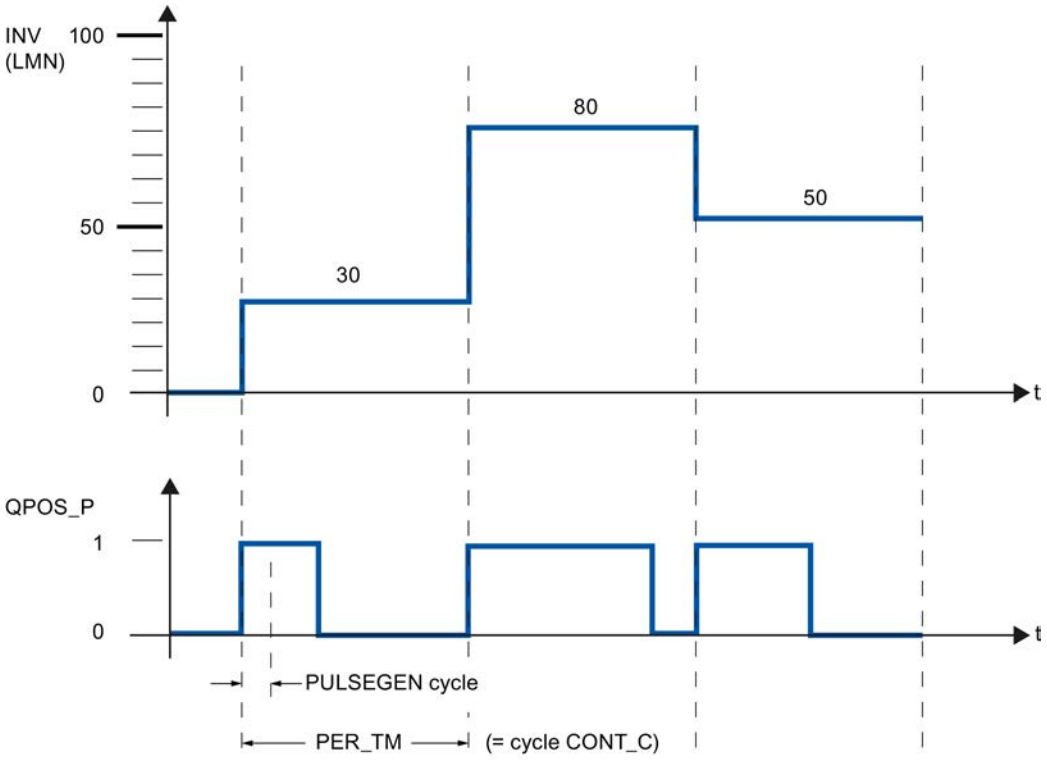
See also

Differences compared to PULSEGEN S7-300/400 (Page 5032)

Mode of operation PULSEGEN

Impulse width modulation

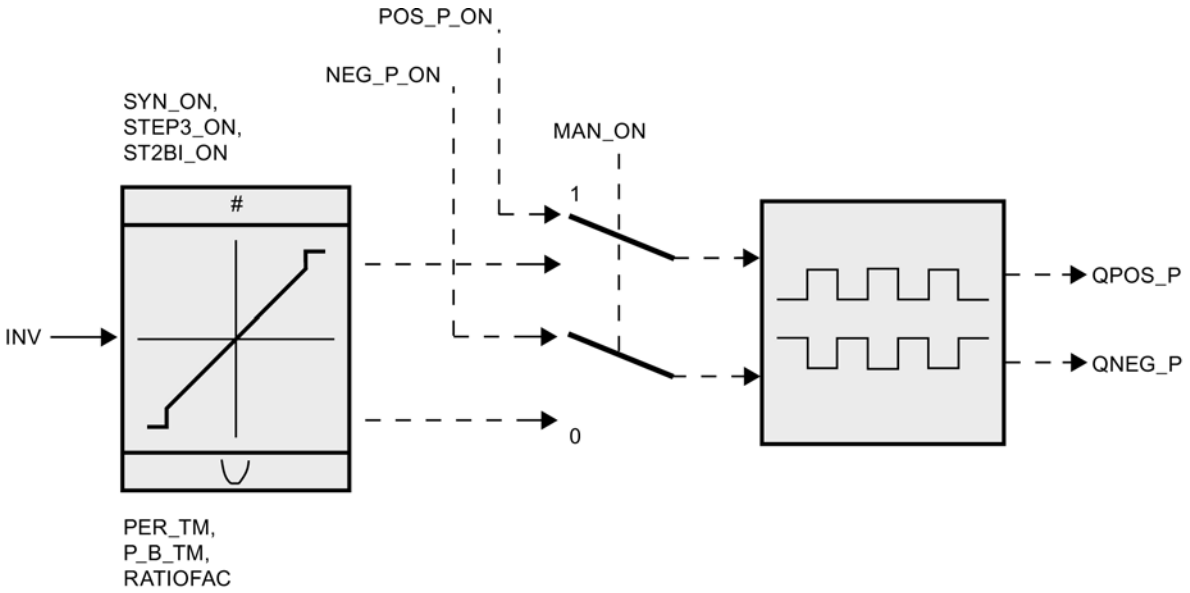
The duration of a pulse per period duration is proportional to the input variable. The cycle assigned via PER_TM is not identical to the processing cycle of the PULSEGEN instruction. Rather, a PER_TM cycle is made up of several processing cycles of the PULSEGEN instruction, whereby the number of PULSEGEN calls per PER_TM cycle determines the accuracy of the pulse width.



An input variable of 30% and 10 PULSEGEN calls per PER_TM mean the following:

- "One" at the QPOS_P output for the first three calls of PULSEGEN (30% of 10 calls)
- "Zero" at the QPOS_P output for seven further calls of PULSEGEN (70% of 10 calls)

Block diagram



Accuracy of the manipulated value

With a "Sampling ratio" of 1:10 (CONT_C calls to PULSEGEN calls) the accuracy of the manipulated value in this example is restricted to 10%, in other words, set input values INV can only be simulated by a pulse duration at the QPOS_P output in steps of 10 %.

The accuracy is increased as the number of PULSEGEN calls per CONT_C call is increased.

If PULSEGEN is called, for example, 100 times more often than CONT_C, a resolution of 1 % of the manipulated value range is achieved.

Note

The reduction ratio of the call frequency must be programmed by the user.

Automatic synchronization

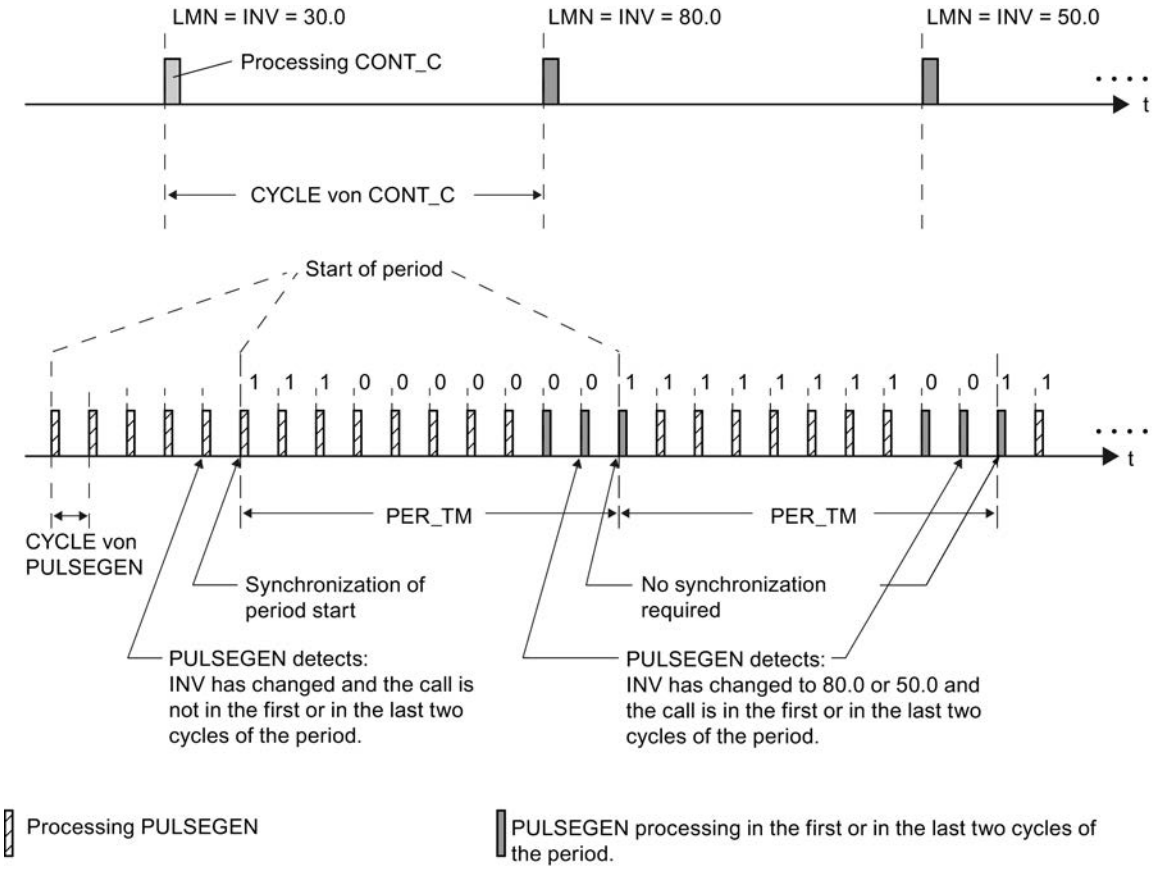
It is possible to automatically synchronize the pulse output with the instruction that updates the input variable INV (e.g. CONT_C). This ensures that a change in the input variable is output as quickly as possible as a pulse.

The pulse shaper evaluates the input value INV at intervals corresponding to the period duration PER_TM and converts the value into a pulse signal of corresponding length.

Since, however, INV is usually calculated in a slower cyclic interrupt class, the pulse shaper should start the conversion of the discrete value into a pulse signal as soon as possible after the updating of INV.

To allow this, the block can synchronize the start of the period using the following procedure:

If INV changes and if the block call is not in the first or last two call cycles of a period, a synchronization is performed. The pulse duration is recalculated and in the next cycle is output with a new period.



The automatic synchronization is switched off, if SYN_ON = FALSE.

Note

The start of a new period and subsequent synchronization usually leads to a certain imprecision when the old value of INV (i.e. of LMN) is mapped to the pulse signal.

See also

Differences compared to PULSEGEN S7-300/400 (Page 5032)

Mode of operation PULSEGEN

Modes

Depending on the parameters assigned to the pulse shaper, PID controllers with a three-step output or with a bipolar or unipolar two-step output can be configured. The following table illustrates the setting of the switch combinations for the possible modes.

Mode	MAN_ON	STEP3_ON	ST2BI_ON
Three-step control	FALSE	TRUE	Any
Two-step control with bi-polar Manipulating range (-100 % to 100 %)	FALSE	FALSE	TRUE
Two-step control with unipolar Manipulating range (0 % to 100 %)	FALSE	FALSE	FALSE
Manual mode	TRUE	Any	Any

Manual mode in two/three-step control

In the manual mode (MAN_ON = TRUE), the binary outputs of the three-step or two-step controller can be set using the signals POS_P_ON and NEG_P_ON regardless of INV.

Control	POS_P_ON	NEG_P_ON	QPOS_P	QNEG_P
Three-step control	FALSE	FALSE	FALSE	FALSE
	TRUE	FALSE	TRUE	FALSE
	FALSE	TRUE	FALSE	TRUE
	TRUE	TRUE	FALSE	FALSE
Two-step control	FALSE	Any	FALSE	TRUE
	TRUE	Any	TRUE	FALSE

See also

Differences compared to PULSEGEN S7-300/400 (Page 5032)

Three-step control

Three-step control

In "Three-step control" mode, it is possible to generate three statuses of the actuating signal. For this, the status values of the binary output signals QPOS_P and QNEG_P are assigned

to the respective operating statuses of the actuator. The table shows the example of a temperature control:

Output signals	Heat	Off	Cool
QPOS_P	TRUE	FALSE	FALSE
QNEG_P	FALSE	FALSE	TRUE

The pulse duration is calculated from the input variable via a characteristic curve. The form of the characteristic curve is defined by the minimum pulse duration or minimum interval and the ratio factor. The normal value for the ratio factor is 1.

The "doglegs" in the curves are caused by the minimum pulse duration or minimum interval.

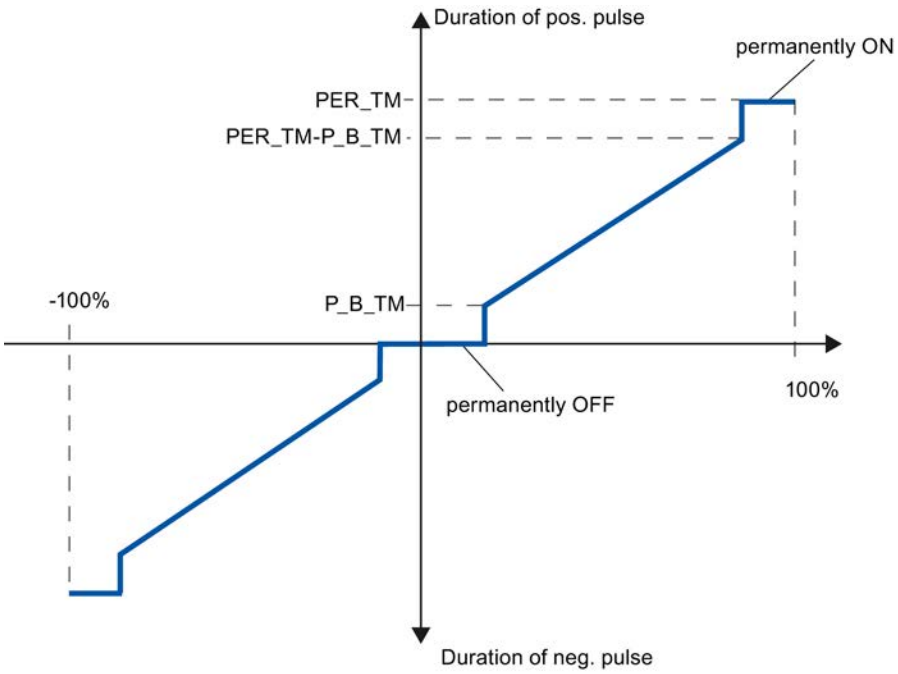
Minimum pulse duration or minimum interval

A correctly assigned minimum pulse duration or minimum interval P_B_TM can prevent short on/off times, which reduce the working life of switching elements and actuators. Small absolute values of input variable LMN that would otherwise generate a pulse duration shorter than P_B_TM are suppressed. Large input values that would generate a pulse duration longer than PER_TM - P_B_TM are set to 100% or -100%.

The duration of positive or negative pulses is calculated by multiplying the input variable (in %) by the period duration:

Pulse duration = INV / 100 * PER_TM

The following figure shows a symmetrical characteristic curve of the three-step controller (ratio factor = 1).



Asymmetrical three-step control

Using the ratio factor RATIOFAC, the ratio of the duration of positive to negative pulses can be changed. In a thermal process, for example, this would allow different system time constants for heating and cooling.

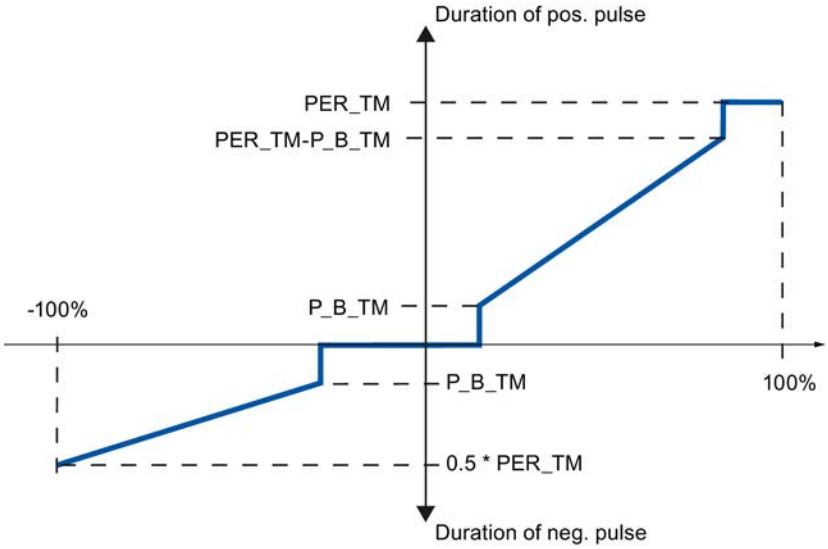
Ratio factor < 1

The pulse duration at the negative pulse output, calculated by multiplying the input variable by the period duration, is multiplied by the ratio factor.

Positive pulse duration = INV /100 * PER_TM

Negative pulse duration = INV / 100 * PER_TM * RATIOFAC

The following figure shows the asymmetrical characteristic curve of the three-step controller (ratio factor = 0.5):



Ratio factor > 1

The pulse duration at the positive pulse output, calculated by multiplying the input variable by the period duration, is divided by the ratio factor.

Positive pulse duration = INV / 100 * PER_TM / RATIOFAC

Negative pulse duration = INV / 100 * PER_TM

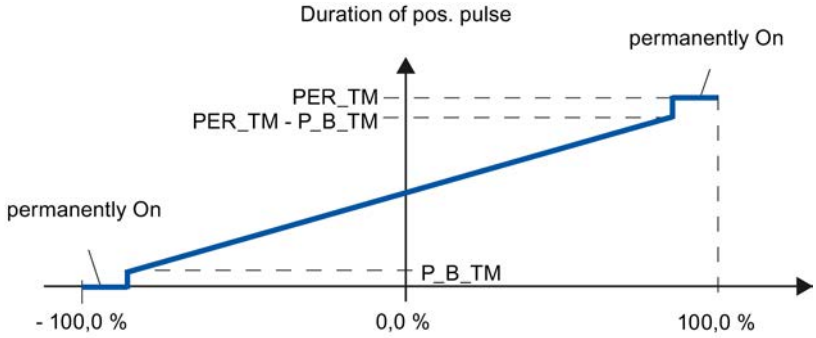
See also

Differences compared to PULSEGEN S7-300/400 (Page 5032)

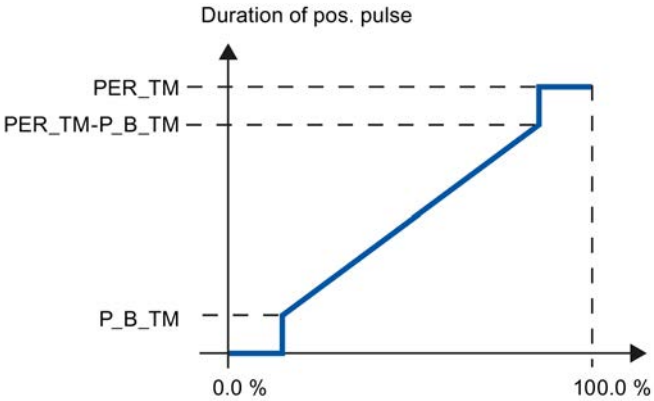
Two-step control

In two-step control, only the positive pulse output QPOS_P of PULSEGEN is connected to the on/off actuator. Depending on the manipulated value range used, the two-step controller has a bipolar or a unipolar manipulated value range.

Two-step control with bipolar manipulated variable range (-100% to 100%)



Two-step control with unipolar manipulated variable range (0% to 100%)



The negated output signal is available at QNEG_P if the connection of the two-step controller in the control loop requires a logically inverted binary signal for the actuating pulses.

Pulse	Actuator On	Actuator Off
QPOS_P	TRUE	FALSE
QNEG_P	FALSE	TRUE

See also

Differences compared to PULSEGEN S7-300/400 (Page 5032)

Input parameters PULSEGEN

The values of the input parameters are not limited in the block. There is no parameter check.

Table 9-188

Parameters	Data type	Default	Description
INV	REAL	0.0	At the input parameter "Input variable" an analog manipulated variable is connected. Values from -100 to 100 % are permitted.
PER_TM	TIME	T#1s	At the parameter "Period duration" the constant period duration of the pulse width modulation is entered. This corresponds to the sampling time of the controller. The ratio between the sampling time of the pulse shaper and the sampling time of the controller determines the accuracy of the pulse width modulation. PER_TM >=20*CYCLE
P_B_TM	TIME	T#50 ms	You can assign a minimum pulse/break time at the parameter "Minimum pulse/break time". P_B_TM >= CYCLE
RATIOFAC	REAL	1.0	Using the "Ratio factor" input parameter the ratio of the duration of positive to negative pulses can be changed. In a thermal process, this would, for example, allow different time constants for heating and cooling to be compensated (for example, in a process with electrical heating and water cooling). Values from 0.1 to 10.0 are permitted.
STEP3_ON	BOOL	TRUE	At the input parameter "Enable three-step control" the appropriate mode is activated. In three-step control both output signals are active.
ST2BI_ON	BOOL	FALSE	At the input parameter "Enable two-step control for bipolar manipulated value range" you can select from the modes "Two-step control for bipolar manipulated value range" and "Two-step control for unipolar manipulated value range". STEP3_ON = FALSE is required.
MAN_ON	BOOL	FALSE	Setting the input parameter "Enable manual mode" allows the output signals to be set manually.
POS_P_ON	BOOL	FALSE	For manual mode three-step control, the output signal QPOS_P can be operated on the input parameter "Positive pulse on". In manual mode with two-step control, QNEG_P is always set inversely to QPOS_P.
NEG_P_ON	BOOL	FALSE	For manual mode three-step control, the output signal QNEG_P can be operated on the input parameter "Negative pulse on". In manual mode with two-step control, QNEG_P is always set inversely to QPOS_P.
SYN_ON	BOOL	TRUE	By setting the input parameter "Enable synchronization", it is possible to synchronize the pulse output automatically with the block that updates the input variable INV. This ensures that a change in the input variable is output as quickly as possible as a pulse.

Parameters	Data type	Default	Description
COM_RST	BOOL	FALSE	The block has an initialization routine that is processed when the input "Restart" is set.
CYCLE	TIME	T#10ms	The time between block calls must be constant. The "Sampling time" input specifies the time between block calls. CYCLE >= 1ms

See also

Differences compared to PULSEGEN S7-300/400 (Page 5032)

Output parameter PULSEGEN

Table 9-189

Parameters	Data type	Default	Description
QPOS_P	BOOL	FALSE	The output parameter "Output signal positive pulse" is set if a pulse will be output. In three-step control, this is always the positive pulse. In two-step control, the QNEG_P is always set inversely to QPOS_P.
QNEG_P	BOOL	FALSE	The output parameter "Output signal negative pulse" is set if a pulse will be output. In three-step control, this is always the negative pulse. In two-step control, QNEG_P is always set inversely to QPOS_P.

See also

Differences compared to PULSEGEN S7-300/400 (Page 5032)

TCONT_CP

Description TCONT_CP

The instruction TCONT_CP is used to control temperature processes with continuous or pulsed control signals. The controller functionality is based on the PID control algorithm with additional functions for temperature processes. To improve the control response with temperature processes, the block includes a control zone and reduction of the proportional component if there is a setpoint step change.

The instruction can set the PI/PID parameters itself using the controller optimization function.

Application

The controller controls one actuator; in other words, with one controller you can either heat or cool but not both. If you use the block for cooling, GAIN must be assigned a negative value.

This inversion of the controller means that if the temperature rises, for example, the manipulated variable LMN and with it the cooling action is increased.

Call

The instruction TCONT_CP must be called equidistant. To achieve this, use a cyclic interrupt priority class (for example, OB35 for an S7-300).

The TCONT_CP instruction has an initialization routine that is run through when input parameter COM_RST = TRUE is set. During initialization, the integral action is set to the initialization value I_ITVAL. All the signal outputs are set to zero. Following execution of the initialization routine, the block sets COM_RST back to FALSE. If you require initialization when the CPU restarts, call the block in OB100 with COM_RST = TRUE.

If you call the instruction TCONT_CP as a multiple instance DB, no technology object is created. No parameter assignment interface or commissioning interface is available. You must assign parameters for TCONT_CP directly in the multiple instance DB and commission it via a watch table.

See also

Operating principle of the pulse generator (Page 6822)

Block diagram TCONT_CP (Page 6824)

Differences compared to TCONT_CP S7-300/400 (Page 5032)

Mode of operation TCONT_CP

Setpoint branch

The setpoint is entered at input SP_INT in floating-point format as a physical value or percentage. The setpoint and process value used to form the control deviation must have the same unit.

Process value options (PVPER_ON)

Depending on PVPER_ON, the process value can be read in, in the I/O or floating-point format.

PVPER_ON	Process Value Input
TRUE	The process value is read in via the analog I/Os (PIW xxx) at input PV_PER.
FALSE	The process value is acquired in floating-point format at input PV_IN.

Process value format conversion CRP_IN (PER_MODE)

The CRP_IN function converts the I/O value PV_PER to floating-point format depending on the PER_MODE switch according to the following rules:

PER_MODE	Output of CRP_IN	Analog Input Type	Unit
0	PV_PER * 0.1	Thermoelements; PT100/NI100; standard	°C;°F
1	PV_PER * 0.01	PT100/NI100; climate;	°C;°F
2	PV_PER * 100/27648	Voltage/current	%

Process value scaling PV_NORM (PF_FAC, PV_OFFS)

The PV_NORM function calculates the output of CRP_IN according to the following rule:

"Output of PV_NORM" = "Output of CRP_IN)" * PV_FAC + PV_OFFS

It can be used for the following purposes:

- Process value adjustment with PV_FAC as process value factor and PV_OFFS as process value offset.
- Scaling of temperature to percentage
  You want to enter the setpoint as a percentage and must now convert the measured temperature value to a percentage.
- Scaling of percentage to temperature
  You want to enter the setpoint in the physical temperature unit and must now convert the measured voltage/current value to a temperature.

Calculation of the parameters:

- PV_FAC = range of PV_NORM/range of CRP_IN;
- PV_OFFS = LL (PV_NORM) - PV_FAC * LL(CRP_IN);
  where LL: Low limit

The scaling is switched off with the default values (PV_FAC = 1.0 and PV_OFFS = 0.0). The effective process value is output at the PV output.

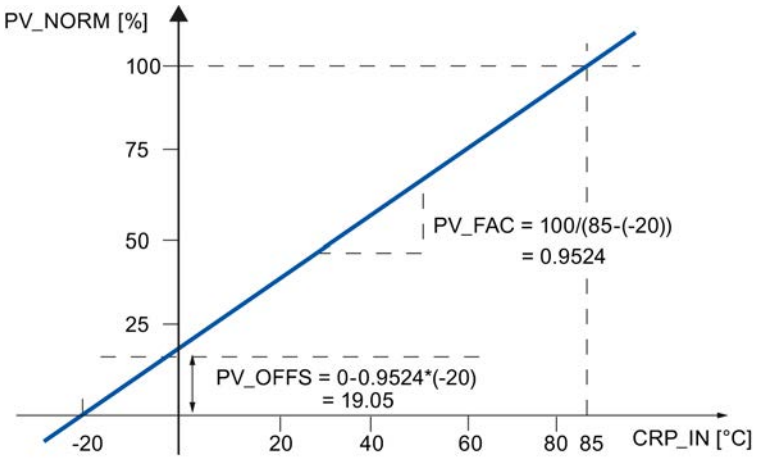
Note

With pulse control, the process value must be transferred to the block in the fast pulse call (reason: mean value filtering). Otherwise, the control quality can deteriorate.

Example of Process Value Scaling

If you want to enter the setpoint as a percentage, and you have a temperature range of -20 to 85 °C applied to , CRP_IN you must normalize the temperature range as a percentage.

The diagram below shows an example of adapting the temperature range -20 to 85 °C to an internal scale of 0 to 100 %:



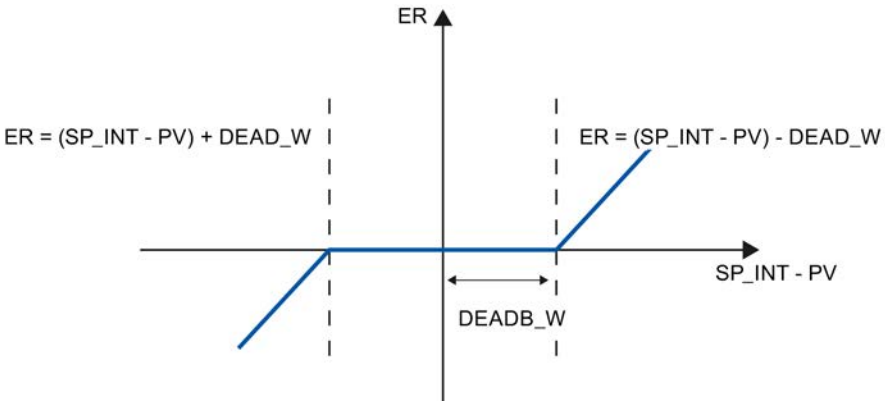
Forming the control deviation

The difference between the setpoint and process value is the control deviation before the dead band.

The setpoint and process value must exist in the same unit.

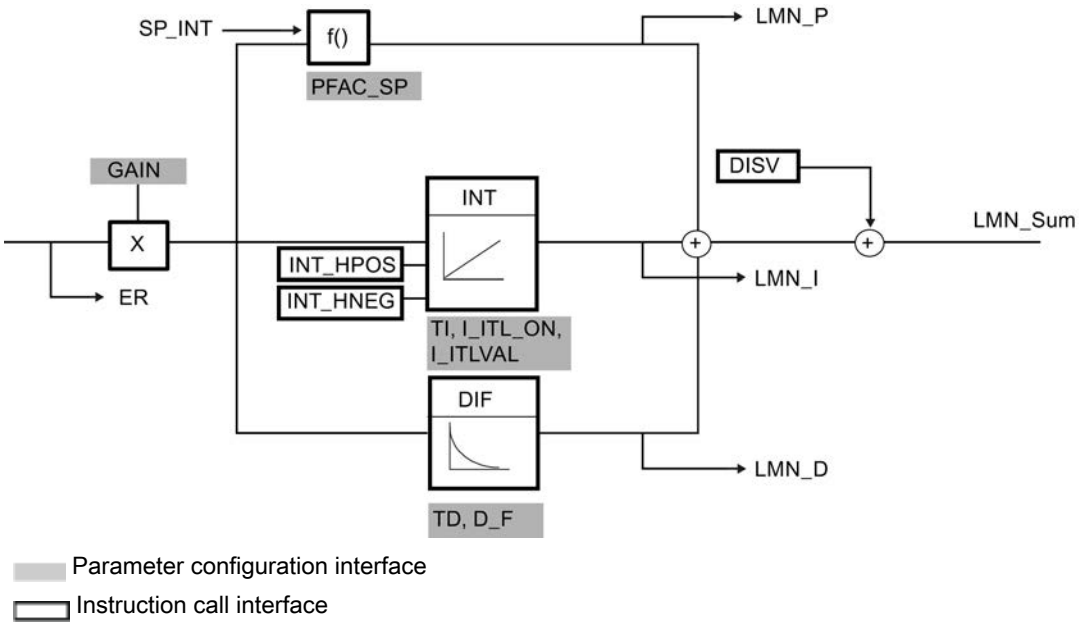
Dead band (DEADB_W)

To suppress a minor sustained oscillation due to the manipulated variable quantization (for example, in pulse width modulation with PULSEGEN) a dead band is applied to the (DEADBAND) control deviation. With DEADB_W = 0.0, the dead band is disabled. The effective control deviation is indicated by the ER parameter.



PID Algorithm

The following figure shows the block diagram of the PID algorithm.



PID Algorithm (GAIN, TI, TD, D_F)

The PID algorithm operates as a position algorithm. The proportional, integral (INT), and derivative (DIF) actions are connected in parallel and can be activated or deactivated individually. This allows P, PI, PD, and PID controllers to be configured.

Controller tuning supports PI and PID controllers. Controller inversion is implemented using a negative GAIN (cooling controller).

If you set TI and TD to 0.0, you obtain a pure P controller at the operating point.

The step response in the time range is:

$$LMN_Sum(t) = GAIN*ER(0)(1+\frac{1}{TI} * t+D_F * e^{\frac{-t}{TD/D_F}}$$

Where:

LMN_Sum(t) the manipulated variable in the controller's automatic mode

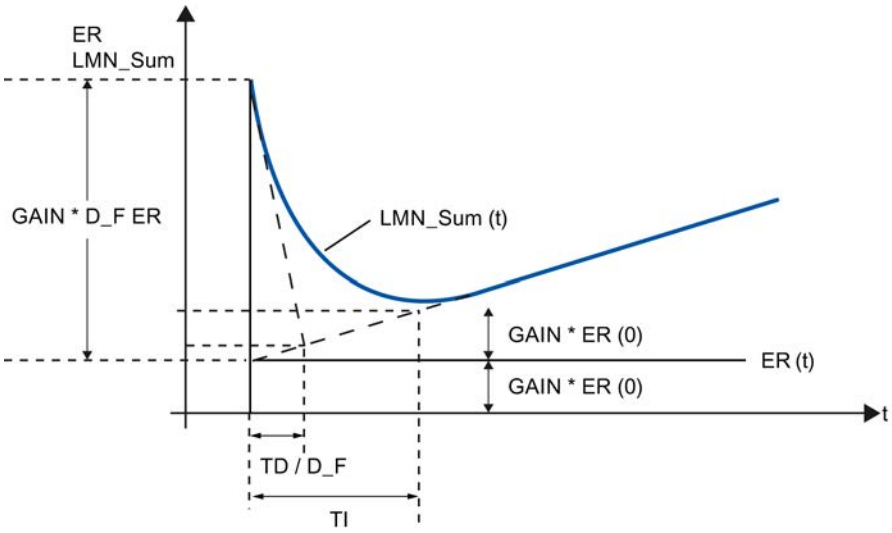
ER (0) is the step height of the normalized control deviation

GAIN is the controller gain

TI is the integration time

TD is the derivative action time

D_F is the derivative factor



Integral action (TI, I_ITL_ON, I_ITLVAL)

In manual mode, it is corrected as follows: LMN_I = LMN - LMN_P - DISV.

If the output value is limited, the integral action is halted. If the control deviation moves the integral action back in the direction of the output range, the integral action is enabled again.

The integral action is also modified by the following measures:

- The integral action of the controller is deactivated by TI = 0.0
- Weakening of the proportional action when setpoint changes occur
- Control zone
- The output value limits can be modified online

Weakening of the proportional action when setpoint changes occur (PFAC_SP)

To prevent overshoot, you can weaken the proportional action using the parameter "Proportional factor for setpoint changes" (PFAC_SP). Using PFAC_SP, you can select continuously between 0.0 and 1.0 to decide the effect of the proportional action when the setpoint changes:

- PFAC_SP = 1.0: Proportional action has full effect if the setpoint changes
- PFAC_SP = 0.0: Proportional action has no effect if the setpoint changes

The weakening of the proportional action is achieved by compensating the integral action.

Derivative action (TD, D_F)

- The derivative action of the controller is deactivated by TD = 0.0
- If the derivative action is active, the following relationship should apply:
  TD = 0.5 * CYCLE * D_F

Parameter Settings of a P or PD Controller with Operating Point

In the user interface, deactivate the integral action (TI = 0.0) and possibly also the derivative action (TD = 0.0). Then make the following parameter settings:

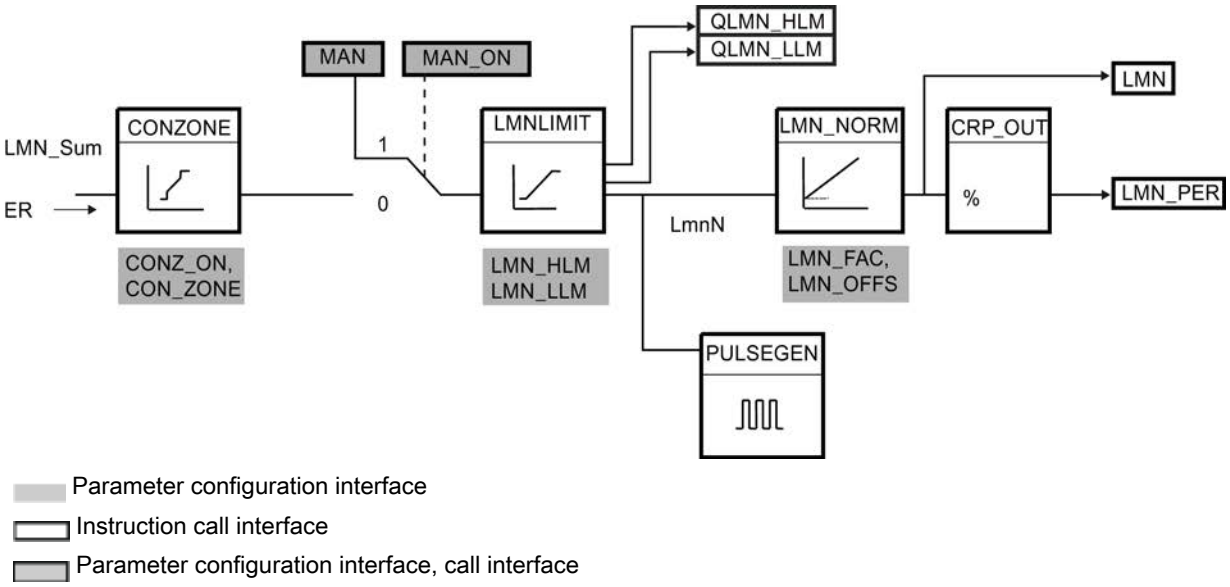
- I_ITL_ON = TRUE
- I_ITLVAL = operating point;

Feedforward control (DISV)

A disturbance variable can be added at the DISV input.

Calculating the output value

The diagram below is the block diagram of the output value calculation:



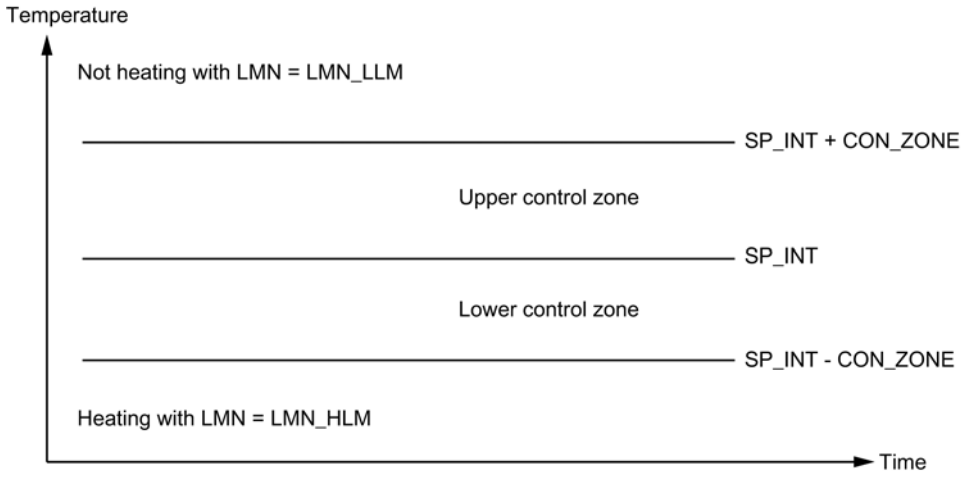
Control zone (CONZ_ON, CON_ZONE)

If CONZ_ON = TRUE, the controller operates with a control zone. This means that the controller operates according to the following algorithm:

- If process value PV exceeds the setpoint SP_INT by more than CON_ZONE, the value LMN_LLM is output as the manipulated variable.
- If the process value PV falls below setpoint SP_INT by more than CON_ZONE, LMN_HLM is output.
- If the process value PV is within the control zone (CON_ZONE), the output value takes its value from the PID algorithm LMN_Sum.

Note

Changing the manipulated variable from LMN_LLM or LMN_HLM to LMN_Sum occurs under compliance of a hysteresis of 20% of the control zone.



Note

Before enabling the control zone manually, make sure that the control zone band is not too narrow. If the control zone band is too small, oscillations will occur in the manipulated variable and process value.

Advantage of the Control Zone

When the process value enters the control zone, the D-action causes an extremely fast reduction of the manipulated variable. This means that the control zone is only useful when the D-action is activated. Without a control zone, only the reducing P-action would essentially reduce the manipulated variable. The control zone leads to faster settling without overshoot or undershoot if the output minimum or maximum manipulated variable is a long way from the manipulated variable required for the new operating point.

Manual value processing (MAN_ON, MAN)

You can change over between manual and automatic mode. In manual mode, the manipulated variable is corrected to a manually selected value.

The integral action (INT) is set internally to LMN - LMN_P - DISV and the derivative action (DIF) is set to 0 and synchronized internally. Changeover to automatic mode is therefore bumpless.

Note

The MAN_ON parameter has no effect during tuning.

Output value limit LMNLIMIT (LMN_HLM, LMN_LLM)

The LMNLIMIT function is used to limit the output value to the limits LMN_HLM and LMN_LLM. If these limits are reached, this is indicated by the message bits QLMN_HLM and QLMN_LLM.

If the output value is limited, the integral action is halted. If the control deviation moves the integral action back in the direction of the output range, the integral action is enabled again.

Changing the Manipulated Value Limits Online

If the range of the output value is reduced and the new unlimited value of the output value is outside the limits, the integral action and therefore the output value shifts.

The output value is reduced by the same amount as the output value limit changed. If the output value was unlimited prior to the change, it is set exactly to the new limit (described here for the high output value limit).

Scaling of output value LMN_NORM (LMN_FAC, LMN_OFFS)

The LMN_NORM function normalizes the output value according to the following rule:

LMN = LmnN * LMN_FAC + LMN_OFFS

It can be used for the following purposes:

- Output value scaling with LMN_FAC as output value factor and LMN_OFFS as output value offset.

The output value is also available in I/O format. The CRP_OUT function converts the LMN floating-point value to an I/O value according to the following rule:

LMN_PER = LMN * 27648/100

The scaling is switched off with the default values (LMN_FAC = 1.0 and LMN_OFFS = 0.0). The effective output value is sent to output LMN.

Save controller parameters SAVE_PAR

If you classify the current controller parameters as utilizable, you can save these before a manual change in structure parameters provided specifically for this in the instance DB of the

instruction TCONT_CP. If you optimize the controller, the saved parameters are overwritten by the values that were valid prior to tuning.

PFAC_SP, GAIN, TI, TD, D_F, CONZ_ON and CONZONE are written to the structure PAR_SAVE.

Reloading Saved Controller Parameters UNDO_PAR

The last controller parameter settings you saved can be activated for the controller again using this function (in manual mode only).

Change between PI and PID parameters LOAD_PID (PID_ON)

Following tuning, the PI and PID parameters are stored in the PI_CON and PID_CON structures. Depending on PID_ON, you can use LOAD_PID in manual mode to write the PI or PID parameters to the effective controller parameters.

PID parameters PID_ON = TRUE	PI parameters PID_ON = FALSE
• GAIN = PID_CON.GAIN • TI = PID_CON.TI • TD = PID_CON.TD	• GAIN = PI_CON.GAIN • TI = PI_CON.TI

Note

The controller parameters are only written back to the controller with UNDO_PAR or LOAD_PID, if the controller gain is not equal to 0:

With LOAD_PID, the parameters are only copied if the corresponding GAIN <> 0 is (either the PI or PID parameters). This strategy takes into account the situation that no tuning has yet been made or that PID parameters are missing. If PID_ON = TRUE and PID.GAIN = FALSE, PID_ON is set to FALSE and the PI parameter is copied.

- D_F, PFAC_SP are preset by the the tuning. These can then be modified by the user. LOAD_PID does not change these parameters.
- With LOAD_PID, the control zone is always recalculated (CON_ZONE = 250/GAIN), even if CONZ_ON = FALSE.

See also

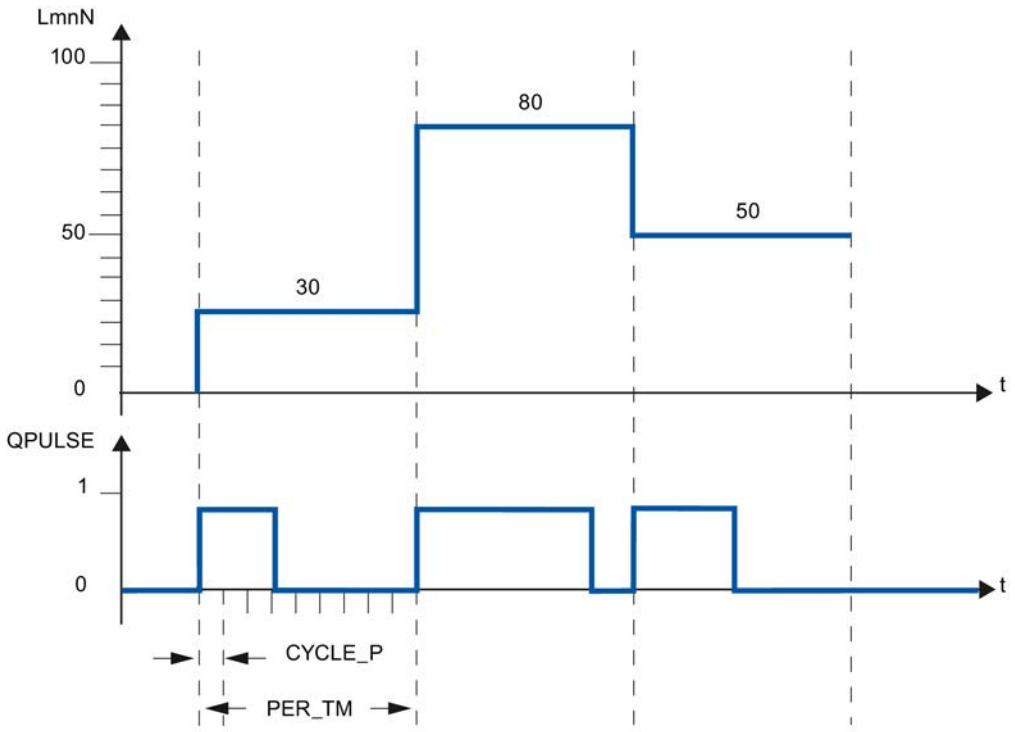
Operating principle of the pulse generator (Page 6822)

Block diagram TCONT_CP (Page 6824)

Differences compared to TCONT_CP S7-300/400 (Page 5032)

Operating principle of the pulse generator

The function PULSEGEN transforms the analog manipulated value LmnN through pulse width module into an impulse sequence with the period duration PER_TM. PULSEGEN is switched on with PULSE_ON = TRUE and is processed in the cycle CYCLE_P.



A manipulated value of LmnN = 30% and 10 PULSEGEN calls per PER_TM therefore means:

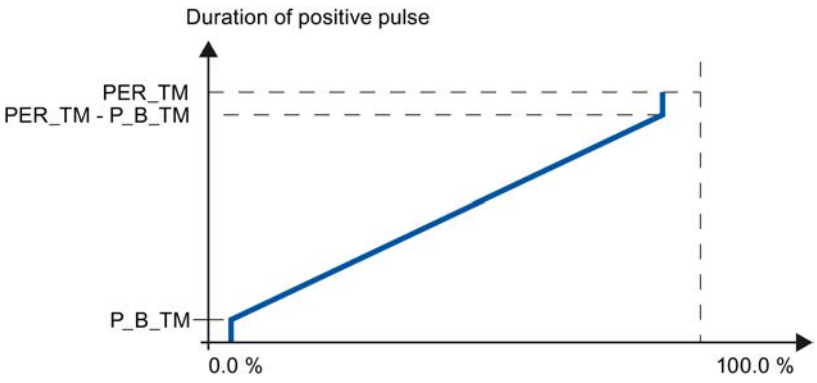
- TRUE at output QPULSE for the first three PULSEGEN calls
  (30% of 10 calls)
- FALSE at output QPULSE for seven further PULSEGEN calls
  (70% of 10 calls)

The duration of a pulse per pulse repetition period is proportional to the manipulated variable and is calculated as follows:

Pulse duration = PER_TM * LmnN /100

By suppressing the minimum pulse or break time, the characteristic curve of the conversion develops "knees" in the start and end regions.

The following diagram illustrates two-step control with a unipolar manipulated variable range (0% to 100%):



Minimum pulse or minimum break time (P_B_TM)

Short on or off times hinder the lifespan of actuators and fine controlling units. These can be avoided by setting a minimum pulse duration or minimum break time P_B_TM.

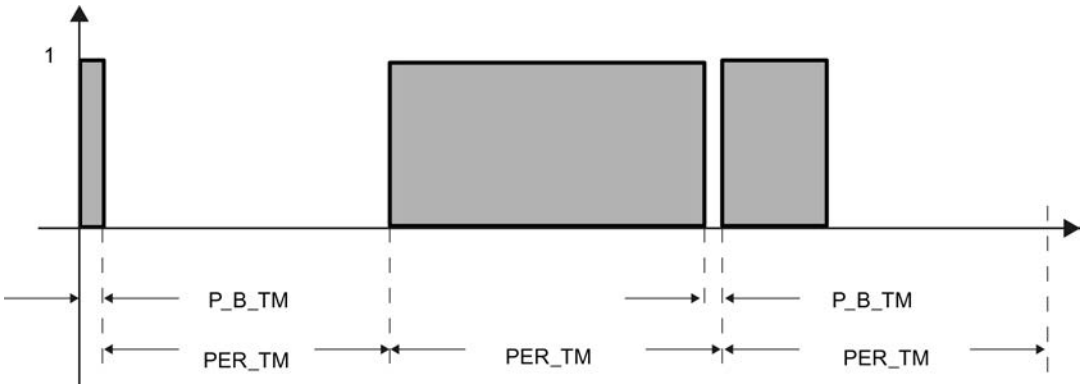
Small absolute values at the input variable LmnN that could otherwise generate a pulse duration shorter than P_B_TM are suppressed.

Large input values that would generate a pulse duration greater than PER_TM - P_B_TM are set to 100%. This reduces the dynamics of pulse generation.

Set values of P_B_TM ≤ 0,1 * PER_TM are recommended for the minimum pulse duration and the minimum break duration.

The "knees" in the curves in the diagram above are caused by the minimum pulse or minimum break times.

The following schematic illustrates the switching response of the pulse output:



Accuracy of pulse generation

The smaller the pulse generator CYCLE_P is compared to the period duration PER_TM, the more precise the pulse width modulation is. To achieve sufficiently accurate control, the following relationship should apply:

CYCLE_P ≤ PER_TM/50

The manipulated value is transformed with a resolution of ≤ 2 % into an impulse.

Note

When calling the controller in the pulse shaper cycle, you must note the following:

Calling the controller in the pulse shaper cycle will cause the process value to be averaged. As a result, at output PV, different values may be at input PV_IN and PV_PER. If you want to track the setpoint value, you must save the process value at input parameter PV_IN at the call times for complete controller processing (QC_ACT = TRUE). For pulse shaper calls occurring between these times, you must supply the input parameters PV_IN and SP_INT with the saved process value.

See also

Description TCONT_CP (Page 6812)

Mode of operation TCONT_CP (Page 6813)

Block diagram TCONT_CP (Page 6824)

Input parameters TCONT_CP (Page 6825)

Output parameters TCONT_CP (Page 6826)

In/out parameters TCONT_CP (Page 6826)

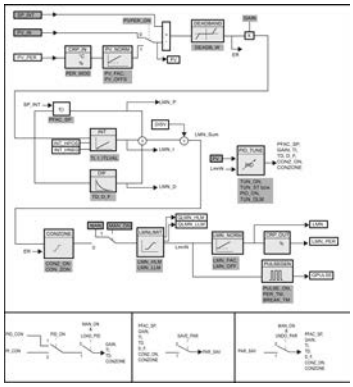
Static variables TCONT_CP (Page 6827)

Parameter STATUS_H (Page 6831)

Parameters STATUS_D (Page 6832)

Differences compared to TCONT_CP S7-300/400 (Page 5032)

Block diagram TCONT_CP



See also

Description TCONT_CP (Page 6812)

Mode of operation TCONT_CP (Page 6813)

Operating principle of the pulse generator (Page 6822)
Input parameters TCONT_CP (Page 6825)
Output parameters TCONT_CP (Page 6826)
In/out parameters TCONT_CP (Page 6826)
Static variables TCONT_CP (Page 6827)
Parameter STATUS_H (Page 6831)
Parameters STATUS_D (Page 6832)
Differences compared to TCONT_CP S7-300/400 (Page 5032)

Input parameters TCONT_CP

Table 9-190

Parameters	Address	Data type	Default	Description
PV_IN	0.0	REAL	0.0	At the "Process value input" you can assign parameters to a commissioning value or you can interconnect an external process value in floating-point format. The valid values depend on the sensors used.
PV_PER	4.0	INT	0	The process value in I/O format is interconnected with the controller at the "Process value I/O" input.
DISV	6.0	REAL	0.0	For feedforward control, the disturbance variable is interconnected to the "Disturbance variable" input.
INT_HPOS	10.0	BOOL	FALSE	The output of the integral action can be held in the positive direction. For this, the input INT_HPOS must be set to TRUE. In a cascade control, INT_HPOS of the primary controller is connected to QLMN_HLM of the secondary controller.
INT_HNEG	10.1	BOOL	FALSE	The output of the integral action can be held in the negative direction. For this, the input INT_HNEG must be set to TRUE. In a cascade control, INT_HNEG of the primary controller is connected to QLMN_LLM of the secondary controller.
SELECT	12.0	INT	0	If the pulse shaper is on, there are several ways of calling the PID algorithm and pulse shaper: • SELECT = 0: The controller is called in a fast cyclic interrupt priority class and the PID algorithm and pulse shaper are processed. • SELECT = 1: The controller is called in OB1 and only the PID algorithm is processed. • SELECT = 2: The controller is called in a fast cyclic interrupt priority class and only the pulse shaper is processed. • SELECT = 3: The controller is called a slow cyclic interrupt priority class and only the PID algorithm is processed.

See also

Operating principle of the pulse generator (Page 6822)
Block diagram TCONT_CP (Page 6824)
Differences compared to TCONT_CP S7-300/400 (Page 5032)

Output parameters TCONT_CP

Table 9-191

Parameter	Address	Data type	Default	Description
PV	14.0	REAL	0.0	The effective process value is output at the "Process value" output. The valid values depend on the sensors used.
LMN	18.0	REAL	0.0	The effective "Manipulated value" is output in floating point format at the "Manipulated value" output.
LMN_PER	22.0	INT	0	The manipulated value in I/O format is interconnected with the controller on the output "Manipulated value I/O".
QPULSE	24.0	BOOL	FALSE	The manipulated value is pulse-width-modulated at the QPULSE output.
QLMN_HLM	24.1	BOOL	FALSE	The manipulated value is always restricted to a high limit and low limit. The output QLMN_HLM signals that the high limit has been reached.
QLMN_LLM	24.2	BOOL	FALSE	The manipulated value is always restricted to a high limit and low limit. The output QLMN_LLM signals that the low limit has been reached.
QC_ACT	24.3	BOOL	TRUE	This parameter indicates whether continuous control component will be processed the next time the block is called (relevant only when SELECT has the value 0 or 1).

See also

Operating principle of the pulse generator (Page 6822)

Block diagram TCONT_CP (Page 6824)

Parameter STATUS_H (Page 6831)

Parameters STATUS_D (Page 6832)

Differences compared to TCONT_CP S7-300/400 (Page 5032)

In/out parameters TCONT_CP

Table 9-192

Parameters	Address	Data type	Default	Description
CYCLE	26.0	REAL	0.1 s	Sets the sampling time for the PID algorithm. In phase 1, the tuner calculates the sampling time and enters it in CYCLE. CYCLE > 0.001 s
CYCLE_P	30.0	REAL	0.02 s	At this input, you set the sampling time for the pulse shaper action. In phase 1, the TCONT_CP instruction calculates the sampling time and enters it in CYCLE_P. CYCLE_P > 0.001 s
SP_INT	34.0	REAL	0.0	The input "Internal setpoint" is used to specify a setpoint. The valid values depend on the sensors used.
MAN	38.0	REAL	0.0	The "Manual value" input is used to set a manual value. In automatic mode, it tracks the manipulated value.

Parameters	Address	Data type	Default	Description
COM_RST	42.0	BOOL	FALSE	The block has an initialization routine that is processed when the COM_RST input is set.
MAN_ON	42.1	BOOL	TRUE	If the input "Enable manual mode" is set then the control loop is interrupted. The manual value MAN is set as manipulated value.

See also

Operating principle of the pulse generator (Page 6822)
Block diagram TCONT_CP (Page 6824)
Differences compared to TCONT_CP S7-300/400 (Page 5032)

Static variables TCONT_CP

Table 9-193

Parameters	Address	Data type	Default	Description
DEADB_W	44.0	REAL	0.0	A deadband is applied to the control deviation. The "Deadband width" input determines the size of the deadband. The valid values depend on the sensors used.
I_ITLVAL	48.0	REAL	0.0	The output of the integrator can be set at the I_ITL_ON input. The initialization value is applied to the "Initialization value of the I-action" input. During a restart COM_RST = TRUE, the I-action is set to the initialization value. Values from -100 to 100 % are permitted.
LMN_HLM	52.0	REAL	100.0	The output value is always restricted to a high limit and low limit. The "Manipulated value high limit" input specifies the high limit. LMN_HLM > LMN_LLM
LMN_LLM	56.0	REAL	0.0	The output value is always restricted to a high limit and low limit. The "Manipulated value low limit" input specifies the low limit. LMN_LLM < LMN_HLM
PV_FAC	60.0	REAL	1.0	The "Process value factor" input is multiplied by the "Process value I/O". The input is used to scale the process value range.
PV_OFFS	64.0	REAL	0.0	The "Process value offset" input is added to the "Process value I/O". The input is used to scale the process value range.
LMN_FAC	68.0	REAL	1.0	The "Output value factor" input is multiplied with the output value. The input is used to scale the output value range.
LMN_OFFS	72.0	REAL	0.0	The "Output value offset" input is added to the output value. The input is used to scale the output value range.
PER_TM	76.0	REAL	1.0 s	The period duration of the pulse width modulation is entered at the PER_TM parameter. The relationship of the period duration to the sampling time of the pulse shaper determines the accuracy of the pulse width modulation. PER_TM ≥ CYCLE
P_B_TM	80.0	REAL	0.02 s	You can assign a minimum pulse or break time at the parameter "Minimum pulse/ break time". P_B_TM is internally limited to > CYCLE_P.

Parameters	Address	Data type	Defaul t	Description
TUN_DLMN	84.0	REAL	20.0	Process excitation for controller tuning results from a output value step change at TUN_DLMN. Values from -100 to 100 % are permitted.
PER_MOD E	88.0	INT	0	You can use this switch to enter the type of I/O module. The process value at input PV_PER is then scaled as follows at the PV output. • PER_MODE = 0: Thermoelements; PT100/NI100; standard PV_PER * 0.1 Unit: °C, °F • PER_MODE = 1: PT100/NI100; climate PV_PER * 0.01 Unit: °C, °F • PER_MODE = 2: Current/voltage PV_PER * 100/27648 Unit: %
PVPER_ON	90.0	BOOL	FALS E	If the process value is to be read in from the I/Os, the PV_PER input must be interconnected with the I/Os and the "Enable process value I/Os" input must be set.
I_ITL_ON	90.1	BOOL	FALS E	The output of the integrator can be set at the I_ITLVAL input. The "Set I-action" input must be set for this.
PULSE_ON	90.2	BOOL	FALS E	If PULSE_ON = TRUE is set, the pulse shaper is activated.
TUN_KEEP	90.3	BOOL	FALS E	The mode changes to automatic only when TUN_KEEP changes to FALSE.
ER	92.0	REAL	0.0	The effective control deviation is output at the "Control deviation" output. The valid values depend on the sensors used.
LMN_P	96.0	REAL	0.0	The "P-action" output contains the proportional action of the manipulated tag.
LMN_I	100.0	REAL	0.0	The "integral action" output contains the integral action of the manipulated tag.
LMN_D	104.0	REAL	0.0	The "D-action" output contains the derivative action of the manipulated tag.
PHASE	108.0	INT	0	The current phase of controller tuning is indicated at the PHASE output. • PHASE = 0: No tuning mode; automatic or manual mode • PHASE = 1: Ready to start tuning; check parameters, wait for excitation, measure the sampling times • PHASE = 2: Actual tuning: Searching for point of inflection with constant output value. Entering the sampling time in instance DB. • PHASE = 3: Calculating process parameters. Saving valid controller parameters prior to tuning. • PHASE = 4: Controller design • PHASE = 5: Following up the controller to the new manipulated tag • PHASE = 7: Validating the process type
STATUS_H	110.0	INT	0	STATUS_H indicates the diagnostic value via the search for the point of inflection during the heating process.
STATUS_D	112.0	INT	0	STATUS_D indicates the diagnostic value via the controller design during the heating process.
QTUN_RUN	114.0	BOOL	0	The tuning manipulated tag has been applied, tuning has started and is still in phase 2 (searching for point of inflection).
PI_CON	116.0	STRU CT		PI controller parameters

Parameters	Address	Data type	Defaul t	Description
GAIN	+0.0	REAL	0.0	PI controller gain %/phys. unit
TI	+4.0	REAL	0.0 s	PI integration time [s]
PID_CON	124.0	STRU CT		PID controller parameters
GAIN	+0.0	REAL	0.0	PID controller gain
TI	+4.0	REAL	0.0s	PID integration time [s]
TD	+8.0	REAL	0.0s	PID derivative action time [s]
PAR_SAVE	136.0	STRU CT		The PID parameters are saved in this structure.
PFAC_SP	+0.0	REAL	1.0	Proportional factor for setpoint changes Values from 0.0 to 1.0 are permitted.
GAIN	+4.0	REAL	0.0	Controller gain %/phys. unit
TI	+8.0	REAL	40.0 s	Integration time [s]
TD	+12.0	REAL	10.0 s	Derivative action time (s)
D_F	+16.0	REAL	5.0	Derivative factor Values from 5.0 to 10.0 are permitted.
CON_ZONE	+20.0	REAL	100.0	Control zone band If the control deviation is greater than the control zone band, the high output value limit is output as output value. If the control deviation is less than the negative control zone band, the low output value limit is output as the output value. CON_ZONE ≥ 0.0
CONZ_ON	+24.0	BOOL	FALS E	Enable control zone
PFAC_SP	162.0	REAL	1.0	PFAC_SP specifies the effective P-action when there is a setpoint change. This is set between 0 and 1. • 1: P-action has full effect if the setpoint changes. • 0: P-action has no effect if the setpoint changes. Values from 0.0 to 1.0 are permitted.
GAIN	166.0	REAL	2.0	The "Proportional gain" input specifies controller amplification. The direction of control can be reversed by giving GAIN a negative sign. %/phys. unit
TI	170.0	REAL	40.0 s	The "Integration time" (integral-action time) input defines the integrator's time response.
TD	174.0	REAL	10.0 s	The "Derivative-action time" (rate time) input decides the time response of the differentiator.
D_F	178.0	REAL	5.0	The derivative factor decides the lag of the D-action. D_F = derivative-action time/"Lag of the D-action" Values from 5.0 to 10.0 are permitted.
CON_ZONE	182.0	REAL	100.0	If the control deviation is greater than the control zone band, the high output value limit is output as output value. If the control deviation is less than the negative control zone band, the low output value limit is output as the output value. The valid values depend on the sensors used.

Parameters	Address	Data type	Defaul t	Description
CONZ_ON	186.0	BOOL	FALS E	You can use CONZ_ON =TRUE to enable the control zone.
TUN_ON	186.1	BOOL	FALS E	If TUN_ON=TRUE, the output value is averaged until the output value excitation TUN_DLMN is enabled either by a setpoint step-change or by TUN_ST=TRUE.
TUN_ST	186.2	BOOL	FALS E	If the setpoint is to remain constant during controller tuning at the operating point, a output value step-change by the amount of TUN_DLMN is activated by TUN_ST=1.
UNDO_PAR	186.3	BOOL	FALS E	Loads the controller parametersPFAC_SP, GAIN, TI, TD, D_FCONZ_ON and CON_ZONE from the data structure PAR_SAVE (only in manual mode).
SAVE_PAR	186.4	BOOL	FALS E	Saves the controller parameters PFAC_SP, GAIN, TI, TD, D_F, CONZ_ON and CON_ZONE in the data structure PAR_SAVE.
LOAD_PID	186.5	BOOL	FALS E	Loads the controller parametersGAIN, TI,TD depending on PID_ON from the data structure PI_CON or PID_CON (only in manual mode)
PID_ON	186.6	BOOL	TRUE	At the PID_ON input, you can specify whether or not the tuned controller will operate as a PI or PID controller. • PID controller: PID_ON = TRUE • PI controller: PID_ON = FALSE With certain process types it is nevertheless possible that only a PI controller will be designed despite PID_ON = TRUE.
GAIN_P	188.0	REAL	0.0	Identified process gain. In the case of process type I, GAIN_P tends to be estimated too low.
TU	192.0	REAL	0.0	Identified time lag of the process. TU ≥ 3*CYCLE
TA	196.0	REAL	0.0	Identified recovery time of the process. In the case of process type I, TA tends to be estimated too low.
KIG	200.0	REAL	0.0	Maximum process value rise at manipulated tag excitation from 0 to 100 % [1/s] GAIN_P = 0.01 * KIG * TA
N_PTN	204.0	REAL	0.0	The parameter specifies the order of the process. "Non-integer values" are also possible. Values from 1.01 to 10.0 are permitted.
TM_LAG_P	208.0	REAL	0.0	Time constants of a PTN model (practical values only for N_PTN >= 2).
T_P_INF	212.0	REAL	0.0	Time from process excitation until the point of inflection.
P_INF	216.0	REAL	0.0	Process value change from process excitation until the point of inflection. The valid values depend on the sensors used.
LMN0	220.0	REAL	0.0	Output value at the start of tuning Detected in phase 1 (mean value). Values from 0 to 100 % are permitted.
PV0	224.0	REAL	0.0	Process value at the start of tuning
PVDT0	228.0	REAL	0.0	Process value slew rate at start of tuning [1/s] Sign adapted.
PVDT	232.0	REAL	0.0	Current process value slew rate [1/s] Sign adapted.
PVDT_MAX	236.0	REAL	0.0	Max. change in the process value per second [1/s] Maximum derivative of the process value at the point of inflection (sign adapted, always > 0); is used to calculate TU and KIG.

Parameters	Address	Data type	Defaul t	Description
NOI_PVDT	240.0	REAL	0.0	Noise action in PVDT_MAX in % The higher the noise action, the less accurate (less aggressive) the control parameters.
NOISE_PV	244.0	REAL	0.0	Absolute noise in process value Difference between maximum and minimum process value in phase 1.
FIL_CYC	248.0	INT	1	Number of cycles of the mean value filter The process value is determined through FIL_CYC cycles. FIL_CYC is increased from 1 to a max. of 1024 if needed.
POI_CMAX	250.0	INT	2	Maximum number of cycles after point of inflection This time is used to find another (i.e. better) inflection point for measuring noise. The tuning is completed only after this time.
POI_CYCL	252.0	INT	0	Number of cycles after inflection point

See also

Operating principle of the pulse generator (Page 6822)

Block diagram TCONT_CP (Page 6824)

Differences compared to TCONT_CP S7-300/400 (Page 5032)

Parameter STATUS_H

STATUS_H	Description	Remedy
0	Default, or no/no new controller parameters	
10000	Tuning completed + suitable controller parameters found	
2xxxx	Tuning completed + controller parameters uncertain	
2xx2x	Point of inflection not reached (only if excited via setpoint step-change)	If the controller oscillates, weaken the controller parameters, or repeat the test with a smaller manipulated value difference TUN_DLMN.
2x1xx	Estimation error (TU < 3*CYCLE)	Reduce CYCLE and repeat attempt. Special case for PT1-only process: Do not repeat test, if necessary reduce controller parameters.
2x3xx	Estimation error TU too high	Repeat test under better conditions.
21xxx	Estimation error N_PTN < 1	Repeat test under better conditions.
22xxx	Estimation error N_PTN > 10	Repeat test under better conditions.
3xxxx	Tuning canceled in phase 1 owing to faulty parameter assignment:	

STATUS_H	Description	Remedy
30002	Effective manipulated value differential < 5%	Correct manipulated value differential TUN_DLMN.
30005	The sampling times CYCLE and CYCLE_P differ by more than 5% of the measured values.	Compare CYCLE and CYCLE_P with the cycle time of the cyclic interrupt priority class and note any loop scheduler. Check CPU load. An excessively loaded CPU can result in prolonged sampling times that are inconsistent with CYCLE or CYCLE_P.

Note

If you cancel tuning in phase 1 or 2, STATUS_H = 0 is set. However, STATUS_D still displays the status of the last controller calculation.

The higher the value of STATUS_D, the higher the order of the control process, the greater the TU/TA ratio and the gentler the controller parameters will be.

See also

Operating principle of the pulse generator (Page 6822)

Block diagram TCONT_CP (Page 6824)

Differences compared to TCONT_CP S7-300/400 (Page 5032)

Parameters STATUS_D

STATUS_D	Description
0	No controller parameters were calculated.
110	N_PTN <= 1.5 Process type I fast
121	N_PTN > 1.5 Process type I
200	N_PTN > 1.9 Process type II (transition range)
310	N_PTN >= 2.1 Process type III fast
320	N_PTN > 2.6 Process type III
111, 122, 201, 311, 321	Parameters have been corrected from phase 7.

Note

The higher the value of STATUS_D, the higher the order of the control process, the greater the TU/TA ratio and the gentler the controller parameters will be.

See also

Operating principle of the pulse generator (Page 6822)

Block diagram TCONT_CP (Page 6824)

Differences compared to TCONT_CP S7-300/400 (Page 5032)

TCONT_S

Description TCONT_S

The TCONT_S instruction is used on SIMATIC S7 automation systems to control technical temperature processes with binary manipulated value output signals for actuators with integrating behavior. The functionality is based on the PI control algorithm of the sampling controller. The step controller operates without a position feedback signal.

Application

You can also use the controller in a cascade control as a secondary position controller. You specify the actuator position via the setpoint input SP_INT. In this case, you must set the process value input and the parameter TI (integration time) to zero. An application might be, for example, temperature control with heating power control using pulse-break activation and cooling control using a butterfly valve. To close the valve completely, the manipulated variable (ER*GAIN) should be negative.

Call

The instruction TCONT_S must be called equidistant. To achieve this, use a cyclic interrupt priority class (for example, OB35 for an S7-300). The sampling time is specified at the CYCLE parameter.

If you call the instruction TCONT_S as a multiple instance DB, no technology object is created. No parameter assignment interface or commissioning interface is available. You must assign parameters for TCONT_S directly in the multiple instance DB and commission it via a watch table.

CYCLE sampling time

The CYCLE sampling time match the time difference between two calls (cycle time of the cyclic interrupt OB taking into account the reduction ratios).

The controller sampling time should not exceed 10% of the calculated integration time of the controller (TI). Generally, you must set the sampling time to a much lower value to achieve the required accuracy of the step controller.

Required accuracy G	MTR_TM	CYCLE = MTR_TM*G	Comment
0.5 %	10 s	0.05 s	The sampling time is determined by the required accuracy of the step controller.

Start-up

The TCONT_S instruction has an initialization routine that is run through when input parameter COM_RST = TRUE is set. Following execution of the initialization routine, the block sets COM_RST back to FALSE. All outputs are set to their initial values. If you require initialization when the CPU restarts, call the block in OB100 with COM_RST = TRUE.

See also

Block diagram TCONT_S (Page 6837)

Differences compared to TCONT_S S7-300/400 (Page 5033)

Mode of operation TCONT_S

Setpoint branch

The setpoint is entered at input SP_INT in floating-point format as a physical value or percentage. The setpoint and process value used to form the control deviation must have the same unit.

Process value options (PVPER_ON)

Depending on PVPER_ON, the process value can be read in, in the I/O or floating-point format.

PVPER_ON	Process Value Input
TRUE	The process value is read in via the analog I/Os (PIW xxx) at input PV_PER.
FALSE	The process value is acquired in floating-point format at input PV_IN.

Process value format conversion CRP_IN (PER_MODE)

The CRP_IN function converts the I/O value PV_PER to floating-point format depending on the PER_MODE switch according to the following rules:

PER_MODE	Output of CRP_IN	Analog Input Type	Unit
0	PV_PER * 0.1	Thermoelements; PT100/NI100; standard	°C;°F
1	PV_PER * 0.01	PT100/NI100; climate;	°C;°F
2	PV_PER * 100/27648	Voltage/current	%

Process value scaling PV_NORM (PF_FAC, PV_OFFS)

The PV_NORM function calculates the output of CRP_IN according to the following rule:

"Output of PV_NORM" = "Output of CRP_IN)" * PV_FAC + PV_OFFS

It can be used for the following purposes:

- Process value adjustment with PV_FAC as process value factor and PV_OFFS as process value offset.
- Normalization of temperature to percentage
  You want to enter the setpoint as a percentage and must now convert the measured temperature value to a percentage.
- Normalization of percentage to temperature
  You want to enter the setpoint in the physical temperature unit and must now convert the measured voltage/current value to a temperature.

Calculation of the parameters:

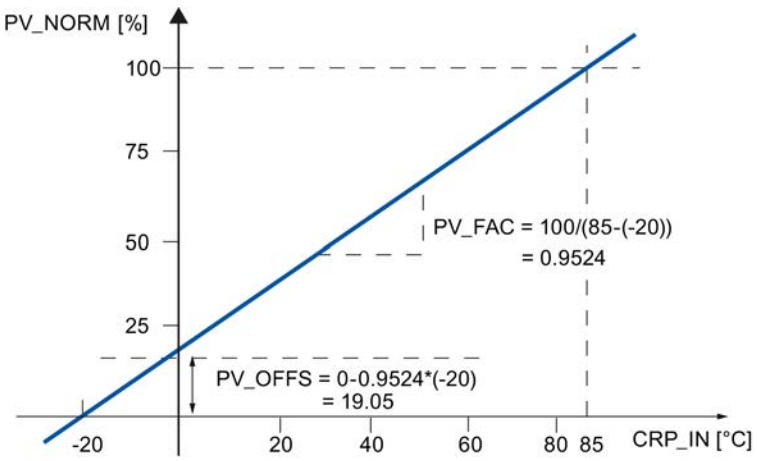
- PV_FAC = range of PV_NORM/range of CRP_IN;
- PV_OFFS = LL (PV_NORM) - PV_FAC * LL(CRP_IN);
  where LL: low limit

The normalization is switched off with the default values (PV_FAC = 1.0 and PV_OFFS = 0.0). The effective process value is output at the PV output.

Example of Process Value Normalization

If you want to enter the setpoint as a percentage, and you have a temperature range of -20 to 85 °C applied to , CRP_IN you must normalize the temperature range as a percentage.

The following diagram shows an example of adapting the temperature range -20 to 85 °C to an internal scale of 0 to 100 %:



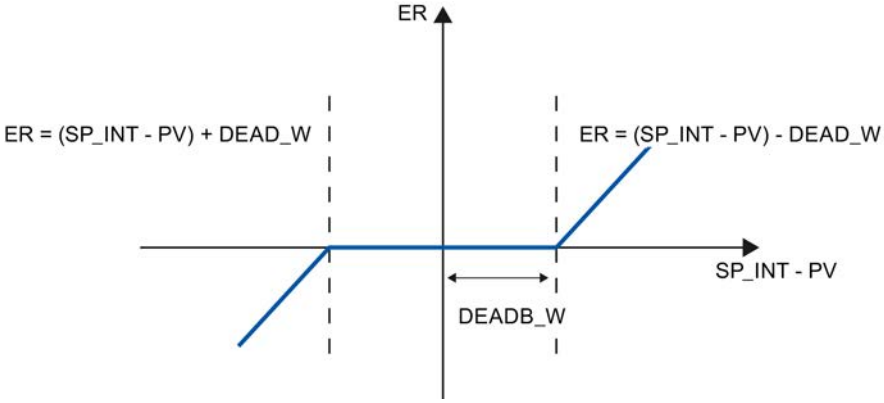
Forming the control deviation

The difference between the setpoint and process value is the control deviation before the dead band.

The setpoint and process value must exist in the same unit.

Dead band (DEADB_W)

To suppress a minor sustained oscillation due to the manipulated variable quantization (for example, in pulse width modulation with PULSEGEN) a dead band is applied to the (DEADBAND) control deviation. With DEADB_W = 0.0, the dead band is switched off.



PI step controller algorithm

The instruction TCONT_S operates without position feedback. The I-action of the PI algorithm and the assumed position feedback signal are calculated in an integrator (INT) and compared as a feedback value with the remaining P-action. The difference is applied to a three-step element (THREE_ST) and a pulse shaper (PULSEOUT) that generates the pulses for the control valve. Adapting the response threshold of the three-step element reduces the switching frequency of the controller.

Weakening of the P-action when setpoint changes occur

To prevent overshoot, you can weaken the P-action using the "Proportional factor for setpoint changes" parameter (PFAC_SP). Using PFAC_SP, you can now select continuously between 0.0 and 1.0 to decide the effect of the P-action when the setpoint changes:

- PFAC_SP = 1.0: P-action has full effect if the setpoint changes
- PFAC_SP = 0.0: P-action has no effect if the setpoint changes

As in the case of the continuous controller, a value of PFAC_SP < 1.0 can reduce the overshoot if the motor run time MTR_TM is small compared with the recovery time TA and the ratio is TU/TA < 0.2. If MTR_TM reaches 20% of TA, only a slight improvement can still be achieved.

Feedforward control

A disturbance variable can be added at the DISV input.

Manual value processing (LMNS_ON, LMNUP, LMNDN)

With LMNS_ON, you can change between manual and automatic mode. In manual mode, the actuator is halted and the integral action (INT) is set to 0 internally. Using LMNUP and LMNDN, the actuator can be adjusted to OPEN and CLOSED. Switching over to automatic mode therefore involves a bump. As a result of the GAIN, the existing control deviation leads to a

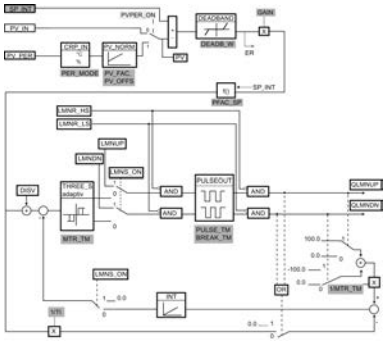
step change in the internal manipulated variable. The integral component of the actuator, however, results in a ramp-shaped excitation of the process.

See also

Block diagram TCONT_S (Page 6837)

Differences compared to TCONT_S S7-300/400 (Page 5033)

Block diagram TCONT_S



Parameter configuration interface

Instruction call interface

Parameter configuration interface, call interface

See also

Differences compared to TCONT_S S7-300/400 (Page 5033)

Description TCONT_S (Page 6833)

Mode of operation TCONT_S (Page 6834)

Input paramters TCONT_S (Page 6838)

Output parameters TCONT_S (Page 6839)

In/out parameters TCONT_S (Page 6839)

Static variables TCONT_S (Page 6839)

Input paramters TCONT_S

Table 9-194

Parameters	Address	Data type	Default	Description
CYCLE	0.0	REAL	0.1 s	At this input, you enter the sampling time for the controller. CYCLE ≥ 0.001
SP_INT	4.0	REAL	0.0	The input "Internal setpoint" is used to specify a setpoint. The valid values depend on the sensors used.
PV_IN	8.0	REAL	0.0	At the "Process variable input" you can assign parameters to a commissioning value or you can interconnect an external process value in floating-point format. The valid values depend on the sensors used.
PV_PER	12.0	INT	0	The process value in I/O format is interconnected with the controller at the "Process value I/O" input.
DISV	14.0	REAL	0.0	For feedforward control, the disturbance variable is interconnected to the "Disturbance variable" input.
LMNR_HS	18.0	BOOL	FALSE	The signal "Control valve at high endstop" is interconnected on the input "High endstop signal of position feedback". • LMNR_HS=TRUE: The control valve is at high endstop.
LMNR_LS	18.1	BOOL	FALSE	The signal "Control valve at low endstop" is interconnected on the input "Low endstop signal of position feedback". • LMNR_LS=TRUE: The control valve is at low endstop.
LMNS_ON	18.2	BOOL	TRUE	Manipulated value signal processing is switched to manual mode at the "Enable manual mode of manipulated signal".
LMNUP	18.3	BOOL	FALSE	In manual mode of manipulated signals, the output parameter QLMNUP is operated at the input parameter "Manipulated signal up".
LMNDN	18.4	BOOL	FALSE	In manual mode of the manipulated signals, the output parameter QLMNDN is operated at the input parameter "Manipulated signal down".

See also

Block diagram TCONT_S (Page 6837)

Differences compared to TCONT_S S7-300/400 (Page 5033)

Output parameters TCONT_S

Table 9-195

Parameters	Address	Data type	Default	Description
QLMNUP	20.0	BOOL	FALSE	If the output "Manipulated value signal up" is set then the control valve should be open.
QLMNDN	20.1	BOOL	FALSE	If the output "Manipulated value signal down" is set then the control valve should be closed.
PV	22.0	REAL	0.0	The effective process value is output at the "Process value" output.
ER	26.0	REAL	0.0	The effective system deviation is output at the "Error signal" output.

See also

Block diagram TCONT_S (Page 6837)

Differences compared to TCONT_S S7-300/400 (Page 5033)

In/out parameters TCONT_S

Table 9-196

Parameters	Address	Data type	Default	Description
COM_RST	30.0	BOOL	FALSE	The block has an initialization routine that is processed when the COM_RST input is set.

See also

Block diagram TCONT_S (Page 6837)

Differences compared to TCONT_S S7-300/400 (Page 5033)

Static variables TCONT_S

Table 9-197

Parameters	Address	Data type	Default	Description
PV_FAC	32.0	REAL	1.0	The "Process value factor" input is multiplied by the process value. The input is used to scale the process value range.
PV_OFFS	36.0	REAL	0.0	The input "Process value offset" is added to the process value. The input is used to scale the process value range. The valid values depend on the sensors used.
DEADB_W	40.0	REAL	0.0	A deadband is applied to the control deviation. The "Deadband width" input determines the size of the deadband. DEADB_W ≥ 0.0

Parameters	Address	Data type	Default	Description
PFAC_SP	44.4	REAL	1.0	PFAC_SP specifies the effective P-action when there is a setpoint change. • 1: P-action has full effect if the setpoint changes. • 0: P-action has no effect if the setpoint changes. Values from 0.0 to 1.0 are permitted.
GAIN	48.0	REAL	2.0	The "Proportional gain" input specifies controller amplification. The direction of control can be reversed by giving GAIN a negative sign. %/phys. unit
TI	52.0	REAL	40.0 s	The "Integration time" (integral-action time) input defines the integrator's time response.
MTR_TM	56.0	REAL	30 s	The runtime from endstop to endstop of the control valve is entered at the "Motor actuating time" parameter. MTR_TM ≥ CYCLE
PULSE_TM	60.0	REAL	0.0 s	A minimum pulse time can be configured at the "Minimum pulse time" parameter.
BREAK_TM	64.0	REAL	0.0 s	You can assign a minimum break time at the parameter "Minimum break time".
PER_MODE	68.0	INT	0	You can use this switch to enter the type of I/O module. The process value at input PV_PER is then scaled as follows at the PV output. • PER_MODE = 0: Thermoelements; PT100/NI100; standard PV_PER * 0.1 Unit: °C, °F • PER_MODE = 1: PT100/NI100; climate PV_PER * 0.01 Unit: °C, °F • PER_MODE = 2: Current/voltage PV_PER * 100/27648 Unit: %
PVPER_ON	70.0	BOOL	FALSE	If the process value is to be read in from the I/Os, the PV_PER input must be interconnected with the I/Os and the "Enable process value I/Os" input must be set.

See also

Block diagram TCONT_S (Page 6837)

Differences compared to TCONT_S S7-300/400 (Page 5033)

Integrated system functions

CONT_C_SF

CONT_C_SF

The instruction CONT_C_SF is integrated in the compact CPUs. The instruction must not be transmitted in the CPU during loading. The scope of function corresponds with the instruction CONT_C.

See also

Description CONT_C (Page 6793)
How CONT_C works (Page 6794)
CONT_C block diagram (Page 6796)
Input parameter CONT_C (Page 6796)
Output parameters CONT_C (Page 6798)

CONT_S_SF

CONT_S_SF

The instruction CONT_S_SF is integrated in the compact CPUs. The instruction must not be transmitted in the CPU during loading. The scope of function corresponds with the instruction CONT_S.

See also

Description CONT_S (Page 6799)
Mode of operation CONT_S (Page 6799)
Block diagram CONT_S (Page 6801)
Input parameters CONT_S (Page 6801)
Output parameters CONT_S (Page 6803)

PULSEGEN_SF

PULSEGEN_SF

The instruction PULSEGEN_SF is integrated in the compact CPUs. The instruction must not be transmitted in the CPU during loading. The scope of function corresponds with the instruction PULSEGEN.

See also

Description PULSEGEN (Page 6803)
Mode of operation PULSEGEN (Page 6804)
Mode of operation PULSEGEN (Page 6807)
Three-step control (Page 6807)
Two-step control (Page 6809)
Input parameters PULSEGEN (Page 6811)
Output parameter PULSEGEN (Page 6812)

PID Self Tuner

TUN_EC

Description TUN_EC

The TUN_EC instruction is used to tune a continuous PID controller. Depending on the controller, you interconnect TUN_EC with one of the following instructions:

- FM 355 C, FM 355 S pulse controller, FM 455 C or FM 455 S pulse controller
  - PID_FM
- PID Basic Functions
  - CONT_C with or without PULSEGEN
- Standard PID Control (PID Professional optional package)
  - PID_CP
- Modular PID Control (PID Professional optional package)
  - PID
  - LMNGEN_C

Call

TUN_EC is called in constant intervals of the cycle time of the calling OB (preferably in a cyclic interrupt OB).

When ADAPT1ST = TRUE , the initialization routine of the TUN_EC instruction is run. The following pre-assignments are made:

- The system switches to PI controller. The parameters of PI_CON are transferred to the controller for this purpose. If the PI_CON.GAIN and PI_CON.TI parameters both have the value 0.0, they are pre-assigned with PI_CON.GAIN = 1.5 and PI_CON.TI = 3600.0 s.
- PROCESS.GAIN = 999.0
- RATIOFAC = 1.0
- STATUS_H = 0; STATUS_D = 0; STATUS_C = 0

Note

Unless explicitly required, do **not** call TUN_ES during a warm restart of the CPU in its initialization routine as is usual for other instructions. If you do, all process parameters will be lost.

If you call the instruction TUN_EC as a multiple instance DB, no technology object is created. No commissioning interface is available. You must commission TUN_EC using a watch table.

CYCLE sampling time

You enter the sampling time in the CYCLE parameter. The sampling time should not exceed 10% of the determined integral action time of the controller. It must agree with the time difference between two calls (cycle time of the cyclic interrupt OB taking into account the reduction ratios).

In order for the process parameters to be identified reliably, the smallest time constant of the process (delay time) should be sampled at least 10 to 20 times.

In connection with the FM355/455 controller modules, the sampling time of TUN_EC must be approximately equal to the resulting cycle time of the controllers on the FM (Parameter setting > Module parameters).

Reaction to error

The error message word RET_VAL is not evaluated by the block.

See also

Mode of operation TUN_EC (Page 6845)

Input parameters TUN_EC (Page 6851)

Output parameters TUN_EC (Page 6852)

In/out parameters TUN_EC (Page 6854)

Parameter TUN_EC.PHASE (Page 6856)

Parameter TUN_EC.STATUS_H (Page 6858)

Parameter TUN_EC.STATUS_H (Page 6859)

Parameter TUN_EC.STATUS_D (Page 6860)

Scope of TUN_EC

The Self-Tuner can be used particularly well for:

- Temperature control systems, but also for
- Level control systems and
- Flow control systems

In the case of flow control systems a distinction must be made as to whether only the control valve itself is to be controlled or whether a process with associated delay follows the control valve. In the case of control valve control only, the Self-Tuner cannot be used.

Process requirements

The process must meet the following requirements:

- **Stable asymptotic transient response with associated delay**
  The process value must settle to steady state after a manipulated variable jump. This therefore excludes processes that already exhibit an oscillating response in the absence of closed-loop control, as well as controlled systems without inherent regulation (integrator in the controlled system).
- **Delay times not too large**
  The range of application can be specified based on the ratio of the delay time t_u and response time t_a. The delay time also encompasses any existing dead time.
  The pretuning or adaptation is designed for the range $t_u < 1/10\ t_a$, within which most controlled temperature systems fall.
  For the range of especially long delay times $1/10\ t_a < t_u < 1/3\ t_a$, the design of a utilizable PI controller is generally still also possible.
- **Adequate linear response across a sufficiently large operating range**
  This means that non-linear effects within this operating range can be ignored both during identification and during normal closed-loop control operation. However, it is possible to re-identify the process when there is a change from one operating point to another if the adaptation process is carried out again close to the new operating point and provided that the non-linearity is not passed through during adaptation.
  If certain static non-linearities (e.g., valve characteristic curves) are known, it is always advisable to compensate for them in advance with a polyline in order to linearize the process response.
- **Minimal disturbances in temperature processes**
  Disturbances such as thermal transfer to neighboring zones or heat gains or losses due to changes from one state of matter to another must not influence the overall temperature process to any great extent. If necessary, adaptation at the operating point may be necessary. For example, when zones of an extruder are tuned, all zones must be heated up simultaneously.
- **Quality of the measuring signals**
  The quality of the measured signals has to be sufficient; in other words, the signal-to-disturbance ratio has to be sufficiently high.
- **Process gains not too high**
  The process gains must not be too high. Normalization of the process values is not required. The process gain K can thus include physical units, such as:
  $K = \Delta PV/\Delta LMN$, [K]= °C/%
  The final controller design is based on a calculation of the process gain K and can thus in principle compensate for any values of K. However, K is initially unknown during the learning phase and overshoots cannot be avoided when there are extreme combinations of gain and learning step changes. The overshoot is also reduced by reducing the parameter LHLM_TUN in the heating process or by increasing LLLM_TUN in the cooling process.

Requirements for processes with a control valve with integral action

In processes with control valves with an integral action, further requirements must be met in addition to those specified above:

The motor run time of the control valve must be less than the time required to find a point of inflection after a manipulated variable jump (also refer to the process response figure). If this

is not the case, the process involved is often a flow rate control system in which only the control valve is effective as the dominating process action. It is not advisable to use the Self-Tuner in this case. You can then set the PI step controller manually in accordance with the following rule of thumb:

GAIN = 1, TI = Motor transition time

See also

Mode of operation TUN_EC (Page 6845)

Input parameters TUN_EC (Page 6851)

Output parameters TUN_EC (Page 6852)

In/out parameters TUN_EC (Page 6854)

Parameter TUN_EC.PHASE (Page 6856)

Parameter TUN_EC.STATUS_H (Page 6858)

Parameter TUN_EC.STATUS_H (Page 6859)

Parameter TUN_EC.STATUS_D (Page 6860)

Mode of operation TUN_EC

TUN_EC has the following functions for the control and optimization of the controller:

- **Manual mode**

  The Self-Tuner forces the controller into manual mode and transfers its manual parameters to it.

- **Manual/automatic changeover with predictive manipulated-value specification**

  After the controller has been optimized successfully, the Self-Tuner can change over the controller from manual mode to automatic mode by feedforwarding a predictive manipulated value in such a manner that the process settles rapidly into the setpoint. This type of changeover involves a bump. You can also set bumpless changeover at the Self-Tuner.

- Controller tuning

  The process is first identified during the optimization of the controller parameters. A controller is designed based on the process characteristics. The new system parameters are calculated on this basis and these are activated for the controller.

  – First adaption

    Pretuning can only be carried out for a positive setpoint jump. It is usually used during initial commissioning or in case of a serious change in the controlled system characteristics.

  – Fine tuning

    The controller can be fine tuned online during a positive setpoint jump when there is a change from one operating point to another or when there are changes in the process response.

  – Cooling tuning
    For control systems that operate with two opposing actuators (for example, in case of temperature control: heating and cooling actuators) the Self-Tuner determines the ratio of the process gains under the influence of a second actuator in the case of a negative manipulated value step (using the same example of a temperature control: under influence of the cooling actuator).

  Tuning the response to setpoint changes
  The controller is designed for optimum disturbance response, in principle. The resultant "Stringent" parameters would result in overshoots of 10% to 20 % of the step height at setpoint steps.
  The control zone represents an initial remedy. To tune the start-up control response for large setpoint steps, the Self-Tuner calculates a control zone band during optimization. The controller is active within the band. Outside the band the Self-Tuner controls the controller with the maximum or minimum manipulated variable of the controller.
  The structure segmentation represents an additional remedy. In this case, the proportional or derivative action is located in the feedback loop. A change in the setpoint thus only acts on the integral action of the controller.
  If structure segmentation is not possible for the controller or if the control zone fails, the following function can be activated to tune the command characteristics:

- Structure switchover:
  Temporary disabling of the I-action or predictive manipulated value specification for large positive setpoint steps. This function is not effective when the control zone is activated.

- Saving the controller parameters

  If you assess the current controller parameters to be utilizable, you can save these in specially provided parameters in the instance DB of the Self-Tuner, before a manual change in parameters. If you tune the controller, the saved parameters are overwritten by the values that were valid prior to tuning.

- **Reloading saved controller parameters**
  The last controller parameter settings you saved can be activated for the controller again using this function.
- **Changing the controller parameters**

  If you want to set the controller parameters yourself for the controller, you can assign these through special parameters in the instance DB of the Self-Tuner and transfer them to the controller by means of this function.

Enabling the functions

Function	Start bit	Enable bit	Start condition: PHASE	Start condition: Setpoint step-change at SP
Manual mode	—	MAN_ON	0 to 7	—
Manual/automatic changeover with predictive manipulated-value specification	—	PRED_ON	Transition 7 -> 4	—
First adaption	ADAPT1ST	—	0, 4	Pos.: > MIN_STEP
First adaption or fine tuning	ADAPT_ON	—	4	Pos.: > MIN_STEP
Cooling tuning	COOLID_ON	—	4	—
Control zone	—	CONZ_ON	4	
Structure switchover	—	STRUC_ON	4	Pos.: > MIN_STEP, Neg: > MIN_STEP × RATIO_FAC
Save controller parameters	SAVE_PAR	—	2 to 7	—
Reloading controller parameters	UNDO_PAR	—	4	—
Change controller parameters	LOAD_PAR	—	4	—

Note

Start bits are reset after the function has been started successfully. Enable bits remain set.

See also

Description TUN_EC (Page 6842)

Input parameters TUN_EC (Page 6851)

Output parameters TUN_EC (Page 6852)

In/out parameters TUN_EC (Page 6854)

Parameter TUN_EC.PHASE (Page 6856)

Parameter TUN_EC.STATUS_H (Page 6858)

Parameter TUN_EC.STATUS_H (Page 6859)

Parameter TUN_EC.STATUS_D (Page 6860)

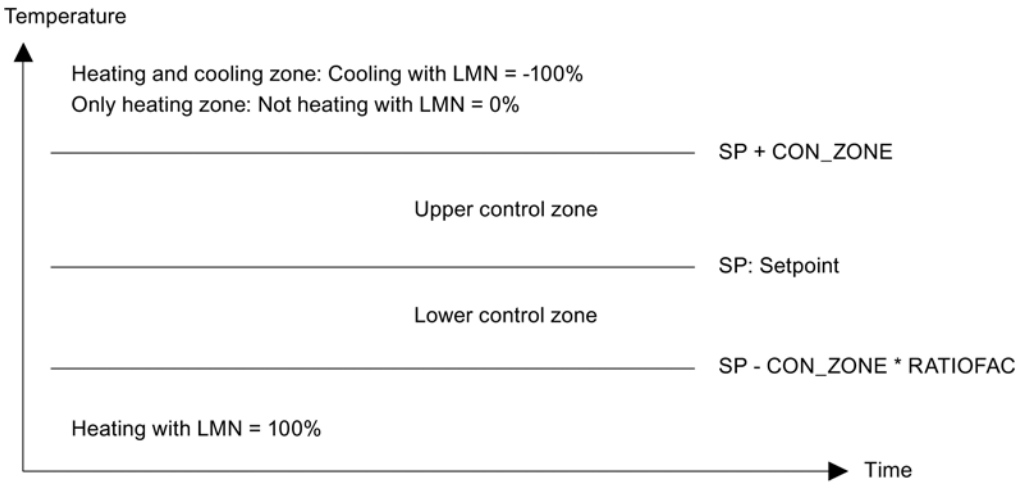
Control zone and structure switchover

Control zone

The Self-Tuner operates with a control zone. This means that the controller operates based on the following algorithm:

- If the control deviation is greater than CON_ZONE, the system is heated with maximum power.
- If the control deviation is less than (-CON_ZONE) × RATIOFAC, the system is cooled with maximum power.

During pretuning or fine tuning, a CON_ZONE control zone is determined by the Self-Tuner and activated if the process type is suitable: CONZ_ON = TRUE. In closed-loop control, you can change the control zone or deactivate it completely (with CONZ_ON = FALSE).



When the control zone is activated (CONZ_ON = TRUE), the structure switchover STRUC_ON is not effective.

Note

When there is a separate controller design for controlled systems with long delay time, a control zone is not determined, for example when STATUS_D = 3 after adaptation.

Before you switch on the control zone manually, make sure that the control zone width is not too small. If the control zone width is set too narrow, oscillations occur in the manipulated variable and the process value trends.

Structure segmentation

Structure segmentation is a function of the controller. In the case of closed-loop control with structure segmentation, the proportional or derivative action lies within the feedback loop, so that a change in the setpoint only acts on the integral action of the controller. When setpoint jumps occur, the manipulated variable does not undergo a jump change but instead follows a ramp-shaped course. The disturbance response is not changed as a result.

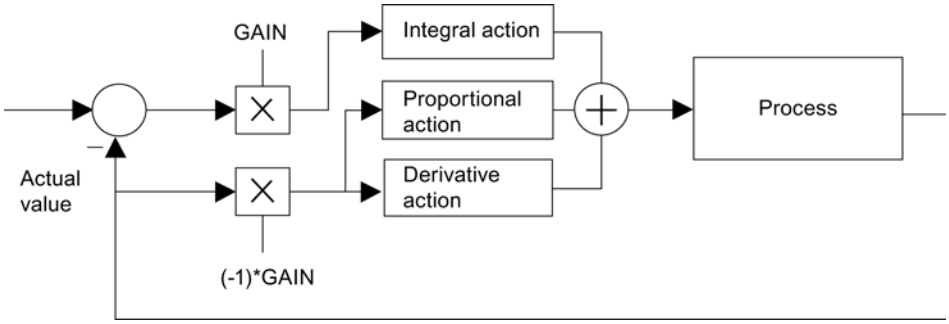


Figure 9-7 Closed-loop control with structure segmentation

In the case of closed-loop control without structure segmentation, the proportional and derivative actions lie within the control deviation, just like the integral action. You can select a structure switchover for this operating mode (STRUC_ON = TRUE).

Closed-loop control with structure segmentation is possible in the following control packages:

- FM 355/455
- Standard PID Control
- Modular PID Control

Note

Only closed-loop control without structure segmentation is possible for the simple controller PID Control of STEP 7 Basic and in the CPU 314 IFM. The structure switchover function should only be selected if there are no other alternatives.

Structure switchover when proportional and derivative actions are within the control deviation

The structure switchover is deactivated in the default setting. It should only be selected if no other alternatives such as, for example, control zone or structure segmentation, are available.

When the control zone is activated (CONZ_ON = TRUE), the structure switchover STRUC_ON is not effective.

You can activate the structure switchover with STRUC_ON = TRUE. The block automatically selects between two control measures depending on the total loop gain (product of the process gain and the controller gain):

- Integral action temporarily deactivated, P(D) control:
  In the case of a positive setpoint jump ≥ MIN_STEP, the integral action of the controller is deactivated temporarily and the gain is increased slightly, which means that a pure P(D) controller is used. In the proximity of the setpoint, the integral action is reactivated and the gain is reduced again (PHASE = 5).
- Temporary predictive manipulated variable specification:
  Processes with a high delay time cannot be controlled well with a P(D) control system. After a positive setpoint jump ≥ MIN_STEP, the manipulated variable required constantly for the new setpoint is therefore output constantly. In the proximity of the setpoint, the system is switched back bumplessly to PI or PID closed-loop control. This results in a slow, but overshoot-free response (PHASE = 6).

Note

If you do not achieve good results (transient response is slow) with the structure switchover in the case of positive setpoint jumps (for example, during heating processes), you can deactivate the structure switchover with STRUC_ON = FALSE, on the condition that small overshoots may occur.

Controller structure modes	CONZ_ON	PFDB_SEL, DFDB_SEL [1]	STRUC_ON
Activate control zone	TRUE	Any	Any
Activate structure segmentation	Any	TRUE	FALSE
Activate structure switchover	FALSE	FALSE	TRUE

[1] Controller parameters PFDB_SEL, DFDB_SEL: When CONT_C, these parameters are not available. This corresponds to the PFDB_SEL = DFDB_SEL = FALSE setting (without structure segmentation).

Note

Notes on selecting the operating modes:

- The Self-Tuner calculates a control zone and activates it automatically, if the ratio of the two time constants of the process is f = TM_LAG1 / TM_LAG2 > 10.
- In the case of jumps within the control zone, the structure segmentation that is selected by default acts, if allowed by the controller.
- If you want to compensate for small setpoint jumps within the control zone as quickly as possible and are willing to accept small overshoots, deactivate the structure segmentation. Numerical example: Control zone 20 °C, then overshoots of approx. 10% to 20% of the step size (< 20 °C) occur, i.e. less than 2 to 4 °C.
- The structure segmentation function prevents overshoots at setpoint jumps of all sizes, even if the process does not permit a control zone or you have deactivated it. However, the rise times are slightly slower than with a control zone.
- If the process does not allow a control zone and the controller does not allow a structure segmentation, activate the structure switchover. The structure switchover must not be activated at the same time as the structure segmentation.

Input parameters TUN_EC

Table 9-198

Parameter	Address	Data type	Default	Description
SP	0.0	REAL	0.0	Enter the setpoint at the SP input. It is interconnected with the controller via SP_OUT. Note that the setpoint must lie within the setpoint limits of the controller.
PV	4.0	REAL	0.0	PV is interconnected with the process value of the controller.
LMN	8.0	REAL	0.0	LMN is interconnected with the manipulated variable of the controller. Values from -100 to 100% are permitted.
MAN	12.0	REAL	0.0	Enter the manual value at the MAN input. If MAN_ON = TRUE, the manual value MAN is transferred to the manipulated variable of the controller. If MAN_ON = TRUE and MAN = 0.0, the heating/cooling function is off. If MAN_ON = FALSE, the manipulated variable is specified by the controller. Values from -100 to 100% are permitted.
MIN_STEP	16.0	REAL	10.0	You can specify the minimum setpoint jump above which pretuning or adaptation can be carried out in the MIN_STEP input. If structure switchover (STRUC_ON = TRUE) is activated, the setpoint jump must be greater than MIN_STEP in order for the structure switchover activation to take place. Recommendation: MIN_STEP ≥ 10 % of the required setpoint-process value range
LHLM_TUN	20.0	REAL	80.0	At the LHLM_TUN input, specify the manipulated variable jump for the first setting. The manipulated variable jump is calculated automatically during subsequent fine tuning. Values from 0 to 100% are permitted.
LLLM_TUN	24.0	REAL	-20.0	You specify the manipulated variable jump for cooling tuning at the LLM_TUN input. Values from -100 to 0% are permitted.
NORM_FAC	28.0	REAL	1.0	A normalization factor is applied for controllers with a normalized controller gain (Standard PID Control). The normalization factor is calculated on the basis of the range limits of the process value of the controller and is interconnected with the NORM_FAC input.
MAN_ON	32.0	BOOL	TRUE	If MAN_ON = TRUE, the manipulated variable of the controller is specified via the MAN input. If MAN_ON = FALSE, the manipulated variable is specified by the controller.
PID_ON	32.1	BOOL	TRUE	At the PID_ON input, you can specify whether or not the tuned controller will operate as a PI or PID controller. PID controller: PID_ON = TRUE PI controller: PID_ON = FALSE With certain controlled system types, however, it is possible that only a PI controller will be designed, despite PID_ON = TRUE.
STRUC_ON	32.2	BOOL	FALSE	If STRUC_ON = TRUE, a structure switchover (from PI(D) to P(D) control) ensures that overshoots are largely avoided at setpoint jumps. The STRUC_ON input must not be set = TRUE if the proportional action or derivative action is fed into the feedback loop of the controller.
WRITE_DIS	32.3	BOOL	FALSE	In connection with the controller module, the Self-Tuner must know whether the driver block is ready to send data to the FM. This is the case if LOAD_OP = LOAD_PAR = FALSE. The result of the OR operation of LOAD_OP and LOAD_PAR is interconnected with WRITE_DIS.

Parameter	Address	Data type	Default	Description
PRED_ON	32.4	BOOL	TRUE	When changing over to automatic mode you can specify whether you want to change over bumplessly from the last manual value or with bumps with a manipulated variable calculated by the Self-Tuner. If PRED_ON = TRUE (default setting) the changeover is carried out with bumps. The changeover with bumps is only effective after tuning and ensures rapid settling into the desired setpoint, in particular for a PI control system.
CYCLE	34.0	TIME	100 ms	The sampling time must be configured in the CYCLE input. CYCLE must correspond to the sampling time of the controller to be set and the time difference between two calls (cycle time of the cyclic interrupt OB taking into account the reduction ratios). CYCLE > 1 ms

See also

Description TUN_EC (Page 6842)
Mode of operation TUN_EC (Page 6845)
Output parameters TUN_EC (Page 6852)
In/out parameters TUN_EC (Page 6854)
Parameter TUN_EC.PHASE (Page 6856)
Parameter TUN_EC.STATUS_H (Page 6858)
Parameter TUN_EC.STATUS_H (Page 6859)
Parameter TUN_EC.STATUS_D (Page 6860)

Output parameters TUN_EC

Table 9-199

Parameter	Address	Data type	Default	Description
MAN_OUT	38.0	REAL	0.0	MAN_OUT is connected to the manual input value of the controller. If MAN_ON = TRUE, the Self-Tuner outputs the value specified at the MAN input to the MAN_OUT output.
SP_OUT	42.0	REAL	0.0	SP_OUT is connected to the setpoint input of the controller. The Self-Tuner writes the setpoint specified at the SP input to the SP_OUT output. If manual mode MAN_ON = TRUE, the setpoint is tracked to the process value: SP_OUT = PV.
GAIN	46.0	REAL	1.5	GAIN is interconnected with the proportional gain (controller gain) of the controller.
TI	50.0	TIME	1 h	TI is interconnected with the integral action time (reset time) of the controller.
TD	54.0	TIME	0 ms	TD is interconnected with the derivative action time (rate time) of the controller.
TM_LAG	58.0	TIME	1 s	TM_LAG is interconnected with the time lag of the controller differentiator.
RATIOFAC	62.0	REAL	1.0	RATIOFAC is interconnected with the heating/cooling ratio factor of the controller or of the PULSEGEN instruction. The parameter is only required for cooling.
PHASE	66.0	INT	0	Parameter TUN_EC.PHASE (Page 6856) displays the current sequence of the controller tuning. During the tuning of Standard PID Control, the PHASE output is interconnected with the PHASE input of the PID_CP instruction.

Parameter	Address	Data type	Default	Description
STATUS_H	68.0	INT	0	Parameter TUN_EC.STATUS_H (Page 6858) displays tuning results regarding the search for the point of inflection during the heating process.
STATUS_D	70.0	INT	0	Parameter TUN_EC.STATUS_D (Page 6860) displays tuning results regarding the controller design during the heating process.
STATUS_C	72.0	INT	0	Parameter TUN_EC.STATUS_H (Page 6859) displays tuning results regarding the search for the point of inflection during the cooling process.
QMAN_ON	74.0	BOOL	FALSE	QMAN_ON is interconnected with the manual/automatic changeover of the controller.
QI_SEL	74.1	BOOL	TRUE	QI_SEL is interconnected with the on/off switch of the controller integral action I_SEL. With the FM355/455, TI must be switched to zero depending on QI_SEL: If QI_SEL = FALSE, then TI = 0.
QD_SEL	74.2	BOOL	FALSE	QD_SEL is interconnected with the on/off switch of the controller derivative action D_SEL. With the FM355/455, TD must be switched to zero depending on QD_SEL: If QD_SEL = FALSE, then TD = 0.
QWRITE	74.3	BOOL	FALSE	QWRITE is interconnected with the LOAD_PAR and LOAD_OP inputs of the PID_FM instruction of FM355/455. As a result, the data records of the controller channel are downloaded to the FM.
PROCESS	76.0	STRUCT		The PROCESS structure contains the parameters of the process.
GAIN	+0.0	REAL	999.0	The process gain is displayed in the PROCESS.GAIN output.
TM_LAG1	+4.0	REAL	0.0	The slow process time constant is displayed at the PROCESS.TM_LAG1 output.
TM_LAG2	+8.0	REAL	0.0	The fast process time constant is displayed at the PROCESS.TM_LAG2 output.
PV00	+12.0	REAL	0.0	The zero mean value of the process is displayed in the PROCESS.PV00 output. The zero mean value is the steady-state process value when manipulated variable = zero. This corresponds to the ambient temperature when the system is cold.
KIG	+16.0	REAL	0.0	The maximum rise ratio of the process value during heating is displayed in the PROCESS.KIG output.
KIG_C	+20.0	REAL	0.0	The maximum fall ratio of the process value during cooling is displayed in the PROCESS.KIG_C output.
PI_CON	100.0	STRUCT		The PI_CON structure contains the parameters of the PI controller.
GAIN	+0.0	REAL	1.5	PI_CON.GAIN displays the proportional gain of the PI controller. You can improve the value manually and make it effective by setting LOAD_PAR.
TI	+4.0	REAL	3600.0	PI_CON.TI displays the integral action time of the PI controller. You can improve the value manually and make it effective by setting LOAD_PAR.
PID_CON	108.0	STRUCT		The PID_CON structure contains the parameters of the PID controller.
GAIN	+0.0	REAL	1.5	PID_CON.GAIN displays the proportional gain of the PID controller. You can improve the value manually and make it effective by setting LOAD_PAR.
TI	+4.0	REAL	3600.0	PID_CON.TI displays the integral action time of the PID controller. You can improve the value manually and make it effective by setting LOAD_PAR.
TD	+8.0	REAL	0.0	PID_CON.TD displays the derivative action time of the PID controller. You can improve the value manually and make it effective by setting LOAD_PAR.
PI_CON_OLD	120.0	STRUCT		The PI_CON_OLD structure contains the old parameters of the PI controller.

Parameter	Address	Data type	Default	Description
GAIN	+0.0	REAL	1.5	The old proportional gain of the PI controller is displayed at the PI_CON_OLD.GAIN output.
TI	+4.0	REAL	3600.0	The old integral action time of the PI controller is displayed at the PI_CON_OLD.TI output.
PID_CON_OLD	128.0	STRUCT		The PID_CON_OLD structure contains the old parameters of the PID controller.
GAIN	+0.0	REAL	0.0	The old proportional gain of the PID controller is displayed at the PID_CON_OLD.GAIN output.
TI	+4.0	REAL	0.0	The old integral action time of the PID controller is displayed at the PID_CON_OLD.TI output.
TD	+8.0	REAL	0.0	The old derivative action time of the PID controller is displayed at the PID_CON_OLD.TD output.
TU	140.0	REAL	0.0	The determined delay time is displayed at the TU output.
TA	144.0	REAL	0.0	The determined recovery time is displayed at the TA output.

See also

Description TUN_EC (Page 6842)

Mode of operation TUN_EC (Page 6845)

Input parameters TUN_EC (Page 6851)

In/out parameters TUN_EC (Page 6854)

In/out parameters TUN_EC

Table 9-200

Parameter	Address	Data type	Default	Description
ADAPT1ST	148.0	BOOL	FALSE	When you carry out the first optimization of the process or do not wish to use the results of previous optimizations such as manipulated value step size or filter time constants, you must carry out a first adaption. To do this, set ADAPT1ST = TRUE. The Self-Tuner then carries out an initialization routine and in the process sets the default values described in the initialization routine. The Self-Tuner sets ADAPT1ST back to FALSE. It does not make sense to set ADAPT1ST to TRUE at a restart of the CPU in OB100.
ADAPT_ON	148.1	BOOL	FALSE	If you want to trigger on-line adaption, set ADAPT_ON = TRUE. If the subsequent setpoint step-change is greater than MIN_STEP, identification is started (PHASE = 2). The Self-Tuner sets ADAPT_ON back to FALSE.
STEADY	148.2	BOOL	FALSE	When the process value has reached a steady state, you can use STEADY to terminate optimization prematurely, even though the Self-Tuner is still in PHASE 3. You can also use STEADY to carry out a renewed controller design in steady state, after the Self-Tuner has passed into PHASE 4.

Parameter	Address	Data type	Default	Description
COOLID_ON	148.3	BOOL	FALSE	If you want to trigger cooling tuning, set COOLID_ON = TRUE. The manipulated value is set to LLLM_TUN and COOLID_ON is reset to FALSE. If there are multiple heating zones, all pretuning and adaptations should have been completed before cooling tuning is started.
UNDO_PAR	148.4	BOOL	FALSE	The old controller parameters are transferred to the current controller parameters and are active immediately depending on PID_ON. PI_CON = PI_CON_OLD PID_CON = PID_CON_OLD The function for undoing changes can only be used in automatic mode (PHASE 4).
SAVE_PAR	148.5	BOOL	FALSE	The current controller parameters are transferred to the old controller parameters. PI_CON_OLD = PI_CON PID_CON_OLD = PID_CON
LOAD_PAR	148.6	BOOL	FALSE	The changed parameters PI_CON and PID_CON are transferred to the controller and are thus active.
CONZ_ON	148.7	BOOL	FALSE	If CONZ_ON = TRUE, the controller operates with a control zone. During controller optimization the Self-Tuner decides whether controlling is carried out with or without control zone and determines an optimum control zone. After optimization has been carried out in PHASE 4 you can de-activate the control zone with CONZ_ON = FALSE or re-configure the control zone.
CON_ZONE	150.0	REAL	0.0	CON_ZONE specifies the width of the control zone. During controller optimization the Self-Tuner decides whether controlling is carried out with or without control zone and determines an optimum control zone. After optimization has been carried out in PHASE 4 you can de-activate the control zone with CONZ_ON = FALSE or re-configure the control zone.

See also

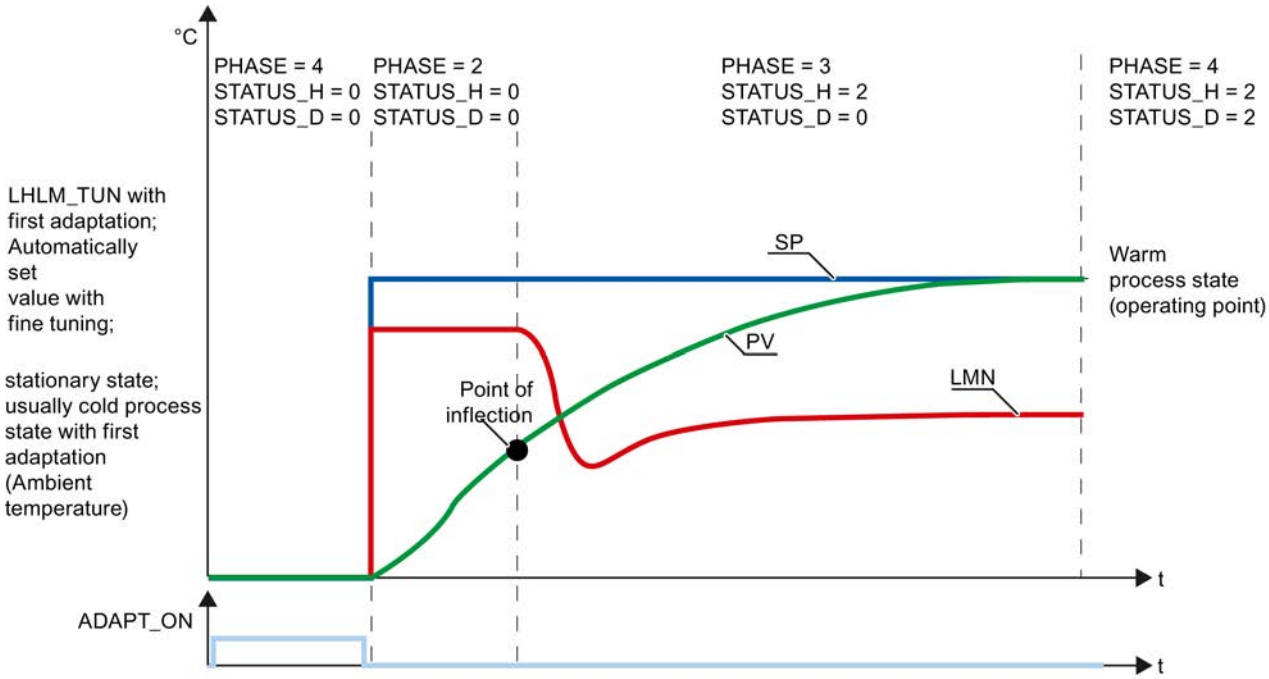
Description TUN_EC (Page 6842)
Mode of operation TUN_EC (Page 6845)
Input parameters TUN_EC (Page 6851)
Output parameters TUN_EC (Page 6852)
Parameter TUN_EC.PHASE (Page 6856)
Parameter TUN_EC.STATUS_H (Page 6858)
Parameter TUN_EC.STATUS_H (Page 6859)
Parameter TUN_EC.STATUS_D (Page 6860)

Parameter TUN_EC.PHASE

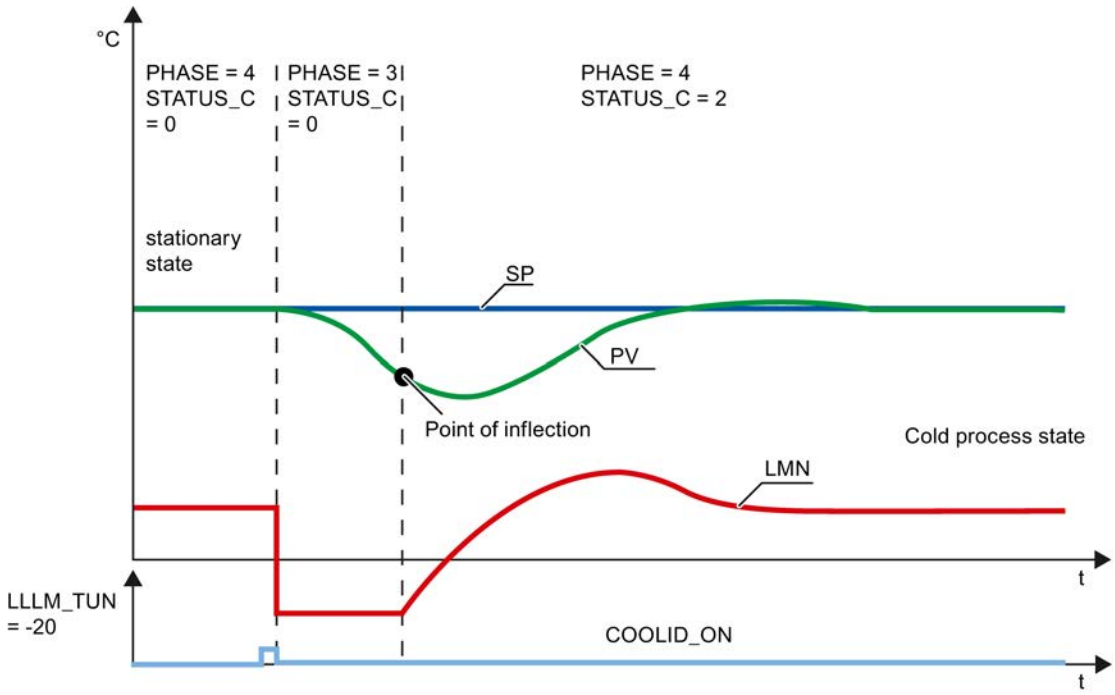
PHASE	Mode	Description
0	Automatic mode: Close-loop control with preset parameters	During the generation of an instance DB for the Self-Tuner, zero is initially assigned to the parameter PHASE. Since TRUE is initially assigned to MAN_ON, the Self-Tuner will change to PHASE 7 (manual mode) after the first run. You can start the first adaption directly from manual mode (ADAPT_ON = TRUE) or change over to automatic mode. When you change over to automatic mode MAN_ON = FALSE for the first time, the parameters are read in from PI_CON. The default values for GAIN and TI are: GAIN = 1.5, TI = 1h.
2	Controller optimization: Searching for point of inflection with constant manipulated value	As soon as you specify a setpoint step-change ≥ MIN_STEP in the positive direction to the operating point of the warm process state, LHLM_TUN is assigned to MAN_OUT during the first adaption, while a value calculated by Self-Tuner is assigned to MAN_OUT during on-line adaption. QMAN_ON = TRUE is also set. Both values are transferred to the PID controller. The PID controller is controlled with them in manual mode. MIN_STEP should be greater than 10% of the operating range of the setpoint and the process value.
3	Controller optimization: Closed-loop control with a PI coarse controller	If the point of inflection of the step response is recognized (STATUS_H = 2) or if the process value has reached 70% of the setpoint step height (STATUS_H = 3), a prudently set PI controller is designed. The controller gain GAIN is limited to the range of 0.2 to 20.0. The controller operates immediately as a PI controller and tries to settle the process to a steady state. The Self-Tuner saves the newly determined PI controller parameters in GAIN, TI and PI_CON. The previous values of the PI controller are saved in PI_CON_OLD. If it takes a long time until the steady state is reached (creeping transient response during temperature processes) you can initiate the control design with the current data when the state is almost steady by setting STEADY = TRUE. You can also re-initiate the controller design with the current data at a later stage in Phase 3 or 4 by specifying STEADY = TRUE. This often improves the controller design further. If an overshoot occurs or if no point of inflection is found, the cause may be that the manipulated value step size LHLM_TUN was selected too large during the first adaption. This does not necessarily produce a bad controller setting. You should select LHLM_TUN approx. 20% smaller at the next first adaption. If the block has recognized the steady state or if the period 30 × TI (TI: Integral-action time of the PI controller set under PHASE = 3) has passed since the setpoint step-change, an improved controller design is realized and the system changes over to PHASE = 4. The Self-Tuner saves the newly determined PI/PID controller parameters in GAIN, TI, TD and PI_CON or PID_CON. The previous values of the PID controller are saved in PID_CON_OLD. If possible, a PID controller and a PI controller are designed. If PID_ON = TRUE, the PID controller is activated, otherwise the PI controller is activated. In PHASE 4 you can switch over bumplessly at any time between the two controllers. The controller gain GAIN is limited in such a manner that the loop gain of the open loop (product of the process gain and the controller gain) lies under 80.0.
4	Automatic mode: Normal closed-loop control with/without control zone	In this phase the controller controls with its optimized parameters. From this state, you can start cooling tuning, fine tuning or pretuning.
5	Automatic mode: Structure switchover at setpoint step-change: I-action deactivated, P(D) control	In the case of a positive setpoint step-change ≥ MIN_STEP the I-action of the controller is de-activated temporarily and the gain is increased slightly, so that a pure P(D) controller is used. In setpoint proximity the I-action is re-activated and the gain is reduced again

PHASE	Mode	Description
6	Automatic mode: Structure switchover at setpoint step-change: Predictive manipulated value specification	Processes with a high time lag cannot be controlled well with a P(D) control system. After a positive setpoint step-change ≥ MIN_STEP, the steady state manipulated variable required for the new setpoint is therefore output as a constant. In setpoint proximity the system changes back bumplessly to PI or PID controller operating mode. This results in a slow action that is, however, free of overshooting.
7	Manual mode	If you set the input MAN_ON to TRUE, the output QMAN_ON is set to TRUE and the output MAN_OUT to MAN. This changes the PID controller over to manual mode. Manual mode takes priority over all other operating modes. Pretuning, fine tuning and cooling tuning are aborted. When manual operation is disabled (MAN_ON = FALSE), the controller changes over to automatic mode (PHASE = 4 in the operation of the Self-Tuner) and controls with the existing controller parameters.

Heating Optimization Phases during First Adaption



Cooling tuning phases



See also

Description TUN_EC (Page 6842)
Mode of operation TUN_EC (Page 6845)
Input parameters TUN_EC (Page 6851)
In/out parameters TUN_EC (Page 6854)
Parameter TUN_EC.STATUS_H (Page 6858)
Parameter TUN_EC.STATUS_H (Page 6859)
Parameter TUN_EC.STATUS_D (Page 6860)

Parameter TUN_EC.STATUS_H

STATUS_H	Description	Remedy
0	Initialization	
1	Point of inflection too low In most cases optimization still results in good controller parameters	To get closer the ideal case, increase LHLM_TUN or decrease the setpoint step-change size.
2	Ideal case: Point of inflection found	

STATUS_H	Description	Remedy
3	Point of inflection too high A safety factor is included. In most cases optimization leads to controller parameters that can still be used but are prudent.	To get closer the ideal case, decrease LHLM_TUN or increase the setpoint step-change size.
11, 12	The time lag could not be detected correctly. Operation continues with an estimated value that can lead to controller parameters that are not optimum. If the process has a pure **VZ1 behavior** (no time lag), the Self-Tuner can often design good controller parameters that are still good. Since calculation is continued with a roughly estimated time lag, it is possible that STATUS_D cannot be determined to 1 (controller for approximating VZ1 behavior) and that the control zone is de-activated, although a control zone makes sense for VZ1 processes.	Repeat optimization and ensure that disturbances do not occur at the process value. If you are not satisfied with the controller parameters, you can at least use the values from PROCESS.GAIN and PROCESS.TM_LAG1 for a VZ1 model and use your own controller design in accordance with the text book.
13	The time lag could not be detected correctly. Operation continues with an estimated value that can lead to controller parameters that are not optimum.	Repeat optimization and ensure that disturbances do not occur at the process value.

See also

Description TUN_EC (Page 6842)
Mode of operation TUN_EC (Page 6845)
Input parameters TUN_EC (Page 6851)
In/out parameters TUN_EC (Page 6854)
Parameter TUN_EC.PHASE (Page 6856)
Parameter TUN_EC.STATUS_H (Page 6859)
Parameter TUN_EC.STATUS_D (Page 6860)

Parameter TUN_EC.STATUS_H

STATUS_C	Description	Remedy
0	Initialization	
1	Turning point was not found. In most cases, optimization still results in good controller parameters.	To get closer to the ideal case, increase LLLM_TUN (for example from -20.0 to -10.0)
2	Ideal case: Cooling point of inflection found	
11, 12, 13	Incorrect estimate The ratio factor is set to the default value: RATIOFAC = 1.0	Repeat optimization and ensure that disturbances do not occur at the process value.

See also

Description TUN_EC (Page 6842)
Mode of operation TUN_EC (Page 6845)
Input parameters TUN_EC (Page 6851)
In/out parameters TUN_EC (Page 6854)
Parameter TUN_EC.PHASE (Page 6856)
Parameter TUN_EC.STATUS_H (Page 6858)
Parameter TUN_EC.STATUS_D (Page 6860)

Parameter TUN_EC.STATUS_D

STATUS_D	Description	Remedy
0	Initialization	
1	The process has almost PT1 behavior;	you can possibly waive the D-action of the contoller.
2	PI and PID controllers possible, VZ2 behavior	
3	Separate PID controller design for very high delay times, dead times or high process order In this case the control zone is switched off.	
4	Incorrect estimate No controller design for on-line adaption. During the first adaption the process is continued with the coarse PI controller from PHASE 3.	Repeat optimization and ensure that disturbances do not occur at the process value.

See also

Description TUN_EC (Page 6842)
Mode of operation TUN_EC (Page 6845)
Input parameters TUN_EC (Page 6851)
In/out parameters TUN_EC (Page 6854)
Parameter TUN_EC.PHASE (Page 6856)
Parameter TUN_EC.STATUS_H (Page 6858)
Parameter TUN_EC.STATUS_H (Page 6859)

TUN_ES

Description TUN_EC

The TUN_ES instruction is used to tune a PID step controller. Depending on the controller, you interconnect TUN_ES with one of the following instructions:

- FM 355 S or FM 455 S
  - PID_FM
- PID Basic Functions
  - CONT_S
  - TCONT_S
- Standard PID Control (PID Professional optional package)
  - PID_ES
- Modular PID Control (PID Professional optional package)
  - PID
  - LMNGEN_S

Call

TUN_ES is called in the constant time base of the cycle time of the calling OB (preferably in a cyclic interrupt OB).

When ADAPT1ST = TRUE , the initialization routine of the TUN_ES instruction is run. The following pre-assignments are made:

- The system switches to PI controller. The parameters of PI_CON are transferred to the controller for this purpose. If the PI_CON.GAIN and PI_CON.TI parameters both have the value 0.0, they are pre-assigned with PI_CON.GAIN = 1.5 and PI_CON.TI = 3600.0 s.
- PROCESS.GAIN = 999.0
- STATUS_H = 0; STATUS_D = 0

Note

Unless explicitly required, do **not** call TUN_ES during a warm restart of the CPU in its initialization routine as is usual for other instructions. If you do, all process parameters will be lost.

If you call the instruction TUN_ES as a multiple instance DB, no technology object is created. No commissioning interface is available. You must commission TUN_EC using a watch table.

CYCLE sampling time

You enter the sampling time in the CYCLE parameter. The sampling time should not exceed 10% of the determined integral action time of the controller. In addition, the sampling time should not exceed 1% of the motor transition time so that the position feedback has a resolution

of at least 1%. You can set the sampling time in the CYCLE parameter of the TUN_ES instruction and the controller. It must agree with the time difference between two calls (cycle time of the cyclic interrupt OB taking into account the reduction ratios).

In connection with the FM355/455 controller modules, the sampling time of TUN_ES must be approximately equal to the resulting cycle time of the controllers on the FM (Parameter setting > Module parameters).

Reaction to error

The error message word RET_VAL is not evaluated by the block.

See also

Mode of operation TUN_ES (Page 6864)

Input parameters TUN_ES (Page 6869)

Output parameters TUN_ES (Page 6871)

In/out parameters TUN_ES (Page 6873)

Parameter TUN_ES.PHASE (Page 6874)

Parameter TUN_ES.STATUS_H (Page 6877)

Parameter TUN_ES.STATUS_D (Page 6878)

Scope of TUN_ES

The Self-Tuner can be used particularly well for:

- Temperature control systems, but also for
- Level control systems and
- Flow control systems

In the case of flow control systems a distinction must be made as to whether only the control valve itself is to be controlled or whether a process with associated delay follows the control valve. In the case of control valve control only, the Self-Tuner cannot be used.

Process requirements

The process must meet the following requirements:

- **Stable asymptotic transient response with associated delay**
  The process value must settle to steady state after a manipulated variable jump. This therefore excludes processes that already exhibit an oscillating response in the absence of closed-loop control, as well as controlled systems without inherent regulation (integrator in the controlled system).
- **Delay times not too large**
  The range of application can be specified based on the ratio of the delay time t_u and response time t_a. The delay time also encompasses any existing dead time.
  The pretuning or adaptation is designed for the range $t_u < 1/10\ t_a$, within which most controlled temperature systems fall.
  For the range of especially long delay times $1/10\ t_a < t_u < 1/3\ t_a$, the design of a utilizable PI controller is generally still also possible.
- **Adequate linear response across a sufficiently large operating range**
  This means that non-linear effects within this operating range can be ignored both during identification and during normal closed-loop control operation. However, it is possible to re-identify the process when there is a change from one operating point to another if the adaptation process is carried out again close to the new operating point and provided that the non-linearity is not passed through during adaptation.
  If certain static non-linearities (e.g., valve characteristic curves) are known, it is always advisable to compensate for them in advance with a polyline in order to linearize the process response.
- **Minimal disturbances in temperature processes**
  Disturbances such as thermal transfer to neighboring zones or heat gains or losses due to changes from one state of matter to another must not influence the overall temperature process to any great extent. If necessary, adaptation at the operating point may be necessary. For example, when zones of an extruder are tuned, all zones must be heated up simultaneously.
- **Quality of the measuring signals**
  The quality of the measured signals has to be sufficient; in other words, the signal-to-disturbance ratio has to be sufficiently high.
- **Process gains not too high**
  The process gains must not be too high. Normalization of the process values is not required. The process gain K can thus include physical units, such as:
  $K = \Delta PV/\Delta LMN$, [K]= °C/%
  The final controller design is based on a calculation of the process gain K and can thus in principle compensate for any values of K. However, K is initially unknown during the learning phase and overshoots cannot be avoided when there are extreme combinations of gain and learning step changes. The overshoot is also reduced by reducing the parameter LHLM_TUN in the heating process or by increasing LLLM_TUN in the cooling process.

Requirements for processes with a control valve with integral action

In processes with control valves with an integral action, further requirements must be met in addition to those specified above:

The motor run time of the control valve must be less than the time required to find a point of inflection after a manipulated variable jump (also refer to the process response figure). If this

is not the case, the process involved is often a flow rate control system in which only the control valve is effective as the dominating process action. It is not advisable to use the Self-Tuner in this case. You can then set the PI step controller manually in accordance with the following rule of thumb:

GAIN = 1, TI = Motor transition time

See also

Mode of operation TUN_ES (Page 6864)

Input parameters TUN_ES (Page 6869)

Output parameters TUN_ES (Page 6871)

In/out parameters TUN_ES (Page 6873)

Parameter TUN_ES.PHASE (Page 6874)

Parameter TUN_ES.STATUS_H (Page 6877)

Parameter TUN_ES.STATUS_D (Page 6878)

Mode of operation TUN_ES

TUN_ES has the following functions for the control or tuning of the controller:

- Manual mode

  The Self-Tuner forces the controller into manual mode and transfers its manual parameters to it. In the case of step controllers with position feedback you can specify the manual value in floating point format and manual value signals such as open/close. In the case of step controllers without position feedback you can only specify the manual value signals.

- Controller tuning
  During tuning of the controller parameters, the actuating time of the actuator is measured and then the controlled system is identified. A controller is designed based on the characteristics of the controlled system. The new controller parameters are calculated on this basis and activated for the controller.

  – First adaption
    Pretuning can only be carried out for a positive setpoint jump. It is usually used during initial commissioning or in case of a serious change in the controlled system characteristics.

  – Fine tuning

    The controller can be fine tuned online during a positive setpoint jump when there is a change from one operating point to another or when there are changes in the process response. Fine tuning is only possible for step controllers with position feedback.

- **Tuning the response to setpoint changes**
  The controller is designed for optimum disturbance response, in principle. The resulting "stringent" parameters would result in overshoots of 10% to 20% of the jump height at setpoint jumps.
  The **structure segmentation** represents an initial remedy. In this case, the proportional or derivative action is located in the feedback loop. A change in the setpoint thus only acts on the integral action of the controller. Structure segmentation may only be used for step controllers with position feedback.
  If structure segmentation is not possible for the controller, the following function can be activated in order to tune the response to setpoint changes:
- **Structure switchover**
  Changeover to a quasi P(D) closed-loop control or predictive manipulated variable specification at large positive setpoint jumps. In the case of step controllers without position feedback, it is only possible to change over to a quasi P(D) closed-loop control.
- **Saving the controller parameters**

  If you assess the current controller parameters to be usable, you can save these before a manual change of the controller parameters in parameters specially provided in the instance data block of the Self-Tuner. If you tune the controller, the saved parameters are overwritten by the values that were valid prior to tuning.
- **Reloading saved controller parameters**
  The last controller parameter settings you saved can be activated for the controller again using this function.
- **Changing the controller parameters**

  If you want to set the controller parameters yourself for the controller, you can assign these through special parameters in the instance DB of the Self-Tuner and transfer them to the controller by means of this function.

No cooling tuning and no control zone mode can be implemented with TUN_ES.

Enabling the functions

Function	Start bit	Enable bit	Start condition: PHASE	Start condition: Setpoint jump at SP
Manual mode	—	MAN_ON LMNS_ON	0 to 7	—
Pretuning with measurement of the motor transition time	ADAPT1ST	—	0, 4	Pos.: > MIN_STEP
Pretuning without measurement of the actuating time of the actuator or fine tuning (only possible for step controller without position feedback)	ADAPT_ON	—	4	Pos.: > MIN_STEP

Function	Start bit	Enable bit	Start condition: PHASE	Start condition: Setpoint jump at SP
Structure switchover	—	STRUC_ON	4	Pos.: > MIN_STEP Neg: > MIN_STEP
Save controller parameters	SAVE_PAR	—	2 to 7	—
Reload controller parameters	UNDO_PAR	—	4	—
Change controller parameters	LOAD_PAR	—	4	—

Note

Start bits are reset after the function has been started successfully. Enable bits remain set.

See also

Description TUN_EC (Page 6861)

Input parameters TUN_ES (Page 6869)

Output parameters TUN_ES (Page 6871)

In/out parameters TUN_ES (Page 6873)

Parameter TUN_ES.PHASE (Page 6874)

Parameter TUN_ES.STATUS_H (Page 6877)

Parameter TUN_ES.STATUS_D (Page 6878)

Structure switchover

Structure segmentation

Structure segmentation is a function of the controller. In the case of closed-loop control with structure segmentation, the proportional or derivative action lies within the feedback loop, so that a change in the setpoint only acts on the integral action of the controller. When setpoint jumps occur, the manipulated variable does not undergo a jump change but instead follows a ramp-shaped course.

The disturbance response is not changed as a result.

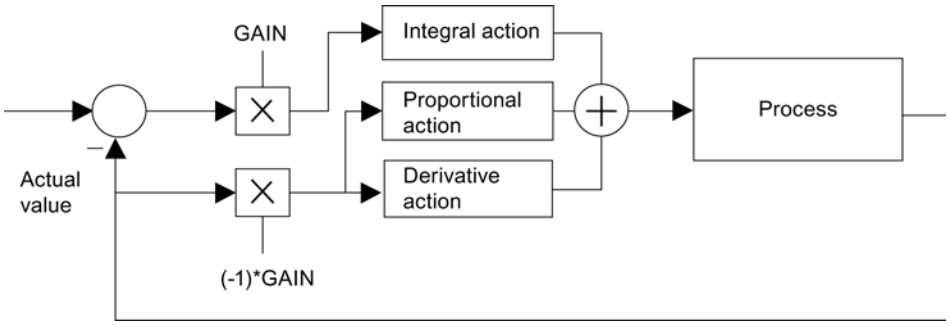


Figure 9-8 Closed-loop control with structure segmentation

In the case of closed-loop control without structure segmentation, the proportional and derivative actions lie within the control deviation, just like the integral action. You can select a structure switchover for this operating mode (STRUC_ON = TRUE).

Closed-loop control with structure segmentation is possible in the following control packages:

- FM355/455
- Standard PID Control
- Modular PID Control

Note

- Structure segmentation must not be used if no position feedback of the control valve is available. This means that you must leave the proportional and derivative actions within the control deviation.
- In closed-loop control with structure segmentation, the dead band in the control deviation is almost without effect.
- With CONT_S and TCONT_S, only closed-loop control without structure segmentation is possible.

Structure switchover when proportional and derivative actions are within the control deviation

The structure switchover is deactivated in the default setting. It should only be selected if structure segmentation is not available. You can activate the structure switchover with

STRUC_ON = TRUE. The block automatically selects between two control measures depending on the total loop gain (product of the process gain and the controller gain):

- Integral action temporarily increased, quasi P(D) control:
  In the case of a positive setpoint jump ≥ MIN_STEP, the integral action of the controller is increased temporarily by a significant amount and the gain is decreased slightly, which means that a quasi P(D) controller is used. In the proximity of the setpoint, the system is switched back to the original values for the integral action and the gain (PHASE = 5).
- Temporary predictive manipulated variable specification:
  Processes with long delay times cannot be controlled well with a P(D) control system. After a positive setpoint jump ≥ MIN_STEP, the manipulated variable required constantly for the new setpoint is therefore output constantly. In the proximity of the setpoint, the system is switched back bumplessly to PI or PID closed-loop control. This results in a slow, but overshoot-free response (PHASE = 6).

Note

In the case of step control without position feedback, only the first control measure (PHASE = 5) is possible.

If you do not achieve good results (transient response is slow) with the structure switchover in the case of positive setpoint jumps (for example, during heating processes), you can deactivate the structure switchover with STRUC_ON = FALSE, on the condition that small overshoots may occur.

Controller structure modes	PFDB_SEL, DFDB_SEL [1]	STRUC_ON
Activate structure segmentation	TRUE	FALSE
Activate structure switchover	FALSE	TRUE

[1] Controller parameters PFDB_SEL, DFDB_SEL: With CONT_S, these parameters are not available. This corresponds to the PFDB_SEL = DFDB_SEL = FALSE setting (without structure segmentation).

Note

- Structure segmentation prevents overshooting of the controlled variable when setpoint jumps occur. Select structure segmentation if possible.
- If the controller does not allow structure segmentation, activate structure switchover. The structure switchover must not be activated at the same time as the structure segmentation. You can deactivate structure switchover if rapid tuning is required and small overshoots are permitted at positive setpoint jumps.

Input parameters TUN_ES

Table 9-201

Parameter	Address	Data type	Default	Description
SP	0.0	REAL	0.0	Enter the setpoint at the SP input. It is interconnected with the controller via SP_OUT. Note that the setpoint must lie within the setpoint limits of the controller.
PV	4.0	REAL	0.0	PV is interconnected with the process value of the controller.
MAN	8.0	REAL	0.0	Enter the manual value at the MAN input. If MAN_ON = TRUE, the manual value MAN is transferred to the manipulated variable of the controller. When MAN_ON = TRUE and MAN = 0.0, the heating is off. If MAN_ON = FALSE, the manipulated variable is specified by the controller. Values from -100 to 100% are permitted.
LMNR	12.0	REAL	0.0	LMNR is interconnected with the position feedback of the controller (if any). Values from 0.0 to 100% are permitted.
MIN_STEP	16.0	REAL	10.0	You can specify the minimum setpoint jump above which pretuning or adaptation can be carried out in the MIN_STEP input. If structure switchover (STRUC_ON=TRUE) is activated, the setpoint jump must be greater than MIN_STEP in order for the structure switchover activation to take place. Recommendation: MIN_STEP ≥ 10 % of the required setpoint/process value range
LHLM_TUN	20.0	REAL	80.0	At the LHLM_TUN input, specify the manipulated variable jump for the first setting. The manipulated variable jump is calculated automatically during subsequent fine tuning. Values from 0.0 to 100% are permitted.
NORM_FAC	24.0	REAL	1.0	A normalization factor is applied for controllers with a normalized controller gain (Standard PID Control). The normalization factor is calculated on the basis of the range limits of the process value of the controller and is interconnected with the NORM_FAC input.
PULSE_TM	28.0	TIME	0 ms	PULSE_TM is interconnected with the minimum pulse duration of the controller. PULSE_TM ≥ 1 ms
C_LMNUP	32.0	BOOL	FALSE	C_LMNUP is interconnected with the manipulated variable signal QLMNUP of the controller.
C_LMNDN	32.1	BOOL	FALSE	C_LMNDN is interconnected with the manipulated variable signal QLMNDN of the controller.
MAN_ON	32.2	BOOL	FALSE	If MAN_ON = TRUE, the manipulated variable of the controller is specified via the MAN input. If MAN_ON = FALSE, the manipulated variable is specified by the controller.
LMNR_HS	32.3	BOOL	FALSE	LMNR_HS is interconnected with the endstop signal LMNR_HR of the controller.
LMNR_ON	32.4	BOOL	FALSE	If LMNR_ON = TRUE, you can use LMNS_ON = TRUE or MAN_ON = TRUE to switch a step controller with position feedback over to manual mode. If LMNR_ON = FALSE, you can use LMNS_ON = TRUE to switch a step controller without position feedback over to manual mode.
LMNS_ON	32.5	BOOL	TRUE	If LMNS_ON is set, you can use the LMNUP and LMNDN parameters to manipulate the actuating signals of the controller.

Parameter	Address	Data type	Default	Description
LMNUP	32.6	BOOL	FALSE	You can use LMNUP to manipulate the actuating signal OPEN of the controller.
LMNDN	32.7	BOOL	FALSE	You can use LMNDN to manipulate the actuating signal CLOSE of the controller.
PID_ON	33.0	BOOL	FALSE	You can specify at the PID_ON input whether the tuned controller is to operate as a PI controller or a PID controller. PID controller: PID_ON = TRUE PI controller: PID_ON = FALSE With certain controlled system types, however, it is possible that only a PI controller will be designed, despite PID_ON = TRUE.
STRUC_ON	33.1	BOOL	FALSE	If STRUC_ON = TRUE, a structure switchover (from PI(D) to P(D) control) ensures that overshoots are largely avoided at setpoint jumps. The STRUC_ON input must not be set = TRUE if the proportional action or derivative action is fed into the feedback loop of the controller.
WRITE_DIS	33.2	BOOL	FALSE	In connection with the controller module, the Self-Tuner must know whether the driver block is ready to send data to the FM. This is the case if LOAD_OP = LOAD_PAR = FALSE. The result of the OR operation of LOAD_OP and LOAD_PAR is interconnected with WRITE_DIS.
CYCLE	34.0	TIME	100 ms	The sampling time must be configured in the CYCLE input. CYCLE must correspond to the sampling time of the controller to be set and the time difference between two calls (cycle time of the cyclic interrupt OB taking into account the reduction ratios). ≥ 1 ms

See also

Description TUN_EC (Page 6861)
Mode of operation TUN_ES (Page 6864)
Output parameters TUN_ES (Page 6871)
In/out parameters TUN_ES (Page 6873)
Parameter TUN_ES.PHASE (Page 6874)
Parameter TUN_ES.STATUS_H (Page 6877)
Parameter TUN_ES.STATUS_D (Page 6878)

Output parameters TUN_ES

Table 9-202

Parameter	Address	Data type	Default	Description
MAN_OUT	38.0	REAL	0.0	MAN_OUT is connected to the manual input value of the controller. If MAN_ON = TRUE, the Self-Tuner outputs the value specified at the MAN input to the MAN_OUT output.
SP_OUT	42.0	REAL	0.0	SP_OUT is connected to the setpoint input of the controller. The Self-Tuner writes the setpoint specified at the SP input to the SP_OUT output. If manual mode MAN_ON = TRUE, the setpoint is tracked to the process value: SP_OUT = PV.
GAIN	46.0	REAL	1.5	GAIN is interconnected with the proportional gain (controller gain) of the controller.
TI	50.0	TIME	1 h	TI is interconnected with the integral action time (reset time) of the controller.
TD	54.0	TIME	0 ms	TD is interconnected with the derivative action time (rate time) of the controller.
TM_LAG	58.0	TIME	1 s	TM_LAG is interconnected with the time lag of the controller differentiator.
MTR_TM	62.0	TIME	30 s	MTR_TM is interconnected with the motor run time of the controller.
DEADB_W	66.0	REAL	0.0	DEADB_W is interconnected with the dead band width of the controller.
PHASE	70.0	INT	0	Parameter TUN_ES.PHASE (Page 6874) displays the current sequence of the controller tuning. During the tuning of Standard PID Control, the PHASE output is interconnected with the PHASE input of the PID_ES instruction.
STATUS_H	72.0	INT	0	Parameter TUN_ES.STATUS_H (Page 6877) displays tuning results regarding the search for the point of inflection during the heating process.
STATUS_D	74.0	INT	0	Parameter TUN_ES.STATUS_D (Page 6878) displays tuning results regarding the controller design during the heating process.
QMAN_ON	76.0	BOOL	FALSE	QMAN_ON is interconnected with the manual/automatic changeover of the controller.
QLMNS_ON	76.1	BOOL		QLMNS_ON is interconnected with the "Enable manipulated signals" parameter of the controller
QLMNUP	76.2	BOOL		QLMNUP is interconnected with the "Manipulated signal open" parameter of the controller
QLMNDN	76.3	BOOL		QLMNDN is interconnected with the "Manipulated signal close" parameter of the controller
QI_SEL	76.4	BOOL	TRUE	QI_SEL is interconnected with the on/off switch of the controller integral action I_SEL. In the case of FM355/455, TI must be switched to zero depending on QI_SEL: If QI_SEL = FALSE, then TI = 0.
QD_SEL	76.5	BOOL	FALSE	QD_SEL is interconnected with the on/off switch of the controller derivative action D_SEL. In the case of FM355/455, TD must be switched to zero depending on QD_SEL: If QD_SEL = FALSE, then TD = 0.
QWRITE	76.6	BOOL	FALSE	QWRITE is interconnected with the LOAD_PAR and LOAD_OP inputs of the PID_FM instruction of FM355/455. As a result, the data records of the controller channel are downloaded to the FM.
PROCESS	78.0	STRUCT		The PROCESS structure contains the parameters of the process.
GAIN	+0.0	REAL	999.0	The process gain is displayed in the PROCESS.GAIN output.

Parameter	Address	Data type	Default	Description
TM_LAG1	+4.0	REAL	0.0	The slow process time constant is displayed at the PROCESS.TM_LAG1 output.
TM_LAG2	+8.0	REAL	0.0	The fast process time constant is displayed at the PROCESS.TM_LAG2 output.
PV00	+12.0	REAL	0.0	The zero mean value of the process is displayed at the PROCESS. PV00 output. The zero mean value is the steady-state process value when manipulated variable = zero. This corresponds to the ambient temperature when the system is cold.
KIG	+16.0	REAL	0.0	The maximum rise ratio of the process value during heating is displayed in the PROCESS.KIG output.
MTR_TM	+20.0	REAL	0.0	The motor run time is displayed at the MTR_TM output.
PI_CON	102.0	STRUCT		The PI_CON structure contains the parameters of the PI controller.
GAIN	+0.0	REAL	1.5	PI_CON.GAIN displays the proportional gain of the PI controller. You can improve the value manually and make it effective by setting LOAD_PAR.
TI	+4.0	REAL	3600.0	PI_CON.TI displays the integral action time of the PI controller. You can improve the value manually and make it effective by setting LOAD_PAR.
PID_CON	110.0	STRUCT		The PID_CON structure contains the parameters of the PID controller.
GAIN	+0.0	REAL	0.0	PID_CON.GAIN displays the proportional gain of the PID controller. You can improve the value manually and make it effective by setting LOAD_PAR.
TI	+4.0	REAL	0.0	PID_CON.TI displays the integral action time of the PID controller. You can improve the value manually and make it effective by setting LOAD_PAR.
TD	+8.0	REAL	0.0	PID_CON.TD displays the derivative action time of the PID controller. You can improve the value manually and make it effective by setting LOAD_PAR.
PI_CON_OLD	122.0	STRUCT		The PI_CON_OLD structure contains the old parameters of the PI controller.
GAIN	+0.0	REAL	1.5	The old proportional gain of the PI controller is displayed at the PI_CON_OLD.GAIN output.
TI	+4.0	REAL	3600.0	The old integral action time of the PI controller is displayed at the PI_CON_OLD.TI output.
PID_CON_OLD	130.0	STRUCT		The PID_CON_OLD structure contains the old parameters of the PID controller.
GAIN	+0.0	REAL	1.5	The old proportional gain of the PID controller is displayed at the PID_CON_OLD.GAIN output.
TI	+4.0	REAL	3600.0	The old integral action time of the PID controller is displayed at the PID_CON_OLD.TI output.
TD	+8.0	REAL	0.0	The old derivative action time of the PID controller is displayed at the PID_CON_OLD.TD output.
TU	142.0	REAL	0.0	The determined delay time is displayed at the TU output.
TA	146.0	REAL	0.0	The determined recovery time is displayed at the TA output.

See also

Description TUN_EC (Page 6861)
Mode of operation TUN_ES (Page 6864)
Input parameters TUN_ES (Page 6869)
In/out parameters TUN_ES (Page 6873)

In/out parameters TUN_ES

Table 9-203

Parameters	Address	Data type	Default	Description
ADAPT1ST	150.0	BOOL	FALSE	When you carry out the first optimization of the process or do not wish to use the results of previous optimizations such as manipulated value step size or filter time constants, you must carry out a first adaption. To do this, set ADAPT1ST = TRUE. The Self-Tuner then carries out an initialization routine and, in the process, sets the default values described in the initialization routine. The Self-Tuner sets ADAPT1ST back to FALSE. It does not make sense to set ADAPT1ST to TRUE at a restart of the CPU in OB100.
ADAPT_ON	150.1	BOOL	FALSE	If you want to trigger on-line adaption, set ADAPT_ON = TRUE. If the subsequent setpoint step-change is greater than MIN_STEP, identification is started (PHASE = 2). The Self-Tuner sets ADAPT_ON back to FALSE.
STEADY	150.2	BOOL	FALSE	When the process value has reached a steady state, you can use STEADY to terminate optimization prematurely, even though the Self-Tuner is still in PHASE 3. After the Self-Tuner has passed into PHASE 4, you can also use STEADY to carry out a renewed controller design in steady state.
UNDO_PAR	150.3	BOOL	FALSE	The old controller parameters are transferred to the current controller parameters and are active immediately depending on PID_ON. PI_CON = PI_CON_OLD PID_CON = PID_CON_OLD The function for undoing changes can only be used in automatic mode (PHASE 4).
SAVE_PAR	150.4	BOOL	FALSE	The current controller parameters are transferred to the old controller parameters. PI_CON_OLD = PI_CON PID_CON_OLD = PID_CON
LOAD_PAR	150.5	BOOL	FALSE	The changed parameters PI_CON und PID_CON are transferred to the controller and are thus active.

See also

Description TUN_EC (Page 6861)
Mode of operation TUN_ES (Page 6864)
Input parameters TUN_ES (Page 6869)
Output parameters TUN_ES (Page 6871)
Parameter TUN_ES.PHASE (Page 6874)

Parameter TUN_ES.STATUS_H (Page 6877)
Parameter TUN_ES.STATUS_D (Page 6878)

Parameter TUN_ES.PHASE

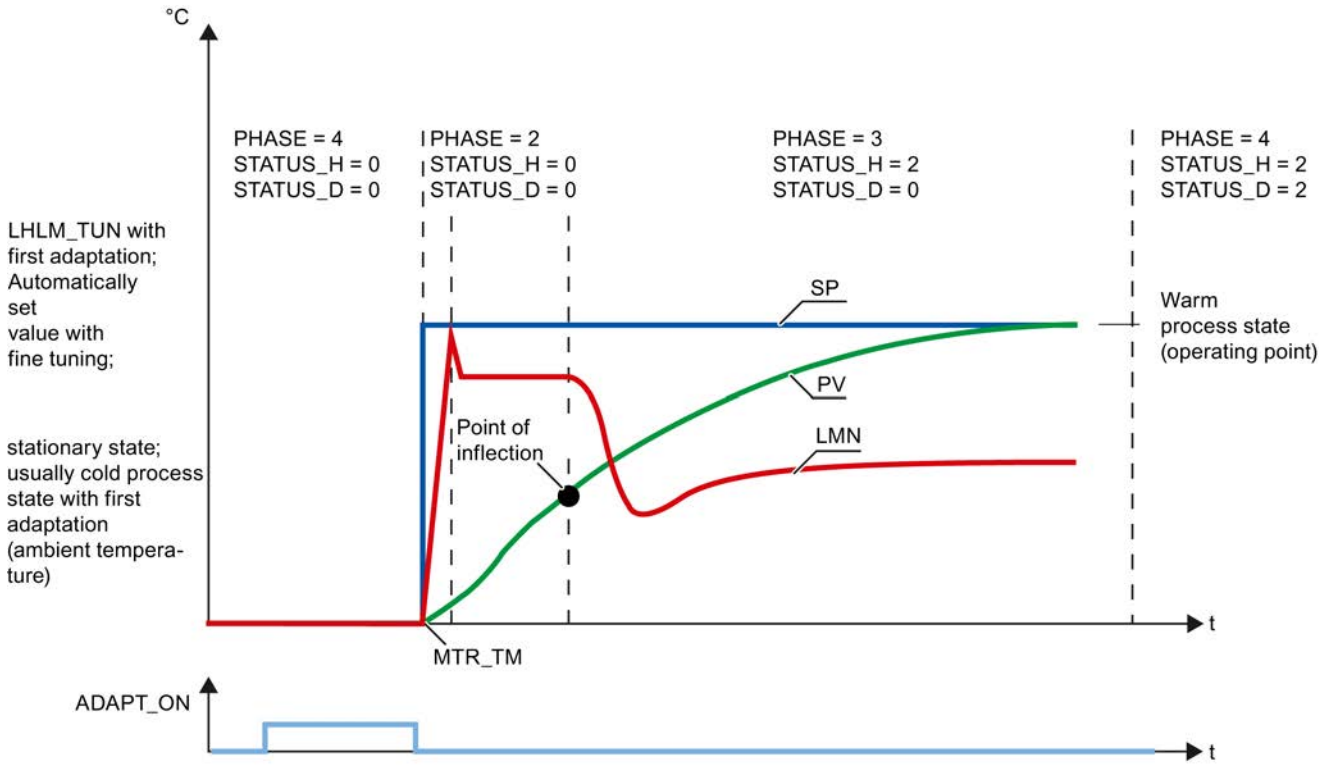
PHASE	Mode	Description
0	Automatic mode: Close-loop control with preassigned parameters	During the generation of an instance DB for the Self-Tuner, thePHASE parameter is pre-assigned with zero. Since LMNS_ON is pre-assigned with TRUE, the Self-Tuner will switch to PHASE 7 (manual mode) after the first run. You can start the pretuning directly from manual mode (ADAPT_ON = TRUE) or switch to automatic mode. When you switch to automatic mode MAN_ON = FALSE for the first time, the parameters of PI_CON are applied. The default values for GAIN and TI are: GAIN = 1.5, TI = 1h. Exiting from the phase via MAN_ON, LMNS_ON, or ADAPT1ST
1	Controller tuning: Heating off and waiting for a setpoint jump	After tuning has been started with ADAPT1ST = TRUE, the Self-Tuner closes the control valve (manipulated variable = 0 or LMNS_ON = LMNDN = TRUE) and waits for a positive setpoint jump ≥ MIN_STEP.
2	Controller tuning: Motor transition time measurement, moving to a constant control valve setting and searching for point of inflection	As soon as you specify a setpoint jump ≥ MIN_STEP in the positive direction to the operating point of the warm process state, with pretuning MAN_OUT is set to LHLM_TUN and with fine tuning MAN_OUT is set to a value calculated by the self-tuner. MIN_STEP should be greater than 10% of the operating range of the setpoint and the process value. In the case of step control with position feedback (LMNR_ON = TRUE), the value TRUE is assigned to QMAN_ON. The controller controls the valve to the value of MAN_OUT, and TUN_ES calculates the motor transition time MTR_TM. In the case of step control without position feedback, QLMNS_ON and QLMNUP are set to TRUE, and the valve is moved to the high endstop position. When the high endstop is reached (LMNR_HS =TRUE), TUN_ES calculates the motor transition time and forwards it to the controller. The valve is then moved toward the closed position until the value of MAN_OUT is reached (QLMNDN = TRUE).

PHASE	Mode	Description
3	Controller tuning: Closed-loop control with a PI coarse controller	When the point of inflection of the jump response is recognized (STATUS_H = 2) or when the process value has reached 70% of the setpoint jump height (STATUS_H = 3), a PI controller with conservative settings is designed. The controller gain GAIN is limited to the range from 0.2 to 20.0. As of this point, the controller immediately starts working as a PI controller and attempts to bring the controlled system to a steady-state condition. The Self-Tuner stores the newly determined PI controller parameters in GAIN, TI, and PI_CON. The previous values of the PI controller are saved in PI_CON_OLD. If it takes a long time to reach steady-state condition (creeping transient response during temperature processes), you can set STEADY = TRUE to initiate controller design with the current data when steady-state condition is almost reached. You can also set STEADY = TRUE to re-initiate the controller design with the current data at a later stage in Phase 3 or 4. This will often provide some additional improvement to the controller design. If an overshoot occurs or if no point of inflection is found, the manipulated variable jump size LHLM_TUN selected for pretuning may have been too large. This does not necessarily result in poor controller tuning. For the next pretuning, the value you select for LHLM_TUN should be approximately 20% less. If the block has recognized steady-state condition or if the period 30 × TI (TI: integral action time of the PI controller set in PHASE = 3) has elapsed since the setpoint jump, the controller design has been improved and the system switches to PHASE = 4. The Self-Tuner stores the newly determined PI/PID controller parameters in GAIN, TI, TD, and PI_CON or PID_CON. The previous values of the PID controller are saved in PID_CON_OLD. If possible, a PID controller and a PI controller are designed. If PID_ON = TRUE, the PID controller is activated, otherwise the PI controller is activated. In PHASE 4, it is possible to switch between the two controllers bumplessly at any time. For difficult processes, the block always designs a PI controller, irrespective of PID_ON. The controller gain GAIN is limited in such a manner that the loop gain of the open loop (product of the process gain and the controller gain) is less than 40.0.
4	Automatic mode: Normal closed-loop control with/without control zone	In this phase, the controller controls using its tuned parameters. From this state, you can start fine tuning or pretuning.
5	Automatic mode, structure switchover at setpoint jump: Integral action deactivated, P(D) control	In the case of a positive setpoint jump ≥ MIN_STEP, the integral action of the controller is increased temporarily by a significant amount and the gain is decreased slightly, so that a quasi P(D) controller is used. In the proximity of the setpoint, the system is switched back to the original values for the integral action and the gain (PHASE = 5 in the Self-Tuner sequence).

PHASE	Mode	Description
6	Automatic mode, structure switchover at setpoint jump: Predictive manipulated variable specification	Only for step controller with position feedback Processes with long delay times cannot be controlled well with a P(D) control system. After a positive setpoint jump ≥ MIN_STEP, the manipulated variable required constantly for the new setpoint is therefore output continually. In the proximity of the setpoint, the system is switched back bumplessly to PI or PID closed-loop control. This results in a slow, but overshoot-free response
7	Manual mode	Manual mode takes priority over all other operating modes. Pretuning or fine tuning is canceled. When manual mode is deactivated (MAN_ON = FALSE and LMNS_ON = FALSE), the controller changes over to automatic mode (PHASE = 4 in the Self-Tuner sequence) and controls using the existing controller parameters. If you use a step controller with position feedback (LMNR_ON = TRUE), you can use LMNS_ON = TRUE or MAN_ON = TRUE to switch to manual mode. If you use a step controller without position feedback (LMNR_ON = FALSE), you can only use LMNS_ON = TRUE to switch to manual mode. If you set the MAN_ON input to TRUE, the QMAN_ON output is set to TRUE and the MAN_OUT output is set to MAN. If you set the LMNS_ON input to TRUE, the QLMNUP output is set to LMNUP and the QLMNDN output is set to LMNDN.

Heating tuning with step controller without position feedback

The diagram shows the phase steps of pretuning with measurement of the motor run time.



See also

Description TUN_EC (Page 6861)
Mode of operation TUN_ES (Page 6864)
Input parameters TUN_ES (Page 6869)
In/out parameters TUN_ES (Page 6873)
Parameter TUN_ES.STATUS_H (Page 6877)
Parameter TUN_ES.STATUS_D (Page 6878)

Parameter TUN_ES.STATUS_H

STATUS_H	Description	Remedy
0	Initialization	
1	Point of inflection too low In most cases optimization still results in good controller parameters.	To get closer the ideal case, increase LHLM_TUN or decrease the setpoint step-change size
2	Ideal case: Point of inflection found	
3	Point of inflection too high A safety factor is included. In most cases optimization leads to controller parameters that can still be used but are prudent.	To get closer the ideal case, decrease LHLM_TUN or increase the setpoint step-change size.
11, 12	The time lag could not be detected correctly. Operation continues with an estimated value that can lead to controller parameters that are not optimum. If the process has a pure VZ1 behavior (no time lag), the Self-Tuner can often design good controller parameters that are still good. Since calculation is continued with a roughly estimated time lag, it is possible that STATUS_D cannot be determined to 1 (controller for approximating VZ1 behavior) and that the control zone is de-activated, although a control zone makes sense for VZ1 processes.	Repeat optimization and ensure that disturbances do not occur at the process value. If you are not satisfied with the controller parameters, you can at least use the values from PROCESS.GAIN and PROCESS.TM_LAG1 for a VZ1 model and use your own controller design in accordance with the text book.
13	The time lag could not be detected correctly. Operation continues with an estimated value that can lead to controller parameters that are not optimum.	Repeat optimization and ensure that disturbances do not occur at the process value.

See also

Description TUN_EC (Page 6861)
Mode of operation TUN_ES (Page 6864)
Input parameters TUN_ES (Page 6869)
In/out parameters TUN_ES (Page 6873)
Parameter TUN_ES.PHASE (Page 6874)
Parameter TUN_ES.STATUS_D (Page 6878)

Parameter TUN_ES.STATUS_D

STATUS_D	Description
0	Initialization
1	Only PI controller design, since the process almost has PT1 behavior
2	Ideal case: PI and PID controllers possible, VZ2 behavior
3	Only PI controller design, since there is a very high time lag / dead time or high order
4	Incorrect estimate No controller design for on-line adaption. During the first adaption the process is continued with the coarse PI controller from PHASE = 3. Repeat optimization and ensure that disturbances do not occur at the process value.

See also

Description TUN_EC (Page 6861)

Mode of operation TUN_ES (Page 6864)

Input parameters TUN_ES (Page 6869)

In/out parameters TUN_ES (Page 6873)

Parameter TUN_ES.PHASE (Page 6874)

Parameter TUN_ES.STATUS_H (Page 6877)

Standard PID Control

Technical specifications of the instructions

CPU utilization

The controlling efficiency and therefore processing speed of actual control loops depends exclusively on the performance of the CPU you are using. To estimate the utilization of a particular CPU due to installation of Standard PID Control, the following information applies:

- The user memory only has to contain a single instance of the instruction, regardless of the number of controllers.
- You will need a DB with approximately 0.5 Kbytes for each controller.
- There is no memory requirement for an L-Stack.
- Alarms are not delayed by PID_CP or PID_ES.
- Key data for typical execution times (process times) of blocks when default parameters for control mode are used:

Instruction	Boundary conditions	Processing time (in ms) CPU 315–2AG10	Processing time (in ms) CPU 416–2XK02
PID_CP	Typical boundary conditions	1.3	0.14
PID_ES	Without position feedback, typical boundary conditions	1.5	0.16

Work memory assignment

Refer to the following tables for the size of the required area in the user memory and, thus, the number of control loops that could theoretically be installed based on the available memory capacity:

Instruction	Load memory requirement	Work memory requirement	Local data
PID_CP	8956 bytes	7796 bytes	122 bytes
PID_ES	9104 bytes	7982 bytes	152 bytes
LP_SCHED	1064 bytes	976 bytes	20 bytes

Instance DB or global DB	Load memory requirement	Work memory requirement:
DB for PID_CP	1168 bytes	510 bytes
DB for PID_ES	1124 bytes	484 bytes
DB_LOOP (for 5 control loops)	184 bytes	100 bytes
DB_RMPSK (with starting point and 4 interpolation points)	142 bytes	78 bytes

Sampling time

The minimum selectable sampling time is dependent on the performance class of the CPU used.

Note

The limited computational accuracy limits the achievable sampling time. As the sampling time becomes shorter, the constants in the algorithms take on smaller and smaller values. This can result in incorrect calculation of the manipulated variable.

Recommendation:

- S7-300: Sampling time ≥ 20 ms
- S7-400: Sampling time ≥ 5 ms

PID_CP

Description PID_CP

In addition to the functions in the setpoint and process value branches, the instruction implements a complete PID-controller with continuous manipulated variable output and the option of manually influencing the manipulated variable. Subfunctions can be enabled or disabled.

With this instruction you are able to control technical processes and systems with continuous input and output variables on the SIMATIC S7 automation systems. Slow (temperatures, filling levels etc.) as well as fast controlling systems (flow, speed etc.) can be controlled.

The controller can be used individually as a fixed setpoint controller, or in multi-loop control systems as a cascade, blending, or ratio controller.

Startup

The PID_CP instruction features an initialization routine that is executed when input parameter COM_RST = TRUE is set.

The integral action is set to the initialization value I_ITLVAL at startup and read out at output LMN_I. When it is called in a cyclic interrupt level, it then continues to work starting at this value.

When ramp/soak is enabled, the time intervals DB_NBR PI[0 ... NBR_PTS].TMV between the time slices are added up and displayed at the total time T_TM and total remaining time RT_TM outputs. In the case of online changes of PI[n].TMV or specification of TM_CONT and TM_SNBR, the total time and the total remaining time of the ramp/soak profile change. Since the calculation of T_TM and RS_TM with many time slices significantly increases the processing time of the RMP_SOAK function, the calculation is only carried out at a restart or when TUPDT_ON = TRUE.

All other outputs are set to their default values.

Hot restart

With a hot restart, the operating state that existed at the time of the interruption is assumed. The control continues to operate with the values that were calculated at the moment the interruption occurred.

Call

The instruction PID_CP must be called equidistant. To achieve this, use a cyclic interrupt level (for example, OB35).

If you call the instruction PID_CP as a multiple instance DB, no technology object is created. No parameter assignment interface or commissioning interface is available. You must assign parameters for PID_CP directly in the multiple instance DB and commission it via a watch table.

Generating control deviation PID_CP

The following functions are always active and cannot be disabled.

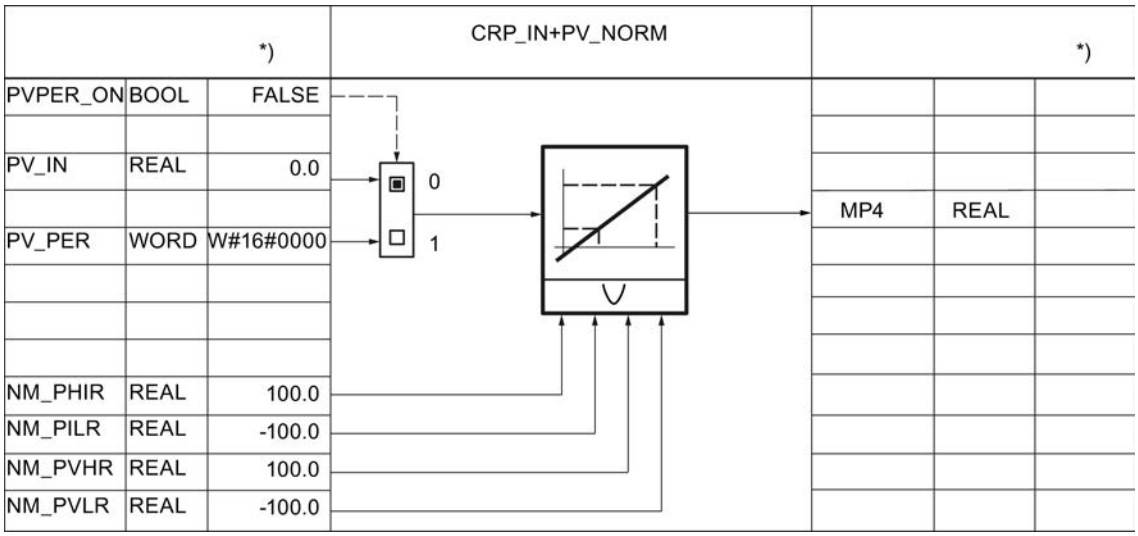
- Setpoint limiting SP_LIMIT,
- Process value monitoring for absolute values PV_ALARM,
- Process value monitoring for size of rate of change ROCALARM,
- Control deviation monitoring for absolute values ER_ALARM,
- Limiting of the output signal LMNLIMIT.

The default values are selected in such a manner that operation requires no individual adjustment. To achieve good controller results, you always have to adapt the parameters of these functions to your process.

Processing the process value

The process value is scaled to the physical value.

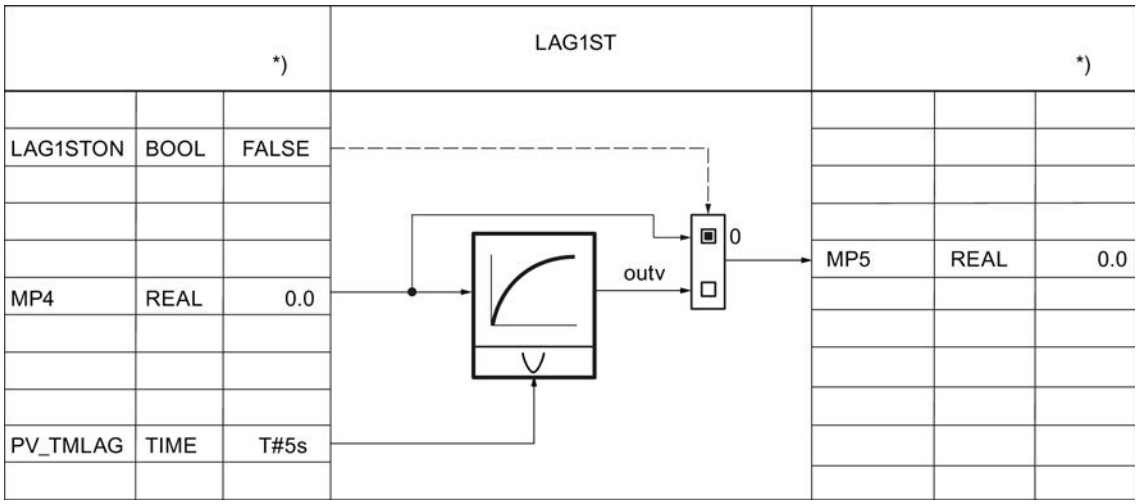
Scaling (PV_NORM) (Page 9654)



*) Default for creation of new technology object.

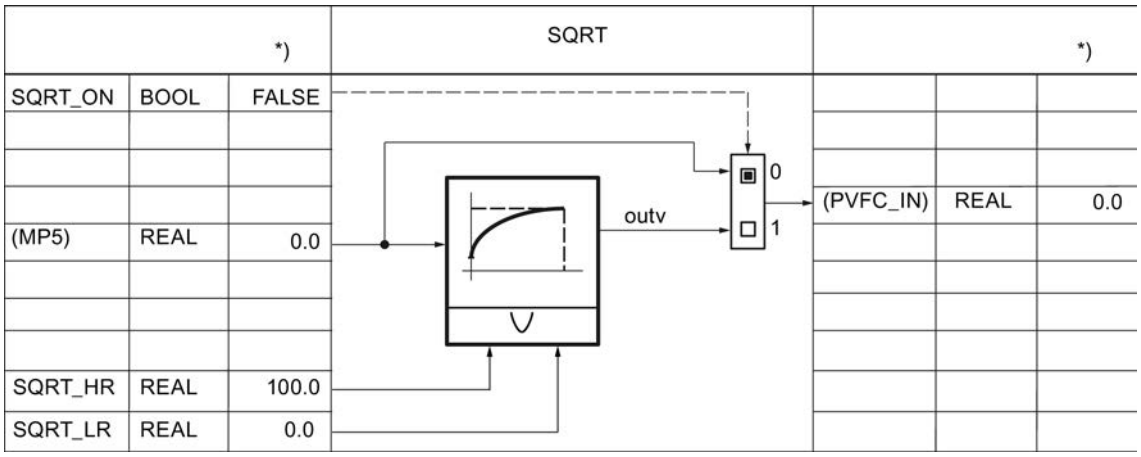
The process value can be smoothed with a first-order delay element.

Smooth controlled variable (LAG1ST) (Page 9655)



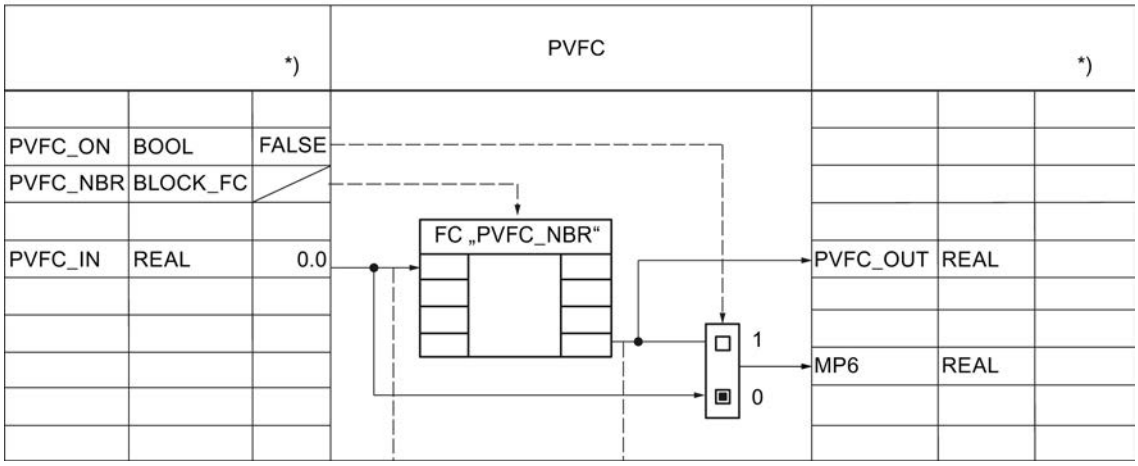
The process value can be linearized with the square root. The following parameters are required for this:

Square root (SQRT) (Page 9656)



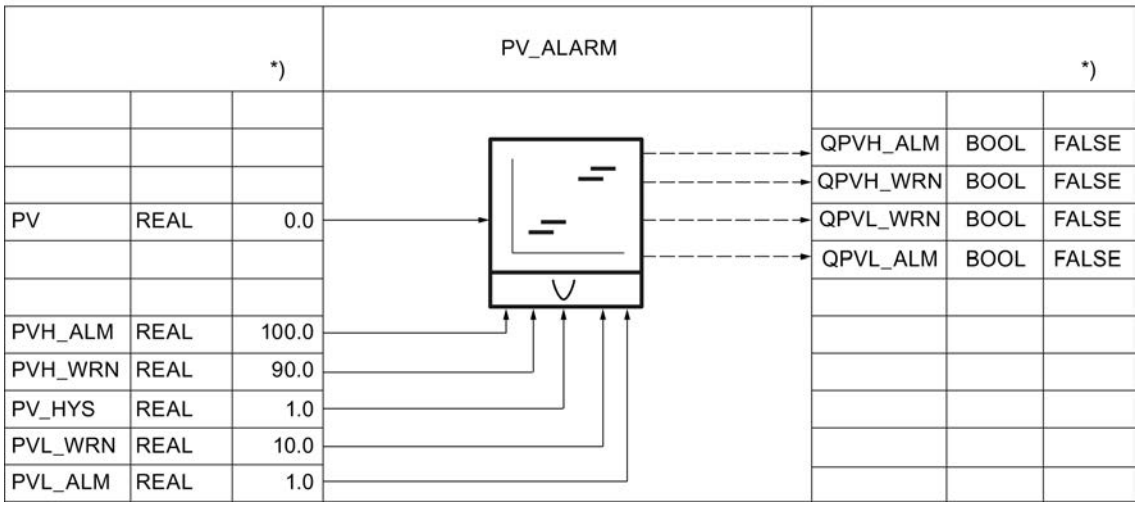
The process value can be prepared with a user-specific function. The following parameters are required for this: The interconnection must be programmed in the user FC.

FC call in the process value branch (PVFC) (Page 9658)



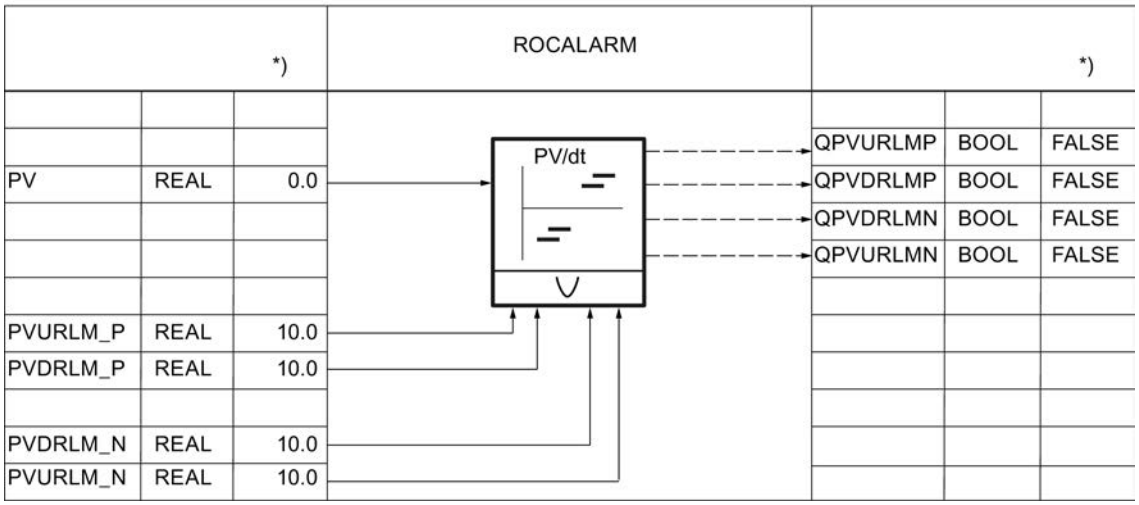
The process value is always monitored with the following parameters for warning and alarm limits.

Monitor process value for limit values (PV_ALARM) (Page 9658)



The rate of change of the process value is always limited with the following parameters.

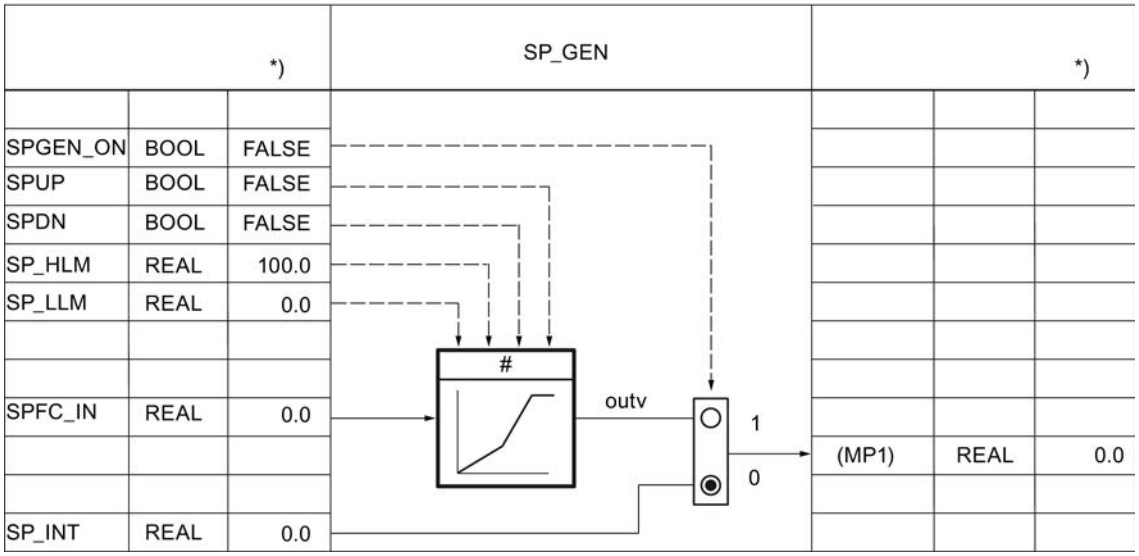
Monitor rate of change of the process value (ROCALARM) (Page 9660)



Processing setpoints

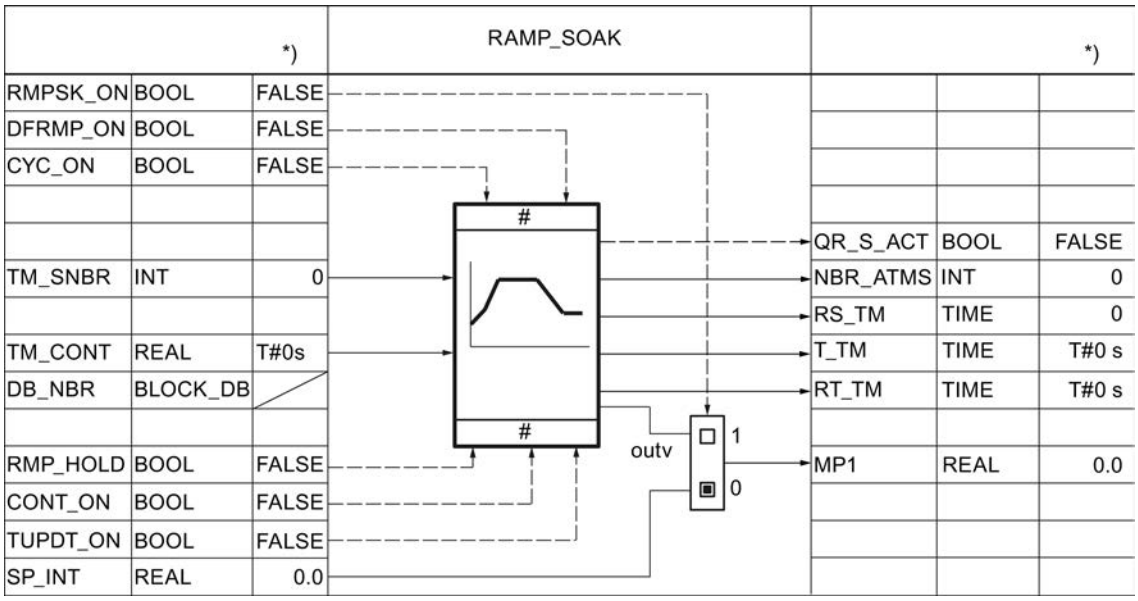
The internal setpoint SP_IN can be incremented or decremented with the following parameters.

Setpoint generator (SP_GEN) (Page 9661)



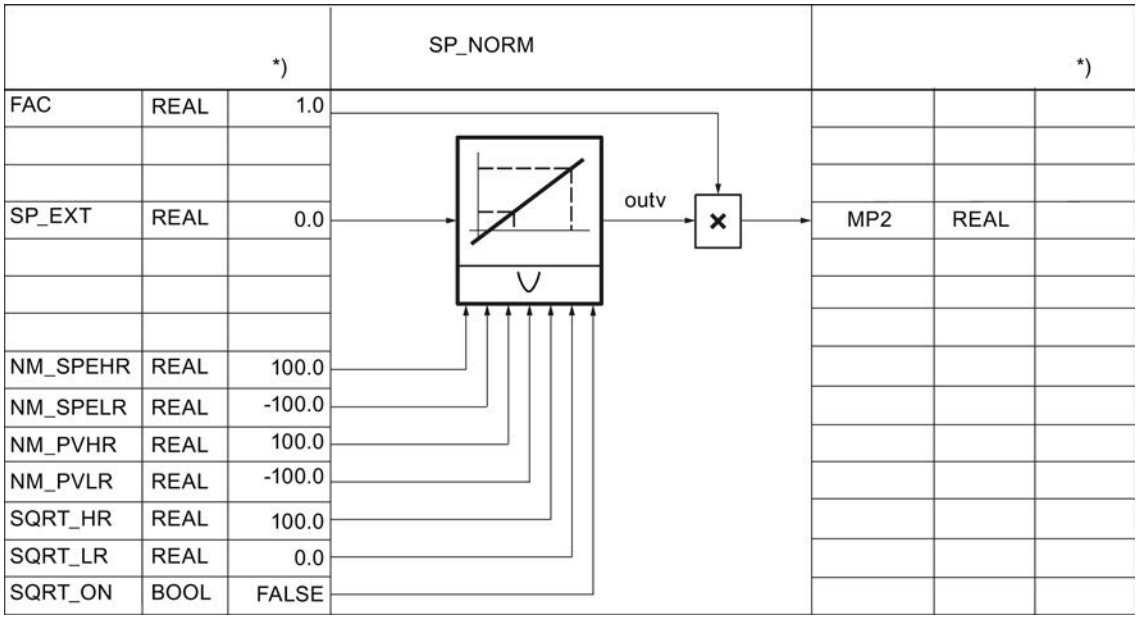
The setpoint can be specified with the following parameters via a ramp/soak function.

Ramp/soak (RMP_SOAK) (Page 9663)



The blending/ratio controller can scale the external setpoint SP_EXT using the following parameters.

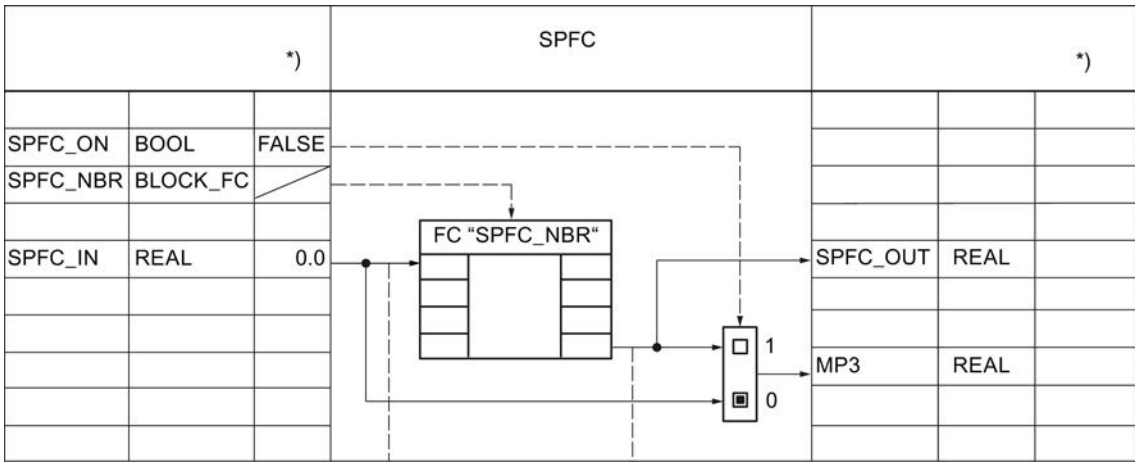
Scaling (Page 9669)



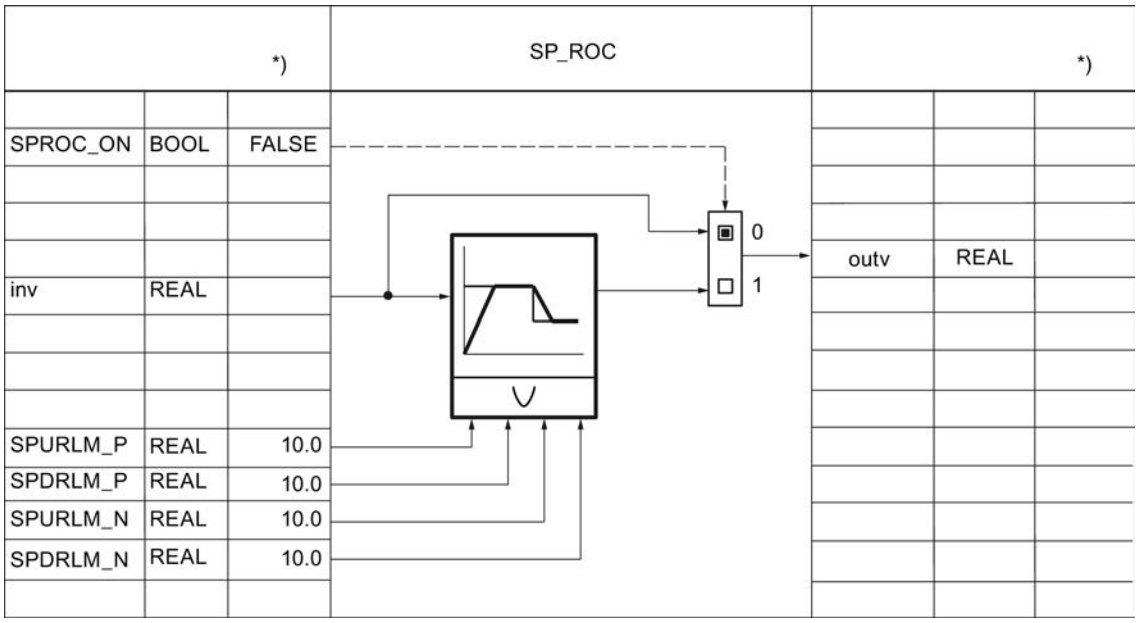
The setpoint can be further processed with a user-specific function. The following parameters are required for this: The interconnection must be programmed in the user FC.

The input value SPFC_IN is an implicit parameter. This can be monitored with the configuration tool, either via the measuring point MP1 (SPEXT_ON = FALSE) or via the measuring point MP2 (SPEXT_ON = TRUE). The output value is accessible at the measuring point MP3. The input SPFC_IN is then switched through on the setpoint branch, if SPFC_ON =FALSE.

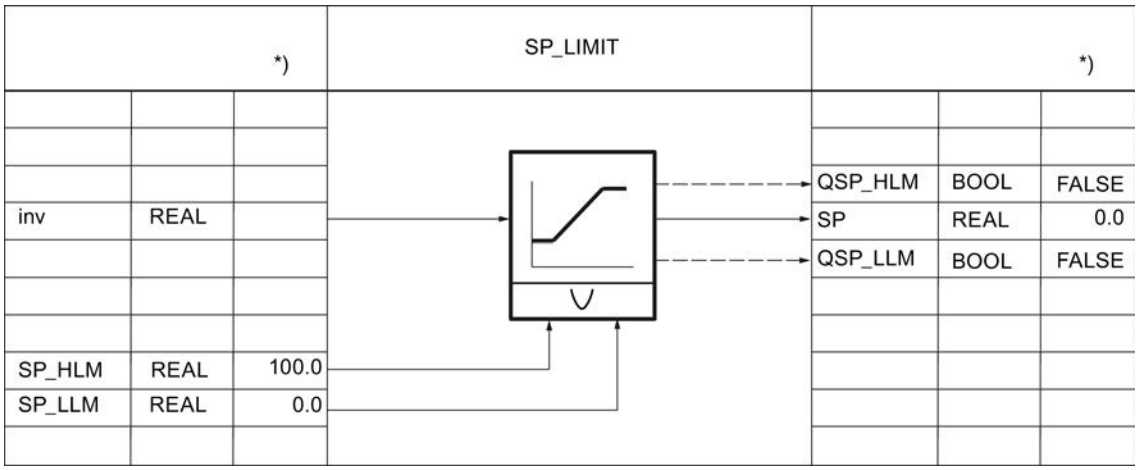
Limiting functions (Page 9670)



The rate of change of the setpoint can be limited with the following parameters.



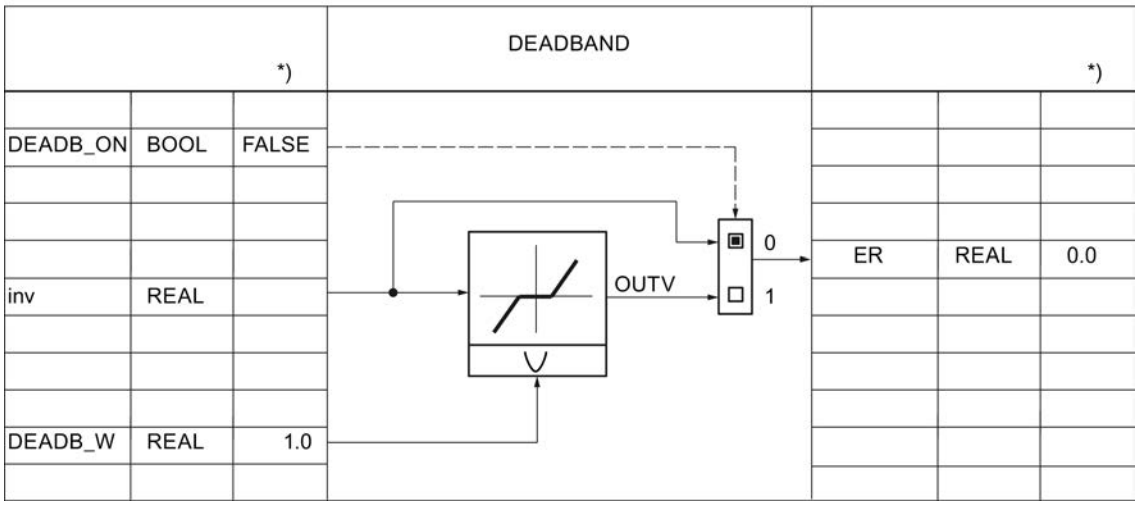
The setpoint value is always monitored with the following parameters for an absolute high and low limit.



Generating control deviation

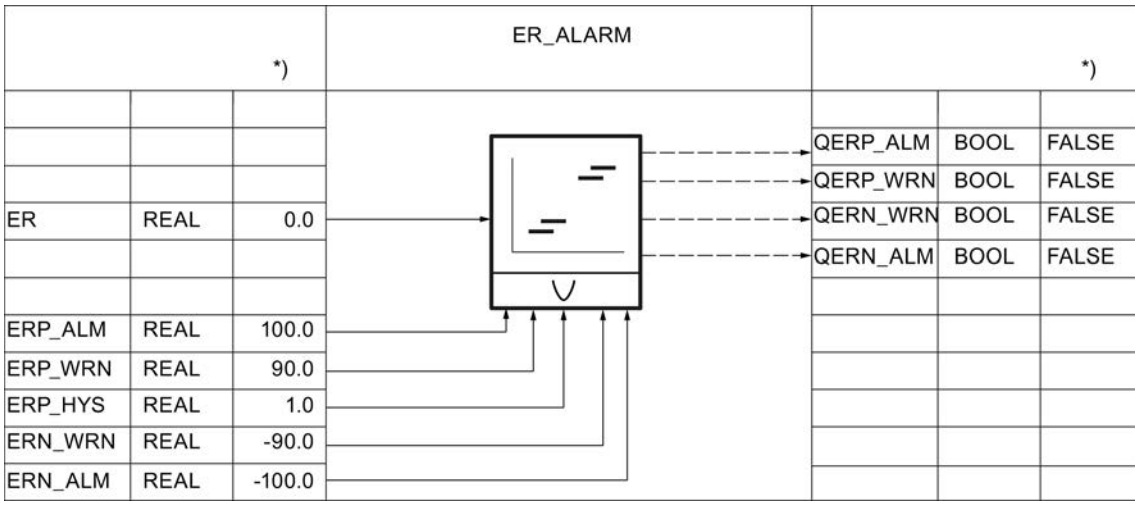
The following parameters can be used to route the control deviation over a deadband.

Deadband (Page 9673)



The control deviation is always monitored with the following parameters for warning and alarm limits.

Alarm (Page 9674)



Control algorithm PID_CIP

Control algorithm

Control algorithm (Page 9675)

A total of five switches are available to specify an effective controller structure.

Controller structures (Page 9677)

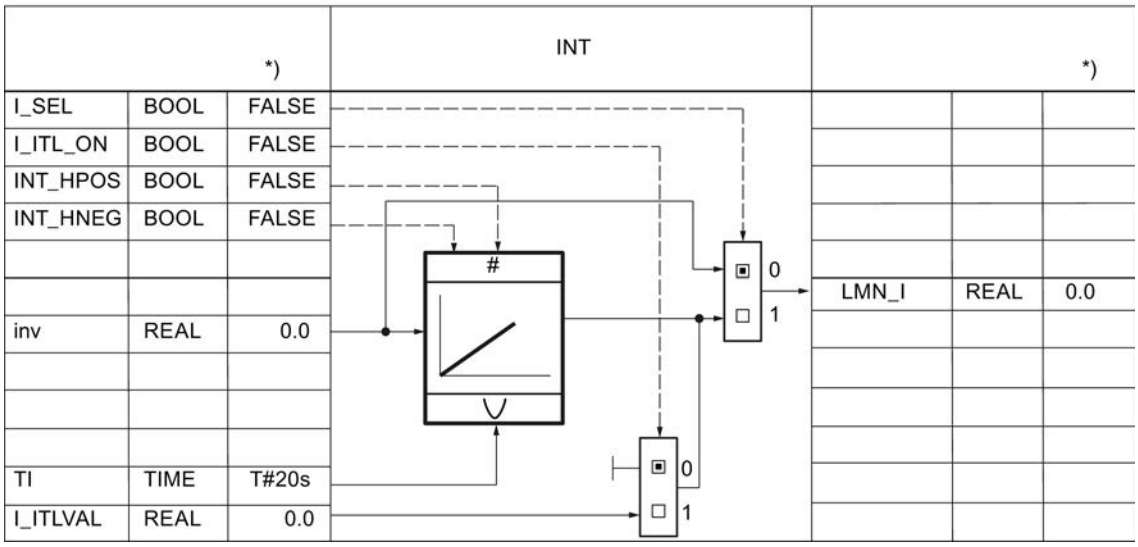
Controller structure	Switch				
	P_SEL	I_SEL	D_SEL	PFDB_SEL	DFDB_SEL
P control	TRUE	FALSE	FALSE	FALSE	FALSE
P control (P in feedback path)	TRUE	FALSE	FALSE	TRUE	FALSE
PI control	TRUE	TRUE	FALSE	FALSE	FALSE
PI control (P in feedback path)	TRUE	TRUE	FALSE	TRUE	FALSE
PD control	TRUE	FALSE	TRUE	FALSE	FALSE
PD control (P in feedback path)	TRUE	FALSE	TRUE	FALSE	TRUE
PID control	TRUE	TRUE	TRUE	FALSE	FALSE
PID control (P/D in feedback path)	TRUE	TRUE	TRUE	FALSE	TRUE

The proportional action is determined by the proportional gain GAIN and can be monitored at LMN_P.

Proportional component (Page 9681)

The integral action is calculated using the following parameters.

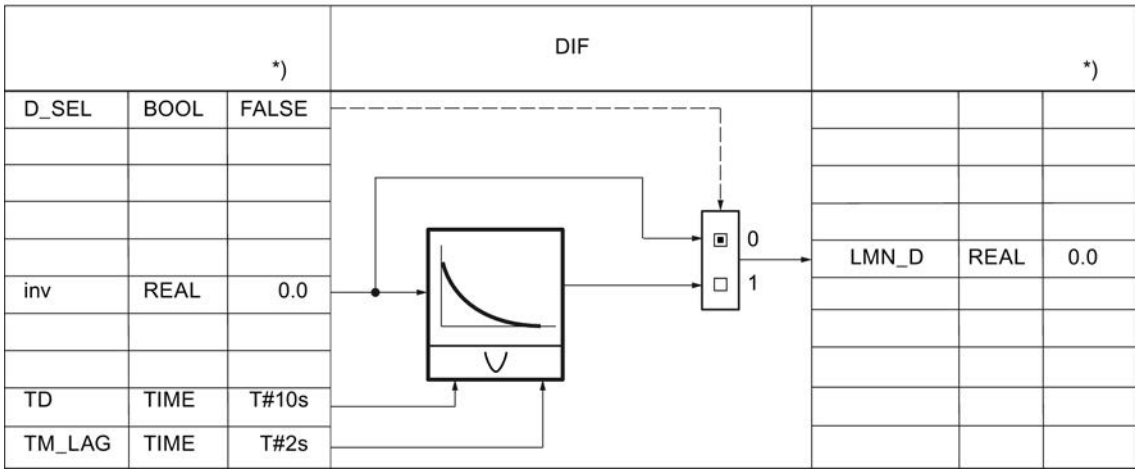
Integral action (Page 9681)



*) Default for creation of new technology object

The derivative action is calculated using the following parameters.

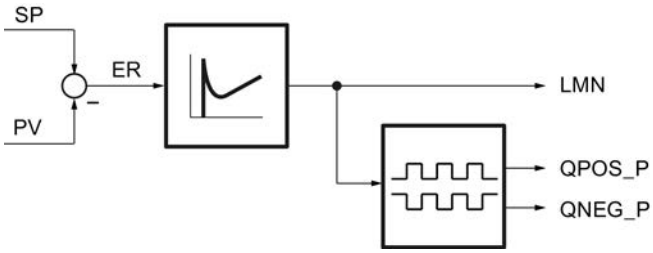
Derivative component (Page 9684)



Manipulated variable PID_CP

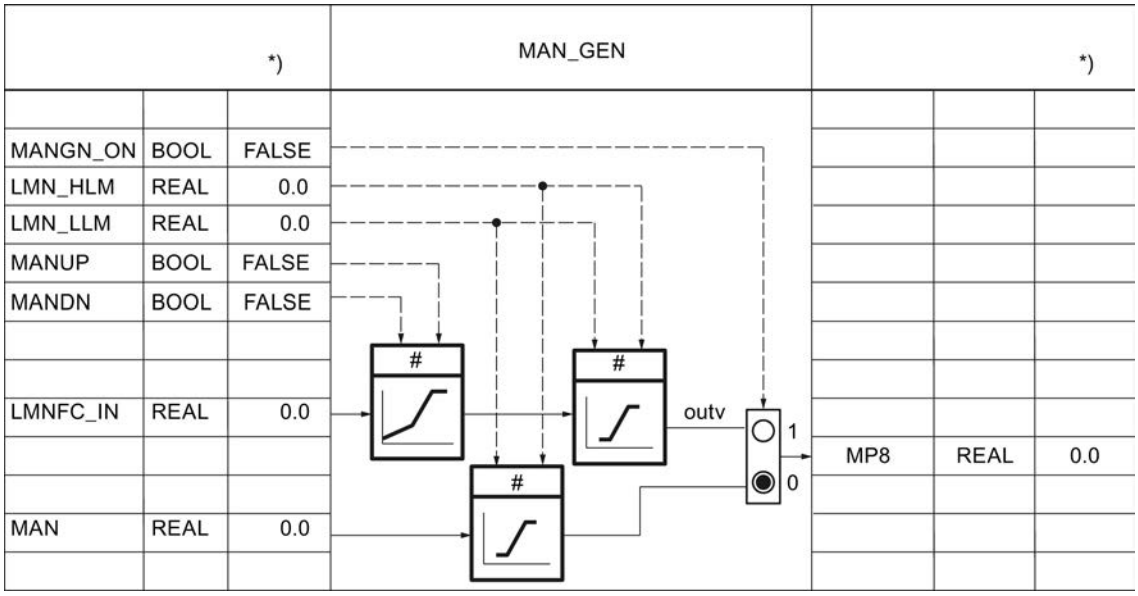
Continuous manipulated variable

The PID control algorithm generates an analog manipulated variable, supplemented if necessary by a pulse generator to generate pulse width modulated output signals for the two-step control or three-step controls with proportional actuators.



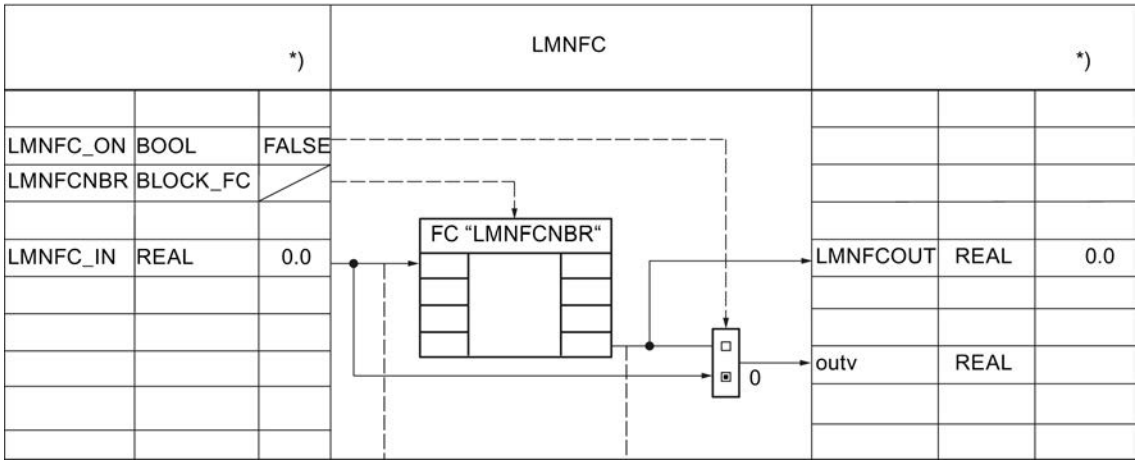
The manipulated variable can be manually influenced using the following parameters.

Manual mode (Page 9685)

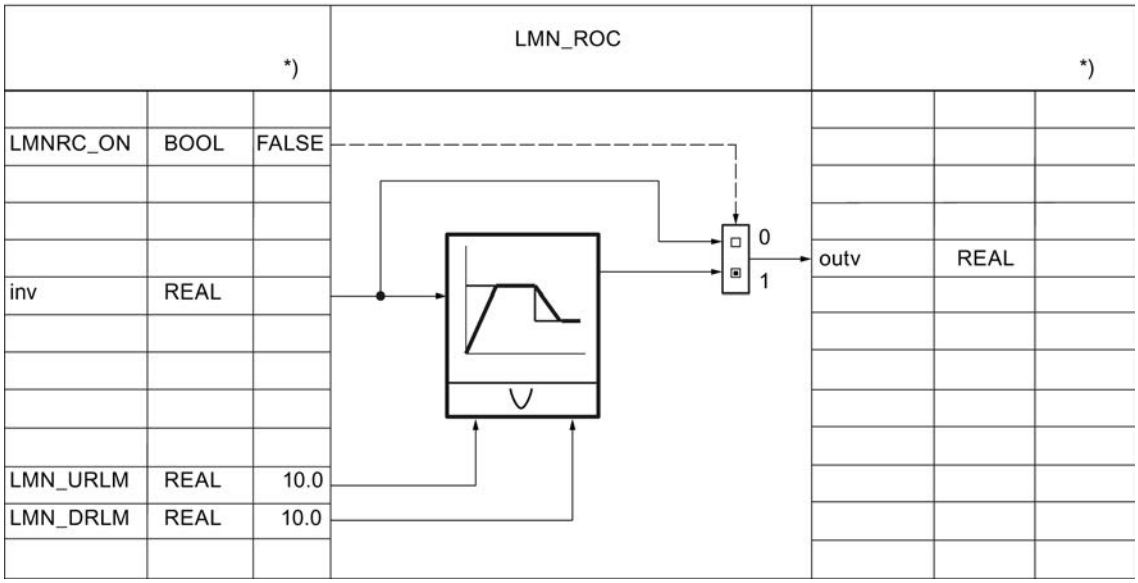


The manipulated variable can be further processed with a user-specific function. The following parameters are required for this: The interconnection must be programmed in the user FC.

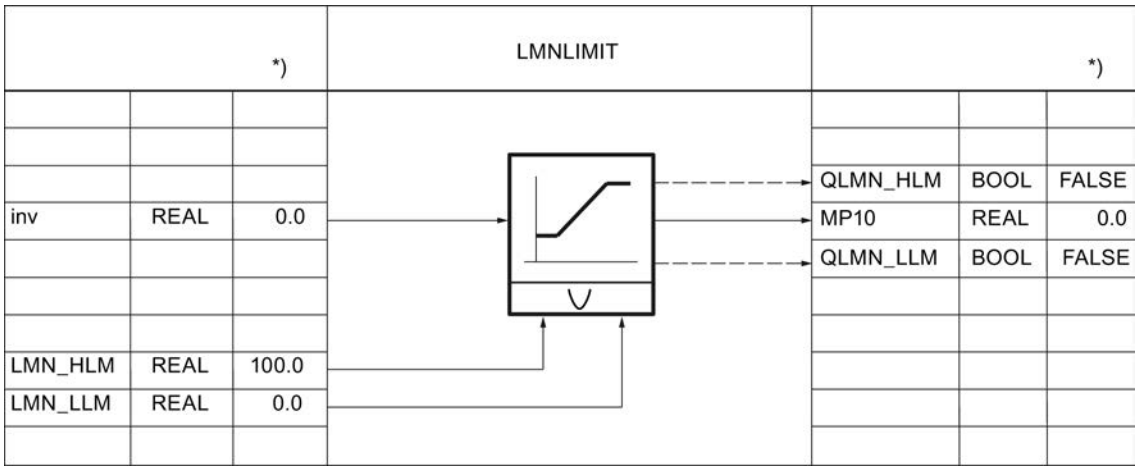
Limiting functions (Page 9688)



The rate of change of the manipulated variable can be limited using the following parameters.

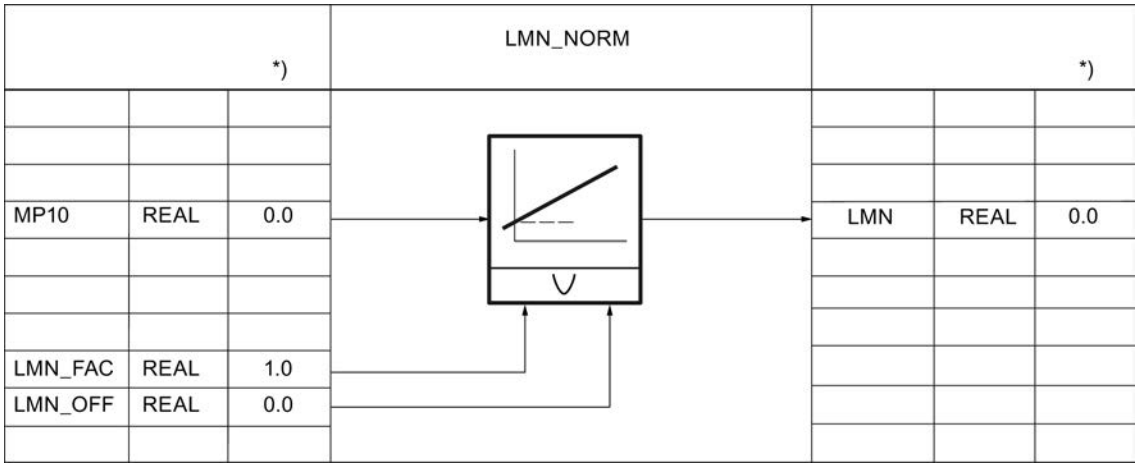


The manipulated variable is always monitored with the following parameters for an absolute high and low limit.



The manipulated variable is scaled to a physical value using the following parameters.

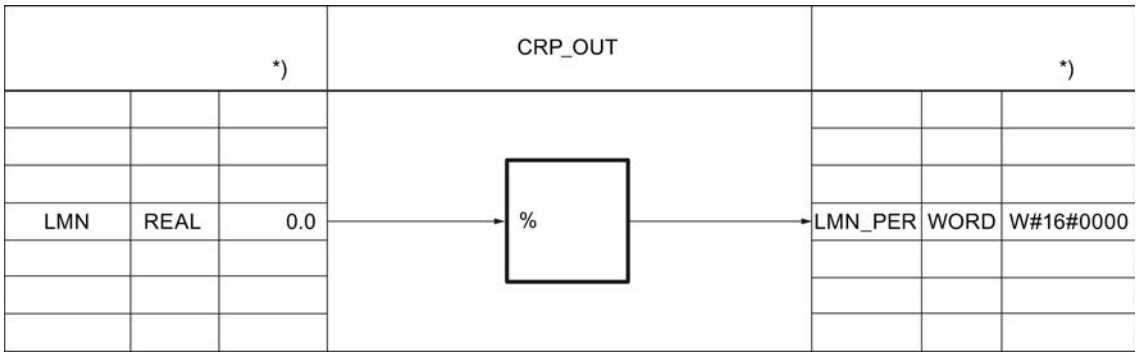
Scaling (Page 9690)



The manipulated variable in floating-point format is converted to I/O format. In this process, no check is carried out for positive/negative overflow or for reaching the overrange/underrange. Module types are not taken into account.

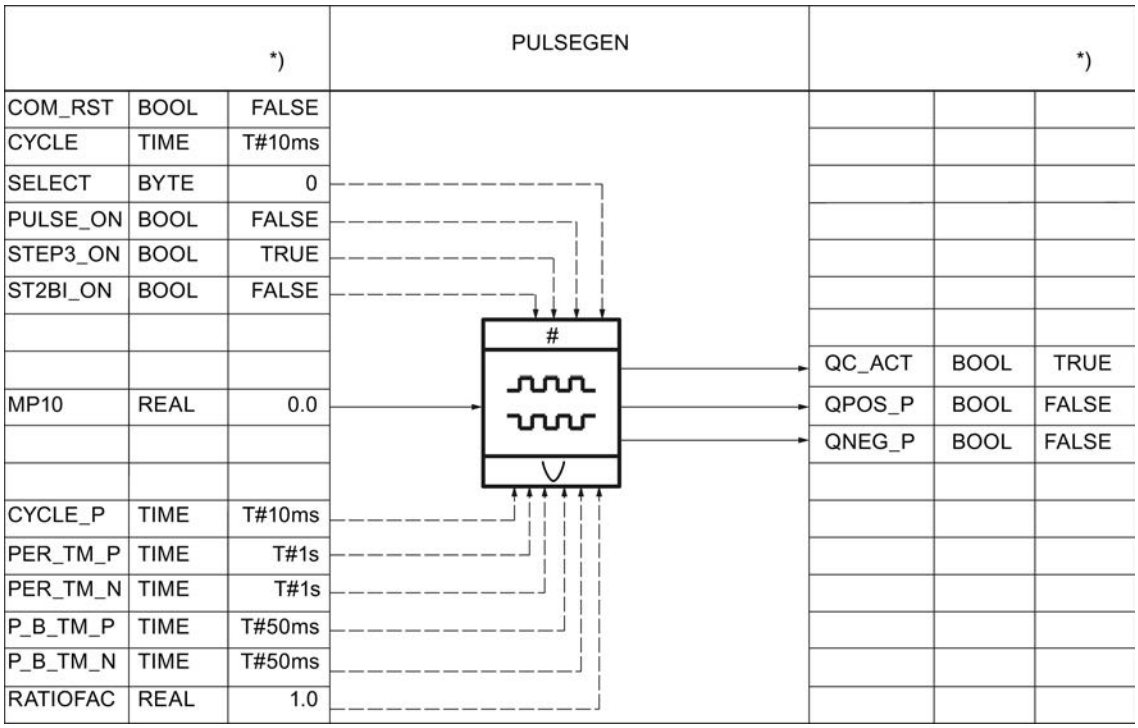
The following table provides an overview of ranges and numerical values before and after processing.

Manipulated value LMN in %	I/O value LMN_PER
118.515	32767
100.000	27648
0.003617	1
0.000	0
-0.003617	-1
-100.000	-27648
-118.519	-32768



The continuous manipulated variable can be output as pulse train via a pulse width modulation. The following parameters are required for this:

Pulse generator (Page 9692)



PID_CP in cascade arrangements

Interruption of the controller cascade

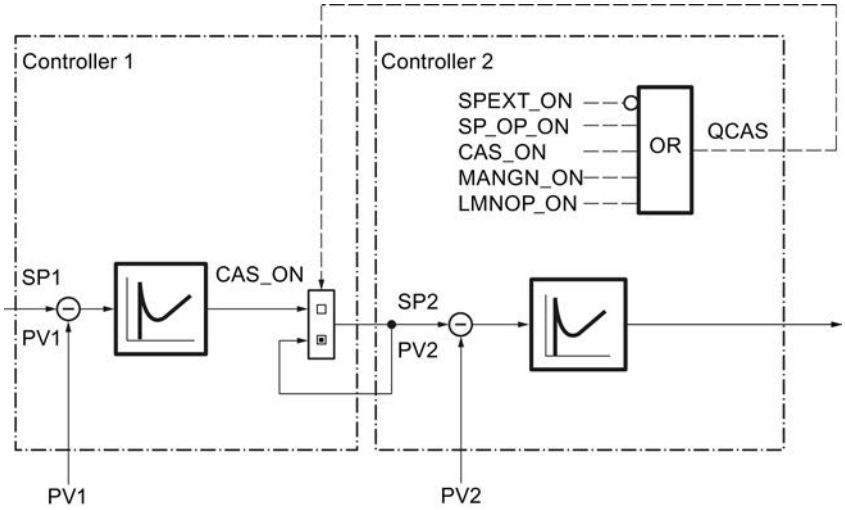
In a cascade, several controllers are in direct relationship with each other. Therefore, you must take precautions to ensure problem-free cascade operation following an interruption of the cascade structure at any point.

In cascade controls, for this reason, a QCAS signal is generated in the secondary controllers by applying OR logic from the status signals of the switches in the setpoint and manipulated variable branches. This signal actuates a switch in the higher-level controllers that converts

the controllers to follow-up mode. The follow-up value is always the setpoint of the lower-level circuit.

A smooth switchover occurs from follow-up mode to automatic mode, just as the switchover from manual to automatic mode.

PID_CP can be used in higher-level control loops as a master controller and in secondary control loops as a follow-up controller.

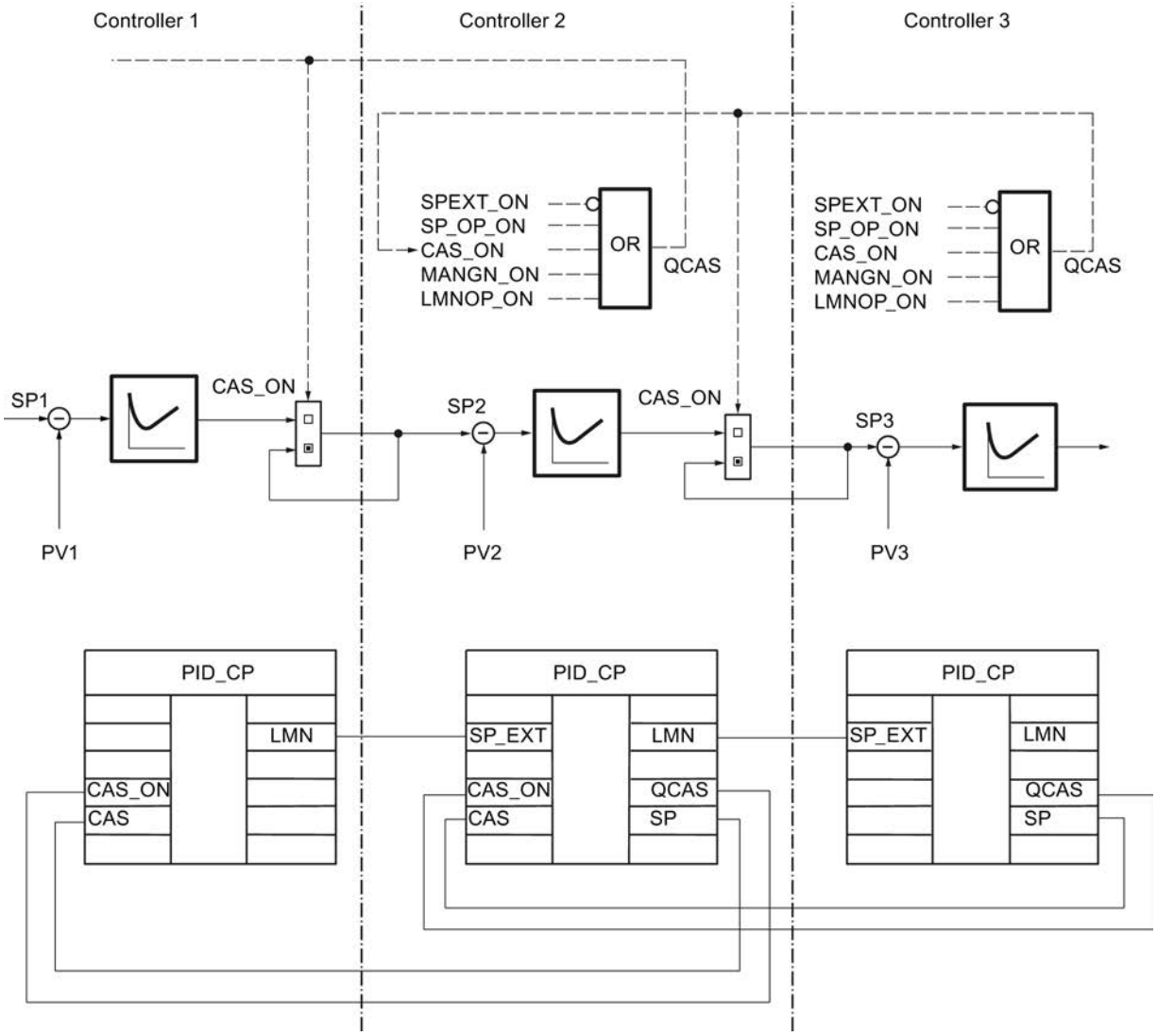


Note

The manipulated variable of the master controller LMN must always be interconnected with the external setpoint SP_EXT of the follow-up controller.

Block interconnection

The following figure shows the principle of the controller interconnection in multi-loop cascades.



Input parameters for PID_CP

Table 9-204

Parameter	Offset	Data type	Range of values	Preassigned value	Description
COM_RST	0.0	BOOL		FALSE	Restart (Initialization routine of the instruction will be processed)
I_SEL	0.1	BOOL		TRUE	Integral action on
D_SEL	0.2	BOOL		FALSE	Derivative action on

Parameter	Offset	Data type	Range of values	Preassigned value	Description
MAN_ON	0.3	BOOL		TRUE	Manual mode on (control loop interrupted, LMN set manually)
CAS_ON	0.4	BOOL		FALSE	Cascade mode ON (interconnected to QCAS of the lower-level controller)
SELECT	1.0	BYTE	0, 1, 2, 3	0	If PULS_ON = TRUE: 0: PID and pulse generator 1: PID (block call in OB 1) 2: Pulse generator (block call in cyclic interrupt OB) 3: PID (block call in cyclic interrupt OB)
CYCLE	2.0	TIME	> 20 ms (S7–300)	T#1s	Sampling time (time between two block calls = constant) Make sure to assign this parameter with the value of the cyclic interrupt cycle clock of the OB in which the PID_CP instruction runs. Otherwise, the time-dependent functions will not work correctly. (Exception: you are using time scaling, e.g., via the loop scheduler)
CYCLE_P	6.0	TIME		T#10 ms	Sampling time of the pulse generator Make sure to assign this parameter with the value of the cyclic interrupt cycle clock of the OB in which the PID_CP instruction runs. Otherwise, the time-dependent functions will not work correctly. (Exception: you are using time scaling, e.g., via the loop scheduler)
SP_INT	10.0	REAL	Engineering value range (physical quantity)	0.0	Internal setpoint SP_INT can be changed via a HMI with SPUP and SPDN
SP_EXT	14.0	REAL	Engineering value range (physical quantity)	0.0	External setpoint SP_EXT is used for blending, ratio and cascade controllers and can be scaled.
PV_IN	18.0	REAL	Engineering value range (physical quantity)	0.0	Process value input (PV in floating-point format)
PV_PER	22.0	INT		W#16#0000	Process value of I/O (PV in I/O format)
GAIN	24.0	REAL	Overall value range (dimensionless)	2.0	Proportional gain (= controller gain)
TI	28.0	TIME	TI ≥ CYCLE	T#20s	Integral action time
TD	32.0	TIME	TD ≥ CYCLE	T#10s	Derivative-action time
TM_LAG	36.0	TIME	TM_LAG ≥ CYCLE/2	T#2s	Time delay of derivative action

Parameter	Offset	Data type	Range of values	Preassigned value	Description
DISV	40.0	REAL		0.0	Disturbance variable Valid value range: -100.0 ... 100.0
CAS	44.0	REAL	Engineering value range (physical quantity)	0.0	Input for cascade mode (interconnected to PV of the lower-level controller)
SP_HLM	48.0	REAL	Engineering value range (physical quantity)	100.0	Setpoint high limit
SP_LLM	52.0	REAL	Engineering value range (physical quantity)	0.0	Setpoint low limit
LMN_HLM	56.0	REAL	LMN_LLM ... 100.0	100.0	Manipulated variable: High limit
LMN_LLM	60.0	REAL	-100.0 ... LMN_HLM	0.0	Manipulated variable: Low limit
DB_NBR	64.0	BLOCK_DB		DB 1	Data block number (DB with the time slices of the ramp/soak profile)
SPFC_NBR	66.0	BLOCK_FC		FC 0	Setpoint FC number (self-defined FC in setpoint branch)
PVFC_NBR	68.0	BLOCK_FC		FC 0	Process value FC number (self-defined FC in process value branch)
LMNFCNBR	70.0	BLOCK_FC		FC 0	Manipulated variable FC number (self-defined FC in manipulated variable branch)

Output parameters PID_CP

Table 9-205

Parameter	Offset	Data type	Preassigned value	Description
LMN	72.0	REAL	0.0	Manipulated variable (Manipulated variable in floating-point format)
LMN_PER	76.0	INT	W#16#0000	Manipulated variable for I/O (LMN in I/O format)
SP	78.0	REAL	0.0	Setpoint (effective setpoint)
PV	82.0	REAL	0.0	Process value (output of effective process value in cascade mode)
QCAS	86.0	BOOL	FALSE	Signal for cascade mode (interconnected to CAS_ON of higher-level controller)
QC_ACT	86.1	BOOL	TRUE	Indicates whether the control functions will be processed at the next block call (only relevant if SELECT has the value 0 or 1)

Parameter	Offset	Data type	Preassigned value	Description
QPOS_P	86.2	BOOL	FALSE	Pulse generator: Positive pulse ON
QNEG_P	86.3	BOOL	FALSE	Pulse generator: Negative pulse ON

In/out parameters PID_CP

Table 9-206

Parameter	Offset	Data type	Preassigned value	Description
MAN	88.0	REAL	0.0	Manual manipulated variable (for manipulated variable assignment by means of operator control and monitoring functions)

Static variables PID_CP

Parameter	Offset	Data type	Range of values	Preassigned value	Description
PVH_ALM	92.0	REAL	PVH_WRN...100.0	100.0	Process value: High limit 'Alarm'
PVH_WRN	96.0	REAL	PVL_WRN... PVH_ALM	90.0	Process value: High limit 'Warning'
PVL_WRN	100.0	REAL	PVL_ALM... PVH_WRN	-90.0	Process value: Low limit 'Warning'
PVL_ALM	104.0	REAL	-100.0...PVL_WRN	-100.0	Process value: Low limit 'Alarm'
SPGEN_ON	108.0	BOOL		FALSE	Setpoint manipulation ON (adjusting the setpoint using the up/down switch)
SPUP	108.1	BOOL		FALSE	Setpoint UP
SPDN	108.2	BOOL		FALSE	Setpoint DOWN
RMPSK_ON	108.3	BOOL		FALSE	Ramp/soak function ON (setpoint is specified as a ramp/soak profile)
SPEXT_ON	108.4	BOOL		FALSE	External setpoint ON (for interconnection with other controller blocks)
MANGN_ON	108.5	BOOL		FALSE	Manual mode ON (LMN set via switch)
MANUP	108.6	BOOL		FALSE	Manual output value UP
MANDN	108.7	BOOL		FALSE	Manual output value DOWN
DFRMP_ON	109.0	BOOL		FALSE	Preassign output of ramp/soak function (output is preassigned with SP_INT)
CYC_ON	109.1	BOOL		FALSE	Cyclic repetition ON (ramp/soak profile will be repeated automatically)

Parameter	Offset	Data type	Range of values	Preassigned value	Description
RMP_HOLD	109.2	BOOL		FALSE	Ramp/soak function HOLD (output of ramp/soak function is frozen)
CONT_ON	109.3	BOOL		FALSE	Continue (with schedule) (ramp/soak profile will be continued at the next time slice)
TUPDT_ON	109.4	BOOL		FALSE	Recalculate total time (total time of ramp/soak profile will be recalculated)
SPFC_ON	109.5	BOOL		FALSE	Call setpoint FC
SPROC_ON	109.6	BOOL		FALSE	Ramp function ON (rate of change limiting of SP)
PVPER_ON	109.7	BOOL		FALSE	Process value of I/O ON (interconnection to I/O modules)
LAG1STON	110.0	BOOL		FALSE	First-order delay element ON
SQRT_ON	110.1	BOOL		FALSE	Square root function ON
PVFC_ON	110.2	BOOL		FALSE	Call process value FC
DEADB_ON	110.3	BOOL		FALSE	Deadband ON (small disturbances and noise will be filtered)
P_SEL	110.4	BOOL		TRUE	P component ON
PFDB_SEL	110.5	BOOL		FALSE	Switch P component to feedback
INT_HPOS	110.6	BOOL		FALSE	Freeze I component in positive direction
INT_HNEG	110.7	BOOL		FALSE	Freeze I component in negative direction
I_ITL_ON	111.0	BOOL		FALSE	Preassign I component
DFDB_SEL	111.1	BOOL		FALSE	Switch D component to feedback
DISV_SEL	111.2	BOOL		FALSE	Apply disturbance variable
LMNFC_ON	111.3	BOOL		FALSE	Call manipulated variable FC
LMNRC_ON	111.4	BOOL		FALSE	Activate manipulated variable ramp function (rate of change limiting of LMN)
SMOO_CHG	111.5	BOOL		TRUE	Smooth changeover from manual to automatic
PULSE_ON	111.6	BOOL		FALSE	Pulse generator ON
STEP3_ON	111.7	BOOL		TRUE	Pulse generator: Three-step control ON
ST2BI_ON	112.0	BOOL		FALSE	Pulse generator: two-step control for bipolar control range ON (for unipolar control range, STEP3_ON = FALSE must be set)
TM_SNBR	114.0	INT	≥ 0 (dimensionless)	0	No. of next time slice for continuation
TM_CONT	116.0	TIME	Overall value range (dimensionless)	T#0s	Continuation point (Time after time slice TM_SNBR at which the ramp/soak function continues processing the ramp/soak profile)
FAC	120.0	REAL	Overall value range (dimensionless)	1.0	Factor (ratio or blending factor)
NM_SPEHR	124.0	REAL		100.0	Setpoint scaling: Operating range input high

Parameter	Offset	Data type	Range of values	Preassigned value	Description
NM_SPELR	128.0	REAL		-100.0	Setpoint scaling: Operating range input low
SPFC_OUT	132.0	REAL	-100.0 ... 100.0	0.0	Setpoint FC output (interconnected to the output of the FC in the setpoint branch)
SPURLM_P	136.0	REAL	≥ 0 [physical quantity/s]	10.0	Setpoint up rate limit in positive range
SPDRLM_P	140.0	REAL	≥ 0 [physical quantity/s]	10.0	Setpoint down rate limit in positive range
SPURLM_N	144.0	REAL	≥ 0 [physical quantity/s]	10.0	Setpoint up rate limit in negative range
SPDRLM_N	148.0	REAL	≥ 0 [physical quantity/s]	10.0	Setpoint down rate limit in negative range
NM_PIHR	152.0	REAL		100.0	Process value scaling: Measuring range input high
NM_PILR	156.0	REAL		-100.0	Process value scaling: Measuring range input low
NM_PVHR	160.0	REAL		100.0	Process value scaling: Measuring range output high
NM_PVLR	164.0	REAL		-100.0	Process value scaling: Measuring range output low
PV_TMLAG	168.0	TIME	Overall value range	T#5s	Process value time delay (delay of PT1 element in the PV branch)
SQRT_HR	172.0	REAL		100.0	Square root: Working range output high
SQRT_LR	176.0	REAL		0.0	Square root: Working range output low
PVFC_OUT	180.0	REAL	-100.0 ... 100.0	0.0	Process value FC output (interconnected to the output of the FC in the process value branch)
PVURLM_P	184.0	REAL	≥ 0 [physical quantity/s]	10.0	Process value up rate limit in the positive range
PVDRLM_P	188.0	REAL	≥ 0 [physical quantity/s]	10.0	Process value down rate limit in the positive range
PVURLM_N	192.0	REAL	≥ 0 [physical quantity/s]	10.0	Process value up rate limit in the negative range
PVDRLM_N	196.0	REAL	≥ 0 [physical quantity/s]	10.0	Process value down rate limit in the negative range
PV_HYS	200.0	REAL	≥ 0	1.0	Process value hysteresis (prevents 'flickering' of monitoring display)
DEADB_W	204.0	REAL	0.0 ... 100.0	1.0	Deadband width (= range from zero to deadband high limit) (determines the size of the deadband area)
ERP_ALM	208.0	REAL	0 ... 200.0	100.0	Control deviation: Positive limit 'Alarm'
ERP_WRN	212.0	REAL	0 ... 200.0	90.0	Control deviation: Positive limit 'Warning'
ERN_WRN	216.0	REAL	-200.0 ... 0	-90.0	Control deviation: Negative limit 'Warning'
ERN_ALM	220.0	REAL	-200.0 ... 0	-100.0	Control deviation: Negative limit 'Alarm'

Parameter	Offset	Data type	Range of values	Preassigned value	Description
ER_HYS	224.0	REAL	≥ 0 [%]	1.0	Control deviation hysteresis (prevents 'flickering' of monitoring display)
I_ITLVAL	228.0	REAL	-100.0 ... 100.0 [%]	0.0	Initialization value for I action
LMNFCOUT	232.0	REAL	-100.0 ... 100.0 [%]	0.0	Manipulated variable FC output (interconnected to the output of the FC in the manipulated variable branch)
LMN_URLM	236.0	REAL	≥ 0 [%/s]	10.0	Manipulated variable up rate limit
LMN_DRLM	240.0	REAL	≥ 0 [%/s]	10.0	Manipulated variable down rate limit
LMN_FAC	244.0	REAL	Overall value range (dimensionless)	1.0	Manipulated variable factor (factor for adaptation of the manipulated variable range)
LMN_OFF	248.0	REAL	Overall value range (dimensionless)	0.0	Manipulated variable offset (zero point of manipulated variable scaling)
PER_TM_P	252.0	TIME		T#1s	Pulse generator: Period of positive pulse
PER_TM_N	256.0	TIME		T#1s	Pulse generator: Period of negative pulse
P_B_TM_P	260.0	TIME		T#50ms	Pulse generator: Minimum pulse time or minimum break time of positive pulse
P_B_TM_N	264.0	TIME		T#50ms	Pulse generator: Minimum pulse time or minimum break time of negative pulse
RATIOFAC	268.0	REAL	0.1 ... 10.0 (dimensionless)	1.0	Pulse generator: Ratio factor (ratio of positive pulse time and negative pulse time)
PHASE	272.0	INT		0	Phase of PID self tuner
QPVH_ALM	274.0	BOOL		FALSE	Process value: High limit 'Alarm' triggered
QPVH_WRN	274.1	BOOL		FALSE	Process value: High limit 'Warning' triggered
QPVL_WRN	274.2	BOOL		FALSE	Process value: Low limit 'Warning' triggered
QPVL_ALM	274.3	BOOL		FALSE	Process value: Low limit 'Alarm' triggered
QR_S_ACT	274.4	BOOL		FALSE	Time table for ramp/soak profile will be processed
QSP_HLM	274.5	BOOL		FALSE	Setpoint: High limit triggered
QSP_LLM	274.6	BOOL		FALSE	Setpoint: Low limit triggered
QPVURLMP	274.7	BOOL		FALSE	Process value: Up rate limit in positive range triggered
QPVDRLMP	275.0	BOOL		FALSE	Process value: Down rate limit in positive range triggered
QPVURLMN	275.1	BOOL		FALSE	Process value: Up rate limit in negative range triggered
QPVDRLMN	275.2	BOOL		FALSE	Process value: Down rate limit in negative range triggered
QERP_ALM	275.3	BOOL		FALSE	Control deviation: Positive limit 'Alarm' triggered
QERP_WRN	275.4	BOOL		FALSE	Control deviation: Positive limit 'Warning' triggered
QERN_WRN	275.5	BOOL		FALSE	Control deviation: Negative limit 'Warning' triggered

Parameter	Offset	Data type	Range of values	Preassigned value	Description
QERN_ALM	275.6	BOOL		FALSE	Control deviation: Negative limit 'Alarm' triggered
QLMN_HLM	275.7	BOOL		FALSE	Manipulated variable: High limit triggered
QLMN_LLM	276.0	BOOL		FALSE	Manipulated variable: Low limit triggered
NBR_ATMS	278.0	INT		0	Number of the time slice currently being approached by the ramp/soak function
RS_TM	280.0	TIME		T#0s	"Current residual time" of ramp/soak profile until the next time slice
T_TM	284.0	TIME		T#0s	"Total time" of ramp/soak profile
RT_TM	288.0	TIME		T#0s	"Total residual time" = time remaining until the end of the ramp/soak profile
ER	292.0	REAL		0.0	Control deviation
LMN_P	296.0	REAL		0.0	P component
LMN_I	300.0	REAL		0.0	I component
LMN_D	304.0	REAL		0.0	D component
SPFC_IN	308.0	REAL		0.0	Setpoint FC input (interconnected to input of the self-defined FC)
PVFC_IN	312.0	REAL		0.0	Process value FC input (interconnected to input of the self-defined FC)
LMNFC_IN	316.0	REAL		0.0	Manipulated variable FC input (interconnected to input of the self-defined FC)
SP_OP_ON	320.0	BOOL		FALSE	Setpoint manipulation ON (value of SP_OP will be applied as the setpoint)
PV_OP_ON	320.1	BOOL		FALSE	Process value manipulation ON (value of PV_OP will be applied as the setpoint)
LMNOP_ON	320.2	BOOL		FALSE	Manipulated variable manipulation ON (value of LMN_OP will be applied as the setpoint)
SP_OP	322.0	REAL		0.0	Setpoint manipulation during commissioning
PV_OP	326.0	REAL		0.0	Process value manipulation during commissioning
LMN_OP	330.0	REAL		0.0	Manipulated variable manipulation during commissioning
MP1	334.0	REAL		0.0	Measuring point 1: Internal setpoint
MP2	338.0	REAL		0.0	Measuring point 2: External setpoint
MP3	342.0	REAL		0.0	Measuring point 3: Unlimited setpoint
MP4	346.0	REAL		0.0	Measuring point 4: Process value of the I/O module
MP5	350.0	REAL		0.0	Measuring point 5: Process value after first-order delay element
MP6	354.0	REAL		0.0	Measuring point 6: Effective process value (PV)
MP7	358.0	REAL		0.0	Measuring point 7: Manipulated variable of PID algorithm
MP8	362.0	REAL		0.0	Measuring point 8: Manual output value

Parameter	Offset	Data type	Range of values	Preassigned value	Description
MP9	366.0	REAL		0.0	Measuring point 9: Unlimited manipulated variable
MP10	370.0	REAL		0.0	Measuring point 10: Limited manipulated variable

Block diagrams for PID_CP

Signal-flow representations

Analogous to the representation of the switches in the configuration tool, the black dot in each switch symbol indicates that the switching signal has the adjacent Boolean value (0=FALSE or 1=TRUE) and that the signal path is switched through this point in each case. The switching signals (binary signals) are identified by dashed lines.

This means that the setpoint is specified as an absolute setpoint via SP_INT; the same applies to the process value input via PV_IN. The default control function is a standard PI controller with P function in the forward branch. The control loop is open and the percentage range of the manipulated variable is influenced by the MAN input. All other functions are passive or, if they cannot be deactivated, marginal signal characteristic parameters are preassigned within the measuring range or operating range so that the functions have no effect.

Symbols and identifiers in the signal flow graphs

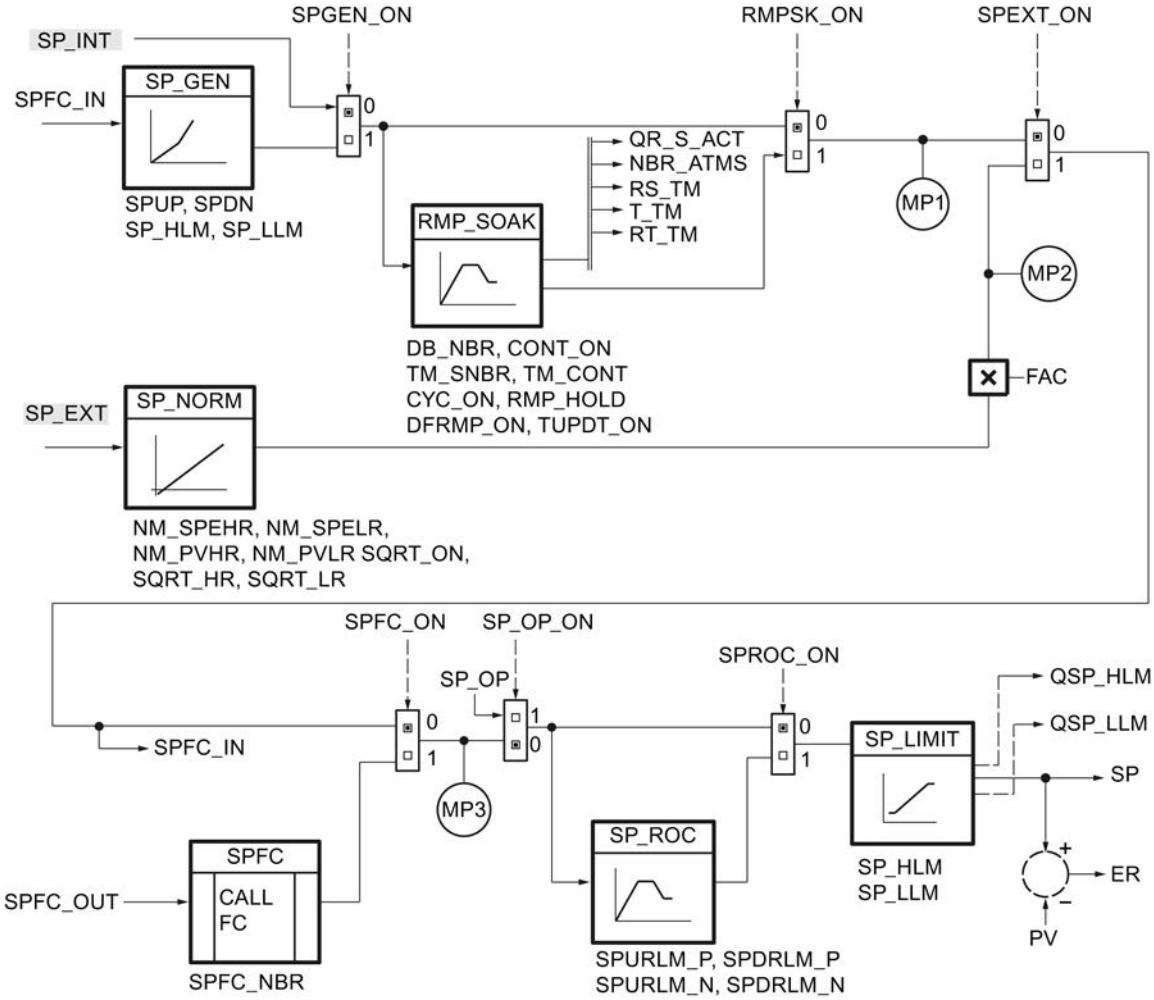
The names of the connectable process variables are shown on a shaded background. This allows you to recognize where the controller structure can be connected to the S7 I/O or directly to the measuring and actuators of the process.

Parameter names containing the combination of letters "OP" (for example, SP_OP/ SP_OP_ON) indicate that an intervention using the Standard PID Control configuration tool is possible at this point. The configuration tool has its own interface to the controller FB.

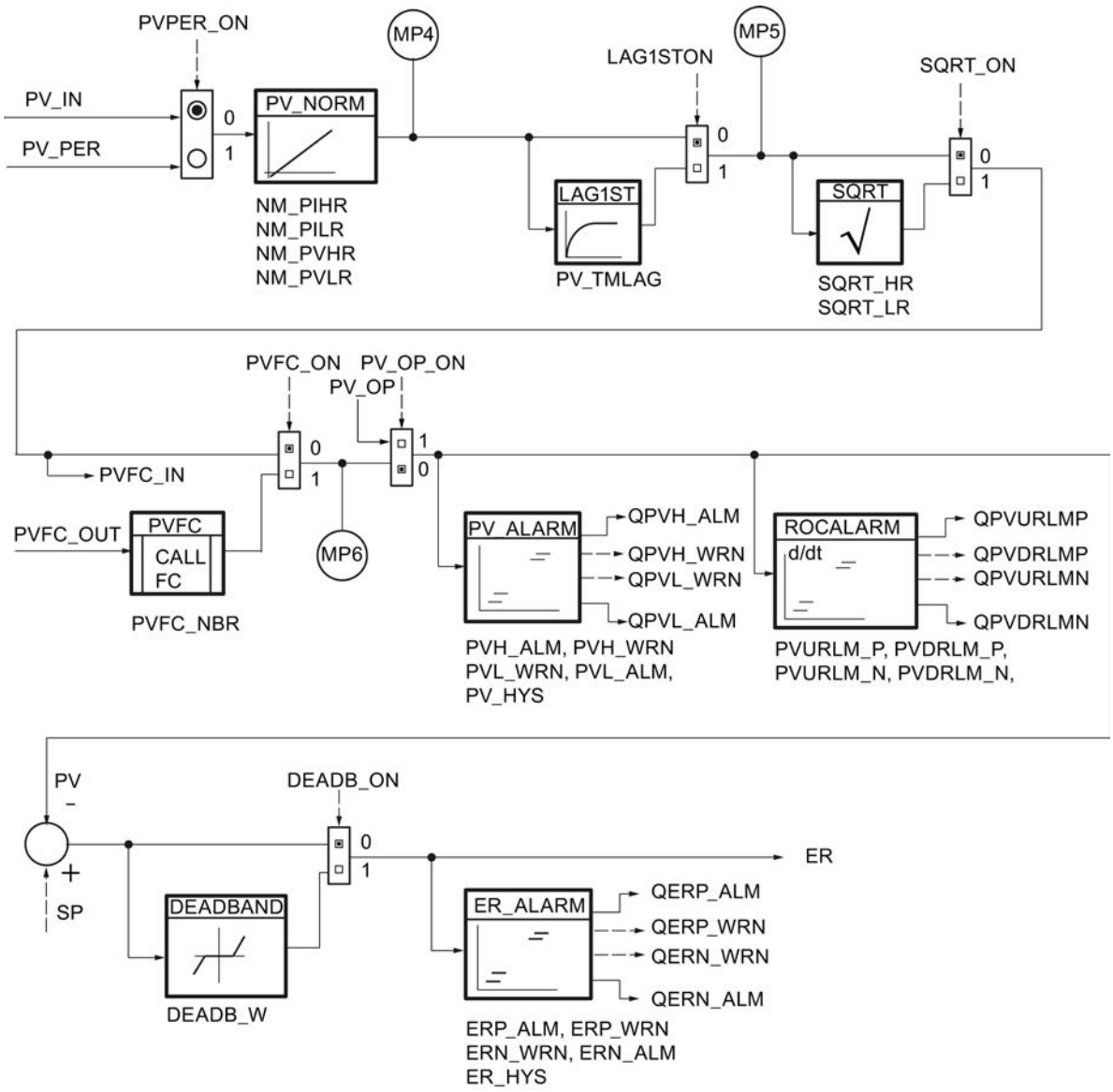
Intermediate variables in the signal characteristic can be monitored at the measuring points MP1 to MP12 marked with circles. These intermediate variables are required in order to match values before triggering "smooth" changeovers or to check the current statuses of the relevant control. The measuring point variables in the configuration tool curve recorder can be represented (statically and dynamically).

For reasons of clarity, the parameters for setting and sizing the processing operations (algorithms) are also indicated in the individual function blocks. Please refer to the descriptions in the reference section and to the representation of the individual subfunctions in the following sections.

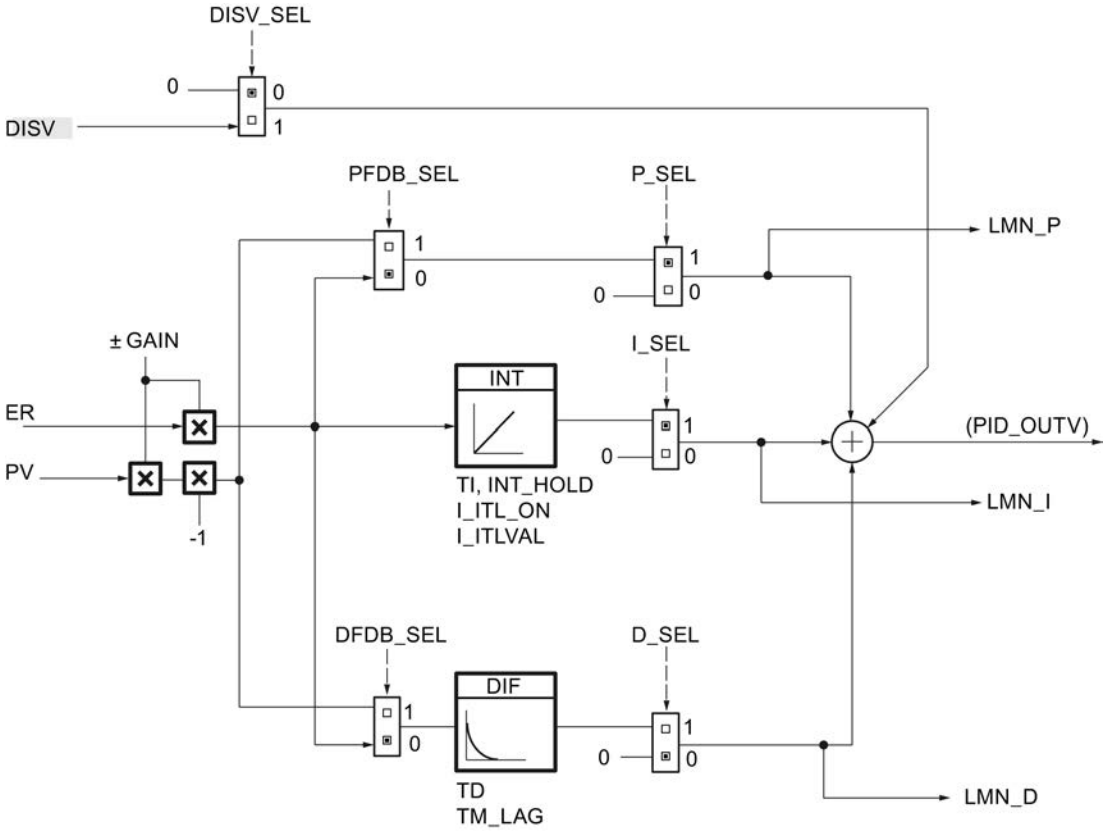
Processing the setpoint



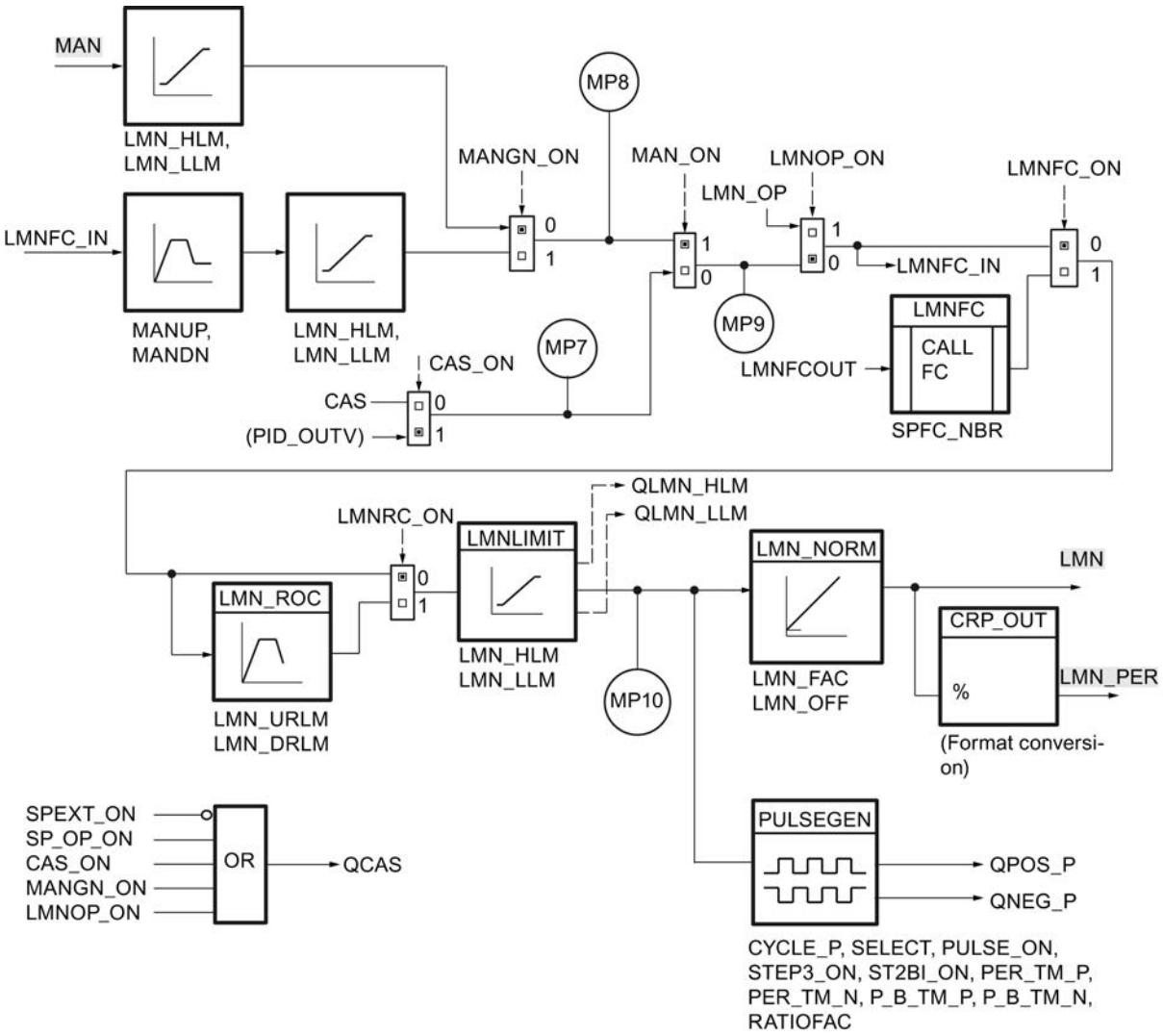
Processing the process value



Control algorithm



Processing the analog manipulated variable



Depending on the combination of the switching states, the OR gate generates an enable signal for the cascade coupling.

See also

Ramp/soak (RMP_SOAK) (Page 9663)

Data block RMP_SOAK

Parameter	Data type	Preassigned value	Description
NBR_PTS	INT	4	Number of time slices -1 1 - 255
PI	Array [0...NBR_PTS] of Struct		Structure for the time slice data. The array must be declared by 0...NBR_PTS.
PI[0].OUTV	REAL	0.0	Output variable [0] Starting point Engineering value range
PI[0].TMV	TIME	T#1s	Time value [0] Starting point
PI[1].OUTV	REAL	0.0	Output variable [1] Time slice 1 Engineering value range
PI[1].TMV	TIME	T#1s	Time value [1] Time slice 1
PI[2].OUTV	REAL	0.0	Output variable [2] Time slice 2 Engineering value range
PI[2].TMV	TIME	T#1s	Time value [2] Time slice 2
PI[3].OUTV	REAL	0.0	Output variable [3] Time slice 3 Engineering value range
PI[3].TMV	TIME	T#1s	Time value [3] Time slice 3
PI[4].OUTV	REAL	0.0	Output variable [4] Time slice 4 Engineering value range
PI[4].TMV	TIME	T#0s	Time value [4] Time slice 4 0 ms

PID_ES

Description PID_ES

In addition to the functions in the setpoint and process value branches, the instruction implements a complete PID controller with binary manipulated variable output. It is possible to influence the manipulated variable manually. Subfunctions can be enabled or disabled.

The instruction enables technical processes and systems with integral-action actuators to be controlled on SIMATIC S7 automation systems. Slow (temperatures, filling levels etc.) as well as fast controlling systems (flow, speed etc.) can be controlled.

The controller can be used individually as a fixed setpoint controller, or in secondary control loops in cascade, blending or ratio controls; however it cannot be used as the primary controller.

Signal processing in the setpoint/process value branch as well as handling and monitoring of the control deviation is identical to that of the continuous controller.

Startup

The PID_ES instruction features an initialization routine that is executed when input parameter COM_RST = TRUE is set.

The integral action is set to the initialization value I_ITLVAL at startup and read out at output LMN_I. When it is called in a cyclic interrupt level, it then continues to work starting at this value.

When ramp/soak is enabled, the time intervals DB_NBR PI[0 ... NBR_PTS].TMV between the time slices are added up and displayed at the total time T_TM and total remaining time RT_TM outputs. In the case of online changes of PI[n].TMV or specification of TM_CONT and TM_SNBR, the total time and the total remaining time of the ramp/soak profile change. Since the calculation of T_TM and RS_TM with many time slices significantly increases the processing time of the RMP_SOAK function, the calculation is only carried out at a restart or when TUPDT_ON = TRUE.

All other outputs are set to their default values.

Hot restart

With a hot restart, the operating state that existed at the time of the interruption is assumed. The control continues to operate with the values that were calculated at the moment the interruption occurred.

Call

The instruction PID_ES must be called with a constant bus cycle time. To achieve this, use a cyclic interrupt level (for example, OB35).

If you call the instruction PID_ES as a multiple instance DB, no technology object is created. No parameter assignment interface or commissioning interface is available. You must assign parameters for PID_ES directly in the multiple instance DB and commission it via a watch table.

Control deviation generation for PID_ES

The following functions are always active and cannot be disabled.

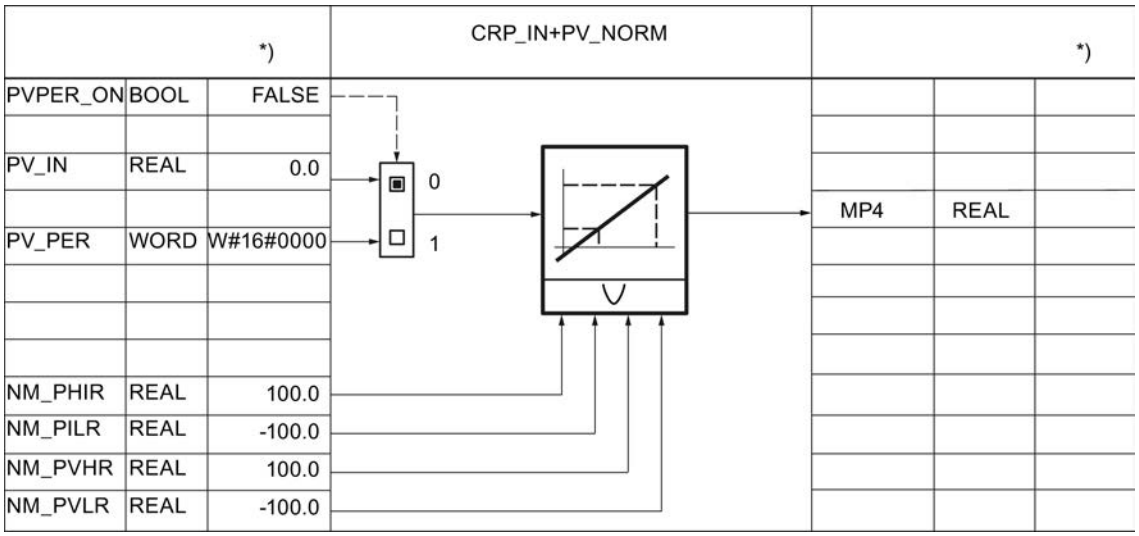
- Setpoint limiting SP_LIMIT,
- Process value monitoring for absolute values PV_ALARM,
- Process value monitoring for size of rate of change ROCALARM,
- Control deviation monitoring for absolute values ER_ALARM,
- Limiting of the output signal LMNLIMIT.

The default values are selected in such a manner that operation requires no individual adjustment. To achieve good controller results, you always have to adapt the parameters of these functions to your process.

Processing the process value

The process value is scaled to the physical value.

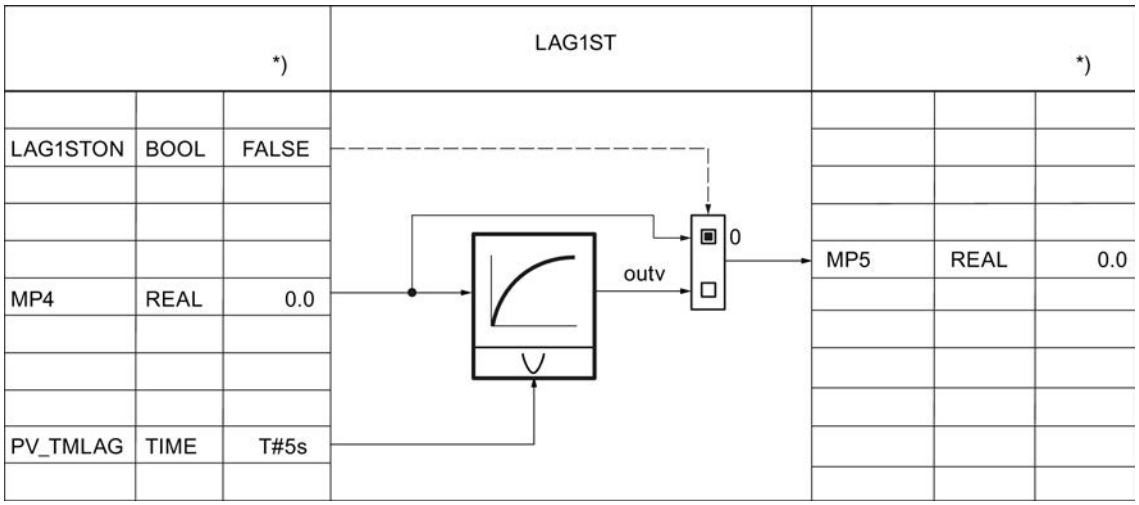
Scaling (PV_NORM) (Page 9704)



*) Default for creation of new technology object.

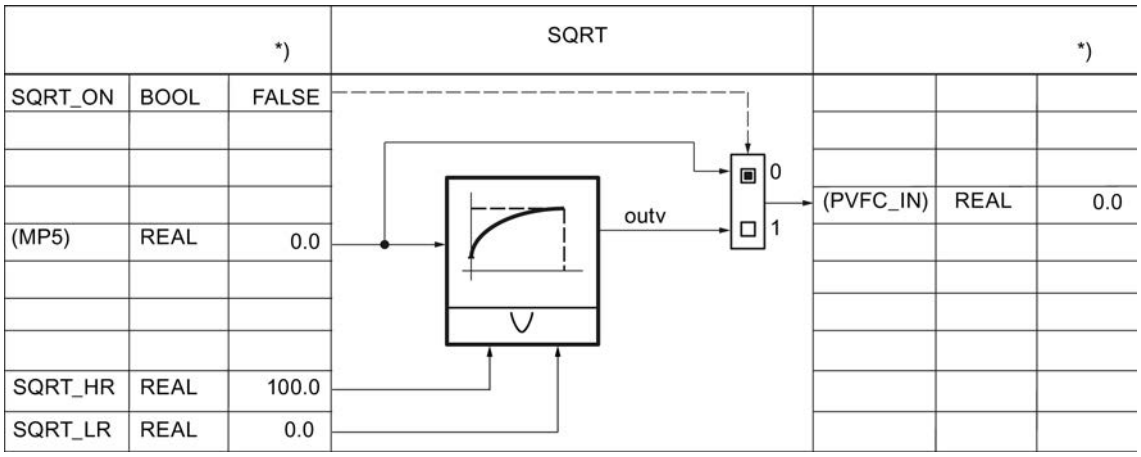
The process value can be smoothed with a first-order delay element.

Smooth controlled variable (LAG1ST) (Page 9705)



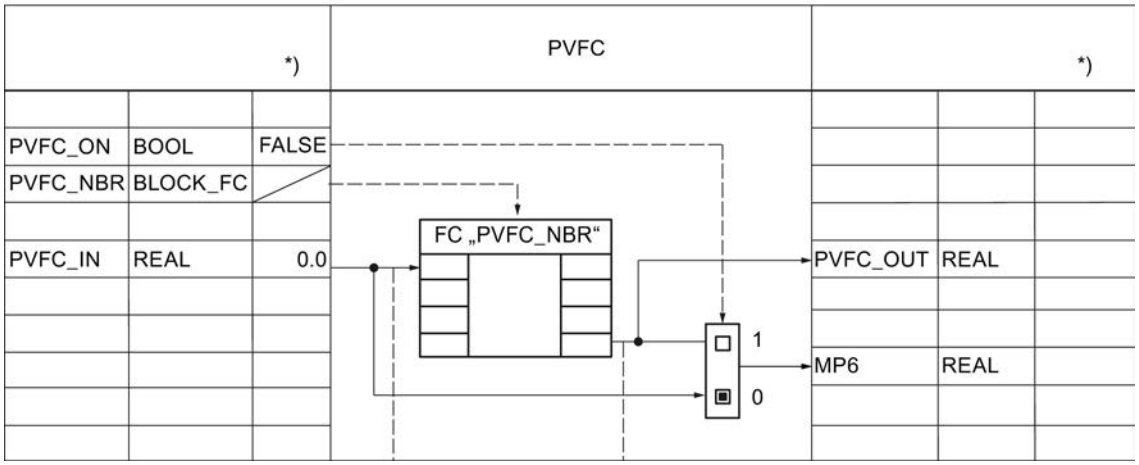
The process value can be linearized with the square root. The following parameters are required for this:

Square root (SQRT) (Page 9706)



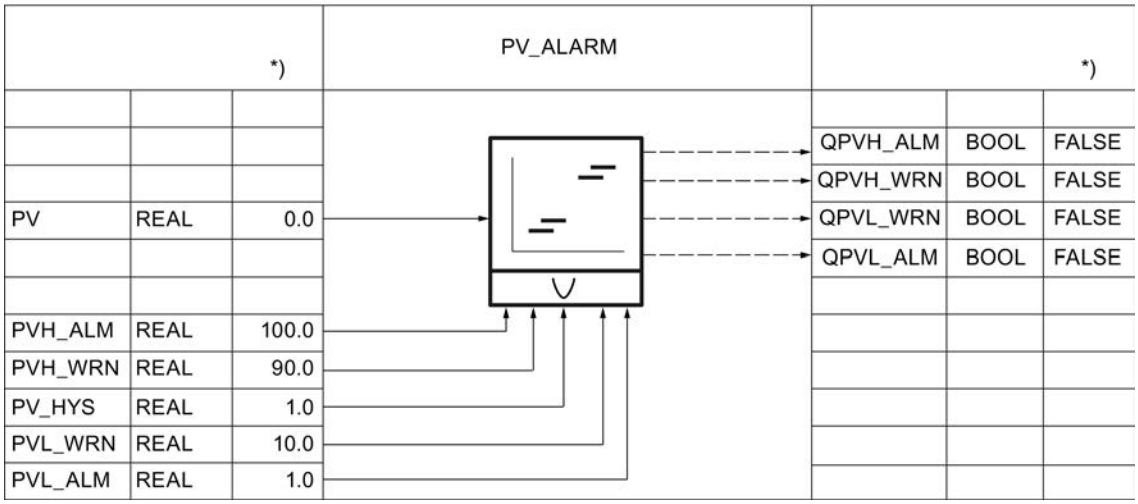
The process value can be prepared with a user-specific function. The following parameters are required for this: The interconnection must be programmed in the user FC.

FC call in the process value branch (PVFC) (Page 9708)



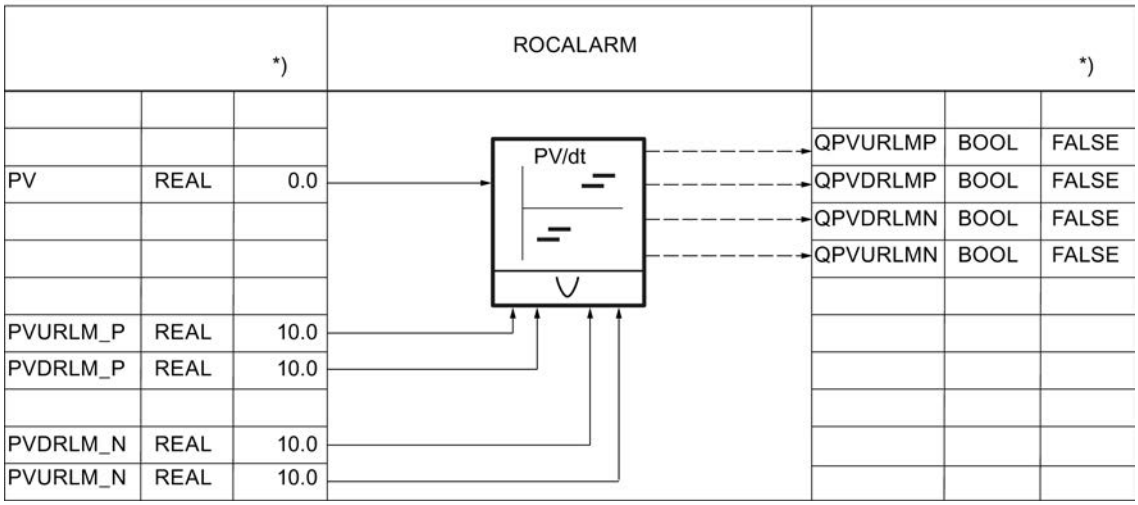
The process value is always monitored with the following parameters for warning and alarm limits.

Monitor process value for limit values (PV_ALARM) (Page 9708)



The rate of change of the process value is always limited with the following parameters.

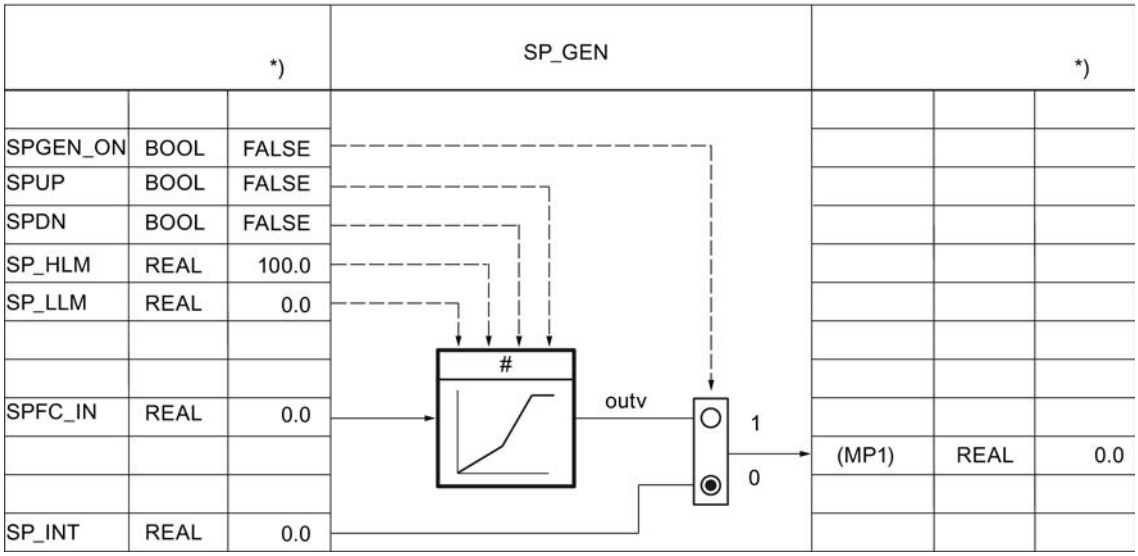
Monitor rate of change of the process value (ROCALARM) (Page 9710)



Processing setpoints

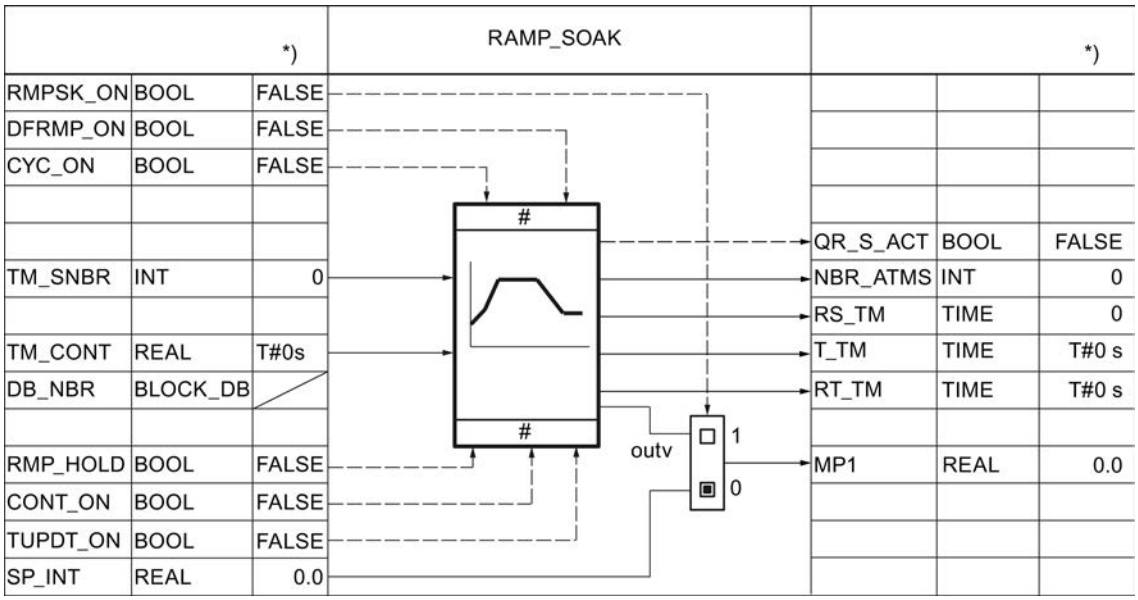
The internal setpoint SP_IN can be incremented or decremented with the following parameters.

Setpoint generator (SP_GEN) (Page 9711)



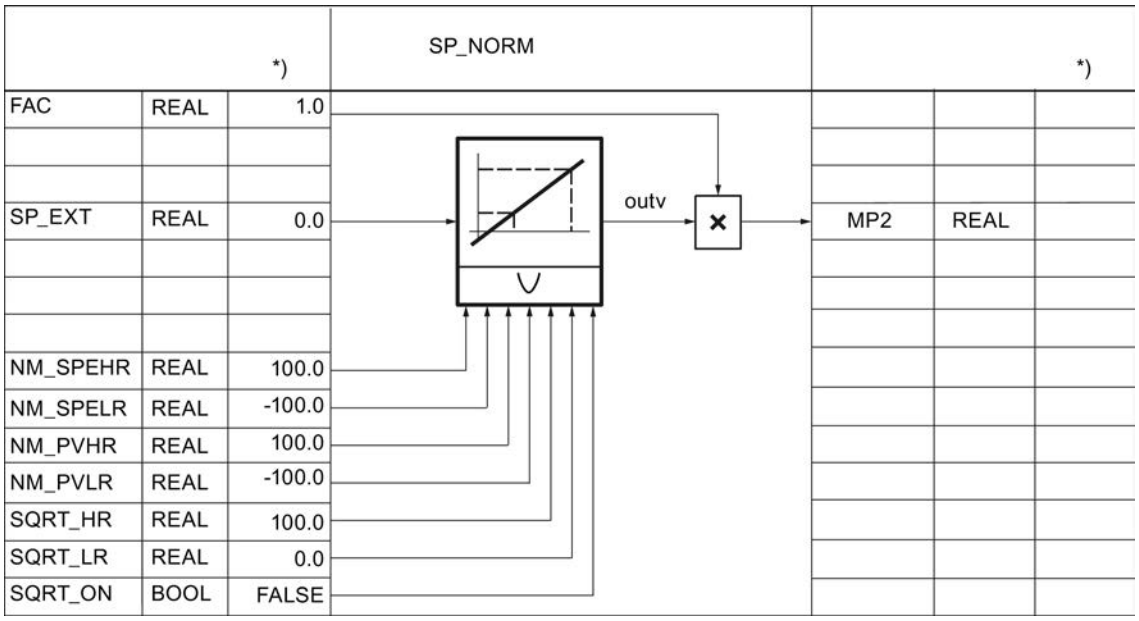
The setpoint can be specified with the following parameters via a ramp/soak function.

Ramp/soak (RMP_SOAK) (Page 9713)



The blending/ratio controller can scale the external setpoint SP_EXT using the following parameters.

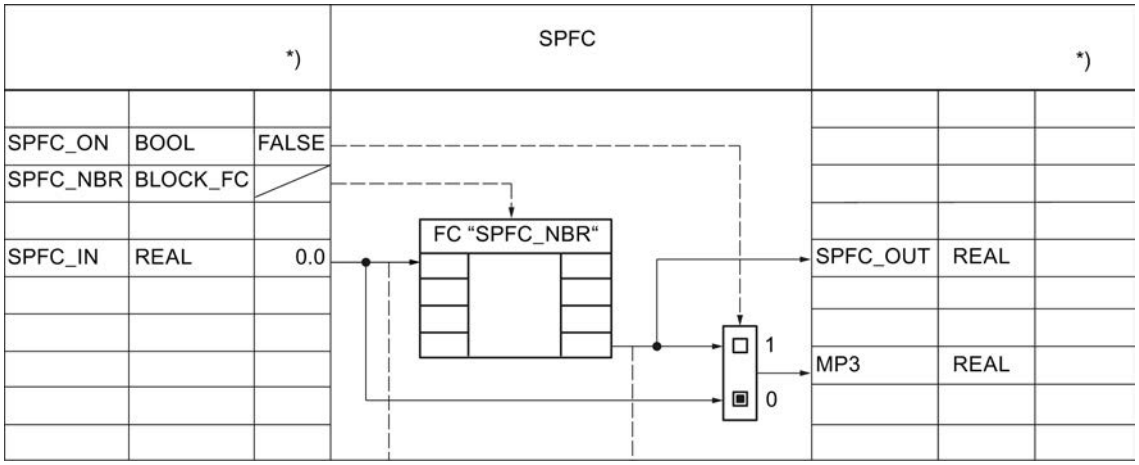
Scaling (Page 9719)



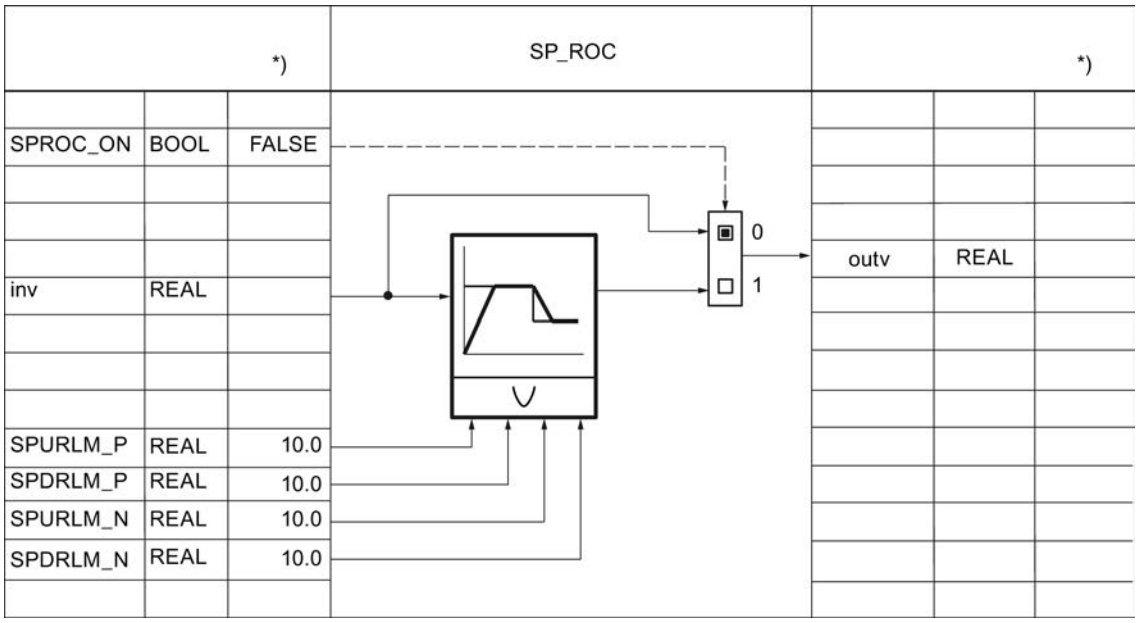
The setpoint can be further processed with a user-specific function. The following parameters are required for this: The interconnection must be programmed in the user FC.

The input value SPFC_IN is an implicit parameter. This can be monitored with the configuration tool, either via the measuring point MP1 (SPEXT_ON = FALSE) or via the measuring point MP2 (SPEXT_ON = TRUE). The output value is accessible at the measuring point MP3. The input SPFC_IN is then switched through on the setpoint branch, if SPFC_ON =FALSE.

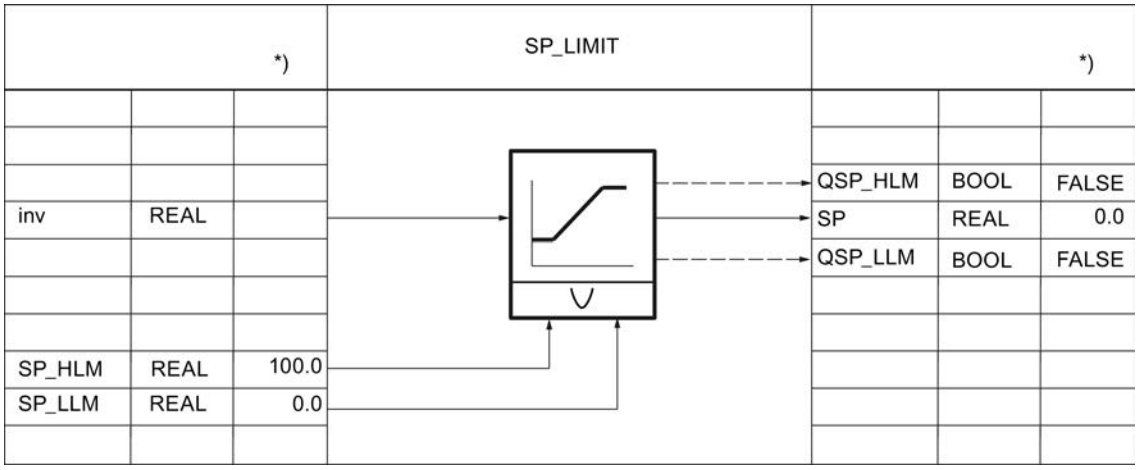
Limiting functions (Page 9720)



The rate of change of the setpoint can be limited with the following parameters.



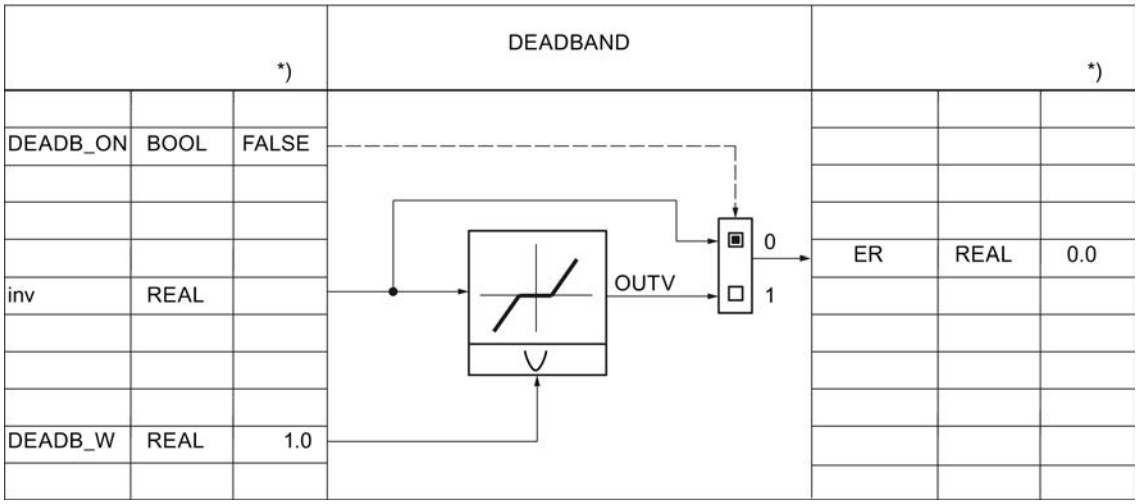
The setpoint value is always monitored with the following parameters for an absolute high and low limit.



Generating control deviation

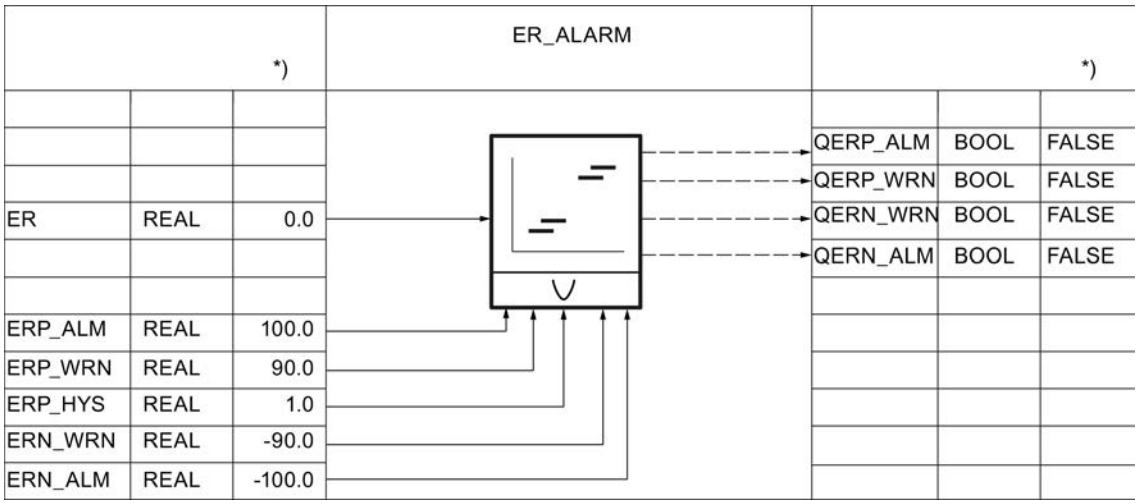
The following parameters can be used to route the control deviation over a deadband.

Deadband (Page 9723)



The control deviation is always monitored with the following parameters for warning and alarm limits.

Alarm (Page 9724)



Control algorithm for PID_ES

Control algorithm

Control algorithm (Page 9725)

A total of five switches are available to specify an effective controller structure.

Controller structures (Page 9727)

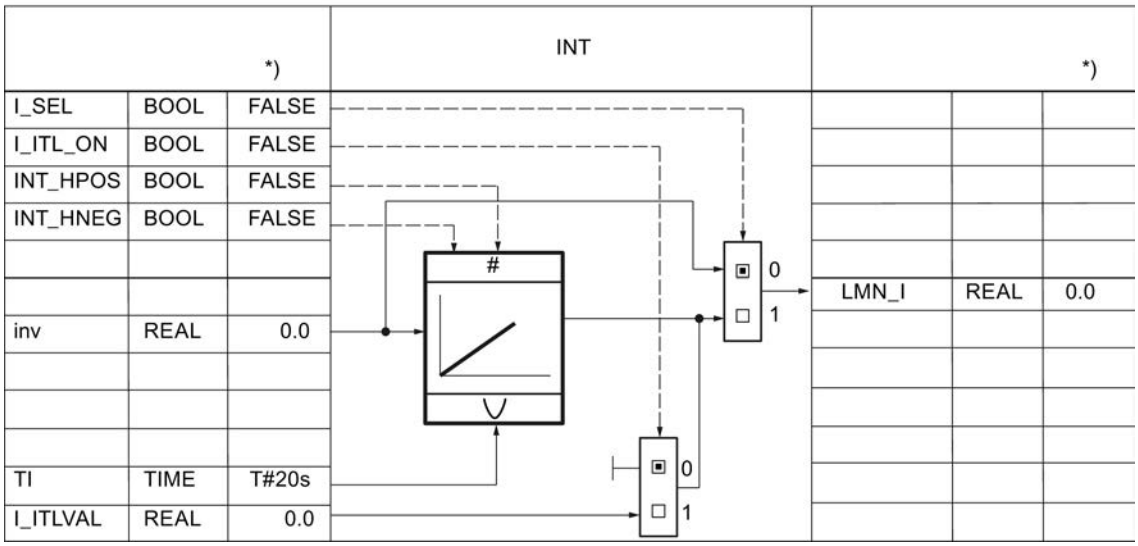
Controller structure	Switch				
	P_SEL	I_SEL	D_SEL	PFDB_SEL	DFDB_SEL
P control	TRUE	FALSE	FALSE	FALSE	FALSE
P control (P in feedback path)	TRUE	FALSE	FALSE	TRUE	FALSE
PI control	TRUE	TRUE	FALSE	FALSE	FALSE
PI control (P in feedback path)	TRUE	TRUE	FALSE	TRUE	FALSE
PD control	TRUE	FALSE	TRUE	FALSE	FALSE
PD control (P in feedback path)	TRUE	FALSE	TRUE	FALSE	TRUE
PID control	TRUE	TRUE	TRUE	FALSE	FALSE
PID control (P/D in feedback path)	TRUE	TRUE	TRUE	FALSE	TRUE

The proportional action is determined by the proportional gain GAIN and can be monitored at LMN_P.

Proportional component (Page 9730)

The integral action is calculated using the following parameters.

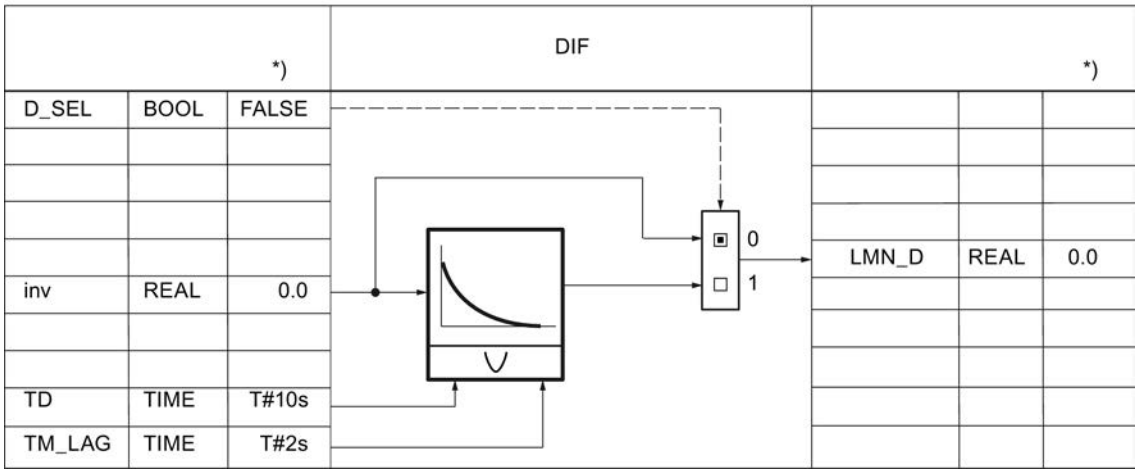
Integral action (Page 9731)



*) Default for creation of new technology object

The derivative action is calculated using the following parameters.

Derivative component (Page 9734)



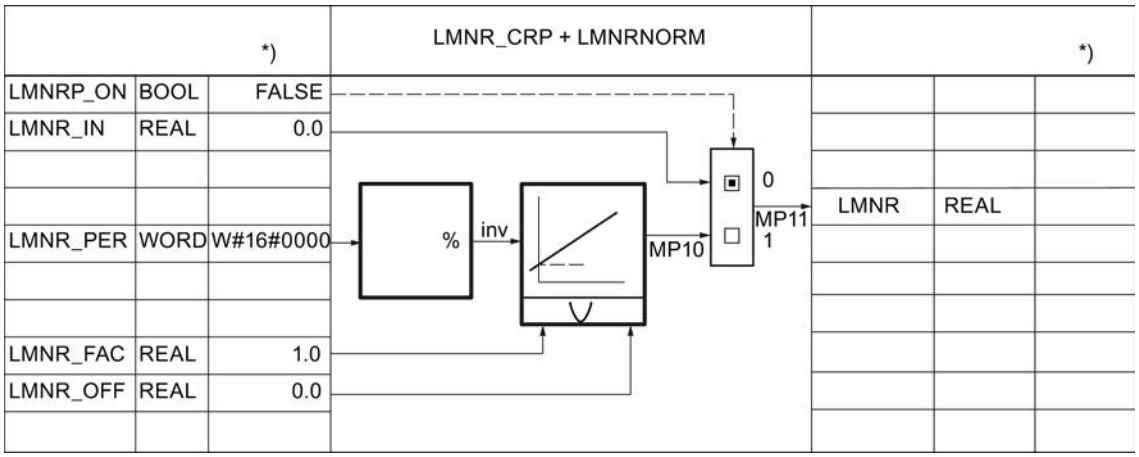
Manipulated variable PID_ES

Scaling the position feedback

The position feedback in I/O format is converted to floating-point format. The position feedback in I/O format can be monitored at MP10.

The following table provides an overview of ranges and numerical values before and after processing.

I/O value LMNR_PER	LMNR in %
32767	118.515
27648	100.000
1	0.003617
0	0.000
-1	-0.003617
-27648	-100.000
-32768	-118.519

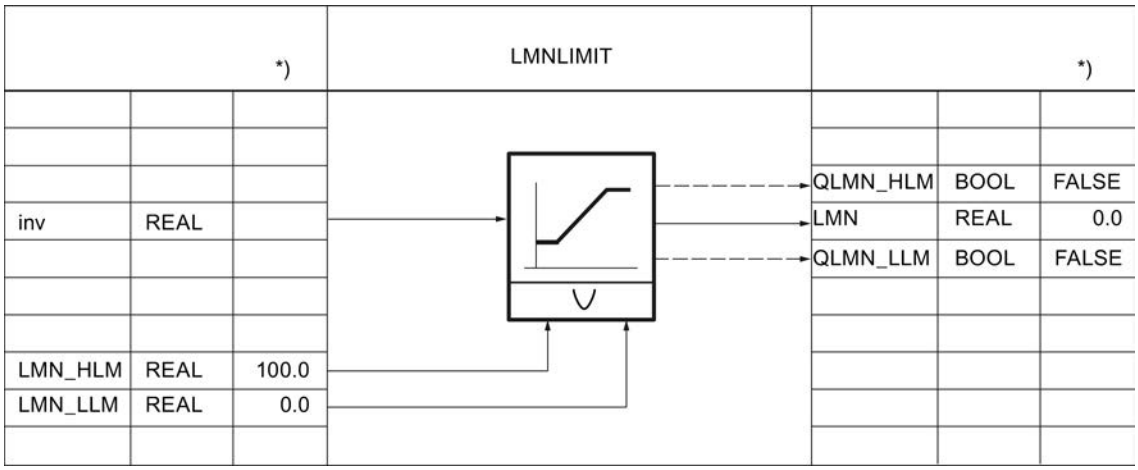


*) Default for creation of new technology object.

Manipulated value limit

The manipulated variable is always monitored with the following parameters for an absolute high and low limit. The implicit parameter can be monitored at MP9.

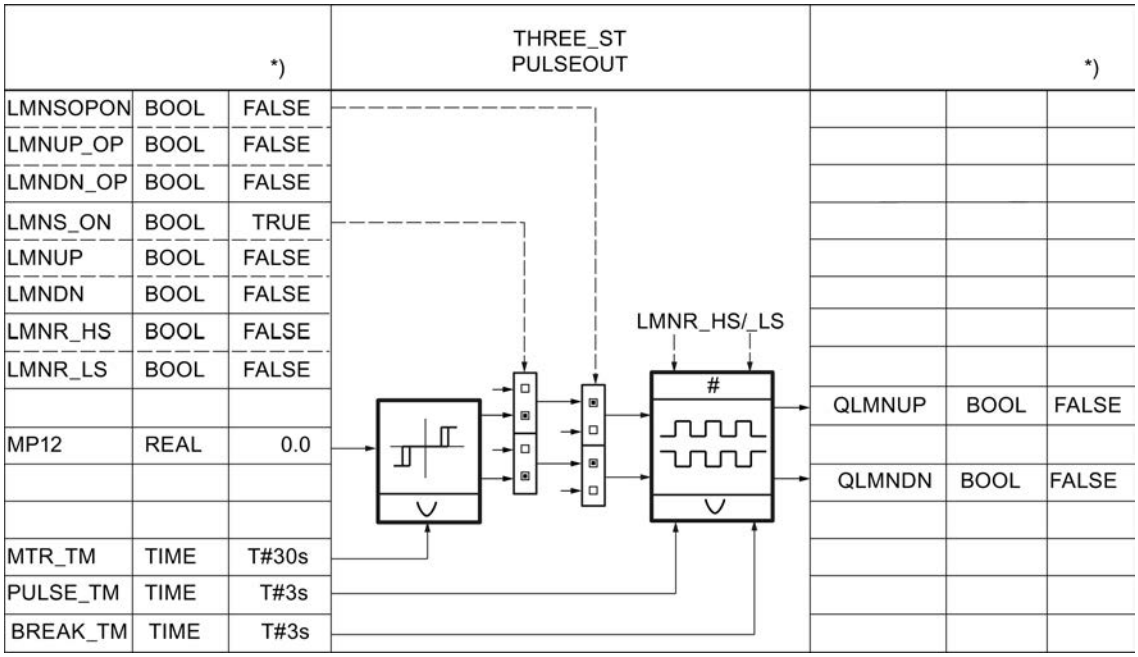
Limiting functions (Page 9743)



Step controller with position feedback

For step controllers with position feedback, the actuating signals are generated using the following parameters:

Step controller with position feedback (Page 9735)

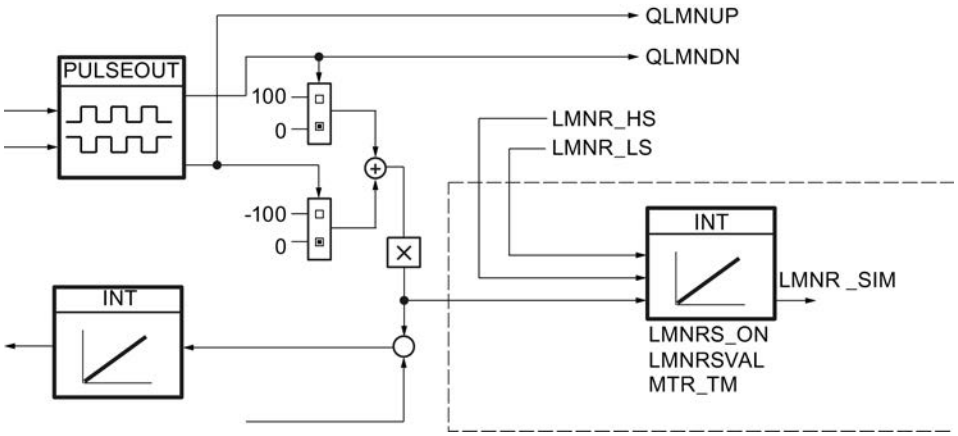


Step controller without position feedback

The position feedback can be simulated. The position feedback is simulated by an integrator with motor transition time MTR_TM as the integral action time constant. When LMNRS_ON = FALSE, the start value of the LMNRSVAL parameter is output at the output of the LMNR_SIM integrator. On switchover to LMNRS_ON = TRUE, the simulation starts up with this start value.

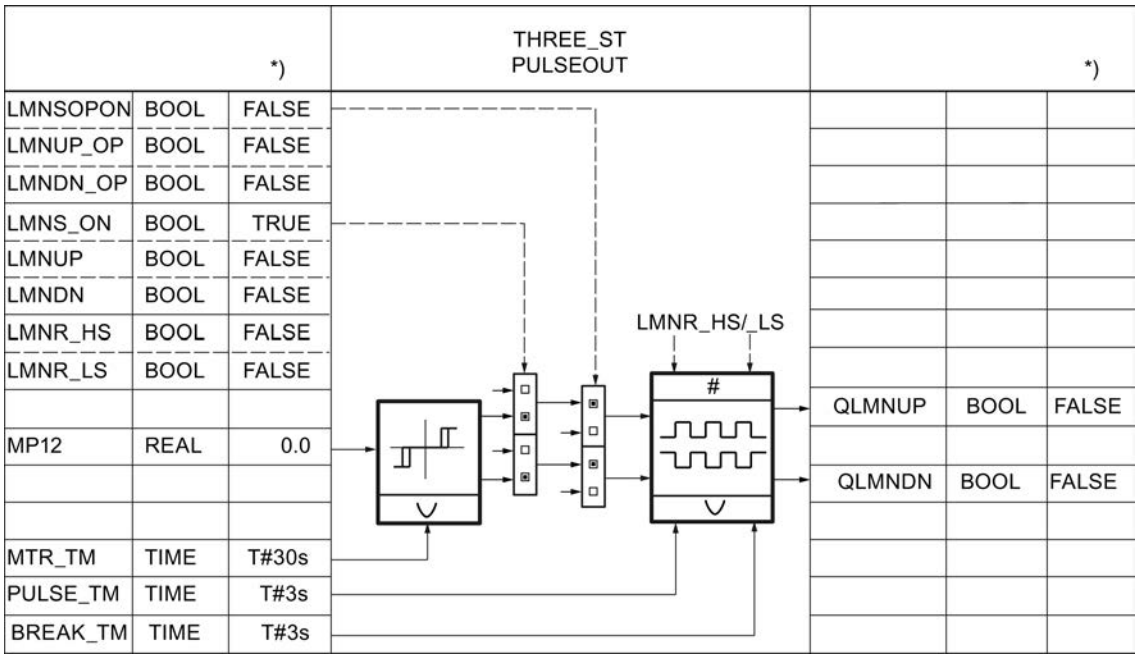
If LMNR_HS = TRUE, the integration is held up, and if LMNR_LS = TRUE, it is held down. The simulated position feedback is not adjusted to the endstop signals.

Step controller without position feedback (Page 9738)



Note

The position feedback is merely simulated. It does not have to match the actual position of the actuator. If a real position feedback is available, it should always be used.



See also

Manual mode (Page 9740)

Pulse generator (Page 9744)

PID_ES in cascade arrangements

Interruption of the controller cascade

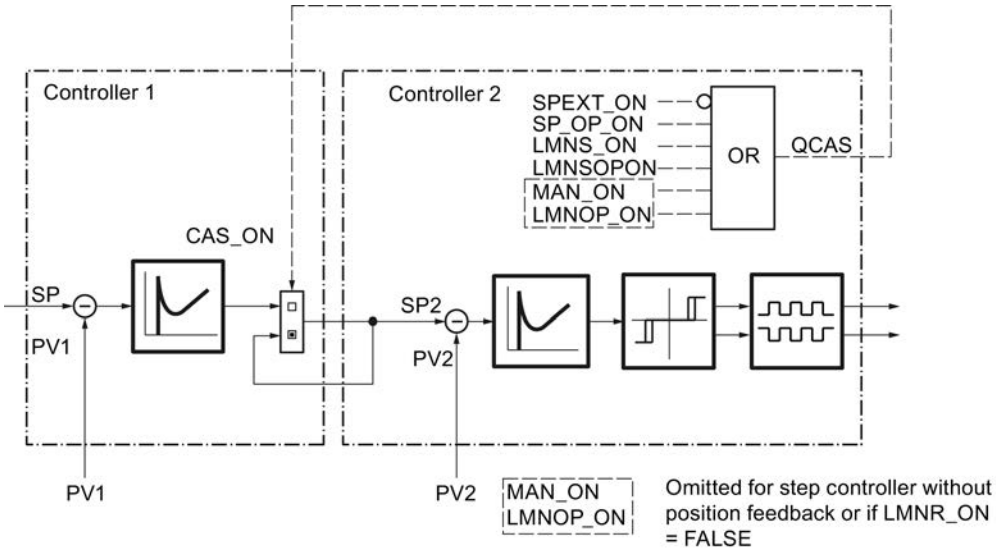
In a cascade, several controllers are in direct relationship with each other. Therefore, you must take precautions to ensure problem-free cascade operation following an interruption of the cascade structure at any point.

In cascade controls, for this reason, a QCAS signal is generated in the secondary controllers by applying OR logic from the status signals of the switches in the setpoint and manipulated variable branches. This signal actuates a switch in the higher-level controllers that converts the controllers to follow-up mode. The follow-up value is always the setpoint SP of the lower-level circuit.

A smooth switchover occurs from follow-up mode to automatic mode, just as the switchover from manual to automatic mode.

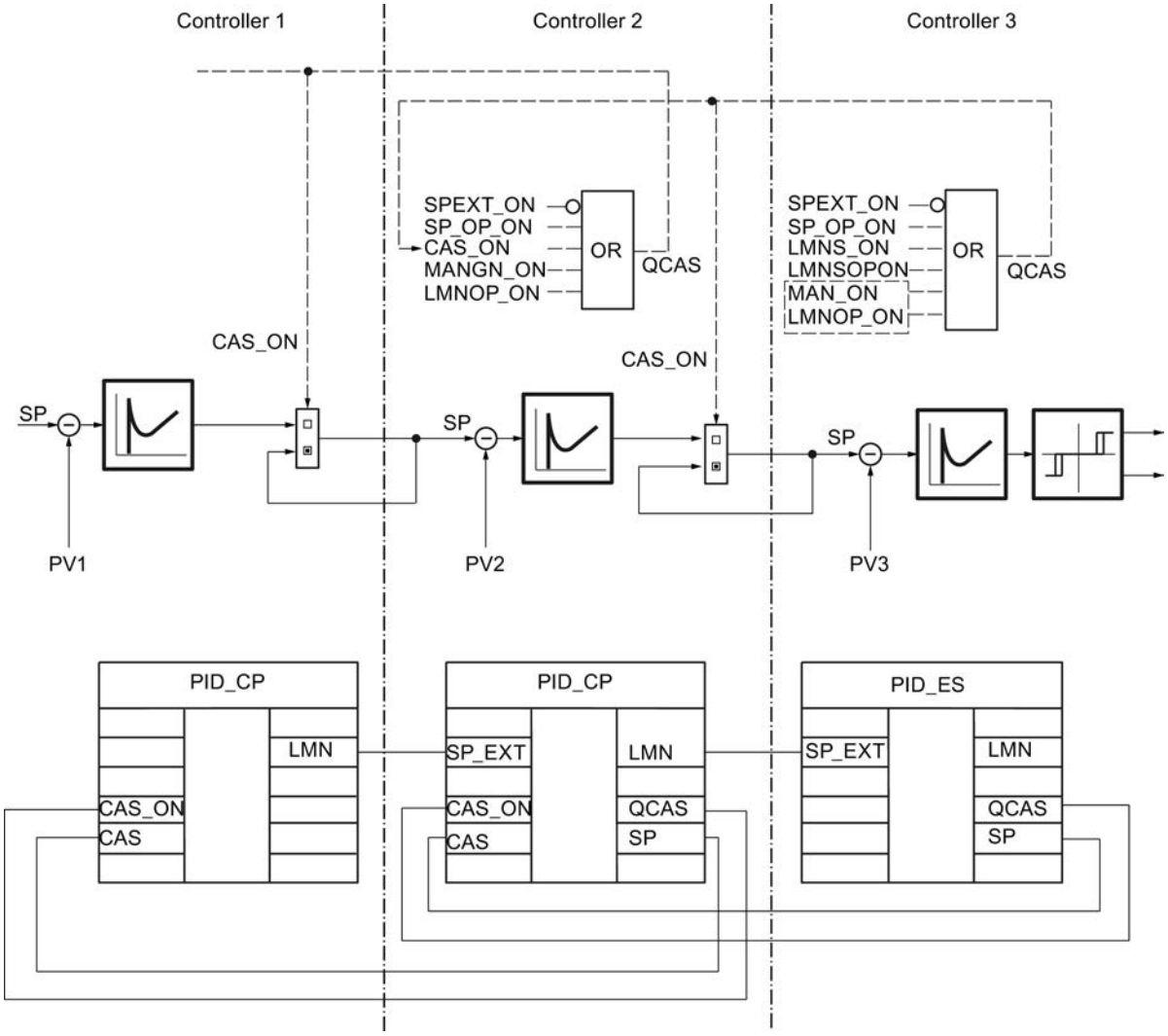
Note

PID_ES can only be used in cascade arrangements as a follow-up controller in lower-level control loops.



Block interconnection

The following figure shows the principle of the controller interconnection in multi-loop cascades.



Input parameters PID_ES

Table 9-207

Parameter	Offset	Data type	Range of values	Default	Description
COM_RST	0.0	BOOL		FALSE	Restart (Initialization routine of the instruction will be processed)
I_SEL	0.1	BOOL		TRUE	Integral action on
D_SEL	0.2	BOOL		FALSE	Derivative action on
MAN_ON	0.3	BOOL		TRUE	Manual mode on (control loop interrupted, LMN set manually)

Parameter	Offset	Data type	Range of values	Default	Description
LMNR_HS	0.4	BOOL		FALSE	High limit signal of position feedback
LMNR_LS	0.5	BOOL		FALSE	Low limit signal of position feedback
CYCLE	2.0	TIME	> 20 ms (S7–300)	T#1s	Sampling time (time between block calls = constant) Make sure to assign this parameter with the value of the cyclic interrupt cycle clock of the OB in which the PID_CP instruction runs. Otherwise, the time-dependent functions will not work correctly. (Exception: You are using time scaling, e.g., via the loop scheduler)
SP_INT	6.0	REAL	Engineering value range (physical quantity)	0.0	Internal setpoint SP_INT can be changed via a HMI with SPUP and SPDN
SP_EXT	10.0	REAL	Engineering value range (physical quantity)	0.0	External setpoint SP_EXT is used for blending, ratio and cascade controllers and can be scaled.
PV_IN	14.0	REAL	Engineering value range (physical quantity)	0.0	Process value input (PV in floating-point format)
PV_PER	18.0	INT		W#16#0000	Process value of I/O
GAIN	20.0	REAL	Overall value range (dimensionless)	2.0	Proportional gain (= controller gain)
TI	24.0	TIME	TI ≥ CYCLE	T#20s	Integral action time
TD	28.0	TIME	TD ≥ CYCLE	T#10s	Derivative-action time
TM_LAG	32.0	TIME	TM_LAG ≥ CYCLE/ 2	T#2s	Time delay of derivative action
DISV	36.0	REAL	-100.0 ... 100.0 [%]	0.0	Disturbance variable
LMNR_IN	40.0	REAL	0.0 ... 100.0 [%]	0.0	Position feedback (LMNR in floating-point format)
LMNR_PER	44.0	INT		W#16#0000	Position feedback of I/O (LMNR in I/O format)
SP_HLM	46.0	REAL	Engineering value range (physical quantity)	100.0	Setpoint high limit
SP_LLM	50.0	REAL	Engineering value range (physical quantity)	0.0	Setpoint low limit
LMN_HLM	54.0	REAL	LMN_LLM ... 100.0 [%]	100.0	Manipulated variable: High limit
LMN_LLM	58.0	REAL	0.0 ... LMN_HLM [%]	0.0	Manipulated variable: Low limit
DB_NBR	62.0	BLOCK_DB		DB 1	Data block number (DB with the time slices of the ramp/soak profile)

Parameter	Offset	Data type	Range of values	Default	Description
SPFC_NBR	64.0	BLOCK_FC		FC 0	Setpoint FC number (self-defined FC in setpoint branch)
PVFC_NBR	66.0	BLOCK_FC		FC 0	Process value FC number (self-defined FC in process value branch)

Output parameters PID_ES

Table 9-208

Parameter	Offset	Data type	Preassigned value	Description
QLMNUP	68.0	BOOL	FALSE	Manipulated variable signal UP
QLMNDN	68.1	BOOL	FALSE	Manipulated variable signal DOWN
QCAS	68.2	BOOL	FALSE	Signal for cascade mode (interconnected to CAS_ON of higher-level controller)
LMN	70.0	REAL	0.0	Manipulated variable signal (according to control algorithm)
SP	74.0	REAL	0.0	Setpoint (effective setpoint)
PV	78.0	REAL	0.0	Process value (output of effective process value in cascade mode)

In/out parameters PID_ES

Table 9-209

Parameter	Offset	Data type	Preassigned value	Description
MAN	82.0	REAL	0.0	Manual output value (for manipulated variable assignment by means of operator control and monitoring functions)

Static variables PID_ES

Parameter	Offset	Data type	Range of values	Preassigned value	Description
PVH_ALM	86.0	REAL	PVH_WRN...100.0	100.0	Process value: High limit 'Alarm' Valid value range:
PVH_WRN	90.0	REAL	PVL_WRN... PVH_ALM	90.0	Process value: High limit 'Warning' Valid value range:
PVL_WRN	94.0	REAL	PVL_ALM... PVH_WRN	-90.0	Process value: Low limit 'Warning' Valid value range:
PVL_ALM	98.0	REAL	-100.0...PVL_WRN	-100.0	Process value: Low limit 'Alarm' Valid value range:

Parameter	Offset	Data type	Range of values	Preassigned value	Description
SPGEN_ON	102.0	BOOL		FALSE	Setpoint manipulation ON (adjusting the setpoint using the up/down switch)
SPUP	102.1	BOOL		FALSE	Setpoint UP
SPDN	102.2	BOOL		FALSE	Setpoint DOWN
RMPSK_ON	102.3	BOOL		FALSE	Ramp/soak function ON (setpoint is specified as a ramp/soak profile)
SPEXT_ON	102.4	BOOL		FALSE	External setpoint ON (for interconnection with other controller blocks)
MANGN_ON	102.5	BOOL		FALSE	Manual mode ON (LMN set via switch)
MANUP	102.6	BOOL		FALSE	Manual output value UP
MANDN	102.7	BOOL		FALSE	Manual output value DOWN
LMNS_ON	103.0	BOOL		FALSE	Manual mode for actuating signals ON
LMNUP	103.1	BOOL		FALSE	Manipulated variable signal UP (output signal QLMNUP will be manipulated)
LMNDN	103.2	BOOL		FALSE	Manipulated variable signal DOWN (output signal QLMNDN will be manipulated)
DFRMP_ON	103.3	BOOL		FALSE	Preassign output of ramp/soak function (output is preassigned with SP_INT)
CYC_ON	103.4	BOOL		FALSE	Cyclic repetition ON (ramp/soak profile will be repeated automatically)
RMP_HOLD	103.5	BOOL		FALSE	Ramp/soak function HOLD (output of ramp/soak function is frozen)
CONT_ON	103.6	BOOL		FALSE	Continue (with schedule) (ramp/soak profile will be continued at the next time slice)
TUPDT_ON	103.7	BOOL		FALSE	Recalculate total time (total time of ramp/soak profile will be recalculated)
SPFC_ON	104.0	BOOL		FALSE	Call setpoint FC
SPROC_ON	104.1	BOOL		FALSE	Ramp function ON (rate of change limiting of SP)
PVPER_ON	104.2	BOOL		FALSE	Process value of I/O ON (interconnection with I/O modules)
LAG1STON	104.3	BOOL		FALSE	First-order delay element ON
SQRT_ON	104.4	BOOL		FALSE	Square root function ON
PVFC_ON	104.5	BOOL		FALSE	Call process value FC
DEADB_ON	104.6	BOOL		FALSE	Deadband ON (small disturbances and noise will be filtered)
P_SEL	104.7	BOOL		TRUE	P component ON

Parameter	Offset	Data type	Range of values	Preassigned value	Description
PFDB_SEL	105.0	BOOL		FALSE	Switch P component to feedback
INT_HPOS	105.1	BOOL		FALSE	Freeze I component in positive direction
INT_HNEG	105.2	BOOL		FALSE	Freeze I component in negative direction
I_ITL_ON	105.3	BOOL		FALSE	Preassign I component
DFDB_SEL	105.4	BOOL		FALSE	Switch D component to feedback
DISV_SEL	105.5	BOOL		FALSE	Apply disturbance variable
LMNR_ON	105.6	BOOL		FALSE	Position feedback ON (operating modes: step controller with/without position feedback) Do not switch while in control mode!
LMNRP_ON	105.7	BOOL		FALSE	Position feedback of I/O ON
TM_SNBR	106.0	INT	≥ 0 (dimensionless)	0	No. of next time slice for continuation of the ramp/soak profile
TM_CONT	108.0	TIME	Overall value range (dimensionless)	T#0s	Time span until continuation of the ramp/soak profile (time that elapses before the ramp/soak function continues processing the ramp/soak profile at time slice TM_SNBR following an interruption)
FAC	112.0	REAL	Overall value range (dimensionless)	1.0	Factor (ratio or blending factor)
NM_SPEHR	116.0	REAL		100.0	Setpoint scaling: Input high
NM_SPELR	120.0	REAL		-100.0	Setpoint scaling: Input low
SPFC_OUT	124.0	REAL	-100.0 ... 100.0	0.0	Setpoint FC output (interconnected to the output of the FC in the setpoint branch)
SPURLM_P	128.0	REAL	≥ 0 [physical quantity/s]	10.0	Setpoint up rate limit in positive range
SPDRLM_P	132.0	REAL	≥ 0 [physical quantity/s]	10.0	Setpoint down rate limit in positive range
SPURLM_N	136.0	REAL	≥ 0 [physical quantity/s]	10.0	Setpoint up rate limit in negative range
SPDRLM_N	140.0	REAL	≥ 0 [physical quantity/s]	10.0	Setpoint down rate limit in negative range
NM_PIHR	144.0	REAL		100.0	Process value scaling: Input high
NM_PILR	148.0	REAL		-100.0	Process value scaling: Input low
NM_PVHR	152.0	REAL		100.0	Process value scaling: Output high
NM_PVLR	156.0	REAL		-100.0	Process value scaling: Output low
PV_TMLAG	160.0	TIME	Overall value range	T#5s	Process value time delay (delay of PT1 element in the PV branch)
SQRT_HR	164.0	REAL		100.0	Square root: Measuring range output high

Parameter	Offset	Data type	Range of values	Preassigned value	Description
SQRT_LR	168.0	REAL		0.0	Square root: Measuring range output low
PVFC_OUT	172.0	REAL	-100.0 ... 100.0	0.0	Process value FC output (interconnected to the output of the FC in the process value branch)
PVURLM_P	176.0	REAL	≥ 0 [physical quantity/s]	10.0	Process value up rate limit in the positive range
PVDRLM_P	180.0	REAL	≥ 0 [physical quantity/s]	10.0	Process value down rate limit in the positive range
PVURLM_N	184.0	REAL	≥ 0 [physical quantity/s]	10.0	Process value up rate limit in the negative range
PVDRLM_N	188.0	REAL	≥ 0 [physical quantity/s]	10.0	Process value down rate limit in the negative range
PV_HYS	192.0	REAL	≥ 0	1.0	Process value hysteresis (prevents 'flickering' of monitoring display)
DEADB_W	196.0	REAL	0.0 ... 100.0	1.0	Deadband width (determines the size of the deadband area)
ERP_ALM	200.0	REAL	0 ... 200.0	100.0	Control deviation: Positive limit 'Alarm'
ERP_WRN	204.0	REAL	0 ... 200.0	90.0	Control deviation: Positive limit 'Warning'
ERN_WRN	208.0	REAL	-200.0 ... 0	-90.0	Control deviation: Negative limit 'Warning'
ERN_ALM	212.0	REAL	-200.0 ... 0	-100.0	Control deviation: Negative limit 'Alarm'
ER_HYS	216.0	REAL	≥0	1.0	Control deviation hysteresis (prevents 'flickering' of monitoring display)
I_ITLVAL	220.0	REAL	-100.0 ... 100.0 [%]	0.0	Initialization value for I action
LMNR_FAC	224.0	REAL	Overall value range (dimensionless)	1.0	Position feedback factor (factor for adaptation of the feedback range)
LMNR_OFF	228.0	REAL	-100.0 ... 100.0 [%]	0.0	Position feedback offset (zero point of feedback scaling)
PULSE_TM	232.0	TIME	= n * CYCLE / n=0,1,2...	T#3s	Minimum pulse time
BREAK_TM	236.0	TIME	= n * CYCLE / n=0,1,2...	T#3s	Minimum break time
MTR_TM	240.0	TIME	≥ CYCLE	T#30s	Motor actuating time
PHASE	244.0	INT		0	Phase of PID self tuner
QPVH_ALM	246.0	BOOL		FALSE	Process value: High limit 'Alarm' triggered
QPVH_WRN	246.1	BOOL		FALSE	Process value: High limit 'Warning' triggered
QPVL_WRN	246.2	BOOL		FALSE	Process value: Low limit 'Warning' triggered
QPVL_ALM	246.3	BOOL		FALSE	Process value: Low limit 'Alarm' triggered
QR_S_ACT	246.4	BOOL		FALSE	Time table for ramp/soak profile will be processed
QSP_HLM	246.5	BOOL		FALSE	Setpoint: High limit triggered

Parameter	Offset	Data type	Range of values	Preassigned value	Description
QSP_LLM	246.6	BOOL		FALSE	Setpoint: Low limit triggered
QPVURLMP	246.7	BOOL		FALSE	Process value: Up rate limit in positive range triggered
QPVDRLMP	247.0	BOOL		FALSE	Process value: Down rate limit in positive range triggered
QPVURLMN	247.1	BOOL		FALSE	Process value: Up rate limit in negative range triggered
QPVDRLMN	247.2	BOOL		FALSE	Process value: Down rate limit in negative range triggered
QERP_ALM	247.3	BOOL		FALSE	Control deviation: Positive limit 'Alarm' triggered
QERP_WRN	247.4	BOOL		FALSE	Control deviation: Positive limit 'Warning' triggered
QERN_WRN	247.5	BOOL		FALSE	Control deviation: Negative limit 'Warning' triggered
QERN_ALM	247.6	BOOL		FALSE	Control deviation: Negative limit 'Alarm' triggered
QLMN_HLM	247.7	BOOL		FALSE	Manipulated variable: High limit triggered
QLMN_LLM	248.0	BOOL		FALSE	Manipulated variable: Low limit triggered
NBR_ATMS	250.0	INT		0	Number of the time slice currently being approached by the ramp/soak function
RS_TM	252.0	TIME		T#0s	"Current residual time" of ramp/soak profile until the next time slice
T_TM	256.0	TIME		T#0s	"Total time" = elapsed time of ramp/soak profile
RT_TM	260.0	TIME		T#0s	"Total residual time" = time remaining until the end of the ramp/soak profile
ER	264.0	REAL		0.0	Control deviation
LMN_P	268.0	REAL		0.0	P component
LMN_I	272.0	REAL		0.0	I component
LMN_D	276.0	REAL		0.0	D component
SPFC_IN	280.0	REAL		0.0	Setpoint FC input (interconnected to input of the self-defined FC)
PVFC_IN	284.0	REAL		0.0	Process value FC input (interconnected to input of the self-defined FC)
SP_OP_ON	288.0	BOOL		FALSE	Setpoint manipulation ON (value of SP_OP will be applied as the setpoint)
PV_OP_ON	288.1	BOOL		FALSE	Process value manipulation ON (value of PV_OP will be applied as the setpoint)
LMNOP_ON	288.2	BOOL		FALSE	Manipulated variable manipulation ON (value of LMN_OP will be applied as the setpoint)

Parameter	Offset	Data type	Range of values	Preassigned value	Description
LMNSOPON	288.3	BOOL		FALSE	Manipulated variable signal manipulation ON (LMNUP_OP and LMNDN_OP will be applied as actuating signals)
LMNUP_OP	288.4	BOOL		FALSE	Manipulated variable signal UP
LMNDN_OP	288.5	BOOL		FALSE	Manipulated variable signal DOWN
LMNRS_ON	288.6	BOOL		FALSE	Simulation of the position feedback ON
SP_OP	290.0	REAL		0.0	Setpoint manipulation in the configuration tool
PV_OP	294.0	REAL		0.0	Process value manipulation in the configuration tool
LMN_OP	298.0	REAL		0.0	Manipulated variable manipulation in the configuration tool
LMNRSVAL	302.0	REAL		0.0	Start value of the simulated position feedback
LMNR_SIM	306.0	REAL		0.0	Current value of the simulated position feedback
MP1	310.0	REAL		0.0	Measuring point 1: Internal setpoint
MP2	314.0	REAL		0.0	Measuring point 2: External setpoint
MP3	318.0	REAL		0.0	Measuring point 3: Unlimited setpoint
MP4	322.0	REAL		0.0	Measuring point 4: Process value of the I/O module
MP5	326.0	REAL		0.0	Measuring point 5: Process value after first-order delay element
MP6	330.0	REAL		0.0	Measuring point 6: Effective process value (PV)
MP7	334.0	REAL		0.0	Measuring point 7: Manipulated variable of PID algorithm
MP8	338.0	REAL		0.0	Measuring point 8: Manual output value
MP9	342.0	REAL		0.0	Measuring point 9: Unlimited manipulated variable
MP10	346.0	REAL		0.0	Measuring point 10: Position feedback I/O
MP11	350.0	REAL		0.0	Measuring point 11: Feedback value (LMNR_ON = FALSE) Position feedback (LMNR_ON = TRUE)
MP12	354.0	REAL		0.0	Measuring point 12: Three-step element input

Block diagrams for PID_ES

Signal flow diagrams

Analogous to the representation of the switches in the configuration tool, the black dot in each switch symbol indicates that the switching signal has the adjacent Boolean value (0=FALSE

or 1=TRUE) and that the signal path is switched through this point in each case. The switching signals (binary signals) are identified by dashed lines.

This means that the setpoint is specified as an absolute setpoint via SP_INT; the same applies to the process value input via PV_IN. The default control function is a standard PI controller with P function in the forward branch. The control loop is open and the percentage range of the manipulated variable is influenced by the MAN input. All other functions are passive or, if they cannot be deactivated, marginal signal characteristic parameters are preassigned within the measuring range or operating range so that the functions have no effect.

Symbols and identifiers in the signal flow graphs

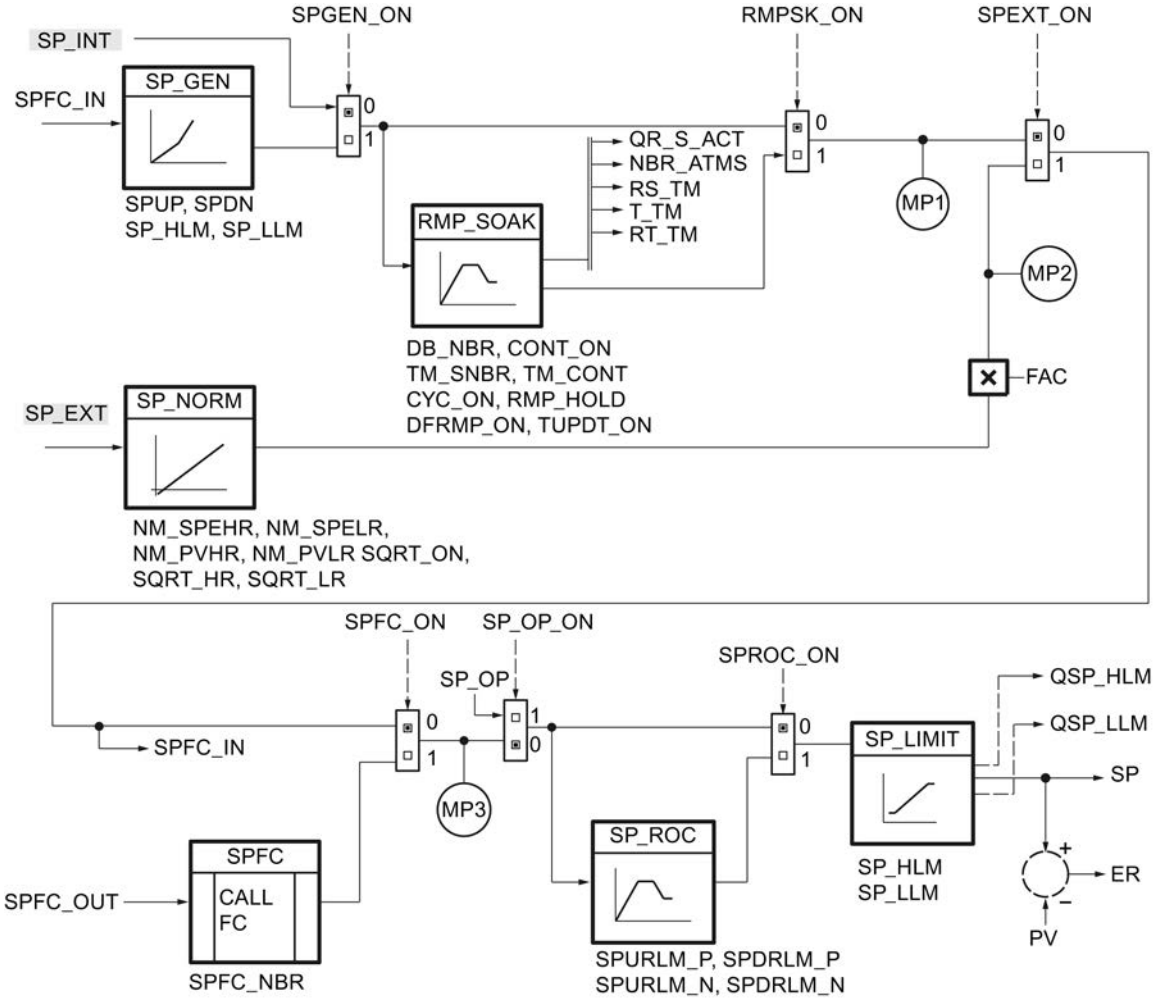
The names of the connectable process variables are shown on a shaded background. This allows you to recognize where the controller structure can be connected to the S7 I/O or directly to the measuring elements and actuators of the process.

Parameter names containing the combination of letters "OP" (for example, SP_OP/ SP_OP_ON) indicate that an intervention using the Standard PID Control configuration tool is possible at this point. The configuration tool has its own interface to the controller FB.

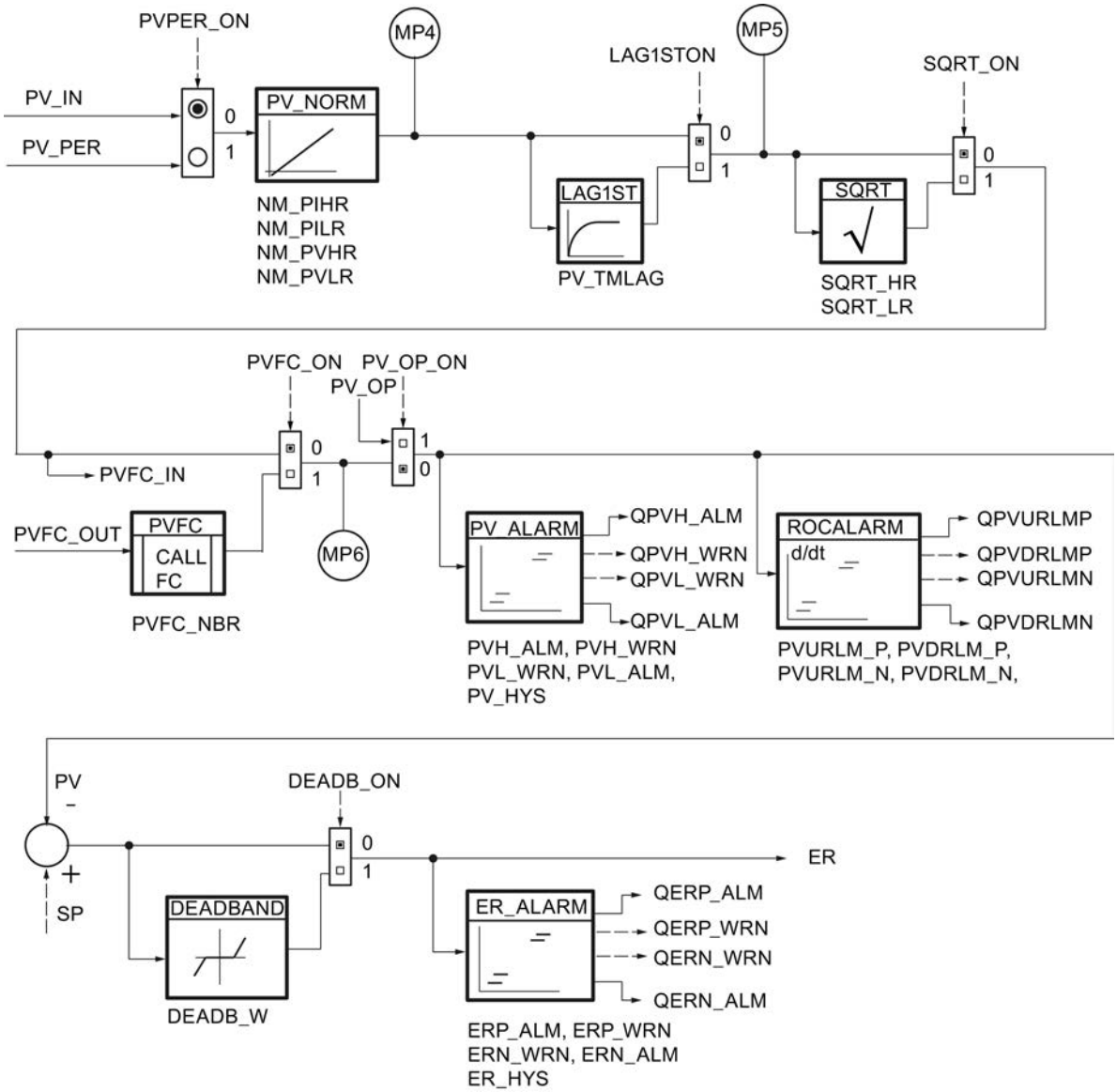
Intermediate variables in the signal characteristic can be monitored at the measuring points MP1 to MP12 marked with circles. These intermediate variables are required in order to match values before triggering "smooth" changeovers or to check the current statuses of the relevant control. The measuring point variables in the configuration tool curve recorder can be represented (statically and dynamically).

For reasons of clarity, the parameters for setting and sizing the processing operations (algorithms) are also indicated in the individual function blocks. Please refer to the descriptions in the reference section and to the representation of the individual subfunctions in the following sections.

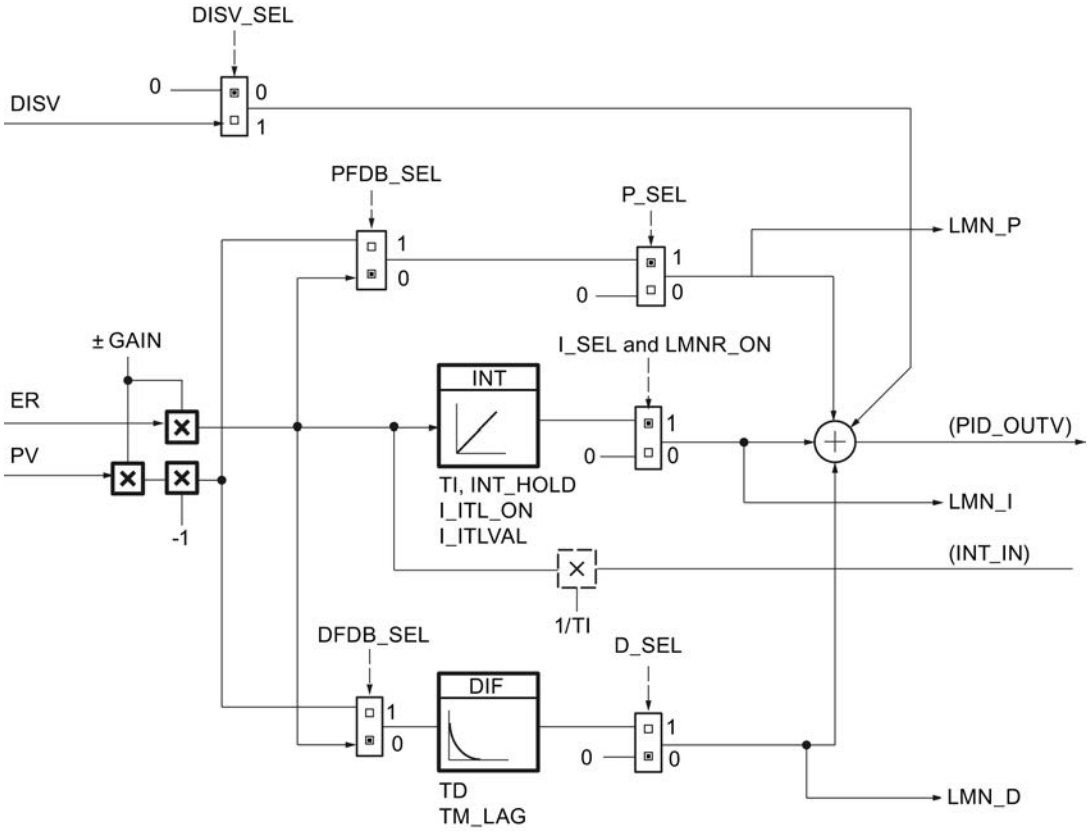
Processing the setpoint



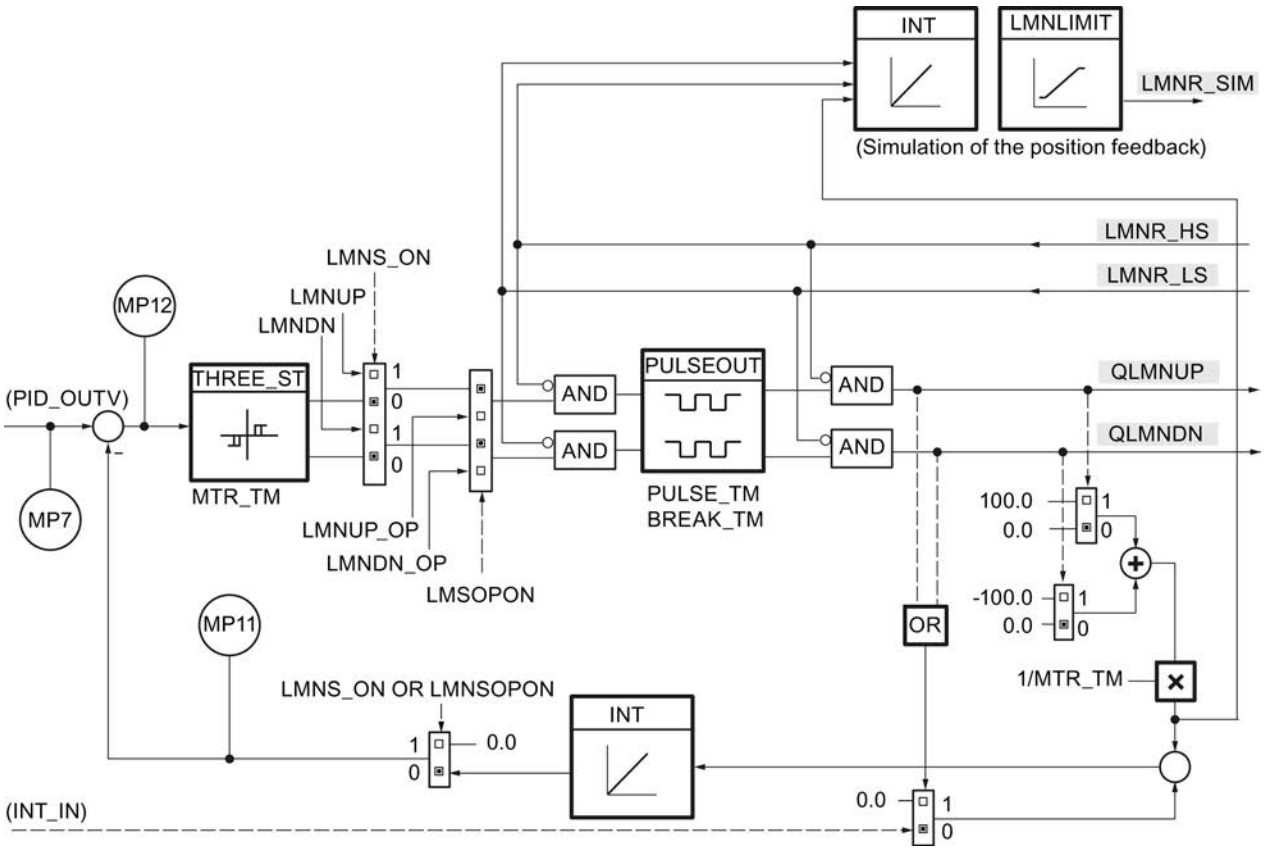
Processing the process value



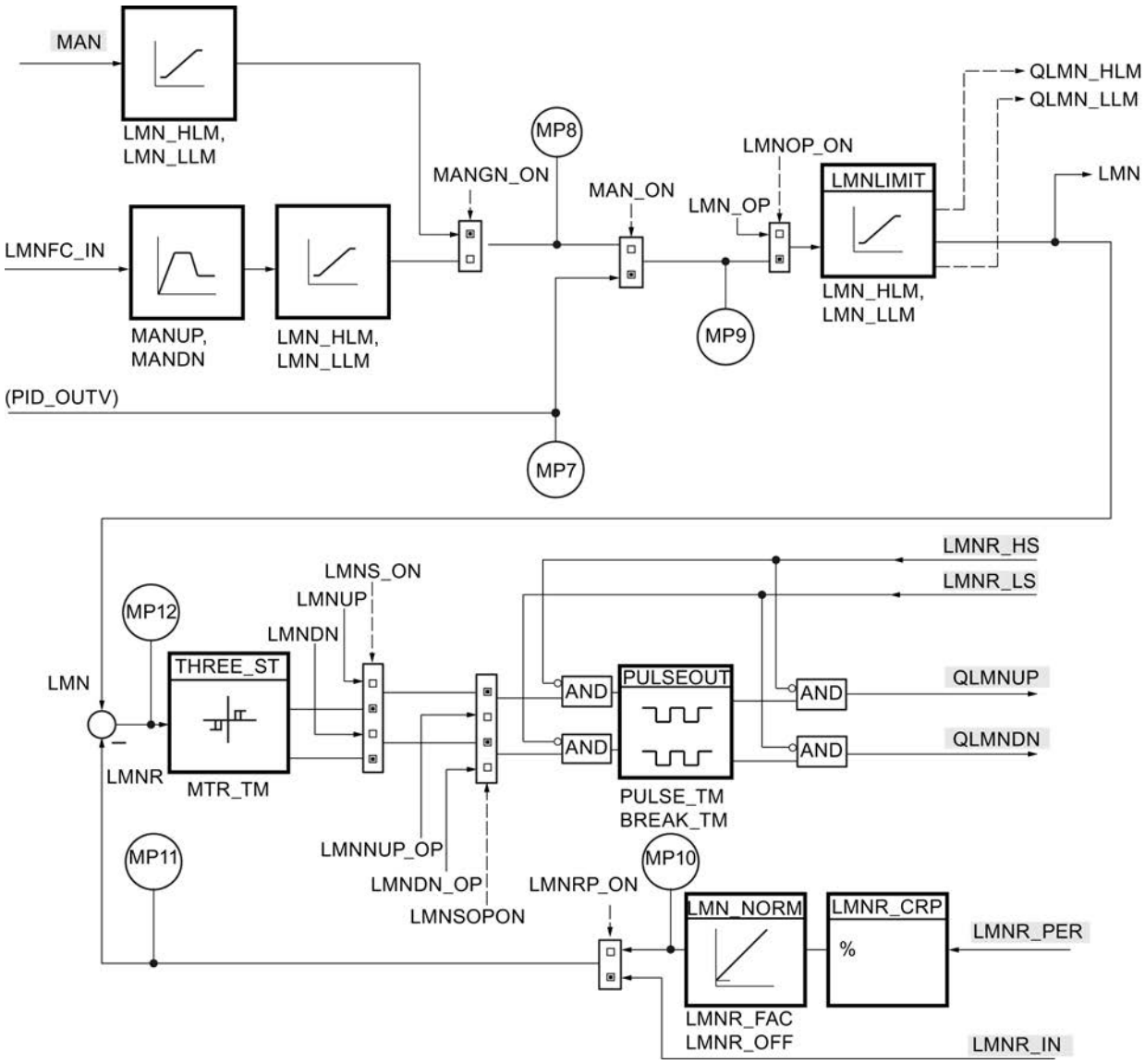
Control algorithm



Step controller without position feedback (LMNR_ON = FALSE)



Step controller with position feedback (LMNR_ON = TRUE)



Data block RMP_SOAK

Parameter	Data type	Preassigned value	Description
NBR_PTS	INT	4	Number of time slices -1 1 - 255
PI	Array [0...NBR_PTS] of Struct		Structure for the time slice data. The array must be declared by 0...NBR_PTS.
PI[0].OUTV	REAL	0.0	Output variable [0] Starting point Engineering value range

Parameter	Data type	Preassigned value	Description
PI[0].TMV	TIME	T#1s	Time value [0] Starting point
PI[1].OUTV	REAL	0.0	Output variable [1] Time slice 1 Engineering value range
PI[1].TMV	TIME	T#1s	Time value [1] Time slice 1
PI[2].OUTV	REAL	0.0	Output variable [2] Time slice 2 Engineering value range
PI[2].TMV	TIME	T#1s	Time value [2] Time slice 2
PI[3].OUTV	REAL	0.0	Output variable [3] Time slice 3 Engineering value range
PI[3].TMV	TIME	T#1s	Time value [3] Time slice 3
PI[4].OUTV	REAL	0.0	Output variable [4] Time slice 4 Engineering value range
PI[4].TMV	TIME	T#0s	Time value [4] Time slice 4 0 ms

See also

Ramp/soak (RMP_SOAK) (Page 9713)

LP_SCHED

Description of LP_SCHED

LP_SCHED is used when the number of cyclic interrupts of a CPU is not sufficient to achieve the desired (different) sampling times. It enables up to 256 control loops to be called with sampling times equal to an integer multiple of the cyclic interrupt cycle.

LP_SCHED reads the parameters you assigned from the global data block DB_LOOP, calculates from these parameters the variables necessary for call scheduling, and stores these variables in the DB_LOOP data block again.

Note

The block does not check to determine whether a global DB with the number DB_NBR actually exists or whether the GLP_NBR parameter (maximum control loop number) adapts to the length of the data block. If the parameter assignment is incorrect, the CPU changes to STOP with an "internal system error".

Call

You must call LP_SCHED in a cyclic interrupt OB prior to all controllers.

Note the following when assigning values to the input parameters:

- TM_BASE: At this input, specify the cycle clock of the cyclic interrupt OB in which LP_SCHED instruction is called.
- COM_RST: At CPU startup, you must call LP_SCHED once with COM_RST = TRUE. During cyclic operation (cyclic interrupt), you must call LP_SCHED with COM_RST = FALSE.
- DB_NBR: At this input, you specify the number of the "DB_LOOP" data block that LP_SCHED is to access.

After the LP_SCHED instruction is called, you must call all associated controllers conditionally. You cannot use LP_SCHED to call the controllers PID_CP or PID_ES, as their input and output parameters must be assigned when the instructions are called.

The following is an example of calling the LP_SCHED instruction and of conditionally calling a controller.

STL

```
    CALL "LP_SCHED"
        TM_BASE:                                    Here, the cycle clock of the cyclic interrupt
                                                    is parameterized. Example: T#100ms or #CYCLE
                                                    where CYCLE = input parameter of the block in
                                                    which LP_SCHED instruction is called.
        COM_RST:                                    Here, the LP_SCHED instruction is notified
                                                    whether or not an initialization pass of the
                                                    called control loop is to take place.
                                                    Example: FALSE or #COM_RST where COM_RST =
                                                    input parameter of the block in which the
                                                    LP_SCHED instruction is called.
        DB_NBR                                      Here, the number of the "DB_LOOP" DB that the
                                                    LP_SCHED instruction is to process is called.
                                                    Example: "DB_LOOP" where DB_LOOP = name of DB
                                                    assigned in the symbol table.
    A "DB_LOOP".LOOP_DAT[1].ENABLE
    SPBN M002                                       Control loop call, if ENABLE = TRUE
    CALL FBx,DBy
      COM_RST:= "DB_LOOP".LOOP_DAT[1].COM_RST
      :                                             Formal operand list
      :                                             Formal operand list
      CYCLE:= "DB_LOOP".LOOP_DAT[1].CYCLE
      :                                             Formal operand list
      :                                             Formal operand list
    CLR
    = "DB_LOOP".LOOP_DAT[1].ENABLE                  Reset ENABLE bit
M002:                                               Continuing in the program, e.g., conditional
                                                    call of the next control loop FB
    :
```

Startup

When COM_RST = TRUE, the following pre-assignments are set:

Current control loop number: ALP_NBR = 0

The call data of all control loops up to **GLB_NBR** is pre-assigned as follows:

Enable:	ENABLE =	NOT MAN_DIS
Sampling time:	CYCLE =	GV(MAN_CYC)
Restart:	COM_RST =	TRUE
Local call number:	ILP_COU =	0
GV(MAN_CYC) = MAN_CYC is rounded to an integer multiple of TM_BASE*GLP_NBR		

During CPU startup, you must call LP_SCHED from the associated startup OB and, at the same time, assign the value TRUE to the COM_RST input. Then call PID_CP or PID_ES conditionally to initialize the controllers.

In the cyclic interrupt OB, you must assign the value FALSE to COM_RST again.

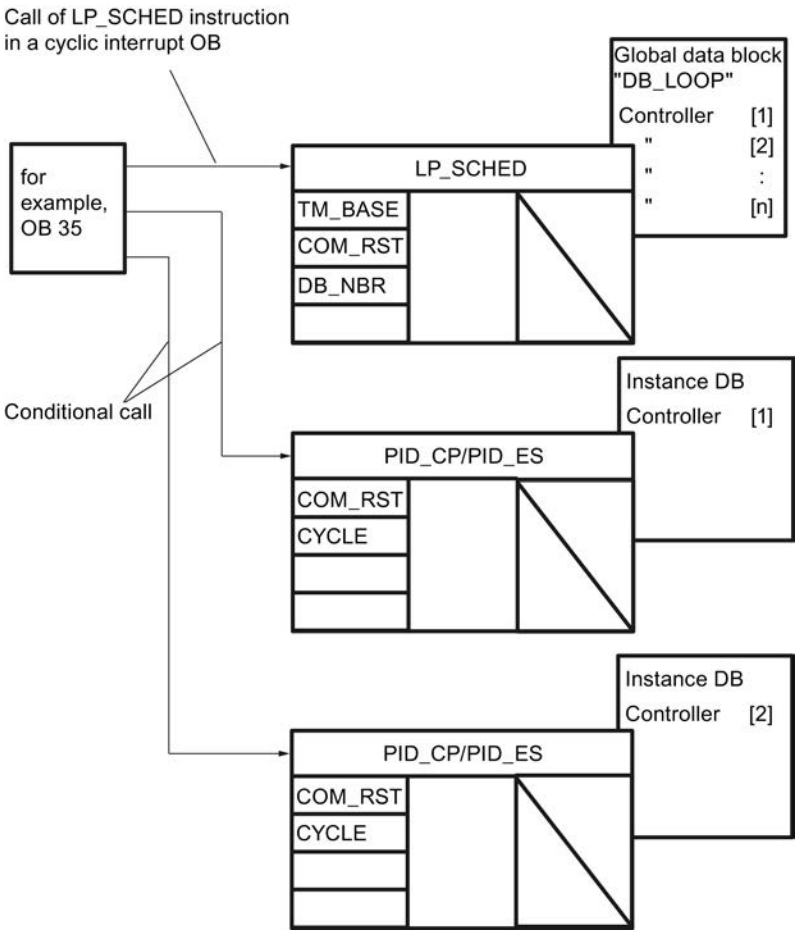
Operating principle of LP_SCHED

Call processing

A controller should be processed if the respective ENABLE bit in the DB_LOOP data block has the value TRUE. This bit was written previously by LP_SCHED. If the controller was processed, you must assign the value FALSE to the ENABLE bit after processing is complete.

When the controller is called, you must interconnect its input parameters COM_RST and CYCLE with the variables COM_RST[x] and CYCLE[x] of the DB_LOOP data block. CYCLE[x] contains the actual sampling time of the controller x and is written by LP_SCHED during each cycle.

During operation, you can manually disable the call of individual controllers and also reset individual controllers.



Parameter assignment of the DB_LOOP data block

The parameters of the controller call distributor must be assigned in the DB editor.

You must assign parameters to the following variables in the DB_LOOP data block:

- GLP_NBR: Number of controllers whose call is managed by LP_SCHED (max. 256)
- MAN_CYC[x], x = 1 ... GLP_NBR: The desired sampling time for the individual controllers. Note here that the condition indicated below for MAN_CYC[x] applies to each controller. Otherwise, the assigned sampling time cannot be guaranteed.

The following variables can change during operation:

- MAN_DIS[x] is used to disable the call of control loop x during operation.
- MAN_CRST[x] is used to initiate an initialization pass for control loop x during operation. The change becomes effective with the next call of the instruction LP_SCHED.

The following variables are written by LP_SCHED:

- LP_SCHED enters the call condition for the controller x in the variable ENABLE[x].
- The ALP_NBR and ILP_COU[x] variables are internal variables of the LP_SCHED instruction. You can use them for monitoring.

The COM_RST[x] and CYCLE[x] variables are interconnected with the input parameters COM_RST and CYCLE of the instructions PID_CP and PID_ES.

Sampling time MAN_CYC[x]

LP_SCHED can process a maximum of one controller per call. For this reason, the time

TM_BASE * GLP_NBR, until all controllers have been processed for the first time. Therefore, when you assign parameters for the desired sampling times MAN_CYC[x], you must adhere to the following condition for each controller:

The sampling time of controller x must be an integer multiple of the product obtained from the time base and the number of controllers to be processed.

$$\text{MAN_CYC[x]} \overset{!}{=} \text{GV (TM_BASE * GLP_NBR), x = 1, ... GLP_NBR}$$

The actual sampling time CYCLE[x] of the control loop x is determined as follows by LP_SCHED in each cycle from MAN_CYC[x]:

- If you adhere to this rule, the actual sampling time CYCLE[x] will be identical to the sampling time MAN_CYC[x] you specified previously.
- If you do not adhere to this rule, CYCLE[x] will contain the value that results if MAN_CYC[x] is rounded down to the next integer multiple of TM_BASE * GLP_NBR.

Adding additional control loops

If you want to insert one or more control loops into the DB_LOOP data block, increase the start value for GLP_NBR and expand the LOOP_DAT[1...GLP_NBR] array.

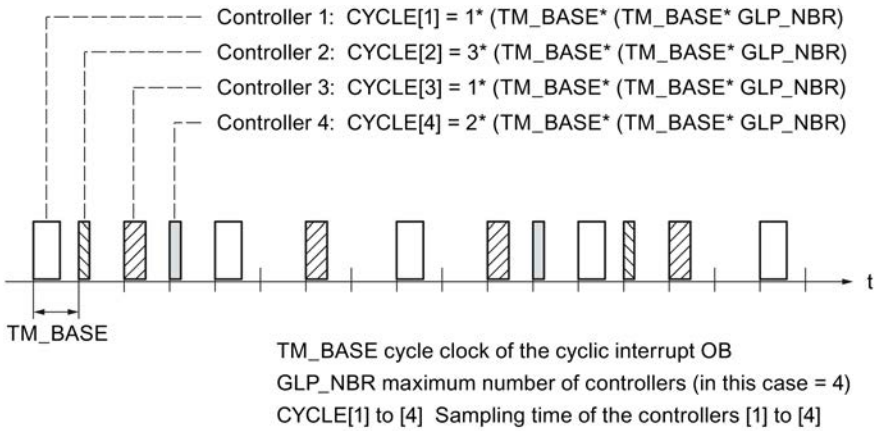
Check the desired sampling time MAN_CYC[x] for each controller. At the same time, note the condition for MAN_CYC.

Pulse generator in conjunction with LP_SCHED

If you have activated the pulse generator for the continuous controller PID_CP, the pulse pattern width CYCLE_P with the LOOP_DAT[x].CYCLE parameter must be interconnected instead of the CYCLE parameter.

Example of a loop schedule

The following example shows the call order of four controllers in a cyclic interrupt OB. A maximum of one controller is processed per unit of the time base TM_BASE. The call order is based on the order of the controllers in the DB_LOOP data block.



Call of more than one controller per cycle

If more than one controller is to be processed in a cycle, LP_SCHED may then be called multiple times. All calls of the LP_SCHED instruction must be made before calling the controller. In the input parameter TM_BASE, you must then enter the cycle clock of the cyclic interrupt OB, divided by the number of LP_SCHED calls.

Example: LP_SCHED will be called in OB 35 twice; OB 35 will be processed every 100 ms. The input parameter TM_BASE will be configured with 50 ms.

Execution times

The sum of the runtimes of the LP_SCHED instruction and the controller which can be processed in one cycle must not be greater than the cycle clock of the cyclic interrupt OB.

Manipulations during operation

The following changes may be made in the "DB_LOOP" data block during operation, provided that only the respective parameter is changed and the complete data block not downloaded to the CPU:

- Disabling individual controllers
  Writing the TRUE value to the MAN_DIS[x] variable disables the processing of controller x during operation. LP_SCHED does not set the ENABLE bit of this controller to TRUE until you write the value FALSE to MAN_DIS[x] again.
- Initializing a control loop
  You can restart an individual controllers by writing the value TRUE to the MAN_CRST[x] variable: In this case, LP_SCHED writes the value TRUE to the COM_RST[x] variable the next time controller x is processed. During the next subsequent processing of this control loop, LP_SCHED writes the value FALSE to the MAN_CRST[x] and COM_RST[x] variables.
- Change in the sampling time of a control loop
  You can change the MAN_CYC[x] parameter during operation:

Note

If the entire DB_LOOP data block is downloaded again to the CPU without requiring a CPU startup, you must pre-assign the internal control loop counters ILP_COU[x], x = 1, ... GLP_NBR and the number of the current control loop ALP_NBR with zero.

Monitoring the instruction LP_SCHED

LP_SCHED enters the number of the controller that is to be processed next in the ALP_NBR variable of the DB_LOOP data block. The number of the respective control loop results from the positioning of its call data in the sequence of the entries in the data block.

The ILP_COU[x] variable is the internal counter. It receives the time period until the next call of the corresponding controller. The time unit of ILP_COU is the product obtained from the time base TM_BASE and the number of controllers GLP_NBR. If ILP_COU = 0, LP_SCHED sets the ENABLE bit of the corresponding controller.

When control loops are not called contrary to expectation

If LP_SCHED is called but individual control loops are not processed, this can be caused by the following:

- Writing the value TRUE to the MAN_DIS[x] variable disables the processing of control loops during operation.
- The number of controllers to be processed by LP_SCHED specified in the GLP_NBR parameter is too low.
- The sampling times MAN_CYC[x] of the individual control loops that you have specified must not be less than the product obtained from the time base TM_BASE and the number of controllers GLP_NBR. Controllers in which this condition is not met will not be processed.

Input parameters LP_SCHED

Table 9-210

Parameter	Data type	Range of values	Preassigned value	Explanation
TM_BASE	TIME	≥ 20 ms (S7-300) ≥ 5 ms (S7-400)	100 ms	Time base (Cycle clock of cyclic interrupt level in which LP_SCHED is called) Permitted values are
COM_RST	BOOL		FALSE	Complete restart (Complete restart routine of LP_SCHED will be processed)
DB_NBR	BLOCK_DB		DB 1	Data block number (DB with the call data of the control loop)

Global data block for DB_LOOP

Parameter	Data type	Preassigned value	Description
GLP_NBR	INT	2	Maximum number of controllers 1 - 256
ALP_NBR	INT	0	Number of current controllers 0 - 256
LOOP_DAT	Array [1...GLP_NBR] of Struct		Structure for the data of the individual controllers. The array must be declared by 1...GLP_NBR.
LOOP_DAT[1].MAN_CYC	TIME	T#1s	Data controller 1: Manual sampling time >= 20 ms (S7-300) >= 5 ms (S7-400)
LOOP_DAT[1].MAN_DIS	BOOL	FALSE	Data controller 1: Disable manual controller call
LOOP_DAT[1].MAN_CRST	BOOL	FALSE	Data controller 1: Set manual restart
LOOP_DAT[1].ENABLE	BOOL	FALSE	Data controller 1: Controller enable
LOOP_DAT[1].COM_RST	BOOL	FALSE	Data controller 1: Complete restart
LOOP_DAT[1].ILP_COU	INT	0	Data controller 1: Internal counter
LOOP_DAT[1].CYCLE	TIME	T#1s	Data controller 1: Sampling time >= 20 ms (S7-300) >= 5 ms (S7-400)
LOOP_DAT[2].MAN_CYC			Data controller 2: Manual sampling time
...			

Modular PID Control

General

Conventions for parameter and block names in the block diagram

Not more than 8 characters were selected for the parameter designations.

The following conventions apply for the designation of the parameters:

Initial character:	
Q	General output of the BOOL type (Boolean variable)
SP	Reference variable
PV	Measured value or controlled variable
LMN	Manipulated variable or analog output signal to be output
DISV	Disturbance variable
Subsequent characters:	
MAN	Manual value
INT	Internal
EXT	External
_ON	Boolean variable for activating a function

Call data

Most Modular PID Control instructions require control-loop-specific call data such as the restart bit and sampling time. These values are transferred to the inputs COM_RST and CYCLE.

Remarks on the block parameter bars (input, output, and in/out parameters)

- **Preassigned value:** The preassigned values for the creation of an instance are displayed here.
- **Range of values**: The values at the input parameter should not exceed the permitted range of values. The range is **not** checked in the course of the block execution. If a range of engineering values is specified, a physical variable whose value should lie between approx. $\pm 10^{6}$ is meant.

A_DEAD_B

Description A_DEAD_B

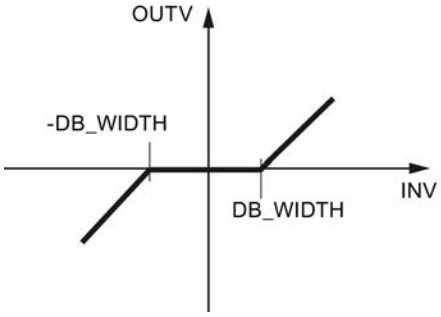
If the process value is affected by noise and the controller is optimally set, the noise also affects the controller output. This results in increased wear of the actuator through a high frequency of operation (step controllers). Oscillation of the controller output is avoided by suppression of the noise component.

Operating principle

The instruction filters high-frequency interfering signals from the control deviation and produces a deadband around the zero point. If the input variable lies within this range, zero is output at the output. The range width is adapted automatically to the amplitude of the interfering signal.

The instruction operates in accordance with the function:

OUTV = INV + DB_WIDTH	for	INV < -DB_WIDTH
OUTV = 0.0	for	-DB_WIDTH ≤ INV ≤ +DB_WIDTH
OUTV = INV - DB_WIDTH	for	INV > +DB_WIDTH



Adaptation of the deadband

The adaptation of the deadband can be activated and deactivated.

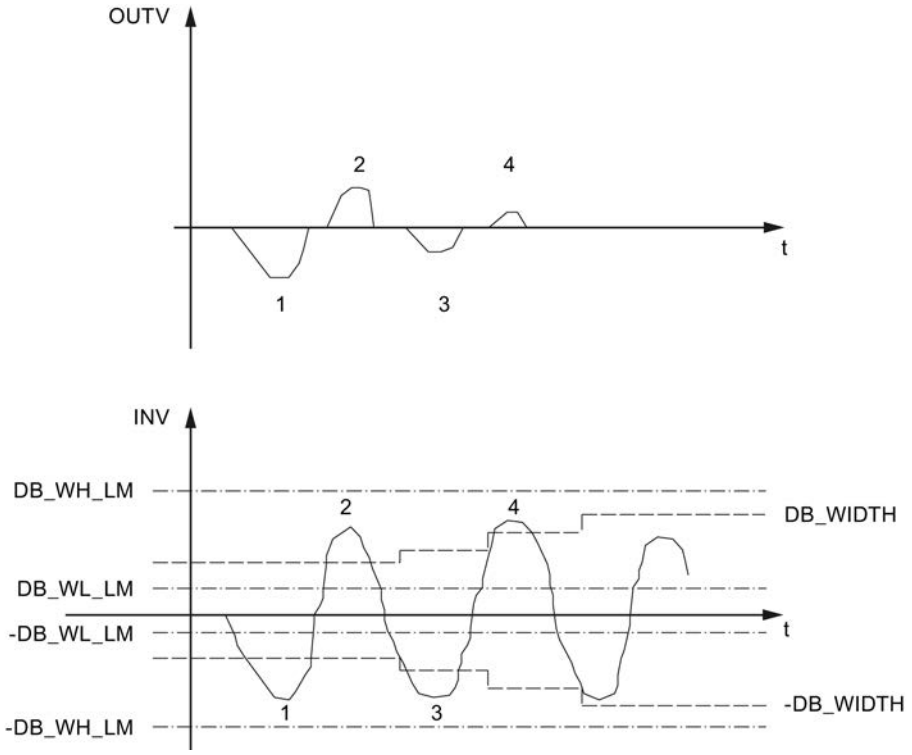
- Adaptation off
  If the adaptation is deactivated (ADAPT_ON = FALSE), the last DB_WIDTH value is frozen as the effective deadband width DB_WIDTH.
- Adaptation on
  With ADAPT_ON = TRUE, an adaptation algorithm that calculates the effective deadband width can be activated. The effect is that the deadband width is adapted to the amplitude of the noise signal that is superimposed on the input variable, so that the noise component is also suppressed at a fluctuating noise amplitude.
  In the case of an acyclic block call, the adaptation has to be deactivated (ADAP_ON = FALSE).

If adaptation is activated during commissioning and a stable deadband width is set after a certain time, the adaptation can be deactivated. The deadband width set through the adaptation remains in force as long as a complete restart is not carried out.

Adaptation on

To ensure stability, the effective deadband width DB_WIDTH is subject to a low limit, which is represented by the DB_WL_LM value that can be specified at the input bar. If input signal INV superimposed by the measurement noise exceeds the currently specified deadband width in

the negative (1), positive (2), and again negative direction (3) within the period 1/CRIT_FRQ, the effective deadband width is increased by the amount 0.1.



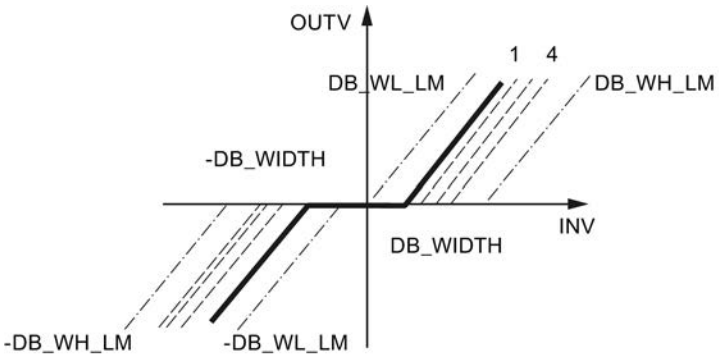
The procedure is started if the limit is exceeded positively or negatively. Whenever the opposing deadband width is exceeded again (3 -> 4) within half the cycle duration, it is increased again by 0.1. This procedure is repeated until the deadband width agrees with the amplitude of the measurement noise. In order not to indiscriminately suppress input signals of any size, the effective deadband width is subject to a high limit, which is represented by input DB_WH_LM. If, on the other hand, the deadband width is not exceeded within time RET_FAC * 1/CRIT_FRQ, it is reduced by the amount 0.1.

CRIT_FRQ specifies the limit frequency from which a signal component is regarded as a disturbance. It is subject to a high limit and low limit as follows:

0.01 ≤ CRIT_FRQ ≤ 1/(3*CYCLE) where CYCLE: Sampling time in sec

The RET_FAC parameter specifies the multiple of 1/CRIT_FRQ based on which the deadband width is reduced again.

The adaptation logic only functions if the input variable without disturbance component lies near zero.



Complete restart

At a complete restart, OUTV=0.0 is set, and the effective deadband width DB_WIDTH = DB_WL_LM is set

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

See also

Generating and processing the control deviation (Page 9757)

Input parameters A_DEAD_B

Parameter	Data type	Preassigned value	Description
INV	REAL	0.0	Input variable Engineering value range
DB_WH_LM	REAL	5.0	High limit of the deadband width Engineering value range DB_WH_LM > DB_WL_LM
DB_WL_LM	REAL	1.0	Low limit of the deadband width Engineering value range DB_WL_LM < DB_WH_LM
CRIT_FRQ	REAL	0.1	Limit frequency CRIT_FRQ ≥ 0.01 and CRIT_FRQ ≤ 1/(3 * CYCLE)
RET_FAC	INT	1	Return factor RET_FAC ≥ 1
ADAPT_ON	BOOL	FALSE	Activate adaptation algorithm

Parameter	Data type	Preassigned value	Description
COM_RST	BOOL	FALSE	Restart
CYCLE	TIME	T#1s	Sampling time CYCLE ≥ 1ms

Output parameters A_DEAD_B

Parameter	Data type	Preassigned value	Description
OUTV	REAL	0.0	Output variable
DB_WIDTH	REAL	0.0	Effective deadband width

CRP_IN

Description CRP_IN

This instruction adapts the range of values of the analog I/O to the internal representation of the modular control. The instruction can be called in the process value branch, for example.

Operating principle

CRP_IN converts an input value in I/O format into a scaled floating-point value of the modular control.

I/O value	Output value as a %
32767	118.515
27648	100.000
1	0.003617
0	0.000
-1	-0.003617
-27648	-100.000
-32768	-118.519

The floating-point value can be adapted by means of a scaling factor and an offset. The output results from:

OUTV = INV_PER * 100/27648 * FACTOR + OFFSET

Note

A check for positive/negative overflow is not performed.

Using a commissioning value

It is possible to switch over to a commissioning value for purposes of commissioning or testing or in case of I/O faults. If START_ON = TRUE, the value in STARTVAL is output at output OUTV.

Complete restart

The block does not have a complete restart routine.

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

See also

Processing the process value (Page 9755)

Applying the disturbance variable (Page 9759)

Processing position feedback (Page 9768)

Input parameters CRP_IN

Parameter	Offset	Data type	Preassigned value	Description
INV_PER	0.0	WORD	0	Input variable for I/O Engineering value range
FACTOR	2.0	REAL	1.0	Scaling factor
OFFSET	6.0	REAL	0.0	Offset Engineering value range
START_ON	10.0	BOOL	TRUE	Activate commissioning value
STARTVAL	12.0	REAL	0.0	Commissioning value Engineering value range

Output parameters CRP_IN

Parameter	Offset	Data type	Preassigned value	Description
OUTV	16.0	REAL	0.0	Output variable

CPR_OUT

Description CRP_OUT

This instruction adapts a floating-point value of the modular control to the I/O format.

Operating principle

CRP_OUT converts an input value (scaled floating-point value of the modular control) into the I/O format of the analog I/O.

Input value in %	I/O value
118.515	32767
100.000	27648
0.003617	1
0.000	0
-0.003617	-1
-100.000	-27648
-118.519	-32768

The floating-point value can be adapted by means of a scaling factor and an offset. The output results from:

OUTV_PER = (INV * FACTOR + OFFSET) * 27648/100

Note

A check for positive/negative overflow is not performed.

Complete restart

The block does not have a complete restart routine.

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

See also

Creating a continual controller (Page 9762)

Input parameters CRP_OUT

Parameter	Offset	Data type	Preassigned value	Description
INV	0.0	REAL	0.0	Input variable Engineering value range
FACTOR	4.0	REAL	1.0	Scaling factor
OFFSET	8.0	REAL	0.0	Offset Engineering value range

Output parameters CRP_OUT

Parameter	Offset	Data type	Preassigned value	Description
OUTV_PER	12.0	WORD	0	Output variable for I/O

DEAD_T

Description DEAD_T

This instruction is used in ratio control systems in which the individual components on transport routes of different lengths are brought together.

Operating principle

The instruction delays the output of an input value by an assignable time (dead time). In the case of online changes in the dead time steps can occur at the output value,

The input values are stored temporarily in a global data block.

The global data block must contain an Array [0..n_{max}] of real.

n_{max} is determined by the dead time DEAD_TM and sampling time CYCLE of the DEAD_T. instruction. The following applies:

- DEAD_TM must be an integer multiple of CYCLE.
- n_{max}=DEAD_TM/CYCLE

In cycle n, the saved input value INV[n] is read from the global data block and output to OUTV. INV[n] is overwritten with the current input value. Counter n is increased by 1. When n = n_{max} , the counter is reset to 0. You specify the number of the global block in input parameter DB_NBR.

Example

At a sampling time of CYCLE=1s and a dead time of DEAD_TM=5s, 5 input values must be buffered. The data area must have a length of 20 bytes.

The data in the global data block are processed like in a ring buffer.

Cycle no.	Counter n	INV of instruction	Variable in global data block	Value of INVI[n] at start of cycle	OUTV of instruction	Value of INVI [n] at end of cycle
0	0	80.0	INVI[0]	0.0	0.0	80.0
1	1	100.0	INVI[1]	0.0	0.0	100.0
2	2	100.0	INVI[2]	0.0	0.0	100.0
3	3	80.0	INVI[3]	0.0	0.0	80.0
4	4	0.0	INVI[4]	0.0	0.0	0.0
5	0	0.0	INVI[0]	80.0	80.0	0.0
6	1	0.0	INVI[1]	100.0	100.0	0.0
7	2	0.0	INVI[2]	100.0	100.0	0.0
8	3	10.0	INVI[3]	80.0	80.0	10.0
9	4	100.0	INVI[4]	0.0	0.0	100.0

The value of INV in cycle 0 is not output at OUTVuntil cycle 5. With a sampling time of 1 s, the output is thus delayed by the dead time of 5 s.

Tracking

When TRACK=TRUE, the input value is output at OUTV without a delay. Buffering of the input values is not interrupted so that, after tracking has been deactivated, the input values can continue to be output after the set dead time. On a falling edge at TRACK, OUTV jumps to INV[DEAD_TM]!

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

Note

The instruction does not check whether a global DB with the number DB_NBR actually exists or whether the parameters DEAD_TM (dead time) and CYCLE (sampling time) are suitable for the data block length. If the parameter assignment is incorrect, the CPU changes to the STOP mode with the message "Internal system error".

Restart

During a restart, all saved input values are deleted and OUTV=0.0 is output.

Input parameters DEAD_T

Parameter	Offset	Data type	Preassigned value	Description
INV	0.0	REAL	0.0	Input variable Engineering value range
DB_NBR	4.0	BLOCK_DB	DB 1	Data block number
DEAD_TM	6.0	TIME	10s	Dead time DEAD_TM ≥ CYCLE
TRACK	10.0	BOOL	FALSE	Tracking OUTV = INV
COM_RST	10.1	BOOL	FALSE	Complete restart
CYCLE	12.0	TIME	1s	Sampling time CYCLE ≥ 1ms

Output parameters DEAD_T

Parameter	Offset	Data type	Preassigned value	Description
OUTV	16.0	REAL	0.0	Output variable

Shared data block DEAD_T

Parameter	Data type	Preassigned value	Description
INVI	Array[0..n_{max}] of real	0.0	Input value [n] Engineering value range

DEADBAND

Description DEADBAND

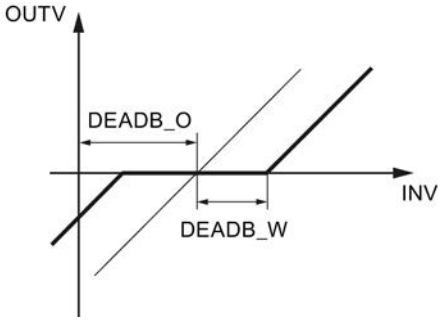
If the process value is affected by noise and the controller is optimally set, the noise also affects the controller output. This results in increased wear of the actuator through a high frequency of operation (step controllers). Oscillation of the controller output is avoided by suppression of the noise component.

Operating principle

The DEADBAND instruction suppresses minor fluctuations of input variable INV around a defined zero point. Outside these fluctuations, output variable OUTV rises proportionally to the input variable. It operates in accordance with the function:

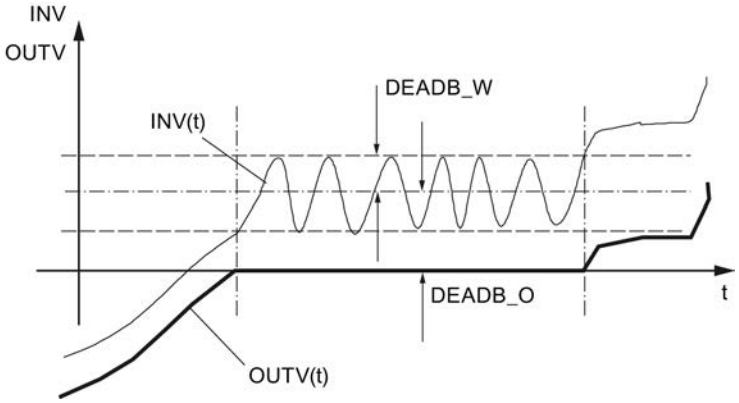
OUTV = INV + DEADB_W - DEADB_O	for	INV < DEADB_O - DEADB_W
OUTV = 0.0	for	DEADB_O - DEADB_W ≤ INV
	and	INV ≤ DEADB_O + DEADB_W
OUTV = INV - DEADB_W - DEADB_O	for	INV > DEADB_O + DEADB_W

The signal corruption corresponds to the value DEADB_W. The center of the deadband is defined with DEADB_O.



Deadband width DEADB_W and deadband offset DEADB_O can be assigned. When used for generating the control deviation, offset DEADB_O=0.0 must be set.

The following figure shows the bypassing of a noise component with offset



Complete restart

The block does not have a complete restart routine.

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check. Only positive values are permitted for the deadband width.

See also

Generating and processing the control deviation (Page 9757)

Input parameters DEADBAND

Parameter	Offset	Data type	Preassigned value	Description
INV	0.0	REAL	0.0	Input variable Engineering value range
DEADB_W	4.0	REAL	1.0	Deadband width Engineering value range and ≥ 0.0
DEADB_O	8.0	REAL	0.0	Deadband offset Engineering value range

Output parameters DEADBAND

Parameter	Offset	Data type	Preassigned value	Description
OUTV	12.0	REAL	0.0	Output variable

DIF

Description DIF

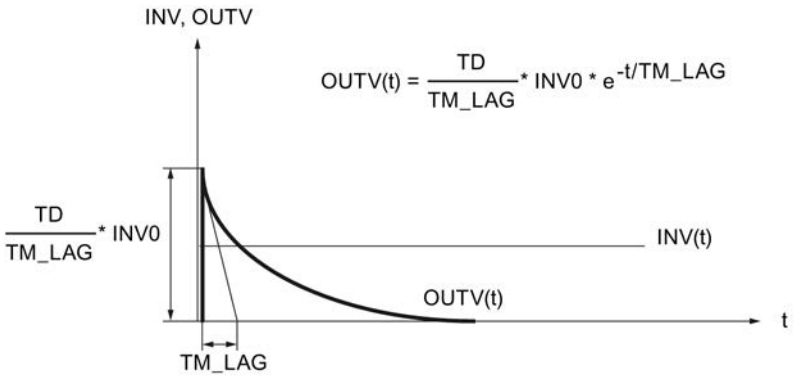
Process variables are differentiated dynamically. Thus, for example, the velocity can be calculated from the distance traversed. The differentiator can be used for disturbance variable compensation, for feedforward control, and to structure a controller.

Operating principle

This instruction differentiates the input variable with respect to time and smoothes the signal with a first-order delay element. It operates in the Laplace domain according the following transfer function:

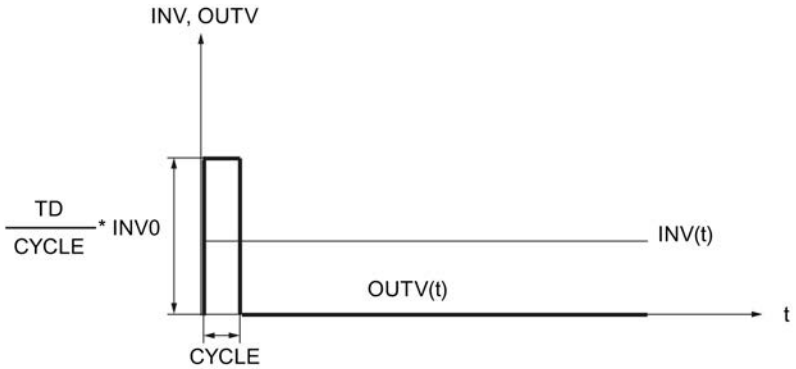
- OUTV(s) / INV(s) = TD / (1+TM_LAG*s)

The time response is defined by the derivative-action time TD and the time delay TM_LAG. The following figures show the step response of DIF.



TD	Derivative-action time
TM_LAG	Time delay
INV0	Input step height
INV	Input variable
OUTV	Output variable

If TM_LAG ≤CYCLE/2 is assigned, DIF works without a delay. An input step-change is applied to the output with the factor TD/CYCLE. After one cycle the output returns to 0.0.



Block-internal limits

The derivative-action time is subject to a low limit equal to the sampling time. The time delay is subject to a low limit equal to half the sampling time.

TD_{intern} = CYCLE	for	TD < CYCLE
TM_LAG_{intern} = CYCLE/2	for	TM_LAG < CYCLE/2

The other values of the input parameters are not limited in the instruction. There is no parameter check.

Complete restart

During a complete restart, all the signal outputs are set to 0. The differentiator is preassigned internally with the current input value INV. The transition to normal operation is thus smooth at a constant input variable.

See also

Applying the disturbance variable (Page 9759)

Input parameters DIF

Parameter	Offset	Data type	Preassigned value	Description
INV	0.0	REAL	0.0	Input variable Engineering value range
TD	4.0	TIME	T#25s	Derivative-action time ≥ CYCLE
TM_LAG	8.0	TIME	T#5s	Time delay
COM_RST	12.0	BOOL	FALSE	Complete restart
CYCLE	14.0	TIME	T#1s	Sampling time ≥ 1ms

Output parameters DIF

Parameter	Offset	Data type	Preassigned value	Description
OUTV	18.0	REAL	0.0	Output variable

ERR_MON

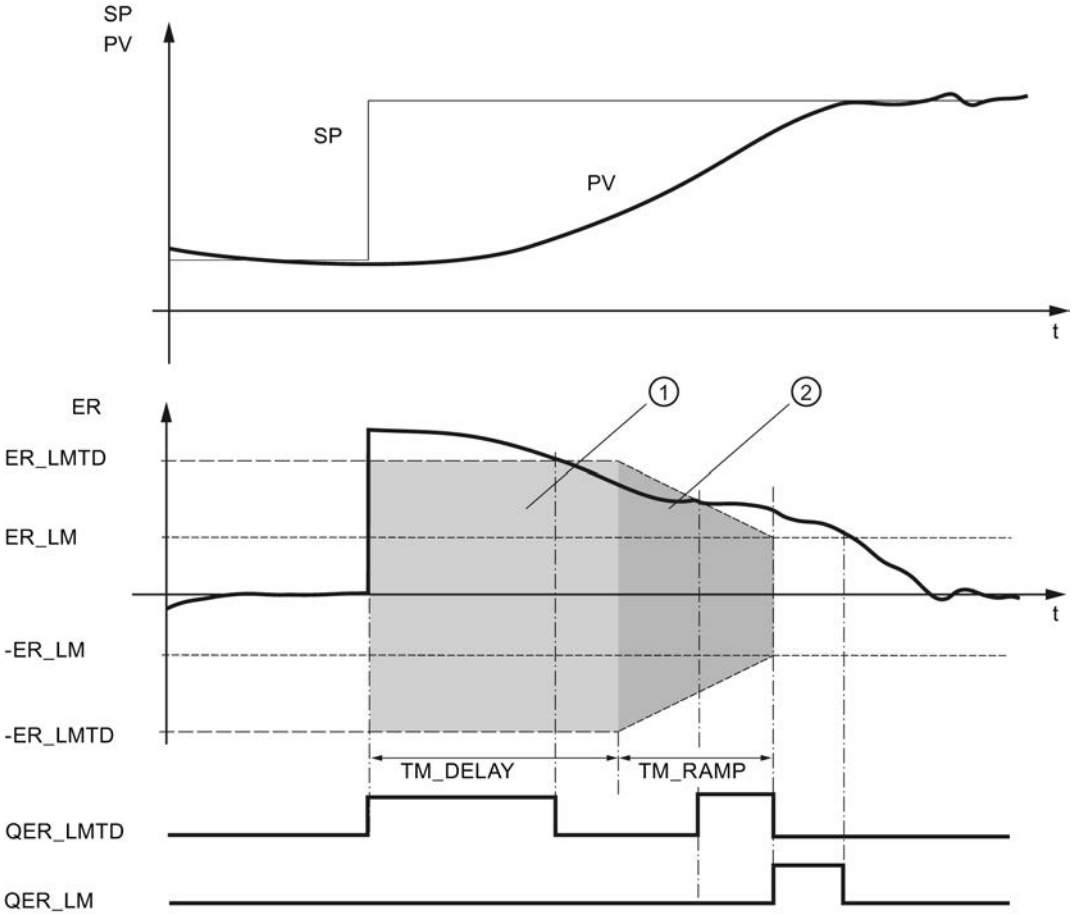
Description ERR_MON

This instruction is used for control deviation generation and monitoring.

Operating principle

The instruction calculates control deviation ER = SP - PV and monitors it for assignable limits. As a result of a setpoint change |ΔSP| > SP_DIFF, the response of limit value signal ER_LM is suppressed for an assignable delay time (TM_DELAY+TM_RAMP). During this time, ER is monitored for the higher limit value ER_LMTD. If ER_LMTD is exceeded, QER_LMTD=TRUE

is output. After the delay time has expired, ER_LMTD is transformed into ER_LM along a ramp. The slope of the ramp can be assigned in the TM_RAMP parameter.



PV:	Process value	TM_DELAY:	On delay of the monitoring signal
SP:	Setpoint		
ER:	Control deviation	TM_RAMP:	Ramp time constant
ER_LM:	Limit value of the control deviation	QER_LM:	Limit value of the control deviation triggered
ER_LMTD:	Limit value of the control deviation during On delay	QER_LMTD:	Limit value of the control deviation during On delay and ramp triggered
①	Suppression of the limit value signal	②	Ramp function

Complete restart

During a complete restart, alarm signals QER_LM and QER_LMTD and control deviation output ER are reset.

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

See also

Generating and processing the control deviation (Page 9757)

Input parameters ERR_MON

Parameter	Offset	Data type	Preassigned value	Description
ER_LM	0.0	REAL	10.0	Limit value of the control deviation Engineering value range and > 0.0 and < ER_LMTD
ER_LMTD	4.0	REAL	100.0	Limit value of the control deviation during On delay Engineering value range and > ER_LM
SP	8.0	REAL	0.0	Setpoint Engineering value range
PV	12.0	REAL	0.0	Controlled variable Engineering value range
SP_DIFF	16.0	REAL	10.0	Setpoint change Engineering value range and > 0.0
TM_DELAY	20.0	TIME	T#60s	On delay of the monitoring signal
TM_RAMP	24.0	TIME	T#60s	Ramp time constant
COM_RST	28.0	BOOL	FALSE	Complete restart
CYCLE	30.0	TIME	T#1s	Sampling time ≥ 1ms

Output parameters ERR_MON

Parameter	Offset	Data type	Preassigned value	Description
QER_LM	34.0	BOOL	FALSE	Limit value of the control deviation triggered
QER_LMTD	34.1	BOOL	FALSE	Limit value of the control deviation during On delay and the ramp triggered
ER	36.0	REAL	0.0	Control deviation

INTEG

Description INTEG

Process variables are integrated dynamically. Thus, for example, the distance traversed is calculated from the velocity. This instruction can be used to set up a controller with integral component.

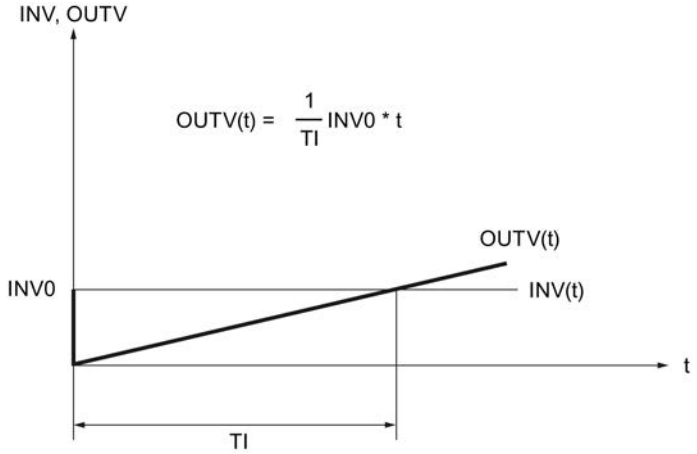
Operating principle

The instruction integrates the input variable with respect to time and limits the integral to a definable preassigned high and low limit. Limiting of the output variable is indicated by means of message bits.

The instruction contains the following functions:

Functions	DFOUT_ON	HOLD
Integrate	FALSE	FALSE
Freeze integral component	FALSE	TRUE
Preassign output	TRUE	Any

- Integrate
  During integration, the instruction works in accordance with the following transfer function:
  In the Laplace domain:
  OUTV(s) / INV(s) = 1 / (TI*s)
  The time response of the integral component is determined by the integral-action time TI.
  The associated step response is shown in the figure below.

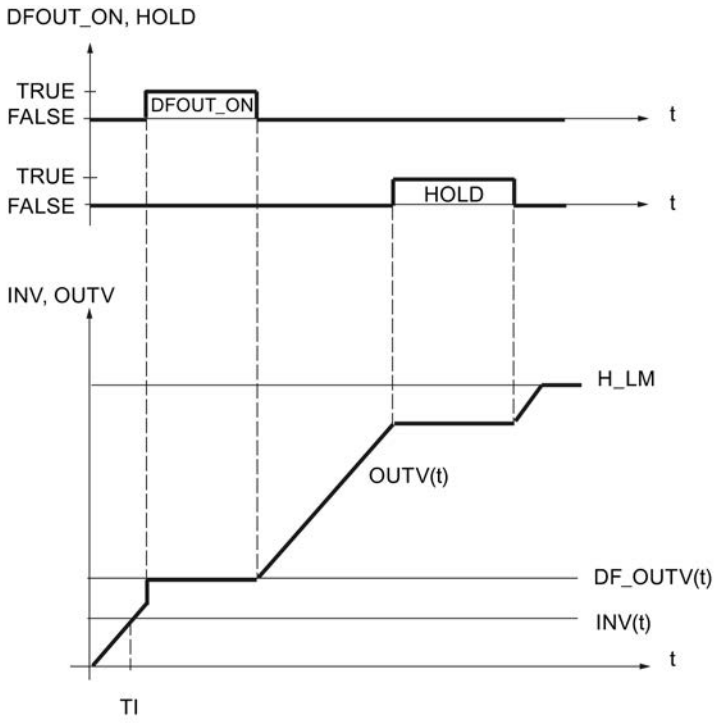


T1:	Integral-action time
INV0:	Input step height
t:	Time
INV:	Input variable
OUTV:	Output variable

The output as well as the memory of the integral component are limited based on the H_LM and L_LM parameters. If the output is outside the range L_LM...H_LM the message bits QH_LM or QL_LM are set to TRUE.

- Freeze integrator
  With HOLD = TRUE, the integral component remains at its current output value OUTV. On reset (HOLD = FALSE), integration is continued starting from the current output value OUTV.
- Preassign output
  If DFOUT_ON = TRUE is set, DF_OUTV is output at the output. Limit is effective. On reset (DF_OUTV_ON = FALSE), INTEG starts integrating from value DF_OUTV.

The following figure shows an example with DFOUT_ON, HOLD, and limiting



Complete restart

During a complete restart, output OUTV is reset to 0.0. If DFOUT_ON = TRUE is set, DF_OUTV is output. Limiting of the output and its display are also in effect when a complete restart occurs. On transition to normal operation, the instruction continues to integrate from OUTV.

If the integral component is to be started from a specific operating point on a complete restart, the operating point must be entered in input DF_OUTV. When the instruction is called in the complete restart routine, DFOUT_ON= TRUE must be set and reset back to DFOUT_ON= FALSE in the cyclic interrupt level.

Block-internal limits

The integral-action time is subject to a low limit equal to the sampling time.

TI_{intern} = CYCLE for TI < CYCLE

The other values of the input parameters are not limited in the instruction. There is no parameter check.

Input parameters INTEG

Parameter	Offset	Data type	Preassigned value	Description
INV	0.0	REAL	0.0	Input variable Engineering value range
H_LM	4.0	REAL	100.0	High limit Engineering value range H_LM > L_LM
L_LM	8.0	REAL	0.0	Low limit Engineering value range L_HM < H_LM
TI	12.0	TIME	T#25s	Integral-action time TI ≥ CYCLE
DF_OUTV	16.0	REAL	0.0	Default setting of output variable Engineering value range
HOLD	20.0	BOOL	FALSE	Freeze integral component
DFOUT_ON	20.1	BOOL	FALSE	Default setting of output variable On
COM_RST	20.2	BOOL	FALSE	Complete restart
CYCLE	22.0	TIME	T#1s	Sampling time CYCLE ≥ 1ms

Output parameters INTEG

Parameter	Offset	Data type	Preassigned value	Description
OUTV	26.0	REAL	0.0	Output variable
QH_LM	30.0	BOOL	FALSE	High limit triggered
QL_LM	30.1	BOOL	FALSE	Low limit triggered

LAG1ST

Description LAG1ST

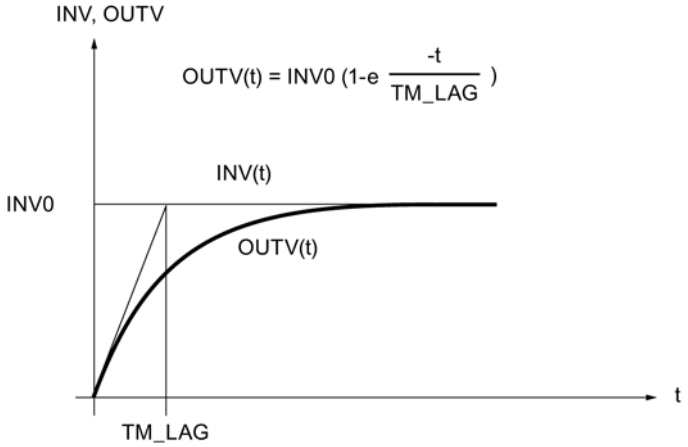
This instruction can be used as a first-order delay and smoothing element. The time delay is parameterizable.

Operating principle

The instruction smoothes the input variable according to the first-order delay and contains the following functions:

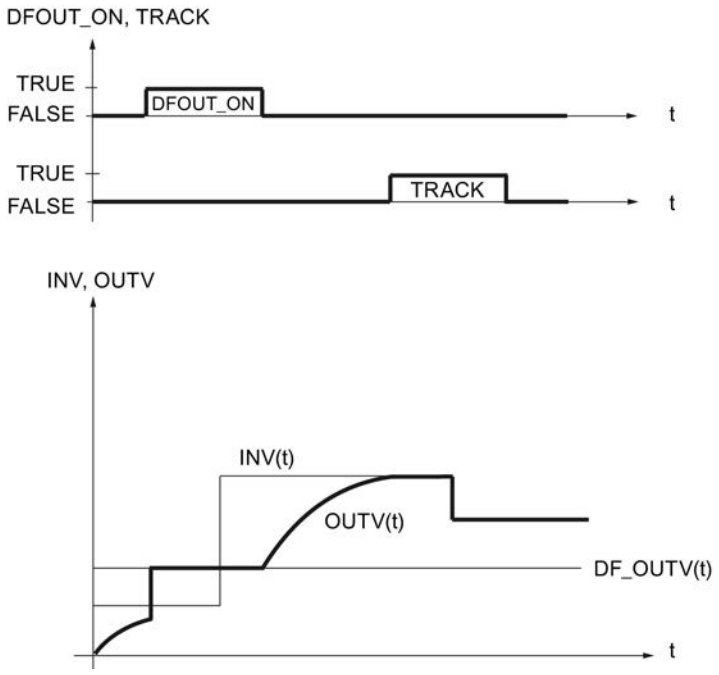
Functions	DFOUT_ON	TRACK
Smoothing	FALSE	FALSE
Tracking	FALSE	TRUE
Preassign output	TRUE	Any

- Smoothing
  With smoothing, the instruction operates in the Laplace domain in accordance with the following transfer function:
  OUTV(s) / INV(s) = 1 / (1+TM_LAG*s)
  The time response of the delay element is determined by the time delay TM_LAG. The associated step response is shown in the figure below.



INV	Input variable
OUTV	Output variable
INV0	Input step height
TM_LAG	Time delay

- Tracking
  With TRACK = TRUE, input value INV is switched through to output OUTV.
- Preassign output
  If DFOUT_ON = TRUE is set, DF_OUTV is output at the output. On reset (DF_OUTV_ON = FALSE), the delay element smoothes starting from value DF_OUTV.
  The following diagram shows the effect of DFOUT_ON and TRACK.



Complete restart

During a complete restart, output OUTV is reset to 0.0. If DFOUT_ON = TRUE is set, DF_OUTV is output. On transition to smoothing, the instruction continues to operate from OUTV.

Block-internal limits

The time delay is subject to a low limit equal to half the sampling time.

TM_LAG_{intern} = CYCLE/**2** for TM_LAG < CYCLE/**2**

The other values of the input parameters are not limited in the instruction. There is no parameter check.

See also

Processing setpoints (Page 9753)

Processing the process value (Page 9755)

Applying the disturbance variable (Page 9759)

Input parameters LAG1ST

Parameter	Offset	Data type	Preassigned value	Description
INV	0.0	REAL	0.0	Input variable Engineering value range
TM_LAG	4.0	TIME	T#25s	Time delay
DF_OUTV	8.0	REAL	0.0	Default setting of output variable Engineering value range
TRACK	12.0	BOOL	FALSE	Tracking OUTV=INV
DFOUT_ON	12.1	BOOL	FALSE	Default setting of output variable On
COM_RST	12.2	BOOL	FALSE	Complete restart
CYCLE	14.0	TIME	T#1s	Sampling time ≥1ms

Output parameters LAG1ST

Parameter	Offset	Data type	Preassigned value	Description
OUTV	18.0	REAL	0.0	Output variable

LAG2ND

Description LAG2ND

This instruction is used to simulate controlled system components for feedforward controls and two-circuit control systems.

Operating principle

The instruction implements an oscillating second-order delay element and contains the following functions:

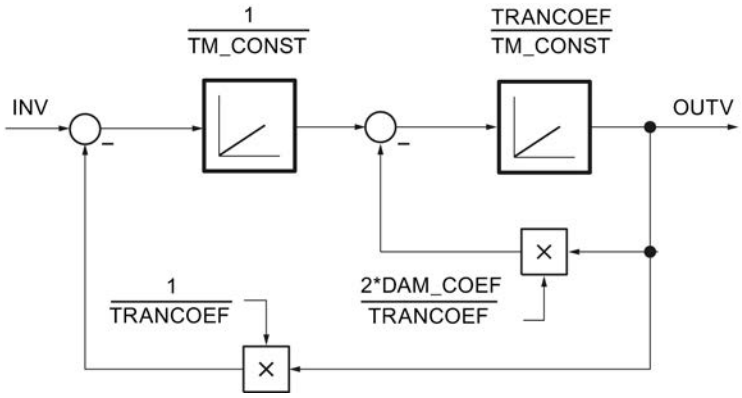
Functions	DFOUT_ON	TRACK
Transferring	FALSE	FALSE
Tracking	FALSE	TRUE
Preassign output	TRUE	Any

- **Transferring**
  The transfer function in the Laplace domain is:
  OUTV(s) / INV(s) = TRANCOEF / (1 + 2*DAM_COEF*TM_CONST*s + TM_CONST^2*s2)
  If DAM_COEF >= 1 (aperiodic case), the transfer element can be represented in a series connection of two first-order delay elements.
  OUTV(s) / INV(s) = TRANCOEF / (1 + T1*s) * 1 / (1 + T2 * s)
  The time constants are converted as follows:

  $$T1 = TM_CONST\ (DAM_COEF + \sqrt{DAM_COEF^2 - 1}\ )$$

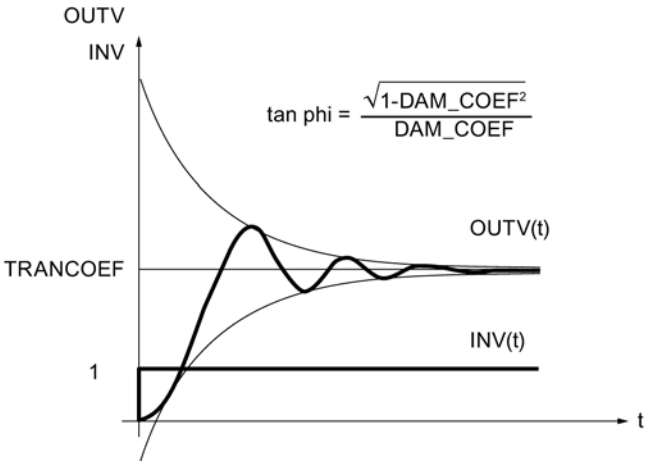
  $$T2 = TM_CONST\ (DAM_COEF - \sqrt{DAM_COEF^2 - 1}\ )$$

  The figure below shows the block diagram.



  The following figure shows the step response of the oscillating LAG2ND element.

$$OUTV(t) = TRANCOEF + \frac{TRANCOEF}{\sqrt{1-DAM_COEF^2}} \cdot e^{-\frac{DAM_COEF}{TM_CONST} t} \sin\left(\frac{\sqrt{1-DAM_COEF^2}}{TM_CONST} t - phi\right)$$



INV	Input variable
OUTV	Output variable
TRANCOEF	Transfer constant
DAM_COEF	Damping constant
phi	Supplement angle

- **Tracking**
  If TRACK =TRUE, OUTV = INV; the internal historical values are set to INV.
- **Preassign output**
  If DFOUT_ON =TRUE, DF_OUTV is output; the internal historical values are set to DF_OUTV. DFOUT_ON has higher priority than TRACK.

Complete restart

During a complete restart, output OUTV is reset to 0.0. If DFOUT_ON = TRUE, DF_OUTV is output.

Block-internal limits

The time constant TM_CONST is subject to a low limit equal to half the sampling time.

$TM_CONST_{intern} = CYCLE/2$ for $TM_CONST < CYCLE/2$

The other values of the input parameters are not limited in the instruction. There is no parameter check.

Input parameters LAG2ND

Parameter	Offset	Data type	Preassigned value	Description
INV	0.0	REAL	0.0	Input variable Engineering value range
TM_CONST	4.0	TIME	T#10s	Time constant
DAM_COEF	8.0	REAL	1.0	Damping constant
TRANCOEF	12.0	REAL	1.0	Transfer constant
DF_OUTV	16.0	REAL	0.0	Default setting of output variable Engineering value range
DFOUT_ON	20.0	BOOL	FALSE	Default setting of output variable On
TRACK	20.1	BOOL	FALSE	Tracking OUTV=INV
COM_RST	20.2	BOOL	FALSE	Complete restart
CYCLE	22.0	TIME	T#1s	Sampling time ≥ 1ms

Output parameters LAG2ND

Parameter	Offset	Data type	Preassigned value	Description
OUTV	26.0	REAL	0.0	Output variable

LIMALARM

Description LIMALARM

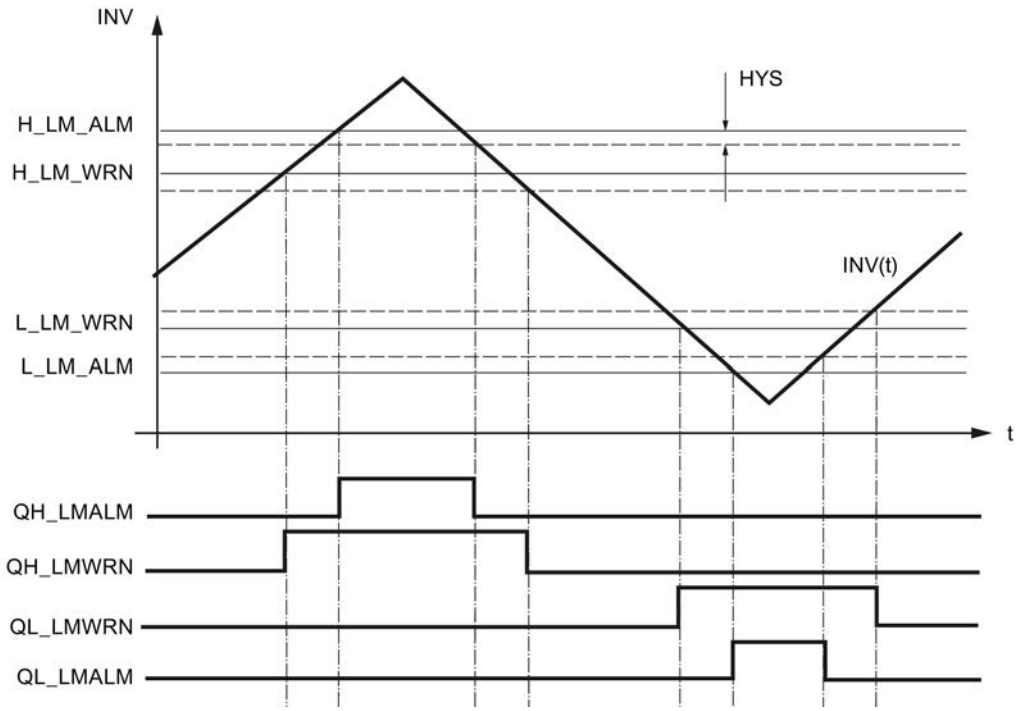
Illegal or dangerous states can occur in a system if process variables (for example, motor speed, temperature or pressure) exceed or fall below critical values. Such limit violations must be detected and signaled to allow an appropriate reaction.

The instruction checks input variable INV for 4 assignable limits. If one of these limits is reached and exceeded, a limit signal is output. A hysteresis can be set for the off threshold.

Operating principle

The limits are set at inputs H_LM_ALM, H_LM_WRN, L_LM_WRN, and L_LM_ALM. If input variable INV exceeds the limits, the message bits QH_LMALM, QH_LMWRN, QL_LMWRN, and QL_LMALM are set. In order to avoid rapid activating and deactivating of the message bits, the input value must also violate a hysteresis HYS before the outputs are reset.

The figure below shows how the LIMALARM instruction functions:



QH_LMALM = TRUE	If	INV rises	and	INV >= H_LM_ALM
	or	INV falls	and	INV >= H_LM_ALM - HYS
QH_LMWRN = TRUE	If	INV rises	and	INV >= H_LM_WRN
	or	INV falls	and	INV >= H_LM_WRN - HYS
QL_LMWRN = TRUE	If	INV falls	and	INV <= L_LM_WRN
	or	INV rises	and	INV <= L_LM_WRN + HYS
QL_LMALM = TRUE	If	INV falls	and	INV <= L_LM_ALM
	or	INV rises	and	INV <= L_LM_ALM + HYS

In order for the instruction to function in a useful way, the following must apply:

L_LM_ALM < L_LM_WRN < H_LM_WRN < H_LM_ALM

Complete restart

During a complete restart, all the signal outputs are set to FALSE.

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

See also

Processing the process value (Page 9755)

Input parameters LIMALARM

Parameter	Offset	Data type	Preassigned value	Description
H_LM_ALM	0.0	REAL	100.0	High limit alarm Engineering value range > H_LM_WRN
H_LM_WRN	4.0	REAL	90.0	High limit warning Engineering value range > L_LM_WRN
L_LM_WRN	8.0	REAL	10.0	Low limit warning Engineering value range > L_LM_ALM
L_LM_ALM	12.0	REAL	0.0	Low limit alarm Engineering value range < L_LM_WRN
INV	16.0	REAL	0.0	Input variable Engineering value range
HYS	20.0	REAL	1.0	Hysteresis Engineering value range
COM_RST	24.0	BOOL	FALSE	Complete restart

Output parameters LIMALARM

Parameter	Offset	Data type	Preassigned value	Description
QH_LMALM	26.0	BOOL	FALSE	High limit alarm triggered
QH_LMWRN	26.1	BOOL	FALSE	High limit warning triggered
QL_LMWRN	26.2	BOOL	FALSE	Low limit warning triggered
QL_LMALM	26.3	BOOL	FALSE	Low limit alarm triggered

LIMITER

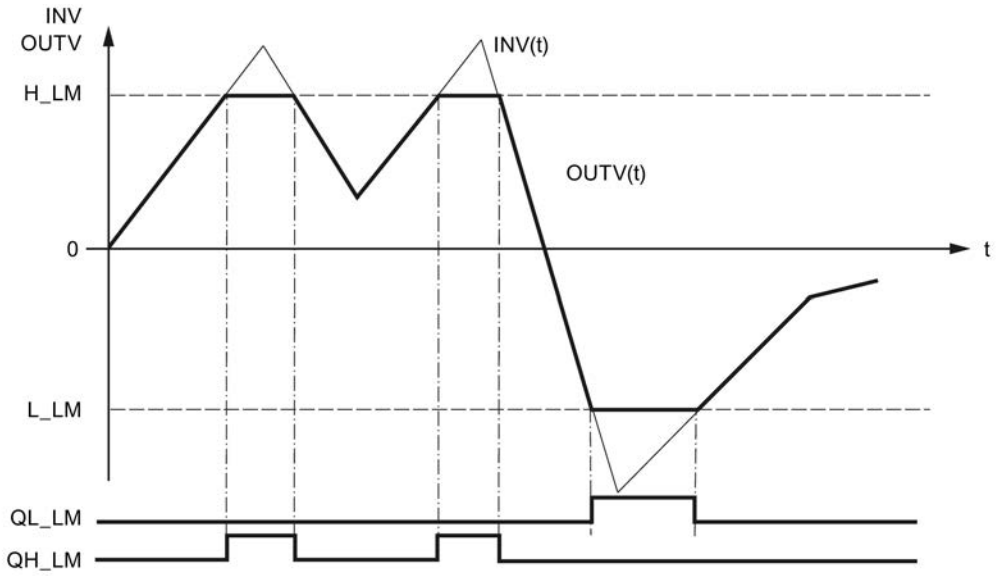
Description LIMITER

If parameters are defined dynamically (for example calculation of setpoints from process variables), these can assume values that are invalid for the process. The LIMITER instruction can be used to keep parameters within a valid range.

Operating principle

The instruction limits output variable OUTV to a definable high and low limit H_LM and L_LM when input variable INV lies outside these limits. Limiting of OUTV is signaled by outputs QH_LM and QL_LM.

The figure below shows how the LIMITER instruction functions:



INV	OUTV	QH_LM	QL_LM
≥ H_LM	H_LM,	TRUE	FALSE
≤ L_LM	L_LM,	FALSE	TRUE
≤ L_LM and ≤H_LM	INV	FALSE	FALSE

In order for the instruction to function in a useful way, the following must apply:

L_LM < H_LM

Complete restart

During a complete restart, all signal outputs are set to FALSE; 0.0 is output at output OUTV.

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

See also

Processing setpoints (Page 9753)
Processing the process value (Page 9755)

Input parameters LIMITER

Parameter	Offset	Data type	Preassigned value	Description
INV	0.0	REAL	0.0	Input variable Engineering value range
H_LM	4.0	REAL	100.0	High limit Engineering value range > L_LM
L_LM	8.0	REAL	0.0	Low limit Engineering value range < H_LM
COM_RST	12.0	BOOL	FALSE	Complete restart

Output parameters LIMITER

Parameter	Offset	Data type	Preassigned value	Description
OUTV	14.0	REAL	0.0	Output variable
QH_LM	18.0	BOOL	FALSE	High limit triggered
QL_LM	18.1	BOOL	FALSE	Low limit triggered

LMNGEN_C

Description LMNGEN_C

This instruction is used to set up a continuous PID controller and determines the manipulated variable or the controller. The manipulated variable can be determined in automatic mode or specified in manual mode. The manual value can be assigned as an absolute value or increased/decreased via a switch. The manipulated variable and the increase in the manipulated variable can be limited to definable values.

Restart

During a complete restart the default value DF_OUTV is switched through to the output LMN irrespective of the default bit DFOUT_ON. Limiting of the output and its display also remains

effective during a complete restart. On transition to normal operation, the instruction continues to operate from DF_OUTV.

The output parameters QLMN_HLM and QLMN_LLM are set to FALSE.

Call

While the PID instruction is called in a cyclic interrupt level whose cycle time is adapted to the dominating system time constant, the LMNGEN_C instruction for manual interventions can be called in a faster cyclic interrupt level. Interconnection is carried out with the structured input-output parameters PID_LMNG and LMNG_PID.

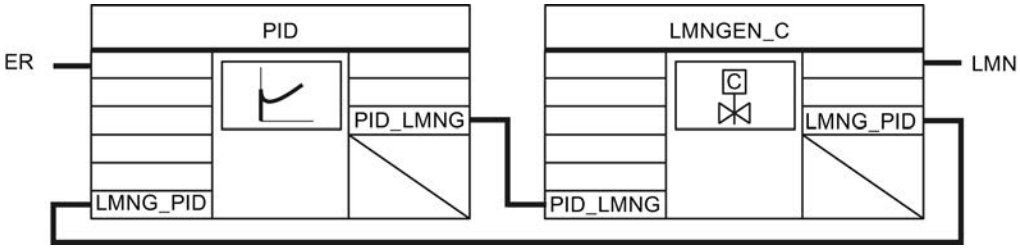
Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

Operating principle

The LMNGEN_C instruction is always used in conjunction with the PID instruction.

The following figure shows the interconnection of the continuous PID controller.



See also

Creating a continual controller (Page 9762)

Creating a pulse controller (Page 9763)

Operating principle LMNGEN_C

Limiting the manipulated variable

The manipulated variable is limited in all operating modes. You enter the high limit of the manipulated variable at input LMN_HLM and the low limit at input LMN_LLM. If the calculated manipulated variable exceeds or falls below these values, the high limit or low limit, respectively, is output at LMN. The limiting of the manipulated variable to the high limit is indicated at output QLMN_HLM, and the limiting to the limit limit at output QLMN_LLM.

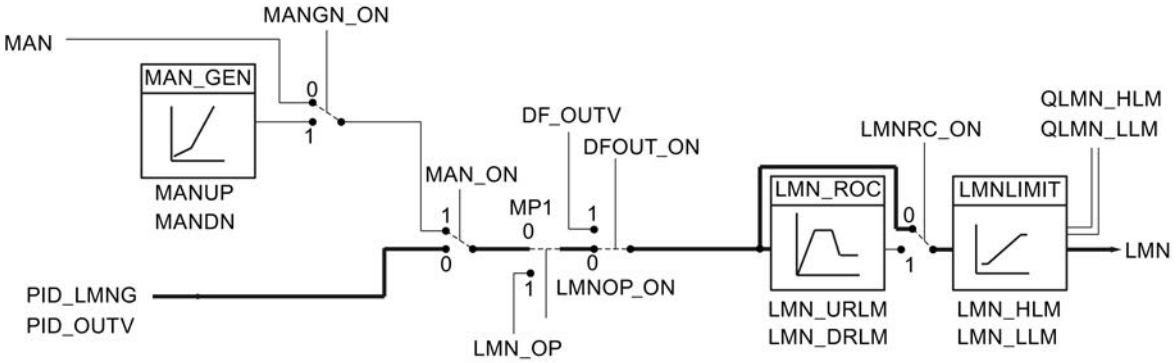
Manipulated variable ramp

You use the manipulated variable ramp to limit the rate of change of the manipulated variable. You activate the manipulated variable with LMNRC_ON=TRUE. At input parameter LMN_URLM, you specify the maximum rate of change for the rising slope of the manipulated variable, and at input parameter LMN_DRLM the maximum rate of change for the falling slope of the manipulated variable.

Operating modes

The operating modes are activated and deactivated using the DFOUT_ON, MAN_ON, and MANGN_ON parameters.

Operating modes	DFOUT_ON	MAN_ON	MANGN_ON
Automatic mode	FALSE	FALSE	FALSE
Manual mode	FALSE	TRUE	FALSE
Manual value generator	FALSE	TRUE	TRUE
Preassign manipulated variable	TRUE	Not relevant	Not relevant

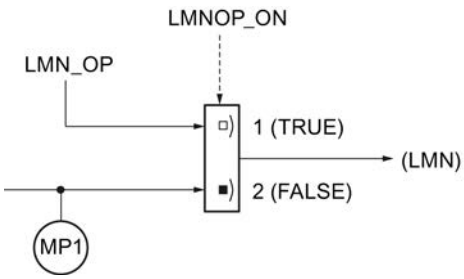


- **Automatic mode**
  The value determined by the PID algorithm is used to determine manipulated variable LMN. The changeover to automatic mode is bumpless if the manipulated variable ramp (LMNRC_ON = TRUE) is activated.
- **Manual mode**
  With MAN_ON = TRUE, you switch to manual mode and thus interrupt the control loop. Manual value MAN is output directly as manipulated variable LMN. MAN must lie within the limits LMN_HLM and LMN_LLM.

- **Manual value generator**
  If MANGN_ON = TRUE as well, the manipulated variable can be increased and decreased using the MANUP and MANDN switches, respectively. The rate of change of the manipulated variable depends on the limits as follows:
  - During the first 3 s after setting of MANUP or MANDN
    dLMN/dt = (LMN_HLM - LMN_LLM) / 100 s
  - Afterwards:
    dLMN/dt = (LMN_HLM - LMN_LLM) / 10 s
- **Preassign manipulated variable**
  When DFOUT_ON = TRUE, the preassigned value DF_OUTV is applied at output LMN. The changeover from or to "Preassign manipulated variable" is only bumpless if the manipulated variable ramp is activated. (LMNRC_ON= TRUE).

Operator control via HMI

You can use static variables to influence the manipulated variable via the HMI (see figure below). The static variables LMN_OP, LMNOP_ON, and MP1 may be operated and monitored via an HMI, for example.



If LMNOP_ON is set to TRUE, LMN_OP is switched through to manipulated variable LMN. MP1 contains the manipulated variable that is being processed by the LMNGEN_C instruction.

If the manipulated variable ramp is activated, bumpless changeover between the values MP1 and LMN_OP is possible. LMN is fed back to the value MP1 in conformance with the specified ramp.

When a static variable is changed, the change takes effect in the process only after the variable has been transferred to the CPU.

Input parameters LMNGEN_C

Parameter	Offset	Data type	Preassigned value	Description
LMN_HLM	0.0	REAL	100.0	High limit of manipulated variable Engineering value range > LMN_LLM
LMN_LLM	4.0	REAL	0.0	Low limit of manipulated variable Engineering value range < LMN_HLM
LMN_URLM	8.0	REAL	10.0	Manipulated variable up rate limit [1/s] > 0.0
LMN_DRLM	12.0	REAL	10.0	Manipulated variable down rate limit [1/s] > 0.0
DF_OUTV	16.0	REAL	0.0	Default setting of output variable Engineering value range
MAN_ON	20.0	BOOL	TRUE	Manual mode ON
MANGN_ON	20.1	BOOL	FALSE	Activate manual value operation
MANUP	20.2	BOOL	FALSE	Manual value up
MANDN	20.3	BOOL	FALSE	Manual value down
LMNRC_ON	20.4	BOOL	FALSE	Manipulated variable rate of change on
DFOUT_ON	20.5	BOOL	FALSE	Default setting of output variable On
COM_RST	20.6	BOOL	FALSE	Complete restart
CYCLE	22.0	TIME	T#1s	Sampling time ≥ 1ms
PID_LMNG	26.0	STRUC		PID-LMNGEN interface

Output parameters LMNGEN_C

Parameter	Offset	Data type	Preassigned value	Description
LMN	44.0	REAL	0.0	Manipulated variable
QLMN_HLM	48.0	BOOL	FALSE	High limit of manipulated variable triggered
QLMN_LLM	48.1	BOOL	FALSE	Low limit of manipulated variable triggered
LMNG_PID	50.0	STRUC		PID-LMNGEN interface

In/out parameters LMNGEN_C

Table 9-211

Parameter	Offset	Data type	Preassigned value	Description
MAN	68.0	REAL	0.0	Manual value Engineering value range

LMNGEN_C static variables

Parameter	Offset	Data type	Preassigned value	Description
MP1	72.0	REAL	0.0	Manipulated variable processed by LMNGEN_C.
LMM_OP	76.0	REAL	0.0	Operator control via HMI Manual value
LMNOP_ON	80.0	BOOL	FALSE	Operator control via HMI Manual mode ON

LMNGEN_S

Description LMNGEN_S

This instruction is used to set up a PID step controller for actuators with integral-action response (e.g., motor-driven valves) and determines the manipulated variable of the controller. The manipulated variable can be determined in automatic mode or specified in manual mode. The manual value can be assigned as an absolute value or increased/decreased via a switch. The manipulated variable can be limited to assignable values. The step controller can operate with or without position feedback.

Complete restart

During a complete restart, all the signal outputs are set to 0.

Call

While the PID instruction is called in a cyclic interrupt level whose cycle time is adapted to the dominating system time constant, the LMNGEN_S instruction for manual interventions can be called in a faster cyclic interrupt level. Interconnection is carried out with the structured input-output parameters PID_LMNG and LMNG_PID.

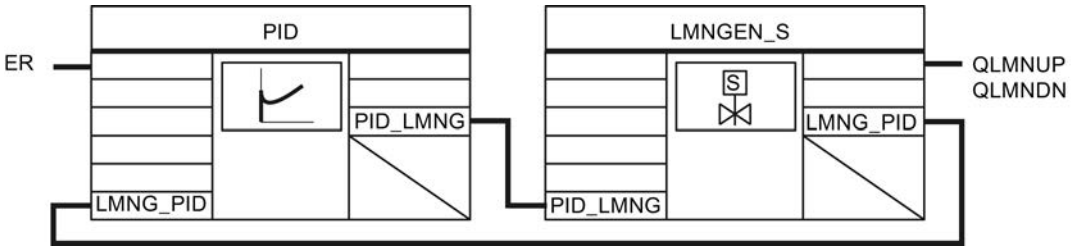
Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

Operating principle

The LMNGEN_S instruction is always used in conjunction with the PID instruction.

The following figure shows the interconnection of a step controller.



See also

Creating a step controller (Page 9767)

Operating principle LMNGEN_S

Limiting the manipulated variable

The manipulated variable is limited in all operating modes. You enter the high limit of the manipulated variable at input LMN_HLM and the low limit at input LMN_LLM. If the calculated manipulated variable exceeds or falls below these values, the high limit or low limit, respectively, is output at LMN. The limiting of the manipulated variable to the high limit is indicated at output QLMN_HLM, and the limiting to the limit limit at output QLMN_LLM.

Limit signals

You interconnect the signals for the upper and lower actuator stop with input parameters LMNR_HS and LMNR_LS. If LMNR_HS= TRUE, the actuator will not open further. If LMNR_LS= TRUE, the actuator will not close further. Output signals QLMNUP and QLMNDN are blocked. The end stops are taken into consideration in all operating modes.

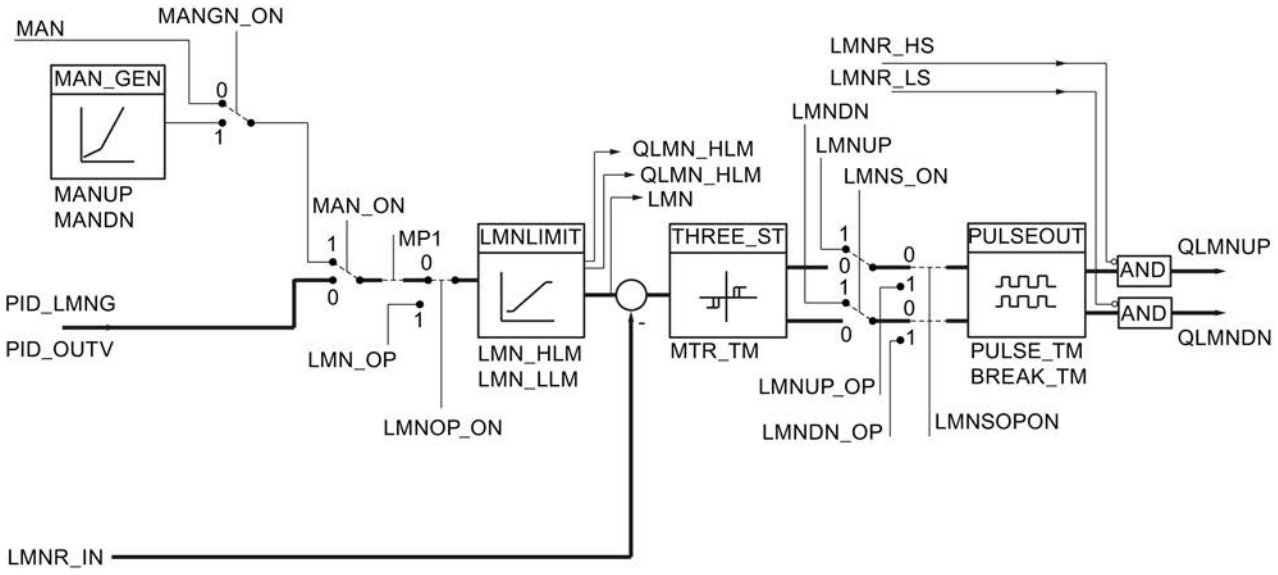
WARNING

If limit signals are not available, the controller cannot recognize a valve stop, and the possibility exists, for example, that the controller will output signals to open the valve even though the valve is at its upper stop. If limit signals are not available, inputs LMNR_HS and LMNR_LS must be assigned the value FALSE.

Step controller with position feedback

If position feedback is available, the instruction uses a three-step element and pulse generator to generate the pulses for controlling the control valve based on the difference between the manipulated variable and the position feedback. You can adapt the response threshold of the three-step element to reduce the switching frequency of the controller.

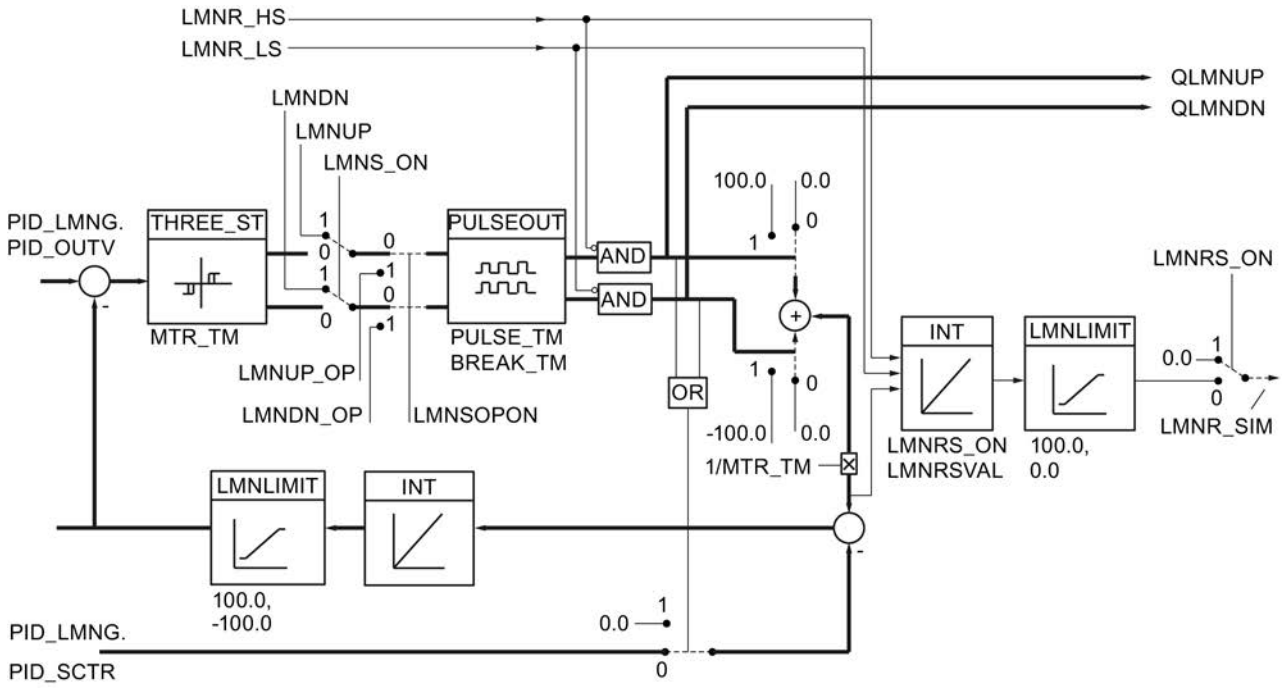
You activate the position feedback with LMNR_ON = TRUE.



Step controller without position feedback

If position feedback is not available, the instruction adds up the difference between the integral component of the PID algorithm and a simulated position feedback in an integrator and then compares this, as a feedback value, with the remaining proportional-derivative component. The difference is applied in turn to the three-step element and the pulse generator that generates the pulses for the control valve. You can adapt the response threshold of the three-step element to reduce the switching frequency of the controller.

You deactivate the position feedback with LMNR_ON = FALSE.



Operating modes

The operating modes are activated and deactivated using the LMN_ON, MAN_ON, and MANGN_ON parameters.

Operating modes	LMN_ON	MAN_ON	MANGN_ON
Automatic mode	FALSE	FALSE	FALSE
Specify manual value	FALSE	TRUE	FALSE
Manual value generator	FALSE	TRUE	TRUE
Manual mode of the manipulated variable output signals	TRUE	Not relevant	Not relevant

The Specify manual value and Manual value generator modes are only possible for step controllers with position feedback. Because information on the valve position is not available in a step controller without position feedback, the actuator cannot be moved to a fixed manual value.

- Specify manual value
  With MAN_ON = TRUE, you switch to manual mode and thus interrupt the conrol loop. Manual value MAN is switched through directly as the manipulated variable. MAN must lie within the limits LMN_HLM and LMN_LLM.
- Manual value generator
  If MANGN_ON = TRUE as well, the manipulated variable can be increased and decreased using the MANUP and MANDN switches, respectively. The rate of change of the manipulated variable depends on the limits as follows:
  - During the first 3 s after setting of MANUP or MANDN
    dLMN/dt = (LMN_HLM - LMN_LLM) / 100 s
  - Afterwards:
    dLMN/dt = (LMN_HLM - LMN_LLM) / 10 s
- Manual mode of the manipulated variable output signals
  If LMNS_ON = TRUE, the binary output signals can be influenced directly. The actuating signal outputs QLMNUP and QLMNDN are set by switches LMNUP and LMNDN. The minimum pulse time PULSE_TM and minimum break time BREAK_TM are also taken into consideration.
- Automatic operation with position feedback
  The value PID_LMNG.PID_OUTV sent by the PID algorithm is limited to assignable values LMN_HLM and LMN_LLM. The difference between the manipulated variable and the position feedback is switched to a three-step element with hysteresis. The trip threshold is calculated from the motor actuating time. In order to reduce the switching frequency, the response threshold is adapted. The pulse generator PULSEOUT ensures that the minimum pulse time and minimum break time are observed.
- Automatic mode without position feedback
  The difference between the integral component of the PID algorithm and a simulated position feedback is summed in integrator INT. The input of the integrator is the difference of 100.0/0.0/-100.0 (depending on the state of output signals QLMNUP and QLMNDN), referenced to motor actuating time MTR_TM, and the control deviation multiplied times the controller gain, referenced to integral-action time TI of the PID algorithm (PID_LMNG.PID_SCTR). The output of the integrator forms the feedback value that is compared with the remaining proportional-derivative component of the PID algorithm (PID_LMNG.PID_OUTV). The difference is switched to a three-step element with hysteresis. The trip threshold is calculated from the motor actuating time.
  In order to reduce the switching frequency, the response threshold is adapted. The pulse generator PULSEOUT ensures that the minimum pulse time and minimum break time are observed.

Simulation of the position feedback

WARNING

The position feedback is only simulated. It does not have to agree with the real position feedback of the control valve.

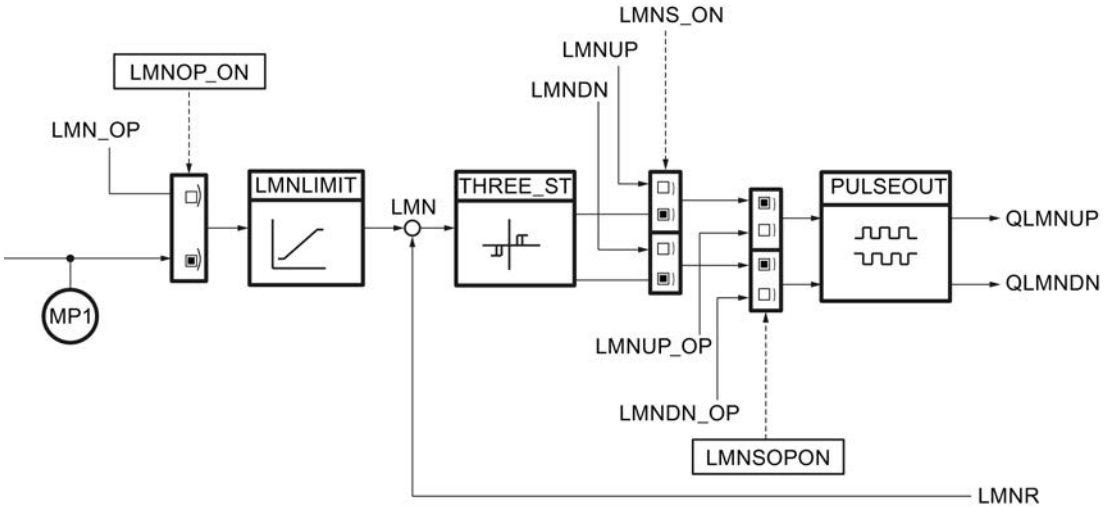
If a real position feedback exists, it should also be used.

The simulation of the position feedback is carried out by an integrator with motor actuating time MTR_TM as the integral-action time constant. A rising edge at LMNRS_ON starts the simulation with start value LMNRSVAL. If the value of LMNRS_ON is FALSE, the start value LMNRSVAL is output in parameter LMNR_SIM.

The limit signals LMNR_HS = TRUE and LMNR_LS = TRUE block the integration at high and low limits.

Operator input via HMI

You can use an HMI and static variables to interrupt the manipulated variable branch and specify your own manipulated variables (see figure below).



If LMNOP_ON is set to TRUE, LMN_OP is switched through to manipulated variable LMN. MP1 contains the manipulated variable that is being processed by the LMNGEN_S instruction.

If the manipulated variable ramp is activated, bumpless changeover between the values MP1 and LMN_OP is possible. LMN is fed back to the value MP1 in conformance with the specified ramp.

If LMNSOPON is set to TRUE, you can increase the manipulated variable with LMNUP_OP or decrease it with LMNDN_OP. This applies to step controllers with and without position feedback.

The static variables LMNOP_ON, LMN_OP, MP1, LMNSOPON, LMNUP_OP, and LMNDN_OP must not be interconnected and are only manipulated and monitored via an HMI.

When a static variable is changed, the change takes effect in the process only after the variable has been transferred to the CPU.

Input parameters LMNGEN_S

Parameter	Offset	Data type	Preassigned value	Description
LMN_HLM	0.0	REAL	100.0	High limit of manipulated variable Engineering value range > LMN_LLM
LMN_LLM	4.0	REAL	0.0	Low limit of manipulated variable Engineering value range < LMN_HLM
LMNR_IN	8.0	REAL	0.0	Position feedback Engineering value range
MTR_TM	12.0	TIME	T#30s	Motor actuating time ≥ CYCLE
PULSE_TM	16.0	TIME	T#3s	Minimum pulse time ≥ CYCLE
BREAK_TM	20.0	TIME	T#3s	Minimum break time ≥ CYCLE
MAN_ON	24.0	BOOL	TRUE	Manual mode ON
MANGN_ON	24.1	BOOL	FALSE	Activate manual value operation
MANUP	24.2	BOOL	FALSE	Manual value up
MANDN	24.3	BOOL	FALSE	Manual value down
LMNR_ON	24.4	BOOL	FALSE	Position feedback on
LMNR_HS	24.5	BOOL	FALSE	High limit signal of position feedback
LMNR_LS	24.6	BOOL	FALSE	Low limit signal of position feedback
LMNS_ON	24.7	BOOL	FALSE	Activate manual mode for manipulated variable signals
LMNUP	25.0	BOOL	FALSE	Manipulated variable signal UP
LMNDN	25.1	BOOL	FALSE	Manipulated variable signal DOWN
COM_RST	25.2	BOOL	FALSE	Complete restart
CYCLE	26.0	TIME	T#1s	Sampling time ≥ 1ms
PID_LMNG	30.0	STRUC		PID-LMNGEN interface

Output parameters LMNGEN_S

Parameter	Offset	Data type	Preassigned value	Description
LMN	48.0	REAL	0.0	Manipulated variable
QLMNUP	52.0	BOOL	FALSE	Manipulated value signal up
QLMNDN	52.1	BOOL	FALSE	Manipulated variable signal DOWN

Parameter	Offset	Data type	Preassigned value	Description
QLMN_HLM	52.2	BOOL	FALSE	High limit of manipulated variable triggered
QLMN_LLM	52.3	BOOL	FALSE	Low limit of manipulated variable triggered
LMNG_PID	54.0	STRUC		PID-LMNGEN interface

In/out parameters LMNGEN_S

Table 9-212

Parameter	Offset	Data type	Preassigned value	Description
MAN	72.0	REAL	0.0	Manual value Engineering value range

LMNGEN_S static variables

You must not modify variables that are not listed. These are used for internal purposes only.

Parameter	Offset	Data type	Preassigned value	Description
LMNSOPON	76.0	BOOL	FALSE	Operator control via HMI Manual mode of the manipulated variable output signals ON
LMNUP_OP	76.1	BOOL	FALSE	Operator control via HMI Manipulated variable UP
LMNDN_OP	76.2	BOOL	FALSE	Operator control via HMI Manipulated variable DOWN
LMNOP_ON	76.3	BOOL	FALSE	Operator control via HMI Manual mode ON
LMNRS_ON	76.4	BOOL	FALSE	Simulated position feedback ON
MP1	78.0	REAL	0.0	Operator control via HMI Manipulated variable processed by LMNGEN_C.
LMM_OP	82.0	REAL	0.0	Operator control via HMI Manual value
LMNRSVAL	86.0	REAL	0.0	Initial value for simulate position feedback
LMNR_SIM	9.0	REAL	0.0	Simulated position feedback

NONLIN

Description NONLIN

The NONLIN instruction can be used to adapt an input value, such as a measured value of a thermocouple, as a function of a definable characteristic (polyline).

Operating principle

NONLIN interpolates the corresponding output variable OUTV for the current input variable INV using time slices on a polyline. The time slices of the polyline are stored in a shared data block.

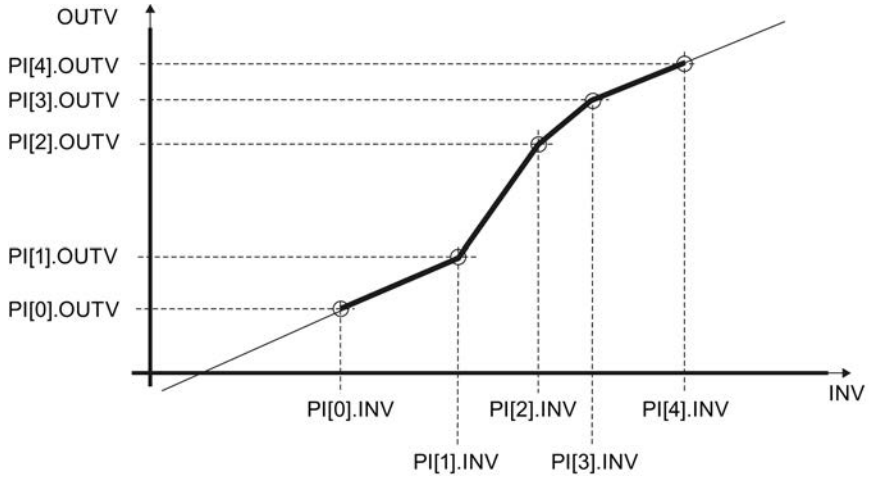
Note

The instruction does not check whether a shared DB with the number DB_NBR actually exists or whether the DB_NBR. NBR_PTS parameter (number of time slices) is suited to the length of the data block. If the parameter assignment is incorrect, the CPU changes to the STOP operating mode with the message "Internal system error".

NONLIN has the following functions:

Functions	DFOUT_ON	TRACK	OUTV
Interpolation	FALSE	FALSE	OUTV = f(INV)
Preassign output	TRUE	Any	DF_OUTV
Tracking	FALSE	TRUE	OUTV = INV

- **Interpolation**
  The following figure shows how output variable OUTV is determined from input variable INV.
  OUTV = f(INV)



  If the input value lies below time slice 0, extrapolation is carried out with the rise of the time slice pair 0/1. It it lies above the last time slice, extrapolation is carried out with the rise of the last time slice pair. In order for the instruction to produce a useful result, the values of the time slices must exhibit a monotonic rise in ascending order.

- **Preassign output**
  If DFOUT_ON = TRUE is set, DF_OUTV is output at the output; the change of OUTV is a step change. On changeover to DFOUT_ON = FALSE the change of OUTV is also a step change.

- **Tracking**
  When TRACK=TRUE, the input value is output directly (OUTV=INV). The change is a step change, as in the case of DFOUT_ON. TRACK has lower priority than DFOUT_ON.

Complete restart

On a complete restart, OUTV = 0.0 is output. If DFOUT_ON=TRUE, DF_OUTV is output.

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

See also

Processing setpoints (Page 9753)

Processing the process value (Page 9755)

Applying the disturbance variable (Page 9759)

Processing position feedback (Page 9768)

Input parameters NONLIN

Parameter	Offset	Data type	Preassigned value	Description
INV	0.0	REAL	0.0	Input variable Engineering value range
DF_OUTV	4.0	REAL	0.0	Default setting of output variable Engineering value range
DB_NBR	8.0	BLOCK_DB	DB 1	Data block number
DFOUT_ON	10.0	BOOL	FALSE	Default setting of output variable On
TRACK	10.1	BOOL	FALSE	Tracking OUTV=INV
COM_RST	10.2	BOOL	FALSE	Complete restart

Output parameters NONLIN

Parameter	Offset	Data type	Preassigned value	Description
OUTV	12.0	REAL	0.0	Output variable

Shared data block DB_NBR

Parameter	Data type	Preassigned value	Description
DB_NBR.NBR_PTS	INT	4	Highest time slice number Range of values 1 - 255
DB_NBR.PI[0].OUTV	REAL	0.0	Output variable [0] Starting point 0 Engineering value range
DB_NBR.PI[0].INV	REAL	0.0	Input variable [0] Starting point 0 Engineering value range
DB_NBR.PI[1].OUTV	REAL	0.0	Output variable [1] Time slice 1 Engineering value range
DB_NBR.PI[1].INV	REAL	0.0	Input variable [1] Time slice 1 Engineering value range
DB_NBR.PI[2].OUTV	REAL	0.0	Output variable [2] Time slice 2 Engineering value range
DB_NBR.PI[2].INV	REAL	0.0	Input variable [2] Time slice 2 Engineering value range
DB_NBR.PI[3].OUTV	REAL	0.0	Output variable [3] Time slice 3 Engineering value range
DB_NBR.PI[3].INV	REAL	0.0	Input variable [3] Time slice 3 Engineering value range

Parameter	Data type	Preassigned value	Description
DB_NBR.PI[4].OUTV	REAL	0.0	Output variable [4] Time slice 4 Engineering value range
DB_NBR.PI[4].INV	REAL	0.0	Input variable [4] Time slice 4 Engineering value range

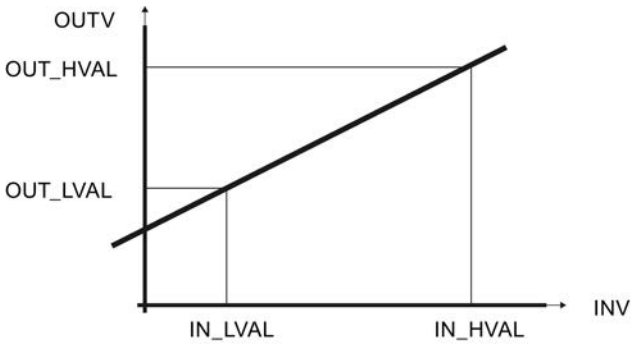
NORM

Description NORM

In the case of process values the value supplied by the encoder often lies in a range that is unfavorable for the user (for example 0 to 10 V correspond to 0 to 1200 ºC or 0 to 10 V correspond to 0 to 3000 rpm). By scaling the setpoint or the process value, both process variables lie in the same range of values.

Operating principle

The instruction scales the input variable into an output variable with a different range of values. Input variable INV is transformed to output variable OUTV based on the scaling line. The scaling line is defined by the two value pairs IN_LVAL, OUT_LVAL and IN_HVAL, OUT_HVAL.



Complete restart

The block does not have a complete restart routine.

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

See also

Processing setpoints (Page 9753)
Processing the process value (Page 9755)
Applying the disturbance variable (Page 9759)

Input parameters NORM

Parameter	Offset	Data type	Preassigned value	Description
INV	0.0	REAL	0.0	Input variable Engineering value range
IN_HVAL	4.0	REAL	100.0	Physical input value below Engineering value range > IN_LVAL
OUT_HVAL	8.0	REAL	100.0	Physical output value below Engineering value range > OUT_LVAL
IN_LVAL	12.0	REAL	0.0	Physical input value above Engineering value range < IN_HVAL
OUT_LVAL	16.0	REAL	0.0	Physical output value above Engineering value range < OUT_HVAL

Output parameters NORM

Parameter	Offset	Data type	Description
OUTV	20.0	REAL	Output variable

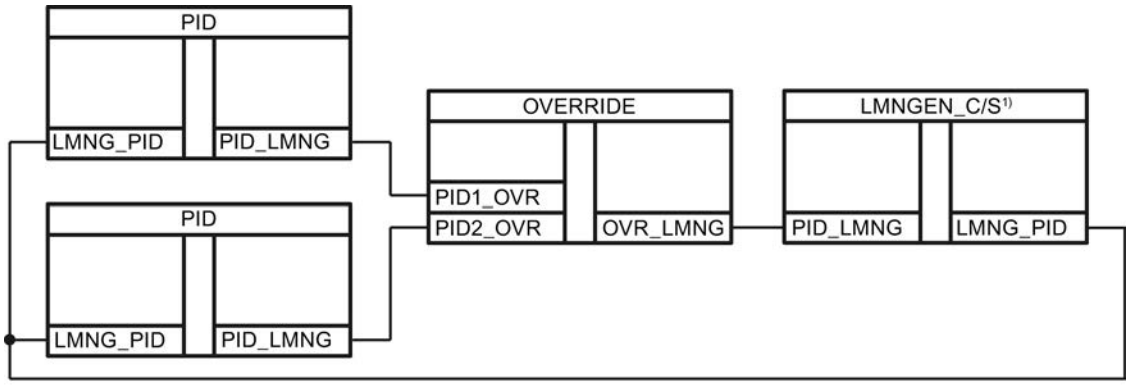
OVERRIDE

Description OVERRIDE

This instruction is required to implement an override control.

Operating principle

Two PID controllers are connected to one actuator. For this purpose, two PID instructions are interconnected by means of instruction OVERRIDE with instruction LMNGEN_C or LMNGEN_S. If you use LMNGEN_S, position feedback must be activated (LMNR_ON=TRUE).



The instruction compares the values PID1_OVR and PID2_OVR and connects either the larger or the smaller value.

The instruction transfers the value in PID1_OVR or in PID2_OVR to LMN_GEN_C or LMN_GEN_S.

If the PID1_ON and PID2_ON switches are unequal, the one value is always transferred. If the PID1_ON and PID2_ON switches are equal, the larger (OVR_MODE=FALSE) or smaller (OVR_MODE=TRUE) of the two values is transferred.

PID1_ON	PID2_ON	OVR_MOD	Function
TRUE	FALSE	Not relevant	PID1_OVR is transferred.
FALSE	TRUE		PID2_OVR is transferred.
FALSE	FALSE	FALSE	The larger value of PID1_OVR and PID2_OVR is transferred.
TRUE	TRUE	FALSE	
FALSE	FALSE	TRUE	The smaller value of PID1_OVR and PID2_OVR is transferred.
TRUE	TRUE	TRUE	

If PID1_OVR is transferred, QPID1=TRUE; if PID2_OVR is transferred, QPID2=TRUE.

Complete restart

The block does not have a complete restart routine.

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

Input parameters OVERRIDE

Parameter	Offset	Data type	Preassigned value	Description
PID1_ON	0.0	BOOL	FALSE	PID controller 1 on
PID2_ON	0.1	BOOL	FALSE	PID controller 2 on
OVR_MODE	0.2	BOOL	FALSE	Override mode FALSE=maximum, TRUE=minimum
PID1_OVR	2.0	STRUC		PID-LMNGEN interface
PID2_OVR	20.0	STRUC		PID-LMNGEN interface

Output parameters OVERRIDE

Parameter	Offset	Data type	Description
QPID1	38.0	BOOL	PID controller 1 active
QPID2	38.1	BOOL	PID controller 2 active
OVR_LMNG	40.0	STRUC	PID-LMNGEN interface

PARA_CTL

Description PARA_CTL

The instruction is used in controller structures with parameter changeovers if optimum controller parameters are required for different operating ranges.

Operating principle

Several controller parameter sets (GAIN, TI, TD, and TM_LAG) can be transferred to a PID controller.

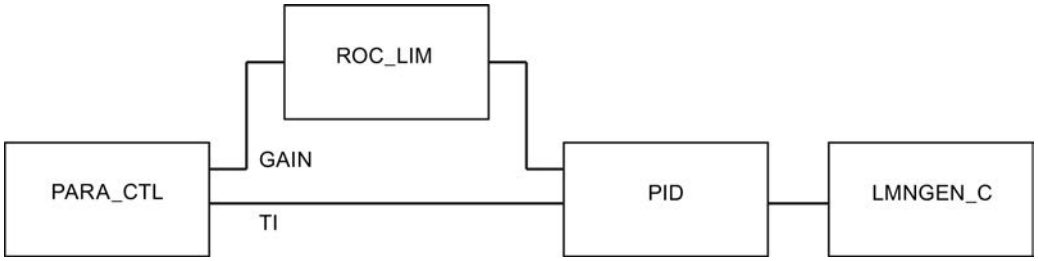
The PSET1_ON...PSET4_ON switches can be used to switch through one of the 4 parameter sets to outputs GAIN, TI, TD, and TM_LAG.

If...	Then...
PSET1_ON = TRUE	GAIN = PARASET1.GAIN
	TI = PARASET1.TR
	TD = PARASET1.TD
	TM_LAG = PARASET1.TM_LAG
PSET2_ON = TRUE	GAIN = PARASET2.GAIN
	TI = PARASET2.TR
	TD = PARASET2.TD
	TM_LAG = PARASET2.TM_LAG

PSET3_ON = TRUE	GAIN = PARASET3.GAIN
	TI = PARASET3.TR
	TD = PARASET3.TD
	TM_LAG = PARASET3.TM_LAG
PSET4_ON = TRUE	GAIN = PARASET4.GAIN
	TI = PARASET4.TR
	TD = PARASET4.TD
	TM_LAG = PARASET4.TM_LAG

If 2 or more switches are set to TRUE, the switch with the lowest number always has the highest priority. If no switch is set to TRUE, the first parameter set is transferred.

In order to achieve a bumpless changeover between the parameter sets, a rate of change limiter (ROC_LIM) must be connected between the PID and PARA_CTL instructions for the GAIN parameter.



Complete restart

The block does not have a complete restart routine.

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

Input parameters PARA_CTL

Parameter	Offset	Data type	Preassigned value	Description
PSET1_ON	0.0	BOOL	FALSE	Transfer Parameter set number 1
PSET2_ON	0.1	BOOL	FALSE	Transfer Parameter set number 2
PSET3_ON	0.2	BOOL	FALSE	Transfer Parameter set number 3
PSET4_ON	0.3	BOOL	FALSE	Transfer Parameter set number 4
PARASET1.GAIN	2.0	REAL		Proportional gain 1
PARASET1.TI	6.0	TIME	5 s	Integral-action time 1
PARASET1.TD	10.0	TIME	10 s	Derivative-action time 1
PARASET1.TM_LAG	14.0	TIME	2 s	Time delay 1

Parameter	Offset	Data type	Preassigned value	Description
PARASET2.GAIN	18.0	REAL		Proportional gain 2
PARASET2.TI	22.0	TIME	5 s	Integral-action time 2
PARASET2.TD	26.0	TIME	10 s	Derivative-action time 2
PARASET2.TM_LAG	30.0	TIME	2 s	Time delay 2
PARASET3.GAIN	34.0	REAL		Proportional gain 3
PARASET3.TI	38.0	TIME	5 s	Integral-action time 3
PARASET3.TD	42.0	TIME	10 s	Derivative-action time 3
PARASET3.TM_LAG	46.0	TIME	2 s	Time delay 3
PARASET4.GAIN	500	REAL		Proportional gain 4
PARASET4.TI	54.0	TIME	5 s	Integral-action time 4
PARASET4.TD	58.0	TIME	10 s	Derivative-action time 4
PARASET4.TM_LAG	62.0	TIME	2 s	Time delay 4

Output parameters PARA_CTL

Parameter	Offset	Data type	Preassigned value	Description
GAIN	66.0	REAL	1.0	Proportional gain
TI	70.0	TIME	10s	Integral action time
TD	74.0	TIME	5s	Derivative-action time
TM_LAG	78.0	TIME	2s	Time delay

PID

Description PID

This instruction contains the PID algorithm for setting up the following controller types:

Continuous PID controller:	PID + LMNGEN_C
PID step controller for proportional actuators:	PID + LMNGEN_C + PULSEGEN
PID step controller for integrating actuators:	PID + LMNGEN_S

The instruction implements the PID algorithm. It is realized as a pure parallel structure and acts solely as a position algorithm. You can activate and deactivate the proportional, integral and derivative components individually. This allows P, PI, PD, and PID controllers to be configured.

The calculation of the proportional and derivative components can be laid into the feedback. Laying the proportional and derivative components into the feedback loop makes the response to setpoint changes "smooth" while retaining the same speed in settling disturbance variables.

It is usually possible to dispense with the common application of a setpoint integrator in order to avoid setpoint steps.

While the PID instruction is placed in a cyclic interrupt level whose cycle time is adapted to the dominating system time constant, the instructions that run the actuators (LMNGEN_C or LMNGEN_S) can be placed in a faster cyclic interrupt level.

Complete restart

During a complete restart zero is pre-assigned to all the output and in/out parameters.

Block-internal limits

The integral-action time is subject to a limit equal to half the sampling time.

The derivative-action time is subject to a low limit equal to the sampling time.

The time lag is subject to a low limit equal to half the sampling time.

TI_{intern} = CYCLE/2	for	TI < CYCLE/2
TD_{intern} = CYCLE	for	TD < CYCLE
TM_LAG_{intern} = CYCLE/2	for	TM_LAG < CYCLE/2

The other values of the input parameters are not limited in the instruction. There is no parameter check.

See also

Creating a continual controller (Page 9762)

Creating a pulse controller (Page 9763)

Creating a step controller (Page 9767)

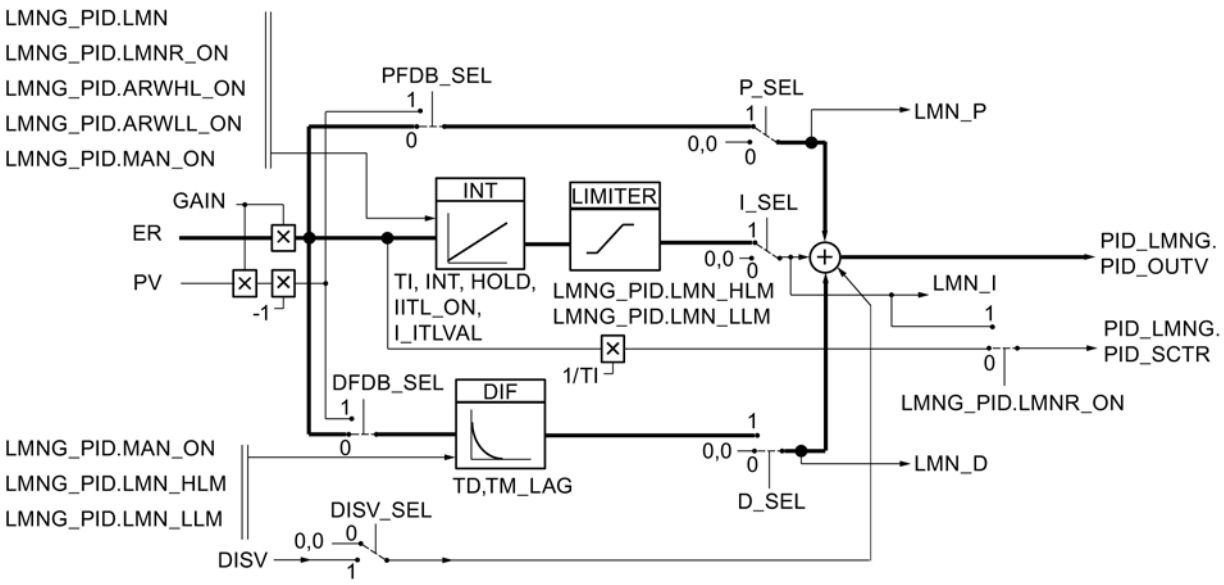
Operating principle PID

Controller structure

The following controller structures are possible with switches P_SEL, I_SEL and D_SEL:

Controller structure	P_SEL	I_SEL	D_SEL
P control	TRUE	TRUE bzw. FALSE	FALSE
PI control	TRUE	TRUE	FALSE
PD control	TRUE	FALSE	TRUE
PID control	TRUE	TRUE	TRUE

Block diagram of the PID algorithm



Proportional component

The proportional action can optionally be activated or deactivated using the P_SEL switch. The proportional action is the product of the control deviation ER and the proportional gain GAIN. It can be laid into the feedback path using the PFDB_SEL switch. If the proportional action has been laid into the feedback path, the proportional action is the product of the the process value PV and the proportional gain.

Integral action

The integral action can optionally be activated or deactivated with the I_SEL switch. The time response is determined by the integral action time TI. When deactivated, the integral action and the internal memory of the integral action are set to zero. The integral action can be held using INT_HOLD. You can assign the integral action the value at input I_ITLVAL with the I_ITL_ON switch. In the case manipulated variable limiting the Integral action time remains at the old value (anti reset wind-up).

The following applies to a step controller without position feedback: LMN_I = 0.0

Derivative component

The derivative action can be activated or deactivated with the D_SEL switch. It can be laid into the feedback path by using the DFDB_SEL switch. The input variable of the derivative action is now the controlled variable PV. The time response is determined by the derivative action time TD.

Oscillation in manipulated variable signals may occur, particularly with rapid processes and an activated derivative action. In this case, the control response can be improved by using the first-order delay integrated in the derivative action. Usually a small TM_LAG time delay is sufficient in order to achieve the desired success.

Feedforward control

A disturbance variable DISV can be fed forward additively to the manipulated variable. It can optionally be activated or deactivated by using the DISV_SEL switch.

Input parameters PID

Parameter	Offset	Data type	Preassigned value	Description
ER	0.0	REAL	0.0	Control deviation Engineering value range
PV	4.0	REAL	0.0	Controlled variable (proportional or derivative component in the feedback) Engineering value range
GAIN	8.0	REAL	1.0	Proportional gain
TI	12.0	TIME	T#20s	Integral action time
I_ITLVAL	16.0	REAL	0.0	Initialization value for the integral component Engineering value range
TD	20.0	TIME	T#10s	Derivative-action time
TM_LAG	24.0	TIME	T#2s	Time delay of the derivative component
DISV	28.0	REAL	0.0	Disturbance variable Engineering value range
P_SEL	32.0	BOOL	TRUE	Activate proportional component
PFDB_SEL	32.1	BOOL	FALSE	Switch proportional component into feedback
DFDB_SEL	32.2	BOOL	FALSE	Switch derivative component into feedback
I_SEL	32.3	BOOL	TRUE	Activate integral component
INT_HPOS	32.4	BOOL	FALSE	Hold integral component in positive direction
INT_HNEG	32.5	BOOL	FALSE	Hold integral component in negative direction
I_ITL_ON	32.6	BOOL	FALSE	Set integral component
D_SEL	32.7	BOOL	FALSE	Activate derivative component
DISV_SEL	33.0	BOOL	TRUE	Apply disturbance variable
SMOO_CHG	33.1	BOOL	TRUE	Smooth changeover from manual mode to automatic mode
COM_RST	33.2	BOOL	FALSE	Complete restart
CYCLE	34.0	TIME	T#1s	Sampling time ≥ 1ms
LMNG_PID	38.0	STRUC		PID-LMNGEN interface

Output parameters PID

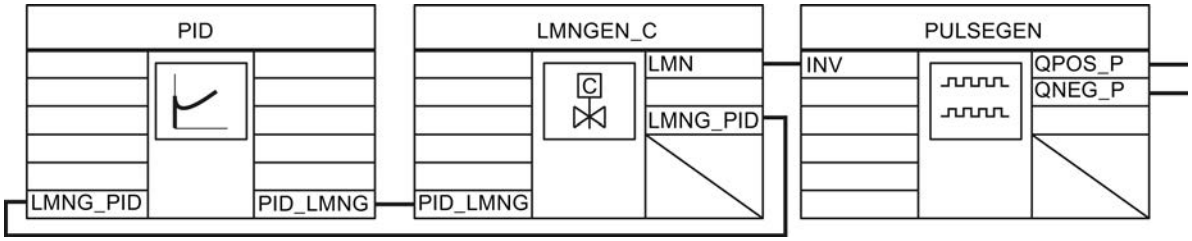
Parameter	Offset	Data type	Preassigned value	Description
LMN_P	56.0	REAL	0.0	Proportional component
LMN_I	60.0	REAL	0.0	Integral component
LMN_D	64.0	REAL	0.0	Derivative component
PID_LMNG	68.0	STRUC		PID-LMNGEN interface

PULSEGEN_M

Description PULSEGEN_M

The instruction PULSEGEN_M is used to set up a PID controller with pulse output for proportional actuators. You can use the PULSEGEN_M instruction to configure two- or three-step PID controllers with pulse width modulation. The instruction is usually used in conjunction with the continuous controller PID + LMNGEN_C.

The diagram below shows how the instructions are connected.



Restart

During a restart, all the signal outputs are set to 0.

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

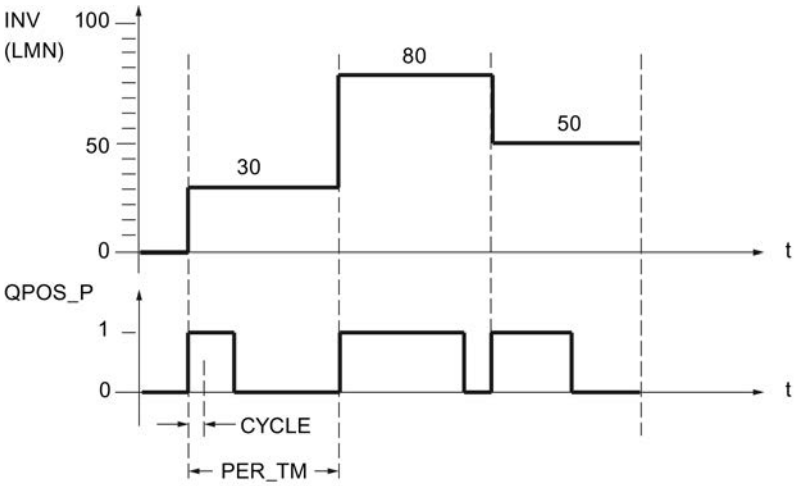
See also

Creating a pulse controller (Page 9763)

Operating principle PULSEGEN_M

Pulse width modulation

PULSEGEN_M transforms the input variable INV (= LMN of the PID controller) by modulating the pulse width into a pulse train with a constant period PER_TM. The period corresponds to the cycle time with which the input variable is updated. The duration of a pulse per period is proportional to the input variable. The cycle assigned via PER_TM is not identical to the processing cycle of the PULSEGEN_M instruction. Rather, a PER_TM cycle is made up of several processing cycles of the PULSEGEN_M instruction, whereby the number of PULSEGEN_M calls per PER_TM cycle measures the accuracy of the pulse width.



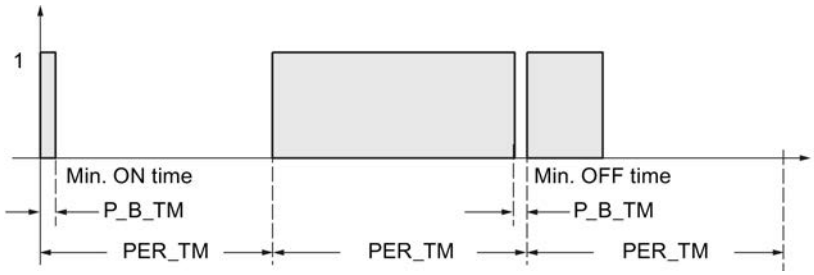
An input variable of 30% and 10 PULSEGEN calls per PER_TM mean the following:

- "One" at the QPOS_P output for the first three calls of PULSEGEN_M (30% of 10 calls)
- "Zero" at the QPOS_P output for seven further calls of PULSEGEN_M (70% of 10 calls)

Minimum pulse time or minimum break time

A correctly configured minimum pulse time or minimum break time P_B_TM can prevent short on/off times, which reduce the working life of actuators. Small absolute values at the input variable LMN that would otherwise generate a pulse duration shorter than P_B_TM are suppressed. Large input values that would generate a pulse duration longer than PER_TM - P_B_TM are set to 100 % or -100 %.

The following formula applies: P_B_TM ≤ 0.1 × PER_TM



Duration of positive or negative pulses is calculated from the input variable (in %) multiplied by the period duration:

Pulse duration = INV / 100 * PER_TM

Accuracy of pulse width modulation

While the continual controller PID+LMGEN_C is placed in a slow cyclic interrupt level whose cycle time is adapted to the dominating system time constant, the PULSEGEN_M block must be placed in a faster cyclic interrupt level.

With a "Sampling ratio" of 1:10 (controller calls to PULSEGEN_M calls), the accuracy of the manipulated variable is restricted to 10%, in other words, set input values INV can only be simulated by a pulse duration at the QPOS_P output with an accuracy of 10%.

The accuracy is increased as the number of PULSEGEN_M calls per controller call is increased.

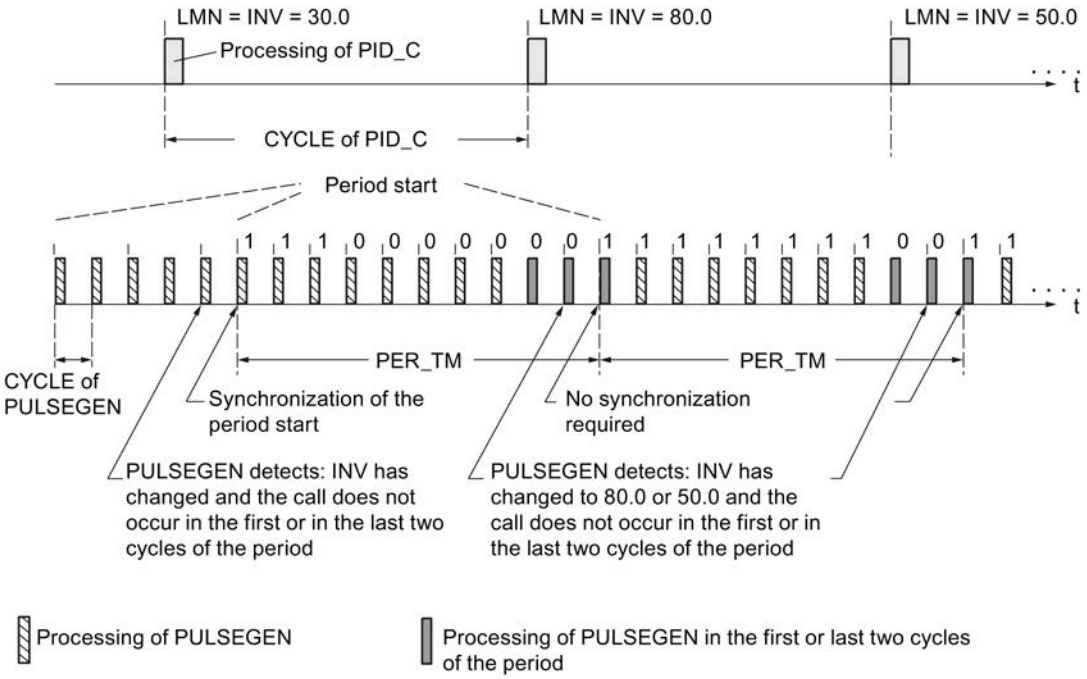
The faster the cyclic interrupt level is, the greater the precision with which the manipulated variable can be output. At a cyclic interrupt level that is 100 times faster, a resolution of 1% of the manipulated variable range is achieved.

Automatic synchronization

It is possible to synchronize the pulse output with the controller instruction automatically. This ensures that a modified value of the manipulated variable LMN(t) is output as fast as possible as a proportionally modified pulse duration of the binary signal.

The pulse generator evaluates the input variable INV within the interval of the period PER_TM in each case. Since, however, INV is calculated in a slower cyclic interrupt class, the pulse generator should start the conversion of the discrete value into a pulse signal as soon as possible after the updating of INV. To allow this, the instruction can synchronize the start of the period using the following procedure. If INV changes and if the block call is not in the first or last two call cycles of a period, synchronization is performed. The pulse duration is recalculated and, in the next cycle, is output with a new period.

The period PER_TM must correspond to the sampling time CYCLE of the controller.



The automatic synchronization is switched off if SYN_ON = FALSE.

Note

The start of a new period and subsequent synchronization usually leads to a certain imprecision when the old value of INV (i.e. of LMN) is mapped to the pulse signal.

PULSEGEN_M modes

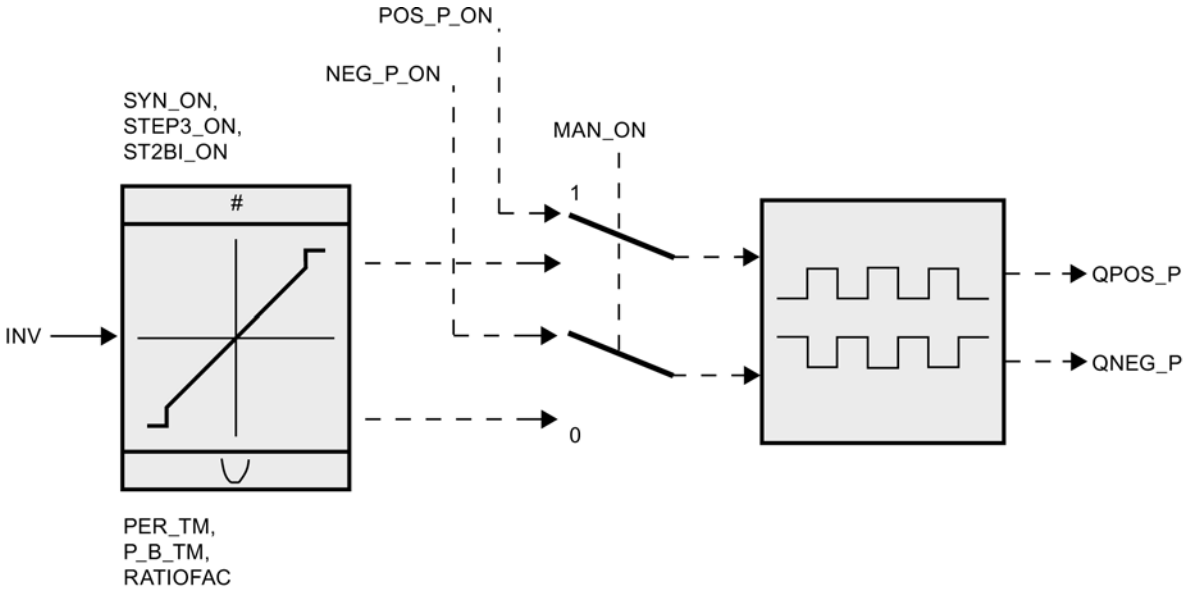
Operating modes

Depending on the parameters assigned to the pulse shaper, PID controllers with a three-step output or with a bipolar or unipolar two-step output can be configured. The following table illustrates the setting of the switch combinations for the possible modes.

Mode	MAN_ON	STEP3_ON	ST2BI_ON
Three-step control	FALSE	TRUE	Any
Two-step control with bi-polar Manipulating range (-100 % to 100 %)	FALSE	FALSE	TRUE
Two-step control with unipolar Manipulating range (0 % to 100 %)	FALSE	FALSE	FALSE
Manual mode	TRUE	Any	Any

Manual mode in two/three-step control

In the manual mode (MAN_ON = TRUE), the binary outputs of the three-step or two-step controller can be set using the signals POS_P_ON and NEG_P_ON regardless of INV.



Control	POS_P_ON	NEG_P_ON	QPOS_P	QNEG_P
Three-step control	FALSE	FALSE	FALSE	FALSE
	TRUE	FALSE	TRUE	FALSE
	FALSE	TRUE	FALSE	TRUE
	TRUE	TRUE	FALSE	FALSE
Two-step control	FALSE	Any	FALSE	TRUE
	TRUE	Any	TRUE	FALSE

PULSEGEN_M three-step control

Three-step control

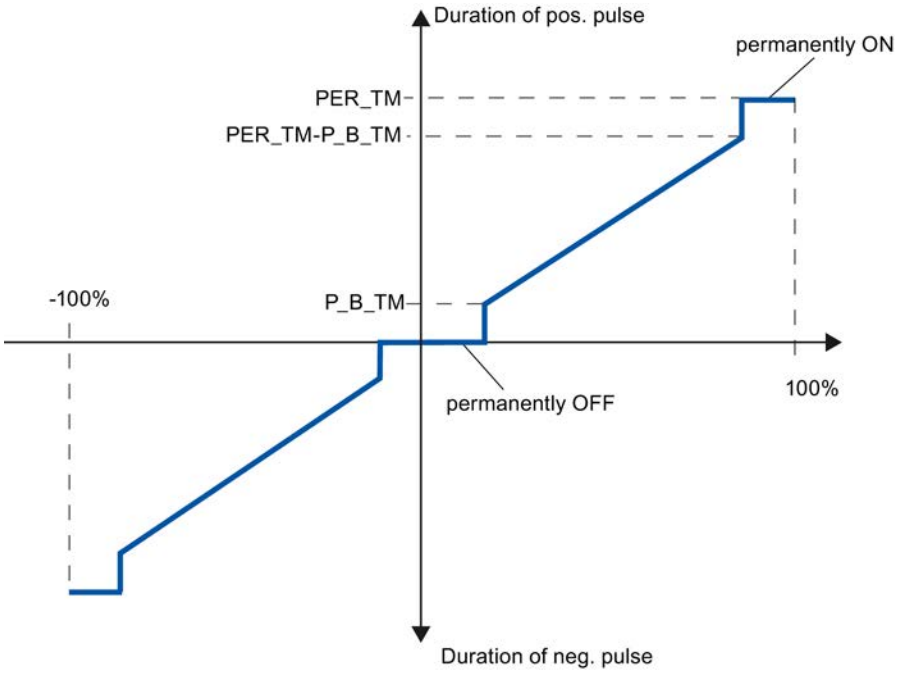
Three actuating signal states can be generated in "three-step control" mode, e.g. more - off - less, forward - stop - backward, heat - off - cool. The values of the binary output signals QPOS_P and QNEG_P are assigned to the specific states of the actuator. The table shows the example of a temperature control:

Output signals	Heat	off	Cool
QPOS_P	TRUE	FALSE	FALSE
QNEG_P	FALSE	FALSE	TRUE

Based on the input variable, a characteristic curve is used to calculate pulse duration. The form of the characteristic curve is defined by the minimum pulse time or minimum break time

and the ratio factor. The normal value for the ratio factor is 1. The bend points in the characteristic curves are caused by the minimum pulse or minimum break times P_B_TM.

The following figure shows a symmetrical curve of the three-step controller (ratio factor = 1).



Three-step control asymmetrical

Using the ratio factor RATIOFAC, the ratio of the duration of positive to negative pulses can be changed. In a thermal process, for example, this would allow different system time constants for heating and cooling.

Ratio factor < 1

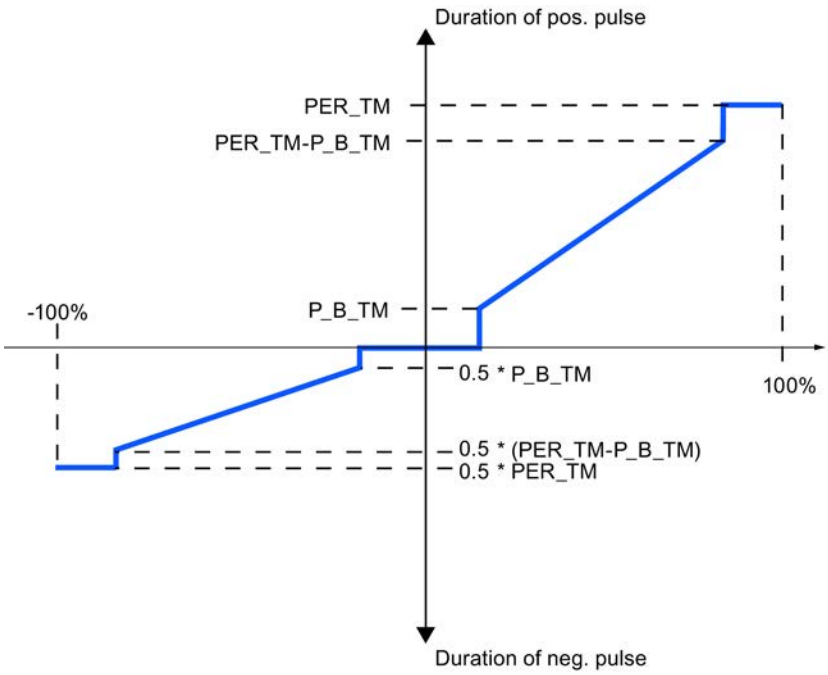
The pulse duration at the negative pulse output calculated from the input variable multiplied by the period duration is reduced by the ratio factor.

Positive pulse duration = INV /100 * PER_TM

Negative pulse duration = INV / 100 * PER_TM * RATIOFAC

The negative pulse duration is multiplied with the ratio factor.

The following figure shows the asymmetric curve of the three-step controller (ratio factor = 0.5):



The ratio factor RATIOFAC also influences the minimum pulse or minimum break time. The threshold value for negative pulses is multiplied by the ratio factor.

Ratio factor > 1

The pulse duration at the positive pulse output calculated from the input variable multiplied by the period duration is reduced by the ratio factor.

Positive pulse duration = INV / 100 * PER_TM / RATIOFAC

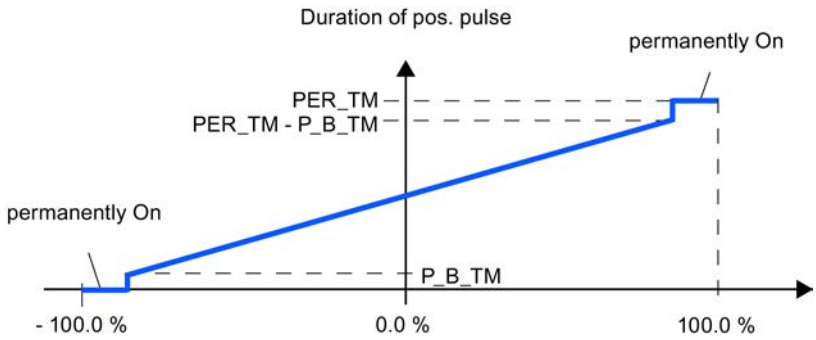
Negative pulse duration = INV / 100 * PER_TM

The ratio factor RATIOFAC also influences the minimum pulse or minimum break time. The threshold value for positive pulses is divided by the ratio factor.

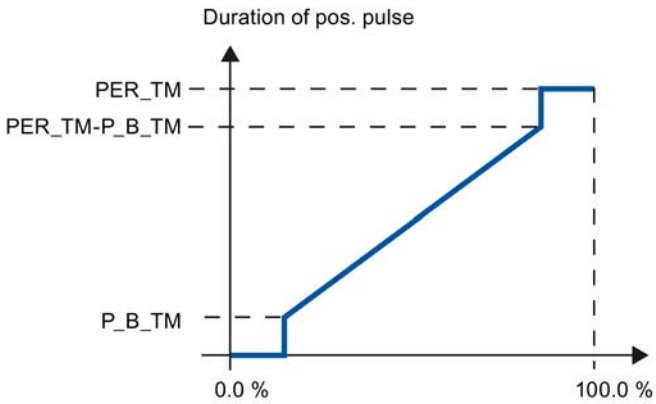
PULSEGEN_M two-step control

In two-step control, only the positive pulse output QPOS_P of PULSEGEN_M is connected to the on/off actuator in question. Depending on the manipulated value range used, the two-step controller has a bipolar or a unipolar manipulated value range.

Two-step control with bipolar manipulated variable range (-100% to 100%)



Two-step control with unipolar manipulated variable range (0% to 100%)



The negated output signal is available at QNEG_P if the connection of the two-step controller in the control loop requires a logically inverted binary signal for the actuating pulses.

Pulse	Actuator On	Actuator Off
QPOS_P	TRUE	FALSE
QNEG_P	FALSE	TRUE

Input parameters PULSEGEN_M

Parameter	Offset	Data type	Preassigned value	Description
INV	0.0	REAL	0.0	Input variable -100.0..100.0 (%)
PER_TM	4.0	TIME	T#1s	Period PER_TM >= 20*CYCLE
P_B_TM	8.0	TIME	T#50ms	Minimum pulse or minimum break time P_B_TM >= CYCLE

Parameter	Offset	Data type	Preassigned value	Description
RATIOFAC	12.0	REAL	1.0	Ratio factor 0.1..10.0 (dimensionless)
STEP3_ON	16.0	BOOL	TRUE	Three-step control ON
ST2BI_ON	16.1	BOOL	FALSE	Activate two-step control for bipolar manipulated variable range
MAN_ON	16.2	BOOL	FALSE	Manual mode ON
POS_P_ON	16.3	BOOL	FALSE	Activate positive pulse
NEG_P_ON	16.4	BOOL	FALSE	Activate negative pulse
SYN_ON	16.5	BOOL	TRUE	Activate synchronization
COM_RST	16.6	BOOL	FALSE	Complete restart
CYCLE	18.0	TIME	T#10ms	Sampling time CYCLE >= 1ms

Output parameters PULSEGEN_M

Parameter	Offset	Data type	Preassigned value	Description
QPOS_P	22.0	BOOL	FALSE	Output signal positive pulse
QNEG_P	22.1	BOOL	FALSE	Output signal negative pulse

RMP_SOAK

Description RMP_SOAK

You use the RMP_SOAK instruction to specify different setpoints, primarily as a function of the process time.

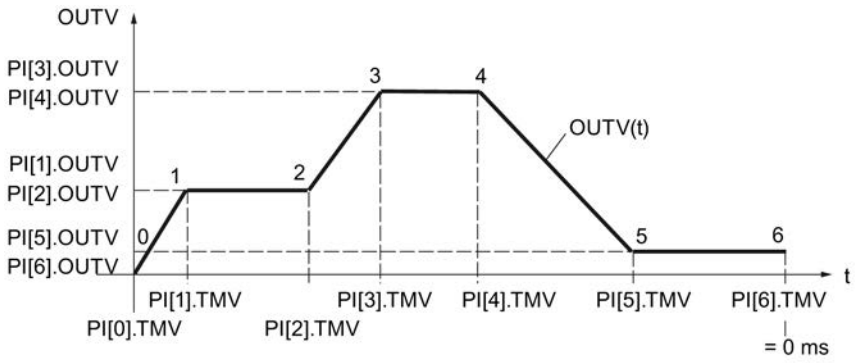
Operating principle

You specify the time slice of the ramp/soak profile in a shared data block. On each call of the instruction, the setpoint valid for this time is output.

You enter the number of time slices in the shared data block in the NBR_PTS variable. You specify the number of the shared block in input parameter DB_NBR.

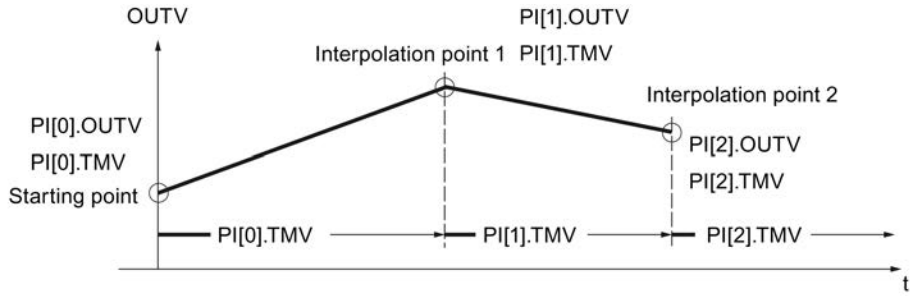
For each time slice, you enter the setpoint in PI[i].OUTV and the time in PI[i].TMV.

Example of a ramp/soak profile with starting point and time slices.



With n time slices, the time value for time slice n = 0 ms.

The method of counting of the time slice values and times is illustrated in the following figure:



Between the time slices, the ramp/soak function interpolates with 0 ≤ n < (NBR_PTS - 1) according to the following function:

$$OUTV(t) = PI[n + 1].OUTV - \frac{RS_TM}{PI[n].TMV} (PI[n + 1].OUTV - PI[n].OUTV)$$

Note

The instruction does not check whether a shared DB with the number DB_NBR actually exists or whether the DB_NBR.NBR_PTS parameter (number of time slices) is suitable for the data block length. If the parameter assignment is incorrect, the CPU changes to the STOP operating mode with the message "Internal system error".

Ramp/soak functions

The instruction has the following functions:

- Activate ramp/soak for single execution
- Pre-assign fixed value to the ramp/soak output
- Activate cyclic operation of the ramp/soak
- Hold processing of the ramp/soak

- Define the processing step and time (the residual time RS_TM and the time slice number TM_SNBR are redefined)
- Update the total processing time and remaining total time

The following table applies for the significance of the control inputs when setting a desired operating mode:

Mode	RMPSK_ ON	DFOUT_ ON	RMP_ HOLD	CONT_ ON	CYC_ ON	TUPDT_ ON	Output signal OUTV
ramp/soak on	TRUE	FALSE	FALSE	**)	FALSE	**)	OUTV(t) Full-scale value is is held after the processing end.
Pre-assign output	TRUE	TRUE	**)	**)	**)	**)	DF_OUTV
Cyclic operation on	TRUE	FALSE	FALSE	**)	TRUE	**)	OUTV(t) After end: Automatic start
Ramp/soak function HOLD	TRUE	FALSE	TRUE	FALSE	**)	**)	Current value of OUTV(t) is held *)
Pre-assign processing step and processing time	TRUE	FALSE	TRUE	TRUE	**)	**)	OUTV (old) *)
			FALSE				Operation is continued with newly defined values.
Update total time	**)	**)	**)	**)	**)	FALSE	Does not influence OUTV
						TRUE	Does not influence OUTV
*) The ramp/soak profile does not have the form configured by the user until the next time slice! **) The operating mode set respectively is executed irrespective of the significance of the control signals in the fields marked with **)							

Complete restart

During a complete restart, output OUTV is reset to 0.0. When DFOUT_ON=TRUE, DF_OUTV is output. The time intervals (0 to NBRPTS-1) between the time slices 0 to NBRPTS are totaled and made available under T_TM. The output QR_S_ACT is reset. Zero is pre-assigned to the outputs NBR_ATMS and RS_TM.

Activating the ramp/soak

On a rising edge at RMPSK_ON, the ramp/soak function is activated. The ramp/soak profile is terminated after the last time slice has been reached. In order to restart the ramp/soak function, you must first set RMPSK_ON = FALSE and then RMPSK_ON = TRUE.

Pre-assigning the output, starting the ramp/soak profile

If the ramp/soak profile is to start with a particular output value, DFOUT_ON = TRUE must be set. In this case, signal value DF_OUTV is applied at the output of the ramp/soak.

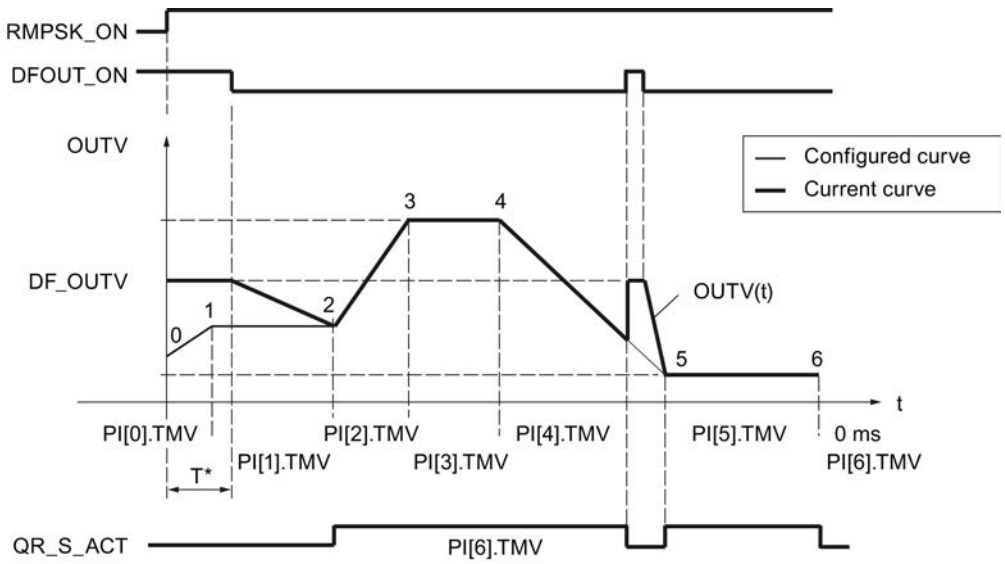
Note

The signal for the output of constant setpoint DFOUT_ON has a higher priority than the start signal for ramp/soak RMPSK_ON.

On changeover (DFOUT_ON = FALSE), OUTV moves linearly from the specified setpoint (DF_OUTV) to the output value of the current time slice number PI[NBR_ATMS].OUTV.

The internal time processing continues running even when a fixed setpoint (RMPSK_ON = TRUE and DFOUT_ON = TRUE) is switched through.

On the start of the ramp/soak profile (RMPSK_ON = TRUE), the fixed setpoint DF_OUTV continues to be output until DFOUT_ON changes from TRUE to FALSE after the duration of T*.



After this moment the time PI[0].TMV and a part of the time PI[1].TMV has expired. OUTV moves from DF_OUTV to PI[2].OUTV, i.e., from the setpoint to time slice 2.

The configured ramp/soak profile is output only starting from time slice 2. The agreement of the configured and current ramp/soak profile is displayed by setting QR_S_ACT = TRUE. On a rising edge at DFOUT_ON during processing of the ramp/soak profile, the output value OUTV jumps to DF_OUTV without a delay.

Cyclic operation activated

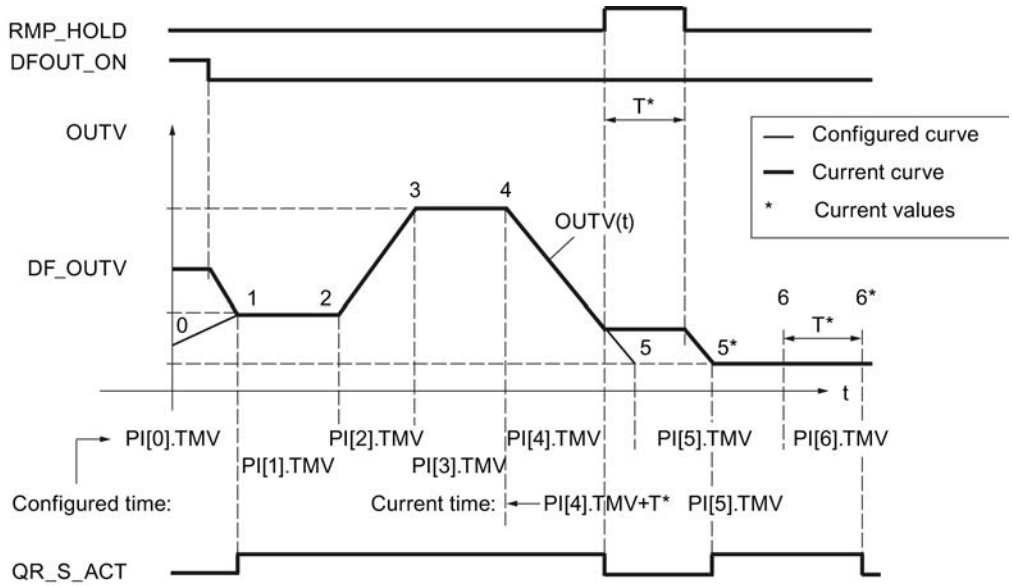
If "Cyclic repetition" operating mode (CYC_ON = TRUE) is activated, the ramp/soak function automatically returns to the starting point after the last time slice value has been output and begins a new run.

There is no interpolation between the last time slice and the starting point. For a smooth transition the following must be true: PI[NBR_PTS].OUTV = PI[0].OUTV.

Ramp/soak function HOLD

With RMP_HOLD = TRUE, the value of the output variable (including the time processing) is frozen. When RMP_HOLD = FALSE is reset, the function resumes at the interruption point PI[x].TMV.

The processing time of the ramp/soak profile is extended by the amount of the time T*. The ramp/soak profile has the configured characteristic from the starting point until the rising edge at RMP_HOLD and from time slice 5* to time slice 6*, i.e., output signal QR_S_ACT has the value TRUE.



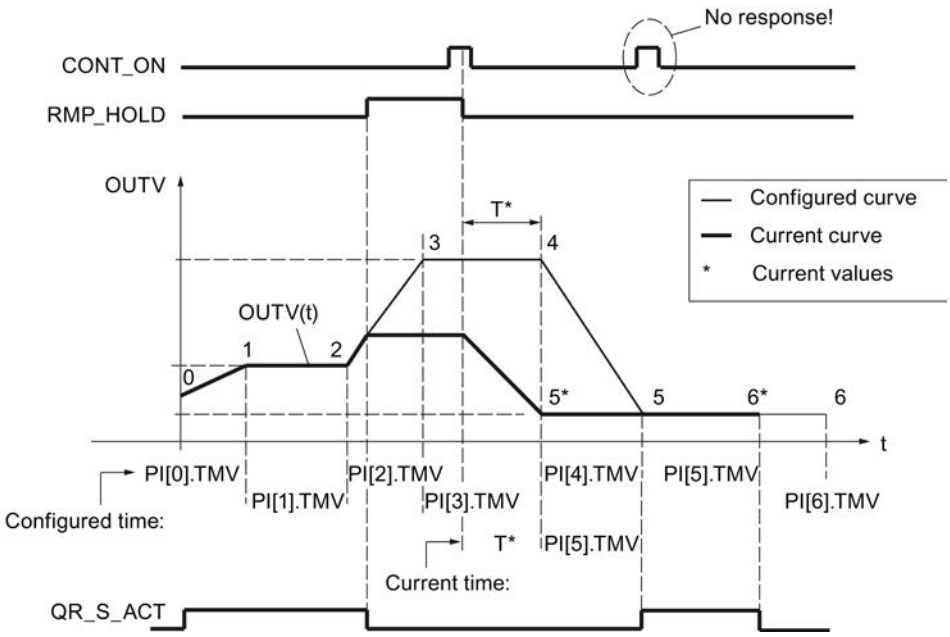
Pre-assigning the processing step and processing time

With CONT_ON, a held ramp/soak profile is restarted at a defined point. If the ramp/soak function is not held, CONT_ON has no effect.

If the control input for continuing (CONT_ON = TRUE) is set, the held ramp/soak function is continued at time slice TM_SNBR. TM_CONT determines the remaining time that the ramp/soak function requires to reach time slice TM_SNBR.

The following figure shows the effect of RMP_HOLD and CONT_ON.

TM_SNBR = 5 and TM_CONT = T*



Time slices 3 and 4 are omitted in this cycle of the ramp/soak function.

On a signal change of RMP_HOLD from TRUE to FALSE, the configured characteristic curve is only reached again starting from time slice 5.

Updating the total time and remaining total time

The current time slice number NBR_ATMS, the current remaining time until the time slice is reached (RS_TM), the total time T_TM, and the total remaining time until the end of the ramp/soak profile (RT_TM) are updated in every cycle.

In the case of online changes of PI[n].TMV, the total time and the total remaining time of the ramp/soak profile change. Since the calculation of T_TM and RT_TM with many interpolations increases the processing time of the function block markedly, the calculation is only carried out at a complete restart or when TUPDT_ON = TRUE. The time intervals PI[0 ... NBR_PTS].TMV between the individual time slices are added up and indicated at the total time T_TM and total remaining time RT_TM outputs.

Please note that determining of the total times requires relatively high CPU runtimes!

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

See also

Processing setpoints (Page 9753)

Input parameters RMP_SOAK

Parameter	Offset	Data type	Preassigned value	Description
DF_OUTV	0.0	REAL	0.0	Default setting of output variable Engineering value range
DB_NBR	4.0	BLOCK_DB	DB 1	Data block number, shared (time slice values) CPU-dependent
TM_SNBR	6.0	INT	0	No. of next time slice for continuation 0 - 255
TM_CONT	8.0	TIME	T#0s	Remaining time for continuing until time slice TM_SNBR Engineering value range
DFOUT_ON	12.0	BOOL	FALSE	Default setting of output variable On
RMPSK_ON	12.1	BOOL	FALSE	Ramp/soak function ON
HOLD	12.2	BOOL	FALSE	Hold the output variable
CONT_ON	12.3	BOOL	FALSE	Continue
CYC_ON	12.4	BOOL	FALSE	Cyclic repetition ON
TUPDT_ON	12.5	BOOL	FALSE	Update total time after change in times (extended block run time!)
COM_RST	12.6	BOOL	FALSE	Complete restart
CYCLE	14.0	TIME	T#1s	Sampling time Engineering value range ≥ 1 ms

Output parameters RMP_SOAK

Parameter	Offset	Data type	Preassigned value	Description
OUTV	18.0	REAL	0.0	Output variable
QR_S_ACT	22.0	BOOL	FALSE	ramp/soak is being executed
NBR_ATMS	24.0	INT	0	Current time slice number (time slice that is being approached)
RS_TM	26.0	TIME	T#0s	Residual slice time
T_TM	30.0	TIME	T#0s	Shared data: Total time ΣTMVI
RT_TM	34.0	TIME	T#0s	Shared data: Residual total time ΣTMVI

The time slices as well as the number of time slices NBRPTS are stored in a shared data block DB_NBR. Output begins with time slice 0 and ends with the time slice NBR_PTS.

Shared data block RMP_SOAK

Parameter	Data type	Preassigned value	Description
NBR_PTS	INT	4	Number of time slices -1 1 - 255
PI	Array [0...NBR_PTS] of Struct		Time slice data structure. The array must be declared by 0...NBR_PTS.
PI[0].OUTV	REAL	0.0	Output variable [0] Starting point Engineering value range
PI[0].TMV	TIME	T#1s	Time value [0] Starting point
PI[1].OUTV	REAL	0.0	Output variable [1] Time slice 1 Engineering value range
PI[1].TMV	TIME	T#1s	Time value [1] Time slice 1
PI[2].OUTV	REAL	0.0	Output variable [2] Time slice 2 Engineering value range
PI[2].TMV	TIME	T#1s	Time value [2] Time slice 2
PI[3].OUTV	REAL	0.0	Output variable [3] Time slice 3 Engineering value range
PI[3].TMV	TIME	T#1s	Time value [3] Time slice 3
PI[4].OUTV	REAL	0.0	Output variable [4] Time slice 4 Engineering value range
PI[4].TMV	TIME	T#0s	Time value [4] Time slice 4 0 ms

ROC_LIM

Description ROC_LIM

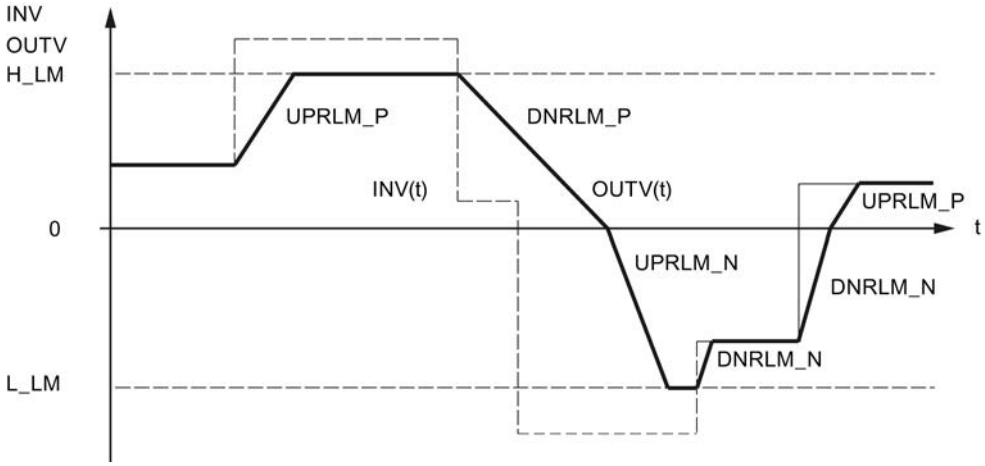
Ramp functions are used when the process does not tolerate an input step. This is for example the case when a gearbox is inserted between a motor and the load to be driven and an excessively rapid increase of the motor speed would result in overloading of the gearbox. The instruction limits the rate of change of an output value. Step jump functions become ramp functions.

Operating principle

A total of four ramps can be assigned for the positive and negative value range of input variable INV for use in determining the output variable:

- UPRLM_P: Rising values in the positive range
  OUTV > 0 and |OUTV| rising
- DNRLM_P: Falling values in the positive range
  OUTV > 0 and |OUTV| falling

- UPRLM_N: Rising values in the negative range
  OUTV < 0 and |OUTV| rising
- DNRLM_N: Falling values in the negative range
  OUTV < 0 and |OUTV| falling



The assigned values of the ramps refer to a rising or falling slope of the output variable per second.

If UPRLM_P = 10.0 and |INV| are rising, OUTV may increase by a maximum of 10.0 in 1 s.

If the input variable rises at a higher rate, OUTV is increased as follows, according to the sampling time CYCLE per call:

CYCLE [s]	Increase of OUTV
1	10.0
0.1	1.0
0.01	0.1

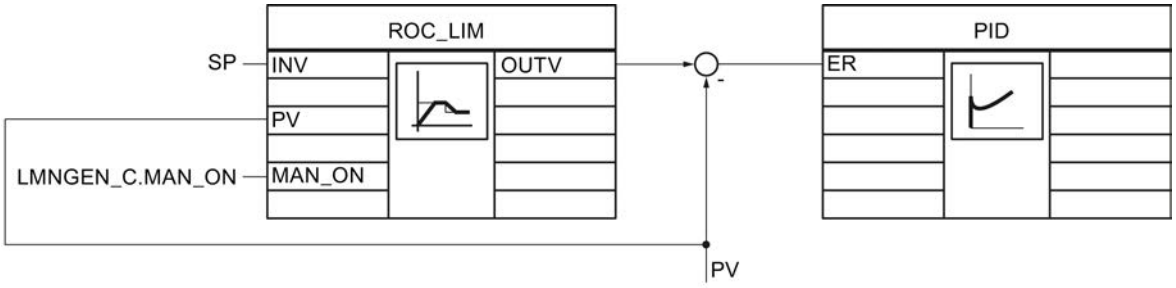
The limiting of the rate of change is indicated in the QUPRLM_P, QDNRLM_P, QUPRLM_N, and QDNRLM_N parameters.

The output variable can be limited by H_LM and L_LM. This is indicated in the QH_LM and QL_LM parameters.

Functions of the instruction

You can use the DFOUT_ON, TRACK, and MAN_ON parameters to influence how the instruction works.

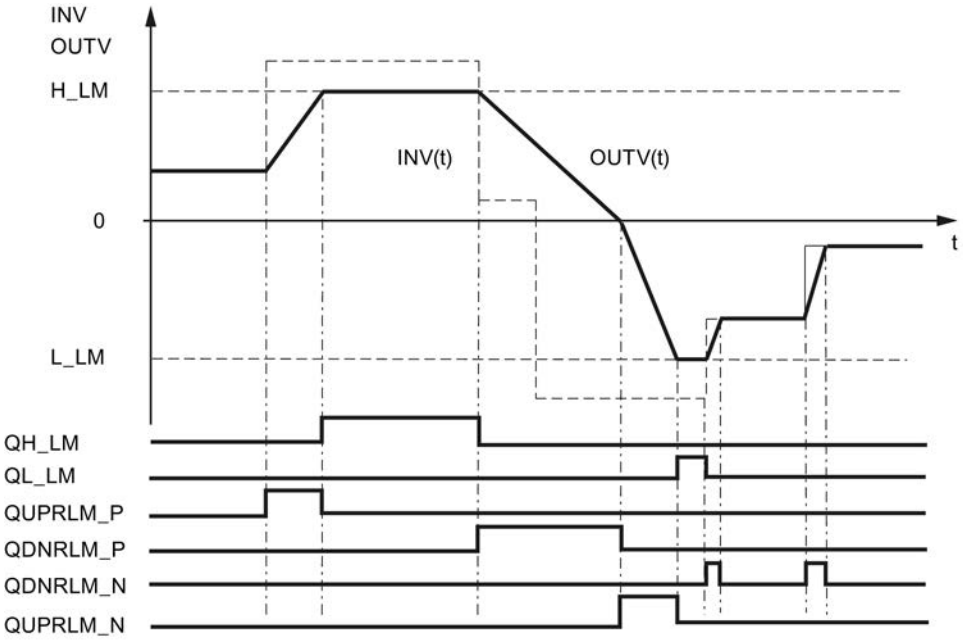
- DFOUT_ON = TRUE
  If DFOUT_ON = TRUE is set, DF_OUTV is output at output OUTV; on a falling edge, OUTV moves from DF_OUTV to INV, and on a rising edge, OUTV moves from INV to DF_OUTV.
- TRACK = TRUE
  Bit TRACK = TRUE is set in order to track OUTV = INV. Since the input variable is switched through directly to the output variable, step changes in the input variable are also output.
- MAN_ON = TRUE
  With MAN_ON = TRUE, you can perform a bumpless changeover between manual and automatic mode. For this purpose, the ROC_LIM instruction is inserted in the setpoint branch directly upstream of the control deviation. The process value is interconnected with input PV and the manual-automatic bit, e.g., LMNGEN_C.MAN_ON is interconnected with input MAN_ON.



On changeover to manual mode (MAN_ON = TRUE), the value applied to input PV is switched through immediately to output OUTV. Since the setpoint and the process value are equal, the control deviation becomes zero and the controller is in a steady-state idle condition. On reset to automatic mode (MAN_ON = FALSE), output OUTV moves from the current value PV to input value INV along a ramp. This causes a bumpless changeover of the controller structure from manual to automatic mode.
When MAN_ON = TRUE, the H_LM and L_LM limits have no effect; DFOUT_ON is not taken into consideration.

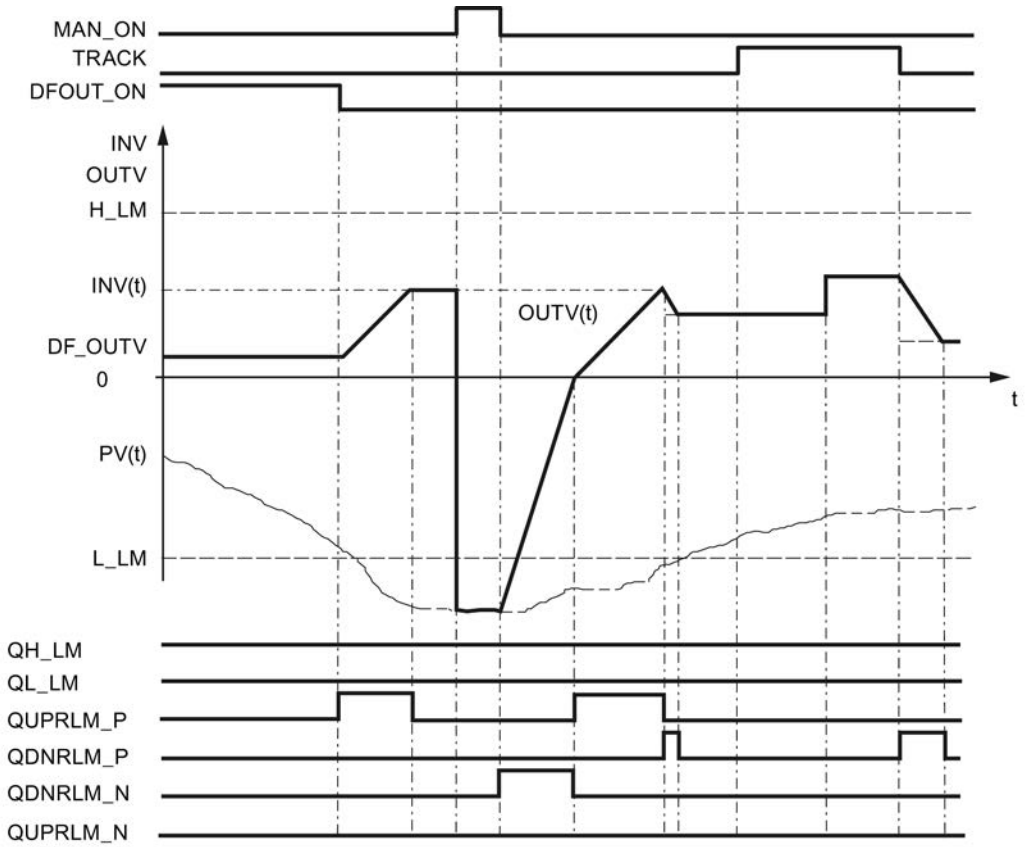
Example 1

When MAN_ON, TRACK, DFOUT_ON equal FALSE , the signal outputs exhibit the following characteristics:



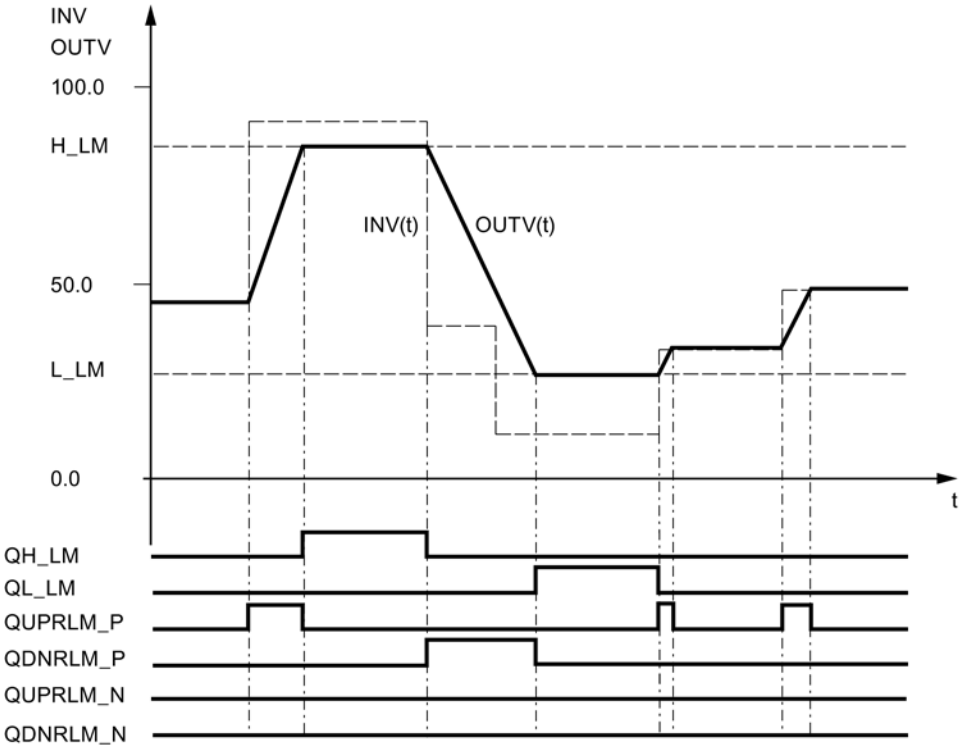
Example 2

Smooth manual-automatic changeover



Example 3

Ramp that is only used in the positive range of values.



Complete restart

During a complete restart, the output OUTV is reset to 0.0. If DFOUT_ON = TRUE is set, DF_OUTV is output. All the signal outputs are set to FALSE.

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

See also

Processing setpoints (Page 9753)

Input parameters ROC_LIM

Parameter	Offset	Data type	Preassigned value	Description
INV	0.0	REAL	0.0	Input variable Engineering value range
UPRLM_P	4.0	REAL	10.0	Rise limiter in the positive range Increment per second [1/s] > 0.0
DNRLM_P	8.0	REAL	10.0	Fall limiter in the positive range Increment per second [1/s] > 0.0
UPRLM_N	12.0	REAL	10.0	Rise limiter in the negative range Increment per second [1/s] > 0.0
DNRLM_N	16.0	REAL	10.0	Fall limiter in the negative range Increment per second [1/s] > 0.0
H_LM	20.0	REAL	100.0	High limit Engineering value range > L_LM
L_LM	24.0	REAL	0.0	Low limit Engineering value range < H_LM
PV	28.0	REAL	0.0	Controlled variable Engineering value range
DF_OUTV	32.0	REAL	0.0	Default setting of output variable Engineering value range
DFOUT_ON	36.0	BOOL	FALSE	Default setting of output variable On
TRACK	36.1	BOOL	FALSE	Tracking OUTV=INV
MAN_ON	36.2	BOOL	FALSE	Manual mode ON
COM_RST	36.3	BOOL	FALSE	Complete restart
CYCLE	38.0	TIME	T#1s	Sampling time

Output parameters ROC_LIM

Parameter	Offset	Data type	Preassigned value	Description
OUTV	42.0	REAL	0.0	Output variable
QUPRLM_P	46.0	BOOL	FALSE	Rise limiter in the positive range triggered
QDNRLM_P	46.1	BOOL	FALSE	Fall limiter in the positive range triggered

Parameter	Offset	Data type	Preassigned value	Description
QUPRLM_N	46.2	BOOL	FALSE	Rise limiter in the negative range triggered
QDNRLM_N	46.3	BOOL	FALSE	Fall limiter in the negative range triggered
QH_LM	46.4	BOOL	FALSE	High limit triggered
QL_LM	46.5	BOOL	FALSE	Low limit triggered

SCALE_M

Description SCALE_M

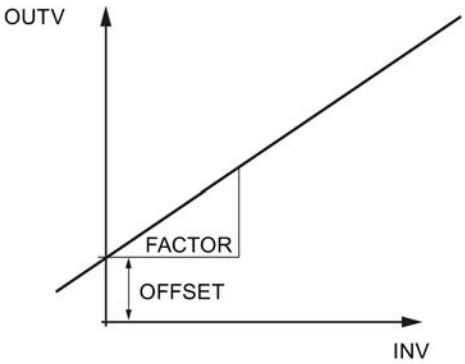
In the case of process values, the value supplied by the encoder often lies in a range that is unfavorable for the user (for example, 0 to 10 V corresponds to 0 to 1200 ºC or 0 to 10 V corresponds to 0 to 3000 rpm). By adapting the setpoint or the process value, both process variables lie in the same range of values.

Operating principle

The instruction scales an analog input variable. Analog input variable INV is transformed to output variable OUTV based on the scaling line. The scaling line is defined by the slope (FACTOR) and the distance (OFFSET) between OUTV when INV=0 and coordinate axis OUTV=0 is defined.

Formula:

OUTV = INV * FACTOR + OFFSET



Complete restart

The block does not have a complete restart routine.

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

See also

Processing setpoints (Page 9753)
Processing the process value (Page 9755)
Applying the disturbance variable (Page 9759)

Input parameters SCALE_M

Parameter	Offset	Data type	Preassigned value	Description
INV	0.0	REAL	0.0	Input variable Engineering value range
FACTOR	4.0	REAL	1.0	Scaling factor
OFFSET	8.0	REAL	0.0	Offset Engineering value range

Output parameters SCALE_M

Parameter	Offset	Data type	Preassigned value	Description
OUTV	12.0	REAL	0.0	Output variable

SP_GEN

Description SP_GEN

An output value can be changed using the SP_GEN instruction and two inputs in order to specify the setpoint manually. To carry out a change in small steps, the instruction should have a sampling time of ≤100 ms.

Operating principle

DFOUT_ON, OUTVUP, and OUTVDN have the following effect on OUTV:

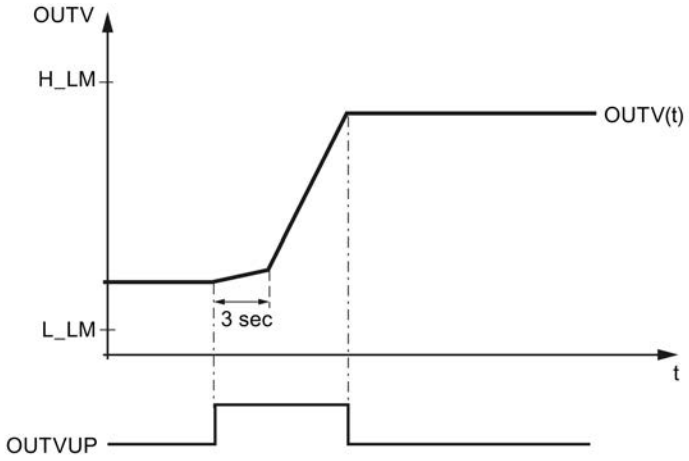
DFOUT_ON	OUTVDN	OUTVUP	OUTV
TRUE	Any	Any	DF_OUTV
FALSE	TRUE	TRUE	OUTV unchanged
	FALSE		OUTV rising
	TRUE	FALSE	OUTV falling
	FALSE		OUTV unchanged

If DFOUT_ON = TRUE is set, DF_OUTV is output at the output. The change in OUTV is a step change. The changeover to DFOUT_ON = FALSE is bumpless.

The OUTVUP input can be used to continuously increase output variable OUTV within the limits H_LM and L_LM; accordingly, the OUTVDN output is used for reducing. The rate of change of OUTV depends on the limits.

- During the first 3 s after setting of OUTVUP or OUTVDN:
  dLMN/dt = (LMN_HLM - LMN_LLM) / 100 s
- Afterwards:
  dLMN/dt = (LMN_HLM - LMN_LLM) / 10 s

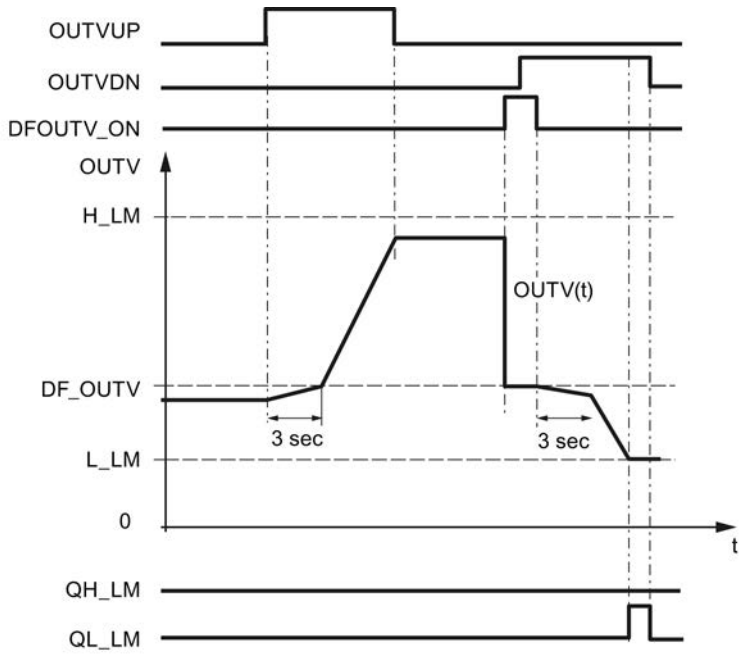
The figure shows the change in the output value when OUTVUP is set.



OUTVUP and OUTVDN are executed only if DFOUT_ON = FALSE.

Output variable OUTV is limited by H_LM and L_LM. This is indicated in the QH_LM and QL_LM parameters.

The figure shows how OUTVUP, OUTVDN, and DFOUTV_ON. affect OUTV.



Complete restart

During a complete restart, the output OUTV is set to 0.0. If DFOUT_ON = TRUE, DF_OUTV is output. The limits remain valid at a compete restart.

Block-internal limits

No values are limited block-internally. The parameters are not checked.

See also

Processing setpoints (Page 9753)

Input parameters SP_GEN

Parameter	Offset	Data type	Preassigned value	Description
DF_OUTV	0.0	REAL	0.0	Default setting of output variable Engineering value range
H_LM	4.0	REAL	100.0	High limit Engineering value range > L_LM
L_LM	8.0	REAL	0.0	Low limit Engineering value range < H_LM

Parameter	Offset	Data type	Preassigned value	Description
OUTVUP	12.0	BOOL	FALSE	OUTV up
OUTVDN	12.1	BOOL	FALSE	OUTV down
DFOUT_ON	12.2	BOOL	FALSE	Default setting of output variable On
COM_RST	12.3	BOOL	FALSE	Complete restart
CYCLE	14.0	TIME	T#100ms	Sampling time

Output parameters SP_GEN

Parameter	Offset	Data type	Preassigned value (on complete restart)	Description
OUTV	18.0	REAL	0.0	Output variable
QH_LM	22.0	BOOL	FALSE	High limit triggered
QL_LM	22.1	BOOL	FALSE	Low limit triggered

SPLT_RAN

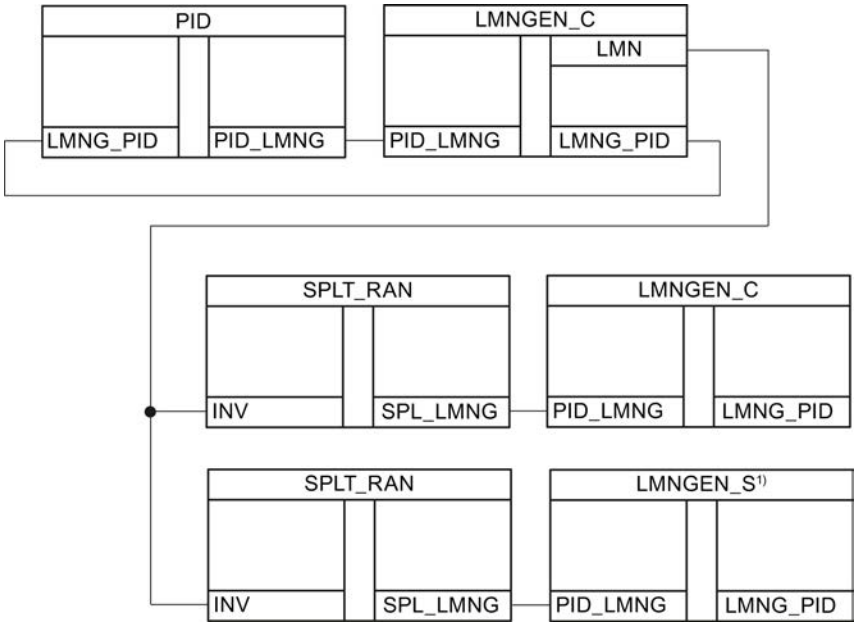
Description SPLT_RAN

The instruction is required, for example to split a temperature control loop into an actuator for cooling and an actuator for heating.

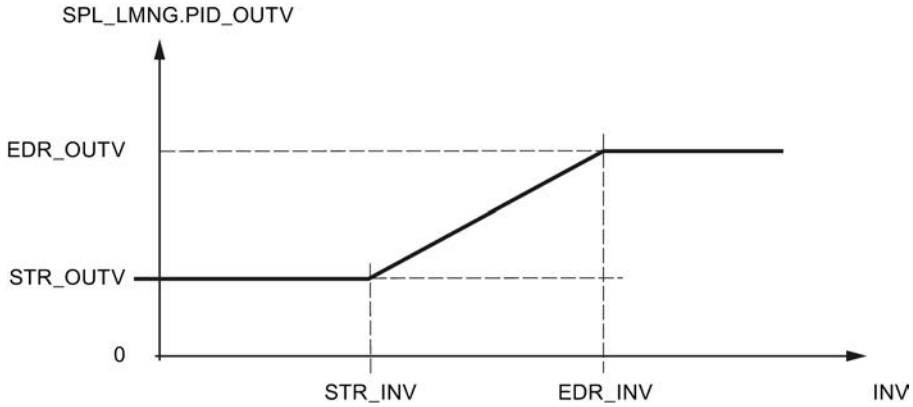
Operating principle

The manipulated variable of a PID controller is spilt into several ranges. The instruction must be called once per range and be interconnected with manipulated variable processing block

LMNGEN_C or LMNGEN_S. If you interconnect LMNGEN_S, position feedback must be activated (LMNR_ON = TRUE).



If input value INV is within the range STR_INV and EDR_INV, output value SPL_LMNG.PID_OUTV is output within the range STR_OUTV and EDR_OUTV. If INV < STR_INV,, then STR_OUTV is output. If INV > STR_INV,, then EDR_OUTV is output.



Complete restart

The block does not have a complete restart routine.

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

Input parameters SPLT_RAN

Parameter	Offset	Data type	Preassigned value	Description
INV	0.0	REAL	0.0	Input variable Engineering value range
STR_INV	4.0	REAL	0.0	Start of range INV Engineering value range
EDR_INV	8.0	REAL	50.0	End of range INV Engineering value range
STR_OUTV	12.0	REAL	0.0	Start of range OUTV Engineering value range
EDR_OUTV	16.0	REAL	100.0	End of range OUTV Engineering value range

Output parameters SPLT_RAN

Parameter	Offset	Data type	Description
SPL_LMNG	20.0	STRUC	PID-LMNGEN interface

SWITCH

Description SWITCH

The instruction is used as an input and/or output multiplexer of 2 input/output variables.

Operating principle

The instruction connects one of two analog input values to one of two output values. The INV1_ON and OUTV1_ON switches determine how the instruction operates.

INV1_ON	OUTV1_ON	OUTV1	OUTV2
FALSE	FALSE	Unchanged	INV2
TRUE	FALSE	Unchanged	INV1
FALSE	TRUE	INV2	Unchanged
TRUE	TRUE	INV1	Unchanged

Unchanged means that the value from the last cycle is retained in the output parameter.

Complete restart

On a complete restart, OUTV1 = 0.0 and OUTV2 = 0.0 are set.

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

Input parameters SWITCH

Parameter	Offset	Data type	Preassigned value	Description
INV1	0.0	REAL	0.0	Input variable 1 Engineering value range
INV2	4.0	REAL	0.0	Input variable 2 Engineering value range
INV1_ON	8.0	BOOL	FALSE	Input variable INV1 is switched through
OUTV1_ON	8.1	BOOL	FALSE	Output variable OUTV1 is switched through
COM_RST	8.2	BOOL	FALSE	Complete restart

Output parameters SWITCH

Parameter	Offset	Data type	Preassigned value	Description
OUTV1	10.0	REAL	0.0	Output variable 1
OUTV2	14.0	REAL	0.0	Output variable 2

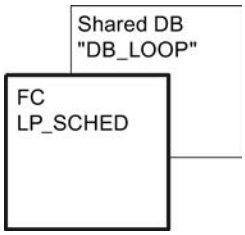
LP_SCHED_M

Description LP_SCHED_M

If many controllers with different sampling times, in particular slow control systems with high sampling times, have to be called, the scope of the priority class model with regard to usable cyclic interrupt levels is insufficient. The controller call distribution function (LP_SCHED) now makes it possible to integrate several controllers with different sampling times in a cyclic interrupt level. The individual controllers can then called up cyclically with their sampling times.

Use of the controller call distribution function is not mandatory. The controller instructions can also be called directly from the OB without the distribution function.

The call distribution of several controllers on a cyclic interrupt level is implemented in the LP_SCHED instruction. The instruction must be called up **before** all control loops. The data for the individual controller calls is stored in a global data block (DB_LOOP).



The call distributor processes the global data block and sets the ENABLE bits depending on the sequence and the parameterized sampling times of the controllers. The time cycle of the cyclic interrupt level is subordinated by this. The individual control loops on this cyclic interrupt level are called up and processed according to their set sampling time. The ENABLE bit has to be reset again after the block call. The block calls and the reset of the ENABLE bits have to be programmed.

The call of individual control loops can be manually blocked. Individual control loops can also be reset (restart).

Note

The instruction does not check whether a global DB with the number DB_NBR actually exists or whether the GLP_NBR parameter (largest control loop number) is suitable for the data block length. If the parameter assignment is incorrect, the CPU changes to STOP with an "internal system error".

See also

Creating a pulse controller (Page 9763)

Operating principle LP_SCHED_M

Restart

When COM_RST = TRUE, the following pre-assignments are set:

Current control loop number: ALP_NBR = 0

The call data of all control loops up to **GLB_NBR** is pre-assigned as follows:

Enable:	ENABLE =	NOT MAN_DIS
Sampling time:	CYCLE =	GV(MAN_CYC)
Restart:	COM_RST =	TRUE
Local call number:	ILP_COU =	0
GV(MAN_CYC) = MAN_CYC is rounded to an integer multiple of TM_BASE*GLP_NBR		

Control loop call via LP_SCHED_M

The LP_SCHED instruction is included by means of three input parameters in the call scheme of the CPU. The time cycle of the cyclic interrupt level is specified at the TM_BASE input. The execution call of the assigned control loops is carried out via a conditional block call from one cyclic interrupt level (for example OB 35). In the course the ENABLE bits in the global data block are sampled.

If the call is carried out from the restart level, input COM_RST = TRUE is set. This call has to be reset again to FALSE in the cyclic interrupt level. The global data block (see previous table) with the time-relevant data of the control loops in the respective cyclic interrupt level is assigned via the input parameter DB_NBR.

Parameter assignment of the control loop call (global DB)

The parameter assignment of the controller call distribution function has to be carried out **without** support by the configuration software.

The data block (DB_LOOP) contains both a parameter that determines the total number of control loops that have to be processed on the respective cyclic interrupt level (max. 256) and a parameter that displays the control loop that is currently being processed:

GLP_NBR	Largest control loop number
ALP_NBR	Number of the control loop processed in the cycle

The number of the respective control loop results from the positioning of its call data in the sequence of the entries in the DB.

The call data of the individual control loops resides structured in the LOOP_DAT field. If control loops are to be added, the field length of LOOP_DAT has to be adapted. To this purpose the ARRAY data type has to be adapted in the declaration view. For example, with 10 control loops, you must enter ARRAY[1..10]. In addition the GLP_NBR parameter has to be adapted in the data view. It may never be larger than the field length.

The parameters COM_RST and CYCLE in the call data must be interconnected to the corresponding input parameters at the FB of the calling control loop. This interconnection is to be programmed by the user. If the ENABLE parameter is set, the corresponding control loop is to be called. The ENABLE bit has to be reset after the controller call. The conditional controller call and the reset of the ENABLE bit have to be programmed by the user.

With the aid of the manually adjustable parameters MAN_CYC/MAN_DIS/MAN_CRST, one can control whether a control loop can or cannot be called. These call data can be modified online - during operation - so long as only the parameters are overwritten and the entire DB is not regenerated. The following meanings apply:

MAN_CYC	The sampling time of the respective controller (is rounded to an integer multiple of TM_BASE * GLP_NBR in CYCLE).
MAN_DIS	Disabling of the controller call
MAN_CRST	Restart for this controller

Call processing

The corresponding control loop is processed in accordance with the previous DB parameter assignment, depending on the value of the ENABLE signal of the controller call data.

The data block is processed from top to bottom. Per cycle the call distribution function moves one control loop number (ALP_NBR) further in the sequence of the DB. The internal counter ILP_COU is counted down by 1 at the same time. If ILP_COU = 0, the call distribution function sets the ENABLE bit of the corresponding control loop. Resetting of the ENABLE bits has to be programmed by the user after the controller call.

The parameter MAN_CYC is transferred to CYCLE during processing:

CYCLE = GV (MAN_CYC), GV = integer multiple

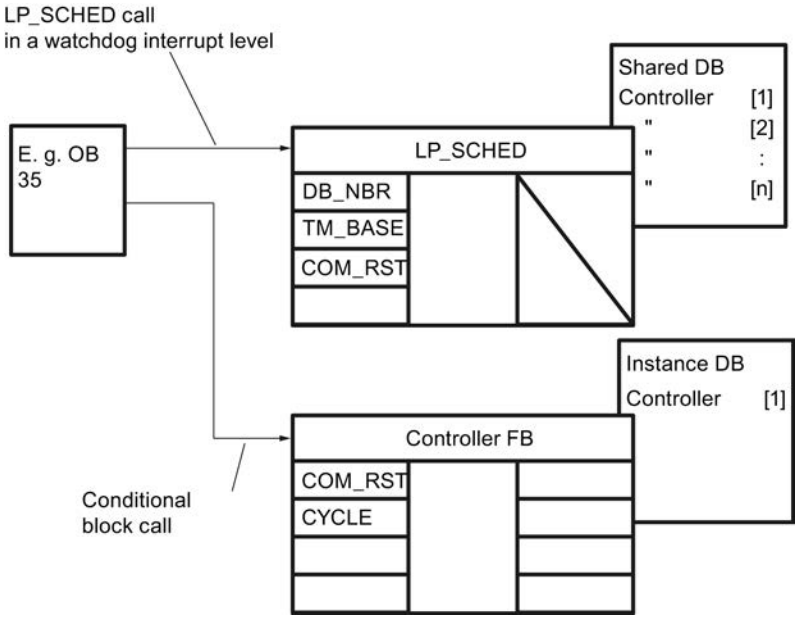


Figure 9-9 Principle of the controller call using the call distribution function LP_SCHED

- Blocking individual control loops:
  If you set the bit "MAN_DIS" by parameter assignment in the DB data block, the "ENABLE" bit is reset to FALSE and the respective control loop is excluded from processing in the call distribution function.
- Resetting individual control loops (restart):
  If you set the "MAN_CRST" bit by parameter assignment in the DB, COM_RST = TRUE and MAN_CRST is subsequently reset. The respective control loop is processed with its restart routine. In the next call cycle, COM_RST = FALSE is also reset automatically.

Note

If a control loop is inserted or deleted, meaning that the whole DB is regenerated, without the call distribution function having to be processed in the restart, the internal control loop counter (ILP_COU[n]) and the parameter for the current control loop number ALP_NBR have to have zero pre-assigned.

Conditions for control loop call via LP_SCHED

In order to ensure that the intervals between the calls of a specific controller remain constant and to utilize the CPU evenly, the cyclic interrupt level may only process one control loop per cycle. The following conditions must therefore be observed during the parameter assignment of the sampling times MAN_CYC in relation to the cycle (TM_BASE):

- The processing times of the individual control loops have to be smaller than the cycle (TM_BASE) of the cyclic interrupt level.
- The sampling time of a control loop (MAN_CYC) has to be an integer multiple (GV) of the product of the time base and number of the controllers to be processed (GLP_NBR): MAN_CYC = GV (TM_BASE * GLP_NBR)

Example for call distribution function

The following example shows the call sequence of four control loops in a cyclic interrupt level (see following figure). Only one control loop is always processed per unit of the time base. The call sequence and thus the displacement times (TD1 ... TD5) result from the sequence of the call data within the global DB.

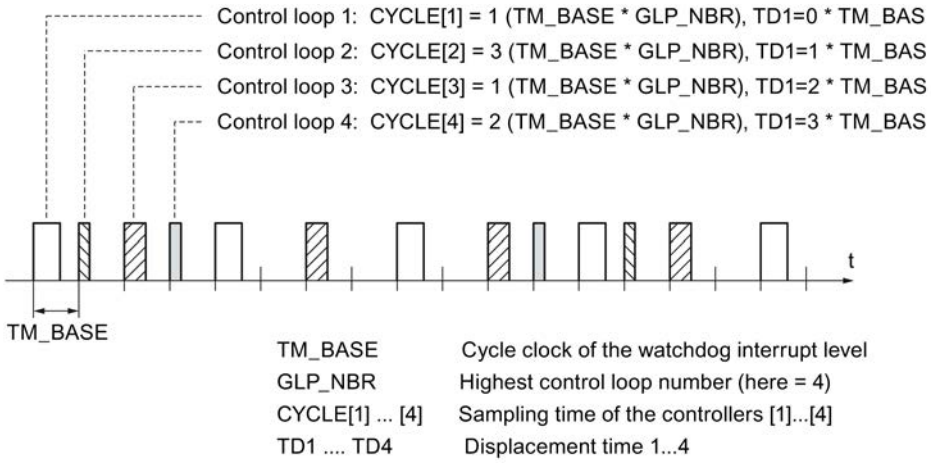


Figure 9-10 Call sequence of four control loops that are processed at varying intervals

Block-internal limits

The values of the input parameters are not limited in the instruction. There is no parameter check.

Input parameters LP_SCHED_M

Parameter	Data type	Description
TM_BASE	TIME	Time base >= 1ms
COM_RST	BOOL	Complete restart
DB_NBR	BLOCK_DB	Data block number

Shared data block DB_LOOP_M

Parameter	Data type	Preassigned value	Description
GLP_NBR	INT	2	Maximum number of controllers 1 - 256
ALP_NBR	INT	0	Number of the current controller 0 - 256
LOOP_DAT	Array [1...GLP_NBR] of Struct		Individual controller data structure. The array must be declared by 1...GLP_NBR.
LOOP_DAT[1].MAN_CYC	TIME	T#1s	Controller 1 data: Manual sampling time >= 1ms
LOOP_DAT[1].MAN_DIS	BOOL	FALSE	Controller 1 data: Disable manual controller call
LOOP_DAT[1].MAN_CRST	BOOL	FALSE	Controller 1 data: Set manual restart
LOOP_DAT[1].ENABLE	BOOL	FALSE	Controller 1 data: Controller enable
LOOP_DAT[1].COM_RST	BOOL	FALSE	Controller 1 data: Restart
LOOP_DAT[1].ILP_COU	INT	0	Controller 1 data: Internal counter
LOOP_DAT[1].CYCLE	TIME	T#1s	Controller 1 data: Sampling time >= 1ms
LOOP_DAT[2].MAN_CYC			Controller 2 data: Manual sampling time
...			

Motion Control

Easy Motion Control

MC_Init

Description

The MC_Init prepares the initialization of all axis blocks by setting all bits in the "Init.Ix" bit array of the axis DB (see "Initialization and parameter changes (Page 10804)").

Operating principle

instruction MC_Init **prevents the unintentional start of the axis** by setting the error displays "Error" = TRUE and "Err.StoppedMotion" = TRUE in the axis DB. If you want to move the axis after instruction MC_Init has been called, you must first cancel the error displays by acknowledging them (see Error displays and error acknowledgment in the axis DB (Page 10820)).

instruction MC_Init checks the parameters in the axis DB. If a parameter error is detected, the "Error" and "Err.ConfigErr" bits are set in the Axis Db for the (group) error display, as well as one of the bits in the "ConfigErr" structure for a more detailed error display.

Parameter errors **cannot be acknowledged**. You must call instruction MC_Init again after correcting the error.

For more information on the error concept, refer to "Error displays and error acknowledgment in the axis DB (Page 10820)" and "Parameter errors (Page 10827)".

Call

Note

instruction MC_Init may only be called when the axis is at a standstill.

Call instruction MC_Init conditionally in the same program block in which you also call the other instructions of this axis. You must call instruction MC_Init in the following situations:

- After every startup of the CPU (OB 100 and OB 101)
- If you change a parameter that is identified by the following symbol in Configuring the Axis TO (Page 10775) without using the configuration software:

Note

Reset the condition used for calling instruction MC_Init once the call has been completed. Otherwise the Easy Motion Control blocks will always repeat only their initialization because the initialization bits in the axis DB are set again.

MC_MoveAbsolute

Description

instruction MC_MoveAbsolute can be used to approach a synchronized axis to the absolute target specified in "Position".

Operating principle

On detecting a positive edge at the "Execute" input, instruction MC_MoveAbsolute determines the direction of travel and accelerates the axis in this direction. On reaching the point of deceleration, the axis is slowed down. The specified velocity does not have to be reached during the movement.

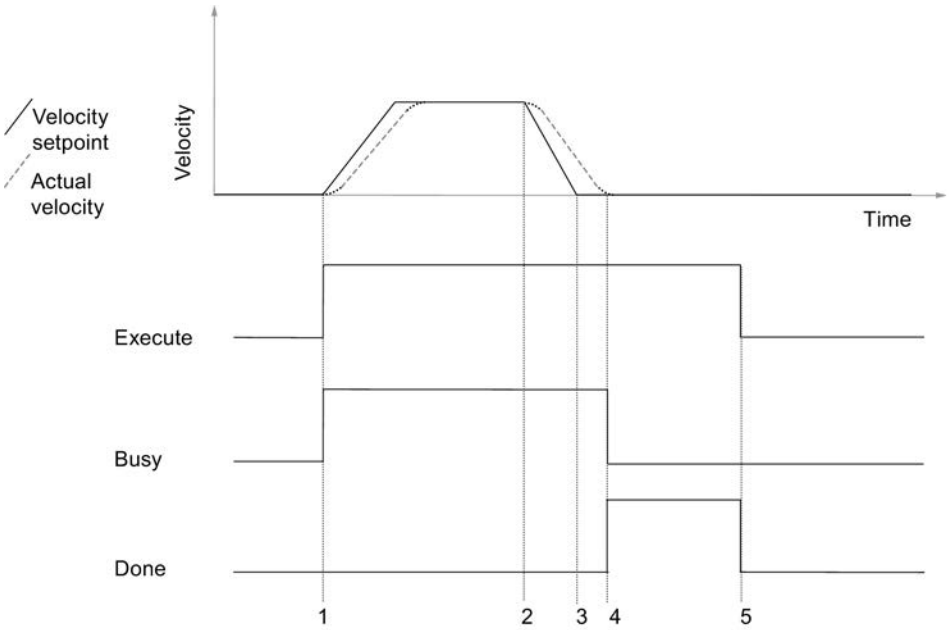
Special feature with a linear axis:

- The direction of travel is determined from the setpoint position at the start and the target.

Special features with a rotary axis:

- instruction MC_MoveAbsolute limits the distance by which you can move the axis from a standstill to less than one rotary axis revolution. Use instruction MC_MoveRelative (Page 7038) to position the rotary axis with several revolutions.
- Polarity selection when starting from axis standstill:
  - Positive direction: target approached in a positive direction
  - Negative direction: target approached in a negative direction
  - Shortest distance: The target is approached in the direction in which the distance between start position and target is equivalent to ≤ ½ rev of the rotary axis.
- Polarity selection when overriding a motion
  - Positive direction: target approached in the positive direction. If the previous motion was in the negative direction, the axis is slowed down and the target is approached in the positive direction. If the target is reached exactly when slowing down with a negative travel direction, the axis then travels one revolution in the positive direction.
  - Negative direction: target approached in the negative direction. If the previous motion was in the positive direction, the axis is slowed down and the target is approached in the negative direction. If the target is reached exactly when slowing down with a positive travel direction, the axis then travels one revolution in the negative direction.
  - Shortest distance: The target is approached in the direction in which the distance between start position and target is equivalent to ≤ ½ rev of the rotary axis. If the braking distance is greater, the axis travels to the target in the original direction.
  - If you have selected travel parameters in such a ways that the braking distance of the previous movement is greater than one rotary axis revolution, the target is approached directly after direction reversal.
  - The entire distance between the start position and target may be greater than one rotary axis revolution.

Signal flow diagram for MC_MoveAbsolute



1 Start of the travel FB
2 Brake application point
3 Setpoint has reach target position
4 Axis has reached the target range
5 Reset of "Execute" resets "Done". The process is completed. If "Execute" is reset before the target position has been reached, "Done" is set for one cycle after the target position has been reached and is then reset again.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	Start of positioning at a positive edge.
Position	INPUT	REAL	Absolute target position in [length unit] Rotary axis: Start of rotary axis ≤ position < end of rotary axis
Velocity	INPUT	REAL	Axis velocity in [length unit/s]
Acceleration	INPUT	REAL	Axis acceleration in [length unit/s^2]
Deceleration	INPUT	REAL	Axis deceleration in [length unit/s^2]
Direction	INPUT	INT	Polarity selection for travel with rotary axis -1 = Negative direction 0 = Shortest distance 1 = Positive direction

Parameter	Declaration	Data type	Description
Busy	OUTPUT	BOOL	1 = Job in progress.
Done	OUTPUT	BOOL	1 = Job completed without error. The bit remains active at least for one call of the instruction until "Execute" is reset.
CommandAborted	OUTPUT	BOOL	1 = Job was canceled by another job. The bit remains active at least for one call of the instruction until "Execute" is reset.
Error	OUTPUT	BOOL	1 = Job was canceled due to an axis error. The bit remains active at least for one call of the instruction until "Execute" is reset.

UDT2

A UDT2 data type is listed and reserved in the block properties of MC_MoveAbsolute. This means that no separate UDT2 data type may be used in the same project. This would result in error messages and inconsistencies within the project. If you want to use a UDT2 data type, this UDT must be renamed, for example, to UDT3.

MC_MoveRelative

Description

instruction MC_MoveRelative can be used to traverse an axis by the distance that is specified in the "Distance" parameter and that is relative to the setpoint at the start of axis movement. The sign of the "Distance" parameter determines the direction.

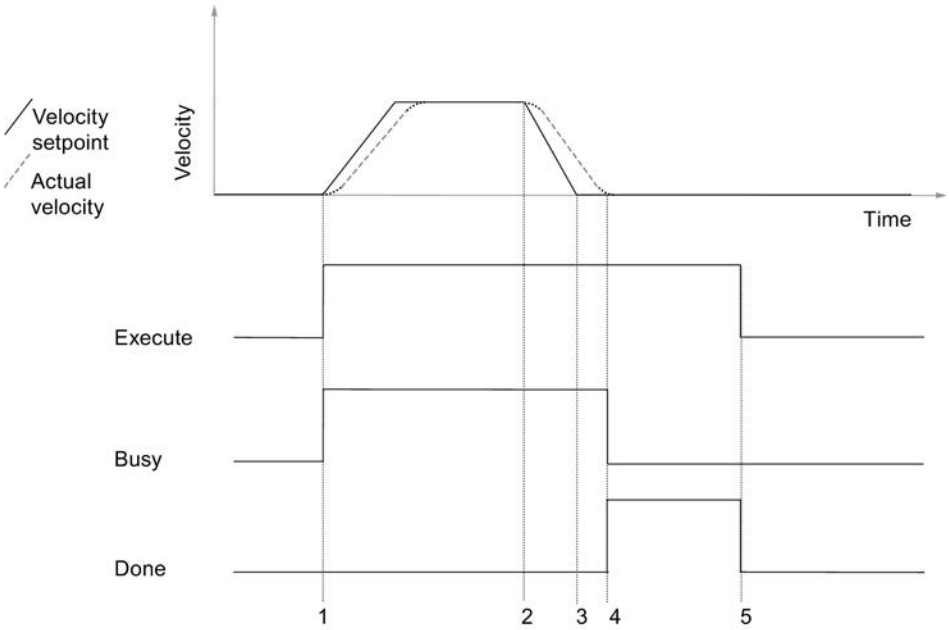
Operating principle

On detecting a positive edge at input "Execute", instruction MC_MoveRelative accelerates the axis in the direction that is specified in the "Distance" parameter. On reaching the point of deceleration, the axis is slowed down. The specified velocity does not have to be reached during the movement. The target is determined from the setpoint position at the start of travel and the distance to be traveled.

Special feature of rotary axes:

- You can realize travel over multiple revolutions by setting values in "Distance" that are greater than one revolution of the of the round axis.

Signal flow diagram for MC_MoveRelative



1 Start of the travel FB
2 Brake application point
3 Setpoint has reach target position
4 Axis has reached the target range
5 Reset of "Execute" resets "Done". The process is completed. If "Execute" is reset before the target position has been reached, "Done" is set for one cycle after the target position has been reached and is then reset again.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	Start of positioning at a positive edge.
Distance	INPUT	REAL	Distance to be traversed in [length unit], relative to the last position setpoint. The sign determines the direction of travel.
Velocity	INPUT	REAL	Axis velocity in [length unit/s]
Acceleration	INPUT	REAL	Axis acceleration in [length unit/s^2]
Deceleration	INPUT	REAL	Axis deceleration in [length unit/s^2]
Busy	OUTPUT	BOOL	1 = Job in progress.
Done	OUTPUT	BOOL	1 = Job completed without error. The bit remains active at least for one call of the instruction until "Execute" is reset.

Parameter	Declaration	Data type	Description
CommandAborted	OUTPUT	BOOL	1 = Job was canceled by another job. The bit remains active at least for one call of the instruction until "Execute" is reset.
Error	OUTPUT	BOOL	1 = Job was canceled due to an axis error. The bit remains active at least for one call of the instruction until "Execute" is reset.

UDT2

A UDT2 data type is listed and reserved in the block properties of MC_MoveRelative. This means that no separate UDT2 data type may be used in the same project. This would result in error messages and inconsistencies within the project. If you want to use a UDT2 data type, this UDT must be renamed, for example, to UDT3.

MC_MoveJog

Description

instruction MC_MoveJog can be used to move an axis using two level-controlled directional inputs "JogPos" and "JogNeg" (jogging).

Operating principle

Once you have set a directional input, MC_MoveJog accelerates the axis to the specified velocity. The axis moves in this direction as long as the directional input remains set. If you reset the directional input again, the axis slows down.

The two directional inputs are evaluated with the following dependencies:

- As long as one direction is selected, setting the opposite direction has no effect. No error is produced.
- To change direction during travel, set the other directional input to TRUE and at the same time set the originally selected directional input to FALSE.
- A start error will be generated if both inputs are set at the start of a movement ("Err.StartErr"; refer to Errors with soft stop (Page 10822)).

Special features of linear axes:

- If the maximum travel distance during travel with a linear axis is exceeded, the MC_MoveJog instruction ramps down the axis to a standstill and the acknowledgeable error message "Err.DistanceErr" is output (refer to Sequence of Travel Movements (Page 10805)). Then the axis can be moved again after the error has been acknowledged.
- A **non-synchronized** linear axis is not monitored and can therefore travel right to the physical end of the axis without slowing down. You must therefore complete the motion in time.

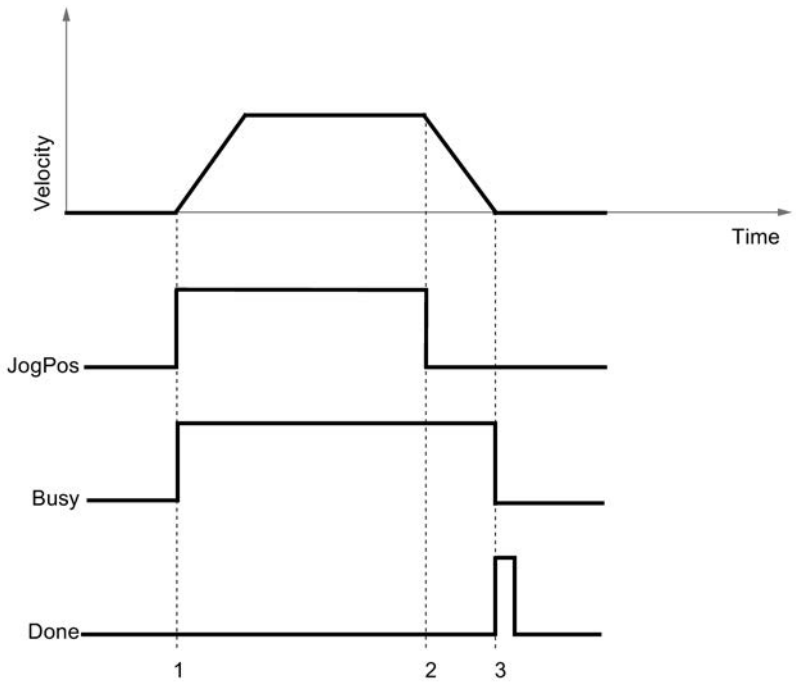
> **WARNING**
>
> **Non-synchronized linear axis**
>
> Injury to persons and property may occur.
>
> instruction MC_MoveJog can be used to traverse a non-synchronized linear axis to its mechanical end.
>
> To avoid injury to persons and damage to objects, take the following precautions:
>
> - Install an EMERGENCY STOP switch in the vicinity of the computer. This is the only way to ensure that the system is reliably switched off in the event of a computer or software failure.
> - Install safety limit switches for direct control of the power units of all drives.
> - Ensure that nobody has access to the area of the system where there are moving parts.

- A **synchronized** linear axis is decelerated when approaching the end of the working range and stops at the software limit switch. The acknowledgeable error message "Err.DistanceErr", as well as the "Err.SWLimitMinExceeded" or "Err.SWLimitMaxExceeded" error messages will be output. After the error has been acknowledged, it is only possible to travel in the opposite direction.

Special features of rotary axes:

- instruction MC_MoveJog does not limit the traversing distance of a rotary axis (infinite rotation).

Signal flow diagram for MC_MoveJog



1 Motion is started by setting "JogPos".
2 Motion is stopped by resetting "JogPos".
3 When the axis reaches a standstill, "Busy" is reset, and "Done" is set for one cycle.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
JogPos	INPUT	BOOL	1 = Movement in positive direction as long as "JogPos" is set.
JogNeg	INPUT	BOOL	1 = Movement in negative direction as long as "JogNeg" is set.
Velocity	INPUT	REAL	Axis velocity in [length unit/s]
Acceleration	INPUT	REAL	Axis acceleration in [length unit/s^2]
Deceleration	INPUT	REAL	Axis deceleration in [length unit/s^2]
Busy	OUTPUT	BOOL	1 = Job in progress.
Done	OUTPUT	BOOL	1 = Job completed without error. After the axis has reached a standstill, the bit is set for exactly for the duration of one call of the instruction.

Parameter	Declaration	Data type	Description
CommandAborted	OUTPUT	BOOL	1 = Job was canceled by another job. The bit remains active until the reset of "JogPos" or "JogNeg".
Error	OUTPUT	BOOL	1 = Job was canceled due to an axis error. The bit remains active until the reset of "JogPos" or "JogNeg".

UDT2

A UDT2 data type is listed and reserved in the block properties of MC_MoveJog. This means that no separate UDT2 data type may be used in the same project. This would result in error messages and inconsistencies within the project. If you want to use a UDT2 data type, this UDT must be renamed, for example, to UDT3.

MC_Home

Description

instruction MC_Home is used for the following tasks:

- Reference point approach with incremental encoder ("Velocity" > 0)
- Set reference point with incremental and absolute encoder ("Velocity" = 0)
  The set reference point function for absolute encoders is also known as absolute encoder adjustment.

Operating principle

Reference point approach ("Velocity" > 0)

Once "Execute" has been set, the axis travels at the selected velocity and in the selected direction until the position decoder module distance and the input driver detect that the axis has reached the reference point. Once the reference point has been reached, the actual position value is set to the value specified in the"Position" parameter, and the axis is slowed down. The axis is now synchronized ("Sync" = TRUE), but is not positioned on the reference point.

If the maximum travel distance (Page 10805) is exceeded in the reference point approach, instruction MC_Home slows down the axis to a standstill (refer to Sequence of travel movements (Page 10805)) and the "Error" and "Err.DistanceErr" displays is set (refer to Errors with soft stop (Page 10822)).

The axis will not be synchronized ("Sync" = FALSE) if the reference point approach is stopped before the reference point has been reached.

instruction MC_Home does not stop at the end of the axis, or automatically reverse the direction of movement.

DANGER

If no reference point is found, you must take suitable measures to ensure that the axis will come to a standstill before it reaches the mechanical end position (for example, EMERGENCY STOP switch, safety limit switch).

Set reference point ("Velocity" = 0)

Set "Velocity" to 0. Once "Execute" has been set, the actual position value is set to the value specified in "Position" and the status bit in the axis DB is set simultaneously to "Sync" = TRUE.

Following a replacement of the absolute encoder or mechanical alterations, you must repeat the set reference point task, even if the axis is still synchronized.

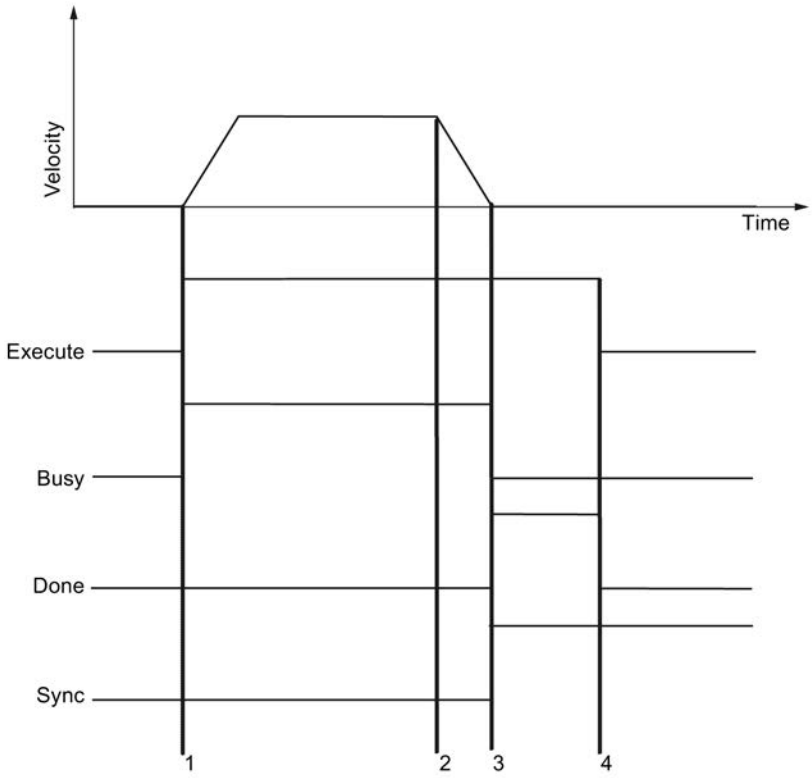
Note

Rule for absolute encoders:

The encoder value at the reference point is saved in the axis DB. For this reason, after setting the reference point, you should add the online axis DB to your offline database. Otherwise, the axis is no longer synchronized after you download the DB to the CPU.

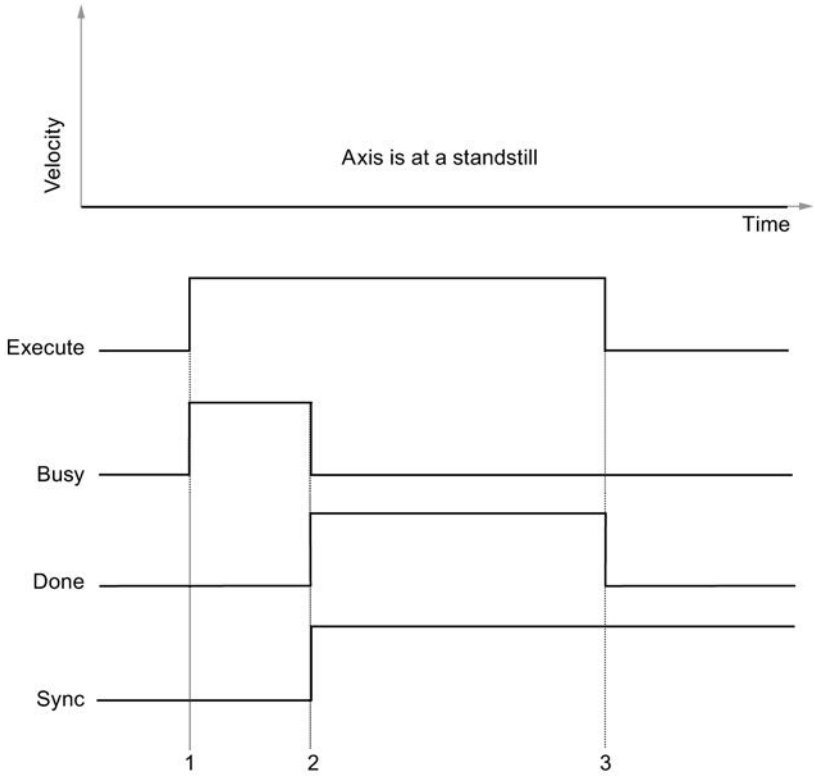
Read the information on Loading the axis DB (Page 10809).

Signal flow diagram for reference point approach



1 Start of the reference point approach with "Execute".
2 Reference position found; axis is decelerating.
3 Axis has reached a standstill; "Busy" is cleared and "Done" is reset.
The axis is synchronized.
4 Reset of "Execute" resets "Done".

Signal flow diagram for reference point setting



1 Start of the reference point setting or absolute encoder adjustment with "Execute".

2 The axis reports the synchronized state.

3 Reset of "Execute" resets "Done". The process is terminated.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

The reference point approach or reference setting can only be started when the axis is at a standstill.

Parameter

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	Start of reference point approach or reference setting with a positive edge.
Position	INPUT	REAL	**Incremental encoder** Reference point coordinate for reference point approach or reference setting. **Absolute encoder** Reference point coordinate for reference setting. The value must be within the working range: • **Linear axis** Within the software limit switch • **Rotary axis:** Rotary axis start ≤ position < Rotary axis end
Velocity	INPUT	REAL	Axis velocity in [length unit/s] Velocity > 0: Reference point approach Velocity = 0: Set reference point
Acceleration	INPUT	REAL	Axis acceleration in [length unit/s^2]
Deceleration	INPUT	REAL	Axis deceleration in [length unit/s^2]
Direction	INPUT	INT	Start direction for reference point approach 1 = Reference point approach in positive direction -1 = Reference point approach in negative direction
Busy	OUTPUT	BOOL	1 = Job in progress.
Done	OUTPUT	BOOL	1 = Job completed without error. The bit remains active at least for one call of the instruction until "Execute" is reset.
CommandAborted	OUTPUT	BOOL	1 = Job was canceled by another job. The bit remains active at least for one call of the instruction until "Execute" is reset.
Error	OUTPUT	BOOL	1 = Job was canceled due to an axis error. The bit remains active at least for one call of the instruction until "Execute" is reset.

MC_StopMotion

Description

instruction MC_StopMotion allows you to interrupt any travel instruction so that the axis comes to a standstill with the set deceleration.

Typical application: A positioning job can by interrupted at any time due to an external event (for example, opening of a safety door).

Operating principle

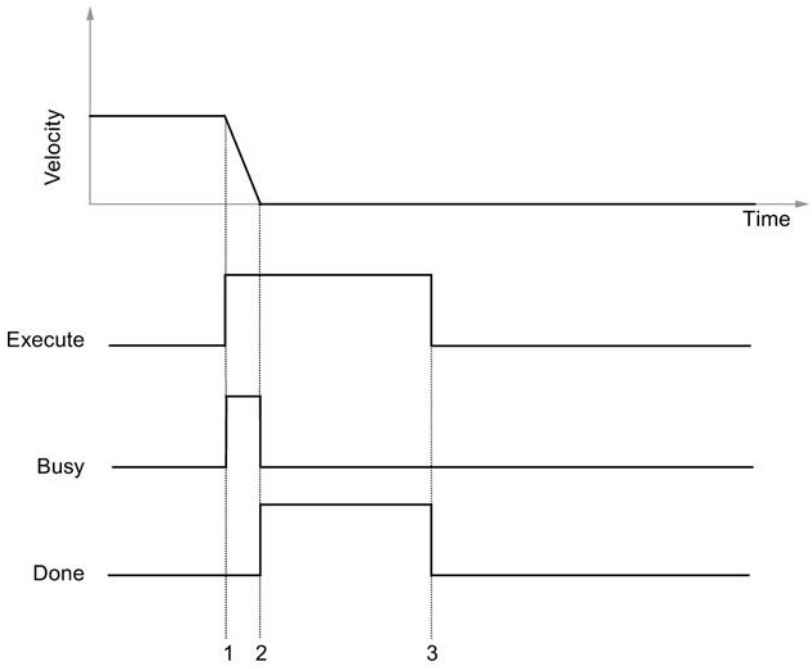
On detecting the positive edge at input "Execute", the MC_StopMotion instruction takes control of the axis and slows it down to a standstill. The MC_StopMotion instruction can only be

interrupted by hard faults (Page 10825) and not by a travel instruction. After the axis has reached a standstill, a new travel movement can be started.

You can set a deceleration value at the MC_StopMotion instruction to suit your application. If this deceleration value is less than the one set in the interrupted instruction, the axis is slowed down with this deceleration value. This ensures that the axis does not overshoot the target set in the interrupted block.

If you interrupt a movement with target (MC_MoveAbsolute, MC_MoveRelative) by calling instruction MC_StopMotion, the instruction continues to calculate the remaining distance. On termination of instruction MC_StopMotion, the remaining distance indicates the distance of the current position setpoint to the target of the interrupted block. You can assign this value to an MC_MoveRelative instruction in order to approach the axis to the original target after interruption.

Signal flow diagram for MC_StopMotion



1 FB Stop is started while a movement is active.

2 MC_StopMotion sets output "Done" as soon as standstill is reached.

3 Reset of "Execute" terminates FB Stop.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	Start of braking with a positive edge
Deceleration	INPUT	REAL	Axis deceleration for deceleration ramp in [length unit/s^2] If you enter values <= 0.0, the maximum deceleration value from the axis DB will be used.
Busy	OUTPUT	BOOL	1 = Job in progress.
Done	OUTPUT	BOOL	1 = Job completed without error. The bit remains active at least for one call of the instruction until "Execute" is reset.
CommandAborted	OUTPUT	BOOL	1 = Job was cancelled due to manual intervention at the controller.
Error	OUTPUT	BOOL	1 = Job was canceled due to an axis error. The bit remains active at least for one call of the instruction until "Execute" is reset.

MC_Control

Description

instruction MC_Control allows you to implement the position control in combination with the I/O drivers.

instruction MC_Control may assume the following controller states:

- Control mode
- Follow-up mode
- Manual mode

Control mode

The MC_Control instruction

- Calculates the velocity setpoint for the output driver as a function of the position setpoint of a travel instruction and actual position value of the input driver.
- Monitors
  - Standstill
  - Target approach
  - Following error
- Processes error acknowledgment (see Error displays and error acknowledgment (Page 10820))

Follow-up mode

In this state, the setpoint position is corrected according to the actual position. The follow-up mode is activated if "EnableDrive" = FALSE, or after an axis fault with hard stop (refer to Error

displays and error acknowledgment (Page 10820)). Output "DriveEnabled" = FALSE in follow-up mode.

Manual mode

In this mode, the manual setpoint "ManVelocity (Page 10786)" is output as the velocity setpoint. This setpoint is limited to "MaxVelocity (Page 10791)". The position setpoint is tracked to the actual position value.

This state is switched on or off using "ManEnable" (Page 10786).

WARNING
Manual control is possible at any time, even if the axis is in error status.

Operating principle

The proportional element of the control algorithm of instruction MC_Control is configured at the "controller gain (Page 10786)" parameter ("FactorP").

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Call instruction MC_Control once per axis unconditionally in the selected OB.

Parameter

Parameter	Declaration	Data type	Meaning
EnableDrive	INPUT	BOOL	0 = Disable drive, tracking control is active. 1 = Enable drive, position control is active. If no error is pending, "DriveEnabled" is set.
DriveEnabled	OUTPUT	BOOL	Drive enable, position control is active. If "EnableDrive" is set and no error is pending, "DriveEnabled" is set. If "DriveEnabled" is not set, the axis is in follow-up mode and standstill monitoring is not active for the axis. This signal should be used in order to control the enable input on the power unit.

MC_Simulation

Description

instruction MC_Simulation allows you to test your traversing program and driver blocks. You do not require any I/O and your axis is not moved.

Operating principle

instruction MC_Simulation feeds back the controller output value to the input driver by simulating an encoder value corresponding to the encoder type selected in the axis DB.

To activate simulation mode, set the bit "Sim" = TRUE in the axis DB.

When simulation is enabled, a reference point approach is not possible. When in simulation mode, configure the "Velocity" input of instruction MC_Home with the value 0.0. This means that a reference setting will be performed instead of a reference point approach.

If you set a reference point in simulation mode, the reference point of your real axis is also altered. It is therefore necessary to resynchronize your axis after exiting simulation mode.

Whenever you activate and deactivate simulation mode, the axis is set to the stop status that requires acknowledgment (refer to Faults with hard stop (Page 10825)) and all bits of the "Init.Ix" bit array is set in the axis DB.

If simulation mode is enabled, the following occurs:

- The output driver does not output values to the I/O.
- The input driver does not read in any encoder values from the I/O.

DANGER

The axis must be at a standstill when simulation mode is enabled.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

MC_GearIn

Description

instruction MC_GearIn allows you to link the velocity setpoint of a following axis to the velocity setpoint of a leading axis by means of gear ratio (setpoint coupling).

Operating principle

As soon as a positive edge is detected at the "Execute" input, instruction MC_GearIn couples a following axis (parameter "Slave") to the velocity or position setpoint of a leading axis (parameter "Master") by means of the set gear ratio. The instruction therefore represents a travel instruction of the following axis.

The values in each cycle of a coupled motion are as follows:

- The distance traversed by the following axis, starting at the coupling position =
  the distance traversed by the leading axis, starting at the coupling position x gear ratio
- Velocity of the following axis = gear velocity =
  velocity of the leading axis x gear ratio
- Acceleration of the following axis =

  acceleration of the leading axis x gear ratio
- Deceleration of the following axis =
  deceleration of the leading axis x gear ratio

To allow the following axis to follow the leading axis without restriction, set the following values (with a reserve factor of 1.1) for the following axis. These values must be valid and attainable:

- Velocity(following axis) =
  Velocity(leading axis) × ABS(gear ratio) × 1.1
- Acceleration(following axis) =
  Acceleration(leading axis) × ABS(gear ratio) × 1.1
- Deceleration(following axis) =
  Deceleration(leading axis) × ABS(gear ratio) × 1.1

To cancel the coupling, call the stop instruction, or any other travel instruction for the following axis.

As with all other travel instructions, the MC_GearIn instruction can override a movement and be overridden itself.

The following axis can be coupled to a moving leading axis, whereby the following axis automatically accelerates or decelerates using the ramps configured at the gear instruction until it has reached the gear velocity.

The "InGear" output is set when the gear velocity has been reached for the first time after the start of the motion. If "Execute" is not set at this point, "InGear" remains active for one call of the instruction. "InGear" is reset at the high to low transition at Execute, when an error has occurred, and when a movement is overridden.

Provided the gear velocity can be reached, the following axis travels in speed and position synchronism. The "Coupled" output of the instruction is set.

If the following axis cannot reach the gear velocity, it moves according to the motion parameters set at the gear instruction. The "Coupled" output of the instruction is then reset. As soon as the gear velocity can be reached again, the following axis moves at this velocity. The "Coupled" output of the instruction is set.

As long as instruction MC_GearIn is active at the following axis, you can call any travel instruction for the leading axis, and also override a movement.
Exception: During referencing of the leading axis, the coupling is released with an error ("Err.MasterErr").

The "Override" axis parameter is useless for the following axis and is therefore not evaluated.

Special features when operating a following axis as linear axis:

- If the maximum traversing distance is exceeded during a gear movement, instruction MC_GearIn slows down the following axis to a standstill with the error message (see Sequence of travel movements (Page 10805)). The coupling is released.
- A non-synchronized linear axis is not monitored and can therefore travel right to the physical end of the axis without slowing down. You must therefore complete the motion in time.
- A synchronized linear axis is decelerated when approaching the end of the working range and stops at the software limit switch. The error message "DistanceErr" is output along with "Err.SWLimitMinExceeded", or "Err.SWLimitMaxExceeded". The coupling is released.

WARNING

Injury to persons and property may occur.

instruction MC_GearIn can be used to traverse a non-synchronized linear axis to its mechanical end.

To avoid injury to persons and objects, take the following precautions:

- Install an EMERGENCY STOP switch in the vicinity of the computer. This is the only way to ensure that the system is reliably switched off in the event of a computer or software failure.
- Install safety limit switches for direct control of the power units of all drives.
- Ensure that nobody has access to the area of the system where there are moving parts.

Special feature when operating a following axis as rotary axis:

- The traversing distance of instruction MC_GearIn is not limited (infinite rotation).

Error handling

Master axis errors:

- The leading axis decelerates when it detects errors that result in a soft stop. The coupled following axis follows this motion and does not release the coupling. If the leading axis error is acknowledged, and the leading axis continues its motion, the following axis also resumes the motion.
- If the leading axis detects errors that result in a hard stop, or if it is operated in manual mode, the gear instruction decelerates the following axis and resolves the coupling. The instruction outputs the "Err.MasterErr" error message.

Slave axis errors:

- If the following axis detects errors that result in a soft stop, the gear instruction decelerates the following axis and resolves the coupling.
- If the following axis detects errors that result in a hard stop, the following axis is brought to a stop, and the coupling is released.
- The maximum traversing distance of the coupling movement of a linear axis is limited to 2^{24} increments or 2^{24} [length unit]. If this limit is exceeded, instruction MC_GearIn decelerates the following axis, outputs a "DistanceErr" error message that you can acknowledge, and resolves the coupling.

Coupling errors:

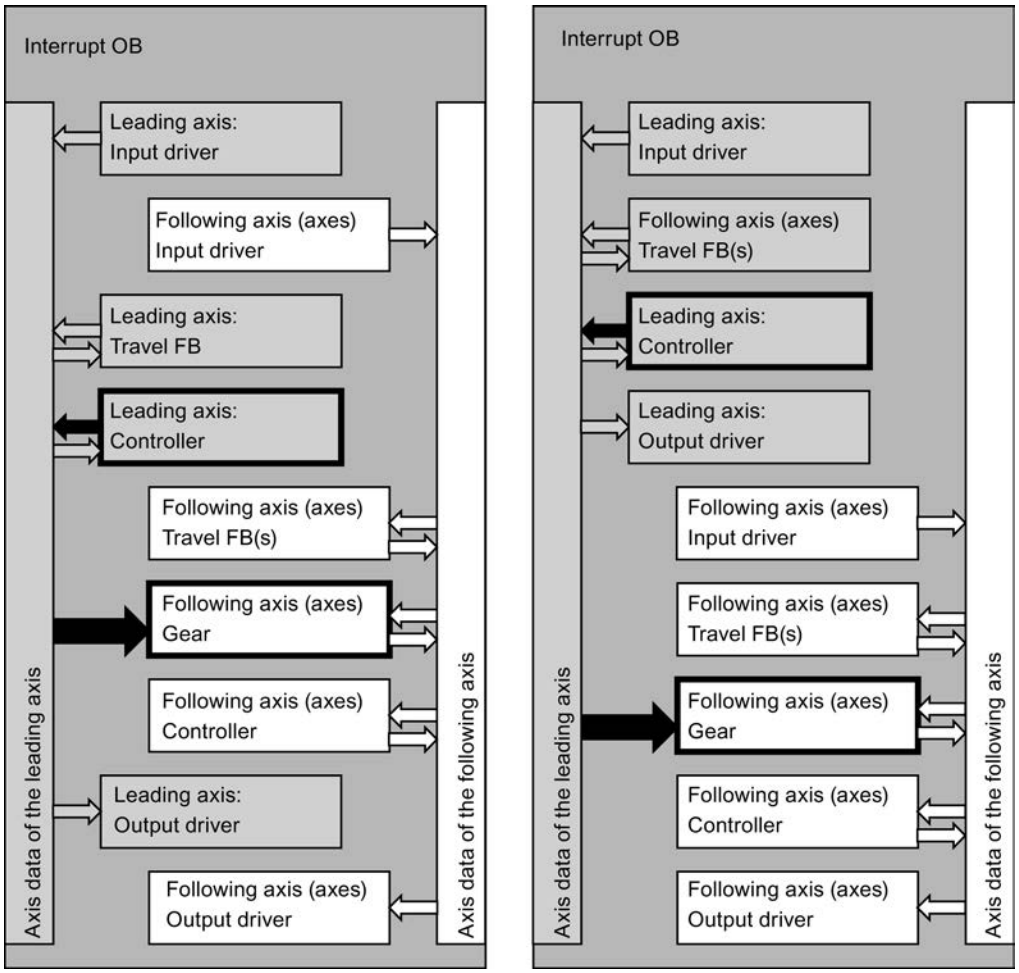
- The following axis travels according to the motion parameters set at the instruction if it cannot reach the gear velocity in the current cycle. This applies particularly when the axis is coupled to a moving leading axis and during a braking operation triggered by an error. The "Coupled" parameter is reset, but the coupling is not released. As soon as the following axis is capable of reaching the necessary values again, it travels according to these values and "Coupled" is set again.

Interaction of instructions for the gear

When using a gear instruction, you should also note the following aspects (refer to "Structure of a user program (Page 10799)"):

- The leading and following axes must be called in the **same** execution level.
- In each processing cycle, the following axis adapts itself again to the default values of the leading axis. The following axis instructions should therefore be called after instructions of the leading axis. In particular, the gear instruction of the leading axis should be called **after** the controller of the leading axis.
  Other call sequences, in particular nested calls of the instructions of the leading and following axes, may lead to greater following errors.
- In isochronous applications, the I/O data of the leading axis **and** the following axis must be located in the same process image partition.

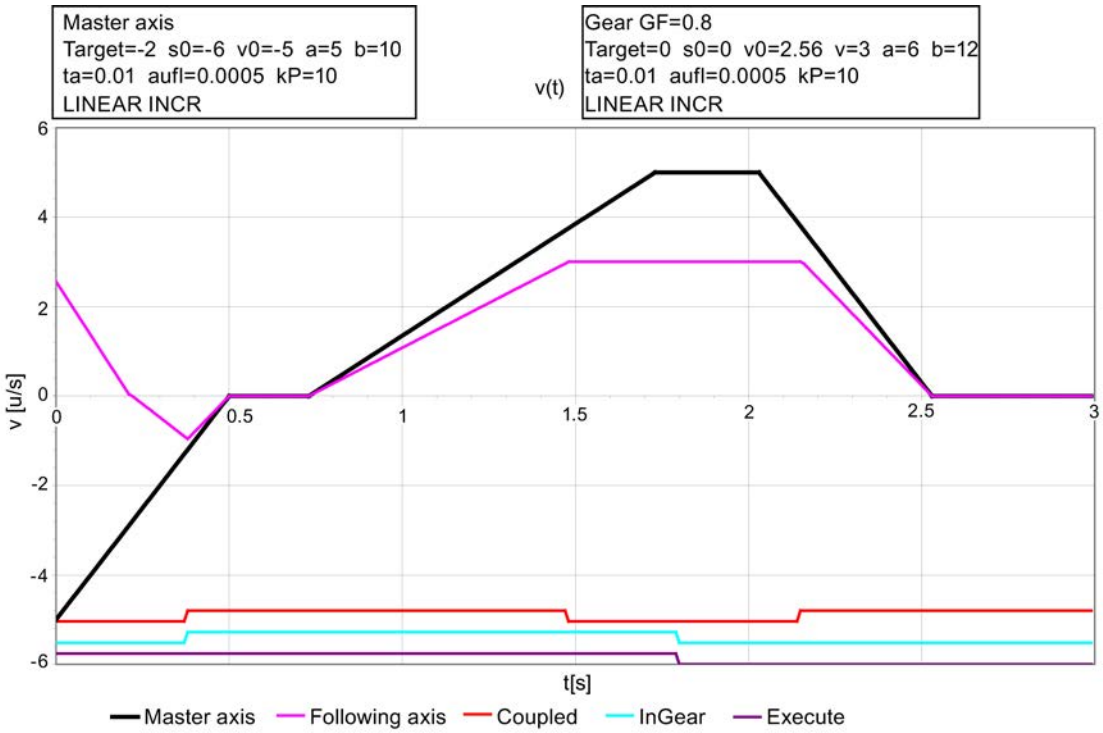
This results in one of the following figures, depending on the selected structure:



Signal Flow Chart

The chart below shows how the gear instruction takes control of the axis movement and couples to a moving leading axis. In this example, the gear ratio is 0.8. The following axis cannot reach the velocity required to maintain this ratio.

The leading axis is stopped between t = 0.5 and t = 0.72. The coupling is maintained.



Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	Start of the gear motion on a positive edge
Rationumerator	INPUT	REAL	Gear ratio numerator Gear ratio = "RatioNumerator" / "RatioDenominator" If "RatioNumerator" < 0, the direction of movement of the leading axis is opposite to that of the following axis. "RatioNumerator" = 0 is rejected with "DataErr" = 1. You can modify the effective gear ratio of the following axis by overriding the active gear instruction with another instance that contains a modified gear ratio.

Parameter	Declaration	Data type	Description
RatioDenominator	INPUT	INT	The gear ratio denominator; must be an integer ≥ 1 You can modify the effective gear ratio of the following axis by overriding the active gear instruction with another instance that contains a modified gear ratio.
Velocity	INPUT	REAL	Axis velocity in [length unit/s] The maximum geared velocity of the following axis is limited to this value.
Acceleration	INPUT	REAL	Axis acceleration in [length unit/s^2] The maximum geared acceleration of the following axis is limited to this value. This parameter is used if: • The following axis is coupled to a faster leading axis. • The acceleration of the leading axis exceeds the capability of the following axis.
Deceleration	INPUT	REAL	Axis deceleration in [length unit/s^2] The maximum geared deceleration of the following axis is limited to this value. This parameter is used if: • A following axis that is already in motion is coupled to a slower leading axis. • The following axis is brought to a stop due to an error. • The deceleration of the leading axis exceeds the capability of the following axis.
Master	IN-OUT	AXIS_REF	Reference to the leading axis data
Slave	IN-OUT	AXIS_REF	Reference to the following axis data
Init	IN-OUT	BOOL	If Init = TRUE, the instruction completes its initialization and then resets the bit.
Busy	OUTPUT	BOOL	1 = Job in progress.
InGear	OUTPUT	BOOL	1 = Gear speed reached Set the first time the gear velocity has been reached. If "Execute" is already reset at this point, "InGear" still remains active for one call of the instruction. It is reset at the high to low transition at Execute, when an error has occurred, and when the movement is overridden.
Coupled	OUTPUT	BOOL	1 = Axis is currently coupled to the leading axis. Reports in each processing cycle whether the gear speed was reached.
CommandAborted	OUTPUT	BOOL	1 = Job was canceled by another job.
Error	OUTPUT	BOOL	1 = Job was canceled due to an axis error. The bit remains active at least for one call of the instruction until "Execute" is reset.

Input driver

Task of the input driver

Function

This input driver is used to detect the actual positions of an axis. It reads the non-standardized axis position and converts this to the selected unit of length.

Suitable input drivers are available for a number of position decoder modules. These drivers are described individually in the following sections. Adaptation to configurations that are not directly supported is possible by means of instruction EncoderUniversal (Page 7080).

Initialization

- Synchronization (Page 10771) of an **incremental encoder** with the axis is reset (parameter "Sync" = FALSE in the axis data). A new reference point approach or reference setting is then necessary.
- Synchronization of an **absolute encoder** with the axis is retained.

Call

The input driver of an axis must be called unconditionally.

Isochronous mode

The input drivers support the isochronous mode of all suitable I/O modules. The driver automatically recognizes isochronous mode by the fact that it is called in an isochronous OB.

Error handling

Errors can be detected by means of the following mechanisms:

- An error detected by the user program (for example, module removed, or station failure) can be signaled to the input driver at input "EncErr". The input driver will not access the position decoder module as long as input "EncErr" is TRUE.
- Some position decoder modules provide error information that is evaluated automatically by the input driver.
- Other position decoder modules only return their error information by means of diagnostic interrupt and data set 1 (DS1). You must therefore perform the following steps:
  - Enable the diagnostics interrupt for the module in the hardware configuration.
  - In OB 82, compare the address returned in the local data element "OB82_MDL_ADDR" with your module address. If these agree, set the "ReadDiagErr" parameter of the input driver to TRUE (for the incoming and outgoing alarm).
    If "ReadDiagErr" = TRUE, the input driver reads DS 1 of the module by calling instruction RD_REC, saves this record to the static "DiagStatus" parameter
    and then sets "ReadDiagErr" = FALSE. The return value of instruction RD_REC is written to the static "JobErr" parameter.

Error Response

- In the event of an error, the"Error" and "Err.EncoderErr" displays is set in the axis DB.
- With **incremental encoder**, status bit "Sync" is always reset in the axis DB.
- With **absolute encoder**, status bit "Sync" is never reset in the axis DB.

Please also note specific error handling in the following sections on individual drivers.

For detailed information on error response, error display, and error acknowledgment, see "Error Displays and Error Acknowledgment in the axis DB (Page 10820)".

Simulation mode

The input driver will not access the position decoder module as long as input "Sim (Page 10778)" is set in the axis DB. The encoder values are simulated by means of instruction MC_Simulation (Page 7050).

EncoderIM178

Description

instruction EncoderIM178 is included for reasons of compatibility with STEP7 V5.x. Siemens is no longer actively marketing the IM 178-4 . You can use IM 174 as an alternative.

IM 178-4 must be used with the same channel number and axis both at the input **and** output drivers, as specific address settings refer to both drivers. A combination with other drivers is not possible.

Configuration of IM 178-4 (hardware configuration)

Select the IM 178-4 from the "Distributed I/O devices" section in the hardware catalog. Configure slot 4 of IM 178-4 as "4Word QQ/12Word QI/cons. 1Word".

Parameter	Setting
Synchronization	Software
Edge selection	Rising, falling, or both edges

Configure the other settings depending on the connected encoder.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
EncErr	INPUT	BOOL	1 = Encoder error Here, you inform the input driver of an error detected by the user program (for example, module removed or station failure).
RefMode	INPUT	BOOL	Referencing type: 0 = With digital input I0 1 = With digital input I1 and zero mark
DI_0	OUTPUT	BOOL	Digital input 0 of the corresponding channel of IM 178-4
DI_1	OUTPUT	BOOL	Digital input 1 of the corresponding channel of IM 178-4
DI_2	OUTPUT	BOOL	Digital input 2 of the corresponding channel of IM 178-4
The following static parameters of the instruction are only used for diagnostic purposes in the event of an error.			
Status.PARA	STAT	BOOL	PARA = TRUE: IM 178-4 was correctly configured by means of the hardware configuration. PARA = FALSE: see SIMATIC IM 178-4 Drive Interface Manual (http://support.automation.siemens.com/WW/view/en/1117389)
Status.EXTF0	STAT	BOOL	1 = Encoder signal line fault
Status.EXTF1	STAT	BOOL	1 = Encoder error
Status.EXTF2	STAT	BOOL	1 = Zero mark fault

Synchronization

Synchronization can be performed by:

- Reference point approach (with incremental encoder)
- Reference point setting (with incremental and absolute encoder)

Depending on the setting, connect the reference point switch to digital input I0 or I1.

During a reference point approach, IM 178-4 can detect the reference point in either of the following cases:

- At a negative edge, positive edge, or at both edges at digital input I0 (RefMode = 0).
- At the positive edge of the first zero mark after detecting the negative edge of digital input I1 (RefMode = 1).

If the axis is already positioned on the reference point switch at the time the reference point approach is started, it is referenced to the current position, as is the case with reference point setting.

Digital Inputs

The digital inputs I0 to I2 of the selected channel of IM 178-4 are indicated at the output parameters "DI_0" to "DI_2" of the input driver.

Error handling

In response to a detected error, the input driver sets the "Error" and "Err.EncoderErr" error codes in the axis DB. The input driver resets the "Sync" status bit in the axis DB if you are using an incremental encoder. For more information, refer to the SIMATIC IM 178-4 Drive Interface Manual (http://support.automation.siemens.com/WW/view/en/1117389).

For detailed information on error response, error display, and error acknowledgment, see "Error Displays and Error Acknowledgment in the axis DB (Page 10820)".

Error acknowledgment

If you set error acknowledgment in the axis DB of Easy Motion Control, errors pending in the module are also acknowledged.

EncoderFM450

Description

instruction EncoderFM450 is used to employ a channel of the FM 450-1 as position decoder module for incremental encoders. The channel number is configured at the axis DB.

Configuration of the module (hardware configuration)

The FM 450-1 module should be configured in accordance with the settings listed in the following table.

Dialog	Setting
Interrupt selection	Enable diagnostic interrupt
Operating modes	• Count range: -31 to +31 bits • Operating mode: Continuous counting • Gate control: None
Inputs	If you want to perform a reference point approach, you must select the following setting: • Synchronization: once • The setting "Evaluate zero mark for setting" may be selected or cleared as a reference point criterion.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
EncErr	INPUT	BOOL	1 = Encoder error Here, you inform the input driver of an error detected by the user program (for example, module removed or station failure).
ReadDiagErr	IN-OUT	BOOL	Set this bit in OB 82 if the module has triggered a diagnostic interrupt (see below under Error handling).
DI_0	OUTPUT	BOOL	Digital input 0 of the selected channel
DI_1	OUTPUT	BOOL	Digital input 1 of the selected channel
DI_2	OUTPUT	BOOL	Digital input 2 of the selected channel
The following static parameters of the instruction are only used for diagnostic purposes in the event of an error.			
JobErr	STAT	INT	Return value of instruction RD_REC, if "ReadDiagErr" = TRUE was set.
DiagStatus.EXT_VOLTAGE	STAT	STRUCT	External auxiliary voltage faulty
DiagStatus.CONFIG_ERR	STAT	STRUCT	Incorrect configuration
DiagStatus.WTCH_DOG_FLT	STAT	STRUCT	Watchdog timeout
DiagStatus.RAM_FLT	STAT	STRUCT	RAM failure
DiagStatus.CH1_SIGA	STAT	STRUCT	Channel 1 signal A defective
DiagStatus.CH1_SIGB	STAT	STRUCT	Channel 1 signal B defective
DiagStatus.CH1_SIGZ	STAT	STRUCT	Channel 1 signal N defective
DiagStatus.CH1_BETW	STAT	STRUCT	Channel 1 fault between channels

Parameter	Declaration	Data type	Description
DiagStatus.CH1_5V2	STAT	STRUCT	Channel 1 encoder supply 5.2 V fault
DiagStatus.CH2_SIGA	STAT	STRUCT	Channel 2 signal A defective
DiagStatus.CH2_SIGB	STAT	STRUCT	Channel 2 signal B defective
DiagStatus.CH2_SIGZ	STAT	STRUCT	Channel 2 signal N defective
DiagStatus.CH2_BETW	STAT	STRUCT	Channel 2 fault between channels
DiagStatus.CH2_5V2	STAT	STRUCT	Channel 2 encoder supply 5.2 V fault

Synchronization

Synchronization can be performed by:

- Reference point approach or
- Set reference point

Connect the reference point switch to digital input I2.

During a reference point approach, the module can detect the reference point in either of the following cases:

- On a rising edge at digital input I2
- On the rising edge of the zero mark, if digital input I2 is set

If the axis is already positioned on the reference point switch at the time the reference point approach is started, it is referenced to the current position, as is the case with reference point setting.

Digital inputs

Digital inputs I0 to I2 of the selected channel are indicated at the output parameters "DI_0" to "DI_2" of the input driver.

Error handling

The module returns its error information by means of dataset 1 (DS1). In the following situations, the input driver reads DS1 from the module by calling instruction RD_REC and saves it to the static "DiagStatus" parameter:

- The input driver is initialized.
- The module reports an error.
- The driver input parameter "ReadDiagErr" is TRUE.

It then sets "ReadDiagErr" = FALSE. The return value of instruction RD_REC is written to the static "JobErr" parameter (see JOB_Err alarms (Page 10828)).

If an error display is registered in dataset 1, the input driver sets the error codes "Error" and "Err.EncoderErr" to TRUE in the axis DB. Status bit "Sync" is set to FALSE.

For detailed information on error response, error display, and error acknowledgment, see Error Displays and Error Acknowledgment in the axis DB (Page 10820).

Note

If you are using an FM 450-1 in single-channel mode only, you must configure the second channel with "24 V channel A+B phase offset", as it will otherwise transmit cyclic diagnostic interrupts (for example, due to missing circuitry).

EncoderET200S1SSI

Description

instruction EncoderET200S1SSI is used to employ the ET 200S 1SSI Fast Mode as position decoder module for absolute encoders.

The driver can also be used for the isochronous variant of this module.

Configuration the ET 200S 1SSI Fast Mode module (hardware configuration)

The input driver is only suitable for the 1SSI module with "**Fast Mode**" parameter setting as operating mode. In the TIA Portal, the "**Fast Mode**" is no longer selected based on the module, but is now set as operating mode.

Select the module from the "Distributed I/O devices" > ET 200S > FM > Position detection" section in the hardware catalog.

Parameter	Setting
Group diagnostics	You can activate this function if you want to evaluate the data in your program.
Detection	Freewheeling
Reversal of the direction of rotation	Deactivated You can reverse the direction of rotation using the instructions of Easy Motion Control (parameter "Set encoder polarity (Page 10785)").
Operating mode	Fast mode

Configure the other settings depending on the connected encoder.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
EncErr	INPUT	BOOL	1 = Encoder error Here, you inform the input driver of an error detected by the user program (for example, module removed or station failure).
The following static parameters of the instruction are only used for diagnostic purposes in the event of an error.			
Status.STS_UP	STAT	BOOL	Forward direction: encoder values increase
Status.STS_DN	STAT	BOOL	Reverse direction: encoder values decrease
Status.STS_DI	STAT	BOOL	Status of digital input
Status.EXTF	STAT	BOOL	Group error or short circuit in encoder supply
Status.ERR_PARA	STAT	BOOL	Parameter errors
Status.RDY	STAT	BOOL	Module is ready for operation

Synchronization

The ET 200S 1SSI Fast Mode module only supports synchronization by means of reference point setting.

Error handling

If the input driver detects an error report of the module, it sets the error codes "Error" and "Err.EncoderErr" in the axis DB.

For detailed information on error response, error display, and error acknowledgment, see Error Displays and Error Acknowledgment in the axis DB (Page 10820).

EncoderAbsSensorDP

Description

instruction EncoderAbsSensorDP is used to employ the absolute encoder SIMODRIVE sensor in combination with PROFIBUS DP for position detection.

Configuration of the absolute encoder SIMODRIVE sensor (hardware configuration)

Select the "PROFIBUS DP > ENCODER > SIMODRIVE sensor > Version 2.2 Single-turn or Version 2.2 Multiturn" encoder from the hardware catalog. If the SIMODRIVE sensor is not yet available, implement the device using the corresponding GSD file. The encoder will then be

available in the "Additional field devices > PROFIBUS DP > Encoder > Siemens AG > SIMODRIVE sensor" folder.

Parameter	Setting	
Code sequence	Increasing clockwise You can reverse the direction of rotation using the instructions of Easy Motion Control (parameter "Set encoder polarity (Page 10785)").	
	Multiturn	**Single-turn**
Scaling function	Enable	Enable
Measuring steps (high)	0	-
Measuring steps (low)	4096	4096
Physical impulses (high)	0	-
Physical impulses (low)	4096	4096
Desired measuring units per	Revolution	Revolution
Total measuring range (high)	256	-
Total measuring range (low)	0	4096
Commissioning mode	Switched off	Switched off
Shorter diagnostics	No	No
Lower limit switch	Switched off	Switched off
Lower limit switch (high)	0	-
Lower limit switch (low)	0	0
Upper limit switch	Switched off	Switched off
Upper limit switch (high)	0	-
Upper limit switch (low)	32767	4096
Velocity output unit	Steps/1000 ms	Steps/1000 ms

Adapt the setting to your selected length unit using the instructions of Easy Motion Control. Fixed parameters can therefore be assigned to the encoder - as indicated above. It is not necessary to perform a "teach-in" of the encoder (refer to the SIMODRIVE sensor absolute encoder with PROFIBUS DP Operating Manual (http://support.automation.siemens.com/WW/view/en/28813901)).

At the parameter "steps per encoder revolution" of Easy Motion Control, enter 4096 ("StepsPerRev (Page 10782)").

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
EncErr	INPUT	BOOL	1 = Encoder error Here, you inform the input driver of an error detected by the user program (for example, module removed or station failure).
The following static parameters of the instruction are only used for diagnostic purposes in the event of an error.			
Status.Ready	STAT	BOOL	1 = Encoder is ready for operation
Status.Mode	STAT	BOOL	Operating mode: 0 = Startup mode, 1 = Normal mode
Status.SWLimitExceeded	STAT	BOOL	1 = Software limit switch MIN or MAX has responded
Status.Dir	STAT	BOOL	Direction of rotation: 0 = Clockwise, 1 = Counter-clockwise

Synchronization

Synchronization with the absolute encoder SIMODRIVE sensor is only possible by means of reference point setting.

Error handling

If the input driver detects that an encoder error alarm is pending, it sets the"Error" and "Err.EncoderErr" error codes in the axis DB. For detailed information on error response, error display, and error acknowledgment, see Error Displays and Error Acknowledgment in the axis DB (Page 10820).

EncoderSM338

Description

instruction EncoderSM338 is used to employ SM 338 POS_INPUT as position decoder module for absolute encoders.

The driver can also be used for the isochronous variant of this module.

Configuration of SM 338 (hardware configuration)

The FREEZE function must be deactivated.

If the input driver detects a FREEZE status at the encoder input, it sets the "Error" and "Err.EncoderErr" error codes in the axis DB.

Configure the other settings depending on the connected encoder.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
EncErr	INPUT	BOOL	1 = Encoder error Here, you inform the input driver of an error detected by the user program (for example, module removed or station failure).
ReadDiagErr	IN-OUT	BOOL	Set this bit in OB 82 if the module has triggered a diagnostic interrupt (see Error handling).
The following static parameters of the instruction are only used for diagnostic purposes in the event of an error.			
Freeze	STAT	BOOL	FREEZE status of the encoder input
JobErr	STAT	INT	Return value of instruction RD_REC, if "ReadDiagErr" = TRUE was set.
DiagStatus.EXT_VOLTAGE	STAT	STRUCT	External auxiliary voltage faulty
DiagStatus.CONFIG_ERR	STAT	STRUCT	Incorrect configuration
DiagStatus.WTCH_DOG_FLT	STAT	STRUCT	Watchdog timeout
DiagStatus.RAM_FLT	STAT	STRUCT	RAM failure
DiagStatus.CH0_INTF	STAT	STRUCT	Configuration or parameter assignment error (internal channel error)
DiagStatus.CH0_EXTF	STAT	STRUCT	Encoder error (external channel error)

Synchronization

Synchronization with SM 338 is only possible by means of reference point setting.

Error handling

This module only provides its error information via diagnostic interrupt and data record 1 (DS1). You must therefore perform the following steps:

- Enable the diagnostics interrupt for the module in the hardware configuration.
- In OB 82, compare the address returned in the local data element "OB82_MDL_ADDR" with your module address. If these agree, set the ReadDiagErr parameter of the input driver to TRUE (for the incoming and outgoing interrupt).
  If "ReadDiagErr" = TRUE, the input driver reads dataset 1 of the module by means of instruction RD_REC and saves it to the static "DiagStatus" parameter.
  It then sets "ReadDiagErr" = FALSE. The return value of instruction RD_REC is written to the static "JobErr" parameter (see JOB_Err alarms (Page 10828)).

If an error display is registered in dataset 1, the input driver sets the error codes "Error" and "Err.EncoderErr" to TRUE in the axis DB.

For detailed information on error response, error display, and error acknowledgment, see Error Displays and Error Acknowledgment in the axis DB (Page 10820).

EncoderET200S1Count

Description

instruction EncoderET200S1Count is used to employ the ET 200S 1Count5V or 1Count24V module as position decoder module for incremental encoders.

The driver can also be used for the isochronous variant of this module.

Configuration of the ET 200S 1Count module (hardware configuration)

The input driver is only suitable for the "1 Count 5V" or "1 Count 24V" from the hardware catalog. The count mode is now set as the operating mode instead of be selecting via the module.

Select the module from the "Distributed I/O devices" > ET 200S > FM > Counting" section in the hardware catalog.

Parameter	Setting
Group diagnostics	You can activate this function if you want to evaluate the data in your program.
Type of counting mode	Continuous counting
DI function, Synchronization	• If you wish to perform a reference point approach, select: – DI function = Synchronization at positive edge – Synchronization = One-time • Otherwise select: – DI function = input
Gate function	Arbitrary
Operating mode	Count

Configure the other settings depending on the connected encoder.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
EncErr	INPUT	BOOL	1 = Encoder error Here, you inform the input driver of an error detected by the user program (for example, module removed or station failure).
DOut_1	INPUT	BOOL	Digital output 1
DOut_2	INPUT	BOOL	Digital output 2 (only available with 1Count5V)
DIn	OUTPUT	BOOL	Digital input of module

Parameter	Declaration	Data type	Description
The following static parameters of the instruction are only used for diagnostic purposes in the event of an error.			
Status.ERR_ENCODER	STAT	BOOL	Short circuit / wire break at encoder signal (only with 1Count5V)
Status.ERR_DO2		BOOL	Short circuit / wire break / overheating DO2 (only with 1Count5V)
Status.ERR_PARA		BOOL	Parameter errors
Status.ERR_DO1		BOOL	Short circuit / wire break / overheating DO1
Status.ERR_24V		BOOL	Short circuit encoder supply

Synchronization

Synchronization can be performed by:

- Reference point approach or
- Reference setting

Connect the reference point switch to digital input DI.

During a reference point approach, ET 200S 1Count is capable of detecting the reference point based a rising edge at digital input DI.

If the axis is already positioned on the reference point switch at the time the reference point approach is started, it is referenced to the current position, as is the case with reference point setting.

Digital Input

The digital input of the module is displayed in output parameter "DIn" of the input driver.

Digital outputs

The digital outputs can be controlled by means of the "DOut_1" and "DOut_2" input parameters of the input driver.

Error handling

On detecting an error report of the module, the input driver sets the "Error" and "Err.EncoderErr" error codes and resets the "Sync" status bit in the axis DB.

For detailed information on error response, error display, and error acknowledgment, see Error Displays and Error Acknowledgment in the axis DB (Page 10820).

EncoderFM350

Description

instruction EncoderFM350 is used to employ the FM 350-1 as position decoder module for incremental encoders.

The driver can also be used for the isochronous variant of this module.

Configuration of the module (hardware configuration)

Dialog	Setting
Interrupt selection	Enable diagnostic interrupts
Operating modes	• Count range: -31 to +31 bits • Operating mode: Continuous counting • Gate control: None
Inputs	If you want to perform a reference point approach, you must select the following setting: • Set counter (DI set): once • The setting "Evaluate zero mark for setting" may be selected or cleared as a reference point criterion.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
EncErr	INPUT	BOOL	1 = Encoder error Here, you inform the input driver of an error detected by the user program (for example, module removed or station failure).
ReadDiagErr	IN-OUT	BOOL	Set this bit in OB 82 if the module has triggered a diagnostic interrupt (see Error handling).
DI_0	OUTPUT	BOOL	Digital input 0
DI_1	OUTPUT	BOOL	Digital input 1
DI_2	OUTPUT	BOOL	Digital input 2
The following static parameters of the instruction are only used for diagnostic purposes in the event of an error.			
JobErr	STAT	INT	Return value of instruction RD_REC, if "ReadDiagErr" = TRUE was set.
DiagStatus.EXT_VOLTAGE	STAT	STRUCT	External auxiliary voltage faulty
DiagStatus.NO_CONFIG	STAT	STRUCT	Configuration missing
DiagStatus.CONFIG_ERR	STAT	STRUCT	Incorrect configuration
DiagStatus.WTCH_DOG_FLT	STAT	STRUCT	Watchdog timeout
DiagStatus.RAM_FLT	STAT	STRUCT	RAM failure
DiagStatus.CH1_SIGA	STAT	STRUCT	Signal A faulty
DiagStatus.CH1_SIGB	STAT	STRUCT	Signal B faulty
DiagStatus.CH1_SIGZ	STAT	STRUCT	Signal N faulty

Parameter	Declaration	Data type	Description
DiagStatus.CH1_BETW	STAT	STRUCT	Error between channels
DiagStatus.CH1_5V2	STAT	STRUCT	Error at the 5.2 V encoder supply

Synchronization

Synchronization can be performed by:

- Reference point approach or
- Reference setting

Connect the reference point switch to digital input I2.

During a reference point approach, the module can detect the reference point in either of the following cases:

- Rising edge at digital input I2
- Rising edge of the zero mark, if digital input I2 is set

If the axis is already positioned on the reference point switch at the time the reference point approach is started, it is referenced to the current position, as is the case with reference point setting.

Digital inputs

The digital inputs I0 to I2 are displayed in the "DI_0" to "DI_2" output parameters of the input driver.

Error handling

The module returns its error information by means of dataset 1 (DS1). In the following situations, the input driver reads DS1 from the module by calling instruction RD_REC and saves it to the static "DiagStatus" parameter:

- The input driver is initialized.
- The module reports an error.
- The driver input parameter "ReadDiagErr" is TRUE.

It then sets "ReadDiagErr" = FALSE. The return value of instruction RD_REC is written to the static "JobErr" parameter (see JOB_Err alarms (Page 10828)).

If an error display is registered in dataset 1, the input driver sets the error codes "Error" and "Err.EncoderErr" to TRUE in the axis DB. Status bit "Sync" is set to FALSE.

For detailed information on error response, error display, and error acknowledgment, see Error Displays and Error Acknowledgment in the axis DB (Page 10820).

EncoderCPU314C

Description

The instruction EncoderCPU314C is used to integrate a counter channel of CPU 314C for position feedback from 24 V incremental encoders.

Up to 4 counter channels are available.

Configuration of the counter function of CPU 314C (hardware configuration)

This input driver is only supports CPU 314C with firmware V2.0 and higher.

For each channel you want to use, set the following parameters in the Properties dialog of the module in slot 2.4 (count):

Parameter	Setting
Mode	Continuous counting
HW gate	No HW gate (check box cleared)

Customize the other settings according to the encoder used and to the requirements for your application.

Enter the address you have assigned in the hardware configuration to the inputs at slot 2.4 (count) in the axis data of the "Start address of inputs of the position decoder module" parameter ("InputModuleInAddress"). Set the address of the counter outputs at the "Start address of outputs of the position decoder module" parameter ("InputModuleOutAddress") .

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
EncErr	INPUT	BOOL	1 = Encoder error Here, you inform the input driver of an error detected by the user program.

Synchronization

Synchronization can be performed by:

- Reference point approach or
- Reference setting

Connect the reference point switch to the latch digital input of the corresponding channel (DI1.4 for channel 0 to DI1.7 for channel 3).

During a reference point approach, CPU 314C is capable of detecting the reference point at a rising edge at digital input DI.

If the axis is already positioned on the reference point switch at the time the reference point approach is started, it is referenced to the current position, as is the case with reference point setting.

Among other things, the driver polls the status of the reference point switch during a reference point approach. The access actions that are necessary for reading the data considerably increase the instruction runtime.

Maximum count frequency

Note

The counter of CPU 314C does not output any values if its maximum counting frequency is exceeded (refer to the SIMATIC S7-300 CPU 31xC Technological Functions Manual (http://support.automation.siemens.com/WW/view/en/12429336)). The axis is halted and the error message "Following error exceeded (Page 10825)" is output.

When specifying the maximum velocity ("MaxVelocity") in the axis data, you should therefore account for the limit frequency of the counter.

Error handling

The CPU 314C does not return encoder error information.

EncoderIM174

Description

instruction EncoderIM174 is used to employ IM 174 as position decoder module for incremental and absolute encoders.

IM 174 provides up to 4 axes. You need a driver instruction EncoderIM174 for each axis.

Configuration of IM 174 (hardware configuration)

Insert an IM 174 at an isochronous PROFIBUS DP in your project. Ensure the I/O addresses are in the process image of the CPU. Adjust the addresses manually if necessary. Assign the

I/O addresses to a process image partition. An isochronous OB must be assigned to the process image partition in the configuration of the S7 CPU.

Parameter	Setting
I/O address	Specify the first input address for the axis to be used in Easy Motion Control in the axis DB. (Example for slot 4: input address 256 to 273; enter input address 256)
frame selection	You can select PROFIBUS frame 3 or frame 81 in the hardware configuration. Enter this information at input TelNumber of instruction EncoderIM174 as well.
611U compliance mode	When using an external reference mark, activate the 611U compliance mode.

Note

If frame 81 is used, IM 174 only transfers the encoder information to the CPU. Frame 81 is employed if you want to use IM 174 only to read encoder data.
In this case, it is not necessary to provide an output driver for IM 174 (OutputIM174).

Note

The hardware configuration implements IM 174 by default as an isochronous module. Isochronous processing is based on the condition that the I/O addresses of IM 174 are in the process image of the CPU.

For more information on the isochronous mode in SIMATIC S7, refer to the Isochronous Mode Function Manual (http://support.automation.siemens.com/WW/view/en/15218045).

The EncoderIM174 driver instruction demands isochronous processing. If you ignore this rule, the instruction is prevented from accessing the hardware addresses.

Configuration of IM 174 in the axis DB

Specify the input address set for this axis in the hardware configuration in the "Module address inputs" field.

The EncoderIM174 input driver calculates the axis number for IM 174 at runtime. This address can be read from the local data of the corresponding DB.

Input driver EncoderIM174 determines the cycle time that was set in the isochronous PROFIBUS DP at runtime and enters it in the axis DB. You are therefore strongly advised to read back the modified axis DB from the controller in order to apply all changes that may have been made.

Call

The instruction is called in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
RefMode	INPUT	INT	Referencing type: • 0 = Encoder zero mark • 1 = Positive edge at external zero mark (demands 611U compliant mode). • 4 = Positive edge at sensor 1 • 5 = Negative edge at sensor 1 • 6 = Positive edge at sensor 2 • 7 = Negative edge at sensor 2
TelNumber	INPUT	INT	Number of the PROFIBUS frame used: • 3 = PROFIBUS frame 3 • 81 = PROFIBUS frame 81
Err_ID	OUTPUT	WORD	Error code of the instruction: • W#16#0001 = Incorrect OB; instruction may only be installed in OB 6x • W#16#0002 = Incorrect frame number; IM 174 only supports PROFIBUS frames 3 and 81 • W#16#0004 = Incorrect axis number; IM 174 supports up to 4 axes. The hardware address you entered in the axis DB could not be assigned to an axis of IM 174. • W#16#0008 = Incorrect referencing mode setting. Check the parameter at input RefMode.

Synchronization

Synchronization can be performed by:

- Reference point approach (with incremental encoder)
- Reference point setting (with incremental and absolute encoder)

You can use two types of reference point switches for reference point approach:

- If operating with an encoder with zero mark that you want to use for referencing, set the parameter at input "RefMode" = 0. The 611U compliance mode may not be selected in this scenario.
- Configure input "RefMode" = 1 if you want to use an external reference mark (BERO) for referencing. Connect the external reference point switch to digital input B1 to B4, depending on the axis used in your application. For this setup, it is mandatory to select the 611U compliance mode in the hardware configuration.

The digital inputs M1 or M2 must be used for both adjustable sensors.

For more information, refer to the Distributed I/O PROFIBUS Module IM 174 manual (http://support.automation.siemens.com/WW/view/en/35014863).

If the axis is already positioned on the reference point switch at the time the reference point approach is started, it is referenced to the current position, as is the case with reference point setting.

Error handling

In response to a detected error, the input driver sets the "Error" and "Err.EncoderErr" error codes in the axis DB. The input driver resets the "Sync" status bit in the axis DB if you are using an incremental encoder.

For more information on error response, error display, and error acknowledgment, see Error displays and error acknowledgment in the axis DB (Page 10820).

Error acknowledgment

If you set error acknowledgment in the axis DB of Easy Motion Control, errors pending in the module are also acknowledged.

EncoderSINAMICS

Description

instruction EncoderSINAMICS is used to integrate a SINAMICS drive in Easy Motion Control. The drive may be integrated using PROFIBUS DP or PROFINET IO. You need an EncoderSINAMICS driver instruction for each SINAMICS axis.

Tools required

You need the following tools when using a SINAMICS in Easy Motion Control and to configure a SINAMICS:

- The drive configuration tool STARTER or S7 Technology, including the current SINAMICS Support Package (SSP)

Configuration of SINAMICS (hardware configuration)

Insert a SINAMICS into an isochronous PROFIBUS DP or PROFINET IO segment in your project. The EncoderSINAMICS driver instruction supports the standard frames 3, 5, 102, and 105 for PROFIBUS DP, as well as the standard frames 3 and 102 for PROFINET IO.

You need a suitable frame (5, 105) and an isochronous connection if you are going to use the Dynamic Servo Control (DSC) control method for your application. This DSC functionality is currently not supported for PROFINET IO.

For isochronous processing, make sure that the I/O addresses are in the process image of the CPU. Adjust the addresses manually if necessary. Assign the I/O addresses to a process image partition. An isochronous OB must be assigned to the process image partition in the configuration of the S7 CPU. For more information on the isochronous mode in SIMATIC S7, refer to the Isochronous Mode Function Manual (http://support.automation.siemens.com/WW/view/en/15218045).

Parameter	Setting
I/O address	Specify the first input address for the axis to be used in Easy Motion Control in the axis DB. (Example for slot 4: input address 256 to 273; enter input address 256)
frame selection	A frame number is selected for communication from the SINAMICS configuration wizard and transferred to the hardware configuration. Enter this information at input TelNumber of instruction EncoderSINAMICS as well.

Note

The standard installation of TIA Portal does not support integration of the SINAMICS in a SIMATIC S7 CPU. Check the availability. Install the corresponding GSD file for SINAMICS. The necessary data is not integrated until you completed this additional installation.

Configuration of SINAMICS in the axis DB

Specify the input address set for this axis in the hardware configuration in the "Module address inputs" field.

Input driver EncoderSINAMICS determines the configured cycle time if a constant bus cycle time or an isochronous mode was detected and enters it in the axis DB. You are therefore strongly advised to read back the modified axis DB from the controller in order to apply all changes that may have been made.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description	
RefMode	INPUT	INT	Referencing type:	
			0 = Function 1 Reference mark 1	Referencing to encoder zero mark or external zero mark (p495[0] in SINAMICS)
			1 = Function 2 Reference mark 2	Second encoder zero mark
			2 = Function 3 Reference mark 3	Third encoder zero mark
			3 = Function 4 Reference mark 4	Fourth encoder zero mark
TelNumber	INPUT	INT	Frame number:	
			The following standard frames are supported: 3, 5, 102, and 105	Only the frames 3 and 102 have been approved for PROFINET IO.
EncoderValid	OUTPUT	BOOL	1= actual position values are valid	SINAMICS reports that the position encoder of the relevant axis returns valid values. This input may be used as activation condition at MC_Control.

Parameter	Declaration	Data type	Description	
Err_ID	OUTPUT	WORD	Error code of the instruction:	
			W#16#0001 = Incorrect OB	The instruction may only be called in OB 6x, OB 3x, or OB 1.
			W#16#0002 = Incorrect frame number	The driver does not support the specified number.
			W#16#0004 = Incorrect referencing mode setting	Check the parameter at input RefMode. Valid values are 0, 1, 2, or 3.

Note

The frame number at the input driver must agree with the frame number configured for SINAMICS. Otherwise, correct operation cannot be guaranteed. No error code is output if you make incorrect entries.

Synchronization

Synchronization can be performed by:

- Reference point approach (with incremental encoder)
- Reference point setting (with incremental and absolute encoder)

The referencing type can be specified using input parameter "RefMode" at the input driver. The following table lists the options available for Easy Motion Control.

RefMode	"Velocity" on MC_Home	Encoder zero mark	External zero mark p495[0]	Description
0	> 0	First encoder zero mark	0	Referencing to first encoder zero mark
1	> 0	Second encoder zero mark	0	Referencing to second encoder zero mark
2	> 0	Third encoder zero mark	0	Referencing to third encoder zero mark
3	> 0	Fourth encoder zero mark	0	Referencing to fourth encoder zero mark

The entry at RefMode controls the corresponding encoder control word for the drive so that the required functionality is realized in the SINAMICS. Input driver EncoderSINAMICS enables only the reference point approach to one or several encoder zero marks and to an external zero mark (alternative zero mark). Referencing on-the-fly is not possible. In order to reference to an external zero mark, set the p495[0] parameter of the axis concerned to a value between 1 and 6 value in the SINAMICS. To enable activation of a reference point approach, a traversing velocity greater than zero must be entered at instruction MC_HOME.

If traversing velocity zero has been entered, the contents of the "RefMode" input and the parameter p495[0] will be ignored.

Error handling

In response to a detected error, the input driver sets the "Error" and "Err.EncoderErr" error codes in the axis DB. The input driver resets the "Sync" status bit in the axis DB if you are using an incremental encoder.

For more information on error response, error display, and error acknowledgment, see Error displays and error acknowledgment in the axis DB (Page 10820).

Error acknowledgment

If you set error acknowledgment in the axis DB of Easy Motion Control, errors pending in the module are also acknowledged.

EncoderUniversal

Description

instruction EncoderUniversal is used to integrate position decoder modules for which Easy Motion Control does not provide any special input driver.

Operating principle

instruction EncoderUniversal expects a value at its "EncoderValue" input that corresponds to the actual position of your axis. It does not access the I/O itself. You can interconnect the value provided by your position decoder module directly with the driver, or convert it beforehand.

If your distance measurement module offers special functions you wish to use, you should integrate these into your user program.

Configuration of the module (hardware configuration)

Select parameters according to your hardware and the encoder used.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
EncErr	INPUT	BOOL	1 = Encoder error With this parameter, you inform the input driver of an error detected by the position decoder module or user program (for example, module removed, or station failure).
EncoderValue	INPUT	DINT	Encoder value

Synchronization

Synchronization with instruction EncoderUniversal is only possible by means of reference point setting.

Error handling

Any errors detected by your distance measurement module should be recorded and processed further in your user program.

If such an error impairs the functions you are performing with Easy Motion Control instructions, you should implement this error into Easy Motion Control error handling by means of "EncErr" input.

For detailed information on error response, error display, and error acknowledgment, see Error Displays and Error Acknowledgment in the axis DB (Page 10820).

Output driver

Task of the output driver

Function

The output driver is used to output the velocity setpoint of an axis which has been calculated by the controller to the connected output module. The instruction converts the velocity from [length unit/s] to the corresponding range of values of the output module and outputs this value.

If input "EnableDrive" = FALSE, the output driver outputs zero to the output module.

Initialization

The output value is set to zero.

Call

The output driver of an axis must be called unconditionally.

Isochronous Mode

The output drivers support the isochronous mode of all suitable I/O modules. The driver automatically recognizes isochronous mode by the fact that it is called in an isochronous OB.

Error handling

An error detected by the user program (for example, module removed or station failure) can be signaled to the output driver at input "OutErr" . The output driver does not access the output module as long as input "OutErr" = TRUE.

If input"OutErr" = TRUE, the output driver sets the "Error" and "Err.OutputErr" error codes in the axis DB.

For detailed information on error response, error display, and error acknowledgment, see Error Displays and Error Acknowledgment in the axis DB (Page 10820).

Simulation mode

As long as the "Sim" bit is in the axis DB, the output driver does not access the output module (see instruction MC_Simulation (Page 7050)).

OutputIM178

Description

instruction OutputIM178 is used to employ a channel of the IM 178-4 as output module. instruction OutputIM178 is included for reasons of compatibility with STEP7 V5.x. Siemens is no longer actively marketing the IM 178-4. You can use IM 174 as an alternative.

IM 178-4 must be used with the same channel number and axis both at the input **and** output drivers, as specific address settings refer to both drivers. A combination with other drivers is not possible.

Call

The instruction is called in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
EnableDrive	INPUT	BOOL	1 = Analog output enable If "EnableDrive" = FALSE, a zero value is transferred to the output module. This input can be interconnected with the "DriveEnabled" output of instruction MC_Control.
Q0	INPUT	BOOL	Digital output 0 of selected channel
Q1	INPUT	BOOL	Digital output 1 of selected channel
Q2	INPUT	BOOL	Digital output 2 of selected channel
OutErr	INPUT	BOOL	1 = Error at output module An error detected by the user program (for example, module removed or station failure) can be signaled to the output driver here.

Digital outputs

The digital outputs "Q0" to "Q2"of the selected channel of IM 178-4 can be controlled using the input parameters "Q0" to "Q2" of the output driver.

OutputSM432

Description

instruction OutputSM432 is used to employ a channel of the SM 432 as output module.

Configuration of SM 432 (hardware configuration)

Set the output voltage of the corresponding SM 432 channel to ±10 V.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
EnableDrive	INPUT	BOOL	1 = Analog output enable If "EnableDrive" = FALSE, a zero value is transferred to the output module. This input can be interconnected with the "DriveEnabled" output of instruction MC_Control.
OutErr	INPUT	BOOL	1 = Error at output module An error detected by the user program (for example, module removed or station failure) can be signaled to the output driver here.

OutputET200S2AO

Description

instruction OutputET200S2AO is used to employ a channel of the ET 200S analog output module 2AO U as output module.

The driver can also be used for the isochronous variant of this module.

Configuration of the ET 200S analog output module 2AO U (hardware configuration)

Set the output voltage of the corresponding channel of the module to ±10 V.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
EnableDrive	INPUT	BOOL	1 = Analog output enable If "EnableDrive" = FALSE, a zero value is transferred to the output module. This input can be interconnected with the "DriveEnabled" output of instruction MC_Control.
OutErr	INPUT	BOOL	1 = Error at output module An error detected by the user program (for example, module removed or station failure) can be signaled to the output driver here.

OutputCPU314C

Description

instruction OutputCPU314C is used to employ an analog output channel of CPU 314C for ±10 V output voltage.

Configuration of the analog outputs of CPU 314C (hardware configuration)

Set the corresponding channel (analog output) of CPU 314C to an output voltage of ±10 V.

Set the address that you have assigned in the hardware configuration to the outputs at slot 2.3 (AI5/AO2) in the axis data parameter "Start address of the outputs of output module" ("OutputModuleOutAddress"). It is not necessary to configure the " Start address of the inputs of the output module" parameter for the output module ("OutputModuleInAddress").

Up to 2 analog output channels are available that allow you to operate the 2 axes entirely by means of CPU 314C.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
EnableDrive	INPUT	BOOL	1 = Analog output enable If "EnableDrive" = FALSE, the zero value is transferred to the analog output. This input can be interconnected with the "DriveEnabled" output of instruction MC_Control.
OutErr	INPUT	BOOL	1 = Analog output error Errors detected by the user program can be reported to the output driver here.

OutputSM332

Description

instruction OutputSM332 is used to employ a channel of the SM 332 as output module.

The driver can also be used for the isochronous variant of this module.

Configuration of SM 332 (hardware configuration)

Set the output voltage of the corresponding SM 332 channel to ±10 V.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
EnableDrive	INPUT	BOOL	1 = Analog output enable If "EnableDrive" = FALSE, a zero value is transferred to the output module. This input can be interconnected with the "DriveEnabled" output of instruction MC_Control.
OutErr	INPUT	BOOL	1 = Error at output module An error detected by the user program (for example, module removed or station failure) can be signaled to the output driver here.

OutputUniversal

Description

instruction OutputUniversal allows you to use any analog output module in combination with the instructions of Easy Motion Control.

Operating principle

instruction OutputUniversal calculates a speed setpoint based on the velocity value output by the controller and proportionality ratio "IntAtMaxVelocity" and provides the result at its outputs.

The driver itself does not access the I/O.

"You must use the value of "IntAtMaxVelocity" to control your analog output module so that it outputs a voltage that will move the axis at "MaxVelocity" (you have already declared the value of this voltage in the "DriveInputAtMaxVel" parameter).

The driver curve always passes through zero and follows a linear course.

The speed setpoint is output both as signed integer value and as an amount with a separate sign.

If these output values already correspond to your application, you can output them directly to the I/O via the user program. If the values are not yet suitable, you can also make an additional adjustment via the user program and output the value that is generated.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
EnableDrive	INPUT	BOOL	1 = Analog output enable A zero value is returned at the outputs as soon as "EnableDrive" = FALSE. This input can be interconnected with the "DriveEnabled" output of instruction MC_Control.
IntAtMaxVelocity	INPUT	INT	Integer value for converting the speed setpoint (see below).
OutErr	INPUT	BOOL	1 = Error at output module An error detected on the output module by the user program (for example, module removed or station failure) can be signaled to the output driver here.
OutputValue	OUTPUT	INT	Calculated speed setpoint
OutValAbs	OUTPUT	INT	Absolute speed setpoint value
OutValSign	OUTPUT	BOOL	Sign of speed setpoint (TRUE = Negative value)

OutputMM4_DP

Description

instruction OutputMM4_DP is used to implement of a MICROMASTER 4 with optional DP add-on module.

Configuration of MM4 (hardware configuration)

For information on MICROMASTER 4, refer to the hardware catalog at:

"Additional field devices > PROFIBUS DP > Drives > Siemens AG > SIMOVERT > MICROMASTER 4".

You may first have to install the corresponding GSD file.

Position the MICROMASTER in your DP master system and assign it a DP address. Insert a "0 PKW, 2 PZD (PPO3)" module into slots 0 and 1. Either accept the default I/O addresses or select your own values.

Transfer these addresses to "Start address of the outputs of the output module" or "Start address of the inputs of the output module" in the axis DB for your axis.

Configuring the MM4

Configure the MICROMASTER (motor data, DP address, operation as DP slave, etc.) using the MICROMASTER documentation according to your requirements.

Transfer the "reference frequency" specified in the P2000 parameter as "reference value for 100 % speed" ("DriveInputAt100") to the axis DB for your axis.

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
EnableDrive	INPUT	BOOL	1 = Analog output enable The zero speed value is output to the MICROMASTER as soon as "EnableDrive" = FALSE. This input can be interconnected with the "DriveEnabled" output of instruction MC_Control.
OutErr	INPUT	BOOL	1 = Output error An error detected on the output module by the user program (for example, station failure) can be signaled to the output driver here.
ON	INPUT	BOOL	1 = Drive switched on "ON" in entered in bit 0 in the controller word.
DriveReady	OUTPUT	BOOL	1 = Drive ready If "DriveReady" = FALSE, starting the travel generates a "DriveError".

OutputIM174

Description

instruction OutputIM174 is used to employ a channel of the IM 174 as output module. IM 174 provides up to 4 axes. You need a driver instruction OutputIM174 for each axis for which PROFIBUS frame 3 is set.

Configuration of IM 174 (hardware configuration)

You only need output driver OutputIM174 if you configure an axis with PROFIBUS frame 3. You do not need an output driver for an axis with PROFIBUS frame 81.

Insert an IM 174 at an isochronous PROFIBUS DP in your project. Ensure the I/O addresses are in the process image of the CPU. Adjust the addresses manually if necessary. Assign the I/O addresses to a process image partition. An isochronous OB must be assigned to the process image partition in the configuration of the S7 CPU.

Parameter	Setting
I/O address	Specify the first output address for the axis to be used in Easy Motion Control in the axis DB. (Example for slot 5: For start address 256 to 265, you must enter output address 256)

Note

The hardware configuration implements IM 174 by default as an isochronous module. Isochronous processing is based on the condition that the I/O addresses of IM 174 are in the process image of the CPU.

For more information on the isochronous mode in SIMATIC S7, refer to the Isochronous Mode Function Manual (http://support.automation.siemens.com/WW/view/en/15218045).

The OutputIM174 driver instruction demands isochronous processing. If you ignore this rule, the instruction is prevented from accessing the hardware addresses.

Configuration of IM 174 in the axis DB

Specify the output address set for this axis in the hardware configuration in the "Module address outputs" field.

Call

The instruction is called in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description
EnableDrive	INPUT	BOOL	1 = Analog output enable The zero speed value is output to the drive as soon as "EnableDrive" = FALSE. The drive enable is locked. This input can be interconnected with the "DriveEnabled" output of instruction MC_Control.
OutErr	INPUT	BOOL	1 = Error at output module An error detected by the user program (for example, module removed or station failure) can be signaled to the output driver here.
Err_ID	OUTPUT	WORD	Error code of the instruction W#16#0001 = Incorrect OB; instruction may only be installed in OB 6x

OutputSINAMICS

Description

instruction OutputSINAMICS is used to integrate a SINAMICS drive in Easy Motion Control. The drive may be integrated using PROFIBUS DP or PROFINET IO. You need an OutputSINAMICS driver instruction for each SINAMICS axis.

Configuration of SINAMICS (hardware configuration)

Insert a SINAMICS into an isochronous PROFIBUS DP or PROFINET IO segment in your project. The OutputSINAMICS driver instruction supports the standard frames 3, 5, 102, and 105 for PROFIBUS DP, as well as the standard frames 3 and 102 for PROFINET IO.

You need a suitable frame (5, 105) and an isochronous connection if you are going to use the Dynamic Servo Control (DSC) control method for your application. This DSC functionality is currently not supported for PROFINET IO.

For isochronous processing, make sure that the I/O addresses are in the process image of the CPU. Adjust the addresses manually if necessary. Assign the I/O addresses to a process image partition. An isochronous OB must be assigned to the process image partition in the configuration of the S7 CPU. For more information on the isochronous mode in SIMATIC S7, refer to the Isochronous Mode Function Manual (http://support.automation.siemens.com/WW/view/en/15218045).

Parameter	Setting
I/O address	Specify the first output address for the axis to be used in Easy Motion Control in the axis DB. (Example for slot 5: For start address 256 to 265, you must enter output address 256)

Note

The standard installation of TIA Portal does not support integration of the SINAMICS in a SIMATIC S7 CPU. Check the availability. Install the corresponding GSD file for SINAMICS. The necessary data is not integrated until you completed this additional installation.

Configuration of SINAMICS in the axis DB

Specify the output address set for this axis in the hardware configuration in the "Module address outputs" field. Specify the frame number once again to enable stand-alone operation of the output driver. The program checks if the frame number agrees with that of the corresponding EncoderSINAMICS input driver (if that exists).

Activating the axis

Each SINAMICS motor module inevitably assigned a central power supply (line module) for the DC link. Always ensure that the central power supply is switched on before you activate the axis by setting input "EnableDrive" of instruction MC_Control. The line module returns

corresponding information on its operating status. Evaluate the necessary feedback signals from the line module and link these as power-on conditions.

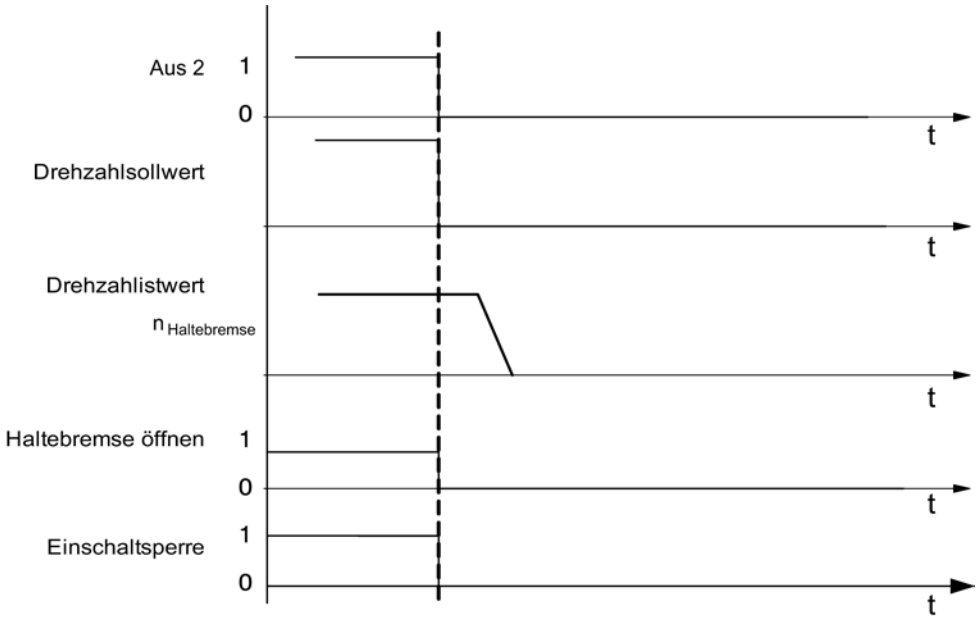
For more information, refer to the SINAMICS Drives documentation (http://www.automation.siemens.com/doconweb).

An example of the application for switching on an active line module is available on the Internet (http://support.automation.siemens.com/WW/view/en/21971603).

Switching off the axis - Configuration of the StopMode

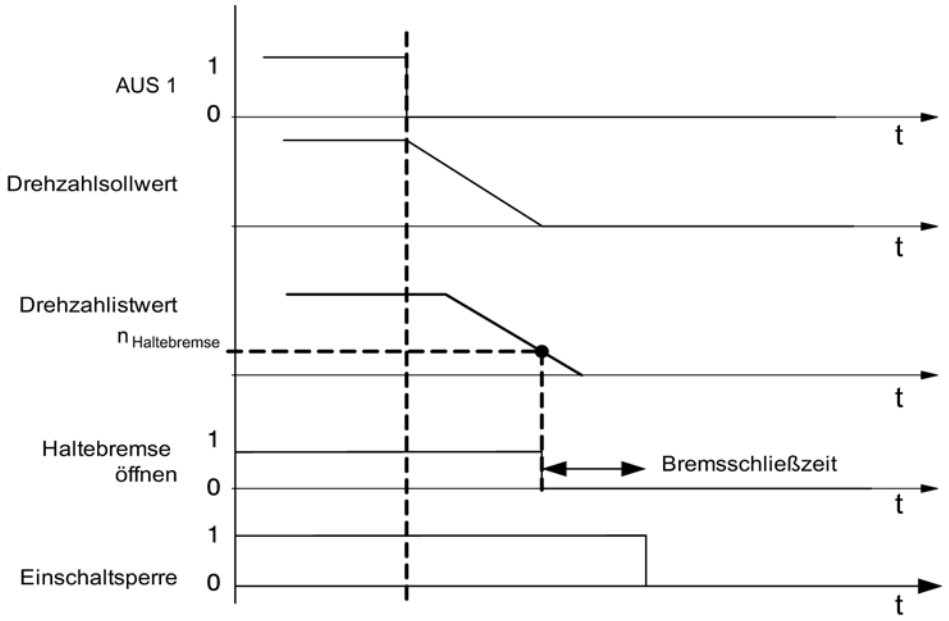
Coasting (coast stop) (AUS2) – StopMode = 3:

In this case, the power enable signal is immediately cleared and the drive coasts to a standstill in the power-off state. The mechanical brake control is enabled immediately and simultaneously. This engages the brake for active deceleration of the drive. Mechanical stress on the brake increases in the process depending on its inertia load.



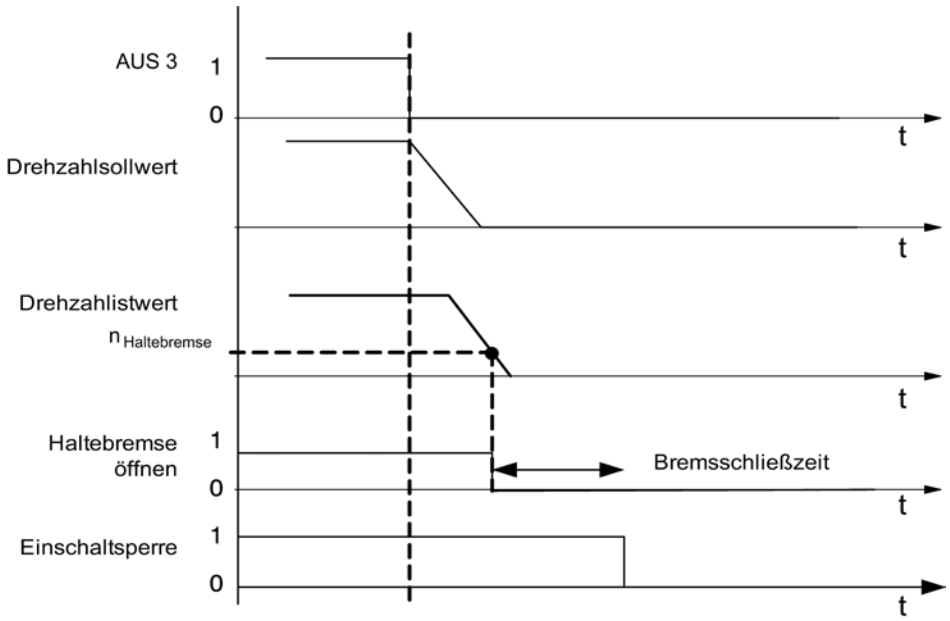
Ramp-down stop (AUS1) – StopMode = 5:

In this mode, the drive is decelerated with its ramp-up function. Once the speed threshold has been reached, the holding brake is engaged and pulse enable is deactivated on expiration of the brake closing time.



Fast stop (AUS3) – StopMode = 6:

In this mode, the drive is decelerated and brought to a stop with the fast stop ramp (emergency stop). Once the speed threshold has been reached, the holding brake is engaged and pulse enable is deactivated on expiration of the brake closing time.



For more information, refer to the SINAMICS S120 product information (http://support.automation.siemens.com/WW/view/en/30119684).

Call

The instruction is called in a cyclic interrupt OB (for example, OB 35), or in a synchronous cycle interrupt OB (for example, OB 61). You must call all EMC instructions of an axis in the same program block.

Parameter

Parameter	Declaration	Data type	Description	
TelNumber	INPUT	INT	Frame number:	
			The following standard frames are supported: 3, 5, 102,105	Only the frames 3 and 102 have been approved for PROFINET IO. If you are using EncoderSINAMICS as input driver, the number of the frame used must always remain the same.
EnableDrive	INPUT	BOOL	1 = Drive enable	The zero speed value is output to the drive as soon as "EnableDrive" = FALSE. The drive enable is locked. This input can be interconnected with the "DriveEnabled" output of instruction MC_Control.
StopMode	INPUT	INT	Stop mode:	
			3 = Coast to stop (AUS2)	The drive coasts to a stop in power-off state
			5 = Ramp stop (AUS1)	Stops the axis by using a ramp function generator
			6 = Quick stop (AUS3)	Stops the axis along a quick stop ramp
OutErr	INPUT	BOOL	1 = Error at output module	An error detected by the user program (for example, module removed or station failure) can be signaled to the output driver here.
Operation_Enabled	OUTPUT	BOOL	1 = SINAMICS reports "Operation enabled"	All axis enable signals are set in the drive and the axis reports that it is ready for movement. The starting current is applied to the axis.
Err_ID	OUTPUT	WORD	Error code of the instruction:	
			W#16#0001 = Incorrect OB	The instruction may only be installed in OB 6x, OB 3x, or OB 1.
			W#16#0002 = Incorrect frame number	The specified number is not supported by the driver, or differs from the frame number that has been configured in the SINAMICS.

OutErr: The drive (SINAMICS) is deactivated on the positive edge of this variable. To reactivate the drive, OutErr = FALSE must be set and the error message must be acknowledged.

Operation_Enabled: This variable reports the current axis status in the SINAMICS. You should not activate a travel instruction while Operation_Enabled = FALSE.

For more information on integration of the SINAMICS, refer to the Internet (http://support.automation.siemens.com/WW/view/en/30119684).

Additional references for Easy Motion Control

Axis data block (axis DB)

Function

The axis DB is the central database of Easy Motion Control. It contains:

- Parameters that provide basic information about the axis (for example, I/O addresses)
- Position control parameters that are calculated from these basic parameters during initialization of the EMC instructions
- Actual values of the position control (for example, position, error states, etc.)
- Initialization bits for coordinating the start of the instructions

Content

The following table shows the content of the axis DB.

Note

You may not alter any data that is not listed in this table.

Address	Parameter	Data type	Initial value	Description
0.0	Sample_T	REAL	0.01	Scan time of the axis instructions [s]
4.0	AxisType	BOOL	FALSE	Axis type • FALSE = Linear axis • TRUE = Rotary axis
4.1	EncoderType	BOOL	FALSE	Encoder type • FALSE = Incremental encoder • TRUE = Absolute encoder
4.2	Sim	BOOL	FALSE	TRUE = Simulation mode is active
6.0	AxisLimitMax	REAL	$1.0e^{+6}$	Software limit switch end [length unit]
10.0	AxisLimitMin	REAL	$-1.0e^{+6}$	Software limit switch start [length unit]
14.0	MaxVelocity	REAL	0.1	Maximum axis velocity [unit of length/s]
18.0	MaxAcceleration	REAL	0.1	Max. axis acceleration in [unit of length/s^2]
22.0	MaxDeceleration	REAL	0.1	Max. axis deceleration in [unit of length/s^2]
26.0	MonTimeTargetAppr	REAL	1.0	Monitoring time for target approach [s]
30.0	TargetRange	REAL	1.0	Target range on positioning [unit of length]
34.0	StandstillRange	REAL	1.5	Standstill range without travel order [unit of length]
38.0	MaxFollowingDist	REAL	5.0	Maximum admissible following error [unit of length]
42.0	MonitorTargetAppr	BOOL	TRUE	TRUE = Enable monitoring of target range
42.1	SWLimitEnable	BOOL	TRUE	TRUE = Enable software limit switch monitoring
44.0	FactorP	REAL	1.0	Proportionality factor [1/s]

Address	Parameter	Data type	Initial value	Description
48.0	ManVelocity	REAL	0.0	Setpoint velocity in manual mode [unit of length/s]
52.0	ManEnable	BOOL	FALSE	TRUE = Enable manual mode
54.0	EmergencyDec	REAL	1000.0	Axis deceleration for emergency stop [length unit/s^2]
58.0	InputModuleInAddr	INT	0	Start address for the position detection module inputs
60.0	InputModuleOutAddr	INT	0	Start address for the position detection module outputs
62.0	InputChannelNo	INT	0	Channel number of position input module
64.0	StepsPerRev	DINT	L#1	Steps per encoder revolution [steps/encoder revolution]
68.0	DisplacementPerRev	REAL	1.0	Distance per encoder revolution [unit of length/encoder revolution]
72.0	NumberRevs	INT	1	Number of revolutions of an absolute encoder
74.0	PolarityEncoder	INT	1	Set encoder polarity
80.0	OutputModuleOutAddr	INT	0	Start address for the output module outputs
82.0	OutputModuleInAddr	INT	0	Start address of the output module inputs
84.0	OutputChannelNo	INT	0	Channel number of the output module
86.0	PolarityDrive	INT	1	Set drive polarity
88.0	DriveInputAtMaxVel	REAL	9.0	Reference value for max. axis velocity [V]
92.0	OffsetCompensation	REAL	0.0	Offset compensation [V]
96.0	DriveInputAt100	REAL	10.0	Reference value for 100% speed [V]
100.0	Override	REAL	100.0	Velocity override [%]
104.0	ActPosition	REAL	0.0	Current actual position [unit of length]
108.0	FollowingDistance	REAL	0.0	Current following error [unit of length]
112.0	RemainingDistance	REAL	0.0	Current residual distance to target [unit of length]
116.0	NomVelocity	REAL	0.0	Setpoint velocity of axis [unit of length/s]
120.0	ActVelocity	REAL	0.0	Actual speed of axis [unit of length/s]
124.0	Sync	BOOL	FALSE	• FALSE = Axis not synchronized • TRUE = Axis synchronized
124.1	Error	BOOL	TRUE	• FALSE = No error • TRUE = Group error
124.2	ErrorAck	BOOL	FALSE	• FALSE = No acknowledgment • TRUE = Acknowledge error
128.0	Err.SWLimitMinExceeded	BOOL	FALSE	TRUE = Software limit switch start exceeded
128.1	Err.SWLimitMaxExceeded	BOOL	FALSE	TRUE = Software limit switch end exceeded
128.2	Err.TargetErr	BOOL	FALSE	TRUE = Target outside travel range
128.3	Err.NoSync	BOOL	FALSE	TRUE = Axis not synchronized
128.4	Err.DirectionErr	BOOL	FALSE	TRUE = Travel in specified direction inadmissible
128.5	Err.DataErr	BOOL	FALSE	TRUE = Invalid parameter in travel instruction
128.6	Err.StartErr	BOOL	FALSE	TRUE = Start not possible in current axis status
128.7	Err.DistanceErr	BOOL	FALSE	TRUE = Overtravel
129.0	Err.MasterErr	BOOL	FALSE	TRUE = Master axis in error state
130.0	Err.StoppedMotion	BOOL	TRUE	TRUE = Axis in stop status requiring acknowledgment
130.1	Err.EnableDriveErr	BOOL	FALSE	TRUE = Drive enable missing

Address	Parameter	Data type	Initial value	Description
130.2	Err.FollowingDistErr	BOOL	FALSE	TRUE = Max. following error exceeded
130.3	Err.StandstillErr	BOOL	FALSE	TRUE = Standstill range exceeded
130.4	Err.TargetApproachErr	BOOL	FALSE	TRUE = Error on target approach
130.5	Err.EncoderErr	BOOL	FALSE	TRUE = Encoder error
130.6	Err.OutputErr	BOOL	FALSE	TRUE = Error at output driver
130.7	Err.ConfigErr	BOOL	FALSE	TRUE = Axis data incorrectly configured
131.0	Err.DriveErr	BOOL	FALSE	TRUE = Drive error
132.0	Config.Err_AxisLimit	BOOL	FALSE	TRUE = Axis limits incorrect
132.1	Config.Err_MaxVelocity	BOOL	FALSE	TRUE = Max. velocity incorrect
132.2	Config.Err_MaxAcceleration	BOOL	FALSE	TRUE = Max. acceleration incorrect
132.3	Config.Err_MaxDeceleration	BOOL	FALSE	TRUE = Max. deceleration incorrect
132.4	Config.Err_MonTimeTargetApr	BOOL	FALSE	TRUE = Monitoring time for target approach incorrect
132.5	Config.Err_TargetRange	BOOL	FALSE	TRUE = Target range incorrect
132.6	Config.Err_StandstillRange	BOOL	FALSE	TRUE = Standstill range incorrect
132.7	Config.Err_MaxFollowingDist	BOOL	FALSE	TRUE = Max. following error incorrect
133.0	Config.Err_EmergencyDec	BOOL	FALSE	TRUE = Deceleration for hard stop incorrect
133.1	Config.Err_StepsPerRev	BOOL	FALSE	TRUE = Steps per encoder revolution incorrect
133.2	Config.Err_DisplacementPerRev	BOOL	FALSE	TRUE = Axis distance per encoder revolution incorrect
133.3	Config.Err_NumberRevs	BOOL	FALSE	TRUE = Incorrect number of encoder revolutions
133.4	Config.Err_PolarityEncoder	BOOL	FALSE	TRUE = Set encoder polarity incorrect
133.5	Config.Err_PolarityDrive	BOOL	FALSE	TRUE = Set drive polarity incorrect
133.6	Config.Err_DriveInputAtMaxVel	BOOL	FALSE	TRUE =reference value for maximum velocity incorrect
133.7	Config.Err_AxisLength	BOOL	FALSE	TRUE = Length of axis incorrect
134.0	Config.Err_EncoderRange	BOOL	FALSE	TRUE = Encoder range is not suitable for axis length
134.1	Config.Err_MaxVelRotaryAxis	BOOL	FALSE	TRUE = Max. velocity and scan time do not match rotary axis module
134.2	Config.Err_DriveInputAt100	BOOL	FALSE	TRUE = Reference value for 100 % speed ≤ 0
136.0	EncoderValue	DINT	L#0	Current encoder value
278.0	Init	STRUCT	TRUE	Bit array for initializing 32 instructions

Function modules

FM x50-1 Counting

CNT_CTRL

Description

The CNT_CTRL instruction is used to control the counting and measuring modes.

You need to use the CNT_CTL1 (Page 7098) instruction to change parameters in runtime or set and reset the outputs of the counter module.

Operating principle

The CNT_CTRL instruction works only in a non-isochronous OB.
In an isochronous OB, use the instruction CNT_CTL1 (Page 7098) or CNT_CTL2 (Page 7100).

The data needed for the CNT_CTRL instruction is stored in the counter DB (Page 7101) in the CPU. The CNT_CTRL instruction cyclically transfers data from this DB to the counter module and fetches data from the counter module.

Call

The CNT_CTRL instruction can be called once for each counter in the cycle or optionally in a time-controlled program. The call is not allowed in an event-driven interrupt program.

Parameter

Parameter	Declaration	Data type	Description	The user ...	The instruction ...
DB_NO	INPUT	INT	Number of the counter data block	Enters this	Queries this
SW_GATE	INPUT	BOOL	Counter control bit SW gate (start/stop)	Sets and resets this	Queries this
GATE_STP	INPUT	BOOL	Counter control bit gate stop	Sets and resets this	Queries this
OT_ERR_A	INPUT	BOOL	Acknowledge operator error	Sets and resets this	Queries this
OT_ERR	OUTPUT	BOOL	Operator error occurred	Queries this	Sets and resets this
L_DIRECT	IN-OUT	BOOL	Trigger bit for direct loading of a counter	Sets this	Queries and resets this
L_PREPAR	IN-OUT	BOOL	Trigger bit for preparatory loading of a counter	Sets this	Queries and resets this
T_CMP_V1	IN-OUT	BOOL	Transfer trigger bit for comparison value 1	Sets this	Queries and resets this

Parameter	Declaration	Data type	Description	The user ...	The instruction ...
T_CMP_V2	IN-OUT	BOOL	Transfer trigger bit for comparison value 2	Sets this	Queries and resets this
RES_SYNC	IN-OUT	BOOL	Delete synchronization status bit	Sets this	Queries and resets this
RES_ZERO	IN-OUT	BOOL	Delete zero crossing status bit	Sets this	Queries and resets this

Executing jobs

Initiate jobs for the counter module using the instruction parameters L_DIRECT, L_PREPAR, T_CMP_V1, T_CMP_V2, RES_SYNC, RES_ZERO, OT_ERR_A and GATE_STP.

Depending on the command, you need to enter the load value or a comparison value prior to the calling of the instruction in the counter DB.

A configured IN/OUT parameter (L_DIRECT, L_PREPAR, T_CMP_V1, T_CMP_V2, RES_SYNC and RES_ZERO) is deleted by the CNT_CTRL instruction upon completion of the command.

Startup characteristics

As soon as the CNT_CTRL instruction detects a startup (CPU or FM startup), any pending command is deferred and the startup is first acknowledged. A command you have already initiated is not lost, but is not executed until the startup is completed.

Reaction to errors

An operator error occurring during the call of the instruction is reported at the OT_ERR parameter. The error information can be read from the counter DB (OT_ERR_B tag). You can then use the OT_ERR_A parameter to acknowledge the operator error. No additional operator error is reported until you acknowledge the previous error.

DIAG_INF

Description

If the trigger parameter is set (IN_DIAG = TRUE), the DIAG_INF instruction reads data record DS 1 from the counter module and makes it available in the counter DB (Page 7101) (starting at DW 54).

Operating principle

The data record is transferred using the RDSYSST instruction. The return code of this instruction (RET_VAL) is copied to the RET_VAL parameter of the DIAG_INF instruction. Once DIAG_INF instruction has been executed, the IN_DIAG trigger parameter is reset and the transfer is reported as completed.

Call

The DIAG_INF instruction can be called in the cycle and in the interrupt program. However, calling it in a time-controlled program is not recommended.

Parameter

Parameter	Declaration	Data type	Description	The user ...	The instruction ...
DB_NO	INPUT	INT	Number of the counter data block	Enters this	Queries this
RET_VAL	OUTPUT	INT	Return code of the instruction RDSYSST	Queries this	Enters this
IN_DIAG	IN-OUT	BOOL	Read trigger bit of diagnostics data record DS 1	Sets and queries this	Resets this

CNT_CTL1

Description

The CNT_CTL1 instruction is used to control the counting and measuring modes.

In addition to the functions in the CNT_CTRL instruction , the CNT_CTL1 instruction enables you to change parameters in runtime and to set and reset the outputs of the counter module.

Operating principle

The data needed for the CNT_CTL1 instruction is stored in the counter DB (Page 7101) in the CPU. The CNT_CTL1 instruction cyclically transfers data from this DB to the counter module and fetches data from the counter module.

Call

The CNT_CTL1 instruction can be called once for each counter in the cycle or optionally in a time-controlled or an isochronous interrupt OB. The call is not allowed in an event-driven interrupt program.

Parameter

Parameter	Declaration	Data type	Description	The user ...	The instruction ...
DB_NO	INPUT	INT	Number of the counter data block	Enters this	Queries this
SW_GATE	INPUT	BOOL	Counter control bit SW gate (start/stop)	Sets and resets this	Queries this
GATE_STP	INPUT	BOOL	Counter control bit gate stop	Sets and resets this	Queries this
OT_ERR_A	INPUT	BOOL	Acknowledge operator error	Sets and resets this	Queries this

Parameter	Declaration	Data type	Description	The user ...	The instruction ...
SET_DO0	INPUT	BOOL	Set/Reset output DO0	Sets and resets this	Queries this
SET_DO1	INPUT	BOOL	Set/Reset output DO1	Sets and resets this	Queries this
OT_ERR	OUTPUT	BOOL	Operator error occurred	Queries this	Sets and resets this
L_DIRECT [2] [3]	IN-OUT	BOOL	**Counting:** Trigger bit for direct and preparatory loading of a counter	Sets this	Queries and resets this
			Measuring: Must NOT be set	-	Queries and resets this
L_PREPAR [2] [3]	IN-OUT	BOOL	**Counting:** Trigger bit for preparatory loading of a counter	Sets this	Queries and resets this
			Measuring: Transfer low limit	Sets this	Queries and resets this
T_CMP_V1 [2]	IN-OUT	BOOL	**Counting:** Transfer trigger bit for comparison value 1	Sets this	Queries and resets this
			Measuring: Transfer high limit	Sets this	Queries and resets this
T_CMP_V2 [2]	IN-OUT	BOOL	**Counting:** Transfer trigger bit for comparison value 2	Sets this	Queries and resets this
			Measuring: Update time	Sets this	Queries and resets this
C_DOPARA [1]	IN-OUT	BOOL	Trigger bit for parameter change	Sets this	Queries and resets this
RES_SYNC	IN-OUT	BOOL	Delete synchronization status bit	Sets this	Queries and resets this
RES_ZERO	IN-OUT	BOOL	Reset status bits for zero pass, overflow, underflow and comparator or measurement end	Sets this	Queries and resets this

[1] You may not set this parameter simultaneously with one of the L_DIRECT, L_PREPAR, T_CMP_V1 or T_CMP_V2 parameters.

[2] You may not set these parameters simultaneously with the C_DOPARA parameter.

[3] You may not set these parameter simultaneously.

Executing jobs

Initiate jobs for FM x50-1 using the instruction parameters L_DIRECT, L_PREPAR, T_CMP_V1, T_CMP_V2, C_DOPARA, RES_SYNC, RES_ZERO and OT_ERR_A.

Depending on the command, enter the corresponding values (load value, comparison value, low limit, high limit and the update time) in the counter DB prior to calling the instruction.

See also:

- Value transfer using CNT_CTL1 (Page 7107)
- DB parameters for value transfer (count modes) (Page 7108)
- DB parameters for value transfer (count modes) (Page 7109)

The CNT_CTL1 instruction deletes an IN-OUT parameter (L_DIRECT, L_PREPAR, T_CMP_V1, T_CMP_V2, C_DOPARA, RES_SYNC and RES_ZERO) when the command is completed. This enables you to recognize that the command has been completed by FM x50-1. If necessary, you can let this information be known to your user program.

Startup characteristics

As soon as the CNT_CTL1 instruction detects a startup (CPU or FM startup), any pending command is deferred and the startup is first acknowledged. A command you have already initiated is not lost, but is not executed until the startup is completed.

Reaction to error

An operator error occurring during the call of the instruction is reported at the OT_ERR parameter. The error information can be read from the counter DB (OT_ERR_B tag). You can then use the OT_ERR_A parameter to acknowledge the operator error. No additional operator error is reported until you acknowledge the previous error.

CNT_CTL2

Description

The function of the CNT_CTL2 instruction is basically the same as the CNT_CTL1 (Page 7098) instruction.

The differences compared to the CNT_CTL1 instruction are discussed in the following section.

Operating principle

The CNT_CTL2 instruction is particularly suitable for applications involving rapid, successive transfer of the same command (for example, load comparison value) to FM x50-1. At the very best, the CNT_CTL1 instruction lets you initiate new jobs at every fifth PROFIBUS DP cycle, while the CNT_CTL2 instruction can handle this at every second PROFIBUS DP cycle.

The instruction is ready to process a command as soon as the corresponding trigger bit is set to 0. The completion of a command is not displayed separately.

If a communication problem or a data or operator error occurs, this can no longer be attributed to a specific command. For this reason, the instruction interrupts command execution and generates an **ID 90** operator error requiring acknowledgment. Once you acknowledged the error by setting the OT_ERR_A parameter, the instruction continues to process any jobs pending.

The acknowledgment of an operator error is successfully completed as soon as OT_ERR parameter is reset. You should keep the OT_ERR_A parameter set until this happens to ensure

reliable acknowledgment. You should not initiate additional jobs until the acknowledgment is successful.

Note

It is not allowed to initiate several value transfers simultaneously with the CNT_CTL2 instruction.

Call

The CNT_CTL2 instruction works only in an isochronous OB.

If the CNT_CTL2 instruction is called in a non-isochronous OB, it generates an **ID 91** operator error. This would also prevent data communication with FM x50-1.

Additional references for FM x50-1 Counter V3

Counter data block

Purpose

The counter DB is the data interface between the user program and FM x50-1. It contains and applies all data necessary to control and operate the module.

Before the parameter assignment of the module, you must provide the following valid data to the counter DB.

- Module address (address 6.0)
- Channel start address (address 8.0)
- Length of the user data interface (address 12.0)

Configuration

The counter DB is divided into different areas:

Channel DB
Instruction parameters, addresses
Transfer area for values to be written
Control interface
Transfer area for values to be read
Error numbers
Feedback interface
Parameters for FM 450-1
Diagnostics data

Contents

The table below shows the contents of the counter DB.

Note

You may not change any data not listed in this table.

Address	Tag	Data type	Initial value	Comment	
				Counting FM 350-1 / FM 450-1	Measuring FM 350-1
Instruction parameters, addresses					
0.0	AR1_BUFFER	DWORD	DW#16#0	AR1 Buffer	AR1 Buffer
4.0	FP	BYTE	B#16#0	Flag byte	Flag byte
5.0	RESERVED	BYTE	B#16#0	Reserved	Reserved
6.0	MOD_ADR	WORD	W#16#0	Module address	Module address
8.0	CH_ADR	DWORD	DW#16#0	Channel address	Channel address
12.0	U_D_LGTH	BYTE	B#16#0	Length user data	Length user data
13.0	A_BYTE_0	BYTE	B#16#0	Reserved	Reserved
Transfer area for values to be written					
14.0	LOAD_VAL [1)]	DINT	L#0	New load value (write user)	Low limit (write user)
18.0	CMP_V1 [1)]	DINT	L#0	New comparison value 1 (write user)	High limit (write user)
22.0	CMP_V2 [1)]	DINT	L#0	New comparison value 2 (write user)	Update time (write user)
Control interface					
26.0	A_BIT0_0	BOOL	FALSE	Reserved	Reserved
26.1	TFB	BOOL	FALSE	Reserved / test free	Reserved
26.2	A_BIT0_2	BOOL	FALSE	Reserved	Reserved
26.3	A_BIT0_3	BOOL	FALSE	Reserved	Reserved
26.4	A_BIT0_4	BOOL	FALSE	Reserved	Reserved
26.5	A_BIT0_5	BOOL	FALSE	Reserved	Reserved
26.6	A_BIT0_6	BOOL	FALSE	Reserved	Reserved
26.7	A_BIT0_7	BOOL	FALSE	Reserved	Reserved
27.0	ENSET_UP [1)]	BOOL	FALSE	Enable setting in forward direction (write user)	-
27.1	ENSET_DN [1)]	BOOL	FALSE	Enable setting in backward direction (write user)	-
27.2	A_BIT1_2	BOOL	FALSE	Reserved	Reserved
27.3	A_BIT1_3	BOOL	FALSE	Reserved	Reserved
27.4	A_BIT1_4	BOOL	FALSE	Reserved	Reserved
27.5	A_BIT1_5	BOOL	FALSE	Reserved	Reserved
27.6	A_BIT1_6	BOOL	FALSE	Reserved	Reserved

Address	Tag	Data type	Initial value	Comment	
				Counting FM 350-1 / FM 450-1	Measuring FM 350-1
27.7	A_BIT1_7	BOOL	FALSE	Reserved	Reserved
28.0	CTRL_DO0 [1]	BOOL	FALSE	Enable digital output DO0 (write user)	Enable digital output DO0 (write user)
28.1	CTRL_DO1 [1]	BOOL	FALSE	Enable digital output DO1 (write user)	Enable digital output DO1 (write user)
28.2	A_BIT2_2	BOOL	FALSE	Reserved	Reserved
28.3	A_BIT2_3	BOOL	FALSE	Reserved	Reserved
28.4	A_BIT2_4	BOOL	FALSE	Reserved	Reserved
28.5	A_BIT2_5	BOOL	FALSE	Reserved	Reserved
28.6	A_BIT2_6	BOOL	FALSE	Reserved	Reserved
28.7	A_BIT2_7	BOOL	FALSE	Reserved	Reserved
29.0	A_BIT3_0	BOOL	FALSE	Reserved	Reserved
29.1	A_BIT3_1	BOOL	FALSE	Reserved	Reserved
29.2	A_BIT3_2	BOOL	FALSE	Reserved	Reserved
29.3	A_BIT3_3	BOOL	FALSE	Reserved	Reserved
29.4	A_BIT3_4	BOOL	FALSE	Reserved	Reserved
29.5	A_BIT3_5	BOOL	FALSE	Reserved	Reserved
29.6	A_BIT3_6	BOOL	FALSE	Reserved	Reserved
29.7	A_BIT3_7	BOOL	FALSE	Reserved	Reserved
Transfer area for values to be read					
30.0	LATCH_LOAD [1]	DINT	L#0	Current load or latch value (read user)	Current measured value (read user)
34.0	ACT_CNTV [1]	DINT	L#0	Current count value (read user)	Current count value (read user)
Error numbers					
38.0	DA_ERR_W [1]	WORD	W#16#0	Data error word (read user)	Data error word (read user)
40.0	OT_ERR_B [1]	BYTE	B#16#0	Operation error byte (read user)	Operation error byte (read user)
Feedback interface					
41.0	E_BIT0_0	BOOL	FALSE	Reserved	Reserved
41.1	STS_TFB	BOOL	FALSE	Reserved / status test free	Reserved
41.2	E_BIT0_2	BOOL	FALSE	FM 350-1: Diagnostic event (read user) FM 450-1: Reserved	Diagnostic event (read user)
41.3	E_BIT0_3	BOOL	FALSE	Reserved	Reserved
41.4	DATA_ERR [1]	BOOL	FALSE	Data error bit (read user)	Data error bit (read user)
41.5	E_BIT0_5	BOOL	FALSE	Reserved	Reserved
41.6	E_BIT0_6	BOOL	FALSE	Reserved	Reserved

Address	Tag	Data type	Initial value	Comment	
				Counting FM 350-1 / FM 450-1	Measuring FM 350-1
41.7	PARA [1]	BOOL	FALSE	Module parameters assigned (read user)	Module parameters assigned (read user)
42.0	E_BYTE_0	BYTE	B#16#0	Reserved	Reserved
43.0	STS_RUN [1]	BOOL	FALSE	Status counter running (read user) This bit corresponds to bit 2^0 of the counter. 0 = LED CR is off 1 = LED CR is lit	Status counter running (read user) This bit corresponds to bit 2^0 of the counter. 0 = LED CR is off 1 = LED CR is lit
43.1	STS_DIR [1]	BOOL	FALSE	Status count direction (read user)	Status count direction (read user)
43.2	STS_ZERO [1]	BOOL	FALSE	Status zero-crossing (read user)	End of measurement (read user)
43.3	STS_OFLW [1]	BOOL	FALSE	Status overflow (read user)	Status overflow (read user)
43.4	STS_UFLW [1]	BOOL	FALSE	Status underflow (read user)	Status underflow (read user)
43.5	STS_SYNC [1]	BOOL	FALSE	Status counter synchronized (read user)	-
43.6	STS_GATE [1]	BOOL	FALSE	Status internal gate (read user)	Status internal gate (read user)
43.7	STS_SW_G [1]	BOOL	FALSE	Status SW gate (read user)	Status SW gate (read user)
44.0	STS_SET [1]	BOOL	FALSE	Status digital input SET (read user)	Status digital input DI Set (read user)
44.1	STS_LATCH [1]	BOOL	FALSE	New latch value (only in isochronous mode) / Reserved	-
44.2	STS_STA [1]	BOOL	FALSE	Status digital input START (read user)	Status digital input DI Start (read user)
44.3	STS_STP [1]	BOOL	FALSE	Status digital input STOP (read user)	Status digital input DI Stop (read user)
44.4	STS_CMP1 [1]	BOOL	FALSE	Status output comparison value 1 (read user)	Status output comparison value 1 (read user)
44.5	STS_CMP2 [1]	BOOL	FALSE	Status output comparison value 2 (read user)	Status output comparison value 2 (read user)
44.6	STS_COMP1 [1]	BOOL	FALSE	Saved status of comparator 1 / Reserved	-
44.7	STS_COMP2 [1]	BOOL	FALSE	Saved status of comparator 2 / Reserved	-
45.0	E_BIT3_0	BOOL	FALSE	Reserved	Reserved
45.1	E_BIT3_1	BOOL	FALSE	Reserved	Reserved
45.2	E_BIT3_2	BOOL	FALSE	Reserved	Reserved

Address	Tag	Data type	Initial value	Comment	
				Counting FM 350-1 / FM 450-1	Measuring FM 350-1
45.3	E_BIT3_3	BOOL	FALSE	Reserved	Reserved
45.4	E_BIT3_4	BOOL	FALSE	Reserved	Reserved
45.5	E_BIT3_5	BOOL	FALSE	Reserved	Reserved
45.6	E_BIT3_6	BOOL	FALSE	Reserved	Reserved
45.7	E_BIT3_7	BOOL	FALSE	Reserved	Reserved
Parameters for FM 450-1					
46.0	ACT_CMP1 [1)]	DINT	L#0	Reserved / Current comparison value 1 (read user)	Reserved
50.0	ACT_CMP2 [1)]	DINT	L#0	Reserved / Current comparison value 2 (read user)	Reserved
The following diagnostic information is entered by the DIAG_INF instruction					
54.0	MDL_DEFECT	BOOL	FALSE	Module fault	Module fault
54.1	INT_FAULT	BOOL	FALSE	Internal error	Internal error
54.2	EXT_FAULT	BOOL	FALSE	External error	External error
54.3	PNT_INFO	BOOL	FALSE	Channel errors (breakdown from DW 58)	Channel errors (breakdown from DW 58)
54.4	EXT_VOLTAGE	BOOL	FALSE	Auxiliary voltage fault	Auxiliary voltage fault
54.5	FLD_CNNCTR	BOOL	FALSE	Front connector	Front connector
54.6	NO_CONFIG	BOOL	FALSE	No parameter assignment	No parameter assignment
54.7	CONFIG_ERR	BOOL	FALSE	Faulty parameters	Faulty parameters
55.0	MDL_TYPE	BYTE	B#16#0	Module type	Module type
56.0	SUB_MDL_ERR	BOOL	FALSE	Wrong/missing interface module	Wrong/missing interface module
56.1	COMM_FAULT	BOOL	FALSE	Communication error	Communication error
56.2	MDL_STOP	BOOL	FALSE	RUN/STOP operating mode display	RUN/STOP operating mode display
56.3	WTCH_DOG_FAULT	BOOL	FALSE	Watchdog (FM)	Watchdog (FM)
56.4	INT_PS_FLT	BOOL	FALSE	Internal power supply fault	Internal power supply fault
56.5	PRIM_BATT_FLT	BOOL	FALSE	Battery monitoring	Battery monitoring
56.6	BCKUP_BATT_FLT	BOOL	FALSE	Bad buffer	Bad buffer
56.7	RESERVED_2	BOOL	FALSE	Reserved	Reserved
57.0	RACK_FLT	BOOL	FALSE	Rack fault	Rack fault
57.1	PROC_FLT	BOOL	FALSE	CPU fault	CPU fault
57.2	EPROM_FLT	BOOL	FALSE	EPROM fault	EPROM fault
57.3	RAM_FLT	BOOL	FALSE	RAM fault	RAM fault
57.4	ADU_FLT	BOOL	FALSE	ADU fault	ADU fault
57.5	FUSE_FLT	BOOL	FALSE	Fuse	Fuse
57.6	HW_INTR_FLT	BOOL	FALSE	Hardware interrupt lost	Hardware interrupt lost
57.7	RESERVED_3	BOOL	FALSE	Reserved	Reserved

Address	Tag	Data type	Initial value	Comment	
				Counting FM 350-1 / FM 450-1	Measuring FM 350-1
58.0	CH_TYPE	BYTE	B#16#0	Channel type	Channel type
59.0	LGTH_DIA	BYTE	B#16#0	Length of the diagnostic data per channel	Length of the diagnostic data per channel
60.0	CH_NO	BYTE	B#16#0	Channel number	Channel number
61.0	GRP_ERR1	BOOL	FALSE	Group error channel 1	Group error channel 1
61.1	GRP_ERR2	BOOL	FALSE	Reserved / Group error channel 2	Not used with FM 350
61.2	D_BIT7_2	BOOL	FALSE	DS1 byte 7 bit 2	DS1 byte 7 bit 2
61.3	D_BIT7_3	BOOL	FALSE	DS1 byte 7 bit 3	DS1 byte 7 bit 3
61.4	D_BIT7_4	BOOL	FALSE	DS1 byte 7 bit 4	DS1 byte 7 bit 4
61.5	D_BIT7_5	BOOL	FALSE	DS1 byte 7 bit 5	DS1 byte 7 bit 5
61.6	D_BIT7_6	BOOL	FALSE	DS1 byte 7 bit 6	DS1 byte 7 bit 6
61.7	D_BIT7_7	BOOL	FALSE	DS1 byte 7 bit 7	DS1 byte 7 bit 7
62.0	CH1_SIGA	BOOL	FALSE	Channel 1, error signal A	Channel 1, error signal A
62.1	CH1_SIGB	BOOL	FALSE	Channel 1, error signal B	Channel 1, error signal B
62.2	CH1_SIGZ	BOOL	FALSE	Channel 1, error zero signal	Channel 1, error zero signal
62.3	CH1_BETW	BOOL	FALSE	Channel 1, error between channels	Channel 1, error between channels
62.4	CH1_5V2	BOOL	FALSE	Channel 1, error sensor power supply 5.2V	Channel 1, error sensor power supply 5.2V
62.5	D_BIT8_5	BOOL	FALSE	DS1 byte 8 bit 5	DS1 byte 8 bit 5
62.6	D_BIT8_6	BOOL	FALSE	DS1 byte 8 bit 6	DS1 byte 8 bit 6
62.7	D_BIT8_7	BOOL	FALSE	DS1 byte 8 bit 7	DS1 byte 8 bit 7
63.0	D_BYTE9	BYTE	B#16#0	DS1 byte 9	DS1 byte 9
64.0	CH2_SIGA	BOOL	FALSE	Reserved / Channel 2, error signal A	Reserved
64.1	CH2_SIGB	BOOL	FALSE	Reserved / Channel 2, error signal B	Reserved
64.2	CH2_SIGZ	BOOL	FALSE	Reserved / Channel 2, error zero signal	Reserved
64.3	CH2_BETW	BOOL	FALSE	Reserved / Channel 2, error between channels	Reserved
64.4	CH2_5V2	BOOL	FALSE	Reserved / Channel 2, error sensor power supply 5.2V	Reserved
64.5	D_BIT10_5	BOOL	FALSE	DS1 byte 10 bit 5	DS1 byte 10 bit 5
64.6	D_BIT10_6	BOOL	FALSE	DS1 byte 10 bit 6	DS1 byte 10 bit 6
64.7	D_BIT10_7	BOOL	FALSE	DS1 byte 10 bit 7	DS1 byte 10 bit 7
65.0	D_BYTE11	BYTE	B#16#0	DS1 byte 11	DS1 byte 11
66.0	D_BYTE12	BYTE	B#16#0	DS1 byte 12	DS1 byte 12
67.0	D_BYTE13	BYTE	B#16#0	DS1 byte 13	DS1 byte 13
68.0	D_BYTE14	BYTE	B#16#0	DS1 byte 14	DS1 byte 14

Address	Tag	Data type	Initial value	Comment	
				Counting FM 350-1 / FM 450-1	Measuring FM 350-1
69.0	D_BYTE15	BYTE	B#16#0	DS1 byte 15	DS1 byte 15
[1] Tags in the DB that you can/must input or read when working with the FM x50-1					

Value transfer using CNT_CTL1

Value transfer

Transfer the operating mode-specific values by setting these instruction parameters:

Mode	Instruction parameters
Counting	L_DIRECT, L_PREPAR, T_CMP_V1, T_CMP_V2, C_DOPARA
Measuring	L_PREPAR, T_CMP_V1, T_CMP_V2, C_DOPARA

You must first place the values to be transferred in the DB beforehand.

See also:

- DB parameters for value transfer (count modes) (Page 7108)
- DB parameters for value transfer (count modes) (Page 7109)

You can transfer several values at the same time:

In operating mode ...	... You can transfer at the same time	
Counting	Load value	(DB parameter LOAD_VAL)
	Comparison value 1	(DB parameter CMP_V1)
	Comparison value 2	(DB parameter CMP_V2)
Measuring	Low limit	(DB parameter LOAD_VAL)
	High limit	(DB parameter CMP_V1)
	Updating time	(DB parameter CMP_V2)

If an incorrect value is transferred, you have to acknowledge this operator error with OT_ERR_A before FM 350-1 can accept further values. You should then correct the value rejected with the operator error and transfer it again.

Note

If you use the L_DIRECT, L_PREPAR, T_CMP_V1 or T_CMP_V2 instruction parameters to load the LOAD_VAL, CMP_V1 or CMP_V2 values, you cannot use the C_DOPARA instruction parameter at the same time to reconfigure parameters.

This would cause an OT_ERR operating error requiring acknowledgment with OT_ERR_A.

Time required to transfer values

Refer to the table below for the time required to transfer values:

Using FM 350-1		Time requirement
Centralized		At least 4 OB 1 cycles
Distributed (non-isochrone mode)		At least 5 PROFIBUS DP cycles
Distributed (isochrone mode)		
• At transmission of only one value		• 5 PROFIBUS DP cycles
• At simultaneous initiation of the transmission of several values		
	• For the 1st value: • For the 2nd value: • For the 3rd value:	• 5 PROFIBUS DP cycles after initiation • 6 PROFIBUS DP cycles after initiation • 7 PROFIBUS DP cycles after initiation

DB parameters for transferring values (counting modes)

Preparation

Before calling the CNT_CTRL (Page 7096), CNT_CTL1 (Page 7098) or CNT_CTL2 (Page 7100) instruction, you need to write the following valid data to the data block:

- Module address (DB address 6.0)
- Channel start address (DB address 8.0)
- Length of the user data interface (DB address 12.0)

These data should be transferred to the DB in OB 100 at every startup.

DB parameters for transferring values (counting modes)

The following table shows the DB area to which you transfer the LOAD_VAL, CMP_V1 and CMP_V2 parameters.

Parameter LOAD_VAL (bytes 14 to 17) has two meanings:

- If you set the L_DIRECT or L_PREPAR instruction parameter, LOAD_VAL is interpreted as a load value.
- If you set instruction parameter C_DOPARA, you can use byte 14 to define the characteristics of outputs DO0 and DO1. Bytes 15 and 16 are interpreted as hysteresis and pulse duration.

You can set only one of the L_DIRECT, L_PREPAR and C_DOPARA instruction parameters at any time.

DB address	Parameters	Data type	Meaning				
14.0	LOAD_VAL	DINT	Load value; direct and preparatory loading with instruction parameter L_DIRECT Load value; preparatory loading with instruction parameter L_PREPAR				
			Characteristics of the DO0 and DO1 outputs, the hysteresis and the pulse duration; defined with the C_DOPARA instruction parameter:				
			Bit 3	**Bit 2**	**Bit 1**	**Bit 0**	**Reaction of output DO0**
			x	0	0	0	Inactive
			x	0	0	1	Active from the comparison value to high limit
			x	0	1	0	Active from the comparison value to low limit
			x	0	1	1	Active on reaching the comparison value for pulse duration (up/down)
			x	1	0	0	Active on reaching the comparison value for pulse duration (up)
			x	1	0	1	Active on reaching the comparison value for pulse duration (down)
			Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Characteristics of output DO1**
			x	0	0	0	Inactive
			x	0	0	1	Active from the comparison value to high limit
			x	0	1	0	Active from the comparison value to low limit
			x	0	1	1	Active on reaching the comparison value for pulse duration (up/down)
			x	1	0	0	Active on reaching the comparison value of the up count pulse width
			x	1	0	1	Active on reaching the comparison value of the down count pulse width
			x	1	1	0	Switch at comparison values
15.0			Hysteresis (value range 0 to 255)				
16.0			Pulse duration (value range 0 to 250)				
17.0			Reserve = 0				
18.0	CMP_V1	DINT	Comparison value 1; load with the T_CMP_V1 instruction parameter				
22.0	CMP_V2	DINT	Comparison value 2; load with the T_CMP_V2 instruction parameter				
x = irrelevant							

DB parameters for transferring values (measuring modes)

Preparation

Before calling the CNT_CTRL (Page 7096), CNT_CTL1 (Page 7098) or CNT_CTL2 (Page 7100) instruction, you need to write the following valid data to the data block:

- Module address (DB address 6.0)
- Channel start address (DB address 8.0)
- Length of the user data interface (DB address 12.0)

These data should be transferred to the DB in OB 100 at every startup.

DB parameters for transferring values (measuring modes)

The following table shows the DB area to which you transfer the LOAD_VAL, CMP_V1 and CMP_V2 parameters.

Parameter LOAD_VAL (bytes 14 to 17) has two meanings:

- If you set the L_PREPAR instruction parameter, LOAD_VAL is interpreted as a low limit.
- If you set the C_DOPARA instruction parameter, you can use byte 14 to specify the characteristics of output DO0.

You can set only one of the L_PREPAR and C_DOPARA instruction parameters at any time.

DB address	Parameters	Data type	Meaning			
14.0	LOAD_VAL	DINT	Low limit; load with the L_PREPAR instruction parameter			
			Characteristics of DO0; specify in the C_DOPARA instruction parameter			
			Bits 2 to 7	**Bit 1**	**Bit 0**	**Reaction of output DO0**
			Irrelevant	0	0	No comparison
			Irrelevant	0	1	Active outside the limits
			Irrelevant	1	0	Active upon violation of low limit
			Irrelevant	1	1	Active upon violation of high limit
15.0			Reserve = 0			
16.0			Reserve = 0			
17.0			Reserve = 0			
18.0	CMP_V1	DINT	High limit; load with the T_CMP_V1 instruction parameter			
22.0	CMP_V2	DINT	Update time; load with the T_CMP_V2 instruction parameter			

FM 350-2 Counting

CNT2WRPN

Description

You can use the CNT2WRPN instruction to load the counters and comparators of the counter module using write jobs. To do this, you need to call the CNT2WRPN instruction once for each module.

You should integrate the CNT2WRPN instruction in the user program only if it is necessary to reload counters and comparators in runtime.

Operating principle

The CNT2WRPN instruction executes the write command (Page 7121) (JOB_WR) from the counter DB (Page 7113), transfers the corresponding data from the counter DB and indicates thestatus of the write command (Page 7122). The write command is handled internally with the WRREC instruction and the return code is passed to the WRREC instruction.

Call

The CNT2WRPN instruction can be called in the cycle or optionally in a time-controlled program. The call is not allowed in an event-driven interrupt program.

Before executing write jobs, you must assign the appropriate values in the data area associated with the write command. The last write command must be completed, which means that JOB_WR.NO (data byte DBB0) is deleted from the counter DB (Page 7113).

Parameters

Parameters	Declaration	Data type	Description	The user ...	The instruction ...
DB_NO	INPUT	WORD	Number of the counter data block	Enters this	Queries this
RET_VAL	OUTPUT	INT	Return code of the instruction WRREC	Queries this	Enters this

CNT2RDPN

Description

You can use the CNT2RDPN instruction to read the counter values and measured values of the counter module using read jobs. Call the CNT2RDPN instruction cyclically once for each module to do this.

Do not integrate the CNT2RDPN instruction in your user program unless you are not processing read jobs.

Operating principle

The instruction executes the read command (Page 7123) (JOB_RD) from the counter DB (Page 7113), transfers the corresponding data to the counter DB and indicates the status of the read command (Page 7124). The read command is handled internally with the RDREC instruction and the return code is passed to the RDREC instruction.

Call

The CNT2RDPN instruction can be called in the cycle or optionally in a time-controlled program. The call is not allowed in an event-driven interrupt program.

The last read command must be completed, which means that JOB_RD.NO (data byte DBB2) is deleted from the counter DB (Page 7113).

Parameters

Parameters	Declaration	Data type	Description	The user ...	The instruction ...
DB_NO	INPUT	WORD	Number of the counter data block	Enters this	Queries this
RET_VAL	OUTPUT	INT	Return code of the instruction RDREC	Queries this	Enters this

CNT2_CTR

Description

The CNT2_CTR instruction is used to control the digital outputs (enable and disable) and software gates of the counter module. In addition, you receive the feedback signals of the module.

Operating principle

The CNT2_CTR instruction

- initializes the counter DB (Page 7113).
- Transfers the counter signals from the counter DB (CONTROL_SIGNALS (Page 7120) structure) module.
- Reads the feedback signals The instruction in the counter DB saves the read values to the CHECKBACK_SIGNALS (Page 7120) structure.

Call

You can use the CNT2_CTR instruction cyclically (in OB1 or in the cyclic interrupts) for each module. Calling in an interrupt program is not permitted.

Before calling the CNT2_CTR instruction, enter the current control signals in the CONTROL_SIGNALS structure in the counter DB (Page 7113). When the call of the CNT2_CTR instruction is completed, the feedback signals in the CHECKBACK_SIGNALS structure are updated in the counter DB for further processing.

Parameters

Parameters	Declaration	Data type	Description	The user ...	The instruction ...
DB_NO	INPUT	WORD	Number of the counter data block	Enters this	Queries this

DIAG_RD

Description

The DIAG_RD instruction can be used to load the data of diagnostic interrupts in the counter DB.

Operating principle

The DIAG_RD instruction reads 16 bytes of diagnostic data from the counter module and enters these data in thecounter DB (Page 7113) in the data area DIAGNOSTIC_IN_INFO. These data contain the diagnostic status of the entire module (across all counters). Diagnostic data is read

internally at the call of the RDSYSST instruction. The return code of this instruction is passed as RET_VAL.

Call

The DIAG_RD instruction can be called only in the interrupt program.

Parameters

Parameters	Declaration	Data type	Description	The user ...	The instruction ...
DB_NO	INPUT	WORD	Number of the counter data block	Enters this	Queries this
RET_VAL	OUTPUT	INT	Return code of the instruction RDSYSST	Queries this	Enters this

Additional references for FM 350-2 Counter V2

Counter data block

Purpose

The counter DB is the data interface between the user program and FM 350-2. It contains and applies all data necessary to control and operate the module.

Before the parameter assignment of the module, you must provide the following valid data to the counter DB.

- Module address (address 12.0)
- Channel start address (address 14.0)
- DS-OFFSET (address 18.0), set to 0

Contents

The table below shows the contents of the counter DB.

Note

You may not change any data not listed in this table.

Address	Tag	Data type	Initial value	Comment
0.0	NO	BYTE	B#16#0	Number
1.0	BUSY	BOOL	FALSE	TRUE: Write command in progress FALSE: Write command not in progress
1.1	DONE	BOOL	FALSE	TRUE: Write command completed FALSE: Write command not completed

Address	Tag	Data type	Initial value	Comment
1.2	IMPOSS	BOOL	FALSE	TRUE: Write command not possible FALSE: Write command possible
1.3	UNKNOWN	BOOL	FALSE	TRUE: Write command unknown FALSE: Write command known
2.0	NO	BYTE	B#16#0	Number
3.0	BUSY	BOOL	FALSE	TRUE: Read command in progress FALSE: Read command not in progress
3.1	DONE	BOOL	FALSE	TRUE: Read command completed FALSE: Read command not completed
3.2	IMPOSS	BOOL	FALSE	TRUE: Read command not possible FALSE: Read command possible
3.3	UNKNOWN	BOOL	FALSE	TRUE: Read command unknown FALSE: Read command known
4.0	RESERV_0	ARRAY [1..3] OF WORD	W#16#0	Reserved
10.0	RESERV_1	WORD	W#16#0	Reserved
12.0	MOD_ADR	WORD	W#16#0	Module address
14.0	CH_ADR	DWORD	DW#16#0	Channel address
18.0	DS_OFFS	BYTE	B#16#0	Offset data block
19.0	RESERV_2	BYTE	B#16#0	Reserved
20.0	BIT0_0	BOOL	FALSE	Reserved
20.1	BIT0_1	BOOL	FALSE	Reserved
20.2	BIT0_2	BOOL	FALSE	Reserved
20.3	BIT0_3	BOOL	FALSE	Reserved
20.4	BIT0_4	BOOL	FALSE	Reserved
20.5	BIT0_5	BOOL	FALSE	Reserved
20.6	BIT0_6	BOOL	FALSE	Reserved
20.7	BIT0_7	BOOL	FALSE	Reserved
21.0	CTRL_DQ0	BOOL	FALSE	TRUE: Output 0 enabled FALSE: Output 0 not enabled
21.1	CTRL_DQ1	BOOL	FALSE	TRUE: Output 1 enabled FALSE: Output 1 not enabled
21.2	CTRL_DQ2	BOOL	FALSE	TRUE: Output 2 enabled FALSE: Output 2 not enabled
21.3	CTRL_DQ3	BOOL	FALSE	TRUE: Output 3 enabled FALSE: Output 3 not enabled
21.4	CTRL_DQ4	BOOL	FALSE	TRUE: Output 4 enabled FALSE: Output 4 not enabled
21.5	CTRL_DQ5	BOOL	FALSE	TRUE: Output 5 enabled FALSE: Output 5 not enabled
21.6	CTRL_DQ6	BOOL	FALSE	TRUE: Output 6 enabled FALSE: Output 6 not enabled

Address	Tag	Data type	Initial value	Comment
21.7	CTRL_DQ7	BOOL	FALSE	TRUE: Output 7 enabled FALSE: Output 7 not enabled
22.0	SET_DQ0	BOOL	FALSE	TRUE: Output 0 set FALSE: Output 0 not set
22.1	SET_DQ1	BOOL	FALSE	TRUE: Output 1 set FALSE: Output 1 not set
22.2	SET_DQ2	BOOL	FALSE	TRUE: Output 2 set FALSE: Output 2 not set
22.3	SET_DQ3	BOOL	FALSE	TRUE: Output 3 set FALSE: Output 3 not set
22.4	SET_DQ4	BOOL	FALSE	TRUE: Output 4 set FALSE: Output 4 not set
22.5	SET_DQ5	BOOL	FALSE	TRUE: Output 5 set FALSE: Output 5 not set
22.6	SET_DQ6	BOOL	FALSE	TRUE: Output 6 set FALSE: Output 6 not set
22.7	SET_DQ7	BOOL	FALSE	TRUE: Output 7 set FALSE: Output 7 not set
23.0	SW_GATE0	BOOL	FALSE	TRUE: SW gate counter 0 open FALSE: SW gate counter 0 closed
23.1	SW_GATE1	BOOL	FALSE	TRUE: SW gate counter 1 open FALSE: SW gate counter 1 closed
23.2	SW_GATE2	BOOL	FALSE	TRUE: SW gate counter 2 open FALSE: SW gate counter 2 closed
23.3	SW_GATE3	BOOL	FALSE	TRUE: SW gate counter 3 open FALSE: SW gate counter 3 closed
23.4	SW_GATE4	BOOL	FALSE	TRUE: SW gate counter 4 open FALSE: SW gate counter 4 closed
23.5	SW_GATE5	BOOL	FALSE	TRUE: SW gate counter 5 open FALSE: SW gate counter 5 closed
23.6	SW_GATE6	BOOL	FALSE	TRUE: SW gate counter 6 open FALSE: SW gate counter 6 closed
23.7	SW_GATE7	BOOL	FALSE	TRUE: SW gate counter 7 open FALSE: SW gate counter 7 closed
24.0	CTRL_DWORD1	DWORD	DW#16#0	Reserved
28.0	CTRL_DWORD2	DWORD	DW#16#0	Reserved
32.0	CTRL_DWORD3	DWORD	DW#16#0	Reserved
36.0	BIT0_0	BOOL	FALSE	Reserved
36.1	STS_TFB	BOOL	FALSE	TRUE: PG mode active FALSE: No PG mode
36.2	BIT0_2	BOOL	FALSE	Reserved
36.3	BIT0_3	BOOL	FALSE	Reserved
36.4	DATA_ERR	BOOL	FALSE	Data error

Address	Tag	Data type	Initial value	Comment
36.5	BIT0_5	BOOL	FALSE	Reserved
36.6	BIT0_6	BOOL	FALSE	Reserved
36.7	PARA	BOOL	FALSE	TRUE: Module parameters assigned FALSE: Module parameters not assigned
37.0	STS_CMP0	BOOL	FALSE	TRUE: Comparator 0 addressed FALSE: Comparator 0 not addressed
37.1	STS_CMP1	BOOL	FALSE	TRUE: Comparator 1 addressed FALSE: Comparator 1 not addressed
37.2	STS_CMP2	BOOL	FALSE	TRUE: Comparator 2 addressed FALSE: Comparator 2 not addressed
37.3	STS_CMP3	BOOL	FALSE	TRUE: Comparator 3 addressed FALSE: Comparator 3 not addressed
37.4	STS_CMP4	BOOL	FALSE	TRUE: Comparator 4 addressed FALSE: Comparator 4 not addressed
37.5	STS_CMP5	BOOL	FALSE	TRUE: Comparator 5 addressed FALSE: Comparator 5 not addressed
37.6	STS_CMP6	BOOL	FALSE	TRUE: Comparator 6 addressed FALSE: Comparator 6 not addressed
37.7	STS_CMP7	BOOL	FALSE	TRUE: Comparator 7 addressed FALSE: Comparator 7 not addressed
38.0	STS_UFLW0	BOOL	FALSE	TRUE: Underflow counter 0 FALSE: No underflow counter 0
38.1	STS_UFLW1	BOOL	FALSE	TRUE: Underflow counter 1 FALSE: No underflow counter 1
38.2	STS_UFLW2	BOOL	FALSE	TRUE: Underflow counter 2 FALSE: No underflow counter 2
38.3	STS_UFLW3	BOOL	FALSE	TRUE: Underflow counter 3 FALSE: No underflow counter 3
38.4	STS_UFLW4	BOOL	FALSE	TRUE: Underflow counter 4 FALSE: No underflow counter 4
38.5	STS_UFLW5	BOOL	FALSE	TRUE: Underflow counter 5 FALSE: No underflow counter 5
38.6	STS_UFLW6	BOOL	FALSE	TRUE: Underflow counter 6 FALSE: No underflow counter 6
38.7	STS_UFLW7	BOOL	FALSE	TRUE: Underflow counter 7 FALSE: No underflow counter 7
39.0	STS_OFLW0	BOOL	FALSE	TRUE: Overflow counter 0 FALSE: No overflow counter 0
39.1	STS_OFLW1	BOOL	FALSE	TRUE: Overflow counter 1 FALSE: No overflow counter 1
39.2	STS_OFLW2	BOOL	FALSE	TRUE: Overflow counter 2 FALSE: No overflow counter 2

Address	Tag	Data type	Initial value	Comment
39.3	STS_OFLW3	BOOL	FALSE	TRUE: Overflow counter 3 FALSE: No overflow counter 3
39.4	STS_OFLW4	BOOL	FALSE	TRUE: Overflow counter 4 FALSE: No overflow counter 4
39.5	STS_OFLW5	BOOL	FALSE	TRUE: Overflow counter 5 FALSE: No overflow counter 5
39.6	STS_OFLW6	BOOL	FALSE	TRUE: Overflow counter 6 FALSE: No overflow counter 6
39.7	STS_OFLW7	BOOL	FALSE	TRUE: Overflow counter 7 FALSE: No overflow counter 7
40.0	STS_DIR0	BOOL	FALSE	TRUE: Count direction counter 0 backwards FALSE: Count direction counter 0 forwards
40.1	STS_DIR1	BOOL	FALSE	TRUE: Count direction counter 1 backwards FALSE: Count direction counter 1 forwards
40.2	STS_DIR2	BOOL	FALSE	TRUE: Count direction counter 2 backwards FALSE: Count direction counter 2 forwards
40.3	STS_DIR3	BOOL	FALSE	TRUE: Count direction counter 3 backwards FALSE: Count direction counter 3 forwards
40.4	STS_DIR4	BOOL	FALSE	TRUE: Count direction counter 4 backwards FALSE: Count direction counter 4 forwards
40.5	STS_DIR5	BOOL	FALSE	TRUE: Count direction counter 5 backwards FALSE: Count direction counter 5 forwards
40.6	STS_DIR6	BOOL	FALSE	TRUE: Count direction counter 6 backwards FALSE: Count direction counter 6 forwards
40.7	STS_DIR7	BOOL	FALSE	TRUE: Count direction counter 7 backwards FALSE: Count direction counter 7 forwards
41.0	STS_DI0	BOOL	FALSE	TRUE: Digital input 0 set FALSE: Digital input 0 not set
41.1	STS_DI1	BOOL	FALSE	TRUE: Digital input 1 set FALSE: Digital input 1 not set
41.2	STS_DI2	BOOL	FALSE	TRUE: Digital input 2 set FALSE: Digital input 2 not set
41.3	STS_DI3	BOOL	FALSE	TRUE: Digital input 3 set FALSE: Digital input 3 not set
41.4	STS_DI4	BOOL	FALSE	TRUE: Digital input 4 set FALSE: Digital input 4 not set
41.5	STS_DI5	BOOL	FALSE	TRUE: Digital input 5 set FALSE: Digital input 5 not set
41.6	STS_DI6	BOOL	FALSE	TRUE: Digital input 6 set FALSE: Digital input 6 not set
41.7	STS_DI7	BOOL	FALSE	TRUE: Digital input 7 set FALSE: Digital input 7 not set

Address	Tag	Data type	Initial value	Comment
42.0	STS_DQ0	BOOL	FALSE	TRUE: Digital output 0 set FALSE: Digital output 0 not set
42.1	STS_DQ1	BOOL	FALSE	TRUE: Digital output 1 set FALSE: Digital output 1 not set
42.2	STS_DQ2	BOOL	FALSE	TRUE: Digital output 2 set FALSE: Digital output 2 not set
42.3	STS_DQ3	BOOL	FALSE	TRUE: Digital output 3 set FALSE: Digital output 3 not set
42.4	STS_DQ4	BOOL	FALSE	TRUE: Digital output 4 set FALSE: Digital output 4 not set
42.5	STS_DQ5	BOOL	FALSE	TRUE: Digital output 5 set FALSE: Digital output 5 not set
42.6	STS_DQ6	BOOL	FALSE	TRUE: Digital output 6 set FALSE: Digital output 6 not set
42.7	STS_DQ7	BOOL	FALSE	TRUE: Digital output 7 set FALSE: Digital output 7 not set
43.0	STS_GATE0	BOOL	FALSE	TRUE: Internal gate counter 0 open FALSE: Internal gate counter 0 closed
43.1	STS_GATE1	BOOL	FALSE	TRUE: Internal gate counter 1 open FALSE: Internal gate counter 1 closed
43.2	STS_GATE2	BOOL	FALSE	TRUE: Internal gate counter 2 open FALSE: Internal gate counter 2 closed
43.3	STS_GATE3	BOOL	FALSE	TRUE: Internal gate counter 3 open FALSE: Internal gate counter 3 closed
43.4	STS_GATE4	BOOL	FALSE	TRUE: Internal gate counter 4 open FALSE: Internal gate counter 4 closed
43.5	STS_GATE5	BOOL	FALSE	TRUE: Internal gate counter 5 open FALSE: Internal gate counter 5 closed
43.6	STS_GATE6	BOOL	FALSE	TRUE: Internal gate counter 6 open FALSE: Internal gate counter 6 closed
43.7	STS_GATE7	BOOL	FALSE	TRUE: Internal gate counter 7 open FALSE: Internal gate counter 7 closed
44	USER_STAT_WORD0	WORD	W#16#0	Count/measured value, depending on parameter assignment
46	USER_STAT_WORD1	WORD	W#16#0	Count/measured value, depending on parameter assignment
48	USER_STAT_WORD2	WORD	W#16#0	Count/measured value, depending on parameter assignment
50	USER_STAT_WORD3	WORD	W#16#0	Count/measured value, depending on parameter assignment
52	LOAD-VAL0	DINT	L#0	Load counter 0 directly
56	LOAD-VAL1	DINT	L#0	Load counter 1 directly
60	LOAD-VAL2	DINT	L#0	Load counter 2 directly
64	LOAD-VAL3	DINT	L#0	Load counter 3 directly

Address	Tag	Data type	Initial value	Comment
68	LOAD-VAL4	DINT	L#0	Load counter 4 directly
72	LOAD-VAL5	DINT	L#0	Load counter 5 directly
76	LOAD-VAL6	DINT	L#0	Load counter 6 directly
80	LOAD-VAL7	DINT	L#0	Load counter 7 directly
84	LOAD-PREPARE-VAL0	DINT	L#0	Load counter 0 in preparation
88	LOAD-PREPARE-VAL1	DINT	L#0	Load counter 1 in preparation
92	LOAD-PREPARE-VAL2	DINT	L#0	Load counter 2 in preparation
96	LOAD-PREPARE-VAL3	DINT	L#0	Load counter 3 in preparation
100	LOAD-PREPARE-VAL4	DINT	L#0	Load counter 4 in preparation
104	LOAD-PREPARE-VAL5	DINT	L#0	Load counter 5 in preparation
108	LOAD-PREPARE-VAL6	DINT	L#0	Load counter 6 in preparation
112	LOAD-PREPARE-VAL7	DINT	L#0	Load counter 7 in preparation
116	CMP-VAL0	DINT	L#0	Load comparator 0
120	CMP-VAL1	DINT	L#0	Load comparator 1
124	CMP-VAL2	DINT	L#0	Load comparator 2
128	CMP-VAL3	DINT	L#0	Load comparator 3
132	CMP-VAL4	DINT	L#0	Load comparator 4
136	CMP-VAL5	DINT	L#0	Load comparator 5
140	CMP-VAL6	DINT	L#0	Load comparator 6
144	CMP-VAL7	DINT	L#0	Load comparator 7
148	ACT_CNTV0	DINT	L#0	Current count 0
152	ACT_MSRV0	DINT	L#0	Measurement result 0
156	ACT_CNTV1	DINT	L#0	Current count 1
160	ACT_MSRV1	DINT	L#0	Measurement result 1
164	ACT_CNTV2	DINT	L#0	Current count 2
168	ACT_MSRV2	DINT	L#0	Measurement result 2
172	ACT_CNTV3	DINT	L#0	Current count 3
176	ACT_MSRV3	DINT	L#0	Measurement result 3
180	ACT_CNTV4	DINT	L#0	Current count 4
184	ACT_MSRV4	DINT	L#0	Measurement result 4
188	ACT_CNTV5	DINT	L#0	Current count 5
192	ACT_MSRV5	DINT	L#0	Measurement result 5
196	ACT_CNTV6	DINT	L#0	Current count 6
200	ACT_MSRV6	DINT	L#0	Measurement result 6
204	ACT_CNTV7	DINT	L#0	Current count 7
208	ACT_MSRV7	DINT	L#0	Measurement result 7
212.0	BYTE0	BYTE	B#16#0	Reserved
213.0	BYTE1	BYTE	B#16#0	Reserved
214.0	BYTE2	BYTE	B#16#0	Reserved
215.0	BYTE3	BYTE	B#16#0	Reserved
216.0	BYTE4	BYTE	B#16#0	Channel type
217.0	BYTE5	BYTE	B#16#0	Length channel info
218.0	BYTE6	BYTE	B#16#0	Number of channels

Address	Tag	Data type	Initial value	Comment
219.0	BYTE7	BYTE	B#16#0	Channel error vector
220.0	BYTE8	BYTE	B#16#0	Error counter 0
221.0	BYTE9	BYTE	B#16#0	Error counter 1
222.0	BYTE10	BYTE	B#16#0	Error counter 2
223.0	BYTE11	BYTE	B#16#0	Error counter 3
224.0	BYTE12	BYTE	B#16#0	Error counter 4
225.0	BYTE13	BYTE	B#16#0	Error counter 5
226.0	BYTE14	BYTE	B#16#0	Error counter 6
227.0	BYTE15	BYTE	B#16#0	Error counter 7

CONTROL_SIGNALS

CONTROL_SIGNALS

Address	Parameters	Data type	Initial value	Description
21.0 … 7	CTRL_DQ0 … 7	BOOL	FALSE	Enable for digital outputs 0 ... 7
22.0 … 7	SET_DQ0 … 7	BOOL	FALSE	Trigger for digital outputs 0 ... 7
23.0 … 7	SW_GATE0 … 7	BOOL	FALSE	Software gate counters 0 ... 7

CHECKBACK_SIGNALS

CHECKBACK_SIGNALS

Address	Parameters	Data type	Initial value	Description
36.1	STS_TFB	BOOL	FALSE	TRUE: Programming device (PG) mode is active, control by CPU is not possible FALSE: Programming device (PG) mode deselected
36.4	DATA_ERR	BOOL	FALSE	TRUE: Data error occurred FALSE: No data error
36.7	PARA	BOOL	FALSE	TRUE: Counter module parameters are assigned, all other CHECKBACK_SIGNALS are valid FALSE: Counter module parameters not assigned
37.0 … 7	STS_CMP0 … 7	BOOL	FALSE	Status comparators 0 ... 7 You need to evaluate the status following the call of the CNT2_CTR instruction because the status is subsequently reset.
38.0 … 7	STS_UFLW0 … 7	BOOL	FALSE	Status of low limit violated counters 0 ... 7 You need to evaluate the status following the call of the CNT2_CTR instruction because the status is subsequently reset.

Address	Parameters	Data type	Initial value	Description
39.0 … 7	STS_OFLW0 … 7	BOOL	FALSE	Status of high limit violated counters 0 ... 7 You need to evaluate the status following the call of the CNT2_CTR instruction because the status is subsequently reset.
40.0 … 7	STS_DIR0 … 7	BOOL	FALSE	Count direction status of counters 0 ... 7 You need to evaluate the status following the call of the CNT2_CTR instruction because the status is subsequently reset.
41.0 … 7	STS_DI0 … 7	BOOL	FALSE	Status of digital inputs 0 ... 7
42.0 … 7	STS_DQ0 … 7	BOOL	FALSE	Status of digital outputs 0 ... 7
43.0 … 7	STS_GATE0 … 7	BOOL	FALSE	Internal gate status of counters 0 ... 7
44	USER_STAT_DWORD0	WORD	W#16#0	Count/measured value, depending on the parameter assignment
46	USER_STAT_DWORD1	WORD	W#16#0	Count/measured value, depending on the parameter assignment
48	USER_STAT_DWORD2	WORD	W#16#0	Count/measured value, depending on the parameter assignment
50	USER_STAT_DWORD3	WORD	W#16#0	Count/measured value, depending on the parameter assignment

Write job

Write command

Initiate a new write command by entering it in JOB_WR.NO. The following write jobs are permissible:

JOB_WR.NO (DBB0)	Entry in CNT2_CHANTYPE	Address in the counter DB	Meaning for the count mode	Meaning for the measuring mode
0	None	None	No write command / last write command finished	
10	LOAD_VAL0	52	Load counter 0 directly	Load low limit 0
11	LOAD_VAL1	56	Load counter 1 directly	Load low limit 1
12	LOAD_VAL2	60	Load counter 2 directly	Load low limit 2
13	LOAD_VAL3	64	Load counter 3 directly	Load low limit 3
14	LOAD_VAL4	68	Load counter 4 directly	Load low limit 4
15	LOAD_VAL5	72	Load counter 5 directly	Load low limit 5
16	LOAD_VAL6	76	Load counter 6 directly	Load low limit 6
17	LOAD_VAL7	80	Load counter 7 directly	Load low limit 7
20	LOAD_PREPARE_VAL0	84	Load counter 0 in preparation	Load high limit 0
21	LOAD_PREPARE_VAL1	88	Load counter 1 in preparation	Load high limit 1
22	LOAD_PREPARE_VAL2	92	Load counter 2 in preparation	Load high limit 2
23	LOAD_PREPARE_VAL3	96	Load counter 3 in preparation	Load high limit 3
24	LOAD_PREPARE_VAL4	100	Load counter 4 in preparation	Load high limit 4

JOB_WR.NO (DBB0)	Entry in CNT2_CHANTYPE	Address in the counter DB	Meaning for the count mode	Meaning for the measuring mode
25	LOAD_PREPARE_VAL5	104	Load counter 5 in preparation	Load high limit 5
26	LOAD_PREPARE_VAL6	108	Load counter 6 in preparation	Load high limit 6
27	LOAD_PREPARE_VAL7	112	Load counter 7 in preparation	Load high limit 7
30	CMP_VAL0	116	Load comparator 0	
31	CMP_VAL1	120	Load comparator 1	
32	CMP_VAL2	124	Load comparator 2	
33	CMP_VAL3	128	Load comparator 3	
34	CMP_VAL4	132	Load comparator 4	
35	CMP_VAL5	136	Load comparator 5	
36	CMP_VAL6	140	Load comparator 6	
37	CMP_VAL7	144	Load comparator 7	
40	LOAD_VAL0 to LOAD_VAL3	52 to 67	Load counters 0 to 3 directly	Load low limit 0 to 3
41	LOAD_VAL4 to LOAD_VAL7	68 to 83	Load counters 4 to 7 directly	Load low limit 4 to 7
42	LOAD_VAL0 to LOAD_VAL7	52 to 83	Load counters 0 to 7 directly	Load low limit 0 to 7
50	LOAD_PREPARE_VAL0 to LOAD_PREPARE_VAL3	84 to 99	Load counters 0 to 3 in preparation	Load high limit 0 to 3
51	LOAD_PREPARE_VAL4 to LOAD_PREPARE_VAL7	100 to 111	Load counters 4 to 7 in preparation	Load high limit 4 to 7
52	LOAD_PREPARE_VAL0 to LOAD_PREPARE_VAL7	84 to 111	Load counters 0 to 7 in preparation	Load high limit 0 to 7
60	CMP_VAL0 to CMP_VAL3	116 to 131	Load comparators 0 to 3	
61	CMP_VAL4 to CMP_VAL7	132 to 147	Load comparators 4 to 7	
62	CMP_VAL0 to CMP_VAL7	116 to 147	Load comparators 0 to 7	

Status of the write job

Status of a write job

The status of a write job is indicated in the counter DB (DBB1 data byte).

Bit in JOB_WR (DBX1.)	Meaning
.BUSY, 0	= 1: Write job active. The CNT2WRPN instruction sets this bit as soon as you execute a write job (JOB_WR.NO > 0 and JOB_WR.IMPOSS = 0). The CNT2WRPN instruction deletes this bit as soon as the write job is completed (JOB_WR.NO = 0).
.DONE, 1	= 1: Write job finished. The CNT2WRPN instruction sets this bit as soon as a write job is completed (regardless of errors). The CNT2WRPN instruction deletes the bit when a new write job begins. You can also delete this bit in the user program.

Bit in JOB_WR (DBX1.)	Meaning
.IMPOSS, 2	= 1: The write job cannot be processed (counter module parameters are not assigned, the call is active or programming device (PG) mode is active). You can leave the write job (JOB_WR) pending or delete it. The CNT2WRPN instruction deletes the bit if the conditions indicated above are fulfilled.
.UNKOWN, 3	= 1: Write job unknown. Your specified write job (JOB_WR) is not in the valid range (see the error message). The CNT2WRPN instruction deletes this bit as soon as a valid number is entered in JOB_WR. The unknown number is retained until this happens.

Error events are indicated in the binary result bit (BR = 0). Possible errors are:

- Unknown write job (see JOB_WR.UNKOWN).
- Error in data transmission with the WRREC instruction. Error information is available at the RET_VAL output parameter.
- The module checks for data errors in the transmitted data transferred and interprets the data. If a data error occurs, the CHECKBACK_SIGNALS.DATA_ERR bit is set to 1 in the counter DB.

Read job

Read job

Initiate a new read job by entering it in JOB_RD.NO. The following read jobs are permissible:

JOB_RD.NO (DBB2)	Entry in CNT2_CHANTYPE	Address in the counter DB	Meaning
0	None	None	No read job / last read job finished
100	ACT_CNTV0 ACT_MSRV0 ACT_CNTV1 ACT_MSRV1 ACT_CNTV2 ACT_MSRV2 ACT_CNTV3 ACT_MSRV3	148	Current counter value 0 to 3 and measurement result 0 to 3
101	ACT_CNTV4 ACT_MSRV4 ACT_CNTV5 ACT_MSRV5 ACT_CNTV6 ACT_MSRV6 ACT_CNTV7 ACT_MSRV7	180	Current counter value 4 to 7 and measurement result 4 to 7

Status of the write job

Status of a read job

The status of a read job is indicated in the counter DB (DBB3 data byte)

Bit in JOB_RD (DBX3.)	Description
.BUSY, 0	= 1: Read job active. The CNT2RDPN instruction sets this bit as soon as you execute a read job (JOB_RD.NO > 0 and JOB_RD.IMPOSS = 0). The CNT2RDPN instruction deletes this bit as soon as the read job is completed (JOB_RD.NO = 0).
.DONE, 1	= 1: Read job finished. The CNT2RDPN instruction sets this bit as soon as a read job is completed (regardless of errors). The CNT2RDPN instruction deletes the bit when a new read job begins. You can also delete this bit in the user program.
.IMPOSS, 2	= 1: The read job cannot be processed (counter module parameters are not assigned, the call is active or programming device (PG) mode is active). You can leave the read job (JOB_RD.NO) pending or delete it. The CNT2RDPN instruction deletes the bit if the conditions indicated above are fulfilled.
.UNKOWN, 3	= 1: Read job unknown. Your specified read job (JOB_RD) is not in the valid range (see Error evaluation). The CNT2RDPN instruction deletes this bit as soon as a valid number is entered in JOB_RD.NO. The unknown number is retained until this happens.

Error events are indicated in the binary result bit (BR = 0). Possible errors are:

- Unknown read job (see JOB_RD.UNKOWN).
- Error in data transmission with the RDREC instruction. Error information is available at the RET_VAL output parameter.

FM x51 Positioning

ABS_INIT

Description

The ABS_INIT instruction initializes the following data in the channel DB:

- The control signals
- The feedback signals
- The trigger bits, done bits and error bits of the jobs
- The function switches and their done bits and error bits
- Job management for the instruction ABS_CTRL (Page 7125) or ABS_CTRL_451 (Page 7129)

Call

The instruction must be executed for each channel after a startup, namely, after power-on of the module or CPU. For this reason, call the instruction in the restart organization block OB 100 and in pulling/plugging OB 83, for example or in the initialization phase of your user program. This ensures that your user program does not access obsolete data after a CPU or module restart.

Data block used

Channel DB (Page 7133):

The module address must be entered in the channel DB. You should enter it in the startup OB (OB 100).

Parameters

Parameters	Declaration	Data type	Description
DB_NO	INPUT	INT	Number of the channel DB

Return values

The instruction does not provide a return value.

ABS_CTRL

Description

Using the ABS_CTRL instruction, you can read the operational data of all module channels, assign the channel parameters and modify them in runtime. To complete such actions, use control signals, feedback signals, function switches as well as write and read commands.

Operating principle

Each time it is called, the instruction performs the following actions:

- Read feedback signals:
  The ABS_CTRL instruction reads all feedback signals of a channel and enters these in the channel OB. The control signals and commands are not processed unless this step is completed, which means that the feedback signals report the channel status given prior to the instruction call.
- Command management (Page 7142):
  The ABS_CTRL instruction processes the read and write commands and transfers the data between the channel DB, parameter DB and the module.
- Write control signals:
  The control signals that are entered in the channel DB are transferred to the module.

Call

The ABS_CTRL instruction must be called cyclically for each channel, for example, in OB 1.

Before calling the ABS_CTRL instruction, enter all data necessary to execute the selected functions in the channel DB.

Data blocks used

- Channel DB (Page 7133):
  The channel DB is the instance DB of the instruction ABS_CTRL. The module address and the channel number must be entered in the channel DB. Incorrect information could result in I/O access errors or in an access to a different module, which in turn gives rise to data corruption.
- Parameter DB (Page 7139):
  If you want to use commands to write or read the machine data, you need a parameter DB, whose number must be entered in the channel DB.

Return values

The instruction provides the following return values:

RETVAL	BR	Description
1	1	At least 1 command is active
0	1	No active command, no errors
-1	0	Errors: Data error (DATA_ERR) or communication error (JOB_ERR)

Control signals

If a STOP signal is set or an operator error is pending or a drive enable is missing, the instruction resets the control signals START, DIR_M and DIR_P.

To enable the restart of a trip, acknowledge the operator error with OT_ERR_A=1. With this acknowledgement you cannot submit any other commands and control signals.

If there is no operator error pending, the instruction sets the acknowledgment status for the operator error OT_ERR_A to 0.

When the channel reports the start of the trip, the instruction sets and resets the start signals START, DIR_P and DIR_M, except in "jog" mode.

If the axis are not configured, the instruction puts all the control signals on hold, except the operator error acknowledgment OT_ERR_A.

Commands and control signals

You can issue several commands at the same time, also together with the control signals necessary for the positioning. If at least one write command is initiated at the same time as the control signals START, DIR_M or DIR_P, the instruction puts these control signals on hold until the write commands have been executed.

Startup

Call instructionABS_INIT (Page 7124) at the startup of the module or CPU. This action includes a reset of the function switches. The ABS_CTRL instruction acknowledges the module startup. During this time, RETVAL and JOBBUSY = 1.

Reaction to error

If incorrect data is found in a write command, the channel returns feedback signal DATA_ERR = 1 in the channel DB. If an error is found in a write or read command during communication with the module, the cause of error is saved to the JOB_ERR parameter in the channel DB.

- Error with a write command:
  The trigger bit is reset and the error bit _ERR and done bit _D are set if the command is corrupt. For all other pending write commands, the trigger bit is also reset and only the error bit _ERR is set. The awaiting write commands will be withdrawn because commands could stack up on top of each other.
  The execution of the pending read commands is continued. JOB_ERR is retriggered accordingly for all commands.
- Error with a read command:
  The trigger bit is reset and the error bit _ERR and done bit _D are set if the command is corrupt.
  The execution of the pending read commands is continued. JOB_ERR is retriggered accordingly for all commands.

ABS_DIAG

Description

Use the ABS_DIAG instruction to read the data from the diagnostics buffer of the module and make it available for display on the HMI system or for programmed evaluation.

Operating principle

The instruction reads the data from the diagnostic buffer when a new entry is displayed in the diagnostics buffer by feedback signal DIAG = 1 in the channel DB. After reading the data from the diagnostics buffer, the module sets the DIAG bit in the channel DB to 0.

Call

The instruction must be called cyclically, for example, in OB 1. Additional calls in an interrupt OB are not allowed. To complete the execution of the instruction, you must call it at least twice (in 2 cycles).

Data block used

Diagnostics DB (Page 7141):

The diagnostics DB is the instance DB of the instruction ABS_DIAG. The module address must be entered in the diagnostics DB. The most recent entry in the diagnostics buffer is written to the DIAG[1] structure and the oldest entry to the DIAG[9] structure.

Return values

The instruction provides the following return values in the RETVAL parameter of the diagnostics DB:

RETVAL	BR	Description
1	1	Command active
0	1	No active command, no errors
-1	0	Errors

Commands

You can read the diagnostic buffer, independent of any new entry, by setting trigger bit DIAGRD_EN in the diagnostics DB. After the diagnostics buffer has been read, the trigger bit is set to 0.

Execute this command after the startup of a CPU and module is completed. This step ensures consistency between the content of the diagnostics DB and the diagnostics buffer of the module, even if the module has not made a new entry in the diagnostics buffer.

Startup

The instruction does not run a startup routine.

Reaction to error

Information about the cause of an execution error is available in the JOB_ERR parameter of the diagnostics DB.

Diagnostics data

Errors	Evaluation using OB 82
Module fault	OB82_MDL_DEFECT
Internal error	OB82_INT_FAULT
External error	OB82_EXT_FAULT
Channel error	OB82_PNT_INFO
External auxiliary voltage is missing	OB82_EXT_VOLTAGE
Watchdog timeout	OB82_WTCH_DOG_FLT
Failure of the internal internal supply voltage of the module	OB82_INT_PS_FLT
Hardware interrupt lost	OB82_HW_INTR_FLT

Error classes

The error events are organized in the following error classes:

- Error class 1: Operational errors (Page 7147)
- Error class 2: Operator errors (Page 7148)
- Error class 4: Data errors (Page 7149)
- Error class 5: Machine data errors (Page 7150)
- Error class 6: Incremental dimension table errors (Page 7153)
- Error class 15: Messages (Page 7153)
- Error class 128: Diagnostics errors (Page 7154)

ABS_CTRL_451

Description

Using the ABS_CTRL_451 instruction, you can read the operational data of all module channels, assign the channel parameters and change them in runtime. To complete such actions, use control signals, feedback signals, function switches as well as write and read jobs.

Operating principle

Each time it is called, the instruction performs the following actions:

- Read feedback signals:
  The ABS_CTRL_451 instruction reads all feedback signals of a channel and enters these in the channel OB. The control signals and jobs are not processed unless this step is completed, which means that the feedback signals report the channel status given prior to the instruction call.
- Job management (Page 7144):
  The ABS_CTRL_451 instruction processes the read and write jobs and transfers the data between the channel DB, parameter DB and the module.
- Write control signals:
  The control signals that are entered in the channel DB are transferred to the module.

Call

The ABS_CTRL_451 instruction must be called cyclically for each channel, for example, in OB 1.

Before calling the ABS_CTRL_451 instruction, enter all data necessary to execute the selected functions in the channel DB.

Data blocks used

- Channel DB (Page 7133):
  The module address and channel number must be entered in the channel DB. Incorrect information can result in I/O access errors or in an access to a different module, which in turn gives rise to data corruption.
- Parameter DB (Page 7139):
  If you want to use jobs to write or read machine data, you need a parameter DB, the number of which must be entered in the channel DB.

Parameters

Parameters	Declaration	Data type	Description
DB_NO	INPUT	INT	Number of the channel DB
RET_VAL	OUTPUT	INT	Return value

Return values

The instruction provides the following return values:

RET_VAL	BR	Description
1	1	At least 1 job is active
0	1	No active job, no errors
-1	0	Errors: Data error (DATA_ERR) or communication error (JOB_ERR)

Control signals

If a STOP signal is set or an operator error is pending or a drive enable is missing, the instruction resets the control signals START, DIR_M and DIR_P.

To enable the restart of a trip, acknowledge the operator error with OT_ERR_A=1. With this acknowledgement you cannot submit any other jobs and control signals.

If there is no operator error pending, the instruction sets the acknowledgment status for the operator error OT_ERR_A to 0.

When the channel reports the start of the trip, the instruction sets and resets the start signals START, DIR_P and DIR_M, except in "jog" mode..

If the axis are not configured, the instruction puts all the control signals on hold, except the operator error acknowledgment OT_ERR_A.

Jobs and control signals

You can transfer several jobs simultaneously, also together with the control signals necessary for the positioning. If at least one write job is initiated at the same time as the control signals START, DIR_M or DIR_P, the instruction puts these control signals on hold until the write jobs have been executed.

Startup

Call the ABS_INIT (Page 7124) instruction at the startup of the module or CPU. This action includes a reset the function switches. The ABS_CTRL_451 instruction acknowledges the module startup. During this time, RETVAL and JOBBUSY = 1.

Reaction to errors

If incorrect data is found in a write job, the channel returns feedback signal DATA_ERR = 1 in the channel DB. If an error is found in a write or read job during communication with the module, the cause of error is saved to the JOB_ERR parameter in the channel DB.

- Write job errors:
  The trigger bit is reset and the error bit _ERR and done bit _D are set if the job is corrupt. For all other pending write jobs, the trigger bit is also reset and only the error bit _ERR is set. The other pending write jobs are revoked to prevent jobs from overlapping.
  The execution of the pending read jobs is continued. JOB_ERR is retriggered accordingly for all jobs.
- Read job errors:
  The trigger bit is reset and the error bit _ERR and done bit _D are set if the job is corrupt. The execution of the pending read jobs is continued. JOB_ERR is retriggered accordingly for all jobs.

ABS_DIAG_451

Description

Use the ABS_DIAG_451 instruction to read the data from the diagnostics buffer of the module and make it available for display on the HMI system or for programmed evaluation.

Operating principle

The instruction reads the data from the diagnostic buffer when a new entry is displayed in the diagnostics buffer by feedback signal DIAG = 1 in the channel DB. After reading the data from the diagnostics buffer, the module sets the DIAG bit in the channel DB to 0.

Call

The instruction must be called cyclically, for example, in OB 1. Additional calls in an interrupt OB are not allowed. To complete the execution of the instruction, you must call it at least twice (in 2 cycles).

Data block used

Diagnostics DB (Page 7141):

The module address must be entered in the diagnostics DB. The most recent entry in the diagnostics buffer is written to the DIAG[1] structure and the oldest entry to the DIAG[9] structure.

Parameters

Parameters	Declaration	Data type	Description
DB_NO	INPUT	INT	Number of the diagnostics DB
RET_VAL	OUTPUT	INT	Return value

Return values

The instruction provides the following return values in the RET_VAL parameter of the diagnostics DB:

RET_VAL	BR	Description
1	1	Job active
0	1	No active job, no errors
-1	0	Errors

Jobs

You can read the diagnostic buffer, independent of any new entry, by setting trigger bit DIAGRD_EN in the diagnostics DB. After the diagnostics buffer has been read, the trigger bit is set to 0.

Execute this job after the startup of a CPU and module is completed. This step ensures consistency between the content of the diagnostics DB and the diagnostics buffer of the module, even if the module has not made a new entry in the diagnostics buffer.

Startup

The instruction does not run a startup routine.

Reaction to errors

Information about the cause of an execution error is available in the JOB_ERR parameter of the diagnostics DB.

Diagnostics data

Errors	Evaluation using OB 82
Module fault	OB82_MDL_DEFECT
Internal error	OB82_INT_FAULT
External error	OB82_EXT_FAULT
Channel error	OB82_PNT_INFO
External auxiliary voltage is missing	OB82_EXT_VOLTAGE
Front connector missing	OB82_FLD_CONNCTR
Watchdog timeout	OB82_WTCH_DOG_FLT

Errors	Evaluation using OB 82
Failure of the internal internal supply voltage of the module	OB82_INT_PS_FLT
Hardware interrupt lost	OB82_HW_INTR_FLT

Error classes

The error events are organized in the following error classes:

- Error class 1: Operating error (Page 7147)
- Error class 2: Operator errors (Page 7148)
- Error class 4: Data errors (Page 7149)
- Error class 5: Machine data errors (Page 7150)
- Error class 6: Incremental dimension table errors (Page 7153)
- Error class 15: Messages (Page 7153)
- Error class 128: Diagnostics errors (Page 7154)

Additional references for FM x51 ABS V1

Channel data block

Purpose

The channel DB is the data interface between the user program and FM x51. It contains and stores all data necessary to control and operate a channel.

Configuration

The channel DB is divided into different areas:

Channel DB
Addresses *)
Control signals
Feedback signals
Function switches
Trigger bits for write jobs
Trigger bits for read jobs
Done bits
Error bits
Job management for instructions
Data for jobs
*) You can enter the module address in the parameter assignment interface.

Content

The table below shows the content of the channel DB.

Note

You may not change any data not listed in this table.

Address	Parameters	Data type	Initial value	Description
Addresses				
0.0	MOD_ADDR	INT	0	Module address
2.0	CH_NO	INT	1	Channel number
10.0	PARADBNO	INT	-1	Number of the parameter DB
Control signals				
14.3	OT_ERR_A	BOOL	FALSE	1 = acknowledge operator error
15.0	START	BOOL	FALSE	1 = start positioning
15.1	STOP	BOOL	FALSE	1 = stop active traversing
15.2	DIR_M	BOOL	FALSE	1 = minus direction
15.3	DIR_P	BOOL	FALSE	1 = plus direction
15.6	SPEED252	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: Start velocity for incremental approach with increment number 252: – 0 = Creep speed – 1 = Rapid traverse
15.7	DRV_EN	BOOL	FALSE	1 = switch on drive enable
16.0	MODE_IN	BYTE	B#16#0	Required mode • 0 = no mode • 1 = Jog • 3 = reference point approach • 4 = relative incremental approach • 5 = absolute incremental approach
17.0	MODE_TYPE	BYTE	B#16#0	• Initial speed for "jog" mode – 0 = Creep speed – 1 = Rapid traverse • Increment number for incremental approach mode
Feedback signals				
22.2	DIAG	BOOL	FALSE	1 = diagnostic buffer modified
22.3	OT_ERR	BOOL	FALSE	1 = operator error occurred
22.4	DATA_ERR	BOOL	FALSE	1 = data error
22.7	PARA	BOOL	FALSE	1 = axis parameters are assigned
23.0	ST_ENBLD	BOOL	FALSE	1 = start enabled
23.1	WORKING	BOOL	FALSE	1 = positioning is active
23.2	WAIT_EI	BOOL	FALSE	1 = axis waiting for external enable
23.4	SPEED_OUT	BOOL	FALSE	• 0 = Creep speed • 1 = Rapid traverse

Address	Parameters	Data type	Initial value	Description
23.5	ZSPEED	BOOL	FALSE	1 = axis is located in the standstill range
23.6	CUTOFF	BOOL	FALSE	1 = axis is located in the switch-off range
23.7	CHGOVER	BOOL	FALSE	1 = axis is located in the changeover range
24.0	MODE_OUT	BYTE	B#16#0	Active operating mode
25.0	SYNC	BOOL	FALSE	1 = axis is synchronized
25.1	MSR_DONE	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: 1 = Length measurement / edge detection completed
25.2	GO_M	BOOL	FALSE	1 = axis moves in minus direction
25.3	GO_P	BOOL	FALSE	1 = axis moves in plus direction
25.5	FVAL_DONE	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: 1 = Set actual value on-the-fly completed
25.7	POS_RCD	BOOL	FALSE	1 = In position
26.0	ACT_POS	DINT	L#0	Current actual value (current position of the axis)
Function switch				
34.0	PLOOP_ON	BOOL	FALSE	1 = Loop approach in plus direction
34.1	MLOOP_ON	BOOL	FALSE	1 = Loop approach in minus direction
34.2	EI_OFF	BOOL	FALSE	1 = Do not evaluate enable input
34.3	EDGE_ON	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: 1 = Edge detection on
34.4	MSR_ON	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: 1 = Length measurement on
Trigger bits for write jobs				
35.0	MDWR_EN	BOOL	FALSE	1 = write machine data
35.1	MD_EN	BOOL	FALSE	1 = Enable machine data
35.2	DELDIST_EN	BOOL	FALSE	1 = Delete remaining distance
35.3	AVALREM_EN	BOOL	FALSE	1 = Cancel set actual value
35.4	TRGL1WR_EN	BOOL	FALSE	1 = Write incremental dimension table 1 (increment number 1 ... 50)
35.5	TRGL2WR_EN	BOOL	FALSE	1 = Write incremental dimension table 2 (increment number 51 ... 100)
35.6	REFPT_EN	BOOL	FALSE	1 = set reference point
35.7	AVAL_EN	BOOL	FALSE	1 = Set actual value
36.0	FVAL_EN	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: 1 = Set actual value on-the-fly
36.1	ZOFF_EN	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: 1 = Set zero offset
36.2	TRG252_254_EN	BOOL	FALSE	1 = Write increment for increment number 252 or 254
36.3	TRG255_EN	BOOL	FALSE	1 = write increment for increment number 255

Address	Parameters	Data type	Initial value	Description
36.4	DELDIAG_EN	BOOL	FALSE	1 = delete diagnostic buffer
Activation bits for read jobs				
36.5	MDRD_EN	BOOL	FALSE	1 = Reading machine data
36.6	TRGL1RD_EN	BOOL	FALSE	1 = Read incremental dimension table 1 (increment number 1 ... 50)
36.7	TRGL2RD_EN	BOOL	FALSE	1 = Read incremental dimension table 2 (increment number 51 ... 100)
37.0	MSRRD_EN	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: 1 = Read measured values
37.1	ACTSPD_EN	BOOL	FALSE	1 = Read actual speed, remaining distance and current increment
37.2	ENCVAL_EN	BOOL	FALSE	1 = read encoder values
Done bits for function switches				
38.0	PLOOP_D	BOOL	FALSE	1 = "loop approach in plus direction" job completed
38.1	MLOOP_D	BOOL	FALSE	1 = "loop approach in minus direction" job completed
38.2	EI_D	BOOL	FALSE	1 = "Do not evaluate enable input" job completed
38.3	EDGE_D	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: 1 = "Edge detection ON" job completed
38.4	MSR_D	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: 1 = "Length measurement ON" job completed
Done bits for write jobs				
39.0	MDWR_D	BOOL	FALSE	1 = "Write machine data" job completed
39.1	MD_D	BOOL	FALSE	1 = "Enable machine data" job completed
39.2	DELDIST_D	BOOL	FALSE	1 = "Delete remaining distance" job completed
39.3	AVALREM_D	BOOL	FALSE	1 = "Cancel set actual value" job completed
39.4	TRGL1WR_D	BOOL	FALSE	1 = "Write incremental dimension table 1" job completed
39.5	TRGL2WR_D	BOOL	FALSE	1 = "Write incremental dimension table 2" job completed
39.6	REFPT_D	BOOL	FALSE	1 = "Set reference point" job completed
39.7	AVAL_D	BOOL	FALSE	1 = "Set actual value" job completed
40.0	FVAL_D	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: 1 = "Set actual value on-the-fly" job completed
40.1	ZOFF_D	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: 1 = "Set zero offset" job completed
40.2	TRG252_254_D	BOOL	FALSE	1 = "Write increment for increment number 252 or 254" job completed
40.3	TRG255_D	BOOL	FALSE	1 = "Write increment for increment number 255" job completed
40.4	DELDIAG_D	BOOL	FALSE	1 = "Delete diagnostic buffer" job completed

Address	Parameters	Data type	Initial value	Description
Done bits for read jobs				
40.5	MDRD_D	BOOL	FALSE	1 = "Read machine data" job completed
40.6	TRGL1RD_D	BOOL	FALSE	1 = "Read incremental dimension table 1" job completed
40.7	TRGL2RD_D	BOOL	FALSE	1 = "Read incremental dimension table 2" job completed
41.0	MSRRD_D	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: 1 = "Read measured values" job completed
41.1	ACTSPD_D	BOOL	FALSE	1 = "Read actual speed, remaining distance and current increment" job completed
41.2	ENCVAL_D	BOOL	FALSE	1 = "Read encoder values" job completed
Error bits for function switches				
42.0	PLOOP_ERR	BOOL	FALSE	1 = Error with "loop approach in plus direction" job
42.1	MLOOP_ERR	BOOL	FALSE	1 = Error with "loop approach in minus direction" job
42.2	EI_ERR	BOOL	FALSE	1 = Error with "do not evaluate enable input" job
42.3	EDGE_ERR	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: 1 = Error during "Edge detection ON" job
42.4	MSR_ERR	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: 1 = Error during "Length measurement ON" job
Error bits for write jobs				
43.0	MDWR_ERR	BOOL	FALSE	1 = Error with "Write machine data" job
43.1	MD_ERR	BOOL	FALSE	1 = Error with "Activate machine data" job
43.2	DELDIST_ERR	BOOL	FALSE	1 = Error with "Delete remaining distance" job
43.3	AVALREM_ERR	BOOL	FALSE	1 = Error with "Cancel set actual value" job
43.4	TRGL1WR_ERR	BOOL	FALSE	1 = Error with "Write incremental dimension table 1" job
43.5	TRGL2WR_ERR	BOOL	FALSE	1 = Error with "Write incremental dimension table 2" job
43.6	REFPT_ERR	BOOL	FALSE	1 = Error with "set reference point" job
43.7	AVAL_ERR	BOOL	FALSE	1 = Error with "set actual value" job
44.0	FVAL_ERR	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: 1 = Error in "set actual value on-the-fly" job
44.1	ZOFF_ERR	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: 1 = Error in "set zero offset" job
44.2	TRG252_254_ERR	BOOL	FALSE	1 = Error with "Write increment for increment number 252 or 254" job
44.3	TRG255_ERR	BOOL	FALSE	1 = Error with "Write increment for incremental dimension table 255" job
44.4	DELDIAG_ERR	BOOL	FALSE	1 = Error with "Delete diagnostic buffer"
Error bits for read jobs				

Address	Parameters	Data type	Initial value	Description
44.5	MDRD_ERR	BOOL	FALSE	1 = Error with "read machine data" job
44.6	TRGL1RD_ERR	BOOL	FALSE	1 = Error with "Read incremental dimension table 1" job
44.7	TRGL2RD_ERR	BOOL	FALSE	1 = Error with "Read incremental dimension table 2" job
45.0	MSRRD_ERR	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: 1 = Error with "Read measured values" job
45.1	ACTSPD_ERR	BOOL	FALSE	1 = Error with "Read actual speed, remaining distance and current increment" job
45.2	ENCVAL_ERR	BOOL	FALSE	1 = Error with "Read current encoder values" job
Job management for the instruction ABS_CTRL or ABS_CTRL_451				
48.0	JOB_ERR	INT	0	Error number of the communication error
50.0	JOBBUSY	BOOL	FALSE	1 = At least 1 job active
50.1	JOBRESET	BOOL	FALSE	1 = Reset all error bits and done bits
Data for "zero offset" job (FM 451)				
80.0	ZOFF	DINT	L#0	• FM 351: 0 (not used) • FM 451: Zero offset
Data for "Set actual value" job				
84.0	AVAL	DINT	L#0	Coordinate for "Set actual value"
Data for "Set actual value on-the-fly" (FM 451)				
88.0	FVAL	DINT	L#0	• FM 351: 0 (not used) • FM 451: Coordinate for "Set actual value on-the-fly"
Data for "Set reference point" job				
92.0	REFPT	DINT	L#0	Coordinate for "Set reference point"
Data for "Write increment for increment number 252 or 254" job				
96.0	TRG252_254	DINT	L#0	Write increment for increment number 252 or 254
Data for "Write increment for increment number 255" job				
100.0	TRG255	DINT	L#0	Increment for increment number 255
104.0	CHGDIF255	DINT	L#0	Changeover difference for increment number 255
108.0	CUTDIF255	DINT	L#0	Switch-off difference for increment number 255
Data for "Read position data" job				
112.0	ACTSPD	DINT	L#0	Current speed
116.0	DIST_TO_GO	DINT	L#0	Remaining distance
120.0	ACT_TRG	DINT	L#0	Current increment
Data for "Read encoder data" job				
124.0	ENCVAL	DINT	L#0	Actual encoder value (internal representation)
128.0	ZEROVAL	DINT	L#0	Last zero mark value (internal representation)
132.0	ENC_ADJ	DINT	L#0	Absolute encoder adjustment
Data for the "Length measurements / edge detection" job (FM 451)				
136.0	BEG_VAL	DINT	L#0	• FM 351: 0 (not used) • FM 451: Start value of length measurement / edge detection

Address	Parameters	Data type	Initial value	Description
140.0	END_VAL	DINT	L#0	• FM 351: 0 (not used) • FM 451: End value of length measurement / edge detection
144.0	LEN_VAL	DINT	L#0	• FM 351: 0 (not used) • FM 451: Length

Parameter data block

Purpose

In order to change machine data and incremental dimension tables in runtime, you need a parameter DB in which this data is stored. The parameters can be changed from the user program or from an HMI system.

You can export data displayed in the parameter assignment interface to a parameter DB. You can also import a parameter DB into the parameter assignment interface and view it there.

Each module channel can have several sets of parameter assignment data, for example, for different recipes. You can switch among these in your program.

Configuration

Parameter DB
Machine data
incremental dimension tables

Content

The table below shows the content of the parameter DB.

Note

You may not change any data not listed in this table.

Address	Parameters	Data type	Initial value	Description
Machine data				
4.0	EDGEDIST	DINT	L#0	• FM 351: 0 (not used) • FM 451: Minimum edge interval
8.0	UNITS	DINT	L#1	System of units
12.0	AXIS_TYPE	DINT	L#0	0 = Linear axis, 1 = Rotary axis
16.0	ENDROTAX	DINT	L#100 000	End of rotary axis
20.0	ENC_TYPE	DINT	L#1	Encoder type, message length
24.0	DISP_REV	DINT	L#80 000	Length per encoder revolution
32.0	INC_REV	DINT	L#500	Increments per encoder revolution

Address	Parameters	Data type	Initial value	Description
36.0	NO_REV	DINT	L#1	Number of encoder revolutions
40.0	BAUDRATE	DINT	L#0	Transmission rate
44.0	REFPT	DINT	L#0	Reference point coordinate
48.0	ENC_ADJ	DINT	L#0	Absolute encoder adjustment
52.0	REFPT_TYPE	DINT	L#0	Type of reference point approach
59.0	CNT_DIR	BOOL	FALSE	Counting direction: • 0 = Normal • 1 = Inverted
63.0	MON_WIRE	BOOL	TRUE	1 = wire break monitoring
63.1	MON_FRAME	BOOL	TRUE	1 = frame error monitoring
63.2	MON_PULSE	BOOL	TRUE	1 = missing pulses monitoring
64.0	SSW_STRT	DINT	L#-100 000 000	Software limit switch start
68.0	SSW_END	DINT	L#100 000 000	Software limit switch end
76.0	TRG_RANGE	DINT	L#1000	Target range
80.0	MON_TIME	DINT	L#2000	Monitoring time [ms]
84.0	ZSPEED_R	DINT	L#1000	Standstill range
88.0	ZSPEED_L	DINT	L#30 000	High limit of the standstill speed
92.0	CTRL_TYPE	DINT	L#1	Control type (1 - 4)
99.0	REFPT_SPD	BOOL	TRUE	Start speed in the case of a reference point approach: • 0 = Rapid traverse • 1 = Creep speed
99.1	EI_TYPE	BOOL	FALSE	• FM 351: 0 (not used) • FM 451: Enable input: – 0 = Level-controlled – 1 = Edge controlled
100.0	CHGDIF_P	DINT	L#5000	Changeover difference plus
104.0	CHGDIF_M	DINT	L#5000	Changeover difference minus
108.0	CUTDIF_P	DINT	L#2000	Switch-off difference plus
112.0	CUTDIF_M	DINT	L#2000	Switch-off difference minus
Increment dimension table 1				
120.0	TRGL1.TRG[1]	DINT	L#0	Increment number 1
. . .	. . .	. . .	. . .	. . .
316.0	TRGL1.TRG[50]	DINT	L#0	Increment number 50
Increment dimension table 2				
320.0	TRGL2.TRG[51]	DINT	L#0	Increment number 51
. . .	. . .	. . .	. . .	. . .
516.0	TRGL2.TRG[100]	DINT	L#0	Increment number 100

Diagnostics data block

Purpose

The diagnostics DB is the storage location for the instruction ABS_DIAG (Page 7127) or ABS_DIAG_451 (Page 7131). It contains the module's diagnostics buffer which is processed by the instruction.

Configuration

Diagnostics DB
Module address
Internal data
Job status
Trigger bit
Processed diagnostics buffer

Content

The table below shows the content of the diagnostics DB.

Note

You may not change any data not listed in this table.

Address	Parameters	Data type	Initial value	Description
0.0	MOD_ADDR	INT	0	Module address
256.0	JOB_ERR	INT	0	Error number of the communication error
258.0	JOBBUSY	BOOL	FALSE	1 = Job active
258.1	DIAGRD_EN	BOOL	FALSE	1 = definitely read diagnostics buffer
260.0	DIAG_CNT	INT	0	Number of valid entries in the list
262.0	DIAG[1]	STRUCT		Diagnostics data - latest entry
272.0	DIAG[2]	STRUCT		Diagnostics data - second entry
282.0	DIAG[3]	STRUCT		Diagnostics data - third entry
292.0	DIAG[4]	STRUCT		Diagnostics data - fourth entry
302.0	DIAG[5]	STRUCT		Diagnostics data - fifth entry
312.0	DIAG[6]	STRUCT		Diagnostics data - sixth entry
322.0	DIAG[7]	STRUCT		Diagnostics data - seventh entry
332.0	DIAG[8]	STRUCT		Diagnostics data - eighth entry
342.0	DIAG[9]	STRUCT		Diagnostics data - ninth entry

Structure of the diagnostics entry

The table below shows the structure of a diagnostics entry DIAG[n].

Address	Parameters	Data type	Initial value	Description
+0.0	STATE	BOOL	FALSE	0 = incoming event 1 = outgoing event
+0.1	INTF	BOOL	FALSE	1 = internal error
+0.2	EXTF	BOOL	FALSE	1 = external error
+2.0	FCL	INT	0	Error class: 1: Operating error 2: Operating errors 4: Data error 5: Machine data error 6: Increment dimension table error 15: Messages 128: Diagnostics errors
+4.0	FNO	INT	0	Error number
+6.0	CH_NO	INT	0	Channel number
+8.0	TRG_NO	INT	0	Increment number

Job management (ABS_CTRL)

Jobs

Data communication with the module other than the transfer of control and feedback signals is handled with jobs.

In order to initiate a job, set the corresponding trigger bit in the channel DB, including the specific data for write jobs. To execute the job, call the ABS_CTRL (Page 7125) instruction.

If you use the FM 351 centrally, a read job needs precisely one cycle. If you use the FM 351 in a distributed configuration, a read job may need several cycles.

Due to the required acknowledgments of the module, at least 3 calls or OB cycles are required to execute write jobs.

Once a job has been completed, the instruction resets the trigger bit. The next job pending is identified and executed at the next call of the instruction.

In addition to the trigger bit with the _EN extension for "enable", each job is provided a done bit and an error bit. Their names have the extension _D for "done" or _ERR for "error". When the job is completed, the ABS_CTRL instruction updates the done bits and error bits. Following evaluation or prior to the initiation of a job, you should set these bits to 0.

When you set the JOBRESET bit, all done and error bits are reset prior to executing jobs pending. The JOBRESET bit is then reset to 0.

Function switches

The function switches toggle the channel states on and off. A job for writing the function switches is only executed when there is a change to a switch setting. The setting of the function switch is retained after the job has been executed.

Function switches and jobs can be used concurrently at the call of the ABS_CTRL instruction.

Similar to jobs, the function switches are provided bits with name extension _ON/_OFF, done bits with name extension _D and error bits with name extension _ERR.

You should set these bits to 0 before you initiate a job to change a function switch, in order to enable the evaluation of the done bits and error bits of the function switches.

Order of job execution

You can initiate several jobs in a single operation. If no jobs are active, the job management of the ABS_CTRL instruction performs a scan to detect set trigger bits or changes to function switches, starting with the MDWR_EN job. Any job found are executed. When the job is completed, the job management searches for the next job to be executed. After having scanned the last ENCVAL_EN job, the function restarts the scan at the MDWR_EN job. This scan is repeated until all jobs have been executed.

The jobs are processed in the following technologically appropriate order:

Order	Address in the channel DB	Parameters	Description	Reset by
Write jobs				
1	35.0	MDWR_EN	Write machine data	Instruction
2	35.1 35.2 35.3 36.4	MD_EN DELDIST_EN AVALREM_EN DELDIAG_EN	Enable machine data Delete remaining distance Cancel set actual value Delete diagnostics buffer	Instruction
3	35.4	TRGL1WR_EN	Write incremental dimension table 1	Instruction
4	35.5	TRGL2WR_EN	Write incremental dimension table 2	Instruction
5	35.6	REFPT_EN	Set reference point	Instruction
6	 34.0 34.1 34.2	Function switch: PLOOP_ON MLOOP_ON EI_OFF	 Loop approach in plus direction Loop approach in minus direction Do not evaluate enable input	User program
7	35.7	AVAL_EN	Set actual value	Instruction
10	36.2	TRG252_254_EN	Write increment for increment number 254	Instruction
11	36.3	TRG255_EN	Write increment for increment number 255	Instruction
Read jobs				
12	36.5	MDRD_EN	Reading machine data	Instruction
13	36.6	TRGL1RD_EN	Read incremental dimension table 1	Instruction
14	36.7	TRGL2RD_EN	Read incremental dimension table 2	Instruction
16	37.1	ACTSPD_EN	Read current velocity, remaining distance and increment	Instruction
17	37.2	ENCVAL_EN	Read encoder data	Instruction

This order enables you to completely trigger a positioning motion using a set of jobs and control signals. The jobs encompass the writing and activation of the machine data, the setup of the external enable input and the writing of increments for the incremental approaches.

Jobs during active positioning

If initiated while a positioning job is active, the write jobs listed in the following table are held until the current positioning job is completed and are then executed at the next instruction call.

Address	Parameters	Data type	Initial value	Description
34.0	PLOOP_ON	BOOL	FALSE	1 = Loop approach in plus direction
34.1	MLOOP_ON	BOOL	FALSE	1 = Loop approach in minus direction
34.2	EI_OFF	BOOL	FALSE	1 = Do not evaluate enable input
35.1	MD_EN	BOOL	FALSE	1 = Enable machine data
35.2	DELDIST_EN	BOOL	FALSE	1 = Delete remaining distance
35.3	AVALREM_EN	BOOL	FALSE	1 = Cancel set actual value
35.6	REFPT_EN	BOOL	FALSE	1 = Set reference point coordinate
35.7	AVAL_EN	BOOL	FALSE	1 = Set actual value
36.4	DELDIAG_EN	BOOL	FALSE	1 = Delete diagnostics buffer

Job status

The status of job execution can be read from the return value RETVAL and activity bit JOBBUSY in the channel DB. The status of an individual job can be evaluated based on its trigger bits, done bits and error bits.

	RETVAL	JOBBUSY	Trigger bit _EN	Done bit _D	Error bit _ERR
Job active	1	1	1	0	0
Job completed without errors	0	0	0	1	0
Job completed with errors	-1	0	0	1	1
Write job cancelled	-1	0	0	0	1

Job management (ABS_CTRL_451)

Jobs

Data communication with the module other than the control and feedback signals is handled using jobs.

To initiate a job, set the corresponding trigger bit in the channel DB and the corresponding data for write jobs. Then call the ABS_CTRL_451 (Page 7129) instruction to execute the job.

Due to the required acknowledgments of the module, at least 3 calls or OB cycles are required to execute write jobs. A read job is executed immediately.

Once a job has been completed, the instruction resets the trigger bit. The next job pending is identified and executed at the next call of the instruction.

In addition to the trigger bit with the _EN extension for "enable", each job is provided a done bit and an error bit. Their names have the extension _D for "done" or _ERR for "error". When the job is completed, the ABS_CTRL_451 instruction updates the done bits and error bits. After the evaluation or prior to issuing a job, these bits should be set to 0.

When you set the JOBRESET bit, all done and error bits are reset before pending jobs are executed. The JOBRESET bit is then reset to 0.

Function switches

The function switches switch the states of the channel on and off. A job for writing the function switches is only executed when there is a change to a switch setting. The setting of the function switch is retained after the job has been executed.

Function switches and jobs can be used concurrently at the call of ABS_CTRL_451.

Similar to jobs, the function switches are provided bits with name extension _ON/_OFF, done bits with name extension _D and error bits with name extension _ERR.

To be able to evaluate the done bits and error bits of the function switches, you should set these bits to 0 before you issue a job to change a function switch.

Order of job execution

You can select several jobs simultaneously. If no jobs are active, the job management of the ABS_CTRL_451 instruction performs a scan to detect set trigger bits or changes to function switches, starting with the MDWR_EN job. If a job is found, it is processed. When the job is concluded, the job management searches for the next job to be processed. After having scanned the last ENCVAL_EN job, the function restarts the scan at the MDWR_EN job. This searching process is repeated until all the jobs have been processed.

The jobs are processed in the following technologically appropriate order:

Order	Address in the channel DB	Parameters	Description	Reset from
Write jobs				
1	35.0	MDWR_EN	Write machine data	Instruction
2	35.1 35.2 35.3 36.4	MD_EN DELDIST_EN AVALREM_EN DELDIAG_EN	Enable machine data Delete remaining distance Cancel set actual value Delete diagnostic buffer	Instruction
3	35.4	TRGL1WR_EN	Write incremental dimension table 1	Instruction
4	35.5	TRGL2WR_EN	Write incremental dimension table 2	Instruction
5	35.6	REFPT_EN	Set reference point	Instruction

Order	Address in the channel DB	Parameters	Description	Reset from
6		Function switch:		User program
	34.0	PLOOP_ON	Loop approach in plus direction	
	34.1	MLOOP_ON	Loop approach in minus direction	
	34.2	EI_OFF	Do not evaluate enable input	
	34.3	EDGE_ON	Edge detection on	
	34.4	MSR_ON	Length measurement on	
7	35.7	AVAL_EN	Set actual value	Instruction
8	36.0	FVAL_EN	Set actual value on-the-fly	Instruction
9	36.1	ZOFF_EN	Set zero point offset	Instruction
10	36.2	TRG252_254_EN	Write increment for increment number 252/254	Instruction
11	36.3	TRG255_EN	Write increment for increment number 255	Instruction
Read jobs				
12	36.5	MDRD_EN	Reading machine data	Instruction
13	36.6	TRGL1RD_EN	Read incremental dimension table 1	Instruction
14	36.7	TRGL2RD_EN	Read incremental dimension table 2	Instruction
15	37.0	MSRRD_EN	Read measured values	Instruction
16	37.1	ACTSPD_EN	Read current velocity, remaining distance and current increment	Instruction
17	37.2	ENCVAL_EN	Read encoder data	Instruction

This order enables you to completely trigger a positioning with a set of jobs and control signals. The jobs go from writing and activating the machine data through the setting of the external enable input up to writing the increments for the incremental approaches.

Jobs during an ongoing positioning

If initiated while a positioning job is active, the write jobs listed in the following table are held until the current positioning job is completed and are then executed at the next instruction call.

Address	Parameters	Data type	Initial value	Description
34.0	PLOOP_ON	BOOL	FALSE	1 = loop approach in plus direction
34.1	MLOOP_ON	BOOL	FALSE	1 = loop approach in minus direction
34.2	EI_OFF	BOOL	FALSE	1 = do not evaluate enable input
34.3	EDGE_ON	BOOL	FALSE	1 = Edge detection on
34.4	MSR_ON	BOOL	FALSE	1 = Length measurement on
35.1	MD_EN	BOOL	FALSE	1 = Enable machine data
35.2	DELDIST_EN	BOOL	FALSE	1 = delete remaining distance
35.3	AVALREM_EN	BOOL	FALSE	1 = cancel set actual value
35.6	REFPT_EN	BOOL	FALSE	1 = set reference point coordinate
35.7	AVAL_EN	BOOL	FALSE	1 = Set actual value
36.1	ZOFF_EN	BOOL	FALSE	1 = Set zero offset
36.4	DELDIAG_EN	BOOL	FALSE	1 = delete diagnostic buffer

Job status

The status of job execution can be read from the return value RET_VAL and activity bit JOBBUSY in the channel DB. The status of an individual job can be evaluated using the trigger bits, done bits, and error bits of this job.

	RET_VAL	JOBBUSY	Trigger bit _EN	Done bit _D	Error bit _ERR
Job active	1	1	1	0	0
Job completed without errors	0	0	0	1	0
Job completed with errors	-1	0	0	1	1
Write job cancelled	-1	0	0	0	1

Error classes

Error class 1: Process error

Description

Operating errors are detected asynchronously to an operation/control. These operating errors cause the positioning to be canceled, except in the case of error number 9, which leads to the shutdown of positioning.

No.	Meaning		Diagnostics interrupt
1	Software limit switch start overrun		Yes
	Cause	The actual value is outside the operating range.	
2	Software limit switch end overrun		Yes
	Cause	The actual value is outside the operating range.	
3	Traversing range start overrun		Yes
	Cause	Traversing range limit overrun (coordinates of the traversing range limits also belong to the traversing range).	
4	Traversing range end overrun		Yes
	Cause	Traversing range limit overrun (coordinates of the traversing range limits also belong to the traversing range).	
5	Error on target approach		Yes
	Cause	Target range has not been reached within the monitoring time.	
6	Standstill range exited		Yes
	Cause	The actual value lies outside of the standstill range.	
7	Positive feedback		Yes
	Cause	Actual value change > ½ standstill range in the wrong direction.	
8	Missing or too slight actual value change		Yes
	Cause	No actual value change or an actual value change against the desired direction within the monitoring time.	

No.	Meaning		Diagnostics interrupt
9	Target overrun (FM 451)		Yes
	Cause	The target has been overrun with "set actual value on-the-fly".	
10	Target range overrun		Yes
	Cause	Target range was overrun after target approach.	
11	Changeover point switched incorrectly		Yes
	Cause	Axis swinging in changeover point.	
12	Switch-off point switched incorrectly		Yes
	Cause	Axis swinging between switch-off / return point.	
13	Start of target range switched incorrectly		Yes
	Cause	Axis oscillating in the target range.	
14	Change greater than half the rotary axis range		Yes
	Cause	Velocity/frequency too high or faulty actual value steps.	
15	Change greater than the rotary axis range		Yes
	Cause	Velocity/frequency too high or faulty actual value steps.	
16	Increment for increment number 252 cannot be transferred (FM 451)		Yes
	Cause	The increment cannot be transferred.	
17	Increment for increment number 252 cannot be approached (FM 451)		Yes
	Cause	The distance between the current actual position and the specified increment is less than the changeover difference or switch-off difference.	
18	Incorrect increment for increment number 252 (FM 451)		Yes
	Cause	The increment lies outside of the operating range.	

Error class 2: Operator error

Description

Operator errors are detected if there is change in the control signals in the user data range. Operator errors lead to the shutdown of positioning.

No.	Meaning		Diagnostics interrupt
1	Impermissible operating mode		No
	Cause	The chosen operating mode is inadmissible.	
3	Inadmissible interface job		No
	Cause	The selected signal is not permitted with this operating mode.	
4	Incorrect operating mode parameter		No
	Cause	• In "jog" mode, the velocity specification is not equal to the rapid traverse or creep speed. • In "incremental approach" mode, the increment is a value other than 1 - 100, 252, 254 or 255	
5	Start enable not available		No
	Cause	Start enable not available upon start.	
7	Target / target range is outside the operating range		No

No.	Meaning		Diagnostics interrupt
	Cause	Default or calculated target is outside the software limit switches.	
8	**Axis parameters not assigned**		No
	Cause	Incorrect machine data or no machine data has been assigned for the axis.	
9	**Axis not synchronized**		No
	Cause	The "incremental approach" mode is only possible with an already synchronized axis.	
10	**Target / distance increment cannot be positioned**		No
	Cause	The distance between the current actual position and the specified target is less than the switch-off difference.	
17	**Reference point approach not possible**		No
	Cause	An SSI encoder is connected.	
18	**Relative or absolute incremental approach is not possible**		No
	Cause	The increment is invalid.	
19	**Switch-off difference is less than ½ the target range with increment number 255**		No
	Cause	The switch-off difference for the increment number 255 is less than half the target range.	
20	**Inadmissible travel in the specified direction**		No
	Cause	Insufficient distance to the software limit switch.	

Error class 4: Data error

Meaning

Operating errors are detected synchronously to an operation / control. Data errors do not trigger an error response.

No.	Meaning		Diagnostics interrupt
6	**Specified increment too great**		No
	Cause	The value is outside the range of ±100 m or ±1000 m. The distance increment / target may not be greater than the traversing range. For a rotary axis, the coordinate must be ≥ 0 and less than the end of the rotary axis.	
10	**Zero offset error**		No
	Cause	The zero offset is greater than ±100 m or ±1000 m. After the zero offset, the software limit switches are positioned outside of the traversing range (-100 m to +100 m or -1000 m to +1000 m). Rotary axis: The zero offset amount is greater than the end of the rotary axis.	
11	**Incorrect actual value**		No
	Cause	Linear axis: The coordinate is outside the current (possibly shifted) software limit switches. Rotary axis: The coordinate is < 0 or greater than the end of the rotary axis.	
12	**Incorrect reference point**		No

No.	Meaning		Diagnostics interrupt
	Cause	Linear axis: The coordinate is outside the current (possibly shifted) software limit switches. Rotary axis: The coordinate is < 0 or greater than the end of the rotary axis.	
20	**Enable machine data not permitted**		No
	Cause	There are no new (error-free) machine data on the module.	
27	**Unauthorized bit-coded setting**		No
	Cause	Unused and, in this case, unwritten bits do not equal 0.	
29	**Inadmissible bit coding**		No
	Cause	Unused and, in this case, unwritten bits do not equal 0.	
34	**Not possible to cancel set actual value**		No
	Cause	With an SSI encoder and a linear axis, the actual position value would lie outside the operating range after making the setting.	
36	**Incorrect changeover difference with the increment number 255**		No
	Cause	The value lies outside the permissible range of numbers of ±100 m or ±1000 m. For a rotary axis, the coordinate must be ≥ 0 and less than the end of the rotary axis.	
37	**Incorrect switch-off difference with the increment number 255**		No
	Cause	The value lies outside the permissible range of numbers of ±100 m or ±1000 m. The switch-off difference must be less than the changeover difference.	
107	**Axis parameters not assigned**		No
	Cause	On the axis there is either no machine data or it has not been enabled.	
108	**Axis not synchronized**		No
	Cause	A "set actual value" or "set actual value on-the-fly" job was initiated although the axis is not synchronized.	

Error class 5: Machine data error

Meaning

The diagnostic interrupt is only triggered in the event of a faulty system data block (SDB). Machine data errors do not trigger an error response.

No.	Meaning		Diagnostics interrupt
5	**Error in hardware interrupt setting**		Yes
	Cause	You have attempted to select a hardware interrupt that the module does not support.	
6	**Incorrect minimum edge distance (FM 451)**		Yes
	Cause	You have entered a value < 0 or > 10^9 µm as the minimum edge distance.	
7	**Incorrect system of units**		Yes
	Cause	The value for the system of units lies outside of the permissible range of 1 to 4 and 6.	

No.	Meaning		Diagnostics interrupt
8	Incorrect axis type		Yes
	Cause	You have entered neither 0 nor 1 as an axis type.	
9	Incorrect end of the rotary axis		Yes
	Cause	The value for the end of the rotary axis is outside the valid range from 1 to 10^9 µm or from 1 to 10^8 µm (depending on the resolution).	
10	Incorrect encoder type		Yes
	Cause	The value for the encoder type lies outside of the permissible range of 1 to 4.	
11	Incorrect distance per encoder revolution		Yes
	Cause	The value for distance / encoder revolution is outside the valid range from 1 to 10^9 µm (regardless of the resolution).	
13	Invalid increments per encoder revolution		Yes
14	Incorrect number of revolutions		Yes
15	Incorrect transmission rate		Yes
	Cause	You have specified a transmission rate outside the permitted range of 0 to 3.	
16	Incorrect reference point coordinate		Yes
	Cause	The coordinate is outside the range from -100 m to +100 m or from 1000 m to +1000 m (depending on the resolution). Linear axis: The coordinate lies outside of the operating range. Rotary axis: The coordinate is < 0 or greater than the end of the rotary axis.	
17	Incorrect absolute encoder adjustment		Yes
	Cause	SSI encoder: The value of the absolute encoder adjustment is not in the encoder range: (increments per encoder revolution x number of revolutions - 1).	
18	Incorrect type of reference point approach		Yes
	Cause	You have specified a value outside the permissible value set of 0, 1, 2 and 3.	
19	Incorrect counting direction		Yes
	Cause	You have specified a value outside of the permissible value quantities of 0 and 1.	
20	It is not possible to monitor the hardware		Yes
	Cause	You have set the monitoring of message frame errors in the parameter DB to "FALSE". The encoder used does not support error pulse monitoring. Disable the MON_PULSE parameter.	
21	Incorrect software limit switch start		Yes
	Cause	Linear axis: The software limit switch start is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m, depending on the resolution). Linear axis: The software limit switch start (including any zero offset) is less than -100 m or -1000 m (depending on the resolution).	
22	Incorrect software limit switch end		Yes

No.	Meaning		Diagnostics interrupt
	Cause	Linear axis: The software limit switch end is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m, depending on the resolution) or less than the software limit switch start. Linear axis: The software limit switch end (including any zero offset) is greater than +100 m or +1000 m (depending on the resolution).	
23	Incorrect maximum velocity		Yes
	Cause	The data not listed in the parameter DB must be 0.	
24	Incorrect target range		Yes
	Cause	Linear axis: Range from 0 to 100 m or 1000 m, depending on the resolution. Rotary axis: Range greater than the end of the rotary axis.	
25	Incorrect monitoring time		Yes
	Cause	The value for the monitoring time lies outside the permissible range from 0 to 100 000 ms.	
26	Incorrect standstill range		Yes
	Cause	Linear axis: Range from 0 to 100 m or 1000 m, depending on the resolution. Rotary axis: Range greater than the end of the rotary axis.	
127	Incorrect standstill velocity		Yes
	Cause	The value for the standstill velocity lies outside the permissible range from 0 to 100 000 µm/min.	
128	Incorrect control mode		Yes
	Cause	You have specified a value for the control mode that lies outside the permissible range of 1 to 4.	
129	Incorrect start velocity for the reference point approach		Yes
	Cause	You have entered neither 0 nor 1 as a start velocity.	
130	Incorrect changeover difference in the direction +		Yes
	Cause	Linear axis: Range from 0 to 100 m or 1000 m (depending on the resolution). Rotary axis: Range greater than the end of the rotary axis or less than ½ the target range.	
131	Incorrect changeover difference in direction –		Yes
	Cause	Linear axis: Range from 0 to 100 m or 1000 m (depending on the resolution). Rotary axis: Range greater than the end of the rotary axis or less than ½ the target range.	
132	Incorrect switch-off difference in direction +		Yes
	Cause	The switch-off difference is greater than the plus changeover difference, less than ½ the target range or outside the valid range from 0 to 100 m or 0 to 1000 m (depending on resolution).	
133	Incorrect switch-off difference in direction –		Yes
	Cause	The switch-off difference is greater than the minus changeover difference, less than ½ the target range or outside the valid range from 0 to 100 m or 0 to 1000 m (depending on resolution).	
200	Incorrect resolution		Yes
	Cause	You specified a resolution < 0.1 µm/pulse or > 1000 µm/pulse. You specified a distance / encoder revolution and a number of pulses/ encoder revolution, that results in a resolution of < 0.1 or > 1000.	

No.	Meaning		Diagnostics interrupt
201	Encoder does not fit the operating range / rotary axis range		Yes
	Cause	SSI encoders and rotary axes: The encoder does not precisely cover the rotary axis range. Linear axis: The encoder does not cover at least the operating range (including software limit switches).	

Error class 6: Increment table error

Meaning

Increment dimension table errors do not trigger an error response.

No.	Meaning		Diagnostics interrupt
6	Specified increment in the incremental dimension table too great		No
	Cause	The value lies outside of ±100 m or ±1000 m. The increment/target may not be greater than the traversing range. For a rotary axis, the coordinate must be ≥ 0 and less than the end of the rotary axis.	

Error class 15: Messages

Meaning

The messages do not trigger an error response.

No.	Meaning		Diagnostics interrupt
1	Start of parameter assignment		No
	Cause	The module has detected a parameter assignment via a system data block.	
2	End of parameter assignment		No
	Cause	The module has processed the parameter assignment via a system data block without errors.	
11	Distance to changeover point too short		No
	Cause	The hardware response times cannot be adhered to because the distance between the switching points is too short.	
12	Distance to reversal point too short		No
	Cause	The hardware response times cannot be adhered to because the distance between the switching points is too short.	
14	Distance to switch-off point too short		No
	Cause	The hardware response times cannot be adhered to because the distance between the switching points is too short.	

No.	Meaning		Diagnostics interrupt
15	**Distance to start of target range too short**		No
	Cause	The hardware response times cannot be adhered to because the distance between the switching points is too short.	

Error class 128: Diagnostics errors

No.	Meaning		Diagnostics interrupt
4	**External auxiliary voltage missing**		Yes
	Cause	• External 24 V auxiliary voltage is not connected or has failed • The fuse on the module is defective • Undervoltage • Ground wire break • Short-circuit (for example, at the connected encoder)	
	Effect	• Positioning is cancelled on all channels. • Outputs are switched off • Synchronization is cancelled for incremental encoders when the auxiliary voltage for the encoder supply is missing • Module parameters not assigned. • Start enable is cleared	
	Remedy	Verify that the 24 V connection is correct (faulty module if 24 V connection is correct).	
5	**Front connector missing (FM 451)**		Yes
	Cause	Front connector of the positioning module is not connected.	
	Effect	• Missing external auxiliary 24 V voltage • Module is not ready	
	Correction	Plug the front connector into the positioning module	
51	**Timeout monitoring responded (watchdog)**		Yes
	Cause	• Module exposed to severe interference • Module error	
	Effect	• Module is reset • Turn off all outputs • If, after resetting the module, no module fault is detected, the module is ready for operation again • The module signals the watchdog timeout with "incoming" and "outgoing"	
	Correction	• Remedying interferences • Contact the relevant sales department who will require details of the circumstances leading to the error. • Replace the module	
144	**Encoder wire break**		Yes

No.	Meaning		Diagnostics interrupt
	Cause	• The encoder cable has sheared off or is not plugged in • Encoders without cross signals • Incorrect pin assignment • Cable length too long • Encoder signals short-circuited • Encoder signal edge fault • Maximum input frequency of the encoder input has been exceeded • Sensor voltage supply failure	
	Effect	• The positioning is canceled • Turn off all outputs • Delete the synchronization with incremental encoders • Start enable is deleted	
	Correction	• Check encoder cable • Comply with encoder specification • Monitoring can be temporarily disabled at the responsibility of the operator using the programming interface. • Comply with the technical specifications of the module	
145	Absolute encoder message frame error		Yes
	Cause	Incorrect or interrupted message frame traffic between the module and the absolute encoder (SSI): • The encoder cable has sheared off or is not plugged in • Incorrect encoder type • The encoder is incorrectly set (programmable encoder) • The message length has been incorrectly specified • The encoder is supplying incorrect values (encoder is faulty) • Interference on the measuring system cable • Chosen transmission rate is too high • Monoflop time of the encoder is greater than 64 µs	
	Effect	• Positioning is canceled • Shutdown of the outputs • Start enable is cleared	
	Correction	• Check encoder cable • Check encoder • Check the message frame traffic between the encoder and module	
146	Missing pulses of incremental encoder		Yes
	Cause	• Encoder monitoring has detected missing pulses • Number of increments per encoder revolution is incorrectly entered • Encoder faulty: does not return the defined number of pulses • incorrect or no zero mark • Crosstalk on the encoder cable	

No.	Meaning		Diagnostics interrupt
	Effect	• Positioning is canceled • Shutdown of the outputs • Start enable is cleared	
	Remedy	• Enter the correct number of increments per encoder revolution (parameter assignment dialog) • Check the encoder and encoder cable • Comply with the shielding and grounding directives • Monitoring can be temporarily disabled at the responsibility of the operator using the programming interface.	

FM x52 Cam Controlling

CAM_INIT

Description

The CAM_INIT instruction initializes the following data in the channel DB:

- The control signals
- The feedback signals
- The trigger, done and error bits of the jobs
- The function switches and their done and error bits
- Job management and the internal buffers for the CAM_CTRL (Page 7157) or CAM_CTRL_452 (Page 7160) instruction
- Job management and the internal buffers for the CAM_MSRM_452 (Page 7163) instruction (FM 452 only)

Call

The instruction must be executed after a startup, in other words, after the power supply to the module or CPU is switched on. Therefore, you should integrate the instruction in the restart organization block OB 100, for example, and plug/pull organization block OB 83 or call it in the initialization phase of your user program. Before you do that, enter the module address in the DB of OB 100. This ensures that your user program does not access obsolete data after a CPU or module restart.

Parameters

Parameters	Declaration	Data type	Description
DB_NO	INPUT	INT	Number of the channel DB

Return values

The instruction does not provide a return value.

CAM_CTRL

Description

You can call CAM_CTRL instruction to read operating data from the module, initialize the module and control the module in runtime. To complete such actions, use control signals, feedback signals, function switches as well as write and read commands.

Operating principle

Each time it is called, the instruction performs the following actions:

- Read feedback signals:
  The CAM_CTRL instruction reads all feedback signals from the module and enters them in the channel DB. The control signals and commands are not processed unless this step is completed, which means that the feedback signals report the module status given prior to the instruction call.
- Write control signals:
  The control signals that are entered in the channel DB are transferred to the module. Enabling of cam processing, however, is delayed as long as the trigger for a "Set reference point" or "Write cam data" command is set. Activation (reactivation) of cam processing is delayed by this time period.
- Command management (Page 7175):
  The next command is executed based on the command trigger bits entered in the channel DB.

Call

This instruction must be called at cyclic intervals.

Before calling the instruction, enter all the data necessary to execute the selected functions in the channel DB.

Data blocks used

- Channel DB (Page 7164):
  The channel DB is the instance DB of the instruction CAM_CTRL. The module address must be entered in the channel DB. .
- Parameter DB (Page 7171):
  To write or read machine or cam data using commands, you need a parameter DB whose number is entered in the channel DB. The parameter DB must be of sufficient length to handle the existing number of cams.

Return values

The instruction provides the following return values in the RETVAL parameter of the channel DB:

RETVAL	BR	Description
1	1	At least 1 command is active
0	1	No active command, no errors
-1	0	Errors: Data error (DAT_ERR) or Communication error (JOB_ERR) has occurred

Startup

Call the CAM_INIT (Page 7156) instruction at the startup of the module or CPU. This action includes a reset of the function switches.

The CAM_CTRL instruction acknowledges the module startup. During this time, RETVAL and JOBBUSY = 1.

Reaction to error

If corrupt data was written by a write command, the module returns the feedback signal DATA_ERR = 1. If an error occurs during communication with the module for a write or read command, the cause of the error is written to the JOB_ERR parameter of the channel DB.

- Error with a write command:
  For the corrupt command, the trigger bit is reset, and error bit _ERR and done bit _D are set. The trigger bit is canceled and error bit _ERR is set for all pending write commands. The execution of the pending read commands is continued. JOB_ERR is retriggered accordingly for all commands.
- Error with a read command:
  For the corrupt command, the trigger bit is reset, and error bit _ERR and done bit _D are set.
  The execution of the pending read commands is continued. JOB_ERR is retriggered accordingly for all commands.

CAM_DIAG

Description

Use the CAM_DIAG instruction to read the data from the diagnostics buffer of the module and make it available for display on the HMI system or for programmed evaluation.

Operating principle

The instruction reads the diagnostics buffer if a new entry is displayed in the diagnostics buffer by feedback signal DIAG = 1. After the diagnostics buffer has been read, the module sets DIAG to 0.

Call

This instruction must be called at cyclic intervals. Additional calls in an event-driven interrupt program are not allowed. To complete the execution of the instruction, you must call it at least twice (in 2 cycles).

Data block used

Diagnostics DB (Page 7174):

The diagnostics DB is the instance DB of the instruction CAM_DIAG. The module address must be entered in the diagnostics DB. The address should entered in the startup OB (OB 100) or prior to its initial use. The most recent entry in the diagnostics buffer is written to the DIAG[1] structure and the oldest entry to the DIAG[4] structure.

Return values

The instruction provides the following return values in the RETVAL parameter of the diagnostics DB:

RETVAL	BR	Description
1	1	Command active
0	1	No active command, no errors
-1	0	Errors

Commands

You can read the diagnostics buffer regardless of any new entry by setting the DIAGRD_EN trigger bit. After the diagnostics buffer has been read, the trigger bit is set to 0.

Startup

The instruction does not run a startup routine.

Reaction to error

The cause of a command error can be read from the JOB_ERR parameter of the diagnostics DB.

Error classes

The error events are organized in the following error classes:

- Error class 1: Operational errors (Page 7178)
- Error class 4: Data errors (Page 7178)
- Error class 5: Machine data errors (Page 7180)
- Error class 7: Cam data errors (Page 7182)

- Error class 15: Messages (Page 7183)
- Error class 128: Diagnostics errors (Page 7183)

CAM_CTRL_452

Description

You can call CAM_CTRL_452 instruction to read operating data from the module, initialize the module and control the module in runtime. To complete such actions, use control signals, feedback signals, function switches as well as write and read jobs.

Operating principle

Each time it is called, the instruction performs the following actions:

- Read feedback signals:
  The CAM_CTRL_452 instruction reads all feedback signals from the module and enters them in the channel DB. The control signals and jobs are not processed unless this step is completed, which means that the feedback signals report the module status given prior to the instruction call.
- Write control signals:
  The control signals that are entered in the channel DB are transferred to the module. Enabling of cam processing, however, is delayed as long as the trigger for a "Set reference point" or "Write cam data" job is set. Activation (reactivation) of cam processing is delayed by this time period.
- Job management (Page 7176):
  The next job is executed based on the job trigger bits entered in the channel DB.

Call

This instruction must be called at cyclic intervals.

Before calling the instruction, enter all the data necessary to execute the selected functions in the channel DB.

Data blocks used

- Channel DB (Page 7164):
  The module address must be entered in the channel DB.
- Parameter DB (Page 7171):
  To write or read machine or cam data using jobs, you need a parameter DB whose number is entered in the channel DB. The parameter DB must be of sufficient length to handle the existing number of cams.

Parameters

Parameters	Declaration	Data type	Description
DB_NO	INPUT	INT	Number of the channel DB
RET_VAL	OUTPUT	INT	Return value

Return values

The instruction provides the following return values in the RET_VAL parameter of the channel DB:

RET_VAL	BR	Description
1	1	At least 1 job is active
0	1	No active job, no errors
-1	0	Errors: Data error (DAT_ERR) or Communication error (JOB_ERR) has occurred

Startup

Call the CAM_INIT (Page 7156) instruction at the startup of the module or CPU. This action includes a reset the function switches.

The CAM_CTRL_452 instruction acknowledges the module startup. During this time, RET_VAL and JOBBUSY = 1.

Reaction to errors

If corrupt data was written by a write job, the module returns the feedback signal DATA_ERR = 1. If an error occurs during communication with the module for a write or read job, the cause of the error is written to the JOB_ERR parameter of the channel DB.

- Write job errors:
  For the corrupt job, the trigger bit is reset, and error bit _ERR and done bit _D are set. The trigger bit is canceled and error bit _ERR is set for all pending write jobs.
  The execution of the pending read jobs is continued. JOB_ERR is retriggered accordingly for all jobs.
- Read job errors:
  For the corrupt job, the trigger bit is reset, and error bit _ERR and done bit _D are set.
  The execution of the pending read jobs is continued. JOB_ERR is retriggered accordingly for all jobs.

CAM_DIAG_452

Description

Use the CAM_DIAG_452 instruction to read the data from the diagnostics buffer of the module and make it available for display on the HMI system or for programmed evaluation.

Operating principle

The instruction reads the diagnostics buffer if a new entry is displayed in the diagnostics buffer by feedback signal DIAG = 1. After the diagnostics buffer has been read, the module sets DIAG to 0.

Call

This instruction must be called at cyclic intervals. Additional calls in an interrupt OB are not allowed. To complete the execution of the instruction, you must call it at least twice (in 2 cycles).

Data block used

Diagnostics DB (Page 7174):

The module address must be entered in the diagnostics DB. The most recent entry in the diagnostics buffer is written to the DIAG[1] structure and the oldest entry to the DIAG[4] structure.

Parameters

Parameters	Declaration	Data type	Description
DB_NO	INPUT	INT	Number of the diagnostics DB
RET_VAL	OUTPUT	INT	Return value

Return values

The instruction provides the following return values in the RET_VAL parameter of the diagnostics DB:

RET_VAL	BR	Description
1	1	Job active
0	1	No active job, no errors
-1	0	Errors

Jobs

You can read the diagnostics buffer regardless of any new entry by setting the DIAGRD_EN trigger bit. After the diagnostics buffer has been read, the trigger bit is set to 0.

Startup

The instruction does not run a startup routine.

Reaction to errors

The cause of a job error can be read from the JOB_ERR parameter of the diagnostics DB.

Error classes

The error events are organized in the following error classes:

- Error class 1: Operating error (Page 7178)
- Error class 4: Data errors (Page 7178)
- Error class 5: Machine data errors (Page 7180)
- Error class 7: Cam data errors (Page 7182)
- Error class 15: Messages (Page 7183)
- Error class 128: Diagnostics errors (Page 7183)

CAM_MSRM_452

Description

Call the CAM_MSRM_452 instruction to immediately evaluate the length measurement or edge detection data of an FM 452 in the hardware interrupt OB.

Call

The instruction is called in a hardware interrupt OB (for example, OB 40).

Data block used

Channel DB (Page 7164):

The module address must be entered in the channel DB. You should enter it in the startup OB (OB 100).

Parameters

Parameters	Declaration	Data type	Description
DB_NO	INPUT	INT	Number of the channel DB
RET_VAL	OUTPUT	INT	Return value

Return values

The instruction provides the following return values in the RET_VAL parameter of the channel DB:

RET_VAL	BR	Description
1	1	Job active
0	1	No active job, no errors
-1	0	Errors

Measurement results and status information

The measurement results and status information are available in the channel DB:

Address	Name	Type	Initial value	Comment
112.0	BEG_VAL	DINT	L#0	Initial value
116.0	END_VAL	DINT	L#0	End value
120.0	LEN_VAL	DINT	L#0	Length
56.0	JOB_ERR_M	INT	0	Communication errors
58.0	JOBBUSY_M	BOOL	FALSE	Job active

Startup

The instruction does not run a startup routine.

Reaction to errors

The cause of a job error can be read from the JOB_ERR_M parameter of the channel DB.

Additional references for FM x52 CAM V2

Channel data block

Purpose

The channel DB is the data interface between the user program and the electronic cam controller FM x52. It contains and saves all data necessary to control and operate the module.

Configuration

The channel DB is divided into different areas:

Channel DB
Addresses* / version switch
Control signals
Feedback signals
Function switches
Trigger bits for write jobs
Trigger bits for read jobs
Done bits
Error bits
Job management for instructions
Data for jobs
* You can enter the module address in the parameter assignment interface.

Content

The table below shows the content of the channel DB.

Note

Do not modify any data not listed in this table.

Address	Parameters	Data type	Initial value	Description
Addresses / version switch				
0.0	MOD_ADDR (Enter!)	INT	0	Module address
2.0	CH_NO	INT	1	Channel number (always 1)
10.0	PARADBNO	INT	-1	Number of the parameter DB -1 = DB not available
12.0	FM_TYPE	BOOL	FALSE	0 = FM 352 up to V4.0 1 = FM 452 or FM 352 V5.0 and higher
Control signals				
15.2	DIR_M	BOOL	FALSE	1 = simulation in negative direction
15.3	DIR_P	BOOL	FALSE	1 = simulation in positive direction
15.4	CAM_EN	BOOL	FALSE	1 = enable cam processing
15.5	CNTC0_EN	BOOL	FALSE	1 = enable count function of counter cam track 0
15.6	CNTC1_EN	BOOL	FALSE	1 = enable count function of counter cam track 1
16.0	TRACK_EN	WORD	W#16#0	Enable cam tracks 0 to 15 Bit 0 = track 0
Feedback signals				

Address	Parameters	Data type	Initial value	Description
22.2	DIAG	BOOL	FALSE	1 = diagnostic buffer modified
22.4	DATA_ERR	BOOL	FALSE	1 = data error
22.7	PARA	BOOL	FALSE	1 = module has parameters assigned
23.4	CAM_ACT	BOOL	FALSE	1 = cam processing busy
25.0	SYNC	BOOL	FALSE	1 = axis is synchronized
25.1	MSR_DONE	BOOL	FALSE	1 = length measurement or edge detection completed
25.2	GO_M	BOOL	FALSE	1 = axis moving in negative direction
25.3	GO_P	BOOL	FALSE	1 = axis moving in positive direction
25.4	HYS	BOOL	FALSE	1 = axis within the hysteresis range
25.5	FVAL_DONE	BOOL	FALSE	1 = set actual value on-the-fly executed
26.0	ACT_POS	DINT	L#0	Current position of axis
30.0	TRACK_OUT	DWORD	DW#16#0	Current signals of tracks 0 to 31 Bit 0 = track 0
Function switches				
34.0	EDGE_ON	BOOL	FALSE	1 = Edge detection on
34.1	SIM_ON	BOOL	FALSE	1 = simulation on
34.2	MSR_ON	BOOL	FALSE	1 = Length measurement on
34.3	REFTR_ON	BOOL	FALSE	1 = retrigger reference point
34.4	SSW_OFF	BOOL	FALSE	1 = Software limit switches disabled
Trigger bits for write jobs				
35.0	MDWR_EN	BOOL	FALSE	1 = write machine data
35.1	MD_EN	BOOL	FALSE	1 = Enable machine data
35.2	AVALREM_EN	BOOL	FALSE	1 = set actual value, cancel set actual value on-the-fly
35.3	CAM1WR_EN	BOOL	FALSE	1 = write cam data 1 (cams 0 to 15)
35.4	CAM2WR_EN	BOOL	FALSE	1 = write cam data 2 (cams 16 to 31)
35.5	CAM3WR_EN	BOOL	FALSE	1 = write cam data 3 (cams 32 to 47)
35.6	CAM4WR_EN	BOOL	FALSE	1 = write cam data 4 (cams 48 to 63)
35.7	CAM5WR_EN	BOOL	FALSE	1 = write cam data 5 (cams 64 to 79)
36.0	CAM6WR_EN	BOOL	FALSE	1 = write cam data 6 (cams 80 to 95)
36.1	CAM7WR_EN	BOOL	FALSE	1 = write cam data 7 (cams 96 to 111)
36.2	CAM8WR_EN	BOOL	FALSE	1 = write cam data 8 (cams 112 to 127)
36.3	REFPT_EN	BOOL	FALSE	1 = set reference point coordinates
36.4	AVAL_EN	BOOL	FALSE	1 = Set actual value

Address	Parameters	Data type	Initial value	Description
36.5	FVAL_EN	BOOL	FALSE	1 = set actual value on-the-fly
36.6	ZOFF_EN	BOOL	FALSE	1 = Set zero offset
36.7	CH01CAM_EN	BOOL	FALSE	1 = write cam edge setting (1 cam)
37.0	CH16CAM_EN	BOOL	FALSE	1 = Write setting for fast cam change (16 cams)
Trigger bits for read jobs				
37.1	MDRD_EN	BOOL	FALSE	1 = read machine data
37.2	CAM1RD_EN	BOOL	FALSE	1 = read cam data 1 (cams 0 to 15)
37.3	CAM2RD_EN	BOOL	FALSE	1 = read cam data 2 (cams 16 to 31)
37.4	CAM3RD_EN	BOOL	FALSE	1 = read cam data 3 (cams 32 to 47)
37.5	CAM4RD_EN	BOOL	FALSE	1 = read cam data 4 (cams 48 to 63)
37.6	CAM5RD_EN	BOOL	FALSE	1 = read cam data 5 (cams 64 to 79)
37.7	CAM6RD_EN	BOOL	FALSE	1 = read cam data 6 (cams 80 to 95)
38.0	CAM7RD_EN	BOOL	FALSE	1 = read cam data 7 (cams 96 to 111)
38.1	CAM8RD_EN	BOOL	FALSE	1 = read cam data 8 (cams 112 to 127)
38.2	MSRRD_EN	BOOL	FALSE	1 = read measured values
38.3	CNTTRC_EN	BOOL	FALSE	1 = read count values of counter cam tracks
38.4	ACTPOS_EN	BOOL	FALSE	1 = read position and track data
38.5	ENCVAL_EN	BOOL	FALSE	1 = read encoder values
38.6	CAMOUT_EN	BOOL	FALSE	1 = read cam and track data
Done bits for function switches				
40.0	EDGE_D	BOOL	FALSE	1 = "activate edge detection" or "deactivate edge detection" completed
40.1	SIM_D	BOOL	FALSE	1 = "activate simulation" or "deactivate simulation" completed
40.2	MSR_D	BOOL	FALSE	1 = "activate length measurement" or "deactivate length measurement" completed
40.3	REFTR_D	BOOL	FALSE	1 = "activate retrigger reference point" or "deactivate retrigger reference point" completed
40.4	SSW_D	BOOL	FALSE	1 = "Enable software limit switches" or "Disable software limit switches" completed
Done bits for write jobs				
41.0	MDWR_D	BOOL	FALSE	1 = "write machine data" job completed
41.1	MD_D	BOOL	FALSE	1 = "enable machine data" job completed
41.2	AVALREM_D	BOOL	FALSE	1 = "cancel set actual value" or "cancel set actual value on-the-fly" completed

Address	Parameters	Data type	Initial value	Description
41.3	CAM1WR_D	BOOL	FALSE	1 = "write cam data 1" job completed
41.4	CAM2WR_D	BOOL	FALSE	1 = "write cam data 2" job completed
41.5	CAM3WR_D	BOOL	FALSE	1 = "write cam data 3" job completed
41.6	CAM4WR_D	BOOL	FALSE	1 = "write cam data 4" job completed
41.7	CAM5WR_D	BOOL	FALSE	1 = "write cam data 5" job completed
42.0	CAM6WR_D	BOOL	FALSE	1 = "write cam data 6" job completed
42.1	CAM7WR_D	BOOL	FALSE	1 = "write cam data 7" job completed
42.2	CAM8WR_D	BOOL	FALSE	1 = "write cam data 8" job completed
42.3	REFPT_D	BOOL	FALSE	1 = "set reference point" job completed
42.4	AVAL_D	BOOL	FALSE	1 = "set actual value" job completed
42.5	FVAL_D	BOOL	FALSE	1 = "Set actual value on-the-fly" job completed
42.6	ZOFF_D	BOOL	FALSE	1 = "set zero offset" job completed
42.7	CH01CAM_D	BOOL	FALSE	1 = "change 1 cam" job completed
43.0	CH16CAM_D	BOOL	FALSE	1 = "change 16 cams" completed (fast cam parameter change)
Done bits for read jobs				
43.1	MDRD_D	BOOL	FALSE	1 = "read machine data" job completed
43.2	CAM1RD_D	BOOL	FALSE	1 = "read cam data 1" job completed
43.3	CAM2RD_D	BOOL	FALSE	1 = "read cam data 2" job completed
43.4	CAM3RD_D	BOOL	FALSE	1 = "read cam data 3" job completed
43.5	CAM4RD_D	BOOL	FALSE	1 = "read cam data 4" job completed
43.6	CAM5RD_D	BOOL	FALSE	1 = "read cam data 5" job completed
43.7	CAM6RD_D	BOOL	FALSE	1 = "read cam data 6" job completed
44.0	CAM7RD_D	BOOL	FALSE	1 = "read cam data 7" job completed
44.1	CAM8RD_D	BOOL	FALSE	1 = "read cam data 8" job completed
44.2	MSRRD_D	BOOL	FALSE	1 = "read measured values" job completed
44.3	CNTTRC_D	BOOL	FALSE	1 = "read count values of counter cam tracks" job completed
44.4	ACTPOS_D	BOOL	FALSE	1 = "read position and track data" job completed
44.5	ENCVAL_D	BOOL	FALSE	1 = "read actual encoder value" job completed
44.6	CAMOUT_D	BOOL	FALSE	1 = "read position and track data" job completed
Error bits for function switches				
46.0	EDGE_ERR	BOOL	FALSE	1 = error in "activate edge detection" or "deactivate edge detection"
46.1	SIM_ERR	BOOL	FALSE	1 = error in "activate simulation" or "deactivate simulation"
46.2	MSR_ERR	BOOL	FALSE	1 = error in "activate length measurement" or "deactivate length measurement"
46.3	REFTR_ERR	BOOL	FALSE	1 = Error in "Enable retrigger reference point" or "Disable retrigger reference point"
46.4	SSW_ERR	BOOL	FALSE	1 = Error in "Enable software limit switches" or "Disable software limit switches"
Error bits for write jobs				
47.0	MDWR_ERR	BOOL	FALSE	1 = error in "write machine data" job

Address	Parameters	Data type	Initial value	Description
47.1	MD_ERR	BOOL	FALSE	1 = error in "activate machine data" job
47.2	AVALREM_ERR	BOOL	FALSE	1 = error in "cancel set actual value" or "cancel set actual value on-the-fly"
47.3	CAM1WR_ERR	BOOL	FALSE	1 = error in "write cam data 1" job
47.4	CAM2WR_ERR	BOOL	FALSE	1 = error in "write cam data 2" job
47.5	CAM3WR_ERR	BOOL	FALSE	1 = error in "write cam data 3" job
47.6	CAM4WR_ERR	BOOL	FALSE	1 = error in "write cam data 4" job
47.7	CAM5WR_ERR	BOOL	FALSE	1 = error in "write cam data 5" job
48.0	CAM6WR_ERR	BOOL	FALSE	1 = error in "write cam data 6" job
48.1	CAM7WR_ERR	BOOL	FALSE	1 = error in "write cam data 7" job
48.2	CAM8WR_ERR	BOOL	FALSE	1 = error in "write cam data 8" job
48.3	REFPT_ERR	BOOL	FALSE	1 = error in "set reference point" job
48.4	AVAL_ERR	BOOL	FALSE	1 = error in "set actual value" job
48.5	FVAL_ERR	BOOL	FALSE	1 = error in "set actual value on-the-fly" job
48.6	ZOFF_ERR	BOOL	FALSE	1 = error in "set zero offset" job
48.7	CH01CAM_ERR	BOOL	FALSE	1 = error in "change 1 cam" job
49.0	CH16CAM_ERR	BOOL	FALSE	1 = error in "change 16 cams" (fast cam parameter change)
Error bits for read jobs				
49.1	MDRD_ERR	BOOL	FALSE	1 = error in "read machine data" job
49.2	CAM1RD_ERR	BOOL	FALSE	1 = error in "read cam data 1" job
49.3	CAM2RD_ERR	BOOL	FALSE	1 = error in "read cam data 2" job
49.4	CAM3RD_ERR	BOOL	FALSE	1 = error in "read cam data 3" job
49.5	CAM4RD_ERR	BOOL	FALSE	1 = error in "read cam data 4" job
49.6	CAM5RD_ERR	BOOL	FALSE	1 = error in "read cam data 5" job
49.7	CAM6RD_ERR	BOOL	FALSE	1 = error in "read cam data 6" job
50.0	CAM7RD_ERR	BOOL	FALSE	1 = error in "read cam data 7" job
50.1	CAM8RD_ERR	BOOL	FALSE	1 = error in "read cam data 8" job
50.2	MSRRD_ERR	BOOL	FALSE	1 = error in "read measured values" job
50.3	CNTTRC_ERR	BOOL	FALSE	1 = error in "read count values of counter cam tracks" job
50.4	ACTPOS_ERR	BOOL	FALSE	1 = error in "read position and track data" job
50.5	ENCVAL_ERR	BOOL	FALSE	1 = error in "read current encoder value" job
50.6	CAMOUT_ERR	BOOL	FALSE	1 = Error in "Read cam and track data" job
Job management for the instructionCAM_CTRL or CAM_CTRL_452				
52.0	JOB_ERR	INT	0	Communication errors
54.0	JOBBUSY	BOOL	FALSE	1 = At least 1 job active
54.1	JOBRESET	BOOL	FALSE	1 = Reset all error bits and done bits
Job management for the instruction CAM_MSRM_452				
56.0	JOB_ERR_M	INT	0	Communication errors
58.0	JOBBUSY_M	BOOL	FALSE	1 = Job active
Data for "Zero offset" job				
86.0	ZOFF	DINT	L#0	Zero offset

Address	Parameters	Data type	Initial value	Description
Data for "Set actual value" job				
90.0	AVAL	DINT	L#0	Coordinate for "Set actual value"
Data for "Set actual value on-the-fly" job				
94.0	FVAL	DINT	L#0	Coordinate for "Set actual value on-the-fly"
Data for "Set reference point" job				
98.0	REFPT	DINT	L#0	Coordinate for "Set reference point"
Data for "change cam edges" job				
102.0	CAM_NO	INT	0	Cam number
104.0	CAM_START	DINT	L#0	Cam start
108.0	CAM_END	DINT	L#0	Cam end
Data for the "Length measurement / edge detection" job				
112.0	BEG_VAL	DINT	L#0	Initial value
116.0	END_VAL	DINT	L#0	End value
120.0	LEN_VAL	DINT	L#0	Length
Data for the "read count values" job				
124.0	CNT_TRC0	INT	0	Current count value of counter cam track 0
126.0	CNT_TRC1	INT	0	Current count value of counter cam track 1
Data for the "read position and track data" job				
128.0	ACTPOS	DINT	L#0	Current position
132.0	ACTSPD	DINT	L#0	Actual velocity
136.0	TRACK_ID	DWORD	DW#16#0	Track identifier bits of tracks 0 to 31
Data for the "read encoder data" job				
140.0	ENCVAL	DINT	L#0	Encoder value
144.0	ZEROVAL	DINT	L#0	Count value at the last zero mark
148.0	ENC_ADJ	DINT	L#0	Absolute encoder adjustment
Data for the "read cam and track data" job				
152.0	CAM_00_31	DWORD	DW#16#0	Cam identifier bits for cams 0 to 31
156.0	CAM_32_63	DWORD	DW#16#0	Cam identifier bits for cams 32 to 63
160.0	CAM_64_95	DWORD	DW#16#0	Cam identifier bits for cams 64 to 95
164.0	CAM_96_127	DWORD	DW#16#0	Cam identifier bits for cams 96 to 127
168.0	TRACK_ID1	DWORD	DW#16#0	Track identifier bits of tracks 0 to 31
172.0	ACTPOS1	DINT	L#0	Current position
Data for the "fast cam parameter change" job				
176.0	C_QTY	BYTE	B#16#0	Number of cams to modify
177.0	DIS_CHECK	BOOL	FALSE	1 = disable data check
180.0	CAM	ARRAY [0...15] STRUCT		Note: The following structure must be completed for each cam to be modified
Relative address				
+0.0	CAM_NO	BYTE	B#16#0	Number of the cam to modify
+1.0	C_EFFDIR	BOOL	FALSE	1 = change the activation direction
+1.1	C_CBEGIN	BOOL	FALSE	1 = change the cam start to the value CBEGIN (new cam start)

Address	Parameters	Data type	Initial value	Description
+1.2	C_CEND	BOOL	FALSE	1 = change the cam end / activation time to the value CEND (new cam end)
+1.3	C_LTIME	BOOL	FALSE	1 = change the lead time to the LTIME value (new lead time)
+1.4	CAM_OFF	BOOL	FALSE	1 = deactivate the cam during the cam change
+1.5	EFFDIR_P	BOOL	FALSE	1 = new effective direction positive (plus)
+1.6	EFFDIR_M	BOOL	FALSE	1 = new effective direction negative (minus)
+2.0	CBEGIN	DINT	L#0	New cam start
+6.0	CEND	DINT	L#0	New cam end / new activation time
+10.0	LTIME	INT	0	New lead time

Parameter data block

Purpose

The machine and cam data is stored in the parameter DB. The parameters can be changed from the user program or from an HMI system. The modified data can be imported to the parameter assignment interface and displayed there. You can export data displayed in the parameter assignment interface to a parameter DB.

A module can contain several parameter data sets (for example, for different recipes) that can be selected from the program.

Configuration

Parameter DB
CAM_P016TYPE (UDT3) Machine data Data of cams 0 to 15
CAM_P032TYPE (UDT4) Machine data Data of cams 0 to 31
CAM_P064TYPE (UDT5) Machine data Data of cams 0 to 63
CAM_P0128TYPE (UDT6) Machine data Data of cams 0 to 127

Content

The table below shows the content of the parameter DB.

Note

You may not change any data not listed in this table.

Address	Parameters	Data type	Initial value	Description
Machine data				
3.1	PI_MEND	BOOL	FALSE	• FM 352: 0 • FM 452: 1: Enable hardware interrupt: Full scale value
3.2	PI_CAM	BOOL	FALSE	1: Enable hardware interrupt: Cam on/off
3.5	PI_MSTRT	BOOL	FALSE	• FM 352: 0 • FM 452: 1: Enable hardware interrupt: Start of measurement
4.0	EDGEDIST	DINT	L#0	Minimum edge distance for edge detection
8.0	UNITS	DINT	L#1	Physical units system
12.0	AXIS_TYPE	DINT	L#0	0: Linear axis, 1: Rotary axis
16.0	ENDROTAX	DINT	L#100 000	End of rotary axis
20.0	ENC_TYPE	DINT	L#1	Encoder type, frame length
24.0	DISP_REV	DINT	L#80 000	Distance per encoder revolution
32.0	INC_REV	DINT	L#500	Increments per encoder revolution
36.0	NO_REV	DINT	L#1024	Number of encoder revolutions
40.0	BAUDRATE	DINT	L#0	Transmission rate
44.0	REFPT	DINT	L#0	Reference point coordinate
48.0	ENC_ADJ	DINT	L#0	Absolute encoder adjustment
52.0	RETR_TYPE	DINT	L#0	Mode for "Retrigger reference point"
56.0	CNT_DIR	DINT	L#0	Count direction: 0: normal, 1: inverted
63.0	MON_WIRE	BOOL	TRUE	1: Wire-break monitoring
63.1	MON_FRAME	BOOL	TRUE	1: Message frame error monitoring
63.2	MON_PULSE	BOOL	TRUE	1: Missing pulse monitoring
64.0	SSW_STRT	DINT	L#-100 000 000	Software limit switch start
68.0	SSW_END	DINT	L#100 000 000	Software limit switch end
76.0	C_QTY	DINT	L#0	Number of cams: 0, 1, 2, 3 = max. 16, 32, 64, 128 cams
80.0	HYS	DINT	L#0	Hysteresis
84.0	SIM_SPD	DINT	L#0	Simulation velocity
90.0	TRACK_OUT	WORD	W#16#0	Control of track outputs: 0 = cam controller, 1 = CPU; Bit number = track number
95.0	EN_IN_I3	BOOL	FALSE	Enable input I3

Address	Parameters	Data type	Initial value	Description
95.1	EN_IN_I4	BOOL	FALSE	• FM 352: 0 • FM 452: Enable input I4
95.2	EN_IN_I5	BOOL	FALSE	• FM 352: 0 • FM 452: Enable input I5
95.3	EN_IN_I6	BOOL	FALSE	• FM 352: 0 • FM 452: Enable input I6
95.4	EN_IN_I7	BOOL	FALSE	• FM 352: 0 • FM 452: Enable input I7
95.5	EN_IN_I8	BOOL	FALSE	• FM 352: 0 • FM 452: Enable input I8
95.6	EN_IN_I9	BOOL	FALSE	• FM 352: 0 • FM 452: Enable input I9
95.7	EN_IN_I10	BOOL	FALSE	• FM 352: 0 • FM 452: Enable input I10
99.0	SPEC_TRC0	BOOL	FALSE	1 = track 0 is counter cam track
99.1	SPEC_TRC1	BOOL	FALSE	1 = track 1 is counter cam track
99.2	SPEC_TRC2	BOOL	FALSE	1 = track 2 is brake cam track
100.0	CNT_LIM0	DINT	L#2	High count value for counter cam track 0
104.0	CNT_LIM1	DINT	L#2	High count value for counter cam track 1
Data for cams 0 to 15 / 0 to 31 / 0 to 63 / 0 to 127				
108.0		STRUCT		(12 bytes length per element)
Relative address				
+0.0	CAMVALID	BOOL	FALSE	1: Cam valid
+0.1	EFFDIR_P	BOOL	TRUE	1: Positive effective direction (plus)
+0.2	EFFDIR_M	BOOL	TRUE	1: Negative effective direction (minus)
+0.3	CAM_TYPE	BOOL	FALSE	0: Distance cam, 1: Time-based cam
+0.4	PI_SW_ON	BOOL	FALSE	1: Hardware interrupt on activation
+0.5	PI_SW_OFF	BOOL	FALSE	1: Hardware interrupt on deactivation
+1.0	TRACK_NO	BYTE	B#16#0	Track number
+2.0	CBEGIN	DINT	L#-100 000 000	Cam start
+6.0	CEND	DINT	L#100 000 000	End of output cam / activation time
+10.0	LTIME	INT	0	Lead time

Diagnostics data block

Purpose

The diagnostics DB is the storage location for the CAM_DIAG (Page 7158) or CAM_DIAG_452 (Page 7162) instruction. It contains the module's diagnostics buffer which is processed by the instruction.

Configuration

Diagnostics DB
Module address
Internal data
Job status
Trigger bit
Processed diagnostics buffer

Content

The table below shows the content of the diagnostics DB.

Note

You may not change any data not listed in this table.

Address	Parameters	Data type	Initial value	Description
0.0	MOD_ADDR (Enter!)	INT	0	Module address
256.0	JOB_ERR	INT	0	Communication errors
258.0	JOBBUSY	BOOL	FALSE	1 = job active
258.1	DIAGRD_EN	BOOL	FALSE	1 = read diagnostic buffer unconditional
260.0	DIAG_CNT	INT	0	Number of valid entries in the list
262.0	DIAG[1]	STRUCT		Diagnostic data latest entry
272.0	DIAG[2]	STRUCT		Diagnostic data second entry
282.0	DIAG[3]	STRUCT		Diagnostic data third entry
292.0	DIAG[4]	STRUCT		Diagnostic data oldest entry

Structure of the Diagnostic Entry

The table below shows the structure of a diagnostics entry DIAG[n].

Address	Parameters	Data type	Initial value	Description
+0.0	STATE	BOOL	FALSE	0 = outgoing event 1 = incoming event
+0.1	INTF	BOOL	FALSE	1 = Internal error
+0.2	EXTF	BOOL	FALSE	1 = external error
+2.0	FCL	INT	0	Error class: 1: Operating error 4: Data error 5: Machine data error 7: Cam data error 15: Messages 128: Diagnostics errors
+4.0	FNO	INT	0	Error number 0 ... 255
+6.0	CH_NO	INT	0	Channel number (always 1)
+8.0	CAMNO	INT	0	Cam numbers 0 to 127 with error class = cam data error

Job management (CAM_CTRL)

Jobs

Data communication with the module other than the transfer of control and feedback signals is handled by means of jobs.

To initiate a job, set the corresponding trigger bit in the channel DB, including the corresponding data for write jobs. To execute the job, call the CAM_CTRL (Page 7157) instruction.

If operating the module centrally, read jobs are executed immediately. If you use the FM 352 in a distributed configuration, a read job may need several cycles.

Due to the required confirmations from the module, a write job requires at least 3 calls (or OB cycles). If you use the FM 352 in a distributed system, a write job may take more than 3 calls.

You can send several jobs at the same time, if necessary, along with control signals. With the exception of the job for writing the function switches, the jobs are executed in the order of the trigger bits specified in the channel DB. When a job is completed, the trigger bit is reset. The next job pending is identified and executed at the next call of the instruction.

In addition to a trigger bit, all jobs are provided a done bit and an error bit. These are distinguished by their name extension:

Bit	Extension
Trigger bit	_EN ("enable")
Done bit	_D ("done")
Error bit	_ERR ("error")

Done and error bits of the job should be set to 0 after they are evaluated or before this job is initiated.

When you set the JOBRESET bit, all done and error bits are reset before pending jobs are executed. The JOBRESET bit is then reset to 0.

Function switches

The function switches enable and disable module states. A job for writing the function switches is only executed when there is a change to a switch setting. The setting of the function switch is retained after the job has been executed.

Length measurements and edge detection cannot be activated concurrently. For this reason, the CAM_CTRL instruction makes sure that when one of the function switches is enabled, the other is disabled. However, if you enable both function switches (0 → 1), the length measurement is enabled.

Function switches and jobs can be used concurrently at the call of CAM_CTRL.

As with the jobs, the function switches are provided done bits with the name extension _D and error bits with the name extension _ERR.

To be able to evaluate the done and error bits, you should set these bits to 0 when changing a function switch.

Job status

The status of job execution can be read from the return value RETVAL and activity bit JOBBUSY in the channel DB. You can evaluate the status of a single job based on its trigger, done and error bits.

	RETVAL	JOBBUSY	Trigger bit _EN	Done bit _D	Error bit _ERR
Job active	1	1	1	0	0
Job completed without errors	0	0	0	1	0
Job completed with errors	-1	0	0	1	1
Write job cancelled	-1	0	0	0	1

Job management (CAM_CTRL_452)

Jobs

Data communication with the module other than the transfer of control and feedback signals is handled by means of jobs.

To initiate a command, set the corresponding trigger bit in the channel DB, including the corresponding data for write jobs. To execute the command, call the CAM_CTRL_452 (Page 7160) instruction.

A read command is executed immediately. Due to the required confirmations from the module, a write command requires at least 3 calls (or OB cycles).

You can send several jobs at the same time, if necessary, along with control signals. With the exception of the command for writing the function switches, the jobs are executed in the order of the trigger bits specified in the channel DB. When a command is completed, the trigger bit is reset. The next command pending is identified and executed at the next call of the instruction.

In addition to a trigger bit, all jobs are provided a done bit and an error bit. These are distinguished by their name extension:

Bit	Extension
Trigger bit	_EN ("enable")
Done bit	_D ("done")
Error bit	_ERR ("error")

Done and error bits of the command should be set to 0 after they are evaluated or before this command is initiated.

When you set the JOBRESET bit, all done and error bits are reset before pending jobs are executed. The JOBRESET bit is then reset to 0.

Function switches

The function switches enable and disable module states. A command for writing the function switches is only executed when there is a change to a switch setting. The setting of the function switch is retained after the command has been executed.

Length measurements and edge detection cannot be activated concurrently. For this reason, the CAM_CTRL_452 instruction makes sure that when one of the function switches is enabled, the other is disabled. However, if you enable both function switches (0 → 1), the length measurement is enabled.

Function switches and jobs can be used concurrently at the call of CAM_CTRL_452.

As with the jobs, the function switches are provided done bits with the name extension _D and error bits with the name extension _ERR.

To be able to evaluate the done and error bits, you should set these bits to 0 when changing a function switch.

Command status

The status of command execution can be read from the return value RET_VAL and activity bit JOBBUSY in the channel DB. You can evaluate the status of a single command based on its trigger, done and error bits.

	RET_VAL	JOBBUSY	Trigger bit _EN	Done bit _D	Error bit _ERR
Command active	1	1	1	0	0
Command completed without errors	0	0	0	1	0

	RET_VAL	JOBBUSY	Trigger bit _EN	Done bit _D	Error bit _ERR
Command completed with errors	-1	0	0	1	1
Write command cancelled	-1	0	0	0	1

Error classes

Error class 1: Operating errors

Description

Operating errors are detected asynchronously to an operator input/control.

No.	Meaning		Diagnostics interrupt
1	Software limit switch start overrun		Yes
2	Software limit switch end overrun		Yes
3	Traversing range start overrun		Yes
4	Traversing range end overrun		Yes
13	Set actual value on-the-fly cannot be executed		Yes
	Cause	After you set the actual value on-the-fly, the software limit switches are outside the traversing range (-100 m to +100 m or -1000 m to +1000 m). The offset derived from the set actual value/set actual value on-the-fly operation is greater than ± 100 m or ± 1000 m.	
	Effect	Axis not synchronized.	

Error class 4: Data error

Description

Data errors are detected synchronously to an operator input/control.

No.	Meaning		Diagnostics interrupt
10	Incorrect zero offset		No
	Cause	The zero offset is greater than ±100 m or ±1000 m. After the zero offset, the software limit switches are outside the traversing range (-100 m to +100 m or -1000 m to +1000 m). Rotary axis: The zero offset amount is greater than the end of the rotary axis.	

No.	Meaning		Diagnostics interrupt
11	Incorrect actual value specification		No
	Cause	Linear axis: The coordinate is outside the current (possibly shifted) software limit switches. Rotary axis: The coordinate is < 0 or greater than the end of the rotary axis.	
12	Incorrect reference position		No
	Cause	Linear axis: The coordinate is outside the current (possibly shifted) software limit switches. Rotary axis: The coordinate is < 0 or greater than the end of the rotary axis.	
20	Enable machine data not permitted		No
	Cause	There is no new (error-free) machine data on the module	
21	Set actual value on-the-fly not permitted		No
	Cause	An attempt was made to execute "Set actual value on-the-fly" while "Retrigger reference point" was enabled.	
27	Invalid bit-coded setting		No
	Cause	Unused and, in this case, unwritten bits do not equal 0. An attempt was made to select "length measurement" and "edge detection" at the same time.	
28	Retrigger reference point is not permitted		No
	Cause	An attempt was made to execute "Retrigger reference point" while "Set actual value on-the-fly" was active. An attempt was made to execute "Retrigger reference point" with an SSI encoder.	
29	Invalid bit-coded command		No
	Cause	Unused and, in this case, unwritten bits do not equal 0.	
30	Incorrect lead time		No
31	Incorrect cam number		No
	Cause	Cam is invalid The cam number is not in the range from 0 to 127.	
32	Incorrect cam start		No
	Cause	The cam start is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m). Rotary axis: Cam start is < 0 or greater than the end of the rotary axis.	
33	Incorrect cam end / cam activation time		No
	Cause	The cam end is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m). Rotary axis: The cam end is < 0 or greater than the end of the rotary axis. The cam is not activated at least for the duration of one pulse. With an inverse cam, there are not at least 4 pulses between the cam start and cam end.	
34	Cancel set actual value not possible		No
	Cause	The actual position value would be outside the working range with an SSI encoder and linear axis after making the setting.	
35	Incorrect actual value specified by "Set actual value" / "Set actual value on-the-fly"		No

No.	Meaning		Diagnostics interrupt
	Cause	The specified actual value is outside the valid number range of ±100 m or ±1000 m. If this setting is made, the software limit switches would be outside the traversing range (-100 m to +100 m or -1000 m to +1000 m). The offset derived from set actual value / set actual value on-the-fly would be greater than ±100 m or ±1000 m.	
107	Axis parameters not assigned		No
	Cause	No machine data on the axis. No machine data activated on the axis.	
108	Axis not synchronized		No
	Cause	One of the settings "Set actual value" or "Set actual value on-the-fly" was initiated although the axis is not synchronized.	
109	Cam processing active.		No
110	Incorrect number of cams to be modified		No

Error class 5: Machine data error

Description

A diagnostics interrupt is only triggered if an error is detected in the system data block (SDB).

No.	Meaning		Diagnostics interrupt
5	Error in hardware interrupt setting		Yes
	Cause	You have attempted to select a hardware interrupt that the module does not support.	
6	Incorrect minimum edge distance		Yes
	Cause	You have entered a value <0 or >10^9 µm as the minimum edge distance	
8	Incorrect axis type		Yes
	Cause	You have specified neither 0, nor 1 as the axis type	
9	Incorrect end of the rotary axis		Yes
	Cause	The value for the end of the rotary axis is outside the valid range from 1 to 10^9 µm or from 1 to 10^8 µm (depending on the resolution).	
10	Incorrect encoder type		Yes
	Cause	The value for the encoder type is outside the valid range from 1 to 10	
11	Incorrect distance / encoder revolution		Yes
	Cause	The value for distance / encoder revolution is outside the valid range from 1 to 10^9 µm (regardless of the resolution).	
13	Incorrect number of increments / encoder revolution		Yes
14	Incorrect number of revolutions		Yes
15	Incorrect transmission rate		Yes
	Cause	You specified a transmission rate outside the valid range from 0 to 3.	

No.	Meaning		Diagnostics interrupt
16	Incorrect reference point coordinate		Yes
	Cause	The coordinate is outside the range from -100 m to +100 m or -1000 m to +1000 m, depending on the resolution. Linear axis: The coordinate is outside the operating range. Rotary axis: The coordinate is < 0 or greater than the end of the rotary axis.	
17	Incorrect absolute encoder adjustment		Yes
	Cause	SSI encoder: The value of the absolute encoder adjustment is not in the encoder range: (increments per encoder revolution x number of revolutions - 1).	
18	Incorrect mode for "Retrigger reference point"		Yes
	Cause	You specified a value outside the valid quantity of 0, 1, 6 and 7.	
19	Incorrect direction adaptation		Yes
	Cause	You have specified a value other than the valid 0 and 1.	
20	Hardware monitoring not possible		Yes
	Cause	You set message frame error monitoring = FALSE at the parameter DB. Encoder does not support error pulse monitoring (disable MON_PULSE)	
21	Incorrect software limit switch start		Yes
	Cause	Linear axis: The software limit switch start is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m, depending on the resolution). Linear axis: The software limit switch start (including any zero offset) is less than -100 m or -1000 m (depending on the resolution).	
22	Incorrect software limit switch end		Yes
	Cause	Linear axis: The software limit switch end is outside the traversing range (-100 m to +100 m or 1000 m to +1000 m, depending on the resolution) or less than the software limit switch start. Linear axis: The software limit switch end (including any zero offset) is greater than +100 m or +1000 m (depending on the resolution).	
144	Incorrect quantity structure		Yes
	Cause	You have specified a quantity structure other than 0 to 3.	
145	Incorrect hysteresis		Yes
	Cause	The hysteresis is outside the range 0 to 65535 × resolution. The hysteresis is greater than ¼ × the working range or ¼ × the rotary axis range.	
146	Incorrect simulation velocity		Yes
	Cause	The simulation velocity is outside the range 1000 × RES to 3×10^7 × RES or is greater than 5×10^8 µm/min. The simulation velocity cannot be set internally.	
147	Incorrect track		Yes
	Cause	Activation of a track outside 0 to 15 (bits 0 to 15) was selected.	
148	Incorrect selection of enable inputs		Yes
	Cause	You wanted to enable a track outside 3 to 10 (bits 0 to 7) using an external signal.	
149	Incorrect selection of special tracks		Yes
	Cause	You wanted to define a track outside 0, 1 and 2 (bits 0, 1 and 2) as a special track	

No.	Meaning		Diagnostics interrupt
150	**Incorrect high count value track 0**		Yes
	Cause	You have specified a count value < 2 or > 65535 as the high count value.	
151	**Incorrect high count value track 1**		Yes
	Cause	You have specified a count value < 2 or > 65535 as the high count value.	
200	**Incorrect resolution**		Yes
	Cause	You have specified a resolution < 0.1 µm/pulse or >1000 µm/pulse. You specified a distance / encoder revolution and a number of pulses / encoder revolutions, that results in a resolution of < 0.1 or > 1000.	
201	**Encoder does not match the operating range / rotary axis range**		Yes
	Cause	SSI encoder and rotary axis: The encoder does not precisely cover the rotary axis range. Linear axis: The encoder does not cover at least the operating range (including software limit switches).	

Error class 7: Cam data errors

Description

A diagnostics interrupt is only triggered if an error is detected in the system data block (SDB).

No.	Meaning		Diagnostics interrupt
1	**Invalid hardware interrupt**		Yes
	Cause	You attempted to define a hardware interrupt for a cam with a cam number > 7.	
2	**Incorrect track number**		Yes
	Cause	The track number is outside the range from 0 to 31	
3	**Incorrect cam start**		Yes
	Cause	The cam start is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m). Rotary axis: Cam start is < 0 or greater than the end of the rotary axis.	
4	**Incorrect cam end**		Yes
	Cause	The cam end is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m). The cam length is not at least 1 pulse. Rotary axis: The cam end is < 0 or greater than the end of the rotary axis. With an inverse cam, there are not at least 4 pulses between the cam start and cam end.	
5	**Incorrect activation time**		Yes
	Cause	The activation time is < 0 µs. The maximum value depends on the quantity structure.	
6	**Incorrect lead time** [1)]		Yes
	Cause	The derivative time is < 0 µs. The maximum value depends on the quantity structure.	

No.	Meaning		Diagnostics interrupt
50	Too many cam records		Yes
	Cause	You wanted to enter more cam records than is possible with this quantity structure.	
51	Axis in operation		Yes
	Cause	You attempted to enter cam records while the cam controller was active.	
52	Axis parameters not assigned		Yes
	Cause	You want to enter cam data although no machine parameters have been activated yet.	

[1] The error message can also be generated if you have assigned the parameter "inverted" as the count direction in connection with an absolute encoder (SSI).

Error class 15: Messages

No.	Meaning		Diagnostics interrupt
1	Start of parameter assignment		No
	Cause	The module has detected a parameter assignment via a system data block.	
2	End of parameter assignment		No
	Cause	The module has completed processing of the parameter assignment via system data block without errors.	

Error class 128: Diagnostics errors

No.	Meaning		Diagnostics interrupt
4	External auxiliary voltage missing		Yes
	Cause	• External 24 V auxiliary voltage is not connected or has failed • Front connector missing • Short-circuit (for example, at the connected encoder)	
	Effect	• Cam processing is disabled • Track outputs is disabled • With incremental encoders, synchronization is canceled • Module parameters not assigned (feedback signal PARA = 0).	
	Remedy	Verify that the 24 V connection is correct (faulty module if 24 V connection is correct).	
5	Front connector missing (FM 452)		Yes
	Cause	Front connector not inserted	
	Effect	• Missing external auxiliary 24 V voltage • Module not ready	
	Remedy	Insert the front connector	
51	Timeout monitoring responded (watchdog)		Yes

No.	Meaning		Diagnostics interrupt
	Cause	• Module exposed to severe interference • Module error	
	Effect	• Module is reset • Provided that no module error response is detected after the module reset, the module is ready for operation again. • The module signals the watchdog timeout with "incoming" and "outgoing"	
	Remedy	• Eliminate the interference • Contact the relevant sales department, which will require exact details about the circumstances leading to the error. • Replace the module	
52	**Internal power supply of the module failed**		Yes
	Cause	Module error	
	Effect	• Module is reset • Provided that no module error response is detected after the module reset, the module is ready for operation again.	
	Remedy	Replace the module	
70	**Hardware interrupt lost**		Yes
	Cause	The module detected a hardware interrupt event that cannot be reported, because the same event has not yet been processed by the user program/ CPU.	
	Effect	• Cam processing is disabled • Track outputs is disabled • With incremental encoders, synchronization is canceled	
	Remedy	• Integrate the hardware interrupt OB in the user program • Check the module's bus connection • Disable the hardware interrupt • Adapt your hardware and software to suit your process requirements (for example, faster CPU, optimize user program).	
144	**Encoder wire break**		Yes
	Cause	• Encoder cable not inserted or sheared off • Encoder has no transverse signals • Incorrect pin assignment • Cable length too long • Short-circuit of the encoder signals	
	Effect	• Cam processing is disabled • Track outputs is disabled • With incremental encoders, synchronization is canceled	
	Remedy	• Check the encoder cable • Comply with the encoder specifications • Monitoring can be temporarily disabled at the responsibility of the operator using the programming interface. • Comply with the technical specifications of the module	
145	**Absolute encoder message frame error**		Yes

No.	Meaning		Diagnostics interrupt
	Cause	Incorrect or interrupted message frame traffic between the module and the absolute encoder (SSI): • Encoder cable not inserted or sheared off • Incorrect encoder type • Incorrect encoder setting (programmable encoders) • Invalid message frame length • Encoder returns incorrect values (faulty encoder) • Interference on the measuring system cable • Selected transmission rate too high	
	Effect	• Cam processing is disabled • Track outputs is disabled • The last valid actual value remains unchanged until the end of the next correct SSI transfer	
	Remedy	• Check the encoder cable • Check the encoder • Check the message frame traffic between the encoder and module	
146	Incremental encoder missing pulses		Yes
	Cause	• Encoder monitoring has detected missing pulses • Incorrect entry of the number of increments per encoder revolution • Encoder failure: does not return the specified number of pulses • Incorrect or missing zero mark • Interference on the encoder cable	
	Effect	• Cam processing is disabled • Track outputs is disabled • Cancelation of synchronization	
	Remedy	• Enter the correct number of increments / encoder revolutions • Check the encoder and its cable • Comply with shielding and grounding rules • Monitoring can be temporarily disabled at the responsibility of the operator using the programming interface.	

FM 355 PID Control

PID_FM

Description PID_FM

During operation, the instruction PID_FM is used to:

- Transfer operating parameters to FM 355
- Transfer controlller parameters to FM 355
- Read process values from FM 355

Operating parameters are all in/out parameters that lie between the op_par and cont_par parameters in the instance data block of the instruction.

Controller parameters are all in/out parameters that are located after the cont_par parameter in the instance data block of the instruction.

Process values are all output parameters of the instruction after the out_par parameter.

The required data are stored in an instance DB on the CPU. You need one instance DB for each controller channel used. All the in/out parameters are set to FALSE after an instance DB has been created.

Call

PID_FM is normally called in the cyclic interrupt OB 35 and requires an initialization routine that is started by setting the parameter COM_RST = TRUE in the start-up of the CPU. A calling of the instruction in the start-up OB is possible, but not necessary. After the initialization routine, PID_FM sets the parameter COM_RST to FALSE. If FM 355 is used in distributed I/Os, this can take a few call cycles.

PID_FM has to be called in the same OB as all the other instructions that access the same FM 355.

If you call the instruction PID_FM as a multiple instance DB, no commissioning interface is available. In this case you must commission the FM 355 via a watch table.

Start-up

During CPU start-up and during CPU STOP-RUN transition, the parameters on the FM 355 are overwritten with the parameters from the SDB of the CPU.

Responses in the event of an error

The RET_VALU output parameter contains the 2nd and 3rd bytes of the STATUS parameter of the RDREC and WRREC instructions. RET_VALU can be evaluated if the READ_VAR and LOAD_PAR parameters are not reset.

When the instruction PID_FM is called, an I/O access error can occur if the FM 355 is not plugged or is not supplied with voltage. In this case the CPU goes into STOP, if no OB 122 is loaded in the CPU.

See also

Mode of operation PID_FM (Page 7187)

Changing parameters with the HMI (Page 9858)

Creating and supplying instance DBs (Page 9851)

Mode of operation PID_FM

Effect of parameters LOAD_OP, READ_VAR and LOAD_PAR

In the initialization routine PID_FM reads the parameters from the system data and stores them in its instance DB. Instance DB and system data are synchronized.

The following table shows how the function of the instruction is influenced by the parameters LOAD_OP, READ_VAR and LOAD_PAR.

Parameters	TRUE	FALSE
LOAD_OP	Transfer operating parameters with the instructions WRREC and RDREC.	Transfer operating parameters via direct I/O access in muliplex mode.
	READ_VAR is set to TRUE	READ_VAR is not changed
	Longer run time	Shorter run time
	The values are effective after a cycle of the CPU or FM. The longer cycle is decisive.	A maximum of 3 cycles of the CPU or PM are required until the values are effective. The longer cycle is decisive
READ_VAR	Transfer process values with the instructions WRREC and RDREC.	Process values are transferred via direct I/O accesses in multplex mode.
	Longer processing time	Shorter processing time
	All parameters are read from FM 355	The following parameters are **nicht** read from FM 355 • SP (setpoint from the FM) • ER (error signal) • DISV (disturbance variable) • LMN_A • LMN_B
	Following parameters are updated at **every** call of the instruction. • PV (process value) • LMN (manipulated value) • The binary displays	Following parameters are updated at every **fourth** call of the instruction. • PV (process value) • LMN (manipulated value) • The binary displays
	At the start-up of the CPU, the following values are applied from the FM 355 to the PID_FM. • SP_OP • LMN_OP • SP_OP_ON • LMNOP_ON If these values were changed on the CPU, they are overwritten with the values from the FM 355 at the next start-up of the CPU	At the start-up of the CPU, the following values are read from the instruction, not from the FM: • SP_OP • LMN_OP • SP_OP_ON • LMNOP_ON
LOAD_PAR	All controller parameters are downloaded from the CPU to the FM 355. This is practical if the controller parameters are to be changed depending on process states.	No transfer of the controller parameters

After data transfer as been completed the parameters LOAD_OP, READ_VAR and LOAD_PAR are reset to FALSE.

Use in distributed I/O

When the FM 355 is used in distributed I/O, the data transfer and the resetting of the parameters can take a few call cycles.

Special features during transfer in multiplex mode

Multiplexing of the data to be transferred during access to the FM 355 via direct I/O accesses is controlled via the PID_FM. This multiplex controlling via PID_FM does not function if two instances of the instruction PID_FM access the same channel number of a module. This results in incorrect parameters in the FM 355 (for example, setpoint and manual manipulated value) and incorrect displays at the output parameters of the instruction PID_FM.

Saving the Parameters in EEPROM

During program controlled re-configuring (LOAD_PAR, LOAD_OP) of the controller module by the instruction PID_FM, its run time is extended. The new parameters are always immediately effective. Controller parameters and the operating parameter SP_ER are stored additionally in a nonvolatile memory (EEPROM). After each saving of the parameters in the EEPROM, a renewed saving is delayed by 30 minutes, since the life of the EEPROM is limited by the number of write processes. It is possible to save the new parameter in the EEPROM as soon as the power supply returns. The selection of the parameters determines whether the reconfiguration of the controller module by the instruction PID_FM is executed without bumps.

Processing times PID_FM

Boundary conditions			Processing time in	
READ_VAR	LOAD_OP	LOAD_PAR	CPU 314	CPU 414-2DP
FALSE	FALSE	FALSE	0.65 ms	0.077 ms
TRUE	FALSE	FALSE	2.85 ms	2.36 ms
TRUE	TRUE	FALSE	4.56 ms	4.48 ms
FALSE	FALSE	TRUE	3.75 ms	2.59 ms
TRUE	FALSE	TRUE	5.95 ms	5.15 ms
TRUE	TRUE	TRUE	7.66 ms	7.1 ms

See also

Description PID_FM (Page 7185)

Changing parameters with the HMI (Page 9858)

Input parameters PID_FM

Table 9-213

Parameters	Address	Data type	Default	Description
MOD_ADDR	0.0	INT	256	The module address that results from the configuration with STEP 7 is set at this input.
CHANNEL	2.0	INT	1	The number of the controller channel to which the instance DB îs referenced is configured at the "Channel number" input. Values of 1 to 4 are permissible.
PHASE	4.0	INT	0	The parameter PHASE can be interconnected with the output parameter PHASE of the instructions TUN_EC or TUN_ES (PID Self-Tuner). The phase state of the PID Self-Tuner can then be displayed in plain text in the "Commissioning" dialog. This parameter is not relevant for the OP.

See also

Description PID_FM (Page 7185)

Output parameters PID_FM

Table 9-214

Parameters	Address	Data type	Default	Description
RET_VALU	6.0	WORD	0	RET_VALU contains the 2nd and 3rd bytes from the STATUS parameter of the RDREC and WRREC instructions. RET_VALU can be evaluated when an error is reported via QMOD_F.
out_par	8.0	WORD	W#16#3130	The out_par parameter may not be modified by users. It marks the start of the output parameter that is read by the module if READ_VAR = TRUE is set.
SP	10.0	REAL	0.0	The setpoint that is currently in effect is available at the "Setpoint" output.
PV	14.0	REAL	0.0	The process value that becomes effective is issued on the "Process value" output.
ER	18.0	REAL	0.0	The control deviation that becomes effective is issued on the "Control deviation" output.
DISV	22.0	REAL	0.0	The interference that becomes effective is issued on the "Interference" output. Values from -100 to 100 % are permitted.
LMN	26.0	REAL	0.0	The manipulated value that becomes effective is issued on the "Manipulated value" output. Step controllers without analog position feedback output the unlimited P- + D-action at the LMN parameter. Values from -100 to 100 % are permitted.
LMN_A	30.0	REAL	0.0	At the output "Manipulated value A of the split range function / position feedback", continuous controllers display the manipulated value A and step controllers with analog position feedback display the position feedback. The output LMN_A can only be used for an approximate display of a corresponding simulated manipulated variable. The start value LMNRSVAL of the simulated position feedback has to be configured accordingly and becomes effective when LMNRS_ON is set. Values from -100 to 100 % are permitted.

Parameters	Address	Data type	Default	Description
LMN_B	34.0	REAL	0.0	Continuous controllers display the manipulated value B of the split-range function at the output “manipulated value B of the split-range function”. Values from -100 to 100 % are permitted.
QH_ALM	38.0	BOOL	FALSE	The process value or the controlled variable is monitored for four limits. Violation of H_ALM limit is reported at the output "High limit alarm triggered".
QH_WRN	38.1	BOOL	FALSE	The process value or the controlled variable is monitored for four limits. Violation of H_WRN limit is reported at the output "High limit warning triggered".
QL_WRN	38.2	BOOL	FALSE	The process value or the controlled variable is monitored for four limits. Violation of the L_WRN limit is reported at output "Low limit warning triggered".
QL_ALM	38.3	BOOL	FALSE	The process value or the controlled variable is monitored for four limits. Violation of the L_ALM limit is reported at output "Low limit alarm triggered".
QLMN_HLM	38.4	BOOL	FALSE	The manipulated value is always limited to a high and a low limit. Violation of the high limit is reported at output "High limit of manipulated value triggered". (Does not apply to step controllers without analog position feedback)
QLMN_LLM	38.5	BOOL	FALSE	The manipulated value is always limited to a high and a low limit. Violation of the low limit is reported at output "Low limit of manipulated value triggered". (Does not apply to step controllers without analog position feedback)
QPARA_F	38.6	BOOL	FALSE	The module checks the validity of the parameters. A parameter assignment error is displayed at the "Parameter assignment error" output. These parameter assignment errors can also be read out in the module's "Online & Diagnostics" dialog.
QCH_F	38.7	BOOL	FALSE	The "Channel error" output is set if the controller channel cannot provide valid results. A channel error (for example, wire break) is also set if QPARA_F = 1 or QMOD_F = 1. If QCH_F = TRUE, then the precise error information in the diagnostic data record DS1 of the module is read out.
QUPRLM	39.0	BOOL	FALSE	The setpoint is limited to a positive and negative slew rate. If the output "Limit of positive setpoint slew rate triggered" is set, the positive setpoint slew rate is limited.
QDNRLM	39.1	BOOL	FALSE	The setpoint is limited to a positive and negative slew rate. If the output "Limit of negative setpoint slew rate triggered" is set, the setpoint fall is limited.
QSP_HLM	39.2	BOOL	FALSE	The setpoint is always limited to a high and a low limit. The output "High limit of setpoint value triggered" indicates that the high limit has been exceeded.
QSP_LLM	39.3	BOOL	FALSE	The setpoint is always limited to a high and a low limit. Violation of the low limit is reported at output "Setpoint low limit triggered".
QLMNUP	39.4	BOOL	FALSE	This is the "Manipulated value signal up" output. (Only in the case of step controllers or pulse controllers)
QLMNDN	39.5	BOOL	FALSE	This is the "Manipulated value signal down" output. (Only in the case of step controllers or pulse controllers)
QID	39.6	BOOL	FALSE	QID = TRUE indicates that an identification is running (not that it is enabled). On completion of the identification, the identification result can be read out from the IDSTATUS parameter of the instruction CH_DIAG.
QSPOPON	40.0	BOOL	FALSE	Output "Setpoint operation on" indicates whether the operator controls the setpoint using the instruction TUN_EC or TUN_ES. If the bit is set, the value SP_OP is applied as setpoint.
QLMNSAFE	40.1	BOOL	FALSE	If the output "Safety mode" is set, the safety manipulated value is output as the manipulated value.
QLMNOPON	40.2	BOOL	FALSE	Output "Setpoint operation on" indicates whether the operator controls the manipulated value using the instruction TUN_EC or TUN_ES. If the bit is set, the value LMN_OP is applied as manipulated value.

Parameters	Address	Data type	Default	Description
QLMNTRK	40.3	BOOL	FALSE	The output "Follow-up mode" indicates if the manipulated value is being adjusted via an analog input.
QLMN_RE	40.4	BOOL	FALSE	The output "Manual = 1; Automatic = 0" indicates whether or not the manipulated value is set on the external manipulated value LMN_RE (manual = 1).
QLMNR_HS	40.5	BOOL	FALSE	The output "High endstop signal of position feedback" indicates if the control valve is at its high endstop. QLMNR_HS = TRUE means: The control valve is at high endstop. (For step controllers only)
QLMNR_LS	40.6	BOOL	FALSE	The output "Low endstop signal of position feedback" indicates whether the control valve is at its low endstop. QLMNR_LS = TRUE means: The control valve is at its low endstop. (For step controllers only)
QLMNR_ON	40.7	BOOL	FALSE	Output "Position feedback enabled" indicates the set operating mode "Step controller with position feedback" or "Step controller without position feedback".
QFUZZY	41.0	BOOL	FALSE	If the output parameter is QFUZZY =1, the controller operates with the fuzzy algorithm.
QSPLEPV	41.1	BOOL	FALSE	The output "Display of FUZZY controller: Setpoint < process value" is set when the fuzzy controller is switched on, if the setpoint is less than the effective process value.
QSPR	41.2	BOOL	FALSE	If the output "Split-range operation" is set, the continuous controller is operating in split-range mode.
QMAN_FC	41.4	BOOL	FALSE	The output "QMAN_FC" is set in the following two cases: The slave controller is in manual mode and the master controller is followed up to the process value of the slave controller. The I-action of the master controller is stopped because the setpoint or manipulated variable of the slave controller is limited or because the slave controller is in manual mode.
QPARABUB	41.7	BOOL	FALSE	This parameter is set by the FM if the operating parameters are changed via the OP. If READ_VAR = TRUE and if this display is set by the FM, the instruction PID_FM reads out the parameters SP_OP_ON, LMNOP_ON, SP_OP and LMN_OP from the FM and saves them in the instance DB. The instruction thus applies over the operating state of the FM. After the reading process the parameter is set to FALSE.
QMOD_F	42.0	BOOL	FALSE	The instruction checks that the reading and writing of a data record has been completed. Output "Module error" is set if errors are found. The error cause can be: An incorrect module address at parameter MOD_ADDR, or an incorrect channel number at parameter CHANNEL, or a faulty module.

See also

RDREC: Read data record (Page 6459)

WRREC: Write data record (Page 6461)

In/out parameters PID_FM

Table 9-215

Parameters	Address	Data type	Default	Description
[1] Default values of the module after the first start-up of the PID_FM with COM_RST = TRUE				
COM_RST	44.0	BOOL	FALSE	If the parameter COM_RST = TRUE, the instruction PID_FM executes an initialization routine. In doing so the controller parameters (these are all the parameters after cont_par) are read from the FM and stored in the instance DB. In addition, the parameters MOD_ADDR and CHANNEL are checked for validity. After the initialization routine the parameter is set to FALSE.
LOAD_OP	44.1	BOOL	FALSE	If the in/out parameter "Load operating parameters to FM 355" is set, the operating parameters are loaded into the module and the in/out parameter is reset.
READ_VAR	44.2	BOOL	FALSE	If the in/out parameter "Read tags from FM 355" is set, the output parameters are read from the module and the in/out parameter is reset.
LOAD_PAR	44.3	BOOL	FALSE	If the in/out parameter "Load control parameters to FM 355" is set, the control parameters are loaded into the module and the in/out parameter is reset.
op_par	46.0	WORD	W#16#3130[1]	The op_par parameter may not be modified by users. It identifies the start of the operating parameters that are transferred to the module, if LOAD_OP = TRUE is set. The end of the operating parameters is identified by cont_par.
SP_RE	48.0	REAL	0.0	An external setpoint is interconnected with the controller at the "External setpoint" input.
LMN_RE	52.0	REAL	0.0	An external manipulated value is interconnected to the controller at the input "External manipulated value". Values from -100.0 to 100.0 (%) are permitted.
SP_OP_ON	56.0	BOOL	FALSE	The instructions of PID Self-Tuner have access to the in/out parameter "Enable setpoint operation". If the bit is set, the value SP_OP is applied as setpoint.
SAFE_ON	56.1	BOOL	FALSE	If the "Adopt safety position" input is set, a safety value is applied as manipulated value. Note: The actuating signal processing via LMNDN_OP, LMNUP_OP and LMNSOPON on a step controller is of higher priority than the safety manipulated value.
LMNOP_ON	56.2	BOOL	FALSE	The instructions of PID Self-Tuner have access to the in/out parameter "Enable manipulated value operation". If the bit is set, the value LMN_OP is applied as manipulated value.
LMNTRKON	56.3	BOOL	FALSE	If input "Follow-up (LMN via AI)" is set, the manipulated value is adjusted to an analog input (AI). (Does not apply to step controllers without analog position feedback)
LMN_REON	56.4	BOOL	FALSE	If input "Enable external manipulated value" is set, the external manipulated value LMN_RE is set as the manipulated value.
LMNRHSRE	56.5	BOOL	FALSE	The signal "Control valve at high endstop" is interconnected at the "High endstop signal of position feedback" input. LMNRHSRE = TRUE means: The control valve is located at the high endstop. (Only in the case of step controllers)
LMNRLSRE	56.6	BOOL	FALSE	The signal "Control valve at low endstop" is interconnected at the "Low endstop signal of position feedback" input. LMNRLSRE = TRUE means: The control valve is located at the low endstop. (Only in the case of step controllers)

Parameters	Address	Data type	Default	Description
LMNSOPON	56.7	BOOL	FALSE	The instructions of PID Self-Tuner have access to the input "Enable manipulated value operation". If the bit is set, the signals LMNUP_OP and LMNDN_OP are applied as manipulated value signals. (Only in the case of step controllers)
LMNUP_OP	57.0	BOOL	FALSE	The instructions of PID Self-Tuner have access to the input "Manipulated value signal up". If LMNSOPON is set, the value at input "Manipulated value signal up operation" is set as the manipulated value signal. (Only in the case of step controllers)
LMNDN_OP	57.1	BOOL	FALSE	The instructions of PID Self-Tuner have access to the input "Manipulated value signal down operation". If LMNSOPON is set, the value at input "Manipulated value signal down operation" is set as the manipulated value signal. (Only in the case of step controllers)
LMNRS_ON	57.3	BOOL	FALSE	You can simulate no position feedback if this is not available. The function is enabled by setting the "Enable simulation of position feedback" input The instructions of PID Self-Tuner have access to this parameter as well since at least one simulated manipulated value is required for optimization if a step controller was configured without position feedback. The simulated value is displayed at parameter LMN_A. When the simulation is enabled the value of the parameter LMNRSVAL is set as the starting value. CAUTION: Offset between the simulation value and real position feedback increases over the time. (Only for step controllers without analog position feedback)
FUZID_ON	57.4	BOOL	FALSE	The identification of the fuzzy algorithm is enabled at the "Enable fuzzy identification" input.
SP_OP	58.0	REAL	0.0	The instructions of PID Self-Tuner have access to the in/out parameter "Setpoint operation". If the SP_OP_ON bit is set, the "Setpoint operation" value is applied as the setpoint.
LMN_OP	62.0	REAL	0.0	The instructions of PID Self-Tuner have access to the in/out parameter "Manipulated value operation". If the LMNOP_ON bit is set, the "Manipulated value operation" value is applied as the manipulated value. Values from -100.0 to 100.0 (%) are permitted
LMNRSVAL	66.0	REAL	0.0	The instructions of PID Self-Tuner have access to input "Starting value of simulated position feedback". The start value of the simulation is entered at the parameter. (Only for step controllers without analog position feedback) Values from -100.0 to 100.0 (%) are permitted
cont_par	70.0	WORD	W#16#3130 [1]	The cont_par parameter may not be modified by users. It indicates the start of the controller parameters which are read from the FM and stored in the instance DB, if COM_RST = TRUE is, and which are transferred to the FM when LOAD_PAR = TRUE is set. The end of the controller parameter is the end of the instance DB.
P_SEL	72.0	BOOL	TRUE [1]	In the PID algorithm, the PID parts can be switched on and off individually. The proportional action is enabled when the "P-action on" input is set.
PFDB_SEL	72.1	BOOL	FALSE [1]	In the PID algorithm, the P- and D-actions can be included in the feedback loop. The proportional action is located in the feedback loop when the input "Enable P-action in feedback loop" is set.
MONERSEL	72.2	BOOL	FALSE [1]	The controller has a limit indicator that can be applied either for the process value or the control deviation. If the input "Monitoring: Process value = 0, control deviation = 1" is set, the control deviation will be monitored.

Parameters	Address	Data type	Default	Description
D_EL_SEL	74.0	INT	0 [1]	The D-action element in the PID algorithm can be positioned at a separate input. This is selected at the input "D-action element input". 0: Control deviation 1 to 4: Analog input 1 to 4 17: Negative process value, D-action in the feedback
SP_HLM	76.0	REAL	100.0 [1]	The setpoint is always limited to a high and a low limit. The "Setpoint high limit" input specifies the high limit. SP_HLM > SP_LLM
SP_LLM	80.0	REAL	0.0 [1]	The setpoint is always limited to a high and a low limit. The "Setpoint low limit" input specifies the low limit. SP_LLM < SP_HLM
H_ALM	84.0	REAL	100.0 [1]	You can assign parameters for four limits for monitoring the process value or the control deviation. The "High limit interrupt" input specifies the highest limit. H_ALM > H_WRN
H_WRN	88.0	REAL	90.0 [1]	You can assign parameters for four limits for monitoring the process value or the control deviation. The "High limit warning" input specifies the second highest limit. Values from H_ALM to L_WRN are permitted.
L_WRN	92.0	REAL	10.0 [1]	You can assign parameters for four limits for monitoring the process value or the control deviation. The "Low limit warning" input specifies the second lowest limit. Values from H_WRN to L_ALM are permitted.
L_ALM	96.0	REAL	0.0 [1]	You can assign parameters for four limits for monitoring the process value or the control deviation. The "Low limit alarm" input specifies the lowest limit. L_ALM < L_WRN
HYS	100.0	REAL	1.0 [1]	To prevent flickering of the monitoring displays, a hysteresis can be configured at the "Hysteresis" input. HYS ≥ 0.0
DEADB_W	104.0	REAL	0.0 [1]	A deadband is applied to the control deviation. The "Deadband width" input determines the size of the deadband. DEADB_W ≥ 0.0
GAIN	108.0	REAL	1.0 [1]	The "Proportional gain" input specifies controller amplification.
TI	112.0	REAL	3000.0 [1]	The "Integration time" input determines the time response of the integral action. If TI = 0, the integral action is disabled TI = 0.0 or TI ≥ 0.5
TD	116.0	REAL	0.0 [1]	The "Derivative-action time" input determines the time response of the differentiator. If TD = 0, the derivative action is disabled. TD = 0.0 or TD ≥ 1.0
TM_LAG	120.0	REAL	5.0 [1]	The algorithm of the derivative action contains a delay for which parameters can be assigned at the input "Time lag of the derivative action". TM_LAG ≥ 0.5
LMN_SAFE	124.0	REAL	0.0 [1]	For the manipulated value, a safety value can be configured at the "Safety manipulated value" input. Values from -100.0 to 100.0 (%) are permitted.

Parameters	Address	Data type	Default	Description
LMN_HLM	128.0	REAL	100.0 1)	The manipulated value is always limited to a high and a low limit. The "Manipulated value high limit" input specifies the high limit. (Does not apply to step controllers without analog position feedback) Values from LMN_LLM to 100.0 (%) are permitted.
LMN_LLM	132.0	REAL	0.0 1)	The manipulated value is always limited to a high and a low limit. The "Manipulated value low limit" input specifies the low limit. (Does not apply to step controllers without analog position feedback) Values from -100.0 to LMN_HLM (%) are permitted.
MTR_TM	136.0	REAL	60.0 1)	The runtime from endstop to endstop of the control valve is entered at the "Motor actuating time" parameter. (Only in the case of step controllers) Values of MTR_TM ≥ 0.001 are permitted.
PULSE_TM	140.0	REAL	0.2 1)	A minimum pulse time can be configured at the "Minimum pulse time" parameter. (Applies to step controllers or pulse controllers only) Values of PULSE_TM ≥ 0.0 are permitted.
BREAK_TM	144.0	REAL	0.2 1)	A minimum pulse break duration can be assigned with the parameter "Minimum break time." (Applies to step controllers or pulse controllers only) Values of BREAK_TM ≥ 0.0 are permitted.

Note

If LOAD_PAR = TRUE is set, all the control parameters are loaded permanently to the EEPROM of the FM 355.

If LOAD_OP = TRUE, only the setpoint SP_RE of the operating parameters is loaded permanently to the EEPROM of the FM 355. All the other operating parameters are assigned the values 0 or FALSE as default during the startup of FM 355

The EEPROM of the module could become damaged by too frequent write operations. To prevent this, the module delays another write process by 30 minutes after writing of the EEPROM.

See also

Description PID_FM (Page 7185)

Block diagram PID_FM

Overview

The following figures show at which points the parameters of the instruction PID_FM act.

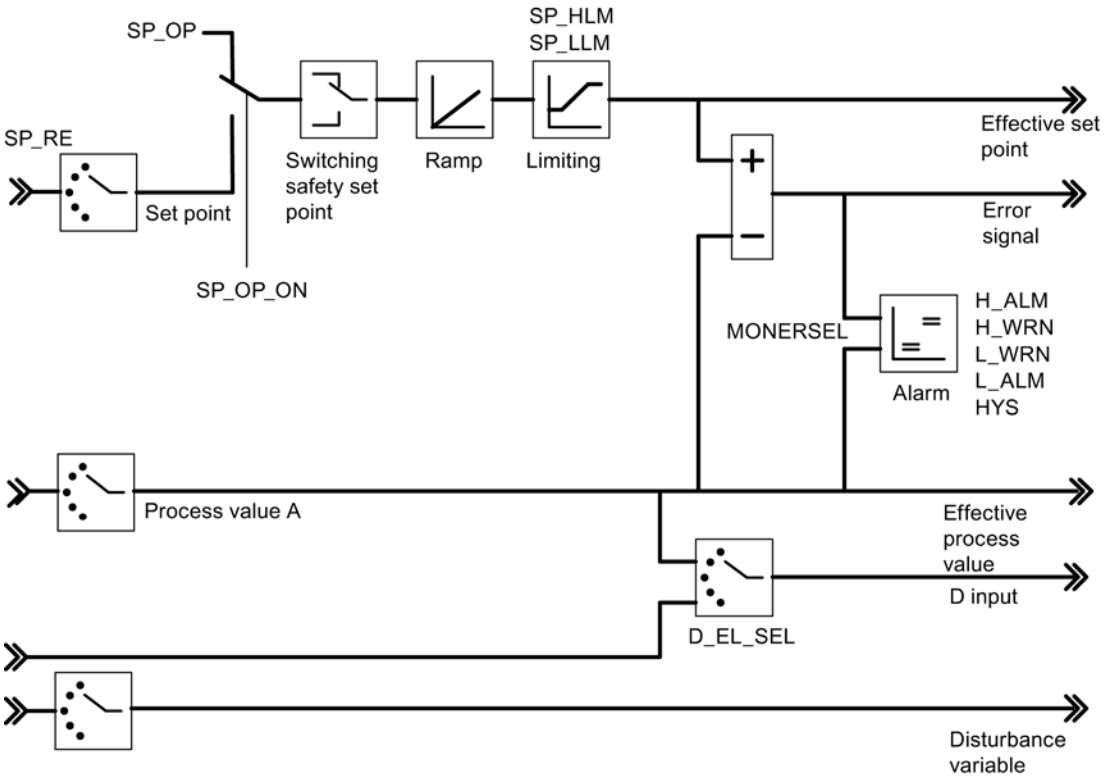
The parameters are effective with three-component controllers, ration/blending controllers, and fixed setpoint or cascade controllers. This also applies for the parameters that exist equally

at continuous controllers, at controllers with a pulse output as well as at step controllers. Therefore, not all block diagrams are depicted and not all parameter are marked in all figures.

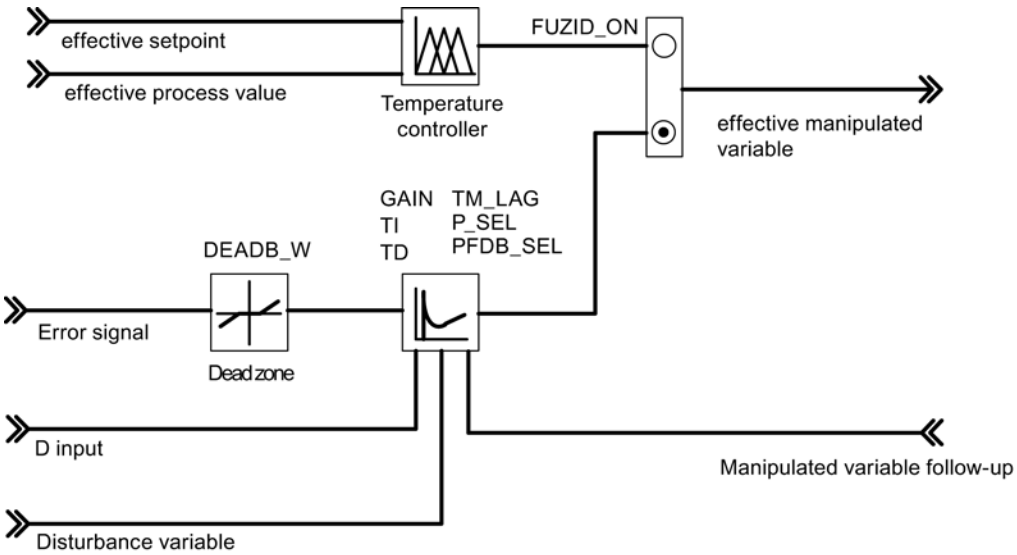
However, the parameters of the instruction PID_FM are contained in all the figures – with the exception of the parameters MOD_ADDR, CHANNEL, QMOD_F, QPARA_F, QCH_F, QLMNR_ON, RET_VALU, COM_RST, LOAD_PAR, READ_VAR, LOAD_OP.

Action of the input parameters

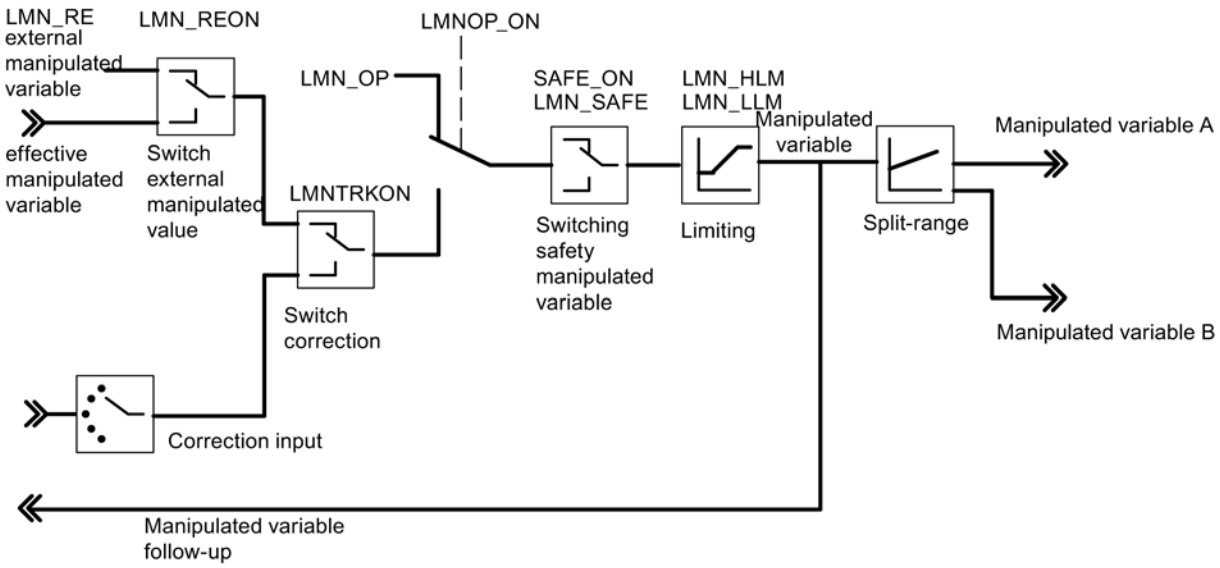
Error signal generation with fixed setpoint or cascade controllers



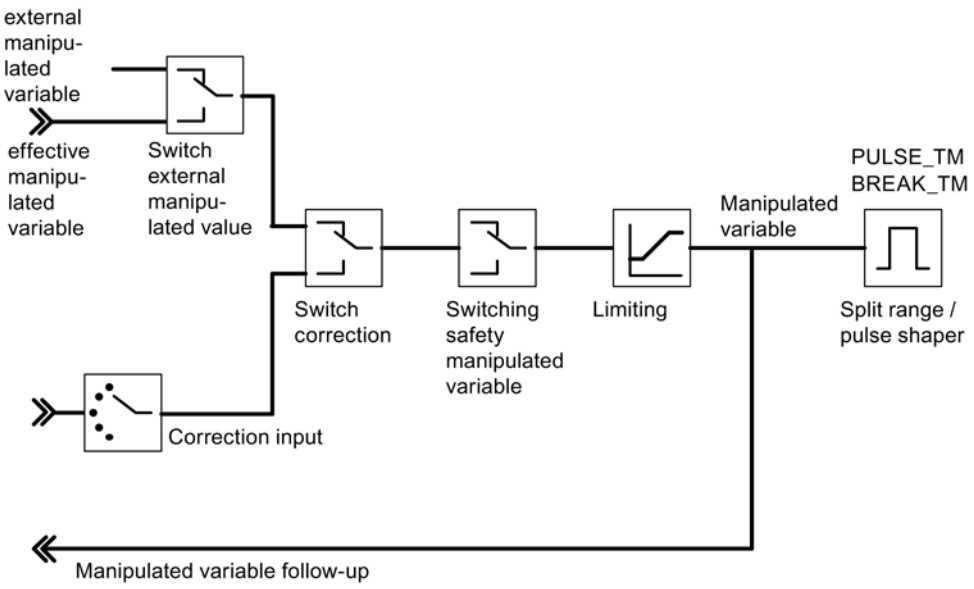
Block diagram of the control algorithm



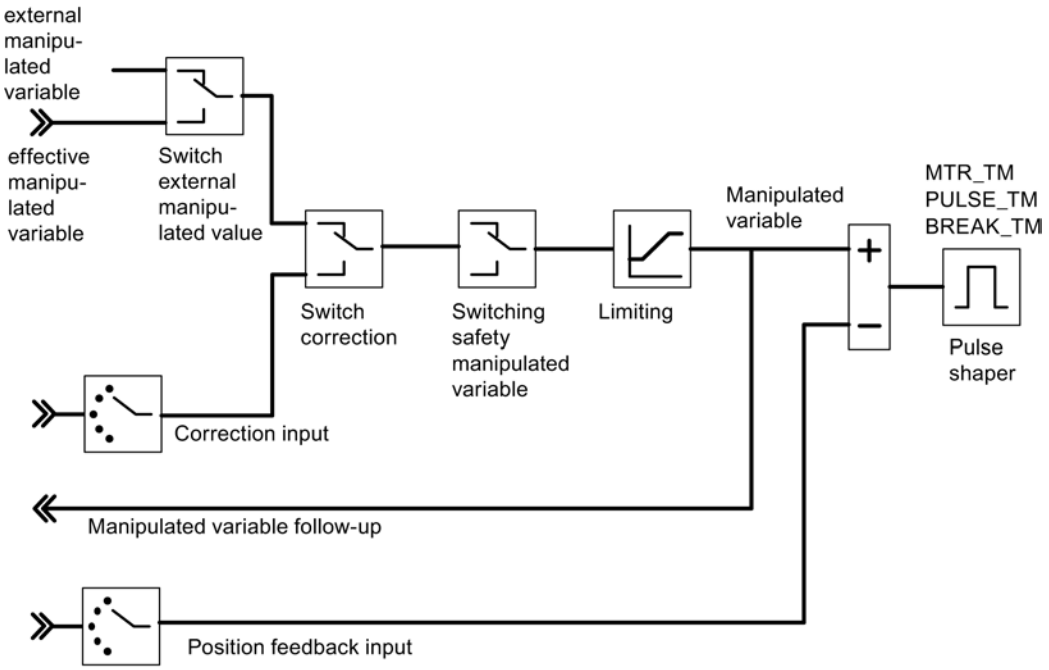
Controller output of the continuous controller



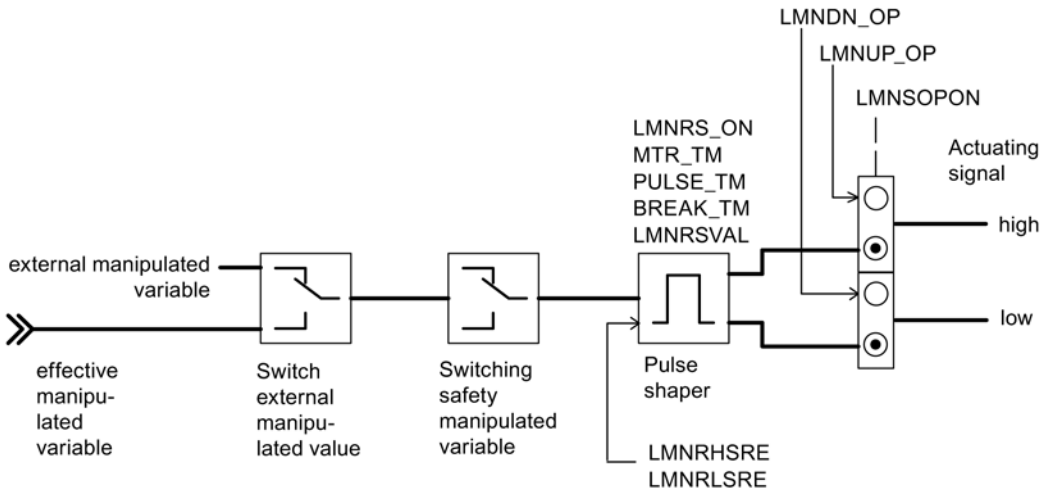
Controller output of the step controller (pulse controller operating mode)



Controller output of the step controller (step controller operating mode with position feedback)



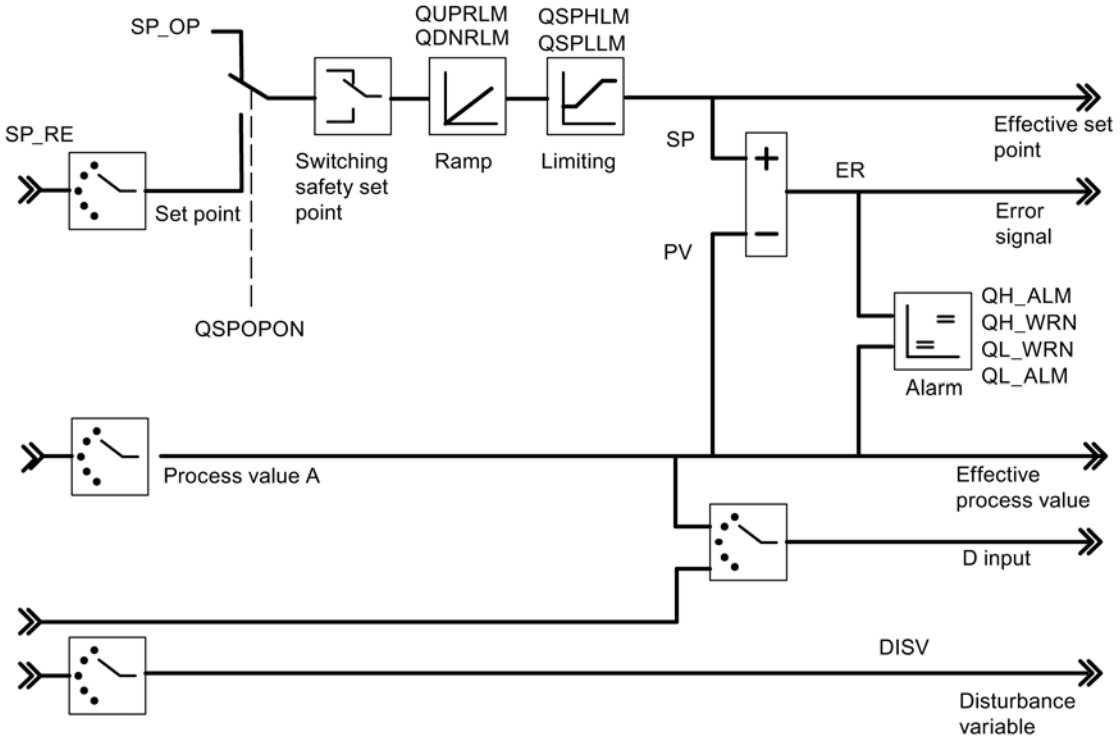
Controller output of the step controller (step controller operating mode without position feedback)



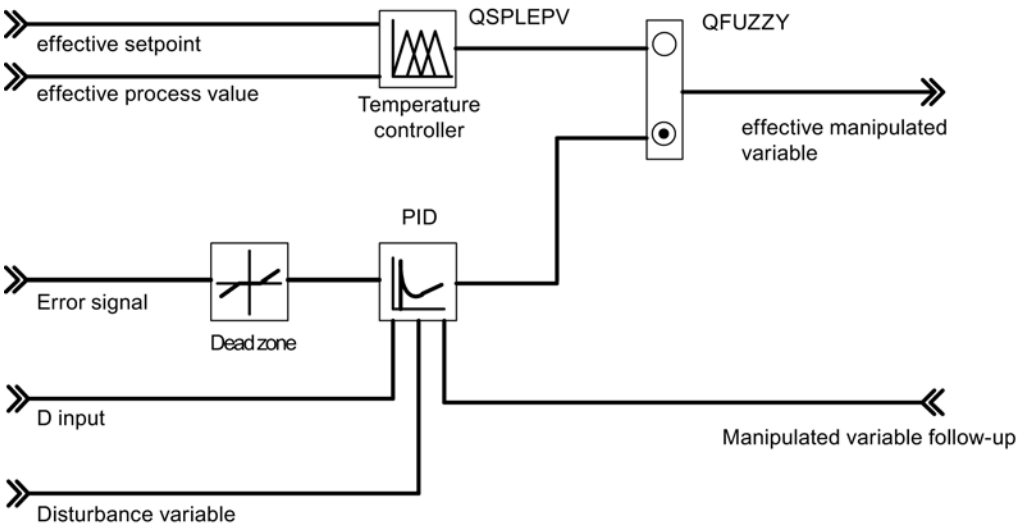
Generating output parameters

The following figures show at which points in the module the output parameters of the instruction PID_FMare generated.

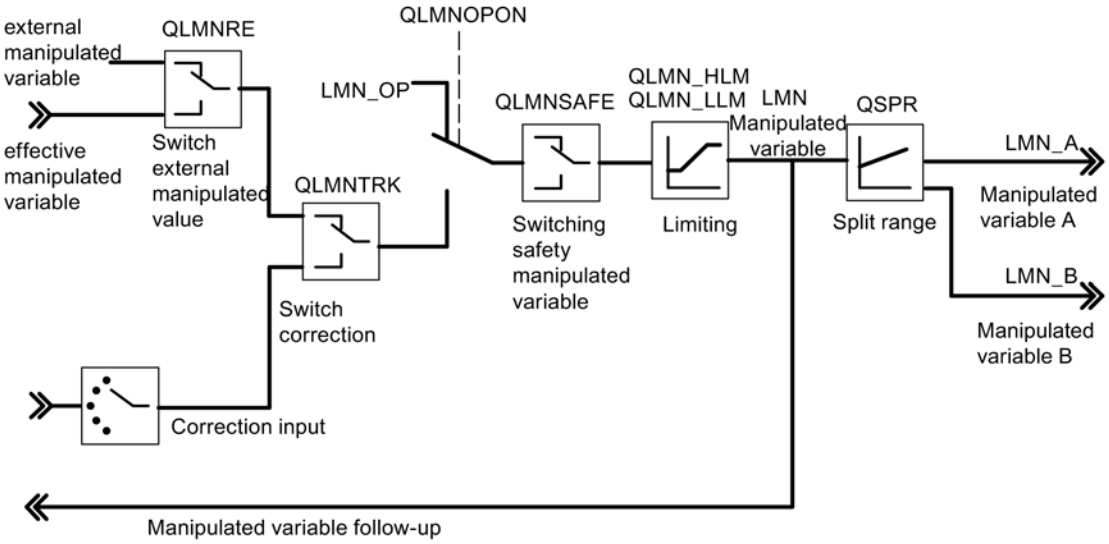
Error signal generation with fixed setpoint or cascade controllers



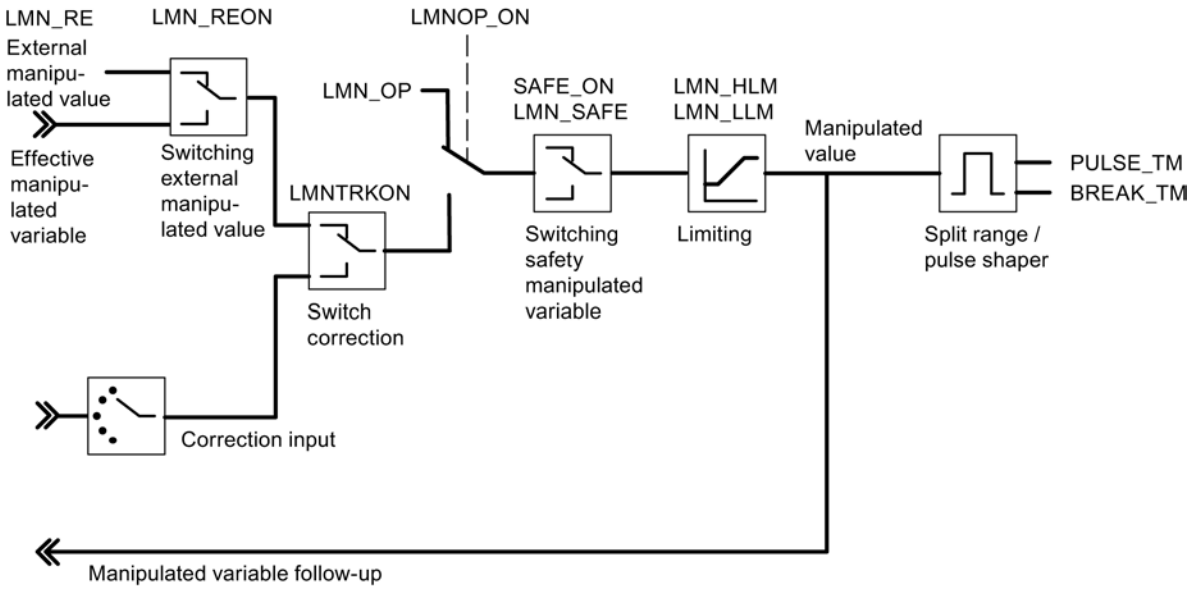
Block diagram of the control algorithm



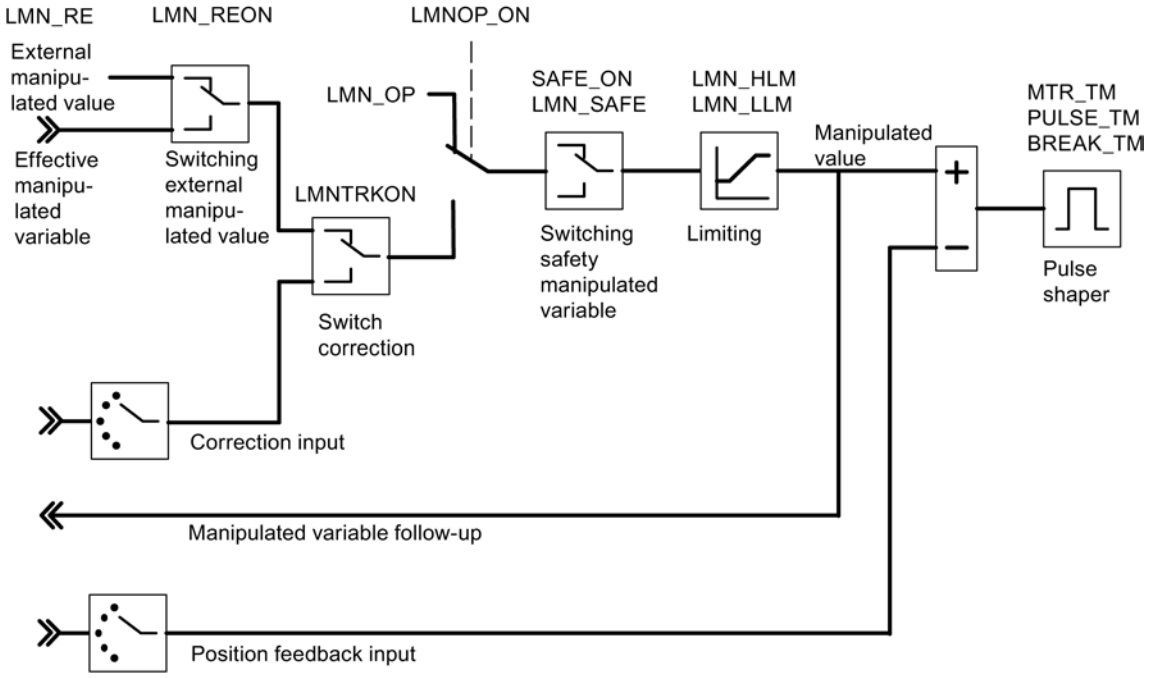
Controller output of the continuous controller



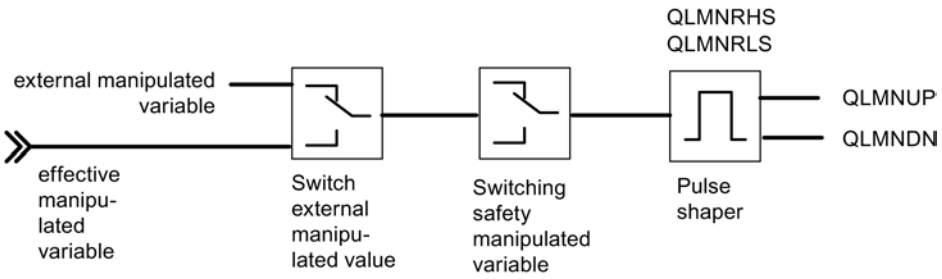
Controller output of the step controller (pulse controller operating mode)



Controller output of the step controller (step controller operating mode with position feedback)



Controller output of the step controller (step controller operating mode without position feedback)



See also

Description PID_FM (Page 7185)

FUZ_355

Description FUZ_355

The instruction FUZ_355 is used to read out and write the controller parameters of the fuzzy temperature controller from FM 355.

This function is suitable for the following applications:

- Transferring the controller parameters of the FM 355 that have been established by identification after the module replacement
- Adapting the FM 355 to different processes

Note

It is not permitted to modify the parameters that have been established by the FM 355 by means of an identification as they have been optimized for the process.

Mode of operation

The parameters READ_PAR and LOAD_PAR are used to control the mode of operation of the instruction FUZ_355

If READ_PAR = TRUE, FUZ_355 reads the parameters of all the temperature controllers of the FM 355 and stores them in the instance DB. After the temperature controller parameters have been successfully read out, FUZ_355 sets the parameter READ_PAR to FALSE.

If LOAD_PAR = TRUE, FUZ_355 writes the parameters of all the temperature controllers of the FM 355 from the instance DB to the FM 355. After the parameters have been successfully transferred, FUZ_355 sets the parameter LOAD_PAR to FALSE.

Call

FUZ_355 requires no initialization routine.

If the FM 355 is used in distributed I/Os, it may take a few call cycles until the parameters READ_PAR and LOAD_PAR are reset to FALSE. Therefore, call the instruction FUZ_355 conditionally, as long as READ_PAR = TRUE or LOAD_PAR = TRUE.

FUZ_355 has to be called in the same OB as all the other instructions that access the same FM 355.

Start-up

During the start-up of the CPU you should set the LOAD_PAR parameter of the FUZ_355 instruction and then call the block in the cyclic program, provided LOAD_PAR = TRUE.

Input parameters FUZ_355

Table 9-216

Parameters	Address	Data type	Default	Description
MOD_ADDR	0.0	INT	256	The module address that results from the configuration with STEP 7 is set at this input.

Output parameters FUZ_355

Table 9-217

Parameters	Address	Data type	Default	Description
RET_VALU	2.0	WORD	0	RET_VALU contains the 2nd and 3rd bytes from the STATUS parameter of the RDREC and WRREC instructions. RET_VALU can be evaluated when an error is reported via QMOD_F.
PARAFFUZ	4.0	WORD	0	A parameter assignment error created by the the instruction FUZ_355 is displayed at the parameter PARAFFUZ as follows: High byte of PARAFFUZ = 01: There is a parameter assignment error.. High byte of PARAFFUZ = 00: There is no parameter assignment error. The low byte contains the offset of the parameter that caused the parameter assignment error, calculated from the static variable FUZ_PAR[1]. For instance, PARAFFUZ = W#16#0104 means that the second parameter is faulty.

See also

RDREC: Read data record (Page 6459)

WRREC: Write data record (Page 6461)

In/out parameters FUZ_355

Table 9-218

Parameters	Address	Data type	Default	Description
READ_PAR	6.0	BOOL	FALSE	If the READ_PAR parameter is set, the fuzzy parameters are read out from the module and stored in the static variables of the instance DB.
LOAD_PAR	6.1	BOOL	FALSE	If the LOAD_PAR parameter is set, the fuzzy parameters are read out from the static variables of the instance DB module and transferred to the module.

FORCE355

Description FORCE355

The instruction FORCE355 is used to simulate (force) the analog and digital input values to support the start-up.

Simulating Digital Values

The simulation of the values for the digital inputs 1 to 8 is enabled via the switches S_DION[i], where 1 ≤ i ≤ 8. You specify the simulation values at the parameters DI_SIM[i]. If S_DION[i] = TRUE, the value DI_SIM[i] is used Instead of the value of digital input i of the module,. LEDs I1 to I8 also always display the state of the corresponding digital input during simulation.

Simulating Analog Values

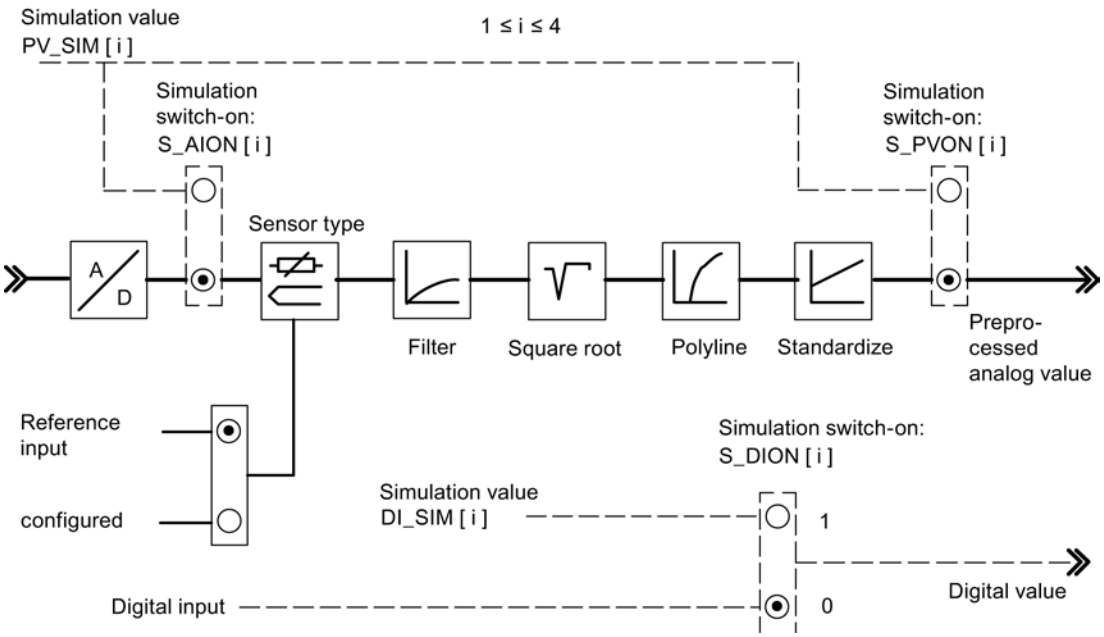
The simulation of the analog values for the channels one to 4 is enabled via the switches S_AION[i] or S_PVON[i], where 1 ≤ i ≤ 4.

You specify the simulation values for the channels one to 4 via the parameter PV_SIM[i].

You can have the simulation values become effective at two points:

- S_AION[i] = TRUE (1 ≤ i ≤ 4)
  The value PV_SIM[i] is used instead of the value of analog input i of the module.
- S_PVON[i] = TRUE (1 ≤ i ≤ 4)
  The value PV_SIM[i] is used instead of the conditioned value of analog input i of the module.

The following figure shows at which point the simulated analog value is effective.



Enabling and specification of the simulation values (forcing) is not carried out via the parameter assignment interface. This is why the relevant switches and connecting lines are drawn as dotted lines.

Call

FORCE355 does not require any initialization routine and is normally called cyclically.

FORCE355 has to be called in the same OB as all the other instructions that access the same FM 355.

Startup characteristics

When FM 355 restarts after power off, the simulation switches on the FM 355 are again set to FALSE.

Input parameters FORCE355

Table 9-219

Parameters	Address	Data type	Default	Description
S_AION	0.0	ARRAY [1..4] of BOOL	FALSE	If, for example, the S_AION[1] switch is set to TRUE, the value PV_SIM[1] is used instead of the analog input value 1 of the module.
S_PVON	2.0	ARRAY [1..4] of BOOL	FALSE	If, for example, the S_PVON[1] switch is set to TRUE, the value PV_SIM[1] is used instead of the conditioned analog input value 1 of the module.

Parameters	Address	Data type	Default	Description
PV_SIM	4.0	ARRAY [1..4] of REAL	0.0	The simulation value for the analog input 1 is specified, for example, at input PV_SIM[1]. If S_PVON = TRUE, then the conditioned analog input value is specified here. If S_PVON = FALSE and S_AION = TRUE, then the analog input value, which is transformed into a conditioned value by means of the conditioning functions, is specified in mA or mV. Permitted are 0.0 to 20.0 [mA], -1500 to +10000 [mV], or a technical range of values.
S_DION	20.0	ARRAY [1..8] of BOOL	FALSE	If, for example, the S_DION[1] switch is set to TRUE, the value DI_SIM[1] is used instead of the digital input value 1 of the module.
DI_SIM	22.0	ARRAY [1..8] of BOOL	FALSE	Simulation value for digital input
MOD_ADDR	24.0	INT	256	The module address that results from the configuration with STEP 7 is set at this input.

Output parameters FORCE355

Table 9-220

Parameters	Address	Data type	Default	Description
RET_VALU	26.0	WORD		RET_VALU contains the 2nd and 3rd bytes from the STATUS parameter of the RDREC and WRREC instructions. RET_VALU can be evaluated when an error is reported via QMOD_F.

See also

RDREC: Read data record (Page 6459)

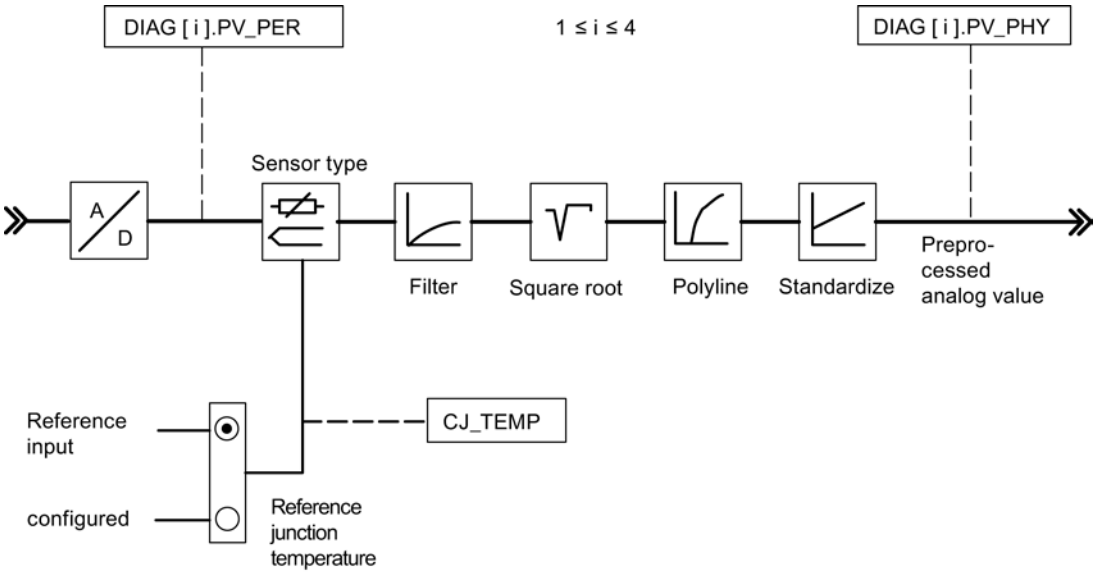
WRREC: Write data record (Page 6461)

READ_355

Description READ_355

The instruction READ_355 is used to read out the digital and analog input values to support the start-up.

The following values are displayed:



Call

READ_355 does not require any initialization routine and is normally called cyclically.

READ_355 has to be called in the same OB as all the other instructions that access the same FM 355.

Input parameters READ_355

Table 9-221

Parameters	Address	Data type	Default setting	Description
MOD_ADDR	0.0	INT	256	The module address that results from the configuration with STEP 7 is set at this input.

Output parameters READ_355

Table 9-222

Parameters	Address	Data type	Default	Description
CJ_TEMP	2.0	REAL	0.0	At parameter CJ_TEMP the reference junction temperature measured at the reference point is expressed degrees C or in degrees F (depending on which temperature unit has been parameterized). If no "Thermocouple" sensor type was configured or if the configured reference junction temperature was selected at all the analog inputs, 0.0 is displayed at the CJ_TEMP parameter.
STAT_DI	6.0	ARRAY [1..8] of BOOL	FALSE	The actual states of digital inputs 1 to 8 are displayed at parameters STAT_DI[1] to STAT_DI[8], even if these are simulated.
DIAG[x].PV_PER	(channel number) x 8	ARRAY [1..4] of STRUCT	0.0	The values of analog inputs one to four are displayed at parameters DIAG[1].PV_PER bis DIAG[4].PV_PER in the unit mA or mV. If the simulation of the analog input value has been enabled via the instruction FORCE355, the simulated value will be shown.
DIAG[x].PV_PHY	(channel number) x 8 + 4	ARRAY [1..4] of STRUCT	0.0	The conditioned analog input value one to four is displayed in physical unit at the parameters DIAG[1].PV_PHY to DIAG[4].PV_PHY If the simulation of the conditioned physical analog input value is enabled via the instruction FORCE355, the simulated value is displayed.
RET_VALU	40.0	WORD	0	RET_VALU contains the 2nd and 3rd bytes from the STATUS parameter of the RDREC and WRREC instructions. RET_VALU can be evaluated when an error is reported via QMOD_F.

See also

RDREC: Read data record (Page 6459)

WRREC: Write data record (Page 6461)

CH_DIAG

Description CH_DIAG

The instruction CH_DIAG is used to read out additional channel-specific diagnostic variables from the module.

Call

CH_DIAG does not require any initialization routine and is normally called cyclically.

CH_DIAG has to be called in the same OB as all the other instructions that access the same FM 355.

Input parameters CH_DIAG

Table 9-223

Parameters	Address	Data type	Default	Description
MOD_ADDR	0.0	INT	256	The module address that results from the configuration with STEP 7 is set at this input.
CHANNEL	2.0	INT	1	The number of the controller channel to which the instance DB is referenced is configured at the "Channel number" input. Values of 1 to 4 are permitted.

See also

Description CH_DIAG (Page 7208)
Output parameters CH_DIAG (Page 7209)
Parameter IDSTATUS (Page 7210)
Block diagrams CH_DIAG (Page 7211)

Output parameters CH_DIAG

Table 9-224

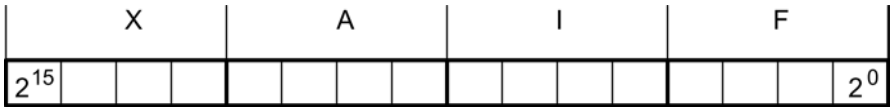
Parameters	Address	Data type	Default	Description
SP_R	4.0	REAL	0.0	If a ratio controller is set, the input value of the setpoint value is assigned to the parameter.
PV_R	8.0	REAL	0.0	The following value is only assigned to the parameter value if the ratio controller is set: (Process value A - Setpoint value offset) / Process value D
DIF_I	12.0	REAL	0.0	The input variable of the D-action is displayed at the DIF_I parameter. This is particularly of interest if, for example, an analog input is configured as the input variable of the D-action.
TRACKPER	16.0	REAL	0.0	The TRACKPER parameter shows the input variable to which the output value is tracked if the track-manipulated-variable function is enabled on the controller.
IDSTATUS	20.0	WORD	0.0	The state of the identification is displayed at Parameter IDSTATUS.
LMN_P	22.0	REAL	0.0	The P-action of the manipulated variable is displayed at the LMN_P parameter.
LMN_I	26.0	REAL	0.0	The I-action of the manipulated variable is displayed at the LMN_I parameter.
LMN_D	30.0	REAL	0.0	The D-action of the manipulated variable is displayed at the LMN_D parameter.
RET_VALU	34.0	WORD	0	RET_VALU contains the 2nd and 3rd bytes from the STATUS parameter of the RDREC and WRREC instructions. RET_VALU can be evaluated when an error is reported via QMOD_F.

See also

RDREC: Read data record (Page 6459)
WRREC: Write data record (Page 6461)

Parameter IDSTATUS

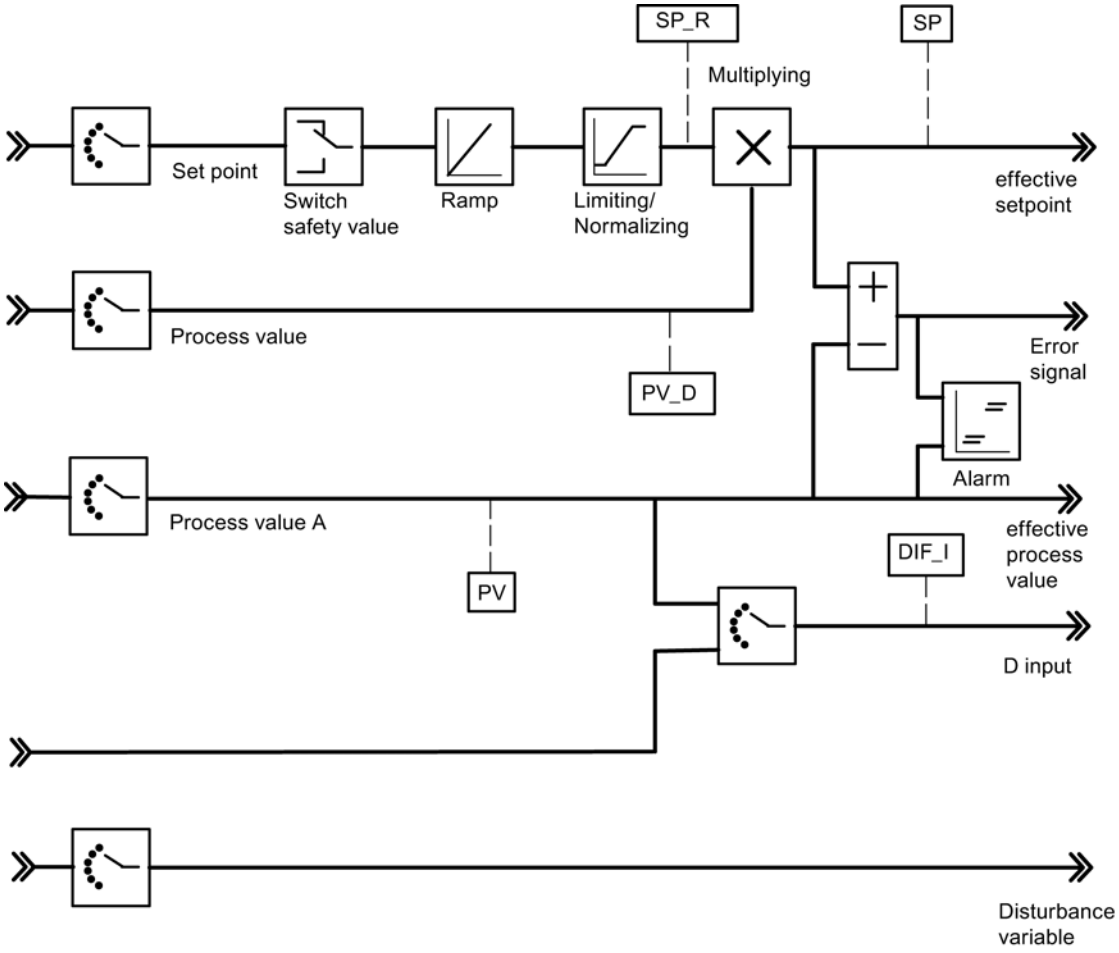
The IDSTATUS parameter supplies information on the status of identification and contains the four hexadecimal values X, A, I and F.



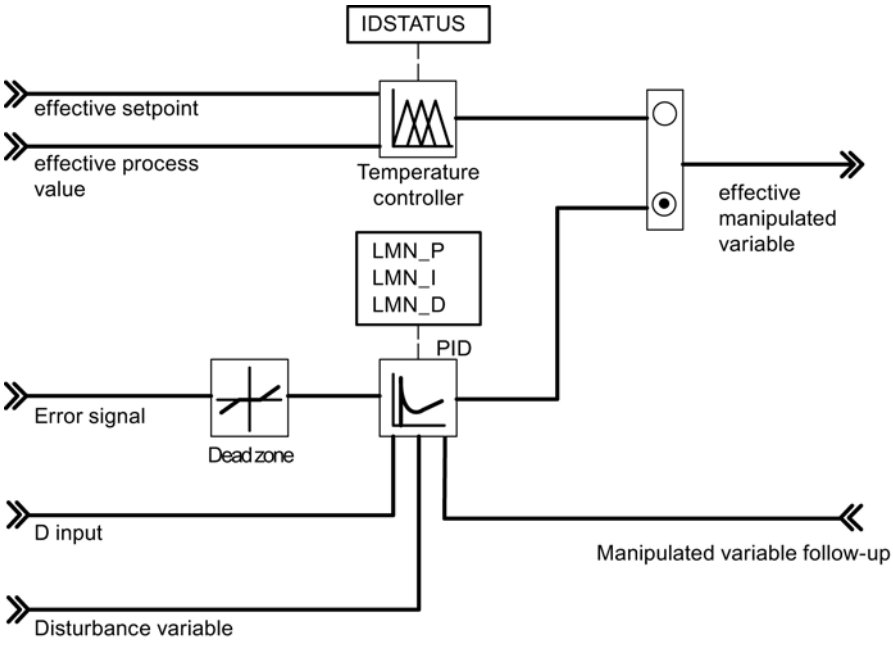
Value	Description
X	Without meaning (always 0)
A	Action number • 0 = Manual mode (or no closed-loop control mode); • 2 = Closed-loop control; • 4 = Tuning enabled (FUZID_ON = true); • 6 = Transition state from manual mode to 2 or 4;
I	Display "Identification running" and "Parameters determined, but not yet stored in EEPROM" • 0 = Identification not running, no new parameters determined • 1 = Identification running, no new parameters determined • 2 = Identification not running, new parameters determined, but not yet stored in EEPROM • 3 = Identification running, new parameters determined, but not yet stored in EEPROM
F	Error number • 0 = No error • 4 = Excessive step-change of the process value during the identification • 5 = Ratio of time lag to system time constant too large or strongly non-linear behavior of the controlled system. • 6 = Temperature drop or rise during identification start too large. System not settled sufficiently

Block diagrams CH_DIAG

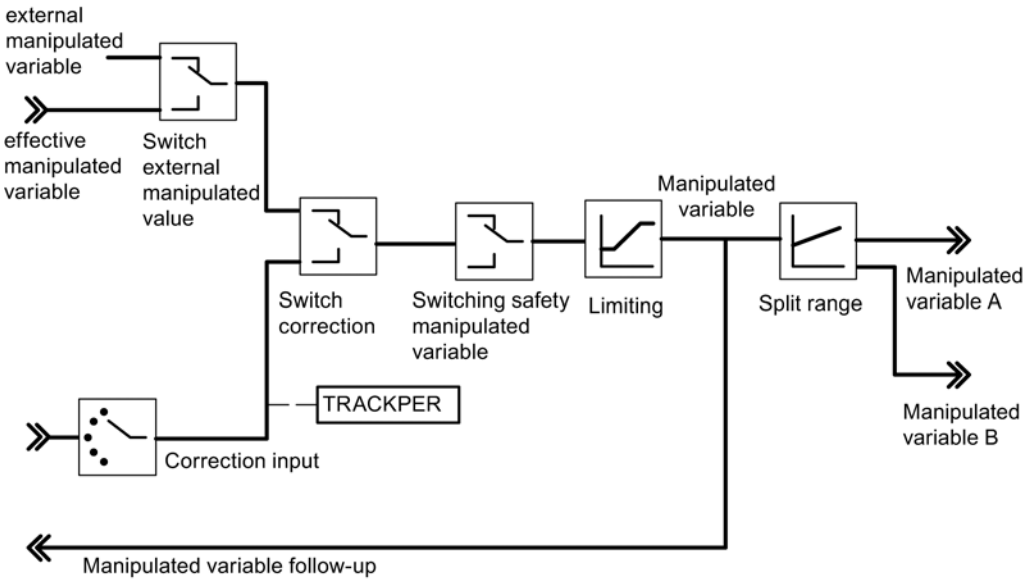
The following diagnostic values of the control deviation are displayed.



The following values of the control algorithm are displayed.



The following values of the continuous controller or step controller are displayed.



PID_PAR

Description PID_PAR

The instruction PID_PAR is used for the online modification of parameters that cannot be defined via the instruction PID_FM. To view the parameters that can be modified, refer to the table Changeable parameters (Page 7215)

Call

PID_PAR requires an initialization routine. It is automatically triggered if the system data (SDB default data of the FM 355) has not yet been read from the instruction PID_PAR. You can also start the initialization yourself with COM_RST=TRUE, which is usually done in OB100 since the system data is sent to the FM 355 after STOP-RUN of the CPU. The initialization process lasts several cycles. No data is sent to the FM 355 via WRREC during the initialization. The block automatically resets the COM_RST parameter after the initialization.

To save run time, PID_PAR should not be called cyclically. It should only be called when parameters are to be changed. COM_RST must then be FALSE.

PID_PAR has to be called in the same OB as all the other instructions that access the same FM 355.

If the FM 355 is used in distributed I/Os, it may take a few call cycles until the parameters have been transferred to the FM 355. The BUSY parameter has the value TRUE until the transfer has been completed. You should call PID_PAR as many times as necessary until BUSY = FALSE and RET_VALU = 0.

Mode of operation

In the initialization routine, PID_PAR reads the parameters from the system data and stores them in static variables. Per call, PID_PAR modifies one of these REAL variables and one of these INT variables. At the input parameter INDEX_R or INDEX_I, specify the index numbers of the parameter that you want to modify. At the input parameter VALUE_R or VALUE_I, enter the new values. PID_PAR transfers the entire data record to FM 355 with the modified variables.

To modify additional parameters you have to call **the same** instance DB several times consecutively with COM_RST = FALSE and with different index numbers.

The COM_RST parameter is an input parameter that is not reset by PID_PAR.

Note

Please note that parameters that you have changed using PID_PAR are overwritten by the parameters from the system data during the start-up.

Start-up

During CPU start-up and during CPU STOP-RUN transition, the parameters on the FM 355 are overwritten with the parameters from the SDB of the CPU.

Example

During operation you want to modify the ramp-up time for the reference variable and, depending on the process state, use different analog input values as process value.

- Call PID_PAR using COM_RST = TRUE during the start-up of the CPU.
- To configure the ramp-up time for the reference variable at 10.0, use INDEX_R = 30, VALUE_R = 10.0 and INDEX_I = 0 to call PID_PAR during operation.
- If you want to parameterize the analog input value 4 of the module as the process value, call PID_PAR with INDEX_R = 0, INDEX_I = 50 and VALUE_I = 4 during operation.

See also

Alterable parameters (Page 7215)

Input parameters PID_PAR (Page 7214)

Output parameters PID_PAR (Page 7215)

Input parameters PID_PAR

Table 9-225

Parameters	Address	Data type	Default	Description
COM_RST	0.0	BOOL	TRUE	If the parameter COM_RST = TRUE, the instruction PID_PAR executes an initialization routine. During this initialization process the parameters are read from the system data of the CPU and saved in the instance DB.
MOD_ADDR	2.0	INT	256	The module address that results from the configuration with STEP 7 is set at this input.
CHANNEL	4.0	INT	1	The number of the controller channel to which the instance DB is referenced is configured at the "Channel number" input. Values 1 to 4 are permitted.
INDEX_R	6.0	INT	0.0	• 0: No REAL parameters are changed • 1 to 48: Each value corresponds to a changeable REAL parameter.
VALUE_R	8.0	REAL	0.0	Depends on the relevant parameter
INDEX_I	12.0	INT	0.0	• 0: No INT parameters are changed. • 49 to 61: Each value corresponds to a changeable INT parameter.
VALUE_I	14.0	INT	0.0	Depends on the relevant parameter

See also

Description PID_PAR (Page 7212)

Output parameters PID_PAR (Page 7215)

Output parameters PID_PAR

Table 9-226

Parameters	Address	Data type	Default	Description
RET_VALU	16.0	WORD	0	RET_VALU contains the 2nd and 3rd bytes from the STATUS parameter of the RDREC and WRREC instructions. RET_VALU can be evaluated when an error is reported via QMOD_F.
BUSY	18.0	BOOL	FALSE	If BUSY = TRUE, the parameters have not yet been transferred from the module (in the case of distributed I/Os). The instruction PID_PAR should then be called again in the next cycle.

See also

RDREC: Read data record (Page 6459)

WRREC: Write data record (Page 6461)

Alterable parameters

Table 9-227 List of the REAL and INT parameters on a FM 355 that are to be modified with PID_PAR

Data type	Description	Index number
–	No parameter selected	0
REAL	Filter time constant for analog input	1
REAL	End of scale (100%)	2
REAL	Start of scale (0%)	3
REAL	Polyline, Interpolation point 1 input side	4
REAL	Polyline, Interpolation point 2 input side	5
REAL	Polyline, Interpolation point 3 input side	6
REAL	Polyline, Interpolation point 4 input side	7
REAL	Polyline, Interpolation point 5 input side	8
REAL	Polyline, Interpolation point 6 input side	9
REAL	Polyline, Interpolation point 7 input side	10
REAL	Polyline, Interpolation point 8 input side	11
REAL	Polyline, Interpolation point 9 input side	12
REAL	Polyline, Interpolation point 10 input side	13
REAL	Polyline, Interpolation point 11 input side	14
REAL	Polyline, Interpolation point 12 input side	15
REAL	Polyline, Interpolation point 13 input side	16
REAL	Polyline, Interpolation point 1 output side	17
REAL	Polyline, Interpolation point 2 output side	18
REAL	Polyline, Interpolation point 3 output side	19
REAL	Polyline, Interpolation point 4 output side	20
REAL	Polyline, Interpolation point 5 output side	21

Data type	Description	Index number
REAL	Polyline, Interpolation point 6 output side	22
REAL	Polyline, Interpolation point 7 output side	23
REAL	Polyline, Interpolation point 8 output side	24
REAL	Polyline, Interpolation point 9 output side	25
REAL	Polyline, Interpolation point 10 output side	26
REAL	Polyline, Interpolation point 11 output side	27
REAL	Polyline, Interpolation point 12 output side	28
REAL	Polyline, Interpolation point 13 output side	29
REAL	Ramp-up time for reference variable	30
REAL	Safety reference variable or safety Reference variable response	31
REAL	Offset for setpoint link (ratio/blending controller)	32
REAL	Factor for process value B (three-component controller)	33
REAL	Factor for process value C (three-component controller)	34
REAL	Offset for process value link (three-component controller)	35
REAL	Factor for disturbance variable link	36
REAL	Operating point	37
REAL	Aggressivity at fuzzy controller	38
REAL	Vertices for split-range function: Start of input signal A range	39
REAL	Vertices for split-range function: End of input signal A range	40
REAL	Vertices for split-range function: Start of output signal A range	41
REAL	Vertices for split-range function: End of output signal A range	42
REAL	Vertices for split-range function: Start of input signal B range	43
REAL	Vertices for split-range function: End of input signal B range	44
REAL	Vertices for split-range function: Start of output signal B range	45
REAL	Vertices for split-range function: End of output signal B range	46
REAL	Minimum pulse time	47
REAL	Minimum break time	48
INT	Selection of the reference variable SP or SP_RE for the controller 0: Setpoint SP_RE of the instruction 1 to 4: Analog input value 1 to 4 17 to 20: Manipulated variable (LMN) of controllers 1 to 4	49
INT	Selection of the main controlled variable process value A for the controller 0: Process value A = 0.0 1 to 4: Analog input value 1 to 4	50
INT	Selection of the auxiliary controlled variable process value B for the controller 0: Process value B = 0.0 1 to 4: Analog input value 1 to 4	51
INT	Selection of the auxiliary controlled variable process value C for the controller 0: Process value C = 0.0 1 to 4: Analog input value 1 to 4	52

Data type	Description	Index number
INT	Selection of the auxiliary controlled variable process value D for the controller 0: Process value D = 0.0 1 to 4: Analog input value 1 to 4 17 to 20: Manipulated variable (LMN) of controllers 1 to 4	53
INT	Selection of the disturbance variable DISV for the controller 0: Disturbance variable = 0.0 1 to 4: Analog input value 1 to 4	54
INT	Selection of position follow-up TRACK_PER for the controller 0: Position follow-up = 0.0 1 to 4: Analog input value 1 to 4	55
INT	Selection of position follow-up LMNR_PER for the controller 0: Position follow-up = 0.0 1 to 4: Analog input value 1 to 4	56
INT	Selection of the signal for switching to the safety value for the manipulated value of the controller 0: Can only be assigned via SAFE_ON of the instruction PID_FM 1 to 8: Assignment via SAFE_ON parameter of the instruction PID_FM ORed with digital input 1 to 8	57
INT	Selecting the signal for switching over to tracking function of the manipulated value of the controller 0: Can only be assigned via LMNTRKON of the instruction PID_FM 1 to 8: Assignment via LMNTRKON parameter of the instruction PID_FM ORed with digital input 1 to 8	58
INT	Selecting the signal for changing over the manipulated value of the controller to LMN_RE 0: Can only be assigned via LMN_REON of the instruction PID_FM 1 to 8: Assignment via LMN_REON parameter of the instruction PID_FM ORed with digital input 1 to 8	59
INT	Selecting the high limit signal of the position feedback 0: Can only be assigned via LMNRHSRE of the instruction PID_FM 1 to 8: Assignment via LMNRHSRE parameter of the instruction PID_FM ORed with digital input 1 to 8	60
INT	Selecting the low limit signal of the position feedback 0: Can only be assigned via LMNRLSRE of the instruction PID_FM 1 to 8: Assignment via LMNRLSRE parameter of the instruction PID_FM ORed with digital input 1 to 8	61

See also

Input parameters PID_PAR (Page 7214)
Output parameters PID_PAR (Page 7215)

CJ_T_PAR

Beschreibung CJ_T_PAR

The instruction CJ_T_PAR is used for online modification of the configured reference junction temperature. This is necessary if a temperature control system with several FM 355 with thermoelement inputs is to be operated without having to connect a Pt 100 to each FM 355.

Call

CJ_T_PAR requires an initialization routine. For this purpose it has to be called once with the COM_RST = TRUE parameter at the start-up of the CPU.

CJ_T_PAR is normally called cyclically. For this purpose COM_RST should be set to FALSE for runtime reasons. The COM_RST parameter is an input parameter that is not reset.

CJ_T_PAR has to be called in the same OB as all the other instructions that access the same FM 355.

If the FM 355 is used in distributed I/Os, it may take a few call cycles until the parameters have been transferred to the FM 355. The BUSY parameter has the value TRUE until the transfer has been completed. You should therefore call the instruction CJ_T_PAR as many times as necessarry when changing parameters until BUSY = FALSE.

Example

If an FM 355 measures the reference junction temperature of an extruder control system with more than four heating zones, this can be read out by READ_355 at parameter CJ_TEMP and can be parameterized for the other FM 355 via CJ_T_PAR.

See also

Input parameters CJ_T_PAR (Page 7218)

Output parameters CJ_T_PAR (Page 7219)

Input parameters CJ_T_PAR

Table 9-228

Parameters	Address	Data type	Default	Description
COM_RST	0.0	BOOL	FALSE	If the parameter COM_RST = TRUE, the instruction CJ_T_PAR executes an initialization routine. During this initialization process the parameters are read from the system data of the CPU and saved in the instance DB.
MOD_ADDR	2.0	INT	256	The module address that results from the configuration with STEP 7 is set at this input.
CJ_T	4.0	REAL	0.0	The reference junction temperature can be specified at parameter CJ_T. The permissible value range depends on the sensor type.

See also

Beschreibung CJ_T_PAR (Page 7218)

Output parameters CJ_T_PAR (Page 7219)

Output parameters CJ_T_PAR

Table 9-229

Parameters	Address	Data type	Default	Description
RET_VALU	8.0	WORD	0	RET_VALU contains the 2nd and 3rd bytes from the STATUS parameter of the RDREC and WRREC instructions. RET_VALU can be evaluated when an error is reported via QMOD_F.
BUSY	10.0	BOOL	FALSE	If BUSY = TRUE, the parameters have not yet been transferred from the module (in the case of distributed I/Os). The instruction PID_PAR should then be called again in the next cycle.

See also

RDREC: Read data record (Page 6459)

WRREC: Write data record (Page 6461)

DB for HMI

Description DB 101 to 104

In order to operate the FM 355 if the CPU fails, you require data blocks 101 to 104. You can establish a maximum of three connections from the FM 355 to OPs via MPI.

Operator control of the FM 355 with the OP is only possible in the STOP mode of the CPU or at a CPU failure.

Monitoring of the FM 355 with the OP is always possible.

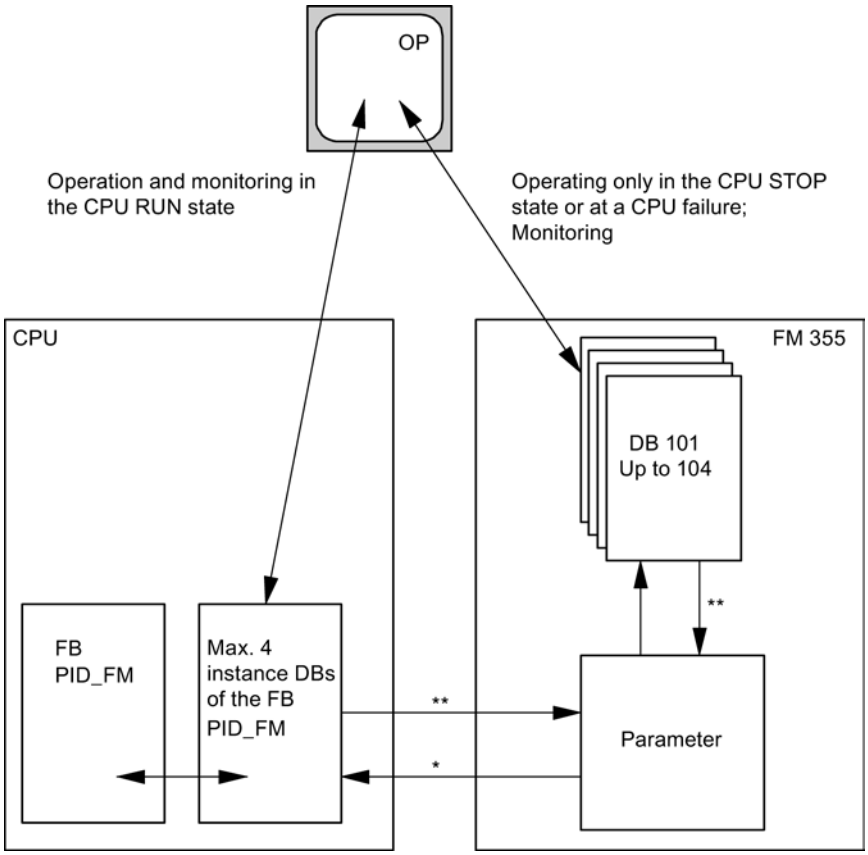
The variable interface of the FM 355 contains four data blocks with the block numbers 101 to 104 for the controller channels 1 to 4 (refer to the following figure).

Note

The contents of the data blocks 101 to 104 do not automatically reflect the parameter value effective at the FM 355. Parameters changed with the OP are only transferred to the FM 355 after the operating bits LOAD_PAR and LOAD_OP have been set.

If you change a parameter using OP operation without setting the corresponding operating bit, the changed parameter value is entered in the data block, but the FM 355 continues to operate internally with the unchanged old value of the parameter.

After the operating bits have been set and the parameters have been transferred to the FM 355, the operating bits LOAD_PAR and LOAD_OP are reset by the FM 355.



* controlled by the READ_VAR parameter of the instance DB

** controlled by the LOAD_OP and LOAD_PAR parameters

See also

Description PID_FM (Page 7185)

Input parameters DB 101 to 104

Table 9-230

Parameters	Address	Data type	Default	Description
SP_HLM	0.0	REAL	100.0	The setpoint is always limited to a high and a low limit. The "Setpoint high limit" input specifies the high limit. SP_HLM > SP_LLM
SP_LLM	4.0	REAL	0.0	The setpoint is always limited to a high and a low limit. The "Setpoint low limit" input specifies the low limit. SP_LLM < SP_HLM

Parameters	Address	Data type	Default	Description
H_ALM	8.0	REAL	100.0	You can configure four limits for monitoring the process value or error signal. The "High limit interrupt" input specifies the highest limit. H_ALM > H_WRN
H_WRN	12.0	REAL	90.0	You can configure four limits for monitoring the process value or error signal. The "High limit warning" input specifies the second to highest limit. Values from H_ALM to L_WRN are permitted.
L_WRN	16.0	REAL	10.0	You can configure four limits for monitoring the process value or error signal. The "Low limit warning" input specifies the second to lowest limit. Values from H_WRN to L_ALM are permitted.
L_ALM	20.0	REAL	0.0	You can configure four limits for monitoring the process value or error signal. The "Low limit alarm" specifies the lowest limit. L_ALM < L_WRN
HYS	24.0	REAL	1.0	To prevent flickering of the monitoring displays, a hysteresis can be configured at the "Hysteresis" input. HYS ≥ 0.0
DEADB_W	28.0	REAL	0.0	A dead band is applied to the error signal. The "Dead band width" input determines the size of the dead band. DEADB_W ≥ 0.0
GAIN	32.0	REAL	1.0	The "Proportional gain" input specifies controller gain.
TI	36.0	REAL	3000.0	The "Integration time" input determines the time response of the integrator. If TI = 0, the integral action is disabled. TI = 0.0 or TI ≥ 0.5
TD	40.0	REAL	0.0	The "Derivative-action time" input determines the time response of the differentiator. If TD = 0, the derivative action is disabled TD = 0.0 or TD ≥ 1.0
TM_LAG	44.0	REAL	5.0	The algorithm of the D-action includes a time lag that can be configured at the "Time lag of the D-action" input. TM_LAG ≥ 0.5
LMN_SAFE	48.0	REAL	0.0	At the "Safety manipulated value" input, a safety value can be configured for the manipulated value, . Values from -100.0 to 100.0 (%) are permitted.
LMN_HLM	52.0	REAL	100.0	The manipulated value is always limited to a high and a low limit. The "High limit limit of manipulated value" input specifies the high limit. (This does not apply to step controllers without analog position feedback) Values from LMN_LLM to 100.0 (%) are permitted.
LMN_LLM	56.0	REAL	0.0	The manipulated value is always limited to a high and a low limit. The "Low limit of manipulated value" input specifies the low limit. (This does not apply to step controllers without analog position feedback) Values from -100.0 to LMN_HLM (%) are permitted.
MTR_TM	60.0	REAL	60.0	The runtime from endstop to endstop of the control valve is entered at the "Motor actuating time" parameter. (Applies to step controllers only). MTR_TM ≥ 0.001

Parameters	Address	Data type	Default	Description
PULSE_TM	64.0	REAL	0.2	A minimum pulse duration can be configured at the "Minimum pulse time" parameter. (Applies to step controllers or pulse controllers only) PULSE_TM ≥ 0.0
BREAK_TM	68.0	REAL	0.2	A minimum break time can be configured at the "Minimum break time" parameter. (Applies to step controllers or pulse controllers only) BREAK_TM ≥ 0.0
SP_RE	72.0	REAL	0.0	An external setpoint is interconnected with the controller at the "External setpoint" input.
LMN_RE	76.0	REAL	0.0	An external manipulated value is interconnected with the controller at the "External manipulated value" input.
LMNRSVAL	80.0	REAL	0.0	The Commissioning dialog provides you with access to the input "Start value of the simulated position feedback". The start value of the simulation is entered at the parameter. (Only for step controllers without analog position feedback) Values from -100.0 to 100.0 (%) are permitted.
SAFE_ON	84.0	BOOL	FALSE	If the "Adopt safety position" input is set, a safety value is applied as manipulated value. Note: The signal processing of manipulated values via LMNDN_OP , LMNUP_OP and LMNSOPON on a step controller is of higher priority than the safety manipulated value.
LMNTRKON	84.1	BOOL	FALSE	If input "Follow-up (LMN via AI)" is set, the manipulated value is adjusted to an analog input (AI). (This does not apply to step controllers without analog position feedback)
LMN_REON	84.2	BOOL	FALSE	If input "Enable external manipulated value" is set, the external manipulated value LMN_RE is set as the manipulated value.
LMNRHSRE	84.3	BOOL	FALSE	The signal "Control valve at high endstop" is interconnected at the "High endstop signal of position feedback" input. LMNRHSRE = TRUE means: The control valve is at high endstop. (Applies to step controllers only).
LMNRLSRE	84.4	BOOL	FALSE	The signal "Control valve at low endstop" is interconnected at the "Low endstop signal of position feedback" input. LMNRLSRE = TRUE means: The control valve is at the low endstop. (Applies to step controllers only).
LMNSOPON	84.5	BOOL	FALSE	If the bit is set at input "Enable manipulated value signal operation", the LMNUP_OP and LMNDN_OP signals are set as the manipulated value signals. (Applies to step controllers only).
LMNUP_OP	84.6	BOOL	FALSE	If LMNSOPON is set, the value at input "Manipulated value signal up operation" is set as the manipulated value signal. (Applies to step controllers only).
LMNDN_OP	84.7	BOOL	FALSE	If LMNSOPON is set, the value at input "Manipulated value signal down operation" is set as the manipulated value signal. (Applies to step controllers only).
MONERSEL	85.0	BOOL	FALSE	The controller possesses a limit value detector that can be applied either for the process value or for the error signal. If the input "Monitoring: Process value = 0, Error signal = 1" is set, the errcr signal will be monitored.

Parameters	Address	Data type	Default	Description
LMNRS_ON	85.1	BOOL	FALSE	You can simulate no position feedback if this is not available. The function is enabled by setting the "Enable simulation of position feedback" input CAUTION: Offset between the simulation value and real position feedback increases over the time. (Only for step controllers without analog position feedback)
FUZID_ON	85.2	BOOL	FALSE	The identification of the fuzzy algorithm is enabled at the input "Enable fuzzy identification".
SPINT_EN	85.3	BOOL	FALSE	The input "Operator input: External = 0, Internal = 1" determines the input that is transferred as a setpoint to the module. SPINT_EN = TRUE: SP_INT is transferred. SPINT_EN = FALSE: SP_RE is transferred.
P_SEL	85.4	BOOL	TRUE	It is possible to enable or disable PID actions individually in the PID algorithm. The proportional action is enabled when the "Enable P-action" input is set.
PFDB_SEL	85.5	BOOL	FALSE	In the PID algorithm, the P- and D-actions can be included in the feedback loop. The proportional action is located in the feedback loop when the "Enable P-action in feedback loop" input is set.
D_EL_SEL	86.0	INT	0	The D-action element in the PID algorithm can be assigned to a separate input. This is selected at the input "D-action element input". 0: Error signal 1 to 4: Analog input 1 to 4 17: negative process value

See also

Description PID_FM (Page 7185)

Output parameters DB 101 to 104

Table 9-231

Parameters	Address	Data type	Default	Description
SP	94.0	REAL	0.0	The setpoint that is currently in effect is available at the "Setpoint" output.
PV	98.0	REAL	0.0	The process value that becomes effective is issued on the "Process value" output.
ER	102.0	REAL	0.0	The control deviation that becomes effective is issued on the "Control deviation" output.
DISV	106.0	REAL	0.0	The interference that becomes effective is issued on the "Interference" output. Values from -100.0 to 100.0 (%) are permitted.
LMN	110.0	REAL	0.0	The manipulated value that becomes effective is issued on the "Manipulated value" output. Step controllers without analog position feedback output the unlimited P- + D-action at the LMN parameter. Values from -100.0 to 100.0 (%) are permitted.

Parameters	Address	Data type	Default	Description
LMN_A	114.0	REAL	0.0	At the output "Manipulated value A of the split range function / position feedback", continuous controllers display the manipulated value A and step controllers with analog position feedback display the position feedback. Step controllers without analog position feedback display the simulated position feedback. Values from -100.0 to 100.0 (%) are permitted.
LMN_B	118.0	REAL	0.0	Continuous controllers display the manipulated value B of the split-range function at the output “Manipulated value B of split-range function”. Values from -100.0 to 100.0 (%) are permitted.
QH_ALM	122.0	BOOL	FALSE	The process value or the controlled variable is monitored for four limits. Violation of H_ALM limit is reported at the output "High limit alarm triggered".
QH_WRN	122.1	BOOL	FALSE	The process value or the controlled variable is monitored for four limits. Exceeding of the limit H_WRN is signaled at the "High limit warning triggered" output.
QL_WRN	122.2	BOOL	FALSE	The process value or the controlled variable is monitored for four limits. Exceeding of the limit L_WRN is signaled at the "Low limit warning triggered" output.
QL_ALM	122.3	BOOL	FALSE	The process value or the controlled variable is monitored for four limits. Exceeding of the limit L_ALM is signaled at the "Low limit alarm triggered" output.
QLMN_HLM	122.4	BOOL	FALSE	The manipulated value is always limited to a high and a low limit. Violation of the high limit is reported at output "High limit of manipulated value triggered". (Does not apply to step controllers without analog position feedback)
QLMN_LLM	122.5	BOOL	FALSE	The manipulated value is always limited to a high and a low limit. Violation of the low limit is reported at output "Low limit of manipulated value triggered". (Does not apply to step controllers without analog position feedback)
QSPINTON	122.6	BOOL	FALSE	The output "Internal setpoint on" indicates that SP_INT was transferred to the module.
QPARA_F	123.0	BOOL	FALSE	The module checks the validity of the parameters. A parameter assignment error is displayed at the "Parameter assignment error" output. These parameter assignment errors can also be read out in the module's "Online & Diagnostics" dialog.
QCH_F	123.1	BOOL	FALSE	The "Channel error" output is set if the controller channel cannot provide valid results. A channel error (for example, wire break) is also set if QPARA_F = 1 or QMOD_F = 1. If QCH_F = TRUE, then the precise error information in the diagnostic data record DS1 of the module is read out.
QUPRLM	123.2	BOOL	FALSE	The setpoint is limited to a positive and negative slew rate. If the output "Limit of positive setpoint slew rate triggered" is set, the positive setpoint slew rate is limited.
QDNRLM	123.3	BOOL	FALSE	The setpoint is limited to a positive and negative slew rate. If the output "Limit of negative setpoint slew rate triggered" is set, the setpoint fall is limited.
QSP_HLM	123.4	BOOL	FALSE	The setpoint is always limited to a high and a low limit. The output "High limit of setpoint value triggered" indicates that the high limit has been exceeded.
QSP_LLM	123.5	BOOL	FALSE	The setpoint is always limited to a high and a low limit. Violation of the low limit is reported at output "Setpoint low limit triggered".
QSPOPON	123.6	BOOL	FALSE	The output "Setpoint operation on" shows if the setpoint is being operated via the "Commissioning" dialog. If the bit is set, the value SP_OP is applied as setpoint.
QLMNSAFE	123.7	BOOL	FALSE	If the output "Safety mode" is set, the safety manipulated value is output as the manipulated value.
QLMNOPON	124.0	BOOL	FALSE	The output "Manipulated value operation on" shows if the manipulated value is being operated via the "Commissioning" dialog. If the bit is set, the value LMN_OP is applied as manipulated value.

Parameters	Address	Data type	Default	Description
QLMNTRK	124.1	BOOL	FALSE	The output "Follow-up mode" indicates if the manipulated value is being adjusted via an analog input.
QLMN_RE	124.2	BOOL	FALSE	The output "Manual = 1; Automatic = 0" indicates whether or not the manipulated value is set on the external manipulated value LMN_RE (manual = 1).
QLMNR_HS	124.3	BOOL	FALSE	The output "High endstop signal of position feedback" indicates if the control valve is at its high endstop. QLMNR_HS = TRUE means: The control valve is at high endstop. (For step controllers only)
QLMNR_LS	124.4	BOOL	FALSE	The output "Low endstop signal of position feedback" indicates whether the control valve is at its low endstop. QLMNR_LS = TRUE means: The control valve is at its low endstop. (For step controllers only)
QLMNR_ON	124.5	BOOL	FALSE	Output "Position feedback enabled" indicates the set operating mode "Step controller with position feedback" or "Step controller without position feedback".
QFUZZY	124.6	BOOL	FALSE	If the output "PID algorithm = 0; fuzzy = 1", the controller is working with the fuzzy algorithm.
QSPLEPV	124.7	BOOL	FALSE	The output "Display of FUZZY controller: Setpoint < process value" is set when the fuzzy controller is switched on, if the setpoint is less than the effective process value.
QSPR	125.0	BOOL	FALSE	If the output "Split-range operation" is set, the continuous controller is operating in split-range mode.
QLMNUP	125.1	BOOL	FALSE	This is the "Manipulated value signal up" output. (Only in the case of step controllers or pulse controllers)
QLMNDN	125.2	BOOL	FALSE	This is the "Manipulated value signal down" output. (Only in the case of step controllers or pulse controllers)
QBACKUP	125.4	BOOL	FALSE	0= No backup state (CPU in RUN) 1= Backup state (CPU in STOP or has failed)
QID	125.5	BOOL	FALSE	QID = TRUE indicates that an identification is running (not that it is enabled). On completion of the identification, the identification result can be read out from the IDSTATUS parameter of the instruction CH_DIAG.
QMAN_FC	125.6	BOOL	FALSE	The controller is a master controller that is adjusted by the manual mode of a slave controller to its process value or its I-action is stopped, because the setpoint or manipulated value of the slave controller lies in the limit.
RET_VALU	126.0	WORD	0	RET_VALU contains the 2nd and 3rd bytes from the STATUS parameter of the RDREC and WRREC instructions. RET_VALU can be evaluated when an error is reported via QMOD_F.

See also

RDREC: Read data record (Page 6459)

WRREC: Write data record (Page 6461)

Description PID_FM (Page 7185)

In/out parameters DB 101 to 104

Table 9-232

Parameters	Address	Data type	Default	Description
SP_INT	128.0	REAL	0.0	The in/out parameter "Internal setpoint" is used to specify a setpoint by means of operating and monitoring functions.
SP_OP	132.0	REAL	0.0	The "Commissioning" dialog can access the in/out parameter "Setpoint operation". If the SP_OP_ON bit is set, the "Setpoint operation" value is applied as the setpoint.
LMN_OP	136.0	REAL	0.0	The "Commissioning" dialog can access the in/out parameter "Manipulated value operation". If the LMNOP_ON bit is set, the "Manipulated value operation" value is applied as the manipulated value. Values from -100.0 to 100.0 (%) are permitted.
SP_OP_ON	140.0	BOOL	FALSE	The "Commissioning" dialog can access the in/out parameter "Enable setpoint operation". If the bit is set, the value SP_OP is applied as setpoint.
LMNOP_ON	140.1	BOOL	FALSE	The "Commissioning" dialog can access the in/out parameter "Enable manipulated value operation". If the bit is set, the value LMN_OP is applied as manipulated value.
LOAD_PAR	140.2	BOOL	FALSE	If the in/out parameter "Load control parameters to FM 355" is set, the control parameters are loaded into the module and the in/out parameter is reset.
LOAD_OP	140.3	BOOL	FALSE	If the in/out parameter "Load operating parameters to FM 355" is set, the operating parameters are loaded into the module and the in/out parameter is reset.

See also

Description DB 101 to 104 (Page 7219)

Description PID_FM (Page 7185)

FM 355-2 Temperature Control

FMT_PID

Description FMT_PID

During operation, the instruction FMT_PID is used to:

- Transfer operating parameters to FM 355-2
- Transfer controlller parameters to FM 355-2
- Read process values from FM 355-2

Operating parameters are all static variables of instance DB in the structure OP.

Controller parameters are all static variables of instance DB in the structure PAR.

Process values are all static variables of instance DB in the structure OUT.

The data required for the FMT_PID are stored in an instance DB on the CPU. The FMT_PID is program controlled to read data from the FM 355-2 and to write data to the FM 355-2.

Call

The instruction FMT_PID has to be called in the same OB as all the other instructions that access the same FM 355-2.

FMT_PID is called in the cyclic interrupt OB. It requires an initialization routine that is started by setting the COM_RST = TRUE parameter in the start-up of the CPU. After the initialization routine, FMT_PID sets the parameter COM_RST to FALSE.

If you call the instruction FMT_PID as a multiple instance DB, no commissioning interface is available. In this case you must commission the FM 355-2 via a watch table or with the instruction FMT_TUN.

Calling in Distributed I/O

The following has to be noted in the case of a distributed structure and simultaneous calling of FMT_PID and FMT_TUN:

- It is not permitted to set LOAD_OP simultaneously with FMT_PID and FMT_TUN.
- It is not permitted to set READ_OUT simultaneously with FMT_PID and FMT_TUN.

Reason: Both FBs access FM 355-2 via the same data records.

You must therefore ensure that only one of the two FBs reads or writes a data record at the same time.

Start-up

During CPU start-up and during CPU STOP-RUN transition, the parameters on the FM 355-2 are overwritten with the parameters from the SDB of the CPU.

Responses in the event of an error

The RET_VALU output parameter contains the 2nd and 3rd bytes of the STATUS parameter of the RDREC and WRREC instructions. RET_VALU can be evaluated if the READ_PAR and LOAD_PAR parameters are not reset.

When the instruction FMT_PID is called, an I/O access error can occur if the FM 355-2 is not plugged or is not supplied with voltage. In this case the CPU goes into STOP, if no OB 122 is loaded in the CPU.

If an error occurs when reading/writing a data set, QMOD_F and QCH_F must be set. In the case of parameter assignment errors, QPAR_F and QCH_F are set.

See also

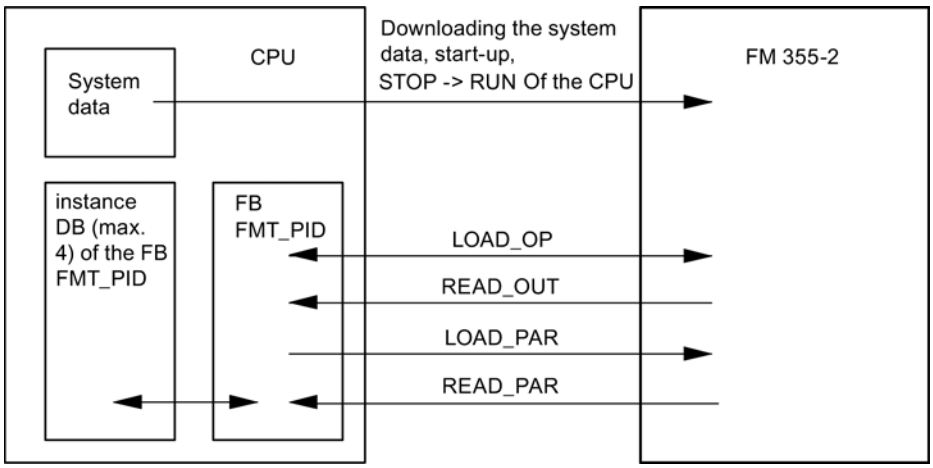
Block diagrams (Page 7238)

Mode of operation FMT_PID

Effect of parameters LOAD_OP, READ_OUT, LOAD_PAR and READ_PAR

In the initialization routine FMT_PID reads the parameters from the system data and stores them in its instance DB. Instance DB and system data are synchronized.

The following figure shows how the function of the instruction is influenced by the parameters LOAD_OP, READ_OUT, LOAD_PAR and READ_PAR.



The following table contains detailed information depending on the values of the parameters.

Parameters	TRUE	FALSE
LOAD_OP	The static variables of the structure OP are transferred cyclically FM 355-2 with the instructions WRREC and RDREC. The output parameters are read from the FM 355-2.	The static variables of the structure OP are transferred via the input and output areas of the module. This is not possible if two instances of the instruction FMT_PID access the same channel number of a module.
	READ_VAR is set to TRUE	READ_VAR is not changed
	In the following situations LOAD_OP is automatically TRUE: • If you have set one of the following parameters: TUN_ST, TUN_CST, SAVE_PAR,UNDO_PAR and LOAD_PID • If you modify TUN_ON • If you modify the setpoint SP_RE.	
	Longer run time	Shorter run time
	The values are effective after a cycle of the CPU or FM. The longer cycle is decisive.	A maximum of 3 to 4 cycles of the CPU or FM are required until the values are effective. The longer cycle is decisive

Parameters	TRUE	FALSE
	All operating parameters are transferred	Only the operating parameters up to and including LMN_DN are transferred. In automatic mode, only the setpoint SP_RE is transferred. In manual mode, only the manual manipulated value LMN_RE is transferred.
READ_OUT	The static variables of the structure OUT are read cyclically from FM 355-2 with the instructions WRREC and RDREC and stored in the structure OUT.	The static variables of the structure OUT are transferred via the input and output areas of the module and stored in the structure OUT.
	In the following situation READ_OUT is automatically TRUE: • During an optimization QTUN_ON = TRUE • When you modify the setpoint SP_RE	This is not possible if two instances of the instruction FMT_PID access the same channel number of a module.
	Longer processing time	Shorter processing time
	All parameters are read from FM 355	The following parameters are **not** read from FM 355 • SP (setpoint from the FM) • ER (error signal) • DISV (disturbance variable) • LMN_A • LMN_B • PHASE • STATUS_H • STATUS_C • STATUS_D • ZONE_TUN
READ_PAR	All static variables of the structure PAR are read from the FM 355-2 an stored in the instance DB on the CPU.	No transfer of the controller parameters to the CPU You can change the individual parameters in the user program.
LOAD_PAR	All static variables of the structure PAR are downloaded from the CPU to the FM 355-2. This is necessary if the control parameters on the CPU have changed based on the process states.	No transfer of the controller parameters to the FM 355-2

After data transfer has been completed, the parameters LOAD_OP, READ_OUT, LOAD_PAR and READ_PAR are reset to FALSE.

Use in distributed I/O

When the FM 355-2 is used in distributed I/O, the data transfer and the resetting of the parameters can take a few call cycles.

Processing times FMT_PID

During program controlled re-configuring (LOAD_PAR, LOAD_OP) of the FM 355-2 by FMT_PID, its runtime is extended. The new parameters are always immediately effective.

Boundary conditions			Processing time in	
READ_OUT	LOAD_OP	LOAD_PAR READ_PAR	CPU 315-2-DP	CPU 416-2 DP
FALSE	FALSE	FALSE	0.65 ms	0.04 ms
TRUE	FALSE	FALSE	2.85 ms	0.30 ms
TRUE	TRUE	FALSE	4.56 ms	0.55 ms
FALSE	FALSE	TRUE	3.58 ms*)	0.30 ms
TRUE	FALSE	TRUE	5.8 ms*)	0.56 ms
TRUE	TRUE	TRUE	7.41 ms*)	0.82 ms
*) If READ_PAR is set instead of LOAD_PAR, the CPU 315-2 DP processing time is increased by 0.35 ms.				

See also

Block diagrams (Page 7238)

Input parameters FMT_PID

Table 9-233

Parameters	Address	Data type	Default	Description
MOD_ADDR	0.0	INT	256	At this input you will find the module address resulting from configuration under STEP 7.
CHANNEL	2.0	INT	0	Number of the controller channel to which the instance DB is referenced. Values from 0 to 3 are permitted

See also

Block diagrams (Page 7238)

Output parameters FMT_PID

Table 9-234

Parameters	Address	Data type	Default	Description
QMOD_F	4.0	BOOL	FALSE	The instruction checks that the reading and writing of a data record has been completed. If errors are identified, the QMOD_F output is set. The cause of errors can be: An incorrect module address on the MOD_ADDR parameter, an incorrect channel number on the CHANNEL parameter or a defective module.
RET_VALU	6.0	WORD	W#16#0	RET_VALU contains the 2nd and 3rd bytes from the STATUS parameter of the instructions RDREC and WRREC.

See also

RDREC: Read data record (Page 6459)
WRREC: Write data record (Page 6461)

In/out parameters FMT_PID

Table 9-235

Parameters	Address	Data type	Default	Description
COM_RST	8.0	BOOL	FALSE	If COM_RST = TRUE, the instruction FMT_PID executes an initialization routine and resets COM_RST. This initialization routine is imperative every time the CPU starts up.
LOAD_OP	8.1	BOOL	FALSE	If the in/out parameter LOAD_OP is set, the operating parameters are downloaded to the module, the output parameters are read and the in/out parameters are reset.
READ_OUT	8.2	BOOL	FALSE	If the in/out parameter READ_OUT is set, the output parameters are read from the module and stored in the OUT structure of the instance DB and the I/O parameters are reset.
LOAD_PAR	8.3	BOOL	FALSE	If the in/out parameter LOAD_PAR is set, the controller parameters are downloaded to the module and the in/out parameter is reset.
READ_PAR	8.4	BOOL	FALSE	If the parameter READ_PAR = TRUE is set, then the controller parameters are read from FM 355-2 and stored in the PAR structure of the instance DB. The in/out parameter is then reset.

See also

Block diagrams (Page 7238)

Static variables FMT_PID.OP

Table 9-236

Parameters	Address	Data type	Default	Description
vers_nr	10.0	WORD	W#16#3230	The vers_nr parameter may not be modified by users. It identifies the start of the operating parameters that are transferred to the module, if LOAD_OP = TRUE is set.
SP_RE	12.0	REAL	0.0	An external setpoint is interconnected with the controller at input SP_RE.
LMN_RE	16.0	REAL	0.0	An external manipulated value is interconnected with the controller at input LMN_RE. Values from -100.0 to 100.00 (%) are permitted.
SAFE_ON	20.0	BOOL	FALSE	When SAFE_ON is set, a safety value is applied as manipulated value.
LMNTRKON	20.1	BOOL	FALSE	If LMNTRKON is set, the manipulated value is corrected to an analog input (AI) (This does not apply to step controllers without analog position feedback)
LMN_REON	20.2	BOOL	FALSE	If LMN_REON is set, the external manipulated value LMN_RE is applied (manual mode).

Parameters	Address	Data type	Default	Description
LMNRHSRE	20.3	BOOL	FALSE	The "Control valve at high endstop" signal can be interconnected with a digital input of the FM 355-2 or at the LMNRHSRE input. LMNRHSRE = TRUE means: The control valve is at high endstop. (Applies to step controllers only)
LMNRLSRE	20.4	BOOL	FALSE	The "Control valve at high endstop" signal can be interconnected with a digital input of the FM 355-2 or at the LMNRLSRE input. LMNRLSRE = TRUE means: The control valve is at the low endstop. (Applies to step controllers only)
LMNS_ON	20.5	BOOL	FALSE	Enable manipulated value operation (Applies to step controllers only; manual mode).
LMN_UP	20.6	BOOL	FALSE	Manipulated value signal up operation (Applies to step controllers only).
LMN_DN	20.7	BOOL	FALSE	Manipulated value signal down operation (Applies to step controllers only).
SAVE_PAR	22.0	BOOL	FALSE	Saves the current contoller parameters to the output parameters of the instruction FMT_TUN. • FMT_TUN.SAV_PFAC=PFAC_SP • FMT_TUN.SAV_GAIN =GAIN • FMT_TUN.SAV_TI =TI • FMT_TUN.SAV_TD = TD • FMT_TUN.SAV_D_F =D_F • FMT_TUN.SAV_CONZ = CON_ZONE • FMT_TUN.SAV_RATI =RATIOFAC • FMT_TUN.SAV_CZON=CONZ_ON • FMT_TUN.SAV_PSEL=P_SEL
UNDO_PAR	22.1	BOOL	FALSE	The saved controller parameters are retrieved again, also in automatic mode. • PFAC_SP= FMT_TUN.SAV_PFAC • GAIN = FMT_TUN.SAV_GAIN • TI = FMT_TUN.SAV_TI • TD = FMT_TUN.SAV_TD • D_F = FMT_TUN.SAV_D_F • CON_ZONE = FMT_TUN.SAV_CONZ • RATIOFAC = FMT_TUN.SAV_RATI • CONZ_ON= FMT_TUN.SAV_CZON • P_SEL= FMT_TUN.SAV_PSEL
LOAD_PID	22.2	BOOL	FALSE	If PID_ON=TRUE, it loads the PID parameters (PID_GAIN, PID_TI, PID_TD). If PID_ON=FALSE, the PI parameters (PI_GAIN, PI_TI) are loaded. (Even in automatic mode)
TUN_ON	22.3	BOOL	FALSE	TUN_ON=TRUE switches the FM 355-2 to ready for optimization (PHASE=1): The manipulated value is averaged until the manipulated value excitation is applied.

Parameters	Address	Data type	Default	Description
TUN_ST	22.4	BOOL	FALSE	If the setpoint is to remain constant during controller optimization at the operating point, a manipulated value step-change by the amount of TUN_DLMN is enabled by TUN_ST=TRUE (PHASE = 1 or PHASE = 2)
TUN_CST	22.5	BOOL	FALSE	Starts cooling optimization by applying a manipulated manipulated value step-change in the amount of TUN_CLMN (PHASE = 1 or PHASE = 2)

See also

Description FMT_PID (Page 7226)

Block diagrams (Page 7238)

Static variables FMT_PID.PAR

Table 9-237

Parameters	Address	Data type	Default	Description
vers_nr	24.0	WORD	W#16#3230	The vers_nr parameter may not be modified by users. It identifies the start of the controller parameters, which are read from the FM with READ_PAR = TRUE and stored in the instance DB or transferred to the FM with LOAD_PAR = TRUE.
P_SEL	26.0	BOOL	TRUE	It is possible to enable or disable PID actions individually in the PID algorithm. The P-action is enabled if the input P_SEL is set.
MONERSEL	26.1	BOOL	FALSE	The controller possesses a limit value detector that can be applied either for the process value or for the error signal. If MONERSEL is set, the error signal is monitored.
PID_ON	26.2	BOOL	TRUE	After optimization or in the case of LOAD_PID =TRUE, the PID parameters are enabled. If PID_ON =FALSE, the PI parameters are loaded. With certain process types it is nevertheless possible that only a PI controller will be designed despite PID_ON = TRUE.
CONZ_ON	26.3	BOOL	FALSE	TRUE = control zone enabled FALSE = control zone off
D_EL_SEL	28.0	INT	16	The D-action element in the PID algorithm can be assigned to a separate input This is selected at the D_EL_SEL parameter. • D_EL_SEL = 0 to D_EL_SEL = 3: Analog input 0 to 3 • D_EL_SEL = 16: error signal • D_EL_SEL = 17: negative process value
SP_HLM	30.0	REAL	100.0	The setpoint is always limited to a high and a low limit. The SP_HLM parameter specifies the high limit. SP_HLM > SP_LLM
SP_LLM	34.0	REAL	0.0	The setpoint is always limited to a high and a low limit. The SP_LLM parameter specifies the low limit. SP_LLM < SP_HLM
H_ALM	38.0	REAL	100.0	Highest limit for monitoring the process value or the error signal. H_ALM > H_WRN

Parameters	Address	Data type	Default	Description
H_WRN	42.0	REAL	90.0	Second to highest limit for monitoring the process value or the error signal. H_WRN > H_ALM
L_WRN	46.0	REAL	10.0	Second to lowest limit for monitoring the process value or the error signal. Values from H_WRN to L_ALM are permitted.
L_ALM	50.0	REAL	0.0	Lowest limit for monitoring the process value or the error signal. Values from L_ALM to L_WRN are permitted.
HYS	54.0	REAL	1.0	The parameters can be assigned for a hysteresis to prevent the monitoring displays from flickering. HYS >= 0.0
DEADB_W	58.0	REAL	0.0	A dead band is applied to the error signal. The DEADB_W parameter determines the size of the dead band. DEADB_W >= 0.0
PFAC_SP	62.0	REAL	1.0	PFAC_SP specifies the effective P-action when there is a setpoint change. Values from 0.0 to 1.0 are permitted.
GAIN	66.0	REAL	1.0	The GAIN parameter specifies the controller gain. The control direction is inverted by means of a negative sign of GAIN.
TI	70.0	REAL	3000.0	The TI parameter (integral-action time) determines the time response of the integrator. If TI = 0.0, the integral action is disabled. TI = 0.0 or TI >= 0.5
TD	74.0	REAL	0.0	The TD (rate time) paramter determines the time response of the derivative action (D-action element). If TD = 0.0, the derivative action is disabled TD = 0.0 or TD >= 1.0
D_F	78.0	REAL	5.0	The time constant TD/D_F that is effective for the D-action element is limited internally to ≥ sampling time/2. Values from 5.0 to 10.0 are permitted.
LMN_SAFE	82.0	REAL	0.0	A safety value can be parameterized for the manipulated value. Values from -100.0 to 100.0 (%) are permitted.
LMN_HLM	86.0	REAL	100.0	The manipulated value is always limited to a high and a low limit. The LMN_HLM parameter specifies the high limit. (This does not apply to step controllers without analog position feedback). Values from LMN_LLM to 100.0 (%) are permitted.
LMN_LLM	90.0	REAL	0.0	The manipulated value is always limited to a high and a low limit. The LMN_LLM parameter specifies the low limit. (This does not apply to step controllers without analog position feedback). Values from -100.0 to LMN_HLM (%) are permitted.
MTR_TM	94.0	REAL	60.0	Runtime of the control valve from stop to stop (Applies to step controllers only). MTR_TM >= 0.001
PULSE_TM	98.0	REAL	0.0	Minimum pulse duration (Applies to step controllers or pulse controllers only)
BREAK_TM	102.0	REAL	0.0	Minimum pulse duration (Applies to step controllers or pulse controllers only)

Parameters	Address	Data type	Default	Description
RATIOFAC	106.0	REAL	0.0	Ratio of heating gain/cooling gain for the process. RATIOFAC <> 0.0 has the following effect: • LMN = LMN * RATIOFAC for LMN<0.0; • CON_ZONE 50 % greater Values from 0.01 to 100.0 or 0.0 are permitted.
CON_ZONE	110.0	REAL	100.0	Control zone width If ER >= CON_ZONE, then LMN = LMN_HLM. If ER <= -CON_ZONE, then LMN = LMN_LLM. If RATIOFAC <> 0.0 and ER <= -CON_ZONE/RATIOFAC, then LMN = LMN_LLM. CON_ZONE > 0.0
TUN_DLMN	114.0	REAL	20.0	Process excitation for controller optimization results from a manipulated value step-change of TUN_DLMN. Values from -100.0 to 100.0 (%) are permitted.
TUN_CLMN	118.0	REAL	-20.0	Manipulated variable step-change after start of cooling optimization by TUN_CST. Values from -100.0 to 100.0 (%) are permitted.

See also

Block diagrams (Page 7238)

Static variables FMT_PID.OUT

Table 9-238

Parameters	Address	Data type	Default	Description
vers_nr	122.0	WORD	W#16#3230	The vers_nr parameter may not be modified by users. It marks the start of the output parameter that is read by the module if READ_OUT = TRUE is set.
PV	124.0	REAL	0.0	Effective process value.
LMN	128.0	REAL	0.0	The effective output value is output at the LMN output. In the case of step controllers without analog position feedback, the unlimited P- + D-action is output at the LMN output. Values from -100.0 to 100.0 (%) are permitted.
QLMNSAFE	132.1	BOOL	FALSE	If the QLMNSAFE output is set, the safety output value is output as output value.
QLMNTRK	132.3	BOOL	FALSE	Output QLMNTRK indicates if the output value is being corrected via an analog input.
QLMN_RE	132.4	BOOL	FALSE	The QLMN_RE output indicates if LMN_REON is set and the external output value LMN_RE is applied as the output value.
QLMNR_HS	132.5	BOOL	FALSE	QLMNR_HS = TRUE means: The control valve is at high endstop. (Applies to step controllers only)
QLMNR_LS	132.6	BOOL	FALSE	QLMNR_LS = TRUE means: The control valve is at the low endstop. (Applies to step controllers only)

Parameters	Address	Data type	Default	Description
QLMNR_ON	132.7	BOOL	FALSE	QLMNR_ON = TRUE means: Step controller with position feedback.
QSTEPCON	133.0	BOOL	FALSE	Step controller / pulse controller • QSTEPCON = 0: Pulse controller or continuous controller • QSTEPCON = 1: Step controller
QSPR	133.1	BOOL	FALSE	If the QSPR output is set, the continuous controller operates in split-range mode.
QMAN_FC	133.3	BOOL	FALSE	This controller is a master controller. It is manually followed up to the process value of a slave controller, or its I-action is disabled because the setpoint or manipulated variable of the slave controller has reached the limit.
QH_ALM	134.0	BOOL	FALSE	High limit H_ALM overshot by process value or error value.
QH_WRN	134.1	BOOL	FALSE	Process value or control deviation exceeds second to highest limit H_WRN.
QL_WRN	134.2	BOOL	FALSE	Process value or control deviation has dropped below the second to lowest limit L_WRN.
QL_ALM	134.3	BOOL	FALSE	Process value or control deviation has dropped below the lowest limit L_ALM.
QLMN_HLM	134.4	BOOL	FALSE	The QLMN_HLM output reports that the high output value limit LMN_HLM has been reached. (This does not apply to step controllers without analog position feedback).
QLMN_LLM	134.5	BOOL	FALSE	The QLMN_LLM output reports that the low output value limit LMN_LLM has been reached. (This does not apply to step controllers without analog position feedback)
QPAR_F	134.6	BOOL	FALSE	The module checks the reliability of the parameters. A parameter assignment error is indicated at output QPAR_F.
QCH_F	134.7	BOOL	FALSE	The QCH_F output is set if the controller channel is unable to supply any valid results. A channel error (for example, wire break) is also set if QPAR_F = TRUE or QMOD_F = TRUE. If QCH_F = TRUE, then the precise error information in the diagnostic data record DS1 of the module is read out.
QUPRLM	135.0	BOOL	FALSE	The setpoint is limited to a positive and negative slew rate. If the QUPRLM output is set, the setpoint rise is limited.
QDNRLM	135.1	BOOL	FALSE	The setpoint is limited to a positive and negative slew rate. If the QDNRLM output is set, the setpoint decay is limited.
QSP_HLM	135.2	BOOL	FALSE	The QSP_HLM output reports that the high setpoint limit SP_HLM has been reached.
QSP_LLM	135.3	BOOL	FALSE	The QSP_LLM output reports that the low setpoint limit SP_LLM has been reached.
QLMNUP	135.4	BOOL	FALSE	"Output value signal up" output (Applies to step controllers or pulse controllers only)
QLMNDN	135.5	BOOL	FALSE	"Output value signal down" output (Applies to step controllers or pulse controllers only)
QTUN_ON	135.6	BOOL	FALSE	QTUN_ON = TRUE indicates that tuning is in progress.
PAR_ACT	135.7	BOOL	FALSE	Updating controller parameters

Parameters	Address	Data type	Default	Description
SP	136.0	REAL	0.0	Effective setpoint
ER	140.0	REAL	0.0	Effective control deviation (SP- PV)
DISV	144.0	REAL	0.0	Effective disturbance variable Values from -100.0 to 100.0 (%) are permitted.
LMN_A	148.0	REAL	0.0	Continuous or pulse controllers display the output value A of the split-range function at the LMN_A output; step controllers with analog position feedback display the position feedback there. Values from -100.0 to 100.0 (%) are permitted.
LMN_B	152.0	REAL	0.0	Continuous or pulse controllers display the output value B of the split-range function at output LMN_B. Values from -100.0 to 100.0 (%) are permitted.
PHASE	156.0	INT	0	Indicates the controller tuning phases. • PHASE = -1: Canceling of controller tuning, for example with parallel tuning of several channels. The value lasts for a few cycles and is used as cancelation signal between the instruction FMT_PID and the module. • PHASE = 0: No tuning mode; automatic or manual mode • PHASE = 1: Ready to start tuning; check parameters, wait for excitation, measure the sampling times • PHASE = 2: Actual tuning: Searching for point of inflection with constant output value. • PHASE = 3: Calculating process parameters. Saving valid controller parameters prior to tuning. • PHASE = 4: Controller design • PHASE = 5: Following up the controller to the new manipulated variable • PHASE = 6: Following up the controller to the new manipulated variable • PHASE = 7: Check of process type, if process type II or III was determined (during heating optimiation only).
STATUS_H	158.0	INT	0	Parameters STATUS_H displays a diagnostic value via the search for the point of inflection during heating process.
STATUS_C	160.0	INT	0	Parameters STATUS_C displays a diagnostic value via the search for the point of inflection during cooling process.
STATUS_D	162.0	INT	0	Parameters STATUS_D indicates a diagnostic value of the controller design during the heating process.
ZONE_TUN	164.0	WORD	W#16#0	In HEX code, each of the four digits represents one channel, arranged in the following order from left to right: Channel 0, 1, 2, 3. ZONE_TUN = 0 means that the selected channel will not be tuned as a group with other channels. A value <> 0000 shows - in each case with a 1 - the channels that will be tuned as a group.

See also

Block diagrams (Page 7238)

Block diagrams

Overview

The following figures show at which points the parameters of the instruction FMT_PID act.

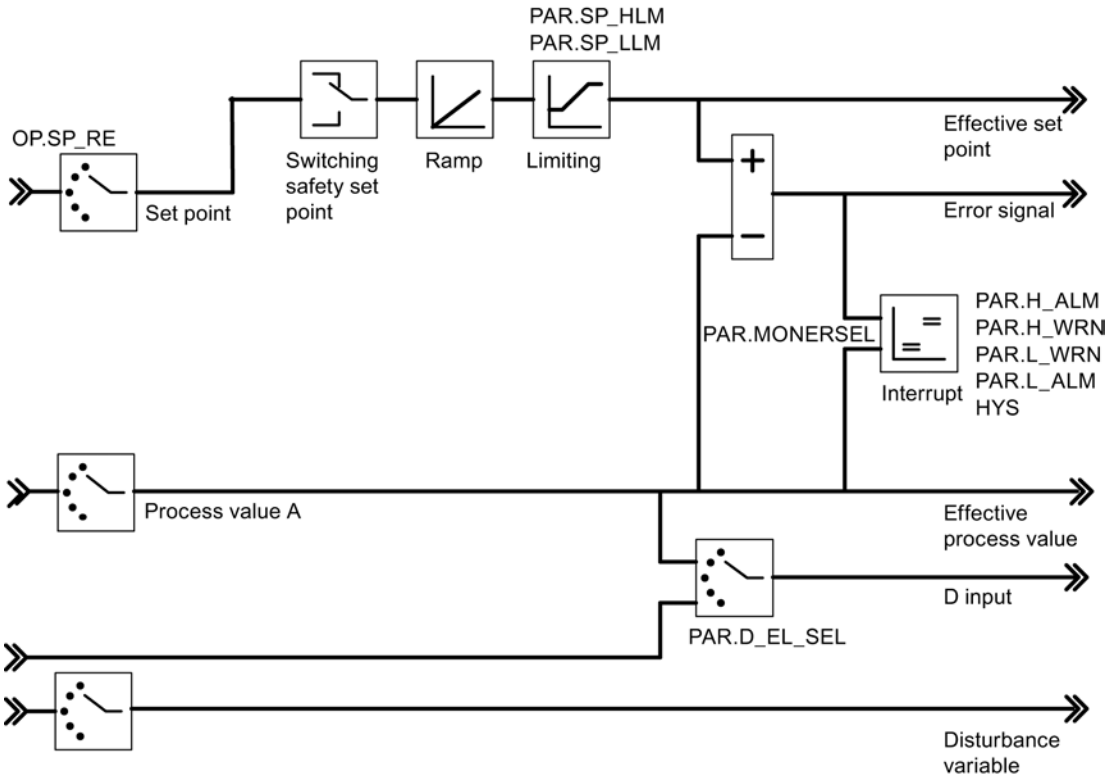
As far as three-component controllers and ratio/mixing controllers are concerned, the parameters affect the same point as with fixed setpoint or cascade controllers. This also applies to the parameters that are equally in place for continuous controllers, controllers with pulse output as well as step controllers. It is generally correct that identical buttons also contain identical parameters. For reasons of clarity, not all structure diagrams are depicted, thereore, and not all parameters are marked in all figures.

However, the parameters of the instruction FMT_PID are contained in all the figures – with the exception of the parameters MOD_ADDR, CHANNEL, QMOD_F, QPAR_F, QCH_F, QLMNR_ON, RET_VALU, COM_RST, LOAD_PAR, READ_PAR, READ_OUT and LOAD_OP.

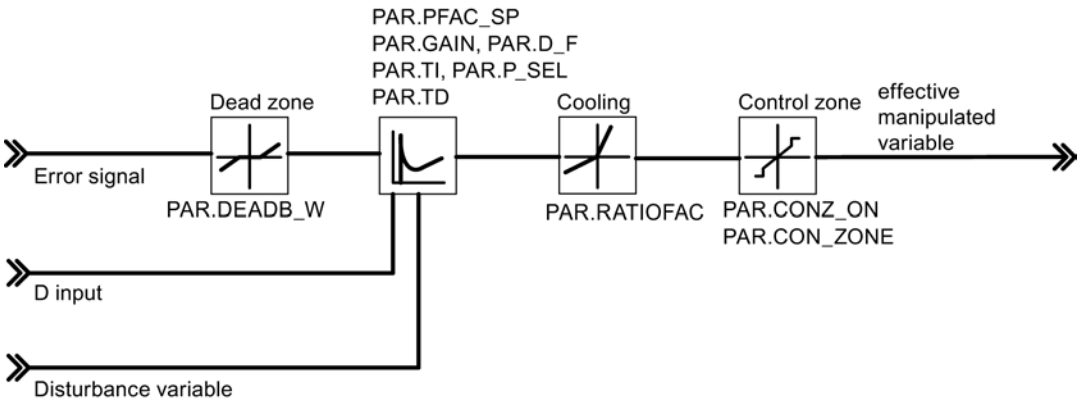
The following figures show the places in the module affected by the parameters of the FMT_PID.

Action of the input parameters

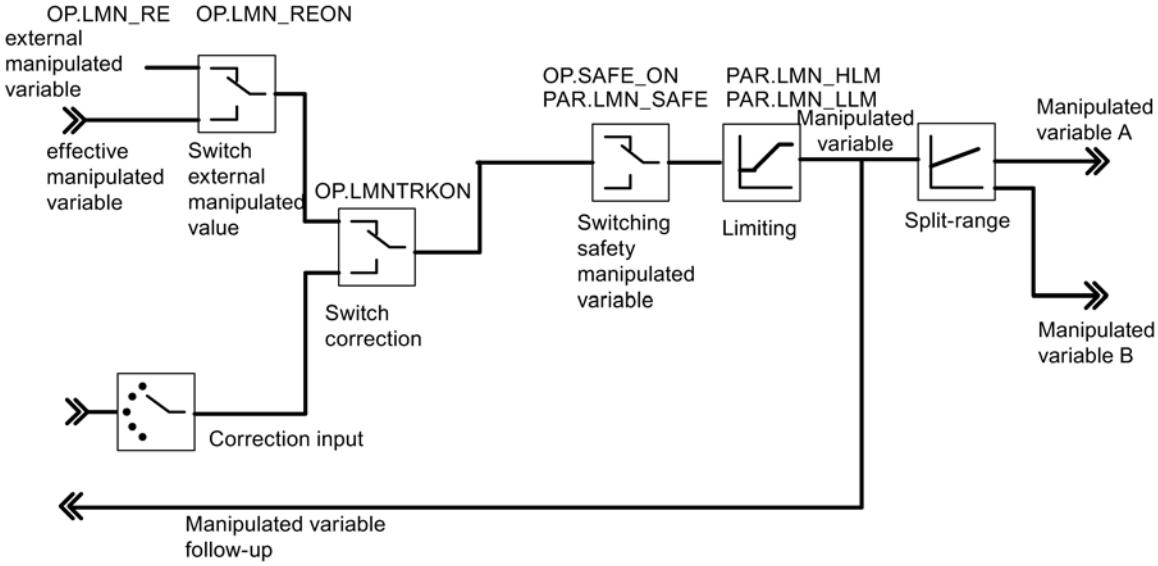
The following figure shows the control deviation for a fixed-value or cascade controller.



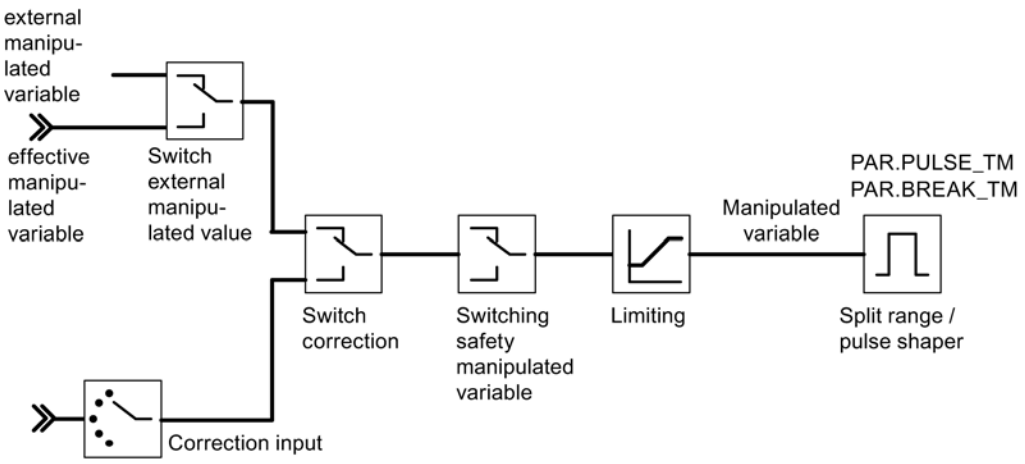
The following figure shows the block diagram of the control algorithm.



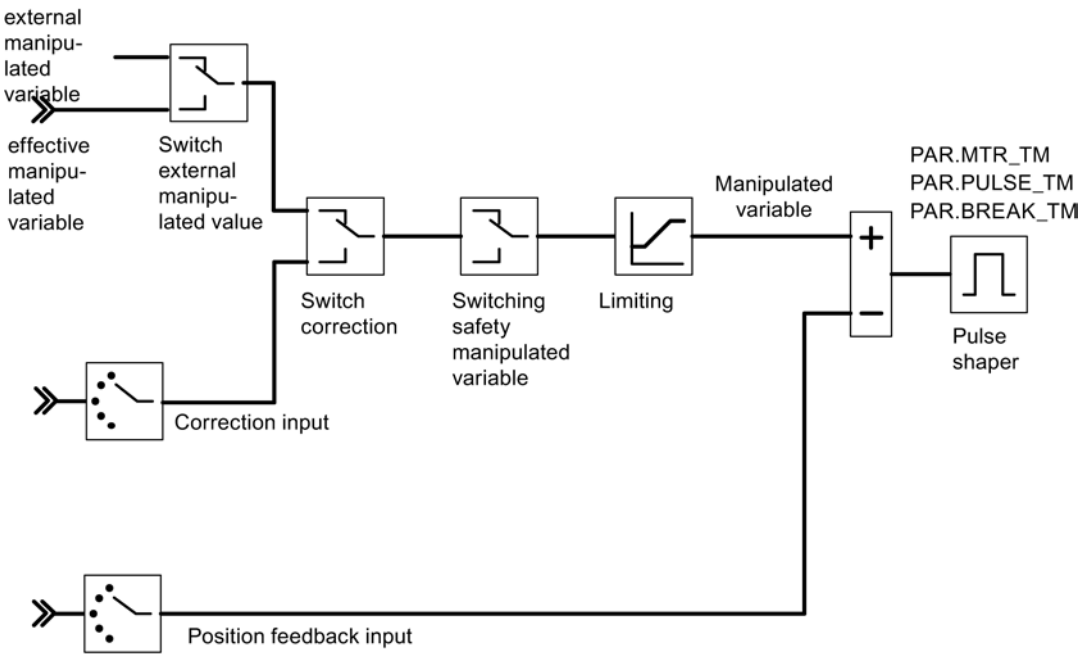
The following figure shows the controller output of the continuous controller (FM 355-2 C).



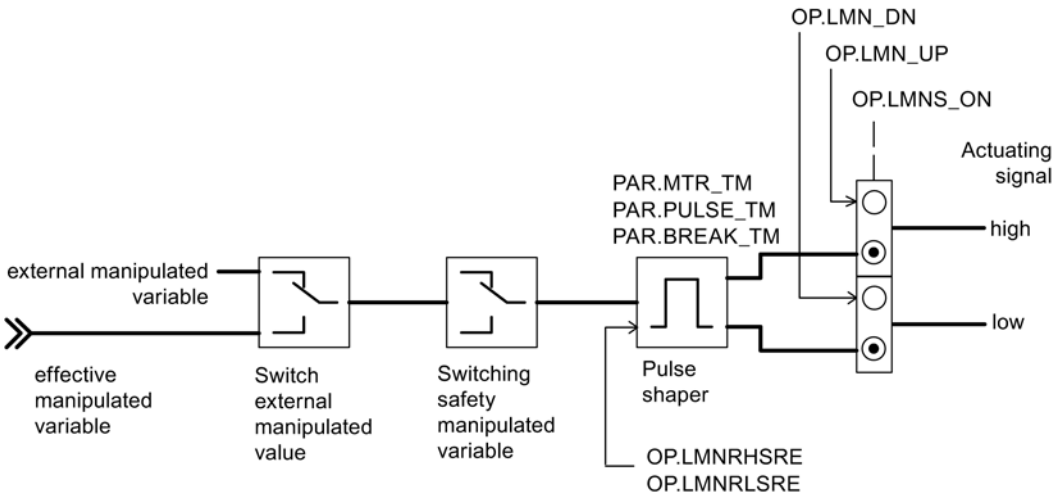
The following figure shows the controller output of the pulse controller (FM 355-2 S).



The following figure shows the controller output of the step controller with position feedback (FM 355-2 S).



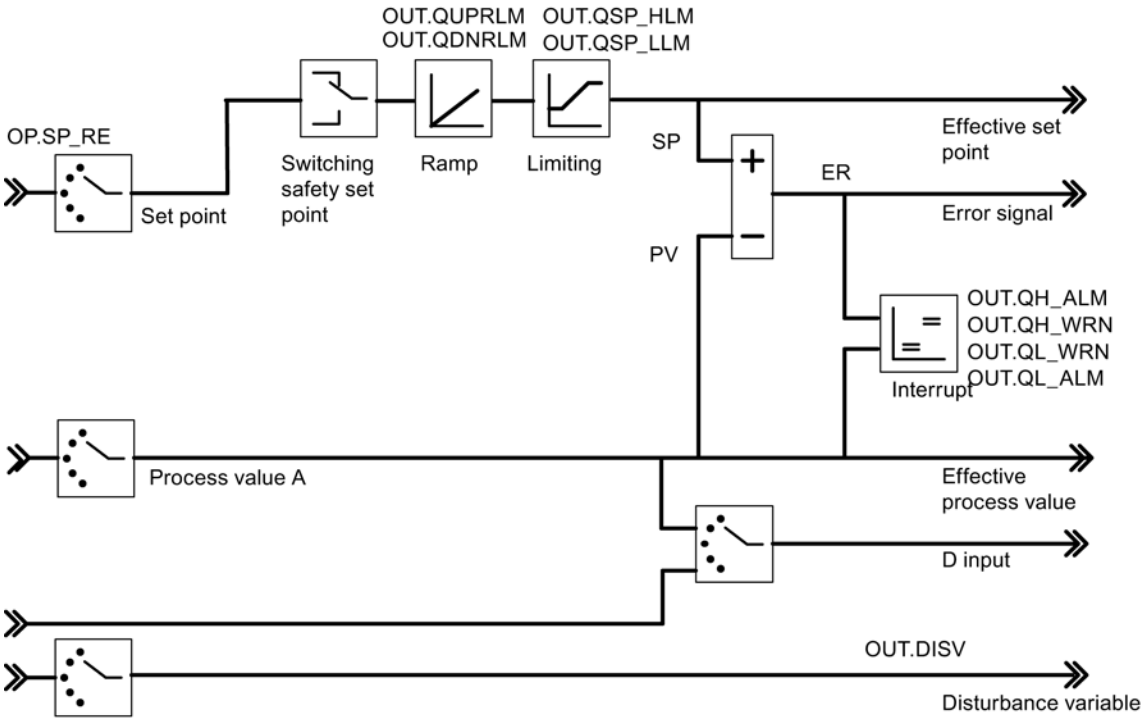
The following figure shows the controller output of the step controller without position feedback (FM 355-2 S).



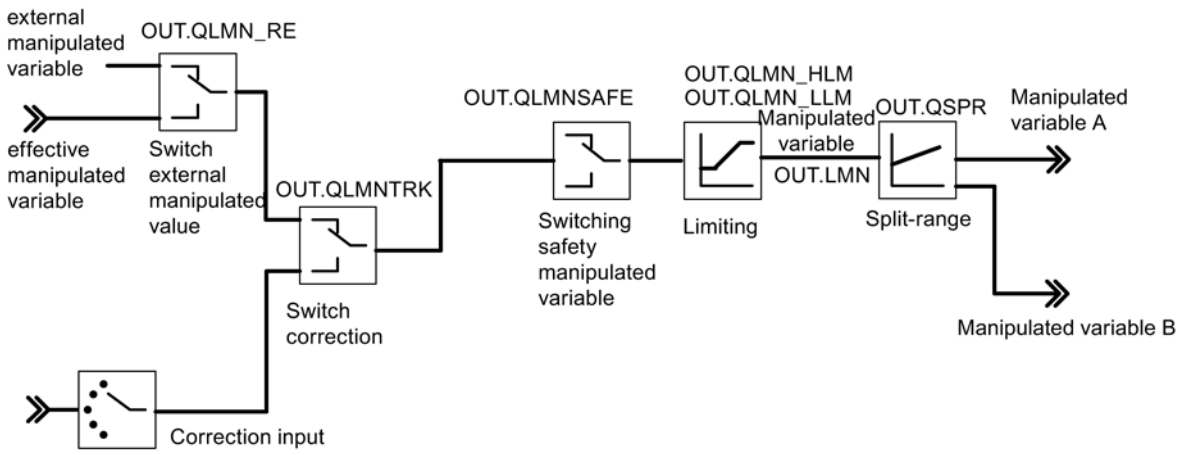
Generating output parameters

The following figures show at which points in the module the output parameters of the instruction FMT_PID are generated.

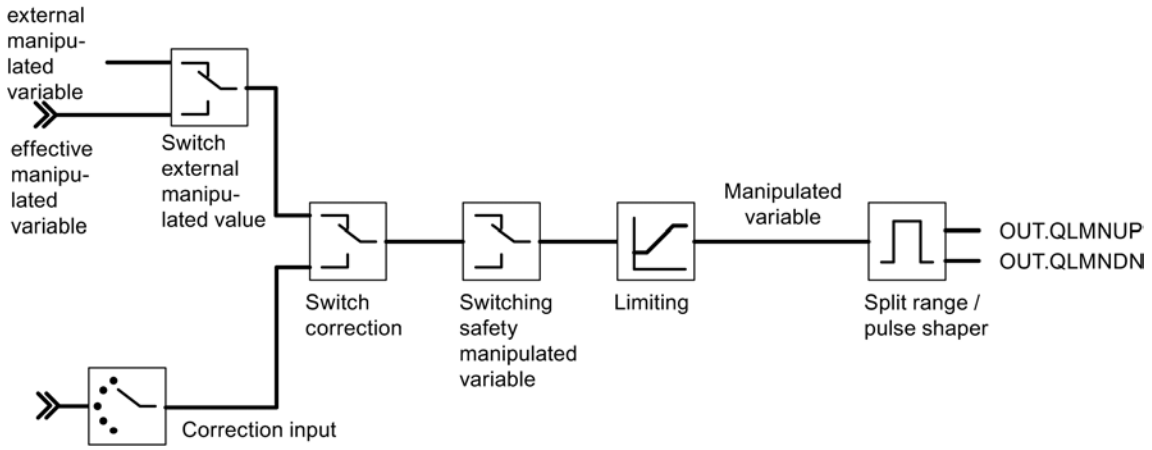
The following figure shows the control deviation for a fixed-value or cascade controller.



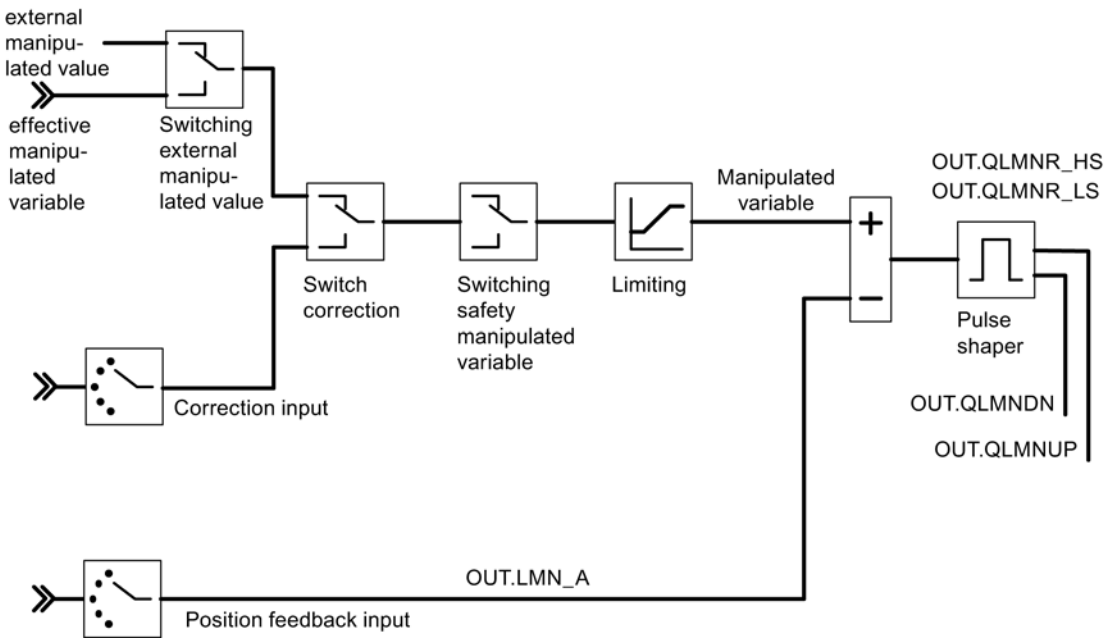
The following figure shows the controller output of the continuous controller (FM 355-2 C).



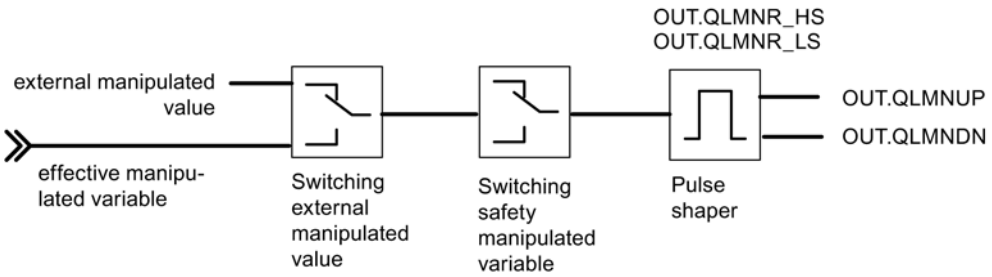
The following figure shows the controller output of the pulse controller (FM 355-2 S).



The following figure shows the controller output of the step controller with position feedback (FM 355-2 S).



The following figure shows the controller output of the step controller without position feedback (FM 355-2 S).



See also

Description FMT_PID (Page 7226)

Parameters STATUS_H

STATUS_H indicates a diagnostic value via the search for the point of inflection during the heating process.

If a diagnostic value 2xxxx is displayed, optimization is completed but the controller parameters are uncertain.

If a diagnostic value 3xxxx is displayed, the optimization is canceled (PHASE = 0 and TUN_ON = FALSE) and the controller switches to the previous operating mode.

STATUS_ H	Description	Remedy
0	Default or no or still no new controller parameters	
10000	Optimization completed + suitable controller parameters found	
2xxxx	Optimization completed + controller parameters uncertain	
2xxx1	effective motor actuating time >= 65% of the point of inflection time T_P_INF	Compensation of TU and T_P_INF is not performed any more. This leads to the design of a softer controller
2xx2x	Point of inflection not reached (only with excitation via setpoint step-change)	If the controller oscillates, reduce the controller parameters or repeat test with lower TUN_DLMN.
2x1xx	Estimation error TU <= 3*Sampling time	Repeat test.
2x2xx	Serious estimation error (TU < sampling time)	Repeat test. Special case PT1-only process: Do not repeat the test; reduce the controller parameters if necessary
2x3xx	Estimation error TU too high	Repeat test under better conditions.
21xxx	Estimation error N_PTN < 1	Repeat test under better conditions.
22xxx	Estimation error N_PTN > 10	Repeat test under better conditions.
3xxxx	Optimization interrupted in phase 1 due to incorrect parameter assignment	
30001	Simultaneous setting of TUN_ON and TUN_ST	Restart optimization.
30002	|TUN_DLMN| < 5 % or value of effective LMN modification < 5 % • Transition in phase 99 (Phase 0 in a parallel optimization process) • Output LMN = LMN0	Either you configured for TUN_DLMN a value that is too low, or the manipulated variable was near to a control limit prior to start of optimization. In this case, you must prevent the controller from settling to the new setpoint and from unnecessary leaving the stationary operating point (not possible with parallel optimization process). You should now proceed as follows: • If necessary, cancel setpoint and set TUN_ON to FALSE. • TUN_DLMN correct. • If TUN_DLMN >= 5%, check manipulated value limits. • Restart optimization.
30004	|TUN_DLMN| < 5 % or effective LMN modification < 5 % The effective manipulated value difference limited by split–range limits and not by manipulated value limits. • Transition in phase 99 (Phase 0 in a parallel optimization process) • Output LMN = LMN0	See STATUS_H=30002: Also take into account that, for example, with LMN_A < 5 % a heating optimization with negative TUN_DLMN is not possible (reason: the cooling power must not be adjusted).
30009	Safety mode	Close safety mode and restart optimization.

Note

At the end of phase 1, STATUS_H = 0 is reset.

If you cancel optimization in phase 2, STATUS_H = 0. However, STATUS_D still displays the status of the last controller calculation.

See also

Block diagrams (Page 7238)

Parameters STATUS_C

STATUS_C indicates a diagnostic value via the search for the point of inflection during cooling process.

If a diagnostic value 2xxxx is displayed, optimization is completed, but the controller parameters are uncertain.

If a diagnostic value 3xxxx is displayed, the optimization is canceled (PHASE = 0 and TUN_ON = FALSE) and the controller switches to the previous operating mode.

STATUS_ C	Description	Remedy
0	Default or no or still no new controller parameters	
10000	Optimization completed + suitable controller parameters found	
2xxxx	Optimization completed + controller parameters uncertain	
2x3xx	Estimation error TU too high	Repeat test under better conditions.
21xxx	Estimation error N_PTN < 1	Repeat test under better conditions.
22xxx	Estimation error N_PTN > 10	Repeat test under better conditions.
3xxxx	Optimization interrupted in phase 1 due to incorrect parameter assignment	
30001	Simultaneous setting of TUN_ON and TUN_CST	Restart optimization.
30002	Only for step controllers without analog position feedback: |TUN_CLMN| < 5 % or value of effective LMN modification < 5 % • Transition to phase 0 • TUN_ON = FALSE	Either you configured for TUN_CLMN a value that is too low, or the manipulated variable was near to a control limit prior to start of optimization. In this case, you must prevent the controller from settling to the new setpoint and from unnecessary leaving the stationary operating point (not possible with parallel optimization process). You should now proceed as follows: • If necessary cancel setpoint and set TUN_ON to FALSE. • TUN_CLMN correct. • Check the manipulated value limits, if | TUN_CLMN| >= 5% and the sign was correct. • Restart optimization.

STATUS_ C	Description	Remedy
30003	TUN_CLMN <= -5%, butLMN_LLM > -5%. • Transition to phase 0 • TUN_ON = FALSE	The low limit LMN_LLM is too high (e.g. 0%) and therefore prevents the output of cooling power. • Correct low limit LMN_LLM. • Restart optimization.
30004	TUN_CLMN > -5 % or effective LMN modification > -5 % The effective manipulated value difference limited by split–range limits and not by manipulated value limits. • Transition to phase 0 • TUN_ON = FALSE	See STATUS_C=30002: Also take into account that, for example, with LMN_A < 5 % a heating optimization with negative TUN_DLMN is not possible (reason: the cooling power must not be adjusted).
30008	TUN_CST but without previous heating optimization • Transition to phase 0 • TUN_ON = FALSE	First start heating optimization
30009	Safety mode	Close safety mode and restart optimization.

See also

Block diagrams (Page 7238)

Parameters STATUS_D

STATUS_D	Description
0	No controller parameters have been calculated.
110	N_PTN <= 1.5 Process type I fast
121	N_PTN > 1.5 Process type I
122	N_PTN = 1.9 process type I after phase 7 (before N_PTN > 1.9)
200	N_PTN > 1.9 Process type II (transition range)
310	N_PTN >= 2.1 Process type III fast
320	N_PTN > 2.6 Process type III

Note

If you interrupt optimization in phase 2, STATUS_H = 0 is. However, STATUS_D still displays the status of the last controller calculation.

The higher the value of STATUS_D, the higher the order of the control process, the greater the ratio TU/TA and the gentler the controller parameters will be.

See also

Block diagrams (Page 7238)

FMT_PAR

Description FMT_PAR

The instruction FMT_PAR is used for online modification of parameters that cannot be defined via instruction FMT_PID.

Call

FMT_PAR has to be called in the same OB as all the other instructions that access the same FM 355-2.

To save runtime, FMT_PAR should only be called using LOAD=TRUE, if parameters are to be modified.

Operating principle

With FMT_PAR, you can modify one REAL-parameter and one of the INT parameters for each call.

At Parameters INDEX specify the index numbers of the parameter you want to modify. At the input parameter VALUE_R or VALUE_I, specify the new values.

If you want to modify several parameters, you must call the same instance DB several times in succession using LOAD_PAR = TRUE and different index numbers.

If FM 355-2 is used in distributed I/O, it may take a few call cycles until the parameters are applied to FM 355-2. As long as the transfer has not been completed, parameter LOAD_PAR retains the value TRUE. When modifying parameters, keep repeating the call of instruction FMT_PAR until LOAD_PAR = FALSE is.

Note

Please note that parameters, modified using FMT_PAR,are overwritten at startup of CPU with the parameters from the system data.

Example

You want to modify in operating mode the startup time of the ramp for the reference variable as well as,depending on the process state, use different analog input values as process value.

- To configure the startup time for the reference variable at 10.0 call FMT_PAR in the current mode with INDEX = 30 and VALUE_R = 10.0.
- If you want to configure the analog input value 4 of the module as process value, call FMT_PAR with INDEX = 50 and VALUE_I = 4 during operation.

Input parameters FMT_PAR

Table 9-239

Parameters	Address	Data type	Default	Description
MOD_ADDR	0.0	INT	256	The module address that results from the configuration with STEP 7 is set at this input.
CHANNEL	2.0	INT	0	Number of controller channel to which the instance DB refers. Values from 0 to 3 are permitted.
INDEX	4.0	INT	0	0 to 100
VALUE_R	6.0	REAL	0.0	Depending on respective parameter
VALUE_I	10.0	INT	0	Depending on respective parameter

Output parameters FMT_PAR

Table 9-240

Parameters	Address	Data type	Default	Description
RET_VALU	12.0	WORD	W#16#0	RET_VALU contains the 2nd and 3rd bytes from parameter STATUS of instruction WRREC.

See also

WRREC: Write data record (Page 6461)

In/out parameters FMT_PAR

Table 9-241

Parameters	Address	Data type	Default	Description
LOAD_PAR	+14.0	BOOL	FALSE	If the in/out parameters LOAD_PAR is set, VALUE_R or VALUE_I are written on the module.INDEX indicates which parameter is overwritten. Thereafter LOAD_PAR is reset.

Parameters INDEX

INDEX	Description	Data type
0	no parameter selected	-
1	Filter time constant for the analog input	REAL
2	Scaling of analog input end of measuring range equals 100%	REAL
3	Scaling of analog input end of measuring range equals 0%	REAL
4	Polyline, interpolation point 1 supply side	REAL

INDEX	Description	Data type
5	Polyline, interpolation point 2 supply side	REAL
6	Polyline, interpolation point 3 supply side	REAL
7	Polyline, interpolation point 4 supply side	REAL
8	Polyline, interpolation point 5 supply side	REAL
9	Polyline, interpolation point 6 supply side	REAL
10	Polyline, interpolation point 7 supply side	REAL
11	Polyline, interpolation point 8 supply side	REAL
12	Polyline, interpolation point 9 supply side	REAL
13	Polyline, interpolation point 10 supply side	REAL
14	Polyline, interpolation point 11 supply side	REAL
15	Polyline, interpolation point 12 supply side	REAL
16	Polyline, interpolation point 13 supply side	REAL
17	Polyline, interpolation point 1 output side	REAL
18	Polyline, interpolation point 2 output side	REAL
19	Polyline, interpolation point 3 output side	REAL
20	Polyline, interpolation point 4 output side	REAL
21	Polyline, interpolation point 5 output side	REAL
22	Polyline, interpolation point 6 output side	REAL
23	Polyline, interpolation point 7 output side	REAL
24	Polyline, interpolation point 8 output side	REAL
25	Polyline, interpolation point 9 output side	REAL
26	Polyline, interpolation point 10 output side	REAL
27	Polyline, interpolation point 11 output side	REAL
28	Polyline, interpolation point 12 output side	REAL
29	Polyline, interpolation point 13 output side	REAL
30	Startup time of the ramp for the reference variable	REAL
31	Safety manipulated value	REAL
32	Offset for setpoint link (ratio/blending controller)	REAL
33	Factor for process value B (three component controller)	REAL
34	Factor for process value C (three component controller)	REAL
35	Offset for process value link (three-component controller)	REAL
36	Factor for disturbance variable link	REAL
37	Operating point	REAL
38	unused	REAL
39	Vertices for split range function Start of input signal A range	REAL
40	Vertices for split range function End of input signal A range	REAL
41	Vertices for split range function Start of output signal A range	REAL
42	Vertices for split range function: End of output signal A range	REAL
43	Vertices for split range function Start of input signal B range	REAL
44	Vertices for split range function End of input signal B range	REAL
45	Vertices for split range function: Start of output signal B range	REAL
46	Vertices for split range function: End of output signal B range	REAL
47	Minimum pulse time	REAL

INDEX	Description	Data type
48	Minimum break time	REAL
49	Selection of the reference variable SP or SP_RE for the controller • -1: Setpoint SP_RE of the instruction • 0 to 3: Analog input value 0 to 3 • 16 to 19: Manipulated variable (LMN) of controllers 0 to 3 • 32 to 35: Control variable A of controllers 0 to 3 • 48 to 51: Control variable B of controllers 0 to 3	INT
50	Selection of the main controlled variable process value A for the controller • -1: Process value A = 0.0 • 0 to 3: Analog input value 0 to 3	INT
51	Selection of the auxiliary controlled variable process value B for the controller • -1: Process value B = 0.0 • 0 to 3: Analog input value 0 to 3	INT
52	Selection of the auxiliary controlled variable process value C for the controller • -1: Process value C = 0.0 • 0 to 3: Analog input value 0 to 3	INT
53	Selection of the auxiliary controlled variable process value D for the controller • -1: Process value D = 0.0 • 0 to 3: Analog input value 0 to 3 • 16 to 19: Manipulated variable (LMN) of controllers 0 to 3	INT
54	Selection of the disturbance variable DISV for the controller • -1: Disturbance variable = 0.0 • 0 to 3: Analog input value 0 to 3	INT
55	Selection on the correction input switch. On the measuring point MP TRACKPER the value can be read off with tool tip. • -1: Position follow-up = 0.0 • 0 to 3: Analog input value 0 to 3	INT
56	Selection on the position feedback input switch for step controllers with position feedback. • -1: Position follow-up = 0.0 • 0 to 3: Analog input value 0 to 3	INT
57	Selecting the signal for transition to safety value for the manipulated value of the controller • -1: Can only be assigned via SAFE_ON of instruction FMT_PID • 0 to 7: Assignment via SAFE_ON parameter of the instruction FMT_PID ORed with digital input 0 to 7	INT
58	Selecting signal for switching over to adjustment function of the manipulated value of the controller. • -1: Can only be assigned via LMNTRKON of instruction FMT_PID • 0 to 7: Assignment via LMNTRKON parameter of the instruction FMT_PID ORed with digital input 0 to 7	INT

INDEX	Description	Data type
59	Selecting the signal for changing over the manipulated value of the controller to LMN_RE • -1: Can only be assigned via LMN_REON of instruction FMT_PID • 0 to 7: Assignment via LMN_REON parameter of the instruction FMT_PID ORed with digital input 0 to 7	INT
60	Selecting the high limit signal of the position feedback • -1: Can only be assigned via LMNRHSRE of instruction FMT_PID • 0 to 7: Assignment via LMNRHSRE parameter of the instruction FMT_PID ORed with digital input 0 to 7	INT
61	Selecting the low limit signal of the position feedback • -1: Can only be assigned via LMNRLSRE of instruction FMT_PID • 0 to 7: Assignment via LMNRLSRE parameter of the instruction FMT_PID ORed with digital input 0 to 7	INT

FMT_DS1

Description FMT_DS1

Instruction FMT_DS1 is available to you for reading the diagnostic data record DS1.

Mode of operation

If parameter READ_DS1 = TRUE, then diagnostic data record DS1 is read from FM 355-2 and written to the structure DS1 of the instance DB. Thereafter the in/out parameter is reset.

This only makes sense, if an error is reported in a channel in DS0.

Call

FMT_DS1 has to be called in the same OB as all the other instructions that access the same FM 355-2.

FMT_DS1 requires no initialization routine.

To always obtain the updated diagnostic values, set the parameter READ_DS1 cyclically to TRUE in OB 82.

After the diagnostic values have been successfully read, FMT_DS1 resets the parameter READ_DS1.

Input parameters FMT_DS1

Table 9-242

Parameters	Address	Data type	Default	Description
MOD_ADDR	0.0	INT	256	The module address that results from the configuration with STEP 7 is set at this input.

Output parameter FMT_DS1

Table 9-243

Parameters	Address	Data type	Default	Description
DS1	2.0	STRUCT		The diagnostic data record DS1 contains bytes from 0 to 12. The first 4 bytes are identical with diagnostic data record DS0. Auto-Hotspot
RET_VALU	16.0	WORD	W#16#0	RET_VALU contains the 2nd and 3rd bytes from parameter STATUS of instruction RDREC.

See also

RDREC: Read data record (Page 6459)

In/out parameters FMT_DS1

Table 9-244

Parameters	Address	Data type	Default	Description
READ_DS1	18.0	BOOL	FALSE	If parameter READ_DS1 = TRUE, then diagnostic data record DS1 is read from FM 355-2 and written to the structure DS1of the instance DB. Thereafter the in/out parameter is reset.

Assignments of DS1

The following table shows the assignments of diagnostic data record DS1. The first 4 bytes are identical to diagnostic data record DS0. Any bits not listed have no meaning and are zero.

Byte	Bit	Meaning	Remark	Event no.
0	0	Module fault	Is set at each diagnostics event	8:x:00
	1	Internal error	Is set at every internal error: • Time monitoring addressed • EPROM error • ADU/DAU error • Analog input hardware error	8:x:01
	2	External error	Is set at every external error: • Missing external auxiliary voltage • Parameter assignment error • Analog input wire break (only in range 4 to 20 mA) • Analog input undershoot of measuring range • Analog input overshoot of measuring range • Analog output load break • Analog output short circuit	8:x:02

Byte	Bit	Meaning	Remark		Event no.
	3	Error in one channel	See from byte 7 for further breakdown		8:x:03
	4	Missing external auxiliary voltage	24 V supply for FM 355-2 failed		8:x:04
	6	Unused	-		-
	7	Parameter assignment error	The module is unable to use a parameter. Reason: Parameters unknown or illegal combination of parameters.		8:x:07
1	0 ... 3	Module class	8 is always assigned.		-
	4	Channel-specific diagnosis	Is set, when the module can provide additional channel information and a channel error exists (see byte 7 to 12)		-
2	3	Time monitoring addressed (watchdog)	Hardware error		8:x:33
3	2	EPROM error	Module defect		8:x:42
	4	ADU/DAU error	Module defect		8:x:44
4	0 ... 7	Channel type	75H is always assigned.		-
5	0 ... 7	Length of the diagnostics information	8 is always assigned.		-
6	0 ... 7	Number of channels	5 always assigned (4 controllers + 1 reference channel)		-
7	0 ... 4	Channel error vector	One bit is assigned to each channel (0...3; 4 for reference channel).		-
8	0	Analog input hardware error	Channel-specific diagnostics channel 0		8:x:B0
	1	Unused			8:x:B1
	2	Analog input wire break (only in range 4 to 20 mA)			8:x:B2
	3	Unused			8:x:B3
	4	Analog input undershoot of measuring range			8:x:B4
	5	Analog input overshoot of measuring range			8:x:B5
	6	Analog output wire break		Only with the current output of the continuous-action controller.	8:x:B6
	7	Analog output short circuit		Only with the voltage output of the continuous-action controller	8:x:B7
9	0 ... 7	See byte 8	Channel-specific diagnostics channel 1		See above
10	0 ... 7	See byte 8	Channel-specific diagnostics channel 2		See above
11	0 ... 7	See byte 8	Channel-specific diagnostics channel 3		See above
12	0 ... 5	See byte 8	Diagnostics for reference channel		See above

FMT_TUN

Description FMT_TUN

With instruction FMT_TUN you obtain additional detailed information (e.g. backup controller parameters) during controller optimization.

Note

However, for performing controller optimization the instruction FMT_PID is adequate. You can start optimization and observe the status information with this instruction.

Call

FMT_TUN has to be called in the same OB as all the other instructions that access the same FM 355-2.

FMT_TUN requires no initialization routine.

Set parameter READ_OUT cyclically to TRUE to receive constantly the updated values.

After the parameters have been successfully read out, FMT_TUN resets the parameter READ_OUT.

Mode of operation

After downloading the controller parameters with LOAD_PAR = TRUE or the operating parameters with LOAD_OP = TRUE the parameters do not take effect immediately. Only after the cycle time of FM355-2 (depending on the number of channels up to 500 ms) the controller parameters are available for reading back with READ_PAR or the output parameters with READ_OUT. This also applies after downloading the system data (CPU transition from STOP to RUN) as well as when reading the output parameters FMT_TUN.

Call at distributed configuration

In the case of disturbed configuration and simultaneous call of FMT_PID and FMT_TUN, the following must be noted.

- LOAD_OP must not be set simultaneously at FMT_PID and FMT_TUN.
- READ_OUT must not be set simultaneously at FMT_PID and FMT_TUN.

Reasons: Both instructions access FM355-2 via the same data records. You must therefore make sure that at the same time only one of the two FBs reads or writes a data record at the same time.

Input parameter FMT_TUN

Table 9-245

Parameters	Address	Data type	Default	Description
MOD_ADDR	0.0	INT	256	The module address that results from the configuration with STEP 7 is set at this input.
CHANNEL	2.0	INT	0	Number of controller channel to which the instance DB refers. Values from 0 to 3 are permitted.

Output parameters FMT_TUN

Table 9-246

Parameters	Address	Data type	Default	Description
PI_GAIN	4.0	REAL	0.0	PI controller parameters
PI_TI	8.0	REAL	0.0	PI controller parameters
PID_GAIN	12.0	REAL	0.0	PID controller parameters
PID_TI	16.0	REAL	0.0	PID controller parameters
PID_TD	20.0	REAL	0.0	PID controller parameters
SAV_PFAC	24.0	REAL	0.0	Saved controller parameters FMT_PID.PAR.PFAC_SP
SAV_GAIN	28.0	REAL	0.0	Saved controller parameters FMT_PID.PAR.GAIN
SAV_TI	32.0	REAL	0.0	Saved controller parameters FMT_PID.PAR.TI
SAV_TD	36.0	REAL	0.0	Saved controller parameters FMT_PID.PAR.TD
SAV_D_F	40.0	REAL	0.0	Saved controller parameters FMT_PID.PAR.D_T
SAV_RATI	44.0	REAL	0.0	Saved controller parameters FMT_PID.PAR.RATIOFAC
SAV_CONZ	48.0	REAL	0.0	Saved controller parameters FMT_PID.PAR.CON_ZONE
SAV_PSEL	52.0	BOOL	FALSE	Saved controller parameters FMT_PID.PAR.P_SEL
SAV_CZON	52.1	BOOL	FALSE	Saved controller parameters FMT_PID.PAR.CONZ_ON
RET_VALU	54.0	WORD	W#16#0	RET_VALU contains the 2nd and 3rd bytes from parameter STATUS of instruction RDREC.

See also

RDREC: Read data record (Page 6459)

In/out parameters FMT_TUN

Table 9-247

Parameters	Address	Data type	Default	Description
READ_OUT	56.0	BOOL	FALSE	If the in/out parameter READ_OUT is set, the output parameters are read from FM 355-2.

Static variables FMT_TUN.OUT

Table 9-248

Parameters	Address	Data type	Default	Description
vers_nr	58.0	WORD	W#16#3230	The vers_nr parameter may not be modified by users. It marks the start of the output parameter that is read by the module if READ_OUT = TRUE is set.
PV	60.0	REAL	0.0	Process value that becomes effective.
LMN	64.0	REAL	0.0	The effective manipulated value that is currently in effect is output at output LMN. For step controller without analog position feedback the unlimited P- + D-action is output at output LMN. Values from -100.0 to 100.0 (%) are permitted.
QLMNSAFE	68.1	BOOL	FALSE	If output QLMNSAFE is set, the safety manipulated value is output as manipulated value.
QLMNTRK	68.3	BOOL	FALSE	Output QLMNTRK indicates if the manipulated value is being compensated via an analog input.
QLMN_RE	68.4	BOOL	FALSE	The output QLMN_RE indicates if LMN_REON is set and the external manipulated value LMN_RE is applied as the manipulated value.
QLMNR_HS	68.5	BOOL	FALSE	QLMNR_HS = TRUE means: Control valve is at the high endstop. (Only in the case of step controllers)
QLMNR_LS	68.6	BOOL	FALSE	QLMNR_LS = TRUE means: The control valve is located at the low endstop. (Only in the case of step controllers)
QLMNR_ON	68.7	BOOL	FALSE	QLMNR_ON = TRUE means: Step controller with position feedback.
QSTEPCON	69.0	BOOL	FALSE	Step controllers/pulse controllers 0 = Pulse controller or continuous controller 1 = step controller
QSPR	69.1	BOOL	FALSE	If output QSPR is set, the continuous controller works in the split-range mode.
QMAN_FC	69.3	BOOL	FALSE	The controller is a master controller that is adjusted by the manual mode of a slave controller to its process value or its I-action is stopped, because the setpoint or manipulated value of the slave controller lies in the limit.
QH_ALM	70.0	BOOL	FALSE	High limit H_ALM exceeded by process value or error signal.
QH_WRN	70.1	BOOL	FALSE	Process value or error signal exceeds second to highest limit H_WRN.
QL_WRN	70.2	BOOL	FALSE	Process value or error signal has dropped below the second to lowest limit L_WRN.

Parameters	Address	Data type	Default	Description
QL_ALM	70.3	BOOL	FALSE	Process value or error signal has dropped below the lowest limit L_ALM.
QLMN_HLM	70.4	BOOL	FALSE	The output QLMN_HLM reports that the high manipulated value limit LMN_HLM. has been reached. (Does not apply to step controllers without analog position feedback).
QLMN_LLM	70.5	BOOL	FALSE	The output QLMN_LLM reports that the high manipulated value limit LMN_LLM has been reached. (Does not apply to step controllers without analog position feedback)
QPAR_F	70.6	BOOL	FALSE	The module checks the parameters for validity. A parameter assignment error is indicated on output QPAR_F.
QCH_F	70.7	BOOL	FALSE	The output QCH_F is set if the controller channel is unable to supply any valid results. A channel error (for example, wire break) is also set if QPAR_F = 1 or QMOD_F = 1. If QCH_F = TRUE, then the precise error information in the diagnostic data record DS1 of the module is read out.
QUPRLM	71.0	BOOL	FALSE	The setpoint is limited to a positive and negative gradient. If the output QUPRLM is set, the setpoint rise is limited.
QDNRLM	71.1	BOOL	FALSE	The setpoint is limited to a positive and negative gradient. If the output QDNRLM is set, the setpoint fall is limited.
QSP_HLM	71.2	BOOL	FALSE	The output QSP_HLM reports that the high setpoint limit SP_HLM has been reached.
QSP_LLM	71.3	BOOL	FALSE	The output QSP_LLM reports that the low setpoint limit SP_LLM has been reached.
QLMNUP	71.4	BOOL	FALSE	"Manipulated value signal up" output. (Only in the case of step controllers or pulse controllers)
QLMNDN	71.5	BOOL	FALSE	"Manipulated value signal down" output. (Only in the case of step controllers or pulse controllers)
QTUN_ON	71.6	BOOL	FALSE	QTUN_ON = TRUE indicates that optimization is in progress.
PAR_ACT	71.7	BOOL	FALSE	Updating controller parameters
SP	72.0	REAL	0.0	Effective actual setpoint.
ER	76.0	REAL	0.0	Negative deviation
DISV	80.0	REAL	0.0	Effective actual disturbance variable. Values from -100.0 to 100.0 (%) are permitted.
LMN_A	84.0	REAL	0.0	In the case of continuous or pulse controllers of manipulated value A of the split-range function and for step controllers with analog position feedback, the position feedback is displayed on output LMN_A. Values from -100.0 to 100.0 (%) are permitted.
LMN_B	88.0	REAL	0.0	On output LMN_B, in the case of continuous or pulse controllers, manipulated value B of the split-range function is displayed. Values from -100.0 to 100.0 (%) are permitted.
PHASE	92.0	INT	0	Indicates the controller optimization phases. *) 0..7
STATUS_H	94.0	INT	0	Diagnostics value for searching for point of inflection during heating optimization See parameter FMT_PID.STATUS_H (Page 7243)

Parameters	Address	Data type	Default	Description
STATUS_C	96.0	INT	0	Diagnostics value for searching for point of inflection during cooling optimization. See parameter FMT_PID.STATUS_C (Page 7245)
STATUS_D	98.0	INT	0	Diagnostics value for controller design during heating optimization See parameter FMT_PID.STATUS_D (Page 7246)
ZONE_TUN	100.0	WORD	W#16#0	In HEX code, each one of the four digits represents one channel, arranged in the following order from left to right: Channel 0, 1, 2, 3. ZONE_TUN = 0 means that the channel selected will not be optimized together with other channels. A value <> 0000 shows - in each case with a 1 - the channels which will be optimized as a group.
GAIN_P	152.0	REAL	0.0	Identified steady state gain (phys. variable) /%. In the case of process type I, GAIN_P tends to be estimated too low.
TU	156.0	REAL	0.0	Identified time lag of process. TU≥ 3*Sampling time
TA	160.0	REAL	0.0	Identified recovery time of process. In the case of process type I, TA tends to be estimated too low.
KIG	164.0	REAL	0.0	Maximum possible rate of rise of the process value. (phys. variable) /s GAIN_P = 0.01 * KIG * TA
N_PTN	168.0	REAL	0.0	Order of the process "Non-integer values" are also possible. Values from 0.1 to 10.0 are permitted.
TM_LAG_P	172.0	REAL	0.0	Time constant of a PTN model. (practical values for N_PTN>=2 only).
T_P_INF	176.0	REAL	0.0	Time from process excitation to inflection point.
P_INF	180.0	REAL	0.0	Change in process value from process excitation to inflection point.
LMN0	184.0	REAL	0.0	Manipulated value at the start of optimization Is established in phase 1 (average) Values from -100.0 to 100.0 (%) are permitted.
PV0	188.0	REAL	0.0	Process value at the start of optimization Is established in phase 1 (average).
PVDT0	192.0	REAL	0.0	Process value slew rate at the start of optimization Sign is adapted. (phys. variable) /s
PVDT	196.0	REAL	0.0	Current process value slew rate Sign is adapted. (phys. variable) /s
PVDT_MAX	200.0	REAL	0.0	Maximal process value modification (phys. variable) /s Maximum derivative of the process value at the point of inflection (sign adapted, always > 0); is used to calculate TU and KIG.
NOI_PVDT	204.0	REAL	0.0	Noise action in PVDT_MAX in % The larger the noise action, the less precise (softer) are the controller parameters.
NOISE_PV	208.0	REAL	0.0	Absolute process value noise action (phys. variable) /s The difference between maximum and minimum process value in phase 1.
FIL_CYC	212.0	INT	0	Number of cycles of the mean value filter The process value is determined through FIL_CYC cycles. FIL_CYC is increased from 1 to a max. of 1024 if needed.

Parameters	Address	Data type	Default	Description
POI_CMAX	214.0	INT	0	Maximum number of cycles after inflection point This time is used to find another (i.e. better) inflection point in case of measuring noise. Only then is optimization completed.
POI_CYCL	216.0	INT	0	Number of cycles after inflection point

See also

Parameters STATUS_H (Page 7243)

Parameters STATUS_C (Page 7245)

Parameters STATUS_D (Page 7246)

FMT_PV

Description FMT_PV

Instruction FMT_PV is for reading or writing process values (analog and digital input values) to support startup.

Call

FMT_PV has to be called in the same OB as all the other instructions that access the same FM 355-2.

FMT_PV requires no initialization routine.

Set parameter LOAD_PV cyclically to TRUE if you wish to write cyclically simulated process values to FM 355-2.

Set parameter READ_PV cyclically to TRUE to receive constantly the updated process values.

After the diagnostic values have been successfully written or read, FMT_PV resets the parameter LOAD_PV or READ_PV .

Simulation of the analog values (LOAD_PV = TRUE)

The simulation of the analog value for the channels 0 to 3 is enabled via the switches S_AION[i] or S_PVON[i], where 0 ≤ i ≤ 3. The figure below shows the point at which the simulated analog value becomes effective.

You specify the simulation values for the channels 0 to 3 via the parameter PV_SIM[i].

You can have the simulation values become effective at two points:

- S_AION[i] = TRUE (0 ≤ i ≤ 3)
  The value PV_SIM[i] is used instead of the value of analog input i of the module.
- S_PVON[i] = TRUE (0 ≤ i ≤ 3)
  The value PV_SIM[i] is used instead of the conditioned value of analog input i of the module.

Simulation of the digital values (LOAD_PV = TRUE)

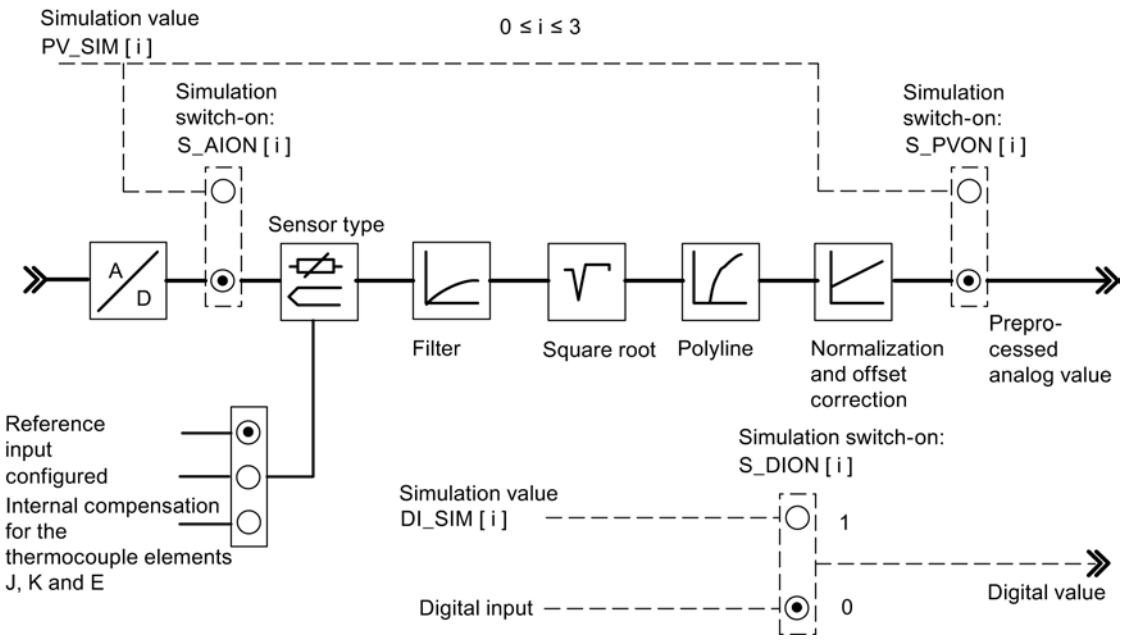
The simulation of the values for the digital inputs 0 to 7 is enabled via the switches S_DION[i], where 0 ≤ i ≤ 7 is.

You lay down the simulation values via parameter DI_SIM[i].

- S_DION[i] = TRUE (0 ≤ i ≤ 7)
  Instead of the value of digital input i of the module, the value DI_SIM[i] is used.

Note

LEDs I0 to I7 always display the state of the associated digital input, also during simulation.



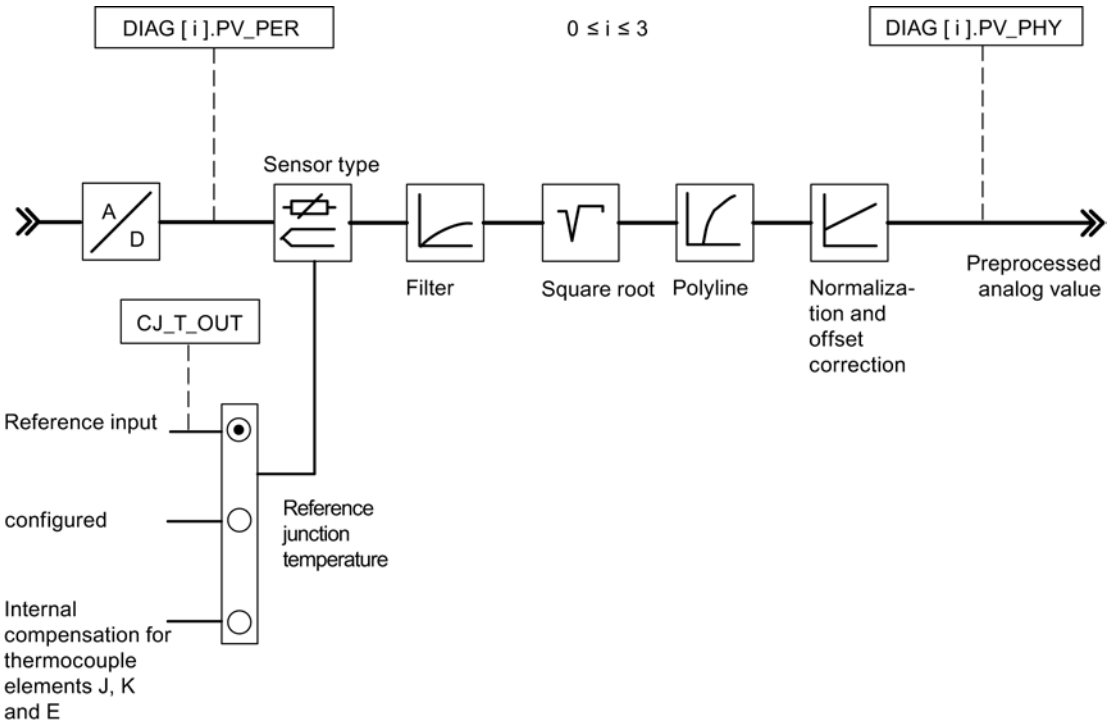
Note

Enabling and specification of the simulation values (Force) does not take place via the configuration software. This is why the relevant switches and connecting lines are drawn as dotted lines.

Displaying the process values (READ_PV = TRUE)

The following values are displayed:

- The actual state of digital inputs 0 to 7 are displayed at parameters STAT_DI[0] to STAT_DI[7], even if these are simulated.
- The value of analog inputs 0 to 3 is displayed on parameters DIAG[0].PV_PER to DIAG[3].PV_PER in mA or mV units. If the simulation of the analog input value was enabled, the simulated value is displayed.
- The conditioned analog input value 0 to 3 is displayed on parameters DIAG[0].PV_PHY to DIAG[3].PV_PHY in physical unit. If the simulation of the conditioned physical analog input value was enabled, the simulated value is displayed.



Note

The updating of the process values can be delayed if FM 355-2 is loaded compared to instruction FMT_PID.

Start-up

When FM 355-2 restarts after power off, the simulation switches on the FM 355-2 are again set to FALSE.

Input parameter FMT_PV

Table 9-249

Parameters	Address	Data type	Default	Description
S_AION	0.0	ARRAY [0..3] of BOOL	FALSE	If, for example, the S_AION[1] switch is set to TRUE, the value PV_SIM[1] is used instead of the analog input value 1 of the module.
S_PVON	2.0	ARRAY [0..3] of BOOL	FALSE	If, for example, the S_PVON[1] switch is set to TRUE, the value PV_SIM[1] is used instead of the conditioned analog input value 1 of the module.
PV_SIM	4.0	ARRAY [0..3] of REAL	0.0	The simulation value for the analog input 1 is specified, for example, at input PV_SIM[1]. If S_PVON = TRUE, then the conditioned analog input value is specified here. If S_PVON = FALSE and S_AION = TRUE, then the analog input value, which is transformed into a conditioned value by means of the conditioning functions, is specified in mA or mV. 0.0 to 20.0 [mA] or -1500 to +10000 [mV] or technical value range
S_DION	20.0	ARRAY [0..7] of BOOL	FALSE	If, for example, the S_DION[1] switch is set to TRUE, the value DI_SIM[1] is used instead of the digital input value 1 of the module.
DI_SIM	22.0	ARRAY [0..7] of BOOL	FALSE	
MOD_ADDR	24.0	INT	256	The module address that results from the configuration with STEP 7 is set at this input.

See also

Output parameter FMT_PV (Page 7264)

in/out parameters FMT_PV (Page 7264)

Output parameter FMT_PV

Table 9-250

Parameters	Address	Data type	Default	Description
STAT_DI	26.0	ARRAY [0..7] of BOOL	FALSE	The states of digital inputs 0 to 7 are displayed at the STAT_DI parameters.
DIAG[x].PV_PER	28+ (Channel number x8)	ARRAY [0..3] of STRUCT	0.0	The analog input value of the module is displayed, for example, in the units mA or mV, at the DIAG[x].PV_PER parameter.
DIAG[x].PV_PHY	32+ (Channel number x8)	ARRAY [0..3] of STRUCT	0.0	The DIAG[x].PV_PHY parameter, for example, displays the conditioned analog input value of the module in the physical unit.
RET_VALU	60.0	WORD	W#16#0	RET_VALU contains the 2nd and 3rd bytes from the STATUS parameter of the instructions RDREC and WRREC.

See also

RDREC: Read data record (Page 6459)

WRREC: Write data record (Page 6461)

in/out parameters FMT_PV

Table 9-251

Parameters	Address	Data type	Default	Description
LOAD_PV	62.0	BOOL	FALSE	When the in/out parameter LOAD_PV is set, the simulation values in PV_SIM and DI_SIM are written to FM 355-2 in accordance with the switch positions and the in/out parameter is reset.
READ_PV	62.1	BOOL	FALSE	If the parameter READ_PV = TRUE, the reference junction temperature is read from the module. Thereafter the in/out parameter is reset.

See also

Input parameter FMT_PV (Page 7263)

Output parameter FMT_PV (Page 7264)

FMT_CJ_T

Description FMT_CJ_T

Instruction FMT_CJ_T is for reading the measured reference junction temperature and for changing the parameterized reference junction temperature online. This is necessary if a temperature control system with several FM 355-2 with thermoelement inputs is to be operated without having to connect a Pt 100 to each FM 355-2.

Call

FMT_PAR has to be called in the same OB as all the other instructions that access the same FM 355-2.

If FM 355-2 is used in distributed I/O, it may take a few call cycles until the parameters are applied to FM 355-2. As long as the transfer has not been completed, parameter LOAD_CJ retains the value TRUE. So keep repeating your call of FMT_CJ_T when modifying parameters until the block sets LOAD_CJ = FALSE.

Mode of operation

If, for example, an FM 355-2 measures the reference junction temperature of an extruder control system with more than four heating zones, it can be read with READ_CJ = TRUE on parameter CJ_T_OUT and can be parameterized for the other FM 355-2 via CJ_TEMP and LOAD_CJ.

On parameter CJ_T_OUT the reference junction temperature measured at the reference point is expressed in degrees C or in degrees F (depending on which temperature unit has been parameterized).

On parameter CJ_T_OUT 0.0. is displayed if:

- no sensor of the "Thermoelement" type has been parameterized,
- the parameterized reference junction temperature was selected on all analog inputs,
- the internal compensation was selected on all analog inputs.

Input parameters FMT_CJ_T

Table 9-252

Parameters	Address	Data type	Default	Description
MOD_ADDR	0.0	INT	256	The module address that results from the configuration with STEP 7 is set at this input.
CJ_TEMP	2.0	REAL	0.0	The reference junction temperature can be specified at parameter CJ_TEMP.

Output parameters FMT_CJ_T

Table 9-253

Parameters	Address	Data type	Default	Description
CJ_T_OUT	6.0	REAL	0.0	At the CJ_T_OUT output, the measured reference junction temperature is displayed by the module if a thermocouple element input is configured and the reference junction temperature is not configured.
RET_VALU	10.0	WORD	W#16#0	RET_VALU contains the 2nd and 3rd bytes from the STATUS parameter of the instructions RDREC and WRREC.

See also

RDREC: Read data record (Page 6459)

WRREC: Write data record (Page 6461)

In/out parameters FMT_CJ_T

Table 9-254

Parameters	Address	Data type	Default	Description
LOAD_CJ	12.0	BOOL	FALSE	If the in/out parameter LOAD_CJ has been set, the configured reference junction temperature on the module is overwritten with the value CJ_TEMP and the in/out parameter is reset.
READ_CJ	12.1	BOOL	FALSE	If the parameter READ_CJ = TRUE, the reference junction temperature is read from the module. Thereafter the in/out parameter is reset.

FM 455 PID Control

PID_FM_455

Description PID_FM_455

During operation, the instruction PID_FM_455 is used to:

- Transfer operating parameters to FM 455
- Transfer controller parameters to FM 455
- Read process values from FM 455

Operating parameters are all in/out parameters that lie between the op_par and cont_par parameters in the instance data block of the instruction.

Controller parameters are all in/out parameters that are located after the cont_par parameter in the instance data block of the instruction.

Process values are all output parameters of the instruction after the out_par parameter.

The required data are stored in an instance DB on the CPU. You need one instance DB for each controller channel used. All the in/out parameters are set to FALSE after an instance DB has been created.

Call

PID_FM_455 is normally called in the cyclic interrupt OB 35 and requires an initialization routine that is started by setting the parameter COM_RST = TRUE in the start-up of the CPU. A calling of the instruction in the start-up OB is possible, but not necessary. After the initialization routine, PID_FM_455 sets the parameter COM_RST to FALSE.

PID_FM_455 has to be called in the same OB as all the other instructions that access the same FM 455.

If you call the instruction PID_FM_455 as a multiple instance DB, no commissioning interface is available. In this case you must commission the FM 455 via a watch table.

Start-up

During CPU start-up and during CPU STOP-RUN transition, the parameters on the FM 455 are overwritten with the parameters from the SDB of the CPU.

Responses in the event of an error

The output parameter RET_VALU includes the return value RET_VAL of the RD_REC und WR_REC instructions. RET_VALU can be evaluated if theREAD_VAR and LOAD_PAR parameters are not reset.

When the instruction PID_FM_455 is called, an I/O access error can occur if the FM 455 is not plugged or is not supplied with voltage. In this case the CPU goes into STOP, if no OB 122 is loaded in the CPU.

Mode of operation PID_FM_455

Effect of parameters LOAD_OP, READ_VAR and LOAD_PAR

In the initialization routine PID_FM_455 reads the parameters from the system data and stores them in its instance DB. Instance DB and system data are synchronized.

The following table shows how the function of the instruction is influenced by the parameters LOAD_OP, READ_VAR and LOAD_PAR.

Parameters	TRUE	FALSE
LOAD_OP	Using the instruction WR_REC to transfer the operating parameters from the CPU to FM 455	Using direct I/O access in multiplex mode to transfer the operating parameters from the CPU to FM 455
	Longer run time	Shorter run time
	The values are effective after a cycle of the CPU or FM. The longer cycle is decisive.	A maximum of 3 cycles of the CPU or FM are required until the values are effective. The longer cycle is decisive

Parameters	TRUE	FALSE
READ_VAR	Using the instruction RD_REC to read the process value from FM 455	Using direct I/O access in multiplex mode to read the process values from FM 455
	Longer processing time	Shorter processing time
	All parameters are read from FM 455	The following parameters are **not** read from FM 455 • SP (setpoint from the FM) • ER (error signal) • DISV (disturbance variable) • LMN_A • LMN_B
	Following parameters are updated at **every** call of the instruction. • PV (process value) • LMN (manipulated value) • The binary displays	Following parameters are updated at every **fourth** call of the instruction. • PV (process value) • LMN (manipulated value) • The binary displays
	At the start-up of the CPU, the following values are applied from the FM 455 to the PID_FM_455. • SP_OP • LMN_OP • SP_OP_ON • LMNOP_ON If these values were changed on the CPU, they are overwritten with the values from the FM 455 at the next start-up of the CPU	At the start-up of the CPU, the following values are read from the instruction, not from the FM: • SP_OP • LMN_OP • SP_OP_ON • LMNOP_ON
LOAD_PAR	All controller parameters are downloaded from the CPU to FM 455. This is practical if the controller parameters are to be changed depending on process states.	No transfer of the controller parameters

After data transfer as been completed the parameters LOAD_OP, READ_VAR and LOAD_PAR are reset to FALSE.

Special features during transfer in multiplex mode

Multiplexing of the data to be transferred during access to the FM 455 via direct I/O accesses is controlled via the PID_FM_455. This multiplex controlling via PID_FM_455 does not function if two instances of the instruction PID_FM_455 access the same channel number of a module. This results in incorrect parameters in the FM 455 (for example, setpoint and manual manipulated value) and incorrect displays at the output parameters of the instruction PID_FM_455.

Saving the Parameters in EEPROM

During program controlled re-configuring (LOAD_PAR, LOAD_OP) of the controller module by the instruction PID_FM_455, its run time is extended. The new parameters are effective immediately and are stored additionally in a nonvolatile memory (EEPROM). After each saving of the parameters in the EEPROM, a renewed saving is delayed by 30 minutes, since the life of the EEPROM is limited by the number of write processes. It is possible to save the new parameter in the EEPROM as soon as the power supply returns. The selection of the

parameters determines whether the reconfiguration of the controller module by the instruction PID_FM_455 is executed without bumps.

Processing times of the PID_FM_455

Boundary conditions			Processing time in CPU 414-2DP
READ_VAR	LOAD_OP	LOAD_PAR	
FALSE	FALSE	FALSE	0.077 ms
TRUE	FALSE	FALSE	2.36 ms
TRUE*)	TRUE	FALSE	4.48 ms
FALSE	FALSE	TRUE	2.59 ms
TRUE	FALSE	TRUE	5.15 ms
TRUE*)	TRUE	TRUE	7.1 ms
*) If LOAD_OP = TRUE, READ_VAR is also set to TRUE by the PID_FM_455.			

Input parameters PID_FM_455

Table 9-255

Parameters	Address	Data type	Default	Description
MOD_ADDR	0.0	INT	256	The module address that results from the configuration with STEP 7 is set at this input.
CHANNEL	2.0	INT	1	The number of the controller channel to which the instance DB is referenced is configured at the "Channel number" input. Values of 1 to 16 are permitted.
PHASE	4.0	INT	0	The parameter PHASE can be interconnected with the output parameter PHASE of the instructions TUN_EC or TUN_ES (PID self-tuner). The phase state of the PID Self-Tuner can then be displayed in plain text in the "Commissioning" dialog. This parameter is of no significance for the OP.

Output parameters PID_FM_455

Parameters	Address	Data type	Default	Description
RET_VALU	6.0	WORD	0	RET_VALU contains the return value RET_VAL of the WR_REC and RD_REC instructions. RET_VALU can be evaluated when QMOD_F reports an error.
out_par	8.0	WORD	W#16#3130	The out_par parameter may not be modified by users. It marks the start of the output parameter that is read by the module if READ_VAR = TRUE is set.
SP	10.0	REAL	0.0	The setpoint that is currently in effect is available at the "Setpoint" output.
PV	14.0	REAL	0.0	The process value that becomes effective is issued on the "Process value" output.
ER	18.0	REAL	0.0	The control deviation that becomes effective is issued on the "Control deviation" output.
DISV	22.0	REAL	0.0	The interference that becomes effective is issued on the "Interference" output. Values from -100 to 100 % are permitted.

Parameters	Address	Data type	Default	Description
LMN	26.0	REAL	0.0	The manipulated value that becomes effective is issued on the "Manipulated value" output. Step controllers without analog position feedback output the unlimited P- + D-action at the LMN parameter. Values from -100 to 100 % are permitted.
LMN_A	30.0	REAL	0.0	At the output "Manipulated value A of the split range function / position feedback", continuous controllers display the manipulated value A and step controllers with analog position feedback display the position feedback. The output LMN_A can only be used for an approximate display of a corresponding simulated manipulated variable. The start value LMNRSVAL of the simulated position feedback has to be configured accordingly and becomes effective when LMNRS_ON is set. Values from -100 to 100 % are permitted.
LMN_B	34.0	REAL	0.0	Continuous controllers display the manipulated value B of the split-range function at the output “Manipulated value B of split-range function”. Values from -100 to 100 % are permitted.
QH_ALM	38.0	BOOL	FALSE	The process value or the controlled variable is monitored for four limits. Violation of H_ALM limit is reported at the output "High limit alarm triggered".
QH_WRN	38.1	BOOL	FALSE	The process value or the controlled variable is monitored for four limits. Violation of H_WRN limit is reported at the output "High limit warning triggered".
QL_WRN	38.2	BOOL	FALSE	The process value or the controlled variable is monitored for four limits. Violation of the L_WRN limit is reported at output "Low limit warning triggered".
QL_ALM	38.3	BOOL	FALSE	The process value or the controlled variable is monitored for four limits. Violation of the L_ALM limit is reported at output "Low limit alarm triggered".
QLMN_HLM	38.4	BOOL	FALSE	The manipulated value is always limited to a high and a low limit. Violation of the high limit is reported at output "High limit of manipulated value triggered". (Does not apply to step controllers without analog position feedback)
QLMN_LLM	38.5	BOOL	FALSE	The manipulated value is always limited to a high and a low limit. Violation of the low limit is reported at output "Low limit of manipulated value triggered". (Does not apply to step controllers without analog position feedback)
QPARA_F	38.6	BOOL	FALSE	The module checks the validity of the parameters. A parameter assignment error is displayed at the "Parameter assignment error" output. These parameter assignment errors can also be read out in the module's "Online & Diagnostics" dialog.
QCH_F	38.7	BOOL	FALSE	The "Channel error" output is set if the controller channel cannot provide valid results. A channel error (for example, wire break) is also set if QPARA_F = 1 or QMOD_F = 1. If QCH_F = TRUE, then the precise error information in the diagnostic data record DS1 of the module is read out.
QUPRLM	39.0	BOOL	FALSE	The setpoint is limited to a positive and negative slew rate. If the output "Limit of positive setpoint slew rate triggered" is set, the positive setpoint slew rate is limited.
QDNRLM	39.1	BOOL	FALSE	The setpoint is limited to a positive and negative slew rate. If the output "Limit of negative setpoint slew rate triggered" is set, the setpoint fall is limited.
QSP_HLM	39.2	BOOL	FALSE	The setpoint is always limited to a high and a low limit. The output "High limit of setpoint value triggered" indicates that the high limit has been exceeded.
QSP_LLM	39.3	BOOL	FALSE	The setpoint is always limited to a high and a low limit. Violation of the low limit is reported at output "Setpoint low limit triggered".
QLMNUP	39.4	BOOL	FALSE	This is the "Manipulated value signal up" output. (Only in the case of step controllers or pulse controllers)

Parameters	Address	Data type	Default	Description
QLMNDN	39.5	BOOL	FALSE	This is the "Manipulated value signal down" output. (Only in the case of step controllers or pulse controllers)
QID	39.6	BOOL	FALSE	QID = TRUE indicates that an identification is running (not that it is enabled). On completion of the identification, the identification result can be read out from the IDSTATUS parameter of the instruction CH_DIAG_455.
QSPOPON	40.0	BOOL	FALSE	The output "Setpoint operation on" shows if the setpoint is being operated by the configuration tool. If the bit is set, the value SP_OP is applied as setpoint.
QLMNSAFE	40.1	BOOL	FALSE	If the output "Safety mode" is set, the safety manipulated value is output as the manipulated value.
QLMNOPON	40.2	BOOL	FALSE	The output “Manipulated value operation on” indicates whether the manipulated value is being operated via the configuration tool. If the bit is set, the value LMN_OP is applied as manipulated value.
QLMNTRK	40.3	BOOL	FALSE	The output "Follow-up mode" indicates if the manipulated value is being adjusted via an analog input.
QLMN_RE	40.4	BOOL	FALSE	The output "Manual = 1; Automatic = 0" indicates whether or not the manipulated value is set on the external manipulated value LMN_RE (manual = 1).
QLMNR_HS	40.5	BOOL	FALSE	The output "High endstop signal of position feedback" indicates if the control valve is at its high endstop. QLMNR_HS = TRUE means: The control valve is at high endstop. (For step controllers only)
QLMNR_LS	40.6	BOOL	FALSE	The output "Low endstop signal of position feedback" indicates whether the control valve is at its low endstop. QLMNR_LS = TRUE means: The control valve is at its low endstop. (For step controllers only)
QLMNR_ON	40.7	BOOL	FALSE	Output "Position feedback enabled" indicates the set operating mode "Step controller with position feedback" or "Step controller without position feedback".
QFUZZY	41.0	BOOL	FALSE	If the output parameter is QFUZZY = 1, the controller operates with the fuzzy algorithm.
QSPLEPV	41.1	BOOL	FALSE	The output "Display of FUZZY controller: Setpoint < process value" is set when the fuzzy controller is switched on, if the setpoint is less than the effective process value.
QSPR	41.2	BOOL	FALSE	If the output "Split-range operation" is set, the continuous controller is operating in split-range mode.
QMAN_FC	41.4	BOOL	FALSE	The output "QMAN_FC" is set in the following two cases: • The slave controller is in manual mode and the master controller is followed up to the process value of the slave controller. • The I-action of the master controller is stopped because the setpoint or manipulated variable of the slave controller is limited or because the slave controller is in manual mode.

Parameters	Address	Data type	Default	Description
QPARABUB	41.7	BOOL	FALSE	This parameter is set by the FM if the operating parameters are changed via the OP. If READ_VAR = TRUE and if this display is set by the FM, the instruction PID_FM_455 reads out the parameters SP_OP_ON, LMNOP_ON, SP_OP and LMN_OP from the FM and saves them in the instance DB. The instruction thus applies over the operating state of the FM. After the reading process the parameter is set to FALSE.
QMOD_F	42.0	BOOL	FALSE	The instruction checks that the reading and writing of a data record has been completed. Output "Module error" is set if errors are found. The error cause can be: An incorrect module address at parameter MOD_ADDR, or an incorrect channel number at parameter CHANNEL, or a faulty module.

See also

RD_REC: Read data record from I/O (Page 6491)

WR_REC: Write data record to I/O (Page 6496)

In/out parameters PID_FM_455

Table 9-256

Parameters	Address	Data type	Default	Description
[1] Default values of the module after the first start-up of the PID_FM_455 with COM_RST = TRUE				
COM_RST	44.0	BOOL	FALSE	If the parameter COM_RST = TRUE, the instruction PID_FM_455 executes an initialization routine. In doing so the controller parameters (these are all the parameters after cont_par) are read from the FM and stored in the instance DB. In addition, the parameters MOD_ADDR and CHANNEL are checked for validity. After the initialization routine the parameter is set to FALSE.
LOAD_OP	44.1	BOOL	FALSE	If the in/out parameter "Load operating parameters to FM 455" is set, the operating parameters are loaded into the module and the in/out parameter is reset.
READ_VAR	44.2	BOOL	FALSE	If the in/out parameter "Read tags from FM 455" is set, the output parameters are read from the module and the in/out parameter is reset.
LOAD_PAR	44.3	BOOL	FALSE	If the in/out parameter "Load control parameters to FM 455" is set, the control parameters are loaded into the module and the in/out parameter is reset.
op_par	46.0	WORD	W#16#3130 [1]	The op_par parameter may not be modified by users. It identifies the start of the operating parameters that are transferred to the module, if LOAD_OP = TRUE is set. The end of the operating parameters is identified by cont_par.
SP_RE	48.0	REAL	0.0	An external setpoint is interconnected with the controller at the "External setpoint" input.
LMN_RE	52.0	REAL	0.0	An external manipulated value is interconnected to the controller at the input "External manipulated value". Values from -100 to 100 % are permitted.
SP_OP_ON	56.0	BOOL	FALSE	The instructions of PID Self-Tuner have access to the in/out parameter "Enable setpoint operation". If the bit is set, the value SP_OP is applied as setpoint.

Parameters	Address	Data type	Default	Description
SAFE_ON	56.1	BOOL	FALSE	If the "Adopt safety position" input is set, a safety value is applied as manipulated value. Note: The actuating signal processing via LMNDN_OP, LMNUP_OP and LMNSOPON on a step controller is of higher priority than the safety manipulated value.
LMNOP_ON	56.2	BOOL	FALSE	The instructions of PID Self-Tuner have access to the in/out parameter "Enable manipulated value operation". If the bit is set, the value LMN_OP is applied as manipulated value.
LMNTRKON	56.3	BOOL	FALSE	If input "Follow-up (LMN via AI)" is set, the manipulated value is adjusted to an analog input (AI). (Does not apply to step controllers without analog position feedback)
LMN_REON	56.4	BOOL	FALSE	If input "Enable external manipulated value" is set, the external manipulated value LMN_RE is set as the manipulated value.
LMNRHSRE	56.5	BOOL	FALSE	The signal "Control valve at high endstop" is interconnected at the "High endstop signal of position feedback" input. LMNRHSRE = TRUE means: The control valve is located at the high endstop. (Only in the case of step controllers)
LMNRLSRE	56.6	BOOL	FALSE	The signal "Control valve at low endstop" is interconnected at the "Low endstop signal of position feedback" input. LMNRLSRE = TRUE means: The control valve is located at the low endstop. (Only in the case of step controllers)
LMNSOPON	56.7	BOOL	FALSE	The instructions of PID Self-Tuner have access to the input "Enable manipulated value operation". If LMNSOPON is set, the signals LMNUP_OP and LMNDN_OP are applied as manipulated value signals. (Only in the case of step controllers)
LMNUP_OP	57.0	BOOL	FALSE	The instructions of PID Self-Tuner have access to the input "Manipulated value signal up". If LMNSOPON is set, the value at input "Manipulated value signal up operation" is set as the manipulated value signal. (Only in the case of step controllers)
LMNDN_OP	57.1	BOOL	FALSE	The instructions of PID Self-Tuner have access to the input "Manipulated value signal down operation". If LMNSOPON is set, the value at input "Manipulated value signal down operation" is set as the manipulated value signal. (Only in the case of step controllers)
LMNRS_ON	57.3	BOOL	FALSE	You can simulate no position feedback if this is not available. The function is enabled by setting the "Enable simulation of position feedback" input The instructions of PID Self-Tuner have access to this parameter as well since at least one simulated manipulated value is required for optimization if a step controller was configured without position feedback. The simulated value is displayed at parameter LMN_A. When the simulation is enabled the value of the parameter LMNRSVAL is set as the starting value. CAUTION: Offset between the simulation value and real position feedback increases over the time. (Only for step controllers without analog position feedback)
FUZID_ON	57.4	BOOL	FALSE	The identification of the fuzzy algorithm is enabled at the "Enable fuzzy identification" input.
SP_OP	58.0	REAL	0.0	The instructions of PID Self-Tuner have access to the in/out parameter "Setpoint operation". If the SP_OP_ON bit is set, the "Setpoint operation" value is applied as the setpoint.

Parameters	Address	Data type	Default	Description
LMN_OP[1]	62.0	REAL	0.0	The instructions of PID Self-Tuner have access to the in/out parameter "Manipulated value operation". If the LMNOP_ON bit is set, the "Manipulated value operation" value is applied as the manipulated value. Values from -100 to 100 % are permitted.
LMNRSVAL	66.0	REAL	0.0	The instructions of PID Self-Tuner have access to input "Starting value of simulated position feedback". The start value of the simulation is entered at the parameter. (Only for step controllers without analog position feedback) Values from -100 to 100 % are permitted.
cont_par	70.0	WORD	W#16#3130[1]	The cont_par parameter may not be modified by users. It indicates the start of the controller parameters which are read from the FM and stored in the instance DB, if COM_RST = TRUE is, and which are transferred to the FM when LOAD_PAR = TRUE is set. The end of the controller parameter is the end of the instance DB.
P_SEL	72.0	BOOL	TRUE[1]	In the PID algorithm, the PID parts can be switched on and off individually. The proportional action is enabled when the “P-action on” input is set.
PFDB_SEL	72.1	BOOL	FALSE[1]	In the PID algorithm, the P- and D-actions can be included in the feedback loop. The proportional action is located in the feedback loop when the input "Enable P-action in feedback loop" is set.
MONERSEL	72.2	BOOL	FALSE[1]	The controller has a limit indicator that can be applied either for the process value or the control deviation. If the input "Monitoring: Process value = 0, control deviation = 1" is set, the control deviation will be monitored.
SDB_SEL	72.4	BOOL	FALSE	SDB_SEL=TRUE: After CPU STOP-RUN, the controller parameters of instruction PID_FM_455 are not overwritten by the SDB parameters in the FMx55.
D_EL_SEL	74.0	INT	0[1]	The D-action element in the PID algorithm can be positioned at a separate input. This is selected via the input “D-action element input for the controller”. • 0: Control deviation • 1 to 16: Analog input 1 to 16 • 17: Negative process value, D-action in the feedback
SP_HLM	76.0	REAL	100.0[1]	The setpoint is always limited to a high and a low limit. The "Setpoint high limit" input specifies the high limit. SP_HLM > SP_LLM
SP_LLM	80.0	REAL	0.0[1]	The setpoint is always limited to a high and a low limit. The "Setpoint low limit" input specifies the low limit. SP_LLM < SP_HLM
H_ALM	84.0	REAL	100.0[1]	You can assign parameters for four limits for monitoring the process value or the control deviation. The "High limit interrupt" input specifies the highest limit. H_ALM > H_WRN
H_WRN	88.0	REAL	90.0[1]	You can assign parameters for four limits for monitoring the process value or the control deviation. The "High limit warning" input specifies the second highest limit. Values from H_ALM to L_WRN are permitted.
L_WRN	92.0	REAL	10.0[1]	You can assign parameters for four limits for monitoring the process value or the control deviation. The "Low limit warning" input specifies the second lowest limit. Values from H_WRN to L_ALM are permitted.
L_ALM	96.0	REAL	0.0[1]	You can assign parameters for four limits for monitoring the process value or the control deviation. The "Low limit alarm" input specifies the lowest limit. L_ALM < L_WRN

Parameters	Address	Data type	Default	Description
HYS	100.0	REAL	1.0[1)]	To prevent flickering of the monitoring displays, a hysteresis can be configured at the "Hysteresis" input. HYS ≥ 0.0
DEADB_W	104.0	REAL	0.0[1)]	A deadband is applied to the control deviation. The "Deadband width" input determines the size of the deadband. DEADB_W ≥ 0.0
GAIN	108.0	REAL	1.0[1)]	The "Proportional gain" input specifies controller amplification.
TI	112.0	REAL	3000.0[1)]	The "Integration time" input determines the time response of the integral action. If TI = 0, the integral action is disabled. TI = 0.0 or TI ≥ 0.5
TD	116.0	REAL	0.0[1)]	The "Derivative-action time" input determines the time response of the differentiator. If TD = 0, the derivative action is disabled. TD = 0.0 or TD ≥ 1.0
TM_LAG	120.0	REAL	5.0[1)]	The algorithm of the derivative action contains a delay for which parameters can be assigned at the input "Time lag of the derivative action". TM_LAG ≥ 0.5
LMN_SAFE	124.0	REAL	0.0[1)]	For the manipulated value, a safety value can be configured at the "Safety manipulated value" input. Values from -100 to 100 % are permitted.
LMN_HLM	128.0	REAL	100.0[1)]	The manipulated value is always limited to a high and a low limit. The "Manipulated value high limit" input specifies the high limit. (Does not apply to step controllers without analog position feedback) Values from LMN_LLM to 100 % are permitted.
LMN_LLM	132.0	REAL	0.0[1)]	The manipulated value is always limited to a high and a low limit. The "Manipulated value low limit" input specifies the low limit. (Does not apply to step controllers without analog position feedback) Values from -100 to LMN_HLM are permitted.
MTR_TM	136.0	REAL	60.0[1)]	The runtime from endstop to endstop of the control valve is entered at the "Motor actuating time" parameter. (Only in the case of step controllers)
PULSE_TM	140.0	REAL	0.2[1)]	A minimum pulse time can be configured at the "Minimum pulse time" parameter. (Applies to step controllers or pulse controllers only)
BREAK_TM	144.0	REAL	0.2[1)]	A minimum pulse break duration can be assigned with the parameter "Minimum break time." (Applies to step controllers or pulse controllers only)

Note

If LOAD_PAR = TRUE is set, all the control parameters are loaded permanently to the EEPROM of FM 455.

If LOAD_OP = TRUE is set, only the setpoint SP_RE of the operating parameters is loaded permanently to the EEPROM of the FM 455. All the other operating parameters are assigned the values 0 or FALSE as default during the startup of FM 455

The EEPROM of the module could become damaged by too frequent write operations. To prevent this, the module delays another write process by 30 minutes after writing of the EEPROM.

Block diagram PID_FM_455

Overview

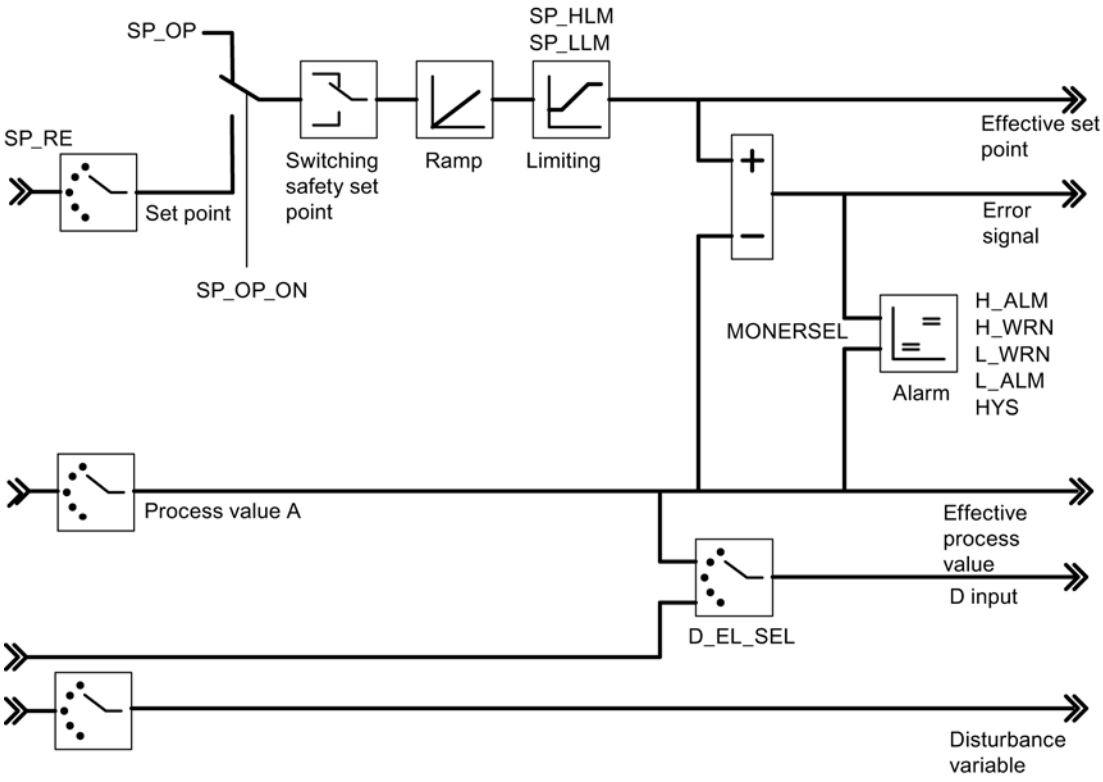
The following figures show at which points the parameters of the instruction PID_FM_455 act.

The parameters are effective with three-component controllers, ration/blending controllers, and fixed setpoint or cascade controllers. This also applies for the parameters that exist equally at continuous-action controllers, at controllers with a pulse output as well as at step controllers. Therefore, not all block diagrams are depicted and not all parameter are marked in all figures.

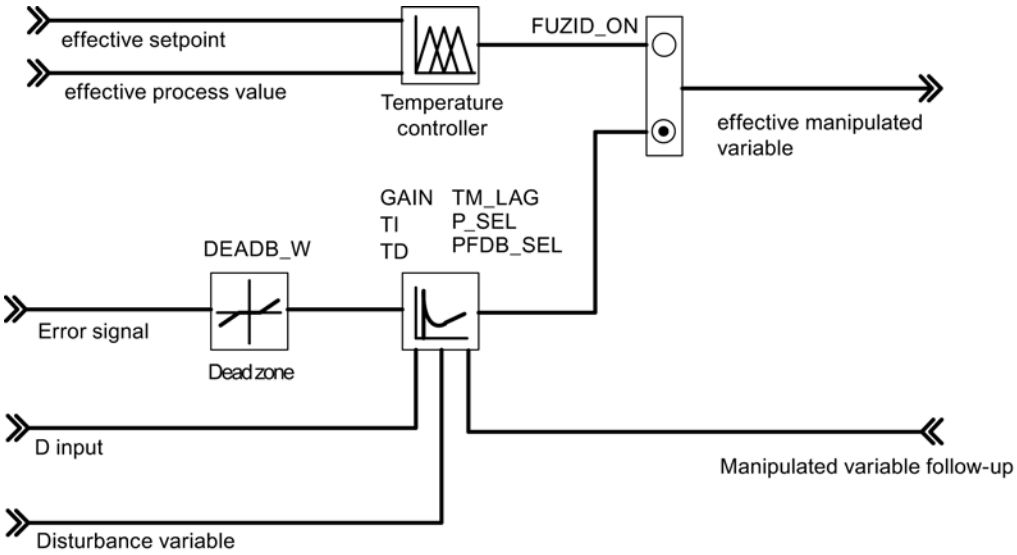
However, the parameters of the instruction PID_FM_455 are contained in all the figures – with the exception of the parameters MOD_ADDR, CHANNEL, QMOD_F, QPARA_F, QCH_F, QLMNR_ON, RET_VALU, COM_RST, LOAD_PAR, READ_VAR, LOAD_OP.

Action of the input parameters

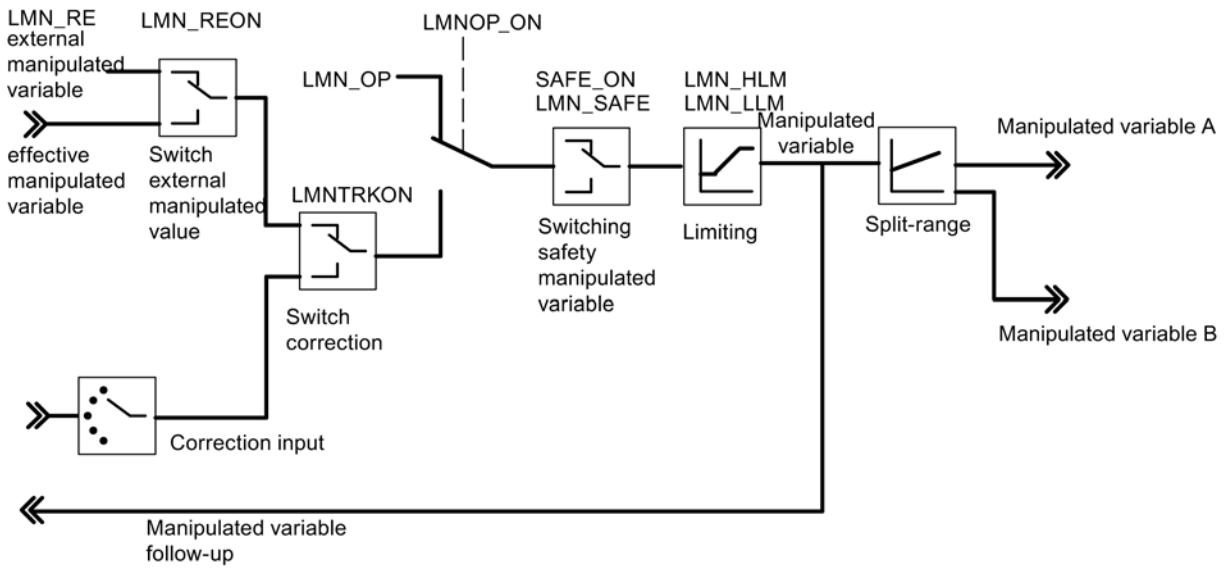
Control deviation generation with fixed-value or cascade controller



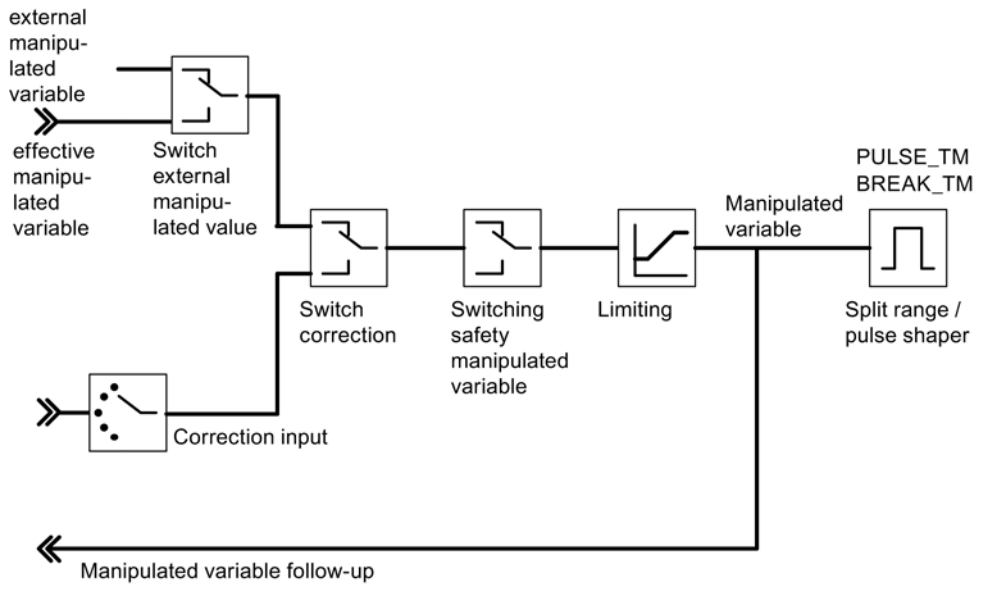
Block diagram of the control algorithm



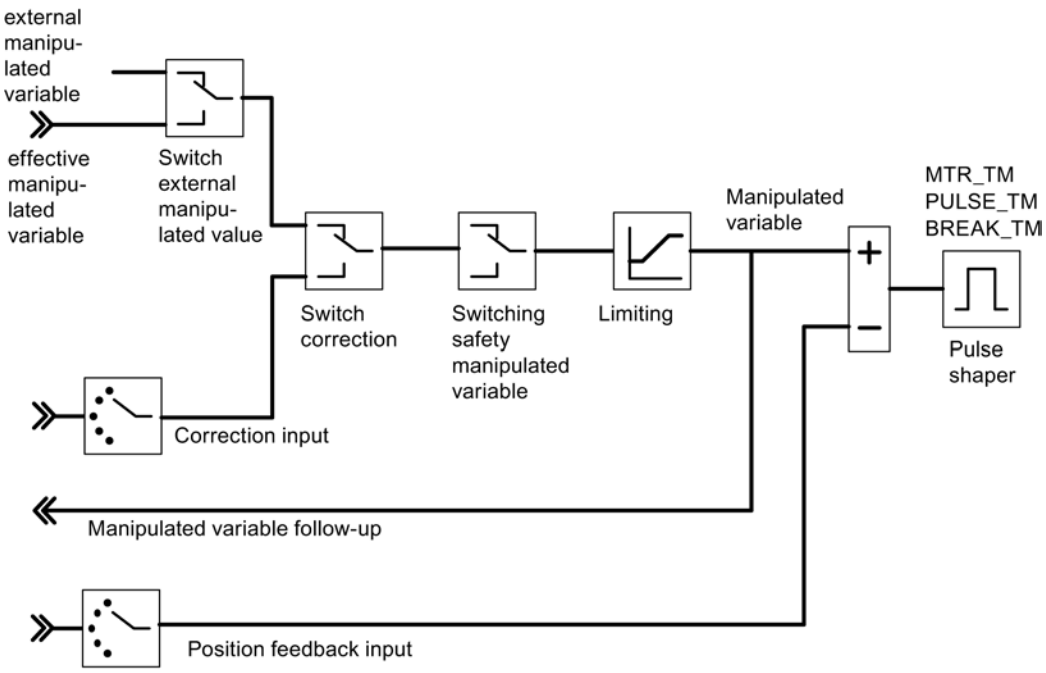
Controller output of the continuous-action controller



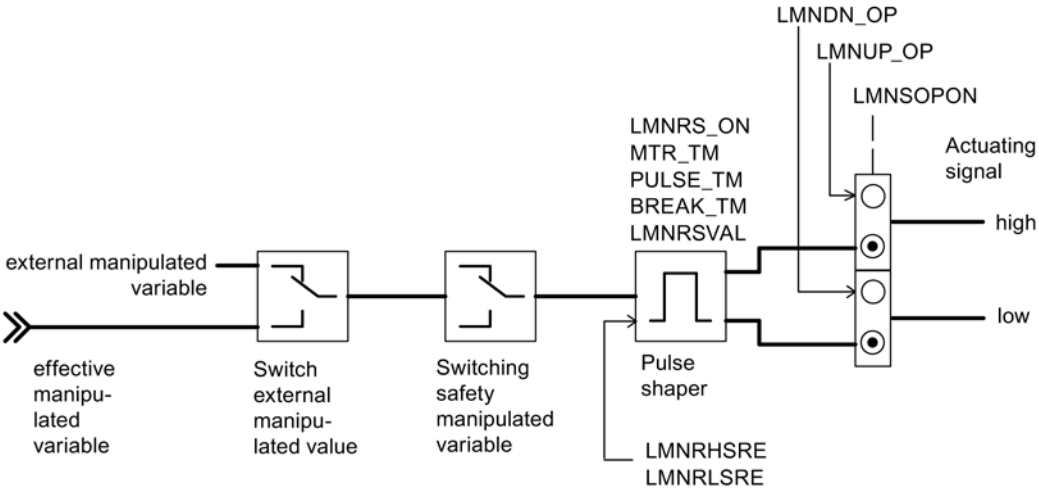
Controller output of the step controller (pulse controller operating mode)



Controller output of the step controller (step controller operating mode with position feedback)



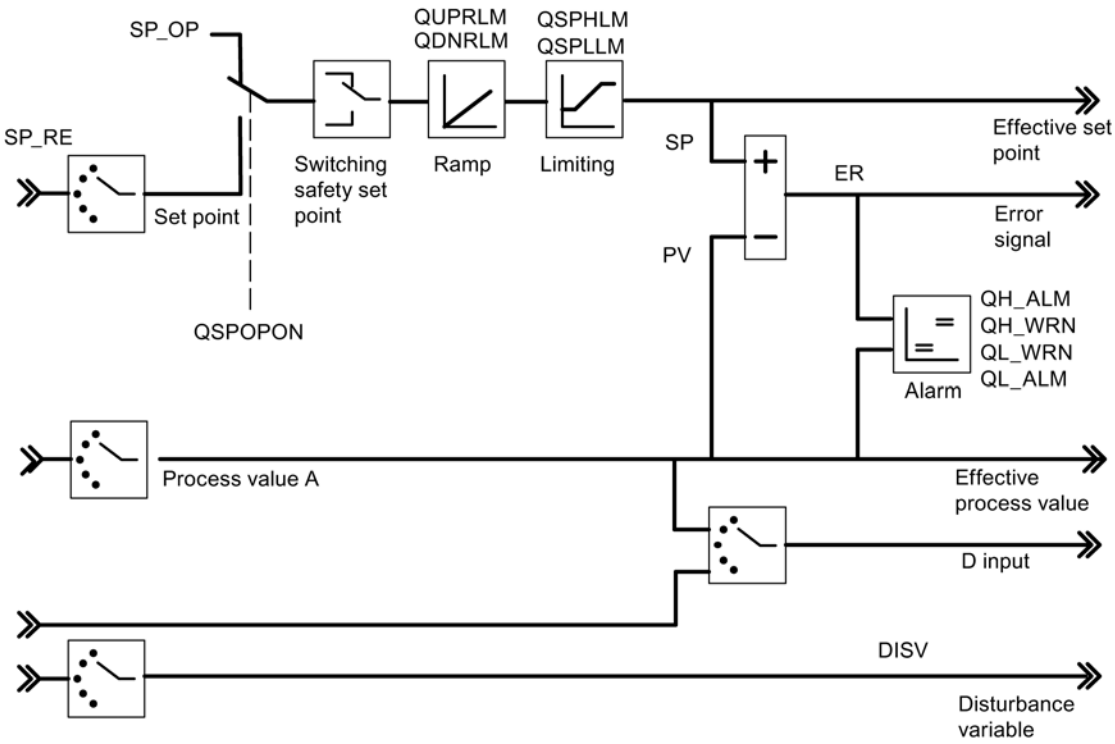
Controller output of the step controller (step controller operating mode without position feedback)



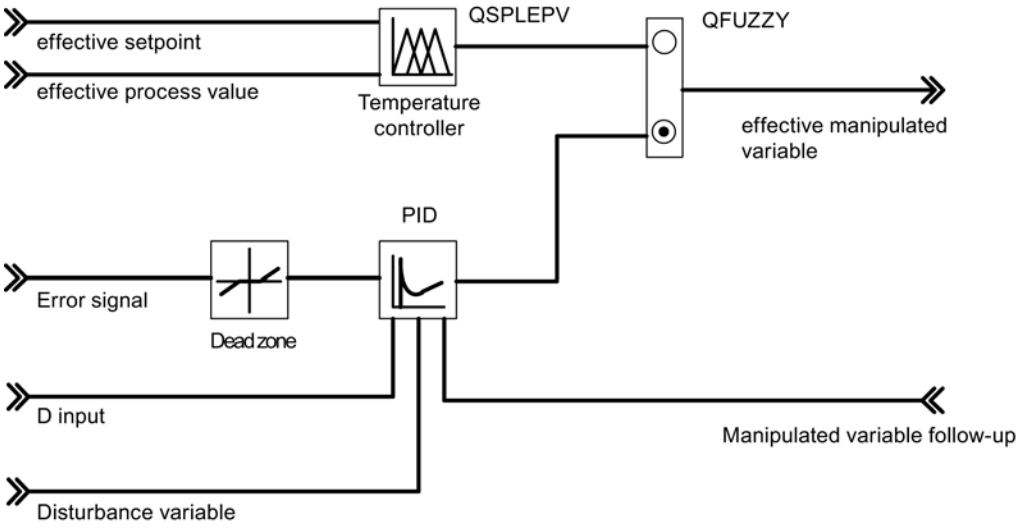
Generating output parameters

The following figures show at which points in the module the output parameters of the instruction PID_FM_455are generated.

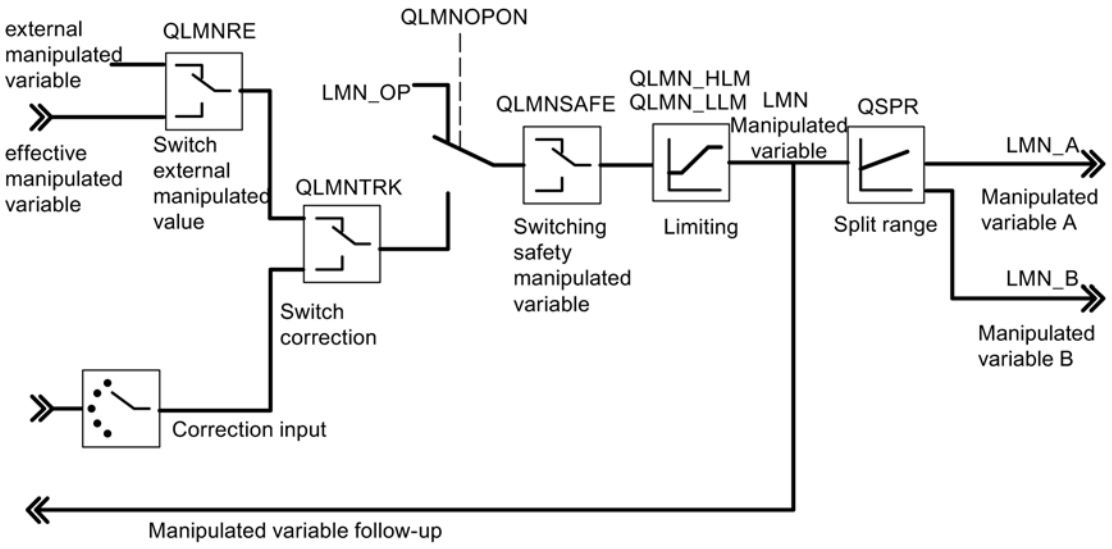
Control deviation generation with fixed-value or cascade controller



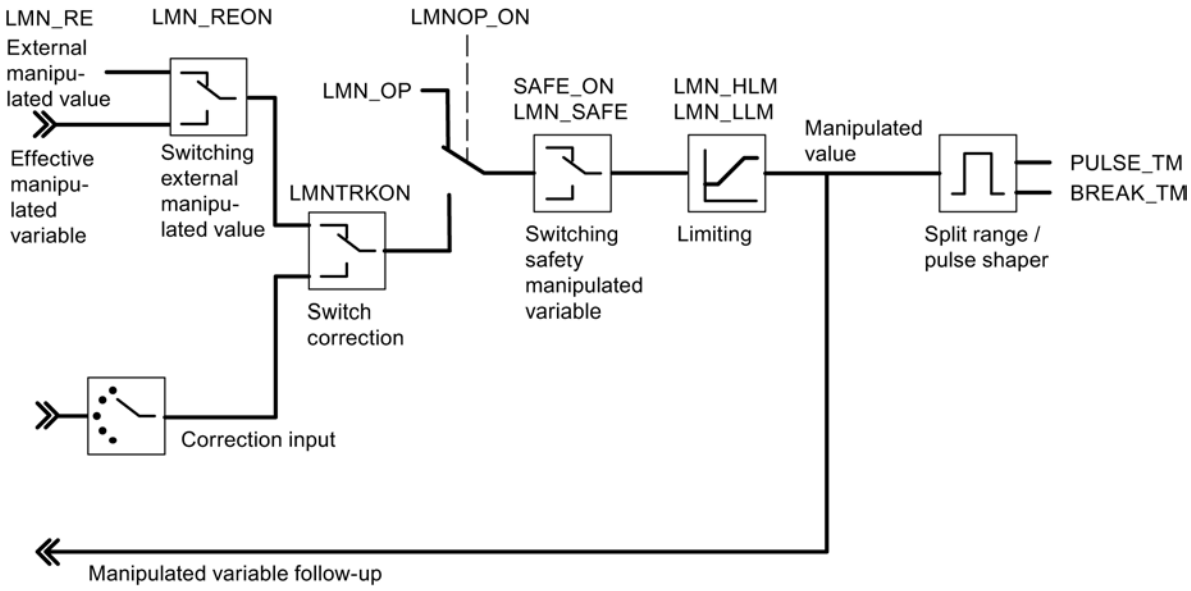
Block diagram of the control algorithm



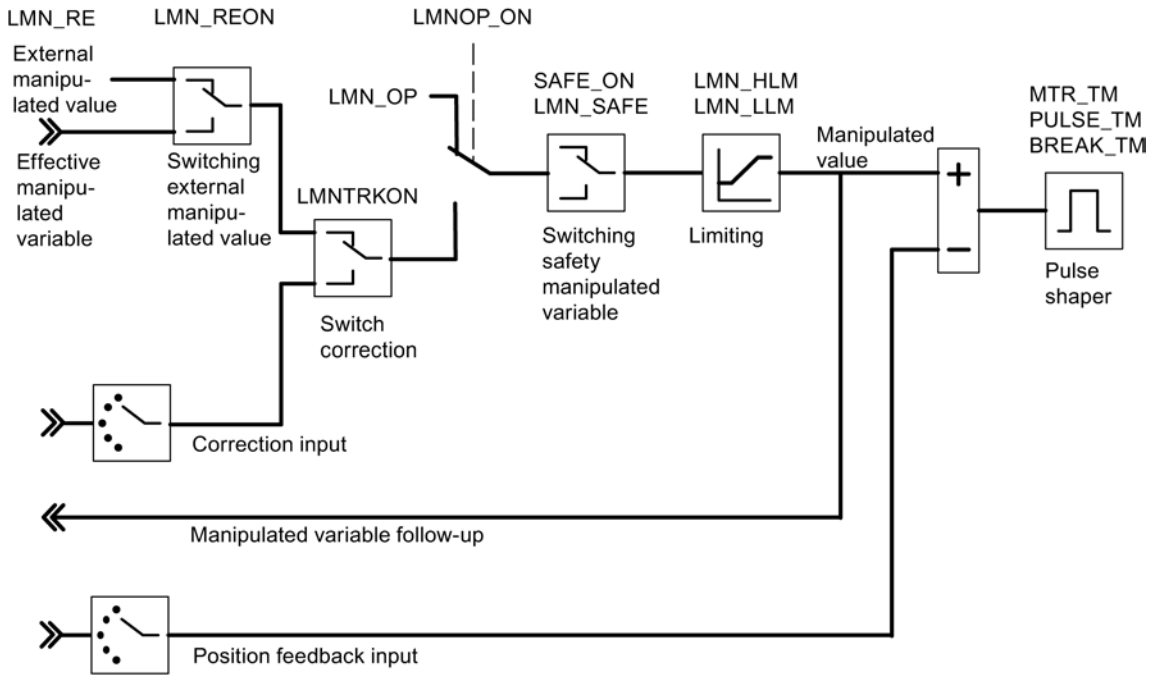
Controller output of the continuous-action controller



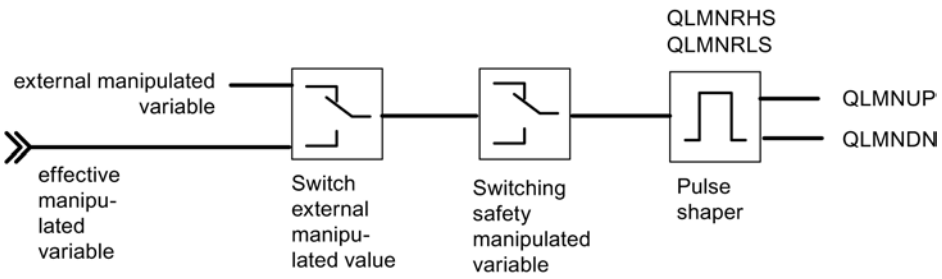
Controller output of the step controller (pulse controller operating mode)



Controller output of the step controller (step controller operating mode with position feedback)



Controller output of the step controller (step controller operating mode without position feedback)



FUZ_455

Description FUZ_455

The instruction FUZ_455 is used to read out and write the controller parameters of the fuzzy temperature controller from FM 455.

This function is suitable for the following applications:

- Transferring the controller parameters of the FM 455 that have been established by identification after the module replacement
- Adapting the FM 455 to different processes

Note

It is not permitted to modify the parameters that have been established by the FM 455 by means of an identification as they have been optimized for the process.

Mode of operation

The parameters READ_PAR and LOAD_PAR are used to control the mode of operation of the instruction FUZ_455

If READ_PAR = TRUE, FUZ_455 reads the parameters of all the temperature controllers of the FM 455 and stores them in the instance DB. After the temperature controller parameters have been successfully read out, FUZ_455 sets the parameter READ_PAR to FALSE.

If LOAD_PAR = TRUE, FUZ_455 writes the parameters of all the temperature controllers of the FM 455 from the instance DB to the FM 455. After the parameters have been successfully transferred, FUZ_455 sets the parameter LOAD_PAR to FALSE.

Call

FUZ_455 requires no initialization routine.

FUZ_455 has to be called in the same OB as all the other instructions that access the same FM 455.

Start-up

During the start-up of the CPU you should set the LOAD_PAR parameter of the FUZ_455 instruction and then call the block in the cyclic program, provided LOAD_PAR = TRUE.

Input parameters FUZ_455

Table 9-257

Parameters	Address	Data type	Default	Description
MOD_ADDR	0.0	INT	512	The module address that results from the configuration with STEP 7 is set at this input.

Output parameters FUZ_455

Table 9-258

Parameters	Address	Data type	Default	Description
RET_VALU	2.0	WORD	0	RET_VALU contains the return value RET_VAL of the WR_REC and RD_REC instructions. RET_VALU can be evaluated when QMOD_F reports an error.
PARAFFUZ	4.0	WORD	0	A parameter assignment error created by the the instruction FUZ_455 is displayed at the parameter PARAFFUZ as follows: High byte of PARAFFUZ = 01: There is a parameter assignment error.. High byte of PARAFFUZ = 00: There is no parameter assignment error. The low byte contains the offset of the parameter that caused the parameter assignment error, calculated from the static variable FUZ_PAR[1]. For instance, PARAFFUZ = W#16#0104 means that the second parameter is faulty.

See also

RD_REC: Read data record from I/O (Page 6491)

WR_REC: Write data record to I/O (Page 6496)

In/out parameters FUZ_455

Table 9-259

Parameters	Address	Data type	Default	Description
READ_PAR	6.0	BOOL	FALSE	If the READ_PAR parameter is set, the fuzzy parameters are read out from the module and stored in the static variables of the instance DB.
LOAD_PAR	6.1	BOOL	FALSE	If the LOAD_PAR parameter is set, the fuzzy parameters are read out from the static variables of the instance DB module and transferred to the module.

FORCE455

Description FORCE455

The instruction FORCE455 is used to simulate (force) the analog and digital input values to support the start-up.

Simulating Digital Values

The simulation of the values for the digital inputs 1 to 16 is enabled via the switches S_DION[i], where 1 ≤ i ≤ 16. You specify the simulation values at the parameters DI_SIM[i]. If S_DION[i] = TRUE, the value DI_SIM[i] is used Instead of the value of digital input i of the module,. The LEDs I1 to I16 always show the status of the associated digital input, even in the case of a simulation.

Simulating Analog Values

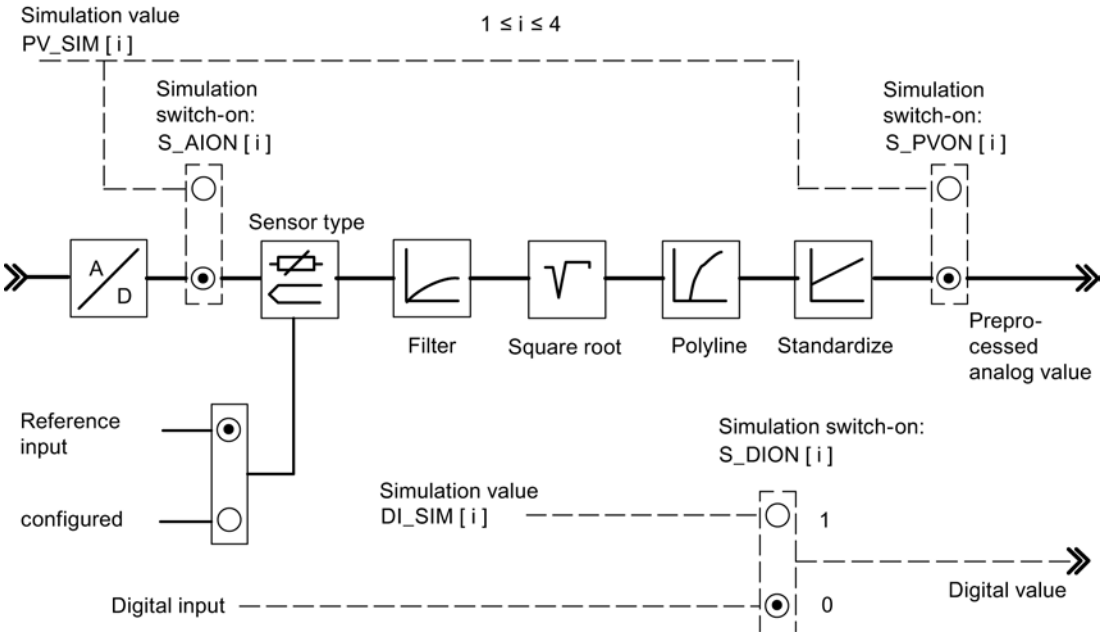
The simulation of the analog value for the channels 1 to 16 is enabled via the switches S_AION[i] or S_PVON[i], where 1 ≤ i ≤ 16.

You specify the simulation values for the channels 1 to 16 via the parameter PV_SIM[i].

You can have the simulation values become effective at two points:

- S_AION[i] = TRUE (1 ≤ i ≤ 16)
  The value PV_SIM[i] is used instead of the value of analog input i of the module.
- S_PVON[i] = TRUE (1 ≤ i ≤ 16)
  The value PV_SIM[i] is used instead of the conditioned value of analog input i of the module.

The following figure shows at which point the simulated analog value is effective.



Enabling and specification of the simulation values (forcing) is not carried out via the parameter assignment interface. This is why the relevant switches and connecting lines are drawn as dotted lines.

Call

FORCE455 does not require any initialization routine and is normally called cyclically.

FORCE455 has to be called in the same OB as all the other instructions that access the same FM 455.

Startup characteristics

When FM 455 restarts after power off, the simulation switches on the FM 455 are again set to FALSE.

Input parameters FORCE455

Table 9-260

Parameters	Address	Data type	Default	Description
S_AION	0.0	ARRAY [1..16] of BOOL	FALSE	If, for example, the S_AION[1] switch is set to TRUE, the value PV_SIM[1] is used instead of the analog input value 1 of the module.
S_PVON	2.0	ARRAY [1..16] of BOOL	FALSE	If, for example, the S_PVON[1] switch is set to TRUE, the value PV_SIM[1] is used instead of the conditioned analog input value 1 of the module.
PV_SIM	4.0	ARRAY [1..16] of REAL	0.0	The simulation value for the analog input 1 is specified, for example, at input PV_SIM[1]. If S_PVON = TRUE, then the conditioned analog input value is specified here. If S_PVON = FALSE and S_AION = TRUE, then the analog input value, which is transformed into a conditioned value by means of the conditioning functions, is specified in mA or mV. Permitted are 0.0 to 20.0 [mA], -1500 to +10000 [mV], or a technical range of values.
S_DION	68.0	ARRAY [1..16] of BOOL	FALSE	If, for example, the S_DION[1] switch is set to TRUE, the value DI_SIM[1] is used instead of the digital input value 1 of the module.
DI_SIM	70.0	ARRAY [1..16] of BOOL	FALSE	Simulation value for digital input
MOD_ADDR	72.0	INT	256	The module address that results from the configuration with STEP 7 is set at this input.

Output parameters FORCE455

Table 9-261

Parameters	Address	Data type	Default	Description
RET_VALU	74.0	WORD	0	RET_VALU contains the return value RET_VAL of the WR_REC and RD_REC instructions. RET_VALU can be evaluated when QMOD_F reports an error.

See also

RD_REC: Read data record from I/O (Page 6491)

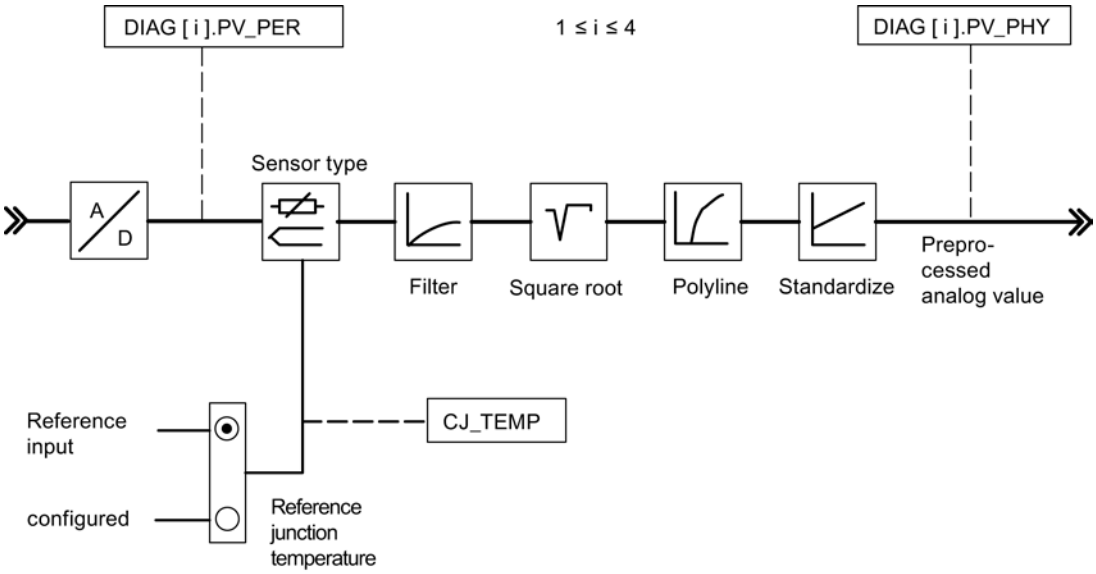
WR_REC: Write data record to I/O (Page 6496)

READ_455

Description Ready_455

The instruction READ_455 is used to read out the digital and analog input values to support the start-up.

The following values are displayed:



Call

READ_455 does not require any initialization routine and is normally called cyclically.

READ_455 has to be called in the same OB as all the other instructions that access the same FM 455.

Input parameters READ_455

Table 9-262

Parameters	Address	Data type	Default setting	Description
MOD_ADDR	0.0	INT	512	The module address that results from the configuration with STEP 7 is set at this input.

Output parameters READ_455

Table 9-263

Parameters	Address	Data type	Default	Description
CJ_TEMP	2.0	REAL	0.0	At the CJ_TEMP output, the measured reference junction temperature is displayed by the module if a thermocouple element input is configured and the reference junction temperature is not configured.
STAT_DI	6.0	ARRAY [1..16] of BOOL	FALSE	The states of digital inputs 1 to 16 are displayed at the STAT_DI parameters.
DIAG[x].PV_PER	(channel number) x 8	ARRAY [1..16] of STRUCT	0.0	The analog input value of the module is displayed, for example, in the units mA or mV, at the DIAG[1].PV_PER parameter.
DIAG[x].PV_PHY	(channel number) x 8 + 4	ARRAY [1..16] of STRUCT	0.0	The DIAG[1].PV_PHY parameter, for example, displays the conditioned analog input value of the module in the physical unit.
RET_VALU	136.0	WORD	0	RET_VALU contains the return value RET_VAL of the WR_REC and RD_REC instructions. RET_VALU can be evaluated when QMOD_F reports an error.

See also

RD_REC: Read data record from I/O (Page 6491)

WR_REC: Write data record to I/O (Page 6496)

CH_DIAG_455

Description CH_DIAG_455

The instruction CH_DIAG_455 is used to read out additional channel-specific diagnostic variables from the module.

Call

CH_DIAG_455 does not require any initialization routine and is normally called cyclically.

CH_DIAG_455 has to be called in the same OB as all the other instructions that access the same FM 455.

Input parameters CH_DIAG_455

Table 9-264

Parameters	Address	Data type	Default	Description
MOD_ADDR	0.0	INT	256	The module address that results from the configuration with STEP 7 is set at this input.
CHANNEL	2.0	INT	1	The number of the controller channel to which the instance DB is referenced is configured at the "Channel number" input. Values of 1 to 16 are permitted.

Output parameters CH_DIAG_455

Table 9-265

Parameters	Address	Data type	Default	Description
SP_R	4.0	REAL	0.0	If a ratio controller is set, the input value of the setpoint value is assigned to the parameter.
PV_R	8.0	REAL	0.0	The following value is only assigned to the parameter value if the ratio controller is set: (Process value A - Setpoint value offset) / Process value D
DIF_I	12.0	REAL	0.0	The input variable of the D-action is displayed at the DIF_I parameter. This is particularly of interest if, for example, an analog input is configured as the input variable of the D-action component.
TRACKPER	16.0	REAL	0.0	The TRACKPER parameter shows the input variable to which the manipulated value is tracked if the track-manipulated-variable function is enabled on the controller.
IDSTATUS	20.0	WORD	0.0	The state of the identification is displayed at Parameter IDSTATUS
LMN_P	22.0	REAL	0.0	The P-action of the manipulated variable is displayed at the LMN_P parameter.
LMN_I	26.0	REAL	0.0	The I-action of the manipulated variable is displayed at the LMN_I parameter.
LMN_D	30.0	REAL	0.0	The D-action of the manipulated variable is displayed at the LMN_D parameter.
RET_VALU	34.0	WORD	0	RET_VALU contains the return value RET_VAL of the WR_REC and RD_REC instructions. RET_VALU can be evaluated when QMOD_F reports an error.

See also

RD_REC: Read data record from I/O (Page 6491)

WR_REC: Write data record to I/O (Page 6496)

Parameter IDSTATUS

The IDSTATUS parameter supplies information on the status of identification and contains the four hexadecimal values X, A, I and F.

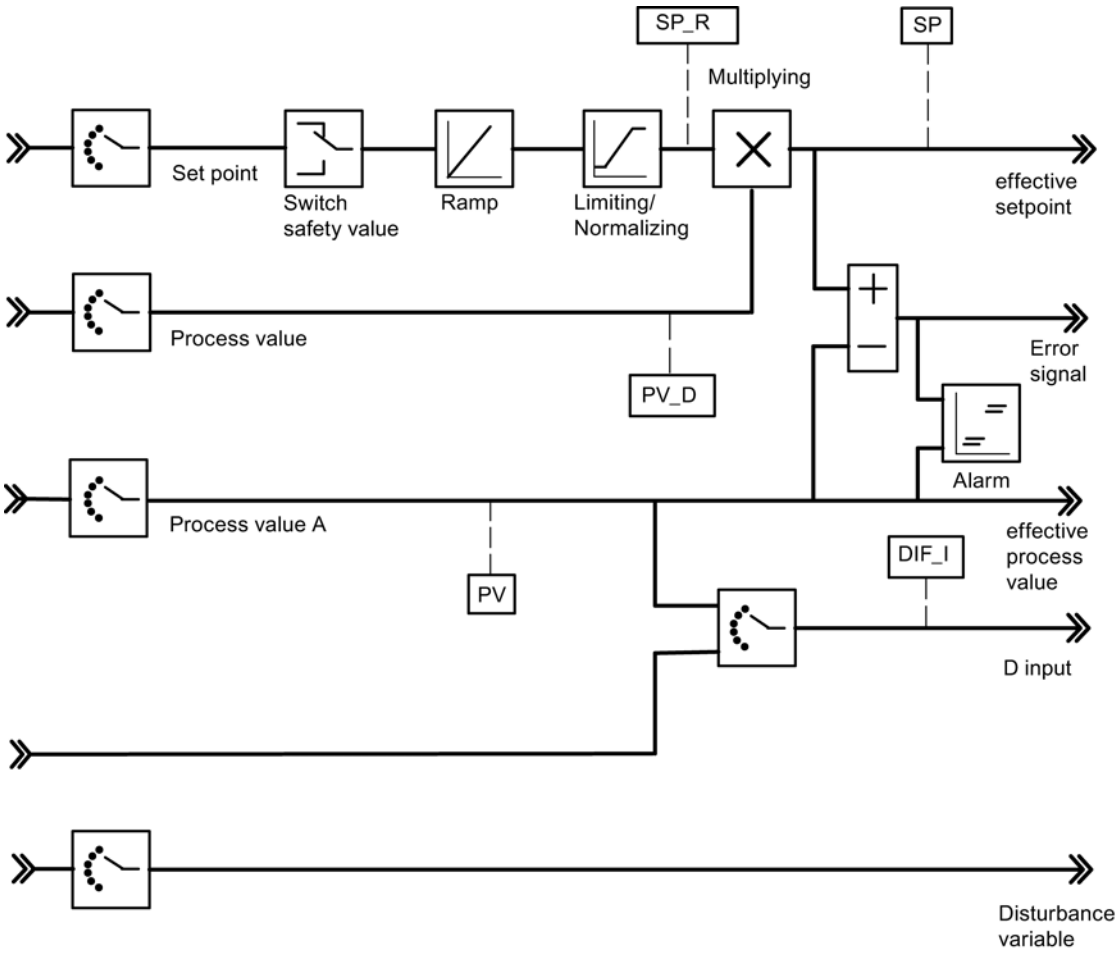
X				A				I				F			
2^{15}															2^{0}

Value	Description
X	Without meaning (always 0)
A	Action number • 0 = Manual mode (or no closed-loop control mode); • 2 = Closed-loop control; • 4 = Tuning enabled (FUZID_ON = true); • 6 = Transition state from manual mode to 2 or 4;
I	Display "Identification running" and "Parameters determined, but not yet stored in EEPROM" • 0 = Identification not running, no new parameters determined • 1 = Identification running, no new parameters determined • 2 = Identification not running, new parameters determined, but not yet stored in EEPROM • 3 = Identification running, new parameters determined, but not yet stored in EEPROM
F	Error number • 0 = No error • 4 = Excessive step-change of the process value during the identification • 5 = Ratio of time lag to system time constant too large or strongly non-linear behavior of the controlled system. • 6 = Temperature drop or rise during identification start too large. System not settled sufficiently

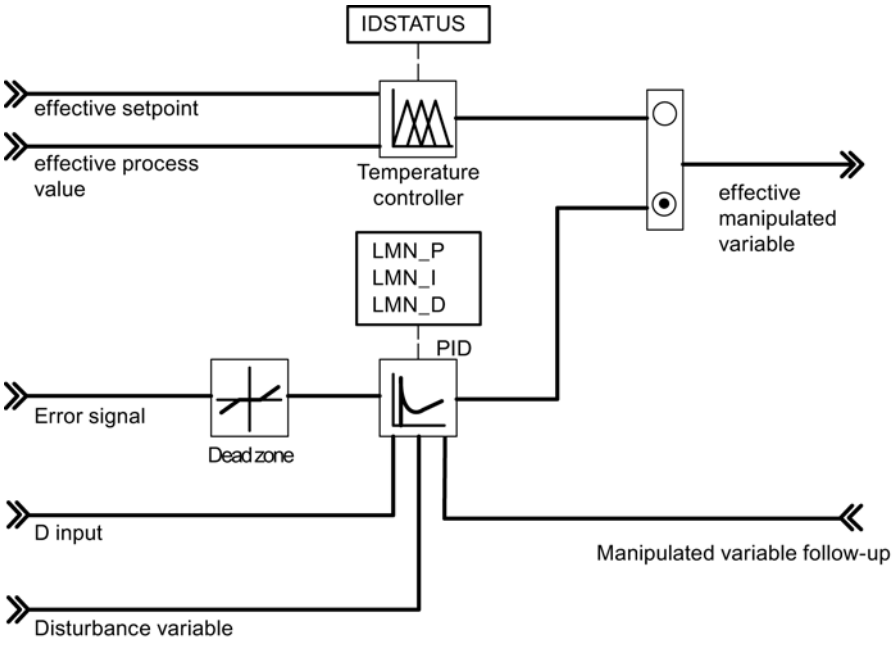
Block diagrams CH_DIAG_455

Block diagrams CH_DIAG_455

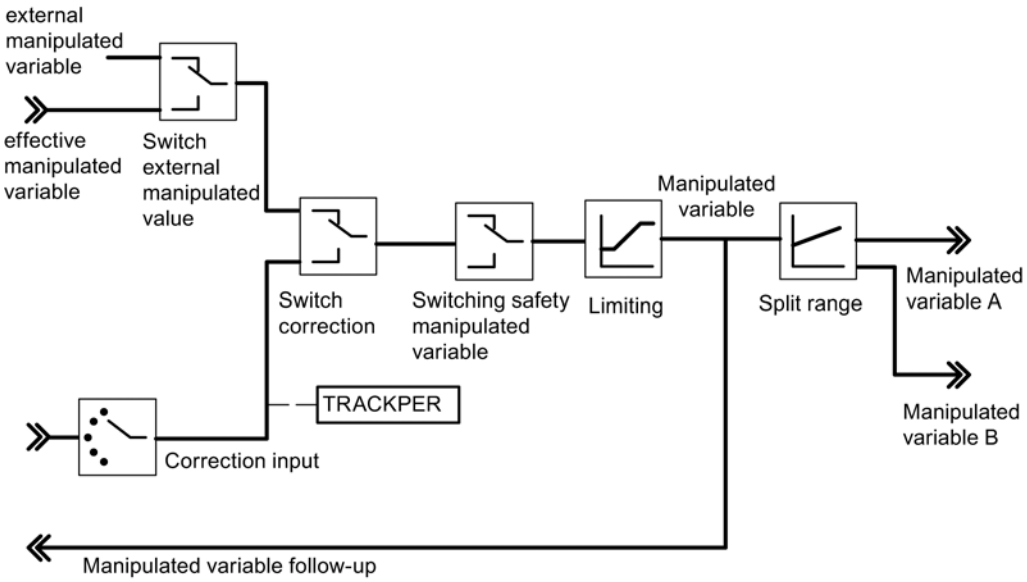
The following diagnostic values of the control deviation are displayed.



The following values of the control algorithm are displayed.



The following values of the continuous-action controller or step controller are displayed.



PID_PAR_455

Description PID_PAR_455

The instruction PID_PAR_455 is used for the online modification of parameters that cannot be defined via the instruction PID_FM_455. To view the parameters that can be modified, refer to the table Alterable parameters

Call

PID_PAR_455 requires an initialization routine, which is initiated by setting the parameter COM_RST = TRUE. To save run time, PID_PAR_455 should not be called cyclically. It should only be called when parameters are to be changed. COM_RST must then be FALSE.

PID_PAR_455 has to be called in the same OB as all the other instructions that access the same FM 455.

Mode of operation

In the initialization routine, PID_PAR_455 reads the parameters from the system data and stores them in static variables. Per call, PID_PAR_455 modifies one of these REAL variables and one of these INT variables. At the input parameter INDEX_R or INDEX_I, specify the index numbers of the parameter that you want to modify. At the input parameter VALUE_R or VALUE_I, enter the new values. PID_PAR_455 transfers the entire data record to FM 455 with the modified variables.

To modify additional parameters you have to call **the same** instance DB several times consecutively with COM_RST = FALSE and with different index numbers.

The COM_RST parameter is an input parameter that is not reset by PID_PAR_455.

Note

Please note that parameters that you have changed using PID_PAR_455 are overwritten by the parameters from the system data during the start-up.

Start-up

During CPU start-up and during CPU STOP-RUN transition, the parameters on the FM 455 are overwritten with the parameters from the SDB of the CPU.

Example

During operation you want to modify the ramp-up time for the reference variable and, depending on the process state, use different analog input values as process value.

- Call PID_PAR_455 using COM_RST = TRUE during the start-up of the CPU.
- To configure the ramp-up time for the reference variable at 10.0, use INDEX_R = 30, VALUE_R = 10.0 and INDEX_I = 0 to call PID_PAR_455 during operation.
- If you want to parameterize the analog input value 4 of the module as the process value, call PID_PAR_455 with INDEX_R = 0, INDEX_I = 50 and VALUE_I = 4 during operation.

Input parameters PID_PAR_455

Table 9-266

Parameters	Address	Data type	Default	Description
COM_RST	0.0	BOOL	TRUE	If the parameter COM_RST = TRUE, the instruction PID_PAR_455 executes an initialization routine. During this initialization process the parameters are read from the system data of the CPU and saved in the instance DB.
MOD_ADDR	2.0	INT	256	The module address that results from the configuration with STEP 7 is set at this input.
CHANNEL	4.0	INT	1	The number of the controller channel to which the instance DB is referenced is configured at the "Channel number" input. Values of 1 to 16 are permitted.
INDEX_R	6.0	INT	0.0	• 0: No REAL parameters are changed • 1 to 48: Each value corresponds to an alterable REAL parameter.
VALUE_R	8.0	REAL	0.0	Depends on the relevant parameter
INDEX_I	12.0	INT	0.0	• 0: No INT parameters are changed. • 49 to 61:: Each value corresponds to an alterable INT parameter.
VALUE_I	14.0	INT	0.0	Depends on the relevant parameter

Output parameters PID_PAR_455

Table 9-267

Parameters	Address	Data type	Default	Description
RET_VALU	16.0	WORD	0	RET_VALU contains the return value RET_VAL of the WR_REC and RD_REC instructions. RET_VALU can be evaluated when QMOD_F reports an error.
BUSY	18.0	BOOL	FALSE	If BUSY = TRUE, the parameters have not yet been applied by the module. The instruction PID_PAR_455 should then be called again in the next cycle.

See also

RD_REC: Read data record from I/O (Page 6491)
WR_REC: Write data record to I/O (Page 6496)

Alterable parameters

Table 9-268 List of the REAL and INT parameters on a FM 455 that are to be modified with PID_PAR_455

Data type	Description	Index number
-	No parameter selected	0
REAL	Filter time constants for analog input	1
REAL	End of scale (100%)	2

Data type	Description	Index number
REAL	Start of scale (0%)	3
REAL	Polyline, interpolation point 1 input side	4
REAL	Polyline, interpolation point 2 input side	5
REAL	Polyline, interpolation point 3 input side	6
REAL	Polyline, interpolation point 4 input side	7
REAL	Polyline, interpolation point 5 input side	8
REAL	Polyline, interpolation point 6 input side	9
REAL	Polyline, interpolation point 7 input side	10
REAL	Polyline, interpolation point 8 input side	11
REAL	Polyline, interpolation point 9 input side	12
REAL	Polyline, interpolation point 10 input side	13
REAL	Polyline, interpolation point 11 input side	14
REAL	Polyline, interpolation point 12 input side	15
REAL	Polyline, interpolation point 13 input side	16
REAL	Polyline, interpolation point 1 output side	17
REAL	Polyline, interpolation point 2 output side	18
REAL	Polyline, interpolation point 3 output side	19
REAL	Polyline, interpolation point 4 output side	20
REAL	Polyline, interpolation point 5 output side	21
REAL	Polyline, interpolation point 6 output side	22
REAL	Polyline, interpolation point 7 output side	23
REAL	Polyline, interpolation point 8 output side	24
REAL	Polyline, interpolation point 9 output side	25
REAL	Polyline, interpolation point 10 output side	26
REAL	Polyline, interpolation point 11 output side	27
REAL	Polyline, interpolation point 12 output side	28
REAL	Polyline, interpolation point 13 output side	29
REAL	Ramp-up time for control variable	30
REAL	Safety reference variable or safety reference variable ratio	31
REAL	Offset for the setpoint logic operation (ratio / mixed controllers)	32
REAL	Factor for process value B (three-component controllers)	33
REAL	Factor for process value C (three-component controllers)	34
REAL	Offset for process value logic operation (three-component controllers)	35
REAL	Factor for disturbance variable logical operation	36
REAL	Operating point	37
REAL	Aggressiveness at fuzzy controller	38
REAL	Vertices for split-range function: Start of input signal A area	39
REAL	Vertices for split-range function: End of input signal A area	40
REAL	Vertices for split-range function: Start of output signal A area	41
REAL	Vertices for split-range function: End of output signal A area	42
REAL	Vertices for split-range function: Start of input signal B area	43
REAL	Vertices for split-range function: End of input signal B area	44
REAL	Vertices for split-range function: Start of output signal B area	45

Data type	Description		Index number
REAL	Vertices for split-range function: End of output signal B area		46
REAL	Minimum pulse time		47
REAL	Minimum break time		48
INT	Selection of the reference variable SP or SP_RE for the controller		49
	0	Setpoint SP_RE of the instruction	
	1 to 16	Analog input value 1 to 16	
	17 to 32	Manipulated variable (LMN) of controllers 1 to 16	
INT	Selecting the main controlled variable process value A for the controllers		50
	0:	Process value A = 0.0	
	1 to 16:	Analog input value 1 to 16	
INT	Selecting the main controlled variable process value B for the controllers		51
	0	Process value B = 0.0	
	1 to 16	Analog input value 1 to 16	
INT	Selecting the auxiliary controlled variable process value C for the controllers		52
	0	Process value C = 0.0	
	1 to 16	Analog input value 1 to 16	
INT	Selecting the auxiliary controlled variable process value D for the controllers		53
	0	Process value D = 0.0	
	1 to 16	Analog input value 1 to 16	
	17 to 32	Manipulated variable (LMN) of controllers 1 to 16	
INT	Selection of the disturbance variable DISV for the controller		54
	0	Disturbance variable = 0.0	
	1 to 16	Analog input value 1 to 16	
INT	Selection of position follow-up TRACK_PER for the controller		55
	0	Position follow-up = 0.0	
	1 to 16	Analog input value 1 to 16	
INT	Selection of position follow-up LMNR_PER for the controller		56
	0	Position follow-up = 0.0	
	1 to 16	Analog input value 1 to 16	
INT	Selecting the signal for changeover to safety value for the manipulated value of the controller		57
	0	Can only be assigned via SAFE_ON of the instruction PID_FM_455	
	1 to 16	Assignment via SAFE_ON parameter of the instruction PID_FM_455 ORed with digital input 1 to 16	
INT	Selecting the signal for changeover to the follow-up function of the manipulated value of the controller		58
	0	Can only be assigned via LMNTRKON of the instruction PID_FM_455	
	1 to 16	Assignment via LMNTRKON parameter of the instruction PID_FM_455 ORed with digital input 1 to 16	
INT	Selecting the signal for changing over the manipulated value of the controller to LMN_RE		59
	0	Can only be assigned via LMN_REON of the instruction PID_FM_455	
	1 to 16	Assignment via LMN_REON parameter of the instruction PID_FM_455 ORed with digital input 1 to 16	
INT	Selecting the high limit signal of the position feedback		60

Data type	Description		Index number
	0	Can only be assigned via LMNRHSRE of the instruction PID_FM_455	
	1 to 16	Assignment via LMNRHSRE parameter of the instruction PID_FM_455 ORed with digital input 1 to 16	
INT	Selecting the low limit signal of the position feedback		61
	0	Can only be assigned via LMNRLSRE of the instruction PID_FM_455	
	1 to 16	Assignment via LMNRLSRE parameter of the instruction PID_FM_455 ORed with digital input 1 to 16	

CJ_T_PAR_455

Description CJ_T_PAR_455

The instruction CJ_T_PAR_455 is used for online modification of the configured reference junction temperature. This is necessary if a temperature control system with several FM 455 with thermoelement inputs is to be operated without having to connect a Pt 100 to each FM 455.

Call

CJ_T_PAR_455 requires an initialization routine. To this purpose it has to be called once with the COM_RST = TRUE parameter at the start-up of the CPU.

CJ_T_PAR_455 is normally called cyclically. For this purpose COM_RST should be set to FALSE for runtime reasons. The COM_RST parameter is an input parameter that is not reset.

CJ_T_PAR_455 has to be called in the same OB as all the other instructions that access the same FM 455.

Example

If an FM 455 measures the reference junction temperature of an extruder control system with more than 16 heating zones, this can be read out by READ_455 at parameter CJ_TEMP and can be parameterized for the other FM 455 via CJ_T_PAR_455.

Input parameters CJ_T_PAR_455

Table 9-269

Parameters	Address	Data type	Default	Description
COM_RST	0.0	BOOL	–	If the parameter COM_RST = TRUE, the instruction CJ_T_PAR_455 executes an initialization routine. During this initialization process the parameters are read from the system data of the CPU and saved in the instance DB.
MOD_ADDR	2.0	INT	256	The module address that results from the configuration with STEP 7 is set at this input.
CJ_T	4.0	REAL	0.0	The reference junction temperature can be specified at parameter CJ_T. The permissible value range depends on the sensor type.

Output parameters CJ_T_PAR_455

Table 9-270

Parameters	Address	Data type	Default	Description
RET_VALU	8.0	WORD	0	RET_VALU contains the return value RET_VAL of the WR_REC and RD_REC instructions. RET_VALU can be evaluated when QMOD_F reports an error.
BUSY	10.0	BOOL	FALSE	If BUSY = TRUE, the parameters have not yet been applied by the module. The instruction CJ_T_PAR_455 should then be called again in the next cycle.

See also

RD_REC: Read data record from I/O (Page 6491)

WR_REC: Write data record to I/O (Page 6496)

LP_ZONE

Description LP_ZONE

To optimize load on CPU resources and processing of the instruction ZONE at constant bus cycle times at during controller optimization, it is necessary to use the loop scheduler LP_ZONE.

LP_ZONE ensures that only one zone is processed per call of the cyclic interrupt OB.

The instruction LP_ZONE calculates the zone sampling times and transfers them to the instances of instruction ZONE.

Call

LP_ZONE is called in a start-up or cyclic interrupt OB. The instructions LP_ZONE, ADM_ZONE and ZONE must all be called at the same cyclic interrupt level, preferably directly in the cyclic interrupt OB. It is not permitted to program separate time reduction ratios for the blocks.

LP_ZONE requires an initialization routine after the insertion or removal of a zone or during the the modification of a call sequence. This initialization routine must be called with the parameter COM_RST = TRUE. The COM_RST bit is passed to the individual zones and their administration block via the LP_SCH interface.

The zone numbers NBR_CALL of the instances of the instructions ZONE and ADM_ZONE and the maximum zone number MNR_CALL of the instruction LP_ZONE are automatically set according to the following routine:

Initial execution of the cyclic interrupt OB after COM_RST of the instruction LP_ZONE has been set:

- LP_ZONE sets the call counter LP_SCH. CALL_CNT to 1.
- ZONE and ADM_ZONE read the call counter and write the value to their in/out parameterNBR_CALL. ZONE increments the call counter by the count by 1. ADM_ZONE leaves it unchanged.

Second run of the cyclic interrupt OB after COM_RST of the instruction LP_ZONE has been set:

- LP_ZONE reads the call counter (total number of zones) and writes the value to MNR_CALL.

Mode of operation

Parameter LP_SCH.CALL_CNT is incremented by the count of 1 at every call of the instruction LP_ZONE. The counter is reset to 1 after it has reached the maximum zone number of MNR_CALL. When the ZONE instruction is called, it first check the counter LP_SCH.CALL_CNT. If the counter value and internal zone number NBR_CALL match, the instruction executes its program. The parameter MNR_CALL of the instruction LP_ZONE and NBR_CALL of the instruction ADM_ZONE and ZONE are automatically set by the blocks after a restart.

Note

Interconnect the in/out parameter LP_SCH of the instructions ZONE and ADM_ZONE to the internal structure LP_SCH of the instance DB of the instruction LP_ZONE.

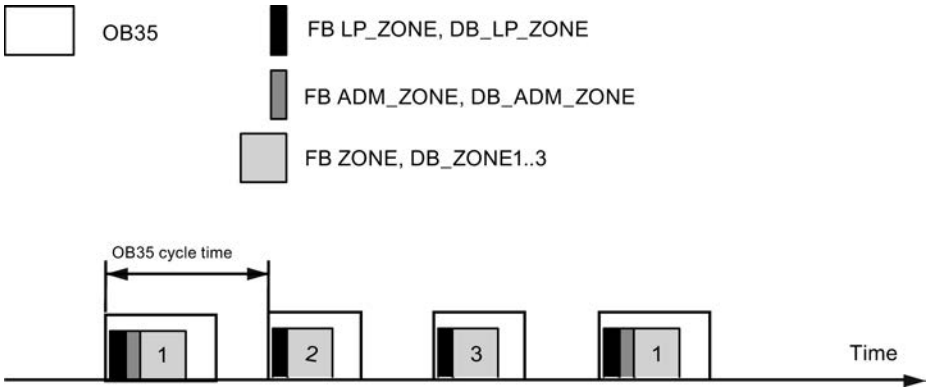
Interconnect input parameter OB_EXC_FREQ of the instruction LP_ZONE to the temporary parameter OBxx_EXC_FREQ of the cyclic interrupt OB.

Example

Call sequence in OB33

```
OB33
 ...
 call LP_ZONE, DB_LP_ZONE
 call ADM_ZONE, DB_ADM_ZONE
 call ZONE, DB_ZONE1
 call ZONE, DB_ZONE2
 call ZONE, DB_ZONE3
 …
 BE
```

leads the following zone processing cycle:



Input parameters LP_ZONE

Table 9-271

Parameters	Address	Data type	Default	Description
OB_EXC_FREQ	0.0	INT	0	Sampling time [ms]; interconnect the parameter with the corresponding local variables of the cyclic interrupt OB.

Output parameters LP_ZONE

Table 9-272

Parameters	Address	Data type	Default	Description
MNR_CALL	2.0	INT	0	Number of zones. Defined automatically after restart.

In/out parameters LP_ZONE

Table 9-273

Parameters	Address	Data type	Default	Description
COM_RST	4.0	BOOL	TRUE	Restart; make sure that this bit is set in OB100.

Static variables LP_ZONE

Table 9-274

Parameters	Address	Data type	Default	Description
LP_SCH	6.0	STRUCT		Interface between the instructions LP_ZONE and ADM_ZONE or ZONE .

This is followed by internal parameters which may not be used by users.

ADM_ZONE

Description ADM_ZONE

In practice, heating/cooling zones are designed to allow relatively unhindered heat transfer between zones. In such cases, the thermally coupled zones must be organized into a group. The instruction ADM_ZONE manages a group of heating/cooling zones and handles the following functions:

- Switching the heating of all zones on and off
- Enabling and disabling controller optimization
- Group display for channel faults
- Group display for controller optimization errors
- Group display for controller optimization warnings
- Control of the global manipulated value step-changes during controller optimization

Call

Call the instructions LP_ZONE, ADM_ZONE and ZONE at the same cyclic interrupt level, preferably immediately in the cyclic interrupt OB. It is not permitted to program separate time reduction ratios for the blocks. Call ADM_ZONE after LP_ZONE and before ZONE.

Start-up

Set the restart bit of ADM_ZONE, for example in OB100, or in the first cycle of the cyclic interrupt OB. The heating is disabled at the selected zones after restart and any active controller optimization is canceled.

Mode of operation

The actual program is only executed if the call counter of the instruction LP_ZONE has the same value as the internal call number: LP_SCH.CALL_CNT = NBR_CALL.

Switching the heating of all zones on and off

Changes at switch HEAT_OFF are transferred as actions to the ADMIN interface and are successively passed to the zones to be administered. If no action is pending, the result is transferred from the ADMIN.QHEAT_OFF zones to in/out parameter HEAT_OFF. Group switch HEAT_OFF is only set to TRUE if the heating is shut down at all zones.

User actions at the HEAT_OFF switch are indicated at the done parameter QHO_DONE. The parameter is set to TRUE by default. It is set to FALSE after a user action. QHO_DONE is reset to TRUE after the heating has been switched on or off at all associated zones.

A zone heating is only in disabled state if the following condition is met: cont_FM455.QLMN_RE = TRUE and cont_FM455.LMN = 0.0

Switching controller optimization on and off at all zones

Changes at switch TUN_ON are transferred as actions to the ADMIN interface and are successively passed to the zones to be administered. If no action is pending, the result is transferred from the ADMIN.QTUN_ON zones to in/out parameter TUN_ON. Group switch TUN_ON is only set to FALSE if controller optimization is disabled at all zones.

Group display for channel faults

Bit QCH_F indicates whether a channel error of FM 455 is pending on at least one zone. Check on which zone a channel error has occurred. If QCH_F = TRUE, then the precise error information is read out in the diagnostic record DS1 of the module.

Group display for controller optimization errors

Bit QTUN_ERR indicates if an error has occurred in at least one instance DB of the ZONESTATUS_H = 3xxxx or STATUS_C = 3xxxx instruction. Check the corresponding zones and to remedy errors follow the steps described at Parameters STATUS_H and STATUS_C.

Group display for controller optimization warnings

Bit QTUN_ERR indicates if an error has occurred in at least one instance DB of the ZONESTATUS_H = 2xxxx or STATUS_C = 2xxxx instruction. Check the corresponding zones and to remedy errors follow the steps described at Parameters STATUS_H and STATUS_C.

Sequential control during controller optimization

The administration block ensures that the manipulated value step-change is carried out simultaneously at all zones of this group during controller optimization. After having found their point of inflection, all zones change over to phase 7.

Note

Interconnect the in/out parameter ADMIN of the instruction ZONE with the internal structure ADMIN of the instance DB of ADM_ZONE.

ADM_ZONE is always used in conjunction with LP_ZONE.

Output parameters ADM_ZONE

Table 9-275

Parameters	Address	Data type	Default	Description
QCH_F	0.0	BOOL	FALSE	Group message: Channel error
QTUN_ERR	0.1	BOOL	FALSE	Group message: Error during controller optimization

Parameters	Address	Data type	Default	Description
QTUN_WRN	0.2	BOOL	FALSE	Group message: Warning during controller optimization
QHO_DONE	0.3	BOOL	TRUE	Feedback of a HEAT_OFF operation • 0: HEAT_OFF operation is triggered • 1: HEAT_OFF operation is active at all zones
NBR_CALL	2.0	INT	0	Call number; entered automatically by the instruction after restart.

In/out parameters ADM_ZONE

Table 9-276

Parameters	Address	Data type	Default	Description
HEAT_OFF	4.0	BOOL	TRUE	Heating off/on; 1 = off
TUN_ON	4.1	BOOL	FALSE	Controller optimization on/off; 1 = on
COM_RST	4.2	BOOL	TRUE	Restart; make sure that this bit is set in OB100.
LP_SCH	6.0	STRUCT	P#P 0.0	Interface between the instructions LP_ZONE and ADM_ZONE or ZONE.

Static variables ADM_ZONE

Table 9-277

Parameters	Address	Data type	Default	Description
ADMIN	12.0	STRUCT		Interface between the instructions ADM_ZONE and ZONE .

This is followed by internal parameters which may not be used by users.

ZONE

Description ZONE

The instance DB of the instruction ZONE contains the controller data of an individual heating or heating/cooling zone. The instruction ZONE reads and writes data to a channel of FM 455.

The controller functions of FM 455 are supplemented with the following functions:

- Control zone operation
- Ratio factor for heating/cooling zones
- Controller optimization

ZONE is therefore particularly suitable for controlling the heating and heating/cooling zones of extruders.

Requirements

The channel of the FM 455 is configured as pulse controller or continuous controller.

The instructions PID_FM_455 and PID_PAR_455 are copied to the user program.

Call

Call the instructions LP_ZONE, ADM_ZONE and ZONE at the same cyclic interrupt level, preferably immediately in the cyclic interrupt OB. It is not permitted to program separate time reduction ratios for the blocks. It is essential to position the associated zones in the block after the call of the instruction ADM_ZONE.

Start-up

In OB100, set the restart bit of ZONE. The bit is set automatically if you are using LP_ZONE.

Following actions are carried out when initialization routine COM_RST=TRUE

is initiated:

- cont_FM455.SDB_SEL = TRUE: SDB controller parameters cont_FM455.P_SEL...cont_FM455.BREAK_TM are not applied by FM 455. The parameters active in the FM remain active.
- cont_FM455.SP_OP_ON = FALSE: This setting activates setpoint SP_RE.
- cont_FM455.LOAD_PAR = TRUE: The controller parameters are transferred from the instance DB to FM 455.

Mode of operation

ZONE includes the instructions PID_FM_455 and PID_PAR_455 as local instances.

ZONE sets the parameters cont_FM455.LOAD_OP = TRUE and cont_FM455.READ_VAR = TRUE at each call. All operating parameters (cont_FM455.SP_RE...cont_FM455.LMNRSVAL) and all process values (cont_FM455.SP...cont_FM455.QMOD_F) are updated as a result.

Controller parameters (cont_FM455.P_SEL...cont_FM455.BREAK_TM) can be modified directly by entering a new value at the instance DB on the CPU and setting the cont_FM455.LOAD_PAR bit for the duration of one cycle. You do not require an auxiliary DB. The bit is reset on successful completion of the transfer. After the restart of the CPU the controllers are not overwritten by the controller parameters of the SDB.

If you are working without LP_ZONE, you must set parameter NBR_CALL=0.

Parameters of instruction PID_FM_455 that are alterable with ZONE

Table 9-278 The following list contains the parameters of instruction PID_FM_455 that are affected by ZONE:

Parameters	Comment
cont_FM455.READ_VAR	Set cyclically to TRUE
cont_FM455.LOAD_OP	Set cyclically to TRUE

Parameters	Comment
cont_FM455.LOAD_PAR	Set to TRUE during controller optimization and during modification of the CONZONE and RATIOFAC parameters.
cont_FM455.MOD_ADDR	Set cyclically at input MOD_ADDR.
cont_FM455.CHANNEL	Set cyclically at input CHANNEL.
cont_FM455.SDB_SEL	Set to TRUE at restart. At the CPU transition from RUN to STOP, FM 455 rejects the controller parameters (P_SEL...BREAK_TM) of the SDB.
cont_FM455.SP_OP_ON	Set to FALSE at the restart so that SP_RE is active.
cont_FM455.SP_RE	Set cyclically to SP_IN
cont_FM455.GAIN cont_FM455.TI cont_FM455.TD cont_FM455.TM_LAG cont_FM455.D_EL_SEL cont_FM455.PFDB_SEL cont_FM455.LOAD_PAR	Set during controller optimization, when restoring controller parameters from backup, and when loading PI/PID controller parameters.
cont_FM455.LMN_LLM cont_FM455.LOAD_PAR	Set after modification of RATIOFAC and COOLZONE
cont_FM455.LMN_REON cont_FM455.LMN_RE cont_FM455.LOAD_OP	Set cyclically

Parameters of instruction PID_PAR_455 that are alterable with ZONE

Table 9-279 The following list contains the parameters of instruction PID_PAR_455, that are affected by ZONE:

Parameters	Comment
pidpar_FM455.COM_RST	Set to TRUE at restart
pidpar_FM455.MOD_ADDR	Set cyclically at input MOD_ADDR.
pidpar_FM455.CHANNEL	Set cyclically at input CHANNEL.
pidpar_FM455. INDEX_R	Set to 43 after modification of RATIOFAC and COOLZONE
pidpar_FM455. VALUE _R	Set after modification of RATIOFAC and COOLZONE:

Input parameters ZONE

Table 9-280

Parameters	Address	Data type	Default	Description
MOD_ADDR	0.0	INT	512	Module address
CHANNEL	2.0	INT	1	Channel number

Parameters	Address	Data type	Default	Description
LPSCH_EN	4.0	BOOL	TRUE	If LPSCH_EN = TRUE, ZONE is controlled via the instruction LP_ZONE. The in/out parameter LP_SCH must be interconnected with the static variable LP_SCH of the instruction LP_ZONE. Otherwise the CPU goes to STOP with "Range error while reading". If LPSCH_EN = FALSE, ZONE is processed at each cyclic interrupt cycle. It is not necessary to supply the in/out parameter LP_SCH. The sampling time must be entered at a static variable CYCLE.
ADMIN_EN	4.1	BOOL	TRUE	If ADMIN_EN = TRUE, several heating/cooling zones are handled via the instruction ADM_ZONE. The in/out parameter ADMIN must be interconnected with the static variable ADMIN of the instruction ADM_ZONE. Otherwise the CPU goes to STOP with "Range error while reading". If ADMIN_EN = FALSE, ZONE works without the instruction ADM_ZONE. It is not necessary to supply the in/out parameter ADMIN.

In/out parameters ZONE

Table 9-281

Parameters	Address	Data type	Default	Description
LP_SCH	6.0	STRUCT	P#P 0.0	Interface between the instructions LP_ZONE and ADM_ZONE or ZONE.
ADMIN	12.0	STRUCT	P#P 0.0	Interface between the instructions ADM_ZONE and ZONE .

Static variables ZONE

Table 9-282

Parameters	Address	Data type	Default	Description
cont_FM455	18.0	FB PID_FM_455		Local instance of the instruction PID_FM_455
pidpar_FM455	172.0	FB PID_PAR_455		Local instance of the instruction PID_PAR_455
SP_IN	426.0	REAL	0.0	The input SP_IN is used to specify a setpoint.
CYCLE	430.0	REAL	0.1	Sets the sampling time for the PID algorithm. If used, the instruction LP_ZONE automatically calculates the sampling time of LP_ZONE.
H_OFF_EN	434.0	BOOL	TRUE	TRUE = switch HEAT_OFF is active and affects the parameters MAN_ON and MAN.
TUN_EN	434.1	BOOL	TRUE	No new controller parameters are calculated if TUN_EN=0 during controller optimization. Caution: the manipulated value step-change is enabled regardless of this setting.

Parameters	Address	Data type	Default	Description
COOLZONE	434.2	BOOL	FALSE	COOLZONE = FALSE: Self-contained heating zone COOLZONE = TRUE: Heating/cooling zone
PID_ON	434.3	BOOL	TRUE	At the PID_ON input, you can specify whether or not the tuned controller will operate as a PI or PID controller. PID controller: PID_ON = TRUE PI controller: PID_ON = FALSE With certain process types it is nevertheless possible that only a PI controller will be designed despite PID_ON = TRUE.
QCH_F	434.4	BOOL	FALSE	QCH_F = TRUE: A channel error of FM 455 has occurred during the calling of the local instance of the instruction PID_FM_455 (parameter cont_FM455.QCH_F).
QTUN_ERR	434.5	BOOL	FALSE	QTUN_ERR = TRUE: Incorrect operator action occurred during controller optimization. STATUS_H or STATUS_C = 3xxxx
QTUN_WRN	434.6	BOOL	FALSE	QTUN_WRN = TRUE: : Warning message occurred during controller optimization. STATUS_H or STATUS_C = 2xxxx
QTUNSTCP	434.7	BOOL	FALSE	QTUNSTCP = TRUE: Tuning was initiated in cold process state and the controller is to be corrected to the operating point after optimization. MAN_ON = FALSE; MAN = 0 (Heating off) SP_IN > PV0 +10 (At the start of optimization, the operating point is greater than the process value by the amount of 10)
QCONTINU	435.0	BOOL	TRUE	Indicates whether the zone is ready for the next phase of controller optimization
PHASE	436.0	INT	0	The currently active phase of controller optimization is indicated at the Parameter PHASE (0 to 7, 11 to13).
STATUS_H	438.0	INT	0	Parameter STATUS_H indicates a diagnostic value via the search for the point of inflection during the heating process.
STATUS_C	440.0	INT	0	Parameter STATUS_C indicates a diagnostic value via the search for the point of inflection during the cooling process.
STATUS_D	442.0	INT	0	Parameter STATUS_D indicates a diagnostic value via the controller design during the heating process.
PI_CON	444.0	STRUCT		PI controller parameters
GAIN	+0.0	REAL	0.0	PI controller gain
TI	+4.0	REAL	0.0	PI integration time [s]
	=8.0	END_STRUCT		
PID_CON	452.0	STRUCT		PID controller parameters
GAIN		REAL	0.0	PID controller gain
TI		REAL	0.0	PID integration time [s]
TD		REAL	0.0	PID derivative action time [s]
	=12.0	END_STRUCT		
PAR_SAVE	464.0	STRUCT		The PID parameters are saved in this structure.
GAIN	+0.0	REAL	0.0	Controller gain

Parameters	Address	Data type	Default	Description
TI	+4.0	REAL	0.0	Integration time [s]
TD	+8.0	REAL	0.0	Derivative action time (s)
TM_LAG	+12.0	REAL	0.0	Time lag of the D-action [s]
D_EL_SEL	+16.0	INT	0	D-action element input
PFDB_SEL	+18.0	BOOL	FALSE	P-action in feedback loop
CONZ_ON	+18.1	BOOL	FALSE	Enable control zone
CON_ZONE	+20.0	REAL	100.0	Control zone band
RATIOFAC	+24.0	REAL	1.0	Ratio of heating gain/cooling gain of the process.
	=28.0	END_STRUCT		
HEAT_OFF	492.0	BOOL	FALSE	TRUE = disable heating You must set H_OFF_EN = TRUE to activate HEAT_OFF. • MAN_ON=TRUE and MAN=0.0 when HEAT_OFF is set. • MAN_ON=FALSE when HEAT_OFF is reset.
TUN_ON	492.1	BOOL	FALSE	Enable controller optimization Set TUN=TRUE only for one cycle. TUN_ON is reset automatically after tuning.
CTUN_ON	492.2	BOOL	FALSE	Enable only cooling optimization Set CTUN =TRUE only for one cycle. CTUN _ON is reset automatically after tuning.
COM_RST	492.3	BOOL	FALSE	A initialization routine is initiated
TUN_LMN	494.0	REAL	80.0	Process excitation for controller optimization starting in cold process state is initiated by a manipulated value step by the amount of TUN_LMN.
TUN_DLMN	498.0	REAL	20.0	Process excitation for controller optimization at the operating point is initiated by a manipulated value step by the amount of TUN_DLMN.
TUN_CLMN	502.0	REAL	-10.0	Manipulated value after the start of cooling optimization.
STAT_TM	506.0	REAL	30.0	Time for steady state condition for controller optimization phase 1 or 11 The process is considered in steady state if the process value or manipulated value do not change by more than the amount of 2*STAT_D_PV or 2*STATDLMN within the time STAT_TM.
STAT_DPV	510.0	REAL	1.0	Process value delta for the steady state condition in controller optimization phase 1 or 11 See STAT_TM .
STATDLMN	514.0	REAL	4.0	Manipulated value delta for the steady state condition in controller optimization phase 1 or 11 See STAT_TM.
RATIOFAC	518.0	REAL	1.0	Ratio of heating gain/cooling gain of the process. The low output limit and "Range start of the input signal of manipulated value B of the split-range function" are adapted automatically.

Parameters	Address	Data type	Default	Description
CON_ZONE	522.0	REAL	100.0	Control zone If ER >= CON_ZONE, then LMN = LMN_HLM. If ER <= -CON_ZONE (or -CON_ZONE /RATIOFAC), then LMN = LMN_LLM.
MAN	526.0	REAL	0.0	Manual value HEAT_OFF affects the parameter if H_OFF_EN=TRUE .
MAN_ON	530.0	BOOL	TRUE	Manual/automatic mode changeover HEAT_OFF affects the parameter if H_OFF_EN=TRUE .
CONZ_ON	530.1	BOOL	FALSE	TRUE = control zone enabled
UNDO_PAR	530.2	BOOL	FALSE	Copies the controller parameters of structure PAR_SAVE to the following parameters: • cont_FM455.GAIN • cont_FM455.TI • cont_FM455.TD • cont_FM455.TM_LAG • cont_FM455.D_EL_SEL • cont_FM455.PFDB_SEL • CONZ_ON • CON_ZONE • RATIOFAC The parameters are then downloaded to FM 455. This function is not executed in automatic mode.
SAVE_PAR	530.3	BOOL	FALSE	Saves the following controller parameters to the structure PAR_SAVE: • cont_FM455.GAIN • cont_FM455.TI • cont_FM455.TD • cont_FM455.TM_LAG • cont_FM455.D_EL_SEL • cont_FM455.PFDB_SEL • CONZ_ON • CON_ZONE • RATIOFAC

Parameters	Address	Data type	Default	Description
LOAD_PID	530.4	BOOL	FALSE	Load PI/PID parameters: If PID_ON=TRUE, it copies the PID parameters of structure PID_CON to the following parameters: • cont_FM455.GAIN • cont_FM455.TI • cont_FM455.TD • cont_FM455.TM_LAG If PID_ON=FALSE, it copies the PI parameters of structure PI_CON to the following parameters: • cont_FM455.GAIN • cont_FM455.TI • cont_FM455.TD = 0.0 • cont_FM455.TM_LAG = 0.5 The parameters are then downloaded to FM455. This function is not executed in automatic mode.
BTUN_DPV	532	REAL	0.0	Process value difference for automatic cancellation of optimization.
BTUN_TM	536	REAL	300.0	Time following the manipulated value step-change for automatic cancellation of optimization.

Variables for controller optimization ZONE

Table 9-283

Parameters	Address	Data type	Default	Description
GAIN_P	540.0	REAL	0.0	Identified process gain.
TU	544.0	REAL	0.0	Identified time lag of the process.
TA	548.0	REAL	0.0	Identified recovery time of the process.
KIG	552.0	REAL	0.0	Maximum possible rate of rise of the process value during heating.
KIG_C	556.0	REAL	0.0	Maximum decay rate of the process value during cooling.
N_PTN	560.0	REAL	0.0	Order of the process. "Non-integer values" are also possible.
TM_LAG_P	564.0	REAL	0.0	Time constant of a PTN model. (practical values for N_PTN>=2 only).
T_P_INF	568.0	REAL	0.0	Time from process excitation until the point of inflection.
P_INF	572.0	REAL	0.0	Process value change from process excitation until the point of inflection.
LMN0	576.0	REAL	0.0	Manipulated value at the start of optimization Calculated in phase 1 (mean value).
PV0	580.0	REAL	0.0	Process value at the start of optimization Calculated in phase 1 (mean value).
PVDT0	584.0	REAL	0.0	Process value slew rate at the start of optimization
PVDT	588.0	REAL	0.0	Current process value slew rate
PVDT_MAX	592.0	REAL	0.0	Maximum derivative of the process value at the point of inflection (sign adapted, always > 0); is used to calculate TU and KIG.

Parameters	Address	Data type	Default	Description
NOI_PVDT	596.0	REAL	0.0	Noise action in PVDT_MAX in % The higher the noise action, the less accurate (less aggressive) the control parameters.
NOISE_PV	600.0	REAL	0.0	Absolute process value noise Difference between maximum and minimum process value in phase 1.
FIL_CYC	604.0	INT	1	Number of cycles of mean value filters 1 to 1024; The process value is determined through FIL_CYC cycles. FIL_CYC is increased from 1 to a max. of 1024 if needed.
POI_CMAX	606.0	INT	0	Maximum number of cycles after inflection point This time is used to find another (i.e. better) inflection point for measuring noise. The optimization is completed only after this time.
POI_CYCL	608.0	INT	0	Number of cycles after inflection point
NBR_CALL	610.0	INT	-1	Call number; is entered automatically after restart.

This is followed by internal parameters which may not be used by users.

Parameters STATUS_H and STATUS_C

STATUS_H indicates a diagnostic value via the search for the point of inflection during the heating process, STATUS_C beim during the cooling process.

If a diagnostic value 2xxxx is displayed, QTUN_WRN = TRUE

If a diagnostic value 3xxxx is displayed, QTUN_ERR = TRUE

STATUS_ H STATUS_C	Description	Remedy
0	Default, or no/no new controller parameters	
10000	Tuning completed + suitable controller parameters found	
2xxxx	Tuning completed + controller parameters uncertain	
2xx2x	Point of inflection not reached (only when optimization from cold process state to the operating point QTUNSTCP =TRUE)	If the controller oscillates, weaken the controller parameters, or repeat the test with a smaller manipulated value difference TUN_LMN.
2x1xx	Estimation error TU too small TU < 3*CYCLE	Reduce CYCLE and repeat the test. Special case self-contained PT1 process:: Do not repeat the test; weaken the controller parameters if need be.
2x2xx	Estimation error T_P_INF too small T_P_INF < 72.8*CYCLE	see 2x1xx
2x3xx	Estimation error TU too small and estimation error T_P_INF too small	see 2x1xx
3xxxx	Tuning canceled. No controller parameter found.	
30002	For heating optimization: Effective manipulated variable difference < 5% For cooling optimization: TUN_CLMN > -5 %	Correct manipulated value difference TUN_LMN, TUN_DLMN or TUN_CLMN.

STATUS_ H STATUS_C	Description	Remedy
30003	Incorrect manipulated variable low output limit for cooling optimization. cont_FM455.LMN_LLM > -5.0	Set zone as heating/cooling zone COOLZONE =TRUE and check cont_FM455.LMN_LLM.
30004	Effective manipulated value difference is limited by split-range limits and not by manipulated value limits.	Check limits of the manipulated values and split-range. Correct manipulated value difference TUN_LMN, TUN_DLMN or TUN_CLMN.
30005	Sampling time CYCLE deviates by more than 5% from the measured value.	Compare CYCLE with the cycle time of the cyclic interrupt level and monitor possibly existing loop scheduler. Check CPU load. Excessive load on CPU resources leads to extended sampling times that no longer match CYCLE.
30008	An attempt is being made to initiate cooling optimization. However, no heating optimization was carried out beforehand (KIG=0.0).	Start with a heating optimization session.
30009	Safety mode: cont_FM455.QLMNSAFE = TRUE	Exit the safety mode and restart optimization.
30010	No adequate temperature change after modification of heating/cooling output.	Check the signal profile of the manipulated variable and time. Check plausibility of the parameters BTUN_TM and BTUN_DPV.

See also

Output parameters ADM_ZONE (Page 7302)

Parameters STATUS_D

STATUS_D	Description
0	No controller parameters were calculated.
110	N_PTN <= 1.5 process type I fast
121	N_PTN > 1.5 process type I
200	N_PTN > 1.9 process type II (transition range)
310	N_PTN >= 2.1 process type III fast
320	N_PTN > 2.6 process type III
111, 122, 201, 311, 321	Parameters have been corrected from phase 7.

Note

STATUS_H = 0 is reset at the end of phase 1. If you cancel optimization in phase 2, STATUS_H = 0 is set. However, STATUS_D still displays the status of the last controller calculation.

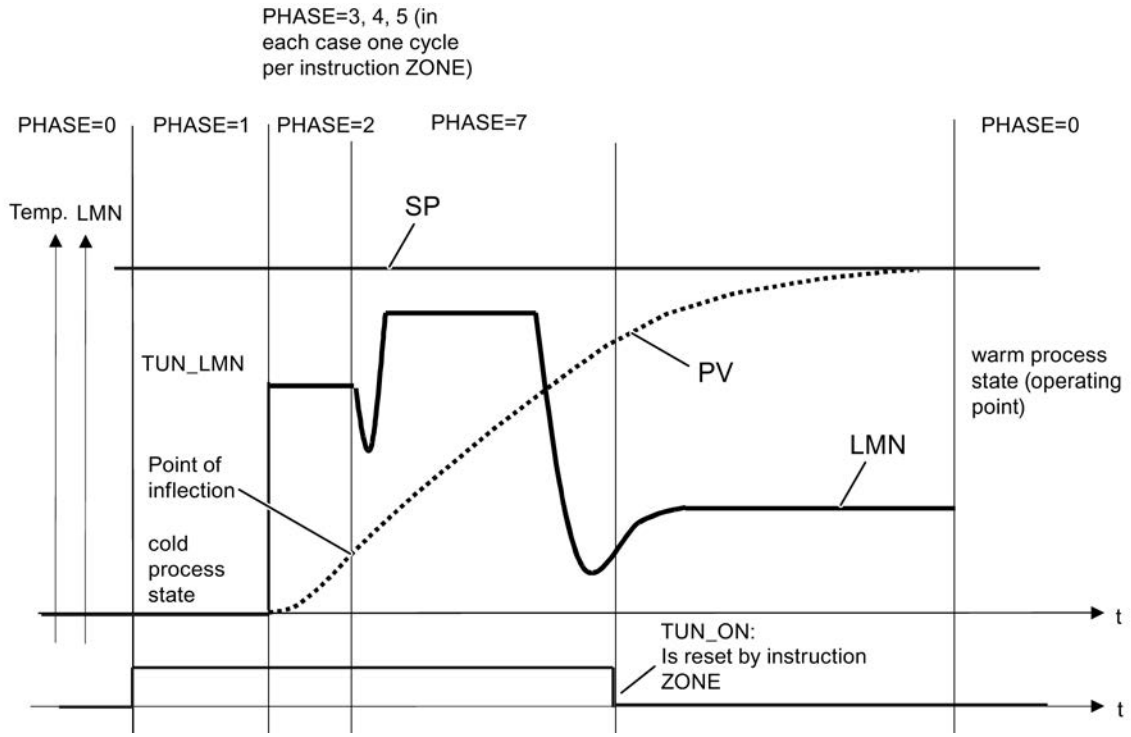
The higher the value of STATUS_D, the higher the order of the control process, the greater the TU/TA ratio and the gentler the controller parameters will be.

Parameter PHASE

PHASE	Description
0	No optimization mode; automatic or manual mode
1	Steady state test; parameter check; feed forwarding of a constant heating manipulated variable
2	Searching for point of inflection with constant manipulated value
3 (1 cycle per instruction ZONE)	Calculating the process parameters. Saving currently valid controller parameters prior to optimization
4 (1 cycle per instruction ZONE)	Controller design
5 (1 cycle per instruction ZONE)	Following up the controller to the new manipulated variable
7	Check of the process type (only for heating optimization)
10	Cooling optimization Wait until all zones zones are in automatic mode (phase > 2).
11	Cooling optimization Steady state test; parameter check; feed forwarding of a constant cooling manipulated variable
12	Cooling optimization: Searching for point of inflection with constant manipulated value
13	Cooling optimization: Calculating the heating/cooling ratio

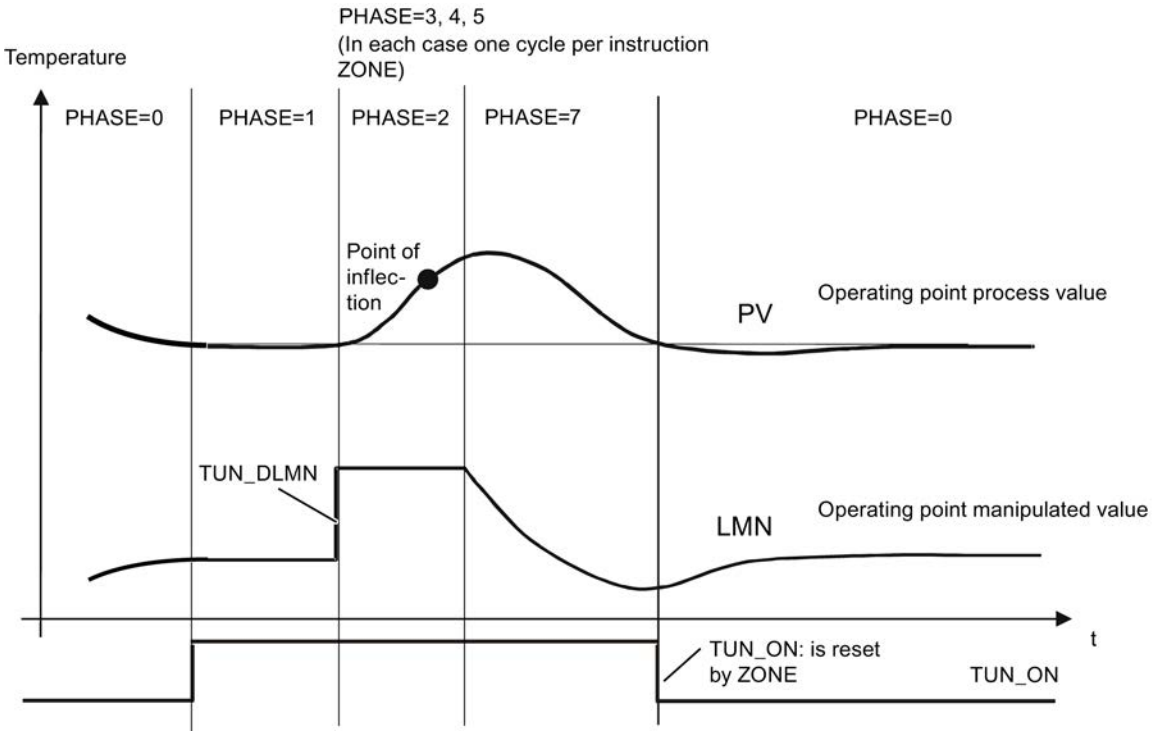
Heating Tuning Phases during First Adaption

The following figure shows the phases of self-contained heating optimization as a function of the ambient temperature and operating point:



Phases of Heating Tuning at the Operating Point

The following figure shows the phases of a self-contained heating optimization at the operating point.



After optimization was completed and the instruction has returned to Phase 0 and TUN_ON=FALSE is set, the parameters STATUS_H, STATUS_C or QTUN_WRN, QTUN_ERR indicate whether optimization was completed without errors.

DB for HMI

Description DB 101 to 116

In order to operate the FM 455 if the CPU fails, you require data blocks 101 to 104. You can establish a maximum of three connections from the FM 455 to OPs via MPI.

Operator control of the FM 455 with the OP is only possible in the STOP mode of the CPU or at a CPU failure.

Monitoring of the FM 455 with the OP is always possible.

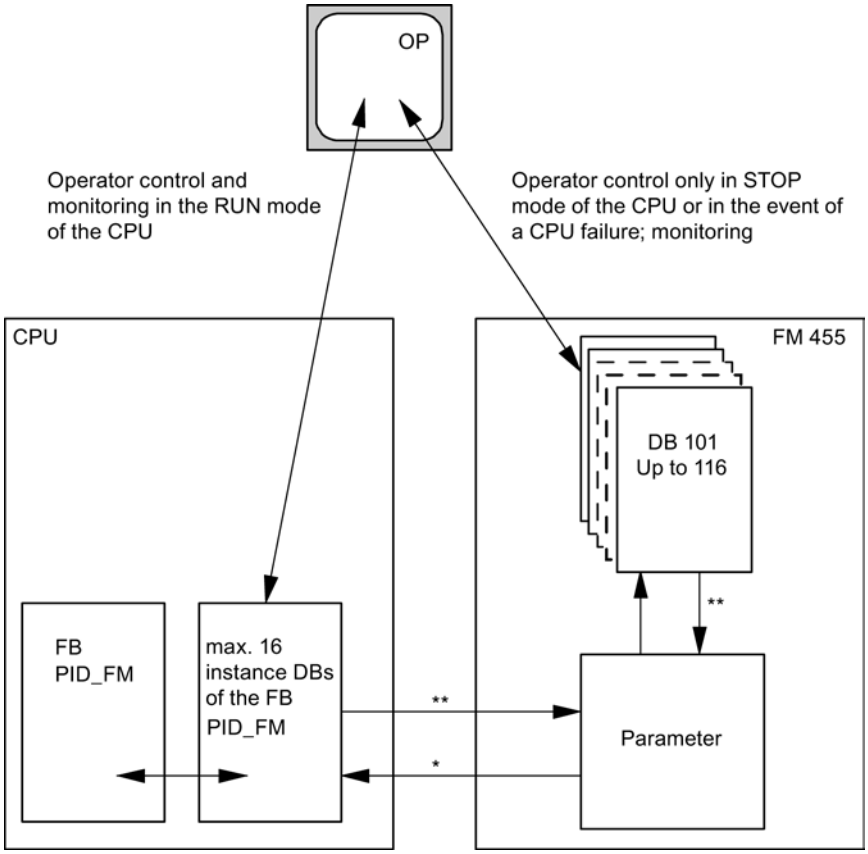
The variable interface of the FM 455 contains 16 data blocks with the block numbers 101 to 116 for the controller channels 1 to 16 (refer to the following figure).

Note

The contents of the data blocks 101 to 116 do not automatically reflect the parameter value effective at the FM 455. Parameters changed with the OP are only transferred to the FM 455 after the operating bits LOAD_PAR or LOAD_OP have been set.

If you change a parameter using OP operation without setting the corresponding operating bit, the changed parameter value is entered in the data block, but the FM 455 continues to operate internally with the unchanged old value of the parameter.

After the operating bits have been set and the parameters have been transferred to the FM 455, the operating bits LOAD_PAR and LOAD_OP are reset by the FM 455.



* controlled by the READ_VAR parameter of the instance DB

** controlled by the LOAD_OP and LOAD_PAR parameters

Input parameters DB 101 to 116

Note

The EEPROM of the module could become damaged in the case of too frequent write operations. To prevent this, after the description of the EEPROM, the module delays another description by 30 minutes.

Table 9-284

Parameters	Address	Data type	Default	Description
1) Control parameters; 2) Operating parameters				
SP_HLM1)	0.0	REAL	100.0	The setpoint is always limited to a high and a low limit. The input "Setpoint high limit" indicates the high limit. SP_HLM > SP_LLM
SP_LLM1)	4.0	REAL	0.0	The setpoint is always limited to a high and a low limit. The input "Setpoint low limit" specifies the low limit. SP_LLM < SP_HLM
H_ALM1)	8.0	REAL	100.0	You can assign parameters for four limits for monitoring the process value or the error signal. The input "High limit interrupt" indicates the highest limit. H_ALM > H_WRN
H_WRN1)	12.0	REAL	90.0	You can assign parameters for four limits for monitoring the process value or the error signal. The input “High limit warning” specifies the second to highest limit. Values from H_ALM to L_WRN are permitted.
L_WRN1)	16.0	REAL	10.0	You can assign parameters for four limits for monitoring the process value or the error signal. The input “Low limit warning” specifies the second to highest limit. Values from H_WRN to L_ALM are permitted.
L_ALM1)	20.0	REAL	0.0	You can assign parameters for four limits for monitoring the process value or the error signal. The input "Low limit alarm" specifies the lowest limit. L_ALM < L_WRN
HYS1)	24.0	REAL	1.0	To prevent flickering of the monitoring displays, a hysteresis can be configured at the "Hysteresis" input. HYS ≥ 0.0
DEADB_W1)	28.0	REAL	0.0	A dead band is applied to the error signal. The "Dead band width" input determines the size of the dead band. DEADB_W ≥ 0.0
GAIN1)	32.0	REAL	1.0	The "Proportional gain" input specifies controller amplification.
TI1)	36.0	REAL	3000.0	The "Integration time" input determines the time response of the integral action. If TI = 0, the integral action is disabled. TI = 0.0 or TI ≥ 0.5
TD1)	40.0	REAL	0.0	The "Derivative-action time" input determines the time response of the differentiator. If TD = 0, the derivative action is disabled TD = 0.0 or TD ≥ 1.0

Parameters	Address	Data type	Default	Description
TM_LAG[1)]	44.0	REAL	5.0	The algorithm of the derivative action contains a delay for which parameters can be assigned at the input "Time lag of the derivative action". TM_LAG ≥ 0.5
LMN_SAFE[1)]	48.0	REAL	0.0	For the manipulated value, a safety value can be configured at the "Safety manipulated value" input. Values from -100 to 100 % are permitted.
LMN_HLM[1)]	52.0	REAL	100.0	The manipulated value is always limited to a high and a low limit. The input "Manipulated value high limit" specifies the high limit. (Does not apply to step controllers without analog position feedback) Values from LMN_LLM to 100 % are permitted.
LMN_LLM[1)]	56.0	REAL	0.0	The manipulated value is always limited to a high and a low limit. The input "Manipulated value low limit" indicates the low limit. (Does not apply to step controllers without analog position feedback) Values from -100 to LMN_HLM are permitted.
MTR_TM[1)]	60.0	REAL	60.0	The runtime of the control valve from limit to limit is entered at the "Motor actuating time" parameter. (Only in the case of step controllers) MTR_TM ≥ 0.001
PULSE_TM[1)]	64.0	REAL	0.2	A minimum pulse duration can be configured on the "Minimum pulse time" parameter. (Only in the case of step controllers or pulse controllers)
BREAK_TM[1)]	68.0	REAL	0.2	A minimum break time can be set at the "Minimum break time" parameter. (Only in the case of step controllers or pulse controllers)
SP_RE[2)]	72.0	REAL	0.0	An external setpoint is interconnected at input "External setpoint" with the controller. Technical range of values (physic. variable)
LMN_RE[2)]	76.0	REAL	0.0	An external manipulated value is interconnected at the input "External manipulated value" with the controller. Values from -100 to 100 % are permitted.
LMNRSVAL[2)]	80.0	REAL	0.0	The Commissioning dialog provides you with access to the input "Start value of the simulated position feedback". The start value of the simulation is entered at the parameter. (only for step controllers without analog position feedback). Values from -100 to 100 % are permitted.
SAFE_ON[2)]	84.0	BOOL	FALSE	If the "Enable safety position" input is set, a safety value is applied as the manipulated value. Note: The signal processing of maninpulated values via LMNDN_OP, LMNUP_OP and LMNSOPON on a step controller is of higher priority than the safety manipulated value.
LMNTRKON[2)]	84.1	BOOL	FALSE	If input "Follow-up (LMN via AI)" is set, the manipulated value is adjusted to an analog input (AI). (Does not apply to step controllers without analog position feedback)
LMN_REON[2)]	84.2	BOOL	FALSE	If input "Enable external manipulated value" is set, the external manipulated value LMN_RE is set as the manipulated value.

Parameters	Address	Data type	Default	Description
LMNRHSRE[2)]	84.3	BOOL	FALSE	The signal “Control valve at high endstop” is interconnected at the input “High stop signal of position feedback”. LMNRHSRE = TRUE means: The control valve is at high endstop. (Only in the case of step controllers)
LMNRLSRE[2)]	84.4	BOOL	FALSE	The signal “Control valve at low endstop” is interconnected at the input “Low endstop signal of position feedback”. LMNRLSRE = TRUE means: The control valve is at low endstop. (Only in the case of step controllers)
LMNSOPON[2)]	84.5	BOOL	FALSE	If the bit is set at input "Enable manipulated value signal operation", the LMNUP_OP and LMNDN_OP signals are set as the manipulated value signals. (Only in the case of step controllers)
LMNUP_OP[2)]	84.6	BOOL	FALSE	If LMNSOPON is set, the value at input "Manipulated value signal up operation" is set as the manipulated value signal. (Only in the case of step controllers)
LMNDN_OP[2)]	84.7	BOOL	FALSE	If LMNSOPON is set, the value at input "Manipulated value signal down operation" is set as the manipulated value signal. (Only in the case of step controllers)
MONERSEL[1)]	85.0	BOOL	FALSE	The controller has a limit indicator that can be applied either for the process value or the error signal. If the input "Monitoring: Process value = 0, control error = 1" is set, the control error is monitored.
LMNRS_ON[2)]	85.1	BOOL	FALSE	You can simulate position feedback if this is not available. The function is enabled by setting the "Enable simulation of position feedback" input CAUTION: Offset between the simulation value and real position feedback increases over the time. (only for step controllers without analog position feedback).
FUZID_ON[2)]	85.2	BOOL	FALSE	The "Enable fuzzy identification" input is used to enable identification of the fuzzy algorithm.
SPINT_EN[2)]	85.3	BOOL	FALSE	The input "Operator input: External = 0, Internal = 1" determines the input that is transferred as a setpoint to the module. SPINT_EN = TRUE: SP_INT is transferred. SPINT_EN = FALSE: SP_RE is transferred.
P_SEL[1)]	85.4	BOOL	TRUE	In the PID algorithm, the PID actions can be enabled and disabled separately. The proportional action is enabled if the "Enable P-action" input is set
PFDB_SEL[1)]	85.5	BOOL	FALSE	In the PID algorithm, the P and D actions can be positioned in the feedback. If input "Enable P-action in feedback" is set, the proportional action is located in the feedback loop.
D_EL_SEL[1)]	86.0	INT	0	The D-action element in the PID algorithm can be positioned at a separate input. This is selected at the input "D-action element input". • 0: Error signal • 1 to 16:: Analog input 1 to 16 • 17: Negative process value

Output parameters DB 101 to 116

Table 9-285

Parameters	Address	Data type	Default	Description
SP	94.0	REAL	0.0	The setpoint that is currently in effect is available at the "Setpoint" output.
PV	98.0	REAL	0.0	The process value that becomes effective is issued on the "Process value" output.
ER	102.0	REAL	0.0	The control deviation that becomes effective is issued on the "Control deviation" output.
DISV	106.0	REAL	0.0	The interference that becomes effective is issued on the "Interference" output. Values from -100 to 100 % are permitted.
LMN	110.0	REAL	0.0	The manipulated value that becomes effective is issued on the "Manipulated value" output. Step controllers without analog position feedback output the unlimited P- + D-action at the LMN parameter. Values from -100 to 100 % are permitted.
LMN_A	114.0	REAL	0.0	At the output "Manipulated value A of the split range function / position feedback", continuous controllers display the manipulated value A and step controllers with analog position feedback display the position feedback. Step controllers without analog position feedback display the simulated position feedback. Values from -100 to 100 % are permitted.
LMN_B	118.0	REAL	0.0	Continuous controllers display the manipulated value B of the split-range function at the output “Manipulated value B of split-range function”. Values from -100 to 100 % are permitted.
QH_ALM	122.0	BOOL	FALSE	The process value or the controlled variable is monitored for four limits. Violation of H_ALM limit is reported at the output "High limit alarm triggered".
QH_WRN	122.1	BOOL	FALSE	The process value or the controlled variable is monitored for four limits. Violation of H_WRN limit is reported at the output "High limit warning triggered".
QL_WRN	122.2	BOOL	FALSE	The process value or the controlled variable is monitored for four limits. Violation of the L_WRN limit is reported at output "Low limit warning triggered".
QL_ALM	122.3	BOOL	FALSE	The process value or the controlled variable is monitored for four limits. Violation of the L_ALM limit is reported at output "Low limit alarm triggered".
QLMN_HLM	122.4	BOOL	FALSE	The manipulated value is always limited to a high and a low limit. Violation of the high limit is reported at output "High limit of manipulated value triggered". (Does not apply to step controllers without analog position feedback)
QLMN_LLM	122.5	BOOL	FALSE	The manipulated value is always limited to a high and a low limit. Violation of the low limit is reported at output "Low limit of manipulated value triggered". (Does not apply to step controllers without analog position feedback)
QSPINTON	122.6	BOOL	FALSE	The output "Internal setpoint on" indicates that SP_INT was transferred to the module.
QPARA_F	123.0	BOOL	FALSE	The module checks the validity of the parameters. A parameter assignment error is displayed at the "Parameter assignment error" output. These parameter assignment errors can also be read out in the module's "Online & Diagnostics" dialog.
QCH_F	123.1	BOOL	FALSE	The "Channel error" output is set if the controller channel cannot provide valid results. A channel error (for example, wire break) is also set if QPARA_F = 1 or QMOD_F = 1. If QCH_F = TRUE, then the precise error information in the diagnostic data record DS1 of the module is read out.

Parameters	Address	Data type	Default	Description
QUPRLM	123.2	BOOL	FALSE	The setpoint is limited to a positive and negative slew rate. If the output "Limit of positive setpoint slew rate triggered" is set, the positive setpoint slew rate is limited.
QDNRLM	123.3	BOOL	FALSE	The setpoint is limited to a positive and negative slew rate. If the output "Limit of negative setpoint slew rate triggered" is set, the setpoint fall is limited.
QSP_HLM	123.4	BOOL	FALSE	The setpoint is always limited to a high and a low limit. The output "High limit of setpoint value triggered" indicates that the high limit has been exceeded.
QSP_LLM	123.5	BOOL	FALSE	The setpoint is always limited to a high and a low limit. Violation of the low limit is reported at output "Setpoint low limit triggered".
QSPOPON	123.6	BOOL	FALSE	The output "Setpoint operation on" shows if the setpoint is being operated via the "Commissioning" dialog. If the bit is set, the value SP_OP is applied as setpoint.
QLMNSAFE	123.7	BOOL	FALSE	If the output "Safety mode" is set, the safety manipulated value is output as the manipulated value.
QLMNOPON	124.0	BOOL	FALSE	The output "Manipulated value operation on" shows if the manipulated value is being operated via the "Commissioning" dialog. If the bit is set, the value LMN_OP is applied as manipulated value.
QLMNTRK	124.1	BOOL	FALSE	The output "Follow-up mode" indicates if the manipulated value is being adjusted via an analog input.
QLMN_RE	124.2	BOOL	FALSE	The output "Manual = 1; Automatic = 0" indicates whether or not the manipulated value is set on the external manipulated value LMN_RE (manual = 1).
QLMNR_HS	124.3	BOOL	FALSE	The output "High endstop signal of position feedback" indicates if the control valve is at its high endstop. QLMNR_HS = TRUE means: The control valve is at high endstop. (For step controllers only)
QLMNR_LS	124.4	BOOL	FALSE	The output "Low endstop signal of position feedback" indicates whether the control valve is at its low endstop. QLMNR_LS = TRUE means: The control valve is at its low endstop. (For step controllers only)
QLMNR_ON	124.5	BOOL	FALSE	Output "Position feedback enabled" indicates the set operating mode "Step controller with position feedback" or "Step controller without position feedback".
QFUZZY	124.6	BOOL	FALSE	If the output "PID algorithm = 0; fuzzy = 1", the controller is working with the fuzzy algorithm.
QSPLEPV	124.7	BOOL	FALSE	The output "Display of FUZZY controller: Setpoint < process value" is set when the fuzzy controller is switched on, if the setpoint is less than the effective process value.
QSPR	125.0	BOOL	FALSE	If the output "Split-range operation" is set, the continuous controller is operating in split-range mode.
QLMNUP	125.1	BOOL	FALSE	This is the "Manipulated value signal up" output. (Only in the case of step controllers or pulse controllers)
QLMNDN	125.2	BOOL	FALSE	This is the "Manipulated value signal down" output. (Only in the case of step controllers or pulse controllers)
QBACKUP	125.4	BOOL	FALSE	• 0= No backup state (CPU in RUN) • 1= Backup state (CPU in STOP or has failed)
QID	125.5	BOOL	FALSE	QID = TRUE indicates that an identification is running (not that it is enabled). On completion of the identification, the identification result can be read out from the IDSTATUS parameter of the instruction CH_DIAG_455.

Parameters	Address	Data type	Default	Description
QMAN_FC	125.6	BOOL	FALSE	The controller is a master controller that is adjusted by the manual mode of a slave controller to its process value or its I-action is stopped, because the setpoint or manipulated value of the slave controller lies in the limit.
RET_VALU	126.0	WORD	0	RET_VALU contains the return value RET_VAL of the RD_REC and WR_REC instructions. RET_VALU can be evaluated when QMOD_F reports an error.

See also

RD_REC: Read data record from I/O (Page 6491)

WR_REC: Write data record to I/O (Page 6496)

In/out parameters DB 101 to 116

Table 9-286

Parameters	Address	Data type	Default	Description
1) Operating parameters				
SP_INT	128.0	REAL	0.0	The in/out parameter "Internal setpoint" is used to specify a setpoint by means of operating and monitoring functions.
SP_OP[1)]	132.0	REAL	0.0	The "Commissioning" dialog can access the in/out parameter "Setpoint operation". If the SP_OP_ON bit is set, the "Setpoint operation" value is applied as the setpoint.
LMN_OP[1)]	136.0	REAL	0.0	The "Commissioning" dialog can access the in/out parameter "Manipulated value operation". If the LMNOP_ON bit is set, the "Manipulated value operation" value is applied as the manipulated value. Values from -100 to 100 % are permitted.
SP_OP_ON [1)]	140.0	BOOL	FALSE	The "Commissioning" dialog can access the in/out parameter "Enable setpoint operation". If the bit is set, the value SP_OP is applied as setpoint.
LMNOP_ON [1)]	140.1	BOOL	FALSE	The "Commissioning" dialog can access the in/out parameter "Enable manipulated value operation". If the bit is set, the value LMN_OP is applied as manipulated value.
LOAD_PAR	140.2	BOOL	FALSE	If the in/out parameter "Load control parameters to FM 455" is set, the control parameters are loaded into the module and the in/out parameter is reset.
LOAD_OP	140.3	BOOL	FALSE	If the in/out parameter "Load operating parameters to FM 455" is set, the operating parameters are loaded into the module and the in/out parameter is reset.

Note

The EEPROM of the module could become damaged by too frequent write operations. To prevent this, the module delays another write process by 30 minutes after writing of the EEPROM.

300C functions

ANALOG: Position with analog output

Description

You use the ANALOG instruction to control the positioning functions of the user program.

A fixed assigned analog output controls the power section with a voltage (voltage signal) of ±10 V or with a current (current signal) of ±20 mA.

- After the acceleration phase (RAM_UP), the drive approaches the destination at the speed ($V_{setpoint}$).
- At the braking point that is calculated by the CPU, the deceleration (RAMP_DN) to the changeover point is initialized.
- Once the changeover point has been reached, the run will continue at creep speed (V_{creep}).
- The drive is switched off at the cut-off point.
- The changeover point and the cut-off point are determined for every target that will be approached in the values you have specified for the changeover difference and cut-off difference parameters. The changeover difference and cut-off difference can be determined differently for forward motion (in plus direction) and for reverse motion (in minus direction).
- Traverse is complete (WORKING = FALSE) when the cut-off point is reached. A new traverse can then be started after this time.
- The specified target has been reached (POS_RCD = TRUE) when the actual position value has reached the target area. If the actual position value again deviates from the target area without a new traverse having been started, the signal "position reached" is not reset.

If the change-over difference is less than the cut-off difference, the drive is slowed linearly to speed setpoint 0 starting at the braking point.

Basic parameters

The parameters of the instruction that are identical for all modes are described here. The parameters specific to the operating mode are described for the individual operating modes.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter

The following table shows the parameters of the instruction ANALOG:

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
LADDR	Input	WORD	0	CPU-specific	W#16#0310	The submodule I/O address, which you have specified during hardware configuration. If the I and O addresses are not equal, the lesser of the two addresses must be specified.
CHANNEL	Input	INT	2	0	0	Channel number
STOP	Input	BOOL	4.4	TRUE/FALSE	FALSE	Stop traverse Use STOP = TRUE to stop/ interrupt the traverse prematurely.
ERR_A	Input	BOOL	4.5	TRUE/FALSE	FALSE	Group acknowledgement of external error External errors are acknowledged with ERR_A = TRUE.
SPEED	Input	DINT	12	Creep speed to 1 000 000 pulses/ s No higher than the maximum speed that will be assigned in the parameter	1000	The axis will be accelerated to the speed "$V_{setpoint}$". It is not possible to change the speed during traverse.
WORKING	Output	BOOL	16.0	TRUE/FALSE	FALSE	Traverse in process
ACT_POS	Output	DINT	18	$-5x10^8$ to $+5x10^8$ pulses	0	Actual position value
MODE_OUT	Output	INT	22	0, 1, 3, 4, 5	0	Active/set operating mode
ERR	Output	WORD	24	Each bit 0 or 1	0	External error: Bit 2: Zero mark monitoring Bit 11: Traverse range monitoring (always 1) Bit 12: Work range monitoring Bit 13: Process value monitoring Bit 14: Destination run-up monitoring Bit 15: Destination area monitoring Remaining bits reserved

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
ST_ENBLD	Output	BOOL	26.0	TRUE/FALSE	TRUE	The CPU sets start enable if all the following conditions apply: • STOP is not active (STOP = FALSE) • No external error pending (ERR = 0) • Drive enable is set (DRV_EN = TRUE) • Positioning not in process (WORKING = FALSE)
ERROR	Output	BOOL	26.1	TRUE/FALSE	FALSE	Error when starting / resuming a traverse
STATUS	Output	WORD	28.0	W#16#0000 to W#16#FFFF	W#16#0000	Error number

Parameters not assigned to the block (static local data):

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
ACCEL	STATIC	DINT	30	1 to 100 000 pulses/s^2	100	Acceleration Change during traverse not possible.
DECEL	STATIC	DINT	34	1 to 100 000 pulses/s^2	100	Deceleration Change during traverse not possible.
CHGDIFF_P	STATIC	DINT	38	0 to $+10^8$ pulses	1000	Changeover difference plus: "Changeover difference plus" defines the changeover point from which the drive continues forward traverse in creep speed.
CUTOFF-DIFF_P	STATIC	DINT	42	0 to $+10^8$ pulses	100	Cut-off difference plus: "Cut-off difference plus" defines the cut-off point at which the drive is shut off from creep speed in the forward direction.
CHGDIFF_M	STATIC	DINT	46	0 to $+10^8$ pulses	1000	Changeover difference minus: "Changeover difference minus" defines the changeover point from which the drive continues in reverse motion at creep speed.

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
CUTOFF-DIFF_M	STATIC	DINT	50	0 to $+10^8$ pulses	100	Cut-off difference minus: "Cut-off difference minus" defines the point at which the drive is switched out of creep speed in the reverse direction.
PARA	STATIC	BOOL	54.0	TRUE/FALSE	FALSE	Parameters have been assigned to the axis
DIR	STATIC	BOOL	54.1	TRUE/FALSE	FALSE	Actual/last direction of movement FALSE = forward (in plus direction) TRUE = reverse (in minus direction)
CUTOFF	STATIC	BOOL	54.2	TRUE/FALSE	FALSE	Drive in cut-off area (from the cut-off point to the start of the next run)
CHGOVER	STATIC	BOOL	54.3	TRUE/FALSE	FALSE	Drive in changeover range (from reaching creep speed to start of next traverse)
RAMP_DN	STATIC	BOOL	54.4	TRUE/FALSE	FALSE	The drive is decelerated (from braking point to changeover point)
RAMP_UP	STATIC	BOOL	54.5	TRUE/FALSE	FALSE	The drive is accelerated (from start until it reaches the speed SPEED ($V_{setpoint}$))
DIST_TO_GO	STATIC	DINT	56	$-5x10^8$ to $+5x10^8$ pulses	0	Actual residual distance
LAST_TRG	STATIC	DINT	60	$-5x10^8$ to $+5x10^8$ pulses	0	Last/current target • Absolute incremental mode: At start of traverse LST_TRG = current absolute target (TARGET). • Relative incremental mode: At start of traverse LST_TRG = LAST_TRG is the specified +/- distance of the previous traverse (TARGET).

Parameters for "jog" mode

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
DRV_EN	Input	BOOL	4.0	TRUE/FALSE	FALSE	Drive enable
DIR_P	Input	BOOL	4.2	TRUE/FALSE	FALSE	Jog in plus direction (positive edge)
DIR_M	Input	BOOL	4.3	TRUE/FALSE	FALSE	Jog in minus direction (positive edge)
MODE_IN	Input	INT	6	0, 1, 3, 4, 5	1	Operating mode, 1 = jog
WORKING	Output	BOOL	16.0	TRUE/FALSE	FALSE	Traverse in process
ACT_POS	Output	DINT	18	$-5x10^8$ to $+5x10^8$ pulses	0	Actual position value
MODE_OUT	Output	INT	22	0, 1, 3, 4, 5	0	Active/set operating mode

Parameters for "search for reference" mode

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
DRV_EN	Input	BOOL	4.0	TRUE/FALSE	FALSE	Drive enable
DIR_P	Input	BOOL	4.2	TRUE/FALSE	FALSE	Search for reference in plus direction (positive edge)
DIR_M	Input	BOOL	4.3	TRUE/FALSE	FALSE	Search for reference in minus direction (positive edge)
MODE_IN	Input	INT	6	0, 1, 3, 4, 5	1	Operating mode, 3 = "search for reference"
WORKING	Output	BOOL	16.0	TRUE/FALSE	FALSE	Traverse in process
SYNC	Output	BOOL	16.3	TRUE/FALSE	FALSE	SYNC = TRUE: Axis is synchronized
ACT_POS	Output	DINT	18	$-5x10^8$ to $+5x10^8$ pulses	0	Actual position value
MODE_OUT	Output	INT	22	0, 1, 3, 4, 5	0	Active/set operating mode

Parameters for "relative incremental mode"

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
DRV_EN	Input	BOOL	4.0	TRUE/FALSE	FALSE	Drive enable
DIR_P	Input	BOOL	4.2	TRUE/FALSE	FALSE	Traverse in plus direction (positive edge)
DIR_M	Input	BOOL	4.3	TRUE/FALSE	FALSE	Traverse in minus direction (positive edge)

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
MODE_IN	Input	INT	6	0, 1, 3, 4, 5	1	Operating mode, 4 = relative incremental mode
TARGET	Input	DINT	8	0 to 10^9 pulses	1000	Distance in pulses (only positive values allowed)
WORKING	Output	BOOL	16.0	TRUE/FALSE	FALSE	Traverse in process
POS_RCD	Output	BOOL	16.1	TRUE/FALSE	FALSE	Position reached
ACT_POS	Output	DINT	18	$-5x10^8$ to $+5x10^8$ pulses	0	Actual position value
MODE_OUT	Output	INT	22	0, 1, 3, 4, 5	0	Active/set operating mode

Parameters for "absolute incremental mode"

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
DRV_EN	Input	BOOL	4.0	TRUE/FALSE	FALSE	Drive enable
START	Input	BOOL	4.1	TRUE/FALSE	FALSE	Start traverse (positive edge)
DIR_P	Input	BOOL	4.2	TRUE/FALSE	FALSE	Traverse in plus direction (positive edge)
DIR_M	Input	BOOL	4.3	TRUE/FALSE	FALSE	Traverse in minus direction (positive edge)
MODE_IN	Input	INT	6	0, 1, 3, 4, 5	1	Operating mode, 5 = absolute incremental mode
TARGET	Input	DINT	8	Linear axis : $-5x10^8$ to $+5x10^8$ Rotary axis: 0 to rotary axis end -1	1000	Target in pulses
WORKING	Output	BOOL	16.0	TRUE/FALSE	FALSE	Traverse in process
POS_RCD	Output	BOOL	16.1	TRUE/FALSE	FALSE	Position reached
ACT_POS	Output	DINT	18	$-5x10^8$ to $+5x10^8$ pulses	0	Actual position value
MODE_OUT	Output	INT	22	0, 1, 3, 4, 5	0	Active/set operating mode

Parameters for the job "set reference point"

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
SYNC	Output	BOOL	16.3	TRUE/FALSE	FALSE	Axis is synchronized

Parameters not assigned to the block (static local data):

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
JOB_REQ	STATIC	BOOL	76.0	TRUE/FALSE	FALSE	Job trigger (positive edge)
JOB_DONE	STATIC	BOOL	76.1	TRUE/FALSE	TRUE	New job can be started
JOB_ERR	STATIC	BOOL	76.2	TRUE/FALSE	FALSE	Faulty job
JOB_ID	STATIC	INT	78	1, 2	0	Job, 1 = "set reference point"
JOB_STAT	STATIC	WORD	80	W#16#0000 to W#16#FFFF	W#16#0000	Job error number
JOB_VAL	STATIC	DINT	82	$5x10^8$ to $+5x10^8$ pulses	0	Job parameter for reference point coordinates

Parameters for the job "delete residual distance"

Parameters not assigned to the block (static local data):

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
JOB_REQ	STATIC	BOOL	76.0	TRUE/FALSE	FALSE	Job trigger (positive edge)
JOB_DONE	STATIC	BOOL	76.1	TRUE/FALSE	TRUE	New job can be started
JOB_ERR	STATIC	BOOL	76.2	TRUE/FALSE	FALSE	Faulty job
JOB_ID	STATIC	INT	78	1, 2	0	Job, 2 = "delete residual distance"
JOB_STAT	STATIC	WORD	80	W#16#0000 to W#16#FFFF	W#16#0000	Job error number
JOB_VAL	STATIC	DINT	82	-	0	Any setting

Parameters for the "length measurement" function

This operation is started at the positive edge on the digital input. There are no specific input parameters.

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
MSR_DONE	OUT	BOOL	16.2	TRUE/FALSE	FALSE	Length measurement complete

Parameters not assigned to the block (static local data):

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
BEG_VAL	STATIC	DINT	64	$-5x10^8$ to $+5x10^8$ pulses	0	Actual position value start of length measurement
END_VAL	STATIC	DINT	68	$-5x10^8$ to $+5x10^8$ pulses	0	Actual position value length measurement end
LEN_VAL	STATIC	DINT	72	0 to 10^9 pulses	0	Measured length

Parameter ERROR and STATUS

Operating mode error (ERROR = TRUE)

The output parameter ERROR is set to TRUE if an error is detected. The parameter STATUS shows the cause of the error.

Event class error code	Description
W#16#2002	Incorrect instruction, use ANALOG
W#16#2004	Incorrect channel number (CHANNEL). Set channel number "0".
W#16#3001	Traverse job rejected because of job error in the same call. Correct the respective JOB parameters.
W#16#3002	A change of Output _IN is not permitted while the drive is still running. Wait for the end of the current positioning.
W#16#3003	Unknown operating mode (MODE_IN). Permitted are 1 (jog), 3 (search for reference), 4 (relative incremental mode) and 5 (absolute incremental mode).
W#16#3004	Only one start request is allowed at a time. Valid start requests are DIR_P, DIR_M or START.
W#16#3005	START is only permitted in operating mode "absolute incremental mode”. Start the traverse with DIR_P or DIR_M.
W#16#3006	DIR_P and DIR_M are not permitted for linear axis and "absolute incremental mode". Start the traverse with START.
W#16#3007	Axis not synchronized. "Absolute incremental mode" is only possible with a synchronized axis.
W#16#3008	Leave work area. Return traverse to work area is only allowed in jog mode.
W#16#3101	Start is not enabled because parameters have not been assigned for the axis. Assign parameters for the "positioning" submodule via HW Config.
W#16#3102	Start not enabled because drive enable is not set Set the "drive enable" on the instruction (DRV_EN=TRUE).
W#16#3103	Start not enabled because STOP is set. Delete STOP on the instruction (STOP=FALSE)
W#16#3104	Start is not enabled because the axis is currently positioning (WORKING=TRUE). Wait for the end of the current positioning.
W#16#3105	Start not enabled because at least one error that is pending has not been acknowledged. First, eliminate and clear all external errors and then restart traverse.
W#16#3202	Wrong SPEED default. The speed default is out of the permitted creep speed range of up to 1000000 pulses/s, though not higher than the parameter-assigned maximum speed.

Event class error code	Description
W#16#3203	The acceleration default in ACCEL is outside the permitted range of 1 to 100,000 pulses/s^2.
W#16#3204	The deceleration default in DECEL is outside the permitted range of 1 to 100,000 pulses/s^2.
W#16#3206	The speed default in SPEED must be greater than or equal to the assigned referencing frequency parameter.
W#16#3301	Changeover/cut-off difference is too high. Set a maximum changeover/cut-off difference of 10^8.
W#16#3304	Cut-off difference too low. The cut-off difference must be at least half the value of the target range.
W#16#3305	Changeover difference too low. The changeover difference must be at least half the value of the target range.
W#16#3401	Target default outside of work range. For a linear axis and absolute incremental mode the target default must be within the range of the software limit switches (inclusive).
W#16#3402	Wrong target default. For a rotary axis the target default must be greater than 0 and less than the rotary axis end value.
W#16#3403	Wrong distance specification. The traverse distance for relative incremental mode must be positive.
W#16#3404	Wrong distance specification. The resulting absolute target coordinate must be greater than $-5x10^8$.
W#16#3405	Wrong distance specification. The resulting absolute target coordinate must be greater than $-5x10^8$.
W#16#3406	Wrong distance specification. The resulting absolute target coordinate must be within the work range (+/- half of the target range).
W#16#3501	Travel range too great. Target coordinate + actual residual distance must be greater than or equal to $-5x10^8$.
W#16#3502	Travel range too great. Target coordinate + actual residual distance must be less than or equal to $5x10^8$.
W#16#3503	Traverse distance insufficient. Traverse distance in plus direction must be greater than the specified cut-off difference for direction plus.
W#16#3504	Traverse distance insufficient. The traverse distance in minus direction must be greater than the specified cut-off difference for the minus direction
W#16#3505	Traverse distance insufficient or the limit switch is already overtraveled in plus direction. The last approachable target in plus direction (work range or traverse range limit) is too close to the current position.
W#16#3506	Traverse distance insufficient, or the limit switch is already overtraveled in the minus direction. The last approachable target in the minus direction (work range or traverse range limit) is too close to the current position.

Job error (JOB_ERR = TRUE)

The output parameter JOB_ERR is set to TRUE if an error is detected. The parameter JOB_STAT shows the cause of the error.

Event class error code	Description
W#16#4001	Parameters not assigned for axis. Assign parameters for the "positioning" submodule via HW Config.
W#16#4002	Job not possible because positioning is in progress. Wait until WORKING = FALSE, then repeat the job.
W#16#4004	Unknown job. Check the job ID and then repeat the job.
W#16#4101	For a linear axis the reference point coordinate should not be outside of the work area limits.
W#16#4102	For a linear axis the specified reference point coordinate + actual residual distance must be greater than or equal to $-5x10^8$.
W#16#4103	For a linear axis the specified reference point coordinate + actual residual distance must be greater than or equal to $5x10^8$.
W#16#4104	For a linear axis the specified reference point coordinate + actual difference to the starting point of traverse must be greater than or equal to $-5x10^8$.
W#16#4105	For a linear axis the specified reference point coordinate + actual difference to the starting point of traverse must be less than or equal to $5x10^8$.
W#16#4106	For a rotary axis the reference point coordinate should not be lower than 0 and greater than or equal to the rotary axis end.

External error (ERR)

The technology monitors traverse, traverse range, and the connected peripheral devices. The prerequisite is that you have switched on monitoring in the "Drive", "Axis" and "Encoder" parameter screen forms beforehand.

An external fault is reported when the monitoring unit is tripped. External errors can occur at anytime regardless of the started functions. You must always acknowledge external errors with ERR_A = TRUE.

External errors are indicated by a set bit in parameter ERR (WORD).

Monitoring	Error code	Bit in ERR-WORD
Missing pulse (zero mark)	W#16#0004	2
Traverse range	W#16#0800	11
Work range	W#16#1000	12
Process value	W#16#2000	13
Destination run-up	W#16#4000	14
Target range	W#16#8000	15

System error

A system error is indicated with BIE (binary result) = FALSE. A system error is triggered by errors while reading/writing the instance DB or by multiple calls of the instruction.

DIGITAL: Position with digital output

Description

You use the DIGITAL instruction to control the positioning functions of the user program.

Four 24 V digital outputs are permanently assigned to the drive and control the power stage (you can find additional information in the documentation for the hardware used). Depending on the trigger mode, the digital outputs control the direction and speed levels (rapid/creep speed).

The distance is measured via an asymmetrical 24 V incremental transducer with two out-of-phase signals at 90 degrees.

- First, the target is approached with the speed (V_{rapid}).
- At the changeover point, the system switches to creep speed (V_{creep}).
- The drive is switched off at the cut-off point.
- You can set the changeover point and cut-off point with the parameter values CHGDIFF_P/M and CUTOFFDIFF_P/M. The changeover difference and cut-off difference can be defined differently for forward motion (in plus direction) and for reverse motion (in minus direction).
- Traverse is complete (WORKING = FALSE) when the cut-off point is reached. A new traverse can then be started after this time.
- The specified target has been reached (POS_RCD = TRUE) when the actual position value has reached the target area. If the actual position value again deviates from the target area without a new traverse having been started, the signal "position reached" is not reset.

Basic parameters:

The parameters of the instruction that are identical for all modes are described here. The parameters specific to the operating mode are described for the individual operating modes.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
LADDR	Input	WORD	0	CPU-specific	W#16#0310	The submodule I/O address, which you have specified during hardware configuration. If the I and O addresses are not equal, the lesser of the two addresses must be specified.
CHANNEL	Input	INT	2	0	0	Channel number
STOP	Input	BOOL	4.4	TRUE/FALSE	FALSE	Stop traverse Use STOP = TRUE to stop/interrupt the traverse prematurely.

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
ERR_A	Input	BOOL	4.5	TRUE/FALSE	FALSE	Group acknowledgement of external error External errors are acknowledged with ERR_A = TRUE.
SPEED	Input	BOOL	12.0	TRUE/FALSE	FALSE	Two speed stages for rapid traverse/creep speed TRUE = rapid traverse FALSE= creep speed
WORKING	Output	BOOL	14.0	TRUE/FALSE	FALSE	Traverse in process
ACT_POS	Output	DINT	16	$-5x10^8$ to $+5x10^8$ pulses	0	Actual position value
MODE_OUT	Output	INT	20	0, 1, 3, 4, 5	0	Active/set operating mode
ERR	Output	WORD	22	Each bit 0 or 1	0	External error • Bit2: Zero mark monitoring • Bit11: Traverse range monitoring (always 1) • Bit12: Work range monitoring • Bit13: Process value monitoring • Bit14: Target area monitoring • Bit15: Target area monitoring Remaining bits reserved
ST_ENBLD	Output	BOOL	24.0	TRUE/FALSE	TRUE	The CPU sets start enable if all the following conditions apply: • STOP is not active (STOP = FALSE) • No external error pending (ERR = 0) • Drive enable is set (DRV_EN = TRUE) • Positioning not in process (WORKING = FALSE)
ERROR	Output	BOOL	24.1	TRUE/FALSE	FALSE	Error when starting /resuming a traverse
STATUS	Output	WORD	26.0	W#16#0000 to W#16#FFFF	W#16#0000	Error number

Parameters not assigned to the block (static local data):

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
CHGDIFF_P	STATIC	DINT	28	0 to $+10^8$pulses	1000	Changeover difference plus: "Changeover difference plus" defines the point at which the drive continues its forward traverse with creep speed.
CUTOFF-DIFF_P	STATIC	DINT	32	0 to $+10^8$ pulses	100	Cut-off difference plus: "Cut-off difference plus" defines the cut-off point at which the drive is shut off from creep speed in the forward direction.
CHGDIFF_M	STATIC	DINT	36	0 to $+10^8$ pulses	1000	Changeover difference minus: "Changeover difference minus" defines the point at which the drive changes over from rapid traverse to creep speed in the reverse direction.
CUTOFF-DIFF_M	STATIC	DINT	40	0 to $+10^8$ pulses	100	Cut-off difference minus: "Cut-off difference minus" defines the point at which the drive is switched out of creep speed in the reverse direction.
PARA	STATIC	BOOL	44.0	TRUE/FALSE	FALSE	Parameters have been assigned to the axis
DIR	STATIC	BOOL	44.1	TRUE/FALSE	FALSE	Actual/last direction of movement FALSE = forward (in plus direction) TRUE = reverse (in minus direction)
CUTOFF	STATIC	BOOL	44.2	TRUE/FALSE	FALSE	Drive in cut-off area (from the cut-off point to the start of the next run)
CHGOVER	STATIC	BOOL	44.3	TRUE/FALSE	FALSE	Drive in changeover range (from reaching creep speed to start of the next traverse)

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
DIST_TO_GO	STATIC	DINT	46	$-5x10^8$ to $+5x10^8$ pulses	0	Actual residual distance
LAST_TRG	STATIC	DINT	50	$-5x10^8$ to $+5x10^8$ pulses	0	Last/current target • Absolute incremental mode: At start of traverse LST_TRG = current absolute target (TARGET). • Relative incremental mode: At start of traverse LST_TRG = LAST_TRG is the specified +/- distance of the previous traverse (TARGET).

Parameters for "jog" mode

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
DRV_EN	Input	BOOL	4.0	TRUE/FALSE	FALSE	Drive enable
DIR_P	Input	BOOL	4.2	TRUE/FALSE	FALSE	Jog in plus direction (positive edge)
DIR_M	Input	BOOL	4.3	TRUE/FALSE	FALSE	Jog in minus direction (positive edge)
MODE_IN	Input	INT	6	0, 1, 3, 4, 5	1	Operating mode, 1 = jog
WORKING	Output	BOOL	14.0	TRUE/FALSE	FALSE	Traverse in process
ACT_POS	Output	DINT	16	$-5x10^8$ to $+5x10^8$ pulses	0	Actual position value
MODE_OUT	Output	INT	20	0, 1, 3, 4, 5	0	Active/set operating mode

Parameters for "search for reference" mode

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
DRV_EN	Input	BOOL	4.0	TRUE/FALSE	FALSE	Drive enable
DIR_P	Input	BOOL	4.2	TRUE/FALSE	FALSE	Search for reference in plus direction (positive edge)
DIR_M	Input	BOOL	4.3	TRUE/FALSE	FALSE	Search for reference in minus direction (positive edge)
MODE_IN	Input	INT	6	0, 1, 3, 4, 5	1	Operating mode, 3 = "search for reference"

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
WORKING	Output	BOOL	14.0	TRUE/FALSE	FALSE	Traverse in process
SYNC	Output	BOOL	14.3	TRUE/FALSE	FALSE	SYNC = TRUE: Axis is synchronized
ACT_POS	Output	DINT	16	$-5x10^8$ to $+5x10^8$ pulses	0	Actual position value
MODE_OUT	Output	INT	20	0, 1, 3, 4, 5	0	Active/set operating mode

Parameters for "relative incremental mode"

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
DRV_EN	Input	BOOL	4.0	TRUE/FALSE	FALSE	Drive enable
DIR_P	Input	BOOL	4.2	TRUE/FALSE	FALSE	Traverse in plus direction (positive edge)
DIR_M	Input	BOOL	4.3	TRUE/FALSE	FALSE	Traverse in minus direction (positive edge)
MODE_IN	Input	INT	6	0, 1, 3, 4, 5	1	Operating mode, 4 = relative incremental mode
TARGET	Input	DINT	8	0 to 10^9 Pulse	1000	Distance in pulses (only positive values allowed)
WORKING	Output	BOOL	14.0	TRUE/FALSE	FALSE	Traverse in process
POS_RCD	Output	BOOL	14.1	TRUE/FALSE	FALSE	Position reached
ACT_POS	Output	DINT	16	$-5x10^8$ to $+5x10^8$ pulses	0	Actual position value
MODE_OUT	Output	INT	20	0, 1, 3, 4, 5	0	Active/set operating mode

Parameters for "absolute incremental mode"

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
DRV_EN	Input	BOOL	4.0	TRUE/FALSE	FALSE	Drive enable
START	Input	BOOL	4.1	TRUE/FALSE	FALSE	Start traverse (positive edge)
DIR_P	Input	BOOL	4.2	TRUE/FALSE	FALSE	Traverse in plus direction (positive edge)
DIR_M	Input	BOOL	4.3	TRUE/FALSE	FALSE	Traverse in minus direction (positive edge)
MODE_IN	Input	INT	6	0, 1, 3, 4, 5	1	Operating mode, 5 = absolute incremental mode

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
TARGET	Input	DINT	8	Linear axis : $-5x10^8$ to $+5x10^8$ Rotary axis: 0 to rotary axis end -1	1000	Target in pulses
WORKING	Output	BOOL	14.0	TRUE/FALSE	FALSE	Traverse in process
POS_RCD	Output	BOOL	14.1	TRUE/FALSE	FALSE	Position reached
ACT_POS	Output	DINT	16	$-5x10^8$ to $+5x10^8$ pulses	0	Actual position value
MODE_OUT	Output	INT	20	0, 1, 3, 4, 5	0	Active/set operating mode

Parameters for the job "set reference point"

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
SYNC	Output	BOOL	14.3	TRUE/FALSE	FALSE	Axis is synchronized

Parameters not assigned to the block (static local data):

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
JOB_REQ	STATIC	BOOL	66.0	TRUE/FALSE	FALSE	Job trigger (positive edge)
JOB_DONE	STATIC	BOOL	66.1	TRUE/FALSE	TRUE	New job can be started
JOB_ERR	STATIC	BOOL	66.2	TRUE/FALSE	FALSE	Faulty job
JOB_ID	STATIC	INT	68	1, 2	0	Job, 1 = "set reference point"
JOB_STAT	STATIC	WORD	70	W#16#0000 to W#16#FFFF	W#16#0000	Job error number
JOB_VAL	STATIC	DINT	72	$-5x10^8$ to $+5x10^8$ pulses	0	Job parameter for reference point coordinates

Parameters for the job "delete residual distance"

Parameters not assigned to the block (static local data):

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
JOB_REQ	STATIC	BOOL	66.0	TRUE/FALSE	FALSE	Job trigger (positive edge)
JOB_DONE	STATIC	BOOL	66.1	TRUE/FALSE	TRUE	New job can be started

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
JOB_ERR	STATIC	BOOL	66.2	TRUE/FALSE	FALSE	Faulty job
JOB_ID	STATIC	INT	68	1, 2	0	Job, 2 = "delete residual distance"
JOB_STAT	STATIC	WORD	70	0 to FFFF hex	0	Job error number
JOB_VAL	STATIC	DINT	72	-	0	None

Parameters for the "length measurement" function

This operation is started at the positive edge on the digital input. There are no specific input parameters.

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
MSR_DONE	Output	BOOL	14.2	TRUE/FALSE	FALSE	Length measurement complete

Parameters not assigned to the block (static local data):

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
BEG_VAL	STATIC	DINT	54	$-5x10^8$ to $+5x10^8$ pulses	0	Actual position value start of length measurement
END_VAL	STATIC	DINT	58	$-5x10^8$ to $+5x10^8$ pulses	0	Actual position value length measurement end
LEN_VAL	STATIC	DINT	62	0 to 10^9pulses	0	Measured length

Parameter ERROR and STATUS

The output parameter ERROR is set to TRUE if an error is detected. The parameter STATUS shows the cause of the error.

STATUS Error code	Description
W#16#2001	Incorrect instruction, use DIGITAL instruction.
W#16#2004	Incorrect channel number (CHANNEL). Set channel number "0".
W#16#3001	Traverse job rejected because of job error in the same call. Correct the respective JOB parameters.
W#16#3002	A change of MODE_IN is not permitted while the drive is still running. Wait for the end of the current positioning.
W#16#3003	Unknown operating mode (MODE_IN). Permitted are 1 (jog), 3 (search for reference), 4 (relative incremental mode) and 5 (absolute incremental mode).
W#16#3004	Only one start request is allowed at a time. Valid start requests are DIR_P, DIR_M or START.

STATUS Error code	Description
W#16#3005	START is only permitted in operating mode "absolute incremental mode”. Start the traverse with DIR_P or DIR_M.
W#16#3006	DIR_P and DIR_M are not permitted for linear axis and "absolute incremental mode". Start the traverse with START.
W#16#3007	Axis not synchronized. "Absolute incremental mode" is only possible with a synchronized axis.
W#16#3008	Leave work area. Return traverse to work area is only allowed in jog mode.
W#16#3101	Start is not enabled because parameters have not been assigned for the axis. Assign parameters for the "positioning" submodule via the hardware configuration.
W#16#3102	Start not enabled because drive enable is not set Set the "drive enable" on the instruction (DRV_EN=TRUE).
W#16#3103	Start not enabled because STOP is set. Delete STOP on the instruction (STOP = FALSE).
W#16#3104	Start is not enabled because the axis is currently positioning (WORKING=TRUE). Wait for the end of the current positioning.
W#16#3105	Start not enabled because at least one error that is pending has not been acknowledged. First, eliminate and clear all external errors and then restart traverse.
W#16#3201	Wrong SPEED default. For positioning with digital outputs only "creep speed" (0) and "rapid traverse" (1) are allowed.
W#16#3301	Changeover/cut-off difference is too high. Set a maximum changeover/cut-off difference of 10^8.
W#16#3303	Changeover difference too low. The changeover difference must be greater than or equal to the cut-off difference.
W#16#3304	Cut-off difference too low. The cut-off difference must be at least half the value of the target range.
W#16#3401	Target default outside of work range. For a linear axis and absolute incremental mode the target default must be within the range of the software limit switches (inclusive).
W#16#3402	Wrong target default. For a rotary axis the target default must be greater than "0" and less than the rotary axis end value.
W#16#3403	Wrong distance specification. The traverse distance for relative incremental mode must be positive.
W#16#3404	Wrong distance specification. The resulting absolute target coordinate must be greater than $-5x10^8$.
W#16#3405	Wrong distance specification. The resulting absolute target coordinate must be greater than $-5x10^8$.
W#16#3406	Wrong distance specification. The resulting absolute target coordinate must be within the work range (+/- half of the target range).
W#16#3501	Traverse distance too great. Target coordinate + actual residual distance must be greater than or equal to $-5x10^8$.
W#16#3502	Traverse distance too great. Target coordinate + actual residual distance must be less than or equal to $5x10^8$.
W#16#3503	Traverse distance insufficient. Traverse distance in plus direction must be greater than the specified cut-off difference for plus direction.
W#16#3504	Traverse distance insufficient. The traverse distance in minus direction must be greater than the specified cut-off difference for the minus direction.

STATUS Error code	Description
W#16#3505	Traverse distance insufficient or the limit switch is already overtraveled in plus direction. The last approachable target in plus direction (work range or traverse distance limit) is too close to the current position.
W#16#3506	Traverse distance insufficient, or the limit switch is already overtraveled in the minus direction. The last approachable target in minus direction (work range or traverse distance limit) is too close to the current position.

Job error (JOB_ERR = TRUE)

The parameter JOB_ERR is set to TRUE if an error is detected. The parameter JOB_STAT shows the cause of the error.

Event class error code	Description
W#16#4001	Parameters not assigned for axis. Assign parameters for the "positioning" submodule via the hardware configuration.
W#16#4002	Job not possible because positioning is in progress. Jobs can only be carried out if positioning is not in progress. Wait until WORKING = FALSE, then repeat the job.
W#16#4004	Unknown job. Check the job ID and then repeat the job.
W#16#4101	For a linear axis the reference point coordinate should not be outside of the work area limits.
W#16#4102	For a linear axis the specified reference point coordinate + actual residual distance must be greater than or equal to $-5x10^8$.
W#16#4103	For a linear axis the specified reference point coordinate + actual residual distance must be greater than or equal to $5x10^8$.
W#16#4104	For a linear axis the specified reference point coordinate + actual difference to the starting point of traverse must be greater than or equal to $-5x10^8$.
W#16#4105	For a linear axis the specified reference point coordinate + actual difference to the starting point of traverse must be less than or equal to $5x10^8$.
W#16#4106	For a rotary axis the reference point coordinate should not be lower than 0 and greater than or equal to the rotary axis end.

External error (ERR)

The technology monitors traverse, traverse range, and the connected peripheral devices. The prerequisite is that you have switched on monitoring in the "Drive", "Axis" and "Encoder" parameter screen forms beforehand.

An external fault is reported when the monitoring unit is tripped. External errors can occur at anytime regardless of the started functions. You must always acknowledge external errors with ERR_A = TRUE.

External errors are indicated by a set bit in parameter ERR (WORD).

Monitoring	Error code	Bit in ERR-WORD
Missing pulse (zero mark)	W#16#0004	2
Traverse range	W#16#0800	11

Monitoring	Error code	Bit in ERR-WORD
Work range	W#16#1000	12
Process value	W#16#2000	13
Destination run-up	W#16#4000	14
Target range	W#16#8000	15

System error

A system error is indicated with BIE (binary result) = FALSE. A system error is triggered by errors while reading/writing the instance DB or by multiple calls of the instruction.

COUNT: Control counter

Description

You use the "COUNT" instruction to control the counter from the user program.

The following functionality is available:

- Starting/stopping the counter via the software gate SW_GATE
- Enabling/controlling the output DO
- Retrieving status bits STS_CMP, STS_OFLW, STS_UFLW and STS_ZP
- Retrieving the current counter value COUNTVAL
- Jobs for reading/writing the internal counter registers
- Retrieving the current period duration TIMEVAL

Parameter

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
LADDR	Input	WORD	0	CPU-specific	W#16#0300	The submodule I/O address, which you have specified during hardware configuration. If the I and O addresses are not equal, the lesser of the two addresses must be specified.
CHANNEL	Input	INT	2	CPU 312C: 0 to 1 CPU 313C: 0 to 2 CPU 314C: 0 to 3	0	Channel number
SW_GATE	Input	BOOL	4.0	TRUE/FALSE	FALSE	Software gate for starting/ stopping the counter
CTRL_DO	Input	BOOL	4.1	TRUE/FALSE	FALSE	Enable output

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
SET_DO	Input	BOOL	4.2	TRUE/FALSE	FALSE	Control output
JOB_REQ	Input	BOOL	4.3	TRUE/FALSE	FALSE	Job trigger (positive edge)
JOB_ID	Input	WORD	6	W#16#0000 Job without function W#16#0001 Write count value W#16#0002 Write load value W#16#0004 Write comparison value W#16#0008 Write hysteresis W#16#0010 Write pulse duration W#16#0082 Read load value W#16#0084 Read comparison value W#16#0088 Read hysteresis W#16#0090 Read pulse duration	W#16#0000	Job number
JOB_VAL	Input	DINT	8	-2^{31} to $+2^{31}$ -1	0	Value for write jobs.
STS_GATE	Output	BOOL	12.0	TRUE/FALSE	FALSE	Status of the internal gate
STS_STRT	Output	BOOL	12.1	TRUE/FALSE	FALSE	Status of the hardware gate (start input)
STS_LTCH	Output	BOOL	12.2	TRUE/FALSE	FALSE	Status of the latch input
STS_DO	Output	BOOL	12.3	TRUE/FALSE	FALSE	Output status
STS_C_DN	Output	BOOL	12.4	TRUE/FALSE	FALSE	Status reverse direction. The last count direction is always displayed. The value of STS_C_DN is FALSE after the first call of the instruction.

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
STS_C_UP	Output	BOOL	12.5	TRUE/FALSE	FALSE	Status forward direction The last count direction is always displayed. The value of STS_C_UP is TRUE after the first call of the instruction.
COUNTVAL	Output	DINT	14	-2^{31} to $+2^{31}-1$	0	Actual count value
LATCHVAL	Output	DINT	18	-2^{31} to $+2^{31}-1$	0	Actual latch value
JOB_DONE	Output	BOOL	22.0	TRUE/FALSE	TRUE	New job can be started
JOB_ERR	Output	BOOL	22.1	TRUE/FALSE	FALSE	Faulty job
JOB_STAT	Output	WORD	24	0 to W#16#FFFF	0	Job error number

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Note

If you have set the parameter "response of the output" to "no comparison" via the configuration interface, the following is valid:

- The output will be switched in the same manner that a normal output is switched.
- The input parameters CTRL_DO and SET_DO are not active.
- The status bits STS_DO and STS_CMP (status comparator in the IDB) remain reset.

Parameters not assigned to the block (static local data):

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
STS_CMP	STATIC	BOOL	26.3	TRUE/FALSE	FALSE	Status comparator. Reset with RES_STS. The status bit STS_CMP indicates that the conditions for comparison of the comparator are satisfied or have been satisfied. STS_CMP also indicates that the output was set (STS_DO = TRUE).
STS_OFLW	STATIC	BOOL	26.5	TRUE/FALSE	FALSE	Status overflow Reset with RES_STS.
STS_UFLW	STATIC	BOOL	26.6	TRUE/FALSE	FALSE	Status underflow Reset with RES_STS.

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
STS_ZP	STATIC	BOOL	26.7	TRUE/FALSE	FALSE	Status zero-crossing Reset with RES_STS. Only set for counters without main count direction. Indicates the zero crossing. Is also set when the counter is set to 0 or if the counter starts counting as of load value=0.
JOB_OVAL	STATIC	DINT	28	-2^{31} to $+2^{31}-1$	0	Output value for read jobs.
RES_STS	STATIC	BOOL	32.2	TRUE/FALSE	FALSE	Reset status bits. Resets the status bits STS_CMP, STS_OFLW, STS_UFLW and STS_ZP. Two calls of the instruction are required to reset the status bits.

Parameter JOB_ERR and JOB_STAT

JOB_ERR = TRUE if a job error occurs. The precise error cause is displayed in JOB_STAT.

Event class error code	Description
W#16#0121	Compare value too low.
W#16#0122	Compare value too great.
W#16#0131	Hysteresis too small.
W#16#0132	Hysteresis too great.
W#16#0141	Pulse duration too low.
W#16#0142	Pulse duration too great.
W#16#0151	Load value too low.
W#16#0152	Load value too high.
W#16#0161	Counter value too low.
W#16#0162	Counter value too high.
W#16#01FF	Invalid job number.

System error

If a system error occurs, BIE (binary result) = FALSE.

Event class error code	Description
W#16#8001	Wrong operating mode or faulty parameters. Set the correct operating mode during hardware configuration or use the instruction that matches the set operating mode.
W#16#8009	Invalid channel number. Set a channel number ≤ 3 (CPU-specific value).

FREQUENC: Control frequency measurement

Description

You use the "FREQUENC" instruction to operate the frequency meter of the user program.

The following functionality is available:

- Starting/stopping via the software gate SW_GATE
- Enabling/controlling the output DO
- Retrieving the status bits STS_CMP, STS_OFLW and STS_UFLW
- Retrieving the current frequency value MEAS_VAL
- Jobs for reading/writing the internal frequency counter registers

Parameter

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
LADDR	Input	WORD	0	CPU-specific	W#16#0300	The submodule I/O address, which you have specified during hardware configuration. If the I and O addresses are not equal, the lesser of the two addresses must be specified.
CHANNEL	Input	INT	2	CPU 312C: 0 to 1 CPU 313C: 0 to 2 CPU 314C: 0 to 3	0	Channel number
SW_GATE	Input	BOOL	4.0	TRUE/FALSE	FALSE	Software gate for starting/ stopping the frequency measurement
MAN_DO	Input	BOOL	4.1	TRUE/FALSE	FALSE	Enable manual output control

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
SET_DO	Input	BOOL	4.2	TRUE/FALSE	FALSE	Control output
JOB_REQ	Input	BOOL	4.3	TRUE/FALSE	FALSE	Job trigger (positive edge)
JOB_ID	Input	WORD	6	W#16#0000 = Job without function W#16#0001 = Write low limit W#16#0002 = Write high limit W#16#0004 = Write integration time W#16#0081 = Read integration time W#16#0082 = Read high limit W#16#0084 = Read integration time	0	Job number
JOB_VAL	Input	DINT	8	-2^{31} to $+2^{31}-1$	0	Value for write jobs.
STS_GATE	Output	BOOL	12.0	TRUE/FALSE	FALSE	Status of the internal gate
STS_STRT	Output	BOOL	12.1	TRUE/FALSE	FALSE	Status of the hardware gate (start input)
STS_DO	Output	BOOL	12.2	TRUE/FALSE	FALSE	Output status
STS_C_DN	Output	BOOL	12.3	TRUE/FALSE	FALSE	Status reverse direction. The last count direction is always displayed. The value of STS_C_DN is FALSE after the first call of the instruction.
STS_C_UP	Output	BOOL	12.4	TRUE/FALSE	FALSE	Status forward direction The last count direction is always displayed. The value of STS_C_UP is TRUE after the first call of the instruction.
MEAS_VAL	Output	DINT	14	0 to $+2^{31}-1$	0	Actual frequency value
COUNTVAL	Output	DINT	18	-2^{31} to $+2^{31}-1$	0	Actual count value (starts every time the gate opens at 0)
JOB_DONE	Output	BOOL	22.0	TRUE/FALSE	TRUE	New job can be started
JOB_ERR	Output	BOOL	22.1	TRUE/FALSE	FALSE	Faulty job
JOB_STAT	Output	WORD	24	W#16#0000 to W#16#FFFF	W#16#0000	Job error number

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Note

If you have set the parameter "response of the output" to "no comparison" via the configuration interface, the following is valid:

- The output will be switched in the same manner that a normal output is switched.
- The input parameters MAN_DO and SET_DO are not active.
- The status bit STS_DO remains reset.

Parameters not interconnected to the block (static local data without the reserved parameter RESxx):

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
STS_CMP	STATIC	BOOL	26.3	TRUE/FALSE	FALSE	End of measurement status Reset with RES_STS . The measured value is updated after every expired time interval. Here, the end of measurement is reported by the status bit STS_CMP.
STS_OFLW	STATIC	BOOL	26.5	TRUE/FALSE	FALSE	Status overflow Reset with RES_STS.
STS_UFLW	STATIC	BOOL	26.6	TRUE/FALSE	FALSE	Status underflow Reset with RES_STS.
JOB_OVAL	STATIC	DINT	28	-2^{31} to 2^{31} -1	0	Output value for read jobs.
RES_STS	STATIC	BOOL	32.2	TRUE/FALSE	FALSE	Reset status bits. Resets the status bits STS_CMP, STS_OFLW and STS_UFLW. Two calls of the instruction are required to reset the status bits.

Parameter JOB_ERR and JOB_STAT

JOB_ERR = TRUE if a job error occurs. The precise error cause is displayed in JOB_STAT.

Event class error code	Description
W#16#0221	Integration time too low.
W#16#0222	Integration time too great.
W#16#0231	Low limit of the frequency is too low.
W#16#0232	High limit of the frequency is too high.

Event class error code	Description
W#16#0241	High limit of the frequency is too low.
W#16#0242	High limit of the frequency is too high.
W#16#02FF	Invalid job number.

System error

If a system error occurs, BIE (binary result) = FALSE.

Event class error code	Description
W#16#8001	Wrong operating mode or faulty parameters. Set the correct operating mode during hardware configuration or use the instruction that matches the set operating mode.
W#16#8009	Invalid channel number. Set a channel number ≤ 3 (CPU-specific value).

PULSE: Control pulse width modulation

Description

You use the "PULSE" instruction to control the pulse width modulation of the user program.

The following functionality is available:

- Starting/stopping via the software gate SW_EN
- Enabling/controlling the output DO
- Retrieving the status bits STS_EN, STS_STRT and STS_DO
- Input of the output value
- Jobs for reading/writing the registers

Parameter

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Meaning
LADDR	Input	WORD	0	CPU-specific	W#16#0300	The submodule I/O address, which you have specified during hardware configuration. If the I and O addresses are not equal, the lesser of the two addresses must be specified.
CHANNEL	Input	INT	2	CPU 312C: 0 to 1 CPU 313C: 0 to 2 CPU 314C: 0 to 3	0	Channel number
SW_EN	Input	BOOL	4.0	TRUE/FALSE	FALSE	Software gate for starting/ stopping the output
MAN_DO	Input	BOOL	4.1	TRUE/FALSE	FALSE	Enable manual output control
SET_DO	Input	BOOL	4.2	TRUE/FALSE	FALSE	Control output
OUTP_VAL	Input	INT	6.0	in ppm: 0 to 1000 as S7 analog value: 0 to 27648	0	Default output value if you enter an output value > 1 000 or 27648, the CPU will limit it to 1 000 or 27648
JOB_REQ	Input	BOOL	8.0	TRUE/FALSE	FALSE	Job trigger (positive edge)
JOB_ID	Input	WORD	10	W#16#0000 = Job without function W#16#0001 = Write period time W#16#0002 = Write switch-on delay W#16#0004 = Write minimum pulse duration W#16#0081 = Read period time W#16#0082 = Read switch-on delay W#16#0084 = Read minimum pulse duration	W#16#0000	Job number
JOB_VAL	Input	DINT	12	-2^{31} to$+2^{31}$ -1	0	Value for write jobs.
STS_EN	Output	BOOL	16.0	TRUE/FALSE	FALSE	Enable status

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Meaning
STS_STRT	Output	BOOL	16.1	TRUE/FALSE	FALSE	Status of the hardware gate (start input)
STS_DO	Output	BOOL	16.2	TRUE/FALSE	FALSE	Output status
JOB_DONE	Output	BOOL	16.3	TRUE/FALSE	TRUE	New job can be started
JOB_ERR	Output	BOOL	16.4	TRUE/FALSE	FALSE	Faulty job
JOB_STAT	Output	WORD	18	W#16#0000 to W#16#FFFF	W#16#0000	Job error number

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameters not assigned to the block (static local data):

Parameter	Declaration	Data type	Address (instance DB)	Range of values	Default	Description
JOB_OVAL	Output	DINT	20	-2^{31} to 2^{31} -1	0	Output value for read jobs.

Parameter JOB_ERR and JOB_STAT

JOB_ERR = TRUE if a job error occurs. The precise error cause is displayed in JOB_STAT.

Event class error code	Description
W#16#0411	Period too low.
W#16#0412	Compare value too great.
W#16#0421	On delay too short.
W#16#0422	On delay too long.
W#16#0431	Minimum pulse duration too low.
W#16#0432	Minimum pulse duration too great.
W#16#04FF	Invalid job number.

System error

If a system error occurs, BIE (binary result) = FALSE.

Event class error code	Description
W#16#8001	Wrong operating mode or faulty parameters. Set the correct operating mode during hardware configuration or use the instruction that matches the set operating mode.
W#16#8009	Invalid channel number. Set a channel number ≤ 3 (CPU-specific value).

9.8.2.5 Communication

Common parameter and error information

Common parameters of instructions for S7 basic communication

REQ parameter

The REQ input parameter (request to activate) is a level-triggered control parameter. It is used to trigger the job (the data transfer or the connection termination):

- If you call the instruction for a job that is not currently active, the job is started with REQ = 1. If at the time of the first call of an instruction there is still no connection to the communication partner, then the connection will be established before starting the data transmission.
- If you have initiated a job and it is not yet completed when you call the instruction again for the same job, REQ is not evaluated by the instruction.

Parameters REQ_ID (only "X_SEND" and "X_RCV")

The input parameter REQ_ID is used to identify your send data. It is passed by the operating system of the sending CPU to the "X_RCV (Page 7924)" instruction of the CPU of the communication partner.

You require the REQ_ID parameter at the receiving end

- if you call several "X_SEND (Page 7923)" instructions with different REQ_ID parameters on one send CPU and transfer the data to a communication partner.
- if you use the "X_SEND (Page 7923)" instruction to send data to one communication partner from several send CPUs.

By evaluating REQ_ID, you can save the received data in different memory areas.

Parameters RET_VAL and BUSY

The instructions for S7 basic communication asynchronous, in other words, the processing of a job extends over more than one call. The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 5367).

CONT parameter

The CONT input parameter (continue) is a control parameter. Using this parameter, you decide whether or not a connection to the communication partner remains established after the job has completed.

- If you select CONT=0 at the first call, the connection is terminated again after the data transfer is completed. The connection is then available again for data exchange with a new communication partner.
  This method ensures that connection resources are only occupied when they are actually in use.
- If you select CONT=1 at the first call, the connection remains established after the completion of the data transfer.
  This method is, for example, useful when you exchange data cyclically between two stations.

Note

You can also use "X_ABORT (Page 7931)" or "I_ABORT (Page 7916)" to explicitly abort a connection established with CONT=1.

Error information of communication instructions for non-configured S7 connections

Display

You can classify the "real" error information for the instructions "X_SEND (Page 7923)", "X_RCV (Page 7924)", "X_GET (Page 7928)", "X_PUT (Page 7930)", "X_ABORT (Page 7931)", "I_PUT (Page 7915)", "I_GET (Page 7913)", and "I_ABORT (Page 7916)" specified in the following table, as follows:

Error code (W#16#...)	Explanation
809x	Error on the CPU on which the instruction is running
80Ax	Permanent communication error
80Bx	Error on the communication partner
80Cx	Temporary error

The following table contains specific error information for the instructions.

Error code (W#16#...)	Explanation (general)	Explanation (instruction-specific)
0000	Execution completed without errors.	"X_ABORT (Page 7923)" and "I_ABORT (Page 7916)": REQ=1, and the specified connection is not established.
		"X_RCV (Page 7924)": EN_DT=1 and RD=NIL

Error code (W#16#...)	Explanation (general)	Explanation (instruction-specific)
00xy	-	"X_RCV (Page 7924)" when NDA=1 and RD<>NIL: RET_VAL contains the length of the received data block (with EN_DT=0) or the data block copied to RD (with EN_DT=1).
		"X_GET (Page 7928)": RET_VAL contains the length of the received block of data.
		"I_GET (Page 7913)": RET_VAL contains the length of the received block of data.
7000	-	"X_SEND (Page 7923)", "X_GET (Page 7928)", "X_PUT (Page 7930)", "X_ABORT (Page 7931)", "I_GET (Page 7913)", "I_PUT (Page 7915)" and "I_ABORT (Page 7916)": Call with REQ=0 (call without processing), BUSY has the value "0", no data transfer active.
		"X_RCV (Page 7924)": EN_DT=0/1 and NDA=0
7001	First call with REQ=1: Data transfer was triggered; BUSY has the value "1".	-
7002	Intermediate call (REQ irrelevant): Data transfer is already active; BUSY has the value "1".	"X_ABORT (Page 7931)" and "I_ABORT (Page 7916)": Intermediate call with REQ=1
8090	Specified destination address of the communication partner is invalid, for example: • Wrong IOID • Wrong base address exists • Wrong MPI address (> 126)	-
8092	Error in SD or RD, for example: addressing of the local data area is not permitted.	"X_SEND (Page 7923)", for example • illegal length for SD • SD=NIL is illegal
		"X_RCV (Page 7924)", for example • More data was received than can fit in the area specified by RD. • RD is of the data type BOOL, the received data is, however, longer than one byte.
		"X_GET (Page 7928)" and "I_GET (Page 7913)", for example • illegal length for RD • The length or the data type of RD does not match the received data. • RD=NIL is illegal.
		"X_PUT (Page 7930)" and "I_PUT (Page 7915)", for example • illegal length for SD • SD=NIL is illegal

Error code (W#16#...)	Explanation (general)	Explanation (instruction-specific)
8095	The block is already being executed in a lower priority class.	
80A0	Error in the received acknowledgment	"X_PUT (Page 7930)" and "I_PUT (Page 7915)": The data type specified in the SD of the sending CPU is not supported by the communication partner.
80A1	Communications problems: Instruction called after existing connection was canceled	"AS_DIAL (Page 7937)": • The processing of a communication instruction was rejected by the local TS Adapter because no remote connection has been set up by "AS_DIAL". • The remote connection to the remote TS Adapter was aborted during the processing of the communication instruction.
80B0	Object is not reachable, for example, DB not loaded	Possible for "X_GET (Page 7928)" and "X_PUT (Page 7930)" and "I_GET (Page 7913)" and "I_PUT (Page 7915)"
80B1	Error in the ANY pointer. The length of the data area to be transferred is incorrect.	-
80B2	Hardware error: module does not exist • The configured slot is not occupied. • Actual module type does not match expected type • Distributed I/O is not available. • No entry for the module in the corresponding SDB.	Possible for "X_GET (Page 7928)" and "X_PUT (Page 7930)" and "I_GET (Page 7913)" and "I_PUT (Page 7915)".
80B3	Data may either only be read or only written, for example, write-protected DB	Possible for "X_GET (Page 7928)" and "X_PUT (Page 7930)" and "I_GET (Page 7913)" and "I_PUT (Page 7915)".
80B4	Data type error in the ANY pointer, or ARRAY of the specified data type not permitted.	"X_GET (Page 7928)" and "X_PUT (Page 7930)" and "I_GET (Page 7913)" and "I_PUT (Page 7915)": The data type specified in VAR_ADDR is not supported by the communication partner.
80B5	Execution rejected due to illegal mode	Possible for "X_SEND (Page 7923)". With "AS_DIAL (Page 7937)": • The processing of a communication instruction was rejected by local TS Adapter because the "DIAL" function of "AS_DIAL" is not yet completed. • The DIAL function of "AS_DIAL" was requested even though a remote connection is already set up for AS-AS remote coupling.

Error code (W#16#...)	Explanation (general)	Explanation (instruction-specific)
80B6	The received acknowledgment contains an unknown error code.	-
80B7	Data type and/or length of the transferred data does not fit in the area on the partner CPU to which it should be written.	Possible for "X_PUT (Page 7930)" and "I_PUT (Page 7915)"
80B8	-	"X_SEND (Page 7923)": "X_RCV (Page 7924)" of the communication partner rejected the data acceptance (RD=NIL).
80B9	-	"X_SEND (Page 7923)": The block of data was identified by the communication partner (calling "X_RCV (Page 7924)" with EN_DT=0), it has not yet been entered in the user program because the partner is in STOP mode.
80BA	The response of the communication partner does not fit in the communication frame.	-
80C0	The specified connection is being used by another job.	-
80C1	Lack of resources on the CPU on which the instruction is executed, for example: • The maximum number of different send jobs is already being executed on the module. • The connection resource is in use, for example, receiving data.	-
80C2	Temporary lack of resources on the communication partner, for example: • The communication partner is currently processing the maximum number of jobs. • The required resources (memory, etc.) are being used. • Not enough work memory. (Compress memory).	With "AS_DIAL (Page 7937)": • Temporary lack of resources on the remote CPU. • The remote CPU with the MPI address is not available or does not exist.
80C3	Error in establishing a connection, for example: • The local S7 station is not attached to the MPI subnet. • You have addressed your own station on the MPI subnet. • The communication partner is no longer reachable. • Temporary lack of resources on the communication partner	-

S7 communication

Overview of the S7 communication instructions

Classification

For S7 communication, connection configuration is needed. The integrated communication functions are called in the user program via instructions.

These instructions can be classified in the following categories:

- Instructions for data exchange
- Instructions for changing the operating mode
- Instructions for querying the operating mode
- One instruction each for querying the connection

Note

If your S7-300 CPU does not have an Ethernet interface, you will need a SIMATIC NET CP (Page 7712) from the S7-300 product range to execute S7-300 instructions. For further information, please see the related documentation.

Instructions for data exchange

Communication instructions for data exchange are used to exchange data between two communication partners. If a communication instruction exists only on the local module, this is known as data exchange configured at one end. If it exists on the local as well as on the remote module, this is known as data exchange configured at both ends.

Instruction	Brief description
"USEND (Page 7371)" / "URCV (Page 7373)"	Fast unacknowledged exchange of data regardless of the sequential processing of the communication function ("URCV") on the communication partner (for example, operational and maintenance alarms). This means that the data can be overwritten by more recent data at the communication partner.
"BSEND (Page 7376)" / "BRCV (Page 7379)"	Secure transfer of a data block to the communication partner. This means that data transmission is not complete until the receive function ("BRCV") on the communication partner has accepted the data.
"GET (Page 7366)"	Program-controlled reading of tags without additional communication function in the user program of the communication partner.
"PUT (Page 7368)"	Program-controlled writing of tags without additional communication function in the user program of the communication partner.
"PRINT (Page 7394)"	Send data to a printer (S7-400 only: CP441).

Instructions for changing the operating mode

With instructions for changing the operating mode, you control the operating mode of a remote device.

The data exchange instructions for changing the operating mode are configured at one end.

Instruction	Brief description
"START (Page 7400)"	Trigger a WARM RESTART of an S7-300/400 CPU if it is in STOP mode.
"STOP (Page 7402)"	STOP of an S7-300/400 CPU if it is in RUN, HOLD, or startup mode
"RESUME (Page 7404)"	Trigger a hot restart of an S7-400 CPU if it is in STOP mode.

Instructions for querying the operating mode

With instructions for querying the operating mode, you can obtain information about the operating mode of a remote device.

With the "STATUS" instruction, data exchange is configured at one end, while with the "USTATUS (Page 7408)" instruction, data exchange is configured at both ends.

Instruction	Brief description
"STATUS (Page 7405)"	Supplies the operating mode of a communication partner (S7-400 CPU) when requested by the user.
"USTATUS (Page 7408)"	Receives the operating mode of an S7-400 CPU when it changes its mode, if the corresponding connection attribute (send operating mode messages) has been set.

Instruction for querying the connection

Instruction for S7-400	Instruction for S7-300	Brief description
"CONTROL (Page 7391)"	-	Queries the state of a connection that belongs to an instance of an instruction.
-	"C_CNTRL (Page 7382)"	Query the status of a connection using the connection ID.

Note

You can also use the "C_DIAG (Page 6743)" instruction to diagnose the current connection status (for S7-400 only).

Data consistency

Definition

The size of the data area which can be modified simultaneously by concurrent processes is called the consistent data area. Data areas which are larger than the consistent data area can thus be falsified as a whole.

This means that a data area which belongs together and which is larger than consistent data area can consist in part of new and of old consistent data blocks at the same time.

Example

An inconsistency can arise if a communication block is interrupted, for example, by a hardware interrupt OB with a higher priority. If the user program in this OB now changes the data which have already been processed in part by the communication block, the transferred data originate:

- In part from the time before the hardware interrupt was processed
- And in part from the time after the hardware interrupt was processed

This means that these data are inconsistent (not coherent).

Effect

If larger packages of data are to be transferred in a consistent form, the transfer should not be interrupted. This can, for example, increase the interrupt reaction time of the CPU.

Data consistency with SIMATIC

- If the user program includes a communication function, for example "BSEND (Page 7376)" / "BRCV (Page 7379)" that accesses common data, access to this data area can, for example, be coordinated using the DONE parameter. The data consistency of the communication areas which are transferred locally with a communication block can therefore be ensured in the user program.
- However, in the case of S7 communication functions, for example "PUT (Page 7368)" / "GET (Page 7366)" or write/read using OP communication, the size of the consistent data areas must already be taken into consideration during programming or configuration, since no communication block is available in the user program of the target device (server) to synchronize communication data in the user program.
- With the S7-300, the communication data are copied consistently into the user memory in blocks of 32 bytes in the cycle control point of the operating system. Data consistency is not guaranteed for larger data areas. If a defined data consistency is required, the communication data in the user program may not exceed 32 bytes (maximum of 8 bytes, depending on the version).

- In the S7-400 by contrast the communication data are not processed in the cycle control point, but in fixed time slices during the program cycle. The consistency of a tag is ensured by the system.
- These communication areas can then be accessed consistently using the "PUT (Page 7368)" / "GET (Page 7366)" instructions or when reading/writing tags, for example by an OP or an OS.

Note

Additional information on data consistency is provided in the description of the specific instructions.

Common parameters of instructions for S7 communication

Classification

The parameters of the instructions for S7 communication can be divided into the following five categories according to their functions:

1. Control parameters are used to activate an instruction.
2. Addressing parameters are used to address the remote communication partner.
3. Send parameters point to the data areas that are to be sent to the remote partner.
4. Receive parameters point to the data areas where the data received from remote partners will be entered.
5. Status parameters are used to monitor whether the instruction has completed its task without error or for the analysis of any errors that have occurred.

Control parameters

Data exchange will only be activated if the appropriate control parameters have a defined value (for example, are set) when the instruction is called or when the value has undergone a specific change since the previous call (for example, a positive edge).

Note

First call for S7-300

For the first call, set the REQ parameter to FALSE .

Addressing parameters

Parameters	Description
ID	Reference to the local connection description (specified by the configuration of the connection). Note: The ID W#16#EEEE is not permitted for S7communication instructions.
R_ID	With the R_ID parameter, you specify that a send and a receive instruction belong together: The R_ID parameter must match in the instruction at the sending end and the instruction at the receiving end. This allows the communication of several instruction pairs via the same logic connection. • R_ID must be specified in the form DW#16#wxyzWXYZ. • The instruction pairs of a logical connection specified in R_ID must be unique for this connection.

The PI_NAME parameter is only described for the relevant instructions (S7-400 only).

Note

Addressing parameters ID and R_ID

S7-300: You can change the settings of the addressing parameters ID and R_ID during runtime. The new parameters are validated with each new job after the previous job has been closed.

You can use the following options to reduce the number of instance DBs and therefore the work memory required:

1. With tag IDs, you can use several connections via one data instance block.
2. With tag R_IDs, you can define several pairs of sending and receiving instructions for one job with a single instance.
3. You can combine cases 1 and 2.

Please note that the new parameters are valid after the last job is executed. When you activate the sending operation, the R_ID parameter in the instruction at the sending end must match its counterpart at the receiving end.

S7-400: The ID and R_ID addressing parameters are evaluated only when the instruction is called the first time or when loading the instance over another (the actual parameters or the predefined values from the instance). The first call therefore specifies the communication relation (connection) with the remote partner until the next warm or cold restart.

Status parameters

With the status parameters, you monitor whether the instruction has completed its task correctly or whether it is still active. The status parameters also indicate errors.

Note

The status parameters are valid for one cycle only, namely from the first command following the call until the next call. As a result, you must evaluate these parameters after each instruction cycle.

Send and receive parameters

If you do not use all send or receive parameters of an instruction, the first unused parameter must be a NIL pointer and the parameters used must be located one after the other without any gaps.

Note

First call for S7-400

During the first call, the ANY pointer specifies the maximum amount of user data that can be transferred for the job. On the basis of this, a communication data buffer is created in the work memory of the CPU to ensure data consistency. This buffer occupies up to 480 bytes of work memory. We recommend you run the first call in the warm or cold restart OB if the block is not reloaded with the instruction call when the CPU is in RUN mode.

At subsequent calls you can send/receive any amount of data, however, no more than with the first call.

The "BSEND (Page 7376)" and "BRCV (Page 7379)" instructions are an exception to this rule. You can use them to transfer up to 64 kbytes per job.

For communication instructions configured at both ends

- The number of the SD_i and RD_i parameters used must match at the send and receive end.
- The data types of the SD_i and RD_i parameters that belong together must match at the send and receive end.
- The amount of data to be sent according to the SD_i parameter must not exceed the range made available by the corresponding RD_i (does not apply to "BSEND (Page 7376)" / "BRCV (Page 7379)").

If you do not keep to the rules above, this is indicated by ERROR = 1 and STATUS = 4.

User data size

With the "USEND (Page 7371)", "URCV (Page 7373)", "GET (Page 7366)" and "PUT (Page 7368)" instructions, the amount of data to be transmitted must not exceed a defined user data length. The maximum user data size depends on:

- The instruction used
- The communication partner.

The guaranteed minimum size of the user data for an instruction with 1-4 tags is listed in the following table:

Instruction	Partner: S7-300	Partner: S7-400
PUT / GET	160 bytes	400 bytes
USEND / URCV	160 bytes	440 bytes
BSEND / BRCV	32768/65534 bytes	65534 bytes

Further information on the user data size can be found in the technical data of the respective CPU.

Exact user data size

If the user data size specified above is insufficient you can determine the maximum byte length of the user data as follows:

- First read the data block size valid for communication from the following table:

Local CPU	Remote CPU	Data block size in bytes
S7-300	Any	240 (S7-300)
S7-400	S7-300	240 (S7-400)
S7-400	S7-400	480

- Use this value in the following table to read the maximum possible user data length in bytes as total of the parameters used. It applies for even lengths of the areas SD_i, RD_i, and ADDR_i. For each range of uneven length the maximum possible user data length is reduced by one byte.

		Number of SD_i, RD_i, ADDR_i parameters used			
Data block size	Instruction	1	2	3	4
240 (S7-300)	PUT/GET/ USEND	160	-	-	-
240 (S7-300 via integrated interface)	PUT	212	-	-	-
	GET	222	-	-	-
	USEND	212	-	-	-
240 (S7-400)	PUT	212	196	180	164
	GET	222	218	214	210
	USEND	212	-	-	-
480	PUT	452	436	420	404
	GET	462	458	454	450
	USEND	452	448	444	440

Startup characteristics of instructions for S7 communication

Requirements

In the following description for S7-400, it is assumed that:

- The connection descriptions (SDBs) exist on the modules.
- The configured connections have been established.
- The actual parameter for ID matches the configured connection ID for each instruction.

Warm restart and cold restart

During a warm and a cold restart, all instructions are set to the NO_INIT status. The actual parameters stored in the instance DBs are unchanged.

Warm restart and cold restart with data exchange instructions configured at both ends

With data exchange instructions configured at both ends, the two modules do not normally both run a warm or cold restart at he same time. The adjustment is made implicitly by the instruction based on the following rules.

- The receive blocks ("URCV (Page 7373)", "BRCV (Page 7379)") react as follows:
  - If the instruction has received a job but has not acknowledged this job at the time of the warm or cold restart, it generates a sequence abort frame ("BRCV (Page 7379)") and then immediately branches to the NO_INIT status.
  - With "BRCV (Page 7379)", it is possible that another data segment will be received despite the transferred sequence abort. This will be discarded locally.
  - With "URCV (Page 7373)", the change to the NO_INIT status occurs immediately.
- The send blocks ("USEND (Page 7371)", "BSEND (Page 7376)") react as follows:
  - If the "BSEND (Page 7376)" instruction has started a job sequence that has not yet been completed, it sends a sequence abort during the warm or cold restart. It then branches to the NO_INIT status immediately afterwards. An acknowledgement that arrives at a later time is discarded locally.
  - If the "BSEND (Page 7376)" instruction has already sent or received a sequence abort when the warm or cold restart is requested, it changes immediately to the NO_INIT status.
  - In all other cases and if the instruction sends only alarms (for example, "USEND (Page 7371)"), local processing is aborted and the instruction immediately branches to the NO_INIT status.

Warm restart and cold restart with data exchange instructions configured at one end

It is assumed that the server on the communication partner is operational after the connections have been established, in other words that it can process jobs or output alarms at any time.

Instructions that send out jobs and expect acknowledgements respond as follows:

- The current processing is aborted followed by a branch to the NO_INIT status immediately afterwards. If an acknowledgement for the job sent prior to the warm or cold restart arrives later, it is discarded locally.
- A new job may have been sent before the acknowledgement of the earlier job is received.

Instructions that output or receive alarms react as follows:

- The current processing is aborted followed by a branch to the NO_INIT status immediately afterwards.
- With "USTATUS (Page 7408)", alarms that arrive during the NO_INIT and DISABLED statuses are discarded locally.

Hot restart

The instructions for S7 communication are set to the NO_INIT status only during a warm or cold restart. This means that they react like user function blocks that can be resumed following a hot restart.

Memory reset

A memory reset always causes all connections to be terminated. Since a warm or cold restart is the only possible startup type for the user program after a memory reset, all instructions for S7 communication (if they still exist) are set to the NO_INIT status and initialized. Partner blocks in a module whose memory was not reset change to the IDLE or ENABLED or DISABLED statuses as a reaction to the connection being aborted.

Error behavior of instructions for S7 communication

Introduction

The following section describes how instructions for S7 communication in S7-400 react to problems.

Connection abort

The connections assigned to the instruction instances are monitored for aborts.

If a connection is aborted, the reaction of the instruction depends on its internal status:

- If the connection abort is detected in the IDLE or ENABLED status, the instruction reacts as follows:
  - It branches to the ERROR status and outputs the error ID "Communication problems" at the ERROR and STATUS output parameters.
  - When it is next called, the instruction returns to its original status and checks the connection again.
- An instruction that is not in the IDLE or DISABLED status reacts as follows:
  - It aborts its processing, changes to the ERROR status immediately or at the next call, and outputs the error ID "Communication problems" at the ERROR and STATUS output parameters.
  - When it is next called, the instruction changes to the IDLE, DISABLED or ENABLED status. In the IDLE and ENABLED statuses, the connection is checked again.

This procedure will also be executed if the connection has again been set up in the meantime.

Power down

A power down with battery backup followed by a restart causes all established connections to be terminated. Therefore, the points made above apply to all instructions involved.

If there is a power down with battery backup followed by an automatic warm or cold restart, the points made about terminated connections and warm or cold restarts apply.

In the special case of an automatic warm or cold restart without battery backup, where a memory reset is executed automatically after power returns, the instructions for S7 communication react as described in the "Startup characteristics of instructions for S7 communication (Page 7362)" section.

Reaction to operating mode changes

If there is an operating mode change between the STOP, STARTUP, RUN, and HOLD modes, the instruction remains in its current status (exception: During a warm or cold restart, it changes to the NO_INIT status). This applies both to communications instructions configured at one end and at both ends.

Error interface to the user program

If an error occurs during the processing of an instruction, it always changes to the ERROR status. At the same time, the ERROR output parameter is set to "1" and the corresponding error ID is entered in the STATUS output parameter. You can evaluate this information in your program.

Examples of possible errors:

- Error when collecting send data.
- Error when copying receive data into the receive areas (for example, attempting to access a DB that does not exist)
- The length of the data area sent does not match the length of the receive area specified in the partner instruction.

GET: Read data from a remote CPU

Description

With the help of the instruction, you can read data from a remote CPU if the connection does not take place via a CP.

- **S7-300:** The data is read on a positive edge at REQ. The parameter values ID, ADDR_1 and RD_1 are adopted on each positive edge at REQ . After a job has been completed, you can assign new values to the ID, ADDR_1 and RD_1 parameters.
- **S7-400:** The instruction is started on a positive edge at control input REQ . The relevant pointers to the areas to be read out (ADDR_i) are then sent to the partner CPU.

The remote partner returns the data. The received data is copied to the configured receive areas (RD_i) at the next call. Make sure that the areas defined with the parameters ADDR_i and RD_i match in terms of number, length and data type.

Completion of this action is indicated by the status parameter NDR having the value "1". Reading can only be activated again after the previous reading process has been completed.

The remote CPU can be in RUN or STOP mode. Errors and warnings are output via ERROR and STATUS if access problems occurred while the data was being read or if the data type check results in an error.

Parameters

The following table shows the parameters of the "GET" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter request, activates the data exchange on a rising edge.
ID	Input	WORD	M, D or constant	Addressing parameter ID See also: Common parameters of instructions for S7 communication (Page 7359)
NDR	Output	BOOL	I, Q, M, D, L	Status parameter NDR: • 0: Job not yet started or still running. • 1: Job successfully completed.
ERROR / STATUS	Output Output	BOOL WORD	I, Q, M, D, L I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: 0000H: Neither warning nor error <> 0000H: Warning, STATUS supplies detailed information. • ERROR=1 An error has occurred. STATUS supplies detailed information on the type of error.
S7-300: ADDR_1 **S7-400:** ADDR_i (1≤ i ≤4)	InOut	ANY	S7-300: M, D S7-400: I, Q, M, D, T, C	Pointers to the areas on the partner CPU that are to be read. **Note:** If the ANY pointer accesses a DB, the DB must always be specified (for example: P#DB10.DBX5.0 byte 10). When transferring data structures (e.g., Struct, Array), the data type CHAR, BYTE, WORD or DWORD must be used at the ADDR parameter.
S7-300: RD_1 **S7-400:** RD_i (1≤ i ≤4)	InOut	ANY	S7-300: M, D S7-400: I, Q, M, D, T, C	Pointers to the areas on the local CPU in which the read data is entered. Only the data types BOOL are permitted (not permitted: Bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, COUNTER, TIMER. **Note:** If the ANY pointer accesses a DB, the DB must always be specified (for example: P#DB10.DBX5.0 byte 10).

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameters ERROR and STATUS

The following table contains all specific error information for the "GET" instruction that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Warning: • New job not active because the previous job is still busy. • The job is already being processed in a priority class with lower priority.
0	25	Communication has started. The job is being processed.
1	1	Communications problems, for example • Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode) • Connection to partner not yet established • Also for S7-300: – Maximum number of parallel jobs/instances exceeded.
1	2	Negative acknowledgement from the partner device. The function cannot be executed.
1	4	Errors in the receive area pointers RD_i relating to the data length or the data type.
1	8	Access error on the partner CPU.
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	12	When the instruction was called • An instance DB was specified that does not belong to "GET" • A global DB was specified instead of an instance DB • No instance DB found (Remedy: Load the associated instance DB again).
1	20	• **S7-400** Not enough work memory. Remedy: Reduce the program code in the memory. • **S7-300** – Maximum number of parallel jobs/instances exceeded – The instances were loaded over others with the CPU in RUN (STOP-RUN change required on the CPU or CP.) – Possible when first called
1	27	**For S7-300 only:** No function code for this instruction exists in the CPU.

Data consistency

The data is received in a consistent state if the following point is observed:

Evaluate the part of the receive area RD_i currently being used completely before initiating another job.

PUT: Write data to a remote CPU

Description

With the help of the instruction, you can write data to a remote CPU if the connection does not take place via a CP.

- **S7-300:** The data is sent on a positive edge at REQ. The parameter values ID, ADDR_1 and SD_1 are adopted on each positive edge at REQ. After a job has been completed, you can assign new values to the ID, ADDR_1 and SD_1 parameters.
- **S7-400:** The instruction is started on a positive edge at control input REQ. The pointers to the areas to be written (ADDR_i) and the data (SD_i) are then sent to the partner CPU.

The remote partner saves the required data under the addresses supplied with the data and returns an execution acknowledgement. Make sure that the areas defined with the parameters ADDR_i and SD_i match in terms of number, length and data type.

If no errors occur, this is indicated at the next call with status parameter DONE = "1". The writing process can only be activated again after the last job is complete.

The remote CPU can be in RUN or STOP mode. Errors and warnings are output via ERROR and STATUS if access problems occurred while the data was being written or if the execution check results in an error.

Parameters

The following table shows the parameters of the "PUT" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter request, activates the data exchange on a rising edge.
ID	Input	WORD	M, D or constant	Addressing parameter ID See also: Common parameters of instructions for S7 communication (Page 7359)
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: • 0: Job not yet started or still executing • 1: Job executed without errors.
ERROR STATUS	Output Output	BOOL WORD	I, Q, M, D, L I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: 0000H: Neither warning nor error <> 0000H: Warning, STATUS supplies detailed information. • ERROR=1 An error has occurred. STATUS supplies detailed information on the type of error.

Parameter	Declaration	Data type	Memory area	Description
S7-300: ADDR_1 **S7-400:** ADDR_i (1≤ i ≤4)	InOut	ANY	S7-300: M, D S7-400: I, Q, M, D, T, C	Pointers to the areas on the partner CPU to which the data will be written. **Note:** If the ANY pointer accesses a DB, the DB must always be specified (for example: P#DB10.DBX5.0 byte 10). When transferring data structures (e.g., Struct, Array), the data type CHAR, BYTE, WORD or DWORD must be used at the ADDR parameter.
S7-300: SD_1 **S7-400:** SD_i (1≤ i ≤4)	InOut	ANY	S7-300: M, D S7-400: I, Q, M, D, T, C	Pointers to the areas on the local CPU which contain the data to be sent. Only the data types BOOL are permitted (not permitted: Bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, COUNTER, TIMER. **Note:** If the ANY pointer accesses a DB, the DB must always be specified (for example: P#DB10.DBX5.0 byte 10).

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameters ERROR and STATUS

The following table contains all specific error information for PUT that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Warning: • New job not active because the previous job is still busy. • The job is already being processed in a priority class with lower priority.
0	25	Communication has started. The job is being processed.
1	1	Communications problems, for example • Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode) • Connection to partner not yet established • Also for S7-300: – Maximum number of parallel jobs/instances exceeded.
1	2	Negative acknowledgement from the partner device. The function cannot be executed.
1	4	Errors in the send area pointers SD_i relating to the data length or the data type
1	8	Access error on the partner CPU.
1	10	Access to the local user memory not possible (for example, access to a deleted DB)

ERROR	STATUS (decimal)	Explanation
1	12	When the instruction was called • An instance DB was specified that does not belong to "PUT" • A global DB was specified instead of an instance DB • No instance DB found (Remedy: Load the associated instance DB again).
1	20	• S7-400 Not enough work memory. Remedy: Load the associated instance DB again. • S7-300 – Maximum number of parallel jobs/instances exceeded – The instances were loaded at CPU-RUN (STOP-RUN transition of the CPU or CP required). – Possible when first called
1	27	For S7-300 only: No function code for this instruction exists in the CPU.

Data consistency for S7-300:

To ensure data consistency, send area SD_1 may not be written to again until the current send job has been completed. This is the case when the value of the status parameter DONE changes to "1".

Data consistency for S7-400 and S7-300 via an integrated interface:

When a send operation is activated (rising edge at REQ), the data to be sent from the send area SD_i is copied from the user program. After the instruction call, you can write to these areas again without corrupting the current send data.

Note

The send operation is only complete when the DONE status parameter has the value "1".

USEND: Send data uncoordinated

Description

The "USEND" instruction sends data to a remote partner instruction of type "URCV (Page 7373)". The sending process is carried out without coordination with the partner instruction. This means that the data transfer is carried out without acknowledgment by the partner instruction.

- S7-300: The data is sent on a positive edge at REQ. The parameter values R_ID, ID and SD_1 are adopted on each positive edge at REQ . After a job has been completed, you can assign new values to the R_ID, ID and SD_1 parameters.
- S7-400:The data is sent on a positive edge at control input REQ. The data to be sent is referenced by the parameters SD_1, ... SD_4 (it is not necessary for all four send parameters to have a value).

You must, however, make sure that the areas defined by the parameters SD_1 to SD_4 / SD_1 and RD_1 to RD_4 / RD_1 (with the associated partner instruction "URCV (Page 7373)") match in terms of:

- Number
- Length and
- Data type

The R_ID parameter must be identical in both instructions.

Successful completion of the send operation is indicated when the value of the DONE status parameter is set to logical "1".

Parameters

The following table shows the parameters of the "USEND" instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter request, activates the data exchange on a positive edge.
ID	Input	WORD	M, D or constant	Addressing parameter ID See also: Common parameters of instructions for S7 communication (Page 7359)
R_ID	Input	DWORD	I, Q, M, D, L or constant	Addressing parameter R_ID See also: Common parameters of instructions for S7 communication (Page 7359)
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: • 0: Job not yet started or still executing • 1: Job executed without errors.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: 0000H: Neither warning nor error <> 0000H: Warning, STATUS supplies detailed information • ERROR=1 An error has occurred, STATUS supplies detailed information on the type of error.
STATUS	Output	WORD	I, Q, M, D, L	
S7-300: SD_1 **S7-400:** SD_i (1≤ i ≤4)	InOut	ANY	S7-300: M, D S7-400: I, Q, M, D, T, C	Pointer on the i-th send area. Only the BOOL data types are permitted (not permitted: Bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER. **Note:** If the ANY pointer accesses a DB, the DB must always be specified (for example: P#DB10.DBX5.0 byte 10).

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameters ERROR and STATUS

ERROR	STATUS (decimal)	Explanation
0	11	Warning: • New job not active because the previous job is still busy. • The job is already being processed in a priority class with lower priority.
0	25	Communication has started. The job is being processed.
1	1	Communication problems, for example • Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode) • Connection to partner not yet established • Also for S7-300: – Maximum number of parallel jobs/instances exceeded
1	4	Errors in the send area pointers SD_i relating to the data length or the data type.
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	12	When the instruction was called • An instance DB was specified that does not belong to "USEND" • A global DB was specified instead of an instance DB • No instance DB found (Remedy: Load the associated instance DB again).
1	18	• R_ID already exists in the connection ID. • Also for S7-300: The instances were overloaded at CPU-RUN (STOP-RUN transition of the CPU or CP required).
1	20	• **S7-400** Not enough work memory. Remedy: Reduce the program code in the memory. • **S7-300** – Maximum number of parallel jobs/instances exceeded – The instances were loaded at CPU-RUN (STOP-RUN transition of the CPU or CP required). – Possible when first called
1	27	**For S7-300 only:** No function code for this instruction exists in the CPU.

Data consistency

S7-300: To ensure data consistency, send area SD_1 may not be written to again until the current send job has been completed. This is the case when the value of the status parameter DONE changes to "1".

S7-400 and S7-300 via an integrated interface: When a send operation is activated (positive edge at REQ), the data to be sent from the send area SD_i is copied from the user program.

After the instruction call, you can write to these areas again without corrupting the current send data.

Note

The send operation is only complete when the DONE status parameter has the value "1".

URCV: Receive data uncoordinated

Description

The "URCV" instruction receives data asynchronously from a remote partner instruction of type "USEND (Page 7370)" and copies this data into the configured receive areas.

The instruction is ready to receive if there is a logical 1 at the EN_R input. An active job can be canceled with EN_R=0.

- **S7-300:** The parameters R_ID, ID and RD_1 are adopted on each positive edge at EN_R. After a job has been completed, you can assign new values to the R_ID, ID and RD_1 parameters.
- **S7-400:** The receive data areas are referenced by the parameters RD_1 to RD_4.

Make sure that the areas defined by the parameters RD_i / RD_1 and SD_i / SD_1 (with the associated partner instruction "USEND (Page 7370)") match in terms of:

- Number
- Length and
- Data type

Successful completion of the copy operation is indicated when the value of the NDR status parameter is set to logical "1". The R_ID parameter must be identical in both instructions.

Parameters

The following table shows the parameters of the "URCV" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L	Control parameter enabled to receive, signals ready to receive if the input is set.
ID	Input	WORD	M, D or constant	Addressing parameter ID See also: Common parameters of instructions for S7 communication (Page 7359)
R_ID	Input	DWORD	I, Q, M, D, L or constant	Addressing parameter R_ID See also: Common parameters of instructions for S7 communication (Page 7359)
NDR	Output	BOOL	I, Q, M, D, L	Status parameter NDR: • 0: Job not yet started or still running. • 1: Job successfully completed.

Parameters	Declaration	Data type	Memory area	Description
ERROR	Output	BOOL	I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: 0000H: Neither warning nor error <> 0000H: Warning, STATUS supplies detailed information. • ERROR=1 An error has occurred, STATUS supplies detailed information on the type of error.
STATUS	Output	WORD	I, Q, M, D, L	
S7-300: RD_1 **S7-400:** RD_i (1≤ i ≤4)	InOut	ANY	S7-300: M, D S7-400: I, Q, M, D, T, C	Pointer to the i-th receive area: Only the BOOL data types are permitted (not permitted: Bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER. **Note:** If the ANY pointer accesses a DB, the DB must always be specified (for example: P#DB10.DBX5.0 byte 10).

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameters ERROR and STATUS

ERROR	STATUS (decimal)	Explanation
0	9	Overrun warning: older received data were overwritten by newer received data.
0	11	Warning: The received data is already being processed in a priority class with lower priority.
0	25	Communication has started. The job is being processed.
1	1	Communication problems, for example • Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode) • Connection to partner not yet established • Also for S7-300: Maximum number of parallel jobs/instances exceeded
1	4	Errors in the receive area pointers RD_i relating to the data length or the data type.
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	12	When the instruction was called • An instance DB was specified that does not belong to "URCV" • A global DB was specified instead of an instance DB • No instance DB found (Remedy: Load the associated instance DB again).
1	18	• R_ID already exists in the connection ID. • Also for S7-300: The instances were loaded over others with the CPU in RUN (STOP-RUN change required on the CPU or CP.)
1	19	The associated "USEND (Page 7370)" instruction sends data faster than this can be copied to the receive areas by "URCV".

ERROR	STATUS (decimal)	Explanation
1	20	• **S7-400** Not enough work memory. Remedy: Reduce the program code in the memory. • **S7-300** – Maximum number of parallel jobs/instances exceeded – The instances were loaded over others with the CPU in RUN (STOP-RUN change required on the CPU or CP.) – Possible when first called
1	27	**For S7-300 only:** No function code for this block exists in the CPU.

Data consistency

The data is received in a consistent state if the following point is observed:

- **S7-300:** After the status parameter NDR has changed to value "1", you must call "URCV" again immediately with value "0" at EN_R. This ensures that the receive area is not already overwritten before you have evaluated it. Evaluate the receive area RD_1 completely before you call the block with value "1" at control input EN_R.
- **S7-400:** After the status parameter NDR has changed to the value "1", there is new receive data in your receive areas (RD_i). A new block call may cause these data to be overwritten with new receive data. If you want to prevent this, call "URCV" (for example, during cyclic block processing) with the value 0 at EN_R until you have finished processing the received data.

BSEND: Send data in segments

Description

The "BSEND" instruction sends data to a remote partner instruction of type "BRCV (Page 7379)". With this type of data transfer, more data can be transported between the communication partners than is possible with all other communication instructions for configured S7 connections. The following aggregates can be transferred:

- 32768 bytes for S7-300 via SIMATIC-Net CPs
- 65534 bytes for S7-400 and S7-300 via an integrated interface

Functional description

The data area to be transmitted is segmented. Each segment is sent individually to the partner. The last segment is acknowledged by the partner when it is received, regardless of the associated "BRCV (Page 7379)" call.

- **S7-300:** The data is sent on a positive edge at REQ. The parameters R_ID, ID, SD_1 and LEN are adopted on each positive edge at REQ . After a job has been completed, you can assign new values to the R_ID, ID, SD_1 and LEN parameters. To transfer segmented data, the instruction must be called cyclically in the user program.
  The start address and the maximum length of the data to be sent are specified by SD_1. The length of the data block is specified by LEN based on the job.
- **S7-400 and S7-300 via an integrated interface:** The send operation is activated after the instruction is called and a positive edge on control input REQ. Sending the data from the user memory is asynchronous to the execution of the user program.
  The start address of the data to be sent is specified by SD_1. The length of the send data is specified by LEN based on the job. In this case, LEN replaces the length section of SD_1.

The R_ID parameter must be identical in the related instructions. If there is a positive edge at control input R, the current sending process is canceled. Successful completion of the send operation is indicated by the value of the DONE status parameter being set to "1".

A new send job cannot be processed until the previous send operation has been completed and the status parameter DONE or ERROR has the value "1".

Due to the asynchronous data transfer, a new data transfer can only be initiated if the previous data has been fetched by a partner instruction call. Until the data is fetched, the status value "7" (see below) will be output when "BSEND" is called.

Parameters

The following table shows the parameters of the "BSEND" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter request, activates the data exchange at a positive edge
R	Input	BOOL	I, Q, M, D, L	Control parameter reset, activates an abort of the current data exchange at a positive edge
ID	Input	WORD	M, D or constant	Addressing parameter ID See also: Common parameters of instructions for S7 communication (Page 7359)
R_ID	Input	DWORD	I, Q, M, D, L or constant	Addressing parameter R_ID See also: Common parameters of instructions for S7 communication (Page 7359) If there is a connection via a CP 441 to S5 or third-party devices, R_ID contains the address information of the remote device. For further information, refer to the description of the CP 441.
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: • 0: Job not yet started or still executing. • 1: Job completed without error.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameters ERROR and STATUS, error code:

Parameter	Declaration	Data type	Memory area	Description
STATUS	Output	WORD	I, Q, M, D, L	• ERROR=0 STATUS has the value: 0000H: Neither warning nor error <> 0000H: Warning, STATUS supplies detailed information • ERROR=1 An error has occurred, STATUS supplies detailed information on the type of error.
SD_1	InOut	ANY	S7-300: M, D S7-400: I, Q, M, D, T, C	Pointer to the send area. Only the BOOL data types are permitted (not permitted: Bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER. **Note:** When the ANY pointer accesses a DB, the DB must always be specified. (For example: P#DB10.DBX5.0 byte 10)
LEN	InOut	WORD	I, Q, M, D, L	Length of the data field to be sent in bytes

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameters ERROR and STATUS

The following table contains all specific error information for "BSEND" that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Warning: • New job not active because the previous job is still busy. • The job is already being processed in a priority class with lower priority.
0	25	Communication has started. The job is being processed.
1	1	• Communication problems, for example: Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode) • Connection to partner not yet established • Also for S7-300: Maximum number of parallel jobs/instances exceeded.
1	2	Negative acknowledgment of partner instruction. The function cannot be executed.
1	3	R_ID is unknown on the connection specified by the ID or the receive block has not yet been called.
1	4	Error in the send area pointer SD_1 involving the data length or the data type, or the value "0" was transferred at the LEN parameter.
1	5	Reset request was executed.
1	6	Partner instruction is in DISABLED status (EN_R has the value "0"). Also check the input parameters of BRCV for consistency with BSEND.
1	7	Partner instruction is in the wrong state. The receive instruction was not called again after the last data transmission.

ERROR	STATUS (decimal)	Explanation
1	8	Access to remote object in the user memory was rejected: The destination area for the associated "BRCV" is too small. The corresponding BRCV reports ERROR = 1, STATUS = 4 or ERROR = 1, STATUS = 10.
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	12	When the instruction was called • an instance DB was specified that does not belong to "BSEND" • A global DB was specified instead of an instance DB • No instance DB found (Remedy: Load the associated instance DB again).
1	18	• R_ID already exists in the connection. • Also for S7-300: The instances were loaded over others with the CPU in RUN (STOP-RUN change required on the CPU or CP.)
1	20	• **S7-400** Not enough work memory. Remedy: Reduce the program code in the memory. • **S7-300** – Maximum number of parallel jobs/instances exceeded – The instances were loaded over others with the CPU in RUN (STOP-RUN change required on the CPU or CP.) – Possible when first called – Memory bottleneck in CP
1	27	**For S7-300 only:** No function code for this instruction exists in the CPU.

Data consistency

To ensure data consistency, you may only write to the part of the currently used sending area SD_1 after the current send operation is complete. This is the case when the value of the status parameter DONE changes to "1".

BRCV: Receive data in segments

Description

The "BRCV" instruction receives data from a remote partner instruction of type "BSEND (Page 7375)". After each received data segment, an acknowledgment is sent to the partner instruction and the LEN parameter is updated.

The instruction is ready to receive data after it is called with the value "1" at control input EN_R. An active job can be canceled with EN_R=0.

The start address and the maximum length of the receive area are specified by RD_1. The length of the received block of data is shown in LEN .

- **S7-300:** The parameters R_ID, ID and RD_1 are adopted on each positive edge at EN_R. After a job has been completed, you can assign new values to the R_ID, ID and RD_1 parameters. For the transmission of segmented data, the instruction must be called cyclically in the user program.
- **S7-400 and S7-300 via an integrated interface:** Receipt of the data from the user memory is asynchronous to the execution of the user program.

The R_ID parameter must be identical in the related instructions.

Value "1" at the NDR status parameter indicates that all data segments have been received without error. The received data remains unchanged until the next call with EN_R=1.

If the instruction is called again while data is being received asynchronously, this leads to a warning being output in the STATUS parameter; if the call is made with EN_R=0, receiving is canceled and the instruction returns to its initial state.

Parameters

The following table shows the parameters of the "BRCV" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L	Control parameter enabled to receive, signals ready to receive if the input is set.
ID	Input	WORD	M, D or constant	Addressing parameter ID See also: Common parameters of instructions for S7 communication (Page 7359)
R_ID	Input	DWORD	I, Q, M, D, L or constant	Addressing parameter R_ID See also: Common parameters of instructions for S7 communication (Page 7359) If there is a connection via a CP 441 to S5 or third-party devices, R_ID contains the address information of the remote device. For further information, refer to the description of the CP 441.
NDR	Output	BOOL	I, Q, M, D, L	Status parameter NDR: • 0: Job not yet started or still running. • 1: Job successfully completed.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: 0000H: Neither warning nor error <> 0000H: Warning, STATUS supplies detailed information. • ERROR=1 An error has occurred, STATUS supplies detailed information on the type of error.
STATUS	Output	WORD	I, Q, M, D, L	

Parameter	Declaration	Data type	Memory area	Description
RD_1	InOut	ANY	S7-300: M, D S7-400: I, Q, M, D, T, C	Pointer to the receive area. The length information specifies the maximum length of the block to be received. Only the BOOL data types are permitted (not permitted: Bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER. **Note:** When the ANY pointer accesses a DB, the DB must always be specified. (For example: P# DB10.DBX5.0 byte 10).
LEN	InOut	WORD	I, Q, M, D, L	Length of the data already received in bytes.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameters ERROR and STATUS

The following table contains all specific error information for "BRCV" that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Warning: The received data is already being processed in a priority class with lower priority.
0	17	Warning: Instruction receives data asynchronously. The LEN parameter shows the amount of data already received in bytes.
0	25	Communication has started. The job is being processed.
1	1	Communication problems, for example • Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode) • Connection to partner not yet established • Also for S7-300: Maximum number of parallel jobs/instances exceeded.
1	2	Function cannot be executed (protocol error)
1	4	Error in the receive area pointer RD_1 relating to the data length or data type. The block of data sent is longer than the receive area.
1	5	Reset request received, incomplete transfer.
1	8	Access error in associated "BSEND (Page 7375)": After the last valid data segment has been sent, ERROR = 1 and STATUS = 4 or ERROR = 1 and STATUS = 10 is reported.
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	12	When the instruction was called • An instance DB was specified that does not belong to "BRCV" • A global DB was specified instead of an instance DB • No instance DB found (Remedy: Load the associated instance DB again).

ERROR	STATUS (decimal)	Explanation
1	18	• R_ID already exists in the connection. • Also for S7-300: The instances were loaded over others with the CPU in RUN (STOP-RUN change required on the CPU or CP.)
1	20	• **S7-400** Not enough work memory. Remedy: Reduce the program code in the memory. • **S7-300** – Maximum number of parallel jobs/instances exceeded – The instances were loaded over others with the CPU in RUN (STOP-RUN change required on the CPU or CP.) – Possible when first called – Memory bottleneck in CP
1	27	**For S7-300 only:** No function code for this instruction exists in the CPU.

Data consistency

The data is received in a consistent state if the following point is observed: Evaluate the currently used part of the receive area RD_1 completely before you call the block again with the value 1 at control input EN_R.

Special case for receiving data (S7-400 only)

If the receiving CPU has a "BRCV" instruction that is ready to receive data (in other words, a call with EN_R =1 has already been made) and the CPU changes to STOP mode before the corresponding send instruction has sent the first data segment of a job, the following will occur:

- The data in the first job after the receiving CPU has gone into STOP mode are fully entered in the receive area.
- The "BSEND (Page 7375)" partner instruction receives a positive acknowledgment of this.
- No more "BSEND (Page 7375)" jobs can be accepted by a receiving CPU in STOP mode.
- As long as the CPU remains in STOP mode, both NDR and LEN have the value "0".

To prevent information about the received data being lost, perform a hot restart on the receiving CPU and call "BRCV" with EN_R=1.

C_CNTRL: Query connection status

Description

With this instruction, you query the status of a connection **for S7-300**. The current status of the connection addressed by ID is obtained following the instruction call with value "1" at the EN_R control input.

Parameters

The following table shows the parameters of the "C_CNTRL" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L	Control parameter, signals ready to receive if the input is set.
ID	Input	WORD	M, D or constant	Addressing parameter ID See also: Common parameters of instructions for S7 communication (Page 7359)
RET_VAL	Output	INT	I, Q, M, D, L	Error information
ERROR STATUS	Output Output	BOOL WORD	I, Q, M, D, L I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: 0000H: Neither warning nor error <> 0000H: Warning, STATUS supplies detailed information. • ERROR=1 An error has occurred, STATUS supplies detailed information on the type of error.
C_CONN	Output	BOOL	I, Q, M, D, L	Status of the corresponding connection. Possible values: • 0: Connection aborted or not yet established. • 1: Connection available.
C_STATUS	Output	WORD	I, Q, M, D, L	Connection status: • W#16#0000: Connection is not set up • W#16#0001: Connection is being set up • W#16#0002: Connection is set up • W#16#000F: No data on connection status available (for example during CP startup) • W#16#00FF: Connection is not configured

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameters RET_VAL, ERROR and STATUS

The RET_VAL output parameter can have the following two values for the "C_CNTRL" instruction:

- 0000H: No error occurred during execution.
- 8000H: An error occurred during execution.

ERROR	STATUS (decimal)	Explanation
1	10	CP access error. Another job is currently running. Repeat job later.
1	27	There is no function code on the CPU for this instruction.

Note

The output parameters ERROR and STATUS need to be evaluated even if the output parameter RET_VAL shows the value 0000H .

Others

GET_S: Read data from a remote CPU

Description

With this instruction, you can read data from a remote CPU **with S7-300**. The connection can take place via the integrated interface as well as a CP.

The data is read on a rising edge at REQ. The parameter values ID, ADDR_1 and RD_1 are adopted on each positive edge at REQ . After a job has been completed, you can assign new values to the ID, ADDR_1 and RD_1 parameters.

The remote partner returns the data. The received data is copied to the configured receive areas (RD_1) at the next call. Make sure that the areas defined with the parameters ADDR_1 and RD_1 match in terms of number, length and data type.

Completion of this action is indicated by the status parameter NDR having the value "1". Reading can only be activated again after the previous reading process has been completed.

The remote CPU can be in RUN or STOP mode. Errors and warnings are output via ERROR and STATUS if access problems occurred while the data was being read or if the data type check results in an error.

Parameters

The following table shows the parameters of the "GET_S" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter request, activates the data exchange on a rising edge.
ID	Input	WORD	M, D or constant	Addressing parameter ID See also: Common parameters of instructions for S7 communication (Page 7359)
NDR	Output	BOOL	I, Q, M, D, L	Status parameter NDR: • 0: Job not yet started or still running. • 1: Job successfully completed.

Parameter	Declaration	Data type	Memory area	Description
ERROR / STATUS	Output Output	BOOL WORD	I, Q, M, D, L I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: 0000H: Neither warning nor error <> 0000H: Warning, STATUS supplies detailed information. • ERROR=1 An error has occurred. STATUS supplies detailed information on the type of error.
ADDR_1	InOut	ANY	M, D	Pointers to the areas on the partner CPU that are to be read.
RD_1	InOut	ANY	M, D	Pointers to the areas on the local CPU in which the read data is entered. Only the data types BOOL are permitted (not permitted: Bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, COUNTER, TIMER. **Note:** If the ANY pointer accesses a DB, the DB must always be specified (for example: P#DB10.DBX5.0 byte 10).

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameters ERROR and STATUS

The following table contains all specific error information for the "GET_S" instruction that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Warning: • New job not active because the previous job is still busy. • The job is already being processed in a priority class with lower priority.
0	25	Communication has started. The job is being processed.
1	1	Communications problems, for example • Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode) • Connection to partner not yet established • Maximum number of parallel jobs/instances exceeded.
1	2	Negative acknowledgement from the partner device. The function cannot be executed.
1	4	Errors in the receive area pointers RD_1 relating to the data length or the data type.
1	8	Access error on the partner CPU.
1	10	Access to the local user memory not possible (for example, access to a deleted DB).

ERROR	STATUS (decimal)	Explanation
1	12	When the instruction was called • An instance DB was specified that does not belong to "GET_S" • A global DB was specified instead of an instance DB • No instance DB found (Remedy: Load the associated instance DB again).
1	20	• Maximum number of parallel jobs/instances exceeded • The instances were loaded over others with the CPU in RUN (STOP-RUN change required on the CPU or CP.) • Possible when first called
1	27	There is no function code on the CPU for this instruction.

Data consistency

The data is received in a consistent state if the following point is observed:

Evaluate the part of the receive area RD_1 currently being used completely before initiating another job.

PUT_S: Write data to a remote CPU

Description

With this instruction, you can write data to a remote CPU **with S7-300**. The connection can take place via the integrated interface as well as a CP.

The data is sent on a rising edge at REQ. The parameter values ID, ADDR_1 and SD_1 are adopted on each positive edge at REQ. After a job has been completed, you can assign new values to the ID, ADDR_1 and SD_1 parameters.

The remote partner saves the required data under the addresses supplied with the data and returns an execution acknowledgement. Make sure that the areas defined with the parameters ADDR_1 and SD_1 match in terms of number, length and data type.

If no errors occur, this is indicated at the next call with status parameter DONE = "1". The writing process can only be activated again after the last job is complete.

The remote CPU can be in RUN or STOP mode. Errors and warnings are output via ERROR and STATUS if access problems occurred while the data was being written or if the execution check results in an error.

Parameters

The following table shows the parameters of the "PUT_S" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter request, activates the data exchange on a rising edge.
ID	Input	WORD	M, D or constant	Addressing parameter ID See also: Common parameters of instructions for S7 communication (Page 7359)
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: • 0: Job not yet started or still executing • 1: Job executed without errors.
ERROR STATUS	Output Output	BOOL WORD	I, Q, M, D, L I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: 0000H: Neither warning nor error <> 0000H: Warning, STATUS supplies detailed information. • ERROR=1 An error has occurred. STATUS supplies detailed information on the type of error.
ADDR_1	InOut	ANY	M, D	Pointers to the areas on the partner CPU to which the data will be written.
SD_1	InOut	ANY	M, D	Pointers to the areas on the local CPU which contain the data to be sent. Only the data types BOOL are permitted (not permitted: Bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, COUNTER, TIMER. **Note:** If the ANY pointer accesses a DB, the DB must always be specified (for example: P#DB10.DBX5.0 byte 10).

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameters ERROR and STATUS

The following table contains all specific error information for "PUT_S" that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Warning: • New job not active because the previous job is still busy. • The job is already being processed in a priority class with lower priority.
0	25	Communication has started. The job is being processed.

ERROR	STATUS (decimal)	Explanation
1	1	Communications problems, for example • Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode) • Connection to partner not yet established • Maximum number of parallel jobs/instances exceeded.
1	2	Negative acknowledgement from the partner device. The function cannot be executed.
1	4	Errors in the send area pointers SD_1 relating to the data length or the data type.
1	8	Access error on the partner CPU.
1	10	Access to the local user memory not possible (for example, access to a deleted DB)
1	12	When the instruction was called • An instance DB was specified that does not belong to "PUT_S" • A global DB was specified instead of an instance DB • No instance DB found (Remedy: Load the associated instance DB again).
1	20	• Maximum number of parallel jobs/instances exceeded • The instances were loaded over others with the CPU in RUN (STOP-RUN change required on the CPU or CP.) • Possible when first called
1	27	There is no function code on the CPU for this instruction.

Data consistency

To ensure data consistency, send area SD_1 may not be written to again until the current send job has been completed. This is the case when the value of the status parameter DONE changes to "1".

USEND_S: Send data uncoordinated

Description

The instruction "USEND_S" sends data to a remote partner instruction of type "URCV_S (Page 7389)" **with S7-300**. The sending process is carried out without coordination with the partner instruction. This means that the data transfer is carried out without acknowledgement by the partner instruction.

The data is sent on a rising edge at REQ. The parameter values R_ID, ID and SD_1 are adopted on each positive edge at REQ . After a job has been completed, you can assign new values to the R_ID, ID and SD_1 parameters.

Make sure that the areas defined by the parameters SD_1 and RD_1 (for the associated partner instruction "URCV_S (Page 7389)") match in terms of:

- Number
- Length and
- Data type

The R_ID parameter must be identical in both instructions.

Successful completion of the send operation is indicated when the value of the DONE status parameter is set to logical "1".

Parameters

The following table shows the parameters of the "USEND_S" instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter request, activates the data exchange on a rising edge.
ID	Input	WORD	M, D or constant	Addressing parameter ID See also: Common parameters of instructions for S7 communication (Page 7359)
R_ID	Input	DWORD	I, Q, M, D, L or constant	Addressing parameter R_ID See also: Common parameters of instructions for S7 communication (Page 7359)
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: • 0: Job not yet started or still executing • 1: Job has been executed error-free.
ERROR STATUS	Output Output	BOOL WORD	I, Q, M, D, L I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: 0000H: Neither warning nor error <> 0000H: Warning, STATUS supplies detailed information • ERROR=1 An error has occurred, STATUS supplies detailed information on the type of error.
SD_1	InOut	ANY	M, D	Pointer to the send area. Only the BOOL data types are permitted (not permitted: Bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER. **Note:** If the ANY pointer accesses a DB, the DB must always be specified (for example: P#DB10.DBX5.0 byte 10).

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

Parameters ERROR and STATUS

ERROR	STATUS(decimal)	Explanation
0	11	Warning: • New job not active because the previous job is still busy. • The job is already being processed in a priority class with lower priority.
0	25	Communication has started. The job is being processed.
1	1	Communications problems, for example • Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode) • Connection to partner not yet established • Maximum number of parallel jobs/instances exceeded
1	4	Errors in the send area pointers SD_1 relating to the data length or the data type.
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	12	When the instruction was called • An instance DB was specified that does not belong to "USEND_S" • A global DB was specified instead of an instance DB • No instance DB found (Solution: Load the associated instance DB again).
1	18	• R_ID already exists in the connection ID. • The instances were overloaded at CPU-RUN (STOP-RUN transition of the CPU or CP required.)
1	20	• Maximum number of parallel jobs/instances exceeded • The instances were loaded over others with the CPU in RUN (STOP-RUN change required on the CPU or CP.) • Possible when first called
1	27	There is no function code on the CPU for this instruction.

Data consistency

To ensure data consistency, send area SD_1 may not be written to again until the current send job has been completed. This is the case when the value of the status parameter DONE changes to "1".

URCV_S: Receive data uncoordinated

Description

The instruction "URCV_S" receives data asynchronously from a remote partner instruction of type "USEND_S (Page 7387)" and copies this data into the configured receive areas **with S7-300**.

The instruction is ready to receive if there is a logical 1 at the EN_R input. An active job can be canceled with EN_R=0.

The parameter values R_ID, ID and RD_1 are adopted on each positive edge at EN_R. After a job has been completed, you can assign new values to the R_ID, ID and RD_1 parameters.

Make sure that the areas defined by the parameters RD_1 and SD_1 (for the associated partner instruction "USEND_S (Page 7387)") match in terms of:

- Number
- Length
- Data type

Successful completion of the copy operation is indicated when the value of the NDR status parameter is set to logical "1". The R_ID parameter must be identical in both instructions.

Parameters

The following table shows the parameters of the "URCV_S" instruction:

Parameters	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L	Control parameter enabled to receive, signals ready to receive if the input is set.
ID	Input	WORD	M, D or constant	Addressing parameter ID See also: Common parameters of instructions for S7 communication (Page 7359)
R_ID	Input	DWORD	I, Q, M, D, L or constant	Addressing parameter R_ID See also: Common parameters of instructions for S7 communication (Page 7359)
NDR	Output	BOOL	I, Q, M, D, L	Status parameter NDR: • 0: Job not yet started or still executing. • 1: Job completed without errors.
ERROR STATUS	Output Output	BOOL WORD	I, Q, M, D, L I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: 0000H: Neither warning nor error <> 0000H: Warning, STATUS supplies detailed information. • ERROR=1 An error has occurred, STATUS supplies detailed information on the type of error.
RD_1	InOut	ANY	M, D	Pointer to the i-nth receive area: Only the data types BOOL are permitted (not permitted: Bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME, COUNTER, TIMER. **Note:** If the ANY pointer accesses a DB, the DB must always be specified (for example: P#DB10.DBX5.0 byte 10).

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

Parameters ERROR and STATUS

ERROR	STATUS(decimal)	Explanation
0	9	Overrun warning: Older received data are overwritten by newer received data.
0	11	Warning: The received data is already being processed in a priority class with lower priority.
0	25	Communication has started. The job is being processed.
1	1	Communications problems, for example • Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode) • Connection to partner not yet established • Maximum number of parallel jobs/instances exceeded
1	4	Errors in the receive area pointers RD_1 relating to the data length or the data type.
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	12	When the instruction was called • An instance DB was specified that does not belong to "URCV_S" • A global DB instead of an instance DB was specified. • No instance DB found (Solution: Load the associated instance DB again).
1	18	• R_ID already exists in the connection ID. • The instances were loaded over others with the CPU in RUN (STOP-RUN change required on the CPU or CP.)
1	19	The associated "USEND_S (Page 7387)" instruction sends data faster than this can be copied to the receive areas by "URCV_S".
1	20	• Maximum number of parallel jobs/instances exceeded • The instances were loaded over others with the CPU in RUN (STOP-RUN change required on the CPU or CP.) • Possible when first called
1	27	There is no function code in the CPU for this block.

Data consistency

The data is received in a consistent state if the following point is observed:

After the status parameter NDR has changed to "1", call "URCV_S" again immediately with the value "0" at EN_R. This ensures that the receive area is not already overwritten before you have evaluated it. Evaluate the receive area RD_1 completely before you call the block with value "1" at control input EN_R.

CONTROL: Determine connection status for the instance of an instruction

Description

With the "CONTROL" instruction, you can obtain the status of the connection belonging to a local communication instruction instance on an **S7-400**.

After the instruction is called with the value "1" at control input EN_R, the current status of the connection belonging to the communication instruction instance selected with I_DB is queried.

Parameters

The following table shows the parameters of the "CONTROL" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L	Control parameter, signals ready to receive if the input is set.
I_DB	Input	WORD	I, Q, M, D, L or constant	Number of the instance DB
OFFSET	Input	WORD	I, Q, M, D, L or constant	Offset of the data record in bytes in the multi-instance DB (if no multi-instance DB exists, "0" must be entered here).
RET_VAL	Return	INT	I, Q, M, D, L	Error information
ERROR STATUS	Output Output	BOOL WORD	I, Q, M, D, L I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: 0000H: Neither warning nor error <> 0000H: Warning, STATUS supplies detailed information. • ERROR=1 An error has occurred, STATUS supplies detailed information on the type of error.
I_TYP	Output	BYTE	I, Q, M, D, L	Identifier for the block type belonging to the selected instance.
I_STATE	Output	BYTE	I, Q, M, D, L	• = 0: The corresponding instruction instance has not been called since the last cold/warm restart or loading. • <> 0: The corresponding instruction instance has been called at least once since the last cold/warm restart or loading.
I_CONN	Output	BOOL	I, Q, M, D, L	Status of the corresponding connection. Possible values: • 0: Connection aborted or not yet established. • 1: Connection available.
I_STATUS	Output	WORD	I, Q, M, D, L	Status parameter STATUS of the queried communication instruction instance.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Output parameter I_TYP

The following table lists the different instruction types and the corresponding identifiers

Instruction type	Identifier (W#16#...)
USEND (Page 7370)	00
URCV (Page 7373)	01
BSEND (Page 7375)	04
BRCV (Page 7378)	05

Instruction type	Identifier (W#16#...)
GET (Page 7365)	06
PUT (Page 7368)	07
PRINT (Page 7394)	08
START (Page 7400)	0B
STOP (Page 7402)	0C
RESUME (Page 7404)	0D
STATUS (Page 7405)	0E
USTATUS (Page 7408)	0F
ALARM (Page 6649)	15
ALARM_8 (Page 6641)	16
ALARM_8P (Page 6643)	17
NOTIFY (Page 6646)	18
AR_SEND (Page 6669)	19
NOTIFY_8P (Page 6639)	1A
(no instruction exists; I_DB or OFFSET wrong)	FF

Parameters RET_VAL, ERROR and STATUS

The output parameter RET_VAL can have the following two values for the "CONTROL" instruction:

- 0000H: No error occurred during execution.
- 8000H: An error occurred during execution.

ERROR	STATUS (decimal)	Explanation
1	10	Access to local user memory not possible (e.g.: A memory byte that does not exist in the CPU being used was specified as the actual parameter for I_TYP).
1	12	When the instruction was called: • An instance DB was specified at the I_DB parameter that does not belong to the "CONTROL" instruction. • A global DB was specified instead of an instance DB. • No instance DB found (Remedy: Load the associated instance DB again).

Note

The output parameters ERROR and STATUS need to be evaluated even if the output parameter RET_VAL shows the value 0000H .

PRINT: Send data to printer

Description of PRINT

Description

With the "PRINT" instruction, you can send data and a formatting instruction to a remote printer **with S7-400**, for example, using the CP 441.

When there is a rising edge at control input REQ , the format description (FORMAT parameter) and the data (SD_i) are sent to the printer selected with ID and PRN_NR .

If you do not use all four send areas, make sure that the first area is described by the SD_1 parameter, the second area (if it exists) by SD_2, the third area (if it exists) by SD_3.

Successful execution of the job is indicated by the status parameter DONE having the value "1". If any errors occur, they are indicated in the status parameters ERROR and STATUS.

Parameters

The following table shows the parameters of the "PRINT" instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter request, activates the data exchange on a rising edge.
ID	Input	WORD	M, D or constant	Addressing parameter ID See also: Common parameters of instructions for S7 communication (Page 7359)
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: • 0: Job not yet started or still executing • 1: Job has been executed error-free.
ERROR STATUS	Output Output	BOOL WORD	I, Q, M, D, L I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: 0000H: Neither warning nor error <> 0000H: Warning, STATUS supplies detailed information. • ERROR=1 An error has occurred, STATUS supplies detailed information on the type of error.
PRN_NR	InOut	BYTE	I, Q, M, D, L	Printer number

Parameters	Declaration	Data type	Memory area	Description
FORMAT (Page 7396)	InOut	STRING	I, Q, M, D, L	Format description
SD_i (1≤ i ≤4)	InOut	ANY	M, D, T, C	Pointer to the "i-nth" send data area. Only the BOOL data types are permitted (not permitted: Bit array), BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME. **Note:** If the ANY pointer accesses a DB, the DB must always be specified (for example: P# DB10.DBX5.0 byte 10).

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

Parameters ERROR and STATUS

The following table contains all specific error information for "PRINT" that can be output via the ERROR and STATUS parameters.

ERROR	STATUS(decimal)	Explanation
0	11	Warning: • New job not active because the previous job is still busy. • The job is already being processed in a priority class with lower priority.
0	25	Communication has started. The job is being processed.
1	1	Communications problems, for example • Connection description not loaded (local or remote). • Connection interrupted (for example: cable, CPU off, CP in STOP mode).
1	2	Negative acknowledgement from printer. The function cannot be executed.
1	3	PRN_NR is unknown on the communication link specified by the ID.
1	4	Data length or data type related error in the FORMAT in/out parameter or in the send area pointers SD_i.
1	6	The remote printer is in OFFLINE mode.
1	7	The remote printer is not in the correct status (for example, paper out).
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	12	When the instruction was called: • An instance DB was specified that does not belong to "PRINT" • A global DB instead of an instance DB was specified. • No instance DB found (Solution: Load the associated instance DB again).
1	13	Error in the FORMAT (Page 7396) in/out parameter.
1	20	**S7-400** Not enough work memory. Solution: Reduce the program code in the memory.

Amount of transferable data

The amount of data that can be transferred to a remote printer must not exceed a maximum length. This maximum data length is calculated as follows:

```
maxleng = 420 - format
```

In this case `format` is the current length of the FORMAT parameter in bytes. The data to be printed can be distributed over one or more send areas.

FORMAT parameter

Description

The character string FORMAT contains character and format elements that are to be printed. It has the following structure:

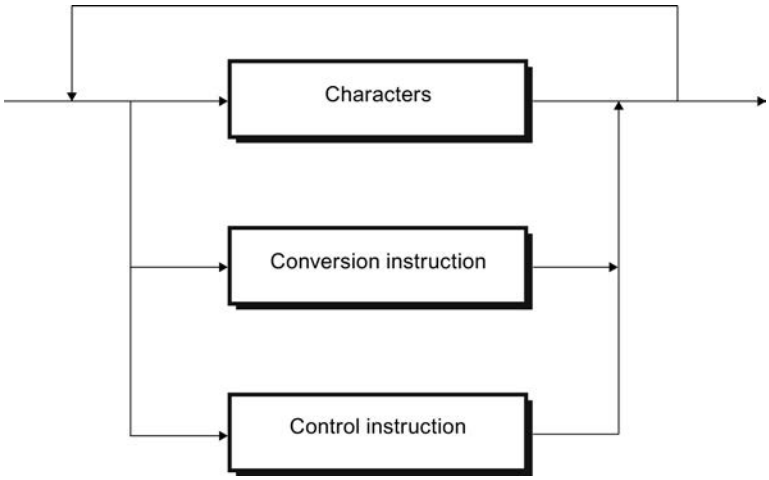


Figure 9-11 Structure of characters and format elements

Exactly one conversion instruction must exist in FORMAT for each send area SD_1 to SD_4 that is to be printed. The conversion instructions are used on the send areas SD_i according to order specified. Characters and instructions can follow each other in any order.

Characters

Permissible:

- All printable characters
- $$ (Dollar character), $' (single inverted comma),$L and $l (line feed), $P and $p (page), $R and $r (carriage return), $T and $t (tabulator)

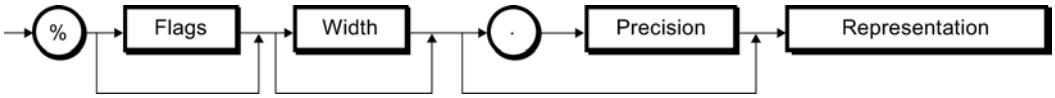


Figure 9-12 Syntax diagram of a conversion instruction

Element of a conversion instruction	Meaning	
Flags	• None: • -:	right-justified output left-justified output
Width	• None: • n:	output in standard representation Exactly n characters are output. If the output is right-justified, this may be preceded by blanks, with left- justified output the blanks come after the characters.
Precision	Precision is only relevant for representations A, D, F and R (see following table).	
	• None: • 0: • n:	Output is in standard representation no output of the decimal point or of decimal places for F and R representations • with F and R: output of the decimal point and n decimal places • with A and D (date): number of digits for the year: Possible values: 2 and 4.
Representation	The following table contains: • The possible representations • The data types possible for each representation • for each representation, the standard representation (the printer output is in standard representation if you specify no width and no precision in the FORMAT parameter.	

Conversion instruction

The following table shows the possible representations in the conversion instruction of the FORMAT parameter.

Representation	Possible data types	Example	Length	Comments
A, a	DATE	25.07.1996	10	-
	DWORD			
C, c	CHAR	K	1	-
	BYTE	M	1	
	WORD	KL	2	
	DWORD	KLMN	4	
	ARRAY of CHAR	KLMNOP	Number of characters	
	ARRAY of BYTE			
D, d	DATE	1996-07-25	10	-
	DWORD			
F, f	REAL	0.345678	8	-
	DWORD			
H, h	all data types incl. ARRAY of BYTE	Depending on data type	Depending on data type	Hexadecimal representation
I, i	INT	- 32 768	max. 6	-
	WORD	- 2 147 483 648	max. 11	

Representation	Possible data types	Example	Length	Comments
N, n	WORD	Text output	-	The relevant send area SD_i contains a reference (number) to a text to be printed. The text is on the module (for example, CP 441), which creates a printable string. If no text is found under the specified number, **** will be output.
R, r	REAL	0.12E-04	8	-
	DWORD			
S, s	STRING	Text output		-
T, t	TIME	2d_3h_10m_5s_250ms	max. 21	If an error occurs, **** will be output.
	DWORD			
U, u	BYTE	255	max. 3	-
	WORD	65 535	max. 5	
	DWORD	4 294 967 295	max. 10	
X, x	BOOL	1	1	-
	BYTE	101 ..	8	
	WORD	101 ..	16	
	DWORD	101 ..	32	
Z, z	TIME_OF_DAY (TOD)	15:38:59.874	12	-

At the points in this table at which a maximum length is specified for the representation, the actual length can of course be shorter.

Note

With the data types C and S, the following points depend on the printer being used:

- which characters can be printed
- what the printer prints for non-printable characters, unless the printer driver has a conversion table for these characters.

Control instruction

Using the control instruction you can do the following:

- Print the characters % and \
- Change the printer settings.

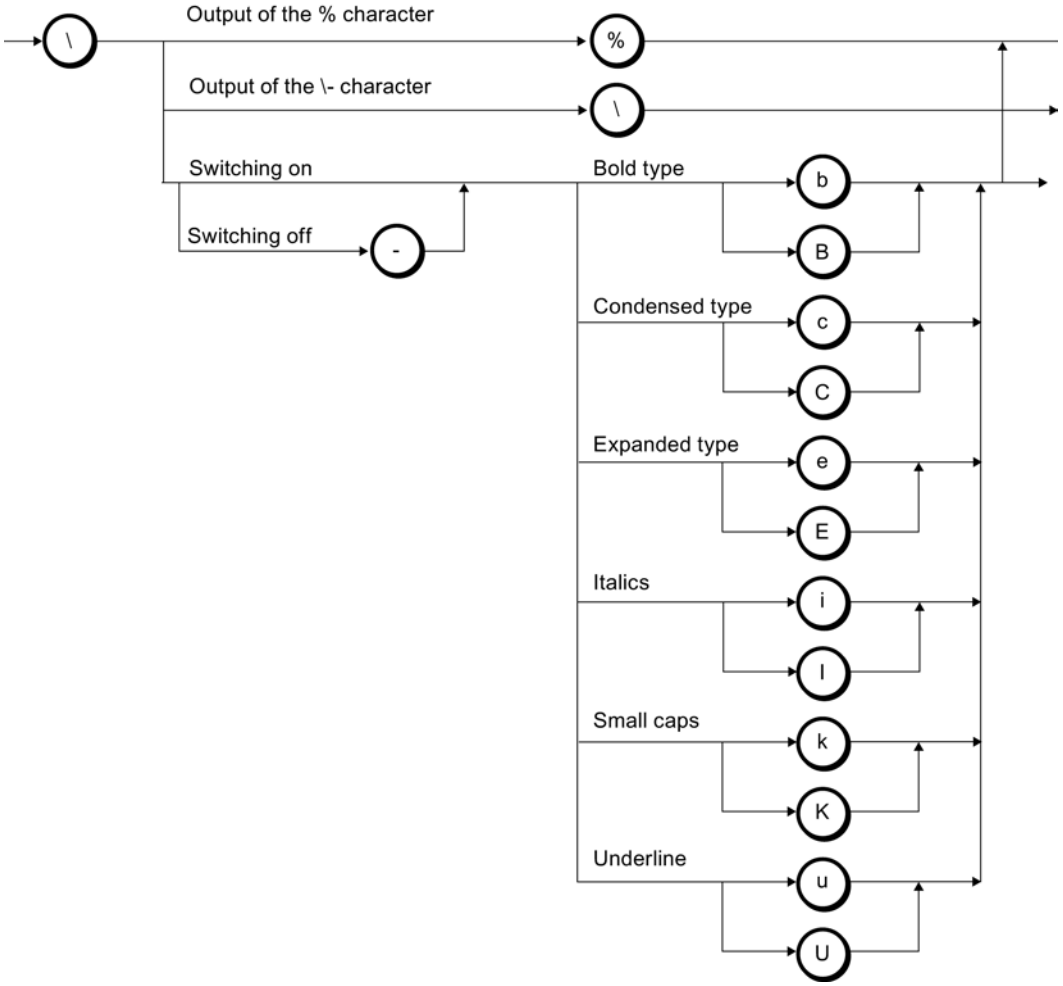


Figure 9-13 Syntax diagram of the control instruction

If you attempt to disable, for example, a font that is not enabled or execute a function that the printer does not recognize, the control instruction is ignored. The following table show the possible errors for the FORMAT in/out parameter.

Error	Printer output
Conversion instruction cannot be executed	* characters are output according to the (maximum) length of the default representation or the specified width.
Specified width too small	In the representations A, C, D, N, S, T, and Z, as many characters are printed as specified by the selected width. With all other representations, * characters are printed across the specified width.
Too many conversion instructions	The conversion instructions to which no send area pointer SD_i belongs are ignored.

Error	Printer output
Too few conversion instructions	Send areas for which there is no conversion instruction are not printed out.
Undefined or unsupported conversion instructions	****** is printed out.
Incomplete conversion instruction	****** is printed out.
Undefined or unsupported control instructions	Control instructions that do not comply with the syntax shown in the figure above are ignored.

START: Execute a warm restart or cold restart on a remote device

Description

If there is a positive edge at control input REQ, the "START" instruction activates a warm restart or a cold restart on the remote device addressed by ID**with S7-400**.

The following conditions must be met if the remote device is a CPU:

- The CPU must be in STOP mode.
- The mode selector of the CPU must be set to "RUN".

Once the warm or cold restart is complete, the device changes to RUN mode and sends a positive execution acknowledgement. The status parameter DONE is set to "1" with the evaluation of the positive acknowledgement. The status parameters ERROR and STATUS show any errors that have occurred.

Another activation of a warm or cold restart is not possible until the last activation is completed.

Parameters

The following table shows the parameters of the "START" instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter request, activates the instruction on a rising edge.
ID	Input	WORD	M, D or constant	Addressing parameter ID See also: Common parameters of instructions for S7 communication (Page 7359)
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: • 0: Job not yet started or still executing. • 1: Job has been executed error-free.

Parameters	Declaration	Data type	Memory area	Description
ERROR STATUS	Output Output	BOOL WORD	I, Q, M, D, L I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: 0000H: Neither warning nor error <> 0000H: Warning, STATUS supplies detailed information • ERROR=1 An error has occurred, STATUS supplies detailed information on the type of error.
PI_NAME	InOut	ANY	I, Q, M, D, T, C	Pointer to memory area in which the name of the program (ASCII code) to be started is located. This name must not contain more than 32 characters. With a standard system from the S7 family, it must be P_PROGRAM.
ARG	InOut	ANY	I, Q, M, D, T, C	Execution argument. • If you do not assign a value to ARG , a warm restart is run on the remote device. • If you assign the value "C," a cold restart is run on the remote device (if the remote device is capable of this type of startup).
IO_STATE	InOut	BYTE	I, Q, M, D, L	Not currently relevant. Do not assign a value to this parameter if your communication partner is an S7 series automation system.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

Parameters ERROR and STATUS

The following table contains all specific error information for START that can be output via the ERROR and STATUS parameters.

ERROR	STATUS(decimal)	Explanation
0	11	Warning: • New job not active because the previous job is still busy. • The job is already being processed in a priority class with lower priority.
0	25	Communication has started. The job is being processed.
1	1	Communications problems, for example • Connection description not loaded (local or remote). • Connection interrupted (for example: cable, CPU off, CP in STOP mode).
1	2	Negative acknowledgement from the partner device. The function cannot be executed.
1	3	The program name entered in PI_NAME is unknown.
1	4	Error in the pointers PI_NAME or ARG relating to the data length or the data type.
1	7	No complete restart possible on the partner device.

ERROR	STATUS(decimal)	Explanation
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	12	When the instruction was called • An instance DB was specified that does not belong to "START" • A global DB was specified instead of an instance DB • No instance DB found (Solution: Load the associated instance DB again).
1	20	S7-400: Not enough work memory. Solution: Reduce the program code in the memory.

STOP: Change a remote device to STOP mode

Description

If there is a rising edge at control input REQ , the "STOP" instruction activates a change to STOP mode on the remote device addressed by ID**with S7-400**. The mode change is possible when the device is in RUN, HOLD, or startup.

Successful execution of the job is indicated by the status parameter DONE having the value "1". If any errors occur, they are indicated in the status parameters ERROR and STATUS.

The mode change can only be repeated on the same remote device if the previous "STOP" instruction has been completed.

Parameters

The following table shows the parameters of the "STOP" instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter request, activates the instruction on a rising edge.
ID	Input	WORD	M, D or constant	Addressing parameter ID See also: Common parameters of instructions for S7 communication (Page 7359)
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: • 0: Job not yet started or still executing. • 1: Job has been executed error-free.
ERROR STATUS	Output Output	BOOL WORD	I, Q, M, D, L I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: 0000H: Neither warning nor error <> 0000H: Warning, STATUS supplies detailed information. • ERROR=1 There is an error. STATUS supplies detailed information on the type of error.

Parameters	Declaration	Data type	Memory area	Description
PI_NAME	InOut	ANY	I, Q, M, D	Pointer to the memory area in which the name of the program (ASCII code) to be started is located. This name must not contain more than 32 characters. With a standard system from the S7 family, it must be P_PROGRAM.
IO_STATE	InOut	BYTE	I, Q, M, D, L	Not currently relevant. Do not assign a value to this parameter if your communication partner is an S7 series automation system.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

Parameters ERROR and STATUS

The following table contains all specific error information for the "STOP" instruction that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Warning: • New job not active because the previous job is still busy. • The job is already being processed in a priority class with lower priority.
0	25	Communication has started. The job is being processed.
1	1	Communications problems, for example • Connection description not loaded (local or remote). • Connection interrupted (for example: cable, CPU off, CP in STOP mode).
1	2	Negative acknowledgement from the partner device. The function cannot be executed.
1	3	The program name entered in PI_NAME is unknown.
1	4	Error in the pointer PI_NAME relating to the data length or the data type.
1	7	The partner device is already in the STOP state.
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	12	When the instruction was called • An instance DB was specified that does not belong to STOP • A global DB was specified instead of an instance DB • No instance DB found (Solution: Load the associated instance DB again).
1	20	S7-400: Not enough work memory. Solution: Reduce the program code in the memory.

RESUME: Initiate a hot restart on a remote device

Description

If there is a rising edge at control input REQ, the "RESUME" instruction activates a hot restart in the remote device addressed by ID**with S7-400**. The following conditions must be present if the remote device is a CPU:

- The CPU must be in STOP mode.
- The mode selector of the CPU must be set to RUN.
- You must have enabled manual hot restart during configuration.
- There must be no condition preventing a hot restart.

Once the hot restart has been completed, the device changes to RUN mode and sends a positive execution acknowledgement. The status parameter DONE is set to "1" with the evaluation of the positive acknowledgement. The status parameters ERROR and STATUS show any errors that have occurred.

Another activation of the hot restart in the same remote device is not possible until the last activation is complete.

Parameters

The following table shows the parameters of the "RESUME" instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter request, activates the instruction on a rising edge.
ID	Input	WORD	M, D or constant	Addressing parameter ID See also: Common parameters of instructions for S7 communication (Page 7359)
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: • 0: Job not yet started or still executing. • 1: Job has been executed error-free.
ERROR STATUS	Output Output	BOOL WORD	I, Q, M, D, L I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: 0000H: Neither warning nor error <> 0000H: Warning, STATUS supplies detailed information. • ERROR=1 An error has occurred, STATUS supplies detailed information on the type of error.
PI_NAME	InOut	ANY	I, Q, M, D	Pointer to memory area in which the name of the program (ASCII code) to be started is located. This name must not contain more than 32 characters. With an S7, it must be P_PROGRAM.

Parameters	Declaration	Data type	Memory area	Description
ARG	InOut	ANY	I, Q, M, D, T, C	Execution argument. Not currently relevant. Do not assign a value to this parameter if your communication partner is an S7 series automation system.
IO_STATE	InOut	BYTE	I, Q, M, D, L	Not currently relevant. Do not assign a value to this parameter if your communication partner is an S7 series automation system.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

Parameters ERROR and STATUS

The following table contains all specific error information for "RESUME" that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Warning: • New job not active because the previous job is still busy. • The job is already being processed in a priority class with lower priority.
0	25	Communication has started. The job is being processed.
1	1	• Communications problems, for example, connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU off, CP in STOP mode).
1	2	Negative acknowledgement from the partner device. The function cannot be executed.
1	3	The program name entered in PI_NAME is unknown.
1	4	Error in the pointers PI_NAME or ARG relating to the data length or the data type.
1	7	Hot restart not possible
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	12	When the instruction was called • An instance DB was specified that does not belong to RESUME. • A global DB was specified instead of an instance DB • No instance DB found (Solution: Load the associated instance DB again).
1	20	S7-400: Not enough work memory. Solution: Reduce the program code in the memory.

STATUS: Query the status of a remote partner

Description

With the "STATUS" instruction, you can query the device status of a remote communication partner **with S7-400**.

If there is a rising edge at control input REQ, a job is sent to the remote partner. The reply is evaluated to determine whether problems have occurred. If no errors occurred, the received

status is copied to the tags of the PHYS, LOG and LOCAL parameters with the next call. Completion of this action is indicated by the status parameter NDR having the value "1".

You can only query the status of the same communications partner again after the last query is complete.

Parameters

The following table shows the parameters of the "STATUS" instruction:

Parameters	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter request, activates the instruction on a rising edge.
ID	Input	WORD	M, D or constant	Addressing parameter ID See also: Common parameters of instructions for S7 communication (Page 7359)
NDR	Output	BOOL	I, Q, M, D, L	Status parameter NDR: • 0: Job not yet started or still executing. • 1: Job completed without errors.
ERROR STATUS	Output Output	BOOL WORD	I, Q, M, D, L I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: 0000H: Neither warning nor error <> 0000H: Warning, STATUS supplies detailed information. • ERROR=1 There is an error. STATUS supplies detailed information on the type of error.
PHYS	InOut	ANY	I, Q, M, D	Physical status (minimum length: one byte) Possible values: • 10H fully functional. • 13H service required.
LOG	InOut	ANY	I, Q, M, D	Logical status (minimum length: one byte) Possible value: • 00H status change permitted.
LOCAL	InOut	ANY	I, Q, M, D	Status if the partner device is an S7 CPU (minimum length: two bytes)

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

In/out parameter LOCAL

If the communications partner is an S7 CPU, the in/out parameter LOCAL contains its current status: The first byte is reserved, the second byte contains an ID for the status.

Operating mode	Corresponding identifier
STOP	00H
STARTUP (warm restart)	01H
RUN	02H
STARTUP (hot restart)	03H
HOLD	04H
STARTUP (cold restart)	06H
RUN	09H
LINK-UP	0BH
UPDATE	0CH

Parameters ERROR and STATUS

The following table contains all specific error information for STATUS that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	11	Warning: • New job not active because the previous job is still busy. • The job is already being processed in a priority class with lower priority.
0	25	Communication has started. The job is being processed.
1	1	Communications problems, for example • Connection description not loaded (local or remote). • Connection interrupted (for example: cable, CPU off, CP in STOP mode).
1	2	Negative acknowledgement from the partner device. The function cannot be executed.
1	4	Error in PHYS, LOG or LOCAL relating to the data length or the data type.
1	8	Access to a remote object was rejected.
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	12	When the instruction was called • An instance DB was specified that does not belong to STATUS • A global DB was specified instead of an instance DB • No instance DB found (Solution: Load the associated instance DB again).
1	20	S7-400: Not enough work memory. Solution: Reduce the program code in the memory.

USTATUS: Uncoordinated receiving of remote device status

Description

The "USTATUS" instruction receives the device status change of a remote communication partner **with S7-400**. The partner sends its status unsolicited when a change occurs if you have configured this behavior.

If EN_R 1 is set at the control input during the call and a frame of the partner is pending, the status information is entered in the tags of the PHYS, LOG and LOCAL parameters at the next call. Completion of this action is indicated by the status parameter NDR having the value "1".

The transfer of the operating status messages must be enabled on the connection used by the "USTATUS" instruction.

Note

You can only place one instance of the "USTATUS" instruction per connection.

Parameters

The following table shows the parameters of the "USTATUS" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L	Control parameter (enabled to receive), signals ready to receive if the input is set.
ID	Input	WORD	M, D or constant	Addressing parameter ID See also: Common parameters of instructions for S7 communication (Page 7359)
NDR	Output	BOOL	I, Q, M, D, L	Status parameter NDR: • 0: Job not yet started or still executing. • 1: Job successfully completed.
ERROR STATUS	Output Output	BOOL WORD	I, Q, M, D, L I, Q, M, D, L	Status parameters ERROR and STATUS, error code: • ERROR=0 STATUS has the value: 0000H: Neither warning nor error 0000H: Warning, STATUS supplies detailed information. • ERROR=1 An error has occurred. STATUS supplies detailed information on the type of error.
PHYS	InOut	ANY	I, Q, M, D	Physical status (minimum length: one byte) Possible values: • 10H fully functional. • 13H service required.

Parameter	Declaration	Data type	Memory area	Description
LOG	InOut	ANY	I, Q, M, D	Logical status (minimum length: one byte) Possible value: • 00H status change permitted.
LOCAL	InOut	ANY	I, Q, M, D	Status if the partner device is an S7 CPU (minimum length: two bytes)

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

In/out parameter LOCAL

If the communication partner is an S7 CPU, the in/out parameter LOCAL contains its current status: The first byte is reserved, the second byte contains an ID for the status.

Operating mode	Corresponding identifier
STOP	00H
STARTUP (warm restart)	01H
RUN	02H
STARTUP (hot restart)	03H
HOLD	04H
STARTUP (cold restart)	06H
RUN	09H
LINK-UP	0BH
UPDATE	0CH

Parameters ERROR and STATUS

The following table contains all specific error information for "USTATUS" that can be output via the ERROR and STATUS parameters.

ERROR	STATUS (decimal)	Explanation
0	9	Overrun warning: an older device status has been overwritten by a more recent device status.
0	11	The received data is already being processed in a priority class with lower priority.
0	25	Communication has started. The job is being processed.
1	1	Communication problems, for example • Connection description not loaded (local or remote). • Connection interrupted (for example: cable, CPU off, CP in STOP mode).
1	4	Error in PHYS, LOG or LOCAL relating to the data length or the data type.
1	10	Access to the local user memory not possible (for example, access to a deleted DB).
1	12	When the instruction was called • An instance DB was specified that does not belong to "USTATUS" • A global DB was specified instead of an instance DB. • No instance DB found (Remedy: Load the associated instance DB again).

ERROR	STATUS (decimal)	Explanation
1	18	There is already an instance for "USTATUS" for the connection identified by ID.
1	19	The remote CPU sends data faster than it can be accepted in the user program by the instruction.
1	20	S7-400: Not enough work memory. Remedy: Reduce the program code in the memory.

Open User Communication

Overview

Open User Communication via Industrial Ethernet

The following instructions and UDTs are available for exchanging data with other communication partners by means of the user program via Ethernet (TCP, ISO-on-TCP, UDP protocols):

- Connection-oriented protocols: TCP as per RFC 793, ISO on TCP as per RFC 1006:
  - UDT 65 "TCON_PAR (Page 7419)" with the data structure for assigning connection parameters
  - UDT 651 to UDT 656 with protocol-specific defaults
  - "TCON (Page 7431)" for establishing the connection
  - "TDISCON (Page 7434)" for terminating the connection
  - "TSEND (Page 7436)" for sending data
  - "TRCV (Page 7439)" for receiving data
- Connectionless protocol: UDP as per RFC 768
  - UDT 65 "TCON_PAR (Page 7419)" with the data structure for assigning parameters of the local communications access point
  - UDT 657 with protocol-specific defaults
  - UDT 66 "TCON_ADR (Page 7421)" with the data structure for assigning addressing parameters of the remote partner
  - UDT 661 with protocol-specific defaults
  - "TCON (Page 7431)" for setting up the local communication access point
  - "TDISCON (Page 7434)" for canceling the local communication access point
  - "TUSEND (Page 7443)" for sending data
  - "TURCV (Page 7446)" for receiving data

Functional description of instructions for Open User Communication via Industrial Ethernet

Connection-oriented and connectionless protocols

The following protocol variants are distinguished in the data communication:

- Connection-oriented protocol variants:
  They establish a logical connection to the communication partner before data transmission is started. After the data transmission is complete, they then terminate the connection, if necessary. Connection-oriented protocols are used for data transmission when reliable, guaranteed delivery is of particular importance. Usually, there can be several logical connections over one physical line.
  With the instructions for Open User Communication via Industrial Ethernet, the following connection-oriented protocols are supported:
  - TCP as per RFC 793
  - ISO on TCP as per RFC 1006
- Connectionless protocol variants:
  They work without a connection. There is thus no establishment and termination of a connection with a remote partner. Unacknowledged connectionless protocols transmit the data to the remote partner and are therefore unprotected. This means the data can be lost without any indication at the block.
  With the instructions for Open User Communication via Industrial Ethernet, the following connectionless protocols are supported:
  - UDP per RFC 768 (broadcast or unicast, no support for multicast)

The functional description of the instructions depends on the protocol variant being used. This is discussed in more detail in the following section.

Receive area

This term will be used repeatedly in the following sections. This means the area in which the instruction enters the received data.

The receive area is specified by the following two variables:

- Pointer to the start of the area
- Length of the area

Depending on the protocol variant being used, the length of the area is specified either by the LEN parameter (if LEN <> 0) or the length information of the DATA parameter (if LEN = 0).

The following data types are permitted in the ANY pointer: BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TIME_OF_DAY, TIME, S5TIME, DATE_AND_TIME.

TCP

During data transmission, no information about the length or about the start and end of a message is transmitted. This is not a problem during sending because the sender knows how many data bytes it will be sending. However, the receiver has no means of detecting where one message ends in the data stream and the next one begins. We therefore recommend that you select the size of receive area of the "TRCV (Page 7439)" instruction to match the value

of the LEN parameter of the "TSEND (Page 7436)" instruction on the communication partner (number of bytes to be sent).

- Date reception in ad-hoc mode:
  The receive area is identical with the area specified by the DATA parameter of the "TRCV (Page 7439)" instruction.
  - Immediately after receiving a block of data, the "TRCV (Page 7439)" instruction transfers this data block to the receive area and sets NDR to "1". The maximum length is 8192 bytes.
  - If you have selected a receive area that is longer than the sent data, the "TRCV (Page 7439)" instruction copies all received data completely in the receive area. It then sets NDR to TRUE and writes the length of the sent data to RCVD_LEN.
  - If you have selected a receive area that is smaller than the sent data, the "TRCV (Page 7439)" instruction copies as many bytes into the receive area until it is full. It then sets NDR to TRUE and writes the length of the receive area to RCVD_LEN. With each subsequent call, you receive a further block of the sent data.
- Data reception with specified length:
  - The receive area is formed by the DATA parameters (start address of the receive area) and LEN (length of the receive area) by "TRCV (Page 7439)".
  - If data are received that do not completely fill the receive area, then these data will not be made available to you initially. They will only become available when additional data has filled up the receive area completely. Please note that in this case data from two different send jobs are located in the same receive area. If you cannot determine the end of the first message or the start of the second message, you will not be able to recognize the first or the second message.
  - If you have selected a receive area that is smaller than the sent data, the "TRCV (Page 7439)" instruction copies as many bytes into the receive area until it is completely full.
  - If "TRCV (Page 7439)" has completely filled the receive area, it will set NDR to TRUE and describes RCVD_LEN with the value of LEN. Thus with each subsequent call, you get an additional block of the sent data.

ISO on TCP

During data transfer, information on the length and end of a message is also transferred.

If you have selected a receive area that is longer than the sent data, then the "TRCV (Page 7439)" instruction enters all the sent data completely in the receive area. It then sets NDR to TRUE and writes the length of the sent data to RCVD_LEN.

If you have selected a receive area shorter than the length of the sent data, then "TRCV (Page 7439)" will not copy any data to the receive area and returns the following error information: ERROR=1, STATUS=W#16#8088.

UDP

There is no connection establishment as is the case with the TCP and ISO on TCP protocols. You must therefore specify a reference to the address parameters of the recipient (IP address and port no.) when you call the send block "TUSEND (Page 7443)". In the same way, on

completion of the receive block "TURCV (Page 7446)", you will receive a reference to the address parameters of the sender (IP address and port no.).

- To use the "TUSEND (Page 7443)" and "TURCV (Page 7446)" instructions, you must first call the "TCON (Page 7431)" instruction both at the sender and receiver end to set up the local communication access point.
- Each time you call "TUSEND (Page 7443)", you can reference the remote partner once again by specifying its IP address and its port number.
- During data transfer, information on the length and end of a message is also transferred.

If you have selected a receive area that is longer than the sent data, "TURCV (Page 7446)" will copy all sent data completely to the receive area. It then sets NDR to TRUE and writes the length of the sent data to RCVD_LEN.

If you have selected a receive area shorter than the length of the sent data, TURCV does not copy any data to the receive area and returns the following error information: ERROR = 1, STATUS = W#16#8088.

Port numbers for TCP and UDP

The address assignment for a connection is made as follows:

- With TCP:
  - Active connection: Via the remote IP address, the remote port and the local port (specify the port number of the local port explicitly or let the operating system set it).
  - Passive connection: Vie the local port. If you want to use the local port several times, you have to either predefine the remote IP address or the remote IP address together with the remote port.
- With UDP: Via the remote IP address, the remote port and the local port (specify the port number of the local port explicitly).

The areas listed below apply to the local port. These areas apply to the remote port of the remote partner is an S7 CPU. Otherwise, there are no restrictions for the remote port.

- For CPUs 31x-2 PN/DP up to and including firmware version V2.6 and CPUs 41x-3 PN/DP up to and including firmware version V5.1:
  - Permissible port numbers in the configuration (UDT 65): 2000 to 5000
  - The operating system assigns a port from the dynamic range of numbers between 49152 and 65534 if a port number is not assigned explicitly.
- For CPUs 31x-2 PN/DP as of firmware version V2.7 and CPUs 41x-3 PN/DP as of firmware version V5.2:
  - Permissible port numbers in the configuration (UDT 65): 1 to 49151
    We recommend restricting the port numbers from 2000 to 5000, since Siemens reserves ports from the number ranges 1 to 1999 and 5001 to 49151 for use by the system.
  - The operating system assigns a port from the dynamic range of numbers between 49152 and 65534 if a port number is not assigned explicitly.
- For CPUs 31x-2 PN/DP as of firmware version V3.2, CPUs 41x-3 PN/DP as of firmware version V6.0 and CPUs 41x-5H PN/DP as of firmware version V6.0:
  - ·Using this port several times is possible.

You can find the port numbers reserved by the system in the following list:

Protocol	Port number	Service
-	0	(This port is reserved. It cannot be used.)
TCP	20 and 21	FTP
TCP	25	SMTP
TCP	80	HTTP
TCP	102	RFC 1006
UDP	135	RPC-DCOM
UDP	161	SNMP REQUEST
TCP, UDP	34962	PNIO
TCP, UDP	34963	PNIO
TCP, UDP	34964	PNIO
UDP	65532	NTP
UDP	65533	NTP
UDP	65534	NTP
UDP	65535	NTP

Also see http://www.iana.org/assignments/port-numbers.

Note

Do not use port numbers reserved for the system.

Assigning parameters for communications connections with TCP and ISO on TCP

Data block for assigning parameters

To be able to assign parameters for TCP and ISO on TCP for communication connections, create a DB that contains the data structure from the UDT 65 "TCON_PAR". This data structure contains the parameters necessary for configuring the connection. You will need such a data structure for every connection. You can assemble this structure in a shared DB.

The CONNECT connection parameter of the "TCON (Page 7431)" instruction contains a reference to the address of the associated connection description (e.g. P#DB100.DBX0.0 byte 64).

Form of the connection description (UDT 65)

Byte	Parameters	Data type	Start value	Description
0 to 1	block_length	WORD	W#16#40	Length of UDT 65: 64 bytes (fixed)
2 to 3	id	WORD	W#16#0001	Reference to the connection (range of values: W#16#0001 to W#16#0FFF) You must specify the value of this parameter in the respective block for the ID.
4	connection_type	BYTE	B#16#11	Protocol variant: • B#16#11: TCP • B#16#12: ISO on TCP • B#16#01: TCP (compatibility mode)
5	active_est	BOOL	FALSE	ID for the manner in which the connection is established: • FALSE: Passive connection establishment • TRUE: Active connection establishment
6	local_device_id	BYTE	B#16#02	• B#16#00: Communication via CP 443-1EX (only with S7-400 and connection_type = B#16#12) Permitted CPs: CP443-1EX4x, CP443-1EX20, CP443-1GX20, CP443-1EX30, CP443-1GX30 • B#16#01: Communication via IE interface on interface slot 1 (IF1) for WinAC RTX (only TCP) • B#16#02: Communication over the integrated IE interface with CPUs 315-2 PN/DP and 317-2 PN/DP • B#16#03: Communication via the integrated IE interface for CPU 319-3 PN/DP • B#16#05: Communication via the integrated IE interface with CPUs 414-3 PN/DP, 416-3 PN/DP, 416-3F PN/DP and 41x-5H PN/DP (Rack 0) • B#16#06: Communication via IE interface on interface slot 2 (IF2) for WinAC RTX (only TCP) • B#16#0B: Communication via IE interface on interface slot 3 (IF3) for WinAC RTX (only TCP) • B#16#0F: Communication via IE interface on interface slot 4 (IF4) for WinAC RTX (only TCP) • B#16#15: Communication via the integrated IE interface for CPUs 41x-5H PN/DP (Rack 1)
7	local_tsap_id_len	BYTE	B#16#02	Length of parameter local_tsap_id used; possible values: • 0 or 2, for connection_type = B#16#01 (active end: 0, passive end: 2) • 0 or 2, for connection_type = B#16#11 (active end: 0 or 2, passive end: 2) • 2 to 16. if connection_type = B#16#12
8	rem_subnet_id_len	BYTE	B#16#00	This parameter is currently not used. Assign with B#16#00.

Byte	Parameters	Data type	Start value	Description
9	rem_staddr_len	BYTE	B#16#00	Length of the address of the remote connection end point: • 0: unspecified, i.e. parameter rem_staddr is irrelevant. • 4: Valid IP address in the rem_staddr parameter
10	rem_tsap_id_len	BYTE	B#16#00	Length of parameter rem_tsap_id used; possible values: • 0 or 2, for connection_type = B#16#01 For the passive end only value B#16#00 is permitted. • 0 or 2. if connection_type = B#16#11 • 0 or 2 to 16, if connection_type = B#16#12 (active end: 2 to 16; passive end: 0 or 2 to 16, 0 means unspecified)
11	next_staddr_len	BYTE	B#16#00	length of the next_staddr parameter used (only relevant for ISO-on-TCP)
12 to 27	local_tsap_id	ARRAY [1..16] of BYTE	B#16#07 B#16#D0 B#16#00 ...	With connection_type = • B#16#11: Local port no. (possible values see: Functional description of instructions for Open User Communication via Industrial Ethernet (Page 7411)), local_tsap_id[1] = high byte of the port no. in hexadecimal, local_tsap_id[2] = low byte of the port number in hexadecimal, local_tsap_id[3-16] = B#16#00 • B#16#12: local TSAP ID: see below. • B#16#01: Local port no. (possible values see: Functional description of instructions for Open User Communication via Industrial Ethernet (Page 7411)), local_tsap_id[1] = low byte of the port no. in hexadecimal, local_tsap_id[2] = high byte of the port number in hexadecimal, local_tsap_id[3-16] = B#16#00 Note: If there are multiple connections to the same communication partner, you have to ensure that each value of local_tsap_id that you use in your CPU is unique.
28 to 33	rem_subnet_id	ARRAY [1..6] of BYTE	B#16#00 ...	This parameter is currently not used. Assign 0 to it.

Byte	Parameters	Data type	Start value	Description
34 to 39	rem_staddr	ARRAY [1..6] of BYTE	B#16#00 ...	IP address of the remote connection end point, for example, 192.168.0.1: With connection_type = • B#16#1x: rem_staddr[1] = B#16#C0 (192), rem_staddr[2] = B#16#A8 (168), rem_staddr[3] = B#16#00 (0), rem_staddr[4] = B#16#01 (1), rem_staddr[5-6]= B#16#00 (reserved) • B#16#01: rem_staddr[1] = B#16#01 (1), rem_staddr[2] = B#16#00 (0), rem_staddr[3] = B#16#A8 (168), rem_staddr[4] = B#16#C0 (192), rem_staddr[5-6]= B#16#00 (reserved)
40 to 55	rem_tsap_id	ARRAY [1..16] of BYTE	B#16#00 ...	With connection_type = • B#16#11: remote port no. (possible values see: Functional description of instructions for Open User Communication via Industrial Ethernet (Page 7411)), rem_tsap_id[1] = high byte of the port no. in hexadecimal, rem_tsap_id[2] = low byte of the port number in hexadecimal, rem_tsap_id[3-16] = B#16#00 • B#16#12: remote TSAP ID: see below. • B#16#01: remote port no. (possible values see: Functional description of instructions for Open User Communication via Industrial Ethernet (Page 7411)), local_tsap_id[1] = low byte of the port no. in hexadecimal, local_tsap_id[2] = high byte of the port number in hexadecimal, local_tsap_id[3-16] = B#16#00
56 to 61	next_staddr	ARRAY [1..6] of BYTE	B#16#00 ...	With local_device_id = • B#16#00: next_staddr[1]: Rack and slot of the associated (local) CP (Bits 0 to 4: slot, bits 5 to 7: rack number) next_staddr[2-6]: B#16#00 • B#16#02, B#16#03, B#16#05: next_staddr[1-6]: B#16#00 • B#16#01, B#16#06, B#16#0B, B#16#0F: next_staddr[1-6]: B#16#00 (In this case, next_staddr_len=B#16#00)
62 to 63	spare	WORD	W#16#0000	Standby: Assign "0" to this parameter.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

Length and form of the local and remote TSAP ID for ISO on TCP

- Active connection establishment:
  - Remote TSAP-ID:
    All character strings with a length between 1 and 16 bytes are permitted. Each character can have a value between B#16#00 and B#16#FF.
  - Remote TSAP-ID:
    For integrated IE interfaces, all character strings with a length between 1 and 16 bytes are permitted. Each character can have a value between B#16#00 and B#16#FF.
- Passive connection establishment:
  - Remote TSAP-ID:
    All character strings with a length between 0 and 16 bytes are permitted. Each character can have a value between B#16#00 and B#16#FF.
  - Local TSAP-ID:

loc_tsap_id_len	local_tsap_id[1]	local_tsap_id[2]	local_tsap_id[3 bis 16]
2	B#16#E0 (connection type T connection)	0 (only for integrated IE interfaces) or rack and slot of the local CPU (bits 0 to 4 slot, bits 5 to 7 rack number)	Does not exist
> 2	B#16#E0 (connection type T connection)	0 (only for integrated IE interfaces) or rack and slot of the local CPU (bits 0 to 4 slot, bits 5 to 7 rack number)	TSAP extension
	Only for integrated IE interfaces: a ASCII character (B#16#20 to B#16#7E)	irrelevant	TSAP extension

CPU dependencies for protocol variants TCP and ISO on TCP

To determine which protocol variants (TCP and ISO on TCP) can be used on which CPU, refer to the following paragraph:

Relationship between CPU and protocol variant used (connection_type) and transferable data length (Page 7422).

For information on the number of possible connections, please refer to the specifications for your CPU.

Connection establishment

A communication partner A must trigger the active connection establishment. A communication partner B must trigger the passive connection establishment. If both communication partners have triggered their connection establishment then the operating system can completely establish the communication connection.

When assigning parameters to the connection specify which communication partner will activate the connection establishment and which will execute a passive communication establishment on the request of the communication partner.

In UDP, both partners must initiate passive connection establishment.

Assigning parameters for the local communications access point with UDP

Data structure for configuring the local communications access point

To assign parameters for the local communications access point, create a DB that contains the data structure from the UDT 65 "TCON_PAR". This data structure contains the parameters necessary for configuring the connection between the user program and the communications layer of the operating system.

The CONNECT parameter of the "TCON (Page 7431)" instruction contains a reference to the address of the associated connection description (e.g. P#DB100.DBX0.0 byte 64).

Form of the connection description for UDP (UDT 65)

Byte	Parameters	Data type	Initial value	Description
0 to 1	block_length	WORD	W#16#40	Length of UDT 65: 64 bytes (fixed)
2 to 3	id	WORD	W#16#0001	Reference to this connection between the user program and the communications level of the operating system (range of values: W#16#0001 to W#16#0FFF) You must specify the value of this parameter in the respective block for the ID.
4	connection_type	BYTE	B#16#13	Protocol variant: • B#16#13: UDP
5	active_est	BOOL	FALSE	ID for the manner in which the connection is established: You must assign FALSE to this parameter.
6	local_device_id	BYTE	B#16#02	• B#16#01: Communication via IE interface on interface slot 1 (IF1) for WinAC RTX • B#16#02: Communication via the integrated IE interface for CPU 317-2 PN/DP • B#16#03: Communication via the integrated IE interface for CPU 319-3 PN/DP • B#16#05: Communication via the integrated IE interface with CPUs 414-3 PN/DP, 416-3 PN/DP, 416-3F PN/DP and 41x-5H PN/DP (Rack 0) • B#16#06: Communication via IE interface on interface slot 2 (IF2) for WinAC RTX • B#16#0B: Communication via IE interface on interface slot 3 (IF3) for WinAC RTX • B#16#0F: Communication via IE interface on interface slot 4 (IF4) for WinAC RTX • B#16#15: Communication via the integrated IE interface for CPUs 41x-5H PN/DP (Rack 1)
7	local_tsap_id_len	BYTE	B#16#02	Length of local_tsap_id parameter (local port) used; 2 bytes

Byte	Parameters	Data type	Initial value	Description
8	rem_subnet_id_len	BYTE	B#16#00	This parameter is not used. Assign with B#16#00.
9	rem_staddr_len	BYTE	B#16#00	This parameter is not used. Assign with B#16#00.
10	rem_tsap_id_len	BYTE	B#16#00	This parameter is not used. Assign with B#16#00.
11	next_staddr_len	BYTE	B#16#00	This parameter is not used. Assign with B#16#00.
12 to 27	local_tsap_id	ARRAY [1..16] of BYTE	B#16#07 B#16#D0 B#16#00 ...	Local port no. (possible values see: Functional description of instructions for Open User Communication via Industrial Ethernet (Page 7411)), local_tsap_id[1] = high byte of the port no. in hexadecimal, local_tsap_id[2] = low byte of the port number in hexadecimal, local_tsap_id[3-16] = B#16#00 (reserved) Note: Ensure that each value of local_tsap_id that you use in your CPU is unique.
28 to 33	rem_subnet_id	ARRAY [1..6] of BYTE	B#16#00 ...	This parameter is not used. Assign "0" to it.
34 to 39	rem_staddr	ARRAY [1..6] of BYTE	B#16#00 ...	This parameter is not used. Assign "0" to it.
40 to 55	rem_tsap_id	ARRAY [1..16] of BYTE	B#16#00 ...	This parameter is not used. Assign "0" to it.
56 to 61	next_staddr	ARRAY [1..6] of BYTE	B#16#00 ...	This parameter is not used. Assign "0" to it.
62 to 63	spare	WORD	W#16#0000	Standby: Assign "0" to this parameter.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

CPU dependency of the UDP protocol variant

To determine which CPUs are suitable for use with the UDP protocol variant, refer to the following paragraph:

Relationship between CPU and protocol variant used (connection_type) and transferable data length (Page 7422)

For information on the number of possible connections between the user program and the communications layer of the operating system, please refer to the specifications for your CPU.

Configuring the local communications access point

Each communications partner must configure its local communications point independently of the other partner. This pertains to establishing the connection between the user program and communications layer of the operating system.

In UDP, both partners must initiate passive connection establishment.

Structure of the address information for the remote partner with UDP

Overview

- With "TUSEND (Page 7443)", you transfer the address information of the receiver at the ADDR parameter. This address information must have the structure specified below.
- With "TURCV (Page 7439)", you receive the address of the sender of the received data at the ADDRparameter. This address information must have the structure specified below.

Data block for the address information of the remote partner

You must create a DB that contains one or more data structures as per UDT 66 "TADDR_PAR".

In parameter ADDR of "TUSEND (Page 7443)", you transfer a pointer to the address of the associated remote partner (e.g. P#DB100.DBX0.0 byte 8). This pointer is received in the ADDRparameter of "TURCV (Page 7439)".

Form of the address information for the remote partner (UDT 66)

Byte	Parameter	Data type	Initial value	Description
0 to 3	rem_ip_addr	ARRAY [1..4] of BYTE	B#16#00 ...	IP address of the remote partner, e.g. 192.168.2.3: • rem_ip_addr[1] = B#16#C0 (192) • rem_ip_addr[2] = B#16#A8 (168) • rem_ip_addr[3] = B#16#02 (2) • rem_ip_addr[4] = B#16#03 (3)
4 to 5	rem_port_nr	ARRAY [1..2] of BYTE	B#16#00 ...	remote port no. (possible values see: Functional description of instructions for Open User Communication via Industrial Ethernet (Page 7411)) • rem_port_nr[1] = high byte of the port no. in hexadecimal notation • rem_port_nr[2] = low byte of port no. in hexadecimal notation
6 to 7	spare	ARRAY [1..2] of BYTE	B#16#00 ...	Standby: Assign "0" to this parameter.

Relationship between CPU and protocol variant used (connection_type) and transferable data length

Relationship between CPU and protocol variant used (connection_type) and transferable data length

The table below shows which protocol variants you can use on which CPU, and which data lengths you can transfer.

Protocol variant	Parameter "connection _type" in UDT 65	CPU	Data length in bytes for communication over integrated IE interface	Data length in bytes for communication via CP
TCP	B#16#11	CPUs 31x-2 PN/DP as of firmware version V2.4	1 to 8192	-
		CPUs 31x-2 PN/DP as of firmware version V3.1	1 to 32767	
		CPUs 31x-3 PN/DP as of firmware version V3.2	1 to 32767	
		CPUs 41x-3 PN/DP	1 to 32767	-
		CPUs 41x-5H PN/DP	1 to 32767	-
	B#16#01	CPUs 31x-2 PN/DP	1 to 1472	-
ISO on TCP	B#16#12	CPUs 31x-2 PN/DP as of firmware version V2.4	1 to 8192	-
		CPUs 31x-2 PN/DP as of firmware version V3.1	1 to 32767	
		CPUs 31x-3 PN/DP as of firmware version V3.2	1 to 32767	
		CPUs 41x as of firmware version V4.1 (without CPUs 41x-4H)	-	1 to 1452
		CPUs 41x-3 PN/DP	1 to 32767	1 to 1452
		CPUs 41x-5H PN/DP	1 to 32767	1 to 1452
UDP	B#16#13	CPUs 31x-2 PN/DP as of firmware version V2.4, CPUs 41x-3 PN/DP and 41x-5H PN/DP	1 to 1472	-

Examples of parameters for communications connections

Example 1: Communication via ISO on TCP and CP 443-1 between two S7-400 CPUs

The two communication partners are two CPUs 414-2. The communication is handled by two CPs 443-1.

The following table shows the most important data for both communications partners:

Property	Communication partner A: CPU 414-2 with CP 443-1	Communication partner B: CPU 414-2 with CP 443-1
Connection establishment	active	passive
IP address	192.168.4.14	192.168.4.16
Physical address of the CPU	Rack 0, slot 3	Rack 0, slot 4
Physical address of the associated CP	Rack 0, slot 6	Rack 1, slot 8
Local TSAP-ID (Note: the coding of the actual TSAP to distinguish the connection occurs as of the third byte)	0xE0 03 54 43 50 2D 31	0xE0 04 54 43 50 2D 31

The following table shows the parameter entries in the DB relevant for active establishment of a connection by communication partner A:

Parameters	Data type	Value in example	Description
id	WORD	W#16#0414	Reference to this connection
connection_type	BYTE	B#16#12	Protocol variant: ISO on TCP
active_est	BOOL	TRUE	Active connection establishment
local_device_id	BYTE	B#16#00	Communication AS-internal via CP
local_tsap_id_len	BYTE	B#16#07	Length of parameter local_tsap_id
rem_staddr_len	BYTE	B#16#04	Length of the address of the remote connection end point: • 4: Valid IP address in the rem_staddr parameter
rem_tsap_id_len	BYTE	B#16#07	Length of parameter rem_tsap_id
next_staddr_len	BYTE	B#16#01	Length of parameter next_staddr
local_tsap_id	ARRAY [1..16] of BYTE	• local_tsap_id[1] = B#16#E0 • local_tsap_id[2] = B#16#03 • local_tsap_id[3] = B#16#54 (ASCII equivalent of "T") • local_tsap_id[4] = B#16#43 (ASCII equivalent of "C") • local_tsap_id[5] = B#16#50 (ASCII equivalent of "P") • local_tsap_id[6] = B#16#2D (ASCII equivalent of "-") • local_tsap_id[7] = B#16#31 (ASCII equivalent of "1") • local_tsap_id[8-16] = B#16#00	local TSAP-ID: 0xE0035443502D31
rem_staddr	ARRAY [1..6] of BYTE	"192.168.4.16" • rem_staddr[1] = B#16#C0 (192) • rem_staddr[2] = B#16#A8 (168) • rem_staddr[3] = B#16#04 (4) • rem_staddr[4] = B#16#10 (16) • rem_staddr[5-6] = B#16#00	IP address of the remote connection end point

Parameters	Data type	Value in example	Description
rem_tsap_id	ARRAY [1..16] of BYTE	• rem_tsap_id[1] = B#16#E0 • rem_tsap_id[2] = B#16#04 • rem_tsap_id[3] = B#16#54 (ASCII equivalent of "T") • rem_tsap_id[4] = B#16#43 (ASCII equivalent of "C") • rem_tsap_id[5] = B#16#50 (ASCII equivalent of "P") • rem_tsap_id[6] = B#16#2D (ASCII equivalent of "-") • rem_tsap_id[7] = B#16#31 (ASCII equivalent of "1") • rem_tsap_id[8-16] = B#16#00	remote TSAP-ID: 0xE0045443502D31
next_staddr	ARRAY [1..6] of BYTE	• next_staddr[1] = B#16#06 • next_staddr[2-6] = B#16#00	Rack = 0, slot = 6 (bits 7 to 5: Rack no., bits 4 to 0: slot no.)

The following table shows the parameter entries in the DB relevant for passive establishment of a connection by communication partner B:

Parameters	Data type	Value in example	Description
id	WORD	W#16#0416	Reference to this connection
connection_type	BYTE	B#16#12	Protocol variant: ISO on TCP
active_est	BOOL	FALSE	Passive connection establishment
local_device_id	BYTE	B#16#00	Communication AS-internal via CP
local_tsap_id_len	BYTE	B#16#07	Length of parameter local_tsap_id used
rem_staddr_len	BYTE	B#16#04	Length of the address of the remote connection end point: • 4: Valid IP address in the rem_staddr parameter
rem_tsap_id_len	BYTE	B#16#07	Length of parameter rem_tsap_id
next_staddr_len	BYTE	B#16#01	Length of parameter next_staddr
local_tsap_id	ARRAY [1..16] of BYTE	• local_tsap_id[1] = B#16#E0 • local_tsap_id[2] = B#16#04 • local_tsap_id[3] = B#16#54 (ASCII equivalent of "T") • local_tsap_id[4] = B#16#43 (ASCII equivalent of "C") • local_tsap_id[5] = B#16#50 (ASCII equivalent of "P") • local_tsap_id[6] = B#16#2D (ASCII equivalent of "-") • local_tsap_id[7] = B#16#31 (ASCII equivalent of "1") • local_tsap_id[8-16] = B#16#00	local TSAP-ID: 0xE0045443502D31

Parameters	Data type	Value in example	Description
rem_staddr	ARRAY [1..6] of BYTE	"192.168.4.14" • rem_staddr[1] = B#16#C0 (192) • rem_staddr[2] = B#16#A8 (168) • rem_staddr[3] = B#16#04 (4) • rem_staddr[4] = B#16#0E (14) • rem_staddr[5-6] = B#16#00	IP address of the remote connection end point
rem_tsap_id	ARRAY [1..16] of BYTE	• rem_tsap_id[1] = B#16#E0 • rem_tsap_id[2] = B#16#03 • rem_tsap_id[3] = B#16#54 (ASCII equivalent of "T") • rem_tsap_id[4] = B#16#43 (ASCII equivalent of "C") • rem_tsap_id[5] = B#16#50 (ASCII equivalent of "P") • rem_tsap_id[6] = B#16#2D (ASCII equivalent of "-") • rem_tsap_id[7] = B#16#31 (ASCII equivalent of "1") • rem_tsap_id[8-16] = B#16#00	remote TSAP-ID: 0xE0035443502D31
next_staddr	ARRAY [1..6] of BYTE	• next_staddr[1] = B#16#28 • next_staddr[2-6] = B#16#00	Rack = 1, slot = 8 (bits 7 to 5: Rack no., bits 4 to 0: slot no.)

Example 2: Communication over TCP and integrated PROFINET interfaces between an S7-400 CPU and an S7-300 CPU

One partner is a CPU 416-3 PN/DP, the other is a CPU 319-3 PN/DP. The following table shows the most important data for both communications partners:

Property	Communication partner A: CPU 416-3 PN/DP	Communication partner B: CPU 319-3 PN/DP
Connection establishment	active	passive
IP address	192.168.3.142	192.168.3.125
Local port no.	irrelevant	2005

The following table shows the parameter entries in the DB relevant for active establishment of a connection by communication partner A:

Parameters	Data type	Value in example	Description
id	WORD	W#16#0014	Reference to this connection
connection_type	BYTE	B#16#11	Protocol variant: TCP
active_est	BOOL	TRUE	Active connection establishment
local_device_id	BYTE	B#16#05	Communication via the integrated Ethernet interface with S7-400 CPUs
local_tsap_id_len	BYTE	B#16#00 (only this value is possible)	Parameter local_tsap_id is not used

Parameters	Data type	Value in example	Description
rem_staddr_len	BYTE	B#16#04	Length of the address of the remote connection end point: • 4: Valid IP address in the rem_staddr parameter
rem_tsap_id_len	BYTE	B#16#02 (only this value is possible)	Length of parameter rem_tsap_id
rem_staddr	ARRAY [1..6] of BYTE	"192.168.3.125" • rem_staddr[1] = B#16#C0 (192) • rem_staddr[2] = B#16#A8 (168) • rem_staddr[3] = B#16#03 (3) • rem_staddr[4] = B#16#7D (125) • rem_staddr[5-6] = B#16#00	IP address of the remote connection end point
rem_tsap_id	ARRAY [1..16] of BYTE	"2005" • rem_tsap_id[1] = B#16#07 • rem_tsap_id[2] = B#16#D5 • rem_tsap_id[3-16] = B#16#00	Remote port no.: 2005 = W#16#07D5

The following table shows the parameter entries in the DB relevant for passive establishment of a connection by communication partner B:

Parameters	Data type	Value in example	Description
id	WORD	W#16#000F	Reference to this connection
connection_type	BYTE	B#16#11	Protocol variant: TCP
active_est	BOOL	FALSE	Passive connection establishment
local_device_id	BYTE	B#16#03	Communication via the integrated Ethernet interface with the CPU 319-3 PN/DP
local_tsap_id_len	BYTE	B#16#02 (only this value is possible)	Length of parameter local_tsap_id
rem_staddr_len	BYTE	B#16#04	Length of the address of the remote connection end point: • 4: Valid IP address in the rem_staddr parameter
rem_tsap_id_len	BYTE	B#16#00 (only this value is possible)	Length of parameter rem_tsap_id
local_tsap_id	ARRAY [1..16] of BYTE	"2005" • local_tsap_id[1] = B#16#07 • local_tsap_id[2] = B#16#D5 • local_tsap_id[3-16] = B#16#00	Local port no.: 2005 = W#16#07D5
rem_staddr	ARRAY [1..6] of BYTE	"192.168.3.142" • rem_staddr[1] = B#16#C0 (192) • rem_staddr[2] = B#16#A8 (168) • rem_staddr[3] = B#16#03 (3) • rem_staddr[4] = B#16#8E (142) • rem_staddr[5-6] = B#16#00	IP address of the remote connection end point

Example 3: Communication over UDP and integrated PROFINET interfaces between an S7-300 CPU and an S7-400 CPU

One partner is a CPU 319-3 PN/DP, the other is a CPU 414-3 PN/DP. The following table shows the most important data for both communications partners:

Property	Communication partner A: CPU 319-3 PN/DP	Communication partner B: CPU 414-3 PN/DP
Sender/receiver	Sender	Receiver
IP address	192.168.3.142	192.168.3.125
Local port no.	2004	2005

The following table shows the parameter entries in the DB relevant for the sender (communication partner A) for assigning parameters to the local communications access point:

Parameter	Data type	Value in example	Description
id	WORD	W#16#0014	Reference to this connection between the application program and the communication layer of the operating system.
connection_type	BYTE	B#16#13	Protocol variant: UDP
active_est	BOOL	FALSE	Only this value can be used with the protocol variant UDP.
local_device_id	BYTE	B#16#03	Communication via the integrated Ethernet interface with the CPU 319-3 PN/DP
local_tsap_id_len	BYTE	B#16#02	Length of parameter local_tsap_id
local_tsap_id	ARRAY [1..16] of BYTE	• local_tsap_id[1] = B#16#07 • local_tsap_id[2] = B#16#D4 • local_tsap_id[3-16] = B#16#00	Local port no.: 2004 = W#16#07D4

The following table shows the parameter entries in the DB relevant for the receiver (communication partner B) for assigning parameters to the local communications access point:

Parameter	Data type	Value in example	Description
id	WORD	W#16#000F	Reference to this connection between the application program and the communication layer of the operating system.
connection_type	BYTE	B#16#13	Protocol variant: UDP
active_est	BOOL	FALSE	Only this value can be used with the protocol variant UDP.
local_device_id	BYTE	B#16#05	Communication via the integrated Ethernet interface with S7-400 CPUs

Parameter	Data type	Value in example	Description
local_tsap_id_len	BYTE	B#16#02	Length of parameter local_tsap_id
local_tsap_id	ARRAY [1..16] of BYTE	• local_tsap_id[1] = B#16#07 • local_tsap_id[2] = B#16#D5 • local_tsap_id[3-16] = B#16#00	Local port no.: 2005 = W#16#07D5

When "TUSEND (Page 7443)" is called on the sender, you transfer the following address parameters of the receiver in a DB:

Parameters	Data type	Value in example	Description
rem_ip_addr	ARRAY [1..4] of BYTE	• rem_ip_addr[1] = B#16#C0 (192) • rem_ip_addr[2] = B#16#A8 (168) • rem_ip_addr[3] = B#16#3 (3) • rem_ip_addr[4] = B#16#7D (125)	IP address of the receiver: 192.168.3.125
rem_port_nr	ARRAY [1..2] of BYTE	• rem_port_nr[1] = B#16#07 • rem_port_nr[2] = B#16#D5	Port no. of the receiver: 2005 = W#16#07D5

When "TURCV (Page 7446)" is called on the receiver, you receive the following address parameters of the sender in a DB:

Parameters	Data type	Value in example	Description
rem_ip_addr	ARRAY [1..4] of BYTE	• rem_ip_addr[1] = B#16#C0 (192) • rem_ip_addr[2] = B#16#A8 (168) • rem_ip_addr[3] = B#16#3 (3) • rem_ip_addr[4] = B#16#8E (142)	IP address of the sender: 192.168.3.142
rem_port_nr	ARRAY [1..2] of BYTE	• rem_port_nr[1] = B#16#07 • rem_port_nr[2] = B#16#D4	Port no. of the sender: 2004 = W#16#07D4

Example 4: Communication over ISO on TCP and integrated PROFINET interfaces between two S7-400 CPUs

One of the communication partners is a CPU 414-3 PN/DP as of firmware version V5.0, the other is a CPU 416-3 PN/DP with firmware version V5.0. Communication is handled over the integrated PROFINET interfaces.

The following table shows the most important data of the two communication partners.

Property	Communication partner A: CPU 414-3 PN/DP	Communication partner B: CPU 416-3 PN/DP
Sender/receiver	Sender	Receiver
Connection establishment	passive	active
IP address	192.168.0.14	192.168.0.16

The following table shows all parameter entries of communication partner A in the corresponding DB.

Parameter	Data type	Value in example	Description
block_length	WORD	W#16#40	Length of UDT65: 64 bytes (fixed)
id	WORD	W#16#0001	Reference to this connection
connection_type	BYTE	B#16#12	Protocol variant: • B#16#12: ISO on TCP
active_est	BOOL	FALSE	Passive connection establishment
local_device_id	BYTE	B#16#05	Communication via the integrated IE interface for CPUs 41x PN/DP
local_tsap_id_len	BYTE	B#16#03	Length of parameter local_tsap_id
rem_subnet_id_len	BYTE	B#16#00	This parameter is not used. Assign with B#16#00.
rem_staddr_len	BYTE	B#16#04	Length of the address of the remote connection end point: • 4: Valid IP address in the rem_staddr parameter
rem_tsap_id_len	BYTE	B#16#03	Length of parameter rem_tsap_id
next_staddr_len	BYTE	B#16#00	This parameter is not used (because communication is over the integrated interface). Assign with B#16#00 .
local_tsap_id	ARRAY [1..16] of BYTE	• local_tsap_id[1] = B#16#E0 • local_tsap_id[2] = B#16#03 • local_tsap_id[3] = B#16#01 • local_tsap_id[4-16] = B#16#00	local TSAP-ID: 0xE00301
rem_subnet_id	ARRAY [1..6] of BYTE	rem_subnet_id[1-6] = B#16#00	This parameter is not used. Assign "0" to it.
rem_staddr	ARRAY [1..6] of BYTE	"192.168.0.16" • rem_staddr[1] = B#16#C0 (192) • rem_staddr[2] = B#16#A8 (168) • rem_staddr[3] = B#16#00 (0) • rem_staddr[4] = B#16#10 (16) • rem_staddr[5-6] = B#16#00	IP address of the remote connection end point
rem_tsap_id	ARRAY [1..16] of BYTE	• rem_tsap_id[1] = B#16#E0 • rem_tsap_id[2] = B#16#03 • rem_tsap_id[3] = B#16#01 • rem_tsap_id[4-16] = B#16#00	remote TSAP-ID: 0xE00301
next_staddr	ARRAY [1..6] of BYTE	next_staddr[1-6] = B#16#00	This parameter is not used (because communication is over the integrated interface)
spare	WORD	W#16#0000	This parameter is not used. Assign "0" to it.

The following table shows all parameter entries of communication partner B in the corresponding DB.

Parameter	Data type	Value in example	Description
block_length	WORD	W#16#40	Length of UDT65: 64 bytes (fixed)
id	WORD	W#16#0001	Reference to this connection
connection_type	BYTE	B#16#12	Protocol variant: • B#16#12: ISO on TCP
active_est	BOOL	TRUE	Active connection establishment
local_device_id	BYTE	B#16#05	Communication via the integrated IE interface for CPUs 41x PN/DP
local_tsap_id_len	BYTE	B#16#03	Length of parameter local_tsap_id
rem_subnet_id_len	BYTE	B#16#00	This parameter is not used. Assign with B#16#00.
rem_staddr_len	BYTE	B#16#04	Length of the address of the remote connection end point: • 4: Valid IP address in the rem_staddr parameter
rem_tsap_id_len	BYTE	B#16#03	Length of parameter rem_tsap_id
next_staddr_len	BYTE	B#16#00	This parameter is not used (because communication is over the integrated interface). Assign with B#16#00.
local_tsap_id	ARRAY [1..16] of BYTE	• local_tsap_id[1] = B#16#E0 • local_tsap_id[2] = B#16#03 • local_tsap_id[3] = B#16#01 • local_tsap_id[4-16] = B#16#00	local TSAP-ID: 0xE00301
rem_subnet_id	ARRAY [1..6] of BYTE	rem_subnet_id[1-6] = B#16#00	This parameter is not used. Assign "0" to it.
rem_staddr	ARRAY [1..6] of BYTE	"192.168.0.14" • rem_staddr[1] = B#16#C0 (192) • rem_staddr[2] = B#16#A8 (168) • rem_staddr[3] = B#16#00 (0) • rem_staddr[4] = B#16#0E (14) • rem_staddr[5-6] = B#16#00	IP address of the remote connection end point
rem_tsap_id	ARRAY [1..16] of BYTE	• rem_tsap_id[1] = B#16#E0 • rem_tsap_id[2] = B#16#03 • rem_tsap_id[3] = B#16#01 • rem_tsap_id[4-16] = B#16#00	remote TSAP-ID: 0xE00301
next_staddr	ARRAY [1..6] of BYTE	next_staddr[1-6] = B#16#00	This parameter is not used (because communication is over the integrated interface)
spare	WORD	W#16#0000	This parameter is not used. Assign "0" to it.

Data types

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

TCON: Establish communication connection

Number of connections possible

For information on the number of possible TCP or UDP connections, refer to the specifications for your CPU.

Use with TCP and ISO on TCP

Both communication partners call the "TCON" instruction to set up and establish the communication connection. In the parameter assignment, you specify which partner is the active communication end point and which is passive.

After the connection is set up and established, it is automatically maintained and monitored by the CPU.

If the connection is terminated, for example due to a line break or due to the remote communication partner, the active partner attempts to re-establish the configured connection. You do not have to call "TCON" again.

An existing connection is terminated and the connection is removed when the "TDISCON" instruction is called or when the CPU has gone into STOP mode. To set up and establish the connection again, call "TCON" again.

Use with UDP

Both communication partners call the "TCON" instruction to configure their local communication access point (local port). A connection is configured between the user program and the communication layer of the operating system. No connection is established to the remote partner.

The local access point is used to send and receive UDP message frames.

The following parameters must match in the configuration of the local access point (UDT 65) or in the configuration of the remote partner (UDT 66).

- With UDP: Reference to the connection and the local port no.

Functional description

The "TCON" instruction is asynchronous, in other words, job processing extends over multiple calls. You start the job for setting up and establishing a connection by calling "TCON" with REQ = 1.

The job status is indicated by the output parameters BUSY and STATUS. Here, STATUS corresponds to the output parameter RET_VAL of the asynchronous instructions.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 5367).

The following table shows the relationship between BUSY, DONE and ERROR. Using this table, you can recognize the current status of "TCON" or when the connection was set up or established (for TCP and ISO-on-TCP).

BUSY	DONE	ERROR	Description
TRUE	irrelevant	irrelevant	The job is being processed.
FALSE	TRUE	FALSE	Job successfully completed.
FALSE	FALSE	TRUE	The job ended with an error. The cause of the error can be found in the STATUS parameter.
FALSE	FALSE	FALSE	The instruction was not assigned a (new) job.

Parameters

The following table shows the parameters of the "TCON" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, T, C	Control parameter REQUEST starts the job for establishing the connection specified by ID. The job starts on a rising edge.
ID	Input	WORD	M, D or constant	Reference to the connection to be established to the remote partner, or between the user program and the communication layer of the operating system. ID must be identical to the corresponding parameter ID in the local connection description. Value range: W#16#0001 to W#16#0FFF
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: • 0: Job not yet started or still executing. • 1: Job executed without errors
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY = 1: The job is not yet completed. • BUSY = 0: The job is completed.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: • ERROR=1: Error occurred during job processing. STATUS supplies detailed information on the type of error

Parameter	Declaration	Data type	Memory area	Description
STATUS	Output	WORD	M, D	Status parameter STATUS: Error information
CONNECT	InOut	ANY	D	Pointers to the associated connection description (UDT 65) See also: • Assigning parameters for communications connections with TCP and ISO on TCP (Page 7414) • Assigning parameters for the local communications access point with UDP (Page 7419) Note: You can enter the CONNECT parameter in these two ways: • absolute. Example: P#DB13.DBX0.0 byte 64 • symbolic. Example: CONN_DB.Conn1

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameters ERROR and STATUS

ERROR	STATUS (W#16#...)	Explanation
0	0000	Connection was established successfully
0	7000	No job processing active
0	7001	Start job processing, establishing connection
0	7002	Intermediate call (REQ irrelevant), connection being established
1	8086	The ID parameter is outside the permitted range.
1	8087	Maximum number of connections reached; no additional connection possible.
1	8089	The CONNECT parameter does not point to a data block.
1	809A	The CONNECT parameter points to a field that does not match the length of the connection description (UDT65).
1	809B	The local_device_id in the connection description does not match the CPU.
1	80A0	Group error for error codes W#16#80A1 and W#16#80A2
1	80A1	Connection or port is already being used by the user
1	80A2	Local or remote port is being used by the system
1	80A3	Attempt being made to re-establish an existing connection
1	80A4	IP address of the remote connection end point is invalid, e.g., it matches the local partner's own IP address
1	80A7	Communication error: You have called "TDISCON (Page 7434)" before "TCON" was completed. Connection establishment was aborted by calling "TDISCON (Page 7434)".
1	80B2	The CONNECT parameter points to a data block that was generated with the keyword UNLINKED.
1	80B3	Inconsistent parameter assignment: Group error for error codes W#16#80A0 to W#16#80A2, W#16#80A4, W#16#80B4 to W#16#80B9

ERROR	STATUS (W#16#...)	Explanation
1	80B4	You have violated one or more of the following conditions when establishing a connection passively (active_est = FALSE) with the ISO-on-TCP protocol variant (connection_type = B#16#12): • local_tsap_id_len >= B#16#02 • local_tsap_id[1] = B#16#E0 • With local_tsap_id_len >= B#16#03, local_tsap_id[1] is an ASCII character. • local_tsap_id[1] is an ASCII character and local_tsap_id_len >= B#16#03
1	80B5	Error in parameter active_est (UDT 65) with the UDP protocol variant
1	80B6	Parameters assignment error in parameter connection_type (UDT 65)
1	80B7	Error in one of the following parameters of UDT 65: block_length, local_tsap_id_len, rem_subnet_id_len, rem_staddr_len, rem_tsap_id_len, next_staddr_len
1	80B8	Parameters in the local connection description (UDT 65) and Parameter ID are different
1	80C3	All connection resources are in use.
1	80C4	Temporary communication error: • The connection cannot be established at this time. • The interface is receiving new parameters. • The configured connection is currently being removed by a "TDISCON (Page 7434)". • The H system is connecting and updating.
1	8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

See also

Assigning parameters for communications connections with TCP and ISO on TCP (Page 7414)

Assigning parameters for the local communications access point with UDP (Page 7419)

TDISCON: Terminate communication connection

Use with TCP and ISO on TCP

The "TDISCON" instruction terminates a communication connection from the CPU to a communication partner.

Use with UDP

The "TDISCON" instruction closes the local communication access point. This means that the connection between the user program and the communication layer of the operating system is terminated.

Functional description

The "TDISCON" instruction is asynchronous, in other words, job processing extends over multiple calls. You start the job for terminating a connection by calling the "TDISCON" instruction with REQ = 1.

After the "TDISCON" instruction has been successfully executed, the ID specified for the "TCON (Page 7431)" instruction is no longer valid and cannot be used for sending or receiving.

The job status is indicated by the output parameters BUSY and STATUS. Here, STATUS corresponds to the output parameter RET_VAL of the asynchronous instructions (see also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 5367)).

Parameters

The following table shows the parameters of the "TDISCON" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, T, C	Starts the job to terminate the connection specified in the ID on a rising edge.
ID	Input	WORD	M, D or constant	Reference to the connection established with "TCON (Page 7431)". Range of values: W#16#0001 to W#16#0FFF
DONE	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or still in progress • 1: Job completed without error
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter with the following values: • 0: Job not yet started or already completed • 1: Job not yet completed. A new job cannot be started.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: • 0: No error • 1: Error occurred
STATUS	Output	WORD	M, D	Status of the instruction

Parameters BUSY, DONE and ERROR

You can check the status of the job with the BUSY, DONE, ERROR and STATUS parameters. The BUSY parameter indicates the processing status. With the DONE parameter, you can check whether or not a job executed successfully. The ERROR parameter is set when errors occurred during execution of TDISCON. The error information is output at the STATUS parameter.

The following table shows the relationship between the BUSY, DONE and ERROR parameters:

BUSY	DONE	ERROR	Description
1	-	-	The job is being processed.
0	1	0	The job was completed successfully.
0	0	1	The job ended with an error. The cause of the error is output at the STATUS parameter.
0	0	0	No new job was assigned.

Parameters ERROR and STATUS

ERROR	STATUS (W#16#...)	Explanation
0	0000	Connection successfully terminated
0	7000	No job processing active
0	7001	Start of job execution, connection is being terminated.
0	7002	Connection is being terminated (REQ irrelevant).
1	8086	The ID parameter is outside the permitted address range.
1	80A3	Attempt being made to terminate a non-existent connection.
1	80C4	Temporary communication error: The interface is currently receiving new parameters, or the connection is currently being established / the H system is connecting and updating.

See also

Overview of the valid data types (Page 2236)

TSEND: Send data via communication connection

Description

The "TSEND" instruction sends data over an existing communication connection.

Functional description

"TSEND" is an asynchronous instruction, in other words, its processing extends over multiple calls. You start the job by calling "TSEND" with REQ = 1.

The job status is indicated by the output parameters BUSY and STATUS. Here, STATUS corresponds to the output parameter RET_VAL of the asynchronous instructions.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 5367).

The DONE output parameter is set if the data to be sent to the local interface was transferred completely.

The following table shows the relationship between BUSY, DONE and ERROR. Using this table, you can determine the current "TSEND" status.

BUSY	DONE	ERROR	Description
TRUE	irrelevant	irrelevant	The job is being processed.
FALSE	TRUE	FALSE	Job successfully completed.
FALSE	FALSE	TRUE	The job ended with an error. The cause of the error can be found in the STATUS parameter.
FALSE	FALSE	FALSE	The instruction was not assigned a (new) job.

Note

Due to the asynchronous operation of "TSEND", make sure the data in the send area remains consistent until the DONE parameter or the ERROR parameter has the value TRUE.

Parameters

The following table shows the parameters of the "TSEND" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, T, C	Control parameter REQUEST starts the send job on a rising edge. The data is transferred from the area specified by DATA and LEN.
ID	Input	WORD	M, D or constant	Reference to the associated connection. ID must be identical to the corresponding parameter ID in the local connection description. Value range: W#16#0001 to W#16#0FFF
LEN	Input	INT	I, Q, M, D, L	Maximum number of bytes to be sent with the job See also: Relationship between CPU and protocol variant used (connection_type) and transferable data length (Page 7422)
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: • 0: Job not yet started or still executing. • 1: The data was transferred completely to the local interface for sending.
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY = 1: The job is not yet completed. A new job cannot be triggered. • BUSY = 0: The job is completed.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: • ERROR=1: Error occurred during processing. STATUS supplies detailed information on the type of error

Parameter	Declaration	Data type	Memory area	Description
STATUS	Output	WORD	M, D	Status parameter STATUS: Error information
DATA	InOut	ANY	I, Q, M, D	Send area, contains address and length The address refers to: • The process image of the inputs • The process image of the outputs • A bit memory • A data block Note: Do not use ARRAY of BOOL as send area. Note: You can enter the DATA parameter in these two ways: • absolute. Example: P#DB13.DBX0.0 byte 64 • symbolic. Example: DB_name.variable

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameters ERROR and STATUS

ERROR	STATUS (W#16#...)	Explanation
0	0000	Send job completed without error
0	7000	No job processing active
0	7001	Start of job processing, data being sent Note: During this processing phase, the operating system accesses the data in the DATA send area.
0	7002	Intermediate call (REQ irrelevant), job is being processed Note: During this processing phase, the operating system accesses the data in the DATA send area.
1	8085	The LEN parameter has the value "0" or is greater than the highest permitted value.
1	8086	The ID parameter is not in the permitted range
1	8088	The LEN parameter is greater than the memory area specified in DATA.
1	80A1	Communication error: • The specified connection has not yet been established. • The specified connection is currently being terminated. Transmission over this connection is not possible. • The interface is being reinitialized.
1	80B3	The connection type (connection_type parameter in the connection description) is set to UDP. Please use "TUSEND (Page 7443)".
1	80C3	• A block with this ID is already being processed in a different priority class. • Internal lack of resources.

ERROR	STATUS (W#16#...)	Explanation
1	80C4	Temporary communication error: • The connection to the partner cannot be established at the moment. • The interface is receiving new parameter settings or the connection is currently being established.
1	8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

TRCV: Receive data via communication connection

Description

The "TRCV" instruction receives data over an existing communication connection.

Receive area

This means the area in which the function block enters the received data.

The receive area is specified by the following two variables:

- Pointer to the start of the area
- Length of the area

Depending on the protocol variant being used, the length of the area is specified either by the LEN parameter (if LEN <> 0) or the length information of the DATA parameter (if LEN = 0).

Receive modes of TRCV

The following table shows how "TRCV" enters the received data in the receive area.

Protocol variant	Entering the data in the receive area	connection_type parameter	Value of the LEN parameter
TCP	Ad-hoc mode	B#16#01, B#16#11	0
TCP	Data reception with specified length	B#16#01, B#16#11	<> 0
ISO on TCP	protocol-controlled	B#16#12	0 (recommended) or <> 0

TCP / ad-hoc mode

The ad-hoc mode exists only with the TCP protocol variant. You set ad-hoc mode by assigning the value "0" to the LEN parameter.

The receive area is identical to the area formed by DATA. A maximum of 8192 bytes are received. Immediately after receiving a block of data, the "TRCV" instruction transfers this to the receive area and sets NDR to "1".

TCP / data reception with specified length

You set data reception with a specified length by assigning a value other than zero to the LEN parameter.

The receive area is defined by the LEN and DATA parameters. As soon as LEN bytes have been received, TRCV transfers them to the receive area and sets NDR to "1".

ISO on TCP / protocol-controlled data transfer

With the ISO-on-TCP protocol variant, data is transferred protocol-controlled.

The receive area is defined by the LEN and DATA parameters. As soon as all job data has been received, "TRCV" transfers it to the receive area and sets NDR to "1".

Data exchange of data type STRING

If you want to exchange data of the data type STRING with the "TSEND" and "TRCV" instructions, only the Ad-hoc-Mode with LEN=0 is appropriate. Besides, you have to set the string length for "TRCV" at least that large as for the "TSEND" instruction. If you do not follow this rule, it can lead to errors during the further string processing.

If you use ARRAY of BYTE as receive area for a sent character string, the first byte contains the maximum length that is set by the sender abd the secind byte the actual character string length.

Functional description

"TRCV" is an asynchronous instruction, in other words, its processing extends over multiple calls. You start the receive job by calling TRCV with EN_R = 1.

The job status is indicated by the output parameters BUSY and STATUS. Here, STATUS corresponds to the output parameter RET_VAL of the asynchronous instructions.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 5367).

The following table shows the relationship between BUSY, NDR and ERROR. Using this table, you can determine the current status of "TRCV" or when the receive operation is completed.

BUSY	NDR	ERROR	Description
TRUE	irrelevant	irrelevant	The job is being processed.
FALSE	TRUE	FALSE	Job successfully completed.
FALSE	FALSE	TRUE	The job ended with an error. The cause of the error can be found in the STATUS parameter.
FALSE	FALSE	FALSE	The instruction was not assigned a (new) job.

Note

Due to the asynchronous operation of "TRCV", the data in the receive area is only consistent when the NDR parameter has the value TRUE.

Parameters

The following table shows the parameters of the "TRCV" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L, T, C	Control parameter enabled to receive: With EN_R = 1, "TRCV" is ready to receive. The receive job is being processed.
ID	Input	WORD	M, D or constant	Reference to the associated connection. ID must be identical to the associated parameter ID in the local connection description. Value range: W#16#0001 to W#16#0FFF
LEN	Input	INT	I, Q, M, D, L	Length of the receive area in bytes For the meaning of LEN = 0 or LEN <> 0, see above (receive modes of "TRCV"). For the value range, see also: Relationship between CPU and protocol variant used (connection_type) and transferable data length (Page 7422).
NDR	Output	BOOL	I, Q, M, D, L	Status parameter NDR: • NDR = 0: Job not yet started or still running • NDR = 1: Job successfully completed
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: • ERROR=1: Error occurred during processing. STATUS supplies detailed information on the type of error
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY = 1: The job is not yet completed. A new job cannot be triggered. • BUSY = 0: The job is completed.
STATUS	Output	WORD	M, D	Status parameter STATUS: Error information
RCVD_LEN	Output	INT	I, Q, M, D, L	Amount of data actually received in bytes
DATA	InOut	ANY	I, Q, M, D	Receive area (definition see above), contains address and length The address refers to: • The process image of the inputs • The process image of the outputs • A bit memory • A data block Note: Do not use ARRAY of BOOL as receive area. Note: You can enter the DATA parameter in these two ways: • absolute. Example: P#DB13.DBX0.0 byte 64 • symbolic. Example: DB_name.variable

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameters ERROR and STATUS

ERROR	STATUS (W#16#...)	Explanation
0	0000	New data was accepted. The current length of the received data is shown in RCVD_LEN.
0	7000	Block not ready to receive
0	7001	Block is ready to receive, receive job was activated
0	7002	Intermediate call, receive job being processed Note: During this processing phase the instruction writes data to the receive area. For this reason, an error could result in inconsistent data in the receive area.
1	8085	The LEN parameter is greater than the largest permitted value, or you changed the value of the LEN or DATA parameter since the first call
1	8086	The ID parameter is not in the permitted range
1	8088	• Receive area is too small • Value in LEN is higher than the receive area specified by DATA
1	80A1	Communication error: • The specified connection has not yet been established. • The specified connection is currently being terminated. A receive job over this connection is not possible. • New parameter settings are being assigned to the interface.
1	80B3	The protocol variant (connection_type parameter in the connection description) is set to UDP. Please use "TURCV (Page 7446)".
1	80C3	• A block with this ID is already being processed in a different priority class. • Internal lack of resources.
1	80C4	Temporary communication error: • The connection to the partner cannot be established at the moment. • The interface is receiving new parameter settings or the connection is currently being established.
1	8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

TUSEND: Send data via Ethernet (UDP)

Description

The "TUSEND" instruction sends data to the remote partner addressed by the ADDR parameter using UDP.

WARNING

Data transfer via UDP

When transferring data with UDP according to RFC 768, the data is transferred to the remote partner without acknowledgement and is therefore unreliable. This means that data can be lost without this being indicated by the block.

Note

For sequential send operations to different partners, you only need to adjust the ADDR parameter when calling "TUSEND". You do not need to call the "TCON (Page 7431)" and "TDISCON (Page 7434)" instructions again.

Functional description

"The TUSEND" instruction works asynchronously, which means its job processing extends over multiple calls. You start the job by calling "TUSEND" with REQ = 1.

The job status is indicated by the output parameters BUSY and STATUS. Here, STATUS corresponds to the output parameter RET_VAL of the asynchronous instructions.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 5367).

The following table shows the relationship between BUSY, DONE and ERROR. Using this table, you can determine the current status of "TUSEND" or when the send process is concluded.

BUSY	DONE	ERROR	Description
TRUE	irrelevant	irrelevant	The job is being processed.
FALSE	TRUE	FALSE	Job successfully completed.
FALSE	FALSE	TRUE	The job ended with an error. The cause of the error can be found in the STATUS parameter.
FALSE	FALSE	FALSE	The instruction was not assigned a (new) job.

Note

Due to the asynchronous operation of "TUSEND", make sure the data in the send area remains consistent until the DONE parameter or the ERROR parameter has the value TRUE.

Parameters

The following table shows the parameters of the "TUSEND" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, T, C	Control parameter REQUEST starts the send job on a rising edge. The data is transferred from the area specified by DATA and LEN.
ID	Input	WORD	M, D or constant	Reference to the associated connection between the user program and the communication layer of the operating system. ID must be identical to the corresponding parameter ID in the local connection description. Value range: W#16#0001 to W#16#0FFF
LEN	Input	INT	I, Q, M, D, L	Number of bytes to be sent with the job Range of values: 1 to 1472
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: • 0: Job not yet started or still executing. • 1: Job executed without errors
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY = 1: The job is not yet completed. A new job cannot be triggered. • BUSY = 0: The job is completed.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: • ERROR=1: Error occurred during processing. STATUS supplies detailed information on the type of error
STATUS	Output	WORD	M, D	Status parameter STATUS: Error information
DATA	InOut	ANY	I, Q, M, D	Send area, contains address and length The address refers to: • The process image of the inputs • The process image of the outputs • A bit memory • A data block Note: You can enter the DATA parameter in these two ways: • absolute. Example: P#DB13.DBX0.0 byte 64 • symbolic. Example: DB_name.variable
ADDR	InOut	ANY	D	Pointers to the address of the receiver (for example, P#DB100.DBX0.0 byte 8) See also: Structure of the address information for the remote partner with UDP (Page 7421) Note: You can enter the ADDR parameter in these two ways: • absolute. Example: P#DB13.DBX0.0 byte 64 • symbolic. Example: DB_name.variable

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameters ERROR and STATUS

ERROR	STATUS (W#16#...)	Explanation
0	0000	Send job completed without error
0	7000	No job processing active
0	7001	Start of job processing, data being sent Note: During this processing phase, the operating system accesses the data in the DATA send area.
0	7002	Intermediate call (REQ irrelevant), job is being processed Note: During this processing phase, the operating system accesses the data in the DATA send area.
1	8085	The LEN parameter has the value "0" or is greater than the highest permitted value.
1	8086	The ID parameter is not in the permitted range
0	8088	The LEN parameter is greater than the memory area specified in DATA.
1	8089	The ADDR parameter does not point to a data block
1	80A1	Communication error: • The specified connection between user program and communication layer of the operating system has not yet been established. • The specified connection between the user program and the communication layer of the operating system is currently being terminated. Transmission over this connection is not possible. • The interface is being reinitialized.
1	80A4	IP address of the remote connection end point is invalid, e.g., it matches the local partner's own IP address.
1	80B3	• The protocol variant (connection_type parameter in the connection description) is not set to UDP. Please use "TSEND (Page 7436)". • ADDR parameter: Invalid information for port no.
1	80C3	• A block with this ID is already being processed in a different priority class. • Internal lack of resources.
1	80C4	Temporary communication error: • The connection between the user program and the communication layer of the operating system cannot be established at this time. • New parameter settings are being assigned to the interface.
1	8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

TURCV: Receive data via Ethernet (UDP)

Description

The "TURCV" instruction receives data via UDP. After successful completion of "TURCV", the ADDR parameter will show you the address of the remote partner (the sender).

> **WARNING**
>
> **Data transfer via UDP according to RFC 768**
>
> When transferring data with UDP according to RFC 768, the data is transferred to the remote partner without acknowledgement and is therefore unreliable. This means that data can be lost without this being indicated by the block.

Functional description

"TURCV" is an asynchronous instruction, in other words, its job processing extends over multiple calls. You start the receive job by calling "TURCV" with EN_R = 1.

The job status is indicated by the output parameters BUSY and STATUS. Here, STATUS corresponds to the output parameter RET_VAL of the asynchronous instructions

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 5367).

The following table shows the relationship between BUSY, NDR and ERROR. Using this table, you can determine the current status of TURCV or when the receive process is concluded.

BUSY	NDR	ERROR	Description
TRUE	irrelevant	irrelevant	The job is being processed.
FALSE	TRUE	FALSE	Job successfully completed.
FALSE	FALSE	TRUE	The job ended with an error. The cause of the error can be found in the STATUS parameter.
FALSE	FALSE	FALSE	The instruction was not assigned a (new) job.

> **Note**
>
> Due to the asynchronous operation of "TURCV", the data in the receive area is only consistent when the NDR parameter has the value TRUE.

Parameters

The following table shows the parameters of the "TURCV" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_R	Input	BOOL	I, Q, M, D, L, T, C	Control parameter enabled to receive: With EN_R = 1, "TURCV" is ready to receive. The receive job is being processed.
ID	Input	WORD	M, D or constant	Reference to the associated connection between the user program and the communication layer of the operating system. ID must be identical to the associated parameter ID in the local connection description. Value range: W#16#0001 to W#16#0FFF
LEN	Input	INT	I, Q, M, D, L	Length of the receive area in bytes: 0 (recommended) or 1 to 1472 See also: Functional description of instructions for Open User Communication via Industrial Ethernet (Page 7411)
NDR	Output	BOOL	I, Q, M, D, L	Status parameter NDR: • NDR = 0: Job not yet started or still running • NDR = 1: Job successfully completed
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: • ERROR=1: Error occurred during processing. STATUS supplies detailed information on the type of error
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY = 1: The job is not yet completed. A new job cannot be triggered. • BUSY = 0: The job is completed.
STATUS	Output	WORD	M, D	Status parameter STATUS: Error information
RCVD_LEN	Output	INT	I, Q, M, D, L	Amount of data actually received in bytes

Parameter	Declaration	Data type	Memory area	Description
DATA	InOut	ANY	I, Q, M, D	Receive area (for definition, see Functional description of instructions for Open User Communication via Industrial Ethernet (Page 7411)) The address references: • The process image of the inputs • The process image of the outputs • A bit memory • A data block Note: You can enter the DATA parameter in these two ways: • absolute. Example: P#DB13.DBX0.0 byte 64 • symbolic. Example: DB_name.variable
ADDR	InOut	ANY	D	Pointers to the address of the receiver (for example, P#DB100.DBX0.0 byte 8), see also: Structure of the address information for the remote partner with UDP (Page 7421) Note: You can enter the ADDR parameter in these two ways: • absolute. Example: P#DB13.DBX0.0 byte 64 • symbolic. Example: DB_name.variable

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameters ERROR and STATUS

ERROR	STATUS (W#16#...)	Explanation
0	0000	New data was accepted. The current length of the received data is shown in RCVD_LEN.
0	7000	Block not ready to receive
0	7001	Block is ready to receive, receive job was activated
0	7002	Intermediate call, receive job being processed Note: During this processing phase, "TURCV" writes data to the receive area. For this reason, an error could result in inconsistent data in the receive area.
1	8085	The LEN parameter is greater than the largest permitted value, or you changed the value of the LEN or DATA parameter since the first call
1	8086	The ID parameter is not in the permitted range
1	8088	• Receive area is too small • Value in LEN is higher than the receive area specified by DATA
1	8089	The ADDR parameter does not point to a data block

ERROR	STATUS (W#16#...)	Explanation
1	80A1	Communication error: • The specified connection between user program and communication layer of the operating system has not yet been established. • The specified connection between the user program and the communication layer of the operating system is currently being terminated. A receive job over this connection is not possible. • New parameter settings are being assigned to the interface.
1	80B3	The protocol variant (connection_type parameter in the connection description) is not set to UDP. Please use "TRCV (Page 7439)".
1	80C3	• A block with this ID is already being processed in a different priority class. • Internal lack of resources.
1	80C4	Temporary communication error: New parameter settings are being assigned to the interface.
1	8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

IP_CONF: Change IP configuration parameters

Description of IP_CONF

Description

The "IP_CONF" instruction is used for the program-controlled configuration of the integrated PROFINET interface of the CPU. The existing configuration data is overwritten.

You can make the following interface configuration settings:

- IP parameters: IP address, subnet mask, router address
- PROFINET IO device name (if the CPU is operated as a PROFINET IO device)

You have to store the configuration data in a configuration DB.

You can make the program-controlled setting of the IP configuration with the "IP_CONF" instruction as an alternative to configuration with STEP 7. It only takes effect, however, if you explicitly specified in the hardware configuration that IP parameters are assigned via the user program.

Note

As an alternative to project engineering, the configuration data of the communication processor can be set using the "IP_CONFIG" instruction (see SIMATIC NET CP).

Functional description

The "IP_CONF" instruction works asynchronously, that is, its execution extends over multiple calls. You start the transfer operation by calling "IP_CONF" with REQ = 1.

Only one job can be active at any time.

The job status is indicated by the output parameters BUSY and STATUS. Here, STATUS corresponds to the output parameter RET_VAL of the asynchronous instructions (see also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 5367)).

The following table shows the relationship between BUSY, DONE and ERROR. Using this table, you can determine the current status of the instruction and when the transfer of configuration data is concluded.

BUSY	DONE	ERROR	Description
TRUE	irrelevant	irrelevant	The job is being processed.
FALSE	TRUE	FALSE	Job successfully completed.
FALSE	FALSE	TRUE	The job ended with an error. The cause of the error can be found in the STATUS parameter.
FALSE	FALSE	FALSE	The instruction was not assigned a (new) job.

Parameters

The following table shows the parameters of the "IP_CONF" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, T, C	Control parameter REQUEST. starts the instruction on a rising edge.
LADDR	Input	WORD	M, D or constant	Diagnostics address of the PROFINET interface
CONF_DB (Page 7452)	Input	ANY	D	Pointer to the configuration data (permitted data types: BYTE, WORD, BLOCK_DB) Note: You can enter the CONF_DB parameter in these two ways: • absolute. Example: P#DB13.DBX0.0 byte 64 • symbolic. Example: DB_name.variable
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE: • 0: Job not yet started or still executing. • 1: Job executed without errors
BUSY	Output	BOOL	I, Q, M, D, L	• 0: Job is complete. • 1: The job is not yet completed. A new job cannot be triggered.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR: ERROR=1: Error occurred during processing. STATUS supplies detailed information on the type of error
STATUS	Output	DWORD	I, Q, M, D, L	Error information
ERR_LOC	Output	DWORD	I, Q, M, D, L	Error source (field_id and subfield_type_id of the parameter block causing the error)

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

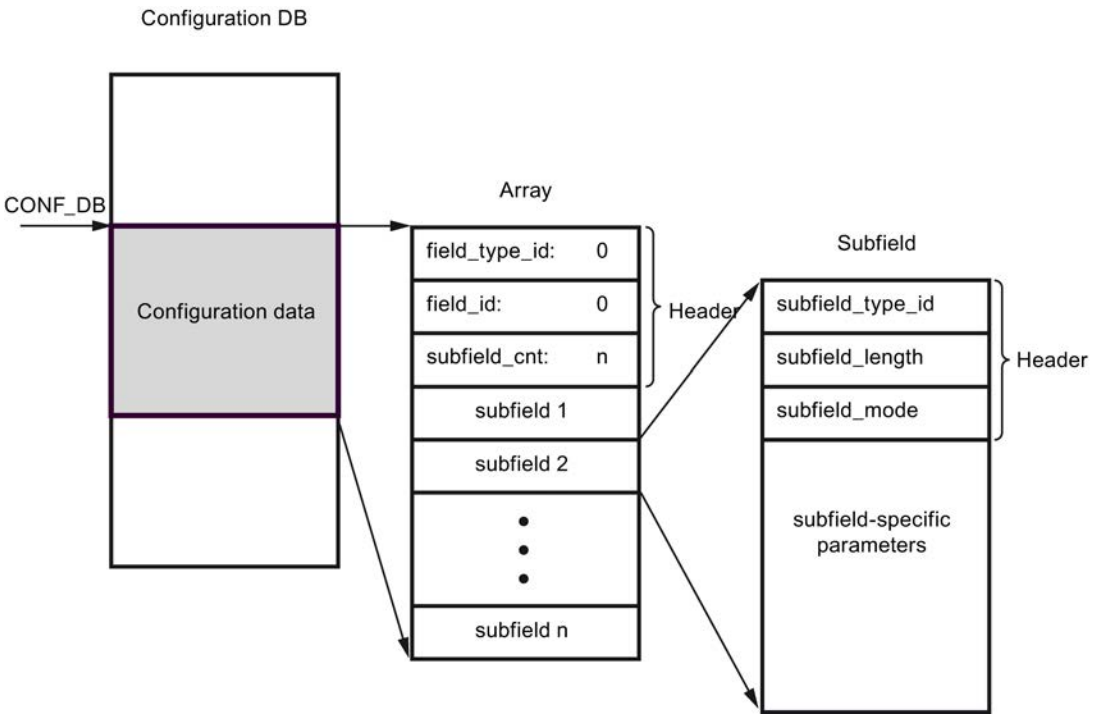
Parameters ERROR and STATUS

ERROR	STATUS (DW#16#..)	ERR_LOC*	Explanation
0	00000000	0	Job execution terminated without error
0	00700000	0	No job processing active
0	00700100	0	Start of job execution
0	00700200	0	Intermediate call (REQ irrelevant)
1	C08xyy00	0	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)
1	C0808000	0	LADDR is invalid
1	C0808100	0	LADDR is not assigned to the supported PROFINET interface
1	C0808200	0	Error in the CONF_DB parameter: Data type is not supported.
1	C0808300	0	Error in the CONF_DB parameter: The pointer points to an unsupported area.
1	C0808400	0	Error in the CONF_DB parameter: Incorrect length of the ANY pointer
1	C0808700	0	Error in the CONF_DB parameter: Inconsistency of the length specified in the configuration data
1	C0808800	s, 0	field_type_id has an illegal value
1	C0808900	s, 0	field_id has an illegal value
1	C0808A00	s, 0	Wrong number for subfield_cnt
1	C0808B00	s, t	subfield_id has an illegal value
1	C0808C00	s, t	Subfield used more than once
1	C0808D00	s, t	subfield_len has an incorrect or illegal value
1	C0808E00	s, t	subfield_mode has an illegal value
1	C0808F00	s, t	There is a conflict with an earlier subfield in the subfield.
1	C0809000	s, t	The parameters of the subfield are write-protected. E.g. parameters are specified via configuration or PNIO mode is enabled.
1	C0809400	s, t	The parameter value in the subfield is not defined or illegal
1	C080C200	0	Transfer cannot be carried out (e.g. because the interface is not reachable).
1	C080C300	0	Insufficient resources (for example, multiple calling of "IP_CONF" with different parameters)
1	C080C400	0	Temporary communication error
* In the table above, f is the field_id and s the subfield_type_id of the parameter block causing the error.			

Data block CONF_DB

Description

The following figure shows how the configuration data to be transferred is stored in the configuration DB.



The configuration data therefore comprises exactly one field and several subfields:

- The field consists of a header and the subfields. The header, in turn, consists of the following elements:
  - field_type_id (data type INT): Zero
  - field_id (data type INT): Zero
  - subfield_cnt (data type INT): Number of subfields:
- Each subfield consists, in turn, of a header (subfield_type_id, subfield_length, subfield_mode) and the subfield-specific parameters. Each subfield must consist of an even number of bytes.
  The following values are permitted for subfield_mode:
  - 1: permanent validity of the configuration data
  - 2: temporary validity of the configuration data including the deletion of existing permanent configuration data

Note

Exactly one field is currently permitted. Its parameters field_type_id and field_id must have the value zero. Other fields with different values for field_type_id and field_id are subject to future extensions.

Permitted subfields

subfield_type_id	Name of the subfield	Explanation
30	SUB_IP_SUITE_IPV4	IP parameters: IP address, subnet mask, router address
40	SUB_NOS	PROFINET IO device name (Name of Station)

Type definition for the subfield "SUB_IP_SUITE_IPV4"

Name		Data type	Start value	Description
SUB_IP_SUITE_V4		Struct		
	Id	INT	30	subfield_type_id
	len	INT	18	subfield_length
	mode	INT	1	subfield_mode (1: permanent, 2: temporary)
	ipaddr_3	BYTE	b#16#C8	IP address high byte: 200
	ipaddr_2	BYTE	b#16#0C	IP address high byte: 12
	ipaddr_1	BYTE	b#16#01	IP address low byte: 1
	ipaddr_0	BYTE	b#16#90	IP address low byte: 144
	snmask_3	BYTE	b#16#FF	Subnet mask high byte: 255
	snmask_2	BYTE	b#16#FF	Subnet mask high byte: 255
	snmask_1	BYTE	b#16#FF	Subnet mask low byte: 255
	snmask_0	BYTE	b#16#00	Subnet mask low byte: 0
	router_3	BYTE	b#16#C8	Router high byte: 200
	router_2	BYTE	b#16#0C	Router high byte: 12

Name		Data type	Start value	Description
	router_1	BYTE	b#16#01	Router low byte: 1
	router_0	BYTE	b#16#01	Router low byte: 1

Type definition for the subfield "SUB_NOS"

Name		Data type	Start value	Description
SUB_NOS		Struct		
	id	INT	40	subfield_type_id
	len	INT	246	subfield_length
	mode	INT	1	subfield_mode (1: permanent, 2: temporary)
	nos	ARRAY[1..240] of BYTE	0	Device name: You must occupy the ARRAY from the first byte. If the ARRAY is longer than the device name to be assigned, enter a zero byte after the actual device name (complying with IEC 61158-6-10). Otherwise, nos is rejected and the "IP_CONF (Page 7449)" instruction enters the error code DW#16#C0809400 in STATUS. If you enter zero the first byte, the device name is deleted.

The device name is subject to the following limitations:

- Restricted to a total of 240 characters (lower case letters, numbers, dash, or dot)
- A name component within the device name, in other words, a character string between two dots, must not exceed 63 characters. The name components must not begin or end with the "-" character.
- No special characters such as umlauts, brackets, underscore, slash, blank space, etc. The only special character permitted is the dash.
- The device name must not begin or end with the "-" or "." characters.
- The device name must not begin with a number.
- The device name form n.n.n.n (n = 0, ... 999) is not permitted.
- The device name must not begin with the string "port-xyz" or "port-xyz-abcde" (a, b, c, d, e, x, y, z = 0, ... 9).
- The character string ".." is not permitted in the device name.

Note

You can also create an ARRAY "nos" that is shorter then 240 bytes, but not less than 2 bytes. In this case, you must adjust the "len" (length of subfield) tag accordingly.

Data exchange using FETCH and WRITE

FW_TCP: Swap data using FETCH and WRITE via TCP

Description

With the "FW_TCP" instruction, the FETCH/WRITE functions are provided via the TCP connection.

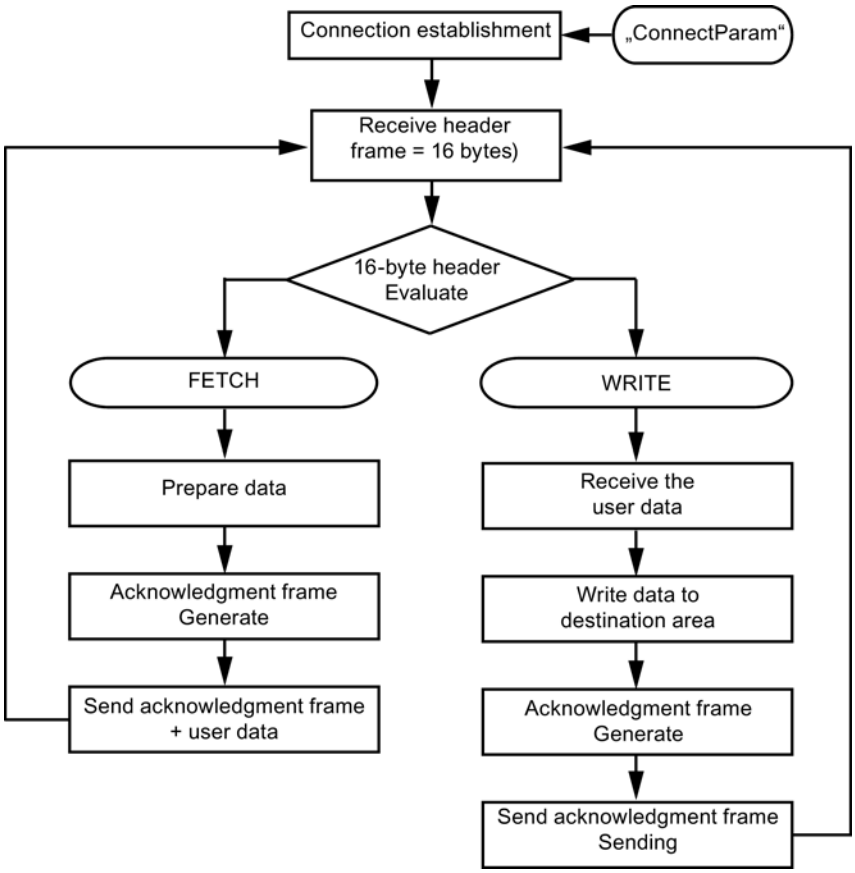
Call

Since a separate TCP connection is required for each of the services FETCH/WRITE, the instruction is called twice in OB1:

- The first call with the FETCH_TCP_DB instance data block provides the FETCH service via the TCP connection.
- The second call with the WRITE_TCP_DB instance data block provides the WRITE service via the TCP connection.

The TCP connection is automatically established after a cold restart, warm restart and hot restart of the CPU, as well as after changes to the connection parameters. The parameters for the connection establishment are stored in the "ConnectParam" data block (description: See below). The input parameter ENABLE provides the option of manually controlling the establishment and termination of the connection. When the connection parameters are changed, the connection is automatically terminated and re-established.

The following figure is a schematic representation of the "FW_TCP" instruction, which is called in OB1 and implements the FETCH and WRITE services via TCP connections.



Connection parameters of the "ConnectParam" instance data block

The following connection parameters are checked before the connection is established:

- connection_type
- id
- local_tsap_id

If one of these parameters is wrongly defined, the communication connection will not be established. No data is sent or received.

- After the communication connection has been established, a 16-byte header frame of the FETCH/WRITE services is received. The header frame contains the information indicating whether a FETCH or WRITE job is pending.
- If a FETCH job is pending, the data requested by the communication partner is prepared in the instance data block DB210 of the instruction and the acknowledgement frame generated. The acknowledgement frame is sent including the prepared data.
- If a WRITE job is pending, the user data is received and written to the destination area after the evaluation of the 16-byte header frame. The acknowledgement frame is then generated in instance data block DB310 and sent.

The following table shows the data structure of the connection parameters for a TCP connection. This is 64 bytes long.

Parameter	Data type	Start value	Description
id	WORD	W#16#1	Connection ID: The value of the parameter must be within the following range: id = W#16#0001 to W#16#0FFF (1 to 4095)
connection_type	BYTE	B#16#11	Connection type. The following is valid for the TCP connection: • connection_type = B#16#11 (TCP native) or • connection_type = B#16#1 (TCP native or compatibility mode)
active_est	BOOL	FALSE	Passive connection establishment
local_device_id	BYTE	B#16#3	ID for the local PN/IE interface (here: CPU 319-3PN/DP).
local_tsap_id_len	BYTE	B#16#2	Length of the local_tsap_id parameter used in bytes: • 0 or 2, if connection type = 17 (TCP) • Only the value "0" is permitted for the active end.
rem_subnet_id_len	BYTE	B#16#0	This parameter is not used.
rem_staddr_len	BYTE	B#16#0	Length of address of partner end point (rem_staddr parameter) in bytes. Since this is an unspecified connection, the parameter is irrelevant.
rem_tsap_id_len	BYTE	B#16#0	Length of the rem_tsap_id parameter used in bytes: • 0 or 2, if connection type = 17 (TCP). • Only the value 0 is permitted for the passive end.
next_staddr_len	BYTE	B#16#0	This parameter is not used.
local_tsap_id	ARRAY [1..16] of BYTE	B#16#7 B#16#D0 B#16#0 : : B#16#0	Local port for the TCP connection. Possible values: 1 to 49151 (recommended values: 2000...5000). • local_tsap_id[1] = high byte of the port no. in hexadecimal notation; • local_tsap_id[2] = low byte of port no. in hexadecimal notation; • local_tsap_id[3-16] = irrelevant Note: Make sure that every value of local_tsap_id is unique within the CPU.
rem_subnet_id	ARRAY [1..6] of BYTE	B#16#0 …	This parameter is not used.
rem_staddr	ARRAY [1..6] of BYTE	B#16#0	IP address of the partner end point, for example, for 192.168.2.3: • rem_staddr[1] = 192 • rem_staddr[2] = 168 • rem_staddr[3] = 2 • rem_staddr[4] = 3 • rem_staddr[5-6] = irrelevant Not relevant for unspecified connections with passive connection establishment.
rem_tsap_id	ARRAY [1..16] of BYTE	B#16#0	Port of the communication partner. For 17 (TCP) possible values: 1 to 49151 (recommended values: 2000...5000). • rem_tsap_id[1] = high byte of the port no. in hexadecimal notation; • rem_tsap_id[2] = low byte of port no. in hexadecimal notation; • rem_tsap_id[3-16] = irrelevant Not relevant for unspecified connections with passive connection establishment.

Parameter	Data type	Start value	Description
next_staddr	ARRAY [1..6] of BYTE	B#16#0	This parameter is not used.
spare	WORD	W#16#0	This parameter is not used.

Parameters

The following table shows the parameters of the "FW_TCP" instruction:

Parameter	Declaration	Data type	Memory area	Description
ENABLE	Input	BOOL	I, Q, M, D, L, T, C	The establishment and termination of the connection is initiated with this parameter: • 0 = Connection is terminated • 1 = Connection is established
CONNECT	Input	ANY	D	ANY pointer to the data area of DB211 that contains the parameters for establishing the TCP connection. This data area must be at least 64 bytes in size.
ADDRMODE	Input	INT	I, Q, M, D, L or constant	This parameter defines how the data of the FETCH or WRITE job are addressed (in S7 or S5 address mode). • 0 = S7 address mode (Start address of the data is interpreted as a byte address) • 1 = S5 address mode (Start address of the data is interpreted as a word address)
NDR	Output	BOOL	I, Q, M, D, L	new data record parameter: This parameter shows that the data of the WRITE job has been successfully adopted and that the acknowledgement frame was generated and sent.
ERROR	Output	BOOL	I, Q, M, D, L	This parameter is set when: • An error occurred establishing or terminating the connection • An error occurred during the sending or receiving of data. • An invalid FETCH/WRITE header frame was received.
MODE	Output	BYTE	I, Q, M, D, L	This parameter shows whether a FETCH or WRITE job was executed. • 0 = No job active • 1 = WRITE job • 2 = FETCH job
STATUS (Page 7463)	Output	WORD	M, D	Internal communication status bits are output in the STATUS parameter.

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

FW_IOT: Swap data using FETCH and WRITE via ISO-on-TCP

Description

With the "FW_IOT" instruction, the FETCH/WRITE services are provided via the ISO-on-TCP connection.

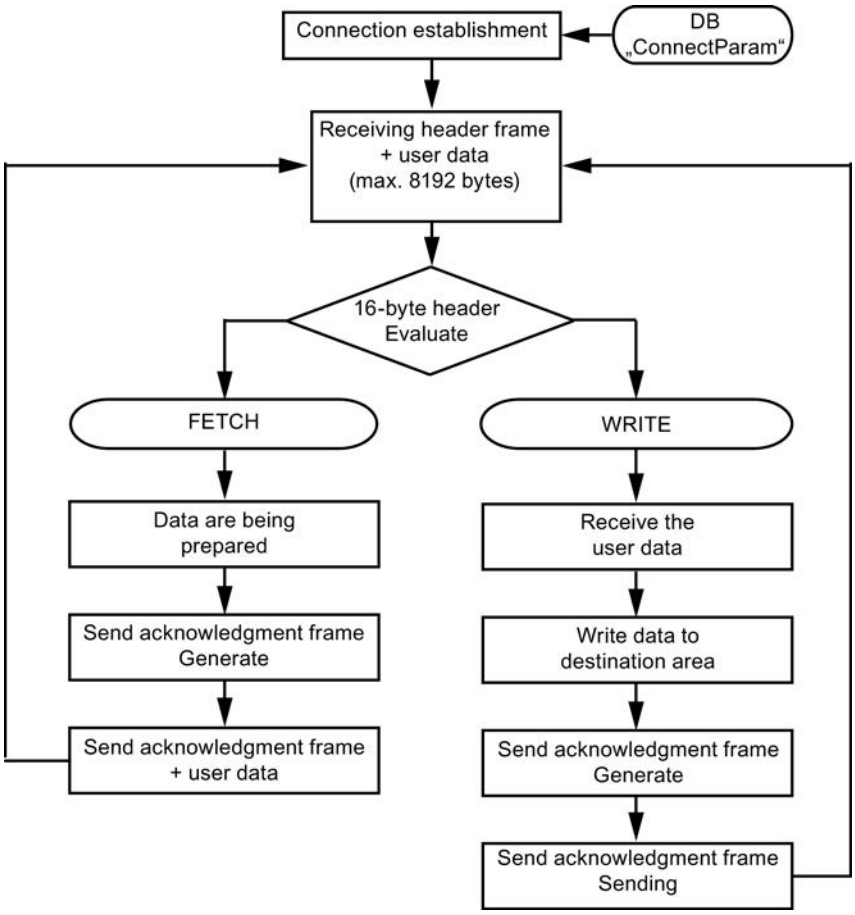
Call

Since a separate TCP connection is required for each of the FETCH/WRITE services, the "FW_IOT" instruction is called twice in OB1:

- The first call with the FETCH_IOT_DB instance data block provides the FETCH service via the ISO-on-TCP connections.
- The second call with the WRITE_IOT_DB instance data block provides the WRITE service via the ISO-on-TCP connection.

The ISO-on-TCP connection is automatically established after a cold restart, warm restart and hot restart of the CPU, as well as after the changes to the connection parameters. The parameters for the establishment of the connection are stored in the "ConnectParam" DB. With the ENABLE input parameter, the user has the option of manually controlling connection establishment and termination. When the connection parameters are changed, the connection is automatically terminated and re-established. The following table shows the data structure of the connection parameters for an ISO-on-TCP connection: This is 64 bytes long.

The following figure is a schematic representation of the "FW_IOT" instruction, which is called in OB1 and implements the FETCH and WRITE services via ISO-on-TCP connections.



Connection parameters of the "ConnectParam" instance data block

The following connection parameters are checked before the connection is established:

- connection_type
- id
- local_tsap_id
- local_tsap_id _len

If one of these parameters is wrongly defined, the communication connection will not be established. No data is sent or received.

- After the communication connection has been established, a 16-byte header frame of the FETCH/WRITE services is received. The header frame contains the information indicating whether a FETCH or WRITE job is pending.
- If a FETCH job is pending, the data requested by the communication partner is prepared in the FETCH_IOT_DB instance data block of the instruction and the acknowledgement frame generated. The acknowledgement frame is sent including the prepared data.
- If a WRITE job is pending, the user data is received and written to the destination area after the evaluation of the 16-byte header frame. The acknowledgement frame is then generated in the WRITE_IOT_DB instance data block and sent.

The following table shows the data structure of the connection parameters for a TCP connection. This is 64 bytes long.

Parameter	Data type	Start value	Description
id	WORD	W#16#1	Connection ID: The value of the parameter must be within the following range: id = W#16#0001 to W#16#0FFF (1 to 4095)
connection_type	BYTE	B#16#12	Connection type. The following is valid for an ISO-on-TCP connection: connection_type = B#16#12
active_est	BOOL	FALSE	Passive connection establishment
local_device_id	BYTE	B#16#3	ID for the local PN/IE interface (here: CPU 319-3PN/DP).
local_tsap_id_len	BYTE	B#16#2	Length of the local_tsap_id parameter used in bytes: • 2 to 16, if connection type = 18 (ISO-on-TCP)
rem_subnet_id_len	BYTE	B#16#0	This parameter is not used.
rem_staddr_len	BYTE	B#16#0	Length of address of partner end point (rem_staddr parameter) in bytes. Since this is an unspecified connection, the parameter is irrelevant.
rem_tsap_id_len	BYTE	B#16#0	Length of the rem_tsap_id parameter used in bytes: • 2 to 16, if connection type = 18 (ISO-on-TCP)
next_staddr_len	BYTE	B#16#0	This parameter is not used.
local_tsap_id	ARRAY [1..16] of BYTE	B#16#7 B#16#D0 B#16#0 : : B#16#0	Local TSAP ID for the ISO-on-TCP connection. • local_tsap_id[1] = B#16#E0; • local_tsap_id[2] = rack and slot of local end points (bits 0 to 4: slot number, bits 5 to 7: rack number); • local_tsap_id[3-16] = TSAP extension, optional Note: Make sure that every value of local_tsap_id is unique within the CPU.
rem_subnet_id	ARRAY [1..6] of BYTE	B#16#0 …	This parameter is not used.
rem_staddr	ARRAY [1..6] of BYTE	B#16#0	IP address of the partner end point, for example, for 192.168.2.3: • rem_staddr[1] = 192 • rem_staddr[2] = 168 • rem_staddr[3] = 2 • rem_staddr[4] = 3 • rem_staddr[5-6] = irrelevant Not relevant for unspecified connections with passive connection establishment.
rem_tsap_id	ARRAY [1..16] of BYTE	B#16#0	Port of the communication partner. For partner TSAP ID with ISO-on-TCP: • rem_tsap_id[1] = B#16#E0; • rem_tsap_id[2] = rack and slot of partner end point (bits 0 to 4: slot number, bits 5 to 7: rack number); • rem_tsap_id[3-16] = TSAP extension, optional Not relevant for unspecified connections with passive connection establishment.

Parameter	Data type	Start value	Description
next_staddr	ARRAY [1..6] of BYTE	B#16#0	This parameter is not used.
spare	WORD	W#16#0	This parameter is not used.

Parameters

The following table shows the parameters of the "FW_IOT" instruction:

Parameter	Declaration	Data type	Memory area	Description
ENABLE	Input	BOOL	I, Q, M, D, L, T, C	The establishment and termination of the connection is initiated with this parameter: • 0 = Connection is terminated • 1 = Connection is established
CONNECT	Input	ANY	D	ANY pointer to the data area of the DB that contains the parameters for establishing the TCP connection. This data area must be at least 64 bytes in size.
ADDRMODE	Input	INT	I, Q, M, D, L or constant	This parameter defines how the data of the FETCH or WRITE job are addressed (in S7 or S5 address mode). • 0 = S7 address mode (Start address of the data is interpreted as a byte address) • 1 = S5 address mode (Start address of the data is interpreted as a word address)
NDR	Output	BOOL	I, Q, M, D, L	new data record parameter: This parameter shows that the data of the WRITE job has been successfully adopted and that the acknowledgement frame was generated and sent.
ERROR	Output	BOOL	I, Q, M, D, L	This parameter is set when: • An error occurred establishing or terminating the connection • An error occurred during the sending or receiving of data. • An invalid FETCH/WRITE header frame was received.
MODE	Output	BYTE	I, Q, M, D, L	This parameter shows whether a FETCH or WRITE job was executed. • 0 = No job active • 1 = WRITE job • 2 = FETCH job
STATUS (Page 7463)	Output	WORD	M, D	Internal communication status bits are output in the STATUS parameter.

You can find additional information on valid data types under "Overview of the valid data types (Page 2236)".

STATUS parameter

Structure of the STATUS parameter

The STATUS output parameter of the "FW_TCP" and "FW_IOT" instructions provides information on the status of the communication between S7 station and PC station.

This status word is structured as follows:

Bit	Instance DB.FBStatus	Description
High byte		
0	NotConnected	No connection established or connection interrupted • The FETCH/WRITE client has not yet initiated the establishment of the connection • Error during connection establishment
1	ErrorConnect	Error in connection establishment (for example: Error in the data structure of the connection parameter)
2	ErrorDisConnect	Error during connection termination
3	ErrorConType	Wrong connection type in the data structure of the connection parameters)
4	Not assigned	
5	BusyConnect	Connection is being established
6	BusyDisConnect	Connection is being terminated
7	Not assigned	
Low byte		
8	ErrorRec1	TCP connection: • Error in receive job • FETCH/WRITE header frame could not be received ISO-on-TCP connection • Error in receive job • FETCH/WRITE header frame including user data could not be received
9	ErrorRec2	TCP connection • Error in receive job • User data could not be received
10	ErrorSend	TCP and ISO-on-TCP connection • Error in send job • FETCH/WRITE acknowledgement frame including user data could not be sent
11	Not assigned	
12	BusyRec1	Receive job is running
13	BusyRec2	Receive job is running
14	BusySend	Send job is running
15	ErrorHeader	Undefined FETCH/WRITE header frame received.

Addresses and tags of the instance data block

The error status of the connection establishment and termination, and the error status of the send and receive job are stored in the instance data block of the instructions "FW_TCP" and "FW_IOT".

- The instance data block of the "FW_TCP" instruction comprises 8348 bytes.
- The instance data block of the "FW_IOT" instruction comprises 16544 bytes.

The following table shows an overview of the addresses and tags of the instance data block where the status is stored.

Tag	Data type	Address	Description
DoneError.STATUS_Connect	WORD	22.0	Connection establishment status
DoneError.STATUS_REC_1	WORD	24.0	Status of the first receive job
DoneError.STATUS_REC_2	WORD	26.0	Status of the second receive job (only for TCP)
DoneError.STATUS_SEND	WORD	28.0	Status of the send job
DoneError.STATUS_DisConnect	WORD	30.0	Connection termination status

Value of the STATUS parameter

The following table provides an overview of the status values of the connection establishment.

STATUS (W#16#...)	Description
0000	Connection was established successfully.
8086	The ID parameter is outside the permitted range.
8087	Maximum number of connections reached; no additional connection possible.
8089	The CONNECT parameter does not point to a data block.
809A	The CONNECT parameter points to a field that does not match the length of the connection description.
809B	The local_device_id in the connection description does not match the CPU.
80A1	Connection or port is already in use.
80A2	Local or remote port is being used by the system.
80A3	Attempt is being made to terminate a non-existent connection.
80A4	IP address of the remote connection end point is invalid; for example, it matches the local partner's own IP address
80B2	The CONNECT parameter points to a data block that was generated with the keyword UNLINKED.
80B3	Inconsistent parameter assignment
80B4	You have violated one or more of the following conditions when establishing a connection passively (active_est = FALSE) with the ISO-on-TCP protocol variant (connection_type = B#16#12): • local_tsap_id_len >= B#16#02 • local_tsap_id[1] = B#16#E0
80B6	Parameter assignment error relating to the connection_type parameter
80B7	Error in the connection description
80C3	All connection resources are in use: • Another block with this ID is already being processed in a different priority class. • Internal lack of resources.

STATUS (W#16#...)	Description
80C4	Temporary communications error: • The connection cannot be established at this time. • New parameter settings are being assigned to the interface. • The configured connection is currently being removed by the "TDISCON" instruction.
82xx	General error information

Web server

WWW: Synchronizing user-defined web pages

Description

The instruction WWW initializes the web server of the CPU or synchronizes user-defined web pages with the user program in the CPU.

User-defined web pages together with the web server make it possible for the CPU to access freely designed web pages of the CPU with a web browser.

Use script instructions (such as Javascript) and HTML code in user-defined web pages to transfer data via a web browser for further processing to the CPU and to display data from the operand area of the CPU in the web browser. Call the WWW instruction in the user program for synchronization of the user program and the web server as well as initialization.

Initialization

User-defined web pages are "packaged" in data blocks for processing by the CPU. You will have to generate appropriate data blocks from the source files (HTML files, screens, Javascript files, ...) during configuration. The Web Control DB has a special role (default: DB 333). It contains status and control information as well as links to additional data blocks with coded web pages. The data blocks with coded web pages are called fragment DBs.

When the data block is downloaded into the CPU, the CPU does not "know" that user-defined web pages are coded inside it. The instruction "WWW" in the startup OB, for example, will inform the CPU which DB is the Web Control DB. The user-defined web pages can be accessed via a web browser after this initialization.

Synchronization

If you want the user program to interact with the user-defined web pages, then the instruction WWW must be used in the cyclical program part.

Examples of interaction between user program and web page:

- Check received data
- Assemble and send back data to the web browser making the request

In this case it must be possible to evaluate the current status information and the web server must receive control information, such as release of a web page requested by a web browser.

Parameter

The following table shows the parameters of the instruction "WWW":

Parameter	Declaration	Data type	Memory area	Description
CTRL_DB	Input	BLOCK_DB	D	Data block that describes the user-defined web pages (Web Control DB)
RET_VAL	Output	INT	I, Q, M, D, L	Error information

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	No error occurred. There are no web page requests that have to be released by the user program.
00xy	x: indicates whether an error has occurred during initialization of the Web Control DB (CTRL_DB): x=0: No errors occurred. x=1: Error occurred. The error is coded in the byte "CTRL_DB.commandstate.last_error" of the Web Control DB, see description of Web Control DB. y: Number of the pending request. Several requests are possible (e. g. requests "0" and "1" are pending: y="3". y="1": Request "0" y="2": Request "1" y="4": Request "2" y="8": Request "3"
803A	The specified Web Control DB does not exist on the CPU.
8081	Incorrect version or incorrect format of the Web Control DB
80C1	There are no resources to initialize the web application, for example, because only two or four web applications may be running.

Communications processor

Point-to-point

Overview of Freeport communication

STEP 7 offers extended instructions that can be used for Freeport communication with a protocol specified in the user program. These instructions can be divided into two categories:

- Configuration instructions
- Communication instructions

Note

CPU capacity

The Freeport instructions communicate with the communication modules by reading or writing data records.
When using the instructions, you should therefore check the capacity of the CPU for reading and writing data records.
If several instructions need to read or write data records simultaneously on a CPU, the user program may need to create a time offset between the calls of each instruction.

Configuration instructions

Before the user program can start the Freeport communication, the communication interface and the parameters for sending and receiving of data must be configured.

The interface configuration and the data configuration can be set for each CM in the device configuration or with the following instructions of your user program:

- Port_Config (Page 7472)
- Send_Config (Page 7474)
- Receive_Config (Page 7476)
- P3964_Config (Page 7480)

NOTICE
Device configuration <-> Configuration instructions The device configuration parameters are transferred to the CM upon each Power On of the CPU (return of voltage). The parameters of the configuration instructions are transferred to the CM as defined in your user program. The parameters of the device configuration are not synchronized with the parameters of the configuration instructions, which means the parameters of the configuration instructions are not applied to the CPU device configuration. With your user program, you determine the parameters that apply in the CM and when they apply.

Communication instructions

The user program uses the instructions for Freeport communication to send data to and receive data from the communication interfaces. The CMs send data to and receive data from the communication stations.

- Send_P2P (Page 7481)
- Receive_P2P (Page 7484)

Note

Data consistency

- If the data to be sent is transmitted consistently, it cannot be changed after the positive edge at the REQ parameter until DONE has been set by the Send_P2P instruction.
- If the receive data is to be read consistently, it may only be evaluated in the cycle in which NDR = TRUE.

The receive buffer can be reset with additional instructions and special RS232 signals can be queried and set.

- Receive_Reset (Page 7486)
- Signal_Get (Page 7486)
- Signal_Set (Page 7487)

The following instructions let you read or write extended functions, as long as these are supported by the module.

- Get_Features (Page 7488)
- Set_Features (Page 7489)

All Freeport instructions work asynchronously. The instructions must therefore be called until the DONE output parameter indicates that execution is complete.

The user program can determine the send and receive status with the help of the query architecture. Send_P2P and Receive_P2P can be run at the same time. The communication

modules buffer the send and receive data as required until a module-specific maximum buffer size has been reached.

Note

Resolution of bit times

The number of bit times is specified with the configured data transmission rate for different parameters. Specifying the parameter in bit times makes it independent of the data transmission rate. All parameters with unit of bit times can be specified with a maximum number of 65535.

Using the instructions

The Freeport instructions must be called cyclically to query received data or the end of transmission for a send process.

Depending on the data volume, data transmission may take place over several calls (program cycles).
If a command is completed with DONE = TRUE or NDR = TRUE, it has been executed without errors.

Note

Backing up STATUS

The DONE, NDR, ERROR and STATUS parameters are only available for one block cycle. To display the STATUS, you should therefore copy it to a free data area.

Master

Typical sequence for a master:

1. The Send_P2P instruction triggers transmission to the CM.
   Data transmission is initiated by a positive edge at the REQ input.
2. The Send_P2P instruction is executed in subsequent cycles to query the status of the transmission process.
3. When the Send_P2P instruction signals that transmission is complete at the DONE output, the user code can prepare the receipt of the answer.
4. The Receive_P2P instruction is run repeatedly to query an answer. If the CM has acquired response data, the Receive_P2P instruction copies the response to the CPU and signals that new data has been received at the NDR output.
5. The user program can process the response.
6. Back to step 1 and repetition of the sequence.

Slave

Typical sequence for a slave:

1. The user program runs the Receive_P2P instruction in each cycle.
2. If the CM has received a request, the Receive_P2P signals that new data is available at the NDR output and the request is copied to the CPU.
3. The user program processes the request and creates a response.
4. The response is returned to the master with the Send_P2P instruction.
5. The Send_P2P instruction must be run repeatedly to ensure that the send process is actually taking place.
6. Back to step 1 and repetition of the sequence.

The slave must ensure that Receive_P2P is called up often enough so that a transmission can be received by the master before it cancels the process due to a timeout while waiting for the response. To do so, the user program Receive_P2P can be called from within a cycle OB whose cycle time is sufficiently short so that the master can receive a transmission before the timeout setting expires. If the OB cycle time is set so that two runs can take place during the timeout setting of the master, the user program can receive all transmissions without any losses.

General parameters for Freeport operations

Table 9-287 General input parameters of the Freeport instructions

Parameter	Description
REQ	Data transmission is initiated by a positive edge at the REQ input. Another edge at REQ may only be created after the command has been completed (DONE or ERROR). Data transmission can take several calls (program cycles), depending on the data volume. When you add a Freeport instruction to your program, STEP 7 prompts you to specify the instance DB (or to have STEP 7 create a corresponding instance DB). Use a unique DB for each PtP instruction call.
PORT	A port address is assigned during configuration of the communication module. The PORT parameter communicates assignment to a specific communication module to the instruction. You can select a symbolic name for the standard port after the configuration. The assigned CM port value is the "Hardware ID" property of the device configuration with S7-1500 and the "Input address" with S7-300/400. The symbolic port name is assigned in the symbol table.

The output parameters DONE, NDR, ERROR and STATUS of the Freeport instructions indicate the execution status of the Freeport functions.

Table 9-288 Output parameters DONE, NDR, ERROR and STATUS

Parameter	Data type	Standard	Description
DONE	Bool	FALSE	Set to TRUE for one cycle to indicate that the last request was completed with errors; otherwise FALSE.
NDR	Bool	FALSE	Set to TRUE for one cycle to indicate that new data has been received; otherwise FALSE.

Parameter	Data type	Standard	Description
ERROR	Bool	FALSE	Set to TRUE for one cycle to indicate that the last request was completed with errors; the corresponding error code can be found in STATUS; otherwise FALSE.
STATUS	Word	16#0000 or 16#7000	Result status: • If the bit DONE or NDR is set, STATUS is set to 0/16#7000 or to a specific error code. • If the bit ERROR is set, STATUS displays an error code. • If none of the bits listed above is set, the instruction can return status results that describe the current status of the function. The value in STATUS is valid until you call this instruction again (with one and the same port address).

Table 9-289 COM_RST in/out parameter

Parameter	Data type	Standard	Description
COM_RST	Bool	FALSE	Initialization of the instruction You initialize the instruction with TRUE. COM_RST is then reset to FALSE. Note: You must set COM_RST to TRUE during startup and should not subsequently change the parameter (that is, do not assign a value when you call the instruction). COM_RST is reset by the instruction following initialization of the instance DB.

Note

Please note that the parameters DONE, NDR, ERROR and STATUS are only set for one cycle.

Table 9-290 Shared error codes

Error code	Description
16#0000	No error
16#7000	Function not active
16#7001	Initial call after request started.
16#7002	Subsequent call after request started.
16#8x3A	Invalid pointer in parameter x

Table 9-291 Shared error classes of the STATUS parameter

Description of the class	Error classes	Description
Port configuration	16#81Ax	For the description of frequent errors in the interface configuration
Send configuration	16#81Bx	For the description of errors in the send configuration
Receive configuration	16#81Cx	For the description of errors in the receive configuration
Sending	16#81Dx	For the description of runtime errors during sending
Receiving	16#81Ex	For the description of runtime errors during receiving
RS232 accompanying signals	16#81Fx	For the description of errors in connection with signal processing

Port_Config: Configure PtP communication port

Description

The Port_Config instruction (port configuration) allows you to change parameters such as the data transmission rate in runtime using your program. The data pending in the CM is deleted with the execution of Port_Config.

Configuration changes of Port_Config are saved on the CM and not in the CPU. When the voltage returns, the CM is configured with the data saved in the device configuration.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the transmission of data to the CM upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	The CM port value can be found in the "Hardware ID" property of the device configuration (S7-1500) (input address; S7-300/400) once you have configured the CM. The symbolic port name is assigned in the "System constants" tab of the PLC tag table. In S7-300/400/WinAC systems, the input address assigned in HWCN must be assigned to the CM port.
PROTOCOL	IN	UInt	Word	0	Protocol • 0 = Freeport protocol • 1 = Protocol 3964(R)
BAUD	IN	UInt	Word	6	Data transmission rate of the port: • 1 = 300 Bit/s • 2 = 600 Bit/s • 3 = 1200 Bit/s • 4 = 2400 Bit/s • 5 = 4800 Bit/s • 6 = 9600 Bit/s • 7 = 19200 Bit/s • 8 = 38400 Bit/s • 9 = 57600 Bit/s • 10 = 76800 Bit/s • 11 = 115200 Bit/s

Parameter	Declaration	Data type		Default	Description
		S7-1500	S7-300/400/WinAC		
PARITY	IN	UInt	Word	1	Parity of the port: • 1 = no parity • 2 = even parity • 3 = odd parity • 4 = mark parity • 5 = space parity • 6 = any
DATABITS	IN	UInt	Word	1	Bits per character: • 1 = 8 data bits • 2 = 7 data bits
STOPBITS	IN	UInt	Word	1	Stop bits: • 1 = 1 stop bit • 2 = 2 stop bits
FLOWCTRL	IN	UInt	Word	1	Flow control: • 1 = no flow control • 2 = XON/XOFF • 3 = Hardware RTS always ON • 4 = Hardware RTS switched • 5 = Hardware RTS always ON, ignore DTR/DSR
XONCHAR	IN	Char		16#0011	Specifies the character that serves as XON character. It is typically a DC1 character (11H). This parameter is only evaluated when software flow control is active.
XOFFCHAR	IN	Char		16#0013	Specifies the character that serves as XOFF character. It is typically a DC3 character (13H). This parameter is only evaluated when software flow control is active.
WAITIME	IN	UInt	Word	2000	Specifies how long to wait for an XOFF character after receipt of an XON character or how long to wait for a CTS = ON signal after CTS = OFF (0 to 65535 ms). This parameter is only evaluated when flow control is active.
MODE	IN	USInt	Byte	0	Operating mode Valid operating modes are: • 0 = Full duplex (RS232) • 1 = Full duplex (RS422) four-wire operation (point-to-point) • 2 = Full duplex (RS 422) four-wire mode (multipoint master; CM PtP (ET 200SP)) • 3 = Full duplex (RS 422) four-wire mode (multipoint slave; CM PtP (ET 200SP)) • 4 = Half duplex (RS485) two-wire mode

Parameter	Declaration	Data type		Default	Description
		S7-1500	S7-300/400/WinAC		
LINE_PRE	IN	USInt	Byte	0	Receive line initial state Valid initial states are: • 0 = "no" initial state • 1 = signal R(A)=5 V, signal R(B)=0 V (break detection): Break detection is possible with this initial state. Can only be selected with: "Full duplex (RS422) four-wire operation (point-to-point connection)" and "Full duplex (RS422) four-wire operation (multipoint slave)". • 2 = signal R(A)=0 V, signal R(B)=5 V: This default setting corresponds to the idle state (no active send operation). No break detection is possible with this initial state.
CABLE_BRK	IN	USInt	Byte	0	Break detection The following settings are valid: • 0 = break detection deactivated • 1 = break detection activated
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Port_Config instruction The instruction is initialized with 1. The instruction then resets COM_RST to 0.
DONE	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed without errors
ERROR	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed with errors
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 7490))

Send_Config: Configure PtP sender

Description

The Send_Config instruction (send configuration) allows you to change send parameters in runtime using your program (conditions that identify the start and the end of the data to be sent). Any data pending in a CM is deleted when Send_Config is executed.

Configuration changes of Send_Config are saved on the CM and not in the CPU. The parameters saved in the device configuration are restored once the voltage returns to the CPU or the communication module.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Activates the configuration change upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	The CM port value can be found in the "Hardware ID" property of the device configuration (S7-1500) (input address; S7-300/400) once you have configured the CM. The symbolic port name is assigned in the "System constants" tab of the PLC tag table. In S7-300/400/WinAC systems, the input address assigned in HWCN must be assigned to the CM port.
RTSONDLY	IN	UInt	Word	0	Number of milliseconds to wait after activation of RTS before a transmission of send data is started. This parameter is only valid if the hardware flow control is active. The valid range is 0 to 65535 ms. The value 0 deactivates the function.
RTSOFFDLY	IN	UInt	Word	0	Number of milliseconds to wait after transmission of send data before RTS is deactivated: This parameter is only valid if the hardware flow control is active. The valid range is 0 to 65535 ms. The value 0 deactivates the function.
BREAK	IN	UInt	Word	0	This parameter specifies that a BREAK is to be sent at the start of each frame for the specified number of bit times. The maximum is 65535 bit times. The value 0 deactivates the function.
IDLELINE	IN	UInt	Word	0	This parameter specifies that the line is to remain in idle for the specified number of bit times prior to the start of each frame. The maximum is 65535 bit times. The value 0 deactivates the function.
USR_END	IN	STRING[2]		0	Input of end delimiters. No more than 2 end delimiters can be configured. All data including the end delimiter(s) is sent, independent of the configured frame length.
APP_END	IN	STRING[5]		0	Input of characters to be appended. You can append up to 5 characters.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Send_Config instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
DONE	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed without errors
ERROR	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed with errors
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 7490))

Receive_Config: Configure PtP recipient

Description

The Receive_Config instruction (receive configuration) allows you to change receive parameters in runtime using your program. This instruction configures the conditions that mark the start and the end of received data. Any data pending in a CM is deleted when Receive_Config is executed.

Configuration changes of Port_Config are saved on the CM and not in the CPU. The parameters saved in the device configuration are restored once the voltage returns to the CPU or the communication module.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Activates the configuration change upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	The CM port value can be found in the "Hardware ID" property of the device configuration (S7-1500) (input address; S7-300/400) once you have configured the CM. The symbolic port name is assigned in the "System constants" tab of the PLC tag table. In S7-300/400/WinAC systems, the input address assigned in HWCN must be assigned to the CM port.
Receive_Conditions	IN	Variant	Any	-	The data structure of Receive_Conditions specifies the start and end conditions used to identify the start and end of a frame.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Receive_Config instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
DONE	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed without errors
ERROR	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed with errors
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 7490))

.

Start conditions for the Receive_P2P instruction

The Receive_P2P instruction uses the configuration specified in the device configuration or by the Receive_Config instruction to determine the start and end of Freeport communication frames. The start of the frame is defined by the start conditions. The start of the frame can be determined with one or several start conditions.
If Break as well as Idle Line is activated, Break must be met first and then Idle Line as well. After that, one of the other conditions (start character or start sequence) is sufficient to start

data transmission.
The start condition "Random character" cannot be combined with other start conditions.

Data type structure of the Receive_Conditions parameter, part 1 (start conditions)

Table 9-292 Structure of Receive_Conditions for start conditions

Parameter	Declaration	Data type	Default	Description
START.START COND	IN	Word	16#0002	Specifying the start condition • 01H - detection of the start character • 02H - Any character • 04H - detection of a line break • 08H - detection of an idle line • 10H - detection of start sequence 1 • 20H - detection of start sequence 2 • 40H - detection of start sequence 3 • 80H - detection of start sequence 4 The start conditions can be combined by adding the values together.
START.IDLETI ME	IN	Word	16#0028	The number of bit times required in the idle state for a new frame start to be detected (default value: W#16#28). Only in connection with the condition "Detection of an idle line". 0 to FFFF
START.START CHAR	IN	Byte	16#0002	The start character for the condition "Start character". (default value: B#16#2)
START.SEQ[1]. CTL	IN	Byte	0	Start sequence 1, deactivate/activate comparison for each character: (default value: B#16#0) These are the activation bits for each character of the start character string. • 01H - character 1 • 02H - character 2 • 04H - character 3 • 08H - character 4 • 10H - character 5 When a bit is deactivated for a specific character, this means that each character in this position in the character string represents a valid start character string (e.g. 1FH = all 5 characters interpreted).
START.SEQ[1]. STR[1] .. START.SEQ[1]. STR.[5]	IN	Char[5]	0	Start sequence 1, start character (5 characters).
START.SEQ[2]. CTL	IN	Byte	0	Start sequence 2, deactivate/activate comparison for each character. Default value: B#16#0)
START.SEQ[2]. STR[1] .. START.SEQ[2]. STR.[5]	IN	Char[5]	0	Start sequence 2, start character (5 characters).

Parameter	Declaration	Data type	Default	Description
START.SEQ[3]. CTL	IN	Byte	0	Start sequence 3, deactivate/activate comparison for each character. Default value: B#16#0
START.SEQ[3]. STR[1] .. START.SEQ[3]. STR.[5]	IN	Char[5]	0	Start sequence 3, start character (5 characters).
START.SEQ[4]. CTL	IN	Byte	0	Start sequence 4, deactivate/activate comparison for each character. Default value: B#16#0
START.SEQ[4]. STR[1] .. START.SEQ[4]. STR.[5]	IN	Char[5]	0	Start sequence 4, start character (5 characters),

Example

Have a look at the following received data in hexadecimal coding: "68 10 aa 68 bb 10 aa 16". The configured start character strings are available in the following table. Start character strings are evaluated once the first character 68H has been successfully received. After the fourth character has been successfully received (the second 68H), start condition 1 has been met. Once the start conditions have been met, the evaluation of the end conditions starts.

Processing of the start character string can be canceled due to different errors in parity, framing or time intervals between characters. These errors prevent receipt of the data because the start condition has not been met (an error message is output).

Table 9-293 Start conditions

Start condition	First character	First character +1	First character +2	First character +3	First character +4
1	68H	xx	xx	68H	xx
2	10H	aaH	xx	xx	xx
3	dcH	aaH	xx	xx	xx
4	e5H	xx	xx	xx	xx

End conditions for the Receive_P2P instruction

The end of a frame is defined by the first occurrence of one or more configured end conditions.

You can configure the end conditions either in the properties of the communication interface in the device configuration, or with the Receive_Config instruction. The receive parameters (start and end conditions) are reset to the settings in the device configuration each time the voltage returns to the CPU or the communication module. When the STEP 7 user program executes Receive_Config, the settings are set to the parameters of Receive_Config .

Data type structure of the Receive_Conditions parameter, part 2 (end conditions)

Table 9-294 Structure of Receive_Conditions for end conditions

Parameter	Declaration	Data type	Default	Description
END.ENDCOND	IN	Word	0	This parameter specifies the condition for the frame end: • 01H - response timeout • 02H - message timeout • 04H - character delay time • 08H - maximum frame length • 10H - read message length from message (N+LEN+M) • 20H - end sequence • 40H - fixed frame length
END.FIXLEN	IN	Word	1	Fixed frame length: Only used if the end condition "Fixed length" has been selected. 1 to 4000 bytes (up to 4 KB depending on the module)
END.MAXLEN	IN	Word	1	Maximum frame length: Only used if the end condition "Maximum length" has been selected. 1 to 4000 bytes (up to 4 KB depending on the module)
END.N	IN	Word	0	Byte position of the length field in the frame. Only used with end condition N+LEN+M. 1 to 4000 bytes (up to 4 KB depending on the module)
END.LENGTHSIZE	IN	Word	0	Size of the length field (1, 2, or 4 bytes). Only used with end condition N+LEN+M.
END.LENGTHM	IN	Word	0	Number of characters after the length field that are not included in the value of the length field. This entry is only used with end condition N+LEN+M. 0 to 255 bytes
END.RCVTIME	IN	Word	200	Specify the wait time for the first received character after a frame has been sent. The receive instruction is terminated with an error message if a character is not received within the specified time. This information is used only with the condition "Response timeout". (0 to 65535 ms). Note: This parameter cannot be used as sole end criterion but only in connection with at least one other end condition.
END.MSGTIME	IN	Word	200	Specify how long to wait for receipt of the complete frame after receipt of the first character. This parameter is used only if the condition "Message timeout" is selected. (0 to 65535 ms)
END.CHARGAP	IN	Word	12	Enter the maximum number of bit times between characters. If the number of bit times between characters exceeds the specified value, the end condition has been met. This information is used only with the condition "Character delay time". (0 to 65535 bit times)

Parameter	Declaration	Data type	Default	Description
END.SEQ.CTL	IN	Byte	0	Character sequence 1, deactivate/activate comparison for each character: These are the activation bits for each character of the end character string. Character 1 is bit 0, character 2 is bit 1, ..., character 5 is bit 4. If a bit is deactivated for a specific character, this means that each character represents a congruence at this position of the character string.
END.SEQ.STR[1] .. END.SEQ.STR[5]	IN	Char[5]	0	Character string 1, start character (5 characters)

Table 9-295 General parameters of the Receive_P2P instruction

Parameter	Declaration	Data type	Default	Description
GENERAL.MBUF_SIZE	IN	Byte	255	Input number of frames that are to be buffered in the receive buffer of the CM. If no other conditions are active that influence the reaction of the receive buffer (prevent timeout, data flow control), additional frames are discarded once the limit has been reached. (1 to 255 frames)
GENERAL.OW_PROT	IN	Byte	0	Activates the no overwriting function of the buffered frame if the CM receives a new frame and the receive buffer of the CM was not yet read. This step prevents already buffered received frames from being lost. • 0 - not activated • 1 - activated
GENERAL.CLR_MBUF	IN	Byte	0	Activates deletion of the receive buffer during CPU startup. The receive buffer is automatically deleted when the CPU switches from STOP to RUN. The receive buffer only contains frames received after CPU startup. • 0 - not activated • 1 - activated

P3964_Config: Configuring the 3964(R) protocol

Description

The P3964_Config instruction (protocol configuration) allows you to change protocol parameters for 3964(R), such as character delay time, priority and block check, in runtime using your program.

Configuration changes of P3964_Config are saved on the CM and not in the CPU. The parameters saved in the device configuration are restored once the voltage returns to the CPU or the communication module.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the instruction upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	The CM port value can be found in the "Hardware ID" property of the device configuration (S7-1500) (input address; S7-300/400) once you have configured the CM. The symbolic port name is assigned in the "System constants" tab of the PLC tag table. In S7-300/400/WinAC systems, the input address assigned in HWCN must be assigned to the CM port.
BCC	IN	SInt	Byte	1	Activates/deactivates the use of the block check • 0 = without block check • 1 = with block check
Priority	IN	SInt	Byte	1	Selection of the priority • 0 = low priority • 1 = high priority
CharacterDelayTime	IN	UInt	Word	16#00DC	Setting the character delay time (depending on the set data transmission rate) (default value: 220 ms) 1 ms to 65535 ms
AcknDelayTime	IN	UInt	Word	16#07D0	Setting the acknowledgment delay time (depending on the set data transmission rate) (default value: 2000 ms) 1 ms to 65535 ms
BuildupAttempts	IN	SInt	Byte	16#0006	Setting the number of connection attempts (default value: 6 connection attempts) 1 to 255
RepetitionAttempts	IN	SInt	Byte	16#0006	Setting the number of transmission attempts (default value: 6 connection attempts) 1 to 255
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the P3964_Config instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
DONE	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed without errors
ERROR	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed with errors
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 7490))

Send_P2P: Sending data

Description

The Send_P2P instruction (send point-to-point data) starts the transmission of data and transmits the contents of the assigned buffer to the communication module. The CPU program is still being executed while the CM sends the data with the data transmission rate. Only one

send instruction per communication module may be pending at any time. The CM signals an error if a second Send_P2P instruction is executed while the CM is already sending a frame.

Parameters

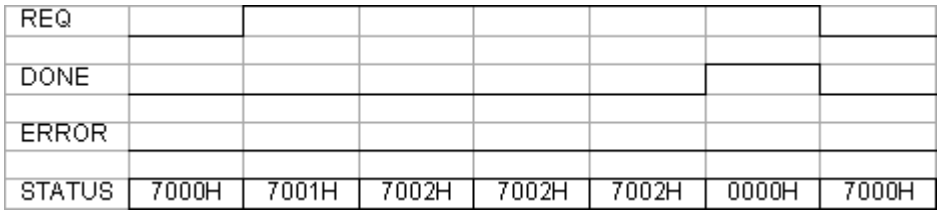
Parameter	Declaration	Data type		Default	Description
		S7-1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the transmission of data to the CM upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	The CM port value can be found in the "Hardware ID" property of the device configuration (S7-1500) (input address; S7-300/400) once you have configured the CM. The symbolic port name is assigned in the "System constants" tab of the PLC tag table. In S7-300/400/WinAC systems, the input address assigned in HWCN must be assigned to the CM port.
BUFFER	IN	Variant	Any	0	This parameter points to the memory area of the send buffer. Notes: • Boolean data and Boolean fields are not supported. • If the send buffer is in the optimized memory area, the maximum permitted length of the sent data is 1024 bytes. Exception: Arrays of Byte, Word or DWord are supported up to a length of 4096 bytes. • If the send buffer is a String or WString, the content of the string is transferred without the current and maximum length.
LENGTH	IN	UInt	Word	0	Length in bytes of the data to be transferred. The memory area addressed in the BUFFER parameter is completely transmitted with LENGTH = 0.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Send_P2P instruction The instruction is initialized with 1. The instruction then resets COM_RST to 0.
DONE	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed without errors
ERROR	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed with errors
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 7490))

Parameters

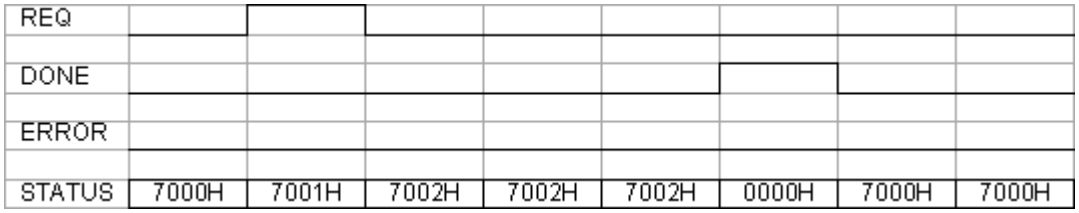
The DONE and ERROR outputs are in FALSE status when a send instruction is being processed. At the end of the send instruction, one of the DONE or ERROR outputs is set to TRUE for one cycle to signal the status of the send instruction. The error code at the STATUS output can be evaluated when the status of ERROR is TRUE.

The instruction outputs the status 16#7001 when the communication interface accepts the send data. Subsequent executions of Send_P2P output the value 16#7002 if the CM is still sending. At the end of the send instruction, the CM outputs the status 16#0000 for the send instruction (if no error has occurred). Subsequent executions of Send_P2P with REQ = 0 output the status 16#7000 (free).

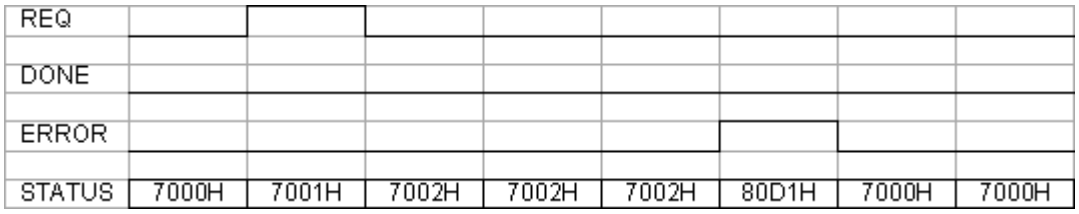
The diagram below shows the relationship between the output values and REQ. It is based on the assumption that the instruction is called cyclically to check the status of the send process (indicated by the STATUS values).



The figure below shows how the DONE and STATUS parameters are only valid for one cycle if a pulse is pending at the REQ line (for one cycle) to trigger the send instruction.



The figure below shows the relationship of the DONE, ERROR and STATUS parameters in case of an error.



The DONE, ERROR and STATUS values are only valid until Send_P2P is executed again with the same instance DB.

Using LENGTH and BUFFER parameters in communication operations

Interaction of LENGTH and BUFFER parameters for Send_P2P

The minimum data size sent by the Send_P2P instruction is one byte. The BUFFER parameter specifies the size of the data to be sent if a "0" is passed at the LENGTH parameter during

call. You cannot use the data type Bool or arrays of the Bool type for the BUFFER parameter.

Table 9-296 LENGTH parameter

LENGTH	Description
> 0	Transfers the configured number of bytes.
= 0	Transfers the entire contents of the send buffer addressed with BUFFER. If BUFFER points to a string, the entire string contents are transferred except for the bytes with the maximum and actual length.

Table 9-297 BUFFER parameter

BUFFER	Description
Elementary data type	When sending: The LENGTH value must include the byte size of this data type. Example: For a Word value, the LENGTH must be two. For a DWord value or Real value, the LENGTH must be four.
Structure	For optimized memory: The maximum permitted length of the BUFFER is 1024 Byte; otherwise 4 KB are permitted depending on the module. For sending: The LENGTH value can include a byte size smaller than the complete byte length of the structure; in this case, only the first LENGTH bytes of the structure from BUFFER are sent.
Array	For optimized memory: If the array data type is not equal to Byte, Word or DWord, the maximum permitted buffer length is 1024 Byte. Depending on the module, up to 4 KB can be transmitted if the memory is not optimized, independent of the data structure. For sending: The LENGTH value can include a byte size smaller than the complete byte length of the array, whereby, this byte size is a multiple of the byte size of the data element. Example: The LENGTH parameter of an array of the Word type must be a multiple of two and a multiple of four for an array of the Real type. If LENGTH is specified, the number of array elements is transmitted. If BUFFER includes an array with 15 DWord elements (a total of 60 Bytes), for example, and you specify LENGTH = 20, the first five DWord elements from the array are transmitted.
String	The LENGTH parameter includes the number or characters to be sent. Only the characters of the String are transmitted. The bytes with the maximum and actual length of the String are not sent.

Receive_P2P: Receiving data

Description

The Receive_P2P instruction (receive data using point-to-point communication) checks the frames received in the CM. If a frame is available, it is transmitted from the CM to the CPU. A receive error is indicated at the STATUS parameter.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1500	S7-300/400/WinAC		
PORT	IN	PORT (UInt)	Word	0	The CM port value can be found in the "Hardware ID" property of the device configuration (S7-1500) (input address; S7-300/400) once you have configured the CM. The symbolic port name is assigned in the "System constants" tab of the PLC tag table. In S7-300/400/WinAC systems, the input address assigned in HWCN must be assigned to the CM port.
BUFFER	IN	Variant	Any	0	This parameter points to the start address of the receive buffer. This buffer must be large enough to receive the maximum frame length. **Note:** • Boolean data or Boolean fields are not supported. • If the receive buffer is in the optimized memory area, the maximum permitted length of the received data is 1024 bytes. Exception: Arrays of Byte, Word or DWord are supported up to a length of 4096 bytes. • If the receive buffer is a String or WString, the received data is written to the content of the string and the current length of the string is set accordingly.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Receive_P2P instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
LENGTH	OUT	UInt	Word	0	Length of the frame received in bytes
NDR	OUT	Bool		FALSE	TRUE for one cycle if new data is available and the instruction has been completed without errors.
ERROR	OUT	Bool		FALSE	TRUE for one cycle once the instruction has been completed with an error.
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 7490))

The error code at the STATUS output can be evaluated if the status of ERROR is TRUE. The STATUS value provides the reason for terminating the receive operation in the CM.
This is usually a positive value which indicates that the receive operation has been successful and the frame criterion that has been detected.
If the STATUS value is negative (the most significant bit of the hexadecimal value is set), the receive operation was terminated due to an error condition, such as a parity, framing or overflow error.

Each communication module can buffer a module-specific number for frames. If several frames are available in the CM, the Receive_P2P instruction outputs the oldest available frame (FIFO).

Receive_Reset: Clear receive buffer

Description

The Receive_Reset instruction (reset receiver) clears the receive buffer in the CM.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the transmission of data to the CM upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	The CM port value can be found in the "Hardware ID" property of the device configuration (S7-1500) (input address; S7-300/400) once you have configured the CM. The symbolic port name is assigned in the "System constants" tab of the PLC tag table. In S7-300/400/WinAC systems, the input address assigned in HWCN must be assigned to the CM port.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Receive_Reset instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
DONE	OUT	Bool		FALSE	TRUE for one cycle means that the last request was completed without errors.
ERROR	OUT	Bool		FALSE	TRUE means that the last request was completed with errors. If this output is TRUE, the STATUS output contains the corresponding error codes.
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 7490))

Signal_Get: Read status

Description

The Signal_Get instruction (get RS232 signals) reads the current states of the RS232 accompanying signals and displays them at the corresponding instruction outputs.

Note
Restriction

This function can only be used with CMs RS232 BA and RS232 HF.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the transmission of data to the CM upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	The CM port value can be found in the "Hardware ID" property of the device configuration (S7-1500) (input address; S7-300/400) once you have configured the CM. The symbolic port name is assigned in the "System constants" tab of the PLC tag table. In S7-300/400/WinAC systems, the input address assigned in HWCN must be assigned to the CM port.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Signal_Get instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
NDR	OUT	Bool		FALSE	TRUE for one cycle if the RS232 accompanying signals have been read and the instruction has been completed without errors.
ERROR	OUT	Bool		FALSE	TRUE for one cycle once the instruction has been completed with an error
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 7490))
DTR	OUT	Bool		FALSE	Data device ready, module ready (output)
DSR	OUT	Bool		FALSE	Data device ready, communication station ready (input)
RTS	OUT	Bool		FALSE	Send request, module ready to send (output)
CTS	OUT	Bool		FALSE	Ready to send, communication station can receive data (input)
DCD	OUT	Bool		FALSE	Data carrier signal detected, signal level received
RING	OUT	Bool		FALSE	Call display, signaling incoming call

Signal_Set: Set accompanying signals

Description

The Signal_Set instruction (set RS232 signals) allows you to set the RS232 communication signals.

This function is only valid for RS232 CM.

Note

Restrictions

This function can only be used with CMs RS232 BA and RS232 HF.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the instruction upon a positive edge of this input.
PORT	IN	PORT (UInt)	Word	0	The CM port value can be found in the "Hardware ID" property of the device configuration (S7-1500) (input address; S7-300/400) once you have configured the CM. The symbolic port name is assigned in the "System constants" tab of the PLC tag table. In S7-300/400/WinAC systems, the input address assigned in HWCN must be assigned to the CM port.
SIGNAL	IN	Byte		0	Selection of the signal to be set (more than one possible): • 01H = RTS • 02H = DTR • 04H = DSR (for interface type DCE only)
RTS	IN	Bool		FALSE	Send request, module ready to send Set this value at the output (TRUE or FALSE), default value: FALSE
DTR	IN	Bool		FALSE	Data terminal ready, module ready Set this value at the output (TRUE or FALSE), default value: FALSE
DSR	IN	Bool		FALSE	Data terminal ready (for DCE interface type only), not used.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Signal_Set instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
DONE	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed without errors
ERROR	OUT	Bool		FALSE	TRUE for one cycle after the last request has been completed with errors
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 7490))

.

Get_Features: Get extended functions

Description

If supported by the module, you can use the Get_Features instruction (get extended functions) to get information on the ability of the module to support CRC and to generate diagnostic messages.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the instruction upon a positive edge of this input.
PORT	IN	PORT	Word	0	The CM port value can be found in the "Hardware ID" property of the device configuration (S7-1500) (input address; S7-300/400) once you have configured the CM. The symbolic port name is assigned in the "System constants" tab of the PLC tag table. In S7-300/400/WinAC systems, the input address assigned in HWCN must be assigned to the CM port.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Get_Features instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
NDR	OUT	Bool		FALSE	TRUE for one cycle if new data is available and the instruction has been completed without errors
MODBUS_CRC	OUT	Bool		FALSE	Modbus CRC support
DIAG_ALARM	OUT	Bool		FALSE	Generation of diagnostic messages
SUPPLY_VOLT	OUT	Bool		FALSE	Diagnostics for missing supply voltage L+ is available
ERROR	OUT	Bool		FALSE	TRUE for one cycle once the instruction has been completed with an error
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 7490))

Set_Features: Set extended functions

Description

If supported by the module, you can use the Set_Features instruction (select extended functions) to activate CRC support and the generation of diagnostic messages.

Parameters

Parameter	Declaration	Data type		Default	Description
		S7-1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	The instruction to set extended functions is started upon a positive edge at this input.
PORT	IN	PORT (UInt)	Word	0	The CM port value can be found in the "Hardware ID" property of the device configuration (S7-1500) (input address; S7-300/400) once you have configured the CM. The symbolic port name is assigned in the "System constants" tab of the PLC tag table. In S7-300/400/WinAC systems, the input address assigned in HWCN must be assigned to the CM port.
EN_MODBUS_CRC	IN	Bool		FALSE	Activate Modbus CRC support
EN_DIAG_ALARM	IN	Bool		FALSE	Activate generation of diagnostic messages
EN_SUPPLY_VOLT	IN	Bool		FALSE	Enable diagnostics for missing supply voltage L+
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Set_Features instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
DONE	OUT	Bool		FALSE	TRUE for one execution once the last request is completed without errors
ERROR	OUT	Bool		FALSE	TRUE for one cycle once the instruction has been completed with an error
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 7490))

Error messages

Overview of error messages - PtP

The error messages are provided at the STATUS output of an instruction and can be evaluated there or processed in the user program.

Error code	Description	Remedy
16#0000	No error	-
RECEIVE status and error codes		
16#0094	frame end identified based on the "Receipt of fixed/maximum frame length"	-
16#0095	frame end identified based on "Message timeout"	-
16#0096	frame end identified based on expiration of the "Character delay time"	-
16#0097	The frame was aborted because the maximum response time was reached.	-

Error code	Description	Remedy
16#0098	frame end identified based on the fulfillment of the "Read message length from message" conditions	-
16#0099	frame end identified based on the receipt of the "End sequence"	-
SEND status and error codes		
16#7000	Block idle	-
16#7001	Initial call for a new frame: Data transmission initiated	-
16#7002	Interim call: Data transmission running	-
16#8085	Invalid length	Select a suitable frame length. The following are valid (depending on the module): 1-1024/2048/4096 (Byte)
16#8088	The specified length exceeds the range set in the receive buffer. Note: If the data type STRING has been specified at the BUFFER parameter, this error code also appears if the current string is shorter than the length specified at the LENGTH parameter.	Change the range in the receive buffer or select a frame length which corresponds to the range set in the receive buffer. The following are valid (depending on the module): 1-1024/2048/4096 (Byte)
16#8090	Configuration error: Odd number of bytes for WString	Select an even number of bytes.
RECEIVE status and error codes		
16#8088	The number of characters received exceeds the number specified at the BUFFER parameter.	Select a suitable frame length. The following are valid (depending on the module): 1-1024/2048/4096 (Byte)
16#8090	Configuration error: Odd number of bytes for WString	Select an even number of bytes.
Error message codes of the special functions		
16#818F	Incorrect parameter number setting (with USS only)	Select a suitable parameter number (PARAM). The following numbers are valid: 0-2047
16#8190	Incorrect setting of the CRC calculation	Select a suitable value for the CRC calculation. The following are valid: deactivated or activated. Check whether the module addressed supports CRC calculation.
16#8191	Incorrect setting of the diagnostic error interrupt	Select a suitable value for "Diagnostic error interrupt". The following are valid: Diagnostic error interrupt deactivated or diagnostic error interrupt activated. Check whether the module addressed supports the generation of diagnostic interrupts.
Error message codes of the "Port configuration"		
16#81A0	The module does not support this protocol.	Select a valid protocol for the module (PROTOCOL).
16#81A1	The module does not support this data transmission rate.	Select a valid data transmission rate for the module (BAUD).

Error code	Description	Remedy
16#81A2	The module does not support this parity setting.	Select a suitable value for "Parity" (PARITY). The following are valid: • None (1) • Even (2) • Odd (3) • Mark (4) • Space (5) • Any (6)
16#81A3	The module does not support this number of data bits.	Select a suitable value for "Number of data bits" (DATABITS). The following are valid: • 7 (2) • 8 (1)
16#81A4	The module does not support this number of stop bits.	Select a suitable value for "Number of stop bits" (STOPBITS). The following are valid: • 1 (1) • 2 (2)
16#81A5	The module does not support this type of data flow control.	Select a valid data flow control for the module (FLOWCTRL).
16#81A7	Invalid value for XON or XOFF	Select suitable values for XON (XONCHAR) and XOFF (XOFFCHAR). Valid range of values: 0...255
16#81AA	Invalid operating mode	Valid operating modes are: • Full duplex (RS232) (0) • Full duplex (RS422) four-wire mode (point-to-point) (1) / (CM PtP (ET 200SP)) • Full duplex (RS422) four-wire mode (multipoint master) (2)/ (CM PtP (ET 200SP)) • Full duplex (RS 422) four-wire mode (multipoint slave) (3) • Half duplex (RS485) two-wire mode. (4)
16#81AB	Invalid receive line initial state	Valid initial states are: • "No" default setting (0) • Signal R(A)=5 V, signal R(B)=0 V (break detection) (1): Can only be selected with: "Full duplex (RS422) four-wire operation (point-to-point connection)" and "Full duplex (RS422) four-wire operation (multipoint slave)". • Signal R(A)=0 V, signal R(B)=5 V (2): This default setting corresponds to the idle state (no active send operation).
16#81AC	Invalid value for "Break detection"	Select a suitable value for "Break detection". The following are valid: • Break detection deactivated (0) • Break detection activated (1).

Error code	Description	Remedy
16#81AF	The module does not support this protocol.	Select a valid protocol for the module.
Error codes of the "Send configuration"		
16#81B5	More than two end delimiters or end sequence > 5 characters	Select suitable values for "End delimiter" and "End sequence". The following are valid: • deactivated (0), • 1 (1) or 2 (2) end delimiters or • deactivated (0), • 1 (1) up to 5 (5) characters for the end sequence.
16#81B6	Send configuration rejected because the 3964(R) protocol was selected	Make sure that no send configuration is transmitted if the 3964(R) protocol is set.
Error codes of the "Receive configuration"		
16#81C0	Invalid start condition	Select a suitable start condition. The following are valid: • Send break before frame start • Send Idle Line.
16#81C1	Invalid end condition or no end condition selected	Select a suitable end condition (see Receiving data with Freeport (Page 991)).
16#81C3	Invalid value for "Maximum message length"	Select a suitable value for "Maximum message length" (MAXLEN). Valid range of values (depending on the module): 1-1024/2048/4096 (Byte)
16#81C4	Invalid value for "Offset of the length specification in the message"	Select a suitable value for "Offset of the length specification in the message". Valid range of values (depending on the module): 1-1024/2048/4096 (Byte)
16#81C5	Invalid value for "Size of length field"	Select a suitable value for "Size of length field" (LENGTHSIZE). Valid range of values in bytes: • 1 (1) • 2 (2) • 4 (4)
16#81C6	Invalid value for "Number of characters not counted in length specification"	Select a suitable value for "Number of characters not counted in length specification" (LENGTHM). Valid range of values: 0 to 255 (bytes)
16#81C7	The total of "Offset in the message + size of length field + number of characters not counted" is greater than the maximum frame length	Select a suitable value for "Offset in message", "Size of length field" and "Number of characters not counted". Valid range of values: • Offset in the message (depending on the module): 0-1024/2048/4096 (Byte) • Size of length field: 1, 2, or 4 (bytes) • Number of characters not counted: 0 to 255 (bytes)

Error code	Description	Remedy
16#81C8	Invalid value for "Response timeout"	Select a suitable value for "Response timeout". Valid range of values: 1-65535 (ms)
16#81C9	Invalid value for "Character delay time"	Select a suitable value for "Character delay time". Valid range of values: 1 to 65535 (bit times)
16#81CB	frame end sequence is activated, but no character is activated for the check	Activate one or several characters for the check.
16#81CC	frame start sequence is activated, but no character is activated for the check	Activate one or several characters for the check.
16#81CD	Invalid value for "Prevent overwriting"	Select a suitable value for "Prevent overwriting". The following are valid: • Prevent overwriting is deactivated (0) or • Prevent overwriting is activated (1)
16#81CE	Invalid value for "Clear receive buffer on startup"	Select a suitable value for "Clear receive buffer on startup". The following are valid: • Clear receive buffer at startup is deactivated (0) • Clear receive buffer at startup is activated (1)
SEND status and error codes		
16#81D0	Receiving send request during runtime of a send command	Make sure that you do not receive an additional send request during runtime of a send command.
16#81D1	The waiting time for XON or CTS = ON has expired.	The communication partner has a fault, is too slow or is offline. Check the communication partner or change the parameters, if necessary.
16#81D2	"Hardware RTS always ON": Send job canceled due to change from DSR = ON to OFF	Check the communication partner. Make sure that DSR is ON for the entire duration of transmission.
16#81D3	Send buffer overflow / send frame too long	Select a shorter frame length. The following are valid (depending on the module): 1 to 1024/2048/4096 (bytes)
16#81D5	Transmission canceled due to parameter changes, detected wire break, or CPU in STOP	Check the parameter assignment, wire break, and CPU status.
16#81D6	Transmission canceled because end identifier was not received	Check the parameter assignment of the end characters and the frame of the communication partner.
16#81D7	Communication error between the user program and module	Check the communication (e.g., matching the sequence number).
16#81D8	Transmission attempt rejected because module is not configured	Configure the module.
Error codes of the receive configuration		
16#81E0	Frame aborted: Send buffer overflow / send frame too long	Increase the call rate for the receive function in the user program or configure communication with data flow control.
16#81E1	Frame aborted: Parity error	Check the connection line of the communication partners, or verify that both devices are configured for the same data transmission rate, parity and stop bit number.
16#81E2	Frame aborted: Character frame error	Check the settings for start bit, data bits, parity bit, data transmission rate, and stop bit(s).

Error code	Description	Remedy
16#81E3	Frame aborted: Character overflow error	Firmware error: Please contact Customer Support.
16#81E4	Frame aborted: The total length of "Offset in the message + size of the length field + number of characters not counted" is greater than the receive buffer	Select a suitable value for offset in message, size of the length field, and number of characters not counted.
16#81E5	Frame aborted: Break	Receive line to partner is interrupted. Reconnect or switch on partner.
16#81E6	Maximum number of "Buffered received frames" exceeded	In the user program call the instruction more often or configure a communication with data flow control or increase the number of buffered frames.
16#81E7	Synchronization error module and Receive_P2P	Make sure that different instances of the Receive_P2P do not access the same module.
16#81E8	Frame aborted: The character delay time has expired before the message end criterion was detected	Partner device faulty or too slow. Check this, if required, using an interface tester that is interconnected in the transmission line.
16#81E9	Modbus CRC error (only communication modules which support Modbus)	Checksum error of the Modbus frame. Check the communication partner.
16#81EA	Modbus frame too short (only communication modules which support Modbus)	Minimum length of Modbus frame not met. Check the communication partner.
16#81EB	Frame aborted: Maximum frame length reached	Select a shorter frame length at the communication partner. The following are valid (depending on the module): 1-1024/2048/4096 (Byte) Check the parameters for end of frame detection.
Error codes V24 accompanying signals		
16#81F0	The module does not support V24 accompanying signals	You have tried to set accompanying signals for a module that does not support V24 accompanying signals. Make sure that this is an RS232 module or that RS232 mode (ET 200SP) is set.
16#81F1	No operation of the V24 accompanying signals	The V24 accompanying signals cannot be operated manually if hardware data flow control is active.
Error codes of the receive configuration		
16#8201 [1)]	Receive_Conditions is a pointer to an invalid data type	Enter a pointer to one of the following data types: DB, BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TIME_OF_DAY, TIME, S5TIME, DATE_AND_TIME, STRING
16#8225	Receive_Conditionspoints to an optimized memory area greater than 1 kB or Receive_Conditions points to an optimized memory area and the receive length is greater than the area addressed by Receive_Conditions.	Enter a pointer to an area with a maximum length of: • Optimized memory area: 1 kByte • Non-optimized area: 4 kByte Note: If the pointer points to an optimized memory area, do not send more than 1 kByte.
16#8229 [1)]	Receive_Conditions is a pointer to BOOL with a number of bits not equal to n * 8	If you are using a pointer to BOOL, the number of bits must be a multiple of 8.
Error codes, general		

Error code	Description	Remedy
16#8280	Negative acknowledgment when reading module	You can find more detailed information on error causes in theRDREC.STATUS static parameters and in the description of the SFB RDREC. • Check the input at the PORT parameter • Set the COM_RST parameter prior to the initial call.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter You can find more detailed information on error causes in the WRREC.STATUS static parameters and in the description of the SFB WRREC.
16#8282	Module not available	Check the input at the PORT parameter and ensure that the module can be reached.
Error codes of the receive configuration		
16#82C1	Invalid value for "Buffered received frames".	Select a suitable value for "Buffered received frames". Valid range of values: 1-255
16#82C2	Receive configuration rejected because the 3964(R) protocol was selected	Make sure that no receive configuration is sent if the 3964(R) protocol is set.
16#8301 1)	Receive_Conditions is a pointer to an invalid data type	Select a valid data type. The following are valid: DB, BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TIME_OF_DAY, TIME, S5TIME, DATE_AND_TIME, STRING
16#8322	Range length error when reading a parameter	Check the input at the Receive_Conditions parameter
16#8324	Range error when reading a parameter	Check the input at the Receive_Conditions parameter
16#8328	Setting error when reading a parameter	Check the input at the Receive_Conditions parameter
SEND status and error codes		
16#8328 1)	BUFFER is a pointer to BOOL with a number of bits not equal to n * 8	If you are using a pointer to BOOL, the number of bits must be a multiple of 8.
Error codes of the receive configuration		
16#8332	Invalid data block at the Receive_Conditions parameter	Check the input at the Receive_Conditions parameter
16#833A	The designation of the data block at the Receive_Conditions parameter refers to a data block which is not loaded.	Check the input at the Receive_Conditions parameter
16#8351	Invalid data type	Check the input at the Receive_Conditions parameter
16#8352 1)	Receive_Conditions does not point to a data block	Check the pointer to Receive_Conditions
16#8353 1)	Receive_Conditions does not point to a structure of the type Receive_Conditions	Check the pointer to Receive_Conditions
Error codes 3964(R) protocol		
16#8380	Parameter assignment error: Invalid value for "Character delay time".	Select a suitable value for "Character delay time" (CharacterDelayTime). Valid range of values: 1 to 65535 (ms)
16#8381	Parameter assignment error: Invalid value for "Response timeout".	Select a suitable value for "Response timeout" (AcknDelayTime). Valid range of values: 1 to 65535 (ms)

Error code	Description	Remedy
16#8382	Parameter assignment error: Invalid value for "Priority".	Select a suitable value for "Priority" (Priority). The following are valid: • High (1) • Low (0)
16#8383	Parameter assignment error: Invalid value for "Block check"	Select a suitable value for "Block check" (BCC). The following are valid: • With block check (1) • Without block check (0)
16#8384	Parameter assignment error: Invalid value for "Connection attempts".	Select a suitable value for "Connection attempts" (BuildupAttempts). Valid range of values: 1-255
16#8385	Parameter assignment error: Invalid value for "Transmission attempts".	Select a suitable value for "Transmission attempts" (RepetitionAttempts). Valid range of values: 1-255
16#8386	Runtime error: Number of connection attempts exceeded	Check the interface cable and the transmission parameters. Also check whether the receive function is configured correctly at the partner device.
16#8387	Runtime error: Number of transmission attempts exceeded	Check the interface cable, the transmission parameters and the configuration of the communication partner.
16#8388	Runtime error: Error at the "Block check character" The internally calculated value of the block check character does not correspond to the block check character received by the partner at the connection end.	Check whether the connection is seriously disrupted; in this case you may also occasionally see error codes. Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#8389	Runtime error: Invalid character received while waiting for free receive buffer	The send request of the communication partner (STX, 02H) is only answered with DLE when the receive buffer is empty. No additional character may be received before (except STX again). Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#838A	Runtime error: Logical error during receiving. After DLE was received, a further random character (other than DLE or ETX) was received.	Check if partner DLE in frame header and in data string is always in duplicate or the connection is released with DLE ETX. Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#838B	Runtime error: Character delay time exceeded	Partner device too slow or faulty. Verify by using an interface test device that is switched into the transmission line, if necessary.
16#838C	Runtime error: Wait time for free receive buffer has started	In the user program call the instruction more often or configure a communication with data flow control.
16#838D	Runtime error: frame repetition does not start within 4 s after NAK	Check the communication partner. A received frame that is possibly corrupted must be repeated by the partner within 4 seconds.

Error code	Description	Remedy
16#838E	Runtime error: In idle mode, one or several characters (other than NAK or STX) were received.	Check for proper function of the partner device, possibly using an interface test device that is switched into the transmission line.
16#838F	Runtime error: Initialization conflict - Both partners have set high priority	Set the "Low" priority at one of the partners
16#8391	Parameter assignment error: 3964 configuration data rejected because Freeport is set	If the Freeport protocol is set, make sure that no 3964 parameter assignment data is sent.

[1] Only with instructions for S7-300/400 CPUs

Overview of error messages - Modbus

Error code	Description	Remedy
16#0000	No error	-
Configuration error of the interface - Modbus_Comm_Load		
16#8181	The module does not support this data transmission rate.	Select a valid data transmission rate for the module at the BAUD parameter.
16#8182	The module does not support this parity setting.	Select a suitable value for "Parity" at the PARITY parameter. The following are valid: • None (1) • Even (2) • Odd (3) • Mark (4) • Space (5) • Any (6)
16#8183	The module does not support this type of data flow control.	Select a valid data flow control for the module at the FLOW_CTRL parameter.
16#8184	Invalid value for "Response timeout".	Select a suitable value for "Response timeout" at the RESP_TO parameter. Valid range of values: 1 to 65535 (ms)
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the Send_Config.RDREC.STATUS or Receive_Config.RDREC.STATUS static parameters or RDREC.STATUS and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the Send_Config.WRREC.STATUS or Receive_Config.WRREC.STATUS static parameters or WRREC.STATUS and in the description of the SFB WRREC.
16#8282	Module not available	Check the input at the PORT parameter and ensure that the module can be reached.
Configuration error - Modbus_Slave		

Error code	Description	Remedy
16#8186	Invalid slave address	Select a suitable slave address at the MB_ADDR parameter. The following are valid: 1-247 at standard address area; 1-65535 at extended address area (0 is reserved for Broadcast)
16#8187	Invalid value at MB_HOLD_REG parameter	Select a suitable value for the hold register at the MB_HOLD_REG parameter.
16#8188	Invalid operating mode or broadcast (MB_ADDR = 0) and MODE parameter ≠ 1	Select the value 1 for MODE in Broadcast mode or select a different operating mode.
16#818C	The pointer to a MB_HOLD_REG area must be a data block or a bit memory address area.	Select a suitable value for the pointer to the MB_HOLD_REG area.
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.WRREC.STATUS or Receive_P2P.WRREC.STATUS, and in the description of the SFB WRREC.
16#8452 [1)]	MB_HOLD_REG is not a pointer to a DB or a bit memory area	Check the MB_HOLD_REG pointer
16#8453 [1)]	MB_HOLD_REG is not a pointer of type BOOL or WORD	Check the MB_HOLD_REG pointer
16#8454 [1)]	The area addressed by MB_HOLD_REG is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the MB_HOLD_REG pointer
16#8455 [1)]	MB_HOLD_REG points to a write-protected DB	Check the MB_HOLD_REG pointer
16#8456 [1)]	Error during instruction execution. The cause of the error is shown in the STATUS parameter.	Determine the value of the SFCSTATUS parameter. Check what this means in the description for SFC51, STATUS parameter.
Configuration error - Modbus_Master		
16#8180	Invalid value for MB_DB parameter	The value configured for MB_DB (instance data DB) at the Modbus_Comm_Load instruction is not valid. Check the interconnection of the Modbus_Comm_Load instruction and its error messages.
16#8186	Invalid station address	Select a suitable station address at the MB_ADDR parameter. The following are valid: 1-247 at standard address area; 1-65535 at extended address area (0 is reserved for Broadcast)
16#8188	Invalid operating mode or broadcast (MB_ADDR = 0) and MODE parameter ≠ 1	Select the value 1 for MODE in Broadcast mode or select a different operating mode.

Error code	Description	Remedy
16#8189	Invalid data address	Select a suitable value for the data address at the DATA_ADDR parameter. See description Modbus_Master (Page 7536) in the Info system
16#818A	Invalid length	Select a suitable data length at the DATA_LEN parameter. See description Modbus_Master (Page 7536) in the Info system
16#818B	Invalid value for DATA_PTR	Select a suitable value for the data pointer at the DATA_PTR parameter (M or DB address). See description Modbus_Master (Page 7536) in the Info system
16#818C	Interconnection error of the DATA_PTR parameter	Check the interconnection of the instruction.
16#818D	The area addressed by DATA_PTR is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the DATA_PTR pointer
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.WRREC.STATUS or Receive_P2P.WRREC.STATUS or Receive_Reset and in the description of the SFB WRREC.
Communication errors - Modbus_Master and Modbus_Slave		
16#80 D1	The waiting time for XON or CTS = ON has expired.	The communication partner has a fault, is too slow or is offline. Check the communication partner or change the parameters, if necessary.
16#80D2	"Hardware RTS always ON": Send job canceled due to change from DSR = ON to OFF	Check the communication partner. Make sure that DSR is ON for the entire duration of transmission.
16#80E0	Frame aborted: Send buffer overflow / send frame too long	In the user program call the instruction more often or configure a communication with data flow control.
16#80E1	Frame aborted: Parity error	Check the connection line of the communication partners, or verify that both devices are configured for the same data transmission rate, parity and stop bit number.
16#80E2	Frame aborted: Character frame error	Check the settings for start bit, data bits, parity bit, data transmission rate, and stop bit(s).
16#80E3	Frame aborted: Character overflow error	Check the number of data in the frame of the communication partner.
16#80E4	Frame aborted: Maximum frame length reached	Select a shorter frame length at the communication partner. The following are valid (depending on the module): 1 to 1024/2048/4096 (bytes)

Error code	Description	Remedy
Communication error - Modbus_Master		
16#80C8	The slave does not respond within the set time	Check the data transmission rate, parity and wiring of the slave.
16#8200	The interface is busy with an ongoing request.	Repeat the command later. Make sure that there are no commands still running before you start a new one.
Protocol error - Modbus_Slave (only communication modules that support Modbus)		
16#8380	CRC error	Checksum error of the Modbus frame. Check the communication partner.
16#8381	The function code is not supported or is not supported for broadcast.	Check the communication partner and make sure that a valid function code is sent.
16#8382	Invalid length information in the request frame	Select a suitable data length at the DATA_LEN parameter.
16#8383	Invalid data address in the request frame	Select a suitable value for the data address at the DATA_ADDR parameter.
16#8384	Invalid data value error in the request frame	Check the data value in the request frame of the Modbus master
16#8385	The diagnostic value is not supported by the Modbus slave (function code 08)	The Modbus slave only supports the diagnostic values 16#0000 and 16#000A.
Protocol error - Modbus_Master (only communication modules that support Modbus)		
16#8380	CRC error	Checksum error of the Modbus frame. Check the communication partner.
16#8381	Response frame from Modbus Slave with the following error message: The function code is not supported.	Check the communication partner and make sure that a valid function code is sent.
16#8382	Response frame from Modbus Slave with the following error message: Invalid length	Select a suitable data length.
16#8383	Response frame from Modbus Slave with the following error message: Invalid data address in the request frame	Select a suitable value for the data address at the DATA_ADDR parameter.
16#8384	Response frame from Modbus Slave with the following error message: Data value error	Check the request frame to the Modbus slave.
16#8385	Response frame from Modbus Slave with the following error message: The diagnostic value is not supported by the Modbus slave	The Modbus slave only supports the diagnostic values 16#0000 and 16#000A.
16#8386	The returned function code does not match the requested function code.	Check the response frame and the addressing of the slave.
16#8387	A slave that was not requested answers	Check the response frame of the slave. Check the address settings of the slave.
16#8388	Error in the response of the slave to a write request.	Check the response frame of the slave.
16#8828 [1)]	DATA_PTR points to a bit address that is not equal to n * 8	Check the DATA_PTR pointer
16#8852 [1)]	DATA_PTR is not a pointer to a DB or a bit memory area	Check the DATA_PTR pointer
16#8853 [1)]	DATA_PTR is not a pointer of type BOOL or WORD	Check the DATA_PTR pointer
16#8855 [1)]	DATA_PTR points to a write-protected DB	Check the DATA_PTR pointer
16#8856 [1)]	Error during call of SFC51	Call the Modbus_Master instruction again
Error - Modbus_Slave (only communication modules that support Modbus)		

Error code	Description	Remedy
16#8428 [1]	MB_HOLD_REG points to a bit address that is not equal to n * 8	Check the MB_HOLD_REG pointer
16#8452 [1]	MB_HOLD_REG is not a pointer to a DB or a bit memory area	Check the MB_HOLD_REG pointer
16#8453 [1]	MB_HOLD_REG is not a pointer of type BOOL or WORD	Check the MB_HOLD_REG pointer
16#8454 [1]	The area addressed by MB_HOLD_REG is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the MB_HOLD_REG pointer
16#8455 [1]	MB_HOLD_REG points to a write-protected DB	Check the MB_HOLD_REG pointer
16#8456 [1]	Error during call of SFC51	Call the Modbus_Slave instruction again

[1] Only with instructions for S7-300/400 CPUs

Overview of error messages - USS

Error code	Description	Remedy
16#0000	No error	-
16#8180	Length error in response of drive	Check the response frame of the drive.
16#8181	Data type error	Select a suitable data type. The following are valid: • Real • Word • Double word
16#8182	Data type error: "Double word" or "Real" may not be returned for the "Word" request.	Check the response frame of the drive.
16#8183	Data type error: "Word" may not be returned for the "Double word" or "Real" request.	Check the response frame of the drive.
16#8184	Checksum error in response of drive	Check the drive and the communication connection.
16#8185	Addressing error	Valid drive address range: 1 to 16
16#8186	Setpoint error	Valid setpoint range: -200 % to +200 %
16#8187	Incorrect drive number returned	Check the response frame of the drive.
16#8188	Invalid PZD length	Permitted PZD lengths: 2, 4, 6, 8 words
16#8189	The module does not support this data transmission rate.	Select a valid data transmission rate for the module.
16#818A	A different request for this drive is currently active.	Repeat the parameter read or write command later.
16#818B	The drive does not respond.	Check the drive.
16#818C	The drive responds with an error message to a parameter request.	Check the response frame of the drive. Check the parameter request.
16#818D	The drive responds with an access error message to a parameter request.	Check the response frame of the drive. Check the parameter request.
16#818E	The drive was not initialized.	Check the user program and make sure that the USS_Drive_Control instruction is called for this drive.

Error code	Description	Remedy
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Port_Config.RDREC.STATUS, Send_Config.RDREC.STATUS , Receive_Config.RDREC.STATUS , Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Port_Config.WRREC.STATUS, Send_Config.WRREC.STATUS , Receive_Config.WRREC.STATUS , Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB WRREC.

[1] Only with instructions for S7-300/400 CPUs

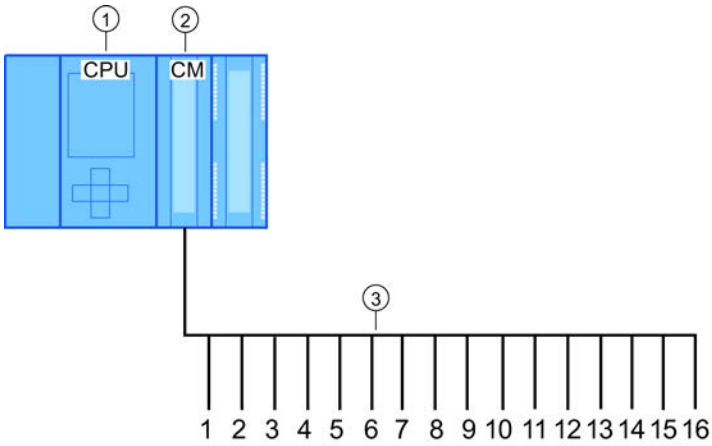
USS

Overview of USS communication

USS communication

The USS instructions control the operation of drives which support the protocol of the universal serial interface (USS). You can communicate with several drives by means of RS485 connections and the USS instructions with the PtP communication modules. Each RS485 port can operate up to 16 drives.

The USS protocol uses a master-slave network for communication via a serial bus. The master uses an address parameter to send data to a selected slave. A slave cannot send without having first received a send request. Communication between individual slaves is not possible. USS communication takes place in half-duplex mode. The figure below shows a network diagram for a sample application with 16 drives.



① CPU
② CM
③ USS drives in a USS network

Figure 9-14 Wiring example with S7-1500 communication module

Note

Communicating with a drive via RS232

You can basically also use CM PtP RS232 BA and CM PtP RS232 HF for communication with a drive. However, only **one** drive can be connected to an RS232 port.

Communicating with a drive via RS422

You can basically also use the RS422 interface of the CM PtP RS422/485 BA and CM PtP RS422/485 HF for communication with a drive. However, only **one** drive can be connected to an RS422 port.

USS instructions in your program

- USS_Port_Scan: The instruction USS_Port_Scan allows you to communicate via a communication module with up to 16 drives using a USS network (must be called cyclically). There is only one USS_Port_Scan instruction per PtP communication port in the program, and it controls transmission to all drives.
- USS_Drive_Control: The instruction USS_Drive_Control allows you to prepare the send data from USS_Port_Scan for a drive and display its receive data. USS_Drive_Control configures the data to be sent and evaluates the data received in a previous request from USS_Port_Scan.
- USS_Read_Param: The instruction USS_Read_Param allows you to read parameters from a drive.
- USS_Write_Param: The instruction USS_Write_Param allows you to change parameters in a drive.

Requirements for using the USS protocol

The four USS instructions use 2 FBs and 2 FCs to support the USS protocol. For each USS network, one instance data block (DB) is used for USS_Port_Scan and one instance data block for all calls of USS_Drive_Conrol.

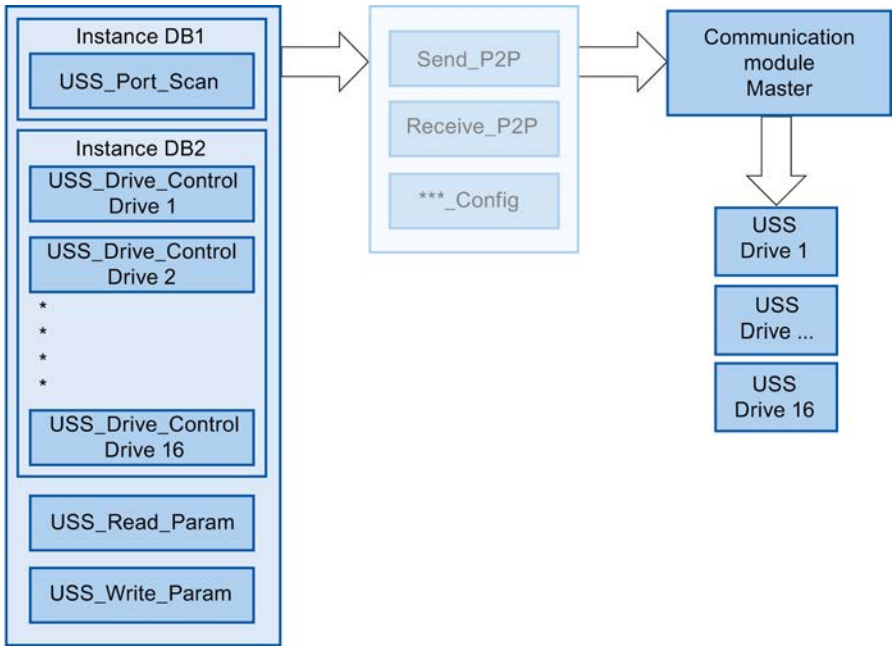


Figure 9-15 USS program sequence

All drives (max. 16) that are connected to one RS485 port are part of the same USS network. All drives that are connected to another RS485 port are part of a different USS network. Each USS network is managed using a unique instance data block for all USS_Drive_Control instructions and one further instance data block for the USS_Port_Scan instruction. All instructions that are part of a USS network must share this instance data block for USS_Drive_Conrol. The USS_Port_Scan, USS_Read_Param and USS_Write_Param instructions have the USS_DB parameter for this function. This parameter must be connected to the (static) USS_DB parameter of the instance DB of the USS_Drive_Control instruction.

- The instructions USS_Drive_Control and USS_Port_Scan are function blocks (FB). If you add the USS_Drive_Control or USS_Port_Scan instruction to the programming editor, you are prompted to assign a DB for this FB in the "Call options" dialog. If it is the first USS_Drive_Control instruction in this program for this USS network, you can apply the DB standard assignment (or change the name, if necessary) and the new DB is created for you. If, however, it is not the first USS_Drive_Control instruction for this drive, you need to select the DB which has already been assigned to this USS network from the drop-down list in the "Call options" dialog.
- The USS_Port_Scan and USS_Read_Param instructions are functions (FCs). No DB is assigned when you add these FCs in the editor. If you add these FCs or the USS_Port_Scan instruction in the editor, you need to assign the USS_DB parameter of the corresponding USS_Drive_Control instance DB to the USS_DB input of these instructions. Double-click the parameter field and click the symbol to display the available DBs. Enter a period "." and select the USS_DB parameter from the drop-down list.

- The USS_Port_Scan function controls communication between the CPU and the drives via the point-to-point RS485 communication port. A communication with the drive is processed every time you call this function. Your program must call this function fast enough so that the drives do not signal a timeout. To ensure a constant time response for frame communication, you should call this instruction in a cyclic interrupt OB.
- The USS_Drive_Control instruction gives your program access to a specified drive in the USS network. Its inputs and outputs correspond to the states and operating functions of the drive. If there are 16 drives in your network, you must call USS_Drive_Control at least 16 times in your program, i.e. once for each drive.
  You should only call the USS_Drive_Control instruction from a cyclic OB.
- The operating parameters of the drive are read and written with the USS_Read_Param and USS_Write_Param functions. These parameters control the internal operation of the drive. See the drive manual for a definition of these parameters. Your program may include any number of these functions, but only one read or write request may be active for a drive at any particular time. You may only call the USS_Read_Param and USS_Write_Param functions from the cycle OB of a main program.

NOTICE
USS instruction calls Only ever call USS_Drive_Control, USS_Read_Param or USS_Write_Param from a cycle OB of the main program. The USS_Port_Scan instruction function can be called from any OB, but it is usually called from a cyclic interrupt OB. Do not use the USS_Drive_Control, USS_Read_Param or USS_Write_Param instruction in an OB with a higher priority than the corresponding USS_Port_Scan instruction. For example, do not add USS_Port_Scan to the main program or USS_Read_Param to a cyclic interrupt OB. You may encounter unexpected errors if the execution of USS_Port_Scan is interrupted by another instruction.

Calculating time for communication with the drive

Communication with the drive takes place asynchronously to the cycle of the CPU. The CPU usually runs through several cycles before communication with a drive is complete.

To ensure that the watchdog set for the drive is not triggered, the send frames must be sent to the drive within the watchdog time. You must allow for the number of any retries which may be needed to complete the transaction if communication errors occur. By default, up to 2 retries are made for each transaction with the USS protocol.
The time interval between two send frames is calculated as follows:

(N * 5 * cycle time + 2 * frame runtime) * number of retries + (received frame timeout) * (number of retries - 1)

N	Number of drives in this network
Factor 5	5 cycles are needed for sending and receiving frames.
Cycle time	Max. cycle time of the cyclic interrupt OB in which the USS_Port_Scan instruction is called.
2 * frame time	Once each for sending and receiving

Frame runtime	Frame time = (number of characters per frame) * (11 Bit per character) / (data transmission rate in Bit/s)
Number of retries	3

The following times apply for "Received frame timeout" (ms)

Bit/s	115200	57600	38400	19200	9600	4800	2400	1200
Receive_Conditions.END.RCVTIME	25	29	33	56	72	100	100	100
Receive_Conditions.END.MSGTIME	25	29	33	56	72	124	240	460

Example:
5 drives
data transmission rate = 9600 Bit/s
28 characters per frame
cycle time = 0.020 sec
time interval = (5 * 5 * 0.02 sec + 2 * 28 * 11 / 1200 sec) * 3 = 1.69 sec
The time monitoring of the drive must be set to approx. 2 seconds in this case.

See also

USS_Port_Scan: Communication by means of a USS network (Page 7507)

USS_Port_Scan: Communication by means of a USS network

Description

The USS_Port_Scan instruction processes the communication by means of a USS network. STEP 7 automatically creates the instance DB when you add the instruction.

Parameters

Parameter	Declaration	Data type		Standard	Description
		S7-1500	S7-300/400/ WinAC		
PORT	IN	Port	Word	0	The CM port value can be found in the "Hardware ID" property of the device configuration (S7-1500) (input address; S7-300/400) once you have configured the CM. The symbolic port name is assigned in the "System constants" tab of the PLC tag table. In S7-300/400/WinAC systems, the input address assigned in HWCN must be assigned to the CM port.
BAUD	IN	DInt		9600	Data transmission rate for USS communication The following are valid: • 1200 Bit/s • 2400 Bit/s • 4800 Bit/s • 9600 Bit/s • 19200 Bit/s • 38400 Bit/s • 57600 Bit/s • 115200 Bit/s
USS_DB	INOUT	USS_BASE		–	The USS_DB parameter must be connected to the (static) USS_DB parameter of the instance DB which is generated and initialized when you add a USS_Drive_Control instruction to your program.
COM_RST	INOUT	---	Bool	FALSE	Initialization of the USS_Port_Scan instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
ERROR	OUT	Bool		FALSE	If TRUE, this output indicates that an error has occurred and that the STATUS output is valid.
STATUS	OUT	Word		0	Error code (see Error messages (Page 7518)).

There is only one USS_Port_Scan instruction per PtP communication port in the program and each call of this instruction controls a transmission to or from all drives in this network. All USS functions assigned to one USS network and one PtP communication port must use the same instance DB.

Your program must execute the USS_Port_Scan instruction often enough to prevent a timeout in the drive (see Auto-Hotspot "Calculating time for communication with the drive").

The USS_Port_Scan instruction is usually called from a cyclic interrupt OB to prevent drive timeouts and to have the last USS data updates available for calls of USS_Drive_Control .

USS_Port_Scan data block tags

The table below shows the public static tags in the instance DB of USS_Port_Scan that you can use in your program.

Table 9-298 Static tags in the instance DB

Tag	Data type	Standard	Description
MODE	USInt	4	Operating mode Valid operating modes are: • 0 = Full duplex (RS232) • 1 = Full duplex (RS422) four-wire operation (point-to-point) • 2 = Full duplex (RS 422) four-wire mode (multipoint master; CM PtP (ET 200SP)) • 3 = Full duplex (RS 422) four-wire mode (multipoint slave; CM PtP (ET 200SP)) • 4 = Half duplex (RS485) two-wire mode
LINE_PRE	USInt	2	Receive line initial state Valid initial states are: • 0 = "no" initial state • 1 = signal R(A)=5 V, signal R(B)=0 V (break detection): Break detection is possible with this initial state. Can only be selected with: "Full duplex (RS422) four-wire operation (point-to-point connection)" and "Full duplex (RS422) four-wire operation (multipoint slave)". • 2 = signal R(A)=0 V, signal R(B)=5 V: This default setting corresponds to the idle state (no active send operation). No break detection is possible with this initial state.
BRK_DET	USInt	0	Activate diagnostics interrupt: • 0 - not activated • 1 - activated
RETRIES_MAX	SInt/Byte	2	Number of retries when communication errors occur. You can use this parameter to set the number of send retries for a request frame if the response frame is not received within the set time.
EN_DIAG_ALARM	Bool	0	Activate diagnostics interrupt: • 0 - not activated • 1 - activated
EN_SUPPLY_VOLT	Bool	0	Enable diagnostics for missing supply voltage L+ • 0 - not activated • 1 - activated

USS_Drive_Control: Preparing and displaying data for the drive

Description

The instruction USS_Drive_Control prepares send data for the drive and evaluates the drive response data. You need to use a separate instance of the instruction for each drive, and all USS functions assigned to one USS network and one PtP communication port must use the

same instance DB. You must enter the DB name when you add the first USS_Drive_Control instruction. You then refer to this DB that was created when the first instruction was added.

STEP 7 automatically creates the DB when you add the instruction.

Parameters

Parameters	Declaration	Data type		Standard	Description
		S7-1500	S7-300/400/ WinAC		
RUN	IN	Bool		FALSE	Start bit of the drive: If this parameter is TRUE, this input allows you to operate the drive with the preset speed. If RUN changes to FALSE during operation of the drive, the motor coasts to a standstill. This behavior differs from disconnection of the power supply (OFF2) and braking of the motor (OFF3).
OFF2	IN	Bool		FALSE	"Coast to standstill" bit: If this parameter is FALSE, this bit causes the drive to coast to a standstill without braking.
OFF3	IN	Bool		FALSE	Fast stop bit: If this parameter is FALSE, this bit causes a fast stop by braking the drive.
F_ACK	IN	Bool		FALSE	Error acknowledgment bit: This bit resets the error bit of a drive. The bit is set after clearing the error and the drive detects this way that the previous error no longer has to be reported.
DIR	IN	Bool		FALSE	Direction control of the drive: This bit is set if the drive is to run forward (if SPEED_SP is positive; see table "Interaction of SPEED_SP and DIR parameters").
DRIVE	IN	USInt	Byte	1	Address of the drive: This input is the address of the USS drive. The valid range is between drive 1 and drive 16.
PZD_LEN	IN	USInt	Byte	2	Word length: This is the number of PZD data words. Valid values are 2, 4, 6 or 8 words.
SPEED_SP	IN	Real		0.0	Speed setpoint: This is the speed of the drive as a percentage of the configured frequency. A positive value means that the drive runs forward (if DIR is true). The valid range is 200.00 to -200.00.
CTRL3	IN	Word		0	Control word 3: Value that is written to a user-defined parameter of the drive. You have to configure it in the drive (optional parameter).
CTRL4	IN	Word		0	Control word 4: Value that is written to a user-defined parameter of the drive. You have to configure it in the drive (optional parameter).
CTRL5	IN	Word		0	Control word 5: Value that is written to a user-defined parameter of the drive. You have to configure it in the drive (optional parameter).
CTRL6	IN	Word		0	Control word 6: Value that is written to a user-defined parameter of the drive. You have to configure it in the drive (optional parameter).
CTRL7	IN	Word		0	Control word 7: Value that is written to a user-defined parameter of the drive. You have to configure it in the drive (optional parameter).

Parameters	Declaration	Data type		Standard	Description
		S7-1500	S7-300/400/ WinAC		
CTRL8	IN	Word		0	Control word 8: Value that is written to a user-defined parameter of the drive. You have to configure it in the drive (optional parameter).
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the USS_Drive_Control instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
NDR	OUT	Bool		FALSE	New data available: If this parameter is TRUE, the bit signals that data of a new communication request is available at the output.
ERROR	OUT	Bool		FALSE	Error occurred: If TRUE, this indicates that an error has occurred and that the STATUS output is valid. All other outputs are set to zero in the event of an error. Communication errors are only signaled at the ERROR and STATUS outputs of the USS_Port_Scan instruction.
STATUS	OUT	Word		0	Error code (see Error messages (Page 7518)).
RUN_EN	OUT	Bool		FALSE	Operation enabled: This bit signals if the drive is running.
D_DIR	OUT	Bool		FALSE	Drive direction: This bit signals if the drive is running forward. • FALSE – forward • TRUE – backward
INHIBIT	OUT	Bool		FALSE	Drive blocked: This bit signals the status of the block bit for the drive. • FALSE – not blocked • TRUE – blocked
FAULT	OUT	Bool		FALSE	Drive error: This bit signals that an error occurred in the drive. You must remedy the fault and set the F_ACK bit to clear this bit.
SPEED	OUT	Real		0.0	Actual value drive speed (scaled value of STATUS 2 of the drive): This is the speed of the drive as a percentage of the configured speed.
STATUS1	OUT	Word		0	STATUS 1 of the drive This value includes fixed status bits of a drive.
STATUS3	OUT	Word		0	STATUS 3 of the drive This value includes a user-definable status word of the drive.
STATUS4	OUT	Word		0	STATUS 4 of the drive This value includes a user-definable status word of the drive.
STATUS5	OUT	Word		0	STATUS 5 of the drive This value includes a user-definable status word of the drive.
STATUS6	OUT	Word		0	STATUS 6 of the drive This value includes a user-definable status word of the drive.
STATUS7	OUT	Word		0	STATUS 7 of the drive This value includes a user-definable status word of the drive.
STATUS8	OUT	Word		0	STATUS 8 of the drive This value includes a user-definable status word of the drive.

When the initial execution of USS_Drive_Control takes place, the drive specified by the USS address (DRIVE parameter) is initialized in the instance DB. After initialization, subsequent USS_Port_Scan instructions can start communication with the drive from this drive number.

If you change the drive number, you must first place the CPU in STOP and then back in RUN to initialize the instance DB. The input parameters are configured in the USS send buffer and any outputs are read from a "previous" valid response buffer. USS_Drive_Control only configures the data to be sent and evaluates data received in a previous request.

You can control the drive's direction of rotation by using the D_IR input (Bool) or the sign (positive or negative) at the SPEED_SP input (Real). The table below explains how these inputs work together to determine the direction of rotation of the drive, provided the motor rotates forward.

Table 9-299 Interaction of SPEED_SP and DIR parameters

SPEED_SP	DIR	Direction of rotation of the drive
Value > 0	0	Backward
Value > 0	1	Forward
Value < 0	0	Forward
Value < 0	1	Backward

USS_Read_Param: Read data from drive

Description

The USS_Read_Param instruction reads a parameter from a drive. All USS functions assigned to one USS network and one PtP communication port must use the instance data block of the USS_Drive_Control instruction. USS_Read_Param must be called from a cycle OB of the main program.

Parameters

Parameters	Declaration	Data type		Standard	Description
		S7-1500	S7-300/400/ WinAC		
REQ	IN	Bool		–	A positive edge at REQ creates a new read request.
DRIVE	IN	USInt	Byte	–	Address of the drive: DRIVE is the address of the USS drive. The valid range is between drive 1 and drive 16.
PARAM	IN	UInt		–	Parameter number: PARAM specifies the drive parameter to write. The range for this parameter is located between 0 and 2047; for some drives the most significant byte of the INDEX parameter can access parameter values greater than 2047. Additional information on access to an extended range is available in your drive manual.

Parameters	Declaration	Data type		Standard	Description
		S7-1500	S7-300/400/ WinAC		
INDEX	IN	UInt		–	Parameter index: INDEX specifies the drive parameter index to which to write. It is a 16-bit value in which the least significant bit is the actual index value with a range of (0 to 255). The drive can also use the most significant byte, which is drive-specific. Additional information is available in your drive manual.
USS_DB	INOUT	USS_BASE		–	The USS_DB parameter must be connected to the (static) USS_DB parameter of the instance DB which is generated and initialized when you add a USS_Drive_Control instruction to your program.
DONE[1]	OUT	Bool		FALSE	If this parameter is TRUE, the previously requested value of the read parameter is available at the VALUE output. This bit is set when the USS_Drive_Control instruction recognizes the read response of the drive. This bit is reset the next time USS_Read_Param is called.
ERROR	OUT	Bool		FALSE	ERROR = TRUE: An error has occurred and the STATUS output is valid. All other outputs are set to zero in the event of an error. Communication errors are only signaled at the ERROR and STATUS outputs of the USS_Port_Scan instruction.
STATUS	OUT	Word		0	Error code (see Error messages (Page 7518)).
VALUE	OUT	Variant (Word, Int, UInt, DWord, DInt, UDInt, Real)	Any (Word, Int, UInt, DWord, DInt, UDInt, Real)	–	This is the value of the parameter that was read and is only valid if the DONE bit is true.

[1] The DONE bit indicates that valid data was read out of the referenced motor drive and transmitted to the CPU. It does not indicate that the instruction is cabable of immediately read an additional parameter. An empty read request must be sent to the motor drive and acknowledged by the instruction before the parameter channel is freed up for use by the respective drive. The immediate call of USS_Read_Param or USS_Write_Param for the specific motor drive results in error 0x818A.

USS_Write_Param: Change data in drive

Note

For EEPROM write instructions (EEPROM in a USS drive):

Keep the number of EEPROM write operations to a minimum in order to maximize the EEPROM service life.

Description

The USS_Write_Param instruction changes a parameter in the drive. All USS functions assigned to one USS network and one PtP communication port must use the instance data block of the USS_Drive_Control.

USS_Write_Param must be called from the cycle OB of a main program.

Parameters

Table 9-300 Data types for the parameters

Parameters	Declaration	Data type		Standard	Description
		S7-1500	S7-300/400/ WinAC		
REQ	IN	Bool		–	A positive edge at REQ creates a new write request.
DRIVE	IN	USInt	Byte	–	Address of the drive: DRIVE is the address of the USS drive. The valid range is between drive 1 and drive 16.
PARAM	IN	UInt		–	Parameter number: PARAM specifies the drive parameter to write. The range for this parameter is located between 0 and 2047; for some drives the most significant byte of the INDEX parameter can access parameter values greater than 2047. Additional information on access to an extended range is available in your drive manual.
INDEX	IN	UInt		–	Parameter index: INDEX specifies the drive parameter index to which to write. It is a 16-bit value in which the least significant bit is the actual index value with a range of (0 to 255). The drive can also use the most significant byte, which is drive-specific. Additional information is available in your drive manual.
EEPROM	IN	Bool		–	Save in EEPROM of the drive: If TRUE, the transaction of a parameter for writing to the drive is saved in the EEPROM of the drive. If FALSE, the written value is saved only temporarily and is lost the next time you switch on the drive.
VALUE	IN	Variant (Word, Int, UInt, DWord, DInt, UDInt, Real)	Any (Word, Int, UInt, DWord, DInt, UDInt, Real)	–	Value of the parameter in which you want to write. It must be valid with a positive edge of REQ.
USS_DB	INOUT	USS_BASE		–	The USS_DB parameter must be connected to the (static) USS_DB parameter of the instance DB which is generated and initialized when you add a USS_Drive_Control instruction to your program.
DONE[1]	OUT	Bool		FALSE	If TRUE, the VALUE input is written to the drive. This bit is set when the USS_Drive_Control instruction recognizes the write response of the drive. This bit is reset the next time USS_Write_Paramis called.

Parameters	Declaration	Data type		Standard	Description
		S7-1500	S7-300/400/ WinAC		
ERROR	OUT	Bool		FALSE	If TRUE, an error has occurred and the STATUS output is valid. All other outputs are set to zero in the event of an error. Communication errors are only signaled at the ERROR and STATUS outputs of the USS_Port_Scan instruction.
STATUS	OUT	Word		0	Error code (see Error messages (Page 7518)).

[1] The DONE bit indicates that valid data was read out of the referenced motor drive and transmitted to the CPU. It does not indicate that the USS library is able to read out an additional parameter immediately. An empty write request must be sent to the motor drive and acknowledged by the instruction before the parameter channel is freed up for use by the respective drive. The immediate call of USS_Read_Param or USS_Write_Param FC for the specific motor drive results in error 0x818A.

General information on drive setup

Requirements for the drive setup

- You need to configure the use of 4 PIV words (ParameterIDValue) for the drives.
- The drives can be configured for 2, 4, 6 or 8 PZD words (Process data area).
- The number of PZD words in the drive must correspond to the PZD_LEN input of the USS_Drive_Control instruction of the drive.
- Make sure that the data transmission rate of all drives corresponds to the BAUD input of the USS_Port_Scan instruction.
- Make sure that the drive is set up for USS communication.
- Make sure that it is specified in the drive that the frequency setpoint is provided by the USS interface.
- Make sure that the drive address is specified (areas: 1-16).
  This address must correspond to the DRIVE input at the USS_Drive_Control block of the drive.
- Make sure that the RS485 network is terminated correctly.

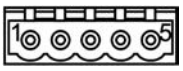
Connection and setup of SINAMICS V20 drive

An application example for operation of a SINAMICS V20 at an S7-1200 is available on the Internet. (http://support.automation.siemens.com/WW/view/en/63696870)

Connecting the SINAMICS V20 drive

Connection example of a SIEMENS G120(C) drive to a USS network. Connection examples for other drives are available in the manual of the respective drive.

The connection of a SINAMICS G120(C) drive to the USS network takes place via plug-in connection. The connection is short-circuit proof and isolated.



1 0 V reference potential
2 RS485N, receiving and sending (-)
3 RS485N, receiving and sending (+)
4 Cable shield
5 Not used

Figure 9-16 USS connection

NOTICE
Different reference voltages If you connect devices that do not have the same reference voltage, you may create unwanted currents in the connection cable. These unwanted currents may cause communication errors or damages in the devices. Make sure that all devices you connect with a communication cable either have the same reference conductor in the circuit or are electrically disconnected to prevent the occurrence of unwanted currents. Make sure that the shield is connected to the ground or pin 1 of the bus connector on the drive. Make sure that wiring terminal 2 (GND) of the G120(C) is connected to the ground.

If the RS485 master (e.g., S7-1200 CPU with CM1241 communication module) is connected by means of a PROFIBUS connector, wire the bus cables as follows:

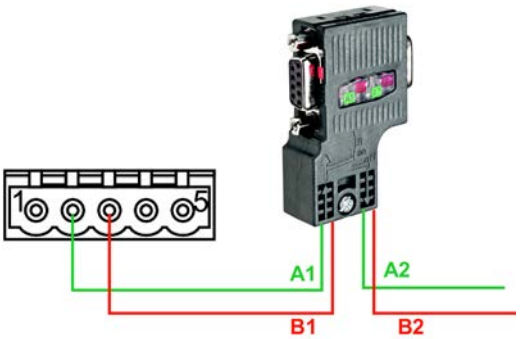


Figure 9-17 Connection of communication module

If the RS485 master is a terminating station in the network or a point-to-point connection, you must use terminals A1 and B1 (not A2 and B2) of the PROFIBUS connector, as these terminals provide termination settings (for example, with the DP plug connector 6ES7972-0BB52-0XA0).

If the G120(C) is configured as terminating station in the network, you must set the switch for the bus terminating resistor to "ON".

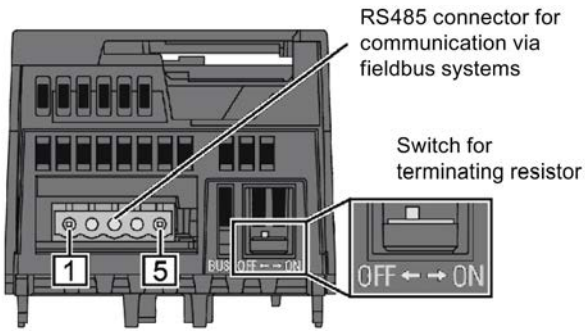


Figure 9-18 Connection of terminating stations

Setup of a G120(C) drive

Before you connect a drive to the S7-1500 or ET 200SP, make sure that the drive has the following system parameters.

Step	Instruction	Operating Instructions	
		G120 [1]	G120C [2]
1	Perform a basic commissioning of the drive with the Operator Panel BOP-2. The converter offers different defaults (macros) for its inputs and outputs and the fieldbus interface. In the ninth step of the basic commissioning (MAC PAR p15), select macro 21 for USS communication. This determines the default for the following parameters: • Data transmission rate (p2020): 38400 bps • Number of PZD (p2022): 2 • Number of PIV (p2023): variable Note: You can also conduct basic commissioning with the STARTER commissioning software or with SINAMICS Startdrive.	Chapter 4.4.3	Chapter 6.4.1
2	Specify the USS address of the converter with the address switch on the control unit of the G120 or on the G120(C). • Valid address range: 1 … 30 Note: Yon can also set the USS address with the parameter p2021 or with STARTER or SINAMICS Startdrive.	Chapter 6.2.2.1	Chapter 8.4.2.1
	With the following steps you are accessing the parameters directly with the BOP-2 by entering their numbers and modifying their values.	Chapter 4.4.2	Chapter 6.4.2

Step	Instruction	Operating Instructions	
		G120 [1]	G120C [2]
3	Adapt the following communication-related converter parameters of your application: • Data transmission rate (p2020), of ≠ 38400 bps (Make sure that setting is identical with the BAUD parameter of the USS_Port_Scan communication instruction.) • Number of PZD (p2022), if ≠ 2 (Make sure that the setting is identical with the PZD_LEN parameter of the USS_Drive_Control communication instruction.) • Number of PIV (p2023) = 4 (Change the value set to "variable" (127) with macro 21 by default to "4" (required by instructions USS_Read_Param and USS_Write_Param.) • Fieldbus SS monitoring time [ms] (p2040)	Chapter 6.2.2.2	Chapter 8.4.2.1
4	Specify the source for the speed setpoint. • n_set Eval (p1000[0]) = 6 (The speed setpoint is provided by the USS bus.)		
5	Set the reference value for speed and frequency. • n_reference f_reference (p2000) = (6,00 min^{-1} to 210000,00 min^{-1}) (all relative speeds or frequencies refer to this reference value. The reference value corresponds to 100% or 4000_{hex} (word) or $4000\ 0000_{hex}$ (double word). The following applies: $f_{reference_value}$ (in Hz) = $n_{reference_value}$ (in ((min^{-1}) / 60) x number of pole pairs))		
6	Transfer the parameters to the non-volatile memory. • Save par (p0971) = 1		

1) G120 (http://support.automation.siemens.com/WW/view/en/62089662)

2) G120(C) (http://support.automation.siemens.com/WW/view/en/61462568)

Error messages

Overview of error messages - PtP

The error messages are provided at the STATUS output of an instruction and can be evaluated there or processed in the user program.

Error code	Description	Remedy
16#0000	No error	-
RECEIVE status and error codes		
16#0094	frame end identified based on the "Receipt of fixed/ maximum frame length"	-
16#0095	frame end identified based on "Message timeout"	-
16#0096	frame end identified based on expiration of the "Character delay time"	-
16#0097	The frame was aborted because the maximum response time was reached.	-

Error code	Description	Remedy
16#0098	frame end identified based on the fulfillment of the "Read message length from message" conditions	-
16#0099	frame end identified based on the receipt of the "End sequence"	-
SEND status and error codes		
16#7000	Block idle	-
16#7001	Initial call for a new frame: Data transmission initiated	-
16#7002	Interim call: Data transmission running	-
16#8085	Invalid length	Select a suitable frame length. The following are valid (depending on the module): 1-1024/2048/4096 (Byte)
16#8088	The specified length exceeds the range set in the receive buffer. Note: If the data type STRING has been specified at the BUFFER parameter, this error code also appears if the current string is shorter than the length specified at the LENGTH parameter.	Change the range in the receive buffer or select a frame length which corresponds to the range set in the receive buffer. The following are valid (depending on the module): 1-1024/2048/4096 (Byte)
16#8090	Configuration error: Odd number of bytes for WString	Select an even number of bytes.
RECEIVE status and error codes		
16#8088	The number of characters received exceeds the number specified at the BUFFER parameter.	Select a suitable frame length. The following are valid (depending on the module): 1-1024/2048/4096 (Byte)
16#8090	Configuration error: Odd number of bytes for WString	Select an even number of bytes.
Error message codes of the special functions		
16#818F	Incorrect parameter number setting (with USS only)	Select a suitable parameter number (PARAM). The following numbers are valid: 0-2047
16#8190	Incorrect setting of the CRC calculation	Select a suitable value for the CRC calculation. The following are valid: deactivated or activated. Check whether the module addressed supports CRC calculation.
16#8191	Incorrect setting of the diagnostic error interrupt	Select a suitable value for "Diagnostic error interrupt". The following are valid: Diagnostic error interrupt deactivated or diagnostic error interrupt activated. Check whether the module addressed supports the generation of diagnostic interrupts.
Error message codes of the "Port configuration"		
16#81A0	The module does not support this protocol.	Select a valid protocol for the module (PROTOCOL).
16#81A1	The module does not support this data transmission rate.	Select a valid data transmission rate for the module (BAUD).

Error code	Description	Remedy
16#81A2	The module does not support this parity setting.	Select a suitable value for "Parity" (PARITY). The following are valid: • None (1) • Even (2) • Odd (3) • Mark (4) • Space (5) • Any (6)
16#81A3	The module does not support this number of data bits.	Select a suitable value for "Number of data bits" (DATABITS). The following are valid: • 7 (2) • 8 (1)
16#81A4	The module does not support this number of stop bits.	Select a suitable value for "Number of stop bits" (STOPBITS). The following are valid: • 1 (1) • 2 (2)
16#81A5	The module does not support this type of data flow control.	Select a valid data flow control for the module (FLOWCTRL).
16#81A7	Invalid value for XON or XOFF	Select suitable values for XON (XONCHAR) and XOFF (XOFFCHAR). Valid range of values: 0...255
16#81AA	Invalid operating mode	Valid operating modes are: • Full duplex (RS232) (0) • Full duplex (RS422) four-wire mode (point-to-point) (1) / (CM PtP (ET 200SP)) • Full duplex (RS422) four-wire mode (multipoint master) (2)/ (CM PtP (ET 200SP)) • Full duplex (RS 422) four-wire mode (multipoint slave) (3) • Half duplex (RS485) two-wire mode. (4)
16#81AB	Invalid receive line initial state	Valid initial states are: • "No" default setting (0) • Signal R(A)=5 V, signal R(B)=0 V (break detection) (1): Can only be selected with: "Full duplex (RS422) four-wire operation (point-to-point connection)" and "Full duplex (RS422) four-wire operation (multipoint slave)". • Signal R(A)=0 V, signal R(B)=5 V (2): This default setting corresponds to the idle state (no active send operation).
16#81AC	Invalid value for "Break detection"	Select a suitable value for "Break detection". The following are valid: • Break detection deactivated (0) • Break detection activated (1).

Error code	Description	Remedy
16#81AF	The module does not support this protocol.	Select a valid protocol for the module.
Error codes of the "Send configuration"		
16#81B5	More than two end delimiters or end sequence > 5 characters	Select suitable values for "End delimiter" and "End sequence". The following are valid: • deactivated (0), • 1 (1) or 2 (2) end delimiters or • deactivated (0), • 1 (1) up to 5 (5) characters for the end sequence.
16#81B6	Send configuration rejected because the 3964(R) protocol was selected	Make sure that no send configuration is transmitted if the 3964(R) protocol is set.
Error codes of the "Receive configuration"		
16#81C0	Invalid start condition	Select a suitable start condition. The following are valid: • Send break before frame start • Send Idle Line.
16#81C1	Invalid end condition or no end condition selected	Select a suitable end condition (see Receiving data with Freeport (Page 991)).
16#81C3	Invalid value for "Maximum message length"	Select a suitable value for "Maximum message length" (MAXLEN). Valid range of values (depending on the module): 1-1024/2048/4096 (Byte)
16#81C4	Invalid value for "Offset of the length specification in the message"	Select a suitable value for "Offset of the length specification in the message". Valid range of values (depending on the module): 1-1024/2048/4096 (Byte)
16#81C5	Invalid value for "Size of length field"	Select a suitable value for "Size of length field" (LENGTHSIZE). Valid range of values in bytes: • 1 (1) • 2 (2) • 4 (4)
16#81C6	Invalid value for "Number of characters not counted in length specification"	Select a suitable value for "Number of characters not counted in length specification" (LENGTHM). Valid range of values: 0 to 255 (bytes)
16#81C7	The total of "Offset in the message + size of length field + number of characters not counted" is greater than the maximum frame length	Select a suitable value for "Offset in message", "Size of length field" and "Number of characters not counted". Valid range of values: • Offset in the message (depending on the module): 0-1024/2048/4096 (Byte) • Size of length field: 1, 2, or 4 (bytes) • Number of characters not counted: 0 to 255 (bytes)

Error code	Description	Remedy
16#81C8	Invalid value for "Response timeout"	Select a suitable value for "Response timeout". Valid range of values: 1-65535 (ms)
16#81C9	Invalid value for "Character delay time"	Select a suitable value for "Character delay time". Valid range of values: 1 to 65535 (bit times)
16#81CB	frame end sequence is activated, but no character is activated for the check	Activate one or several characters for the check.
16#81CC	frame start sequence is activated, but no character is activated for the check	Activate one or several characters for the check.
16#81CD	Invalid value for "Prevent overwriting"	Select a suitable value for "Prevent overwriting". The following are valid: • Prevent overwriting is deactivated (0) or • Prevent overwriting is activated (1)
16#81CE	Invalid value for "Clear receive buffer on startup"	Select a suitable value for "Clear receive buffer on startup". The following are valid: • Clear receive buffer at startup is deactivated (0) • Clear receive buffer at startup is activated (1)
SEND status and error codes		
16#81D0	Receiving send request during runtime of a send command	Make sure that you do not receive an additional send request during runtime of a send command.
16#81D1	The waiting time for XON or CTS = ON has expired.	The communication partner has a fault, is too slow or is offline. Check the communication partner or change the parameters, if necessary.
16#81D2	"Hardware RTS always ON": Send job canceled due to change from DSR = ON to OFF	Check the communication partner. Make sure that DSR is ON for the entire duration of transmission.
16#81D3	Send buffer overflow / send frame too long	Select a shorter frame length. The following are valid (depending on the module): 1 to 1024/2048/4096 (bytes)
16#81D5	Transmission canceled due to parameter changes, detected wire break, or CPU in STOP	Check the parameter assignment, wire break, and CPU status.
16#81D6	Transmission canceled because end identifier was not received	Check the parameter assignment of the end characters and the frame of the communication partner.
16#81D7	Communication error between the user program and module	Check the communication (e.g., matching the sequence number).
16#81D8	Transmission attempt rejected because module is not configured	Configure the module.
Error codes of the receive configuration		
16#81E0	Frame aborted: Send buffer overflow / send frame too long	Increase the call rate for the receive function in the user program or configure communication with data flow control.
16#81E1	Frame aborted: Parity error	Check the connection line of the communication partners, or verify that both devices are configured for the same data transmission rate, parity and stop bit number.
16#81E2	Frame aborted: Character frame error	Check the settings for start bit, data bits, parity bit, data transmission rate, and stop bit(s).

Error code	Description	Remedy
16#81E3	Frame aborted: Character overflow error	Firmware error: Please contact Customer Support.
16#81E4	Frame aborted: The total length of "Offset in the message + size of the length field + number of characters not counted" is greater than the receive buffer	Select a suitable value for offset in message, size of the length field, and number of characters not counted.
16#81E5	Frame aborted: Break	Receive line to partner is interrupted. Reconnect or switch on partner.
16#81E6	Maximum number of "Buffered received frames" exceeded	In the user program call the instruction more often or configure a communication with data flow control or increase the number of buffered frames.
16#81E7	Synchronization error module and Receive_P2P	Make sure that different instances of the Receive_P2P do not access the same module.
16#81E8	Frame aborted: The character delay time has expired before the message end criterion was detected	Partner device faulty or too slow. Check this, if required, using an interface tester that is interconnected in the transmission line.
16#81E9	Modbus CRC error (only communication modules which support Modbus)	Checksum error of the Modbus frame. Check the communication partner.
16#81EA	Modbus frame too short (only communication modules which support Modbus)	Minimum length of Modbus frame not met. Check the communication partner.
16#81EB	Frame aborted: Maximum frame length reached	Select a shorter frame length at the communication partner. The following are valid (depending on the module): 1-1024/2048/4096 (Byte) Check the parameters for end of frame detection.
Error codes V24 accompanying signals		
16#81F0	The module does not support V24 accompanying signals	You have tried to set accompanying signals for a module that does not support V24 accompanying signals. Make sure that this is an RS232 module or that RS232 mode (ET 200SP) is set.
16#81F1	No operation of the V24 accompanying signals	The V24 accompanying signals cannot be operated manually if hardware data flow control is active.
Error codes of the receive configuration		
16#8201 [1)]	Receive_Conditions is a pointer to an invalid data type	Enter a pointer to one of the following data types: DB, BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TIME_OF_DAY, TIME, S5TIME, DATE_AND_TIME, STRING
16#8225	Receive_Conditionspoints to an optimized memory area greater than 1 kB or Receive_Conditions points to an optimized memory area and the receive length is greater than the area addressed by Receive_Conditions.	Enter a pointer to an area with a maximum length of: • Optimized memory area: 1 kByte • Non-optimized area: 4 kByte Note: If the pointer points to an optimized memory area, do not send more than 1 kByte.
16#8229 [1)]	Receive_Conditions is a pointer to BOOL with a number of bits not equal to n * 8	If you are using a pointer to BOOL, the number of bits must be a multiple of 8.
Error codes, general		

Error code	Description	Remedy
16#8280	Negative acknowledgment when reading module	You can find more detailed information on error causes in theRDREC.STATUS static parameters and in the description of the SFB RDREC. • Check the input at the PORT parameter • Set the COM_RST parameter prior to the initial call.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter You can find more detailed information on error causes in the WRREC.STATUS static parameters and in the description of the SFB WRREC.
16#8282	Module not available	Check the input at the PORT parameter and ensure that the module can be reached.
Error codes of the receive configuration		
16#82C1	Invalid value for "Buffered received frames".	Select a suitable value for "Buffered received frames". Valid range of values: 1-255
16#82C2	Receive configuration rejected because the 3964(R) protocol was selected	Make sure that no receive configuration is sent if the 3964(R) protocol is set.
16#8301 1)	Receive_Conditions is a pointer to an invalid data type	Select a valid data type. The following are valid: DB, BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TIME_OF_DAY, TIME, S5TIME, DATE_AND_TIME, STRING
16#8322	Range length error when reading a parameter	Check the input at the Receive_Conditions parameter
16#8324	Range error when reading a parameter	Check the input at the Receive_Conditions parameter
16#8328	Setting error when reading a parameter	Check the input at the Receive_Conditions parameter
SEND status and error codes		
16#8328 1)	BUFFER is a pointer to BOOL with a number of bits not equal to n * 8	If you are using a pointer to BOOL, the number of bits must be a multiple of 8.
Error codes of the receive configuration		
16#8332	Invalid data block at the Receive_Conditions parameter	Check the input at the Receive_Conditions parameter
16#833A	The designation of the data block at the Receive_Conditions parameter refers to a data block which is not loaded.	Check the input at the Receive_Conditions parameter
16#8351	Invalid data type	Check the input at the Receive_Conditions parameter
16#8352 1)	Receive_Conditions does not point to a data block	Check the pointer to Receive_Conditions
16#8353 1)	Receive_Conditions does not point to a structure of the type Receive_Conditions	Check the pointer to Receive_Conditions
Error codes 3964(R) protocol		
16#8380	Parameter assignment error: Invalid value for "Character delay time".	Select a suitable value for "Character delay time" (CharacterDelayTime). Valid range of values: 1 to 65535 (ms)
16#8381	Parameter assignment error: Invalid value for "Response timeout".	Select a suitable value for "Response timeout" (AcknDelayTime). Valid range of values: 1 to 65535 (ms)

Error code	Description	Remedy
16#8382	Parameter assignment error: Invalid value for "Priority".	Select a suitable value for "Priority" (Priority). The following are valid: • High (1) • Low (0)
16#8383	Parameter assignment error: Invalid value for "Block check"	Select a suitable value for "Block check" (BCC). The following are valid: • With block check (1) • Without block check (0)
16#8384	Parameter assignment error: Invalid value for "Connection attempts".	Select a suitable value for "Connection attempts" (BuildupAttempts). Valid range of values: 1-255
16#8385	Parameter assignment error: Invalid value for "Transmission attempts".	Select a suitable value for "Transmission attempts" (RepetitionAttempts). Valid range of values: 1-255
16#8386	Runtime error: Number of connection attempts exceeded	Check the interface cable and the transmission parameters. Also check whether the receive function is configured correctly at the partner device.
16#8387	Runtime error: Number of transmission attempts exceeded	Check the interface cable, the transmission parameters and the configuration of the communication partner.
16#8388	Runtime error: Error at the "Block check character" The internally calculated value of the block check character does not correspond to the block check character received by the partner at the connection end.	Check whether the connection is seriously disrupted; in this case you may also occasionally see error codes. Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#8389	Runtime error: Invalid character received while waiting for free receive buffer	The send request of the communication partner (STX, 02H) is only answered with DLE when the receive buffer is empty. No additional character may be received before (except STX again). Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#838A	Runtime error: Logical error during receiving. After DLE was received, a further random character (other than DLE or ETX) was received.	Check if partner DLE in frame header and in data string is always in duplicate or the connection is released with DLE ETX. Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#838B	Runtime error: Character delay time exceeded	Partner device too slow or faulty. Verify by using an interface test device that is switched into the transmission line, if necessary.
16#838C	Runtime error: Wait time for free receive buffer has started	In the user program call the instruction more often or configure a communication with data flow control.
16#838D	Runtime error: frame repetition does not start within 4 s after NAK	Check the communication partner. A received frame that is possibly corrupted must be repeated by the partner within 4 seconds.

Error code	Description	Remedy
16#838E	Runtime error: In idle mode, one or several characters (other than NAK or STX) were received.	Check for proper function of the partner device, possibly using an interface test device that is switched into the transmission line.
16#838F	Runtime error: Initialization conflict - Both partners have set high priority	Set the "Low" priority at one of the partners
16#8391	Parameter assignment error: 3964 configuration data rejected because Freeport is set	If the Freeport protocol is set, make sure that no 3964 parameter assignment data is sent.

[1] Only with instructions for S7-300/400 CPUs

Overview of error messages - Modbus

Error code	Description	Remedy
16#0000	No error	-
Configuration error of the interface - Modbus_Comm_Load		
16#8181	The module does not support this data transmission rate.	Select a valid data transmission rate for the module at the BAUD parameter.
16#8182	The module does not support this parity setting.	Select a suitable value for "Parity" at the PARITY parameter. The following are valid: • None (1) • Even (2) • Odd (3) • Mark (4) • Space (5) • Any (6)
16#8183	The module does not support this type of data flow control.	Select a valid data flow control for the module at the FLOW_CTRL parameter.
16#8184	Invalid value for "Response timeout".	Select a suitable value for "Response timeout" at the RESP_TO parameter. Valid range of values: 1 to 65535 (ms)
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the Send_Config.RDREC.STATUS or Receive_Config.RDREC.STATUS static parameters or RDREC.STATUS and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the Send_Config.WRREC.STATUS or Receive_Config.WRREC.STATUS static parameters or WRREC.STATUS and in the description of the SFB WRREC.
16#8282	Module not available	Check the input at the PORT parameter and ensure that the module can be reached.
Configuration error - Modbus_Slave		

Error code	Description	Remedy
16#8186	Invalid slave address	Select a suitable slave address at the MB_ADDR parameter. The following are valid: 1-247 at standard address area; 1-65535 at extended address area (0 is reserved for Broadcast)
16#8187	Invalid value at MB_HOLD_REG parameter	Select a suitable value for the hold register at the MB_HOLD_REG parameter.
16#8188	Invalid operating mode or broadcast (MB_ADDR = 0) and MODE parameter ≠ 1	Select the value 1 for MODE in Broadcast mode or select a different operating mode.
16#818C	The pointer to a MB_HOLD_REG area must be a data block or a bit memory address area.	Select a suitable value for the pointer to the MB_HOLD_REG area.
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.WRREC.STATUS or Receive_P2P.WRREC.STATUS, and in the description of the SFB WRREC.
16#8452 [1)]	MB_HOLD_REG is not a pointer to a DB or a bit memory area	Check the MB_HOLD_REG pointer
16#8453 [1)]	MB_HOLD_REG is not a pointer of type BOOL or WORD	Check the MB_HOLD_REG pointer
16#8454 [1)]	The area addressed by MB_HOLD_REG is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the MB_HOLD_REG pointer
16#8455 [1)]	MB_HOLD_REG points to a write-protected DB	Check the MB_HOLD_REG pointer
16#8456 [1)]	Error during instruction execution. The cause of the error is shown in the STATUS parameter.	Determine the value of the SFCSTATUS parameter. Check what this means in the description for SFC51, STATUS parameter.
Configuration error - Modbus_Master		
16#8180	Invalid value for MB_DB parameter	The value configured for MB_DB (instance data DB) at the Modbus_Comm_Load instruction is not valid. Check the interconnection of the Modbus_Comm_Load instruction and its error messages.
16#8186	Invalid station address	Select a suitable station address at the MB_ADDR parameter. The following are valid: 1-247 at standard address area; 1-65535 at extended address area (0 is reserved for Broadcast)
16#8188	Invalid operating mode or broadcast (MB_ADDR = 0) and MODE parameter ≠ 1	Select the value 1 for MODE in Broadcast mode or select a different operating mode.

Error code	Description	Remedy
16#8189	Invalid data address	Select a suitable value for the data address at the DATA_ADDR parameter. See description Modbus_Master (Page 7536) in the Info system
16#818A	Invalid length	Select a suitable data length at the DATA_LEN parameter. See description Modbus_Master (Page 7536) in the Info system
16#818B	Invalid value for DATA_PTR	Select a suitable value for the data pointer at the DATA_PTR parameter (M or DB address). See description Modbus_Master (Page 7536) in the Info system
16#818C	Interconnection error of the DATA_PTR parameter	Check the interconnection of the instruction.
16#818D	The area addressed by DATA_PTR is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the DATA_PTR pointer
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.WRREC.STATUS or Receive_P2P.WRREC.STATUS or Receive_Reset and in the description of the SFB WRREC.
Communication errors - Modbus_Master and Modbus_Slave		
16#80 D1	The waiting time for XON or CTS = ON has expired.	The communication partner has a fault, is too slow or is offline. Check the communication partner or change the parameters, if necessary.
16#80D2	"Hardware RTS always ON": Send job canceled due to change from DSR = ON to OFF	Check the communication partner. Make sure that DSR is ON for the entire duration of transmission.
16#80E0	Frame aborted: Send buffer overflow / send frame too long	In the user program call the instruction more often or configure a communication with data flow control.
16#80E1	Frame aborted: Parity error	Check the connection line of the communication partners, or verify that both devices are configured for the same data transmission rate, parity and stop bit number.
16#80E2	Frame aborted: Character frame error	Check the settings for start bit, data bits, parity bit, data transmission rate, and stop bit(s).
16#80E3	Frame aborted: Character overflow error	Check the number of data in the frame of the communication partner.
16#80E4	Frame aborted: Maximum frame length reached	Select a shorter frame length at the communication partner. The following are valid (depending on the module): 1 to 1024/2048/4096 (bytes)

Error code	Description	Remedy
Communication error - Modbus_Master		
16#80C8	The slave does not respond within the set time	Check the data transmission rate, parity and wiring of the slave.
16#8200	The interface is busy with an ongoing request.	Repeat the command later. Make sure that there are no commands still running before you start a new one.
Protocol error - Modbus_Slave (only communication modules that support Modbus)		
16#8380	CRC error	Checksum error of the Modbus frame. Check the communication partner.
16#8381	The function code is not supported or is not supported for broadcast.	Check the communication partner and make sure that a valid function code is sent.
16#8382	Invalid length information in the request frame	Select a suitable data length at the DATA_LEN parameter.
16#8383	Invalid data address in the request frame	Select a suitable value for the data address at the DATA_ADDR parameter.
16#8384	Invalid data value error in the request frame	Check the data value in the request frame of the Modbus master
16#8385	The diagnostic value is not supported by the Modbus slave (function code 08)	The Modbus slave only supports the diagnostic values 16#0000 and 16#000A.
Protocol error - Modbus_Master (only communication modules that support Modbus)		
16#8380	CRC error	Checksum error of the Modbus frame. Check the communication partner.
16#8381	Response frame from Modbus Slave with the following error message: The function code is not supported.	Check the communication partner and make sure that a valid function code is sent.
16#8382	Response frame from Modbus Slave with the following error message: Invalid length	Select a suitable data length.
16#8383	Response frame from Modbus Slave with the following error message: Invalid data address in the request frame	Select a suitable value for the data address at the DATA_ADDR parameter.
16#8384	Response frame from Modbus Slave with the following error message: Data value error	Check the request frame to the Modbus slave.
16#8385	Response frame from Modbus Slave with the following error message: The diagnostic value is not supported by the Modbus slave	The Modbus slave only supports the diagnostic values 16#0000 and 16#000A.
16#8386	The returned function code does not match the requested function code.	Check the response frame and the addressing of the slave.
16#8387	A slave that was not requested answers	Check the response frame of the slave. Check the address settings of the slave.
16#8388	Error in the response of the slave to a write request.	Check the response frame of the slave.
16#8828 [1)]	DATA_PTR points to a bit address that is not equal to n * 8	Check the DATA_PTR pointer
16#8852 [1)]	DATA_PTR is not a pointer to a DB or a bit memory area	Check the DATA_PTR pointer
16#8853 [1)]	DATA_PTR is not a pointer of type BOOL or WORD	Check the DATA_PTR pointer
16#8855 [1)]	DATA_PTR points to a write-protected DB	Check the DATA_PTR pointer
16#8856 [1)]	Error during call of SFC51	Call the Modbus_Master instruction again
Error - Modbus_Slave (only communication modules that support Modbus)		

Error code	Description	Remedy
16#8428 [1]	MB_HOLD_REG points to a bit address that is not equal to n * 8	Check the MB_HOLD_REG pointer
16#8452 [1]	MB_HOLD_REG is not a pointer to a DB or a bit memory area	Check the MB_HOLD_REG pointer
16#8453 [1]	MB_HOLD_REG is not a pointer of type BOOL or WORD	Check the MB_HOLD_REG pointer
16#8454 [1]	The area addressed by MB_HOLD_REG is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the MB_HOLD_REG pointer
16#8455 [1]	MB_HOLD_REG points to a write-protected DB	Check the MB_HOLD_REG pointer
16#8456 [1]	Error during call of SFC51	Call the Modbus_Slave instruction again

[1] Only with instructions for S7-300/400 CPUs

Overview of error messages - USS

Error code	Description	Remedy
16#0000	No error	-
16#8180	Length error in response of drive	Check the response frame of the drive.
16#8181	Data type error	Select a suitable data type. The following are valid: • Real • Word • Double word
16#8182	Data type error: "Double word" or "Real" may not be returned for the "Word" request.	Check the response frame of the drive.
16#8183	Data type error: "Word" may not be returned for the "Double word" or "Real" request.	Check the response frame of the drive.
16#8184	Checksum error in response of drive	Check the drive and the communication connection.
16#8185	Addressing error	Valid drive address range: 1 to 16
16#8186	Setpoint error	Valid setpoint range: -200 % to +200 %
16#8187	Incorrect drive number returned	Check the response frame of the drive.
16#8188	Invalid PZD length	Permitted PZD lengths: 2, 4, 6, 8 words
16#8189	The module does not support this data transmission rate.	Select a valid data transmission rate for the module.
16#818A	A different request for this drive is currently active.	Repeat the parameter read or write command later.
16#818B	The drive does not respond.	Check the drive.
16#818C	The drive responds with an error message to a parameter request.	Check the response frame of the drive. Check the parameter request.
16#818D	The drive responds with an access error message to a parameter request.	Check the response frame of the drive. Check the parameter request.
16#818E	The drive was not initialized.	Check the user program and make sure that the USS_Drive_Control instruction is called for this drive.

Error code	Description	Remedy
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Port_Config.RDREC.STATUS, Send_Config.RDREC.STATUS , Receive_Config.RDREC.STATUS , Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Port_Config.WRREC.STATUS, Send_Config.WRREC.STATUS , Receive_Config.WRREC.STATUS , Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB WRREC.

[1] Only with instructions for S7-300/400 CPUs

MODBUS (RTU)

Overview of the Modbus RTU communication

Modbus RTU communication

Modbus RTU (Remote Terminal Unit) is a standard protocol for communication in the network and uses the RS232 or RS422/485 connection for serial data transmission between Modbus devices in the network.

Modbus RTU uses a master/slave network in which all communication is triggered by a single master device while the slaves can only respond to the request of the master. The master sends a request to a slave address and only the slave with this slave address responds to the command.

Exception: Modbus slave address 0 sends a broadcast frame to all slaves (without slave response).

Modbus function codes

- A CPU that is operated as a Modbus RTU master can read and write data and I/O states in a Modbus RTU slave connected by means of a communication connection.
- A CPU operated as a Modbus RTU slave allows a Modbus RTU master connected over a communication connection to read and write data and I/O states in its own CPU.

Table 9-301 Functions for reading data: Reading distributed I/O and program data

Modbus function code	Functions for reading data from the slave (server) - standard addressing
01	Read output bits: 1 to 2000/1992[1)] bits per request
02	Read input bits: 1 to 2000/1992[1)] bits per request
03	Read hold register: 1 to 125/124[1)] words per request
04	Read input words: 1 to 125/124[1)] words per request

1) for extended addressing

Table 9-302 Functions for writing data: Changing distributed I/O and program data

Modbus function code	Functions for writing of data in the slave (server) - standard addressing
05	Write one output bit: 1 bit per request
06	Write one hold register: 1 word per request
15	Write one or several output bits: 1 to 1960 bits per request
16	Write one or several hold registers: 1 to 122 words per request

- The Modbus function codes 08 and 11 offer diagnostic options for communication with the slave device.
- Modbus slave address 0 sends a broadcast frame to all slaves (without slave response; for function codes 5, 6, 15, 16).

Table 9-303 Station addresses in the Modbus network

Station		Address
RTU station	Standard station address	1 to 247 and 0 for broadcast
	Extended station address	1 to 65535 and 0 for broadcast

Modbus memory addresses

The number of Modbus memory addresses (input/output addresses) that is actually available depends on the CPU version and the available work memory.

Modbus RTU instructions in your program

- Modbus_Comm_Load: You need to run Modbus_Comm_Load to set up PtP parameters such as data transmission rate, parity and data flow control. Once you have configured the communication module for the Modbus RTU protocol, it can only be used by the Modbus_Master instruction or the Modbus_Slave instruction.
- Modbus_Master: The CPU can be used as Modbus RTU master device with the Modbus master instruction for communication with one or more Modbus slave devices.
- Modbus_Slave: The CPU can be used as Modbus RTU slave device with the Modbus slave instruction for communication with one Modbus master device.

Modbus_Comm_Load: Configure communication module for Modbus

Description

The Modbus_Comm_Load instruction configures a communication module for communication by means of the Modbus RTU protocol. An instance data block is automatically assigned when you add the Modbus_Comm_Load instruction in your program.

Parameters

Parameters	Declaration	Data type		Standard	Description
		S7-1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	Starts the instruction upon a positive edge of this input.
PORT	IN	Port	Laddr	0	The CM port value can be found in the "Hardware ID" property of the device configuration (S7-1500) (input address; S7-300/400) once you have configured the CM. The symbolic port name is assigned in the "System constants" tab of the PLC tag table. In S7-300/400/WinAC systems, the input address assigned in HWCN must be assigned to the CM port.
BAUD	IN	UDInt	DWord	9600	Selection of the data transmission rate Valid values are: 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 76800, 115200 bit/s.
PARITY	IN	UInt	Word	1	Selection of parity: • 0 – None • 1 – Odd • 2 – Even
FLOW_CTRL	IN	UInt	Word	0	Selection of flow control: • 0 – (default) no flow control • 1 – Hardware flow control with RTS always ON (not with RS422/485 CMs) • 2 – Hardware flow control with RTS switched (not with RS422/485 CMs)
RTS_ON_DLY	IN	UInt	Word	0	Selection RTS ON delay: • 0 – No delay from "RTS active" until the first character of the frame is sent. • 1 to 65535 – Delay in milliseconds from "RTS active" until the first character of the frame is sent (not with RS422/485 CMs). RTS delays must be used independent of the selection FLOW_CTRL.

Parameters	Declaration	Data type		Standard	Description
		S7-1500	S7-300/400/WinAC		
RTS_OFF_DLY	IN	UInt	Word	0	Selection RTS OFF delay: • 0 – No delay after transmission of last character until "RTS inactive" • 1 to 65535 – Delay in milliseconds after transmission of last character until "RTS inactive" (not with RS422/485 ports). RTS delays must be used independent of the selection FLOW_CTRL.
RESP_TO	IN	UInt	Word	1000	Response timeout: 5 ms to 65535 ms - Time in milliseconds that Modbus_Master waits for a response from the slave. If the slave does not respond within this period, Modbus_Master repeats the request or terminates the request with an error if the specified number of repetitions (see below, RETRIES parameter) has been sent.
MB_DB	IN/OUT	MB_BASE		-	A reference to the instance data block of the Modbus_Master or Modbus_Slave instructions. The MB_DB parameter must be connected with the (static and therefore not visible in the instruction) MB_DB parameter of the Modbus_Master or Modbus_Slave instruction.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Modbus_Comm_Load instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
DONE	OUT	Bool		FALSE	The DONE bit is TRUE for one cycle after the last request has been completed without errors.
ERROR	OUT	Bool		FALSE	The ERROR bit is TRUE for one cycle after the last request has been completed with errors. The error code in the STATUS parameter is only valid in the cycle in which ERROR = TRUE.
STATUS	OUT	Word		16#7000	Error code (see Error messages (Page 7551))

Modbus_Comm_Load is executed to configure a port for the Modbus RTU protocol. Once you have configured the port for the Modbus RTU protocol, it can only be used by the Modbus_Master or Modbus_Slave instructions.

You have to run Modbus_Comm_Load for the configuration of each communication port that is to be used for Modbus communication. You must assign a unique Modbus_Comm_Load instance DB to each port that you use. Only run Modbus_Comm_Load again if you need to change communication parameters, such as data transmission rate or parity, or in case the network has returned.

For example, an instance data block is assigned to the instruction if you add Modbus_Master or Modbus_Slave to your program. You need to connect the MB_DB parameter of the Modbus_Comm_Load instruction to the MB_DB parameter of the Modbus_Master or Modbus_Slave instruction.

Modbus_Comm_Load data block tags

The table below shows the public static tags in the instance DB of Modbus_Comm_Load that you can use in your program.

Table 9-304 Static tags in the instance DB

Tag	Data type		Standard	Description
	S7-1500	S7-300/400/WinAC		
ICHAR_GAP	Word		0	Maximum character delay time between characters. This parameter is specified in milliseconds and increases the anticipated period between the received characters. The corresponding number of bit times for this parameter is added to the Modbus default value of 35 bit times (3.5 character times).
RETRIES	Word		2	Number of retries that the master executes before the error code 0x80C8 for "No response" is returned.
EN_SUPPLY_VOLT	Bool		0	Enable diagnostics for missing supply voltage L+
MODE	USInt	Byte	0	Operating mode Valid operating modes are: • 0 = Full duplex (RS232) • 1 = Full duplex (RS422) four-wire operation (point-to-point) • 2 = Full duplex (RS 422) four-wire mode (multipoint master, CM PtP (ET 200SP)) • 3 = Full duplex (RS 422) four-wire mode (multipoint slave, CM PtP (ET 200SP)) • 4 = Half duplex (RS485) two-wire mode
LINE_PRE	USInt	Byte	0	Receive line initial state Valid initial states are: • 0 = "no" initial state • 1 = signal R(A)=5 V, signal R(B)=0 V (break detection): Break detection is possible with this initial state. Can only be selected with: "Full duplex (RS422) four-wire operation (point-to-point connection)" and "Full duplex (RS422) four-wire operation (multipoint slave)". • 2 = signal R(A)=0 V, signal R(B)=5 V: This default setting corresponds to the idle state (no active send operation). No break detection is possible with this initial state.
CABLE_BRK	USInt	Byte	0	Break detection The following are valid: • 0 = break detection deactivated • 1 = break detection activated
EN_DIAG_ALARM	Bool		0	Activate diagnostics interrupt: • 0 - not activated • 1 - activated

Modbus_Master: Communicate as Modbus master

Description

The Modbus_Master instruction communicates as Modbus master via a port configured by the Modbus_Comm_Load instruction. An instance data block is automatically assigned when you add the Modbus_Master instruction in your program. The MB_DB parameter of the Modbus_Comm_Load instruction must be connected to the (static) MB_DB parameter of the Modbus_Master instruction.

Parameters

Parameters	Declaration	Data type		Standard	Description
		S7-1500	S7-300/400/ WinAC		
REQ	IN	Bool		FALSE	FALSE = no request TRUE = request to send data to the Modbus slave
MB_ADDR	IN	UInt	Word	-	Modbus RTU station address: Standard addressing range (1 to 247 as well as 0 for Broadcast) Extended addressing range (1 to 65535 as well as 0 for Broadcast) The value 0 is reserved for the broadcast of a frame to all Modbus slaves. Only the Modbus function codes 05, 06, 15 and 16 are supported for the broadcast.
MODE	IN	USInt	Byte	0	Mode selection: Specifies the type of request (read, write or diagnostics). Additional information is available in the table of Modbus functions below.
DATA_ADDR	IN	UDInt	DWord	0	Start address in the slave: Specifies the start address of the data that is accessed in the Modbus slave. The valid addresses are listed in the table of Modbus functions below.
DATA_LEN	IN	UInt	Word	0	Data length: Specifies the number of bits or words this instruction is to access. The valid lengths are listed in the table of Modbus functions below.
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Modbus_Master instruction You initialize the instruction with TRUE. The instruction then resets COM_RST to FALSE.
DATA_PTR	IN/OUT	Variant	Any	-	Data pointer: Points to the flag or DB address for the data to be written or read.
DONE	OUT	Bool		FALSE	The DONE bit is TRUE for one cycle after the last request has been completed without errors.
BUSY	OUT	Bool		-	• FALSE – no command active for Modbus_Master • TRUE – command for Modbus_Master in progress
ERROR	OUT	Bool		FALSE	The ERROR bit is TRUE for one cycle after the last request has been completed with errors. The error code in the STATUS parameter is only valid in the cycle in which ERROR = TRUE.
STATUS	OUT	Word		0	Error code (see Error messages (Page 7551))

Tags in the data block of the Modbus master

The table below shows the public static tags in the instance DB of Modbus_Master that you can use in your program.

Table 9-305 Static tags in the instance DB

Tag	Data type	Standard	Description
Blocked_Proc_Timeout	Real	0.5	Duration (in seconds) for which to wait for a blocked Modbus master instance before this instance is removed as ACTIVE. This may happen, for example, if a master request was output and the program then stops to call the master function before it has completely finished the request. The time value must be greater than 0 and less than 55 seconds to avoid an error to occur. See also "Rules for communication by the Modbus-Master" and "Calling theModbus_Master instruction with different parameter settings".
Extended_Addressing	Bool	FALSE	Configures the slave station address as single or double byte. • FALSE = One-byte address; 0 to 247 • TRUE = Two-byte address (corresponds to extended addressing); 0 to 65535
Compatibility_Mode [1]	Bool	FALSE	Compatibility mode with CP 341 and CP 441-2 with driver for Modbus RTU and with ET 200SP 1SI for Modbus. The default value is 0. • FALSE = as per Modbus specification, not compatible • TRUE = compatible – For FC1 and FC2: The data read from the received frame is written word for word to the addressed CPU memory and exchanged byte by byte. If the number of bits to be transmitted is not a multiple of 16, the bits which are not relevant are set to null in the last word. – For FC15: The words to be transmitted are read word by word from the addressed memory and written byte by byte to the send frame. If the number of bits to be transmitted is not a multiple of 8, the bits in the last byte which are not relevant are read unchanged from the addressed memory and entered in the send frame.
MB_DB	MB_BASE	-	The MB_DB parameter of the Modbus_Comm_Load instruction must be connected to this MB_DB parameter of the Modbus_Master instruction.

[1] The PtP communication modules respond as defined in the Modbus specification. To retain a response as with CP 341, CP 441-2 and ET 200SP 1SI for Modbus, use the "Compatibility_Mode" parameter.

You program can write values to the Blocked_Proc_Timeout and Extended_Addressing variables to control the Modbus master operations.

Rules for communication by the Modbus-Master

- Modbus_Comm_Load must be run to configure a port so that the Modbus_Master instruction can communicate with this port.
- A port which is to be used as Modbus master must not be used by Modbus _Slave . You can use one or several instances of Modbus_Master [1)] with this port. But all versions of the Modbus_Master must use the same instance DB for the port.
- The Modbus instructions do not use communication alarm events to control the communication process. Your program must query the Modbus_Master instruction for completed commands (DONE, ERROR).
- We recommend to call all executions of Modbus_Master for a specific port from a program cycle OB. Modbus master instructions can only be executed in one program cycle or in one cyclical/time-controlled processing level. They may not be processed in different processing levels. The priority interruption of a Modbus master instruction by another Modbus master instruction in a processing level with higher priority results in improper operation. Modbus master instructions may mot be processed in startup, diagnostic or time error levels.

[1)] "Instance of Modbus master" here means a call of the Modbus_Master instruction with the same interconnection to a Modbus_Comm_Load instruction and the same setting for the MB_ADDR, MODE, DATA_ADDR and DATA_LEN parameters.

Example

Modbus_Master is called with MODE=0 and DATA_ADDR=10

This job is now active until it is completed with DONE=1 or ERROR=1 or until the time monitoring configured at the Blocked_Proc_Timeout parameter has expired. If a new command is started after the watchdog time expires and before the previous command has been completed, the previous command is aborted without an error message.

If, while this command is running, the instruction is now called a second time with the same instance data but different MODE and DATA_ADDR parameter settings, this second call is terminated with ERROR=1 and STATUS=8200.

Calling the Modbus_Master instruction with different parameter settings

If multiple calls of the Modbus_Master instruction with different settings for MB_ADDR, MODE, DATA_ADDR or DATA_LEN are placed in your program, you must ensure that only one of these calls is active at any given time. Otherwise, the error message 0x8200 is output (interface is busy with an ongoing request).

If a call cannot be processed in full, the watchdog is activated by the Blocked_Proc_Timeout parameter and terminates the ongoing command.

REQ parameter

FALSE = no request; TRUE = request to send data to the Modbus slave

Enable the requested transmission. This transmits the contents of the buffer to the point-to-point communication interface.

You use the DATA_ADDR and MODE parameters to select the Modbus function code.

DATA_ADDR (Modbus start address in the slave): Specifies the start address of the data that is accessed in the Modbus slave.

The Modbus_Master instruction uses the MODE input instead of a function code input. The combination of MODE and DATA_ADDR specifies the function code that is used in the actual Modbus frame. The table below shows how the MODE parameter, the Modbus function code and the Modbus address range in DATA_ADDR are related.

Table 9-306 Modbus functions

MODE	DATA_ADDR (Modbus address)			DATA_LEN (data length)			Modbus function code	Operation and data		
0				Bits per request			01	Read output bits:		
	1	to	9999	1	to	2000/1992 [1]		0	to	9998
0				Bits per request			02	Read input bits:		
	10001	to	19999	1	to	2000/1992 [1]		0	to	9998
0				Words per request			03	Read hold register:		
	40001	to	49999	1	to	125/124 [1]		0	to	9998
	400001	to	465535	1	to	125/124 [1]		0	to	65534
0				Words per request			04	Read input words:		
	30001	to	39999	1	to	125/124 [1]		0	to	9998
1				Bits per request			05	Write one output bit:		
	1	to	9999	1				0	to	9998
1				1 word per request			06	Write one hold register:		
	40001	to	49999	1				0	to	9998
	400001	to	465535	1				0	to	65524
1				Bits per request			15	Write multiple output bits:		
	1	to	9999	2	to	1968/1960 [1]		0	to	9998
1				Words per request			16	Write multiple hold registers:		
	40001	to	49999	2	to	123/122		0	to	9998
	400001	to	465534	2	to	123/122 [1]		0	to	65534
2 [2]				Bits per request			15	Write one or several output bits:		
	1	to	9999	2	to	1968/1960 [1]		0	to	9998
2 [2]				Words per request			16	Write one or several hold registers:		
	40001	to	49999	1	to	123		0	to	9998
	400001	to	465535	1	to	122 [1]		0	to	65534
11	Both DATA_ADDR and DATA_LEN operands of the Modbus_Master are ignored with this function.						11	Read status word and event counter of the slave communication. The status word indicates busy (0 – not busy, 0xFFFF - busy). The event counter is incremented for each successful processing of a frame.		

MODE	DATA_ADDR (Modbus address)			DATA_LEN (data length)			Modbus function code	Operation and data		
80				1 word per request			08	Check slave status with data diagnostic code 0x0000 (loopback test – slave returns an echo of the request)		
	-			1				-		
81				1 word per request			08	Reset slave event counter using data diagnostic code 0x000A		
	-			1				-		
3 to 10, 12 to 79, 82 to 255	-			-				Reserved		

[1] In extended addressing, see the Extended_Adressing parameter, the maximum data length is shorter by 1 byte or 1 word depending on the data type of the function.

[2] MODE 2 allows you to write one or more output bits and one or more hhold registers using the Modbus functions 15 and 16. MODE 1 uses the Modbus functions 5 and 6 to write 1 output bit and 1 hold register, and Modbus functions 15 and 16 to write multiple output bits and multiple hold registers.

DATA_PTR parameter

The DATA_PTR parameter points to the DB or bit memory address in which reading or writing is performed. If you use a data block, you must create a global data block that provides the data memory for read and write processes on Modbus slaves.

Note

S7-1500 - The data block addressed using DATA_PTR must support direct addressing

The data block must permit direct (absolute) and symbolic addressing.

Data block structures for the DATA_PTR parameter

- These data types are valid for **reading words** of Modbus addresses (DATA_PTR) 30001 to 39999, 40001 to 49999 and 400001 to 465535 as well as for **writing words** to Modbus addresses (DATA_PTR parameter) 40001 to 49999 and 400001 to 465535.
  - Standard array of data types WORD, UINT or INT
  - Named structure of the WORD, UINT or INT type in which each element has a unique name and a 16-bit data type.
  - Named complex structure in which each element has a unique name and a 16-bit or 32-bit data type.
- For **reading** and writing bits for the Modbus addresses (DATA_PTR parameter) 00001 to 09999 and for reading of bits from 10001 to 19999.
  - Standard field from Boolean data types.
  - Named Boolean structure from uniquely named Boolean variables.
- It is not essential, but nevertheless advisable, to allocate each Modbus_Master instruction its own separate memory area. The reason for this is that data destruction is far more likely if multiple Modbus_Master instructions are reading and writing in the same memory area.
- It is not necessary for the data areas for DATA_PTR to be located in the same global data block. You can create a data block with several areas for Modbus read processes, a data block for Modbus write processes or a data block for each slave station.

See also

Modbus_Slave: Communicate as Modbus slave (Page 7541)

Modbus_Slave: Communicate as Modbus slave

Description

Your program can use the Modbus_Slave instruction to communicate as a Modbus slave by using a CM (RS422/485 or RS232). STEP 7 automatically creates an instance DB when you add the instruction. The MB_DB parameter of the Modbus_Comm_Load instruction must be connected to the (static) MB_DB parameter of the Modbus_Slave instruction.

Parameters

Parameters	Declaration	Data type		Standard	Description
		S7-1500	S7-300/400/ WinAC		
MB_ADDR	IN	UInt	Word	-	Standard address of the Modbus slave: Standard addressing range (1 to 247) Extended addressing range (0 to 65535) Note: 0 is the broadcast address
COM_RST	IN/OUT	---	Bool	FALSE	Initialization of the Modbus_Slave instruction The instruction is initialized with TRUE. The instruction then resets COM_RST to FALSE.
MB_HOLD_REG	IN/OUT	Variant	Any	-	Pointer to the Modbus hold register DB: The Modbus hold register may be the memory area of the flags or a data block.
NDR	OUT	Bool		FALSE	New data available: • FALSE – No new data • TRUE – Indicates that new data was written by the Modbus master The NDR bit is TRUE for one cycle after the last request has been completed without errors.
DR	OUT	Bool		FALSE	Read data: • FALSE – No data read • TRUE – Indicates that the instruction has stored the data received by the Modbus master in the target area. The DR bit is TRUE for one cycle after the last request has been completed without errors.
ERROR	OUT	Bool		FALSE	The ERROR bit is TRUE for one cycle after the last request has been completed with errors. If the execution was terminated with an error, the error code in the STATUS parameter is only valid in the cycle in which ERROR = TRUE.
STATUS	OUT	Word		0	Error code (see Error messages (Page 7551))

The function codes of the Modbus communication (1, 2, 4, 5 and 15) can read and write bits and words directly in the process image of the inputs and in the process image of the outputs of the CPU. The MB_HOLD_REG parameter must be defined as data type greater than one byte for these function codes. The table below shows the sample assignment of Modbus addresses to the process image in the CPU.

Table 9-307 Assignment of Modbus addresses to the process image

Modbus functions						S7-1500 / S7-300 / S7-400			
Function code	Function	Data area	Address area			Data area	CPU address		
01	Read bits	Output	0	to	9998	Process image of outputs	O0.0	to	O1248.6
02	Read bits	Input	0	to	9998	Process image of inputs	I0.0	to	I1248.6
04	Read words	Input	0	to	9998	Process image of inputs	IW0	to	IW19996

Modbus functions						S7-1500 / S7-300 / S7-400			
Function code	Function	Data area	Address area			Data area	CPU address		
05	Write bit	Output	0	to	9998	Process image of outputs	O0.0	to	O1248.6
15	Write bits	Output	0	to	9998	Process image of outputs	O0.0	to	O1248.6

Note

The available address area may be smaller, depending on the memory configuration of the CPU.

The function codes of the Modbus communication (3, 6, 16) use a Modbus hold register which is an address area in the memory area of the flags or a data block. The type of hold register is specified by the MB_HOLD_REG parameter of the Modbus_Slave instruction.

Note

S7-1500 - Type of MB_HOLD_REG data block

The data block with Modbus hold register must permit direct (absolute) and symbolic addressing.

Variables of the Modbus slave

This table below shows the public static tags in the instance data block of Modbus_Slave that you can use in your program.

Table 9-308 Variables of the Modbus slave

Tag	Data type	Standard	Description
HR_Start_Offset	Word	0	Specifies the start address of the Modbus hold register (default = 0)
Extended_Addressing	Bool	FALSE	Extended addressing, configures slave addressing as single or double byte (FALSE = single byte address, TRUE = double byte address)
Request_Count	Word	0	The number of all requests received by this slave
Slave_Message_Count	Word	0	The number of requests received for this specific slave
Bad_CRC_Count	Word	0	The number of received requests that have a CRC error
Broadcast_Count	Word	0	The number of received broadcast requests
Exception_Count	Word	0	Modbus-specific errors that are acknowledged with an exception to the master
Success_Count	Word	0	The number of received requests without protocol errors for this specific slave
MB_DB	MB_BASE	-	The MB_DB parameter of the Modbus_Comm_Load instruction must be connected to this MB_DB parameter of the Modbus_Master instruction.

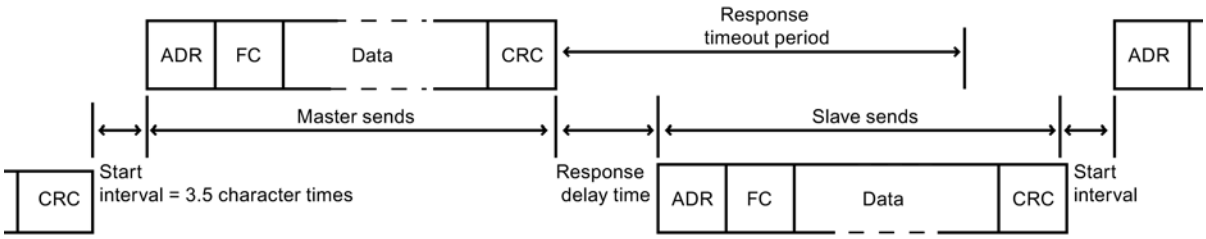
You program can write values to the HR_Start_Offset and Extended_Addressing variables and control the Modbus slave operations. The other variables can be read to monitor the Modbus status.

Rules for Modbus slave communication

- Modbus_Comm_Load must be run to configure a port so that the Modbus_Slave instruction can communicate by means of this port.
- If a port is to respond as slave to a Modbus master, this port may not be programmed with the Modbus_Master instruction.
- Only one instance of Modbus_Slave can be used with a specific port; otherwise you may encounter unexpected behavior.
- The Modbus instructions do not use communication alarm events to control the communication process. Your program must control the communication process by querying the Modbus_Slave instruction for completed send and receive processes.
- The Modbus_Slave instruction must be executed regularly with a frequency that allows a timely response to incoming requests of a Modbus master. We recommend executing Modbus_Slave in each cycle from a program cycle OB. Modbus_Slave can be executed from a cyclic interrupt OB but we do not recommend it, because excessive time delays in the interrupt program can temporarily block the execution of other interrupt programs.

Time control of the Modbus signal

Modbus_Slave must be executed regularly to receive each request of the Modbus master and respond accordingly. The frequency with which Modbus_Slave is executed depends on the timeout value specified for the response by the Modbus master. This can be seen in the figure below.



The timeout period of the (RESP_TO) response is the duration that a Modbus master waits for the beginning of an answer from a Modbus slave. This period is not defined by the Modbus protocol, but by a parameter of the Modbus_Comm_Load instruction. As both receiving and sending a frame requires multiple calls of the Modbus_Slave instruction (at least three), you should execute Modbus_Slave at least twelve times during the timeout period for the response of the Modbus master so that the Modbus slave receives and sends data twice as many times as specified by the timeout period.

HR_Start_Offset

The addresses of the Modbus hold register start with 40001 or 400001; these addresses correspond to the start address of the hold register in the target system memory. But you can

configure the HR_Start_Offset tag to configure a start address different than 40001 or 400001 for the Modbus hold register.

The address 0 in the received frame correspond to the start address of the hold register in the target system memory. Use the tag HR_Start_Offset to configure a start address other than 0 for the Modbus hold register.

You can, for example, configure a hold register with start at MW100 and a length of 100 words. With HR_Start_Offset = 20 the address 20 in the received frame corresponds to the start address of the hold register in the target memory (MW100). Each address in the received frame below 20 and above 119 results in an addressing error.

Table 9-309 Example for addressing the Modbus hold register when DATA_PTR is a pointer to MW100 with a length of 100 words

HR_Start_Offset	Address	Minimum	Maximum
0	Modbus address (word)	0	99
	S7-1500 address	MW100	MW298
20	Modbus address (word)	20	119
	S7-1500 address	MW100	MW298

HR_Start_Offset is a word value which specifies the start address of the Modbus hold register and is saved in the Modbus_Slave instance data block. You select this public static tag by means of the parameter drop-down list once you have added Modbus_Slave in your program.

If you have added Modbus_Slave to an LAD network, for example, you can go to a previous network and assign the value HR_Start_Offset using the move command. The value must be assigned prior to execution of Modbus_Slave.

Enter Modbus slave tag using the standard DB name:

1. Position the cursor in the OUT1 parameter field and enter the character m.
2. Select the required instance DB of the Modbus_Slave instruction from the drop-down list.
3. Position the cursor to the right of the DB name (after the quotation mark) and enter a point.
4. Select "Modbus_Slave_DB.HR_Start_Offset" in the drop-down list.

Frame structure

Extended_Addressing

You access the Extended_Addressing tag as described for the HR_Start_Offset reference, except that the Extended_Addressing tag is a Boolean value.

You can configure a single byte (Modbus standard) or two bytes (Extended_Adressing = TRUE) with Extended_Adressing = FALSE for addressing the Modbus slave. Extended addressing is used to address more than 247 devices in a single network. With Extended_Adressing = TRUE you can address up to 65535 addresses. The following example shows a Modbus frame.

Table 9-310 Slave address with one byte (Byte 0)

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	
Request	Slave address	Function code	Start address		Data		
Valid response	Slave address	Function code	Length	Data...			
Error message	Slave address	0xxx	Exception code				

Table 9-311 Slave address with two bytes (Byte 0 and Byte 1)

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Request	Slave address		Function code	Start address		Data	
Valid response	Slave address		Function code	Length	Data...		
Error message	Slave address		0xxx	Exception code			

Frame description

Data traffic between master and slave / slave and master starts with the slave address, following by the function code. The data is then transferred. The structure of the data field depends on the function code used. The checksum (CRC) is transmitted at the end of the frame.

Function code 1 - This function allows individual output bits to be read

Table 9-312 FC 1 - Read output bits

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 1	Start address		Number of outputs	
Valid response	Slave address	Function code 1	Length [1]	Output data [3)]		
Error message	Slave address	0x81	Exception code [2]	---		

[1] Length: If there is a remainder when the number of outputs is divided by 8, the number of bytes must be increased by 1.

[2] E code: 01 or 02 or 03 or 04

[3] The output data can comprise multiple bytes

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 1	Start address		Number of outputs	
Valid response	Slave address		Function code 1	Length [1]	Output data		
Error message	Slave address		0x81	Exception code [2]	---		

[1] Length: If there is a remainder when the number of outputs is divided by 8, the number of bytes must be increased by 1.

[2] E code: 01 or 02 or 03 or 04

[3] The output data can comprise multiple bytes

Function code 2 - This function allows individual input bits to be read

Table 9-313 FC 2 - Read input bits

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 2	Start address		Number of inputs	
Valid response	Slave address	Function code 2	Length [1]	Input data		
Error message	Slave address	0x82	Exception code [2]	---		

[1] Length: If there is a remainder when the number of inputs is divided by 8, the number of bytes must be increased by 1.

[2] E code: 01 or 02 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 2	Start address		Number of inputs	
Valid response	Slave address		Function code 2	Length [1]	Input data		
Error message	Slave address		0x82	Exception code [2]	---		

[1] Length: If there is a remainder when the number of inputs is divided by 8, the number of bytes must be increased by 1.

[2] E code: 01 or 02 or 03 or 04

Function code 3 - This function allows individual registers to be read

Table 9-314 FC 3 - Read hold register

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 3	Start address		Number of registers	
Valid response	Slave address	Function code 3	Length [1]	Register data		
Error message	Slave address	0x83	Exception code [2]	---		

[1] Length: Number of bytes

[2] E code: 01 or 02 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 3	Start address		Number of registers	
Valid response	Slave address		Function code 3	Length [1]	Register data		
Error message	Slave address		0x83	Exception code [2]	---		

[1] Length: Number of bytes

[2] E code: 01 or 02 or 03 or 04

Function code 4 - This function allows individual registers to be read

Table 9-315 FC 4 - Read input words

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 4	Start address		Number of input words	
Valid response	Slave address	Function code 4	Length [1]	Input data		
Error message	Slave address	0x84	Exception code [2]	---		

[1] Length: 2 * number of input words

[2] E code: 01 or 02 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 4	Start address		Number of input words	
Valid response	Slave address		Function code 4	Length [1]	Input data		
Error message	Slave address		0x84	Exception code [2]	---		

[1] Length: 2 * number of input words

[2] E code: 01 or 02 or 03 or 04

Function code 5 - This function can set or delete individual bits

Table 9-316 FC 5 - Write an output bit

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 5	Start address		Value	
Valid response	Slave address	Function code 5	Length	Value		
Error message	Slave address	0x85	Exception code [1]	---		

[1] E code: 01 or 02 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 5	Start address		Value	
Valid response	Slave address		Function code 5	Length	Value		
Error message	Slave address		0x85	Exception code [1]	---		

[1] E code: 01 or 02 or 03 or 04

Function code 6 - This function allows individual registers to be written

Table 9-317 FC 6 - Write hold register

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 6	Address		Register	
Valid response	Slave address	Function code 6	Address		Register	
Error message	Slave address	0x86	Exception code [1]	---		

[1] E code: 01 or 02 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 6	Address		Register	
Valid response	Slave address		Function code 6	Address		Register	
Error message	Slave address		0x86	Exception code [1]	---		

[1] E code: 01 or 02 or 03 or 04

Function code 8 - This function is used to check the communication connection

Table 9-318 FC 8 - Slave status

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 8	Diagnostic code		Test value	
Valid response	Slave address	Function code 8	Diagnostic code		Test value	
Error message	Slave address	0x88	Exception code [1]	---		

[1] E code: 01 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 8	Diagnostic code		Test value	
Valid response	Slave address		Function code 8	Diagnostic code		Test value	
Error message	Slave address		0x88	Exception code [1]	---		

[1] E code: 01 or 03 or 04

Function code 11 - This function can read 2 bytes of "Status word" and 2 bytes of "Event counter"

Table 9-319 FC 11 - Event counter for slave communication

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Query	Slave address	Function code 11	---			
Valid response	Slave address	Function code 11	Status		Event counter	
Error message	Slave address	0x8B	Exception code [1]	---		

[1] E code: 01 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Query	Slave address		Function code 11	---			
Valid response	Slave address		Function code 11	Status		Event counter	
Error message	Slave address		0x8B	Exception code [1]	---		

[1] E code: 01 or 04

Function code 15 - This function allows multiple bits to be written

Table 9-320 FC 15 - Write one/multiple output bits

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte n
Query	Slave address	Function code 15	Start address		Number of output words		Byte counter [1]	Value	
Valid response	Slave address	Function code 15	Start address		Number of output words		---		
Error message	Slave address	0x8F	Exception code [2]	---					

[1] Byte counter: If there is a remainder when the number of bytes is divided by 8, the number of bytes must be increased by 1.

[2] E code: 01 or 02 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte n
Query	Slave address		Function code 15	Start address		Number of output words		Byte counter [1]	Value	
Valid response	Slave address		Function code 15	Start address		Number of output words		---		
Error message	Slave address		0x8F	Exception code [2]		---				

[1] Byte counter: If there is a remainder when the number of bytes is divided by 8, the number of bytes must be increased by 1.

[2] E code: 01 or 02 or 03 or 04

Function code 16 - This function allows individual or multiple registers to be written

Table 9-321 FC 16 - Write one/multiple hold registers

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte n
Query	Slave address	Function code 16	Start address		Number of registers		Byte counter [1]	Value	
Valid response	Slave address	Function code 16	Start address		Number of registers		---		
Error message	Slave address	0x90	Exception code [2]	---					

[1] Byte counter: Number of registers * 2

[2] E code: 01 or 02 or 03 or 04

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte n
Query	Slave address		Function code 16	Start address		Number of registers		Byte counter [1]	Value	
Valid response	Slave address		Function code 16	Start address		Number of registers		---		
Error message	Slave address		0x90	Exception code [2]		---				

[1] Byte counter: Number of registers * 2

[2] E code: 01 or 02 or 03 or 04

Error messages

Overview of error messages - PtP

The error messages are provided at the STATUS output of an instruction and can be evaluated there or processed in the user program.

Error code	Description	Remedy
16#0000	No error	-
RECEIVE status and error codes		

Error code	Description	Remedy
16#0094	frame end identified based on the "Receipt of fixed/ maximum frame length"	-
16#0095	frame end identified based on "Message timeout"	-
16#0096	frame end identified based on expiration of the "Character delay time"	-
16#0097	The frame was aborted because the maximum response time was reached.	-
16#0098	frame end identified based on the fulfillment of the "Read message length from message" conditions	-
16#0099	frame end identified based on the receipt of the "End sequence"	-
SEND status and error codes		
16#7000	Block idle	-
16#7001	Initial call for a new frame: Data transmission initiated	-
16#7002	Interim call: Data transmission running	-
16#8085	Invalid length	Select a suitable frame length. The following are valid (depending on the module): 1-1024/2048/4096 (Byte)
16#8088	The specified length exceeds the range set in the receive buffer. Note: If the data type STRING has been specified at the BUFFER parameter, this error code also appears if the current string is shorter than the length specified at the LENGTH parameter.	Change the range in the receive buffer or select a frame length which corresponds to the range set in the receive buffer. The following are valid (depending on the module): 1-1024/2048/4096 (Byte)
16#8090	Configuration error: Odd number of bytes for WString	Select an even number of bytes.
RECEIVE status and error codes		
16#8088	The number of characters received exceeds the number specified at the BUFFER parameter.	Select a suitable frame length. The following are valid (depending on the module): 1-1024/2048/4096 (Byte)
16#8090	Configuration error: Odd number of bytes for WString	Select an even number of bytes.
Error message codes of the special functions		
16#818F	Incorrect parameter number setting (with USS only)	Select a suitable parameter number (PARAM). The following numbers are valid: 0-2047
16#8190	Incorrect setting of the CRC calculation	Select a suitable value for the CRC calculation. The following are valid: deactivated or activated. Check whether the module addressed supports CRC calculation.
16#8191	Incorrect setting of the diagnostic error interrupt	Select a suitable value for "Diagnostic error interrupt". The following are valid: Diagnostic error interrupt deactivated or diagnostic error interrupt activated. Check whether the module addressed supports the generation of diagnostic interrupts.
Error message codes of the "Port configuration"		
16#81A0	The module does not support this protocol.	Select a valid protocol for the module (PROTOCOL).
16#81A1	The module does not support this data transmission rate.	Select a valid data transmission rate for the module (BAUD).

Error code	Description	Remedy
16#81A2	The module does not support this parity setting.	Select a suitable value for "Parity" (PARITY). The following are valid: • None (1) • Even (2) • Odd (3) • Mark (4) • Space (5) • Any (6)
16#81A3	The module does not support this number of data bits.	Select a suitable value for "Number of data bits" (DATABITS). The following are valid: • 7 (2) • 8 (1)
16#81A4	The module does not support this number of stop bits.	Select a suitable value for "Number of stop bits" (STOPBITS). The following are valid: • 1 (1) • 2 (2)
16#81A5	The module does not support this type of data flow control.	Select a valid data flow control for the module (FLOWCTRL).
16#81A7	Invalid value for XON or XOFF	Select suitable values for XON (XONCHAR) and XOFF (XOFFCHAR). Valid range of values: 0...255
16#81AA	Invalid operating mode	Valid operating modes are: • Full duplex (RS232) (0) • Full duplex (RS422) four-wire mode (point-to-point) (1) / (CM PtP (ET 200SP)) • Full duplex (RS422) four-wire mode (multipoint master) (2)/ (CM PtP (ET 200SP)) • Full duplex (RS 422) four-wire mode (multipoint slave) (3) • Half duplex (RS485) two-wire mode. (4)
16#81AB	Invalid receive line initial state	Valid initial states are: • "No" default setting (0) • Signal R(A)=5 V, signal R(B)=0 V (break detection) (1): Can only be selected with: "Full duplex (RS422) four-wire operation (point-to-point connection)" and "Full duplex (RS422) four-wire operation (multipoint slave)". • Signal R(A)=0 V, signal R(B)=5 V (2): This default setting corresponds to the idle state (no active send operation).
16#81AC	Invalid value for "Break detection"	Select a suitable value for "Break detection". The following are valid: • Break detection deactivated (0) • Break detection activated (1).

Error code	Description	Remedy
16#81AF	The module does not support this protocol.	Select a valid protocol for the module.
Error codes of the "Send configuration"		
16#81B5	More than two end delimiters or end sequence > 5 characters	Select suitable values for "End delimiter" and "End sequence". The following are valid: • deactivated (0), • 1 (1) or 2 (2) end delimiters or • deactivated (0), • 1 (1) up to 5 (5) characters for the end sequence.
16#81B6	Send configuration rejected because the 3964(R) protocol was selected	Make sure that no send configuration is transmitted if the 3964(R) protocol is set.
Error codes of the "Receive configuration"		
16#81C0	Invalid start condition	Select a suitable start condition. The following are valid: • Send break before frame start • Send Idle Line.
16#81C1	Invalid end condition or no end condition selected	Select a suitable end condition (see Receiving data with Freeport (Page 991)).
16#81C3	Invalid value for "Maximum message length"	Select a suitable value for "Maximum message length" (MAXLEN). Valid range of values (depending on the module): 1-1024/2048/4096 (Byte)
16#81C4	Invalid value for "Offset of the length specification in the message"	Select a suitable value for "Offset of the length specification in the message". Valid range of values (depending on the module): 1-1024/2048/4096 (Byte)
16#81C5	Invalid value for "Size of length field"	Select a suitable value for "Size of length field" (LENGTHSIZE). Valid range of values in bytes: • 1 (1) • 2 (2) • 4 (4)
16#81C6	Invalid value for "Number of characters not counted in length specification"	Select a suitable value for "Number of characters not counted in length specification" (LENGTHM). Valid range of values: 0 to 255 (bytes)
16#81C7	The total of "Offset in the message + size of length field + number of characters not counted" is greater than the maximum frame length	Select a suitable value for "Offset in message", "Size of length field" and "Number of characters not counted". Valid range of values: • Offset in the message (depending on the module): 0-1024/2048/4096 (Byte) • Size of length field: 1, 2, or 4 (bytes) • Number of characters not counted: 0 to 255 (bytes)

Error code	Description	Remedy
16#81C8	Invalid value for "Response timeout"	Select a suitable value for "Response timeout". Valid range of values: 1-65535 (ms)
16#81C9	Invalid value for "Character delay time"	Select a suitable value for "Character delay time". Valid range of values: 1 to 65535 (bit times)
16#81CB	frame end sequence is activated, but no character is activated for the check	Activate one or several characters for the check.
16#81CC	frame start sequence is activated, but no character is activated for the check	Activate one or several characters for the check.
16#81CD	Invalid value for "Prevent overwriting"	Select a suitable value for "Prevent overwriting". The following are valid: • Prevent overwriting is deactivated (0) or • Prevent overwriting is activated (1)
16#81CE	Invalid value for "Clear receive buffer on startup"	Select a suitable value for "Clear receive buffer on startup". The following are valid: • Clear receive buffer at startup is deactivated (0) • Clear receive buffer at startup is activated (1)
SEND status and error codes		
16#81D0	Receiving send request during runtime of a send command	Make sure that you do not receive an additional send request during runtime of a send command.
16#81D1	The waiting time for XON or CTS = ON has expired.	The communication partner has a fault, is too slow or is offline. Check the communication partner or change the parameters, if necessary.
16#81D2	"Hardware RTS always ON": Send job canceled due to change from DSR = ON to OFF	Check the communication partner. Make sure that DSR is ON for the entire duration of transmission.
16#81D3	Send buffer overflow / send frame too long	Select a shorter frame length. The following are valid (depending on the module): 1 to 1024/2048/4096 (bytes)
16#81D5	Transmission canceled due to parameter changes, detected wire break, or CPU in STOP	Check the parameter assignment, wire break, and CPU status.
16#81D6	Transmission canceled because end identifier was not received	Check the parameter assignment of the end characters and the frame of the communication partner.
16#81D7	Communication error between the user program and module	Check the communication (e.g., matching the sequence number).
16#81D8	Transmission attempt rejected because module is not configured	Configure the module.
Error codes of the receive configuration		
16#81E0	Frame aborted: Send buffer overflow / send frame too long	Increase the call rate for the receive function in the user program or configure communication with data flow control.
16#81E1	Frame aborted: Parity error	Check the connection line of the communication partners, or verify that both devices are configured for the same data transmission rate, parity and stop bit number.
16#81E2	Frame aborted: Character frame error	Check the settings for start bit, data bits, parity bit, data transmission rate, and stop bit(s).

Error code	Description	Remedy
16#81E3	Frame aborted: Character overflow error	Firmware error: Please contact Customer Support.
16#81E4	Frame aborted: The total length of "Offset in the message + size of the length field + number of characters not counted" is greater than the receive buffer	Select a suitable value for offset in message, size of the length field, and number of characters not counted.
16#81E5	Frame aborted: Break	Receive line to partner is interrupted. Reconnect or switch on partner.
16#81E6	Maximum number of "Buffered received frames" exceeded	In the user program call the instruction more often or configure a communication with data flow control or increase the number of buffered frames.
16#81E7	Synchronization error module and Receive_P2P	Make sure that different instances of the Receive_P2P do not access the same module.
16#81E8	Frame aborted: The character delay time has expired before the message end criterion was detected	Partner device faulty or too slow. Check this, if required, using an interface tester that is interconnected in the transmission line.
16#81E9	Modbus CRC error (only communication modules which support Modbus)	Checksum error of the Modbus frame. Check the communication partner.
16#81EA	Modbus frame too short (only communication modules which support Modbus)	Minimum length of Modbus frame not met. Check the communication partner.
16#81EB	Frame aborted: Maximum frame length reached	Select a shorter frame length at the communication partner. The following are valid (depending on the module): 1-1024/2048/4096 (Byte) Check the parameters for end of frame detection.
Error codes V24 accompanying signals		
16#81F0	The module does not support V24 accompanying signals	You have tried to set accompanying signals for a module that does not support V24 accompanying signals. Make sure that this is an RS232 module or that RS232 mode (ET 200SP) is set.
16#81F1	No operation of the V24 accompanying signals	The V24 accompanying signals cannot be operated manually if hardware data flow control is active.
Error codes of the receive configuration		
16#8201 [1)]	Receive_Conditions is a pointer to an invalid data type	Enter a pointer to one of the following data types: DB, BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TIME_OF_DAY, TIME, S5TIME, DATE_AND_TIME, STRING
16#8225	Receive_Conditionspoints to an optimized memory area greater than 1 kB or Receive_Conditions points to an optimized memory area and the receive length is greater than the area addressed by Receive_Conditions.	Enter a pointer to an area with a maximum length of: • Optimized memory area: 1 kByte • Non-optimized area: 4 kByte Note: If the pointer points to an optimized memory area, do not send more than 1 kByte.
16#8229 [1)]	Receive_Conditions is a pointer to BOOL with a number of bits not equal to n * 8	If you are using a pointer to BOOL, the number of bits must be a multiple of 8.
Error codes, general		

Error code	Description	Remedy
16#8280	Negative acknowledgment when reading module	You can find more detailed information on error causes in theRDREC.STATUS static parameters and in the description of the SFB RDREC. • Check the input at the PORT parameter • Set the COM_RST parameter prior to the initial call.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter You can find more detailed information on error causes in the WRREC.STATUS static parameters and in the description of the SFB WRREC.
16#8282	Module not available	Check the input at the PORT parameter and ensure that the module can be reached.
Error codes of the receive configuration		
16#82C1	Invalid value for "Buffered received frames".	Select a suitable value for "Buffered received frames". Valid range of values: 1-255
16#82C2	Receive configuration rejected because the 3964(R) protocol was selected	Make sure that no receive configuration is sent if the 3964(R) protocol is set.
16#8301 1)	Receive_Conditions is a pointer to an invalid data type	Select a valid data type. The following are valid: DB, BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TIME_OF_DAY, TIME, S5TIME, DATE_AND_TIME, STRING
16#8322	Range length error when reading a parameter	Check the input at the Receive_Conditions parameter
16#8324	Range error when reading a parameter	Check the input at the Receive_Conditions parameter
16#8328	Setting error when reading a parameter	Check the input at the Receive_Conditions parameter
SEND status and error codes		
16#8328 1)	BUFFER is a pointer to BOOL with a number of bits not equal to n * 8	If you are using a pointer to BOOL, the number of bits must be a multiple of 8.
Error codes of the receive configuration		
16#8332	Invalid data block at the Receive_Conditions parameter	Check the input at the Receive_Conditions parameter
16#833A	The designation of the data block at the Receive_Conditions parameter refers to a data block which is not loaded.	Check the input at the Receive_Conditions parameter
16#8351	Invalid data type	Check the input at the Receive_Conditions parameter
16#8352 1)	Receive_Conditions does not point to a data block	Check the pointer to Receive_Conditions
16#8353 1)	Receive_Conditions does not point to a structure of the type Receive_Conditions	Check the pointer to Receive_Conditions
Error codes 3964(R) protocol		
16#8380	Parameter assignment error: Invalid value for "Character delay time".	Select a suitable value for "Character delay time" (CharacterDelayTime). Valid range of values: 1 to 65535 (ms)
16#8381	Parameter assignment error: Invalid value for "Response timeout".	Select a suitable value for "Response timeout" (AcknDelayTime). Valid range of values: 1 to 65535 (ms)

Error code	Description	Remedy
16#8382	Parameter assignment error: Invalid value for "Priority".	Select a suitable value for "Priority" (Priority). The following are valid: • High (1) • Low (0)
16#8383	Parameter assignment error: Invalid value for "Block check"	Select a suitable value for "Block check" (BCC). The following are valid: • With block check (1) • Without block check (0)
16#8384	Parameter assignment error: Invalid value for "Connection attempts".	Select a suitable value for "Connection attempts" (BuildupAttempts). Valid range of values: 1-255
16#8385	Parameter assignment error: Invalid value for "Transmission attempts".	Select a suitable value for "Transmission attempts" (RepetitionAttempts). Valid range of values: 1-255
16#8386	Runtime error: Number of connection attempts exceeded	Check the interface cable and the transmission parameters. Also check whether the receive function is configured correctly at the partner device.
16#8387	Runtime error: Number of transmission attempts exceeded	Check the interface cable, the transmission parameters and the configuration of the communication partner.
16#8388	Runtime error: Error at the "Block check character" The internally calculated value of the block check character does not correspond to the block check character received by the partner at the connection end.	Check whether the connection is seriously disrupted; in this case you may also occasionally see error codes. Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#8389	Runtime error: Invalid character received while waiting for free receive buffer	The send request of the communication partner (STX, 02H) is only answered with DLE when the receive buffer is empty. No additional character may be received before (except STX again). Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#838A	Runtime error: Logical error during receiving. After DLE was received, a further random character (other than DLE or ETX) was received.	Check if partner DLE in frame header and in data string is always in duplicate or the connection is released with DLE ETX. Check for proper function at the partner device, possibly by using an interface test device that is switched into the transmission line.
16#838B	Runtime error: Character delay time exceeded	Partner device too slow or faulty. Verify by using an interface test device that is switched into the transmission line, if necessary.
16#838C	Runtime error: Wait time for free receive buffer has started	In the user program call the instruction more often or configure a communication with data flow control.
16#838D	Runtime error: frame repetition does not start within 4 s after NAK	Check the communication partner. A received frame that is possibly corrupted must be repeated by the partner within 4 seconds.

Error code	Description	Remedy
16#838E	Runtime error: In idle mode, one or several characters (other than NAK or STX) were received.	Check for proper function of the partner device, possibly using an interface test device that is switched into the transmission line.
16#838F	Runtime error: Initialization conflict - Both partners have set high priority	Set the "Low" priority at one of the partners
16#8391	Parameter assignment error: 3964 configuration data rejected because Freeport is set	If the Freeport protocol is set, make sure that no 3964 parameter assignment data is sent.

[1] Only with instructions for S7-300/400 CPUs

Overview of error messages - Modbus

Error code	Description	Remedy
16#0000	No error	-
Configuration error of the interface - Modbus_Comm_Load		
16#8181	The module does not support this data transmission rate.	Select a valid data transmission rate for the module at the BAUD parameter.
16#8182	The module does not support this parity setting.	Select a suitable value for "Parity" at the PARITY parameter. The following are valid: • None (1) • Even (2) • Odd (3) • Mark (4) • Space (5) • Any (6)
16#8183	The module does not support this type of data flow control.	Select a valid data flow control for the module at the FLOW_CTRL parameter.
16#8184	Invalid value for "Response timeout".	Select a suitable value for "Response timeout" at the RESP_TO parameter. Valid range of values: 1 to 65535 (ms)
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the Send_Config.RDREC.STATUS or Receive_Config.RDREC.STATUS static parameters or RDREC.STATUS and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the Send_Config.WRREC.STATUS or Receive_Config.WRREC.STATUS static parameters or WRREC.STATUS and in the description of the SFB WRREC.
16#8282	Module not available	Check the input at the PORT parameter and ensure that the module can be reached.
Configuration error - Modbus_Slave		

Error code	Description	Remedy
16#8186	Invalid slave address	Select a suitable slave address at the MB_ADDR parameter. The following are valid: 1-247 at standard address area; 1-65535 at extended address area (0 is reserved for Broadcast)
16#8187	Invalid value at MB_HOLD_REG parameter	Select a suitable value for the hold register at the MB_HOLD_REG parameter.
16#8188	Invalid operating mode or broadcast (MB_ADDR = 0) and MODE parameter ≠ 1	Select the value 1 for MODE in Broadcast mode or select a different operating mode.
16#818C	The pointer to a MB_HOLD_REG area must be a data block or a bit memory address area.	Select a suitable value for the pointer to the MB_HOLD_REG area.
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.WRREC.STATUS or Receive_P2P.WRREC.STATUS, and in the description of the SFB WRREC.
16#8452 [1)]	MB_HOLD_REG is not a pointer to a DB or a bit memory area	Check the MB_HOLD_REG pointer
16#8453 [1)]	MB_HOLD_REG is not a pointer of type BOOL or WORD	Check the MB_HOLD_REG pointer
16#8454 [1)]	The area addressed by MB_HOLD_REG is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the MB_HOLD_REG pointer
16#8455 [1)]	MB_HOLD_REG points to a write-protected DB	Check the MB_HOLD_REG pointer
16#8456 [1)]	Error during instruction execution. The cause of the error is shown in the STATUS parameter.	Determine the value of the SFCSTATUS parameter. Check what this means in the description for SFC51, STATUS parameter.
Configuration error - Modbus_Master		
16#8180	Invalid value for MB_DB parameter	The value configured for MB_DB (instance data DB) at the Modbus_Comm_Load instruction is not valid. Check the interconnection of the Modbus_Comm_Load instruction and its error messages.
16#8186	Invalid station address	Select a suitable station address at the MB_ADDR parameter. The following are valid: 1-247 at standard address area; 1-65535 at extended address area (0 is reserved for Broadcast)
16#8188	Invalid operating mode or broadcast (MB_ADDR = 0) and MODE parameter ≠ 1	Select the value 1 for MODE in Broadcast mode or select a different operating mode.

Error code	Description	Remedy
16#8189	Invalid data address	Select a suitable value for the data address at the DATA_ADDR parameter. See description Modbus_Master (Page 7536) in the Info system
16#818A	Invalid length	Select a suitable data length at the DATA_LEN parameter. See description Modbus_Master (Page 7536) in the Info system
16#818B	Invalid value for DATA_PTR	Select a suitable value for the data pointer at the DATA_PTR parameter (M or DB address). See description Modbus_Master (Page 7536) in the Info system
16#818C	Interconnection error of the DATA_PTR parameter	Check the interconnection of the instruction.
16#818D	The area addressed by DATA_PTR is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the DATA_PTR pointer
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Send_P2P.WRREC.STATUS or Receive_P2P.WRREC.STATUS or Receive_Reset and in the description of the SFB WRREC.
Communication errors - Modbus_Master and Modbus_Slave		
16#80 D1	The waiting time for XON or CTS = ON has expired.	The communication partner has a fault, is too slow or is offline. Check the communication partner or change the parameters, if necessary.
16#80D2	"Hardware RTS always ON": Send job canceled due to change from DSR = ON to OFF	Check the communication partner. Make sure that DSR is ON for the entire duration of transmission.
16#80E0	Frame aborted: Send buffer overflow / send frame too long	In the user program call the instruction more often or configure a communication with data flow control.
16#80E1	Frame aborted: Parity error	Check the connection line of the communication partners, or verify that both devices are configured for the same data transmission rate, parity and stop bit number.
16#80E2	Frame aborted: Character frame error	Check the settings for start bit, data bits, parity bit, data transmission rate, and stop bit(s).
16#80E3	Frame aborted: Character overflow error	Check the number of data in the frame of the communication partner.
16#80E4	Frame aborted: Maximum frame length reached	Select a shorter frame length at the communication partner. The following are valid (depending on the module): 1 to 1024/2048/4096 (bytes)

Error code	Description	Remedy
Communication error - Modbus_Master		
16#80C8	The slave does not respond within the set time	Check the data transmission rate, parity and wiring of the slave.
16#8200	The interface is busy with an ongoing request.	Repeat the command later. Make sure that there are no commands still running before you start a new one.
Protocol error - Modbus_Slave (only communication modules that support Modbus)		
16#8380	CRC error	Checksum error of the Modbus frame. Check the communication partner.
16#8381	The function code is not supported or is not supported for broadcast.	Check the communication partner and make sure that a valid function code is sent.
16#8382	Invalid length information in the request frame	Select a suitable data length at the DATA_LEN parameter.
16#8383	Invalid data address in the request frame	Select a suitable value for the data address at the DATA_ADDR parameter.
16#8384	Invalid data value error in the request frame	Check the data value in the request frame of the Modbus master
16#8385	The diagnostic value is not supported by the Modbus slave (function code 08)	The Modbus slave only supports the diagnostic values 16#0000 and 16#000A.
Protocol error - Modbus_Master (only communication modules that support Modbus)		
16#8380	CRC error	Checksum error of the Modbus frame. Check the communication partner.
16#8381	Response frame from Modbus Slave with the following error message: The function code is not supported.	Check the communication partner and make sure that a valid function code is sent.
16#8382	Response frame from Modbus Slave with the following error message: Invalid length	Select a suitable data length.
16#8383	Response frame from Modbus Slave with the following error message: Invalid data address in the request frame	Select a suitable value for the data address at the DATA_ADDR parameter.
16#8384	Response frame from Modbus Slave with the following error message: Data value error	Check the request frame to the Modbus slave.
16#8385	Response frame from Modbus Slave with the following error message: The diagnostic value is not supported by the Modbus slave	The Modbus slave only supports the diagnostic values 16#0000 and 16#000A.
16#8386	The returned function code does not match the requested function code.	Check the response frame and the addressing of the slave.
16#8387	A slave that was not requested answers	Check the response frame of the slave. Check the address settings of the slave.
16#8388	Error in the response of the slave to a write request.	Check the response frame of the slave.
16#8828 [1)]	DATA_PTR points to a bit address that is not equal to n * 8	Check the DATA_PTR pointer
16#8852 [1)]	DATA_PTR is not a pointer to a DB or a bit memory area	Check the DATA_PTR pointer
16#8853 [1)]	DATA_PTR is not a pointer of type BOOL or WORD	Check the DATA_PTR pointer
16#8855 [1)]	DATA_PTR points to a write-protected DB	Check the DATA_PTR pointer
16#8856 [1)]	Error during call of SFC51	Call the Modbus_Master instruction again
Error - Modbus_Slave (only communication modules that support Modbus)		

Error code	Description	Remedy
16#8428 [1]	MB_HOLD_REG points to a bit address that is not equal to n * 8	Check the MB_HOLD_REG pointer
16#8452 [1]	MB_HOLD_REG is not a pointer to a DB or a bit memory area	Check the MB_HOLD_REG pointer
16#8453 [1]	MB_HOLD_REG is not a pointer of type BOOL or WORD	Check the MB_HOLD_REG pointer
16#8454 [1]	The area addressed by MB_HOLD_REG is longer than the DB, or the area addressed is too small for the number of data bytes to be read or written.	Check the MB_HOLD_REG pointer
16#8455 [1]	MB_HOLD_REG points to a write-protected DB	Check the MB_HOLD_REG pointer
16#8456 [1]	Error during call of SFC51	Call the Modbus_Slave instruction again

[1] Only with instructions for S7-300/400 CPUs

Overview of error messages - USS

Error code	Description	Remedy
16#0000	No error	-
16#8180	Length error in response of drive	Check the response frame of the drive.
16#8181	Data type error	Select a suitable data type. The following are valid: • Real • Word • Double word
16#8182	Data type error: "Double word" or "Real" may not be returned for the "Word" request.	Check the response frame of the drive.
16#8183	Data type error: "Word" may not be returned for the "Double word" or "Real" request.	Check the response frame of the drive.
16#8184	Checksum error in response of drive	Check the drive and the communication connection.
16#8185	Addressing error	Valid drive address range: 1 to 16
16#8186	Setpoint error	Valid setpoint range: -200 % to +200 %
16#8187	Incorrect drive number returned	Check the response frame of the drive.
16#8188	Invalid PZD length	Permitted PZD lengths: 2, 4, 6, 8 words
16#8189	The module does not support this data transmission rate.	Select a valid data transmission rate for the module.
16#818A	A different request for this drive is currently active.	Repeat the parameter read or write command later.
16#818B	The drive does not respond.	Check the drive.
16#818C	The drive responds with an error message to a parameter request.	Check the response frame of the drive. Check the parameter request.
16#818D	The drive responds with an access error message to a parameter request.	Check the response frame of the drive. Check the parameter request.
16#818E	The drive was not initialized.	Check the user program and make sure that the USS_Drive_Control instruction is called for this drive.

Error code	Description	Remedy
16#8280	Negative acknowledgment when reading module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Port_Config.RDREC.STATUS, Send_Config.RDREC.STATUS , Receive_Config.RDREC.STATUS , Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB RDREC.
16#8281	Negative acknowledgment when writing module	Check the input at the PORT parameter. You can find more detailed information on error causes in the static parameters Port_Config.WRREC.STATUS, Send_Config.WRREC.STATUS , Receive_Config.WRREC.STATUS , Send_P2P.RDREC.STATUS or Receive_P2P.RDREC.STATUS, and in the description of the SFB WRREC.

[1] Only with instructions for S7-300/400 CPUs

PtP data link CP 340

P_SEND: Sending Data

Description

The P_SEND instruction transfers data blocks from a DB to CP 340 as specified at the DB_NO, DBB_NO and LEN parameters. For data transmission, the P_SEND instruction is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program.

Operating principle

Data transmission is initiated by a positive edge at the input REQ. Data transmission can take several calls (program cycles), depending on the data volume.

The P_SEND instruction can be called in the cycle by setting signal state "1" at the R parameter input. This setting cancels the transmission to CP 340 and resets the P_SEND instruction to the initial state. Data that has already been received by CP 340 is still sent to the communication partner. A static signal state "1" at the input R indicates that data transmission is disabled.

Parameter LADDR specifies the address of the CP 340 to be addressed.

Output DONE indicates "Job completed without errors". ERROR indicates error events. A corresponding event number is displayed in STATUS parameter (Page 7578) if an error occurs. If no error occurs, STATUS has a value 0. DONE and ERROR/STATUS are also output for RESET of the P_SEND instruction. The binary result BR is reset if an error occurs. If the block is completed without errors, the binary result has the status "1".

Parameters

Parameters	Declaration	Data type	Description
REQ	INPUT	BOOL	Job trigger at the positive edge
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Transmission is disabled.
LADDR	INPUT	INT	Start address of CP 340 The start address is taken from STEP 7.
DB_NO	INPUT	INT	Data block number Send DB no.: CPU-specific, zero not allowed
DBB_NO	INPUT	INT	Data byte number 0 ≤ DBB_NO ≤ 8190 send data, starting from data byte
LEN	INPUT	INT	Data length 1 ≤ LEN ≤ 1024, specified in number of bytes
DONE[1]	OUTPUT	BOOL	Job completed without errors STATUS parameter == 16#00;
ERROR[1]	OUTPUT	BOOL	Job canceled with errors STATUS parameter (Page 7578) contains the error information.
STATUS[1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 7578) contains the error information.
[1] The parameter is available until the next call of the instruction!			

Allocation in the data area

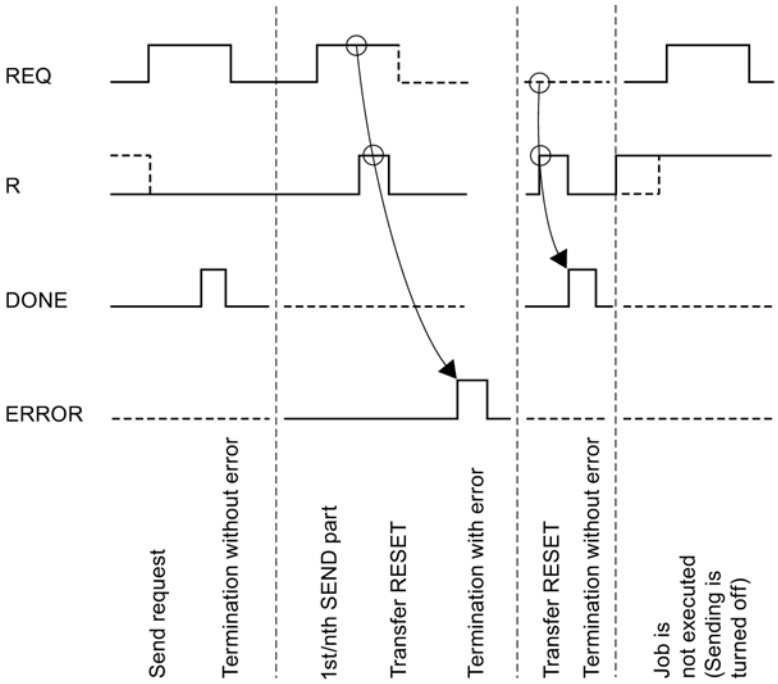
The P_SEND instruction works in combination with an I_SEND instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited

Note

Exception: If the error STATUS == W#16#1E0F occurs, you can examine the SFCERR or SFCSTATUS variable for additional details.

Time sequence chart

The figure below illustrates the characteristics of the DONE and ERROR parameters, depending on the input circuit of REQ and R.



Note

Input REQ is edge-triggered. A positive edge at the input REQ is sufficient. The RLO (result of the logic operation) does not necessarily have to be "1" for the entire duration of the transmission.

Rules

Note

The P_SEND instruction has no parameter check. If incorrect parameters are assigned, the CPU can change to STOP mode.

Note

The CP-CPU startup routine of the P_SEND instruction must be completed before the CP 340 can execute a job triggered after the CPU changed from STOP to RUN . A job initiated in the meantime is not lost. Once the startup coordination is completed, the job is sent to the CP 340 .

P_RCV: receiving data

Description

The P_RCV instruction transfers data from the CP 340 to an S7 data area as specified at the DB_NO, DBB_NO and LEN parameters. For data transmission, the P_RCV instruction is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program.

Operating principle

The (static) signal state "1" at parameter EN_R enables the check for data to be read from CP 340. An active transmission can be canceled by setting signal state "0" at parameter EN_R. The canceled receive request is terminated with an error message (STATUS output). Receiving is disabled as long as signal state "0" is set at parameter EN_R. Data transmission can take several calls (program cycles), depending on the data volume.

If the instruction detects signal state "1" at the R parameter, the current transmission job is canceled and the P_RCV instruction is reset to the initial state. Receiving is disabled as long as signal state "1" is set at parameter R. If the signal state returns to "0", the canceled message frame is received again from the beginning.

Parameter LADDR is used to select the CP 340 to be addressed.

Output NDR indicates "Job completed without errors/data received" (all data read) and ERROR indicates error events. STATUS (Page 7578) indicates the error event numbers. If no error occurs, STATUS has a value 0. NDR and ERROR/STATUS are also output by the RESET of P_RCV (parameter LEN == 16#00). The binary result BR is reset if an error occurs. If the block is completed without errors, the binary result has the status "1".

Parameters

Parameters	Declaration	Data type	Description
EN_R	INPUT	BOOL	Enable data reading
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Receiving is disabled.
LADDR	INPUT	INT	Start address of CP 340 The start address is taken from STEP 7.
DB_NO	INPUT	INT	Data block number Receive data block no.: CPU-specific, zero not allowed
DBB_NO	INPUT	INT	Data byte number 0 ≤ DBB_NO ≤ 8190 Receive data starting from data byte
NDR[1]	OUTPUT	BOOL	Job completed without errors, data received STATUS parameter == 16#00;

Parameters	Declaration	Data type	Description
ERROR[1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 7578) contains the error information.
LEN[1]	OUTPUT	INT	Length of the message frame received 1 ≤ LEN ≤ 1024, specified in number of bytes
STATUS[1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 7578) contains the error information.
[1] The parameter is available until the next call of the instruction!			

Allocation in the data area

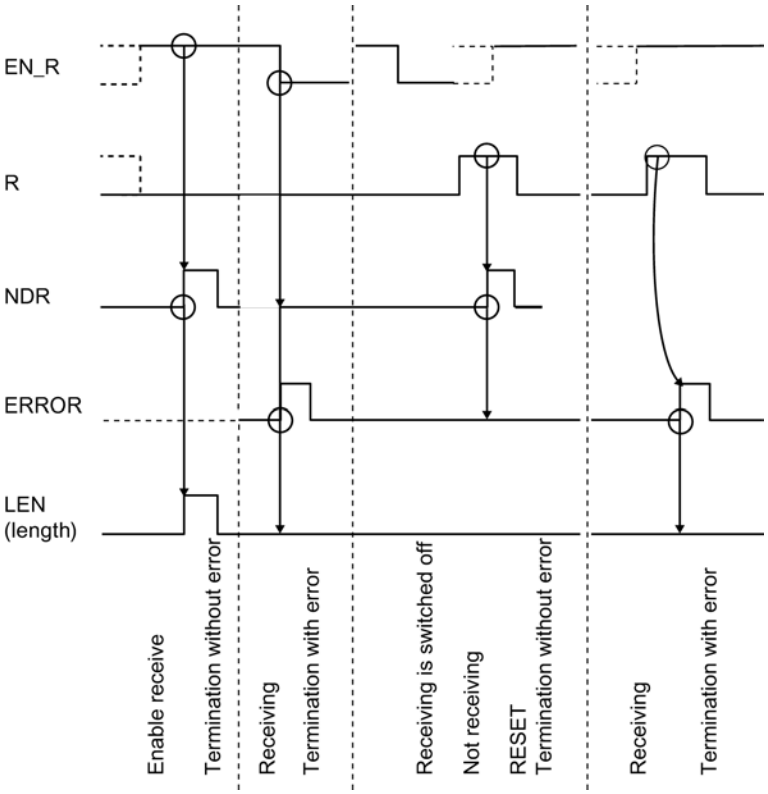
The P_RCV instruction works in combination with an I_RCV instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited

Note

Exception: If the error STATUS == W#16#1E0E occurs, you can examine the SFCERR or SFCSTATUS variable for additional details.

Time sequence chart

The figure below illustrates the characteristics of the NDR, LEN and ERROR parameters, depending on the input circuit of EN_R and R.



Note

Input EN_R must be set to the static signal state "1". Parameter EN_R must be provided the logic operation result "1" for the duration of the receive request.

Rules

Note

The P_RCV instruction has no parameter check. If incorrect parameters are assigned, the CPU can go into STOP mode.

Note

The CP-CPU startup routine of the P_RCV instruction must be completed before the CP 340 can execute a job triggered after the CPU changed from STOP to RUN.

P_PRINT: Print message text with up to 4 variables

Description

The P_PRINT instruction is available for printing message texts. The P_PRINT instruction transfers a process message to the CP 340, for example. CP 340 logs the process message to the connected printer.

The P_PRINT instruction sends up to four variables to the CP 340. For data transmission, the P_PRINT instruction is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program.

Operating principle

The DB_NO and DBB_NO parameters enable access to the pointers (to data blocks) for the format string and the four variables. The pointers must be stored without gaps and in a specific sequence in the parameterized data block. This is the pointer DB (see Figure "Pointer DB").

A positive edge at the input REQ initiates the transmission of the message text. The frame starts with the format string of the message text, This is followed by tags 1 to 4.

Data transmission can take several calls (program cycles), depending on the data volume.

The P_PRINT instruction can be called in the cycle by setting signal state "1" at parameter input R. This setting cancels the transmission to CP 340 and resets the P_PRINT instruction to the initial state. Data that has already been received by CP 340 is still sent to the communication partner. A static signal state "1" at the input R indicates that the transmission of print jobs is disabled.

Parameter LADDR specifies the address of the CP 340 to be addressed.

Output DONE indicates "Job completed without errors" and ERROR indicates error events. STATUS displays the corresponding event number (##link to the error code topic) if an error occurs. If no error occurs, STATUS has a value 0. DONE and ERROR/STATUS are also output for RESET of the P_PRINT instruction. The binary result BR is reset if an error occurs. If the block is completed without errors, the binary result has the status "1".

Parameters

Parameters	Declaration	Data type	Description
REQ	INPUT	BOOL	Job trigger at the positive edge
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Printing is disabled
LADDR	INPUT	INT	Start address of CP 340 The start address is taken from STEP 7.

Parameters	Declaration	Data type	Description
DB_NO	INPUT	INT	Data block number Pointer to pointer DB: CPU-specific, zero not allowed (The pointers to variables and format string are stored in the pointer DB in a fixed order (see previous figure).)
DBB_NO	INPUT	INT	Data byte number 0 ≤ DBB_NO ≤ 8162 Pointer as of data byte
DONE [1]	OUTPUT	BOOL	Job completed without errors STATUS parameter == 16#00;
ERROR [1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 7578) contains the error information.
STATUS [1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 7578) contains the error information.
[1] The parameter is available until the next call of the instruction!			

Assignment in the Data Area, Instance DB

The P_PRINT instruction works in combination with an I_PRINT instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited

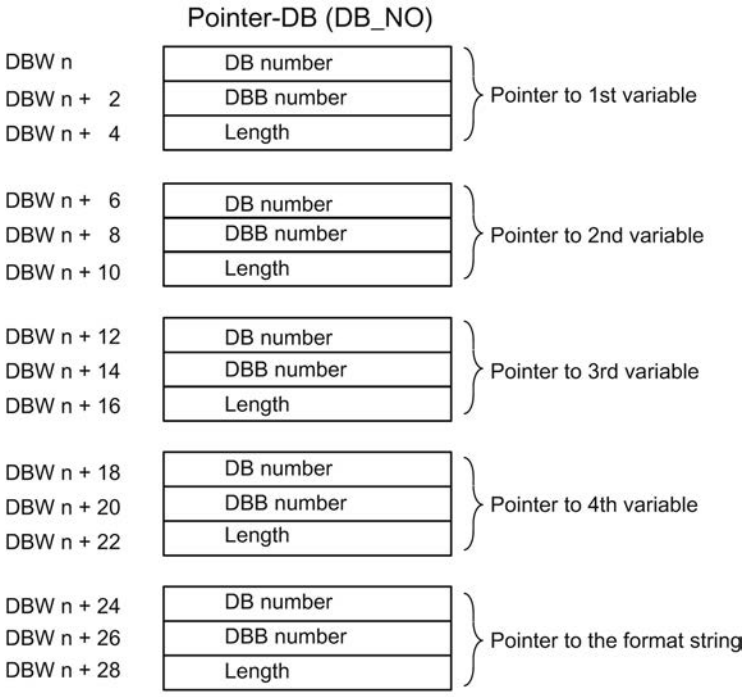
Note

Exception: If the error STATUS == W#16#1E0F occurs, you can examine the SFCERR or SFCSTATUS variable for additional details.

Assignment in the Data Area, Pointer DB

The P_PRINT instruction uses the DB_NO and DBB_NO parameters to access a pointer DB in which the pointers to the data blocks containing the message texts and variables are stored in a fixed sequence. You have to create the pointer DB.

The figure shows the syntax of the pointer DB that is addressed by means of the DB_NO and DBB_NO parameters of the P_PRINT instruction.



Permissible DB Number

The permissible DB numbers are CPU-specific. If the value 16#00 is specified as the DB number for "Pointer to variable", this variable is interpreted as not present and the pointer is set to the next variable or the format string.

If the DB number is equal to the value 16#00 in "Pointer to format string", the print job is canceled and event ID 16#1E43 is indicated at parameter output STATUS of the P_PRINT instruction.

Permissible DBB Number

The variable or format string begins at the parameterized DBB number. The variables can have a maximum length of 32 bytes and the format string can have a maximum length of 150 bytes.

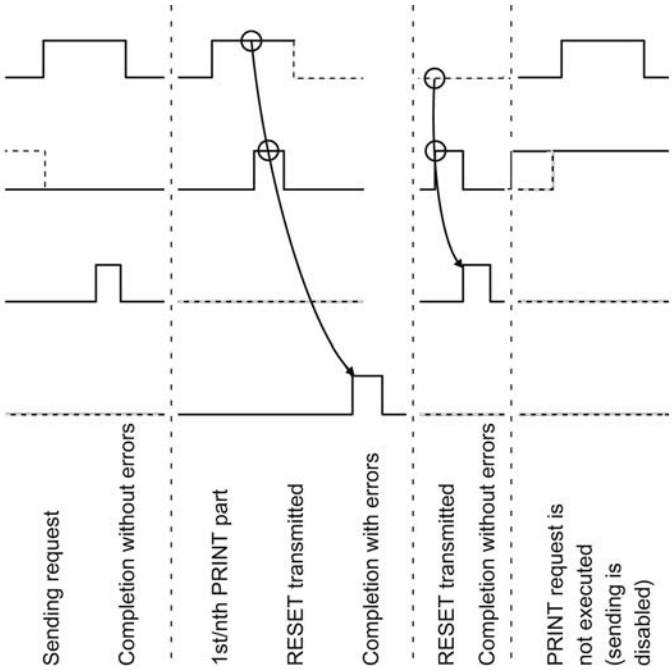
If the maximum length is exceeded, the print job is canceled and event ID 16#1E41 is indicated at parameter output STATUS of the P_PRINT instruction.

Permissible Length

The entry length in the pointer DB is to be set for each display type (data type) independently from the precision used.

Time sequence chart

The figure below illustrates the characteristics of the DONE and ERROR parameters, depending on the input circuit of REQ and R.



Note

Input REQ is edge-triggered. A positive edge at the input REQ is sufficient. It does not have to have a signal state of "1" during the entire transmission operation.

Rules

Note

The P_PRINT instruction has no parameter check. If incorrect parameters are assigned, the CPU can go into STOP mode.

Note

The CP-CPU startup routine of the P_PRINT instruction must be completed before the CP 340 can execute a job triggered after the CPU changed from STOP to RUN. A job initiated in the meantime is not lost. Once the startup coordination is completed, the job is sent to the CP 340.

See also

Data transmission with the printer driver (Page 1072)

P_RESET: Delete receive buffer

Description

The P_RESET instruction clears the entire input buffer of CP 340. All message frames in memory are discarded. A message frame received at the time the P_RESET instruction is called is saved.

A positive edge at the input REQ enables the instruction. The job can run over several calls (program cycles).

Parameter LADDR specifies the address of the CP 340 to be addressed.

The DONE output indicates "Job completed without errors". ERROR indicates error events. A corresponding event number is displayed in STATUS parameter (Page 7578) if an error occurs. If no errors occurs, STATUS has the value 0. If an error occurs, the binary result BR is reset. If the block is completed without errors, the binary result has the status "1".

Parameters

Parameters	Declaration	Data type	Description
REQ	INPUT	BOOL	Job trigger at the positive edge
LADDR	INPUT	INT	Start address of CP 340 The start address is taken from STEP 7.
DONE [1]	OUTPUT	BOOL	Job completed without errors STATUS parameter == 16#00;
ERROR [1]	OUTPUT	BOOL	Job canceled with errors STATUS parameter (Page 7578) contains the error information.
STATUS [1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 7578) contains the error information
[1] The parameter is available until the next call of the instruction!			

Allocation in the data area

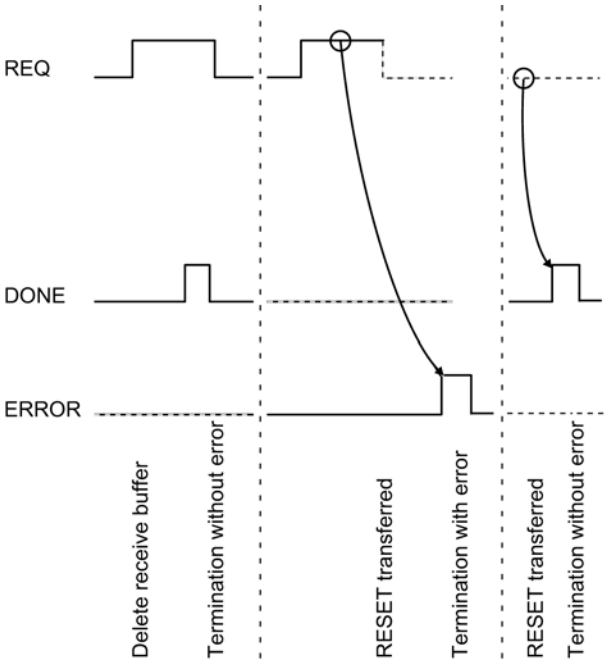
The P_RESET instruction works in combination with an I_P_RESET instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited

Note

Exception: If the error STATUS == W#16#1E0F occurs, you can examine the SFCERR or SFCSTATUS variable for additional details.

Time sequence chart

The figure below illustrates the characteristics of the DONE and ERROR parameters, depending on the input circuit of REQ.



Note

Input REQ is edge-triggered. A positive edge at the input REQ is sufficient. The RLO (result of the logic operation) does not necessarily have to be "1" for the entire duration of the transmission.

Rule

Note

The P_RESET instruction has no parameter check. If incorrect parameters are assigned, the CPU can go into STOP mode.

V24_STAT_340 / V24_STAT: Reading accompanying signals from the RS232C interface

Description

The V24_STAT_340 (CP 340) / V24_STAT (CP 341) instruction reads the RS 232C accompanying signals from the CP 34x and makes them available to you at the block

parameters. The V24_STAT_340 / V24_STAT instruction is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program.

The RS 232C accompanying signals are updated each time the instruction is called (cyclic polling). The CP 34x updates the I/O status at intervals of 20 ms. The inputs/outputs are constantly updated independently of this.

The binary result BR is not affected. The instruction does not output an error message.

Parameter LADDR is used to select the CP 34x be addressed.

Parameters

Parameters	Declaration	Data type	Description
LADDR	INPUT	INT	Start address of CP 34x The start address is taken from STEP 7.
DTR_OUT	OUTPUT	BOOL	**D**ata **t**erminal **r**eady, CP 34x ready (CP 34x output)
DSR_IN	OUTPUT	BOOL	**D**ata **s**et **r**eady, Communication partner ready (CP 34x input)
RTS_OUT	OUTPUT	BOOL	**R**equest **t**o **s**end, CP 34x ready to send (CP 34x output)
CTS_IN	OUTPUT	BOOL	**C**lear **t**o **s**end, Communication partner can receive data from CP 34x (response to RTS = ON of CP 34x) (CP 34x input)
DCD_IN	OUTPUT	BOOL	**D**ata **C**arrier **d**etect, Receive signal level (CP 34x input)
RI_IN	OUTPUT	BOOL	**R**ing **I**ndicator, Ring indicator (CP 34x input)

Allocation in the data area

The V24_STAT_340 / V24_STAT instruction does not occupy any data areas.

Rule

Note

A minimum pulse duration is necessary to detect a signal change. The CPU cycle time, the update time on CP 34x and the response time of the communication partner are decisive variables.

See also

RS232 mode (CP 34x, 44x, CPU 31xC-2 PtP + ET 200S 1SI) (Page 1048)

V24_SET_340 / V24_SET: Writing accompanying signals to the RS232C interface

Description

You can set or reset the interface outputs via the parameter inputs of instruction V24_SET_340 (CP 340) / V24_SET (CP 341). The V24_SET_340 / V24_SET instruction is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program.

The binary result BR is not affected. The instruction does not output an error message.

Parameter LADDR is used to select the CP 34x be addressed.

Parameters

Parameters	Declaration	Data type	Description
LADDR	INPUT	INT	Start address of CP 34x The start address is taken from STEP 7.
RTS	INPUT	BOOL	Request to send, CP 34x ready to send (Control CP 34x output)
DTR	INPUT	BOOL	Data terminal ready, CP 34x ready (Control CP 34x output)

Allocation in the data area

The V24_SET_340 / V24_SET instruction does not occupy any data areas.

See also

RS232 mode (CP 34x, 44x, CPU 31xC-2 PtP + ET 200S 1SI) (Page 1048)

STATUS parameter

Event classes

Event class	Meaning
5	Error executing a CPU job
7	Send errors
8	Receive errors
30 ($1E_H$)	Errors in communication between the CP and CPU

STATUS parameter

Error code (W#16#...)	Description	Remedy
0502	Job not permitted in this CP operating mode (e.g., device interface not parameterized)	Evaluate the diagnostic interrupt and rectify the error accordingly.
0505	**Only for printer drivers:** System data block with message texts not available on the CP	Use the parameter assignment software to configure the message texts and then carry out a warm restart.
0506	**Only for printer drivers:** Message text not available	Use the parameter assignment software to configure the message texts and then carry out a warm restart.
0507	**Only for printer drivers:** Message text too long	Edit the message text to reduce it to a length of fewer than 150 characters (or no more than 250 characters if it contains variables).
0508	**Only for printer drivers:** Too many conversion instructions	You have configured more conversion instructions than variables. Conversion instructions without associated variables will be ignored.
0509	**Only for printer drivers:** Too many variables	You have configured more variables than conversion instructions. Variables without associated conversion instructions will not be output.
050A	**Only for printer drivers:** Unknown conversion instruction	Check the conversion instruction. Undefined or unsupported conversion instructions are replaced in the printout with ******.
050B	**Only for printer drivers:** Unknown control instruction	Check the control instruction. Undefined or unsupported control instructions will be ignored. The control instruction will not be output as text either.

Error code (W#16#...)	Description	Remedy
050C	**Only for printer drivers:** Conversion instruction not executable	Check the conversion instruction. Conversion instructions that cannot be executed are output in the printout in accordance with the defined width and the valid remainder of the conversion instruction or the default display with * characters.
050D	**Only for printer drivers:** Width in conversion instruction too small or too great	Correct the specified width of the variable in the conversion instruction on the basis of the variable's maximum possible number of characters in text-based display types (A, C, D, S, T, Y, Z). Only as many characters as will fit in the specified width appear in the printout; the text is truncated to this width. In all other cases, * characters are output corresponding to the width.
050E	**Only for 3964(R) and ASCII drivers:** Invalid telegram length	Telegram length is > 1,024 bytes. The rest of the telegram (> 1,024 bytes) is still being received by the CP 340; this means that the first part of the telegram is discarded. Select a shorter telegram length.
051B	**Only for printer drivers:** Precision invalid	Correct the specified precision in the conversion instruction. The precision is always initialized with a dot prefix to identify and limit the width (for example, ".2" to output the decimal point and two decimals.) Precision is only relevant to display types A, D, F and R. It is ignored otherwise.
051C	**Only for printer drivers:** Variable invalid (variable length incorrect/incorrect type)	Correct the specified variable. The corresponding table indicates the possible data types for each display type.
051E	**Only for printer drivers:** The "line end sequences" sent with this job (i.e.: $R / $L / $N) do not fit (any longer) on the (initial) page.	Increase the length of your page, reduce the number of lines (or line feeds) or spread your printout over a number of pages.
0701	**Only for 3964(R):** Sending of the first repetition: • An error was detected when transmitting the telegram or • The partner requested a repetition by means of a negative acknowledgment character (NAK).	A repetition is not an error, but it can indicate that there is interference on the data link or the partner device has malfunctioned. If the message frame has not been transmitted after the maximum number of repetitions, an error number describing the first error that occurred is output.
0702	**Only for 3964(R):** Error establishing connection: After STX was sent, NAK or any other code (except for DLE or STX) was received.	Check for malfunction of the partner device; you may need to use an interface test device (FOXPG) interconnected in the data link.

Error code (W#16#...)	Description	Remedy
0703	**Only for 3964(R):** Acknowledgment delay time (QVZ) exceeded: After STX was sent, partner did not respond within the acknowledgment delay time.	The partner device is too slow or not ready to receive or there is a break in the transmission line, for example. Check for malfunction of the partner device; you may need to use an interface test device (FOXPG) interconnected in the data link.
0704	**Only for 3964(R):** Termination by partner: One or more codes were received from the partner during sending.	Check if the partner is also indicating errors, possibly because not all transmitted data arrived (e.g., break in the transmission line), fatal errors are pending or the partner device has malfunctioned. You can interconnect an interface tester (FOXPG) in the transmission line to check this state.
0705	**Only for 3964(R):** Negative acknowledgment when sending	Check if the partner is also indicating errors, possibly because not all transmitted data arrived (e.g., break in the transmission line), fatal errors are pending or the partner device has malfunctioned. You can interconnect an interface tester (FOXPG) in the transmission line to check this state.
0706	**Only for 3964(R):** End-of-connection error: • Partner rejected telegram at end of connection with NAK or a random string (except for DLE) or • Acknowledgment characters (DLE) received too early.	Check if the partner is also indicating errors, possibly because not all transmitted data arrived (e.g., break in the transmission line), fatal errors are pending or the partner device has malfunctioned. You can interconnect an interface tester (FOXPG) in the transmission line to check this state.
0707	**Only for 3964(R):** Acknowledgment delay time exceeded at end of connection or response monitoring time exceeded after a send telegram: After connection termination with DLE ETX, no response received from partner within acknowledgment delay time.	Partner device too slow or faulty. You can interconnect an interface tester (FOXPG) in the transmission line to check this state.
0708	**Only for ASCII drivers and printer drivers:** The wait time for XON or CTS = ON has expired.	The communication partner is faulty, too slow or has been taken offline. Check the communication partner; you may need to change the parameter assignment.
070B	**Only for 3964(R):** Initialization conflict cannot be resolved because both partners have high priority.	Change the parameter assignment.
070C	**Only for 3964(R):** Initialization conflict cannot be resolved because both partners have low priority.	Change the parameter assignment.

Error code (W#16#...)	Description	Remedy
0801	**Only for 3964(R):** Expectation of the first repetition: An error was detected on receiving a telegram and the CP requested repetition from the partner via a negative acknowledgment (NAK).	A repetition is not an error, but it can indicate that there is interference on the data link or the partner device has malfunctioned. If the message frame has not been transmitted after the maximum number of repetitions, an error number describing the first error that occurred is output.
0802	**Only for 3964(R):** Error establishing connection: • In idle mode, one or more random codes (other than NAK or STX) were received or • After an STX was received, the partner sent more characters without waiting for the response DLE. After partner power ON: • While partner is being switched on, the CP receives an undefined character.	Check for malfunction of the partner device; you may need to use an interface test device (FOXPG) interconnected in the data link.
0805	**Only for 3964(R):** Logical error while receiving: After DLE was received, a further random code (other than DLE or ETX) was received.	Check if the partner always duplicates the DLE in the telegram header and data string or the connection is terminated with DLE ETX. Check for malfunction of the partner device; you may need to use an interface test device (FOXPG) interconnected in the data link.
0806	**Character delay time (ZVZ) exceeded:** • Two successive characters were not received within character delay time or **Only for 3964(R):** • 1. character after sending of DLE while establishing connection was not received within the character delay time.	Partner device too slow or faulty. Check this using an interface test device (FOXPG) interconnected in the data link.
0807	**Illegal telegram length:** A zero-length telegram has been received.	Receipt of a zero-length telegram is not an error. Check why the communication partner is sending telegrams without user data.
0808	**Only for 3964(R):** Error in block check character (BCC): The value of BCC calculated internally does not match the BCC received by the partner when the connection was terminated.	Check if the connection is seriously disrupted; in this case you may also occasionally see error codes. Check for malfunction of the partner device; you may need to use an interface test device (FOXPG) interconnected in the data link.
0809	**Only for 3964(R):** The number of repetitions must the set to the same value.	Assign parameters to the same block wait time at communications partner as at CP 340. Check for malfunction of the communication partner; you may need to use an interface test device (FOXPG) interconnected in the data link.

Error code (W#16#...)	Description	Remedy
080A	There is no free receive buffer available: No receive buffer space available for receiving data.	Increase the call rate for the P_RCV instruction.
080C	Transmission error: • A transmission error (parity error, stop bit error, overflow error) was detected. Only for 3964(R): • If this happens during send or receive operations, repetition is started. • If a corrupted character is received in idle mode, the error is reported immediately so that disturbances on the transmission line can be detected early. • If the SF LED (red) and the RxD LED (green) light up, there is a break in the connecting cable (cable break) between the two communication partners.	Disturbances on the data link cause telegram repetitions, thus lowering user data throughput. The risk of undetected error increases. Change your system setup or cable wiring. Check the connecting cables of the communication partners or verify that both devices have matching settings for baud rate, parity and number of stop bits.
080D	BREAK: Break in receive line to partner.	Reconnect or switch on partner.
0810	Only for ASCII drivers: Parity error: If the SF LED (red) and the RxD LED (green) light up, there is a break in the connecting cable (cable break) between the two communication partners.	Check the connecting cables of the communication partners or verify that both devices have matching settings for baud rate, parity and number of stop bits. Change your system setup or cable wiring.
0811	Only for ASCII drivers: Character frame error: If the SF LED (red) and the RxD LED (green) light up, there is a break in the connecting cable (cable break) between the two communication partners.	Check the connecting cables of the communication partners or verify that both devices have matching settings for baud rate, parity and number of stop bits. Change your system setup or cable wiring.
0812	Only for ASCII drivers: More characters were received after the CP had sent XOFF or set CTS to OFF.	Reset the parameters for the communication partner or read data from CP more quickly.
0818	Only for ASCII drivers: DSR = OFF or CTS = OFF	The partner has switched the DSR or CTS signal to "OFF" before or during a transmission. Check the partner's control of the RS 232C accompanying signals.
1E0D	Job aborted due to warm restart, hot restart or reset	

Error code (W#16#...)	Description	Remedy
1E0E	Static error when calling RD_REC SFC or RDREC SFB. The RET_VAL return value for the SFC/SFB is made available to you for evaluation in the SFCERR or SFCSTATUS variable respectively on the instance DB.	Load the SFCERR or SFCSTATUS variable from the instance DB.
1E0F	Static error when calling WR_REC SFC or RDREC SFB. The RET_VAL return value for the SFC/SFB is made available to you for evaluation in the SFCERR or SFCSTATUS variable respectively on the instance DB.	Load the SFCERR or SFCSTATUS variable from the instance DB.
1E41	The number of bytes specified at the FBs' LEN parameter is not permissible.	You must stay within a range of values of 1 to 1,024 bytes.
1E41	P_PRINT instruction: The number of bytes specified for the variable or format string in the pointer DB under length is not permissible.	You must specify a permissible length: 32 bytes for variables, 150 bytes for format strings.
1E43	P_PRINT instruction: No pointer available for format string.	Enter the data block no. and data word no. for the format string in the pointer DB.

Note

An error message is only output if the ERROR bit (job canceled with error) is also set. In all other cases the STATUS word is zero.

Calling the SFCERR or SFCSTATUS variable

You can call the SFCERR or SFCSTATUS variable to obtain more detailed information about the pending event class 30 error, 14 (1E0EH) or 15 (1E0FH).

You can load the SFCERR or SFCSTATUS variable only by means of symbolic access to instance DB of the corresponding instruction.

Error messages entered in the SFCERR variable can be found in the system functions SFC 58 "WR_REC: Write data record to I/O (Page 6496)" and SFC 59 "RD_REC: Read data record from I/O (Page 6491)".

Error messages entered in the SFCSTATUS variable can be found in the system functions SFB 52 "RDREC: Read data record (Page 6459)" and SFB 53 "WRREC: Write data record (Page 6461)".

PtP data link CP 341

P_SND_RK: Fetching or sending data

P_SND_RK

The P_SND_RK instruction distinguishes the following applications:

- P_SND_RK: Send data with 3964(R) and ASCII drivers (Page 7584)
- P_SND_RK: Send data with RK 512 (Page 7587)
- P_SND_RK: Fetch data with RK 512 (Page 7591)

See also

Data transmission from CPU to communication module with P_SND_RK (CP 341) (Page 1092)

P_SND_RK: Send data with 3964(R) and ASCII drivers

Description

The P_SND_RK instruction transfers data blocks from a DB to the CP 341 as specified at the DB_NO, DBB_NO and LEN parameters. For data transmission, the P_SND_RK instruction is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program.

Operating principle

Data transmission is initiated by a positive edge at the input REQ. Data transmission can take several calls (program cycles), depending on the data volume.

The P_SND_RK instruction can be called in the cycle by setting signal state "1" at the R parameter input. This setting cancels the transmission to CP 341 and resets the P_SND_RK instruction to the initial state. Data that has already been received by CP 341 is still sent to the communication partner. If the signal state remains static at "1" at the input R, sending is disabled.

Parameter LADDR specifies the address of the CP 341 to be addressed.

Output DONE indicates "Job completed without errors" and ERROR indicates error events. A corresponding event number is displayed in STATUS if an error occurs. If no error occurs, the STATUS has a value 0. DONE and ERROR/STATUS are also output for a RESET of the P_SND_RK instruction. The binary result BR is reset if an error occurs. If the block is completed without errors, the binary result has the status "1".

Parameters

Parameters	Declaration	Data type	Description
REQ	INPUT	BOOL	Job trigger at the positive edge
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Transmission is disabled.
LADDR	INPUT	INT	Start address of CP 341 The start address is taken from STEP 7.
DB_NO	INPUT	INT	Data block number Send DB no.: CPU-specific, zero is not allowed
DBB_NO	INPUT	INT	Data byte number 0 ≤ DBB_NO ≤ 8190 Send data starting from data byte
LEN	INPUT	INT	Data length 1 ≤ LEN ≤ 4096, specified in number of bytes
DONE[1]	OUTPUT	BOOL	Job completed without errors STATUS parameter == 16#00;
ERROR[1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 7612) contains the error information.
STATUS[1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 7612) contains the error information.
[1] The parameter is available until the next call of the instruction!			

Note

The R_CPU_NO, R_TYP, R_NO, R_OFFSET, R_CF_BYT and R_CF_BIT parameters are irrelevant for procedure 3964(R) and the ASCII driver and require no data. Similarly, the SF parameter requires no data since 'S' for Send is set by default.

Allocation in the data area

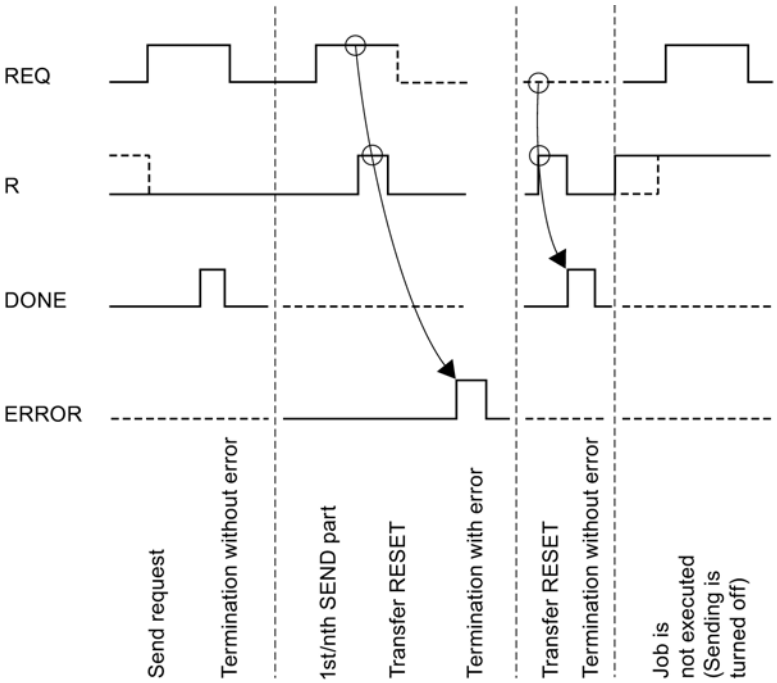
The P_SND_RK instruction works in combination with an I_SND_RK instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited

Note

Exception: If the error STATUS == W#16#1E0F occurs, you can examine the SFCERR or SFCSTATUS variable for additional details. This error variable must be loaded via symbolic access to the instance DB.

Time sequence chart

The figure below illustrates the characteristics of the DONE and ERROR parameters, depending on the input circuit of REQ and R.



Note

Input REQ is edge-triggered. A positive edge at the input REQ is sufficient. The RLO (result of the logic operation) does not necessarily have to be "1" for the entire duration of the transmission.

Rules

Note

The P_SND_RK instruction does not have a parameter check. If the parameter assignment is incorrect, the CPU can go to STOP mode.

Note

The CP-CPU startup routine of the P_SND_RK instruction must be completed before the CP 341 can execute a job triggered after the CPU changed from STOP to RUN . A job initiated in the meantime is not lost. Once the startup coordination is completed, the job is sent to the CP 341.

P_SND_RK: Send data with RK 512

Description

The P_SND_RK instruction can be used with parameter setting SF = 'S' to send data from an S7 data area to a CP 341.

Operating principle

Data transmission is initiated by a positive edge at the input REQ. Data transmission can take several calls (program cycles), depending on the data volume (LEN).

Parameter LADDR specifies the address of the CP 341 to be addressed.

The area of the data blocks is the only permissible source for data to be sent. The source is appropriately specified by setting the DB number (DB_NO) and offset (DBB_NO) of the first data byte to be transmitted from this data block.

Data blocks (DB) and extended data blocks (DX) are valid destination data types (R_TYP). The destination is appropriately specified by means of the CPU number (R_CPU_NO, relevant only to multi-processor communication), the data type (R_TYP: DB or DX), the DB number (R_NO) and the offset (R_OFFSET) to which the first byte is to be written.

The interprocessor communication flag byte and bit on the partner CPU are specified in R_CF_BYT and R_CF_BIT.

The P_SND_RK instruction can be called in the cycle by setting signal state "1" at the R parameter input. This setting cancels the transmission to CP 341 and resets the P_SND_RK instruction to the initial state. Data that has already been received by CP 341 is still sent to the communication partner. A static signal state "1" at the input R indicates that data transmission is disabled.

Output DONE indicates "Job completed without errors" and ERROR indicates error events. A corresponding event number is displayed in STATUS if an error occurs. If no error occurs, the STATUS has a value 0. DONE and ERROR/STATUS are also output for a RESET of the P_SND_RK instruction. The binary result BR is reset if an error occurs. If the block is completed without errors, the binary result has the status "1".

Parameters

Parameters	Declaration	Data type	Description
SF	INPUT	CHAR	Selection for Send data or Fetch data SF = 'S' (Send) Default: 'S'
REQ	INPUT	BOOL	Job trigger at the positive edge
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Transmission is disabled. Default: 0
LADDR	INPUT	INT	Start address of CP 341 The start address is taken from STEP 7.
DB_NO	INPUT	INT	Data block number of source Send DB no.: CPU-specific, zero not allowed
DBB_NO	INPUT	INT	Data byte number of source 0 ≤ DBB_NO ≤ 8190 Send data starting from data byte
LEN	INPUT	INT	Data length of the message frame to be sent 1 ≤ LEN ≤ 4096, specified in number of bytes; only even-numbered values are appropriate
R_CPU_NO	INPUT	INT	CPU number of the partner CPU 0 ≤ R_CPU_NO ≤ 4, only for multiprocessor mode. Default: 1
R_TYP	INPUT	CHAR	Address type on the partner CPU 'D': Data block ‘X’: Expanded data block
R_NO	INPUT	INT	Data block number on the partner CPU 0 ≤ R_NO ≤ 255
R_OFFSET	INPUT	INT	Data byte number on the partner CPU 0 ≤ R_OFFSET ≤ 510, even-numbered values only
R_CF_BYT	INPUT	INT	Interprocessor communication flag byte on partner CPU 0 ≤ R_CF_BYTE ≤ 255 Default: 255 (means: without interprocessor communication flag)
R_CF_BIT	INPUT	INT	Interprocessor communication flag bit on the partner CPU 0 ≤ R_CF_BIT ≤ 7
DONE[1]	OUTPUT	BOOL	Job completed without errors STATUS parameter == 16#00;
ERROR[1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 7612) contains the error information.
STATUS[1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 7612) contains the error information.
[1] The parameter is available until the next call of the instruction!			

Information in the message frame header

The table below shows the information in the header of the RK 512 message frame.

Table 9-322 Information in the RK 512 message frame header for "Send data" request

Source on your S7 automation system (local CPU)	To the destination, partner CPU	Message frame header, bytes		
		3/4 command type	5/6 Z-DBNR/Z offset	7/8 Number in
Data block	Data block	AD	DB/DW	Words
Data block	Expanded data block	AD	DB/DW	Words
D-DBNO: Destination data block number D-Offset: Destination start address DW: Offset in words				

Allocation in the data area

The P_SND_RK instruction works in combination with an I_SND_RK instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited

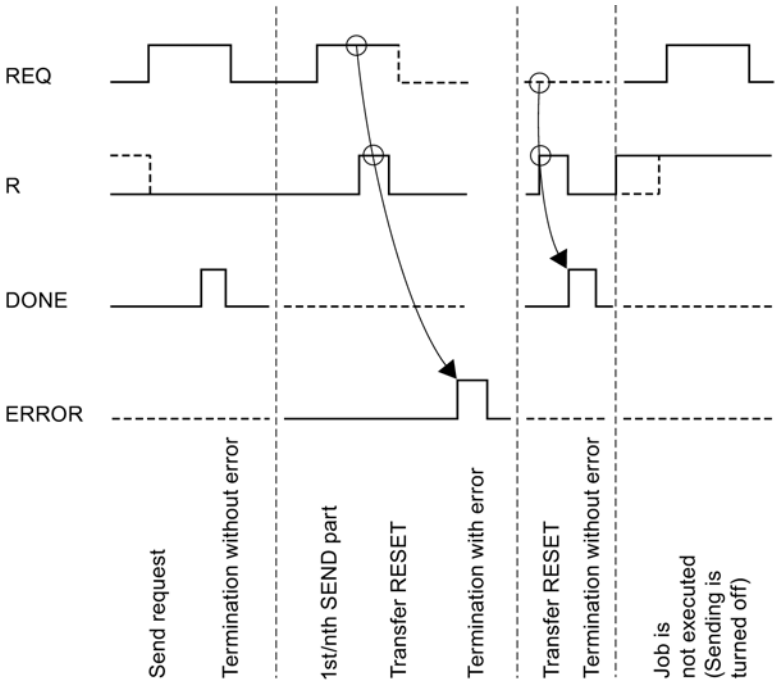
Special features for sending data

Note the following special features with regard to sending data:

- RK 512 allows only an even-numbered amount of data to be sent. If you specify an odd-numbered amount of data for the length (LEN), an additional filler byte with a value of "0" is sent at the end of the data.
- RK 512 allows only an even-numbered offset. If you specify an odd-numbered offset, the data is stored in the partner starting from the next lower even-numbered offset.
  Example: Offset is 7, data is stored from byte 6.

Time sequence chart

The figure below illustrates the characteristics of the DONE and ERROR parameters, depending on the input circuit of REQ and R.



Note

Input REQ is edge-triggered. A positive edge at the input REQ is sufficient. The RLO (result of the logic operation) does not necessarily have to be "1" for the entire duration of the transmission.

Rule

Note

The P_SND_RK instruction has no parameter check. If incorrect parameters are assigned, the CPU can go into STOP mode.

P_SND_RK: Fetch data with RK 512

Description

The P_SND_RK instruction can be used with parameter setting SF = F to fetch data from a remote communication partner and save it to an S7 data area of your automation system.

Note

To fetch data from a CP 341, you must always program a P_RCV_RK instruction on the CP 341.

Operating principle

Data transmission is initiated by a positive edge at the input REQ. Data transmission can take several calls (program cycles), depending on the data volume (LEN).

Parameter LADDR specifies the address of the CP 341 to be addressed.

The communication partner from which the data is fetched is specified by the CPU number (R_CPU_NO, relevant only to multiprocessor communication). The following data types (R_TYP) are valid sources for the data to be fetched: data blocks, extended data blocks, bit memory, inputs, outputs, counters and timers. The source is fully specified by the data type (R_TYP), the data block number (R_NO, relevant only to data blocks and expanded data blocks) and the offset(R_OFFSET) of the first data byte in this area to transmit.

The interprocessor communication flag byte and bit on the partner CPU are specified in R_CF_BYT and R_CF_BIT.

The only permissible destination areas are data blocks (DB). The destination is specified by the data block number (DB_NO) and offset (DBB_NO) to which the first byte is written.

The P_SND_RK instruction can be called in the cycle by setting signal state "1" at the R parameter input. This setting cancels the transmission from CP 341 and resets the P_SND_RK instruction to the initial state. A static signal status "1" at the input R indicates that fetching is disabled.

Output DONE indicates "Job completed without errors" and ERROR indicates error events. A corresponding event number is displayed in STATUS if an error occurs. If no error occurs, the STATUS has a value 0. DONE and ERROR/STATUS are also output for a RESET of the P_SND_RK instruction. The binary result BR is reset if an error occurs. If the block is completed without errors, the binary result has the status "1".

Parameters

Parameters	Declaration	Data type	Description
SF	INPUT	CHAR	Selection for Send data or Fetch data SF = 'F' (Fetch)default value: 'S' (Send)
REQ	INPUT	BOOL	Job trigger at the positive edge

Parameters	Declaration	Data type	Description
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Fetching is disabled. Default: 0
LADDR	INPUT	INT	Start address of CP 341 The start address is taken from STEP 7.
DB_NO	INPUT	INT	Data block number of destination Send DB no.: CPU-specific, zero not allowed
DBB_NO	INPUT	INT	Data byte number of destination 0 ≤ DBB_NO ≤ 8190 Send data starting from data byte
LEN	INPUT	INT	Data length of the message frame to be fetched 1 ≤ LEN ≤ 4096, specified in number of bytes[1]
R_CPU_NO	INPUT	INT	CPU number of the partner CPU 0 ≤ R_CPU_NO ≤ 4, only for multiprocessor mode. Default: 1
R_TYP	INPUT	CHAR	Address type on the partner CPU 'D': Data block 'X': Expanded data block 'M': Memory bit 'I': Inputs 'O': Outputs 'C': Counters 'T': Timers
R_NO	INPUT	INT	Data block number on the partner CPU 0 ≤ R_NO ≤ 255
R_OFFSET	INPUT	INT	Data byte number on the partner CPU
R_CF_BYT	INPUT	INT	Interprocessor communication flag byte on partner CPU 0 ≤ CF_BYTE ≤ 255 Default: 255 (means: without interprocessor communication flag)
R_CF_BIT	INPUT	INT	Interprocessor communication flag bit on the partner CPU 0 ≤ CF_BIT ≤ 7
DONE[1]	OUTPUT	BOOL	Job completed without errors STATUS parameter == 16#00;
ERROR[1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 7612) contains the error information.
STATUS[1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 7612) contains the error information.
[1] The parameter is available until the next call of the instruction!			

R_TYP, R_NO and R_OFFSET parameters

Source on the partner CPU	R_TYP	R_NO	R_OFFSET (in bytes) (This value is specified by the partner CPU.)
Data block	'D'	0 - 255	0 - 510 (only even-numbered values are appropriate!)
Expanded data block	'X'	0 - 255	0 - 510 (only even-numbered values are appropriate!)
Bit memory	'M'	irrelevant	0 - 255
Inputs	'I'	irrelevant	0 - 255
Outputs	'O'	irrelevant	0 - 255
Counters	'C'	irrelevant	0 - 255
Timers	'T'	irrelevant	0 - 255

Information in the message frame header

The table below shows the information in the header of the RK 512 message frame.

Source on the partner CPU	To the destination, your S7 automation system (local CPU)	Message frame header, bytes		
		3/4 command type	5/6 Q-DBNR/Q offset	7/8 Number in
Data block	Data block	ED	DB/DW	Words
Expanded data block	Data block	EX	DB/DW	Words
Bit memory	Data block	EM	Byte address	Bytes
Inputs	Data block	EI	Byte address	Bytes
Outputs	Data block	EO	Byte address	Bytes
Counters	Data block	EC	Counter no.	Words
Timers	Data block	ET	Timer number	Words
S-DBNO: Source data block number S-Offset: Source start address				

Allocation in the data area

The P_SND_RK instruction works in combination with an I_SND_RK instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited

Special features for (expanded) data blocks

Note the following special features with regard to fetching data from data blocks and expanded data blocks:

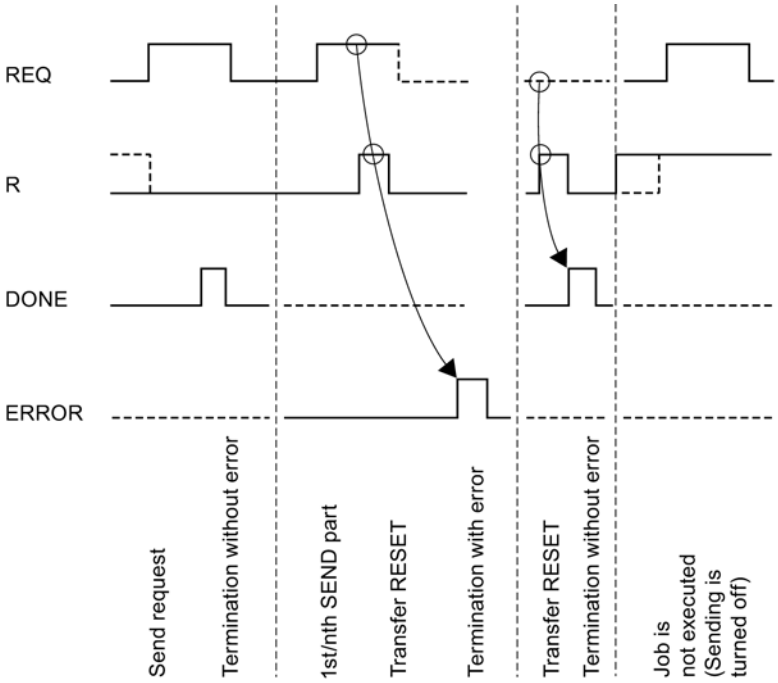
- RK 512 allows only an even-numbered amount of data to be fetched. If you specify an odd-numbered amount for the length (LEN), an extra byte is always sent. In the destination DB, however, the correct amount of data is entered.
- RK 512 allows only an even-numbered offset. If you specify an odd-numbered offset, the data is fetched from the partner starting from the next lower even-numbered offset. Example: Offset is 7, data is fetched as of byte 6.

Special features for timers and counters

If you fetch timers or counters from the communication partner, remember that 2 bytes are fetched for each timer or counter. For example, if you want to fetch 10 counters, you must enter 20 as the length.

Time sequence chart

The figure below illustrates the characteristics of the DONE and ERRORparameters, depending on the input circuit of REQ and R.



Note

Input REQ is edge-triggered. A positive edge at the input REQ is sufficient. The RLO (result of the logic operation) does not necessarily have to be "1" for the entire duration of the transmission.

Rule

Note

The P_SND_RK instruction has no parameter check. If incorrect parameters are assigned, the CPU can go into STOP mode.

P_RCV_RK: Receive data or Provide data

P_RCV_RK

The P_RCV_RK instruction distinguishes the following applications:

- P_RCV_RK: Receiving data with 3964(R) or ASCII driver (Page 7595)
- P_RCV_RK: Receive data with RK 512 (Page 7599)
- P_RCV_RK: Provide data with RK 512 (Page 7602)

See also

Data transmission from the communication module to the CPU with P_RCV_RK (CP 341) (Page 1094)

P_RCV_RK: Receiving data with 3964(R) or ASCII driver

Description

The P_RCV_RK instruction transfers data from CP 341 to an S7 data area as specified at the DB_NO, DBB_NO and LEN parameters. For data transmission, the P_RCV_RK instruction is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program.

Operating principle

The (static) signal state "1" at parameter EN_R enables the check for data to be read from CP 341 . An active transmission can be canceled by setting signal state "0" at parameter EN_R . The canceled receive request is terminated with an error message (STATUS output). Receiving is disabled as long as signal state "0" is set at parameter EN_R . Data transmission can take several calls (program cycles), depending on the data volume.

If the instruction detects signal state "1" at parameter R, the current transmission job is canceled and the P_RCV_RK instruction is reset to the initial state. Receiving is disabled as long as the signal state at the R parameter is "1".

Parameter LADDR is used to select the CP 341 to be addressed.

Output NDR indicates "Job completed without errors/data received" (all data read) and ERROR indicates error events. A corresponding event number is displayed in STATUS if an error occurs. If no error occurs, the STATUS has a value 0. NDR and ERROR/STATUS are also output for the RESET of the P_RCV_RK instruction (parameter LEN == 16#00). The binary result BR is reset if an error occurs. If the block is completed without errors, the binary result has the status "1".

Parameters

Parameters	Declaration	Data type	Description
EN_R	INPUT	BOOL	Enable data reading
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Receiving is disabled.
LADDR	INPUT	INT	Start address of CP 341 The start address is taken from STEP 7.
DB_NO	INPUT	INT	Data block number Receive DB No.: CPU-specific, zero is not allowed
DBB_NO	INPUT	INT	Data byte number 0 ≤ DBB_NO ≤ 8190 Receive data starting from data byte
NDR[1]	OUTPUT	BOOL	Job completed without errors, data received STATUS parameter == 16#00;
ERROR[1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 7612) contains the error information.
LEN[1]	OUTPUT	INT	Length of the message frame received 1 ≤ LEN ≤ 4096, specified in number of bytes

Parameters	Declaration	Data type	Description
STATUS[1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 7612) contains the error information.
[1] The parameter is available until the next call of the instruction!			

Note

The L_TYP, L_NO, L_OFFSET, L_CF_BYT and L_CF_BIT parameters are irrelevant for the 3964(R) procedure and ASCII driver and require no data.

Allocation in the data area

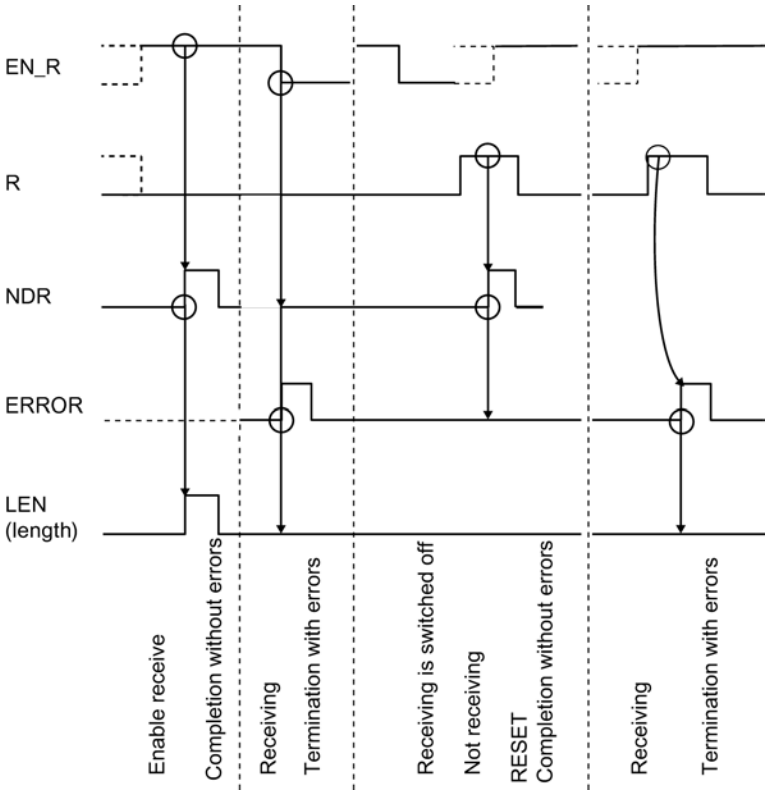
The P_RCV_RK instruction works in combination with an I_RCV_RK instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited

Note

Exception: If the error STATUS == W#16#1E0E occurs, you can examine the SFCERR or SFCSTATUS variable for additional details. This error variable must be loaded via symbolic access to the instance DB.

Time sequence chart

The figure below illustrates the characteristics of the NDR, LEN and ERROR parameters, depending on the input circuit of EN_R and R.



Note

Input EN_R must be set to the static signal state "1". Parameter EN_R must be provided the logic operation result "1" for the duration of the receive request.

Rules

Note

The P_RCV_RK instruction has no parameter check. If incorrect parameters are assigned, the CPU can go into STOP mode.

Note

The CP-CPU startup routine of the P_RCV_RK instruction must be completed before the CP 341 can execute a job triggered after the CPU changed from STOP to RUN.

P_RCV_RK: Receive data with RK 512

Description

The P_RCV_RK instruction sends data from the CP 341 to an S7 data area. For data transmission, the P_RCV_RK instruction is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program.

Operating principle

The (static) signal state "1" at parameter EN_R enables the check for data to be read from CP 341 . An active transmission can be canceled by setting signal state "0" at parameter EN_R . The canceled receive request is terminated with an error message (STATUS output). Receiving is disabled as long as signal state "0" is set at parameter EN_R . Data transmission can take several calls (program cycles), depending on the data volume.

Parameter LADDR specifies the address of the CP 341 to be addressed.

If the communication partner specifies the destination "DB", the data is placed in the data area specified in the RK 512 message frame header. Using the (L_...) parameter, you can identify the destination area type (L_TYP), the destination data block number (L_NO, only relevant for L_TYP = DB), the offset in the destination area (L_OFFSET) and the length (LEN) of transmitted data. If the partner specifies the data destination "DX", the data is saved to the data block (DB) that is specified at the DB_NO and DBB_NO parameters.

If the instruction detects signal state "1" at parameter R, the current transmission job is canceled and the P_RCV_RK instruction is reset to the initial state. Receiving is disabled as long as the signal state at the R parameter is "1".

Output NDR indicates "Job completed without errors/data received" (all data read). The L_TYP, L_NO and L_OFFSET parameters indicate the data storage location for the duration of one cycle. The L_CF_BYT and L_CF_BIT and the length LEN parameters of the respective job are also indicated for one cycle.

Using interprocessor communication flags

The interprocessor communication flag specified in the RK 512 message frame header is verified prior to data reception. The data is not transmitted unless the value of the interprocessor communication flag is "0". When transmission is completed, the interprocessor communication flag is set to the value "1" and output for one cycle (NDR) at the instruction.

The user program can evaluate the interprocessor communication flag in order to find out if the transmitted data can be processed. Once the data has been processed, the user must reset the interprocessor communication flag to "0". The communication partner can now issue a SEND request again.

Parameters

Parameters	Declaration	Data type	Description
EN_R	INPUT	BOOL	Enables data receipt
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Receiving is disabled. Default: 0
LADDR	INPUT	INT	Start address of CP 341 The start address is taken from STEP 7.
DB_NO	INPUT	INT	Data block number of the receive data (destination) Receive data block no.: CPU-specific, zero not allowed (Relevant only for DX data destination)
DBB_NO	INPUT	INT	Data byte number of the receive data (destination) 0 ≤ DBB_NO ≤ 8190 Receive data starting from data byte (Relevant only for DX data destination)
L_TYP[1]	OUTPUT	CHAR	Type of area on local CPU (destination) 'D': Data block
L_NO [1]	OUTPUT	INT	Data block number on local CPU (destination) 0 ≤ L_NO ≤ 255
L_OFFSET[1]	OUTPUT	INT	Data byte number on local CPU (destination) 0 ≤ L_OFFSET ≤ 510
L_CF_BYT[1]	OUTPUT	INT	Interprocessor communication flag byte on local CPU 0 ≤ L_CF_BYTE ≤ 255255 means: Without interprocessor communication flag
L_CF_BIT [1]	OUTPUT	INT	Interprocessor communication flag bit on local CPU 0 ≤ L_CF_BIT ≤ 7
NDR[1]	OUTPUT	BOOL	Job completed without errors, data received STATUS parameter == 16#00;
ERROR[1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 7612) contains the error information.
LEN[1]	OUTPUT	INT	Length of the message frame received 0 ≤ LEN ≤ 4096, specified in number of bytes

Parameters	Declaration	Data type	Description
STATUS[1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 7612) contains the error information.
[1] The parameter is available until the next call of the instruction!			

Allocation in the data area

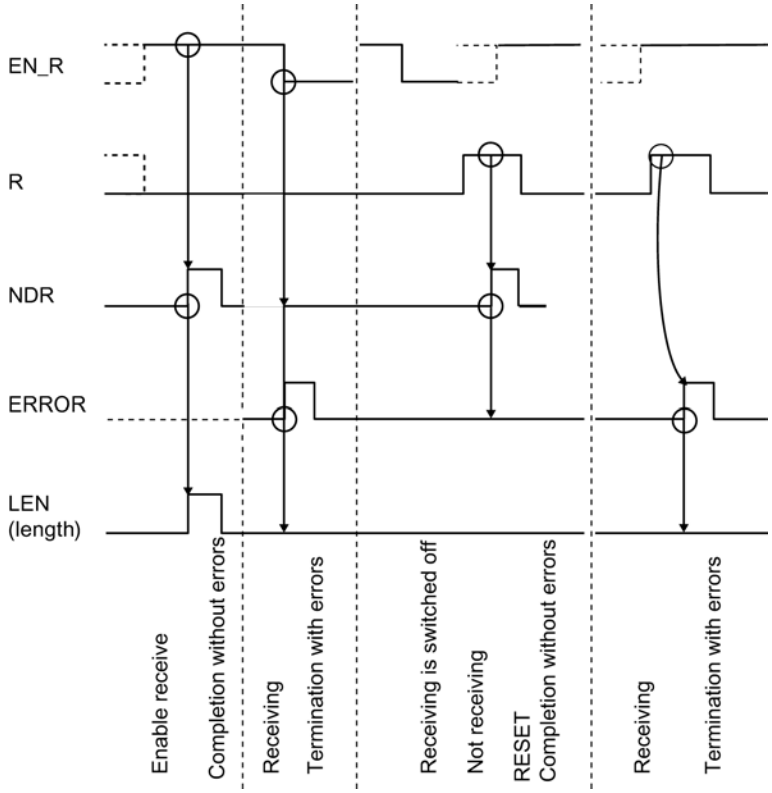
The P_RCV_RK instruction works in combination with an I_RCV_RK instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited

Note

Exception: If the error STATUS == W#16#1E0E occurs, you can examine the SFCERR or SFCSTATUS variable for additional details. This error variable must be loaded via symbolic access to the instance DB.

Time sequence chart

The figure below illustrates the characteristics of the NDR, LEN and ERROR parameters, depending on the input circuit of EN_R and R.



Note

Input EN_R must be set to the static signal state "1". Parameter EN_R must be provided the logic operation result "1" for the duration of the receive request.

How to Handle an Error

ERROR indicates error events. A corresponding event number is displayed in STATUS if an error occurs. If no error occurs, STATUS has a value 0. NDR and ERROR/STATUS are also output for the RESET of the P_RCV_RK instruction (parameter LEN == 16#00). The binary result BR is reset if an error occurs. If the block is completed without errors, the binary result has the status "1".

Rule

Note

The P_RCV_RK instruction has no parameter check. If incorrect parameters are assigned, the CPU can go into STOP mode.

P_RCV_RK: Provide data with RK 512

Description

The P_RCV_RK instruction must be called whenever the communication partner executes a "Fetch data" job (FETCH job).

The P_RCV_RK instruction provides data from an S7 data area to CP 341 . For data transmission, the P_RCV_RK instruction is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program.

Operating principle

A (static) signal state "1" at the EN_R parameter enables a check that determines whether data is to be provided to CP 341 . An active transmission can be canceled by setting signal state "0" at parameter EN_R . The canceled request is terminated with an error message (STATUS output). The job is disabled as long as signal state "0" is set at parameter EN_R . Data transmission can take several calls (program cycles), depending on the data volume.

The first RK 512 message frame specifies the type of source area (L_TYP), the source data block number (L_NO, relevant only if L_TYP = DB), the offset in the source area (L_OFFSET) and the length (LEN) of data to be provided. The instruction evaluates this message frame information and transmits the requested data to the CP 341. The DB_NO and DBB_NO parameters are insignificant for the P_RCV_RK instruction.

Parameter LADDR specifies the address of the CP 341 to be addressed.

If the instruction detects signal state "1" at parameter R, the current transmission job is canceled and the P_RCV_RK instruction is reset to the initial state. The request is disabled as long as the signal state at the R parameter is "1".

Output NDR indicates "Job completed without errors/data received" (all data read). The source of fetched data is indicated at the L_TYP, L_NO and L_OFFSET parameters for the duration of one cycle (valid data types: data blocks, input bytes, output bytes, timers and counters). The L_CF_BYT and L_CF_BIT and the length LEN parameters of the respective job are also indicated for one cycle.

Note

The length of timers or counters the communication partner fetches from the CP 341 is limited to 32 bytes (16 timers or counters, 2 bytes each).

Using interprocessor communication flags

Following receipt of the message frame, the interprocessor communication flags specified in the RK 512 message frame header are checked. The data is not provided unless the value of the interprocessor communication flag is "0". When transmission is completed, the interprocessor communication flag is set to the value "1" and output for one cycle (NDR) at the instruction.

The user program can evaluate the interprocessor communication flag in order to find out if the provided data can be accessed again. Once the data has been processed, the user must reset the interprocessor communication flag to "0". The communication partner can now issue a FETCH request again.

Parameters

Parameters	Declaration	Data type	Description
EN_R	INPUT	BOOL	Enables data provision
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Disables providing. Default: 0
LADDR	INPUT	INT	Start address of CP 341 The start address is taken from STEP 7.
DB_NO	INPUT	INT	irrelevant
DBB_NO	INPUT	INT	irrelevant
L_TYP[1]	OUTPUT	CHAR	Type of area on local CPU (source) 'D': Data block 'M' Memory bit 'I': Inputs 'O': Outputs 'C': Counters 'T': Timers
L_NO[1]	OUTPUT	INT	Data block number on local CPU (source) 0 ≤ L_NO ≤ 255 (only relevant, if L_TYP = D)

Parameters	Declaration	Data type	Description
L_OFFSET[1]	OUTPUT	INT	Data byte number on local CPU (source) 0 ≤ L_OFFSET ≤ 510 (dependent on the area type)
L_CF_BYT[1]	OUTPUT	INT	Interprocessor communication flag byte on local CPU 0 ≤ CF_BYTE ≤ 255 255 means: Without interprocessor communication flag
L_CF_BIT[1]	OUTPUT	INT	Interprocessor communication flag bit on local CPU 0 ≤ CF_BIT ≤ 7
NDR[1]	OUTPUT	BOOL	Job completed without errors, data received STATUS parameter == 16#00;
ERROR[1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 7612) contains the error information.
LEN[1]	OUTPUT	INT	Length of the message frame received 0 ≤ LEN ≤ 4096, specified in number of bytes
STATUS[1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 7612) contains the error information.
[1] The parameter is available until the next call of the instruction!			

Allocation in the data area

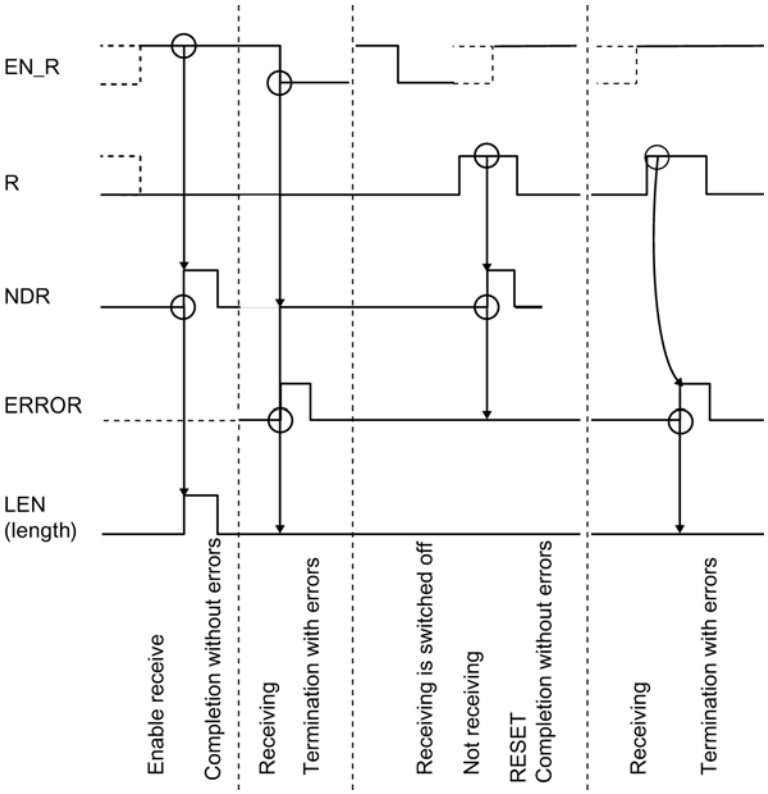
The P_RCV_RK instruction works in combination with an I_RCV_RK instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited

Note

Exception: If the error STATUS == W#16#1E0E occurs, you can examine the SFCERR or SFCSTATUS variable for additional details.

Time sequence chart

The figure below illustrates the characteristics of the NDR, LEN and ERROR parameters, depending on the input circuit of EN_R and R.



Note

Input EN_R must be set to the static signal state "1". Parameter EN_R must be provided the logic operation result "1" for the duration of the receive request.

How to Handle an Error

ERROR indicates error events. A corresponding event number is displayed in STATUS if an error occurs. If no error occurs, the STATUS has a value 0. NDR and ERROR/STATUS are also output for the RESET of the P_RCV_RK instruction (parameter LEN == 16#00). The binary result BR is reset if an error occurs. If the block is completed without errors, the binary result has the status "1".

Rule

Note

The P_RCV_RK instruction has no parameter check. If incorrect parameters are assigned, the CPU can go into STOP mode.

P_PRT341: Print message text with up to 4 variables

Description

The P_PRT341 instruction is available for printing message texts. The P_PRT341 instruction transfers a process message to the CP 341, for example. CP 341 logs the process message to the connected printer.

Operating principle

The P_PRT341 instruction transmits a message text with up to four variables to CP 341. For data transmission, the P_PRT341 instruction is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program.

The DB_NO and DBB_NO parameters enable access to the pointers (to data blocks) for the format string and the four variables. The pointers must be stored without gaps and in a specific sequence in the parameterized data block. This is the pointer DB (see Figure "Pointer DB").

A positive edge at the input REQ initiates the transmission of the message text. The frame starts with the format string of the message text, This is followed by tags 1 to 4.

Data transmission can take several calls (program cycles), depending on the data volume.

The P_PRT341 instruction can be called in the cycle by setting signal state "1" at the R parameter input. This setting cancels the transmission to CP 341 and resets the P_PRT341 instruction to the initial state. Data that has already been received by CP 341 is still sent to the communication partner. A static signal state "1" at the input R indicates that the transmission of print jobs is disabled.

Parameter LADDR specifies the address of the CP 341 to be addressed.

Output DONE indicates "Job completed without errors" and ERROR indicates error events. A corresponding event number is displayed in STATUS if an error occurs. If no error occurs, the STATUS has a value 0. DONE and ERROR/STATUS are also output for a RESET of the P_PRT341 instruction. The binary result BR is reset if an error occurs. If the block is completed without errors, the binary result has the status "1".

Parameters

Parameters	Declaration	Data type	Description
REQ	INPUT	BOOL	Job trigger at the positive edge
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Printing is disabled
LADDR	INPUT	INT	Start address of CP 341 The start address is taken from STEP 7.
DB_NO	INPUT	INT	Data block number Pointer to pointer DB: CPU-specific, zero is not allowed (The pointers to variables and format string are stored in the pointer DB in a fixed order.)
DBB_NO	INPUT	INT	Data byte number 0 ≤ DBB_NO ≤ 8162 Pointer as of data byte
DONE[1]	OUTPUT	BOOL	Job completed without errors STATUS parameter == 16#00;
ERROR[1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 7612) contains the error information.
STATUS[1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 7612) contains the error information.
[1] The parameter is available until the next call of the instruction!			

Assignment in the Data Area, Instance DB

The P_PRT341 instruction works in combination with an I_PRINT instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited.

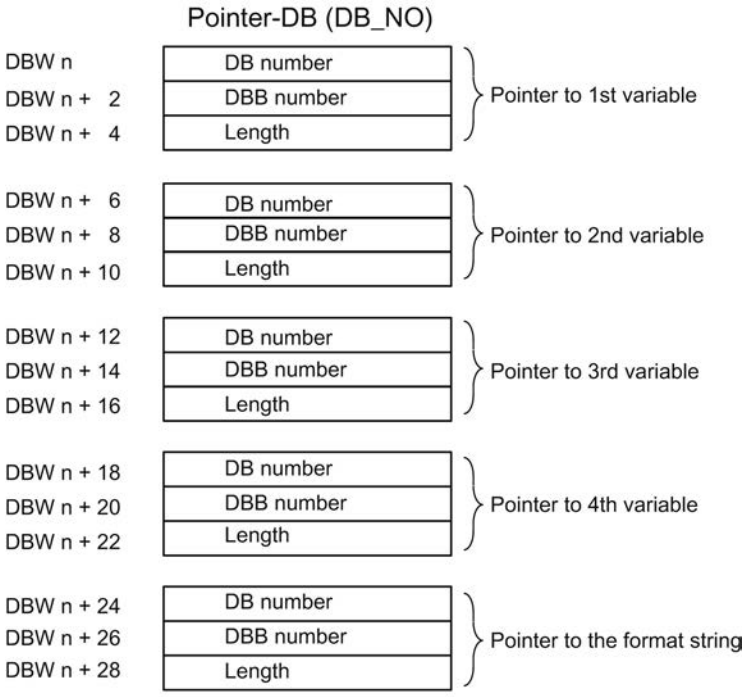
Note

Exception: If the error STATUS == W#16#1E0F occurs, you can examine the SFCERR or SFCSTATUS variable for additional details.

Assignment in the Data Area, Pointer DB

The P_PRT341 instruction uses the DB_NO and DBB_NO parameters to access a pointer DB in which the pointers to the data blocks containing the message texts and variables are stored in a fixed sequence. You have to create the pointer DB.

The figure shows the syntax of the pointer DB that is addressed by means of the DB_NO and DBB_NO parameters of the P_PRT341 instruction.



Permissible DB Number

The permissible DB numbers are CPU-specific. If the value 16#00 is specified as the DB number for "Pointer to variable", this variable is interpreted as not present and the pointer is set to the next variable or the format string.

If the DB number is equal to the value 16#00 in "Pointer to format string", the print job is canceled and event ID 16#1E43 is indicated at parameter output STATUS of the P_PRT341 instruction.

Permissible DBB Number

The variable or format string begins at the parameterized DBB number. The variables can have a maximum length of 32 bytes and the format string can have a maximum length of 150 bytes.

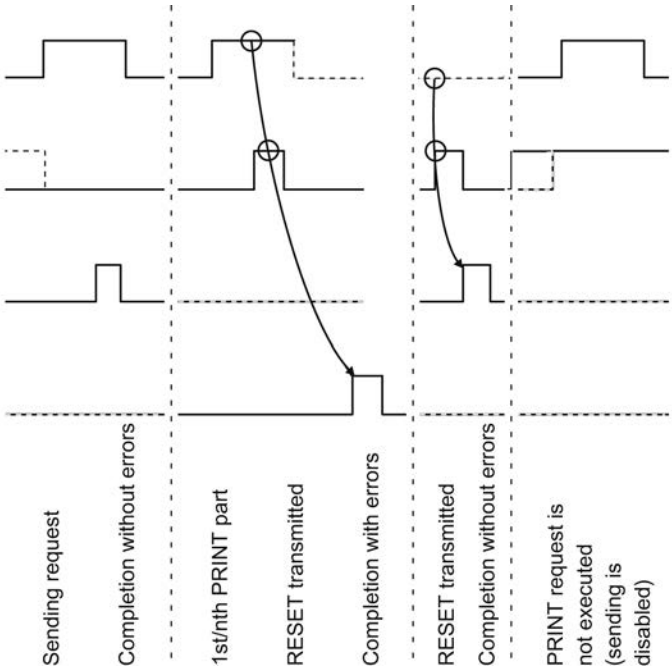
If the maximum length is exceeded, the print job is canceled and event ID 16#1E41 is indicated at parameter output STATUS of the P_PRT341 instruction.

Permissible Length

The entry length in the pointer DB is to be set for each display type (data type) independently from the precision used.

Time sequence chart

The figure below illustrates the characteristics of the DONE and ERROR parameters, depending on the input circuit of REQ and R.



Note

Input REQ is edge-triggered. A positive edge at the input REQ is sufficient. It does not have to have a signal state of "1" during the entire transmission operation.

Rules

Note

The P_PRT341 instruction has no parameter check. If incorrect parameters are assigned, the CPU can go into STOP mode.

Note

The CP-CPU startup routine of the P_PRT341 instruction must be completed before the CP 341 can execute a job triggered after the CPU changed from STOP to RUN. A job initiated in the meantime is not lost. Once the startup coordination is completed, the job is sent to the CP 341.

See also

Data transmission with the printer driver (Page 1072)

V24_STAT_340 / V24_STAT: Reading accompanying signals from the RS232C interface

Description

The V24_STAT_340 (CP 340) / V24_STAT (CP 341) instruction reads the RS 232C accompanying signals from the CP 34x and makes them available to you at the block parameters. The V24_STAT_340 / V24_STAT instruction is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program.

The RS 232C accompanying signals are updated each time the instruction is called (cyclic polling). The CP 34x updates the I/O status at intervals of 20 ms. The inputs/outputs are constantly updated independently of this.

The binary result BR is not affected. The instruction does not output an error message.

Parameter LADDR is used to select the CP 34x be addressed.

Parameters

Parameters	Declaration	Data type	Description
LADDR	INPUT	INT	Start address of CP 34x The start address is taken from STEP 7.
DTR_OUT	OUTPUT	BOOL	Data terminal ready, CP 34x ready (CP 34x output)
DSR_IN	OUTPUT	BOOL	Data set ready, Communication partner ready (CP 34x input)
RTS_OUT	OUTPUT	BOOL	Request to send, CP 34x ready to send (CP 34x output)
CTS_IN	OUTPUT	BOOL	Clear to send, Communication partner can receive data from CP 34x (response to RTS = ON of CP 34x) (CP 34x input)
DCD_IN	OUTPUT	BOOL	Data Carrier detect, Receive signal level (CP 34x input)
RI_IN	OUTPUT	BOOL	Ring Indicator, Ring indicator (CP 34x input)

Allocation in the data area

The V24_STAT_340 / V24_STAT instruction does not occupy any data areas.

Rule

Note

A minimum pulse duration is necessary to detect a signal change. The CPU cycle time, the update time on CP 34x and the response time of the communication partner are decisive variables.

See also

RS232 mode (CP 34x, 44x, CPU 31xC-2 PtP + ET 200S 1SI) (Page 1048)

V24_SET_340 / V24_SET: Writing accompanying signals to the RS232C interface

Description

You can set or reset the interface outputs via the parameter inputs of instruction V24_SET_340 (CP 340) / V24_SET (CP 341). The V24_SET_340 / V24_SET instruction is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program.

The binary result BR is not affected. The instruction does not output an error message.

Parameter LADDR is used to select the CP 34x be addressed.

Parameters

Parameters	Declaration	Data type	Description
LADDR	INPUT	INT	Start address of CP 34x The start address is taken from STEP 7.
RTS	INPUT	BOOL	Request to send, CP 34x ready to send (Control CP 34x output)
DTR	INPUT	BOOL	Data terminal ready, CP 34x ready (Control CP 34x output)

Allocation in the data area

The V24_SET_340 / V24_SET instruction does not occupy any data areas.

See also

RS232 mode (CP 34x, 44x, CPU 31xC-2 PtP + ET 200S 1SI) (Page 1048)

STATUS parameter

Event classes

Event class	Meaning
0	CP startup
1	CP hardware fault
2	Error initializing
3	Incorrect parameter assignment for the instruction
4	The CP detected errors in CP - CPU data traffic
5	Error executing a CPU job
6	Error executing a partner job (for RK 512 only)
7	Send errors
8	Receive errors
9	Received response message frame with error or error message frame from the interconnection partner
10 ($0A_H$)	The CP detected errors in the response message frame of the partner
30 ($1E_H$)	Errors in communication between the CP and CPU

STATUS parameter

Error code(W#16#...)	Description	Remedy
0003	PtP parameters applied	-
0004	Parameter already on CP (timers match)	-
0007	CPU status transition to STOP	-
0008	CPU status transition to RUN/STARTUP	-
0101	Fault while testing operating system EPROM of CP	CP defective; replace CP.
0102	RAM test of CP errored	
0103	Request interface of CP defective	
0110	Fault in CP firmware	Switch module off and on again. If necessary, replace module.
020F	Invalid parameter assignment detected at start of parameter communication. Interface could not be parametered.	Correct invalid parameters and restart.

Error code(W#16#...)	Description	Remedy
0301	Invalid source/destination data type or area does not exist (start address, length) Invalid or no DB (e.g. DB 0) or Other data type invalid or missing Interprocessor communication byte number invalid or Invalid interprocessor communication flag bit number or 'S' or 'F' was not selected (for the P_SND_RK instruction)	Check parameter assignment on CPU and CP and correct if necessary. **RK 512 only:** Partner returns invalid parameters in telegram header. Check parameters on CPU and CP; possibly create block. See request tables for valid data types. **RK 512 only:** Partner provides incorrect parameters in the message frame header.
0403	Incorrect, unknown or illegal data type	Check the program, for example to find incorrect parameter assignments in the instruction.
0407	An error has occurred during data transmission between the CPU and CP	If the error message persists, check the parameter assignment of the instructions called in the user program. If the error is signaled immediately after POWER ON, a connection to the CPU is not set up at this time. With the 3964(R) procedure and the ASCII driver, the data transmission of the receiving CP 341 is repeated until the data are transferred to the CPU. With RK 512, the request is acknowledged negatively and must be repeated in the user program. If the message occurs sporadically during active data transmission, the CPU can not accept the data at times. With the 3964(R) procedure and the ASCII driver, the data transmission of the receiving CP 341 is repeated until the data are transferred to the CPU. With RK 512, the request is acknowledged negatively and must be repeated in the user program. To correct or avoid errors, you should increase the call rate for the P_RCV_RK instruction in the user program.
0408	Error during data transmission between the CPU and the CP (receipt)	
	• CPU is temporarily overloaded, request is repeated	• Reduce the number of communication calls
	• The data area of the CPU cannot be accessed for the time being, e.g., because the receive block is not called often enough.	• Increase the call rate for the receive block
	• The data area of the CPU cannot be accessed for the time being, e.g., because the receive block is disabled in the interim (EN=false).	• Check if the receive block has been disabled too long

Error code(W#16#...)	Description	Remedy
0409	Data reception is not possible. Error during data transmission between the CPU and the CP (receipt). Data reception is not possible. After multiple attempts, the request was cancelled after 10 s, because	
	• The receive block is not called.	• Check your user program to determine whether the receive block is being executed.
	• The receive block is disabled.	• Check if the receive block is disabled.
	• The data area of the CPU cannot be accessed.	• Check if the data area to which the data is to be transferred is available.
	• Insufficient length of the data area on the CPU.	• Check the length of the data area
040A	An error has occurred during data transmission between the CPU and the CP. The data transmission was canceled due to a RESET, because: • Destination DB is not available • Destination DB is too short • RESET bit set at the instruction	Create destination DB in the user program or increase the length of the existing destination DB.
0501	Current request aborted as a result of CP restart.	No help available during POWER ON. When changing the parameters of the CP in the programming device, before writing an interface you should ensure there are no more requests running from the CPU.
0502	Job not permitted in this CP operating mode (e.g., device interface not parameterized)	Set the parameters for the device interface.
0505	Only for printer drivers: System data block with message texts not available on the CP	Use the parameter assignment software to configure the message texts and then carry out a warm restart.
0506	Only for printer drivers: Message text not available	Use the parameter assignment software to configure the message texts and then carry out a warm restart.
0507	Only for printer drivers: Message text too long	Edit the message text to reduce it to a length of fewer than 150 characters (or no more than 250 characters if it contains variables).
0508	Only for printer drivers: Too many conversion instructions	You have configured more conversion instructions than variables. Conversion instructions without associated variables will be ignored.
0509	Only for printer drivers: Too many variables	You have configured more variables than conversion instructions. Variables without associated conversion instructions will not be output.

Error code(W#16#...)	Description	Remedy
050A	**Only for printer drivers:** Unknown conversion instruction	Check the conversion instruction. Undefined or unsupported conversion instructions are replaced in the printout with ******.
050B	**Only for printer drivers:** Unknown control instruction	Check the control instruction. Undefined or unsupported control instructions will be ignored. The control instruction will not be output as text either.
050C	**Only for printer drivers:** Conversion instruction not executable	Check the conversion instruction. Conversion instructions that cannot be executed are output in the printout in accordance with the defined width and the valid remainder of the conversion instruction or the default display with * characters.
050D	**Only for printer drivers:** Width in conversion instruction too small or too great	Correct the specified width of the variable in the conversion instruction on the basis of the variable's maximum possible number of characters in text-based display types (A, C, D, S, T, Y, Z). Only as many characters as will fit in the specified width appear in the printout; the text is truncated to this width. In all other cases, * characters are output corresponding to the width.
050E	**Only for ASCII drivers:** An error occurred while sending. The defined end-of-text characters did not occur within the maximum allowed length or in the case of automatic appending, the maximum allowed transmission length was exceeded.	Extend the end-of-text characters in the transmission buffer at the desired point or select a shorter telegram length for automatic appending.
0514	Specified start addresses too high for desired data type or start address or DB/DX number too low.	Refer to the request tables for the permissible start addresses and DB/DX numbers that can be specified in the program.
0515	**RK 512 only:** Incorrect bit number specified in the interprocessor communication flag.	Permissible bit numbers: 0 to 7
0516	**RK 512 only:** Specified CPU number too high.	Permissible CPU no.: none, 0, 1, 2, 3 or 4
0517	Transmission length > 1 kbyte is too long for CP or length is too short for interface parameter.	Split the request up into several shorter requests.
051A	**RK 512 only:** Error sending a command message frame An associated procedure number has just been entered in STATUS.	See the remedy for the previous error number.

Error code(W#16#...)	Description	Remedy
051B	**Only for printer drivers:** Precision invalid	Correct the specified precision in the conversion instruction. The precision is always initialized with a dot prefix to identify and limit the width (for example, ".2" to output the decimal point and two decimals.) Precision is only relevant to display types A, D, F and R. It is ignored otherwise.
051C	**Only for printer drivers:** Variable invalid (variable length incorrect/incorrect type)	Correct the specified variable. The corresponding table indicates the possible data types for each display type.
051E	**Only for printer drivers:** The "line end sequences" sent with this job (i.e.: $R / $L / $N) do not fit (any longer) on the (initial) page.	Increase the length of your page, reduce the number of lines (or line feeds) or spread your printout over a number of pages.
0601	Error in 1st command byte (not 00 or FFH)	Header layout error at partner. You can interconnect an interface tester in the transmission line to debug the partner device.
0602	Error in 3rd command byte (not A, 0 or FFH)	Header layout error at partner. You can interconnect an interface tester in the transmission line to debug the partner device.
0603	Error in 3rd command byte in the case of continuation telegrams (command not as for 1st telegram)	Header layout error at partner. You can interconnect an interface tester in the transmission line to debug the partner device.
0604	Error in 4th command byte (command letter incorrect)	Header layout error at partner or a command combination has been requested that is not permitted at the CP. Check the permissible commands. You can interconnect an interface tester in the transmission line to debug the partner device.
0605	Error in 4th command byte in the case of continuation telegrams (command not as for 1st telegram)	Header layout error at partner. You can interconnect an interface tester in the transmission line to debug the partner device.
0606	Error in 5th command byte (DB number not permissible)	Refer to the request tables for the permissible DB numbers, start addresses or lengths.
0607	Error in 5th or 6th command byte (start address too high)	Refer to the request tables for the permissible DB numbers, start addresses or lengths.
0608	Error in 7th or 8th command byte (impermissible length)	Obtain from the request tables the permissible DB/DX numbers, start addresses or lengths.
0609	Error in 9th and 10th command byte (coordination flag for this data type impermissible or bit number too high)	Header layout error at partner. Find out from the request tables when a coordination flag is permitted.

Error code(W#16#...)	Description	Remedy
060A	Error in the 10th command byte (illegal CPU number)	Header layout error at partner.
060B	SEND message frame was longer/shorter than expected (more/less data received than announced in message frame header).	Correction required at the partner
060C	FETCH command message frame with user data received.	Correction required at the partner
060D	The CP received a message frame during an invalid operating mode:	
	• Receive connection between the CPU and CP is not set up or not yet correctly set up	• Check the parameter assignment of the addressed connection
	• CP startup is not yet completed	• This error message can occur only during CP startup. Repeat the request.
	• The receiving CPU is in STOP mode	• Place the CPU back in RUN mode and repeat the request.
	• Parameters for the interface are currently being assigned	• This is a temporary error. Repeat the request.
060E	Synchronous fault of partner • New (continuation) command message frame received before response message frame was sent. • 1. 1st command telegram expected and continuation telegram came. • Continuation command telegram expected and 1st telegram came	This error may be reported after your own automation system is restarted in the case of long message frames or when the partner is restarted. These cases represent normal system start-up characteristics. The error can also occur during operation as a consequence of error statuses only recognized by the partner. Otherwise, you have to assume an error on the part of the partner device. The error may not occur in the case of requests <128 bytes.
060F	DB locked by coordination function	In local program: Reset the interprocessor communication flag after the last transmission data was processed. In partner program: Repeat the request
0610	Message frame received too short (length < 4 bytes in the case of continuation or response message frames or <10 bytes in the case of command message frames)	You can interconnect an interface tester in the transmission line to debug the partner device.
0611	Telegram length and length specified in telegram header are not the same.	You can interconnect an interface tester in the transmission line to debug the partner device.
0612	Error while sending the (continuation) response message frame. An associated procedure error number has been entered in STATUS immediately beforehand.	See remedy for the error number entered immediately beforehand in STATUS.

Error code(W#16#...)	Description	Remedy
0701	Sending of the first repetition: • An error was detected when transmitting the telegram or • The partner requested a repetition by means of a negative acknowledgment character (NAK).	A repetition is not an error, but it can indicate that there is interference on the data link or the partner device has malfunctioned. If the message frame still has not been transmitted after the maximum number of repetitions, an error number describing the first error that occurred is output.
0702	**Only for 3964(R):** Error establishing connection: After STX was sent, NAK or any other code (except for DLE or STX) was received.	Check for malfunction of the partner device, possibly by using interface test device interconnected in the data link.
0703	**Only for 3964(R):** Acknowledgment delay time (QVZ) exceeded: After STX was sent, partner did not respond within the acknowledgment delay time.	The partner device is too slow or not ready to receive or there is a break in the transmission line, for example. You can interconnect an interface tester in the transmission line to debug the partner device.
0704	**Only for 3964(R):** Termination by partner: One or more codes were received from the partner during sending.	Check if the partner is also indicating errors, possibly because not all transmitted data arrived (e.g., break in the transmission line), fatal errors are pending or the partner device has malfunctioned. If necessary, use an interface test device switched into the transmission line to check.
0706	**Only for 3964(R):** End-of-connection error: • Partner rejected telegram at end of connection with NAK or a random string (except for DLE) or • Acknowledgment characters (DLE) received too early.	Check if the partner is also indicating errors, possibly because not all transmitted data arrived (e.g., break in the transmission line), fatal errors are pending or the partner device has malfunctioned. If necessary, use an interface test device switched into the transmission line to check.
0707	**Only for 3964(R):** Acknowledgment delay time exceeded at end of connection or response monitoring time exceeded after a send telegram: After connection termination with DLE ETX, no response received from partner within acknowledgment delay time.	Partner device too slow or faulty. If necessary, use an interface test device switched into the transmission line to check.
0708	**ASCII driver and printer driver only:** The wait time for XON or CTS = ON has expired.	The communication partner is faulty, too slow or has been taken offline. Check the communication partner; you may need to change the parameter assignment.
0709	Connection setup not possible. Number of permitted setup attempts exceeded.	Check the interface cable or the transmission parameters. Also check that receive function between CPU and CP is correctly configured at the partner device.

Error code(W#16#...)	Description	Remedy
070A	The data could not be transmitted. The permitted number of transfer attempts was exceeded.	Check the interface cable or the transmission parameters.
0801	Expectation of the first repetition: An error was detected on receiving a message frame and the CP requested repetition from the partner via a negative acknowledgment (NAK).	A repetition is not an error, but it can indicate that there is interference on the data link or the partner device has malfunctioned. If the message frame still has not been transmitted after the maximum number of repetitions, an error number describing the first error that occurred is output.
0802	**Only for 3964(R):** Error establishing connection: • In idle mode, one or more random codes (other than NAK or STX) were received or • After an STX was received, the partner sent more characters without waiting for the response DLE. After partner power ON: • While partner is being switched on, the CP receives an undefined character.	Check for malfunction at partner device, possibly by using interface test device switched into the transmission line.
0805	**Only for 3964(R):** Logical error while receiving: After DLE was received, a further random code (other than DLE or ETX) was received.	Check if the partner always duplicates the DLE in the telegram header and data string or the connection is terminated with DLE ETX. Check for malfunction at partner device, possibly by using interface test device switched into the transmission line.
0806	Character delay time (ZVZ) exceeded: • Two successive characters were not received within character delay time or Only for 3964(R): • 1. character after sending of DLE while establishing connection was not received within the character delay time.	Partner device too slow or faulty. Interconnect an interface tester in the transmission line to see if this is the case.
0808	**Only for 3964(R):** Error in block check character (BCC): The value of BCC calculated internally does not match the BCC received by the partner when the connection was terminated.	Check if the connection is seriously disrupted; in this case you may also occasionally see error codes. Check for malfunction at partner device, possibly by using interface test device switched into the transmission line.
080A	There is no free receive buffer available: No receive buffer space available for receiving data.	Increase the call rate for the P_RCV_RK instruction.

Error code(W#16#...)	Description	Remedy
080C	Transmission error: • A transmission error (parity error, stop bit error, overflow error) was detected. **Only for 3964(R):** • If a corrupted character is received in idle mode, the error is reported immediately so that disturbances on the transmission line can be detected early. **Only in the case of RK 512 and 3964(R):** • If this happens during send or receive operations, repetition is started.	Disturbances on the data link cause telegram repetitions, thus lowering user data throughput. The risk of undetected error increases. Change your system setup or cable wiring. Check the connecting cables of the communication partners or verify that both devices have matching settings for baud rate, parity and number of stop bits.
080D	BREAK: Break in receive line to partner.	Reconnect or switch on partner.
0815	Discrepancy between settings for transmission attempts of the CP and the communication partner.	Set the same number of transfer attempts at communications partner as at CP. You can interconnect an interface tester in the transmission line to debug the partner device.
0816	The length of a received message frame exceeded the maximum specified length.	Correction required at the partner
0818	**For ASCII driver only:** DSR = OFF or CTS = OFF	The partner has switched the DSR or CTS signal to "OFF" before or during a transmission. Check the partner's control of the RS 232C accompanying signals.
0902	**RK 512 only:** Memory access error in partner (memory does not exist) **With SIMATIC S5 as partner:** • Incorrect area at status word or • Data area does not exist (except DB/DX) or • Data area too short (except DB/DX)	Check that the partner has the desired data area and that it is big enough or check the parameters of the called system function block. Check the specified length at the system function block.
0903	**RK 512 only:** DB/DX access error in the partner (DB/DX does not exist or is too short) **With SIMATIC S5 as partner:** • DB/DX does not exist or • DB/DX too short or • DB/DX number impermissible Permissible source area exceeded with FETCH request	Check that the partner has the desired data area and that it is big enough or check the parameters of the called system function block. Check the specified length at the system function block.
0904	**RK 512 only:** Partner reports "Illegal request type".	Partner malfunction, because the CP never outputs a system command.

Error code(W#16#...)	Description	Remedy
0905	**RK 512 only:** Error at partner or at SIMATIC S5 as partner: • Source/destination type not permissible or • Memory error in partner programmable controller or • Error notifying CP/CPU at the partner or • Partner programmable controller is in STOP mode	Check if the partner can transmit the desired data type. Check the structure of the hardware at the partner. Set the mode selector switch of the partner programmable controller to RUN.
0908	**RK 512 only:** Partner detects synchronous error: Telegram sequence error.	This error occurs at restart of your own programmable controller or of the partner. This represents normal system start-up characteristics. You do not need to correct anything. The error is also conceivable during operation as a consequence of previous errors. Otherwise, you can assume an error on the part of the partner device.
0909	**RK 512 only:** DB/DX locked at the partner by coordination flag	In partner program: Reset the coordination memory after the last transmission data was processed! In own program: Repeat the request.
090A	**RK 512 only:** Errors in message frame header that are detected by the partner: 3rd command byte in header is incorrect	Check if the error is the result of disturbances or of a malfunction at the partner. Interconnect an interface tester in the transmission line to see if this is the case.
090B	**RK 512 only:** Error in message frame header: 1. or incorrect command byte 4 in header	Check if the error is the result of disturbances or of a malfunction at the partner. Interconnect an interface tester in the transmission line to see if this is the case.
090C	**RK 512 only:** Partner detects incorrect message frame length (total length).	Check if the error is the result of disturbances or of a malfunction at the partner. Interconnect an interface tester in the transmission line to see if this is the case.
090D	**RK 512 only:** Partner was not yet restarted.	Restart the partner programmable controller or set the mode selector switch on the CP to RUN.
090E	**RK 512 only:** Unknown error number received in the response message frame.	Check if the error is the result of disturbances or of a malfunction at the partner. Interconnect an interface tester in the transmission line to see if this is the case.

Error code(W#16#...)	Description	Remedy
0A01	**RK 512 only:** Synchronization error of partner, because: • Response telegram without request • Response telegram received before continuation telegram sent • Continuation response message frame received after an initial message frame was sent • A first response telegram was received after a continuation telegram was sent	This error is reported after your own programming device is restarted in the case of long telegrams or when the partner is restarted. This represents normal system start-up characteristics. You do not have to correct anything. The error can also occur during operation as a consequence of error statuses only recognized by the partner. Otherwise, you can assume an error on the part of the partner device. The error may not occur in the case of requests <128 bytes.
0A02	RK 512 only: Error in the structure of the received response message frame (1st byte not 00 or FF)	Check for malfunction at partner device, possibly by using interface test device switched into the transmission line.
0A03	RK 512 only: Received response message frame has too many or too few data.	Check for malfunction at partner device, possibly by using interface test device switched into the transmission line.
0A04	RK 512 only: Response message frame for SEND request arrived with data.	Check for malfunction at partner device, possibly by using interface test device switched into the transmission line.
0A05	RK 512 only: No response message frame received from partner within monitoring time.	Is the partner a slow device? This error is also often displayed as a consequence of a previous error. For example, procedure receive errors (event class 8) can be displayed after a FETCH message frame was sent. Reason: The response message frame could not be received because of disturbances. The monitoring time expires. This error also possibly occurs if the partner is restarted before it could respond to the last received FETCH message frame.
1E0D	Job aborted due to warm restart, hot restart or reset	
1E0E	Static error when the RD_REC SFC was called. Return value RET_VAL of SFC is available for evaluation in SFCERR variable in instance DB.	Load SFCERR variable from instance DB.
1E0F	Static error when the WR_REC SFC was called. Return value RET_VAL of SFC is available for evaluation in SFCERR variable in instance DB.	Load SFCERR variable from instance DB.
1E41	Invalid number of bytes specified at instruction parameter LEN	You must stay within a range of values of 1 to 4096 bytes.
1E42	P_PRINT_RK instruction: The number of bytes specified for the variable or format string in the pointer DB under length is not permissible.	You must specify a permissible length: 32 bytes for variables, 150 bytes for format strings.
1E43	P_PRINT_RK instruction: No pointer available for format string.	Enter the data block no. and data word no. for the format string in the pointer DB.

Note

An error message is only output if the ERROR bit (job canceled with error) is also set. In all other cases, the STATUS word is zero.

Calling the SFCERR or SFCSTATUS variable

You can call the SFCERR or SFCSTATUS variable to obtain more detailed information about the pending event class 30 error, 14 (1E0EH) or 15 (1E0FH).

You can load the SFCERR or SFCSTATUS variable only by means of symbolic access to instance DB of the corresponding instruction.

Error messages entered in the SFCERR variable can be found in the system functions SFC 58 "WR_REC: Write data record to I/O (Page 6496)" and SFC 59 "RD_REC: Read data record from I/O (Page 6491)".

Error messages entered in the SFCSTATUS variable can be found in the system functions SFB 52 "RDREC: Read data record (Page 6459)" and SFB 53 "WRREC: Write data record (Page 6461)".

PtP data link CP 440

SEND_440: Sending Data

Description

The SEND_440 instruction transmits a data block from a DB to the CP 440 as specified at the DB_NO, DBB_NO and LEN parameters. For data transmission, the SEND_440 instruction is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program.

Operating principle

Data transmission is initiated by a positive edge at the input REQ. Data transmission can take several calls (program cycles), depending on the data volume.

The SEND_440 instruction can be called in the cycle by setting signal state "1" at parameter input R. This setting cancels the transmission to CP 440 and resets the SEND_440 instruction to the initial state. Data that has already been received by CP 440 is still sent to the communication partner. A static signal state "1" at the input R indicates that data transmission is disabled.

Parameter LADDR specifies the address of the CP 440 to be addressed.

Output DONE indicates "Job completed without errors" and ERROR indicates error events. A corresponding event number is displayed in STATUS if an error occurs. If no error occurs, the STATUS has a value 0. DONE and ERROR/STATUS are also output for a RESET of the

SEND_440 instruction. The binary result BR is reset if an error occurs. If the block is completed without errors, the binary result has the status "1".

Parameters

Parameters	Declaration	Data type	Description
REQ	INPUT	BOOL	Job trigger at the positive edge
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Transmission is disabled.
LADDR	INPUT	INT	Start address of CP 440 The start address is taken from STEP 7.
DB_NO	INPUT	INT	Data block number Send DB no.: CPU-specific, zero not allowed
DBB_NO	INPUT	INT	Data byte number 0 ≤ DBB_NO ≤ 8190 Transmitted data as of data byte; Offset is CPU-specific
LEN	INPUT	INT	Data length 1 ≤ LEN ≤ 400, specified in number of bytes
DONE[1]	OUTPUT	BOOL	Job completed without errors STATUS parameter == 16#00;
ERROR[1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 7631) contains the error information.
STATUS[1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 7631) contains the error information.
[1] The parameter is available until the next call of the instruction!			

Allocation in the data area

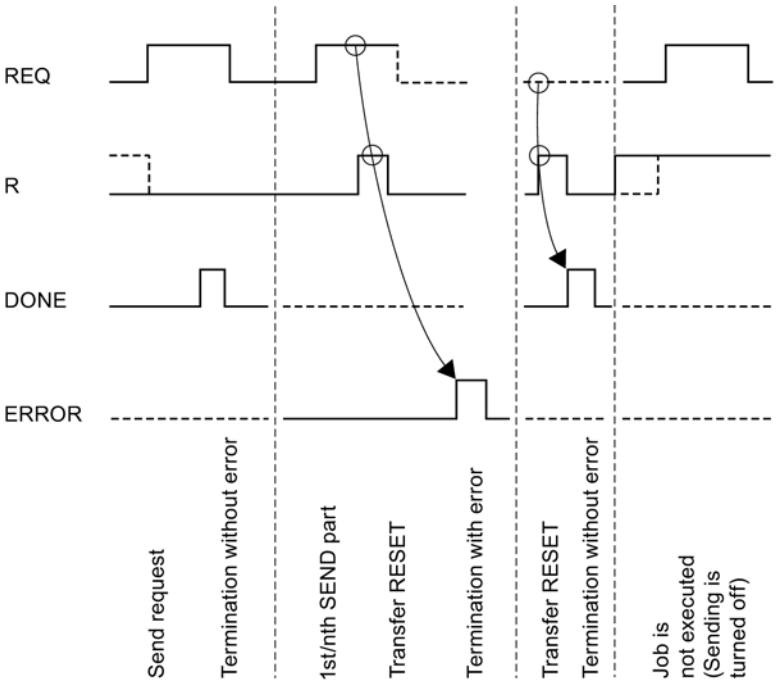
The SEND_440 instruction works in combination with an I_SEND_440 instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited

Note

Exception: If the error STATUS == W#16#1E0F occurs, you can examine the SFCERR variable for additional details.

Time sequence chart

The figure below illustrates the characteristics of the DONE and ERROR parameters, depending on the input circuit of REQ and R.



Note

Input REQ is edge-triggered. A positive edge at the input REQ is sufficient. The RLO (result of the logic operation) does not necessarily have to be "1" for the entire duration of the transmission.

Rules

Note

The SEND_440 instruction has no parameter check. If incorrect parameters are assigned, the CPU can go into STOP mode.

RECV_440: receiving data

Description

The RECV_440 instruction transfers data from the CP 440 to an S7 data area as specified at the DB_NO, DBB_NO and LEN parameters. For data transmission, the RECV_440 instruction

is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program. .

Operating principle

The (static) signal state "1" at parameter EN_R enables the check for data to be read from CP 440. An active transmission can be canceled by setting signal state "0" at parameter EN_R. The canceled receive request is terminated with an error message (STATUS output). Receiving is disabled as long as signal state "0" is set at parameter EN_R. Data transmission can take several calls (program cycles), depending on the data volume.

If the instruction detects signal state "1" at the R parameter, the current transmission job is canceled and the RECV_440 instruction is reset to the initial state. Receiving is disabled as long as signal state "1" is set at parameter R.

Parameter LADDR is used to select the CP 440 to be addressed.

Output NDR indicates "Job completed without errors/data received" (all data read) and ERROR indicates error events. A corresponding event number is displayed in STATUS if an error occurs. If no error occurs, STATUS has a value 0. NDR and ERROR/STATUS are also output for the RESET of the RECV_440 instruction (parameter LEN == 16#00). The binary result BR is reset if an error occurs. If the block is completed without errors, the binary result has the status "1".

Parameters

Parameters	Declaration	Data type	Description
EN_R	INPUT	BOOL	Enable data reading
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Receiving is disabled.
LADDR	INPUT	INT	Start address of CP 440 The basic address can be found in the configuration table in STEP 7.
DB_NO	INPUT	INT	Data block number Receive data block no.: CPU-specific, zero not allowed
DBB_NO	INPUT	INT	Data byte number Offset is CPU-specific
NDR[1]	OUTPUT	BOOL	Job completed without errors, data received STATUS parameter == 16#00;
ERROR[1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 7631) contains the error information.
LEN[1]	OUTPUT	INT	Length of the message frame received 1 ≤ LEN ≤ 400, specified in number of bytes

Parameters	Declaration	Data type	Description
STATUS[1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 7631) contains the error information.
[1] The parameter is available until the next call of the instruction!			

Allocation in the data area

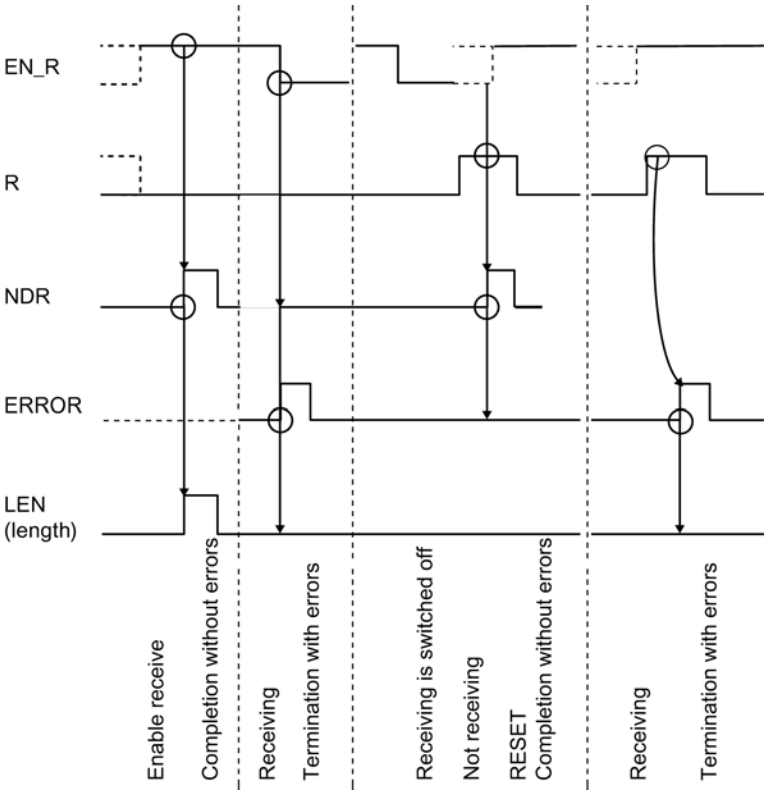
The RECV_440 instruction works in combination with an I_RECV_440 instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited

Note

Exception: If the error STATUS == W#16#1E0E occurs, you can examine the SFCERR variable for additional details.

Time sequence chart

The figure below illustrates the characteristics of the NDR, LEN and ERROR parameters, depending on the input circuit of EN_R and R.



Note

Input EN_R must be set to the static signal state "1". Parameter EN_R must be provided the logic operation result "1" for the duration of the receive request.

Rule

Note

The RECV_440 instruction has no parameter check. If incorrect parameters are assigned, the CPU can go into STOP mode.

RES_RECV: Delete receive buffer

Description

The RES_RECV instruction clears the entire input buffer of CP 440. All message frames in memory are discarded. A message frame received at the time the RES_RECV instruction is called is saved.

Operating principle

A positive edge at the input REQ enables the instruction. The job can run over several calls (program cycles).

The RES_RECV instruction can be called in the cycle by setting signal state "1" at parameter input R. This action cancels the delete operation and resets the RES_RECV instruction to the initial state.

Parameter LADDR specifies the address of the CP 440 to be addressed.

Output DONE indicates "Job completed without errors" and ERROR indicates error events. A corresponding event number is displayed in STATUS if an error occurs. If no error occurs, STATUS has a value 0. DONE and ERROR/STATUS are also output for RESET of the RES_RECV instruction. The binary result BR is reset if an error occurs. If the block is completed without errors, the binary result has the status "1".

Parameters

Parameters	Declaration	Data type	Description
REQ	INPUT	BOOL	Job trigger at the positive edge
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Transmission is disabled.

Parameters	Declaration	Data type	Description
LADDR	INPUT	INT	Start address of CP 440 The start address is taken from STEP 7.
DONE[1]	OUTPUT	BOOL	Job completed without errors STATUS parameter == 16#00;
ERROR[1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 7631) contains the error information.
STATUS[1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 7631) contains the error information.
[1] The parameter is available until the next call of the instruction!			

Allocation in the data area

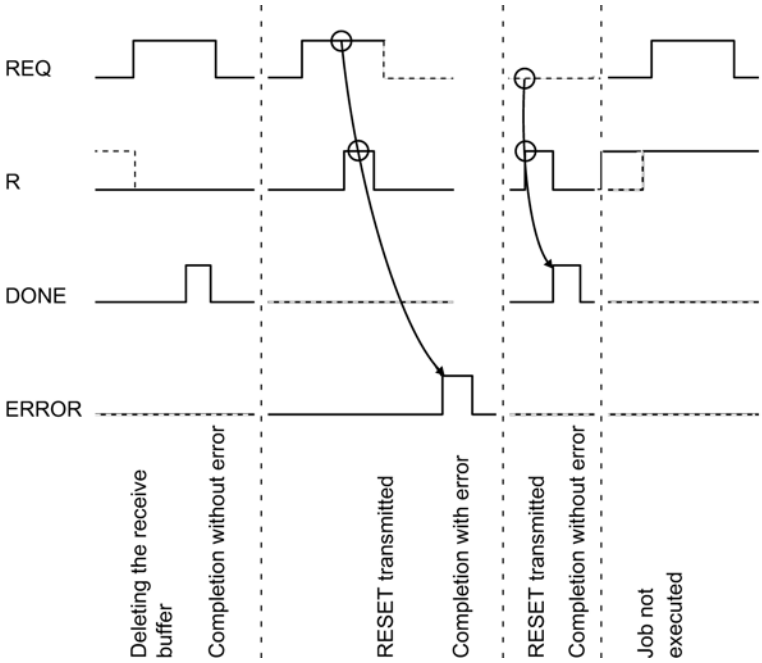
The RES_RECV instruction works in combination with an I_RES_RECV instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited

Note

Exception: If the error STATUS == W#16#1E0F occurs, you can examine the SFCERR variable for additional details.

Time sequence chart

The figure below illustrates the characteristics of the DONE and ERROR parameters, depending on the input circuit of REQ and R.



Note

Input REQ is edge-triggered. A positive edge at the input REQ is sufficient. The RLO (result of the logic operation) does not necessarily have to be "1" for the entire duration of the transmission.

Rule

Note

The RES_RECV instruction has no parameter check. If incorrect parameters are assigned, the CPU can go into STOP mode.

STATUS parameter

Event classes

Event class	Meaning
0 (0_H)	CP startup
2 (02_H)	Error initializing
5 (05_H)	Error executing a CPU job
7 (07_H)	Send errors
8 (08_H)	Receive errors
11 ($0B_H$)	Warnings
30 ($1E_H$)	Errors in communication between the CP and CPU
128 (80_H)	Module fault

STATUS parameter

Error code(W#16#...)	Description	Remedy
0001	Initialization of the CP completed	-
0003	PtP parameters applied	-
0004	Parameter already on CP (timers match)	-
0007	CPU status transition to STOP	-
0008	CPU status transition to RUN/STARTUP	-
0201	No parameters Parameter memory empty or has unknown contents	Load interface parameters.
0204	Impermissible character frame	Correct the impermissible parameter assignment, load the parameters onto the module and carry out a cold restart.
0205	Impermissible transmission rate	Correct the impermissible parameter assignment, load the parameters onto the module and carry out a cold restart.
020F	Invalid parameter assignment detected at start of parameter communication. Interface could not be parametered.	Correct invalid parameters and restart.
0212	Impermissible • Monitoring time • Character Delay Time Only for 3964(R): • Acknowledgment delay	Correct the impermissible parameter assignment, load the parameters onto the module and carry out a cold restart.

Error code(W#16#...)	Description	Remedy
0501	Current request aborted as a result of CP restart.	No help available during POWER ON. When changing the parameters of the CP in the programming device, before writing an interface you should ensure there are no more requests running from the CPU.
0502	Job not permitted in this CP operating mode (e.g., device interface not parameterized)	Set the parameters for the device interface.
050E	• Invalid frame length or	• Telegram length is > 400 bytes. Reduce the message frame length or
	• The configured end-of-text characters did not occur within the maximum permissible length.	• Add the end-of-text characters at the place you want in the send buffer.
0517	Transmission length > 400 bytes exceeds CP capacity.	Split the job up into several shorter jobs.
051D	Send/receive job interrupted by: • Communication block reset • Parameter reassignment	Repeat the call for the communication block.
0522	A new SEND job has been started, although the old job has not yet been completed.	Only start the new SEND job when the old job has been completed with DONE or ERROR.
0701	Only for 3964(R): Sending of the first repetition: • An error was detected when transmitting the telegram or • The partner requested a repetition by means of a negative acknowledgment character (NAK).	A repetition is not an error, but it can indicate that there is interference on the data link or the partner device has malfunctioned. If the message frame has not been transmitted after the maximum number of repetitions, an error number describing the first error that occurred is output.
0702	Only for 3964(R): Error establishing connection: After STX was sent, NAK or any other code (except for DLE or STX) was received.	Check for malfunctioning of the partner device, possibly using an interface test device switched into the transmission line.
0703	Only for 3964(R): Acknowledgment delay time (QVZ) exceeded: After STX was sent, partner did not respond within the acknowledgment delay time.	The partner device is too slow or not ready to receive or there is a break on the send line, for example. Identify the malfunctioning of the partner device, possibly by using an interface test device switched into the transmission line.
0704	Only for 3964(R): Termination by partner: One or more codes were received from the partner during sending.	Check if the partner is also indicating errors, possibly because not all transmitted data arrived (e.g., break in the transmission line), fatal errors are pending or the partner device has malfunctioned. You can interconnect an interface tester in the transmission line to check the status.

Error code(W#16#...)	Description	Remedy
0705	Only for 3964(R): Negative acknowledgment when sending	Check if the partner also indicates an error; possibly it has not received all of the transmitted data (for example, due to an interrupted data link) or because fatal errors are pending or the characteristics of the partner device is faulty. You can interconnect an interface tester in the transmission line to check the status.
0706	Only for 3964(R): End-of-connection error: • Partner rejected telegram at end of connection with NAK or a random string (except for DLE) or • Acknowledgment characters (DLE) received too early.	Check if the partner is also indicating errors, possibly because not all transmitted data arrived (e.g., break in the transmission line), fatal errors are pending or the partner device has malfunctioned. You can interconnect an interface tester in the transmission line to check the status.
0707	Only for 3964(R): Acknowledgment delay time exceeded at end of connection or response monitoring time exceeded after a send telegram: After connection termination with DLE ETX, no response received from partner within acknowledgment delay time.	Partner device faulty or too slow. You can interconnect an interface tester in the transmission line to check the status.
0708	For ASCII driver only: The wait time for XON or CTS = ON has expired.	The communication partner is faulty, too slow or has been taken offline. Check the communication partner; you may need to change the parameter assignment.
0709	Only for 3964(R): Connection setup not possible. Number of permitted setup attempts exceeded.	Check the interface cable or the transmission parameters. Also check that receive function between CPU and CP is correctly configured at the partner device.
070A	Only for 3964(R): The data could not be transmitted. The permitted number of transfer attempts was exceeded.	Check the interface cable or the transmission parameters.
070B	Only for 3964(R): Initialization conflict cannot be resolved because both partners have high priority.	Change the parameter assignment.
070C	Only for 3964(R): Initialization conflict cannot be resolved because both partners have low priority.	Change the parameter assignment.

Error code(W#16#...)	Description	Remedy
0801	Only for 3964(R): Expectation of the first repetition: An error was detected on receiving a telegram and the CP requested repetition from the partner via a negative acknowledgment (NAK).	A repetition is not an error, but it can indicate that there is interference on the data link or the partner device has malfunctioned. If the message frame has not been transmitted after the maximum number of repetitions, an error number describing the first error that occurred is output.
0802	Only for 3964(R): Error establishing connection: • In idle mode, one or more random codes (other than NAK or STX) were received or • After an STX was received, the partner sent more characters without waiting for the response DLE. After partner power ON: • While partner is being switched on, the CP receives an undefined character.	You can interconnect an interface tester in the transmission line to check the error response of the partner device.
0805	Only for 3964(R): Logical error while receiving: After DLE was received, a further random code (other than DLE or ETX) was received.	Check if the partner always duplicates the DLE in the telegram header and data string or the connection is terminated with DLE ETX. You can interconnect an interface tester in the transmission line to check the error response of the partner device.
0806	Character delay time (ZVZ) exceeded: • Two successive characters were not received within character delay time or Only for 3964(R): • 1. character after sending of DLE while establishing connection was not received within the character delay time.	Partner device faulty or too slow. You can interconnect an interface tester in the transmission line to check the status.
0807	Illegal telegram length: A zero-length telegram has been received.	Receipt of a zero-length telegram is not an error. Check why the communication partner is sending message frames without user data.
0808	Only for 3964(R): Error in block check character (BCC): The value of BCC calculated internally does not match the BCC received by the partner when the connection was terminated.	Check if the connection is seriously disrupted; in this case you may also occasionally see error codes. You can interconnect an interface tester in the transmission line to check the error response of the partner device.

Error code(W#16#...)	Description	Remedy
0809	Only for 3964(R): Waiting time for block repetition has expired	Configure the same block wait time in the communication partner as for the CP 440. Identify the malfunction in the communication partner, possibly by using an interface test device switched into the transmission line.
080A	There is no free receive buffer available: No receive buffer space available for receiving data.	Increase the call rate for the RECV_440 instruction.
080C	Transmission error: • A transmission error (parity error, stop bit error, overflow error) was detected. Only for 3964(R): • If a corrupted character is received in idle mode, the error is reported immediately so that disturbances on the transmission line can be detected early. Only for 3964(R): • If this happens during send or receive operations, repetition is started.	Disturbances on the data link cause telegram repetitions, thus lowering user data throughput. The risk of undetected error increases. Change your system setup or cable wiring. Check the connecting cables of the communication partners or verify that both devices have matching settings for baud rate, parity and number of stop bits.
080D	BREAK: Break in receive line to partner.	Reconnect or switch on partner.
080E	Receive buffer overflow when flow control not enabled.	Increase the call rate for the receive instruction in the user program or configure the parameters for communication with flow control.
0810	Parity error	Check the connecting cables of the communication partners or verify that both devices have matching settings for baud rate, parity and number of stop bits.
0811	Character frame error	Check the connecting cables of the communication partners or verify that both devices have matching settings for baud rate, parity and number of stop bits. Change your system setup or cable wiring.
0812	For ASCII driver only: More characters were received after the CP had sent XOFF or set CTS to OFF.	Reassign the parameters for the communication partner or deal with the data in the CP more quickly.
0814	For ASCII driver only: A message frame or several message frames have got lost, because you have been working without flow control.	Work with flow control as much as you can. Use the entire receive buffer. Set the "Reaction to CPU Stop" parameter in the basic parameters to "Continue".
0816	The length of a received message frame was longer than the maximum specified length.	Correction required at the partner

Error code(W#16#...)	Description	Remedy
0820	Static error when calling the SFC RD_REC. The RET_VAL return value of the SFC is provided in the SFCERR variable in the instance DB for evaluation.	Analyze the SFCERR variable from the instance DB.
0B01	Receive buffer more than 2/3 full	Call the receive block more often to avoid the receive buffer overflowing.
8000	Module firmware not found	Perform a firmware update
1E0D	• Job canceled due to a cold restart, warm restart, reset or	• Repeat the request.
	• Module firmware does not exist	• Check if the module is inserted.
1E0E	Static error when the RD_REC SFC was called. Return value RET_VAL of SFC is available for evaluation in SFCERR variable in instance DB.	Load SFCERR variable from instance DB.
1E0F	Static error when the WR_REC SFC was called. Return value RET_VAL of SFC is available for evaluation in SFCERR variable in instance DB.	Load SFCERR variable from instance DB.
1E41	Invalid number of bytes specified at instruction parameter LEN.	Keep to the value range 1 to 400 bytes.

Note

An error message is only output if the ERROR bit (job canceled with error) is also set. In all other cases, the STATUS word is zero.

Calling the SFCERR Variable

You can obtain more information on errors 14 (1E0EH) and 15 (1E0FH) in event class 30 by means of the SFCERR variable.

You can load the SFCERR variable from the instance DB of the corresponding instruction.

Error messages entered in the SFCERR variable can be found in the system functions SFC 58 "WR_REC: Write data record to I/O (Page 6496)" and SFC 59 "RD_REC: Read data record from I/O (Page 6491)".

PtP data link CP 441

Communication instructions

Introduction

Instead of using its own blocks, the CP 441 communicates by means of instructions of the S7-400 automation system

The S7-400 automation system provides a number of instructions which can be called in the user program to initiate and control communication between the CPU and CP 441 communication processor. The instructions are saved to non-volatile memory of the CPU.

Instructions from the automation system S7-400

The table below lists the instructions that the S7-400 automation system provides for communication between the CPU and CP 441.

Table 9-323 Instructions from the automation system S7-400

SFB	Meaning
BSEND	With the instruction BSEND, you can send data from a S7-data range to a communication partner with a destination.
BRCV	With the instruction BRCV, you can receive the data from a communication partner and transfer it to a S7-data range.
FETCH	RK 512 only: With the instruction FETCH, you can fetch the data from a communication partner.
PUT	RK 512 only: With the instruction PUT, you can send the data to the communication partner with dynamic destination addresses..
PRINT	With the instruction PRINT, you can print a message text on a printer with up to four variables.
STATUS	With the instruction STATUS, you can call the status of a communication connection.

Further Information

A detailed description of the instructions from the automation system S7-400 and their settings can be found under: Overview of the S7 communication instructions (Page 7356)

See also

Data transmission from CPU to communication module with BSEND (CP 441-2) (Page 1095)

Data transmission from the communication module to the CPU with BRCV (CP 441-2) (Page 1097)

Fetching data with RK512 (Page 1067)

V24_STAT_441: Reading accompanying signals from the RS232C interface

Description

The V24_STAT_441 instruction reads the RS 232C accompanying signals from an interface of the CP 441 and makes them available to you at the block parameters. The V24_STAT_441 instruction is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program.

Operating principle

The RS 232C accompanying signals are updated each time the instruction is called (cyclic polling). At the V24_STAT_441 instruction, select the interface by setting an ID that represents the "local ID" of one of the connections operating on this interface.

The binary result BR is not affected.

Note

A positive voltage at the RS232 C input signals DSR, CTS, DCD and RI is mapped accordingly to signal status "1" of the input signals DSR_IN, CTS_IN, DCD_IN and RI_IN of the V24_STAT_441 instruction.

Parameters

Parameters	Declaration	Data type	Description
REQ	VAR_INPUT	BOOL	Enables transmission at the rising edge
ID	VAR_INPUT	WORD	Clear communication connection to a communication partner
NDR	VAR_OUTPUT	BOOL	Rising edge indicates that new receive data is available to the user program
ERROR	VAR_OUTPUT	BOOL	Rising edge indicates errors
STATUS	VAR_OUTPUT	WORD	Contains detailed error information or a warning
DTR_OUT	OUTPUT	BOOL	Data terminal ready, CP 441 ready (CP 441 output)
DSR_IN	OUTPUT	BOOL	Data set ready, Communication partner ready (CP 441 input)
RTS_OUT	OUTPUT	BOOL	Request to send, CP 441 ready to send[1] (CP 441 output)

Parameters	Declaration	Data type	Description
CTS_IN	OUTPUT	BOOL	**C**lear **t**o **s**end, Communication partner is ready to receive data from CP 441 (response to RTS = ON of the CP 441) (CP 441 input)
DCD_IN	OUTPUT	BOOL	**D**ata **C**arrier **d**etect, receive signal level (CP 441 input)
RI_IN	OUTPUT	BOOL	**R**ing **I**ndicator, Indication of incoming call (CP 441 input)

Example

Table 9-324 Example of the call of the V24_STAT_441 instruction

STL

```
        CALL FB 5, DB55
                 REQ                 := DB30.DBX10.0
                 ID                  :=W#16#1000
                 NDR                 := DB30.DBX10.1
                 ERROR               := DB30.DBX10.2
                 STATUS              := DB30.DBW20
                 DTR_OUT             := DB30.DBX30.0
                 DSR_IN              := DB30.DBX30.1
                 RTS_OUT             := DB30.DBX30.2
                 CTS_IN              := DB30.DBX30.3
                 DCD_IN              := DB30.DBX30.4
                 RI_IN               := DB30.DBX30.5
```

See also

RS232 mode (CP 34x, 44x, CPU 31xC-2 PtP + ET 200S 1SI) (Page 1048)

V24_SET_441: Writing accompanying signals to the RS232C interface

Description

You can set or reset the interface outputs of CP 441 via the parameter inputs of the V24_SET_441 instruction. The V24_SET_441 instruction is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program.

Operating principle

At the V24_SET_441 instruction, select the interface by setting an ID that represents the "local ID" of one of the connections operating on this interface.

The binary result BR is not affected.

Parameters

Parameters	Declaration	Data type	Description
REQ	VAR_INPUT	BOOL	Enables transmission at the rising edge
ID	VAR_INPUT	WORD	Clear communication connection to a communication partner
DONE	VAR_OUTPUT	BOOL	Signals error-free completion of the job at a rising edge
ERROR	VAR_OUTPUT	BOOL	Rising edge indicates errors
STATUS	VAR_OUTPUT	WORD	Contains detailed error information or a warning
RTS	INPUT	BOOL	Request to send, CP 441 ready to send (Control CP 441 output)
DTR	INPUT	BOOL	Data terminal ready, CP 441 ready (Control CP 441 output)

Example

Table 9-325 Example of the call of the V24_SET_441 instruction

STL

```
        CALL FB 6, DB56
                  REQ         := DB40.DBX10.0
                  ID          :=W#16#1000
                  DONE        := DB40.DBX10.1
                  ERROR       := DB40.DBX10.2
                  STATUS      := DB40.DBW20
                  RTS         := DB40.DBX30.2
                  DTR         := DB40.DBX30.0
```

See also

RS232 mode (CP 34x, 44x, CPU 31xC-2 PtP + ET 200S 1SI) (Page 1048)

SYSTAT error message area

Error-Signaling Area SYSTAT

The error-signaling area SYSTAT is an error area which is available for every interface (ID number). The SYSTAT records all errors/events which can occur during data transmission on an interface.

Note

Because the STATUS request is executed asynchronously to the rest of the requests running at an interface, an SFB with a specific R_ID cannot be assigned to the error messages. This means that although SYSTAT can display which errors have occurred at an interface, it cannot show which SFB call (R_ID number) triggered the error.

Errors/Events

The SYSTAT messages are entered in bytes 2 to 15 of the LOCAL parameter when the STATUS SFB is called. In addition to the error byte (byte 2), the first six errors/events are displayed. Error event 1 is the oldest.

If other error events occur, these cannot be reported until the "old" entries are deleted. The error-signaling area must therefore be deleted in good time. This is done when the STATUS SFB is called.

The errors/events are stored as follows:

- Byte 0 — Operating state of CP (02H for RUN, 05H for defective)
- Byte 1 — Reserved
- Byte 2 — Bit 0 -F Enter error in SYSTAT
  Bit 1 -U Error overflow
  Bit 2 -B Break
- Byte 3 — Reserved
- Byte 4/5 — Event 1
- Byte 6/7 — Event 2
- Byte 8/9 — Event 3
- Byte 10/11 — Event 4
- Byte 12/13 — Event 5
- Byte 14/15 — Event 6

Event classes

Event class	Meaning
1	CP hardware fault
2	Error initializing
3	Incorrect parameter assignment for the instruction

Event class	Meaning
4	The CP detected errors in CP - CPU data traffic
5	Error executing a CPU job
6	Error executing a partner job (for RK 512 only)
7	Send errors
8	Receive errors
9	Received response message frame with error or error message frame from the interconnection partner
10 ($0A_H$)	The CP detected errors in the response message frame of the partner

Event numbers

Error code(W#16 #...)	Description	Remedy
0101	Fault while testing operating system EPROM of CP	CP defective; replace CP.
0102	RAM test of CP errored	
0103	Request interface of CP defective	
0104	No interface submodule inserted	Insert suitable interface submodule for CP.
0105	• Parameter memory defective • Interface submodule unplugged after parameter assignment	Exchange CP or insert suitable interface submodule for CP.
0110	Fault in CP firmware	Switch module off and on again. If necessary, replace module.
0201	No parameters Parameter memory empty or has unknown contents	Load interface parameters.
0208	Parameter assignment and interface submodule incompatible	Check parameters set for interface submodule.
020F	Invalid parameter assignment detected at start of parameter communication. Interface could not be parametered.	Correct invalid parameters and restart.
0210	Total baud rate exceeded	Reduce the baud rates of the two interfaces so that the total baud rate is not exceeded.
0211	Total baud rate back in range	Total baud rate was out of range. It is now back in range after the excursion.
0301	Invalid or no source/destination data type Invalid area (start address, length) Invalid or no DB (e.g. DB 0) or Other data type invalid or missing	Check parameters on CPU and CP and correct if necessary. **RK 512 only:** Partner returns invalid parameters in telegram header. Check parameters on CPU and CP; possibly create block. See request tables for valid data types. **RK 512 only:** Partner provides incorrect parameters in the message frame header.

Error code(W#16 #...)	Description	Remedy
0302	Area too short	Check parameters on CPU and CP; possibly check block/area. **RK 512 only:**Partner provides incorrect parameters in the message frame header.
0303	Area cannot be accessed	Check parameters on CPU and CP. Obtain the permissible start addresses and lengths from the request tables. **RK 512 only:** Partner provides incorrect parameters in the message frame header.
0401	CP cannot accept requests (overload)	In your user program, reduce number of requests called concurrently for CP.
0402	CP cannot process request type	Check if the system function blocks you have called in user program are valid for CP.
0403	Incorrect, unknown or illegal data type	Check program, e.g. for incorrect parameters of SFB.
0407	An error has occurred during data transmission between the CPU and the CP. Data cannot be received because there is no access to the CPU destination frame or the CPU destination frame does not exist or is too short. Writing data to the CPU destination frame or reading data from the CPU source frame during a CPU stop when assigning parameters is not permitted.	Check the destination frame on the CPU. Check the parameter assignment for "Response to a CPU Stop".
0408	• **Only for 3964R and ASCII drivers** • A temporary error has occurred during data transmission between the CPU and the CP (receive). The request is queued for repetition because the CPU is temporarily overloaded or the receive block (BRCV) is requested too infrequently or the receive block has been temporarily blocked.	• Reduce the number of communication requests. Call the receive block more often. Check if the receive block has been blocked too long.
0409	• **Only for 3964R and ASCII drivers** • An error has occurred during data transmission between the CPU and the CP. Data reception is not possible. After multiple attempts (see (04)07H), the request has canceled after 10 seconds because the receive block (BRCV) could not be called or is blocked.	• Check if the receive block has been called or has been blocked.
040B	Error during data transmission between CPU and CP, because • no connection has been configured • no receipt possible via configured connection	• Configure the connection in "NetPro" • Enter in "NetPro" under "Object Properties Connection) as communication direction: – 2: Partner → local or – 3: Local ↔ partner

Error code(W#16 #...)	Description	Remedy
0501	Current request aborted as a result of CP restart.	No remedy is possible at POWER ON. When changing the parameters of the CP in the programming device, before writing an interface you should ensure there are no more requests running from the CPU.
0502	Job not permitted in this CP operating mode (e.g., device interface not parameterized)	Set the parameters for the device interface.
0503	Wrong time, incorrect format	Check the time parameters.
0505	**Only for printer drivers:** System data block with message texts not available on the CP	Use the parameter assignment software to configure the message text and then carry out a restart.
0506	**Only for printer drivers:** Message text not available	Use the parameter assignment software to configure the message texts and then carry out a warm restart.
0507	**Only for printer drivers:** Message text too long	Edit the message text to reduce it to a length of fewer than 150 characters (or no more than 250 characters if it contains variables).
0508	**Only for printer drivers:** Too many conversion instructions	You have configured more conversion instructions than variables. Conversion instructions without associated variables will be ignored.
0509	**Only for printer drivers:** Too many variables	You have configured more variables than conversion instructions. Variables without associated conversion instructions will not be output.
050A	**Only for printer drivers:** Unknown conversion instruction	Check the conversion instruction. Undefined or unsupported conversion instructions are replaced in the printout with ******.
050B	**Only for printer drivers:** Unknown control instruction	Check the control instruction. Undefined or unsupported control instructions will be ignored. The control instruction will not be output as text either.
050C	**Only for printer drivers:** Conversion instruction not executable	Check the conversion instruction. Conversion instructions that cannot be executed are output in the expression in accordance with the defined width and the valid remainder of the conversion instruction or the standard representation with * characters.

Error code(W#16 #...)	Description	Remedy
050D	**Only for printer drivers:** Width in conversion instruction too small or too great	Correct the specified width of the variable in the conversion instruction on the basis of the variable's maximum possible number of characters in text-based display types (A, C, D, S, T, Y, Z). Only as many characters as will fit in the specified width appear in the printout; the text is truncated to this width. In all other cases, * characters are output corresponding to the width.
050E	**Only for ASCII drivers:** An error occurred while sending. The defined end-of-text characters did not occur within the maximum allowed length or in the case of automatic appending, the maximum allowed transmission length was exceeded.	Extend the end-of-text characters in the transmission buffer at the desired point or select a shorter telegram length for automatic appending.
050F	Number of requests that can be processed simultaneously too great	Change your STEP 7 program so that fewer requests can run simultaneously.
0510	Area occupied (resource)	Repeat the request.
0511	Length not permissible for this request type	Divide up the data to be transmitted into several requests.
0512	**RK 512 only:**Mismatch between SFB's source and destination parameters.	Obtain the permissible values from the request tables.
0513	Data type error (DB ...): Unknown or impermissible data type (e.g. DE) **RK 512 only:** Mismatch between SFB's source and destination data types.	Obtain the permissible data types and their combinations from the request tables.
0514	Specified start addresses too high for desired data type or start address or DB/DX number too low.	Obtain from the request tables the permissible start addresses and DB/DX numbers that can be specified in the program.
0515	**RK 512 only:** Wrong bit number specified for coordination flag.	Permissible bit numbers: 0 to 7
0516	**RK 512 only:** Specified CPU too high.	Permissible CPU numbers: none, 1, 2, 3 or 4
0517	An error occurred while receiving. The receive telegram is longer than 4 KB or is longer than the defined "fixed receive length" or the receive telegram does not fit into the destination frame.	Reduce the length of your connection partner's telegram or increase the length of your receive DB.
0518	Transmission length at sending too great (>4 KB)	RK 512 only: Obtain the permissible lengths from the request tables. Split the request up into several shorter requests.
0519	CP in wrong mode for PLC request	Check if the addressed interface is parametered.

Error code(W#16 #...)	Description	Remedy
051A	**RK 512 only:** Error sending a command telegram An associated procedure number has just been entered in STATUS.	See the remedy for the previous error number.
051B	**Only for printer drivers:** Precision invalid	Correct the specified precision in the conversion instruction. The precision is always initialized with a dot prefix to identify and limit the width (for example, ".2" to output the decimal point and two decimals.) Precision is only relevant to display types A, D, F and R. It is ignored otherwise.
051C	**Only for printer drivers:** Variable invalid (variable length incorrect/incorrect type)	Correct the specified send variable.
0601	Error in 1st command byte (not 00 or FFH)	Header layout error at partner. You can interconnect an interface tester in the transmission line to debug the partner device.
0602	Error in 3rd command byte (not A, 0 or FFH)	Header layout error at partner. You can interconnect an interface tester in the transmission line to debug the partner device.
0603	Error in 3rd command byte in the case of continuation telegrams (command not as for 1st telegram)	Header layout error at partner. You can interconnect an interface tester in the transmission line to debug the partner device.
0604	Error in 4th command byte (command letter incorrect)	Header layout error at partner or a command combination has been requested that is not permitted at the CP. Check the permissible commands. You can interconnect an interface tester in the transmission line to debug the partner device.
0605	Error in 4th command byte in the case of continuation telegrams (command not as for 1st telegram)	Header layout error at partner. You can interconnect an interface tester in the transmission line to debug the partner device.
0606	Error in 5th command byte (DB number not permissible)	Obtain from the request tables the permissible DB numbers, start addresses or lengths.
0607	Error in 5th or 6th command byte (start address too high)	Obtain from the request tables the permissible DB numbers, start addresses or lengths.
0608	Error in 7th or 8th command byte (impermissible length)	Obtain from the request tables the permissible DB/DX numbers, start addresses or lengths.
0609	Error in 9th and 10th command byte (coordination flag for this data type impermissible or bit number too high)	Header layout error at partner. Find out from the request tables when a coordination flag is permitted.

Error code(W#16 #...)	Description	Remedy
060A	Error in 10th command byte (CPU number not permitted)	Header layout error at partner.
060B	SEND telegram was longer/shorter than expected (more/less data received than announced in telegram header).	Correction required at the partner
060C	GET command telegram received with user data	Correction required at the partner
060D	The CP received a telegram during an invalid operating mode. • Receive connection between CPU and CP not set up or not yet correctly set up • CP startup is not fully completed. • Parameters for the interface are currently being assigned	• Check if the addressed connection has been assigned the correct parameters. • This error message can occur only during CP startup. Repeat the request. • This is a temporary error. Repeat the request.
060E	Synchronous fault of partner • New (continuation) command telegram received before response telegram sent. • 1. 1st command telegram expected and continuation telegram came. • Continuation command telegram expected and 1st telegram came	This error may be reported after your own programming device is restarted in the case of long telegrams or when the partner is restarted. These cases represent normal system start-up characteristics. The error can also occur during operation as a consequence of error statuses only recognized by the partner. Otherwise, you have to assume an error on the part of the partner device. The error may not occur in the case of requests <128 bytes.
060F	DB locked by coordination function	In local program: After processing of the last transmission data, enable the last receive block with "EN". In partner program: Repeat the request
0610	Telegram received too short (length <4 bytes in the case of continuation or response telegrams or <10 bytes in the case of command telegrams)	You can interconnect an interface tester in the transmission line to debug the partner device.
0611	Telegram length and length specified in telegram header are not the same.	You can interconnect an interface tester in the transmission line to debug the partner device.
0612	Error sending the (continuation) response telegram. An associated procedure error number has been entered in STATUS immediately beforehand.	See remedy for the error number entered immediately beforehand in STATUS.
0701	Sending of the first repetition: • An error was detected when transmitting the telegram or • The partner requested a repetition by means of a negative acknowledgment character (NAK).	A repetition is not an error, but it can indicate that there is interference on the data link or the partner device has malfunctioned. If the message frame has not been transmitted after the maximum number of repetitions, an error number describing the first error that occurred is output.

Error code(W#16 #...)	Description	Remedy
0702	Error establishing connection: • After STX was sent, NAK or another code (except DLE or STX) was received or • The response came too early or • An initialization conflict occurred	Check for malfunction at partner device, possibly by using interface test device switched into the transmission line.
0703	Acknowledgment delay time (QVZ) exceeded: After STX was sent, partner did not respond within the acknowledgment delay time.	The partner device is too slow or not ready to receive or there is a break in the transmission line, for example. You can interconnect an interface tester in the transmission line to debug the partner device.
0704	Termination by partner: One or more codes were received from the partner during sending	Check if the partner also indicates an error; possibly it has not received all of the transmitted data (for example, due to an interrupted data link) or because fatal errors are pending or the characteristics of the partner device is faulty. If necessary, use an interface test device switched into the transmission line to check.
0705	Negative acknowledgment when sending	Check if the partner also indicates an error; possibly it has not received all of the transmitted data (for example, due to an interrupted data link) or because fatal errors are pending or the characteristics of the partner device is faulty. If necessary, use an interface test device switched into the transmission line to check.
0706	End-of-connection error: • Partner rejected telegram at end of connection with NAK or a random string (except for DLE) or • Acknowledgment characters (DLE) received too early.	Check if the partner also indicates an error; possibly it has not received all of the transmitted data (for example, due to an interrupted data link) or because fatal errors are pending or the characteristics of the partner device is faulty. If necessary, use an interface test device switched into the transmission line to check.
0707	Acknowledgment delay time exceeded at end of connection or response monitoring time exceeded after a send telegram: After connection termination with DLE ETX, no response received from partner within acknowledgment delay time.	Partner device too slow or faulty. If necessary, use an interface test device switched into the transmission line to check.
0708	ASCII Driver and printer driver only: The wait time for XON or CTS = ON has expired.	The communication partner is faulty, too slow or has been taken offline. Check the communication partner; you may need to change the parameter assignment.
0709	Connection setup not possible. Number of permitted setup attempts exceeded.	Check the interface cable or the transmission parameters.

Error code(W#16 #...)	Description	Remedy
070A	The data could not be transmitted. The permitted number of transfer attempts was exceeded.	Check the interface cable or the transmission parameters.
0801	Expectation of the first repetition: An error was detected on receiving a telegram and the CP requested repetition from the partner via a negative acknowledgment (NAK).	A repetition is not an error, but it can indicate that there is interference on the data link or the partner device has malfunctioned. If the message frame has not been transmitted after the maximum number of repetitions, an error number describing the first error that occurred is output.
0802	**Error establishing connection:** • In idle mode, one or more random codes (other than NAK or STX) were received or • After an STX was received, the partner sent more characters without waiting for the response DLE. **After POWER ON of the partner:** • While partner is being switched on, the CP receives an undefined character.	Check for malfunction at partner device, possibly by using interface test device switched into the transmission line.
0805	Logical error while receiving: After DLE was received, a further random code (other than DLE or ETX) was received.	Check if the partner always duplicates the DLE in the telegram header and data string or the connection is terminated with DLE ETX. Check for malfunction at partner device, possibly by using interface test device switched into the transmission line.
0806	Character delay time (ZVZ) exceeded: • Two successive characters were not received within character delay time or • 1. character after sending of DLE while establishing connection was not received within the character delay time.	Partner device too slow or faulty. Interconnect an interface tester in the transmission line to see if this is the case.
0808	Block check character (BCC) error (only in the case of RK 512 with the 3964R procedure and the 3964R procedure) The value of BCC calculated internally does not match the BCC received by the partner when the connection was terminated.	Check if there is a serious problem with the connection. In this case, error codes of the event class 8/event number 12 sometimes occur. Check for malfunction at partner device, possibly by using interface test device switched into the transmission line.
080A	There is no free receive buffer available: After receipt of STX, there was no empty receive buffer available to the procedure at connection setup and after an additional waiting time	Increase the call rate for the receive instruction in the user program.

Error code(W#16 #...)	Description	Remedy
080C	Transmission error: A transmission error (parity error, stop bit error, overflow error) was detected. If a corrupted character is received in idle mode, the error is reported immediately so that disturbances on the transmission line can be detected early. **Only in the case of RK 512 and 3964(R):** If this occurs during send or receive operation, repetitions are initiated.	Disturbances on the data link cause telegram repetitions, thus lowering user data throughput. The risk of undetected error increases. Change your system setup or cable wiring. Check that the settings for baud rate, parity and number of stop bits are the same on both devices.
080D	BREAK The connection line (receive line) to the partner device is interrupted	Establish the connection between the devices or switch on the partner device. In the case of TTY, Check if there is a current loop in the idle state.
0812	**Only for ASCII drivers:** More characters were received after the CP had sent XOFF or set CTS to OFF.	Reset the parameters for the communication partner or read data from CP more quickly.
0815	Discrepancy between settings for transfer attempts at CP and communication partner.	Set the same number of transfer attempts at communications partner as at CP. You can interconnect an interface tester in the transmission line to debug the partner device.
0816	• The length of a received telegram was longer than the length agreed upon or • the length of the parametered receive buffer (with CP 441 only) is too short.	• A correction is necessary at the partner or • the length of the receive buffer (with CP 441 only) must be enlargened
0818	**Only for ASCII drivers:** DSR = OFF or CTS = OFF	The partner has switched the DSR or CTS signal to "OFF" before or during a transmission. Check the partner's control of the RS 232C accompanying signals.
0902	**RK 512 only:** Memory access error at partner (memory does not exist) With SIMATIC S5 as partner: • Incorrect area at status word or • Data area does not exist (except DB/DX) or • Data area too short (except DB/DX)	Check that the partner has the desired data area and that it is big enough or check the parameters of the called system function block. Check the specified length at the system function block.

Error code(W#16 #...)	Description	Remedy
0903	**RK 512 only:** DB/DX access error at the partner (DB/DX does not exist or is too short) With SIMATIC S5 as partner: • DB/DX does not exist or • DB/DX too short or • DB/DX number impermissible Permissible source area for GET request exceeded	Check that the partner has the desired data area and that it is big enough or check the parameters of the called system function block. Check the specified length at the system function block.
0904	**RK 512 only:** Partner returns "Request type not permitted".	Partner malfunction, because a system command is never issued from the CP 441.
0905	**RK 512 only:** Error at partner or at SIMATIC S5 as partner: • Source/destination type not permissible or • Memory error in partner programmable controller or • Error notifying CP/CPU at the partner or • Partner programmable controller is in STOP state	Check if the partner can transmit the desired data type. Check the structure of the hardware at the partner. Set the partner programmable controller to RUN.
0908	**RK 512 only:** Partner detecting synchronization error: Telegram sequence error.	This error occurs at restart of your own programmable controller or of the partner. This represents normal system start-up characteristics. You do not need to correct anything. The error is also conceivable during operation as a consequence of previous errors. Otherwise, you can assume an error on the part of the partner device.
0909	**RK 512 only:** DB/DX disabled at partner by coordination flag	In partner program: After processing of the last transmission data, reset the coordination flag. In own program: Repeat the request.
090A	**RK 512 only:** Error detected by partner in telegram header: 3rd command byte in header is incorrect	Check if the error is the result of disturbances or of a malfunction at the partner. Interconnect an interface tester in the transmission line to see if this is the case.
090B	**RK 512 only:** Error in telegram header: 1. or incorrect command byte 4 in header	Check if the error is the result of disturbances or of a malfunction at the partner. Interconnect an interface tester in the transmission line to see if this is the case.
090C	**RK 512 only:** Partner detects incorrect telegram length (total length).	Check if the error is the result of disturbances or of a malfunction at the partner. Interconnect an interface tester in the transmission line to see if this is the case.

Error code(W#16 #...)	Description	Remedy
090D	**RK 512 only:** Partner has not yet restarted.	Restart the partner programmable controller or set the mode selector on the CP or CPU to RUN.
090E	**RK 512 only:** Unknown error number received in response telegram.	Check if the error is the result of disturbances or of a malfunction at the partner. Interconnect an interface tester in the transmission line to see if this is the case.
0A01	**RK 512 only:** Synchronization error of partner, because: • Response telegram without request • Response telegram received before continuation telegram sent • Continuation response telegram received after an initial telegram was sent • A first response telegram was received after a continuation telegram was sent	This error is reported after your own programming device is restarted in the case of long telegrams or when the partner is restarted. This represents normal system start-up characteristics. You do not have to correct anything. The error can also occur during operation as a consequence of error statuses only recognized by the partner. Otherwise, you can assume an error on the part of the partner device. The error may not occur in the case of requests <128 bytes.
0A02	**RK 512 only:** Error in the structure of the received response telegram (1st byte not 00 or FF)	Check for malfunction at partner device, possibly by using interface test device switched into the transmission line.
0A03	**RK 512 only:** Received response telegram has too many data or not enough data.	Check for malfunction at partner device, possibly by using interface test device switched into the transmission line.
0A04	**RK 512 only:** Response telegram for SEND request arrived with data.	Check for malfunction at partner device, possibly by using interface test device switched into the transmission line.
0A05	**RK 512 only:** No response telegram from partner within monitoring time.	Is the partner a slow device? This error is also often displayed as a consequence of a previous error. For example, procedure receive errors (event class 8) can be displayed after a GET telegram was sent. Reason: As a result of disturbances, the response telegram could not be received and the monitoring time elapsed. This error can also occur if a restart was carried out at the partner before it could respond to the most recently received GET telegram.
0A06	**RK 512 only:** Received response telegram after GET request has not enough data.	Check for malfunction at partner device, possibly by using interface test device switched into the transmission line.

Modbus slave (RTU)

Modbus instructions

Introduction

The Modbus masters do not need separate Modbus instructions for communication.

Instructions of CP 341

The data transfer between CP 341 and the CPU is handled using the instructions P_SND_RK and P_RCV_RK.

Table 9-326 Instructions of CP 341

Instruction	Meaning
P_RCV_RK	Using the P_RCV_RK instruction, you can receive data from a communication partner and save the data to a data block or provide data for the communication partner.
P_SND_RK	Using the P_SND_RK instruction, you can send the entire area or a section of a data block to a communication partner or fetch data from a communication partner.

Instructions from the automation system S7-400

The data transfer between CP 441 and the CPU is handled using the instructions BSEND and BRCV.

Table 9-327 Instructions from the automation system S7-400

SFB	Meaning
BSEND	With the instruction BSEND, you can send data from a S7-data range to a communication partner with a destination.
BRCV	With the instruction BRCV, you can receive the data from a communication partner and transfer it to a S7-data range.

MODB_341: Modbus slave instruction for CP 341

Description

The MODB_341 instruction and associated driver enable a communication connection between a Modbus master control system and the CP 341 communication module as "Modbus-capable" slave system.

Operating principle

In the user program, the MODB_341 instruction is is called in the cyclic routine. The instruction uses the associated IDB_MODB_341 instance DB as the working area.

The MODB_341 instruction initializes the CP and executes the Modbus functions which the driver cannot execute by itself.

The MODB_341 instruction must be performed following every cold or warm restart of the CPU. Initialization is triggered by a rising edge at the input CP_START. The instruction deletes the instance DB, calls system instruction SZL_LESEN to read the I, Q, F, T and C address areas from the CPU and saves these to the instance DB. This procedure allows you to check whether the write requests from the Modbus master system are out of range.

A SEND job reports the number of the instance DB and the currently successful initialization progress to the CP.

Once the SEND job is completed without errors, output CP_START_OK is set and initialization of the instruction is completed.

Parameters

Parameters	Declaration	Data type	Description
LADDR	INPUT	Int	Base address of the CP Use HW Config assignment
START_TIMER	INPUT	Timer	Timer for start of monitoring time
START_TIME	INPUT	S5Time	Time value for start of monitoring time
OB_MASK	INPUT	BOOL	Mask I/O access errors, delay interrupts FALSE: I/O access errors are not masked. TRUE: I/O access errors are masked and interrupts are delayed.
CP_START	INPUT	BOOL	Start instruction initialization
CP_START_FM	INPUT	BOOL	Initialization is triggered with a rising edge at CP_START.
CP_START_NDR	OUTPUT	BOOL	Info: Write request from the CP
CP_START_OK	OUTPUT	BOOL	Startup completed without error TRUE: The initialization job could be completed without error before the monitoring time elapsed.
CP_START_ERROR	OUTPUT	BOOL	Startup canceled with error TRUE: The initialization job could not be completed without error even after the monitoring time had elapsed.
ERROR_NR	OUTPUT	Word	Error number Allocation, see ERROR_NR, ERROR_INFO parameters (Page 7657)
ERROR_INFO	OUTPUT	Word	Additional error info Allocation, see ERROR_NR, ERROR_INFO parameters (Page 7657)

How to Handle an Error

If the SEND job is terminated with errors, CP_START is reset and CP_START_ERROR is set.

If initialization ends with errors, Modbus communication is not possible. All requests from the Modbus master system are answered with an exception code frame.

After power on, the CP needs several seconds to complete the hardware and memory test before it is ready for operation. The initialization attempts carried out by the MODB_341 instruction within this period are terminated with error. For this reason, the instruction repeats its initialization job several times within the monitoring time.

If the initialization is successfully completed within the configured START-TIME of the START-TIMERS, CP_START_OK is set. If the initialization job cannot be completed without errors even after the monitoring time has elapsed, CP_START_ERROR is set.

MODB_441: Modbus slave instruction for CP 441

Description

The MODB_441 instruction and associated driver enable a communication connection between a Modbus master control system and the CP 441-2 communication module as "Modbus-capable" slave system.

Operating principle

In the user program, the MODB_441 instruction is is called in the cyclic routine. The instruction uses the associated IDB_MODB_441 instance DB as the working area.

The Modbus function codes 01, 02, 03, 04, 06 and 16 are processed directly by the CP. For function codes 05 and 15, the instruction handles the bit-by-bit entry of the data in the SIMATIC memory range.

The MODB_441 instruction initializes the CP and executes the Modbus functions which the driver cannot execute by itself.

The MODB_441 instruction must be performed following every cold or warm restart of the CPU. Initialization is triggered by a rising edge at the input CP_START. The instruction deletes the instance DB, calls system instruction SZL_LESEN to read the I, Q, F, T and C address areas from the CPU and saves these to the instance DB. This procedure allows you to check whether the write requests from the Modbus master system are out of range.

A SEND job reports the number of the instance DB and the currently successful initialization progress to the CP.

Once the SEND job is completed without errors, output CP_START_OK is set and initialization of the instruction is completed.

Parameters

Parameters	Declaration	Data type	Description
ID	INPUT	Word	Connection ID Connection ID from configuration of the data link.
START_TIMER	INPUT	Timer	Timer for start of monitoring time
START_TIME	INPUT	S5Time	Time value for start of monitoring time
STATUS_TIMER	INPUT	Timer	Read Timer for SYSTAT
STATUS_TIME	INPUT	S5Time	Read time interval for SYSTAT
OB_MASK	INPUT	BOOL	Mask I/O access errors, delay interrupts FALSE: I/O access errors are not masked. TRUE: I/O access errors are masked and interrupts are delayed.
CP_START	INPUT	BOOL	Start instruction initialization
CP_START_FM	INPUT	BOOL	Initialization is triggered with a rising edge at CP_START.
CP_START_NDR	OUTPUT	BOOL	Info: Write request from the CP
CP_START_OK	OUTPUT	BOOL	Startup completed without error TRUE: The initialization job could be completed without error before the monitoring time elapsed.
CP_START_ERROR	OUTPUT	BOOL	Startup canceled with error TRUE: The initialization job could not be completed without error even after the monitoring time had elapsed.
ERROR_NR	OUTPUT	Word	Error number Allocation, see ERROR_NR, ERROR_INFO parameters (Page 7657)
ERROR_INFO	OUTPUT	Word	Additional error info Allocation, see ERROR_NR, ERROR_INFO parameters (Page 7657)

How to Handle an Error

If the SEND job is terminated with errors, CP_START is reset and CP_START_ERROR is set.

If initialization ends with errors, Modbus communication is not possible. All requests from the Modbus master system are answered with an exception code frame.

After power on, the CP needs several seconds to complete the hardware and memory test before it is ready for operation. The initialization attempts carried out by the MODB_441 instruction within this period are terminated with error. For this reason, the instruction repeats its initialization job several times within the monitoring time.

If the initialization is successfully completed within the configured START-TIME of the START-TIMERS, CP_START_OK is set. If the initialization job cannot be completed without errors even after the monitoring time has elapsed, CP_START_ERROR is set.

ERROR_NR, ERROR_INFO parameters

Error classes

Error numbers from ... to	Description	Meaning
1 ... 9	Instruction and CP initialization error	Error numbers 1 to 9 indicate that initialization was terminated with errors. The CP_START_ERROR parameter is set to 1. Modbus communication to the master system is not possible.
10 ... 19	Error executing a function code	Error numbers 10 to 19 indicate that an error occurred during the processing of a function code. The CP transmitted an invalid execution job to the communication instruction. The error is also reported to the driver. Subsequent execution jobs continue to be processed.
90 ... 99	Other faults/errors	A processing error has occurred. The error is not reported to the driver. Subsequent execution jobs continue to be processed.

ERROR_NR, ERROR_INFO parameters

ERROR_ No (decimal)	ERROR_INFO	Error Text	Remedy
0	0	No error	
1	SFC 51->RET_VAL	Error reading the system status list with SFC 51.	Analyze RET_VAL in ERROR_INFO; eliminate the cause.
2		Timeout of CP initialization or error during CP initialization	Check if the CP interface was assigned the "Modbus slave" protocol in the parameters. Analyze ERROR_INFO.
	P_SND_RK -> STATUS	Error in P_SND_RK job	Check module address LADDR at the communication instruction.
	SFB 12->STATUS	Error in BSEND job	Check the connection ID in the communication instruction.
10	Processing Code	The driver transferred an invalid processing function to the communication instruction.	CP restart (Power_On)
11	Start address	The driver transferred an invalid start address to the communication instruction.	Check Modbus address of Modbus master system.

ERROR_ No (decimal)	ERROR_INFO	Error Text	Remedy
12	Register number	The driver transferred an invalid number of registers to the communications instruction: Number of registers = 0.	Check number of registers of Modbus master system, if required restart CP (Power_ON)
13	Register number	The driver transferred an invalid number of registers to the communications instruction: Number of registers > 128.	Check number of registers of Modbus master system, if required restart CP (Power_ON)
14	Memory bits M - End Address	Attempted access to SIMATIC memory area "memory bits" in excess of range end. Attention: Range length in SIMATIC CPU is CPU type-dependent.	Reduce Modbus start address and/or access length in Modbus master system.
15	Outputs Q – End Address	Attempted access to SIMATIC memory area "outputs" in excess of range end. Attention: Range length in SIMATIC CPU is CPU type-dependent.	Reduce Modbus start address and/or access length in Modbus master system.
16	Timers T – End Address	Attempted access to SIMATIC memory area "timers" in excess of range end. Attention: Range length in SIMATIC CPU is CPU type-dependent.	Reduce Modbus start address and/or access length in Modbus master system.
17	Counters C – End Address	Attempted access to SIMATIC memory area "counters" in excess of range end. Attention: Range length in SIMATIC CPU is CPU type-dependent.	Reduce Modbus start address and/or access length in Modbus master system.
18	0	The driver transferred an invalid SIMATIC memory area to the communication instruction.	If required, restart CP (Power_On)
19		Error during access to SIMATIC I/Os.	Check if the required I/Os exist and are error-free.
90	P_SND_RK -> STATUS	Error transmitting an acknowledgment message to the driver using the P_SND_RK instruction	Analyze the STATUS information
	SFB 12->STATUS	Error transmitting an acknowledgment message to the driver using SFB 12 BSEND	Analyze the STATUS information
91	SFB 22->STATUS	Error reading SYSTAT with SFB 22 (STATUS)	Analyze the STATUS information
92	P_RCV_RK -> STATUS	Error executing a RECEIVE/FETCH call using the P_RCV_RK instruction	Analyze P_RCV_RK-STATUS

SYSTAT error message area

Event classes

Event class	Meaning
5	Error executing a CPU job
8	Receive errors
14 ($0E_H$)	General processing error

STATUS parameter

Error code(W#16#...)	Description	Remedy
0518	Transmission frame length exceeded (> 4 KB) or insufficient transmission frame length for SEND.	Check the communication instruction; reload it if necessary.
0806	Character delay time ZVZ exceeded	Error by the partner device or repair disruptions on the transmission line.
080C	A character transmission error was detected (parity error, overrun error or stop bit error (frame)).	Check if the transmission lines are affected by interference. Modify the system configuration or the cable routing. Check the consistency of the settings for the protocol parameters baud rate, number of data bits, parity and number of stop bits at the CP and link partner.
080D	BREAK The receive line to the partner device is disrupted.	Establish the connection between the devices or switch on the partner device. In TTY mode, check if there is a line current flow in idle state. For an RS422/485(X27) connection, check or modify the default configuration of the 2-wire receive line R(A) and R(B).
0830	Broadcast is not allowed with this function code.	The Modbus master system can use the broadcast function only for the function codes enabled for this purpose.
0831	Function code received is invalid.	This function code cannot be used with this driver.
0832	Maximum number of bits or registers exceeded or the number of bits cannot be divided by the value 16 for access to the SIMATIC memory area for timers and counters.	Limit the maximum number of bits to 2040 and of the registers to 127. Access to SIMATIC timers/counters only in 16-bit intervals only.
0833	The number of bits or registers for function code FC 15/16 and frame element byte_count do not match.	Adjust the number of bits, number of registers or the byte_count.
0834	Illegal bit coding for "Set bit / Reset bit" recognized.	Always use the codes 0000 hex or FF00 hex for FC 05.
0835	Invalid diagnostics subcode (≠ 0000 hex) detected for function code FC 08 "Loop back test".	Always use subcode 0000 hex for FC 08.

Error code(W#16#...)	Description	Remedy
0836	The internally calculated value of the CRC 16 checksum does not match the CRC checksum received.	Check the calculation of the CRC checksum at the Modbus master system.
0837	Message frame sequence error: The Modbus master system sent a new request message frame before the driver transmitted the last response message frame.	Increase the timeout for the slave response message frame at the Modbus master system.
0E01	Error during initialization of the driver-specific SCC process.	Reassign parameters of driver and reload.
0E02	Error during startup of driver: Wrong SCC process active (SCC driver). The driver cannot function with this SCC driver.	Reassign parameters of driver and reload.
0E03	Driver startup error: Incorrect data transfer process is active (interface to the SFBs). The driver does not work with this data transfer process.	Reassign parameters of driver and reload.
0E04	Driver startup error: Invalid interface module. The driver cannot be operated with the configured interface module.	Check and correct the parameter assignment.
0E05	Error with driver dongle: No dongle plugged in, or inserted dongle is faulty. The driver is not ready to run.	Check if a driver dongle is plugged into the CP. If the inserted dongle is faulty, replace it with a correct dongle.
0E06	Error with driver dongle: The dongle has no valid content. The driver is not ready to run.	Obtain a correct dongle from the Siemens office which supplied you with the driver.
0E10	Internal error procedure: default branch in Send automatic device.	CP restart (Power_On)
0E11	Internal error procedure: default branch in Receive automatic device.	CP restart (Power_On)
0E12	Internal error active automatic device: Default branch	CP restart (Power_On)
0E13	Internal error passive automatic device: Default branch	CP restart (Power_On)
0E20	This connection requires 8 data bits. The driver is not ready for operation.	Correct the parameter assignment of the driver.
0E21	The configured multiplication factor for the character delay time is outside the range from 1 to 10. The driver is operating with default setting 1.	Correct the parameter assignment of the driver.
0E22	The operating mode set for the driver is illegal. Specify "Normal" or "Interference Suppression". The driver is not ready for operation.	Correct the parameter assignment of the driver.
0E23	Parameter assignment with invalid value for the slave address. Slave address 0 is invalid. The driver is not ready for operation.	Correct the parameter assignment of the driver.
0E24	Invalid limits were configured for write access. The driver is not ready.	Correct the parameter assignment of the driver.

Error code(W#16#...)	Description	Remedy
0E25	An invalid "from/to" combination was configured in the specification of the "Conversion of the Modbus addresses with function code FC 01, 05, 15" area. (Memory bits, outputs, timers, counters area). The driver is not ready.	Correct the parameter assignment of the driver.
0E26	An invalid "from/to" combination was configured in the specification of the "Conversion of the Modbus addresses with function code FC 02" area. (Memory bits, inputs area). The driver is not ready.	Correct the parameter assignment of the driver.
0E27	An overlap was configured in the specification of the "Conversion of the Modbus addresses with function code FC 01, 05, 15" area. (Memory bits, outputs, timers, counters area). The driver is not ready.	Correct the parameter assignment of the driver.
0E28	An overlaps was configured in the specification of the "Conversion of the Modbus addresses with function code FC 02" area. (Memory bits, inputs area). The driver is not ready.	Correct the parameter assignment of the driver.
0E2E	An error occurred when reading the interface parameter file. The driver is not ready to run.	Restart CP (Power_On).
0E30	Internal error during data transmission to CPU: Unexpected acknowledgment Passive.	Can be ignored if it happens intermittently.
0E31	Timeout during data transfer to the CPU.	Check CP-CPU interface.
0E32	Error occurred during data transmission to CPU with RCV: Exact failure reason (detailed error) is in SYSTAT before this entry.	Check CP-CPU interface.
0E33	Internal error during data transfer to the CPU: Illegal status of automatic device	Check CP-CPU interface.
:		
0E38	An error has occurred accessing SIMATIC areas "Memory bit, output, timers, counters, input" with the function code FC 01 or FC 02: For example, input not available or attempt to read beyond the end of the area.	Check whether the addressed SIMATIC area exists and that no attempt was made to access beyond the end of the area.
0E39	An error has occurred accessing the SIMATIC "Data block" area with the function code FC 03, 04, 06, 16: Data block not available or too short.	Check if the addressed data block exists and is sufficiently long.
0E3A	Error executing a write job with function codes FC 05, 15: Instance DB of the Modbus instruction does not exist or is of insufficient length.	Check if the instance DB assigned in the parameters of the Modbus communications instruction exists and whether it is of sufficient length.
0E3B	Timeout (waiting for acknowledgement) in the execution of a write job by the Modbus communication instruction.	Check the connection configuration and the CP-CPU interface (SFB SEND); if necessary, reload the Modbus communication instruction.
0E3C	Illegal job with this driver.	Only SFB SEND, STATUS (CP 441-2 only) are permitted.

Error code(W#16#...)	Description	Remedy
0E50	In the word-based SIMATIC timers/counters areas, the resulting residual bit number from the Modbus address is ≠ 0.	Only use Modbus addresses which result in valid bit numbers.
0E51	The received Modbus address is outside the configured "from/to" areas.	Only use addresses as address information in the request message frame that have previously been defined in the configuration.
0E52	• A SIMATIC area limit was exceeded during an access attempt by the Modbus master system: Resulting DB number < 1 • Write access to an unreleased area (configuration) • Write access to the instance DB of the communication instruction	Access area limited to valid SIMATIC memory areas.
0E53	A SIMATIC area limit was exceeded during an access attempt by the Modbus master system: For example, overflow from the formation of the resulting DB no. (> 65535)	Access area limited to valid SIMATIC memory areas.
0E54	Access beyond the configured area end or access beyond the SIMATIC area end.	Access area limited to valid SIMATIC memory areas.
0E55	Write access to this SIMATIC memory area is not allowed.	Write only to the SIMATIC memory bits, outputs data areas.
0E56	Data link operation not possible because the communication instruction is inactive.	Call the Modbus communication instruction at cyclic intervals in the STEP 7 user program. If necessary, re-initialize the communication instruction.
0E57	An error has occurred executing the Modbus function code in the communication instruction.	Analyze the exact cause as described in section "Auto-Hotspot".

ET 200S serial interface

Notes on use of an S7-1500

S_SEND / S_RCV / S_USSI / S_MODB on S7-1500

Note the following when using the S_SEND, S_RCV, S_USSI or S_MODB instructions of the ET 200S 1SI distributed I/O on an S7-1500:

- S_SEND
  Data type of the parameter DB_NO: UInt.
  Only data from non-optimized memory areas can be sent. Keep this in mind during assignment of the DB_NO, DBB_NO and LEN parameters.
- S_RCV
  Data type of the parameter DB_NO: UInt.
  Only data from non-optimized memory areas can be received. Keep this in mind during assignment of the DB_NO, DBB_NO and LEN parameters.

- S_USSI
  - Data type of the parameters DBND, DBPA and DBCP: UInt
    Invalid DB numbers for the DBND, DBPA and DBCP parameters: 60000 to 60999
  - Reaction of CPU to error
    The CPU is not switched to STOP in response to error.
    If the CPU is to switch to STOP in response to error, the ANZ parameter must be evaluated in the user program
    (if ANZ is not equal to zero, the STP instruction must be started):
- S_MODB
  Data type of the parameter DB_NO: DB_ANY.
  Only data from non-optimized memory areas can be exchanged. Keep in mind during parameter assignment of the DB_NO parameter and the data block with the Modbus data conversion table.

Instructions of the "Communication processor > ET200S serial interface" library

Note the following when using the instructions of the ET 200S 1SI distributed I/O on an S7-1500 (library: "Communication processor > ET200S serial interface"):

- The hardware identifier of the communication module must be configured at the LADDR parameter.
- The send buffer and the receive buffer must both be located outside the optimized memory area.

ET 200S 1SI for 3964(R) and ASCII

S_RCV: receiving data

Description

The S_RCV instruction sends data from the ET 200S 1SI module to an S7 data area as specified at the DB_NO and DBB_NO parameters. For data transmission, the S_RCV instruction is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program.

Operating principle

The (static) signal state "1" at parameter EN_R enables the check for data to be read from the ET 200S 1SI module. An active transmission can be canceled by setting signal state "0" at parameter EN_R and with signal state "1" at parameter R. (Because transmission may sometimes not start up again after setting parameter EN_R to "1", you should set parameter R to "1" in addition to setting EN_R to "0" to cancel an active transmission.) The canceled receive request is terminated with an error message (STATUS output). Receiving is disabled as long as signal state "0" is set at parameter EN_R. Data transmission can take several calls (program cycles), depending on the data volume.

If the instruction detects signal state "1" at the R parameter, the current transmission job is canceled and the S_RCV instruction is reset to the initial state. Receiving is disabled as long

as signal state "1" is set at parameter R. If the signal state returns to "0", the canceled message frame is received again from the beginning.

Parameter LADDR specifies the address of the ET 200S 1SI to be addressed.

Output NDR indicates "Job completed without errors/data received" (all data read) and ERROR indicates error events. A corresponding error number is displayed in STATUS if an error occurs. If the receive buffer is filled to more than 2/3 of its capacity, STATUS returns a warning at each call of S_RCV. If no errors or warnings are pending, STATUS has a value of "0".

NDR and ERROR/STATUS are also output after the reset of the S_RCV instruction (LEN == 16#00) (see time sequence chart). The binary result BR is reset if an error occurs. If the block is completed without errors, the binary result has the status "1".

Parameters

Parameters	Declaration	Data type	Description
EN_R	INPUT	BOOL	Enable data reading
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Receiving is disabled.
LADDR	INPUT	INT	Start address of the ET 200S 1SI module The start address is taken from STEP 7.
DB_NO	INPUT	INT	Data block number Receive data block no.: CPU-specific, zero not allowed
DBB_NO	INPUT	INT	Data byte number (received data) **S7-1500**: 0 ≤ DBB_NO and DBB_NO + LEN ≤ size of the DB **S7-300/400**: 0 ≤ DBB_NO and DBB_NO + LEN < 8190
NDR [1]	OUTPUT	BOOL	Job completed without errors, data received STATUS parameter == 16#00
ERROR [1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 7677) contains the error information.
LEN [1]	OUTPUT	INT	Length of the message frame received 1 ≤ LEN ≤ 224 specified in number of bytes
STATUS [1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 7677) contains the error information.
COM_RST	IN_OUT	BOOL	Instruction restart
[1] On successful completion of a receive job, the NDR, ERROR, LEN and STATUS parameters remain available for the duration of **one** CPU cycle!			

Allocation in the data area

The S_RCV instruction works in combination with an I_RCV instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited.

Note

Exception: If the error STATUS == W#16#1Exx occurs, you can examine the SFCERR variable for additional details. This error variable must be loaded via symbolic access to the instance DB.

Startup

The COM_RST parameter of S_RCV instruction is used to report a startup to the instruction.

In the startup OB, set parameter COM_RST to 1.

Call the instruction in the cyclic mode without setting or resetting the COM_RST parameter.

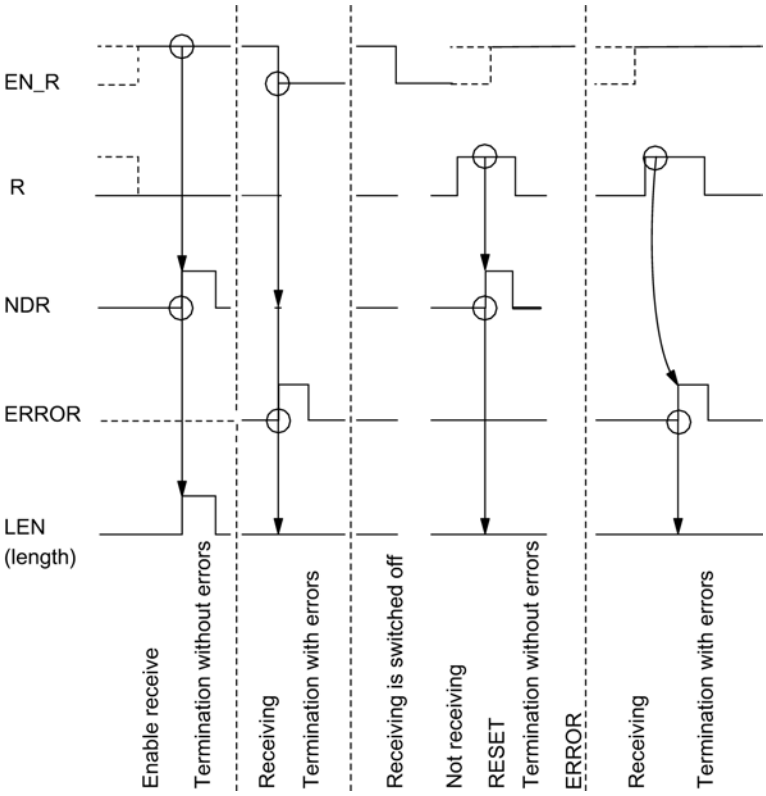
If the COM_RST parameter is set:

- The instruction acquires information about the ET 200S 1SI module (number of bytes in the I/O area, whether or not in the distributed I/O).
- The instruction resets itself, while cancelling any job started previously (prior the last CPU transition to STOP).

Once the instruction acquired information about the ET 200S 1SI module, it automatically resets the COM_RST parameter.

Time sequence chart

The figure below illustrates the characteristics of the NDR, LEN and ERROR parameters, depending on the wiring of the EN_R and R inputs.



Note

Input EN_R must be set to the static signal state "1". Parameter EN_R must be provided the logic operation result "1" for the duration of the receive request.

Rules

Note

The S_RCV instruction has no parameter check. If incorrect parameters are assigned, the CPU can go into STOP mode.

Note

The CPU startup routine of the ET 200S module for the S_RCV instruction must be completed before the ET 200S 1SI module can receive a job initiated after the status transition of the CPU from STOP to RUN.

S_SEND: Sending Data

Description

The S_SEND instruction transmits a data block from a DB to the ET 200S 1SI as specified at the DB_NO, DBB_NO and LEN parameters. For data transmission, the S_SEND instruction is called in a static (unconditional) operation in the cycle or **alternatively**, in a time-controlled program.

Operating principle

Data transmission is initiated by a positive edge at the REQ input. Data transmission can take several calls (program cycles), depending on the data volume.

The S_SEND instruction can be called in the cycle by setting signal state "1" at parameter input R. This setting cancels the transmission to the ET 200S 1SI module and resets the S_SEND instruction to the initial state. Data that has already been received by the ET 200S 1SI module is still transmitted to the communication partner. A static signal state "1" at the input R indicates that data transmission is disabled.

Parameter LADDR specifies the address of the ET 200S 1SI to be addressed.

Output DONE indicates "Job completed without errors" and ERROR indicates error events. A corresponding event number is displayed in STATUS if an error occurs. If no errors occurs, the STATUS has a value 0. DONE and ERROR/STATUS are also output for RESET of the S_SEND instruction (see the time sequence chart). The binary result BR is reset if an error occurs. If the block is completed without errors, the binary result has the status "1".

Parameters

Parameters	Declaration	Data type	Description
REQ	INPUT	BOOL	Job trigger at the positive edge
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Transmission is disabled.
LADDR	INPUT	INT	Start address of the ET 200S 1SI module The start address is taken from STEP 7.
DB_NO	INPUT	INT	Data block number Send DB no.: CPU-specific, zero not allowed
DBB_NO	INPUT	INT	Data byte number (transmit data) **S7-1500**: 0 ≤ DBB_NO and DBB_NO + LEN ≤ size of the DB **S7-300/400**: 0 ≤ DBB_NO and DBB_NO + LEN < 8190
LEN	INPUT	INT	Data length 1 ≤ LEN ≤ 224 specified in number of bytes

Parameters	Declaration	Data type	Description
DONE [1]	OUTPUT	BOOL	Job completed without errors STATUS parameter == 16#00
ERROR [1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 7677) contains the error information.
STATUS [1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 7677) contains the error information.
COM_RST	IN_OUT	BOOL	Instruction restart
[1] On successful completion of a transmission job, the DONE, ERROR and STATUS parameters remain available for the duration of **one** CPU cycle!			

Allocation in the data area

The S_SEND instruction works in combination with an I_SEND instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited.

Note

Exception: If the error STATUS == W#16#1Exx occurs, you can examine the SFCERR variable for additional details. This error variable must be loaded via symbolic access to the instance DB.

Startup

The COM_RST parameter of S_SEND instruction is used to report a startup to the instruction.

In the startup OB, set parameter COM_RST to 1.

Call the instruction in the cyclic mode without setting or resetting the COM_RST parameter.

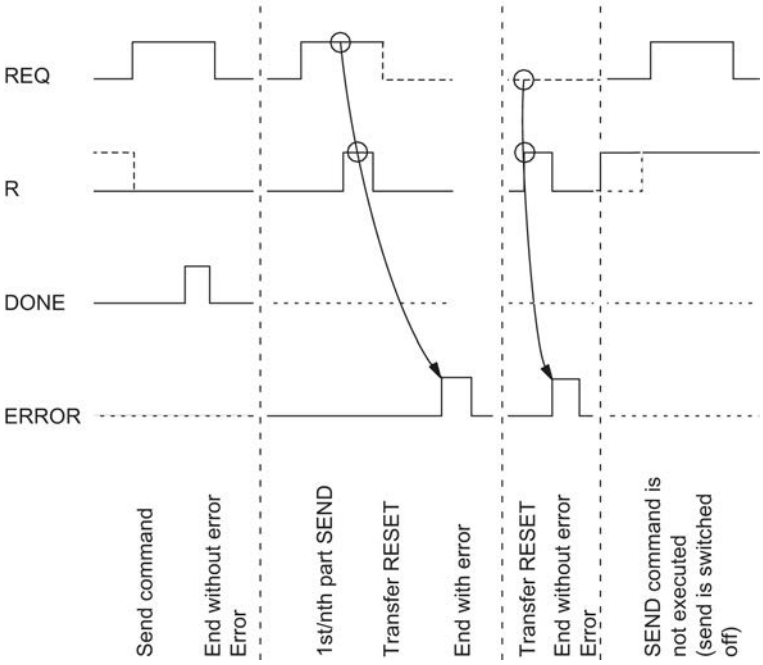
If the COM_RST parameter is set:

- The instruction acquires information about the ET 200S 1SI module (number of bytes in the I/O area, whether or not in the distributed I/O).
- The instruction resets itself, while cancelling any job started previously (prior the last CPU transition to STOP).

Once the instruction acquired information about the ET 200S 1SI module, it automatically resets the COM_RST parameter.

Time sequence chart

The figure below illustrates the characteristics of the DONE and ERROR parameters, depending on the wiring of the REQ and R inputs.



Note

Input REQ is edge-triggered. A positive edge at the input REQ is sufficient. The RLO (result of the logic operation) does not necessarily have to be "1" for the entire duration of the transmission.

Rules

Note

The S_SEND instruction has no parameter check. If incorrect parameters are assigned, the CPU can go into STOP mode.

Note

The CPU startup routine of the ET 200S module for the S_SEND instruction must be completed (see above) before the ET 200S 1SI module can execute a job initiated after the status transition of the CPU from STOP to RUN. A job initiated in the meantime is not lost. Once the startup coordination is completed, the job is sent to the ET 200S 1SI module.

S_VSTAT: Read accompanying signals at the RS 232C interface

Description

The S_VSTAT instruction reads the RS 232C accompanying signals of the ET 200S 1SI module and makes them available to you in the block parameters. For data transmission, the S_VSTAT instruction is called by means of (unconditional) static operation in the cycle or alternatively, in a time-controlled program.

Operating principle

The RS 232C accompanying signals are updated each time the instruction is called (cyclic polling).

The address of the ET 200S 1SI module to be addressed is specified in the LADDR parameter.

Parameters

Parameters	Declaration	Data type	Description
REQ	INPUT	BOOL	Job trigger at the positive edge
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Transmission is disabled.
LADDR	INPUT	INT	Start address of the ET 200S 1SI module The start address is taken from STEP 7.
DONE[1]	OUTPUT	BOOL	Indicates that the instruction has terminated. (ET 200S 1SI output)
ERROR[1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 7677) contains the error information.
STATUS[1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 7677) contains the error information.
DTR_OUT [1]	OUTPUT	BOOL	Data terminal ready, ET 200S 1SI ready for operation. (ET 200S 1SI output)
DSR_IN [1]	OUTPUT	BOOL	Data set ready; communication partner ready for operation. (ET 200S 1SI input)
RTS_OUT [1]	OUTPUT	BOOL	Request to send, ET 200S 1SI clear to send. (ET 200S 1SI output)
CTS_IN [1]	OUTPUT	BOOL	Clear to send; communication partner can receive data from the ET 200S 1SI module (response to RTS = ON of the ET 200S 1SI) (ET 200S 1SI input)

Parameters	Declaration	Data type	Description
DCD_IN [1]	OUTPUT	BOOL	Data carrier detect (ET 200S 1SI input)
COM_RST	IN_OUT	BOOL	Instruction restart
[1] On successful completion of a job, these parameters remain available for the duration of **one** CPU cycle.			

Allocation in the data area

The S_VSTAT instruction works in combination with an I_STAT instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited

Note

A minimum pulse duration is necessary to detect a signal change. Determining factors are the CPU cycle time, the update time on the ET 200S 1SI module and the response time of the communication partner.

Startup

The COM_RST parameter of the S_VSTAT instruction is used to report a startup to the instruction.

Set the COM_RST parameter in the startup OB to 1.

Call the instruction in cyclic mode without setting or resetting the COM_RST parameter.

If the COM_RST parameter is set:

- The instruction acquires information about the ET 200S 1SI module (number of bytes in the I/O area, whether or not in the distributed I/O).
- The instruction resets itself, while cancelling any job started previously (prior the last CPU transition to STOP).

Once the instruction acquired information about the ET 200S 1SI module, it automatically resets the COM_RST parameter.

See also

RS232 mode (CP 34x, 44x, CPU 31xC-2 PtP + ET 200S 1SI) (Page 1048)

S_VSET: Write accompanying signals at the RS 232C interface

Description

You can set and reset the interface outputs using the corresponding parameter inputs of the S_VSET instruction. The S_VSET instruction is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program.

The address of the ET 200S 1SI module to be addressed is specified in the LADDR parameter.

Parameters

Parameters	Declaration	Data type	Description
REQ	INPUT	BOOL	Job trigger at the positive edge
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Transmission is disabled.
LADDR	INPUT	INT	Start address of the ET 200S 1SI module The start address is taken from STEP 7.
RTS	INPUT	BOOL	Request to send, ET 200S 1SI clear to send. (Controls ET 200S 1SI output)
DTR	INPUT	BOOL	Data terminal ready, ET 200S 1SI ready for operation. (Controls ET 200S 1SI output)
DONE[1]	OUTPUT	BOOL	(ET 200S 1SI output)
ERROR[1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 7677) contains the error information.
STATUS[1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 7677) contains the error information.
COM_RST	IN_OUT	BOOL	Instruction restart
[1] On successful completion of a job, these parameters remain available for the duration of **one** CPU cycle.			

Allocation in the data area

The S_VSET instruction works in combination with an I_SET instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited

Startup

The COM_RST parameter of the S_VSET instruction is used to report a startup to the instruction.

Set the COM_RST parameter in the startup OB to 1.

Call the instruction in cyclic mode without setting or resetting the COM_RST parameter.

If the COM_RST parameter is set:

- The instruction acquires information about the ET 200S 1SI module (number of bytes in the I/O area, whether or not in the distributed I/O).
- The instruction resets itself, while cancelling any job started previously (prior the last CPU transition to STOP).

Once the instruction acquired information about the ET 200S 1SI module, it automatically resets the COM_RST parameter.

See also

RS232 mode (CP 34x, 44x, CPU 31xC-2 PtP + ET 200S 1SI) (Page 1048)

S_XON: Assign parameters for XON/XOFF data flow control

Description

Using the S_XON instruction, you can set additional parameters if the module is configured for XON/XOFF data flow control.

Note

The instruction S_XON must be started before the instructions S_SEND, S_RCV, S_VSET and S_VSTAT and have completed its task (DONE = TRUE).

Parameters

Parameters	Declaration	Data type	Description
REQ	INPUT	BOOL	Job trigger at the positive edge
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Transmission is disabled.
LADDR	INPUT	INT	Start address of the ET 200S 1SI module The start address is taken from STEP 7.
XON	INPUT	BYTE	XON character 0 to $7F_H$ (7 data bits) 0 to FF_H (8 data bits) Default: 11 (DC1)
XOFF	INPUT	BYTE	XOFF character 0 to $7F_H$ (7 data bits) 0 to FF_H (8 data bits) Default: 13 (DC3)
WAIT_FOR_XON	INPUT	TIME	Wait time for XON after XOFF 20 ms to 10 min 55 s 350 ms Default: 2 s
DONE[1]	OUTPUT	BOOL	Job completed without errors STATUS parameter == 16#00
ERROR[1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 7677) contains the error information.
STATUS[1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 7677) contains the error information.

Parameters	Declaration	Data type	Description
COM_RST	IN_OUT	BOOL	Instruction restart
[1] The DONE, ERROR and STATUS parameters are available for **one** CPU cycle following a successful request.			

Allocation in the data area

The S_XON instruction works in combination with an I_XON instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited.

Note

Exception: If error STATUS == W#16#1Exx occurs, you can view the SFCERR variable for more precise error information.

Startup

The COM_RST parameter of the S_XON instruction is used to report a startup to the instruction.

Set the COM_RST parameter in the startup OB to 1.

Call the instruction in cyclic mode without setting or resetting the COM_RST parameter.

If the COM_RST parameter is set:

- The instruction acquires information about the ET 200S 1SI module (number of bytes in the I/O area, whether or not in the distributed I/O).
- The instruction resets itself, while cancelling any job started previously (prior the last CPU transition to STOP).

Once the instruction acquired information about the ET 200S 1SI module, it automatically resets the COM_RST parameter.

See also

Data flow control and secondary signals (Page 1077)

S_RTS: Assign parameters for RTS/CTS data flow control

Description

Using the S_RTS instruction, you can configure additional parameters if the module is configured for RTS/CTS data flow control.

Note

The instruction S_RTS must be started before the instructions S_SEND, S_RCV, S_VSET and S_VSTAT and have completed its task (DONE = TRUE).

Parameters

Parameters	Declaration	Data type	Description
REQ	INPUT	BOOL	Job trigger at the positive edge
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Transmission is disabled.
LADDR	INPUT	INT	Start address of the ET 200S SI module The start address is taken from STEP 7.
WAIT_FOR_CTS	INPUT	TIME	Wait time for CTS = ON 20 ms to 10 min 55 s 350 ms Default: 2 s
DONE[1]	OUTPUT	BOOL	Job completed without errors STATUS parameter == 16#00
ERROR[1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 7677) contains the error information.
STATUS[1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 7677) contains the error information.
COM_RST	IN_OUT	BOOL	Instruction restart
[1] The DONE, ERROR and STATUS parameters are available for one CPU cycle following a successful request.			

Allocation in the data area

The S_RTS instruction works in combination with an I_RTS instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited.

Note

Exception: If error STATUS == W#16#1Exx occurs, you can view the SFCERR variable for more precise error information.

Startup

The COM_RST parameter of the S_RTS instruction is used to report a startup to the instruction.

Set the COM_RST parameter in the startup OB to 1.

Call the instruction in cyclic mode without setting or resetting the COM_RST parameter.

If the COM_RST parameter is set:

- The instruction acquires information about the ET 200S 1SI module (number of bytes in the I/O area, whether or not in the distributed I/O).
- The instruction resets itself, while cancelling any job started previously (prior the last CPU transition to STOP).

Once the instruction acquired information about the ET 200S 1SI module, it automatically resets the COM_RST parameter.

See also

Data flow control and secondary signals (Page 1077)

S_V24: Assign parameters for data flow control using automatic operation of the RS 232C accompanying signals

Description

Using the S_V24 instruction, you can configure additional parameters if the module is configured for automatic control of the RS 232C accompanying signals.

Note

The instruction S_V24 must be started before the instructions S_SEND, S_RCV, S_VSET and S_VSTAT and have completed its task (DONE = TRUE).

Parameters

Parameters	Declaration	Data type	Description
REQ	INPUT	BOOL	Job trigger at the positive edge
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Transmission is disabled.
LADDR	INPUT	INT	Start address of the ET 200S 1SI module The start address is taken from STEP 7.
TIME_RTS_OFF	INPUT	TIME	Time that must elapse after transmission before RTS is disabled. 0 ms to 10 min 55 s 350 ms Default: 10 ms
DATA_WAIT_TIME	INPUT	TIME	Time to wait until the partner sets CTS = ON once RTS has been set. 0 ms to 10 min 55 s 350 ms Default: 10 ms
DONE[1]	OUTPUT	BOOL	Job completed without errors STATUS parameter == 16#00
ERROR[1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 7677) contains the error information.
STATUS[1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 7677) contains the error information.

Parameters	Declaration	Data type	Description
COM_RST	IN_OUT	BOOL	Instruction restart
[1] The DONE, ERROR and STATUS parameters are available for **one** CPU cycle following a successful request.			

Allocation in the data area

The S_V24 instruction works in combination with an I_V24 instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited.

Note

Exception: If error STATUS == W#16#1Exx occurs, you can view the SFCERR variable for more precise error information.

Startup

The COM_RST parameter of the S_V24 instruction is used to report a startup to the instruction.

Set the COM_RST parameter in the startup OB to 1.

Call the instruction in cyclic mode without setting or resetting the COM_RST parameter.

If the COM_RST parameter is set:

- The instruction acquires information about the ET 200S 1SI module (number of bytes in the I/O area, whether or not in the distributed I/O).
- The instruction resets itself, while cancelling any job started previously (prior the last CPU transition to STOP).

Once the instruction acquired information about the ET 200S 1SI module, it automatically resets the COM_RST parameter.

See also

RS232 mode (CP 34x, 44x, CPU 31xC-2 PtP + ET 200S 1SI) (Page 1048)

Data flow control and secondary signals (Page 1077)

STATUS parameter

Event classes

Event class	Meaning
2	Error initializing
5	Error executing a CPU job
7	Send errors
8	Receive errors

Event class	Meaning
11 ($0B_H$)	Warning
30 ($1E_H$)	Errors in communication between the module and CPU

STATUS parameter

Error code(W#16#...)	Description	Remedy
0201	No (valid) configuration available.	Provide the module with correct parameters. If necessary, ensure the system is correctly installed.
0502	Request not permitted in this operating mode of the ET 200S 1SI module (e.g., device interface not parameterized).	The send message frame is longer than 224 bytes. The ET 200S 1SI module canceled the transmission job. Select a shorter message frame length.
050E	Invalid telegram length	The send message frame is longer than 224 bytes. The ET 200S 1SI module canceled the transmission job. Select a shorter telegram length.
0550	The parameter update request is invalid for the currently selected form of ET 200S 1SI module data flow control.	Edit the instruction parameters (S_XON, S_RTS, S_V24) in the AS program or modify the data flow control function of the ET 200S 1SI module in the hardware configuration to ensure compatibility.
0551	Frame execution error during communication between the ET 200S 1SI module and the automation system. The error occurred in the automation system during transmission of a received message frame from the ET 200S 1SI module.	The module and the automation system have canceled the transmission. Repeat the receive request; the ET 200S 1SI module sends the received message again.
0702	Only for 3964(R): Error establishing connection: After STX was sent, NAK or any other code (except for DLE or STX) was received.	You can interconnect an interface tester (FOXPG) in the transmission line to debug the partner device.
0703	Only for 3964(R): Acknowledgment delay time (QVZ) exceeded: After STX was sent, partner did not respond within the acknowledgment delay time.	The partner device is too slow or not ready to receive or there is a break in the transmission line, for example. You can interconnect an interface tester (FOXPG) in the transmission line to debug the partner device.
0704	Only for 3964(R): Termination by partner: One or more characters were received from the partner during sending.	Check if the partner is also indicating errors, possibly because not all transmitted data arrived (e.g., break in the transmission line), fatal errors are pending or the partner device has malfunctioned. You can interconnect an interface tester (FOXPG) in the transmission line to debug the partner device.
0705	Only for 3964(R): Negative acknowledgment when sending	Check if the partner is also indicating errors, possibly because not all transmit data has arrived (e.g., break in the transmission line), fatal errors are pending or the partner device has malfunctioned. You can interconnect an interface tester (FOXPG) in the transmission line to debug the partner device.

Error code(W#16#...)	Description	Remedy
0706	Only for 3964(R): End-of-connection error: • Partner rejected telegram at end of connection with NAK or a random string (except for DLE) or • Acknowledgment characters (DLE) received too early.	Check if the partner is also indicating errors, possibly because not all transmit data has arrived (e.g., break in the transmission line), fatal errors are pending or the partner device has malfunctioned. You can interconnect an interface tester (FOXPG) in the transmission line to debug the partner device.
0707	Only for 3964(R): Acknowledgment delay time exceeded at end of connection or response monitoring time exceeded after a send telegram: After connection termination with DLE ETX, no response received from partner within acknowledgment delay time.	Partner device too slow or faulty. If necessary, use an interface test device switched into the transmission line to check.
0708	Only for ASCII drivers: The wait time for XON or CTS = ON has expired.	The communication partner is faulty, too slow or has been taken offline. Check the communication partner; you may need to change the parameter assignment.
070B	Only for 3964(R): Initialization conflict cannot be resolved because both partners have high priority.	Change the parameter assignment.
070C	Only for 3964(R): Initialization conflict cannot be resolved because both partners have low priority.	Change the parameter assignment.
0802	Only for 3964(R): Error establishing connection: • In idle mode, one or more random codes (other than NAK or STX) were received or • After an STX was received, partner sent more characters without waiting for response DLE. After partner power ON: • The module receives an undefined character while the partner is being switched on.	You can interconnect an interface tester (FOXPG) in the transmission line to debug the partner device.
0805	Only for 3964(R): Logical error while receiving: After DLE was received, a further random code (other than DLE or ETX) was received.	Check if the partner always duplicates the DLE in the telegram header and data string or the connection is terminated with DLE ETX. You can interconnect an interface tester (FOXPG) in the transmission line to debug the partner device.
0806	Character delay time (ZVZ) exceeded: • Two successive characters were not received within character delay time or Only for 3964(R): • 1. character after sending of DLE while establishing connection was not received within the character delay time.	Partner device too slow or faulty. You can interconnect an interface tester (FOXPG) in the transmission line to debug the partner device.

Error code(W#16#...)	Description	Remedy
0807	Only for 3964(R): Illegal telegram length: A zero-length telegram has been received.	Receipt of a zero-length telegram is not an error. Check why the communication partner is sending telegrams without user data.
0808	Only for 3964(R): Error in block check character (BCC): The value of BCC calculated internally does not match the BCC received by the partner when the connection was terminated.	Check if the connection is seriously disrupted; in this case you may also occasionally see error codes. You can interconnect an interface tester (FOXPG) in the transmission line to debug the partner device.
0809	Only for 3964(R): The number of repetitions must the set to the same value.	Declare a block wait time at the communication partner identical to that in your module. Check for malfunction of the communication partner; you may need to use an interface test device that is switched into the transmission line.
080A	There is no free receive buffer available: No receive buffer space available for receiving data.	Increase the call rate for the S_RCV instruction.
080C	Transmission error: • Transmission error (parity/stop bit/overflow error) detected. Only for 3964(R): • If this happens during send or receive operations, repetition is started. • If a corrupted character is received in idle mode, the error is reported immediately so that disturbances on the transmission line can be detected early. • If the SF LED (red) lights up, there is a break in the connecting cable between the two communication partners.	Disturbances on the transmission line cause message frame repetitions, thus lowering user data throughput. The risk of undetected error increases. Change your system setup or cable wiring. Check the cables of the communication partners or verify that both devices have matching settings for the baud rate, parity and number of stop bits.
080D	BREAK: Break in receive line to partner.	Reconnect or switch on partner.
0810	Only for ASCII drivers: Parity error: • If the SF LED (red) lights up, there is a break in the connecting cable between the two communication partners.	Check the cables of the communication partners or verify that both devices have matching settings for the baud rate, parity and number of stop bits. Change your system setup or cable wiring.
0811	Only for ASCII drivers: Character frame error: • If the SF LED (red) lights up, there is a break in the connecting cable between the two communication partners.	Check the cables of the communication partners or verify that both devices have matching settings for the baud rate, parity and number of stop bits. Change your system setup or cable wiring.
0812	Only for ASCII drivers: More characters were received after the module had sent XOFF or set CTS to OFF.	Reconfigure the communication partner or read module data more rapidly.
0818	Only for ASCII drivers: DSR = OFF or CTS = OFF	The partner has switched the DSR or CTS signal to "OFF" before or during a transmission. Check the partner's control of the RS 232C accompanying signals.

Error code(W#16#...)	Description	Remedy
0850	The length of the receive message frame is greater than 224 bytes or the defined message frame length.	Adjust the message frame length of the partner
0B01	Reception buffer is more than 2/3 full	
1E0D	"Job aborted due to warm restart, hot restart or reset"	
1E0E	Static error at the call of SFC DPRD_DAT. The RET_VAL return value of the SFC is available for evaluation in the SFCERR variable of the instance DB.	Load the SFCERR tag from the instance DB.
1E0F	Static error at the call of SFC DPWR_DAT. The RET_VAL return value of the SFC is available for evaluation in the SFCERR variable of the instance DB.	Load the SFCERR tag from the instance DB.
1E10	Static error during the call of SFC RD_LGADR. The RET_VAL return value of the SFC is made available for evaluation in the SFCERR variable at the instance DB.	Load the SFCERR tag from the instance DB.
1E11	Static error during the call of SFC RDSYSST. The RET_VAL return value of the SFC is made available for evaluation in the SFCERR variable at the instance DB.	Load the SFCERR tag from the instance DB.
1E20	Parameter out of range.	Modify the input of the instruction so that it is within the valid range.
1E41	Invalid number of bytes specified at instruction parameter LEN	You must stay within a range of values of 1 to 224 bytes.

Evaluating the SFCERR variable

More detailed information on the error events (1E) $0E_H$, (1E) $0F_H$, (1E) 10_H and (1E) 11_H of event class 30 is available in the SFCERR variable.

You can load the SFCERR variable from the instance DB of the corresponding instruction.

The error messages entered in the SFCERR variable are described in the sections on the "DPRD_DAR" and SFC15 "DPWR_DAT" system functions in the *System Software for S7-300/400, System and Standard Functions* Reference Manual.

ET 200S 1SI for Modbus

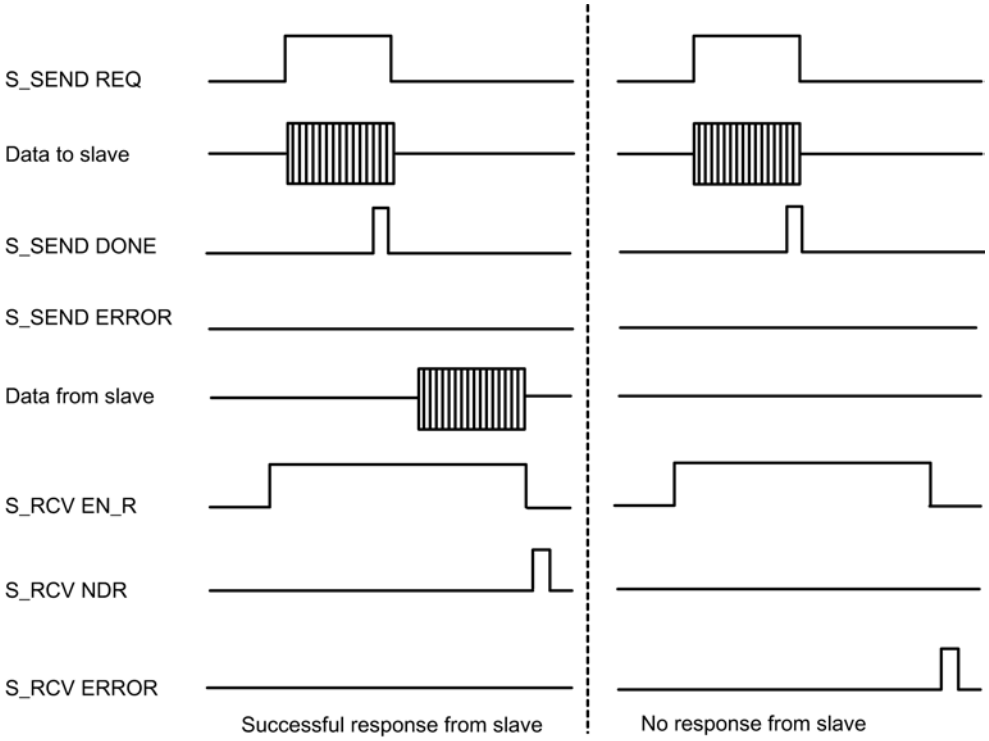
Sending and receiving as Modbus master

Description

Data is transferred between the module and CPU by means of the S_SEND and S_RCV instructions. The S_SEND instruction is activated by an edge at the REQ input if data is to be output. The S_RCV instruction is set ready to receive with EN_R=1. An S_RCV is required for all reading function codes.

Sending and receiving data using the Modbus master protocol

The S_SEND and S_RCV instructions must be activated in order to execute a Modbus master job. The S_SEND instruction is activated by an edge at the REQ input when data is to be output to the module. The S_RCV instruction is made ready to receive data from the module with EN_R=1. An S_RCV is required for all reading function codes. The figure below shows the overall characteristics of the S_SEND and S_RCV parameters when a Modbus job is executed.



Note

Send and receive sequence

As the interface between the user program and the interface module operates in half-duplex mode you must observe the following items:

After positive acknowledgment of the Modbus master read job, you first have to fetch the receive data from the interface module by calling the S_RCV instruction before starting a new Modbus master send job.

S_RCV: receiving data

Description

The S_RCV instruction sends data from the ET 200S 1SI module to an S7 data area as specified at the DB_NO and DBB_NO parameters. For data transmission, the S_RCV instruction is called in the cycle or alternatively, in a static (unconditional) operation of a time-controlled program.

Operating principle

The (static) signal state "1" at parameter EN_R enables the check for data to be read from the ET 200S 1SI module. An active transmission can be canceled by setting signal state "0" at parameter EN_R and with signal state "1" at parameter R. (Because transmission may sometimes not start up again after setting parameter EN_R to "1", you should set parameter R to "1" in addition to setting EN_R to "0" to cancel an active transmission.) The canceled receive request is terminated with an error message (STATUS output). Receiving is disabled as long as signal state "0" is set at parameter EN_R. Data transmission can take several calls (program cycles), depending on the data volume.

If the instruction detects signal state "1" at the R parameter, the current transmission job is canceled and the S_RCV instruction is reset to the initial state. Receiving is disabled as long as signal state "1" is set at parameter R. If the signal state returns to "0", the canceled message frame is received again from the beginning.

Parameter LADDR specifies the address of the ET 200S 1SI to be addressed.

Output NDR indicates "Job completed without errors/data received" (all data read) and ERROR indicates error events. A corresponding error number is displayed in STATUS if an error occurs. If the receive buffer is filled to more than 2/3 of its capacity, STATUS returns a warning at each call of S_RCV. If no errors or warnings are pending, STATUS has a value of "0".

NDR and ERROR/STATUS are also output after the reset of the S_RCV instruction (LEN == 16#00) (see time sequence chart). The binary result BR is reset if an error occurs. If the block is completed without errors, the binary result has the status "1".

Parameters

Parameters	Declaration	Data type	Description
EN_R	INPUT	BOOL	Enable data reading
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Receiving is disabled.
LADDR	INPUT	INT	Start address of the ET 200S 1SI module The start address is taken from STEP 7.
DB_NO	INPUT	INT	Data block number Receive data block no.: CPU-specific, zero not allowed

Parameters	Declaration	Data type	Description
DBB_NO	INPUT	INT	Data byte number (received data) **S7-1500**: 0 ≤ DBB_NO and DBB_NO + LEN ≤ size of the DB **S7-300/400**: 0 ≤ DBB_NO and DBB_NO + LEN < 8190
NDR [1]	OUTPUT	BOOL	Job completed without errors, data received STATUS parameter == 16#00
ERROR [1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 7690) contains the error information.
LEN [1]	OUTPUT	INT	Length of the message frame received 1 ≤ LEN ≤ 224 specified in number of bytes
STATUS [1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 7690) contains the error information.
COM_RST	IN_OUT	BOOL	Instruction restart
[1] On successful completion of a receive job, the NDR, ERROR, LEN and STATUS parameters remain available for the duration of **one** CPU cycle!			

Allocation in the data area

The S_RCV instruction works in combination with an I_RCV instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited.

Note

Exception: If the error STATUS == W#16#1Exx occurs, you can examine the SFCERR variable for additional details. This error variable must be loaded via symbolic access to the instance DB.

Startup

The COM_RST parameter of S_RCV instruction is used to report a startup to the instruction.

In the startup OB, set parameter COM_RST to 1.

Call the instruction in the cyclic mode without setting or resetting the COM_RST parameter.

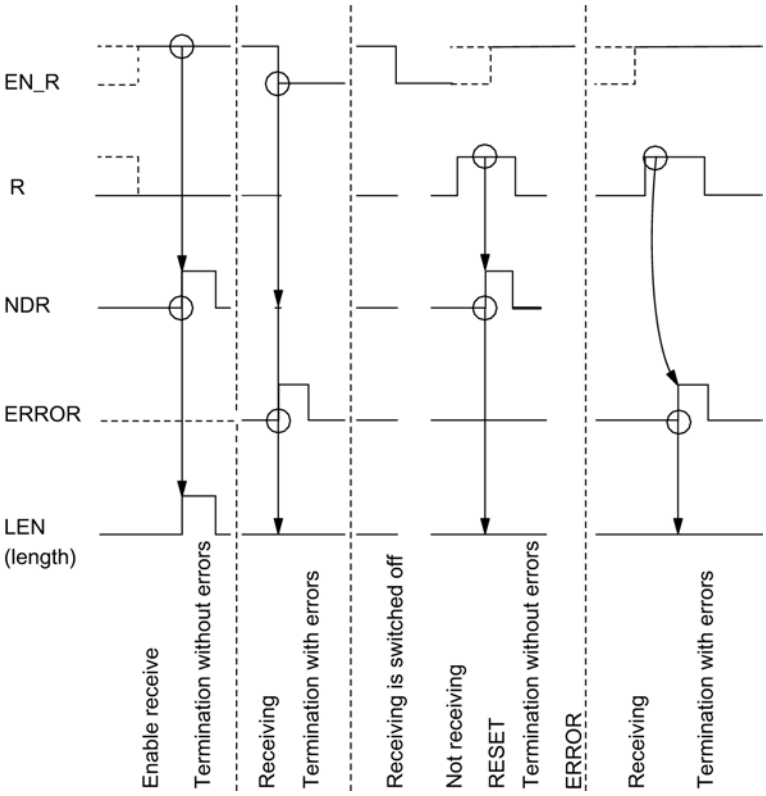
If the COM_RST parameter is set:

- The instruction acquires information about the ET 200S 1SI module (number of bytes in the I/O area, whether or not in the distributed I/O).
- The instruction resets itself, while cancelling any job started previously (prior the last CPU transition to STOP).

Once the instruction acquired information about the ET 200S 1SI module, it automatically resets the COM_RST parameter.

Time sequence chart

The figure below illustrates the characteristics of the NDR, LEN and ERROR parameters, depending on the wiring of the EN_R and R inputs.



Note

Input EN_R must be set to the static signal state "1". Parameter EN_R must be provided the logic operation result "1" for the duration of the receive request.

Rules

Note

The S_RCV instruction has no parameter check. If incorrect parameters are assigned, the CPU can go into STOP mode.

Note

The CPU startup routine of the ET 200S module for the S_RCV instruction must be completed before the ET 200S 1SI module can receive a job initiated after the status transition of the CPU from STOP to RUN.

See also

Data transmission with the ET 200S 1SI Modbus master (Page 1098)

S_SEND: Sending Data

Description

The S_SEND instruction transmits a data block from a DB to the ET 200S 1SI as specified at the DB_NO, DBB_NO and LEN parameters. For data transmission, the S_SEND instruction is called in a static (unconditional) operation in the cycle or **alternatively**, in a time-controlled program.

Operating principle

Data transmission is initiated by a positive edge at the REQ input. Data transmission can take several calls (program cycles), depending on the data volume.

The S_SEND instruction can be called in the cycle by setting signal state "1" at parameter input R. This setting cancels the transmission to the ET 200S 1SI module and resets the S_SEND instruction to the initial state. Data that has already been received by the ET 200S 1SI module is still transmitted to the communication partner. A static signal state "1" at the input R indicates that data transmission is disabled.

Parameter LADDR specifies the address of the ET 200S 1SI to be addressed.

Output DONE indicates "Job completed without errors" and ERROR indicates error events. A corresponding event number is displayed in STATUS if an error occurs. If no errors occurs, the STATUS has a value 0. DONE and ERROR/STATUS are also output for RESET of the S_SEND instruction (see the time sequence chart). The binary result BR is reset if an error occurs. If the block is completed without errors, the binary result has the status "1".

Parameters

Parameters	Declaration	Data type	Description
REQ	INPUT	BOOL	Job trigger at the positive edge
R	INPUT	BOOL	Cancel the job Cancels the job in progress. Transmission is disabled.
LADDR	INPUT	INT	Start address of the ET 200S 1SI module The start address is taken from STEP 7.
DB_NO	INPUT	INT	Data block number Send DB no.: CPU-specific, zero not allowed
DBB_NO	INPUT	INT	Data byte number (transmit data) **S7-1500**: 0 ≤ DBB_NO and DBB_NO + LEN ≤ size of the DB **S7-300/400**: 0 ≤ DBB_NO and DBB_NO + LEN < 8190

Parameters	Declaration	Data type	Description
LEN	INPUT	INT	Data length 1 ≤ LEN ≤ 224 specified in number of bytes
DONE [1]	OUTPUT	BOOL	Job completed without errors STATUS parameter == 16#00
ERROR [1]	OUTPUT	BOOL	Job canceled with errors The STATUS parameter (Page 7690) contains the error information.
STATUS [1]	OUTPUT	WORD	Error specification If ERROR == 1, STATUS parameter (Page 7690) contains the error information.
COM_RST	IN_OUT	BOOL	Instruction restart
[1] On successful completion of a transmission job, the DONE, ERROR and STATUS parameters remain available for the duration of **one** CPU cycle!			

Allocation in the data area

The S_SEND instruction works in combination with an I_SEND instance DB. The DB number is included with the call. Access to the data in the instance DB is prohibited.

Note

Exception: If the error STATUS == W#16#1Exx occurs, you can examine the SFCERR variable for additional details. This error variable must be loaded via symbolic access to the instance DB.

Startup

The COM_RST parameter of S_SEND instruction is used to report a startup to the instruction.

In the startup OB, set parameter COM_RST to 1.

Call the instruction in the cyclic mode without setting or resetting the COM_RST parameter.

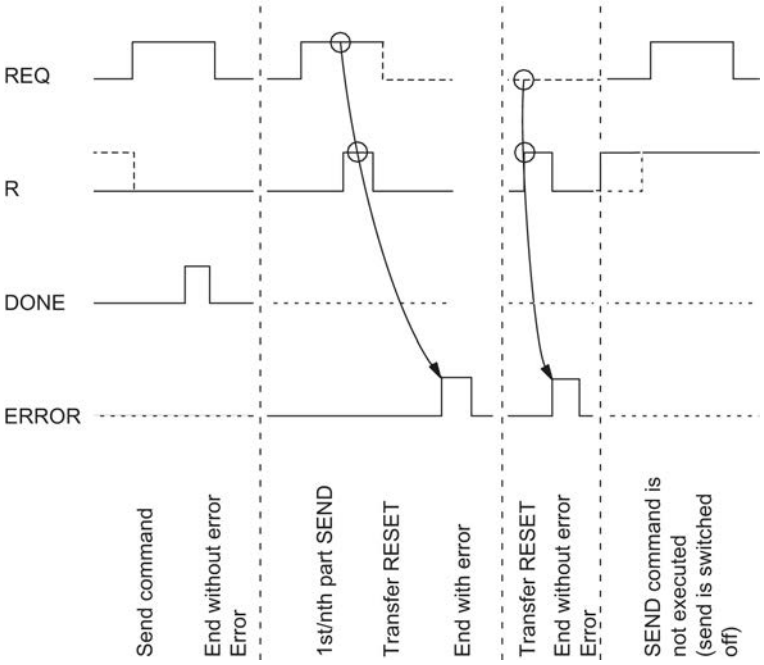
If the COM_RST parameter is set:

- The instruction acquires information about the ET 200S 1SI module (number of bytes in the I/O area, whether or not in the distributed I/O).
- The instruction resets itself, while cancelling any job started previously (prior the last CPU transition to STOP).

Once the instruction acquired information about the ET 200S 1SI module, it automatically resets the COM_RST parameter.

Time sequence chart

The figure below illustrates the characteristics of the DONE and ERROR parameters, depending on the wiring of the REQ and R inputs.



Note

Input REQ is edge-triggered. A positive edge at the input REQ is sufficient. The RLO (result of the logic operation) does not necessarily have to be "1" for the entire duration of the transmission.

Rules

Note

The S_SEND instruction has no parameter check. If incorrect parameters are assigned, the CPU can go into STOP mode.

Note

The CPU startup routine of the ET 200S module for the S_SEND instruction must be completed (see above) before the ET 200S 1SI module can execute a job initiated after the status transition of the CPU from STOP to RUN. A job initiated in the meantime is not lost. Once the startup coordination is completed, the job is sent to the ET 200S 1SI module.

See also

Data transmission with the ET 200S 1SI Modbus master (Page 1098)

S_MODB: Modbus slave instruction for ET 200S 1SI

Description

To enable execution of a Modbus slave job, the S_MODB instruction must be called cyclically in the user program. The S_MODB instruction receives the job from serial interface module ET200S Modbus/USS, executes it and returns the response to the module. Communication between the CPU and module is handled by means of the S_SEND and S_RCV instructions which are called by S_MODB .

Operating principle

Following each warm restart of the CPU, the user program must initialize the Modbus communication instruction. Initialization is triggered by a rising edge at the CP_START input. The instruction records the size of the I, Q, F, T and C address areas of the CPU in its instance data block. On successful completion of initialization, the FB sets the CP_START_OK output.

An initialization error is specified by the CP_START_ERROR output. In this case, Modbus communication is not possible and all requests from the Modbus master are answered with an exception code message.

S_MODB uses a Modbus data conversion table that is located in the data block in order to map the Modbus addresses to the SIMATIC S7 memory areas.

Parameters

Parameters	Declaration	Data type	Description
LADDR	INPUT	INT	Start address of serial interface module ET 200S 1SI. The start address is taken from STEP 7.
START_TIMER	INPUT	TIMER	Time for initialization timeout: CPU time that the instruction uses to prevent the initialization from freezing.
START_TIME	INPUT	S5TIME	Time value for initialization timeout: Valid maximum time for the instruction's initialization
DB_No	INPUT	BLOCK_DB	Number of the data block that contains the Modbus conversion table: See the data areas in the SIMATICCPU
OB_MASK	INPUT	BOOL	Masks I/O access errors and delays interrupts: TRUE can be set to specify that access errors to non-existing I/O are reported to the Modbus master only
CP_START	INPUT	BOOL	Start initialization of the instruction

Parameters	Declaration	Data type	Description
CP_START_FM	INPUT	BOOL	Initialization is triggered by a rising edge at the input START: Memory element to which the instruction saves the previous value of CP_START
CP_NDR	OUTPUT	BOOL	New Modbus write function completed
CP_START_OK	OUTPUT	BOOL	Initialization successfully completed: Initialization was completed without errors before expiration of the monitoring time.
CP_START_ERROR	OUTPUT	BOOL	Initialization terminated with error: Initialization could not be completed without errors, even after expiration of the monitoring time.
ERROR_NR	OUTPUT	WORD	Error number See Diagnostics
ERROR_INFO	OUTPUT	WORD	Additional error information See Diagnostics

Allocation in the data area

The S_MODB instruction works in combination with an IDB_MODB instance DB, the number of which is specified in the call.

Access to the data in the instance DB is prohibited.

How to Handle an Error

Input parameter OB_MASK can be used to instruct the Modbus instruction to mask I/O access errors. If an attempt is made to write to an I/O that is not available, the CPU does not go into STOP mode or call an error OB. The instruction detects the access error, cancels the job and returns an error response to the Modbus master.

STATUS parameter (Modbus / USS)

Event classes

Event class	Meaning
2	Error initializing
5	Error executing a CPU job
8	Receive errors
14 ($0E_H$)	General processing errors: Parameter assignment, processing of an S_SEND job. Evaluation of data received, receipt of an exception code message
30 ($1E_H$)	Errors in communication between the module and CPU

STATUS parameter

Error code(W#16#...)	Description	Remedy
0201	No (valid) configuration available.	Provide the module with correct parameters. If necessary, ensure the system is correctly installed.
0502	Order inadmissible in this operating mode of the serial interface module ET 200S Modbus/USS (example: device interface is not configured).	Evaluate the diagnostic interrupt and rectify the error accordingly.
050E	Invalid telegram length	The send message frame is longer than 224 bytes. The send job was aborted by the ET 200S Modbus/USS module. Select a shorter telegram length.
0530	Modbus master send job rejected because the response of the communication partner to a previous reading Modbus master send job has not yet been retrieved.	After a successful read Modbus master send job, you must first read the response from the communication partner from the module before you start a new Modbus master send job.
0551	Frame sequence errors in the communication between the serial interface module ET 200S Modbus/USS and the CPU. The error occurred while transmitting a received message frame of the ET 200S SI serial interface module to the CPU.	The module and the CPU have canceled the transmission. Repeat the receive job. The ET 200S Modbus/USS serial interface module sends the received message again.
0806	Character delay time exceeded. Two successive characters were not received within the character delay time.	Partner device too slow or faulty. You can interconnect an interface tester (FOXPG) in the transmission line to debug the partner device.
080A	Overflow of the receive buffer at the master during the reception of the response message frame.	Check the protocol settings of the slave.
080C	Transmission error (parity/stop bit/overflow error) detected.	Disturbances on the data link cause telegram repetitions, thus lowering user data throughput. The risk of undetected error increases. Change your system setup or cable wiring. Check the cables of the communication partners or verify that both devices have matching settings for the baud rate, parity and number of stop bits.
080D	BREAK: Break in receive line to partner.	Reconnect or switch on partner.
0810	Parity error: If the SF LED (red) lights up, there is a break in the cable that connects the two communication partners.	Check the cables of the communication partners or verify that both devices have matching settings for the baud rate, parity and number of stop bits. Change your system setup or cable wiring.

Error code(W#16#...)	Description	Remedy
0811	Character frame error: If the SF LED (red) lights up, there is a break in the cable that connects the two communication partners.	Check the cables of the communication partners or verify that both devices have matching settings for the baud rate, parity and number of stop bits. Change your system setup or cable wiring.
0812	Additional characters were received after the CTS serial port was set to OFF.	Reconfigure the communications partner or fetch from the serial interface faster.
0830	Master: A request message frame has been sent and the response monitoring time has elapsed without the start of a response message frame being detected. Slave: Broadcast not permitted with this function code.	Check if the transmission line has been interrupted (interface analysis may be necessary). Check that the same settings have been made on the module and communication partner for the following protocol parameters: transmission rate, number of data bits, parity and number of stop bits. Check that the value for the response monitoring time in PtP_PARAM is large enough. Check to see if the specified slave address is available. The Modbus master system is only allowed to use the broadcast function for the function codes enabled for this purpose.
0831	Master: The first character in the slave response message frame is different from the slave address that was sent in the request message frame (for normal operation). Slave: Function code received not permissible.	The wrong slave has responded. Check if the transmission line has been interrupted (interface analysis may be necessary). This function code cannot be used for this driver.
0832	Maximum number of bits or registers exceeded or number of bits cannot be divided by 16 if the SIMATIC timers or counters memory areas are accessed.	Limit the maximum number of bits to 2,040 and the maximum number of registers to 127. Access to SIMATIC timers, counters in 16-bit intervals only.
0833	Bit or register number for function code FC 15/16 and message frame element byte_count do not match.	Correct the bit/register number or byte_count.
0834	Illegal bit coding for "Set bit / Reset bit" recognized.	Use only the 0000Hex or FF00Hex codes for FC05.
0835	Illegal diagnostic subcode (not 0000Hex) at function code FC 08 "Loop back test" recognized.	Use only the subcode 0000Hex for FC08.
0836	The internally generated value of the CRC 16 checksum does not match the CRC checksum received.	Check the formation of the CRC checksum on the Modbus master system.

Error code(W#16#...)	Description	Remedy
0837	Message frame sequence error: The Modbus master system sent a new request message frame before the driver transmitted the last response message frame.	Increase the timeout on the slave response message frame for Modbus master system.
0850	Length of the receive message frame greater than 224 bytes or greater than the configured message frame length.	Adjust the message frame length of the partner.
0E20	This connection requires 8 data bits. The driver is not ready for operation.	Correct the parameter assignment of the driver.
0E21	The parameterized multiplication factor for the character delay time is not within the range of 1 to 10. The driver is operating with a default setting of 1.	Correct the parameter assignment of the driver.
0E22	The operating mode set for the driver is illegal. "Normal operation" or "Noise suppression operation" must be specified. The driver is not ready for operation.	Correct the parameter assignment of the driver.
0E23	Master: An illegal value has been set for the response monitoring time: Valid values are between 50 and 655,000 ms. The driver is not ready for operation. Slave: An illegal value has been set for the slave address. Slave address 0 is illegal. The driver is not ready for operation.	Correct the parameter assignment of the driver. Correct the parameter assignment of the driver.
0E2E	An error occurred while the interface parameter file was being read. The driver is not ready for operation.	Restart the master (Mains_ON).
0E40	The value specified for LEN with S_SEND is too small.	The minimum length is 2 bytes.
0E41	The value specified for LEN with S_SEND is too small. A greater length is required for the transmitted function code.	The minimum length for this function code is 6 bytes.
0E42	The transmitted function code is illegal.	Only use permissible function codes.
0E43	Slave address 0 (= broadcast) is not permissible with this function code.	Only use the slave address 0 with suitable function codes.
0E44	The value of the "number of bits" transmitted is not within the range of 1 to 2,040.	The "number of bits" must be within the range of 1 to 2,040.
0E45	The value of the "number of registers" transmitted is not within the range of 1 to 127.	The "number of registers" must be within the range of 1 to 127.

Error code(W#16#...)	Description	Remedy
0E46	Function code 15 or 16: The values of the "number of bits"/"number of registers" transmitted is not in the range of 1 to 2,040 or 1 - 127.	The "number of bits"/"number of registers" must be within the range of 1 to 2,040 or 1 to 127.
0E47	Function code 15 or 16: LEN for S_SEND does not correspond to the "number of bits"/"number of registers" transmitted. LEN is too small.	Increase LEN for SEND until enough user data is transmitted to the module. More user data must be sent to the module because of the "number of bits"/"number of registers".
0E48	Function code 5: The code specified in the SEND source DB for "set bit" (FF00H) or "delete bit" (0000H) is incorrect.	The only permissible codes are "set bit" (FF00H), "delete bit" or 0000H.
0E49	Function code 8: The code specified in the SEND source code for "diagnostic code" is incorrect.	The only permissible code is "diagnostic code" 0000H.
0E4A	The length of this function code is greater than the maximum length.	The manual contains the maximum lengths for each function code.
0E50	The master received a response without sending anything.	A slave or another master is on the network. Check if the transmission line has been interrupted (interface analysis may be necessary).
0E51	Function code incorrect: The function code received in the response message frame is different from the function code sent.	Check the slave device.
0E52	Byte underflow: The number of characters received is less than indicated by the byte counter of the response message frame or expected for this function code.	Check the slave device.
0E53	Byte overflow: The number of characters received is greater than indicated by the byte counter of the response message frame or expected for this function code.	Check the slave device.
0E54	Byte counter incorrect: The byte counter received in the response message frame is too small.	Check the slave device.
0E55	Byte counter wrong: The byte counter received in the response message frame is wrong.	Check the slave device.

Error code(W#16#...)	Description	Remedy
0E56	Echo incorrect: The response message frame data echoed by the slave (number of bits, etc.) is different from the data sent in the response message frame.	Check the slave device.
0E57	CRC check incorrect: An error occurred when checking the CRC 16 checksum of the response message frame from the slave.	Check the slave device.
0E61	Response message frame with exception code 01: Illegal function	Refer to the manual for the slave device.
0E62	Response message frame with exception code 02: Illegal data address	Refer to the manual for the slave device.
0E63	Response message frame with exception code 03: Illegal data value	Refer to the manual for the slave device.
0E64	Response message frame with exception code 04: Failure in associated device	Refer to the manual for the slave device.
0E65	Response message frame with exception code 05: Acknowledgement	Refer to the manual for the slave device.
0E66	Response message frame with exception code 06: Occupied; message frame rejected	Refer to the manual for the slave device.
0E67	Response message frame with exception code 07: Negative acknowledgement	Refer to the manual for the slave device.
1E0D	"Job aborted due to warm restart, hot restart or reset"	
1E0E	Static error during call of the SFC DP_RDDAT. The RET_VAL return value of the SFC is made available for evaluation in the SFCERR variable at the instance DB.	Load the SFCERR tag from the instance DB.
1E0F	Static error during call of the SFC DP_WRDAT. The RET_VAL return value of the SFC is made available for evaluation in the SFCERR variable at the instance DB.	Load the SFCERR tag from the instance DB.
1E10	Static error during the call of SFC RD_LGADR. The RET_VAL return value of the SFC is made available for evaluation in the SFCERR variable at the instance DB.	Load the SFCERR tag from the instance DB.

Error code(W#16#...)	Description	Remedy
1E11	Static error during the call of SFC RDSYSST. The RET_VAL return value of the SFC is made available for evaluation in the SFCERR variable at the instance DB.	Load the SFCERR tag from the instance DB.
1E20	Parameter out of range.	Enter a parameter within the valid range for the instruction.
1E41	Invalid number of bytes specified at instruction parameter LEN	You must stay within a range of values of 1 to 256 bytes.

Evaluating the SFCERR variable

More detailed information on errors (1E) $0E_H$, (1E) $0F_H$, (1E) 10_H and (1E) 11_H belonging to event class 30 can be obtained via the SFCERR variable.

You can load the SFCERR variable from the instance DB of the corresponding instruction.

Error messages entered in the SFCERR variable can be found in the system functions "DPRD_DAT: Read consistent data of a DP standard slave (Page 6498)" and SFC15 "DPWR_DAT: Write consistent data of a DP standard slave (Page 6500)" and RD_LGADR: Determine all logical addresses of a module (Page 6787).

ERROR_NR, ERROR_INFO parameters

Error classes for diagnostics

Error numbers from ... to	Description	Meaning
1 ... 9	Instruction and CP initialization error	Error numbers 1 to 9 indicate that initialization was terminated with errors. The CP_START_ERROR parameter is set to 1. Modbus communication to the master system is not possible.
10 ... 19	Error executing a function code	Error numbers 10 to 19 indicate that an error occurred during the processing of a function code. The CP transmitted an invalid execution job to the communication instruction. The error is also reported to the driver. Subsequent execution jobs continue to be processed.
90 ... 99	Other faults/errors	A processing error has occurred. The error is not reported to the driver. Subsequent execution jobs continue to be processed.

Error numbers

ERROR_ No (decimal)	ERROR_INFO	Error Text	Remedy
0	0	No error	
1	SFC 51->RET_VAL	Error reading the system status list with SFC 51.	Analyze RET_VAL in ERROR_INFO; eliminate the cause.
2	S_SEND->STATUS, S_RCV->STATUS	Timeout or error during module initialization (error in S_SEND request).	Check if this interface was assigned parameters for operation with "Modbus slave" protocol. Verify the "ID" specified in the communication instruction. Analyze ERROR_INFO.
11	Start address	The driver transferred an invalid start address to the communication instruction.	Check the Modbus address of the Modbus master system.
12	Number of registers	The driver transferred an invalid number of registers to the communication instruction. Number of registers = 0.	Check the number of registers of the Modbus master system. If necessary, restart the module (Mains_ON).
13	Number of registers	The driver transferred an invalid number of registers to the communication instruction: Number of registers > 128.	Check the number of registers of the Modbus master system. If necessary, restart the module (Mains_ON).
14	Bit memory M – End address	Attempt to access the SIMATIC flags memory area beyond the end of the area. Notice: The area length in the SIMATIC CPU depends on the type of CPU.	Reduce the length of the Modbus start address or the access length in the Modbus master system.
15	Outputs Q – End address Inputs I – End address	Attempt to access the SIMATIC outputs memory area beyond the end of the area. Notice: The area length in the SIMATIC CPU depends on the type of CPU.	Reduce the length of the Modbus start address or the access length in the Modbus master system.
16	Timers T – End address	Attempt to access the SIMATIC timers memory area beyond the end of the area. Notice: The area length in the SIMATIC CPU depends on the type of CPU.	Reduce the length of the Modbus start address or the access length in the Modbus master system.
17	Counters C – End address	Attempt to access the SIMATIC counters memory area beyond the end of the area. Notice: The area length in the SIMATIC CPU depends on the type of CPU.	Reduce the length of the Modbus start address or the access length in the Modbus master system.

ERROR_ No (decimal)	ERROR_INFO	Error Text	Remedy
18	0	The driver transferred an invalid SIMATIC memory area to the communication instruction.	If necessary, restart the module (Mains_ON).
19		Error during access to SIMATIC I/Os.	Check if the required I/Os exist and are error-free.
20	DB#	DB does not exist.	Add the DB to your project.
21	DB#	DB length invalid	Increase the DB length.
22	DB#	DB# is below the minimum DB value.	Change the minimum DB value.
23	DB#	DB# is above the maximum DB value.	Change the maximum DB value.
24	Flag address	Flag is below the lower limit.	Change the lower limits of the flags in the conversion DB.
25	Flag address	Flag is above the upper limit.	Change the upper limits of the flags in the conversion DB.
26	Output address	Output is below the lower limit.	Change the lower limits of the outputs in the conversion DB.
27	Output address	Output is above the upper limit.	Change the upper limits of the outputs in the conversion DB.
90	S_SEND-> STATUS	Error during transmission of an acknowledgement message frame to the driver with S_SEND.	Analyze the STATUS information.
94	S_RCV->STATUS	Error reading SYSTAT with S_RCV (STATUS).	Analyze the STATUS information.

ET 200S 1SI for USS

S_USST: Send data to a USS slave

Description

The S_USST instruction handles the transmission of network data (PZD and any PKW data) to the slaves, depending on the network data structure used.

Operating principle

The instruction fetches the parameter assignment of the current slave from the polling list (parameter DB) and transmits the data from the network data DB. It evaluates the communication control word of the current slave (triggering of a PKW request/ acknowledgement of a parameter change report), completes the USS transmit data and transmits it to the send buffer of the communications processor DB. It then initiates network data transmission to the slave by calling the S_SEND instruction.

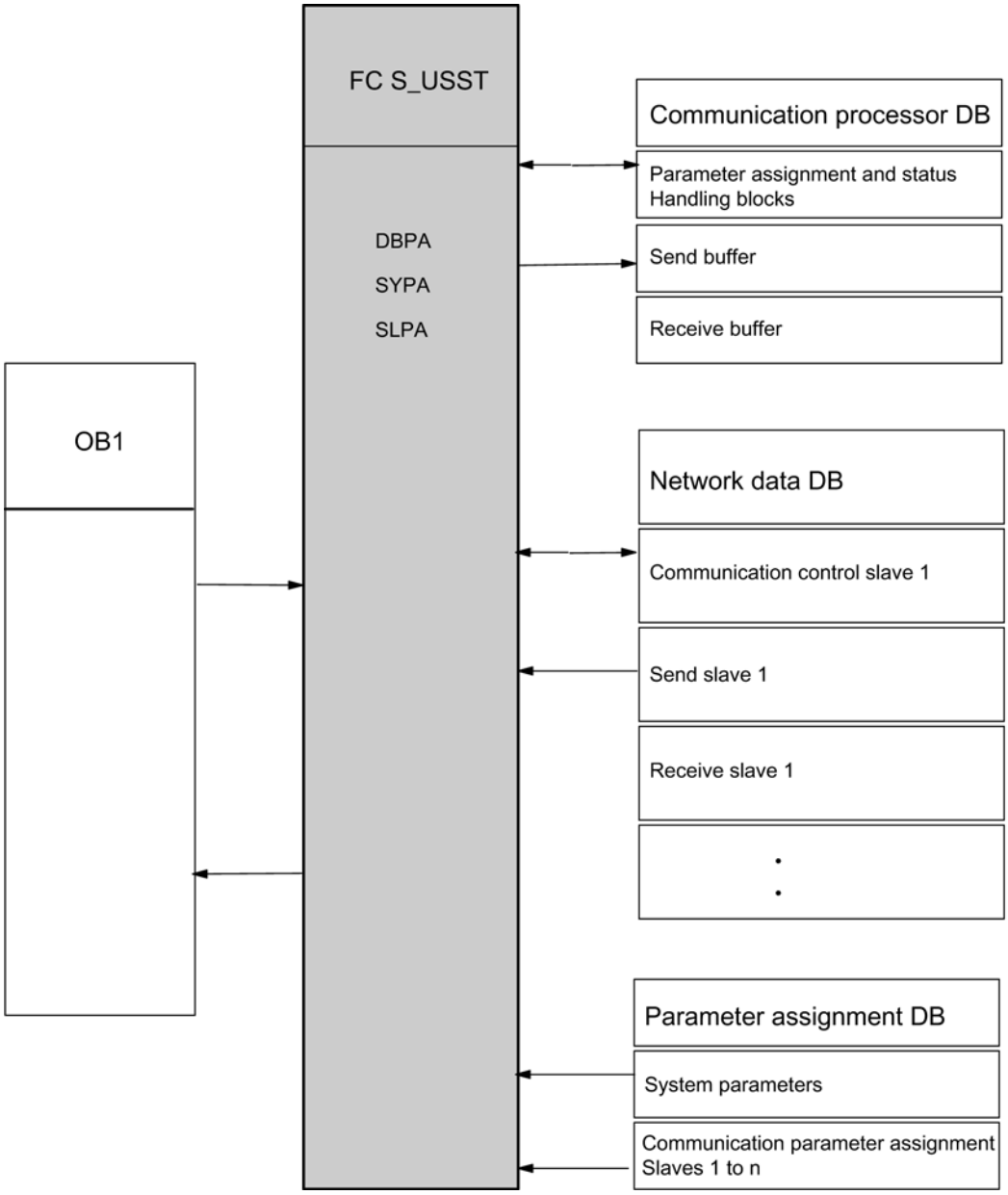
If the instruction detects a parameter assignment error in the parameter DB, an error signal is stored in the parameter assignment error 2 byte of the network data DB.

The instruction is called once per automation system cycle.

Parameters

Parameters	Declaration	Data type	Description
DBPA	INPUT	INT	Block number of the parameter DB CPU-specific (zero is not allowed)
SYPA	INPUT	INT	Start address of the system parameters in the parameter DB 0 <= SYPA <= 8174
SLPA	INPUT	INT	Start address of the slave parameters in the parameter DB 0 <= SLPA <= 8184

Program structure of S_USST



See also

Overview of USS communication (Page 1149)

S_USSR: Receive data from a USS slave

Description

The S_USSR instruction handles the receipt of network data (PZD and any PKW data) from the slaves, depending on the network data structure used.

Operating principle

The instruction fetches the parameter assignment of the current slave from the polling list (parameter DB) and evaluates the status word of the TRANSMIT block.

If the current request has been terminated without errors (bit 9 = 0 in the communication status word of the network data DB), the incoming data from the receive buffer of the communications DB is transmitted to the network data DB and evaluated. The communication status word is then updated in the network data DB.

If the current request has been terminated with an error (bit 9 = 1 in the communication status word of the network data DB), the data of the current slave is not accepted from the receive buffer of the communications processor DB. The instruction specifies this in the communication status word of the network data DB and enters the cause of error in the communication error word.

If the block detects a parameter error in the parameter DB, an error signal is stored in the parameter error 1 byte of the network data DB.

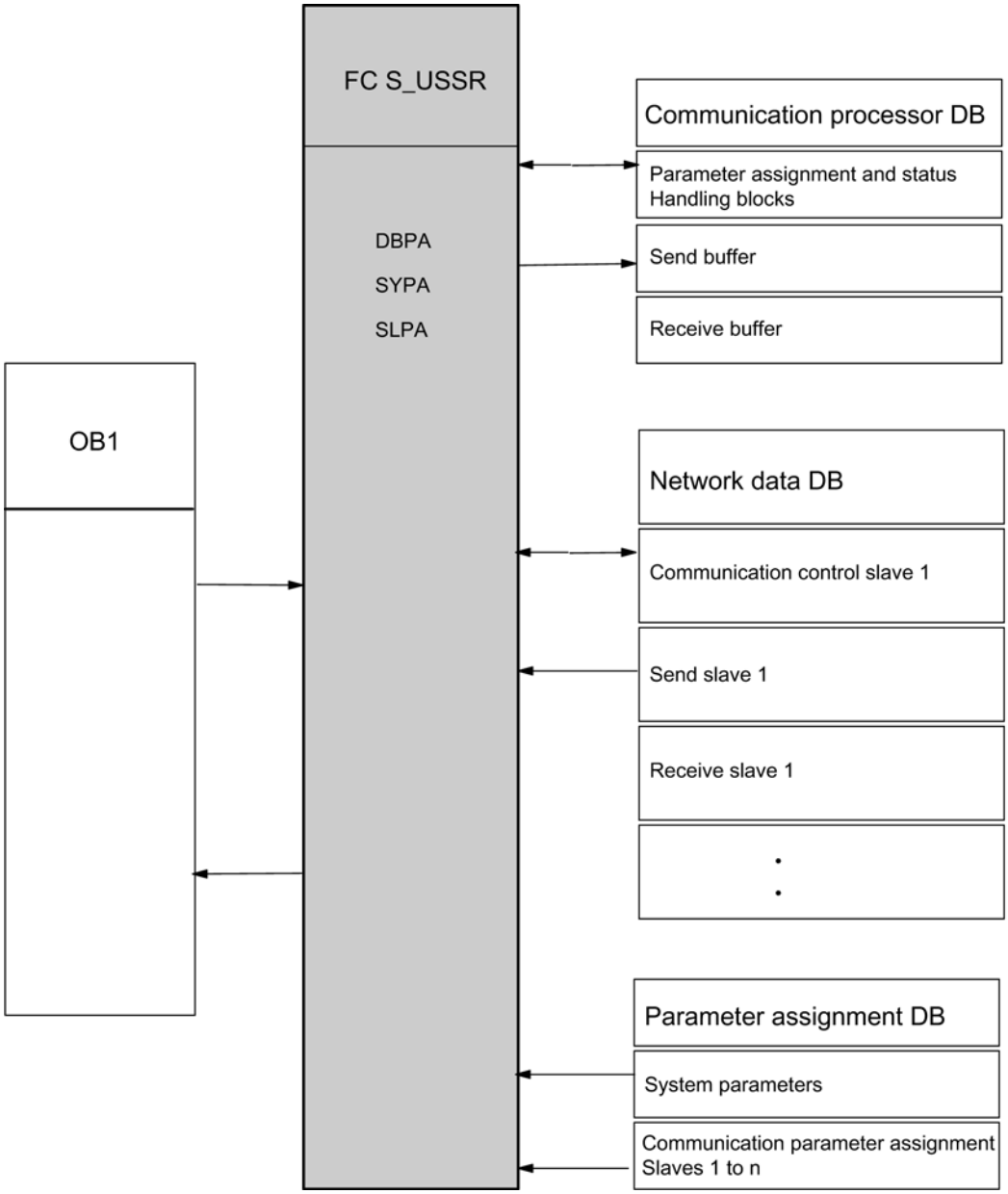
The instruction is called once per automation system cycle.

Parameters

Parameters	Declaration	Data type	Description
DBPA	INPUT	INT	Block number of the parameter DB CPU-specific (zero is not allowed)
SYPA	INPUT	INT	Start address of the system parameters in the parameter DB 0 <= SYPA <= 8174
SLPA	INPUT	INT	Start address of the slave parameters in the parameter DB 0 <= SLPA <= 8184

Program syntax of S_USSR

The diagram below shows the program syntax of S_USSR.



Rule

Note

The parameters of the U_USST instruction correspond to the parameters of the S_USSR instruction. The two functions access the same configuration (system and slave parameters) in the parameter DB and, consequently, have to be assigned identical parameters.

See also

Overview of USS communication (Page 1149)

S_USSI: Initialize USS

Description

The S_USSI instruction is an optional function.

Call of this instruction during the startup of the S7 system initiates the generation of the communications processor DB, network data DB and parameter DB. These data blocks are indispensible for communication. The DBPA is also preset. The S_USSI instruction can be used to generate and preset the specified data area, provided all slaves have the same network data structure.

Operating principle

When called, the instruction first checks the plausibility of its parameter assignment in terms of the number of slaves, network data structure, start node number and PKW repetitions. If the block detects an error during this, the data blocks are not generated or preset. The CPU goes into STOP mode and the user receives an error message via the error byte of the S_USSI instruction. Once the parameter error has been eliminated, all data blocks that have already been generated must be deleted before a restart.

After the plausibility check, the block checks whether the data blocks to be generated already exist:

- If the data blocks to be generated do not already exist, they are generated and the DBPA is preset.
- If the data blocks to be generated already exist, the length of each data block is checked. If the DB is long enough, the parameter DB is preset again and the contents of the network data DB and the communications processor DB are deleted. If a DB is too short, the CPU goes into STOP mode. You can identify the faulty DB by the condition code byte of the S_USSI instruction. For the error to be eliminated, the three data blocks must be completely deleted. The data blocks are then generated again at the next restart and the parameter DB is preset.

The S_USSI instruction must be called once during system startup (OB 100).

Parameters

Parameters	Declaration	Data type	Description
SANZ	INPUT	INT	Number of slaves with the same network data structure (system parameters in the DBPA) 1 <= SANZ <= 31
TNU1	INPUT	INT	Start node number (station number) 0 <= TNU1 <= 31

Parameters	Declaration	Data type	Description
PKW	INPUT	INT	PKW, number Number of words of the PKW interface 0, 3 or 4
PZD	INPUT	INT	PZD, number Number of words PZD interface 0 <= PZD <= 16
DBND	INPUT	INT	Network data DB number CPU-specific (zero is not allowed)
DBPA	INPUT	INT	Parameter DB number CPU-specific (zero is not allowed)
DBCP	INPUT	INT	Communications processor DB number CPU-specific (zero is not allowed)
WDH	INPUT	INT	Number of permissible PKW request repetitions 0 <= WDH <= 32,767
ANZ	OUTPUT	BYTE	Error byte 0: No error 1: Too many slaves 2: Invalid data for the network data structure 3: Configuration DB: Invalid number or DB too short 4: Network DB: Invalid number or DB too short 5: Incorrect station number 6: Communication processor DB: Invalid number or DB too short 7: Free 8: Repetition counter: Incorrect value

See also

Overview of USS communication (Page 1149)

Network data DB

Description

The network data DB forms the interface between the communication and control programs. The user must make this block available as "empty" and it must be sufficiently long.

Operating principle

You can generate and preset these blocks by calling the S_USSI instruction during CPU startup (DBPA only) or enter them manually.

Only the transmit data for a slave is entered in the network data DB send buffer that is assigned to the slave by the control program. The response data from the slave is accepted from the appropriate reception buffer (after evaluation of bit 9 in the communication control word). Status

words can be used to precisely monitor communication, while the control word allows you to precisely control the beginning of a parameter assignment request.

The communication interface contains the following data for each slave:

- Slave-related communication data (communication control, tracking, 6 data words)
- Buffer for the current PKW job (only if a PKW area exists)
- Send buffer for network data (maximum of 20 data words)
- Reception buffer for network data (maximum of 20 data words)

The lengths of the send and receive buffer depend on the network data structure selected. If the PKW interface does not exist, the buffer for the current PKW job is not used.

The total length of the network data DB required depends on the number of slaves and the network data structure used.

Number of data words per slave = 2 x (PKW + PZD) + PKW + 6

where PKW = 0, 3 or 4 and 0 <= PZD <= 16

Example: A drive with a PKW area of 3 words and a PZD area of 2 words requires 19 data words in the network data DB.

With 31 slaves and the maximum network data length, the network data DB is 1,550 data words long. DBW0 is reserved.

Slave data assignment in the network data DB with 4 words in the PKW area and 0 to 16 words in the PZD area.

<table>
<tr><td>DBWn</td><td colspan="2">Communication control word (KSTW)</td><td colspan="2" rowspan="6">Communication control

Communication tracking
Error status
PKW attempt counter
Parameter error</td></tr>
<tr><td>DBWn+2</td><td colspan="2">Internal</td></tr>
<tr><td>DBWn+4</td><td colspan="2">Communication status word</td></tr>
<tr><td>DBWn+6</td><td colspan="2">Communication error word</td></tr>
<tr><td>DBWn+8</td><td colspan="2">Internal</td></tr>
<tr><td>DBWn+10</td><td colspan="2">Parameter error 1 byte, parameter error 2 byte</td></tr>
<tr><td>DBWn+12</td><td>Parameter ID</td><td>PKE</td><td colspan="2" rowspan="4">Buffer for current
PKW job</td></tr>
<tr><td>DBWn+14</td><td>Index</td><td>IND</td></tr>
<tr><td>DBWn+16</td><td>Parameter value 1</td><td>PWE1</td></tr>
<tr><td>DBWn+18</td><td>Parameter value 2</td><td>PWE2</td></tr>
<tr><td>DBWn+20</td><td>Parameter ID</td><td>PKE</td><td rowspan="4">PKW area</td><td rowspan="8">Send buffer</td></tr>
<tr><td>DBWn+22</td><td>Index</td><td>IND</td></tr>
<tr><td>DBWn+24</td><td>Parameter value 1</td><td>PWE1</td></tr>
<tr><td>DBWn+26</td><td>Parameter value 2</td><td>PWE2</td></tr>
<tr><td>DBWn+28</td><td>Control word (STW)</td><td>PZD1</td><td rowspan="4">PZD area
(max. 16 words PZD)</td></tr>
<tr><td>DBWn+30</td><td>Main setpoint (HSW)</td><td>PZD2</td></tr>
<tr><td>DBWn+32</td><td>Setpoint/Additional control word</td><td>PZD3</td></tr>
<tr><td>DBWn+34</td><td>Setpoint/Additional control word</td><td>PZD4</td></tr>
<tr><td>...</td><td colspan="2">...</td><td></td></tr>
</table>

DBWn+58	Setpoint/Additional control word	PZD16	PKW area	Receive buffer
DBWn+60	Parameter ID	PKE		
DBWn+62	Index	IND		
DBWn+64	Parameter value 1	PWE1		
DBWn+66	Parameter value 2	PWE2		
DBWn+68	Status word (ZSW)	PZD1	PZD area (max. 16 words PZD)	
DBWn+70	Main actual value (HIW)	PZD2		
DBWn+72	Actual value/Additional status word	PZD3		
DBWn+74	Actual value/Additional status word	PZD4		
...	...			
DBWn+98	Actual value/Additional status word	PZD16		
	•			
(n = 2, 4 , 6...)	•			

Note

If there is no PKW area, neither the buffer for current PKW jobs nor the PKW area in the send buffer exists.

Communication control word KSTW (DBWn)

The bits in the communication control word coordinate the user program and the S_USST instruction.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

- Bit 0: Triggers the PKW job
  Bit 0 is set by the user if a new PKW job is in the send buffer and is to be processed. The instruction resets the bit if the PKW job is accepted.
- Bit 1: Acceptance of parameter change report
  Bit 1 is set by the user if the parameter change report has been accepted. The instruction resets the bit in order to acknowledge acceptance. Following this acknowledgement, the slave either continues to process the current job or transmits the next parameter change report.

Communication status word (DBWn+4)

The S_USST and S_USSR instructions set the bits in the communication status word.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

- Bit 0: PKW job in progress
  If the PKW job was accepted and the parameter ID (PKE) contains a valid job ID, instruction S_USST sets the bit to 0. Instruction S_USSR resets the bit after the PKW job has been executed (with or without errors) or if there are problems with the PKW interface.
- Bit 1: PKW job completed without errors
  Instruction S_USSR sets bit 1 if a PKW job has been executed without errors. The response should be taken from the receive buffer. Instruction S_USST resets the bit when a new PKW job is initiated.

  Note

  The PKW jobs for the slave are processed in the order specified in the polling list (DBPA). Only one job is active for each slave at any one time. If more than one slave is entered in the polling list, the response data for a new PKW job is only available on a positive edge of bit 1 (or bit 2).

- Bit 2: PKW job completed with errors
  Instruction S_USSR sets bit 2 for the response ID in the PKE. The error number is in the PWE of the slave response. Instruction S_USST resets the bit when a new PKW job is initiated.

  Note

  The last PKW job transmitted by the user is stored in the send interface after processing. Transmission to the slave is repeated until a new job is entered. This might require additional responses in the user program in the event of the status PKW job being terminated with an error (bit 2) and a PKW interface error (bit 4).

- Bit 3: PKW job ID is invalid.
  Instruction S_USST sets bit 3 if job ID 15 is set in the PKE or index 255 is entered in job ID 4. Instruction S_USST resets the bit if the next PKW job is triggered with a valid job ID in the PKE.
- Bit 4: PKW interface with an error (counter overflow).
  Instruction S_USSR sets bit 4 if the slave does not respond to the PKW job within a configurable number of job repetitions (WDH parameter in the parameter DB) or in the case of response ID 8 in the PKE. Instruction S_USSR resets the bit when a new PKW job is initiated and correctly executed.
- Bit 5: Response data contains a parameter change report.
  Instruction S_USSR sets bit 5 if a parameter change report from the slave is present (response IDs 9 to 12 and toggle bit 11 inverted). Instruction S_USST resets the bit when the user acknowledges the parameter change report (communication control word, bit 1).
- Bit 6: Operational fault on the slave.
  Instruction S_USSR sets and resets bit 6. The instruction evaluates the slave's status word (bit 3).

- Bit 7: There is a warning from the slave.
  Instruction S_USSR sets and resets bit 7. The instruction evaluates the slave's status word (bit 7).
- Bit 8: Automation system control requested.
  Instruction S_USSR sets and resets bit 8. The instruction evaluates the status word (bit 9) and the control word (bit 10).
- Bit 9: Group communication fault.
  Instruction S_USSR sets and resets bit 9. The instruction evaluates the feedback messages from the S_SEND and S_RCV standard blocks and checks the message frame received with respect to ADR, STX, BCC and LGE. The instruction also reports a timeout of the message frame monitoring time.

Note

The receive data from the network data DB is only valid where bit 9 = 0.

Structure of the communication error word (DBWn+6)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

- Bit 0: Addressing error (ADR)
- Bit 3: Start of message frame not detected (no STX for first character)
- Bit 4: Incorrect block check character (BCC)
- Bit 6: Incorrect message frame length (LGE)
  Instruction S_USSR sets bits 0, 3, 4 and 6 if an error is detected in the received frame (ADR, STX, BCC, LGE).
- Bit 7: Message frame monitoring time has elapsed
  Instruction S_USSR sets bit 7 if the time between the transmission of the message frame from the master to the slave and the arrival of the response from the slave exceeds the valid time calculated by the program (frame monitoring time).

The remaining bits are not used.

Parameter error 1 byte

Error message from the S_USSR instruction; parameter assignment error in parameter DB

- Value 0: No error
- Value 1: Incorrect data for PKW/PZD

Parameter error 2 byte

Error message from the S_USST instruction; parameter assignment error in parameter DB

- Value 0: No error
- Value 1: Incorrect data for PKW/PZD

Parameter ID PKE in the send buffer

The user must assign the parameter number (bits 0 to 10) and the job ID (bits 12 to 15). The toggle bit for the parameter change report (bit 11) is masked by the S_USSR and S_USST instructions.

Network data DB

Description

The arameter DB contains the program parameters required for communication control. The user needs to generate this block and set suitable defaults for the communication system (S_USSI or manually). The slaves on the bus are processed in the sequence in which they are entered in the DBPA (polling list).

A slave can also be entered more than once in the parameter DB in order to effectively step up its priority class.

Operating principle

The length of the parameter DB depends on the number (n) of slaves to be addressed in a bus cycle.
Number of data words of the parameter DB = (n x 4) + 5.

4 data words are required for each instance of slave communication and 4 data words are assigned once for the system parameters. DBW0 is reserved.

Data block allocation

DBW 0	Free	System parameters
DBW 2	DBCP	
DBW 4	SANZ	
DBW 6	SLAV	
DBW 8	WDH	
DBW 10	Number of PKW, number of PZD	Communications Parameter set slave 1
DBW 12	TUN	
DBW 14	DBND	
DBW 16	KSTW	
DBW 18	Number of PKW, number of PZD	Communications Parameter set slave 2
DBW 20	TUN	
DBW 22	DBND	
DBW 24	KSTW	
	Number of PKW, number of PZD	Communications Parameter set slave n
	TUN	

	DBND	
DBW (n x 8 + 8)	KSTW	

System parameters

DBCP	Block number of the communications processor DB
SANZ	Total number of slave parameter sets in the parameter DB. If some individual slaves are to be addressed more frequently than others in a bus cycle, their slave parameters must be entered more than once in the parameter DB. The SANZ system parameter must be adjusted accordingly.
SLAV	Number of the current slave (consecutive). Required by the S_USST FC and the S_USSR FC to calculate the current parameter set. This data word must be preset to 1. This is carried out by the S_USSI FC, if it is being used.
WDH	Number of permissible PKW job repetitions (value range: 0 through 32,767). If the current PKW job is not terminated within the set number, problems at the PKW interface are reported.

Slave communication parameter

Number of PKW, number of PZD	Definition of network data structure Left byte: Number of words for PKW area (0, 3, 4) Right byte: Number of words for PZD area (0 to 16) Any deviations from this are detected as parameter errors (by the S_USST and S_USSR FCs) and entered in the parameter error 1 byte, parameter error 2 byte of the network data DB.
TUN	Node number corresponding to the bus address set on the drive (0 to 31).
DBND	Block number of the network data DB.
KSTW	Address of the communication control word KSTW for the slave in the network data DB.

Communication processor DB

Description

This data block is used to handle data traffic between the CPU and serial interface module ET 200S Modbus/USS. Provide this block with a sufficient length. The communication processor DB must have a length of at least 50 words (DBW 0 to 98).

Syntax of the communication processor DB

DBW 0	Communication status		SEND and RECEIVE
DBW 2	Maximum number of cycles to expire when waiting to receive data	Cycle counter for timeout formation when waiting to receive data	FC17

DBW 4	Measured start interval	FC17
DBW 6	Duration of the last cycle (OB1_MIN_CYCLE)	FC17, OB1
DBW 8	Transmission frame length (LEN)	SEND
DBB10	Free	
DBB 11 : : DBB 54	Send buffer	Send frame to module (length is determined by the network data structure of the current slave)
DBB 55 : : DBB 98	Receive buffer	Received frame from module (length is determined by the network data structure of the current slave)

Communication status DBW0

Bit no.		
0	Input REQ for S_SEND	This bit is reset if bit 8 is set.
1	Input R for S_SEND	S_USST resets this bit at cyclic intervals.
2	Output DONE of S_SEND	
3	Output ERROR of S_SEND	
4	Input EN_R for S_RCV	S_USSR sets this bit at cyclic intervals.
5	Input R for S_RCV	S_USSR resets this bit at cyclic intervals.
6	Output NDR of S_RCV	
7	Output ERROR of S_RCV	
8	Job in progress (S_SEND saves the DONE bit)	S_USST resets this bit at cyclic intervals.

Duration of the last cycle DBW6

S_USST uses this parameter to measure slave response times. Before each call of S_USST , the user program must copy the cycle time of the automation system (OB1_MIN_CYCLE) to this parameter.

SIMATIC NET CPs

Industrial Ethernet

Introduction to Ethernet

General notes on the FCs / FBs

Available instructions

The following list shows the numbers of the program blocks (instructions) as supplied. You can change these numbers.

Please note that you must use different instructions the S7300 and S7400 (separate libraries).

Communication service / functional area	Type of instruction	Library	
		CP 300	CP 400
SEND / RECEIVE (open communications services)	AG_SEND	x	x
	AG_RECV	x	x
	AG_LSEND	x [2]	x
	AG_LRECV	x [2]	x
	AG_LOCK	x	x
	AG_UNLOCK	x	x
	AG_CNTEX	x	x [3]
	AG_CNTRL	x	x [3]
Programmed communication connections	IP_CONFIG	x	x
S7 communication	BSEND	x	
	BRCV	x	
	PUT	x	
	GET	x	
	USEND	x	
	URCV	x	
	C_CNTRL	x	
FTP	FTP_CMD	x	x
PROFINET IO	PNIO_SEND	x	
	PNIO_RECV	x	
	PNIO_RW_REC	x	
	PNIO_ALARM	x	

1) PN_InOut / PN_InOut_Fast is supplied along with the SIMATIC iMap engineering tool.

2) Not to be used with the current CPs and no longer part of the current SIMATIC_NET_CP library

3) Depending on the CP type

Which version of the instructions should I use?

The following descriptions also include information on differences in behavior between the various versions of the instructions. Please check and note the version identifiers of the instructions you are using.

The latest release of the version history for SIMATIC NET instructions can be found on the Internet using the following entry ID:

9836605 (http://support.automation.siemens.com/WW/view/de/9836605)

The global libraries installed with the Automation Workbench contain the versions of the instructions current when the Workbench was released.

Note

We recommend that you always use the latest versions of the instructions for all module types.

You will find information on the current versions of the instructions and the current instructions to download from the Internet in our customer support:

http://support.automation.siemens.com/WW/view/en/8797900

With older module types, this recommendation assumes that you are using the latest firmware for the particular module type.

Instructions when modules are replaced

Module replacement in this sense means the replacement of a module with another module that may be a more recent version.

Note

Remember that if you replace a module, you must only use the instructions permitted for the configured CP type in the user program.

We recommend that you always use the latest versions of the instructions for all module types. With older module types, this recommendation assumes that you are using the latest firmware for the particular module type.

You will find more information on replacing modules in our Customer Support on the Internet.

The manuals for specific S7 CPs contain information on the compatibility of the S7CPs and the corresponding instructions.

See also

Programming manual "Functions (FC) and Function Blocks (FB) for SIMATIC NET S7 CPs" (http://support.automation.siemens.com/ww/view/de/30564821)

Setting parameters for FC calls

Before describing the instruction in detail, a few general comments on calling and setting parameters for instructions will be useful at this point.

The general information below applies to the following parameter groups that exist for all instructions:

- Parameters for CP and connection assignment (input parameters)
- Parameters for specifying a CPU data area (input parameters)
- Status information (output parameters)

Calling communications instruction for an S7300

NOTICE
Calling the communications instructions for S7-300 in more than one priority class is not permitted! If, for example, you call a communication instruction in OB1 and in OB35, execution of the instruction could be interrupted by the higherpriority OB. If you call instructions in more than one OB, write your program so that a communications instruction that is currently executing cannot be interrupted by another communications instruction (for example by disabling/enabling interrupts with system instructions).

Parameters for specifying a CPU data area (input parameters)

Specifying the data area on the CPU

When you call an instruction, you transfer the address and length of the data area on the CPU in which the user data is available or will be stored or which can contain further parameter information

The ANY pointer data type is used to address this area.

Status information (output parameters)

Evaluating status codes

For status evaluation, the following parameters must be evaluated in the user program:

- DONE or NDR
  These parameters (DONE with send jobs and NDR with receive jobs) signal (successful) completion of the job.
- ERROR
  This indicates that the job could not be executed errorfree.
- STATUS
  This parameter supplies detailed information about the execution of the job. Status codes can be returned during execution of the job (DONE=0 and ERROR=0).

Note

Remember that the status codes DONE, NDR, ERROR, STATUS are updated at each instruction call.

Status codes during CP startup

With a warm or hot restart on the Ethernet CP, the output parameters of the instruction are reset as follows:

- DONE = 0
- NDR = 0
- ERROR = 0
- STATUS =
  - 8180_H for AG_RECV / AG_LRECV
  - 8181_H for AG_SEND / AG_LSEND / AG_SSEND

Instructions for open communications services (SEND/RECEIVE interface)

Instructions for the SEND/RECEIVE interface

Overview

The following instructions are available for transferring data on the SEND/RECEIVE interface:

Instruction	Can be used with [1]		Meaning
S7-300	S7-400		
AG_SEND	x	x	for sending data
AG_RECV	x	x	for receiving data
AG_LSEND		x	for sending data

Instruction	Can be used with [1]		Meaning
AG_LRECV		x	for receiving data
AG_SSEND		x	for sending data
AG_SRECV		x	for receiving data

1) Notes on the Instructions for an S7300 and S7400

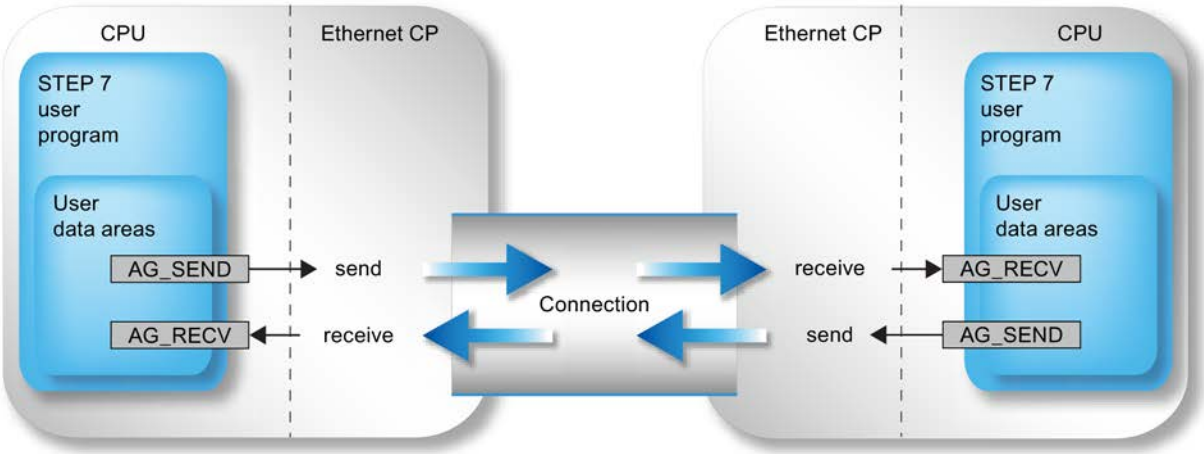
- The following applies to S7300:
  - With the latest versions of the Ethernet CPs, only AG_SEND and AG_RECV are used; the data length can be up to 8192 bytes.
  - With S7–300 CPs (up to 6GK7 343–1EX10–0XE0 with firmware version V2.2), use AG_LRECV on TCP connections instead of AG_RECV.
- The following applies to S7400:
  - With AG_SEND / AG_RECV, the data length per job is restricted to <=240 bytes. Longer data records (up to 8192 bytes) can be transferred with the AG_LSEND or AG_LRECV instructions.
  - AG_SSEND and AG_SRECV are for accelerated transfer of data by using optimized communication between CPU and CP in the S7 station. The fast communication has no effect on LAN communication. These two instructions are supported as of STEP 7 V5.4 SP3.
  - On an S7–400, AG_RECV cannot be used on TCP connections but only the AG_LRECV or AG_SRECV instructions.

Further information

Please check the supported data area for the S7CP you are using in the manual for the specific device. You will find an overview of the versions of the instructions in the SIMATIC NET block history.

Application

The following diagram illustrates the use of the instructions described here for bidirectional data transfer on a configured connection.



Specifying the data area on the CPU

When you call an instructions, you transfer the address and length of the data area in the CPU. Remember, that the maximum length of the data area depends on the type and version of the instruction being used.

- AG_SEND and AG_RECV
  Up to version V3.0 of these instructions, a maximum of 240 bytes can be sent or received. The current versions allow a data area of up to 8192 bytes for an S7300. With an S7400, AG_LSEND / AG_LRECV must still be used for larger data areas.
- AG_LSEND / AG_LRECV
  Using the CPs of the S7400 and with earlier versions of the S7300, larger data areas can only be transferred with AG_LSEND or AG_LRECV. Please check the length of the data area in the product information of the CP.
- AG_SSEND / AG_SRECV
  With CPs of the S7−400 that support PROFINET communication in conjunction with CPUs as of version 5.1, data can be transferred at higher transmission speeds with AG_SSEND or AG_SRECV (does not apply to the CP 443−1 Advanced 6GK7 443–1EX41–0XE0). You can check which CP types are supported by CPUs as of version 5.1 in the manual of your CP (Section "Requirements for use").

The following table shows the limit values of the various connection types.

INSTRUCTION	ISO transport	ISO-on-TCP	TCP	UDP
AG_LSEND (S7-400) AG_SEND (S7-300)	8192 bytes	8192 bytes	8192 bytes	2048 bytes
AG_SEND (S7-400)	240 bytes	240 bytes	240 bytes	240 bytes
AG_LRECV (S7-400) AG_RECV (S7-300)	8192 bytes	8192 bytes	8192 bytes	2048 bytes

INSTRUCTION	ISO transport	ISO-on-TCP	TCP	UDP
AG_RECV (S7-400)	240 bytes	240 bytes	240 bytes	240 bytes
AG_SSEND (S7-400) AG_SRECV (S7-400)	1452 bytes	1452 bytes	1452 bytes	1452 bytes

Note

For information on the length of the data area you can transfer with older versions of the Ethernet CPs, refer to the product information / manual of the Ethernet CP you are using.

Use without job header

On specified connections, the address and job parameters are specified by the connection configuration. The user program only provides the user data in the UDP data area when sending with AG_SEND / AG_LSEND / AG_SSEND or receives the data with AG_RECV / AG_LRECV / AG_SRECV.

Use with header

Free UDP connections require a job header in the user data area.

The following schematic illustrates the structure of the job buffer and the meaning and location (high byte / low byte) of the parameters in the job header.

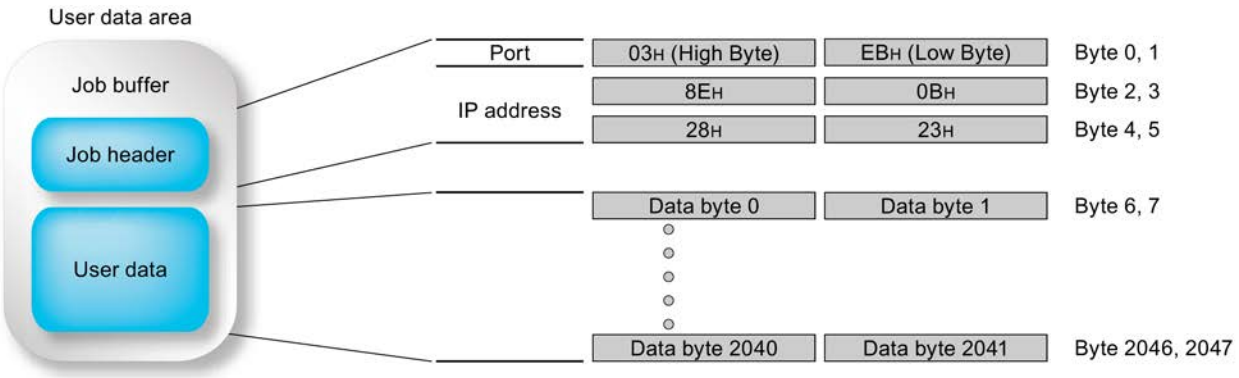


Figure 9-19 Sending and receiving on a free UDP connection with programmed addresses

- In the diagram (entries in hexadecimal) the following IP address is assumed as an example: 142.11.40.35;
- For the port address 1003, the following would be entered: For high byte: 03_H; For low byte: EB_H.
- The user data area can be up to 2048 bytes. Up to 2042 bytes of user data can be transferred. 6 bytes are reserved for the job header.
  Please note that the data length specified in the instruction call (LEN parameter) must include the header and the user data!

Change call parameters only after job confirmation

Note

Once the job has been triggered, you can only change the call parameters of the call interface of the AG_SEND or AG_RECV instruction after the instruction has confirmed completion of the job with DONE=1 or with ERROR=1.

If you do not keep to this rule, it is possible that the job will be aborted with an error.

Status display on the call interface of the instruction; Special situations with AG_SEND and AG_RECV (only for S7-300) *)

With AG_SEND and AG_RECV, you receive the following status codes for the operating situations mentioned:

- CP is in STOP;
- Connection is not configured;
- Connection is not established
- Connection is aborted;

Codes:

- AG_SEND:
  DONE=0; ERROR=1; Status=8183_H
- AG_RECV:
  DONE=0; ERROR=0; Status=8180_H
  or
  DONE=0; ERROR=1; Status=8183_H

*) applies to instructions as of version 4.0

See also

General notes on the FCs / FBs (Page 7712)

AG_SEND / AG_LSEND / AG_SSEND

AG_SEND / AG_LSEND / AG_SSEND (Ind. Ethernet)

Description

AG_SEND / AG_LSEND / AG_SSEND pass data to the Ethernet CP for transfer over a configured connection.

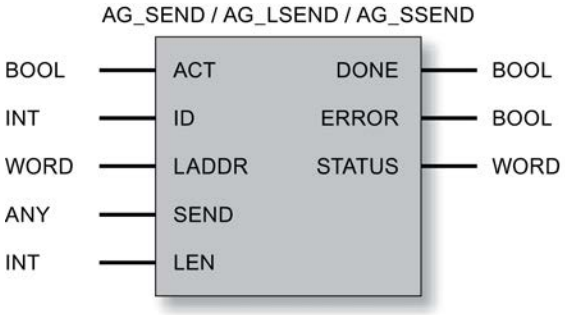
The selected data area can be a memory bit area or a data block area.

Errorfree execution is indicated when the entire user data area could be sent over Ethernet.

Note:
Unless otherwise stated, all the following information applies equally to AG_SEND, AG_LSEND and AG_SSEND.

Call

Call interface in FBD representation



Note

If you want to use the AG_SSEND instruction, you will need to select the "SPEED SEND/ RECV" mode in the connection properties during configuration of the connection.

See also

Parameters for AG_SEND / AG_LSEND / AG_SSEND (IE) (Page 7721)

Parameters DONE, ERROR, STATUS (Page 7722)

Programming manual "Functions (FC) and Function Blocks (FB) for SIMATIC NET S7 CPs" (http://support.automation.siemens.com/ww/view/de/30564821)

Parameters for AG_SEND / AG_LSEND / AG_SSEND (IE)

Explanation of the formal parameters

The following table explains all the formal parameters for the AG_SEND / AG_LSEND / AG_SSEND instructions:

Parameter	Declaration	Data type	Range of values	Description
ACT	INPUT	BOOL	0,1	If the instruction is called with ACT=1, LEN bytes are sent from the ISO transport data area specified with the SEND parameter. If the instruction is called with ACT = 0, the status codes DONE, ERROR and STATUS are updated.
ID	INPUT	INT	1,2...64 (S7-400) 1,2...16 (S7-300)	The connection number of the connection is specified in the parameter ID.
LADDR	INPUT	WORD		Module start address When you configure the CP with STEP 7, the module start address is displayed in the configuration table. Specify this address here.
SEND	INPUT	ANY		Specifies the address and length The address of the data area points to one of the alternatives: • Memory bit area • Data block area
LEN	INPUT	INT	On ISO transport and ISOonTCP / TCP: 1, 2 to 8192 (or up to "length specified for SEND parameter") On UDP: 1, 2 to 2048 (or up to "length specified for SEND parameter")	Number of bytes to be sent from the data area with this job. The possible values range from 1 to length specified for the SEND parameter. • Note the type of instruction: – For S7-300 The current versions of AG_SEND allow up to 8192 bytes (2048 bytes for UDP) to be transferred. – For S7-400 With AG_SEND, the data area is restricted to a maximum of 240 bytes. Note the following with an S7-400: • Improved performance with shorter data records: Transfer of data records up to 240 bytes results in better performance! This applies regardless of the type used (AG_SEND / AG_LSEND). • With AG_SSEND, the data area is restricted to a maximum of 1452 bytes.
DONE	OUTPUT	BOOL	0: Job active 1: Job done	The status parameter indicates whether or not the job was completed without errors. As long as DONE = 0, no further job can be triggered. When the job is accepted, DONE is set to 0 by the CP.

Parameter	Declaration	Data type	Range of values	Description
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning in conjunction with the parameters DONE and STATUS, refer to Parameters DONE, ERROR, STATUS (Page 7722)
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the parameters DONE and ERROR, refer to Parameters DONE, ERROR, STATUS (Page 7722)

See also

AG_SEND / AG_LSEND / AG_SSEND (Ind. Ethernet) (Page 7719)

Parameters DONE, ERROR, STATUS

Condition codes

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

Note

For entries coded with $8Fxx_H$ in STATUS, refer to the information about the output parameter RET_VAL in the descriptions of the referenced system program blocks

Which system program blocks are used and are relevant for error evaluation, can be queried in STEP 7.

Table 9-328 AG_SEND / AG_LSEND / AG_SSEND status codes

DONE	ERROR	STATUS	Meaning
1	0	0000_H	Job completed without error.
0	0	0000_H	No job being executed.
0	0	8181_H	Job active.
0	1	7000_H	The condition code is possible only with S7-400: The instruction was called with ACT=0; the job has not yet been processed.
0	1	8183_H	No configuration or the ISO/TCP service has not yet started on the Ethernet CP.
0	1	8184_H	• Illegal data type specified for the SEND parameter. • System error (the source data area is incorrect).
0	1	8185_H	LEN parameter longer than SEND source area.
0	1	8186_H	ID parameter invalid. • ID != 1, 2 to 16 (S7-300) • ID != 1, 2 to 64.(S7-400)
0	1	8302_H	No receive resources on the destination station. Receiving station cannot process received data quickly enough or has not made any receive resources available.

DONE	ERROR	STATUS	Meaning
0	1	8304_H	The connection is not established. The send job should only be attempted again after waiting for at least 100 ms.
0	1	8311_H	The destination station cannot be obtained under the specified Ethernet address.
0	1	8312_H	Ethernet error on the CP.
0	1	$8F22_H$	Source area invalid, e.g.: Area does not exist in the DB LEN parameter < 0
0	1	$8F24_H$	Area error reading a parameter.
0	1	$8F28_H$	Alignment error reading a parameter.
0	1	$8F32_H$	Parameter contains a DB number that is too high.
0	1	$8F33_H$	Wrong DB number
0	1	$8F3A_H$	Area not loaded (DB).
0	1	$8F42_H$	Timeout reading a parameter from the I/O area.
0	1	$8F44_H$	Access to a parameter to be read during execution of the instruction is prevented.
0	1	$8F7F_H$	Internal error, e.g. illegal ANY reference e.g. parameter LEN=0.
0	1	8090_H	• Module with this module start address does not exist; • The instruction being used does not match the system family being used (remember to use different instructions for S7300 and S7400).
0	1	8091_H	Module start address not at a doubleword boundary.
0	1	8092_H	In the ANY reference, a type other than BYTE is specified. (S7-400 only)
0	1	$80A4_H$	The communication bus connection between the CPU and CP is not established. (With newer CPU versions)
0	1	$80B0_H$	The module does not recognize the data record.
0	1	$80B1_H$	The specified length (in the LEN parameter) is incorrect.
0	1	$80B2_H$	The communication bus connection between the CPU and CP is not established.
0	1	$80C0_H$	The data record cannot be read.
0	1	$80C1_H$	The specified data record is currently being processed.
0	1	$80C2_H$	There are too many jobs pending.
0	1	$80C3_H$	CPU resources (memory) occupied.
0	1	$80C4_H$	Communications error Occurs temporarily and a repetition in the user program will often remedy the problem
0	1	$80D2_H$	Module start address incorrect.

See also

AG_SEND / AG_LSEND / AG_SSEND (Ind. Ethernet) (Page 7719)

AG_RECV / AG_LRECV / AG_SRECV

AG_RECV / AG_LRECV / AG_SRECV (Ind. Ethernet)

Description

The AG_RECV / AG_LRECV / AG_SRECV instruction receives the data transferred on a configured connection from the Ethernet CP.

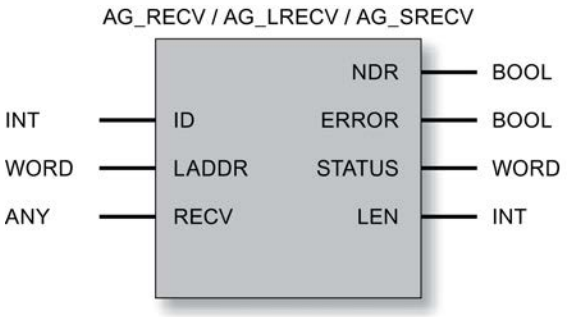
The data area specified for the receive data can be a memory bit area or a data block area.

Errorfree execution is indicated when the data could be received from the Ethernet CP.

Note:
Unless otherwise stated, all the following information applies equally to AG_RECV and AG_LRECV / AG_SRECV.

Call

Call interface in FBD representation



Note

If you want to use the AG_SRECV instruction, you will need to select the "SPEED SEND/ RECV" mode in the connection properties during configuration of the connection.

See also

Parameters for AG_RECV / AG_LRECV / AG_SRECV (IE) (Page 7725)

Parameters DONE, ERROR, STATUS (IE) (Page 7726)

Parameters for AG_RECV / AG_LRECV / AG_SRECV (IE)

Explanation of the formal parameters

The following table explains all the formal parameters for the AG_RECV / AG_LRECV / AG_SRECV instructions:

Parameter	Declaration	Data type	Range of values	Description
ID	INPUT	INT	1, 2 to 64 (S7-400) 1, 2 to 16 (S7-300)	The connection number of the ISO transport connection is specified in the ID parameter.
LADDR	INPUT	WORD		Module start address When you configure the CP with STEP 7, the module start address is displayed in the configuration table. Specify this address here.
RECV	INPUT	ANY		Specifies the address and length The address of the data area points to one of the alternatives: • Memory bit area • Data block area Note on length: Performance is improved when transferring data records up to 212 bytes if you also restrict the length to 212 bytes at the RECV parameter. Point to note with AG_SRECV: With AG_SRECV, always set RECV to the maximum receive buffer length of 1452 bytes. Otherwise, the following error can occur in certain situations: NDR=0; ERROR=1; STATUS=8185_H
NDR	OUTPUT	BOOL	0: - 1: new data	The parameter indicates whether or not new data was accepted. For the meaning in conjunction with the ERROR and STATUS parameters, refer to AG_RECV / AG_LRECV / AG_SRECV (Ind. Ethernet) (Page 7724)
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning in conjunction with the NDR and STATUS parameters, refer to AG_RECV / AG_LRECV / AG_SRECV (Ind. Ethernet) (Page 7724)

Parameter	Declaration	Data type	Range of values	Description
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the NDR and ERROR parameters, refer to AG_RECV / AG_LRECV / AG_SRECV (Ind. Ethernet) (Page 7724)
LEN	OUTPUT	INT	**On ISO Transport and ISOonTCP:** 1,2,...8192 **On UDP:** 1,2,...2048	Specifies the number of bytes accepted from the Ethernet CP and entered in the data area. Note the type of instruction: • For S7-300 The current versions of AG_RECV allow up to 8192 bytes (2048 bytes for UDP) to be transferred. • For S7-400 With AG_RECV, the data area is restricted to a maximum of 240 bytes. With AG_SRECV, the data area is restricted to a maximum of 1452 bytes.

Parameters DONE, ERROR, STATUS (IE)

Condition codes

The following table shows the codes formed by the NDR, ERROR and STATUS parameters that must be evaluated by the user program.

Note

For entries coded with $8Fxx_H$ in STATUS, refer to the information about the output parameter RET_VAL in the descriptions of the referenced system program blocks.

Which system program blocks are used and are relevant for error evaluation, can be queried in STEP 7.

Table 9-329 AG_RECV / AG_LRECV / AG_SRECV condition codes

NDR	ERROR	STATUS	Meaning
1	0	0000_H	New data accepted.
0	0	8180_H	There is no data available yet (not with AG_SRECV).
0	0	8181_H	Job active.
0	1	8183_H	• The configuration is missing; • The ISO transport service has not yet started on the Ethernet CP; • The connection is not established.
0	1	8184_H	• Illegal type specified for the RECV parameter; • System error.
0	1	8185_H	Destination buffer (RECV) is too short.

NDR	ERROR	STATUS	Meaning
0	1	8186_H	ID parameter invalid. ID != 1, 2 to 16 (S7-300). ID != 1, 2 to 64.(S7-400)
0	1	8304_H	The connection is not established. The send job should only be attempted again after waiting for at least 100 ms.
0	1	$8F23_H$	Source area invalid, e.g.: Area does note exist in the DB.
0	1	$8F25_H$	Area error writing a parameter.
0	1	$8F29_H$	Alignment error writing a parameter
0	1	$8F30_H$	Parameter is in the writeprotected first current data block.
0	1	$8F31_H$	Parameter is in the writeprotected second current data block.
0	1	$8F32_H$	Parameter contains a DB number that is too high.
0	1	$8F33_H$	DB number error
0	1	$8F3A_H$	Destination area not loaded (DB).
0	1	$8F43_H$	Timeout writing a parameter to the I/O area.
0	1	$8F45_H$	Address of the parameter to be written is disabled in the access track.
0	1	$8F7F_H$	Internal error, for example illegal ANY reference.
0	1	8090_H	• No module with this module start address exists or the CPU is in STOP mode • The instruction being used does not match the system family being used (remember to use different instructions for S7300 and S7400).
0	1	8091_H	Module start address not at a doubleword boundary.
0	1	8092_H	In the ANY reference, a type other than BYTE is specified. (S7-400 only)
0	1	$80A0_H$	Negative acknowledgment reading from the module.
0	1	$80A4_H$	The communication bus connection between the CPU and CP is not established.
0	1	$80B0_H$	The module does not recognize the data record.
0	1	$80B1_H$	Possible causes: • The destination area is invalid. • The destination area is too short. • The destination area for the received data was adequately dimensioned. Remedy: Run another receive call with maximum receive buffer size. This applies regardless of the connection type (unicast / multicast / broadcast) and the device family (S7-300 / S7-400).
0	1	$80B2_H$	The communication bus connection between the CPU and CP is not established.
0	1	$80C0_H$	The data record cannot be read.
0	1	$80C1_H$	The specified data record is currently being processed.
0	1	$80C2_H$	There are too many jobs pending.
0	1	$80C3_H$	CPU resources (memory) occupied.
0	1	$80C4_H$	Communications error Occurs temporarily and a repetition in the user program will often remedy the problem
0	1	$80D2_H$	Module start address incorrect.

See also

AG_RECV / AG_LRECV / AG_SRECV (Ind. Ethernet) (Page 7724)

Instructions for access coordination with FETCH/WRITE

Instructions for access coordination with FETCH/WRITE

Overview

The following instructions are available for the FETCH/WRITE function to coordinate access:

Instruction	can be used with:		Meaning
S7-300	S7-400		
AG_LOCK	x	x	Locks external data access with FETCH/WRITE.
AG_UNLOCK	x	x	Releases external data access with FETCH/WRITE.

Caution when Configuring

If you use AG_LOCK and AG_UNLOCK, you must specify the following information for CPs in S7400 stations in the configuration:

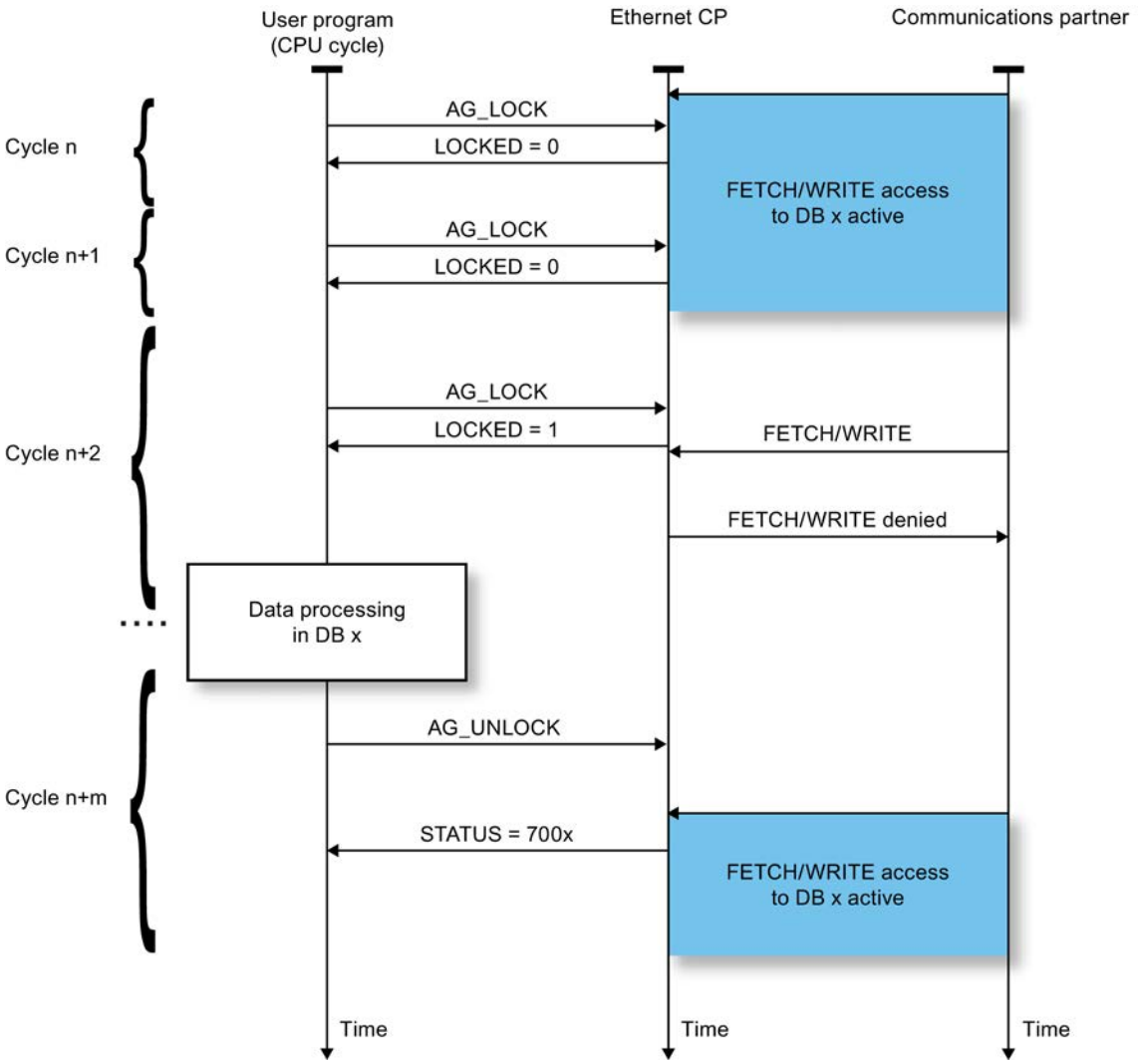
- Under "Properties > Addresses"
  The "Address setting for LOCK/UNLOCK" option must be selected if the selection is available.

How It works

With these instructions, you can coordinate access to system memory areas so that no inconsistent data is created and transferred. The control is from the user program in the S7 CPU that can, if necessary, disable an external FETCH/WRITE access using an AG_LOCK call. After a certain time or after the local write/read access is completed, an AG_UNLOCK job can be used to enable external access again.

Another advantage is that this access lock only applies to the FETCH/WRITE connection specified in the call. If more than one FETCH/WRITE connection is configured, these can, for example, be used for certain specific system areas and a selective access coordination can be implemented.

The following diagram illustrates the usual chronological sequence of memory access coordination controlled in the user program with AG_LOCK and AG_UNLOCK.



The lock job must first be monitored in the user program using the code in the return parameter LOCKED. As long as LOCKED=0 is indicated, it must be assumed that there is still an external FETCH/WRITE access active.

If LOCKED=1 is indicated, this shows that the lock is active; data can now be modified by the user program.

The status code is updated at each instruction call.

AG_LOCK

AG_LOCK

Description

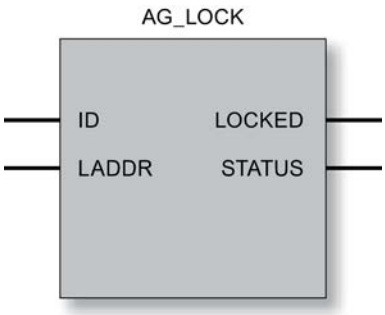
Using the AG_LOCK instruction, the data exchange using FETCH or WRITE on the connection selected with the parameter ID is disabled. The LOCKED output indicates whether or not the

lock was successful. If the lock was not successful, the job must be triggered again in a later CPU cycle.

The STATUS output indicates the status of the CP for this connection.

Call

Call interface in FBD representation



See also

Parameters for AG_LOCK (Page 7730)

STATUS parameter (Page 7731)

Parameters for AG_LOCK

Explanation of the formal parameters

The following table explains all the formal parameters for AG_LOCK:

Parameter	Declaration	Data type	Range of values	Description
ID	INPUT	INT	1, 2 to 16 for S7300 1, 2 to 64 for S7400	The connection number of the connection is specified in the parameter ID.
LADDR	INPUT	WORD		Module start address When you configure the CP with STEP 7, the module start address is displayed in the configuration table. Specify this address here.
LOCKED	OUTPUT	BOOL	0: not (yet) locked 1: locked	Shows the status of the access lock requested on the specified FETCH/WRITE connection.
STATUS	OUTPUT	WORD		Status code For the meaning, see STATUS parameter (Page 7731)

See also

AG_LOCK (Page 7729)

STATUS parameter

Condition codes

The following table shows the STATUS code that must be evaluated by the user program.

Table 9-330 AG_LOCK condition codes

STATUS	Meaning
7000_H	CP is not processing a job
7001_H	FETCH active
7002_H	WRITE active
8183_H	FETCH/WRITE not configured for this connection (S7-400 only)
8186_H	ID number not in permitted range (e.g. 1...64 for S7-400 Industrial Ethernet CPs)
$80A4_H$	The communication bus connection between the CPU and CP is not established (with newer CPU versions).
$80B0_H$	The module does not recognize the data record.
$80B1_H$	The specified length (in the LEN parameter) is incorrect.
$80B2_H$	The communication bus connection between the CPU and CP is not established.
$80C0_H$	The data record cannot be read.
$80C1_H$	The specified data record is currently being processed.
$80C2_H$	There are too many jobs pending.
$80C3_H$	CPU resources (memory) occupied.
$80C4_H$	Communications error Occurs temporarily and a repetition in the user program will often remedy the problem.
$80D2_H$	Module start address incorrect.

See also

AG_LOCK (Page 7729)

AG_UNLOCK

AG_UNLOCK

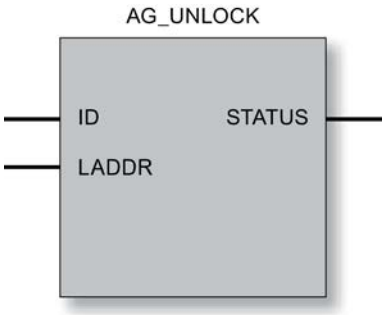
Description

With the aid of the AG_UNLOCK instruction, you enable external access to user memory areas of the S7-CPU. With FETCH or WRITE, access via the connection selected with the ID parameter is then possible.

The AG_UNLOCK follows an access lock with AG_LOCK.

Call

Call interface in FBD representation



How It works

To release the connection again, the instruction must clear the LOCK request bit again. The instruction also shows the current status using error messages.

See also

STATUS parameter (Page 7733)

Parameters for AG_UNLOCK (Page 7732)

Parameters for AG_UNLOCK

Explanation of the formal parameters

The following table explains all the formal parameters for AG_UNLOCK:

Parameter	Declaration	Data type	Range of values	Description
ID	INPUT	INT	1, 2 to 16 for S7300 1, 2 to 64 for S7400	The connection number of the connection is specified in the parameter ID. (See Configuration)
LADDR	INPUT	WORD		Module start address When you configure the CP with STEP 7, the module start address is displayed. Specify this address here.
STATUS	OUTPUT	WORD		Status code: For the meaning, see STATUS parameter (Page 7733)

See also

AG_UNLOCK (Page 7731)

STATUS parameter

Condition codes

The following table shows the STATUS code that must be evaluated by the user program.

Table 9-331 AG_UNLOCK condition codes

STATUS	Meaning
7000_H	CP is not processing a job
7001_H	FETCH active
7002_H	WRITE active
8183_H	FETCH/WRITE not configured for this connection (S7-400 only)
8186_H	ID number not in permitted range (e.g. 1...64 for S7-400 Industrial Ethernet CPs)
$80A4_H$	The communication bus connection between the CPU and CP is not established (with newer CPU versions).
$80B0_H$	The module does not recognize the data record.
$80B1_H$	The specified length (in the LEN parameter) is incorrect.
$80B2_H$	The communication bus connection between the CPU and CP is not established.
$80C0_H$	The data record cannot be read.
$80C1_H$	The specified data record is currently being processed.
$80C2_H$	There are too many jobs pending.
$80C3_H$	CPU resources (memory) occupied.
$80C4_H$	communications error Occurs temporarily and a repetition in the user program will often remedy the problem.
$80D2_H$	Module start address incorrect.

See also

AG_UNLOCK (Page 7731)

Instructions for connection diagnostics

AG_CNTEX

AG_CNTEX

Significance and how It works

With the AG_CNTEX instruction, it is possible run diagnostics on connections and to address devices using the ping command via the network. When necessary, you can initialize connection establishment again using AG_CNTEX.

Note

AG_CNTRL and AG_CNTEX

The AG_CNTEX program block provides expanded functionality compared with the AG_CNTRL program block.

All the functions of AG_CNTRL are included in AG_CNTEX and they can be used identically in the user program.

The following actions are possible by setting commands:

- Reading out connection information
  Based on status information, you can decide whether or not it would be useful to reset all or individual connections of the CP.
- Resetting configured connections
  You can reset individual connections or all connections of a CP.
- Aborting the active connection and establishing it again
- Reading out connection types configured on the CP (expanded functionality compared with AG_CNTRL)
- Send PING command (expansion compared with AG_CNTRL)
  You can check whether a specific node is reachable in the network.

The commands of the AG_CNTEX program block are permitted only for SEND/RECV connections based on the ISO / RFC / TCP / UDP protocols.

Note

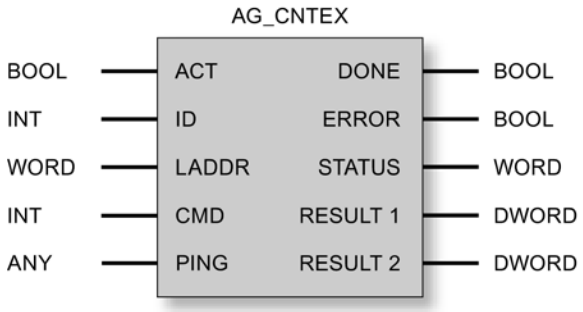
Availability in the block library

If the program block AG_CNTEX is not yet available in the SIMATIC_NET_CP block library, install the current SIMATIC NET block library; you will find this under the following entry ID on the Internet:

8797900 (http://support.automation.siemens.com/WW/view/de/8797900)

Call interface

Call interface in FBD representation



See also

How the PING function works (Page 7735)

Parameters for AG_CNTEX (Page 7736)

Parameters DONE, ERROR, STATUS (Page 7737)

Commands and job results - AG_CNTEX (Page 7739)

How the PING function works

Mode of operation / call sequence

With the CMD=8 PING command, you instruct the CP to send 4 successive PING requests over the network to the IP address specified in the job. The PING echo is expected by the CP within the period of time you set in the PING job field.

The CP registers the response times and enters these in the RESULT 1/2 parameters.

The RESULT 1/2 parameters can be queried with PING command CMD=9. As soon as the 4 PING requests have been replied to, or their set monitoring time has been exceeded, execution is confirmed in the DONE=1 parameter. The PING result can then be queried within a maximum of 30 seconds; afterwards, the RESULT entries become invalid.

Note

PING is only possible over a configured connection

The PING command is only possible if at least **one** connection for the SEND/RECEIVE interface (TCP, ISO-on-TCP, ISO-Transport, UDP) is configured.

Several PING requests at the same time

You can send up to 4 PING requests at the same time to different IP addresses. To do this, you must use the same instance DB for the ping requests. Further PING requests are possible only after completion of at least one of the current PING requests.

If too many PING requests are sent at the same time, an error message to this effect is output (STATUS parameter = $828A_H$).

When are PING requests completed?

PING requests count as being completed when one of the following conditions is met:

- The PING result was read out:
- The PING result was not read out but 30 seconds have elapsed since the Ping result was available.

See also

AG_CNTEX (Page 7734)

Commands and job results - AG_CNTEX (Page 7739)

Parameters for AG_CNTEX

Explanation of the formal parameters

The following table explains all the formal parameters for the AG_CNTEX function:

Parameter	Declaration	Data type	Range of values	Description
ACT	INPUT	BOOL	0, 1	The FB must be called with ACT=1. If it is called with ACT=0, there is no function call and the block is exited again immediately.
ID	INPUT	INT	• 1, 2, .., n, or • 0	The connection number of the connection is specified in the parameter ID. The connection number can be found in the configuration. n is the maximum number of connections and is dependent on the product (S7-300 or S7-400). For a call that addresses all connections, 0 must be specified as the ID. This affects: • the functions CN_STATUS_ALL (CMD 3) and CN_RESET_ALL (CMD 4) • PING command with CMD 8 or CMD 9
LADDR	INPUT	WORD		Module start address When you configure the CP with STEP 7, the module start address is displayed. Specify this address here.
CMD	INPUT	INT		Command to FB AG_CNTEX
PING	INPUT	ANY		References a block of data (for example DB) that contains the data structure for the PING command. The block of data contains the IP address and optional information about the time monitoring and the number of bytes to be transferred in the PING request. Data structure, see below

Parameter	Declaration	Data type	Range of values	Description
DONE	OUTPUT	BOOL	0: Job still being processed or not yet triggered 1: Job done	This parameter indicates whether or not the job was completed without errors. For the meaning in conjunction with the parameters ERROR and STATUS, refer to Parameters DONE, ERROR, STATUS (Page 7737) Note: If DONE=1, RESULT can be evaluated
ERROR	OUTPUT	BOOL	0: No error 1: Error	Error code For the meaning in conjunction with the parameters DONE and STATUS, refer to Parameters DONE, ERROR, STATUS (Page 7737)
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the parameters DONE and ERROR, refer to Parameters DONE, ERROR, STATUS (Page 7737)
RESULT1	OUTPUT	DWORD		Information returned to AG_CNTEX according to the command.
RESULT2	OUTPUT	DWORD		Part 2 of the information returned to AG_CNTEX according to the command.

PING block of data

The PING parameter references a block of data with the following data structure:

Parameter	Data type	Range of values	Note
IP address	ARRAY [1..4] of Byte		
TIMEOUT	INT	1..60000 ms	Can be specified as an option; default value = 1000 ms
Size	INT	1..1000 bytes	Can be specified as an option; default value = 32 bytes

See also

AG_CNTEX (Page 7734)

Parameters DONE, ERROR, STATUS

Condition codes

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

The command results in the RESULT1/2 parameters must also be evaluated according to Commands and job results - AG_CNTEX (Page 7739).

Table 9-332 AG_CNTEX codes

DONE	ERROR	STATUS	Meaning
1	0	0000_H	A job (CMD) was transferred successfully to the CP (for example RESET) or a status was read successfully from the CP. The RESULT1/2 parameters can be evaluated.
0	0	0000_H	There has been no block call yet or the program block was called with ACT=0.
0	0	8181_H	Job active The block call must be repeated with the same parameters until DONE or ERROR is signaled.
0	1	8183_H	No configuration or the service has not yet started on the Ethernet CP.
0	1	8184_H	System error or wrong parameter type. The cause can be: • Data type of the ANY pointer not correct for the PING parameter. • The ANY pointer references an odd bit address.
0	1	8186_H	The ID parameter is invalid. The permitted ID depends on the selected command.
0	1	8187_H	The CMD parameter is invalid.
0	1	8090_H	Possible meanings: • No module with this module start address exists; • The program block being used does not match the system family being used (remember to use different program blocks for S7300 and S7400); • The function is not supported by this module.
0	1	8091_H	The module start address is not at a doubleword boundary.
0	1	8092_H	The module start address is incorrect.
0	1	$80B0_H$	The module does not recognize the data record.
0	1	$80B2_H$	The communication bus connection between the CPU and CP is not established. The corresponding CPU in the H system is in STOP mode.
0	1	$80C0_H$	The data record cannot be read.
0	1	$80C1_H$	The specified data record is currently being processed.
0	1	$80C2_H$	There are too many jobs pending.
0	1	$80C3_H$	CPU resources (memory) occupied.
0	1	$80C4_H$	Communications error The error occurs temporarily; it is usually best to repeat the job in the user program.
0	1	8286_H	The value for the "Timeout" in the PING data block is outside the valid range of values.
0	1	8287_H	The IP address specified in the PING DB is reserved and therefore not permitted.
0	1	8288_H	The display occurs only with the PING result request command. Possible meanings: • The IP address was not reached by the PING command (for example because the CP was changed to STOP mode after the PING request command was sent); • The PING result has already been read out; • The PING result was not read out within the maximum time of 30 seconds.
0	1	8289_H	The data volume for the PING request has exceeded the permitted range (maximum 1000 bytes; see data structure for the PING command)
0	1	$828A_H$	There are already 4 ping requests being processed. New requests are only possible again after processing the existing requests.
0	1	$828B_H$	There is already a PING request being processed for the specified IP address. Use the PING result request to complete the current processing.

See also

AG_CNTEX (Page 7734)

Commands and job results - AG_CNTEX

Commands and evaluating the job results

The following table shows you the possible commands and the results that can be evaluated in the RESULT1/2 parameters.

Note

Command evaluation with older CP types or firmware versions

The commands described below are supported by the current CP types or firmware versions. You should also check the more detailed information under the following entry ID:

http://support.automation.siemens.com/WW/view/en/33414377 (http://support.automation.siemens.com/ww/view/en/33414377)

Table 9-333 Commands to FC AG_CNTRL

CMD	Meaning		
0	NOP – no operation The block executes without a job being sent to the CP.		
	RESULT (for CMD = 0)		**Meaning**
	Parameter	**Hex value/range**	
	RESULT1	$0000\ 0001_H$	Executed without error
	RESULT2	$0000\ 0000_H$	Default

CMD	Meaning			
1	CN_STATUS – connection status This command returns the status of the connection selected with the ID. The CP is selected in the LADDR parameter. If bit 15 (reset ID) is set, this is automatically reset (this action corresponds to the CN_CLEAR_RESET job - see CMD = 5).			
	RESULT (for CMD = 1)			**Value / meaning**
	Parameter	**Hex value/range**	**Bit**	
	RESULT1	$0000\ 000*_H$	Bits 0-3: Codes for the send direction (excluded values: 0x2)	
			Bit 0	Connection Type • 0: No send and receive connection • 1: Connection reserved for send and receive jobs

CMD	Meaning			
			Bit 1	Status of current job • 0: No send job being executed • 1: Send job being executed
			Bits 2+3	Previous job: • 00: No information available on previous send job • 01: previous send job completed successfully • 10: previous send job not completed successfully

CMD	Meaning			
1	CN_STATUS – connection status **(continued for CMD=1)** This command returns the status of the connection selected with the ID. The CP is selected in the LADDR parameter. If bit 15 (reset ID) is set, this is automatically reset (this action corresponds to the CN_CLEAR_RESET job - see CMD = 5).			
	RESULT (for CMD = 1)			**Value / meaning**
	Parameter	**Hex value/range**	**Bit**	
	RESULT1	0000 00*0_H	Bits 4−7: Codes for the receive direction (excluded values: 0x2)	
			Bit 4	Connection Type • 0: No send and receive connection • 1: Connection reserved for send and receive jobs
			Bit 5	Status of current job • 0: No receive job being executed • 1: Receive job being executed
			Bits 6+7	Previous job: • 00: No information available on previous receive job • 01: previous receive job completed successfully • 10: previous receive job not completed successfully

CMD	Meaning			
1	CN_STATUS – connection status **(continued for CMD=1)** This command returns the status of the connection selected with the ID. The CP is selected in the LADDR parameter. If bit 15 (reset ID) is set, this is automatically reset (this action corresponds to the CN_CLEAR_RESET job - see CMD = 5).			
	RESULT (for CMD = 1)			**Value / meaning**
	Parameter	**Hex value/range**	**Bit**	
	RESULT1	0000 0*00_H	Bits 8-11: Codes for FETCH/WRITE (excluded values: 0x3, 0x7, 0x8, 0xB, 0xF)	
			Bit 8	Connection type: • 0: No FETCH connection • 1: Connection reserved for FETCH jobs

CMD	Meaning			
			Bit 9	Connection type: • 0: No WRITE connection • 1: Connection reserved for WRITE jobs
			Bit 10	Job status (FETCH/WRITE): • 0: Job status OK • 1: Job status NOT OK This ID is set in the following situations: – The job was acknowledged negatively by the CPU – The job could not be forwarded to the CPU because the connection was in the "LOCKED" status. – The job was rejected because the FETCH/WRITE header did not have the correct structure.
			Bit 11	Status of FETCH/WRITE job • 0: No job active • 1: Job from LAN active

CMD	Meaning			
1	CN_STATUS – connection status **(continued for CMD=1)** This command returns the status of the connection selected with the ID. The CP is selected in the LADDR parameter. If bit 15 (reset ID) is set, this is automatically reset (this action corresponds to the CN_CLEAR_RESET job - see CMD = 5).			
	RESULT (for CMD = 1)			**Value / meaning**
	Parameter	**Hex value/range**	**Bit**	
	RESULT1	0000 *000_H	Bits 12-15: General CP information (excluded values: 0x3, 0xB)	
			Bit 12 + 13	Information on connection status: (only available for SEND/RECV connections based on the ISO/RFC/TCP protocols, with UDP, the corresponding internal information is output) • 00: Connection is terminated • 01: Connection establishment active • 10: Connection termination active • 11: Connection is established
			Bit 14	CP information: • 0: CP in STOP • 1: CP in RUN
			Bit 15	Reset ID • 0: AG_CNTEX has not yet reset a connection or the reset ID was cleared. • 1: AG_CNTEX caused a connection reset.
	RESULT1	**** 0000_H		Bits 16-31: Reserved 0 – reserved for later expansions
	RESULT2	0000 0000_H		- reserved for later expansions

CMD	Meaning		
2	CN_RESET – connection reset This command resets the connection selected with ID. The CP is selected in the LADDR parameter. Resetting the connection means that a connection is aborted and established again (active or passive depending on the configuration). Data that has been received but not yet entered in the user program when the connection aborts is deleted. An entry is also generated in the diagnostics buffer in which the job result can be found.		
	RESULT (for CMD = 2)		**Meaning**
	Parameter	**Hex value/range**	
	RESULT1	$0000\ 0001_H$	The reset job was transferred to the CP successfully. The connection abort and subsequent connection establishment were triggered.
		$0000\ 0002_H$	The reset job could not be transferred to the CP because the service has not started on the CP (for example, CP in STOP).
	RESULT2	$0000\ 0000_H$	Default

CMD	Meaning		
3	CN_STATUS_ALL – all connections status This command returns the connection status of all connections (established/terminated) in the RESULT1/2 parameters (at total of 8 bytes of group information). The ID parameter must be set to "0" (checked for 0). The CP is selected in the LADDR parameter. When necessary, you can obtain detailed information about a terminated or unconfigured connection using a further connection status call with CMD=1.		
	RESULT (for CMD = 3)		**Meaning**
	Parameter	**Hex value/range**	
	RESULT1	$**** ****_H$ 32 bits with the following validity: • For S7-400: Bits 0-31 for connections 1 - 32 • For S7-300: Bits 0-15 for connections 1 - 16	For the relevant connection: • 0 – connection terminated / not configured • 1 – connection established
	RESULT2	$**** ****_H$ 32 bits with the following validity: • For S7-400: Bits 0-31 for connections 33 - 64	For the relevant connection: • 0 – connection terminated / not configured • 1 – connection established

CMD	Meaning		
4	CN_RESET_ALL – all connections reset This command resets all connections. The ID parameter must be set to "0" (checked for 0). The CP is selected in the LADDR parameter. Resetting the connections means that connections are aborted and established again (active or passive depending on the configuration). Data that has been received but not yet entered in the user program when the connection aborts is deleted. An entry is also generated in the diagnostics buffer in which the job result can be found.		
	RESULT (for CMD = 4)		Meaning
	Parameter	Hex value/range	
	RESULT1	$0000\ 0001_H$	The reset job was transferred to the CP successfully. The connection abort and subsequent connection establishment of all connections were triggered.
	RESULT1	$0000\ 0002_H$	The reset job could not be transferred to the CP because the service has not started on the CP (for example, CP in STOP).
	RESULT2	$0000\ 0000_H$	Default

CMD	Meaning		
5	CN_CLEAR_RESET - Clear the reset ID This command resets the reset ID (bit 15 in RESULT1) for the connection selected with ID. The CP is selected in the LADDR parameter. This job executes automatically when the connection status is read (CMD=1); the separate job described here is therefore only required in special situations.		
	RESULT (for CMD = 5)		Meaning
	Parameter	Hex value/range	
	RESULT1	$0000\ 0001_H$	The clear job was transferred to the CP successfully.
	RESULT1	$0000\ 0002_H$	The Clear job could not be transferred to the CP because the service has not started on the CP (for example, CP in STOP).
	RESULT2	$0000\ 0000_H$	Default

CMD	Meaning		
6	CN_DISCON − connection disconnect This command resets the connection selected with ID and LADDR. Resetting the connection is achieved by aborting the connection. Any data in the stack is lost without any message being displayed. The connection is not established again automatically afterwards. The connection can be established again with the CN_STARTCON control job. A diagnostics buffer entry is created in which you will find the job result.		
	RESULT (for CMD = 6)		Meaning
	Parameter	Hex value/range	
	RESULT1	$0000\ 0001_H$	The job was transferred to the CP successfully. The connection abort was initiated.

CMD	Meaning		
	RESULT1	0000 0002_H	The job could not be transferred to the CP because the service has not started on the CP (for example, CP in STOP).
	RESULT2	0000 0000_H	Default

CMD	Meaning		
7	CN_STARTCON − start connection This command establishes a connection selected with ID and LADDR and aborted earlier with the control job CN_DISCON. A diagnostics buffer entry is created in which you will find the job result.		
	RESULT (for CMD = 7)		Meaning
	Parameter	Hex value/range	
	RESULT1	0000 0001_H	The connection establishment job was transferred to the CP successfully. The connection establishment was initiated.
	RESULT1	0000 0002_H	The connection establishment job could not be transferred to the CP because the service has not started on the CP (for example, CP in STOP).
	RESULT2	0000 0000_H	Default

CMD	Meaning		
8	PING_REQUEST - Send a PING request This command sends a PING command to the CP. The CP then initiates 4 PING echo requests to the specified IP address.		
	RESULT (for CMD = 8)		Meaning
	Parameter	Hex value/range	
	RESULT1	0000 0001_H	The ping request was sent to the CP successfully.
	RESULT2	0000 0002_H	The ping request could not be sent to the CP because the corresponding service was not available on the CP. A possible cause might, for example, be: CP in STOP mode

CMD	Meaning		
9	PING_RESULT - Query ping result This command sends a PING result request to the CP. The CP transfers the results of the 4 executed PING echo requests in the RESULT parameter. The call is successful when the 4 ping echo requests have been completed on the part of the CP.		
	RESULT (for CMD = 9)		Meaning
	Parameter	Hex value/range	

CMD	Meaning		
	RESULT1	**** ****$_H$	1. Word: Reply time in ms for the 1st PING echo request. 2. Word: Reply time in ms for the 2nd PING echo request. Example: 0005 FFFF$_H$ Echo 1 -> received after 5 ms Echo 2 -> no echo in the set monitoring time
	RESULT2	**** ****$_H$	1. Word: Reply time in ms for the 3rd PING echo request. 2. Word: Reply time in ms for the 4th PING echo request. Example: 0002 3456$_H$ Echo 3 -> received after 2 ms Echo 4 -> received after 13398 ms
	Range of values for data words in RESULT1 / RESULT 2:		
		0000$_H$	not used
		0001$_H$ ... EA60$_H$	Reply time in ms 0001$_H$ = 1 ms EA60$_H$ = 60000 ms
		EA61$_H$ ... FFFE$_H$	not used
		FFFF$_H$	Timeout: No echo within the specified monitoring time.

CMD	Meaning		
10	CONN_TYPE - Connection type This command requests the CP to specify the current connection type for the specified connection ID.		
	RESULT (for CMD = 10)		**Meaning**
	Parameter	**Hex value/range**	
	RESULT1	0000 000*$_H$	The request returns the following values for the possible connection types: 0: no connection set up 1: UDP connection 2: SMTP connection 3: TCP connection 4: Free UDP connection 5: FTP connection 6: ISO transport connection 7: ISOonTCP connection
	RESULT2	0000 0000$_H$	Default

See also

You will find further information in the FAQ under the following entry ID: http://support.automation.siemens.com/WW/view/en/33414377 (http://support.automation.siemens.com/ww/view/en/33414377)

See also

AG_CNTEX (Page 7734)

How the PING function works (Page 7735)

AG_CNTRL for connection diagnostics

AG_CNTRL

Description

With the AG_CNTRL instruction, you can diagnose connections. When necessary, you can initialize connection establishment again using the instruction.

Note

AG_CNTRL and AG_CNTEX

The AG_CNTEX instruction provides expanded functionality compared with the AG_CNTRL instruction.

All the functions of AG_CNTRL are included in AG_CNTEX and they can be used identically in the user program.

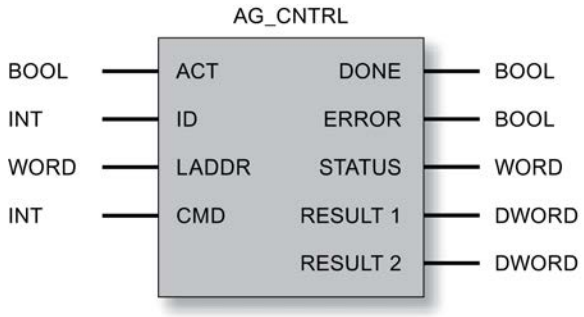
The following actions are possible by setting commands:

- Reading out connection information
  Based on status information, you can decide whether or not it would be useful to reset all or individual connections of the CP.
- Resetting configured connections
  You can reset individual connections or all connections of a CP.
- Aborting the active connection and establishing it again

The commands of the AG_CNTRL instruction are permitted only for SEND/RECV connections based on the ISO / RFC / TCP / UDP protocols.

Call interface

Call interface in FBD representation



Further information

FAQ under entry ID 33414377 (http://support.automation.siemens.com/WW/view/de/33414377)

See also

Parameters for AG_CNTRL (Page 7747)

Parameters DONE, ERROR, STATUS (Page 7748)

Commands and job results for AG_CNTRL (Page 7749)

Parameters for AG_CNTRL

Explanation of the formal parameters

The following table explains all the formal parameters for AG_CNTRL:

Parameter	Declaration	Data type	Range of values	Description
ACT	INPUT	BOOL	0, 1	AG_UNLOCK must be called with ACT=1. If it is called with ACT=0, there is no function call and the instruction is exited again immediately.
ID	INPUT	INT	• 1, 2, .., n, or • 0	The connection number of the connection is specified in the parameter ID. The connection number can be found in the configuration. n is the maximum number of connections and is dependent on the product (S7-300 or S7-400). If the call addresses all connections (_ALL function with CMD 3 or 4), 0 must be specified as the ID.
LADDR	INPUT	WORD		Module start address When you configure the CP with STEP 7, the module start address is displayed in the configuration table. Specify this address here.

Parameter	Declaration	Data type	Range of values	Description
CMD	INPUT	INT		Command to AG_CNTRL.
DONE	OUTPUT	BOOL	0: Job still being processed or not yet triggered 1: Job done	This parameter indicates whether or not the job was completed without errors. For the meaning in conjunction with the parameters ERROR and STATUS, refer to Parameters DONE, ERROR, STATUS (Page 7748) Note: If DONE=1, RESULT can be evaluated
ERROR	OUTPUT	BOOL	0: No error 1: Error	Error code For the meaning in conjunction with the parameters DONE and STATUS, refer to Parameters DONE, ERROR, STATUS (Page 7748)
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the parameters DONE and ERROR, refer to Parameters DONE, ERROR, STATUS (Page 7748)
RESULT1	OUTPUT	DWORD		Information returned according to the command sent to AG_CNTRL.
RESULT2	OUTPUT	DWORD		Only to be evaluated for S7-400: Part 2 of the information returned according to the command sent to AG_CNTRL.

See also

AG_CNTRL (Page 7746)

Parameters DONE, ERROR, STATUS

Condition codes

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

The command results in the RESULT1/2 parameters must also be evaluated according to "Commands and job results for AG_CNTRL (Page 7749)".

Table 9-334 AG_CNTRL codes

DONE	ERROR	STATUS	Meaning
1	0	0000_H	A job (CMD) was transferred successfully to the CP (for example RESET) or a status was read successfully from the CP. The RESULT1/2 parameters can be evaluated.
0	0	0000_H	There has been no instruction call yet or the instruction was called with ACT=0.
0	0	8181_H	Job active The instruction call must be repeated with the same parameters until DONE or ERROR is signaled.
0	1	8183_H	No configuration or the service has not yet started on the Ethernet CP.

DONE	ERROR	STATUS	Meaning
0	1	8186_H	The ID parameter is invalid. The permitted ID depends on the selected command; see CMD parameter in "Commands and job results for AG_CNTRL (Page 7749)"
0	1	8187_H	The CMD parameter is invalid.
0	1	8188_H	Sequence error in the ACT control (Note: this code does not occur in the product version of the CP / firmware).
0	1	8189_H	The CP version / firmware used does not support AG_CNTRL. The code is set when you call a CP 3431-EX20 with firmware as of V1.3.9; with other CP types, the code 80B0H is set instead. Note: AG_CNTRL in version V1.0 is supported by the CPs as of CP 343-1EX21/GX21; this code does not occur with these modules.
0	1	8090_H	• No module with this module start address exists. or • The instruction being used does not match the system family being used (remember to use different instructions for S7300 and S7400). or • The function is not supported by this module.
0	1	8091_H	The module start address is not at a doubleword boundary.
0	1	$80B0_H$	The module does not recognize the data record.
0	1	$80B2_H$	The communication bus connection between the CPU and CP is not established. The corresponding CPU in the H system is in STOP mode.
0	1	$80C0_H$	The data record cannot be read.
0	1	$80C1_H$	The specified data record is currently being processed.
0	1	$80C2_H$	There are too many jobs pending.
0	1	$80C3_H$	CPU resources (memory) occupied.
0	1	$80C4_H$	Communications error The error occurs temporarily; it is usually best to repeat the job in the user program.
0	1	$80D2_H$	The module start address is incorrect.

See also

AG_CNTRL (Page 7746)

Commands and job results for AG_CNTRL

Commands and evaluating the job results

The following table shows you the possible commands and the results that can be evaluated in the RESULT1/2 parameters.

Table 9-335 Commands to AG_CNTRL

CMD	Meaning	
0	NOP – no operation AG_CNTRL executes without a job being sent to the CP.	
	RESULT (for CMD = 0)	Meaning

CMD	Meaning		
	Parameter	Hex value/range	
	RESULT1	$0000\ 0001_H$	Executed without error
	RESULT2	$0000\ 0000_H$	Default

CMD	Meaning			
1	CN_STATUS – connection status This command returns the status of the connection selected with the ID. The CP is selected in the LADDR parameter. If bit 15 (reset ID) is set, this is automatically reset (this action corresponds to the CN_CLEAR_RESET job - see CMD = 5).			
	RESULT (for CMD = 1)			Value / meaning
	Parameter	Hex value/range	Bit	
	RESULT1	$0000\ 000*_H$	Bits 0-3: Codes for the send direction (excluded values: 0x2)	
			Bit 0	Connection Type • 0: No send and receive connection • 1: Connection reserved for send and receive jobs
			Bit 1	Status of current job • 0: No send job being executed • 1: Send job being executed
			Bits 2+3	Previous job: • 00: No information available on previous send job • 01: previous send job completed successfully • 10: previous send job not completed successfully

CMD	Meaning			
1	CN_STATUS – connection status **(CMD=1 continued)** This command returns the status of the connection selected with the ID. The CP is selected in the LADDR parameter. If bit 15 (reset ID) is set, this is automatically reset (this action corresponds to the CN_CLEAR_RESET job - see CMD = 5).			
	RESULT (for CMD = 1)			Value / meaning
	Parameter	Hex value/range	Bit	
	RESULT1	$0000\ 00*0_H$	Bits 4−7: Codes for the receive direction (excluded values: 0x2)	
			Bit 4	Connection Type • 0: No send and receive connection • 1: Connection reserved for send and receive jobs

CMD	Meaning			
			Bit 5	Status of current job • 0: No receive job being executed • 1: Receive job being executed
			Bits 6+7	Previous job: • 00: No information available on previous receive job • 01: previous receive job completed successfully • 10: previous receive job not completed successfully

CMD	Meaning			
1	CN_STATUS – connection status **(CMD=1 continued)** This command returns the status of the connection selected with the ID. The CP is selected in the LADDR parameter. If bit 15 (reset ID) is set, this is automatically reset (this action corresponds to the CN_CLEAR_RESET job - see CMD = 5).			
	RESULT (for CMD = 1)			**Value / meaning**
	Parameter	**Hex value/range**	**Bit**	
	RESULT1	0000 0*00$_H$	Bits 8-11: Codes for FETCH/WRITE (excluded values: 0x3, 0x7, 0x8, 0xB, 0xF)	
			Bit 8	Connection type: • 0: No FETCH connection • 1: Connection reserved for FETCH jobs
			Bit 9	Connection type: • 0: No WRITE connection • 1: Connection reserved for WRITE jobs
			Bit 10	Job status (FETCH/WRITE): • 0: Job status OK • 1: Job status NOT OK This ID is set in the following situations: – The job was acknowledged negatively by the CPU – The job could not be forwarded to the CPU because the connection was in the "LOCKED" status. – The job was rejected because the FETCH/WRITE header did not have the correct structure.
			Bit 11	Status of FETCH/WRITE job • 0: No job active • 1: Job from LAN active

CMD	Meaning	
1	CN_STATUS – connection status **(CMD=1 continued)** This command returns the status of the connection selected with the ID. The CP is selected in the LADDR parameter. If bit 15 (reset ID) is set, this is automatically reset (this action corresponds to the CN_CLEAR_RESET job - see CMD = 5).	
	RESULT (for CMD = 1)	**Value / meaning**

CMD	Meaning			
	Parameter	Hex value/range	Bit	
	RESULT1	0000 *000_H	Bits 12-15: General CP information (excluded values: 0x3, 0xB)	
			Bit 12 + 13	Information on connection status: (only available for SEND/RECV connections based on the ISO/RFC/TCP protocols, with UDP, the corresponding internal information is output) • 00: Connection is terminated • 01: Connection establishment active • 10: Connection termination active • 11: Connection is established
			Bit 14	CP information: • 0: CP in STOP • 1: CP in RUN
			Bit 15	Reset ID • 0: AG_CNTRL has not yet reset a connection or the reset ID was cleared. • 1: The control instruction caused a connection reset.
	RESULT1	**** 0000_H		Bits 16-31: Reserved 0 – reserved for later expansions
	RESULT2	0000 0000_H		- reserved for later expansions

CMD	Meaning		
2	CN_RESET – connection reset This command resets the connection selected with ID. The CP is selected in the LADDR parameter. Resetting the connection means that a connection is aborted and established again (active or passive depending on the configuration). Data that has been received but not yet entered in the user program when the connection aborts is deleted. An entry is also generated in the diagnostics buffer in which the job result can be found.		
	RESULT (for CMD = 2)		Meaning
	Parameter	Hex value/range	
	RESULT1	0000 0001_H	The reset job was transferred to the CP successfully. The connection abort and subsequent connection establishment were triggered.
		0000 0002_H	The reset job could not be transferred to the CP because the service has not started on the CP (for example, CP in STOP).
	RESULT2	0000 0000_H	Default

CMD	Meaning		
3	CN_STATUS_ALL – all connections status This command returns the connection status of all connections (established/terminated) in the RESULT1/2 parameters (at total of 8 bytes of group information). The ID parameter must be set to "0" (checked for 0). The CP is selected in the LADDR parameter. When necessary, you can obtain detailed information about a terminated or unconfigured connection using a further connection status call with CMD=1.		
	RESULT (for CMD = 3)		Meaning
	Parameter	Hex value/range	
	RESULT1	**** ****$_H$	32 bits: Connection 1 - 32 • 0 – connection terminated / not configured • 1 – connection established
	RESULT2	**** ****$_H$	32 bits: Connection 33 - 64 • 0 – connection terminated / not configured • 1 – connection established

CMD	Meaning		
4	CN_RESET_ALL – all connections reset: This command resets all connections. The ID parameter must be set to "0" (checked for 0). The CP is selected in the LADDR parameter. Resetting the connections means that connections are aborted and established again (active or passive depending on the configuration). Data that has been received but not yet entered in the user program when the connection aborts is deleted. An entry is also generated in the diagnostics buffer in which the job result can be found.		
	RESULT (for CMD = 4)		Meaning
	Parameter	Hex value/range	
	RESULT1	0000 0001$_H$	The reset job was transferred to the CP successfully. The connection abort and subsequent connection establishment of all connections were triggered.
	RESULT1	0000 0002$_H$	The reset job could not be transferred to the CP because the service has not started on the CP (for example, CP in STOP).
	RESULT2	0000 0000$_H$	Default

CMD	Meaning		
5	CN_CLEAR_RESET – Clear the reset ID This command resets the reset ID (bit 15 in RESULT1) for the connection selected with ID. The CP is selected in the LADDR parameter. This job executes automatically when the connection status is read (CMD=1); the separate job described here is therefore only required in special situations.		
	RESULT (for CMD = 5)		Meaning
	Parameter	Hex value/range	
	RESULT1	0000 0001$_H$	The clear job was transferred to the CP successfully.

CMD	Meaning		
	RESULT1	0000 0002_H	The Clear job could not be transferred to the CP because the service has not started on the CP (for example, CP in STOP).
	RESULT2	0000 0000_H	Default

CMD	Meaning		
6	CN_DISCON − connection disconnect This command resets the connection selected with ID and LADDR. Resetting the connection is achieved by aborting the connection. Any data in the stack is lost without any message being displayed. The connection is not established again automatically afterwards. The connection can be established again with the CN_STARTCON control job. A diagnostics buffer entry is created in which you will find the job result.		
	RESULT (for CMD = 6)		**Meaning**
	Parameter	**Hex value/range**	
	RESULT1	0000 0001_H	The job was transferred to the CP successfully. The connection abort was initiated.
	RESULT1	0000 0002_H	The job could not be transferred to the CP because the service has not started on the CP (for example, CP in STOP).
	RESULT2	0000 0000_H	Default

CMD	Meaning		
7	CN_STARTCON − start connection This command establishes a connection selected with ID and LADDR and aborted earlier with the control job CN_DISCON. A diagnostics buffer entry is created in which you will find the job result.		
	RESULT (for CMD = 6)		**Meaning**
	Parameter	**Hex value/range**	
	RESULT1	0000 0001_H	The connection establishment job was transferred to the CP successfully. The connection establishment was initiated.
	RESULT1	0000 0002_H	The connection establishment job could not be transferred to the CP because the service has not started on the CP (for example, CP in STOP).
	RESULT2	0000 0000_H	Default

Further information

You will find further information in the FAQ under the following entry ID: 33414377 (http://support.automation.siemens.com/WW/view/en/33414377)

See also

AG_CNTRL (Page 7746)

Instructions for PROFINET IO

Instructions for PROFINET IO

Overview

The instructions listed below are available for transferring data cyclically on the PROFINET IO interface. The significance of the instructions differs depending on how you use the CP (as a PROFINET IO controller or PROFINET IO device) in an S7 station.

Instruction	can be used with:		Meaning
S7-300	S7-400		
PNIO_SEND	x	-	Depending on the mode of the CP: • On a PROFINET IO controller Sending output data to the PROFINET IO devices. • On a PROFINET IO device Forwarding process input data to the PROFINET IO controller.
PNIO_RECV	x	-	Depending on the mode of the CP: • On a PROFINET IO controller Receiving process input data from the PROFINET IO devices. • On a PROFINET IO device Receiving process output data from the PROFINET IO controller.

For CPs operating as PROFINET IO controller and IO device at the same time, the instructions are available as of version 2.0.

The instructions listed below are available for transferring data (data records, interrupt information) acyclically on the PROFINET IO interface. The two instructions can only be used in PROFINET IO controller mode.

Instruction	can be used with:		Meaning
S7-300	S7-400		
PNIO_RW_REC	x	-	• Read data record (from a PROFINET IO device) • Write data record (to a PROFINET IO device)
PNIO_ALARM	x	-	Receive alarm information from the PROFINET IO devices

Instructions for cyclic transmission of user data

PNIO_SEND

PNIO_SEND

Description

The PNIO_SEND instruction is used to transfer data in the PROFINET IO controller or PROFINET IO device modes of the CP.

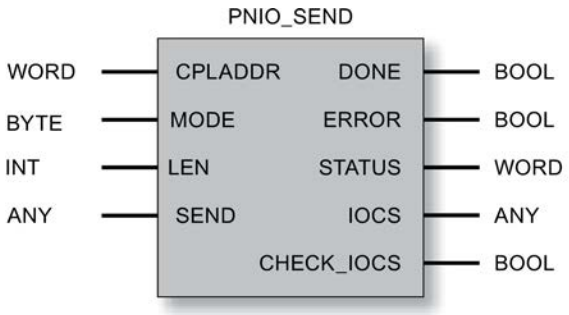
- Operating as PROFINET IO controller
  The instruction transfers the process data (outputs) of a specified output area to the CP to be forwarded to PROFINET IO devices and as status display returns the IO consumer status (IOCS) of the outputs of the PROFINET IO devices.
- Operation as PROFINET IO device
  The instruction reads the preprocessed inputs of the CPU on the PROFINET IO device and transfers them to the PROFINET IO controller (configured I-addresses). In addition to this, the instruction also returns the IO Consumer Status (IOCS) of the PROFINET IO controller as a status code.

The preprocessed process data is available in a DB or bit memory area.

For CPs operating as PROFINET IO controller and IO device at the same time, the instruction is available as of version 2.0. With the additional MODE parameter, you set the mode for which the instruction will be called.

Call interface (version 2.0)

Call interface in FBD representation



See also

General characteristics of the instructions for PROFINET IO (Page 7768)

Data consistency (Page 7769)

Parameters for PNIO_SEND (Page 7757)

Parameters DONE, ERROR, STATUS (IE) (Page 7761)

Parameters for PNIO_SEND

Explanation of the formal parameters

The following table explains all the formal parameters for the PNIO_SEND function:

Parameter	Declaration	Data type	Range of values	Description
CPLADDR	INPUT	WORD	-	Module start address
MODE (parameters version 2.0 or later)	INPUT	BYTE	Values for X: • $0Y_H$: Status bits are transferred in IOCS. • $8Y_H$: Restriction to group message in CHECK_IOCS; no status bits in IOCS. Values for Y: • $X0_H$: – IO controller mode – IO device mode (without parallel operation) There is compatibility with the instruction in version 1.0. • $X1_H$: IO device mode (with parallel operation)	Specified in the form XY (hexadecimal): • X sets the transfer of status information. • Y specifies the mode of the CP as IO controller or IO device. Notes on compatibility;: • The version 1.0 instruction can continue to be used as long as the CP is not being operated as an IO controller and IO device at the same time. • When MODE=0, the instruction as of version 2.0 behaves like the instruction version 1.0.

Parameter	Declaration	Data type	Range of values	Description
SEND	IN_OUT	ANY (as VARTYPE only BYTE is permitted)	The address of the data area points to one of the alternatives: • Memory bit area • Data block area	Specifies the address and length **IO controller mode:** The length should match the total length of the distributed IO configured, whereby address gaps are also transmitted. The length can also be shorter than the total length of the distributed IO, for example when the instruction is called more than once in one OB. It must, however, have the total length in at least one call. **IO device mode:** The data structure results from the order of the slots of the input modules configured for this PROFINET IO device on the PROFINET IO controller line and their length without address gaps. (Please note the more extensive explanations or examples for your CP in the devicespecific Part B of this manual) Notes: • The instruction begins to transfer the data at address 0 regardless of how you configured the addresses (regardless of the lowest configured address). • Specifying an I/O area is not permitted since you must first check the IOCS for GOOD before data can be accepted in the I/O.

Parameter	Declaration	Data type	Range of values	Description
LEN	INPUT	INT	Value > 0 The maximum total length of the data areas to be transferred can be found in the devicespecific Part B of the device manual in the "Performance data" chapter. This may differ for controller or device mode.	Length of the data area to be transferred in bytes. The transfer of the data always begins with address 0 regardless of the configuration. Please note that the IO address "0" with a length of 1 is included. **IO controller mode:** • The highest configured address of the devices must be specified here. The individual areas are not grouped together. If the instruction is called more than once, LEN can also be shorter than the highest address. The highest address should be specified in at least one call (compare "SEND" parameter). • The data is transferred in the order of the logical addresses (as with PROFIBUS DP). **IO device mode:** • The data is transferred in the order of the slots corresponding to the configuration of the input modules on the PROFINET IO controller line for this PROFINET IO device. Note: Make sure that the length programmed here and the configuration of the PROFINET IO controller are consistent. The entire data area length including any gaps is transferred for the device.
DONE	OUTPUT	BOOL	0: - 1: New data accepted	This parameter indicates whether or not the job was completed without errors.
ERROR	OUTPUT	BOOL	0: -1: Error	Error code
STATUS	OUTPUT	WORD	-	Status code

Parameter	Declaration	Data type	Range of values	Description
CHECK_IOCS	OUTPUT	BOOL	0: All IOCS set to GOOD 1: At least one IOCS set to BAD	Auxiliary bit that indicates whether or not it is necessary to evaluate the IOCS status area.
IOCS	OUTPUT	ANY (as VARTYPE only BYTE is permitted)	The address of the data area points to one of the alternatives: • Memory bit area • Data block area Length: For the maximum value, refer to the devicespecific Part B of this manual in the section "Performance data". This may differ for controller or device mode.	A status bit is transferred per byte of user data. The length information depends on the length in the LEN parameter (one bit per byte) = (Length LEN + 7/ 8) Controller mode: Address gaps are also transferred according to the SEND parameter. Address gaps are transferred with the status GOOD. Device mode: Address gaps are not transferred. The instruction begins the transfer of the status for address 0. Note: The minimum length of the ANY pointer is (length LEN + 7/8)

Note

Remember that all output parameters may only be evaluated when the instruction signals either DONE = 1 or ERROR = 1.

Note

You must assume that the returned IOCS status does not arrive timesynchronized with the data (SEND parameter) but delayed by one user program cycle. This means: User data and IOCS are not consistent.

See also

PNIO_SEND (Page 7756)

Parameters DONE, ERROR, STATUS (IE)

Condition codes

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

Note

For entries coded with $8Fxx_H$ in STATUS, refer to the information in the STEP 7 Standard and System Functions reference manual. The chapter describing error evaluation with the RET_VAL output parameter contains detailed information.

To find out which system instructions are used and are relevant for error evaluation, display the properties dialog of the instruction described here in the "Calls" tab.

Table 9-336 Condition codes PNIO_SEND

DONE	ERROR	STATUS	Meaning
0	0	8180_H	• Data transfer active; or • The CP is in STOP mode.
0	0	8181_H	Module does not support version 2.0 of the instruction Solution: Use version 1.0.
1	0	0000_H	New data transferred without error.
0	1	8183_H	• PROFINET IO configuration missing; or • wrong CPLADDR; or • The CP is in STOP mode. or • Interconnection of MODE does not match module configuration or incorrect interconnection with MODE > 1 Extra in device mode: • The connection between PROFINET IO controller and PROFINET IO device is down, or • PROFINET IO controller not reachable or • Total lengths (configuration and LEN parameter) are not consistent.
0	1	8184_H	System error or bad parameter type.
0	1	8185_H	Parameter LEN is greater than source area SEND or target buffer (IOCS) is too small.
0	1	$8F22_H$	Area length error reading a parameter (e.g. DB too short).
0	1	$8F23_H$	Area length error writing a parameter (e.g. DB too short).
0	1	$8F24_H$	Range error when reading a parameter.
0	1	$8F25_H$	Range error when writing a parameter.

DONE	ERROR	STATUS	Meaning
0	1	$8F28_H$	Alignment error when reading a parameter.
0	1	$8F29_H$	Alignment error when writing a parameter.
0	1	$8F30_H$	Parameter is in the write-protected 1st current data block.
0	1	$8F31_H$	Parameter is in the write-protected 2nd current data block.
0	1	$8F32_H$	Parameter contains a DB number that is too high.
0	1	$8F3A_H$	Destination area is not loaded (DB).
0	1	$8F42_H$	Timeout reading a parameter from the I/O area.
0	1	$8F43_H$	Timeout writing a parameter to the I/O area.
0	1	$8F44_H$	Access to a parameter to be read during execution of the instruction is prevented.
0	1	$8F45_H$	Access to a parameter to be written during execution of the instruction is prevented.
0	1	$8F7F_H$	Internal error, e.g. illegal ANY reference.
0	1	8090_H	Module with this address does not exist.
0	1	$80A0_H$	Negative acknowledgment writing to the module.
0	1	$80A1_H$	Negative acknowledgment writing to the module.
0	1	$80B0_H$	The module does not recognize the data record.
0	1	$80B1_H$	• The specified data record length is incorrect. or • The CP changes to STOP.
0	1	$80C0_H$	The data record cannot be read.
0	1	$80C1_H$	The specified data record is currently being processed.
0	1	$80C2_H$	There are too many jobs pending.
0	1	$80C3_H$	Resources occupied (memory).
0	1	$80C4_H$	Communication error (occurs temporarily, it is usually best to repeat the job in the user program).

See also

PNIO_SEND (Page 7756)

PNIO_RECV

PNIO_RECV

Description

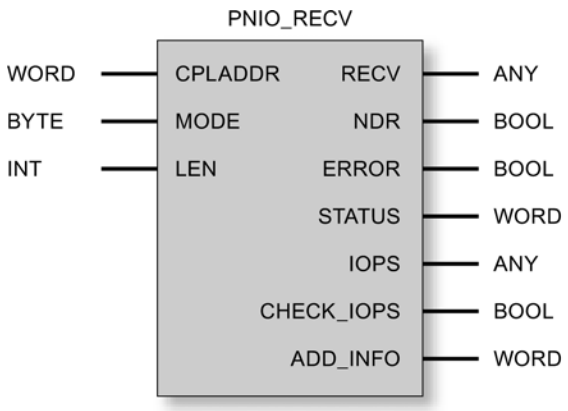
The PNIO_RECV instruction is used to receive data in the PROFINET IO controller or PROFINET IO device modes of the CP.

- Operating as PROFINET IO controller
  The instruction accepts the process data from PROFINET IO devices (inputs of the controller) and transfers the IO provider status (IOPS) from the PROFINET IO devices to the specified input areas.
- Operating as PROFINET IO device
  The instruction receives the data transferred by the PROFINET IO controller (configured O-addresses) as well as the IO Provider Status (IOPS) of the PROFINET IO controller and writes them to the data areas on the CPU of the PROFINET IO device reserved for the process outputs.

For CPs operating as PROFINET IO controller and IO device at the same time, the instruction is available as of version 2.0. With the additional MODE parameter, you set the mode for which the instruction will be called.

Call interface (version 2.0)

Call interface in FBD representation



See also

General characteristics of the instructions for PROFINET IO (Page 7768)

Data consistency (Page 7769)

Parameters for PNIO_RECV (Page 7764)

Parameters NDR, ERROR, STATUS (IE) (Page 7767)

Parameters for PNIO_RECV

Explanation of the formal parameters

The following table explains all the formal parameters for the PNIO_RECV instruction:

Parameter	Declaration	Data type	Range of values	Description
CPLADDR	INPUT	WORD	-	Module start address
MODE (parameters version 2.0 or later)	INPUT	BYTE	Values for X: • $0Y_H$: Status bits are transferred in IOCS. • $8Y_H$: Restriction to group message in CHECK_IOCS; no status bits in IOCS. Values for Y: • $X0_{H:}$ – IO controller mode – IO device mode (without parallel operation) Compatible with instruction in version 1.0 • $X1_H$: IO device mode (with parallel operation)	Specified in the form XY (hexadecimal): • X sets the transfer of status information. • Y specifies the mode of the CP as IO controller or IO device. Notes on compatibility;: • The version 1.0 instruction can continue to be used as long as the CP is not being operated as an IO controller and IO device at the same time. • When MODE=0, the instruction as of version 2.0 behaves like the instruction version 1.0.
RECV	IN_OUT	ANY (as VARTYPE only BYTE is permitted)	The address of the data area points to one of the alternatives: • Memory bit area • Data block area	Specifies the address and length IO controller mode: The length should match the total length of the distributed IO configured, whereby address gaps are also transmitted. The length can also be shorter than the total length of the distributed IO, for example when the instruction is called more than once in one OB. It must, however, have the total length in at least one call. IO device mode: The data structure results from the order of the slots of the output modules configured for this PROFINET IO device on the PROFINET IO controller line and their length without address gaps. Notes: • The instruction begins to transfer the data at address 0 regardless of how you configured the addresses (regardless of the lowest configured address). • Specifying an I/O area is not permitted since you must first change the IOPS for GOOD before data can be accepted in the I/O.

Parameter	Declaration	Data type	Range of values	Description
LEN	INPUT	INT	Value > 0 The maximum total length of the data to be transferred can be found in Part B of the device manual in the "Performance data" chapter. This may differ for controller or device mode.	Length of the data area to be transferred in bytes. The transfer of the data always begins with address 0 regardless of the configuration. Please note that the IO address "0" with a length of 1 is included. IO controller mode: • The highest configured address of the devices must be specified here. The individual areas are not grouped together. If the instruction is called more than once, LEN can also be shorter than the highest address. The highest address should be specified in at least one call (compare "RECV" parameter). • The data is transferred in the order of the logical addresses (as with PROFIBUS DP). IO device mode: • The data is transferred in the order of the slots corresponding to the configuration of the input modules on the PROFINET IO controller line for this PROFINET IO device. • Note: Make sure that the length programmed here and the configuration of the PROFINET IO controller are consistent. The entire data area length including any gaps is transferred for the device.
NDR	OUTPUT	BOOL	0: - 1: Data accepted	This parameter indicates whether or not the job was completed without errors.
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code
STATUS	OUTPUT	WORD	-	Status code
CHECK_ IOPS	OUTPUT	BOOL	0: All IOPS set to GOOD 1: At least one IOPS set to BAD	Auxiliary bit that indicates whether or not it is necessary to evaluate the IOPS status area.

Parameter	Declaration	Data type	Range of values	Description
IOPS	OUTPUT	ANY (as VARTYPE only BYTE is permitted)	The address of the data area points to one of the alternatives: • Memory bit area • Data block area Length: For the maximum value, refer to Part B of the device manual in the section "Performance data". This may differ for controller or device mode.	A status bit is transferred per byte of user data. The length information depends on the length in the RECV parameter (one bit per byte) = (Length LEN + 7/ 8) Controller mode: Address gaps are also transferred according to the RECV parameter. Address gaps are transferred with the status GOOD. Device mode: Address gaps are not transferred. The instruction begins the transfer of the status for address 0. Note: • The minimum length of the ANY pointer is (length LEN + 7/8)
ADD_INFO	OUTPUT	WORD	Additional Diagnostic Information In controller mode: • 0: No alarm • >0: Number of pending alarms In device mode, the parameter is always = 0.	Parameter expansion Note: The ADD_INFO parameter is also updated when there are no INPUT addresses configured on the PROFINET IO controller. In this case, the PNIO_RECV instruction is called with a length LEN > 0 (for example LEN = 1 byte). It then transfers an address gap of 1 byte. The parameter expansion can be used for CPs as of the following firmware version: • CP 343−1 (EX30) as of firmware V2.0 • CP 343−1 Lean (CX10) as of firmware V2.0 • CP 343−1 Advanced (GX30) as of firmware V1.0 In older firmware versions, the parameter is reserved.

Note

Remember that all output parameters may only be evaluated when the instruction signals either NDR = 1 or ERROR = 1.

See also

PNIO_RECV (Page 7763)

Parameters NDR, ERROR, STATUS (IE)

Condition codes

The following table shows the codes formed by the NDR, ERROR and STATUS parameters that must be evaluated by the user program.

Note

For entries coded with $8Fxx_H$ in STATUS, refer to the information in the STEP 7 Standard and System Functions reference manual. The chapter describing error evaluation with the RET_VAL output parameter contains detailed information.

To find out which system instructions are used and are relevant for error evaluation, display the properties dialog of the instruction described here in the "Calls" tab.

Table 9-337 Condition codes PNIO_RECV

NDR	ERROR	STATUS	Meaning
0	0	8180_H	• Data acceptance active; or • The CP is in STOP mode.
0	0	8181_H	Module does not support version 2.0 of the instruction. Solution: Use version 1.0.
1	0	0000_H	New data accepted without error.
0	1	8183_H	• PROFINET IO configuration missing; or • wrong CPLADDR; or • The CP is in STOP mode. or • Interconnection of MODE does not match module configuration or incorrect interconnection with MODE > 1. Extra in device mode: • The connection between PROFINET IO controller and PROFINET IO device is down or • PROFINET IO controller not reachable or • Total lengths (configuration and LEN parameter) are not consistent
0	1	8184_H	System error or bad parameter type.
0	1	8185_H	Destination buffer (RECV of IOCS) is too small.
0	1	$8F22_H$	Area length error reading a parameter (e.g. DB too short).
0	1	$8F23_H$	Area length error writing a parameter (e.g. DB too short).
0	1	$8F24_H$	Range error when reading a parameter.

NDR	ERROR	STATUS	Meaning
0	1	$8F25_H$	Range error when writing a parameter.
0	1	$8F28_H$	Alignment error when reading a parameter.
0	1	$8F29_H$	Alignment error when writing a parameter.
0	1	$8F30_H$	Parameter is in the write-protected 1st current data block.
0	1	$8F31_H$	Parameter is in the write-protected 2nd current data block.
0	1	$8F32_H$	Parameter contains a DB number that is too high.
0	1	$8F3A_H$	Destination area is not loaded (DB).
0	1	$8F42_H$	Timeout reading a parameter from the I/O area.
0	1	$8F43_H$	Timeout writing a parameter to the I/O area.
0	1	$8F44_H$	Access to a parameter to be read during execution of the instruction is prevented.
0	1	$8F45_H$	Access to a parameter to be written during execution of the instruction is prevented.
0	1	$8F7F_H$	Internal error, e.g. illegal ANY reference.
0	1	8090_H	Module with this address does not exist.
0	1	$80A0_H$	Negative acknowledgment writing to the module.
0	1	$80A1_H$	Negative acknowledgment writing to the module.
0	1	$80B0_H$	The module does not recognize the data record.
0	1	$80B1_H$	• The specified data record length is incorrect. or • The CP changes to STOP.
0	1	$80C0_H$	The data record cannot be read.
0	1	$80C1_H$	The specified data record is currently being processed.
0	1	$80C2_H$	There are too many jobs pending.
0	1	$80C3_H$	Resources occupied (memory).
0	1	$80C4_H$	Communication error (occurs temporarily, it is usually best to repeat the job in the user program).

See also

PNIO_RECV (Page 7763)

General characteristics of the instructions for PROFINET IO

IO Consumer Status (IOCS) and IO Provider Status (IOPS)

For both communication partners - CPU/CP on the one hand and IO device on the other - there is the status information GOOD or BAD for the data. This status information is transferred parallel to the data. The status of the partner that sends the data is called IOPS (IO Provider Status), the status of the receiving partner is called IOCS (IO Consumer Status).

The IOPS and IOCS status are not necessarily identical. It is, for example, possible that the S7-300 CPU is in STOP mode (output disable or no PROFINET IO instructions active). In this case, the CP as PROFINET IO controller transfers the BAD status to the IO devices.

Relationship between instruction call and IO data

- Operation as PROFINET IO controller
  As a PROFINET IO controller, the CP does not monitor the cyclic calls of the PNIO_SEND/RECV instructions. If the instructions are not called, the last transferred IO data and IOCS/IOPS data are taken as valid.
- Operation as PROFINET IO device
  PNIO_SEND and PNIO_RECV each have their own watchdog. Depending on the CPU cycle time, the connection to the PROFINET IO controller is terminated if one of the two instructions is no longer called following the initialization phase.

Optimizing data transfer (only when operating as PROFINET IO controller)

It is possible to call the instructions with a length (LEN parameter) that is shorter than the configured total length of the IO data on the PNIO chain.

You can use this so that timecritical data is transferred in every CPU cycle whereas non critical data is not transferred in every cycle.

Example:
You could, for example, transfer only the first data area (timecritical data) in every cycle and the total length of the configured IO data in every second cycle. To do this, place the timecritical data in the lower area (starting at IO address 0) during configuration.

Data consistency

The entire input or output data area of the PROFINET IO controller is always transferred in its entirety and is therefore consistent.

- Operating as PROFINET IO controller
  Regardless of this, using the length information in the instruction call, you can also read or output an input or output area smaller than the configured area consistently.

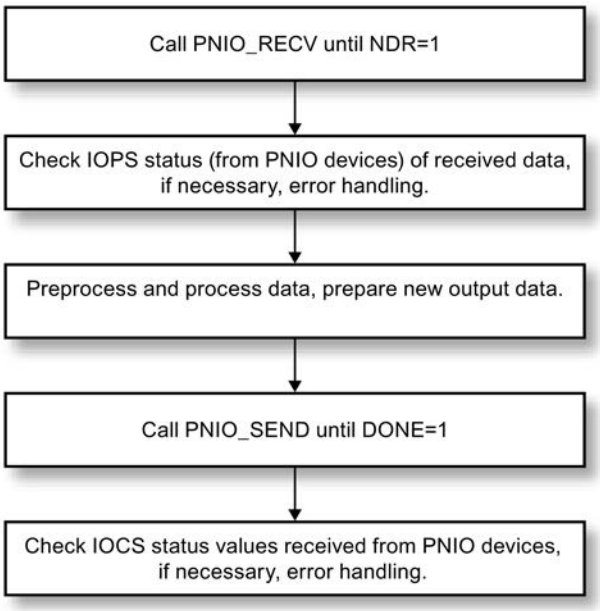
Note: You should, however, bear in mind that in terms of the "IO user data" within a PROFINET IO system, data consistency can only be guaranteed within the individual IO slots. This applies regardless of the fact that consistent data transfer between CPU and IO controller is guaranteed for the instructions described here.

Instruction call

To guarantee data consistency, you may, however, only access the IO data when the instruction has completed free of errors (output parameter NDR = TRUE). You must also check that the IOCS or IOPS status for the data is GOOD.

Example

In a normal situation (depending on the total length of the IO data), the instruction will run over several user program cycles until the condition code DONE/NDR = 1 is signaled.



Note: The user program cycle and the cycle of the IO data exchange between the PROFINET IO controller and PROFINET IO devices are independent of each other.

Substitute values

Operational situations

The setting of substitute values is supported for the two following operational situations:

- Substitute values during startup (mode change on the CPU from STOP to RUN)
- Substitute values if problems are detected (remove/insert or station failure/return)

Substitute values during startup

You can initialize the outputs with substitute values by setting a memory bit ("startup" memory bit) in the startup OB. In cyclic mode (OB1), evaluate this "startup" memory bit to call the PNIO_SEND instruction with the initialization values when appropriate.

Substitute values if a problem occurs (only when operating as PROFINET IO controller)

If there is a fault (device/submodule failed), you can find out which submodules have failed by querying the status information IOCS / IOPS status. You then have the option of setting substitute values.

PNIO_RW_REC

PNIO_RW_REC

Description

RW_REC is used both for the "read data record" and the "write data record" function in PROFINET IO controller mode. RW_REC can only execute one of the functions at any one time. The "read data record" or "write data record" function is controlled by the WRITE_REC parameter.

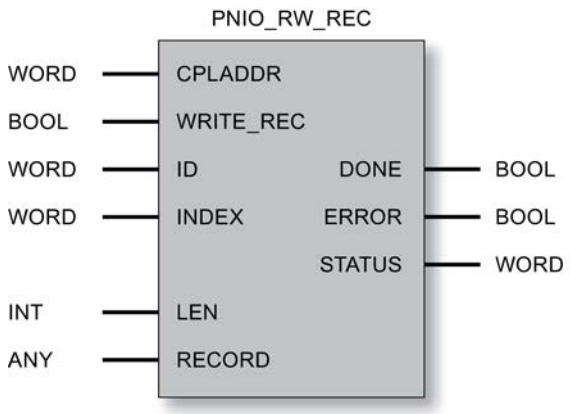
Example: The CP can be informed of the location ID and plant designation using the "write data record" function (if this parameter was not already set in the properties dialog of the CP in STEP 7). This is done using the maintenance data record "IM1" with index AFF1H.

You will find details of the supported data records and their structure at the following Internet address:

`http://support.automation.siemens.com/WW/view/en/19289930`

Call interface

Call interface in FBD representation:



See also

Parameters DONE, ERROR, STATUS (IE) (Page 7773)

Parameters for PNIO_RW_REC (Page 7772)

Parameters for PNIO_RW_REC

Explanation of the formal parameters

The following table explains all the formal parameters for the PNIO_RW_REC instruction:

Parameter	Declaration	Data type	Possible values	Description
CPLADDR	INPUT	WORD	-	Module start address
WRITE_REC	INPUT	BOOL	0: Read data record 1: Write data record	Job type; The parameter must not be changed while the instruction is executing.
ID	INPUT	WORD		Logical address of the PROFINET IO component (module or submodule). For an output module, bit 15 is set (example of output address 5: ID:=DW#16#8005). For a mixed module, the lower of the two addresses must be specified.
INDEX	INPUT	WORD	See vendor information for the data record numbers supported by the module.	Data record number that the user wants to read or write.
DONE	OUTPUT	BOOL	0: - 1: Data record transferred successfully	This parameter indicates whether or not the job was completed without errors.
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code
STATUS	OUTPUT	WORD	0: No error Different value: Error (see "Auto-Hotspot")	Status code
LEN	IN_OUT	INT	The maximum length is 480 bytes.	• Read data record: OUTPUT parameter only; after a successful read, the length of the read data record is indicated; otherwise 0. • Write data record: INPUT parameter only; length of the data record to be written is entered here by the user. The length must match the definition of the data record.
RECORD	IN_OUT	ANY (as VARTYPE, BYTE, WORD and DWORD are permitted)	The address of the data area points to one of the alternatives: • Memory bit area • Data block area The length of the ANY pointer must be greater than or equal to the definition of the data record.	• Read data record: OUTPUT parameter only; after a successful read, the data of the data record is stored here. If the ANY pointer is too short, as much data as possible is transferred. • Write data record: INPUT parameter only; the data to be written from the data record is stored here by the user. The ANY pointer must be at least as long as specified in the LEN parameter.

See also

PNIO_RW_REC (Page 7771)

Parameters DONE, ERROR, STATUS (IE)

Condition codes

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

Note

For entries with the coding $8Fxx_H$ under STATUS, note the information in the Reference Manual "STEP 7 - System and Standard Functions for S7-300 and S7-400". The chapter describing error evaluation with the RET_VAL output parameter contains detailed information.

Table 9-338 PNIO_RW_REC condition codes

DONE	ERROR	STATUS	Meaning
0	0	8180_H	Data transfer active
1	0	0000_H	Data record transferred successfully
0	1	8183_H	• No PROFINET IO controller configuration, • wrong CPLADDR or • CP in STOP mode
0	1	8184_H	System error or illegal parameter type
0	1	8185_H	Destination buffer (RECORD) is too short
0	1	$8F22_H$	Area length error reading a parameter (e.g. DB too short)
0	1	$8F23_H$	Area length error writing a parameter (e.g. DB too short)
0	1	$8F24_H$	Area error reading a parameter
0	1	$8F25_H$	Area error writing a parameter
0	1	$8F28_H$	Orientation error when reading a parameter
0	1	$8F29_H$	Alignment error writing a parameter
0	1	$8F30_H$	Parameter is in the write-protected first active data block
0	1	$8F31_H$	Parameter is in the write-protected second active data block
0	1	$8F32_H$	The DB number in the parameter is too high
0	1	$8F3A_H$	Destination area not loaded (DB)
0	1	$8F42_H$	Timeout reading a parameter from the I/O area
0	1	$8F43_H$	Timeout writing a parameter to the I/O area
0	1	$8F44_H$	Access to a parameter to be read during execution of the instruction is prevented
0	1	$8F45_H$	Access to a parameter to be written during execution of the instruction is prevented
0	1	$8F7F_H$	Internal error, e.g. illegal ANY reference
0	1	8090_H	Module with this address does not exist
0	1	$80A0_H$	Negative acknowledgment reading from the module

DONE	ERROR	STATUS	Meaning
0	1	$80A1_H$	Negative acknowledgment writing to the module
0	1	$80A3_H$	General PROFINET IO context management error
0	1	$80A9_H$	PROFINET IO device or module reports an illegal type
0	1	$80B0_H$	Module does not recognize the data record
0	1	$80B1_H$	• The specified data record length is incorrect or • The CP changes to STOP
0	1	$80B2_H$	The logical address or the configured slot is not in use
0	1	$80B4_H$	PROFINET IO device or module signaling access to an illegal area
0	1	$80B6_H$	PROFINET IO device or module denies access
0	1	$80B8_H$	The module is signaling an illegal parameter
0	1	$80C0_H$	The data record cannot be read
0	1	$80C1_H$	The specified data record is being processed
0	1	$80C2_H$	Too many jobs pending
0	1	$80C3_H$	Resources (memory) occupied
0	1	$80C4_H$	Communication error (occurs temporarily, it is usually best to repeat the job in the user program).

See also

PNIO_RW_REC (Page 7771)

PNIO_Alarm

PNIO_ALARM

Description

The PNIO_ALARM instruction is used for evaluation of alarms by a CP 343-1 operating as PROFINET IO controller and should be called in its user program if the parameter ADD_INFO does not equal 0 in PNIO_RECV. After full and error-free transfer of all OUTPUT parameters of the PNIO_ALARM, the received alarms are automatically acknowledged.

The alarms are forwarded to the user program in the chronological order in which they were signaled. Older alarms that have not yet been signaled to the user program and that become invalid due to more recent alarms are not deleted by the newer alarms.

Note

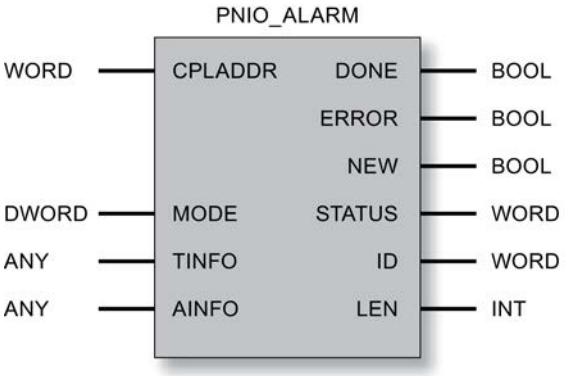
As long as PNIO_ALARM has not yet been called, the alarms are acknowledged automatically (internally) the CP.

If PNIO_ALARM has been called (at least) once in the user program, it must continue to be called to acknowledge pending alarms. This is the situation when the PNIO_RECV instruction signals a value not equal to "0" in the ADD_INFO parameter.

If PNIO_ALARM is no longer called after it has been called once or more in the user program, alarms are not acknowledged and there is no guarantee that the IO image will be updated correctly. The can occur, for example, following a station return alarm. The need to call PNIO_ALARM can only be reset by restarting the CP (power cycle).

Call interface

Call interface in FBD representation



See also

Parameters DONE, NEW, ERROR, STATUS (IE) (Page 7777)

Parameters for PNIO_ALARM (Page 7776)

Parameters for PNIO_ALARM

Explanation of the formal parameters

The following table explains all the formal parameters for the PNIO_ALARM instruction:

Parameter	Declaration	Data type	Possible values	Description
CPLADDR	INPUT	WORD	-	Start address of the module that caused the error
DONE	OUTPUT	BOOL	0: - 1: Alarm information transferred successfully	This parameter indicates whether or not the job was completed without errors. If DONE = 1, the NEW parameter must also be checked.
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code
NEW	OUTPUT	BOOL	0: Data transfer active or no new alarm 1: New alarm received and acknowledged	If DONE = 1 and NEW = 1, a new received alarm is signaled.
STATUS	OUTPUT	WORD	0: No error Different value: Error (see Table 1–14)	Status code
ID	OUTPUT	WORD		Logical start address of the PNIO component that triggers the alarm (module or submodule). For an output module, bit 15 is set (example of output address 5: ID:=DW#16#8005). For a mixed module, the lower of the two addresses is specified.
LEN	OUTPUT	INT		Length of the received alarm information (AINFO)
MODE	IN_OUT	DWORD	0	Reserved

Parameter	Declaration	Data type	Possible values	Description
TINFO	IN_OUT	ANY (as VARTYPE, BYTE, WORD and DWORD are permitted)	The address of the data area points to one of the alternatives: • Memory bit area • Data block area The length of the ANY pointer must be >= 32 bytes.	(task information) Destination area for the alarm management information. The error OB start information (OB header = byte 0 to 19 of TINFO) is reproduced as far as possible by the CP firmware. See also 1)
AINFO	IN_OUT	ANY (as VARTYPE, BYTE, WORD and DWORD are permitted)	The address of the data area points to one of the alternatives: • Memory bit area • Data block area The length of the ANY pointer must be greater than or equal to the maximum additional alarm information that can be expected, maximum 1432 bytes (see LEN parameter)	(alarm information) Destination area for header information and additional alarm information. If the ANY pointer AINFO is too low, the information will be truncated. See also 1)

1) Reference Manual "STEP 7 - System and Standard Functions for S7-300 and S7-400", receiving an alarm with the "RALRM" instruction

See also

PNIO_ALARM (Page 7774)

Parameters DONE, NEW, ERROR, STATUS (IE)

Condition codes

The following table shows the condition codes formed by the DONE, NEW, ERROR and STATUS parameters that must be evaluated by the user program.

Note

For entries with the coding $8Fxx_H$ under STATUS, note the information in the Reference Manual "STEP 7 - System and Standard Functions for S7-300 and S7-400". The chapter describing error evaluation with the RET_VAL output parameter contains detailed information.

Table 9-339 PNIO_ALARM status codes

DONE	NEW	ERROR	STATUS	Meaning
0	0	0	8180_H	Data transfer active
1	1	0	0000_H	Alarm data successfully transferred and alarm acknowledged
1	0	0	0000_H	No alarm data exist
0	0	1	8183_H	• No PROFINET IO controller configuration, • wrong CPLADDR or • CP in STOP mode
0	0	1	8184_H	System error or illegal parameter type
0	0	1	8185_H	Destination buffer (TINFO or AINFO) is too short
0	0	1	$8F22_H$	Area length error reading a parameter (e.g. DB too short)
0	0	1	$8F23_H$	Area length error writing a parameter (e.g. DB too short)
0	0	1	$8F24_H$	Area error reading a parameter
0	0	1	$8F25_H$	Area error writing a parameter
0	0	1	$8F28_H$	Orientation error when reading a parameter
0	0	1	$8F29_H$	Alignment error writing a parameter
0	0	1	$8F30_H$	Parameter is in the write-protected first active data block
0	0	1	$8F31_H$	Parameter is in the write-protected second active data block
0	0	1	$8F32_H$	The DB number in the parameter is too high
0	0	1	$8F3A_H$	Destination area not loaded (DB)
0	0	1	$8F42_H$	Timeout reading a parameter from the I/O area
0	0	1	$8F43_H$	Timeout writing a parameter to the I/O area
0	0	1	$8F44_H$	Access to a parameter to be read during execution of the instruction is prevented.
0	0	1	$8F45_H$	Access to a parameter to be written during execution of the instruction is prevented.
0	0	1	$8F7F_H$	Internal error, e.g. illegal ANY reference
0	0	1	8090_H	Module with this address does not exist
0	0	1	$80A0_H$	Negative acknowledgment reading from the module
0	0	1	$80A1_H$	Negative acknowledgment writing to the module
0	0	1	$80B0_H$	Module does not recognize the data record
0	0	1	$80B1_H$	• The specified data record length is incorrect or • The CP changes to STOP
0	0	1	$80C0_H$	The data record cannot be read
0	0	1	$80C1_H$	The specified data record is being processed
0	0	1	$80C2_H$	Too many jobs pending
0	0	1	$80C3_H$	Resources (memory) occupied
0	0	1	$80C4_H$	Communication error (occurs temporarily, it is usually best to repeat the job in the user program).

See also

PNIO_ALARM (Page 7774)

PROFIenergy

PROFIenergy

The PROFIenergy functions in PROFINET are used for energy management of plants. These include the planned or spontaneous shutdown of individual field devices, units or plant sections to save energy. The shutdown takes place during times without production or during breaks in production. Energy and diagnostics data can be read from devices included in the energy concept and that support these functions.

PROFIenergy controller

The commands for shutting down are output by the higher-level controller, in PROFINET IO, the IO controller.

With the SIMATIC S7-300, an S7-300 CPU with PROFIenergy functionality can be the PROFIenergy controller.

PROFIenergy devices

The commands of the PROFIenergy controller are processed by the IO devices with PROFIenergy functionality to shut down connected devices in the field.

In the context of the PROFIenergy program blocks, an IO device with PROFIenergy functionality is known as a PROFIenergy device. With the SIMATIC S7-300, an S7-300 CPU with PROFIenergy functionality can be the PROFIenergy device.

I-devices

In SIMATIC S7, an intelligent device (I-device) itself can have subordinate PROFIenergy devices. In this case, the I-device also has the function of a PROFIenergy controller.

Energy saving modes and PE_MODE_ID

Many devices support only the operating states "ready to operate" (power ON) and "pause" (power OFF). Scaled energy-saving states with different energy-saving modes can be specified for devices that support this or for groups of units in the controller of the PROFIenergy device. With PROFIenergy, these various states of energy consumption can be assigned to the devices that will be shut down in the field.

The various states of energy consumption are known as "energy-saving modes". For each individual energy-saving mode, a defined "PE_MODE_ID" is specified.

Programming of the features of the energy saving modes

The details of the energy-saving modes (addressed field device, pause duration etc) are programmed in the user program of the CPU of the PROFIenergy device.

PROFIenergy program blocks for the CP 300

Implementation of the PROFIenergy functions in S7-300

With a SIMATIC S7-300, the PROFIenergy functions are provided by program blocks for the IO controller and the IO device.

Note that an S7-300 CPU and a CP 300 use different PROFIenergy program blocks.

PROFIenergy specification

The functions of the PROFIenergy program blocks for the CP 300 are based on the following specification of the PROFIBUS Users Organization (PNO):

Common Application Profile PROFIenergy, Technical Specification for PROFINET, Version 1.0, January 2010, Order No. 3.802

PROFIenergy program blocks for the CP 300

The PROFIenergy program blocks are called by the user program of the CPU. The following PROFIenergy program blocks are available for the PROFIenergy functions of the CP 300:

- CP 300 as IO controller:
  - PE_START_END_CP
    Program block for initiating and ending pauses for power supply and setting defined energy-saving modes for the PROFIenergy device.
  - PE_CMD_CP
    Program block for initiating and ending pauses for power supply and setting defined energy-saving modes and for querying measured energy values from the PROFIenergy device.

  The two program blocks can be used as alternatives. Compared with PE_START_END_CP, PE_CMD_CP has an expanded range of functions for the integration of measured energy values.
  For each PROFIenergy device, the program block must be called separately.

  - PE_DS3_WRITE_ET200_CP
    Does not belong to the PROFIenergy function blocks, but expands the PROFIenergy functions for an ET 200S.
    With PE_DS3_WRITE_ET200_CP, the settings for the switching behavior of up to 8 slots (in this case: power modules) of the ET 200S are specified.
- CP 300 as IO device:
  - PE_I_DEV_CP
    Receives all PROFIenergy commands and allows the user program to execute the PROFIenergy functions.
    Makes the response frames of the IO device available to the IO controller.
    PE_I_DEV_CP is called cyclically by the user program of the IO device.
  - Supplementary program blocks (FC 0...FC 8) for PE_I_DEV_CP:
    These FCs make the response data available for PE_I_DEV_CP. The FCs must be called in the user program and linked with PE_I_DEV_CP.

  If the PROFIenergy device is an I-device and itself has subordinate PROFIenergy devices, PE_START_END_CP or PE_CMD_CP is called in the CPU of the I-device for the subordinate PROFIenergy devices.

System and program blocks for transferring data records

The PROFIenergy commands and status information between IO controller and IO device are exchanged by reading and writing data records. This is implemented using the program blocks RDREC and RWREC.

The PROFIenergy data records are described below along with the response data of the individual program blocks.

Note

Block calls

PE_START_END_CP, PE_CMD_CP, PE_I_DEV_CP and PE_DS3_WRITE_ET200_CP must not be called at the same time. The next program block can only be called after one of these program blocks as signaled "no error" (VALID = 1) or "error" (ERROR = 1).

The program block PNIO_RW_REC must also not be called at the same time as PE_START_END_CP, PE_CMD_CP, PE_I_DEV_CP or PE_DS3_WRITE_ET200_CP.

PE_START_END_CP

Meaning and call - PE_START_END_CP

Significance and how it works

PE_START_END_CP can be used as an alternative to PE_CMD_CP.

PE_START_END_CP is used on the IO controller. It triggers an energy saving pause or ends a pause on the assigned PROFIenergy device.

The program block can be used ideally on IO controllers with IO devices that have only field devices connected to them and no energy data needs to be or can be read out from them.

The energy-saving modes are configured in the user program of the IO device. The energy-saving mode actually adopted is reported back by the IO device after execution of PE_START_END_CP and output at the PE_MODE_ID parameter.

The Pause_Time parameter specifies the length of the energy-saving pause for the IO device. On the IO device, the PE_I_DEV_CP program block checks whether or not the specified duration of the pause is adequately long and can be implemented.

Sequence

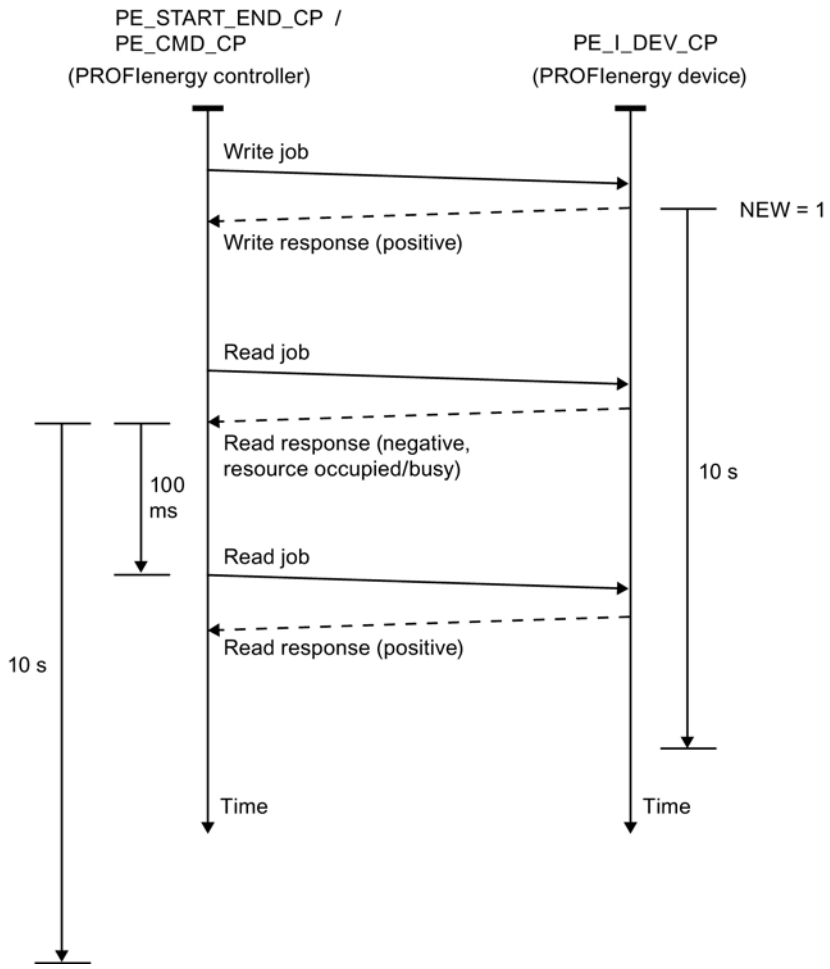


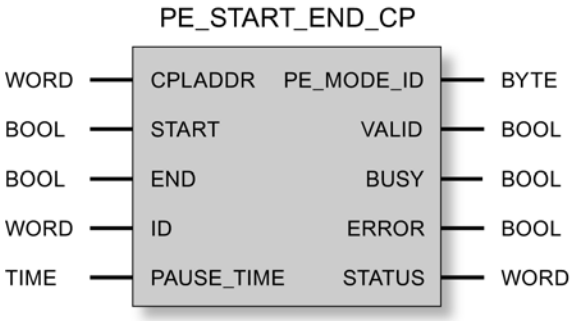
Figure 9-20 Flow chart of the write/read jobs of PE_START_END_CP and PE_CMD_CP

Using WRREC, PE_START_END_CP sends a PROFIenergy command as a write job to the IO device. Following this, PE_START_END_CP waits for the acknowledgment from the IO device. To achieve this, the acknowledgement data record is read every 100 milliseconds using the program block RDREC.

As long as no acknowledgement has arrived from the IO device, the read job is repeated for 10 seconds at intervals of 100 ms.

The response data of the IO device is read with RDREC.

Call interface in FBD representation



Explanation of the formal parameters of PE_START_END_CP

Explanation of the formal parameters of PE_START_END_CP

Parameter	Declaration	Data type	Range of values	Description
CPLADDR	INPUT	WORD	I, Q, M, D, L, const.	Module start address of the CP
START	INPUT	BOOL	• 1 = command active • 0 = command not active	A rising edge enables the "Start_Pause" command
END	INPUT	BOOL	• 1 = command active • 0 = command not active	A rising edge enables the "End_Pause" command
ID	INPUT	WORD		Logical address of the destination PROFIenergy device
PAUSE_ TIME	INPUT	TIME	T#-24D_20H_31M_23S_648MS to T#24D_20H_31M_23S_647MS	IEC time in steps of 1 ms, integer with sign
PE_MODE_ ID	OUTPUT	BYTE	• 00_h: Power OFF (pause) • 01_h...FE_h: Configurable • FF_h: Ready for operation	ID of the energy-saving mode adopted by the IO device after execution of the command.
VALID	OUTPUT	BOOL	0: - 1: Execution completed successfully	This parameter indicates whether or not the job was completed without errors.
BUSY	OUTPUT	BOOL	0: Execution completed, aborted or not yet started 1: Execution active	Condition code of the processing status of the program block

Parameter	Declaration	Data type	Range of values	Description
ERROR	OUTPUT	BOOL	0: - 1: Errors	Error code For the meaning in conjunction with the STATUS parameter, refer to Condition codes of PE_START_END_CP (Page 7785).
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the ERROR parameter, refer to Condition codes of PE_START_END_CP (Page 7785).

See also

Response data (Page 7791)

Condition codes of PE_START_END_CP

Condition codes of PE_START_END_CP

PE_START_END_CP is based on the program block PNIO_RW_REC and returns all condition codes of PNIO_RW_REC, see condition codes of the block PNIO_RW_REC.

The following additional PROFIenergy-specific condition codes are output. The error codes of STATUS are valid only in conjunction with ERROR = 1.

Table 9-340 Specific condition codes of PE_START_END_CP

STATUS	Meaning
Block-specific errors	
8080_h	Rising edge at START and END at the same time
8081_h	Length conflict between CMD_PARAM and CMD_PARAM_LEN
PROFIenergy-specific errors	
$FE01_h$	Invalid Service_Request_ID
$FE02_h$	Invalid Request_Reference
$FE03_h$	Invalid CMD_MODIFIER
$FE04_h$	Invalid information about the data structure of a command (Data_Structure_Identifier_RQ) in the frame for writing the PROFIenergy data record
$FE05_h$	Invalid information about the data structure of a command (Data_Structure_Identifier_RS) in the frame for reading the PROFIenergy data record
$FE06_h$	Energy saving mode (PE_Mode_ID) not supported
$FE07_h$	Response longer than max transfer length
$FE08_h$	Invalid number of commands
$FE09_h$	Invalid block type (see frame header)
$FE0A_h$	Invalid block length (see frame header)
$FE0B_h$	Invalid block version (see frame header)
$FE50_h$	Not a suitable energy saving mode (PE_Mode_ID)

STATUS	Meaning
$FE51_h$	Value for PAUSE_TIME not supported
$FE52_h$	PE_Mode_ID not supported

Details on the parameters of the PROFIenergy-specific errors can be found in the section Response data.

PE_CMD_CP

Meaning and call - PE_CMD_CP

Significance and how it works

PE_CMD_CP can be used as an alternative to PE_START_END_CP.

PE_CMD_CP is used on the IO controller and initiates an energy-saving pause or ends a pause on the assigned PROFIenergy device. PE_CMD_CP can also read out further information and energy measured values from an IO device.

The program block can be used ideally on IO controllers with IO devices that have field devices connected to them and energy data needs to be read out from them.

You will find a flowchart of the write/read jobs of PE_CMD_CP in section Meaning and call - PE_START_END_CP.

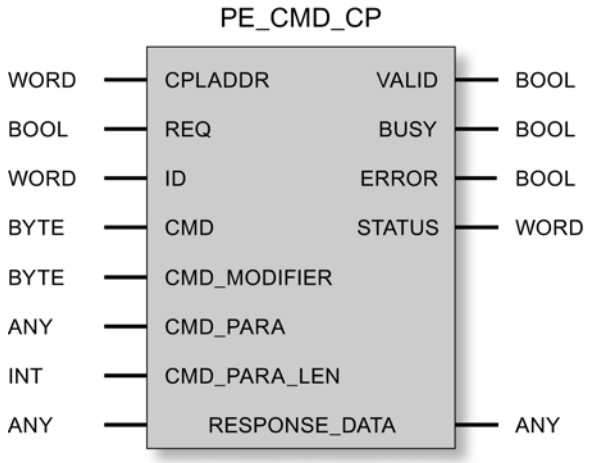
The individual commands that can be transferred to the IO device with the program block are assigned defined "Service_Request_IDs". The Service_Request_IDs 01...05 and 16 are assigned in the CMD parameter.

The CMD_MODIFIER parameter specifies the two commands 04 (Query_Modes) and 16 (Query_Measurement) in greater detail.

The CMD_PARA parameter assigns the values for certain parameters to some commands using an Any pointer. The CMD_PARA_LEN parameter specifies the length of this parameter.

The RESPONSE_DATA parameter points to the data area of the response data of the IO device.

Call interface in FBD representation



See also

Explanation of the formal parameters of PE_CMD_CP (Page 7787)

Explanation of the formal parameters of PE_CMD_CP

Explanation of the formal parameters of PE_CMD_CP

Parameters	Declaration	Data type	Range of values	Description
CPLADDR	INPUT	WORD	I, Q, M, D, L, const.	Module start address of the CP
REQ	INPUT	BOOL		Starts the transfer of the PROFIenergy commands on a rising edge.
ID	INPUT	WORD		Logical address of the destination PROFIenergy device
CMD	INPUT	BYTE	• 01: Start_Pause • 02: End_Pause • 03: Query_Modes • 04: PEM_Status • 05: PE_Identity • 16: Query_Measurement	Service_Request_ID of the PROFIenergy command. You will find the meaning of the commands below this table.

Parameters	Declaration	Data type	Range of values	Description
CMD_ MODIFIER	INPUT	BYTE	For "Start_Pause": 00 For "End_Pause": 00 For "Query_Modes": • 01: List_Energy_Saving _Modes • 02: Get_Mode For "PEM_Status": 00 For "PE_Identity": 00 For "Query_Measurement": • 01: Get_Measurement_List • 02: Get_Measurement_Values	Modifier of the PROFIenergy command, meaning: • "Query_Modes" command – Modifier 01: Reads all supported energy-saving modes (PE_Mode_ID). – Modifier 02 reads the parameters of the selected PE_Mode_ID. • "Query_Measurement" command – Modifier 01: Reads the configured Measurement_IDs. – Modifier 02: Reads the measured values of the selected Measurement_IDs. You will find information on the parameters in Response data in the section for the particular command. Modifier 00 means "no options".
CMD_PARA	INPUT	ANY		Any pointer to parameters for commands • For command 01 Start_Pause: "Pause_Time" • For command 02 End_Pause: Irrelevant • For command 03 Query_Modes: – For modifier 01: Irrelevant – For modifier 02 Get_Mode: "PE_Mode_ID" • For command 04 PEM_Status: Irrelevant • For command 05 PE_Identity: Irrelevant • For command 16 Query_Measurement: – For modifier 01: Irrelevant – For modifier 02 Get_Measurement_Values: ANY pointer to the data structure with the parameters "Count" and "Measurement_IDs" You will find information on the parameters in Response data in the section for the particular command. The entire data area of the data record to be written (Service_Data_Request) is entered. Maximum length: 234 bytes
CMD_PARA _LEN	INPUT	INT		Actual length of the parameters in CMD_PARA. Max. length: 234 bytes
RESPONSE _DATA	INOUT	ANY		Pointer to the address of the response data of the IO device (complete frame including block header) Note: If the area selected is not large enough, only the configured number of bytes is saved.

Parameters	Declaration	Data type	Range of values	Description
VALID	OUTPUT	BOOL	0: - 1: Execution completed successfully	The status parameter of the program block indicates whether or not the job was completed without errors.
BUSY	OUTPUT	BOOL	0: Execution not yet started, completed or aborted 1: Execution active	Condition code of the processing status of the program block
ERROR	OUTPUT	BOOL	0: - 1: Errors	Error code For the meaning in conjunction with the STATUS parameter, refer to Condition codes of PE_CMD_CP.
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the ERROR parameter, refer to Condition codes of PE_CMD_CP.

Service_Request_IDs and meaning of the PROFIenergy commands

The PROFIenergy commands with Service_Request_ID 01...05 and 16 have the following significance:

- 01 = Start_Pause
  Command for starting an energy-saving pause.
  The IO device selects the configured energy-saving mode. The energy-saving mode is reported back to the controller in the response data.
- 02 = End_Pause
  Command for ending an energy saving pause
- 03 = Query_Modes
  Queries the configured energy-saving modes with all corresponding time and energy information on the IO device.
  The queried information is detailed using the CMD_MODIFIER parameter:
  - List_Energy_Saving_Modes
    Reads all supported PROFIenergy modes of the IO device.
  - Get_Mode
    Reads the data of the selected PROFIenergy mode.
- 04 = PEM_Status
  Query of the energy-saving mode actually adopted by the field device or the unit group.
- 05 = PE_Identity
  Queries the PROFIenergy services supported by the IO device.
- 16 = Query_Measurement
  Queries the energy data of the IO device.
  The queried information is detailed using the CMD_MODIFIER parameter:
  - Get_Measurement_List
    Reads all the configured Measurement_IDs on the device.
  - Get_Measurement_Values
    Reads the measured energy values of the selected Measurement_IDs.

Commands for various device classes

The devices that can be included in PROFIenergy concepts can be divided into three classes that are addressed by the IO controller with different commands:

- IO modules, actuators, motor starters
  Supported commands:
  - Start_Pause, End_Pause
  - Query_Modes, PEM_Status, PE_Identify
- Measuring devices for electrical variables
  Supported commands:
  - Query_Measurement
- Frequency converters
  Supported commands:
  - Start_Pause, End_Pause
  - Query_Modes, PEM_Status, PE_Identify
  - Query_Measurement
    Data of electrical variables data acquired by frequency converters can also be queried.

Condition codes of PE_CMD_CP

Condition codes of PE_CMD_CP

PE_CMD_CP is based on the program block PNIO_RW_REC and returns all condition codes of PNIO_RW_REC, see condition codes of the block PNIO_RW_REC.

The following additional PROFIenergy-specific condition codes are output. The error codes of STATUS are valid only in conjunction with ERROR = 1.

Table 9-341 Specific condition codes of PE_CMD_CP

STATUS	Meaning
Block-specific errors	
8081_h	Length conflict between CMD_PARAM and CMD_PARAM_LEN
PROFIenergy-specific errors	
$FE01_h$	Invalid Service_Request_ID
$FE02_h$	Invalid Request_Reference
$FE03_h$	Invalid CMD_MODIFIER
$FE04_h$	Invalid information on the data structure of a command (Data_Structure_Identifier_RQ) in the frame for the PROFIenergy data record to be written
$FE05_h$	Invalid information on the data structure of a command (Data_Structure_Identifier_RS) in the frame for the PROFIenergy data record to be read
$FE06_h$	Energy saving mode (PE_Mode_ID) not supported
$FE07_h$	Response longer than max transfer length
$FE08_h$	Invalid number of commands
$FE09_h$	Invalid block type (see frame header)

STATUS	Meaning
$FE0A_h$	Invalid block length (see frame header)
$FE0B_h$	Invalid block version (see frame header)
$FE50_h$	Not a suitable energy saving mode (PE_Mode_ID)
$FE51_h$	Value for PAUSE_TIME not supported
$FE52_h$	PE_Mode_ID not supported

Details on the parameters of the PROFIenergy-specific errors can be found in the section Response data.

Response data

Structure of the response data

The following table shows the structure of the data record ($80A0_h$) of the response data of PE_START_END_CP and PE_CMD_CP.

The following table shows an overview of the structure of the data record of the returned response data according to the PROFIenergy specification. The composition of the "Service Data Response" area is described below for the individual PROFIenergy commands.

Table 9-342 Structure of the response data

Block definitions	Attributes	Value	Data type	Description
Block header	BlockType	0801_h	Unsigned16	
	BlockLength		Unsigned16	Frame length (without the "BlockType" and "BlockLength" fields)
	BlockVersionHigh	01_h	Unsigned8	
	BlockVersionLow	00_h	Unsigned8	
Response header	Service_Request_ID	01_h...FF_h	Unsigned8	01_h: Start_Pause 02_h: End_Pause 03_h: Query_Modes 04_h: PEM_Status 05_h: PE_Identify 06_h...09_h: - Reserved - 10_h: Query_Measurement 11_h...CF: - Reserved - $D0_h$...FF_h: Vendor-specific
	Request_Reference	01_h...FF_h	Unsigned8	Identification number of the query (mirrored in the response of the IO device)

Block definitions	Attributes	Value	Data type	Description
Service header response	Status	01_h...FF_h	Unsigned8	00_h: - Reserved - 01_h: Done 02_h: Done with error(s) 03_h: Data incomplete 04_h...CF_h: - Reserved - $D0_h$...FF_h: Depends on the Service_Request_ID
	Data_Structure_Identifier_RS	01_h...FF_h	Unsigned8	00_h: - Reserved - 01_h...FF_h: Data structure dependent on the Service_Request_ID FF_h: error
Service data response				Response data of the IO device Depending on the particular PROFIenergy command (described below)

Meaning of "Service data request" and "Service data response"

The following sections explain the parameter values for the queries of the IO controller to the IO device (Service Data Request) for each PROFIenergy command and the structure of the response data of the IO device (Service Data Response).

- Service data request
  Parameter values for IO controller queries
- Service data response
  Structure of the response data of the IO device

PROFIenergy command "Start_Pause"

- Service data request
  - CMD = 01
  - CMD_MODIFIER = 00
  - CMD_PARA_LEN = 04
  - CMD_PARA = Any pointer to the value for "Pause_Time" (data type "TIME")
    IEC time in steps of 1 ms, integer with sign
    Value: T#-24D_20H_31M_23S_648MS to T#24D_20H_31M_23S_647MS
- Service data response

Parameters	Value	Data type
PE_Mode_ID *	01_h...FF_h	Unsigned8
- Reserved -	00_h	Unsigned8

* Identification number of the energy-saving mode

PROFIenergy command "End_Pause"

- Service data request
  - CMD = 02
  - CMD_MODIFIER = 00
  - CMD_PARA_LEN = 00
  - CMD_PARA = irrelevant
- Service data response

Parameters	Value	Data type
Time_to_operate *		Unsigned32

* Expected time for switching over the PROFIenergy device to "ready to operate"

PROFIenergy command "Query_Modes" – List_Energy_Saving_Modes

- Service data request
  - CMD = 03
  - CMD_MODIFIER = 01
  - CMD_PARA_LEN = 00
  - CMD_PARA = irrelevant
- Service data response

Parameters	Value	Data type
Number_of_PE_Mode_IDs *	01_h	Unsigned8
PE_Mode_IDs		Unsigned8 array of Number_of_PE_Mode_IDs (unique ID for mode)

* Number of energy-saving modes

PROFIenergy command "Query_Modes" – Get_Mode

- Service data request
  - CMD = 03
  - CMD_MODIFIER = 02
  - CMD_PARA_LEN = 01
  - CMD_PARA = Any pointer to value for PE_MODE_ID (unsigned8)
- Service data response

Parameters	Value	Data type
PE_Mode_ID	01_h...FF_h	Unsigned8
PE_Mode_Attributes *	00_h...01_h	Unsigned8
Time_min_Pause		Unsigned32
Time_to_Pause		Unsigned32
Time_to_operate		Unsigned32
Time_min_length_of_stay		Unsigned32
Time_max_length_of_stay		Unsigned32
Mode_Power_Consumption		Float32
Energy_Consumption_to_pause		Float32
Energy_Consumption_to_operate		Float32

* Coding of bit 0:
0 = Only static time and energy measured values available.
1 = Dynamic time and energy measured values available.
Bits 1...7: Reserved

PROFIenergy command "PEM_Status"

- Service data request
  - CMD = 04
  - CMD_MODIFIER = 00
  - CMD_PARA_LEN = 00
  - CMD_PARA = irrelevant
- Service data response

Parameters	Value	Data type
PE_Mode_ID_Source *	00_h 01_h...FE_h FF_h	Unsigned8
PE_Mode_ID_Destination *	00_h 01_h...FE_h FF_h	Unsigned8
Time_to_operate		Unsigned32
Remaining_time_to_destination		Unsigned32
Mode_Power_Consumption		Float32
Energy_Consumption_to_Destination		Float32
Energy_Consumption_to_operate		Float32

* Possible values for "PE_Mode_ID_Source" and "PE_Mode_ID_Destination":
00_h: PE_Power_off
01_h...FE_h: Freely configurable
FF_h: PE_Ready_to_operate

PROFIenergy command "PE_Identify"

- Service data request
  - CMD = 05
  - CMD_MODIFIER = 00
  - CMD_PARA_LEN = 00
  - CMD_PARA = irrelevant
- Service data response

Parameters	Value	Data type
Count *	6	Unsigned8
Start_Pause **	01_h	Unsigned8
End_Pause	02_h	Unsigned8
Query_Modes	03_h	Unsigned8
PEM_Status	04_h	Unsigned8
PE_Identify	05_h	Unsigned8
Query_Measurement ***	10_h	Unsigned8

* Number of supported PROFIenergy commands

** Service_Request_ID of the first supported PROFIenergy command

*** Service_Request_ID of the last supported PROFIenergy command

PROFIenergy command "Query_Measurement" – Get_Measurement_List

- Service data request
  - CMD = 16
  - CMD_MODIFIER = 01
  - CMD_PARA_LEN = 00
  - CMD_PARA = irrelevant
- Service data response

Parameters	Value	Data type
Count *		Unsigned8
- Reserved -		Unsigned8
Measurement_ID **		Unsigned16
Accuracy_Domain [1]		Unsigned8
Accuracy_Class [2]		Unsigned8
Range [3]		Float32
…		
Measurement_ID ***		Unsigned16
Accuracy_Domain [1]		Unsigned8

Parameters	Value	Data type
Accuracy_Class [2]		Unsigned8
Range [3]		Float32

* Number of Measurement_IDs

** First supported Measurement_ID

*** Last supported Measurement_ID

[1] Accuracy domain (range 1...4):
0 = Reserved
1 = Percentage of the measuring range
2 = Percentage of the current measured values
3 = Accuracy according to IEC 61557-12
4 = Accuracy according to EN 50470-3 section 8

[2] Accuracy class (range 1...15):
0 = Reserved
1 (0.01%) ... 15 (>20%)

[3] Measuring range if Accuracy_Domain = 1; otherwise undefined

PROFIenergy command "Query_Measurement" – Get_Measurement_Values

- Service data request
  - CMD = 16
  - CMD_MODIFIER = 02
  - CMD_PARA_LEN = length of the data structure in bytes
  - CMD_PARA = Any pointer to data structure with the following structure:

Parameters	Value	Data type
Count *		Unsigned8
- Reserved -		Unsigned8
Measurement_ID **		Unsigned16
…		
Measurement_ID ***		Unsigned16

* Number of Measurement_IDs

** First queried measured value

*** Last queried measured value

- Service data response

Parameters	Value	Data type
Count *		Unsigned8
- Reserved -		Unsigned8
Length_of_Structure	0002_h...$FFFF_h$	Unsigned16
Measurement_Data_Structure_ID	1 = simple value	Unsigned8
Measurement_ID **	00_h...FF_h	Unsigned16

Parameters	Value	Data type
Status_of_Measurement_Value	1 = valid 2 = not available 3 = not available at times	Unsigned8
Transmission_Data_Type		Float32
End_of_demand		Unsigned32 or Unsigned16
Length_of_Structure		Unsigned16
Measurement_Data_Structure_ID		Unsigned8
Measurement_ID ***		Unsigned16
Status_of_Measurement_Value		Unsigned8
Transmission_Data_Type		Float32
End_of_demand		Unsigned32 or Unsigned16

* Number of Measurement_IDs

** First queried measured value

*** Last queried measured value

See also

Explanation of the formal parameters of PE_START_END_CP (Page 7784)

PE_I_DEV_CP

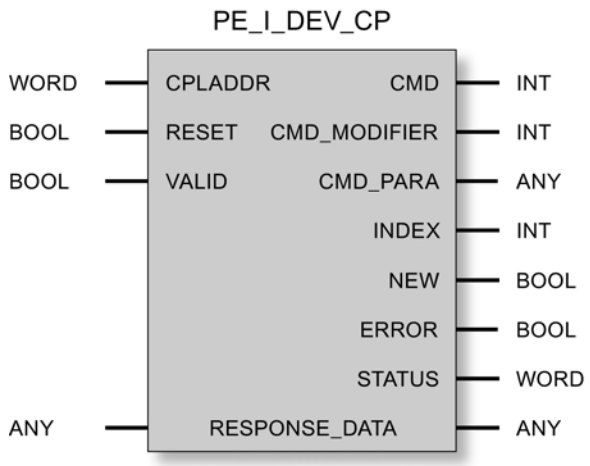
Meaning and call - PE_I_DEV_CP

Significance and how it works

The program block PE_I_DEV_CP is used on the PROFIenergy device where it handles the PROFIenergy commands of the IO controller. The PROFIenergy data records ($80A0_h$) sent by the IO controller are forwarded by the CP firmware to PE_I_DEV_CP. The PROFIenergy data of the IO device is made available to the IO controller as the response by PE_I_DEV_CP using the PROFIenergy data record ($80A0_h$).

The response data of PE_I_DEV_CP is generated by the supplementary functions FC 0 to FC 8, see section Supplementary program blocks for PE_I_DEV_CP.

Call interface in FBD representation



Explanation of the formal parameters of PE_I_DEV_CP

Explanation of the formal parameters of PE_I_DEV_CP

Parameters	Declaration	Data type	Range of values	Description
CPLADDR	INPUT	WORD	I, Q, M, D, L, const.	Module start address of the CP
RESET	INPUT	BOOL		Resets processing of the program block. NEW is set to 0.
VALID	INPUT	BOOL		If the response data was written to the relevant memory area of the IO device, VALID = 1 must be set by the user program. Following this, the program block makes the data available to the IO controller. NEW is set to 0.
CMD	OUTPUT	INT	• 01: Start_Pause • 02: End_Pause • 03: Query_Modes • 04: PEM_Status • 05: PE_Identity • 16: Query_Measurement	Service ID of the PROFIenergy command

Parameters	Declaration	Data type	Range of values	Description
CMD_ MODIFIER	OUTPUT	INT	• Modifier for Start_Pause: 00 • Modifier for End_Pause: 00 • Query_Modes, Modifier: – 01 (List_Energy_ Saving_Modes) – 02 (Get_Mode) • Modifier for PEM_Status: 00 • Modifier for PE_Identity: 00 • Query_Measurement, Modifier: – 01 (Get_Measurement _List) – 02 (Get_Measurement _Values)	Modifier of the PROFIenergy commands Meaning of the modifiers for commands: • "Query_Modes" command, Modifier: – 01 (List_Energy_Saving_Modes): Reads all supported PROFIenergy modes – 02 (Get_Mode): Reads the data of the selected PROFIenergy mode • "Query_Measurement" command, Modifier: – 01 (Get_Measurement_List): Reads all configured Measurement_IDs. – 02 (Get_Measurement_Values): Reads the measured values of the selected Measurement_IDs.
CMD_PARA	OUTPUT	ANY		Any pointer to parameters for the following command modifiers (compare CMD_MODIFIER parameter): • For "Get_Mode": PE_Mode_ID (ID of the energy-saving mode) length = 1 • For "Get_Measurement_Values": measured values of the Measurement_IDs length = max. 236 bytes (complete frame of the controller command without header)
INDEX	OUTPUT	INT		Number of the PROFIenergy data record ($80A0_h$)
NEW	OUTPUT	BOOL	0: Execution not yet started, completed or aborted 1: Execution active	Condition codes of the processing status of the program block
ERROR	OUTPUT	BOOL	0: - 1: Errors	Error code For the meaning in conjunction with the STATUS parameter, refer to Condition codes of PE_I_DEV_CP (Page 7800).

Parameters	Declaration	Data type	Range of values	Description
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the ERROR parameter, refer to Condition codes of PE_I_DEV_CP (Page 7800).
RESPONSE _DATA	INOUT	ANY	See "Response data" of the program block	Pointer to the data area of the response of the IO device (complete response frame including header). The data area must match the data area of the supplementary program blocks FC 0 - FC 8 (parameter "DATA_ERRORRSP"). Recommended size: At least 244 bytes. If the data area is too small, only the data of the configured bytes are transferred.

Condition codes of PE_I_DEV_CP

Condition codes of PE_I_DEV_CP

PE_I_DEV_CP is based on the program block PNIO_RW_REC and returns all condition codes of PNIO_RW_REC, see condition codes of the block PNIO_RW_REC.

Supplementary program blocks for PE_I_DEV_CP

Overview of the FCs

Function

The supplementary program blocks FC 0 to FC 8 support the preparation of the response data made available to the controller by PE_I_DEV_CP:

- For the response data of each PROFIenergy command, there is a separate program block (FC 1 - FC 8).
- FC 0 generates a common negative response for all PROFIenergy commands.

The FCs are called in the user program. In STEP 7 V5.5, they are available in the standard library in the "PROFIenergy" folder.

The FCs have several common parameters as well as individual parameters. Some of the common parameters of the FCs are interconnected with parameters of PE_I_DEV_CP. With some of the individual input parameters of the FCs, the response data is entered as plain language for the user or stored in the memory area of the IO device.

Overview of the FCs

The following supplementary program blocks are made available:

Table 9-343 Overview of the supplementary FCs

Number	Name
FC 0	PE_ERROR_RSP
FC 1	PE_START_RSP
FC 2	PE_END_RSP
FC 3	PE_LIST_MODES_RSP
FC 4	PE_GET_MODE_RSP
FC 5	PE_PEM_STATUS_RSP
FC 6	PE_IDENTIFY_RSP
FC 7	PE_MEASUREMENT_LIST_RSP
FC 8	PE_MEASUREMENT_VALUE_RSP

Interconnection of the FCs with PE_I_DEV_CP

Interconnection of the FCs with the program block PE_I_DEV_CP

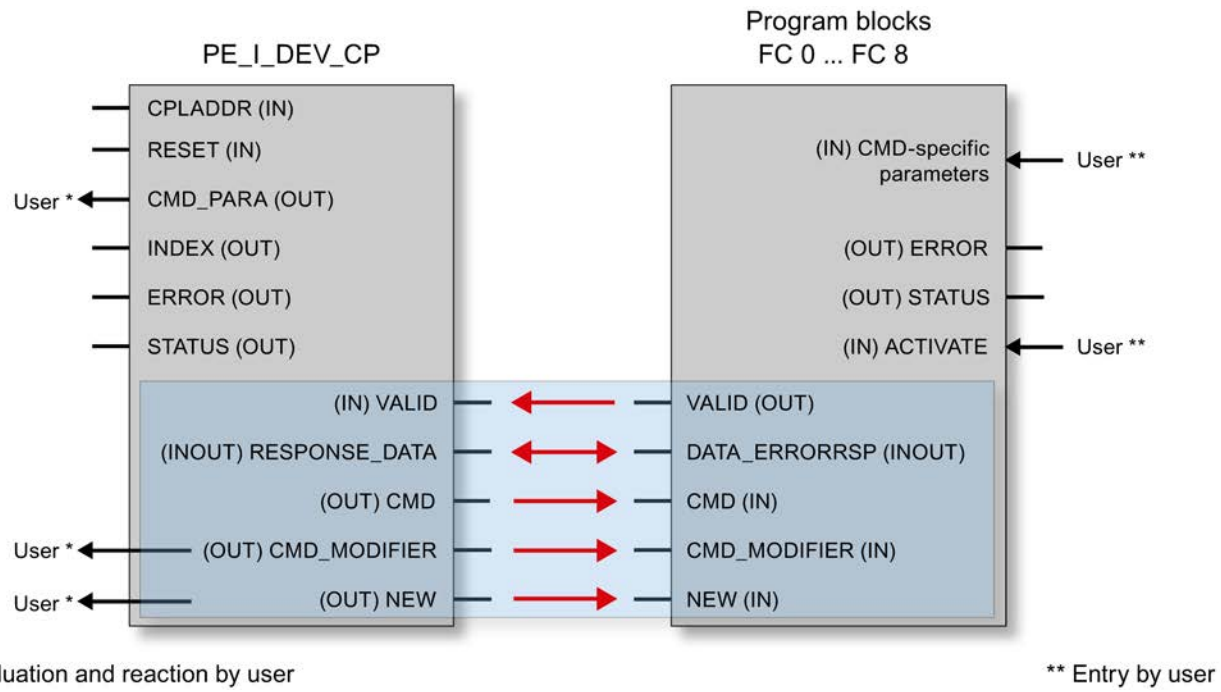


Figure 9-21 Interconnection of the FCs with PE_I_DEV_CP

Note

Interconnection of the program blocks is an absolute necessity

PE_I_DEV_CP must be interconnected with FC 0...FC 8 at the parameters shown on a light blue background that are assigned to the corresponding parameters of the FCs indicated by red arrows.

Common parameters of the FCs

Common parameters of the supplementary program blocks FC 0 - FC 8

Parameters	Declaration	Data type	Range of values	Description
Input parameters				
ACTIVATE	INPUT	BOOL		Instructs the block to copy the input parameters to the "DATA_ERRORRSP" data area are on a rising edge. Is then reset by the block. Must be set by the user within 10 seconds after a positive edge was detected at PE_I_DEV_NEW.
PE_I_DEV_NEW	INPUT	BOOL		Must be interconnected with the NEW output parameter of PE_I_DEV_CP. The block is processed only when 1 is set.
CMD	INPUT	INT		Must be interconnected with the CMD output parameter of PE_I_DEV_CP.
CMD_MODIFIER	INPUT	INT		Must be interconnected with the CMD_MODIFIER output parameter of PE_I_DEV_CP.
Output parameters				
DATA_ERRORRSP	OUTPUT	ANY		Pointer to the data area in which the response data will be stored (complete response frame including header). Must be interconnected with the RESPONSE_DATA output parameter of PE_I_DEV_CP. Recommended size: At least 244 bytes.
VALID	OUTPUT	BOOL	0: - 1: No error	Is set by the block. Must be interconnected with the VALID input parameter of PE_I_DEV_CP.
ERROR	OUTPUT	BOOL	0: No error 1: Errors	Error code
STATUS	OUTPUT	WORD	0: No error	Status code $80B1_h$: Error in ANY information (for example wrong area)

Individual parameters of the FCs

Individual parameters of FC 0 to FC 8

Below you will find a description of the individual parameters of the FCs.

PE_ERROR_RSP

Generates a negative response if the required PROFIenergy command is generally or temporarily not supported. The negative response is not dependent on the requesting command.

Table 9-344 Individual parameters of FC 0 PE_ERROR_RSP

Parameters	Declaration	Data type	Range of values	Description
ERROR_CODE	INPUT	BYTE		Error number

PE_START_RSP

Initiates an energy saving pause. Generates the response to the "Start_Pause" command. Returns the energy-saving mode adopted by the device.

Table 9-345 Individual parameters of FC 1 PE_START_RSP

Parameters	Declaration	Data type	Range of values	Description
PE_Mode_ID	INPUT	BYTE		ID of the energy-saving mode that the device or the unit group adopts.

Return message with the PE_Mode_ID of the energy-saving mode that the field devices or the unit group have adopted.

PE_END_RSP

Generates the response to the "End_Pause". command

Table 9-346 Individual parameters of FC 2 PE_END_RSP

Parameters	Declaration	Data type	Range of values	Description
Time_to_Operate	INPUT	DWORD		Time required to change from the current energy-saving mode after "ready to operate".

PE_LIST_MODES_RSP

Generates the response to the "Query_Modes" > modifier "List_Modes" command (list of the supported energy-saving modes).

The IDs of the energy-saving modes must be specified in the user program.

Table 9-347 Individual parameters of FC 3 PE_LIST_MODES_RSP

Parameters	Declaration	Data type	Range of values	Description
Number_of_PE_Mode_IDs	INPUT	BYTE		Number of supported energy-saving modes
PE_Mode_ID	INPUT	ANY	• 00_h • 01_h...FE_h • FF_h	Pointer to the area in which the energy-saving modes are stored. As the user, you will need to store the IDs of the energy-saving modes here. An energy-saving mode ID is configured in the Unsigned8 format. Permitted range: 1 to 254 bytes.

If the devices or a group need to react differently to different lengths of pause you can set up different energy-saving modes (PE_Mode) to achieve this. You assign a different PE_Mode_ID to the various energy-saving modes.

Possible values for "PE_Mode_ID":

- 00_h: PE_Power_off
- 01_h...FE_h: Freely configurable or vendor-specific
- FF_h: PE_Ready_to_operate

PE_GET_MODE_RSP

Generates the response to the "Query_Modes" > Modifier "Get_Mode". command

Table 9-348 Individual parameters of FC 4 PE_GET_MODE_RSP

Parameters	Declaration	Data type	Range of values	Description
PE_Mode_ID	INPUT	BYTE		Currently used energy-saving mode ID
Time_Min_Pause *	INPUT	Unsigned32		Minimum pause duration for this PE energy-saving mode. It is the sum of the three parameters: • Time_to_Pause • Time_to_operate • Time_min_length_of_stay
Time_to_Pause *	INPUT	Unsigned32		Time from the START edge until the requested energy-saving mode is reached
Time_to_operate *	INPUT	Unsigned32		Max. time after turn on before PE_ready_to_operate Time_to_operate can be used directly for the relevant calculations. The value can either be a static MAX value or can be calculated dynamically by the PE device.
Time_min_length_of_stay *	INPUT	Unsigned32		Minimum time that the PE device must remain in this PE_Mode.
Time_max_length_of_stay *	INPUT	Unsigned32		Maximum time that the PE device can remain in this PE_Mode.
Mode_Power _Consumption **	INPUT	Float32		Energy consumption in current PE_Mode [kW]

Parameters	Declaration	Data type	Range of values	Description
Energy_Consumption _to_pause **	INPUT	Float32		Energy consumption of PE_ready_to_operate until the current PE_Mode [kWh]
Energy_Consumption _to_operate **	INPUT	Float32		Energy consumption from current PE_Mode until PE_ready_to_operate [kWh]

* The PROFIenergy profile does not specify an invalid time format.
If the time is unlimited, the maximum value $FFFFFFFF_h$ can be specified.
If the time is zero, 00_h can be used.

** If an energy consumption value is not defined, 0.0 (Float32) can be specified.

PE_PEM_STATUS_RSP

Generates the response to the "PEM_STATUS". command

Table 9-349 Individual parameters of FC 5 PE_PEM_STATUS_RSP

Parameters	Declaration	Data type	Range of values	Description
PE_Mode_ID_Source	INPUT	BYTE	• 00_h • 01_h...FE_h • FF_h	ID of the energy-saving mode actually adopted
PE_Mode_ID_Destination	INPUT	BYTE	• 00_h • 01_h...FE_h • FF_h	ID of the energy-saving mode set by the controller
Time_to_operate *	INPUT	Unsigned32		Max. time after turn on before PE_ready_to_operate Time_to_operate can be used directly for the relevant calculations. The value can either be a static MAX value or can be calculated dynamically by the PE device.
Remaining_time_to _destination *	INPUT	Unsigned32		Optional: Time remaining until the requested PE_Mode. Dynamic value or static MAX value
Mode_Power _Consumption **	INPUT	Float32		Energy consumption in current PE_Mode [kW]
Energy_Consumption _to_Destination **	INPUT	Float32		Energy consumption until the requested PE_Mode [kWh]
Energy_Consumption _to_operate **	INPUT	Float32		Energy consumption from current PE_Mode until PE_ready_to_operate [kWh]

* The PROFIenergy profile does not specify an invalid time format.
If the time is unlimited, the maximum value $FFFFFFFF_h$ can be specified.
If the time is zero, 00_h can be used.

** If an energy consumption value is not defined, 0.0 (Float32) can be specified.

Possible values for "PE_Mode_ID_Source" and "PE_Mode_ID_Destination":

- 00_h: PE_Power_off
- 01_h...FE_h: Freely configurable or vendor-specific
- FF_h: PE_Ready_to_operate

PE_IDENTIFY_RSP

Generates the response to the "PE_Identify". command

As the user you need to specify which PROFIenergy commands are supported.

Table 9-350 Individual parameters of FC 6 PE_IDENTIFY_RSP

Parameters	Declaration	Data type	Range of values	Description
Count	INPUT	BYTE	0...6	Meaning of supported PROFIenergy commands
Start_Pause	INPUT	BOOL	0...1	• 1: Command is supported • 0: Command is not supported
End_Pause	INPUT	BOOL	0...1	• 1: Command is supported • 0: Command is not supported
Query_Modes	INPUT	BOOL	0...1	• 1: Command is supported • 0: Command is not supported
PEM_Status	INPUT	BOOL	0...1	• 1: Command is supported • 0: Command is not supported
PEM_Identify	INPUT	BOOL	0...1	• 1: Command is supported • 0: Command is not supported
Query_Measurement	INPUT	BOOL	0...1	• 1: Command is supported • 0: Command is not supported

PE_MEASUREMENT_LIST_RSP

Generates the response to the "Query_Measurement" > Modifier "Get_Measurement_List". command

Table 9-351 Individual parameters of FC 7 PE_MEASUREMENT_LIST_RSP

Parameters	Declaration	Data type	Range of values	Description
Count	INPUT	BYTE		Number of supported measured value IDs (Measurement_ID)
Measurement_List	INPUT	ANY		Pointer to the data area with the supported measured value IDs. As the user, you store the measured value IDs in this data area. Per frame, a maximum of 29 measured value IDs can be transferred. For information on the structure of the array, refer to section Response data > "Query_Measurement" – Get_Measurement_List.

PE_MEASUREMENT_VALUE_RSP

Generates the response to the "Query_Measurement" > Modifier "Get_Measurement_Values". command

Table 9-352 Individual parameters of FC 8 PE_MEASUREMENT_VALUE_RSP

Parameters	Declaration	Data type	Range of values	Description
Count	INPUT	BYTE		Number of supported Measurement_Values
Measurement_Values	INPUT	ANY		Pointer to the data area of the measured values (Measurement_Values). As the user, you store the measured values in this data area. Per frame, a maximum of 116 measured values can be transferred. For information on the structure of the array, refer to section Response data > "Query_Measurement" – Get_Measurement_List.

PE_DS3_WRITE_ET200S_CP

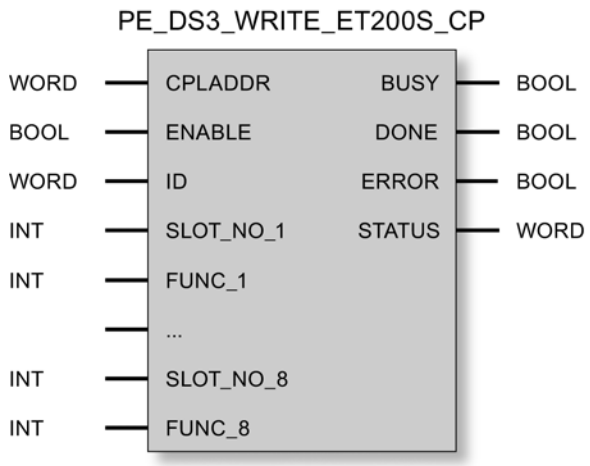
Meaning and call - PE_DS3_WRITE_ET200_CP

Significance and how it works

PE_DS3_WRITE_ET200_CP is used in the CPU of the CP 300 as a PROFIenergy controller to transfer the settings for switching behavior from power modules to an ET 200S. The switching behavior for up to 8 slots (in this case: power modules) can be transferred.

PE_DS3_WRITE_ET200_CP is not a PROFIenergy program block.

Call interface in FBD representation



Explanation of the formal parameters of PE_DS3_WRITE_ET200S_CP

Explanation of the formal parameters of PE_DS3_WRITE_ET200_CP

Parameters	Declaration	Data type	Range of values	Description
CPLADDR	INPUT	WORD	I, Q, M, D, L, const.	Module start address of the CP
ENABLE	INPUT	BOOL		Starts the processing of the program block on a rising edge.
ID	INPUT	WORD		Logical address of the header slot of the IO device
SLOT_NO_1	INPUT	INT		Slot number of the first power module
FUNC_1	INPUT	INT		Specifies the switching behavior for the power module in terms of starting or ending energy-saving pauses • 0 (FALSE) – PAUSE_START: No influence (power module remains turned on) – PAUSE_STOP: Turns the power module on again. • 1 (TRUE) – PAUSE_START: Turns the power module off. – PAUSE_STOP: Turns the power module on again.
...	INPUT	INT		
...	INPUT	INT		
SLOT_NO_8	INPUT	INT		Slot number of the eighth power module
FUNC_8	INPUT	INT		See "FUNC_1"

Parameters	Declaration	Data type	Range of values	Description
BUSY	OUTPUT	BOOL	0: Execution not yet started, completed or aborted 1: Execution active	Condition code of the processing status of the program block
DONE	OUTPUT	BOOL	0: - 1: Data record transferred successfully	This parameter indicates whether or not the job was completed without errors.
ERROR	OUTPUT	BOOL	0: - 1: Errors	Error code
STATUS	OUTPUT	WORD		Status code

For the meaning of DONE, ERROR and STATUS, see Condition codes of PE_DS3_WRITE_ET200S_CP (Page 7809).

Condition codes of PE_DS3_WRITE_ET200S_CP

Condition codes of PE_DS3_WRITE_ET200_CP

PE_DS3_WRITE_ET200_CP is based on the program block PNIO_RW_REC and returns all condition codes of PNIO_RW_REC, see condition codes of the block PNIO_RW_REC.

Instructions for programmed connections

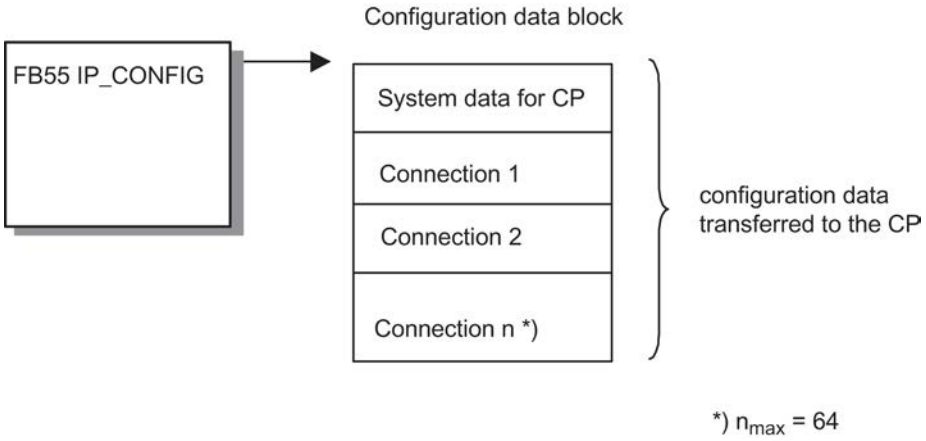
Overview

Interplay between programming and configuration

You configure connections on the SEND/RECEIVE interface and the IP configuration of a CP either with STEP 7 or you configure them during runtime of the S7 station via the user program. Mixing these variants on a CP is not possible!

Principle of programmed configuration

Configuration data for communication connections and the IP configuration can be transferred to the CPU using an instruction called in the user program.



The configuration DB can be loaded on the CP at any time. The previously applicable connections and configuration data (IP address, subnet mask, default router, NTP time server and other parameters) are overwritten.

Based on the configuration data, the Ethernet CP recognizes that the communication connections must be set up by the user program.

Note

The functions can only be executed if "Not locked" was configured for the module access protection. Refer to the "Options" tab in the properties dialog of the CP (not available for every CP).

The "Set IP address in user program" option must also be enabled (see properties dialog of the CP or the Ethernet interface of the CP, "IP Configuration" tab).

Based on the configuration data, the Ethernet CP recognizes that the communication connections must be set up by the user program.

Note

As soon as the user program transfers the connection data via the IP_CONFIG instruction, the CPU switches the CP briefly to STOP. The CP receives the system data (including the IP address) and the new connection data and processes them during startup (RUN).

Quantity framework

A maximum of 64 connections can be specified in the CP_CONFIG instruction. The most important factor, however, is the maximum number of connections supported by the CP type you are using.

Special features / restrictions

- Consistency check only in the configuration
  The connection configuration involves consistency checks that are not possible or only possible with restrictions when using the programmed configuration!
- Connection configuration required on the partner
  When configuring specified connections, you implicitly create the connection for the partner; with a programmed configuration, this is not possible! In this case, you must configure suitable connections for the partner.
- Configuring IP access protection
  Using IP access protection gives you the opportunity of restricting communication over the CP of the local S7 station to partners with specific IP addresses. This parameter assignment also applies to programmed communications connections. You either disable IP access protection in the configuration (= default) or authorize the communications partner.
- DHCP / DNS is supported
  With a programmed configuration, IP addressing is also possible using DHCP (and DNS for the mail service).
  The use of a DHCP server is defined in this case in the IP_CONFIG instruction (not in the configuration).
- No connection information when uploading
  When you upload the S7 station data into the project engineering, this does not contain the data of the programmed configuration.
- Configuring connections for CPs with several interfaces
  If you are using CPs with several interfaces (for example with a gigabit interface), check the device manual to see whether or not the connection configuration is supported for both interfaces.
- PROFINET IO is not possible at the same time
  On a device you intend to operate as a PROFINET IO controller or IO device, it is not possible to set up the connection using FB55 as described here.
- No use of the IP_CONFIG instruction when operating the CP with fault-tolerant S7 connections
  If you configure fault-tolerant S7 connections via the CP, you cannot use the IP_CONFIG instruction for IP configuration of the CP.

See also

Configuration data block (Page 7817)

IP_CONFIG for programmed communications connections (Page 7812)

IP_CONFIG

IP_CONFIG for programmed communications connections

Description

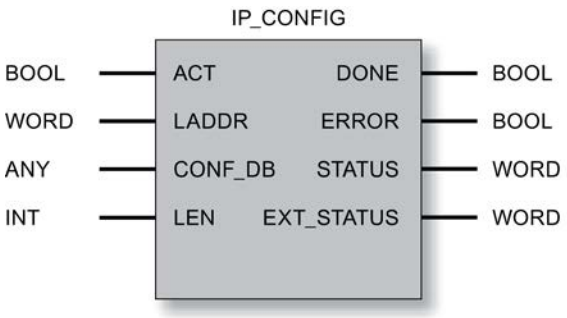
The IP_CONFIG instruction transfers the connection data specified in a data block (configuration DB) to the CP. The configuration DB contains all the connection data to allow the connections for the SEND/RECEIVE interface of an Ethernet CP to be set up.

You can use this variant of programmed communication connections as an alternative to connection configuration with STEP 7.

Depending on the size of the configuration DB, the data may be transferred to the CP in several segments. This means that the instruction must continue to be called until it signals complete transfer by setting the DONE bit to 1.

Call

Call interface in FBD representation



Note

No use of the IP_CONFIG instruction when operating the CP with fault-tolerant S7 connections

If you configure fault-tolerant S7 connections via the CP, you cannot use the IP_CONFIG instruction for IP configuration of the CP.

Note

Avoid possible double addressing

If you use the IP_CONFIG instruction, make sure that the assignment of IP addresses is unique. If an address is detected twice, it is possible that the CP will not become active in the network.

See also

Overview (Page 7809)
Configuration data block (Page 7817)
Parameters for IP_CONFIG (Page 7813)
Reserved port numbers - IP_CONFIG (Page 7814)
Parameters DONE, ERROR, STATUS (Page 7815)

Parameters for IP_CONFIG

Explanation of the formal parameters

The following table explains the formal parameters for the call interface of the "IP_CONFIG" instruction:

Parameter	Declaration	Data type	Range of values	Description
ACT	INPUT	BOOL, TIMER, COUNTER	0,1	When the instruction is called with ACT = 1, DBxx is sent to the CP. If the instruction is called with ACT = 0, only the status codes DONE, ERROR and STATUS are updated.
LADDR	INPUT	WORD		Module start address When you configure the CP with STEP 7, the module start address is displayed in the configuration table. Specify this address here.
CONF_DB	INPUT	ANY		The parameter points to the start address of the configuration data area in a data block (data type: byte).
LEN	INPUT	INT		Length information in bytes for the configuration data area.
DONE	OUTPUT	BOOL	0: - 1: Job completed with data transfer.	The parameter indicates whether the configuration data area was completely transferred. Remember that it may be necessary to call the instruction several times depending on the size of the configuration data area (in several cycles) until the DONE parameter is set to 1 to signal completion of the transfer. For the meaning of this parameter in conjunction with the ERROR and STATUS parameters, refer to the following table.
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning of this parameter in conjunction with the DONE and STATUS parameters, refer to the following table.

Parameter	Declaration	Data type	Range of values	Description
STATUS	OUTPUT	WORD	See following table	Status code For the meaning of this parameter in conjunction with the DONE and ERROR parameters, refer to the following table.
EXT_Status	OUTPUT	WORD		If an error occurs in the execution of a job, the parameter indicates which parameter was detected as the cause of the error in the configuration DB. High byte: Index of the parameter field Low byte: Index of the subfield within the parameter field

See also

IP_CONFIG for programmed communications connections (Page 7812)

Reserved port numbers - IP_CONFIG

Reserved Port Numbers

The following local port numbers are reserved; do not use these in the connection project engineering.

Table 9-353 Reserved Port Numbers

Protocol	Port number	Service
TCP	20, 21	FTP
TCP	25	SMTP
TCP	80	HTTP
TCP	102	RFC1006
TCP	135	RPC-DCOM
TCP	502	ASA application protocol
UDP	161	SNMP_REQUEST
UDP	34964	PN IO
UDP	65532	NTP
UDP	65533	NTP
UDP	65534	NTP
UDP	65535	NTP

See also

IP_CONFIG for programmed communications connections (Page 7812)

Parameters DONE, ERROR, STATUS

Condition codes

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

Table 9-354 IP_CONFIG status codes

DONE	ERROR	STATUS	Meaning
General codes relating to job execution			
1	0	0000_H	Job completed without errors
0	0	8181_H	Job active
Errors detected on the interface between CPU and CP.			
0	1	$80A4_H$	• Communication error on the K-bus or • Data error: Configuration by the user program is not set.
0	1	$80B1_H$	The number of data bytes to be sent exceeds the upper limit for this service. (upper limit = 16 Kbytes)
0	1	$80C4_H$	Communication error The error can occur temporarily; it is usually best to repeat the job in the user program.
0	1	$80D2_H$	Configuration error The module you are using does not support this service.
Errors detected in the evaluation of the IP_CONFIG in the CPU or on the interface between CPU and CP.			
0	1	8183_H	The CP rejects the requested data record number.
0	1	8184_H	System error or bad parameter type. (data type of the ANY pointer CONF_DB not OK) (Currently only the byte data type is accepted)
0	1	8185_H	The value of the LEN parameter is larger than the CONF_DB less the reserved header (4 bytes) or the length information is incorrect.
0	1	8186_H	Illegal parameter detected The ANY pointer CONF_DB does not point to a data block.
0	1	8187_H	Illegal status of the IP_CONFIG Data in the header of CONF_DB was possibly overwritten.
Further errors detected on the interface between the CPU and CP.			
0	1	$8A01_H$	The status code in the data record is invalid (value is >= 3).
0	1	$8A02_H$	There is no job running on the CP. IP_CONFIG did, however, expect an acknowledgment for the executed job.
0	1	$8A03_H$	There is no job running on the CP and the CP is not ready. IP_CONFIG triggered the first job to read a data record.
0	1	$8A04_H$	There is no job running on the CP and the CP is not ready. IP_CONFIG did, however, expect an acknowledgment for the executed job.
0	1	$8A05_H$	A job is running, there has, however, not been an acknowledgment yet. IP_CONFIG, however, triggered the first job to read a data record.
0	1	$8A06_H$	A job is completed. IP_CONFIG, however, triggered the first job to read a data record.
Errors detected when evaluating IP_CONFIG on the CP.			
0	1	$8B01_H$	Communication error The DB could not be transferred.

DONE	ERROR	STATUS	Meaning
0	1	$8B02_H$	Parameter error Double parameter field
0	1	$8B03_H$	Parameter error The subfield in the parameter field is not permitted.
0	1	$8B04_H$	Parameter error The length specified in the instruction does not match the length of the parameter fields / subfields.
0	1	$8B05_H$	Parameter error The length of the parameter field is invalid.
0	1	$8B06_H$	Parameter error The length of the subfield is invalid.
0	1	$8B07_H$	Parameter error The ID of the parameter field is invalid
0	1	$8B08_H$	Parameter error The ID of the subfield is invalid
0	1	$8B09_H$	System error The connection does not exist
0	1	$8B0A_H$	Data error The content of the subfield is not correct.
0	1	$8B0B_H$	Structure error A subfield exists twice.
0	1	$8B0C_H$	Data error The parameter does not contain all the necessary parameters.
0	1	$8B0D_H$	Data error The CONF_DB does not contain a parameter field for system data.
0	1	$8B0E_H$	Data error / structure error The CONF_DB type is invalid.
0	1	$8B0F_H$	System error The CP does not have enough resources to process CONF_DB completely.
0	1	$8B10_H$	Data error Configuration by the user program is not set.
0	1	$8B11_H$	Data error The specified type of the parameter field is invalid.
0	1	$8B12_H$	Data error Too many connections were specified (either in total or too many for a specific type; for example, only one Email connection is possible).
0	1	$8B13_H$	CPinternal error
0	1	$8B14_H$	The active protection level does not permit the action of making changes.
Further errors detected on the program interfaces within the CPU (system instruction errors).			
0	1	$8F22_H$	Area length error reading a parameter (e.g. DB too short).
0	1	$8F23_H$	Area length error writing a parameter (e.g. DB too short).
0	1	$8F24_H$	Area error reading a parameter.
0	1	$8F25_H$	Area error writing a parameter.

DONE	ERROR	STATUS	Meaning
0	1	$8F28_H$	Alignment error reading a parameter.
0	1	$8F29_H$	Alignment error writing a parameter.
0	1	$8F30_H$	The parameter is in the writeprotected first current data block.
0	1	$8F31_H$	The parameter is in the writeprotected second current data block.
0	1	$8F32_H$	The parameter contains a DB number that is too high.
0	1	$8F33_H$	DB number error
0	1	$8F3A_H$	The target area was not loaded (DB).
0	1	$8F42_H$	Timeout reading a parameter from the I/O area.
0	1	$8F43_H$	Timeout writing a parameter to the I/O area.
0	1	$8F44_H$	Access to a parameter to be read during execution of the instruction is prevented.
0	1	$8F45_H$	Access to a parameter to be written during execution of the instruction is prevented.
0	1	$8F7F_H$	Internal error For example, an illegal ANY reference was detected.

See also

IP_CONFIG for programmed communications connections (Page 7812)

Configuration data block

Configuration data block (CONF_DB)

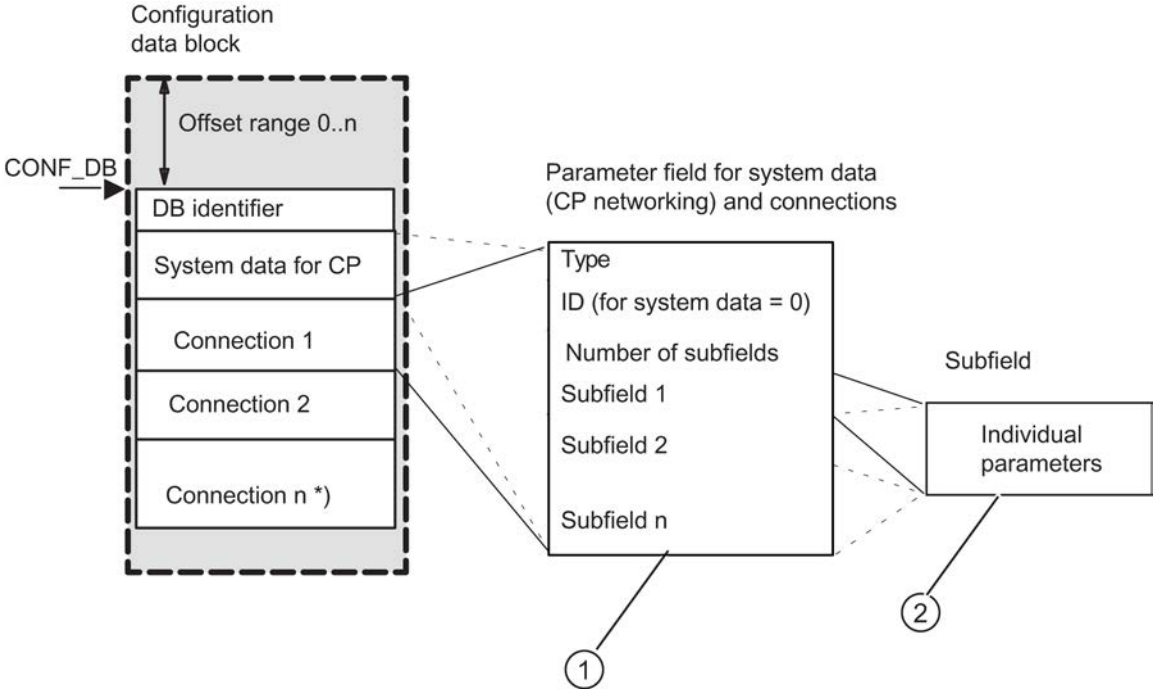
Meaning

The configuration data block (CONF_DB) contains all the connection data and configuration data (IP address, subnet mask, default router, NTP time server and other parameters) for an Ethernet CP. The configuration data block is transferred to the CP with the IP_CONFIG instruction.

Block and Data Structure

The graphic below shows the following:

- Structure resulting from parameter fields and subfields
  - The connections and specific system data are is described by an identically structured parameter field.
  - Individual parameters are characterized by subfields.
- Offset range
  The CONF_DB can start at any point within a data block as specified by an offset range. The address (or offset) must simply be an even number.



① Parameter fields are described below in Parameter field for system data (IP configuration) (Page 7818)

② Subfield types are described below in Parameter fields for connection types (Page 7822)

Parameter field for system data (IP configuration)

Meaning

Below, you will find the parameter field for system data relevant to the IP configuration of the CP and the subfields that need to be specified in it.

Some applications do not require all the subfield types - refer to the table for details.

Layout

On CPs with several interfaces, the structure of the parameter field described below applies only to the PROFINET interface.

• **Type = 0**
• **ID = 0**
• Number of subfields = n
• Subfield 1
• Subfield 2
• Subfield n ...

Usable subfields

Subfield		Parameter	
ID	Type	Special features / notes	Application ***)
1	SUB_IP_V4	Local IP address	++
2	SUBNET_MASK	-	++
8	SUB_DEF_ROUTER	-	+
4	SUB_DNS_SERV_ADDR *)	This subfield can occur from 0 to 4 times. The first entry is the primary DNS server.	+
14	SUB_DHCP_ENABLE	0: No DHCP 1: DHCP	+
15	SUB_CLIENT_ID	- Note: Only useful when SUB_DHCP_ENABLE = 1	+
30**)	SUB_DEVICE_NAME	Device name complying with PROFINET IO convention Enter a device name to make the device individually recognizable for analysis and diagnostics in the network.	+
*)The subfield type is used only for E-mail connections. **) ID is supported only be certain CP types. ***) ++ = mandatory; + = optional			

See also

Subfield types (Page 7819)

Subfield types

Different parameters are required depending on the parameter field. Each parameter is described by a subfield. Which subfields are required is explained in the descriptions of the system data and the connection types in the previous sections.

Each subfield consists of the specific parameter section and the header (4 Byte).

Example

The following excerpt from a CONF_DB illustrates the structure of a subfield based on the example of the SUBNET_MASK subfield type.

Address	Name	Type	Initial value	Comment
+14.0	Sub_field_2	STRUCT		// Subfield 2 type SUBNET_MASK
+0.0	Sub_field_ID	INT	2	// Subfield ID
+2.0	Sub_field_len	INT	8	// Total length of the subfield in bytes
+4.0	Parameter	STRUCT		Parameter range of SUBNET_MASK
+0.0	Value_1	BYTE	B#16#FF	
+1.0	Value_2	BYTE	B#16#FF	
+2.0	Value_3	BYTE	B#16#FF	
+3.0	Value_4	BYTE	B#16#0	
=4.0	END_	STRUCT		
=8.0	END_	STRUCT		

In total, the following subfield types are available:

Subfield ID [1)]	Subfield type	Subfield length (in bytes)	Meaning of the Parameter
1	SUB_IP_V4	4 + 4	IP address according to IPv4
2	SUBNET_MASK	4 + 4	Subnet mask
3	SUB_DNS_NAME	Length of DNS name + 4	DNS name
4	SUB_DNS_SERV_ADDR	4 + 4	DNS server address
8	SUB_DEF_ROUTER	4 + 4	IP address of default router
9	SUB_LOC_PORT	2 + 4	Local port
10	SUB_REM_PORT	2 + 4	Remote port, also for Email connections
11	SUB_LOC_TSAP	TSAP length + 4	Local TSAP
12	SUB_REM_TSAP	TSAP length + 4	Remote TSAP
13	SUB_EMAIL_SENDER	Length of the sender E-mail address + 4	Email address of the sender
14	SUB_DHCP_ENABLE	2 + 4	Obtain an IP address from a DHCP server • Range of values: 0 = no DHCP 1 = DHCP (optional)
15	SUB_CLIENT_ID	Length of the client ID + 4	(optional)
18	SUB_CONNECT_NAME	Length of the name + 4	Name of the connection Possible characters are: a...z, A...Z, 0...9, -, _

Subfield ID [1)]	Subfield type	Subfield length (in bytes)	Meaning of the Parameter
19	SUB_LOC_MODE	1 + 4	Local mode of the connection • Range of values: 0x00 = SEND/RECV 0x01 = FTP protocol (TCP connection only) 0x10 = S5 addressing mode for FETCH/ WRITE *) 0x20 = SPEED SEND/RECV (permitted only for CP 443-1 Advanced) 0x80 = FETCH *) 0x40 = WRITE *) If you do not set the parameter, the default setting is SEND/RECV. Note: FETCH / WRITE require the passive connection establishment setting (see SUB_CON_ESTABL.
20	SUB_REM_MODE	1 + 4	Setting the mode on the communication partner. (not currently supported)
21	SUB_KBUS_ADR	5	KBUS address of the CPU (relevant only for S7-400)
22	SUB_CON_ESTABL	1 + 4	Type of connection establishment. With this option, you specify whether connection establishment from this S7 station is active or passive. • Range of values: 0 = passive 1 = active
23	SUB_ADDR_IN_DATA-BLOCK	1 + 4	Select free UDP connection. The remote node is entered in the job header of the job buffer by the user program when it calls AG_SEND. This allows any node on Ethernet/LAN/WAN to be reached. • Range of values: 1 = free UDP connection 0 = other The parameter is practical only for a UDP connection.
24	SUB_NTP_SERVER	4 + 4	The subfield defines an NTP server from which the CP can obtain its time via the NTP protocol. For the situation when one or more NTP servers are defined, up to 4 subfields of ID 24 can be defined. The subfields of ID 24 may only be installed in the system parameter field type 0 / ID 0.

Subfield ID [1]	Subfield type	Subfield length (in bytes)	Meaning of the Parameter
30	SUB_DEVICE_NAME	Length of the name + 4	Device name complying with PROFINET IO convention The device name must comply with DNS conventions, in other words; • Restriction to a total of 127 characters (letters, numbers, hyphen or period) • Parts of the name within the device name; in other words, a string between two periods, must not exceed a maximum of 63 characters. • No special characters such as umlauts (ä, ö etc.), brackets, underscore, slash, blank etc. The dash (hyphen) is the only permitted special character. • The device name must not begin or end with the "-" or "." character, nor may either of these be the last character. • The device name must not begin with numbers. • The device name must not have the format n.n.n.n (n = 0...999). • The device name must not begin with the character string "port-xyz-" (x, y, z = 0...9).
1) Note: ID numbers not listed are not currently used.			

Parameter fields for connection types

General

Below, you will see which values need to be entered in the parameter fields and which subfields are used for the various connection types.

Some applications do not require all the subfield types - refer once again to the table for details.

Connection ID

The ID parameter that precedes each connection parameter field beside the type ID is particularly important.

On programmed connections, you can assign this ID freely within the permitted range of values. You must then use this ID on the call interface of the FCs for the SEND/RECV interface to identify the connection.

Range of values for the connection ID:

- S7-400: 1,2...64
- S7-300: 1,2...16

See also

Subfield types (Page 7819)

Parameter field for TCP connection

Layout

Enter the parameters in the parameter field for TCP connections as follows:

• **Type = 1 -> ①**
• **ID = connection ID -> ②**
• Number of subfields = n
• Subfield 1
• Subfield 2
• Subfield n ...

Legend:
① Identifier for the connection type
② Freely selectable connection reference; must be specified in AG_SEND / AG_RECV.
Range of values for the connection ID:
for S7-400: 1, 2...64
for S7-300: 1,2...16

Usable subfields

Subfield		Parameter	
ID	Type	Special features / notes	Application ***)
1	SUB_IP_V4	IP address of the partner	++ *)
9	SUB_LOC_PORT	-	++
10	SUB_REM_PORT	-	++ **)
18	SUB_CONNECT_NAME	-	+
19	SUB_LOC_MODE	-	+
21	SUB_KBUS_ADR	This value is always set to 2 for CPs for the S7300 and does not need to be specified.	++ (for S7-400)
22	SUB_CON_ESTABL	-	++
*) optional for a passive connection. ***) ++ = mandatory; + = optional			

See also

Subfield types (Page 7819)

Parameter field for UDP connection

Layout

Enter the parameters in the parameter field for UDP connections as follows:

• **Type = 2 -> ①**
• **ID = connection ID -> ②**
• Number of subfields = n
• Subfield 1
• Subfield 2
• Subfield n ...

Legend:
① Identifier for the connection type
② Freely selectable connection reference; must be specified in AG_SEND / AG_RECV.
Range of values for the connection ID:
for S7-400: 1, 2...64
for S7-300: 1,2...16

Usable subfields

Subfield		Parameter	
ID	Type	Special features / notes	Application ***)
1	SUB_IP_V4	IP address of the partner	++
9	SUB_LOC_PORT	-	++
10	SUB_REM_PORT	-	++
18	SUB_CONNECT_NAME	-	+
19	SUB_LOC_MODE	-	+
21	SUB_KBUS_ADR	This value is always set to 2 for CPs for the S7300 and does not need to be specified.	++ (for S7-400)
23	SUB_ADDR_IN_DATABLOCK	If the "Free UDP connection" is selected for this parameter, the parameters SUB_IP_V4 and SUB_REM_PORT are omitted.	+
***) ++ = mandatory; + = optional			

See also

Subfield types (Page 7819)

Parameter field for an ISOonTCP connection

Layout

Enter the parameters in the parameter field for ISO-on-TCP connections as follows:

• **Type = 3 -> ①**
• **ID = connection ID -> ②**
• Number of subfields = n
• Subfield 1
• Subfield 2
• Subfield n ...

Legend:
① Identifier for the connection type
② Freely selectable connection reference; must be specified in AG_SEND / AG_RECV.
Range of values for the connection ID:
for S7-400: 1, 2...64
for S7-300: 1,2...16

Usable subfields

Subfield		Parameter	
ID	Type	Special features / notes	Application ***)
1	SUB_IP_V4	IP address of the partner	++ *)
11	SUB_LOC_TSAP	-	++
12	SUB_REM_TSAP	-	++ *)
18	SUB_CONNECT_NAME	-	+
19	SUB_LOC_MODE	-	+
21	SUB_KBUS_ADR	This value is always set to 2 for CPs for the S7300 and does not need to be specified.	++ (for S7-400)
22	SUB_CON_ESTABL	-	++
*) optional for a passive connection. ***) ++ = mandatory; + = optional			

See also

Subfield types (Page 7819)

Parameter field for an Email connection

Meaning

To send Emails, one Email connection must be set up per Advanced CP. The Email connection specifies the mail server via which all the mails sent by the Advanced CP are delivered.

Layout

Enter the parameters in the parameter field for E-mail connections as follows:

• **Type = 4 -> ①**
• **ID = connection ID -> ②**
• Number of subfields = n
• Subfield 1
• Subfield 2
• Subfield n ...

Legend:
① Identifier for the connection type
② Freely selectable connection reference; must be specified in AG_SEND / AG_RECV.
Range of values for the connection ID:
for S7-400: 1, 2...64
for S7-300: 1,2...16

Usable subfields

Subfield		Parameter	
ID	Type	Special features / notes	Application ***)
1	SUB_IP_V4	IP address of the mail server, over which the Emails are sent. You can specify an absolute or alias IP address. The use of an alias assumes that the Advanced CP knows the address of the domain name server (DNS). This entry must be made when configuring the Advanced CP in HW Config. For more detailed information refer to the online help.	++ / + *)
3	SUB_DNS_NAME	DNS name of the Email server	++ / + *)
13	SUB_EMAIL_SENDER	Email address of the sender	++
18	SUB_CONNECT_NAME	-	+
21	SUB_KBUS_ADR	This value is always set to 0 for CPs for the S7300 and does not need to be specified.	++ (for S7-400)

Subfield		Parameter	
ID	Type	Special features / notes	Application ***)
22	SUB_CON_ESTABL	-	++
) The parameters SUB_IP_V4 and SUB_DNS_NAME are mutually exclusive; one or the other parameter must be specified. *) ++ = mandatory; + = optional			

Note

Mail server ports are "wellknown ports" and do not need to be specified.

See also

Subfield types (Page 7819)

Parameter field for FTP connection

Meaning

To run an FTP job sequence between the S7 station acting as the FTP client and an FTP server, the Advanced CP must establish a connection to the S7 CPU. This connection is known as an FTP connection.

FTP connections are TCP connections, with the parameter SUB_LOC_MODE set to the "FTP" mode.

Layout

Enter the parameters in the parameter field for FTP connections as follows:

• **Type = 1 -> ①**
• **ID = connection ID -> ②**
• Number of subfields = n
• Subfield 1
• Subfield 2
• Subfield n ...

Legend:

① Identifier for the connection type

② Freely selectable connection reference; must be specified in AG_SEND / AG_RECV.
Range of values for the connection ID:
for S7-400: 1, 2...64
for S7-300: 1,2...16

Usable subfields

Subfield		Parameter	
ID	Type	Special features / notes	Application ***)
18	SUB_CONNECT_NAME	-	+
19	SUB_LOC_MODE	here: 0x01 = FTP protocol	++
21	SUB_KBUS_ADR	This value is always set to 0 for CPs for the S7300 and does not need to be specified.	++ (for S7-400)
***) ++ = mandatory; + = optional			

See also

Subfield types (Page 7819)

Instructions for FTP services

FTP_CMD for FTP services

Migration from FC 40-44 to FB40

Comparison of the FTP_CMD program block with older functions FC40...44

All CPs with FTP functionality support the functions FC40...44. This means that existing user programs can be used unchanged.

If you want to convert from the FTP functions FC40...44 to FTP_CMD, you will need to modify your user program.

The following table shows the FTP_CMD commands used to replace the functions FC40...44.

- Correlation is indicated by "X".
- Where there is no correlation, this is indicated by "-".

	Commands of the "CMD" parameter of FTP_CMD					
Older FTP functions FC40...44	CMD = 1	CMD = 2	CMD = 3	CMD = 4	CMD = 5	CMD = 6 CMD = 7 CMD = 17
FC40	X [1)]					
FC41		X [2)]				
FC42			X [3)]			
FC43				X [4)]		
FC44					X	
						-
[1)]...[4)] The parameters of FC40...43 and CMD 1...4 (FTP_CMD) are not identical. (See table below)						

The corresponding parameters that specify a particular function in the functions FC40...FC43 or in the commands of FTP_CMD are listed in the following table.

Parameters of the FC			Parameters in FTP_CMD (with CMD 1...4)	
FC40:	LOGIN	→	CMD = 1:	NAME_STR
FC41:	FILE_NAME	→	CMD = 2:	NAME_STR
FC42:	FILE_NAME	→	CMD = 3:	NAME_STR
FC43:	FILE_NAME	→	CMD = 4:	NAME_STR
FC40...43:	BUFFER_DB_NR	→	Omitted (replaced by instance DB)	

See also

Overview of FTP_CMD (Page 7829)

Overview of FTP_CMD

Meaning

Using the FTP_CMD instruction, you can establish FTP connections and transfer files from and to an FTP server.

The advantages of FTP_CMD are as follows:

- Simplification in the user program by using a command variable instead of different function calls
- Additional function "APPEND"
  "APPEND" allows data to be appended to an existing file.
- Additional function "RETR_PART"
  "RETR_PART" allows selected data areas to be retrieved from a file.
- Additional function "CONNECT_TLS_PRIVATE"
  "CONNECT_TLS_PRIVATE" allows the establishment of secure SSL connections
- The AG_SEND (FC5) function is not required here.

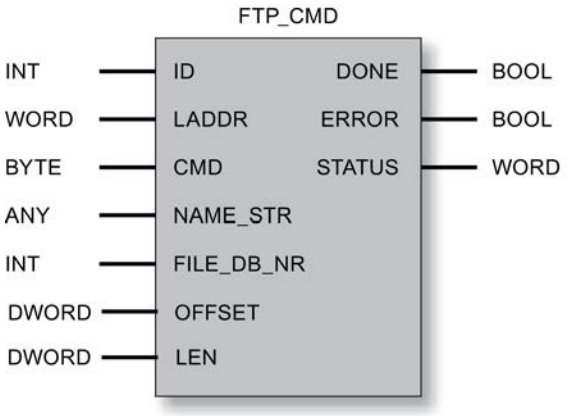
Validity

FB40 can be used as of the following module types:

- As of CP 343-1 Advanced (GX30* / GX31)
- As of CP 443-1 Advanced (GX20* / GX30)
  *) "CONNECT-TLS-PRIVATE" function cannot be used.

Call interface

Call interface in FBD representation



System functions called

The following system functions are called by the program block FTP_CMD:

SFC 1, SFC 20, SFC 24, SFC 58, SFC 59

Note

Note that the FTP client services of old SIMATIC S7-300 CPUs, for example the CPU 312 or CPU 315-1AF01, cannot be used because they do not support SFC24.

See also

Structure and use of the file DB (Page 7837)

Output parameters and status information FTP_CMD (Page 7834)

Input parameter - FTP_CMD (Page 7831)

Migration from FC 40-44 to FB40 (Page 7828)

Input parameter - FTP_CMD

Explanation of the input parameters

Each FTP block call must be supplied with the following input parameters:

Table 9-355 Formal parameters of FB40 (FTP_CMD) - input parameters

Parameter	Declaration	Type	Range of values	Meaning / remarks
ID	INPUT	INT	For S7-300: 1, 2...16 For S7-400: 1, 2...64	The FTP jobs are handled on FTP connections. The parameter identifies the connection being used.
LADDR	INPUT	WORD		Module start address When you call an FC, you transfer the module start address of the ADVANCED-CP in the LADDR parameter. You will find the module start address of the ADVANCED CP in the configuration of the ADVANCED CP in "Properties>Addresses>Inputs".
CMD	INPUT	BYTE	See table below - FTP commands in the "CMD" parameter	FTP commands executed when FTP_CMD is called. You will find further information following the table. If a command is not supported by the CP firmware, an error message with STATUS = $8F6B_H$ is output. Examples of FTP commands: • RETRIEVE: B#16#3 • CONNECT_TLS_PRIVATE: B#16#11
NAME_STR	INPUT	ANY	Only "BYTE" is permitted as VARTYPE.	The address references a data block area. Here, you specify the address and length of the data area in which the target data is entered. • When CMD = 1, 17: With this command, the "NAME_STR" parameter specifies the FTP server to be addressed over the FTP connection with the following attributes: - IP address of the FTP server - User name - Password for the login These values must be specified as three consecutive strings in the destination range of the ANY pointer. • When CMD = 2, 3, 4, 6, 7: With this command, the "NAME_STR" parameter specifies the file name on the FTP server, in other words, the data source or data destination. The file name is specified as a string in the destination range of the ANY pointer. • When CMD = 5: Parameter not relevant You will find example of content further below.

Parameter	Declaration	Type	Range of values	Meaning / remarks
FILE_DB_NR	INPUT	INT		The data block specified here contains the file DB to be read / written. The parameter is relevant only when CMD = 2, 3, 6 and 7.
OFFSET	INPUT	DWORD		Only when CMD = 7: Offset in bytes starting at which the file will be read.
LEN	INPUT	DWORD		Only when CMD = 7: Sublength in bytes that is read starting at the value specified in "OFFSET". Special features: • If "DW#16#FFFFFFFF" is specified, the available rest of the file will be read. Result OK (DONE = 1, STATUS = 0) if no other error occurred. • When OFFSET > length of the original file: The length of the destination file is displayed in this case in the ACT_LENGTH parameter in the file DB: 0 bytes on the CPU. Result OK (DONE = 1, STATUS = 0) if no other error occurred. • When OFFSET + LEN > length of the original file (and LEN ≠ 0xFFFFFFFF): The length of the destination file is displayed in this case in the ACT_LENGTH parameter in the file DB: Available bytes starting at "OFFSET". Result OK (DONE = 1, STATUS = 0) if no other error occurred.

FTP commands in the "CMD" parameter

The following table explains the meaning of the commands of the "CMD" parameter and which input parameters need to be supplied. The ID and LADDR parameters must always be set to identify the connection.

CMD	Relevant input parameters (in addition to ID and LADDR)	Meaning / handling
0 (NOOP)	-	The called instruction does not execute any action. The status codes are set as follows when these parameters are supplied: • DONE=1; ERROR=0; STATUS=0
1 (CONNECT)	NAME_STR	With this command, the FTP client establishes an FTP connection to an FTP server (port 21). The connection is available under the connection ID specified here for all further FTP commands. Data is then exchanged with the FTP server specified for this user.
2 (STORE)	NAME_STR FILE_DB_NR	This function call transfers a data block (file DB) from the FTP client (S7-CPU) to the FTP server. Caution: If the file (file DB) already exists on the FTP server, it will be overwritten.

CMD	Relevant input parameters (in addition to ID and LADDR)	Meaning / handling
3 (RETRIEVE)	NAME_STR FILE_DB_NR	This function call transfers a file from the FTP server to the FTP client (S7-CPU). Caution: If the data block (file DB) on the FTP client already contains a file, it will be overwritten.
4 (DELETE)	NAME_STR	With this function call, you delete a file on the FTP server.
5 (QUIT)	No others	With this function call, you establish the FTP connections selected with the ID.
6 (APPEND)	NAME_STR FILE_DB_NR	Similar to "STORE", the "APPEND" command saves a file on the FTP server. With "APPEND", the file on the FTP server is, however, not overwritten. The new content is appended to the existing file. If the file (file DB) does not exist on the FTP server, it will be created.
7 (RETR_PART)	NAME_STR FILE_DB_NR OFFSET LEN	Using the "RETR_PART" command (retrieve part) , you can request a section of a file from the FTP server. If very large files are involved, this allows you to restrict the read to the part you currently require. To do this, you need to know the structure of the file. Enter the required part of the file using the two parameters "OFFSET" and "LEN" in FB40.
17 (CONNECT_TLS_PRIVATE)	NAME_STR	With the "CONNECT_TLS_PRIVATE" command, the FTP client sets up an SSL-secured FTP connection (FTPS) to an FTP server (port 21). The data of the control connection and the data connection is secure. The connection is available under the connection ID specified here for all further FTP commands. Data is then exchanged with the FTP server specified for this user. Requirements: • The "Security" option must be activated on the CP. • For the FTP connection secured with SSL, certificates must be provides in the CP configuration.

Examples of the content of the "NAME_STR" parameter

The parameter record has the following content:

Table 9-356 Content of the parameter record for CMD = 1, 17

Relative address 2)	Name	Type 1)	Example	Meaning
0.0	ip_address	STRING[100]	’142.11.25.135’	IP address of the FTP server
102.0	username	STRING[32]	'user'	User name for the login on the FTP server
136.0	password	STRING[32]	'password'	Password for the login on the FTP server
1) The maximum possible string length is specified 2) The specified values relate to the string lengths specified in "Type".				

Table 9-357 Content of the parameter record for CMD = 2, 3, 4, 6, 7

Relative address 2)	Name	Type 1)	Example	Meaning
170.0	filename	STRING[220] or STRING[212]	'plant1/tank2/press.dat'	File name of the source or destination file Note: If CMD=7 (RETR_PART), the maximum length of the file name is limited to 212 characters.
1) The maximum possible string length is specified 2) The specified values relate to the string lengths specified in "Type".				

See also

Overview of FTP_CMD (Page 7829)

Output parameters and status information FTP_CMD

Introduction

For status evaluation, the following parameters must be evaluated in the user program:

Table 9-358 Formal parameters of FB40 (FTP_CMD) - output parameters

Parameter	Declaration	Type	Range of values	Meaning / remarks
DONE	OUTPUT	BOOL	0: - 1: Job executed	This parameter indicates whether or not the job was completed without errors.
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code This parameter signals that the job could not be executed error-free.
STATUS	OUTPUT	WORD	See following table	Status code This parameter supplies detailed information about the execution of the job.

The DONE, ERROR and STATUS parameters are updated at every block call.

Example

During job execution, FTP_CMD returns the following codes:

- DONE=0
- ERROR=0
- STATUS=8181_H

Meaning: Job still running.

Evaluating status codes

Note

For entries coded with 8FxxH in STATUS, refer to the information in the STEP 7 Standard and System Functions reference manual. The chapter describing error evaluation with the RET_VAL output parameter contains detailed information.

Table 9-359 FB 40: Meaning of the STATUS parameter in conjunction with DONE and ERROR

DONE	ERROR	STATUS	Meaning
0	0	0000_H	No job being executed.
1	0	0000_H	Job completed without error.
0	0	8181_H	Job active. If 8181_H is indicated permanently: The CP is not released for FTP_CMD (an illegal command for the firmware version (CMD 6, CMD 7 or CMD 17) was called.)
0	1	8090_H	• No module with this module start address exists. • The block being used does not match the system family being used (remember to use different blocks for S7300 and S7400).
0	1	8091_H	Module start address not at a doubleword boundary
0	1	8092_H	Type information in the ANY pointer is not byte
0	1	$80A4_H$	The communication bus connection between the CPU and CP is not established (with newer CPU versions). This can, for example, be caused by the following: • No connection configuration • Maximum number of CPs operating at the same time was exceeded
0	1	$80B0_H$	The module does not recognize the data record.
0	1	$80B1_H$	Destination area invalid; for example, destination area > 240 bytes.
0	1	$80B2_H$	The communication bus connection between the CPU and CP is not established (with older CPU versions). (with newer CPU versions, see $80A4_H$)
0	1	$80C0_H$	The data record cannot be read.
0	1	$80C1_H$	The specified data record is currently being processed.
0	1	$80C2_H$	There are too many jobs pending.
0	1	$80C3_H$	Resources occupied (memory).
0	1	$80C4_H$	Communications error (occurs temporarily, it is usually best to repeat the job in the user program).
0	1	$80D2_H$	Module start address incorrect.
0	1	8183_H	The configuration does not match the job parameters.
0	1	8184_H	Illegal data type specified for the NAME_STR parameter.
0	1	8186_H	ID parameter invalid. ID = 1, 2....64
0	1	8302_H	
0	1	$8F22_H$	Source area invalid, for example: Area does not exist in the DB

DONE	ERROR	STATUS	Meaning
0	1	$8F23_H$	Area length error writing a parameter (e.g. DB too short).
0	1	$8F24_H$	Area error reading a parameter
0	1	$8F28_H$	Alignment error reading a parameter
0	1	$8F32_H$	Parameter contains a DB number that is too high.
0	1	$8F33_H$	DB number error
0	1	$8F3A_H$	Area not loaded (DB)
0	1	$8F50_H$	File DB DB 0 or DB does not exist
0	1	$8F51_H$	Specified file DB data area larger than existing data area
0	1	$8F52_H$	File DB in write-protected memory
0	1	$8F53_H$	File DB max. length < current length
0	1	$8F54_H$	File DB does not contain any valid data.
0	1	$8F55_H$	Header status bit: Locked
0	1	$8F56_H$	The NEW bit in the file DB header was not reset
0	1	$8F57_H$	The FTP client does not have write access to the file DB but rather the FTP server (header status bit: WriteAccess).
0	1	$8F60_H$	Bad user data, for example bad IP address of the FTP server
0	1	$8F61_H$	FTP server not obtainable
0	1	$8F62_H$	Possible meanings: • Job not supported or rejected by FTP server • The FTP server does not support SSL-secured connections.
0	1	$8F63_H$	File transfer aborted by the FTP server
0	1	$8F64_H$	Error on the FTP control connection; data could not be sent or received; the FTP control connection must be established again after such an error.
0	1	$8F65_H$	Error on the FTP data connection; data could not be sent or received. The job must be called again. This error can, for example, be caused by RETRIEVE (CMD=3) when the addressed file is already open on the FTP server.
0	1	$8F66_H$	Error reading/writing data from/to the CPU (for example DB does not exist or too short)
0	1	$8F67_H$	Error in the FTP client on the CP; for example attempting to open more than the maximum number of FTP connections.
0	1	$8F68_H$	The job was rejected by the FTP client. This error can, for example, be caused by RETRIEVE (CMD=3) when the value for the parameter MAX_LENGTH was selected too low in the file DB header.
0	1	$8F69_H$	The FTP connection is in an incorrect status, for example: • The connection is called without a previous connection termination (with the same connection ID) • There is a connection termination for a connection that has already been terminated: • A STORE command was sent on a connection that is not established.
0	1	$8F6A_H$	The connection could not be established due to a temporary resource bottleneck. Remedy: Repeat the block call.

DONE	ERROR	STATUS	Meaning
0	1	$8F6B_H$	Possible causes: • Wrong value for the CMD parameter • An FTP_CMD command is not supported. Possible cause: Wrong firmware on the CP Remedy: Firmware update
0	1	$8F6C_H$	A value > 7FFF FFF_H was set in the OFFSET parameter.
0	1	$8F6D_H$	The FTP client does not support SSL-secured connections.
0	1	$8F6F_H$	Possible causes: • The certificate contains an invalid value for "notBefore". • The certificate is invalid. The "notBefore" entry contains a time after the current time.
0	1	$8F70_H$	Possible causes: • The certificate contains an invalid value for "notAfter". • The certificate has elapsed: The "notAfter" entry contains a time before the current time.
0	1	$8F71_H$	The issuer certificate of a non-trustworthy certificate could not be found.
0	1	$8F72_H$	The original CA certificate is invalid. This is either not a CA certificate or its expansions are not consistent with the intended purpose.
0	1	$8F73_H$	The original CA certificate is marked as non-trustworthy for the specified purpose.
0	1	$8F74_H$	Other errors occurred during the verification of a certificate.
0	1	$8F7F_H$	Internal error, for example illegal ANY reference

See also

Overview of FTP_CMD (Page 7829)

Structure and use of the file DB

Structure of the data blocks (file DBs) for FTP services - FTP server mode

Procedure

To transfer data with FTP, create data blocks (file DBs) on the CPU of your S7 station. These data blocks must have certain structure to allow them to be handled as transferable files by the FTP services. They consist of the following sections:

- Section 1: File DB header (has a fixed length (20 bytes) and structure)
- Section 2: User data (has a variable length and structure)

File DB header for FTP server mode

Note: The file DB header described here is largely identical to the file DB header for client mode. The differences relate to the following parameters:

- WRITE_ACCESS
- FTP_REPLY_CODE

Parameter	Type	Value / meaning	Supply
EXIST	BOOL	The EXIST bit indicates whether the user data area contains valid data. The retrieve FTP command executes the job only when EXIST=1. • 0: The file DB does not contain valid user data ("file does not exist"). • 1: The file DB contains valid user data ("file exists").	The dele FTP command sets EXIST=0; The store FTP command sets EXIST=1;
LOCKED	BOOL	The LOCKED bit is used to restrict access to the file DB. • 0: The file DB can be accessed. • 1: The file DB is locked.	The stor and retr FTP commands set LOCKED=1 when they are executed. The following function is also possible when writing from the user program: The user program on the S7 CPU can set or reset LOCKED during write access to achieve data consistency. Recommended sequence in the user program: 1. Check LOCKED bit; if = 0 2. Set WRITEACCESS bit = 0 3. Check LOCKED bit; if = 0 4. Set LOCKED bit = 1 5. Write data 6. Set LOCKED bit = 0

Parameter	Type	Value / meaning	Supply
NEW	BOOL	The NEW bit indicates whether data has been modified since the last read access. • 0: The content of the file DB is unchanged since the last write access. The user program of the S7 CPU has registered the last modification. • 1: The user program of the S7 CPU has not yet registered the last write access.	After execution, the stor FTP command sets NEW=1 After reading the data, the user program on the S7-CPU must set NEW=0 to allow store to be used again or to be able to delete the file with the dele FTP command.
WRITE_ACCESS	BOOL	0: The FTP client on the PG/PC has no write access rights for the file DBs on the S7 CPU. 1: The FTP client on the PG/PC has write access rights for the file DBs on the S7 CPU.	During the configuration of the DB, the bit is set to an initialization value. Recommendation: Whenever possible, the bit should remain unchanged! In special situations, adaptation during operation is possible.
ACT_LENGTH	DINT	Current length of the user data area. The content of this field is only valid when EXIST = 1.	The current length is updated following write access.
MAX_LENGTH	DINT	Maximum length of the user data area (length of the entire DB less 20 bytes header).	The maximum length should be specified during configuration of the DB. The value can also be modified by the user program during operation.
FTP_REPLY_CODE	INT	This parameter is irrelevant in FTP server mode.	Is set to "0" by the FTP server.
DATE_TIME	DATE_AND_TIME	Date and time of the last modification to the file. The content of this field is only valid when EXIST = 1.	The current date is updated following a write access. If the function for forwarding the time of day is used, the entry corresponds to the time that was passed on. If the function for forwarding the time of day is not used, a relative time is entered. This time relates to the startup of the IT-CP (the initialization value is 1.1.1994 0.0 (midnight)).

Structure of the data blocks (file DBs) for FTP services - FTP client mode

Procedure

To transfer data with FTP, create data blocks (file DBs) on the CPU of your S7 station. These data blocks must have certain structure to allow them to be handled as transferable files by the FTP services. They consist of the following sections:

- Section 1: File DB header (has a fixed length of 20 bytes)
- Section 2: User data (has a variable length and structure)

File DB header for FTP client mode

Note: The file DB header described here is largely identical to the file DB header for server mode. The differences relate to the following parameters:

- WRITE_ACCESS
- FTP_REPLY_CODE

Parameter	Type	Value / meaning	Supply
EXIST	BOOL	The EXIST bit indicates whether the user data area contains valid data. The retrieve FTP command executes the job only when EXIST=1. • 0: The file DB does not contain valid user data ("file does not exist"). • 1: The file DB contains valid user data ("file exists").	The dele FTP command sets EXIST=0; The stor FTP command sets EXIST=1;
LOCKED	BOOL	The LOCKED bit is used to restrict access to the file DB. • 0: The file DB can be accessed. • 1: The file DB is locked.	The stor and retr FTP commands set LOCKED=1 when they are executed. The following function is also possible when writing from the user program: The user program on the S7 CPU can set or reset LOCKED during write access to achieve data consistency. Recommended sequence in the user program: 1. Check LOCKED bit; if = 0 2. Set WRITEACCESS bit = 0 3. Check LOCKED bit; if = 0 4. Set LOCKED bit = 1 5. Write data 6. Set LOCKED bit = 0
NEW	BOOL	The NEW bit indicates whether data has been modified since the last read access. • 0: The content of the file DB is unchanged since the last write access. The user program of the S7 CPU has registered the last modification. • 1: The user program of the S7 CPU has not yet registered the last write access.	After execution, the stor FTP command sets NEW=1 After reading the data, the user program in the S7-CPU must set NEW=0 to allow a new retr command.

Parameter	Type	Value / meaning	Supply
WRITE_ACCESS	BOOL	0: The user program (FTP client blocks) has write access rights for the file DBs on the S7 CPU. 1: The user program (FTP client blocks) has no write access rights for the file DBs on the S7 CPU.	During the configuration of the DB, the bit is set to an initialization value. Recommendation: Whenever possible, the bit should remain unchanged! In special situations, adaptation during operation is possible.
ACT_LENGTH	DINT	Current length of the user data area. The content of this field is only valid when EXIST = 1.	The current length is updated following write access.
MAX_LENGTH	DINT	Maximum length of the user data area (length of the entire DB less 20 bytes header).	The maximum length should be specified during configuration of the DB. The value can also be modified by the user program during operation.
FTP_REPLY_CODE	INT	Unsigned integer (16-bit) containing the last reply code from FTP as a binary value. The content of this field is only valid when EXIST = 1.	This is updated by the FTP client when the FTP command is executed.
DATE_TIME	DATE_AND_TIME	Date and time of the last modification to the file. The content of this field is only valid when EXIST = 1.	The current date is updated following a write access. If the function for forwarding the time of day is used, the entry corresponds to the time that was passed on. If the function for forwarding the time of day is not used, a relative time is entered. This time relates to the startup of the IT-CP (the initialization value is 1.1.1994 0.0 (midnight)).

FILE_DB_HEADER data block as a template

Meaning

The data type FILE_DB_HEADER is predefined for creating a file DB header.

Procedure

To transfer data with FTP, create data blocks (file DBs) on the CPU of your S7 station. These data blocks must have certain structure to allow them to be handled as transferable files by the FTP services. They consist of the following sections:

- Section 1: File DB header (has a fixed length of 20 bytes)
- Section 2: User data (has a variable length and structure)

Follow the steps outlined below:

1. For the PLC CPU on which you create the user program with the FTP instructions, create a data block of the type "Global DB".
2. Select the line you want to use as the start line for the file DB.
3. From the drop-down list in the "Data type" column, select a structure element of the type FILE_DB_HEADER.

Result: A data structure with the header structure required for the file DB is created.

Note

"Add new block" function - type selection

When you create new data blocks, the "FILE_DB_HEADER" block type is also available under the "Type" entry in the drop-down list. Do not use this option! The data block created in this way only contains the header structure and cannot be expanded by the area required for storing user data.

FILE_DB_HEADER data block - example and template for the file DB header

In the declaration view, you will see the following structure:

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	bit08	BOOL	FALSE	reserved
+0.1	bit09	BOOL	FALSE	reserved
+0.2	bit10	BOOL	FALSE	reserved
+0.3	bit11	BOOL	FALSE	reserved
+0.4	bit12	BOOL	FALSE	reserved
+0.5	bit13	BOOL	FALSE	reserved
+0.6	bit14	BOOL	FALSE	reserved
+0.7	bit15	BOOL	FALSE	reserved
+1.0	EXIST	BOOL	FALSE	if TRUE: FileDB content is valid data
+1.1	LOCKED	BOOL	FALSE	if TRUE: FileDB is locked caused by changes of the content
+1.2	NEW	BOOL	FALSE	if TRUE: FileDB content is new and may not be overwritten
+1.3	WRITE_ACCESS	BOOL	FALSE	if TRUE: Ftp-Server of the IT-CP has write access, else Ftp-Server
+1.4	bit04	BOOL	FALSE	reserved
+1.5	bit05	BOOL	FALSE	reserved
+1.6	bit06	BOOL	FALSE	reserved
+1.7	bit07	BOOL	FALSE	reserved

Address	Name	Type	Initial value	Comment
+2.0	ACT_LENGTH	DINT	L#0	actual size of the content in bytes (not including the header of 20 bytes)
+6.0	MAX_LENGTH	DINT	L#0	max. size of the content in bytes (not including the header of 20 bytes)
+10.0	FTP_REPLY_CODE	INT	0	last reply code from the remote FTP server
+12.0	DATE_TIME	DATE_AND_TIME	DT#00-1-1-0:0:0.000	date and time of last change of the content of the FileDB
=20.0		END_STRUCT		

Differences in the modes

File DB header for FTP client mode

The file DB header described here is largely identical for the FTP client mode and FTP server mode. The differences relate to the following parameters:

- WRITE_ACCESS
- FTP_REPLY_CODE

Note the ways in which the descriptions of the parameters differ:

- Structure of the data blocks (file DBs) for FTP services - FTP server mode (Page 7837)
- Structure of the data blocks (file DBs) for FTP services - FTP client mode (Page 7840)

Instructions for ERPC-CP

LOGICAL_TRIGGER for the logical trigger

Meaning of the instruction

The LOGICAL_TRIGGER instruction is available if you want to use a logical trigger for ERPC communication.

To start a logical trigger, call the LOGICAL_TRIGGER instruction in the user program of the CPU in OB1.

Further blocks are required for the LOGICAL_TRIGGER instruction call:

- An automatically generated instance DB
- A data block "CONF_DB"
  This configuration DB contains the configuration data of the logical trigger. You create and configure the configuration DB available in the STEP 7 project.
  If you want to call more than one logical trigger, you will also need to make more than one configuration DB available.

You can change the numbers of the LOGICAL_TRIGGER instruction and the instance DB.

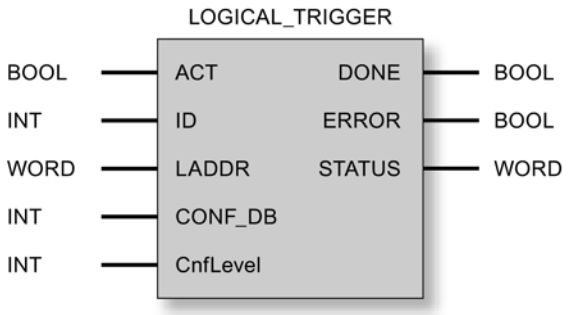
Validity

The LOGICAL_TRIGGER instruction can be used with the following module types:

- CP 343-1 ERPC

Call

Call interface in FBD representation



See also

The configuration data block (Page 7849)
How LOGICAL_TRIGGER works (Page 7845)
Explanation of the formal parameters for LOGICAL_TRIGGER (Page 7846)
LOGICAL_TRIGGER instruction codes (Page 7847)

How LOGICAL_TRIGGER works

Operating principle

The following table shows the steps involved in a trigger call by the user program of the CPU.

Step	Meaning
1	The LOGICAL_TRIGGER instruction is called at the intended point in the user program of the CPU with the corresponding instance DB and the selected configuration data block CONF_DB. • If the LOGICAL_TRIGGER instruction is called with ACT = 1, the current trigger data is read and sent to the CP firmware. • If the LOGICAL_TRIGGER instruction is called with ACT = 0, the status codes DONE, ERROR and STATUS are updated.
2	The LOGICAL_TRIGGER instruction reads the current trigger data.
3	the LOGICAL_TRIGGER instruction creates the PDU that will be sent to the CP firmware with the current trigger data.
4	The CP firmware creates the data frame and transfers it to the ERPC application.
5	The ERPC application sends the data frame to the ERP subscriber (ERP system or MES).

See also

LOGICAL_TRIGGER for the logical trigger (Page 7844)

Explanation of the formal parameters for LOGICAL_TRIGGER

Explanation of the formal parameters

The following table explains the formal parameters for the call interface of the LOGICAL_TRIGGER instruction:

Parameter	Declaration	Data type	Possible values	Description
ACT	INPUT	BOOL	0	If the LOGICAL_TRIGGER instruction is called with ACT = 0, the status codes DONE, ERROR and STATUS are updated.
			1	If the LOGICAL_TRIGGER instruction is called with ACT = 1, the current trigger data is read in and sent to the CP.
ID	INPUT	INT		Trigger ID This value identifies the logical trigger configured in the ILS Workbench.
LADDR	INPUT	WORD		Module start address When you configure the CP with STEP 7, the module start address is displayed in the configuration table by HW Config. Specify this address here.
CONF_DB	INPUT	INT		This data block contains the configuration data of the configured logical trigger.
CnfLevel	INPUT	INT	0: Transport acknowledgment 1: End-to-end acknowledgment	Acknowledgment mode You can find the relevant acknowledgment based on the STATUS value in the codes of the LOGICAL_TRIGGER instruction. • 0 = transport acknowledgment (STATUS = 0000_H) The job is reported as successful, as soon as the data is transferred to the ERPC application. This does not necessarily mean that the data frame was sent to the ERP subscriber (ERP system or MES) and does not preclude the ERPC application detecting an error later. • 1 = end-to-end acknowledgment (STATUS = 0001_H) The job is only acknowledged after the ERPC application has checked the data. The "TriggerResponse" variable of the configuration DB (DB_CONF) is used to report whether or not the ERP subscriber could be reached an whether the ERPC application is in store-and-forward mode. Compared with the transport acknowledgment, the end-to-end acknowledgment means a longer job execution time.

Parameter	Declaration	Data type	Possible values	Description
DONE	OUTPUT	BOOL	0: Job active 1: Job completed	The parameter indicates whether or not the job for transferring the configuration data area was handled free of errors. When the job is accepted, DONE is set to 0 by the CP. As long as DONE = 0, no further job can be triggered. For the meaning in the context of the ERROR and STATUS parameters, refer to the table "LOGICAL_TRIGGER instruction codes".
ERROR	OUTPUT	BOOL	0: - 1: Error situation	Error code For the meaning in the context of the DONE and STATUS parameters, refer to the table "LOGICAL_TRIGGER instruction codes".
STATUS	OUTPUT	WORD	Refer to the table "LOGICAL_TRIGGER instruction codes".	Status code For the meaning in the context of the DONE and ERROR parameters, refer to the table "LOGICAL_TRIGGER instruction codes".

See also

LOGICAL_TRIGGER for the logical trigger (Page 7844)

LOGICAL_TRIGGER instruction codes (Page 7847)

LOGICAL_TRIGGER instruction codes

Condition codes

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

Table 9-360 LOGICAL_TRIGGER instruction codes

DONE	ERROR	STATUS	Meaning
Codes relating to job execution			
1	0	0000_H	Job completed without error. The logical trigger was completed successfully.
1	0	0001_H	Job completed without error. The database is unreachable (store-and-forward mode).
0	0	8181_H	Job active.
0	1	7000_H	The LOGICAL_TRIGGER instruction was called with ACT = 0. The job will, however, not be executed. Call the LOGICAL_TRIGGER instruction at least once with ACT = 1.
Codes relating to configuration and the sequence of the logical trigger			
0	1	$80D2_H$	The CP in use does not support ERPC communication (wrong CP type).
0	1	8183_H	The CP in use does not support ERPC communication (wrong CP type).
0	1	8187_H	Invalid status of the LOGICAL_TRIGGER instruction (unknown LOGICAL_TRIGGER_STATE). Call the LOGICAL_TRIGGER instruction again.
0	1	$8A01_H$	The number of configured logical triggers equals 0.

DONE	ERROR	STATUS	Meaning
0	1	$8A02_H$	The is no configuration in the configuration DB for this logical trigger. Check the ILS Workbench configuration.
0	1	$8A03_H$	The structure of the configuration DB is incorrect. The "header identifier" does not have the correct value. Correct the value of the "ident" variable in the configuration DB (see manual of the ERPC-CP).
0	1	$8A04_H$	The structure of the configuration DB is incorrect. Download the ILS Workbench configuration to the CP again, create and configure the configuration DB(s) again (see "ERPC-CP" manual).
0	1	$8A05_H$	The configured configuration DB does not exist on the CPU.
0	1	$8A06_H$	The next call called a trigger that is still running with a different ID. Check the "ID" in the called LOGICAL_TRIGGER instruction.
0	1	$8A08_H$	The configuration data in the configuration DB does not exist or is incomplete. If the error occurs only during startup of the S7 station, the cause may be that the configuration data of the logical trigger was not completely transferred to the configuration DB. If the error continues to occur, check the configuration of the ERPC symbols.
0	1	$8A09_H$	An unknown error was reported in the configuration DB.
0	1	$8A0A_H$	The logical trigger cannot be started because a new trigger configuration is currently being loaded.
0	1	$8A0B_H$	Error identifying the time stamp of the current data record (CPU data)
0	1	$8A0C_H$	The configuration DB was created with the "Unlinked" property. Correct the object properties of the LOGICAL_TRIGGER instruction.
0	1	$8A0D_H$	Error in the the input parameter CONF_DB of the LOGICAL_TRIGGER instruction. The parameter has the value "0" or higher than the maximum DB number for the CPU.
0	1	$8A0E_H$	The transferred trigger ID is not in the permitted range of 1...16. Correct the value in the LOGICAL_TRIGGER instruction call in the user program.
0	1	$8A0F_H$	The set acknowledgment mode (CnfLevel) is invalid. Correct the value in the LOGICAL_TRIGGER instruction call in the user program.
0	1	$8Bxx_H$	Error copying the current variable values to the PDU of the logical trigger. The last two places (xx) are the variable number. Check the configuration of the symbol involved in the symbol table of the CPU and in the list of ERPC symbols in the properties dialog of the CP.
0	1	$8C01_H$	The internal status code of the LOGICAL_TRIGGER instruction is invalid. Download the ILS Workbench configuration to the CP again, create and configure the configuration DB(s) again (see "ERPC-CP" manual).
0	1	$8C02_H$	The return value of the end-to-end acknowledgment is invalid. Download the ILS Workbench configuration to the CP again, create and configure the configuration DB(s) again (see "ERPC-CP" manual).
0	1	$8C03_H$	The logical trigger contains more than 255 variables.
0	1	$8C06_H$	Error reading the data record.
0	1	$8D03_H$	The firmware is signaling a timeout during a database action.
0	1	$8D04_H$	The database application is signaling a general error in the acknowledgment of the current action.

DONE	ERROR	STATUS	Meaning
0	1	$8E01_H$	The configured configuration DB on the CPU is not large enough. Change the size of the configuration DB.
0	1	$8EXX_H$	These status codes with values for XX_H in the range $02_H..FF_H$ are generated from an internal trigger response. If such values occur, they are relevant for service purposes.

See also

LOGICAL_TRIGGER for the logical trigger (Page 7844)

Explanation of the formal parameters for LOGICAL_TRIGGER (Page 7846)

The configuration data block

Preparing the configuration data block "CONF_DB"

If you use the "logical trigger" ERPC function, you will need to create a data block (DB) in STEP 7 for the configuration data of the logical trigger and specify it in the call parameters of the LOGICAL_TRIGGER instruction. The LOGICAL_TRIGGER instruction accesses DB CONF_DB. CONF_DB has no further significance for the user program.

Programming the configuration data block

To identify the newly created DB, you will need to open the DB and specify the "header identifier" and the DB size in the first two free lines.

Open the DB in STEP 7 and configure the first two free lines with the variables "ident" and "data" as follows:

Address	Name	Type	Initial value	Comment (*optional*)
*)		`STRUCT` *)		
*)	`ident`	`DWORD`	`DW#16#45525043`	*header identifier*
*)	`data`	`array[1..2048]`		*DB size (see warning below)*
*)		`Byte`		
*)		`END_STRUCT` *)		
*) Values are entered by the program				

Note

DB size

2 048 bytes are recommended as the DB size. If it becomes apparent during commissioning that this value is not enough, increase it. If a value is too low, this is reported by the LOGICAL_TRIGGER instruction with an error and the STATUS "$8A05_H$".

See also

LOGICAL_TRIGGER for the logical trigger (Page 7844)

PROFIBUS

Introduction to PROFIBUS

General information on the FCs / FBs for PROFIBUS CPs

Available instructions

The following list shows the symbolic names of the program blocks (instructions) as supplied. You cannot change these symbolic names.

Communication service / functional area	Instruction (symbolic name) 1)	Library	
		CP 300	CP 400
PROFIBUS DP	DP_SEND	x	x
	DP_RECV	x	x
	DP_DIAG	x	x
	DP_CTRL	x	x
SEND / RECEIVE (open communications services)	AG_SEND		x
	AG_RECV		x
	AG_LSEND		x
	AG_LRECV		x
S7 communication	BSEND		x
	BRCV		x
	PUT		x
	GET		x
	USEND		x
	URCV		x
	C_CNTRL		x

1) Note:
The following descriptions also include information on differences in behavior between the various versions of the instructions. Please check and note the version identifiers of the instructions you are using.
The SIMATIC Manager global libraries installed with the Automation Workbench contain the versions of the instructions current when STEP 7 was released.

Which block version should I use?

The following descriptions also include information on differences in behavior between the various versions of the instructions. Please check and note the version identifiers of the instructions you are using.

Note

We recommend that you always use the latest versions of the instructions for all module types.

You will find information on the current versions of the instructions and the current instructions to download from the Internet in our customer support:

http://support.automation.siemens.com/WW/view/en/8797900

With older module types, this recommendation assumes that you are using the latest firmware for the particular module type.

Instructions when modules are replaced

Module replacement in this sense means the replacement of a module with another module that may be a more recent version.

Note

Please remember that if you replace a module, you must only use the instructions permitted for the configured CP type in the user program.

We recommend that you always use the latest versions of the instructions for all module types. With older module types, this recommendation assumes that you are using the latest firmware for the particular module type.

You will find more information on replacing modules in our Customer Support on the Internet.

http://support.automation.siemens.com/WW/view/en/7806643

The manuals contain information on the compatibility of the S7-CPs and the corresponding instructions.

Setting parameters for block / function calls

Before describing the instruction in detail, a few general comments on calling and setting parameters for instructions will be useful at this point.

The general information below applies to the following parameter groups that exist for all instructions:

- Parameters for CP and connection assignment (input parameters)
- Parameters for specifying a CPU data area (input parameters)
- Status information (output parameters)

Calling communications instruction for an S7300

NOTICE
Calling the communications instructions for S7-300 in more than one priority class is not permitted! If, for example, you call a communication instruction in OB1 and in OB35, execution of the instruction could be interrupted by the higherpriority OB. If you call instructions in more than one OB, write your program so that a communications instruction that is currently executing cannot be interrupted by another communications instruction (for example by disabling/enabling interrupts with system instructions).

Parameters for specifying a CPU data area (input parameters)

Specifying the data area on the CPU

When you call an instruction, you transfer the address and length of the data area on the CPU in which the user data is available or will be stored or which can contain further parameter information.

The ANY pointer data type is used to address this area.

Status information (output parameters)

Evaluating status codes

For status evaluation, the following parameters must be evaluated in the user program:

- DONE or NDR
  These parameters (DONE with send jobs and NDR with receive jobs) signal (successful) completion of the job.
- ERROR
  This indicates that the job could not be executed errorfree.
- STATUS
  This parameter supplies detailed information about the execution of the job. Status codes can be returned during execution of the job (DONE=0 and ERROR=0).

Note

Remember that the status codes DONE, NDR, ERROR, STATUS are updated at each instruction call.

Status codes during CP startup

With a warm or hot restart on the Ethernet CP, the output parameters of the instruction are reset as follows:

- DONE = 0
- NDR = 0
- ERROR = 0
- STATUS = 8180H or 8181H

Instructions for open communications services (SEND/RECEIVE interface)

Instructions for FDL connections (SEND/RECEIVE interface)

Overview

The following instructions are available for the SEND/RECEIVE interface for transferring data on configured FDL connections:

Instruction	Can be used with [1]		Meaning
S7-300	S7-400		
AG_SEND	x	x	for sending data
AG_RECV	x	x	for receiving data
AG_LSEND		x	for sending data
AG_LRECV		x	for receiving data

1) Notes on the Instructions for an S7300 and S7400

To ensure the compatibility of PROFIBUS and Industrial Ethernet on the interface in the user program, you can use the AG_LSEND and AG_LRECV instructions on PROFIBUS as alternatives to AG_SEND and AG_RECV. There is no difference in the interface or the way they function. On PROFIBUS, however, you can only transfer data up to a maximum of 240 bytes even with these instructions although they are intended for longer data records on Industrial Ethernet.

This is only possible if the type and version of the instructions are permitted for the CP type you are using.

With the S7 CPs for S7300, only the AG_SEND and AG_RECV instructions are used; on Industrial Ethernet even for the transfer of longer data records.

The manuals contain information on the compatibility of the S7-CPs and the corresponding instructions. You will find an overview of the versions of the instructions in the documentation and block history.

Application

The following diagram illustrates the use of the AG_SEND / AG_LSEND and AG_RECV / AG_LRECV instructions for bi-directional data transfer on a configured FDL connection. With certain connection types, a job header should be included in the user data area.

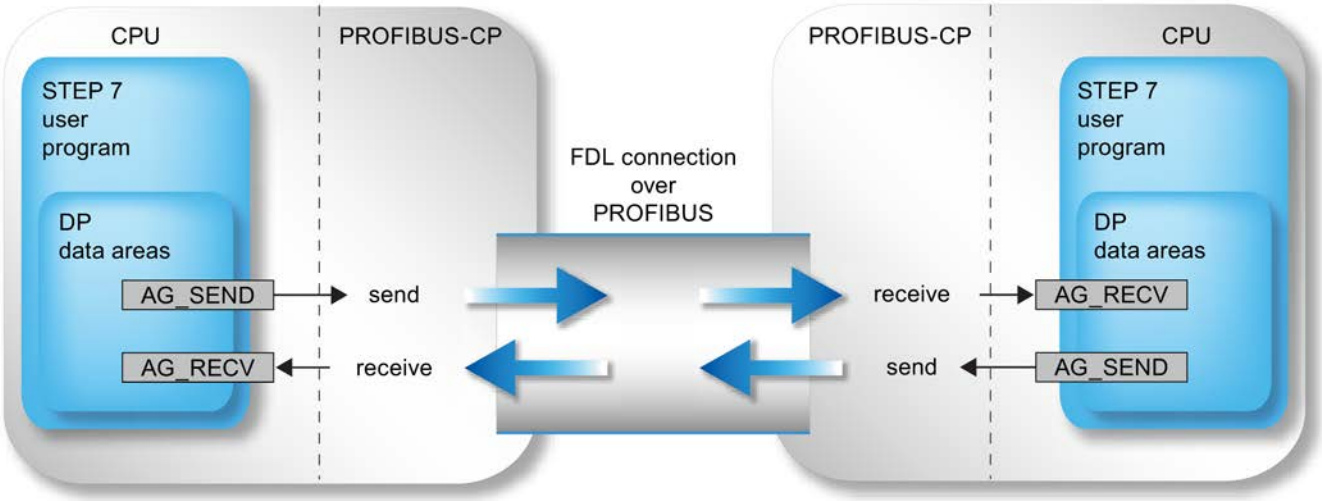


Figure 9-22 Using AG_SEND and AG_RECV on both communications partners

Application without job header

With a specified FDL connection, the address and job parameters are specified by the configuration of the connection. The user program only provides the user data in the FDL data area when sending with AG_SEND / AG_LSEND or receives the data with AG_RECV / AG_LRECV.

Up to 240 bytes of user data can be transferred. This information applies to AG_SEND and AG_LSEND on PROFIBUS.

Working with the job header

The following connection types require a job header in the FDL (user) data area:

- Unspecified FDL connection with free layer 2 access
- FDL connection with broadcast
- FDL connection with multicast

The following schematic illustrates the structure of the job buffer and the meaning and location of the parameters in the job header.

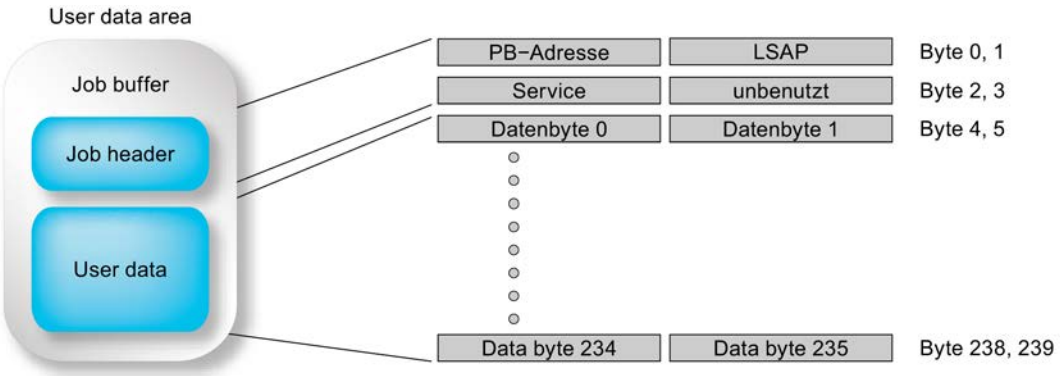


Figure 9-23 Sending and receiving via an FDL connection with programmed broadcast addressing

|The user data area can be up to 240 bytes. Up to 236 bytes of user data can be transferred. 4 bytes are reserved for the job header.

Please note that the data length specified in the instruction call (LEN parameter) must include the header and the user data!

AG_SEND / AG_LSEND

AG_SEND / AG_LSEND (PROFIBUS)

Description

The FC AG_SEND / AG_LSEND instruction transfers data to the PROFIBUS CP for transmission on a configured FDL connection.

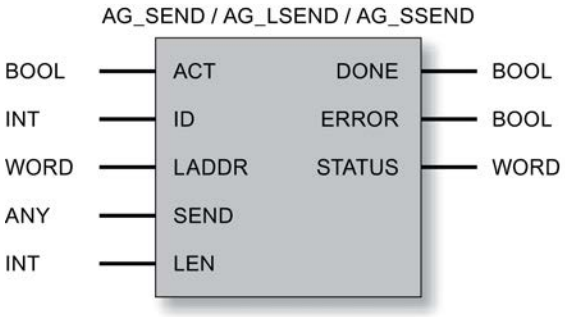
The selected data area can be a process image area, a memory bit area or a data block area.

Error free execution of the function is indicated when the entire FDL data area could be sent on PROFIBUS.

Note:
Unless otherwise stated, all the following information applies equally to AG_SEND and AG_LSEND.

Call

Call interface in FBD representation



Calls with job header

The following table shows the connection types and job types for which parameters must be supplied in the job header.

The job header is located in the FDL (user) data area. It occupies the first 4 bytes and must be added to the length specified in the LEN parameter. The maximum user data length is therefore reduced for jobs with a job header to 236 bytes.

Table 9-361 Supplying the job header in the user data area

Parameter	FDL connection type		
Unspecified: free layer 2 [2]	**Broadcast**	**Multicast**	
PB address	Address of the destination station Range of values: 0 to 126 depending on node / 127 for broadcast/multicast	For AG_SEND no relevance; but area must be reserved.	For AG_SEND no relevance; but area must be reserved.
LSAP	LSAP of the destination station Range of values: 0 to 62 depending on node / 63 for broadcast	No significance but area must be reserved.	No significance but area must be reserved.
Service [1]	SDA (Send Data with Acknowledge): Value: 00_H SDN (Send Data with No Acknowledge): Value: 01_H	No significance but area must be reserved.	No significance but area must be reserved.

[1] for broadcast and multicast, only the SDN service is possible.

[2] The information on broadcast and multicast in this column is relevant only when an unspecified FDL connection is used for broadcast or multicast. On a configured FDL connection (recommended application) with broadcast or multicast as the connection partner, the address parameters are assigned automatically according to the configuration.

See also

Parameters for AG_SEND / AG_LSEND (PB) (Page 7857)
Parameters DONE, ERROR, STATUS (PB) (Page 7858)
Example of AG_SEND (PB) (Page 7860)

Parameters for AG_SEND / AG_LSEND (PB)

Explanation of the formal parameters

The following table explains all the formal parameters for the AG_SEND /AG_LSEND instructions:

Parameter	Declaration	Data type	Possible values	Description
ACT	INPUT	BOOL	0, 1	If the instruction is called with ACT=1, LEN bytes are sent from the ISO transport data area specified with the SEND parameter. If the instruction is called with ACT = 0, the status codes DONE, ERROR and STATUS are updated.
ID	INPUT	INT	1,2...64 (S7-400) 1,2...16 (S7-300)	The connection number of the FDL connection is specified in the parameter ID.
LADDR	INPUT	WORD		Module start address When you configure the CP, the module start address is displayed in the configuration table. Specify this address here.
SEND	INPUT	ANY (only the following are permitted as VARTYPE: WORD and DWORD are permitted)		Specifies the address and length The address of the data area points to one of the alternatives: • PI area • Memory bit area • Data block area With a call with job header, the FDL data area contains the job header and the user data.
LEN	INPUT	INT	1, 2, to 240 (or up to "length specified for SEND parameter")	Number of bytes to be sent from the FDL data area with this job. The possible values range from 1 to length specified for the SEND parameter. In a call, with job header, the length information is made up of the job header (4 bytes) + user data (1 to 236 bytes). Therefore LEN >= 4 !
DONE	OUTPUT	BOOL	0: - 1: new data	The status parameter indicates whether or not the job was completed without errors. For the meaning in conjunction with the ERROR and STATUS parameters, refer to the following table.

Parameter	Declaration	Data type	Possible values	Description
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning in conjunction with the DONE and STATUS parameters, refer to the following table.
STATUS	OUTPUT	WORD	See following table	Status code For the meaning in conjunction with the DONE and ERROR parameters, refer to the following table.

See also

AG_SEND / AG_LSEND (PROFIBUS) (Page 7855)

Parameters DONE, ERROR, STATUS (PB)

Condition codes

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

Note

For entries coded with 8FxxH in STATUS, refer to the information about the output parameter RET_VAL in the descriptions of the referenced system program blocks.

Which system program blocks are used and are relevant for error evaluation, can be queried in STEP 7.

Table 9-362 AG_SEND condition codes

DONE	ERROR	STATUS	Meaning
1	0	0000_H	Job completed without error.
0	0	0000_H	No job being executed.
0	0	8181_H	Job active.
0	1	7000_H	The condition code is possible only with S7-400: The instruction was called with ACT=0; the job has not yet been processed.
0	1	8183_H	No configuration or the FDL service has not yet started on the PROFIBUS CP.
0	1	8184_H	Possible causes: • Illegal data type specified for the SEND parameter. • FDL connection without job buffer: System error. • FDL connection with job buffer: Parameter LEN<4 or illegal parameter in job header (with free layer 2 access).
0	1	8185_H	LEN parameter longer than SEND source area.
0	1	8186_H	ID parameter invalid. ID!=1, 2 to 15, 16.
0	1	8301_H	SAP not activated on destination station.
0	1	8302_H	No receive resources on the destination station; the receiving station cannot process received data quickly enough or has not prepared any receive resources.

DONE	ERROR	STATUS	Meaning
0	1	8303_H	The PROFIBUS service (SDA Send Data with Acknowledge) is not supported on this SAP by the destination station. This condition code can also occur temporarily when connections or gateways are downloaded "in RUN".
0	1	8304_H	The FDL connection is not established.
0	1	8311_H	The destination station is not obtainable at the specified PROFIBUS address or the service is not possible for the specified PROFIBUS address.
0	1	8312_H	PROFIBUS error on the CP: for example, bus short-circuit, own station not in ring.
0	1	8315_H	Possible causes: • Internal parameter error on an FDL connection with job header: Parameter LEN<4 or illegal parameter in job header (with free layer 2 access). • Bus disruption Possible additional meaning: • This error code can also occur with bus problems (for example physical disturbances due to bad power connections or different settings for the transmission speed on the nodes).
0	1	$8F22_H$	Source area invalid, e.g.: Area does not exist in the DB LEN parameter < 0
0	1	$8F24_H$	Area error reading a parameter.
0	1	$8F28_H$	Alignment error reading a parameter.
0	1	$8F32_H$	Parameter contains a DB number that is too high.
0	1	$8F33_H$	DB number error.
0	1	$8F3A_H$	Area not loaded (DB).
0	1	$8F42_H$	Timeout reading a parameter from the I/O area.
0	1	$8F44_H$	Address of the parameter to be read is disabled in the access track.
0	1	$8F7F_H$	Internal error, e.g. illegal ANY reference e.g. parameter LEN=0
0	1	8090_H	• No module with this module start address exists. • The instruction being used does not match the system family being used (remember to use different instructions for S7300 and S7400).
0	1	8091_H	Module start address not at a doubleword boundary.
0	1	8092_H	In the ANY reference, a type other than BYTE is specified. (S7-400 only)
0	1	$80A4_H$	The communication bus connection between the CPU and CP is not established. (with newer CPU versions). This can, for example, be caused by the following: • No connection configuration; • The maximum number of CPs that can be operated at one time has been exceeded (for further information, refer to the CP manual).
0	1	$80B0_H$	The module does not recognize the data record.
0	1	$80B1_H$	The destination area is invalid. The amount of data to be sent exceeds the upper limit permitted for this service (e.g. destination area > 240 bytes).
0	1	$80B2_H$	The communication bus connection between the CPU and CP is not established (with older CPU versions; otherwise $80A4_H$; for further information, refer to this code)

DONE	ERROR	STATUS	Meaning
0	1	$80C0_H$	The data record cannot be read.
0	1	$80C1_H$	The specified data record is currently being processed.
0	1	$80C2_H$	There are too many jobs pending.
0	1	$80C3_H$	Resources occupied (memory).
0	1	$80C4_H$	Communications error (occurs temporarily, it is usually best to repeat the job in the user program).
0	1	$80D2_H$	Module start address incorrect.

See also

AG_SEND / AG_LSEND (PROFIBUS) (Page 7855)

Example of AG_SEND (PB)

Example of call and evaluation of parameters

Below you will find an executable example of an AG_SEND instruction call and parameter evaluation.

The OB100 listed below belongs to the "AE_460_1" instruction selected here in which the send call takes place; OB100 sets the ACT bit correctly when the CPU starts up.

To function correctly, a DB100 with a size of at least 240 bytes must be loaded.

The program requires a CP at address 256 and a configured connection of the type ISO Transport / ISOonTCP / TCP or FDL with ID=1 (please adapt your configuration where necessary !).

```
//-----------------------------------------------------------------
---
FUNCTION FC 100: VOID
TITLE = SENDE_DEMO
AUTHOR : Tester
FAMILY : S7300
NAME : FC5_Demo
VERSION : 1.0
//-----------------------------------------------------------------
---
BEGIN
    CALL FC 5 (
    ACT        := M100.0,
    ID            := 1,
    LADDR        := W#16#100,
    SEND        := P#DB100.dbx0.0 BYTE 240,
    LEN        := 240,
    DONE        := M100.1,
    ERROR    := M100.2,
    STATUS    := MW102 );
//-----------------------------------------------------------------
---
    R M100.0;          //   Reset parameter ACT for all further
    SET;               //   AG_SEND calls
    A M100.1;          //
    JC done;           //   Test for DONE = TRUE
    SET;               //
    A M100.2;          //
    JC err;            //   Test for ERROR = TRUE
//-----------------------------------------------------------------
---
    BEU;               //   Neither DONE nor ERROR is set, the job is
                       //   still running.
//-----------------------------------------------------------------
---
done:    S M100.0;     //   Job completed without error. Set ACT =
    BEU;               //   TRUE so that the following call can
                            trigger the new job.
//-----------------------------------------------------------------
---
err:    NOP 1;         //   An error occurred. The status word can be
    NOP 1;             //   evaluated here. Set ACT to TRUE in any
    S M100.0;          //   case, so that a new send job can be
    BEU;               //   triggered if the error disappears.
```

```
//-------------------------------------------------------------------
---
END_FUNCTION
ORGANIZATION_BLOCK OB100
TITLE         = Init_for_FC100
FAMILY:    S7300
NAME:        SENDE_DEMO_INIT
VERSION:    1.0
VAR_TEMP
OB1_System: array [1..20] of byte;
END_VAR
//-------------------------------------------------------------------
---
BEGIN                  //
    SET                //
    S M100.0           //   Initialize ACT parameter
END_ORGANIZATION_BLOCK
```

AG_RECV / AG_LRECV

AG_RECV / AG_LRECV (PROFIBUS)

Description

The AG_RECV / AG_LRECV instruction receives the data transferred on a configured FDL connection from the PROFIBUS CP.

The data area specified for the receive data can be a process image area, a bit address area or a data block area.

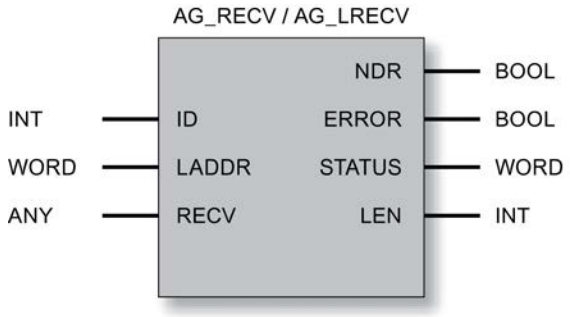
Errorfree execution is indicated when the data could be received from the PROFIBUS CP.

Note:

Unless otherwise stated, all the following information applies equally to AG_RECV and AG_LRECV.

Call interface

Call interface in FBD representation



Calls with job header

Table 9-363 Return parameters in the job header in the FDL (user) data area

Parameter	FDL connection type		
Unspecified: free layer 2)	**Broadcast**	**Multicast**	
PB address	Address of the sender Range of values: 0 to 126 depending on node		
LSAP	LSAP of the sender Values: 0 to 63 depending on node		
Service	SDN indication (Send Data with No Acknowledge - Indication): Value: 01_H or SDA indication (Send Data with Acknowledge - Indication): Value: 00_H	SDN indication (Send Data with No Acknowledge - Indication): Value: $7F_H$	SDN indication (Send Data with No Acknowledge - Indication): Value: $7F_H$

See also

Parameters for AG_RECV / AG_LRECV (PB) (Page 7864)

Parameters DONE, ERROR, STATUS (PB) (Page 7865)

Parameters for AG_RECV / AG_LRECV (PB)

Explanation of the formal parameters

The following table explains all the formal parameters for the AG_RECV / AG_LRECV instruction:

Parameter	Declaration	Data type	Possible values	Description
ID	INPUT	INT	1, 2 to 16 (S7-300) 1, 2 to 32 (S7-400)	The connection number of the FDL connection is specified in the parameter ID.
LADDR	INPUT	WORD		Module start address When you configure the CP, the module start address is displayed in the configuration table. Specify this address here.
RECV	INPUT	ANY (only the following are permitted as VARTYPE: WORD and DWORD are permitted)		Specifies the address and length The address of the FDL data area points to one of the alternatives: • PI area • Memory bit area • Data block area With a call with job header, the FDL data area contains the job header and the user data.
LEN	OUTPUT	INT	1,2,...240	Specifies the number of bytes to be received in the FDL data area from the PROFIBUS CP. In a call, with job header, the length information is made up of the job header (4 bytes) + user data (1 to 236 bytes). Therefore LEN >= 4 !
NDR	OUTPUT	BOOL	0: - 1: new data	This parameter indicates whether new data were received. For the meaning in conjunction with the ERROR and STATUS parameters, refer to the following table.
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning of this parameter in conjunction with the NDR and STATUS parameters, refer to the following table.
STATUS	OUTPUT	WORD	See following table	Status code For the meaning in conjunction with the NDR and ERROR parameters, refer to the following table.

See also

AG_RECV / AG_LRECV (PROFIBUS) (Page 7862)

Parameters DONE, ERROR, STATUS (PB)

Condition codes

The following table shows the codes formed by the NDR, ERROR and STATUS parameters that must be evaluated by the user program.

Note

For entries coded with $8Fxx_H$ in STATUS, refer to the information about the output parameter RET_VAL in the descriptions of the referenced system program blocks.

Which system program blocks are used and are relevant for error evaluation, can be queried in STEP 7.

Table 9-364 AG_RECV / AG_LRECV condition codes

NDR	ERROR	STATUS	Meaning
1	0	0000_H	New data accepted.
0	0	8180_H	• There is no data available yet. The configuration is missing or the FDL service has not started on the PROFIBUS CP (occurs here instead of the code 0,1,8183_H).
0	0	8181_H	Job active.
0	1	8183_H	No configuration or the FDL service has not yet started on the PROFIBUS CP.
0	1	8184_H	• Illegal data type specified for the RECV parameter. • System error.
0	1	8185_H	Destination buffer (RECV) is too short.
0	1	8186_H	ID parameter invalid. ID!=1, 2 to 15, 16.
0	1	8303_H	The PROFIBUS service (SDA - Send Data with Acknowledge) is not supported on this SAP. This condition code can also occur temporarily when connections or gateways are downloaded "in RUN".
0	1	8304_H	The FDL connection is not established.
0	1	$8F23_H$	Source area invalid, e.g.: Area does note exist in the DB.
0	1	$8F25_H$	Area error writing a parameter.
0	1	$8F29_H$	Alignment error writing a parameter
0	1	$8F30_H$	Parameter is in the writeprotected 1st current data block.
0	1	$8F31_H$	Parameter is in the writeprotected 2nd current data block.
0	1	$8F32_H$	Parameter contains a DB number that is too high.
0	1	$8F33_H$	DB number error.
0	1	$8F3A_H$	Destination area not loaded (DB).
0	1	$8F43_H$	Timeout writing a parameter to the I/O area.
0	1	$8F45_H$	Address of the parameter to be written is disabled in the access track.
0	1	$8F7F_H$	Internal error, e.g. illegal ANY reference.

NDR	ERROR	STATUS	Meaning
0	1	8090_H	• No module with this module start address exists. • The instruction being used does not match the system family being used (remember to use different instructions for S7300 and S7400).
0	1	8091_H	Module start address not at a doubleword boundary.
0	1	8092_H	In the ANY reference, a type other than BYTE is specified. (S7-400 only)
0	1	$80A0_H$	Negative acknowledgment reading from the module.
0	1	$80A4_H$	The communication bus connection between the CPU and CP is not established. (with newer CPU versions). This can, for example, be caused by the following: • No connection configuration; • The maximum number of CPs that can be operated at one time has been exceeded (for further information, refer to the CP manual).
0	1	$80B0_H$	The module does not recognize the data record.
0	1	$80B1_H$	Possible causes: • The destination area is invalid. • The destination area is too short. • The destination area for the received data was adequately dimensioned. Remedy: Run another receive call with maximum receive buffer size. This applies regardless of the connection type (unicast / multicast / broadcast) and the device family (S7-300 / S7-400).
0	1	$80B2_H$	The communication bus connection between the CPU and CP is not established.
0	1	$80C0_H$	The data record cannot be read.
0	1	$80C1_H$	The specified data record is currently being processed.
0	1	$80C2_H$	There are too many jobs pending.
0	1	80C3H	Resources occupied (memory).
0	1	$80C4_H$	Communications error (occurs temporarily, it is usually best to repeat the job in the user program).
0	1	$80D2_H$	Module start address incorrect.

See also

AG_RECV / AG_LRECV (PROFIBUS) (Page 7862)

Instructions for DP (distributed I/O) with S7-300

Instructions for the S7300 DP mode

Overview

The following instructions are available for the DP master and DP slave modes with an S7300:

Instruction	can be used with:		Meaning
	DP master	DP slave	
DP_SEND	X	X	for sending data
DP_RECV	X	X	for receiving data
DP_DIAG	X	-	for diagnostic functions initiated by the DP master
DP_CTRL	X	-	for control functions

Application

The following diagram illustrates the use of the DP_SEND and DP_RECV instructions on the DP master and DP slave.

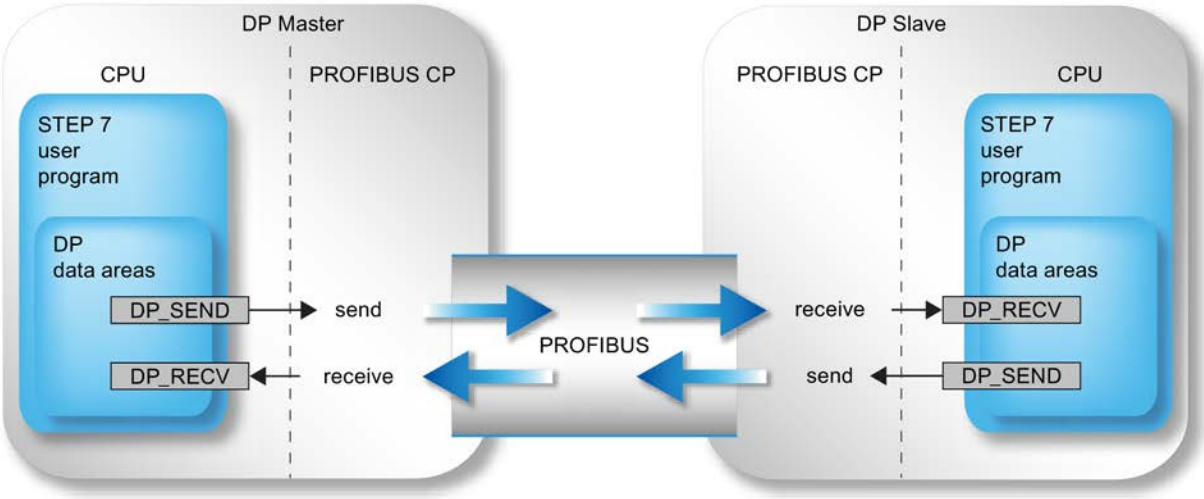


Figure 9-24 Using the DP_SEND and DP_RECV instructions with DP master and DP slave

DP_SEND

DP_SEND

Description

The DP_SEND instruction transfers data to the PROFIBUS CP. Depending on the mode of the PROFIBUS CP, DP_SEND has the following significance:

- When used in the DP master

The instruction transfers the data of a specified DP output area to the PROFIBUS CP for output to the distributed I/O system.

- When used in the DP slave

The instruction transfers the input data of the DP slave to the PROFIBUS CP for transfer to the DP master

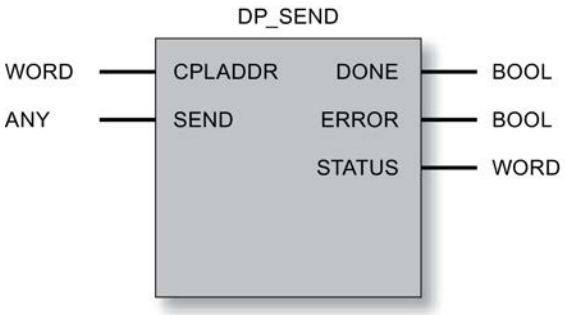
The selected data area can be a process image area, a memory bit area or a data block area.

Correct execution is signaled when the entire DP data area could be accepted by the PROFIBUS CP.

To start the DP master exactly one DP-SEND or DP-RECV call must precede the call sequence. The following applies to this first call:

If DP-SEND is used for initialization, the transferred data area is not accepted and "0" is sent to the slaves. The user data to be transferred is accepted only with the second block call.

Call interface



See also

Parameters for DP_SEND (Page 7869)

Parameters DONE, ERROR, STATUS (Page 7870)

Parameters for DP_SEND

Explanation of the formal parameters

The following table explains all the formal parameters for the DP_SEND instruction:

Parameter	Declaration	Data type	Range of values	Description
CPLADDR	INPUT	WORD		Module start address When you configure the CP, the module start address is displayed in the configuration table. Specify this address here.
SEND	INPUT	ANY (only the following are permitted as VARTYPE: With DP_SEND as of V3: BYTE With DP_SEND up to V2.x: BYTE, WORD and DWORD)		Specifies the address and length The address of the DP data area points to one of the alternatives: • PI area • Memory bit area • Data block area The length must be set for • DP master: 1...21600 • DP slave: 1...240
DONE	OUTPUT	BOOL	0: - 1: new data	The status parameter indicates whether or not the job was completed without errors. For the meaning in conjunction with the ERROR and STATUS parameters, refer toParameters DONE, ERROR, STATUS (Page 7870).
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning in conjunction with the DONE and STATUS parameters, refer to Parameters DONE, ERROR, STATUS (Page 7870)
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the DONE and ERROR parameters, refer to Parameters DONE, ERROR, STATUS (Page 7870)

See also

DP_SEND (Page 7868)

Parameters DONE, ERROR, STATUS

Condition codes

The following table shows the condition codes formed based on DONE, ERROR and STATUS that must be evaluated by the user program.

Note

For entries coded with $8Fxx_H$ in STATUS, refer to the information about the output parameter RET_VAL in the descriptions of the referenced system program blocks.

Which system program blocks are used and are relevant for error evaluation, can be queried in STEP 7.

Table 9-365 DP_SEND condition codes

DONE	ERROR	STATUS	Meaning
0	0	8180_H	• Startup: The DP service was started but data acceptance is not yet possible. • Normal operation Data transfer active. • DP has not started due to: – CP STOP or – "No parameter assignment" (occurs here instead of the code 0,1,8183_H).
1	0	0000_H	New data transferred without error.
0	1	8183_H	No configuration or the DP service has not yet started on the PROFIBUS CP.
0	1	8184_H	System error or bad parameter type.
0	1	$8F22_H$	Area length error reading a parameter (e.g. DB too short).
0	1	$8F23_H$	Area length error writing a parameter (e.g. DB too short).
0	1	$8F24_H$	Area error reading a parameter.
0	1	$8F25_H$	Area error writing a parameter.
0	1	$8F28_H$	Alignment error reading a parameter.
0	1	$8F29_H$	Alignment error writing a parameter.
0	1	$8F30_H$	Parameter is in the writeprotected 1st current data block.
0	1	$8F31_H$	Parameter is in the writeprotected 2nd current data block.
0	1	$8F32_H$	Parameter contains a DB number that is too high.
0	1	$8F33_H$	DB number error.
0	1	$8F3A_H$	Destination area not loaded (DB).
0	1	$8F42_H$	Timeout reading a parameter from the I/O area.
0	1	$8F43_H$	Timeout writing a parameter to the I/O area.
0	1	$8F44_H$	Address of the parameter to be read is disabled in the access track.
0	1	$8F45_H$	Address of the parameter to be written is disabled in the access track.
0	1	$8F7F_H$	Internal error, e.g. illegal ANY reference.
0	1	8090_H	No module with this address exists.

DONE	ERROR	STATUS	Meaning
0	1	8091_H	Logical base address not at a double word boundary.
0	1	$80A1_H$	Negative acknowledgment writing to the module.
0	1	$80B0_H$	The module does not recognize the data record.
0	1	$80B1_H$	The number of data bytes to be sent exceeds the upper limit for this service (applies to DP master and DP slave mode).
0	1	$80C0_H$	The data record cannot be read.
0	1	$80C1_H$	The specified data record is currently being processed.
0	1	$80C2_H$	There are too many jobs pending.
0	1	$80C3_H$	Resources occupied (memory).
0	1	$80C4_H$	Communications error (occurs temporarily, it is usually best to repeat the job in the user program).
0	1	$80D2_H$	Logical base address incorrect.

See also

DP_SEND (Page 7868)

DP_RECV

DP_RECV

Description

The DP_RECV instruction receives data over PROFIBUS. DP_RECV has the following significance depending on the mode of the PROFIBUS CP:

- When used in the DP master
  DP_RECV receives the process data from the distributed I/O along with status information and enters this in a specified DP input area.
- When used on the DP slave
  DP_RECV accepts the output data transferred by the DP master in the DP data area specified in the instruction.

The data area specified for the receive data can be a process image area, a bit address area or a data block area.

Errorfree execution is signaled when the entire DP data input area could be transferred by the PROFIBUS CP.

Note that the DP_RECV instruction must be called successfully at least once on the DP slave in the user program if output data was configured for this DP slave. Please read the information in the manual.

To start the DP master exactly one DP-SEND or DP-RECV call must precede the call sequence. The following applies to this first call:

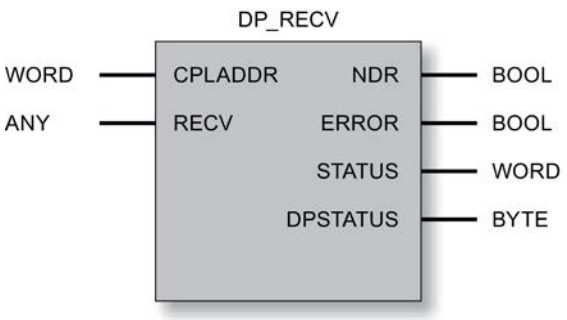
- If DP-RECV is used for initialization, the received data is not adopted. The user data to be received is accepted only with the second block call.

Additional task: Entering the status byte

The DP_RECV instruction has the following additional task:

- Updating the DP status byte DPSTATUS. This means that DP_RECV handles tasks for DP diagnostics
  If no receive data is configured, DP_RECV must be called with a length of 1 to update the DPSTATUS status byte (applies only to DP masters).
  Please read the information in the manual as well.
- Enabling the station list (see DP_DIAG (Page 7877)).

Call interface



See also

Parameters for DP_RECV (Page 7873)

Parameters NDR, ERROR, STATUS (Page 7874)

DPSTATUS - DP_RECV parameters (Page 7875)

Parameters for DP_RECV

Explanation of the formal parameters

The following table explains all the formal parameters for the DP_RECV instruction:

Parameter	Declaration	Data type	Range of values	Description
CPLADDR	INPUT	WORD		Module start address When you configure the CP, the module start address is displayed in the configuration table. Specify this address here.
RECV	INPUT	ANY (only the following are permitted as VARTYPE: With DP_RECV as of V3: BYTE With DP_RECV up to V2.x: BYTE, WORD and DWORD)		Specifies the address and length The address of the DP data area points to one of the alternatives: • PI area • Memory bit area • Data block area The length must be set for: • DP master: 1...2160 • DP slave: 1...240 • DP master; only read status byte: 1 (see also CP manual)
NDR	OUTPUT	BOOL	0: - 1: New data accepted	The status parameter indicates whether or not new data was accepted. For the meaning in conjunction with the ERROR and STATUS parameters, refer to Parameters NDR, ERROR, STATUS (Page 7874).
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning in conjunction with the NDR and STATUS parameters, refer to Parameters NDR, ERROR, STATUS (Page 7874)
STATUS	OUTPUT	WORD		Status code For the meaning in conjunction with the parameters NDR and ERROR, refer to Parameters NDR, ERROR, STATUS (Page 7874)
DPSTATUS	OUTPUT	Byte	For coding, see below under DPSTATUS	DP status code

See also

DP_RECV (Page 7871)

Parameters NDR, ERROR, STATUS

Condition codes

The following table shows the codes formed by the NDR, ERROR and STATUS parameters that must be evaluated by the user program.

Note

For entries coded with $8Fxx_H$ in STATUS, refer to the information about the output parameter RET_VAL in the descriptions of the referenced system program blocks.

Which system program blocks are used and are relevant for error evaluation, can be queried in STEP 7.

NDR	ERROR	STATUS	Meaning
0	0	8180_H	• Startup: The DP service was started but data acceptance is not yet possible. • Normal operation Data transfer active. • DP has not started due to: – CP STOP or – "No parameter assignment" (occurs here instead of the code 0,1,8183_H).
1	0	0000_H	New data accepted without error.
0	1	8183_H	No configuration or the DP service has not yet started on the PROFIBUS CP.
0	1	8184_H	System error or bad parameter type.
0	1	$8F22_H$	Area length error reading a parameter (e.g. DB too short).
0	1	$8F23_H$	Area length error writing a parameter (e.g. DB too short).
0	1	$8F24_H$	Area error reading a parameter.
0	1	$8F25_H$	Area error writing a parameter.
0	1	$8F28_H$	Alignment error reading a parameter.
0	1	$8F29_H$	Alignment error writing a parameter.
0	1	$8F30_H$	Parameter is in the writeprotected 1st current data block.
0	1	$8F31_H$	Parameter is in the writeprotected 2nd current data block.
0	1	$8F32_H$	Parameter contains a DB number that is too high.
0	1	$8F33_H$	DB number error.
0	1	$8F3A_H$	Destination area not loaded (DB).
0	1	$8F42_H$	Timeout reading a parameter from the I/O area.
0	1	$8F43_H$	Timeout writing a parameter to the I/O area.
0	1	$8F44_H$	Address of the parameter to be read is disabled in the access track.
0	1	$8F45_H$	Address of the parameter to be written is disabled in the access track.
0	1	$8F7F_H$	Internal error, e.g. illegal ANY reference.
0	1	8090_H	No module with this address exists.
0	1	8091_H	Logical base address not at a double word boundary.

NDR	ERROR	STATUS	Meaning
0	1	$80A0_H$	Negative acknowledgment reading the module.
0	1	$80B0_H$	The module does not recognize the data record.
0	1	$80B1_H$	The number of data bytes to be sent exceeds the upper limit for this service (applies to DP master and DP slave mode).
0	1	$80C0_H$	The data record cannot be read.
0	1	$80C1_H$	The specified data record is currently being processed.
0	1	$80C2_H$	There are too many jobs pending.
0	1	$80C3_H$	Resources occupied (memory).
0	1	$80C4_H$	Communications error (occurs temporarily, it is usually best to repeat the job in the user program).
0	1	$80D2_H$	Logical base address incorrect.

See also

DP_RECV (Page 7871)

DPSTATUS - DP_RECV parameters

DPSTATUS

The coding of the DPSTATUS output parameter is different for the DP master mode and DP slave mode.

DP master mode

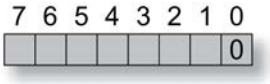


Table 9-366 Meaning of the bits in DPSTATUS in DP master mode

Bit	Meaning
7	not used
6	This bit is not set. Please read the information in the manual as well.
5, 4	Values for DPSTATUS of the DP master: 00 RUN 01 CLEAR 10 STOP (this is now the OFFLINE mode) 11 OFFLINE Please read the information in the manual as well.
3	Value 1: Cyclic synchronization is active
2	Value 0: No new diagnostic data exists Value 1: Evaluation of diagnostic list useful; at least one station has new diagnostic data

Bit	Meaning
1	Value 0: All DP slaves are in the data transfer phase Value 1: Evaluating the station list is useful
0	DP mode Value 0: DP master mode The other bits only have the specified meaning when this bit is not set.

DP slave mode

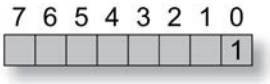


Table 9-367 Meaning of the bits in DPSTATUS in DP slave mode

Bit	Meaning
7-5	not used
4	This bit is not set. Please read the information in the manual as well.
3	This bit is not set. Please read the information in the manual as well.
2	Value 1: DP master 1 is in the CLEAR mode. The DP slave receives the value 0 in the DP data intended for the outputs. This has no effect on the send data.
1	Value 1: The configuration/parameter assignment is not yet completed.
0	Value 1: DP slave mode. **The other bits only have the specified meaning when this bit is set.**

Note

Please note, that DPSTATUS must not be evaluated until the return parameter NDR=1 is set.

See also

DP_RECV (Page 7871)

DP_DIAG

DP_DIAG

Description

The DP_DIAG instruction is used to request diagnostic information. The following types of job are possible:

- Request DP station list
- Request DP diagnostics list;
- Request DP single status;
- Read input/output data of a DP slave acyclically
- Read older DP single diagnostic information
- Read DP status.
- Read DP mode for PLC/CP stop
- Read current status of the DP slave.

Diagnostics data can also be requested for a specific slave by specifying a station address.

To transfer the diagnostic data to the CPU, you should reserve a memory area in the CPU and specify this area in the call. This memory area can be a data block area or a bit memory area. The maximum length of the available memory area must also be specified in the job.

Note

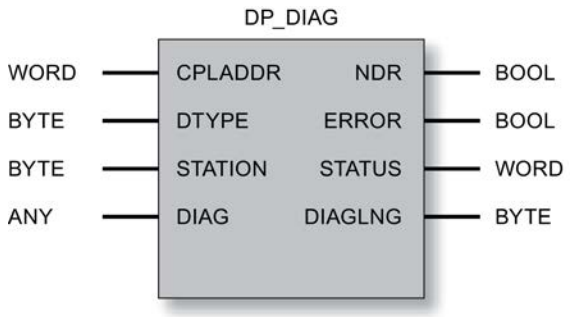
The DP_DIAG instruction is only of practical use in the DP master mode.

Exclusion

As long as this instruction is running, it must not be supplied with new job data.

Exception: Requesting the DP station list or DP diagnostics list.

Call interface



See also

Parameters for DP_DIAG (Page 7878)
Job types for DP_DIAG (Page 7879)
Ring buffer for diagnostics data - DP_DIAG (Page 7881)
Parameters NDR, ERROR, STATUS (Page 7881)

Parameters for DP_DIAG

Explanation of the formal parameters

The following table explains all the formal parameters for the DP_DIAG instruction:

Parameter	Declaration	Data type	Range of values	Description
CPLADDR	INPUT	WORD		Module start address When you configure the CP, the module start address is displayed in the configuration table. Specify this address here.
DTYPE	INPUT	BYTE	0: Station list 1: Diagnostic list 2: Current diagnostic info 3: Older diagnostics info 4: Read status 5: Read status for CPU STOP 6: Read status for CP STOP 7: Read input data (acyclically) 8: Read output data (acyclically) 10: Read current status of the DP slave	Diagnostics type
STATION	INPUT	BYTE		Station address of the DP slave
DIAG	INPUT	ANY (only the following are permitted as VARTYPE: BYTE, WORD and DWORD)	The length must be set from 1 to 240	Specifies the address and length Address of the data area. References the following alternatives: • PI area • Memory bit area • Data block area
NDR	OUTPUT	BOOL	0: - 1: new data	This parameter indicates whether or not new data were accepted. For the meaning in conjunction with the parameters ERROR and STATUS, refer to Parameters NDR, ERROR, STATUS (Page 7881)

Parameter	Declaration	Data type	Range of values	Description
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning in conjunction with the parameters NDR and STATUS, refer to Parameters NDR, ERROR, STATUS (Page 7881)
STATUS	OUTPUT	WORD	See list	Status code For the meaning in conjunction with the parameters NDR and ERROR, refer to Parameters NDR, ERROR, STATUS (Page 7881)
DIAGLNG	OUTPUT	BYTE	See list	This contains the actual length (in bytes) of the data made available by the PROFIBUS CP, regardless of the buffer size specified in the DIAG parameter. The following applies to job types with DTYPE 4, 5 and 6 Here, DIAGLNG always has the value "1". The value returned in the DIAG parameter is not relevant for the evaluation in these cases. In these cases, the relevant value is contained in the STATUS parameter.

See also

DP_DIAG (Page 7877)

Job types for DP_DIAG

Job types

The following overview of the specifications for DTYPE, STATION and DIAGLNG shows the permitted or useful entries.

Table 9-368 Job types for DP_DIAG

DTYPE	Corresponds to job	Parameter STATION	DIAGLNG	Acknowledgement code (contained in the STATUS parameter; shown in Table "DP_DIAG codes")
0	Read DP station list	---	- ignored -	Module start address When you configure the CP, the module start address is displayed in the configuration table. Specify this address here.
1	Read DP diagnostics list	---	- ignored -	The DP diagnostics list informs the CPU program about the DP slaves with new diagnostics data.
2	Read current DP single diagnostics data	1...126	>=6	The current DP single diagnostics informs the CPU program of the current diagnostics data of a DP slave.

DTYPE	Corresponds to job	Parameter STATION	DIAGLNG	Acknowledgement code (contained in the STATUS parameter; shown in Table "DP_DIAG codes")
3	Read older DP single diagnostic information	1...126	>=6	The older DP single diagnostics informs the CPU program of the older diagnostics data of a DP slave. This data is stored on the PROFIBUS CP and read according to the "last in - first out" principle in the ring buffer. The structure of the ring buffer is explained below. If changes occur quickly in the DP slave diagnostic data, this function allows the diagnostic data of a DP slave to be acquired and evaluated in the CPU program of the DP master.
4	Read the operating status requested with DP-CTRL job (CTYPE=4)		=1	With this job, the DP operating status can be read that was set previously with the DP-CTRL job (CTYPE=4). Note: The operating status that is read out does not necessarily match the current operating status. The following statuses are possible: : • RUN • CLEAR • STOP (is mapped to the OFFLINE status) *) • OFFLINE
5	Read DP status for CPU STOP		=1	With this job you can find out the DP status to which the PROFIBUS CP changes if the CPU changes to STOP: • RUN • CLEAR • STOP (is mapped to the OFFLINE status) *) • OFFLINE As default, the PROFIBUS CP changes to the DP status CLEAR if the CPU changes to STOP.
6	Read DP status for CP STOP		=1	With this job you can find out the DP status to which the PROFIBUS CP changes if the CP changes to STOP: • STOP (is mapped to the OFFLINE status) *) • OFFLINE As default, the PROFIBUS CP changes to the DP status OFFLINE if the CP changes to STOP.
7	Read input data	1...126	>=1	With this job, the DP master (class 2) reads the input data of the DP slave. This function is also known as shared input.
8	Read output data	1...126	>=1	With this job, the DP master (class 2) reads the output data of a DP slave. This function is also known as shared output.
10	Read current status of the DP slave	1...126	>=0	With this job, you can read out the current status of the DP slave. The following statuses are possible : • The DP master exchanges data with the DP slave cyclically. • The DP master reads the input data of the DP slave cyclically. • The DP master reads the output data of the DP slaves cyclically. • The DP master is not currently processing this DP slave cyclically.

*) The STOP status is no longer supported on the latest modules (as of module type DA02).

See also

DP_DIAG (Page 7877)

Ring buffer for diagnostics data - DP_DIAG

Ring Buffer for Diagnostic Data

The following diagram illustrates how diagnostic data is read using the "read older DP single diagnostic data" function. The first access reads the **most recent of the older diagnostic data**.

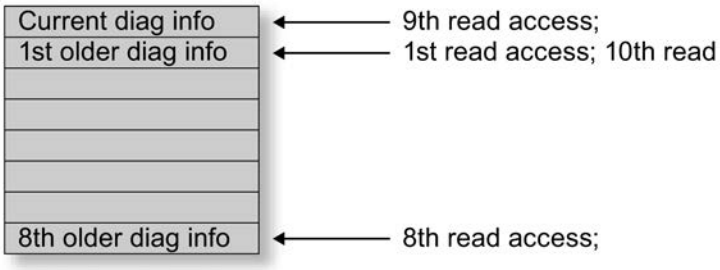


Figure 9-25 Ring Buffer for Diagnostic Data

When the current diagnostic data is read out, the read pointer is reset to the first older diagnostic data.

See also

DP_DIAG (Page 7877)

Parameters NDR, ERROR, STATUS

Condition codes

The following table shows the codes formed by the NDR, ERROR and STATUS parameters that must be evaluated by the user program.

Note

For entries coded with $8Fxx_H$ in STATUS, refer to the information about the output parameter RET_VAL in the descriptions of the referenced system program blocks.

Which system program blocks are used and are relevant for error evaluation, can be queried in STEP 7.

Table 9-369 DP_DIAG codes

NDR	ERROR	STATUS	Possible with DTYPE	Meaning
0	0	8181_H	2-10	Job active. DP master not started due to • CP STOP or • No parameter assignment (occurs here instead of the code 0,1,8183_H).
0	0	8182_H	0	Triggering job pointless. DP master not started due to • CP STOP or • No parameter assignment (occurs here instead of the code 0,1,8183_H).
0	0	8182_H	1	No new diagnostic data exist. DP master not started due to • CP STOP or • No parameter assignment (occurs here instead of the code 0,1,8183_H).
1	0	0000_H	0, 1 and 4-9	Job completed without error. Note: With DTYPE 2, 3 and 10, error-free execution is indicated by a status code other than "0". Below you will see the detailed status codes for error-free execution for the range: 82XXH If an error occurs in execution, you receive status codes in the following ranges: 80XXH, 83XXH, 8FXXH
1	0	8222_H	7,8	Job completed without error. The length of the DP slave data that was read is not the same as the data length expected by the DP master based on the module list of the DP slave in the CP database.
1	0	8227_H	7,8	Job completed without error. Message: No data exists.
1	0	8231_H	4,5,6	Job completed without error. Message: The DP status is already "RUN"
1	0	8232_H	4,5,6	Job completed without error. Message: The DP status is already "CLEAR"
1	0	8233_H	4,5,6	Job completed without error. Message: The DP status is already STOP The STOP status is now the OFFLINE status (here code 8234_H). Please read the information in the manual as well.
1	0	8234_H	4,5,6	Job completed without error. Message: The DP status is already "OFFLINE"
1	0	$823A_H$	2,3,7,8	Job completed without error. Message: 241 or 242 bytes of data were read. 240 bytes of data are available.

NDR	ERROR	STATUS	Possible with DTYPE	Meaning
1	0	8241_H	2,3,10	Job completed without error. Message: The specified DP slave was not configured.
1	0	8243_H	2,3,10	Job completed without error. Message: The module list of the DP slave in the CP database only contains empty modules.
1	0	8245_H	2,3,10	Job completed without error. Message: The DP slave is in the "read input data cyclically" mode.
1	0	8246_H	2,3,10	Job completed without error. Message: The DP slave is in the "read output data cyclically" mode.
1	0	8248_H	2,3,10	Job completed without error. Note: This is the default code for the named diagnostics types if there is no special situation to signal.
1	0	8249_H	2,3,10	Job completed without error. Message: The DP slave is deactivated due to a DP mode change (e.g. CP mode selector set to STOP).
1	0	$824A_H$	2,3,10	Job completed without error. Message: The DP slave is deactivated due to a DP_CTRL job in the CPU program.
0	1	8090_H	0-10	Logical base address of the module is invalid
0	1	$80B0_H$	0-10	The module does not recognize the data record or is changing from RUN --> STOP.
0	1	$80B1_H$	0-10	Specified data record length incorrect
0	1	$80C0_H$	0-10	Data record cannot be read
0	1	$80C1_H$	0-10	The specified data record is being processed
0	1	$80C2_H$	0-10	Too many jobs pending
0	1	$80C3_H$	0-8	Resources (memory) occupied
0	1	$80C4_H$	0-10	Communications error
0	1	$80D2_H$	0-10	Logical base address wrong
0	1	8183_H	0-10	DP master not configured.
0	1	8184_H	0-8	System error or bad parameter type.
0	1	8311_H	>=2	DTYPE parameter outside range of values.
0	1	8313_H	2,3,7,8,10	STATION parameter outside range of values.
0	1	8321_H	>=2	The DP slave is not providing any valid data.
0	1	8326_H	7,8	The DP slave has more than 242 bytes of data available. The PROFIBUS CP supports a maximum of 242 bytes.
0	1	8335_H	7,8	The PROFIBUS CP is in PROFIBUS status: "Station not in ring".
0	1	8341_H	2,3,7,8,10	The specified slave was not configured
0	1	8342_H	7,8	The DP slave with the PROFIBUS address specified in the STATION parameter is not obtainable.
0	1	8349_H	7,8	The DP master is in the OFFLINE mode.
0	1	$8F22_H$	0-10	Area length error reading a parameter (e.g. DB too short)
0	1	$8F23_H$	0-10	Area length error writing a parameter (e.g. DB too short)
0	1	$8F24_H$	0-10	Area error reading a parameter
0	1	$8F25_H$	0-10	Area error writing a parameter

NDR	ERROR	STATUS	Possible with DTYPE	Meaning
0	1	$8F28_H$	0-10	Orientation error when reading a parameter
0	1	$8F29_H$	0-10	Alignment error writing a parameter
0	1	$8F30_H$	0-10	Parameter is in the writeprotected 1st current data block
0	1	$8F31_H$	0-10	Parameter is in the writeprotected 2nd current data block
0	1	$8F32_H$	0-10	The DB number in the parameter is too high
0	1	$8F33_H$	0-10	DB number error
0	1	$8F3A_H$	0-10	Area not loaded (DB)
0	1	$8F42_H$	0-10	Timeout reading a parameter from the I/O area
0	1	$8F43_H$	0-10	Timeout writing a parameter to the I/O area
0	1	$8F44_H$	0-10	Address of the parameter to be read locked in the access track
0	1	$8F45_H$	0-10	Address of the parameter to be written is disabled in the access track
0	1	$8F7F_H$	0-10	Internal error, e.g. illegal ANY reference

See also

DP_DIAG (Page 7877)

DP_CTRL

DP_CTRL

Description

The DP_CTRL instruction transfers control jobs to the PROFIBUS CP. You specify a job field (CONTROL parameter) to specify the control job in greater detail.

The following types of job are possible:

- Global control acyclic/cyclic;
- Delete older diagnostic data;
- Set current DP mode;
- Set DP mode for PLC/CP STOP;
- Read input/output data cyclically;
- Set the operating mode of the DP slave.

There are restrictions relating to the job types listed here (please refer to the information in the manual for the module).

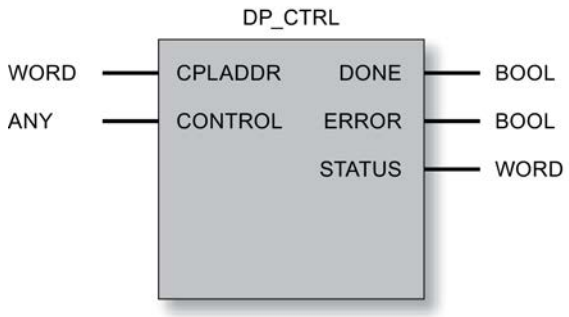
Note

The DP_CTRL instruction is only of practical use in the DP master mode.

Connector

As long as this instruction is running, it must not be supplied with new job data.

Call interface



See also

Parameters for DP_CTRL (Page 7885)
Job types for DP_CTRL (Page 7887)
Command mode and group select - DP_CTRL (Page 7890)
Parameters DONE, ERROR, STATUS (Page 7891)

Parameters for DP_CTRL

Explanation of the formal parameters

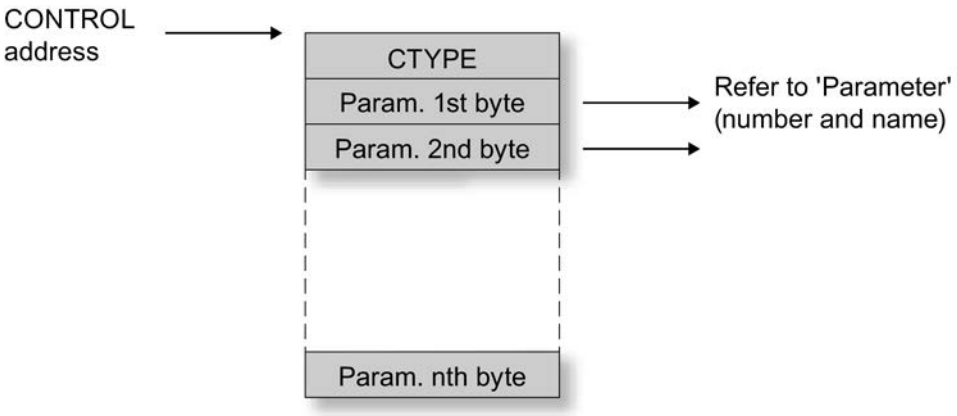
The following table explains all the formal parameters for the DP_CTRL instruction:

Parameter	Declaration	Data type	Range of values	Description
CPLADDR	INPUT	WORD		Module start address When you configure the CP, the module start address is displayed in the configuration table. Specify this address here.
CONTROL	INPUT	ANY (only the following are permitted as VARTYPE: BYTE, WORD and DWORD)	The length must be set from 1 to 240	Specifies the address and length of the CONTROL job field Address of the data area. References the following alternatives: • PI area • Memory bit area • Data block area The length must be at least as long as the number of parameters.

Parameter	Declaration	Data type	Range of values	Description
DONE	OUTPUT	BOOL	0: - 1: Job executed without error.	Indicates whether the job was sent and completed without errors. For the meaning in conjunction with the parameters ERROR and STATUS, refer to Parameters DONE, ERROR, STATUS (Page 7891)
ERROR	OUTPUT	BOOL	0: - 1: Error	Error code For the meaning in conjunction with the DONE and STATUS parameters, refer to Parameters DONE, ERROR, STATUS (Page 7891)
STATUS	OUTPUT	WORD	See following table 'Return Codes'	Status code For the meaning in conjunction with the DONE and ERROR parameters, refer to Parameters DONE, ERROR, STATUS (Page 7891)

Structure of the CONTROL job field

The control job has the following structure:



Example of the job field

With a job field as shown below, a cyclic global control job SYNC and UNFREEZE is sent for group 4 and group 5 without the autoclear option.

DB 14

Byte 0	01H	CTYPE
Byte 1	24H	Command Mode
Byte 2	18H	Group Select
Byte 3	00H	Autoclear

The length in the ANY pointer must be at least 4 (in the example, 30 has been selected).

See also

DP_CTRL (Page 7884)

Job types for DP_CTRL

Job types

Permitted or feasible specifications for the job are shown in the following overview based on the specification for CTYPE and the information in the job field.

CTYPE	Corresponds to job	Parameter in job field		Description
Name	Quantity			
0	Trigger global control	1. byte: command mode 2nd byte: group select (See section following this table.)	2	A single global control job is sent to the DP slaves selected with group select. The command mode parameter specifies the following global control jobs: • SYNC • UNSYNC • FREEZE • UNFREEZE • CLEAR - is not supported (please read the information in the manual as well) It is possible to specify more than one job in the command mode parameter.

CTYPE	Corresponds to job	Parameter in job field		Description
1*)	Trigger cyclic global control	1. byte: command mode 2nd byte: group select 3rd byte: autoclear (See section following this table.)	3	The sending of cyclic global control jobs to the DP slaves selected with group select is triggered on the PROFIBUS CP. The autoclear parameter is only evaluated with the SYNC global control job. If at least one DP slave in the selected group is not in the data transfer phase and autoclear = 1 is set, the CLEAR mode is activated. In other words, the output data of the DP slaves is set to "0". The following global jobs can be activated in the command mode parameter: • SYNC • FREEZE • CLEAR (CLEAR-Bit = 1) - is not supported (please read the information in the manual as well) or deactivated: • UNSYNC • UNFREEZE • UNCLEAR (CLEAR bit = 0) It is possible to specify more than one job in the command mode parameter. An active cyclic global control job can only be terminated by a further global control job (cyclic or acyclic). To terminate the job set in the command mode, the job must be canceled. For example, the SYNC job is canceled by an UNSYNC job.
3	Delete older DP single diagnostic data	1. byte: Slave address 1..126 127 = all slaves	1	The older diagnostic data stored on the PROFIBUS CP is deleted for one or all DP slaves.
4	Set current DP mode	1. byte: RUN = 00_H CLEAR = 01_H OFFLINE = 03_H RUN with AUTOCLEAR = 04_H RUN without AUTOCLEAR = 04_H	1	The DP mode can be set with this job as follows: • RUN • CLEAR • OFFLINE The AUTOCLEAR parameter means that the DP master class 1 changes to the CLEAR status automatically when the following condition is met: at least one of the DP slaves with which the DP master class 1 wants to exchange data is not in the data transfer phase. The RUN without AUTOCLEAR parameter resets AUTOCLEAR. Notes: The STOP = 02_H mode is no longer supported on the later modules (as of module type DA02). STOP = 02_H is mapped to the OFFLINE mode.

CTYPE	Corresponds to job	Parameter in job field		Description
5	Set DP mode for CPU STOP	1. byte: RUN = 00_H CLEAR = 01_H OFFLINE = 03_H	1	This job specifies which DP mode the PROFIBUS CP changes to if the CPU changes to STOP: • RUN • CLEAR • OFFLINE As default, the PROFIBUS CP changes to the DP status CLEAR if the CPU changes to STOP. This mode remains set during a CP mode change from RUN --> STOP --> RUN. Notes: The STOP = 02_H mode is no longer supported on the later modules (as of module type DA02). STOP = 02_H is mapped to the OFFLINE mode.
6	Set DP mode for CP STOP	1. byte: OFFLINE=03_H	1	This job specifies which DP mode the PROFIBUS CP changes to if the CP changes to STOP. • OFFLINE As default, the PROFIBUS CP changes to the DP status OFFLINE if the CP changes to STOP. This mode remains set during a CP mode change from RUN --> STOP --> RUN. Notes: The STOP = 02_H mode is no longer supported on the later modules (as of module type DA02). STOP = 02_H is mapped to the OFFLINE mode.
7 *)	Read input data cyclically (DP master class 2)	1. byte: slave address 1 to 125	1	This job is not supported. Please read the information in the manual as well.
8 *)	Read output data cyclically (DP master class 2)	1. byte: slave address 1 to 125	1	This job is not supported. Please read the information in the manual as well.
9	Terminate cyclic processing of the DP slave by the DP master (class 1, class 2)	1. byte: slave address 1 to 125	1	This job terminates the cyclic reading of the input data or output data of the addressed DP slave or the data transfer (DP master class 1). The DP slave is then no longer processed by the PROFIBUS CP acting as DP master (class 2). **This deactivates the DP slave.**
10	Start cyclic processing as DP master (class 1)	1. byte: slave address 1 to 125	1	The PROFIBUS CP acting as the DP master (class 1) then assigns parameters to the addressed DP slave and starts cyclic data transfer (writing outputs/reading inputs). **This activates the DP slave.**

*) This CTYPE is no longer supported on the latest modules (as of module type DA02).

See also

DP_CTRL (Page 7884)

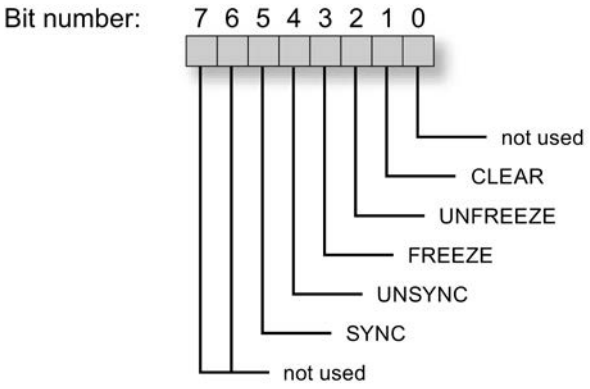
Command mode and group select - DP_CTRL

Structure of command mode

In the command mode parameter, you specify the modes for input and output data for the cyclic and acyclic global control jobs.

The meaning is as follows:

1 = activated
0 = not activated



Structure of group select

In the group select parameter, you specify the group to be addressed by the control job specified in the command mode parameter. The group select parameter occupies the second byte in the control job. Each bit defines a possible DP slave group.

The meaning is as follows:

1 = assigned
0 = not assigned

See also

DP_CTRL (Page 7884)

Parameters DONE, ERROR, STATUS

Condition codes

The following table shows the return codes formed by the DONE, ERROR and STATUS parameters that must be evaluated by the user program.

Note

For entries coded with $8Fxx_H$ in STATUS, refer to the information about the output parameter RET_VAL in the descriptions of the referenced system program blocks.

Which system program blocks are used and are relevant for error evaluation, can be queried in STEP 7.

Table 9-370 DP_CTRL condition codes

DONE	ERROR	STATUS	Possible with CTYPE	Meaning
0	0	8181_H	0..10	Job active. DP master not started due to: • CP STOP or • "no parameter assignment" Note: The code described here occurs instead of one of the codes described later: 0, 1, 8183_H 0, 1, 8333_H 0, 1, 8334_H
1	0	0000_H	0..10	**Job completed without error.**
1	0	8214_H	0,1	**Job completed without error.** Message: Cyclic global control job is sent as acyclic global control job
1	0	8215_H	0,1	**Job completed without error.** The slaves addressed in the selected group are all deactivated.
1	0	8219_H	0,1	**Job completed without error.** An attempt was made to send an already active cyclic global control again. The global control continues unchanged.
1	0	8228_H	0,1	**Job completed without error.** Message: The DP slaves addressed in the selected groups do not have any input modules.
1	0	8229_H	0,1	**Job completed without error.** Message: The DP slaves addressed in the selected groups do not have any output modules.
1	0	8231_H	4,5,6	**Job completed without error.** Message: The DP status is already "RUN"
1	0	8232_H	4,5,6	**Job completed without error.** Message: The DP status is already "CLEAR"
1	0	8233_H	4,5,6	**Job completed without error.** Message: The DP status is already "STOP"

DONE	ERROR	STATUS	Possible with CTYPE	Meaning
1	0	8234_H	4,5,6	**Job completed without error.** Message: The DP status is already "OFFLINE"
1	0	8235_H	4	**Job completed without error.** Message: The DP status is already "RUN" with activated AUTOCLEAR
1	0	8236_H	4	**Job completed without error.** Message: The DP status is already "RUN" with deactivated AUTOCLEAR
1	0	8241_H	7-10	**Job completed without error.** Message: The specified DP slave was not configured.
1	0	8243_H	7-10	**Job completed without error.** Message: The DP slave is already deactivated since the module list of the DP slave in the CP database only contains empty modules.
1	0	8245_H	7-10	**Job completed without error.** Message: The DP slave is already in the "read input data cyclically" mode
1	0	8246_H	7-10	**Job completed without error.** Message: The DP slave is already in the "read output data cyclically" mode
1	0	8248_H	7-10	**Job completed without error.** Message: The module list of the DP slave in the CP database contains input, output, or input/output modules.
1	0	8249_H	7-10	**Job completed without error.** Message: This slave is deactivated due to a change in the DP mode.
1	0	$824A_H$	7-10	**Job completed without error.** Message: The DP slave is already deactivated due to a DP_CTRL job in the CPU program.
0	1	8090_H	0..10	No module with this address exists.
0	1	8091_H	0..10	Logical address not at a double word boundary.
0	1	$80B0_H$	0..10	The module does not recognize the data record.
0	1	$80B1_H$	0..10	The specified data record length is incorrect.
0	1	$80C0_H$	0..10	The data record cannot be read.
0	1	$80C1_H$	0..10	The specified data record is currently being processed.
0	1	$80C2_H$	0..10	There are too many jobs pending.
0	1	$80C3_H$		Resources occupied (memory).
0	1	8183_H	0..10	The DP master is not configured... Note: If the DP master is in "STOP" status, the status 8181_H can also be output.
0	1	8184_H		System error or illegal parameter type...
0	1	8311_H	0..10	CTYPE parameter outside the range of values
0	1	8312_H	0..10	The length of the area in the CONTROL parameter is too short.
0	1	8313_H	3,7,8,9, 10	The slave address parameter is outside the range of values.
0	1	8315_H	0,1	All DP slaves of the group specified in the global control are deactivated (always occurs with an empty group).
0	1	8317_H	8	The length of the configured output data is greater than the configured receive area of the DP slave. Activating the slave mode "Read output data" is not possible.
0	1	8318_H	0,1,4,5,6	The parameter 1st byte of the job data field is outside the range of values. With GLOBAL CONTROL, CLEAR was used with SYNC or a GLOBAL CONTROL with CLEAR set was sent to group 0.

DONE	ERROR	STATUS	Possible with CTYPE	Meaning
0	1	$831A_H$	0,1	At least one DP slave cannot handle FREEZE.
0	1	$831B_H$	0,1	At least one DP slave cannot handle SYNC.
0	1	8333_H	0,1	This job is not permitted in the DP "STOP" mode. Note: If no DP master is configured, the status 8181_H can also be output.
0	1	8334_H	0, 1	This job is not permitted in the DP "OFFLINE" mode. Note: If no DP master is configured, the status 8181_H can also be output.
0	1	8335_H	0, 1	The PROFIBUS CP is in PROFIBUS status: "Station not in ring".
0	1	8339_H	0, 1	At least one DP slave in the selected group is not in the data transfer phase.
0	1	$833C_H$	1	Cyclic global control must not be used in the "PLC <-> CP free running" mode. This error does not occur on the CP 3425 because this mode is not possible with this CP (PBUS data records are always used for data transfer).
0	1	8341_H	7-10	The specified DP slave was not configured.
0	1	8183_H	0..10	DP master not configured.
0	1	8184_H	-	System error or bad parameter type.
0	1	$8F22_H$	0..10	Area length error reading a parameter (e.g. DB too short).
0	1	$8F23_H$	0..10	Area length error writing a parameter.
0	1	$8F24_H$	0..10	Area error reading a parameter.
0	1	$8F25_H$	0..10	Area error writing a parameter.
0	1	$8F28_H$	0..10	Alignment error reading a parameter.
0	1	$8F29_H$	0..10	Alignment error writing a parameter.
0	1	$8F30_H$	0..10	The parameter is in the writeprotected first current data block.
0	1	$8F31_H$	0..10	The parameter is in the writeprotected second current data block.
0	1	$8F32_H$	0..10	Parameter contains a DB number that is too high.
0	1	$8F33_H$	0..10	DB number error.
0	1	$8F3A_H$	0..10	Area not loaded (DB).
0	1	$8F42_H$	0..10	Timeout reading a parameter from the I/O area.
0	1	$8F43_H$	0..10	Timeout writing the parameter to the I/O area.
0	1	$8F44_H$	0..10	Access to a parameter to be read during execution of the instruction is prevented.
0	1	$8F45_H$	0..10	Access to a parameter to be written during execution of the instruction is prevented.
0	1	$8F7F_H$	0..10	Internal error, e.g. illegal ANY reference.
0	1	$80C4_H$	0..10	Communications error (occurs temporarily, it is usually best to repeat the job in the user program).
0	1	$80D2_H$	0..10	Logical base address incorrect.

See also

DP_CTRL (Page 7884)

300C functions

ASCII, 3964(R)

SEND_PTP: Send data (ASCII, 3964(R))

Description

You use the SEND_PTP instruction to send a block of data from a data block.

- The sending process is activated after the block is called and a positive edge is detected at the control input REQ.
- The range of data to be transmitted is specified by SD_1 (DB number and start address), and the length of the data block by LEN.
- For the instruction to process the job, you must call it with R (Reset) = FALSE. With a positive edge at the control input R, an active sending process will be aborted and the instruction reset to the initial state. A canceled job is terminated with an error message (STATUS output).
- Use LADDR to specify the submodule I/O address, which you have defined during hardware configuration.
- DONE will be set to TRUE if the job was completed without errors; ERROR will be set to TRUE if the job was completed with errors.

If the job was processed with DONE = TRUE this means that:

- When using the ASCII driver: The data where transmitted to the communication partner. It is not ensured that all data were received by the communication partner.
- When using the procedure 3964(R): The data have been transmitted to the communication partner and they where acknowledged positively by the partner. It is not ensured that the data were also passed onto the partner CPU.

In STATUS, the CPU indicates the respective event ID as the result of an error or warning. DONE or ERROR/STATUS are also output on RESET of the instruction (R = TRUE).

The binary result BR is reset if an error has occurred. The status of the binary result is TRUE if the instruction was completed without errors.

Note

A parameter check is not included in the instruction. The CPU can go to STOP mode if the parameter assignment is incorrect.

Instance DB

The "SEND_PTP" instruction operates in combination with an instance DB. The DB number is passed on with the call. Access to the data in the instance DB is not allowed.

Parameters

The following table shows the parameters of the instruction "SEND_PTP":

Parameters	Declaration	Data type	Range of values	Default	Description
REQ	Input	BOOL	TRUE/FALSE	FALSE	Control parameter Request: activates data exchange at the positive edge.
R	Input	BOOL	TRUE/FALSE	FALSE	Control parameter Reset. Job is aborted. Transmission is disabled.
LADDR	Input	WORD	CPU-specific	W#16#03FF	The I/O address of the submodule, which you have specified during hardware configuration.
DONE	Output	BOOL	TRUE/FALSE	FALSE	Status parameter (this parameter is only set for the duration of one call): • FALSE: Job not yet started or still running. • TRUE: Job was executed error-free.
ERROR	Output	BOOL	TRUE/FALSE	FALSE	Status parameter (this parameter is only set for the duration of one call): Job complete with error
STATUS	Output	WORD	W#16#0000 to W#16#FFFF	W#16#0000	Status parameter (This parameter is only set for the duration of one call. To display the status, you should copy STATUS to a free data area.) STATUS has the following meaning, dependent on the ERROR bit: • ERROR=FALSE: – STATUS has the value W#16#0000: Neither warning nor error – STATUS has the value <> W#16#0000: Warning, STATUS supplies detailed information. • ERROR = TRUE: An error has occurred, STATUS supplies detailed information on the type of error.

Parameters	Declaration	Data type	Range of values	Default	Description
SD_1	InOut	ANY	CPU-specific	0	Send parameters: Here you enter the following values: • The number of the DB from which the data will be transmitted. • The data byte number, starting from which data will be transmitted. Possible values: 0 to 8190 For example: DB 10 from byte 2 -> DB10.DBB2 Note: Note that for S7-300 CPUs, the parameter SD_1 always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and leads to an error message in the user program.
LEN	InOut	INT	1 to 1024	1	Here you specify the length of the data block that is to be transmitted (length is set here indirectly).

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Data consistency

Data consistency is limited to 206 bytes. If you want consistent data transmission exceeding 206 bytes, you must take the following into account:

Do not write to the currently used section of the send range SD_1 again until the sending process has concluded. This is the case when the status parameter DONE assumes the value TRUE.

RCV_PTP: Fetch data (ASCII, 3964(R))

Description

You use the RCV_PTP instruction to receive data and then store them in a data block.

- After the instruction has been called with the value TRUE at control input EN_R, it is ready to receive data. You cancel the current transmission by setting the signal status of parameter EN_R to FALSE. A canceled job is terminated with an error message (STATUS output). The input is switched off as long as the signal status of parameter EN_R is set to FALSE.
- The receive area is specified by RD_1 (DB number and start address), and the length of the data block by LEN.

- For the instruction to process the job, you must call it with R (Reset) = FALSE. With a positive edge at control input R, the current transmission will be canceled and the instruction reset to its initial state. A canceled receive job is terminated with an error message (STATUS output).
- Use LADDR to specify the submodule I/O address, which you have defined during hardware configuration.
- NDR will be set to TRUE if the job was completed without errors; ERROR will be set to TRUE if the job was completed with errors.
- In STATUS , the CPU indicates the respective event ID as the result of an error or warning.
- NDR or ERROR/STATUS are also output on RESET of the instruction (R = TRUE) (parameter LEN == 16#00).

The binary result BR is reset if an error has occurred. The status of the binary result is TRUE if the instruction was completed without errors.

Note

A parameter check is not included in the instruction. The CPU can go to STOP mode if the parameter assignment is incorrect.

Instance DB

The "RCV_PTP" instruction operates in combination with an instance DB. The DB number is passed on with the call. Access to the data in the instance DB is not allowed.

Parameters

The following table shows the parameters of the instruction "RCV_PTP":

Parameters	Declaration	Data type	Range of values	Default	Description
EN_R	Input	BOOL	TRUE/FALSE	FALSE	Control parameter Enable to receive: Receive enable
R	Input	BOOL	TRUE/FALSE	FALSE	Control parameter Reset: Job is aborted.
LADDR	Input	WORD	CPU-specific	W#16#03FF	The I/O address of the submodule, which you have specified during hardware configuration.
NDR	Output	BOOL	TRUE/FALSE	FALSE	Job done without error, data were accepted • FALSE: Job not started yet or still running • TRUE: Job completed without errors
ERROR	Output	BOOL	TRUE/FALSE	FALSE	Status parameter (this parameter is only set for the duration of one call): Job complete with error

Parameters	Declaration	Data type	Range of values	Default	Description
STATUS	Output	WORD	W#16#0000 to W#16#FFFF	W#16#0000	Status parameter (This parameter is only set for the duration of one call. To display the status, you should copy STATUS to a free data area.) STATUS has the following meaning, dependent on the ERROR bit: • ERROR=FALSE: – STATUS has the value W#16#0000: Neither warning nor error – STATUS has the value <> W#16#0000: Warning, STATUS supplies detailed information. • ERROR = TRUE: An error has occurred, STATUS supplies detailed information on the type of error.
RD_1	InOut	ANY	CPU-specific	0	Receive parameter: Here you specify: • The number of the DB in which the received data are to be stored. • The data byte number starting at which the received data will be stored. Possible values: 0 to 8190 For example: DB 20 starting with byte 5 -> DB20.DBB5 Note: Note that for S7-300 CPUs, the parameter RD_1 always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and leads to an error message in the user program.
LEN	InOut	INT	0 to 1024	0	Output of the data length (number of bytes)

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Data consistency

Data consistency is limited to 206 bytes. If you want consistent data transmission exceeding 206 bytes, you must take the following into account:

Do not access the receive DB again until the data have been completely received (NDR = TRUE). Then, disable the receiving DB (EN_R = FALSE) until you have processed the data.

RES_RCVB: Reset receive buffer (ASCII, 3964(R))

Description

You use the RES_RECV instruction to delete the entire receive buffer of the module. All stored message frames are discarded. A message frame that is incoming when the RES_RCVB instruction is called will be saved.

- The job is activated after the instruction is called and a positive edge is detected at control input REQ. The job can run across multiple calls (program cycles).
- For the instruction to process the job, you must call it with R (Reset) = FALSE. With a positive edge at control input R, the deleting process will be canceled and the instruction reset to its initial state. A canceled job is terminated with an error message (STATUS output).
- Use LADDR to specify the submodule I/O address, which you have defined during hardware configuration.
- DONE will be set to TRUE if the job was completed without errors; ERROR will be set to TRUE if the job was completed with errors.
- In STATUS the CPU indicates the respective event ID as the result of an error or warning.
- DONE or ERROR/STATUS are also output on RESET of the instruction (R = TRUE).

The binary result BR is reset if an error has occurred. The status of the binary result is TRUE if the block was completed without errors.

Note

A parameter check is not included in the instruction. The CPU can go to STOP mode if the parameter assignment is incorrect.

Instance DB

The RES_RCVB instruction operates in combination with an instance DB. The DB number is passed on with the call. Access to the data in the instance DB is not allowed.

Parameter

The following table shows the parameters of the instruction RES_RCVB:

Parameter	Declaration	Data type	Range of values	Default	Description
REQ	Input	BOOL	TRUE/FALSE	FALSE	Control parameter Request: Activates the job on a positive edge.
R	Input	BOOL	TRUE/FALSE	FALSE	Control parameter Reset: Job is aborted.
LADDR	Input	WORD	CPU-specific	W#16#03FF	The I/O address of the submodule, which you have specified during hardware configuration.

Parameter	Declaration	Data type	Range of values	Default	Description
DONE	Output	BOOL	TRUE/FALSE	FALSE	Status parameter (this parameter is only set for the duration of one call): • FALSE: Job not yet started or still running. • TRUE: Job was executed error-free.
ERROR	Output	BOOL	TRUE/FALSE	FALSE	Status parameter (this parameter is only set for the duration of one call): Job complete with error
STATUS	Output	WORD	W#16#0000 to W#16#FFFF	W#16#0000	Status parameter (This parameter is only set for the duration of one call. To display the status, you should copy STATUS to a free data area.) STATUS has the following meaning, dependent on the ERROR bit: • ERROR=FALSE: – STATUS has the value W#16#0000: Neither warning nor error – STATUS has the value <> W#16#0000: Warning, STATUS supplies detailed information. • ERROR = TRUE: An error has occurred, STATUS supplies detailed information on the type of error.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

RK 512

SEND_RK: Send data (RK 512)

Description

You use the SEND_RK to send a block of data from a data block.

- The send job is activated after the instruction is called and a positive edge is detected at control input REQ.
- The range of data to be transmitted is specified by SD_1 (DB number and start address), and the length of the data block by LEN.
- You also specify the receive area on the partner in the instruction. The CPU enters this information in the message frame header and transfers it to the partner.
- The destination is specified by the CPU number in R_CPU (only relevant for multiprocessor communication), the data type in R_TYPE (data blocks (DB) and expanded data blocks (DX)), the data block number in R_DBNO and the offset in R_OFFSET to which the first byte is to be written.

- In R_CF_BYT and R_CF_BIT you declare the interprocessor communication flag byte and bit on the partner CPU.
- With the SYNC_DB parameter, you specify the DB where the data that are common to all instructions used for initialization during startup and synchronization will be stored. The DB number must be identical for all instructions used in your user program.
- For the instruction to process the job, you must call it with R (Reset) = FALSE. With a positive edge at control input R, the active sending process will be aborted and the instruction reset to the initial state. A canceled job is terminated with an error message (STATUS output).
- Use LADDR to specify the submodule I/O address, which you have defined during hardware configuration.
- DONE will be set to TRUE if the job was completed without errors; ERROR will be set to TRUE if the job was completed with errors.
- If the job was processed with DONE = TRUE, the data were sent to the communication partner, the communication partner acknowledged them positively, and they were then passed on to the partner CPU.
- In STATUS, the CPU indicates the respective event ID as the result of an error or warning.
- DONE or ERROR/STATUS are also output on RESET of the instruction (R = TRUE).

The binary result BR is reset if an error has occurred. The status of the binary result is TRUE if the block was completed without errors.

Note

A parameter check is not included in the instruction. The CPU can go to STOP mode if the parameter assignment is incorrect.

Instance DB

The SEND_RK instruction operates in combination with an instance DB. The DB number is passed on with the call. Access to the data in the instance DB is not allowed.

Special features when sending data

Take the following special features into account when "sending data":

- With RK 512 you can only send an even number of data. If you declare an odd length (LEN) of data, an additional fill byte with the value "0" will be appended to the transmitted data.
- In RK 512 you can only declare an even offset. If you declare an odd offset the data are stored in the partner station from the next lower even offset.
  Example: Offset is "7", the data will be stored as of byte "6".

Parameters

The following table shows the parameters of the instruction SEND_RK:

Parameters	Declaration	Data type	Range of values	Default	Description
SYNC_DB	Input	INT	CPU-specific	0	Number of the DB where the common data for the synchronization of the RK instructions are stored (minimum length = 240 bytes).
REQ	Input	BOOL	TRUE/FALSE	FALSE	Control parameter Request: Activates the job on a positive edge.
R	Input	BOOL	TRUE/FALSE	FALSE	Control parameter Reset: Job is aborted.
LADDR	Input	WORD	CPU-specific	W#16#03FF	The I/O address of the submodule, which you have specified during hardware configuration.
R_CPU	Input	INT	0 to 4	1	CPU number of the partner CPU (only for multiprocessor operation)
R_TYPE	Input	CHAR	‘D’, ‘X’	'D'	Address type on the partner CPU (only uppercase allowed) 'D': Data block 'X': Expanded data block
R_DBNO	Input	INT	0 to 255	0	Data block number on the partner CPU
R_OFFSET	Input	INT	0 to 510 (even values only)	0	Data byte number on the partner CPU
R_CF_BYT	Input	INT	0 to 255	255	Communication flag byte on partner CPU (255: Means: without interprocessor communication flag)
R_CF_BIT	Input	INT	0 to 7	0	Communication flag bit on the partner CPU
DONE	Output	BOOL	TRUE/FALSE	FALSE	Status parameter (this parameter is only set for the duration of one call): • FALSE: Job not yet started or still running. • TRUE: Job was executed error-free.
ERROR	Output	BOOL	TRUE/FALSE	FALSE	Status parameter (this parameter is only set for the duration of one call): Job complete with error

Parameters	Declaration	Data type	Range of values	Default	Description
STATUS	Output	WORD	W#16#0000 to W#16#FFFF	W#16#0000	Status parameter (This parameter is only set for the duration of one call. To display the status, you should copy STATUS to a free data area.) STATUS has the following meaning, dependent on the ERROR bit: • ERROR=FALSE: – STATUS has the value W#16#0000: Neither warning nor error – STATUS has the value <> W#16#0000: Warning, STATUS supplies detailed information. • ERROR=TRUE: An error has occurred, STATUS supplies detailed information on the type of error.
SD_1	Input	ANY	CPU-specific	0	Send parameters: Here you specify: • The number of the DB from which the data will be transmitted. • The data byte number, starting from which data will be transmitted. Possible values: 0 to 8190 For example: DB 10 from byte 2 -> DB10.DBB2 Note: Note that for S7-300 CPUs, the parameter SD_1 always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and leads to an error message in the user program.
LEN	Input	INT	1 to 1024	1	Here you specify the length of the data block that is to be transmitted (length is set here indirectly).

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Data in the message frame header

The table below shows the data in the message frame header of the RK 512 message frame.

Source on your S7 automation system (local CPU)	To target, partner CPU	Message frame header, bytes		
		3/4 command type	5/6 D-DBNO[1] / D offset[2]	7/8 number in
Data block	Data block	AD	DB/DW[3]	Words
Data block	Expanded data block	AD	DB/DW[3]	Words

[1] D-DBNO: Destination data block number
[2] D offset: Destination start address
[3] DW: Offset in words

Data consistency

Data consistency is limited to 128 bytes. If you want consistent data transmission exceeding 128 bytes, you must take the following into account:

Do not write to the currently used section of the send range SD_1 again until the sending process has concluded. This is the case when the status parameter DONE assumes the value TRUE.

FETCH_RK: Fetch data (RK 512)

Description

You use the FETCH_RK instruction to retrieve a block of data from the partner and store the data in a data block.

- The sending process is activated after the block is called and a positive edge is detected at the control input REQ.
- The area in which the retrieved data are stored is specified by RD_1 (DB number and start address), and the length of the data block by LEN.
- You also specify the partner area from which the data will be fetched for the instruction. The CPU enters this information in the RK512 message frame header and transfers it to the partner.
- The partner area is determined by the CPU number in R_CPU (only relevant for multiprocessor communication), the data type in R_TYPE (data blocks, expanded data blocks, memory bits, inputs, outputs, counters and timers), the data block number in R_DBNO (only relevant for data blocks and expanded data blocks) and the offset in R_OFFSET, from where the first byte is to be retrieved.
- Use R_CF_BYT and R_CF_BIT to declare the interprocessor communication flag byte and bit on the partner CPU.
- With the SYNC_DB parameter, you specify the DB where the data that are common to all instructions used for initialization during startup and synchronization will be stored. The DB number must be identical for all instructions used in your user program.

- For the instruction to process the job, you must call it with R (Reset) = FALSE. With a positive edge at control input R, the current transmission will be canceled and the instruction reset to its initial state. A canceled job is terminated with an error message (STATUS output).
- Use LADDR to specify the submodule I/O address, which you have defined during hardware configuration.
- DONE will be set to TRUE if the job was completed without errors; ERROR will be set to TRUE if the job was completed with errors.
- In STATUS, the CPU indicates the respective event ID as the result of an error or warning.
- DONE or ERROR/STATUS are also output on RESET of the instruction (R = TRUE).

The binary result BR is reset if an error has occurred. The status of the binary result is TRUE if the instruction was completed without errors.

Note

A parameter check is not included in the instruction. The CPU can go to STOP mode if the parameter assignment is incorrect. If data will be retrieved from your CPU, you must program a SERVE_RK (Page 7909) instruction on your CPU.

Instance DB

The FETCH_RK instruction operates in combination with an instance DB. The DB number is passed on with the call. Access to the data in the instance DB is not allowed.

Special features for (expanded) data blocks

Note the following special features when "retrieving data" from a data block and expanded data blocks:

- With RK 512 you can only send an even number of data. If you declare an odd length (LEN) of data, an additional byte will always be appended to the transmitted data. In the destination DB, however, the correct number of data is always entered.
- In RK 512 you can only declare an even offset. If you declare an odd offset the data are stored in the partner station starting with the next smaller even offset.
  Example: Offset is 7, the data are stored from byte 6.

Special features for timers and counters

When you retrieve times or counters from your communication partner, you must take into account that you must retrieve two bytes for every time or counter. For example, if you want to retrieve 10 counters you must declare a length of 20.

Parameters

The following table shows the parameters of the instruction "FETCH_RK":

Parameters	Declaration	Data type	Range of values	Default	Description
SYNC_DB	Input	INT	CPU-specific	0	Number of the DB where the common data for the synchronization of the RK instructions are stored (minimum length = 240 bytes).
REQ	Input	BOOL	TRUE/FALSE	FALSE	Control parameter Request: Activates the job on a positive edge.
R	Input	BOOL	TRUE/FALSE	FALSE	Control parameter Reset: Job is aborted.
LADDR	Input	WORD	CPU-specific	W#16#03FF	The I/O address of your sub-module that you specified during hardware configuration.
R_CPU	Input	INT	0 to 4	1	CPU number of the partner CPU (only for multiprocessor operation)
R_TYPE	Input	CHAR	'D', 'X', 'M', 'I', 'Q', 'C', 'T'	'D'	Address type on the partner CPU • 'D': Data block • 'X': Expanded data block • 'M': Bit memory • 'I': Inputs • 'Q': Outputs • 'C': Counters • 'T': Timers
R_DBNO	Input	INT	0 to 255	0	Data block number on the partner CPU
R_OFFSET	Input	INT	See table: "Parameters for data source (partner CPU)"	0	Data byte number on the partner CPU
R_CF_BYT	Input	INT	0 to 255	255	Communication flag byte on partner CPU (255: Means: without interprocessor communication flag)
R_CF_BIT	Input	INT	0 to 7	0	Communication flag bit on the partner CPU
DONE	Output	BOOL	TRUE/FALSE	FALSE	Status parameter (this parameter is only set for the duration of one call): • FALSE: Job not yet started or still running. • TRUE: Job was executed error-free.
ERROR	Output	BOOL	TRUE/FALSE	FALSE	Status parameter (this parameter is only set for the duration of one call): Job complete with errors

Parameters	Declaration	Data type	Range of values	Default	Description
STATUS	Output	WORD	W#16#0000 to W#16#FFFF	W#16#0000	Status parameter (This parameter is only set for the duration of one call. To display the status, you should copy STATUS to a free data area.) STATUS has the following meaning, dependent on the ERROR bit: • ERROR=FALSE: – STATUS has the value W#16#0000: Neither warning nor error – STATUS has the value <> W#16#0000: Warning, STATUS supplies detailed information. • ERROR=TRUE: An error has occurred, STATUS supplies detailed information on the type of error.
RD_1	Input	ANY	CPU-specific	0	Receive parameter: Here you specify: • The number of the DB in which the retrieved data are to be stored. • The data byte number starting at which the retrieved data are to be stored. Possible values: 0 to 8190 For example: DB 10 from byte 2 -> DB10.DBB2 Note: Note that for S7-300 CPUs, the parameter RD_1 always requires the full specification of the DB parameters (for example, P#DB13.DBX0.0 byte 100). The omission of an explicit DB number is only permitted for S7-300 CPUs and leads to an error message in the user program.
LEN	Input	INT	1 to 1024	1	Here you specify the length of the data block that is to be retrieved (length is set here indirectly). A length of two bytes must be declared for each timer and counter.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameters for data source (partner CPU)

The data types that can be transmitted are shown in the table below.
The value R_OFFSET is specified by the partner CPU.

Source on the partner CPU	R_TYPE	R_NO	R_OFFSET (in bytes)
Data block	'D'	0 - 255	0 - 510, only even values
Expanded data block	'X'	0 - 255	0 - 510, only even values
Bit memory	’M’	irrelevant	0 - 255
Inputs	'I'	irrelevant	0 - 255
Outputs	’Q’	irrelevant	0 - 255
Counters	'C'	irrelevant	0 - 255
Timers	'T'	irrelevant	0 - 255

Data in the message frame header

The table below shows the data in the message frame header of the RK 512 message frame.

Source on the partner CPU	To the destination, your S7 automation system (local CPU)	Message frame header, bytes		
		3/4 command type	5/6 S-DBNO[1] / S offset[2]	7/8 number in
Data block	Data block	ED	DB/DW	Words
Expanded data block	Data block	EX	DB/DW	Words
Bit memory	Data block	EM	Byte address	Bytes
Inputs	Data block	EE	Byte address	Bytes
Outputs	Data block	IO	Byte address	Bytes
Counters	Data block	EC	Counter number	Words
Timers	Data block	ET	Time number	Words

[1] S-DBNO: Source Data Block Number
[2] S offset: Source start address

Data consistency

Data consistency is limited to 128 bytes. If you want consistent data transmission exceeding 128 bytes, you must take the following into account:

Do not write to the currently used section of the receive range RD_1 again until the transmission process has concluded. This is the case when the status parameter DONE assumes the value TRUE.

SERVE_RK: Receive and provide data (RK 512)

Description

You use the SERVE_RK instruction to receive and provide data.

- Receiving data: The data are stored in the area that is specified by the partner in the RK 512 message frame header. A call of the instruction is required when the communication partner executes a "Send data" job (SEND job).
- Serving data: The data are retrieved from the area that is specified by the partner in the RK 512 message frame header. A call of the instruction is required when the communication partner executes a "Fetch data" job (FETCH job).

After the instruction has been called with the value TRUE at control input EN_R, it is ready to fetch data. You cancel the current transmission by setting the signal status of parameter EN_R to FALSE. A canceled job is terminated with an error message (STATUS output). The input is switched off as long as the signal status of parameter EN_R is set to FALSE.

With the SYNC_DB parameter, you specify the DB where the data that are common to all instructions used for initialization during startup and synchronization will be stored. The DB number must be identical for all instructions used in your user program.

For the instruction to process the job, you must call it with R (Reset) = FALSE. With a positive edge at control input R, the current transmission will be canceled and the instruction reset to its initial state. A canceled job is terminated with an error message (STATUS output).

Use LADDR to specify the submodule I/O address, which you have defined in HW Config.

NDR will be set to TRUE if the job was completed without errors; ERROR will be set to TRUE if the job was completed with errors.

With NDR = TRUE for an instruction call, the CPU indicates in the parameters L_TYPE, L_DBNO and L_OFFSET the area where data were stored or from where they were retrieved. Also shown are the parameters L_CF_BYT and L_CF_BIT of a call as well as the length LEN of the respective job.

In STATUS, the CPU indicates the respective event ID as the result of an error or warning.

NDR or ERROR/STATUS are also output on RESET of the instruction (R = TRUE) (parameter LEN == 16#00).

The binary result BR is reset if an error has occurred. The status of the binary result is TRUE if the block was completed without errors.

Note

A parameter check is not included in the instruction. The CPU can go to STOP mode if the parameter assignment is incorrect.

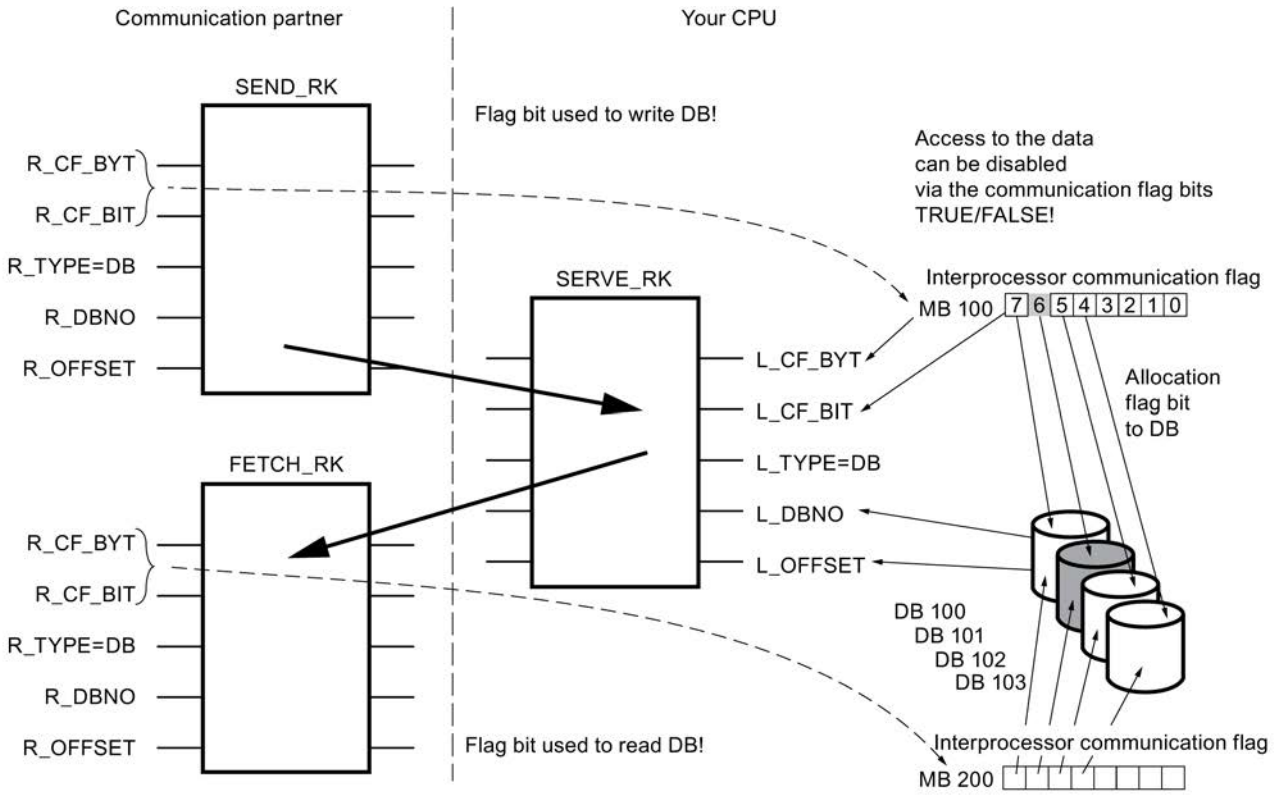
Instance DB

The SERVE_RK instruction operates in combination with an instance DB. The DB number is passed on with the call. Access to the data in the instance DB is not allowed.

Using interprocessor communication flags

You can lock or enable SEND and FETCH jobs of your communication partner via an interprocessor communication flag. Thus, you can prevent overwriting or reading of data that have not yet been processed.

You can specify an interprocessor communication flag for every job.



Example: SEND_RK with interprocessor communication flag:

In this example the communication partner transmits data to DB 101 on your CPU

1. On your CPU, set the interprocessor communication flag 100.6 to FALSE.
2. On your communication partner, specify interprocessor communication flag 100.6 (parameters R_CF_BYT, R_CF_BIT) for the SEND job.
   The interprocessor communication flag is transferred to your CPU in the RK 512 message frame header.
   Before it processes the job, the CPU verifies the interprocessor communication flag that is specified in the RK 512 message frame header. The job is only processed if the interprocessor communication flag is set to FALSE on your CPU. If the interprocessor communication flag is set to TRUE, then the error message "32 hex" will be transmitted to the communication partner in the response message frame.
   After the data have been transferred to DB 101, interprocessor communication flag 100.6 will be set to TRUE on your CPU by SERVE. Also, the communication flag byte and bit on SERVE are output for the duration of one call (if NDR = TRUE).

3. When you evaluate the interprocessor communication flag (interprocessor communication flag 100.6 = TRUE) in your user program you can see whether the job is complete and the transmitted data can be processed.
4. After you have processed the data in your user program you must reset the interprocessor communication flag 100.6 to FALSE. Your communication partner can only execute the job again without error when this has been done.

Parameter

The following table shows the parameters of the instruction "SERVE_RK":

Parameter	Declaration	Data type	Range of values	Default	Description
SYNC_DB	Input	INT	CPU-specific	0	Number of the DB where the common data for the synchronization of the RK instructions are stored (minimum length = 240 bytes).
EN_R	Input	BOOL	TRUE/FALSE	FALSE	Control parameter Enable to receive: Job enable
R	Input	BOOL	TRUE/FALSE	FALSE	Control parameter Reset: Job is aborted.
LADDR	Input	WORD	CPU-specific	W#16#03FF	The submodule I/O address, which you have specified during hardware configuration.
NDR	Output	BOOL	TRUE/FALSE	FALSE	Status parameter New Data Ready (this parameter is only set for the duration of one call): • FALSE: Job not yet started or still running. • TRUE: Job was executed error-free.
ERROR	Output	BOOL	TRUE/FALSE	FALSE	Status parameter (this parameter is only set for the duration of one call): Job complete with errors
STATUS	Output	WORD	W#16#0000 to W#16#FFFF	W#16#0000	Status parameter (This parameter is only set for the duration of one call. To display the status, you should copy STATUS to a free data area.) STATUS has the following meaning, dependent on the ERROR bit: • ERROR=FALSE: – STATUS has the value W#16#0000: Neither warning nor error – STATUS has the value <> W#16#0000: Warning, STATUS supplies detailed information. • ERROR=TRUE: An error has occurred, STATUS supplies detailed information on the type of error.

Parameter	Declaration	Data type	Range of values	Default	Description
L_TYPE	Output	CHAR	'D' 'D', 'M', 'I', 'Q', 'C', 'T',	' '	**Receiving data:** Type of the destination area on the local CPU (only upper case allowed): • 'D': Data block **Serving data:** Type of the source area on the local CPU (only uppercase allowed): • 'D': Data block • 'M': Bit memory • 'I': Inputs • 'Q': Outputs • 'C': Counters • 'T': Timers This parameter is only set for the duration of one call.
L_DBNO	Output	INT	CPU-specific	0	Data block number on local CPU. This parameter is only set for the duration of one call.
L_OFFSET	Output	INT	0 - 510	0	Data byte number on local CPU. This parameter is only set for the duration of one call.
L_CF_BYT	Output	INT	0 to 255	0	Communication flag byte on local CPU. This parameter is only set for the duration of one call. (255: Means: without interprocessor communication flag)
L_CF_BIT	Output	INT	0 to 7	0	interprocessor communication flag on local CPU. This parameter is only set for the duration of one call.
LEN	InOut	INT	0 to 1024	0	Length of message frame, number in bytes (this parameter is only set for the duration of one call).

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Data consistency

Data consistency is limited to 128 bytes. If you want consistent data transmission exceeding 128 bytes, you must take the following into account:

Use the connection memory function. Do not access the data again until they have been completely transmitted (evaluation of the interprocessor communication flag specified for this job; the interprocessor communication flag is active for one instruction call if NDR = TRUE). Do not reset the interprocessor communication flag to FALSE until you have processed the data.

Communication with iSlave / iDevice

I_GET: Read data from a communication partner within the local S7 station

Description

With the "I_GET" instruction, you can read data from a communication partner that is inside the local S7 station. The communication partner can be located in the central controller, an expansion unit, or be part of the distributed configuration. Please make sure that you have assigned distributed communication partners to the local CPU. There is no corresponding instruction on the communication partner.

The receive operation is started after the instruction is called with REQ=1. Following this, you continue to call the instruction until the receipt of the data is indicated by BUSY=0. RET_VAL contains the length of the received block of data in bytes.

Make sure that the receive area defined with the RD parameter (on the receiving CPU) is at least as long as the read area defined by the VAR_ADDR parameter (on the communication partner). The data types of RD and VAR_ADDR must also match.

Parameters

The following table shows the parameters of the "I_GET" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter "request to activate" See also: Auto-Hotspot
CONT	Input	BOOL	I, Q, M, D, L	Control parameter "continue" See also: Auto-Hotspot
IOID	Input	BYTE	I, Q, M, D, L, P or constant	Identifier of the address area on the partner module: • B#16#54 = Peripheral input (PI) • B#16#55 = Peripheral output (PQ) If the module is a mixed module, the area identifier of the lower address must be specified. If the addresses are identical, specify B#16#54.
LADDR	Input	WORD	I, Q, M, D, L, P or constant	Logical address of the partner module For a mixed module, the lower of the two addresses must be specified.
VAR_ADDR	Input	ANY	I, Q, M, D	Reference to the area on the partner CPU from which the data will be read. Choose a data type that is supported by the communication partner.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains the corresponding error code. If no error occurs, RET_VAL contains the length in bytes of the block of data copied to the receive area RD as a positive number.

Parameter	Declaration	Data type	Memory area	Description
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY = 1: Receiving is not yet completed. • BUSY = 0: Receiving is completed or there is no receive operation active.
RD	Output	ANY	I, Q, M, D	Reference to the receive area (receive data area). The following data types are allowed: BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5_TIME, DATE_AND_TIME and arrays of the specified data types with the exception of BOOL. The receive area RD must be at least as long as the read area VAR_ADDR on the communication partner. The data types for RD and VAR_ADDR must also match. The maximum size of the receive area is 76 bytes.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Change to STOP mode

If the CPU changes to STOP mode, the connection established by "I_GET" is terminated. Whether the received data located in a buffer of the operating system are lost depends on the type of subsequent restart performed:

- Following a hot restart (not on the S7-300 and the S7-400H), the data is copied to the area defined by RD.
- Following a warm or cold restart, the data are discarded.

Transition of communication partner to STOP mode

If the CPU of the communication partner changes to STOP mode, this does not affect the data transfer with "I_GET": The data will also be read in STOP mode.

Data consistency

The data are received in a consistent state.

RET_VAL parameter

See also:

- Auto-Hotspot
- Evaluating errors with output parameter RET_VAL (Page 5370)

I_PUT: Write data to a communication partner within the local S7 station

Description

With the "I_PUT" instruction, you write data to a communication partner that is inside the local S7 station. The communication partner can be located in the central controller, an expansion unit, or be part of the distributed configuration. Please make sure that you have assigned distributed communication partners to the local CPU. There is no corresponding instruction on the communication partner.

The send operation is started after the instruction is called with signal level 1 at the REQ control input.

Make sure that the send area defined with the SD parameter (on the sending CPU) is the same length as the receive area defined by the VAR_ADDR parameter (on the communication partner). The data types of SD and VAR_ADDR must also match.

Parameters

The following table shows the parameters of the "I_PUT" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter "request to activate" See also: Auto-Hotspot
CONT	Input	BOOL	I, Q, M, D, L	Control parameter "continue" See also: Auto-Hotspot
IOID	Input	BYTE	I, Q, M, D, L, P or constant	Identifier of the address area on the partner module: • B#16#54 = Peripheral input (PI) • B#16#55 = Peripheral output (PQ) If the module is a mixed module, the area identifier of the lower address must be specified. If the addresses are identical, specify B#16#54.
LADDR	Input	WORD	I, Q, M, D, L, P or constant	Logical address of the partner module. For a mixed module, the lower of the two addresses must be specified.
VAR_ADDR	Input	ANY	I, Q, M, D, L	Reference to the area on the partner CPU to which the data will be written. Choose a data type that is supported by the communication partner.
SD	Input	ANY	I, Q, M, D	Reference to the area on the local CPU that contains the data to be sent. The following data types are allowed: BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME and arrays of the specified data types with the exception of BOOL. SD must be the same length as the VAR_ADDR parameter of the communication partner. The data types for SD and VAR_ADDR must also match. The maximum size of the send area is 76 bytes.

Parameter	Declaration	Data type	Memory area	Description
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains the corresponding error code.
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY = 1: Sending is not yet completed. • BUSY = 0: Sending is completed or there is no send operation active.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Change to STOP mode

If the CPU changes to STOP mode, the connection established by "I_PUT" is terminated. Data can no longer be sent. If the send data have already been copied to the internal buffer when the CPU changes mode, the contents of the buffer are discarded.

Transition of communication partner to STOP mode

If the CPU of the communication partner changes to STOP mode, this does not affect the data transfer with "I_PUT". The transmitted data will be written nevertheless.

Data consistency

The data are sent in a consistent state.

RET_VAL parameter

See also:

- Auto-Hotspot
- Evaluating errors with output parameter RET_VAL (Page 5370)

I_ABORT: Abort an existing connection to a communication partner within the local S7 station

Description

With the "I_ABORT" instruction, you abort a connection that was established with an "I_GET (Page 7913)" or "I_PUT (Page 7915)" instruction to a communication partner that is within the local S7 station. If the job belonging to "I_GET (Page 7913)" or "I_PUT (Page 7915)" is completed (BUSY = 0), the connection resources used at both ends are released after "I_ABORT" is called.

If the job belonging to "I_GET (Page 7913)" or "I_PUT (Page 7915)" is not yet completed (BUSY = 1), you will need to call the relevant instruction again with REQ = 0 and CONT = 0 after the connection has been aborted and then wait for BUSY = 0. Only then are all the connection resources released again.

You can only call "I_ABORT" at the end that is executing "I_PUT (Page 7915)" or "I_GET (Page 7913)" (in other words, at the client end).

The connection abort is activated by calling the instruction with REQ=1.

Parameters

The following table shows the parameters of the "I_ABORT" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter "request to activate" See also: Auto-Hotspot
IOID	Input	BYTE	I, Q, M, D, L, P or constant	Identifier of the address area on the partner module: • B#16#54 = Peripheral input (PI) • B#16#55 = Peripheral output (PQ) If the module is a mixed module, the area identifier of the lower address must be specified. If the addresses are identical, specify B#16#54.
LADDR	Input	WORD	I, Q, M, D, L, P or constant	Logical address of the partner module. For a mixed module, the lower of the two addresses must be specified.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains the corresponding error code.
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY = 1: The connection abort is not yet completed. • BUSY = 0: The connection abort is completed.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Change to STOP mode

If the CPU changes to STOP mode, the connection abort started with "I_ABORT" will be completed.

Transition of communication partner to STOP mode

If the CPU of the communication partner changes to STOP mode, this does not affect the connection abort with "I_ABORT". The connection is aborted.

RET_VAL parameter

See also:

- Auto-Hotspot
- Evaluating errors with output parameter RET_VAL (Page 5370)

PROFINET CBA

Information regarding PN_IN, PN_OUT, and PN_DP

Note

By default, the operating system updates both the PROFINET CBA component as well as the DP interconnections at the cycle control point. If, however, you have disabled these automatic updates in the configuration (for example to improve the time response of the CPU), you will have to perform the updates yourself. This is done by calling the "PN_IN (Page 7920)", "PN_OUT (Page 7921)" and "PN_DP (Page 7922)" instructions at appropriate times. These updates can only be disabled as a group and not individually in the configuration.

Shadow memory

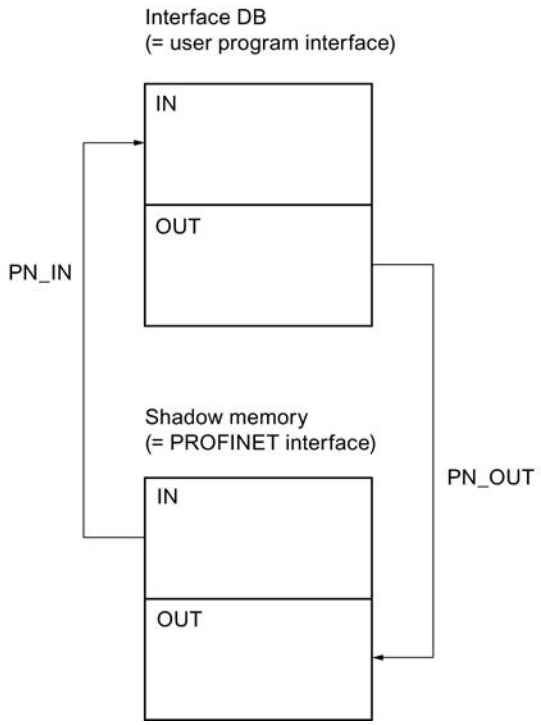
The interface DB is the user program interface for the PROFINET CBA component. To ensure that the inputs and outputs are consistent when the associated program sections are running, each interface DB has an identically structured memory area that is managed by the operating system. This area is called "shadow memory". In your program, you can only access the interface DB, and other (external) PROFINET CBA components can only access the shadow memory. This arrangement prevents conflicts in accessing the inputs and outputs of the interface DB.

Updating the PROFINET CBA component

The data consistency provided by the use of the shadow memory means that updating the PROFINET CBA component takes place in the following two steps:

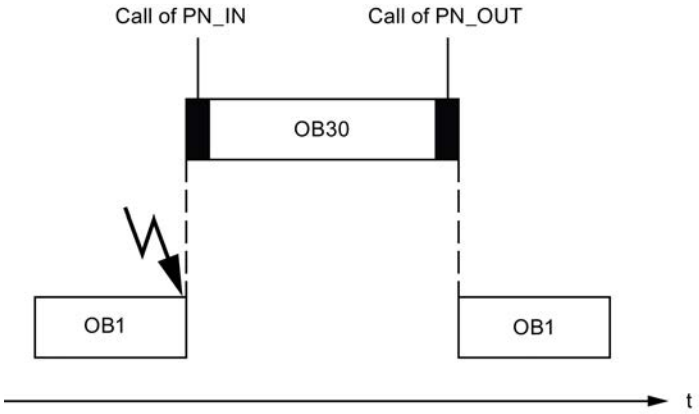
- The inputs in the shadow memory are copied to the inputs of the interface DB before your program for the PROFINET CBA component starts
- The outputs of the interface DB are copied to the outputs of the shadow memory after your program for the PROFINET CBA component has run

The PROFINET CBA component is updated on CPUs with an integrated PROFINET interface either by the operating system or by the PN_IN (Page 7920)" and "PN_OUT (Page 7921)" instructions (depending on how you configured the PROFINET CBA component when you created it). The following schematic diagram illustrates the update by the "PN_IN (Page 7920)" and "PN_OUT (Page 7921)" instructions.



The system-side update always occurs at the cycle control point.

If, on the other hand, you perform the update using the "PN_IN (Page 7920)" and "PN_OUT (Page 7921)" instructions, you will need to call "PN_IN (Page 7920)" at the start of the OB containing the program associated with the PROFINET CBA component and "PN_OUT (Page 7921)" at the end of this OB. This procedure is shown using OB 30 as an example in the following Fig.



Note

(More than one PROFINET interface DB on a device):

With multifunction components, it is possible to update all interface DBs with one job (DBNO=0) – similar to the update at the cycle control point but triggered by an instruction. If an error occurs when updating an interface DB, the updating of the other interface DBs continues. A negative RET_VAL relates to one of the updated interface DBs.

Updating the DP interconnections

The DP interconnections are updated either by the operating system or by the "PN_DP (Page 7922)" instruction, depending on how you configured the PROFINET CBA component when you created it.

PN_IN: Update the inputs of the user program interface

Description

With the "PN_IN" instruction, you copy the input data in PROFINET CBA from the shadow memory of the PROFINET CBA component to the associated interface DB. After conclusion of the instruction, your application has the current input data available to it.

Parameters

The following table shows the parameters of the "PN_IN" instruction:

Parameters	Declaration	Data type	Memory area	Description
DBNO	Input	WORD	I, Q, M, D, L or constant	DB no. of the interface DB (DBNO=0: Updating all PROFINET CBA interface DBs)
RET_VAL	Return	INT	I, Q, M, D, L	Error information

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

RET_VAL parameter

Error code (W#16#...)	Explanation
0000	No error occurred.
8001	PROFINET CBA configuration is not available or invalid.
8002	The DB number does not match the one in the component configuration.
8004	The DB number matches the one in the component configuration, but the DB is not yet loaded.

Error code (W#16#...)	Explanation
8005	The interface DB was compiled with the keyword UNLINKED, in other words, it is only in the load memory but not in the work memory.
8006	The interface DB is write-protected in the CPU.
80B1	Length error when reading or writing. The component configuration does not fit the loaded DB.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

PN_OUT: Update the outputs of the user program interface

Description

With the instruction "PN_OUT", you copy the output data generated in your application from the interface DB of the PROFINET CBA component to the associated shadow memory. After conclusion of the instruction, the other PROFINET CBA components have the current output data available to them.

Parameters

The following table shows the parameters of the "PN_OUT" instruction:

Parameters	Declaration	Data type	Memory area	Description
DBNO	Input	WORD	I, Q, M, D, L or constant	DB no. of the interface DB (DBNO=0: Updating of all PROFINET interface DBs)
RET_VAL	Return	INT	I, Q, M, D, L	Error information

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

RET_VAL parameter

Error code (W#16#...)	Explanation
0000	No error occurred.
8001	PROFINET CBA configuration is not available or invalid.
8002	The DB number does not match the one in the component configuration.
8004	The DB number matches the one in the component configuration, but the DB is not yet loaded.
8005	The interface DB was compiled with the keyword UNLINKED, in other words, it is only in the load memory but not in the work memory.
8006	The interface DB is write-protected in the CPU.

Error code (W#16#...)	Explanation
80B1	Length error when reading or writing. The component configuration does not fit the loaded DB.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

PN_DP: Update DP interconnections

Description

With the "PN_DP" instruction you update all

- interconnections between PROFINET CBA components on the local PROFIBUS and
- interconnections with cyclic transmission between PROFINET CBA components on the local PROFIBUS and external PROFINET CBA components. The interconnections apply between networks (between Industrial Ethernet and PROFIBUS DP).

Functional description

"PN_DP" is an asynchronous instruction, its processing extends, if necessary, over multiple calls. To start updating the DP interconnections, call "PN_DP" with REQ=1.

The output parameters RET_VAL and BUSY indicate the status of the job.

See also: Meaning of the parameters REQ, RET_VAL and BUSY with asynchronous instructions (Page 5367).

Parameters

The following table shows the parameters of the "PN_DP" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	REQ = 1: Initiate update of the DP interconnections
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains an error code.
BUSY	Output	BOOL	I, Q, M, D, L	BUSY=1: The updating of the DP interconnection is not yet complete.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Parameter RET_VAL

Error code (W#16#...)	Explanation
0000	Job completed without error.
7000	First call with REQ = 0: Update of DP interconnections not initiated. BUSY has the value "0".
7001	(only relevant for S7-400:) First call with REQ = 1: BUSY has the value "1".
7002	(only relevant for S7-400:) Intermediate call (REQ irrelevant). The updating of the DP interconnections is not yet completed. BUSY has the value "1".
8001	PROFINET CBA configuration is not available or invalid.
8095	You have initiated another update of the DP interconnections in a higher priority class. However, the update in the priority class with lower priority (by the operating system or a "PN_DP" processing) is still running.
8xyy	General error information See also: Evaluating errors with output parameter RET_VAL (Page 5370)

MPI communication

X_SEND: Send data to a communication partner outside the local S7 station

Description

With the "X_SEND" instruction, you send data to a communication partner outside the local S7 station. The data is received on the communication partner using the "X_RCV (Page 7924)" instruction. The send operation is started after the instruction is called with REQ=1.

Make sure that the send area defined by the SD parameter (on the sending CPU) is smaller than or the same size as the receive area defined by the RD parameter (on the communication partner). If SD is data type BOOL, RD must also be data type BOOL.

Parameters

The following table shows the parameters of the "X_SEND" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter "request to activate" See also: Auto-Hotspot
CONT	Input	BOOL	I, Q, M, D, L	Control parameter "continue" See also: Auto-Hotspot
DEST_ID	Input	WORD	I, Q, M, D, L or constant	Addressing parameter "destination ID". It contains the configured MPI address of the communication partner.
REQ_ID	Input	DWORD	I, Q, M, D, L or constant	Job identifier to identify the data on the communication partner.

Parameter	Declaration	Data type	Memory area	Description
SD	Input	ANY	I, Q, M, D	Reference to the send area. The following data types are allowed: BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5TIME, DATE_AND_TIME and arrays of the specified data types with the exception of BOOL. The maximum size of the send area is 76 bytes.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains the corresponding error code.
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY = 1: Sending is not yet completed. • BUSY = 0: Sending is completed or there is no send operation active.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Data consistency

The data are sent in a consistent state.

Parameter RET_VAL

See also:

- Auto-Hotspot
- Evaluating errors with output parameter RET_VAL (Page 5370)

X_RCV: Receive data from a communication partner outside the local S7 station

Description

With the "X_RCV" instruction, you receive the data sent by one or more communication partners outside the local S7 station using the "X_SEND (Page 7923)" instruction.

With the "X_RCV" instruction

- you can check whether data are available at the current time. The data were entered in an internal queue by the operating system.
- You can copy the oldest block of data from the queue to a selected receive area.

Parameters

The following table shows the parameters of the "X_RCV" instruction:

Parameter	Declaration	Data type	Memory area	Description
EN_DT	Input	BOOL	I, Q, M, D, L	Control parameter "enable data transfer". Use the value 0 to verify whether at least one data block is available. The value 1 causes the oldest data block in the queue to copied into the work memory area that you have specified with RD.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains the corresponding error code. If no error occurs, RET_VAL contains • if EN_DT=0/1 and NDA = 0: W#16#7000. In this case, no block of data is in the queue. • If EN_DT=0 and NDA=1 the length of the oldest block of data entered in the queue as a positive number in bytes. • If EN_DT=1 and NDA=1 the length of the oldest block of data copied to the receive area RD as a positive number in bytes.
REQ_ID	Output	DWORD	I, Q, M, D, L	Job identifier of the "X_SEND (Page 7923)" instruction whose sent data is first in the queue, in other words, the oldest data in the queue. If there is no block of data in the queue, REQ_ID has the value "0".
NDA	Output	BOOL	I, Q, M, D, L	Status parameter "new data arrived". NDA=0: • There is no block of data in the queue. NDA=1: • The queue contains at least one block of data. ("X_RCV" call with EN_DT=0). • The oldest block of data in the queue was copied to the user program ("X_RCV" call with EN_DT=1).
RD	Output	ANY	I, Q, M, D	Reference to the receive area (receive data area). The following data types are allowed: BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5_TIME, DATE_AND_TIME and arrays of the specified data types with the exception of BOOL. If you want to discard the oldest block of data in the queue, assign RD the value NIL. .The maximum size of the receive area is 76 bytes.

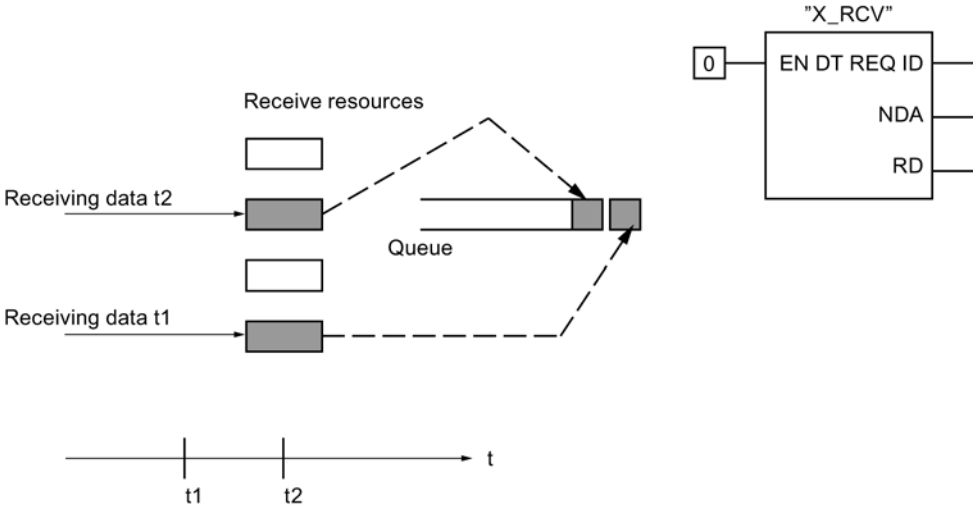
For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Show data reception with EN_DT=0

As soon as data from a communication partner arrive, they are entered in the queue by the operating system in the order in which they are received.

If you want to check whether at least one block of data is in the queue, call "X_RCV" with EN_DT=0 and evaluate the output parameter NDA as follows:

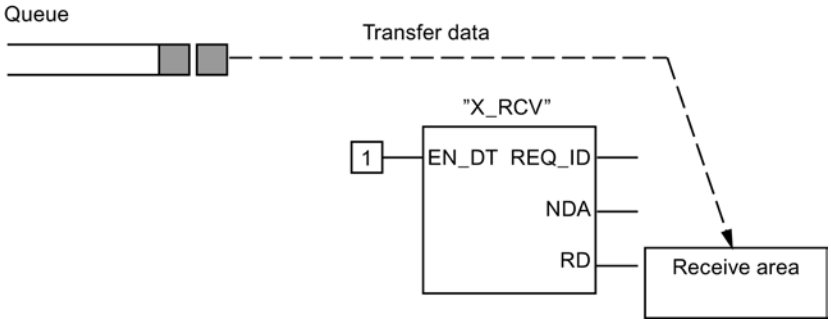
- NDA=0 means that the queue does not contain a block of data. REQ_ID is irrelevant, RET_VAL contains W#16#7000.
- NDA=1 means that there is at least one block of data that can be fetched in the queue . In this case, you also evaluate the output parameters RET_VAL and possibly REQ_ID. RET_VAL contains the length of the data block in bytes, REQ_ID has the job ID of the send block. If there are several blocks of data in the queue, REQ_ID and RET_VAL belong to the oldest block of data in the queue.



Accepting data in the receive area with EN_DT=1

If you call "X_RCV" with EN_DT=1 , the oldest available block of data in the queue is copied to the area of the work memory specified by RD. RD must be greater than or equal to the send area of the associated "X_SEND (Page 7923)" instruction that is defined by the SD parameter. If its input parameter SD is data type BOOL, RD must also be data type BOOL. If you want to store the received data in different areas, you can determine REQ_ID (instruction call with EN_DT=0) and select RD (with EN_DT=1) accordingly in the subsequent call. If an error

occurred when copying, then the length of the copied block of data in bytes is in RET_VAL and a positive acknowledgment will be sent to the sender.



Discarding data

If you do not want to accept the data, assign the value NIL to RD. In this case, the sender receives a negative acknowledgment (RET_VAL of the corresponding "X_SEND (Page 7923)" instruction has the value W#16#80B8) and "0" is entered in RET_VAL of the "X_RCV" instruction.

Data consistency

After a call with EN_DT=1 and RETVAL=W#16#00xy, there is new data in the receive area RD. A further block call can overwrite this data. To prevent this, do not call "X_RCV" with the same receive area RD until you have evaluated the received data.

Change to STOP mode

If the CPU changes to STOP mode

- All newly arriving jobs are acknowledged negatively.
- Applies for the jobs that have already arrived: All jobs that have arrived and are in the queue are acknowledged negatively.
  - Following a warm or cold restart, the data are discarded.
  - If there is a subsequent hot restart (not on an S7-300 or S7-400H), the data block belonging to the oldest job is entered in the user program, if it was queried before the transition to STOP mode (by calling "X_RCV" with EN_DT=0). Otherwise it is discarded. All other blocks of data are discarded.

Connection abort

If the connection is terminated a job belonging to the connection that is already in the queue is discarded.

Exception: If this job is the oldest in the queue and you have already detected its presence by calling X_RCV" with EN_DT=0, you can enter it in the receive area with EN_DT=1.

Parameter RET_VAL

See also:

- Auto-Hotspot
- Evaluating errors with output parameter RET_VAL (Page 5370)

X_GET: Read data from a communication partner outside the local S7 station

Description

With the "X_GET" instruction, you can read data from a communication partner that is inside the local S7 station. There is no corresponding instruction on the communication partner.

The read operation is started after the instruction is called with REQ=1. Following this, you continue to call "X_GET" until the receipt of the data is indicated with BUSY=0. RET_VAL contains the length of the received block of data in bytes.

Make sure that the receive area defined with the RD parameter (on the receiving CPU) is at least as long as the read area defined by the VAR_ADDR parameter (on the communication partner). The data types of RD and VAR_ADDR must also match.

Parameters

The following table shows the parameters of the "X_GET" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter "request to activate" See also: Auto-Hotspot
CONT	Input	BOOL	I, Q, M, D, L	Control parameter "continue" See also: Auto-Hotspot
DEST_ID	Input	WORD	I, Q, M, D, L or constant	Addressing parameter "destination ID". It contains the MPI address of the communication partner. You have configured this.
VAR_ADDR	Input	ANY	I, Q, M, D	Reference to the area on the partner CPU from which the data will be read. Choose a data type that is supported by the communication partner.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains the corresponding error code. If no error occurs, RET_VAL contains the length in bytes of the block of data copied to the receive area RD as a positive number.

Parameter	Declaration	Data type	Memory area	Description
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY = 1: Receiving is not yet completed. • BUSY = 0: Receiving is completed or there is no receive operation active.
RD	Output	ANY	I, Q, M, D	Reference to the receive area (receive data area). The following data types are allowed: BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5_TIME, DATE_AND_TIME and arrays of the specified data types with the exception of BOOL. The receive area RD must be at least as long as the read area VAR_ADDR on the communication partner. The data types for RD and VAR_ADDR must also match. The maximum size of the receive area is 76 bytes.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Change to STOP mode

If the CPU changes to STOP mode, the connection established by "X_GET" is terminated. Whether the received data located in a buffer of the operating system are lost depends on the type of subsequent restart performed:

- Following a hot restart (not on the S7-300 and the S7-400H), the data is copied to the area defined by RD.
- Following a warm or cold restart, the data are discarded.

Transition of communication partner to STOP mode

If the CPU of the communication partner changes to STOP mode, this does not affect the data transfer with "X_GET": The data will also be read in STOP mode.

Data consistency

The data are received in a consistent state.

Parameter RET_VAL

See also:

- Auto-Hotspot
- Evaluating errors with output parameter RET_VAL (Page 5370)

X_PUT: Write data to a communication partner outside the local S7 station

Description

With the "X_PUT" instruction, you write data to a communication partner that is outside the local S7 station. There is no corresponding instruction on the communication partner.

The read operation is started after the instruction is called with REQ=1. Following this, you continue to call "X_PUT" until the receipt of the data is indicated with BUSY=0.

Make sure that the send area defined with the SD parameter (on the sending CPU) is the same length as the receive area defined by the VAR_ADDR parameter (on the communication partner). The data types of SD and VAR_ADDR must also match.

Parameters

The following table shows the parameters of the "X_PUT" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter "request to activate" See also: Auto-Hotspot
CONT	Input	BOOL	I, Q, M, D, L	Control parameter "continue" See also: Auto-Hotspot.
DEST_ID	Input	WORD	I, Q, M, D, L or constant	Addressing parameter "destination ID". It contains the MPI address of the communication partner. You have configured this.
VAR_ADDR	Input	ANY	I, Q, M, D	Reference to the area on the partner CPU to which the data will be written. Choose a data type that is supported by the communication partner.
SD	Input	ANY	I, Q, M, D	Reference to the area on the local CPU that contains the data to be sent. The following data types are allowed: BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME,S5_TIME, DATE_AND_TIME and arrays of the specified data types with the exception of BOOL. SD must be the same length as the VAR_ADDR parameter of the communication partner. The data types for SD and VAR_ADDR must also match. The maximum size of the send area is 76 bytes.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains the corresponding error code.
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY = 1: Sending is not yet completed. • BUSY = 0: Sending is completed or there is no send operation active.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Change to STOP mode

If the CPU changes to STOP mode, the connection established by "X_PUT" is terminated. Data can no longer be sent. If the send data have already been copied to the internal buffer when the CPU changes mode, the contents of the buffer are discarded.

Transition of communication partner to STOP mode

If the CPU of the communication partner changes to STOP mode, this does not affect the data transfer with "X_PUT": The transmitted data will be written nevertheless.

Data consistency

The data are sent in a consistent state.

Parameter RET_VAL

See also:

- Auto-Hotspot
- Evaluating errors with output parameter RET_VAL (Page 5370)

X_ABORT: Abort an existing connection to a communication partner outside the local S7 station

Description

With the "X_ABORT" instruction, you abort a connection that was established with an "X_SEND (Page 7923)", "X_GET (Page 7928)" or "X_PUT (Page 7930)" instruction to a communication partner that is outside the local S7 station.

- If the job belonging to "X_SEND (Page 7923)", "X_GET (Page 7928)" or "X_PUT (Page 7930)" is completed (BUSY=0), the connection resources used at both ends are released again after the "X_ABORT" instruction is called.
- If the job belonging to "X_SEND (Page 7923)", "X_GET (Page 7928)" or "X_PUT (Page 7930)" is not yet completed (BUSY=1), you will need to call the relevant instruction again with REQ=0 and CONT=0 after the connection has aborted and then wait for BUSY=0. Only then are all the connection resources released again.

You can only call "X_ABORT" at the end that is executing "X_SEND (Page 7923)", "X_PUT (Page 7930)" or "X_GET (Page 7928)". The connection abort is activated by calling the instruction with REQ=1.

Parameters

The following table shows the parameters of the "X_ABORT" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L	Control parameter "request to activate" See also: Auto-Hotspot
DEST_ID	Input	WORD	I, Q, M, D, L or constant	Addressing parameter "destination ID". It contains the MPI address of the communication partner. You have configured this.
RET_VAL	Return	INT	I, Q, M, D, L	If an error occurs while the instruction is being executed, the return value contains the corresponding error code.
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY = 1: The connection abort is not yet completed. • BUSY = 0: The connection abort is completed.

For additional information on valid data types, refer to "Overview of the valid data types (Page 2236)".

Change to STOP mode

If the CPU changes to STOP mode, the connection abort started with "X_ABORT" will be completed.

Transition of communication partner to STOP mode

If the CPU of the communication partner changes to STOP mode, this does not affect the connection abort with "X_ABORT". The connection is aborted.

Parameter RET_VAL

See also:

- Auto-Hotspot
- Evaluating errors with output parameter RET_VAL (Page 5370)

TeleService

PG_DIAL: Establish remote connection to programming device/PC

Description of PG_DIAL

Description

The "PG_DIAL" instruction transfers a telephone number and an event ID to a TS Adapter. Using the specified telephone number, the TS Adapter establishes a remote connection to a programming device/PC. The event ID is transferred to the programming device/PC and passed to a waiting application.

If the event ID (EVENT_ID (Page 7935) parameter) was successfully transferred to the application, the TS Adapter receives an acknowledgement that is passed to the "PG_DIAL" instruction. The processing of "PG_DIAL" then is terminated and the status transferred to the caller of "PG_DIAL". The application on the programming device/PC is responsible for terminating the remote connection.

If an error occurs during the processing that leads to the processing aborting, the error code is transferred to the caller of "PG_DIAL". Any established remote connection will be terminated by the TS Adapter.

Calling "PG_DIAL"

You can call the "PG_DIAL" instruction in the cycle or a time-controlled program. An instance DB must be specified when "PG_DIAL" is called.

"PG_DIAL" must be called more than once to process a block function. This means that calling "PG_DIAL" in a "queue" is pointless. The end of the block function is indicated by BUSY = 0.

Communication connection abort

If the CPU changes to STOP mode while "PG_DIAL" is active, the communication connection to the TS Adapter is aborted.

The communication connection to the TS Adapter is also lost if serious communication problems arise on the MPI bus or, for example, the CPU is switched off. In these cases the telephone number and event ID that were already transferred to the TS Adapter and not discarded by the adapter. The TS Adapter establishes the remote connection to the PG/PC and transfers the event ID. However, the acknowledgement that is transferred from the programming device/PC to the TS Adapter, is discarded by the TS Adapter.

If a user program attempts to establish a remote connection while the TS Adapter is in the status described above, "PG_DIAL" is terminated with the return value W#16#B10A. The user program can later repeat the establishment of a remote connection. You also receive the return

value W#16#B10A, if various CPUs attempt to establish a remote connection at the same time via the same TS Adapter.

NOTICE
Changing user programs You can change the parts of your user program that directly affect calls of "PG_DIAL" only when the CPU is in STOP mode. This relates, in particular, to deleting and replacing program blocks that contain "PG_DIAL" calls. Ignoring this restriction can tie up connection resources. The status of the automation system relating to the communication instructions for non-configured S7 connections may become undefined. A warm or cold restart of the CPU is required after the changes are transferred.

Data consistency

The input parameters of the function block are copied to in internal buffer when "PG_DIAL" is first called. Do not change this data before the first call is completed (return value W#16#7001), otherwise inconsistent data could be transferred.

Parameters

The following table shows the parameters of the "PG_DIAL" instruction:

Parameters	Declaration	Data type	Memory area	Description
MPI_TS_ADAPTER	Input	INT	I, Q, M, D, L or constant	MPI address of the TS Adapter
PHONE_NO (Page 7935)	Input	ANY	D	Reference to a data string with a maximum length of 31 characters
EVENT_ID (Page 7935)	Input	ANY	I, Q, M, D	Reference to a byte array with a maximum length of 16 characters
BUSY (Page 7936)	Output	BOOL	I, Q, M, D, L	• BUSY=1: The establishment of the remote connection is not yet completed. • BUSY=0: The processing of "PG_DIAL" was stopped.
STATUS (Page 7936)	Output	INT	I, Q, M, D, L	Return value of "PG_DIAL"

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

Virtual block parameters EN and ENO and the BR bit

The virtual block parameters EN and ENO occur only when "PG_DIAL" is included in the LAD or FBD representation of the STEP 7 editor. They are closely linked with the BR bit (binary result) of the status word.

- Input parameters EN
  The status of the result of logic operation (RLO) is saved to the BR bit during the block call.
- Output parameters ENO
  If "PG_DIAL" is processed error-free, the status of the BR bit that existed when the block was called is restored again at the end of the block call.
  If an error message is output via the output parameter STATUS, the BR bit is set to "0" immediately before exiting "PG_DIAL".

See also

Establishing a connection from and to remote systens (PG-AS-remote coupling) (Page 10966)

PHONE_NO parameter

Description

The input parameter PHONE_NO specifies the telephone number to which a remote connection will be established. Enter the complete telephone number including country code, area code and subscriber number. The string is transferred unchanged to the modem. If you use characters that are not numerals, make sure that the modem used supports these characters.

The TS Adapter uses the following parameter values to establish a remote connection:

- Location: Dial procedure, access code
- Call settings: "Wait for dial tone before dialing", "Number of redial attempts" and "Redial after"

EVENT_ID parameter

Description

The EVENT_ID input parameter specifies the event ID. This is transferred transparently from the user program of the automation system to the user program on the PG/PC by "PG_DIAL", the TS Adapter and the TeleService application on the PG/PC. You can structure the event ID as required and transfer any information from the automation system to the PG/PC.

If a length less than 16 is transferred when "PG_DIAL" is called, the remaining bytes of the array are padded with B#16#00.

BUSY parameter

Description

"PG_DIAL" is an asynchronous instruction, in other words, its processing extends over multiple calls.

- If the BUSY output parameter = 1, the internal status of "PG_DIAL" is output in the output parameter STATUS (Page 7936).
- When processing is completed, this is indicated by the output parameter BUSY = 0. The STATUS (Page 7936) output parameter then indicates whether the job was completed error-free (STATUS = W#16#0000) or with errors.

STATUS parameter

Description

The return values of "PG_DIAL" can be classified as follows:

- W#16#0000: "PG_DIAL" was completed successfully
- W#16#7xxx: Status of "PG_DIAL"
- W#16#8xxx: An error was reported during the internal call of a communication instruction or the "BLKMOV" instruction
- W#16#9xxx: Parameter error when calling "PG_DIAL"
- W#16#Bxxx: An error was reported by the TS Adapter

The following tables shows the return values of "PG_DIAL" except for error codes of the communication instructions used.

Return value (W#16#...)	Explanation	Notes
0000	The processing of "PG_DIAL" was completed without errors.	
7000	"PG_DIAL" was reset (the communication with the TS Adapter was terminated in the meantime).	Call "PG_DIAL" again.
7001	"PG_DIAL" is active (first call, BUSY = 1). The function has just been started.	
7002	"PG_DIAL" is active (subsequent call, BUSY = 1). The processing of the function is not yet completed.	
8xxx or 8zxx	The processing of "PG_DIAL" was completed with an error code of the internally called communication instructions. If the error message originates from the "BLKMOV" instruction, the meaning is as follows • z = 2: Error copying the PHONE_NO parameter to the internal buffer • z = 3: Error copying the EVENT_ID parameter to the internal buffer	You will find more detailed information in the error information of the communication instructions: Auto-Hotspot

Return value (W#16#...)	Explanation	Notes
9001	Length of PHONE_NO = 0 or > 31	The telephone number must consist of at least 1 character and not more than 31 characters.
9002	Length of EVENT_ID = 0 or > 16	The event ID must consist of at least 1 character and not more than 16 characters.
B000	It was possible to establish the remote connection to the PG/PC. With TeleService, however, no program logged on for the transferred event ID.	
B10A	A job is already stored on the TS Adapter and this job is currently being processed by the TS Adapter.	Call "PG_DIAL" again.
B10B	After the establishment of the remote connection, the connection was terminated by the partner before the event ID could be transferred.	Call "PG_DIAL" again.
B206	It was not possible to establish the remote connection to the PG/PC.	Check the parameter assignment of the modem in the TS Adapter (local and remote).
B302	The TS Adapter is directly connected to a PG/PC (direct connection).	Terminate the communication on the direct connection.
B305	A remote connection to a PG/PC is still established.	Call "PG_DIAL" again later.

AS_DIAL: Establish remote connection to AS

Description of AS_DIAL

Definition

Local and remote automation system

- Local: The S7 automation system from which the initiative to establish the remote connection originates is described as local.
- Remote: The S7 automation system to which the remote connection will be established is described as remote.

Description

With the "AS_DIAL" instruction, you can establish a remote connection from a local S7 automation system to a remote S7 automation system and then exchange process data. You can use the "X_GET (Page 7928)", "X_PUT (Page 7930)" and "X_SEND (Page 7923)" instructions to exchange process data.

The "AS_DIAL" instruction has two functions:

- **Function DIAL:** Establishing a remote connection to a remote TS Adapter. You use the REQ_DIAL (Page 7940) input parameter to request this function.
- **Function HANGUP:** Terminating an existing remote connection to a remote TS Adapter. You use the REQ_HANGUP (Page 7940) input parameter to request this function.

Only one of the functions can be active at one time. The DIAL function is terminated by the HANGUP function request.

If an error occurs during the processing that leads to the processing aborting, the error code is transferred to the caller of "AS_DIAL". Any established remote connection will be terminated by the TS Adapter.

Calling "AS_DIAL"

You can call "AS_DIAL" in the cycle or in a time-controlled program. If you call "AS_DIAL" in different tasks, make sure that the calls are interlocked.

- An instance DB must be specified when "AS_DIAL" is called. Always use the same instance for a remote connection.
  Exception: If the local CPU communicates via multiple local TS Adapters, assign a separate instance to each TS Adapter.
- "AS_DIAL" must be called more than once to process a block function. This means that calling "AS_DIAL" in a "queue" is pointless. The end of the block function is indicated by BUSY = 0.
- To communicate with the TS Adapter, "AS_DIAL" also uses communication instructions for non-configured S7 connections.
  Therefore, make sure that no communication instructions for non-configured S7 connections ("X_PUT (Page 7930)", "X_GET (Page 7928)", "X_SEND (Page 7923)", "X_RCV (Page 7924)", "X_ABORT (Page 7931)") that were started with DEST_ID = "MPI address of the local TS Adapter" are active when the remote connection is being established or terminated by "AS_DIAL".

Parameters

The following table shows the parameters of the "AS_DIAL" instruction:

Parameters	Declaration	Data type	Memory area	Description
ADDR_TS_ ADAPTER	Input	INT	I, Q, M, D, L or constant	Input parameters of the DIAL and HANGUP functions: MPI address of the TS Adapter on the local MPI network
PHONE_NO (Page 7939)	Input	ANY	D	Input parameters of the DIAL function: Telephone number of the modem of the remote automation system (max. 31 characters)
LOGIN	Input	ANY	D	Input parameters of the DIAL function: User name for authorization on the remote TS Adapter (max. 8 characters)
PASSWORD	Input	ANY	D	Input parameters of the DIAL function: Password for authorization on the remote TS Adapter (max. 8 characters)
ADDR_CPU	Input	INT	I, Q, M, D, L or constant	Input parameters of the DIAL function: MPI address of the CPU on the remote MPI network to which the connection is to be established.

Parameters	Declaration	Data type	Memory area	Description
REQ_DIAL (Page 7940)	Input	BOOL	I, Q, M, D	Input parameters for requesting the DIAL function: Establishing a remote connection to a remote TS Adapter
REQ_HANGUP (Page 7940)	Input	BOOL	I, Q, M, D	Input parameters for requesting the HANGUP function: Terminating an existing remote connection to a remote TS Adapter
STATUS (Page 7941)	Output	INT	I, Q, M, D, L	Return value of "AS_DIAL"
BUSY (Page 7941)	Output	BOOL	I, Q, M, D, L	• BUSY = 1: The DIAL or HANGUP function is still active; in other words, establishment/termination of the remote connection is not yet completed. • BUSY=0: Processing of the "AS_DIAL" instruction is completed. No function is still active.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

Virtual block parameters EN and ENO and the BR bit

The virtual block parameters EN and ENO occur only with the integration of "AS_DIAL" in the LAD or FBD representation.

- Input parameter EN: The status of the result of logic operation (RLO) is saved to the BR bit during the block call.
- Output parameter ENO:
  If "AS_DIAL" is processed error-free, the status of the BR bit that existed when the block was called is restored again at the end of the block call.
  If an error message is output via the STATUS (Page 7941) output parameter, the BR bit is set to "0" immediately before exiting "AS_DIAL".

See also

Data exchange between remote systems (AS-AS-remote coupling) (Page 10967)

PHONE_NO parameter

Description

The input parameter PHONE_NO specifies the telephone number to which a remote connection will be established. Enter the complete telephone number including country code, area code and subscriber number. The string is transferred unchanged to the modem. If you use characters that are not numerals, make sure that the modem used supports these characters.

The TS Adapter uses the following parameter values to establish a remote connection:

- Location: Dial procedure, access code
- Call settings: "Wait for dial tone before dialing", "Number of redial attempts" and "Redial after"

Parameter REQ_DIAL

Description

The DIAL function can only be requested if BUSY (Page 7941) = 0 was reported at the last of "AS_DIAL" call. With the REQ_DIAL input parameter, you start the DIAL function. The REQ_DIAL input parameter is level-triggered.

REQ_DIAL triggers the DIAL function of the "AS_DIAL" instruction. The "AS_DIAL" instruction checks the parameters for plausibility and transfers these to the local TS Adapter.

If an error is detected during the processing of the DIAL function, this causes processing of the DIAL function to be aborted. "AS_DIAL" automatically executes the HANGUP (Page 7940) function and then outputs the error message in the STATUS (Page 7941) return value.

Result: With BUSY (Page 7941) = 0, the result of the connection establishment is displayed in the STATUS (Page 7941) return value.

REQ_HANGUP parameter

Description

With the REQ_HANGUP input parameter, you start the HANGUP function. The HANGUP function terminates a remote connection to the remote TS Adapter, which was established earlier by the DIAL (Page 7940) function.

The REQ_HANGUP input parameter is level-triggered. The HANGUP function can also be started if the DIAL (Page 7940) function was already started.

The end of the HANGUP function is notified by the output parameter BUSY (Page 7941) = 0. The STATUS (Page 7941) output parameter then supplies the result of the function processing.

Value	Meaning
W#16#0000	The remote connection was terminated normally by the HANGUP function.
W#16#3007	The remote connection was terminated by the local TS Adapter because an error occurred.
W#16#3008	The remote connection has broken down spontaneously or the remote TS Adapter has hung up.

BUSY parameter

Description

"AS_DIAL" is an asynchronous instruction, in other words, its processing extends over multiple calls.

- If BUSY = 1, the internal STATUS (Page 7941) of "AS_DIAL" is output in the STATUS (Page 7941) output parameter. At least one further "AS_DIAL" call is necessary.
- If BUSY = 0, the processing is completed. The STATUS (Page 7941) output parameter then indicates whether the job was completed error-free or with errors.

STATUS parameter

Description

The return values of "AS_DIAL" can be classified as follows:

- W#16#0000: "AS_DIAL" was completed without errors
- W#16#3xxx: Most recent status of remote connection in the TS Adapter
- W#16#7xxx: Status of "AS_DIAL"
- W#16#8xxx: An error was reported during the internal call of a communication instruction ("X_ABORT (Page 7931)", "X_GET (Page 7928)", "X_PUT (Page 7930)", "X_RCV (Page 7924)" and "X_SEND (Page 7923)").
- W#16#9xxx: Parameter error when calling "AS_DIAL"
- W#16#Bxxx: Error during connection establishment

The following table shows the return values of "AS_DIAL" except for error codes of the communication instructions used:

Return value (W#16#...)	Explanation	Notes
0000	No function is active. The processing of "AS_DIAL" was completed without errors.	This message is only displayed once after successful completion of the relevant function.
3007	The local TS Adapter has hung up.	Is only displayed by the HANGUP function.
3008	The remote connection has broken down spontaneously or the remote TS Adapter has hung up.	Is only displayed by the HANGUP function.
7000	No function is active.	No function was processed or requested via the input parameters.
7001	DIAL function is active (first call, BUSY = 1). The function has just been started.	The REQ_DIAL input parameter is not evaluated until the end of the (BUSY = 0) function.
7002	DIAL function is active (subsequent call, BUSY = 1). The processing of the function is not yet completed.	The REQ_DIAL input parameter is not evaluated until the end of the (BUSY = 0) function.

Return value (W#16#...)	Explanation	Notes
7701	The HANGUP function is active (first call). The function has just been started.	The input parameters REQ_DIAL and REQ_HANGUP are not evaluated until the end of the (BUSY = 0) function.
7702	HANGUP function is active (subsequent call). The processing of the function is not yet completed.	The input parameters REQ_DIAL and REQ_HANGUP are not evaluated until the end of the (BUSY = 0) function.
8xxx	Error message: The processing of the active block function was terminated because of an error message (RET_VAL < 0) of an instruction called within the block.	You will find more detailed information in the error information of the communication instructions: Auto-Hotspot.
9001	Length of PHONE_NO = 0 or > 31	The telephone number must consist of at least 1 character and not more than 31 characters.
9003	Length of LOGIN > 8	The user name for authorization on the remote TS Adapter must not exceed 8 characters.
9004	Length of PASSWORD > 8	The password for authorization on the remote TS Adapter must not exceed 8 characters.
B100	TS Adapter cannot currently establish a connection.	The TS Adapter cannot currently call a connection to the modem. Call "AS_DIAL" again.
B10A	An event ID of a "PG_DIAL" function is already stored in the TS Adapter or an "AS_DIAL" function is active that is still to be transferred.	Call "AS_DIAL" again.
B10B	The remote connection was terminated again after successful establishment of the remote connection.	Call "AS_DIAL" again.
B206	It was not possible to establish the remote connection to the TS Adapter.	Check the parameter assignment of the modem in the TS Adapter (local and remote).
B20A	The remote TS Adapter type does not support AS-AS remote coupling.	Use a TS Adapter that supports the AS-AS remote coupling.
B20B	The "AS_DIAL" instruction and the version of the TS Adapter are incompatible.	Use a TS Adapter that supports the AS-AS remote coupling.
B20C	The remote TS Adapter cannot join the MPI network. Cause: Network parameter error	Check the parameter assignment of the network parameters in the remote TS Adapter.
B20D	The version of the remote TS Adapter does not support an AS-AS remote coupling.	Use a TS Adapter that supports the AS-AS remote coupling.
B20E	Data transfer to the remote TS Adapter was aborted. Cause: Transmission error	Call "AS_DIAL" again.
B20F	The remote TS Adapter has not called back.	Check the access protection parameter settings (callback number) and modem in the remote TS Adapter.
B210	The data transmission to the remote TS Adapter is disrupted. Cause: Timeout	Call "AS_DIAL" again.
B252	The remote TS Adapter rejects the job due to lack of authorization.	You have not specified a user name and password for the "AS_DIAL" call.

Return value (W#16#...)	Explanation	Notes
B253	The remote TS Adapter rejects the authorization. Cause: Unknown user name	Check the call parameters of "AS_DIAL" in your user program.
B254	The remote TS Adapter rejects the authorization. Cause: Incorrect password	Check the call parameters of "AS_DIAL" in your user program.
B302	The TS Adapter is directly connected to a PG/PC (direct connection).	Terminate the communication on the direct connection.
B305	A remote connection to a PG/PC is still established by the local TS Adapter.	Call "AS_DIAL" again later.

SMS_SEND: Send text (SMS) message

Description of SMS_SEND

Description

The "SMS_SEND" instruction transfers a telephone number, a service center number and an SMS message to a TS Adapter. The TS Adapter transfers this data via GSM commands to a wireless modem.

After the SMS message has been sent, the TS Adapter receives an acknowledgement that is forwarded to the "SMS_SEND" instruction. The processing of "SMS_SEND" is then terminated and the status communicated to the caller. This status is only an acknowledgement that the SMS has been sent, not a confirmation of receipt.

If an error occurs during the processing that leads to the processing aborting, the error code is transferred to the caller of "SMS_SEND".

Calling "SMS_SEND"

You can call the "SMS_SEND" instruction in the cycle or a time-controlled program.

An instance DB must be specified when "SMS_SEND" is called.

"SMS_SEND" must be called more than once to process a block function. This means that calling "SMS_SEND" in a "queue" is pointless. The end of the block function is indicated by BUSY = 0.

Communication connection abort

If the CPU changes to STOP mode while "SMS_SEND" is active, the communication connection to the TS Adapter is aborted. The communication connection to the TS Adapter is also lost if serious communication problems arise on the MPI bus or, for example, the CPU is switched off.

In these cases the telephone number, service center number and SMS message that were already transferred to the TS adapter are not discarded by the TS adapter. The TS Adapter transfers the data to the wireless modem. However, the acknowledgement that is transferred from the wireless modem to the TS Adapter, is discarded by the TS Adapter.

If a user program attempts to send an SMS to the CPU while the TS Adapter is in the status described above, "SMS_SEND" is terminated with the return value W#16#B10A. The user program can later repeat the sending of an SMS. You also receive the return value W#16#B10A, if different CPUs attempt to send an SMS at the same time via the same TS Adapter.

NOTICE
Changing the STEP 7 user program You can only change the parts of your STEP 7 user program that directly affect calls of "SMS_SEND" when the CPU is in STOP mode. This relates, in particular, to deleting and replacing program blocks that contain "SMS_SEND" calls. Ignoring this restriction can tie up connection resources. The automation system can change to an undefined status for the communication instructions for non-configured S7 connections ("X_PUT (Page 7930)", "X_GET (Page 7928)", "X_SEND (Page 7923)", "X_RCV (Page 7924)", "X_ABORT (Page 7931)"). A warm or cold restart of the CPU is required after the changes are transferred.

Data consistency

The input parameters of the function block are copied to in internal buffer when "SMS_SEND" is first called. Do not change this data before the first call is completed (return value W#16#7001), otherwise inconsistent data could be transferred.

Transfer of PIN to the wireless modem

If you use a SIM card on which the PIN check is enabled, the PIN must be transferred during the initialization phase of the wireless modem. This is possible using the init string of the TS Adapter, which you can set using TeleService. Set the init string as follows (example for PIN = 4711):

- AT+CPIN="4711";AT&F.....

When power is restored, the TS Adapter sends the string "AT+CPIN="4711" once, along with the PIN for the SIM card used in the wireless modem, to the connected components. The wireless modem is then initialized with "AT&F...".

Caution: An incorrect PIN is not reported during the initialization, but only when an SMS is sent via the feedback message of "SMS_SEND".

Sending a fax

By prefixing a special number of the network provider (for example, "99") before actual number, the SMS is converted by the network provider into a fax and sent to a fax machine. This is a service of the network provider, not a function of the TS Adapter.

The fax number is entered with the prefix of the network provider in the PHONE_NO parameter.

Example of a network with the fax number 07214711:

- PHONE_NO = '9907214711'

Sending an e-mail

By dialing a special number of the network provider (for example, "8000") and appending an e-mail address before the SMS message, the SMS can be sent to an e-mail address. This is a service of the network provider, not a function of the TS Adapter.

You enter the corresponding number in the PHONE_NO (Page 7946) parameter.

In the SMS message, the e-mail address prefixes the actual text of the message and is delimited from this by a separator. This shortens the maximum length of the SMS payload by the length of the e-mail address + separator. Separators are, for example, blanks (" ") or colons (":").

Some service providers demand a "*" instead of the "@" character.

Example:

- The text 'CPU in STOP' is to be sent to the e-mail address "Surname*provider.com".
- PHONE_NO = 8000
- MESSAGE = Surname*provider.comCPU in STOP

Parameters

The following table shows the parameters of the "SMS_SEND" instruction:

Parameters	Declaration	Data type	Memory area	Description
MPI_TS_ ADAPTER	Input	INT	I, Q, M, D, L or constant	MPI address of the TS Adapter
PHONE_NO (Page 7946)	Input	ANY	D	Reference to a data string with a maximum length of 31 characters
SCENTER_NO (Page 7946)	Input	ANY	D	Reference to a data string with a maximum length of 20 characters
MESSAGE (Page 7946)	Input	ANY	D	Reference to a data string with a maximum length of 160 characters
BUSY (Page 7947)	Output	BOOL	I, Q, M, D, L	• BUSY=1: The sending of the SMS is not yet completed • BUSY=0: Processing of "SMS_SEND" was terminated
STATUS (Page 7947)	Output	INT	I, Q, M, D, L	Return value of "SMS_SEND"

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

Virtual parameters EN and ENO and the BR bit

The virtual block parameters EN and ENO occur only with the integration of "SMS_SEND" in the LAD or FBD representation.

- Input parameter EN:
  The status of the result of logic operation (RLO) is saved to the BR bit during the block call.
- Output parameter ENO:
  If "SMS_SEND" is processed error-free, the status of the BR bit that existed when the block was called is restored again at the end of the block call.
  If an error message is output via the output parameter STATUS, the BR bit is set to "0" immediately before exiting "SMS_SEND".

See also

Send SMS from a system (Page 10969)

PHONE_NO parameter

Description

The input parameter PHONE_NO specifies the telephone number to which an SMS is to be sent. Enter the complete telephone number including country code, area code and subscriber number. The string is transferred unchanged to the wireless modem. If you use characters that are not numerals, make sure that the wireless modem used supports these characters.

Parameter SCENTER_NO

Description

The SCENTER_NO input parameter specifies the service center number to which the SMS will be sent. The string is transferred unchanged to the wireless modem.

MESSAGE parameter

Description

The MESSAGE input parameter specifies the SMS message. This is transferred using the "SMS_SEND" instruction, via the TS Adapter and the GSM wireless modem to the network provider.

BUSY parameter

Description

"SMS_SEND" is an asynchronous instruction, in other words, its processing extends over multiple calls.

- If the BUSY output parameter = 1, the internal status of "SMS_SEND" is output in the output parameter STATUS (Page 7947).
- When processing is completed, this is indicated by the output parameter BUSY = 0. The STATUS (Page 7947) output parameter then indicates whether the job was completed error-free (STATUS = W#16#0000) or with errors.

STATUS parameter

Description

The return values of "SMS_SEND" can be classified as follows:

- W#16#0000: "SMS_SEND" was completed successfully
- W#16#7xxx: Status of "SMS_SEND"
- W#16#8xxx: An error was reported during the internal call of a communication instruction or the instruction "BLKMOV"
- W#16#9xxx: Parameter error when calling "SMS_SEND"
- W#16#Bxxx: An error was reported by the TS Adapter
- W#16#Cxxx: An error was reported by the wireless modem

The following table shows the return values of "SMS_SEND" except for error codes of the communication instructions used or the wireless modem.

Return value (W#16#...)	Explanation	Notes
0000	The processing of "SMS_SEND" was completed without errors.	
7000	"SMS_SEND" was reset (the communication with the TS Adapter was terminated in the meantime).	Call "SMS_SEND" again.
7001	"SMS_SEND" is active (first call, BUSY = 1). The function has just been started.	
7002	"SMS_SEND" is active (subsequent call, BUSY = 1). The processing of the function is not yet completed.	

Return value (W#16#...)	Explanation	Notes
8xxx or 8zxx	Processing of "SMS_SEND" was completed with an error code of the communication instructions called internally or the "BLKMOV" instruction. If the error message originates from the "BLKMOV" instruction, the meaning is as follows • z = 2 : Error copying the parameter PHONE_NO to the internal buffer • z = 3 : Error copying the parameter SCENTER_NO to the internal buffer • z = 4 : Error copying the parameter MESSAGE to the internal buffer	For detailed information, refer to the description of the "BLKMOV" instruction and the error information of the communication instructions: Auto-Hotspot
9001	Length of PHONE_NO = 0 or > 31	The telephone number must consist of at least 1 character and not more than 31 characters.
9002	Length of SCENTER_NO = 0 or > 20	The service center number must consist of at least 1 character and not more than 20 characters.
9003	Length of MESSAGE = 0 or > 160	The SMS message must consist of at least 1 character and not more than 160 characters.
B10A	A job is already stored on the TS Adapter and this job is currently being processed by the TS Adapter.	Call "SMS_SEND" again.
B301	A remote connection to a PG/PC is still established.	Call "SMS_SEND" again later.
B302	The TS Adapter is directly connected to a PG/PC (direct connection).	Connect the TS Adapter to a GSM wireless modem
B303	The TS Adapter is not connected to a wireless modem or a PG/PC.	Connect the TS Adapter to a GSM wireless modem
B304	The interface to the wireless modem is not currently ready to send an SMS.	Call "SMS_SEND" again later.
B614	The connected modem did not reply to the SMS-specific command within the monitoring period.	Check whether you have connected a GSM wireless modem. Call "SMS_SEND" again later.
B615	The TS Adapter has received an unspecific error message from the connected modem.	Check whether you have connected a GSM wireless modem.
Cxxx	The processing of "SMS_SEND" was terminated with an error code of the GSM modem. xxx: Error number	For a list of the error numbers, refer to the your wireless modem manual or to the GSM standards GSM 04.11, GSM 03.40 and GSM 07.05. Examples: • C136 = No SIM card present in the GSM modem • C137 = Incorrect PIN for GSM modem (see transfer of PIN to the Funkmodem_bergabe_der_PIN_an_das_Funkmodem) • C001 = Incorrect service center number

AS_MAIL: Transfer email

Description of AS_MAIL

Required instructions

The "AS_MAIL" instruction uses the Simple Mail Transfer Protocol (SMTP) to transfer an e-mail from a CPU to a mail server. "To run, "AS_MAIL" requires the following instructions:

- "S_COMP: Compare character strings (Page 6440)"
- "FIND: Find characters in a character string (Page 6452)"
- "INSERT: Insert characters in a character string (Page 6450)"
- "LEFT: Read the left character of a character string (Page 6448)"
- "LEN: Determine the length of a character string (Page 6446)"
- "RIGHT: Read the right characters of a character string (Page 6448)"

Description

"AS_MAIL" is an asynchronous instruction, in other words, its processing extends over multiple calls. When you call the "AS_MAIL" instruction, you must specify an instance DB.

You initialize the block by a call with COM_RST = 1.

You start the sending of an e-mail with an edge change from "0" to "1" for the REQ parameter.

The job status is indicated by the output parameters BUSY , "DONE", "ERROR", "STATUS" and "SFC_STATUS". "SFC_STATUS" corresponds in this case to the "STATUS" output parameter of the called communication blocks or the "BLKMOV" instruction.

The output parameters DONE, ERROR, STATUS, and SFC_STATUS are each displayed for only one cycle if the status of the BUSY output parameter changes from "1" to "0".

The following table shows the relationship between BUSY, DONE, and ERROR. Using this table, you can determine the current status of the instruction "AS_MAIL" or when the sending of the e-mail is completed.

BUSY	DONE	ERROR	Description
1	Irrelevant	Irrelevant	The job is being processed.
0	1	0	Job successfully completed.
0	0	1	The job ended with an error. The cause of the error can be found in the STATUS and SFC_STATUS parameters.
0	0	0	The "AS_MAIL" instruction was not assigned a (new) job.

If the CPU changes to STOP mode while "AS_MAIL" is active, the communication connection to the mail server aborts. The communication connection to the mail server will also be lost if

communication problems occur on the Industrial Ethernet bus. In this case, the transfer of the e-mail will be interrupted and it will not reach its recipient.

NOTICE
Changing user programs You can change the parts of your user program that directly affect calls of "AS_MAIL" only: • when the CPU is in "STOP" mode or • when no mail is being sent (REQ = 0 and BUSY = 0). This relates, in particular, to deleting and replacing program blocks that contain "AS_MAIL" calls or calls for the instance DB of "AS_MAIL". Ignoring this restriction can tie up connection resources. The automation system can change to an undefined status for the TPC/IP communication functions via Industrial Ethernet. After transferring the changes, run a warm or cold restart on the CPU or set the COM_RST parameter of the "AS_MAIL" instruction.

Data consistency

The ADDR_MAIL_SERVER input parameter of the instruction is re-adopted by the "AS_MAIL" instruction each time the sending of an e-mail is triggered. If a change is made during operation, the "new" value only becomes effective once an e-mail is triggered again.

In contrast, the parameter settings of WATCH_DOG_TIME, TO_S, CC, FROM, SUB, TEXT, ATTACHMENT, and, if applicable, USERNAME and PASSWORD are adopted while the "AS_MAIL" instruction is running, which means that they may only be changed when the job has completed (BUSY = 0)

Setting the parameters of the TS Adapter IE

On the TS Adapter IE, you need to specify parameters for outgoing calls in such a way as to enable the TS Adapter IE to establish a connection to the dial-up server of your service provider.

If you set "When required" for connection establishment, the connection will only be established when an e-mail needs to be sent.

Connection establishment can take longer (approx. one minute) for an analog modem connection. When you specify the WATCH_DOG_TIME parameter, remember to allow for the time required to establish the connection.

Parameters

The following table shows the parameters of the "AS_MAIL" instruction:

Parameter	Declaration	Data type	Memory area	Description
COM_RST	Input / Output	BOOL	I, Q, M, D, L	Input/output parameter "COMPLETE" RESTART: The block has an initialization routine that is processed when the COM_RST input is set. The COM_RST parameter of the "AS_MAIL" instruction must be set to "1": • once, before the first call and • every time that the corresponding instance DB is reloaded. After initialization, the "AS_MAIL" instruction, resets the COM_RST parameter.
REQ	Input	BOOL	I, Q, M, D, L	Control parameter REQUEST: Activates the sending of an e-mail on a rising edge.
ADDR_MAIL_ SERVER	Input	DWORD	I, Q, M, D, L	IP address input parameter of the mail server: Specify as a data word in HEX format, for example: IP address = 192.168.0.200. ADDR_MAIL_SERVER = DW#16#C0A800C8, where: • 192 = 16#C0, • 168 =16#A8 • 0 = 16#00 and • 200 = 16#C8.
WATCH_DOG_ TIME	Input	TIME	I, Q, M, D, L	Input parameter for max. period of time: The "AS_MAIL" instruction should establish a connection within the period specified by WATCH_DOG_TIME. The block is exited with an error if this period is exceeded. The time before the block is exited and the error is output can exceed the WATCH_DOG_TIME because connection termination also takes a certain amount of time. To begin with, you should set a period of 2 minutes. If you have an ISDN telephone connection, a significantly shorter period can be selected.
USERNAME	Input	ANY	D	Input parameter for user name: Reference to a STRING type tag with a maximum length of 180 characters. A user name is an absolute requirement for authentication.
PASSWORD	Input	ANY	D	Input parameter for password: Reference to a STRING type tag with a maximum length of 180 characters. A password is an absolute requirement for authentication.
TO_S	Input	ANY	D	Input parameter for receiver addresses: Reference to a STRING type tag with a maximum length of 240 characters (see call example).

Parameter	Declaration	Data type	Memory area	Description
CC	Input	ANY	D	Input parameter for CC recipient addresses: Reference to a STRING type tag with a maximum length of 240 characters (see call example).
FROM	Input	ANY	D	Input parameter for sender address: Reference to a STRING type tag with a maximum length of 240 characters (see call example).
SUB	Input	ANY	D	Input parameter for subject of the e-mail: Reference to a STRING type tag with a maximum length of 240 characters.
TEXT	Input	ANY	D	Input parameter for text of the e-mail: Reference to a data string with a maximum length of 240 characters.
ATTACHMENT	Input	ANY	I, Q, M, D, L	Input parameter for e-mail attachment: Reference to a byte/word/double word field with a maximum length of 65534 bytes.
BUSY	Output	BOOL	I, Q, M, D, L	• BUSY = 1: E-mail transmission is not yet complete. • BUSY=0: The processing of "AS_MAIL" was stopped.
DONE	Output	BOOL	I, Q, M, D, L	• DONE = 0: Job not yet started or still executing. • DONE = 1: Job was executed without errors.
ERROR	Output	BOOL	I, Q, M, D, L	ERROR=1: An error occurred during processing. STATUS and SFC_STATUS supply detailed information about the type of error.
STATUS (Page 7954)	Output	INT	I, Q, M, D, L	Output/status parameter STATUS: Return value or error information of the "AS_MAIL" instruction.
SFC_STATUS (Page 7954)	Output	INT	I, Q, M, D, L	Output/status parameter "SFC_STATUS": Error information of the called communication blocks or the "BLKMOV" instruction.

You will find more detailed information on valid data types in "Overview of the valid data types (Page 2236)".

Note

Notes to the parameters for the instruction

- If the assignment of the parameters USERNAME and PASSWORD is removed, re-entered or changed in the block in which "AS_MAIL" is called, this change will only become effective if the corresponding instance DB is reloaded and the COM_RST parameter is set. Otherwise, the pointer remains on the corresponding parameter in the instance DB.
- Because of runtime and memory space, the "AS_MAIL" instruction does not perform any syntax check of the parameters TO_S, CC and FROM. Please note the correct syntax of the parameter.
- The optional parameters CC, TEXT and ATTACHMENT are only sent with the e-mail if they are assigned in the block in which the "AS_MAIL" instruction is called. If such an assignment is removed from the call interface of the "AS_MAIL" instruction, this change only becomes effective when the corresponding instance DB is reloaded and the COM_RST parameter is set. Otherwise, the pointer remains on the corresponding parameter in the instance DB.
  Please note that empty strings are not permitted in a parameter assignment.
- If the ATTACHMENT parameter points to an array in a data block and if the size of this array is changed and stored, the ANY pointer of the ATTACHMENT parameter must be entered again at the interface of the "AS_MAIL" instruction.

SMTP authentication

Authentication refers here to a procedure for ensuring an identity, for example, by a password query.

The "AS_MAIL" instruction supports the SMTP authentication procedure AUTH-LOGIN, that is requested by most mail servers. For information about the authentication procedure of your mail server, please refer to your mail server manual or the website of your Internet service provider.

To use the AUTH-LOGIN authentication procedure, the "AS_MAIL" instruction requires the user name with which it can log in with the mail server. This user name corresponds to the user name with which you set up a mail account on your mail server. It is available to the "AS_MAIL" instruction in the USERNAME parameter.

To log on, the "AS_MAIL" instruction also requires the associated password. This password corresponds to the password you specified when you set up your mail account. It is made available to the "AS_MAIL" instruction in the PASSWORD parameter.

The user name and the password are each transferred to the mail server unencrypted (BASE64-coded).

If no user name is specified in the DB, the AUTH-LOGIN authentication procedure is not used. The e-mail is then sent without authentication.

See also

Send an email from a system (Page 10971)

STATUS and SFC_STATUS parameters

Description

The return values of the "AS_MAIL" instruction can be classified as follows:

- W#16#0000: "AS_MAIL" was completed successfully
- W#16#7xxx: Status of "AS_MAIL"
- W#16#8xxx: An error was reported during the internal call of a communication instruction, during the "BLKMOV" instruction call, or by the mail server.

The following table shows the return values of "AS_MAIL" except for error codes of the communication blocks called internally and the "BLKMOV" instruction.

Return value STATUS (W#16#...):	Return value SFC_STATUS (W#16#...):	Explanation	Notes
0000	-	The processing of "AS_MAIL" was completed without errors.	A error-free completion of "AS_MAIL" does not mean that the sent e-mail will necessarily arrive (see below - note on point 1)
7001		"AS_MAIL" is active (BUSY = 1).	
7002	7002	"AS_MAIL" is active (BUSY = 1).	
8xxx	xxxx	Processing of "AS_MAIL" was completed with an error code of the communication instructions called internally or the "BLKMOV" instruction.	For detailed information on the evaluation of the SFC_STATUS parameter, refer to the descriptions of the STATUS parameter of the "BLKMOV" instruction.
800z	xxxx	The error message originates from the "BLKMOV" instruction and means: • z = 1 Error copying the TO_S parameter to the internal buffer • z = 2 Error copying the CC parameter to the internal buffer • z = 3 Error copying the FROM parameter to the internal buffer • z = 4 Error copying the SUB parameter to the internal buffer • z = 5 Error copying the TEXT parameter to the internal buffer • z = 6 Error copying the ATTACHMENT parameter to the internal buffer • z = 7 Error copying the USERNAME parameter to the internal buffer • z = 8 Error copying the PASSWORD parameter to the internal buffer	
8010	xxxx	Error during connection establishment.	COM_RST was possibly not set after loading instance DB.

Return value STATUS (W#16#...):	Return value SFC_STATUS (W#16#...):	Explanation	Notes
8011	xxxx	Error sending the data.	For detailed information on the evaluation of SFC_STATUS, refer to the descriptions of the STATUS parameter of the "TSEND (Page 7436)" instruction.
8012	xxxx	Error receiving the data.	For more information on the evaluation of SFC_STATUS, refer to the descriptions of the STATUS parameter of the "TRCV (Page 7439)" instruction.
8013	xxxx	Error during connection establishment.	For more information on the evaluation of SFC_STATUS, refer to the descriptions of the STATUS parameter of the "TCON (Page 7431)" and "TDISCON (Page 7434)" instructions.
8014	-	Establishment of a connection is not possible.	You have possibly entered an incorrect mail server IP address (ADDR_MAIL_SERVER) or a time span that is too short (WATCH_DOG_TIME) to establish the connection. It is also possible that the CPU has no connection to the network or that the CPU configuration is incorrect.
82xx, 84xx, or 85xx	-	The error message originates from the mail server and corresponds, except for the "8", to the error number of the SMTP protocol. The following columns list several error codes that can occur:	See point 2 in the note.
8450	-	Action not executed: Mailbox not available/ not reachable.	Try again later.
8451	-	Action aborted: Local processing error	Try again later.
8500	-	Syntax error: Error not recognized. This also includes the error when a command string is too long. This can also be occur when the e-mail server does not support the LOGIN authentication procedure.	Check the parameters of "AS_MAIL". Try to send an e-mail without authentication. To do this, replace the USERNAME parameter with an empty string.
8501	-	Syntax error: Parameter or argument incorrect	You have possibly entered an incorrect address in TO_S or CC.
8502	-	Command unknown or not implemented.	Check your entries, in particular the FROM parameter. This is possibly incomplete and you have forgotten "@" or ".".
8535	-	SMTP authentication incomplete.	You have possibly entered an incorrect user name or incorrect password.

Return value STATUS (W#16#...):	Return value SFC_STATUS (W#16#...):	Explanation	Notes
8550	-	Mail server cannot be reached, you have no access rights.	You have possibly entered an incorrect user name or password, or the mail server does not support your LOGIN. Another cause of error could be an incorrect entry of the domain name after the "@" in TO_S or CC.
8552	-	Action aborted: Assigned memory size has been exceeded	Try again later.
8554	-	Transmission failed.	Try again later.

Note

Status error

1. An incorrect entry of the recipient addresses does not generate a status error of the "AS_MAIL" instruction. In this case, there is no guarantee that the e-mail will be sent to other recipients, even if these were entered correctly.
2. You will find more detailed information on the SMTP error code and other error codes in SMTP protocol on the Internet or in the error documentation of the mail server. You can also view the most recent message from the mail server in your instance DB in the BUFFER1 parameter. If you look under "Data", you will find the data most recently sent by the "AS_MAIL" instruction.

Example of "AS_MAIL" call

Description

The data that is available to the "AS_MAIL" instruction based on the ANY pointer is stored in a data block. In this example, DB 2 is used for this.

Because of runtime and memory space, the "AS_MAIL" instruction does not perform any syntax check of the parameters TO_S, CC and FROM.

Parameters TO_S, CC and FROM

The TO_S, CC and FROM parameters are ANY pointers to strings with, for example, the following content:

- TO: <wenna@mydomain.com>, <ruby@mydomain.com>,
- CC: <admin@mydomain.com>, <judy@mydomain.com>,
- FROM: <admin@mydomain.com>

Note the following rules when entering the parameters:

- The "TO:", "CC:" and "FROM:" characters must be entered.
- A space and an opening angled bracket "<" must be entered before each address.

- A closing angled bracket ">" must be entered after each address.
- A comma must be entered after each address in TO_S and CC.
- Only one e-mail may be entered in FROM, and there must be no comma at the end of this address.

Initialization / startup in OB 100

The COM_RST parameter of the "AS_MAIL" instruction must be set before the first call of the instruction, or always after a reloading of the associated instance DB.

In the following example, DB 1 is the instance DB of the "AS_MAIL" instruction.

SET		//Set RLO
R	DB1.REQ	//REQ is reset so that the "AS_MAIL" instruction does not start to send an e-mail immediately after COM_RST is set.
S	DB1.COM_RST	//COM_RST is set; this must be set once for initialization before the first call of the "AS_MAIL" instruction, or always after reloading the associated instance DB.

Cyclic call

This example of the "AS_MAIL" instruction call must be called cyclically (for example, in OB1). In the user program, use "Modify tags" to set the REQ parameter to "1".

You can copy these lines to an STL source file and compile them. To do this, however, the blocks DB 1 and DB 2 must be entered in the project's tag table.

Example of "AS_MAIL" call	Explanation
`DATA_BLOCK "DB2"`	`//Global data block with the data for the "AS_MAIL" instruction`
`TITLE =`	
`VERSION : 1.0`	
`STRUCT`	
`TO_S : STRING [240 ] := 'TO: <wenna@mydomain.com>, <ruby@mydomain.com>,';`	`//Address of recipient`
`CC : STRING [240 ] := 'CC: <admin@mydomain.com>, <judy@mydomain.com>,';`	`//CC Address of recipient`
`FROM : STRING [60 ] := 'FROM: <admin@mydomain.com>';`	`//Address of sender`
`SUB : STRING [60 ] := 'Status plant 7';`	`//Subject`
`TEXT : STRING [240 ] := 'Fault in plant 7 sector 2';`	`//Text`
`USERNAME : STRING [60 ] := 'admin';`	`//User name for LOGIN authentication`
`PASSWORD : STRING [60 ] := 'test';`	`//Password for LOGIN authentication`

Example of "AS_MAIL" call	Explanation
ATTACHMENT : ARRAY [1 .. 178] OF BYTE;	//Attachment
END_STRUCT ;	
BEGIN	
END_DATA_BLOCK	
DATA_BLOCK "DB1"	//Instance DB of the "AS_MAIL" instruction
TITLE =	
AUTHOR : TELESERV	
FAMILY : TELESERV	
VERSION : 1.0	
"AS_MAIL"	
BEGIN	
END_DATA_BLOCK	
FUNCTION_BLOCK FB 1	
TITLE =Example AS_MAIL	The COM_RST parameter of the "AS_MAIL" instruction must be set before the first "AS_MAIL" call, or always after reloading the associated instance DB. The call condition is set during the execution of the user program.
VERSION : 1.0	
VAR	
CallCondition: BOOL ;	
END_VAR	
BEGIN	
NETWORK	
CALL AS_MAIL, DB1 (	//Calling AS_MAIL
REQ := # CallCondition,	//Job triggered on rising edge
ADR_MAIL_SERVER := DW#16#C0A800C8,	//Corresponds to the IP address 192.168.0.200 in HEX format
WATCH_DOG_TIME := T#2M,	//Time span of 2 minutes within which sending should be completed
USERNAME := DB2.USERNAME,	//Pointer to user name of the sender. If there is no DB2.USERNAME , in other words, no parameter is assigned, the LOGIN authentication procedure is not used.
PASSWORD := DB2.PASSWORD,	//Pointer to the password
TO_S := DB2.TO_S,	//Pointer to the recipient address
CC := DB2.CC,	//Pointer to the CC recipient address (optional). If there is no DB2.CC, in other words, if this is not assigned, the e-mail is not sent to the CC recipient.
FROM := DB2.FROM,	//Pointer to the address of the sender
SUB := DB2.SUB,	//Pointer to the subject of the mail

Example of "AS_MAIL" call	Explanation
TEXT := DB2.TEXT,	//Pointer to the text of the mail (optional). If there is no DB2.TEXT, in other words, if this is not assigned, the e-mail is sent without text.
ATTACHMENT := DB2.ATTACHMENT,	//Pointer to the mail attachment (optional). Example: 101 bytes in the mail attachment - ATTACHMENT := P#DB2.DBX974.0 BYTE 101 (example relates to the above DB). //If there is no DB2.ATTACHMENT, in other words, if this is not assigned, the e-mail is sent without an attachment.
BUSY := M46.0,	//Output/status parameter BUSY
DONE := M46.1,	//Output/status parameter DONE
ERROR := M46.2,	//Output/status parameter ERROR
STATUS := MW48,	//Output/status parameter STATUS
SFC_STATUS := MW50	//Output/status parameter SFC_STATUS
);	
	//Checking the call results at the end of the processing
U M 46.0;	//Is BUSY ==0?
SPB End;	//No, BUSY ==1, "AS_MAIL" is still processing the job
U M 46.1;	//Was the processing of "AS_MAIL" completed without errors?
SPB End;	
NOP 0;	//No, an error occurred during processing, BUSY ==0, BR ==0.
	//Add further error evaluation here, if necessary, STATUS, SFC_STATUS
End:BE;	
END_FUNCTION_BLOCK	

Note

Optional parameters

For the optional parameters USERNAME, PASSWORD, CC, TEXT and ATTACHMENT, the following applies:

If an assignment is removed or re-inserted, this change will only be valid when the associated instance DB (in this example, DB 1) is reloaded and the "COM_RST" parameter is set.

Visualize processes 10

10.1 Creating screens

10.1.1 Basics

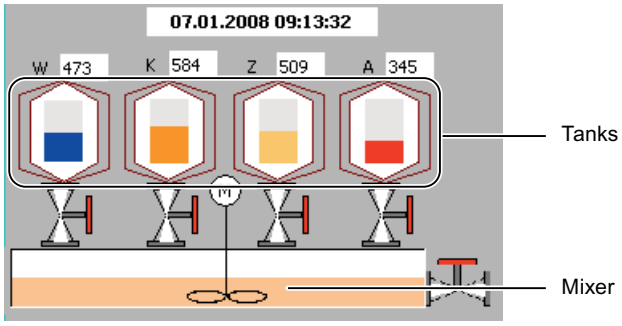
10.1.1.1 Screen basics

Introduction

In WinCC you create screens that an operator can use to control and monitor machines and plants. When you create your screens, the object templates included support you in visualizing processes, creating images of your plant, and defining process values.

Application example

The figure shows a screen that was created in WinCC. With the help of this screen, the operator can operate and monitor the mixing station of a fruit juice manufacturing system. Fruit juice base is supplied from various tanks to a mixing unit. The screen indicates the fill level of the tanks.



Screen design

Insert an object you need to represent a process into your screen. Configure the object to correspond to the requirements of your process.

A screen may consist of static and dynamic elements.

- Static elements such as text or graphic objects do not change their status in runtime. The tank labels (W, K, Z, A) shown in this example of a mixing unit are static elements.
- Dynamic elements change their status based on the process. Visual current process values as follows:
  - From the memory of the PLC
  - From the memory of the HMI device in the form of alphanumeric displays, trends and bars.

  Input fields on the HMI device are also considered dynamic objects. The fill level values of the tanks in our example of a mixing plant are dynamic objects.

Process values and operator inputs are exchanged between the controller and the HMI device via tags.

Screen properties

The screen layout is determined by the features of the HMI device you are configuring. It corresponds with the layout of the user interface of this device. The screen properties such as the screen resolution, fonts and colors are also determined by the characteristics of the selected HMI device. If the set HMI device has function keys, the screen shows these function keys.

A function key is a key on the HMI device. You can assign one or several functions in WinCC. These functions are triggered when the operator presses the relevant key on the HMI device.

A function key can be assigned global or local functions.

- Global function keys always trigger the same action, regardless of the currently displayed screen.
- Function keys with local assignment trigger different actions, based on the currently displayed screen on the operator station. This assignment applies only to the screen in which you have defined the function key.

Opening screens

In order for the operator to be able to call a screen in runtime, you must integrate each configured screen in the operating process. You have various options of configuring these functions:

- You use the "Screen" editor to configure buttons and function keys for opening other screens.
- You use the "Global Screen" editor to configure globally assigned function keys.

10.1.1.2 Device-specific functional scope of screens

Introduction

The functions of an HMI device determine the display of the device in WinCC and the scope of functions of the editors.

The following screen properties are determined by the functions of the selected HMI device:

- Device layout
- Screen resolution
- Number of colors
- Fonts
- Objects available

Device layout

The device layout of a screen forms the image of the HMI device in your configuration. The device layout of the screen shows all the function keys available on the HMI device, for example.

Screen resolution

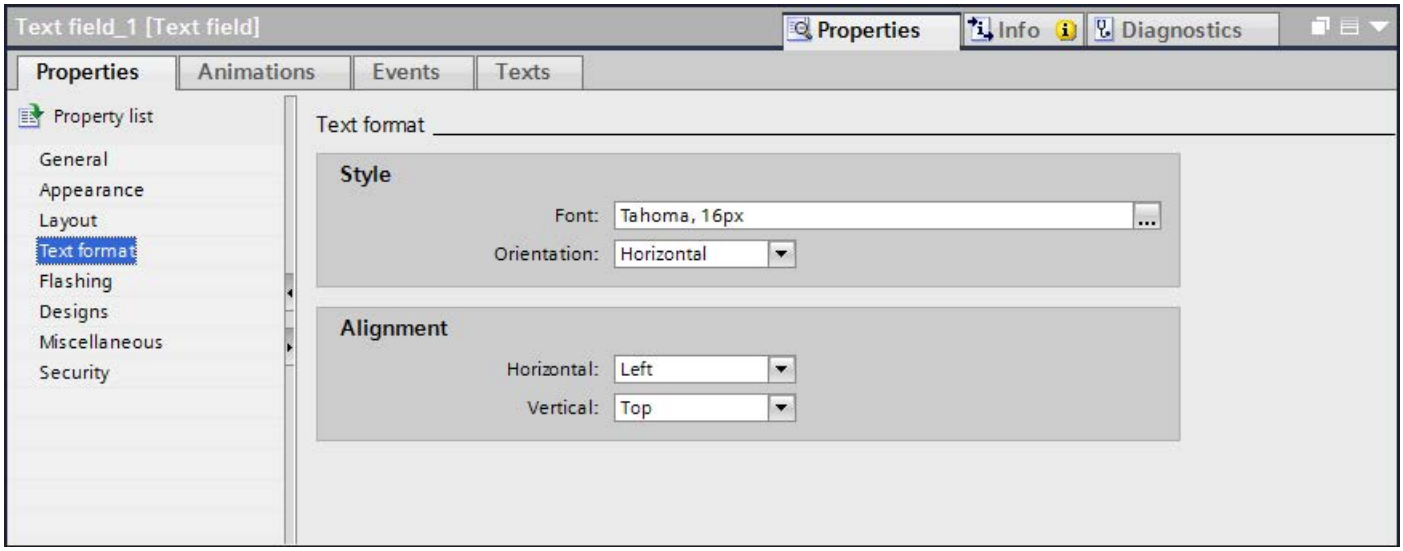
The screen resolution is determined by the different display dimensions of the various operator panels. You can only change the screen resolution if you configure the "WinCC Runtime Advanced" or "WinCC Runtime Professional" HMI device.

Number of colors

You can assign colors to the screen objects. The number of possible colors is determined by the color depth and specific colors supported on the selected HMI device.

Fonts

You can customize the appearance of the texts in all the screen objects that contain static or dynamic text. How to highlight individual texts in a screen. Select the font, font style and size, and set additional effects such as underscoring, for example.



The settings for the text markups such as font style and effects always refer to the entire text of a screen object. That is, you can display the complete title in bold format, but not its individual characters or words, for example.

Objects available

Some of the screen objects can not be configured globally for all HMI devices. These screen objects are not displayed in the "Tools" task card. For a KTP 1000 touch panel unit you can not configure a slider, for example.

10.1.1.3 Basics

Task cards

Introduction

The following task cards are available in the "Screens" editor:

- Tools: Display and operating objects
- Animations: Templates for dynamic configuration
- Layout: Aid for customizing the display
- Libraries: Administration of the project library and of the global libraries

Note
WinCC Basic

The "Animations" task card is not available in WinCC Basic.

Tools

The "Tools" task card contains objects in different panes:

- Basic objects
- Elements
- Controls
- User controls (optional)
- Graphics

You paste objects from the palettes into your screens by drag&drop or a double click. The objects available for selection are determined by the features of the HMI device you are configuring. The following icons are used to change the display mode:

Icon	Meaning
	Displays the objects as a list.
	Displays the objects as a graphic.

Animations

The "Animations" task card contains the possible dynamizations of a screen object in the palettes. You paste the animations to a screen object by drag&drop or a double click from the "Movements", "Display" and "Tag Binding" palettes.

Layout

The "Layout" task card contains the following panes for displaying objects and elements:

- Layers: Serves to manage screen object layers The layers are displayed in a tree view and contain information about the active layer and the visibility of all layers.
- Grid: You specify whether you want to align the objects to a grid or to other objects and set the grid size for a grid.
- Objects out of range: Objects that lie outside the visible area are displayed with name, position and type.

Libraries

The "Libraries" task card show the following libraries in separate panes:

- Project library: The project library is stored together with the project.
- Global library: The global library is stored in a separate file in the specified path on your configuration PC.

Move view

Introduction

To display only a section of the entire screen in the work area, use the icon of the "Screens" editor:

Requirement

- A screen is open.
- The view shows only a screen section.

Procedure

To move the view:

1. Click the icon at the bottom right corner of the work area and press the left mouse button. A miniature view of the full screen is shown. An orange frame shows the currently selected area.
2. Hold down the mouse button and drag the frame to the desired area.

Note

The screen is scrolled when you drag a screen object from the visible to a currently hidden section.

Zooming the view

Introduction

To view a small screen section in closer detail, use the zoom tool to magnify the screen in the working area. The maximum zoom amounts to 800%.

You can zoom with the toolbar in the work area or with the "Layout > Zoom" task card.

There are different ways to increase the screen, for example, with the zoom factor or by adapting the work area to the height of the screen.

Requirement

The screen is opened.

Procedure

Proceed as follows to zoom a view with the selection frame:

1. Click the toolbar button.
2. Use the mouse to draw a selection frame in the screen.

After you have released the mouse button, the section enclosed by the selection frame is zoomed to fit the complete work area.

Alternatively, use the slider in the lower right-hand corner of the screen.

Result

The selected screen section is magnified.

10.1.1.4 Working with screens

Steps

Steps

To create screens, you need to take the following initial steps:

- Plan the structure of the process visualization: Number of screens and their layout
  Example: Subprocesses are visualized in separate screens, and merged in a master screen.
- Define your screen navigation control strategies.
- Adapt the templates and the global screen.
  You define objects centrally and assign function keys for example.
- Create the screens. Use the following options of efficient screen creation:
  - Working with libraries
  - Working with layers
  - Working with faceplates

Creating a new screen

Introduction

Create screens to display processes in your system.

Requirement

- The project has been created.
- The Inspector window is open.

Procedure

1. Double-click "Screens > Add New Screen" in the project navigation.
   The screen is generated in the project and appears in your view. The screen properties are shown in the Inspector window.
2. Enter a meaningful name for the screen.
3. Configure the screen properties in the Inspector window:
   - Specify whether and on which template the screen is based.
   - Set the "Background Color" and the "Screen Number."
   - Specify a documenting text under "Tooltip".
   - Specify the layers to be displayed under "Layers" in the engineering system.
   - Select dynamic screen update under "Animations."
   - Select "Events" to define which functions you want to execute in Runtime when you call and exit the screen or at other events.

   Note

   Not all HMI devices support the "Visibility" animation.

Result

You created the screen in your project. You can now paste objects and control elements from the "Tools" task card and assign function keys in further work steps.

Managing Screens

Introduction

You can move screens within a project to other groups, or copy, rename, and delete them.

Moving screens in a group

1. Select the "Screens" folder in the project tree.
2. Select the "Add group" command from the shortcut menu.
   A folder called "Group_x" is inserted.
3. Select the screen in the project tree.
4. Drag-and-drop the screen to the required group.
   The screen is moved into this group.

Copy screen

1. Select the screen in the project tree.
2. Select the "Copy" command in the shortcut menu to copy the screen to the clipboard.

3. In the project tree, select the screen insert position.
4. Select "Paste" from the shortcut menu to insert the screen.
   A copy of the screen is inserted. A consecutive number is appended to the name of the original in the copy.

Alternatively, press <Ctrl> while you drag the screen to the required position.

Note

If you copy a screen with interconnected template for several devices and projects, the template will also be copied. Any existing matching template is not used. This holds particularly true when you copy the screens with drag-and-drop.

Rename screen

1. Select the screen in the project tree.
2. Select "Rename" from the shortcut menu.
3. Type in a new name.
4. Press <Enter>.

As an option, use the <F2> function key to rename the screen.

Delete screen

1. Select the screen in the project tree.
2. Select "Delete" from the shortcut menu.
   The screen and all its objects are deleted from the current project.

Defining the start screen of the project

Introduction

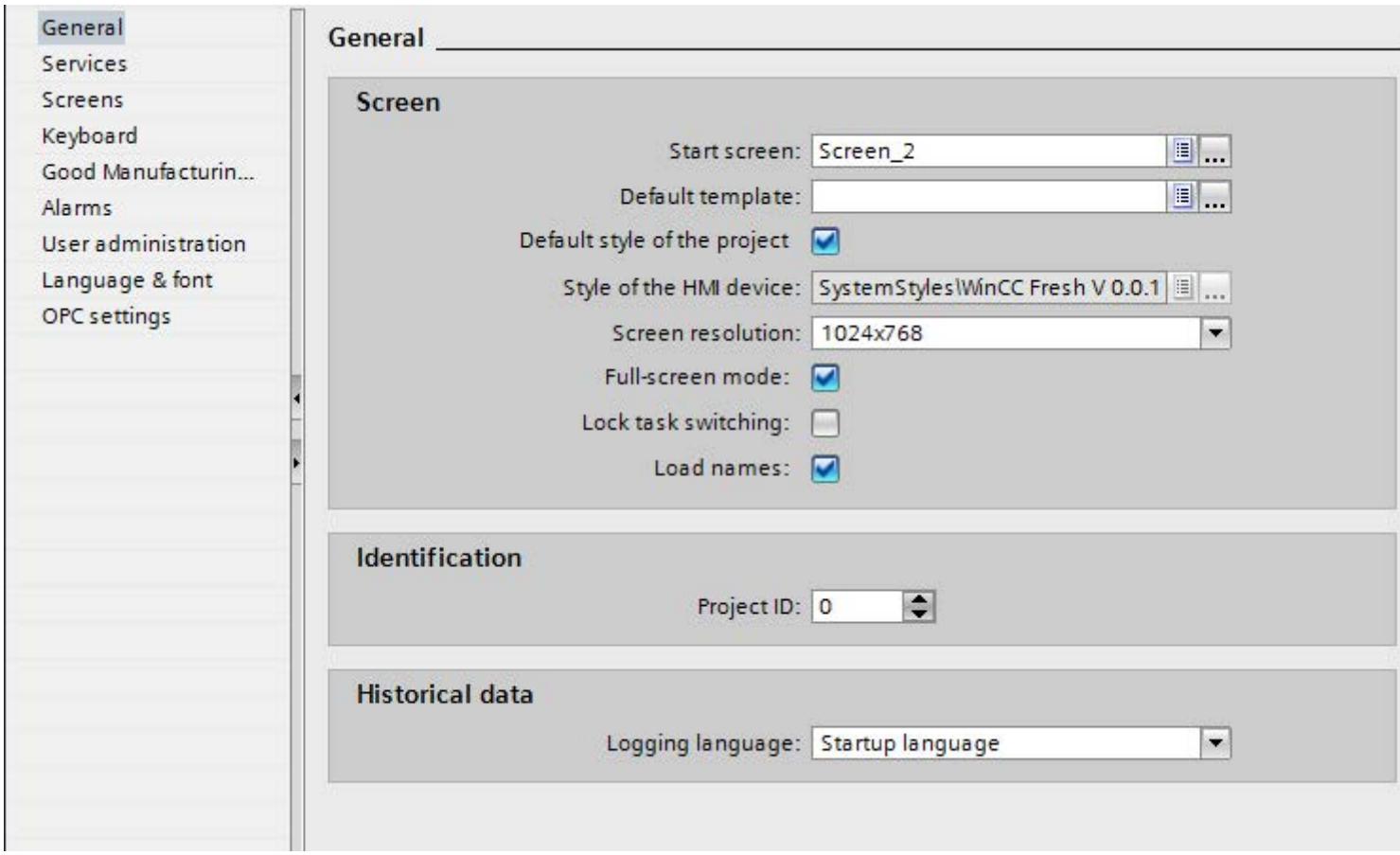
The start screen is the initial screen opened at the start of project in Runtime. You can define a different start screen for each one of the HMI devices. Beginning at this start screen, the operator calls the other screens.

Requirement

The project contains the screen you want to use as the start screen.

Procedure

1. Double-click "Runtime settings > General" in the project tree.



2. Select the desired screen as "Start screen".

Alternatively select a screen in the project tree and select "Use as start screen" in the shortcut menu.

Result

The start screen opens on the HMI at the start of Runtime.

10.1.1.5 Working with Templates

Basics on working with templates

Introduction

Configure the objects in a template which are to be displayed in all screens based on this template.

Note the following rules:

- A screen must not be based on a template.
- A screen is only based on one template.
- You can create several templates for one device.
- A template cannot be based on another template.

Objects for a template

You determine functions and objects in the template which are to apply for all screens based on this template:

- Assignment of function keys: You also assign the function keys in the template for HMI devices with function keys. This assignment overwrites a possible global assignment.
- Permanent window: Some devices support a permanent window for all screens in the top area of the screen. In contrast to the template, the permanent window occupies an area of the screen for itself alone.
- Operator controls: You can paste all screen objects which you also use for a screen into a template.

Application examples

- You want to assign the ActivateScreen" function to a function key in the template. The operator uses this key to switch to another screen in runtime. This configuration applies to all screens that are based on this template.
- A graphic with your company logo can be added to the template. The logo appears on all screens that are based on this template.

Note

If an object from the template has the same position as an object in the screen, the template object is covered.

See also

Creating a new template (Page 7973)

Managing templates (Page 7974)

Global screen (Page 7971)

Using a template in the screen (Page 7975)

Global screen

Introduction

You define global elements for all screens of an HMI device independently of the used template.

Function keys

For HMI devices with function keys you assign the function keys globally in the "Global Screen" editor. This global assignment applies for all screens of the HMI device.

Proceed as follows to assign function keys locally in screens or templates:

1. Click the function key in your screens or templates.
2. Deactivate "Properties > Properties > General > Use Global Assignment" in the inspection window.

Indicator and control objects for alarms

The "Alarm window" and "Alarm indicator" objects that are available as global objects are configured within the "Global screen" editor.

The "Alarm window" and "Alarm indicator" are always shown in the foreground.

For Comfort Panels you also have the possibility of configuring a "System diagnostic view" in the global screen.

Note

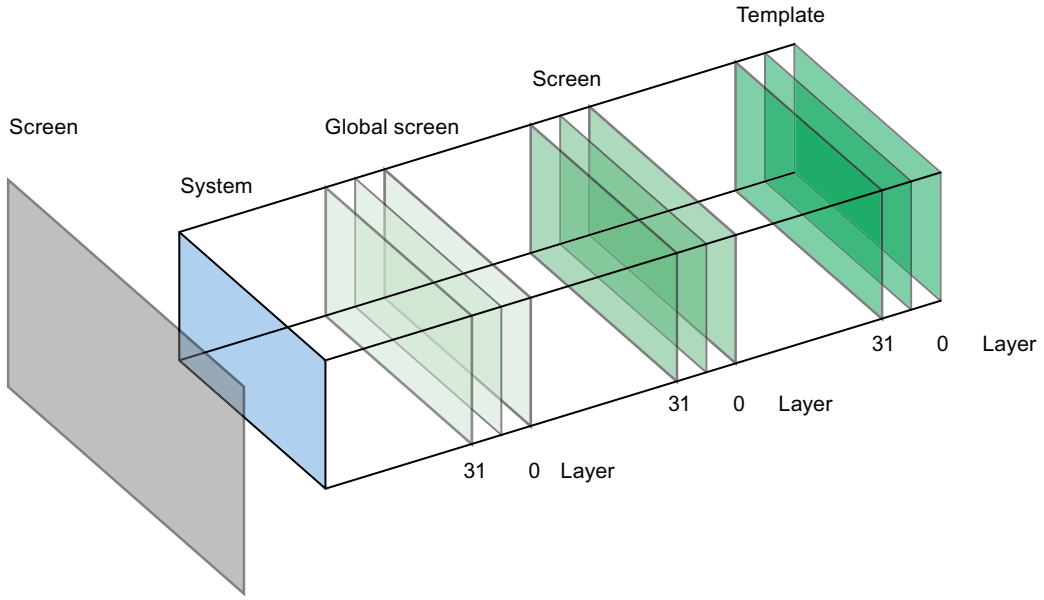
If you have configured a permanent window in a template, do not position the alarm window and alarm indicator in the area of the permanent window. Otherwise, the alarm window and the alarm indicator will not be displayed in Runtime.

The permanent window is not visible in the "Global screen" editor.

Order of configuration of screens

The following order applies for the configuration:

- The global screen comes before screens and templates
- Screens come before templates



The system layer is not configurable. This contains

- input dialog boxes
- alarms from the operating system
- the direct keys for touch panels

See also

Basics on working with templates (Page 7970)

Creating a new template

Introduction

In a template, you can centrally modify objects and function keys. Changes to an object or of a function key assignment in the template are applied to the object in all the screens which are based on this template.

Note

HMI device dependency

Function keys are not available on all HMI devices.

Requirement

- The project has been created.
- The Inspector window is open.

Procedure

1. Select "Screen management > Templates" in the project tree and then double-click "Add new template".
   The template is created in the project, and appears in your view.
   The properties of the template are displayed in the Inspector window.
2. Define the name of the template under "Properties > Properties > General" in the Inspector window.
3. Specify the layers in the engineering system that are displayed under "Properties >Properties >Layers" in the Inspector window.
4. Add the necessary objects from the "Tools" task card.
5. Configure the function keys.

Result

The template is created in your project.

See also

Basics on working with templates (Page 7970)

Managing templates

Introduction

You can move, copy, rename, and delete templates within a project in the Project window.

Moving a template into a group

1. Select the templates in the project navigation "Screen management > Templates".
2. Select "Add group" in the shortcut menu.
   A folder called "Group_x" is inserted.
3. Select the template in the project navigation.
4. Drag-and-drop the template to the required group.
   The template is moved to this group.

Copying templates

1. Select the template in the project navigation.
2. Select "Copy" in the shortcut menu.
3. Select the position in the project navigation where you want to paste the template.
4. Select "Paste" from the shortcut menu to insert the template.
   A unique name is assigned automatically to the copy.

Alternatively, you can hold down the <Ctrl> key, and drag the template into position.

Deleting a template

1. In the project navigation, select the template to be deleted.
2. Select "Delete" in the shortcut menu.
   The template, and all its objects are deleted from the current project.

Assigning a template to a screen

1. In the project navigation, select the screen to which you want to assign the template.
2. In the Inspector window, select "Properties > Properties > General".
3. Select the desired template under "Template."
   The selected template and all the objects contained in it are assigned to the screen.

See also

Basics on working with templates (Page 7970)

Using a template in the screen

Introduction

You use a template in the screen. All the objects configured in the template are also available in the screen.

Requirement

A template has been created.

A screen has been created.

Procedure

Proceed as follows to use a template in a screen:

1. Double click a screen in the project tree. The screen opens in the work area.
2. Open "Properties > Properties > General" in the inspector window.
3. Select a template that is to be applied to the screen under "Template".

Show template in screen

When you edit a screen, you can show an existing template in the screen.

Proceed as follows to show a template in the screen:

1. Activate "Extras > Settings > Visualization > Show template in screens" in the menu.

Result

The screen is based on the selected template. All objects which you have configured in the template are available in the screen. The template is displayed in the screen.

See also

Basics on working with templates (Page 7970)

10.1.1.6 Working with styles

Basics on working with styles

Introduction

The style editor is a global editor. You define a uniform appearance for display and operating elements in the style editor. In this way, you harmonize the display of objects in Runtime. The style editor offers predefined styles. You can choose one of the predefined styles or create your own style. You use the style editor independently of devices and projects.

Views in the style editor

Objects are grouped in the style editor according to their main visual features.

The following groups are available:

- Buttons, lines, and polygon-based objects
- Text-based objects
- Trend-based objects, such as trend view
- Graphic or value-based objects, such as value table.

Displaying predefined styles

Introduction

In the "Style" editor, you see predefined styles and create your own styles for representing display and operating objects in Runtime.

Open "Styles" editor

1. Double-click "Common data" in the project navigation.
2. Click the "Styles" editor. The editor opens.

Note

Style editor in the project navigation

The style editor is only visible in the project navigation if a device that supports styles, e.g., Comfort Panels, is created in the project.

3. The standard styles are displayed in the work area of the "Style" editor:

- WinCC Light
- WinCC Dark
- WinCC Fresh
- WinCC Wireframe.

Result

The settings for the groups and individual objects are displayed in the work area.

You will assign predefined styles or define your own styles in the next steps.

Predefined styles are write-protected and cannot be changed.

Managing styles

Deleting a style

You can delete styles that you have defined yourself.

1. Open the "Common data" folder in the project navigation.
2. Open the "Style" editor.
3. Select the style in the work area that you want to delete.

Note

You cannot delete the predefined styles of WinCC.

4. Select the "Delete" command in the shortcut menu.

The style is deleted.

Editing a style

You can edit styles that you have defined yourself.

1. Open the project library.
2. Select the released version of the style that you want to change.
3. Select "Edit type" in the shortcut menu.
   A new version "in progress" is created.
4. Make the required changes in the Inspector window.
5. Select the "Release version" command from the shortcut menu of the edited version to put the changes into effect.
   The changes made take effect.

Determining a standard style for a project

A project can be assigned a standard style to achieve a visual conformity of all display objects in the project.

1. Open the "Common data" folder in the project navigation.
2. Open the "Style" editor.
3. Activate a style in the work area.

The activated style is used as standard style in the project. When you add a device to the project, this device uses the standard style.

Changing the style for a specific device

Within a project, you change the style for a specific device.

1. Open the "Runtime settings" editor of the HMI device.
2. Select a device style under "General".

The selected style is applied to the device.

If the device style is not assigned, the device uses the standard style of the project.

10.1.2 Working with objects

10.1.2.1 Overview of objects

Introduction

Objects are graphics elements which you use to design the screens of your project.

The "Tools" task card contains all objects that can be used for the HMI device. Display the Task Card with menu command "View" by activating the "Task Card" option.

The toolbox contains various palettes, depending on the currently active editor. If the "Screens" editor is open, the toolbox contains the following palettes:

- "Basic objects"
  The basic objects include basic graphic objects such as "Line", "Circle", "Text field" or "Graphic view".
- "Elements"
  The elements include basic control elements, e.g. "I/O field", "Button" or "Gauge".
- "Controls"
  The controls provide advanced functions. They also represent process operations dynamically, for example Trend view and Recipe view.
- "Graphics"
  Graphics are broken down into subjects in the form of a directory tree structure. The various folders contain the following graphic illustrations:
  - Machine and plant areas
  - Measuring equipment
  - Control elements
  - Flags
  - Buildings

  You can create links to your own graphic folders. The external graphics are located in these folders and subfolders. They are displayed in the toolbox and incorporated into the project using links.
- "Libraries" task card
  In addition to the display and operating elements, the library objects are available. They are located within the palettes of the "Libraries" task card. A library contains preconfigured objects such as graphics of pipes, pumps, or preconfigured buttons. You can also integrate multiple instances of library objects into your project without having to reconfigure them.
  The WinCC software package includes libraries, e.g. "HMI Buttons & Switches".
  You can also store customized objects, and faceplates in user libraries.
  Faceplates are objects that you create from existing screen objects, and for which you define the configurable properties.

Note

HMI device dependency

Some of the toolbox objects are either available with restricted functionality, or not at all. This depends on the HMI device you are configuring. Unavailable properties of an object are displayed as deactivated, and cannot be selected.

Basic objects

Icon	Object	Instructions
	"Line"	-
	"Ellipse"	-
	"Circle"	-
	"Rectangle"	-
	"Text field"	One or more lines of text. The font and layout are adjustable.
	"Graphic view"	Displays graphics from external graphic programs, and inserts OLE objects. The following graphic formats can be used: "*.emf", "*.wmf", "*.dib", "*.bmp", "*.jpg", "*.jpeg", "*.gif" and "*.tif".

Elements

Icon	Object	Instructions
	"I/O field"	Outputs the values of a tag, and/or writes values to a tag. You can define limits for the tag values shown in the I/O field. To hide the operator input in runtime, configure "Hidden input".
	"Button"	Executes a function list, or a script as configured.
	"Symbolic I/O field"	Outputs the values of a tag, and/or writes values to a tag. A text from a text list is displayed in relation to the tag value.
	"Graphic I/O field"	Outputs the values of a tag, and/or writes values to a tag. A graphic from a graphics list is displayed in relation to the tag value.
	"Date/time field"	Outputs the system date and time, or the time and date from a tag. This allows the operator to enter new values. The display format is adjustable.
	"Bar"	The bar represents a value from the PLC in the form of a scaled bar graph.
	"Switch"	Toggles between two defined states. You can label a switch with text, or a graphic.

Controls

Icon	Object	Description
	"Alarm view"	Shows currently pending alarms or alarm events from the alarm buffer or alarm log.
	"Trend view"	Represents multiple curves with values from the PLC, or from a log.
	"User view"	Allows an administrator to administer users on the HMI device. It also allows an operator without administrator rights to change his password.

Icon	Object	Description
	"Recipe view"	Displays data records, and allows them to be edited.
	"System diagnostics view"	Provides an overview of all devices capable of diagnostics. Displays errors in the plant.

See also

Inserting an object (Page 7982)
Positioning an object (Page 7985)
Resizing an object (Page 7986)
Selecting multiple objects (Page 7988)
Aligning objects (Page 7990)
Moving an object forward or backward (Page 7991)
Show objects outside the screen area (Page 7992)
Rotating objects (Page 7993)
Flipping objects (Page 7995)
Inserting multiple objects of the same type (stamping tool) (Page 8006)
Repositioning and resizing multiple objects (Page 8008)
External graphics (Page 8008)
Managing external graphics (Page 8009)
Storing an external image in the graphics library. (Page 8011)
Basics on groups (Page 8013)
Overview of keyboard access (Page 8020)
Example: Inserting and configuring a rectangle (Page 8022)
Objects for Basic Panels (Page 8075)

10.1.2.2 Options for Editing Objects

Introduction

Objects are graphics elements which you use to design the screens of your project.

You have the following options for editing objects:

- Copying, pasting or deleting objects using the shortcut menu If you copy an object in a screen and the screen already includes an object of the same name, the name of the object is changed.
- Maintaining the standard size of the objects you are inserting or customizing their size on insertion.

- Moving an object in front of or behind other objects
- Rotating objects
- Mirroring objects
- Defining the tab sequence for objects
- Stamping: Inserting several objects of the same type
- Selecting several objects simultaneously
- Repositioning and resizing multiple objects
- You can assign external graphics to objects, e.g. in the Graphic View.
  You can view only the images you have previously stored in the graphic browser of your WinCC project.
  You can save graphics in the graphic browser:
  - Via drag & drop from the "Graphics" object group to the working area
  - As graphic files in the following formats: *.bmp, *.dib, *.ico, *.emf, *.wmf, *.gif, *.tif, *.jpeg or *.jpg
  - As an OLE object
    You either create a new OLE object or save an existing graphic file as an OLE object.
    To save an OLE object, an OLE-compatible graphics program must be installed on the configuration computer.

See also

Overview of objects (Page 7978)

10.1.2.3 Inserting an object

Introduction

In the "Screens" or "Reports" editor, insert the objects to the "Toolbox" task card. Use the mouse to drag the objects into the work area. You either keep the objects in their original size, or scale them up or down when you paste them.

In addition, you can copy or move objects via the clipboard from one editor to another, for example to transfer a screen object to a report. Alternatively, you can also use the mouse instead of the clipboard for copying and moving:

- Copying: <Ctrl + Drag&Drop>
- Moving: Drag&drop

Note
Basic Panels

The "Reports" editor is not available for Basic Panels.

Requirement

The "Tools" task card is open.

Inserting objects in their standard size

1. In the "Toolbox" task card, select the desired graphic object or the desired graphic in the WinCC graphics folder.
   When you move the cursor across the work area, it turns into a crosshair with an appended object icon.
2. Click the location in the work area where you want to insert the object or graphic.
   The object is inserted with its standard size at the desired position in the work area.

Alternatively, double-click the object in the "Toolbox" task card.

Copying an object

1. Select the desired object.
2. Select "Copy" in the shortcut menu.
3. Click the desired location and select "Paste" in the shortcut menu.

WinCC inserts a copy of the object at the desired location. You can only change the properties that are appropriate for the relevant context.

Example: In the "Screens" editor, you can set for I/O fields and the mode for input and output. In the "Reports" editor, the mode is set to "Output".

Original and copy are not interconnected and are configured independently from one another.

Inserting lines

1. Select the desired graphic object in the "Tools" task card.
2. Click on a location in the work area. A line in the standard size is inserted.

Inserting a polygon or polyline

1. Select the desired object "Polyline" or "Polygon" in the "Tools" task card.
2. Click on a location in the work area. This fixes the starting point of the object.
3. Click another location in the work area. A corner point is set.
4. For every additional corner point, click on the corresponding location in the work area.
5. Double-click on a location in the work area. The last corner point is set.
   This fixes all the points of the polygon or polyline.

Note

Basic Panels

The "Polyline" and "Polygon" objects are not available for Basic Panels.

Note

If you want to insert several objects of the same type, use the "Stamp" function. This avoids having to reselect the object in the "Tools" task card every time before inserting it. To do so, select the icon in the toolbar of the "Tools" task card.

See also

Overview of objects (Page 7978)

10.1.2.4 Deleting an Object

Introduction

You can delete objects individually or with a multiple selection.

Requirement

You have opened the work area containing at least one object.

Procedure

1. Select the object that you want to delete.
   To delete multiple objects, keep the <Shift> key pressed and select the objects to be deleted one after the other. Alternatively, drag and maximize an area around the desired objects with the mouse.
2. Select "Delete" from the shortcut menu.

Result

The selected objects are deleted.

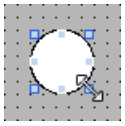
See also

Overview of objects (Page 7978)

10.1.2.5 Positioning an object

Introduction

When you select an object, it is enclosed by a rectangle with resizing handles. This rectangle is the rectangle which surrounds the object. The position of an object is defined by the coordinates of the top left corner of the rectangle surrounding the object.



Note

If the position is outside the work area the object is not displayed in Runtime.

Position and align

You have the possibility of having a grid displayed in the work area. You have three options for easier positioning of objects:

- "Snap to grid" When you resposition objects, they are automatically snapped and pasted to the grid. If you hold down the <Alt> key, the object is no longer snapped to the grid.
- "Snap to objects" When you reposition objects, they are displayed with help lines. You can use other objects for orientation during positioning.
- "None": You position the objects in any position.

You activate and deactivate the grid and options as follows:

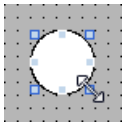
- In the "Options > Settings > Visualization > Screens" menu
- In the "Layout > Grid" task card

Requirement

You have opened the work area containing at least one object.

Procedure

1. Select the object you want to move.
   The selected object is framed by a rectangle with resizing handles.

2. Left-click the object and keep the mouse button pressed.

3. Move the mouse pointer onto the new position.
   The contour of the object moves with the mouse and displays the new position for the object.

   The object initially remains at its original position.
4. Now release the mouse button.
   The object is moved into the position indicated by the contour of the selection rectangle.

Alternative procedure

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter the X and Y values for the position under "Position & Size".

Result

The object appears at its new position.

See also

Overview of objects (Page 7978)

10.1.2.6 Resizing an object

Introduction

When you select an object, it is enclosed by a rectangle with handles. You have the following options of resizing an object:

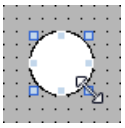
- Drag the handles using the mouse.
- Modify the "Size" property in the Inspector window.

Requirement

You have opened the work area containing at least one object.

Procedure

1. Select the object you want to resize.
   The selection rectangle appears. The following figure shows a selected object:



2. Drag a resizing contact of the rectangle to a new position.
   The size of the object changes.
   – The size of the object is aligned to the grid pattern, provided the "Snap to grid" function is set.
   – Press <ALT> to disable this function while you drag the object.
     In order to scale the object proportionally, keep the <Shift> key pressed while changing the size with the mouse.

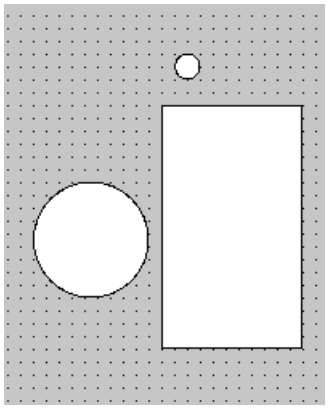
Alternative procedure

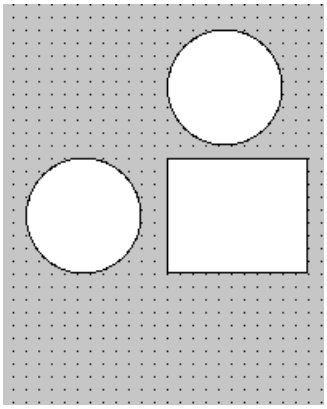
1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter the size of the object under "Position & Size".

Harmonizing the object size

1. Select the objects.
2. Now, click one of the following buttons: or or
   The size of the selected objects is matched to each other.

The following screen shows how the selected objects are adapted to the height of the reference object:





Icon	Description
	Aligns the selected objects to the width of the reference object.
	Aligns the selected objects to the height of the reference object.
	Aligns the selected objects to the width and height of the reference object.

Result

The object now appears with its new size.

See also

Overview of objects (Page 7978)

10.1.2.7 Selecting multiple objects

Introduction

Select all objects you want to align with each other or to change global properties. This procedure is called "multiple selection."

The Inspector window shows all the properties of the selected objects.

You now have several options of selecting multiple objects:

- Draw a selection frame around the objects.
- Hold down the <Shift> key, and click the required objects.

Selection frame of a multiple selection

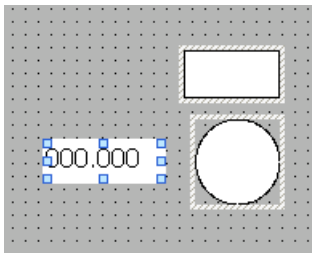
The selection frame surrounds all objects of a multiple selection. The selection frame is comparable with the rectangle that surrounds an individual object.

The selection frame is not visible. When you have made your multiple selection, the following frame is displayed:

- The reference object is indicated by the rectangle around it.
- The other selected objects are indicated by a dashed-line frame.

Specifying a reference object

The reference object is the object upon which the other objects are oriented. The reference object is framed by a rectangle with handles. The following figure shows a reference object with two other selected objects:



You have the following options to specify the reference object:

- Select the objects via multiple selection. The object selected first is then the reference object.
- Draw a selection frame around the objects. The reference object compiled automatically. If you wish to specify a different object within the selection as the reference object, click on the desired object. This action does not cancel your multiple selection.

Requirement

You have opened the work area containing at least two objects.

Selecting multiple objects with a selection frame

1. Position the mouse pointer in the work area close to one of the objects to be selected.
2. Hold down the mouse button, and draw a selection frame around the objects to be selected.

Or:

1. Hold down the <Shift> key.
2. Click the relevant objects, working in succession.
   All the selected objects are identified by frames.
   The object selected first is identified as reference object.

Note

To remove an object from the multiple selection, press <SHIFT>, hold it down and then click the relevant object once again.

Result

Multiple objects are selected. One of those is identified as the reference object. You can now perform the following steps:

- Changing the object properties of all the objects
- Resizing all the objects by the same ratio, by dragging the selection frame to increase or reduce the size

- Moving all the objects in one group
- Aligning the objects to the reference object

See also

Overview of objects (Page 7978)

10.1.2.8 Aligning objects

Procedure

1. Select the objects via multiple selection.
2. Specify an object as the reference object.
3. Select the desired command in the toolbar or the shortcut menu - see table below.
   The selected objects will be aligned.

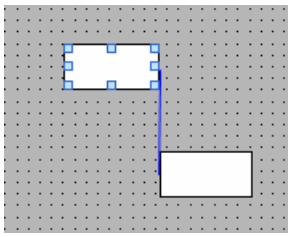
Aligning objects flush

The selected objects will be aligned flush to the reference object.

Icon	Description
	Aligns the selected objects to the left edge of the reference object.
	Aligns the selected objects to the vertical center axis of the reference object.
	Aligns the selected objects to the right edge of the reference object.
	Aligns the selected objects to the upper edge of the reference object.
	Aligns the selected objects to the horizontal center axis of the reference object.
	Aligns the selected objects to the lower edge of the reference object.
	Centers the selected objects to the center points of the reference object.
	Centers the selected objects vertically in the screen.

Snap to object

When you reposition objects, they are displayed with help lines. You can use other objects for orientation during positioning.



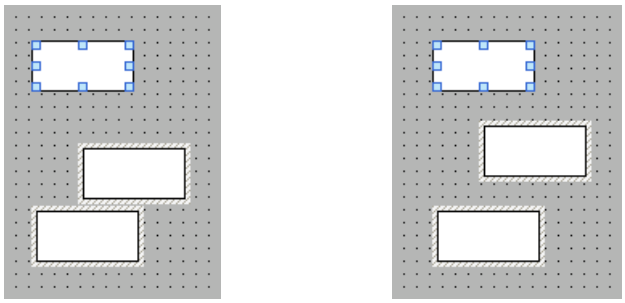
If you are working with the keyboard, press the Alt key. When you move the selected object with the arrow keys, the next anchor point is displayed.

Distributing objects evenly

You need at least three selected objects. A reference object is not required.

1. Select the objects.
2. Click one of the buttons "Distribute horizontally equal" or "Distribute vertically equal". The selected objects are distributed at equal distances.

The following screen shows how you align the vertical spacing of the selected objects:



Icon	Description
	Aligns the horizontal distance between the objects. The position of the objects on the extreme left and right side remains unchanged. All other objects are distributed evenly between them.
	Aligns the vertical distance between the objects. The position of the objects at the extreme top and bottom (right and left) remains unchanged. All other objects are distributed evenly between them.

See also

Overview of objects (Page 7978)

10.1.2.9 Moving an object forward or backward

Introduction

You can use the "Order" functions in the shortcut menu of a selected object or in the toolbar to move a selected object in front of or behind other objects within an object layer.

Note

ActiveX controls are always positioned in front of an object layer (.NET property).

Requirement

You have opened a screen which contains a layer with multiple objects.

Procedure

1. Select the object you want to move forward or backward.
2. Select the "Sort" command and one of the following commands from the shortcut menu:

Icon	Description
	Moves the selected object before all the other objects of the same layer
	Moves the selected object to the lowest position in the same layer
	Moves the selected object up by one position
	Moves the selected object down by one position

Alternative procedure

1. Open the "Layers" palette of the "Layout" task card.
2. Navigate to the required object.
3. Hold down the mouse button, and drag the object in the tree topology to the required position in the layer.
4. Now release the mouse button.

Result

The object is moved up or down.

See also

Overview of objects (Page 7978)

10.1.2.10 Show objects outside the screen area

Introduction

If you assign objects to positions that are outside the configurable area, these objects will be hidden. The functions of the "Objects outside the visible area" palette in the "Layout" task card are used to move these objects back into the screen.

Requirement

- You have opened a screen which contains objects that are outside the configurable area.
- The "Layout" task card is open.

Procedure

1. Open the "Layout > Objects outside the area" task card.
   This displays a list of objects that are outside the configurable area.
2. Select the the object which you want to move back into the screen from the list.
3. Select "Move to screen" in the object shortcut menu.

Alternatively open the "Layout > Layer" task card. Objects outside the area are indicated by the icon. If you click this icon, the object is moved back into the screen.

Result

The objects are moved to the configurable area.

See also

Overview of objects (Page 7978)

10.1.2.11 Rotating objects

Introduction

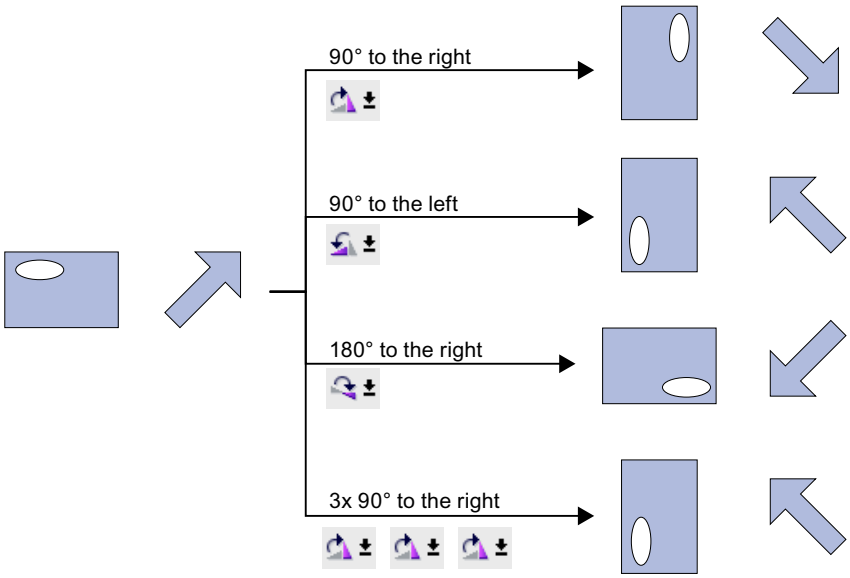
You can rotate a suitable object clockwise or counterclockwise around its center axis in steps of 90°.

Note

Not all the objects can be rotated. Some objects that can be rotated in screens cannot be rotated in reports.

You can also rotate multiple objects using the multiple selection function. Certain WinCC objects such as buttons cannot be rotated.

The alignment of elements in an object will change in a rotated object. The following figure shows how a rectangle and an ellipse behave under the different commands for rotating an object:



Requirement

You have opened the work area containing at least one object.

Procedure

1. Select the object that you want to rotate.
2. Click one of the following toolbar icons:
   , to rotate the object clockwise around its center point. The angle of rotation is 90°.
   , to rotate the object counterclockwise around its center point. The angle of rotation is 90°.
   , to rotate the object clockwise by 180°.

Result

The object is shown at its new angle.

See also

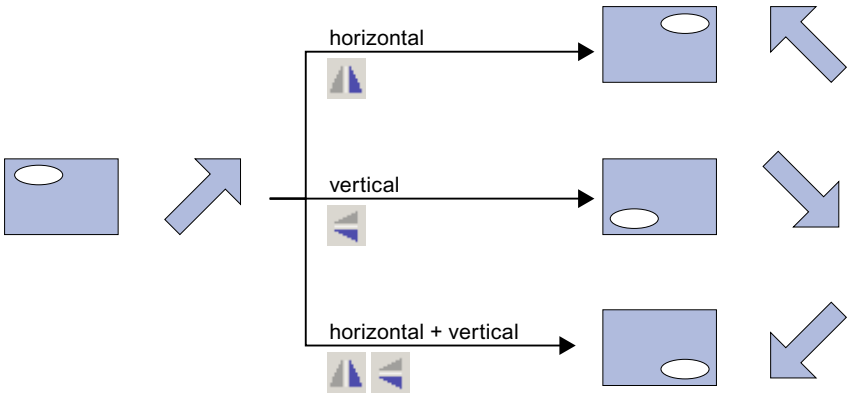
Overview of objects (Page 7978)

Ellipse (Page 8081)

10.1.2.12 Flipping objects

Introduction

You can flip an object along its vertical or horizontal center axis. The alignment of elements in an object will change when you flip an object. The following figure shows how a rectangle and an ellipse behave under the different commands for flipping an object.



Requirement

You have opened a screen which contains at least one object.

Procedure

1. Select the object that you want to flip.
2. Click the "Flip" command in the shortcut menu and select one of the options displayed:
   – , to flip the selected object along its vertical center axis.
   – , to flip the selected object along its horizontal center axis.

Result

The object is shown at its flipped position.

See also

Overview of objects (Page 7978)

Ellipse (Page 8081)

10.1.2.13 Designing an object

Introduction

You design the border and background of an object.

Requirements

A line has been created in a screen.

Procedure

1. Select the line on your screen.
2. In the Inspector window, select "Properties > Properties > Appearance":
3. Select "Dash" as the style.
4. To display the dashed line in two colors, select the line width "1".
5. Select the setting "Arrow" in the "Line ends" area.

Result

The line is displayed in two colors as a dashed line. The end of the line is an arrow.

10.1.2.14 Designing the fill pattern

Introduction

WinCC lets you design the background color and the fill pattern of an object. The design options change in the Inspector window depending on object for which you are making the filling pattern.

For certain objects, you can not only define the color, but also a transparent background or a background with a color gradient.

Note

Device-specific nature of the properties

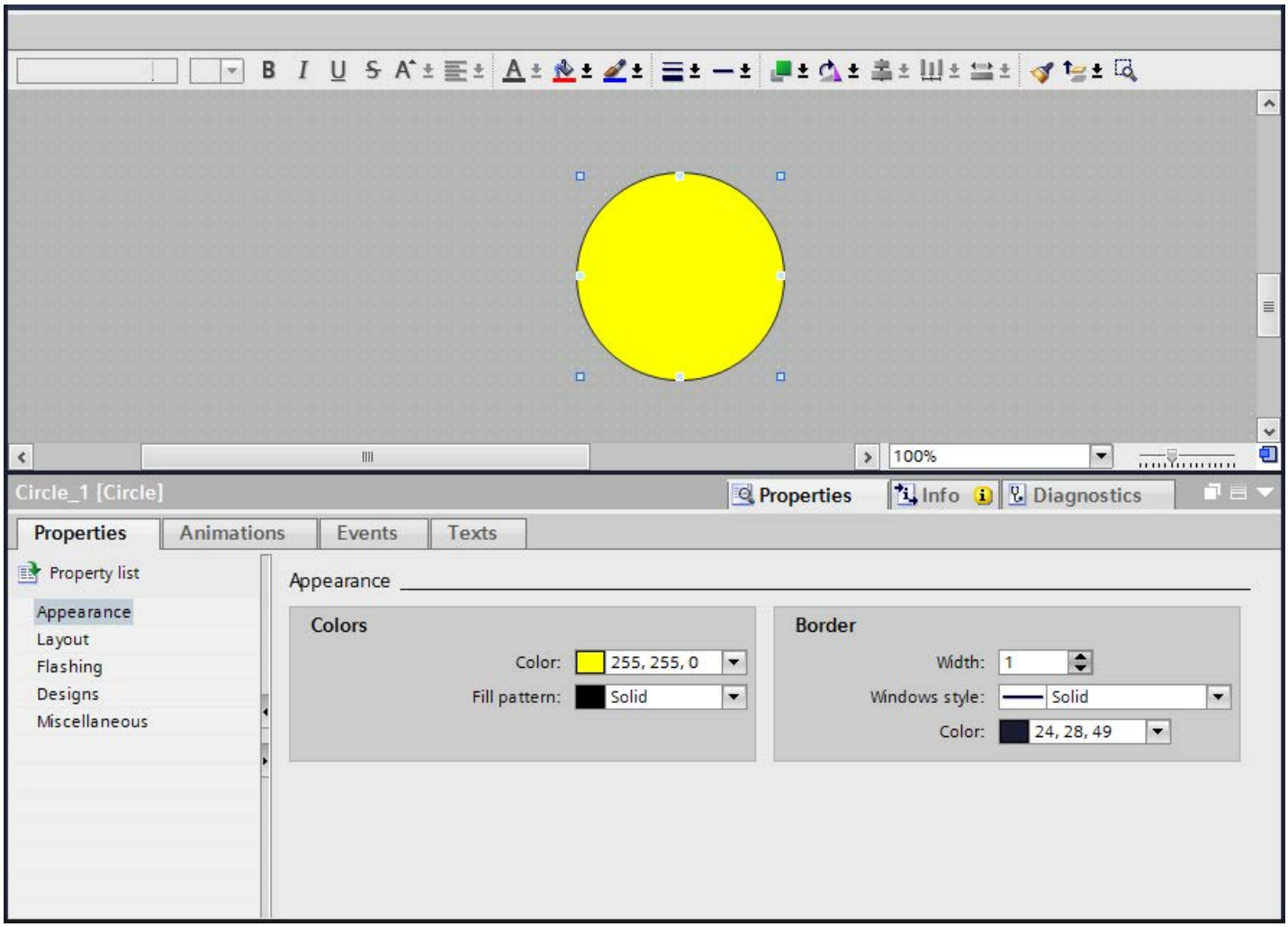
The fill style that is available depends on the object and the HMI device used.

Requirement

The object has been created and selected.

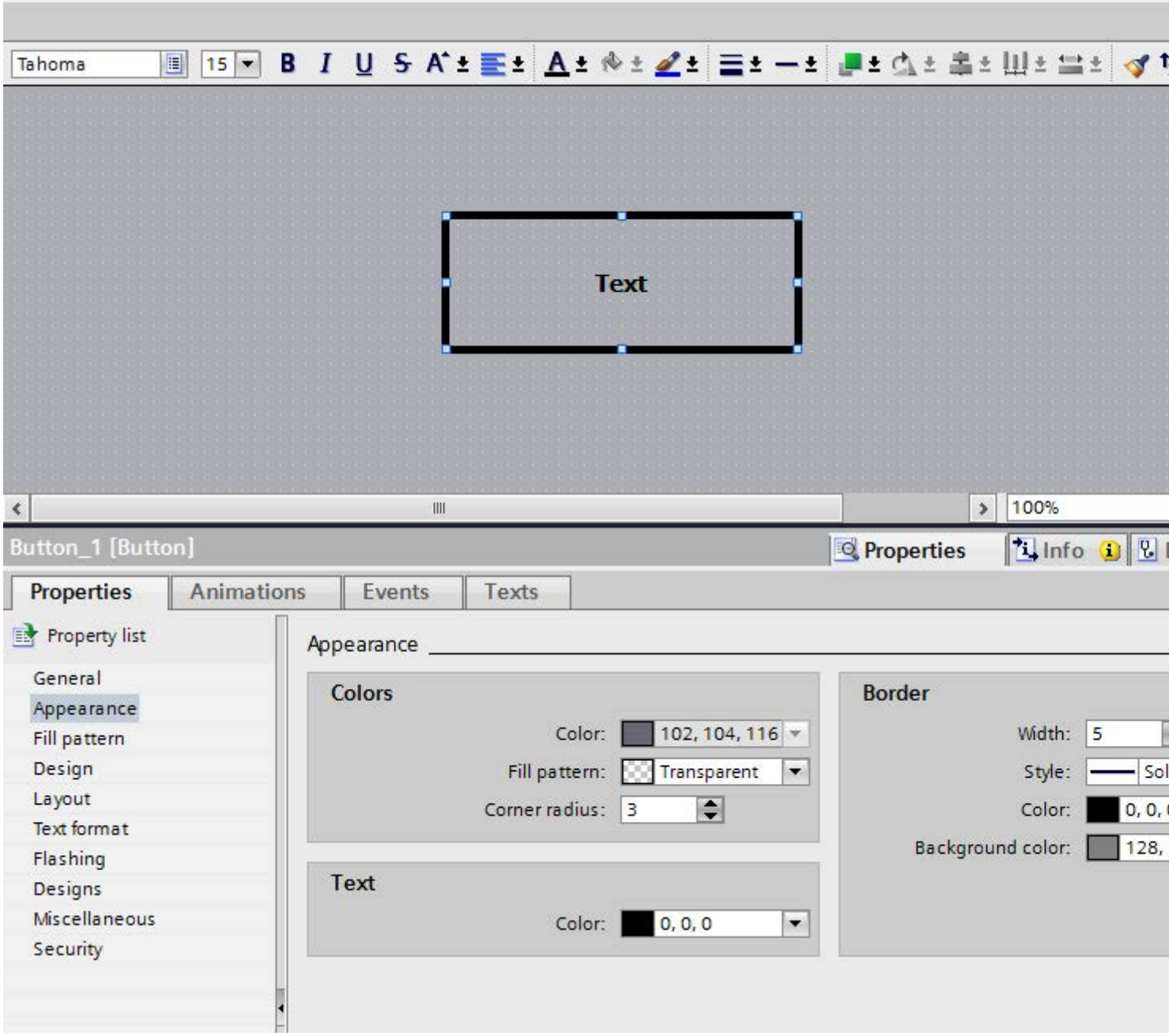
Designing the background color of an object

1. In the Inspector window, select "Properties > Properties > Appearance".
2. Select a color for the background of the object, for example, yellow.
   The object is filled with the selected color.



Designing the fill pattern of an object

1. In the Inspector window, select "Properties > Properties > Appearance".
2. To define a transparent background for the object, select transparent.
   You can find information about designing a fill pattern with a color gradient in the section Defining color gradients (Page 8004).
   The object is shown as transparent.



You can optionally set the fill pattern in the Inspector window under "Properties > Properties > Fill Pattern".

See also

Defining color gradients (Page 8004)

10.1.2.15 Formatting graphics in an object

Introduction

WinCC lets you insert graphics into some objects and format them. You can change the size, orientation and distances of a graphic to the object border. You change the properties of the graphic in the Inspector window of the relevant object.

Requirement

The selected object is in graphic mode.

The selected object contains at least one graphic.

Resizing a graphic

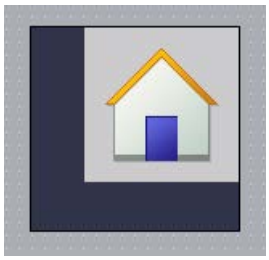
1. In the Inspector window, select "Properties > Properties > Layout".
2. Select "No stretching of picture" or "Stretch picture".
   The contained graphic is displayed in its original size or stretched to the size of the object.

Note

To adjust the object size to the contained graphic, select "Fit object to contents".

Aligning a graphic horizontally and vertically

1. In the Inspector window, select "Properties > Properties > Layout".
2. To determine the horizontal position of the graphic, select "Right", for example.
3. To determine the vertical position of the graphic, select "Top", for example.
   The graphic is then shown at the top right of the object.



Defining the distance from the object border

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter the value for the distance to the object border, for example, "20" for the top border. The graphic is shown with a distance of 20 pixels to the top border of the object.

Icon	Description
	Sets the distance from the left border of the object.
	Sets the distance from the right border of the object.
	Sets the distance from the top border of the object.
	Sets the distance from the bottom border of the object.

See also

Button (Page 8099)

Switch (Page 8098)

10.1.2.16 Formatting text in an object

Introduction

Some objects, for example, the I/O field, support text within the object.

You have several options to align text.

Configuring the distance from the object border

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter the value "5" for the left border, for example.

The text is aligned five pixels from the left border of the object.

Aligning the text position

1. In the Inspector window, select "Properties > Properties > Text format".
2. Select the horizontal alignment, for example, centered.
3. Select the vertical alignment, for example, top.
   The text is shown at the top center in the object.

Defining the text orientation

1. In the Inspector window, select "Properties > Properties > Text format".
2. Select the orientation of the text, for example, vertical, right.
   The text flow is shown from bottom to top.

10.1.2.17 Designing table-based objects

Introduction

WinCC offers various properties for designing display and operating objects.

You change the properties of table-based objects in the Inspector window of the relevant object.

Designing colors

Select the colors for font, selections and areas under "Properties > Properties > Appearance".

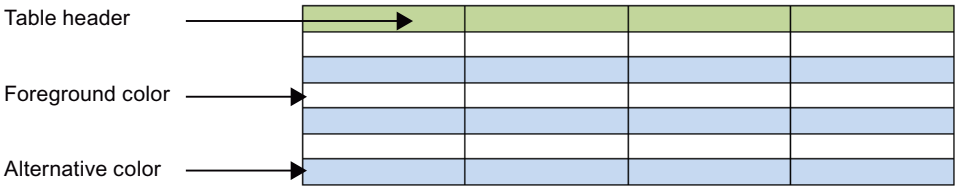


Figure 10-1 Selecting table properties

Designing the border of the table header

1. In the Inspector window, open "Properties > Properties > Table header border".
2. Select "Solid", for example, as the style.

3. Select 5, for example, as the "Width".
4. Select red, for example, as a foreground color.

The border of the table header is shown with a red border and width of 5 pixels. With "Solid" border style, only the foreground color is visible.

Colors and color gradient in the table header

1. Select the table-based object in the screen.
2. In the Inspector window, open "Properties > Properties > Table header fill pattern".
3. Select the fill pattern, for example "Horizontal gradient".
4. Select a background color, for example, blue, under "Gradient".
5. Activate "Gradient 1".
6. Select a color for "Gradient 1", for example, white.
7. Select a "Width" for the color gradient, for example, 12.
8. Activate Gradient 2.
9. Select a color for Gradient 2, for example, yellow.
10. Select a "Width" for the color gradient, for example, 10

The table header is displayed with a color gradient.

See also

Designing a border (Page 8002)

10.1.2.18 Designing a border

Introduction

WinCC offers various properties for designing display and operating objects.

The I/O field example below shows the possible settings for designing borders.

You change the appearance in the Inspector window of the relevant object.

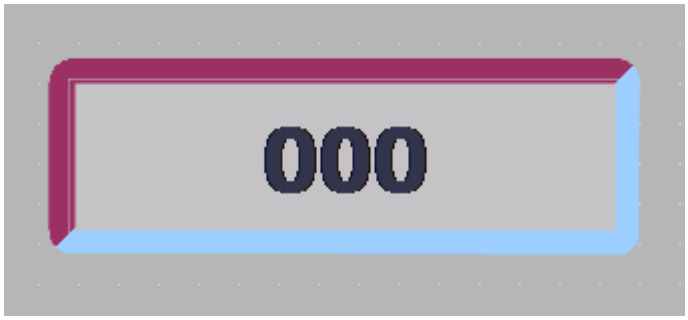
Designing a border

1. Open "Properties > Properties > Appearance".
2. Select the width of the border, e.g. "5".
3. Select the style of the border, e.g. double line.
4. Select the foreground color.
5. Select the background color.

Figure 10-2 "Double line" border style

Border style

The figure below shows a border in "3D style".



The figure below shows a border in "Solid" style.

Note

Device-specific nature of the properties

The border style that is available depends on the object and the HMI device used.

See also

Designing table-based objects (Page 8001)

10.1.2.19 Defining color gradients

Introduction

For the objects in WinCC, color gradients can be specified as background for various surfaces.

Change the category in the Inspector window, depending on which surface you fill with a color gradient. The procedure remains the same.

The following section describes how to configure the color gradient of buttons.

Horizontal color gradient with two colors

1. Select an object with buttons, for example, a button.
2. In the Inspector window, select "Properties > Properties > Fill Pattern".
3. Select "Fill Pattern > Horizontal gradient".
4. Select a background color for the vertical color gradient, for example, orange.
5. Activate "Gradient 1".
6. Select a "Color" for Gradient 1, for example, red.
7. Select a "Width" for Gradient 1, for example, "10".

The background of the button is displayed in orange.

For the horizontal color gradient, Gradient 1 is shown from the left border. Gradient 1 is 10 pixels wide.

Vertical color gradient with three colors

1. Select a button in the screen.
2. In the Inspector window, select "Properties > Properties > Fill Pattern".
3. Select the "Vertical gradient" background.
4. Select a background color under "Gradient", for example, orange.
5. Activate "Gradient 1".
6. Select a color for "Gradient 1", for example, red.
7. Select a "Width" for the color gradient, for example, 8.
8. Activate Gradient 2.

9. Select a color for Gradient 2, for example, yellow.
10. Select a "Width" for the color gradient, for example, 10.

	← Color Gradient 1
	← Background color
	← Color Gradient 2

For the vertical color gradient, Gradient 1 is shown from top to bottom. Gradient 1 is 8 pixels wide.

The background of the button is displayed in orange.

Gradient 2 is shown at the bottom border. Gradient 2 is 10 pixels wide.

10.1.2.20 Applying predefined styles

Introduction

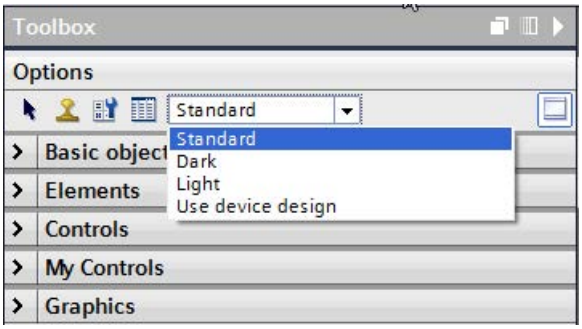
You can assign predefined styles to objects and faceplates in Runtime. You can change the background color of the display and operating elements using predefined styles. In this way, you harmonize the display in Runtime.

Requirement

The "Tools" task card is open.

Setting predefined styles

1. Select the object that you want to insert in the "Toolbox" task card.
2. Select one of the following options in the "Toolbox" task card:
   - "Default" to use the default style
   - "Dark" to use the dark color scheme
   - "Light" to use the light color scheme
   - "Use device design" to use the settings from the current device design



3. Insert the required object in the work area.
   The object is displayed in the selected style.
4. The objects are created in the selected style as long as the style remains activated in the toolbar.
   To return to the default style, select the "Default" option in the toolbar.

See also

Working with styles (Page 7976)

10.1.2.21 Inserting multiple objects of the same type (stamping tool)

Introduction

WinCC offers the possibility to "stamp" several objects of the same type directly one after the other, i.e. paste without having to reselect the object every time. In addition you have the possibility of multiplying an object that has already been inserted.

Requirement

The "Tools" task card is open.

Inserting several objects of one type

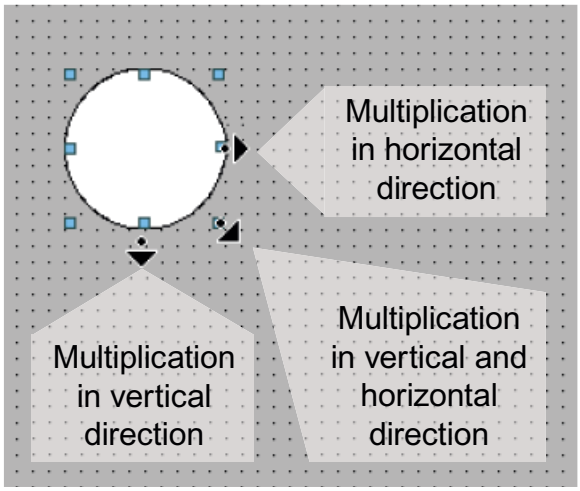
1. Select the object that you want to insert in the "Tools" task card.
2. Click the icon in the toolbar of the "Tools" task card.
   The "Stamp" function is activated.
3. To insert the object with its standard size, click the relevant insertion position in the work area.
   To insert the object with another size, position the mouse pointer at the desired location in the work area. Press the left mouse button and drag the object to the required size.
   The object is inserted in the work area as soon as you release the mouse button.
4. Repeat step 3 to insert further objects of the same type.
5. Click the icon again.
   The "Stamp" function is deactivated.

Note

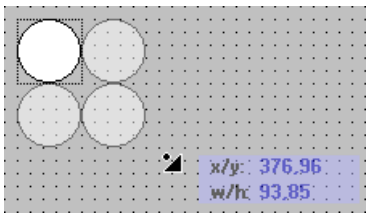
You can copy existing objects using the drag-and-drop +<CTRL> function. The existing object is not moved in this case. You paste a copy of this object into the new position instead.

Inserting and multiplying an object

1. Insert the desired object from the "Tools" task card.
2. Press the <Ctrl> key and position the cursor on one of the handles displayed in the figure shown below.



3. Drag the handles to the right and/or down while keeping the left mouse button pressed.
4. The object is multiplied depending on available space if you keep moving the cursor.



Result

You have pasted and stamped an object in a screen.

See also

Overview of objects (Page 7978)

10.1.2.22 Repositioning and resizing multiple objects

Possible modifications

After you have selected multiple objects, you edit them:

- Shift using the mouse
  - To change the absolute position of the marked objects, position the mouse pointer over an object, and shift the multiple selection with the mouse button pressed.
  - To resize all the objects by the same ratio, grab the resizing handles of the reference object.
- Move over the work area with the icons of the toolbar
  - Change the position of the marked objects with respect to each other
  - Align the height and width of the marked objects
- Moving with the shortcut menu commands of the work area
  - Change the position of the marked objects with respect to each other
  - Align the height and width of the marked objects

See also

Overview of objects (Page 7978)

10.1.2.23 External graphics

Introduction

You can use graphics created with an external graphic program in WinCC. To use these graphics you store them in the graphic browser of the WinCC project.

You can save graphics in the graphic browser:

- When you drag-and-drop graphics objects from the "Graphics" pane into the work area, these are stored automatically in the graphic browser. The graphic names are numbered in the order of their creation, for example, "Graphic_1." Use the <F2> function key to rename the graphic.
- As a graphic file with the following formats:
  *.bmp, *.ico, *.emf, *.wmf, *.gif, *.tif, *.png, *.jpeg or *.jpg
- As an OLE object that is embedded in WinCC and is linked to an external graphic editor. In the case of an OLE link, you open the external graphic editor from WinCC. The linked object is edited using the graphic editor. An OLE link only works if the external graphic editor is installed on your PC, and supports OLE.

Use of graphics from the graphic browser

Graphics from the graphic browser are used in your screens:

- In a Graphic view
- In a graphics list
- As labeling for a button/function key

Transparent graphics

In WinCC you also use graphics with a transparent background. When a graphic with a transparent background is inserted into a graphic object of WinCC, the transparency is replaced by the background color specified for the graphic object. The selected background color is linked firmly with the graphic. If you use the graphic in another graphic object of WinCC, this object is displayed with the same background color as the graphic object that was configured first. If you want to use the graphic with different background colors, include this graphic in the graphic browser again under a different name. The additional background color is configured when the graphic is used at the corresponding graphic object of WinCC.

Managing graphics

An extensive collection of graphics, icons and symbols is installed with WinCC, for example:

In the Toolbox window of the "Graphic" pane the graphic objects are structured by topic in the "WinCC graphics folder." The link to the WinCC graphics folder cannot be removed, edited or renamed.

The "Graphics" pane is also used to manage the external graphics. The following possibilities are available:

- Creating links to graphics folders
  The external graphic objects in this folder, and in the subfolders, are displayed in the toolbox and are thus integrated in the project.
- Editing folder links
- You open the program required for editing of the external graphic in WinCC.

See also

Overview of objects (Page 7978)

10.1.2.24 Managing external graphics

Introduction

External graphics that you want to use in WinCC are managed in the "Screens" editor by using the "Tools" task card in the "Graphics" pane.

Requirement

- The "Screens" editor is open.
- The "Tools" task card is open.
- The graphics are available.
- The graphics have the following formats:
  *.bmp, *.ico, *.emf, *.wmf, *.gif, *.tif, *.jpeg or *.jpg

Creating a folder link

1. Click "My graphics folder."
2. Select "Link" in the shortcut menu.
   The "Create link to folder" dialog is opened. The dialog suggests a name for the folder link.
3. Edit the name as required. Select the path containing the graphic objects.
4. Click "OK" to confirm your input.
   The new folder link is added to the "Graphics" object group. The external graphics that are located in the target folder and in sub-folders are displayed in the toolbox.

Editing folder links

1. Select the folder link to edit.
2. Select the "Edit link..." command from the shortcut menu.
   The "Create link to folder" dialog is opened.
3. Edit the name and path of the folder link as required.
4. Click "OK" to confirm your input.

Renaming the folder link

1. Select the folder link to rename.
2. Select "Rename" from the shortcut menu.
3. Assign a name to the new folder link.

Removing a folder link

1. Select the folder link you want to delete.
2. Select "Remove" in the shortcut menu.

Edit external graphics

1. Select the graphic you want to edit.
2. Select the "Edit graphic" command from the shortcut menu.
   This opens the screen editor associated with the graphic object file.

Editing graphics folders from WinCC

1. Select the graphic you want to edit.
2. Select the "Open parent folder" command from the shortcut menu.
   The Windows Explorer opens.

See also

Overview of objects (Page 7978)

10.1.2.25 Storing an external image in the graphics library.

Introduction

To display graphics that have been created in an external graphics program in your screens, you will first have to store these graphics in the graphics browser of the WinCC project.

Requirement

- A screen has been created.
- A Graphic View is inserted in the screen.
- The Inspector window of the Graphic view is open.

To store an external graphic in the image browser:

- A graphic is available.

To store an OLE object in the browser:

- An OLE-compatible graphics program is installed on your configuration computer.

Save graphics file

1. Open the Windows Explorer.
2. Select the graphic that you want to store.
3. Drag-and-drop the graphic into the graphic browser.

Creating and saving a new graphic as an OLE object

1. Select the Graphic view on your screen.
2. In the Inspector window, select "Properties > Properties > General":
3. Open the graphic selection list.
4. Click .

5. The "Insert object" dialog box opens.

Note

In addition, the dialog "External application running..." will open. The dialog will not close until you exit the external application.

6. From the "Insert object" dialog box, select "New" and an object type. The settings in "Settings > "OLE settings" determine which object types are shown.
7. Click "OK." The associated graphic program is opened.
   When you are finished creating graphics, end the graphic programming software with "File > Close" or "File > Close & return to WinCC."
   The graphic will be stored in the graphic programming software standard format and added to the graphic browser.

Inserting created graphics in WinCC

Note

A new graphic object created as OLE object may not be directly accepted in WinCC when you save it to an external graphics program.

1. Reopen the dialog for inserting a graphic.
2. From the "Insert object" dialog box, select "Create from file."
3. Click the "Browse" button.
4. Navigate to the created graphic and select it.

Saving an existing graphic object as an OLE object

1. In the Inspector window, select "Properties > Properties > General":
2. Open the graphic selection list.
3. Click .
4. The "Insert object" dialog box opens.

Note

In addition, the dialog "External application running..." will open. The dialog will not close until you exit the external application.

5. From the "Insert object" dialog box, select "Create from file."

6. Click the "Browse" button.
7. Use the dialog to help you navigate to the folder in which the graphic file is saved.

Note

To import graphics files, note the following size restrictions:

*.bmp, *.tif, *.emf, *.wmf ≤4 MB

*.jpg, *.jpeg, *.ico, *.gif "*≤1 MB

Result

The image file is now stored in your image browser. It is shown in a screen with a Graphic view , or is added as a list element in an image list.

You can double-click OLE objects in your library to open them for editing in the corresponding graphic editor. When you have finished editing graphics, end the graphic programming software with "File > Close" or "File > Close & return to WinCC." The changes are applied to WinCC.

See also

Overview of objects (Page 7978)

10.1.2.26 Working with object groups

Basics on groups

Introduction

Groups are several objects that are grouped together with the "Group" function. You edit a group in the same way as any other object.

Overview

WinCC offers the following methods for editing multiple objects:

- Multiple selection
- Creating object groups

Editing mode

To edit an individual object in a group, select the object in the "Layout > Layers" task card.

Alternatively select "Group > Edit group" in the object group's shortcut menu.

Hierarchical groups

You add further objects or groups to extend a group. The group is enlarged by the new objects and is structured hierarchically in main and sub-groups. Such hierarchical groups must be broken up in stages. You also break up the group in the same order in which you grouped the objects or groups. It takes exactly the same number of steps to break up these hierarchical groups as it did to create them.

Rectangle surrounding the object

Only one surrounding rectangle is now displayed for the whole group. The surrounding rectangles of all objects are displayed for a multiple selection on the other hand.

Layers

All objects of a group are located in the same layer.

See also

Overview of objects (Page 7978)

Creating object groups

Introduction

The "Group" command combines multiple objects to form a group.

You can change the size and position of the group. The following rules apply:

- The system automatically adapts the position coordinates of the grouped objects when you reposition the group. The relative position of the grouped objects to the group is not affected.
- The system automatically adapts the height and width of the grouped objects in proportion to a change of the group size.
- To change the size of the group proportionately, hold down the <Shift> key and drag the rectangle around the object until has the required size.

Note

To create a hierarchical group, organize the individual groups like objects.

Requirement

- You have opened a screen which contains at least two objects.

Creating object groups

1. Select all the objects you want to organize in a group.
2. Select the command "Group > Group" from the shortcut menu.

The objects of the group are displayed with a rectangle around the objects.

Grouping objects within a group

1. Select the group you want to edit.
2. Select the command "Group > Edit group" from the shortcut menu.
   The group that you are editing is highlighted by a red frame.
3. Select the objects of a group that you want to combine into a subgroup.
4. Select the command "Group > Group" from the shortcut menu.

A subgroup with the objects is created.

Adding objects into an existing group

1. Select the group to which you want to add objects.
2. Press the <Shift> key and select the object you want to add to the group.
3. Select the "Group > Add to group" command from the shortcut menu.

The object is added to this group.

Alternative procedure

You can also edit groups in the "Layout" task card. Using drag-and-drop you can also easily edit nested groups in the "Layers" pane.

Result

The selected objects are combined in a group. The multiple selection rectangle becomes the rectangle surroundings the objects in the group. The handles are shown only for the group. The group in in the active layer.

Ungroup

Introduction

Select the "Ungroup" command to split a group into the original object fractions.

Requirement

- You have opened a screen that contains a group.

Ungroup

1. Select the group.
2. Select the "Group > Ungroup" command from the shortcut menu.

Ungrouping a group within a group

1. Select the higher-level group.
2. Select the command "Group > Edit group" from the shortcut menu.
   The group that you are editing is highlighted by a red frame.
3. Select the lower-level group.
4. Select the "Group > Ungroup" command from the shortcut menu.

Result

The lower-level group is ungrouped. The objects are assigned to the next higher group.

Alternative procedure

You can also edit groups in the "Layout" task card. Using drag-and-drop you can also easily edit nested groups in the "Layers" pane.

Adding objects to a group

Introduction

The "Add objects to group" command is used to add objects to a group, without ungrouping it first.

Requirements

A screen with one group and at least one other object is opened.

Procedure

1. Select the group.
2. Press the <Shift> key and select the object you want to add to the group.
3. Select the "Group > Add to group" command from the shortcut menu.

Result

The group consists of the original objects, and the newly-added objects. The added objects are arranged at the front of the group.

Alternative procedure

You can also edit groups in the "Layout" task card. Using drag&drop you can also easily edit hierarchical groups in the "Layers" palette.

Removing Objects from the Group

Introduction

You use the "Remove objects from group" command to remove individual objects from a group, without ungrouping it first.

You do not have to remove the object from the group to edit an object in a group. You can edit the objects of a group individually.

Requirement

- You have opened a screen that contains a group.

Removing objects from a group

To remove an object from a group:

1. Select the group.
2. Select the command "Group > Edit group" from the shortcut menu.
   The group you want to edit is highlighted by a red frame.
3. Select all objects in the group that you want to remove from the group.
4. Select the "Group > Remove from group" command from the shortcut menu.

The objects are removed from the group.

Note

If there are only two objects in the group, the menu command "Remove from group" is not available.

Deleting objects from a group

To remove an object from the group, and from the screen:

1. Select the group.
2. Select the command "Group > Edit group" from the shortcut menu.
   The group you want to edit is highlighted by a red frame.
3. Select the objects in the group that you want to delete.
4. Select "Delete" from the shortcut menu.

   Note

   If there are only two objects in the group, the menu command "Delete" is not available.

Alternative procedure

You can also edit groups in the "Layout" task card. Using drag-and-drop you can also easily edit nested groups in the "Layers" pane.

Editing an Object in a Group

Introduction

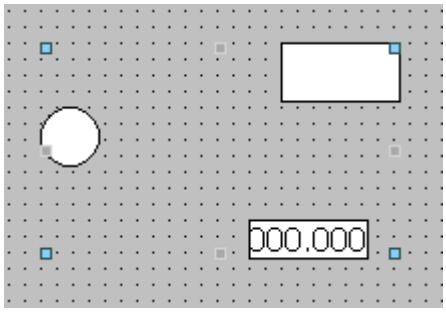
You can edit the objects of a group individually.

Requirement

You have opened a screen that contains a group.

Editing grouped objects

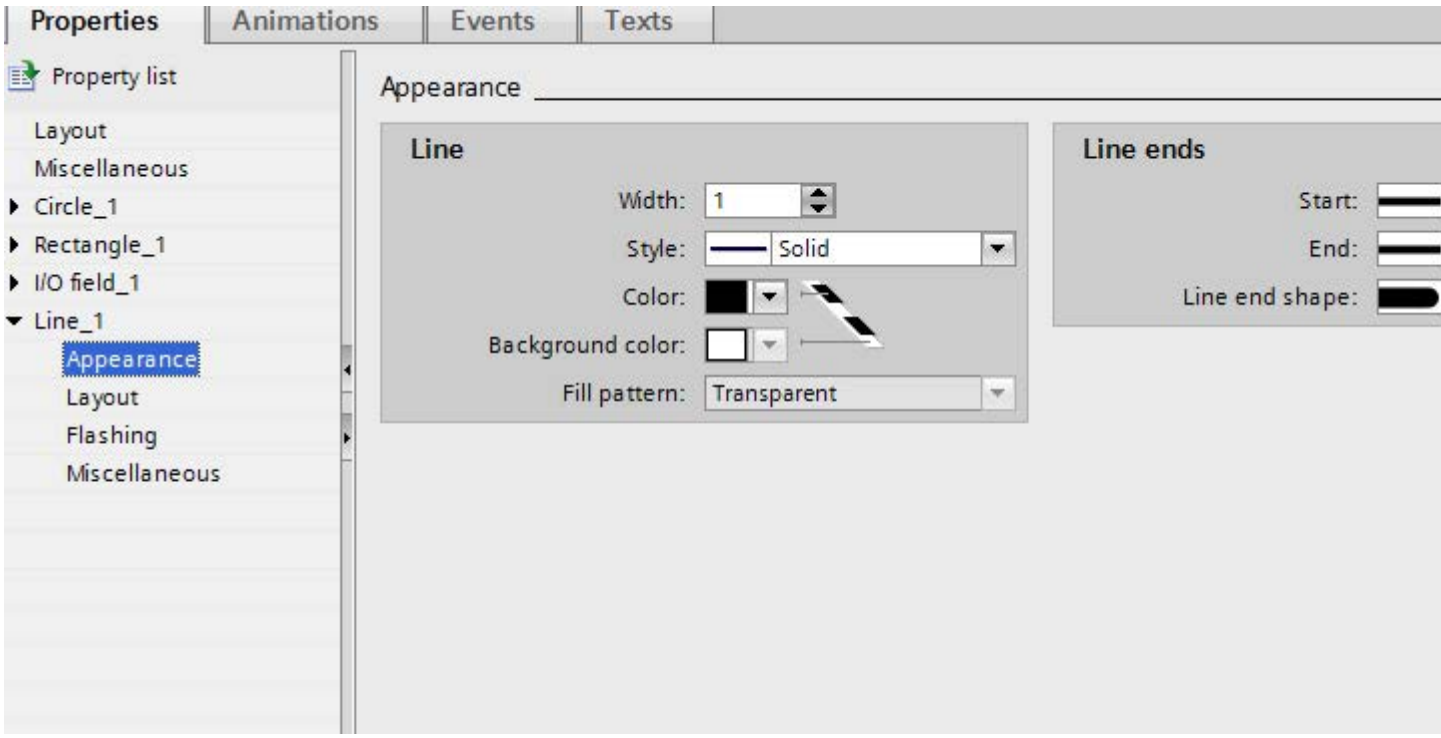
1. Select the group.



   The properties of the group are displayed in the Inspector window.
2. Change the position and size of the grouped objects in "Properties > Properties >Layout."
3. Change the name of the group in "Properties > Properties > Miscellaneous."

Modifying the properties of an object within a group

1. Select the group.
2. Select the object whose properties you want to change in the Inspector window.



The properties of the object are displayed.

3. Change the object properties.

Note

The dynamization of properties for all objects of the group which have these properties is not possible in a group. You can only dynamize the properties of the objects associated with a group for each object itself.

Result

Although you have edited the object, it is still an element of the group. These changes do not affect the other objects of the group.

10.1.2.27 Configuring the keyboard access

Overview of keyboard access

Introduction

For keyboard units without a mouse, the operator activates control objects using the <Tab> key. You can set up keyboard input to make the process easier to run, and to make sure that the operator enters all the necessary values. If you are using the keyboard, use the <Tab> key to activate the objects in a certain order, and to enter the necessary values.

For HMI devices without key, you can simulate the <Tab> key by configuring the "SimulateSystemKey" system function to a function key.

Operator authorizations and operator control enables

If you configure an object for operator input with the <Tab> key, the object must have both an operator authorization, and an operator control enable.

Editing the tab sequence

The tab sequence is determined automatically when the control objects are created. The numbers of the tab sequence are assigned in the sequence in which the libraries are created.

It makes sense to change the tab sequence in the following cases:

- The operator changes directly to a specific control object.
- The screen requires a specific sequence

Change to the tab sequence mode to change the tab sequence. In this mode, the tab sequence number is displayed at the top left of the control objects. The tab sequence numbers of hidden objects are also shown. The distribution of these numbers is edited using the mouse.

Note

Other functions are not available in the tab sequence mode.

See also

Example: Inserting and configuring a rectangle (Page 8022)

Overview of objects (Page 7978)

Defining the Operator Authorization and Operator Control Enable for an Object

Introduction

If you configure an object for operator input with the <Tab> key, the object must have both an operator authorization, and an operator control enable.

Requirement

You have opened a screen which contains at least one object.

Procedure

1. Select the object.
2. Select "Properties > Properties > Security" in the Inspector window.
3. Select the operator authorization under "Authorization."
4. Activate the authorization to operate.

Result

The operator can use the <Tab> key in Runtime to select the object.

See also

Example: Inserting and configuring a rectangle (Page 8022)

Setting the tab sequence order

Introduction

All operable objects can be reached in runtime with the <Tab> key. You use the "Tab sequence" command to define the order in which the operator can activate objects in runtime.

Note

You cannot reach objects with the "Output" or "Two states" mode in runtime with the <Tab> key.

You can operate the screen in runtime:

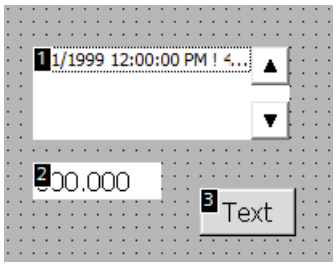
- Using the <Tab> key
- Using the mouse
- Using a configured hotkey

Requirement

- The active screen contains operable objects.
- No object is selected.
- The objects have been enabled for use in runtime, and have operator authorization.

Procedure

1. Select "Edit tab sequence" in the "Edit" menu.
   Tab sequence mode is activated. The tab sequence number is displayed for all objects that can be used. The tab sequence number is also displayed for hidden objects.
2. Use edit the tab sequence mode, click the accessible objects in the order in which you want them to be activated using <Tab> in runtime.
   The following figure shows how the tab sequence is defined in the screen. In runtime, the <Tab> key first activates the alarm view (number 1), then the I/O field (number 2), and then the button (number 3):



3. To exclude a screen object from the tab sequence, press the key combination <Shift+Ctrl> and click on the desired object.
   The tab sequence number is no longer displayed in the screen object. The screen object is now excluded from the tab sequence. The remaining tab sequence numbers are automatically decreased by 1.
4. To reenter an excluded screen object in the tab sequence, repeat step 3.
   The screen object entered as the first object in the tab sequence.

Result

The operator selects the objects in the specified order in Runtime with the <Tab> key.

See also

Example: Inserting and configuring a rectangle (Page 8022)

10.1.2.28 Examples

Example: Inserting and configuring a rectangle

Task

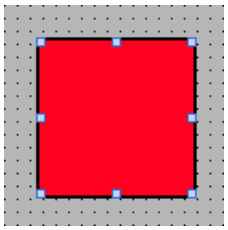
In this example, you insert a rectangle in a screen. You can configure the following properties:

- Name = "MyRectangle"
- Position = (20, 20)
- Size = (100,100)

- Color = red
- Black frame 2 pixels wide

Principle

The rectangle is a closed object which can be filled with a color or pattern. The height and width of a rectangle can be adjusted to allow its horizontal and vertical alignment.



Overview

Carry out the following steps in order to create a rectangle:

- Inserting a rectangle
- Configuring a rectangle

See also

Basics on groups (Page 8013)

Overview of objects (Page 7978)

Example: Inserting a rectangle

Task

In this example, you insert and rename a rectangle. Do not use the special characters ?, ", /, \, *, <, > for the name.

Requirement

- A screen is open.
- The Inspector window is open.
- The "Tools" task card is open.

Procedure

1. Click the "Basic objects" palette in the "Tools" task card.
2. Drag the "Rectangle" object into the screen.
3. In the Inspector window, select "Properties > Properties > Miscellaneous".
4. Type in the new name "MyRectangle".

Result

The rectangle is now inserted and named "MyRectangle". The rectangle has the default properties of the "rectangle" object.

Example: Configuring a rectangle

Task

In this example you configure a rectangle:

- Color = red
- Black frame 2 pixels wide
- Position = (20, 20)
- Size = (100,100)

Changing the color of the rectangle

To change the color of the rectangle:

1. Select the rectangle.
2. Define the background color in "Properties > Properties > Appearance > Background > Color" in the Inspector window.
3. Select "Solid" as the fill pattern.
4. Define the color for the border in "Properties > Properties > Appearance > Border > Color" in the Inspector window.
5. Enter the value "2" for "width".
6. Select "Solid" as the "Style".

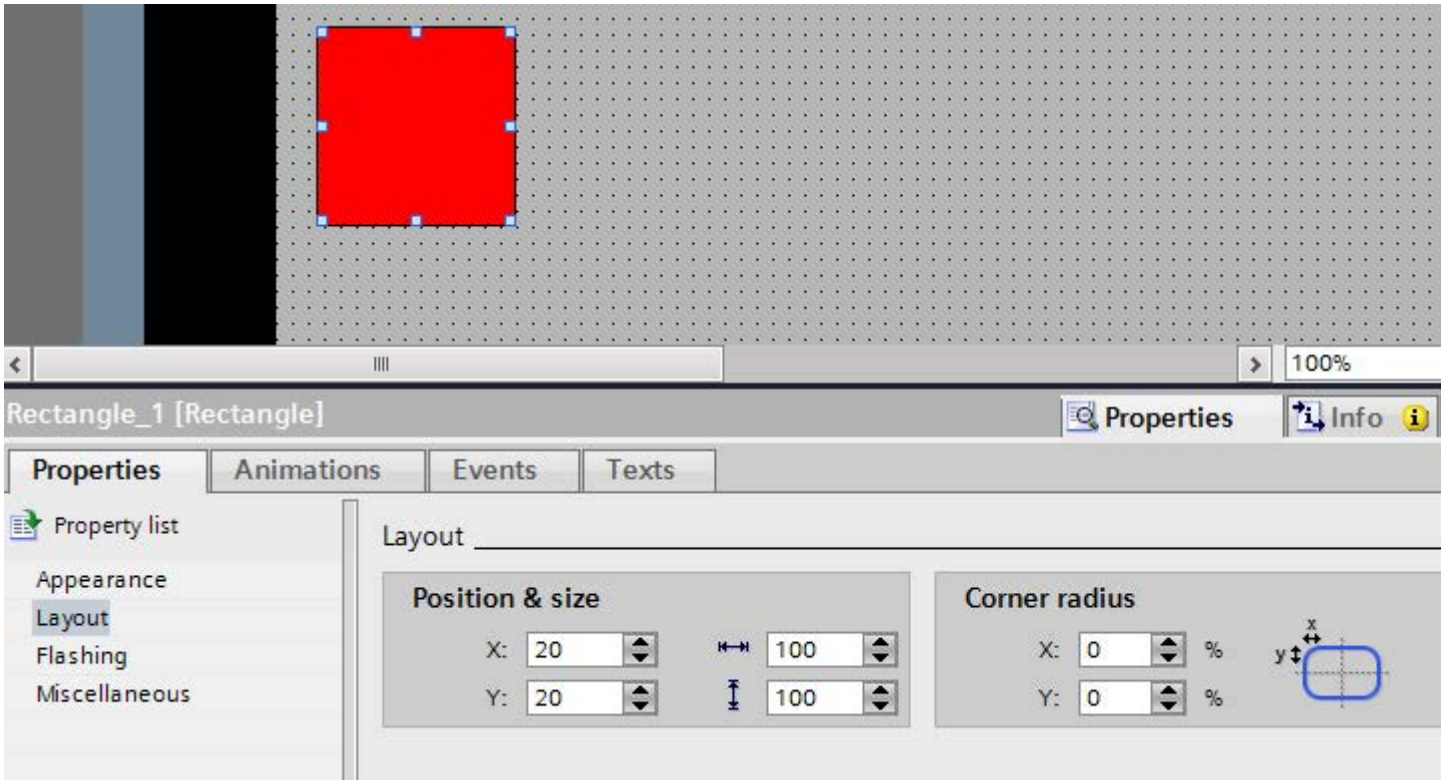
Interim result

The rectangle is red and has a black frame with a width of two pixels.

Repositioning and resizing the rectangle

To change the position and size of the rectangle:

1. Select the rectangle.
2. In the Inspector window, select "Properties > Properties > Layout".



3. Set "20" for the both the X and Y coordinates under "Position & Size".
4. Set "100" for the height and for the width.

Result

The rectangle is positioned at the coordinates (20, 20), and has a width and height of 100 pixels.

10.1.3 Working with text lists and graphics lists

10.1.3.1 Working with text lists

Basics on text lists

Introduction

Texts are assigned to the values of a tag in a text list. Assign the text list to a symbolic I/O field for example in the configuration. This supplies the text to be displayed to the object. The text lists are created in the ""Text List" editor. You configure the interface between the text list and a tag at the object that uses the text list.

The selection of objects that can have a text list assigned depends on the Runtime.

Application

The text list is used, for example, to display a drop-down list in a symbolic I/O field.

If the symbolic I/O field is a display field, the associated texts will differ according to the value of the configured tags. If the symbolic I/O field is an input field, the configured tag assumes the associated value when the operator selects the corresponding text in Runtime.

Note

Display of tag values without text

The display of tag values to which no text has been assigned depends on the Runtime:

- The display and operating object remains empty.
- Three asterisks *** are displayed.

Multilingual texts

You can configure multiple languages for the texts in a text list. The texts will then be displayed in the set language in Runtime. To this purpose you set the languages in the Project window under "Language support > Project languages."

Configuration steps

The following steps are necessary to display texts in a symbolic I/O field for example:

1. Creating the text list
2. Assignment of the texts to values or value ranges of a text list
3. Assignment of a text list in the display object, e.g. the symbolic I/O field

Creating a text list

Introduction

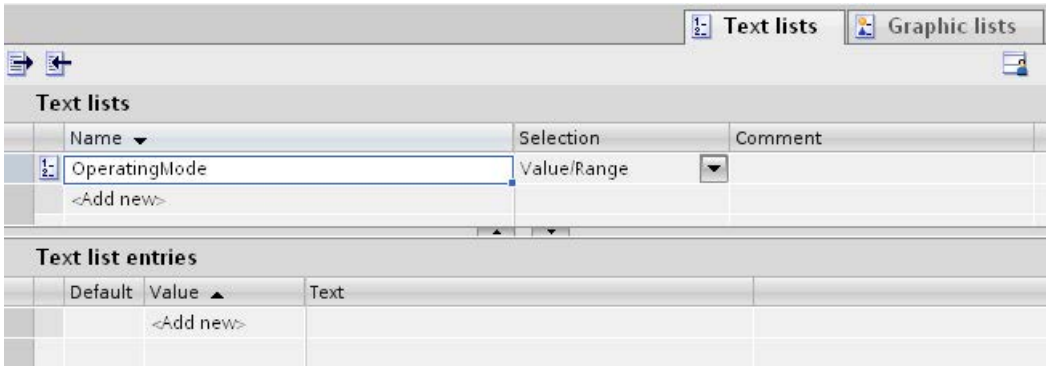
The text list allows you to assign specific texts to values and to output these in Runtime, for example in a symbolic I/O field. The type of symbolic I/O field can be specified, for example as a pure input field.

The following types of list are available:

- Value/Range
- Bit
- Bit Number

Procedure

1. Double-click "Text and graphics lists" in the project window.
2. Open the "Text lists" tab.



3. Click "Add" in the "Text lists" table.
   The Inspector window of the text list is open.
4. Assign a name to the text list that indicates its function.
5. Select the text list type under "Selection":
   - Value/Range: Text from the text list is displayed when the tag has a value that lies within the specified range.
   - Bit (0,1): A text from the text list is displayed when the tag has the value 0. A different text from the text list is displayed when the tag has the value 1.
   - Bit number (0-31): Text from the text list is displayed when the tag has the value of the assigned bit number.
6. Enter a comment for the text list.

Note

You must not use semicolons in the texts in a text list. The semicolon is a control character and is automatically deleted from a text.

Result

A text list is created.

Assigning texts and values to a range text list

Introduction

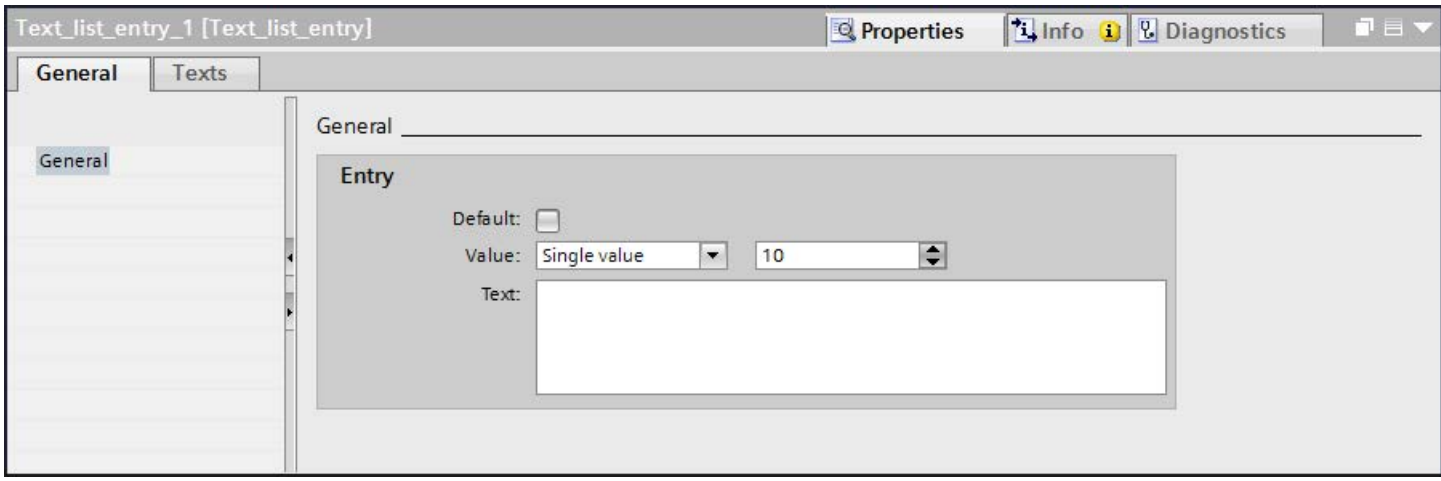
For each area text list you specify which texts are displayed at which value range.

Requirement

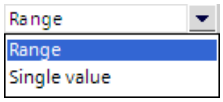
- The "Text and graphics list" editor is open.
- The "Text lists" tab is open.
- An area text list has been created and selected.

Procedure

1. Click "Add" in the "Text list entries" table.
   The Inspector window for this list entry opens.



2. Select the setting "Range" in "Properties > Properties > General > Value" in the Inspector window.



   – Enter the value "1" for "Min" for example.
   – Enter the value "20" for "Max" for example.
   – For "Text", enter the text that is displayed in Runtime if the tag is within the specified value range.

   Note

   Use a maximum of 255 characters and no semicolons for the text.

3. Click "Add" in the "Text list entries" table. A second list entry is created.
4. Select the setting "Range" in "Properties > Properties > General > Value" in the Inspector window.
   – Enter the value "21" for "Min" for example.
   – Enter the value "40" for "Max" for example.
   – For "Text", enter the text that is displayed in Runtime when the tag is within the specified value range.
5. If required, activate the "default entry".
   The entered text is always displayed when the tag has an undefined value. Only one default entry is possible per list.

Result

An area text list is created. Texts have been assigned to the possible value ranges.

Assigning texts and values to a bit text list

Introduction

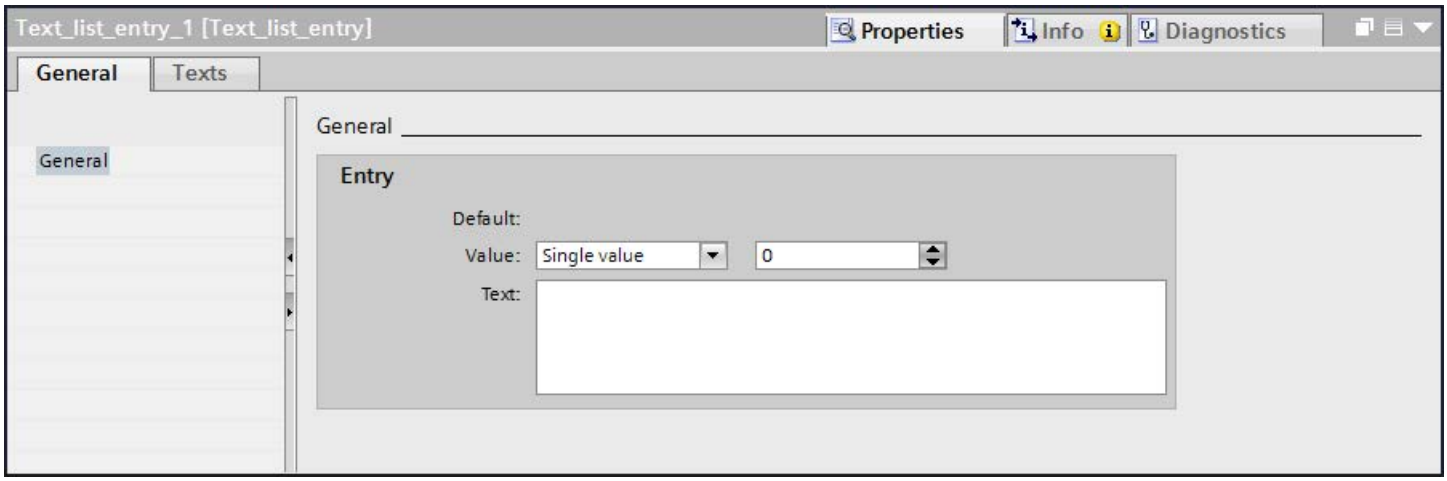
For each text list, you specify which text is displayed at which bit value.

Requirement

- The "Text and graphics list" editor is open.
- The "Text lists" tab is open.
- A bit text list has been created and selected.

Procedure

1. Click "Add" in the "Text list entries" table.
   The Inspector window for this list entry opens.



2. Select the setting "Single value" in "Properties > Properties > General > Value" in the Inspector window.
   - Enter "0" for "Value."
   - Enter the text which is displayed in Runtime under "Text" if the bit tag is set to "0".

   Note

   Use a maximum of 255 characters and no semicolons for the text.

3. Click "Add" in the "Text list entries" table. A second list entry is created.
4. Select the setting "Single value" in "Properties > Properties > General > Value" in the Inspector window.
   - Enter "1" under "Value."
   - Enter the text which is to be displayed in Runtime under "Text" if the bit tag is set to "1".

Result

A bit text list is created. Texts that appear in Runtime are assigned to the possible values "0" and "1".

Assigning texts and values to a bit number text list

Introduction

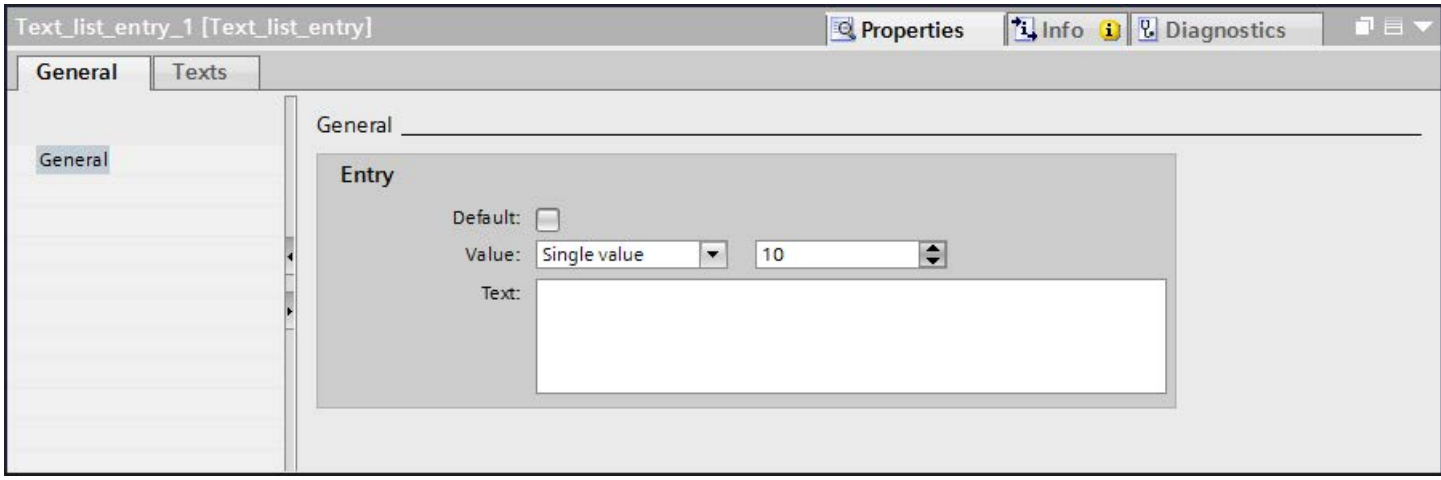
For each bit number text list you specify which texts are displayed at which bit number.

Requirement

- The "Text and graphics list" editor is open.
- The "Text lists" tab is open.
- A bit number text list has been created and selected.

Procedure

1. Click "Add" in the "Text list entries" table.
   The Inspector window for this list entry opens.



2. Select the setting "Single value" in "Properties > Properties > General > Value" in the Inspector window.
   - Enter "10", for example, for "Value".
   - Under "Text", enter the text that is displayed in Runtime when the tag has the value "10".

   Note

   Use a maximum of 255 characters and no semicolons for the text.

3. If required, activate the "default entry".
   The entered text is always displayed when the tag has an undefined value. Only one default entry is possible per list.
4. Create further list entries for additional bit numbers of the same text list.

Result

A bit number text list is created. Texts that appear in Runtime are assigned to the specified bit numbers.

Configuring objects with a text list

Introduction

The output value and value application for text lists are specified in the display and operating object that displays the texts of the text list in Runtime. The properties of these objects are configured as required.

Requirement

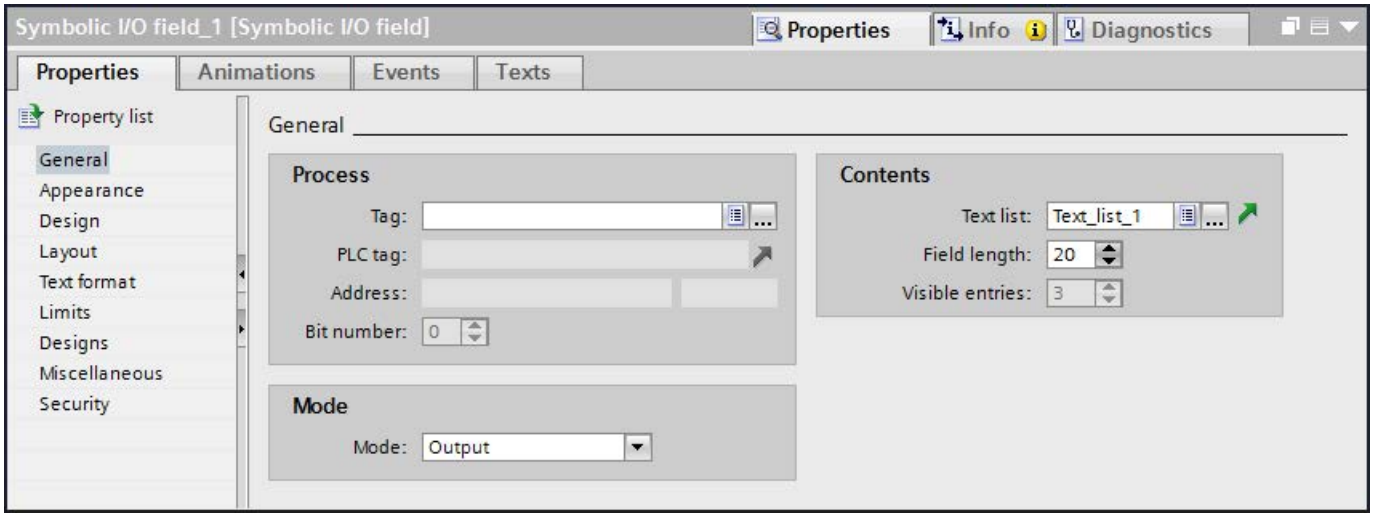
- A text list is created.
- You have created a tag.
- The "Screens" editor is open.
- A screen with a symbolic I/O field is open. The object is edited.

Procedure

1. Select the text list which you want to have displayed in Runtime in "Properties > Properties > General > Text list" in the Inspection window.
2. Select the setting "Output" as the "Mode".

   Note
   Runtime dependency
   Different field types are available for a symbolic I/O field depending on the Runtime.

3. Select the tag the value of which determines the display in the symbolic I/O field as "Tag".



Result

The defined texts of the text list are displayed in the symbolic I/O field in Runtime when the tag has the specified value.

10.1.3.2 Working with graphics lists

Basic principles of graphics lists

Introduction

The possible values of a tag are assigned to specific graphics in a graphics list. During configuration, you can create a graphics list for a button or a graphic I/O field. This supplies the graphics to be displayed to the object.

The graphics lists are created with the "Text and graphics list" editor. You configure the interface between the graphics list and a tag at the object that uses the graphics list. The availability of the graphics list is determined by the HMI device used.

Application

You can configure the graphics list for the following situations:

- Drop-down list with a graphic I/O field
- State-specific graphic for a button

The graphics in a graphics list can be configured as multilingual. The graphics will then be displayed in the set language in runtime.

Graphic sources

Graphics can be added to the graphics list from the following sources:

- By selecting from a graphic browser
- By selecting an existing file
  You can use the following file types:
  *.bmp, *.ico, *.emf, *.wmf, *.gif, *.tiff, *.png, *.jpeg and *.jpg.
- By creating a new file

Function

If the graphic I/O field is a display field, the associated graphics will differ according to the value of the configured tags. If the graphic I/O field is an input field, the configured tag assumes the associated value when the operator selects a graphic in runtime.

Configuration steps

The following tasks are required to display graphics, for example, in a graphic I/O field:

1. Creating the graphics list
2. Assignment of the graphics to values or value ranges of a graphics list
3. Assigning a graphics list in the display object, for example the graphic I/O field

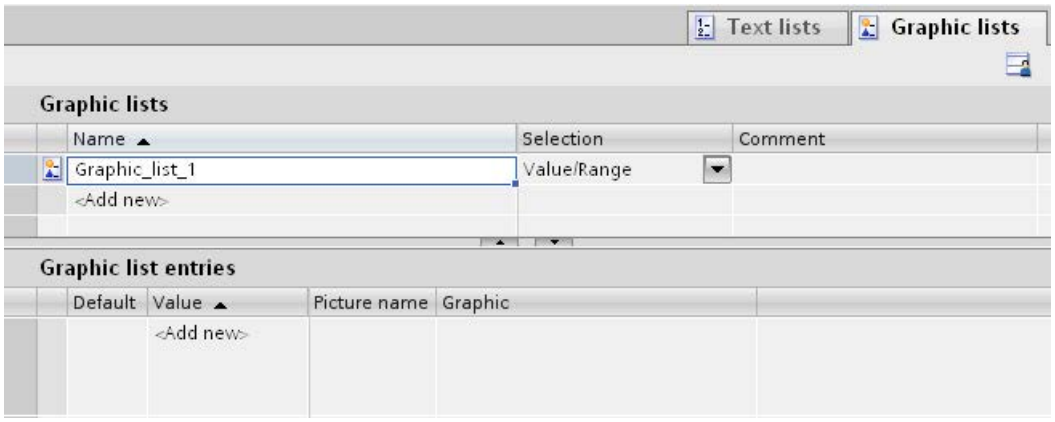
Creating a graphics list

Introduction

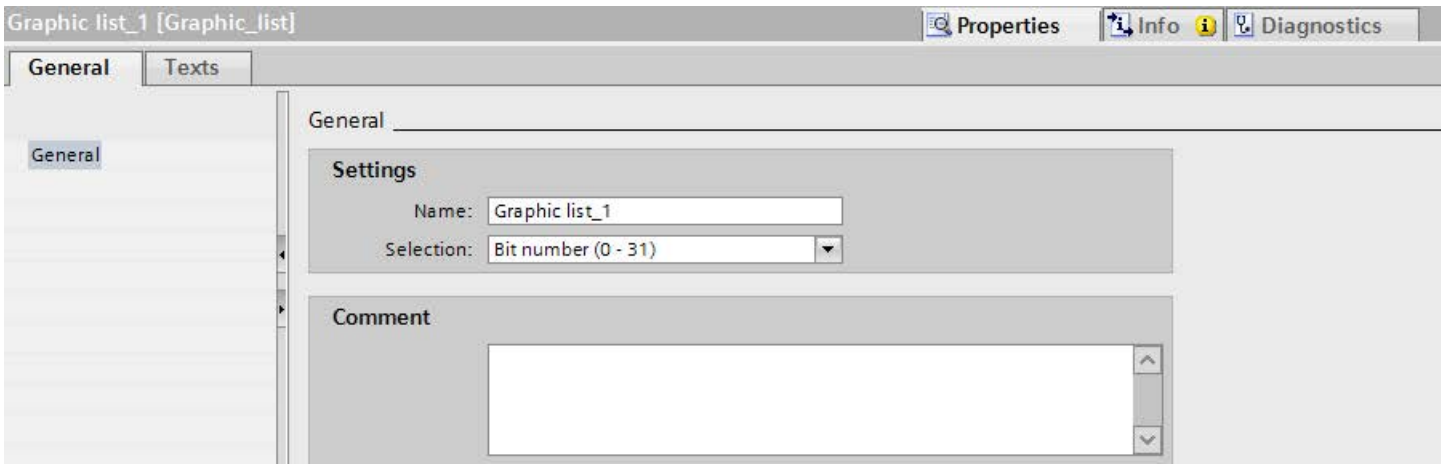
The graphics list allows you to assign specific graphics to variable values and to output these in a graphic IO field in Runtime. You can specify the type of graphic I/O field, for example as a pure output field.

Procedure

1. Double-click "Text and graphics lists" in the project window.
2. Open the "Graphics lists" tab.



3. Click "Add" in the "Graphics lists" table. The Inspector window of the graphics list will open up.



4. Assign a name to the graphics list that indicates its function.
5. Select the graphics list type "Bit number (0 - 31)" for example under "Select".
6. Enter a comment for the graphics list.

Result

A graphics list of the type "Range (0 - 31)" is created.

Assigning a graphic and values to a range graphics list

Introduction

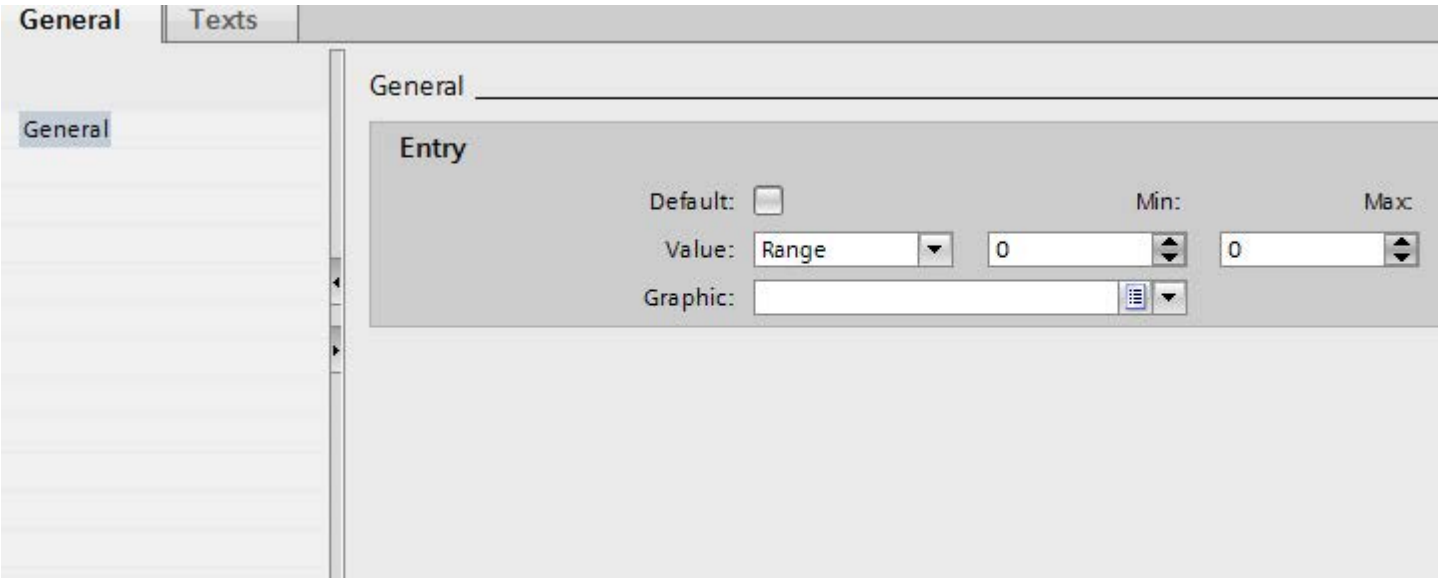
For each area graphics list you specify which graphics are displayed at which value range.

Requirement

- The "Text and graphics list" editor is open.
- The "Graphics list" tab is open.
- An area graphics list has been created and selected.

Procedure

1. Click "Add" in the "Graphics list entries" table.
   The Inspector window for this list entry opens.

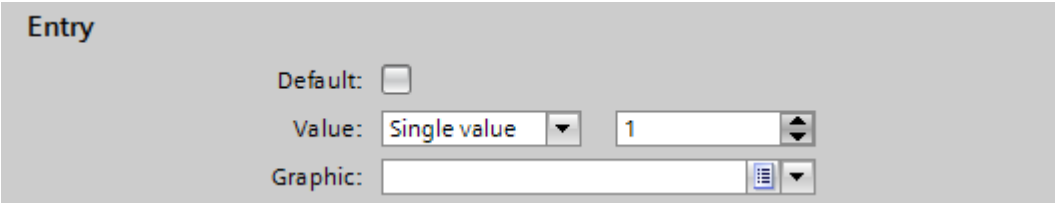


2. Select the settings "Range" in "Properties > Properties > General > Value" in the Inspector window:
   - Enter the value "1" for "Min" for example.
   - Enter the value "20" for "Max" for example.
   - Select a graphic that is displayed in Runtime when the tag is within the specified value range.

> **Note**
>
> As an alternative to the drop-down menu, you can insert graphics from libraries or from your file system:
>
> 1. Select a graphic in the library or in your file system.
> 2. Drag-and-drop the graphic into the "Graphics list entries > Graphic" table.

3. Click "Add" in the "Graphics list entries" table. A further list entry is created.

4. Select the settings "Single value" in "Properties > Properties > General > Value" in the Inspector window:
   - Enter the value "21" for example.
   - Select a graphic which is displayed in Runtime if the bit "21" is set in the tag.



5. If required, activate the "default entry".
   The graphic is always displayed when the tag has an undefined value. Only one default entry is possible per list.

Result

An area graphics list is created. Graphics that appear in Runtime are assigned to the possible values.

Assigning graphics and values to a bit graphics list

Introduction

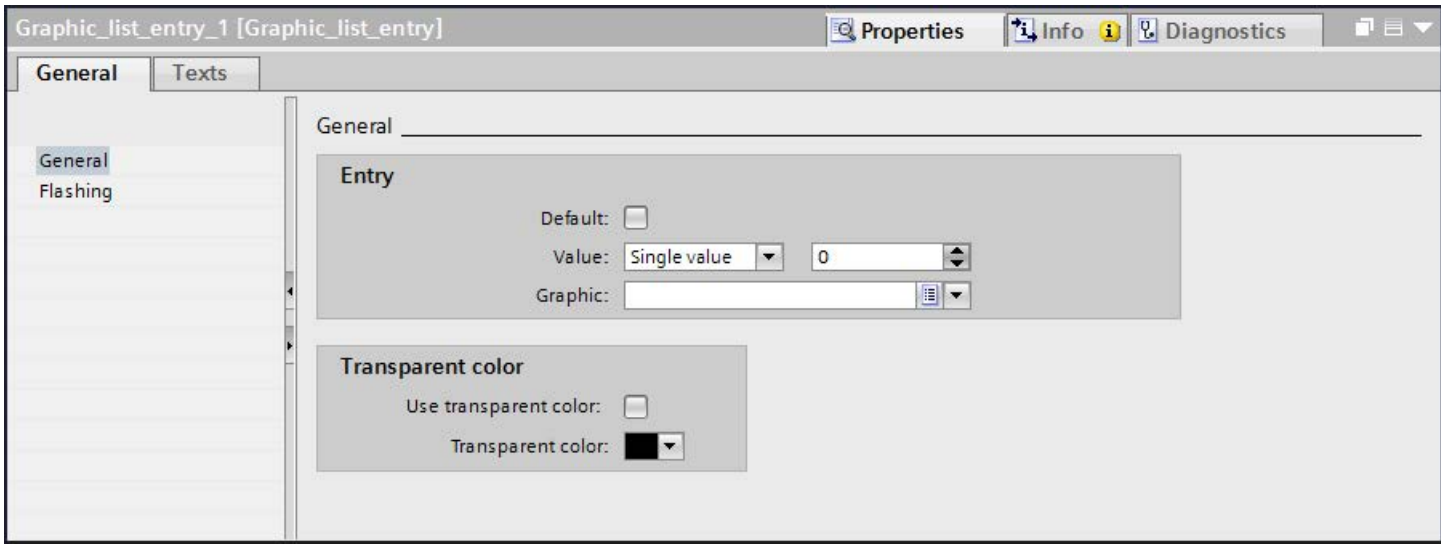
For each graphics list you specify which graphic is displayed at which bit value.

Requirement

- The "Text and graphics list" editor is open.
- The "Graphics list" tab is opened.
- A bit graphics list has been created and selected.

Procedure

1. Click "Add" in the "Graphics list entries" table.
   The Inspector window for this list entry opens.



2. Select the settings "Single value" in the inspector window "Properties > Properties > General > Value":
   - Enter "0" as the value.
   - Select a graphic which is displayed in Runtime if the bit "0" is set in the tag.

   Note

   As an alternative to the drop-down menu, you can insert graphics from libraries or from your file system:
   1. Select a graphic in the library or in your file system.
   2. Drag-and-drop the graphic into the "Graphics list entries > Graphic" table.

3. Click "Add" in the "Graphics list entries" table. A new list entry is created.
4. Select "Properties > Properties > General > Value > Single value": in the Inspector window.
   - Enter "1" as the value.
   - Select a graphic which is displayed in Runtime if the bit "1" is set in the tag.

Result

A bit graphics list is created. Graphics that appear in Runtime are assigned to the values "0" and "1".

Assigning graphics and values to a bit number graphics list

Introduction

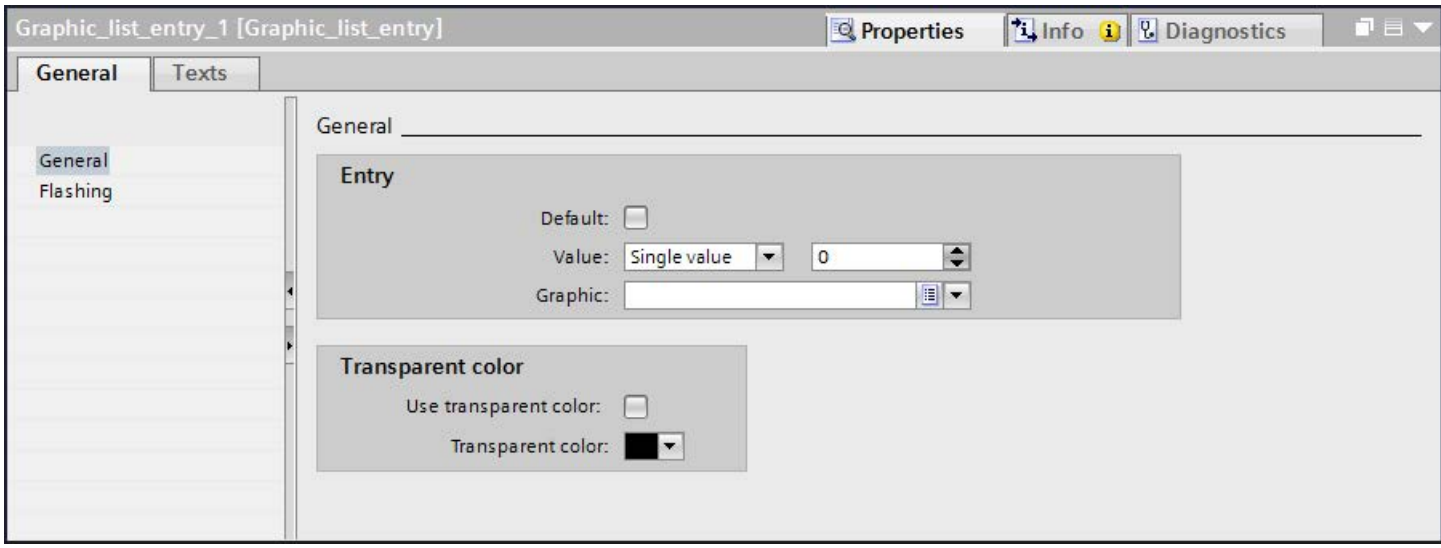
For each bit number graphics list you specify which graphics are displayed at which bit number.

Requirement

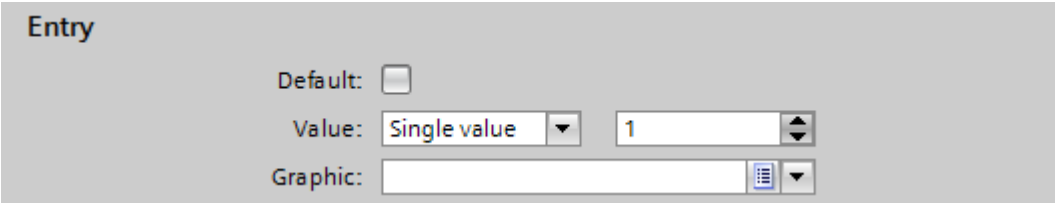
- The "Text and graphics list" editor is open.
- The "Graphics list" tab is open.
- A bit number graphics list has been created and selected.

Procedure

1. Click "Add" in the "Graphics list entries" table.
   The Inspector window for this list entry opens.



2. Select the settings "Single value" in "Properties > Properties > General > Value" in the Inspector window:
   - Enter the value "0" for example.
   - Select a graphic which is displayed in Runtime if the bit "0" is set in the tag.



> **Note**
>
> As an alternative to the drop-down menu, you can insert graphics from libraries or from your file system:
>
> 1. Select a graphic in the library or in your file system.
> 2. Drag-and-drop the graphic into the "Graphics list entries > Graphic" table.

3. If required, activate the "default entry".
   The graphic is always displayed when the tag has an undefined value. Only one default entry is possible per list.
4. Create further list entries for additional bit numbers of the same graphics list.

Result

A bit number graphics list is created. Graphics that appear in Runtime are assigned to the specified bit numbers.

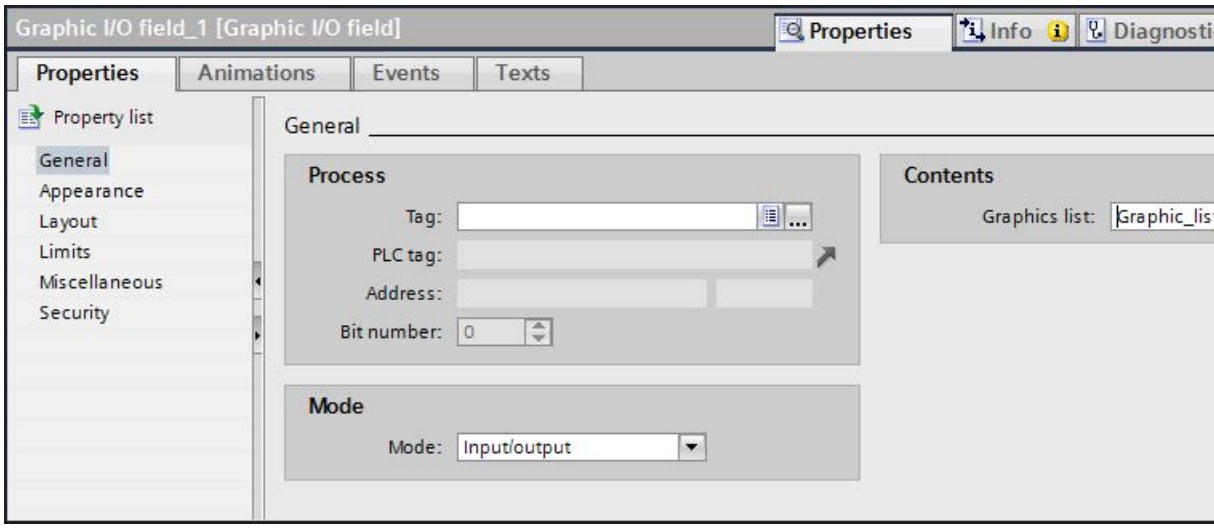
Configuring objects with a graphics list

Introduction

The output value and value application for graphics list are specified in the display and operating object that displays the graphics of the graphics list in Runtime. The properties of these objects are configured as required.

Requirement

-
-
-
-



Procedure

1.
2.

as the "Mode".

Note

Runtime dependency

Different field types are available for a graphic I/O field depending on the Runtime.

3. As "Tag", select the tag whose values are defined by the display in the graphic I/O field.

Result

The defined graphics of the graphics list are displayed in the graphic I/O field in Runtime when the tag has the specified value.

10.1.4 Dynamizing screens

10.1.4.1 Basics on dynamization

Dynamizing objects

In WinCC you dynamize objects to map your system and show processes on HMI devices.

You implement dynamizations by

- Animations
- Tags
- System functions

One example is the mapping of a tank, the liquid level of which rises or falls in relation to a process value.

The options for dynamization depend on the object involved. When you copy an object, its dynamization functions are included.

See also

Dynamization in the inspector window (Page 8043)

Configuring a new animation (Page 8045)

Basic on events (Page 8054)

10.1.4.2 Dynamization in the inspector window

Introduction

Basically, you can dynamize all the screen objects which you have configured in a screen. Which dynamization possibilities and which events are available depends on the device and the selected object.

Animations

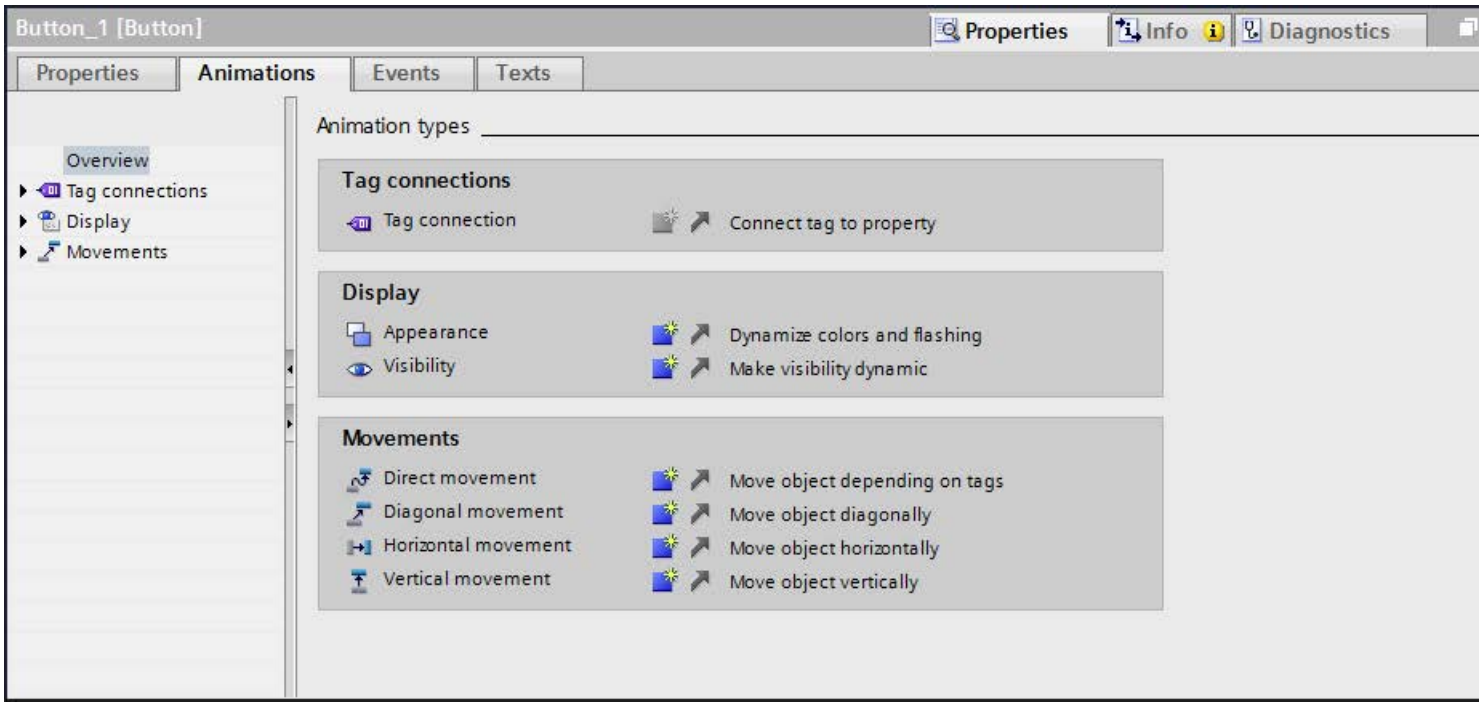
WinCC helps you to implement dynamization using predefined animations. If you want to animate an object, first configure the desired animation in the object's inspector window. Then adapt the animation to the requirements of your project.

The selection of the supported animations depends on the HMI device and the selected object. You choose between the following types of animation:

- Layout: Appearance, visibility
- Movements: direct, diagonal, horizontal and vertical movement
- Variable binding

You can configure the "Variable binding" type of animation several times for one object.

You configure animations in the "Properties > Animations" inspector window.

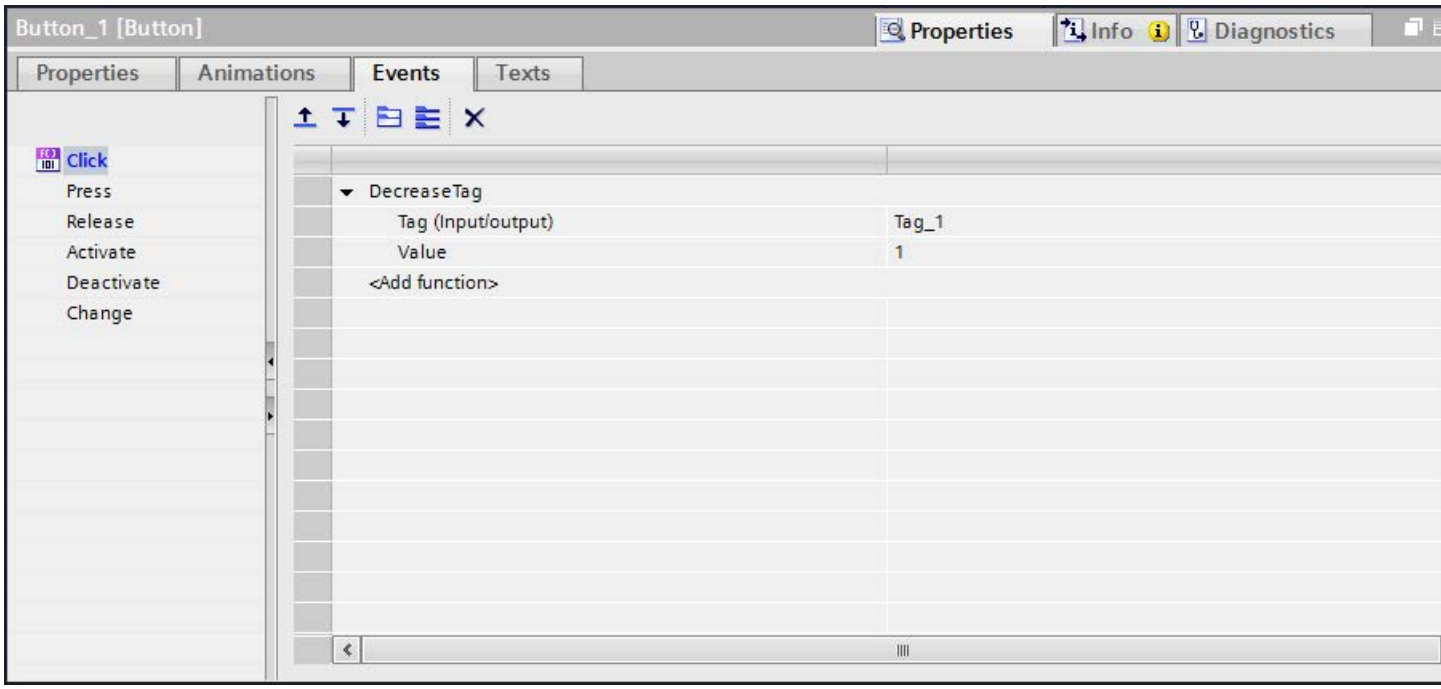


Events

Operable objects also react to events, such as a mouse click.

You configure a function list with system functions on an event. The system functions are processed as a reaction to the triggered event.

You configure events in the "Properties > Events" inspector window.



You will find further information in "Working with function lists".

See also

Basics on dynamization (Page 8042)

10.1.4.3 Dynamization with animations

Configuring a new animation

Introduction

Use predefined animations to dynamize screen objects.

Requirement

- You have opened a screen which contains at least one dynamic object.
- The Inspector window is open.
- The toolbox window is displayed.

Procedure in the inspector window

1. In the Inspector window, select "Properties > Animations".
2. Select the animation you want to use.
3. Click the button.

Procedure in the "Animations" task card.

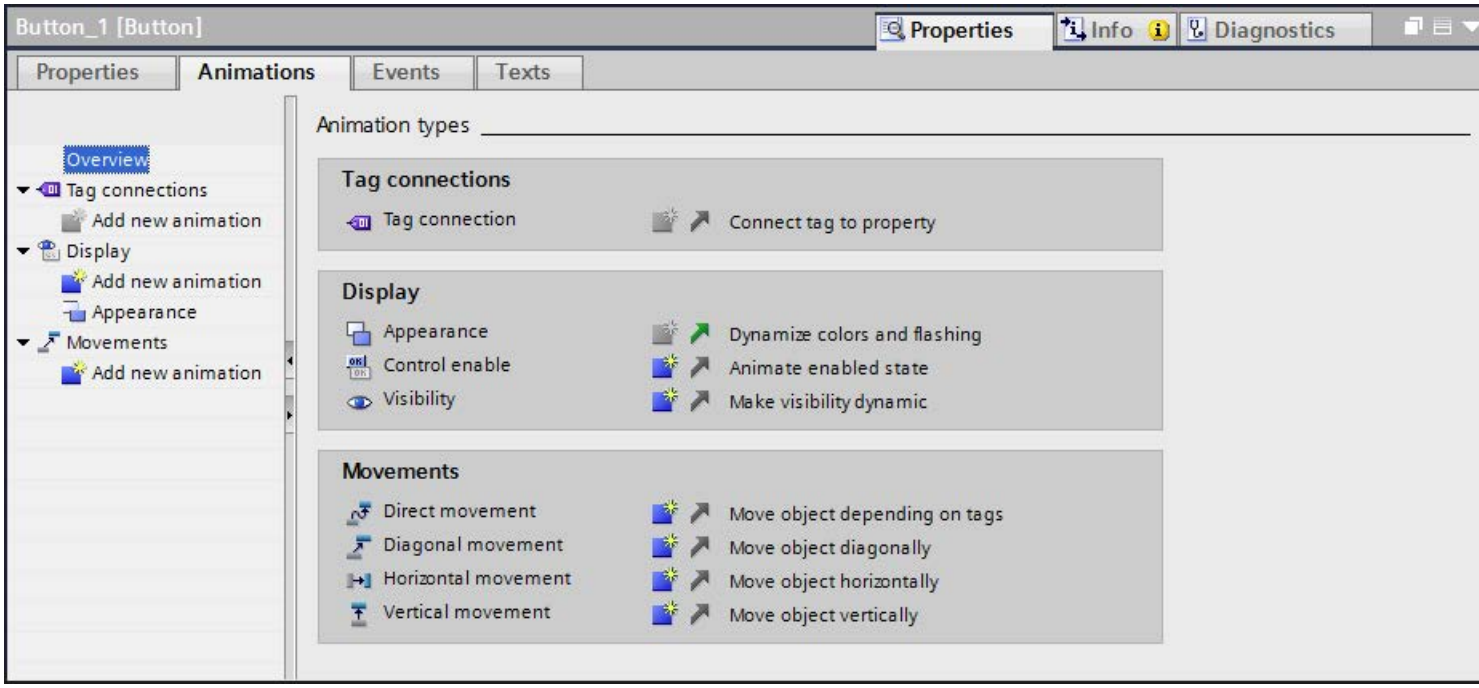
1. Open the object group containing the required animation in the "Animations" task card.
2. Drag the animation onto the object that you want to make dynamic.

Alternatively you select the object in the screen and double click the desired animation in the "Animation" task card.

Result

The animation appears in the Inspector window of the object. You configure the animation in the following steps.

In the animations overview the green arrow indicates which animation is already configured. The configured animation opens in the inspector window when you click the green arrow.



See also

Dynamizing the visibility of an object (Page 8051)

Basics on dynamization (Page 8042)

Dynamize appearance of an object

Introduction

The appearance of a screen object is controlled by changing the value of a tag in runtime. When the tag adopts a certain value, the screen object changes its color or flashing characteristics according to the configuration.

Type

Areas or single values of the tag are observed in Runtime depending on the selection. The appearance of the object changes according to the configuration.

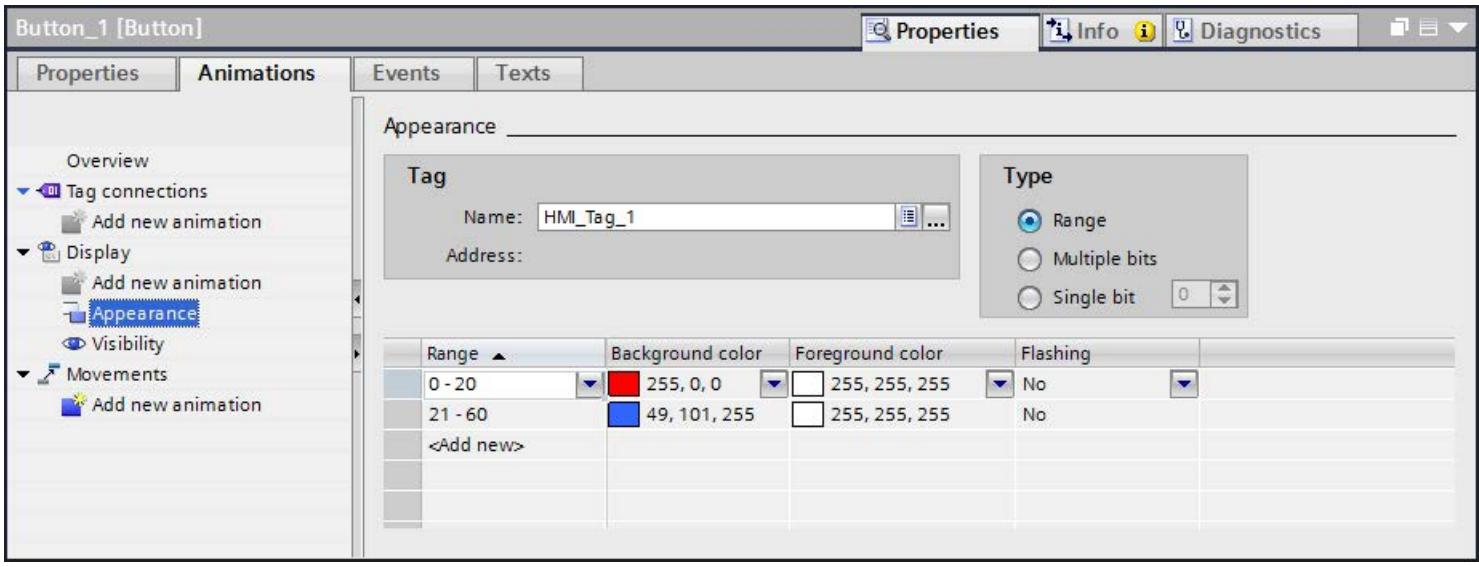
Requirement

- A screen is open.
- A dynamic object is contained and selected in the screen.
- The Inspector window is open.
- The toolbox window is displayed.

Procedure

1. In the Inspector window, select "Properties > Animations".
   The animations available for the selected object are displayed.
2. Select the animation "Appearance" and click the button.
   The parameters of the animation are displayed.
3. Select a tag in "Tag > Name".
4. Select "Type > Range" for example.
5. Click "Add" in the table.
6. Enter the tag interval "0 - 20" in the "Range" column for example.
7. For "Foreground color" and "Background color", select the color the object is to acquire in Runtime when the tag reaches the interval.

8. Select a flashing mode for the object from the "Flashing" list.
9. Repeat steps 5 to 8 to create another tag interval, e.g. "21 - 60".



Result

In Runtime, the flashing response, and color of the object change dynamically according to the process value returned in the tag.

Configuring movement

Introduction

You can configure dynamic objects in such a way that they move on a certain track. The movement is controlled via tags. The object moves every time the tag is updated.

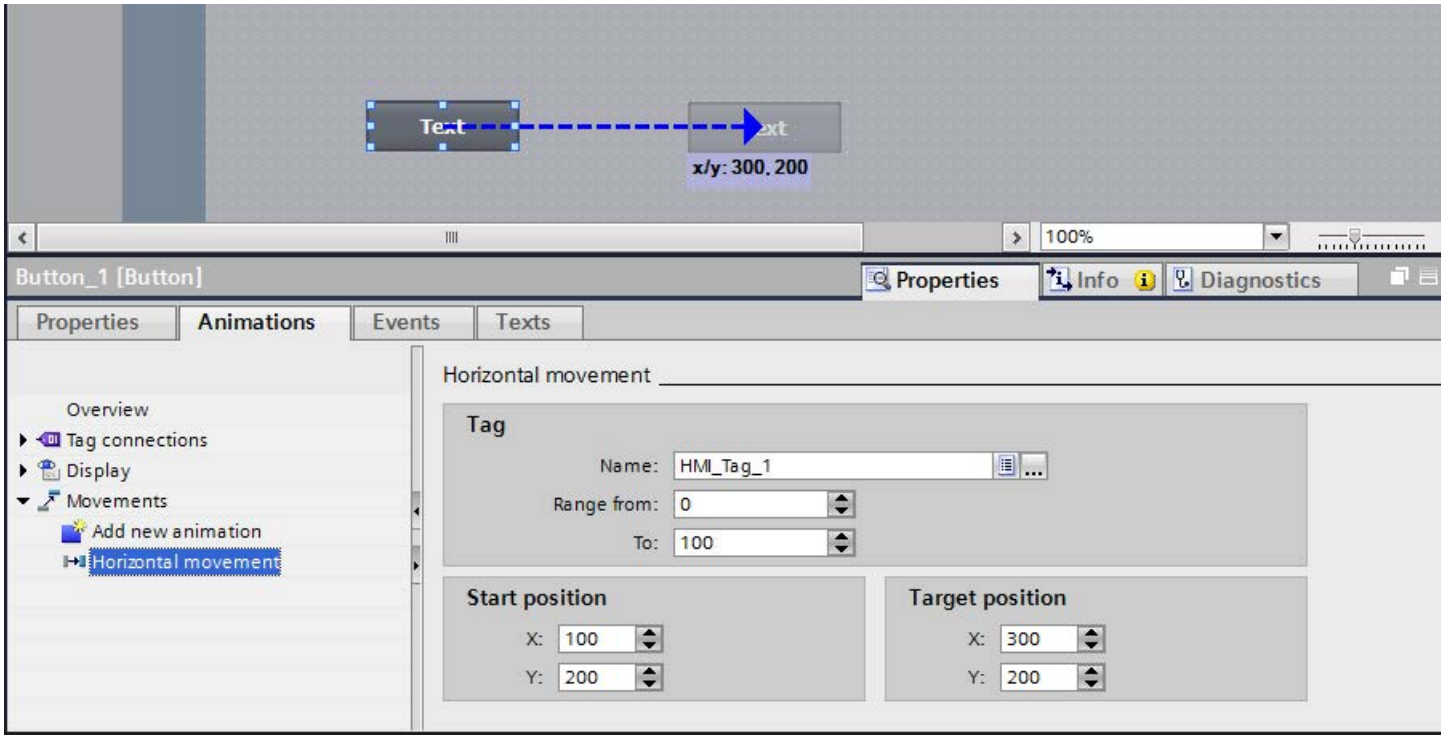
You can only program one type of movement for each object.

Requirement

- You have created a tag.
- You have opened a screen which contains at least one dynamic object.
- The Inspector window is open.
- The toolbox window is displayed.

Procedure

1. Select the screen object you want to control dynamically.
   The object properties are displayed in the Inspector window.
2. In the Inspector window, select "Properties > Animations".
   The animations available for the selected object are displayed.
3. Select "Horizontal movement" and click the button.
   The parameters of the animation are displayed.
   A transparent copy of the object is shown in the work area, which is connected to the source object by means of an arrow.
4. Select a tag for control of movement.
5. Move the object copy to the relevant destination. The system automatically enters the pixel values of the final position in the Inspector window.
6. Customize the range of values for the tag as required.



Result

In Runtime, the object moves in response to every change in value of the tag that is used to control the movement. The direction of movement corresponds to the configured type of movement "horizontal".

Note

You configure vertical and diagonal movements similar to horizontal movements

Configuring direct movement

Introduction

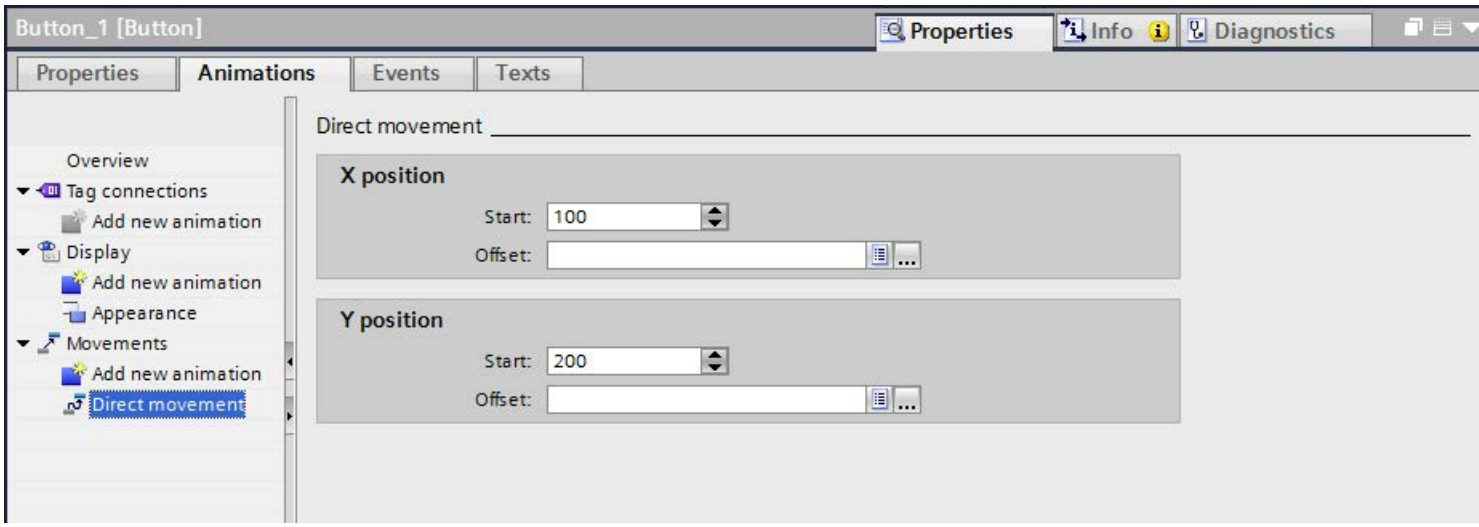
The object is moved respectively in x direction and y direction with "Direct movement". Two tags define the number of pixels by which the object moves from its original static start position.

Requirement

- Two tags are set up.
- You have opened a screen which contains at least one dynamic object.
- The Inspector window is open.
- The toolbox window is displayed.

Configuring "Direct movement"

1. Select the screen object you want to control dynamically.
   The object properties are displayed in the Inspector window.
2. In the Inspector window, select "Properties > Animations".
3. Select "Direct movement" and click the button.
   The parameters of the animation are displayed.
4. Select a tag for the X position with which the movement in x direction is controlled.
5. Select a tag for the Y position with which the movement in y direction is controlled.



Result

In Runtime, the object moves in response to every change in value of the tag that is used to control the movement.

Dynamizing the visibility of an object

Introduction

Dynamization of the "Visibility" property allows you to output an alarm to your screen, which is triggered when the tag value exceeds a critical range, for example. The alarm is cleared when the tag value returns to the non-critical range.

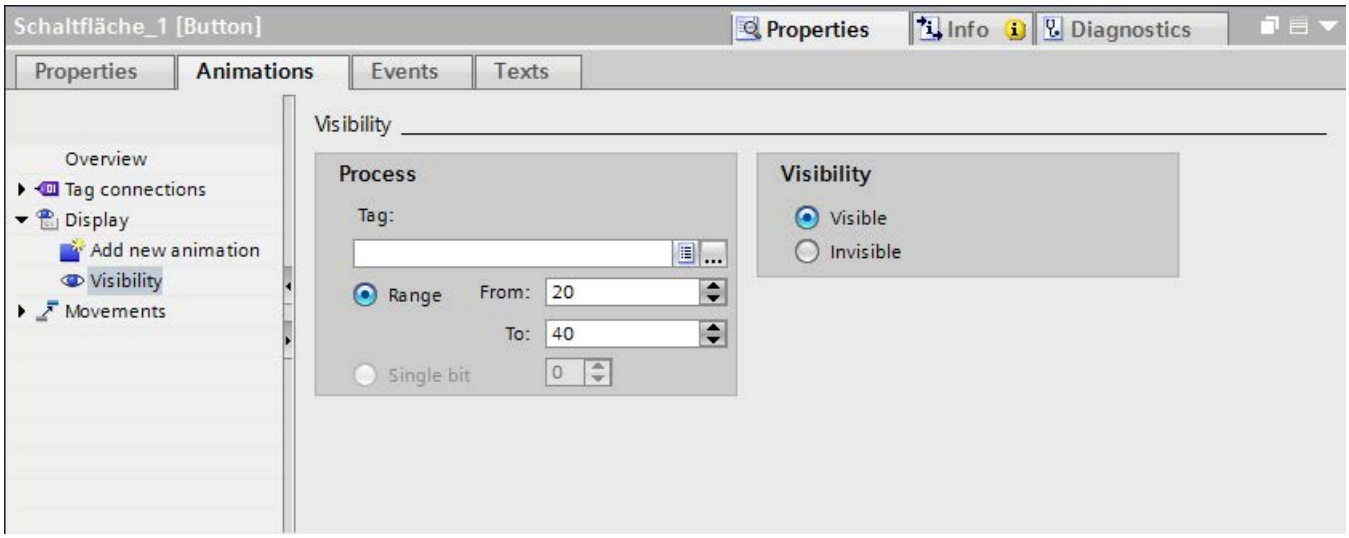
The "Simple recipe view" and "Simple alarm view" objects are always visible.

Requirement

- You have created a tag.
- You have opened a screen containing an object that you want to show or hide in Runtime.
- The Inspector window is open.

Procedure

1. Select the screen object you want to control dynamically.
   The object properties are displayed in the Inspector window.
2. In the Inspector window, select "Properties > Animations".
   The animations available for the selected object are displayed.
3. Select "Visibility" and click the button.
   The parameters of the animation are displayed.
4. Select a tag.
5. Activate "Range".
6. Select, for example, "from 20" and "to 40"
7. Activate "Visible".



Result

The screen object is shown or hidden in Runtime depending on the tag value.

- If the tag value matches the configured range between 20 and 40, the screen object is shown.
- If the tag value is outside the configured range, the screen object is hidden.

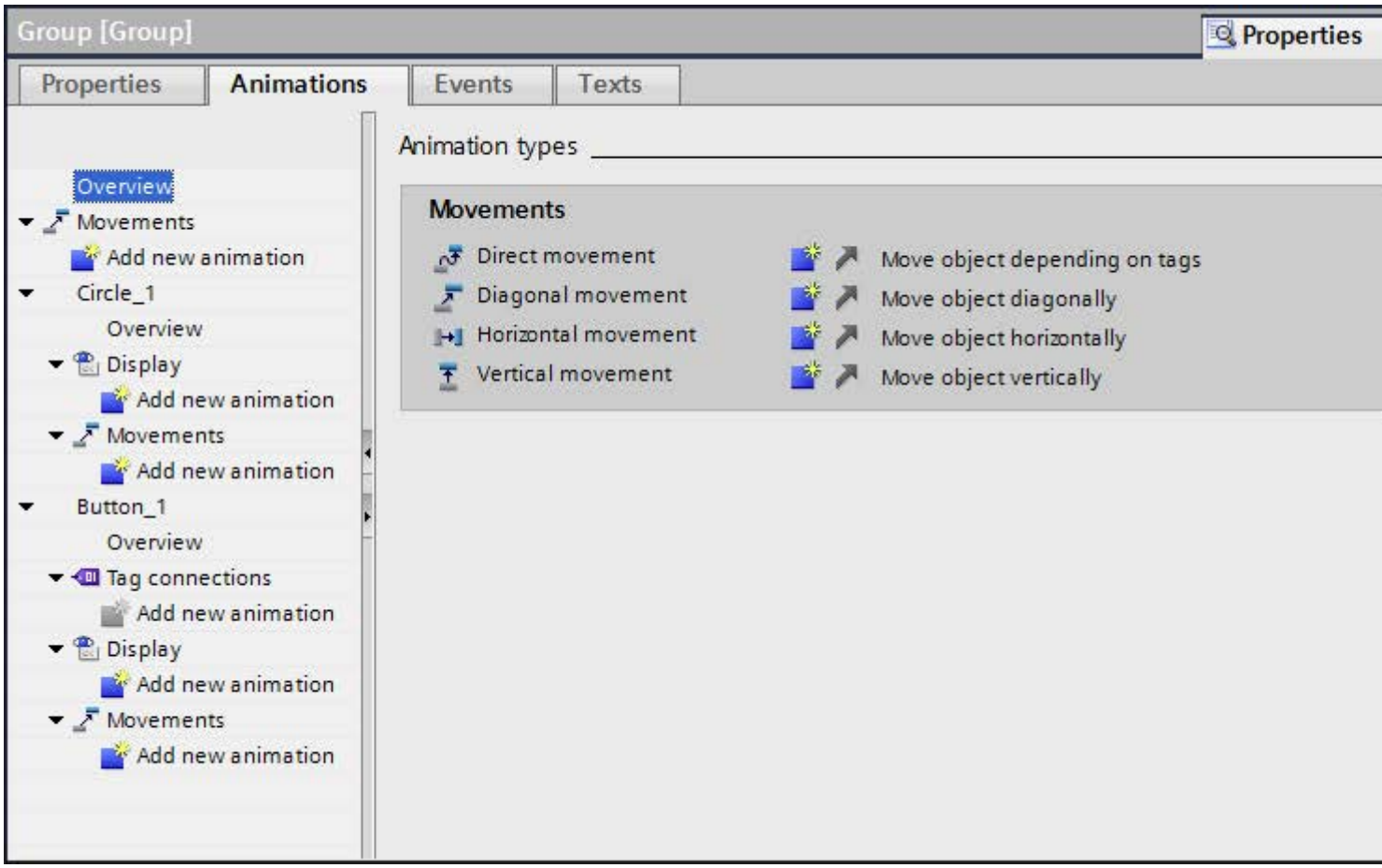
See also

Configuring a new animation (Page 8045)

Animations of object groups

Applying animations to object groups

The inspector window shows all objects of a group and their possible animations. The animation types which are supported by all objects in the group are also listed separately.



If you configure an animation for an object group, this animation will apply to all individual objects that support this animation.

Application example

The "horizontal movement" animation is configured for the object of an object group. The "direct movement" animation is configured for the whole object group. Only the animation of the object group i.e. "direct movement" is executed in Runtime. This also applies for object groups within object groups. Only the animation of the group on the top layer is listed.

Animations for object groups and for multiple selection

Changing animations for multiple objects

For a multiple selection, the animations that are configured for the reference object appear in the Inspector window. You can change the animations as usual. The changes will apply to all the objects in the multiple selection that support the configured animation. This means that even objects for which you have not yet configured an animation will have the reference object's animation.

Application example

Select a button, and a circle at the same time. The button is the reference object. The "Appearance" animation is already configured for the button, so it appears in the Inspector window of the multiple selection. When you activate "Properties > Animations >Appearance > Flashing" in the inspector window, the settings of the "Appearance" animation apply for the button and for the circle.

Configuring new animations for multiple objects

If you configure a new animation for the objects of a multiple selection, this animation will apply to all selected objects that support the configured animation. If the new animation replaces an existing animation, a security prompt appears.

Application example

You select a circle, and a rectangle. The "Diagonal movement" animation is already configured for the circle. You configure the "Horizontal movement" animation for the multiple selection. This animation applies to the rectangle since no animation of the Movement type is yet configured for it. For the circle, you are asked to confirm that you want to replace the existing "Diagonal movement" animation with the new "Horizontal movement" animation.

10.1.4.4 Dynamize with system functions

Basic on events

Introduction

Screen objects react to events. You configure a function list with system functions on the events of an object.

Events

Which events and system functions are available depends on the object used.

If the operator activates a screen object for example, the configured system function is executed.

You can find additional information under "Basics on the function list (Page 8276)".

See also

Basics on dynamization (Page 8042)

Example: Configuring a button for language switching (Page 8055)

Configure system function on the "Click" event (Page 8054)

Configure system function on the "Click" event

Introduction

You configure a function list on an object event. The linked system function is executed when the event occurs in runtime.

Requirements

A screen is open.

A button has been created in the screen.

The inspector window is open.

Procedure

1. Select the button.
2. Click "Properties> Events" in the Inspector window.
3. Select the "Click" event.
4. Click "Add function" in the table.
5. Select the "ShowAlarmWindow" system function.

Result

The alarm window opens in the screen if the operator clicks the button in runtime.

See also

Basic on events (Page 8054)

Example: Configuring a button for language switching

Introduction

In this example, you configure a button that can be used to toggle between multiple runtime languages during runtime.

Requirements

- You have completed the "Configuring a button in multiple languages" example.
- The "Screen_1" screen is open.
- The button on the screen has been selected.

Procedure

1. In the Inspector window, select "Properties > Events > Press".
2. Click on "Add function" in the table.
3. Select the "SetLanguage" system function and the "Switch" setting.

Result

You have assigned the button the function "SetLanguage". Pressing the button during runtime will switch the runtime language. The runtime languages are switched in the order specified by the number sequence in the "Languages and fonts" editor.

See also

Basic on events (Page 8054)

Button (Page 8099)

10.1.5 Working with function keys

10.1.5.1 Working with function keys

Introduction

The function key is a physical key on your HMI device and its functions can be configured. A function list can be configured for the events "Key pressed" and "Release key".

A function key can be assigned global or local functions.

Note

Availability for specific HMI devices

Function keys are not available on all HMI devices.

Global function keys

Global function keys always trigger the same action, regardless of the displayed screen.

Global function keys are configured in the "Global Screen" editor. The global assignment applies for all the screens of the set HMI device.

Global function keys reduce programming considerably, because there is no need to assign these global keys to each individual screen.

Local function keys in screens

Local function keys in screens can trigger a different action in each screen. This assignment applies only to the screen in which you have defined the function key.

Within a screen, a function key has only one function assignment, either a global or local one. The project planner specifies which assignment has priority.

Note

If a screen with local function keys is overlapped by an alarm view or an alarm window, then the function keys are still active in runtime. This may happen in particular on HMI devices with a small display.

Local function keys in templates

Local functions keys that are assigned in templates are valid for all the screens based on this template. They can trigger a different action in every screen. Function keys for templates are assigned in the template of the "Screens" editor. Within a template, a function key has only one function assignment, either a global or local one. The project planner specifies which assignment has priority.

Hotkey assignment

You can assign hotkeys, such as buttons, to control objects. The available hotkeys depend on the HMI device.

Note

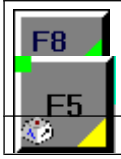
The function key has a local or global action assigned to it. If the function key also has a hotkey assigned to it, then the hotkey function will be executed in Runtime.

Graphics

When a function key is placed directly next to the display, you can assign a graphic to it to make the function of the function key more clear.

Display of assignment

Table 10-1 The following table shows which symbols display the assignment of the function keys:

Function key	Description
	Not assigned
	Global assignment
	Assigned locally in the template
	Local assignment
	Local assignment (local assignment of the template overwrites global assignment)
	Local assignment (local assignment overwrites global assignment)
	Local assignment (local assignment overwrites local assignment of the template)
	Local assignment (local assignment overwrites local assignment of the template, which already overwrites global assignment)
	Assigning buttons with screen navigation

Note

Basic Panels

The "Screen Navigation" editor is not available for Basic Panels.

10.1.5.2 Assigning function keys globally

Introduction

You define the global assignment of a function key in the "Global Screen" editor. The global assignment applies to all screens of the set HMI device.

Note

Availability for specific HMI devices

Function keys are not available on all HMI devices.

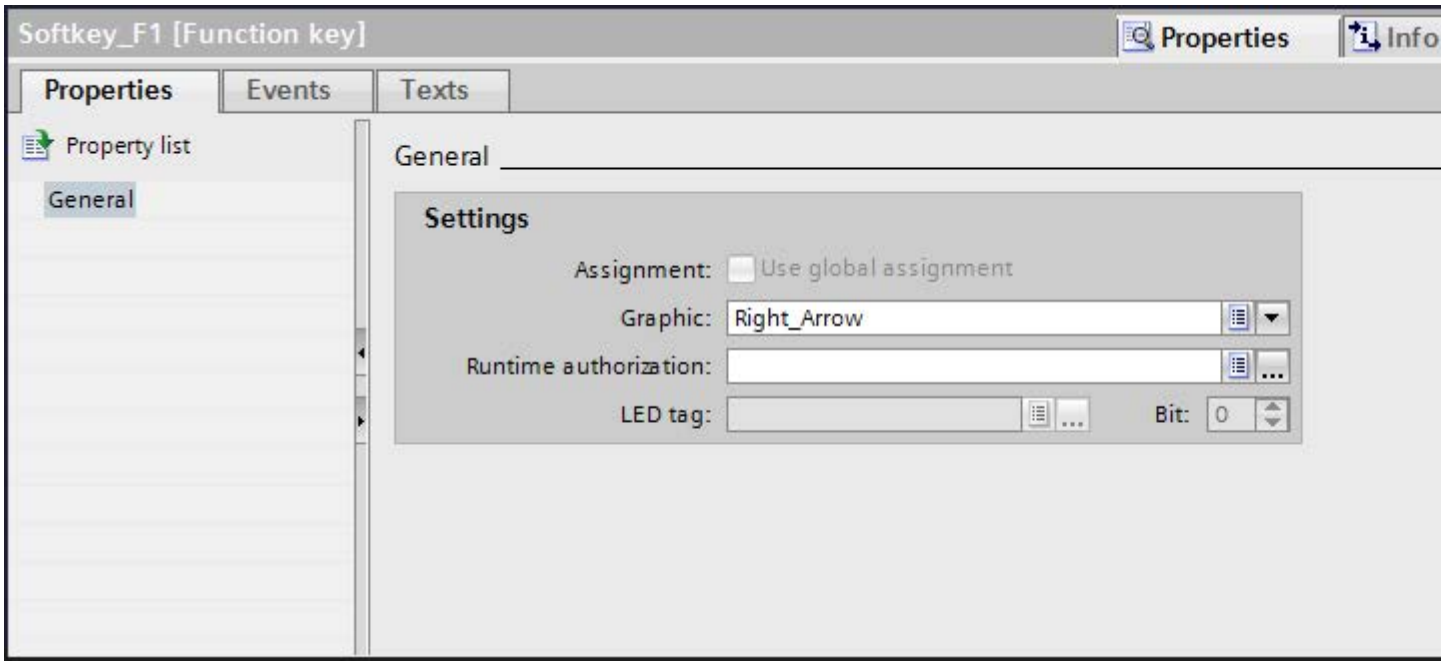
Requirement

- You have opened the project.
- The Inspector window is open.

Procedure

Proceed as follows to assign a screen-based function to a function key:

1. To open the "Global Screen" editor, double-click "Global Screen" in the "Screen management" group of the Project window.
2. Select the desired function key.
   The properties of the function key are shown in the Inspector window.



3. Under "General", configure a graphic and an operator authorization for the function key.
4. Configure a function list for the required event under "Events".

Result

If a local assignment does not overwrite the global assignment, the assignment of the function key changes in all the screens of the set HMI device in accordance with your entry.

10.1.5.3 Local assignment of function keys

Introduction

Function keys are assigned globally and locally. A local assignment of the function keys is only valid for the screen or the template in which it was defined. The following local function keys are available:

- Local function keys of a screen
  Different functions are assigned to the function key for each screen. This assignment applies only to the screen in which you have defined the function key.
- Local function keys of a template
  You assign the function keys in a template. The assignment applies to all the screens that are based on this template and are not overwritten by a local assignment in a screen.

A local assignment overwrites the global assignment of a function key.

Note

Availability for specific HMI devices

Function keys are not available on all HMI devices.

Using existing assignments

The option for using existing assignments is referred to as follows in the Inspector window:

- In a template: "Use global assignment"
- In a screen:
  - Screen based on a template: "Use local template"
  - Screen not based on a template: "Use global assignment"

The "Use local template" option includes the use of the local assignment in the template and the global assignment.

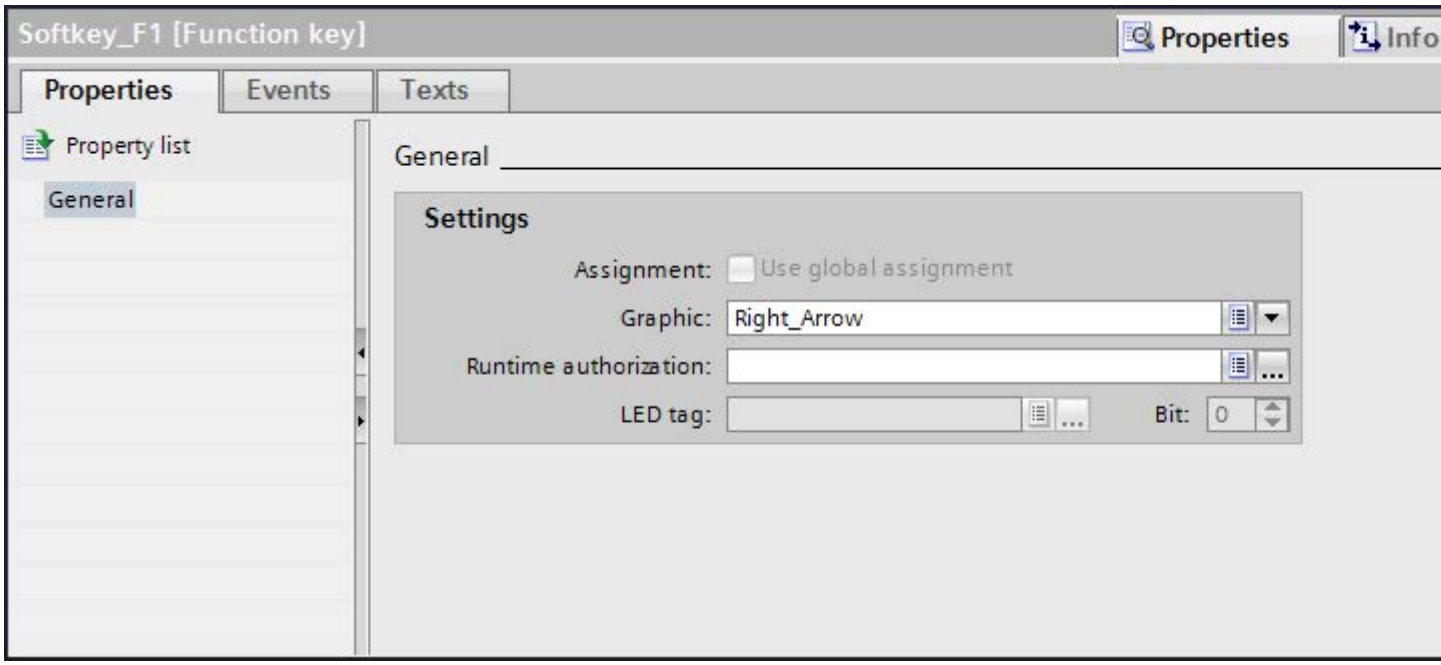
Requirement

- You have opened the screen or the template in which you want to assign a function key locally.
- The Inspector window is open.

Procedure

Proceed as follows:

1. Select the desired function key in the screen or in the template.
   The properties of the function key are shown in the Inspector window.
2. Click "General" in the Inspector window.



3. Disable the "Use local template" or "Use global assignment" option.
4. Under "General", configure a graphic and an operator authorization for the function key.
5. Configure a function list for the required event under "Events".

Result

The function key is assigned the configured function in the screen or in the template.

10.1.5.4 Assigning a function key to a function

Introduction

A function key can have two states:

- Pressed: Defined by the "Key pressed" event.
- Released: Defined by the "Release key" event.

Both of these events are configured in the Inspector window of the function key. You can assign any event a function list which contains system functions or scripts. Execution of this function list is event-driven in runtime.

Note
Availability for specific HMI devices

Function keys are not available on all HMI devices.

Note
Basic Panels

Scripts are not available for Basic Panels.

Requirement

To assign the function key a global function:

- The "Global Screen" editor is open.

To assign the function key a local function:

- The screen in which you want to assign a function key is open.

If you want to assign a function key locally in a template:

- The template in which you want to assign a function key is open.
- The Inspector window is open.

Procedure

Proceed as follows:

1. Select the function key you want to define.
   The properties of the function key are shown in the Inspector window.
2. Configure the function list for the desired result in the Inspector window under "Properties" in the "General" group.

Result

The function list is executed in runtime when the operator presses or releases the function key.

10.1.5.5 Assigning operator authorization for a function key

Introduction

In WinCC you can assign an operator authorization for a function key in runtime. This allows you to restrict access to the function keys to specific persons or operator groups when you

configure your project. Only authorized personnel can then change important parameters and settings in runtime.

You configure access protection in order to avoid operator errors and increase plant or machine security.

Note

Availability for specific HMI devices

Function keys are not available on all HMI devices.

Requirement

- The user groups have been defined.

To protect a global function key:

- The "Global Screen" editor is open.

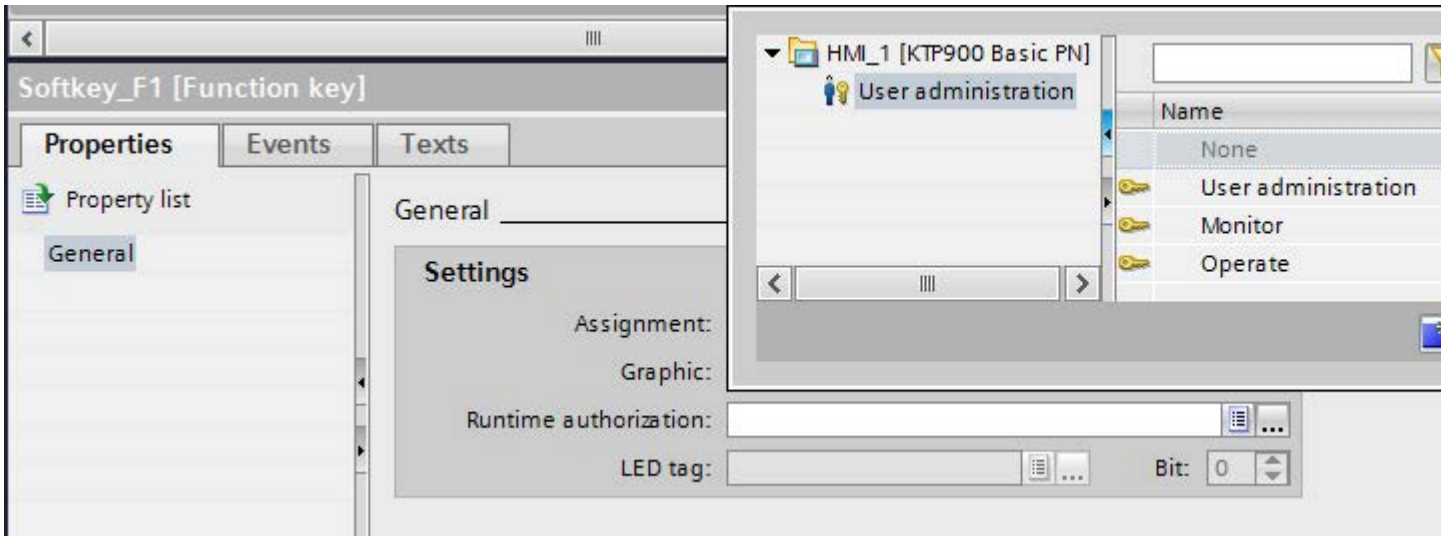
If you want to protect a local function key of a screen or of a template:

- The screen or the template which contains the function key is open.
- The Inspector window is open.

Procedure

Proceed as follows:

1. Select the relevant function key.
   The properties of the function key are shown in the Inspector window.
2. Click "General" in the Inspector window.



3. In the "Authorization" list, select the user group you want to allow runtime access to the function key.

Result

The operator authorization is configured.

10.1.5.6 Assigning a function key to a graphic

Introduction

In order to make the function of a key more clear, you can insert a graphic in the screen alongside the function key. Graphics can only be assigned to function keys that border the screen margin of the HMI device.

Note

Availability for specific HMI devices

Function keys are not available on all HMI devices.

Requirement

To assign a graphic to a global function key:

- The "Global Screen" editor is open.

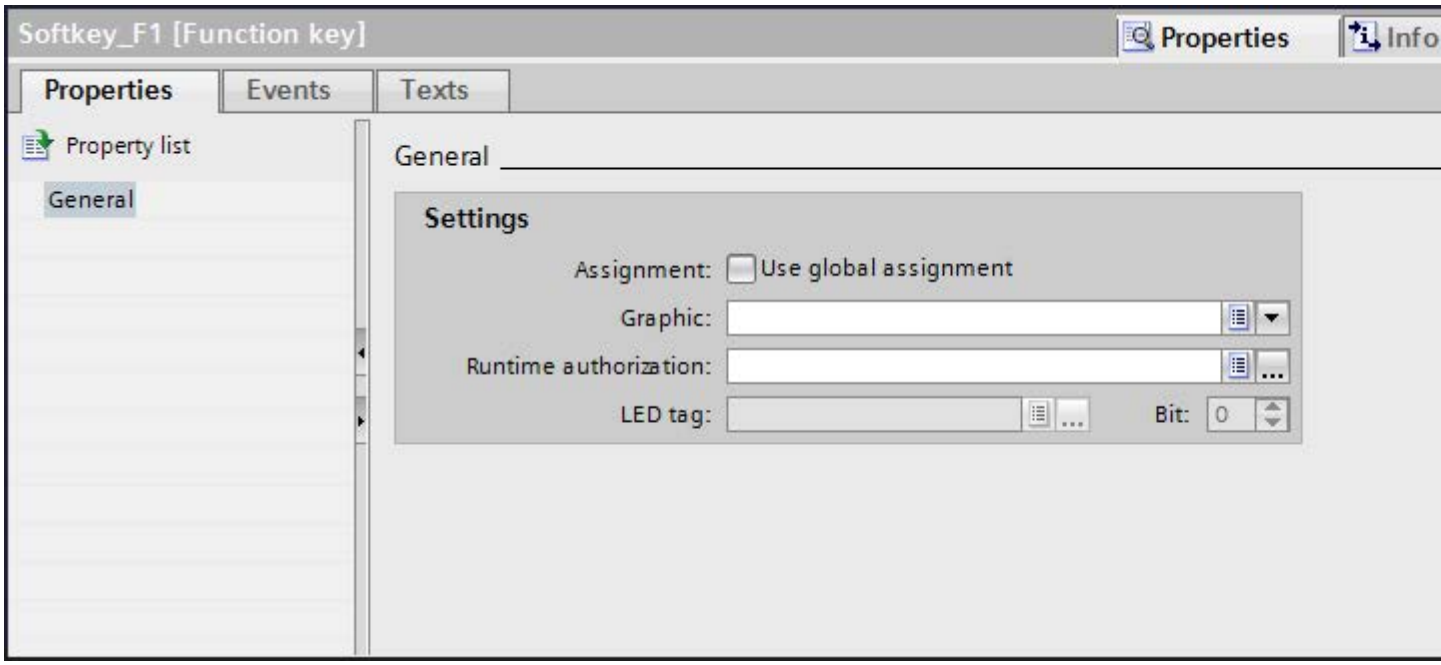
If you want to assign a graphic to a local function key in a screen or template:

- The screen or the template that contains the corresponding function key is open.
- The Inspector window is open.
- You have created the graphic for the function key.

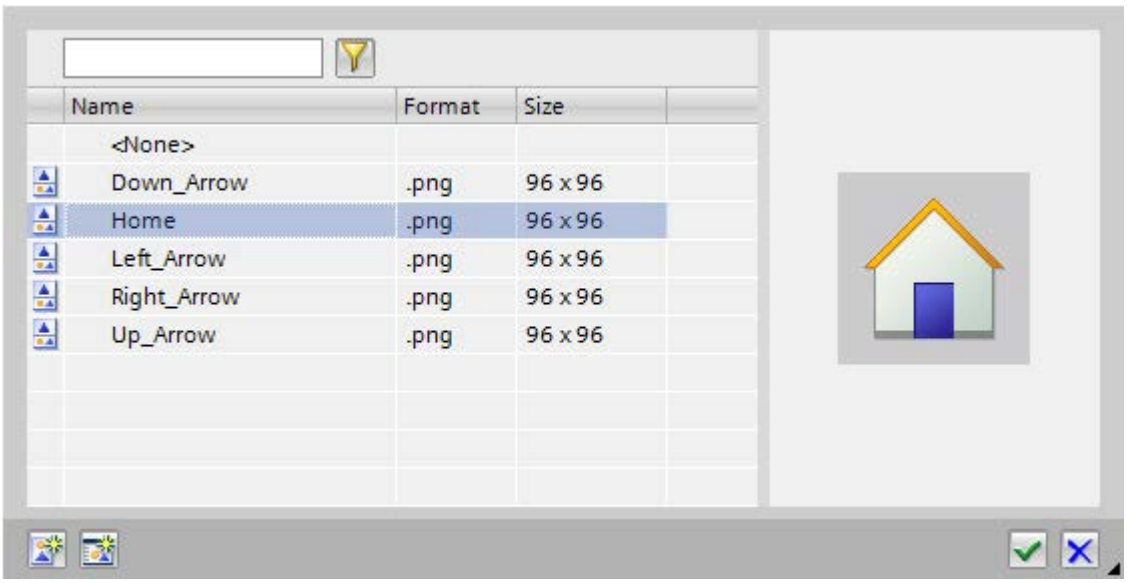
Procedure

Proceed as follows:

1. Select the relevant function key.
   The properties of the function key are shown in the Inspector window.
2. Click "General" in the Inspector window.



3. Click in the "Graphic" area of the list.
   The graphic browser of your WinCC project appears. The pane on the left side shows the external graphics which are stored in the browser. The pane on the right side shows you a preview of the graphic you have selected in the browser.



   Using the and icons, you can display the collection either in form of thumbnails or as a list.
   In order to open and edit OLE objects with the associated graphics program, double-click on the object.

4. In the browser click the desired graphic or store the relevant graphic in the graphic browser.
   The graphic preview is shown in the right pane.
5. Click "Select" to add the graphic to the screen.
   Click "Clear" to remove the graphic from the screen.

Result

The graphic is displayed next to the function key.

10.1.5.7 Configuring LED tags

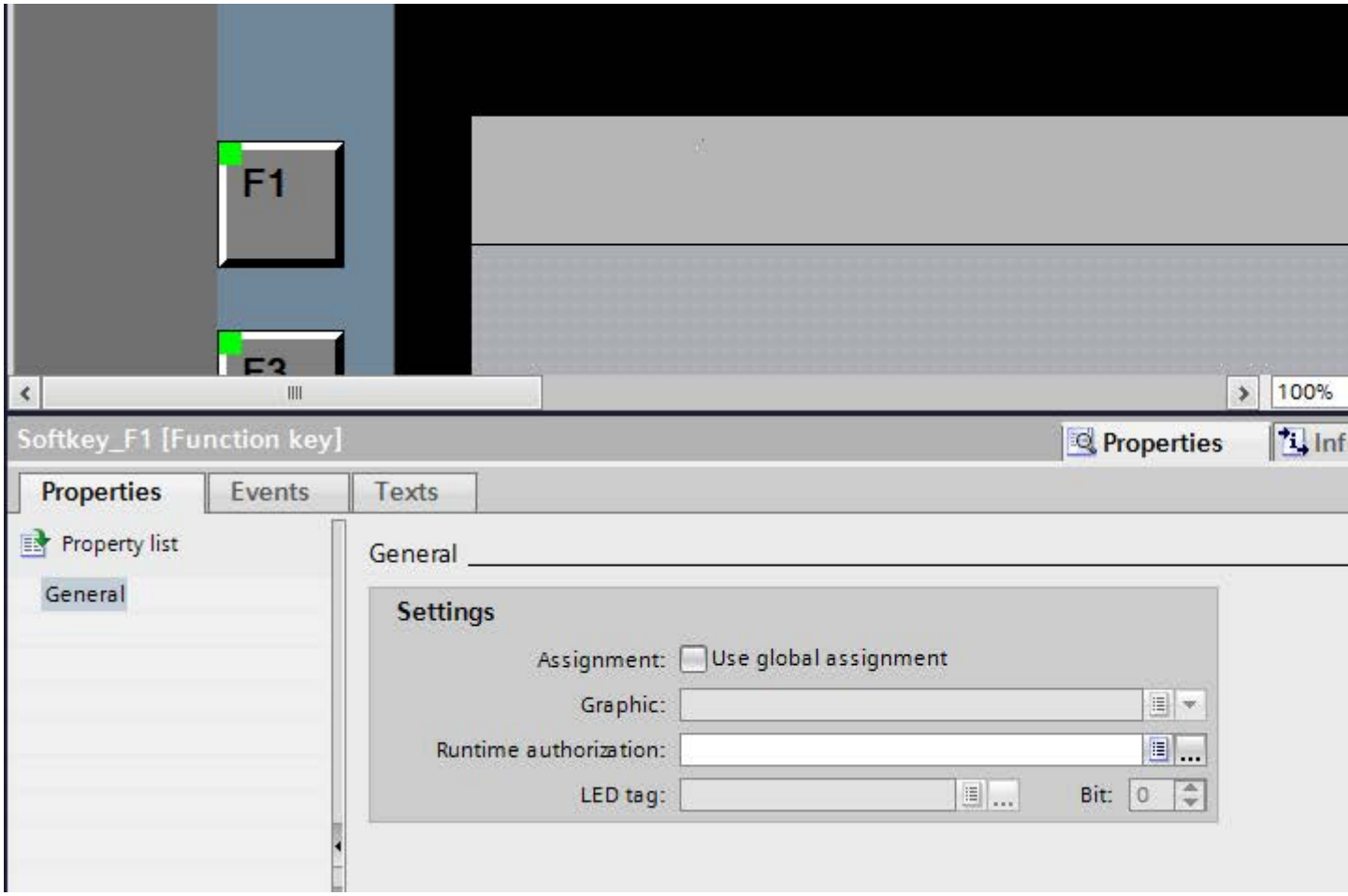
Requirements

- An HMI device with key operation has been created.
- You have created an LED tag.

Procedure

1. Create a new screen.
2. Click an F-key of the HMI device.

3. In the Inspector window, click "Properties > Properties > General".



4. Select a tag under "LED tag" in the "General > Settings" area.
5. Under "Bit" enter the correct bit number.
   The correct bit number depends on the HMI device and the input and output assignments on the HMI device.

Assignment of inputs and outputs

The exact assignment of inputs and outputs can be found under:

- PROFINET IO direct keys: Auto-Hotspot
- PROFIBUS DP direct keys: Auto-Hotspot

10.1.5.8 Example: Using function keys for screen navigation

Task

In this example you create a local function key in a screen. When the operator presses this function key, a screen change to a predefined screen is triggered, for example "Boiler 2".

Note

Availability for specific HMI devices

Function keys are not available on all HMI devices.

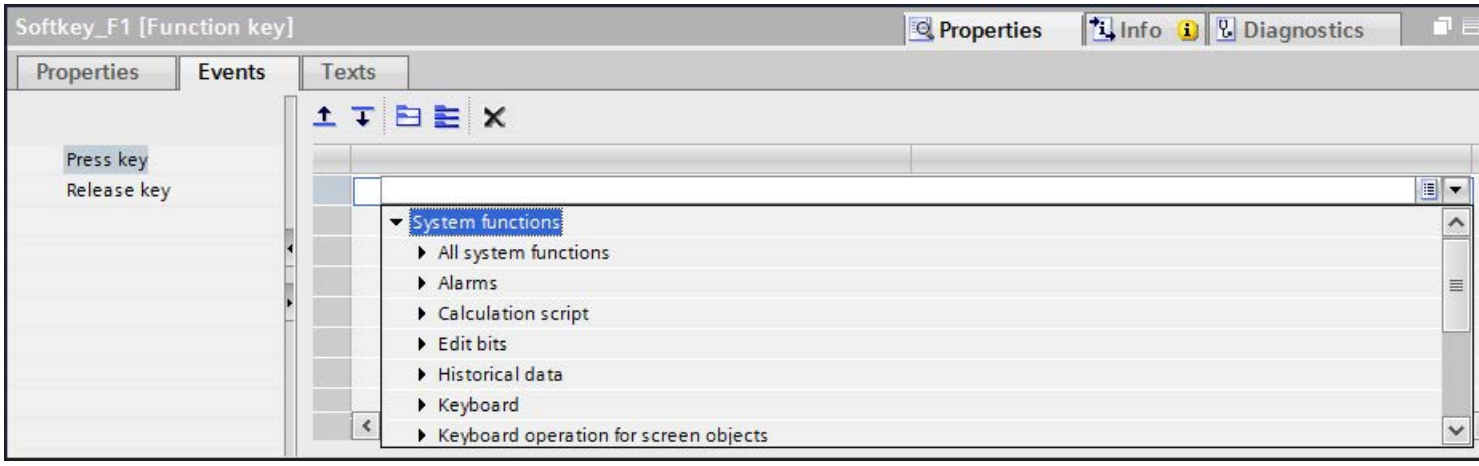
Requirement

- The screen in which you want to assign the function key is open.
- You have created the "Boiler 2" screen.
- The Inspector window is open.

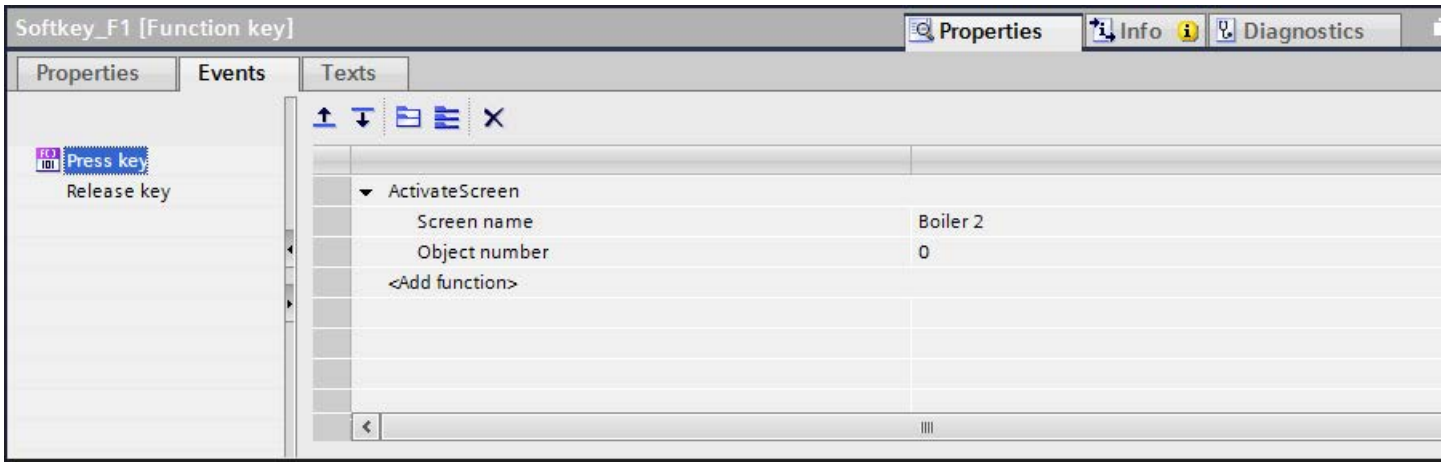
Procedure

Proceed as follows to use the "ActivateScreen" function:

1. Select the desired function key.
   The properties of the function key are shown in the Inspector window.
2. Click "General."
3. To overwrite a global assignment, disable the "Use local template" option.
4. Click "Key pressed" under "Events".



5. Select the "ActivateScreen" system function from the list.
   The "ActivateScreen" function appears in the "Function list" dialog box, including the "Screen name" and "Object number" parameters.



6. Select the "Boiler 2" screen from the "Screen name" list.

Result

The operator changes to the "Boiler 2" screen in runtime by pressing the selected function key.

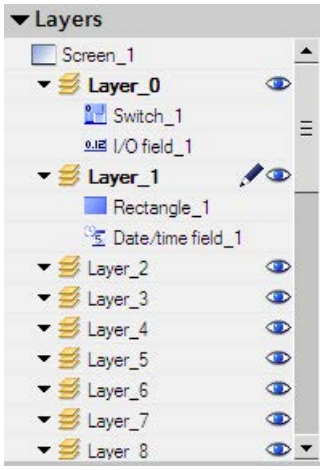
10.1.6 Working with layers

10.1.6.1 Basics on working with layers

Layers

Use layers in order to achieve differentiated editing of the objects in a screen. A screen consists of 32 layers that you can give any names. If you assign objects to the layers, you thereby define

the screen depth. Objects of layer 0 are located at the screen background, while objects of layer 31 are located in the foreground.



The objects of a single layer are also arranged hierarchically. When you create a screen, the object inserted first is located at the rear within the layer. Each further object is placed one position towards the front. You can shift objects forwards and backwards within a layer.

Principle of the layer technique

Always one layer of the 32 layers is active. New objects you add to the screen are always assigned to the active layer. The number of the active level is displayed in the inspector window of the screen and in the "Layout > Layers" task card.

When you open a screen, all 32 layers of the screen are displayed. You can hide all the layers except for the active layer in the inspector window of the screen and in the "Layout > Layers" task card. You then explicitly edit objects of the active layer.

In the tree view of the "Layers" palette in the "Layout" task card, you administer layers and objects with drag-and-drop and the context menu.

Application examples

Use layers, for example, in the following cases:

- To hide the labeling of objects when editing,
- To hide objects, e.g. alarm windows, while configuring other objects

10.1.6.2 Moving objects between layers

Introduction

By default, a new object is inserted on the active layer. You can, however, assign an object to another layer at a later time.

Requirement

- A screen with an object is open.
- The Inspector window is open.

Procedure

1. Select the object in the screen.
   The object properties are displayed in the Inspector window.
2. Enter the layer to which you want to move the object in "Properties > Properties > Miscellaneous > Layer" in the Inspector window.

Alternatively, select the object from the "Layout" task card and drag it to the required layer.

Changing the order of objects

1. Select the object in the screen.
   The object properties are displayed in the Inspector window.
2. To move the object to the front or back, select the "Order" > "Move backward" or "Move forward" command from the shortcut menu.
   Alternatively, use the or button in the toolbar.

Result

The object is assigned to the selected layer, and positioned at the top of the layer.

10.1.6.3 Setting the active layer

Introduction

The screen objects are always assigned to one of the 32 layers. There is always an active layer in the screen. New objects you add to the screen are always assigned to the active layer.

The number of the active layer is indicated in the "Layer" toolbar. The active layer is indicated by the icon in the "Layout > Layers" task card.

Layer 0 is the active layer when you start programming. You can activate a different layer during configuration, if necessary.

Requirement

- You have opened a screen which contains at least one object.
- The Inspector window of the active screen is open.

Procedure

1. Click "Properties > Properties > Layers" in the Inspector window of the current screen.
2. Enter the layer number in "Settings > Active layer".

Alternative procedure

1. Select "Layout > Layers" in the "Layout" task card.
2. Select the "Set to active" command from the shortcut menu of a layer.

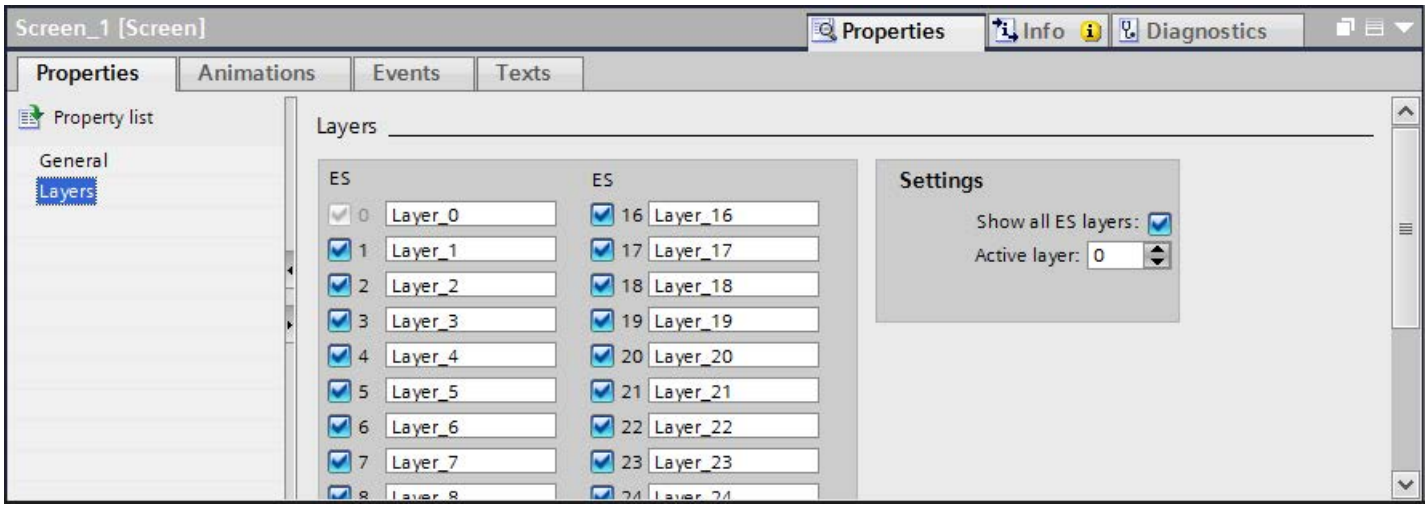
Result

The layer with the specified number is now active.

10.1.6.4 Show and hide layers

Introduction

You can show or hide the layers of a screen as required. You specify the layers that are shown in the Engineering System. When you open a screen, all the layers are always shown.



Requirement

- The screen is opened.
- The "Layout" task card is open.

Procedure

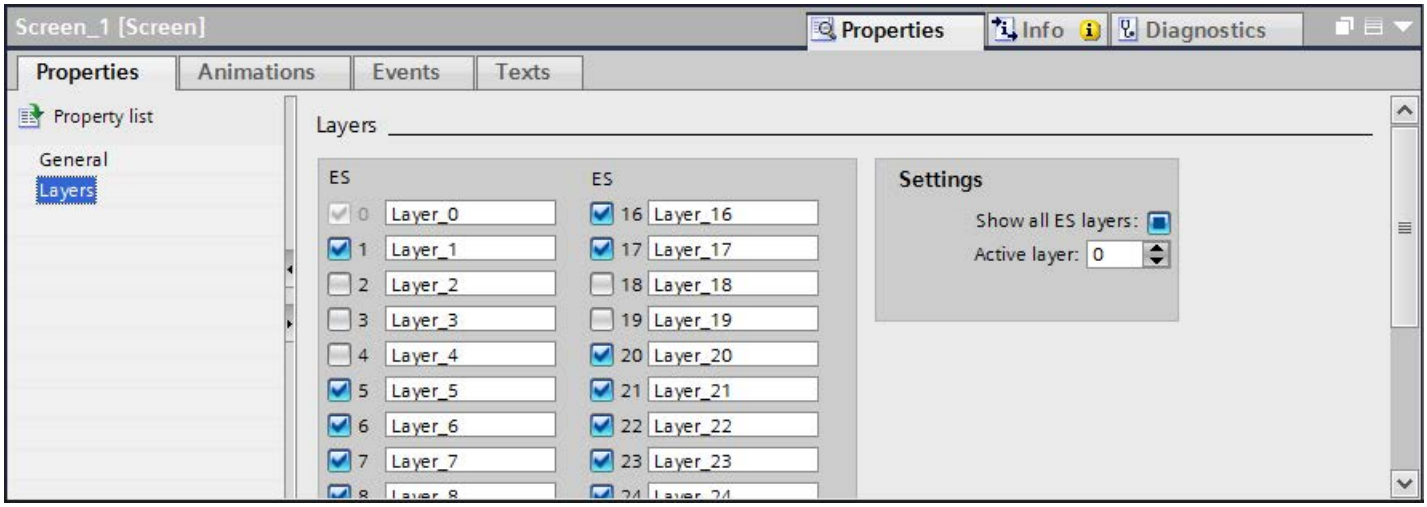
1. Select the layer that you want to hide or show in the "Layout > Layers" task card.
2. Click one of the icons next to the corresponding layer:
   - A shown layer is hidden
   - A hidden layer is shown

Note

The active layer cannot be hidden.

Alternative procedure

1. Click in an area of the screen that does not contain an object.
   The screen properties are shown in the Inspector window.
2. In the Inspector window, select "Properties > Properties > Layers":



3. In the list, disable the levels you wish to hide.
   If you activate "All ES layers" for a layer, the objects in this layer will be shown in the Engineering System.

Result

The layers are shown according to your settings.

10.1.6.5 Renaming layers

Introduction

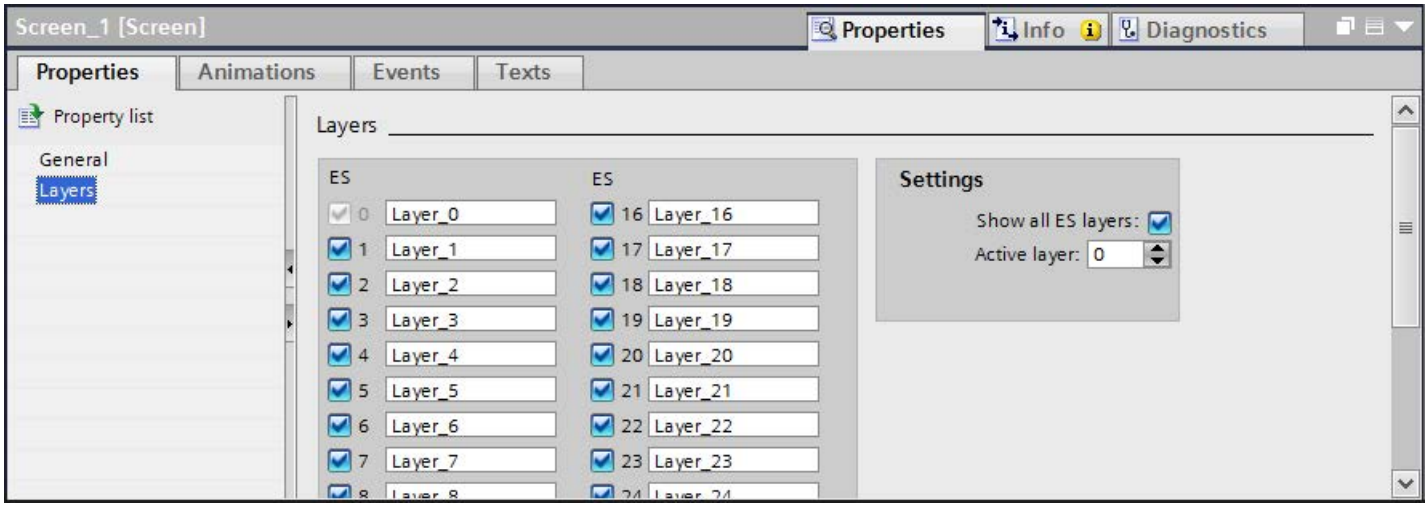
When you create a screen, the 32 layers are numbered consecutively by default. To improve clarity, you can rename the layers to suit your requirements.

Requirement

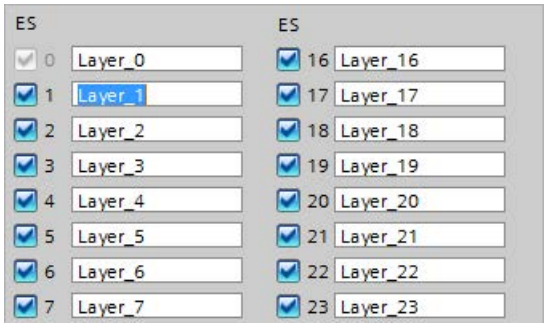
- The screen is opened.

Procedure

1. Click in an area of the screen that does not contain an object.
   The screen properties are shown in the Inspector window.
2. In the Inspector window, select "Properties > Properties > Layers".



3. Enter the new layer name.



Result

The layer is displayed with the new name.

10.1.7 Display and operating objects

10.1.7.1 Device-Specific Nature of the Objects

Objects for Basic Panels

Availability of display and operating objects for Basic Panels

Only the objects which can be used for the device you are configuring will be shown in the object window. The following table shows the availability of indicator and operating objects for the Basic Panels.

Overview

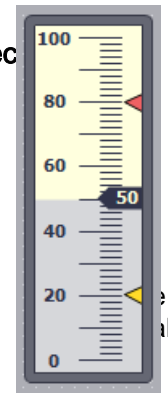
	KP300 Basic KP400 Basic	KTP400 Basic KTP600 Basic KTP1000 Basic TP1500 Basic	KTP700 Basic KTP900 Basic
Bar	Yes	Yes	Yes
User view	Yes	Yes	Yes[1)]
Date/time field	Yes	Yes	Yes
I/O field	Yes	Yes	Yes
Ellipse	Yes	Yes	Yes
Graphic view	Yes	Yes	Yes
Graphic I/O field	Yes	Yes	Yes
Help indicator	Yes	No	No
Circle	Yes	Yes	Yes
Trend view	Yes	Yes	Yes[1)]
Line	Yes	Yes	Yes
Alarm view Alarm window	Yes	Yes	Yes[1)]
Alarm indicator	Yes	Yes	Yes
Rectangle	Yes	Yes	Yes
Recipe view	Yes	Yes	Yes[1)]
Button	Yes	Yes	Yes
Switch	Yes	Yes	Yes
Symbolic I/O field	Yes	Yes	Yes
System diagnostics view	Yes	Yes	Yes[1)]
Text field	Yes	Yes	Yes

1) The objects are displayed on the panels as table-based objects.

See also

Overview of objects (Page 7978)

10.1.7.2 Objec



Bar

Application

e displayed graphically using the "Bar" object. The bar graph can be labeled with alues.

Layout

In the Inspector window, you customize the settings for the position, shape, style, color, and font types of the object. You can adapt the following properties in particular:

- Color transition: Specifies the change in color display when limit values are exceeded.
- Displaying the limit lines / limit markers: Shows the configured limit as a line or marker.
- Define bar segments: Defines the gradations on the bar scale.
- Define scale gradation: Defines the subdivisions, scale markings and intervals of a bar scale.

Color transition

You define how the color change is represented in "Properties > Properties > Appearance" in the Inspector window.

Color transition	Description
"Segmented"	If a particular limit was reached, the bar changes color segment by segment. With segment by segment representation, you visualize, for example, which limits are exceeded by the displayed value.
"Entire bar"	If a particular limit was reached, the entire bar changes color.

Displaying limit lines and limit markers

You display the configured limit in the bar as a line or marking in Runtime using the "Lines" and "Markings" property:

1. In the Inspector window, select "Properties > Properties > Appearance":
2. Activate "Lines" and "Markings".

Define bar segments

Use the "Subdivisions" property to define the number of segments into which the bar is divided by the main gradations on the scale.

Use the "Interval" property to divide the distance between the main gradations. The value appears as the difference in value between two adjacent main gradations:

1. In the Inspector window, select "Properties > Properties > Scales":
2. Activate "Show scale."
3. Select the corresponding value for "Settings > Subdivisions".
4. Select the corresponding value for "Settings > Marks label".
5. Select the corresponding value for "Large interval > Interval".

See also

Device-Specific Nature of the Objects (Page 8075)

User view

Application

The "User view" object is used to set up and administer users and authorizations.

Administrator	Administrator group
Meister	Group_2
Operator_1	Operator
PLC User	Unauthorized
<New user>	

Note

Do not use the user view in a group.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Number of lines: Specifies the maximum number of visible entries.

Number of lines

The number of lines in the user view displayed in Runtime is specified in the Inspector window. The setting for the number of lines is only effective if the property "Fit object to contents" is active.

1. In the Inspector window, select "Properties > Properties > View".
2. Enter an integer value under "Number of lines".
3. In the Inspector window, activate "Properties > Properties > Layout".
4. Activate "Fit object to contents".

See also

Device-Specific Nature of the Objects (Page 8075)

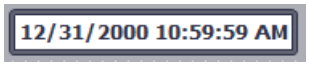
Simple user view (Page 8936)

Configuring a user view (Page 8258)

Date/time field

Use

The "Date/time field" object shows the system time and the system date. The appearance of the "Date/time field" depends on the language set on the HMI device.



Layout

In the Inspector window, you customize the position, style, colors and font types of the object. You can adapt the following properties in particular:

- Display system time: Specifies that the system time is displayed.
- Include tag: Specifies that the time of the connected tag is displayed.
- Long date/time format: This setting defines the format displayed for the data and time.

Display system time

The time displayed in the "Date/time field" on the HMI device is specified in the inspector window.

1. In the Inspector window, select "Properties > Properties > General".
2. Activate "Format > System time".

Using tags

The time of the interconnected tag is displayed in the date/time field.

1. In the Inspector window, select "Properties > Properties > General".
2. In the "Format" area, select a tag with the "DateTime" data type, e.g. an internal tag.

Long date/time format

Visualization of the date and time is specified in the "Format" area under "General" in the inspector window.

Option	Description
"Enabled"	Date and time are displayed in full, e.g. "Sunday, December 31, 2000 10:59:59 AM"
"Disabled"	Date and time are displayed in short form, e.g. "12/31/2000 10:59:59 AM"

See also

Device-Specific Nature of the Objects (Page 8075)

I/O field

Application

The "I/O field" object is used to enter and display process values.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Mode: Specifies the response of the object in Runtime.
- Display format: Specifies the display format in the I/O field for input and output of values.
- Hidden input: Specifies whether the input value is displayed normally or encrypted during input.

Note

Reports

In reports, I/O fields only output data. "Output" mode is preset. Properties for configuring input are not available, e.g. "hidden input".

Mode

The response of the I/O field is specified in the Inspector window in "Properties > Properties > General > Type".

Mode	Description
"Input"	Values can only be input into the I/O field in runtime.
"Input/output"	Values can be input and output in the I/O field in runtime.
"Output"	The I/O field is used for the output of values only.

Layout

The "display format" for the input and output of values is specified in "Properties > Properties > General > Format" in the Inspector window.

Layout	
"Binary"	Input and output of values in binary form
"Date"	Input and output of date information. The format depends on the language setting on the HMI device.
"Date/time"	Input and output of date and time information. The format depends on the language setting on the HMI device.
"Decimal"	Input and output of values in decimal form
"Hexadecimal"	Input and output of values in hexadecimal form
"Time"	Input and output of times. The format depends on the language setting on the HMI device.
"Character string"	Input and output of character strings.

Note
Data formats

Not all data formats are available for selection for Runtime Professional.

Hidden input

In Runtime the input can be displayed normally or encrypted, for example for hidden input of a password. A "*" is displayed for every character during hidden input. The data format of the value entered cannot be recognized.

1. In the Inspector window, select "Properties > Properties > Response":
2. Activate "Hidden input".

Avoid overlaps in output fields

If several I/O fields are configured as output fields with a transparent background in a screen, these I/O fields may overlap. The transparent part of the one field covers the digits of the other field. This may cause display problems in Runtime. In order to avoid such overlaps, set the margins of the I/O fields to zero in the object properties under "Properties > Properties > Appearance". Activate "Properties > Properties > Layout > Fit object to contents."

Limits

Under "Properties > Properties > Limits" in the Inspector window, set the colors for the values that violate the high or low limits.
You define the limits via the properties of a tag.
In Runtime Professional, you can also define the limit range for the entry in the I/O field via "Properties > Properties > Limits".

See also

Device-Specific Nature of the Objects (Page 8075)

Ellipse

Application

The "Ellipse" is an enclosed object that can be filled with a color or pattern.

Layout

In the Inspector window you can customize the settings for the object position, geometry, style, frame and color. You can adapt the following properties in particular:

- Horizontal radius: Specifies the horizontal radius of the elliptical object.
- Vertical radius: Specifies the vertical radius of the elliptical object.

Horizontal radius

The horizontal radius of the "Ellipse" object is specified in the Inspector window. The value is entered in pixels.

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter a value between 0 and 2500 under "Horizontal."

Vertical radius

The vertical radius of the "Ellipse" object is specified in the Inspector window. The value is entered in pixels.

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter a value between 0 and 2500 at "Vertical."

See also

Device-Specific Nature of the Objects (Page 8075)

Rotating objects (Page 7993)

Flipping objects (Page 7995)

Designing a border (Page 8002)

Graphic view

Application

The "Graphic view" object is used to display graphics.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Graphic: Specifies the graphic file that is displayed in the object.
- Stretch graphic: Specifies the automatic size stretching for objects with graphics.
- Transparent color: Specify whether or not the transparent color is used for the graphic.

Inserting graphics

The following graphic format is used in the "Graphic view" object: *.bmp, *.tif, *.png, *.ico, *.emf, *.wmf, *.gif, *.jpg or *.jpeg. You may also use graphics as OLE objects in the Graphic view .

1. In the Inspector window, select "Properties > Properties > General":
2. Select the graphic that you wish to insert.
   The graphic preview is shown in the right pane.
3. Click "Apply" to insert the graphic in the Graphic view .

Stretch graphic

Whether a graphic displayed in a Graphic view is stretched to the size of the Graphic view in runtime is specified in the Inspector window.

1. Click "Properties > Properties > Layout" in the inspector window.
2. Select the required size adjustment for the graphic.

Transparent color

This property defines whether the transparent color is used for the graphic to be displayed.

1. Click "Properties > Properties > Appearance" in the inspector window:
2. Activate "Background > Transparent".
3. Select a transparent color.

Note

When using bitmaps in WinCC screens the "Transparent color" setting demands a high character performance in the layout on the panel. Visualization performance is enhanced by disabling the "Transparent" setting in the properties of the relevant display object. This restriction applies in particular when bitmaps are used as background image.

Note

Basic Panels

The "Transparent" property is not available for Basic Panels.

See also

Device-Specific Nature of the Objects (Page 8075)

Storing an external image in the graphics library. (Page 8011)

Options for Editing Objects (Page 7981)

Objects for Basic Panels (Page 8075)

Graphic I/O field

Application

The "Graphic I/O field" object can be used to configure a list for display and selection of graphic files.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Mode: Specifies the response of the object in Runtime.
- Scroll bar type: Specifies the graphic layout of the scroll bar.

Note

Basic Panels

The scroll bar is not available for Basic Panels.

Note

Reports

Graphic I/O fields output exclusively graphics in reports. "Output" mode is preset. Properties for configuring the selection of graphics are not available, e.g. "scroll bar".

Mode

The response of the "Graphic I/O field" object is specified under "Properties > Properties > General > Type > Mode" in the Inspector window.

Mode	Description
"Input"	The "Graphic I/O field" object is only used to select graphics.
"Input/output"	The "Graphic I/O field" object is used to select and display graphics.

Mode	Description
"Output"	The "Graphic I/O field" object is used to display graphics only.
"Two states"	The "Graphic I/O field" object is only used to display graphics and can have a maximum of two states. You use no graphics list but insert one graphic each for the "ON" and "OFF" state.

Stretch graphic

Whether a graphic displayed in a graphic I/O field is stretched to the size of the view in runtime is specified in the Inspector window.

1. In the Inspector window, select "Properties > Properties > Layout".
2. Select the required size adjustment for the graphic.

Scroll bar type

The response for the graphic representation of the scroll bar is specified under "Properties > Properties > Appearance > Scroll Bar > Type" in the Inspector window.

Type	Description
"Permanent"	The scroll bar is always visible.
"No scrollbar"	The scroll bar is not visible.
"Visible after clicking"	The scroll bar is made visible by a mouse click.

See also

Device-Specific Nature of the Objects (Page 8075)

Symbolic I/O field (Page 8101)

Help indicator

Application

The object "help indicator" is available for the HMI devices OP 73 and KP300 Basic. If a a tooltip exists for the selected object, the help indicator is displayed during runtime. If a tooltip was configured for the opened screen, the help indicator always remains visible.

You configure the object "help indicator" exclusively in the global screen.

Layout

You can adapt the following properties in the Inspector window:

- Position: Determines the position of the object "Help indicator."

Position

You can use this property to set the position of the object "Help indicator."

1. Select the object "Help indicator" in the template.
2. In the Inspector window, select "Properties > Properties > Layout".
3. Enter a value for X and Y. You can also use the cursor keys to position the selected object.

If you have configured a screen object at this position, the visible help indicator covers the screen object. The help indicator is covered only by incoming system alarms and dialogs.

See also

Device-Specific Nature of the Objects (Page 8075)

Circle

Application

The "Circle" object is a closed object which can be filled with a color or pattern.

Layout

In the Inspector window you can customize the settings for the object position, geometry, style, frame and color. You can adapt the following properties in particular:

- Radius: Specifies the size of the circle.

Radius

The radius of the "Circle" object is specified in the Inspector window. The value is entered in pixels.

1. In the Inspector window, select "Properties > Properties > Layout".
2. Enter a value between 0 and 2500 in the "Radius" area.

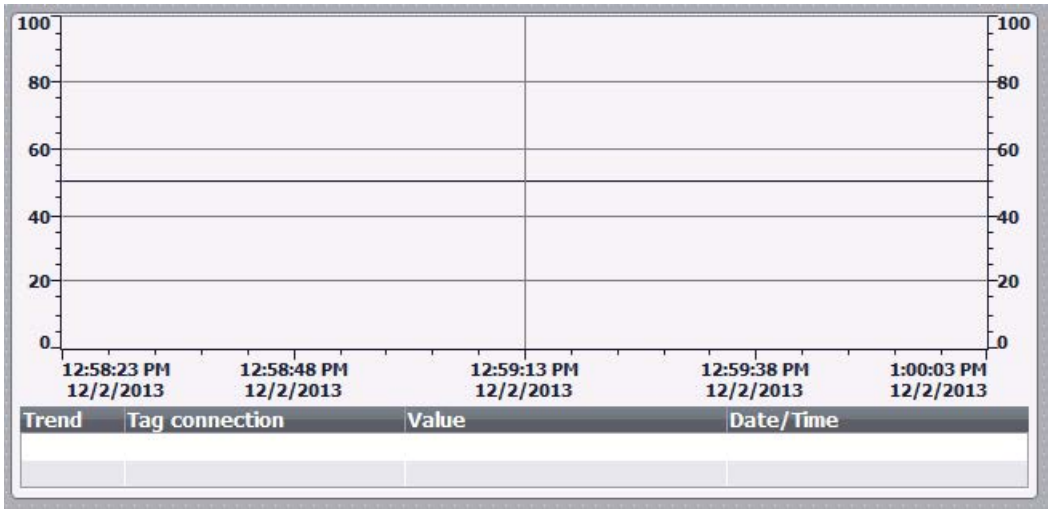
See also

Device-Specific Nature of the Objects (Page 8075)

Trend view

Application

The trend view is meant for the graphical representation of tag values from the current process or from a log in form of trends.



Layout

In the Inspector window, you customize the position, geometry, style, color, and font types of the object. You can adapt the following properties in particular:

- Display value table, ruler and grid: Specifies whether a value table, a ruler or a grid is displayed in addition to the coordinate system to improve legibility.
- Toolbars: Defines the display of the control elements.

Display value table, ruler and grid

For improved legibility a value table, a ruler and a grid can be displayed in Runtime.

1. Activate "Properties > Properties > Appearance > Show ruler".
2. Activate "Properties > Properties > Table > Show table".
3. Activate "Properties > Properties > Table > Show grid".

Toolbars

The layout of the control elements is defined in the "Properties > Properties > Toolbar" inspector window.

Note

Basic Panels

As archiving is not possible for Basic Panels, the control elements are not available.

Toolbar button	Brief description	Description
	"Go to start"	Scrolls back to the beginning of the trend recording. The start values with which the trend recording started are displayed.
	"Zoom in"	Zooms out of the displayed time section
	"Zoom out"	Zooms into the displayed time section.
	"Ruler backward"	Moves the ruler back.
	"Ruler forward"	Moves the ruler forward.
	"Backward"	Scrolls back one display width.
	"Forward"	Scrolls forward one display width.
	"Ruler"	Shows or hides the ruler. The ruler displays the X-value associated with a Y-value.
	"Start/stop"	Stops trend recording or continues trend recording.

Configuration behavior

Displaying column headers

The layout of the table in the trend view depends on the view settings in the Control Panel. Depending on the setting, the column headers may be truncated. This setting is found under "Display > Appearance" in the control panel. To display column headers correctly, set the display in "Windows and buttons" to "Windows Classic" style.

This behavior only occurs during configuration. The column headers are displayed correctly in Runtime.

Consistency test

If warnings or errors are displayed in the output window during a consistency check in connection with trend views, clicking "Go to Error/Tag" on the shortcut menu will not always take you to the exact error position. In some cases only the trend view is shown as cause of error.

Adding, configuring, and removing trends

The trends of the trend view are managed in the Inspector window under "Properties > Properties > Trend". You can copy trends between different trend views.

See also

Device-Specific Nature of the Objects (Page 8075)

Configuring trend displays for values from the PLC (Page 8147)

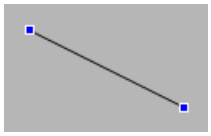
Touch and key operation (Page 8930)

Overview (Page 8929)

Line

Application

The "Line" object is an open object. The line length and gradient slope are defined by the height and width of the rectangle enclosing the object.



Layout

In the Inspector window, you customize the settings for the object position, shape, style, and color. You can adapt the following properties in particular:

- Line style
- Line start and end

Line style

The representation of the line is specified under "Properties > Properties > Appearance" in the Inspector window. The line is shown without interruption if you select "Solid", for example.

Note

The line styles available depend on the selected HMI device.

Line start and end

The start and end points of the line are specified under "Properties > Properties > Appearance > Line ends" in the Inspector window.

Use arrow point, for example, as start and end point. The available start and end points depend on the device.

See also

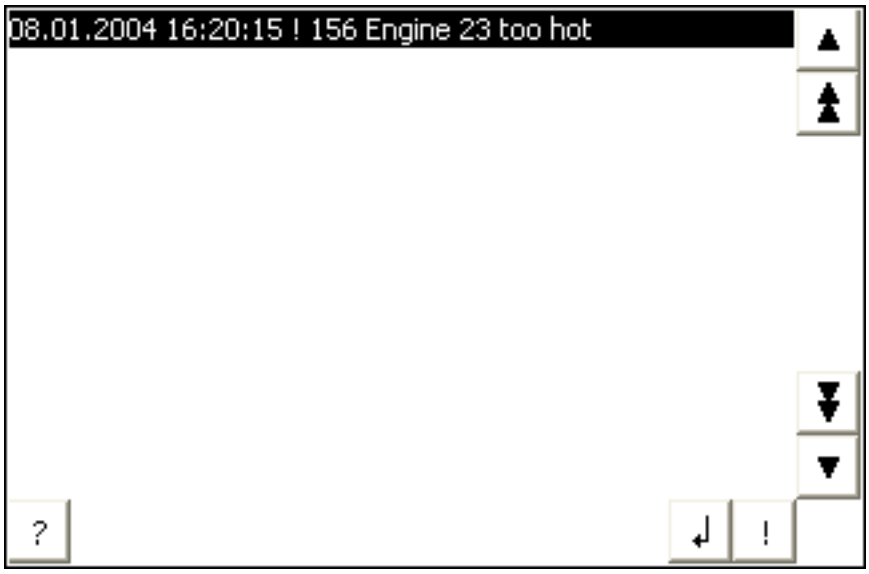
Device-Specific Nature of the Objects (Page 8075)

Alarm view

Application

Alarms are indicated in the Alarm view or in the Alarm window of the HMI device.

The following screen contains a simple alarm view:



Layout

In the Inspector window, you customize the position, geometry, style, color and font types of the object.

Note

The fonts available for selection depend on the fonts you have configured in the Runtime settings under "Language & font" and the fonts supported by your HMI device.

You can adapt the following properties in particular:

- Control elements: Defines the operator controls of the alarm display.
- Alarm classes: This setting defines which alarm classes are displayed in the alarm view.
- Columns: Specifies the displayed columns in runtime.

Note

If you have different alarm classes output, these will be initially sorted into alarm classes in runtime, and then by when the alarm occurred.

Control elements

The control elements that can be used to control the alarm display in runtime are specified in the Inspector window under "Display > Settings". The following table shows the control elements in the alarm view, and what they do:

Button		Function
"Info text"		Displays info text for an alarm.
"Acknowledge"		Acknowledges an alarm.
"Loop-In-Alarm"		If a screen change is configured, the display switches to a screen containing information about the alarm.

Select alarm classes

1. Click "Properties" in the Inspector window.
2. Under "Alarm classes" activate the alarm classes to be displayed in the alarm view in runtime .

Define columns

Define the columns to be displayed in the alarm view in runtime in the Inspector window.

1. In the Inspector window, click "Properties > Columns".
2. Activate the columns that are to be displayed in runtime under "Columns".

Displaying column headers

The layout of the alarm view is dependent on the view settings in the control panel. Depending on the setting, the column headers may be truncated. This setting is found under "Display > Layout tab" in the control panel. To display column headers correctly, set the display in "Windows and buttons" to "Windows Classic" style.

This behavior only occurs during configuration. The column headers are displayed correctly in runtime.

Note

In the engineering system you can dynamically control the visibility of an object, for example, in the "Animations" group of the Inspector window. In runtime, the "Simple alarm view" does not support animations. If you have configured an animation and, for example, wish to perform a consistency check of the project, then an error alarm is issued in the Output window.

See also

Device-Specific Nature of the Objects (Page 8075)

Alarm window (Page 8092)

Alarm indicator (Page 8095)

Configuring an alarm view (Page 8173)

Alarm window

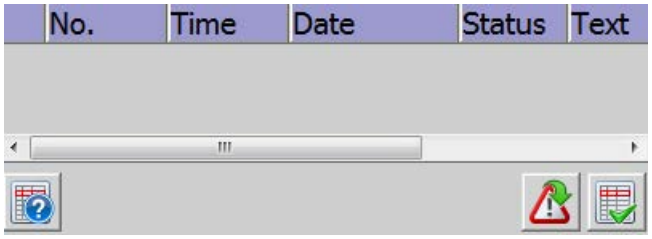
Use

Alarms are indicated in the Alarm view or in the Alarm window of the HMI device. The layout and operation of the Alarm window are similar to that of the Alarm view. The Alarm window has the following characteristics that are the same as in the Alarm view:

- Alarm window
- Alarm line: The alarm line is not available on Basic Panels.

The Alarm window is configured in the "Global screen" editor.

The alarm window is not assigned to any screen. Depending on the configuration the alarm window is opened when an alarm that belongs to a specific alarm class is active. Depending on the configuration, it is not closed until the alarm is acknowledged.



Note

In the engineering system you dynamize, for example, the visibility of an object in "Properties > Animations" in the Inspector window. In runtime, the "Simple alarm window" object does not support animations. If you have configured an animation and, for example, wish to perform a consistency check of the project, then an error alarm is issued in the Output window.

Layout

In the Inspector window, you customize the position, geometry, style, color and font types of the object. You configure the Alarm window in the same way as the Alarm view. In addition you adapt the following properties:

- Fixed alarm window: Specifies that the Alarm window retains the focus after a screen change.
- Window: You define the operator input and response of the Alarm window in runtime.

Note

If you have different alarm classes output, these will be initially sorted into alarm classes in runtime, and then by when the alarm occurred.

Operator controls

The control elements that can be used to control the alarm view in runtime are specified in the Inspector window under "Properties >Display > Settings". The following table shows the control elements in the Alarm window, and what they do:

Button		Function
"Tooltip"		Displays a tooltip for an alarm.
"Acknowledge"		Acknowledges an alarm.
"Loop-In-Alarm"		If a screen change is configured, the display switches to a screen containing information about the alarm.

Access protection in Runtime

Configure access protection under "Properties > Properties > Security" in the Inspector window of the alarm view. If a logged-on user has the required authorization, he can acknowledge and edit alarms using the operator controls in the alarm view. If the logged-in user does not have

the required authorization, or if no user is logged in, clicking the "Acknowledge" or "Edit" buttons or double-clicking an alarm line opens the login dialog box.

Note

Basic Panels

Access protection is not available for Basic Panels.

Activating the focus of the Alarm window

Select the following option so that the Alarm window does not lose the focus after a screen change:

1. In the Inspector window, select "Properties > Properties > Mode".
2. Enable "Label".

Window

Define the response of the Alarm window under "Properties > Properties > Mode > Window" in the Inspector window. The following table shows the possible properties:

Option	Function
Automatic display	The Alarm window is automatically displayed when a system alarm occurs, for example.
Closable	The window closes again after a set time has elapsed. You define the display duration in the alarm settings.
Modal	The Alarm window is linked to a confirmation, such as: Alarm must be acknowledged. If the modal alarm window has the focus, the buttons in the screen behind it cannot be used. The functions configured on a function key are carried out.
Sizeable	You can change the size of the Alarm window in runtime.

See also

Device-Specific Nature of the Objects (Page 8075)

Alarm view (Page 8090)

Configuring an alarm window (Page 8174)

Alarm indicator

Application

The alarm indicator is a graphic symbol that shows current errors or errors that need to be acknowledged, depending on the configuration. The alarm indicator is configured in the "Global screen" editor. The following figure shows an alarm indicator:

Alarm indicator OP 73

A "simple" alarm indicator is available for the HMI OP 73. The following diagram shows the alarm indicator for the OP 73 HMI devices:

The "simple" alarm indicator shows alarms to be acknowledged or alarms which have already been acknowledged and have not yet gone. Only the position can be defined for the "simple" alarm indicator. The alarm indicator is displayed on the device at the selected position. If you have configured a screen object at this position, the visible alarm indicator covers the screen object. The alarm indicator is covered by system dialogs, such as the login dialog, Help dialog, and alarm windows.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Alarm classes: Establishes the alarm classes where the alarm indicator is displayed.
- Operator control in Runtime: Defines the operator actions in Runtime that cause the Alarm window to open.

Alarm classes

You define which alarm classes are shown with an alarm indicator in "General > Alarm classes" in the Inspector window. Alarm classes, such as "Warnings" or "Errors".

Define operator control in Runtime

1. Select the alarm indicator in the screen.
2. Click "Events > Click" or "Click when flashing" in the Inspector window.
3. The "function list" opens. Click the first line of the function list. The list of system functions, and scripts available in the project opens.

4. Select the "ShowAlarmWindow". system function under "Alarms."
5. Under "object name" select the name of the Alarm window from the selection list. Under "Layout", define whether the Alarm window should be visible, hidden, or should toggle between the two states.

See also

Device-Specific Nature of the Objects (Page 8075)

Alarm view (Page 8090)

Configuring an alarm indicator (Page 8175)

Rectangle

Application

The "Rectangle" is a closed object which you can fill with a color.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Corner radius: Specifies the horizontal and vertical distance between the corner of the rectangle and the start point of a rounded corner.

Corner radius

The corners of the "Rectangle" object can be rounded to suit your requirements. When the properties "X" and "Y" are set to the 100 % value, the rectangle is displayed as an ellipse. As soon as one of the properties has the value 0%, a normal rectangle without a rounded corner is shown.

1. Click "Properties > Properties > Layout" in the inspector window.
2. Enter a value for "X" in the "Corner radius" area.
   The input value is the percentage proportion of half the width of the rectangle.
3. Enter a value for "Y" in the "Corner radius" area.
   The input value is the percentage proportion of half the height of the rectangle.

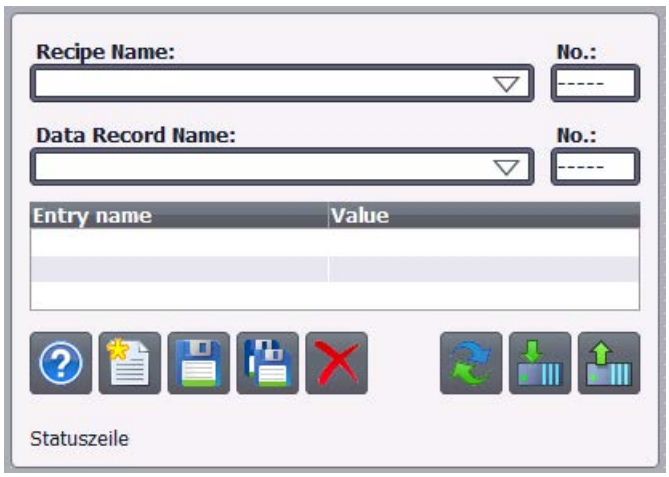
See also

Device-Specific Nature of the Objects (Page 8075)
Rotating objects (Page 7993)
Flipping objects (Page 7995)

Recipe view

Application

The "Recipe view" object is used to display and modify recipes.



Layout

In the Inspector window, you customize the position, geometry, style, color and font types of the object. You can adapt the following properties in particular:

- Toolbar: Specifies the menu commands of the recipe view.

Toolbar

The menu items with which the recipe view is operated in Runtime are configured under "Properties > Properties > Toolbar" in the Inspector window.

Menu command	Description
"Tooltip"	Calls up the configured tooltip for the selected recipe.
"New record"	Creates a new recipe record in the recipe.
"Delete record"	Deletes the selected record.
"Saving"	Saves the modified record with its current name.
"Save as"	Saves the modified record with a new name.
"Write to PLC"	Sends the current value to the PLC.
"Read from PLC"	Reads the current value from the PLC.

Menu command	Description
"Rename"	Specifies that the "Rename" button is shown.
"Forward"	Specifies that the menu buttons are visible.
"Back"	Specifies that the "Back" button is shown.

See also

Device-Specific Nature of the Objects (Page 8075)

Simple recipe view (Page 8219)

Displaying recipes (Page 8214)

Configuring the simple recipe view (Page 8230)

Description of the simple recipe view (Page 8233)

Switch

Application

The "Switch" object is used to configure a switch that is used to switch between two predefined states in runtime. The current state of the "Switch" object can be visualized with either a label or a graphic.

The following figure shows a "Switch" type switch.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. In particular, you can customize the following property:

- Type: Defines the graphic representation of the object.

Type

Button visualization is specified at "Properties > Properties > General >Type" in the Inspector window.

Type	Description
"Switch"	The two states of the "Switch" are displayed in the form of a switch. The position of the switch indicates the current state. The state is changed in runtime by sliding the switch. You specify the direction of movement of the switch in "Switch orientation" with this type.
"Switch with text"	The switch is shown as a button. The current state is visualized with a label. In runtime click on the button to actuate the switch.
"Switch with graphic"	The switch is shown as a button. The current state is visualized with a graphic. In runtime click on the button to actuate the switch.

Note

Basic Panels

The "Switch" type is not available for Basic Panels.

See also

Device-Specific Nature of the Objects (Page 8075)

Overview of objects (Page 7978)

Button

Application

The "Button" object allows you to configure an object that the operator can use in runtime to execute any configurable function.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Mode: Defines the graphic representation of the object.
- Text / Graphic: Defines whether the Graphic view is static or dynamic.
- Define hotkey: Defines a key, or shortcut that the operator can use to actuate the button.

Note

You can only define a hotkey for HMI devices with keys.

Mode

The button display is defined in "Properties > Properties > General >Mode" in the Inspector window.

Mode	Description
"Invisible"	The button is not visible in runtime.
"Text"	The button is displayed with text. This text explains the function of the button.
"Graphic"	The button is displayed with a graphic. This graphics represents the function of the button.

Depending on the device, further options are available.

Text / Graphic

The "Mode" property settings are used to define whether the display is static or dynamic. The display is defined in "Properties > Properties > General >Text" or "Graphic" in the Inspector window.

You can, for example, select the following options for the "Graphic" or "Text" type.

Type	Option	Description
"Graphic"	"Graphic"	Use "Graphic when button is "not pressed"" to specify a graphic displayed in the button for the "OFF" state. Use "Graphic when button is "pressed"" to specify a graphic for the "ON" state.
	"Graphics list"	The graphic in the button depends on the state. The corresponding entry from the graphics list is displayed depending on the state.
"Text"	"Text"	Use "Text when button is "not pressed"" to specify the text displayed in the button for the "OFF" state. Use "Text when button is "pressed"" to specify text for the "ON" state.
	"Text list"	The text in the button depends on the state. The entry from the text list corresponding to the state is displayed.

Define hotkey

In the Inspector window, a key or key combination is defined that the operator can use to control the button in runtime.

1. In the Inspector window, select "Properties > Properties > General".
2. Select a key or key combination from the selection list in the "Hotkey" area.

See also

Device-Specific Nature of the Objects (Page 8075)

Example: Configuring a button for language switching (Page 8055)

Example: Configuring a button with logon dialog box (Page 8265)

Example: Configuring a button with access protection (Page 8269)

Symbolic I/O field

Application

The "Symbolic I/O field" object can be used to configure a selection list for input and output of texts in runtime.

Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Mode: Specifies the response of the object in runtime.
- Text list: Specifies the text list that is linked to the object.
- Button for selection list: Specifies that the object has a button to open the selection list.

Note

Reports

In reports, symbolic I/O fields only output data. "Output" mode is preset. Properties for configuring the selection of graphics are not available, e.g. "button for selection list".

Mode

The response of the symbolic I/O field is specified in the Inspector window in "Properties > Properties > General > Type".

Mode	Description
"Output"	The symbolic I/O field is used to output values.
"Input"	The symbolic I/O field is used to input values.
"Input/output"	The symbolic I/O field is used for the input and output of values.
"Two states"	The symbolic I/O field is used only to output values and has a maximum of two states. The field switches between two predefined texts. This is used, for example, to visualize the two states of a valve: closed or open.

Note

The behavior possible for the symbolic I/O field depends on the Runtime.

Text list

In the Inspector window, you specify which text list is linked to the symbolic I/O field.

1. In the Inspector window, select "Properties > Properties > General".
2. Under "Contents" open the selection list for "Text list".
3. Select a text list.

Button for selection list

The "Button for selection list" property is used to display a button for opening the selection list.

1. Select "Properties > Properties > Layout" in the Inspector window.
2. Activate "Response > Button for selection list".

Note
Basic Panels

The "Button for selection list" option is not available for Basic Panels.

See also

Device-Specific Nature of the Objects (Page 8075)

Graphic I/O field (Page 8084)

System diagnostics view

Introduction

The system diagnostics view offers you an overview of all the available devices in your plant. You navigate directly to the cause of the error and to the relevant device. You have access to all diagnostics-capable devices which you have configured in the "Devices & networks" editor.

Use

The system diagnostics view enables you to achieve the maximum level of detail of the diagnostics data. A precise diagnosis is possible, as all the available data is displayed. You have the system status of the entire plant at one glance.

Three different views are available in the system diagnostics view.

- Device view
- Diagnostic buffer view
- Detail view

Device view

The device view of the system diagnostics view shows all the available connections in tabular form. Double-clicking a connection opens the detail view. The device view is only displayed if more than one connection has been created in the "Devices & Networks" editor.

Diagnostic buffer view

In the diagnostic buffer view, the current data from the diagnostic buffer is displayed.

Detail view

The detail view shows detailed information on the selected connection. You cannot sort error texts in the detail view. The detail view is only available if there is an integrated connection to an S7 1200 or S7 1500.

Layout

You change the settings for the position, geometry, style, color, and font of the object in the Inspector window. You can adapt the following properties in particular:

- Lines per entry: Specifies the number of lines that are shown for an entry.

Configuring the system diagnostics view

1. Drag-and-drop the system diagnostics view from the toolbox.
2. In the Inspector window, select "Properties > Layout".
3. Enter a number under "Lines per entry", i.e. 5.
4. Select an authorization for operation in "Properties > Properties > Security".

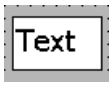
See also

System diagnostics views (Page 8208)

Text field

Application

The "Text field" is a closed object which you can fill with a color.



Layout

In the Inspector window, you customize the position, shape, style, color and font types of the object. You can adapt the following properties in particular:

- Text: Specifies the text for the text field.
- Size of text field: Defines whether the size of the object is adapted to the space required by the largest list entry.

Text

Specify the text for the text field in the Inspector window.

1. In the Inspector window, select "Properties > Properties > General".
2. Enter a text.
   For texts over several lines you can set a line break by pressing the key combination <Shift + Enter>.

Size of text field

In the Inspector window, you can define whether the size of the object is adapted to the space required by the largest list entry.

1. In the Inspector window, select "Properties > Properties > Layout".
2. Activate "Resize > Fit to contents".

See also

Device-Specific Nature of the Objects (Page 8075)

10.1.8 Configuring screen navigation

10.1.8.1 Basics for screen navigation

Types of navigation for the screen change

For a production process consisting of multiple subprocesses, you will configure multiple screens. You have the following options to enable the operator to switch from one screen to the next in Runtime:

- Assign buttons to screen changes
- Configuring screen changes at local function keys

Procedure

Before you create a screen change, define the plant structure and derive from it the screen changes that you want to configure.

Create the start screen under "Runtime Settings > General > Start screen".

See also

Assign button with screen change (Page 8106)

10.1.8.2 Assign button with screen change

Introduction

Configure a button in the screen to switch between the screens on the HMI device during operation.

Note

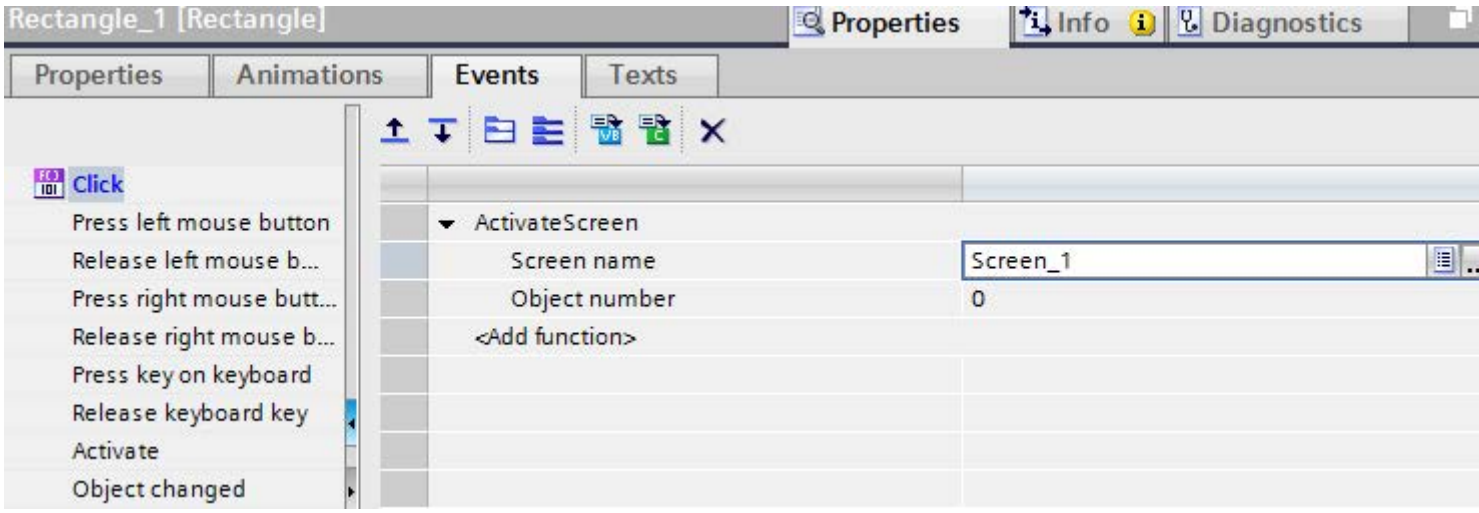
If you have set the "Visibility" of animations to "Hidden" in the Inspector window of a screen, this screen cannot be called up in Runtime.

Requirements

- You have created the project.
- You have created the "Screen_2" screen.
- "Screen_1" is created.

Procedure

1. Double-click "Screen_1" in the project navigation. The screen is displayed in the work area.
2. Move "Screen_2" from the project tree to the open screen by drag&drop.
   A button with the name "Screen_1" is inserted.
3. In the Inspector window, select "Properties > Events > Click".
   The "ActivateScreen" system function is displayed in the "Function list".



4. At the "Object number" attribute, define, if required, the tab sequence number of the object on which the focus is to be set after a screen change. You can also specify a tag that contains the object number.

Alternative procedure

1. Move a button from the "Tools" task card to "Screen2" by drag&drop.
2. In the Inspector window, select "Properties > Events > Click".
3. Select the "ActivateScreen" system function.
4. Select "Screen_2" for the "Screen number".

Result

The operator goes to "Screen_1 with the button in Runtime. If you have specified an object number, the object with this object number has the focus following a screen change.

See also

Basics for screen navigation (Page 8105)

10.1.8.3 Assign screen change to function key

Introduction

Configure a screen change function key in the screen to switch between the screens on the HMI device during operation.

Note

If you have set the "Visibility" of animations to "Hidden" in the inspector window of a screen, this screen cannot be called up in Runtime.

Requirements

- You have created a project.
- You have created the "Screen_2" screen.
- You have created the "Screen_1" screen.

Procedure

1. Double click "Screen_1" in the project tree. The screen is displayed in the work area.
2. Move "Screen_2" from the project tree to a function key, e.g. "F2".
   The configured function key displays a yellow triangle.
3. Click "Properties > Events > Press key" in the inspector window.
   The "ActivateScreen" system function is displayed.

Result

The operator goes to the specified "Screen_2" with function key "F2" in Runtime.

10.2 Working with Tags

10.2.1 Basics

10.2.1.1 Basics of tags

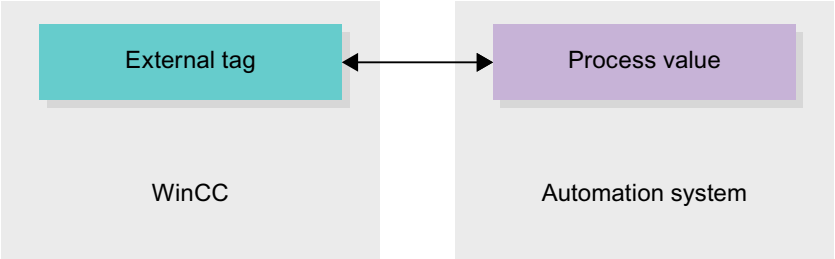
Introduction

Process values are forwarded in runtime using tags. Process values are data which is stored in the memory of one of the connected automation systems. They represent the status of a plant in the form of temperatures, fill levels or switching states, for example. Define external tags for processing the process values in WinCC.

WinCC works with two types of tag:

- External tags
- Internal tags

The external tags form the link between WinCC and the automation systems. The values of external tags correspond to the process values from the memory of an automation system. The value of an external tag is determined by reading the process value from the memory of the automation system. It is also possible to rewrite a process value in the memory of the automation system.



Internal tags do not have a process link and only convey values within the WinCC. Tag values are only available while Runtime is running.

Tags in WinCC

For external tags, the properties of the tag are used to define the connection that the WinCC uses to communicate with the automation system and form of data exchange.

Tags that are not supplied with values by the process - the internal tags - are not connected to the automation system. In the tag's "Connection" property, this is identified by the "Internal tag" entry.

You can create tags in different tag tables for greater clarity. You then directly access the individual tag tables in the "HMI tags" node in the project tree. The tags from all tag tables can be displayed with the help of the table "Show all tags".

See also

Overview of HMI tag tables (Page 8109)

Internal tags (Page 8114)

External tags (Page 8110)

Addressing external tags (Page 8112)

Creating external tags (Page 8115)

Basics on arrays (Page 8142)

Cycle basics (Page 8145)

10.2.1.2 Overview of HMI tag tables

Introduction

HMI tag tables contain the definitions of the HMI tags that apply across all devices. A tag table is created automatically for each HMI device created in the project.

In the project tree there is an "HMI tags" folder for each HMI device. The following tables can be contained in this folder:

- Standard tag table
- User-defined tag tables
- All tags

The following tables are also available in an HMI tag table:

- Discrete alarms
- Analog alarms

With the help of these tables you configure alarms for the currently selected HMI tag.

In the project tree you can create additional tag tables in the HMI tags folder and use these to sort and group tags and constants. You can move tags to a different tag table using a drag-and-drop operation or with the help of the "Tag table" field. You activate the "Tag table" field using the shortcut menu of the column headings.

Standard tag table

There is one standard tag table for each HMI device of the project. It cannot be deleted, renamed or moved. The standard tag table contains HMI tags and, depending on the HMI device, also system tags. You can declare all HMI tags in the standard tag table, or create additional user-defined tag tables as you want.

User-defined tag tables

You can create multiple user-defined tag tables for each HMI device in order to group tags according to your requirements. You can rename, gather into groups, or delete user-defined tag tables. To group tag tables, create additional subfolders in the HMI tags folder.

All tags

The "All tags" table shows an overview of all HMI tags and system tags of the HMI device in question. This table cannot be deleted, renamed or moved.

Discrete alarms table

In the "Discrete alarms" table, you configure discrete alarms to the HMI tag selected in the HMI tag table. When you configure a discrete alarm, multiple selection in the HMI tag table is not possible. You configure the discrete alarms for each HMI tag separately.

Analog alarms table

In the "Analog alarms" table, you configure analog alarms to the HMI tag selected in the HMI tag table. When you configure an analog alarm, multiple selection in the HMI tag table is not possible. You configure the analog alarms for each HMI tag separately.

See also

Basics of tags (Page 8108)

10.2.1.3 External tags

Introduction

External tags allow communication (data exchange) between the components of an automation system, for example, between an HMI device and a PLC.

Principle

An external tag is the image of a defined memory location in the PLC. You have read and write access to this storage location from both the HMI device and from the PLC.

Since external tags are the image of a storage location in the PLC, the applicable data types depend on the PLC which is connected to the HMI device.

If you write a PLC control program in STEP 7, the PLC tags created in the control program will be added to the PLC tag table. If you want to connect an external tag to a PLC tag, access the PLC tags directly via the PLC tag table and connect them to the external tag.

Data types

All the data types which are available at the connected PLC are available at an external tag in WinCC. Information about data types which are available for connection to other PLCs can be found in the documentation about the respective communication drivers.

See "Basics of communication (Page 8373)" for additional information.

Note

As well as external tags, area pointers are also available for communication between the HMI device and PLC. You can set up and enable the area indicators in the "Connections" editor.

Synchronization with PLC tags

A variety of options for synchronizing external tags with the PLC tags are available in the Runtime settings under "Settings for tags".

When you perform synchronization, you have the option of automatically applying the tag names of the PLC to external tags and reconnecting the existing tags.

The generated tag name is derived from the position of the data value in the hierarchical structure of the data block.

Update of tag values

For external tags, the current tag values are transmitted in runtime via the communication connection between WinCC and the connected automation systems and then saved in the runtime memory. Next, the tag value will be updated to the set cycle time. For use in the runtime project, WinCC accesses tag values in the runtime memory that were read from the PLC at the previous cycle time. As a result, the value in the PLC can already change whilst the value from the runtime memory is being processed.

Note

PLC array elements in conjunction with S7-1200 or S7-1500

The index of the PLC array elements can begin with any number. In WinCC, indexing always starts with 0.

A PLC tag "Array [1..3] of Int", for example, is mapped to "Array [0..2] of Int" in WinCC.

When you access an array in a script, pay attention to the correct indexing.

See also

Addressing external tags (Page 8112)

Basics of communication (Page 8373)

Basics of tags (Page 8108)

10.2.1.4 Addressing external tags

Introduction

The options for addressing external tags depend on the type of connection between WinCC and the PLC in question. A distinction must be made between the following connection types:

- Integrated connection
  Connections of devices which are within a project and were created with the "Devices & Networks" editor are referred to as integrated connections.
- Non-integrated connection
  Connections of devices which were created with the "Connections" editor are referred to as non-integrated connections. It is not necessary that all of the devices be within a single project.

The connection type can also be recognized by its icon.

	Integrated connection
	Non-integrated connection

You can find additional information in the section "Basics of communication (Page 8373)".

Addressing with integrated connections

An integrated connection offers the advantage that you can address a tag both symbolically and absolutely.

For symbolic addressing, you select the PLC tag via its name and connect it to the HMI tag. The valid data type for the HMI tag is automatically selected by the system. You have to distinguish between the following cases when you address elements in data blocks:

Symbolic addressing of data blocks with optimized access and standard access:

During the symbolic addressing of a data block with optimized access and standard access, the address of an element in the data block is dynamically assigned and is automatically adopted in the HMI tag in the event of a change. You do not need to compile the connected data block or the WinCC project for this step.
For data blocks with optimized access, only symbolic addressing is available.

For symbolic addressing of elements in a data block, you only need to recompile and reload the WinCC project in case of the following changes:

- If the name or the data type of the linked data block element or global PLC tag has changed.
- If the name or the data type of the higher level structure node of a linked element in the data block element or global PLC tag has changed.
- If the name of the connected data block has changed.

Symbolic addressing is currently available with the following PLCs:

- SIMATIC S7 1200
- SIMATIC S7 1500

Symbolic addressing is also available if you have an integrated link.

Note

Symbolic addressing is not possible if you are using the "Date/Time" area pointer.

You can also use absolute addressing with an integrated connection. You have to use absolute addressing for PLC tags from a SIMATIC S7 300/400 PLC. If you have connected an HMI tag with a PLC tag and the address of the PLC tag changes, you only have to recompile the control program to update the new address in WinCC. Then you recompile the WinCC project and load it onto the HMI device.

In WinCC, symbolic addressing is the default method. To change the default setting, select the menu command "Options > Settings". Select "Visualization > Tags" in the "Settings" dialog. If required, disable the "Symbolic access" option.

The availability of an integrated connection depends on the PLC used. The following table shows the availability:

PLC	Integrated connection	Comments
S7 300/400	Yes	The linking of tags is not checked in Runtime. If the tag address changes in the PLC and the HMI device is not compiled again and loaded, the change is not registered in runtime.
S7 1200	Yes	A validity check of the tag connection is performed in runtime during symbolic addressing. If an address is changed in the PLC, the change is registered and an error message is issued. In the case of absolute addressing, the following behavior applies to the S7 300/400.
S7 1500	Yes	A validity check of the tag connection is performed in runtime during symbolic addressing. If an address is changed in the PLC, the change is registered and an error message is issued. In the case of absolute addressing, the following behavior applies to the S7 300/400.

Create an integrated connection in the "Devices & Networks" editor. If the PLC is contained in the project and integrated connections are supported, you can then also have the connection created automatically. To do this, when configuring the HMI tag, simply select an existing PLC tag to which you want to connect the HMI tag. The integrated connection is then automatically created by the system.

Addressing with non-integrated connections

In the case of a project with a non-integrated connection, you always configure a tag connection with absolute addressing. Select the valid data type yourself. If the address of a PLC tag changes in a project with a non-integrated connection during the course of the project, you also have to make the change in WinCC. The tag connection cannot be checked for validity in Runtime, an error message is not issued.

A non-integrated connection is available for all supported PLCs.

Symbolic addressing is not available in a non-integrated connection.

With a non-integrated connection, the control program does not need to be part of the WinCC project. You can perform the configuration of the PLC and the WinCC project independently of each other. For configuration in WinCC, only the addresses used in the PLC and their function have to be known.

See also

External tags (Page 8110)

Basics of tags (Page 8108)

Basics of communication (Page 8373)

10.2.1.5 Internal tags

Introduction

Internal tags do not have any connection to the PLC. Internal tags transport values within the HMI device. Tag values are only available while Runtime is running.

Principle

Internal tags are stored in the memory of the HMI device. Therefore, only this HMI device has read and write access to the internal tags. You can create internal tags to perform local calculations, for example.

You can use the HMI data types for internal tags.

The following HMI data types are available:

HMI data type	Data format
Array	One-dimensional array
Bool	Binary tag
DateTime	Date/time format
DInt	Signed 32-bit value
Int	Signed 16-bit value
LReal	Floating-point number 64-bit IEEE 754
Real	Floating-point number 32-bit IEEE 754
SInt	Signed 8-bit value
UDInt	Unsigned 32-bit value
UInt	Unsigned 16-bit value
USInt	Unsigned 8-bit value
WString	Text tag, 16-bit character set

See also

Basics of tags (Page 8108)

10.2.2 Working with tags

10.2.2.1 Creating tags

Creating external tags

Introduction

You can access an address in the PLC via a PLC tag using an external tag. The following options are available for addressing:

- Symbolic addressing
- Absolute Addressing

You can find further details on symbolic addressing in section "Addressing external tags (Page 8112)". If possible, use symbolic addressing when configuring a tag. You create tags either in the standard tag table or in a tag table you defined yourself.

Requirement

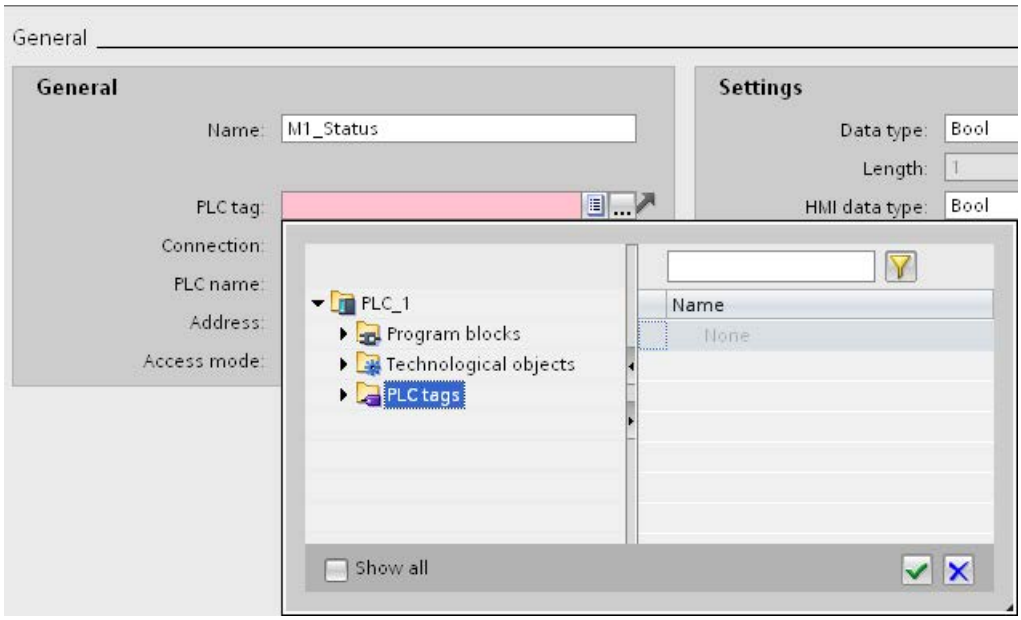
- You have opened the project.
- A connection to the PLC is configured.
- The Inspector window is open.

Procedure

To create an external tag, proceed as follows:

1. Open the "HMI tags" folder in the project tree and double-click the standard tag table. The tag table open.
   Alternatively, create a new tag table and then open it.
2. In the "Name" column, double-click "Add" in the tag table. A new tag is created.
3. Select the "Properties > Properties >General" category in the Inspector window and, if required, enter a unique tag name in the "Name" field. The tag name must be unique throughout the device.
4. If required, select the "Display name" field to enter a name to be displayed in Runtime. The name to be displayed is language-specific and can be translated for the required Runtime languages. The display name is available for Basic Panels, Panels and Runtime Advanced.

5. Select the connection to the required PLC in the "Connection" box. If the connection you require is not displayed, you must first create the connection to the PLC. You create the connection to a SIMATIC S7 PLC in the "Devices & Networks" editor. You create the connection to external PLCs in the "Connections" editor.
   If the project contains the PLC and supports integrated connections, you can also have the connection created automatically. To do this, when configuring the HMI tag, simply select an existing PLC tag to which you want to connect the HMI tag. The integrated connection is then automatically created by the system.
6. If you are working with an integrated connection, click the ... button in the "PLC tag" field and select an already created PLC tag in the object list. Click the ✓ button to confirm the selection.



7. If you are working with a non-integrated connection, enter the address from the PLC in the "Address" field. The "PLC tag" field remains empty.
8. Configure the other properties of the tag in the inspector window.

You can also configure all tag properties directly in the tag table. To view hidden columns, activate the column titles using the shortcut menu.

You can also create new tags alternatively directly at the application point, e.g. on an I/O field. To do this, click the <Add new> button in the object list. You then configure the new tag in the Inspector window.

Result

An external tag has been created and linked to a PLC tag or an address in the PLC.

Alternative procedure

You can also create external HMI tags by dragging and dropping data block elements or global PLC tags in an HMI tag table.

See also

Creating internal tags (Page 8117)

Creating multiple tags (Page 8118)

Editing a Tag (Page 8119)

Tag limits (Page 8125)

Basics of tags (Page 8108)

Addressing external tags (Page 8112)

Creating internal tags

Introduction

You must at least set the name and data type for internal tags. Select the "Internal tag" item, rather than a connection to a PLC.

For documentation purposes, it is a good idea to enter a comment for every tag.

Requirement

You have opened the project.

Procedure

1. Open the "HMI tags" folder in the project tree and double-click the entry "Standard tag table". The tag table opens.
   Alternatively, create and then open a new tag table.
2. Double-click "Add" in the "Name" column of the tag table. A new tag is created.
3. If the Inspector window is not open, select the "Inspector window" option in the "View" menu.
4. Select the "Properties > Properties >General" category in the Inspector window and, if required, enter a unique tag name in the "Name" field. This tag name must be unique throughout the project.
5. If required, select the "Display name" field to enter a name to be displayed in Runtime. The name to be displayed is language-specific and can be translated for the required Runtime languages. The display name is available for Basic Panels, Panels and Runtime Advanced.
6. Select "Internal tag" as the connection in the "Connection" field.
7. Select the required data type in the "Data type" field.
8. In the "Length" field, you must specify the maximum number of characters to be stored in the tag according to the selected data type. The length is automatically defined by the data type for numerical tags.
9. As an option, you can enter a comment regarding the use of the tag. To do so, click "Properties > Properties > Comment" in the Inspector window and enter a text.

You can also configure all tag properties directly in the tag table. To view hidden columns, activate the column titles using the shortcut menu.

You can also create new tags alternatively directly at the application point, e.g. on an I/O field. Click the <Add new> button in the object list. You can then configure the new tag in the Properties window that opens.

Result

An internal tag is created. You can now use this in your project.

In additional steps you can configure the tag, for example, by setting the start value and limits.

See also

Creating external tags (Page 8115)

Creating multiple tags

Introduction

In a tag table, you create additional identical tags by automatically filling the rows of the table below a tag.

The tag names are incremented automatically when filling in automatically.

You can also use the auto fill function to fill table cells below a tag with a single tag property and thus modify the corresponding tags.

If you apply the automatic filling in to already filled cells of a tab table, you will be asked whether you want to overwrite the cells or insert new tags.

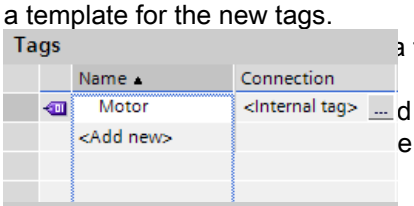
If you do not want to overwrite already configured tags, activate insert mode. Activate insert mode by keeping the <Ctrl> key pressed during insertion. Already existing entries in the tag table are moved down if insert mode is activated.

Requirement

- You have opened the project.
- A tag table is open.
- The tag which is to serve as a template for other tags is configured.

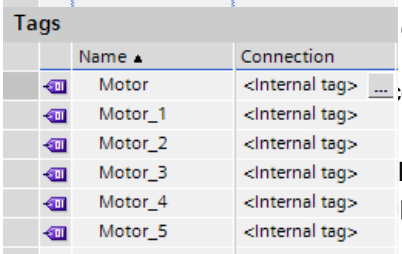
Procedure

1. If you want to create new tags, mark in the "Name" column the tag that should be used as a template for the new tags.



a tag to the tags below it, select the cell which contains d in color and a small blue square will appear in its bottom e over this square, the cursor will change to a black cross.

2. drag this square over the cells below that you wish to fill over this area.



3. ll of the marked cells will be filled automatically. pty cells in the marked area.

Result

Depending on which cells were selected, the function may automatically fill individual properties or create new tags.

See also

Creating external tags (Page 8115)

10.2.2.2 Editing tags

Editing a Tag

Introduction

You can always rename, copy or delete tags.

When a tag is renamed, the new name must be unique for the whole device.

If you use the "Copy" command to copy a tag to the clipboard, the objects and references linked to the tag are copied as well.

If you use the "Insert" command to add a tag to another device, the tag will be added without the connected references. Only the object name of the reference will be inserted. If a reference of the same name and valid properties exists in the target system, the existing reference will then be connected to the copied tag.

If you copy a tag, the alarms linked to the tag are copied as well. If you add the copied tag to another device, the tag is added together with the linked alarms.

Requirement

- The tag which you wish to rename, copy or delete must exist.
- The tag table is open.

Renaming tags

1. In the "Name" field, select the tag in the tag table.
2. Select "Rename" from the shortcut menu.
3. Type in a new name.
   The tag appears under its new name.

Copying tags

1. Select one or more tags in the tag table or in the Detail window.
2. Select "Copy" from the shortcut menu.
3. Click on the point at which you want to insert the tag. For example, click another tag table in the same device or the tag table in a second device.
4. Select the "Insert" or "Extended insert" command from the shortcut menu. The tag is inserted as described above.

Deleting a tag

1. Select one or more tags in the tag table.
2. Select the "Cross-reference" command from the "Tools" menu. In the "Cross-reference" editor, check to see where the tags are used. In this manner, you can see what impact the deletion of the tag will have on your project.
3. Select "Delete" in the pop-up menu of the tag.
   All marked tags will be deleted.

Export and import of tags

WinCC gives you the option to export and import tags. With Export and Import, you have the option to export tags from one project and import them into another project. There is also the option to create larger numbers of tags outside of WinCC, edit them and subsequently import into any WinCC project. See Importing and exporting tags (Page 8807) for additional information.

See also

Changing the tag configuration (Page 8121)

Configuring multiple tags simultaneously (Page 8121)

Using multiple tags simultaneously in a screen (Page 8122)

Synchronizing tags (Page 8124)

Creating external tags (Page 8115)

Importing and exporting tags (Page 8807)

Changing the tag configuration

Introduction

You can modify tags at any time to adapt them to changed requirements in the project.

Changing the tag configuration

if you want to change the configuration of a tag, open the tag table in which the tag is contained. Open the "Show all tags" tag table alternatively.

In the tag tables, you can perform such tasks as comparing and adjusting the properties of multiple tags or sorting the tags by their properties.

Change the properties either directly in the table or in the inspector window.

If you change a tag property and this change causes a conflict with another property, it will be highlighted in color to draw your attention to this fact. This could happen, for example, if you connect the tag to another PLC which does not support this data type.

See also

Editing a Tag (Page 8119)

Configuring multiple tags simultaneously

Introduction

In WinCC, you can assign the same properties to multiple tags in a single operation. This facilitates efficient programming.

Requirement

- You created the tags you want to configure.
- The tag table is open.
- The Inspector window is open.

Procedure

1. In the tag table, select all the tags that you want to configure at the same time.
   If the selected property is identical for all the tags, the setting for this property will appear in the Inspector window. The associated field will remain blank otherwise.
2. You can define the shared properties in the Inspector window or directly in the tag table.

if you change a property commonly on several tags, only this one property is changed. The other properties of the tag remain unchanged.

Result

All marked tags will be reconfigured.

To edit tag properties which differ from one tag to the other, simply clear the multiple selection.

See also

Editing a Tag (Page 8119)

Using multiple tags simultaneously in a screen

Introduction

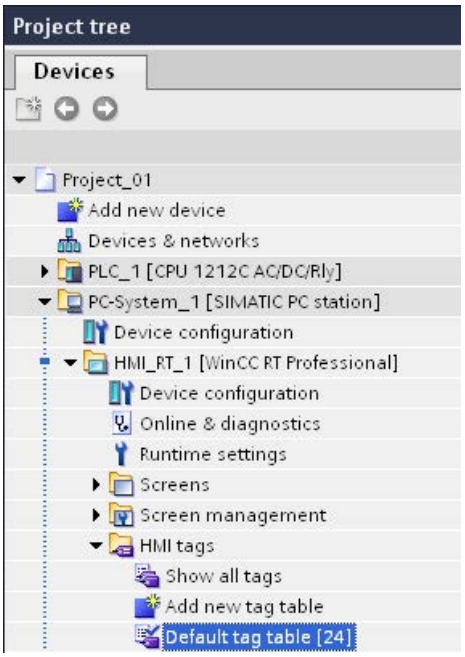
In WinCC, you can create multiple I/O fields that are linked with tags in one screen in a single operation. This facilitates efficient programming.

Requirement

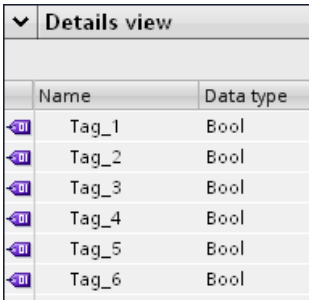
- Several tags are set up.
- A screen is open.

Procedure

1. In the project tree, select the required tag table under "HMI tags".



2. Select the detail view at the bottom of the project tree. The detail view shows the tags that exist in the selected tag group.



3. Mark the tags in the detail window.
4. Drag the tags to the screen. For each tag, this creates an I/O field that is connected to the tag.

Note

When you move a PLC tag from the detail window to the work area by drag&drop, a network and a connection are created additionally in the "Devices & Networks" editor.

See also

Editing a Tag (Page 8119)

Synchronizing tags

Introduction

To synchronize the PLC and HMI tags, WinCC offers the following options:

- Synchronizing tags with or without name matching between PLC and WinCC
  The following options are available for this:
- Link tags with addresses in the PLC
  This procedure is suitable, for example, if changes were made to the connection between the HMI device and the PLC and the tag connections were lost. The function can also be used if you have configured the control program and HMI project separately.

Requirement

- External HMI tags have been created.
- PLC tags have been created.
- An HMI connection to a PLC in the project has been established.

Procedure

To synchronize HMI tags with PLC tags, follow these steps:

1. In the project tree, select the directory that contains the tags in question.
2. Select "Connect HMI tags to matching PLC tags" from the shortcut menu.
   A dialog opens.

3. Select the option you want to use.
   If you want to synchronize the tags without name matching, disable "Replace WinCC tag name with PLC tag name".
   If you want to connect HMI tags new with absolute access, select "Data type and absolute address match".
4. Confirm with "Synchronize".
   The system searches for a suitable PLC tag according to the selected option.

Result

The external HMI tags are synchronized with the PLC tags.

If you have selected the option "Data type and absolute address match", the tag connection is established as soon as a suitable PLC tag is found.

If you have selected a different option, the WinCC tags are updated accordingly and the PLC tag names are applied in WinCC.

See also

Editing a Tag (Page 8119)

10.2.2.3 Configuring tags

Tag limits

Introduction

You can restrict the value range with limits for numerical tags.

Principle

You can specify a value range defined by a high and low limit for numerical tags.

If the process value violates the value range, you trigger a function list. If the operator enters a value for the tag that is outside the configured value range, the input is rejected. The value is not accepted.

Note

You enter the text of the analog alarms for limit violations in the "Analog alarms" editor.

Application example

Use the limit to warn the operator in good time when the value of a tag enters a critical range, for example.

See also

Applying linear scaling to a tag (Page 8131)

Defining the start value of a tag (Page 8128)

Defining the acquisition cycle for a tag (Page 8135)

Defining Limits for a Tag (Page 8126)

Start value of a tag (Page 8127)

Updating the tag value in Runtime (Page 8128)
Linear scaling of a tag (Page 8129)
Connecting a tag to another PLC (Page 8131)
Indirect addressing of tags (Page 8132)
Addressing tags indirectly (Page 8133)
Using tags to trigger functions (Page 8134)
Address multiplexing (Page 8136)
Configuring address multiplexing with absolute addressing (Page 8137)
Configuring address multiplexing with symbolic addressing (Page 8139)
Creating external tags (Page 8115)

Defining Limits for a Tag

Introduction

For numerical tags, you can specify a value range by defining a low and high limit.

Additionally, you configure the system to process a function list whenever a tag value drops below or exceeds its configured value range.

Requirement

- You created the tag for which you want to set limits.
- The Inspector window with the properties for this tag is open.

Procedure

To define limits of a tag, follow these steps:

1. In the Inspector window select "Properties > Properties > Limits." If you want to define one of the limits as a constant value, select "Constant" using the button. Enter a number in the relevant field.
   If you want to define one of the limits as a tag value, select "HMI tag" using the button. Use the object list to define the tag for the limit.
2. To set an additional limit value for the tag, repeat step 1 with the appropriate settings.

Alternative procedure

You can also configure the high and low limit directly in the tag table. To view hidden columns, activate the column titles using the shortcut menu.

Configuring a function list

You can configure a function list for exceeding the value range as follows:

1. If you want to start a function list when the value drops below the value range, click "Properties > Events > Minimum violated" in the Inspector window. Create a function list in this dialog.
2. If you want to start a function list when the value exceeds the value range, click "Properties > Events > Maximum violated" in the Inspector window. Create a function list in this dialog.

Result

You have set a value range defined by a high and low limit for the selected tag. If the value range is exceeded or undershot, a function list is carried out.

See also

Tag limits (Page 8125)

Start value of a tag

Value of a tag at start of Runtime

You can configure a start value for numeric tags and tags for date/time values. The tag will be preset to this value when Runtime starts. In this way, you can ensure that the tag has a defined status when Runtime starts.

For external tags, the start value will be displayed on the HMI device until it is overwritten by the PLC or by input.

If no start value was configured, the tag contains the value "0" when starting Runtime.

In WinCC Runtime Professional you can enter a tag value in place of the start value on a tag with the "String" data type. The tag value is saved in the "Project texts" editor and is multilingual. After the text has been translated, it is displayed in Runtime as a language-dependent start value.

Application example

You can assign a default value to an I/O field. Enter the desired default value as start value for the tag that is linked to the I/O field.

See also

Defining the start value of a tag (Page 8128)

Tag limits (Page 8125)

Defining the start value of a tag

Introduction

In WinCC you can configure a start value for a numeric tag and a tag for date/time values which this adopts at Runtime start.

Requirement

- You have created the tag for which you want to define a start value.
- The Inspector window with the tag properties is open.

Procedure

To configure a start value, proceed as follows:

1. In the Inspector window select "Properties > Properties > Values."
2. Enter the desired "Start value."

Alternative procedure

You can also configure the start value directly in the tag table. To view hidden columns, activate the column titles using the shortcut menu.

Result

The start value you selected for the tag is transferred to the project.

See also

Tag limits (Page 8125)

Start value of a tag (Page 8127)

Updating the tag value in Runtime

Introduction

Tags contain process values which change while Runtime is running. Value changes are handled differently at internal and external tags.

Principle

When Runtime starts, the value of a tag is equal to its start value. Tag values change in Runtime.

In Runtime, you have the following options for changing the value of a tag:

- A value change in an external tag in the PLC.
- By input, for example, in an I/O field.
- By running a system function, such as "SetValue."

Updating the Value of External Tags

The value of an external tag is updated as follows:

- Cyclic in operation
  If you select the "Cyclic in operation" acquisition mode, the tag is updated in runtime as long as it is displayed in a screen. The acquisition cycle determines the update cycle for tag value updates on the HMI device. Cyclic acquisition is based on the selected scan cycle time.
- Cyclic continuous
  If you select the "Cyclic continuous" acquisition mode, the tag will be updated continuously in Runtime, even if it is not in the currently-open screen. This setting is activated for tags that are configured to trigger a function list when their value changes, for example.
  Only use the "Cyclic continuous" setting for tags that must truly be updated. Frequent read operations increase communication load.
- On demand
  If you select the "On demand" acquisition mode, the tag is not updated cyclically. It will only be updated on request, for example, by using the "Update Tag" system function.

See also

Tag limits (Page 8125)

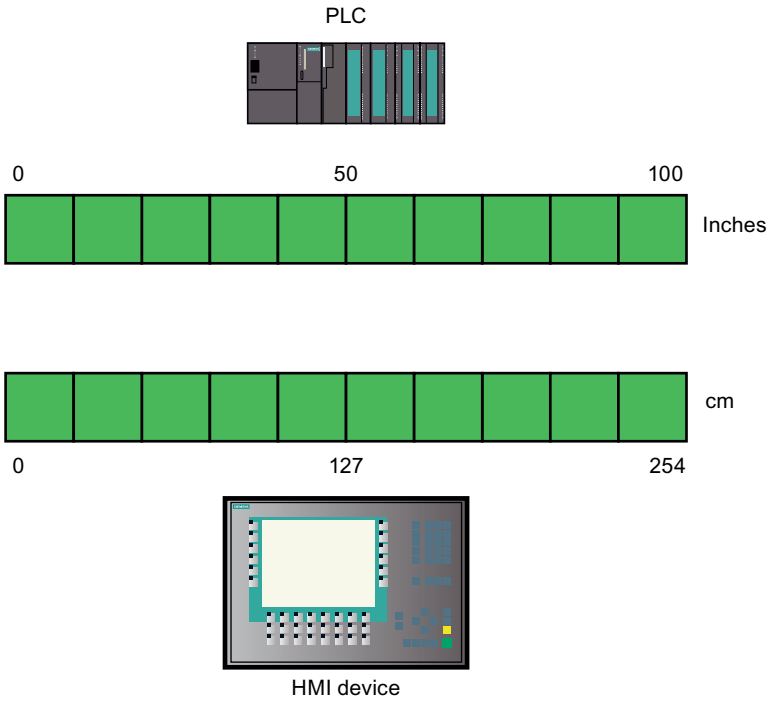
Linear scaling of a tag

Introduction

Numeric data types can be processed with linear scaling. The process values in the PLC for an external tag can be mapped onto a specific value range in the project.

Principle

To apply linear scaling to a tag, you must specify one value range on the HMI device and one on the PLC. The value ranges will be mapped to each other linearly.



As soon as data from the HMI device is written to an external tag, it will be automatically mapped to the value range of the PLC. As soon as data from the HMI device is read from the external tag, a corresponding transformation will be performed in the other direction.

Note

You can also use the system functions "LinearScaling" and "InvertLinearScaling" to automatically convert process values.

Application example

The user enters length dimensions in centimeters but the PLC is expecting inches. The entered values are automatically converted before they are forwarded to the controller. Using linear scaling, the value range [0 to 100] on the PLC can be mapped onto the value range [0 to 254] on the HMI device.

See also

Tag limits (Page 8125)

Applying linear scaling to a tag

Introduction

To apply linear scaling to a tag, you must specify one value range on the HMI device and one on the PLC. The value ranges will be mapped to each other linearly.

Requirement

- The external tag to which linear scaling is to be applied must exist.
- The Inspector window with the properties for this tag is open.

Procedure

To apply linear scaling to a tag, follow these steps:

1. In the Inspector window select "Properties > Properties > Linear scaling."
2. Click on "Enable" to switch on linear scaling.
   Using this option, you can temporarily switch off linear scaling for testing purposes, for example. Settings which were made earlier for linear scaling remain unchanged.
3. In the "PLC" area, enter the start and end values of the value range to be applied to the process values on the PLC.
4. In the "HMI device" area, enter the end and start values of the value range to be applied to the process values on the HMI device.

Result

In Runtime the data will be automatically mapped from one value range to the other.

Note

You can also use the "LinearScaling" and "InvertLinearScaling" system functions to automatically convert process values.

See also

Tag limits (Page 8125)

Connecting a tag to another PLC

Introduction

In WinCC, you can change the PLC connection of a tag at any time. This is needed when you change the configuration of your plant, for example.

Depending on the PLC selected, you may need to modify the configuration of the tag. The tag properties which must be changed will be highlighted in color.

Requirement

- The external tag, whose connection you wish to change, must already exist.
- The connection to the PLC must already exist.
- The Properties window for this tag is open.

Procedure

To change the PLC connection of a tag, proceed as follows:

1. In the Inspector window select "Properties > Properties > General."
2. Select the new connection in the "Connection" field.
   The tag properties that you must change will be highlighted in color in the tag table and in the Inspector window.
3. Change all highlighted properties of the tag to suit the requirements of the new PLC.

Result

The external tag is connected to the new PLC.

See also

Tag limits (Page 8125)

Indirect addressing of tags

Principle

In multiplexes, a type of indirect addressing, the tag used is first determined at runtime. A list of tags is defined for the multiplex tag. The relevant tag is selected from the list of tags in runtime. The selection of the tag depends on the value of the index tag.

In Runtime, the system first reads the value of the index tag. Then the tag which is specified in the corresponding place in the tag list is accessed.

Application example

Using indirect addressing, you could configure the following scenario:

The operator selects one of several machines from a selection list. Depending on the operator's selection, data from the selected machine will be displayed in an output field.

To configure such a scenario, configure the index tag for a symbolic I/O field. Configure the multiplex tag at an I/O field. Configure the tag list of the multiplex tag to reflect the structure of the selection list.

If the operator selects another machine, the value of the index tag will change. The selection field then displays the content of the tag that is pointed to in the tag list in the multiplex tag by the new index value.

See also

Tag limits (Page 8125)

Addressing tags indirectly

Introduction

With indirect addressing, the tag used is first determined at runtime. Instead of a single tag, a list of tags is defined. The list entries consist of an index value and the name of the tag to be used. Using an index tag, you can control which entry in the tag list will be accessed.

Requirement

- The tag which you wish to address indirectly must already exist.
- The index tag must exist.
- The tags which will be contained in the tag list must already exist.
- The Inspector window with the tag properties is open.

Procedure

To address tags indirectly, proceed as follows:

1. In the Inspector window select "Properties > Properties > Multiplexing".
2. Select the "Multiplexing" option to activate indirect addressing.
   Using this option, you can temporarily switch off indirect addressing for testing purposes, for example. Settings which were made earlier for indirect addressing remain unchanged.
3. Select an "Index tag" or define a new tag using the object list.
4. Click the first entry in the "Tags" column in the tag list.
5. Select a tag as a list entry or define a new tag using the object list.
   The entry in the "Index" column will be generated automatically.
6. Repeat step 5 for all tags that you wish to add to the tag list.
7. If necessary, you can use drag-and-drop to change the order of the entries in the list.

Result

In runtime, the system will dynamically access the tag in the tag list which has the same index value as the value currently in the index tag.

See also

Tag limits (Page 8125)

Using tags to trigger functions

Introduction

You can use the values of variables as the triggering event for an action in runtime. To start an action in Runtime, configure a function list for a tag. Include one or more system functions in the function list. The function list is processed when the configured event occurs.

The following events are available for a tag:

- Change in value of the tag
  Function list processing is triggered by each change in the value of the variable.
  If the tag contains arrays, the function list is processed whenever an element of the array changes.
- Violation of the tag's high limit
  The function list is processed when the high limit is violated.
- Violation of the tag's low limit
  The function list is processed when the low limit is violated.

Requirement

- The tag whose value you wish to use as an event already exists.
- The Inspector window with the properties for this tag is open.

Procedure

To configure a tag with a function list, proceed as follows:

1. Under "Properties > Events >" in the Inspector window, select the event for which you want to create a function list.
   The function list associated with the selected event is shown.
2. Click "<Add function>". The second table column contains a selection button.
3. Click the selection button and select a system function.
4. Define the parameter values.

Result

The function list is processed when the configured event occurs in Runtime.

See also

Tag limits (Page 8125)

Defining the acquisition cycle for a tag

Introduction

The value of an external tag can be changed in Runtime by the PLC to which the tag is linked. To ensure that the HMI device is informed of any changes in tag values by the PLC, the values must be updated on the HMI. The value is updated at regular intervals while the tag is displayed in the process screen or is logged. The interval for regular updates is set with the acquisition cycle. The update can also be made continuous.

Requirement

- You have created the tag for which you want to define an acquisition cycle.
- The Inspector window with the tag properties is open.

Procedure

To configure an acquisition cycle for a tag, follow these steps:

1. In the Inspector window select "Properties > Properties > General."
2. If you want to update the tag at regular intervals as long as it is being displayed on the screen or logged, select "Cyclic in operation" as the acquisition mode.
   Or:
   If you want to update the tag at regular intervals even though it is not being displayed on the screen or logged, select "Cyclic continuous" as the acquisition mode.
   The "Cyclic continuous" setting is selected for a tag, for example, that has a function list configured for a change of its value and that is not directly visible in a screen.
3. Select the required cycle time in the "Acquisition cycle" field or define a new acquisition cycle using the object list.

Alternatively, you can configure the acquisition cycle directly in the work area of the tag table. To view hidden columns, activate the column titles using the shortcut menu.

Note

Only use the "Cyclic continuous" acquisition mode for tags that really have to be continuously updated. Frequent read operations generate a heavy communication load.

Result

The configured tag is updated in Runtime with the selected acquisition cycle.

See also

Tag limits (Page 8125)

Address multiplexing

Introduction

Using address multiplexing, you can use a single tag to access a multitude of memory locations within the PLC's address range. You read and write to the addresses without defining a tag for each individual address.

Multiplexing with absolute addressing

When using multiplexing with absolute addressing, you configure tags as placeholders for the address in the PLC to be addressed.

If you want to access, for example, an address of the format "%DBx.DBWy", the expression for multiplexing is as follows:

"%DB[HMITag1].DBW[HMITag2]"

In Runtime, you supply the tag "HMITag1" with the required value for the data block you want to address.

In Runtime, you supply the tag "HMITag2" with the required address from the data block.

Tags are supplied with values, for example, with the help of values from the PLC or via a script.

Multiplexing with absolute addresses is supported for the following PLCs and communication drivers.

- SIMATIC S7-300/400
- SIMATIC S7-1200
- SIMATIC S7-1500

Multiplexing with absolute addresses is not available for data blocks with optimized access.

Multiplexing with symbolic addressing

When multiplexing with symbolic addressing, you access an array element of an array tag in a data block of the connected PLC by means of a multiplex tag and an index tag. The multiplex tag contains the symbolic address of the data block which you want to access. The symbolic address also contains the index tag via which you access the index of the array tag.

If you want to access, for example, the array tag "Arraytag_1" in the data block "Datablock_1", the expression for symbolic addressing is as follows:

"Datablock_1.Arraytag_1["HMITag_1"]

You control the access to the index of the array elements with the HMI-Variable "HMITag_1". In Runtime, you supply the tag with the index of the array element that you want to access.

Multiplexing with symbolic addressing is only available if the following components support symbolic addressing:

- the HMI device
- PLC
- Communication driver

Symbolic addressing is supported by communication drivers:

- SIMATIC S7-1200
- SIMATIC S7-1500

See also

Tag limits (Page 8125)

Configuring address multiplexing with absolute addressing

Introduction

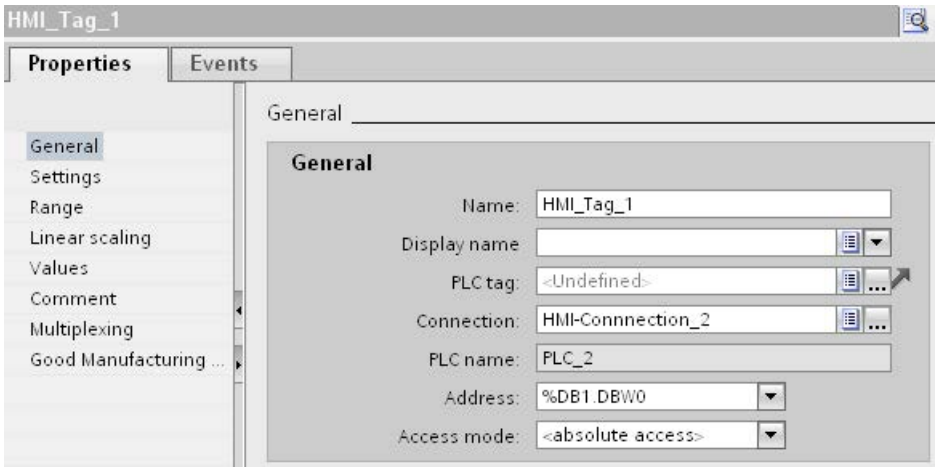
When using address multiplexing, you can efficiently access different addresses in the PLC with the help of a small number of tags. Instead of the absolute address in the PLC, you use tags in order to be able to change the address in Runtime.

Requirement

- The tag for address multiplexing is created and connected to the PLC.
- The Properties window for this tag is open.

Procedure

1. Select the tag for address multiplexing in the tag table, and select "Properties > Properties > General" in the Inspector window. The general properties of the tag are displayed.

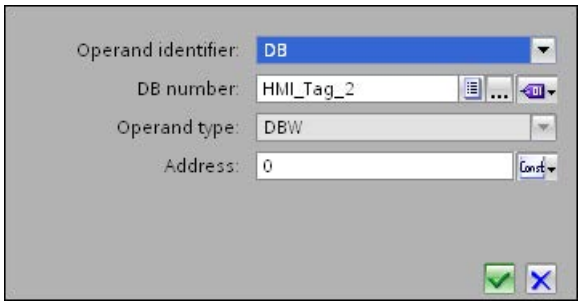


2. Select the "Int" data type for this example.
3. Select the access type "Absolute addressing".

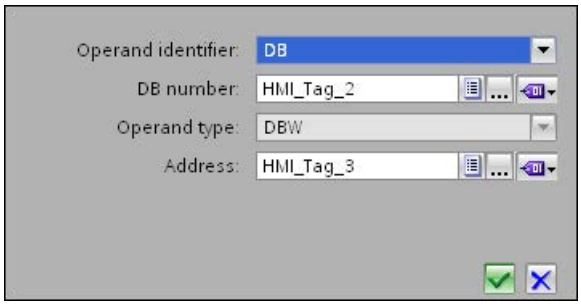
4. Click the selection button in the "Address" field. The address dialog opens.

5. Click the selection button in the "DB number" field and select the entry "HMI tag".

6. In the "DB number" field, click the ... button and select a tag for the DB number in the object list. Or create a new tag with the help of the object list. Accept the tag by clicking the ✓ button.



7. Repeat steps 3 and 4 for the "Address" field and configure a further tag for calling the address area in the data block.



The selection options in the Address dialog depend on the selected data type of the multiplex tag. The Address dialog offers only address settings that can be configured with the selected data type.

Result

In runtime, the multiplex tag is used to access the memory location corresponding to the address currently found in the tag. You control access to the data block with the tag in the DB number field. You control access to the address in the selected data block with the tag in the "Address" field.

Note

The value in the memory location will only be read at the next update cycle for the addressed tag.

If, for example, you use a multiplex tag in a script, do not attempt to access contents of the memory location directly after changing it.

See also

Tag limits (Page 8125)

Configuring address multiplexing with symbolic addressing

Introduction

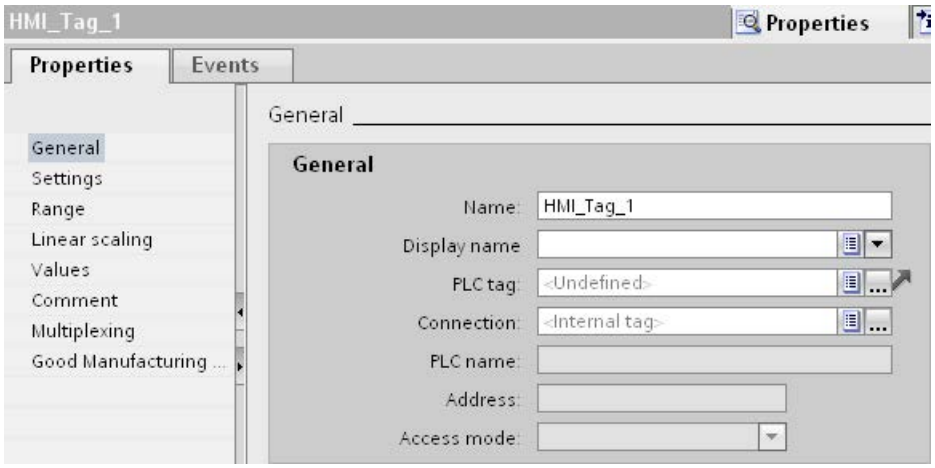
When using address multiplexing, you can efficiently access different addresses in the PLC with the help of a small number of tags. Instead of the symbolic address in the PLC, you use tags in order to be able to change the address in Runtime.

Requirement

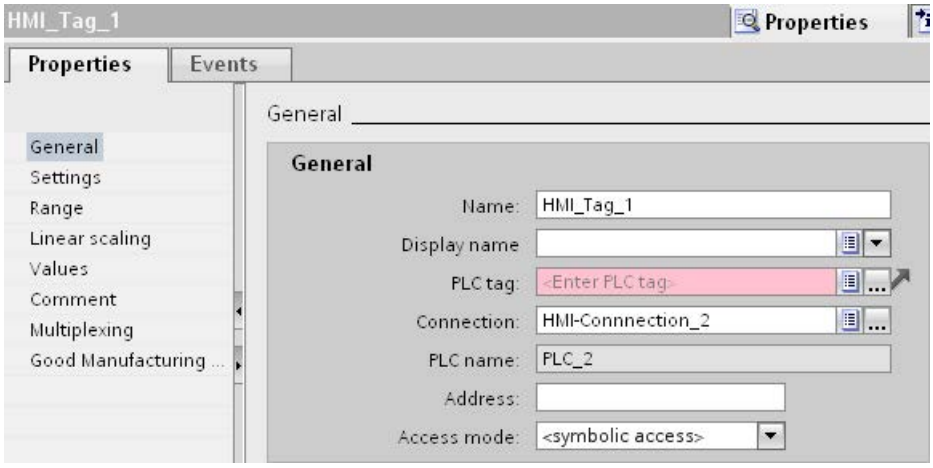
- The tag for address multiplexing is created.
- The Properties window for this tag is open.
- A data block with an array tag is created in the connected PLC.
- The data block was compiled.

Procedure

1. Select the tag for address multiplexing in the tag table, and select "Properties > Properties > General" in the Inspector window. The general properties of the tag are displayed.

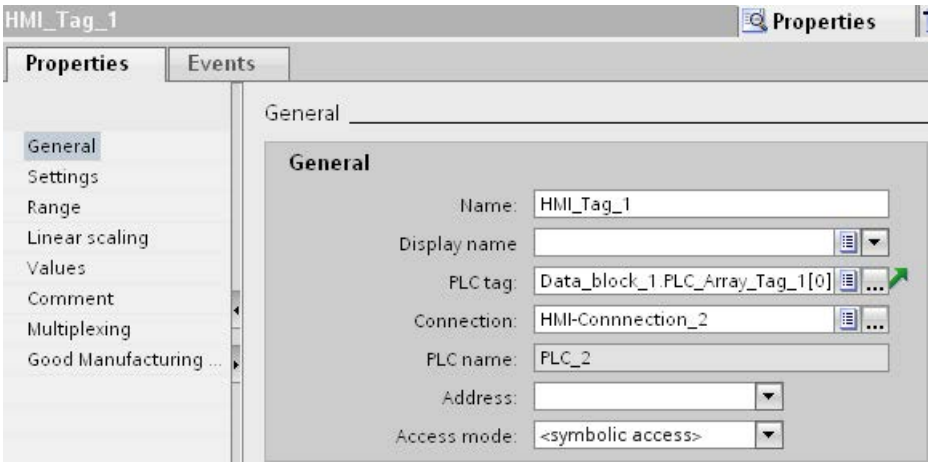


2. Select the connection to the PLC via the "Connection" field.

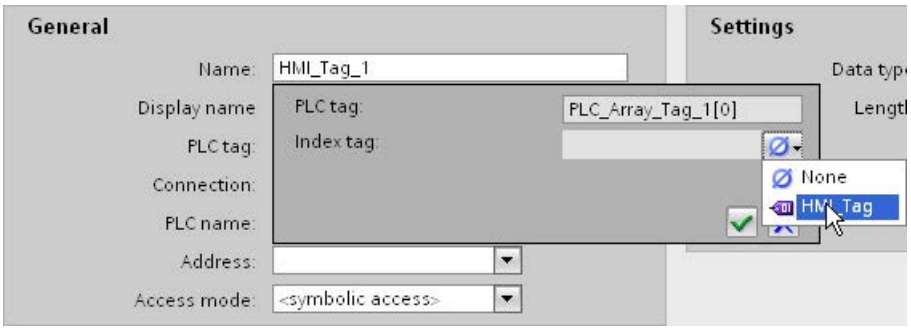


3. Select the access type "Symbolic addressing".

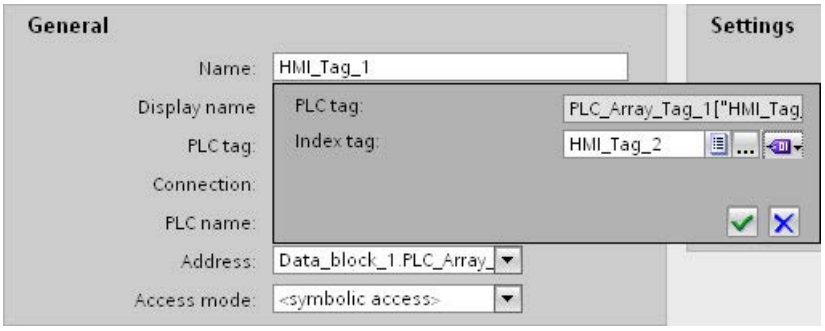
4. Navigate to the data block of the PLC via the "PLC tag" field and select an array element of the array tag.



5. Click the selection button in the "Address" field. The address dialog opens.
6. Click the selection button in the "Index tag" field and select the entry "HMI tag".



7. In the "Index tag" field, click the ... button and select a tag for the array index in the object list. Or create a new tag with the help of the object list. Accept the tag by clicking the ✓ button.



Result

In Runtime, the array element whose index value is contained in the index tag is accessed.

See also

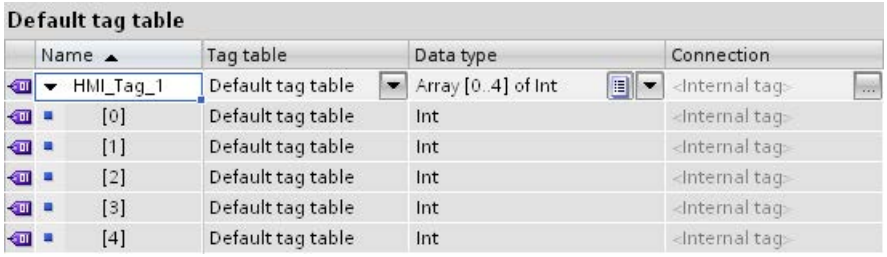
Tag limits (Page 8125)

10.2.3 Working with arrays

10.2.3.1 Basics on arrays

Definition

Array data of a uniform data type is successively arranged and is addressed within the address space to allow access to these data by means of an index. The individual array elements are addressed by means of an integer index. The properties of each array element are the same and are configured at the array tag in a data block of the PLC program.



Default tag table

Name	Tag table	Data type	Connection
HMI_Tag_1	Default tag table	Array [0..4] of Int	<Internal tag>
[0]	Default tag table	Int	<Internal tag>
[1]	Default tag table	Int	<Internal tag>
[2]	Default tag table	Int	<Internal tag>
[3]	Default tag table	Int	<Internal tag>
[4]	Default tag table	Int	<Internal tag>

Advantages

You can configure multiple array elements with the same properties at one time using a single array tag. You can then use each array element as any other tag in your configuration.

Restrictions

The following restrictions apply to the use of arrays:

- Not all HMI devices support array tags.
- An array may contain only one dimension.
- The lower index of an array must begin with "0".

Application examples

Array tags can be used in the following situations:

- To group process values in profile trends: You map process values to trends which are acquired at different points in time, for example.
- To access specific values which are grouped in trends: You output all values of the profile trend by stepping up the index tag, for example.

- To configure discrete alarms with successive bit number.
- To save the complete machine data records in a recipe.

License rule for runtime

An array tag is counted in WinCC Runtime as one PowerTag, regardless of the number of array elements.

Special features

WARNING

Increased system load and performance losses

Read or write access to a single array element always includes read or write access to all array elements of the array tag. Transfer of the data of large arrays from and to the PLC usually takes longer compared to the transfer of a basic data type. This may cause communication overload and disruption as a result.

Example:

- An array tag which consists of 100 array elements of data type "Real" was configured.
- If an array element with a length of four bytes changes, 100 x 4 bytes are written to the PLC.

CAUTION

Data inconsistency at array tags

If the value of a single element must be changed in an array tag, the whole array is read, changed and rewritten as a complete array. Changes carried out in the meantime to other array elements in the PLC are overwritten during rewriting.

You should always prevent the HMI device and the PLC from concurrently writing values to the same array tag. Use synchronous transfer of recipe data records to synchronize an array tag with the PLC.

See also

Creating array tags (Page 8144)

Examples of arrays (Page 8145)

Basics of tags (Page 8108)

10.2.3.2 Creating array tags

Introduction

Create an array tag to configure a large number of tags of the same data type. The array elements are saved to a consecutive address space.

You can create an array tag as an internal tag or as an external tag.

If you want to create an array tag as an external tag, first configure an array tag in a data block of the connected PLC. You then connect the array tag to an HMI tag.

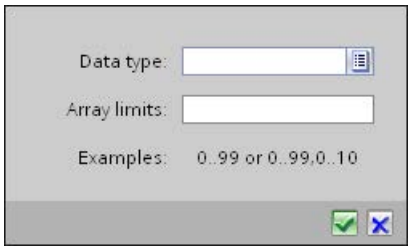
Requirement

- The HMI tag table is open.

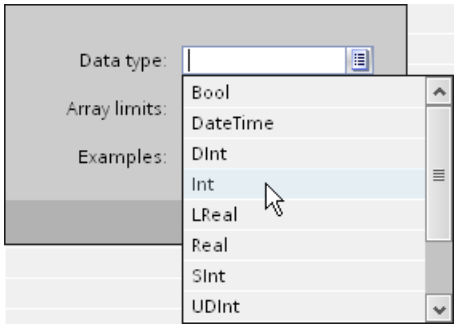
Procedure

To create an array tag, follow these steps:

1. Double click <Add> in the "Name" column of the HMI tag table.
   A new HMI tag is created.
2. Click in the Data type column and select the "Array" data type.
3. Click ... in the data type column. The dialog box for configuring the array is opened.



4. Select the desired data type for the array tag in the "Date type" field.



5. Define the number of array elements in the "Array limits" field. The lower limit must begin with "0".
6. Click . The settings for the array are saved.
7. Save the project.

Result

An array tag is created. The properties of the array elements are inherited from the parent array tag.

See also

Basics on arrays (Page 8142)

10.2.3.3 Examples of arrays

Introduction

Array tags combine a number of tags, e.g. 100 array elements. You use array tags as complete arrays in the following places:

- In the "Alarms" editor
- In the "Recipes" editor
- For address multiplexing
- In the trend view

You use individual array elements everywhere in the configuration like HMI tags.

Examples

You can configure an array tag with the corresponding number of array elements to handle multiple tags of the same data type.

- The individual array elements can be accessed indirectly by means of a multiplex index tag, for example.
- Use these index tags to operate and monitor the array elements.

See also

Basics on arrays (Page 8142)

10.2.4 Working with cycles

10.2.4.1 Cycle basics

Introduction

Cycles are used to control actions that regularly occur in runtime. Common applications are the acquisition cycle and the update cycle.

Principle

In Runtime, actions that are performed regularly are controlled by cycles. Typical applications for cycles:

- Acquisition of external tags
  The acquisition cycle determines when the HMI device will read the process value of an external tag from the PLC. Set the acquisition cycle to suit the rate of change of the process values. The temperature of an oven, for example, changes much more slowly than the speed of an electrical drive.
  Do not set the acquisition cycle too low, since this will unnecessarily increase the communication load of the process.

The smallest possible value for the cycle depends on the HMI device that will be used in your project. For most HMIs, this value is 100 ms. The values of all other cycles are always an integer multiple of the smallest value.

Application example

You can use cycles for the following tasks:

- To regularly update a tag.
- To draw attention to maintenance intervals.

See also

Basics of tags (Page 8108)

10.2.5 Displaying tags

10.2.5.1 Outputting tag values in screens

Introduction

In runtime you can output tag values in the screens of the operator device in the form of a trend. A trend is a graphic representation of the values that a tag takes during runtime. Use the "Trend display" graphic object to represent it. Process values for the trend display are loaded by the PLC from the ongoing process.

The values to be displayed are determined individually within a fixed, configurable cycle. Cyclically-triggered trends are suitable for representing continuous curves, such as the changes in the operating temperature of a motor.

Displayed values

You will need to configure a trend view in a screen so that tag values are displayed on the HMI device. When configuring the trend view, specify which tag values are to be displayed.

You can control the updating of the trend display by defining the cycle time.

10.2.5.2 Configuring trend displays for values from the PLC

Introduction

You use a trend view to graphically represent values that a tag assumes during the process.

Requirement

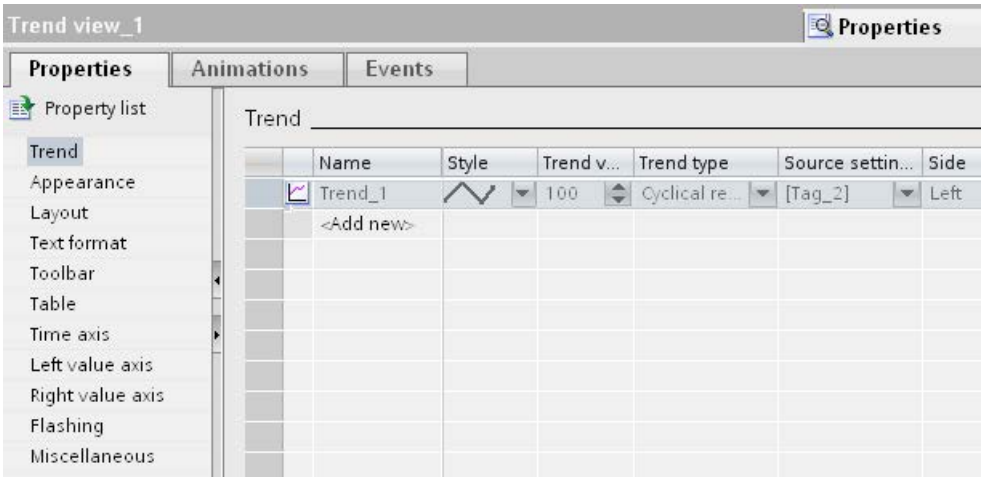
- A screen is open.
- The Inspector window with the trend view properties is open.

Procedure

To configure a trend view, follow these steps:

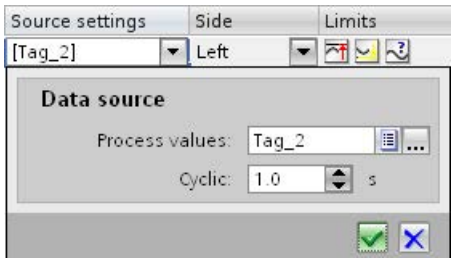
1. Add the "Trend view" object from the toolbox in the "Control" group to the screen.

2. Select the "Trend" category from the "Properties" group in the Inspector window and double-click "<Add>" in the "Name" column.



3. Assign a name to the trend in the "Name" column.
4. In the "Style" column, use the selection button to open the "Style" dialog and select the style of the line.
5. Select the number of trend values in the "Trend values" column.

6. In the "Settings" column, use the selection button to open the "Data source" dialog and select the tags to supply the trend with values.
   Specify the cycle for reading the tags from the PLC.



7. You can make other settings in the dialogs of the Inspector window. For example, you can select the "Display table" option in the "Table" category to display a value table beneath the trend view.

Note

If you hold down the <CTRL> key and double-click the trend view, the trend view is activated. You set the column width and the position of the columns in the table header of the values table in active mode. In order to activate the trend view the zoom factor has to be set to 100 %.

Result

In runtime, the values of the selected tags are displayed in the configured trend view.

10.3 Working with alarms

10.3.1 Basics

10.3.1.1 Alarm system in WinCC

Introduction

The alarm system allows you to display and record operating states and faults on the HMI device that are present or occur in a plant.

An alarm may have the following content:

No.	Time	Date	Alarm text	Status	Alarm class
5	12:50:24:590	24.02.2007	Boiler pressure above high limit.	Incoming Outgoing	Warning: Color Red

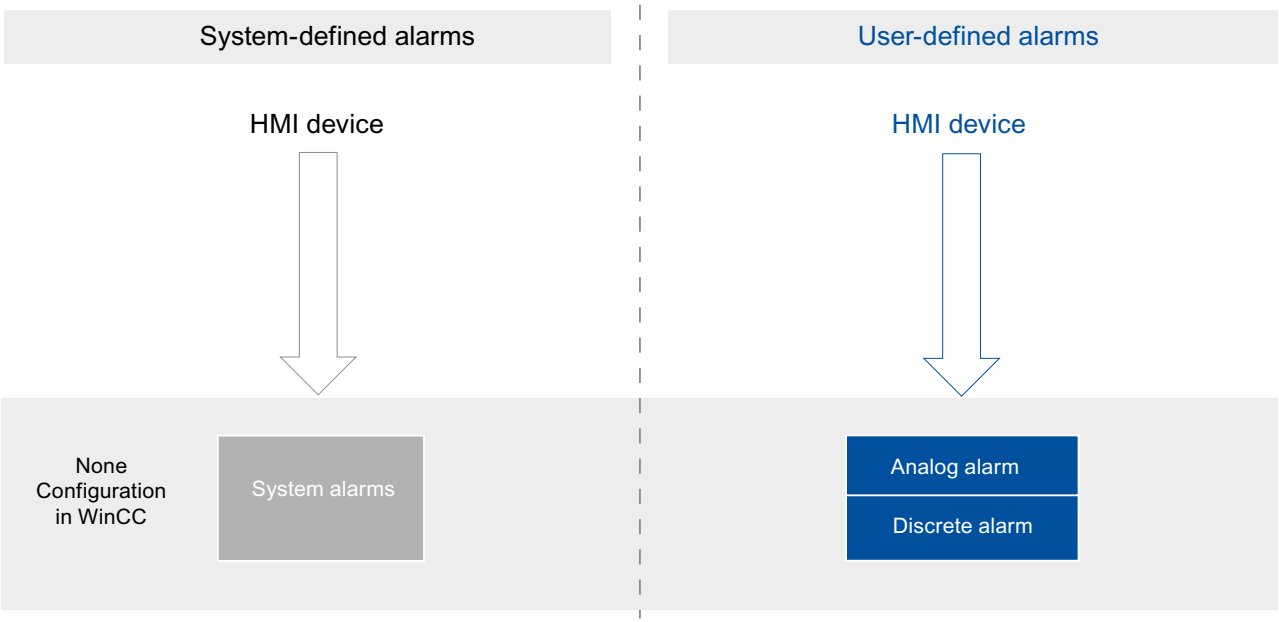
Alarm system in WinCC

The alarm system processes a variety of alarm types. The alarm procedures can be broken down into system-defined alarms and user-defined alarms:

- User-defined alarms are used to monitor the plant process.
- System-defined alarms monitor the HMI device.

The detected alarm events are displayed on the HMI device. Targeted access to the alarms combined with supplementary information about individual alarms ensures that faults are localized and cleared quickly. This reduces stoppages or even prevents them altogether.

The following figure shows the alarm system structure:



10.3.1.2 Alarm types

Overview of the alarm types

Introduction

The alarm types serve various purposes for monitoring the plant. The alarms from the individual alarm types are configured and triggered in different ways.

Select the relevant tab in the "HMI alarms" editor to configure alarms based on the individual alarm types.

Alarm types in WinCC

WinCC supports the following alarm types:

User-defined alarms

- Analog alarms
  - Analog alarms are used to monitor limit violations.
- Discrete alarms
  - Discrete alarms are used to monitor states.

System-defined alarms

- System events
  - System events belong to the HMI device and are imported into the project.
  - System events monitor the HMI device.

System-defined alarms

System alarms

Examples for alarms

- "An online connection to the PLC is established."

Description

A system alarm indicates the status of the system, plus communication errors between the HMI device and system.

Under "Runtime settings > Alarms" you specify how long a system alarm is shown on the HMI device.

Support

The reference contains a list of the possible system events, along with the cause and possible remedies. If you contact online support because of a system alarm on the HMI device, you will need the alarm number and tags used in the system alarm.

User-defined alarms

Analog alarms

Description

Analog alarms signal limit violations during the process.

Example

The speed of the mixer in a fruit juice mixing plant must not be too high or too low. You can configure analog alarms to monitor the speed of the mixer. If the high or low limit for the speed of the mixer is violated, an alarm is output on the HMI device containing the following alarm text, for example: "Mixer speed is too low".

Discrete alarms

Description

Discrete alarms indicate a status in the current process.

Example

A fruit juice mixing plant consists of several tanks containing the ingredients. To ensure the correct mixing ratio of water, fruit concentrate, sugar, and flavoring, the valves in the intakes open and close at the right moment. This operation should be monitored.

You configure a suitable discrete alarm for all the valve states. If a valve on one of the four tanks opens or closes, an alarm is displayed, such as "Water valve closed".

The operator can thus monitor whether the plant is producing correctly.

10.3.1.3 Alarm states

Introduction

An alarm assumes various alarm states in Runtime. The user analyzes and reports on the process execution with reference to the alarm states.

Description

Every alarm has an alarm status. The alarm states are made up of the following events:

- Incoming
  The condition for triggering an alarm is satisfied. The alarm is displayed, such as "Boiler pressure too high."
- Outgoing
  The condition for triggering an alarm is no longer satisfied. The alarm is no longer displayed as the boiler was vented.
- Acknowledge
  The operator acknowledges the alarm.

Alarms without acknowledgment

The following table shows the alarm states for alarms that do not have to be acknowledged:

Status	Description
Incoming	The condition for an alarm is satisfied.
Outgoing	The condition for an alarm is no longer satisfied.

Alarms with acknowledgment

The following table shows the alarm states for alarms that have to be acknowledged:

Status	Description
Incoming	The condition for an alarm is satisfied
Outgoing, not acknowledged	The condition for an alarm is no longer satisfied. The operator has not acknowledged the alarm.
Outgoing, subsequently acknowledged	The condition for an alarm is no longer satisfied. The operator has acknowledged the alarm after this time.
Incoming, acknowledged	The condition for an alarm is satisfied. The operator has acknowledged the alarm.
Outgoing, acknowledged first	The condition for an alarm is no longer satisfied. The operator acknowledged the alarm while the condition was still satisfied.

If necessary, the HMI device shows you each time one of these states occurs.

Note

The display text for alarm states depends on the language and configuration.

10.3.1.4 Alarm classes

Basics on alarm classes

Introduction

Many alarms occur in a plant. These are all of different importance. You can assign the alarms of your project to alarm classes to clearly show the user which of the alarms are most important.

Description

The alarm class defines how an alarm is displayed. The alarm class specifies if and how the user has to acknowledge alarms of this alarm class.

A new alarm class with mandatory acknowledgment is generated in WinCC.

Note

The choice of display modes for alarm classes depends on the options on your HMI device.

Examples of how to use alarm classes

- The alarm "Speed of fan 1 in upper tolerance range" has alarm class "Warnings". The alarm is displayed with a white background. The alarm does not have to be acknowledged.
- The alarm "Speed of fan 2 has exceeded upper warning range" is assigned to the "Errors" alarm class. The alarm is displayed with a red background and flashes at high frequency in runtime. The alarm is displayed until the user acknowledges it.

Using alarm classes

Use the following alarm classes to define the acknowledgment model and display of alarms for your project:

- Predefined alarm classes
  You cannot delete predefined alarm classes and edit them only to a limited extent. Predefined alarm classes have been created for each HMI device under "HMI alarms > Alarm classes".
- Custom alarm classes
  You can create new alarm classes under "HMI alarms > Alarm classes", configure how you want the alarms to be displayed, and define an acknowledgment model for alarms of this alarm class. The possible number of custom alarm classes depends on which runtime is used in your project.

See also

Creating alarm classes (Page 8159)

Predefined alarm classes

Predefined alarm classes

The following alarm classes already created in WinCC for every HMI device:

Alarm classes for user-defined alarms

- "Warnings"
  The "Warnings" alarm class is intended to show regular states and routines in the process. Users do not acknowledge alarms from this alarm class.
- "Errors"
  The "Errors" alarm class is intended to show critical or dangerous states or limit violations in the process. The user must acknowledge alarms from this alarm class.

Alarm class for system-defined alarms

- "System"
  The "System" alarm class contains alarms that display states of the HMI device and the PLCs.

See also

Creating alarm classes (Page 8159)

10.3.1.5 Acknowledgment

Acknowledging alarms

Introduction

To make sure that an alarm was registered by the plant operator, configure this alarm so that it is displayed until acknowledged by the operator. Alarms that display critical or hazardous states in the process have to be acknowledged.

Description

Acknowledging an alarm changes the alarm status from "Incoming" to "Acknowledged". When the operator acknowledges an alarm, the operator confirms that he or she has processed the status that triggered the alarm.

Triggering acknowledgment of an alarm

In Runtime, you trigger alarm acknowledgments in various ways:

- Acknowledgement by the authorized user at the HMI device
- Automatic acknowledgment by the system without operator action, e.g. by means of
  - Tags
  - PLC
  - System functions in function lists

Acknowledging alarms that belong together

To make the alarm system clearer and easier to use in Runtime, you can configure an alarm group. You can acknowledge all alarms belonging to this alarm group in a single pass.

Acknowledgment by the PLC

Discrete alarms will be automatically acknowledged by the PLC, if necessary. The acknowledgment is triggered by a bit in the "Acknowledgment tag PLC". You define the bit and tag at the configuration stage.

Acknowledgment of an alarm on the HMI device

In Runtime, the user acknowledges an alarm in one of the following ways, depending on the configuration:

- Using the acknowledgment button <ACK> on the HMI device
- Using the button in the alarm view
- Using configured function keys or buttons in screens

Note

Acknowledgment button <ACK> on the HMI device

To ensure that critical alarms are processed only by authorized users, protect the "ACK" button on the HMI devices, including the operating controls and display objects of the alarms. Use the appropriate operator authorization for this.

Note

HMI device dependency

The acknowledgement key <ACK> is not available on all HMI devices.

Acknowledgment model

Overview

You define the acknowledgment model for an alarm class. Alarms that are assigned to this alarm class will be acknowledged on the basis of this acknowledgment model. The following acknowledgment model is used in WinCC:

- Alarm without acknowledgment
  This alarm comes and goes without having to be acknowledged. There is no visible response from the system.
- Alarm with simple acknowledgment
  This alarm must be acknowledged as soon as the event that triggers the alarm occurs. The alarm remains pending until it is acknowledged.

10.3.1.6 Alarm groups

Introduction

Many alarms from different areas and processes occur in a plant. You can compile associated alarms into alarm groups.

Alarm groups

You can use the alarm groups to monitor the parts of the plant and to acknowledge the associated alarms together as required.

Alarm groups can contain alarms from different alarm classes. You only assign alarms that require acknowledgment to alarm groups.

Using alarm groups

It is a good idea to compile alarm groups for alarms such as the following:

- Alarms that are caused by the same fault.
- Alarms of the same type
- Alarms from a machine unit, such as "Fault in drive XY"
- Alarms from an associated part of the process, such as "Fault in cooling water supply"

Display in Runtime

In Runtime, the "Alarm group" column displays the number of the alarm group to which the alarm belongs.

10.3.1.7 Alarm number

Assigning alarm numbers

The system assigns unique alarm numbers within an alarm type.

Note

When adapting alarm numbers, observe the uniqueness of the alarm number within an alarm type.

10.3.2 Working with alarms

10.3.2.1 Alarm components and properties

Overview

You configure the components of alarms in WinCC. The following table shows the basic components of alarms:

Alarm class	Alarm number	Time of day	Date	Alarm status	Alarm text	Alarm group	Tooltip	Trigger tag	Limit value
Warning	1	11:09:14	06.08.2007	IO	Maximum speed reached	2	This alarm is ...	speed_1	27
System	110001	11:25:58	06.08.2007	I	Switch to "Online" mode	0	This alarm is ...	PLC-Variable_1	–

Alarm class

Alarm classes, such as "Warnings" or "Errors." The alarm class defines the following for an alarm:

- Acknowledgment model
- Appearance in Runtime (e.g. color)

Alarm number

An alarm is identified by a unique alarm number. The alarm number is assigned by the system. You can change the alarm number to a sequential alarm number, if necessary, to identify alarms associated in your project.

Time and date

Every alarm has a time stamp that shows the time and date at which the alarm was triggered.

Alarm status

An alarm has the events "Incoming," "Outgoing," "Acknowledge." For each event, a new alarm is output with the current status of the alarm.

Alarm text

The alarm text describes the cause of the alarm.

The alarm text can contain output fields for current values. The values you can insert depend on the Runtime in use. The value is retained at the time at which the alarm status changes.

Alarm group

The alarm group bundles individual alarms.

Tooltip

You can configure a separate tooltip for each alarm; the user can display this tooltip in Runtime.

Trigger tag

Each alarm is assigned a tag as trigger. The alarm is output when this trigger tag meets the defined condition, e.g. when its state changes or it exceeds a limit.

Limit value

Analog alarms indicate limit violations. Depending on the configuration, WinCC outputs the analog alarm as soon as the trigger tag exceeds or undershoots the limit value.

10.3.2.2 Configuring alarms

Overview of alarm configuration tasks

Steps to configure alarms

Configuring alarms in WinCC involves the following steps:

1. Edit and create alarm classes
   You use the alarm class to define how an alarm will be displayed in runtime and to define the acknowledgment model for it.
2. Creating tags in the "HMI tags" editor
   - Configure the tags for your project.
   - You create range values for the tags.
3. Creating tags in the "HMI alarms " editor
   - Create custom alarms and assign these the tag to be monitored, alarm classes, alarm groups, and other properties.
   - You can also assign system functions or scripts to the alarm events.
4. Output of configured alarms
   To output configured alarms, configure an alarm view or an alarm window in the "Screens" editor.

Additional configuration tasks

Additional tasks may be necessary for configuring alarms, depending on the requirements of your project:

1. Creating alarm groups
   You assign the alarms of your project to alarm groups according to their association, such as by the cause of the problem (power failure) or source of the error (Motor 1).
2. Configuring Loop-In-Alarm
   A Loop-In-Alarm is configured in order to change to a screen containing relevant information on an alarm received.

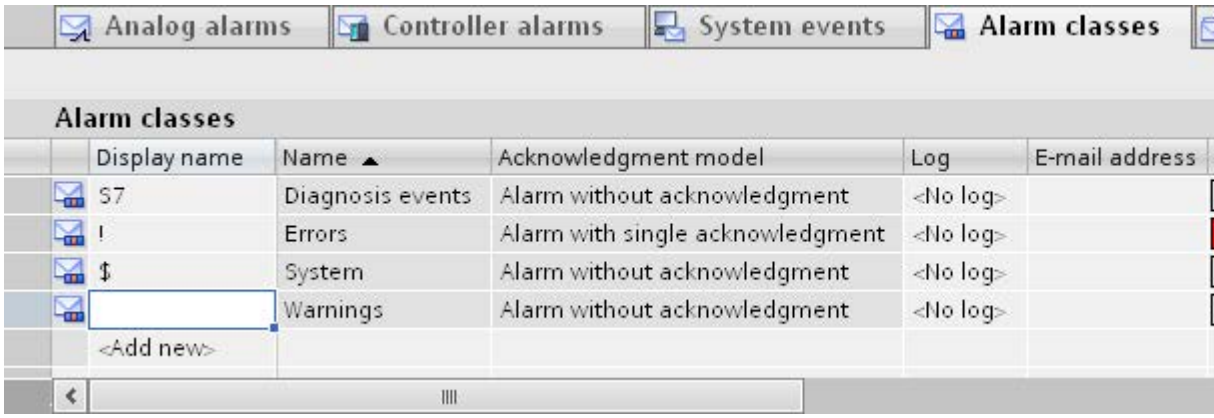
Creating alarm classes

Introduction

Create alarm classes in the "Alarm classes" tab of the "HMI alarms" editor. Some default alarm classes are already created for every project. You can create additional custom alarm classes. You can create up to 32 alarm classes.

Requirement

-
-



Procedure

To

1.

1. Double-click "<Add>" in the table.
   A new alarm class is created. Each new alarm is automatically assigned a static ID.
   The properties of the new alarm class are shown in the Inspector window.
2. Configure the alarm class under "Properties > Properties >General" in the Inspector window.
   – Enter a "Name" and the "Display name".
   – Depending on the HMI device, you can also activate logging, or automatic sending of e-mails.
3. Define the acknowledgment model for the alarm class under "Properties > Properties > Acknowledgment" in the Inspector window.
4. Change the default text under "Properties > Properties > Status" in the Inspector window.
   This text indicates the status of an alarm in Runtime.
5. Change the default colors under "Properties > Properties > Colors" in the Inspector window.
   Depending on the HMI device, also change the flashing characteristics.

These settings define how alarms from this alarm class are displayed in Runtime.

Note

To display the alarm classes in color in Runtime, the "Use alarm class colors" option must be activated. In the project navigation, enable "Runtime settings > Alarms > General > Use alarm class colors" accordingly. This option is selected in a new project in WinCC.

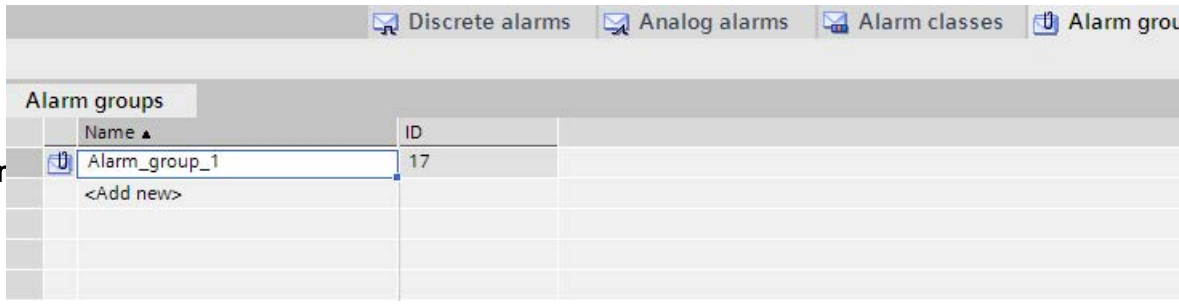
Configuring alarm groups

Introduction

Create alarm groups on the "Alarm Groups" tab in the "HMI alarms" editor. An alarm group is a compilation of single alarms. You assign alarms in an alarm group by association, such as cause of the problem or source of the error. If you acknowledge an alarm from this alarm group in Runtime, all other alarms in the alarm group are acknowledged automatically.

Requirement

- You have created a project.
-
-



Creating a new alarr

1.
2. In the work area of the table, double-click "<Add>" in the first free row.
   A new alarm group is created.
3. You can overwrite the proposed "Name".

Result

An alarm group is created. For the group acknowledgment of alarms in Runtime, assign the associated alarms that require acknowledgment to an alarm group.

Configuring discrete alarms

Introduction

Discrete alarms triggered by the PLC indicate status changes in a plant. They indicate the opened or closed state of a valve, for example.

The following sections describes the configuration procedures in the "HMI alarms" editor. You can also configure discrete alarms in the "HMI tags" editor.

Requirements

- The "HMI alarms" editor is open.
- The Inspector window is open.
- You have created the required alarm classes and alarm groups.

Procedure

To configure a discrete alarm, proceed as follows:

1. Open the "Discrete alarms" tab.
2. To create a new discrete alarm, double-click in the work area on "<Add>".
   A new discrete alarm is created.

3. To configure the alarm, select "Properties > Properties >General" in the Inspector window:
   - Enter an alarm text as event text.
     Use the functions of the shortcut menu to format the text on a character-by-character basis, or to insert output fields for HMI tags, or texts from the text lists.
   - You can renumber the alarm.
   - Select the alarm class and the alarm group, if necessary.
4. In the Inspector window, select the tag and the bit that triggers the alarm under "Properties > Properties > Trigger". Note the following information:
   - Use the data types "Int" or "UInt" to select an HMI tag.
   - When you select a PLC tag, use the data types "Int" or "Word". The input and output area of a PLC tag is unsuitable as trigger.
   - Use trigger tag bits only for alarms.
   - Do not use trigger tags for anything else.
   - If you want to acknowledge the alarm via the PLC, use this tag also as PLC acknowledgment tag.

Note

Note the method used to count bits in the utilized PLC when specifying the bit. For more information, refer to the "Communication" section in the PLC Online Help.

Note

If the object does not yet exist in the selection list, create it directly in the object list and change its properties later.

Status-dependent alarm texts

To display a different text independent of the alarm status, link a text list to the alarm text. You control the text list with a tag.

Additional settings for discrete alarms

Creating a tooltip

To configure a tooltip for the alarm, follow these steps:

- Enter your text under "Properties > Properties > Tooltip".

Configuring event-driven tasks

To configure event-driven tasks, such as a loop-in alarm, follow these steps:

1. Select the discrete alarm.
2. Select "Properties > Events" in the Inspector window and configure a new function list for the relevant event.

See also

Configuring loop-in alarm (Page 8167)

Configuring analog alarms

Introduction

Analog alarms indicate limit violations. For example, if the speed of a motor drops below a certain value, an analog alarm is triggered.

Requirements

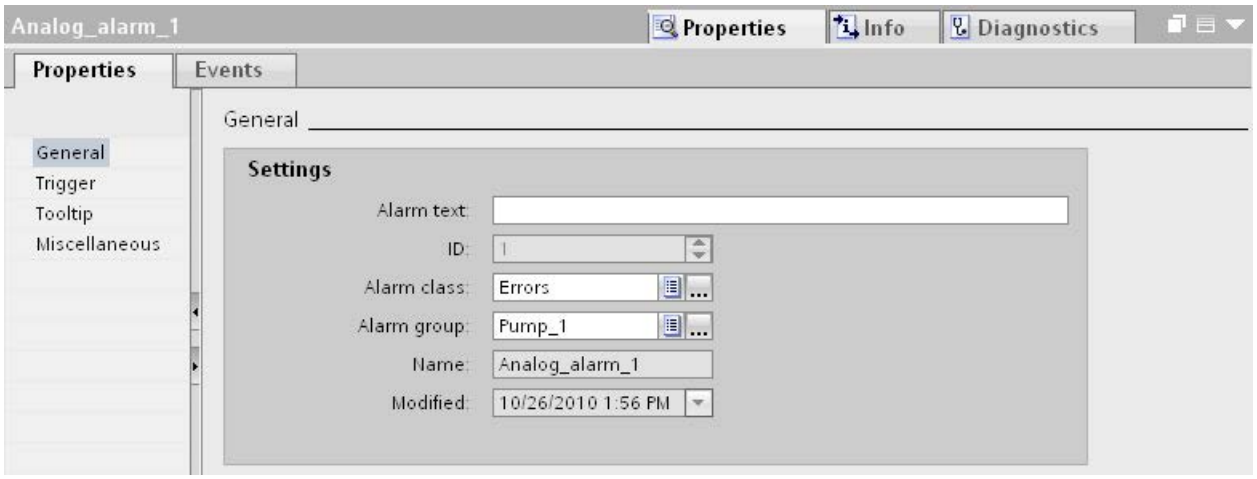
- The "HMI alarms" editor is open.
- The Inspector window is open.
- You have created the required alarm classes and alarm groups.

Procedure

To configure an analog alarm, proceed as follows:

1. Click the "Analog alarms" tab.
2. To create a new analog alarm, double-click in the table on "<Add>".
   A new analog alarm is created.

3. To configure the alarm, select "Properties > Properties >General" in the Inspector window:
   - Enter an alarm text as event text.
     Format the text character-for-character using the shortcut menu.
     Using the shortcut menu, you can insert output fields for HMI tags, or text from text lists.
   - You can renumber the alarm.
   - Select the alarm class and the alarm group, if necessary.



4. Configure the tag that triggers the alarm under "Properties > Properties > Trigger > Settings".
   Do not use trigger tags for anything else.

Configure limit values for an analog alarm

1. In the Inspector window, click the button under "Properties > Properties > Trigger > Limit > Value".
   - To use a constant as limit value, select "Constant".
     Enter the required limit value.
   - To use a tag as limit value, select "HMI tag".
     The button is shown. Use this button to select the tag you want to use.

   Note

   If the tag included in the selection does not yet exist, create it in the object list and change its properties later.

2. Select the mode:
   - "High limit violation": The alarm is triggered when the limit is exceeded.
   - "Low limit violation": The alarm is triggered when the limit is undershot.

Optional settings for analog alarms

Setting the delay time

To set the delay time, proceed as follows:

- Enter a time period in the Inspector window under "Properties > Properties> Trigger > Settings > Delay".
  The alarm is only triggered when the trigger condition is still present after the delay time has elapsed.

Setting the deadband

Note

If a process value fluctuates around the limit, the alarm associated with this fault may be triggered multiple times. To prevent this from happening, configure a deadband or delay time.

To enter the deadband, follow these steps:

1. Under "Properties > Properties> Trigger > Deadband > Mode", select the change in alarm status for which the deadband is to be taken into account.
2. Enter a constant value under "Value".
3. To define the deadband value as a percentage of the limit, set the "in %" check box.

Creating a tooltip

To configure a tooltip for the alarm, follow these steps:

- Select "Properties > Properties > Tooltip" in the Inspector window and enter your text.

Configuring event-driven tasks

To configure event-driven tasks, such as a loop-in alarm, follow these steps:

1. Select the analog alarm.
2. Select "Properties > Events" in the Inspector window and configure a new function list for the relevant event.

See also

Configuring loop-in alarm (Page 8167)

Adding an output field to alarm text

Introduction

In WinCC, you can insert output fields into the alarm text which display the content of tags.

Requirements

- The "HMI alarms" editor is open.
- The alarm is selected.

Output of a tag value in the alarm text

To insert an output field for a tag value in the alarm text, proceed as follows:

1. Place the cursor onto the required position in the event text.
2. Select "Insert tag output field" in the shortcut menu.
3. Open the object list under "Tag" and select a tag.
   You can also create the tag in the object list.
4. Under "Format", specify the length of the output field and the format for tag value output in the alarm text.
   Configure an output field of sufficient size. Otherwise, the tag content is not output to the full extent in the alarm.
5. Click to save your entries.

WinCC inserts a placeholder for the output field into the alarm text: "<tag: n, [tag name]>" whereby n = text string length.

Editing output field properties

To edit the properties of an output field, proceed as follows:

- Double-click on the output field in the alarm text and edit the settings.

Deleting an output field from the alarm text

To delete an output field from the alarm text, proceed as follows:

- Select the output field in the alarm text and then select the "Delete" command from the shortcut menu.

Note

The sequence of the tag output fields in the alarm text depends on the language. Changing the tag of an output field in one language causes the modified output field to appear at the end of the alarm text in all other languages.

Formatting alarm text

Requirement

- The "HMI alarms" editor is open.
- An alarm has been created.

Procedure

To format an alarm text, proceed as follows:

1. Select the alarm to edit.
2. In the Inspector window, select the characters to format under "Properties > Properties > General > Alarm text".
3. Select the formatting from the shortcut menu, e.g. "Underscored" or "Uppercase".

Note

It is not possible to underline tags and text list entries.

Result

The selected characters are displayed in Runtime with the selected formatting.

Removing format settings

To remove all text formats, proceed as follows:

1. In the Inspector window, select the characters whose formatting you want to remove in the alarm text.
2. Select "Delete formatting characters" from the shortcut menu.

Result

The selected characters are displayed in unformatted notation in Runtime.

Configuring loop-in alarm

Introduction

A Loop-In-Alarm is configured in order to change to a screen containing relevant information on an alarm received.

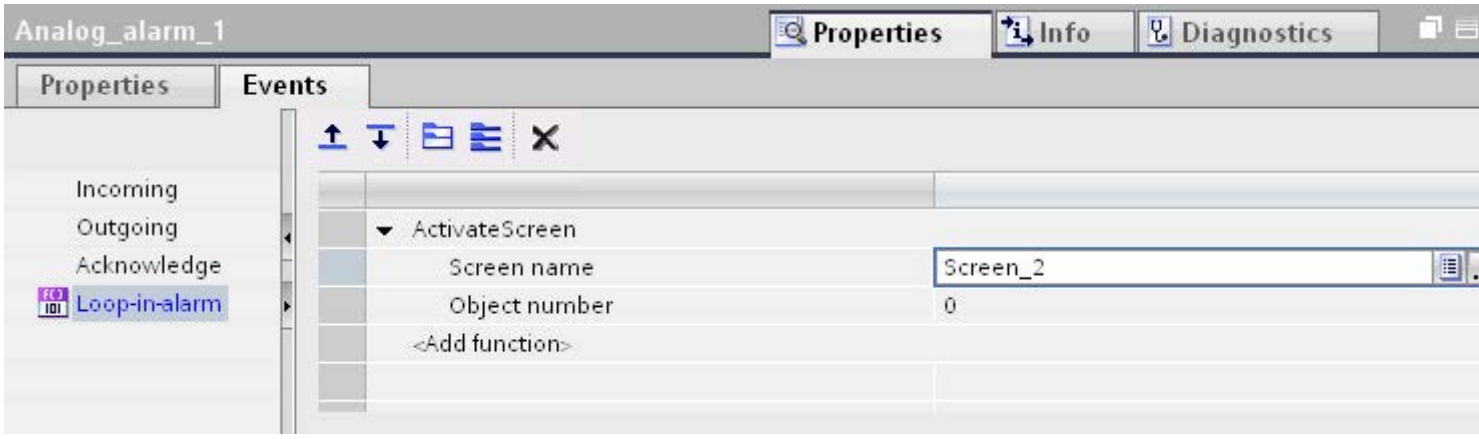
Requirement

- The screen called by the Loop-In-Alarm has been created.
- The "HMI alarms" editor is open.

Procedure

To configure a Loop-In-Alarm for an alarm, proceed as follows:

1. Click the tab that contains the alarm for which you want to configure the Loop-In-Alarm.
2. Select the alarm.
3. In the Inspector window, select "Properties > Events > Loop-In-Alarm".
4. Select the "ActivateScreen" system function.
5. Select the screen called by the Loop-In-Alarm as parameter.



Note

To configure the Loop-In-Alarm for an alarm view with an "alarm line" format, use a button with the following system functions:

- "EditAlarm" for HMI devices with keys
- "AlarmViewEditAlarm" for HMI devices without keys

The system functions trigger the "Loop-In-Alarm" event. The alarm line has no buttons.

Result

If you click on the "Loop-In-Alarm" button of the alarm view in Runtime, a screen is opened with information on the selected alarm.

See also

Configuring analog alarms (Page 8163)

Configuring discrete alarms (Page 8161)

Alarms in the "HMI tags" Editor

Configuring discrete alarms in the "HMI tags" editor

Introduction

In WinCC, you can create and edit discrete and analog alarms, including the trigger tags, in the "HMI tags" editor.

Note

If you delete, move or copy objects in the "HMI tags" editor, the changes also take effect in the "HMI alarms" editor.

Requirements

The "HMI tags" editor is open.

Procedure

To configure a discrete alarm, proceed as follows:

1. To create a tag, click on "<Add>" in the table at the top of the work area.
   A new tag is created.
2. Configure an internal or external tag as required.
   - Use the data types "Int" or "UInt" to select an HMI tag.
   - When you select a PLC tag, use the data types "Int" or "Word". The input and output area of a PLC tag is unsuitable as trigger.
3. Select the tag at the top of the work area.
4. Click on "<Add>" in the table on the "Discrete alarms" tab at the bottom of the work area.
   A new discrete alarm is created for the tag. If you have selected the incorrect data type, the tag will be marked in the discrete alarm.
5. Configure the discrete alarm in the Inspector window:
   - Enter the alarm text under "Properties > Properties > General > Alarm text".
     You can also insert output fields into the alarm text.
   - Select an alarm class.
   - Select the trigger bit of the tag that triggers the discrete alarm under "Properties > Trigger".
6. You can create additional discrete alarms to monitor the tags.

   Note

   A tag is monitored using only one alarm type. You should therefore create either analog alarms **or** discrete alarms for a tag.

Result

The configured discrete alarms are created in the "HMI tags" editor and displayed in the "HMI alarms" and "HMI tags" editors.

Configuring analog alarms in the "HMI tags" editor

Introduction

In WinCC, create the discrete and analog alarms, including the trigger tags, in the "HMI tags" editor. You can also edit the alarms as in the "HMI alarms" editor. You can create up to two range values for a tag. You monitor these limits with analog alarms.

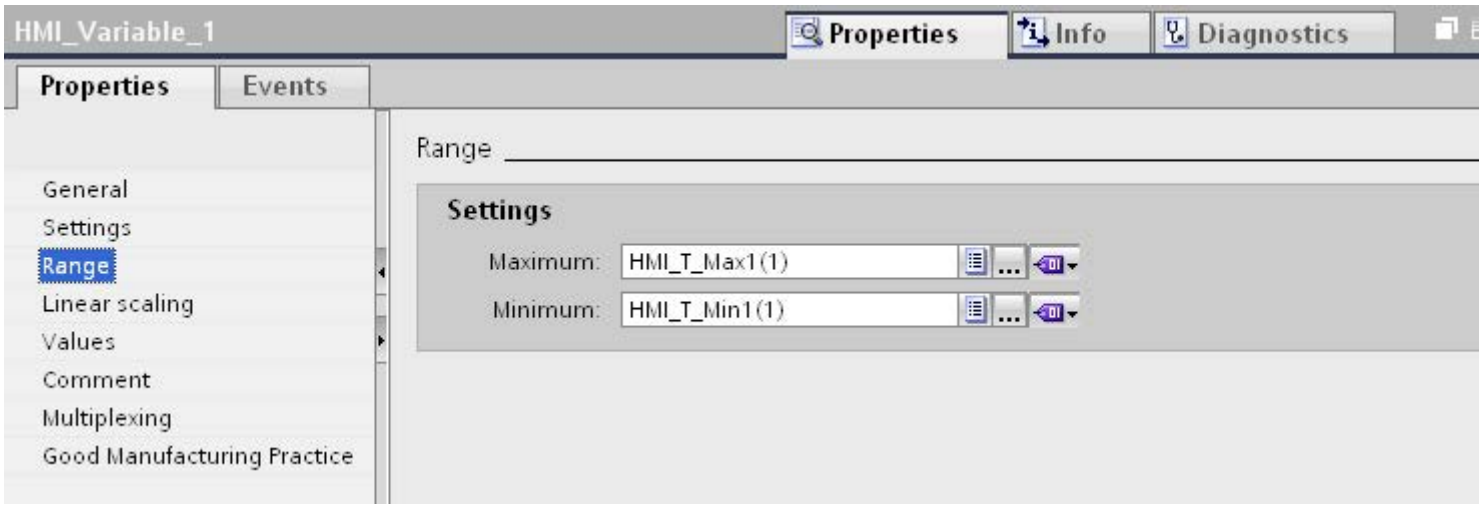
Requirements

The "HMI tags" editor is open.

Procedure

To configure an analog alarm in the "HMI tags" editor, proceed as follows:

1. To create a tag, click on "<Add>" in the table at the top of the work area.
   A new tag is created.
2. Configure an internal or external tag as required.
3. In the Inspector window, configure the range values of the tags under "Properties > Properties > Range":
   – Select whether to use a "Constant" or an "HMI tag" as limit value for your range values. The object list opens when you select "HMI tag". Select the tag you want to use.



1. Click the "Analog Alarms" tab at the bottom of the work area.
   Create an analog alarm for both range values.
2. Select an analog alarm and configure it in the Inspector window:
   – Enter the alarm text under "Properties > Properties > General > Alarm text".
   – You can also insert output fields into the alarm text.
   – You can change the default alarm class.
3. Configure the analog alarms as in the "HMI alarms" editor.
4. Complete the configuration of all analog alarms.

Note

A tag is monitored using only one alarm type. You should therefore create either analog alarms **or** discrete alarms for a tag.

Result

The configured analog alarms are created in the "HMI tags" editor and displayed in the "HMI alarms" and "HMI tags" editors.

10.3.2.3 Configuring alarm output

Overview of configuring alarm output

Steps to complete when configuring alarm output

You configure the alarm output in WinCC in the following steps:

1. Create alarm view
   Use the display and control objects in the "Screens" editor to display alarms in Runtime.
2. Configure acknowledgment
   In the "Screens" editor, you can set the operator action that will trigger the acknowledgment.

Additional configuration tasks

Additional tasks may be necessary for configuring alarm views, depending on the requirements of your project:

1. Setting up authorizations
   To make sure only authorized operators process the alarms, assign authorizations for the alarm view and the function keys of the HMI device.
2. Configuring the filtering of the alarm view
   You configure the filtering of the alarms in Runtime in the "Screens" editor. You can also configure alarm views that only display selected alarms.
3. Configure operator input alarms
   Configure the operator input alarms on the operator controls of the HMI device in the "Screens" editor. A preconfigured operator input alarm is output for an operator action. An operator input is, e.g. the acknowledgment of an alarm.

Displaying alarms

Options for displaying alarms on the HMI device

WinCC offers the following options for displaying alarms on the HMI device:

- Alarm view
  The alarm view is configured in a screen. More than one alarm can be displayed simultaneously, depending on the configured size. You can configure multiple alarm views with different contents.
- Alarm window
  The Alarm window is configured in the "Global screen" editor. The alarm window can display multiple alarms at the same time, depending on the configured size. The alarm window is not assigned to any screen. Depending on the configuration, the alarm window is opened when an alarm of a specific alarm class is pending. Based on the configuration, it is not closed until the alarm is acknowledged.

Additional signals

- Alarm indicator
  The alarm indicator is a configurable, graphical icon. When an alarm comes in, the alarm indicator is displayed on the HMI device. You configure the alarm indicator in the "Global screen" editor.
  The alarm indicator has two states:
  - Flashing: At least one alarm that requires acknowledgment is pending.
  - Static: The alarms are acknowledged but at least one of them has not gone out yet. The alarm indicator also displays the number of pending alarms according to the HMI device.
- System functions
  You can configure a list of functions for the event associated with an alarm. These functions must be executed in Runtime when the event occurs.
  Use system functions for alarms in WinCC to control the alarm view or the alarm window other than via the toolbar.

Displaying the predefined alarm classes in Runtime

The following table shows the symbols used to display the predefined alarm classes in the alarm view:

Alarm class	Displayed icon
"Errors"	!
"System"	$
"Warnings"	<No symbol>

Configuring an alarm view

Introduction

Current alarms are displayed in Runtime in an alarm view or alarm window.

Requirement

- A screen is open in the "Screen" editor.
- The "Tools" task card is open.

Configuring alarms for the alarm view

To specify the alarms that will be shown in the alarm view, proceed as follows:

1. Insert an "Alarm view" object from the "Tools" task card into the screen.
2. Select the alarm view.
   – In the Inspector window, select "Properties > Properties > General > View > Current alarm states".
     Set whether to display alarms with and/or without mandatory acknowledgment.
   – To display all alarms in the alarm buffer, enable "Alarm buffer".
3. In the table, activate the alarm classes to be displayed in the alarm view.

Configuring the layout of the alarm view

To specify how the alarms are shown in the alarm view, proceed as follows.

1. Under "Properties > Properties > Layout > Settings > Lines per alarm" in the Inspection window, specify the number of lines to display for each alarm.
2. In "Properties > Properties > View", select the control elements that are available on the HMI device.
3. Configure the columns under "Properties > Properties > Columns":
   - Under "Visible columns" select the columns to be output in the alarm view.
   - Under "Properties Column", define the properties of the columns.
   - Under "Sort", select the sorting order of the alarms.

Result

Alarms of various alarm classes are output in the alarm view during runtime.

See also

Alarm view (Page 8090)

Configuring an alarm window

Introduction

The alarm window displays current alarms. The alarm window is configured in the "Global screen" editor. The alarm window is not assigned to any screen. Depending on the configuration the alarm window is opened when an alarm that belongs to a specific alarm class is active. Depending on the configuration, it is not closed until the alarm is acknowledged. The HMI device can still be used, even if alarms are pending and displayed. An alarm window is displayed and configured like an alarm view.

To hide an alarm window during configuration, create it on its own level.

Requirement

- The "Global Screen" editor is open.
- The "Tools" task card is displayed.
- The Inspector window is open.

Procedure

Proceed as follows to configure an alarm window:

1. Insert an "Alarm window" object from the "Tools" task card into the global screen.
2. Configure the alarm window like an alarm view.
3. Under "Properties > Properties > Mode > Window" in the Inspector window, select how the alarm window reacts and is operated in Runtime.
   – Activate "Modal" if the alarm window is to retain the focus in Runtime after a screen change.
     This option is important, as switching back and forth between the screen and different windows with <Ctrl+TAB> is not supported.

Result

During runtime, the alarms of the selected alarm class are displayed in the alarm window.

See also

Alarm window (Page 8092)

Configuring an alarm indicator

Introduction

The alarm indicator uses a warning triangle to indicate that alarms are pending or require acknowledgement. If an alarm of the configured alarm class occurs, the alarm indicator is displayed.

The alarm indicator has two states:

- Flashing: At least one alarm that requires acknowledgment is pending.
- Static: At least one of the acknowledged alarms has not gone out yet.

During configuration, specify whether Runtime has to open an alarm window when you operate the alarm indicator.

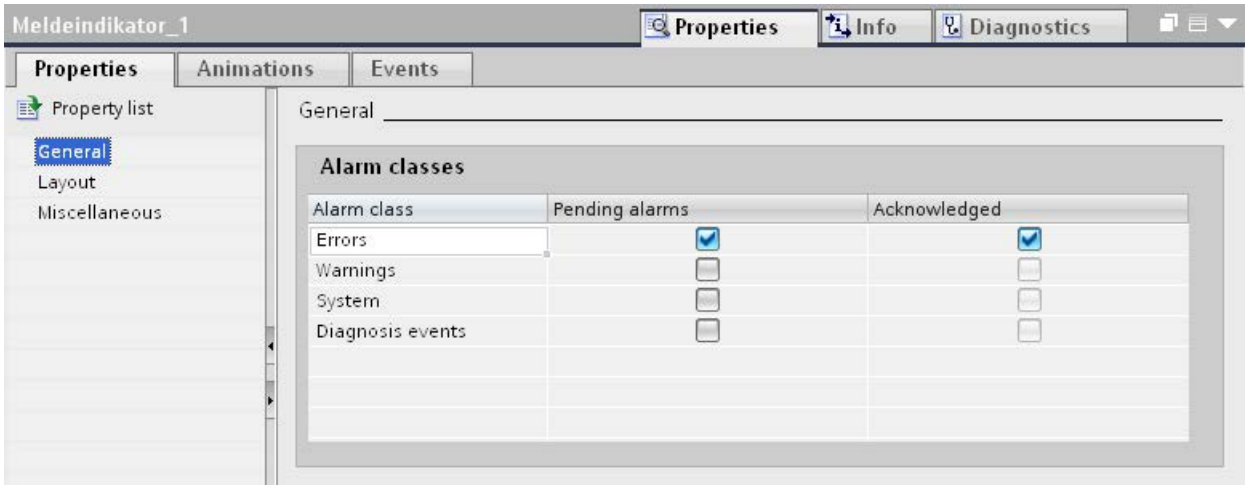
Requirement

- The "Global Screen" editor is open.
- The "Tools" task card is open.
- The Inspector window is open.

Procedure

Proceed as follows to configure the alarm indicator:

1. Insert the "Alarm indicator" object from the "Tools" task card into the work area.
2. Select the alarm indicator.
3. Under "Properties > Properties > General" in the Inspector window, select the alarm classes to be displayed by the alarm indicator.
   Specify whether to display pending and/or acknowledged alarms in the alarm indicator.



4. Under "Properties > Event", assign system function "ShowAlarmWindow" to an event of the alarm indicator.

Note

If you have configured a permanent window in the screen or template, do not position the alarm window and alarm indicator in the vicinity of the permanent window. Otherwise the alarm window and the alarm indicator are not displayed in Runtime. However, the permanent window is not visible in the "Global screen" editor.

Result

The alarm indicator is displayed if alarms from the selected alarm class are pending or need to be acknowledged in Runtime. The alarm window opens when the user operates the alarm indicator.

See also

Alarm indicator (Page 8095)

10.3.2.4 Acknowledging alarms

Configuring alarm acknowledgment by means of alarm class

Introduction

To configure an alarm with alarm acknowledgment, assign it to an alarm class with the "Alarm with single acknowledgment" acknowledgment model.

Requirement

- The "HMI alarms" editor is open.
- The required alarm class has been created.
- The required alarm has been created.

Selecting the acknowledgment model for an alarm class

The acknowledgment model for a predefined alarm class has already been set. You can only set the acknowledgment model for user-defined alarm classes. Proceed as follows:

1. In the "HMI alarms" editor, click the "Alarm class" tab and select the alarm class.
2. Select the required acknowledgment model under "Properties > Properties > Acknowledgment" in the Inspector window.

Assign alarms to an alarm class requiring acknowledgment

Proceed as follows to assign an alarm to an alarm class requiring acknowledgment.

1. In the "HMI alarms" editor, click the tab for the alarm type and select the alarm.
2. Under "Properties > Properties > General" in the Inspector window, select the alarm class of the alarm.

Result

The alarm will not disappear in Runtime until it is acknowledged by the operator.

Configuring trigger for alarm acknowledgment

Introduction

You always specify the acknowledgment requirement for an alarm using the alarm class. Then the operator acknowledges the alarm using the "ACK" function key of the HMI device or the "Acknowledgment" button of the alarm view.

The following options are also available to trigger acknowledgment:

- Configuring a button to acknowledge an alarm
- Acknowledgment of a Discrete Alarm by the PLC

Requirements

- The "HMI alarms" editor is open.
- The required alarm class has been created.
- The required alarm has been created.
- An alarm view and a button are created in the "Screens" editor.

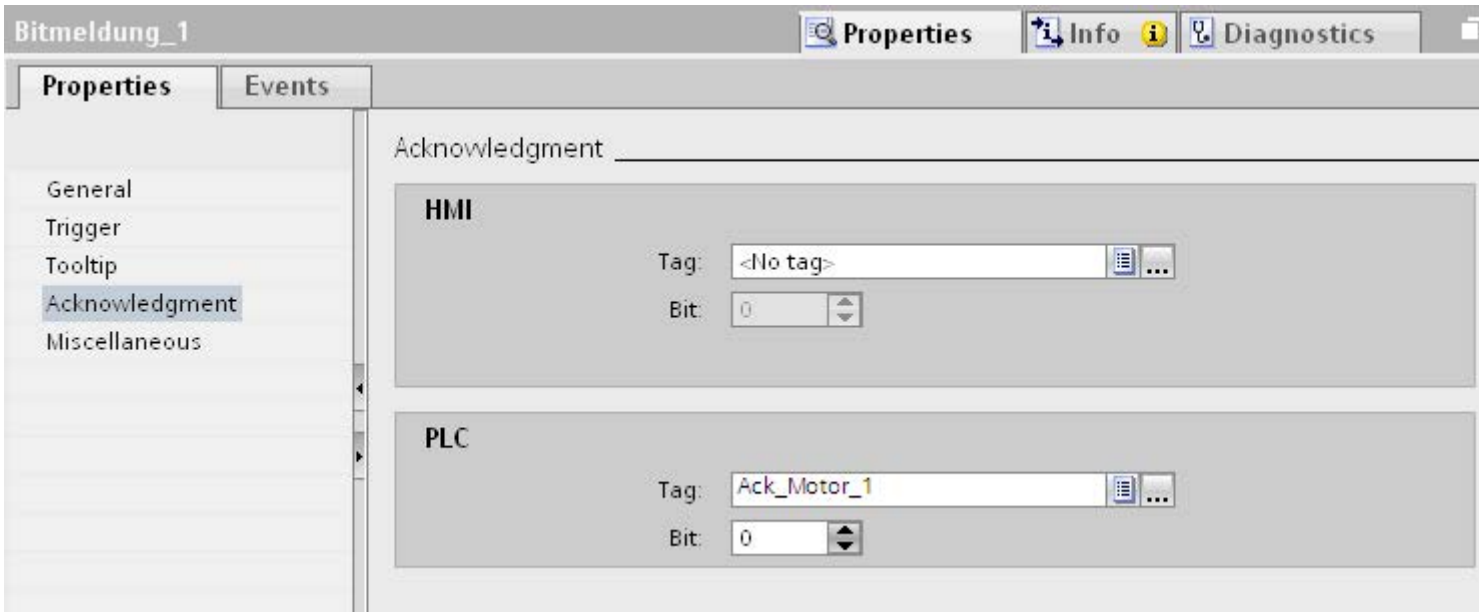
Configuring a button to acknowledge an alarm

To configure a button for acknowledging an alarm, proceed as follows:

1. Select the button in the "Screens" editor.
2. Under "Properties > Events" in the Inspector window, assign the "AlarmViewAcknowledgeAlarm" system function to the "Click" event.
3. Select the alarm view as parameter.

Acknowledgment of a Discrete Alarm by the PLC

1. In the "HMI alarms" editor, click the "Discrete alarm" tab and select the discrete alarm.
2. In the Inspector window, select the tag and the bit that acknowledges the PLC alarm under "Properties > Properties > Acknowledgment > PLC".



Acknowledging alarms in an alarm group

The various acknowledgment options have the following effect in alarm groups:

- Operator acknowledges the alarm
  You acknowledge the alarm of an alarm group, for example, using the <ACK> acknowledgment key or a function key. All alarms of the alarm group are acknowledged.
- The PLC acknowledges the alarm
  The alarm of an alarm group is acknowledged by setting a bit in the tag.
  Only the alarm to be acknowledged is acknowledged.

Sending alarm acknowledgments to the PLC

Requirement

- The "HMI alarms" editor is open.
- The required alarm has been created and assigned to an alarm class requiring acknowledgment.

Note

You cannot send the acknowledgment of analog alarms to the PLC.

Sending alarm acknowledgments to the PLC

To configure that acknowledgment of an alarm is sent to the PLC, follow these steps:

1. In the "HMI alarms" editor, click the "Discrete alarm" tab and select the discrete alarm.
2. In the Inspector window, select "Properties > Properties > Acknowledgment".
3. Under "HMI", select the tag and the bit set by the alarm acknowledgment function.

Note

The HMI device and PLC only have read access to the acknowledgment tag memory area.

Result

If the operator acknowledges the alarm in Runtime, the operating step is forwarded to the PLC.

10.3.3 Operating alarms in Runtime

10.3.3.1 Alarms in Runtime

Alarms

Alarms indicate events and states on the HMI device which have occurred in the system, in the process or on the HMI device itself. A status is reported when it is received.

An alarm could trigger one of the following alarm events:

- Incoming
- Outgoing
- Acknowledge
- Loop-in-alarm

The configuration engineer defines which alarms must be acknowledged by the user.

An alarm may contain the following information:

- Date
- Time
- Alarm text
- Location of fault
- Status
- Alarm class
- Alarm number
- Alarm group

Alarm classes

Alarms are assigned to various alarm classes.

- "Warnings"
  Alarms of this class usually indicate states of a plant such as "Motor switched on". Alarms in this class do not require acknowledgement.
- "Errors"
  Alarms in this class must always be acknowledged. Error alarms normally indicate critical errors within the plant such as "Motor temperature too high".
- "System"
  System alarms indicate states or events which occur on the HMI device.
  System alarms provide information on occurrences such as operator errors or communication faults.
- Custom alarm classes
  The properties of this alarm class must be defined in the configuration.

Alarm buffer

Alarm events are saved to an internal buffer. The size of this alarm buffer depends on the HMI device type.

Alarm view

The alarm view shows selected alarms or alarm events from the alarm buffer. Whether alarm events have to be acknowledged or not is specified in your configuration.

Alarm window

The alarm window is not assigned to any screen. Depending on the configuration the alarm window is opened when an alarm that belongs to a specific alarm class is active.

You can configure the order in which the alarms are displayed. You can choose to display the alarms in ascending or descending order of their occurrence. The alarm window can also be set to indicate the exact location of the fault, including the date and time of the alarm event. By means of configuration, the display can be filtered in such a way that only alarms that contain a specific character string will be shown.

Alarm indicator

The alarm indicator is a graphic symbol that is displayed on the screen when an alarm of the specified alarm class is activated.

The alarm indicator can have one of two states:

- Flashing: At least one unacknowledged alarm is pending.
- Static: The alarms are acknowledged but at least one of them is not yet deactivated. The number indicates the number of queued alarms.

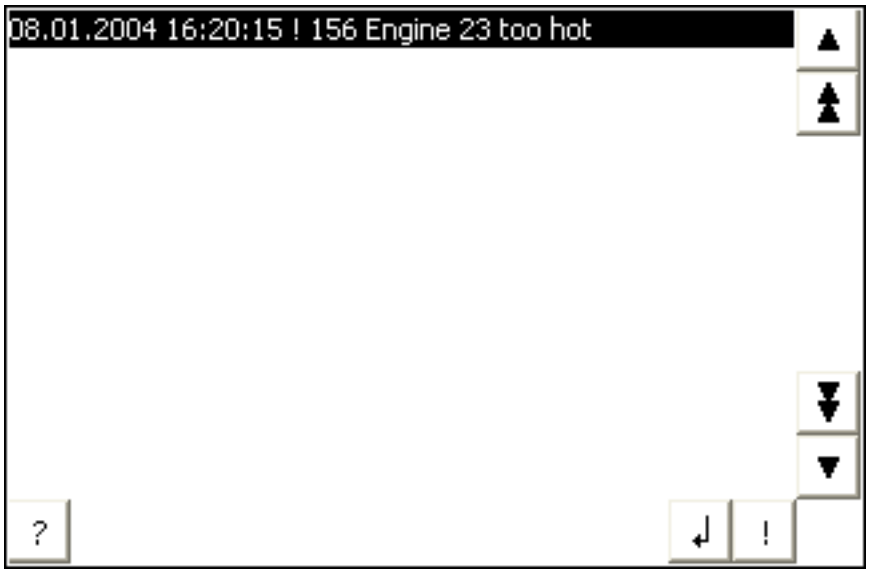
10.3.3.2 Simple alarm view, simple alarm window in runtime

Application

The simple alarm view shows selected alarms or alarm events from the alarm buffer. The layout and operation of the simple alarm window correspond to that of the simple alarm view.

Note

In the Engineering System, for example, dynamize the visibility of an object in the "Animations" tab of the Inspector window. In Runtime, the "Simple alarm view" does not support animations. If you configured an animation and, for example, run a consistency check on the project, an error alarm is displayed in the output window.



Layout

Depending on the configuration, in the alarm view different columns with information regarding an alarm or an alarm event are displayed.

To differentiate between the different alarm classes, the first column in the alarm view contains an icon:

Icon	Alarm class
!	"Errors"
empty	"Warnings"
depends on the configuration	Custom alarm classes
$	"System"

Operation

You use the alarm view as follows, depending on how it is configured:

- Acknowledging alarms
- Editing alarms

Control elements

The buttons have the following functions:

Button	Function
!	Acknowledge alarm
↲	Loop-In-Alarm Changes to the screen that contains information about the error event

Button	Function
?	Displaying a tooltip for an alarm
▶	Displays the full text of the selected alarm in a separate window, namely the alarm text window In the alarm text window, you can view alarm texts that exceed the space available in the Alarm view. Close the alarm text window with the ✕ button.
▲	Scrolls one alarm up.
⏫	Scrolls one page up in the alarm view.
⏬	Scrolls one page down in the alarm view.
▼	Scrolls one alarm down.

Format of the control elements

The display of the buttons for using the simple alarm view depends on the configured size. You should therefore check on the HMI device whether all the required buttons are available.

10.3.3.3 Alarm indicator in Runtime

Application

The alarm indicator is displayed if alarms of the specified alarm class are pending or require acknowledgment.

Layout

The alarm indicator can have one of two states:

- Flashing: At least one unacknowledged alarm is pending.
- Static: The alarms are acknowledged but at least one of them has not gone out yet. The displayed number indicates the number of queued alarms.

Operation

Depending on the configuration, when operating the alarm indicator an alarm window is opened. The alarm indicator can only be operated with the touch screen.

10.3.3.4 Acknowledging alarms

Introduction

You can acknowledge alarms in Runtime according to your project configuration settings. You can acknowledge alarms as follows:

- Using the display and control object buttons
- Using the "ACK" key on your HMI device
- Using individually-configured function keys or buttons

If an operator authorization is configured for an individual control, the alarms can only be acknowledged by authorized users.

To automatically acknowledge alarms in Runtime, use the system functions and the option "Acknowledgment by the PLC".

Acknowledgment variants

You acknowledge individual alarms or multiple alarms together in Runtime. They are distinguished as follows:

- Single acknowledgment
  Acknowledgment of an alarm using a button or a function key.
- Acknowledge alarm groups
  Acknowledgment of all the alarms of an alarm group using a button or a function key.

Requirement

- An alarm is displayed on the HMI device.

Procedure

To acknowledge an alarm, proceed as follows:

1. Select the alarm.
2. Click on the ! button.

Result

The alarm status changes to "Acknowledged". If the condition for triggering an alarm no longer applies, the alarm status also changes to "Outgoing", and it is no longer displayed on the HMI device.

10.3.4 Reference

10.3.4.1 System functions for alarms

System functions

System functions are predefined functions you can use to implement many tasks in runtime, even with no programming knowledge. You use system functions in a function list.

The table shows all the system functions available for displaying and editing alarms.

System function	Effect
EditAlarm	Triggers the Loop-In-Alarm event for all selected alarms.
ClearAlarmBuffer	Deletes alarms from the alarm buffer on the HMI device.
ClearAlarmBufferProtoolLegacy	Function such as "ClearAlarmBuffer". This system function has been retained to ensure compatibility and uses the old ProTool numbering.
AlarmViewEditAlarm	Triggers the event Loop-In-Alarm for all alarms selected in the specified alarm view.
AlarmViewAcknowledgeAlarm	Acknowledges the alarms that are selected in the specified alarm view.
AlarmViewShowOperatorNotes	Displays the configured tooltip for the alarm selected in the specified alarm view.
AcknowledgeAlarm	Acknowledges all selected alarms.
ShowAlarmWindow	Hides or shows the alarm window on the HMI device.

10.3.4.2 System events

Basics on system events

System events

System events on the HMI device provide information about internal states of the HMI device and PLC.

The following overview illustrates when a system event occurs and how to eliminate the cause of error.

Note

HMI device dependency

Some of the system events described in this section apply to the individual HMI devices based on their scope of functions.

Note

System events are output in an alarm view. System events are output in the language currently set on your HMI device.

System event parameters

System events may contain encrypted parameters. The parameters are of relevance when troubleshooting because they provide a reference to the source code of the Runtime software. These parameters are output after the "Error code:" text.

30000 - Alarms errors when using system functions

Meaning of the system events

All system events that can be displayed are listed below. The system events are divided into different ranges.

Table 10-2 30000 - Alarms errors when using system functions

Number	Effect/causes	Remedy
30010	The tag could not accept the function result, e.g. when it has exceeded the value range.	Check the tag types of the system function parameters.
30011	A system function could not be executed because the function was assigned an invalid value or type in the parameter.	Check the parameter value and tag type of the invalid parameter. If a tag is used as a parameter, check its value.
30012	A system function could not be executed because the function was assigned an invalid value or type in the parameter.	Check the parameter value and tag type of the invalid parameter. If a tag is used as a parameter, check its value.

40000 - Linear scaling alarms

Meaning of the system alarms

All system alarms that can be displayed are listed below. The system alarms are divided into different ranges:

Table 10-3 40000 - Linear scaling alarms

Number	Effect/causes	Remedy
40010	The system function could not be executed since the parameters could not be converted to a common tag type.	Check the parameter types in the configuration.
40011	The system function could not be executed since the parameters could not be converted to a common tag type.	Check the parameter types in the configuration.

50000 - Data server alarms

Meaning of the system alarms

All system alarms that can be displayed are listed below. The system alarms are divided into different ranges:

Table 10-4 50000 - Data server alarms

Number	Effect/causes	Remedy
50000	The HMI device is receiving data faster than it is capable of processing. Therefore, no further data is accepted until all current data have been processed. Data exchange then resumes.	--
50001	Data exchange has been resumed.	--

70000 - Win32 function alarms

Meaning of the system events

All system events that can be displayed are listed below.

Table 10-5 70000 - Win32 function alarms

Number	Effect/causes	Remedy
70010	The application could not be started because it could not be found in the path specified or there is insufficient memory space.	Check whether the application exists in the specified path or close other applications.
70011	The system time could not be modified. The error alarm only appears in connection with area pointer "Date/time PLC". Possible causes: • An invalid time was transferred in the job mailbox. • The Windows user has no right to modify the system time. If the first parameter in the system event is displayed with the value 13, the second parameter indicates the byte containing the incorrect value.	Check the time which is to be set. Using Windows NT/XP: Users running WinCC Runtime must be granted the right to set the system time of the operating system.

Number	Effect/causes	Remedy
70012	Error when executing the function "StopRuntime" with the "Runtime and operating system" option. Windows and WinCC Runtime are not closed. The error was possibly generated because other programs cannot be closed.	Close all programs currently running. Then close Windows.
70013	The system time could not be modified because an invalid value was entered. Incorrect separators may have been used.	Check the time which is to be set.
70014	The system time could not be modified. Possible causes: • An invalid time was transferred. • The Windows user has no right to modify the system time. Windows rejects the setting request.	Check the time which is to be set. Using Windows NT/XP: Users running WinCC Runtime must be granted the right to set the system time of the operating system.
70015	The system time could not be read because Windows rejects the reading function.	--
70016	An attempt was made to select a screen by means of a system function or job. This is not possible because the screen number specified does not exist. Or: A screen could not be generated due to insufficient system memory. Or: The screen is blocked. Or: Screen call has not been executed correctly.	Check the screen number in the function or job with the screen numbers configured. Assign the number to a screen if necessary. Check the details for the screen call and whether the screen is blocked for specific users.
70017	Date/time is not read from the area pointer because the address set in the PLC is either not available or has not been set up.	Change the address or set up the address in the PLC.
70018	Acknowledgment that the password list has been successfully imported.	--
70019	Acknowledgment that the password list has been successfully exported.	--
70020	Acknowledgment for activation of alarm reporting.	--
70021	Acknowledgment for deactivation of alarm reporting.	--
70022	Acknowledgment to starting the Import Password List action.	--
70023	Acknowledgment to starting the Export Password List action.	--
70024	The range of values of the tag was exceeded in the system function. No calculation of the system function.	Check and correct the calculation.
70025	The range of values of the tag was exceeded in the system function. No calculation of the system function.	Check and correct the calculation.
70026	No other screens are stored in the internal screen memory. No other screens can be selected.	--
70027	The backup of the RAM file system has been started.	--

Number	Effect/causes	Remedy
70028	The files from the RAM have been copied in the Flash memory. The files from the RAM have been copied in the Flash memory. Following a restart, these saved files are copied back to the RAM file system.	--
70029	Backup of the RAM file system has failed. No backup copy of the RAM file system has been made.	Check the settings in the "Control Panel > OP" dialog and save the RAM file system using the "Save Files" button in the "Persistent Storage" tab.
70030	The parameters configured for the system function are faulty. The connection to the new PLC was not established.	Compare the parameters configured for the system function with the parameters configured for the PLCs and correct them as necessary.
70031	The PLC configured in the system function is not an S7 PLC. The connection to the new PLC was not established.	Compare the S7 PLC name parameter configured for the system function with the parameters configured for the PLC and correct them as necessary.
70032	The object configured with this number in the tab sequence is not available in the selected screen. The screen changes but the focus is set to the first object.	Check the number of the tab sequence and correct it if necessary.
70033	An e-mail cannot be sent because a TCP/IP connection to the SMTP server no longer exists. This system event is generated only at the first attempt. All subsequent unsuccessful attempts to send an e-mail will no longer generate a system event. The event is regenerated when an e-mail has been successfully sent in the meantime. The central e-mail component in WinCC Runtime attempts to connect to the SMTP server at cyclic intervals (1 minute) in order to transmit the remaining e-mails.	Check the network connection to the SMTP server and re-establish it if necessary.
70034	Following a disruption, the TCP/IP connection to the SMTP server could be re-established. The queued e-mails are then sent.	--
70036	No SMTP server for sending e-mails is configured. An attempt to connect to an SMTP server has failed and it is not possible to send e-mails. WinCC Runtime generates the system event after the first attempt was made to send an e-mail.	Configure an SMTP server: In the WinCC Engineering System using "Device settings > Device settings" In the Windows CE operating system using "Control Panel > Internet Settings > E-mail > SMTP Server"
70037	An e-mail cannot be sent for unknown reasons. The contents of the e-mail are lost.	Check the e-mail parameters (recipient etc.).
70038	The SMTP server has rejected sending or forwarding an e-mail because the domain of the recipient is unknown to the server or because the SMTP server requires authentication. The contents of the e-mail are lost.	Check the domain of the recipient address or disable the authentication on the SMTP server if possible. SMTP authentication is currently not used in WinCC Runtime.
70039	The syntax of the e-mail address is incorrect or contains illegal characters. The contents of the e-mail are discarded.	Check the e-mail address of the recipient.
70040	The syntax of the e-mail address is incorrect or contains illegal characters.	--

Number	Effect/causes	Remedy
70041	The import of the user management was aborted due to an error. Nothing was imported.	Check your user administration or download it again to the panel.
70042	The range of values of the tag was exceeded while executing the system function. The system function was not calculated.	Check and correct the calculation.
70043	The range of values of the tag was exceeded while executing the system function. The system function was not calculated.	Check and correct the calculation.
70044	An error occurred while sending the e-mails. The e-mails were not sent.	Check the SMTP settings and the error message in the system event.
70045	Cannot load a file required for encrypting the e-mail.	Update the operating system and Runtime.
70046	The server does not support encryption.	Select an SMTP server that supports encryption.
70047	The SSL versions of the HMI device and SMTP server may not be compatible.	Contact your network administrator or the operator of the SMTP server.

110000 - Offline function alarms

Meaning of the system events

All system events that can be displayed are listed below.

Table 10-6 110000 - Offline function alarms

Number	Effect/causes	Remedy
110000	The operating mode was changed. "Offline" mode is now set.	--
110001	The operating mode was changed. "Online" mode is now set.	--
110002	The operating mode was not changed.	Check the connection to the PLCs. Check whether the address range for the “Coordination" area pointer is present in the PLC.
110003	The operating mode of the specified PLC was changed by the "SetConnectionMode" system function. The "offline" operating mode is now set.	--
110004	The operating mode of the specified PLC was changed by the "SetConnectionMode" system function. The "online" operating mode is now set.	--
110005	An attempt was made to use the "SetConnectionMode" system function to set the specified PLC to "online" mode, although the entire system is in "offline" mode. This changeover is not allowed. The PLC remains in "offline" mode.	Switch the complete system to "online" mode and repeat execution of the system function.
110006	The content of the "project ID" area pointer does not match the project ID configured in WinCC. The WinCC Runtime is therefore terminated.	Check: • the project ID entered on the PLC. • the project ID entered in WinCC.

120000 - Trend alarms

Meaning of the system alarms

All system alarms that can be displayed are listed below.

Table 10-7 120000 - Trend alarms

Number	Effect/causes	Remedy
120000	The trend is not displayed because you configured an incorrect axis to the trend or an incorrect trend.	Change the configuration.
120001	The trend is not displayed because you configured an incorrect axis to the trend or an incorrect trend.	Change the configuration.
120002	The trend is not displayed because the tag assigned attempts to access an invalid PLC address.	Check whether the data area for the tag exists in the PLC, the configured address is correct and the value range for the tag is correct.

140000 - Connection alarms: Connection + device

Meaning of the system events

All system events that can be displayed are listed below.

Table 10-8 140000 - Connection alarms: Connection + device

Number	Effect/causes	Remedy
140000	An online connection to the PLC is established.	--
140001	The online connection to the PLC was shut down.	--
140003	No tag update or write operations are executed.	Check whether the connection is up and the PLC is switched on. In the Control Panel, check the set parameters using the "Set PG/PC interface" function. Restart the system.
140004	No tag update or write operations are executed due to an incorrect access point, or incorrect module configuration.	Check the connection and whether the PLC is switched on. Check the access point or the module configuration (MPI, PPI, PROFIBUS) in the Control Panel with "Set PG/PC interface". Restart the system.
140005	No tag update or write operations are executed due to an incorrect HMI device address (possibly too high).	Use a different address for the HMI device. Check the connection and whether the PLC is switched on. Check the parameters set in the "Set PG/PC interface" dialog of the Control Panel. Restart the system.
140006	No tag update or write operations are executed due to an incorrect baud rate setting.	Select a different baud rate in WinCC (according to module, profile, communication peer, etc.).

Number	Effect/causes	Remedy
140007	An incorrect bus profile prevents tag updates or write operations (see %1). The following parameters could not be written to the registry database: 1: Tslot 2: Tqui 3: Tset 4: MinTsdr 5: MaxTsdr 6: Trdy 7: Tid1 8: Tid2 9: Gap Factor 10: Retry Limit	Check the custom bus profile. Check the connection and whether the PLC is switched on. Check the parameters set in the "Set PG/PC interface" dialog of the Control Panel. Restart the system.
140008	An incorrect baud rate prevents tag updates or write operations. The following parameters could not be written to the registry database: 0: General error 1: Wrong version 2: Profile cannot be written to the registry database. 3: The subnet type cannot be written to the registry database. 4: The Target Rotation Time cannot be written to the registry database. 5: Incorrect Highest Station Address (HSA).	Check whether the connection is up and the PLC is switched on. In the Control Panel, check the set parameters using the "Set PG/PC interface" function. Restart the system.
140009	No tag updates or write operations because the S7 communication module was not found.	To reinstall the module, open the Control Panel and select "Set PG/PC interface".
140010	No S7 communication partner found because the PLC is shut down. DP/T: The option "PG/PC is the only master" is not set in the Control Panel under "Set PG/PC interface".	Switch on the PLC. DP/T: If only one master is connected to the network, disable "PG/PC is the only master" in "Set PG/PC interface". If several masters are connected to the network, enable these. Do not change any settings, for this will cause bus errors.
140011	No tag updates or write operations because communication is down.	Check the connection and whether the communication partner is switched on.
140012	Initialization error (e.g. if WinCC Runtime was closed in Task Manager). Or: Another application (e.g. STEP 7) with different bus parameters is active and the driver cannot be started with the new bus parameters (baud rate, for example).	Restart the HMI device. Or: Start WinCC Runtime and then start your other applications.
140013	The MPI cable is disconnected and, therefore, there is no power supply.	Check the connections.
140014	The configured bus address is in use by another application.	Change the HMI device address in the PLC configuration.
140015	Incorrect baud rate Or: Incorrect bus parameters (e.g. HSA) Or: OP address > HSA or: Incorrect interrupt vector (interrupt not registered by the driver)	Correct the parameters.

Number	Effect/causes	Remedy
140016	The hardware does not support the configured interrupt.	Change the interrupt number.
140017	The set interrupt is in use by another driver.	Change the interrupt number.
140018	SIMOTION Scout disabled the consistency check. Only a corresponding note appears.	In SIMOTION Scout, reactivate the consistency check and once again download the project to the PLC.
140019	SIMOTION Scout is downloading a new project to the PLC. Connection to the PLC is canceled.	Wait until the end of the reconfiguration.
140020	Mismatch of the PLC and project (FWX file) versions. The connection to the PLC is canceled.	The following remedies are available: Download the current version to the PLC using SIMOTION Scout. Recompile the project using WinCC ES, close WinCC Runtime, and restart with the new configuration.
140021	Setup of the connection to the PLC failed. Incorrect configuration of the "Access password" for the connection to the PLC.	Select the "Access password" area in the "Connections" editor to check the password entered for the connection to the PLC. Assign the correct password. The "Password" for the connection to the PLC is assigned in the "Security" area of the PLC properties.
140022	Setup of the connection to the PLC failed. Incorrect configuration of the access password for the connection to the PLC.	Select the "Access password" area in the "Connections" editor to check the password entered for the connection to the PLC. The "Password" for the connection to the PLC is assigned in the "Security" area of the PLC properties.
140025	Setup of the connection to the PLC failed. The access password of the PLC is disabled in the PLC display.	Enable the access password in the PLC display.

180000 - General alarms

Meaning of the system events

All system events that can be displayed are listed below.

Table 10-9 180000 - General alarms

Number	Effect/causes	Remedy
180000	A component/OCX received configuration data with a version ID which is not supported.	Install a newer component.
180001	System overload because too many actions are running in parallel. Certain actions can be executed, while others are discarded.	Several remedies are available: • Generate alarms at a slower rate (polling). • Initiate scripts and functions at greater intervals. If the alarm appears more frequently: Restart the HMI device.
180002	The screen keyboard could not be activated. Possible causes: "TouchInputPC.exe" was not registered due to faulty Setup.	Reinstall WinCC Runtime.

190000 - Tag alarms

Meaning of the system events

All system events that can be displayed are listed below.

Table 10-10 190000 - Tag alarms

Number	Effect/causes	Remedy
190000	It is possible that the tag is not updated.	--
190001	The tag is updated after the cause of the last error state has been eliminated (recovery of normal operation).	--
190002	The tag is not updated because communication with the PLC is down.	Select system function "SetOnline" to enable communication.
190004	The tag is not updated because the address configured for this tag does not exist.	Check the configuration.
190005	The tag is not updated because the configured PLC type does not exist for this tag.	Check the configuration.
190006	The tag is not updated because it is not possible to map the PLC type in the data type of the tag.	Check the configuration.
190007	The tag value is not modified because the connection to the PLC is interrupted or the tag is offline.	Set online mode or reconnect to the PLC.
190008	The configured tag limits were violated due to one of the following events: • Value input • System function • Script	Observe the configured or current tag limits.
190009	An attempt was made to assign this tag a value that is outside the valid range of values for this data type. For example, input of the value 260 for a byte tag, or input of the value -3 for an unsigned word tag.	Observe the range of values for the data type of the tags.
190010	The rate at which values are written to the tag is too high (e.g. initiated in a script loop). Values will be lost because buffer capacity is limited to 100 operations.	The following remedies are available: • Extend the interval between multiple write actions. • Do not use an array tag longer than 6 words when configuring an acknowledgment on the HMI device using "HMI acknowledgment tag".
190011	Possible cause 1: The value entered could not be written to the configured PLC tag because the high or low limit was exceeded. The system discarded the entry and restored the original value. Possible cause 2: The connection to the PLC was interrupted.	Note that the value entered must be within the range of values of the control tag. Check the connection to the PLC.

Number	Effect/causes	Remedy
190012	It is not possible to convert a value from a source format to a target format, for example: An attempt is being made to write a counter value that is outside the valid, PLC-specific range of values. A tag of the type Integer should be assigned a value of the type string.	Check the range of values, or the data type of the tag.
190013	You entered a string that exceeds the tag length. The string is truncated automatically to a valid length.	Always enter strings that do not exceed the valid tag length.

190100 - Area pointer alarms

190100 - Area pointer alarms

Number	Effect/causes	Remedy
190100	The area pointer is not updated because the address configured for this pointer does not exist. Type 1 Warnings 2 Errors 3 PLC acknowledgment 4 HMI device acknowledgment 5 LED image 6 Trend request 7 Trend transfer 1 8 Trend transfer 2 No.: Consecutive number displayed in WinCC ES.	Check the configuration.
190101	The area pointer is not updated because it is not possible to map the PLC type to the area pointer type. Parameter type and no.: see alarm 190100	--
190102	The area pointer is updated after the cause of the last error state has been eliminated (recovery of normal operation). Parameter type and no.: See alarm 190100.	--

200000 - PLC coordination alarms

200000 - PLC coordination alarms

Number	Effect/causes	Remedy
200000	Coordination is not executed because the address configured in the PLC does not exist/is not set.	Change the address or set up the address in the PLC.
200001	Coordination is canceled because the write access to the address configured in the PLC is not possible.	Change the address or set the address in the PLC at an area which allows write access.
200002	Coordination is not carried out at the moment because the address format of the area pointer does not match the internal storage format.	Internal error
200003	Coordination can be executed again because the last error state is eliminated (return to normal operation).	--
200004	The coordination may not be executed.	--
200005	No more data is read or written. Possible causes: • The cable is defective. • The PLC does not respond, is defective, etc. • System overload	Ensure that the cable is plugged in and the PLC is operational. Restart the system if the system alarm persists.

210000 - PLC job alarms

210000 - PLC job alarms

Number	Effect/causes	Remedy
210000	Jobs are not processed because the configured address does not exist/has not been set up in the PLC.	Change the address or set up the address in the PLC.
210001	Jobs are not processed because read/write access to the configured address is not possible in the PLC.	Change the address, or set up the address in a PLC area at which read/write access is possible.
210002	Jobs are not executed because the address format of the area pointer does not match the internal storage format.	Internal error
210003	The job buffer is processed again because the last error status has been eliminated (recovery of normal operation).	--
210004	The job buffer is possibly not going to be processed.	--
210005	A job mailbox with invalid number was initiated.	Check the PLC program.
210006	An error occurred while executing the job mailbox. As a result, the control job is not executed. Observe the next/previous system event.	Check the parameters of the control job. Recompile the configuration data.

220000 - WinCC communication driver alarms

220000 - WinCC communication driver alarms

Number	Effect/causes	Remedy
220001	The tag is not downloaded because write access to data type Bool/Bit is not supported by the sublevel communication driver/HMI device.	Change the configuration.
220002	The tag is not downloaded because write access to data type Byte is not supported by the sublevel communication driver/HMI device.	Change the configuration.
220003	The communication driver cannot be loaded as it is possibly not installed.	Install the driver by reinstalling WinCC Runtime.
220004	Communication is down and no update data is transferred because the cable is not connected or defective etc.	Check the connection.
220005	Communication is up.	--
220006	The connection between the specified PLC and the specified port is active.	--
220007	The connection to the specified PLC is interrupted at the specified port.	Check the following: • Is the cable plugged in? • Is the PLC OK? • Is the right port being used? • Is your configuration OK (port parameters, protocol settings, PLC address)? Restart the system if the system event persists.
220008	The communication driver cannot access or open the specified port. This port might be in use by another application, or a port that does not exist on the target device is being used. No communication with the PLC.	Close all applications that access the port and restart the computer. Use another port that exists in the system.

230000 - Screen object alarms

230000 - Screen object alarms

Number	Effect/causes	Remedy
230000	The value entered could not be used. The system discards this entry and restores the previous value. Possible causes: • The range of values is exceeded. • You entered invalid characters • The valid maximum number of users has been exceeded.	Enter a practical value, or delete a user that is no longer required.
230002	The user currently logged on does not have the necessary authorization, so the system discards the entry and restores the previous value.	Log on as user with appropriate authorization.
230003	Change to the specified screen failed because this screen is not available/configured. The current screen remains selected.	Configure the screen and check the selection function.
230005	The range of values of the tag has been exceeded in the I/O field. The original tag value is retained.	Observe the range of values for the tag when entering a value.
230100	During navigation in the Web browser, the system returned a message which may be of interest to the user. The Web browser continues to run but may not (fully) show the new page.	Navigate to another page.
230200	The HTTP channel connection was interrupted due to an error. This error is explained in detail by another system event. Data is no longer exchanged.	Check the network connection. Check the server configuration.
230201	The HTTP channel connection is set up. Data is exchanged.	--

Number	Effect/causes	Remedy
230202	WININET.DLL has detected an error. Usually, this error occurs if it is not possible to connect to the server, or the server denies a connection because the client lacks proper authorization. An unknown server certificate may also be the cause if the connection is encrypted by means of SSL. The alarm text provides details. This text is always in the language of the Windows installation because it is returned by the Windows OS. Process values are no longer exchanged. The part of the alarm returned by the Windows OS might not be displayed, e.g. "An error has occurred". WININET.DLL returns the following error: Number: 12055 Text:HTTP: <no error text available>."	Depending on the cause: When an attempt to connect fails, or a timeout occurs: • Check the network connection and the network. • Check the server address. • Check whether the WebServer is actually running on the target station. Incorrect authorization: • The configured user name and/or password do not match the entries on the server. Set consistent data. If the server certificate is rejected: Certificate signed by an unknown CA ( ): • Ignore this point, or install a certificate that has been signed with one of the root certificates known to the client station. The date of the certificate is invalid: • Ignore this point, or install a certificate with valid date on the server. Invalid CN (Common Name or Computer Name): • Ignore this point, or install a certificate with a name that corresponds to the server address.
230203	Although a connection can be made to the server, the HTTP server refused to connect. Possible causes: • WinCC Runtime does not run on the server • The HTTP channel is not supported (503 Service unavailable). Other errors can only occur if the Webserver does not support the HTTP channel. The language of the alarm text depends on the Webserver. Data is not exchanged.	On 503 Service unavailable error: Check whether WinCC Runtime is running on the server and whether the HTTP channel is supported.
230301	Internal error. An English text explains the error in more detail. The error may be caused by insufficient memory. OCX does not work.	--
230302	The name of the remote server cannot be resolved. An attempt to connect has failed.	Check the configured server address. Check whether the DNS service is available on the network.
230303	The remote server is not running on the addressed computer. incorrect server address. An attempt to connect has failed.	Check the configured server address. Check whether the remote server is running on the target computer.
230304	The remote server on the addressed computer is incompatible with VNCOCX. An attempt to connect failed.	Use a compatible remote server.
230305	Authentication has failed due to incorrect password. An attempt to connect failed.	Configure the correct password.

Number	Effect/causes	Remedy
230306	Error in the connection to the remote server. This may occur as a result of network problems. An attempt to connect failed.	Check whether the network cable is plugged in, or whether there are network problems.
230307	The connection to the remote server was shut down. Possible causes: • The remote server was shut down • The user instructed the server to close all connections. The connection is cancelled.	--
230308	This alarm provides information on the connection status. An attempt is made to connect.	--

260000 - Password system alarms

260000 - Password system alarms

Number	Effect/causes	Remedy
260000	An unknown user or an unknown password has been entered in the system. The current user is logged off from the system.	Log on to the system as a user with a valid password.
260001	The logged in user does not have sufficient authorization to execute the protected functions on the system.	Log on to the system as a user with sufficient authorization.
260002	This alarm output when the "TrackUserChange" system function is triggered.	--
260003	The user has logged off from the system.	--
260004	The user name entered into the user view already exists in the user management.	Select another user name because user names have to be unique in the user management.
260005	The entry is discarded.	Enter a shorter user name.
260006	The entry is discarded.	Use a shorter or longer password.
260007	The logoff time entered is outside the valid range from 0 to 60 minutes. The new value entered is discarded and the original value is retained.	Enter a logon timeout value between 0 and 60 minutes.
260008	An attempt was made in WinCC to read a PTProRun.pwl file created with ProTool V 6.0. Reading of the file was canceled due to incompatibility of the format.	--
260009	You have attempted to delete the user "Administrator" or "PLC User". These users are fixed components of the user management and cannot be deleted.	If you need to delete a user, because perhaps you have exceeded the maximum number permitted, delete another user.
260012	The password entries in the "Change Password" dialog and in the confirmation field do not match. The password is not changed. User will be logged off.	You have to log on to the system again. Then enter the identical password twice to be able to change the password.

Number	Effect/causes	Remedy
260013	The password entered in the "Change Password" dialog is invalid because it is already in use. The password is not changed. User will be logged off.	You have to log on to the system again. Then enter a new password that has not been used before.
260014	You have tried to log on with an incorrect password three times in a row. You will be locked out and assigned to group no. 0.	You can log on to the system with your correct password. Only an administrator can change the assignment to a group.
260024	The password you entered does not meet the necessary security guidelines.	Enter a password that contains at least one number.
260025	The password you entered does not meet the necessary security guidelines.	Enter a password that comprises at least three characters.
260028	Upon system start-up, an attempt to log on, or when trying to change the password of a SIMATIC log-on user, the system attempts to access the SIMATIC Logon Server. If attempting to log on, the new user is not logged in. If a different user was logged on before, then this user is logged off.	Check the connection to the SIMATIC Logon Server and its configuration; for example: 1. Port number 2. IP address 3. Server name 4. Functional transfer cable Or use a local user.
260030	The SIMATIC Logon user could not change his password on the SIMATIC Logon Server. The new password is possibly noncompliant with password rules set on the server, or the user is not authorized to change his password. The old password remains and the user is logged off.	Log in again and choose a different password. Check the password rules on the SIMATIC Logon Server.
260033	The action change password or log on user could not be carried out.	Check the connection to the SIMATIC Logon Server and its configuration; for example: 1. Port number 2. IP address 3. Server name 4. Functional transfer cable Or use a local user.
260034	The last logon operation has not yet ended. A user action or a logon dialog can therefore not be called. The logon dialog is not opened. The user action is not executed.	Wait until the logon operation is complete.
260035	The last attempt to change the password was not completed. A user action or a logon dialog can therefore not be called. The logon dialog is not opened. The user action is not executed.	Wait until the procedure is complete.
260036	There are insufficient licenses on the SIMATIC Logon Sever. The logon is not authorized.	Check the licensing on the SIMATIC Logon Server.

Number	Effect/causes	Remedy
260037	There is no license on the SIMATIC Logon Sever. A logon is not possible. It is not possible to log on via the SIMATIC Logon Server, only via a local user.	Check the licensing on the SIMATIC Logon Server.
260040	The system attempts to access the SIMATIC Logon Server upon system start-up or when trying to change the password. If attempting to log on, the new user is not logged in. If a different user was logged on before, then this user is logged off.	Check connection to the domain and its configuration in the Runtime security settings editor. Or use a local user.
260043	It was not possible to log the user on to the SIMATIC Logon Server. The user name or the password could be incorrect or the user does not have sufficient rights to log on. The new user is not logged in. If a different user was logged on before, then this user is logged off.	Try again. If necessary, check the password data on the SIMATIC Logon Server.
260044	It was not possible to log the user on to the SIMATIC Logon Server as his account is blocked. The new user is not logged in. If a different user was logged on before, then this user is logged off.	Check the user data on the SIMATIC Logon Server.
260045	The SIMATIC Logon user is not associated to any or several groups. The new user is not logged in. If a different user was logged on before, then this user is logged off.	Check user data on the SIMATIC Logon Server and the configuration in your WinCC project. A user may only be assigned to one group.

270000 - System alarms

270000 - System Alarms

Number	Effect/causes	Remedy
270000	The alarm does not indicate the tag because it is accessing an invalid address in the PLC.	Check whether the data area for the tag exists on the PLC, whether the configured address is correct, and whether the value range for the tag is correct.
270001	There is a device-specific limit as to how many alarms may be queued for viewing (see the operating instructions). This limit has been exceeded. The view no longer contains all the alarms. However, all alarms are written to the alarm buffer.	--
270002	The view shows alarms of a log for which there is no data in the current project. Placeholders are output for the alarms.	Delete old log data, if necessary.

Number	Effect/causes	Remedy
270003	The service cannot be set up because too many devices want to access this service. A maximum of four devices can execute this action.	Reduce the number of HMI devices which want to use the service.
270004	Access to the persistent alarm buffer is not possible. Alarms cannot be restored or backed up.	If the problems persist at the next restart, contact Customer Support (delete Flash).
270005	Persistent alarm buffer corrupted: Alarms cannot be restored.	If the problems persist at the next restart, contact Customer Support (delete Flash).
270006	Project modified: Alarms cannot be restored from the persistent alarm buffer.	The project was compiled and downloaded again to the HMI device. The error should no longer occur at the next restart of the HMI device.
270007	A configuration problem is preventing you from the restoring the data (e.g. a DLL was deleted, or unknown directory).	Update the operating system and download your project again to the HMI device.

290000 - Recipe system alarms

290000 - Recipe system alarms

Number	Effect/causes	Remedy
290000	The recipe tag could not be read or written. It is assigned the start value. The alarm can be entered in the alarm buffer for up to four more faulty tags. After that, alarm 290003 is output.	Check the configuration to see whether the address has been set up in the PLC.
290001	An attempt was made to assign the recipe tag a value that is outside the valid range of values for this type. The alarm can be entered in the alarm buffer for up to four more faulty tags if necessary. After that, alarm 290004 is output.	Observe the range of values for the tag type.
290002	It is not possible to convert a value from a source format to a target format. The alarm can be entered in the alarm buffer for up to four more faulty recipe tags if necessary. After that, alarm 290005 is output.	Check the range of values or type of the tag.
290003	This alarm is output after alarm 290000 was triggered more than five times. In this case, no further separate alarms are generated.	Check the configuration to see whether the tag addresses have been set up in the PLC.
290004	This alarm is output after alarm 290001 was triggered more than five times. In this case, no further separate alarms are generated.	Observe the range of values for the tag type.
290005	This alarm is output after alarm 290002 was triggered more than five times. In this case, no further separate alarms are generated.	Check the range of values or type of the tag.

Number	Effect/causes	Remedy
290006	The limits configured for the tag have been exceeded by the values entered.	Observe the configured or current tag limits.
290007	There is a difference between the source and target structure in the recipe currently being processed. The target structure contains an additional recipe tag that is not available in the source structure. The recipe tag specified is assigned its start value.	Insert the specified recipe tag into the source structure.
290008	There is a difference between the source and target structure in the recipe currently being processed.The source structure contains an additional recipe tag that is not available in the target structure and cannot be assigned. The value is discarded.	Remove the specified recipe tag in the specified recipe from the project.
290010	The storage location configured for the recipe is invalid. Possible causes: Invalid characters, write protection, data carrier out of space or not available.	Check the configured storage location.
290011	A data record of the specified number does not exist.	Check the source for the number (constant or tag value).
290012	A recipe of the specified number does not exist.	Check the source for the number (constant or tag value).
290013	An attempt was made to save a data record under a data record number that already exists. The operation is not executed.	The following remedies are available: • Check the source for the number (constant or tag value). • First, delete the data record. • Modify the "Overwrite" function parameter.
290014	The specified import file was not found.	Check the following: • The file name • Ensure that the file is in the specified directory.
290020	Check back to verify that the download of data records from the HMI device to the PLC has started.	--
290021	Check back to verify that the download of records from the HMI device to the PLC was completed.	--
290022	Check back to indicate that the download of data records from the HMI device to the PLC was canceled due to an error.	Check the following conditions in the configuration: • Are the tag addresses configured in the PLC? • Does the recipe number exist? • Does the data record number exist? • Is the "Overwrite" function parameter set?
290023	Check back to verify that the download of data records from the PLC to the HMI device has started.	--
290024	Check back to verify that the download of data records from the PLC to the HMI device was completed.	---

Number	Effect/causes	Remedy
290025	Check back to indicate that the download of data records from the PLC to the HMI device was canceled due to an error.	Check the following conditions in the configuration: • Are the tag addresses configured in the PLC? • Does the recipe number exist? • Does the data record number exist? • Is the "Overwrite" function parameter set?
290026	An attempt was made to read/write a data record that is not free at present. This error can occur if recipes were configured for download with synchronization.	Set the mailbox status to zero.
290027	Unable to connect to the PLC at present. As a result, the data record cannot be read or written. Possible causes: No hardware connection to the PLC (no cable plugged in, cable is defect), or the PLC is switched off.	Check the connection to the PLC.
290030	This alarm is output after you selected screen which contains a recipe view in which a data record is already selected.	Reload the data record from the storage location, or retain the current values.
290031	While saving, it was detected that a data record with the specified number already exists.	Overwrite the data record, or cancel the action.
290032	During the export of data records, a file with the specified name was found.	Overwrite the file, or cancel the process.
290033	Confirmation prompt before deleting data records.	--
290040	A data record error with error code %1 that cannot be described in more detail occurred. The action is canceled. It is possible that the mailbox was not installed correctly on the PLC.	Check the storage location, the data record, the "Data record" area pointer, and the connection to the PLC. Restart the action after a short waiting time. If the error persists, contact Customer Support. Forward the relevant error code to Customer Support.
290041	A data record or file cannot be saved because the storage location is out of sufficient space.	Delete files no longer required.
290042	An attempt was made to execute several recipe actions simultaneously. The last action is not executed.	Retrigger the action after a short waiting time.
290043	Confirmation prompt before saving data records.	--
290044	The database for the recipe was corrupted and will be deleted.	--
290050	A check back indicates that the export of data records was started.	--
290051	A check back indicates successful completion of the export of data records.	--
290052	A check back indicates that the export of data records was canceled due to an error.	Ensure that the structure of the data records at the storage location and the current recipe structure on the HMI device are identical.
290053	A check back indicates that the import of records was started.	--
290054	A check back indicates successful completion of the import of data records.	--

Number	Effect/causes	Remedy
290055	A check back indicates that the import of data records was canceled due to an error.	Ensure that the structure of the data records at the storage location and the current recipe structure on the HMI device are identical.
290056	Error when reading/writing the value in the specified row/column. The action was canceled.	Check the specified row/column.
290057	The tags of the recipe specified were toggled from "offline" to "online" mode. Each change of a tag in this recipe is now immediately transferred to the PLC.	--
290058	The tags of the specified recipe were toggled from "online" to "offline" mode. Modifications to tags in this recipe are no longer immediately transferred to the PLC but must be transferred there explicitly by downloading a data record.	--
290059	A check back indicates that the specified record was successfully saved.	--
290060	A check back indicates that the specified data record memory was cleared.	--
290061	A check back indicates that deletion of the data record memory was canceled due to an error.	--
290062	The data record number exceeds the maximum of 65536. This data record cannot be created.	Select another number.
290063	This occurs when you execute system function "ExportDataRecords" while the "Overwrite" parameter is set to "No". An attempt was made to save a recipe under a file name that already exists. The export is canceled.	Check the parameters of the "ExportDataRecords" system function.
290064	A check back indicates that the deletion of data records was started.	--
290065	A check back indicates successful completion of the deletion of data records.	--
290066	Confirmation prompt before deleting data records.	--
290068	Confirmation prompt for deletion of all recipe data records.	--
290069	Confirmation prompt for deletion of all recipe data records.	--
290070	The data record specified was not found in the import file.	Check the source of the data record number, or the data record name (constant, or tag value).
290071	When the editing data record values, you entered a value that is below the low limit of the recipe tag. The entry is discarded.	Enter a value within the recipe tag limits.
290072	When editing data record values, you entered a value that exceeds the high limit of the recipe tag. The entry is discarded.	Enter a value within the recipe tag limits.

Number	Effect/causes	Remedy
290073	An action (e.g. saving a record) failed for an unknown reason. The error corresponds to status alarm IDS_OUT_CMD_EXE_ERR in the large recipe view.	--
290074	While saving, a data record with the specified number but with different name was found.	Overwrite the record, change the record number or cancel the action.
290075	A data record of this name already exists. The data record is not saved.	Select a different data record name.
290110	The default values could not be set due to an error.	--
290111	The recipes subsystem cannot be used. Recipe views have no content and recipe-specific functions will not be executed. Possible causes: • Error when loading the recipes. • The recipe structure was changed in the ES. The recipes were not included in the latest project download. This means that the new configuration data no longer matches the old recipes on the device.	Download the project to the device again, including the recipes (the corresponding check box in the download dialog must be check marked).

10.3.5 Configuring system diagnostics

10.3.5.1 System diagnostics basics

Introduction

Using system diagnostics, you can display the messages from the diagnostic buffer of all integrated connections.

System diagnostics view

The system diagnostics display is an operating and display object that you use in a screen.

You navigate directly to the cause of the error and the associated connection. You have access to all integrated connections that you have configured in the "Devices & networks" editor.

10.3.5.2 System diagnostics views

Introduction

Three different views are available in the simple system diagnostic view.

- Device view
- Diagnostic buffer view
- Detail view

Device view

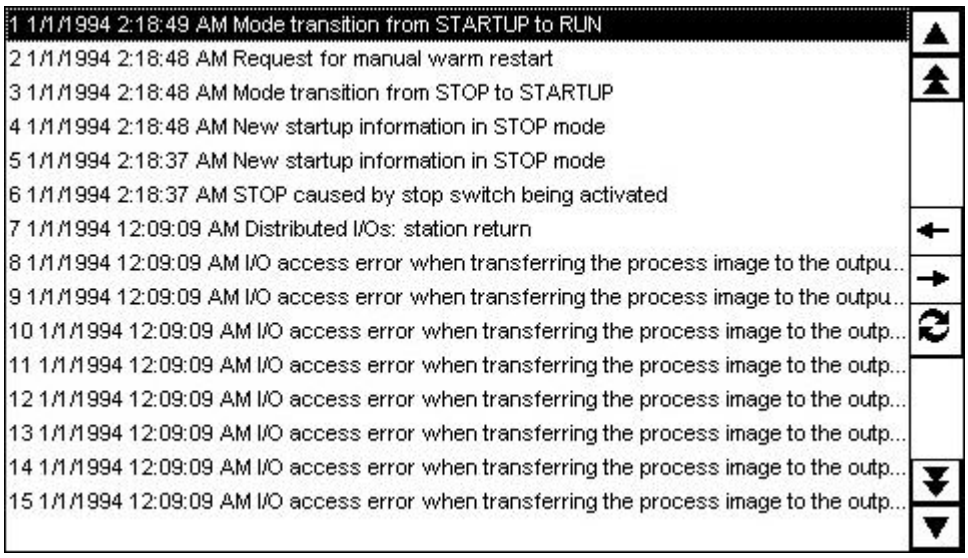
The device view is only displayed if more than one integrated connection has been configured.

The device view shows all available connections in a table. Double-clicking a connection opens the diagnostic buffer view.

Diagnostic buffer view

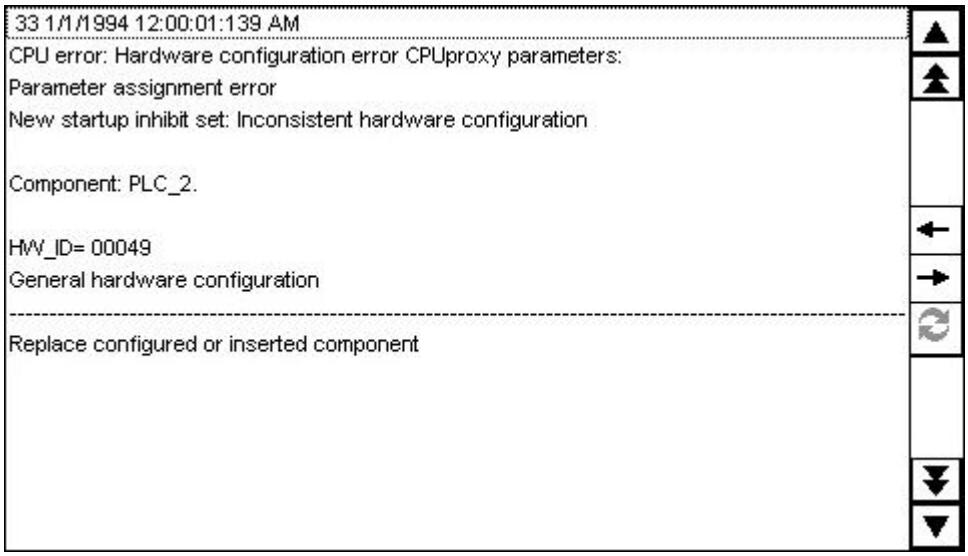
The diagnostic buffer view shows the current data from the diagnostic buffer.

To update the diagnostic buffer view, you click .



Detail view

The detail view gives detailed information about the selected connection and any pending errors. Check whether the data is correct in the detail view.



Note

Contents of the detail view

The contents of the detail view are only available for integrated connections with S7 1200 and S7 1500 PLCs.

Navigation buttons

Button	Key	Function
	Enter key	In the device view: Opens the diagnostic buffer view of the selected device. In the diagnostic buffer view: Opens the detail view.
	Esc key	In the diagnostic buffer view: Opens the device view. In the detail view: Opens the diagnostic buffer view.
	Configured function key, e.g. F1.	Updates the diagnostic buffer view.

See also

System diagnostics view (Page 8103)

Configuring the system diagnostic view (Page 8210)

10.3.5.3 Configuring the system diagnostic view

Introduction

For an overview of all integrated connections, you insert a system diagnostics view in your project.

Requirements

- A PLC has been created.
- A Basic Panel has been created.
- An integrated connection has been created in the "Devices & Networks" editor.
- You have created a screen.
- The Inspector window is open.

Procedure

1. Double-click the "System diagnostics view" object in the "Tools" task card. The object is added to the screen.
2. In the Inspector window, select "Properties > Layout".
3. Enter a number under "Lines per entry", i.e. 5.

Result

The system diagnostics view has been added to the screen.

To obtain the latest alarms, update the diagnostic buffer.

See also

System diagnostics views (Page 8208)

10.4 Working with recipes

10.4.1 Basics

10.4.1.1 Definition and applications

Introduction

Related data, e.g. machine parameter assignments or production data, are combined in recipes.

Examples:

- Machine parameter settings that are needed to convert production to a different product variant.
- Product components that result in different compositions for different end products.

A recipe has a fixed data structure. The structure of a recipe is defined in the configuration. A recipe contains recipe data records. These differ in terms of their values, but not their structure.

Recipes are saved on the HMI device. A recipe data record is always transferred completely and in a single pass between the HMI device and the PLC.

Note

Restrictions in the import/export

It is not possible to export or import the recipes for Basic Panels.

Complete recipe data but not individual recipe data records can be exported and imported with ProSave to the CSV format and transmitted to the HMI device. Runtime is stopped in the meantime.

Using recipes

Recipes can be used in the following situations:

- Manual production
  You select the required recipe data and display it on the HMI device. You modify the recipe data as required and save it on the HMI device. You transfer the recipe data to the PLC.
- Automatic production
  The control program starts transfer of the recipe data between the PLC and HMI device. You can also start the transfer from the HMI device. Production is then implemented automatically. It is not essential to display or modify the data.
- Teach-in mode
  You optimize production data that was optimized manually on the system, e.g. axis positions or filling volumes. The values thus determined are transferred to the HMI device and saved in a recipe data record. You can then transfer the saved recipe data back to the PLC at a later date.

Entering and modifying the recipe data

You enter the data in the individual recipe data records and modify it as required. The following options are available:

- Data entry during configuration
  If the production data exists already, you enter the data in the "Recipes" editor during recipe configuration.
- Entering the data in Runtime
  If you have to frequently modify production data, you can do this directly in Runtime as follows:
  - Enter the data directly on the HMI device.
  - Set the parameters directly on the machine. You then transfer the data from the PLC to the HMI device and save it in the recipe.

10.4.1.2 Examples for using recipes

Recipes are used in the manufacturing industry and mechanical engineering, for example. The following recipes show typical applications which you can implement with the recipe function of WinCC:

- Machine parameter assignment
  One field of application for recipes is the assignment of machine parameters in the manufacturing industry: A machine cuts wooden boards to a certain size and drills holes. The guide rails and drill have to be moved to new positions according to the board size. The required position data are stored as data records in a recipe. You reassign the machine parameters using "Teach in" mode if, for example, a new board size is to be processed. You transfer the new position data directly from the PLC to the HMI device and save it as a new data record.
- Batch production
  Batch production in the food processing industry represents another field of application for recipes: A filling station in a fruit juice plant produces juice, nectar, and fruit drinks in a variety of flavors. The ingredients are always the same, differing only in their mixing ratios. Each flavor corresponds to a recipe. Each mixing ratio corresponds to a data record. All of the required data for a mixing ratio can be transferred to the machine control at the touch of a button.

10.4.1.3 Structure of recipes

Introduction

The basic structure of a recipe is illustrated with reference to the filling station in a fruit juice plant.

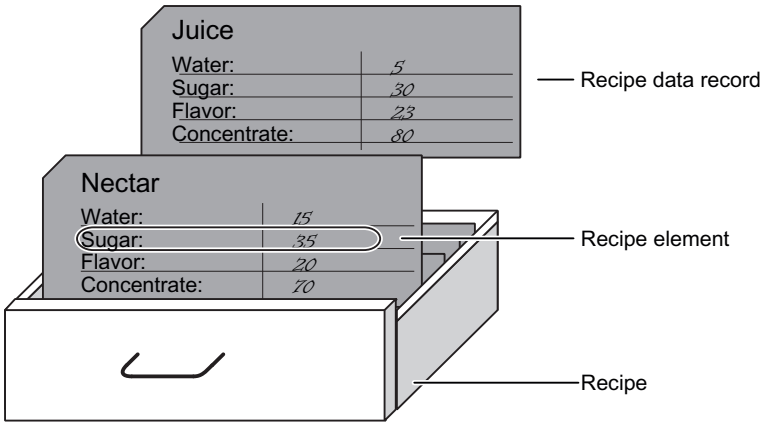
There may be several different recipes in an HMI device. A recipe can be compared to an index card box that contains several index cards. The index card box contains several variants for manufacturing a product family. All the data for each manufacturing variant is contained on a single index card.

Example:

In a soft drinks production plant, a recipe is needed for different flavors. Drink variants include fruit juice drink, juice and nectar.

Recipe

The recipe contains all the recipe data records for the different drink variants.



Recipe data records

Each index card represents a recipe data record needed to manufacture a product variant.

Recipe entries

Each index card in a drawer has the same structure. All the index cards contain fields for the different ingredients. Each field corresponds to a recipe entry. All the records of a recipe thus contain the same entries. The records differ, however, in the value of the individual entries.

Example:

All the drinks contain the following ingredients:

- Water
- Concentrate
- Sugar
- Flavoring

The records for juice drink, fruit juice or nectar differ, however, in the quantity of sugar used in production.

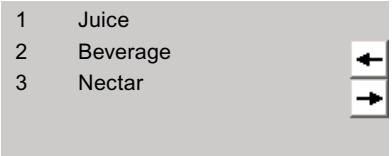
10.4.1.4 Displaying recipes

Introduction

You need to configure the recipe view to display recipes. You can change the values of a recipe in the recipe view and thereby influence the manufacturing process or a machine.

Recipe view

The recipe view is an off-the-shelf WinCC display and operator control for managing recipe data records. The recipe view is always part of a screen. The recipe view shows recipe data records in tabular form. You adapt the appearance and the possible operations to suit your specific needs.



If you are editing recipes with a recipe view in your project, the values are saved in recipe data records. The values are not transferred between the HMI device and PLC until you use the relevant operator control.

10.4.1.5 Flow of data for recipes

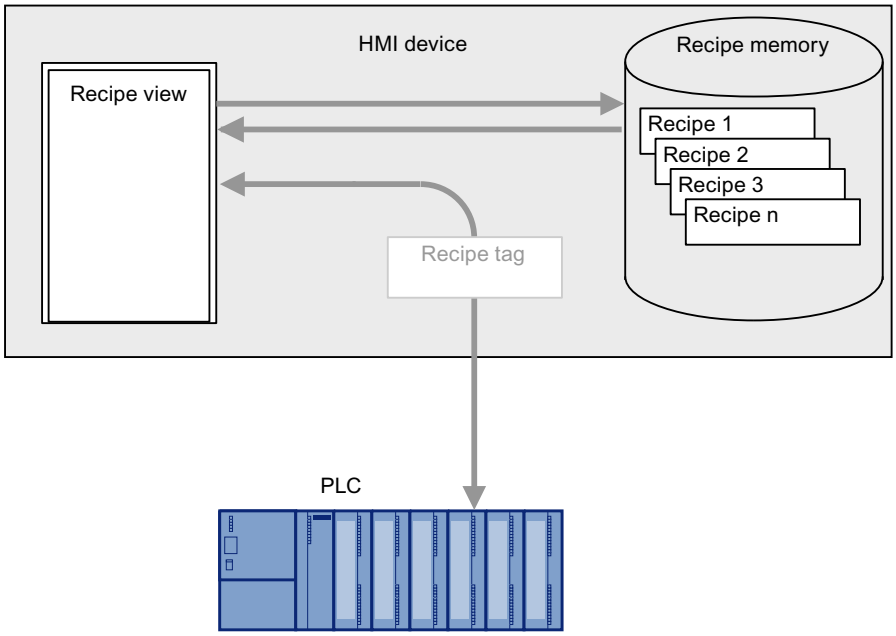
Interaction between the components

There is interaction between the following components at runtime:

- Recipe view
  Recipes are displayed and edited in the recipe view on the HMI device.
  The recipe data records from the internal memory of the HMI device are displayed and edited in the recipe view.
- HMI device recipe memory
  Recipes are saved in the form of recipe data records in the HMI device's recipe memory.
- Recipe tags
  The recipe tags contain recipe data.

Overview of the flow of data

The following figure illustrates the flow of data in recipes:



To transfer recipe data records to the PLC, use the "To PLC" button in the recipe view or an operator control with the system function "RecipeViewSetDataRecordToPLC".

Data are exchanged with the PLC by recipe tags. On Basic Panels you cannot use recipe tags outside a recipe, e.g. not in I/O fields.

10.4.1.6 Synchronization of recipe data records with the PLC

Overview

When recipe data records are transferred between the HMI device and PLC, both communication peers access common communication areas on the other peer.

Recipe data records are always transferred directly. The values of the tags are written directly to or read directly from the configured addresses without being placed on the clipboard.

Data transfer types

There are two ways to transfer recipe data records between the HMI device and PLC:

- Transfer without coordination
- Coordinated transfer via the "Data record" area pointer.

Note

Coordinated transfer

Transfer with coordinated transfer is used to prevent the uncontrolled overwriting of data in either direction in your control program.

Requirements for coordinated transfer

The following requirements apply to coordinated transfer:

- The "Data record" area pointer must be set up for the required connection in the "Communication > Connections" editor.
- In the properties of the recipe "Coordinated transfer of data records" is activated.
- The connection to the PLC is specified in the properties of the recipe with which the HMI device coordinates the transfer.

Coordinated transfer

In the case of coordinated transfer, both the PLC and the HMI device set the status bits in the shared data compartment.

Coordinated transfer of recipe data records can be a useful solution in the following cases:

- The PLC is the "active partner" for the transfer of recipe data records.
- The PLC evaluates information about the recipe number and name, as well as the recipe data record number and name.
- The transfer of recipe data records is started by the following PLC jobs:
  - "Set_data_record_in_PLC"
  - "Get_data_record_from_PLC"

10.4.2 Elements and basic settings

10.4.2.1 "Recipes" editor

Introduction

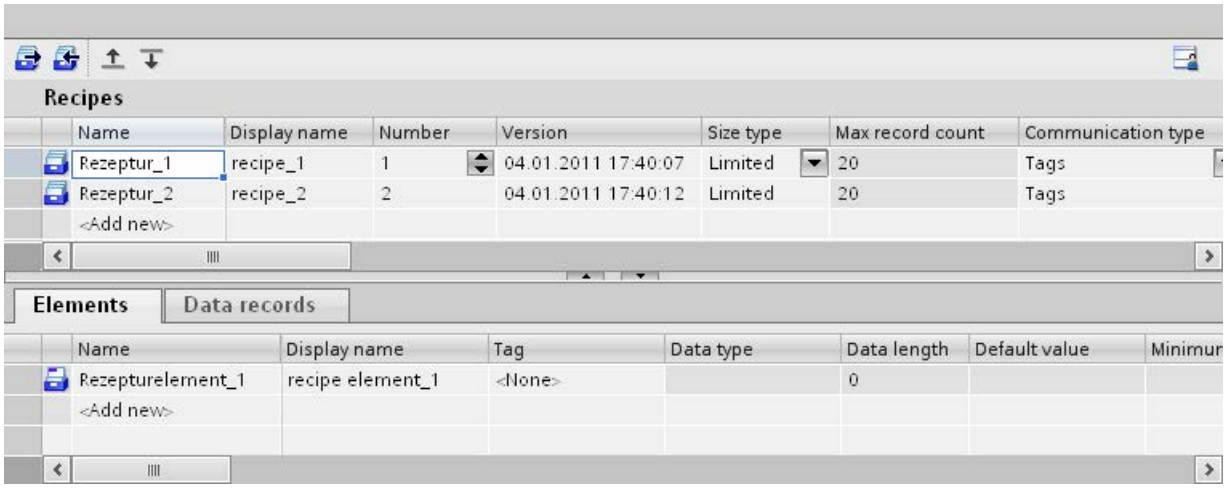
You can create, configure and edit recipes, recipe entries and recipe data records in the "Recipes" editor. The "Recipes" editor also allows you to enter values in recipe data records.

Structure of the "Recipes" editor

You create recipes in the top part of the table editor. You can also configure them there or in the Inspector window.

The bottom part of the table editor has the following tabs:

- Elements
  Define the recipe elements of the selected recipe using the table cells provided here. You can move recipe elements within the table with the shortcut menu commands, "Up" and "Down".
- Data records
  Define the values of the data records of the selected recipe using the table cells provided here.



You can then configure the selected recipe, the recipe element or the recipe data record in the Inspector window. You will find further notes on configuring the components of a recipe under "Configuring Recipes".

Recipe settings

The following settings are available for recipes:

Setting	Description
Name of the recipe	This is a unique identification for the recipe within the HMI device.
Display name	Appears in the recipe view, for example, in Runtime. You can configure display names in multiple languages. Assign descriptive names or designations which the operator can associate directly with a recipe, e.g. "fruit juice drink".
Recipe number	This is a unique identification for the recipe within the HMI device.
Version	Information about the recipe. The date and time of the last change to the recipe is set by default.
Path	Defines the storage location for recipes. The recipes are stored as a file.
Size type [fixed]	The recipe data records are limited to a predetermined number by default.
Number of data records [fixed]	Maximum number of data records in a recipe in Runtime. The number is limited by the recipe memory of the HMI device.

Setting	Description
Communication type [fixed]	The recipe data records are written directly to the addresses of the recipe tags and read from there.
Tooltip	Tooltip for the recipe which is shown to the operator in Runtime.

Note

Path

The storage location depends on the storage media available on the HMI device.

Basic Panels and OP77A, TP177A (Portrait)

These HMI devices have no external memory. Recipes are always saved in the internal Flash memory. The "Path" setting is therefore not available.

Recipe element settings

You can make the following settings on the "Elements" tab:

Setting	Description
Name of the recipe element	Identifies a recipe element uniquely within the recipe. Enter meaningful names or labels that you can allocate uniquely, such as axis labels on a machine or ingredients such as "Flavoring".
Display name	Appears in the recipe view, for example, in Runtime. You can configure display names in multiple languages. Assign meaningful names or designations which the operator can associate directly, e.g. "fruit juice flavoring".
Recipe tag	An assigned tag in Runtime stores the current value of the recipe element in the recipe data record.
Data type	Data type of the recipe tag.
Data length [fixed]	Data length of the recipe tag, depending on the data type.
Text list	Text is assigned to a value or range of values in a text list. You can display this text in an output field, for example. The assigned recipe tag must have the data type of a number. The tag value must be within the range of values of the text list.
Default value	This is used as the default entry when you create a new recipe data record.
Minimum value [fixed]	The smallest representable value of a number-based recipe tag, depending on the type of data.
Maximum value [fixed]	The largest representable value of a number-based recipe tag, depending on the type of data.
Decimal places	Determines how many places a decimal number is rounded to, e.g. 3 decimal places and vice versa by what power of ten an integer value is multiplied, e.g. 1,000.
Tooltip	Tooltip about the recipe element which is shown to the operator in Runtime.

Recipe data record settings

You can make the following settings on the "Data records" tab:

Setting	Description
Name of the recipe data record	Identifies a recipe data record uniquely within the recipe.
Display name	Appears in the recipe view, for example, in Runtime. You can configure display names in multiple languages. Assign meaningful names or product numbers which the operator can associate directly with a product, e.g. "yellow fruit juice E231".
Recipe data record number	Identifies a recipe data record uniquely within the recipe.
Recipe elements 1 to n	You can store various values for each recipe element even during configuration. Together with the values of the other recipe elements, a value always forms a recipe data record. You can store multiple recipe data records. If enabled in the transfer settings, the recipe data records are transferred to the HMI device when downloading the project and existing data records on the HMI device are overwritten.
Comment	Comment about the recipe data record

10.4.3 Displaying and editing recipes in Runtime

10.4.3.1 Simple recipe view

Recipe view

The simple recipe view is a ready-made display element and operator control that is used to manage recipe data records. The recipe view shows recipe data records in tabular form.

The displayed buttons and information in the columns are adjustable.

The values displayed or entered in the recipe view are saved in recipe data records. The displayed recipe data record can be written into the PLC by buttons or values can be read in from the PLC.

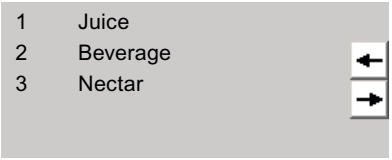
Layout of the display

The simple recipe view consists of three areas:

- Recipe list
- Data record list
- Element list

In the simple recipe view, each area is shown separately on the HMI device. Depending on the configuration, the simple recipe view starts with the recipe list.

The figure below shows an example of the data record list.



Display of values

Note

Processed recipe data record is changed in the background

Only applies for Basic Panels: if an operator has changed a recipe data record and a PLC job wants to read or write any recipe data record of this recipe, the PLC job is stopped and a system alarm is output. On the other hand, the changed value is displayed immediately if only the PLC job and no operator has changed recipe data.

Does not apply for Basic Panels: If an operator has changed a recipe data record and a PLC job has changed the values of the recipe data record concerned, the recipe view is not updated automatically. To update the recipe view, reselect the respective recipe data record.

See also

Recipe view (Page 8097)

10.4.3.2 Behavior of the recipe view in Runtime

Screen change

If you change to another screen and have not yet saved changes to the recipe data in the recipe view, you will be prompted to save the recipe data. The recipe name and the name of the recipe data record are displayed to show which recipe data have not been saved yet.

Create, change, copy or delete recipe data records

If you attempt to create a new recipe data record and a recipe data record already exists, a system alarm will appear on screen.

Operating the recipe view with function keys

The Recipe view can be operated with function keys, e.g. if the HMI device does not have touch functionality. You can assign functions such as "SaveDataRecord" to the function keys on the HMI device.

Display after import of recipe data

Note

Availability

Import and export of recipe data is not available for Basic Panels and OP77A, TP177A (Portrait).

If you open the recipe view during the import of recipe data, only the recipe data that is already completely imported will be displayed. The recipe view is not automatically updated with a data import. In order to have a complete view of all the recipe data, do not open the recipe view until the system prompts you that the recipe data has been imported successfully. Alternatively, update the recipe view after successful completion of the import procedure.

Updating tag for recipes and recipe data records

Note

Availability

Tags for recipes and recipe data records are not available for Basic Panels and OP77A, TP177A (Portrait).

The current recipe data record or its number can be saved to a tag, depending on the configuration. The tag will be updated under the following conditions:

- The recipe data record has been loaded.
- The screen with the recipe view was not exited during loading.

This operation may take some time.

10.4.4 Configuring recipes

10.4.4.1 General configuration procedure

Carry out the following configuration steps when you create a new recipe:

Step	Description
1	Define the structure of the recipe.
2	Create tags according to the recipe structure. Assign process names to these tags.
3	Create the recipe.

Step	Description
4	Enter the required properties for the recipe: • Language-dependent view name of the recipe • "Coordinated transfer of data records" option Not for Basic Panels: • Recipe storage location • "Synchronize recipe view and recipe tags" option • "Manual transfer of individual modified values (teach-in mode)" option
5	Create the recipe elements and enter the required properties: • Language-dependent view name of the recipe elements • Tag binding of the recipe elements • Standard values and decimal places (power of ten) for the recipe elements
6	Create the recipe data records. Enter the language-specific display names for the recipe data records.
7	Configure a screen with recipe view or a recipe screen.

Note

Basic Panels and OP77A, TP177A (Portrait)

The selection of the storage location is not available for these devices. The recipes are always saved in the internal Flash memory.

Recipe tags cannot be used outside a recipe, e.g. not in I/O fields, not in alarms as trigger tags, not in systems functions as parameters, etc.

Note

Restrictions recipe view and recipe image

Only the simple recipe view is available in Basic Panels and OP77A, TP177A. Recipe images are not available in Basic Panels and OP73, OP77A, TP177A (Portrait).

10.4.4.2 Creating and Editing Recipes

Creating a new recipe

Introduction

To create a complete recipe, start by creating a new recipe, assign the corresponding recipe elements and then define the associated values in a recipe data record.

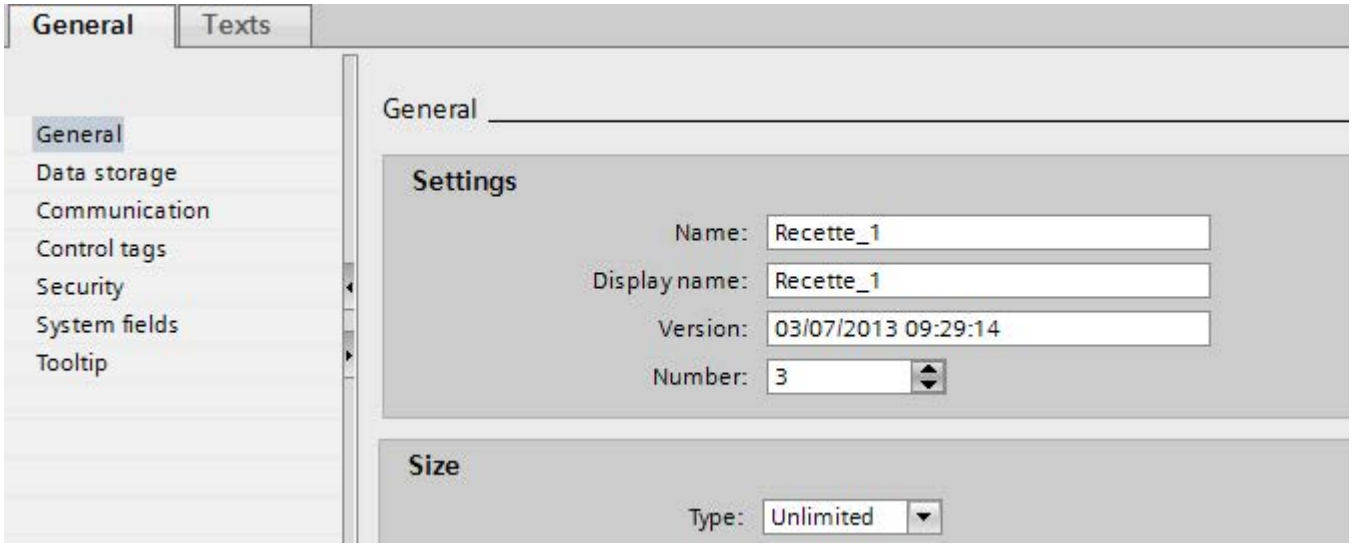
Requirement

- The tags for the recipe have been created.
- The "Recipes" editor is open.

Create recipe

Create a recipe as follows:

1. Click "Add" in the first free row of the table in the "Recipes" editor.
   The new recipe is created and displayed on a line.



2. Enter a descriptive name for the recipe under "Name" in the "General" area.
   This name identifies the recipe unambiguously within the project.
3. Select "Display name" to enter the language-specific name to be displayed in runtime.
4. Select a recipe number in "Number".
   The number identifies the recipe unambiguously within the HMI device.
   The recipe is automatically assigned a version that indicates the date and time of the last change. As an alternative, you can enter specific information relating to the recipe.
5. Specify the storage location for recipe data records in "Data medium". The options offered depend on the specific HMI device used.

   Note

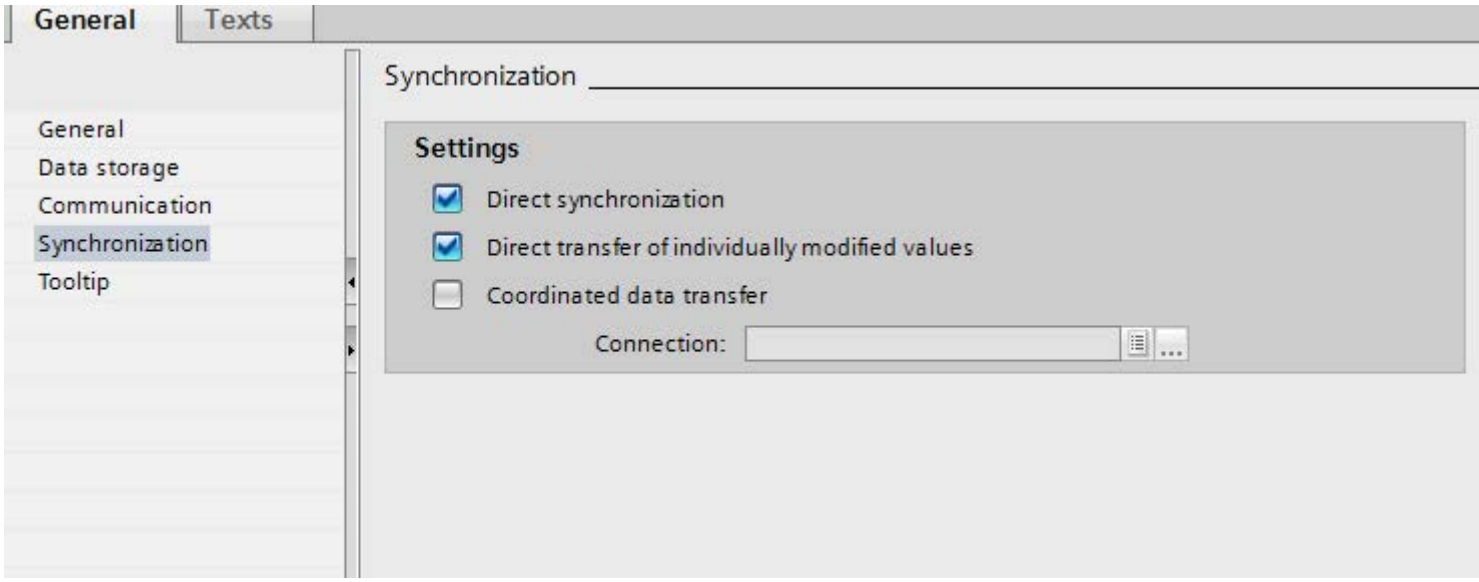
   Basic Panels and OP77A, TP177A (Portrait)

   The selection of the storage location is not available for these devices. The recipes are always saved in the internal Flash memory.

   Recipe tags cannot be used outside a recipe, e.g. not in I/O fields, not in alarms as trigger tags, not in systems functions as parameters, etc.

6. Enter a tooltip that is shown to the operator in runtime.

7. To compare recipe tags which are configured in I/O fields with the recipe view in Runtime, activate "Synchronize recipe view and recipe tags" in the Inspector window under "Properties > Synchronization".



Note

Basic Panels and OP77A, TP177A (Portrait)

Because the recipe tags cannot be additionally used in I/O fields in screens for Basic Panels, the "Synchronize recipe view and recipe tags" is not available; you will also not be able to use the "Manual transfer of individual modified values (teach-in mode)" option.

8. Deactivate "Manual transfer of individual modified values (teach-in mode)" to specify that the recipe tags are automatically transferred to the PLC when editing the I/O fields.
9. Activate "Coordinated transfer of data records" to monitor the transfer of recipe data in Runtime using area pointers.
10. Select the appropriate connection to the PLC for coordinated transfer under "Synchronize with".

Create recipe element

To create recipe elements, proceed as follows:

1. Click the "Elements" tab.
2. Click "Add" in the first free line of the table editor.
   A new recipe element is created.
3. Enter a descriptive name for the element under "Name".
   The name identifies the element uniquely within the recipe.
4. Enter a language-specific display name for the element under "Display name".
   The display name appears in the recipe view, for example, in runtime.

5. Select the tag you want to link to the recipe element under "Tag".
   The value of the recipe data element is saved in Runtime in this tag, which is stored in a recipe data record.

Elements | Data records

Name	Display name	Tag	Default value	Decimal places	Infotext
Water	Water	LitreWater	0	0	
<Add new>					

6. Enter a tooltip.
   The tooltip is shown to the operator in Runtime.
7. Under "Default value", enter the value that you want to use as the default entry when you create a new recipe data record.
8. To assign text to a value or range of values, select the relevant text list here. The assigned recipe tag must have the data type of a number. The tag value must be within the range of values of the text list.
   The text stored in the text list is displayed in an output field, for example, in Runtime.
9. Determine exactly how many places a decimal number is rounded to in the "Decimal places" column, e.g. 3 decimal places and vice versa by what power of ten an integer value is multiplied, e.g. 1,000.
   Examples for 3 decimal places: Entering "5" for a recipe element with the "Integer" data type gives the value "5000". Entering "5.6789" for a recipe element with the "Real" data type gives the value "5.679".
10. Create as many recipe entries as needed for the recipe. The maximum number of recipe entries possible depends on the HMI device being used.

Elements | Data records

Name	Display name	Tag	Default value	Decimal places	Infotext
Water	Water	LitreWater	0	0	
Concentrat	Concentrat	LitreConcentrat	0	0	
Sugar	Sugar	KiloSugar	0	0	
Aroma	Aroma	GramAroma	0	0	
<Add new>					

Create recipe data record with known recipe values

To create recipe elements, proceed as follows:

1. Click the "Data records" tab.
2. Click "Add" in the first free line of the table editor.
   A new recipe data record is created. The recipe data record has a separate column for every recipe element created in the recipe.

Elements | Data records

Name	Display name	Number	Water	Concentrat	Sugar	Aroma	Comment
Recipe_data_record_1	Recipe_data_record_1	1	0	0	0	0	
<Add new>							

3. Enter a descriptive name under "Name".
   The name identifies the data record uniquely within the recipe.
4. Enter a language-specific name under "Display name".
   The display name appears in the recipe view, for example, in runtime.
5. Enter a recipe data record number under "Number".
   The recipe data record number identifies the recipe data record uniquely within the recipe.
6. If you already know the recipe values at the configuration stage, you can enter the relevant value for each recipe element.

Elements | Data records

Name	Display name	Number	Water	Concentrat	Sugar	Aroma	Comment
Beverage	Beverage	1	30	70	45	600	
<Add new>							

7. Create as many data records as you need for the recipe.

Elements | Data records

Name	Display name	Number	Water	Concentrat	Sugar	Aroma	Comment
Beverage	Beverage	1	30	70	45	600	
Nectar	Nectar	2	50	50	10	300	
Juice	Juice	3	5	95	3	100	
<Add new>							

Enter the values in runtime

The following options are available for entering values in the recipe data records at runtime:

- Transfer data directly from the PLC (Teach-in mode)
- Import of values from a CSV file
- Input values on the HMI device

Note

Basic Panels and OP77A, TP177A (Portrait)

The import of values is not available for these devices.

Result

The complete recipe is configured.

Recipe data records with date or time stamp

If you use date or time data, make sure that the system setting for time and date on the configuring computer match those on the target system. Example: You load a recipe data record on the target system at 13:55 in which 14 h is stored as the processing time. If it is already 14:05 on the target computer, the recipe will not be processed. If an operator processes the recipe, change information will not written back correctly into the database.

After loading to the target system, check the recipes with date or time stamps on the target system.

Editing a recipe

Purpose

You want to modify, extend or delete parts of a recipe.

Requirement

- You have created at least one recipe.
- The "Recipes" editor is open.

Changing recipe settings

To change the recipe settings, proceed as follows:

1. Select the recipe that you want to change in the "Recipes" editor.
   The Inspector window opens.
2. Change the recipe configuration in the Inspector window.

You change recipe elements and recipe data records in the same way.

Change recipe values

To change recipe values, proceed as follows:

1. Select the recipe whose values you want to change.
2. Click the "Data records" tab.
3. Enter the new values in the value columns.

Adding a recipe element

To add more recipe elements to a recipe, proceed as follows:

1. Select the recipe to which you want to add more elements in the "Recipes" editor.
2. Click the "Elements" tab.
3. Click "Add" in the first free line.
   The recipe element is created.
4. Configure the recipe element.

You add recipe data records in the same way.

Managing recipes

Requirement

- You have created a recipe with recipe elements and recipe data record.
- The "Recipes" editor is open.

Renaming recipes

We distinguish between internal names and display names for recipes, recipe entries and recipe data records.

To rename recipe elements, proceed as follows:

1. Select the recipe that you want to rename.
   The Inspector window opens.
2. Select the "Rename" command from the shortcut menu.
3. Enter the new name.
   You rename recipe elements and recipe data records on the relevant tab in the same way.

Note

The view names in the "Recipes" editor can also be renamed under "Languages & Resources > Project Texts". This possibility is useful when you have already configured in several languages for example.

Copying and pasting recipes

To copy and paste recipes, proceed as follows:

1. Select the recipe that you want to copy.
2. Select the "Copy" command from the shortcut menu.
3. Select the "Paste" command from the shortcut menu in the first free table row.

The copied recipe is pasted into the table. The recipe elements and recipe data records are also copied in the appropriate tab with the recipe.

You also copy the recipe elements and recipe data records on the appropriate tab in the same way.

If a recipe data record of the same name already exists, the name of the copied recipe data record is extended by one digit. This ensures that the name is unique. Recipe data records can only be copied or pasted within the same recipe.

Deleting a recipe

To delete a recipe, proceed as follows:

1. Select the recipe that you want to delete.
2. Select the "Delete" command from the shortcut menu.
   The recipe is deleted.

You delete recipe elements and recipe data records on the relevant tab in the same way.

Note

When a recipe is deleted, the recipe data records contained in the recipe are also deleted.

Note

When you delete a recipe element, the associated values in the recipe data records are also deleted. The assigned tags are retained.

10.4.4.3 Configuring the display of recipes

Configuring the simple recipe view

Requirement

- The recipe has been created.
- The "Screens" editor is open.
- The screen has been created and opened.

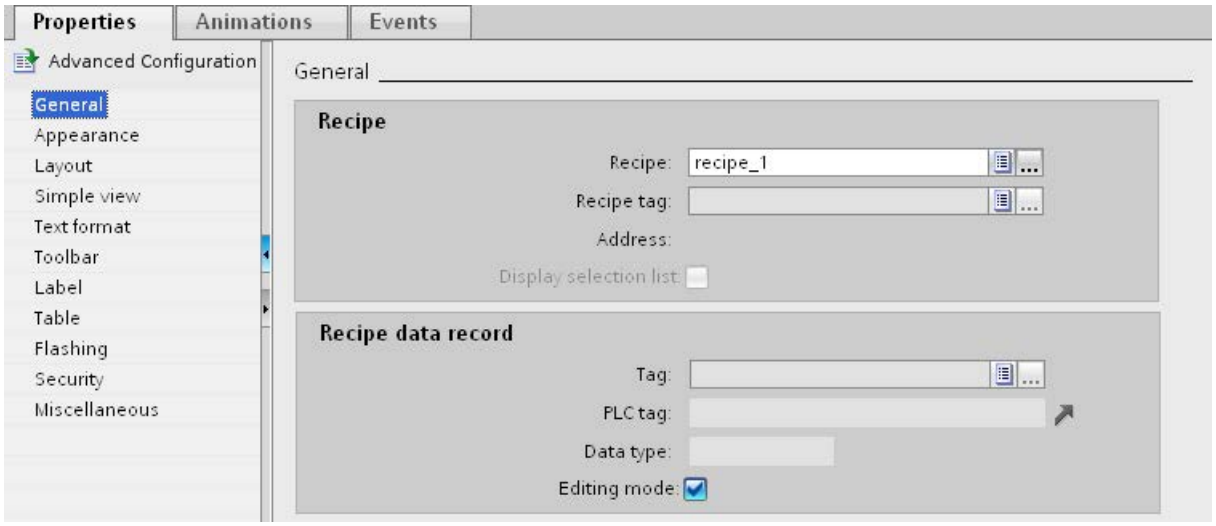
NOTICE
Data loss with several recipe views in the screen Applies only to Basic Panels, OP73, OP77A, TP177A and TP177A (Portrait): If two or more recipe views show the same recipe in a screen, you have a conflict when accessing the data. The result is data loss and unpredictable status of recipe data. Make sure the operators do not select and edit the same recipe in different recipe views. • Display only one recipe in a recipe view. • Display a different recipe in each recipe view.

Procedure

To configure a simple recipe view, proceed as follows:

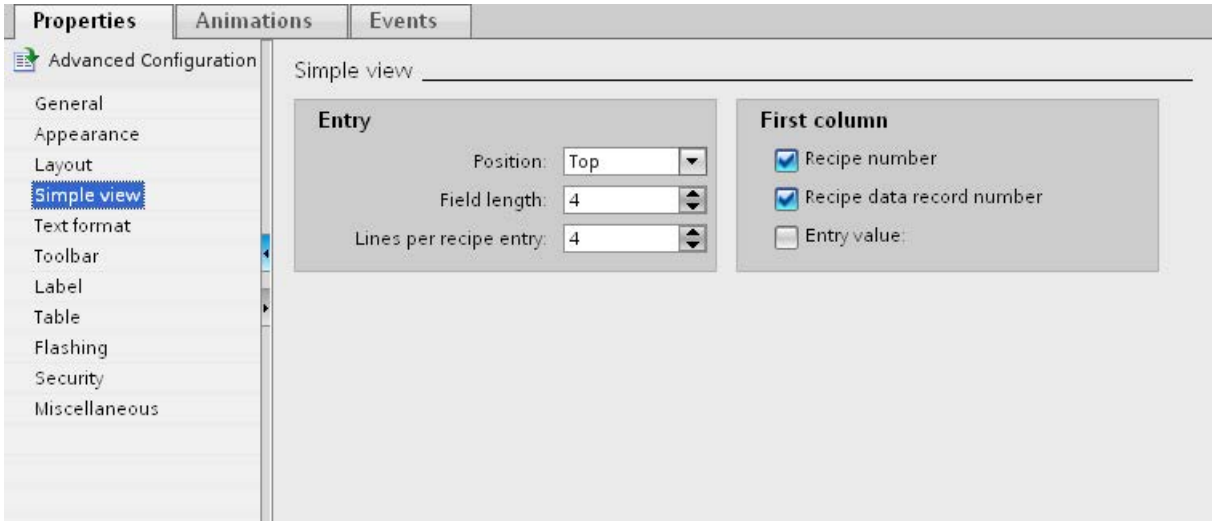
1. Paste the recipe view into the screen. You will find the recipe view under "Controls" in the "Tools" task card.
2. Only in devices which also support the extended recipe view: Activate "Simple view" under "Properties > Display > Mode".

3. If you want to display only the recipe data records of a specific recipe in the recipe view, select the specific recipe under "Properties > General > Recipe".

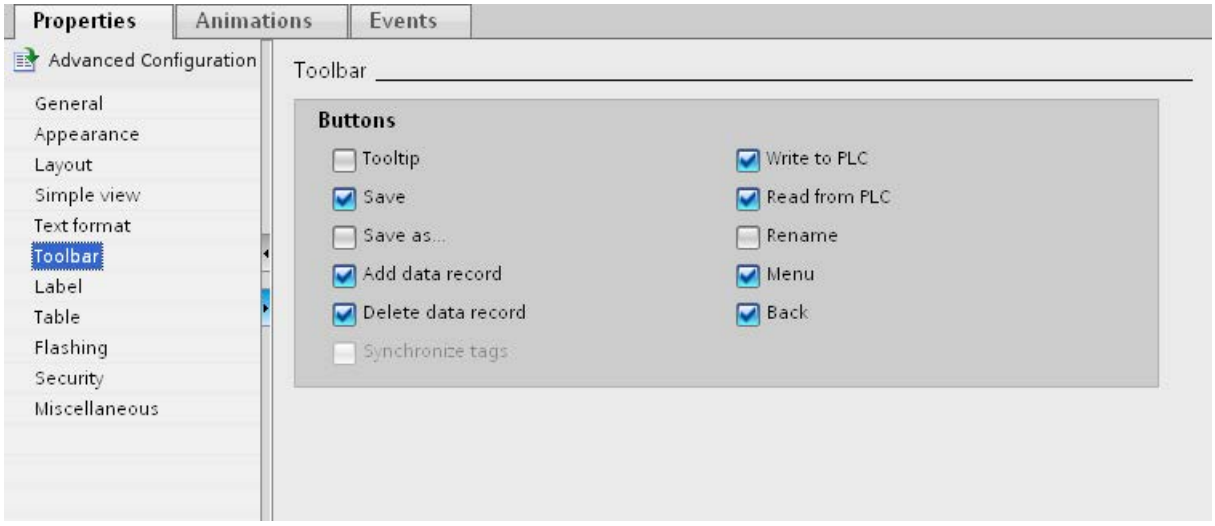


4. If you only want to display the recipe data in the recipe view, deactivate "Processing mode" in the "Recipe data record" area.
5. You can define additional options for the recipe view under "Properties > Appearance" and "Properties > Layout".

6. Select "Properties > Simple view" to select the position, the field length, and the number of lines required.
   Select "Position > Top" to display the recipe value in the first line of the recipe entry.
   Select "Position > Bottom" to display the recipe value in the last line of the recipe entry.



7. Under "Properties" > Toolbar" specify which menu commands are available in the recipe view in Runtime.



Result

The simple recipe view is configured. You can use the recipe view to display and edit recipe data during runtime.

Deactivation of the editing mode in "Properties > Properties > General" has no impact on the toolbar icons. The buttons you activated in "Properties > Toolbar" can still be used even if editing mode is disabled.

10.4.5 Using recipes in Runtime

10.4.5.1 Using the simple recipe view

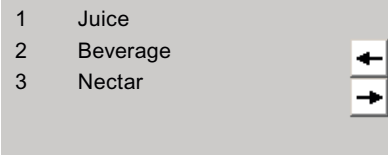
Description of the simple recipe view

Layout

The simple recipe view consists of the following display areas:

- Recipe list
- Data record list
- Element list

This application is illustrated below:



In the simple recipe view, each area is shown separately on the HMI device. You can use the shortcut menu to operate each of these display areas.

The simple recipe view always begins with the recipe list.

Operation

You have the following options for using the simple recipe view, according to the configuration:

- Create, change, copy or delete recipe data records
- Read recipe data records from the PLC or transfer to the PLC

Using the display area and shortcut menu

Toggle between the display areas and the shortcut menus to operate the simple recipe views.

The table below shows the operation of the display area.

Button	Key	Function
	<Enter>	The next lowest display area is opened, i.e. the data record list or the element list.
	<Esc>	The previous display area opens.

Button	Key	Function
	<INS>	Creates a new data record for the selected recipe if the list of recipes or recipe data records is displayed. Then changes to the list of recipe element. Requirement: "Properties >General > Processing mode" is activated. The button can be simulated with the "Key SimulateSystemKey" function even on devices without keys.
	<DEL>	Deletes the selected recipe data record in the list of recipe data records. Requirement: "Properties >General > Processing mode" is activated.
	<Up>/<Down>	Selects the previous/next entry.
	<Pg Up>/<Pg Down>	Moves the display up or down one page.
	<Home>/<End>	Selects the first/last entry. The first/last entry is selected.

The table below shows the operation of the shortcut menu:

Button	Key	Function
→	<Right>	The shortcut menu of the display area opens.
←	<Esc>	The menu is closed. The display area opens.
	Input of the number of the menu command	The menu command is executed.

Shortcut menus of the simple recipe view

You can click the → button in each display area to call up a selection of commands. The command selection lists those commands that are available in the current display area. A number is assigned to each command. The command is executed when you enter this number. Alternatively select the command and press the <Return> key.

Shortcut menus in the recipe list

Menu command	Function
New	A new recipe data record is created for the selected recipe. If a start value is configured, it is displayed in the input field.
Display tooltip	The tooltip configured for the recipe is displayed.
Open	The record list of the selected recipe opens.

Shortcut menus of the recipe data record list

Menu command	Function
New	Creates a new recipe data record. If a start value is configured, it is displayed in the input field.
Deleting	The displayed record is deleted.
Save as	The selected data record is saved under a different name. A dialog box opens where you can enter the name.
Rename	Renames the selected data record. A dialog box opens where you can enter the name.
Open	The element list of the selected data record opens.
Previous	The recipe list opens.

Shortcut menus of the recipe element list

Menu command	Function
Save	The selected data record with the recipe element is saved.
To PLC	The displayed values of the selected data record are transferred from the HMI device to the PLC.
From PLC	The recipe values from the PLC are displayed in the recipe view of the HMI device.
Save as	The data record is saved under a new name. A dialog box opens where you can enter the name.
Display tooltip	The tooltip configured for the recipe element is displayed.
Rename	The selected recipe element is renamed. A dialog box opens where you can enter the name.
Previous	The data record list opens.

Shortcut menus in the data record list

Note

HMI device dependency

The following menu commands are configured in Basic Panels and in OP 77A, TP 177A, TP 177A (Portrait) and TB 177B.

Menu command	Function
To PLC	The displayed values of the selected data record are transferred from the HMI device to the PLC.
From PLC	The recipe values from the PLC are displayed in the recipe view of the HMI device.

Managing recipe data records

Recipe data record administration

You have the following options for managing the simple recipe view, according to the configuration:

- Creating new recipe data records
- Copy recipe data records
- Edit recipe data records
- Delete recipe data records

Creating new recipe data records

To create a new recipe data record, proceed as follows:

1. Select the recipe on the HMI device in which you want to create a new recipe data record.
2. Select the "New" command from the shortcut menu for the recipe list.
   A new data record with the next available number will be created.
   The element list of the new recipe data record opens.
3. Enter values for the elements of the recipe data record.
   The configuration data may already contain default values for the recipe data record.
4. Select the "Save" command from the shortcut menu for the element list.
   The dialog "Save as" opens.
5. Enter the name and number of the recipe data record.
6. Click the "OK" button.

Result

The new recipe data records will be saved to the selected recipe. If the recipe data records already exists, a system event will be output to the screen.

Copying a recipe data record

To copy a recipe data record, proceed as follows:

1. Select the recipe on the HMI device in which you want to copy an existing recipe data record.
2. On the HMI device, select the recipe data record of which you want to save a copy.
3. Select the "Save As" command from the shortcut menu for the data record list.
   The dialog "Save as" opens. The recipe data record is automatically given the next free recipe data record number.
4. Under name, enter the name of the record.
5. Click the "OK" button.

Result

The recipe data record is stored under the new name.

Modify recipe data record

To change a recipe data record, proceed as follows:

1. Select the recipe on the HMI device in which you want to edit an existing recipe data record.
2. Select the recipe data record that you want to edit on the HMI device.
3. Select the recipe data record.
   The element list of the recipe data record is displayed.
4. Replace the old values with new ones.
5. Select the "Save" command from the shortcut menu for the element list.

Result

The modified values are applied to the recipe data record.

Deleting a recipe data record

To delete a recipe data record, proceed as follows:

1. Select the recipe on the HMI device from which you want to delete an existing recipe data record.
2. Select the recipe data record that you want to delete on the HMI device.
3. Select the "Delete" command from the shortcut menu for the data record list.
4. Confirm this security prompt to delete the data record.

Result

The recipe data record is deleted.

Read recipe data record from PLC

Introduction

In Runtime, you can change values directly in the plant that are also stored in recipes in the HMI device. This applies if a valve was opened further directly in the plant than was specified in the recipe. The values of the recipe data records saved in the HMI device possibly no longer match the values in the PLC.

You can read the values of the recipe tags from the PLC and write them to a recipe data record.

The read values are written to the recipe data record that is currently displayed on the HMI device.

Procedure

To read a recipe data record from the PLC, proceed as follows:

1. Open the recipe on the HMI device.
   The data record list opens.
2. Select the element list of the recipe data record to which you want to apply the values from the PLC.
3. Select the "From PLC" command from the shortcut menu for the element list.
   The values are read from the PLC and displayed in the current recipe data record.
4. If you want to save the values, select the "Save" or "Save As" command.

Result

The values are read from the PLC, visualized on the HMI device and saved to the recipe data record.

Transferring a recipe data record to the PLC

Introduction

For the values of a data record that was changed in the recipe view to take effect, you must transfer the values to the PLC.

The values displayed in the recipe view are always transferred to the PLC.

Procedure

To transfer a recipe data record to the PLC, proceed as follows:

1. Open the recipe you want to use.
   The data record list opens.
2. Select the element list of the recipe data record whose values you want to transfer to the PLC.
3. Select the "To PLC" command from the shortcut menu for the element list.

Result

The values of the recipe data record are transferred to the PLC.

10.4.6 Example

10.4.6.1 Example of creating a recipe

Task

In this example, you create three recipes for a fruit juice mixing machine. The fruit juice mixing machine produces drinks with "orange", "apple" and "tropical" flavors. You create a recipe for each flavor.

Each recipe contains a recipe data record for the following mixing ratios:

- Beverage
- Nectar
- Juice

Settings

The settings relate to an HMI device which is connected to a SIMATIC S7-300 or SIMATIC S7-400.

In this example, you will need the following tags, recipes, recipe entries and recipe data records:

Tags:

Name	PLC connection	Address	Type
Liter water	Yes	DB 120, DBW 0	Integer
Liter concentrate	Yes	DB 120, DBW 4	Integer
Kilo sugar	Yes	DB 120, DBW 8	Integer
Gram flavoring	Yes	DB 120, DBW 12	Integer

Recipes:

- Orange
- Apple
- Tropical

Recipe entries:

Recipe element	Associated tag
Liter water	Liter water
Liter concentrate	Liter concentrate
Kilo sugar	Kilo sugar
Gram flavoring	Gram flavoring

Recipe data records for drink, nectar and juice:

Data record name	Liter water	Liter concentrate	Kilo sugar	Gram flavoring
Beverage	30	70	45	600
Nectar	50	50	10	300
Juice	5	95	3	100

Procedure

To create a recipe, proceed as follows:

1. Create the following tags with the settings specified above: "LiterWater", "LiterConcentrate", "KiloSugar" and "GramFlavoring".
2. Create the "Orange", "Apple" and "Tropical" recipes with the settings indicated above. Create the recipe entries in each recipe.

Elements | Data records

Name	Display name	Tag	Default value	Decimal places	Infotext
Water	Water	LitreWater	0	0	
Concentrat	Concentrat	LitreConcentrat	0	0	
Sugar	Sugar	KiloSugar	0	0	
Aroma	Aroma	GramAroma	0	0	
<Add new>					

3. Not for Basic Panels: Configure each recipe so that you can synchronize the recipe data records between the recipe screen and recipe view. The values of the recipe tags should not be transferred automatically to the PLC.
   You will have to make the following settings in the Properties dialog for the recipe concerned: Under "Properties > Options":
   – Activate the "Synchronize recipe view and recipe tags" option.
   – Activate the "Manual transfer of individual modified values (teach-in mode)" option.
4. Create the data records indicated above in each recipe. Enter the values indicated above in each of the data records.

Elements | Data records

Name	Display name	Number	Water	Concentrat	Sugar	Aroma	Comment
Beverage	Beverage	1	30	70	45	600	
Nectar	Nectar	2	50	50	10	300	
Juice	Juice	3	5	95	3	100	
<Add new>							

Result

The "Orange", "Apple" and "Tropical" recipes have been created.

10.5 Configuring user administration

10.5.1 Field of application of the user administration

Principle

The access protection controls access to data and functions in Runtime. This feature protects your applications against unauthorized operation. Safety-related operations are already limited to specific user groups when a project is being created. To this purpose you set up users and user groups that you equip with characteristic access rights, so-called authorizations. You then configure the authorizations required for operation of safety-related objects. Operators only have access, for example, to specific operator controls. Commissioners, for example, have unlimited access in Runtime.

Definition

You administer users, user groups and authorizations centrally in the user administration of WinCC. You transfer users and user groups together with the project to the HMI device. The users and passwords are managed on the HMI device in the User view.

Application example

You configure the "Service" authorization so that only service technicians have access to the configuration parameters. You assign the authorization to the "Service technician" user group. This allows all members of this group to set the protected configuration parameters.

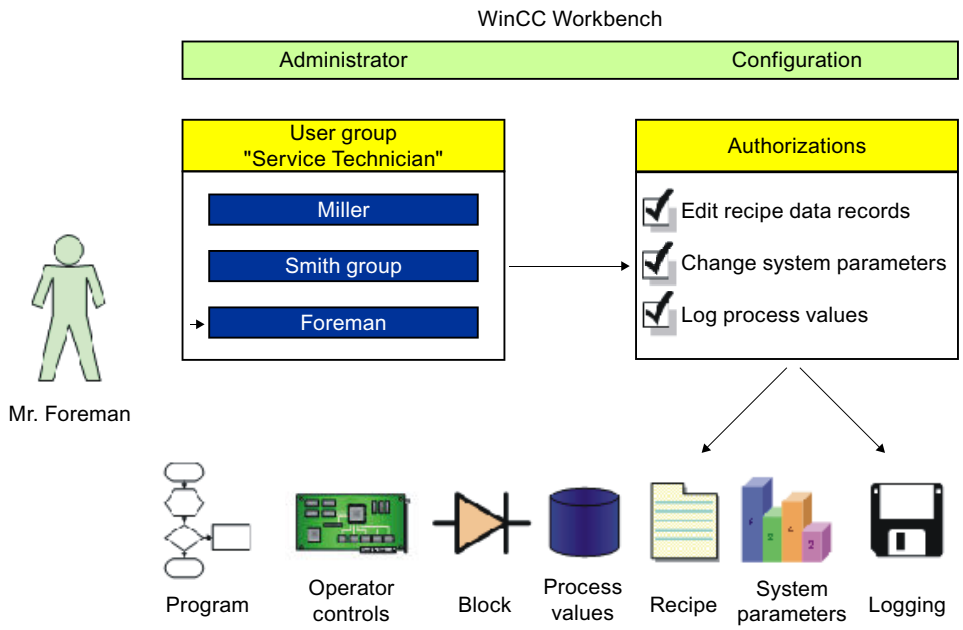
NOTICE
Access protection does not protect against incorrect operations. It is your job to ensure that only authorized personnel with appropriate training will design, commission, operate and maintain plants and machines. Access protection is not suitable for defining work routines and monitoring their observance.

10.5.2 Form of the user administration

Introduction

In case of a project in manufacturing engineering, the environment at the equipment manufacturer has to be differentiated from the environment at the end customer as plant operator.

The equipment manufacturer allows users, for example Mr. Foreman, a specific access within the application or HMI device. However, a user Foreman does not exist at the end customer. The machine manufacturer cannot know the end users and the tasks they have to perform for configuration. The final users are usually set after commissioning at the end customer.



Principle

To minimize the work required for management, authorizations are assigned via user groups and not directly to individual users.

A user group assembles configured authorizations according to common jobs. For example, all permissions required for a service job are collected in a "Service technician" group. When you create a user who should be responsible for servicing, you simply assign him to the "Service technician" group.

The user view enables user administration in Runtime. Use user view to create, delete and assign an authorization to users in Runtime.

The user administration separates the administration of the users from the configuration of the authorizations. This ensures flexibility at the access protection.

Defaults can be set for the user administration during the configuration phase in the Engineering System.

10.5.3 Basics

10.5.3.1 Users

Introduction

You can create users in the "Users" tab of the "User administration" editor and assign them to user groups. The "Users" tab is part of the user administration in WinCC.

Open

To open the "Users" tab, double-click "User administration" in the project window.

Work area

The users are managed in the work area:

- You create or delete users.
- You assign users to user groups.

Note

You can assign a user to exactly one user group.

Inspector window

When you select a user, you can change the password in the "General" group. Under "Automatic logoff" you can specify if the user is to be automatically logged off by the HMI device when there is no operator activity after the specified time.

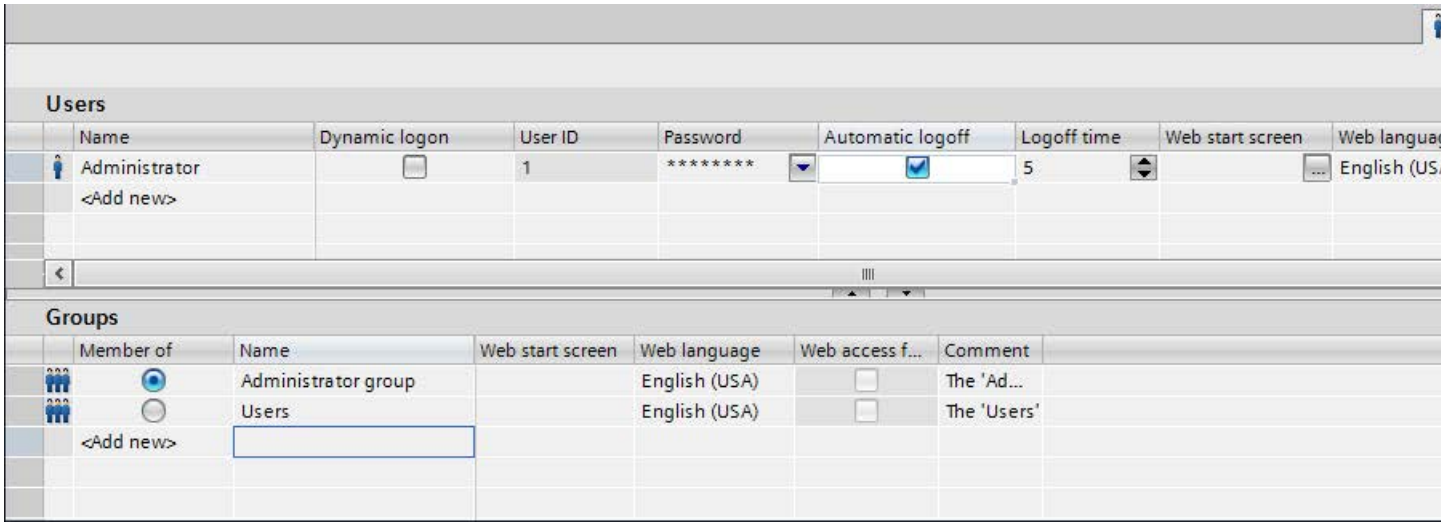
10.5.3.2 Users work area

Introduction

The "Users" work area lists the users and user groups in table form. You administrate the users and assign them to a user group.

Principle

The work area consists of the "Users" and "Groups" tables.



The "Users" table shows the existing users. When you select a user in this table, the "Groups" table shows the user group to which the user belongs.

Note

For the user "Administrator", the default password is "administrator". For security reasons, you should change the password of this user.

10.5.3.3 User groups

Introduction

You can create users and authorizations in the "User groups" tab of the "User Administration" editor. The "User groups" tab is part of the user administration in WinCC.

Open

Double-click the "User administration" in the project window. Open the "User groups" tab.

Work area

The user groups and authorizations are managed in the work area:

- You create new user groups and authorizations or delete them.
- You assign the authorizations to the user groups.

Inspector window

When a user group or an authorization is selected, you can edit the name in the "General" group. You can also enter a brief description in the "Comment" group.

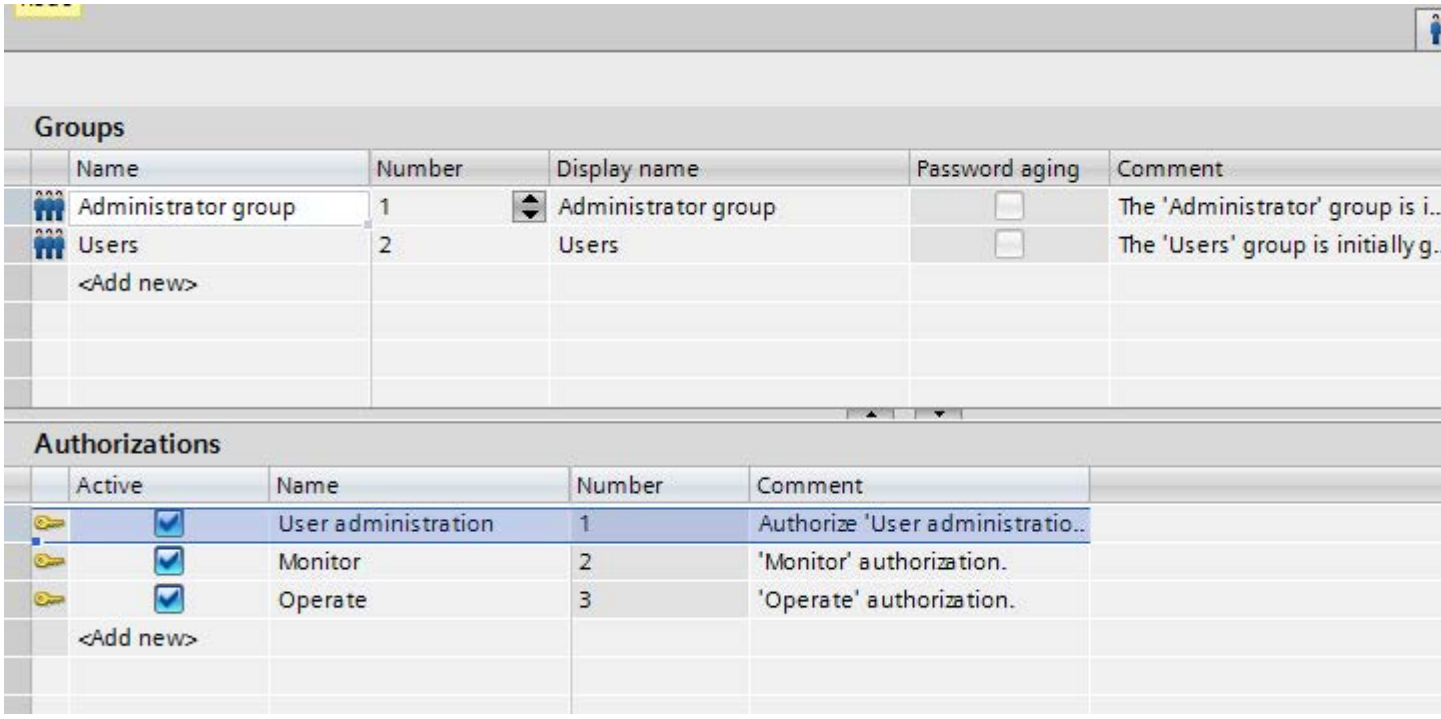
10.5.3.4 User groups work area

Introduction

The "User groups" work area shows a table of the groups and their authorizations. You administer the user groups and assign authorizations to them.

Principle

The work area consists of the "Groups" and "Authorizations" tables.



The "Groups" table shows the existing user groups. When you select a user group in this table, the "Active" column of the "Authorizations" table shows the authorizations which were assigned to the user group.

The number of the user group and of the authorization is assigned by the user administration. The designations and descriptions are assigned by you.

The number of predefined authorizations are fixed. Authorizations that you create can be freely edited. Ensure that the assigned numbers are unique.

10.5.3.5 Settings for the user administration

Introduction

In the "Runtime settings > User administration" editor, configure the security settings for users and their passwords in runtime.

Open

Double-click the "Runtime settings" editor in the project window. Click "User administration".

Work area

You carry out the settings for the validity of passwords in runtime in the work area. You determine the complexity of the password, for example.

Effects in Runtime

The security settings have the following effects in runtime, depending on the configuration.

- "General" group
  - "Enable limit for logon attempts" check box selected
    The number entered in the "Number of incorrect logon attempts" box defines the number of logon attempts a user is allowed before being assigned to the "Unauthorized" group.
    "Enable limit for logon attempts" check box not selected
    The user has an unlimited number of logon attempts in runtime.
  - "Number of incorrect logon attempts" field
    If you enter "4" in the field, for example, and the fourth logon attempt fails, the user is automatically assigned to the "Unauthorized" group.
    You can specify 1 to 9 attempts.
  - "Logon only with password" check box
    When the check box is selected, the user will be authenticated by the password. The user name is not required.
    To match users to passwords, you cannot configure passwords more than once.
- "Hierarchy level" group
  - "Group-specific rights for user administration" check box
    When this check box is selected, administrators only manage users whose group number is less than or equal to their own.
    For example, an administrator whose group number is 5 can only manage users whose group number is less than or equal to 5. This means that the administrator can assign users only to groups with a group number less than or equal to 5.

- "Password" group
  - "Enable password aging" checkbox selected
    The password expires after the number of days set in the "Validity of the password (days)" field.
    The "Password aging" column is selected in the "User groups" editor. This enables you to specify group-by-group, if the passwords should expire or if the password generations should be saved. Passwords never expire for groups for which password aging is not enabled.
  - "Prewarning time (days)" field
    The user is informed that the password will expire the specified number of days before the password expires.
  - "Password generations" field
    If the user changes the password, the new password must be different from the specified number of previous passwords. The number of password generations ranges from 1 to 5.
- "Password complexity" group
  - "Must include special characters" check box selected
    The user must enter a password containing at least one special character at any position.
  - "Must include number" check box selected
    The user must enter a password containing at least one number at any position.
  - "Minimum password length" field
    The user must enter a password with a minimum length, as specified in the "Minimum password length" field.
    The minimum length of the password is 3 characters.

10.5.4 Setting up the user administration

10.5.4.1 Basics on user administration

Principle

This section addresses four target groups. The topics are organized correspondingly. The target groups serve as examples for different groups of persons who use the user administration.

1. Administrator OEM
2. Administrator RT
3. Planners
4. Operator

As Administrator OEM you create the user groups, users and authorizations for Runtime in the Engineering System of, for example, an equipment manufacturer.

As Administrator RT you administer users in Runtime by means of the "User view."

As the project engineer you assign the authorizations to the user groups in the Engineering System. In addition, you configure the authorizations for objects.

As Operator you log on in runtime. You can only access a protected object if you have the required authorization.

Note

The Administrator RT target group already exists in the Runtime user administration as the predefined user group "Administrator group." For a better understanding the predefined user groups and authorizations are not used below.

10.5.4.2 Administering users for Runtime

Creating an authorization

Introduction

You create an authorization and assign it to one or more user groups.

Requirements

The "User groups" work area is open.

Procedure

1. Double-click "Add" in the "Authorizations" table.
2. Enter "Stop runtime" as the name of the authorization.
3. Enter a brief description as the "Comment".

Creating a user group

Introduction

User groups are created so that you do not have to assign an authorization to every single user. You create a user group, assign authorizations and then assign users to it.

Note

The name of the user group has to be unique within the project. Otherwise the input is not accepted.

Note
Using SIMATIC Logon

Ensure that the names of the user groups in Windows and WinCC are identical.

Requirements

The "User groups" work area is open.

Procedure

1. Double-click "Add" in the "Groups" table.
2. Enter "Operators" as the "Name" of the user group.
3. Change the "Number" of the user group as required.
4. Enter "Display name" of the "Operators" user group.
5. Enter a brief description as the "Comment".

In runtime, the user view shows the display name of the user group. The display name of the user group depends on the language. You can specify the name in several languages and switch between languages in runtime.

See also

Assigning an authorization (Page 8249)
Creating users (Page 8250)
Assigning a user to a user group (Page 8251)
Managing user groups (Page 8253)
User administration (Page 8252)

Assigning an authorization

Introduction

When you allocate an authorization to a user group, all assigned users have this authorization.

Requirements

- A "Stop runtime" authorization has been created.
- An "Operators" user group has been created.
- The "User groups" work area is open.

Procedure

1. Click on the "Operators" user group in the "Groups" table. The "Authorizations" table shows all authorizations.
2. Enable the "Stop runtime" authorization in the "Authorizations" table.

Note

The "Stop runtime" authorization is only a designation and does not have any relation to the function "StopRuntime". You have to establish this relation yourself. To do so, configure the "StopRuntime" system functions at a button and select "Stop runtime" as the "Authorization".

See also

Creating a user group (Page 8248)

Creating users

Introduction

You create a user so that users can log on with their user names in runtime after loading to the device.

As an alternative, you can create and change users by means of the "User view" in Runtime.

In order for a created user to have authorizations you have to assign him to a user group and allocate authorizations to the user group.

The logon is successful when the user name entered during the logon matches a user in Runtime. In addition, the entered password must agree with the stored password of the user.

Note

Note that the entry is case-sensitive.

Requirements

The "Users" work area is open.

Procedure

1. Double-click "Add" in the "Users" table.
2. Enter "Foreman" as the user name.

   Note

   The user name must be unique within the project. Otherwise the input is not accepted.

   Note

   Do not use special characters such as / " § $ % ? when you enter a user name and a password. ' &.

3. Click the button in the "Password" column. A dialog box for entering the password is displayed.
4. Enter the password of the user.
5. To confirm the password enter it a second time in the lower field.
6. Close the dialog box using the icon.
7. If a user is to be logged off after a specific time period, activate "Automatic logoff".
8. Click in the "Logoff time" column. The preset value for "Logoff time" is 5 minutes.
9. Enter a brief description as the "Comment".

See also

Creating a user group (Page 8248)

Assigning a user to a user group

Introduction

When you assign a user to a user group, the user has the authorizations of the user group.

Note

You have to assign a user to exactly one user group. The assignment is checked during the consistency check and generation of the project.

Requirements

- The user "Foreman" has been created.
- An "Operators" user group has been created.
- The "Users" work area is open.

Procedure

1. Click on the "Foreman" user in the "Users" table. The "Groups" table shows all user groups.
2. Activate the "Operators" user group in the "Groups" table.

See also

Creating a user group (Page 8248)

User administration

Introduction

In the work area, you can administer users and assign them to user groups.

Requirements

The "Users" work area is open.

Changing the user name

1. In the "Users" table, double-click the field in the "Name" column to change the user name.
2. Change the user name.
3. Confirm your entry with <Return>.

Alternatively, select the user in the work area. Change the user name under "Properties > Properties > General" in the Inspector window.

Changing the password of the user

1. Click the button in the "Password" column of the "Users" table. A dialog for entering the password opens.
2. In the "Enter password" field, enter the new password.
3. Reenter the new password in the "Confirm password" field.
4. Confirm your entry with <Return>.

Alternatively, select the user in the work area. Change the password under "Properties > Properties > General" in the Inspector window.

Displaying invisible columns

1. Position the mouse cursor on the title of the "Users" table.
2. Right-click to open the shortcut menu and enable the display of the "Logoff time" column, for example.

Changing the logoff time of the user

1. In the "Users" area, double-click on the field in the "Logoff time" column to change the logoff time.
2. Change the logoff time.
3. Confirm your entry with <Return>.

Alternatively, select the user in the work area. Change the logoff time under "Properties > Properties > Automatic logoff" in the Inspector window.

Deleting a user

1. Select the line of the user to be deleted.
2. Open the shortcut menu with the right mouse button and select the "Delete" command.

Note

Predefined users cannot be deleted.

See also

Creating a user group (Page 8248)

Managing user groups

Introduction

In the workplace you administer user groups and assign authorizations for use in runtime.

Requirements

The "User groups" work area is open.

Changing the name of the user group

1. In the "Groups" table, double-click the field in the "Name" column to change the name of the user group.
2. Change the name of the user group.
3. Confirm your entry with <Return>.

Alternatively, select the user group in the work area. Change the name under "Properties > Properties > General" in the Inspector window.

Note

Predefined user groups cannot be deleted.

Displaying invisible columns

1. Position the mouse cursor on the title of the "Users" table.
2. Right-click to open the shortcut menu and enable the display of the "Display name" column, for example.

Changing the displayed name of the user group

1. In the "Groups" table, double-click the field in the "Display name" column to change the display name of the user group.
2. Change the displayed name of the user group.
3. Confirm your entry with <Return>.

Alternatively, select the user group in the work area. Change the display name under "Properties > Properties > General" in the Inspector window.

Deleting a user group

1. Mark the line of the user group to be deleted.
2. Open the shortcut menu with the right mouse button and select the "Delete" command.

Note

Predefined user groups cannot be deleted.

Changing the name of the authorization

1. In the "Authorizations" table, double-click the field in the "Name" column to change the name of the authorization.
2. Change the name of the authorization.
3. Confirm your entry with <Return>.

Alternatively, select the authorization in the work area. Change the name under "Properties > Properties > General" in the Inspector window.

Deleting authorizations

1. Mark the line of the authorization to be deleted.
2. Open the shortcut menu with the right mouse button and select the "Delete" command.

Note

Predefined authorizations cannot be deleted.

See also

Creating a user group (Page 8248)

10.5.4.3 Managing users in Runtime

Users in Runtime

Principle

You create users and user groups and assign authorizations to them. You configure objects with authorizations. After download to the HMI device, all objects which were configured with an authorization are protected against unauthorized access in Runtime.

User view

When you configure a user view in the Engineering System, you administer users in this user view following download to the HMI device.

NOTICE
Changes in the user view are effective immediately in Runtime. Changes in runtime are not updated in the engineering system. When downloading the user administration to the HMI device, all changes in the user view are overwritten after a security prompt and based on the settings.

Users who have a "User administration" authorization have unlimited access to the user view. This allows them to administer all users. Any other user has only limited access to the user view for self administration.

User view

Purpose

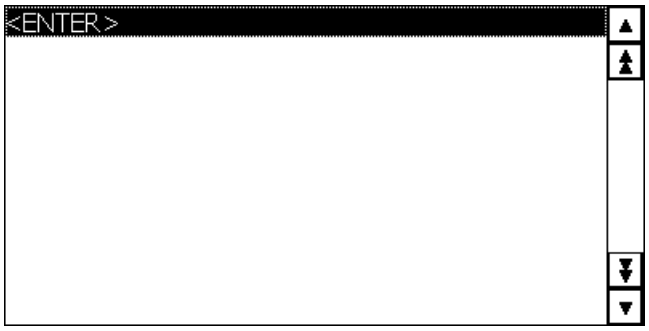
You configure a user view in the engineering system to also administer the users in Runtime.

Structure

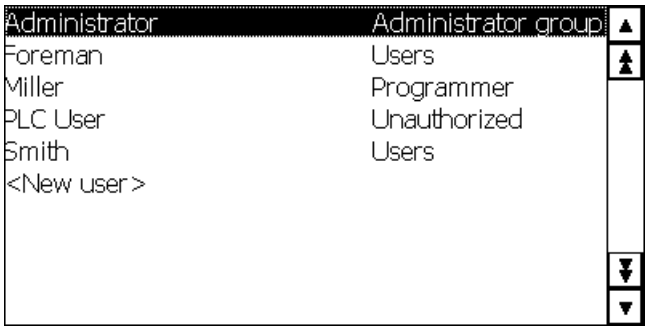
The user view shows the following in each line:

- The user
- The corresponding user group.

If no user is logged on, the user view is empty. The content of the individual fields is displayed after logon.

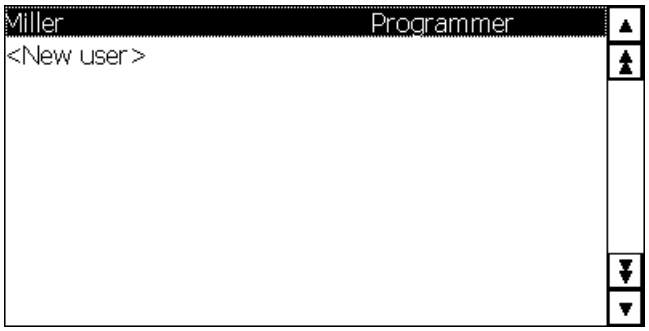


User view of an administrator



When an administrator is logged on, the user view shows all the users. The administrator changes the user name and the password. The administrator creates new users and assigns them to an existing user group.

User view of a user



When no administrator is logged on, the user view shows only the logged-on user. Users can change their own passwords.

User view

Purpose

You configure a user view in the engineering system to also administer the users in Runtime.

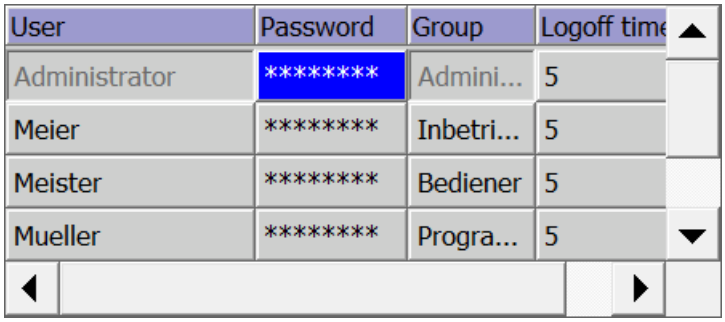
Structure

The user view shows the following in each line:

- The user
- The encrypted password
- The corresponding user group
- The logoff time

If no user is logged on, the user view is empty. The content of the individual fields is displayed after logon.

User view of an administrator



User	Password	Group	Logoff time
Administrator	*********	Admini...	5
Meier	********	Inbetri...	5
Meister	********	Bediener	5
Mueller	********	Progra...	5

When an administrator is logged on, the user view shows all the users. The administrator changes the user name and the password. The administrator creates new users and assigns them to an existing user group.

User view of a user

Benutzer	Kennwort	Gruppe	Abmeldezeit
Mueller	*********	Progra...	5

When no administrator is logged on, the user view shows only the logged-on user. Users can change their own passwords and logoff time.

Configuring a user view

Introduction

You configure a user view in the Engineering System to also administer users in Runtime.

Requirements

A screen has been created.

Procedure

1. Select the "User view" object from the "Controls" category in the toolbox.
2. Drag-and-drop the "User view" object into the screen.
3. Click on "Properties > Properties" in the Inspector window.
4. Specify the appearance of the "User view".
5. You can, for example, select " "Display mode > Fit to size > Fit object to contents".

Result

You have created a user view in the screen.

Creating users

Introduction

You create a user so that users can log on under their user name in runtime.

As an alternative, you can create users in the engineering system and download them to the HMI device.

The logon is successful only when the user name entered during the logon matches a user in runtime. In addition, the password entered at logon has to match that of the user.

Note

Note that the entry is case-sensitive.

Note

Do not use special characters such as / " § $ % ? when you enter a user name and a password. ' &.

You assign the user to a user group. The user then has the authorizations of the user group.

Note

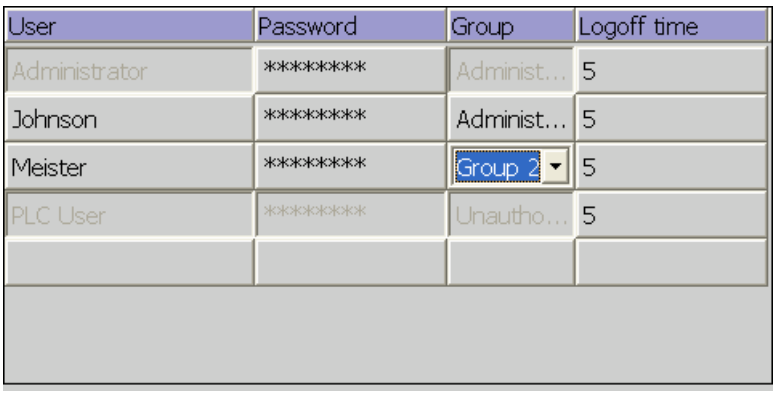
Runtime users must be assigned to a user group. The user group is created in the engineering system. The designation of the user group is language-dependent.

Requirements

- The user view is open.
- A "Group 2" user group has been created.

Procedure

1. Click "<New User>" in the user view. A dialog opens.
2. Enter "Foreman" as the user name.
3. Press the <Return> button.
4. Click "Password."
5. Enter the password of the user.
6. Press the <Return> button. The password is hidden.
7. Click in the "Group" column.
8. Select "Group 2" as the "Group".



9. Press the <Return> button.
10. Click in the "Logoff time" column.
11. Enter the time after which the user is logged off automatically.

Managing users in the user view

Introduction

If you have configured a user view in the engineering system, the users and user groups can be managed in runtime.

NOTICE
Changes in the user view are effective immediately in runtime. Changes in runtime are not updated in the engineering system. When downloading the user administration to the HMI device, all changes in the user view are overwritten after a security prompt and based on the settings.

Requirements

- Runtime is enabled.
- A complex user view has been created.
- The screen with the user view is open.
- You have the default "User administration" authorization.

Note

If you do not have a "User administration" authorization, you can only change your own password and your logoff time.

Changing the user name

1. Enter a new user name in the "Users" column in the user display.
2. Confirm your entry with <Return>.

Note

The user can then no longer log on with his previous password in runtime. If you delete the name and press <Return>, the user is deleted.

Changing the password of the user

Availability of complex user display in selected devices only.

1. Enter a new password in the "Password" column in the user display.
2. Confirm your entry with <Return>.

Note

The user can then no longer log on with his previous password in runtime.

If you delete the password in the complex user view and press <Return>, the user will be deleted.

Changing the logoff time of the user

1. Enter a new logoff time in the "Logoff time" column in the user display.
2. Confirm your entry with <Return>.

Deleting a user

1. Click the name of the user to be deleted.
2. Delete the name.
3. Press the <Return> button.

Note

The user can no longer log on in runtime.

Assigning a user to a different user group

1. Activate the user group field for the corresponding user.
2. Select a user group.
3. Confirm your selection with <Return>.

Unlocking users

Unlock locked out users

The check box "Activate limit for login attempts" is activated in the "Runtime settings > User administration".
The number 3 is entered in the field "Number of invalid login attempts".

If users have three failed attempts at login, e.g. by entering an incorrect password, they are assigned to the "Unauthorized" group. The user loses all authorizations. The user can still log on, but no longer has any authorizations. Only a user with administrator rights re-assigns the unauthorized user to a user group.

Logging on as a user

Introduction

As a rule you log-on as a user by means of a special button. The logon dialog box is displayed to this purpose.

The logon dialog box is displayed by default during access to a protected object if

- No user is logged on in runtime.
- The logged-on user does not have the required authorization.

Note

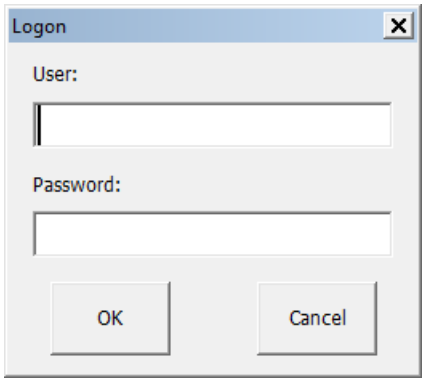
The system always opens the logon dialog on the OP 73, OP 77A, TP 177A and Basic Panels HMI devices when you press an access-protected button:

Requirements

- Under "Runtime settings > User administration" the
  - "Enable limit for logon attempts" check box has been selected.
  - The number 3 is entered in the field "Number of incorrect login attempts".
- The "ShowLogonDialog" system function is configured on a button called "Logon".

Procedure

1. Click the "Logon" button. The logon dialog box is displayed.



2. Enter your user name as it was specified in the user administration, for example "Foreman". If someone has logged on before you, the name of the user will be displayed.
3. Enter the corresponding password. The input is concealed.
4. Click "OK" to close the dialog box.

Logon was successful

If you have entered the user name "Foreman" and the entered password corresponds with the stored password, you are logged on as the user "Foreman" in runtime. You have the authorizations of the user "Foreman".

When the user "Foreman" accesses a protected object such as the "Logging" button, access to this protected object will only be authorized if the user "Foreman" has the required authorization. The programmed function is executed immediately.

If you do not have the required authorization after the successful log-on, a corresponding error message is displayed. However, you remain logged on in runtime.

Logon was unsuccessful

An error message is displayed.

In order to maintain security, you or the user logged-on before you no longer has any authorizations. However, access to unprotected objects remains possible. The user view does not, however, show any entries. The user view and the authorizations change after the next successful log-on.

If the third login attempt has failed, the user will be assigned to the "Unauthorized" group. For this reason, do not configure a user group with this display name.

If the "Log off" function is called up or the logoff time of the user has expired, the user is logged off.

10.5.4.4 Configuring access protection

Access protection

Introduction

You configure an authorization at an object in order to protect it against access. All logged-on users who have this authorization can then access the object. If a user does not have authorization to operate an object, the logon dialog is displayed automatically.

Note

Several system functions are available under "User administration" so that user, password, and user group can be edited, for example, in the PLC.

Configuring authorization (Basic,Advanced; Professional)

Introduction

You configure the "Stop runtime" authorization on a button. Then only those users who have the appropriate authorization have access to this button, for example all the users of the "Operators" user group.

This ensures that access to the command button is protected. If a logged-on user who belongs to the "Operators" user group and has the required authorizations clicks the button, runtime is stopped.

An example describes in detail how to configure a command button with access protection.

Requirements

- The "Operators" user group has been created.
- The "Stop runtime" authorization has been created.
- A screen has been created and opened.
- The screen contains a button.

Procedure

1. Click the button in the screen.
2. Click "Properties > Properties > Security" in the Inspector window.
3. Select "Stop runtime" as the "Authorization".
4. In the Inspector window, select "Properties > Events > Click".
5. Select a system function from the function list, for example, "StopRuntime".

Note

The "Enable" and "Disable" events are only used to detect whether an object was selected or deselected. The events do not, however, trigger a password prompt.

Do not use the "Enable"or "Disable" event if you want to configure access protection at the function call of the object.

10.5.5 Reference

10.5.5.1 Objects with access protection

Introduction

The following objects can be configured with an authorization:

- Date/time field
- I/O field
- Graphic I/O field
- Recipe view
- Switch

- Button
- Symbolic I/O field
- System diagnostic view

10.5.5.2 Default user groups and authorizations

Principle

The predefined user groups and authorizations have the following numbers:

User group	Number
"Administrator group"	1
"Users"	2

Authorization	Number
"User administration"	1
"Monitor"	2
"Operate"	3

10.5.6 Examples

10.5.6.1 Example: Configuring a button with logon dialog box

Task

In the following example, you configure the function "ShowLogonDialog" for a button. A different user can then log on in runtime when the shift changes, for example. In the process the user previously logged on is logged off.

Note

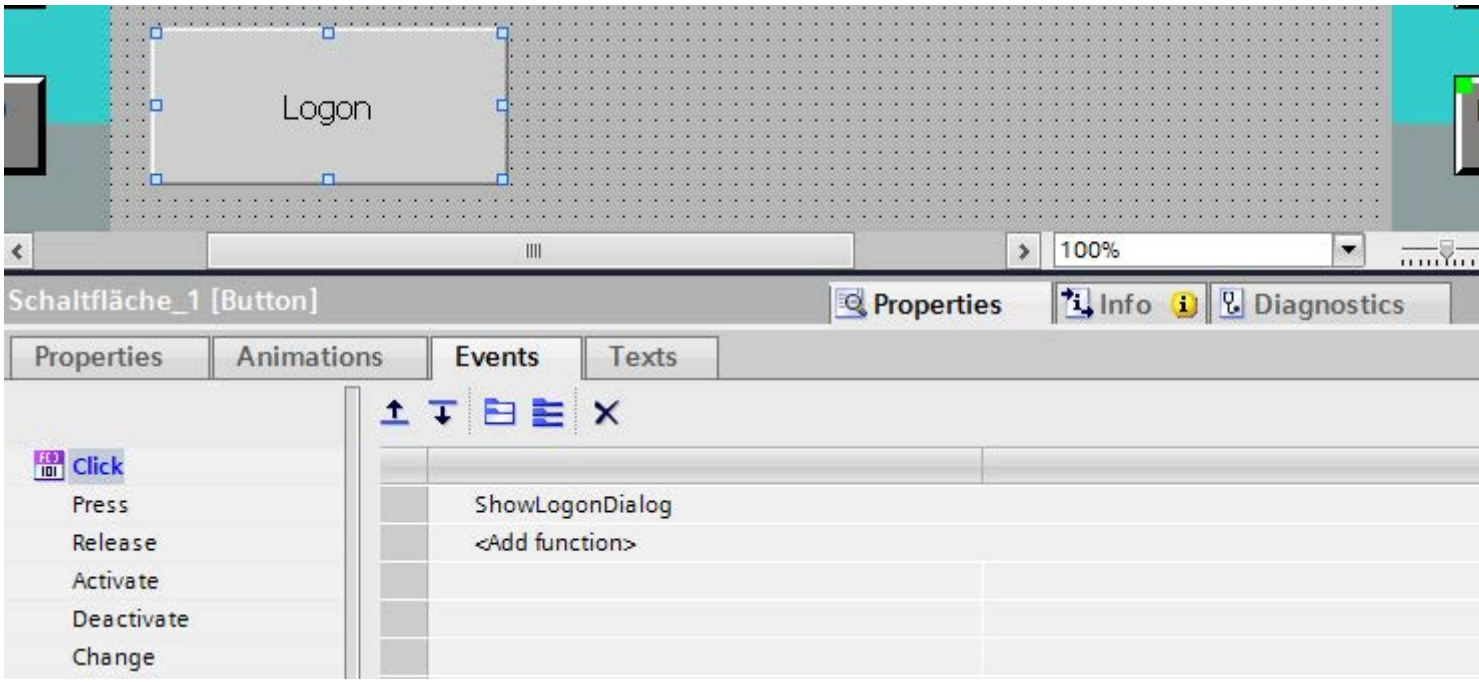
In runtime the logon dialog box is not displayed by default until you access a protected object. Either no user is logged on or the logged-on user does not have the required authorization.

Requirements

- A screen has been created.
- A button has been created in the screen.

Procedure

1. Click the button in the screen.
2. Click "Properties > Events > Release" in the Inspector window.
3. Click the entry "Add function" in the "Function list" table.
4. Select the system function "ShowLogonDialog" from the "User administration" group.



Result

If the user clicks on the button in runtime, the function "ShowLogonDialog" is called up. When the function "ShowLogonDialog" is called up, the logon dialog box is displayed. The user logs on with his user name and password.

10.5.6.2 Example: Logging the logon and logoff events

Task

In the following example, you configure the function "TraceUserChange" to the event "User change".

Principle

The "TraceUserChange" function is called when a user logs on or off. When a function is called up, a system message with information about the corresponding user is output.

This system message can be archived. When archiving, the system message is provided with a date stamp and time stamp. This ensures that you can track which user was logged on at the HMI device at which time and for how long.

Requirements

- You have created an HMI device with Runtime Advanced.
- The Inspector window is open.

Procedure

1. Double-click the "Scheduler" in the Project view.
2. Double-click "Add" in the table of the tasks.
3. Enter "Logon-Protocol" as the "Name".
4. Select "User change" as the "Trigger".
5. Open "Properties > Events" in the Inspector window.
6. Click the entry "Add function" in the "Function list" table.
7. Select the "TraceUserChange" system function.

Result

A system message is output when a user logs on or logs off.

10.5.6.3 Example of user management

Example: Structure of user management

Task

In the following example you set up a user administration for different users and user groups. The example orientates itself to a typical requirement profile from manufacturing engineering.

Principle

Completely different groups of persons are involved in a plant or project. Each group of persons protects its respective data and functions against access by others. For this purpose, users are created and assigned to a user group.

You can reproduce different views through user groups.

Example:

- Organizational view: Commissioners, Operators, Shift I, Shift II
- Technological view: Axis control, Tool changers, Plant North, Plant South

The following example orientates itself to the organizational view.

Every user group has characteristic requirements regarding access protection: A user group has operating authorizations for specific application cases. A programmer changes, for example, recipe data records.

In the example the users Miller, Group Smith and Foreman are created and assigned to different user groups.

Ms. Miller works as a programmer in the engineering system. The Group Smith are commissioners. Mr. Foreman is an operator.

Requirements

- A new project has been created.
- The "User administration" editor is open.

Procedures overview

Working with user administration has the following procedure in the example:

1. Creating authorizations The planner specifies which authorizations are required for access protection.
2. Configuring authorizations: The planner specifies which objects may be operated and which functions may be executed.
3. Creating user groups and allocating authorizations: The administrator creates the user groups together with the planner. The planner uses the authorizations to specify who may operate objects and change parameters.
4. Creating users and assigning them to a user group: The administrator administers the users.

Result

The aim is the following structure of the user administration of users, user groups and authorizations:

Users			User groups	Authorizations			
Miller	Smith	Foreman	Roles	Changing recipe records	Changing system parameters	Changing process parameters	Managing
			Administrator group				x
X			Programmer	X			
	X		Commissioning engineers	X	X	X	
		X	Operators	x			

The user "Foreman" who belongs to the "Operators" user group has access to the configured "To Recipe view" button.

Note

Alternatively, you can create several users as operators with different operating authorizations, for example, Operator Level 1, Operator Level 2.

Example: Creating and configuring authorizations

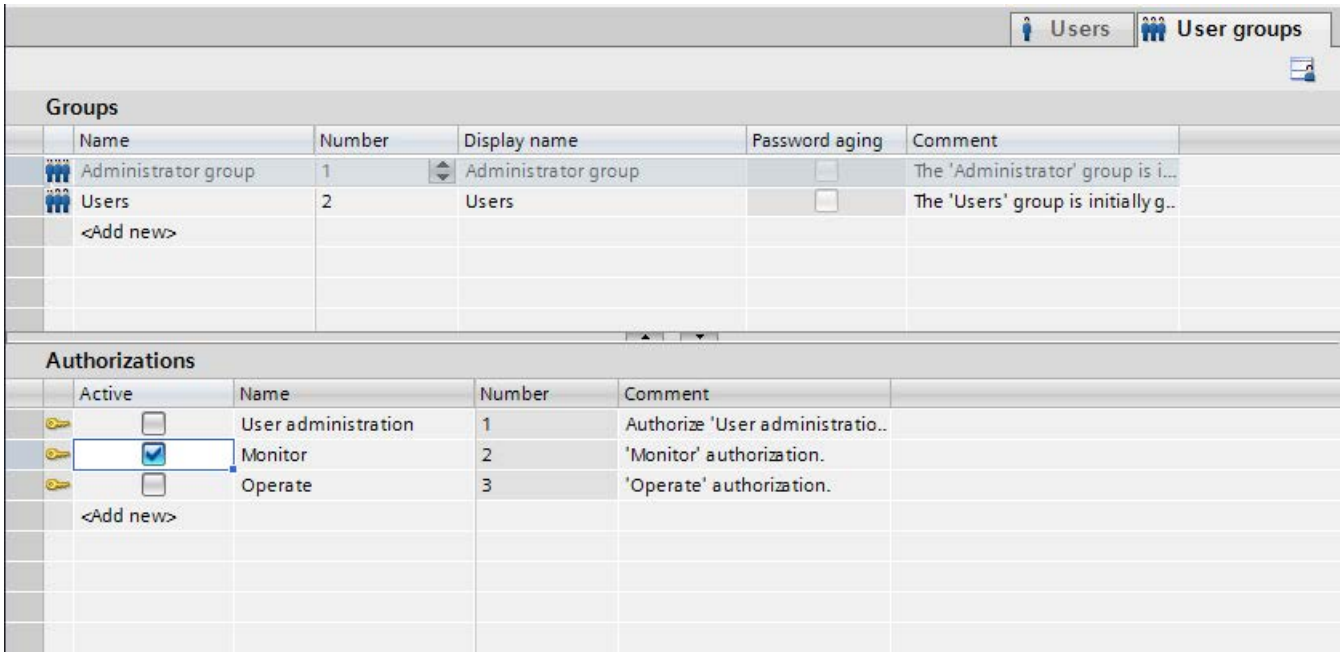
Task

The following example shows you how to create the authorizations

Procedure

1. Open the "User groups" work area.
2. Double-click "Add" in the "Authorizations" table.
3. Enter "Change recipe data records" as the "Name" of the authorization.
4. Repeat steps 2 and 3 to create additional authorizations: "Change system parameters", "Change process parameters".

Result



Example: Configuring a button with access protection

Task

In the following example you use a system function to create a button for a screen change. You protect the "To Recipe view" button against unauthorized operation. To do so, you configure the "Change recipe data records" authorization at the "To Recipe view" button.

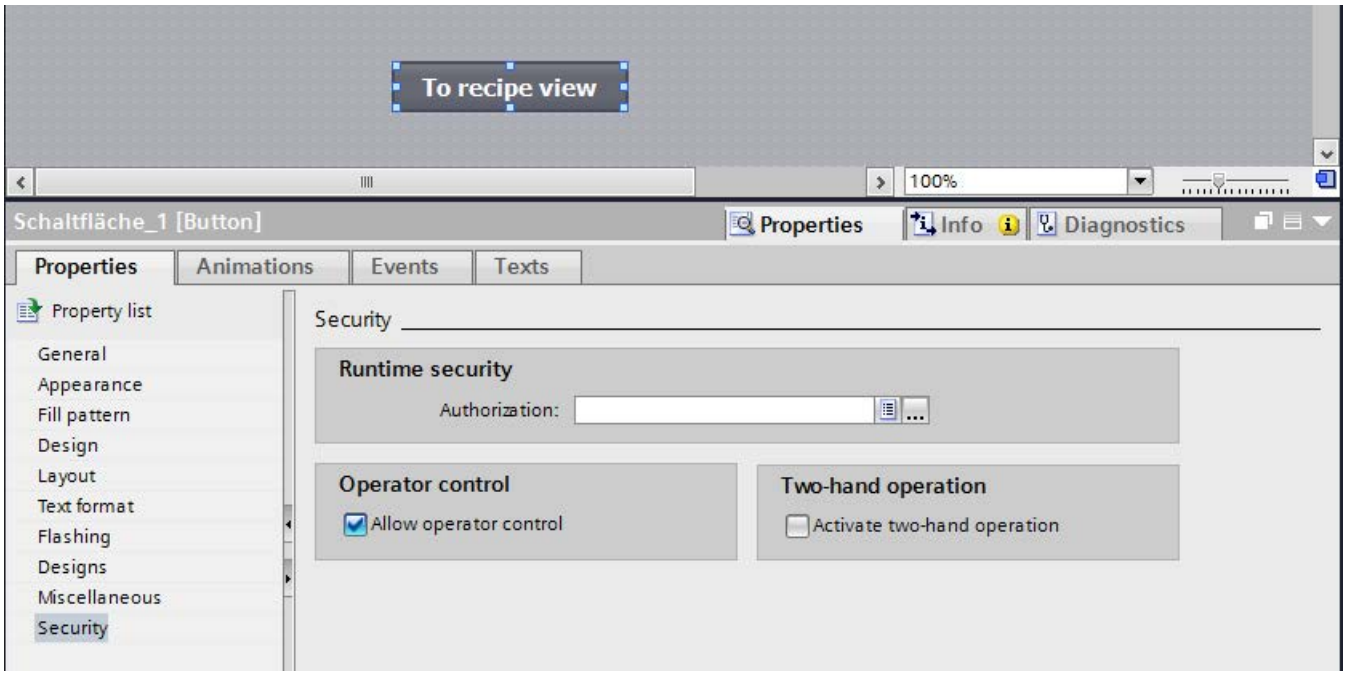
Requirements

- A "Change recipe data records" authorization has been created.
- A "Recipes" screen has been created.
- A "Start" screen has been created and opened.
- A button has been created and marked in the "Start" screen.

Procedure

1. Click "Properties > Properties > General" in the Inspector window.
2. Enter "To Recipe view" as the text.
3. Click "Properties > Events > Click" in the Inspector window.
4. Click the "Add function" entry in the first line of the "Function list" table.
5. Select the "ActivateScreen" system function in the "Screens" group.
6. Click the ... button in the "Screen name" field. A dialog box for selecting the screen opens.
7. Select the "Recipes" screen and use the ✓ button to close the dialog box.
8. Click "Properties > Properties > Security" in the Inspector window.
9. Select "Change recipe data records" as the "Authorization."

Result



Access to the "To Recipe view" button is protected. If, for example, the user "Smith" clicks the button in Runtime, the function "Recipe view" screen is called up. Prerequisite is that the user

"Smith" has logged on correctly and has the required authorization. The "Recipes" screen contains a recipe view and other screen objects.

If the logged-on user does not have the required authorization or if no user is logged on, the "Logon dialog box" is displayed.

Example: Creating user groups and assigning authorizations:

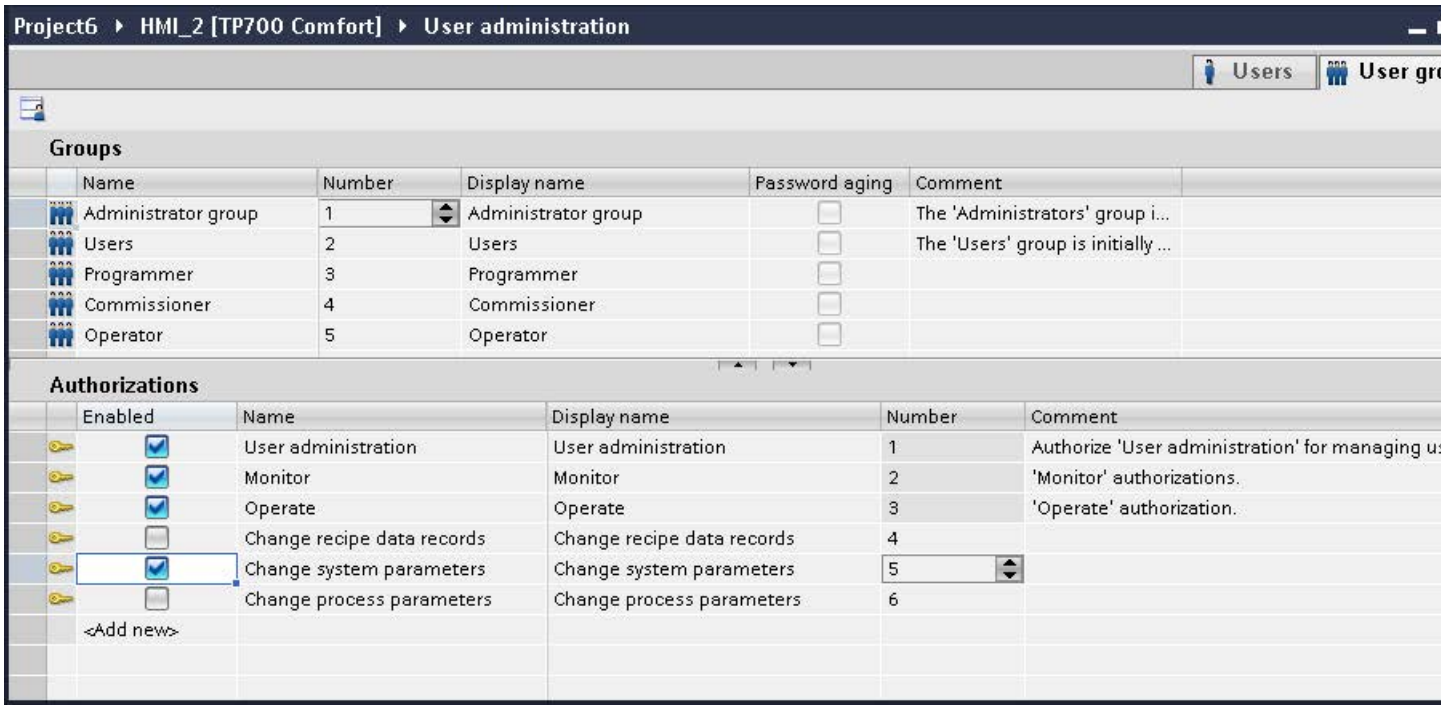
Task

In the following example you create the user groups and assign authorizations to them.

Procedure

1. Open the "User groups" work area.
2. Double-click "Add" in the "Groups" table.
3. Enter "Programmer" as the "Name".
4. Repeat steps 2 and 3 to create the "Commissioner" and "Operator" user groups.
5. Click "Administrator" in the "Groups" table.
6. Activate the "Change system parameters" authorization in the "Authorizations" table.

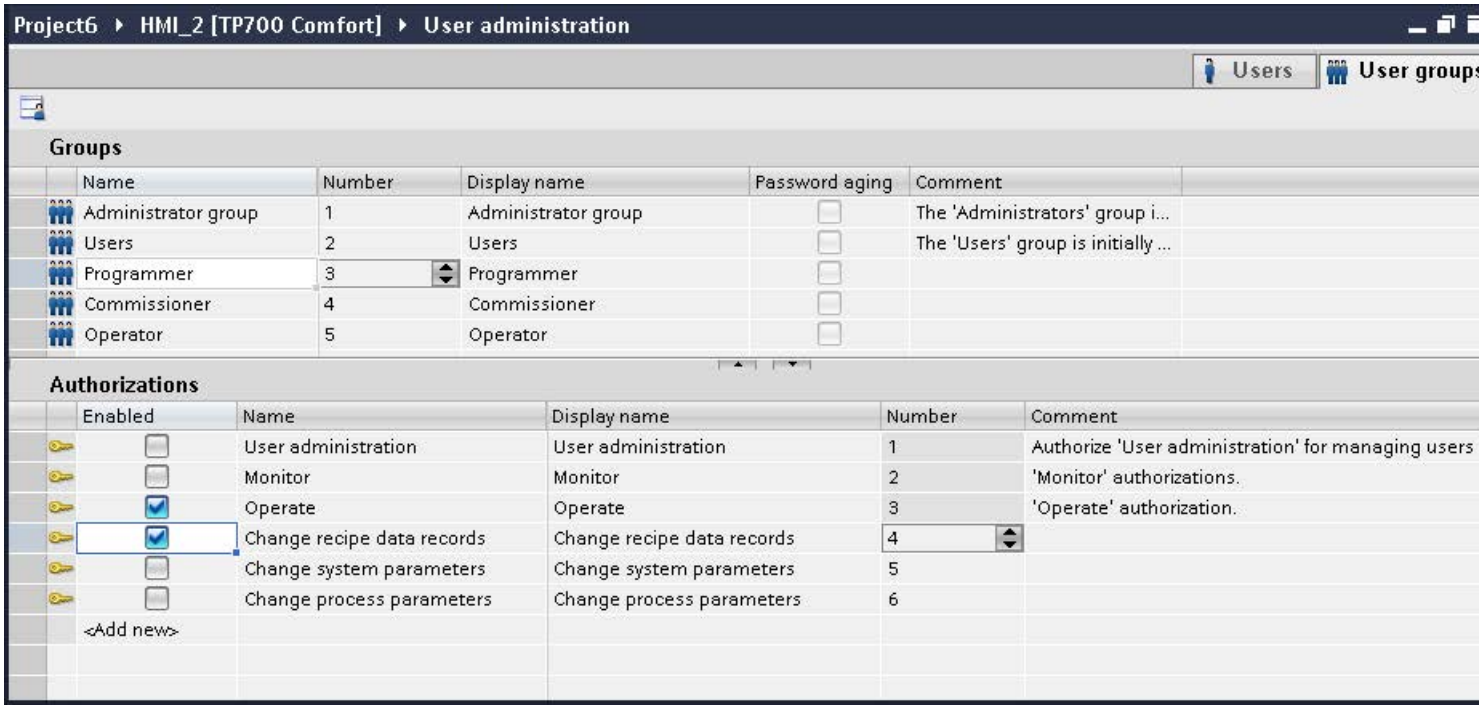
Interim result



Procedure

1. Click "Operator" in the "Groups" table.
2. Activate the "Change recipe data records" authorization in the "Authorizations" table.
3. Click "Commissioner" in the "Groups" table.
4. Activate the authorizations "Change recipe data records", "Change system parameters" and "Change process parameters" in the "Authorizations" table.
5. Click "Programmer" in the "Groups" table.
6. Activate the "Change recipe data records" authorization in the "Authorizations" table.

Result



Example: Creating users and assigning them to a user group

Task

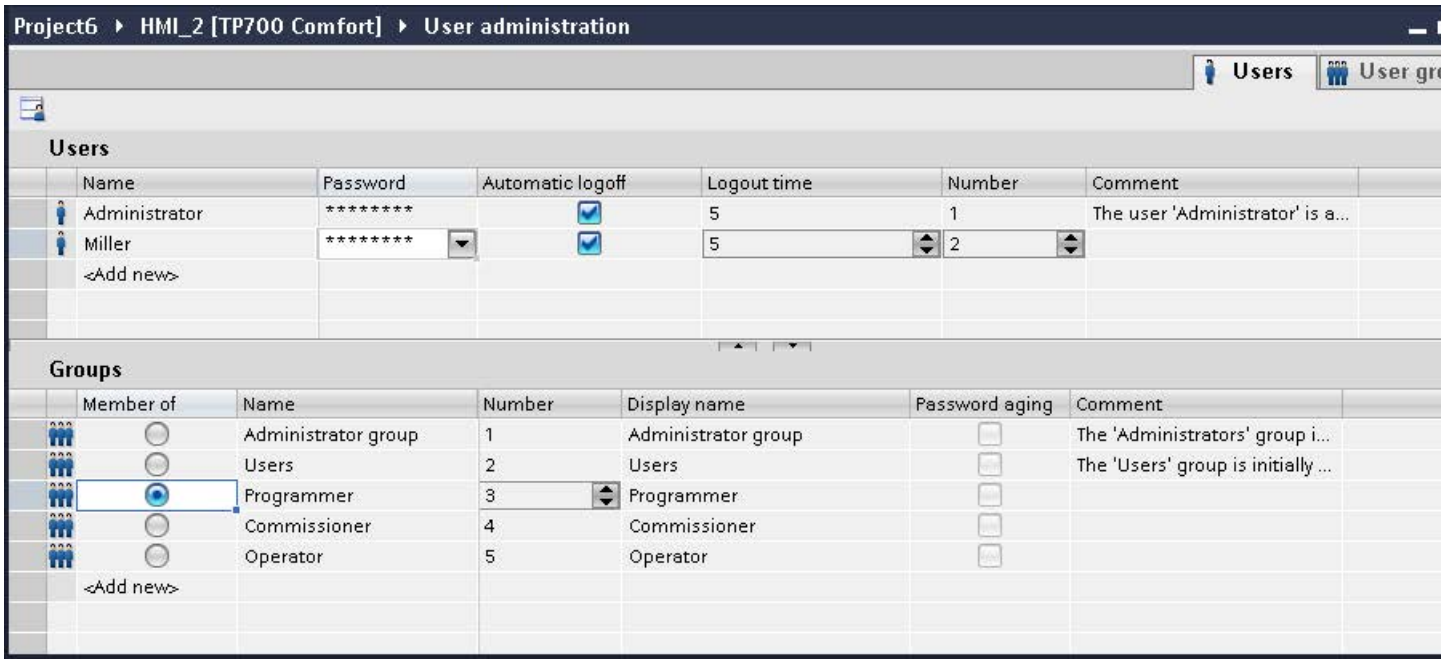
In the following example you create the users and assign user groups to them. The users are sorted alphabetically immediately after the name has been entered.

Procedure

1. Open the "Users" work area.
2. Double-click "Add" in the "Users" table.

3. Enter "Miller" as the user name.
4. Click the button in the "Password" column. The dialog box for entering the password is displayed.
5. Enter "miller" as the password.
6. To confirm the password enter it a second time in the lower field.
7. Close the dialog box by using the icon.
8. Activate the "Programmer" user group in the "Groups" table.

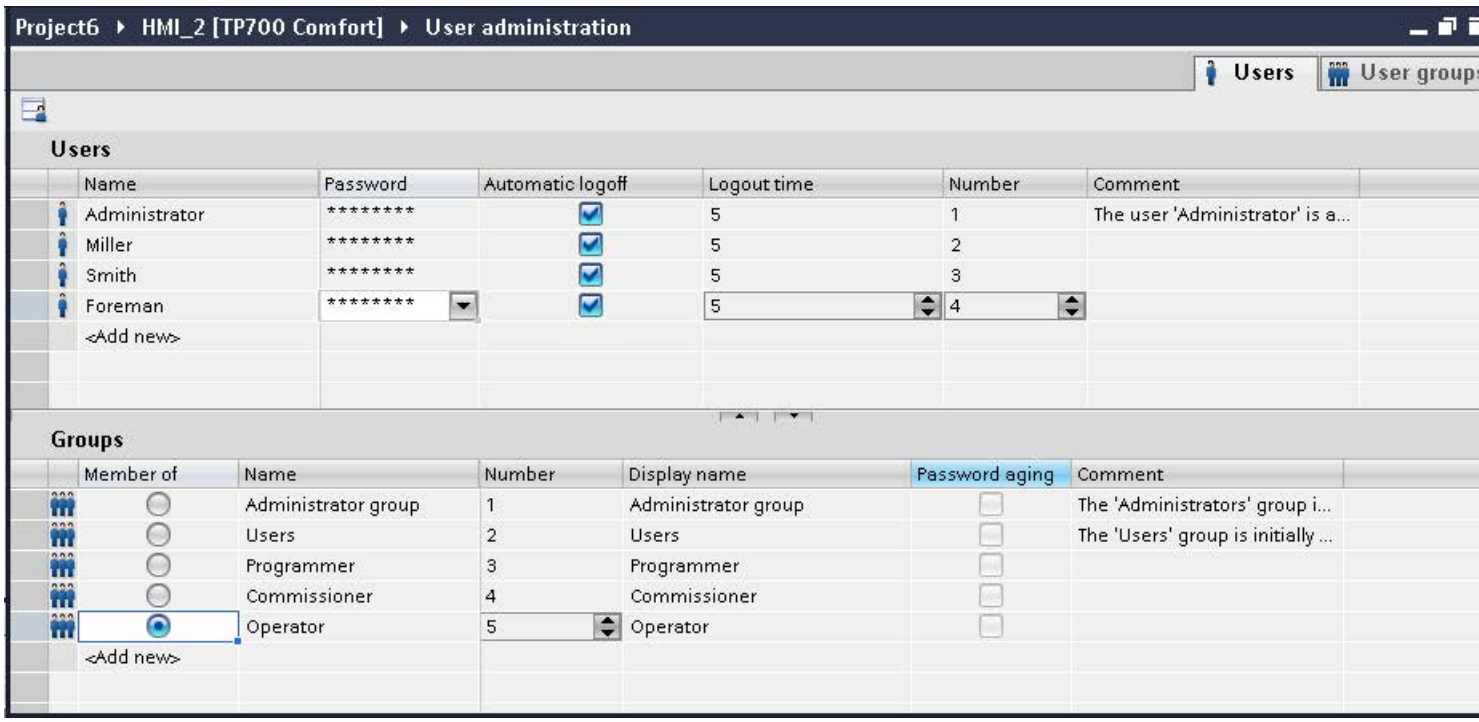
Interim result



Procedure

1. Double-click "Add" in the "Users" table.
2. Enter "Smith" as the user name.
3. Click the button in the "Password" column. The dialog box for entering the password is displayed.
4. Enter "smith" as the password.
5. To confirm the password enter it a second time in the lower field.
6. Close the dialog box by using the icon.
7. Activate the "Commissioner" user group in the "Groups" table.
8. Repeat steps 2 to 6 for the user "Foreman."
9. Activate the "Operators" user group in the "Groups" table.

Result



10.6 Working with system functions

10.6.1 Basics

10.6.1.1 System functions

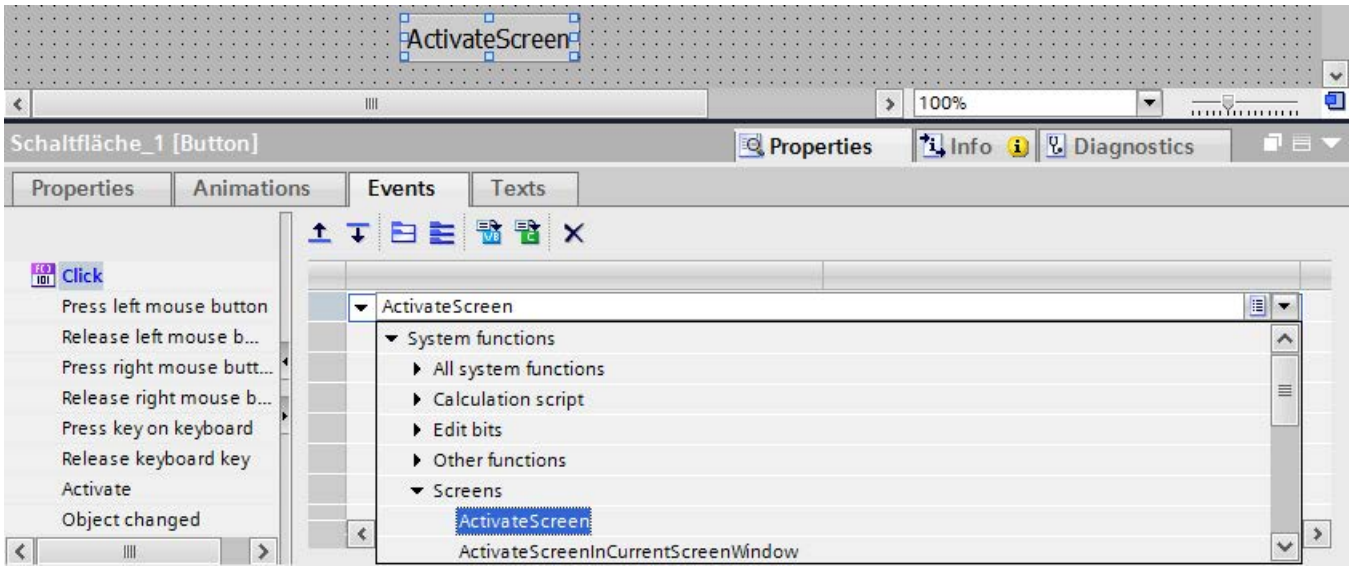
Introduction

System functions are functions supplied by WinCC. They are predefined and cannot be changed. You can use system functions to implement many tasks in Runtime even without having any programming knowledge, for example:

- Calculations, e.g. increasing a tag value by a specific or variable amount.
- Logging functions, e.g. starting a process value log.
- Settings, e.g. changing the PLC or setting a bit in the PLC.
- Alarms, e.g after a different user logs on.

Usage

You use system functions in a function list. When configuring a function list, you select the system functions from a selection list that is sorted by categories:



Each system function in WinCC is logically linked with an object and an event. As soon as the event occurs, the system function is triggered.

Language dependency

The names of the system functions are dependent on the set project language. The functionality can then be recognized immediately by the project planner.

Availability

You only configure system functions within a function list which are supported by the selected HMI device. If you use a project for several HMI devices, the system functions that are not supported by an HMI device are marked in color.

Events

The object and the selected function determine which events can be defined as triggers for executing a system function.

For example, the "Change value", "Low limit violated" and "Upper limit exceeded" events are associated with the "Tag" object. The "Loaded" and "Cleared" events are associated with the "Screen" object.

10.6.1.2 Use of system functions

Introduction

A function list is processed when a configured event occurs in runtime. The operator can trigger an event, for example, by pressing a function key on the HMI device. The system could also trigger an event if a process value falls below a specified limit, for example.

Applications

You can configure system functions on all the objects that are able to react to an event. You can use system functions directly in function lists and thereby control the sequence.

- Function list
  The system functions will be processed one line at a time in a function list. To avoid wait times, system functions in WinCC Runtime with a longer running time are executed simultaneously. For instance, a subsequent system function can already be performed even though the previous system function has not yet been completed.

An example for the configuration of a function list can be found under "Changing the operating mode on the HMI device with the current display".

10.6.2 Working with function lists

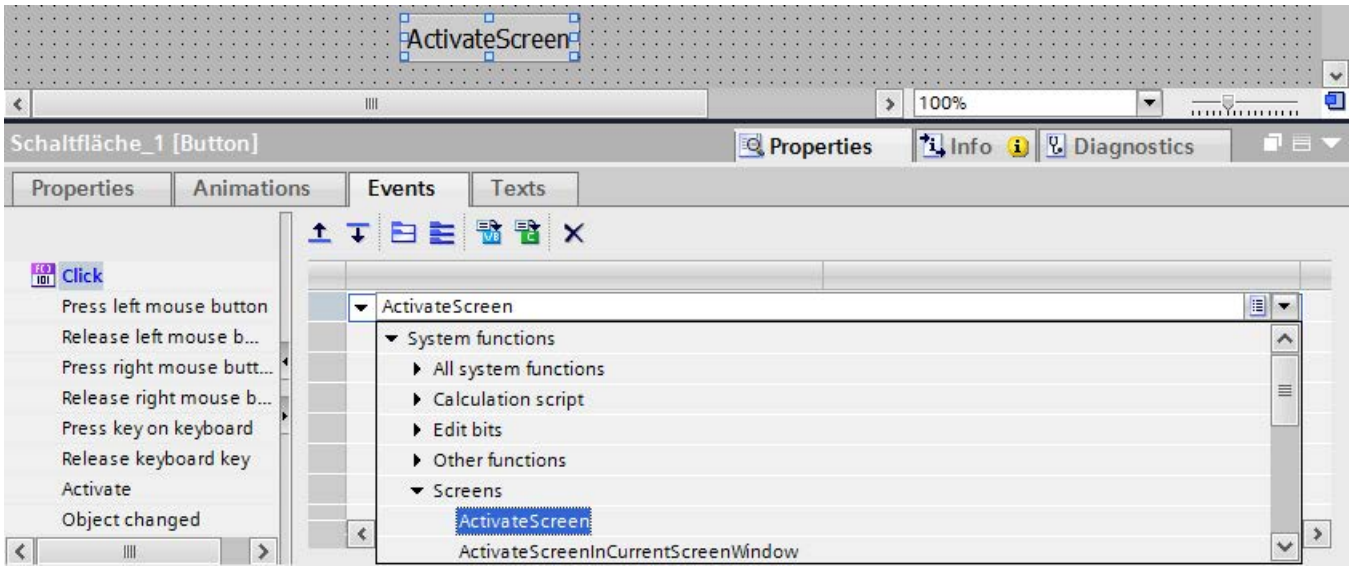
10.6.2.1 Basic of the functions list

Introduction

When the configured event occurs, several system functions can be performed with the function list.

Principle

The function list is configured for an event of an object, e.g. a screen object or a tag. The events which are available depend on the selected object and the HMI device.



Events occur only when the project is in Runtime. Events include:

- Value changes of a tag
- Pressing of a button
- Activation of Runtime

You can configure exactly one function list for each event.

Note

The choice of configurable system functions in a function list depends on the HMI device chosen.

10.6.2.2 Properties of a function list

Status information

During configuration the project data is tested in the background.

With the following causes the function list is not executed in Runtime and the incorrect entries are marked red:

- At least one system function is not completely supplied with parameters.
- At least one system function is contained which is not supported by the selected HMI device, for example, by changing the device type.

Executing system functions

System functions in a function list are executed in runtime sequentially from top to bottom. To avoid wait times, system functions with a longer running time (such as file operations) are processed simultaneously. For instance, a subsequent system function can already be performed even though the previous system function has not yet been completed.

Use a script with loops, conditional statements and abort conditions to program non-sequential and conditional procedures.

Note

Availability for specific devices

User-defined functions are not available on Basic Panels.

10.6.2.3 Configuring a function list

Introduction

You can configure a function list by selecting system functions from a drop-down list. The system functions are arranged in the drop-down list according to categories.

Requirement

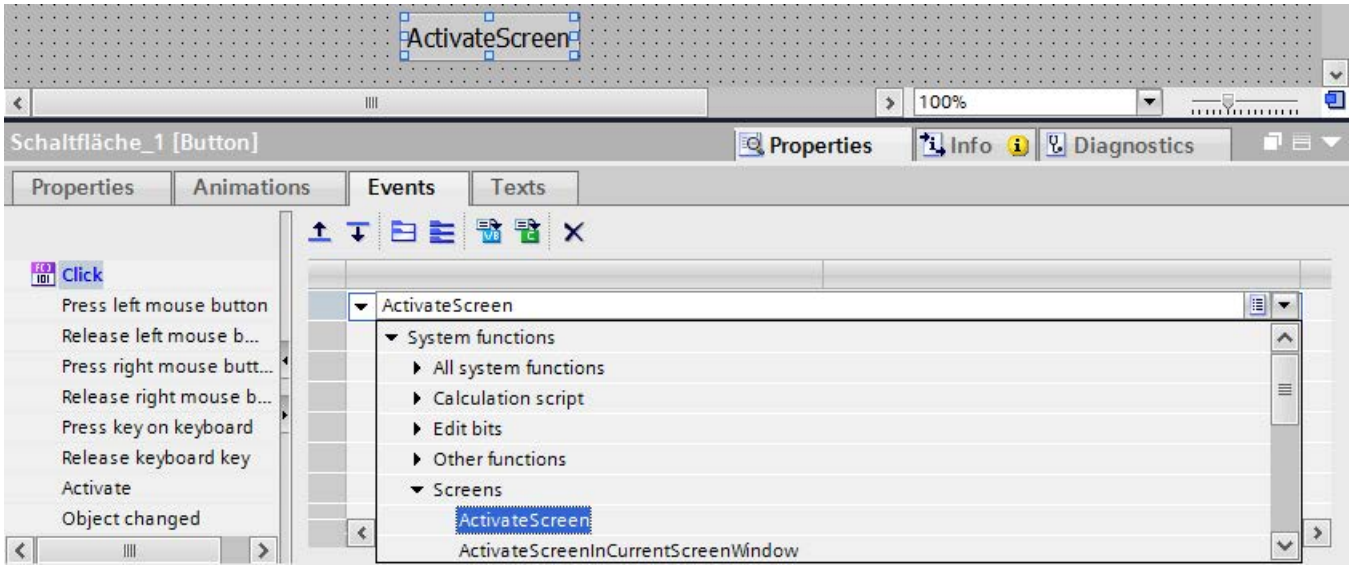
Object has at least one configurable event.

Procedure

Proceed as follows to configure a function list:

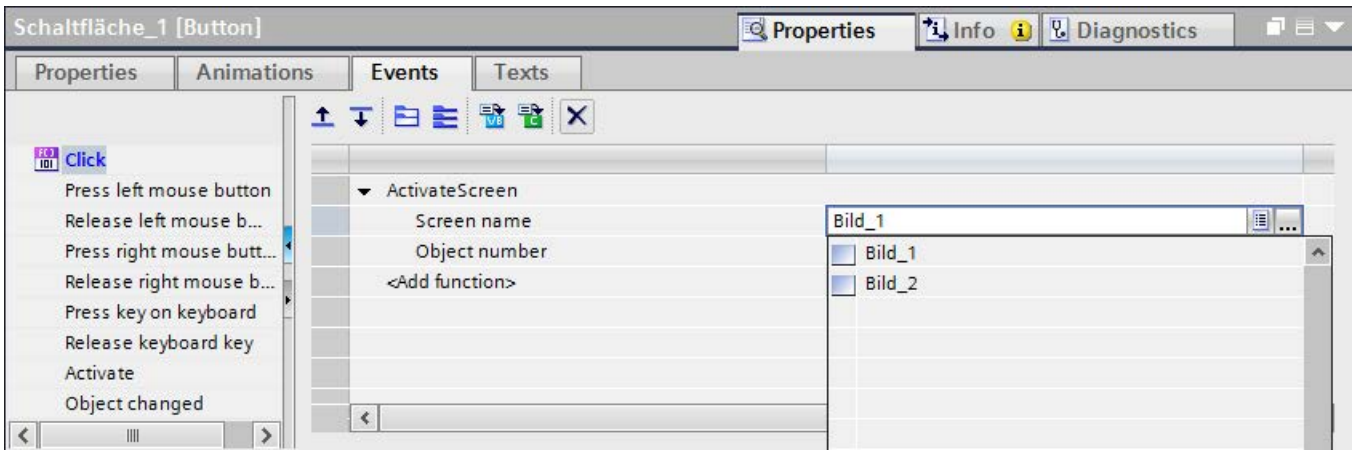
1. Open the editor in WinCC in which the object is located.
2. Select the object.
3. Click "Properties > Events" in the Inspector window. Select the event for which you want to configure the function list.
4. Select the "<No Function>" entry in the drop-down list of the Inspector window.
5. Select the desired system function.

6. You can also enter the name of the system function.



The system function is entered in the function list.

7. If the system function has parameters, specify the values for the parameters.



8. If you want to add other system functions or functions to the function list, then repeat steps four to seven.

Result

The function list is configured. In addition, to the configured event, the status of the function list is displayed in the Inspector window. When the configured event occurs in Runtime, the function list is completed from top to bottom.

10.6.2.4 Editing a function list

Introduction

A function list can be edited as follows:

- Changing the order of execution for system functions
- Removing a system function

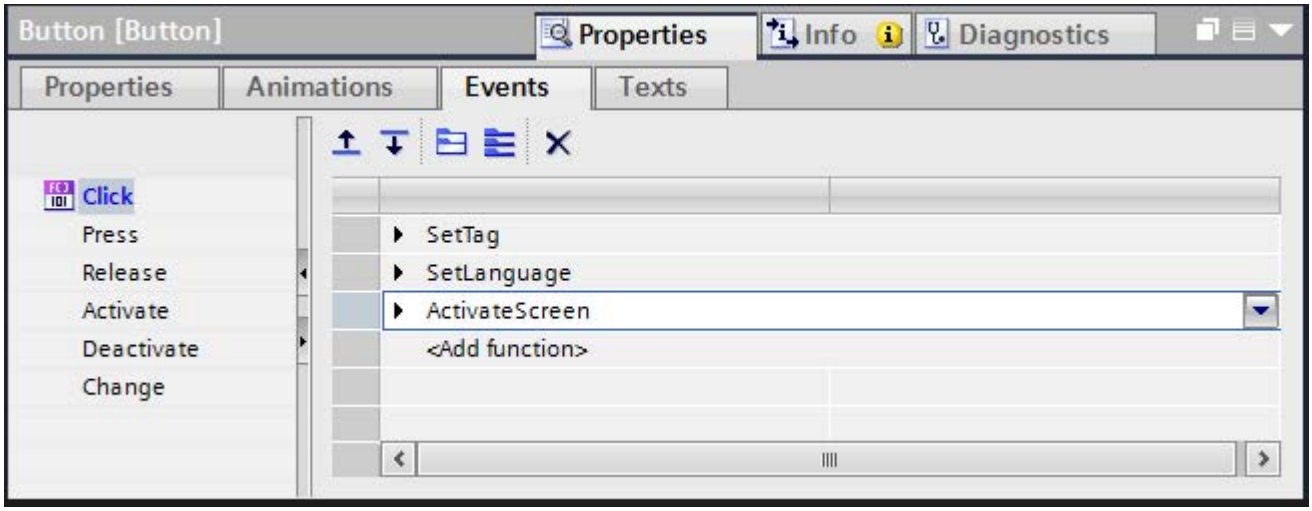
For additional information, refer to "Configuring a function list".

Requirement

The function list is configured.

Changing the order of a system function

1. Select the desired system function in the function list.
2. Then click the appropriate direction arrow in the inspector window until the system function or customized function is in the desired position.



Changing the order of several system functions

1. Hold down the <Shift> key.
2. Click the relevant system functions, working in succession.
3. Move the selection to the desired position by drag&drop.

Removing a system function

1. Select the desired system function in the function list.
2. Select "Delete" from the shortcut menu.

10.6.2.5 Executing a function list in Runtime

Principle

A function list is executed from top to bottom in Runtime. A distinction is made between synchronous and asynchronous execution, so that no waiting periods ensue during execution. The distinction is made by the system by evaluating the different runtimes of the system functions. User-defined functions are always executed synchronously independent of the runtime. If a system function returns an error status, the execution of the function list is cancelled.

Synchronous execution

During synchronous execution, the system functions in a function list are executed one after the other. The previous system function must be finished before the next system function can be executed.

Asynchronous execution

System functions that perform file operations such as saving and reading have a longer runtime than system functions that, for example, set a tag value.

Therefore, system functions with longer runtimes are executed asynchronously. For example, while a system function is writing a recipe data record to a storage medium, the next system function is already being executed. Due to the parallel execution of system functions, waiting periods at the HMI device are avoided.

10.6.3 Example

10.6.3.1 Changing the operating mode on the HMI device with the current display

Scheduled task

In this example, you use the "SetDeviceMode" system function to switch between the "online" and "offline" modes on the HMI device. You also display the current set operating mode on the HMI device.

Requirements

A screen has been created.

Settings

For this example you require a HMI-tag and a text list with the following settings:

HMI tag:

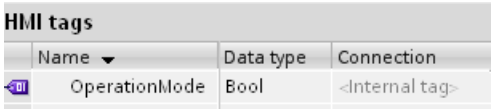
Name	PLC connection	Type
OperatingMode	No	Bool

Text list:

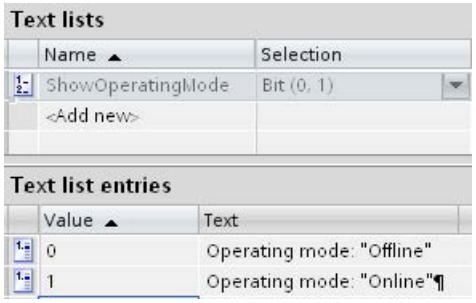
Name	Contains	Values
ShowOperatingMode	Bit (0/1)	1: Operating mode: "Online" 0: Operating mode: "Offline"

Procedure

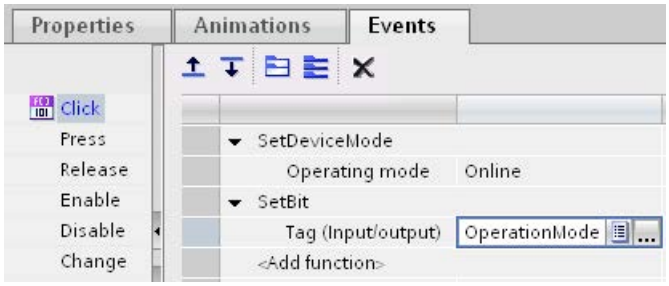
1. Create the "OperatingMode" HMI-tag indicated above.



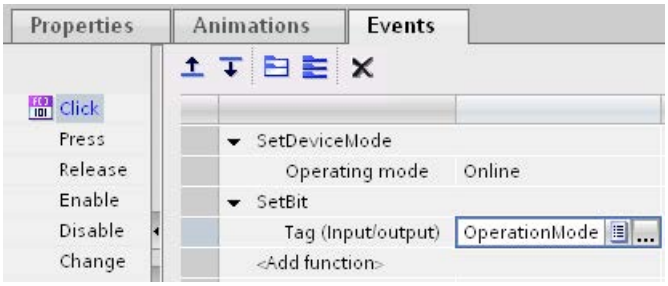
2. Create the "ShowOperatingMode" text list indicated above.



3. Open the screen and insert a button for which you configure the operating mode change to "online".
4. Click "Properties> Events" in the Inspector window. Select the "Press" event.
5. Configure the "SetDeviceMode" system function for the "Press" event. The system function is found in the selection list under "Settings".
6. For the "Mode" parameter, select the "Online" entry.
7. Configure the system function "SetBit" on the event "Press". The system function is found in the selection list under "Bit processing".
8. Select the HMI-tag "Operating mode" from the selection list for the parameter "Tag".



9. Add a button in the process screen for which you configure the operating mode change to "offline".
10. Repeat steps four to seven. For the "Mode" parameter, select the "Offline" entry. Configure the system function "ResetBit" in place of the system function "SetBit."



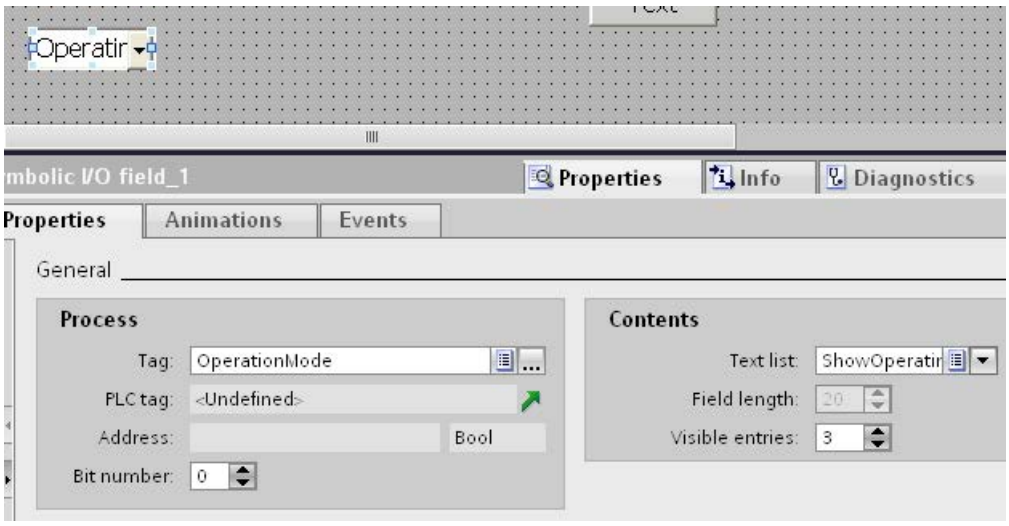
Interim result

You can toggle the operating mode of the HMI device with the two buttons in Runtime.

You want to display the current set operating mode in an output field on the HMI device.

Procedure

1. Create a "Symbolic I/O field" in the process image. Click "Properties > Properties" in the Inspector window.
2. Make the following settings in the "General" group:
   - Select "Output" as the "Mode".
   - Select the text list "Show operating mode" as "Text list".
   - Select "Operating mode" as "Tag".



Result

When you change the operating mode with the buttons, the currently set operating mode on the HMI device is always shown.

10.6.4 Reference

10.6.4.1 Function list

Device-based dependency of system functions

Availability of system functions

The following table shows the availability of system functions and user-defined functions on the HMI devices.

Technical data subject to change.

Overview

	KP300 Basic PN	KTP400 Basic PN	KTP 600 Basic DP	KTP 600 Basic PN	KTP 1000 Basic DP	KTP 1000 Basic PN	TP 1500 Basic PN
User-defined functions	No	No	No	No	No	No	No
Logoff (Page 8287)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
AdjustContrast (Page 8287)	Yes	Yes	No	Yes [1)]	Yes	Yes	Yes
ActivateScreen (Page 8288)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ActivateScreenByNumber (Page 8289)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ActivateCleanScreen (Page 8290)	No	Yes	Yes	Yes	Yes	Yes	Yes
ActivatePreviousScreen (Page 8291)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
UpdateTag (Page 8292)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Logon (Page 8292)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
EditAlarm (Page 8293)	Yes	Yes	Yes	Yes	Yes	Yes	No
ScreenObjectCursorDown (Page 8294)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ScreenObjectCursorUp (Page 8294)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ScreenObjectPageDown (Page 8295)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ScreenObjectPageUp (Page 8295)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
IncreaseFocusedValue (Page 8296)	Yes	Yes	Yes	Yes	Yes	Yes	No
IncreaseTag (Page 8297)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
GoToHome (Page 8297)	Yes	Yes	Yes	Yes	Yes	Yes	No
GoToEnd (Page 8298)	Yes	Yes	Yes	Yes	Yes	Yes	No

	KP300 Basic PN	KTP400 Basic PN	KTP 600 Basic DP	KTP 600 Basic PN	KTP 1000 Basic DP	KTP 1000 Basic PN	TP 1500 Basic PN
InvertBit (Page 8299)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
InvertBitInTag (Page 8300)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
InvertLinearScaling (Page 8301)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
CalibrateTouchScreen (Page 8302)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
TrendViewScrollForward (Page 8303)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
TrendViewScrollBack (Page 8304)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
TrendViewExtend (Page 8304)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
TrendViewCompress (Page 8305)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
TrendViewRulerRight (Page 8306)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
TrendViewRulerLeft (Page 8305)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
TrendViewSetRulerMode (Page 8306)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
TrendViewStartStop (Page 8307)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
TrendViewBackToBeginning (Page 8307)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
GetUserName (Page 8308)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
GetGroupNumber (Page 8309)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
GetPassword (Page 8309)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
LinearScaling (Page 8311)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ClearAlarmBuffer (Page 8312)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ClearAlarmBufferProTool (Page 8313)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
AlarmViewEditAlarm (Page 8314)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
AlarmViewAcknowledgeAlarm (Page 8315)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
AlarmViewShowOperatorNotes (Page 8315)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
AcknowledgeAlarm (Page 8316)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewNewDataRecord (Page 8317)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewGetDataRecordFromPLC (Page 8317)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewClearDataRecord (Page 8317)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewMenu (Page 8318)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewOpen (Page 8318)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewSetDataRecordToPLC (Page 8319)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewSaveDataRecord (Page 8320)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewSaveAsDataRecord (Page 8320)	Yes	Yes	Yes	Yes	Yes	Yes	Yes

	KP300 Basic PN	KTP400 Basic PN	KTP 600 Basic DP	KTP 600 Basic PN	KTP 1000 Basic DP	KTP 1000 Basic PN	TP 1500 Basic PN
RecipeViewRenameDataRecord (Page 8320)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewShowOperatorNotes (Page 8321)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
RecipeViewBack (Page 8321)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ResetBit (Page 8322)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ResetBitInTag (Page 8323)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
PressButton (Page 8324)	Yes	Yes	Yes	Yes	Yes	Yes	No
ReleaseButton (Page 8325)	Yes	Yes	Yes	Yes	Yes	Yes	No
ShiftAndMask (Page 8326)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
PageDown (Page 8328)	Yes	Yes	Yes	Yes	Yes	Yes	No
PageUp (Page 8327)	Yes	Yes	Yes	Yes	Yes	Yes	No
SetDeviceMode (Page 8328)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SetBit (Page 8329)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SetBitInTag (Page 8330)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SetBitWhileKeyPressed (Page 8332)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SetBacklightColor (Page 8333)	Yes	No	No	No	No	No	No
SetLanguage (Page 8335)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SetTag (Page 8336)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SetConnectionMode (Page 8337)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SimulateSystemKey (Page 8339)	Yes	Yes	Yes	Yes	Yes	Yes	No
SimulateTag (Page 8340)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
StopRuntime (Page 8341)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
TraceUserChange (Page 8342)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
DecreaseFocusedValue (Page 8342)	Yes	Yes	Yes	Yes	Yes	Yes	No
DecreaseTag (Page 8343)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ChangeConnection (Page 8343)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ShowLogonDialog (Page 8345)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ShowOperatorNotes (Page 8346)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ShowAlarmWindow (Page 8347)	Yes	Yes	Yes	Yes	Yes	Yes	Yes

1) For KTP600 Basic mono PN only.

System functions

Logoff

Description

Logs off the current user on the HMI device.

Use in the function list

Logoff

Use in user-defined functions

Logoff

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

See also

Device-based dependency of system functions (Page 8284)

Logon (Page 8292)

AdjustContrast

Description

Changes the contrast of the display one level on the HMI device.

Use in the function list

AdjustContrast (Adjust)

Use in user-defined functions

-

Parameters

Adjust

Specifies how the contrast is changed:

0 (hmiDecrease) = Decrease: Decreases the contrast one level.

1 (hmiIncrease) = Increase: Increases the contrast one level.

Application example

Objective

One button each for increasing and decreasing the screen contrast is desired.

Notes on configuring

Configure two buttons and configure the "AdjustContrast" system function on the "Press" event. The parameters "Increase" and "Decrease" are allocated.

Procedure on HMI device

When one of the two buttons is pressed in runtime, the contrast is increased or decreased one level.

See also

Device-based dependency of system functions (Page 8284)

Logoff (Page 8287)

ActivateScreen

Description

Performs a screen change to the given screen.

Use the "ActivateScreenByNumber" system function to change from the root screen to the permanent window or vice versa.

Use in the function list

ActivateScreen (Screen name, Object number)

Use in user-defined functions

ActivateScreen (Screen_name, Object_number)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Screen name

Name of the screen to which you change.

Object number

The operator control element which receives the focus in the given screen after the screen change. The number of the operator control element is to be determined using the tabulator sequence during configuration.

When "0" is specified:

- If the focus is in the permanent window when the system function is called up, the permanent window maintains the focus.
- If the focus is in the root screen when the system function is called up, the first operator control element in the given screen receives the focus.

Note

If the "Reach margin" event is assigned to the "ActivateScreen" system function, only the value "0" is valid for the "Object number" parameter. The active object is not defined by the object number, but rather by the X position it had prior to the screen change.

Example

The ActivateScreen function is used in the following program code to activate the "Screen_2" screen when you click a button.

```
{

// User defined code
// i.e. when pressing a button
ActivateScreen ("Screen_2", 0);
...
}
```

See also

Device-based dependency of system functions (Page 8284)

ActivateScreenByNumber (Page 8289)

ActivateScreenByNumber

Description

Performs a screen change to a screen depending on a tag value.

The screen is identified by its screen number.

Use in the function list

ActivateScreenByNumber (Screen number, Object number)

Use in user-defined functions

ActivateScreenByNumber (Screen_number, Object_number)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Screen number

Tag which contains the screen number of the destination screen.

When a change from the root screen to the permanent window is desired, "0" or "-1" is specified:

0 = Change from root screen to permanent window

-1 = Change from permanent window to root screen

Object number

The number of the screen object which receives the focus in the given screen after the screen change. The number of the operator control element is to be determined using the tabulator sequence during configuration.

When "0" is specified:

- If the focus is in the permanent window when the system function is called up, the permanent window maintains the focus.
- If the focus is in the root screen when the system function is called up, the first operator control element in the given screen receives the focus.

See also

Device-based dependency of system functions (Page 8284)

ActivateScreen (Page 8288)

ActivateCleanScreen

Description

Activates the clean screen on the HMI device. The display of the HMI device is disabled for the given time period.

When the display of the HMI device is deactivated, it can be cleaned without triggering touch functions by mistake.

Use in the function list

ActivateCleanScreen (Time period)

Use in user-defined functions

--

Parameters

Time period

Time period for which the display is disabled. The time remaining is displayed as a progress bar.

Value range in seconds from 10 through 300.

Note

The system function ActivateCleanScreen cannot be simulated.

See also

Logoff (Page 8287)

ActivatePreviousScreen

Description

Performs a screen change to the screen which was activated before the current screen. The screen change is not performed if no screen was activated beforehand.

The last 10 screens that were called up are saved. A system alarm is output when you change to a screen which is no longer saved.

Note

If you want to use the system function, the screen to which you change has to be used in the navigation structure.

Use in the function list

ActivatePreviousScreen

Use in user-defined functions

ActivatePreviousScreen

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

See also

Device-based dependency of system functions (Page 8284)

UpdateTag

Description

Reads the current value of the tag with the specified Update ID from the PLC.

Use in the function list

UpdateTag (Update ID)

Use in user-defined functions

-

Parameters

Update ID

Update ID assigned to the tag that will be updated.

See also

Logoff (Page 8287)

Logon

Description

Logs on the current user on the HMI device.

Use in the function list

Logon (Password, User name)

Use in user-defined functions

Logon (Password, User_name)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Password

The tag from which the password for the user logging on is read.

If the user is logged on, the password in the tag is deleted.

User name

The tag from which the user name for the user logging on is read.

See also

Device-based dependency of system functions (Page 8284)

Logoff (Page 8287)

EditAlarm

Description

Triggers the "Edit" event for all selected alarms.

If the alarms to be edited have not yet been acknowledged, the acknowledgment takes place automatically when this system function is called up.

This system function can only be used for function keys.

Use in the function list

EditAlarm

Use in user-defined functions

EditAlarm

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

See also

Device-based dependency of system functions (Page 8284)

ScreenObjectCursorUp

Description

Results in a line-by-line, upward cursor movement in the specified screen object.

The system function can be used for the following screen objects:

- User view
- Alarm view
- Recipe view

Use in the function list

ScreenObjectCursorUp (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the screen object in which the key function is triggered.

ScreenObjectCursorDown

Description

Results in a line-by-line, downward cursor movement in the specified screen object.

The system function can be used for the following screen objects:

- User view
- Alarm view
- Recipe view

Use in the function list

ScreenObjectCursorDown (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the screen object in which the key function is triggered.

ScreenObjectPageUp

Description

Results in a page-by-page, upward cursor movement in the specified screen object.

The system function can be used for the following screen objects:

- User view
- Alarm view
- Recipe view

Use in the function list

ScreenObjectPageUp (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the screen object in which the key function is triggered.

ScreenObjectPageDown

Description

Results in a page-by-page, downward cursor movement in the specified screen object.

The system function can be used for the following screen objects:

- User view
- Alarm view
- Recipe view

Use in the function list

ScreenObjectPageDown (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the screen object in which the key function is triggered.

IncreaseFocusedValue

Description

Adds the given value to the value of the tag which is connected to the input field (drop-down list, graphic selection list, slider bar) which has the current focus.

This system function can only be used for function keys.

Use in the function list

IncreaseFocusedValue (Value)

Use in user-defined functions

-

Parameters

Value

The value which is added to the tag value.

See also

Device-based dependency of system functions (Page 8284)

IncreaseTag

Description

Adds the given value to the value of the tags.

X = X + a

Note

The system function uses the same tag as input and output values. When this system function is used to convert a value, help tags must be used. You can use the "SetTag" system function to assign the tag value to the auxiliary tags.

If you configure the system function on the events of an alarm and the tag is not being used in the current screen, it is not ensured that the actual value of the tags is being used in the PLC. You can improve the situation by setting the "Cyclic continuous" acquisition mode.

Use in the function list

IncreaseTag (Tag, Value)

Use in user-defined functions

IncreaseTag (Tag, Value)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Tag

The tag to which the given value is added.

Value

The value that is added.

See also

Device-based dependency of system functions (Page 8284)

SetTag (Page 8336)

GoToHome

Description

Executes the key function <Home> on the HMI device.

This system function is used when the HMI device does not have this functionality by default.

The system function can only be used for the following function keys.

Use in the function list

GoToHome

Use in user-defined functions

GoToHome

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

See also

Device-based dependency of system functions (Page 8284)

GoToEnd

Description

Executes the key function <End> on the HMI device.

This system function is used when the HMI device does not have this functionality by default.

The system function can only be used for the following function keys.

Use in the function list

GoToEnd

Use in user-defined functions

GoToEnd

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

See also

Device-based dependency of system functions (Page 8284)

InvertBit

Description

Inverts the value of the given tag of the "Bool" type:

- If the tag has the value of 1 (TRUE), it will be set to 0 (FALSE).
- If the tag has the value of 0 (FALSE), it will be set to 1 (TRUE).

Use in the function list

InvertBit (Tag)

Use in user-defined functions

InvertBit (Tag)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The tag whose bit is set.

Example

The following program code inverts the value of the boolean tag b_value and outputs the result along with the original b_saved value.

```
{
BOOL b_value = 0;
BOOL b_saved = b_value;

//Invert variable
invertBit(b_value);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",b_value, b_saved);
  ...
}
```

See also

Device-based dependency of system functions (Page 8284)

InvertBitInTag (Page 8300)

InvertBitInTag

Description

Inverts a bit in the given tag:

- If the bit in the tag has the value of 1 (TRUE), it will be set to 0 (FALSE).
- If the bit in the tag has the value of 0 (FALSE), it will be set to 1 (TRUE).

After changing the given bit, the system function transfers the entire tag back to the PLC. It is not checked whether other bits in the tags have changed in the meantime. Operator and PLC have read-only access to the indicated tag until it is transferred back to the PLC.

Note

If the PLC supports BOOL tags, do not use this system function. Use the "InvertBit" system function instead.

Use in the function list

InvertBitInTag (Tag, Bit)

Use in user-defined functions

InvertBitInTag (Tag, Bit)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The tag in which the given bit is set.

Bit

The number of the bit that is set.

When this system function is used in a user-defined function, the bits in a tag are counted from right to left. The counting begins with 0.

Example

The following program code inverts a bit at the specified bitposition in the bvalue tag and outputs the result along with the original bsaved value.

```
{
BYTE bvalue;
BYTE bsaved = bvalue;
BYTE bitposition = 2;

//Invert bit in bitposition
InvertBitInTag (bvalue, bitposition);
//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bvalue, bsaved);
  ...
}
```

See also

Device-based dependency of system functions (Page 8284)

InvertBit (Page 8299)

InvertLinearScaling

Description

Assigns a value to the tag X, which is calculated from the value of the given tag Y using the linear function X = (Y - b) / a.

The tags X and Y must not be identical. This system function is the inverse of the "LinearScaling" system function.

Note

The tags X and Y must not be identical. If a tag is to be converted into itself, a auxiliary tag must be used.

The "SetTag" system function can be used to assign the value of the tags to be converted to the auxiliary tags.

Use in the function list

InvertLinearScaling (X, Y, b, a)

Use in user-defined functions

InverseLinearScaling (X, Y, b, a)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

X

The tag which is assigned the value calculated from the linear equation.

Y

The tag that contains the value used for calculation.

b

The value which is subtracted.

a

The value through which is divided.

Example

The following program code assigns a value to the varX tag by means of the InverseLinearScaling function.

```
{
BYTE varX;
BYTE Yvalue = 10;
BYTE bvalue = 3;
BYTE avalue = 4;

//Inverse linear scaling
 InverseLinearScaling (varX, Yvalue, bvalue, avalue);

printf ("varX = %d\r\n, varX);
...
}
```

The saved return value can be processed in the following code.

See also

Device-based dependency of system functions (Page 8284)

LinearScaling (Page 8311)

CalibrateTouchScreen

Description

Calls a program for calibrating the touch screen.

During the calibration process, there is a prompt to touch five positions on the screen display. Touch the screen display within 30 seconds, to confirm the calibration process. If the calibration is not completed within this time span, the calibration settings are discarded. The user prompt is in English.

Use this system function the first time you start the HMI device.

Use in the function list

CalibrateTouchScreen

Use in user-defined functions

CalibrateTouchScreen

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

Note

The CalibrateTouchScreen system function cannot be simulated.

See also

Device-based dependency of system functions (Page 8284)

TrendViewScrollForward

Description

Scrolls forward one display width in the Trend view.

Use in the function list

TrendViewScrollForward (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the trend view in which is scrolled forward.

TrendViewScrollBack

Description

Scrolls back one display width to the left in the trend view.

Use in the function list

TrendViewScrollBack (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the trend view in which is scrolled back.

TrendViewExtend

Description

Reduces the time period which is displayed in the trend view.

Use in the function list

TrendViewExtend (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the trend view in which the displayed time period is reduced.

TrendViewCompress

Description

Increases the time period which is displayed in the trend view.

Use in the function list

TrendViewCompress (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the trend view in which the displayed time period is increased.

TrendViewRulerLeft

Description

Moves the read-line backwards (to the left) in the trend view.

Note

In order to be able to move the read-line, the read-line must have been switched on. This is done using the "TrendViewSetRulerMode" system function.

Use in the function list

TrendViewRulerLeft (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the trend view in which the read-line is moved backwards.

See also

TrendViewSetRulerMode (Page 8306)

TrendViewRulerRight

Description

Moves the read-line forwards (to the right) in the trend view.

Note

In order to be able to move the read-line, the read-line must have been switched on. This is done using the "TrendViewSetRulerMode" system function.

Use in the function list

TrendViewRulerRight (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the trend view in which the read-line is moved forward.

See also

TrendViewSetRulerMode (Page 8306)

TrendViewSetRulerMode

Description

Hides or shows the read-line in the trend view. The read-line displays the Y value belonging to the X value.

Note

To ensure that the ruler is displayed, you have to activate the setting "Show ruler" in the trend view properties.

Use in the function list

TrendViewSetRulerMode (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the trend view in which the read-line is hidden or shown.

See also

TrendViewRulerLeft (Page 8305)
TrendViewRulerRight (Page 8306)

TrendViewStartStop

Description

Stops the trend recording or continues the trend recording in the trend view.

Use in the function list

TrendViewStartStop (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the trend view in which the recording of the trend is started or stopped.

TrendViewBackToBeginning

Description

Scrolls back to the beginning of the display range in the trend view.

Use in the function list

TrendViewBackToBeginning (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the trend view in which you scroll to the beginning of the display range.

GetUserName

Description

Writes the user name of the user currently logged on to the HMI device in the given tag.

If the given tag has a control connection, the user name is also available in the PLC connection. This system function makes it possible, for example, to implement a user-dependent release of certain functionalities.

Use in the function list

GetUserName (Tag)

Use in user-defined functions

GetUserName (Tag)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Tag

The tag to which the user name is written.

See also

Device-based dependency of system functions (Page 8284)

GetGroupNumber

Description

Reads the number of the group to which the user logged on to the HMI device belongs, and writes it to the given tag.

Use in the function list

GetGroupNumber (Tag)

Use in user-defined functions

GetGroupNumber (Tag)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Tag

The tag to which the number of the group is written.

See also

Device-based dependency of system functions (Page 8284)

GetPassword

Description

Writes the password of the user currently logged on to the HMI device in the given tag.

Note

Make sure that the value of the given tag is not displayed in another place in the project.

Note

The passwords of SIMATIC Logon users cannot be read.

Use in the function list

GetPassword (Tag)

Use in user-defined functions

GetPassword (Tag)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Tag

The tag to which the password is written.

See also

Device-based dependency of system functions (Page 8284)

ReadPLCMode

Description

Evaluates the current state of the connected PLC.

The system function "ReadPLCMode" can only be configured for the following PLCs:

- SIMATIC S7-1200
- SIMATIC S7-1500

Use in the function list

ReadPLCMode (connection, mode)

Use in user-defined functions

GetPLCMode (Connection, Mode)

Parameters

Connection

Connection of PLC and HMI device.

Mode

Evaluates the mode of the connected PLC.
8: RUN = The PLC program is being executed.
4: STOP = The PLC program has been interrupted.
Select a suitable user variable for the evaluation.

LinearScaling

Description

Assigns a value to the tag Y, which is calculated from the value of the given tag X using the linear function Y= (a *X) + b.

The inverse of this function is the "InvertLinearScaling" system function.

Note

The tags X and Y must not be identical. If a tag is to be converted into itself, a auxiliary tag must be used.

The "SetTag" system function can be used to assign the value of the tags to be converted to the auxiliary tags.

Use in the function list

LinearScaling (Y, a, X, b)

Use in user-defined functions

LinearScaling (Y, a, X, b)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Y

The tag which is assigned the value calculated from the linear equation.

a

The value with which is multiplied.

X

The tag that contains the value used for calculation.

b

The value that is added.

Example

The following program code uses the LinearScaling function to assign a value to the Yvar tag.

```
{
```

```
BYTE Yvar;
BYTE Xvalue = 10;
BYTE bvalue = 3;
BYTE avalue = 4;

// linear scaling
 LinearScaling ( Yvar, avalue, Xvalue, bvalue);

printf ("Yvar = %d\r\n, Yvar);
...
}
```

The saved return value can be processed in the following code.

See also

Device-based dependency of system functions (Page 8284)

InvertLinearScaling (Page 8301)

ClearAlarmBuffer

Description

Deletes alarms from the alarm buffer on the HMI device.

Note

Alarms which have not yet been acknowledged are also deleted.

Use in the function list

ClearAlarmBuffer (Alarm class number)

Use in user-defined functions

ClearAlarmBuffer (Alarm_class_number)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Alarm class number

Determines which alarms are to be deleted from the alarm buffer:

0 (hmiAll) = All alarms/events

1 (hmiAlarms) = Alarms of alarm class "Errors"

2 (hmiEvents) = Alarms of alarm class "Warnings"

3 (hmiSystem) = Alarms of alarm class "System"

4 (hmiS7Diagnosis) = Alarms of alarm class "Diagnosis Events"

Note

Device dependency

Alarms of alarm class "Diagnosis Events" are not available on Basic Panels.

See also

Device-based dependency of system functions (Page 8284)

ClearAlarmBufferProTool

Description

The system function exists to ensure compatibility.

It has the same functionality as the "ClearAlarmBuffer" system function, but uses the old ProTool numbering.

Use in the function list

ClearAlarmBufferProTool (Alarm class number)

Use in user-defined functions

ClearAlarmBufferProtoolLegacy (Alarm_class_number)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Alarm class number

Alarm class number whose messages are to be deleted:

-1 (hmiAllProtoolLegacy) = All alarms/events

0 (hmiAlarmsProtoolLegacy) = Alarms of alarm class "Errors"

1 (hmiEventsProtoolLegacy) = Alarms of alarm class "Warnings"

2 (hmiSystemProtoolLegacy) = Alarms of alarm class "System"

3 (hmiS7DiagnosisProtoolLegacy) = Alarms of alarm class "Diagnosis Events"

Note

Device dependency

Alarms of alarm class "Diagnosis Events" are not available on Basic Panels.

See also

Device-based dependency of system functions (Page 8284)

ClearAlarmBuffer (Page 8312)

AlarmViewEditAlarm

Description

Triggers the event "Edit" for all alarms selected in the given alarm view.

This system function is used when the integrated button of the ActiveX control should not be used.

A system function can in turn be configured on the "Edit" event. For example, it is possible to change to the process screen in which the alarm appeared.

Note

If the alarms to be edited have not yet been acknowledged, the acknowledgment takes place automatically when this system function is called up.

Use in the function list

AlarmViewEditAlarm (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the alarm view in which the event is triggered.

Note

The HMI devices listed below do not support this system function for the "screen" object: OP 73, OP 77A, TP 177A.

AlarmViewAcknowledgeAlarm

Description

Acknowledges the alarms selected in the given alarm view.

This system function is used when the integrated button of the ActiveX control should not be used.

Use in the function list

AlarmViewAcknowledgeAlarm (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the alarm view in which the event is triggered.

Note

The HMI devices listed below do not support this system function for the "screen" object: OP 73, OP 77A, TP 177A.

AlarmViewShowOperatorNotes

Description

Displays the configured tooltip of the alarm selected in the given alarm view.

Use in the function list

AlarmViewShowOperatorNotes (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the alarm view in which the event is triggered.

Note

The HMI devices listed below do not support this system function for the "screen" object: OP 73, OP 77A, TP 177A.

AcknowledgeAlarm

Description

Acknowledges all selected alarms.

This system function is used when the HMI device does not have an ACK key or when the integrated key of the alarm view should not be used.

This system function can only be used for function keys.

Use in the function list

AcknowledgeAlarm

Use in user-defined functions

AcknowledgeAlarm

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

--

See also

Device-based dependency of system functions (Page 8284)

RecipeViewNewDataRecord

Description

Creates a new data record in the given recipe view.

Use in the function list

RecipeViewNewDataRecord (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view in which the new recipe data record is created.

RecipeViewGetDataRecordFromPLC

Description

Transfers the data record that is currently loaded in the PLC to the HMI device and displays it in the recipe view.

Use in the function list

RecipeViewGetDataRecordFromPLC (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view in which the recipe data record from the PLC is displayed.

RecipeViewClearDataRecord

Description

Deletes the data record which is currently displayed in the recipe view.

Use in the function list

RecipeViewClearDataRecord (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view in which the displayed recipe data record is deleted.

RecipeViewMenu

Description

Opens the menu of the specified simple recipe view.

Only use this system function at a simple recipe view.

Use in the function list

RecipeViewMenu (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view in which the menu is to be opened.

RecipeViewOpen

Description

Displays the data record values in the given recipe view or changes to the next selection field. The system function has no effect if the selection field for the recipe data record values is displayed on the HMI device.

Operation sequence of the selection lists in runtime:

- Recipe name
- Data record name
- RecipeDataRecordValues

This system function is used when a simple recipe view has been configured. In the simple recipe view, only one selection list is displayed at a time on the HMI device. Use the "RecipeViewBack" system function to display the previous selection list.

Use in the function list

RecipeViewOpen (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view in which the command is triggered.

See also

RecipeViewBack (Page 8321)

RecipeViewSetDataRecordToPLC

Description

Transfers the recipe data record which is currently displayed in the recipe view to the PLC.

Use in the function list

RecipeViewSetDataRecordToPLC (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view from which the recipe data record is transferred to the connected PLC.

RecipeViewSaveDataRecord

Description

Saves the recipe data record which is currently displayed in the recipe view.

Use in the function list

RecipeViewSaveDataRecord (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view in which the recipe data record is saved.

RecipeViewSaveAsDataRecord

Description

Saves the data record currently being displayed in the recipe view under a new name.

Use in the function list

RecipeViewSaveAsDataRecord (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Saves the data record currently being displayed in the recipe view under a new name and/or new number.

RecipeViewRenameDataRecord

Description

Renames the selected data record in the given recipe view.

Only use this system function at a simple recipe view.

Use in the function list

RecipeViewRenameDataRecord (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view in which the recipe data record is renamed.

RecipeViewShowOperatorNotes

Description

Displays the configured tooltip of the specified recipe view.

Use in the function list

RecipeViewShowOperatorNotes (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view whose configured tooltip is displayed.

RecipeViewBack

Description

Returns to the previous selection list in the given recipe view.

The system function has no effect if the selection field for the recipe is displayed on the HMI device. Operation sequence of the selection lists in runtime:

- Recipe name
- Data record name
- RecipeDataRecordValues

This system function is used when a simple recipe view has been configured. In the simple recipe view, only one selection list is displayed at a time on the HMI device. Use the "RecipeViewOpen" system function to display the recipe data record values or the next selection field.

Use in the function list

RecipeViewBack (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the recipe view in which the command is triggered.

ResetBit

Description

Sets the value of a "Bool" type tag to 0 (FALSE).

Use in the function list

ResetBit (Tag)

Use in user-defined functions

ResetBit (Tag)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The BOOL type tag which is set to 0 (FALSE).

Example

The following program code resets the value of the boolean tag b_value to 0 by means of the ResetBit function and outputs the result along with the original b_saved value.

```
{
BOOL b_value = 1;
BOOL b_saved = b_value;

//Reset bit
  ResetBit (b_value);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",b_value, b_saved);
...
}
```

See also

Device-based dependency of system functions (Page 8284)

ResetBitInTag (Page 8323)

ResetBitInTag

Description

Sets a bit in the specified tag to 0 (FALSE).

After changing the given bit, the system function transfers the entire tag back to the PLC. It is not checked whether other bits in the tags have changed in the meantime. Operator and PLC have read-only access to the indicated tag until it is transferred back to the PLC.

Note

If the PLC supports BOOL tags, do not use this system function. Use the "ResetBit" system function instead.

Use in the function list

ResetBitInTag (Tag, Bit)

Use in user-defined functions

ResetBitInTag (Tag, Bit)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The tag in which a bit is set to 0 (FALSE).

Bit

The number of the bit that is set to 0 (FALSE).

When this system function is used in a user-defined function, the bits in the specified tag will be counted from right to left independent of the PLC used. The counting begins with 0.

Example

The following program code sets a bit at the specified bitposition in the bvalue tag to 0 and outputs the result along with the original bsaved value.

```
{
BYTE bvalue;
BYTE bsaved = bvalue;
BYTE bitposition = 2;

//Reset bit in bitposition
ResetBitInTag (bvalue, bitposition);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bvalue, bsaved);
  ...
}
```

See also

Device-based dependency of system functions (Page 8284)

ResetBit (Page 8322)

PressButton

Description

The system function can only be configured on the function keys of an HMI device and triggers the "Press key" event at the specified screen object.

Use this system function when you want to operate a button in a screen with a function key of the HMI device, for example.

Note

The "PressButton" and "ReleaseButton" system functions must always be configured together. If you configure the "PressButton" system function on the "Press key" event for a function key, the "ReleaseButton" system function is configured on the "Release" event for the same function key.

Use in the function list

PressButton (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the screen object on which the event is triggered.

ReleaseButton

Description

The system function can only be configured on the function keys of an HMI device and triggers the "Release button" event at the specified screen object.

Use this system function when you want to operate a button in a screen with a function key of the HMI device, for example.

Note

The "PressButton" and "ReleaseButton" system functions must always be configured together. If you configure the "PressButton" system function on the "Press key" event for a function key, then the "ReleaseButton" system function is configured on the "Release key" event for the same function key.

Use in the function list

ReleaseButton (Screen object)

Use in user-defined functions

-

Parameters

Screen object

Name of the screen object on which the event is triggered.

ShiftAndMask

Description

This system function converts the input bit pattern of the source tags into an output bit pattern of the target tags. This involves bit shifting and masking.

Note

If the source and target tag have a different number of bits, using the system function in the target tag can result in a violation of the value range.

Use in the function list

ShiftAndMask (Source tag, Target tag, Bits to shift, Bits to mask)

Use in user-defined functions

ShiftAndMask (Source_tag, Target_tag, Bits_to_shift, Bits_to_mask)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Source tag

The tag includes the input bit pattern. Integer-type tags, e.g. "Byte", "Char", "Int", "UInt", "Long" and "ULong" are permitted.

Example: The actual value 72 is set at the 16-bit integer source tag: 0000000001001000.

Target tag

The output bit pattern is saved in the tag. Integer type tags, e.g. "Byte", "Char", "Int", "UInt", "Long" and "ULong" are permitted.

Example: The shifted input bit pattern is multiplied by the bit mask, with bit-by-bit logical AND operation: 0000000000001001. The resultant decimal value "8" is saved to the target tag.

Please note the following:

- The source and target tags have the same number of bits.
- The number of bits to shift is less than the number of bits in the source tag and target tag.
- Bits to mask does not have more bits than the source tag and the target tag.

Bits to shift

Number of bits by which the input bit pattern is shifted right. A negative value shifts the input bit pattern to the left.

Example: "Bits to shift" has the value "+3". The input bit pattern is shifted right by three bits when the system function is called: 0000000000001001.
Bits to the left are padded with "0". Three bits are truncated on the right. The new decimal value is "9".

Note

The left bit is "1" in a source tag of the data type with negative signed integer. This sign bit is padded with "0" when the bits are shifted right. The sign changes to "+".

Bits to mask

An integer serves as bit mask. The bit pattern is used to multiply the shifted input bit pattern. Example: Integer "2478" with the bit pattern "0000100110101110".

You can enter the bit mask in three different ways:

- Hexadecimal: First enter the prefix "0h" or "0H", followed by an optional space for better readability. Then group the bit pattern in blocks of four (0000)(1001)(1010)(1110) and set each block in hexadecimal code: (0)(9)(A)(E). Only the characters 0-9, A-F, a-f are allowed: "0h 09AE".
- Binary: First enter the prefix "0b" or "0B", followed by an optional space for better readability. Then group the binary bit pattern into blocks of four 0000 1001 1010 1110 with spaces in between as a check. Only the characters "0" or "1" are allowed: "0b 0000 1001 1010 1110".
- Decimal: Enter the value "2478" directly, without a prefix.

See also

Device-based dependency of system functions (Page 8284)

PageUp

Description

Executes the key function <PageUp> on the HMI device.

This system function can only be used for function keys and tasks with a time trigger.

Use in the function list

PageUp

Use in user-defined functions

PageUp

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

-

See also

Device-based dependency of system functions (Page 8284)

PageDown

Description

Executes the key function <Pagedown> on the HMI device.

This system function can only be used for function keys.

Use in the function list

PageDown

Use in user-defined functions

PageDown

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

-

See also

Device-based dependency of system functions (Page 8284)

SetDeviceMode

Description

Toggles the operating mode on the HMI device. The following types of operation are possible: "Online", "Offline" and "Load".

Use in the function list

SetDeviceMode (Operating mode)

Use in user-defined functions

SetDeviceMode (Operating_mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Operating mode

Determines the operating mode of the HMI device:

0 (hmiOnline) = Online: The connection to the PLC is established. The configured connection status is always set in this process. The states that were last used in Runtime are not considered.

1 (hmiOffline) = Offline: The connection to the PLC is disconnected.

2 (hmiTransfer) = load: A project can be transferred from the configuration computer to the HMI device.

Note

If you use a PC as an HMI device, the runtime software will be exited when you change operating mode after "Load".

See also

Device-based dependency of system functions (Page 8284)

SetConnectionMode (Page 8337)

SetBit

Description

Sets the value of a "Bool" type tag to 1 (TRUE).

Use in the function list

SetBit (Tag)

Use in user-defined functions

SetBit (Tag)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The BOOL type tag which is set to 1 (TRUE).

Example

The following program code sets the value of the boolean tag b_value to 1 by means of the SetBit function and outputs the result along with the original b_saved value.

```
{
BOOL b_value = 0;
BOOL b_saved = b_value;

//Set bit
  SetBit (b_value);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",b_value, b_saved);
...
}
```

See also

Device-based dependency of system functions (Page 8284)

SetBitInTag

Description

Sets a bit in the given tag to 1 (TRUE).

After changing the given bit, the system function transfers the entire tag back to the PLC. It is not checked whether other bits in the tags have changed in the meantime. Operator and PLC have read-only access to the indicated tag until it is transferred back to the PLC.

Note

If the PLC supports BOOL tags, do not use this system function. Use the "SetBit" system function instead.

Use in the function list

SetBitInTag (Tag, Bit)

Use in user-defined functions

SetBitInTag(Tag, Bit)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The tag in which a bit is set to 1 (TRUE).

Bit

The number of the bit that is set to 1 (TRUE).

When this system function is used in a user-defined function, the bits in the specified tag will be counted from right to left independent of the PLC used. The counting begins with 0.

Note

The guaranteed update of the tags used with actual process values is absolutely conditional in terms of reliable functionality. You should therefore configure the tag in an I/O field or assign the system function to a screen object, such as a button.

If you have configured a short event such as the activation of an alarm for the system function you can only access the actual process values by setting the tag for continuous reading.

Example

The following program code sets a bit to 1 at the specified bitposition in the bvalue tag and outputs the result along with the original bsaved value.

```
{
BYTE bvalue;
BYTE bsaved = bvalue;
BYTE bitposition = 2;

//Reset bit in bitposition
SetBitInTag (bvalue, bitposition);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bvalue, bsaved);
  ...
}
```

See also

Device-based dependency of system functions (Page 8284)

SetBit (Page 8329)

SetBitWhileKeyPressed

Description

Sets the bit of the given tag to 1 (TRUE) as long as the user keeps the configured key pressed.

After changing the given bit, the system function transfers the entire tag back to the PLC. It is not checked whether other bits in the tags have changed in the meantime. Operator and PLC have read-only access to the indicated tag until it is transferred back to the PLC. You should only access tags of the BOOL type with this system function to avoid problems with overlapping access to the same tag.

Note

All functions on the event "Release" are performed immediately by means of a screen change configured for a key, even if the key is kept pressed.

If the "SetBitWhileKeyPressed" system function is configured for a function key, the bit will be reset immediately following a screen change. This action is necessary since the key assignments change after the screen change.

If the PLC supports BOOL tags, do not use this system function. Use the "SetBit" system function instead.

Use in the function list

SetBitWhileKeyPressed (Tag, Bit)

Use in user-defined functions

-

Parameters

Tag

The tag in which a bit is temporarily set to 1 (TRUE). Use only tags of the type BOOL, as far as allowed by the PLC.

Bit

The number of the bit that is temporarily set to 1 (TRUE).

Note

The guaranteed update of the tags used with actual process values is absolutely conditional in terms of reliable functionality. You should therefore configure the tag in an IO field, or assign the function to a screen element such as a button.

If you configured a short event such as the activation of an alarm for the function you can only access the actual process values by setting the tag for continuous reading.

See also

SetBit (Page 8329)

Device-based dependency of system functions (Page 8284)

SetBacklightColor

Description

Defines the background lighting of the button.

Note

The configuration that was set at Switch off is reestablished when restarting the HMI device.

Use in the function list

SetBacklightColor (Value)

Use in user-defined functions

-

Parameters

Value

Defines the background lighting of the button:

0 (hmiWhite) = White: No color

1 (hmiGreen) = Green: Green color

2 (hmiYellow) = Yellow: Yellow color

3 (hmiRed) = Red: Red color

SetPLCDateTime

Description

Changes the data and the time of the linked PLC

The system function "SetPLCDateTime" can only be configured for the following PLCs:

- SIMATIC S7 1200
- SIMATIC S7 1500

Use in the function list

SetPLCDateTime (connection, time)

Use in user-defined functions

SetPLCDateTime (Connection, Time)

Parameters

Connection

Connection of PLC and HMI device.

Time

Transfers the date and the time of the HMI device to the PLC. The PLC applies the date and the time of the HMI device.

SetPLCMode

Description

Switches the operating mode of the PLC to one of the following states:

- RUN
- STOP

The system function "SetPLCMode" can only be configured for the following PLCs:

- SIMATIC S7 1200
- SIMATIC S7 1500

Use in the function list

SetPLCMode (connection, mode)

Use in user-defined functions

SetPLCMode (Connection, Mode)

Parameters

Connection

Connection of PLC and HMI device.

Mode

Specifies the operating mode of the PLC:

RUN = the PLC is switched to the RUN state. The PLC program is executed.

STOP = the PLC is switched to the STOP state. The PLC program is interrupted.

SetLanguage

Description

Toggles the language on the HMI device. All configured text and system events are displayed on the HMI device in the newly set language.

Use in the function list

SetLanguage (Language)

Use in user-defined functions

SetLanguage (Language)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Language

Determines which language is set on the HMI device. The following specifications are possible:

- -1 (hmiToggle) = Toggle: Changes to the next language. The sequence is determined during configuration in the "Project languages" editor.
- Number you have defined under "Languages and fonts" in the "Runtime Settings" editor. Changes to the language with the given number.

- Language you have defined under "Languages and fonts" in the "Runtime Settings" editor.
- Language abbreviation in accordance with the VBScript5 reference: This changes to the language corresponding to the specified language code, e.g. "de-DE" for German (Germany) or "en-US" for English (United States).
  An overview of the language abbreviations is available in the basic information of VBScript under the topic "Area diagram-ID (LCID) Diagram".

See also

Device-based dependency of system functions (Page 8284)

SetTag

Description

Assigns a new value to the given tag.

Note

This system function can be used to assign strings and numbers, depending on the type of tag.

Use in the function list

SetTag (Tag, Value)

Use in user-defined functions

SetTag (Tag, Value)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Tag

The tag to which the given value is assigned.

Value

The value which the given tag is assigned.

Note

The "SetTag" system function is only executed after a connection has been established.

Example

The following program code uses the SetTag function to set the value of the gs_tag_bit tag to TRUE and saves the return value to the ok tag.

```
{
BOOL ok;
BOOL bvalue;

//Set the tag to true
ok = SetTag("gs_tag_bit", TRUE);
//error handling
if(ok)
{
  // succeeded
  printf ( "Function has run through.\r\n" );
bvalue = GetTagBit("gs_tag_bit");
  printf ("Value of gs_tag_bit: %d\r\n", bvalue);
}
else
{
  // failed
  printf ( "Error - function failed." );
}
...
}
```

The saved return value can be processed in the following code.

See also

Device-based dependency of system functions (Page 8284)

IncreaseTag (Page 8297)

SetConnectionMode

Description

Connects or disconnects the given connection.

Note

A connection to the PLC cannot be established until the operating mode ONLINE has been set on the HMI device. Use the "SetDeviceMode" system function for this purpose.

Use in the function list

SetConnectionMode (Mode, Connection)

Use in user-defined functions

SetConnectionMode (Mode, Connection)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Mode

Determines whether a connection to the PLC is established or disconnected:

0 (hmiOnline) = Online: Connection is established.

1 (hmiOffline) = Offline: Connection is disconnected.

Connection

The PLC to which the HMI device is connected. You specify the name of the PLC in the connection editor.

Multiple use of the system function in a user-defined function

If you use the "SetConnectionMode" system function for different connections, it may be possible that not all system functions are executed correctly. Proceed as follows to prevent this situation:

1. Create a "BOOL" type tag with the start value "0".
2. Configure the "SetConnectionMode" system function on the "Value change" event of the HMI tags. If you want to disconnect three connections, for example, you must configure the system function three times.
3. In the user-defined function, apply the "InvertBit" system function to the HMI tag.

Application example

Two typical application examples for this system function are as follows:

- Test
  As long as no PLC is connected to the HMI device, no error messages will be output during the test on the HMI device. If the HMI device is connected to a PLC, the connection to the PLC can be established by pressing a key.
- Commissioning
  Several PLCs are to be configured for a system. At first, all PLCs except one are configured "Offline". After commissioning of the first PLC, the connection to each of the other PLCs is established by pressing a key. In this way, the other PLCs are started up one after another.

See also

Device-based dependency of system functions (Page 8284)

SetDeviceMode (Page 8328)

SimulateSystemKey

Description

Simulates the behavior of a System Key. Use this system function if a system key, such as the "ACK" key, "Input" key or the number pad is not available on the HMI device.

Use in the function list

SimulateSystemKey (System key)

Use in user-defined functions

-

Parameters

System key

System Key, the behavior for which is to be simulated.

System key "+/-"

With the SimulateSystemKey system function, the system key "+/-" is only supported for the following HMI devices:

- KP300 Basic
- KP400 Basic
- KTP400 Basic mono PN
- KTP400 Basic color PN
- KTP400 Basic color PN Portrait
- KTP600 Basic mono PN
- KTP600 Basic color PN
- KTP600 Basic color DP
- KTP1000 Basic PN
- KTP1000 Basic DP

Use the system keys "+" and "-" separately for all other HMI devices.

SimulateTag

Description

Simulates the behavior of tags and dynamic objects such as text lists, without having the HMI device connected to a PLC. You can, for example, configure the system function to the "Loaded" event of a screen.

This system function is used, for example, to demonstrate the functionality of a project.

Only tags of the data type Integer can be used for simulation. Tags of the data types Integer and Double Integer, however, can be used with OP 73, OP 77A, TP 177A.

Note

If you use the system function "SimulateTag" with a short cycle time on a Basic Panel, the HMI device may be overloaded.

Use in the function list

SimulateTag (Tag, Cycle, Maximum value, Minimum value, Value)

Use in user-defined functions

-

Parameter

Tag

The tag whose value is changed.

Cycle

The factor by which the basic cycle of 200 milliseconds is multiplied. The cycle defines when the tag value is changed by the specified value. Possible cycles between 1 and 32767.

Maximum value

The maximum value that the tag can assume during simulation. The maximum value must be greater than the minimum value but less than / equal to 32767.

Minimum value

The minimum value that the tag can assume during simulation. The minimum value must be lower than the maximum value but greater than / equal to -32768.

Value

The value by which the tag value is changed during each cycle. Possible values between -32768 and 32767.

- A positive value increases the tag value. When the maximum value is reached, the tag value is set to the minimum value after the next update cycle.
- A negative value reduces the tag value. When the minimum value is reached, the tag value is set to the maximum value after the next update cycle.

StopRuntime

Description

Exits the runtime software and thereby the project running on the HMI device.

Use in the function list

StopRuntime (Mode)

Use in user-defined functions

StopRuntime (Mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameters

Mode

Determines whether the operating system is shut down after exiting runtime.

0 (hmiStopRuntime) = Runtime: Operating system is not shut down

1 (hmiStopRuntimeAndOperatingSystem) = Runtime and operating system: The operating system is shut down (not possible with WinCE)

Example

The following program code shuts down Runtime and the operating system.

```
{

//Stop runtime and shutdown
StopRuntime (hmiStopRuntimeAndOperationSystem);

}
```

The saved return value can be processed in the following code.

See also

Device-based dependency of system functions (Page 8284)

TraceUserChange

Description

Outputs a system event that shows which user is currently logged in on the HMI device.

This system function can only be used in the Scheduler.

Use in the function list

TraceUserChange

Use in user-defined functions

-

Parameters

--

DecreaseFocusedValue

Description

Subtracts the specified value from the value of the tag which is connected to the screen object and currently has the focus.

This system function can only be used for function keys.

Use in the function list

DecreaseFocusedValue (Value)

Use in user-defined functions

-

Parameters

Value

The value which is subtracted from the tag value.

DecreaseTag

Description

Subtracts the given value from the tag value.

X = X - a

Note

The system function uses the same tag as input and output values. When this system function is used to convert a value, auxiliary tags must be used. The auxiliary tags are assigned to the tag value with the "SetTag" system function.

If you configure the system function on the events of an alarm and the tag is not being used in the current screen, it is not ensured that the actual value of the tags is being used in the PLC. You can improve the situation by setting the "Cyclic continuous" acquisition mode.

Use in the function list

DecreaseTag (Tag, Value)

Use in user-defined functions

DecreaseTag (Tag, Value)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Tag

The tag from which the given value is subtracted.

Value

The value which is subtracted.

See also

Device-based dependency of system functions (Page 8284)

ChangeConnection

Description

Disconnects the connection to the currently used PLC in use and establishes a connection to a PLC with another address. The newly connected PLC must belong to the same device class

(S7-1200, S7-300, ...etc). With S7-1200, the use of the function is also only permitted for absolute addressing.

Note

When changing to another address, ensure that the address is not already being used by another HMI device.

The follows address types are supported:

- IP address
- MPI address

The follows PLC types are supported:

- SIMATIC S7 300/400
- SIMATIC S7-NC
- SIMOTION

Use in the function list

ChangeConnection (Connection, Address, Slot, Rack)

Use in user-defined functions

ChangeConnection (Connection, Address, Slot, Rack)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Connection

Name of the connection that is disconnected. The name is set during configuration, for example, in the "Connections" editor.

Address

MPI/PROFIBUS or IP address of the PLC to which the connection will be established.

Note

Set the address by means of a tag. The objects list displays the tags of all data types. Select only tags of the following data types:

- Ethernet connection: "String" data type
- MPI connection: "Int" data type

Slot

Slot of the PLC to which the connection will be established.

Rack

Rack of the PLC to which the connection will be established.

Application example

You want to operate one HMI device on several machines. Configure the necessary PLCs in the project, to which you want to change by pressing a key. When changing the PLC, the connection to the PLC in use is disconnected. Then the connection to the new PLC with other address parameters is reestablished. To access the values of the new PLC, configure the same tags for the PLC used.

The PLC which you have indicated when creating the project will be used as default.

1. Enter the name and address of the PLC in the "Connections" editor.
2. Configure a button in the process screen.
3. Configure the "ChangeConnection" system function on the "Press" event.
4. Enter the name of the connection and address of the PLC as parameters.

See also

Device-based dependency of system functions (Page 8284)

ShowLogonDialog

Description

Opens a dialog on the HMI device with which the user can log on to the HMI device.

Use in the function list

ShowLogonDialog

Use in user-defined functions

-

Parameters

--

ShowOperatorNotes

Application

Displays the tooltip configured for the selected object.

If the system function is configured on a function key, the tooltip for the screen object that currently has the focus is displayed. If a tooltip is configured for the screen itself, you can switch to this text by pressing <Enter> or by double-clicking on the help window.

If the system function is configured on a button, only the tooltip for the current screen is displayed. If a tooltip is configured on the button itself, initially only the tooltip for the button is displayed. You can press <Enter> or double-click on the help window to switch to the tooltip for the current screen.

Note

No other screen object can be used while the help window is open. To use the screen object, close the help window.

Closing the help window

You can close the help window in the following ways:

Using the keys:

- By pressing the <HELP> key again
- By pressing the <ESC> key

Using the touch screen:

- By pressing the ❌ button

Use in the function list

ShowOperatorNotes (Layout)

Use in user-defined functions

ShowOperatorNotes (Display_mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Display mode

Determines whether the configured tooltip is hidden or shown:

0 (hmiOff) = Off: Configured tooltip is hidden

1 (hmiOn) = On: Configured tooltip is shown

-1 (hmiToggle) = Toggle: Toggles between the two modes

See also

Device-based dependency of system functions (Page 8284)

ShowAlarmWindow

Description

Hides or shows the alarm window on the HMI device.

Use in the function list

ShowAlarmWindow (Object name, Layout)

Use in user-defined functions

ShowAlarmWindow (Object_name, Display_mode)

Can be used if the configured device supports user-defined functions. For additional information, refer to "Device dependency".

Parameter

Object name

Name of the alarm view which is hidden or shown.

Layout

Determines whether the alarm window is hidden or shown:

0 (hmiOff) = Off: Alarm view is hidden

1 (hmiOn) = On: Alarm view is shown

-1 (hmiToggle) = Toggle: Toggles between the two modes

See also

Device-based dependency of system functions (Page 8284)

10.6.4.2 Events

Overview

Editors

Introduction

The following table shows which events occur in which editor.

Technical data subject to change.

Icon	Editor
	Screens
	HMI alarms
	HMI tags
	Scheduler

1	Cleared (Page 8353)	X	--	--	--
2	Activate (Page 8353)	--	--	--	--
3	Change (Page 8354)	--	--	--	--
4	Loaded (Page 8354)	X	--	--	--
5	Execute (Page 8354)	--	--	--	X
6	Selection changed (Page 8355)	--	--	--	--
7	In the event of high limit violation (Page 8355)	--	--	X	--
8	In the event of low limit violation (Page 8355)	--	--	X	--
9	When dialog is opened (Page 8355)	--	--	--	X
10	When dialog is closed (Page 8356)	--	--	--	X
11	User change (Page 8356)	--	--	--	X
12	Screen change (Page 8356)	--	--	--	X
13	Deactivate (Page 8356)	--	--	--	X
14	Double-click (Page 8357)	--	--	--	--
15	Press (Page 8357)	--	--	--	--
16	On Finish Input (Page 8358)	--	--	--	--
17	Press ESC twice (Page 8358)	--	--	--	--
18	Outgoing (Page 8358)	--	X	--	--
19	Incoming (Page 8359)	--	X	--	--
20	Click (Page 8359)	--	--	--	--
21	Click when flashing (Page 8359)	--	--	--	--
22	Loop-in alarm (Page 8360)	--	X	--	--
23	Release (Page 8360)	--	--	--	--
24	Alarm buffer overflow (Page 8360)	--	--	--	X

25	Acknowledge (Page 8361)	--	X	--	--
26	Reach margin (Page 8361)	--	--	--	--
27	Runtime Stop (Page 8361)	--	--	--	X
28	Press key (Page 8362)	--	--	--	--
29	Release key (Page 8362)	--	--	--	--
30	Overflow (Page 8362)	--	--	--	--
31	Switch OFF (Page 8363)	--	--	--	--
32	Switch ON (Page 8363)	--	--	--	--
33	Low free storage space (Page 8363)	--	--	--	--
34	Free space critically low (Page 8363)	--	--	--	--
35	Value change (Page 8363)	--	--	X	--
36	Time expired (Page 8364)	--	--	--	--

Basic objects

Introduction

The following table shows which events occur on which objects.

Technical data subject to change.

Icon	Object
	Line
	Ellipse
	Circle
	Rectangle
A	Text field
	Graphic view

1	Cleared (Page 8353)	--	--	--	--	--	--
2	Activate (Page 8353)	--	--	--	--	--	--
3	Change (Page 8354)	--	--	--	--	--	--
4	Loaded (Page 8354)	--	--	--	--	--	--
5	Execute (Page 8354)	--	--	--	--	--	--
6	Selection changed (Page 8355)	--	--	--	--	--	--
7	In the event of high limit violation (Page 8355)	--	--	--	--	--	--

						A	
8	In the event of low limit violation (Page 8355)	--	--	--	--	--	--
9	When dialog is opened (Page 8355)	--	--	--	--	--	--
10	When dialog is closed (Page 8356)	--	--	--	--	--	--
11	User change (Page 8356)	--	--	--	--	--	--
12	Screen change (Page 8356)	--	--	--	--	--	--
13	Deactivate (Page 8356)	--	--	--	--	--	--
14	Double-click (Page 8357)	--	--	--	--	--	--
15	Press (Page 8357)	--	--	--	--	--	--
16	On Finish Input (Page 8358)	--	--	--	--	--	--
17	Press ESC twice (Page 8358)	--	--	--	--	--	--
18	Outgoing (Page 8358)	--	--	--	--	--	--
19	Incoming (Page 8359)	--	--	--	--	--	--
20	Click (Page 8359)	--	--	--	--	--	--
21	Click when flashing (Page 8359)	--	--	--	--	--	--
22	Loop-in alarm (Page 8360)	--	--	--	--	--	--
23	Release (Page 8360)	--	--	--	--	--	--
24	Alarm buffer overflow (Page 8360)	--	--	--	--	--	--
25	Acknowledge (Page 8361)	--	--	--	--	--	--
26	Reach margin (Page 8361)	--	--	--	--	--	--
27	Runtime Stop (Page 8361)	--	--	--	--	--	--
28	Press key (Page 8362)	--	--	--	--	--	--
29	Release key (Page 8362)	--	--	--	--	--	--
30	Overflow (Page 8362)	--	--	--	--	--	--
31	Switch OFF (Page 8363)	--	--	--	--	--	--
32	Switch ON (Page 8363)	--	--	--	--	--	--
33	Low free storage space (Page 8363)	--	--	--	--	--	--
34	Free space critically low (Page 8363)	--	--	--	--	--	--
35	Value change (Page 8363)	--	--	--	--	--	--
36	Time expired (Page 8364)	--	--	--	--	--	--

Elements

Introduction

The following table shows which events occur on which objects.

Technical data subject to change.

Icon	Object
	IO field
	Button
	Symbolic IO field
	Graphic IO field

Icon	Object
	Date/time field
	Bar
	Switch

1	Cleared (Page 8353)	--	--	--	--	--	--	--
2	Activate (Page 8353)	X	X	X	X	--	--	X
3	Change (Page 8354)	--	X	X	--	--	--	X
4	Loaded (Page 8354)	--	--	--	--	--	--	--
5	Execute (Page 8354)	--	--	--	--	--	--	--
6	Selection changed (Page 8355)	--	--	--	--	--	--	--
7	In the event of high limit violation (Page 8355)	--	--	--	--	--	--	--
8	In the event of low limit violation (Page 8355)	--	--	--	--	--	--	--
9	When dialog is opened (Page 8355)	--	--	--	--	--	--	--
10	When dialog is closed (Page 8356)	--	--	--	--	--	--	--
11	User change (Page 8356)	--	--	--	--	--	--	--
12	Screen change (Page 8356)	--	--	--	--	--	--	--
13	Deactivate (Page 8356)	X	X	X	X	--	--	X
14	Double-click (Page 8357)	--	--	--	--	--	--	--
15	Press (Page 8357)	--	X	--	--	--	--	--
16	On Finish Input (Page 8358)	--	--	--	--	--	--	--
17	Press ESC twice (Page 8358)	--	--	--	--	--	--	--
18	Outgoing (Page 8358)	--	--	--	--	--	--	--
19	Incoming (Page 8359)	--	--	--	--	--	--	--
20	Click (Page 8359)	--	X	--	--	--	--	--
21	Click when flashing (Page 8359)	--	--	--	--	--	--	--
22	Loop-in alarm (Page 8360)	--	--	--	--	--	--	--
23	Release (Page 8360)	--	X	--	--	--	--	--
24	Alarm buffer overflow (Page 8360)	--	--	--	--	--	--	--
25	Acknowledge (Page 8361)	--	--	--	--	--	--	--
26	Reach margin (Page 8361)	--	--	--	--	--	--	--
27	Runtime Stop (Page 8361)	--	--	--	--	--	--	--
28	Press key (Page 8362)	--	--	--	--	--	--	--
29	Release key (Page 8362)	--	--	--	--	--	--	--
30	Overflow (Page 8362)	--	--	--	--	--	--	--
31	Switch OFF (Page 8363)	--	--	--	--	--	--	X
32	Switch ON (Page 8363)	--	--	--	--	--	--	X
33	Low free storage space (Page 8363)	--	--	--	--	--	--	--
34	Free space critically low (Page 8363)	--	--	--	--	--	--	--
35	Value change (Page 8363)	--	--	--	--	--	--	--
36	Time expired (Page 8364)	--	--	--	--	--	--	--

Controls

Introduction

The following table shows which events occur on which objects.

Technical data subject to change.

Icon	Object
	Alarm view/alarm window
	Alarm indicator
	Trend view
	User view
	Recipe view
	Auxiliary indicator

1	Cleared (Page 8353)	--	--	--	--	--	--
2	Activate (Page 8353)	X	--	X	X	--	--
3	Change (Page 8354)	--	--	--	--	--	--
4	Loaded (Page 8354)	--	--	--	--	--	--
5	Execute (Page 8354)	--	--	--	--	--	--
6	Selection changed (Page 8355)	--	--	--	--	--	--
7	In the event of high limit violation (Page 8355)	--	--	--	--	--	--
8	In the event of low limit violation (Page 8355)	--	--	--	--	--	--
9	When dialog is opened (Page 8355)	--	--	--	--	--	--
10	When dialog is closed (Page 8356)	--	--	--	--	--	--
11	User change (Page 8356)	--	--	--	--	--	--
12	Screen change (Page 8356)	--	--	--	--	--	--
13	Deactivate (Page 8356)	X	--	X	X	--	--
14	Double-click (Page 8357)	--	--	--	--	--	--
15	Press (Page 8357)	--	--	--	--	--	--
16	On Finish Input (Page 8358)	--	--	--	--	--	--
17	Press ESC twice (Page 8358)	--	--	--	--	--	--
18	Outgoing (Page 8358)	--	--	--	--	--	--
19	Incoming (Page 8359)						
20	Click (Page 8359)	--	X	--	--	--	--
21	Click when flashing (Page 8359)	--	X	--	--	--	--
22	Loop-in alarm (Page 8360)	--	--	--	--	--	--
23	Release (Page 8360)	--	--	--	--	--	--
24	Alarm buffer overflow (Page 8360)	--	--	--	--	--	--
25	Acknowledge (Page 8361)	--	--	--	--	--	--
26	Reach margin (Page 8361)	--	--	--	--	--	--

27	Runtime Stop (Page 8361)	--	--	--	--	--	--
28	Press key (Page 8362)	--	--	--	--	--	--
29	Release key (Page 8362)	--	--	--	--	--	--
30	Overflow (Page 8362)	--	--	--	--	--	--
31	Switch OFF (Page 8363)	--	--	--	--	--	--
32	Switch ON (Page 8363)	--	--	--	--	--	--
33	Low free storage space (Page 8363)	--	--	--	--	--	--
34	Free space critically low (Page 8363)	--	--	--	--	--	--
35	Value change (Page 8363)	--	--	--	--	--	--
36	Time expired (Page 8364)	--	--	--	--	--	--

Events

Cleared

Description

Occurs when the active screen on the HMI device is cleared.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Activate

Description

Occurs when the user selects a display or operating object using the configured tab sequence.

Note

Note that the availability of the event depends on the HMI device and object type.

Note

If the user e.g. clicks a button with the mouse, the "Click" event is triggered. Users wishing to trigger the "Enable" event must select the button using the tab key.

The "Enable" event is only used to detect whether an object was selected. The event does not trigger a password prompt.

For this reason, do not use the "Enable" event if you want to configure access protection on the function call of the object.

Change

Description

Occurs if the status of a display and operator control object changes.

The status of an object changes if, for example, the user presses the key.

Note

Note that the availability of the event depends on the HMI device and object type.

Loaded

Description

Occurs when all configured display and operating objects are loaded in the active screen after a screen change.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Note

Enable a screen change to ensure that the connection with the control is established after switch-on.

Execute

Description

Occurs when the scheduled task has been executed.

Selection changed

Description

Occurs when the user changes the selection.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

In the event of high limit violation

Description

Occurs when the high limit of a tag is exceeded.

Note

Note that the availability of the event depends on the HMI device and object type.

In the event of low limit violation

Description

Occurs when the low limit of a tag is undershot.

Note

Note that the availability of the event depends on the HMI device and object type.

When dialog is opened

Description

The event is triggered when a modal dialog opens.

Note

Please note that the availability of the event depends on the HMI device and object type.

When dialog is closed

Description

The event is triggered when a modal dialog closes.

Note

Please note that the availability of the event depends on the HMI device and object type.

User change

Description

Occurs when a user logs off at an HMI device or another user logs on at the HMI device.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Screen change

Description

Occurs when all configured display and operating objects are loaded in the screen after a screen change.

Use the "Loaded" event if you want to perform other system functions during a screen change to a certain screen.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Deactivate

Description

Occurs when the user takes the focus from a display and operating object.

A screen object can be disabled using the configured tab order or by performing another action with the mouse.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Note

System functions or user-defined functions on the "Deactivate" event of a screen are not executed when a screen is being closed.

The "Deactivate" event is only used to detect whether an object was deselected. The event does not trigger a password prompt.

For this reason, do not use the "Deactivate" event if you want to configure access protection on the function call of the object.

Double-click

Description

Occurs when the user double-clicks on an object from the symbol library.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Press

Description

Occurs when the user clicks on a button with the left mouse button, presses <RETURN> or <SPACE>.

Also occurs when the user right-clicks on an object of the symbol library.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

On Finish Input

Description

Triggered when you confirm input at an I/O field by pressing ENTER, or by mouse click, or by touch screen operation.

The "On Finish Input" event is also started if the value of a tag does not change, for example, if a value is exceeded, or if a user cancels the dialog to acknowledge a tag (Audit option package) that has to be acknowledged.

The event is not triggered, on the other hand, by user logon or by input fields configured with an authorization.

Note

Note that the availability of the event depends on the HMI device and object type.

Press ESC twice

Description

Occurs when the user presses the <ESC> key twice at the HMI device.

Note

Note that the availability of the event depends on the HMI device and object type.

Outgoing

Description

Occurs when an alarm is deactivated.

Note

Please note that the availability of the event depends on the HMI device and object type.

Incoming

Description

Occurs when an alarm is triggered and displayed in the alarm view.

Note

Please note that the availability of the event depends on the HMI device and object type.

Click

Description

Occurs if the user clicks a display and operating object with the mouse or touches the touch display with a finger.

In case you click the incorrect object, prevent processing of configured function list as follows:

- Move the mouse pointer away from the object while keeping the mouse button pressed. Release the mouse button as soon as the mouse pointer leaves the object. The function list will then not be processed.
- On touch displays, the display must be touched with the finger until a reaction occurs, e.g., a screen change.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Click when flashing

Description

Occurs when the user clicks a flashing alarm indicator with the mouse or touches it with a finger.

Note

Note that the availability of the event depends on the HMI device and object type.

Loop-in alarm

Description

Occurs as soon as the user selects an alarm in the alarm view and then clicks on the "Loop-In-Alarm" button or double clicks on the alarm.

For the "Loop-In-Alarm" event, you configure system functions, such as a change to the screen in which the alarm occurred.

You cannot configure local scripts for the "Loop-In-Alarm" event in Runtime Professional.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Release

Description

Occurs when the user releases a button.

This even does not occur, as long as the button remains pressed.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Alarm buffer overflow

Description

Occurs when the configured size of the alarm buffer is reached.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Acknowledge

Description

Occurs when the user acknowledges an alarm.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Reach margin

Description

Occurs when the user reaches the beginning or the end of a scrollable area.

Note

Note that the availability of the event depends on the HMI device and object type.

Note

A user-defined function must not be configured for the "Boundary reached" event.

Configurable objects

The event can only be configured on the <Up> and <Down> keys, or on the keys on which you have configured the "ScreenObjectPageUp" or "ScreenObjectPageDown" system functions.

Runtime Stop

Description

Occurs when the user exits the Runtime software on the HMI device.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Note

A user-defined function must not be configured for the "Runtime stop" event.

Press key

Description

Occurs when the user presses a function key.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Release key

Description

Occurs when the user releases a function key.

Note

Note that the availability of the event depends on the HMI device and object type.

Overflow

Description

Occurs when the configured size of the log is reached. You use the log type "Trigger event".

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

Switch OFF

Description

Occurs when the user moves the display and operating object "Switch" to the OFF position.

Note

Note that the availability of the event depends on the HMI device and object type.

Switch ON

Description

Occurs when the user moves the display and operating object "Switch" to the ON position.

Note

Note that the availability of the event depends on the HMI device and object type.

Low free storage space

Description

This event is triggered if the storage space available on the medium to which the Audit Trail is less than the configured minimum.

Free space critically low

Description

This event is triggered if the storage medium to which an Audit Trail is saved provides insufficient storage space due to hardware restrictions.

Value change

Description

Occurs when the value of a tag or the value of an array element changes.

The value change of a tag is triggered by the PLC or by the user, e.g. when a new value is entered. No event is output if a system function changes the value.

Note

Please note that the availability of the event depends on the HMI device and object type.

Time expired

Description

Occurs when the time configured in the scheduler expires.

Note

Please note that the availability of the event is dependent upon the HMI device and object type.

10.7 Planning tasks

10.7.1 Field of application of the Scheduler

Definition

You can use the Scheduler to configure tasks to run independent of the screen in the background. You create tasks by linking system functions or scripts to a trigger. The linked functions will be called when the triggering event occurs.

Example of an application

The Scheduler is used to execute event-controlled tasks automatically. For example, you use a task to automate the following:

- Regular swap out of log data
- Printout of an alarm report when an alarm buffer overflow occurs
- Printout of a report at shift end
- Monitoring a tag
- Monitoring of a user change

Note

The availability of the listed examples is determined by the HMI device.

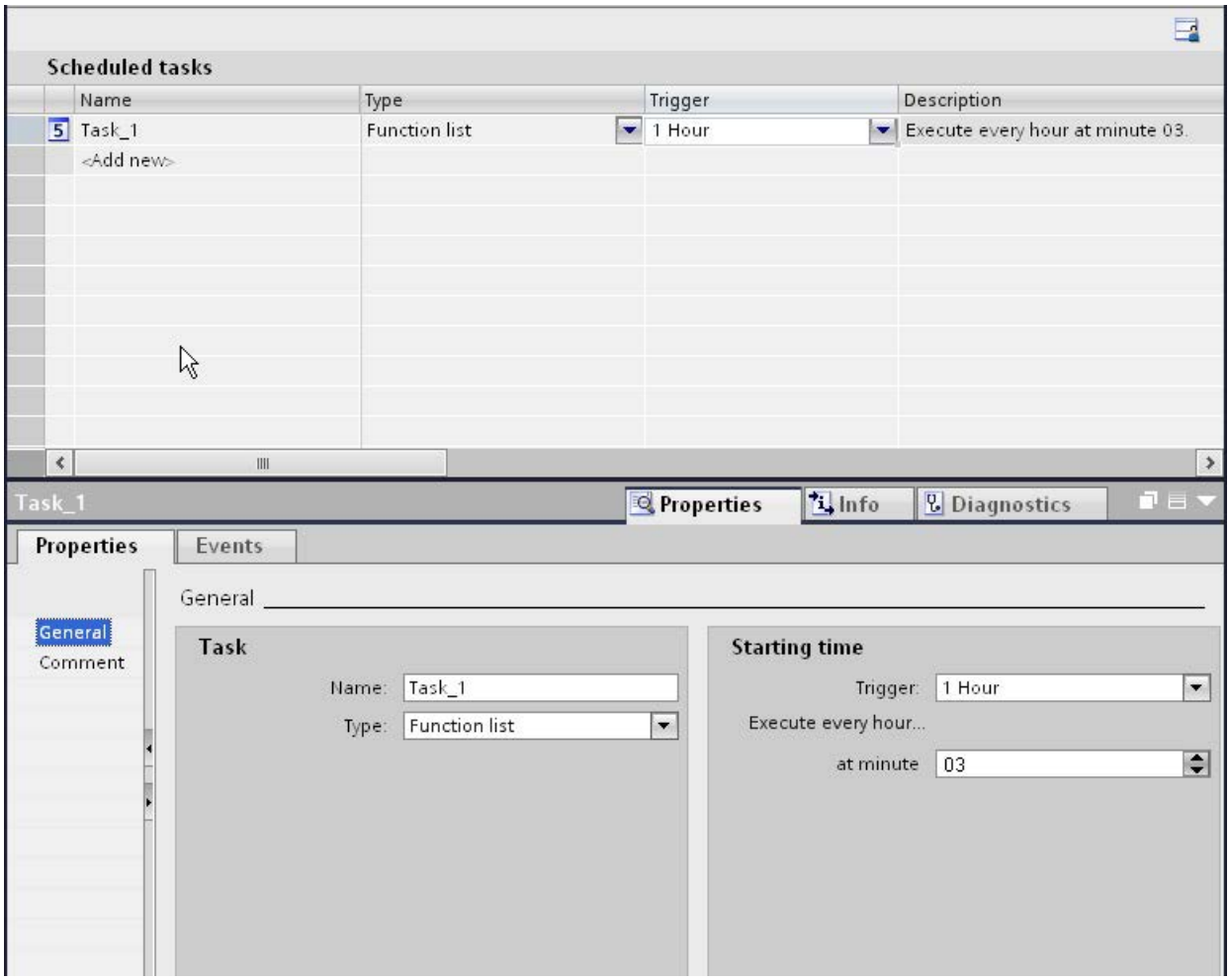
See also

Working with tasks and triggers (Page 8366)
Example: Update user following change of user (Page 8371)
Work area of the "Scheduler" editor (Page 8367)

10.7.2 Working with tasks and triggers

Introduction

A task consists of a trigger and a task type.



Starting a task

Controlled by a trigger, the Scheduler starts the task linked to the trigger.

See also

Field of application of the Scheduler (Page 8364)

10.7.3 Basics

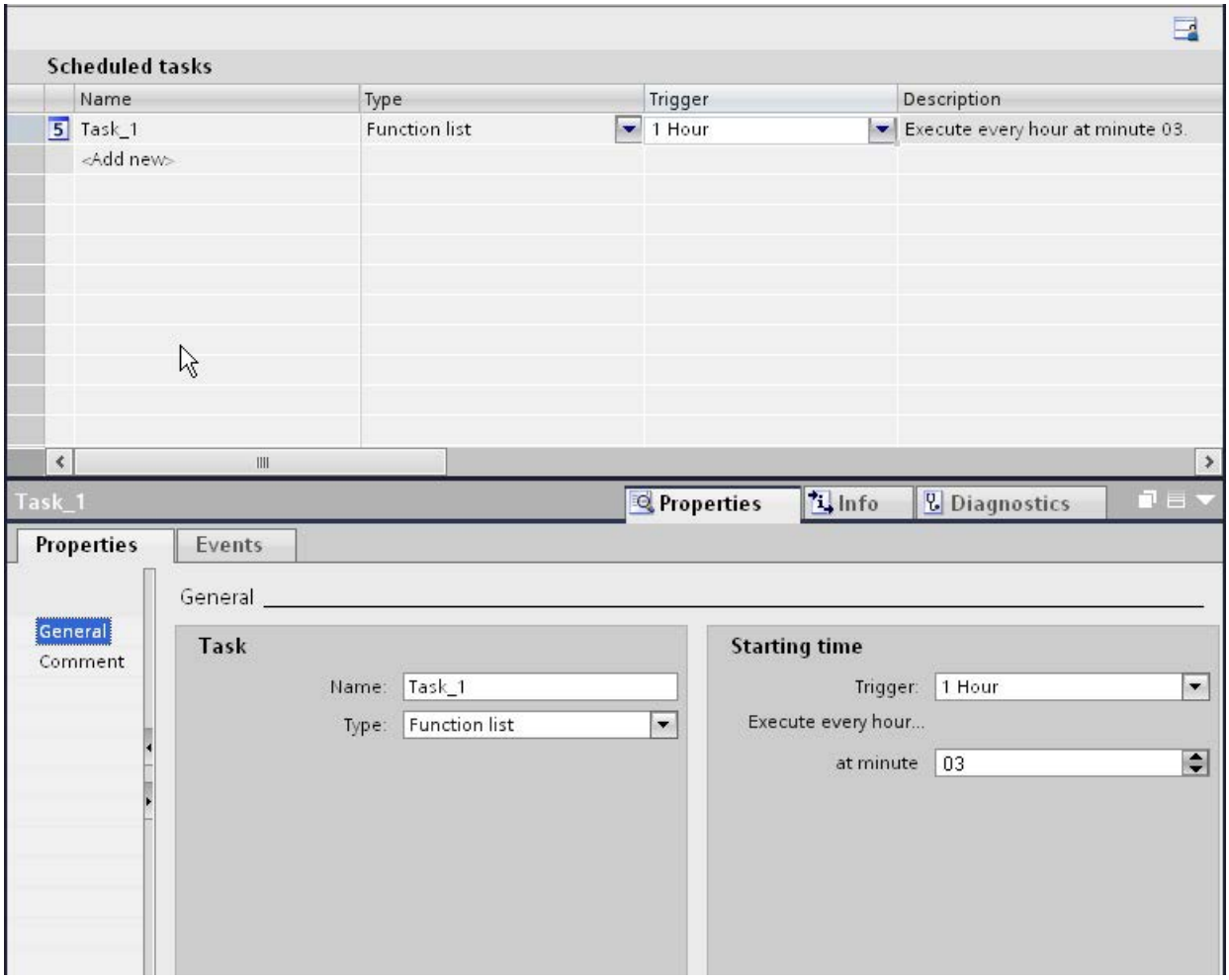
10.7.3.1 Work area of the "Scheduler" editor

Introduction

Double-click on "Scheduler" to open it in the project view. The work area shows the scheduled tasks, which consist of the trigger and the task type, for example, the function list.

Structure

The work area consists of the table of jobs.



The table of tasks shows specified tasks with their properties, such as triggers. You select a task type and a trigger. You assign a name and a comment to the task. The description provides a written summary of the task including the timing for the task.

Inspector window

The "Properties" tab of the Inspector window is split into two parts.

The "Job" area lists the name of the job and the job type. The "Starting time" area shows the trigger. The area is different depending on the trigger selected.

In the "Events" tab use the function list with system functions that will be executed in the task.

Note

You can obtain more detailed information about the elements of the user interface using the tooltips. To do so, move the mouse pointer to the relevant object or press <F1> if the object has already been selected.

See also

Field of application of the Scheduler (Page 8364)

Planning tasks with event triggers (Page 8369)

Triggers (Page 8368)

Function list (Page 8368)

10.7.3.2 Function list

Function list

A trigger starts the function list. The function list is executed line by line. Each line contains a system function. You can configure exactly one function list for each task.

Note

The choice of configurable system functions in a function list depends on the selected trigger and the HMI device.

See also

Work area of the "Scheduler" editor (Page 8367)

10.7.3.3 Triggers

Introduction

A trigger is linked to a task and forms the triggering event which will call this task. The task is executed when the trigger occurs.

Event trigger

When a task is linked to a system event, the task will be triggered by the event. System events include, for example, Runtime stop, screen change, user change, etc.

Each system event can only be configured once for each HMI device.

Deactivating job

If you do not need a certain job temporarily, deactivate the job in the Engineering System. You also use the trigger "Deactivated" to make a previously configured system event available once again.

Example: Task "A" is planned with the system event "Shutdown". This system event is then no longer available for another task "B". Select "Disabled" as the trigger for task "A" to make the "Runtime stop" system event available again.

Note

The available triggers depend on the HMI device.

See also

Work area of the "Scheduler" editor (Page 8367)

10.7.3.4 Planning tasks with event triggers

Introduction

You plan a task that generates a screen change when the user changes.

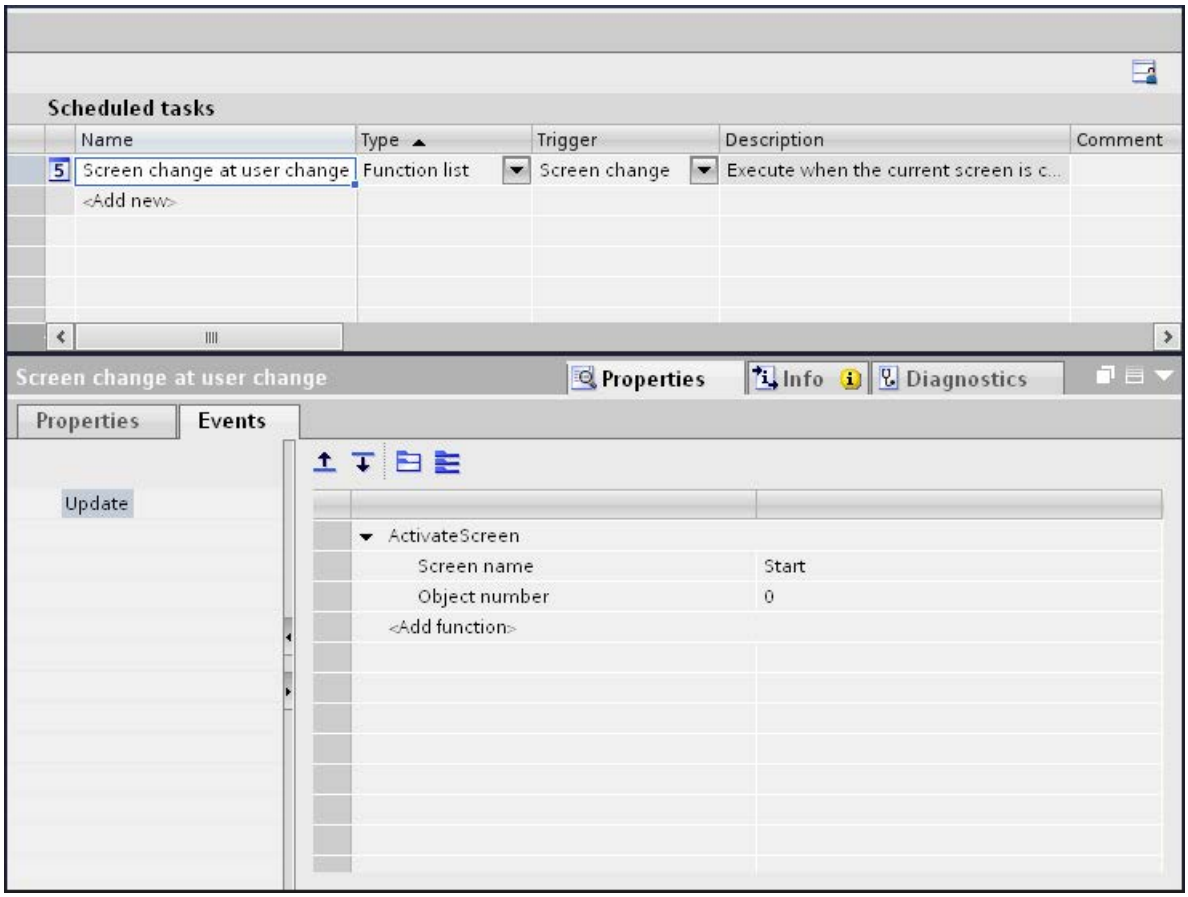
Requirements

- The "Scheduler" work area is open.
- You have created the "Start" screen.

Procedure

1. Click "Add..." in the table of the task area.
2. Enter "Screen change at user change" as the "Name."
3. Select "User change" as the "Trigger."
4. In the Inspector window, select "Properties > Events".

5. Select the "Screen/ActivateScreen" system function in the function list.
6. Select the "Start" screen in the screen name field.



Result

The task is executed with the "User change" event. When a new user logs on successfully, the "Start" screen is called up.

See also

Work area of the "Scheduler" editor (Page 8367)

10.7.4 Examples

10.7.4.1 Example: Update user following change of user

Task

Configure an I/O field which displays the logged on user. Configure a task which updates the I/O field when the logged on user changes.

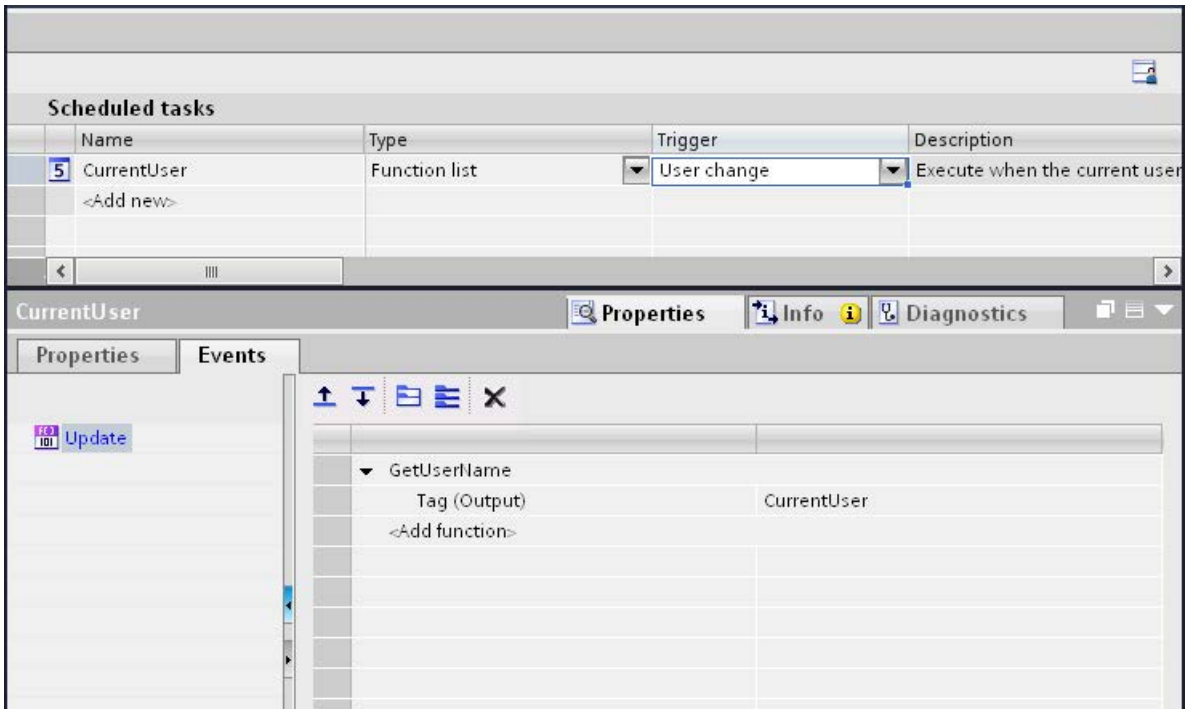
Requirements

- A "CurrentUser" tag of the "String" type is created.
- A screen has been created and opened.
- An I/O field is created in the screen.

Procedure

1. Click on the "I/O field" object.
2. In the Inspector window, select "Properties > Properties > General":
   – Select "String" as the "Display format."
   – Select "CurrentUser" as the "Variable."
   – Select "Output" as the mode.
3. Change to the work area of the Scheduler.
4. Click "Add..." in the table of the task area.
5. Enter "CurrentUser" as the "Name".
6. Select "User change" as the "Trigger."
7. In the Inspector window, select "Properties > Events".

8. Select the system function "ReadUserName" from the "User Management" group in the function list.
9. Select "CurrentUser" as the "Variable."



Result

When a new user logs on successfully, the "ReadUserName" function is called up. The "CurrentUser" tag is updated and displayed in the I/O field of the newly logged on user.

If a user does not log on successfully, the logged on user is logged off. The I/O field continues to display the user previously logged on until a new user logs on successfully.

See also

Field of application of the Scheduler (Page 8364)

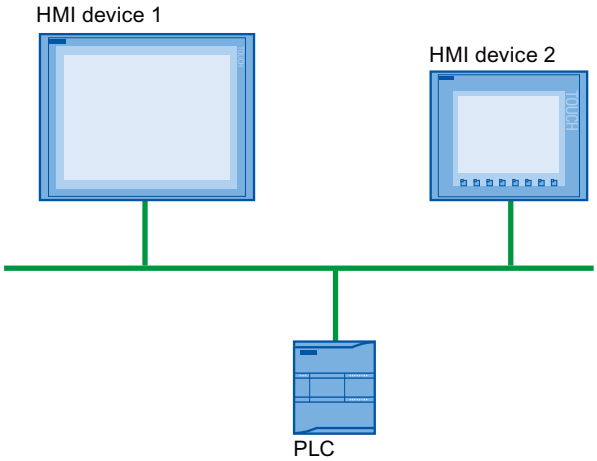
10.8 Communicating with PLCs

10.8.1 Basics of communication

10.8.1.1 Communication between devices

Communication

Data exchange between devices is known as communication. The devices can be interconnected directly or via a network. The interconnected devices in communication are referred to as communication partners.



Data transferred between the communication partners may serve different purposes:

- Display processes
- Operate processes
- Output alarms
- Administer process parameters and machine parameters

Communication partners

Communication between the following devices is described in more detail in this section:

- PLC
  The PLC controls a process by means of a user program.
- HMI device
  You use the HMI device to operate and monitor the process.

Basic information for all communication

The basis for all types of communication is a network configuration. In a network configuration, you specify the connection that exists between the configured devices.

With the network configuration, you also ensure the necessary prerequisites for communication, in other words:

- Every device in a network is assigned a unique address.
- The devices carry out communication with consistent transmission characteristics.

Automation system

The following characteristics describe an automation system:

- The PLC and HMI device are interconnected
- The network between the PLC and HMI device is configured

Communication between HMI devices

The HTTP protocol is available for communication between HMI devices.

For more detailed information, refer to the documentation on the SIMATIC HMI HTTP protocol.

Communication via a uniform and vendor-neutral interface

With OPC (Openess Productivity Collaboration), WinCC has a uniform and manufacturer-neutral software interface. This interface enables standardized data exchange between industrial, office, and manufacturing applications.

For more detailed information, refer to the documentation for OPC.

10.8.1.2 Devices and networks in the automation system

Introduction

To set up an automation system, you must configure, parameterize, and interconnect the individual devices.

You insert PLCs and HMI devices into the project in the same way. Likewise, you configure the two devices in the same way.

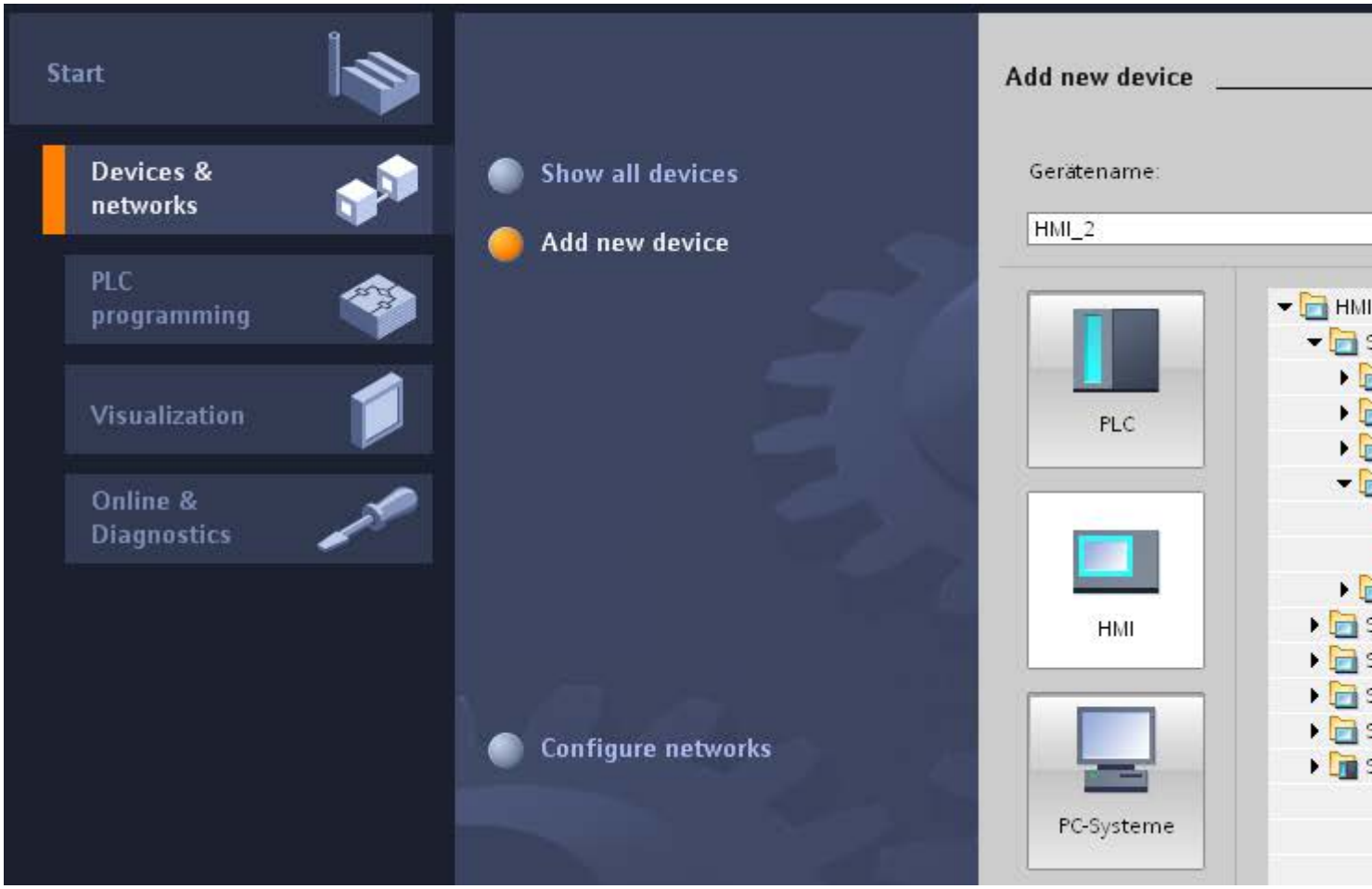
Automation system setup:

1. Insert PLC into the project.
2. Insert HMI device into the project.
3. Network the devices together.
4. Interconnect the devices.

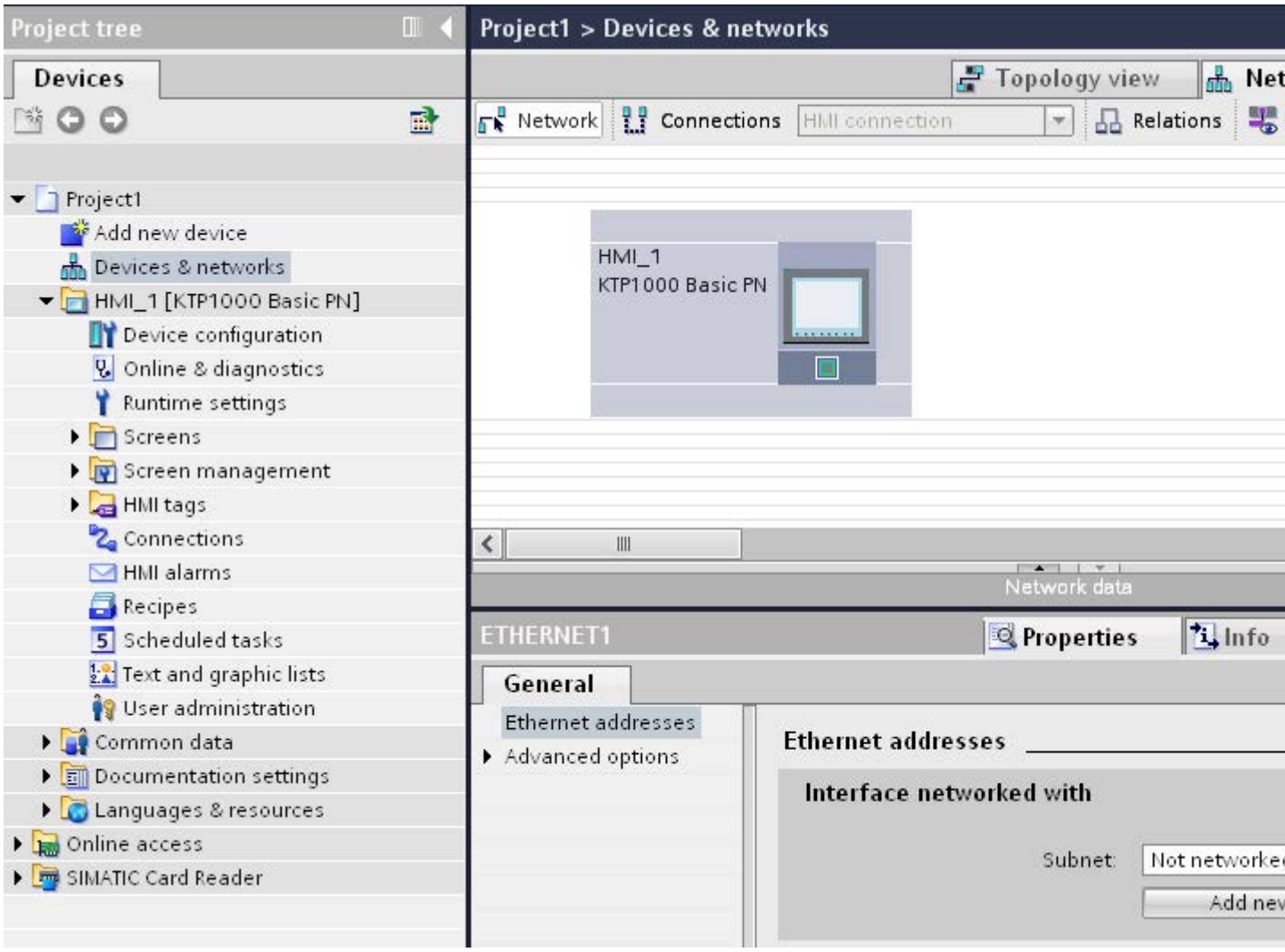
Inserting devices

If you have created a project, you can add a device in the portal view or project view.

- Portal view

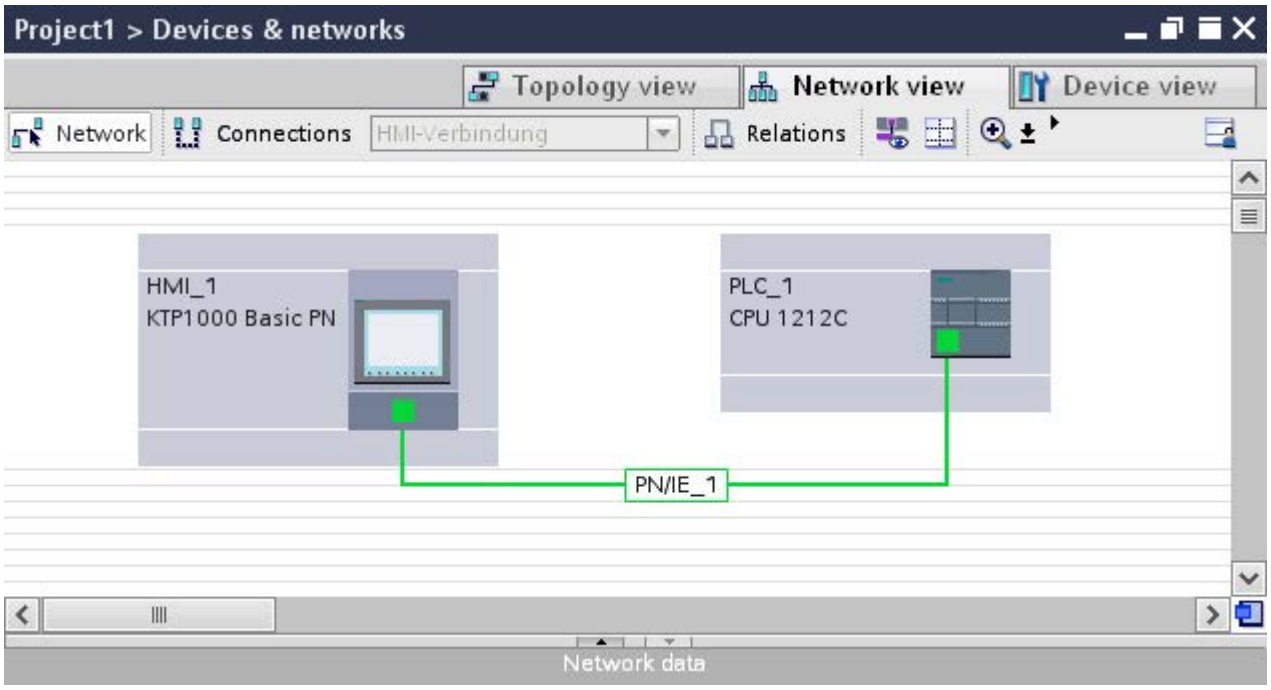


- Project view


Project tree
Devices
Project1
Add new device
Devices & networks
HMI_1 [KTP1000 Basic PN]
Device configuration
Online & diagnostics
Runtime settings
Screens
Screen management
HMI tags
Connections
HMI alarms
Recipes
Scheduled tasks
Text and graphic lists
User administration
Common data
Documentation settings
Languages & resources
Online access
SIMATIC Card Reader
Project1 > Devices & networks
Topology view
Network
Connections
HMI connection
Relations
HMI_1
KTP1000 Basic PN
Network data
ETHERNET1
Properties
Info
General
Ethernet addresses
Advanced options
Ethernet addresses
Interface networked with
Subnet:
Not networke
Add nev

Networking devices

You can network the interfaces of the communication-capable devices conveniently in the "Devices & Networks" editor. In the networking step, you configure the physical device connections.

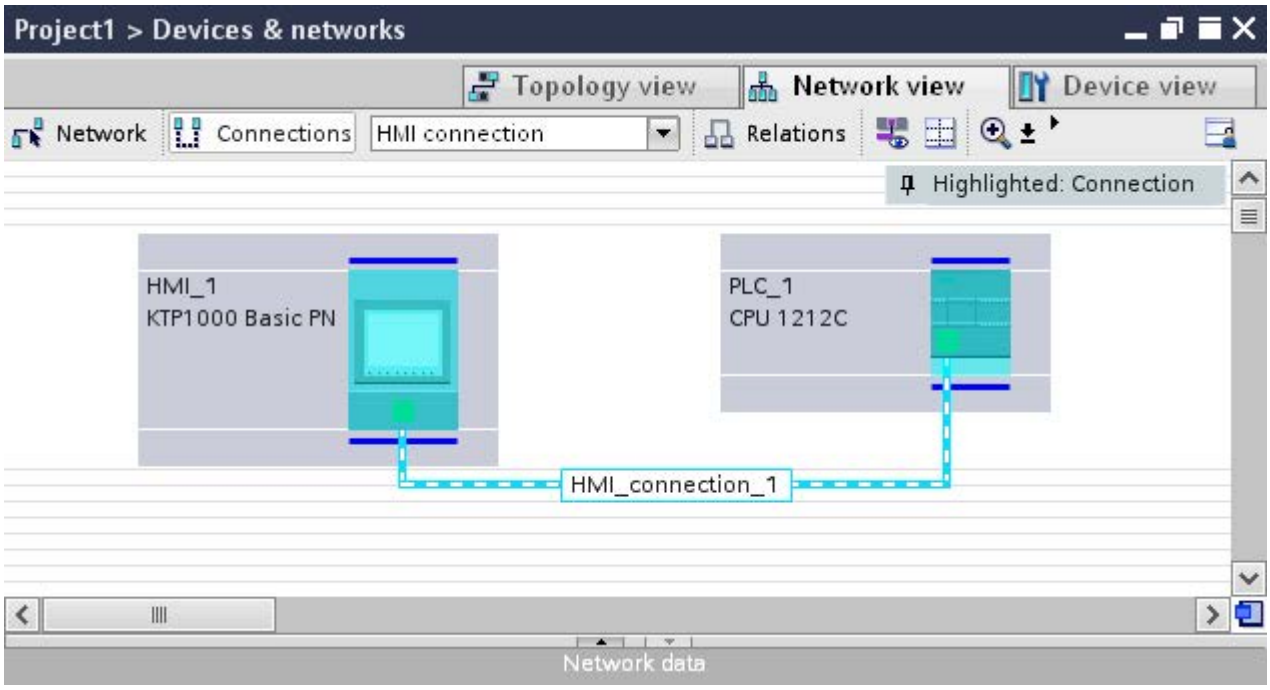


The tabular network overview supplements the graphical network view with the following additional functions:

- You obtain detailed information on the structure and parameter settings of the devices.
- Using the "Subnet" column, you can connect communication-capable components to subnets that have been created.

Connecting devices

After you network the devices together, you configure the connection. You configure the "HMI connection" connection type for communication with the HMI device.



10.8.1.3 Data exchange using tags

Communication using tags

Process values are forwarded in runtime using tags. Process values are data which is stored in the memory of one of the connected automation systems. They represent the status of a plant in the form of temperatures, fill levels or switching states, for example. Define external tags for processing the process values in WinCC.

WinCC works with two types of tag:

- External tags
- Internal tags

Working with tags

See the chapter "Working with tags (Page 8115)" for further information about configuring tags.

10.8.1.4 Data exchange using area pointers

Communication using area pointers

Area pointers are parameter fields. WinCC receives information from these parameter fields in Runtime during the course of the project. This information contains data on the location and size of data areas in the PLC.

During communication, the PLC and the HMI device alternately access these data areas for read and write operations.

Based on the evaluation of data stored in the data areas, the PLC and HMI device initiate mutually defined actions.

The area pointers are managed centrally in the "Connections" editor. Area pointers are used to exchange data from specific user data areas.

You use the following area pointers in WinCC:

- Data record
- Date/time
- Coordination
- Job mailbox
- Date/time PLC
- Project ID
- Screen number

The availability of the various area pointers is determined by the HMI device used.

10.8.1.5 Communication drivers

Communication drivers

A communication driver is a software component that establishes a connection between a PLC and an HMI device. The communication driver thus enables the assignment of process values to HMI tags.

The interface as well as the profile and transmission speed can be chosen, depending on the HMI device used and the connected communication partner.

10.8.2 Editors for communication

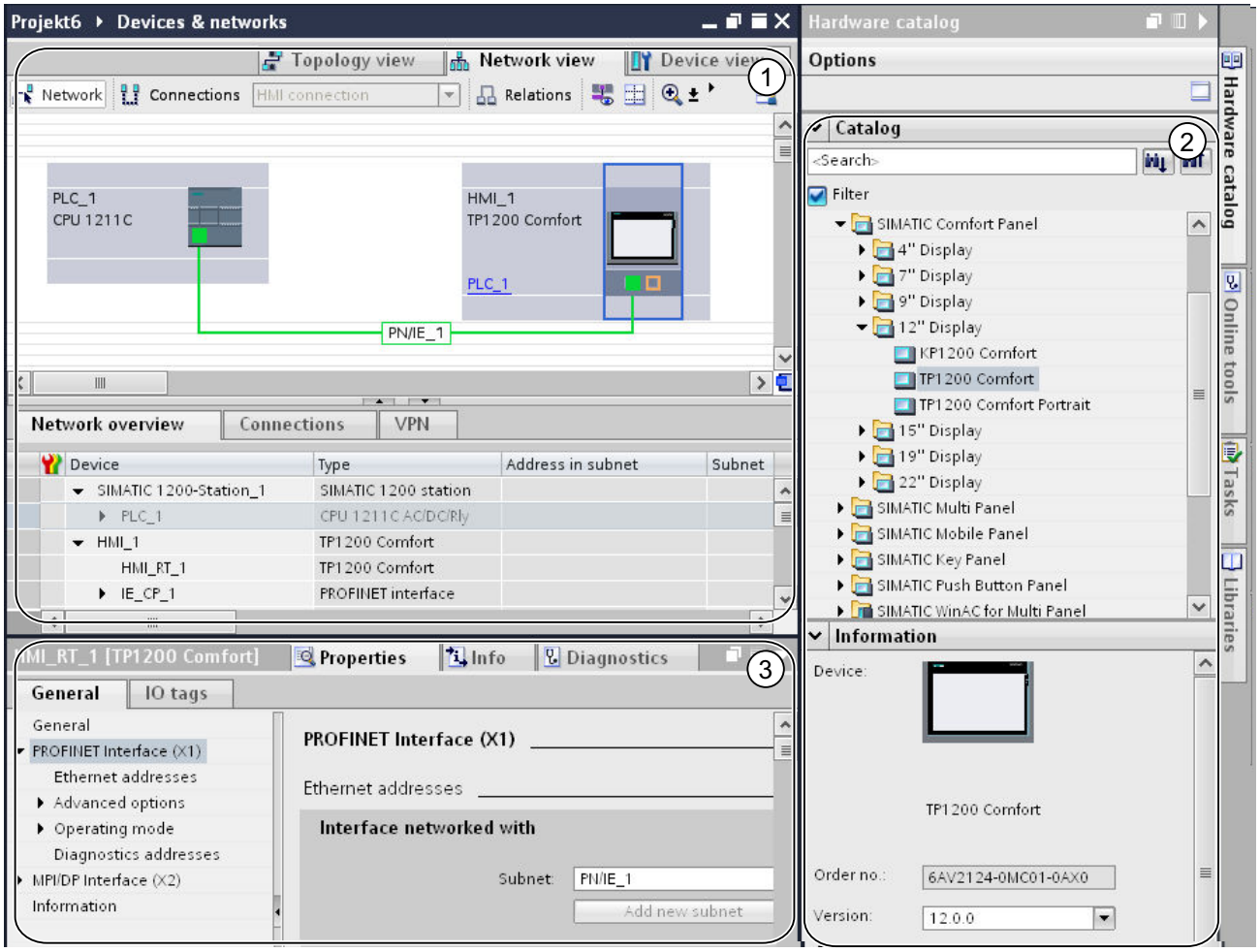
10.8.2.1 "Devices & networks" editor

Function of the hardware and network editor

The "Devices & networks" editor is the development environment for networking, configuring and assigning parameters to devices and modules.

Configuration

The "Devices & networks" editor consists of the following components:



1 Device view, network view, topology view
2 Hardware catalog
3 Inspector window

The "Devices & networks" editor provides you with three different views of your project. You can switch between these three views at any time depending on whether you want to produce and edit individual devices and modules, entire networks and device configurations or the topological structure of your project.

The inspector window contains information on the object currently marked. Here you can change the settings for the object marked.

Drag the devices and modules you need for your automation system from the hardware catalog to the network, device or topology view.

10.8.2.2 Network view

Introduction

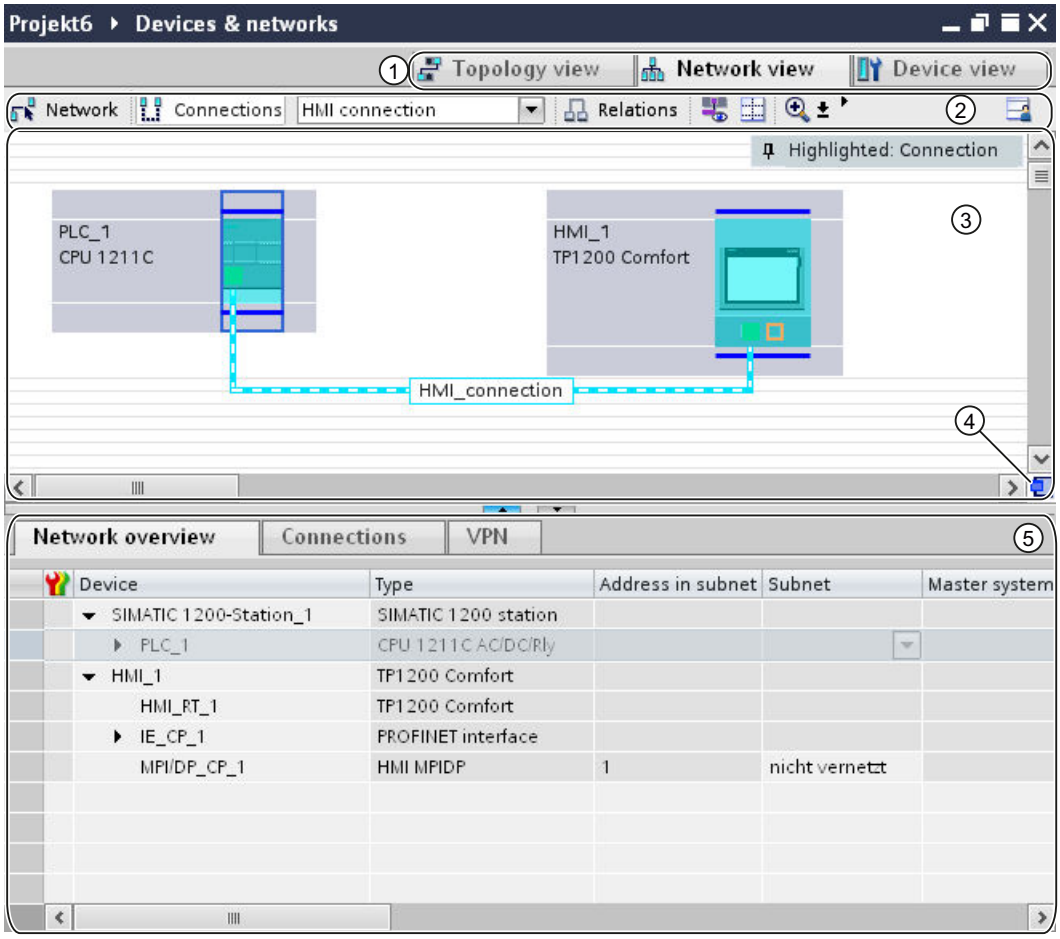
The network view is a working area of the hardware and network editor.

You undertake the following tasks in network view:

- Configuring and assign device parameters
- Networking devices with one another

Structure

The following diagram shows the components of the network view:



1 Changeover switch: network view/device view/topology view

2 Toolbar of network view

3 Graphic area of network view

4 Overview navigation

5 Table area of network view

You can use your mouse to change the spacing between the graphic and table areas of the network view.

To do this, click on the upper edge of the table view and expand or contract this by moving the mouse with the mouse button held down.

You can use the two small arrow keys to minimize, maximize or select the latest table structure of the table view with just one click.

Toolbar

The toolbar provides the following functions:

Icon	Meaning
	Mode to network devices.
	Mode to create connections. You can select the connection type using the adjacent drop-down list.
	Mode to create relations.
	Show interface addresses.
	Adjust the zoom setting. You can select the zoom setting or enter it directly in the adjacent drop-down list. You can also zoom in or zoom out the view in steps using the zoom symbol or draw a frame around an area to be zoomed in.
	Show page breaks Enables the page break preview. Dashed lines will be displayed at the positions where the pages will break when printed.
	Remember layout Saves the current table view. The layout, width and visibility of columns in the table view is stored.

Graphic area

The graphic area of the network view displays any network-related devices, networks, connections and relations. In this area, you add devices from the hardware catalog, connect them with each other via their interfaces and configure the communication settings.

Overview navigation

Click in the overview navigation to obtain an overview of the created objects in the graphic area. By holding down the mouse button, you can quickly navigate to the desired objects and display them in the graphic area.

Table area

The table area of the network view includes various tables for the devices, connections and communication settings present:

- Network overview
- Connections
- I/O communication

You can use the shortcut menu of the title bar of the table to adjust the tabular display.

10.8.2.3 Network data

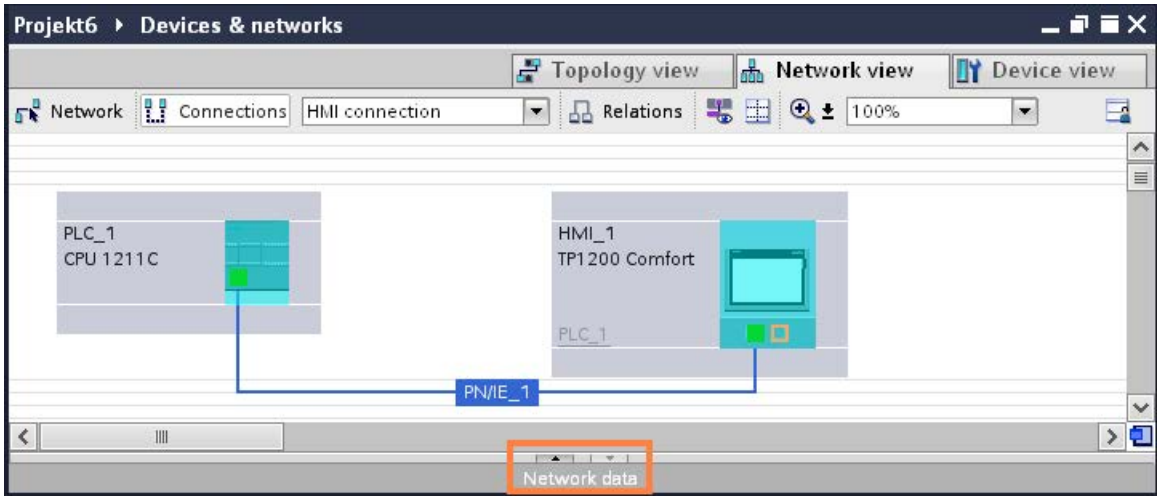
Introduction

The "Network view" editor also gives you a tabular view of the "Network data" in addition to the graphic network view.

You have the following selections in the "Network data" editor:

- Network overview
- Connections
- VPN
- I/O communication

You open the "Network data" below the graphic network overview.



Basic functions

The network data are displayed in tabular form and support the following basic functions for editing a table:

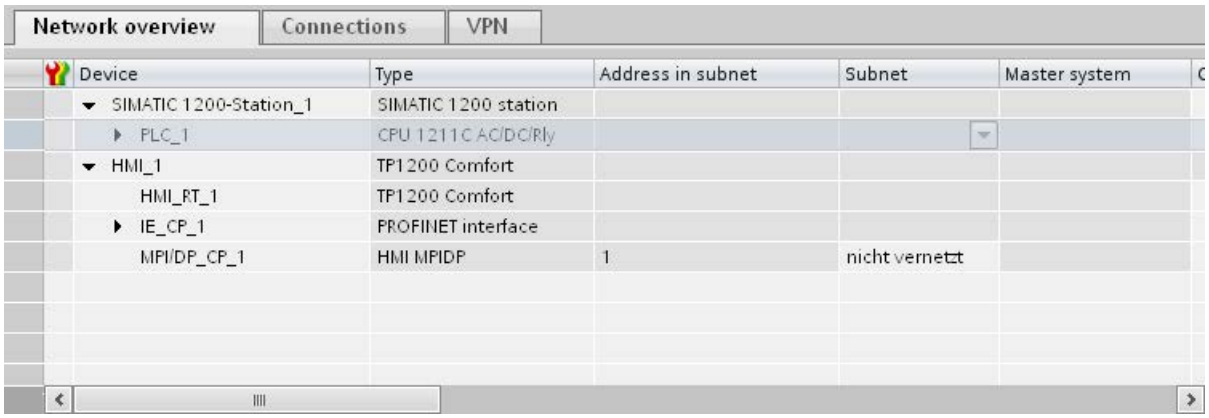
- Displaying and hiding table columns
  Note: The columns of relevance to configuration cannot be hidden.
- Optimizing column width
- Sorting table
- Displaying the meaning of a column, a row or cell using tooltips.

Network overview

The tabular network overview adds the following functions to the graphic network view:

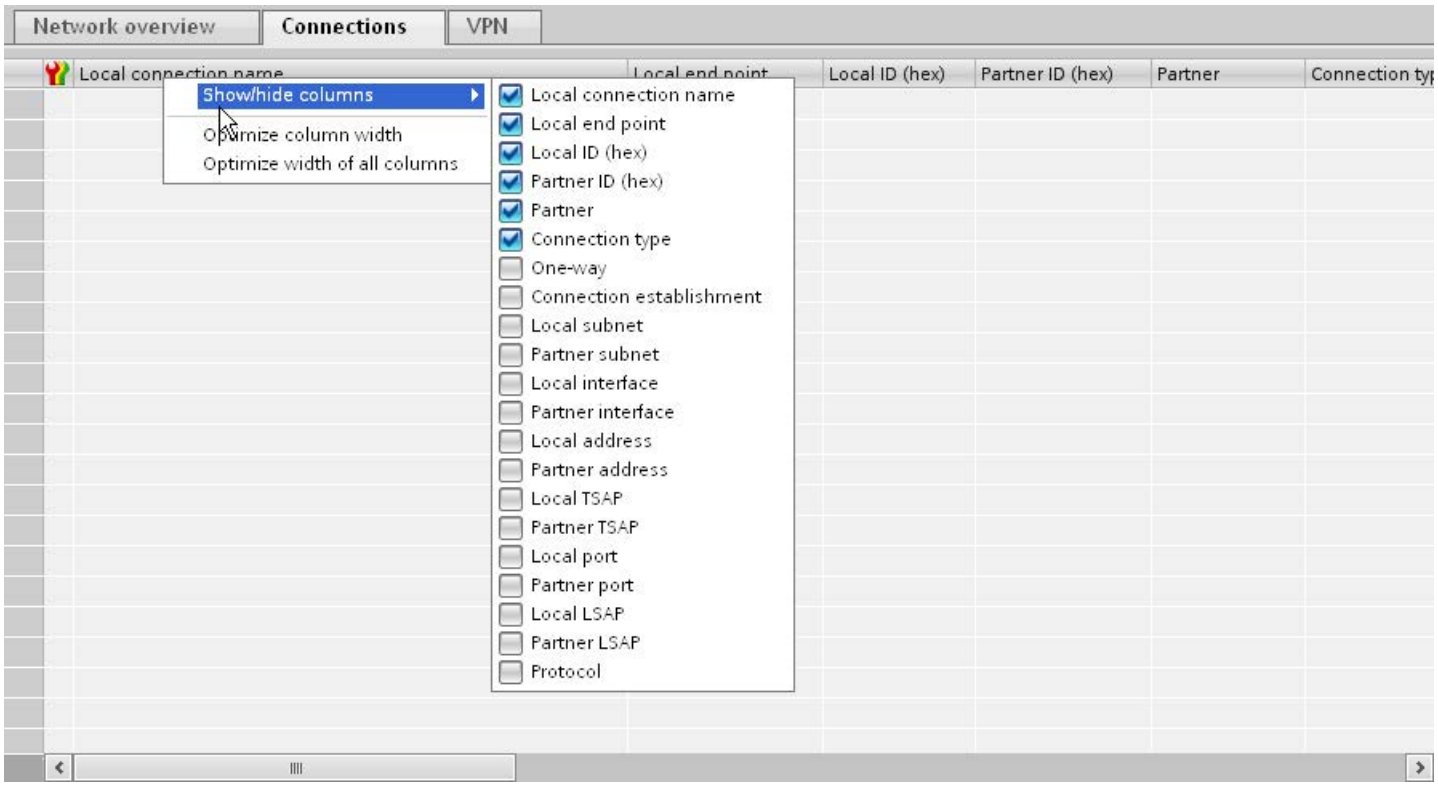
- You obtain detailed information on the structure and parameter settings of the devices.
- Using the "Subnet" column, you can connect components capable of communication with created subnets.

The



Connections

You will find additional network data under "Connections".



10.8.2.4 Diagnostics of online connections

Diagnostics of online connections

You can read out the diagnostics data of existing connections in the TIA Portal.

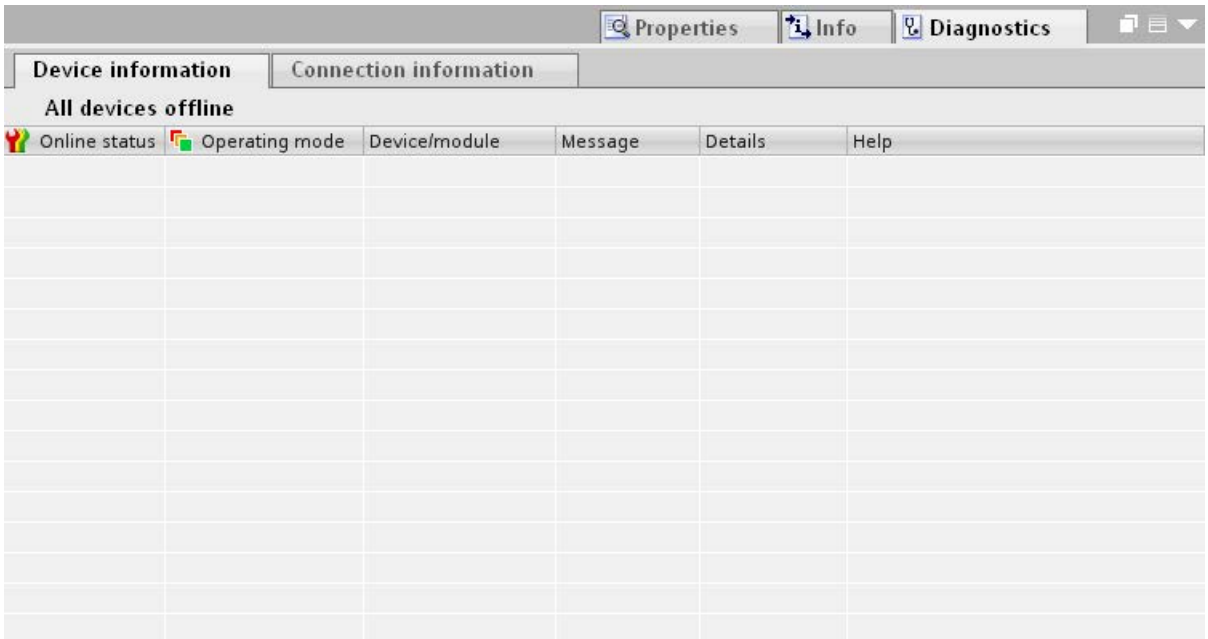
The "Diagnostics" function shows the connection data in tabular form in the Inspector window.

Requirements

- Devices must be in "Online" mode.

Device information

The diagnostics data of all devices in "Online" mode is displayed in the "Diagnostics > Device information" Inspector window.



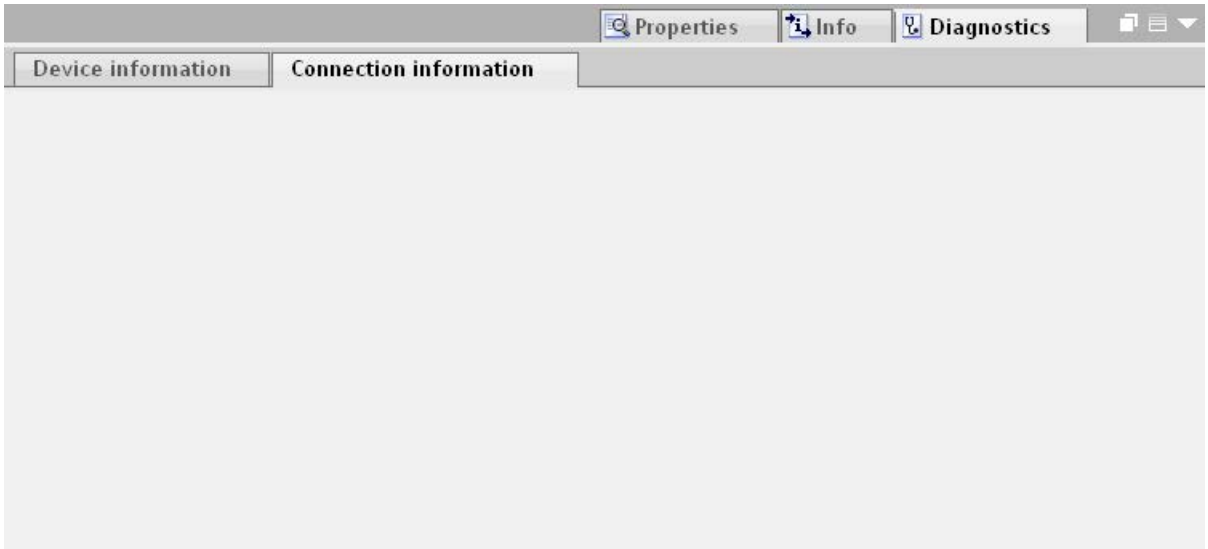
The following data is displayed:

- Online status
- Operating mode
- Device / module
- Alarm
- Details
- Help

Connection information

Use the "Connection information" function to display the diagnostics data of the connection selected in the "Devices&Networks" editor.

A graphic displays the communication partners of the connection and by which communication driver they are connected with each other.



The following data is displayed:

- End point
- Interface
- Subnet
- Address
- TSAP
- Number of HMI resources

10.8.2.5 Device view

Introduction

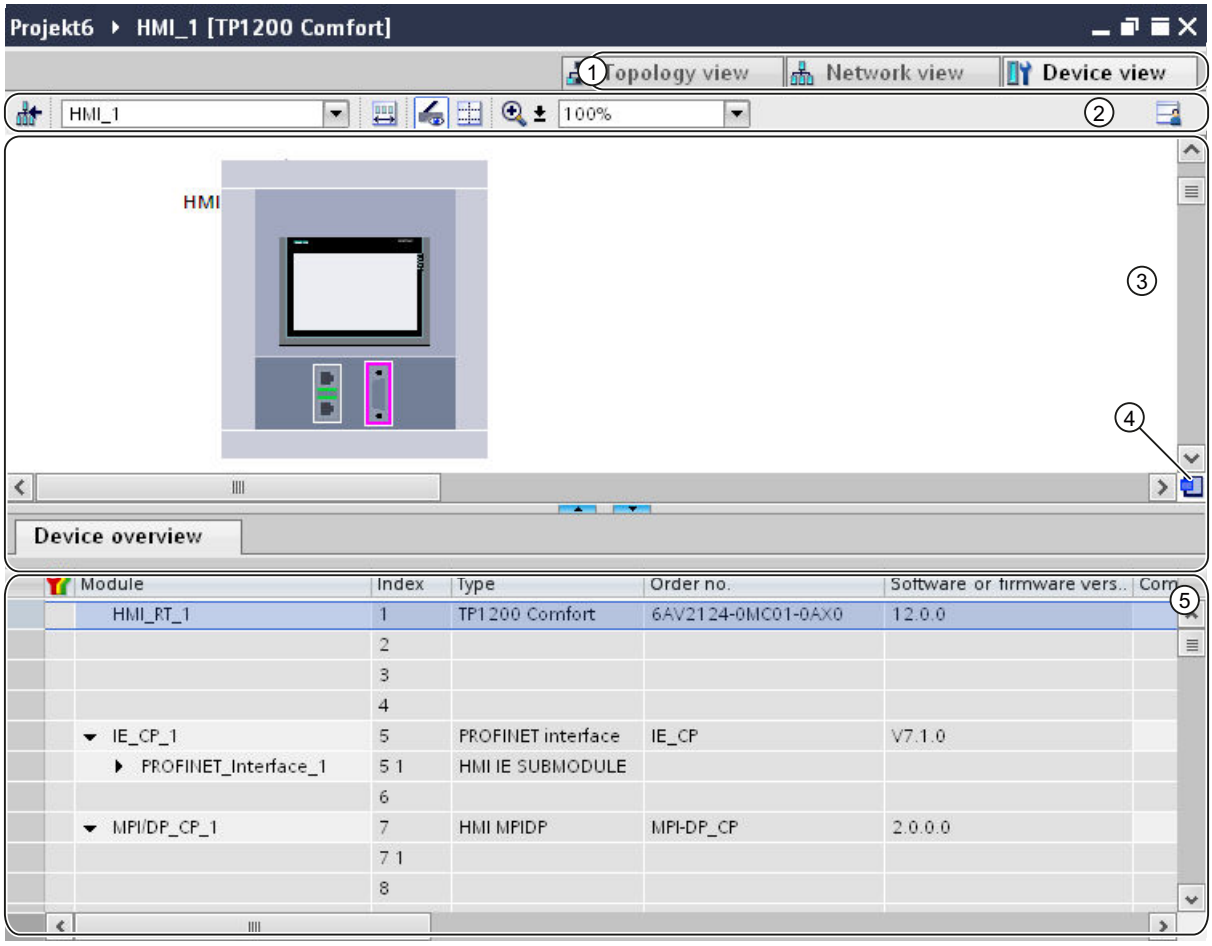
The device view is a working area of the hardware and network editor.

You undertake the following tasks in the device view:

- Configuring and assign device parameters
- Configuring and assign module parameters

Structure

The following diagram shows the components of the device view:



1 Changeover switch: network view/device view/topology view

2 Toolbar of device view

3 Graphic area of the device view

4 Overview navigation

5 Table area of device view

You can use your mouse to change the spacing between the graphic and table areas of the device view.

To do this, click on the upper edge of the table view and expand or contract this by moving the mouse with the mouse button held down. You can use the two small arrow keys to minimize, maximize or select the latest table structure of the table view with just one click.

Toolbar

The toolbar provides the following functions:

Icon	Meaning
	Switches to the network view. Note: The device view can switch between the existing devices using the drop-down list.
	Show the area of unplugged modules.
	Show module labels.
	Adjust the zoom setting. Select the zoom setting or enter it directly in the adjacent drop-down list. You can use the Zoom button to zoom in or out incrementally or to drag a frame around an area to be enlarged. You can read the address descriptions of the I/O channels of signal modules at a zoom level setting of 200% or higher.
	Show page breaks Enables the page break preview. Dashed lines will be displayed at the positions where the pages will break when printed.
	Remember layout Saves the current table view. The layout, width and visibility of columns in the table view is stored.

Graphic area

The graphic area of the device view displays hardware components and if necessary the associated modules that are assigned to each other via one or more racks. In the case of devices with racks, you have the option of installing additional hardware objects from the hardware catalog into the slots on the racks.

Overview navigation

Click in the overview navigation to obtain an overview of the created objects in the graphic area. By holding down the mouse button, you can quickly navigate to the desired objects and display them in the graphic area.

Table area

The table area of the device view gives you an overview of the modules used and the most important technical and organizational data.

You can use the shortcut menu of the title bar of the table to adjust the tabular display.

10.8.2.6 Topology view

Introduction

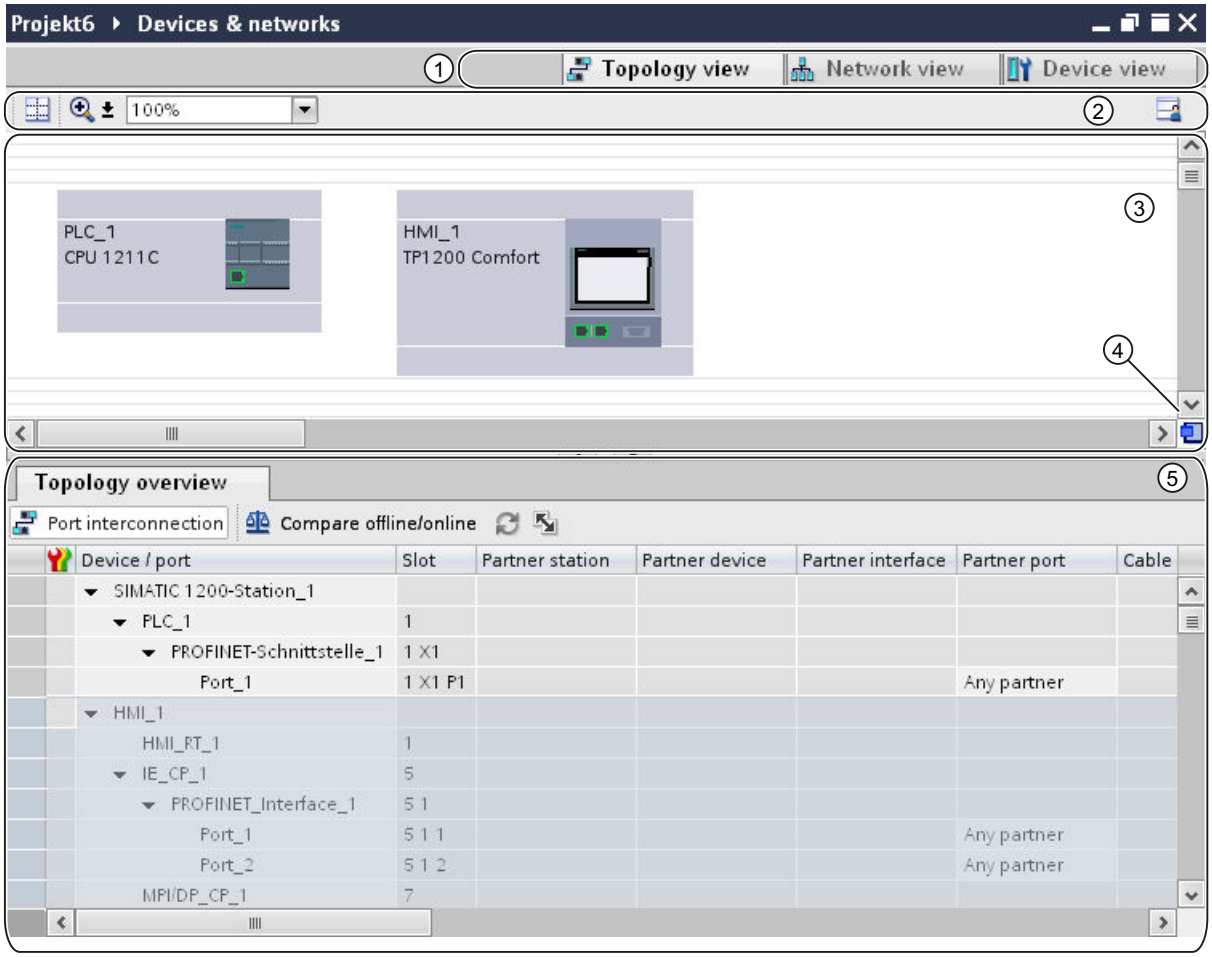
The topology view is a working area of the hardware and network editor.

You undertake the following tasks in topology view:

- Displaying the Ethernet topology
- Configuring the Ethernet topology
- Identify and minimize differences between the desired and actual topology

Structure

The following figure provides an overview of the topology view.



1 Changeover switch: device view / network view / topology view

2 Topology view toolbar

3 Graphic area of the topology view

4 Overview navigation

5 Table area of the topology view

You can use your mouse to change the spacing between the graphic and table areas of the topology view.

To do this, click on the upper edge of the table view and expand or contract this by moving the mouse with the mouse button held down. You can use the two small arrow keys to minimize, maximize or select the latest table structure of the table view with just one click.

Toolbar

The toolbar provides the following functions:

Icon	Meaning
	Adjust the zoom setting. You can select the zoom setting via the adjacent drop-down list or enter it directly. You can also zoom in or zoom out the view in steps using the zoom symbol or draw a frame around an area to be zoomed in.
	Show page breaks Enables the page break preview. Dashed lines will be displayed at the positions where the pages will break when printed.
	Remember layout Saves the current table view. The layout, width and visibility of columns in the table view is stored.

Graphic area

The graphic area of the topology view displays Ethernet modules with their appropriate ports and port connections. Here you can add additional hardware objects with Ethernet interfaces.

Overview navigation

Click in the overview navigation to obtain an overview of the created objects in the graphic area. By holding down the mouse button, you can quickly navigate to the desired objects and display them in the graphic area.

Table area

This displays the Ethernet or PROFINET modules with their appropriate ports and port connections in a table. This table corresponds to the network overview table in the network view.

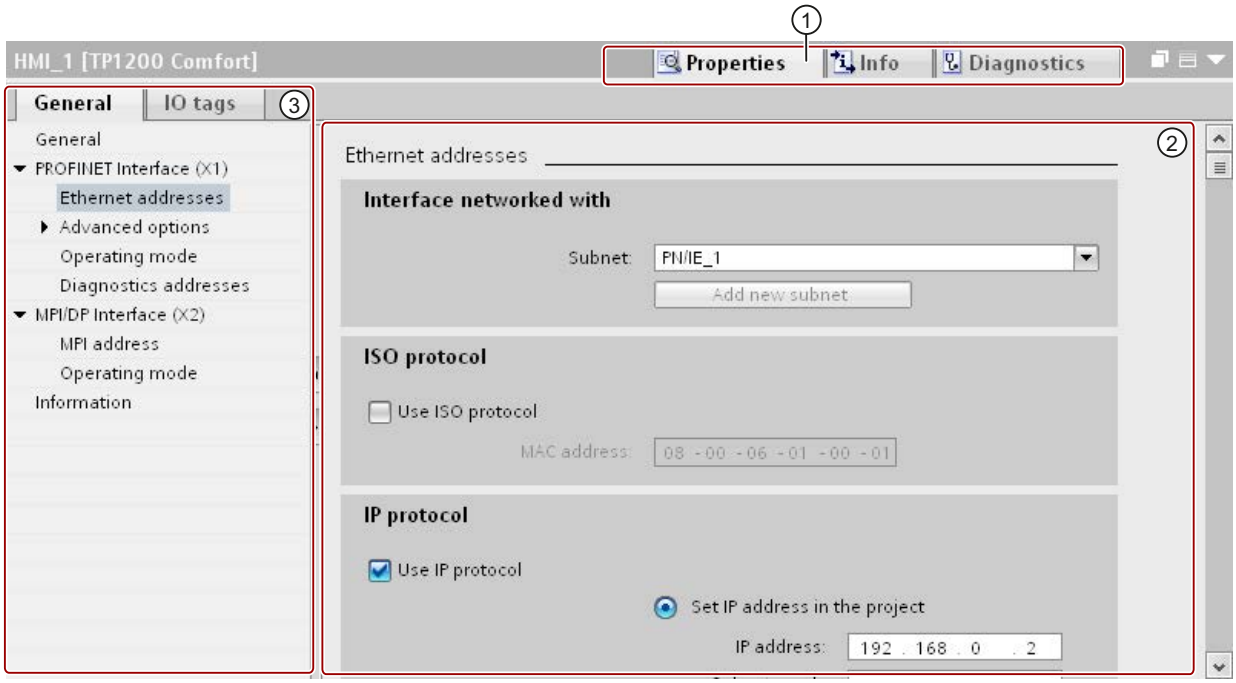
You can use the shortcut menu of the title bar of the table to adjust the tabular display.

10.8.2.7 Inspector window

The properties and parameters shown for the object selected can be edited in the inspector window.

Structure

The inspector window consists of the following components:



1 Switch between various information and work areas

2 Navigation between various pieces of information and parameters

3 Display showing the selected information and parameters

Function

The information and parameters in the inspector window are split into different types of information:

- Properties
- Info
- Diagnostics

To display the corresponding information and parameters, click in the area you want. The "Properties" area is the most important one for configuring an automation system. This area is displayed by default.

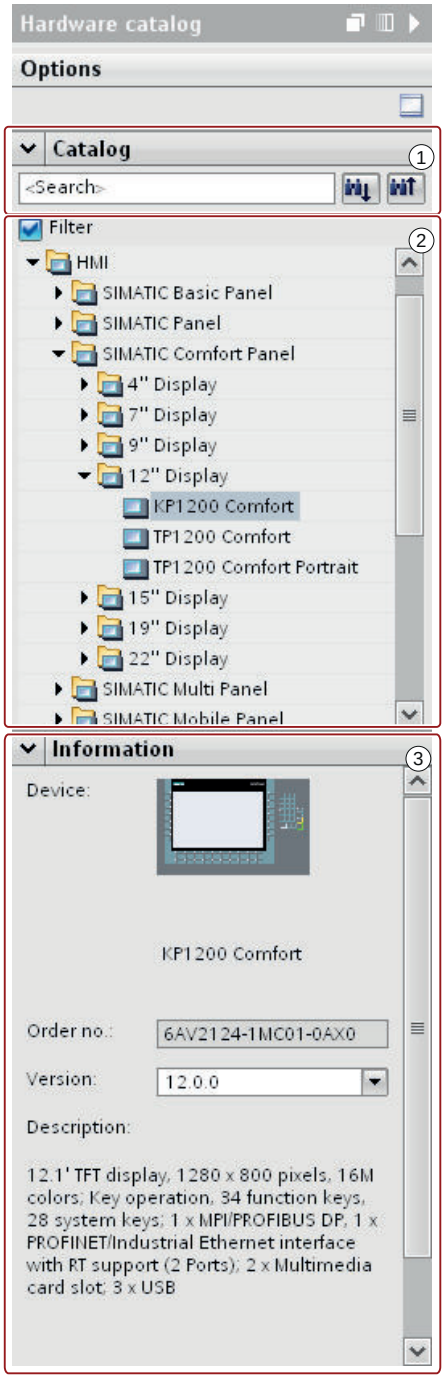
The left pane of the inspector window is used for area navigation. Information and parameters are arranged there in groups. If you click on the arrow symbol to the left of the group name, you can expand the group if sub-groups are available. If you select a group or sub-group, the corresponding information and parameters are displayed in the right pane of the inspector window and can be edited there too.

10.8.2.8 Hardware catalog

The "Hardware catalog" task card gives you easy access to a wide range of hardware components.

Structure

The "Hardware catalog" task card consists of the following panes:



1 "Catalog" pane, search and filter function

2 "Catalog" pane, component selection

3 "Information" pane

Search and filter function

The search and filter functions of the "Catalog" pane make it easy to search for particular hardware components. You can limit the display of the hardware components to certain criteria using the filter function. For example, you can limit the display to objects that you can also place within the current context or which contain certain functions.

Objects that can be used in the current context include, for example, interconnectable objects in the network view or only modules compatible with the device in the device view.

Component selection

The component selection in the "Catalog" pane contains the installed hardware components in a tree structure. You can move the devices or modules you want from the catalog to the graphic work area of the device or network view.

Installed hardware components without a license are grayed out. You cannot use non-licensed hardware components.

Hardware components belonging to various components groups thematically are partially implemented as linked objects. When you click on such linked hardware components, a catalog tree opens in which you can find the appropriate hardware components.

Information

The "Information" pane contains detailed information on the object selected from the catalog:

- Schematic representation
- Name
- Order number
- Version number
- Description

10.8.2.9 Information on hardware components

In the hardware catalog, you can display information on selected hardware components in the "Information" pane. You can also display further information on the selected hardware components using the shortcut menu.

Access to further information

If you select a hardware object in the hardware catalog and open the shortcut menu, you not only have the "Copy" function available but also three options for accessing information on Service & Support:

- Information regarding product support
- FAQs
- Manuals

The required information is displayed in the work area of the hardware and network editor.

Note

You can only access Service & Support when you are connected to the Internet and the function is enabled. By default, the function is disabled.

To enable the function, refer to the instructions in the section "Enabling product support".

Information regarding product support

Here, you have access to general information on hardware and software components. The order number of the selected hardware object is entered automatically in the search mask. You can, however, also search for other hardware and software components.

FAQs

Here, you have access to "Frequently Asked Questions" (FAQs). You can view various entries on hardware and software questions. Using a detailed search mask, you can filter the required topics.

Manuals

Here, you have access to the manuals of the various hardware components. This is particularly useful if the configuration, addressing or parameter assignment you are planning requires more detailed knowledge of the hardware you are using.

10.8.3 Networks and connections

10.8.3.1 SIMATIC communication networks

Communication networks

Overview

Communication networks are a central component of modern automation solutions. Industrial networks have to fulfill special requirements, for example:

- Coupling of automation systems as well as simple sensors, actuators, and computers.
- The information has to be correct and has to be transferred at the right moment.
- Robust against electromagnetic disturbances, mechanical stresses and soiling
- Flexible adaptation to the production requirements

Industrial networks belong to the LANs (Local Area Networks) and allow communication within a limited area.

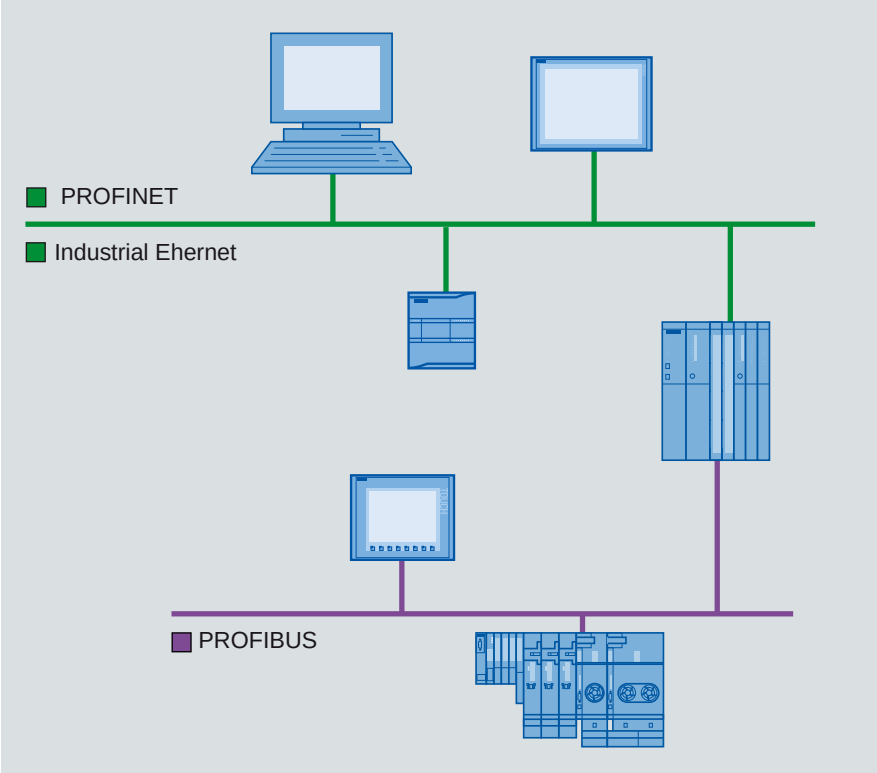
Industrial networks fulfill the following communication functions:

- Process and field communication of the automation systems including sensors and actuators
- Data communication between automation systems
- IT communication for integrating the modern information technology

Overview of the networks

This section examines the following networks:

- **Industrial Ethernet**
  The industrial network standard for all levels
- **PROFINET**
  The open Industrial Ethernet standard for automation
- **PROFIBUS**
  The international standard for the field area and market leader at the field busses
- **MPI**
  The integrated interface of the SIMATIC products
- **PPI**
  The integrated interface specially for the S7-200



PROFINET and Ethernet

Industrial Ethernet

Industrial Ethernet, which is based on IEEE 802.3, enables you to connect your automation system to your office networks. Industrial Ethernet provides IT services that you can use to access production data from the office environment.

Ethernet network

An Ethernet network allows you to interconnect all devices that are connected to the network via an integrated Ethernet interface or a communication module. This enables connection of multiple HMI devices to one SIMATIC S7 PLC and multiple SIMATIC S7 PLCs to one HMI device. The maximum number of communication partners that you can connect to an HMI device is dependent on the HMI device used. Additional information is available in the documentation for the respective HMI device.

PROFINET

PROFINET is an open standard for industrial automation defined by IEEE 61158 and based on Industrial Ethernet. PROFINET makes use of IT standards all the way to the field level and enables plant-wide engineering.

With PROFINET, you can realize high-performance automation solutions for applications with stringent real-time requirements.

PROFIBUS

PROFIBUS DP

PROFIBUS DP (distributed I/O) is used to connect the following devices:

- PLCs, PCs, HMI devices
- Distributed I/O devices, e.g., SIMATIC ET 200
- Valves
- Drives

PROFIBUS DP's fast response times make it ideally suited for the manufacturing industry.

Its basic functionality includes cyclic exchange of process data between the master and PROFIBUS DP slaves, as well as diagnostics.

PROFIBUS network

You can connect an HMI device within the PROFIBUS network to any SIMATIC S7 module that has an integrated PROFIBUS or PROFIBUS DP interface. You can thereby connect multiple HMI devices to one SIMATIC S7 PLC and multiple SIMATIC S7 PLCs to one HMI device.

The maximum number of communication partners that you can connect to an HMI device is dependent on the HMI device used. Additional information is available in the documentation for the respective HMI device.

You configure the SIMATIC S7-200 PLC as a passive device in the network. You connect the SIMATIC S7-200 using the DP connector or a PROFIBUS communication module.

MPI

MPI

MPI (Multi-Point Interface) is the integrated interface for SIMATIC products:

- PLCs
- HMI devices
- Programming device/PC

Small subnets with the following characteristics are set up with MPI:

- Short distances
- Few devices
- Small data quantities

MPI network

You connect the HMI device to the MPI interface of the SIMATIC S7 PLC. This enables connection of multiple HMI devices to one SIMATIC S7 PLC and multiple SIMATIC S7 PLCs to one HMI device. The maximum number of communication partners that you can connect to an HMI device is dependent on the HMI device used. Additional information is available in the documentation for the respective HMI device.

Network architectures

MPI is based on the PROFIBUS standard (IEC 61158 and EN 50170) and supports the following bus topologies:

- Line
- Star
- Tree

An MPI subnet contains a maximum of 127 devices and consists of multiple segments. Each segment contains a maximum of 32 devices and is limited by terminating resistors. Repeaters are used to connect segments. The maximum cable length without a repeater is 50 m.

PPI

Introduction

PPI (point-to-point interface) is an integrated interface that was developed specially for the SIMATIC S7-200. A PPI network typically connects S7-200 PLCs. However, other SIMATIC PLCs (e.g., S7-300 and S7-400) or HMI devices can communicate with a SIMATIC S7-200 in the PPI network.

PPI network

A PPI connection is a point-to-point connection. The HMI device is the master. The SIMATIC S7-200 is the slave.

You can connect a maximum of one SIMATIC S7-200 to an HMI device. You use the serial connector of the CPU to connect the HMI device. You can connect multiple HMI devices to one SIMATIC S7-200. From the perspective of the SIMATIC S7-200, only one connection at a time is possible.

Note

The PPI network can contain a maximum of four masters in addition to the HMI device. For performance reasons, do not configure more than four devices at a time as a master in the PPI network.

Network architectures

PPI is based on the PROFIBUS standard (IEC 61158 and EN 50170) and supports the following bus topologies:

- Line
- Star

Multi-master networks with a maximum of 32 masters are set up with PPI:

- An unlimited number of masters can communicate with each slave.
- A slave can be assigned to multiple masters.

The RS 485 repeater can be used to extend the PPI network. Modems can also be connected to the PPI network.

10.8.3.2 Configuring networks and connections

Networking devices

Introduction

The "Devices & Networks" editor is provided for configuring connections. You can network devices in the editor. You can also configure and assign parameters to devices and interfaces. You then configure the required connections between the networked devices.

In the "Devices & Networks" editor you configure HMI connections with the PLCs:

- SIMATIC S7 1500
- SIMATIC S7 1200
- SIMATIC S7 300
- SIMATIC S7 400

You configure the HMI connections to other PLCs in the "Connections" editor of the respective HMI device.

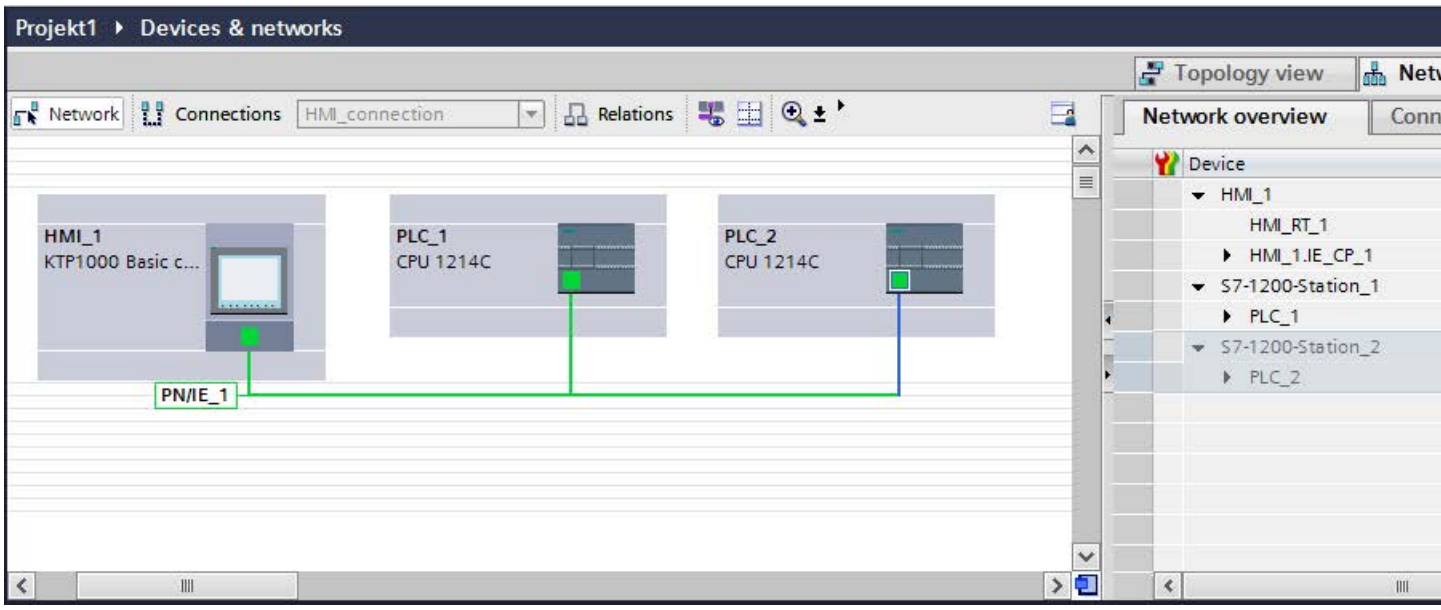
Networking devices

The network view of the "Devices & Networks" editor includes a graphical area and a tabular area. You can use the graphical area to network the devices in the project with drag-and-drop. The tabular area provides an overview of the devices and their components.

You can network the following PLCs together with HMI devices in the "Devices & Networks" area.

- SIMATIC S7 1500
- SIMATIC S7 1200
- SIMATIC S7 300
- SIMATIC S7 400

All other PLCs are available in the TIA-Portal and are configured "not integrated". You configure "not integrated" connections in the "Connections" editor of the HMI device.



With the networking step, you configure the physical connection of the communication partners. The networking of devices is depicted by lines that are colored according to the interface.

Configuring an integrated connection in the "Devices & Networks" editor

Introduction

You configure an HMI connection between the HMI device and a SIMATIC S7 PLC in the "Devices & Networks" editor. This HMI connection is the direct connection between the communication partners that you have created in a project.

Integrated connections

Connections of devices within a project are referred to as integrated connections. In the case of integrated connections, you can directly configure addresses of PLC tags.

Note

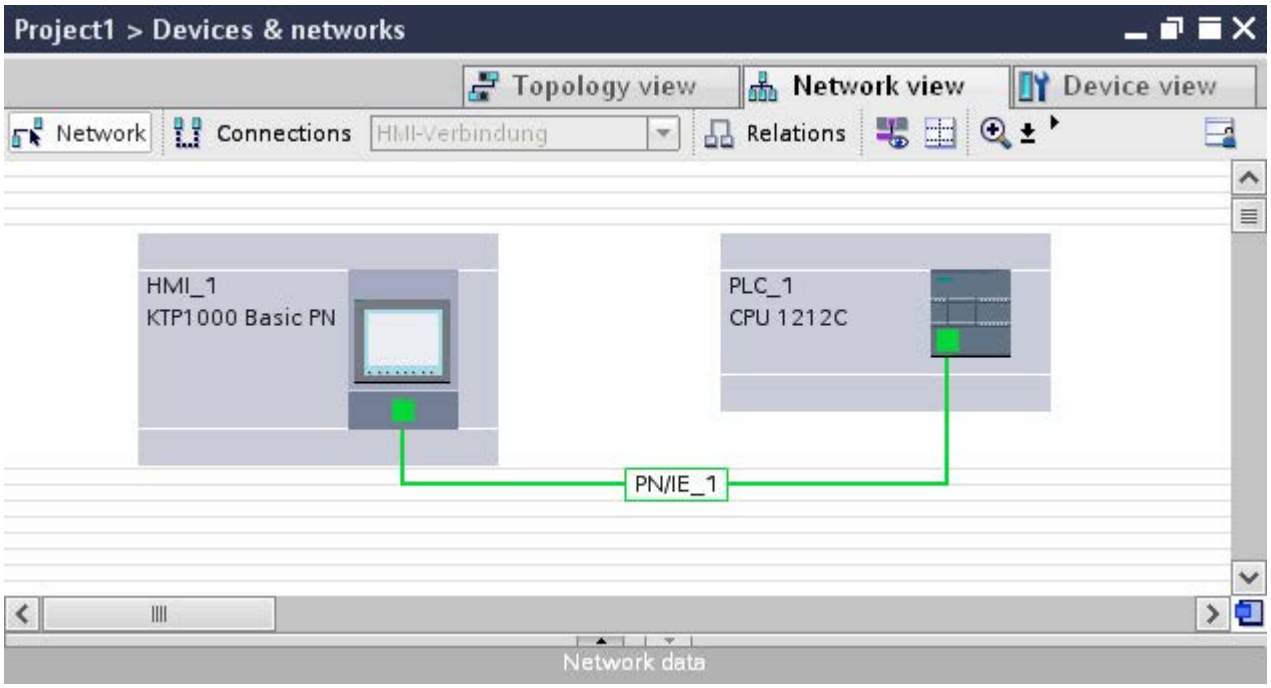
An HMI connection can be configured in the "Devices & Networks" editor for the following PLCs only:

- SIMATIC S7 1500
- SIMATIC S7 1200
- SIMATIC S7 300
- SIMATIC S7 400

You configure the HMI connections to all other PLCs in the "Connections" editor of the HMI device.

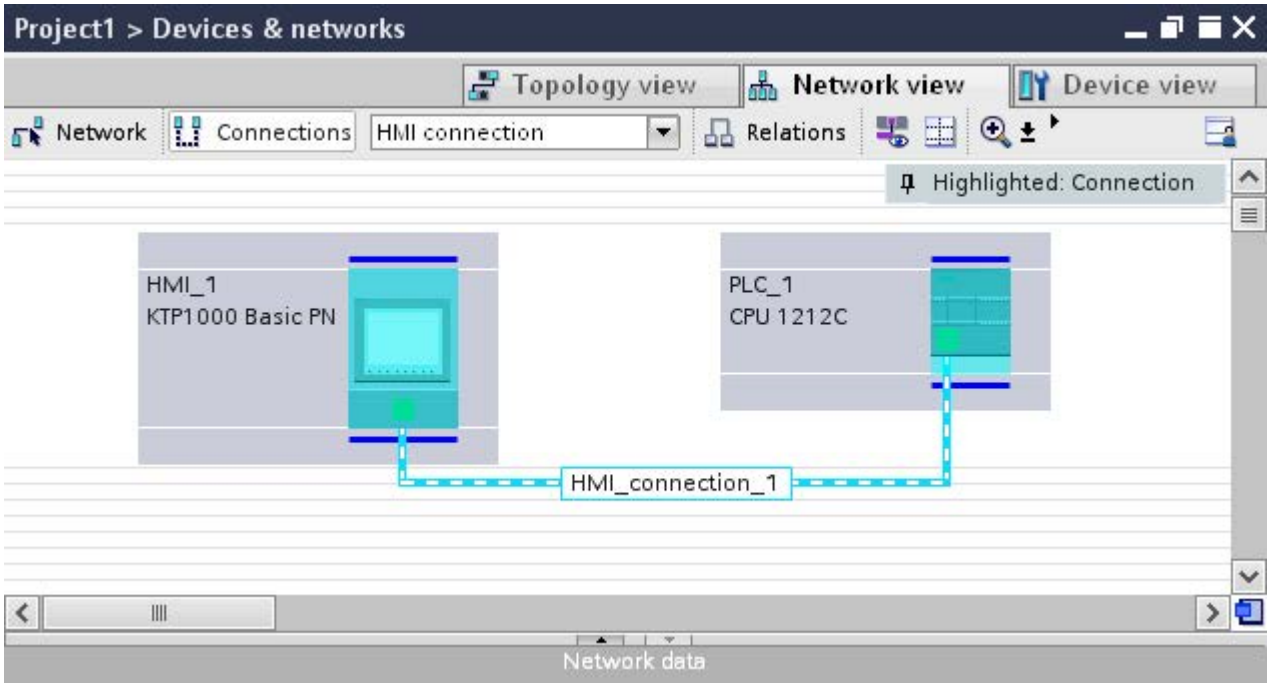
Configuring an HMI connection in the "Devices & Networks" editor

1. Insert an HMI device and a SIMATIC S7 PLC in your project.



2. Switch to "Connections" mode.
3. Select the "HMI connection" connection type.

4. Use a drag-and-drop operation to interconnect the two PROFINET interfaces.



5. Change the IP address and subnet mask address parameters according to the requirements of your project.

Special considerations of the "Devices & Networks" editor

Introduction

If you are configuring or have already configured networks or HMI connections, the "Devices & Networks" editor supports you with the following functions:

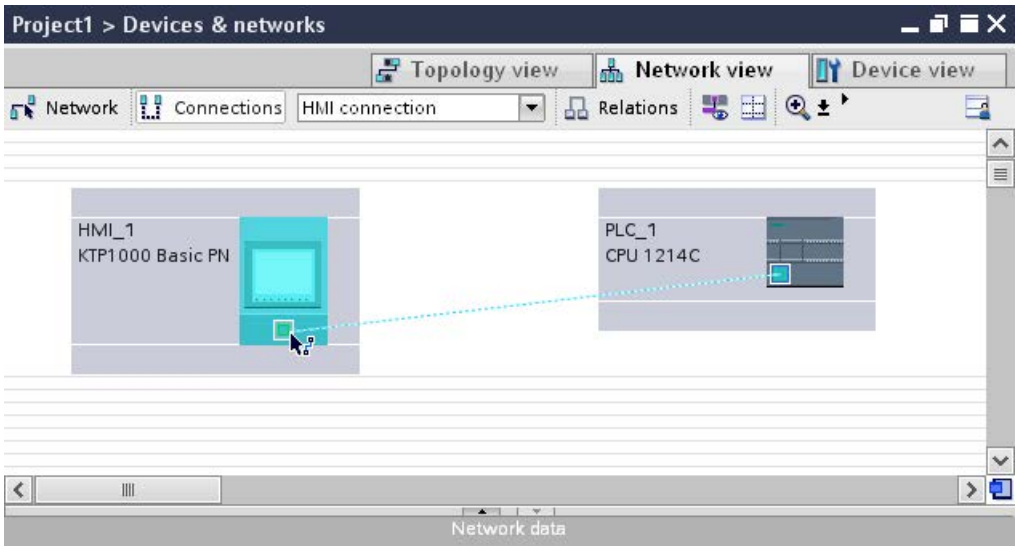
- Highlighting of communication partners
- Highlighting of HMI connections
- Automatic creation of subnets

Highlighting of communication partners

All communication partners for which an HMI connection is possible are highlighted in turquoise if you have selected the "HMI connection" type.

Starting from the interface of a device create an HMI connection to the device of another device using a drag-and-drop operation. During the drag-and-drop operation all potential communication partners are highlighted in turquoise.

Use the ESC key to stop connecting interfaces using a drag-and-drop operation.



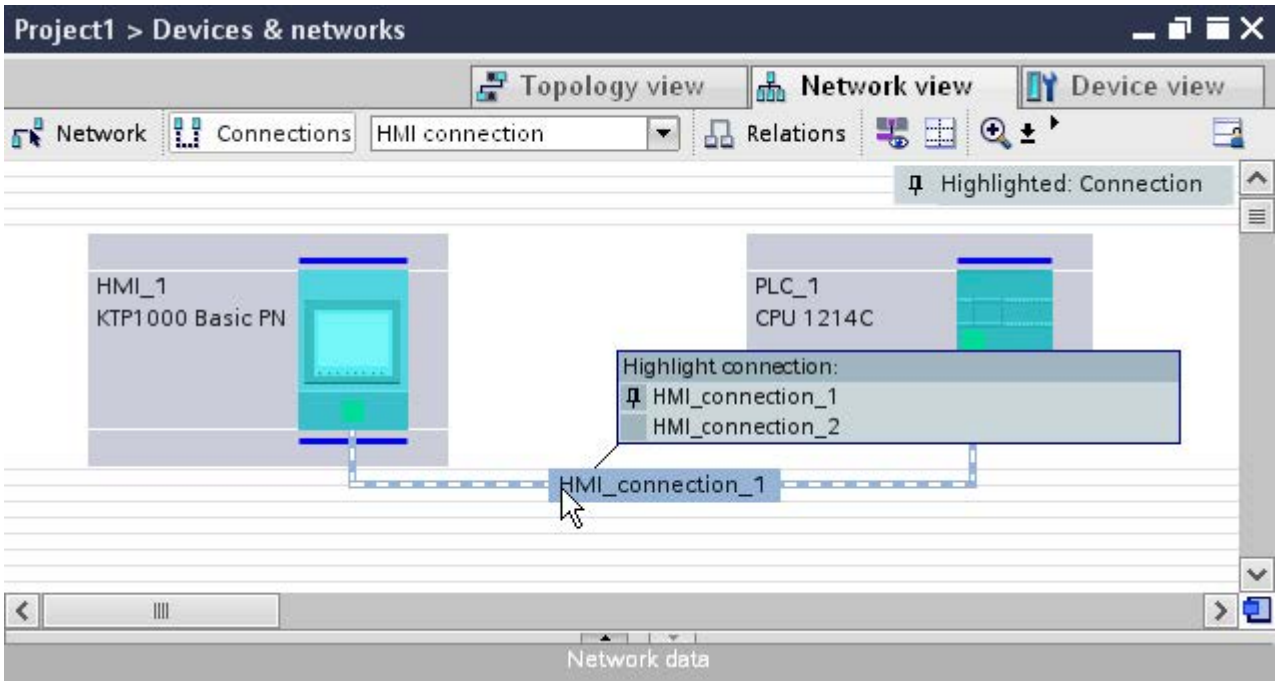
When the mouse pointer is moved over the interface of a device, the following icons indicate whether a connection is possible:

A connection is possible.

A connection is not possible.

Highlighting of HMI connections

A turquoise highlighting of the connection indicates that a HMI connection was created. If several HMI connections are created, you can select one of the already created HMI connections in a dialog.



Then you can configure the parameters of the selected HMI connection and the communication partners in the inspector window.

Subnets

Subnetworks are automatically created and used under the following conditions:

- If both communication partners are not already interconnected in different networks
- If a free interface is available to both communication partners.
- If a subnetwork already exists then that subnetwork is automatically used for the HMI connection.

Configuring a non-integrated connection in the "Connections" editor

Introduction

You use the "Connections" editor of the HMI device to configure a connection between an HMI device and a PLC that cannot be configured in the "Devices & Networks" editor.

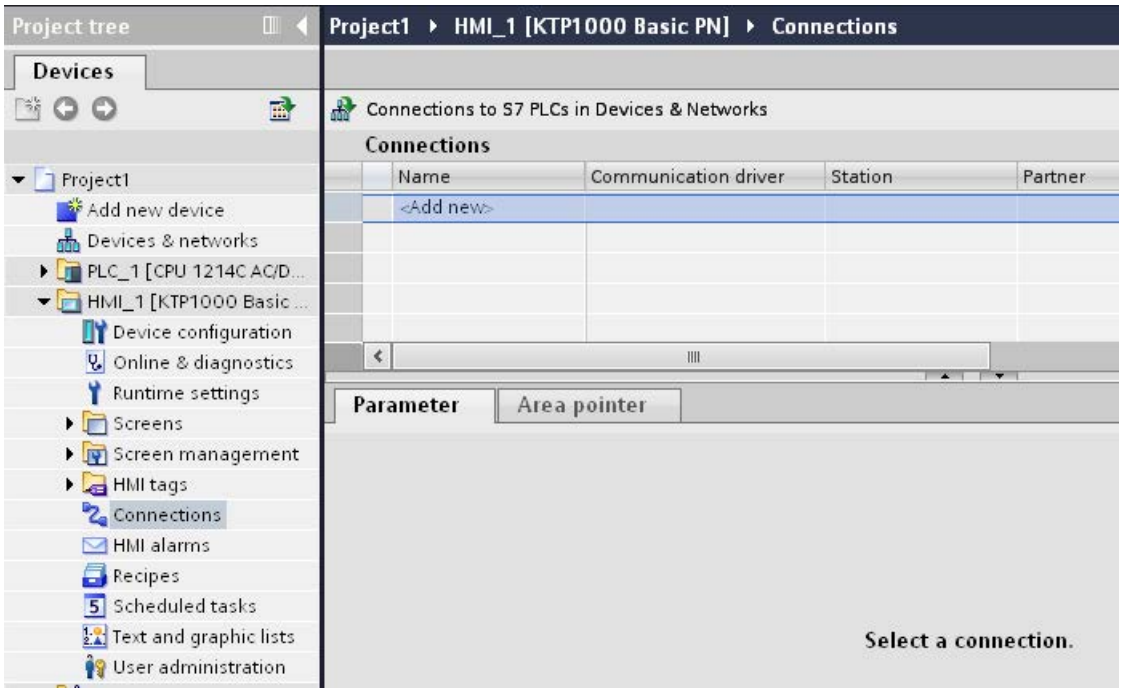
These connections are referred to as non-integrated connections.

Requirements

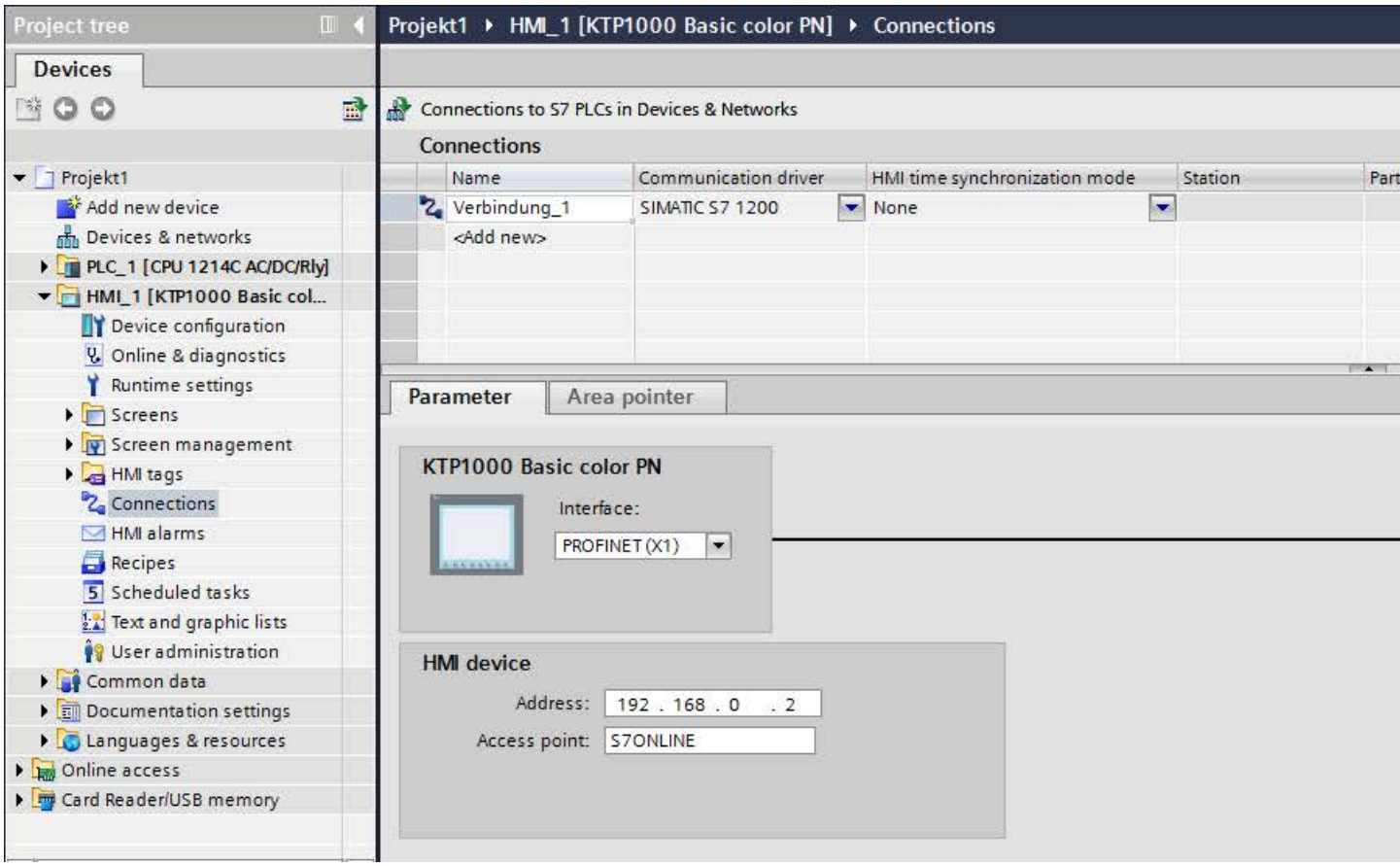
- A project is open.
- An HMI device has been created.

Configuring a connection in the "Connections" editor

1. Open the "Connections" editor of the HMI device.
2. Create a new connection.

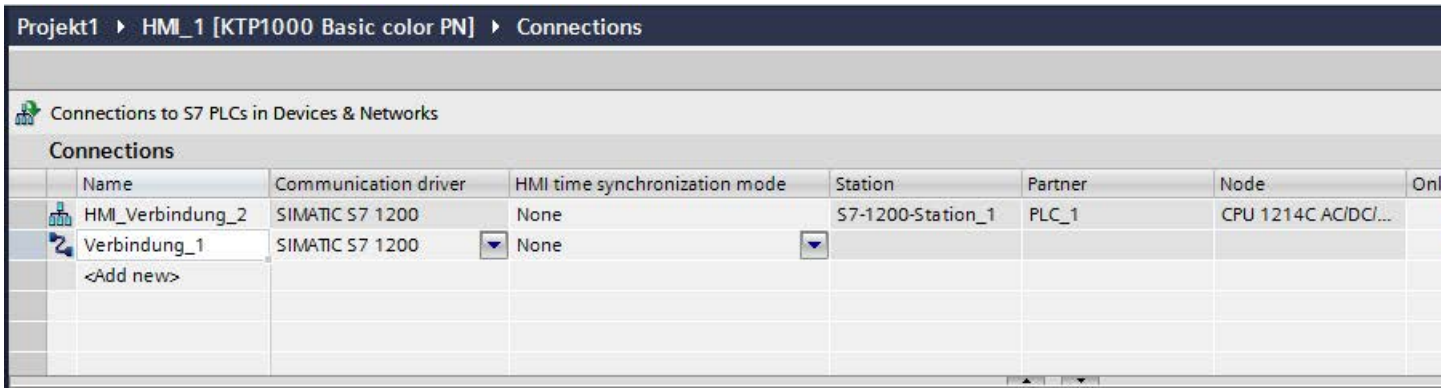


3. Select the communication driver.
4. Set the connection parameters.



Integrated connections in the "Connections" editor

If you have already configured the integrated connections of the HMI device in the "Devices & Networks" editor, they are also displayed in the "Connections" editor.



Meaning of the icons used:

Integrated connection

Non-integrated connection

10.8.4 Data exchange

10.8.4.1 Data exchange using tags

Basics of tags

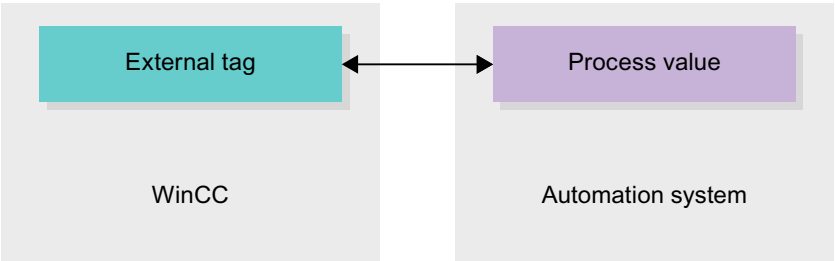
Introduction

Process values are forwarded in runtime using tags. Process values are data which is stored in the memory of one of the connected automation systems. They represent the status of a plant in the form of temperatures, fill levels or switching states, for example. Define external tags for processing the process values in WinCC.

WinCC works with two types of tag:

- External tags
- Internal tags

The external tags form the link between WinCC and the automation systems. The values of external tags correspond to the process values from the memory of an automation system. The value of an external tag is determined by reading the process value from the memory of the automation system. It is also possible to rewrite a process value in the memory of the automation system.



Internal tags do not have a process link and only convey values within the WinCC. The tag values are only available as long as runtime is running.

Tags in WinCC

For external tags, the properties of the tag are used to define the connection that the WinCC uses to communicate with the automation system and form of data exchange.

Tags that are not supplied with values by the process - the internal tags - are not connected to the automation system. In the tag's "Connection" property, this is identified by the "Internal tag" entry.

You can create tags in different tag tables for greater clarity. You then directly access the individual tag tables in the "HMI tags" node in the project tree. The tags from all tag tables can be displayed with the help of the table "Show all tags".

With structures you bundle a number of different tags that form one logical unit. Structures are project-associated data and are available for all HMI devices of the project. You use the "Types" editor in the project library to create and edit a structure.

Overview of HMI tag tables

Introduction

HMI tag tables contain the definitions of the HMI tags that apply across all devices. A tag table is created automatically for each HMI device created in the project.

In the project tree there is an "HMI tags" folder for each HMI device. The following tables can be contained in this folder:

- Default tag table
- User-defined tag tables
- Table of all tags

In the project tree you can create additional tag tables in the HMI tags folder and use these to sort and group tags and constants. You can move tags to a different tag table using a drag-and-drop operation or with the help of the "Tag table" field. Activate the "Tags table" field using the shortcut menu of the column headings.

Default tag table

There is one default tag table for each HMI device of the project. It cannot be deleted or moved. The default tag table contains HMI tags and, depending on the HMI device, also system tags. You can declare all HMI tags in the standard tags table or, as necessary, additional user-defined tables of tags.

User-defined tag tables

You can create multiple user-defined tag tables for each HMI device in order to group tags according to your requirements. You can rename, gather into groups, or delete user-defined tag tables. To group tag tables, create additional subfolders in the HMI tags folder.

All tags

The "All tags" table shows an overview of all HMI tags and system tags of the HMI device in question. This table cannot be deleted, renamed or moved. This table also contains the "Tags table" column, which indicates the tag table of where a tag is included. Using the "Tags table" field, the assignment of a tag to a tags table can be changed.

With devices for Runtime Professional, the table "All tags" contain an additional tab "System tags". The system tags are created by the system and used for internal management of the project. The names of the system tags begin with the "@" character. System tags cannot be deleted or renamed. You can evaluate the value of a system tag, but cannot modify it.

Additional tables

The following tables are also available in an HMI tag table:

- Discrete alarms
- Analog alarms
- Logging tags

With the help of these tables you configure alarms and logging tags for the currently selected HMI tag.

Discrete alarms table

In the "Discrete alarms" table, you configure discrete alarms to the HMI tag selected in the HMI tag table. When you configure a discrete alarm, multiple selection in the HMI tag table is not possible. You configure the discrete alarms for each HMI tag separately.

Analog alarms table

In the "Analog alarms" table, you configure analog alarms to the HMI tag selected in the HMI tag table. When you configure an analog alarm, multiple selection in the HMI tag table is not possible. You configure the analog alarms for each HMI tag separately.

Logging tags table

In the "Logging tags" table, you configure logging tags to the HMI tag selected in the HMI tag table. When you configure a logging tag, multiple selection in the HMI tag table is not possible. You configure the logging tags for each HMI tag separately. The "Logging tags" table is only available if the HMI device used supports logging.
If WinCC Runtime Professional is used, you can also assign several log tags to a tag. With the other HMI devices, you can only assign one log tag to a tag.

External tags

Introduction

External tags allow communication (data exchange) between the components of an automation system, for example, between an HMI device and a PLC.

Principle

An external tag is the image of a defined memory location in the PLC. You have read and write access to this storage location from both the HMI device and from the PLC.

Since external tags are the image of a storage location in the PLC, the applicable data types depend on the PLC which is connected to the HMI device.

If you write a PLC control program in STEP 7, the PLC tags created in the control program will be added to the PLC tag table. If you want to connect an external tag to a PLC tag, access the PLC tags directly via the PLC tag table and connect them to the external tag.

Data types

All the data types which are available at the connected PLC are available at an external tag in WinCC. Information about data types which are available for connection to other PLCs can be found in the documentation about the respective communication drivers.

See "Auto-Hotspot" for additional information.

Note

As well as external tags, area pointers are also available for communication between the HMI device and PLC. You can set up and enable the area indicators in the "Connections" editor.

Synchronization with PLC tags

A variety of options for synchronizing external tags with the PLC tags are available in the Runtime settings under "Settings for tags".

When you perform synchronization, you have the option of automatically applying the tag names of the PLC to external tags and reconnecting the existing tags.

The generated tag name is derived from the position of the data value in the hierarchical structure of the data block.

Update of tag values

For external tags, the current tag values are transmitted in runtime via the communication connection between WinCC and the connected automation systems and then saved in the runtime memory. Next, the tag value will be updated to the set cycle time. For use in the runtime project, WinCC accesses tag values in the runtime memory that were read from the PLC at the previous cycle time. As a result, the value in the PLC can already change whilst the value from the runtime memory is being processed.

Note

PLC array elements in conjunction with S7-1200 or S7-1500

The index of the PLC array elements can begin with any number. In WinCC, indexing always starts with 0.

A PLC tag "Array [1..3] of Int", for example, is mapped to "Array [0..2] of Int" in WinCC.

When you access an array in a script, pay attention to the correct indexing.

Addressing external tags

Introduction

The options for addressing external tags depend on the type of connection between WinCC and the PLC in question. A distinction must be made between the following connection types:

- Integrated connection
  Connections of devices which are within a project and were created with the "Devices & Networks" editor are referred to as integrated connections.
- Non-integrated connection
  Connections of devices which were created with the "Connections" editor are referred to as non-integrated connections. It is not necessary that all of the devices be within a single project.

The connection type can also be recognized by its icon.

	Integrated connection
	Non-integrated connection

You can find additional information in the section "Basics of communication (Page 8373)".

Addressing with integrated connections

An integrated connection offers the advantage that you can address a tag both symbolically and absolutely.

For symbolic addressing, you select the PLC tag via its name and connect it to the HMI tag. The valid data type for the HMI tag is automatically selected by the system. You have to distinguish between the following cases when you address elements in data blocks:

Symbolic addressing of data blocks with optimized access and standard access:

During the symbolic addressing of a data block with optimized access and standard access, the address of an element in the data block is dynamically assigned and is automatically adopted in the HMI tag in the event of a change. You do not need to compile the connected data block or the WinCC project for this step.
For data blocks with optimized access, only symbolic addressing is available.

For symbolic addressing of elements in a data block, you only need to recompile and reload the WinCC project in case of the following changes:

- If the name or the data type of the linked data block element or global PLC tag has changed.
- If the name or the data type of the higher level structure node of a linked element in the data block element or global PLC tag has changed.
- If the name of the connected data block has changed.

Symbolic addressing is currently available with the following PLCs:

- SIMATIC S7 1200
- SIMATIC S7 1500

Symbolic addressing is also available if you have an integrated link.

Note

Symbolic addressing is not possible if you are using the "Date/Time" area pointer.

You can also use absolute addressing with an integrated connection. You have to use absolute addressing for PLC tags from a SIMATIC S7 300/400 PLC. If you have connected an HMI tag with a PLC tag and the address of the PLC tag changes, you only have to recompile the control program to update the new address in WinCC. Then you recompile the WinCC project and load it onto the HMI device.

In WinCC, symbolic addressing is the default method. To change the default setting, select the menu command "Options > Settings". Select "Visualization > Tags" in the "Settings" dialog. If required, disable the "Symbolic access" option.

The availability of an integrated connection depends on the PLC used. The following table shows the availability:

PLC	Integrated connection	Comments
S7 300/400	Yes	The linking of tags is not checked in Runtime. If the tag address changes in the PLC and the HMI device is not compiled again and loaded, the change is not registered in runtime.
S7 1200	Yes	A validity check of the tag connection is performed in runtime during symbolic addressing. If an address is changed in the PLC, the change is registered and an error message is issued. In the case of absolute addressing, the following behavior applies to the S7 300/400.
S7 1500	Yes	A validity check of the tag connection is performed in runtime during symbolic addressing. If an address is changed in the PLC, the change is registered and an error message is issued. In the case of absolute addressing, the following behavior applies to the S7 300/400.

Create an integrated connection in the "Devices & Networks" editor. If the PLC is contained in the project and integrated connections are supported, you can then also have the connection created automatically. To do this, when configuring the HMI tag, simply select an existing PLC tag to which you want to connect the HMI tag. The integrated connection is then automatically created by the system.

Addressing with non-integrated connections

In the case of a project with a non-integrated connection, you always configure a tag connection with absolute addressing. Select the valid data type yourself. If the address of a PLC tag changes in a project with a non-integrated connection during the course of the project, you also have to make the change in WinCC. The tag connection cannot be checked for validity in Runtime, an error message is not issued.

A non-integrated connection is available for all supported PLCs.

Symbolic addressing is not available in a non-integrated connection.

With a non-integrated connection, the control program does not need to be part of the WinCC project. You can perform the configuration of the PLC and the WinCC project independently of each other. For configuration in WinCC, only the addresses used in the PLC and their function have to be known.

See also

Basics of communication (Page 8373)

Internal Tags

Introduction

Internal tags do not have any connection to the PLC. Internal tags transport values within the HMI device. The tag values are only available as long as runtime is running.

Principle

Internal tags are stored in the memory of the HMI device. Therefore, only this HMI device has read and write access to the internal tags. You can create internal tags to perform local calculations, for example.

You can use the HMI data types for internal tags. Availability depends on the HMI device being used.

The following HMI data types are available:

HMI data type	Data format
Array	One-dimensional array
Bool	Binary tag
DateTime	Date/time format
DInt	Signed 32-bit value
Int	Signed 16-bit value
LReal	Floating-point number 64-bit IEEE 754
Real	Floating-point number 32-bit IEEE 754
SInt	Signed 8-bit value
UDnt	Unsigned 32-bit value
UInt	Unsigned 16-bit value
USInt	Unsigned 8-bit value
WString	Text tag, 16-bit character set

10.8.4.2 Data exchange using area pointers

Basic information on area pointers

Introduction

You use an area pointer to access a data area in the PLC. During communication, the PLC and the HMI device alternately access these data areas for read and write operations. The PLC and the HMI device trigger defined interactions based on the evaluation of stored data.

For example, area pointers are required for the following data:

- Recipes
- Job mailboxes
- Sign-of-life monitoring

Area pointer

The following area pointers are supported:

Area pointer

Area pointers can be configured for connections.

- Data record
- Date/time
- Coordination
- Job mailbox

Global area pointers of the HMI device

Global area pointers can be configured to only one connection for each project.

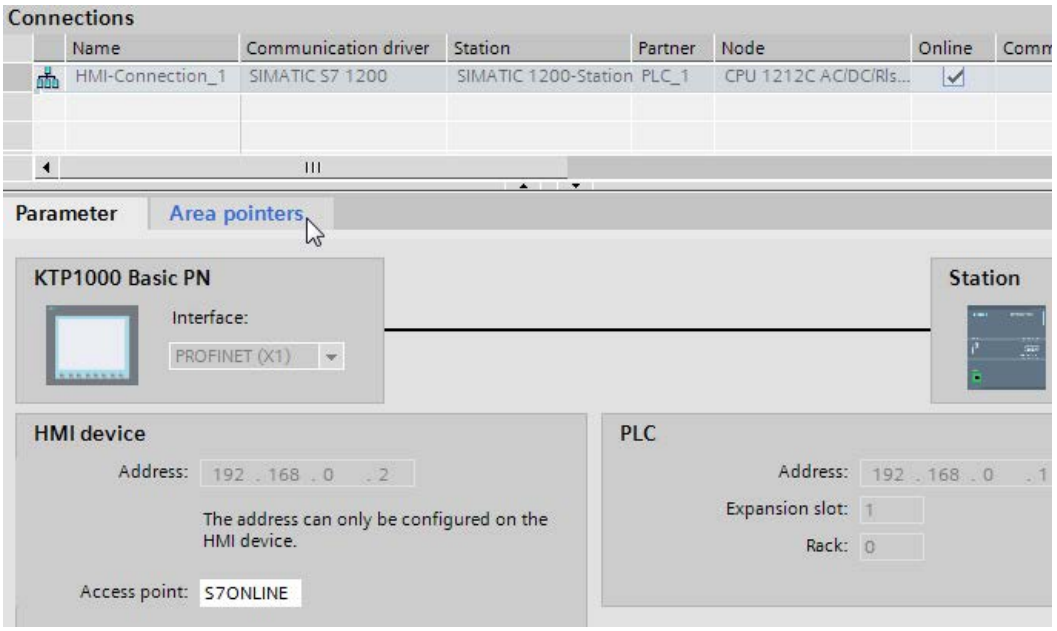
- Screen number
- Date/time PLC
- Project ID

Area pointers for connections

Introduction

Using the "Area pointer" tab of the "Connections" editor, you can configure the usage of the available area pointers.

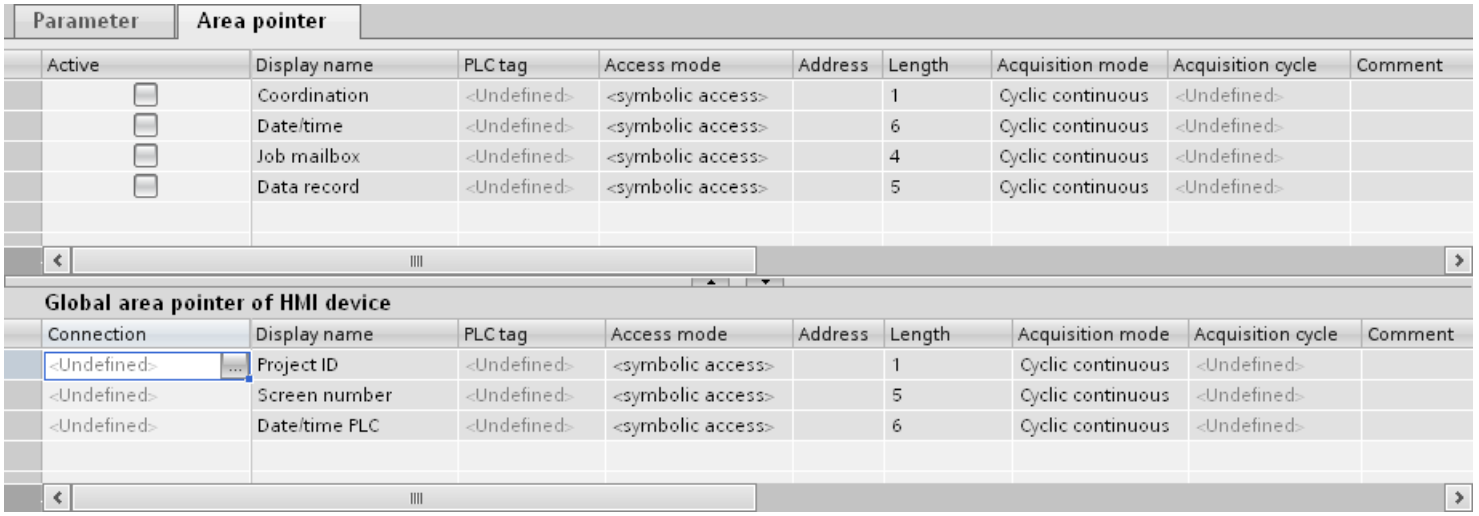
To configure the area pointers, open the "Connections" editor and open the "Area pointer" tab.



Structure

The "Area pointer" tab contains two tables of area pointers. The top part of the table contains the area pointers you can create and enable separately for each available connection.

The "Global area pointers of HMI device" table contains the area pointers which are created only once in the project and can be used for only one connection.



Use of area pointers

"Area pointer" tab

Before you use an area pointer, you enable it under "Connections > Area pointer". You then assign the area pointer parameters.

You assign the following parameters in the "Area pointer" tab:

Parameter | Area pointer

Active	Display name	PLC tag	Access mode	Address	Length	Acquisition mode	Acquisition cycle	Comment
	Coordination	<Undefined>	<symbolic access>		1	Cyclic continuous	<Undefined>	
	Date/time	<Undefined>	<symbolic access>		6	Cyclic continuous	<Undefined>	
	Job mailbox	<Undefined>	<symbolic access>		4	Cyclic continuous	<Undefined>	
	Data record	<Undefined>	<symbolic access>		5	Cyclic continuous	<Undefined>	

Global area pointer of HMI device

Connection	Display name	PLC tag	Access mode	Address	Length	Acquisition mode	Acquisition cycle	Commer
<Undefined>	Project ID	<Undefined>	<symbolic access>		1	Cyclic continuous	<Undefined>	
<Undefined>	Screen number	<Undefined>	<symbolic access>		5	Cyclic continuous	<Undefined>	
<Undefined>	Date/time PLC	<Undefined>	<symbolic access>		6	Cyclic continuous	<Undefined>	

- Active
  Enables the area pointer.
- Pointer name
  Name of the area pointer specified by WinCC.
- PLC tag
  Here you select the PLC tag or the tag array that you have configured as the data area for the area pointer.
- Access mode
  Here you can select from the following access modes:
  - Symbolic access
  - Absolute access
- Address
  If you selected "Symbolic access", no address is entered in this field.
  If you selected "Absolute access", enter the address of a tag in the "Address" field.
- Length
  WinCC specifies the length of the area pointer.
- Acquisition cycle
  You specify the acquisition cycle in this field for area pointers that are read by the HMI device. Note that a very short acquisition time may have a negative impact on HMI device performance.
- Comment
  Enter a comment, for example, to describe the purpose of the area pointer.

Accessing data areas

Accessing data areas

The following table shows how HMI devices and PLCs access individual data areas for read (R) or write (W) operations.

Data area	Required for	HMI device	PLC
Screen number	Evaluation by the PLC in order to determine the active screen.	W	R
Data record	Transfer of data records with synchronization	R/W	R/W
Date/time	Transfer of the date and time from the HMI device to the PLC	W	R
Date/time PLC	Transfer of the date and time from the PLC to the HMI device	R	W
Coordination	Requesting the HMI device status in the PLC program	W	R
Project ID	Runtime checks for consistency between the WinCC project ID and the project in the PLC	R	W
Job mailbox	Triggering of HMI device functions by the PLC program	R/W	R/W

Configuring area pointers

Configuration of area pointers

Introduction

You use an area pointer to access a data area in the PLC. The data area is stored in the PLC.

Prior to configuring area pointers

Before you use the area pointer, you must enable and parameterize it under "Connections > Area Pointer".

Global data block

To access the data area in the PLC, you have to create a global data block in the PLC program. The following example shows the use of a data block.

Length of area pointers

For area pointers with a length >= 1, you set up the data area as a tag array in a global data block or instance data block.

You also have the option to use a PLC tag for area pointers with a length = 1.

The configuration of the tags in a data block is dependent on the length of the area pointer you want to use. The unit for the length of an area pointer is a 16-bit word.

If, for example, you want to use an area pointer with a length of "5", you must create an array with 5 array elements of the data type UINT in the data block.

Alternative procedure

Alternatively, you can also use the absolute access mode to access area pointers. Absolute access mode only works on standard PLC data blocks.

Parameterizing a global data block

Introduction

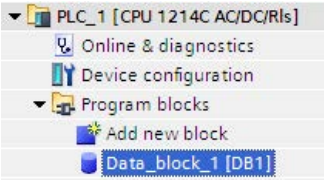
To access the data area in the PLC, a global data block for the area pointer must be parameterized in the PLC program.

Requirements

- A PLC is created in the project.
- A connection is configured between the PLC and the HMI device.
- The PLC program contains a global data block.

Procedure

1. Open "PLC > Program blocks" in the project tree.
2. Double-click the global data block you created previously.
   The data block opens.



3. Enter a tag name in the "Name" column.
4. Select "Array[lo .. hi] of type" as the data type in the "Data type" column.
5. Replace the "lo" entry by the low value for the dimension of the array.
6. Replace the "hi" entry by the high value for the dimension of the array.
   Example: If you configure an area pointer with the length "4", enter the value "0" for "lo" and the value "3" for "hi" inside the brackets.

7. Replace the "type" designation with the "word" data type.
   The full data type for an array of 4 tags appears as follows: "Array[0 .. 3] of word".
   The tag array is created after the entry is confirmed.
8. Click "Compile".
   The project is compiled.

Data_block_1

	Name	Data type	Default value	Initial value	Retain	Comment
1	Static					
2	Job_mailbox	Array [0 .. 3] of word				
3	Job_mailbox[0]	Word	W#16#0000	W#16#0000		
4	Job_mailbox[1]	Word	W#16#0000	W#16#0000		
5	Job_mailbox[2]	Word	W#16#0000	W#16#0000		
6	Job_mailbox[3]	Word	W#16#0000	W#16#0000		

Configuring an area pointer for a connection

Introduction

After you have parameterized the global data block, you now create the area pointer for the connection.

Requirements

- The global data block has been parameterized in the PLC program.

Procedure

1. Open "HMI >Connections" in the project tree.
2. Click the "Area pointer" tab.
3. Enable the required area pointer.
   You enable a global area pointer by selecting the connection in the "Connection" field.
4. Click the navigation button in the "PLC tag" field.
   The object list opens.

5. Navigate to the data block in the object list, and select the tag in the right window. You do not need an array tag to configure an area pointer with the length of "1".

Parameter | Area pointers

Active	Display name	PLC tag	Address	Length	Acquisition mode
☐	Data record			5	Cyclic continuous
☐	Date/time			6	Cyclic continuous
☐	Coordination			1	Cyclic continuous
☑	Job mailbox	Data_block_1.Job_mailbox ...	<symbolic access>	4	Cyclic continuous

Global area pointer of HMI device

Connection	Display name	PLC tag	Address	Length	Acquisition mode	Acquisition cycle
<Undefined> ...	Screen number			5	Cyclic continuous	<Undefined>
<Undefined>	Date/time PLC			6	Cyclic continuous	<Undefined>
<Undefined>	Project ID			1	Cyclic continuous	<Undefined>

6. Select the "Word" data type when creating the tag in the data block.

If required, set additional parameters, such as the acquisition cycle, during configuration.

Result

The area pointer is enabled and connected to the PLC tag in the global data block.

10.8.5 Device dependency

10.8.5.1 Basic Panel

Communication drivers

SIMATIC communication drivers

Device dependency of the Basic Panels

Various communication drivers are configured for Basic Panels depending on the communication.

The following tables show the communication drivers released for integrated and non-integrated communication.

There are different versions of the HMI devices within the integrated communication.

Communication drivers for integrated communication (V11.0)

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)	SIMATIC S7-1500	SIMATIC S7-300/400
KP300 Basic	Yes	Yes	Yes	Yes	Yes	No	Yes
KP400 Basic	Yes	Yes	Yes	Yes	Yes	No	Yes
KTP400 Basic PN	Yes	Yes	Yes	Yes	Yes	No	Yes
KTP600 Basic DP	Yes	Yes	Yes	Yes	Yes	No	Yes
KTP600 Basic PN	Yes	Yes	Yes	Yes	Yes	No	Yes
KTP1000 Basic DP	Yes	Yes	Yes	Yes	Yes	No	Yes
KTP1000 Basic PN	Yes	Yes	Yes	Yes	Yes	No	Yes
TP1500 Basic PN	Yes	Yes	Yes	Yes	Yes	No	Yes

Communication drivers for integrated communication (V12.0)

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)	SIMATIC S7-1500	SIMATIC S7-300/400
KP300 Basic	No	Yes	Yes	Yes	Yes	Yes	Yes
KP400 Basic	No	Yes	Yes	Yes	Yes	Yes	Yes
KTP400 Basic PN	No	Yes	Yes	Yes	Yes	Yes	Yes
KTP600 Basic DP	No	Yes	Yes	Yes	Yes	Yes	Yes
KTP600 Basic PN	No	Yes	Yes	Yes	Yes	Yes	Yes
KTP1000 Basic DP	No	Yes	Yes	Yes	Yes	Yes	Yes
KTP1000 Basic PN	No	Yes	Yes	Yes	Yes	Yes	Yes
TP1500 Basic PN	No	Yes	Yes	Yes	Yes	Yes	Yes

Communication drivers for integrated communication (V13.0)

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)	SIMATIC S7-1500	SIMATIC S7-300/400
KTP400 Basic	No	Yes	Yes	Yes	Yes	Yes	Yes
KTP700 Basic	No	Yes	Yes	Yes	Yes	Yes	Yes
KTP900 Basic	No	Yes	Yes	Yes	Yes	Yes	Yes
KTP1200 Basic	No	Yes	Yes	Yes	Yes	Yes	Yes

Communication drivers for non-integrated communication

HMI devices	SIMATIC S7-1200	SIMATIC S7-1500	SIMATIC S7-300/400	SIMATIC S7-200	SIMATIC LOGO!	SIMATIC HTTP protocol
KP300 Basic	Yes	Yes	Yes	Yes	Yes	No
KP400 Basic	Yes	Yes	Yes	Yes	Yes	No
KTP400 Basic PN	Yes	Yes	Yes	Yes	Yes	No
KTP600 Basic DP	Yes	Yes	Yes	Yes	Yes	No
KTP600 Basic PN	Yes	Yes	Yes	Yes	Yes	No
KTP700 Basic PN	Yes	Yes	Yes	Yes	Yes	No
KTP900 Basic PN	Yes	Yes	Yes	Yes	Yes	No
KTP1000 Basic DP	Yes	Yes	Yes	Yes	Yes	No
KTP1000 Basic PN	Yes	Yes	Yes	Yes	Yes	No
TP1500 Basic PN	Yes	Yes	Yes	Yes	Yes	No

Other communication drivers

Device dependency of the Basic Panels

The following table shows which communication drivers you can configure with the various Basic Panels.

Communication drivers

HMI devices	OPC	Allen-Bradley EtherNet/IP	Allen-Bradley DF1	Mitsubishi MC TCP/IP	Mitsubishi FX	Modicon Modbus TCP/IP	Modicon Modbus RTU	Omron Host Link
KP300 Basic	No	Yes	No	Yes	No	Yes	No	No
KP400 Basic	No	Yes	No	Yes	No	Yes	No	No
KTP400 Basic PN	No	Yes	No	Yes	No	Yes	No	No
KTP600 Basic DP	No	No	Yes[2)]	No	Yes	No	Yes[1)]	Yes
KTP600 Basic PN	No	Yes	No	Yes	No	Yes	No	No
KTP700 Basic PN	No	Yes	No	Yes	No	Yes	No	No
KTP900 Basic PN	No	Yes	No	Yes	No	Yes	No	No
KTP1000 Basic DP	No	No	Yes[2)]	No	Yes	No	Yes[1)]	Yes
KTP1000 Basic PN	No	Yes	No	Yes	No	Yes	No	No
TP1500 Basic PN	No	Yes	No	Yes	No	Yes	No	No

1) only with RS 422-RS232 converter
Order number: 6AV6 671-8XE00-0AX0

2) Direct communication with PLC 5 or KF2 module, otherwise only approved with RS422-RS232 converter (option).
Order number: 6AV6 671-8XE00-0AX0

Interfaces of the Basic Panels

Device dependency of the Basic Panels

The following table shows which HMI device interfaces are available for the communication driver protocols.

Table 10-11 Basic Panels

	KP300 Basic KP400 Basic KTP400 Basic PN KTP600 Basic PN KTP700 Basic PN KTP900 Basic PN KTP1000 Basic PN TP1500 Basic PN	KTP600 Basic DP KTP1000 Basic DP
SIMATIC S7 - PPI [1]	—	MPI/DP (X2)
SIMATIC S7 - MPI	—	MPI/DP (X2)
SIMATIC S7 - PROFIBUS	—	MPI/DP (X2)
SIMATIC S7 - PROFINET	PROFINET (X1)	—
SIMATIC HMI HTTP protocol	—	—
OPC	—	—
Allen-Bradley EtherNet/IP	PROFINET (X1)	—
Allen-Bradley DF1	—	MPI/DP (X2) [2]
Mitsubishi TCP/IP	PROFINET (X1)	—
Mitsubishi FX	—	MPI/DP (X2) (RS422)
Modicon Modbus TCP	PROFINET (X1)	—
Modicon Modbus RTU	—	MPI/DP (X2) [3]
Omron Host Link	—	MPI/DP (X2) (RS422)

[1] For SIMATIC S7-200 only

[2] Direct communication with PLC5 or KF2 module, otherwise only approved with RS422-RS232 converter (option).
Order number: 6AV6 671-8XE00-0AX0

[3] only approved with RS 422-RS232 converter
Order number: 6AV6 671-8XE00-0AX0

Area pointers for Basic Panels

Introduction

Area pointers are parameter fields from which the HMI device obtains information about the location and size of data areas in the PLC. During communication, the PLC and the HMI device alternately access these data areas for read and write operations. Based on the evaluation of data stored in the data areas, the PLC and HMI device initiate mutually defined actions.

WinCC uses the following area pointers:

- Job mailbox
- Project ID
- Screen number
- Data record
- Date/time
- Date/time PLC
- Coordination

Availability of the area pointers

The following table shows the availability of the area pointers on the HMI devices. Note that the area pointers can be used only for available communication drivers.

Area pointer

	KP300 Basic	KTP400 Basic PN	KTP600 Basic PN	KTP600 Basic DP	KTP700 Basic PN	KTP900 Basic PN	KTP1000 Basic PN	KTP1000 Basic DP	TP1500 Basic PN
Screen number	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Data record	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Date/time	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Date/time PLC	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Coordination	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Project ID	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Job mailbox	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

10.8.6 Communicating with SIMATIC S7 1500

10.8.6.1 Communication with SIMATIC S7 1500

Introduction

This section describes the communication between an HMI device and the SIMATIC S7 1500 PLC.

You can configure the following communication channels for the SIMATIC S7 1500 PLC:

- PROFINET
- PROFIBUS

HMI connection for communication

Configure connections between the HMI device and a SIMATIC S7 1500 in the "Devices & Networks" Editor.

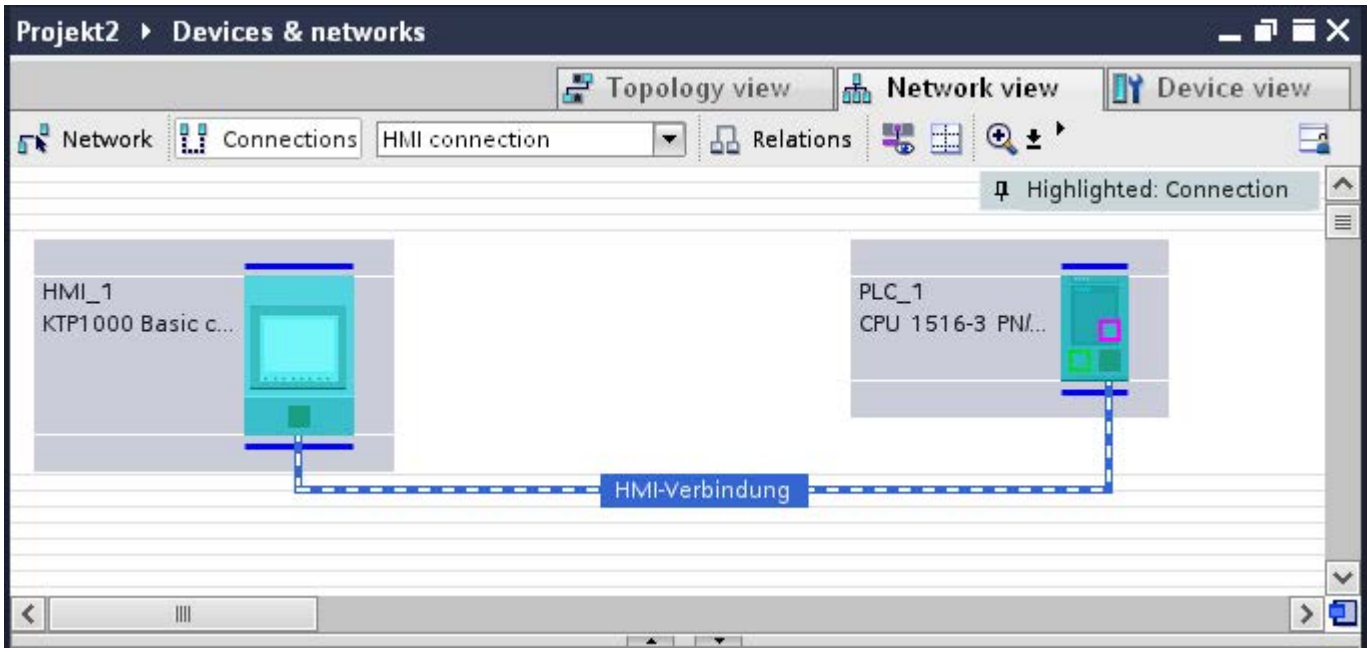
10.8.6.2 Communication via PROFINET

Configuring an HMI connection

Communication via PROFINET

HMI connections via PROFINET

If you have inserted an HMI device and a SIMATIC S7 1500 into the project, interconnect the two PROFINET interfaces in the "Devices & Networks" editor.



You can also connect multiple HMI devices to a single SIMATIC S7 1500 and multiple SIMATIC S7 1500 to a single HMI device.

The maximum number of communication partners that you can connect to an HMI device is dependent on the HMI device used.

Additional information is available in the documentation for the respective HMI device.

HMI connection in the "Devices & Networks" editor

You configure the HMI connection between the PLC and the HMI device via PROFINET in the "Devices & Networks" editor.

Connection in the "Connections" editor

Alternatively, you configure the connection between the PLC and HMI device via PROFINET in the "Connections" editor of the HMI device.

Configuring an HMI connection via PROFINET

Introduction

You configure an HMI connection between HMI devices and a SIMATIC S7 1500 via PROFINET or Ethernet in the "Devices & Networks" editor.

CAUTION
Communication via Ethernet In Ethernet-based communication, the end user is responsible for the security of his data network. Targeted attacks can overload the device and interfere with proper functioning.

Requirements

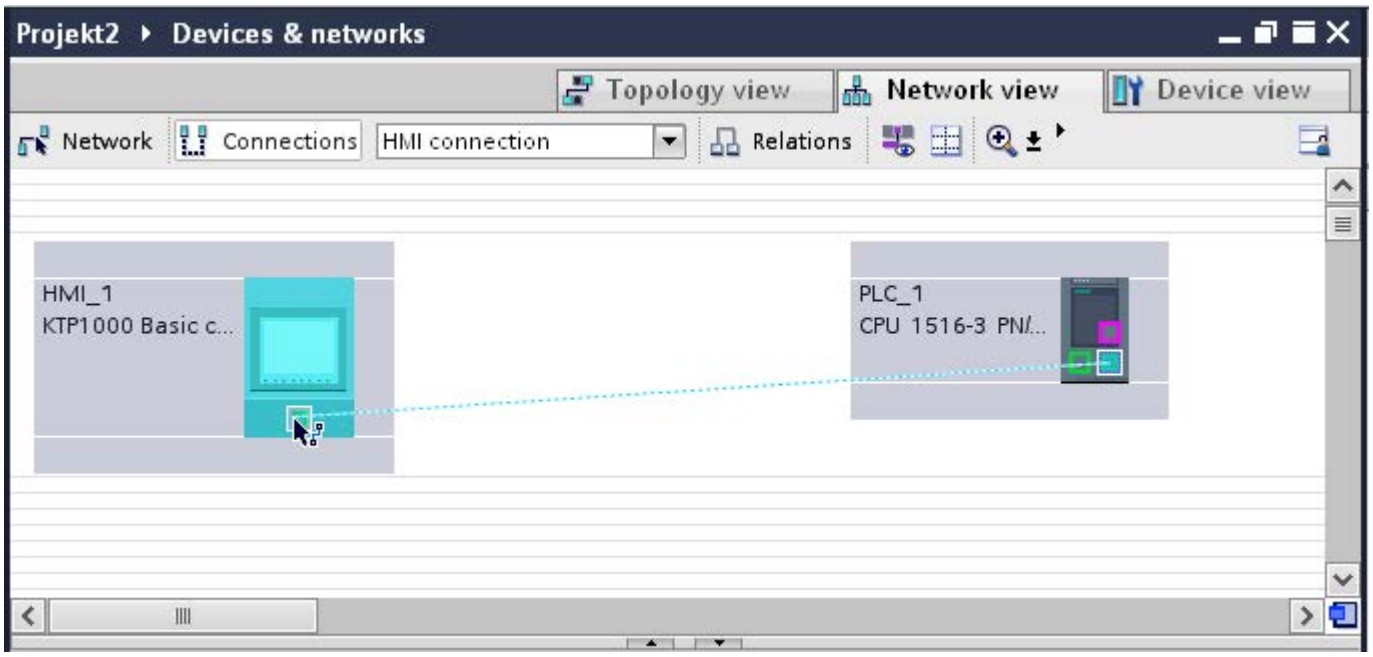
The following communication partners are created in the "Devices & Networks" editor:

- HMI device with PROFINET or Ethernet interface
- SIMATIC S7 1500 with PROFINET interface

Procedure

1. Double-click the "Devices & Networks" item in the project tree.
   The available communication partners in the project are displayed graphically in the network view.
2. Click the "Connections" button and select "HMI connection" for the connection type.
   The devices available for connection are highlighted in color.

3. Click the PROFINET interface of the PLC and use a drag-and-drop operation to draw a connection to the PROFINET or Ethernet interface of the HMI device.



4. Click the connecting line.
5. Click "Highlight HMI connection" and select the HMI connection.
   The connection is displayed graphically in the Inspector window.
6. Click the communication partners in the "Network view" and change the PROFINET parameters in the Inspector window according to the requirements of your project.
   See the chapter "Auto-Hotspot" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. You check the connection parameters in the table.

You can change the local name for the connection only in the table.

Result

You have created a connection between an HMI device and a SIMATIC S7 1500. The IP address and subnet mask connection parameters are configured.

PROFINET parameters

PROFINET parameters for the HMI connection

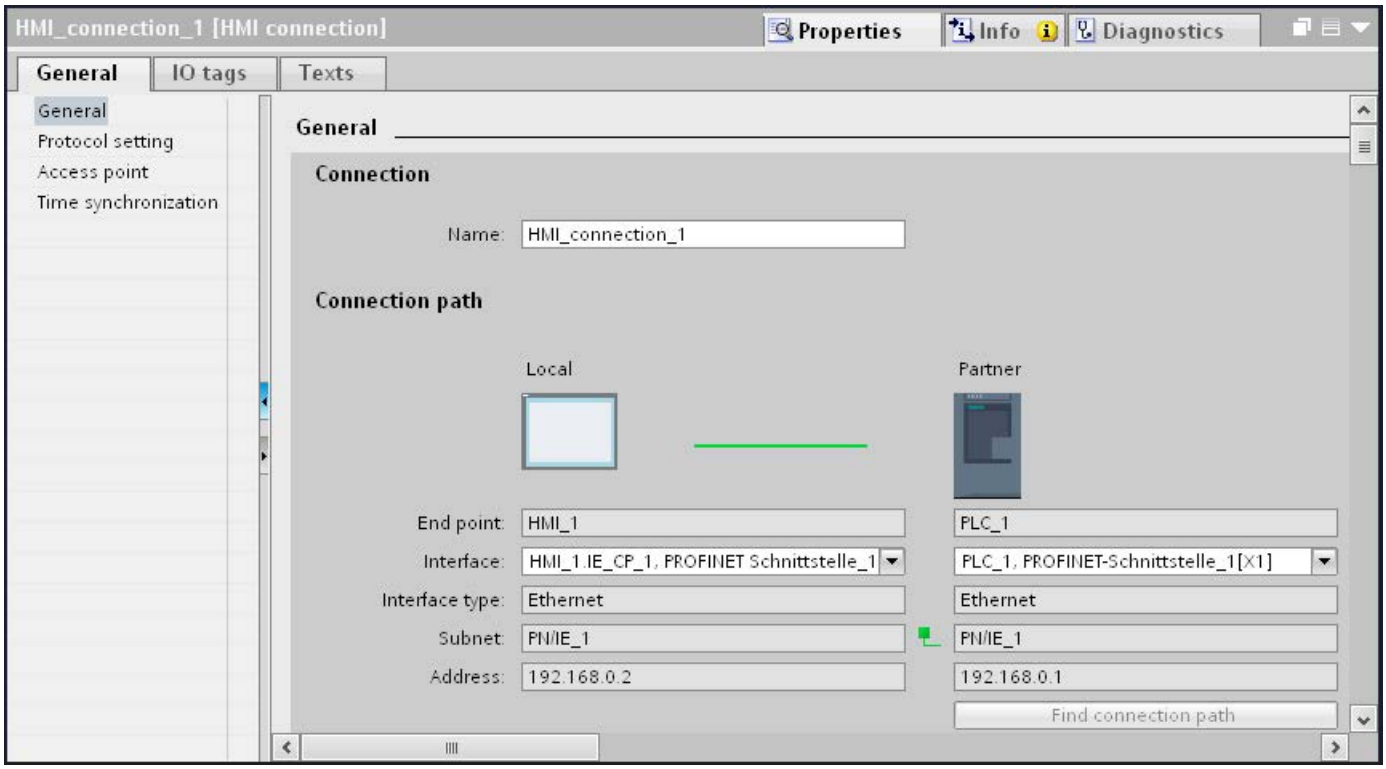
PROFINET parameters for the HMI connection

An overview of the configured HMI connection parameters can be found in the properties for the HMI connection.

Only limited changes are possible in this Inspector window.

Displaying and changing the HMI connection parameters

1. Click the HMI connection in the "Devices & Networks" editor.
2. Change the parameters of the HMI connection in the Inspector window under "Properties > General > General".



"Connection"

Shows the name of the HMI connection.

"Connection path"

The communication partners of the selected HMI connection and the associated PROFINET parameters are displayed in the "Connection path" area. Some of the areas displayed cannot be edited in this dialog.

- "End point"
  Displays the device name. This area cannot be edited.
- "Interface"
  Displays the selected interface of the device. You can choose between several interfaces, depending on the device.
- "Interface type"
  Displays the selected interface type. This area cannot be edited.
- "Subnet"
  Displays the selected subnet. This area cannot be edited.
- "Address"
  Displays the selected IP address of the device. This area cannot be edited.
- "Find connection path" button
  Enables the subsequent specification of connections.

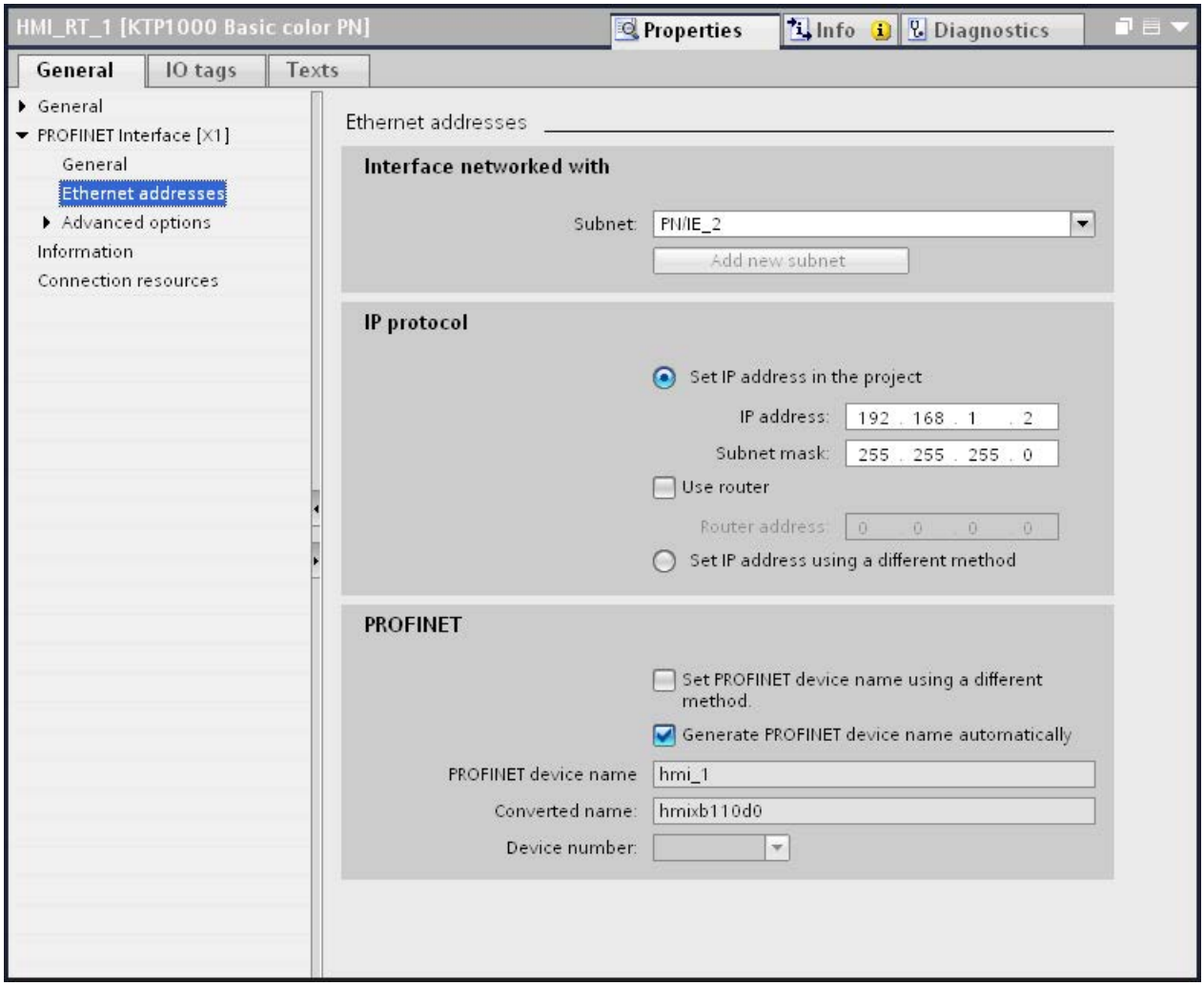
PROFINET parameters for the HMI device

PROFINET parameters for the HMI device

An overview of the configured HMI device parameters can be found in the properties for the HMI device.

Displaying and changing PROFINET parameters of the HMI device

1. Click the HMI device in the "Devices & Networks" editor.
2. Change the parameters of the HMI device in the Inspector window under "Properties > General".



"Interface networked with"

In the "Interface networked with" area, select the subnet of the HMI connection via which the HMI device is connected to the network. You use the "Add new subnet" button to create a new subnet.

"IP protocol"

- "Set IP address in the project"
  When you transfer the WinCC project to the HMI device, this IP address is set up directly in the HMI device.

  Note

  The device is automatically restarted in the case of HMI devices with the Windows CE 3.0 operating system.

  HMI devices with Windows CE 3.0:
  - OP 77B
  - TP 177B color PN/DP
  - TP 177B mono DP
  - OP 177B color PN/DP
  - OP 177B mono DP
  - Mobile Panel 177 PN
  - Mobile Panel 177 DP
  - TP 277 6"
  - OP 277 6"

- "Subnet mask"
  You assign data of the subnet mask in the "Subnet mask" area.
- "Use IP router"
  If you are using an IP router, select "Use IP router" and enter the router address in the "Router address" field.
- "Set IP address using a different method"
  If the function "Set IP address using a different method" is activated, the IP address is not taken from the project. You have to enter the IP address directly in the Control Panel of the HMI device.

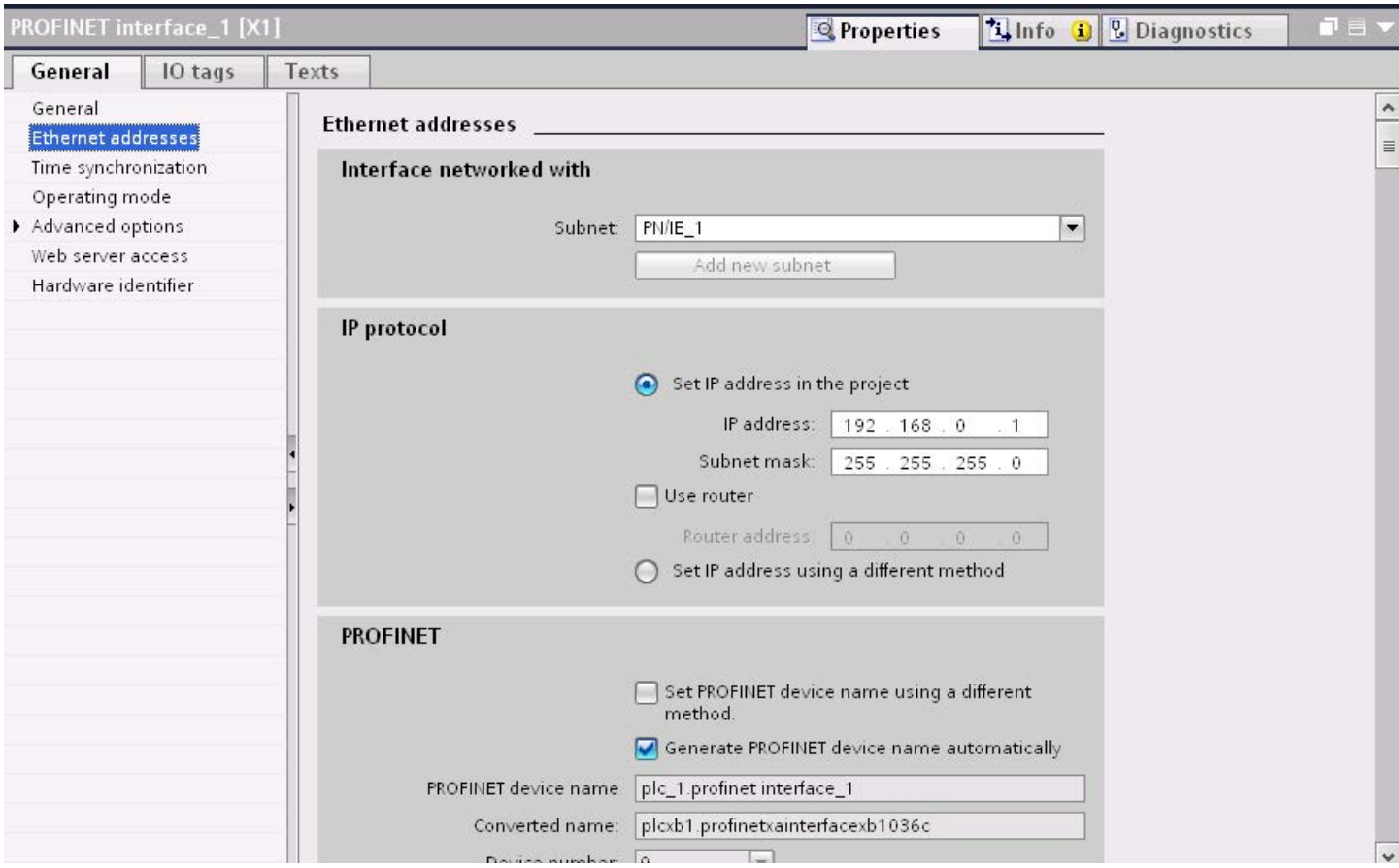
PROFINET parameters for the PLC

PROFINET parameters for the PLC

An overview of the configured parameters can be found in the properties for the PLC.

Displaying and changing PROFINET parameters of the PLC

1. Click the PLC in the "Devices & Networks" editor.
2. Change the parameters of the PLC in the Inspector window under "Properties > General > General".



"Interface networked with"

In the "Subnet" area, select the subnet of the HMI connection via which the PLC is connected to the network. You use the "Add new subnet" button to create a new subnet.

"IP protocol"

- "Interface type"
  Depending on the HMI device type, you have various interfaces to choose from.
- "IP address"
  You assign the IP address of the HMI device in the "IP address" area.
- "Subnet mask"
  You assign data of the subnet mask in the "Subnet mask" area.
  If you are using an IP router, select "Use IP router" and enter the router address in the field.

Protection of communication

Security levels

If you want to protect the PLC and HMI device communication, you can assign protection levels for the communication.

For a SIMATIC S7-1500 CPU, you can enter multiple passwords and thereby set up different access rights for various user groups.

The passwords are entered in a table, so that exactly one protection level is assigned to each password.

The effect of the password is given in the "Protection" column.

For the SIMATIC S7-1200 controller, several aspects need to be considered when setting protection levels. For additional information on this, see: S7-1200 protection levels

Example

You select the "Complete protection" protection level for a standard CPU (i.e., not an F-CPU) when configuring it.

Afterwards, you enter a separate password for every protection level above it in the table.

For users who do not know any of the passwords, the CPU is completely protected. Not even HMI access is possible.

For users who know one of the assigned passwords, the effect depends on the table row in which the password occurs:

- The password in row 1 (no protection) allows access as if the CPU were completely unprotected. Users who know this password have unrestricted access to the CPU.
- The password in row 2 (write protection) allows access as if the CPU were write-protected. Despite knowing the password, users who know this password only have read access to the CPU.
- The password in row 3 (read and write protection) allows access as if the CPU were read-protected and write-protected, so that only HMI access is possible for users who know this password.

Access password for the HMI connection

Introduction

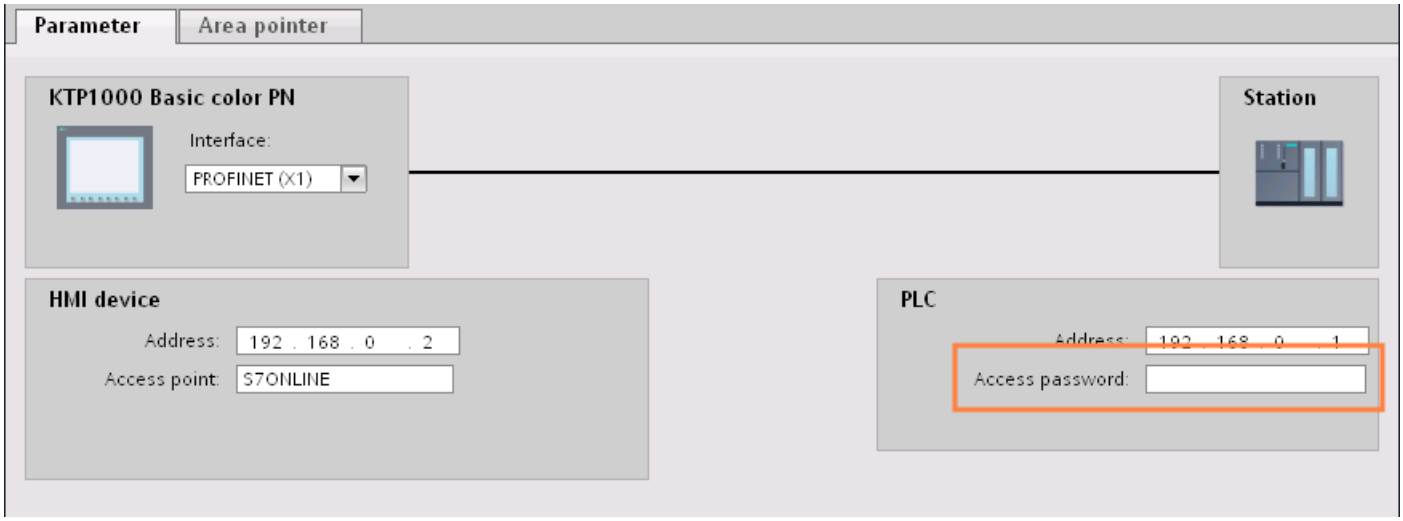
You secure access to a PLC by assigning a password.

You assign the password when you configure the password.

A password is mandatory as of protection level "Complete protection". Communication to the PLC is denied if an incorrect password or no password is entered.

Entering access password

You enter the password for the PLC in the "Connections" editor.



Setting port options

Setting the port options

Changing connection settings for the PROFINET IO port

You can change the network settings for the PROFINET IO port as required. By default, the settings are made automatically. In normal situations, this guarantees problem-free communication.

Possible settings for transmission rate / duplex

Depending on the selected device, you can make the following settings for "Transmission rate / duplex":

- Automatic setting
  Recommended default setting of the port. The transmission settings are automatically "negotiated" with the peer port. The "Enable autonegotiation" option is also enabled as a default, in other words, you can use cross cables or patch cables for the connection.
- TP/ITP at x Mbps full duplex (half duplex)
  Setting of the transmission rate and the full duplex/half duplex mode. The effectiveness depends on the "Enable autonegotiation" setting:
  - Autonegotiation enabled
    You can use both cross cable and patch cable.
  - Autonegotiation disabled
    Make sure that you use the correct cable (patch cable or cross cable)! The port is also monitored with this setting.
- Deactivated
  Depending on the module type, the drop down list box can contain the "- Disabled -" option. This option, for example, allows you to prevent access to an unused port for security reasons. With this setting, diagnostic events are not generated.

"Monitor" option

This option enables or disables port diagnostics. Examples of port diagnostics: The link status is monitored, in other words, the diagnostics are generated during link-down and the system reserve is monitored in the case of fiber optic ports.

Option "Enable autonegotiation "

The autonegotiation setting can only be changed if a concrete medium (for example, TP 100 Mbps full duplex) is selected. Whether or not a concrete medium can be set depends on the properties of the module.

If autonegotiation is disabled, this causes the port to be permanently specified, as for example, is necessary for a prioritized startup of the IO device.

You must make sure the partner port has the same settings because with this option the operating parameters of the connected network are not detected and the data transmission rate and transmission mode can accordingly not be optimally set.

Note

When a local port is connected, STEP 7 makes the setting for the partner port if the partner port supports the setting. If the partner port does not accept the setting, an error message is generated.

Wiring rules for disabled autonegotiation

Requirements

You have made the following settings for the port in question, for example, to accelerate the startup time of the IO device:

- Fixed transmission rate
- Autonegotiation incl. autocrossing disabled

The time for negotiating the transmission rate during startup has been saved.

If you have disabled autonegotiation, you must observe the wiring rules.

Wiring rules for disabled autonegotiation

PROFINET devices have the following two types of ports:

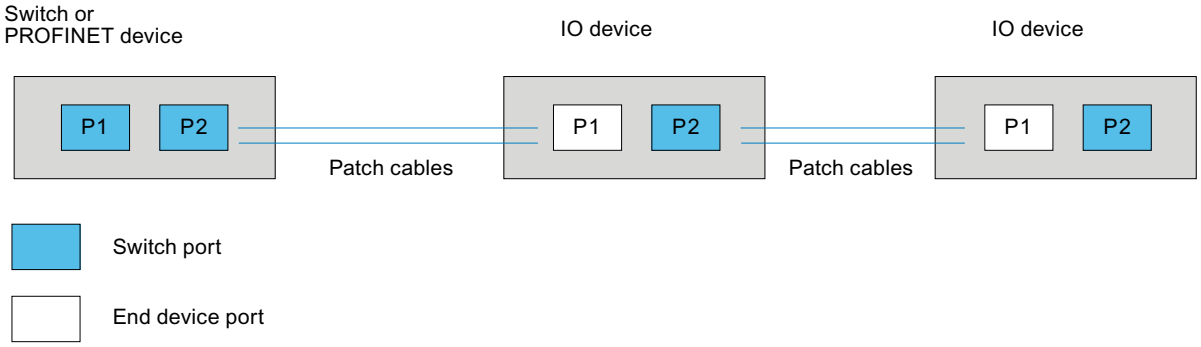
Type of port	PROFINET devices	Note
Switch port with crossed pin assignment	For IO devices: Port 2 For S7 CPUs with 2 ports: Ports 1 and 2	Crossed pin assignment means that the pin assignment for the ports for sending and receiving between the respective PROFINET devices is exchanged internally.
End device port with uncrossed pin assignment	For IO devices: Port 1 For S7 CPUs with one port: Port 1	-

Validity of the wiring rules

The cabling rules described in the following paragraph apply exclusively for the situation in which you have specified a fixed port setting.

Rules for cabling

You can connect several IO devices in line using a single cable type (patch cable). To do this, you connect port 2 of the IO device (distributed I/O) with port 1 of the next IO device. The following graphic gives an example with two IO devices.



Boundaries at the port

Requirements

To use boundaries, the respective device must have more than one port. If the PROFINET does not support boundary settings, they are not shown.

Enable boundaries

"Boundaries" are limits for transmission of certain Ethernet frames. The following boundaries can be set at a port:

- "End of discovery of accessible devices"
  No forwarding of DCP frames to identify accessible devices. Devices downstream from this port cannot be reached by the project tree under "Accessible devices". Devices downstream from this port cannot be reached by the CPU.
- "End of topology discovery"
  LLDP frames (Link Layer Discovery Protocol) are not forwarded for topology detection.
- "End of sync domain"
  No forwarding of sync frames transmitted to synchronize nodes within a sync domain.
  If you operate, for example, a PROFINET device with more than two ports in a ring, you should prevent the sync frame from being fed into the ring by setting a sync boundary (at the ports not inside the ring).
  Additional example: If you want to use several sync domains, configure a sync domain boundary for the port connected to a PROFINET device from the other sync domain.

Restrictions

The following restrictions must be observed:

- The individual check boxes can only be used if the port supports the function in question.
- If a partner port has been determined for the port, the following check boxes cannot be used:
  - "End of discovery of accessible devices"
  - "End of topology discovery"
- If autonegotiation is disabled, none of the check boxes can be used.

10.8.6.3 Communication via PROFIBUS

Configuring an HMI connection

Configuring an HMI connection via PROFIBUS

Introduction

You configure an HMI connection over PROFIBUS between HMI devices and a SIMATIC S7 1500 in the "Devices & Networks" editor.

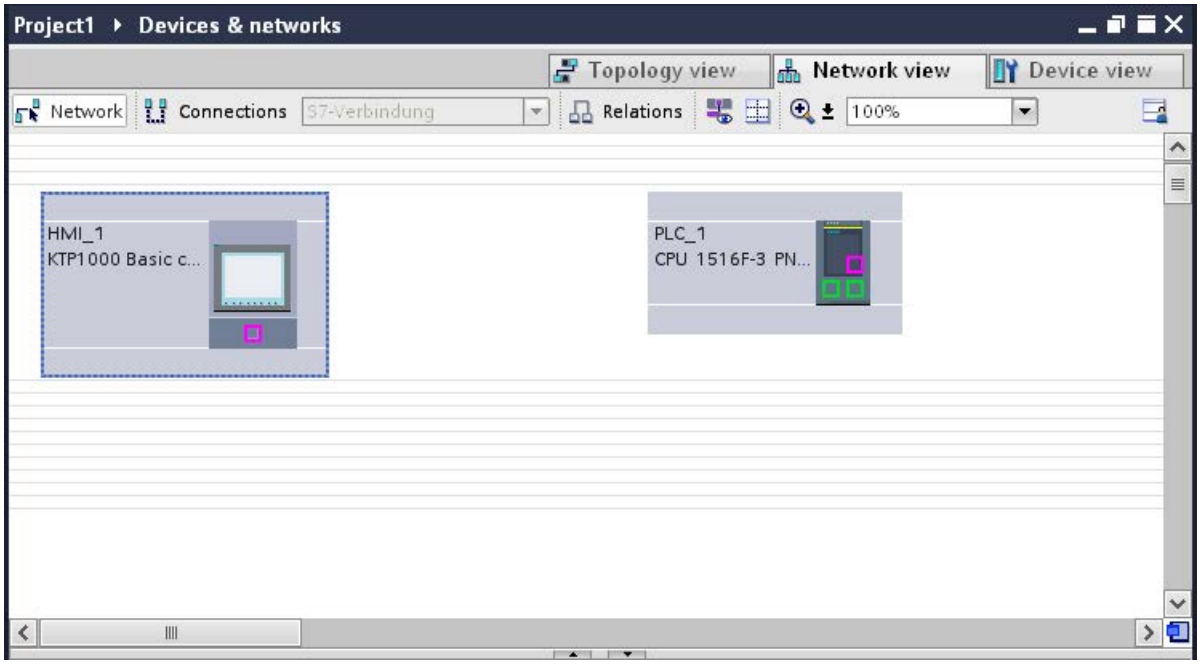
Requirements

The following communication partners are created in the "Devices & Networks" editor:

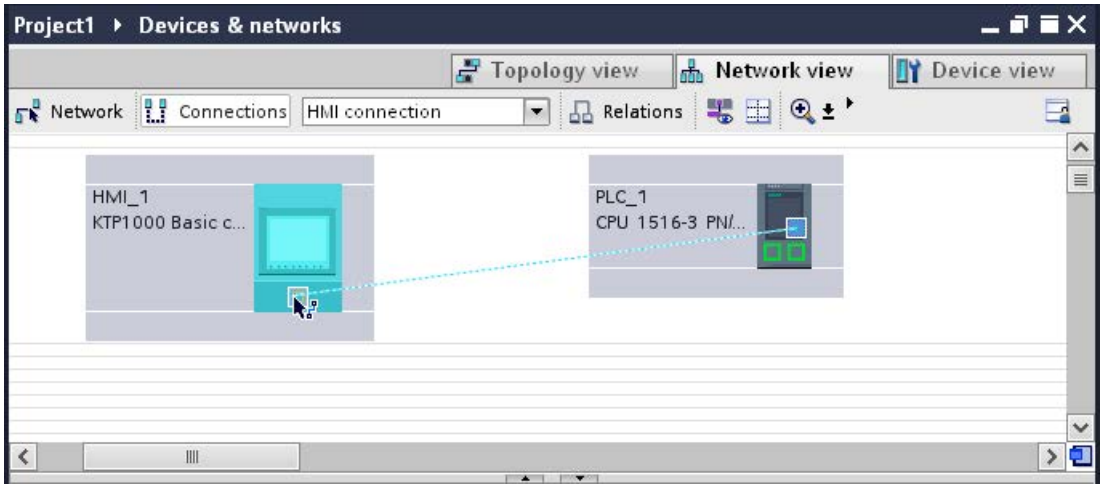
- HMI device with MPI/DP interface
- SIMATIC S7 1500 with PROFIBUS interface

Procedure

1. Double-click the "Devices & Networks" item in the project tree.
   The available communication partners in the project are displayed graphically in the network view.



2. Click the "Connections" button.
   The devices available for connection are highlighted in color.
3. Click the HMI device interface.
4. Select the "PROFIBUS" interface type in the Inspector window under "Properties > General > HMI MPIDP > Parameters".
5. Click the interface of the PLC and use a drag-and-drop operation to draw a connection to the HMI device.



6. Click the connecting line.

7. Click "Highlight HMI connection" and select the HMI connection.
   The connection is displayed graphically in the Inspector window.
8. Click the communication partners in the "Network view" and change the PROFINET parameters in the Inspector window according to the requirements of your project.
   See the chapter "Auto-Hotspot" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. You check the connection parameters in the table.

You can change the local name for the connection only in the table.

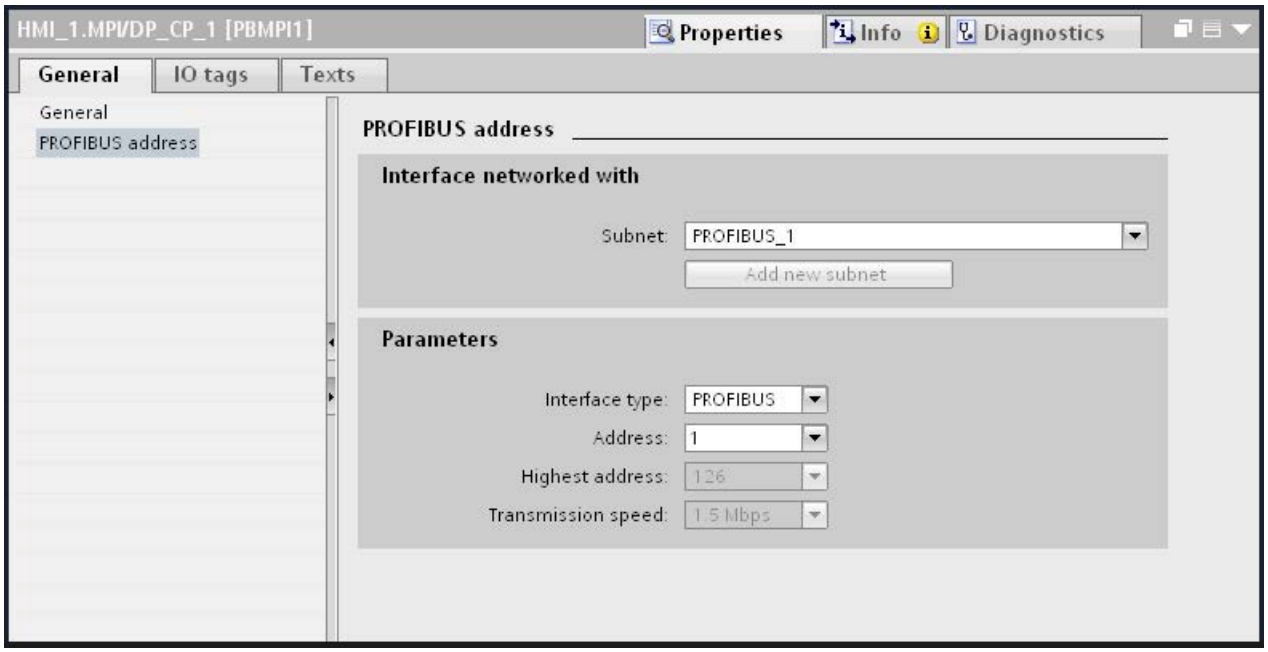
Result

You have created an HMI connection over PROFIBUS between an HMI device and a SIMATIC S7 1500.

Communication via PROFIBUS

HMI connections via PROFIBUS

If you have inserted an HMI device and a SIMATIC S7 1500 into the project, interconnect the two PROFIBUS interfaces in the "Devices & Networks" editor.



HMI connection in the "Devices & Networks" editor

You configure the HMI connection between the PLC and the HMI device via PROFIBUS in the "Devices & Networks" editor.

Connection in the "Connections" editor

Alternatively, you configure the connection between the PLC and HMI device in the "Connections" editor of the HMI device.

PROFIBUS parameters

PROFIBUS parameters for the HMI connection

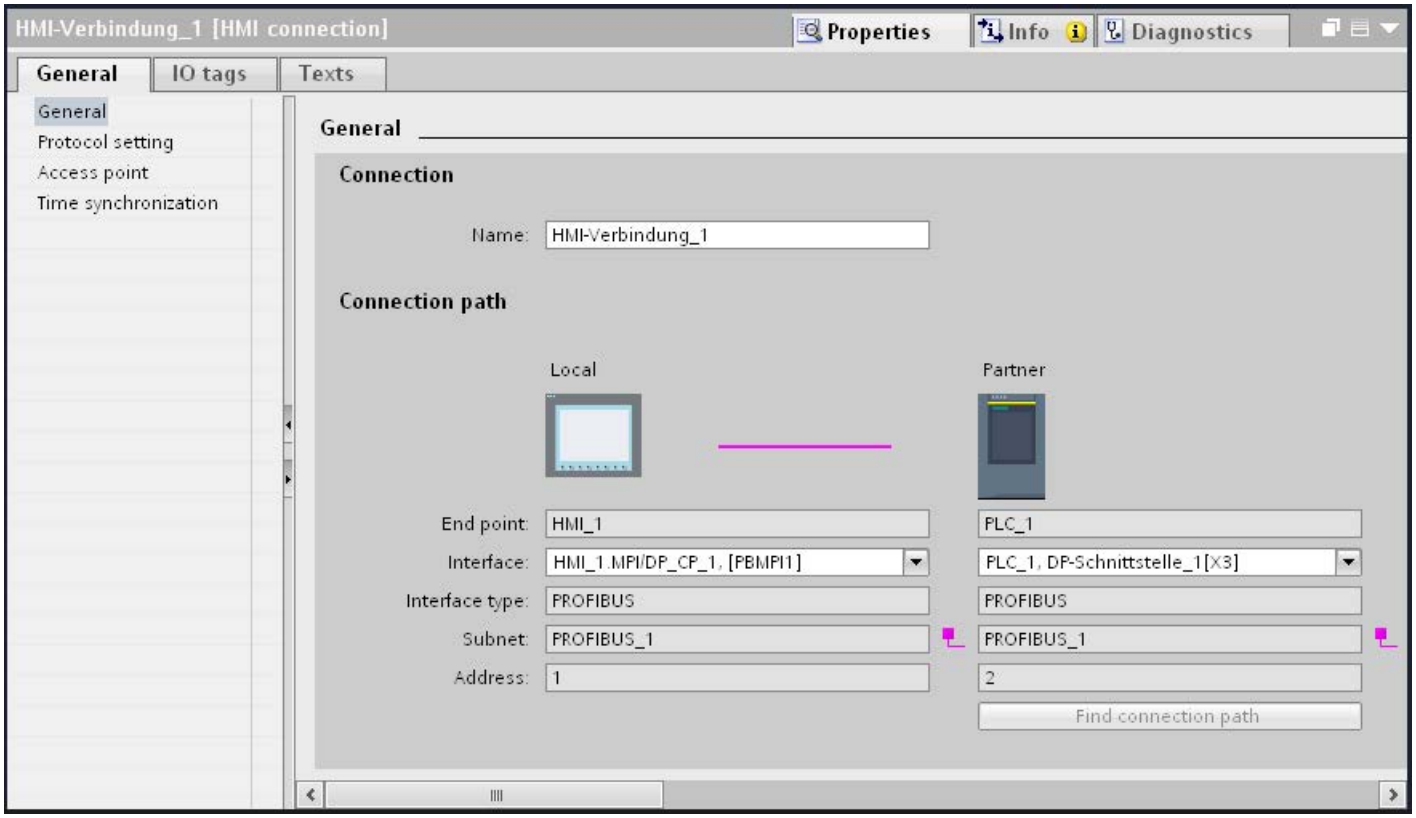
PROFIBUS parameters for the HMI connection

An overview of the configured HMI connection parameters can be found in the properties for the HMI connection.

Only limited changes are possible in this Inspector window.

Displaying and changing the HMI connection parameters

1. Click the HMI connection in the "Devices & Networks" editor.
2. Change the parameters of the HMI connection in the Inspector window under "Properties > General > General".



"Connection"

Displays whether the devices are networked together.

- displayed if the devices are networked together.

- displayed if the devices are not networked together.

"Connection path"

The communication partners of the selected HMI connection and the associated PROFIBUS parameters are displayed in the "Connection path" area. Some of the areas displayed cannot be edited in this dialog.

- "End point"
  Displays the device name. This area cannot be edited.
- "Interface"
  Displays the selected interface of the device. You can choose between several interfaces, depending on the device.

- "Interface type"
  Displays the selected interface type. This area cannot be edited.
- "Subnet"
  Displays the selected subnet. This area cannot be edited.
- "Address"
  Displays the PROFIBUS address of the device. This area cannot be edited.
- "Find connection path" button
  Enables the subsequent specification of connections.

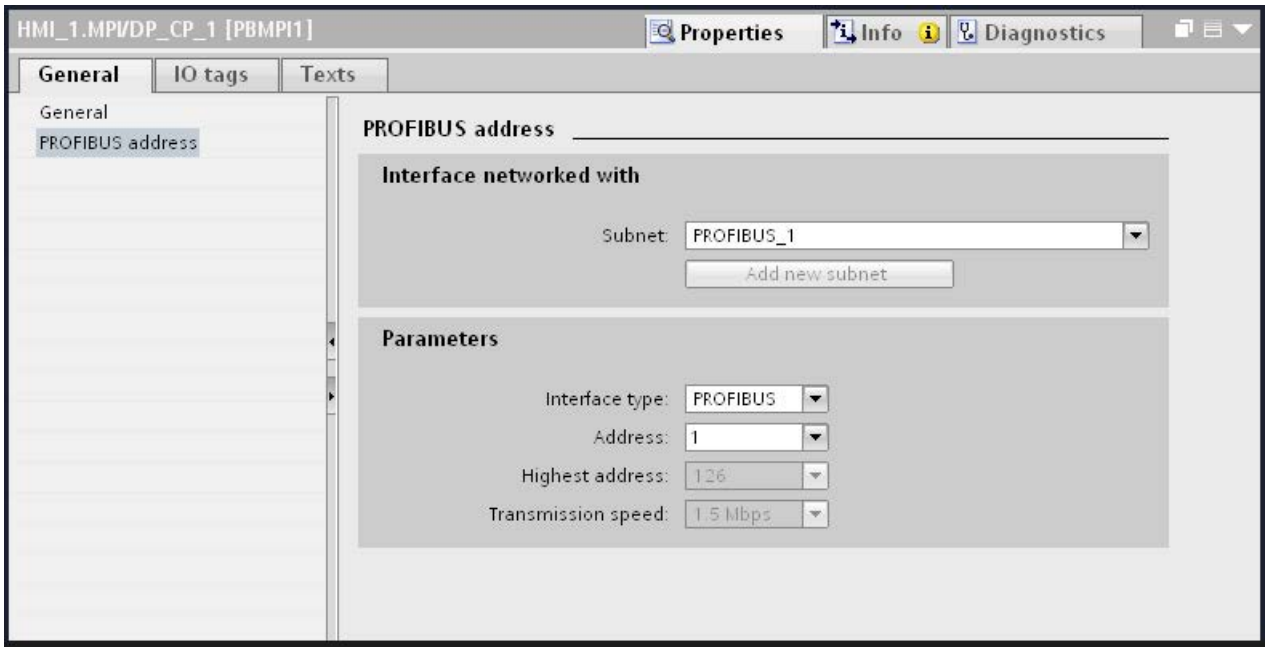
PROFIBUS parameters for the HMI device

PROFIBUS parameters for the HMI device

An overview of the configured HMI device parameters can be found in the properties for the HMI device.

Displaying and changing PROFINET parameters of the HMI device

1. Click the HMI device in the "Devices & Networks" editor.
2. Change the parameters of the HMI device in the Inspector window under "Properties > General > General".



"Interface networked with"

In the "Interface networked with" area, select the subnet of the HMI connection via which the HMI device is connected to the network. You use the "Add new subnet" button to create a new subnet.

"Parameters"

- "Interface type"
  You assign the interface type in the "Interface type" area. Depending on the HMI device type, you have various interfaces to choose from.
- "Address"
  You assign the PROFIBUS address of the HMI device in the "Address" area. The PROFIBUS address must be unique throughout the PROFIBUS network.
- "Highest address"
  The "Highest address" area displays the highest address of the PROFIBUS network.
- "Transmission speed"
  The "Transmission speed" is determined by the slowest device connected to the network. The setting is identical throughout the network.

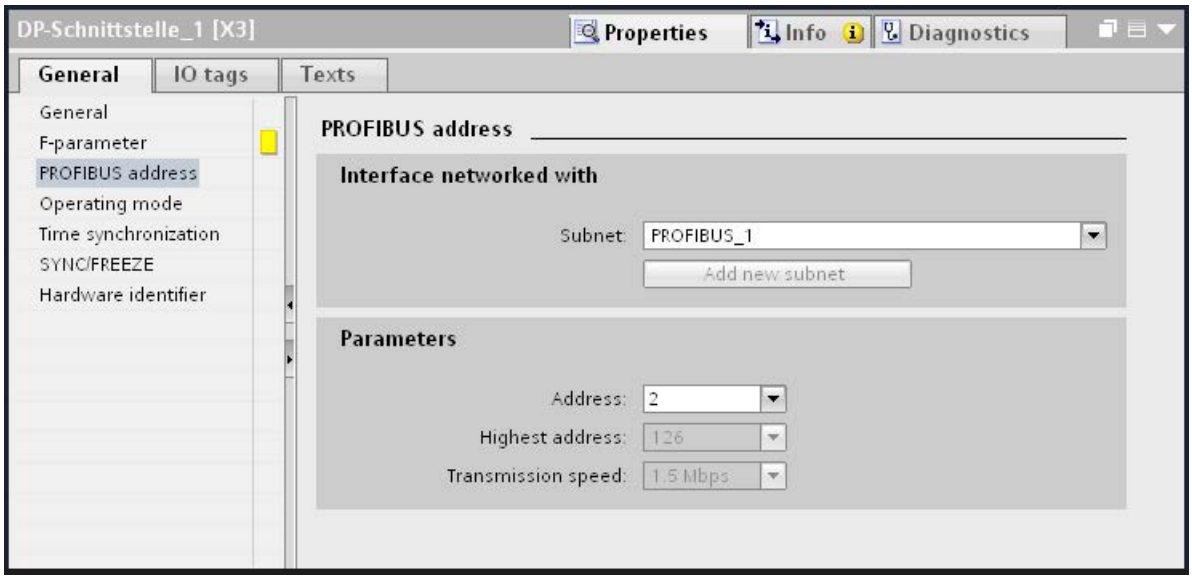
PROFIBUS parameters for the PLC

PROFIBUS parameters for the PLC

An overview of the configured parameters can be found in the properties for the PLC.

Displaying and changing PROFIBUS parameters of the PLC

1. Click the PLC in the "Devices & Networks" editor.
2. Change the parameters of the PLC in the Inspector window under "Properties > General > General".



"Interface networked with"

In the "Subnet" area, select the subnet of the HMI connection via which the PLC is connected to the network. You use the "Add new subnet" button to create a new subnet.

"Parameters"

- "Interface type"
  Depending on the HMI device type, you have various interfaces to choose from.
- "Address"
  You assign the PROFIBUS address of the HMI device in the "Address" area. The PROFIBUS address must be unique throughout the PROFIBUS network.
- "Highest address"
  The "Highest address" area displays the highest address of the PROFIBUS network.
- "Transmission speed"
  The "Transmission speed" is determined by the slowest device connected to the network. The setting is identical throughout the network.

Protection of communication

Access password for the HMI connection

Introduction

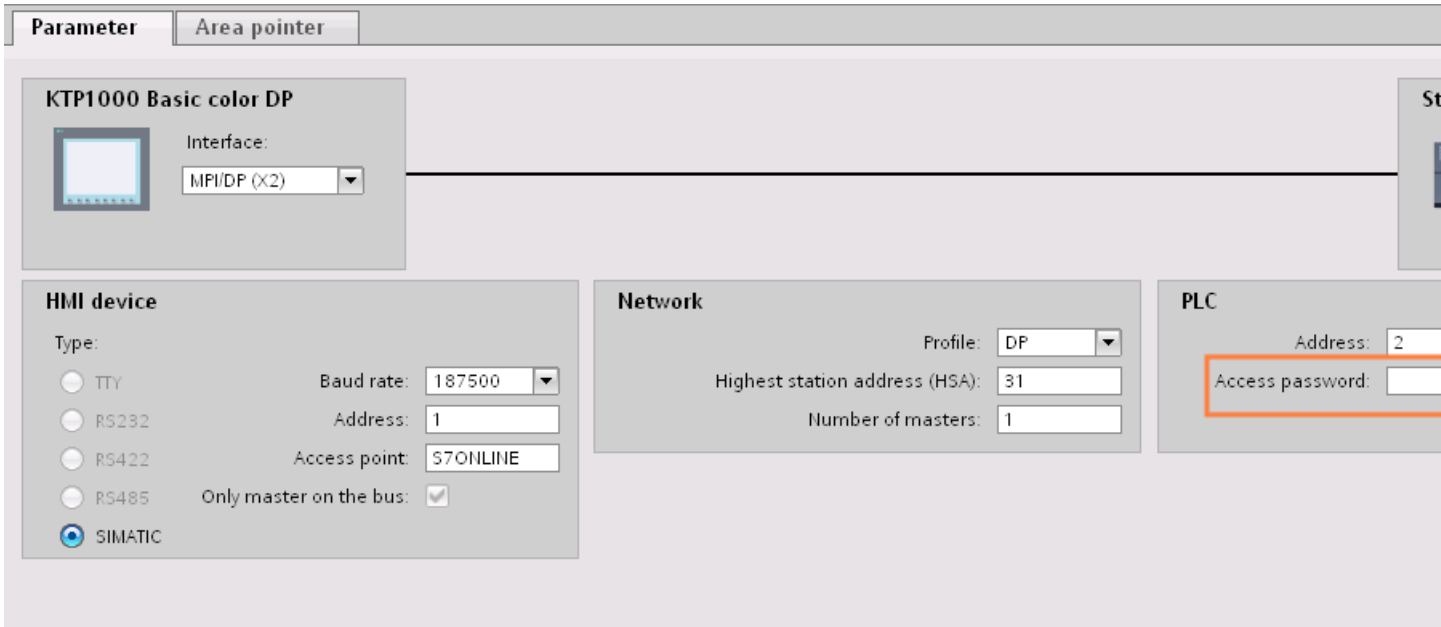
You secure access to a PLC by assigning a password.

You assign the password when you configure the password.

A password is mandatory as of protection level "Complete protection". Communication cannot be established with the PLC if you enter no password or an incorrect password.

Assigning password

You enter the password for the PLC in the "Connections" editor.



Security levels

If you want to protect the PLC and HMI device communication, you can assign protection levels for the communication.

For a SIMATIC S7-1500 CPU, you can enter multiple passwords and thereby set up different access rights for various user groups.

The passwords are entered in a table, so that exactly one protection level is assigned to each password.

The effect of the password is given in the "Protection" column.

For the SIMATIC S7-1200 controller, several aspects need to be considered when setting protection levels. For additional information on this, see: Auto-Hotspot

Example

You select the "Complete protection" protection level for a standard CPU (i.e., not an F-CPU) when configuring it.

Afterwards, you enter a separate password for every protection level above it in the table.

For users who do not know any of the passwords, the CPU is completely protected. Not even HMI access is possible.

For users who know one of the assigned passwords, the effect depends on the table row in which the password occurs:

- The password in row 1 (no protection) allows access as if the CPU were completely unprotected. Users who know this password have unrestricted access to the CPU.
- The password in row 2 (write protection) allows access as if the CPU were write-protected. Despite knowing the password, users who know this password only have read access to the CPU.
- The password in row 3 (read and write protection) allows access as if the CPU were read-protected and write-protected, so that only HMI access is possible for users who know this password.

10.8.6.4 Data exchange

Area pointers

General information on area pointers

Introduction

You use an area pointer to access a data area in the PLC. During communication, the PLC and the HMI device alternately access these data areas for read and write operations.

The PLC and the HMI device trigger defined interactions based on the evaluation of stored data.

Configuration of area pointers

Before you use an area pointer, you enable it under "Connections > Area pointer". You then assign the area pointer parameters.

You can find more detailed information on configuring area pointers in:

Data exchange using area pointers (Page 8417)

Restrictions

You can only configure the following data types for communication with SIMATIC S7 1500 for data exchange using area pointers:

- UInt and array of UInt
- Word and array of Word
- Int and array of Int
- "Array[0..15] of Bool" for area pointer "Coordination"
- Date_And_Time
- DTL and LDT

"Screen number" area pointer

Function

The HMI device saves information about the screen called on the HMI device to the "Screen number" area pointer.

This allows the transfer of the current screen contents from the HMI device to the PLC. The PLC can trigger specific reactions such as the call of a different screen.

Use

Configure and enable the area pointer in "Communication > Connections" before you put it into use. You can create only **one** instance of the "Screen number" area pointer and only on **one** PLC.

The screen number is always transferred to the PLC when a new screen is activated or when the focus within a screen changes from one screen object to another.

Structure

The area pointer is a data area in the memory of the PLC with a fixed length of 5 words.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1st word	Current screen type															
2nd word	Current screen number															
3rd word	Reserved															
4th word	Current field number															
5th word	Reserved															

- Current screen type
  "1" for root screen or
  "4" for permanent window
- Current screen number
  1 to 32767
- Current field number
  1 to 32767

"Date/time" area pointer

Function

This area pointer is used to transfer the date and time from the HMI device to the PLC.

The PLC writes control job "41" or "40" to the job mailbox.

When it evaluates the control job, the HMI device writes its current date and the time in the data area configured in the "Date/time" area pointer. All definitions are coded in BCD format.

Note

You cannot use the "Date/Time PLC" area pointer if you have configured the "Date/Time" area pointer.

The "Date/Time" area pointer when used in a project which contains multiple connections must be enabled for each configured connection.

The date/time data area has the following structure:

Data word	Most significant byte								Least significant byte								
	7							0	7							0	
n+0	Reserved								Hour (0 to 23)								Time
n+1	Minute (0 to 59)								Second (0 to 59)								
n+2	Reserved								Reserved								
n+3	Reserved								Weekday (1 to 7, 1=Sunday)								Date
n+4	Day (1 to 31)								Month (1 to 12)								
n+5	Year (80 to 99/0 to 29)								Reserved								

Note

When making entries in the "Year" data area, you should note that values 80 to 99 result in years 1980 through 1999, while the values 0 to 29 result in the years 2000 through 2029.

Using data types

The data types "Date_And_Time, DTL" and "LDT" can only be used with the "Date/time" and "Date/time PLC" area pointers.

The data format of the "Date/time" area pointer depends on job mailbox 40/41.

If there are no control tags linked to the area pointer, or a control tag is linked with the data type "Array[0..5] of UInt/Word/Int", the following applies:

The configuration of the "Date/time" area pointer is only used for job mailbox 41.

If job mailbox 40 is used, the data format "DATE_AND_TIME (BCD-encoded)" is used (shown in the next section).

If the "Date/time" and "Date/time PLC" area pointers are linked to a control tag with the data type "DATE_AND_TIME", "DTL" or "LDT", the associated data format is used in the corresponding area pointer.

"Date/time PLC" area pointer

Function

This area pointer is used to transfer the date and time from the PLC to the HMI device. Use this area pointer if the PLC is the time master.

The PLC loads the data area of the area pointer. All definitions are coded in BCD format.

The HMI device reads the data cyclically within the configured acquisition cycle and synchronizes itself.

Note

Set an acquisition cycle of sufficient length for the Date/time area pointer in order to avoid any negative impact on HMI device performance.
Recommended: Acquisition cycle of 1 minute, if the process allows this.

"Date/Time PLC" is a global area pointer and may be configured only once per project.

Note

You cannot use the "Date/Time" area pointer if you have configured the "Date/Time PLC" area pointer.

The date/time data area has the following structure:

DATE_AND_TIME format (in BCD code)

Data word	Most significant byte			Least significant byte		
	7	. . . . .	0	7	. . . . .	0
n+0	Year (80 to 99/0 to 29)			Month (1 to 12)		
n+1	Day (1 to 31)			Hour (0 to 23)		
n+2	Minute (0 to 59)			Second (0 to 59)		
n+3	Reserved			Reserved		Weekday (1 to 7, 1=Sunday)
n+4 [1)]	Reserved			Reserved		
n+5 [1)]	Reserved			Reserved		

1) The two data words must exist in the data area to ensure that the data format matches WinCC flexible and to avoid reading false information.

Note

When making entries in the "Year" data area, you should note that values 80 to 99 result in years 1980 through 1999, while the values 0 to 29 result in the years 2000 through 2029.

Using data types

The data types "Date_And_Time, DTL" and "LDT" can only be used with the "Date/time" and "Date/time PLC" area pointers.

The data format of the "Date/time" area pointer depends on job mailbox 40/41.

If there are no control tags linked to the area pointer, or a control tag is linked with the data type "Array[0..5] of UInt/Word/Int", the following applies:

The configuration of the "Date/time" area pointer is only used for job mailbox 41.

If job mailbox 40 is used, the data format "DATE_AND_TIME (BCD-encoded)" is used (shown in the next section).

If the "Date/time" and "Date/time PLC" area pointers are linked to a control tag with the data type "DATE_AND_TIME", "DTL" or "LDT", the associated data format is used in the corresponding area pointer.

"Coordination" area pointer

Function

The "Coordination" area pointer is used to implement the following functions:

- Detecting the startup of the HMI device in the control program
- Detecting the current operating mode of the HMI device in the control program
- Detecting whether the HMI device is ready to communicate in the control program

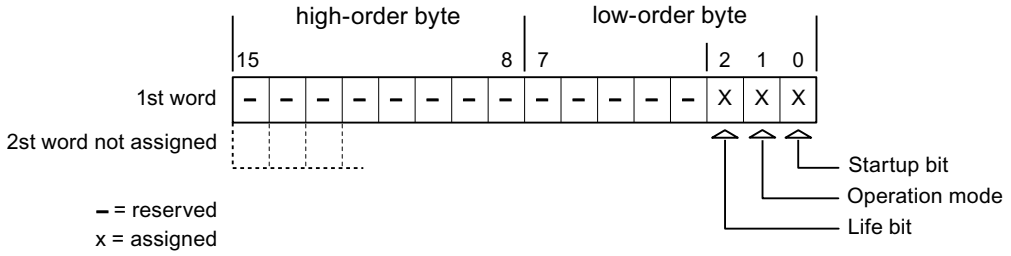
By default, the "Coordination" area pointer has the length of one word and cannot be changed.

Usage

Note

The HMI device always writes the entire coordination area when updating the area pointer. The control program can for this reason not make changes to the coordination area.

Assignment of the bits in the "Coordination" area pointer



Startup bit

The startup bit is set briefly to "0" by the HMI device during startup. It sets the bit permanently to "1" when startup is completed.

Operating mode

The operating mode bit is set to 1 as soon as the user switches the HMI device offline. The status of the operating mode bit is "0" during normal operation of the HMI device. You can determine the current operating mode of the HMI device by reading this bit in the control program.

Life bit

The HMI device inverts the life bit at intervals of approximately one second. You can check whether or not there is still a connection to the HMI device by querying this bit in the control program.

"Project ID" area pointer

Function

When Runtime starts, a check can be carried out as to whether the HMI device is connected to the correct PLC. This check is important when operating with several HMI devices.

For this, the HMI device compares a value stored on the PLC with the value specified in configuration. This ensures compatibility of the configuration data with the PLC program. If there is no concordance, a system event is given on the HMI device and Runtime is stopped.

Use

Note

HMI connections cannot be switched "online".

The HMI connection in which the "Project ID" area pointer is used must be switched "online".

To use this area pointer, set up the following during the configuration:

- Define the version of the configuration. Values between 1 and 255 are possible.
  You enter the version in the editor "Runtime settings > General" in the "Identification" area.
- Data address of the value for the version that is stored in the PLC:
  You enter the data address in the editor "Communication > Connections" under "Address".

Connection failure

A connection failure to a device on which the "project ID" area pointer is configured results in all the other connections of the device being switched to "offline".

This behavior has the following requirements:

- You have configured several connections in a project.
- You are using the "project ID" area pointer in at least one connection.

Causes which may set connections "offline":

- The PLC is not available.
- The connection has been switched offline in the engineering system.

"Job mailbox" area pointer

Function

The PLC can use the job mailbox to transfer jobs to the HMI device to trigger corresponding actions on the HMI device. These functions include, for example:

- Display screen
- Set date and time

Data structure

The first word of the job mailbox contains the job number. Depending on the job mailbox, up to three parameters can be transferred.

Word	Most significant byte	Least significant byte
n+0	0	Job number
n+1	Parameter 1	
n+2	Parameter 2	
n+3	Parameter 3	

The HMI device evaluates the job mailbox if the first word of this job is not equal to zero. This means that the parameters must be entered in the job mailbox first, followed by the job number.

When the HMI device accepts the job mailbox, the first word is set to 0 again. The execution of the job mailbox is generally not completed at this point in time.

Job mailboxes

All job mailboxes and their parameters are listed below. The "No." column contains the job number of the job mailbox. Job mailboxes can only be triggered by the PLC when the HMI device is online.

No.	Function	
14	Set time (BCD-coded)	
	Parameter 1	Left byte: - Right byte: hours (0-23)
	Parameter 2	Left byte: minutes (0-59) Right byte: seconds (0-59)
	Parameter 3	-
15	Setting the date (BCD coded) [3) 4)]	
	Parameter 1	Left byte: - Right byte: weekday (1-7: Sunday-Saturday)
	Parameter 2	Left byte: day (1-31) Right byte: month (1-12)
	Parameter 3	Left byte: year
23	User logon	
	Logs the user on with the name "PLC user" at the HMI device with the group number transferred in Parameter 1. The logon is possible only when the transferred group number exists in the project.	
	Parameter 1	Group number 1 to 255
	Parameter 2, 3	-
24	User logoff	
	Logs off the current user. (The function corresponds to the "logoff" system function)	
	Parameter 1, 2, 3	-
40	Transfer date/time to PLC	
	(in the S7 format DATE_AND_TIME) An interval of at least 5 seconds must be maintained between two successive jobs to prevent overload of the HMI device.	
	Parameter 1, 2, 3	-
41	Transfer date/time to PLC	
	(In OP/MP format) An interval of at least 5 seconds must be maintained between successive jobs to prevent overload of the HMI device.	
	Parameter 1, 2, 3	-
46	Update tag	
	Causes the HMI device to read the current value of the tags from the PLC whose update ID matches the value transferred in Parameter 1. (Function corresponds to the "UpdateTag" system function.)	
	Parameter 1	1 - 100
49	Delete alarm buffer	
	Deletes all analog alarms and discrete alarms of the "Warnings" class from the alarm buffer.	

No.	Function	
14	**Set time (BCD-coded)**	
	Parameter 1, 2, 3	-
50	**Delete alarm buffer**	
	Deletes all analog alarms and discrete alarms of the "Errors" class from the alarm buffer.	
	Parameter 1, 2, 3	-
51	**Screen selection** [2]	
	Parameter 1	Screen number
	Parameter 2	-
	Parameter 3	Field number
69	**Reading data record from PLC** [1]	
	Parameter 1	Recipe number (1-999)
	Parameter 2	Data record number (1-65535)
	Parameter 3	0: Do not overwrite existing data record 1: Overwrite existing data record
70	**Writing data record to PLC** [1]	
	Parameter 1	Recipe number (1-999)
	Parameter 2	Data record number (1-65535)
	Parameter 3	-

[1]	Only for devices supporting recipes.
[2]	OP 73, OP 77A and TP 177A HMI devices also execute the "Screen selection" job mailbox if the on-screen keyboard is active.
[3]	The weekday is ignored on HMI device KTP 600 BASIC PN.
[4]	The weekday is ignored when you configure the "Date/Time PLC" area pointer.

"Data record" area pointer

"Data record" area pointer

Function

When data records are transferred between the HMI device and PLC, both partners access common communications areas on the PLC.

Data transfer types

There are two ways of transferring data records between the HMI device and PLC:

- Transfer without synchronization
- Transfer with synchronization via the data mailbox

Data records are always transferred directly, which means that the tag values are read straight from an address or written straight to an address configured for this tag without being redirected via an interim memory.

Initiating the transfer of data records

There are three ways of triggering the transfer:

- Operator input in the recipe view
- Job mailboxes
  The transfer of data records can also be triggered by the PLC.
- Triggering by configured functions

If the transfer of data records is triggered by a configured function or by a job mailbox, the recipe view on the HMI device remains operable. The data records are transferred in the background.

Simultaneous processing of several transfer requests is, however, not possible. In this case, the HMI device rejects the other transfer requests with a system alarm.

Transfer without synchronization

If you select asynchronous transfer of data records between the HMI device and PLC, there is no coordination over the common data areas. It is therefore unnecessary to set up a data area during configuration.

Asynchronous data record transfer can be a useful alternative, for example when:

- The system is capable of excluding the risk of uncontrolled overwriting of data by the communication peer.
- The PLC does not require information about the recipe number and data record number.
- The transfer of data records is triggered by the operator of the HMI device.

Reading values

When a read job is triggered, the values are read from the PLC addresses and transferred to the HMI device.

- Triggering by the operator in the recipe view:
  The values are downloaded to the HMI device. You can then, for example, process, edit, or save these values in the HMI device.
- Triggering by a function or job mailbox:
  The values are saved immediately to the data volume.

Writing values

When a write job is triggered, the values are written to the PLC addresses.

- Triggering by the operator in the recipe view:
  The current values are written to the PLC.
- Triggering by a function or job mailbox:
  The current values are written to the PLC from the data medium.

Transfer with synchronization

If you select synchronous transfer, both communication partners set status bits in the common data area. You can use this mechanism to prevent uncontrolled overwriting of data in either direction in your control program.

Application

Synchronous data record transfer can be a useful solution, for example, when:

- The PLC is the "active partner" in the transfer of data records.
- The PLC evaluates the information about the recipe number and data record number.
- The transfer of data records is triggered by means of a Job mailbox.

Requirements

In order to synchronize transfer of data records between the HMI device and the PLC, the following requirements must be met during configuration:

- An area pointer has been set up: "Communication > Connections" editor in "Area pointer".
- The PLC with which the HMI device synchronizes transfer of data records is specified in the recipe:
  "Recipes" editor in the inspector window the option "Coordinated transfer of data records" under "General > Synchronization > Settings"

Structure of the data area

The data area has a fixed length of 5 words. Structure of the data area:

	15		0
1. Word	Current recipe number (1 - 999)		
2. Word	Current data record number (0 - 65535)		
3. Word	Reserved		
4. Word	Status (0, 2, 4, 12)		
5. Word	Reserved		

- Status
  The status word (word 4) can adopt the following values:

Value		Meaning
Decimal	Binary	
0	0000 0000	Transfer permitted, data mailbox free
2	0000 0010	Transferring
4	0000 0100	Transfer completed without error
12	0000 1100	Transfer completed with error

Sequence of a transfer triggered by a job mailbox

The transfer of data records between the HMI device and the PLC can be initiated by either one of these stations.

The two job mailboxs No. 69 and No. 70 are available for this type of transfer.

No. 69: Read data record from PLC ("PLC → DAT")

Job mailbox no. 69 transfers data records from the PLC to the HMI device. The job mailbox is structured as follows:

	Left byte (LB)	Right byte (RB)
Word 1	0	69
Word 2	Recipe number (1-999)	
Word 3	Data record number (1 to 65535)	
Word 4	Do not overwrite existing data record: 0 Overwrite existing data record: 1	

No. 70: Write data record to PLC ("DAT → PLC")

Job mailbox No. 70 transfers data records from the HMI device to the PLC. The job mailbox is structured as follows:

	Left byte (LB)	Right byte (RB)
Word 1	0	70
Word 2	Recipe number (1-999)	
Word 3	Data record number (1-65,535)	
Word 4	—	

Sequence when reading from the PLC with job mailbox "PLC → DAT" (no. 69)

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the job and the status "Transferring" in the data record.	Abort without return message.
3	The HMI device reads the values from the PLC and saves these to the data record defined in the job mailbox.	
4	• If "Overwrite" was selected in the job, an existing data record is overwritten without any prompt for confirmation. The HMI device sets the status "Transfer completed." • If "Do not overwrite" was selected in the job, and the data record already exists, the HMI device aborts the job and enters 0000 1100 in the status word of the data mailbox.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Sequence of writing to the PLC with job mailbox "DAT → PLC" (no. 70)

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the job and the status "Transferring" in the data mailbox.	Abort without return message.
3	The HMI device fetches the values of the data record specified in the function from the data medium and writes the values to the PLC.	
4	The HMI device sets the status "Transfer completed."	
5	The control program can now evaluate the transferred data. The control program must reset the status word to zero in order to enable further transfers.	

Sequence of the transfer when triggered by a configured function

Reading from the PLC using a configured function

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the function and the status "Transferring" in the data mailbox.	Abort with system event.
3	The HMI device reads the values from the PLC and stores them in the data record specified in the function.	

Step	Action	
4	• If "Yes" was selected for the "Overwrite" function, an existing data record is overwritten without any prompt for confirmation. The HMI device sets the status "Transfer completed." • If "No" was selected for the "Overwrite" function and the data record already exists, the HMI device aborts the job and enters 0000 1100 in the status word of the data mailbox.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Writing to the PLC by means of configured function

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the function and the status "Transferring" in the data mailbox.	Abort with system event.
3	The HMI device fetches the values of the data record specified in the function from the data medium and transfers the values to the PLC.	
4	The HMI device sets the status "Transfer completed."	
5	The control program can now evaluate the transferred data. The control program must reset the status word to zero in order to enable further transfers.	

Possible causes of error when transferring data records

Possible causes of error

The section below shows possible error causes which lead to the cancellation of data record transfer:

- Tag address not set up on the PLC
- Overwriting data records not possible

- Recipe number does not exist
- Data record number does not exist

Note

The status word may only be set by the HMI device. The PLC may only reset the status word to zero.

Note

The PLC may only evaluate the recipe and data record numbers when data inconsistency is detected if one of the conditions outlined below has been met:

- The data mailbox status is set to "Transfer completed".
- The data mailbox status is set to "Transfer completed with error".

Reaction to an aborted transfer due to errors

If the transfer of data records is aborted due to errors, the HMI device reacts as follows:

- Triggering by the operator in the recipe view
  Information in the status bar of the recipe view and output of system alarms
- Triggered by function
  Output of system alarms
- Triggering by job mailbox
  No return message on the HMI device

You can nonetheless evaluate the status of the transfer by querying the status word in the data record.

Trends

General information on trends

Trends

A trend is the graphical representation of one or more values from the PLC. The value is read out time-triggered for Basic Panels.

For additional information see:

Auto-Hotspot

Time-triggered trends

The HMI device reads in the trend values cyclically at an interval specified in the configuration.

Time-triggered trends are suitable for continuous curves, such as the operating temperature of a motor.

Trend request and trend transfer

Trend request area

The HMI device sets corresponding bits in the trend request area when you open a screen which contains one or more trends on the HMI device. After closing the screen, the HMI device resets the relevant bits in the trend request area.

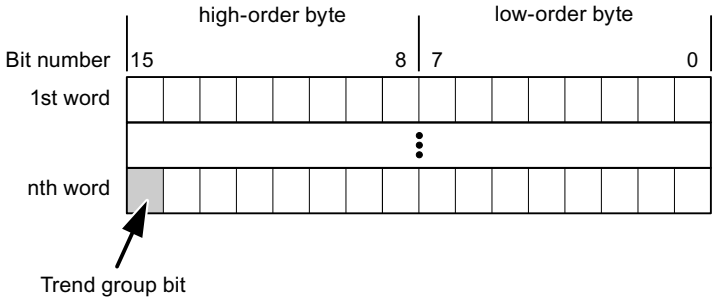
The trend request area can be used for evaluation purposes in the PLC to determine which trend is currently being displayed on the HMI device. Trends can also be triggered without evaluation of the trend request area.

Trend transfer area 1

This area is used to trigger trends. You must set the bit assigned to the trend in the trend transfer area and set the trend group bit in your control program. The trend group bit is the last bit in the trend transfer area.

The HMI device detects the trigger. The HMI device reads either a value or the entire buffer from the PLC. It then resets the trend bit and the trend group bit.

The following picture shows the structure of a trend transfer area.



The trend transfer area must not be modified by the PLC program until the trend group bit has been reset.

Trend transfer area 2

Trend transfer area 2 is required for trends configured with a switch buffer. The trend transfer areas 1 and 2 have a similar structure.

Switch buffer

The switch buffer is a second buffer for the same trend that can be set up during configuration.

The PLC writes to Buffer 2 while the HMI device reads values from Buffer 1, and writes to Buffer 1 when the HMI device is reading Buffer 2. This prevents the PLC from overwriting trend values while the trend is being read by the HMI device.

Permitted data types for trends

For SIMATIC S7

You assign one bit to each trend during configuration. Tags and array tags of the "Word" or "Int" data type are permitted.

Alarms

General information on trends

Trends

A trend is the graphical representation of one or more values from the PLC. The value is read out either time- or bit-triggered, depending on the configuration.

For additional information see:

Auto-Hotspot

Note

The value is read out time-triggered for Basic Panels.

Time-triggered trends

The HMI device reads in the trend values cyclically at an interval specified in the configuration. Time-triggered trends are suitable for continuous curves, such as the operating temperature of a motor.

Bit-triggered trends

Through a trigger bit set in the trend request tag, the HMI device either reads in a trend value or an entire trend buffer. This setting is defined in the configuration. Bit-triggered trends are normally used to represent fast changing values. One example might be the injection pressure in the production of plastic parts.

To trigger bit-triggered trends, appropriate external tags must be created in the "HMI tags" editor and connected to trend areas during configuration. The HMI device and PLC then communicate with each other via these trend areas.

The following areas are available for trends:

- Trend request area
- Trend transfer area 1
- Trend transfer area 2 (required only with switch buffers)

Acknowledgment of alarms

Procedure

Create suitable tags on the PLC to acknowledge an error alarm. You assign these tags to an alarm in the "Bit messages" editor. You make the assignment in "Properties > Acknowledgment".

Distinction in terms of acknowledgment:

- Acknowledgment by the PLC
- Acknowledgment on the HMI device

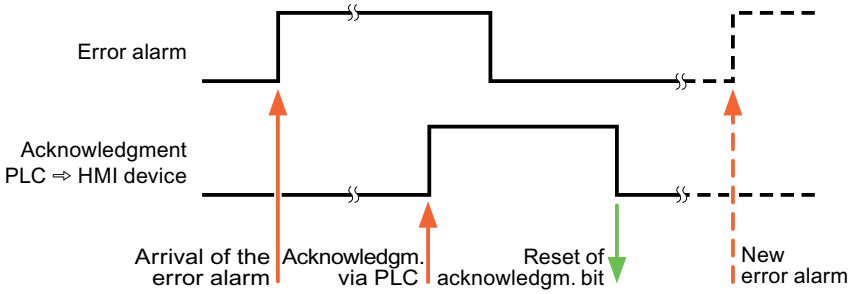
Acknowledgment by the PLC

In "PLC acknowledgment tag", configure the tag or array tag and the bit number that the HMI device uses to identify a PLC acknowledgment.

A bit set in the tag triggers acknowledgment of the assigned error alarm bit at the HMI device. This tag bit returns a function similar to acknowledgment on the HMI device which is triggered by pressing the "ACK" button, for example.

The acknowledgment bit must be located in the same tag as the bit for the error alarm.

Reset the acknowledgment bit before setting the bit in the alarm area again. The figure below shows the pulse diagram.



Acknowledgment on the HMI device

In the "HMI acknowledgment tag" area, configure the tag or array tag as well as the bit number that the HMI device writes to the PLC after acknowledgment. Make sure when you use an array tag that it is not longer than 6 words.

To always create a signal change when setting an assigned acknowledgment bit of a discrete alarm that must be acknowledged, the HMI device will reset the acknowledgment bit assigned to the alarm as soon as it detects an alarm subject to acknowledgment and write the acknowledgment tag in the PLC. There will be a certain delay between detecting the message and writing the acknowledgment tag in the PLC because the HMI device has to process the operations.

If a discrete alarm subject to acknowledgment is acknowledged by the HMI device, then the corresponding bit in the assigned acknowledgment tag will be set. The entire acknowledgment

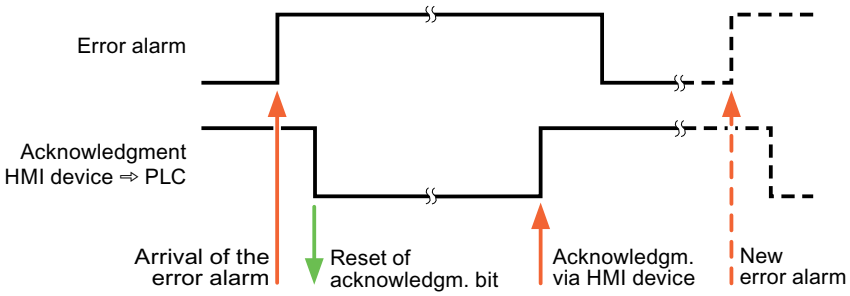
tag is then written to the PLC by the HMI device. This allows the PLC to recognize that a certain alarm message has been acknowledged at the HMI device.

Note

All alarm bits acknowledged since the last Runtime start will remain in the acknowledgment tag until a new incoming of the respective discrete alarms is detected.

This area should only be read by the PLC because the entire section of the HMI device will be overwritten once the next acknowledgment tag is written.

The figure below shows the pulse diagram.



10.8.6.5 Performance features of communication

S7-1500 device dependency

Device dependency

If you use devices from an earlier version of the TIA Portal with TIA Portal V13, it may not be possible to configure connections to certain HMI devices.

Basic Panels V11.0

HMI devices	SIMATIC S7 -1500
KP300 Basic	No
KP400 Basic	No
KTP400 Basic PN	No
KTP600 Basic DP	No
KTP600 Basic PN	No
KTP1000 Basic DP	No
KTP1000 Basic PN	No
TP1500 Basic PN	No

Basic Panels V12.0

HMI devices	SIMATIC S7 -1500
KP300 Basic	Yes
KP400 Basic	Yes
KTP400 Basic PN	Yes
KTP600 Basic DP	Yes
KTP600 Basic PN	Yes
KTP1000 Basic DP	Yes
KTP1000 Basic PN	Yes
TP1500 Basic PN	Yes

Basic Panels V13.0

HMI devices	SIMATIC S7 -1500
KTP400 Basic	Yes
KTP700 Basic	Yes
KTP900 Basic	Yes
KTP1200 Basic	Yes

Valid data types for SIMATIC S7 1500

Valid data types for connections with SIMATIC S7 1500

The table lists the data types that can be used when configuring tags and area pointers.

Data type	Length
BOOL	1 bit
BYTE	1 byte
WORD	2 bytes
DWORD	4 bytes
CHAR	1 byte
INT	2 bytes
DINT	4 bytes
REAL	4 bytes
TIME	4 bytes
DATE	2 bytes
TIME_OF_DAY	4 bytes
S5TIME	2 bytes
COUNTER	2 bytes
TIMER	2 bytes

Data type	Length
DATE_AND_TIME	8 bytes
STRING	(2+n) bytes, n = 0 to 254
DTL	12 bytes
LDT	8 bytes
LINT	8 bytes
LREAL	8 bytes
LTIME	8 bytes
LTIME_OF_DAY	8 bytes
SINT	1 byte
UDINT	4 bytes
UINT	2 bytes
ULINT	8 bytes
USINT	1 byte

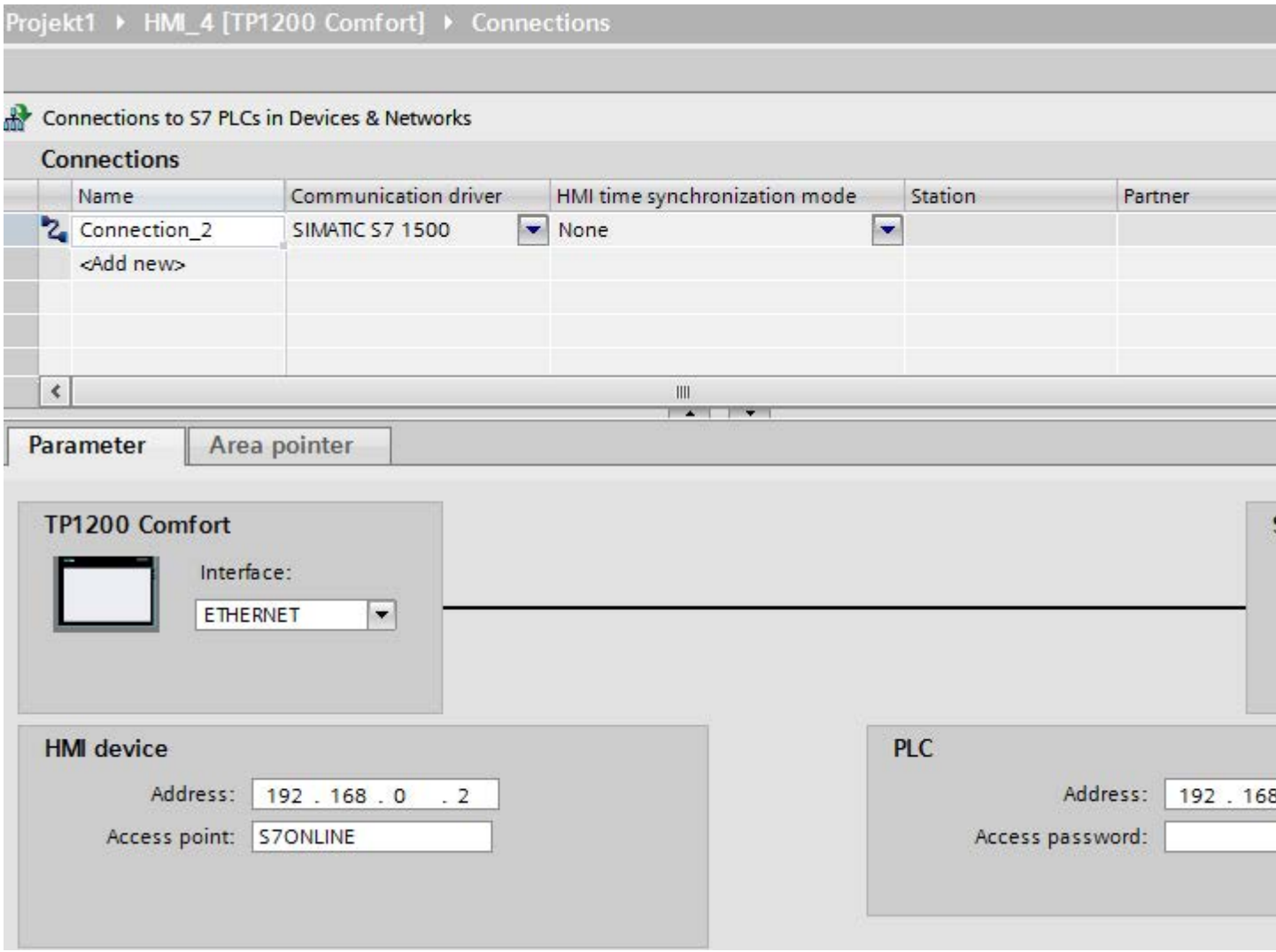
10.8.6.6 Configuring connections in the "Connections" editor

Connection parameters

Parameters to be set

To set the connection parameters, such as addresses and profiles, click the connection that you have created in the "Connections" editor.

The communication partners are displayed schematically in the Inspector window under "Parameters". The "HMI device", "Network", and "PLC" areas are available for assigning parameters according to the interface used.


Projekt1 ▸ HMI_4 [TP1200 Comfort] ▸ Connections
Connections to S7 PLCs in Devices & Networks
Connections
Name
Communication driver
HMI time synchronization mode
Station
Partner
Connection_2
SIMATIC S7 1500
None
<Add new>
Parameter
Area pointer
TP1200 Comfort
Interface:
ETHERNET
HMI device
Address: 192 . 168 . 0 . 2
Access point: S7ONLINE
PLC
Address: 192 . 168
Access password:

Ethernet parameters

Parameters for the HMI device

You set the parameters for the HMI device in the network under "HMI device".. The changes are not transferred automatically to the HMI device. You must change the settings in the Control Panel of the HMI device.

- "Interface"
  If you are directly connected to the HMI device during configuration, you can set up the IP address of the HMI device in WinCC.

  Note

  The IP address in the Control Panel will be overwritten upon subsequent loading if you have already set up the IP address in the HMI device control panel.

  The IP address already set up in the Control Panel will be retained upon subsequent loading if you activate "Set IP address using a different method".

  The IP address is transferred to the HMI device during project transfer.
  To set up the IP address of the HMI device:

  – Click the HMI device.
  – Open the "Device configuration" editor.
  – Click the Ethernet interface.
  – Assign the IP address in the inspector window under:
    "General > PROFINET interface > Ethernet addresses"

- "Address"
  You assign the IP address of the HMI device in the "Address" area.
  When you transfer the WinCC project to the HMI device, this IP address is set up directly in the HMI device.

- "Access point"
  Specifies the access point for the PG/PC interface that can be used to reach the communication partner.

Parameters for the PLC

Under "PLC", you address the S7 module with which the HMI device will exchange data. Assign a name for the connection for each communication partner.

- "Address"
  Under "Address", set the IP address of the S7 module to which the HMI device is connected.
- "Access password"
  Enter a password in the "Access password" field. This password must match the one you saved to the PLC.

Note

You only need a password if you have set "Complete protection" at the PLC.

No connection is set up to the PLC if the "Complete protection" security level is stored on the PLC and you do not enter a password.

PROFIBUS parameters

Parameters for the HMI device

You assign the parameters for the HMI device in the network once under "HMI device". The change applies to each communication partner.

- "Type"
  Specifies the physical connection used.
- "Interface"
  For "Interface", you select the HMI device interface via which the HMI device is connected to the PROFIBUS network.
- "Baud rate"
  For "Baud rate", you set the transmission speed of the data in the network. The baud rate is determined by the slowest HMI device connected to the network. The setting must be identical throughout the network.

Note

If you set a baud rate of 1.5 Mbaud for OP 73 or OP 77A, the highest station address must be less than or equal to 63.

- "Address"
  You set the PROFIBUS address of the HMI device under "Address". The PROFIBUS address must be unique in the PROFIBUS network.

- "Only master on bus"
  Disables an additional safety feature against bus faults when the HMI device is connected to the network. A passive station (slave) can only send data if it is requested to do so by an active station (master).
  In S7-200, you must set an HMI device as the master.
- "Access point"
  The access point defines a logical device name through which the communication partner can be reached.

Parameters for the network

Under "Network", you set the parameters for the PROFIBUS network to which the HMI device is linked.

- "Profile"
  For "Profile", you select the network profile that is used in the network. In "Profile", set "DP", "Universal", or "Standard". The setting must be identical throughout the network.
- "Highest address"
  For "Highest station address", set the highest station address. The highest station address must be greater than or equal to the highest actual PROFIBUS address. The setting must be identical throughout the network.

  Note

  If you set a baud rate of 1.5 Mbaud in the OP 73 or the OP 77A, the highest station address must be less than or equal 63.

- "Number of masters"
  For "Number of masters", set the number of masters in the PROFIBUS network. This information is necessary to correctly calculate the bus parameters.

Parameters for the PLC

Under "PLC", you address the S7 module with which the HMI device will exchange data. Assign a name for the connection for each communication partner.

- "Address"
  For "Address", set the PROFIBUS address of the S7 module (CPU, FM, or CP) to which the HMI device is connected.
- "Access password"
  Enter a password in the "Access password" field. This password must match the one you saved to the PLC.

Note

You only need a password if you have set "Complete protection" at the PLC.

No connection is set up to the PLC if the "Complete protection" security level is stored on the PLC and you do not enter a password.

10.8.7 Communicating with SIMATIC S7 1200

10.8.7.1 Communication with SIMATIC S7 1200

Introduction

This section describes the communication between an HMI device and the SIMATIC S7 1200 PLC.

You can configure the following communication channels for the SIMATIC S7 1200 PLC:

- PROFINET
- PROFIBUS

HMI connection for communication

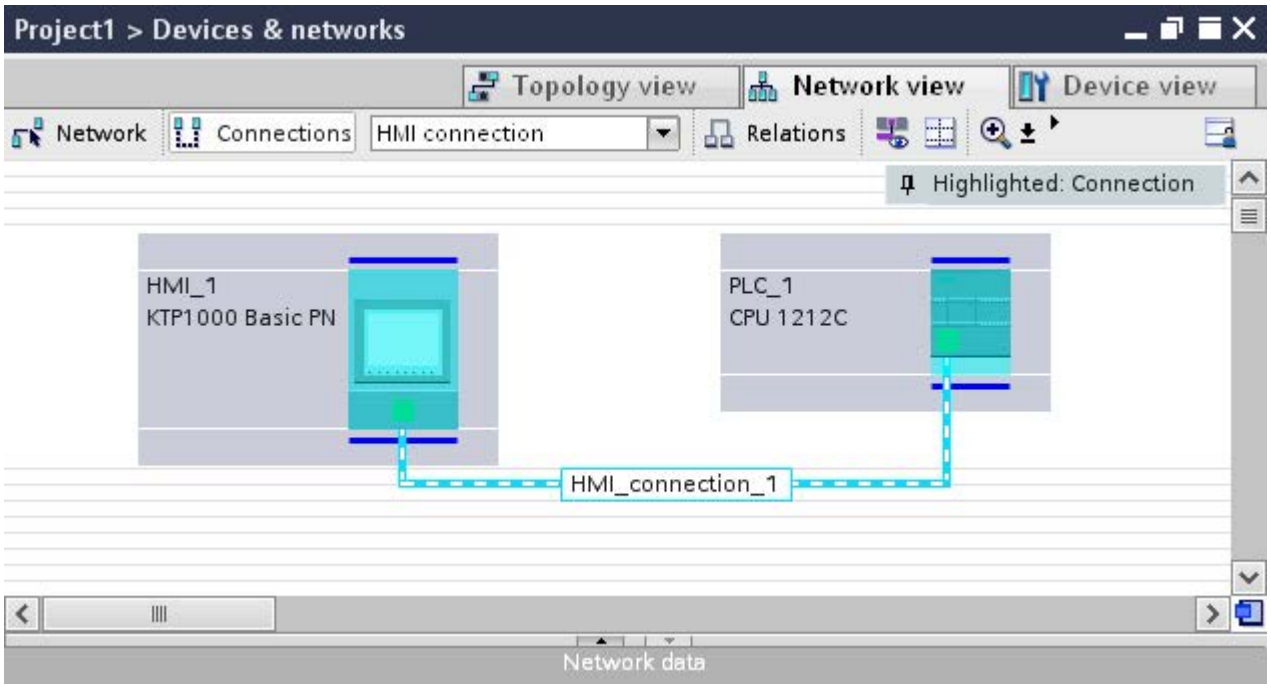
You configure connections between the HMI device and a SIMATIC S7 1200 in the "Devices & Networks" editor. If you have configured a HMI device with a serial port, you must configure a PROFIBUS-capable communication module to the SIMATIC S7 1200.

10.8.7.2 Communication via PROFINET

Communication via PROFINET

HMI connections via PROFINET

If you have inserted an HMI device and a SIMATIC S7 1200 into the project, you interconnect the two PROFINET interfaces in the "Devices & Networks" editor.



You can also connect multiple HMI devices to one SIMATIC S7 1200 and multiple SIMATIC S7 1200s to one HMI device. The maximum number of communication partners that you can connect to an HMI device is dependent on the HMI device used.

Additional information is available in the documentation for the respective HMI device.

HMI connection in the "Devices & Networks" editor

You configure the HMI connection between the PLC and the HMI device via PROFINET in the "Devices & Networks" editor.

Connection in the "Connections" editor

Alternatively, you configure the connection between the PLC and HMI device via PROFINET in the "Connections" editor of the HMI device.

Configuring an HMI connection via PROFINET

Introduction

You configure an HMI connection between HMI devices and a SIMATIC S7 1200 via PROFINET or Ethernet in the "Devices & Networks" editor.

> **CAUTION**
>
> **Communication via Ethernet**
>
> In Ethernet-based communication, the end user is responsible for the security of his data network.
>
> Targeted attacks can overload the device and interfere with proper functioning.

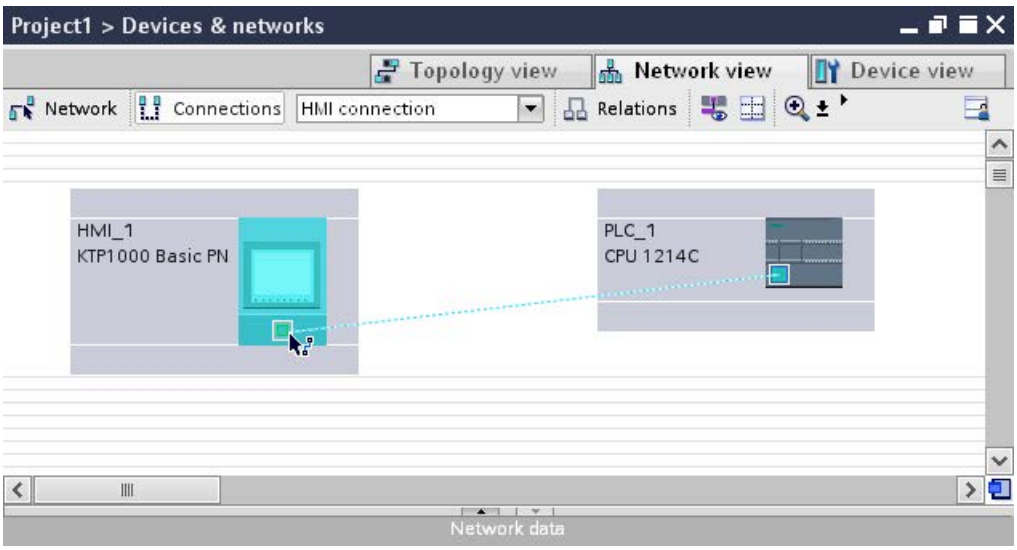
Requirements

The following communication partners are created in the "Devices & Networks" editor:

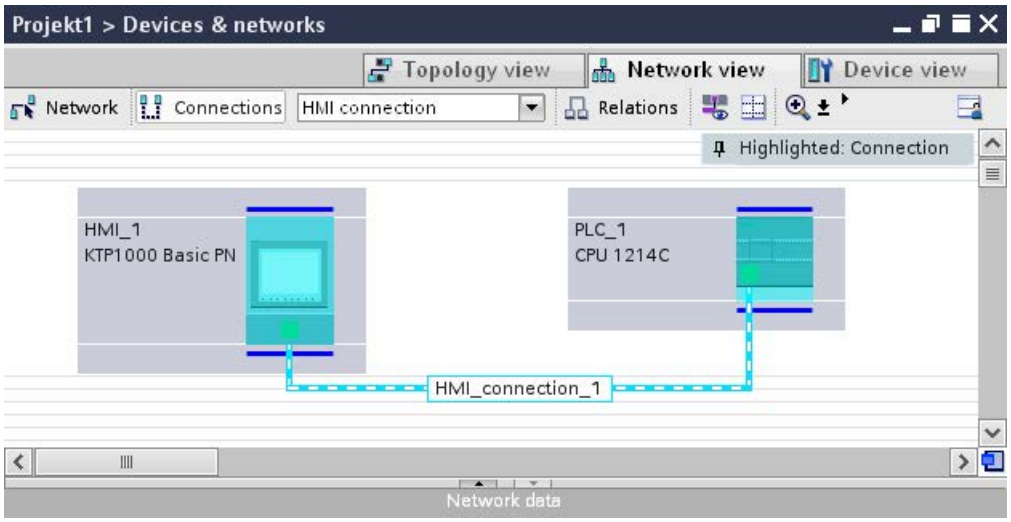
- SIMATIC S7 1200
- HMI device with PROFINET or Ethernet interface

Procedure

1. Double-click the "Devices & Networks" item in the project tree.
   The available communication partners in the project are displayed graphically in the network view.
2. Click the "Connections" button and select "HMI connection" for the connection type.
   The devices available for connection are highlighted in color.
3. Click the PROFINET interface of the PLC and use a drag-and-drop operation to draw a connection to the PROFINET or Ethernet interface of the HMI device.



4. Click the connecting line.
5. Click "Highlight HMI connection" and select the HMI connection.



The connection is displayed graphically in the Inspector window.

6. Click the communication partners in the "Network view" and change the PROFINET parameters in the Inspector window according to the requirements of your project. See the chapter "PROFINET parameters (Page 8480)" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. You check the connection parameters in the table.

You can change the local name for the connection only in the table.

Result

You have created a connection between an HMI device and a SIMATIC S7 1200. The IP address and subnet mask connection parameters are configured.

PROFINET parameters

PROFINET parameters for the HMI connection

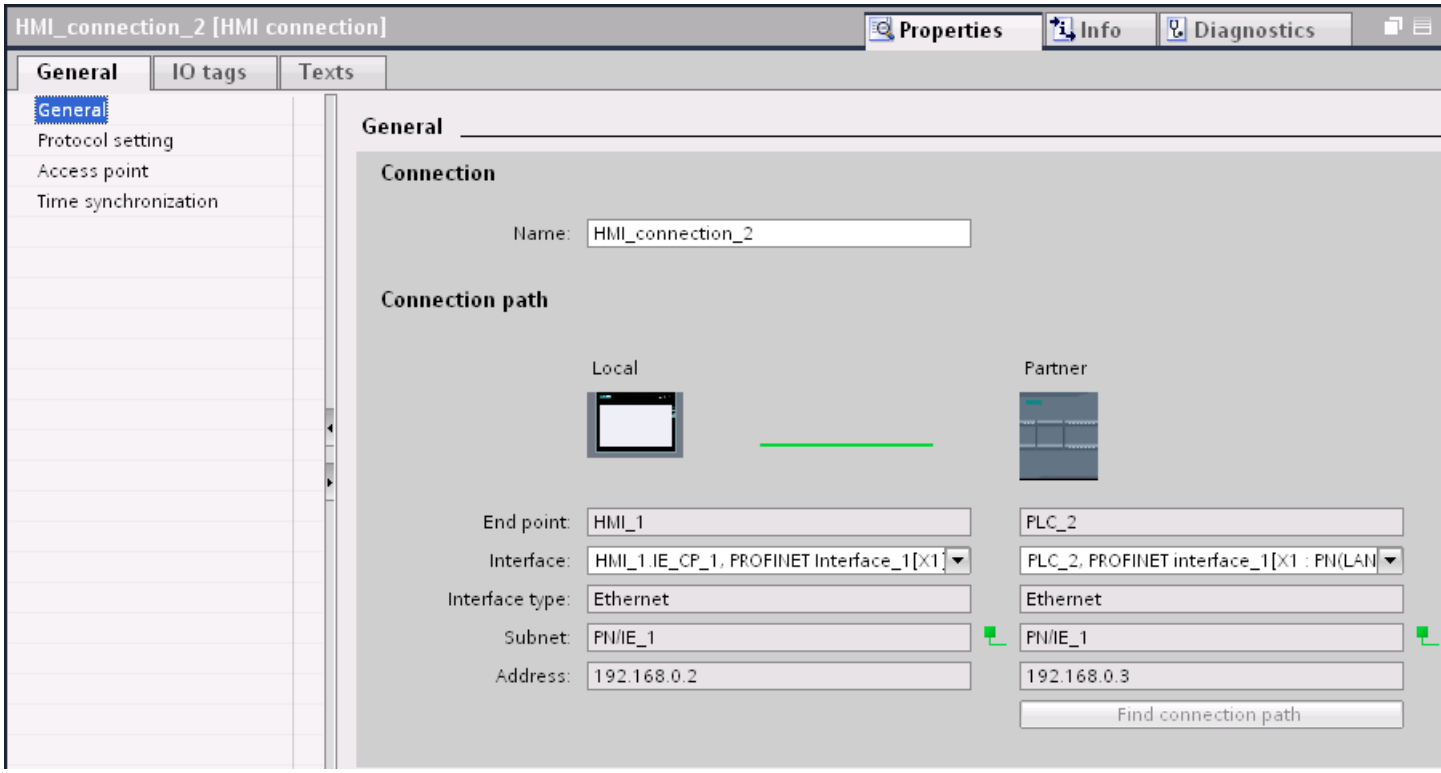
PROFINET parameters for the HMI connection

An overview of the configured HMI connection parameters can be found in the properties for the HMI connection.

Only limited changes are possible in this Inspector window.

Displaying and editing HMI connection parameters

1. Click the HMI connection in the "Devices & Networks" editor.
2. Change the parameters of the HMI connection in the Inspector window under "Properties > General > General".



Connection

The "Connection" area displays the HMI connection created for communication between the devices.

You can edit the name of the HMI connection in this area.

"Connection path"

The communication partners of the selected HMI connection and the associated PROFINET parameters are displayed in the "Connection path" area. Some of the areas displayed cannot be edited in this dialog.

- "End point"
  Displays the device name. This area cannot be edited.
- "Interface"
  Displays the selected interface of the device. You can choose between several interfaces, depending on the device.
- "Interface type"
  Displays the selected interface type. This area cannot be edited.

- "Subnet"
  Displays the selected subnet. This area cannot be edited.
- "Address"
  Displays the selected IP address of the device. This area cannot be edited.
- "Find connection path" button
  Enables the subsequent specification of connections.

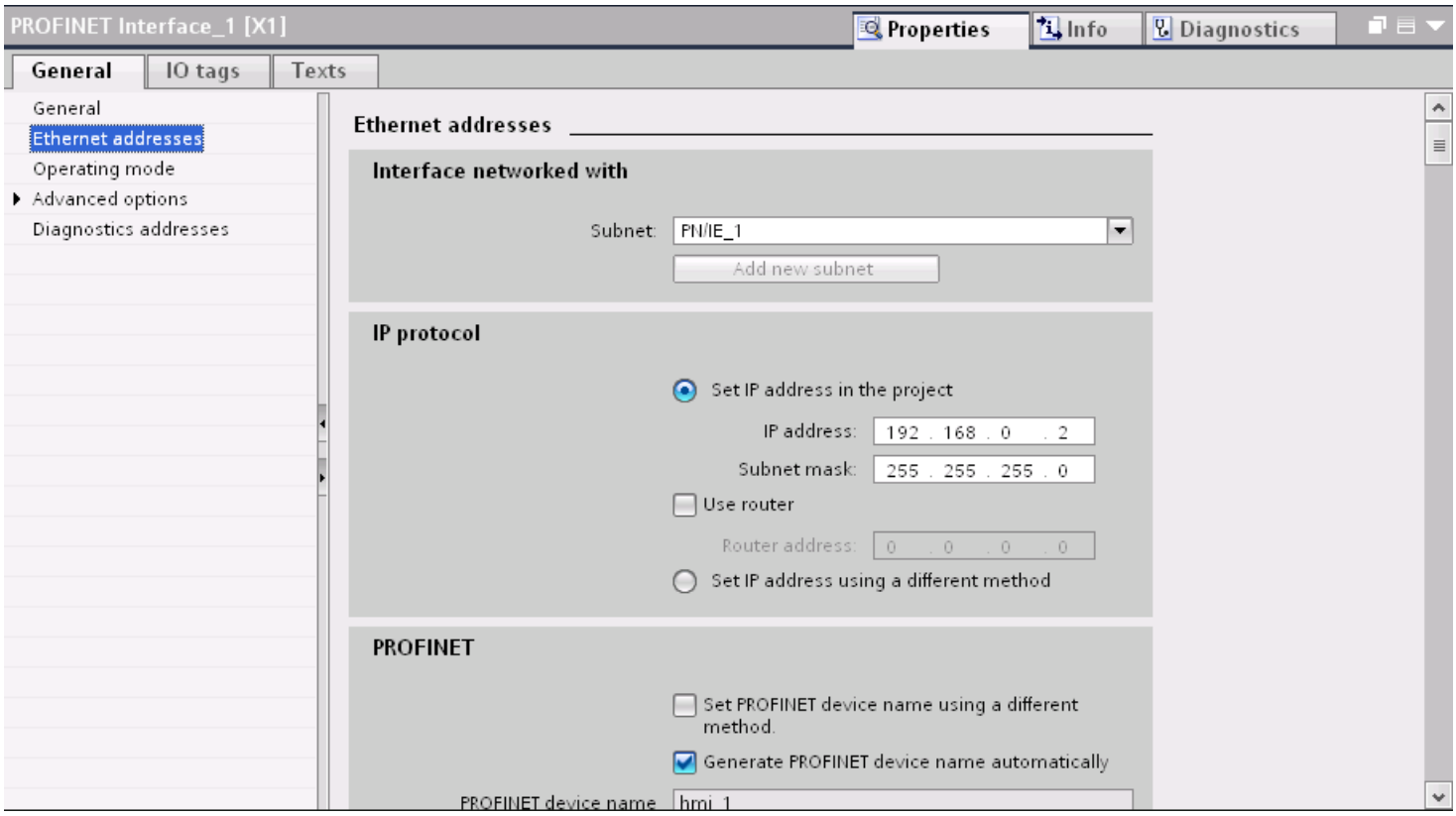
PROFINET parameters for the HMI device

PROFINET parameters for the HMI device

An overview of the configured HMI device parameters can be found in the properties for the HMI device.

Displaying and changing PROFINET parameters of the HMI device

1. Click the HMI device in the "Devices & Networks" editor.
2. Change the parameters of the HMI device in the Inspector window under "Properties > General > General".



"Interface networked with"

In the "Interface networked with" area, select the subnet of the HMI connection via which the HMI device is connected to the network. You use the "Add new subnet" button to create a new subnet.

"IP protocol"

- "Set IP address in the project"
  When you transfer the WinCC project to the HMI device, this IP address is set up directly in the HMI device.

  Note

  The device is automatically restarted in the case of HMI devices with the Windows CE 3.0 operating system.

  HMI devices with Windows CE 3.0:
  - OP 77B
  - TP 177B color PN/DP
  - TP 177B mono DP
  - OP 177B color PN/DP
  - OP 177B mono DP
  - Mobile Panel 177 PN
  - Mobile Panel 177 DP
  - TP 277 6"
  - OP 277 6"

- "Subnet mask"
  You assign data of the subnet mask in the "Subnet mask" area.
- "Use IP router"
  If you are using an IP router, select "Use IP router" and enter the router address in the "Router address" field.
- "Set IP address using a different method"
  If the function "Set IP address using a different method" is activated, the IP address is not taken from the project. You have to enter the IP address directly in the Control Panel of the HMI device.

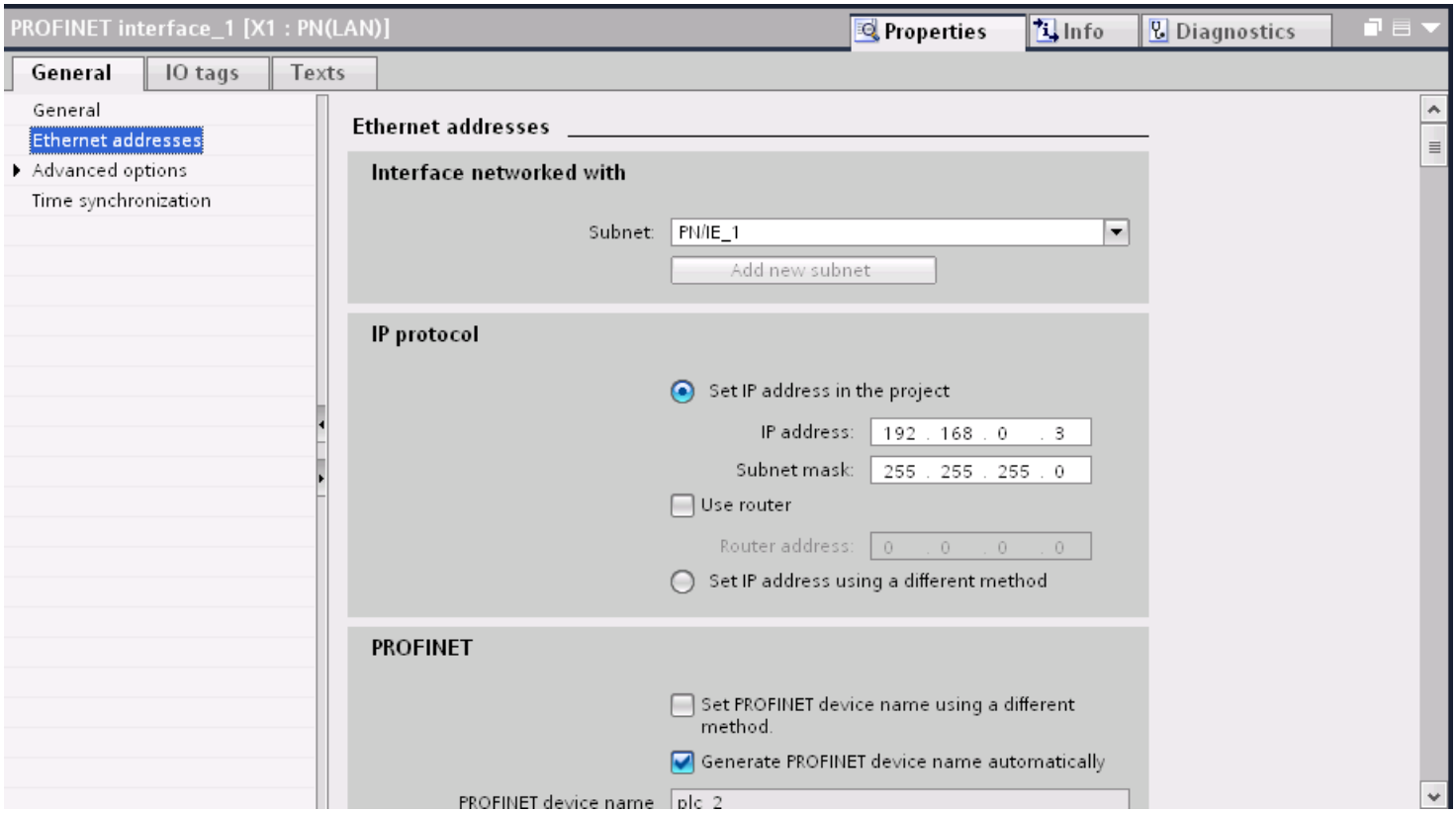
PROFINET parameters for the PLC

PROFINET parameters for the PLC

An overview of the configured parameters can be found in the properties for the PLC.

Displaying and changing PROFINET parameters of the PLC

1. Click the PLC in the "Devices & Networks" editor.
2. Change the parameters of the PLC in the Inspector window under "Properties > General > General".



"Interface networked with"

In the "Subnet" area, select the subnet of the HMI connection via which the PLC is connected to the network. You use the "Add new subnet" button to create a new subnet.

"IP protocol"

- "Interface type"
  Depending on the HMI device type, you have various interfaces to choose from.
- "IP address"
  You assign the IP address of the HMI device in the "IP address" area.
- "Subnet mask"
  You assign data of the subnet mask in the "Subnet mask" area.
  If you are using an IP router, select "Use IP router" and enter the router address in the field.

Configuring Industrial Ethernet

Rules for the network configuration

The Ethernet interfaces of the devices have a default IP address that you can change.

IP address

The IP parameters are visible if the communication-capable devices support the TCP/IP protocol.

The IP address consists of 4 decimal figures in the range of 0 to 255. The decimal figures are separated from one another by a dot.

Example: 140.80.0.2

The IP address consists of the following:

- The address of the (sub) net
- The address of the node (generally also called host or network node)

Subnet mask

The subnet mask splits these two addresses. It determines which part of the IP address addresses the network and which part of the IP address addresses the node.

The set bits of the subnet mask determine the network part of the IP address.

Example:

Subnet mask: 255.255.0.0 = 11111111.11111111.00000000.00000000

In the example given for the above IP address, the subnet mask shown here has the following meaning:

The first 2 bytes of the IP address identify the subnet - i.e. 140.80. The last two bytes address the node, thus 0.2.

It is generally true that:

- The network address results from AND linking the IP address and subnet mask.
- The node address results from AND NOT linking the IP address and subnet mask.

Relation between IP address and default subnet mask

An agreement exists relating to the assignment of IP address ranges and so-called "Default subnet masks". The first decimal number (from the left) in the IP address determines the structure of the default subnet mask. It determines the number of "1" values (binary) as follows:

IP address (decimal)	IP address (binary)	Address class	Default subnet mask
0 to 126	0xxxxxxx.xxxxxxxx....	A	255.0.0.0
128 to 191	10xxxxxx.xxxxxxxx...	B	255.255.0.0
192 to 223	110xxxxx.xxxxxxxx...	C	255.255.255.0

Note

Range of values for the first decimal point

A value of between 224 and 255 is also possible for the first decimal number of the IP address (address class D etc). This is, however, not recommended because there is no address check for these values.

Masking other subnets

You can use the subnet mask to add further structures and form "private" subnets for a subnet that is assigned one of the address classes A, B or C. This is done by setting other lower points of the subnet mask to "1". For each bit set to "1", the number of "private" networks doubles and the number of nodes they contain is halved. Externally, the network functions like an individual network as it did previously.

Example:

You have a subnet of address class B (e.g. IP address 129.80.xxx.xxx) and change the default subnet mask as follows:

Masks	Decimal	Binary
Default subnet mask	255.255.0.0	11111111.11111111.00000000.00000000
Subnet mask	255.255.128.0	11111111.11111111.10000000.00000000

Result:

All nodes with addresses between 129.80.001.xxx and 129.80.127.xxx are on one subnet, all nodes with addresses between 129.80.128.xxx and 129.80.255.xxx are on another subnet.

Router

The job of the routers is to connect the subnets. If an IP datagram is to be sent to another network, it first has to be conveyed to a router. To make this possible, in this case you have to enter the address of the router for each node in the subnet.

The IP address of a node in the subnet and the address of the router may only differ at the points at which there is a "0" in the subnet mask.

Protection of communication

Security levels

If you want to protect the PLC and HMI device communication, you can assign protection levels for the communication.

For a SIMATIC S7-1500 CPU, you can enter multiple passwords and thereby set up different access rights for various user groups.

The passwords are entered in a table, so that exactly one protection level is assigned to each password.

The effect of the password is given in the "Protection" column.

For the SIMATIC S7-1200 controller, several aspects need to be considered when setting protection levels. For additional information on this, see: S7-1200 protection levels

Example

You select the "Complete protection" protection level for a standard CPU (i.e., not an F-CPU) when configuring it.

Afterwards, you enter a separate password for every protection level above it in the table.

For users who do not know any of the passwords, the CPU is completely protected. Not even HMI access is possible.

For users who know one of the assigned passwords, the effect depends on the table row in which the password occurs:

- The password in row 1 (no protection) allows access as if the CPU were completely unprotected. Users who know this password have unrestricted access to the CPU.
- The password in row 2 (write protection) allows access as if the CPU were write-protected. Despite knowing the password, users who know this password only have read access to the CPU.
- The password in row 3 (read and write protection) allows access as if the CPU were read-protected and write-protected, so that only HMI access is possible for users who know this password.

Access password for the HMI connection

Introduction

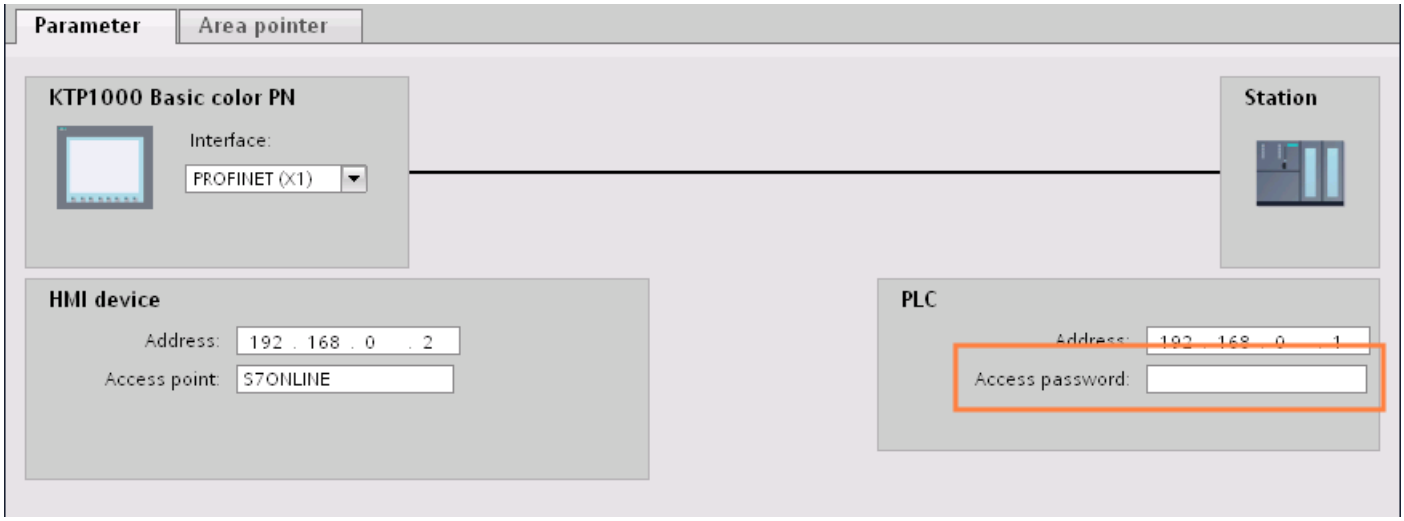
You secure access to a PLC by assigning a password.

You assign the password when you configure the password.

A password is mandatory as of protection level "Complete protection". Communication to the PLC is denied if an incorrect password or no password is entered.

Entering access password

You enter the password for the PLC in the "Connections" editor.



Setting port options

Setting the port options

Changing connection settings for the PROFINET IO port

You can change the network settings for the PROFINET IO port as required. By default, the settings are made automatically. In normal situations, this guarantees problem-free communication.

Possible settings for transmission rate / duplex

Depending on the selected device, you can make the following settings for "Transmission rate / duplex":

- Automatic setting
  Recommended default setting of the port. The transmission settings are automatically "negotiated" with the peer port. The "Enable autonegotiation" option is also enabled as a default, in other words, you can use cross cables or patch cables for the connection.
- TP/ITP at x Mbps full duplex (half duplex)
  Setting of the transmission rate and the full duplex/half duplex mode. The effectiveness depends on the "Enable autonegotiation" setting:
  - Autonegotiation enabled
    You can use both cross cable and patch cable.
  - Autonegotiation disabled
    Make sure that you use the correct cable (patch cable or cross cable)! The port is also monitored with this setting.
- Deactivated
  Depending on the module type, the drop down list box can contain the "- Disabled -" option. This option, for example, allows you to prevent access to an unused port for security reasons. With this setting, diagnostic events are not generated.

"Monitor" option

This option enables or disables port diagnostics. Examples of port diagnostics: The link status is monitored, in other words, the diagnostics are generated during link-down and the system reserve is monitored in the case of fiber optic ports.

Option "Enable autonegotiation "

The autonegotiation setting can only be changed if a concrete medium (for example, TP 100 Mbps full duplex) is selected. Whether or not a concrete medium can be set depends on the properties of the module.

If autonegotiation is disabled, this causes the port to be permanently specified, as for example, is necessary for a prioritized startup of the IO device.

You must make sure the partner port has the same settings because with this option the operating parameters of the connected network are not detected and the data transmission rate and transmission mode can accordingly not be optimally set.

Note

When a local port is connected, STEP 7 makes the setting for the partner port if the partner port supports the setting. If the partner port does not accept the setting, an error message is generated.

Wiring rules for disabled autonegotiation

Requirements

You have made the following settings for the port in question, for example, to accelerate the startup time of the IO device:

- Fixed transmission rate
- Autonegotiation incl. autocrossing disabled

The time for negotiating the transmission rate during startup has been saved.

If you have disabled autonegotiation, you must observe the wiring rules.

Wiring rules for disabled autonegotiation

PROFINET devices have the following two types of ports:

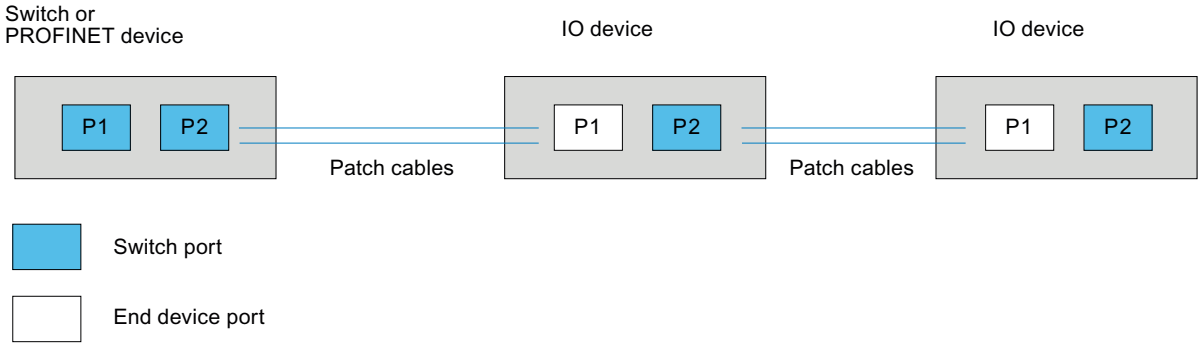
Type of port	PROFINET devices	Note
Switch port with crossed pin assignment	For IO devices: Port 2 For S7 CPUs with 2 ports: Ports 1 and 2	Crossed pin assignment means that the pin assignment for the ports for sending and receiving between the respective PROFINET devices is exchanged internally.
End device port with uncrossed pin assignment	For IO devices: Port 1 For S7 CPUs with one port: Port 1	-

Validity of the wiring rules

The cabling rules described in the following paragraph apply exclusively for the situation in which you have specified a fixed port setting.

Rules for cabling

You can connect several IO devices in line using a single cable type (patch cable). To do this, you connect port 2 of the IO device (distributed I/O) with port 1 of the next IO device. The following graphic gives an example with two IO devices.



Boundaries at the port

Requirements

To use boundaries, the respective device must have more than one port. If the PROFINET does not support boundary settings, they are not shown.

Enable boundaries

"Boundaries" are limits for transmission of certain Ethernet frames. The following boundaries can be set at a port:

- "End of discovery of accessible devices"
  No forwarding of DCP frames to identify accessible devices. Devices downstream from this port cannot be reached by the project tree under "Accessible devices". Devices downstream from this port cannot be reached by the CPU.
- "End of topology discovery"
  LLDP frames (Link Layer Discovery Protocol) are not forwarded for topology detection.
- "End of sync domain"
  No forwarding of sync frames transmitted to synchronize nodes within a sync domain.
  If you operate, for example, a PROFINET device with more than two ports in a ring, you should prevent the sync frame from being fed into the ring by setting a sync boundary (at the ports not inside the ring).
  Additional example: If you want to use several sync domains, configure a sync domain boundary for the port connected to a PROFINET device from the other sync domain.

Restrictions

The following restrictions must be observed:

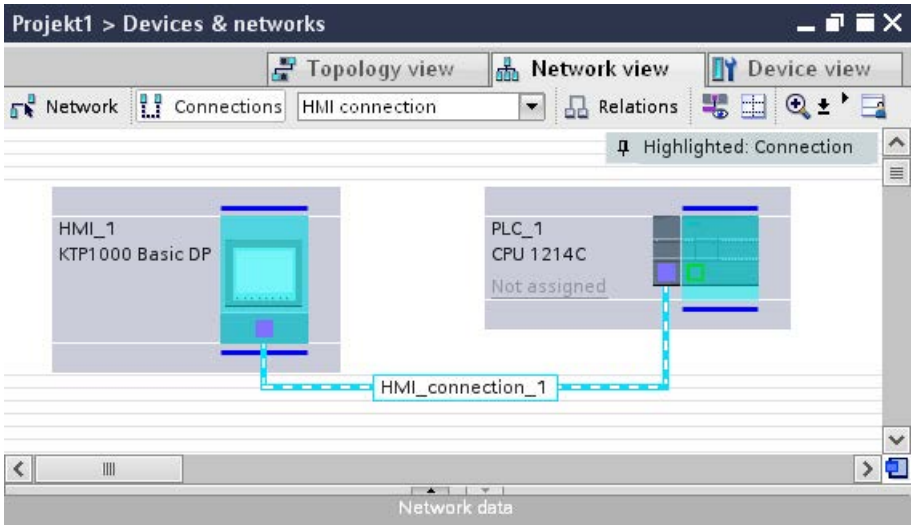
- The individual check boxes can only be used if the port supports the function in question.
- If a partner port has been determined for the port, the following check boxes cannot be used:
  - "End of discovery of accessible devices"
  - "End of topology discovery"
- If autonegotiation is disabled, none of the check boxes can be used.

10.8.7.3 Communication via PROFIBUS

Communication via PROFIBUS

HMI connections via PROFIBUS

If you want to connect a SIMATIC S7 1200 to a HMI device via PROFIBUS, you must configure a PROFIBUS-capable communication module to a slot of the controller first.



HMI connection in the "Devices & Networks" editor

You configure the HMI connection between the PLC and the HMI device via PROFIBUS in the "Devices & Networks" editor.

Connection in the "Connections" editor

Alternatively, you configure the connection between the PLC and HMI device in the "Connections" editor of the HMI device.

Configuring an HMI connection via PROFIBUS

Introduction

You configure an HMI connection between HMI devices and a SIMATIC S7-1200 via PROFIBUS in the "Devices & Networks" editor.

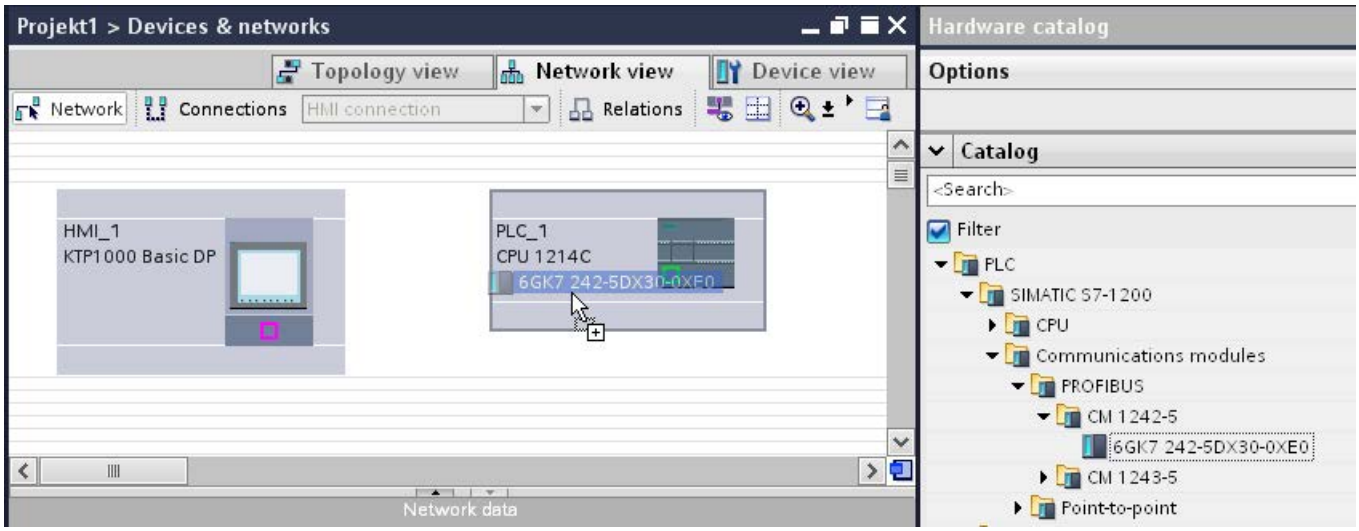
Requirements

The following communication partners are created in the "Devices & Networks" editor:

- HMI device with MPI/DP interface
- SIMATIC S7-1200

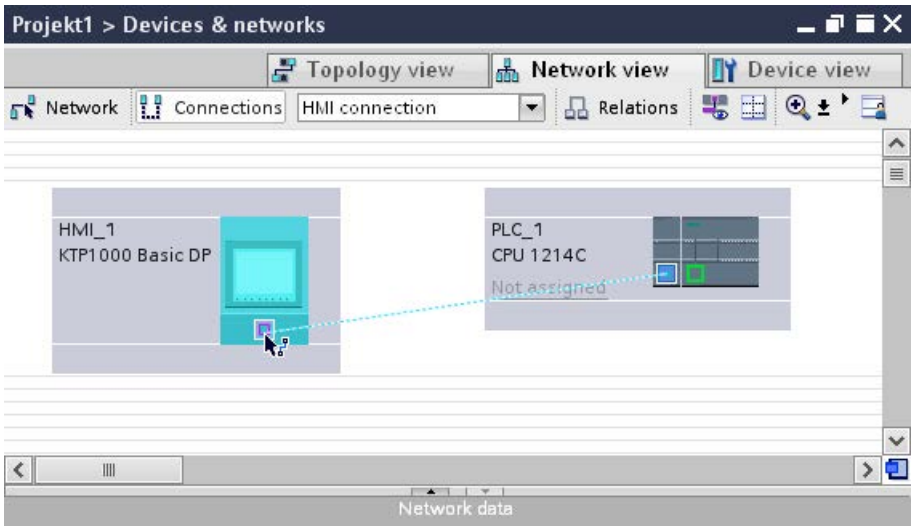
Procedure

1. Double-click the "Devices & Networks" item in the project tree.
   The available communication partners in the project are displayed graphically in the network view.
2. Click the "Connections" button.
   The devices available for connection are highlighted in color.
3. Use a drag-and-drop operation to move a PROFIBUS-capable communication module from the hardware catalog to the PLC.



4. Click the HMI device interface.
5. Select the "PROFIBUS" interface type in the Inspector window under "Properties > General > PROFIBUS address/ MPI address > Parameters".

6. Click the interface of the communication module and use a drag-and-drop operation to draw a connection to the HMI device.



7. Click the name of the connection.
   The connection is displayed graphically in the Inspector window.
8. Click "Highlight HMI connection" and select the HMI connection.
9. Click the communication partners in the "Network view" and change the PROFIBUS parameters in the Inspector window according to the requirements of your project.
   See the section "PROFIBUS parameters (Page 8494)" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. You check the connection parameters in the table.

You can change the local name for the connection only in the table.

Result

You have created an HMI connection between an HMI device and a SIMATIC S7-1200 via PROFIBUS.

PROFIBUS parameters

PROFIBUS parameters for the HMI connection

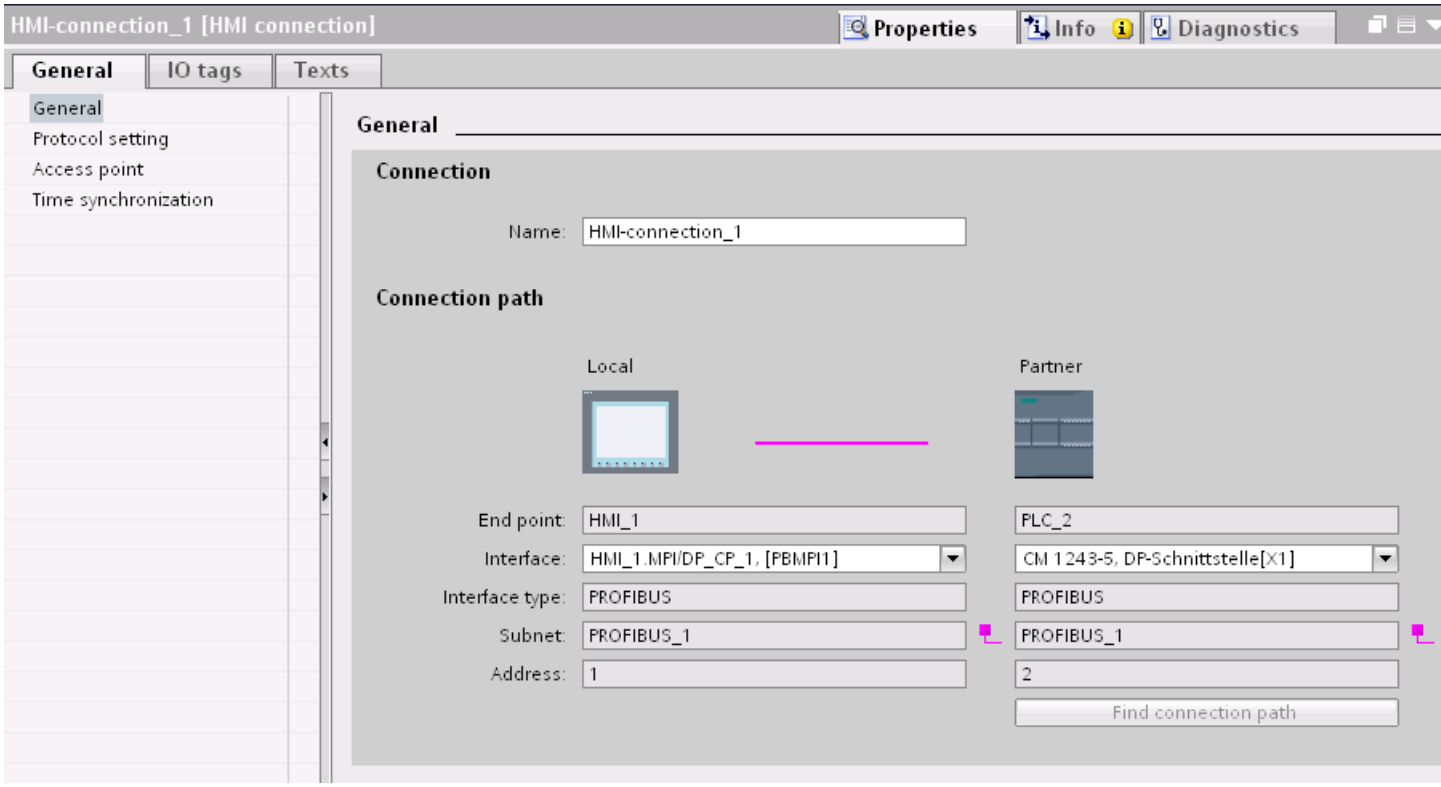
PROFIBUS parameters for the HMI connection

An overview of the configured HMI connection parameters can be found in the properties for the HMI connection.

Only limited changes are possible in this Inspector window.

Displaying and editing HMI connection parameters

1. Click the HMI connection in the "Devices & Networks" editor.
2. Change the parameters of the HMI connection in the Inspector window under "Properties > General > General".



"Connection"

The "Connection" area displays the HMI connection created for communication between the devices.

You can edit the name of the HMI connection in this area.

"Connection path"

The communication partners of the selected HMI connection and the associated PROFIBUS parameters are displayed in the "Connection path" area. Some of the areas displayed cannot be edited in this dialog.

- "End point"
  Displays the device name. This area cannot be edited.
- "Interface"
  Displays the selected interface of the device. You can choose between several interfaces, depending on the device.
- "Interface type"

Displays the selected interface type. This area cannot be edited.

- "Subnet"
  Displays the selected subnet. This area cannot be edited.
- "Address"
  Displays the PROFIBUS address of the device. This area cannot be edited.
- "Find connection path" button
  Enables the subsequent specification of connections.

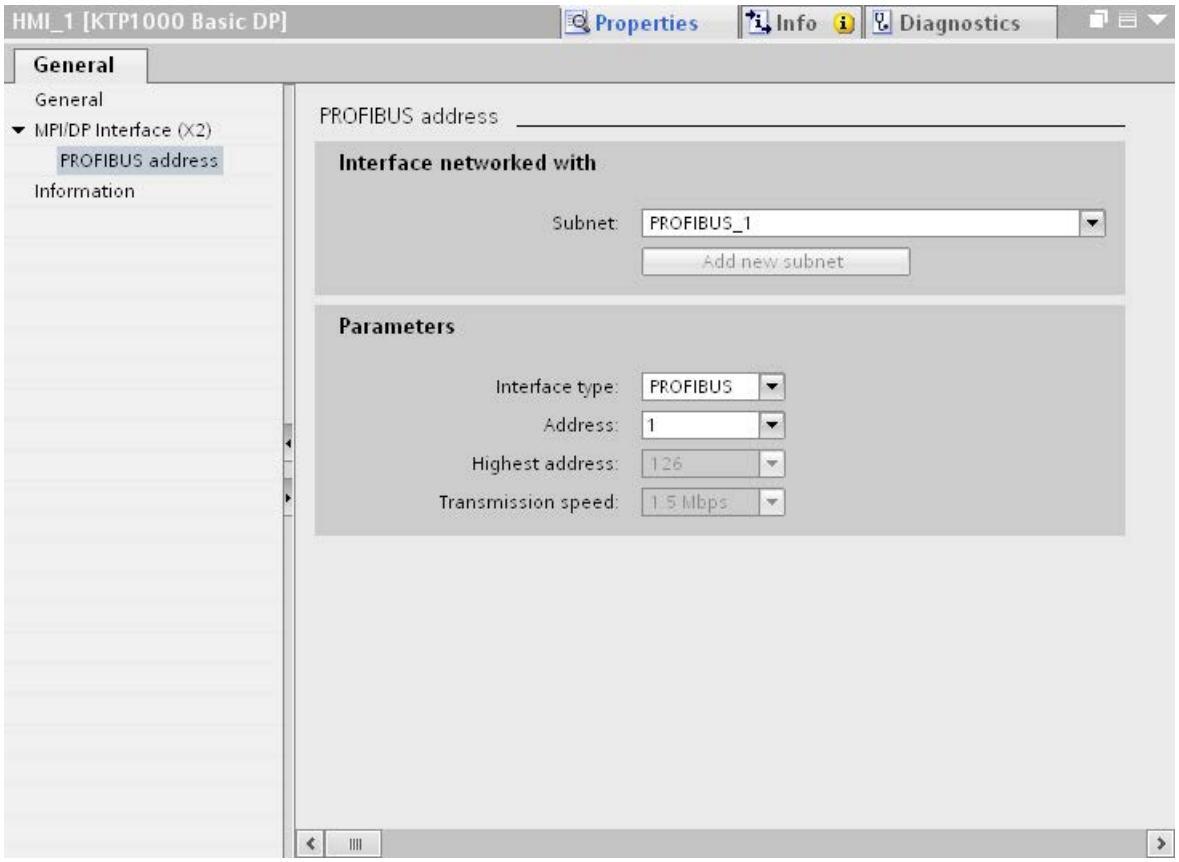
PROFIBUS parameters for the HMI device

PROFIBUS parameters for the HMI device

An overview of the configured HMI device parameters can be found in the properties for the HMI device.

Displaying and changing PROFIBUS parameters of the HMI device

1. Click the HMI device in the "Devices & Networks" editor.
2. Change the parameters of the HMI device in the Inspector window under "Properties > General > General".



"Interface networked with"

In the "Interface networked with" area, select the subnet of the HMI connection via which the HMI device is connected to the network. You use the "Add new subnet" button to create a new subnet.

"Parameters"

- "Interface type"
  Depending on the HMI device type, you have various interfaces to choose from.
- "Address"
  You assign the PROFIBUS address of the HMI device in the "Address" area. The PROFIBUS address must be unique throughout the PROFIBUS network.
- "Highest address"
  The "Highest address" area displays the highest address of the PROFIBUS network.
- "Transmission speed"
  The "Transmission speed" is determined by the slowest device connected to the network. The setting is identical throughout the network.

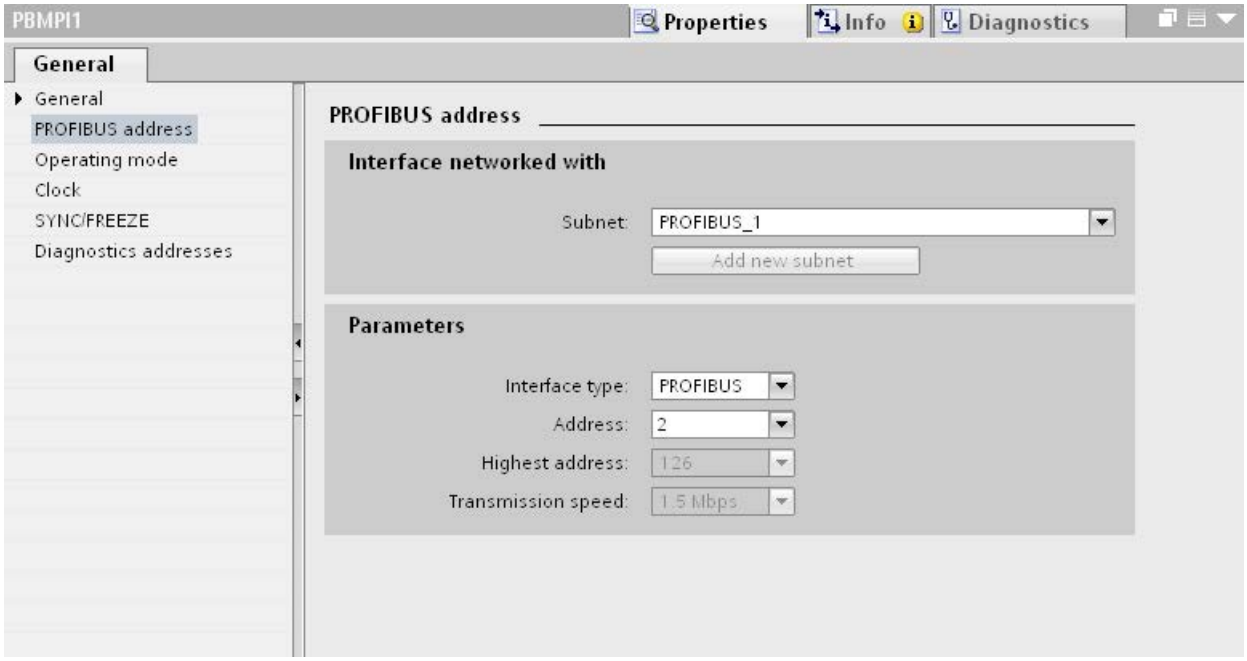
PROFIBUS parameters for the PLC

PROFIBUS parameters for the PLC

An overview of the configured parameters can be found in the properties for the PLC.

Displaying and changing PROFIBUS parameters of the PLC

1. Click the PLC in the "Devices & Networks" editor.
2. Change the parameters of the PLC in the Inspector window under "Properties > General > General".



"Interface networked with"

In the "Subnet" area, select the subnet of the HMI connection via which the PLC is connected to the network. You use the "Add new subnet" button to create a new subnet.

"Parameters"

- "Interface type"
  Depending on the HMI device type, you have various interfaces to choose from.
- "Address"
  You assign the PROFIBUS address of the HMI device in the "Address" area. The PROFIBUS address must be unique throughout the PROFIBUS network.
- "Highest address"
  The "Highest address" area displays the highest address of the PROFIBUS network.
- "Transmission speed"
  The "Transmission speed" is determined by the slowest device connected to the network. The setting is identical throughout the network.

Bus profiles with PROFIBUS

Introduction

Depending on the device types connected and protocols used on the PROFIBUS, different profiles are available. The profiles differ in terms of the setting options and calculation of bus parameters. The profiles are explained below.

Devices with different profiles on the same PROFIBUS subnet

The PROFIBUS subnet only functions without problem if the bus parameters of all devices have the same values.

Profiles and transmission rates

Profiles	Supported transmission speeds in Kbits/s
DP	9,6 19,2 45,45 93,75 187,5 500 1500 3000 6000 12000
Standard	9,6 19,2 45,45 93,75 187,5 500 1500 3000 6000 12000
Universal	9,6 19,2 93,75 187,5 500 1500

Meaning of profiles

Profile	Meaning
DP	Select the "DP" bus profile when the only devices connected to the PROFIBUS subnet are those which satisfy the requirements of standard EN 50170 Volume 2/3, Part 8-2 PROFIBUS. The bus parameter setting is optimized on these devices. This includes devices with DP master and DP slave interfaces of the SIMATIC S7 and distributed I/Os of other manufacturers.
Standard	Compared to the "DP" profile, the "Standard" profile also offers scope for devices of another project or devices which have not been configured here to be taken into account when calculating the bus parameters. The bus parameters are then calculated following a simple, non-optimized algorithm.
Universal	Select the "Universal" bus profile when individual devices on the PROFIBUS subnet use the PROFIBUS-FMS service. This includes the following devices for example: • CP 343-5 • PROFIBUS-FMS devices of other manufacturers As with the "Standard" profile, this profile allows you to take other devices into account when calculating the bus parameters.

Protection of communication

Security levels

If you want to protect the PLC and HMI device communication, you can assign protection levels for the communication.

For a SIMATIC S7-1500 CPU, you can enter multiple passwords and thereby set up different access rights for various user groups.

The passwords are entered in a table, so that exactly one protection level is assigned to each password.

The effect of the password is given in the "Protection" column.

For the SIMATIC S7-1200 controller, several aspects need to be considered when setting protection levels. For additional information on this, see: Auto-Hotspot

Example

You select the "Complete protection" protection level for a standard CPU (i.e., not an F-CPU) when configuring it.

Afterwards, you enter a separate password for every protection level above it in the table.

For users who do not know any of the passwords, the CPU is completely protected. Not even HMI access is possible.

For users who know one of the assigned passwords, the effect depends on the table row in which the password occurs:

- The password in row 1 (no protection) allows access as if the CPU were completely unprotected. Users who know this password have unrestricted access to the CPU.
- The password in row 2 (write protection) allows access as if the CPU were write-protected. Despite knowing the password, users who know this password only have read access to the CPU.
- The password in row 3 (read and write protection) allows access as if the CPU were read-protected and write-protected, so that only HMI access is possible for users who know this password.

Access password for the HMI connection

Introduction

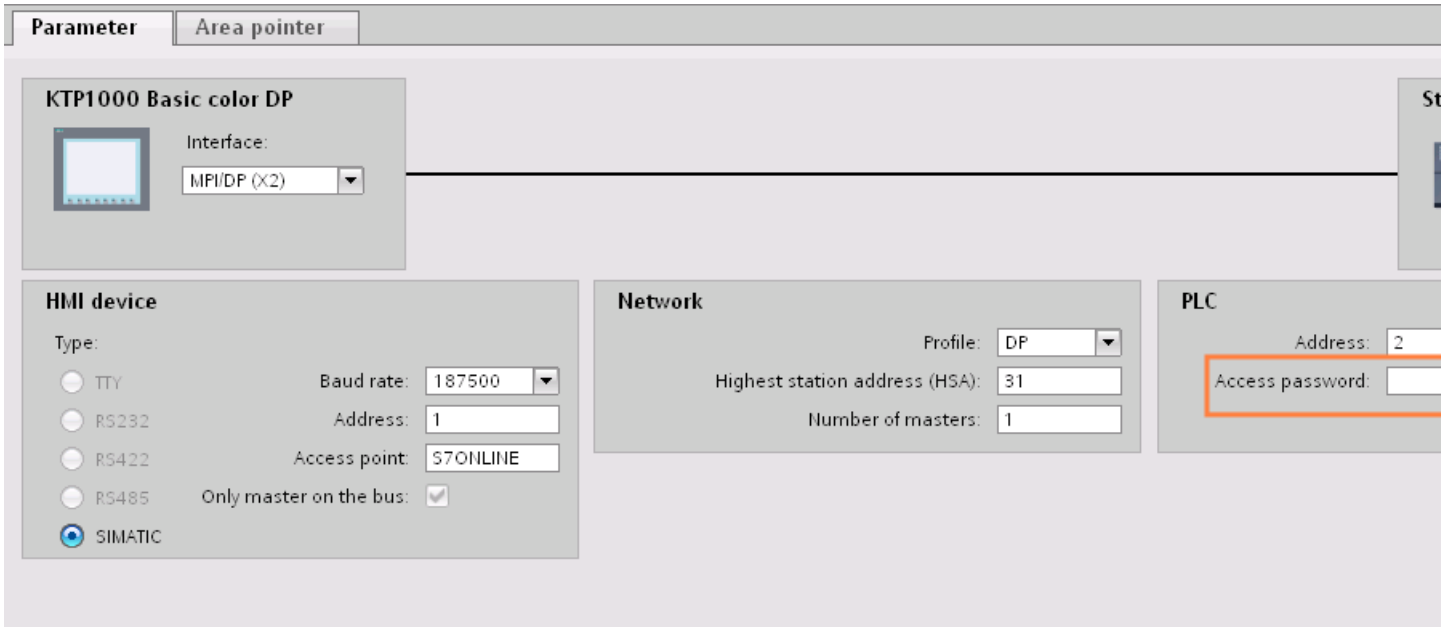
You secure access to a PLC by assigning a password.

You assign the password when you configure the password.

A password is mandatory as of protection level "Complete protection". Communication cannot be established with the PLC if you enter no password or an incorrect password.

Assigning password

You enter the password for the PLC in the "Connections" editor.



10.8.7.4 Data exchange

Data exchange using area pointers

General information on area pointers

Introduction

You use an area pointer to access a data area in the PLC. During communication, the PLC and the HMI device alternately access these data areas for read and write operations.

The PLC and the HMI device trigger defined interactions based on the evaluation of stored data.

Configuration of area pointers

Before you use an area pointer, you enable it under "Connections > Area pointer". You then assign the area pointer parameters.

You can find more detailed information on configuring area pointers in:

Configuration of area pointers

Area pointer "Date/time"

Function

This area pointer is used to transfer the date and time from the HMI device to the PLC.

The PLC writes control job "41" to the job mailbox.

When it evaluates the control job, the HMI device writes its current date and the time in the data area configured in the "Date/time" area pointer.

Note

You cannot use the "Date/Time PLC" area pointer if you have configured the "Date/Time" area pointer.

The "Date/Time" area pointer when used in a project which contains multiple connections must be enabled for each configured connection.

The date/time data area has the following structure:

Data word	Most significant byte								Least significant byte								
	7							0	7							0	
n+0	Reserved								Hour (0 to 23)								Time
n+1	Minute (0 to 59)								Second (0 to 59)								
n+2	Reserved								Reserved								
n+3	Reserved								Weekday (1 to 7, 1=Sunday)								Date
n+4	Day (1 to 31)								Month (1 to 12)								
n+5	Year (80 to 99/0 to 29)								Reserved								

Note

When making entries in the "Year" data area, you should note that values 80 to 99 result in years 1980 through 1999, while the values 0 to 29 result in the years 2000 through 2029.

Permitted data types

You can use the following data types when you configure the "Date/Time" area pointer.

- Word
- UInt
- DTL

Use of the "DTL" data type

Use of data type "DTL" with communication driver S7-1200. A tag of the "DTL" data type has a length of 12 bytes and saves information on date and time in a predefined structure.

The "DTL" data type has the following structure:

Byte	Component	Data type	Value range
0	Year	UINT	1970 to 2554
1			
2	Month	USINT	0 to 12
3	Day	USINT	1 to 31
4	Day of week	USINT	1(Sunday) to 7(Saturday) The weekday is not considered in the value entry.
5	Hour	USINT	0 to 23
6	Minute	USINT	0 to 59
7	Second	USINT	0 to 59
8	Nanoseconds	UDINT	0 to 999 999 999
9			
10			
11			

The "DTL" data type supports time information down to the nanosecond range. Because panels only support time information down to the millisecond, you may encounter the following restriction when using the area pointers:

For the transmission of time information from a panel to the controller, the smallest unit of time is 1 millisecond. The value range from microseconds to nanoseconds of the "DTL" data type is filled with zeros.

"Date/time PLC" area pointer

Function

This area pointer is used to transfer the date and time from the PLC to the HMI device. Use this area pointer if the PLC is the time master.

The PLC loads the data area of the area pointer.

The HMI device reads the data cyclically within the configured acquisition cycle and synchronizes itself.

Note

Set an acquisition cycle of sufficient length for the date/time area pointer PLC to avoid any negative impact on HMI device performance.
Recommended: Acquisition cycle of 1 minute, if the process allows this.

"Date/Time PLC" is a global area pointer and may be configured only once per project.

Note

You cannot use the "Date/Time" area pointer if you have configured the "Date/Time PLC" area pointer.

The date/time data area has the following structure:

DATE_AND_TIME format (in BCD code)

Data word	Most significant byte			Least significant byte		
	7	. . . . .	0	7	. . . . .	0
n+0	Year (80 to 99/0 to 29)			Month (1 to 12)		
n+1	Day (1 to 31)			Hour (0 to 23)		
n+2	Minute (0 to 59)			Second (0 to 59)		
n+3	Reserved			Reserved		Weekday (1 to 7, 1=Sunday)
n+4 1)	Reserved			Reserved		
n+5 1)	Reserved			Reserved		

1) The two data words must exist in the data area to ensure that the data format matches WinCC flexible and to avoid reading false information.

Note

When making entries in the "Year" data area, you should note that values 80 to 99 result in years 1980 through 1999, while the values 0 to 29 result in the years 2000 through 2029.

Permitted data types

You can use the following data types when you configure the "Date/Time PLC" area pointer:

- DTL

Use of the "DTL" data type

Use of data type "DTL" with communication driver S7-1200. A tag of the "DTL" data type has a length of 12 bytes and saves information on date and time in a predefined structure.

The "DTL" data type has the following structure:

Byte	Component	Data type	Value range
0	Year	UINT	1970 to 2554
1			
2	Month	USINT	0 to 12
3	Day	USINT	1 to 31
4	Day of week	USINT	1(Sunday) to 7(Saturday) The weekday is not considered in the value entry.
5	Hour	USINT	0 to 23
6	Minute	USINT	0 to 59
7	Second	USINT	0 to 59
8	Nanoseconds	UDINT	0 to 999 999 999
9			
10			
11			

The HMI devices do not support the use of nanoseconds. Values in the nanosecond range will be ignored during processing in Runtime.

The "DTL" data type supports time information down to the nanosecond range. Because panels only support time information down to the millisecond, you may encounter the following restriction when using the area pointers:

For the transmission of time information from a controller to a panel, the range from microseconds to nanoseconds is ignored. The time information is processed on the panel down to milliseconds.

Area pointer "Coordination"

Function

The "Coordination" area pointer is used to implement the following functions:

- Detecting the startup of the HMI device in the control program
- Detecting the current operating mode of the HMI device in the control program
- Detecting whether the HMI device is ready to communicate in the control program

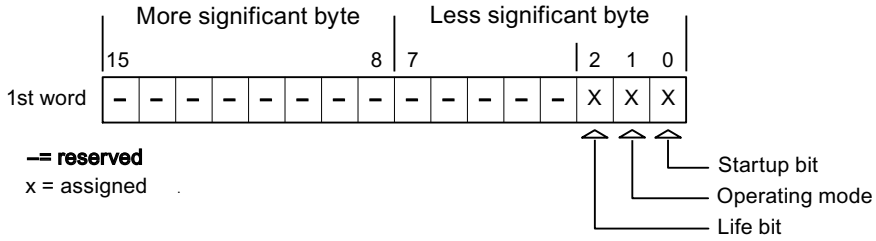
By default, the "Coordination" area pointer has the length of one word and cannot be changed.

Usage

Note

The HMI device always writes the entire coordination area when updating the area pointer. The control program can for this reason not make changes to the coordination area.

Assignment of the bits in the "Coordination" area pointer



Startup bit

The startup bit is set briefly to "0" by the HMI device during startup. It sets the bit permanently to "1" when startup is completed.

Operating mode

The operating mode bit is set to 1 as soon as the user switches the HMI device offline. The status of the operating mode bit is "0" during normal operation of the HMI device. You can determine the current operating mode of the HMI device by reading this bit in the control program.

Life bit

The HMI device inverts the life bit at intervals of approximately one second. You can check whether or not there is still a connection to the HMI device by querying this bit in the control program.

Processing in the PLC

For a simpler evaluation in the PLC program, use a Bool array for this area pointer when using the SIMATIC S7 1200 communication driver. You will have to map the complete 16-bit word of the area pointer. Configure a tag of the data type "Array [0 .. 15] of bool" for this purpose.

Permitted data types

You can use the following data types when you configure the "Coordination" area pointer.

- Word
- UInt
- Bool

Area pointer "Screen number"

Function

The HMI devices store information about the screen called up on the HMI device in the "Screen number" area pointer.

This allows the transfer of the current screen contents from the HMI device to the PLC. Certain reactions can be triggered in the PLC, such as the call of a different screen.

Use

Before the "Screen number" area pointer can be used, it must be set up and activated by selecting "Communication ▸ Area pointer". You can create only **one** instance of the "Screen number" area pointer and only on **one** PLC.

The screen number is always transferred to the PLC when a new screen is activated or when the focus within a screen changes from one screen object to another.

Structure

The area pointer is a data area in the memory of the PLC with a fixed length of 5 words.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1. Word	Current screen type															
2. Word	Current screen number															
3. Word	Reserved															
4th word	Current field number															
5. Word	Reserved															

- Current screen type
  "1" for root screen or
  "4" for permanent window
- Current screen number
  1 to 32767
- Current field number
  1 to 32767

Note

Device dependency

Permanent windows are not available on Basic Panels.

Permitted data types

You can use the following data types when you configure the "Screen number" area pointer.

- Word
- UInt

Area pointer "Project ID"

Function

When Runtime starts, a check can be carried out as to whether the HMI device is connected to the correct PLC. This check is important when operating with several HMI devices.

For this, the HMI device compares a value stored on the PLC with the value specified in configuration. This ensures compatibility of the configuration data with the PLC program.

A missing compatibility results in a corresponding alarm and Runtime will not be started.

Use

Note

HMI connections cannot be switched "online".

The HMI connection in which the "Project ID" area pointer is used must be switched "online".

To use this area pointer, set up the following during the configuration:

- Define the version of the configuration. Values between 1 and 255 are possible.
  You enter the version in the editor "Runtime settings > General" in the "Identification" area.
- This is where you select the PLC tag or the tag array that you have configured as the data area for the area pointer.

Connection failure

A connection failure to a device on which the "project ID" area pointer is configured results in all the other connections of the device being switched to "offline".

This behavior has the following requirements:

- You have configured several connections in a project.
- You are using the "project ID" area pointer in at least one connection.

Causes which may set connections "offline":

- The PLC is not available.
- The connection has been switched offline in the engineering system.

Permitted data types

You can use the following data types when you configure the "Project ID" area pointer.

- Word
- UInt

Area pointer "Job mailbox"

Function

The PLC can use the job mailbox to transfer jobs to the HMI device to trigger corresponding actions on the HMI device. These functions include, for example:

- Display screen
- Set date and time

Data structure

The first word of the job mailbox contains the job number. Depending on the job mailbox, up to three parameters can be transferred.

Word	Most significant byte	Least significant byte
n+0	0	Job number
n+1	Parameter 1	
n+2	Parameter 2	
n+3	Parameter 3	

The HMI device evaluates the job mailbox if the first word of this job is not equal to zero. This means that the parameters must be entered in the job mailbox first, followed by the job number.

When the HMI device accepts the job mailbox, the first word is set to 0 again. The execution of the job mailbox is generally not completed at this point in time.

Job mailboxes

All job mailboxes and their parameters are listed below. The "No." column contains the job number of the job mailbox. Job mailboxes can only be triggered by the PLC when the HMI device is online.

No.	Function	
14	Set time (BCD-coded)	
	Parameter 1	Left byte: - Right byte: hours (0-23)
	Parameter 2	Left byte: minutes (0-59) Right byte: seconds (0-59)
	Parameter 3	-

No.	Function	
14	Set time (BCD-coded)	
15	Set date (BCD-coded)	
	Parameter 1	Left byte: - Right byte: weekday (1-7: Sunday-Saturday)
	Parameter 2	Left byte: day (1-31) Right byte: month (1-12)
	Parameter 3	Left byte: year
23	User logon	
	Logs the user on with the name "PLC user" at the HMI device with the group number transferred in Parameter 1. The logon is possible only when the transferred group number exists in the project.	
	Parameter 1	Group number 1 to 255
	Parameter 2, 3	-
24	User logoff	
	Logs off the current user. (The function corresponds to the "logoff" system function)	
	Parameter 1, 2, 3	-
40	Transfer date/time to PLC	
	An interval of at least 5 seconds must be maintained between two successive jobs to prevent overload of the HMI device.	
	Parameter 1, 2, 3	-
41	Transfer date/time to PLC	
	An interval of at least 5 seconds must be maintained between two successive jobs to prevent overload of the HMI device.	
	Parameter 1, 2, 3	-
46	Update tag	
	Causes the HMI device to read the current value of the tags from the PLC whose update ID matches the value transferred in Parameter 1. (Function corresponds to the "UpdateTag" system function.)	
	Parameter 1	1 - 100
49	Clear event buffer	
	Deletes all analog alarms and discrete alarms of the "Warnings" class from the alarm buffer.	
	Parameter 1, 2, 3	-
50	Clear error alarm buffer	
	Deletes all analog alarms and discrete alarms of the "Errors" class from the alarm buffer.	
	Parameter 1, 2, 3	-
51	Display selection	
	Parameter 1	Screen number
	Parameter 2	-
	Parameter 3	Field number
69	Reading data record from PLC [1)]	
	Parameter 1	Recipe number (1-999)
	Parameter 2	Data record number (1-65535)

No.	Function	
14	Set time (BCD-coded)	
	Parameter 3	0: Do not overwrite existing data record 1: Overwrite existing data record
70	Writing data record to PLC [1]	
	Parameter 1	Recipe number (1-999)
	Parameter 2	Data record number (1-65535)
	Parameter 3	-

[1]	Only devices supporting recipes
[2]	OP 73, OP 77A and TP 177A HMI devices also execute the "Screen selection" job mailbox if the on-screen keyboard is active.
[3]	The weekday is ignored on HMI device KTP 600 BASIC PN.

Permitted data types

You can use the following data types when you configure the "Screen number" area pointer:

- Word
- UInt

"Data record" area pointer

"Data mailbox" area pointer

Function

When data records are transferred between the HMI device and PLC, both partners access common communications areas on the PLC.

Data transfer types

There are two ways of transferring data records between the HMI device and PLC:

- Transfer without synchronization
- Transfer with synchronization over the data mailbox

Data records are always transferred directly. That is, the tag values are read from an address or written to an address configured for this tag directly, without redirecting the values by means of interim memory.

Initiating the transfer of data records

There are three ways of triggering the transfer:

- Operator input in the recipe view
- Job mailboxes
  The transfer of data records can also be triggered by the PLC.
- Triggering by configured functions

If the transfer of data records is triggered by a job mailbox, the data in the recipe view will be updated as well. Avoid operating the recipe view while job mailboxes for transfer of data records are being triggered. If you have already started editing a data record and a job mailbox is triggered for transfer of data records, then this job mailbox will be rejected.

Permitted data types

You can use the following data types when you configure the "Data record" area pointer.

- Word
- UInt

Transfer without synchronization

If you select asynchronous transfer of data records between the HMI device and PLC, there is no coordination over the common data areas. It is therefore unnecessary to set up a data area during configuration.

Asynchronous data record transfer can be a useful alternative, for example, when:

- The system is capable of excluding the risk of uncontrolled overwriting of data by the communication peer.
- The PLC does not require information about the recipe number and data record number.
- The transfer of data records is triggered by the operator of the HMI device.

Reading values

When a read job is triggered, the values are read from the PLC addresses and transferred to the HMI device.

- Triggering by the operator in the recipe view:
  The values are downloaded to the HMI device. You can then process, edit, or save these values, for example.
- Triggering by a function or job mailbox:
  The values are saved immediately to the data volume.

Writing values

When a write job is triggered, the values are written to the PLC addresses.

- Triggering by the operator in the recipe view:
  The current values are written to the PLC.
- Triggering by a function or job mailbox:
  The current values are written to the PLC from the data medium.

Sequence of a transfer started by the operator in the recipe display

Reading from the PLC started by the operator in the recipe view

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe number to be read and the status "Transferring" in the data mailbox and sets the data record number to 0.	Abort with system alarm.
3	The HMI device reads the values from the PLC and displays them in the recipe view. If the recipes have synchronized tags, the values from the PLC are also written to the tags.	
4	The HMI device sets the status "Transfer completed."	
5	The control program must reset the status word to zero in order to enable further transfers.	

Writing to the PLC started by the operator in the recipe view

Step	Action	
	Check: Status word = 0?	
1	Yes	No
	The HMI device enters the recipe and data record number to be written and the status "Transferring" in the data mailbox.	Abort with system alarm.
2	The HMI device writes the current values to the PLC. If the recipes have synchronized tags, the changed values are synchronized between the recipe view and tags and then written to the PLC.	
3	The HMI device sets the status "Transfer completed."	
4	If required, the control program can now evaluate the transferred data.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Note

The status word may only be set by the HMI device. The PLC may only reset the status word to zero.

Note

The PLC may only evaluate the recipe and data record numbers when data inconsistency is detected if one of the conditions outlined below has been met:

- The data mailbox status is set to "Transfer completed".
- The data mailbox status is set to "Transfer completed with error".

Sequence of the transfer triggered by a job mailbox

The transfer of data records between the HMI device and the PLC can be initiated by either one of these stations.

The two job mailboxes No. 69 and No. 70 are available for this type of transfer.

No. 69: Read data record from PLC ("PLC → DAT")

Job mailbox no. 69 transfers data mailboxes from the PLC to the HMI device. The job mailbox is structured as follows:

	Most significant byte	Least significant byte
Word 1	0	69
Word 2	Recipe number (1-999)	
Word 3	Data record number (1-65,535)	
Word 4	Do not overwrite existing data record: 0 Overwrite existing data record: 1	

No. 70: Write data record to PLC ("DAT → PLC")

Job mailbox no. 70 transfers data mailboxes from the HMI device to the PLC. The job mailbox is structured as follows:

	Most significant byte	Least significant byte
Word 1	0	70
Word 2	Recipe number (1-999)	
Word 3	Data record number (1-65,535)	
Word 4	—	

Sequence when reading from the PLC with job mailbox "PLC → DAT" (no. 69)

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the job and the status "Transferring" in the data mailbox.	Abort without return message.
3	The HMI device reads the values and stores the values in the data record specified in the job mailbox.	
4	• If "Overwrite" was selected in the job, an existing data record is overwritten without any prompt for confirmation. The HMI device sets the status "Transfer completed". • If "Do not overwrite" was selected in the job, and the data record already exists, the HMI device aborts the job and enters 0000 1100 in the status word of the data mailbox.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Sequence writing to the PLC with job mailbox "DAT → PLC" (no. 70)

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the job and the status "Transferring" in the data mailbox.	Abort without return message.
3	The HMI device fetches the values of the data record specified in the job from the data medium and writes the values to the PLC.	
4	The HMI device sets the status "Transfer completed."	
5	The PLC program can now evaluate the transferred data. To allow further transfers, the PLC program must set the status word to 0 again.	

Sequence of the transfer when triggered by a configured function

Reading from the PLC using a configured function

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the function and the status "Transferring" in the data mailbox.	Abort with system alarm.
3	The HMI device reads the values from the PLC and stores them in the data record specified in the function.	

Step	Action	
4	• If "Yes" was selected for the "Overwrite" function, an existing data record is overwritten without any prompt for confirmation. The HMI device sets the status "Transfer completed." • If "No" was selected for the "Overwrite" function and the data record already exists, the HMI device aborts the job and enters 0000 1100 in the status word of the data mailbox.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Writing to the PLC by means of configured function

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the function and the status "Transferring" in the data mailbox.	Abort with system alarm.
3	The HMI device fetches the values of the data record specified in the function from the data medium and transfers the values to the PLC.	
4	The HMI device sets the status "Transfer completed."	
5	The control program can now evaluate the transferred data. The control program must reset the status word to zero in order to enable further transfers.	

Possible causes of error when transferring data records

Possible causes of error

The section below shows possible error causes which lead to the cancellation of data record transfer:

- Tag address not set up on the PLC
- Overwriting data records not possible

- Recipe number does not exist
- Data record number does not exist

Note

The status word may only be set by the HMI device. The PLC may only reset the status word to zero.

Note

The PLC may only evaluate the recipe and data record numbers when data inconsistency is detected if one of the conditions outlined below has been met:

- The data mailbox status is set to "Transfer completed".
- The data mailbox status is set to "Transfer completed with error".

Reaction to an aborted transfer due to errors

If the transfer of data records is aborted due to errors, the HMI device reacts as follows:

- Triggering by the operator in the recipe view
  Information in the status bar of the recipe view and output of system alarms
- Triggered by function
  Output of system alarms
- Triggering by job mailbox
  No return message on the HMI device

You can nonetheless evaluate the status of the transfer by querying the status word in the data mailbox.

Note
Availability for specific devices

Notes in the status bar of the recipe view are not available in Basic Panels.

Trends

Trends

A trend is the graphical representation of one or more values from the PLC. The value is read out time-triggered for Basic Panels.

For additional information see:

Configuring trend displays for values from the PLC (Page 8147)

Time-triggered trends

The HMI device reads in the trend values cyclically at an interval specified in the configuration.

Time-triggered trends are suitable for continuous curves, such as the operating temperature of a motor.

Alarms

Configuring alarms

Configure alarms

Several steps are needed to configure alarms, such as operational messages, error alarms, and acknowledgement.

- Step 1: Create tags
- Step 2: Configure alarms
- Step 3: Configure acknowledgment

You can find additional information in the section:

Working with alarms (Page 8157)

Distinctive features when configuring alarms

If you are configuring connections of HMI devices to PLCs of other manufacturers, note the following distinctive features when configuring:

- Data types of the tags
- Addressing of tags
- How the bit positions are counted

Data types

For connections with a SIMATIC communication driver, the following data types are supported:

PLC	Permitted data types	
	Discrete alarms	Analog alarms
SIMATIC S7 PLCs	WORD, INT	BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, TIMER

How the bit positions are counted

For connections with a SIMATIC communication driver, the following counting method applies:

How the bit positions are counted	Byte 0								Byte 1							
	Most significant byte								Least significant byte							
In SIMATIC S7 PLCs	7							0	7							0
In WinCC you configure:	15							8	7							0

Acknowledgment of alarms

Procedure

Create suitable tags on the PLC to acknowledge an error alarm. You assign these tags to an alarm in the "Bit messages" editor. You make the assignment in "Properties > Acknowledgment".

Distinction in terms of acknowledgment:

- Acknowledgment by the PLC
- Acknowledgment on the HMI device

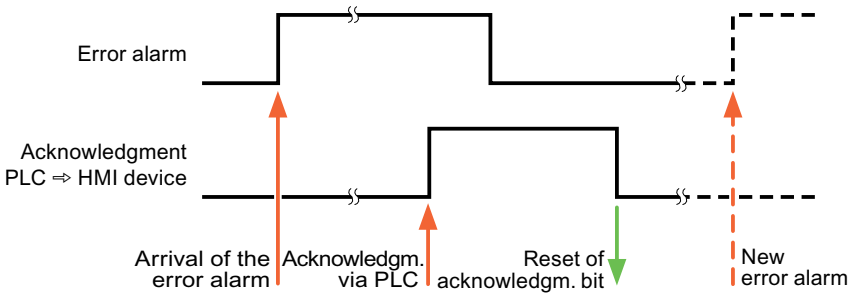
Acknowledgment by the PLC

In "PLC acknowledgment tag", configure the tag or array tag and the bit number that the HMI device uses to identify a PLC acknowledgment.

A bit set in the tag triggers acknowledgment of the assigned error alarm bit at the HMI device. This tag bit returns a function similar to acknowledgment on the HMI device which is triggered by pressing the "ACK" button, for example.

The acknowledgment bit must be located in the same tag as the bit for the error alarm.

Reset the acknowledgment bit before setting the bit in the alarm area again. The figure below shows the pulse diagram.



Acknowledgment on the HMI device

In the "HMI acknowledgment tag" area, configure the tag or array tag as well as the bit number that the HMI device writes to the PLC after acknowledgment. Make sure when you use an array tag that it is not longer than 6 words.

To always create a signal change when setting an assigned acknowledgment bit of a discrete alarm that must be acknowledged, the HMI device will reset the acknowledgment bit assigned to the alarm as soon as it detects an alarm subject to acknowledgment and write the acknowledgment tag in the PLC. There will be a certain delay between detecting the message and writing the acknowledgment tag in the PLC because the HMI device has to process the operations.

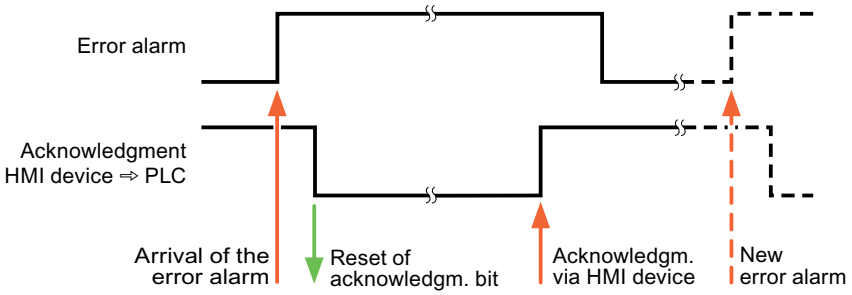
If a discrete alarm subject to acknowledgment is acknowledged by the HMI device, then the corresponding bit in the assigned acknowledgment tag will be set. The entire acknowledgment tag is then written to the PLC by the HMI device. This allows the PLC to recognize that a certain alarm message has been acknowledged at the HMI device.

Note

All alarm bits acknowledged since the last Runtime start will remain in the acknowledgment tag until a new incoming of the respective discrete alarms is detected.

This area should only be read by the PLC because the entire section of the HMI device will be overwritten once the next acknowledgment tag is written.

The figure below shows the pulse diagram.



10.8.7.5 Performance features of communication

S7-1200 device dependency

Communication with the SIMATIC S7-1200 controller

If you use devices from an earlier version of the TIA Portal with TIA Portal V12 SP1, it may not be possible to configure integrated connections to certain HMI devices.

Basic Panels V11.0

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)
KP300 Basic	Yes	Yes	Yes	Yes	Yes
KP400 Basic	Yes	Yes	Yes	Yes	Yes
KTP400 Basic PN	Yes	Yes	Yes	Yes	Yes
KTP600 Basic DP	Yes	Yes	Yes	Yes	Yes
KTP600 Basic PN	Yes	Yes	Yes	Yes	Yes
KTP1000 Basic DP	Yes	Yes	Yes	Yes	Yes
KTP1000 Basic PN	Yes	Yes	Yes	Yes	Yes
TP1500 Basic PN	Yes	Yes	Yes	Yes	Yes

Basic Panels V12.0

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)
KP300 Basic	No	Yes	Yes	Yes	Yes
KP400 Basic	No	Yes	Yes	Yes	Yes
KTP400 Basic PN	No	Yes	Yes	Yes	Yes
KTP600 Basic DP	No	Yes	Yes	Yes	Yes
KTP600 Basic PN	No	Yes	Yes	Yes	Yes
KTP1000 Basic DP	No	Yes	Yes	Yes	Yes
KTP1000 Basic PN	No	Yes	Yes	Yes	Yes
TP1500 Basic PN	No	Yes	Yes	Yes	Yes

Basic Panels V13.0

HMI devices	SIMATIC S7-1200 (V1)	SIMATIC S7-1200 (V2)	SIMATIC S7-1200 (V2.2)	SIMATIC S7-1200 (V3)	SIMATIC S7-1200 (V4)
KTP700 Basic PN	No	Yes	Yes	Yes	Yes
KTP900 Basic PN	No	Yes	Yes	Yes	Yes

Valid data types for SIMATIC S7-1200

Valid data types for connections with SIMATIC S7-1200

The table lists the data types that can be used when configuring tags and area pointers.

Data type	Length
BOOL	1 bit
SINT	1 byte
INT	2 bytes
DINT	4 bytes
USINT	1 byte
UINT	2 bytes
UDINT	4 bytes
REAL	4 bytes
LREAL	8 bytes
TIME	4 bytes
DATE	2 bytes
DTL	12 bytes
TIME_OF_DAY, TOD	4 bytes
STRING	(2+n) bytes, n = 0 to 254
CHAR	1 byte
Array of CHAR	--
BYTE	1 byte
WORD	2 bytes
DWORD	4 bytes

10.8.7.6 Creating connections in the "Connections" editor

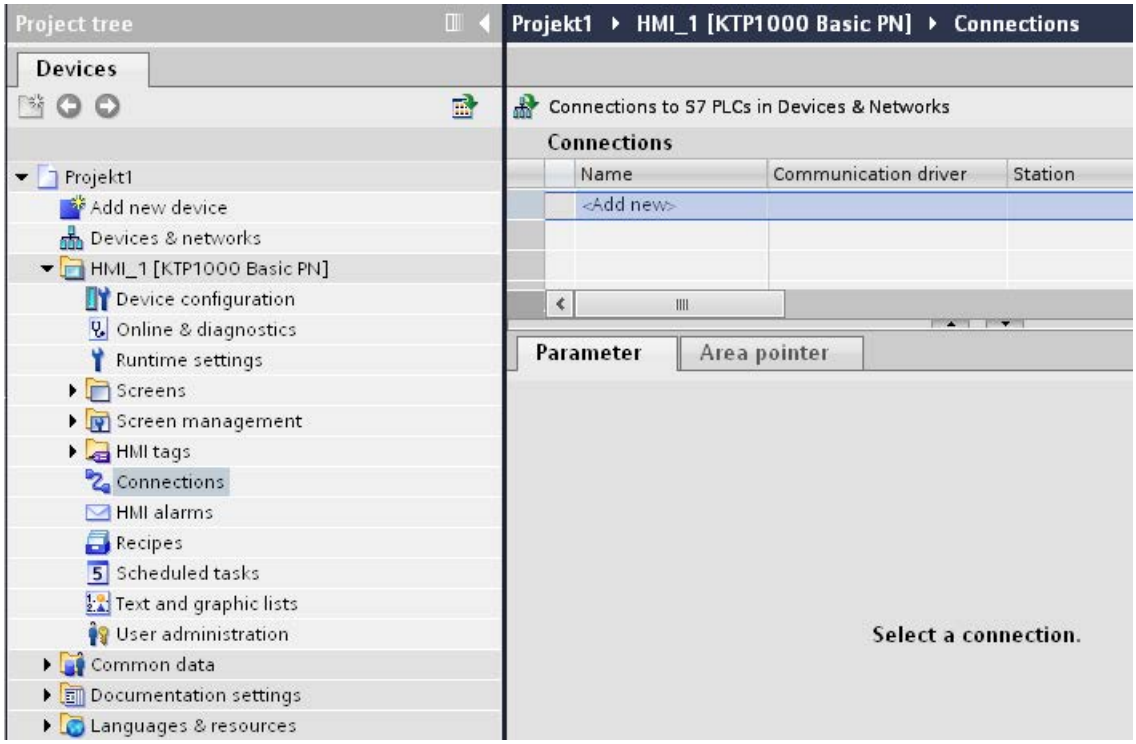
Creating a PROFINET connection

Requirements

- A project is open.
- An HMI device with a PROFINET interface has been created.

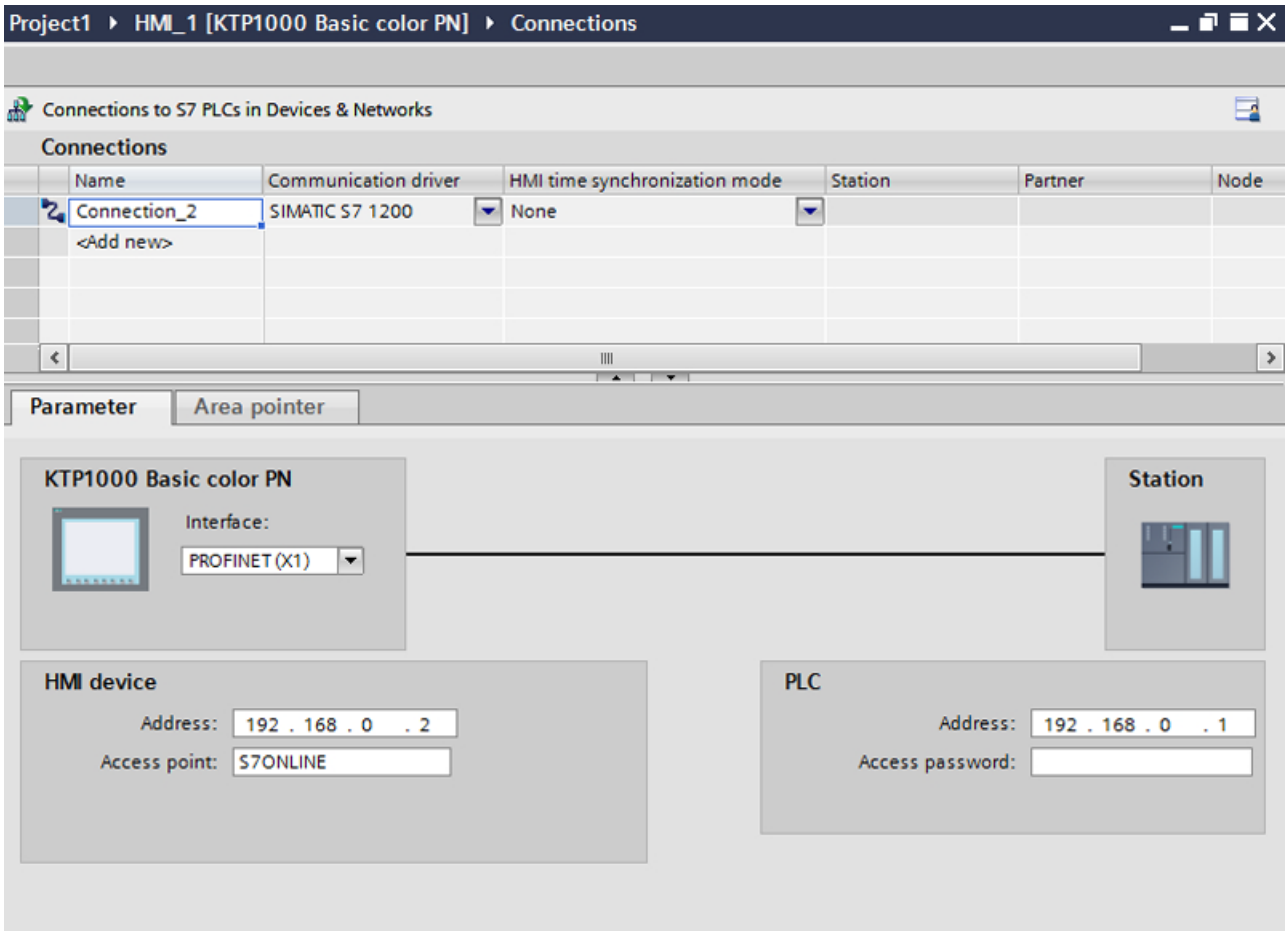
Procedure

1. Open the "Connections" editor of the HMI device.
2. Double-click "<Add>".



3. In the "Communication drivers" column, select the "SIMATIC S7 1200" driver.
4. Click the name of the connection.

5. Select a PROFINET interface of the HMI device in the Inspector window under "Parameters > Interface".



6. Set the IP addresses of the communication partners in the Inspector window:
   - HMI device: "Parameters > HMI device > Address"
   - PLC: "Parameters > PLC > Address"

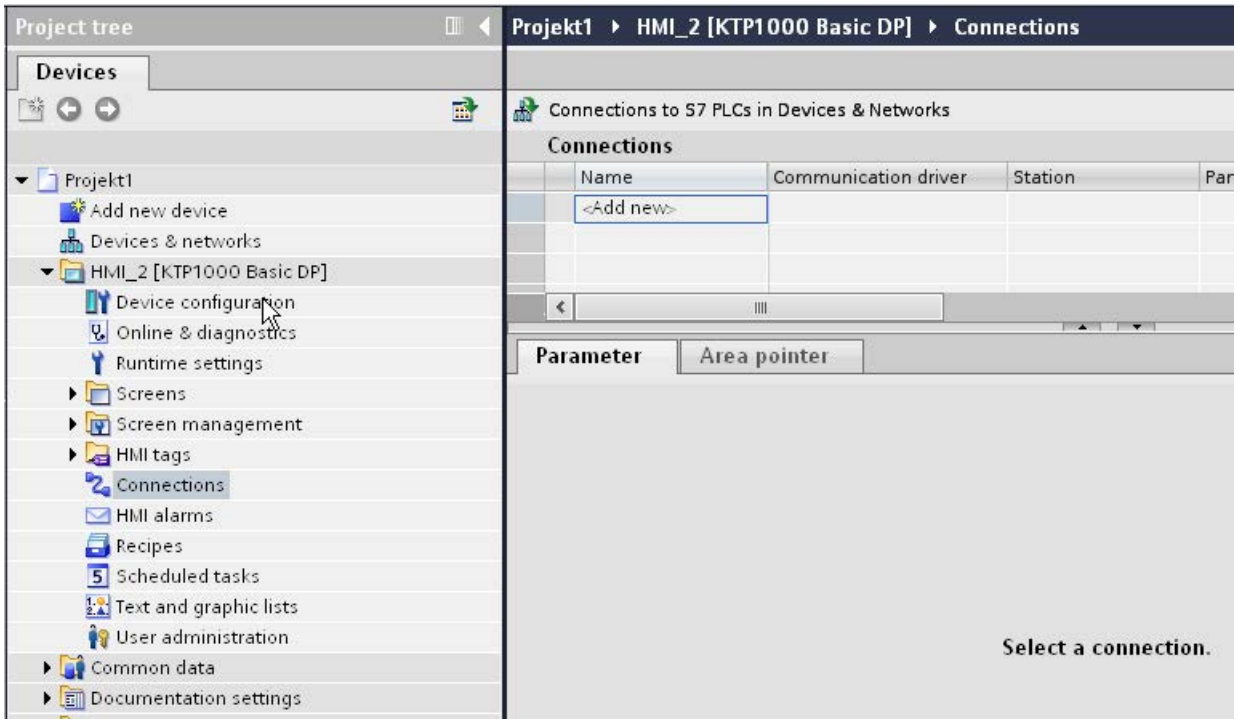
Creating a PROFIBUS DP connection

Requirements

- A project is open.
- An HMI device with a PROFIBUS interface has been created.

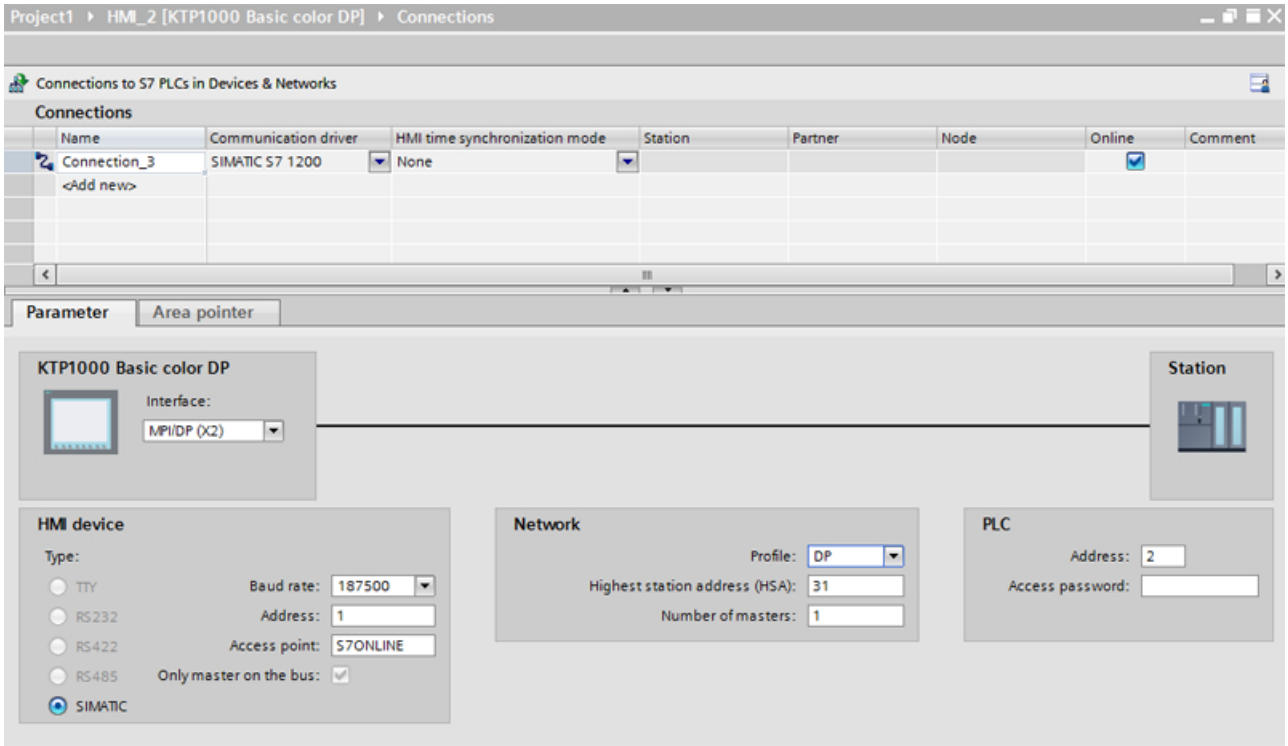
Procedure

1. Open the "Connections" editor of the HMI device.
2. Double-click "<Add>".



3. In the "Communication drivers" column, select the "SIMATIC S7 1200" driver.
4. Click the name of the connection.
5. Select the "MPI/DP" interface in the Inspector window under "Parameters".

6. Select the "DP" profile in the Inspector window under "Parameters > Network".



7. Set the addresses of the communication partners in the Inspector window:
   - HMI device: "Parameters > HMI device > Address"
   - PLC: "Parameters > PLC > Address"

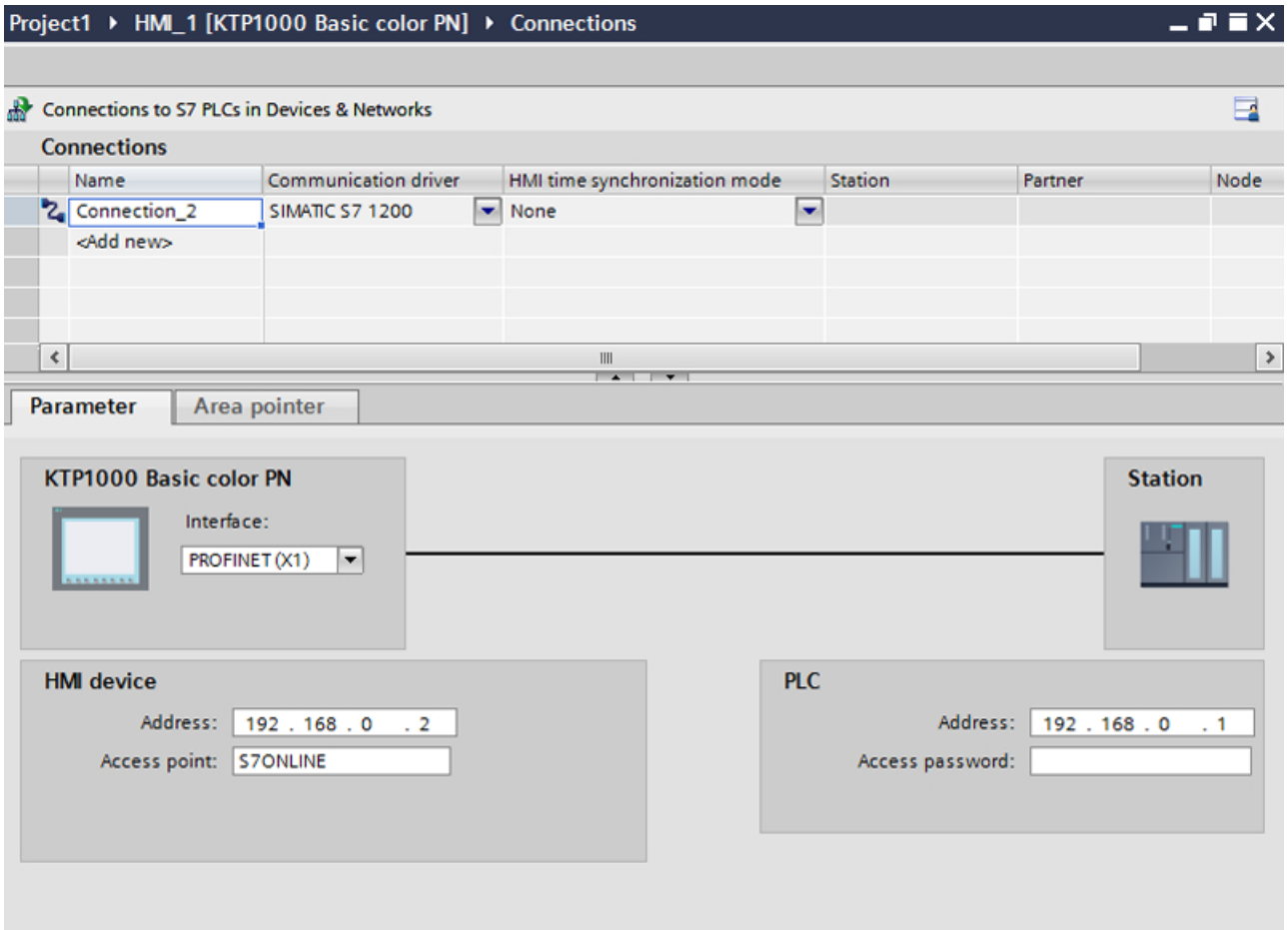
Parameters for the connection

Parameters for the connection (SIMATIC S7 1200)

Parameters to be set

To assign the connection parameters, such as addresses and profiles, click the connection that you have created in the "Connections" editor.

The communication partners are displayed schematically in the Inspector window under "Parameters". The "HMI device", "Network", and "PLC" areas are available for assigning parameters according to the interface used.


Project1 ▸ HMI_1 [KTP1000 Basic color PN] ▸ Connections
Connections to S7 PLCs in Devices & Networks
Connections
Name
Communication driver
HMI time synchronization mode
Station
Partner
Node
Connection_2
SIMATIC S7 1200
None
<Add new>
Parameter
Area pointer
KTP1000 Basic color PN
Interface:
PROFINET (X1)
Station
HMI device
Address: 192 . 168 . 0 . 2
Access point: S7ONLINE
PLC
Address: 192 . 168 . 0 . 1
Access password:

Ethernet parameters

Parameters for the HMI device

You set the parameters for the HMI device in the network under "HMI device".. The changes are not transferred automatically to the HMI device. You must change the settings in the Control Panel of the HMI device.

- "Interface"
  If you are directly connected to the HMI device during configuration, you can set up the IP address of the HMI device in WinCC.

  Note

  The IP address in the Control Panel will be overwritten upon subsequent loading if you have already set up the IP address in the HMI device control panel.

  The IP address already set up in the Control Panel will be retained upon subsequent loading if you activate "Set IP address using a different method".

  The IP address is transferred to the HMI device during project transfer.
  To set up the IP address of the HMI device:

  – Click the HMI device.
  – Open the "Device configuration" editor.
  – Click the Ethernet interface.
  – Assign the IP address in the inspector window under:
    "General > PROFINET interface > Ethernet addresses"

- "Address"
  You assign the IP address of the HMI device in the "Address" area.
  When you transfer the WinCC project to the HMI device, this IP address is set up directly in the HMI device.

- "Access point"
  Specifies the access point for the PG/PC interface that can be used to reach the communication partner.

Parameters for the PLC

Under "PLC", you address the S7 module with which the HMI device will exchange data. Assign a name for the connection for each communication partner.

- "Address"
  Under "Address", set the IP address of the S7 module to which the HMI device is connected.
- "Access password"
  Enter a password in the "Access password" field. This password must match the one you saved to the PLC.

Note

You only need a password if you have set "Complete protection" at the PLC.

No connection is set up to the PLC if the "Complete protection" security level is stored on the PLC and you do not enter a password.

PROFIBUS parameters

Parameters for the HMI device

You assign the parameters for the HMI device in the network once under "HMI device". The change applies to each communication partner.

- "Type"
  Specifies the physical connection used.
- "Interface"
  For "Interface", you select the HMI device interface via which the HMI device is connected to the PROFIBUS network.
- "Baud rate"
  For "Baud rate", you set the transmission speed of the data in the network. The baud rate is determined by the slowest HMI device connected to the network. The setting must be identical throughout the network.

Note

If you set a baud rate of 1.5 Mbaud for OP 73 or OP 77A, the highest station address must be less than or equal to 63.

- "Address"
  You set the PROFIBUS address of the HMI device under "Address". The PROFIBUS address must be unique in the PROFIBUS network.

- "Only master on bus"
  Disables an additional safety feature against bus faults when the HMI device is connected to the network. A passive station (slave) can only send data if it is requested to do so by an active station (master).
  In S7-200, you must set an HMI device as the master.
- "Access point"
  The access point defines a logical device name through which the communication partner can be reached.

Parameters for the network

Under "Network", you set the parameters for the PROFIBUS network to which the HMI device is linked.

- "Profile"
  For "Profile", you select the network profile that is used in the network. In "Profile", set "DP", "Universal", or "Standard". The setting must be identical throughout the network.
- "Highest address"
  For "Highest station address", set the highest station address. The highest station address must be greater than or equal to the highest actual PROFIBUS address. The setting must be identical throughout the network.

  Note

  If you set a baud rate of 1.5 Mbaud in the OP 73 or the OP 77A, the highest station address must be less than or equal 63.

- "Number of masters"
  For "Number of masters", set the number of masters in the PROFIBUS network. This information is necessary to correctly calculate the bus parameters.

Parameters for the PLC

Under "PLC", you address the S7 module with which the HMI device will exchange data. Assign a name for the connection for each communication partner.

- "Address"
  For "Address", set the PROFIBUS address of the S7 module (CPU, FM, or CP) to which the HMI device is connected.
- "Access password"
  Enter a password in the "Access password" field. This password must match the one you saved to the PLC.

Note

You only need a password if you have set "Complete protection" at the PLC.

No connection is set up to the PLC if the "Complete protection" security level is stored on the PLC and you do not enter a password.

10.8.8 Communicating with SIMATIC S7 300/400

10.8.8.1 Communication with SIMATIC S7 300/400

Introduction

This section describes the communication between an HMI device and the SIMATIC S7 300 and S7 400 PLCs. These two PLCs will be referred to jointly as SIMATIC S7 300/400.

You can configure the following communication channels for the SIMATIC S7 300/400 PLC:

- PROFINET
- PROFIBUS
- MPI

HMI connection for communication

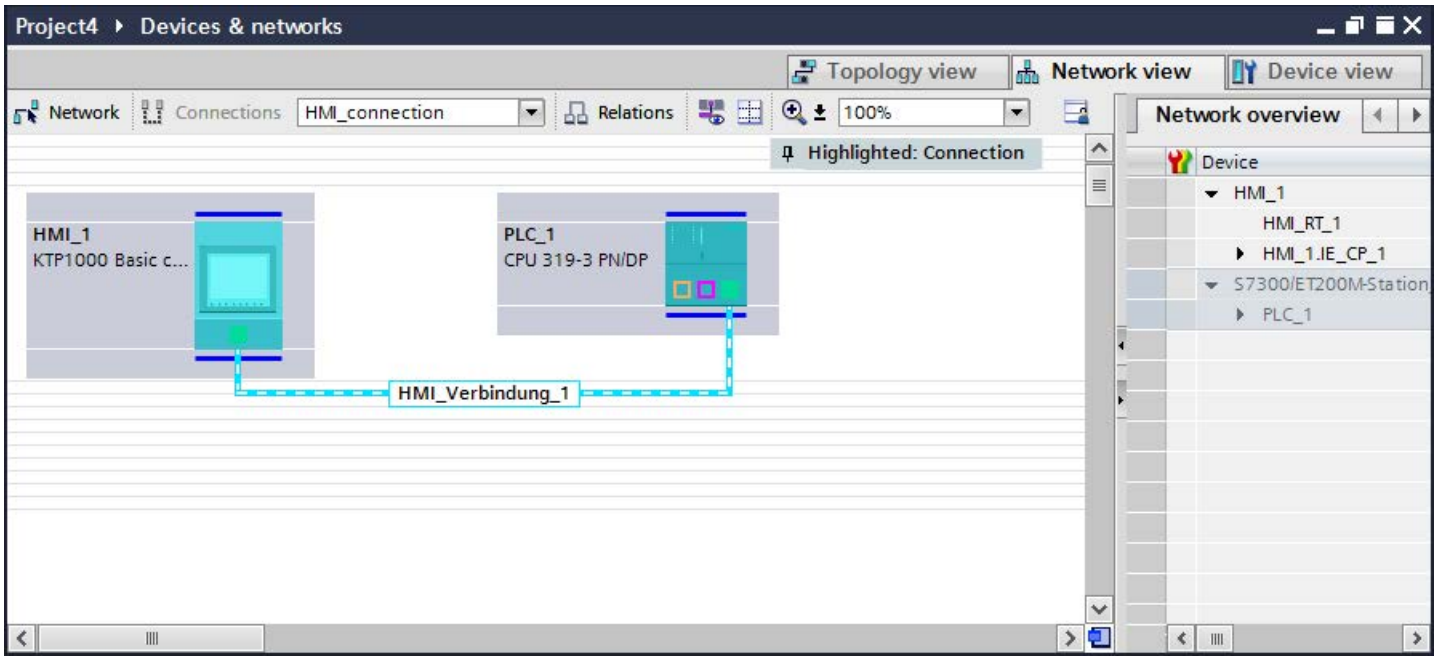
You configure connections between the HMI device and a SIMATIC S7 300/400 in the "Devices & Networks" editor.

10.8.8.2 Communication via PROFINET

Communication via PROFINET

HMI connections via PROFINET

If you have inserted an HMI device and a SIMATIC S7 300/400 into the project, you interconnect the two PROFINET interfaces in the "Devices & Networks" editor.



You can also connect multiple HMI devices to one SIMATIC S7 300/400 and multiple SIMATIC S7 300/400s to one HMI device. The maximum number of communication partners that you can connect to an HMI device is dependent on the HMI device used.

Additional information is available in the documentation for the respective HMI device.

HMI connection in the "Devices & Networks" editor

You configure the HMI connection between the PLC and the HMI device via PROFINET in the "Devices & Networks" editor.

Connection in the "Connections" editor

Alternatively, you configure the connection between the PLC and HMI device via PROFINET in the "Connections" editor of the HMI device.

Configuring an HMI connection via PROFINET

Introduction

You configure an HMI connection between HMI devices and a SIMATIC S7 300/400 via PROFINET or Ethernet in the "Devices & Networks" editor.

CAUTION
Communication via Ethernet In Ethernet-based communication, the end user is responsible for the security of his data network. Targeted attacks can overload the device and interfere with proper functioning.

Requirements

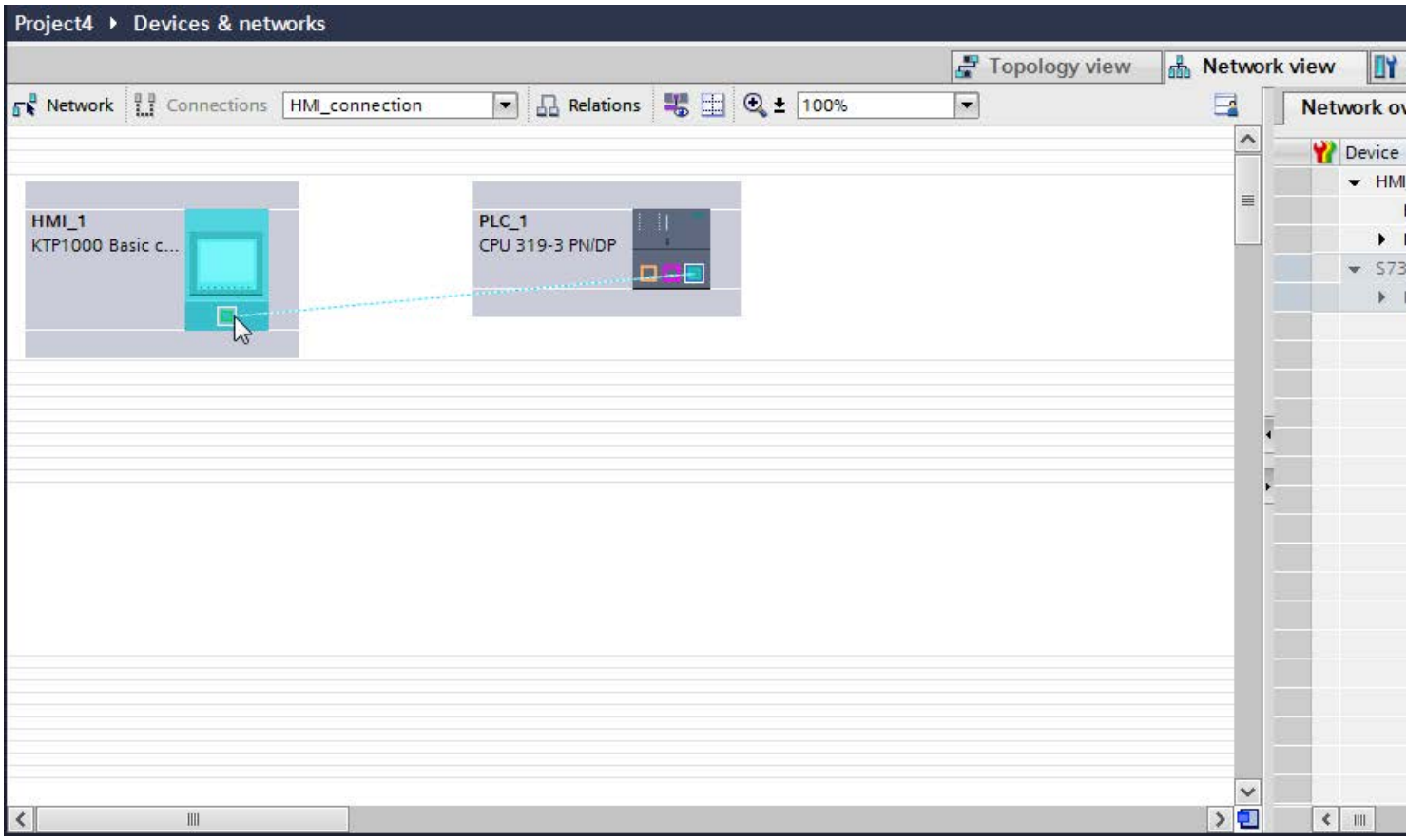
The following communication partners are created in the "Devices & Networks" editor:

- HMI device with PROFINET or Ethernet interface
- SIMATIC S7 300/400 with PROFINET interface.

Procedure

1. Double-click the "Devices & Networks" item in the project tree.
   The available communication partners in the project are displayed graphically in the network view.
2. Click the "Connections" button and select "HMI connection" for the connection type.
   The devices available for connection are highlighted in color.

3. Click the PROFINET interface of the PLC and use a drag-and-drop operation to draw a connection to the PROFINET or Ethernet interface of the HMI device.



4. Click the connecting line.
5. Click "Highlight HMI connection" and select the HMI connection.
   The connection is displayed graphically in the Inspector window.
6. Click the communication partners in the "Network view" and change the PROFINET parameters in the Inspector window according to the requirements of your project.
   See the chapter "PROFINET parameters (Page 8535)" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. You check the connection parameters in the table.

You can change the local name for the connection only in the table.

Result

You have created a connection between an HMI device and a SIMATIC S7 300/400. The IP address and subnet mask connection parameters are configured.

PROFINET parameters

PROFINET parameters for the HMI connection

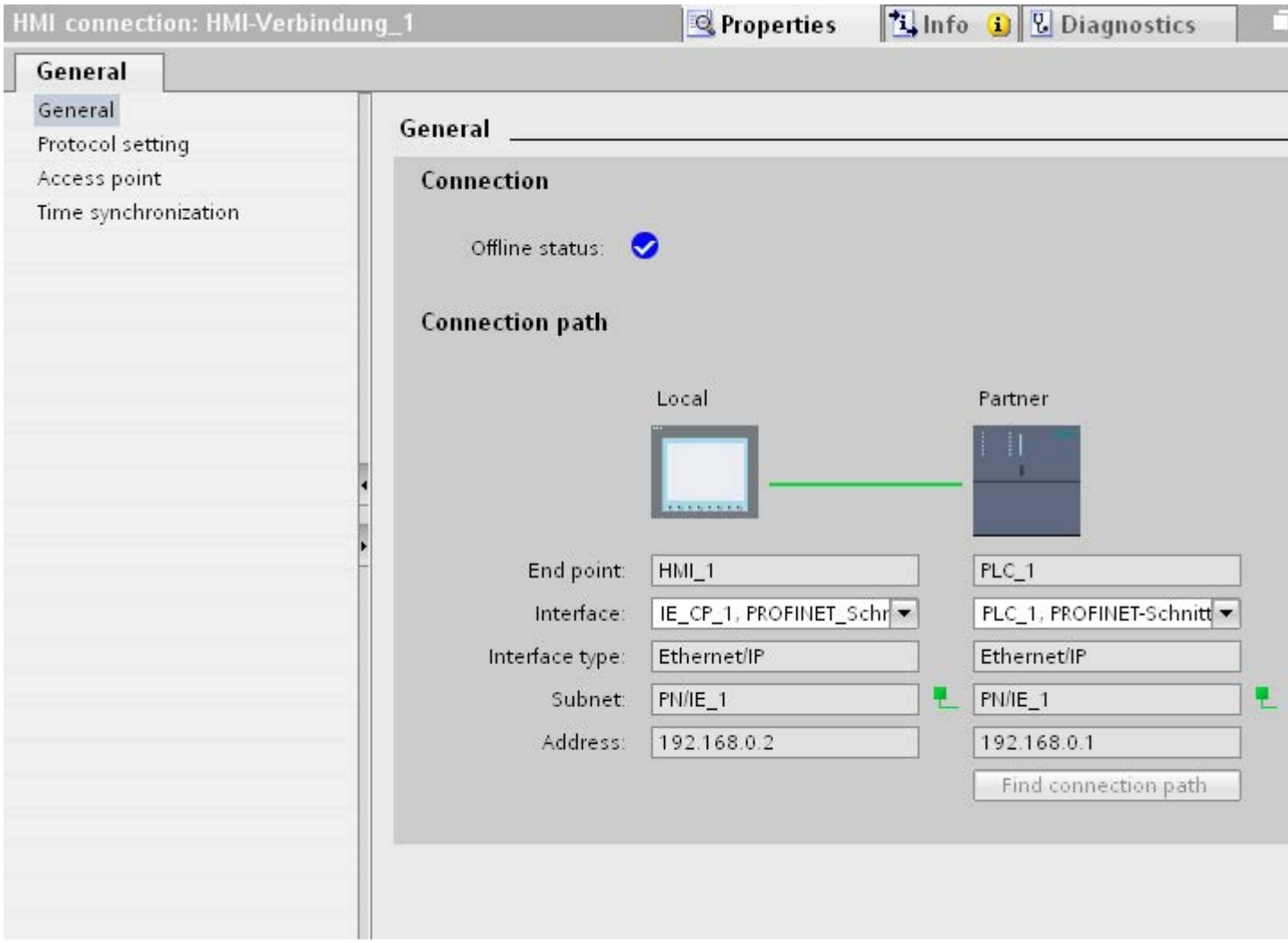
PROFINET parameters for the HMI connection

An overview of the configured HMI connection parameters can be found in the properties for the HMI connection.

Only limited changes are possible in this Inspector window.

Displaying and changing the HMI connection parameters

1. Click the HMI connection in the "Devices & Networks" editor.
2. Change the parameters of the HMI connection in the Inspector window under "Properties > General > General".



"Connection"

Displays whether the devices are networked together.

- displayed if the devices are networked together.

- displayed if the devices are not networked together.

"Connection path"

The communication partners of the selected HMI connection and the associated PROFINET parameters are displayed in the "Connection path" area. Some of the areas displayed cannot be edited in this dialog.

- "End point"
  Displays the device name. This area cannot be edited.
- "Interface"
  Displays the selected interface of the device. You can choose between several interfaces, depending on the device.
- "Interface type"
  Displays the selected interface type. This area cannot be edited.
- "Subnet"
  Displays the selected subnet. This area cannot be edited.
- "Address"
  Displays the selected IP address of the device. This area cannot be edited.
- "Find connection path" button
  Enables the subsequent specification of connections.

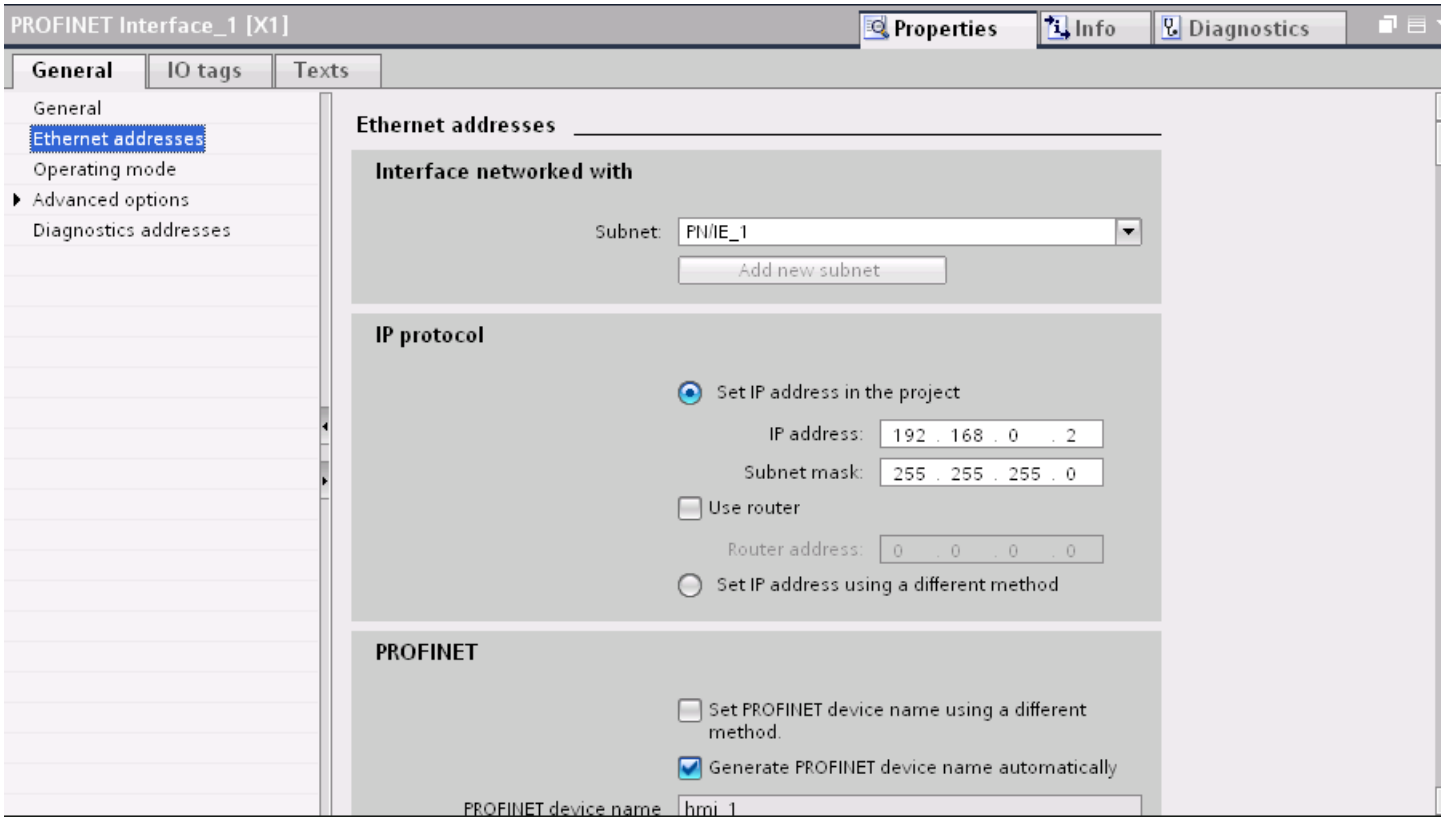
PROFINET parameters for the HMI device

PROFINET parameters for the HMI device

An overview of the configured HMI device parameters can be found in the properties for the HMI device.

Displaying and changing PROFINET parameters of the HMI device

1. Click the HMI device in the "Devices & Networks" editor.
2. Change the parameters of the HMI device in the Inspector window under "Properties > General".



"Interface networked with"

In the "Interface networked with" area, select the subnet of the HMI connection via which the HMI device is connected to the network. You use the "Add new subnet" button to create a new subnet.

"IP protocol"

- "Set IP address in the project"
  When you transfer the WinCC project to the HMI device, this IP address is set up directly in the HMI device.

  Note

  The device is automatically restarted in the case of HMI devices with the Windows CE 3.0 operating system.

  HMI devices with Windows CE 3.0:
  - OP 77B
  - TP 177B color PN/DP
  - TP 177B mono DP
  - OP 177B color PN/DP
  - OP 177B mono DP
  - Mobile Panel 177 PN
  - Mobile Panel 177 DP
  - TP 277 6"
  - OP 277 6"

- "Subnet mask"
  You assign data of the subnet mask in the "Subnet mask" area.
- "Use IP router"
  If you are using an IP router, select "Use IP router" and enter the router address in the "Router address" field.
- "Set IP address using a different method"
  If the function "Set IP address using a different method" is activated, the IP address is not taken from the project. You have to enter the IP address directly in the Control Panel of the HMI device.

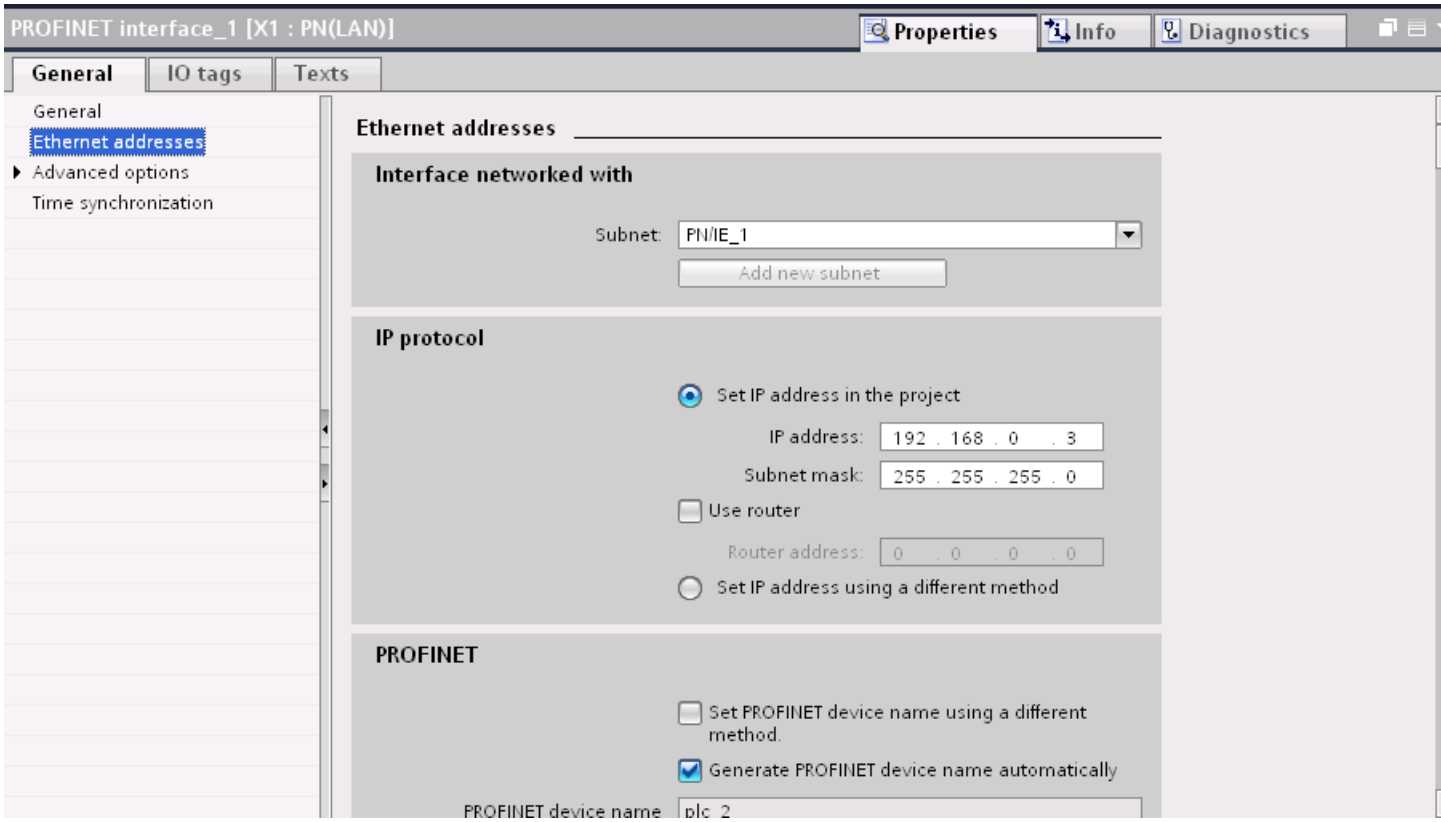
PROFINET parameters for the PLC

PROFINET parameters for the PLC

An overview of the configured parameters can be found in the properties for the PLC.

Displaying and changing PROFINET parameters of the PLC

1. Click the PLC in the "Devices & Networks" editor.
2. Change the parameters of the PLC in the Inspector window under "Properties > General > General".



"Interface networked with"

In the "Subnet" area, select the subnet of the HMI connection via which the PLC is connected to the network. You use the "Add new subnet" button to create a new subnet.

"IP protocol"

- "Interface type"
  Depending on the HMI device type, you have various interfaces to choose from.
- "IP address"
  You assign the IP address of the HMI device in the "IP address" area.
- "Subnet mask"
  You assign data of the subnet mask in the "Subnet mask" area.
  If you are using an IP router, select "Use IP router" and enter the router address in the field.

Configuring Industrial Ethernet

Rules for the network configuration

The Ethernet interfaces of the devices have a default IP address that you can change.

IP address

The IP parameters are visible if the communication-capable devices support the TCP/IP protocol.

The IP address consists of 4 decimal figures in the range of 0 to 255. The decimal figures are separated from one another by a dot.

Example: 140.80.0.2

The IP address consists of the following:

- The address of the (sub) net
- The address of the node (generally also called host or network node)

Subnet mask

The subnet mask splits these two addresses. It determines which part of the IP address addresses the network and which part of the IP address addresses the node.

The set bits of the subnet mask determine the network part of the IP address.

Example:

Subnet mask: 255.255.0.0 = 11111111.11111111.00000000.00000000

In the example given for the above IP address, the subnet mask shown here has the following meaning:

The first 2 bytes of the IP address identify the subnet - i.e. 140.80. The last two bytes address the node, thus 0.2.

It is generally true that:

- The network address results from AND linking the IP address and subnet mask.
- The node address results from AND NOT linking the IP address and subnet mask.

Relation between IP address and default subnet mask

An agreement exists relating to the assignment of IP address ranges and so-called "Default subnet masks". The first decimal number (from the left) in the IP address determines the structure of the default subnet mask. It determines the number of "1" values (binary) as follows:

IP address (decimal)	IP address (binary)	Address class	Default subnet mask
0 to 126	0xxxxxxx.xxxxxxxx....	A	255.0.0.0
128 to 191	10xxxxxx.xxxxxxxx...	B	255.255.0.0
192 to 223	110xxxxx.xxxxxxxx...	C	255.255.255.0

Note

Range of values for the first decimal point

A value of between 224 and 255 is also possible for the first decimal number of the IP address (address class D etc). This is, however, not recommended because there is no address check for these values.

Masking other subnets

You can use the subnet mask to add further structures and form "private" subnets for a subnet that is assigned one of the address classes A, B or C. This is done by setting other lower points of the subnet mask to "1". For each bit set to "1", the number of "private" networks doubles and the number of nodes they contain is halved. Externally, the network functions like an individual network as it did previously.

Example:

You have a subnet of address class B (e.g. IP address 129.80.xxx.xxx) and change the default subnet mask as follows:

Masks	Decimal	Binary
Default subnet mask	255.255.0.0	11111111.11111111.00000000.00000000
Subnet mask	255.255.128.0	11111111.11111111.10000000.00000000

Result:

All nodes with addresses between 129.80.001.xxx and 129.80.127.xxx are on one subnet, all nodes with addresses between 129.80.128.xxx and 129.80.255.xxx are on another subnet.

Router

The job of the routers is to connect the subnets. If an IP datagram is to be sent to another network, it first has to be conveyed to a router. To make this possible, in this case you have to enter the address of the router for each node in the subnet.

The IP address of a node in the subnet and the address of the router may only differ at the points at which there is a "0" in the subnet mask.

Setting port options

Setting the port options

Changing connection settings for the PROFINET IO port

You can change the network settings for the PROFINET IO port as required. By default, the settings are made automatically. In normal situations, this guarantees problem-free communication.

Possible settings for transmission rate / duplex

Depending on the selected device, you can make the following settings for "Transmission rate / duplex":

- Automatic setting
  Recommended default setting of the port. The transmission settings are automatically "negotiated" with the peer port. The "Enable autonegotiation" option is also enabled as a default, in other words, you can use cross cables or patch cables for the connection.
- TP/ITP at x Mbps full duplex (half duplex)
  Setting of the transmission rate and the full duplex/half duplex mode. The effectiveness depends on the "Enable autonegotiation" setting:
  - Autonegotiation enabled
    You can use both cross cable and patch cable.
  - Autonegotiation disabled
    Make sure that you use the correct cable (patch cable or cross cable)! The port is also monitored with this setting.
- Deactivated
  Depending on the module type, the drop down list box can contain the "- Disabled -" option. This option, for example, allows you to prevent access to an unused port for security reasons. With this setting, diagnostic events are not generated.

"Monitor" option

This option enables or disables port diagnostics. Examples of port diagnostics: The link status is monitored, in other words, the diagnostics are generated during link-down and the system reserve is monitored in the case of fiber optic ports.

Option "Enable autonegotiation "

The autonegotiation setting can only be changed if a concrete medium (for example, TP 100 Mbps full duplex) is selected. Whether or not a concrete medium can be set depends on the properties of the module.

If autonegotiation is disabled, this causes the port to be permanently specified, as for example, is necessary for a prioritized startup of the IO device.

You must make sure the partner port has the same settings because with this option the operating parameters of the connected network are not detected and the data transmission rate and transmission mode can accordingly not be optimally set.

Note

When a local port is connected, STEP 7 makes the setting for the partner port if the partner port supports the setting. If the partner port does not accept the setting, an error message is generated.

Wiring rules for disabled autonegotiation

Requirements

You have made the following settings for the port in question, for example, to accelerate the startup time of the IO device:

- Fixed transmission rate
- Autonegotiation incl. autocrossing disabled

The time for negotiating the transmission rate during startup has been saved.

If you have disabled autonegotiation, you must observe the wiring rules.

Wiring rules for disabled autonegotiation

PROFINET devices have the following two types of ports:

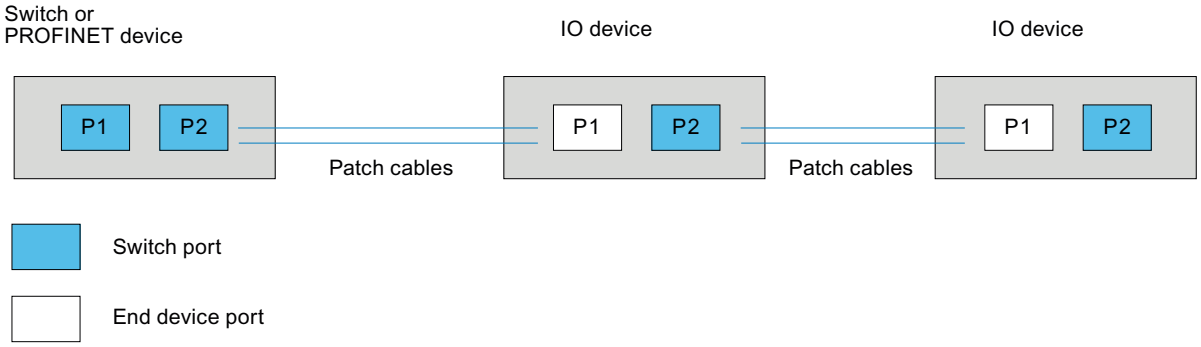
Type of port	PROFINET devices	Note
Switch port with crossed pin assignment	For IO devices: Port 2 For S7 CPUs with 2 ports: Ports 1 and 2	Crossed pin assignment means that the pin assignment for the ports for sending and receiving between the respective PROFINET devices is exchanged internally.
End device port with uncrossed pin assignment	For IO devices: Port 1 For S7 CPUs with one port: Port 1	-

Validity of the wiring rules

The cabling rules described in the following paragraph apply exclusively for the situation in which you have specified a fixed port setting.

Rules for cabling

You can connect several IO devices in line using a single cable type (patch cable). To do this, you connect port 2 of the IO device (distributed I/O) with port 1 of the next IO device. The following graphic gives an example with two IO devices.



Boundaries at the port

Requirements

To use boundaries, the respective device must have more than one port. If the PROFINET does not support boundary settings, they are not shown.

Enable boundaries

"Boundaries" are limits for transmission of certain Ethernet frames. The following boundaries can be set at a port:

- "End of discovery of accessible devices"
  No forwarding of DCP frames to identify accessible devices. Devices downstream from this port cannot be reached by the project tree under "Accessible devices". Devices downstream from this port cannot be reached by the CPU.
- "End of topology discovery"
  LLDP frames (Link Layer Discovery Protocol) are not forwarded for topology detection.
- "End of sync domain"
  No forwarding of sync frames transmitted to synchronize nodes within a sync domain.
  If you operate, for example, a PROFINET device with more than two ports in a ring, you should prevent the sync frame from being fed into the ring by setting a sync boundary (at the ports not inside the ring).
  Additional example: If you want to use several sync domains, configure a sync domain boundary for the port connected to a PROFINET device from the other sync domain.

Restrictions

The following restrictions must be observed:

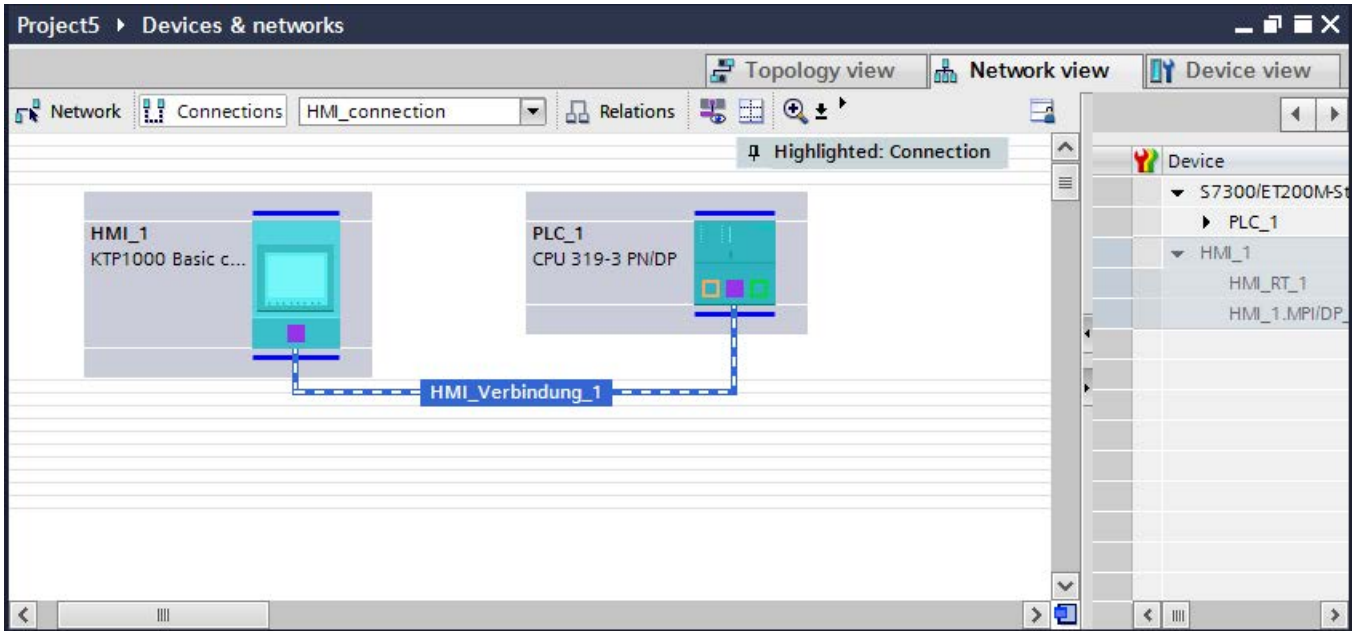
- The individual check boxes can only be used if the port supports the function in question.
- If a partner port has been determined for the port, the following check boxes cannot be used:
  - "End of discovery of accessible devices"
  - "End of topology discovery"
- If autonegotiation is disabled, none of the check boxes can be used.

10.8.8.3 Communication via PROFIBUS

Communication via PROFIBUS

HMI connections via PROFIBUS

If you have inserted an HMI device and a SIMATIC S7 300/400 into the project, you interconnect the two PROFIBUS interfaces in the "Devices & Networks" editor.



HMI connection in the "Devices & Networks" editor

You configure the HMI connection between the PLC and the HMI device via PROFIBUS in the "Devices & Networks" editor.

Connection in the "Connections" editor

Alternatively, you configure the connection between the PLC and HMI device in the "Connections" editor of the HMI device.

Configuring an HMI connection via PROFIBUS

Introduction

You configure an HMI connection between HMI devices and a SIMATIC S7 300/400 via PROFIBUS in the "Devices & Networks" editor.

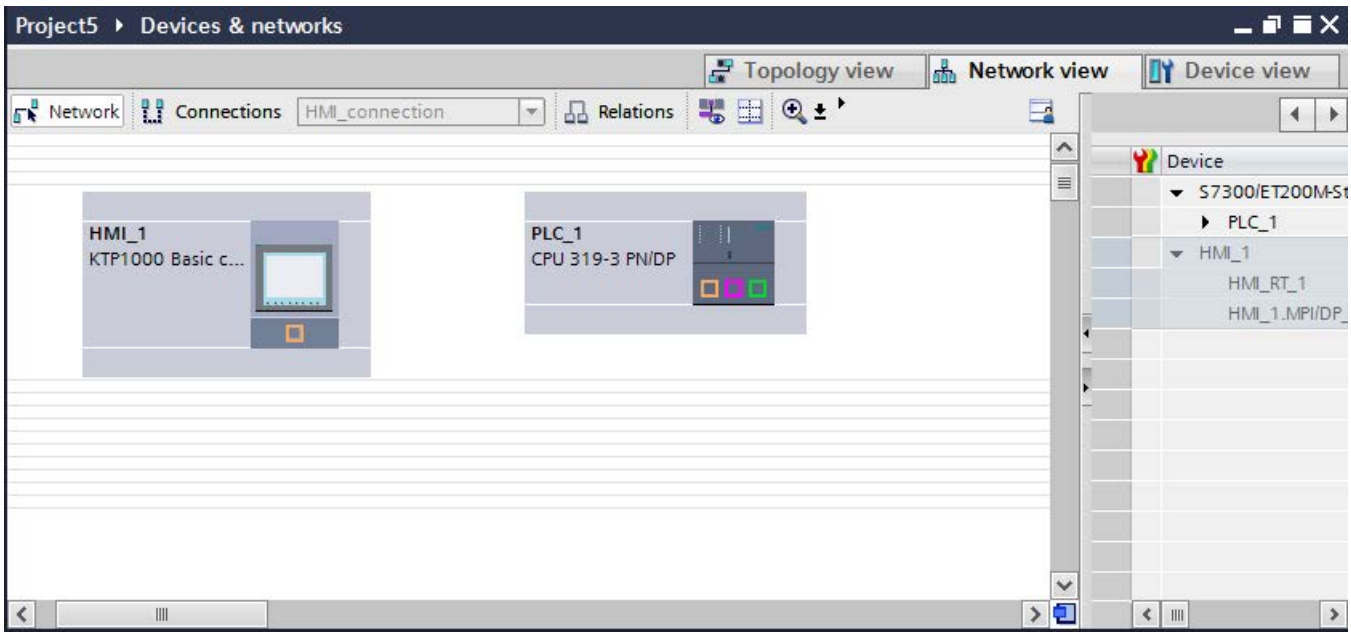
Requirements

The following communication partners are created in the "Devices & Networks" editor:

- HMI device with MPI/DP interface
- SIMATIC S7 300/400 with PROFIBUS interface

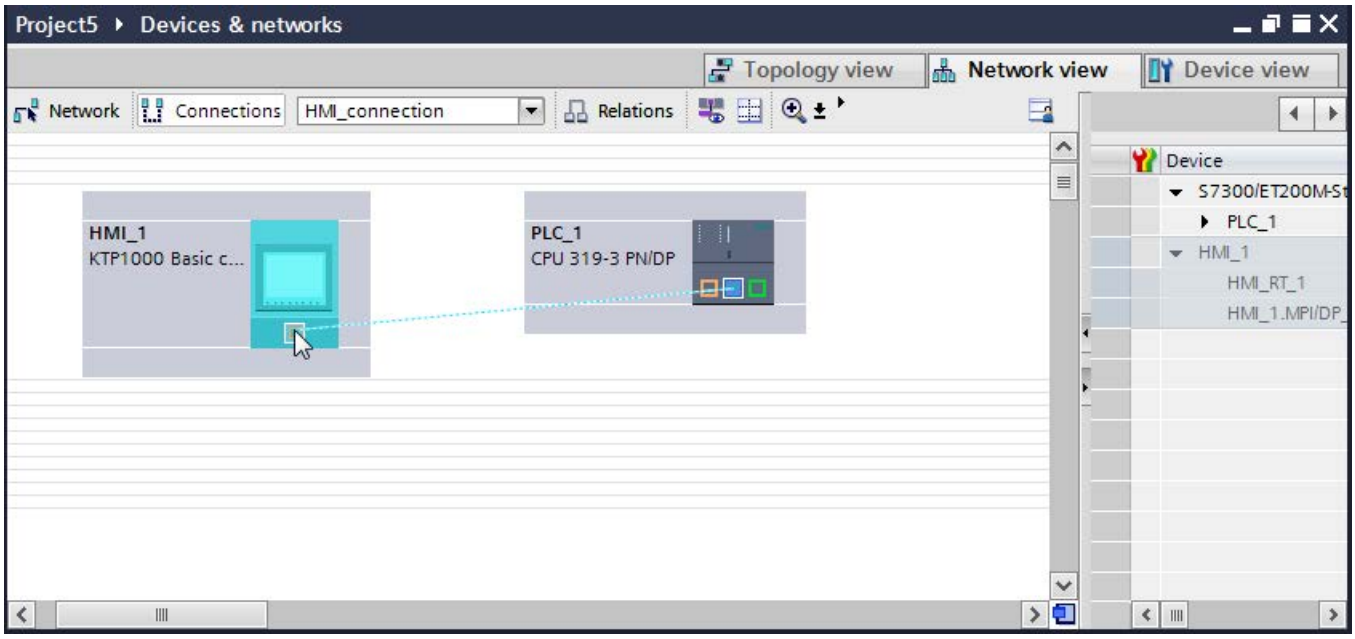
Procedure

1. Double-click the "Devices & Networks" item in the project tree.
   The available communication partners in the project are displayed graphically in the network view.



2. Click the "Connections" button.
   The devices available for connection are highlighted in color.
3. Click the HMI device interface.
4. Select the "PROFIBUS" interface type in the Inspector window under "Properties > General > HMI MPIDP > Parameters".

5. Click the interface of the PLC and use a drag-and-drop operation to draw a connection to the HMI device.



6. Click the connecting line.
7. Click "Highlight HMI connection" and select the HMI connection.
   The connection is displayed graphically in the Inspector window.
8. Click the communication partners in the "Network view" and change the PROFINET parameters in the Inspector window according to the requirements of your project.
   See the chapter "PROFIBUS parameters (Page 8548)" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. You check the connection parameters in the table.

You can change the local name for the connection only in the table.

Result

You have created an HMI connection between an HMI device and a SIMATIC S7 300/400 via PROFIBUS.

PROFIBUS parameters

PROFIBUS parameters for the HMI connection

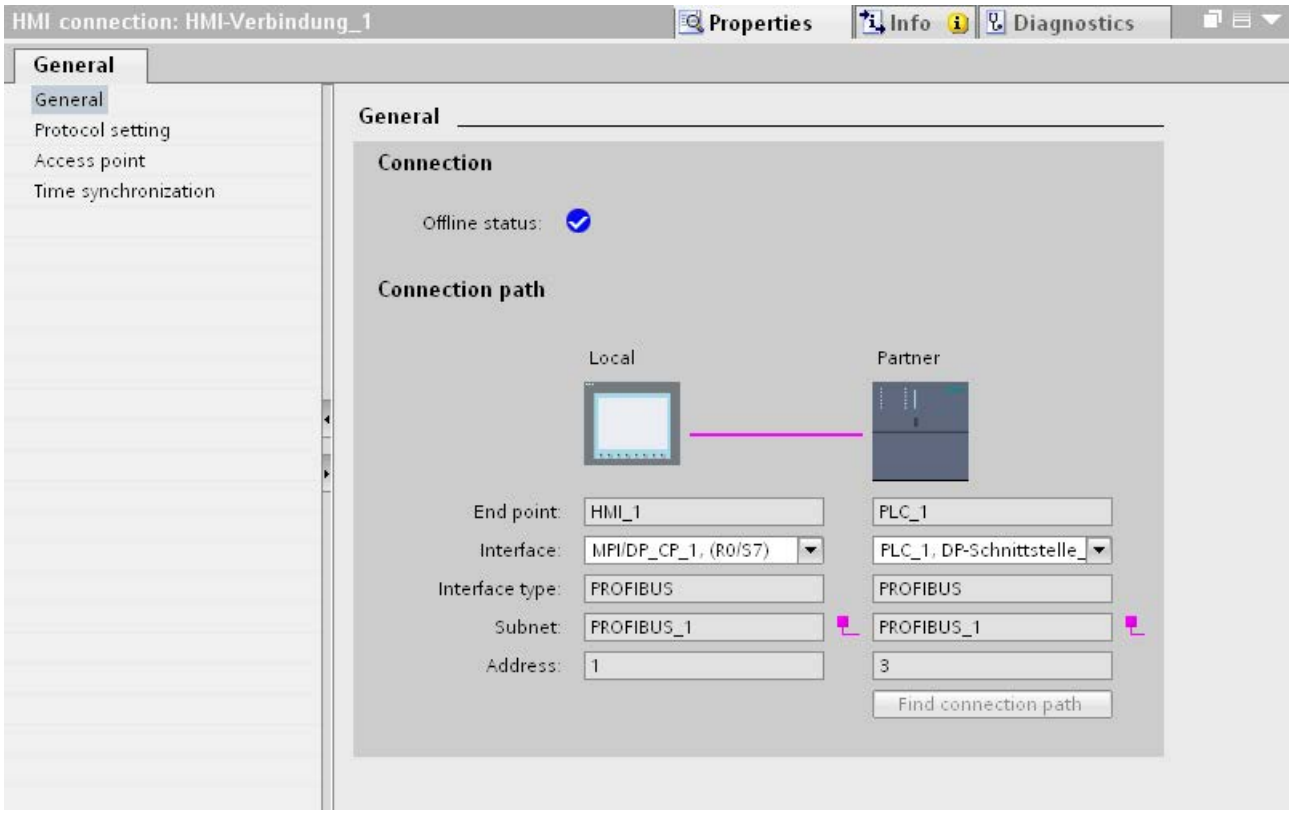
PROFIBUS parameters for the HMI connection

An overview of the configured HMI connection parameters can be found in the properties for the HMI connection.

Only limited changes are possible in this Inspector window.

Displaying and changing the HMI connection parameters

1. Click the HMI connection in the "Devices & Networks" editor.
2. Change the parameters of the HMI connection in the Inspector window under "Properties > General > General".



"Connection"

Displays whether the devices are networked together.

- displayed if the devices are networked together.

- displayed if the devices are not networked together.

"Connection path"

The communication partners of the selected HMI connection and the associated PROFIBUS parameters are displayed in the "Connection path" area. Some of the areas displayed cannot be edited in this dialog.

- "End point"
  Displays the device name. This area is not editable.
- "Interface"
  Displays the selected interface of the device. You can choose between several interfaces, depending on the device.
- "Interface type"
  Displays the selected interface type. This area is not editable.
- "Subnet"
  Displays the selected subnet. This area is not editable.
- "Address"
  Displays the PROFIBUS address of the device. This area is not editable.
- "Find connection path" button
  Enables the subsequent specification of connections.

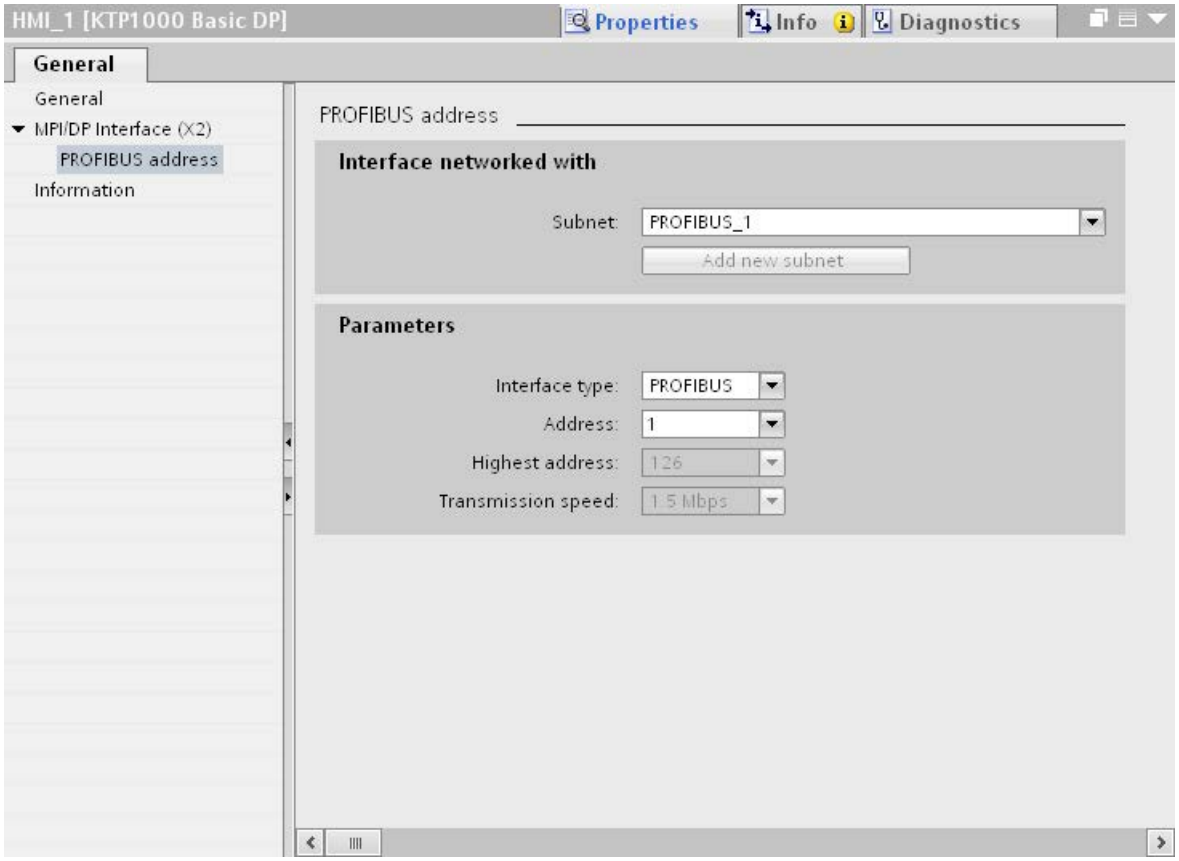
PROFIBUS parameters for the HMI device

PROFIBUS parameters for the HMI device

An overview of the configured HMI device parameters can be found in the properties for the HMI device.

Displaying and changing PROFINET parameters of the HMI device

1. Click the HMI device in the "Devices & Networks" editor.
2. Change the parameters of the HMI device in the Inspector window under "Properties > General > General".



"Interface networked with"

In the "Interface networked with" area, select the subnet of the HMI connection via which the HMI device is connected to the network. You use the "Add new subnet" button to create a new subnet.

"Parameters"

- "Interface type"
  You assign the interface type in the "Interface type" area. Depending on the HMI device type, you have various interfaces to choose from.
- "Address"
  You assign the PROFIBUS address of the HMI device in the "Address" area. The PROFIBUS address must be unique throughout the PROFIBUS network.

- "Highest address"
  The "Highest address" area displays the highest address of the PROFIBUS network.
- "Transmission speed"
  The "Transmission speed" is determined by the slowest device connected to the network. The setting is identical throughout the network.

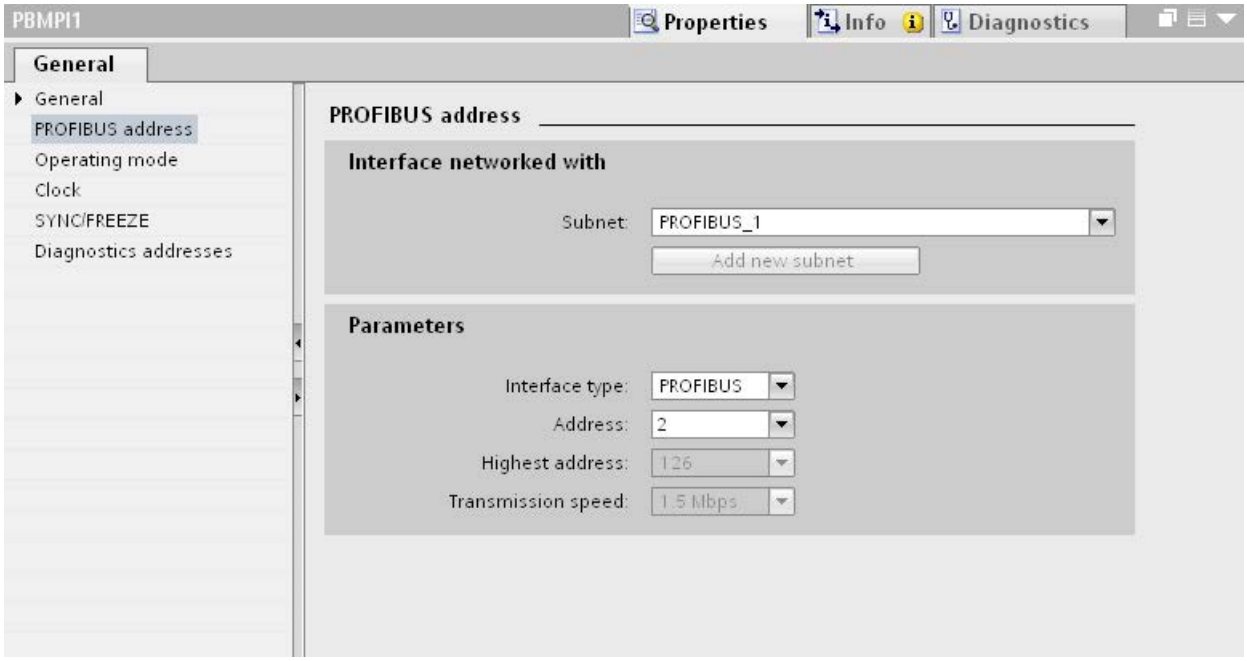
PROFIBUS parameters for the PLC

PROFIBUS parameters for the PLC

An overview of the configured parameters can be found in the properties for the PLC.

Displaying and changing PROFIBUS parameters of the PLC

1. Click the PLC in the "Devices & Networks" editor.
2. Change the parameters of the PLC in the Inspector window under "Properties > General > General".



"Interface networked with"

In the "Subnet" area, select the subnet of the HMI connection via which the PLC is connected to the network. You use the "Add new subnet" button to create a new subnet.

"Parameters"

- "Interface type"
  Depending on the HMI device type, you have various interfaces to choose from.
- "Address"
  You assign the PROFIBUS address of the HMI device in the "Address" area. The PROFIBUS address must be unique throughout the PROFIBUS network.
- "Highest address"
  The "Highest address" area displays the highest address of the PROFIBUS network.
- "Transmission speed"
  The "Transmission speed" is determined by the slowest device connected to the network. The setting is identical throughout the network.

Bus profiles with PROFIBUS

Introduction

Depending on the device types connected and protocols used on the PROFIBUS, different profiles are available. The profiles differ in terms of the setting options and calculation of bus parameters. The profiles are explained below.

Devices with different profiles on the same PROFIBUS subnet

The PROFIBUS subnet only functions without problem if the bus parameters of all devices have the same values.

Profiles and transmission rates

Profiles	Supported transmission speeds in Kbits/s
DP	9,6 19,2 45,45 93,75 187,5 500 1500 3000 6000 12000
Standard	9,6 19,2 45,45 93,75 187,5 500 1500 3000 6000 12000
Universal	9,6 19,2 93,75 187,5 500 1500

Meaning of profiles

Profile	Meaning
DP	Select the "DP" bus profile when the only devices connected to the PROFIBUS subnet are those which satisfy the requirements of standard EN 50170 Volume 2/3, Part 8-2 PROFIBUS. The bus parameter setting is optimized on these devices. This includes devices with DP master and DP slave interfaces of the SIMATIC S7 and distributed I/Os of other manufacturers.
Standard	Compared to the "DP" profile, the "Standard" profile also offers scope for devices of another project or devices which have not been configured here to be taken into account when calculating the bus parameters. The bus parameters are then calculated following a simple, non-optimized algorithm.
Universal	Select the "Universal" bus profile when individual devices on the PROFIBUS subnet use the PROFIBUS-FMS service. This includes the following devices for example: • CP 343-5 • PROFIBUS-FMS devices of other manufacturers As with the "Standard" profile, this profile allows you to take other devices into account when calculating the bus parameters.

10.8.8.4 Communication via MPI

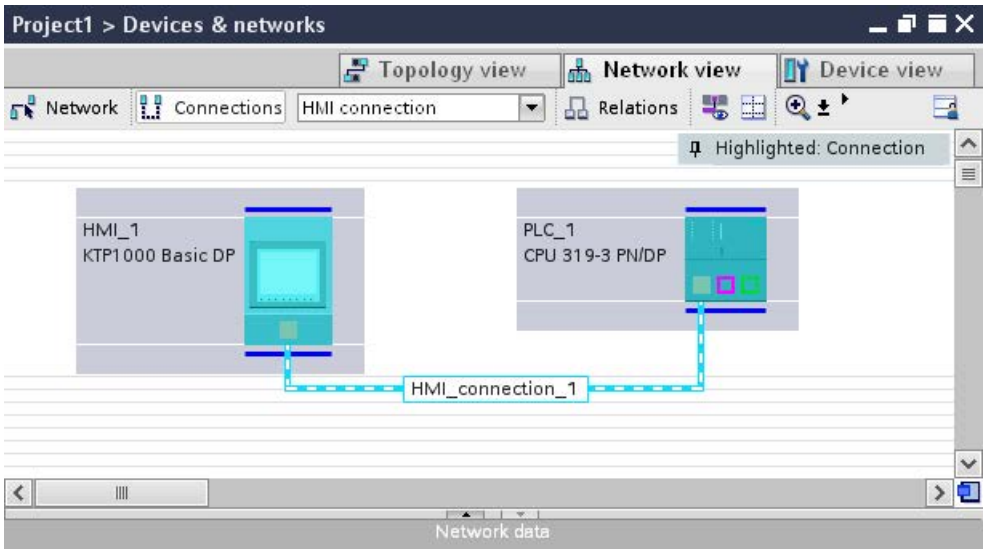
Communication via MPI

HMI connections via MPI

If you have inserted an HMI device and a SIMATIC S7 300/400 into the project, you interconnect the two MPI interfaces in the "Devices & Networks" editor.

HMI connection in the "Devices & Networks" editor

You configure the HMI connection between the PLC and the HMI device via PROFIBUS in the "Devices & Networks" editor.



Connection in the "Connections" editor

Alternatively, you configure the connection between the PLC and HMI device in the "Connections" editor of the HMI device.

Configuring an HMI connection via MPI

Introduction

You configure an HMI connection between HMI devices and a SIMATIC S7 300/400 via MPI in the "Devices & Networks" editor.

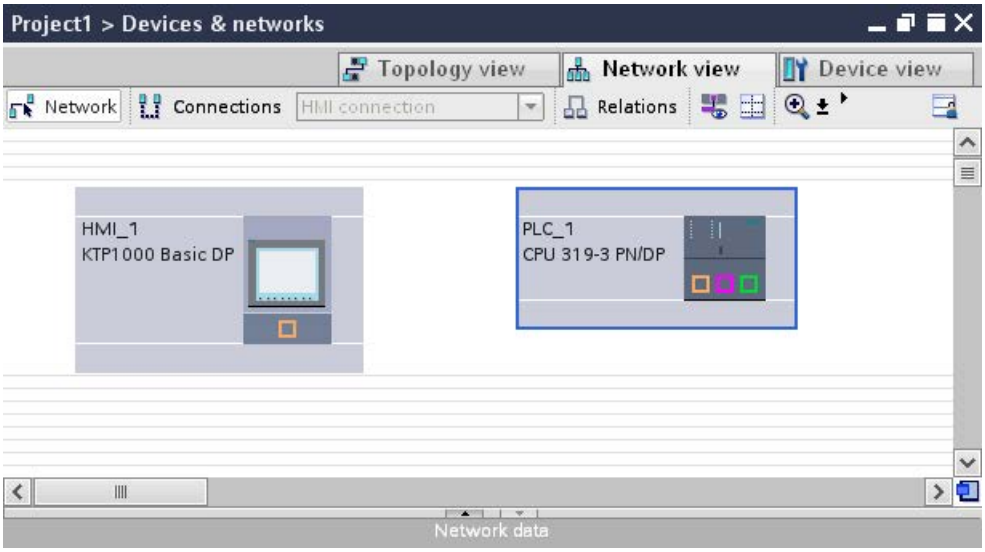
Requirements

The following communication partners are created in the "Devices & Networks" editor:

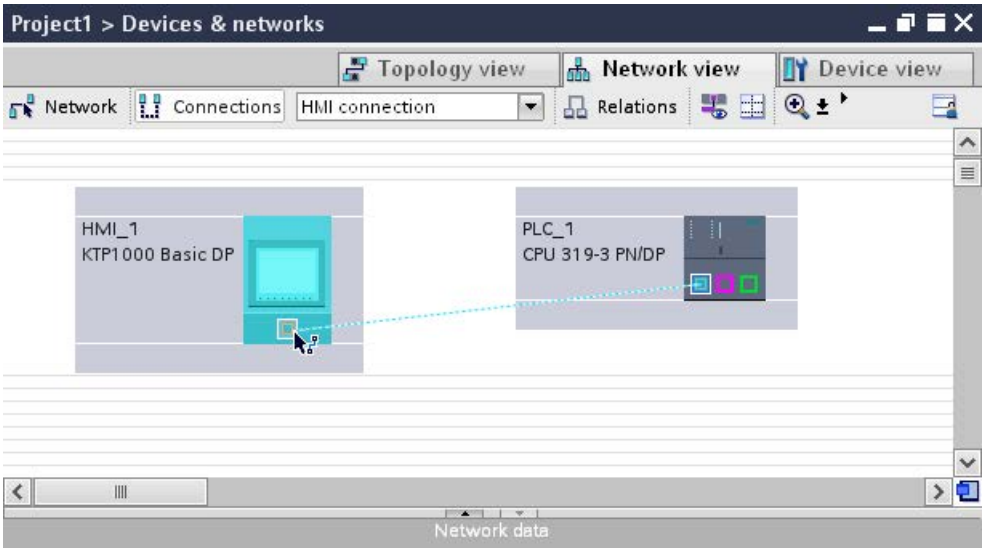
- HMI device with MPI/DP interface
- SIMATIC S7 300/400 with MPI/DP interface

Procedure

1. Double-click the "Devices & Networks" item in the project tree.
   The available communication partners in the project are displayed graphically in the network view.



2. Click the "Connections" button.
   The devices available for connection are highlighted in color.
3. Click the interface of the PLC and use a drag-and-drop operation to draw a connection to the HMI device.



4. Click the connecting line.
   The connection is displayed graphically in the Inspector window.

5. Click "Highlight HMI connection" and select the HMI connection.
6. Click the communication partners in the "Network view" and change the MPI parameters in the Inspector window according to the requirements of your project.
   See the chapter "MPI parameters (Page 8556)" for additional details.

Note

The created HMI connection is also shown in the tabular area of the editor on the "Connections" tab. Use the table to monitor the connection parameters and change the connection partner. You can change the local name for the connection only in the table.

Result

You have created an HMI connection between an HMI device and a SIMATIC S7 300/400 via MPI.

MPI parameters

MPI parameters for the HMI connection

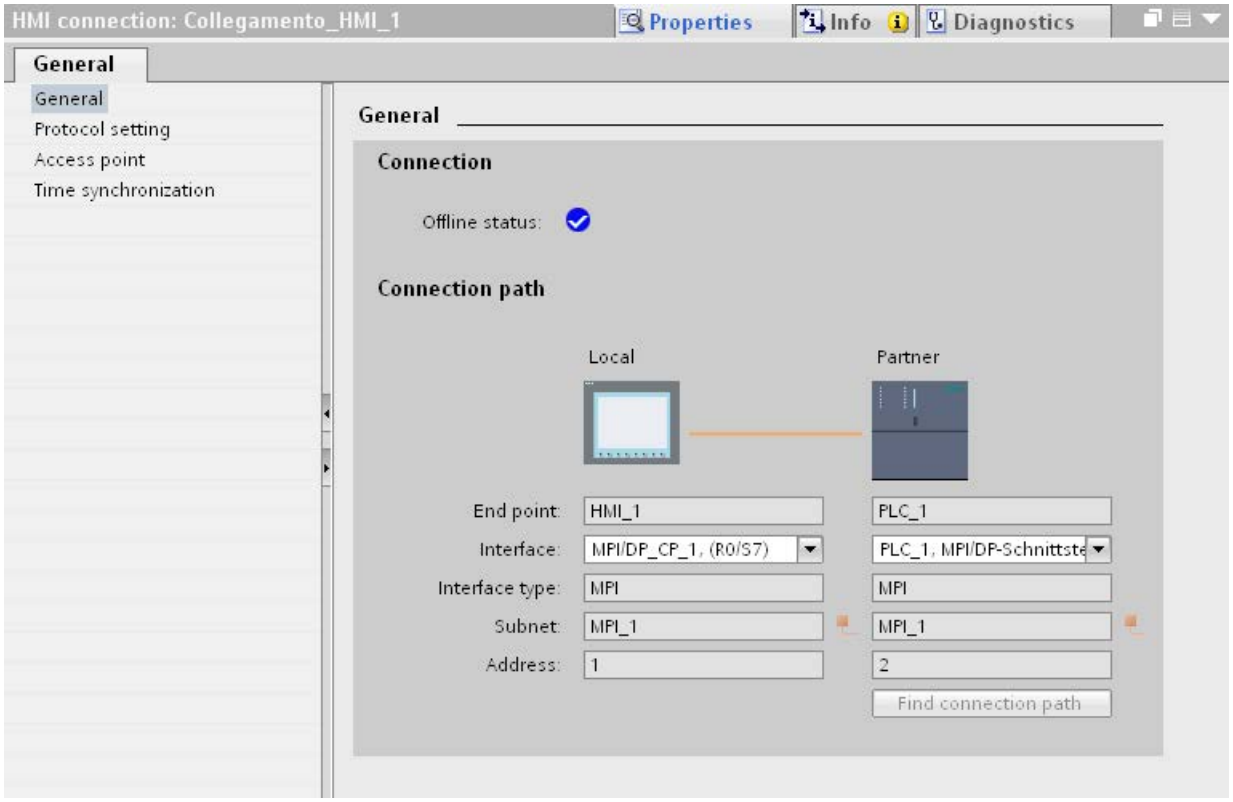
MPI parameters for the HMI connection

An overview of the configured HMI connection parameters can be found in the properties for the HMI connection.

Only limited changes are possible in this Inspector window.

Displaying and changing the HMI connection parameters

1. Click the HMI connection in the "Devices & Networks" editor.
2. Change the parameters of the HMI connection in the Inspector window under "Properties > General > General".



"Connection"

Displays whether the devices are networked together.

- displayed if the devices are networked together.

- displayed if the devices are not networked together.

"Connection path"

The communication partners of the selected HMI connection and the associated MPI parameters are displayed in the "Connection path" area. Some of the areas displayed cannot be edited in this dialog.

- "End point"
  Displays the name of the device. This area is not editable.
- "Interface"
  Displays the selected interface of the device. You can choose between several interfaces, depending on the device.

- "Interface type"
  Displays the selected interface type. This area is not editable.
- "Subnet"
  Displays the selected subnet. This area is not editable.
- "Address"
  Displays the MPI address of the device. This area is not editable.
- "Find connection path" button
  Enables the subsequent specification of connections.

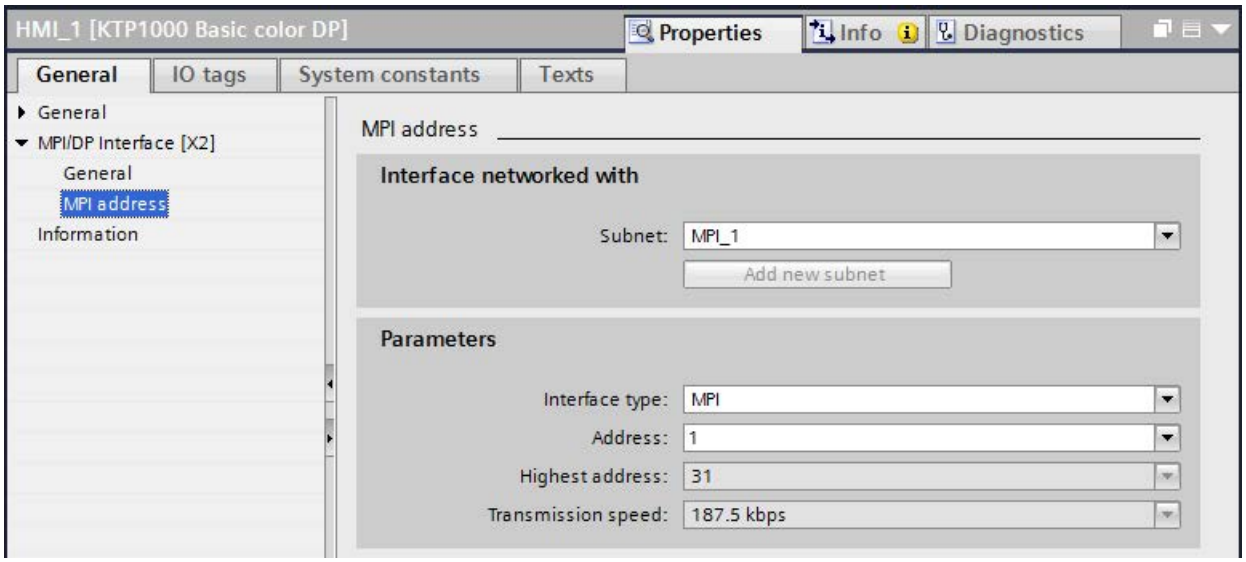
MPI parameters for the HMI device

MPI parameters for the HMI device

An overview of the configured HMI device parameters can be found in the properties for the HMI device.

Displaying and changing MPI parameters of the HMI device

1. Click the HMI device in the "Devices & Networks" editor.
2. Change the parameters of the HMI device in the Inspector window under "Properties > General > General".



"Interface networked with"

In the "Interface networked with" area, select the subnet of the HMI connection via which the HMI device is connected to the network. You use the "Add new subnet" button to create a new subnet.

"Parameters"

- "Interface type"
  You assign the interface type in the "Interface type" area. Depending on the HMI device type, you have various interfaces to choose from.
- "Address"
  You assign the MPI address of the HMI device in the "Address" area. The MPI address must be unique throughout the MPI network.
- "Highest address"
  The "Highest address" area displays the highest address of the MPI network.
- "Transmission speed"
  The "Transmission speed" is determined by the slowest device connected to the network. The setting is identical throughout the network.

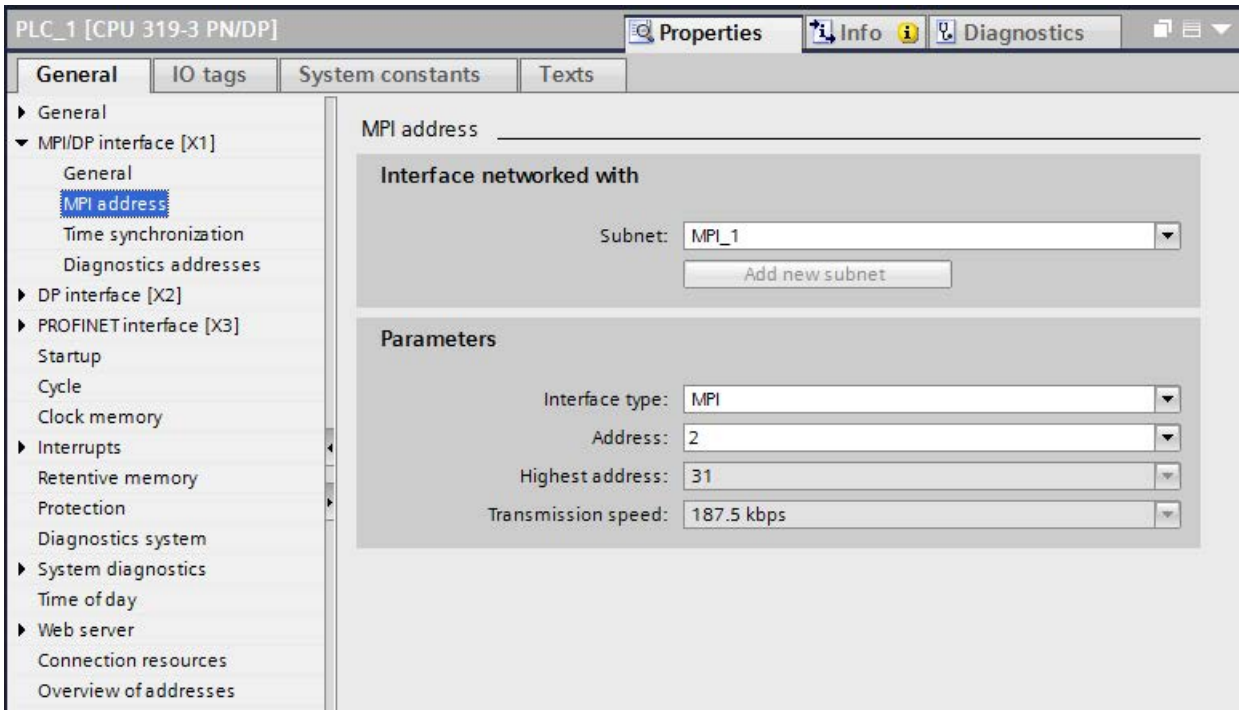
MPI parameters for the PLC

MPI parameters for the PLC

An overview of the configured parameters can be found in the properties for the PLC.

Displaying and changing PROFIBUS parameters of the PLC

1. Click the PLC in the "Devices & Networks" editor.
2. Change the parameters of the PLC in the Inspector window under "Properties > General > General".



"Interface networked with"

In the "Subnet" area, select the subnet of the HMI connection via which the HMI device is connected to the network. You use the "Add new subnet" button to create a new subnet.

"Parameters"

- "Interface type"
  Depending on the HMI device type, you have various interfaces to choose from.
- "Address"
  You assign the MPI address of the HMI device in the "Address" area. The MPI address must be unique throughout the MPI network.
- "Highest address"
  The "Highest address" area displays the highest address of the MPI network.
- "Transmission speed"
  The "Transmission speed" is determined by the slowest device connected to the network. The setting is identical throughout the network.

Addressing of the PLC via MPI

Introduction

Each communication partner must be assigned an MPI network address.

Each S7 module which supports communication functions and is operated the SIMATIC S7-300/400 PLC is assigned a unique MPI address. Only one CPU may be used per rack.

Note

HMI devices cannot be operated with incorrect addressing

Always avoid redundant addressing on the MPI bus.

MPI address of the communication partner of a SIMATIC S7-300

When assigning addresses, you have to distinguish between communication partners with and without separate MPI address.

- If the communications partner has its own MPI address, you only need to define the MPI address.
- If the communication partners do not have a separate MPI address, specify the MPI address of the communications partner used for the connection. In addition, define the slot and rack of a communication partner without its own MPI address.

MPI address of the communication partner of a SIMATIC S7-400

Only S7 modules with an MPI connector are assigned an MPI address. Modules without an MPI connector are addressed indirectly:

- MPI address of the module to which the HMI is connected.
- The slot and the rack of the module with which the HMI device communicates.

10.8.8.5 Data exchange

Data exchange using area pointers

General information on area pointers

Introduction

You use an area pointer to access a data area in the PLC. During communication, the PLC and the HMI device alternately access these data areas for read and write operations.

The PLC and the HMI device trigger defined interactions based on the evaluation of stored data.

Configuration of area pointers

Before you use an area pointer, you enable it under "Connections > Area pointer". You then assign the area pointer parameters.

You can find more detailed information on configuring area pointers in:

Configuring area pointers (Page 8420)

"Screen number" area pointer

Function

The HMI device saves information about the screen called on the HMI device to the "Screen number" area pointer.

This allows the transfer of the current screen contents from the HMI device to the PLC. The PLC can trigger specific reactions such as the call of a different screen.

Use

Configure and enable the area pointer in "Communication > Connections" before you put it into use. You can create only **one** instance of the "Screen number" area pointer and only on **one** PLC.

The screen number is always transferred to the PLC when a new screen is activated or when the focus within a screen changes from one screen object to another.

Structure

The area pointer is a data area in the memory of the PLC with a fixed length of 5 words.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1st word	Current screen type															
2nd word	Current screen number															
3rd word	Reserved															
4th word	Current field number															
5th word	Reserved															

- Current screen type
  "1" for root screen or
  "4" for permanent window
- Current screen number
  1 to 32767
- Current field number
  1 to 32767

"Date/time" area pointer

Function

This area pointer is used to transfer the date and time from the HMI device to the PLC.

The PLC writes control job "41" or "40" to the job mailbox.

When it evaluates the control job, the HMI device writes its current date and the time in the data area configured in the "Date/time" area pointer. All definitions are coded in BCD format.

The "Date/Time" area pointer when used in a project which contains multiple connections must be enabled for each configured connection.

Note

You cannot use the "Date/Time PLC" area pointer if you have configured the "Date/Time" area pointer.

Note

Symbolic addressing is not possible if you are using the "Date/Time" area pointer.

The "Date/Time" area pointer when used in a project which contains multiple connections must be enabled for each configured connection.

The date/time data area has the following structure:

Data word	Most significant byte								Least significant byte								
	7							0	7							0	
n+0	Reserved								Hour (0 to 23)								Time
n+1	Minute (0 to 59)								Second (0 to 59)								
n+2	Reserved								Reserved								
n+3	Reserved								Weekday (1 to 7, 1=Sunday)								Date
n+4	Day (1 to 31)								Month (1 to 12)								
n+5	Year (80 to 99/0 to 29)								Reserved								

Note

When making entries in the "Year" data area, you should note that values 80 to 99 result in years 1980 through 1999, while the values 0 to 29 result in the years 2000 through 2029.

"Date/time PLC" area pointer

Function

This area pointer is used to transfer the date and time from the PLC to the HMI device. Use this area pointer if the PLC is the time master.

The PLC loads the data area of the area pointer. All definitions are coded in BCD format.

The HMI device reads the data cyclically within the configured acquisition cycle and synchronizes itself.

Note

Set an acquisition cycle of sufficient length for the Date/time area pointer in order to avoid any negative impact on HMI device performance.
Recommended: Acquisition cycle of 1 minute, if the process allows this.

You cannot use the "Date/Time" area pointer if you have configured the "Date/Time PLC" area pointer.

"Date/Time PLC" is a global area pointer and may be configured only once per project.

Note

You cannot use the "Date/Time" area pointer if you have configured the "Date/Time PLC" area pointer.

The date/time data area has the following structure:

DATE_AND_TIME format (in BCD code)

Data word	Most significant byte			Least significant byte		
	7	. . . . .	0	7	. . . . .	0
n+0	Year (80 to 99/0 to 29)			Month (1 to 12)		
n+1	Day (1 to 31)			Hour (0 to 23)		
n+2	Minute (0 to 59)			Second (0 to 59)		
n+3	Reserved			Reserved		Weekday (1 to 7, 1=Sunday)
n+4 [1)]	Reserved			Reserved		
n+5 [1)]	Reserved			Reserved		

1) The two data words must exist in the data area to ensure that the data format matches WinCC flexible and to avoid reading false information.

Note

When making entries in the "Year" data area, you should note that values 80 to 99 result in years 1980 through 1999, while the values 0 to 29 result in the years 2000 through 2029.

"Coordination" area pointer

Function

The "Coordination" area pointer is used to implement the following functions:

- Detecting the startup of the HMI device in the control program
- Detecting the current operating mode of the HMI device in the control program
- Detecting whether the HMI device is ready to communicate in the control program

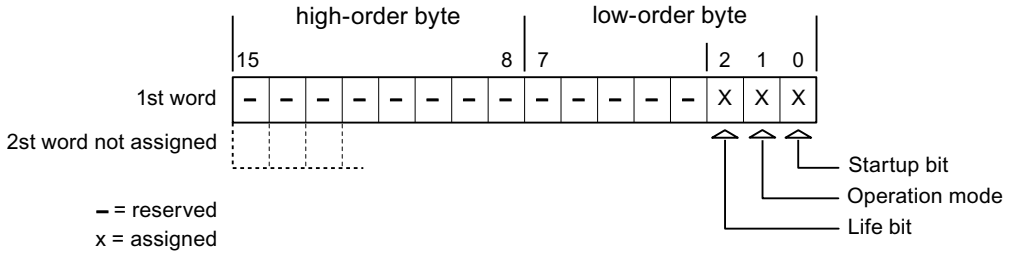
By default, the "Coordination" area pointer has the length of one word and cannot be changed.

Application

Note

The HMI device always writes the entire coordination area when updating the area pointer. The control program can for this reason not make changes to the coordination area.

Assignment of bits in the "Coordination" area pointer



Startup bit

The startup bit is set briefly to "0" by the HMI device during startup. It sets the bit permanently to "1" when startup is completed.

Operating mode

The operating mode bit is set to 1 as soon as the user switches the HMI device offline. The status of the operating mode bit is "0" during normal operation of the HMI device. You can determine the current operating mode of the HMI device by reading this bit in the control program.

Life bit

The HMI device inverts the life bit at intervals of approximately one second. You can check whether or not there is still a connection to the HMI device by querying this bit in the control program.

"Project ID" area pointer

Function

When Runtime starts, a check can be carried out as to whether the HMI device is connected to the correct PLC. This check is important when operating with several HMI devices.

For this, the HMI device compares a value stored on the PLC with the value specified in configuration. This ensures compatibility of the configuration data with the PLC program. If there is no concordance, a system event is given on the HMI device and Runtime is stopped.

Use

Note

HMI connections cannot be switched "online".

The HMI connection in which the "Project ID" area pointer is used must be switched "online".

To use this area pointer, set up the following during the configuration:

- Define the version of the configuration. Values between 1 and 255 are possible.
  You enter the version in the editor "Runtime settings > General" in the "Identification" area.
- Data address of the value for the version that is stored in the PLC:
  You enter the data address in the editor "Communication > Connections" under "Address".

Connection failure

A connection failure to a device on which the "Project ID" area pointer is configured results in all the other connections of the device being switched to "offline".

This behavior has the following requirements:

- You have configured several connections in a project.
- You are using the "Project ID" area pointer in at least one connection.

Causes which may set connections "offline":

- The PLC is not available.
- The connection has been switched offline in the Engineering System.

"Job mailbox" area pointer

Function

The PLC can use the job mailbox to transfer jobs to the HMI device to trigger corresponding actions on the HMI device. These functions include, for example:

- Display screen
- Set date and time

Data structure

The first word of the job mailbox contains the job number. Depending on the job mailbox, up to three parameters can be transferred.

Word	Most significant byte	Least significant byte
n+0	0	Job number
n+1	Parameter 1	
n+2	Parameter 2	
n+3	Parameter 3	

The HMI device evaluates the job mailbox if the first word of this job is not equal to zero. This means that the parameters must be entered in the job mailbox first, followed by the job number.

When the HMI device accepts the job mailbox, the first word is set to 0 again. The execution of the job mailbox is generally not completed at this point in time.

Job mailboxes

All job mailboxes and their parameters are listed below. The "No." column contains the job number of the job mailbox. Job mailboxes can only be triggered by the PLC when the HMI device is online.

No.	Function	
14	Set time (BCD-coded)	
	Parameter 1	Left byte: - Right byte: hours (0-23)
	Parameter 2	Left byte: minutes (0-59) Right byte: seconds (0-59)
	Parameter 3	-
15	Setting the date (BCD coded) [3) 4)]	
	Parameter 1	Left byte: - Right byte: weekday (1-7: Sunday-Saturday)
	Parameter 2	Left byte: day (1-31) Right byte: month (1-12)
	Parameter 3	Left byte: year
23	User logon	
	Logs the user on with the name "PLC user" at the HMI device with the group number transferred in Parameter 1. The logon is possible only when the transferred group number exists in the project.	
	Parameter 1	Group number 1 to 255
	Parameter 2, 3	-
24	User logoff	
	Logs off the current user. (The function corresponds to the "logoff" system function)	
	Parameter 1, 2, 3	-
40	Transfer date/time to PLC	
	(in the S7 format DATE_AND_TIME) An interval of at least 5 seconds must be maintained between two successive jobs to prevent overload of the HMI device.	
	Parameter 1, 2, 3	-
41	Transfer date/time to PLC	
	(In OP/MP format) An interval of at least 5 seconds must be maintained between successive jobs to prevent overload of the HMI device.	
	Parameter 1, 2, 3	-
46	Update tag	
	Causes the HMI device to read the current value of the tags from the PLC whose update ID matches the value transferred in Parameter 1. (Function corresponds to the "UpdateTag" system function.)	
	Parameter 1	1 - 100
49	Delete alarm buffer	
	Deletes all analog alarms and discrete alarms of the "Warnings" class from the alarm buffer.	

No.	Function	
14	**Set time (BCD-coded)**	
	Parameter 1, 2, 3	-
50	**Delete alarm buffer**	
	Deletes all analog alarms and discrete alarms of the "Errors" class from the alarm buffer.	
	Parameter 1, 2, 3	-
51	**Screen selection** [2]	
	Parameter 1	Screen number
	Parameter 2	-
	Parameter 3	Field number
69	**Reading data record from PLC** [1]	
	Parameter 1	Recipe number (1-999)
	Parameter 2	Data record number (1-65535)
	Parameter 3	0: Do not overwrite existing data record 1: Overwrite existing data record
70	**Writing data record to PLC** [1]	
	Parameter 1	Recipe number (1-999)
	Parameter 2	Data record number (1-65535)
	Parameter 3	-

[1]	Only for devices supporting recipes.
[2]	OP 73, OP 77A and TP 177A HMI devices also execute the "Screen selection" job mailbox if the on-screen keyboard is active.
[3]	The weekday is ignored on HMI device KTP 600 BASIC PN.
[4]	The weekday is ignored when you configure the "Date/Time PLC" area pointer.

"Data record" area pointer

"Data record" area pointer

Function

When data records are transferred between the HMI device and PLC, both partners access common communications areas on the PLC.

Data transfer types

There are two ways of transferring data records between the HMI device and PLC:

- Transfer without synchronization
- Transfer with synchronization via the data mailbox

Data records are always transferred directly, which means that the tag values are read straight from an address or written straight to an address configured for this tag without being redirected via an interim memory.

Initiating the transfer of data records

There are three ways of triggering the transfer:

- Operator input in the recipe view
- Job mailboxes
  The transfer of data records can also be triggered by the PLC.
- Triggering by configured functions

If the transfer of data records is triggered by a configured function or by a job mailbox, the recipe view on the HMI device remains operable. The data records are transferred in the background.

Simultaneous processing of several transfer requests is, however, not possible. In this case, the HMI device rejects the other transfer requests with a system alarm.

Transfer without synchronization

If you select asynchronous transfer of data records between the HMI device and PLC, there is no coordination over the common data areas. It is therefore unnecessary to set up a data area during configuration.

Asynchronous data record transfer can be a useful alternative, for example when:

- The system is capable of excluding the risk of uncontrolled overwriting of data by the communication peer.
- The PLC does not require information about the recipe number and data record number.
- The transfer of data records is triggered by the operator of the HMI device.

Reading values

When a read job is triggered, the values are read from the PLC addresses and transferred to the HMI device.

- Triggering by the operator in the recipe view:
  The values are downloaded to the HMI device. You can then, for example, process, edit, or save these values in the HMI device.
- Triggering by a function or job mailbox:
  The values are saved immediately to the data volume.

Writing values

When a write job is triggered, the values are written to the PLC addresses.

- Triggering by the operator in the recipe view:
  The current values are written to the PLC.
- Triggering by a function or job mailbox:
  The current values are written to the PLC from the data medium.

Transfer with synchronization

If you select synchronous transfer, both communication partners set status bits in the common data area. You can use this mechanism to prevent uncontrolled overwriting of data in either direction in your control program.

Application

Synchronous data record transfer can be a useful solution, for example, when:

- The PLC is the "active partner" in the transfer of data records.
- The PLC evaluates the information about the recipe number and data record number.
- The transfer of data records is triggered by means of a Job mailbox.

Requirements

In order to synchronize transfer of data records between the HMI device and the PLC, the following requirements must be met during configuration:

- An area pointer has been set up: "Communication > Connections" editor in "Area pointer".
- The PLC with which the HMI device synchronizes transfer of data records is specified in the recipe:
  "Recipes" editor in the inspector window the option "Coordinated transfer of data records" under "General > Synchronization > Settings"

Structure of the data area

The data area has a fixed length of 5 words. Structure of the data area:

	15		0
1. Word	Current recipe number (1 - 999)		
2. Word	Current data record number (0 - 65535)		
3. Word	Reserved		
4. Word	Status (0, 2, 4, 12)		
5. Word	Reserved		

- Status
  The status word (word 4) can adopt the following values:

Value		Meaning
Decimal	Binary	
0	0000 0000	Transfer permitted, data mailbox free
2	0000 0010	Transferring
4	0000 0100	Transfer completed without error
12	0000 1100	Transfer completed with error

Sequence of a transfer started by the operator in the recipe view

Reading from the PLC started by the operator in the recipe view

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe number to be read and the status "Transferring" in the data record and sets the data record number to 0.	Abort with system event.
3	The HMI device reads the values from the PLC and displays them in the recipe view. If the recipes have synchronized tags, the values from the PLC are also written to the tags.	
4	The HMI device sets the status "Transfer completed."	
5	The control program must reset the status word to zero in order to enable further transfers.	

Writing to the PLC started by the operator in the recipe view

Step	Action	
	Check: Status word = 0?	
1	Yes	No
	The HMI device enters the recipe and data record number to be written and the status "Transferring" in the data record.	Abort with system event.
2	The HMI device writes the current values to the PLC. If the recipes have synchronized tags, the changed values are synchronized between the recipe view and tags and then written to the PLC.	
3	The HMI device sets the status "Transfer completed."	
4	If required, the control program can now evaluate the transferred data.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Note

The status word may only be set by the HMI device. The PLC may only reset the status word to zero.

Note

The PLC may only evaluate the recipe and data record numbers when data inconsistency is detected if one of the conditions outlined below has been met:

- The data mailbox status is set to "Transfer completed".
- The data mailbox status is set to "Transfer completed with error".

Sequence of the transfer triggered by a job mailbox

The transfer of data records between the HMI device and the PLC can be initiated by either one of these stations.

The two job mailboxs No. 69 and No. 70 are available for this type of transfer.

No. 69: Read data record from PLC ("PLC → DAT")

Job mailbox no. 69 transfers data records from the PLC to the HMI device. The job mailbox is structured as follows:

	Left byte (LB)	Right byte (RB)
Word 1	0	69
Word 2	Recipe number (1-999)	
Word 3	Data record number (1 to 65535)	
Word 4	Do not overwrite existing data record: 0 Overwrite existing data record: 1	

No. 70: Write data record to PLC ("DAT → PLC")

Job mailbox No. 70 transfers data records from the HMI device to the PLC. The job mailbox is structured as follows:

	Left byte (LB)	Right byte (RB)
Word 1	0	70
Word 2	Recipe number (1-999)	
Word 3	Data record number (1-65,535)	
Word 4	—	

Sequence when reading from the PLC with job mailbox "PLC → DAT" (no. 69)

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the job and the status "Transferring" in the data record.	Abort without return message.
3	The HMI device reads the values from the PLC and saves these to the data record defined in the job mailbox.	
4	• If "Overwrite" was selected in the job, an existing data record is overwritten without any prompt for confirmation. The HMI device sets the status "Transfer completed." • If "Do not overwrite" was selected in the job, and the data record already exists, the HMI device aborts the job and enters 0000 1100 in the status word of the data mailbox.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Sequence of writing to the PLC with job mailbox "DAT → PLC" (no. 70)

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the job and the status "Transferring" in the data mailbox.	Abort without return message.
3	The HMI device fetches the values of the data record specified in the function from the data medium and writes the values to the PLC.	
4	The HMI device sets the status "Transfer completed."	
5	The control program can now evaluate the transferred data. The control program must reset the status word to zero in order to enable further transfers.	

Sequence of the transfer when triggered by a configured function

Reading from the PLC using a configured function

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the function and the status "Transferring" in the data mailbox.	Abort with system event.
3	The HMI device reads the values from the PLC and stores them in the data record specified in the function.	

Step	Action	
4	• If "Yes" was selected for the "Overwrite" function, an existing data record is overwritten without any prompt for confirmation. The HMI device sets the status "Transfer completed." • If "No" was selected for the "Overwrite" function and the data record already exists, the HMI device aborts the job and enters 0000 1100 in the status word of the data mailbox.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Writing to the PLC by means of configured function

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the function and the status "Transferring" in the data mailbox.	Abort with system event.
3	The HMI device fetches the values of the data record specified in the function from the data medium and transfers the values to the PLC.	
4	The HMI device sets the status "Transfer completed."	
5	The control program can now evaluate the transferred data. The control program must reset the status word to zero in order to enable further transfers.	

Possible causes of error when transferring data records

Possible causes of error

The section below shows possible error causes which lead to the cancellation of data record transfer:

- Tag address not set up on the PLC
- Overwriting data records not possible

- Recipe number does not exist
- Data record number does not exist

Note

The status word may only be set by the HMI device. The PLC may only reset the status word to zero.

Note

The PLC may only evaluate the recipe and data record numbers when data inconsistency is detected if one of the conditions outlined below has been met:

- The data mailbox status is set to "Transfer completed".
- The data mailbox status is set to "Transfer completed with error".

Reaction to an aborted transfer due to errors

If the transfer of data records is aborted due to errors, the HMI device reacts as follows:

- Triggering by the operator in the recipe view
  Information in the status bar of the recipe view and output of system alarms
- Triggered by function
  Output of system alarms
- Triggering by job mailbox
  No return message on the HMI device

You can nonetheless evaluate the status of the transfer by querying the status word in the data record.

Trends

Trends

A trend is the graphical representation of one or more values from the PLC. The value is read out time-triggered for Basic Panels.

For additional information see:

Configuring trend displays for values from the PLC (Page 8147)

Time-triggered trends

The HMI device reads in the trend values cyclically at an interval specified in the configuration.

Time-triggered trends are suitable for continuous curves, such as the operating temperature of a motor.

Alarms

Configuring alarms

Configure alarms

Several steps are needed to configure alarms, such as operational messages, error alarms, and acknowledgement.

- Step 1: Create tags
- Step 2: Configure alarms
- Step 3: Configure acknowledgment

You can find additional information in the section:

Working with alarms (Page 8157)

Distinctive features when configuring alarms

If you are configuring connections of HMI devices to PLCs of other manufacturers, note the following distinctive features when configuring:

- Data types of the tags
- Addressing of tags
- How the bit positions are counted

Data types

For connections with a SIMATIC communication driver, the following data types are supported:

PLC	Permitted data types	
	Discrete alarms	Analog alarms
SIMATIC S7 300/400	WORD, INT	BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, COUNTER, TIME

How the bit positions are counted

For connections with a SIMATIC communication driver, the following counting method applies:

How the bit positions are counted	Byte 0								Byte 1							
	Most significant byte								Least significant byte							
In SIMATIC S7 PLCs	7							0	7							0
In WinCC you configure:	15							8	7							0

Acknowledgment of alarms

Procedure

Create suitable tags on the PLC to acknowledge an error alarm. You assign these tags to an alarm in the "Bit messages" editor. You make the assignment in "Properties > Acknowledgment".

Distinction in terms of acknowledgment:

- Acknowledgment by the PLC
- Acknowledgment on the HMI device

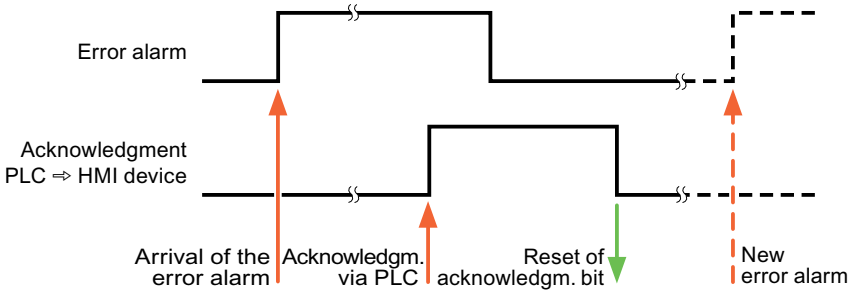
Acknowledgment by the PLC

In "PLC acknowledgment tag", configure the tag or array tag and the bit number that the HMI device uses to identify a PLC acknowledgment.

A bit set in the tag triggers acknowledgment of the assigned error alarm bit at the HMI device. This tag bit returns a function similar to acknowledgment on the HMI device which is triggered by pressing the "ACK" button, for example.

The acknowledgment bit must be located in the same tag as the bit for the error alarm.

Reset the acknowledgment bit before setting the bit in the alarm area again. The figure below shows the pulse diagram.



Acknowledgment on the HMI device

In the "HMI acknowledgment tag" area, configure the tag or array tag as well as the bit number that the HMI device writes to the PLC after acknowledgment. Make sure when you use an array tag that it is not longer than 6 words.

To always create a signal change when setting an assigned acknowledgment bit of a discrete alarm that must be acknowledged, the HMI device will reset the acknowledgment bit assigned to the alarm as soon as it detects an alarm subject to acknowledgment and write the acknowledgment tag in the PLC. There will be a certain delay between detecting the message and writing the acknowledgment tag in the PLC because the HMI device has to process the operations.

If a discrete alarm subject to acknowledgment is acknowledged by the HMI device, then the corresponding bit in the assigned acknowledgment tag will be set. The entire acknowledgment

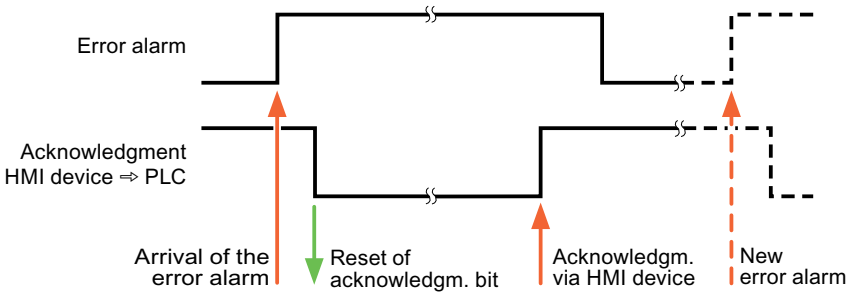
tag is then written to the PLC by the HMI device. This allows the PLC to recognize that a certain alarm message has been acknowledged at the HMI device.

Note

All alarm bits acknowledged since the last Runtime start will remain in the acknowledgment tag until a new incoming of the respective discrete alarms is detected.

This area should only be read by the PLC because the entire section of the HMI device will be overwritten once the next acknowledgment tag is written.

The figure below shows the pulse diagram.



10.8.8.6 Performance features of communication

Permitted data types for SIMATIC S7 300/400

Permitted data types for connections with SIMATIC S7 300/400

The table lists the data types that can be used when configuring tags and area pointers.

Data type	Length
BOOL	1-bit
BYTE	1 byte
WORD	2 bytes
DWORD	4 bytes
CHAR	1 byte
INT	2 byte
DINT	4 bytes
REAL	4 bytes
TIME	4 bytes
DATE	2 bytes
TIME_OF_DAY, TOD	4 bytes
S5TIME	2 bytes

Data type	Length
COUNTER	2 bytes
TIMER	2 bytes
DATE_AND_TIME	8 bytes
STRING	(2+n) bytes, n = 0 to 254

10.8.8.7 Creating connections in the "Connections" editor

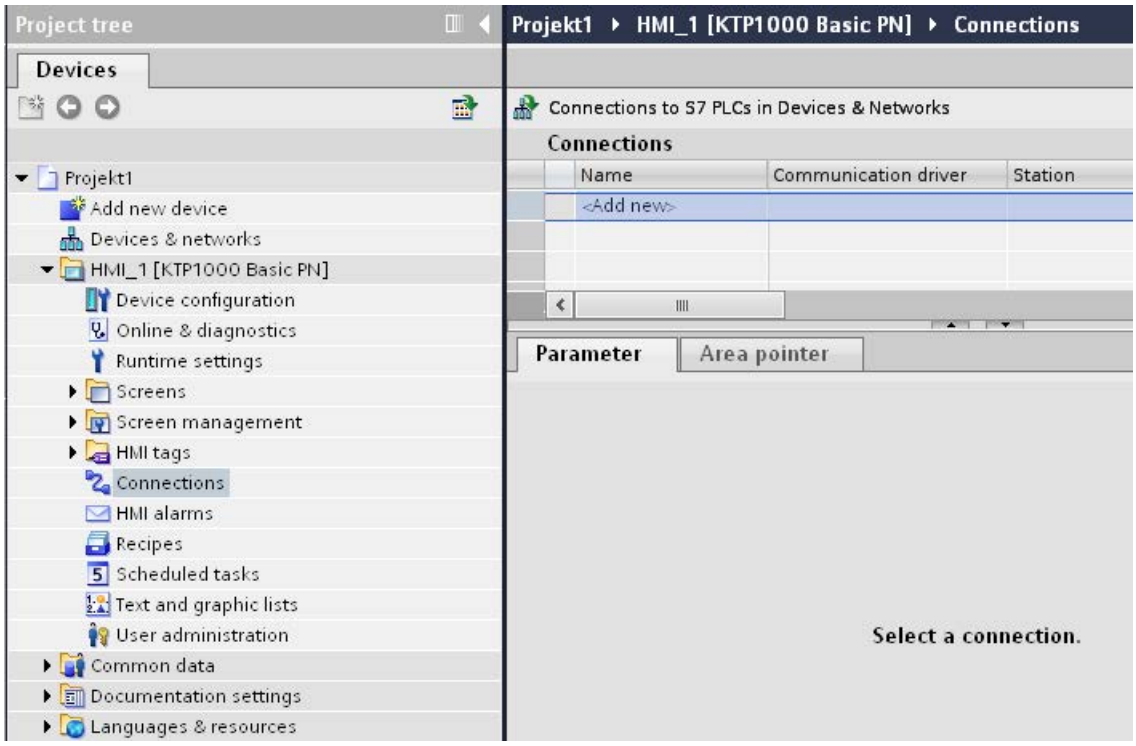
Creating a PROFINET connection

Requirements

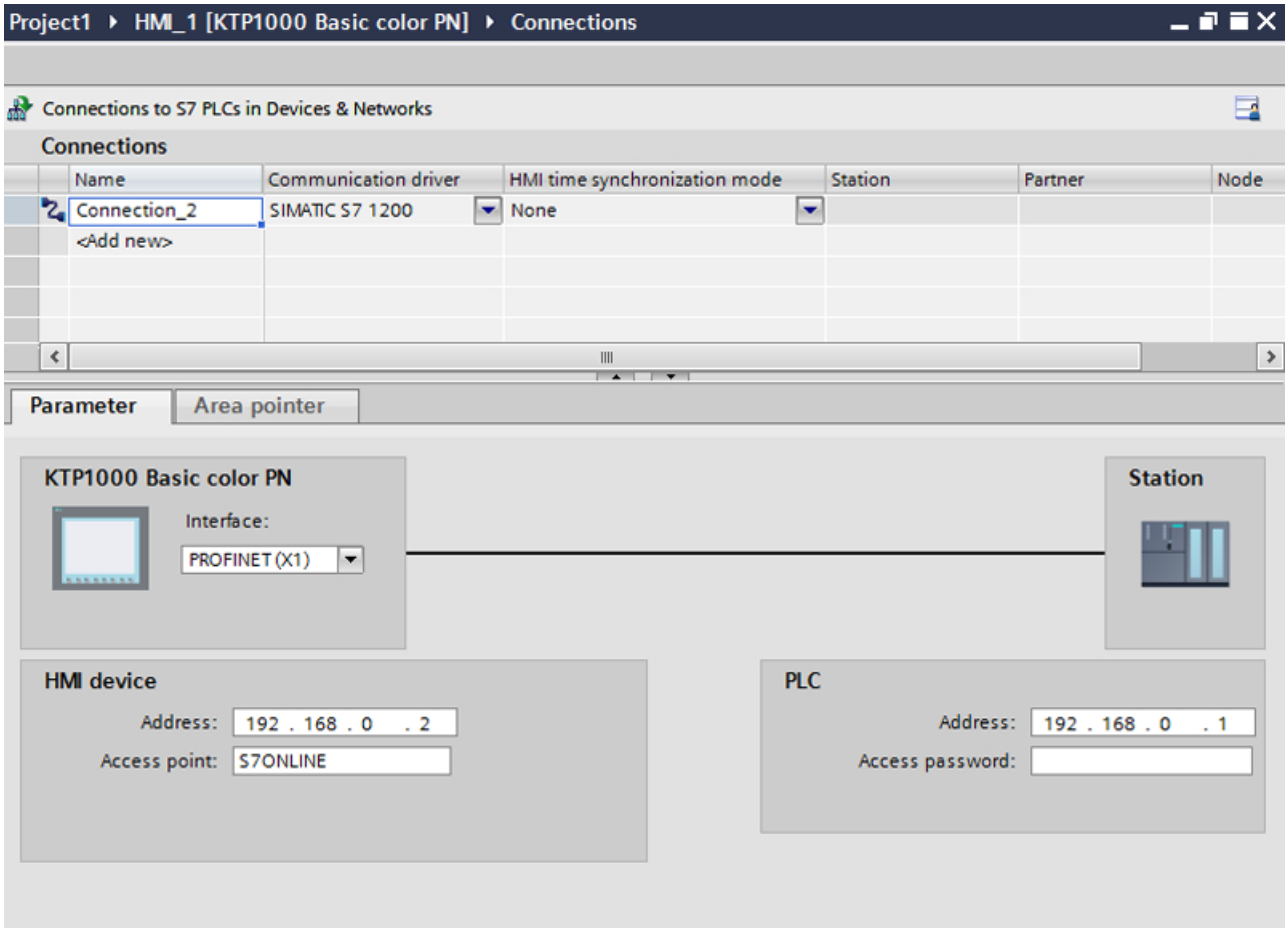
- A project is open.
- An HMI device with a PROFINET interface has been created.

Procedure

1. Open the "Connections" editor of the HMI device.
2. Double-click "<Add>".



3. Select the driver in the "Communication driver" column.



4. Click the name of the connection.
5. Select a PROFINET interface of the HMI device in the Inspector window under "Parameters > Interface".
6. Set the IP addresses of the communication partners in the Inspector window:
   – HMI device: "Parameters > HMI device > Address"
   – PLC: "Parameters > PLC > Address"

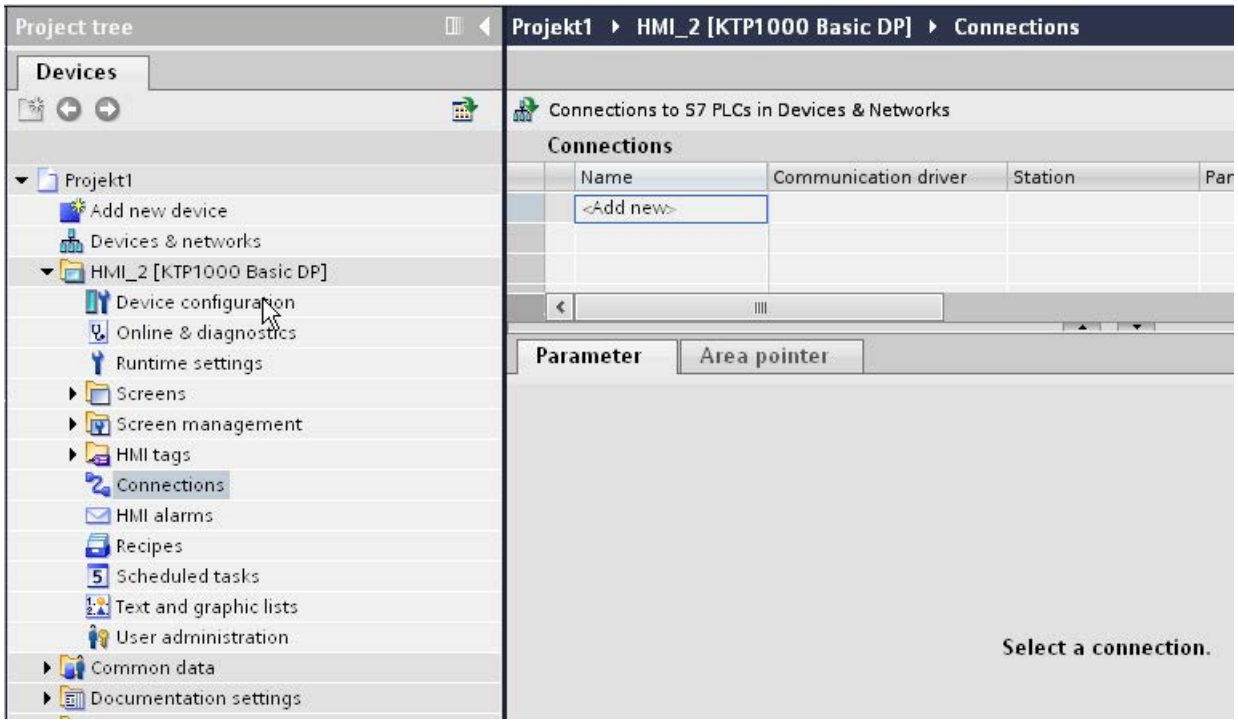
Creating a PROFIBUS connection

Requirements

- A project is open.
- An HMI device with a PROFIBUS interface has been created.

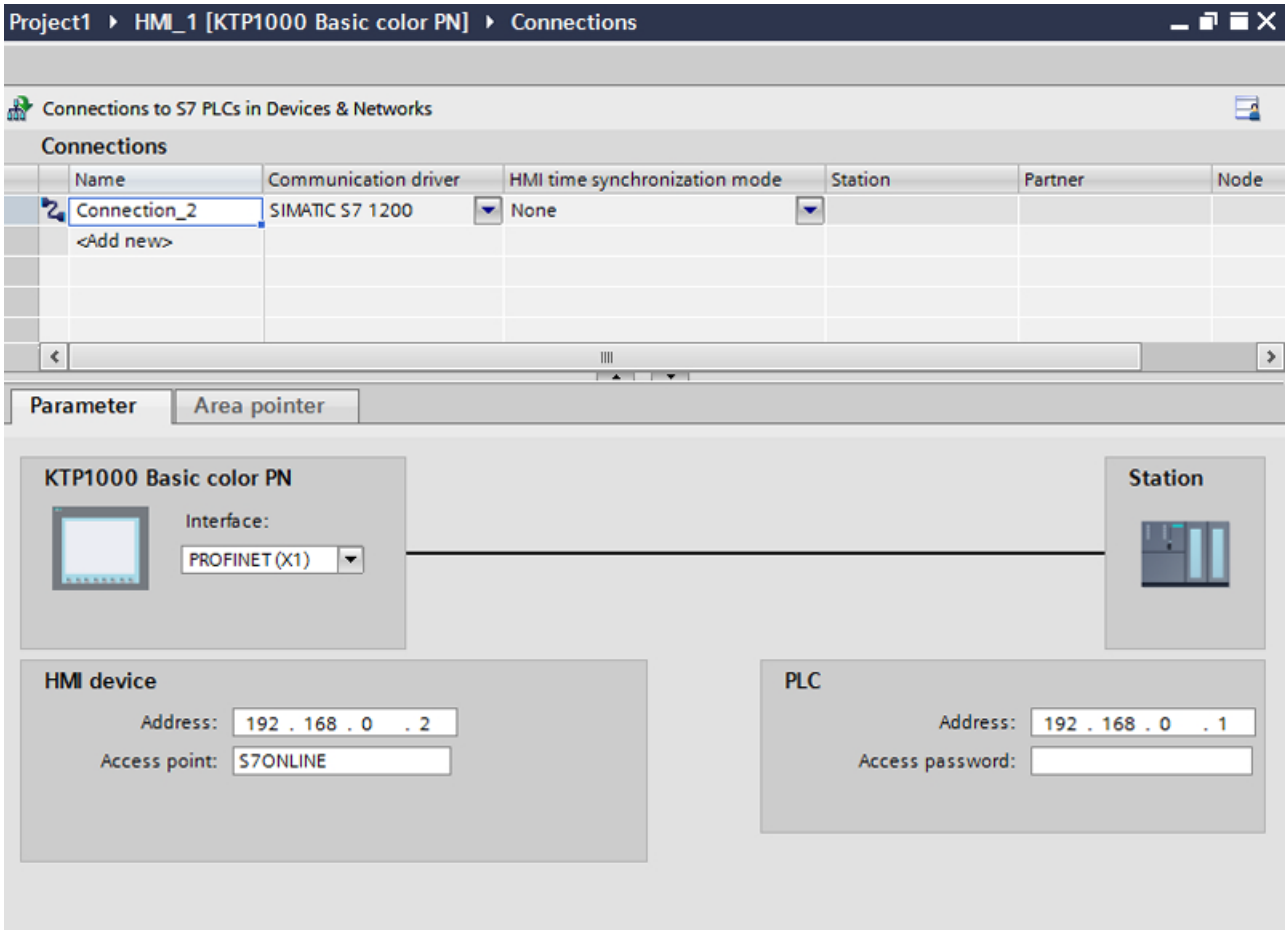
Procedure

1. Open the "Connections" editor of the HMI device.
2. Double-click "<Add>".



3. Select the driver in the "Communication driver" column.
4. Click the name of the connection.
5. Select the "MPI/DP" interface in the Inspector window under "Parameters > Interface".

6. Select the "DP" profile in the Inspector window under "Parameters > Network".



7. Set the addresses of the communication partners in the inspector window:
   - HMI device: "Parameters > HMI device > Address"
   - PLC: "Parameters > PLC > Address"

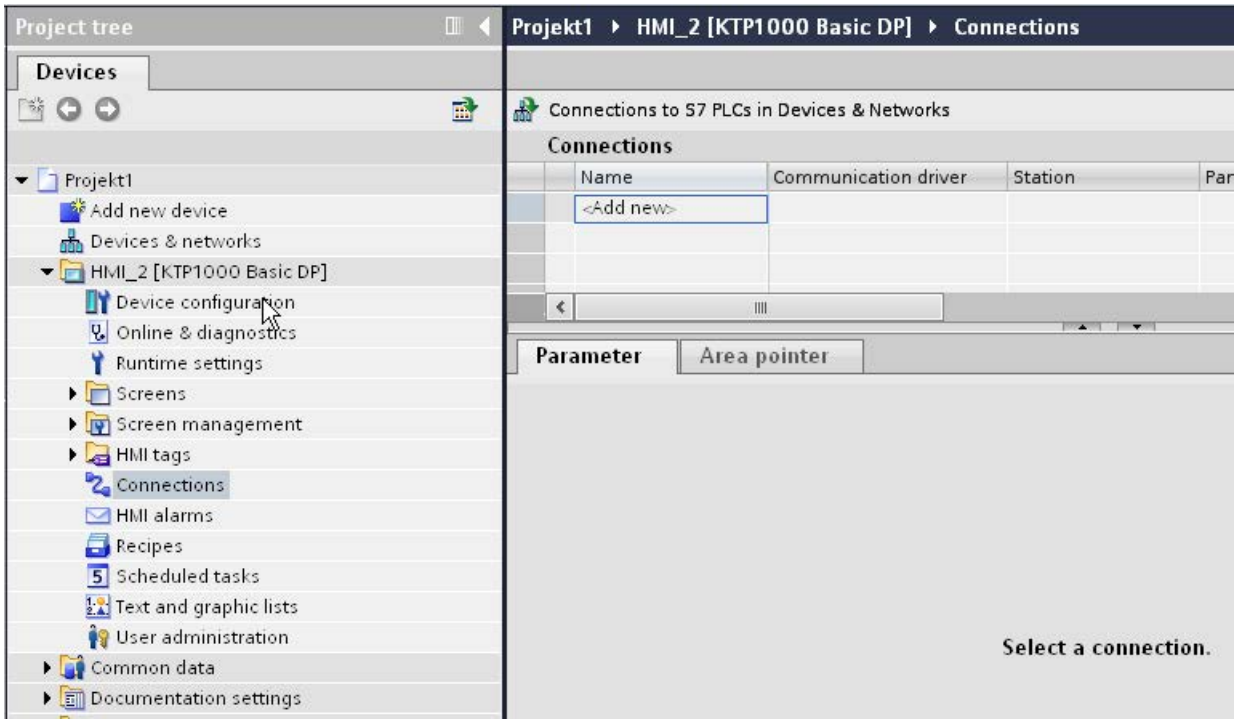
Creating an MPI connection

Requirements

- A project is open.
- An HMI device with an MPI interface has been created.

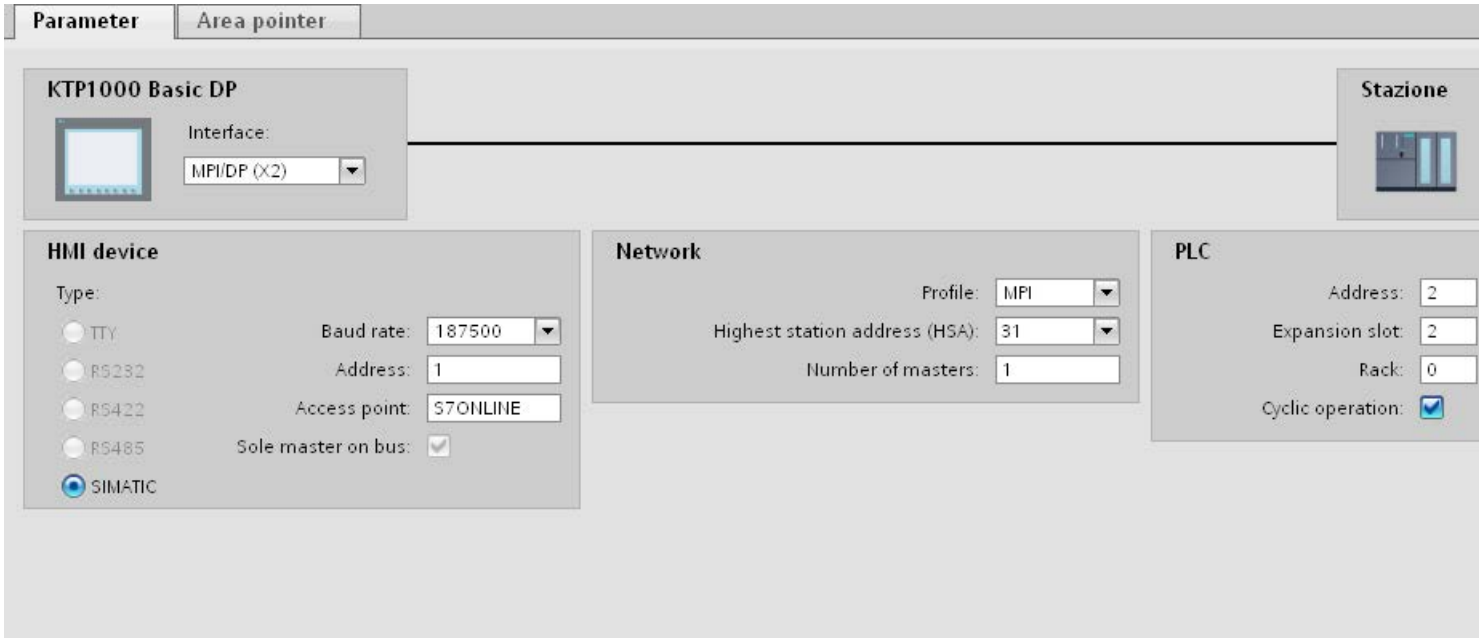
Procedure

1. Open the "Connections" editor of the HMI device.
2. Double-click "<Add>".



3. Select the driver in the "Communication driver" column.
4. Click the name of the connection.
5. Select the "MPI/DP" interface in the Inspector window under "Parameters > Interface".

6. Select the "MPI" profile in the Inspector window under "Parameters > Network".



7. Set the addresses of the communication partners in the inspector window:
   - HMI device: "Parameters > HMI device > Address"
   - PLC: "Parameters > PLC > Address"

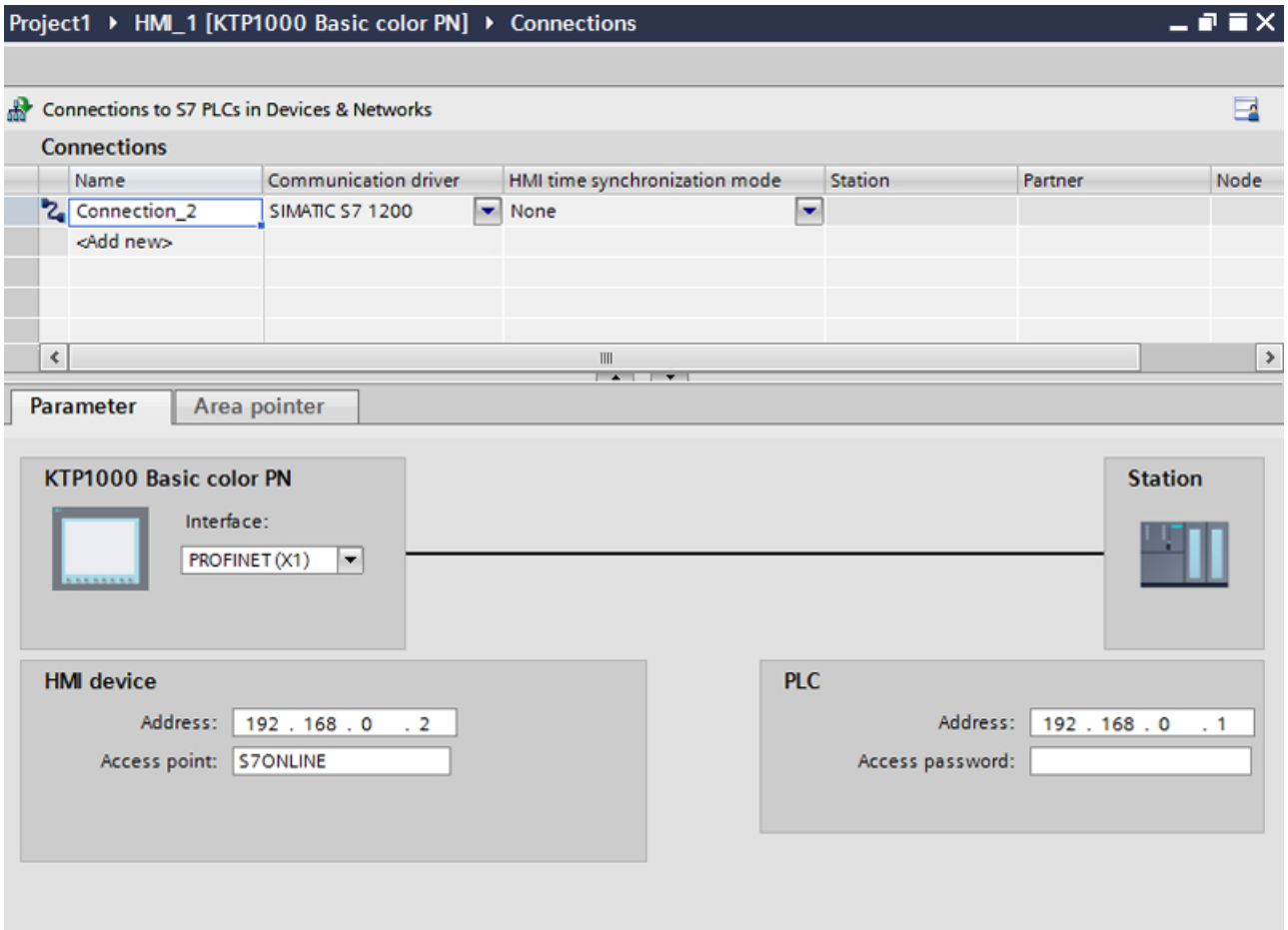
Parameters for the connection

Parameters for the connection (SIMATIC S7 300/400)

Parameters to be set

To set the connection parameters, such as addresses and profiles, click the connection that you have created in the "Connections" editor.

The communication partners are displayed schematically in the Inspector window under "Parameters". The "HMI device", "Network", and "PLC" areas are available for assigning parameters according to the interface used.


Project1 ▸ HMI_1 [KTP1000 Basic color PN] ▸ Connections
Connections to S7 PLCs in Devices & Networks
Connections
Name
Communication driver
HMI time synchronization mode
Station
Partner
Node
Connection_2
SIMATIC S7 1200
None
<Add new>
Parameter
Area pointer
KTP1000 Basic color PN
Interface:
PROFINET (X1)
Station
HMI device
Address: 192 . 168 . 0 . 2
Access point: S7ONLINE
PLC
Address: 192 . 168 . 0 . 1
Access password:

Ethernet parameters

Parameters for the HMI device

You set the parameters for the HMI device in the network under "HMI device".. The changes are not transferred automatically to the HMI device. You must change the settings in the Control Panel of the HMI device.

- "Interface"
  If you are directly connected to the HMI device during configuration, you can set up the IP address of the HMI device in WinCC.

  Note

  The IP address in the Control Panel will be overwritten upon subsequent loading if you have already set up the IP address in the HMI device control panel.

  The IP address already set up in the Control Panel will be retained upon subsequent loading if you activate "Set IP address using a different method".

  The IP address is transferred to the HMI device during project transfer.
  To set up the IP address of the HMI device:

  – Click the HMI device.
  – Open the "Device configuration" editor.
  – Click the Ethernet interface.
  – Assign the IP address in the inspector window under:
    "General > PROFINET interface > Ethernet addresses"

- "Address"
  You assign the IP address of the HMI device in the "Address" area.
  When you transfer the WinCC project to the HMI device, this IP address is set up directly in the HMI device.
- "Access point"
  The access point defines a logical device name through which the communication partner can be reached.

Parameters for the PLC

Under "PLC", you address the S7 module with which the HMI device will exchange data. Assign a name for the connection for each communication partner.

- "Address"
  Under "Address", set the IP address of the S7 module to which the HMI device is connected.
- "Expansion slot"
  Defines the number of the expansion slot of the CPU to be addressed.

- "Rack"
  Defines the rack number of the CPU to be addressed.
- "Cyclic operation"

  Note

  The setting "Cyclic operation" cannot be configured for the SIMATIC S7 1200 PLC.

  When cyclic operation is enabled, the PLC optimizes the data exchange between the HMI device and the PLC. This increases system performance.
  Disable cyclic mode if you are operating several HMI devices in parallel.

PROFIBUS parameters

Parameters for the HMI device

You assign the parameters for the HMI device in the network once under "HMI device". The change applies to each communication partner.

- "Type"
  Specifies the physical connection used.
- "Interface"
  For "Interface", you select the HMI device interface via which the HMI device is connected to the PROFIBUS network.
- "Baud rate"
  For "Baud rate", you set the transmission speed of the data in the network. The baud rate is determined by the slowest HMI device connected to the network. The setting must be identical throughout the network.

  Note

  If you set a baud rate of 1.5 Mbaud for OP 73 or OP 77A, the highest station address must be less than or equal to 63.

- "Address"
  You set the PROFIBUS address of the HMI device under "Address". The PROFIBUS address must be unique in the PROFIBUS network.
- "Sole master on bus"
  Disables an additional safety feature against bus faults when the HMI device is connected to the network. A passive station (slave) can only send data if it is requested to do so by an active station (master).
  In S7 200, you must set an HMI device as the master.
- "Access point"
  The access point defines a logical device name through which the communication partner can be reached.

Parameters for the network

Under "Network", you set the parameters for the PROFIBUS network to which the HMI device is linked.

- "Profile"
  For "Profile", you select the network profile that is used in the network. In "Profile", set "DP", "Universal", or "Standard". The setting must be identical throughout the network.
- "Highest address"
  For "Highest station address", set the highest station address. The highest station address must be greater than or equal to the highest actual PROFIBUS address. The setting must be identical throughout the network.

  Note

  If you set a baud rate of 1.5 Mbaud in the OP 73 or the OP 77A, the highest station address must be less than or equal 63.

- "Number of masters"
  For "Number of masters", set the number of masters in the PROFIBUS network. This information is necessary to correctly calculate the bus parameters.

Parameters for the PLC

Under "PLC", you address the S7 module with which the HMI device will exchange data. Assign a name for the connection for each communication partner.

- "Address"
  For "Address", set the PROFIBUS address of the S7 module (CPU, FM, or CP) to which the HMI device is connected.
- "Cyclic operation"

  Note

  The setting "Cyclic operation" cannot be configured for the SIMATIC S7 1200 PLC.

  When cyclic operation is enabled, the PLC optimizes the data exchange between the HMI device and the PLC. This increases system performance. Disable cyclic mode if you are operating several HMI devices in parallel. This setting is not required for SIMATIC S7-200 PLCs.

MPI parameters

Parameters for the HMI device

You assign the parameters for the HMI device in the network once under "HMI device". The change applies to each communication partner.

- "Type"
  Specifies the physical connection used.
- "Interface"
  For "Interface", you select the HMI device interface via which the HMI device is connected to the MPI network.
- "Baud rate"
  For "Baud rate", you set the transmission speed of the data in the network. The baud rate is determined by the slowest HMI device connected to the network. The setting must be identical throughout the network.

  Note

  If you set a baud rate of 1.5 Mbaud for OP 73 or OP 77A, the highest station address must be less than or equal to 63.

- "Address"
  For "Address", you set the MPI address of the HMI device. The MPI address must be unique throughout the MPI network.
- "Sole master on bus"
  Disables an additional safety feature against bus faults when the HMI device is connected to the network. A passive station (slave) can only send data if it is requested to do so by an active station (master). If you have connected only slaves to the HMI device, you must therefore disable the "Sole master on bus" safety feature.
  In S7 200, you must set an HMI device as the master.

Parameters for the network

Under "Network", you set the parameters for the MPI network to which the HMI device is linked.

- "Profile"
  For "Profile", you select the network profile that is used in the network. In "Profile", set "MPI". The setting must be identical throughout the network.
- "Highest address"
  For "Highest station address", set the highest station address. The highest station address must be greater than or equal to the highest actual MPI address. The setting must be identical throughout the network.
- "Number of masters"
  This setting is not required for MPI.

Parameters for the PLC

Under "PLC", you address the S7 module with which the HMI device will exchange data. Assign a name for the connection for each communication partner.

- "Address"
  For "Address", set the MPI address of the S7 module (CPU, FM, or CP) to which the HMI device is connected.
- "Cyclic mode"
  When cyclic mode is enabled, the PLC optimizes the data exchange between the HMI device and the PLC. This improves system performance. Disable cyclic mode if you are operating several HMI devices in parallel. This setting is not required for SIMATIC S7-200.

Cyclic operation

Handling the "Cyclic operation" selection

If "Cyclic operation" is enabled, the HMI device sends a message frame to the CPU at the beginning of communication indicating that certain tags are required on a recurring basis.

The CPU then always transmits the data at the same cyclic interval. This saves the HMI device from having to output new requests for the data.

If cyclic operation is disabled, the HMI device sends a request whenever information is required.

Additional properties:

- Cyclic operation reduces data transmission load at the HMI device. The PLC resources are used to relieve load on the HMI device.
- The PLC only supports a certain number of cyclic services. The HMI device handles the operation if the PLC cannot provide any further resources for cyclic services.
- The HMI device generates the cycle if the PLC does not support the cyclic mode.
- Screen tags are not integrated in cyclic operation.
- Cyclic mode is only set up at the restart of Runtime.
- The HMI device transfers several jobs to the PLC if cyclic mode is enabled, depending on the PLC.
- The HMI device only transfers one job to the PLC if cyclic mode is disabled.

10.8.9 Communicating with SIMATIC S7 200

10.8.9.1 Communication with SIMATIC S7 200

Introduction

This section describes the communication between an HMI device and the SIMATIC S7 200 PLC.

You can configure the following communication channels for the SIMATIC S7 200 PLC:

- PROFINET and Ethernet
- PROFIBUS
- MPI
- PPI

HMI connection for communication

You configure connections between the HMI device and a SIMATIC S7 200 in the "Connections" editor of the HMI device.

10.8.9.2 Creating a connection to SIMATIC S7 200

Introduction

You configure a connection to the SIMATIC S7 200 PLC in the "Connections" editor of the HMI device. The interfaces are named differently depending on the HMI device.

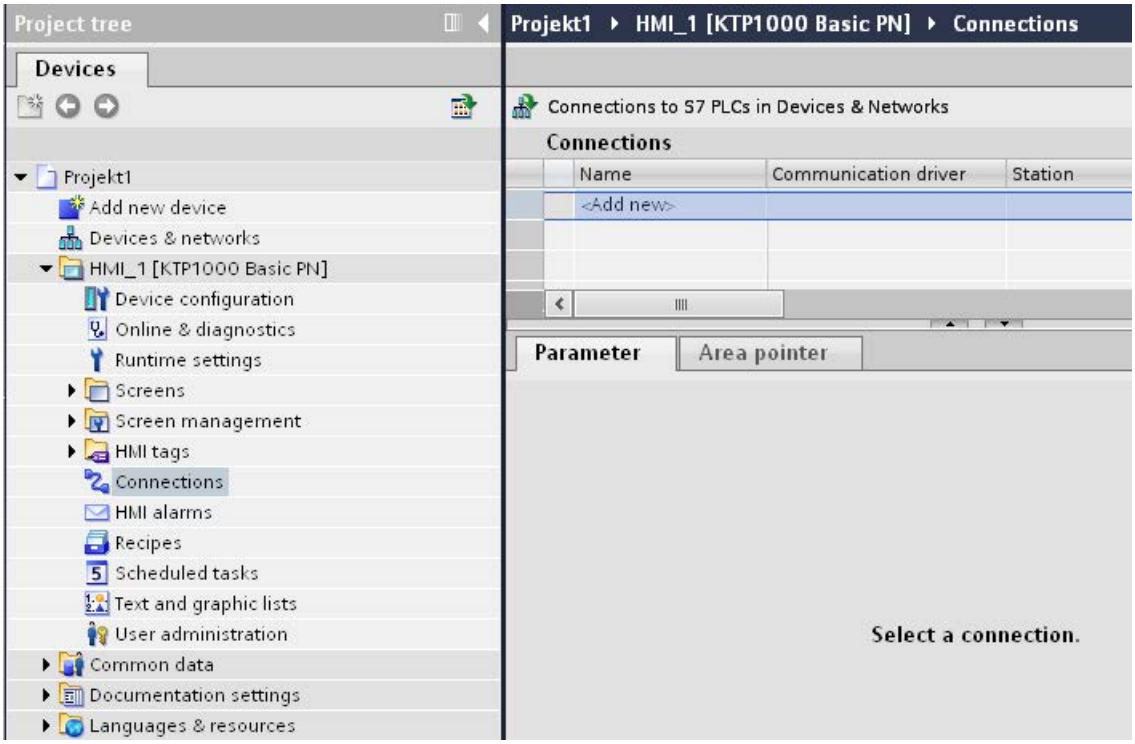
Requirements

- A project is open.
- An HMI device has been created.

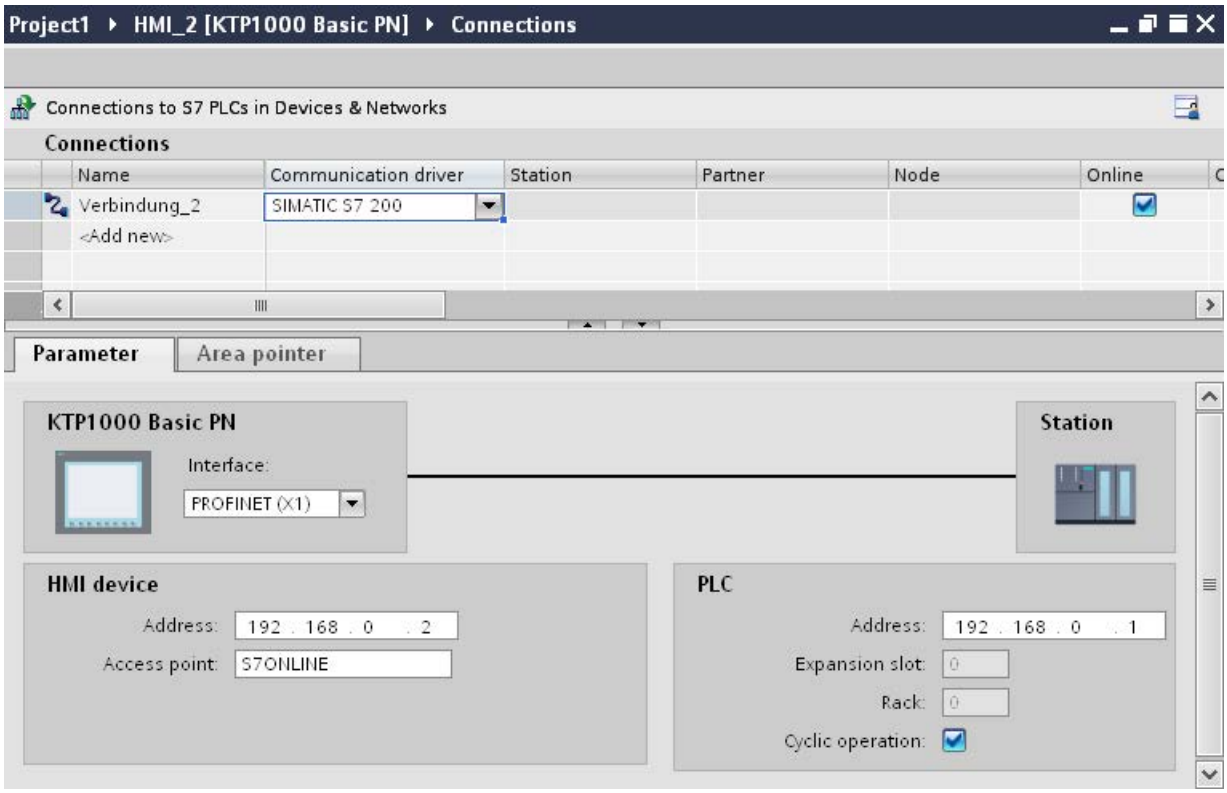
Procedure

1. Double-click the HMI device under "Devices" in the project tree.
2. Double-click the "Connections" item.

3. Double-click "<Add>" in the "Connections" editor.



4. In the "Communication drivers" column, select the "SIMATIC S7 200" driver.
5. Select all necessary connection parameters for the interface in the Inspector window under "Parameters".



See the chapter "Parameters for the connection (Page 8593)" for additional details.

10.8.9.3 Parameters for the connection

Cyclic operation

Handling the "Cyclic operation" selection

If "Cyclic operation" is enabled, the HMI device sends a message frame to the CPU at the beginning of communication indicating that certain tags are required on a recurring basis.

The CPU then always transmits the data at the same cyclic interval. This saves the HMI device from having to output new requests for the data.

If cyclic operation is disabled, the HMI device sends a request whenever information is required.

Additional properties:

- Cyclic operation reduces data transmission load at the HMI device. The PLC resources are used to relieve load on the HMI device.
- The PLC only supports a certain number of cyclic services. The HMI device handles the operation if the PLC cannot provide any further resources for cyclic services.

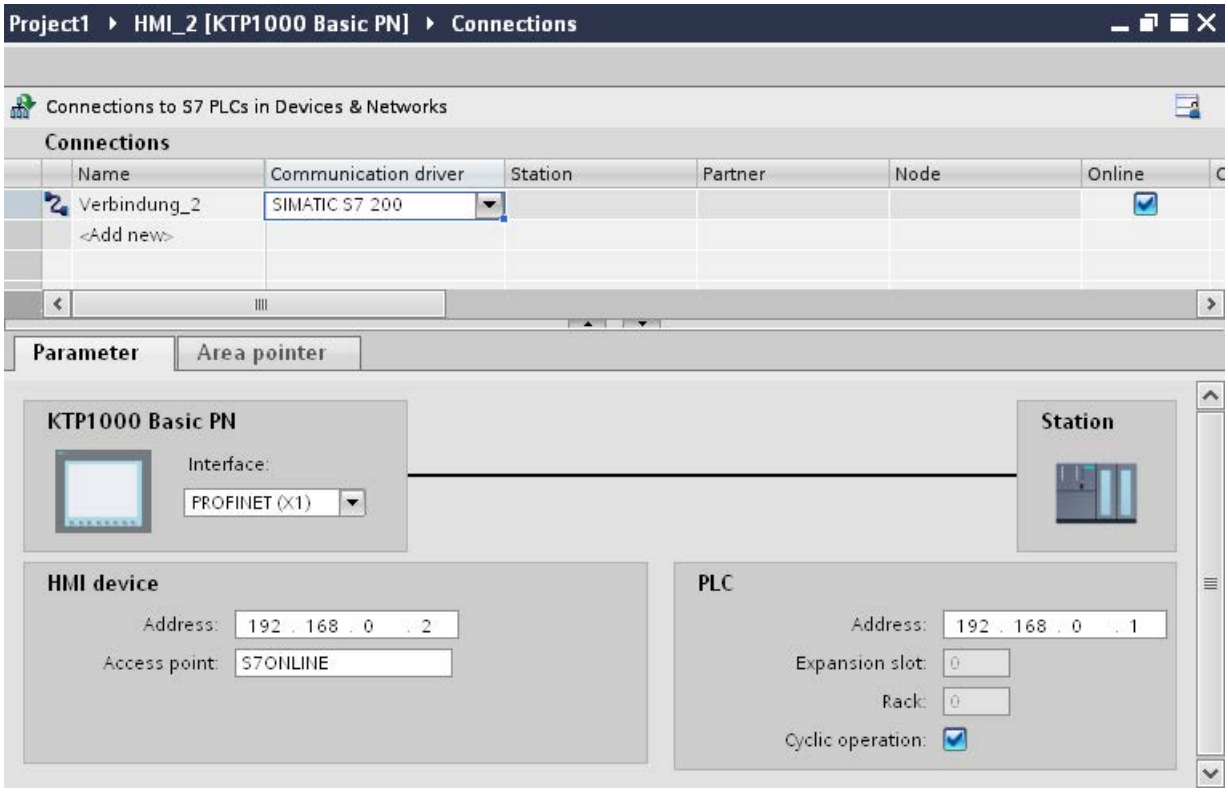
- The HMI device generates the cycle if the PLC does not support the cyclic mode.
- Screen tags are not integrated in cyclic operation.
- Cyclic mode is only set up at the restart of Runtime.
- The HMI device transfers several jobs to the PLC if cyclic mode is enabled, depending on the PLC.
- The HMI device only transfers one job to the PLC if cyclic mode is disabled.

Parameters for the connection (SIMATIC S7 200)

Parameters to be set

To set the connection parameters, such as addresses and profiles, click the connection that you have created in the "Connections" editor.

The communication partners are displayed schematically in the Inspector window under "Parameters". The "HMI device" and "PLC" areas are available for assigning parameters according to the interface used.



Ethernet parameters

Parameters for the HMI device

You set the parameters for the HMI device in the network under "HMI device".. The changes are not transferred automatically to the HMI device. You must change the settings in the Control Panel of the HMI device.

- "Interface"
  If you are directly connected to the HMI device during configuration, you can set up the IP address of the HMI device in WinCC.

  Note

  The IP address in the Control Panel will be overwritten upon subsequent loading if you have already set up the IP address in the HMI device control panel.

  The IP address already set up in the Control Panel will be retained upon subsequent loading if you activate "Set IP address using a different method".

  The IP address is transferred to the HMI device during project transfer.
  To set up the IP address of the HMI device:

  – Click the HMI device.
  – Open the "Device configuration" editor.
  – Click the Ethernet interface.
  – Assign the IP address in the inspector window under:
    "General > PROFINET interface > Ethernet addresses"

- "Address"
  You assign the IP address of the HMI device in the "Address" area.
  When you transfer the WinCC project to the HMI device, this IP address is set up directly in the HMI device.
- "Access point"
  The access point defines a logical device name through which the communication partner can be reached.

Parameters for the PLC

Under "PLC", you address the S7 module with which the HMI device will exchange data. Assign a name for the connection for each communication partner.

- "Address"
  Under "Address", set the IP address of the S7 module to which the HMI device is connected.
- "Expansion slot"
  Defines the number of the expansion slot of the CPU to be addressed.

- "Rack"
  Defines the rack number of the CPU to be addressed.
- "Cyclic operation"

  Note

  The setting "Cyclic operation" cannot be configured for the SIMATIC S7 1200 PLC.

  When cyclic operation is enabled, the PLC optimizes the data exchange between the HMI device and the PLC. This increases system performance.
  Disable cyclic mode if you are operating several HMI devices in parallel.

PROFIBUS parameters

Parameters for the HMI device

You assign the parameters for the HMI device in the network once under "HMI device". The change applies to each communication partner.

- "Type"
  Specifies the physical connection used.
- "Interface"
  For "Interface", you select the HMI device interface via which the HMI device is connected to the PROFIBUS network.
- "Baud rate"
  For "Baud rate", you set the transmission speed of the data in the network. The baud rate is determined by the slowest HMI device connected to the network. The setting must be identical throughout the network.

  Note

  If you set a baud rate of 1.5 Mbaud for OP 73 or OP 77A, the highest station address must be less than or equal to 63.

- "Address"
  You set the PROFIBUS address of the HMI device under "Address". The PROFIBUS address must be unique in the PROFIBUS network.
- "Sole master on bus"
  Disables an additional safety feature against bus faults when the HMI device is connected to the network. A passive station (slave) can only send data if it is requested to do so by an active station (master).
  In S7 200, you must set an HMI device as the master.
- "Access point"
  The access point defines a logical device name through which the communication partner can be reached.

Parameters for the network

Under "Network", you set the parameters for the PROFIBUS network to which the HMI device is linked.

- "Profile"
  For "Profile", you select the network profile that is used in the network. In "Profile", set "DP", "Universal", or "Standard". The setting must be identical throughout the network.
- "Highest address"
  For "Highest station address", set the highest station address. The highest station address must be greater than or equal to the highest actual PROFIBUS address. The setting must be identical throughout the network.

  Note

  If you set a baud rate of 1.5 Mbaud in the OP 73 or the OP 77A, the highest station address must be less than or equal 63.

- "Number of masters"
  For "Number of masters", set the number of masters in the PROFIBUS network. This information is necessary to correctly calculate the bus parameters.

Parameters for the PLC

Under "PLC", you address the S7 module with which the HMI device will exchange data. Assign a name for the connection for each communication partner.

- "Address"
  For "Address", set the PROFIBUS address of the S7 module (CPU, FM, or CP) to which the HMI device is connected.
- "Cyclic operation"

  Note

  The setting "Cyclic operation" cannot be configured for the SIMATIC S7 1200 PLC.

  When cyclic operation is enabled, the PLC optimizes the data exchange between the HMI device and the PLC. This increases system performance. Disable cyclic mode if you are operating several HMI devices in parallel. This setting is not required for SIMATIC S7-200 PLCs.

MPI parameters

Parameters for the HMI device

You assign the parameters for the HMI device in the network once under "HMI device". The change applies to each communication partner.

- "Type"
  Specifies the physical connection used.
- "Interface"
  For "Interface", you select the HMI device interface via which the HMI device is connected to the MPI network.
- "Baud rate"
  For "Baud rate", you set the transmission speed of the data in the network. The baud rate is determined by the slowest HMI device connected to the network. The setting must be identical throughout the network.

  Note

  If you set a baud rate of 1.5 Mbaud for OP 73 or OP 77A, the highest station address must be less than or equal to 63.

- "Address"
  For "Address", you set the MPI address of the HMI device. The MPI address must be unique throughout the MPI network.
- "Sole master on bus"
  Disables an additional safety feature against bus faults when the HMI device is connected to the network. A passive station (slave) can only send data if it is requested to do so by an active station (master). If you have connected only slaves to the HMI device, you must therefore disable the "Sole master on bus" safety feature.
  In S7 200, you must set an HMI device as the master.

Parameters for the network

Under "Network", you set the parameters for the MPI network to which the HMI device is linked.

- "Profile"
  For "Profile", you select the network profile that is used in the network. In "Profile", set "MPI". The setting must be identical throughout the network.
- "Highest address"
  For "Highest station address", set the highest station address. The highest station address must be greater than or equal to the highest actual MPI address. The setting must be identical throughout the network.
- "Number of masters"
  This setting is not required for MPI.

Parameters for the PLC

Under "PLC", you address the S7 module with which the HMI device will exchange data. Assign a name for the connection for each communication partner.

- "Address"
  For "Address", set the MPI address of the S7 module (CPU, FM, or CP) to which the HMI device is connected.
- "Cyclic mode"
  When cyclic mode is enabled, the PLC optimizes the data exchange between the HMI device and the PLC. This improves system performance. Disable cyclic mode if you are operating several HMI devices in parallel. This setting is not required for SIMATIC S7-200.

PPI parameters

Parameters for the HMI device

You assign the parameters for the HMI device in the network once under "HMI device". The change applies to each communication partner.

- "Type"
  Specifies the physical connection used.
- "Interface"
  For "Interface", you select the HMI device interface via which the HMI device is connected to the PP network.
- "Baud rate"
  For "Baud rate", you set the transmission speed of the data in the network. The baud rate is determined by the slowest HMI device connected to the network. The setting must be identical throughout the network.

  Note

  If you set a baud rate of 1.5 Mbaud for OP 73 or OP 77A, the highest station address must be less than or equal to 63.

- "Address"
  For "Address", you set the PPI address of the HMI device. The PPI address must be unique throughout the PPI network.
- "Access point"
  For "Access point", you set the access point via which the communication partner is reached.
- "Sole master on bus"
  Disables an additional safety feature against bus faults when the HMI device is connected to the network. A passive station (slave) can only send data if it is requested to do so by an active station (master). If you have connected only slaves to the HMI device, you must therefore disable the "Sole master on bus" safety feature.
  In S7 200, you must set an HMI device as the master.

Parameters for the network

Under "Network", you set the parameters for the network to which the HMI device is linked.

- "Profile"
  For "Profile", you select the network profile that is used in the network. In "Profile", set "PPI". The setting must be identical throughout the network.
- "Highest address"
  For "Highest station address", set the highest station address. The highest station address must be greater than or equal to the highest actual MPI address. The setting must be identical throughout the network.
- "Number of masters"
  Set the number of the master on the network to "1".

Parameters for the PLC

Under "PLC", you address the S7 module with which the HMI device will exchange data. Assign a name for the connection for each communication partner.

- "Address"
  For "Address", set the PPI address of the S7 module to which the HMI device is connected.
- "Cyclic operation"
  This parameter is not required for communication via PPI.

10.8.9.4 Data exchange

Data exchange using area pointers

General information on area pointers

Introduction

You use an area pointer to access a data area in the PLC. During communication, the PLC and the HMI device alternately access these data areas for read and write operations.

The PLC and the HMI device trigger defined interactions based on the evaluation of stored data.

Configuration of area pointers

Before you use an area pointer, you enable it under "Connections > Area pointer". You then assign the area pointer parameters.

You can find more detailed information on configuring area pointers in:

Configuring area pointers (Page 8420)

"Screen number" area pointer

Function

The HMI device saves information about the screen called on the HMI device to the "Screen number" area pointer.

This allows the transfer of the current screen contents from the HMI device to the PLC. The PLC can trigger specific reactions such as the call of a different screen.

Application

Configure and enable the area pointer in "Communication > Connections" before you put it into use. You can create only **one** instance of the "Screen number" area pointer and only on **one** PLC.

The screen number is always transferred to the PLC when a new screen is activated or when the focus within a screen changes from one screen object to another.

Structure

The area pointer is a data area in the memory of the PLC with a fixed length of 5 words.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1st word	Current screen type															
2nd word	Current screen number															
3rd word	Reserved															
4th word	Current field number															
5th word	Reserved															

- Current screen type
  "1" for root screen or
  "4" for permanent window
- Current screen number
  1 to 32767
- Current field number
  1 to 32767

"Date/time" area pointer

Function

This area pointer is used to transfer the date and time from the HMI device to the PLC.

The PLC writes control job "41" to the job mailbox.

When it evaluates the control job, the HMI device writes its current date and the time in the data area configured in the "Date/time" area pointer. All definitions are coded in BCD format.

The "Date/Time" area pointer when used in a project which contains multiple connections must be enabled for each configured connection.

Note

You cannot use the "Date/Time PLC" area pointer if you have configured the "Date/Time" area pointer.

Note

Symbolic addressing is not possible if you are using the "Date/Time" area pointer.

The "Date/Time" area pointer when used in a project which contains multiple connections must be enabled for each configured connection.

The date/time data area has the following structure:

Data word	Most significant byte								Least significant byte								
	7							0	7							0	
n+0	Reserved								Hour (0 to 23)								Time
n+1	Minute (0 to 59)								Second (0 to 59)								
n+2	Reserved								Reserved								
n+3	Reserved								Weekday (1 to 7, 1=Sunday)								Date
n+4	Day (1 to 31)								Month (1 to 12)								
n+5	Year (80 to 99/0 to 29)								Reserved								

Note

When making entries in the "Year" data area, you should note that values 80 to 99 result in years 1980 through 1999, while the values 0 to 29 result in the years 2000 through 2029.

"Date/time PLC" area pointer

Function

This area pointer is used to transfer the date and time from the PLC to the HMI device. Use this area pointer if the PLC is the time master.

The PLC loads the data area of the area pointer. All definitions are coded in BCD format.

The HMI device reads the data cyclically within the configured acquisition cycle and synchronizes itself.

Note

Set an acquisition cycle of sufficient length for the Date/time area pointer in order to avoid any negative impact on HMI device performance.
Recommended: Acquisition cycle of 1 minute, if the process allows this.

"Date/Time PLC" is a global area pointer and may be configured only once per project.

Note

You cannot use the "Date/Time" area pointer if you have configured the "Date/Time PLC" area pointer.

The date/time data area has the following structure:

DATE_AND_TIME format (in BCD code)

Data word	Most significant byte			Least significant byte		
	7	. . . . .	0	7	. . . . .	0
n+0	Year (80 to 99/0 to 29)			Month (1 to 12)		
n+1	Day (1 to 31)			Hour (0 to 23)		
n+2	Minute (0 to 59)			Second (0 to 59)		
n+3	Reserved			Reserved		Weekday (1 to 7, 1=Sunday)
n+4 [1)]	Reserved			Reserved		
n+5 [1)]	Reserved			Reserved		

1) The two data words must exist in the data area to ensure that the data format matches WinCC flexible and to avoid reading false information.

Note

When making entries in the "Year" data area, you should note that values 80 to 99 result in years 1980 through 1999, while the values 0 to 29 result in the years 2000 through 2029.

"Coordination" area pointer

Function

The "Coordination" area pointer is used to implement the following functions:

- Detecting the startup of the HMI device in the control program
- Detecting the current operating mode of the HMI device in the control program
- Detecting whether the HMI device is ready to communicate in the control program

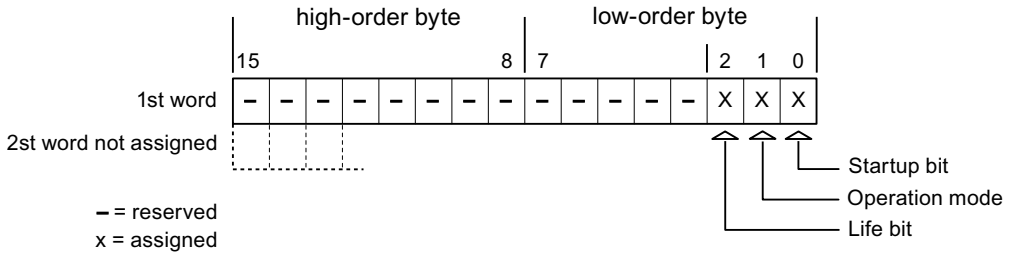
By default, the "Coordination" area pointer has the length of one word and cannot be changed.

Usage

Note

The HMI device always writes the entire coordination area when updating the area pointer. The control program can for this reason not make changes to the coordination area.

Assignment of the bits in the "Coordination" area pointer



Startup bit

The startup bit is set briefly to "0" by the HMI device during startup. It sets the bit permanently to "1" when startup is completed.

Operating mode

The operating mode bit is set to 1 as soon as the user switches the HMI device offline. The status of the operating mode bit is "0" during normal operation of the HMI device. You can determine the current operating mode of the HMI device by reading this bit in the control program.

Life bit

The HMI device inverts the life bit at intervals of approximately one second. You can check whether or not there is still a connection to the HMI device by querying this bit in the control program.

"Project ID" area pointer

Function

When Runtime starts, a check can be carried out as to whether the HMI device is connected to the correct PLC. This check is important when operating with several HMI devices.

For this, the HMI device compares a value stored on the PLC with the value specified in configuration. This ensures compatibility of the configuration data with the PLC program. If there is no concordance, a system event is given on the HMI device and Runtime is stopped.

Use

Note

HMI connections cannot be switched "online".

The HMI connection in which the "Project ID" area pointer is used must be switched "online".

To use this area pointer, set up the following during the configuration:

- Define the version of the configuration. Values between 1 and 255 are possible.
  You enter the version in the editor "Runtime settings > General" in the "Identification" area.
- Data address of the value for the version that is stored in the PLC:
  You enter the data address in the editor "Communication > Connections" under "Address".

Connection failure

A connection failure to a device on which the "Project ID" area pointer is configured results in all the other connections of the device being switched to "offline".

This behavior has the following requirements:

- You have configured several connections in a project.
- You are using the "Project ID" area pointer in at least one connection.

Causes which may set connections "offline":

- The PLC is not available.
- The connection has been switched offline in the Engineering System.

"Job mailbox" area pointer

Function

The PLC can use the job mailbox to transfer jobs to the HMI device to trigger corresponding actions on the HMI device. These functions include, for example:

- Display screen
- Set date and time

Data structure

The first word of the job mailbox contains the job number. Depending on the job mailbox, up to three parameters can be transferred.

Word	Most significant byte	Least significant byte
n+0	0	Job number
n+1	Parameter 1	
n+2	Parameter 2	
n+3	Parameter 3	

The HMI device evaluates the job mailbox if the first word of this job is not equal to zero. This means that the parameters must be entered in the job mailbox first, followed by the job number.

When the HMI device accepts the job mailbox, the first word is set to 0 again. The execution of the job mailbox is generally not completed at this point in time.

Job mailboxes

All job mailboxes and their parameters are listed below. The "No." column contains the job number of the job mailbox. Job mailboxes can only be triggered by the PLC when the HMI device is online.

No.	Function	
14	Set time (BCD-coded)	
	Parameter 1	Left byte: - Right byte: hours (0-23)
	Parameter 2	Left byte: minutes (0-59) Right byte: seconds (0-59)
	Parameter 3	-
15	Setting the date (BCD coded) [3)]	
	Parameter 1	Left byte: - Right byte: Weekday (1-7: Sunday-Saturday)
	Parameter 2	Left byte: day (1-31) Right byte: month (1-12)
	Parameter 3	Left byte: year

No.	Function	
14	Set time (BCD-coded)	
23	User logon	
	Logs the user on with the name "PLC user" at the HMI device with the group number transferred in Parameter 1. The logon is possible only when the transferred group number exists in the project.	
	Parameter 1	Group number 1 to 255
	Parameter 2, 3	-
24	User logoff	
	Logs off the current user. (The function corresponds to the "logoff" system function)	
	Parameter 1, 2, 3	-
40	Transfer date/time to PLC	
	(In the S7 format DATE_AND_TIME) An interval of at least 5 seconds must be maintained between two successive jobs to prevent overload of the HMI device.	
	Parameter 1, 2, 3	-
41	Transfer date/time to PLC	
	(In OP/MP format) An interval of at least 5 seconds must be maintained between successive jobs to prevent overload of the HMI device.	
	Parameter 1, 2, 3	-
46	Update tag	
	Causes the HMI device to read the current value of the tags from the PLC whose update ID matches the value transferred in Parameter 1. (Function corresponds to the "UpdateTag" system function.)	
	Parameter 1	1 - 100
49	Delete alarm buffer	
	Deletes all analog alarms and discrete alarms of the "Warnings" class from the alarm buffer.	
	Parameter 1, 2, 3	-
50	Delete alarm buffer	
	Deletes all analog alarms and discrete alarms of the "Errors" class from the alarm buffer.	
	Parameter 1, 2, 3	-
51	Screen selection [2]	
	Parameter 1	Screen number
	Parameter 2	-
	Parameter 3	Field number
69	Reading data record from PLC [1]	
	Parameter 1	Recipe number (1-999)
	Parameter 2	Data record number (1-65535)
	Parameter 3	0: Do not overwrite existing data record 1: Overwrite existing data record
70	Writing data record to PLC [1]	
	Parameter 1	Recipe number (1-999)

No.	Function	
14	Set time (BCD-coded)	
	Parameter 2	Data record number (1-65535)
	Parameter 3	-

1)	Only devices supporting recipes
2)	OP 73, OP 77A and TP 177A HMI devices also execute the "Screen selection" job mailbox if the on-screen keyboard is active.
3)	The weekday is ignored on HMI device KTP 600 BASIC PN.

"Data record" area pointer

"Data record" area pointer

Function

When data records are transferred between the HMI device and PLC, both partners access common communications areas on the PLC.

Data transfer types

There are two ways of transferring data records between the HMI device and PLC:

- Transfer without synchronization
- Transfer with synchronization over the data record

Data records are always transferred directly. That is, the tag values are read from an address or written to an address configured for this tag directly, without redirecting the values by means of interim memory.

Initiating the transfer of data records

There are three ways of triggering the transfer:

- Operator input in the recipe view
- Job mailboxes
  The transfer of data records can also be triggered by the PLC.
- Triggering by configured functions

If the transfer of data records is triggered by a configured function or by a job mailbox, the recipe view on the HMI device remains operable. The data records are transferred in the background.

Simultaneous processing of several transfer requests is, however, not possible. In this case, the HMI device rejects the other transfer requests with a system event.

Sequence of a transfer started by the operator in the recipe view

Reading from the PLC started by the operator in the recipe view

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe number to be read and the status "Transferring" in the data record and sets the data record number to 0.	Abort with system event.
3	The HMI device reads the values from the PLC and displays them in the recipe view. If the recipes have synchronized tags, the values from the PLC are also written to the tags.	
4	The HMI device sets the status "Transfer completed."	
5	The control program must reset the status word to zero in order to enable further transfers.	

Writing to the PLC started by the operator in the recipe view

Step	Action	
	Check: Status word = 0?	
1	Yes	No
	The HMI device enters the recipe and data record number to be written and the status "Transferring" in the data mailbox.	Abort with system event.
2	The HMI device writes the current values to the PLC. If the recipes have synchronized tags, the changed values are synchronized in the recipe view and tags and then written to the PLC.	
3	The HMI device sets the status "Transfer completed."	
4	If required, the control program can now evaluate the transferred data.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Note

The status word may only be set by the HMI device. The PLC may only reset the status word to zero.

Note

The PLC may only evaluate the recipe and data record numbers when data inconsistency is detected if one of the conditions outlined below has been met:

- The data mailbox status is set to "Transfer completed".
- The data mailbox status is set to "Transfer completed with error".

Sequence of the transfer triggered by a job mailbox

The transfer of data records between the HMI device and the PLC can be initiated by the HMI device or by the PLC.

The two job mailboxes No. 69 and No. 70 are available for this type of transfer.

No. 69: Read data record from PLC ("PLC → DAT")

Job mailbox no. 69 transfers data records from the PLC to the HMI device. The job mailbox is structured as follows:

	Left byte (LB)	Right byte (RB)
Word 1	0	69
Word 2	Recipe number (1-999)	
Word 3	Data record number (1-65,535)	
Word 4	Do not overwrite existing data record: 0 Overwrite existing data record: 1	

No. 70: Write data record to PLC ("DAT → PLC")

Job mailbox No. 70 transfers data records from the HMI device to the PLC. The job mailbox is structured as follows:

	Left byte (LB)	Right byte (RB)
Word 1	0	70
Word 2	Recipe number (1-999)	
Word 3	Data record number (1-65,535)	
Word 4	—	

Sequence when reading from the PLC with job mailbox "PLC → DAT" (no. 69)

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the job and the status "Transferring" in the data mailbox.	Abort without return message.
3	The HMI device reads the values from the PLC and saves these to the data record defined in the job mailbox.	
4	• If "Overwrite" was selected in the job, an existing data record is overwritten without any prompt for confirmation. The HMI device sets the status "Transfer completed." • If "Do not overwrite" was selected in the job, and the data record already exists, the HMI device aborts the job and enters 0000 1100 in the status word of the data mailbox.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Sequence of writing to the PLC using job mailbox "DAT → PLC" (no. 70)

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the job and the status "Transferring" in the data mailbox.	Abort without return message.
3	The HMI device fetches the values of the data record specified in the job from the data medium and writes the values to the PLC.	
4	The HMI device sets the status "Transfer completed."	
5	The control program can now evaluate the transferred data. The control program must reset the status word to zero in order to enable further transfers.	

Sequence of the transfer when triggered by a configured function

Reading from the PLC using a configured function

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the function and the status "Transferring" in the data mailbox.	Abort with system event.
3	The HMI device reads the values from the PLC and stores them in the data record specified in the function.	
4	• If "Yes" was selected for the "Overwrite" function, an existing data record is overwritten without any prompt for confirmation. The HMI device sets the status "Transfer completed." • If "No" was selected for the "Overwrite" function and the data record already exists, the HMI device aborts the job and enters 0000 1100 in the status word of the data mailbox.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Writing to the PLC by means of configured function

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the function and the status "Transferring" in the data mailbox.	Abort with system event.

Step	Action	
3	The HMI device fetches the values of the data record specified in the function from the data medium and transfers the values to the PLC.	
4	The HMI device sets the status "Transfer completed."	
5	The control program can now evaluate the transferred data. The control program must reset the status word to zero in order to enable further transfers.	

Possible causes of error when transferring data records

Possible causes of faults

The section below shows possible causes of errors which lead to a data record transfer being terminated with errors:

- Tag address not set up on the PLC
- Overwriting data records not possible
- Recipe number does not exist
- Data record number does not exist

Note

The status word may only be set by the HMI device. The PLC may only reset the status word to zero.

Note

The PLC may only evaluate the recipe and data record numbers when data inconsistency is detected if one of the conditions outlined below has been met:

- The data mailbox status is set to "Transfer completed".
- The data mailbox status is set to "Transfer completed with error".

Reaction to an aborted transfer due to errors

If the transfer of data records is aborted due to errors, the HMI device reacts as follows:

- Triggering by the operator in the recipe view
  Information in the status bar of the recipe view and output of system events
- Triggered by function
  Output of system events
- Triggering by job mailbox
  No return message on the HMI device

You can nonetheless evaluate the status of the transfer by querying the status word in the data mailbox.

Transfer without synchronization

If you select asynchronous transfer of data records between the HMI device and PLC, there is no coordination over the common data areas. It is therefore unnecessary to set up a data area during configuration.

Asynchronous data record transfer can be a useful alternative, for example, when:

- The system is capable of excluding the risk of uncontrolled overwriting of data by the communication peer.
- The PLC does not require information about the recipe number and data record number.
- The transfer of data records is triggered by the operator of the HMI device.

Reading values

When a read job is triggered, the values are read from the PLC addresses and transferred to the HMI device.

- Triggering by the operator in the recipe view:
  The values are downloaded to the HMI device. You can then process, edit, or save these values, for example.
- Triggering by a function or job mailbox:
  The values are saved immediately to the data volume.

Writing values

When a write job is triggered, the values are written to the PLC addresses.

- Triggering by the operator in the recipe view:
  The current values are written to the PLC.
- Triggering by a function or job mailbox:
  The current values are written to the PLC from the data medium.

Transfer with synchronization

If you select synchronous transfer, both communication partners set status bits in the common data area. You can use this mechanism to prevent uncontrolled overwriting of data in either direction in your control program.

Application

Synchronous data record transfer can be a useful solution, for example, when:

- The PLC is the "active partner" in the transfer of data records.
- The PLC evaluates the information about the recipe number and data record number.
- The transfer of data records is triggered by means of a Job mailbox.

Requirements

In order to synchronize transfer of data records between the HMI device and the PLC, the following requirements must be met during configuration:

- An area pointer has been set up: "Communication > Connections" editor in "Area pointer".
- The PLC with which the HMI device synchronizes transfer of data records is specified in the recipe:
  "Recipes" editor in the Inspector window the "Coordinated transfer of data records" option under "General > Synchronization > Settings".

Structure of the data area

The data area has a fixed length of 5 words. Structure of the data area:

	15		0
1. Word	Current recipe number (1 - 999)		
2. Word	Current data record number (0 - 65535)		
3. Word	Reserved		
4. Word	Status (0, 2, 4, 12)		
5. Word	Reserved		

- Status
  The status word (word 4) can adopt the following values:

Value		Meaning
Decimal	Binary	
0	0000 0000	Transfer permitted, data record free
2	0000 0010	Transferring
4	0000 0100	Transfer completed without error
12	0000 1100	Transfer completed with error

Trends

Trends

A trend is the graphical representation of one or more values from the PLC. The value is read out time-triggered for Basic Panels.

For additional information see:

Configuring trend displays for values from the PLC (Page 8147)

Time-triggered trends

The HMI device reads in the trend values cyclically at an interval specified in the configuration.

Time-triggered trends are suitable for continuous curves, such as the operating temperature of a motor.

Alarms

Configuring alarms

Configure alarms

Several steps are needed to configure alarms, such as operational messages, error alarms, and acknowledgement.

- Step 1: Create tags
- Step 2: Configure alarms
- Step 3: Configure acknowledgment

You can find additional information in the section:

Working with alarms (Page 8157)

Distinctive features when configuring alarms

If you are configuring connections of HMI devices to PLCs of other manufacturers, note the following distinctive features when configuring:

- Data types of the tags
- Addressing of tags
- How the bit positions are counted

Data types

For connections with a SIMATIC communication driver, the following data types are supported:

PLC	Permitted data types	
	Discrete alarms	Analog alarms
SIMATIC S7 PLCs	WORD, INT	BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, TIMER

How the bit positions are counted

For connections with a SIMATIC communication driver, the following counting method applies:

How the bit positions are counted	Byte 0								Byte 1							
	Most significant byte								Least significant byte							
In SIMATIC S7 PLCs	7							0	7							0
In WinCC you configure:	15							8	7							0

Acknowledgment of alarms

Procedure

Create suitable tags on the PLC to acknowledge an error alarm. You assign these tags to an alarm in the "Bit messages" editor. You make the assignment in "Properties > Acknowledgment".

Distinction in terms of acknowledgment:

- Acknowledgment by the PLC
- Acknowledgment on the HMI device

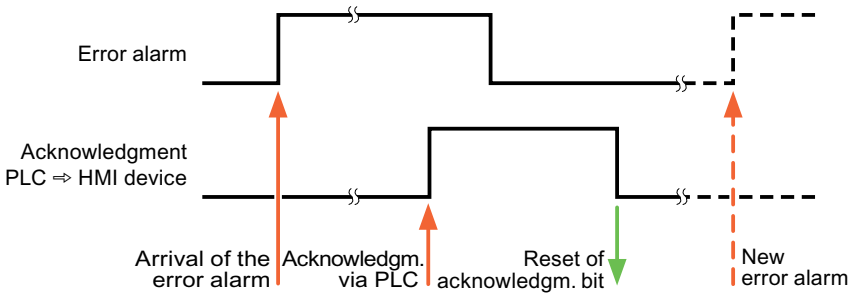
Acknowledgment by the PLC

In "PLC acknowledgment tag", configure the tag or array tag and the bit number that the HMI device uses to identify a PLC acknowledgment.

A bit set in the tag triggers acknowledgment of the assigned error alarm bit at the HMI device. This tag bit returns a function similar to acknowledgment on the HMI device which is triggered by pressing the "ACK" button, for example.

The acknowledgment bit must be located in the same tag as the bit for the error alarm.

Reset the acknowledgment bit before setting the bit in the alarm area again. The figure below shows the pulse diagram.



Acknowledgment on the HMI device

In the "HMI acknowledgment tag" area, configure the tag or array tag as well as the bit number that the HMI device writes to the PLC after acknowledgment. Make sure when you use an array tag that it is not longer than 6 words.

To always create a signal change when setting an assigned acknowledgment bit of a discrete alarm that must be acknowledged, the HMI device will reset the acknowledgment bit assigned to the alarm as soon as it detects an alarm subject to acknowledgment and write the acknowledgment tag in the PLC. There will be a certain delay between detecting the message and writing the acknowledgment tag in the PLC because the HMI device has to process the operations.

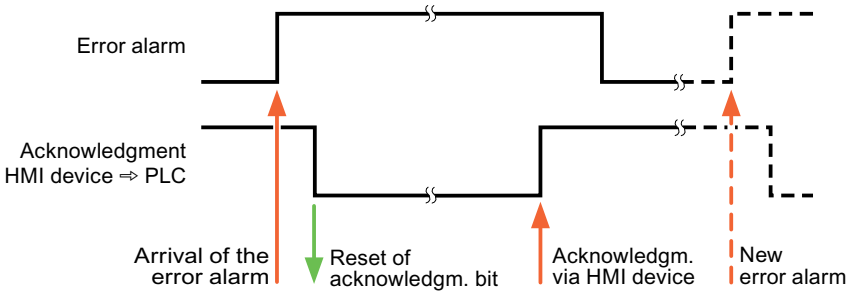
If a discrete alarm subject to acknowledgment is acknowledged by the HMI device, then the corresponding bit in the assigned acknowledgment tag will be set. The entire acknowledgment tag is then written to the PLC by the HMI device. This allows the PLC to recognize that a certain alarm message has been acknowledged at the HMI device.

Note

All alarm bits acknowledged since the last Runtime start will remain in the acknowledgment tag until a new incoming of the respective discrete alarms is detected.

This area should only be read by the PLC because the entire section of the HMI device will be overwritten once the next acknowledgment tag is written.

The figure below shows the pulse diagram.



10.8.9.5 Performance features of communication

Permitted data types for SIMATIC S7 200

Permitted data types for connections with SIMATIC S7 200

The table lists the data types that can be used when configuring tags and area pointers.

Data type	Length
Bool	1 bit
Byte	1 byte
Char	1 byte
Word	2 bytes
Int	2 bytes
DWord	4 bytes
DInt	4 bytes
Real	4 bytes
StringChar	--
Timer	2 bytes
Array	--

Note

Disconnection with a PPI network

If you are using arrays in the configuration, an array size of approximately 1000 bytes may cause an interruption of the connection.

Use smaller arrays in your configuration.

10.8.10 Communicating with SIMATIC LOGO!

10.8.10.1 Communication with SIMATIC LOGO!

Introduction

This section describes the communication between an HMI device and the SIMATIC SIMATIC LOGO! controller.

You can configure the following communication channels for the SIMATIC LOGO! controller:

- PROFINET
- Ethernet

HMI connection for communication

You configure connections between the HMI device and SIMATIC LOGO! in the "Connections" editor of the HMI device.

Data exchange

Data exchange with the SIMATIC LOGO! control system is possible by means of tags.

Data cannot be exchanged using area pointers.

10.8.10.2 Creating a connection to SIMATIC LOGO!

Introduction

You configure a connection to the SIMATIC LOGO! controller in the "Connections" editor of the HMI device. The interfaces are named differently depending on the HMI device.

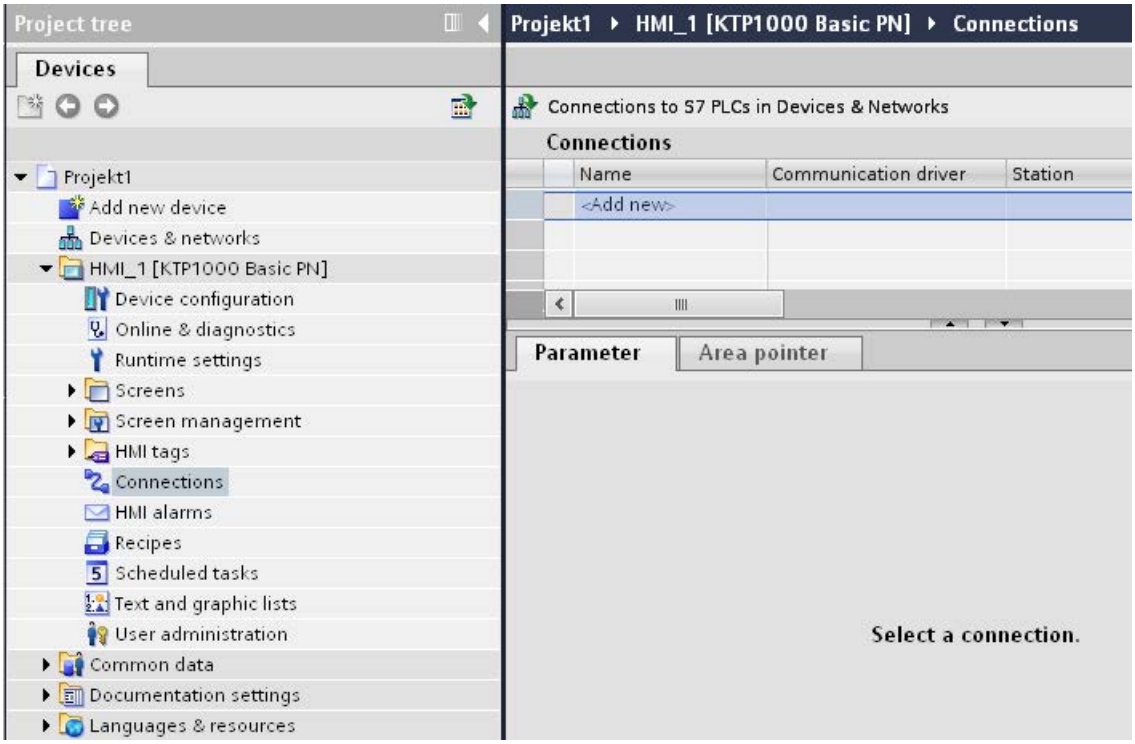
Requirements

- A project is open.
- An HMI device has been created.

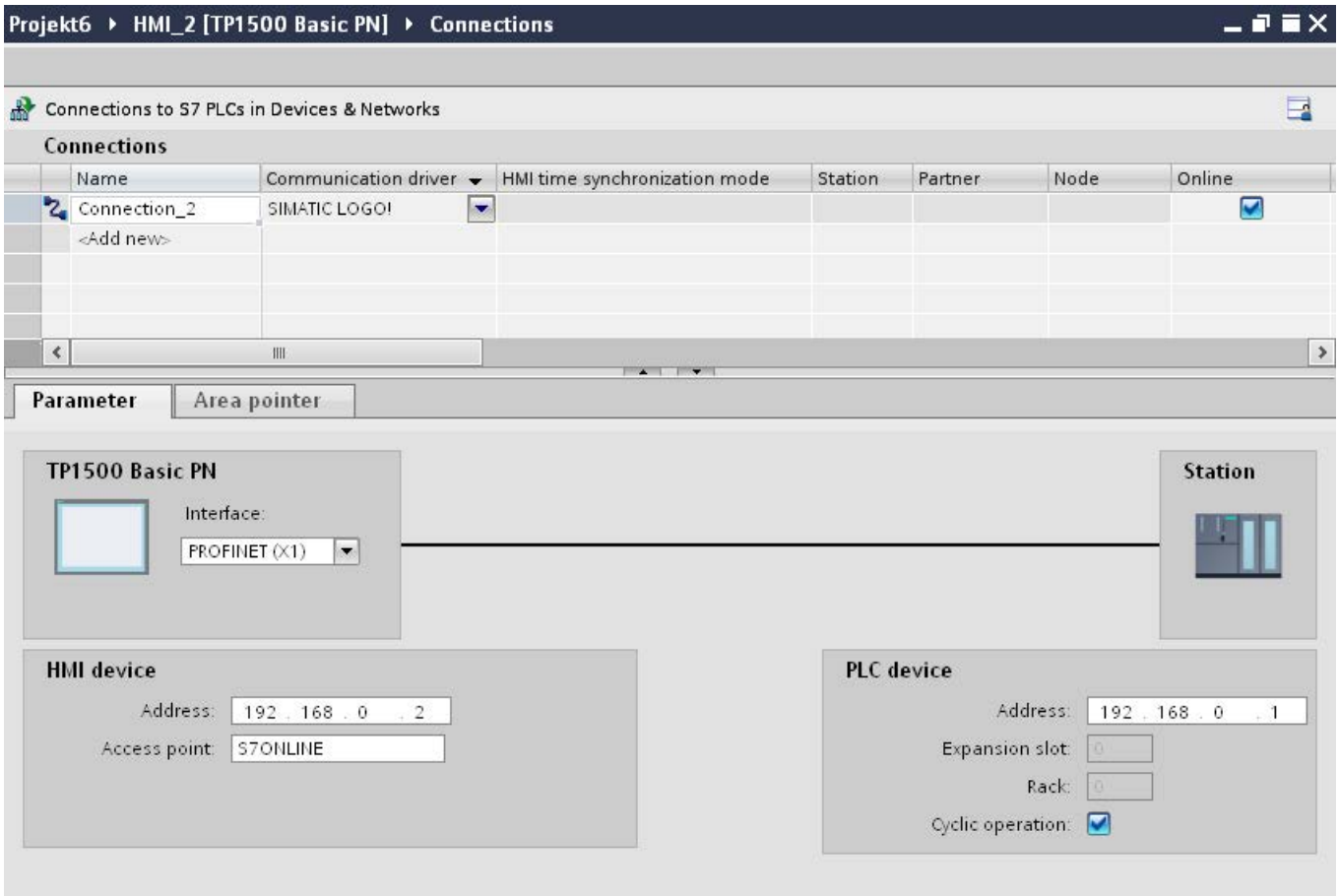
Procedure

1. Double-click the HMI device under "Devices" in the project tree.
2. Double-click the "Connections" item.

3. Double-click "<Add>" in the "Connections" editor.



4. In the "Communication drivers" column, select the "SIMATIC LOGO!" driver.
5. Select all necessary connection parameters for the interface in the Inspector window under "Parameters".



See the chapter "Auto-Hotspot" for additional details.

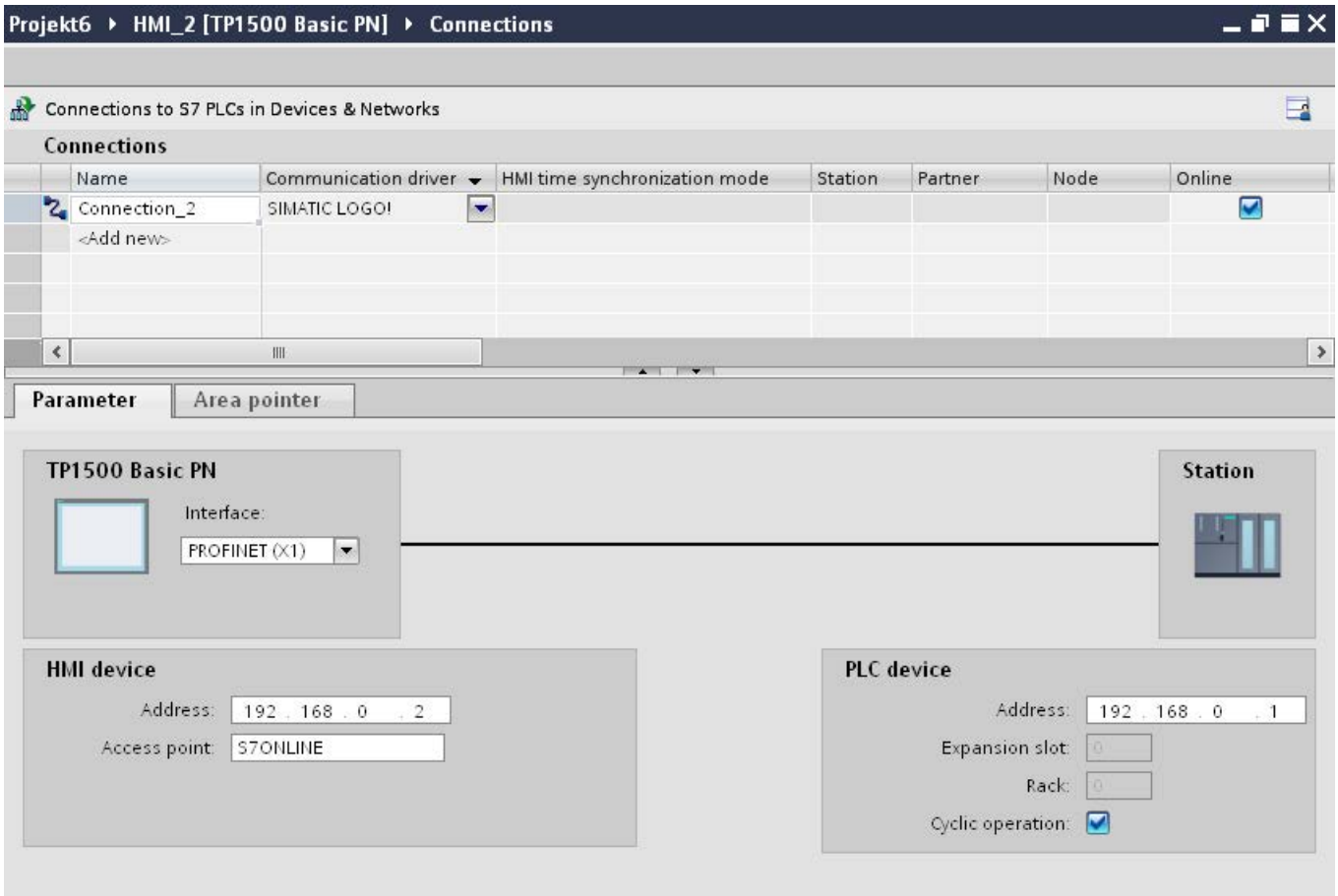
10.8.10.3 Connection parameters

Connection parameters

Parameters to be set

To set the connection parameters, such as addresses and profiles, click the connection that you have created in the "Connections" editor.

The communication partners are displayed schematically in the Inspector window under "Parameters". The "HMI device" and "PLC" areas are available for assigning parameters according to the interface used.


Projekt6 ▸ HMI_2 [TP1500 Basic PN] ▸ Connections
Connections to S7 PLCs in Devices & Networks
Connections
Name
Communication driver
HMI time synchronization mode
Station
Partner
Node
Online
Connection_2
SIMATIC LOGO!
<Add new>
Parameter
Area pointer
TP1500 Basic PN
Interface:
PROFINET (X1)
Station
HMI device
Address:
192 . 168 . 0 . 2
Access point:
S7ONLINE
PLC device
Address:
192 . 168 . 0 . 1
Expansion slot:
Rack:
Cyclic operation:

Ethernet parameters

Parameters for the HMI device

You set the parameters for the HMI device in the network under "HMI device".. The changes are not transferred automatically to the HMI device. You must change the settings in the Control Panel of the HMI device.

- "Interface"
  If you are directly connected to the HMI device during configuration, you can set up the IP address of the HMI device in WinCC.

  Note

  The IP address in the Control Panel will be overwritten upon subsequent loading if you have already set up the IP address in the HMI device control panel.

  The IP address already set up in the Control Panel will be retained upon subsequent loading if you activate "Set IP address using a different method".

  The IP address is transferred to the HMI device during project transfer.
  To set up the IP address of the HMI device:

  – Click the HMI device.
  – Open the "Device configuration" editor.
  – Click the Ethernet interface.
  – Assign the IP address in the inspector window under:
    "General > PROFINET interface > Ethernet addresses"

- "Address"
  You assign the IP address of the HMI device in the "Address" area.
  When you transfer the WinCC project to the HMI device, this IP address is set up directly in the HMI device.
- "Access point"
  The access point defines a logical device name through which the communication partner can be reached.

Parameters for the PLC

Under "PLC", you address the S7 module with which the HMI device will exchange data. Assign a name for the connection for each communication partner.

- "Address"
  Under "Address", set the IP address of the S7 module to which the HMI device is connected.
- "Expansion slot"
  Defines the number of the expansion slot of the CPU to be addressed.

- "Rack"
  Defines the rack number of the CPU to be addressed.
- "Cyclic operation"

  Note

  The setting "Cyclic operation" cannot be configured for the SIMATIC S7 1200 PLC.

  When cyclic operation is enabled, the PLC optimizes the data exchange between the HMI device and the PLC. This increases system performance.
  Disable cyclic mode if you are operating several HMI devices in parallel.

Cyclic operation

Handling the "Cyclic operation" selection

If "Cyclic operation" is enabled, the HMI device sends a message frame to the CPU at the beginning of communication indicating that certain tags are required on a recurring basis.

The CPU then always transmits the data at the same cyclic interval. This saves the HMI device from having to output new requests for the data.

If cyclic operation is disabled, the HMI device sends a request whenever information is required.

Additional properties:

- Cyclic operation reduces data transmission load at the HMI device. The PLC resources are used to relieve load on the HMI device.
- The PLC only supports a certain number of cyclic services. The HMI device handles the operation if the PLC cannot provide any further resources for cyclic services.
- The HMI device generates the cycle if the PLC does not support the cyclic mode.
- Screen tags are not integrated in cyclic operation.
- Cyclic mode is only set up at the restart of Runtime.
- The HMI device transfers several jobs to the PLC if cyclic mode is enabled, depending on the PLC.
- The HMI device only transfers one job to the PLC if cyclic mode is disabled.

10.8.10.4 Data exchange

Trends

General information on trends

Trends

A trend is the graphical representation of one or more values from the PLC. The value is read out either time- or bit-triggered, depending on the configuration.

For additional information see:

Auto-Hotspot

Note

The value is read out time-triggered for Basic Panels.

Time-triggered trends

The HMI device reads in the trend values cyclically at an interval specified in the configuration. Time-triggered trends are suitable for continuous curves, such as the operating temperature of a motor.

Bit-triggered trends

Through a trigger bit set in the trend request tag, the HMI device either reads in a trend value or an entire trend buffer. This setting is defined in the configuration. Bit-triggered trends are normally used to represent fast changing values. One example might be the injection pressure in the production of plastic parts.

To trigger bit-triggered trends, appropriate external tags must be created in the "HMI tags" editor and connected to trend areas during configuration. The HMI device and PLC then communicate with each other via these trend areas.

The following areas are available for trends:

- Trend request area
- Trend transfer area 1
- Trend transfer area 2 (required only with switch buffers)

Trend request and trend transfer

Trend request area

The HMI device sets corresponding bits in the trend request area when you open a screen which contains one or more trends on the HMI device. After closing the screen, the HMI device resets the relevant bits in the trend request area.

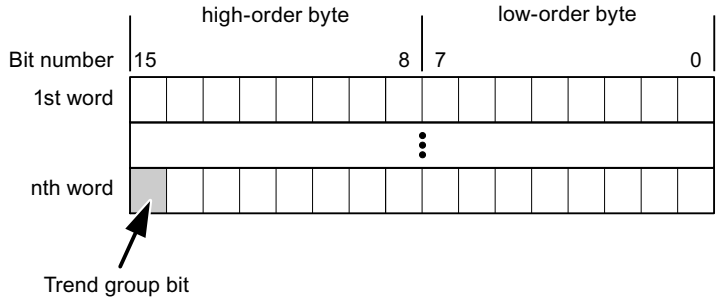
The trend request area can be used for evaluation purposes in the PLC to determine which trend is currently being displayed on the HMI device. Trends can also be triggered without evaluation of the trend request area.

Trend transfer area 1

This area is used to trigger trends. You must set the bit assigned to the trend in the trend transfer area and set the trend group bit in your control program. The trend group bit is the last bit in the trend transfer area.

The HMI device detects the trigger. The HMI device reads either a value or the entire buffer from the PLC. It then resets the trend bit and the trend group bit.

The following picture shows the structure of a trend transfer area.



The trend transfer area must not be modified by the PLC program until the trend group bit has been reset.

Trend transfer area 2

Trend transfer area 2 is required for trends configured with a switch buffer. The trend transfer areas 1 and 2 have a similar structure.

Switch buffer

The switch buffer is a second buffer for the same trend that can be set up during configuration.

The PLC writes to Buffer 2 while the HMI device reads values from Buffer 1, and writes to Buffer 1 when the HMI device is reading Buffer 2. This prevents the PLC from overwriting trend values while the trend is being read by the HMI device.

Permitted data types for trends

For SIMATIC S7

You assign one bit to each trend during configuration. Tags and array tags of the "Word" or "Int" data type are permitted.

Alarms

Configuring alarms

Configure alarms

Several steps are needed to configure alarms, such as operational messages, error alarms, and acknowledgement.

- Step 1: Create tags
- Step 2: Configure alarms
- Step 3: Configure acknowledgment

You can find additional information in the section:

Auto-Hotspot

Distinctive features when configuring alarms

If you are configuring connections of HMI devices to PLCs of other manufacturers, note the following distinctive features when configuring:

- Data types of the tags
- Addressing of tags
- How the bit positions are counted

Data types

For connections with a SIMATIC communication driver, the following data types are supported:

PLC	Permitted data types	
	Discrete alarms	Analog alarms
SIMATIC S7 PLCs	WORD, INT	BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, TIMER

How the bit positions are counted

For connections with a SIMATIC communication driver, the following counting method applies:

How the bit positions are counted	Byte 0								Byte 1							
	Most significant byte								Least significant byte							
In SIMATIC S7 PLCs	7							0	7							0
In WinCC you configure:	15							8	7							0

Acknowledgment of alarms

Procedure

Create suitable tags on the PLC to acknowledge an error alarm. You assign these tags to an alarm in the "Bit messages" editor. You make the assignment in "Properties > Acknowledgment".

Distinction in terms of acknowledgment:

- Acknowledgment by the PLC
- Acknowledgment on the HMI device

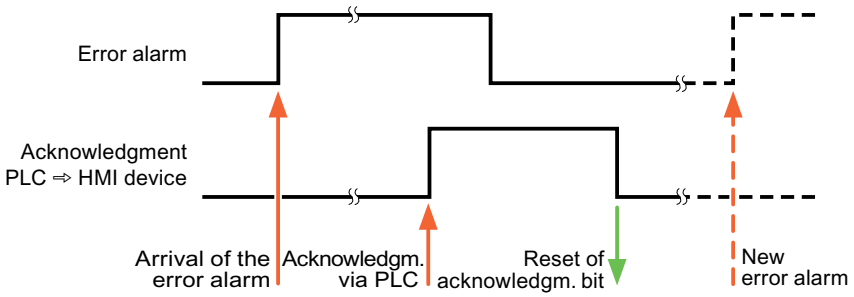
Acknowledgment by the PLC

In "PLC acknowledgment tag", configure the tag or array tag and the bit number that the HMI device uses to identify a PLC acknowledgment.

A bit set in the tag triggers acknowledgment of the assigned error alarm bit at the HMI device. This tag bit returns a function similar to acknowledgment on the HMI device which is triggered by pressing the "ACK" button, for example.

The acknowledgment bit must be located in the same tag as the bit for the error alarm.

Reset the acknowledgment bit before setting the bit in the alarm area again. The figure below shows the pulse diagram.



Acknowledgment on the HMI device

In the "HMI acknowledgment tag" area, configure the tag or array tag as well as the bit number that the HMI device writes to the PLC after acknowledgment. Make sure when you use an array tag that it is not longer than 6 words.

To always create a signal change when setting an assigned acknowledgment bit of a discrete alarm that must be acknowledged, the HMI device will reset the acknowledgment bit assigned to the alarm as soon as it detects an alarm subject to acknowledgment and write the acknowledgment tag in the PLC. There will be a certain delay between detecting the message and writing the acknowledgment tag in the PLC because the HMI device has to process the operations.

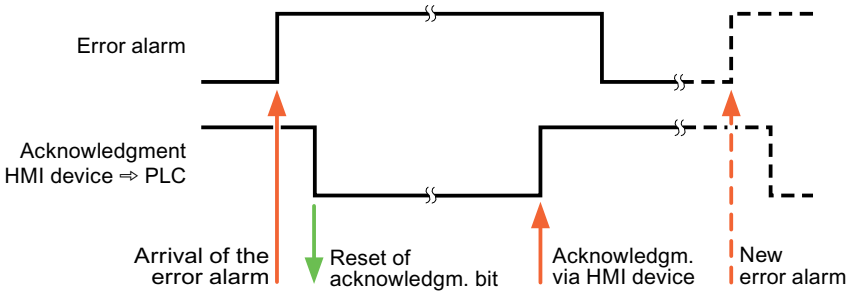
If a discrete alarm subject to acknowledgment is acknowledged by the HMI device, then the corresponding bit in the assigned acknowledgment tag will be set. The entire acknowledgment tag is then written to the PLC by the HMI device. This allows the PLC to recognize that a certain alarm message has been acknowledged at the HMI device.

Note

All alarm bits acknowledged since the last Runtime start will remain in the acknowledgment tag until a new incoming of the respective discrete alarms is detected.

This area should only be read by the PLC because the entire section of the HMI device will be overwritten once the next acknowledgment tag is written.

The figure below shows the pulse diagram.



10.8.10.5 Performance features of communication

Valid data types for SIMATIC LOGO!

Valid data types for connections with SIMATIC LOGO!

Data type	Length
Bool	1 bit
Byte	1 byte

Data type	Length
Int	2 bytes
DInt	4 bytes
Word	2 bytes
DWord	4 bytes
Array	--

10.8.11 Communication with other PLCs

10.8.11.1 Communication with other PLCs

Introduction

Communication with other PLCs is communication with PLCs that are not in the SIMATIC family.

These PLCs have proprietary protocols for data exchange. The protocols are configured as communication drivers in WinCC.

Communication drivers

The following communication drivers are supported in WinCC and are already installed:

- Allen-Bradley
  - Allen-Bradley EtherNet/IP
  - Allen-Bradley DF1
- Mitsubishi
  - Mitsubishi MC TCP/IP
  - Mitsubishi FX
- Modicon Modbus
  - Modicon Modbus TCP/IP
  - Modicon Modbus RTU
- Omron
  - Omron Host Link

Communication drivers in WinCC RT Professional

The following communication drivers are supported for RT Professional:

- Allen-Bradley
  - Allen-Bradley EtherNet/IP
- Mitsubishi
  - Mitsubishi MC TCP/IP
- Modicon Modbus
  - Modicon Modbus TCP

Connections between HMI devices and other PLCs

You configure the connections between HMI devices and other PLCs in the "Connections" editor of the HMI device. These connections are non-integrated connections.

10.8.11.2 Distinctive features when configuring

Distinctive features for data exchange

Distinctive features apply when configuring connections to other PLCs, compared to configuring integrated connections.

Note the following distinctive features when configuring:

- Addressing of tags
- Permitted data types
- Distinctive features when configuring area pointers
- Distinctive features when configuring alarms
- Distinctive features when configuring trends

For more detailed information on distinctive features when configuring, refer to Section "Data exchange" of the respective communication driver.

10.8.11.3 Communication drivers

Allen-Bradley

Allen-Bradley communication drivers

Introduction

This section describes the communication between an HMI device and PLCs that use Allen-Bradley communication drivers.

The following communication drivers are supported:

- Allen-Bradley EtherNet/IP
- Allen-Bradley DF1

Data exchange

Data is exchanged by means of tags or area pointers.

- Tags
  The PLC and the HMI device use process values for data exchange. You create tags in the configuration that point to addresses in the PLC. The HMI device reads the value from the defined address, and then displays it. The operator may also enter values on the HMI device, which are then written to the address in the PLC.
- Area pointers
  Area pointers are used to exchange specific data and are only set up when these data are used.

Allen-Bradley EtherNet/IP

Configuring a connection via Allen-Bradley EtherNet/IP

Introduction

You configure a connection to a PLC with an Allen-Bradley EtherNet/IP communication driver in the "Connections" editor of the HMI device.

The Ethernet interfaces are named differently depending on the HMI device.

Example: PROFINET interface corresponds to the Ethernet interface

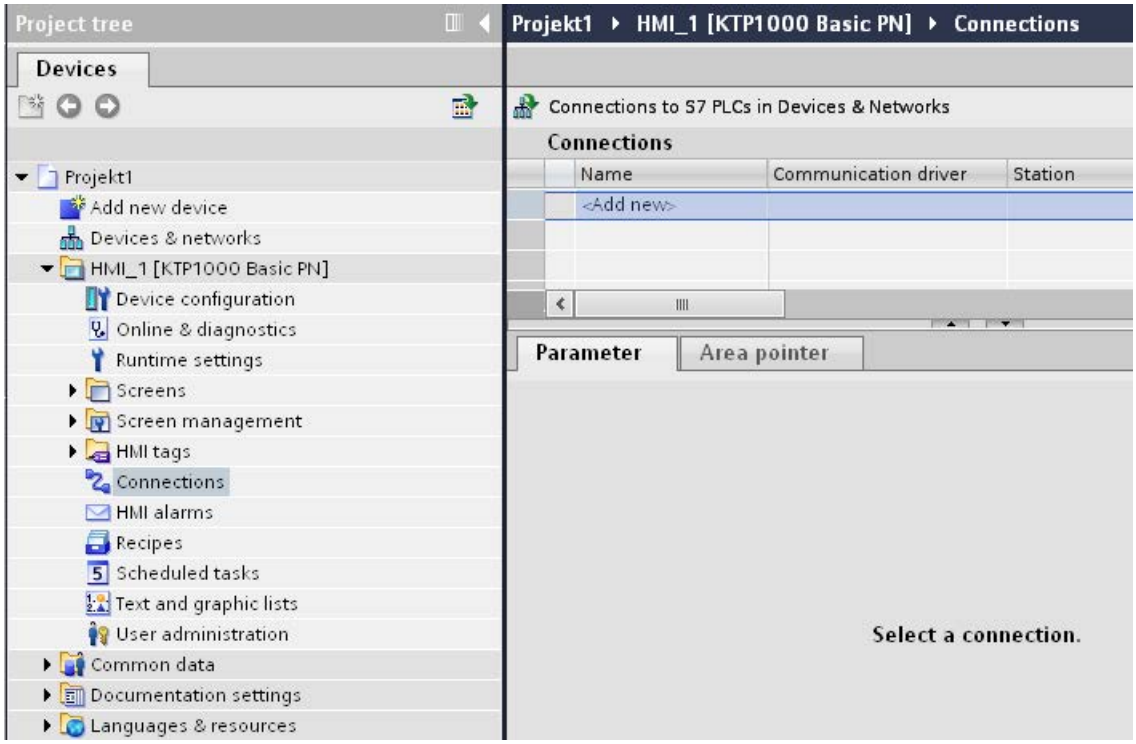
Requirements

- A project is open.
- An HMI device has been created.

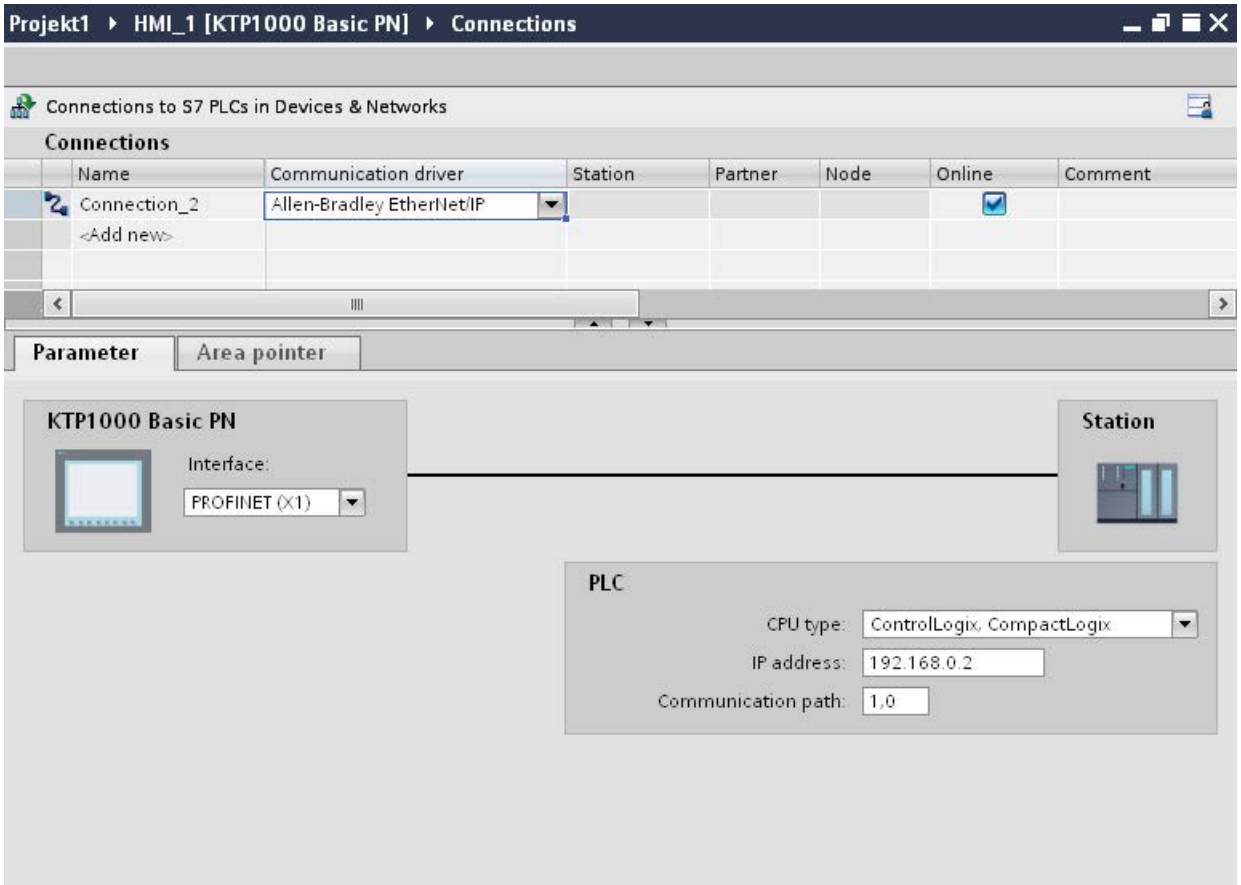
Procedure

1. Double-click the HMI device under "Devices" in the project tree.
2. Double-click the "Connections" item.

3. Double-click "<Add>" in the "Connections" editor.



4. In the "Communication drivers" column, select the "Allen-Bradley EtherNet/IP" driver.



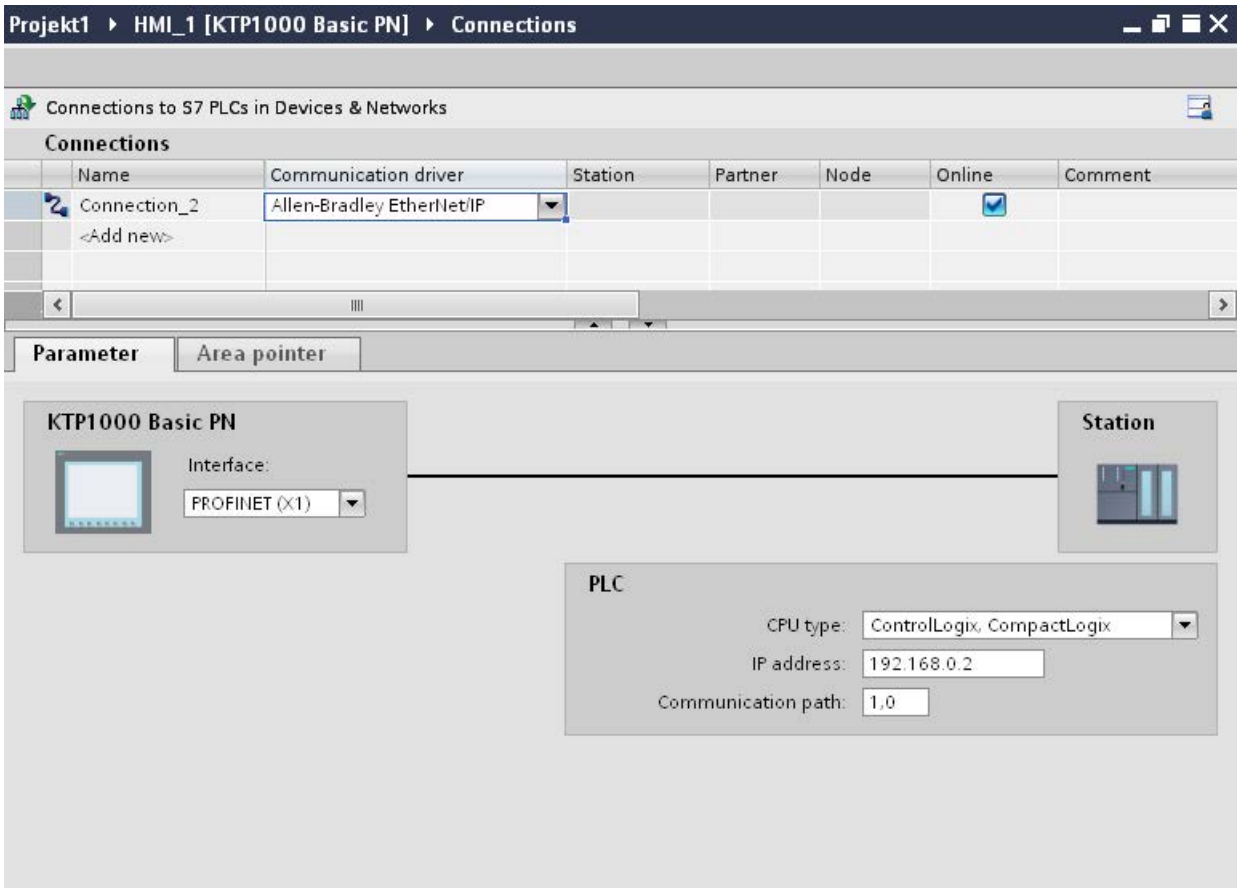
5. Select all necessary connection parameters for the interface in the Inspector window under "Parameters".

Parameters for the connection (Allen-Bradley EtherNet/IP)

Parameters to be set

To assign the connection parameters, such as addresses and profiles, click the connection that you have created in the "Connections" editor.

The communication partners are displayed schematically in the Inspector window under "Parameters". The "PLC" area is available for assigning parameters according to the interface used.



Parameters for the HMI device

You can select only one interface for the HMI device in the Inspector window under "Parameters". Depending on the HMI device, there are several interfaces available.

If you are directly connected to the HMI device during configuration, you can set up the IP address of the HMI device in WinCC. The IP address is transferred to the HMI device upon subsequent loading.

Note

The IP address in the control panel will be overwritten upon subsequent loading if you have already set up the IP address in the HMI device control panel.

The IP address already set up in the control panel will be retained upon subsequent loading if you activate "Set IP address using a different method".

To set up the IP address of the HMI device:

1. Click on the HMI device.
2. Open the "Device configuration" editor.

3. Click the Ethernet interface.
4. Assign the IP address in the inspector window under:
   "General > PROFINET interface > Ethernet addresses"

Parameters for the PLC

- CPU type
  For "CPU type", set the CPU type of the PLC used.
- IP address
  Set the IP address or host name of the Ethernet/IP module of the PLC. Only the IP address can be used on a Basic Panel.
- Communication path
  Set the CIP path from the Ethernet module to the PLC. This establishes a logical connection between the Ethernet module and PLC, even if both devices are located in different CIP networks.
  For additional information see: Examples: Communication path

Connecting HMI device to PLC

Connections via Allen-Bradley EtherNet/IP

Connection

The HMI device can be connected to the Allen-Bradley PLC using the following components:

- Existing Ethernet network that also contains the PLCs
- Cross-over Ethernet cable connected directly to the Ethernet interface of the CPU or the communication module

The connection of the HMI device to an Allen-Bradley PLC is limited primarily to the physical connection of the HMI device. Special blocks for the connection are not required in the PLC.

Communication types

Approved communication types with Allen-Bradley EtherNet/IP

The following communication types are system-tested and approved:

- Point-to-point connection to the approved PLCs
- Multipoint connection from a HMI device (Allen-Bradley Ethernet/IP-Client) with up to 4 PLCs with the respectively approved PLCs. CPU types can be mixed.

Connection

Connection with the following PLCs is approved with Allen-Bradley EtherNet/IP:

- CPU type: "ControlLogix, Compact Logix"
  - ControlLogix
    556x(1756-L6x) with Ethernet module 1756-ENBT
  - Guard Logix-System ControlLogix
    556xS(1756-L6xS) with Ethernet module 1756-ENBT
  - CompactLogix
  - 533xE(1769-L3xE) with Ethernet interface onboard
  - 532xE(1769-L2xE) with Ethernet interface onboard
  - 534x (1768-L4x) with Ethernet module 1768-ENBT
- CPU type: "SLC, MicroLogix"
  - MicroLogix 1100 (with Ethernet interface onboard)
  - MicroLogix 1400 (with Ethernet interface onboard)
  - SLC 5/05 (with Ethernet interface onboard)

Performance features of communication

Permitted data types for Allen-Bradley EtherNet/IP

Permitted data types

The table lists the data types that can be used when configuring tags and area pointers.

CPU type: ControlLogix, CompactLogix

Data type	Length
Bool	1 bit
DInt	4 bytes
Int	2 bytes
Real	4 bytes
SInt	1 byte
String	1 to 80 characters
UDInt	4 bytes
UInt	2 bytes
USInt	1 byte

Permitted data types arrays

Address	Permitted data types
Array	SInt, USInt, Int, UInt, DInt, UDInt, Real
Individual bits from the basic data types of the PLC SInt, USInt, Int, UInt, DInt, UDInt	Bool*

* Any changed value of certain defined bits is written back to the PLC. There is no check to determine whether any other bits have changed. The PLC (or other PLCs) may only read access the value.

CPU type: SLC, MicroLogix

Data type	Operand type	Length
ASCII	A	0 to 80 characters
Bool	N, R, C, T, B, S, I, O	1 bit
DInt	N	4 bytes
Int	N, R, C, T, S	2 bytes
Real	N, F	4 bytes
String	ST	1 to 82 characters
UDInt	N	4 bytes
UInt	N, R, C, T, B, I, O	2 bytes

Permitted data types arrays

Address	Permitted data types
Array	Int, UInt, DInt, UDInt, Real

Distinctive features for connections with Allen-Bradley Ethernet/IP

With the communication driver Allen Bradley Ethernet/IP and the CPU type SLC, MicroLogix, you can only use array tags for discrete alarms and trends.

Note

I/O modules with 8 or 16 ports occupy one data word on the PLC.

I/O modules with 24 or 32 ports occupy two data words.

The HMI device does not output an error message if using non-existent bits.

You should always make sure that I/O modules with 8 or 24 ports only occupy the bits that are actually assigned to a port.

Supported CPU types for Allen-Bradley EtherNet/IP

CPU types

The following CPU types are supported for configuring the Allen-Bradley EtherNet/IP communication driver.

- CompactLogix
  - 1769-L2xE with Ethernet interface onboard
  - 1769-L3xE with Ethernet interface onboard
  - 1768-L4x with Ethernet module 1768-ENBT
- ControlLogix
  - 1756-L6x with Ethernet module 1756-ENBT
- GuardLogix
  - 1756-L61S with Ethernet module 1756-ENBT
  - 1756-L62S with Ethernet module 1756-ENBT
  - 1756-L63S with Ethernet module 1756-ENBT
- MicroLogix
  - MicroLogix 1100 / 1400
- SLC50x
  - SLC5/05

Addressing in the C.Logix CPU type

Addressing

Addressing

A tag is uniquely referenced in WinCC by means of an address in the PLC. The address must correspond with the tag name in the PLC. The tag address is defined by a string with a length of up to 128 characters.

Using characters for addressing

Valid characters for tag addressing:

- Letters (a to z, A to Z)
- Numbers (0 to 9)
- Underscore (_)

The tag address consists of tag name and other character strings used to specify the tag in the PLC.

Tag name properties:

- The tag name may begin but not end with an underscore character.
- Strings with successive underscore and space characters are invalid.
- The address may not exceed a length of 128 characters.

Note

The characters reserved for tag addressing may not be used in program/tag names or at any other address instance.

The reserved characters are listed below:

Reserved character	Function
.	Element delimiter
:	Definition of a program tag
,	Delimiter for addressing multi-dimensional arrays
/	Reserved for bit addressing.
[ ]	Addressing of array elements or arrays

PLC and program tags

The Allen-Bradley EtherNet/IP communication driver supports addressing of PLC tags (global project tags) and/or program tags (global program tags).

A program tag is declared based on the program name in the PLC and actual tag name which are delimited by colon. PLC tags are simply addressed by their name.

Note

Addressing errors

Addressing errors occur when the tag name and data type are inconsistent.

Note that the tag name defined in the address field in WinCC must match the tag name in the PLC. Make sure that the data types of tags in WinCC match the data types in the PLC.

Note

Module-specific tags, e.g. for data on input and output modules, cannot be addressed directly. Instead, use an alias tag in the PLC.

Example: Local:3:O. Data cannot be addressed in WinCC.

If the alias "MyOut" is defined for Local:3:O in the PLC, you can address with WinCC via MyOut.Data.

Addressing syntax

Notation of addresses

The tables below define the notation for the individual addressing options for Allen-Bradley EtherNet/IP.

Table 10-12 Access to arrays, basic data types and structure elements

Data types	Type	Address
Basic data types	PLC tag	Tag name
	Program tag	Programname:tagname
Arrays	PLC tag	Array tag
	Program tag	Program name: array tag
Bits	PLC tag	Tagname/bitnumber
	Program tag	Programname:tagname/bitnumber
Structure elements	PLC tag	Structure tag. Structure element
	Program tag	Program name: structure tag. structure element

Note

Bit addressing with the data types Bool, Real and String is not permitted and will cause an addressing fault.

Description of the syntax

Syntax description:

```
(Programname:)tagname([x(,y)(,z)]){.tagname([x(,y)(,z)])}(/
bitnumber)
```

- The "( )" defines an optional, single instance of an expression.
- The "{ }" defines an optional expression with multiple single instances.

The address string length may not exceed 128 characters.

Addressing types

Arrays

An array is a data structure that includes a number of data of the same type. WinCC only supports one-dimensional arrays.

In the address column of the tag editor, enter the array name possibly by specifying a start element. The length is defined in the Array Elements input box of the tag editor. If array limits in the PLC are exceeded (due to faulty indexing), addressing errors result.

These arrays must be declared in the PLC as controller or program tags.

Two- or three-dimensional arrays in the PLC can only be addressed in WinCC if these can be mapped area-wise onto one-dimensional arrays .

Note

During all read accesses and all write accesses, all array elements of a tag are always read or written, respectively. The contents of an array tag which is interconnected with a PLC are always transferred whenever there is a change. The HMI device and the PLC cannot concurrently write data to the same array tag for this reason. Instead of writing data only to a single element, the program writes the entire array to the PLC.

Array elements

Elements of one-dimensional, two-dimensional and three-dimensional arrays in the PLC are indexed by setting an index and the corresponding notation in the tag editor. Array addressing starts at element "0", with arrays of all basic types being valid for element addressing. Read/write operations are only carried out at the addressed element, and not for the entire array.

Bits and bit tags

Bit access is allowed to all basic data types with the exception of Bool, Real and String. Bit addressing is also allowed at array/structure elements. The Bool data type is set in WinCC when bits and bit tags in the basic data types are addressed.

Single-digit bit numbers are addressed with "/x" or "/0x" (x = bit number). Bit numbers are defined by up to two digits.

Note

With the "Bool" data type in the data types SInt, Int and DInt, after changing the specified bit the complete tag is then written in the PLC again. In the meantime, no check is made as to whether other bits in the tag have since changed. Therefore, the PLC may have only read access to the specified tag.

Structures

User-defined data types are created by means of structures. These structures group tags of different data types. Structures may consist of basic types, arrays and of other structures. In WinCC, only structure elements are addressed and not entire structures.

Structure elements

Structure elements are addressed by means of the name of the structure and of the required structure element. This addressing is separated by point. In addition to basic data types, the

structure elements may represent arrays or other structures. Only one-dimensional arrays may be used as a structure element.

Note

The nesting depth of structures is only limited by the maximum length of 128 characters for the address.

Address multiplexing

Address multiplexing

Address multiplexing is possible with the CompactLogix, ControlLogix CPU type.

Address multiplexing requires two tags:

- "Tag_1" of data type "String"; contains a logical address such as "HMI:Robot5.Block5" as value.
  The value may change to a second valid address, for example, "HMI:Robot4.Block3".
- "Tag_2" is a tag in which the "Allen-Bradley EtherNet/IP" communication driver is set up as a connection.
  Enter a valid name of an HMI_tag in square brackets as the address.
  - e.g.: "[Tag_1]"
  - The tag must be of the String data type.
  - The square brackets indicate address multiplexing.
  - The address is derived from the actual value in "Tag_1".

  Note

  You can only multiplex entire Allen-Bradley EtherNet/IP addresses. Multiplexing of address elements is not possible. "HMI:Robot[Tag_1].Block5" is an invalid address.

  You can optionally click the arrow right icon in the "Address" column. Replace the "Constant" with the "Multiplex" entry by clicking the arrow on the left edge of the next address dialog box. Now the tag selection list only returns tags of data type "String".

  You can also configure a function triggered by a "change of value" event for multiplexed tags.

Examples for addressing

Example of a table for addressing

The table below defines the basic variants for addressing PLC tags. Other addressing variants are possible by means of combination.

Type	Type	Address
General	PLC tag	Tag name
	Program tag	Program:tagname
Array	Access to an element of a 2-dimensional array	Arraytag[Dim1,Dim2]
	Element of structure array (1-dimensional)	Arraytag[Dim1].structureelement
	Bit in element basic type array (2-dimensional)	Arraytag[Dim1,Dim2]/Bit
Structure	Array in structure	Structuretag.arraytag
	Bit in the element of an array in the substructure	Structuretag.structure2.arraytag [element]/ bit

Note

Program tags are addressed by leading the address with the program name derived from the PLC with colon delimiter.

Example: Programname:arraytag[Dim1,Dim2]

Access to array elements

Type	Address
PLC tag	Arraytag[Dim1]
	Arraytag[Dim1,Dim2]
	Arraytag[Dim1,Dim2,Dim3]
Program tag	Programname:arraytag[Dim1]
	Programname:arraytag[Dim1,Dim2]
	Programname:arraytag[Dim1,Dim2,Dim3]

Examples: Communication path

Example 1:

Connection with a PLC in the same Allen-Bradley rack.

1,0

Number	Meaning
1	Stands for a backplane connection.
0	Stands for a CPU slot number.

Example 2:

Connection with a PLC in remote Allen-Bradley racks. Two Allen-Bradley racks are networked on Ethernet.

1,2,2,190.130.3.101,1,5

Number	Meaning
1	Backplane connection
2	Stands for the CPU slot number of the second Ethernet module.
2	Stands for an Ethernet connection.
190.130.3.101	IP address of a remote AB rack on the network – in particular the third Ethernet module
1	Backplane connection
5	Slot number of the CPU

Commissioning components

Transferring a project to the HMI device

1. Switch the HMI device to "transfer mode".
2. Set all necessary transfer parameters.
   - Interface
   - Transfer parameters
   - Target storage location
3. Start the project transfer.
   The project is compiled automatically.
   All compilation and transfer steps are logged to a message window.

Note

If running the CompactLogix PLC with firmware earlier than version 18, you will possibly have to restart the HMI device following the transfer of the PLC program.

You could also terminate the connection before transferring the PLC program and set up the connection again after having completed the transfer of the PLC program.

Interconnecting the PLC with the HMI device

1. Interconnect the PLC with the HMI device using a suitable cable.
2. The message "Connection to PLC .... is established" is output to the HMI device.

Optimizing the configuration

Acquisition cycle and update time

The acquisition cycles for the "Area pointers" and of the tags defined in the configuration software are decisive in terms of the update times which can actually be achieved.

The update time is equivalent to the acquisition cycle + transmission time + processing time.

Items to observe when optimizing the update times in configuration data:

- Optimize the maximum and minimum size of the data areas.
- Acquisition cycles which are too short lead to unnecessary load on overall performance. Set the acquisition cycle according to the rate of change of the process values. The rate of temperature changes at a furnace, for example, is significantly slower compared to the speed rate of an electrical drive. A time of approx. 1 second is a benchmark for the acquisition cycle.
- Avoid any gaps when entering the alarm or screen tags in a data area.
- Changes in the PLC can only be detected reliably if these are available at least within the actual acquisition cycle.

Screens

The refresh rate of screens is determined by the type and volume of data to be visualized.

Only configure short acquisition cycles for objects which actually require shorter refresh cycles. This procedure reduces update times.

Trends

The HMI device always updates all bit-triggered trends whose group bit is set in the "Trend transfer area". It resets the bits in the next cycle.

The group bit in the PLC program can only be set again after the HMI device has reset all bits.

Job mailboxes

A high rate and volume of job mailboxes transferred may lead to overload in communication between the HMI device and the PLC.

The HMI device confirms acceptance of the job mailbox by entering the value zero in the first data word of the job mailbox. The HMI device now processes the job for which it requires a certain time slice. It may take the HMI device some time to process a new job mailbox which is transferred in immediate succession to the job mailbox. The next job mailbox is only accepted if sufficient computing resources are available.

Allen-Bradley DF1

Configuring a connection via Allen-Bradley DF1

Introduction

You configure a connection to a PLC with an Allen-Bradley DF1 communication driver in the "Connections" editor of the HMI device.

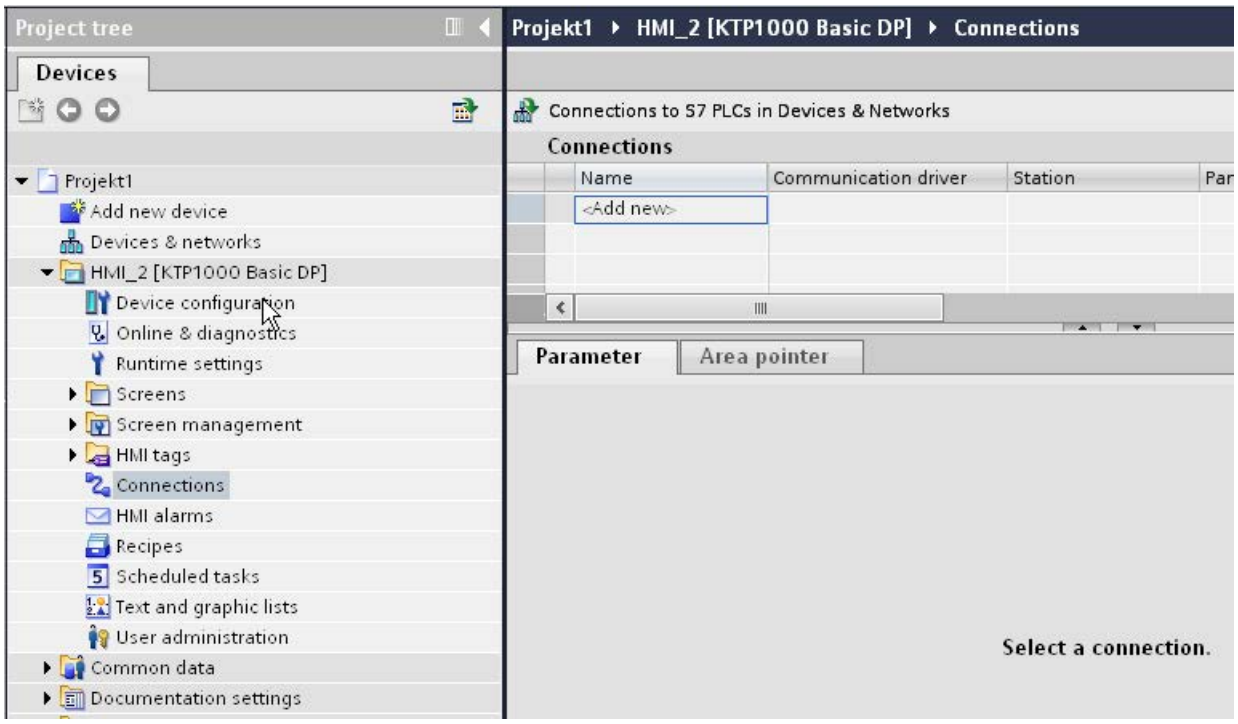
The interfaces are named differently depending on the HMI device.

Requirements

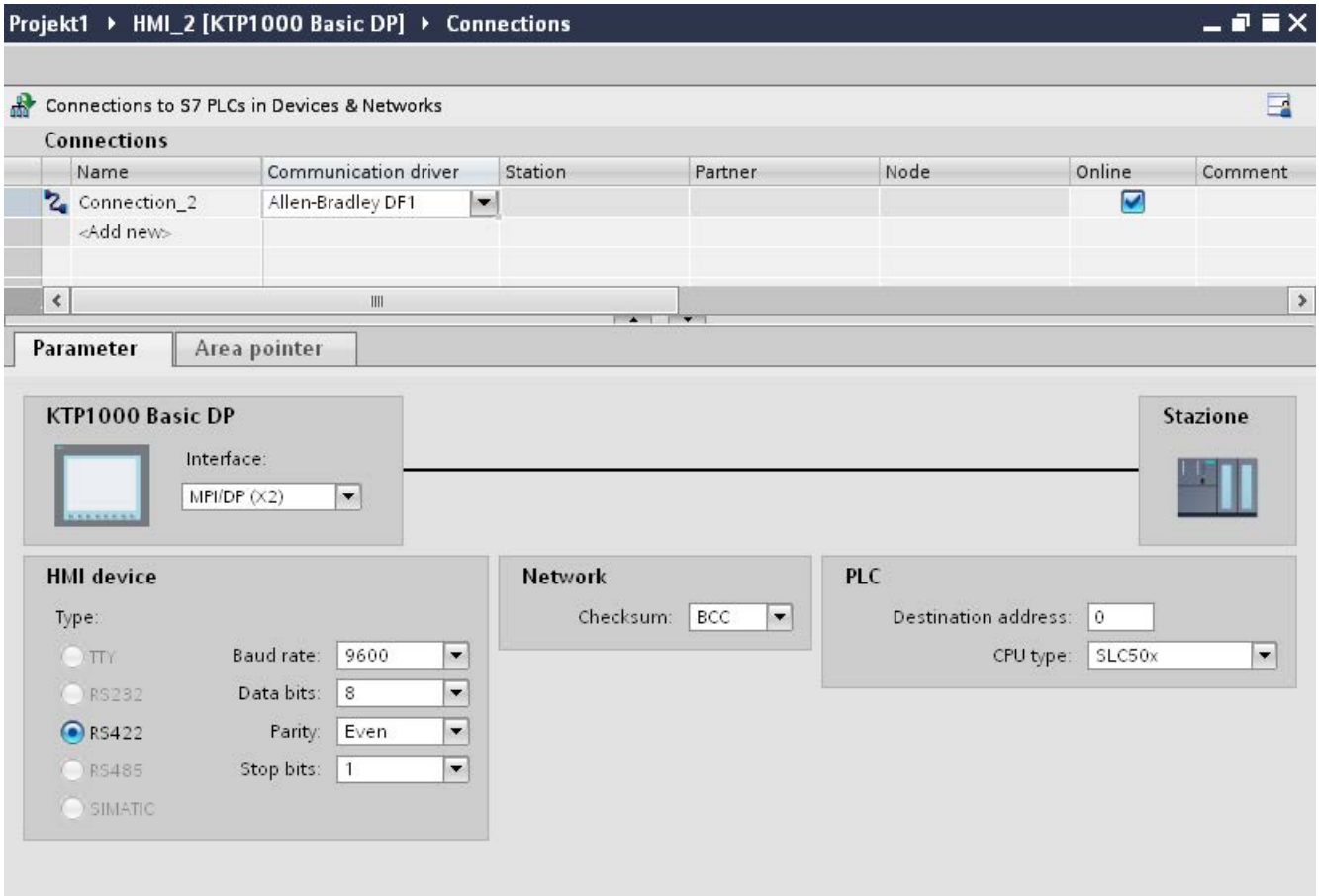
- A project is open.
- An HMI device has been created.

Procedure

1. Double-click the HMI device under "Devices" in the project tree.
2. Double-click the "Connections" item.
3. Double-click "<Add>" in the "Connections" editor.



4. In the "Communication drivers" column, select the "Allen-Bradley DF1" driver.



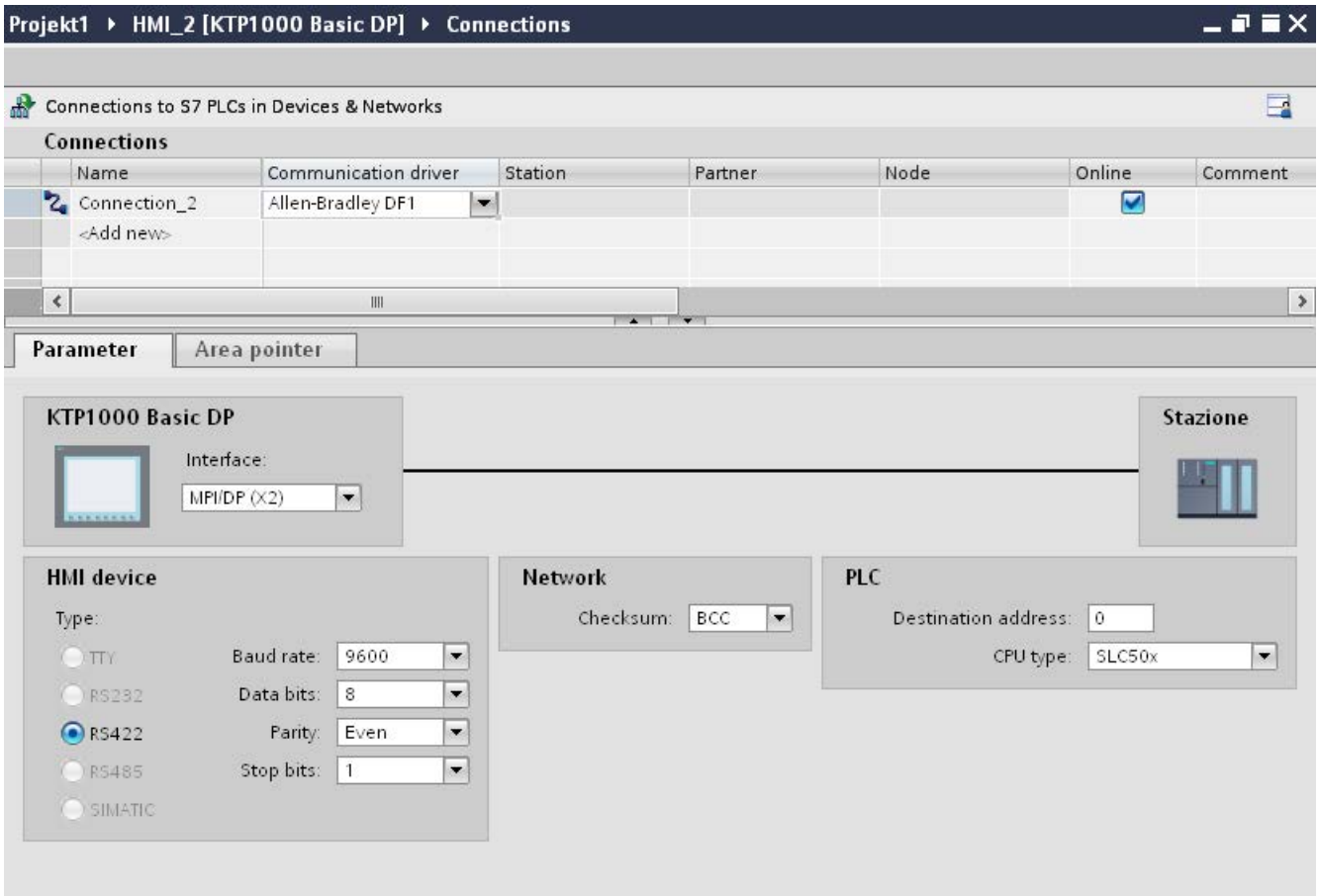
5. Select all necessary connection parameters for the interface in the Inspector window under "Parameters".

Parameters for the connection (Allen-Bradley DF1)

Parameters to be set

To set the connection parameters, such as addresses and profiles, click the connection that you have created in the "Connections" editor.

The communication partners are displayed schematically in the Inspector window under "Parameters". The "HMI device", "Network", and "PLC" areas are available for assigning parameters according to the interface used.



Parameters for the HMI device

- Interface
  Under "Interface", you select the interface of the HMI device to which the PLC is connected. For additional information, refer to the device manual for the HMI device.
- Type
  Specifies the physical connection used.

  Note

  If you are using the IF1B interface, you must also reconnect the RS-485 receive data and the RTS signal to the rear of the HMI devices via 4 DIP switches.

- Baud rate
  For "Baud rate", select the transmission speed between the HMI device and PLC.
- Data bits
  For "Data bits", you can choose between "7 bits" and "8 bits".
- Parity
  For "Parity", you can choose from "None", "Even", and "Odd".
- Stop bits
  For "Stop bits", you can choose between 1 and 2 bits.

Parameters for the network

- Checksum
  For "Checksum", choose the method for determining the error code: "BCC" or "CRC".

Parameters for the PLC

- Destination address
  For "Destination address", choose the PLC address. If there is a point-to-point DF1 connection, you set the address "0".
- CPU type
  For "CPU type", set the CPU type of the PLC used.

Note

Assign the DF1 FULL-DUPLEX driver in the CPU as follows: "NO HANDSHAKING" for "Control Line" and "AUTO-DETECT" for "Embedded Responses".

Connecting HMI device to PLC

Connections via Allen-Bradley DF1

Connection

The connection is established when you have matched the parameters of the PLC and the HMI device. Special blocks for the connection are not required in the PLC.

Note

Rockwell offers a variety of communication adapters for integrating "DF1 devices" for the DH485, DH, and DH+ networks. Of these connections, the direct connection and the connection via KF2 and KF3 module are approved. None of the other connections have been system-tested by SIEMENS AG and are therefore not approved.

Communication partners for Allen-Bradley DF1

Connectable PLCs

The communication drivers listed below support Allen-Bradley PLCs: :

PLC	DF1 (point-to-point) RS-232	DF1 (point-to-point) RS-422	DF1 (multipoint) over KF2 module to DH+ LAN RS-232/RS-422	DF1 (multipoint) over KF3 module to DH485 LAN RS-232
SLC500	–	–	–	X
SLC501	–	–	–	X
SLC502	–	–	–	X
SLC503	X	–	–	X
SLC504	X	–	X	X
SLC505	X	–	–	X
MicroLogix	X	–	–	X
PLC-5 [1]	X	X	X	–

[1] Processors released for PLC-5: PLC-5/11, PLC-5/20, PLC-5/30, PLC-5/40, PLC-5/60, and PLC-5/80.

Communication types

PLCs with Allen-Bradley DF1 communication driver

The communication between the HMI device and the following Allen Bradley PLCs is described in this section:

- SLC500
- SLC501
- SLC502
- SLC503
- SLC504
- SLC505
- PLC5
- MicroLogix

In these PLCs the connection is made by the PLC-internal protocols Allen Bradley DF1, Allen Bradley DH485 and Allen Bradley DH+.

The Allen-Bradley DF1 communication driver is used here, the protocol of which is converted into one of the other two PLC-internal protocols in multipoint communication with the communication modules KF2 (Allen Bradley DH+) and KF3(Allen Bradley DH485).

Enabled types of communication with Allen-Bradley DF1

The following communication types are system-tested and enabled:

- HMI (Allen Bradley DF1)
  Point-to-point connection
- HMI (Allen Bradley DF1)
  Via KF2 module to Allen Bradley DH+ (communication with up to 4 PLCs)
- HMI (Allen Bradley DF1)
  Via KF3 module to Allen Bradley DH485 (communication with up to 4 PLCs)

Connectable PLCs

The Allen Bradley DF1 communication driver is available for the following Allen-Bradley PLCs:

PLC	DF1 (point-to-point) RS 232	DF1 (point-to-point) RS 422	DF1 (multipoint) via KF2 module to DH+ LAN RS 232/RS 422	DF1 (multipoint) via KF3 module to DH485 LAN RS 232 [2]
SLC500	–	–	–	X
SLC501	–	–	–	X
SLC502	–	–	–	X
SLC503	X [2]	–	–	X
SLC504	X [2]	–	X	X
SLC505	X [2]	–	–	X
MicroLogix	X [2]	–	–	X
PLC-5 [1]	X	X	X	–

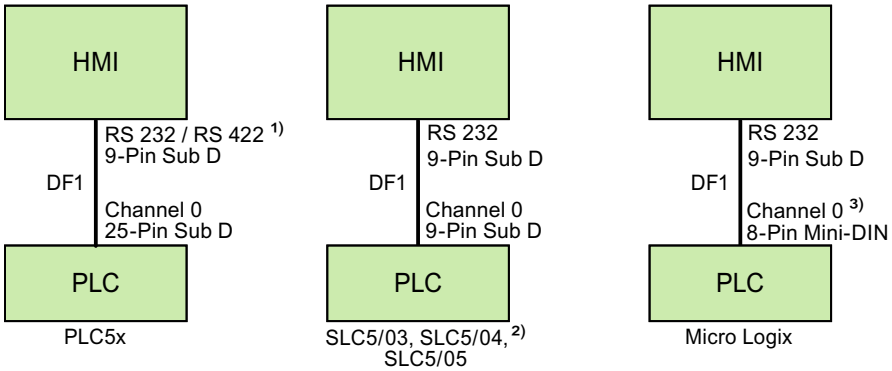
[1] Only the following processors are approved for PLC-5: PLC-5/11, PLC-5/20, PLC-5/30, PLC-5/40, PLC-5/60 und PLC-5/80.

[2] For HMI devices which only have an RS 422/485 interface and the communication partner is an RS 232 interface, the RS 422/232 converter is tested and approved.
Order number: 6AV6 671-8XE00-0AX0

DF1 protocol with multi-point connection

Point-to-point connection with DF1 protocol

Only point-to-point connections can be established with the DF1 protocol.



1) Only RS 232 is possible for Panel PC and PC.

2) A point-to-point connection to the SLC500, SLC501, and SLC502 PLCs via DF1 is not possible.

3) For MicroLogix ML1500 LRP, Channel 1 (9-pin Sub D) is also possible.

Connecting cable

HMI panel interface used	For connection to PLC5x	For connection to SLC5/03, SLC5/04, SLC5/05	For connection to MicroLogix
RS 232 9-pin	Allen-Bradley cable 1784-CP10	Allen-Bradley cable 1747-CP3	Allen-Bradley cable 1761-CBL-PM02
RS 422 9-pin	Connecting cable 9-pin Sub D RS 422	—	—

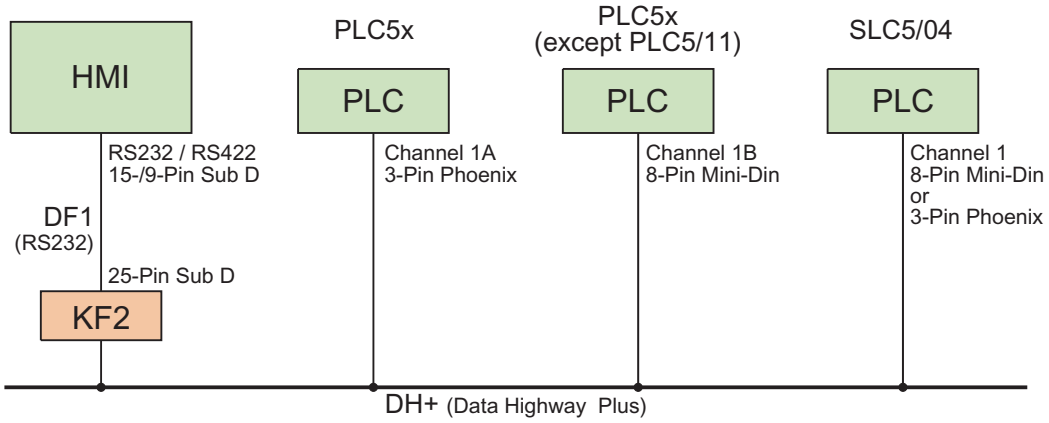
Refer to the relevant device manual to determine which HMI device interface is to be used.

The cable pin assignments can be found in Section "Connecting cables for Allen-Bradley".

DF1 protocol with multi-point connection via KF2 module

DF1 protocol with multi-point connection via KF2 module to DH+ LAN

The use of a KF2 protocol interface module enables a connection to be made to PLCs in the DH+ LAN (Data Highway Plus Local Area Network).



Connecting cable

HMI panel interface used	For connection to KF2 interface module
RS 232 9-pin	Allen-Bradley cable 1784-CP10 and 25-pin socket/socket adapter
RS 422 9-pin	9-pin Sub D RS 422 connecting cable and 25-pin female/female adapter

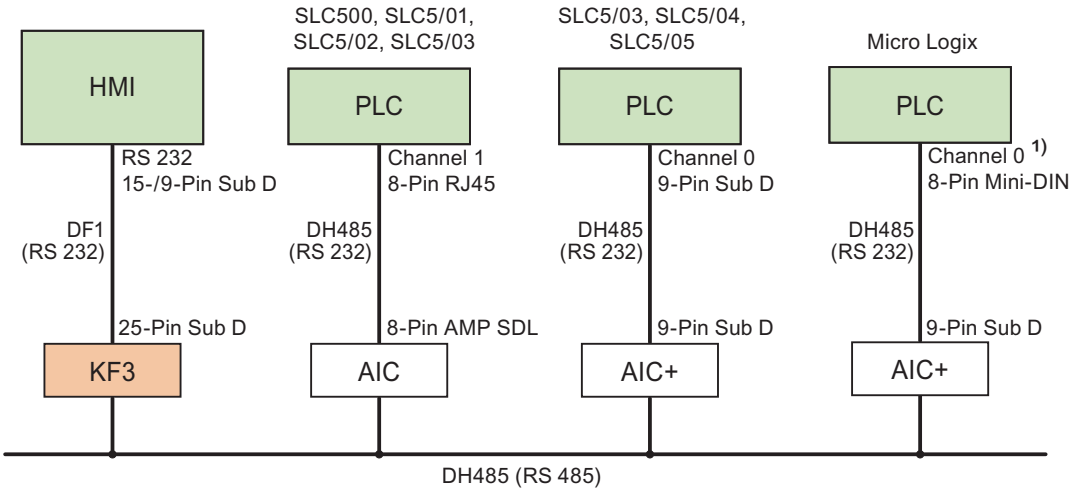
Refer to the Allen-Bradley documentation for the cable connection from the PLCs to the DH+ data bus.

Refer to the relevant device manual to determine which HMI device interface is to be used.

The cable pin assignments can be found in Section "Connecting cables for Allen-Bradley".

DF1 protocol with multi-point connection via KF3 module

DF1 protocol with multi-point connection via KF3 module to DH485 LAN



1) For MicroLogix ML1500 LRP, Channel 1 (9-pin Sub D) is also possible.

Connecting cable

HMI panel interface used	For connection to KF3 interface module
RS 232 9-pin	Allen-Bradley cable 1784-CP10 and 25-pin socket/socket adapter

Refer to the relevant device manual to determine which HMI device interface is to be used.

The cable pin assignments can be found in Section "Connecting cables for Allen-Bradley".

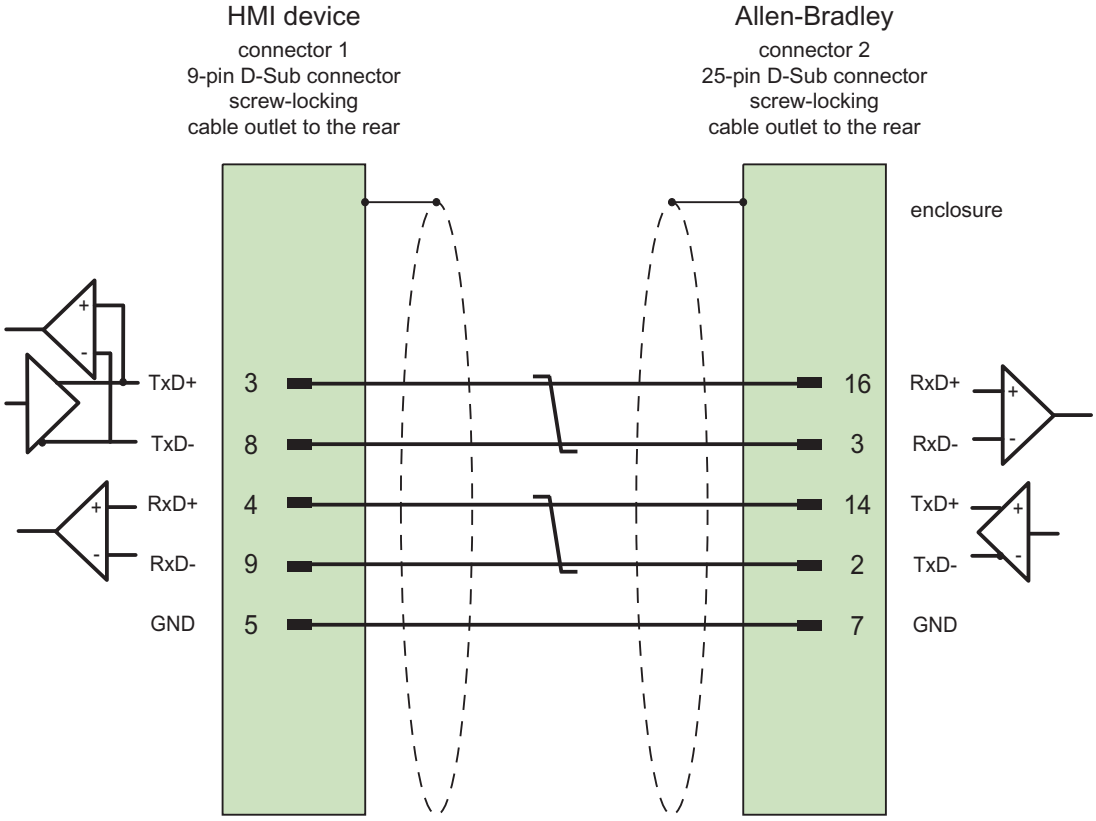
Connecting cables for Allen-Bradley DF1

Connecting cable 9-pin Sub D RS 422 for Allen-Bradley

Connecting cable 9-pin Sub D RS 422

For interconnecting the HMI device (RS 422, 9-pin sub D) - PLC5x, KF2, KF3

You require an additional 25-pin, female / female adapter (gender changer) for interconnections with KF2 and KF3.



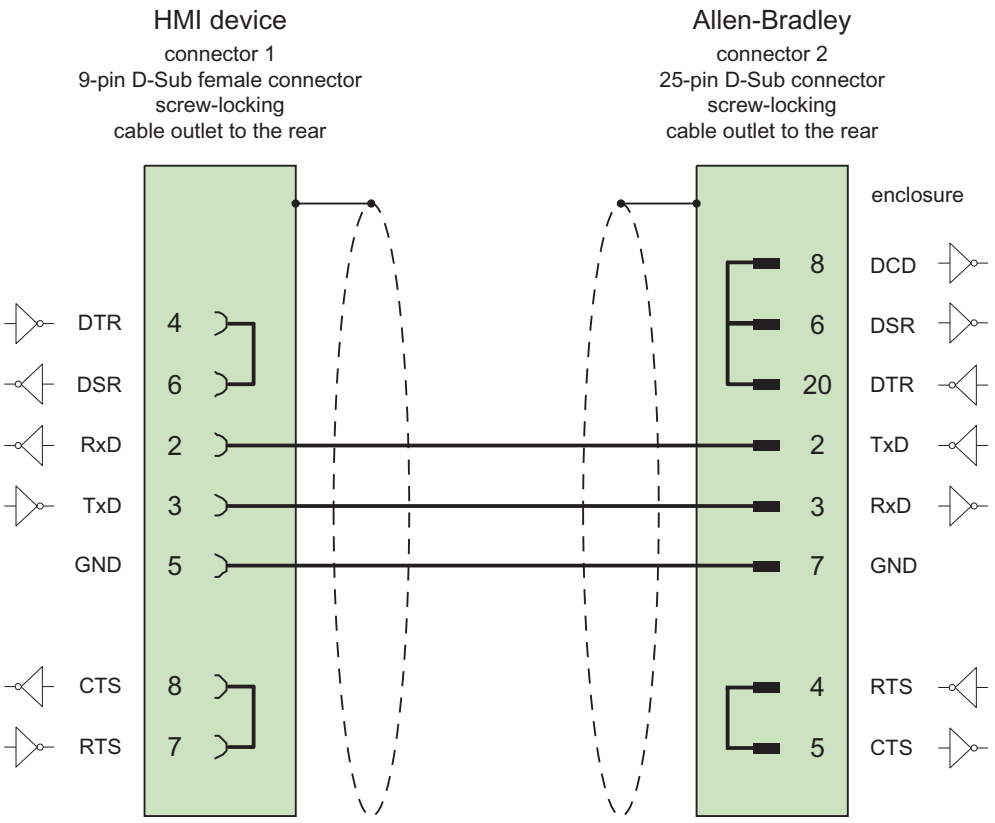
Shield with large-area contact to housing at both ends, interconnected shield contacts
Cable: 3 x 2 x 0.14 mm^2, shielded,
max. length 60 m

Connecting cable 1784-CP10, RS 232, for Allen-Bradley

Allen-Bradley cable 1784-CP10

For interconnecting the HMI device (RS 232, 9-pin sub D) - PLC5x, KF2, KF3

You require an additional 25-pin, female / female adapter (gender changer) for interconnections with KF2 and KF3.



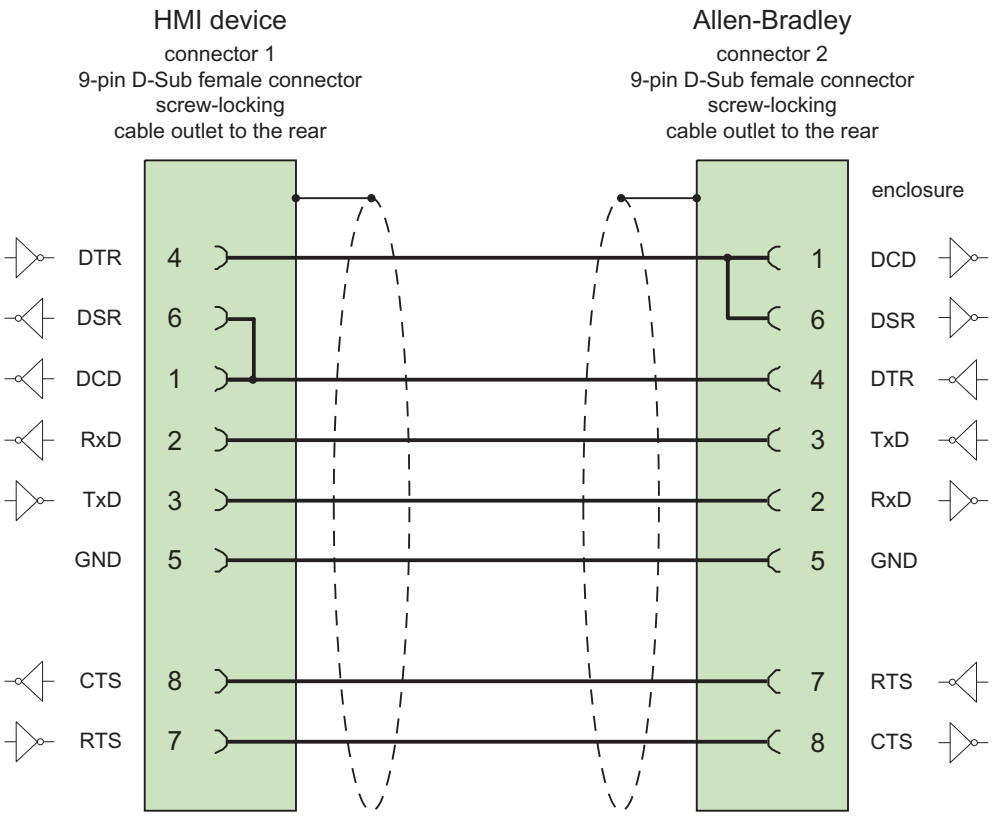
Screen connected with housing over large area on both sides

max. length 15 m

Connecting cable 1747-CP3, RS-232, for Allen-Bradley

Allen-Bradley cable 1747-CP3

For interconnecting the HMI device (RS 232, 9-pin sub D) - SLC503, SLC504, SLC505 (Channel 0), AIC+



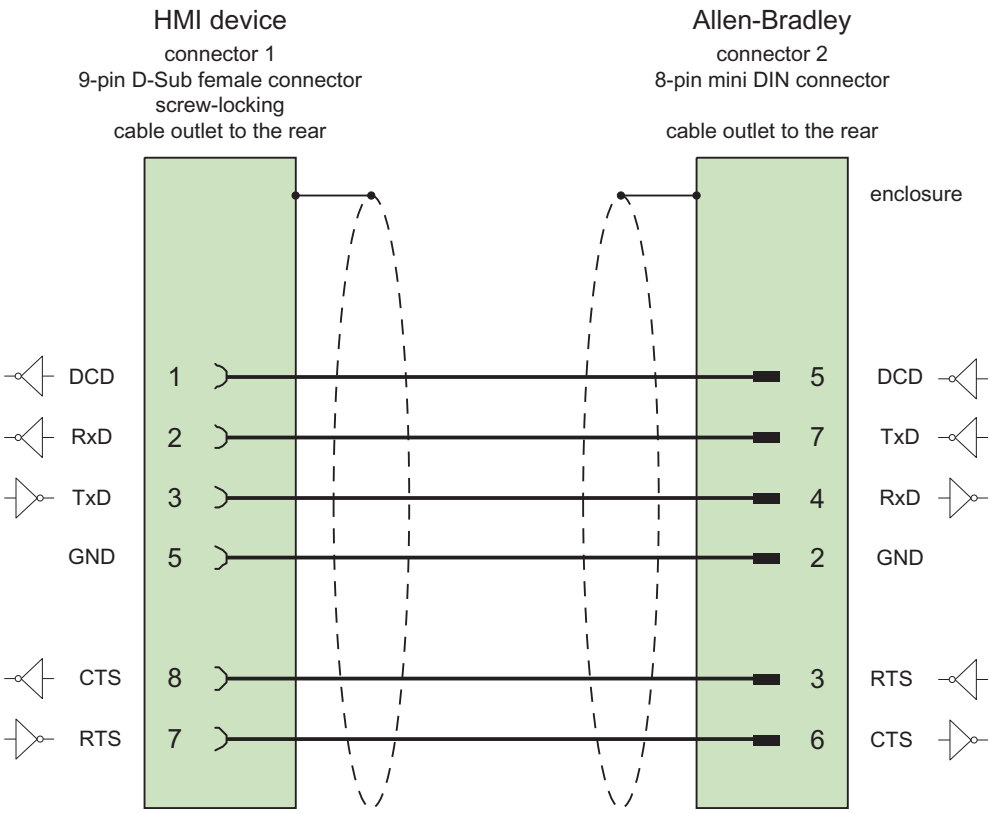
Screen connected with housing over large area on both sides

max. length 3 m

Connecting cable 1761-CBL-PM02, RS-232, for Allen-Bradley

Allen-Bradley cable 1761-CBL-PM02

For interconnecting the HMI device (RS 232, 9-pin sub D) - Micro Logix, AIC+



Screen connected with housing over large area on both sides

max. length 15 m

Performance features of communication

Permitted data types for Allen-Bradley DF1

Permitted data types for Allen-Bradley DF1

The table lists the user data types that can be used when configuring tags and area pointers.

Data type	Operand type	Length
ASCII	A [1]	1 to 80 characters
Bool	N, R, C, T, B, S, I, O	1 bit
Int	N, R, C, T, S	2 bytes
DInt	N, D [2]	4 bytes
UInt	N, R, C, T, B, I, O, D [2]	2 bytes
UDInt	N, D [2]	4 bytes
Real	N, F [1]	4 bytes

1) Selectable depending on the selected CPU type.
2) Only for PLC5 CPU type

Abbreviations

In WinCC, formats of the data types are abbreviated as follows:

- UNSIGNED INT = UInt
- UNSIGNED LONG = UDInt
- SIGNED INT = Int
- SIGNED LONG = DInt

Distinctive features for connections with Allen-Bradley DF1

With Allen Bradley DF1, array tags may only be used for discrete alarms and trends.

Note

I/O modules with 8 or 16 ports occupy one data word on the PLC.

I/O modules with 24 or 32 ports occupy two data words.

The HMI device does not output an error message if using non-existent bits.

You should always make sure that I/O modules with 8 or 24 ports only occupy the bits that are actually assigned to a port.

Supported CPU types for Allen-Bradley DF1

CPU types

The following CPU types are supported for configuring the Allen-Bradley DF1 communication driver.

- SLC
  - SLC500
  - SLC501
  - SLC502
  - SLC503
  - SLC504
  - SLC505
- MicroLogix
  - MicroLogix 1x00
  - MicroLogix 1100 / 1400
- PLC 5
  - PLC-5/11
  - PLC-5/20
  - PLC-5/40
  - PLC-5/60
  - PLC-5/80

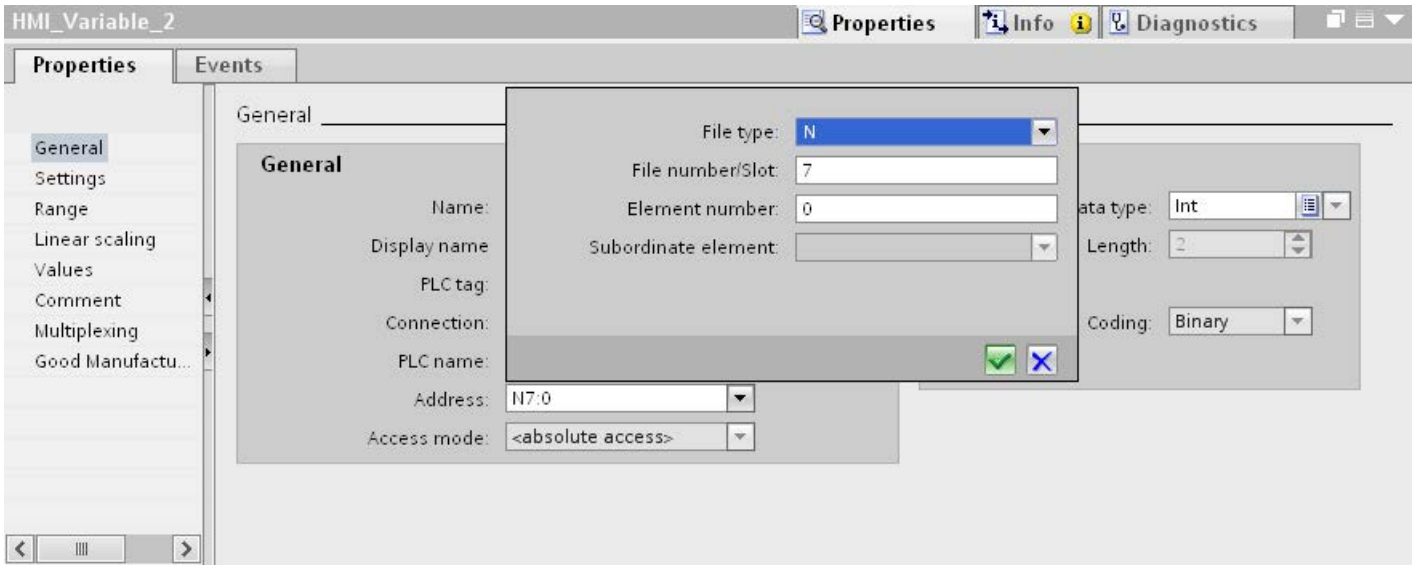
Addressing

Addressing

The addressing is entered in the following order in the Allen-Bradley DF1 communication driver:

- Operand type
- File number
- Element number

- Child element
- Bit number



The address then appears in the following format without spaces:

- File type file number : Element number . Child element
- e.g. T8:2.ACC

Operand type

You have the following options under operand type:

- I
- O
- S
- B
- T
- C
- R
- N
- A
- D only for PLC5 CPU type

File number

Select the number between two limits under file number:

- Low limit
- High limit

The limit values depend on the selected file type.

Child element

You can select a child element when you have selected one of the following data types:

- R
- C
- T

Address areas for Allen Bradley DF1

MicroLogix

Address areas	Data types					
	Bool	Int	UInt	DInt	UDInt	Real
N	N7:0/0 - N255:255/15	N7:0 - N255:255	N7:0 - N255:255	N7:0 - N255:254	N7:0 - N255:254	N7:0 - N255:254
F	--	--	--	--	F8:0 - F255:255	--
R	R6:0.EN - R255:255.ER - R255:255.DN - R255:255.FD - R255:255.IN - R255:255.EU - R255:255.EM - R255:255.UL	R6:0.LEN - R255:255.POS	R6:0.LEN - R255:255.POS	--	--	--
C	C5:0.CU - C255:255.CD - C255:255.DN - C255:255.OV - C255:255.UN	C5:0.PRE - C255:255.ACC	C5:0.PRE - C255:255.ACC	--	--	--
T	T4:0.DN - T255:255.TT - T255:255.EN	T4:0.PRE - T255:255.ACC	T4:0.PRE - T255:255.ACC	--	--	--
B	B3:0/0 - B255:255/15	--	B3:0 - B255:255	--	--	--
S	S2:0/0 - S2:65/15	S2:0 - S2:65	--	--	--	--
I	I0:0/0 - I38:255/15	--	I0:0 - I38:255	--	--	--
O	O0:0/0 - O38:255/15	--	O0:0/0 - O38:255	--	--	--

SLC500

Address areas	Data types					
	Bool	Int	UInt	DInt	UDInt	Real
N	N7:0/0 - N255:255/15	N7:0 - N255:255	N7:0 - N255:255	N7:0 - N255:254	N7:0 - N255:254	N7:0 - N255:254
R	R6:0.EN - R255:255.ER - R255:255.DN - R255:255.FD - R255:255.IN - R255:255.EU - R255:255.EM - R255:255.UL	R6:0.LEN - R255:255.POS	R6:0.LEN - R255:255.POS	--	--	--
C	C5:0.CU - C255:255.CD - C255:255.DN - C255:255.OV - C255:255.UN	C5:0.PRE - C255:255.ACC	C5:0.PRE - C255:255.ACC	--	--	--
T	T4:0.DN - T255:255.TT - T255:255.EN	T4:0.PRE - T255:255.ACC	T4:0.PRE - T255:255.ACC	--	--	--
B	B3:0/0 - B255:255/15	--	B3:0 - B255:255	--	--	--
S	S2:0/0 - S2:15/15	S2:0 - S2:15	--	--	--	--
I	I0:0/0 - I38:255/15	--	I0:0 - I38:255	--	--	--
O	O0:0/0 - O38:255/15	--	O0:0 - O38:255	--	--	--

SLC501/502

Address areas	Data types					
	Bool	Int	UInt	DInt	UDInt	Real
N	N7:0/0 - N255:255/15	N7:0 - N255:255	N7:0 - N255:255	N7:0 - N255:254	N7:0 - N255:254	N7:0 - N255:254
R	R6:0.EN - R255:255.ER - R255:255.DN - R255:255.FD - R255:255.IN - R255:255.EU - R255:255.EM - R255:255.UL	R6:0.LEN - R255:255.POS	R6:0.LEN - R255:255.POS	--	--	--

Address areas	Data types					
	Bool	Int	UInt	DInt	UDInt	Real
C	C5:0.CU - C255:255.CD - C255:255.DN - C255:255.OV - C255:255.UN	C5:0.PRE - C255:255.ACC	C5:0.PRE - C255:255.ACC	--	--	--
T	T4:0.DN - T255:255.TT - T255:255.EN	T4:0.PRE - T255:255.ACC	T4:0.PRE - T255:255.ACC	--	--	--
B	B3:0/0 - B255:255/15	--	B3:0 - B255:255	--	--	--
S	S2:0/0 - S2:32/15	S2:0 - S2:32	--	--	--	--
I	I0:0/0 - I38:255/15	--	I0:0 - I38:255	--	--	--
O	O0:0/0 - O38:255/15	--	O0:0 - O38:255	--	--	--

PLC5

Address areas	Data types						
	Bool	Int	UInt	DInt	UDInt	Real	ASCII
N	N3:0/0 - N999:999/15	N3:0 - N999:999	N3:0 - N999:999	N3:0 - N999:999	N3:0 - N999:998	N3:0 - N999:998	--
F	--	--	--	--	--	F3:0 - F999:999	--
A	--	--	--	--	--	--	A3:0 - A999:999
R	R3:0.EN - R999:999.ER - R999:999.DN - R999:999.FD - R999:999.IN - R999:999.EU - R999:999.EM - R999:999.UL	R3:0.LEN - R999:999.POS	R3:0.LEN - R999:999.POS	--	--	--	--

Address areas	Data types						
	Bool	Int	UInt	DInt	UDInt	Real	ASCII
C	C3:0.CU - C999:999.CD - C999:999.DN - C999:999.OV - C999:999.UN	C3:0.PRE - C999:999.ACC	C3:0.PRE - C999:999.ACC	--	--	--	--
T	T3:0.DN - T999:999.TT - T999:999.EN	T3:0.PRE - T999:999.ACC	T3:0.PRE - T999:999.ACC	--	--	--	--
B	B3:0/0 - B999:999/15	--	B3:0 - B999:999	--	--	--	--
S	S2:0/0 - S2:127/15	S2:0 - S2:127	--	--	--	--	--
I	I1:0/0 - I1:277/17	--	I1:0 - I1:277	--	--	--	--
O	O0:0/0 - O0:277/17	--	O0:0 - O0:277	--	--	--	--
D	D3:0/0 - D999:999/15	D3:0 - D999:999	D3:0 - D999:999	--	D3:0 - D999:998	--	--

Commissioning components

Transferring a project to the HMI device

1. Switch the HMI device to "transfer mode".
2. Set all necessary transfer parameters.
   - Interface
   - Transfer parameters
   - Target storage location
3. Start the project transfer.
   The project is compiled automatically.
   All compilation and transfer steps are logged to a message window.

Interconnecting the PLC with the HMI device

1. Interconnect the PLC with the HMI device using a suitable cable.
2. The message "Connection to PLC .... is established" is output to the HMI device.

Optimizing the configuration

Acquisition cycle and update time

The acquisition cycles for the "Area pointers" and of the tags defined in the configuration software are decisive in terms of the update times which can actually be achieved.

The update time is equivalent to the acquisition cycle + transmission time + processing time.

Items to observe when optimizing the update times in configuration data:

- Optimize the maximum and minimum size of the data areas.
- Acquisition cycles which are too short lead to unnecessary load on overall performance. Set the acquisition cycle according to the rate of change of the process values. The rate of temperature changes at a furnace, for example, is significantly slower compared to the speed rate of an electrical drive. A time of approx. 1 second is a benchmark for the acquisition cycle.
- Avoid any gaps when entering the alarm or screen tags in a data area.
- Changes in the PLC can only be detected reliably if these are available at least within the actual acquisition cycle.

Screens

The refresh rate of screens is determined by the type and volume of data to be visualized.

Only configure short acquisition cycles for objects which actually require shorter refresh cycles. This procedure reduces update times.

Trends

The HMI device always updates all bit-triggered trends whose group bit is set in the "Trend transfer area". It resets the bits in the next cycle.

The group bit in the PLC program can only be set again after the HMI device has reset all bits.

Job mailboxes

A high rate and volume of job mailboxes transferred may lead to overload in communication between the HMI device and the PLC.

The HMI device confirms acceptance of the job mailbox by entering the value zero in the first data word of the job mailbox. The HMI device now processes the job for which it requires a certain time slice. It may take the HMI device some time to process a new job mailbox which is transferred in immediate succession to the job mailbox. The next job mailbox is only accepted if sufficient computing resources are available.

Data exchange

Area pointers for Allen-Bradley

Area pointers for connections using an Allen-Bradley communication driver

You use an area pointer to access a data area in the PLC.

For more detailed information on area pointers and their configuration, refer to Section "Data exchange using area pointers".

Distinctive features for connections via Allen-Bradley EtherNet/IP

You can configure the following area pointers

Area pointer	Allen-Bradley EtherNet/IP	Allen-Bradley DF1
Screen number	Yes	Yes
Date/time	Yes	Yes
Date/time PLC	Yes	Yes
Coordination	Yes	Yes
Project ID	Yes	Yes
Job mailbox	Yes	Yes
Data record	Yes	Yes

Restrictions Allen-Bradley Ethernet/IP

The following restrictions apply for configuring area pointers.

CPU type	Data types	File types
ControlLogix,CompactLogix	Int, UInt	--
SLC, MicroLogix	Int, UInt	N, B

Restrictions Allen-Bradley DF1

The following restrictions apply for configuring area pointers.

CPU type	Data types	File types
MicroLogix	--	N, O, I, B
SLC50x	--	N, O, I, B
PLC5	--	N, O, I, B

See also

Data exchange using area pointers (Page 8745)

Trends

Trends

A trend is the graphical representation of one or more values from the PLC. The value is read out time-triggered for Basic Panels.

For additional information see:

Configuring trend displays for values from the PLC (Page 8147)

Time-triggered trends

The HMI device reads in the trend values cyclically at an interval specified in the configuration.

Time-triggered trends are suitable for continuous curves, such as the operating temperature of a motor.

Alarms

Configuring alarms

Configuring alarms for non-integrated connections

Several steps are necessary to configure alarms such as warnings, error messages and acknowledgement.

- Step 1: Create tags
- Step 2: Configure alarms
- Step 3: Configure acknowledgment

You can find additional information in the section:

Working with alarms (Page 8157)

Distinctive features when configuring alarms

If you are configuring connections of HMI devices to PLCs of other manufacturers, note the following distinctive features when configuring:

- Data types of the tags
- Addressing of tags
- How the bit positions are counted

Restrictions

Only tags whose "File type" is "N", "O", "I", "S" and "B" are allowed for use as a "trigger tag" for discrete alarms. These tags are only valid for the data types "Int" and "UInt".

Data types

For connections with an Allen-Bradley communication driver, the following data types are supported:

Communication drivers	PLC	Permitted data types	
		Discrete alarms	Analog alarms
Allen-Bradley DF1	SLC500, SLC501, SLC502, SLC503, SLC504, SLC505, PLC5, MicroLogix	Int, UInt	Int, UInt, Long, ULong, Real
Allen-Bradley EtherNet/IP	ControlLogix, CompactLogix, SLC, Micrologix	Int, UInt	SInt, USInt, Int, UInt, DInt, UDInt, Real

How the bit positions are counted

For connections with an Allen-Bradley communication driver, the following counting method applies:

How the bit positions are counted	Left byte								Right byte							
In Allen-Bradley PLCs	15							8	7							0
In WinCC you configure:	15							8	7							0

See also

Alarm system in WinCC (Page 8148)

Acknowledgment of alarms

Procedure

Create suitable tags on the PLC to acknowledge an error alarm. You assign these tags to an alarm in the "Bit messages" editor. You make the assignment in "Properties > Acknowledgment".

Distinction in terms of acknowledgment:

- Acknowledgment by the PLC
- Acknowledgment on the HMI device

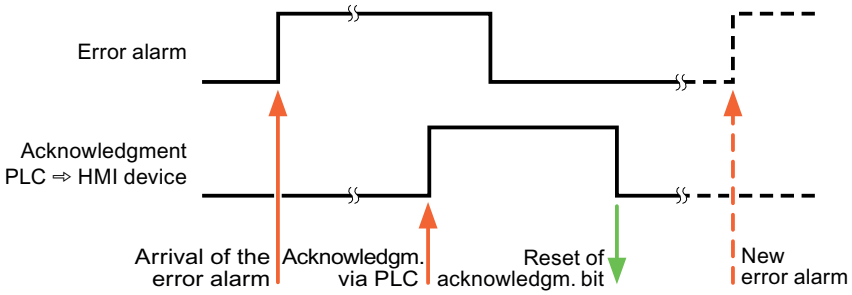
Acknowledgment by the PLC

In "PLC acknowledgment tag", configure the tag or array tag and the bit number that the HMI device uses to identify a PLC acknowledgment.

A bit set in the tag triggers acknowledgment of the assigned error alarm bit at the HMI device. This tag bit returns a function similar to acknowledgment on the HMI device which is triggered by pressing the "ACK" button, for example.

The acknowledgment bit must be located in the same tag as the bit for the error alarm.

Reset the acknowledgment bit before setting the bit in the alarm area again. The figure below shows the pulse diagram.



Acknowledgment on the HMI device

In the "HMI acknowledgment tag" area, configure the tag or array tag as well as the bit number that the HMI device writes to the PLC after acknowledgment. Make sure when you use an array tag that it is not longer than 6 words.

To always create a signal change when setting an assigned acknowledgment bit of a discrete alarm that must be acknowledged, the HMI device will reset the acknowledgment bit assigned to the alarm as soon as it detects an alarm subject to acknowledgment and write the acknowledgment tag in the PLC. There will be a certain delay between detecting the message and writing the acknowledgment tag in the PLC because the HMI device has to process the operations.

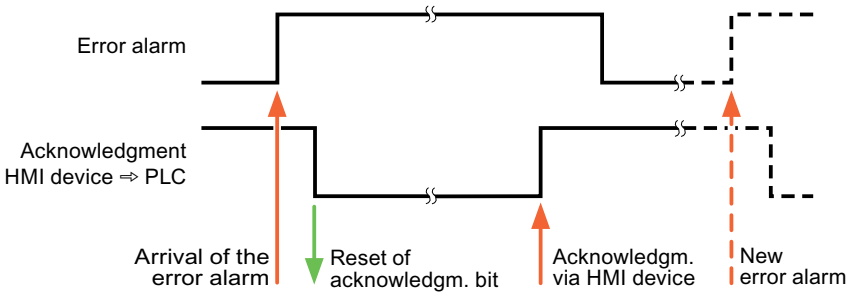
If a discrete alarm subject to acknowledgment is acknowledged by the HMI device, then the corresponding bit in the assigned acknowledgment tag will be set. The entire acknowledgment tag is then written to the PLC by the HMI device. This allows the PLC to recognize that a certain alarm message has been acknowledged at the HMI device.

Note

All alarm bits acknowledged since the last Runtime start will remain in the acknowledgment tag until a new incoming of the respective discrete alarms is detected.

This area should only be read by the PLC because the entire section of the HMI device will be overwritten once the next acknowledgment tag is written.

The figure below shows the pulse diagram.



Mitsubishi

Mitsubishi communication drivers

Introduction

This section describes the communication between an HMI device and PLCs that use Mitsubishi communication drivers.

The following communication drivers are supported:

- Mitsubishi MC TCP/IP
- Mitsubishi FX

Data exchange

Data is exchanged by means of tags or area pointers.

- Tags
  The PLC and the HMI device use process values for data exchange. You create tags in the configuration that point to addresses in the PLC. The HMI device reads the value from the defined address, and then displays it. The operator may also enter values on the HMI device, which are then written to the address in the PLC.
- Area pointers
  Area pointers are used to exchange specific data and are only set up when these data are used.

Mitsubishi MC TCP/IP

Configuring a connection via Mitsubishi MC TCP/IP

Introduction

You configure a connection to a PLC with a Mitsubishi MC TCPI/IP communication driver in the "Connections" editor of the HMI device.

The Ethernet interfaces are named differently depending on the HMI device.

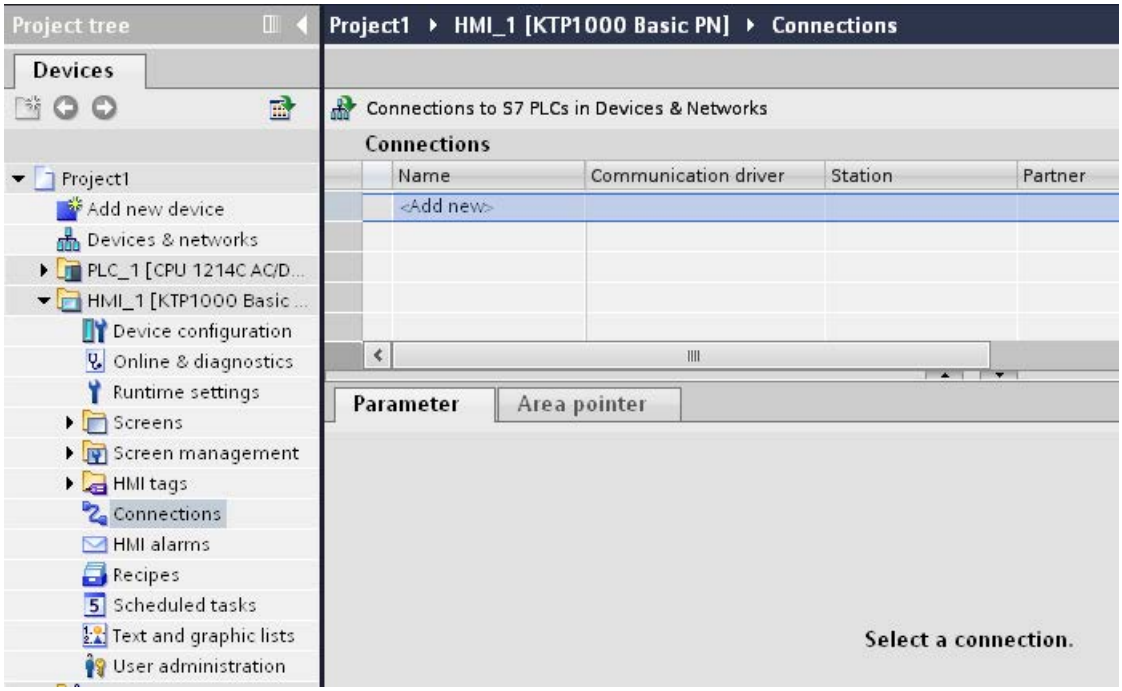
Example: PROFINET interface corresponds to the Ethernet interface

Requirements

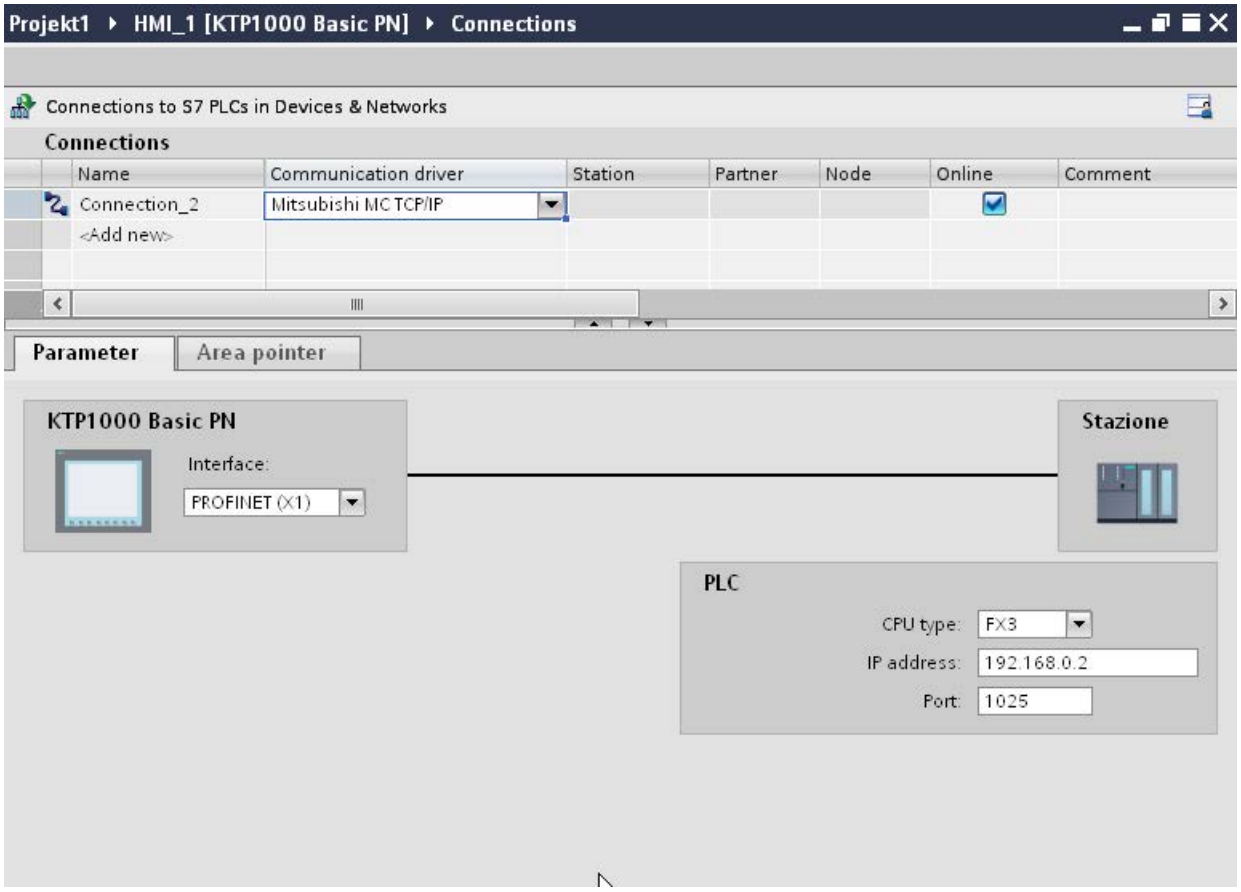
- A project is open.
- An HMI device has been created.

Procedure

1. Double-click the HMI device under "Devices" in the project tree.
2. Double-click the "Connections" item.
3. Double-click "<Add>" in the "Connections" editor.



4. In the "Communication drivers" column, select the "Mitsubishi MC TCPI/IP" driver.



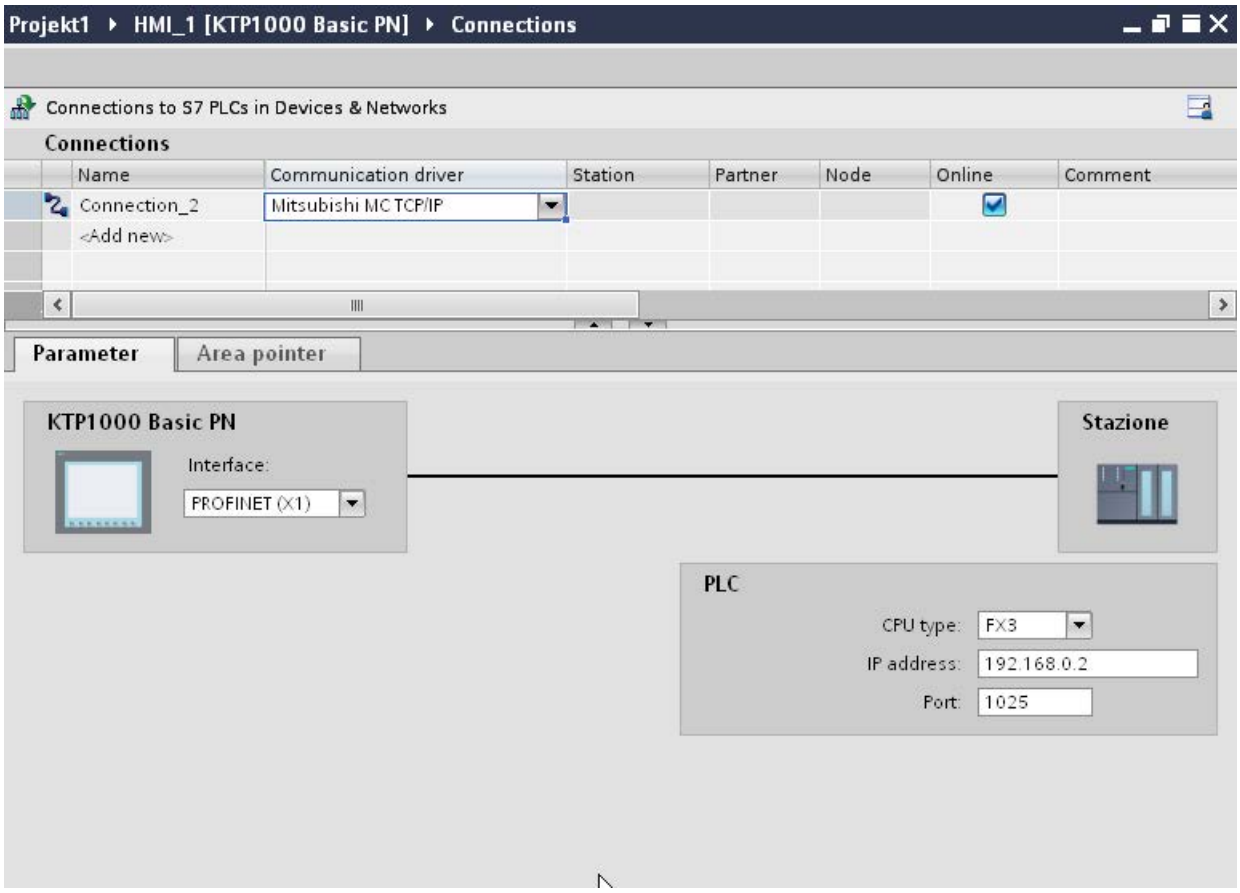
5. Select all necessary connection parameters for the interface in the Inspector window under "Parameters".

Parameters for the connection (Mitsubishi MC TCP/IP)

Parameters to be set

To set the connection parameters, such as addresses and profiles, click the connection that you have created in the "Connections" editor.

The communication partners are displayed schematically in the Inspector window under "Parameters". The "HMI device" and "PLC" areas are available for assigning parameters according to the interface used.



Parameters for the HMI device

You can select only one interface for the HMI device in the Inspector window under "Parameters". Depending on the HMI device, there are several interfaces available.

If you are directly connected to the HMI device during configuration, you can set up the IP address of the HMI device in WinCC. The IP address is transferred to the HMI device during project transfer.

Note

The IP address in the control panel will be overwritten upon subsequent loading if you have already set up the IP address in the HMI device control panel.

The IP address already set up in the control panel will be retained upon subsequent loading if you activate "Set IP address using a different method".

To set up the IP address of the HMI device:

1. Click on the HMI device.
2. Open the "Device configuration" editor.

3. Click the Ethernet interface.
4. Assign the IP address in the inspector window under:
   "General > PROFINET interface > Ethernet addresses"

Parameters for the PLC

- CPU type
  For "CPU type", you set the type of PLC to which the HMI device is connected.
  The following settings are possible:
  –FX3
  –Q
  If you select the FX3 CPU type, the Mitsubishi MC protocol "1E" is used and "3E" for the "Q" CPU type.
  The "Binary code" protocol variant is always used.

  Note

  If the CPU type is changed for a configured connection, tags with the following properties must be revised:

  - Operands that do not exist for the new CPU type, such as "W", "B", "F".
  - Inputs and outputs with different addressing (hexadecimal/octal)
  - Addresses greater than the valid address area of the new CPU type

- IP address
  Set the IP address or host name of the Ethernet/IP module of the PLC. Only the IP address can be used on a Basic Panel.
- Port
  Set the port number of the module of the PLC.

Connecting HMI device to PLC

Connections via Mitsubishi MC TCP/IP

Connection

The HMI device can be connected to the Mitsubishi PLC using the following components:

- Existing Ethernet network that also contains the PLCs
- Cross-over Ethernet cable connected directly to the Ethernet interface of the CPU or the communication module

The connection of the HMI device to a Mitsubishi PLC is limited primarily to the physical connection of the HMI device. Special blocks for the connection are not required in the PLC.

Connect the HMI device to one or several Q-series and/or FX3 PLCs. Connect the HMI device via the following interfaces:

- Communication interface OnBoard
- Approved communication module suitable for the PLC

Note

Timeout response with TCP/IP (Ethernet)

Due to the use of the TCP/IP protocol, the breakdown of a connection is detected at the earliest after approximately one minute. Communication failure cannot be reliably detected if no tags are requested, for example, no output tags in the current screen.

Configure area pointer coordination for each PLC. This setting ensures that a communication failure is recognized after approximately two minutes, even in the aforementioned scenario.

Communication types

Approved communication types

- Only applies for Mitsubishi FX(PG protocol):
  The point-to-point connection from a HMI device to an approved Mitsubishi FX–CPU via Mitsubishi FX is system-tested and approved by Siemens AG.
- Only applies for Mitsubishi MC TCP/IP:
  The following communication types are system-tested and approved:
  - Point-to-point connection to the approved PLCs
  - Multipoint connection from a HMI device with up to 4 PLCs with the respectively approved PLCs. CPU types (FX3 and Q) can be mixed.

Note

The HMI device is a client and the PLC must operate as a server.

Connectable PLCs

Connections can be implemented for the following Mitsubishi PLCs:

PLC	Mitsubishi FX (PG protocol)	Mitsubishi MC TCP/IP
MELSEC FX1n, FX2n	Yes	No
MELSEC FX3U, FX3UC, FX3G with communication module FX3U-ENET	No	Yes
MELSEC System Q • Q-series with the communication module QJ71E71-100 • QnUDEH CPU with Ethernet interface onboard	No	Yes

Parameterization of the communications modules

FX3 PLCs

Procedure

1. Start the FX-Configurator.
2. Select the module.
3. Assign the following settings in the "Operational settings" dialog:
   - Communication data code:
     Binary code
   - Initial timing:
     Always wait for OPEN
   - IP address:
     IP address
   - Send frame setting:
     Ethernet(V2.0)
   - TCP Existence confirmation setting:
     Use the Ping

4. Assign the following settings in the "Open Settings" dialog:
   - Protocol:
     TCP
   - Open system:
     Unpassive
   - Fixed buffer:
     Receive
   - Fixed buffer communication procedure:
     Procedure exist(MC)
   - Pairing open
     Disable
   - Existence confirmation
     No confirm
   - Host station Port No. (DEC)
     Port number

   Note

   The port number chosen in the communication module must match the port number in WinCC. A connection with a port number must be assigned for each connected HMI device.

   You must specify port numbers in decimal values.

5. Confirm the default settings of the other dialog boxes.

The network no. and station no. parameters are not relevant for the connection and can be chosen as required.

Q PLCs

Procedure

1. Click "Edit network parameters".
2. Select the network type:
   - Ethernet
     The network number and the group / station number are not evaluated and can be freely assigned

3. Assign the following settings in the "Operational settings" dialog:
   - Communication data code:
     Binary code
   - Initial timing:
     Always wait for OPEN
   - IP address:
     IP address
   - Send frame setting:
     Ethernet(V2.0)
   - Enable write operations during RUN
4. Assign the following settings in the "Open settings" dialog:
   - Protocol:
     TCP
   - Open system:
     Unpassive
   - Pairing open
     Disable
   - Existence confirmation
     No confirm
   - Host station Port No. (HEX)
     Port-Nummer

Note

The port number chosen in the communication module must match the port number in WinCC. A connection with a port number must be assigned for each connected HMI device.

You must specify port numbers in hexadecimal values.

Internal Ethernet port of the Q0xUDEH CPU

Procedure

1. Assign the following settings in the "Internal Ethernet Port" dialog:
   - IP address:
     IP address
   - Communication data code:
     Binary code
   - Enable online changes
2. Assign the following settings in the "Open settings" dialog:
   - Protocol:
     TCP
   - Open system:
     MC-Protocol
   - Host station Port No. (HEX)
     Port number

   Note

   The port number chosen in the communication module must match the port number in WinCC. A connection with a port number must be assigned for each connected HMI device.

Performance features of communication

Permitted data types for Mitsubishi MC TCPI/IP

Permitted data types

The table lists the data types that can be used when configuring tags and area pointers.

Data type	Operand type	Length
4-bit block	M, X, Y, B, F	1 byte
8-bit block	M, X, Y B, F	1 byte
12-bit block	M, X, Y B, F	2 bytes
16-bit block	M, X, Y B, F	2 bytes
20-bit block	M, X, Y B, F	4 bytes
24-bit block	M, X, Y B, F	4 bytes
28-bit block	M, X, Y B, F	4 bytes
32-bit block	M, X, Y B, F	4 bytes
Bool	M, D, X, Y B, F	1-bit

Data type	Operand type	Length
DInt	D, W	4 bytes
DWord	D, C, W	4 bytes
Int	D, W	2 bytes
Real 1)	D, W	4 bytes
String 1)	D	1 to 80 characters
Word	D, T, C, W	2 bytes

1) The "String" and "Real" data types are not available for all CPUs.
2) Operand types B, F and W are only available for CPU type "Q".

Note

Note the following for write accesses:

Tags can only be written if "Enable online changes" or "Enable write operations during RUN" was selected when parameterizing the Mitsubishi communication modules.

For data type "Bool" in operand type "D", the entire word is written back to the PLC following a change to the specified bit. There is no check to determine whether any other bits in the word have changed. As a result, the PLC only has read access to the specified word.

Note

Array elements in I/O fields cannot be used in communication with a Mitsubishi PLC.

Supported CPU types for Mitsubishi MC TCP/IP

CPU types

The following CPU types are supported for configuring the Mitsubishi MC TCP/IP communication driver.

- FX3 series
  - FX 3G / FX 3G with communication modul FX3U-ENET
  - FX 3U / FX 3U with communication modul FX3U-ENET
  - FX 3UC / FX 3UC with communication modul FX3U-ENET
- Q series
  - Q-Series with QJ71E71-100 communication module
- iQ series / QnUD
  - QnUDEHCPU with built in ethernet module

Addresses for Mitsubishi MC TCP/IP

Address areas for connections via Mitsubishi MC TCP/IP

The address area boundaries differ for the different series; refer to the Mitsubishi Computerlink manuals for this information.

Examples of address area boundaries dependent on the CPU and communication format:

Name	Operand type	Max. address FX3	Max. address Q-Series
Output/Input	Y/X	Octal X/Y 0 - 777	HEX X/Y 0 - 7FF
Bit memory	M	M0 - M3071 and M8000 - M8255	M/L/S 0 - 8191
Data register	D	D0 - 7999 D8000 - D8255	D0 - 8191 D9000 - D9255 becomes SD1000 - SD1255
Counter	C	C0 - 255	C0 - 1023
Timer	T	T0 - 255	T0 - 2047
Link register	W	--	Hex: W0 - FFF
Link flag	B	--	Hex: B0 - FFF
Error flag	F	--	F0 - 2047

Address areas for Mitsubishi MC TCP/IP

FX0

Address areas	Data types												
	Bool	Word	DWord	Real	String	4-bit block	8-bit block	12-bit block	16-bit block	20-bit block	24-bit block	28-bit block	32-bit block
M	M0 - M9999	--	--	--	--	M0 - M9996	M0 - M9992	M0 - M9988	M0 - M9984	M0 - M9980	M0 - M9976	M0 - M9972	M0 - M9968
D	D0.0 - D999.15	D0 - D999	D0 - D998	D0 - D998	D0 - D998	--	--	--	--	--	--	--	--
T		T0 - T255	--	--	--	--	--	--	--	--	--	--	--
C-16-Bit	--	C-16-Bit 0 - C-16-Bit 199	--	--	--	--	--	--	--	--	--	--	--

Address areas	Data types												
	Bool	Word	DWord	Real	String	4-bit block	8-bit block	12-bit block	16-bit block	20-bit block	24-bit block	28-bit block	32-bit block
C-32-Bit	--	--	C-32-Bit 200 - C-32-Bit 255	--	--	--	--	--	--	--	--	--	--
X	X0 - X255	--	--	--	--	X0 - X252	X0 - X248	X0 - X244	X0 - X240	X0 - X236	X0 - X232	X0 - X228	X0 - X224
Y	Y0 - X255	--	--	--	--	Y0 - Y252	Y0 - Y248	Y0 - Y244	Y0 - Y240	Y0 - Y236	Y0 - Y232	Y0 - Y228	Y0 - Y224

Commissioning components

Transferring a project to the HMI device

1. Switch the HMI device to "transfer mode".
2. Set all necessary transfer parameters.
   - Interface
   - Transfer parameters
   - Target storage location
3. Start the project transfer.
   The project is compiled automatically.
   All compilation and transfer steps are logged to a message window.

Interconnecting the PLC with the HMI device

1. Interconnect the PLC with the HMI device using a suitable cable.
2. The message "Connection to PLC .... is established" is output to the HMI device.

Optimizing the configuration

Acquisition cycle and update time

The acquisition cycles for the "Area pointers" and of the tags defined in the configuration software are decisive in terms of the update times which can actually be achieved.

The update time is equivalent to the acquisition cycle + transmission time + processing time.

Items to observe when optimizing the update times in configuration data:

- Optimize the maximum and minimum size of the data areas.
- Acquisition cycles which are too short lead to unnecessary load on overall performance. Set the acquisition cycle according to the rate of change of the process values. The rate of temperature changes at a furnace, for example, is significantly slower compared to the speed rate of an electrical drive. A time of approx. 1 second is a benchmark for the acquisition cycle.
- Avoid any gaps when entering the alarm or screen tags in a data area.
- Changes in the PLC can only be detected reliably if these are available at least within the actual acquisition cycle.

Screens

The refresh rate of screens is determined by the type and volume of data to be visualized.

Only configure short acquisition cycles for objects which actually require shorter refresh cycles. This procedure reduces update times.

Trends

The HMI device always updates all bit-triggered trends whose group bit is set in the "Trend transfer area". It resets the bits in the next cycle.

The group bit in the PLC program can only be set again after the HMI device has reset all bits.

Job mailboxes

A high rate and volume of job mailboxes transferred may lead to overload in communication between the HMI device and the PLC.

The HMI device confirms acceptance of the job mailbox by entering the value zero in the first data word of the job mailbox. The HMI device now processes the job for which it requires a certain time slice. It may take the HMI device some time to process a new job mailbox which is transferred in immediate succession to the job mailbox. The next job mailbox is only accepted if sufficient computing resources are available.

Mitsubishi FX

Configuring a connection via Mitsubishi FX

Introduction

You configure a connection to a PLC with a Mitsubishi FX communication driver in the "Connections" editor of the HMI device.

The Mitsubishi FX protocol is also referred to as the Mitsubishi PG protocol.

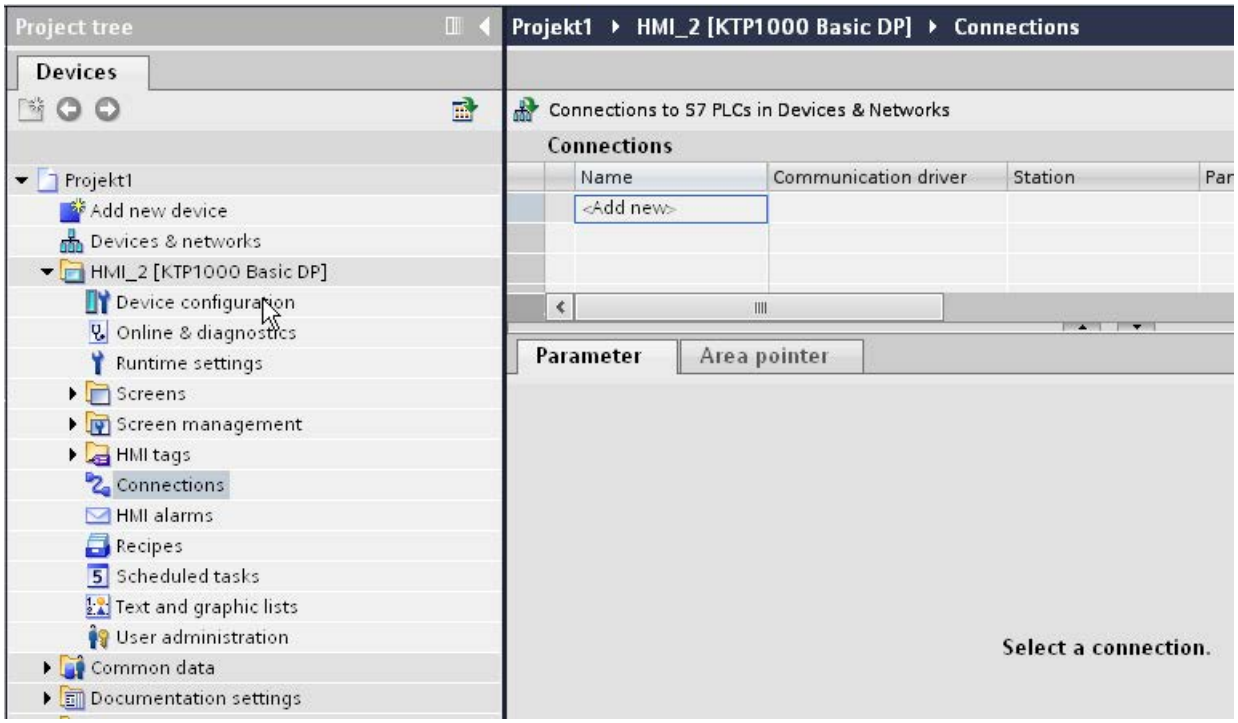
The interfaces are named differently depending on the HMI device.

Requirements

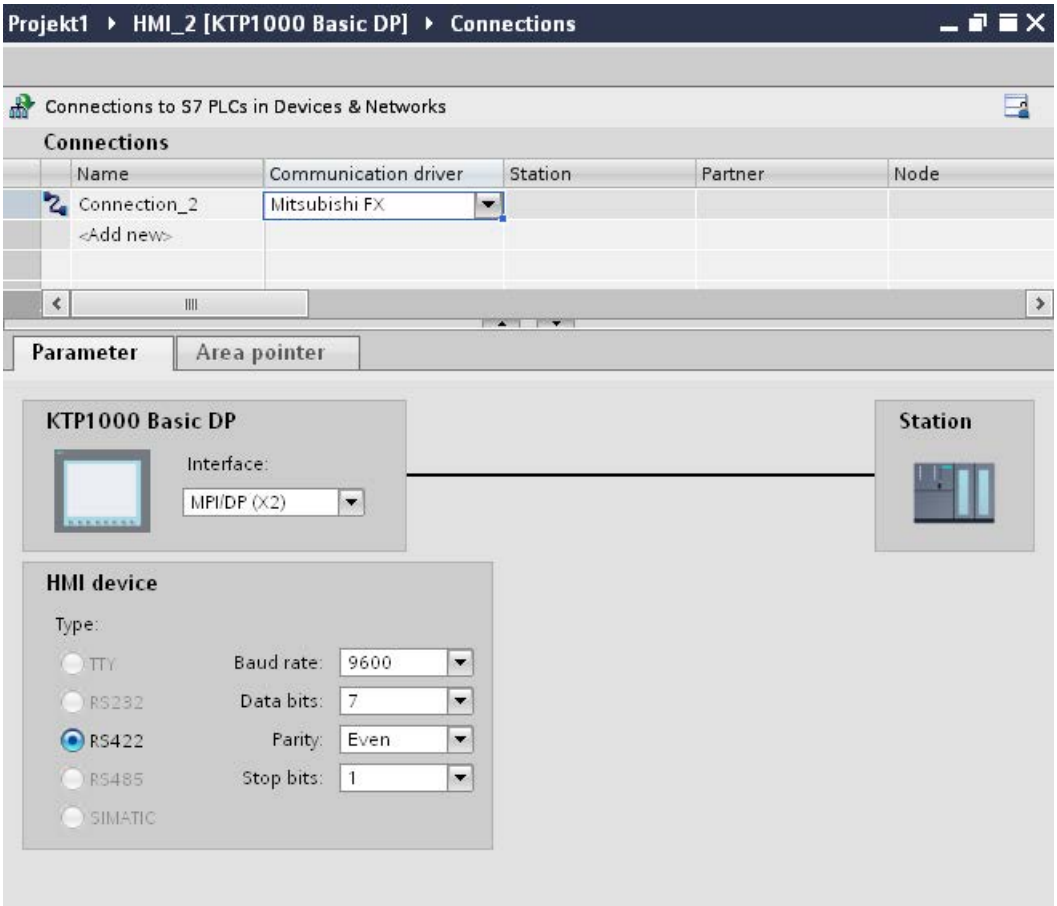
- A project is open.
- An HMI device has been created.

Procedure

1. Double-click the HMI device under "Devices" in the project tree.
2. Double-click the "Connections" item.
3. Double-click "<Add>" in the "Connections" editor.



4. In the "Communication drivers" column, select the "Mitsubishi FX" driver.



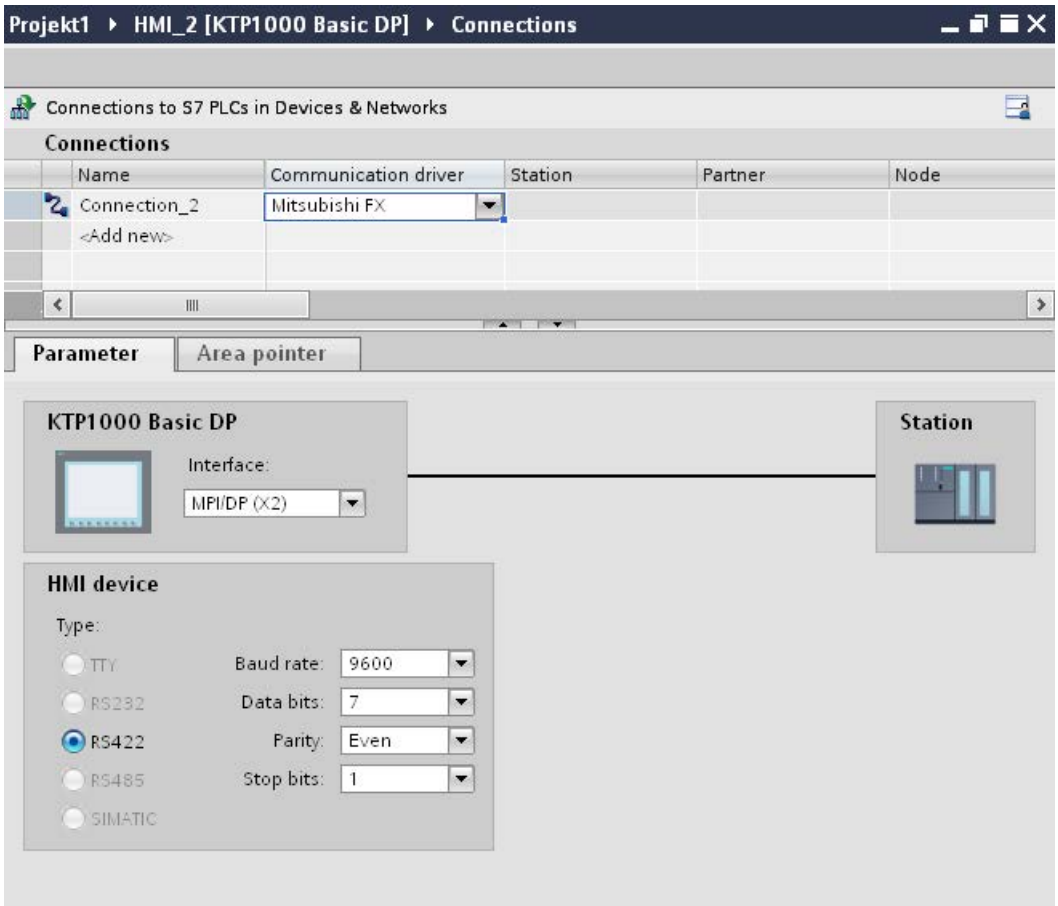
5. Select all necessary connection parameters for the interface in the Inspector window under "Parameters".

Parameters for the connection (Mitsubishi FX)

Parameters to be set

To assign the connection parameters, such as addresses and profiles, click the connection that you have created in the "Connections" editor.

The communication partners are displayed schematically in the Inspector window under "Parameters". The "HMI device", "Network", and "PLC" areas are available for assigning parameters according to the interface used.



Parameters for the HMI device

You can select an interface for the HMI device in the Inspector window under "Parameters". Depending on the HMI device, there are several interfaces available.

- "Type"
  Specifies the physical connection used.

  Note

  If you use the IF1B interface, you must switch over the RS422 receive data and the RTS signal additionally by 4 DIL-switches on the back of the HMI device.

Parameters for the PLC

- Baud rate: For "Baud rate", select the transmission speed between the HMI device and PLC. Select the baud rate "9600".
- Data bits: Under "Data bits", select "7 bits".
- Parity: Under "Parity", select "Even".
- Stop bits: Under "Stop bits", select "1 bit".

Connecting HMI device to PLC

Communication types

Approved communication types

- Only applies for Mitsubishi FX(PG protocol):
  The point-to-point connection from a HMI device to an approved Mitsubishi FX–CPU via Mitsubishi FX (PG protocol := protocol for access to the program and memory elements of the FX series PC CPU version V1.21 and after) is system-tested and approved by Siemens AG.
- Only applies for Mitsubishi MC TCP/IP:
  The following communication types are system-tested and approved:
  – Point-to-point connection to the approved PLCs
  – Multipoint connection from a HMI device with up to 4 PLCs with the respectively approved PLCs. CPU types (FX3 and Q) can be mixed.

  Note

  The HMI device is a client and the PLC must operate as a server.

Connectable PLCs

Connections can be implemented for the following Mitsubishi PLCs:

PLC	Mitsubishi FX (PG protocol)	Mitsubishi MC TCP/IP
MELSEC FX1n, FX2n	Yes	No
MELSEC FX3U, FX3UC, FX3G with communication module FX3U-ENET	No	Yes
MELSEC System Q • Q-series with the communication module QJ71E71-100 • QnUDEH CPU with Ethernet interface onboard	No	Yes

Connections via Mitsubishi FX

Connection

Connect the HMI device to the programming interface of the CPU (RS 422) (see documentation of the PLC).

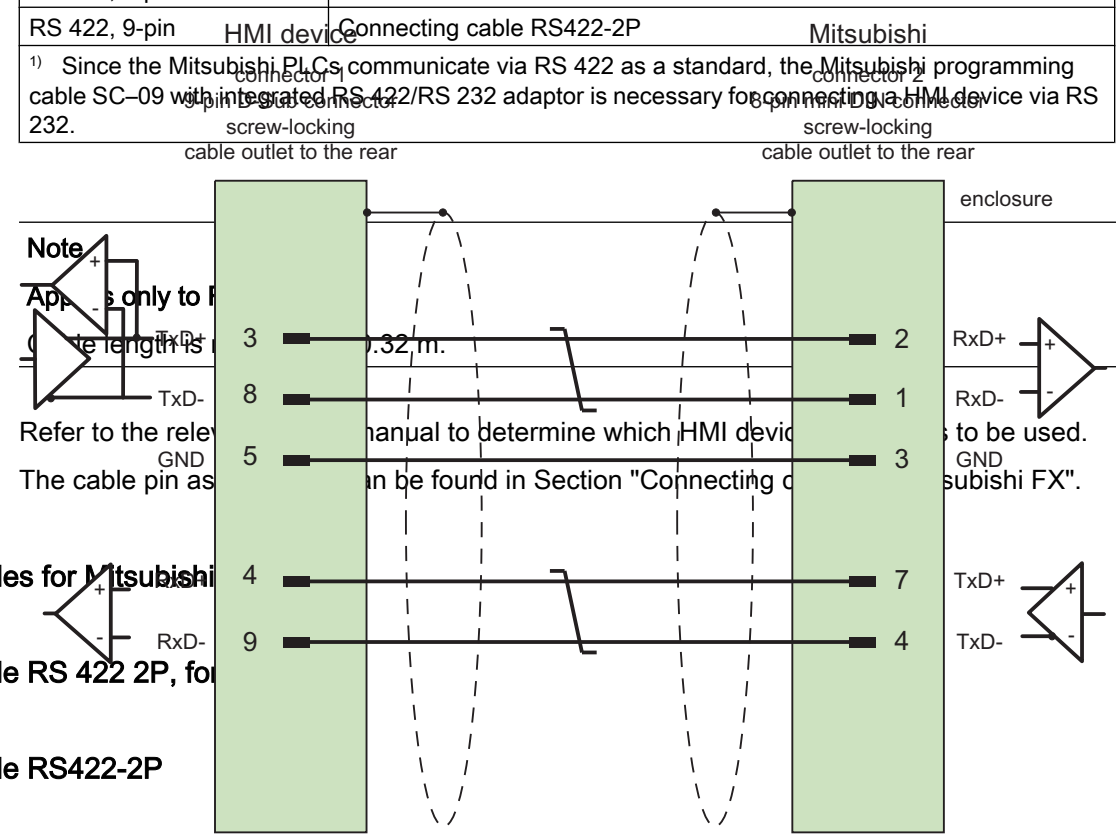
The connection between the HMI device and the Mitsubishi PLC is basically restricted to setting the interface parameters. Special blocks for the connection are not required in the PLC.

Connecting cable

The following connecting cables are available to connect the HMI device to the PLC.

Interface to HMI device or adapter	Mitsubishi Electric PLC via FX protocol
	FX1n, Fx2n, Mini DIN, 8–pin
RS 232, 9-pin	Mitsubishi SC-09 [1]
RS 422, 9-pin	Connecting cable RS422-2P

[1] Since the Mitsubishi PLCs communicate via RS 422 as a standard, the Mitsubishi programming cable SC–09 with integrated RS 422/RS 232 adaptor is necessary for connecting a HMI device via RS 232.



Note

Appl s only to H

Cable length is m .32 m.

Refer to the relev anual to determine which HMI devic s to be used.

The cable pin as n be found in Section "Connecting c subishi FX".

Connecting cables for Mitsubishi

Connecting cable RS 422 2P, for

Connecting cable RS422-2P

Shield with large-area contact to housing at both ends
Cable: 3 x 2 x 0.14 mm^2, shielded,
max. length 500 m

Performance features of communication

Permitted data types for Mitsubishi FX

Permitted data types

The table lists the data types that can be used when configuring tags and area pointers.

Data type	Operand type	Length
4-bit block	M, X, Y	1 byte
8-bit block	M, X, Y	1 byte
12-bit block	M, X, Y	2 bytes
16-bit block	M, X, Y	2 bytes
20-bit block	M, X, Y	4 bytes
24-bit block	M, X, Y	4 bytes
28-bit block	M, X, Y	4 bytes
32-bit block	M, X, Y	4 bytes
Bool	D, M, X, Y	1-bit
DWord	D, C-32 bit	4 bytes
Real	D	4 bytes
String	D	1 to 50 characters
Word	D, T, C-16 bit	2 bytes

Note

Note the following for write accesses:

For data type "Bool" in operand type "D", the entire word is written back to the PLC following a change to the specified bit. There is no check to determine whether any other bits in the word have changed. As a result, the PLC only has read access to the specified word.

Note

Array elements in I/O fields cannot be used in communication with a Mitsubishi PLC.

Supported CPU types for Mitsubishi FX

CPU types

The following CPU types are supported for configuring the Mitsubishi FX communication driver.

- FX1 series
  - FX1n
- FX2 series
  - FX2n

Address areas for Mitsubishi FX

FX3

Address areas	Data types														
	Bool	Int	Word	DInt	DWord	Real	String	4-bit block	8-bit block	12-bit block	16-bit block	20-bit block	24-bit block	28-bit block	32-bit block
M	M0 - M9999	--	--	--	--	--	--	M0 - M9996	M0 - M9992	M0 - M9988	M0 - M9984	M0 - M9980	M0 - M9976	M0 - M9972	M0 - M9968
D	D0.0 - D9999.15	D0 - D9999	D0 - D9999	D0 - D9999	D0 - D9999	D0 - D9999	D0 - D9999	--	--	--	--	--	--	--	--
T	--	--	T0 - T999	--	--	--	--	--	--	--	--	--	--	--	--
C	--	C0 - C999	C0 - C999	C0 - C998	C0 - C998	--	--	--	--	--	--	--	--	--	--
X	X0 - X777	--	--	--	--	--	--	X0 - X774	X0 - X770	X0 - X764	X0 - X760	X0 - X754	X0 - X750	X0 - X744	X0 - X740
Y	Y0 - Y777	--	--	--	--	--	--	Y0 - Y774	Y0 - Y770	Y0 - Y764	Y0 - Y760	Y0 - Y754	Y0 - Y750	Y0 - Y744	Y0 - Y740

Q

Address areas	Data types														
	Bool	Int	Word	DInt	DWord	Real	String	4-bit block	8-bit block	12-bit block	16-bit block	20-bit block	24-bit block	28-bit block	32-bit block
M	M0 - M9999	--	--	--	--	--	--	M0 - M9996	M0 - M9992	M0 - M9988	M0 - M9984	M0 - M9980	M0 - M9976	M0 - M9972	M0 - M9968
F	F0 - F9999	--	--	--	--	--	--	F0 - F9996	F0 - F9992	F0 - F9988	F0 - F9984	F0 - F9980	F0 - F9976	F0 - F9972	F0 - F9968
B	B0 - BFFFF	--	--	--	--	--	--	B0 - BFFFC	B0 - BFFF8	B0 - BFFF4	B0 - BFFF0	B0 - BFFEC	B0 - BFFE8	B0 - BFFE4	B0 - BFFE0
D	D0.0 - D65534.15	D0 - D65534	D0 - D65534	D0 - D65533	D0 - D65533	D0 - D65533	D0 - D65534	--	--	--	--	--	--	--	--
T	--	--	T0 - T2047	--	--	--	--	--	--	--	--	--	--	--	--
C		C0 - C2047	C0 - C2047	C0 - C2046	C0 - C2046	--	--	--	--	--	--	--	--	--	--
W	--	W0 - WFFFF	W0 - WFFFF	W0 - WFFFE	W0 - WFFFE	W0 - WFFFE	--	--	--	--	--	--	--	--	--
X	X0 - XFFFF	--	--	--	--	--	--	X0 - XFFFC	X0 - XFFF8	X0 - XFFF4	X0 - XFFF0	X0 - XFFEC	X0 - XFFE8	X0 - XFFE4	X0 - XFFE0
Y	Y0 - YFFFF	--	--	--	--	--	--	Y0 - YFFFC	Y0 - YFFF8	Y0 - YFFF4	Y0 - YFFF0	Y0 - YFFEC	Y0 - YFFE8	Y0 - YFFE4	Y0 - YFFE0

FX0

Address areas	Data types												
	Bool	Word	DWord	Real	String	4-bit block	8-bit block	12-bit block	16-bit block	20-bit block	24-bit block	28-bit block	32-bit block
M	M0 - M9999					M0 - M9996	M0 - M9992	M0 - M9988	M0 - M9984	M0 - M9980	M0 - M9976	M0 - M9972	M0 - M9968
D	D0.0 - D999.15	D0 - D999	D0 - D998	D0 - D998	D0 - D998								

Address areas	Data types												
	Bool	Word	DWord	Real	String	4-bit block	8-bit block	12-bit block	16-bit block	20-bit block	24-bit block	28-bit block	32-bit block
T		T0 - T255											
C-16-Bit		C-16-Bit 0 - C-16-Bit 199											
C-32-Bit			C-32-Bit 200 - C-32-Bit 255										
X	X0 - X255					X0 - X252	X0 - X248	X0 - X244	X0 - X240	X0 - X236	X0 - X232	X0 - X228	X0 - X224
Y	Y0 - X255					Y0 - Y252	Y0 - Y248	Y0 - Y244	Y0 - Y240	Y0 - Y236	Y0 - Y232	Y0 - Y228	Y0 - Y224

Commissioning components

Transferring a project to the HMI device

1. Switch the HMI device to "transfer mode".
2. Set all necessary transfer parameters.
   - Interface
   - Transfer parameters
   - Target storage location
3. Start the project transfer.
   The project is compiled automatically.
   All compilation and transfer steps are logged to a message window.

Interconnecting the PLC with the HMI device

1. Interconnect the PLC with the HMI device using a suitable cable.
2. The message "Connection to PLC .... is established" is output to the HMI device.

Optimizing the configuration

Acquisition cycle and update time

The acquisition cycles for the "Area pointers" and of the tags defined in the configuration software are decisive in terms of the update times which can actually be achieved.

The update time is equivalent to the acquisition cycle + transmission time + processing time.

Items to observe when optimizing the update times in configuration data:

- Optimize the maximum and minimum size of the data areas.
- Acquisition cycles which are too short lead to unnecessary load on overall performance. Set the acquisition cycle according to the rate of change of the process values. The rate of temperature changes at a furnace, for example, is significantly slower compared to the speed rate of an electrical drive. A time of approx. 1 second is a benchmark for the acquisition cycle.
- Avoid any gaps when entering the alarm or screen tags in a data area.
- Changes in the PLC can only be detected reliably if these are available at least within the actual acquisition cycle.

Screens

The refresh rate of screens is determined by the type and volume of data to be visualized.

Only configure short acquisition cycles for objects which actually require shorter refresh cycles. This procedure reduces update times.

Trends

The HMI device always updates all bit-triggered trends whose group bit is set in the "Trend transfer area". It resets the bits in the next cycle.

The group bit in the PLC program can only be set again after the HMI device has reset all bits.

Job mailboxes

A high rate and volume of job mailboxes transferred may lead to overload in communication between the HMI device and the PLC.

The HMI device confirms acceptance of the job mailbox by entering the value zero in the first data word of the job mailbox. The HMI device now processes the job for which it requires a certain time slice. It may take the HMI device some time to process a new job mailbox which is transferred in immediate succession to the job mailbox. The next job mailbox is only accepted if sufficient computing resources are available.

Data exchange

Area pointers for Mitsubishi

Area pointers for connections via Mitsubishi communication drivers

You use an area pointer to access a data area in the PLC.

For more detailed information on area pointers and their configuration, refer to Section "Data exchange using area pointers".

Special considerations for connections via Mitsubishi communication drivers

You can configure the following area pointers

Area pointers	Mitsubishi MC TCP/IP	Mitsubishi FX
Screen number	Yes	Yes
Date/time	Yes	Yes
Date/time PLC	Yes	Yes
Coordination	Yes	Yes
Project ID	Yes	Yes
Job mailbox	Yes	Yes
Data record	Yes	Yes

Restrictions Mitsubishi MC TCP/IP

The following restrictions apply for configuring area pointers.

CPU type	Data types	Operand type
FX3	Int, Word	D
Q	Int, Word	D

Mitsubishi FX restrictions

You cannot use the D operand type for configuring area pointers.

See also

Data exchange using area pointers (Page 8745)

Trends

Trends

A trend is the graphical representation of one or more values from the PLC. The value is read out time-triggered for Basic Panels.

For additional information see:

Configuring trend displays for values from the PLC (Page 8147)

Time-triggered trends

The HMI device reads in the trend values cyclically at an interval specified in the configuration.

Time-triggered trends are suitable for continuous curves, such as the operating temperature of a motor.

Alarms

Configuring alarms

Configuring alarms for non-integrated connections

Several steps are necessary to configure alarms such as warnings, error messages and acknowledgement.

- Step 1: Create tags
- Step 2: Configure alarms
- Step 3: Configure acknowledgment

You can find additional information in the section:

Working with alarms (Page 8157)

Distinctive features when configuring alarms

If you are configuring connections of HMI devices to PLCs of other manufacturers, note the following distinctive features when configuring:

- Data types of the tags
- Addressing of tags
- How the bit positions are counted

Data types

For connections with a Mitsubishi communication driver, the following data types are supported:

PLC	Permitted data types	
	Discrete alarms	Analog alarms
FX1n, FX2n, FX3 series, Q-Series, iQ-Series	Word, Int [1]	4-bit block, 8-bit block, 12-bit block, 16-bit block, 20-bit block, 24-bit block, 28-bit block, 32-bit block, Word, DWord, Int [1], DInt [1], Real,
[1] Not for Mitsubishi FX communication driver		

How the bit positions are counted

For connections with a Mitsubishi communication driver, the following counting method applies:

How the bit positions are counted	Left byte								Right byte							
In Mitsubishi PLCs	15							8	7							0
In WinCC you configure:	15							8	7							0

Restrictions on alarms

- Mitsubishi MC TCP/IP
  Only tags of operand type "D" and data types "Word" and "Int" are permitted as trigger tags for discrete alarms. You can use array tags (operand type: "D"; data types: "ARRAY [x..y] of Word" or "ARRAY [x..y] of Int") for discrete alarms.
- Mitsubishi FX
  Only tags of operand type "D" and data type "Word" are permitted as trigger tags for discrete alarms. You can use array tags (operand type "D"; data type "ARRAY [x..y] of Word") for discrete alarms."

Acknowledgment of alarms

Procedure

Create suitable tags on the PLC to acknowledge an error alarm. You assign these tags to an alarm in the "Bit messages" editor. You make the assignment in "Properties > Acknowledgment".

Distinction in terms of acknowledgment:

- Acknowledgment by the PLC
- Acknowledgment on the HMI device

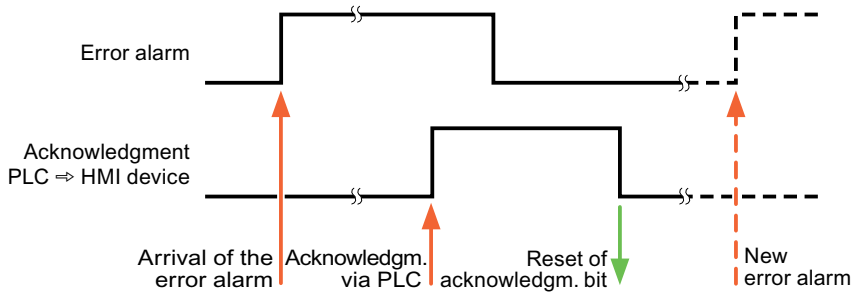
Acknowledgment by the PLC

In "PLC acknowledgment tag", configure the tag or array tag and the bit number that the HMI device uses to identify a PLC acknowledgment.

A bit set in the tag triggers acknowledgment of the assigned error alarm bit at the HMI device. This tag bit returns a function similar to acknowledgment on the HMI device which is triggered by pressing the "ACK" button, for example.

The acknowledgment bit must be located in the same tag as the bit for the error alarm.

Reset the acknowledgment bit before setting the bit in the alarm area again. The figure below shows the pulse diagram.



Acknowledgment on the HMI device

In the "HMI acknowledgment tag" area, configure the tag or array tag as well as the bit number that the HMI device writes to the PLC after acknowledgment. Make sure when you use an array tag that it is not longer than 6 words.

To always create a signal change when setting an assigned acknowledgment bit of a discrete alarm that must be acknowledged, the HMI device will reset the acknowledgment bit assigned to the alarm as soon as it detects an alarm subject to acknowledgment and write the acknowledgment tag in the PLC. There will be a certain delay between detecting the message and writing the acknowledgment tag in the PLC because the HMI device has to process the operations.

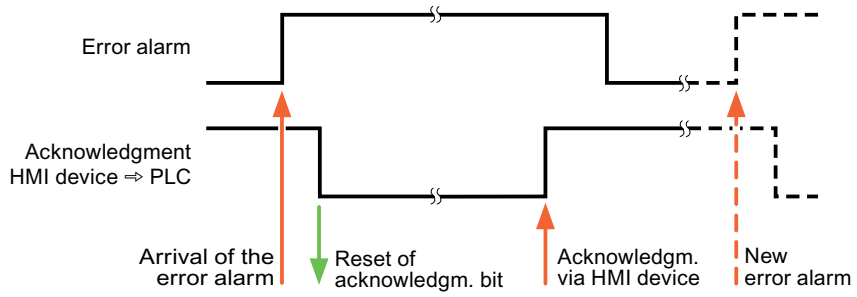
If a discrete alarm subject to acknowledgment is acknowledged by the HMI device, then the corresponding bit in the assigned acknowledgment tag will be set. The entire acknowledgment tag is then written to the PLC by the HMI device. This allows the PLC to recognize that a certain alarm message has been acknowledged at the HMI device.

Note

All alarm bits acknowledged since the last Runtime start will remain in the acknowledgment tag until a new incoming of the respective discrete alarms is detected.

This area should only be read by the PLC because the entire section of the HMI device will be overwritten once the next acknowledgment tag is written.

The figure below shows the pulse diagram.



Modicon Modbus

Modicon Modbus communication drivers

Introduction

This section describes the communication between an HMI device and PLCs that use Modicon Modbus communication drivers.

The following communication drivers are supported:

- Modicon Modbus TCP/IP
- Modicon Modbus RTU

Data exchange

Data is exchanged by means of tags or area pointers.

- Tags
  The PLC and the HMI device use process values for data exchange. You create tags in the configuration that point to addresses in the PLC. The HMI device reads the value from the defined address, and then displays it. The operator may also enter values on the HMI device, which are then written to the address in the PLC.
- Area pointers
  Area pointers are used to exchange specific data and are only set up when these data are used.

Modicon Modbus TCP/IP

Configuring a connection via Modicon Modbus TCP/IP

Introduction

You configure a connection to one of the PLCs with Modicon Modbus TCP/IP communication driver in the "Connections" editor of the HMI device.

The Ethernet interfaces are named differently depending on the HMI device.

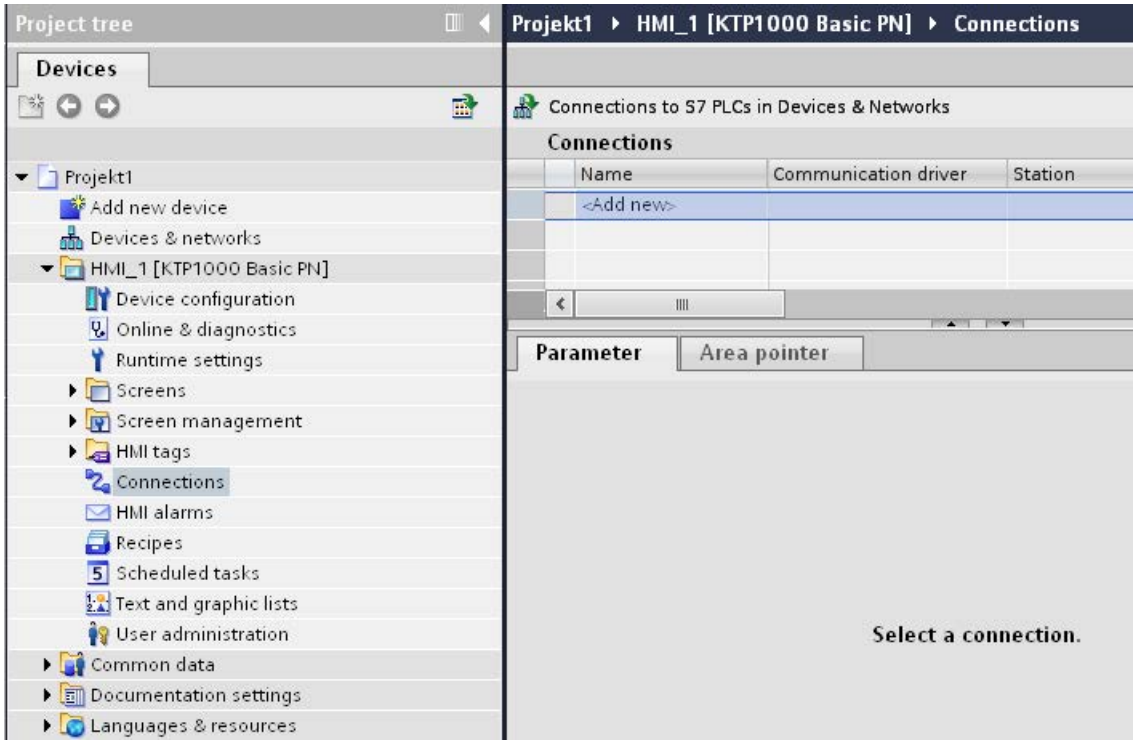
Example: PROFINET interface corresponds to the Ethernet interface

Requirements

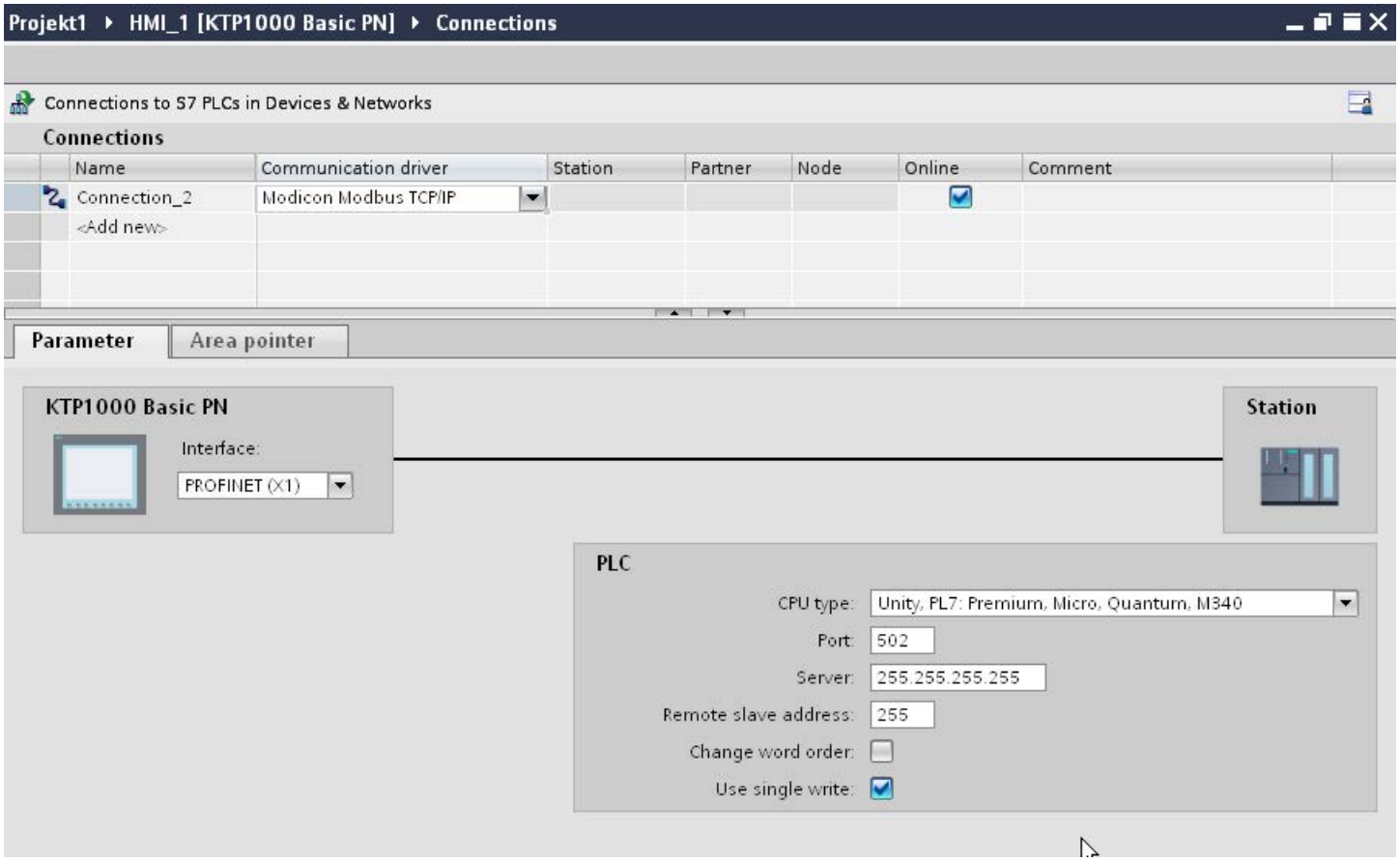
- A project is open.
- An HMI device has been created.

Procedure

1. Double-click the HMI device under "Devices" in the project tree.
2. Double-click the "Connections" item.
3. Double-click "<Add>" in the "Connections" editor.



4. Select the "Modicon Modbus TCP" driver in the "Communication driver" column.



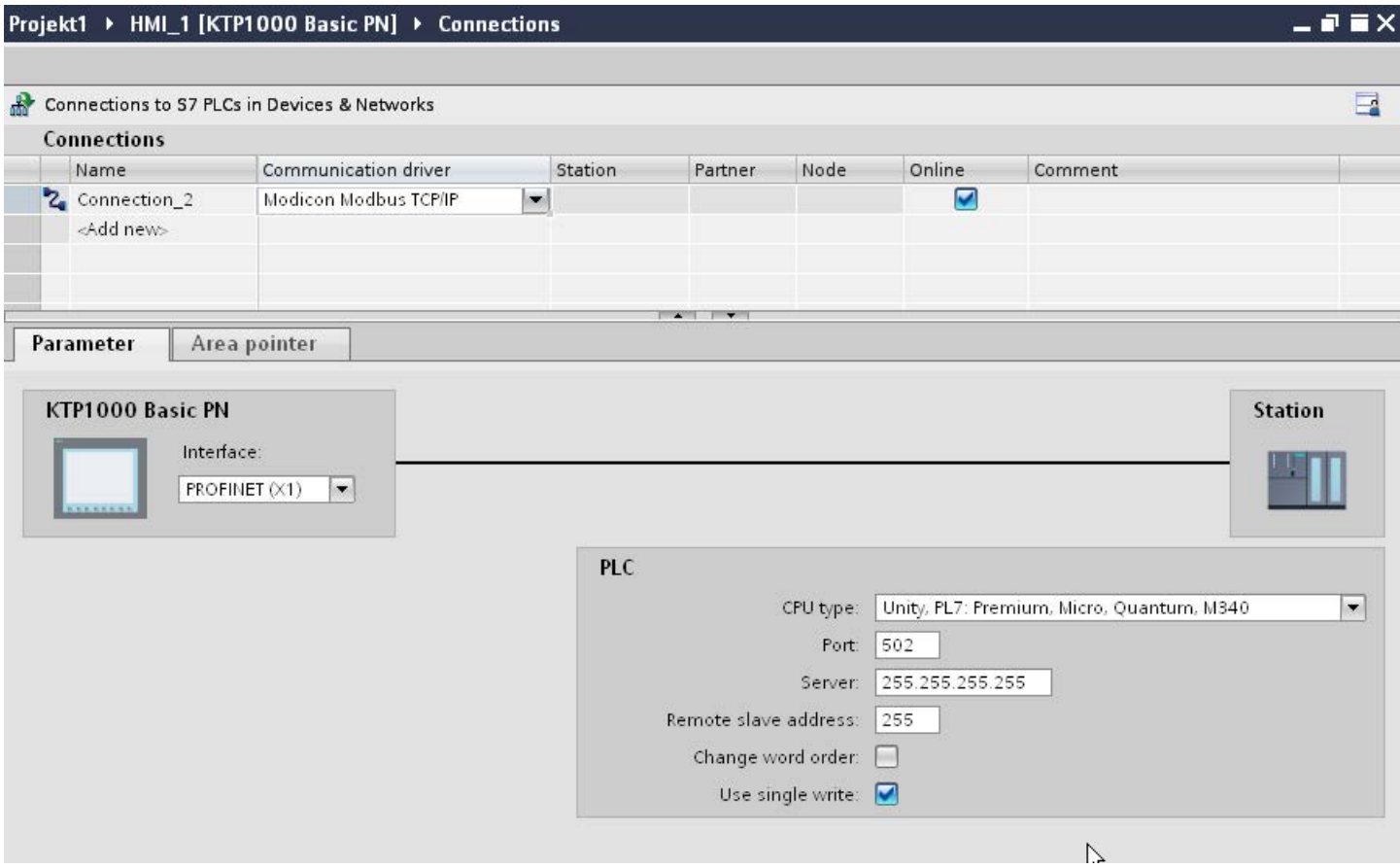
5. Select all necessary connection parameters for the interface in the Inspector window under "Parameters".

Parameters for the connection (Modicon Modbus TCP/IP)

Parameters to be set

To set the connection parameters, such as addresses and profiles, click the connection that you have created in the "Connections" editor.

The communication partners are displayed schematically in the Inspector window under "Parameters". The "HMI device" and "PLC" areas are available for assigning parameters according to the interface used.



Parameters for the HMI device

You can select only one interface for the HMI device in the Inspector window under "Parameters". Depending on the HMI device, there are several interfaces available.

If you are directly connected to the HMI device during configuration, you can set up the IP address of the HMI device in WinCC. The IP address is transferred to the HMI device during project transfer.

Note

The IP address in the control panel will be overwritten upon subsequent loading if you have already set up the IP address in the HMI device control panel.

The IP address already set up in the control panel will be retained upon subsequent loading if you activate "Set IP address using a different method".

To set up the IP address of the HMI device:

1. Click the HMI device.
2. Open the "Device configuration" editor.

3. Click the Ethernet interface.
4. Assign the IP address in the inspector window under:
   "General > PROFINET interface > Ethernet addresses"

Parameters for the PLC

- "CPU type"
  For "CPU type", you set the Modicon PLC to which the HMI device is connected.
- "Port"
  For "Port", you set the port that is used for the TCP/IP connection. The port used by the Modicon PLCs is 502.
- "Server"
  You set the IP address or host name of the PLC under "Server". Only the IP address can be used on a Basic Panel.
- "Distributed slave address"
  For "Distributed slave address", set the slave address of the remote PLC, but only if a bridge is used.
  If a bridge is not used, the default value "255" (or "0") must be retained.
- "Change word order"
  The "Change word order " parameter only affects the word order of the 32-bit values display. The setting pertains to the data types Double, Double+/-, and Float. The byte order cannot be changed.
  - "Change word order" not activated
    The most significant byte is sent first.
    For double words, the least significant word is sent before the most significant word.
    This setting has been system-tested for all approved PLCs.
  - "Change word order" activated
    The most significant byte is sent first.
    For double words, the most significant word is sent before the least significant word.

    Note

    This setting must be used for the SIEMENS SENTRON PAC3200 and PAC4200 multi-function meters and can be used for PLCs of other manufacturers.
- "Use single write"
  If you deselect this function, only function codes 15H and 16H are used for writing into the PLC.
  If this function remains selected, the function codes 05H, 06H, 15H, and 16H are used.

Connecting HMI device to PLC

Connections via Modicon Modbus TCP/IP

Connection

The HMI device can be connected to the Modicon Modbus PLC using the following components:

- Existing Ethernet network that also contains the PLCs
- Cross-over Ethernet cable connected directly to the Ethernet interface of the CPU or the communication module

The connection of the HMI device to a Modicon Modbus PLC is limited primarily to the physical connection of the HMI device. Special blocks for the connection are not required in the PLC.

Note

Timeout response with TCP/IP (Ethernet)

Due to the use of the TCP/IP protocol, the breakdown of a connection is detected at the earliest after approximately one minute. Communication failure cannot be reliably detected if no tags are requested, for example, no output tags in the current screen.

Configure area pointer coordination for each PLC. This setting ensures that a communication failure is recognized after approximately two minutes, even in the aforementioned scenario.

Communication types

Approved communication types

The following communication types are system-tested and approved:

- Point-to-point coupling:
- Multiple point coupling of a HMI device (Modbus TCP/IP Client) with up to 4 PLCs, each with different couplings. CPU types can be mixed.
  The following couplings are possible:
  - Coupling the Ethernet CPU interface of the TSX Unity Quantum.
  - Coupling via the communication modules for Ethernet 140 NOE 771 01 for the TSX Quantum and TSX Unity Quantum series
  - Coupling via the Ethernet interface of the 171 CCC 980 30 CPU adapter of the Momentum series
  - Coupling the Ethernet CPU interface of the TSX Unity Premium.
  - Coupling via the Ethernet TCP/IP connect module TSX ETY 110 for the TSX Premium and TSX Unity Premium series
  - Coupling via the Ethernet TCP/IP connect module TSX ETY 410 for the Micro series
  - Coupling via the Ethernet TCP/IP Modbus Plus Bridge 174 CEV 200 40 to the Modbus Plus interface of the Compact, the TSX Quantum and the TSX Unity Quantum series

Via the TCP/IP Modbus Plus Bridge, 174 CEV 200 40, the PLCs can be accessed at their Remote Slave Address via the Ethernet interface of this bridge.

Note

Integration of the HMI device in a Modbus network via a bridge is not possible. The HMI device is the Modbus master.

Restrictions

The coupling of the HMI device to PLCs of other manufacturers who offer a Modbus TCP/IP interface is not system-tested and thus, not enabled.

However, if another PLC is to be used, observe the following instructions:

- Use the following CPU types, because these operate without address offset and in the usual bit count manner.
  - Unity, PL7: Premium, Micro, Quantum, M340
- The following function codes are used for the respective data areas:

Reading function codes		Address range	
01	ReadCoilStatus	0x / %M	DIGITAL_OUT
02	ReadInputStatus	1x / %I	DIGITAL_IN
03	ReadHoldingRegisters	4x / %MW	USERDATA

Reading function codes		Address range	
04	ReadInputRegisters	3x / %IW	ANALOG_IN
20 (14Hex)	ReadGeneralReference	6x / –	EXTENDEDMEMORY (not for all CPUs)

Writing function codes		Address range	
06 [1]	PresetSingleRegister	4x / %MW	USERDATA Single
16 (10Hex)	PresetMultipleRegisters	4x / %MW	USERDATA Multiple
05 [1]	ForceSingleCoil	0x / %M	DIGITAL_OUT with BIT
15 (0FHex)	ForceMultipleCoils	0x / %M	DIGITAL_OUT with 16 BIT GROUP
21 (15Hex)	WriteGeneralReference	6x / –	EXTENDEDMEMORY (not for all CPUs)

[1] Select use with "Use single write".

Connectable PLCs

Connections can be implemented for the following Modicon Modbus PLCs:

Modicon Modbus PLC	Supported protocol	
	Modicon Modbus RTU [2]	Modicon Modbus TCP/IP
TSX Compact	x	x [1]
TSX Quantum	x	x
Momentum	x	x
Premium	-	x
Micro	-	x
M340 20x0 (without 2010)	-	x

[1] Only via Ethernet TCP/IP Modbus Plus Bridge

[2] Communication via RS 232 is tested and enabled for the PLC. In the HMIs that only have a RS 422/485 interface, the RS 422/232 converter with the order number 6AV6 671-8XE00-0AX0 was tested and enabled.

Performance features of communication

Permissible data types for Modicon Modbus TCP/IP

Permitted data types

The table lists the data types that can be used when configuring tags and area pointers.

Note

If you change the Modicon Modbus RTU communication driver to Modicon Modbus TCP/IP, the string in the "String" data type may be different.

Permitted data types for CPU type "Unity, PLC: Premium, Micro, Quantum M340"

Data type	Operand type	Length
+/- Double	%MW	4 bytes
+/- Int	%MW, %IW	2 bytes
16-bit group	%MW, %I	2 bytes
ASCII	%MW	0 to 80 characters
Bit	%MW, %IW, %M, %I	1-bit
Double	%MW	4 bytes
Float	%MW	4 bytes
Int	%MW, %IW	2 bytes

Note

The ranges "%I" and "%IW" are not supported for the following CPU types:

- Premium
- Micro
- M340

Permitted data types for CPU type "Concept, ProWORX: Compact, Quantum, Momentum"

Data type	Operand type	Length
+/- Double	4x, 6x	4 bytes
+/- Int	3x, 4x, 6x	2 bytes
16-bit group	0x, 1x	2 bytes
ASCII	4x, 6x	0 to 80 characters
Bit	0x, 1x, 3x, 4x, 6x	1-bit

Data type	Operand type	Length
Double	4x, 6x	4 bytes
Float	4x, 6x	4 bytes
Int	3x, 4x, 6x	2 bytes

Bit counting method

The usual bit counting method "16 LSB - 1 MSB" in the following CPU types is only used in the "HMI tags" editor with the selected "Bit" data type:

- Concept, ProWORX: Compact, Quantum, Momentum

The following bit location assignment applies:

	Left byte								Right byte							
Counting with tags	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Format for "Signed"

The placeholder "+/-" stands for the data types "Signed Int" and "Signed Double".

Supported CPU types for Modicon Modbus TCP/IP

CPU types

The following CPU types are supported for configuring the Modicon Modbus TCP/IP communication driver.

- Compact
- Momentum
- Quantum
  - Concept Quantum
  - Unity Quantum
- Micro
- Premium
- Modicon M340
  - 20x0 (except 2010)

Address areas for Modicon Modbus TCP/IP

UnityPl7

Address areas	Data types							
	Bool	16 Bit Group	Int	+/- Int	DInt	+/- DInt	Float	ASCII
%1	%I0 - %I65535	%I65535 %I0 - %I65520	--	--	--	--	--	--
%M	%M0 - %M65535	%M65535 %M0 - %M65520	--	--	--	--	--	--
%IW	%IW0.0 - %IW65535.15	--	%IW0 - %IW65535	%IW0 - %IW65535	--	--	--	--
%MW	%MW0.0 - %MW65535.15	--	%MW0 - %MW65535	%MW0 - %MW65535	%MW0 - %MW65534	%MW0 - %MW65534	%MW0 - %MW65534	%MW0 - %MW65535

ConceptProWORX

Address areas	Data types							
	Bool	16 Bit Group	Int	+/- Int	DInt	+/- DInt	Float	ASCII
0x	0x1 - 0x65535	0x1 - 0x65520	--	--	--	--	--	--
1x	1x100001 - 1x165535	1x100001 - 1x165520	--	--	--	--	--	--
3x	3x300001.1 - 3x365535.16	--	3x300001 - 3x365535	3x300001 - 3x365535	--	--	--	--
4x	4x400001.1 - 4x465535.16	--	4x400001 - 4x465535	4x400001 - 4x465535	4x400001 - 4x465534	4x400001 - 4x465534	4x400001 - 4x465534	4x400001 - 4x465535
6x	6x60000.1:1 - 6x69999.16:10	--	6x60000:1 - 6x69999:10	6x60000:1 - 6x69999:10	6x60000:1 - 6x69998:10	6x60000:1 - 6x69998:10	6x60000:1 - 6x69998:10	6x60000:1 - 6x69999:10

Commissioning components

Transferring a project to the HMI device

1. Switch the HMI device to "transfer mode".
2. Set all necessary transfer parameters.
   - Interface
   - Transfer parameters
   - Target storage location
3. Start the project transfer.
   The project is compiled automatically.
   All compilation and transfer steps are logged to a message window.

Interconnecting the PLC with the HMI device

1. Interconnect the PLC with the HMI device using a suitable cable.
2. The message "Connection to PLC .... is established" is output to the HMI device.

Optimizing the configuration

Acquisition cycle and update time

The acquisition cycles for the "Area pointers" and of the tags defined in the configuration software are decisive in terms of the update times which can actually be achieved.

The update time is equivalent to the acquisition cycle + transmission time + processing time.

Items to observe when optimizing the update times in configuration data:

- Optimize the maximum and minimum size of the data areas.
- Acquisition cycles which are too short lead to unnecessary load on overall performance. Set the acquisition cycle according to the rate of change of the process values. The rate of temperature changes at a furnace, for example, is significantly slower compared to the speed rate of an electrical drive. A time of approx. 1 second is a benchmark for the acquisition cycle.
- Avoid any gaps when entering the alarm or screen tags in a data area.
- Changes in the PLC can only be detected reliably if these are available at least within the actual acquisition cycle.

Screens

The refresh rate of screens is determined by the type and volume of data to be visualized.

Only configure short acquisition cycles for objects which actually require shorter refresh cycles. This procedure reduces update times.

Trends

The HMI device always updates all bit-triggered trends whose group bit is set in the "Trend transfer area". It resets the bits in the next cycle.

The group bit in the PLC program can only be set again after the HMI device has reset all bits.

Job mailboxes

A high rate and volume of job mailboxes transferred may lead to overload in communication between the HMI device and the PLC.

The HMI device confirms acceptance of the job mailbox by entering the value zero in the first data word of the job mailbox. The HMI device now processes the job for which it requires a certain time slice. It may take the HMI device some time to process a new job mailbox which is transferred in immediate succession to the job mailbox. The next job mailbox is only accepted if sufficient computing resources are available.

Modicon Modbus RTU

Configuring a connection via Modicon Modbus RTU

Introduction

You configure a connection to a PLC with a Modicon Modbus RTU communication driver in the "Connections" editor of the HMI device.

The interfaces are named differently depending on the HMI device.

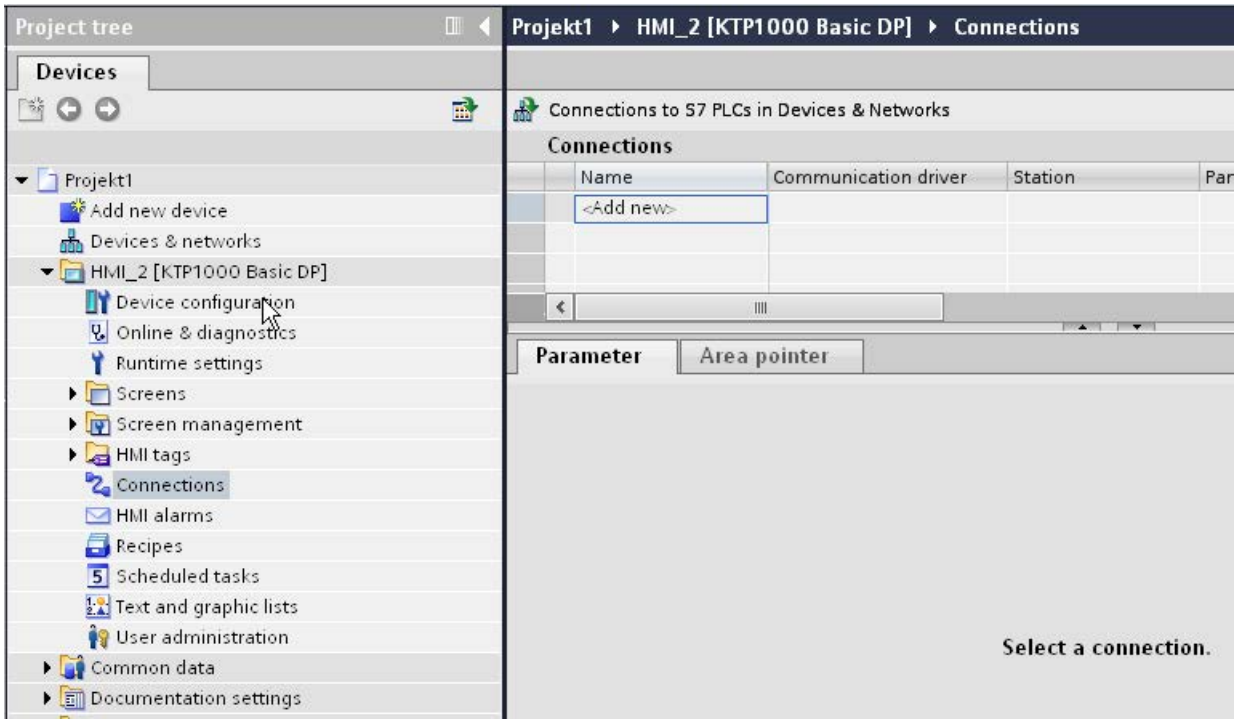
Requirements

- A project is open.
- An HMI device has been created.

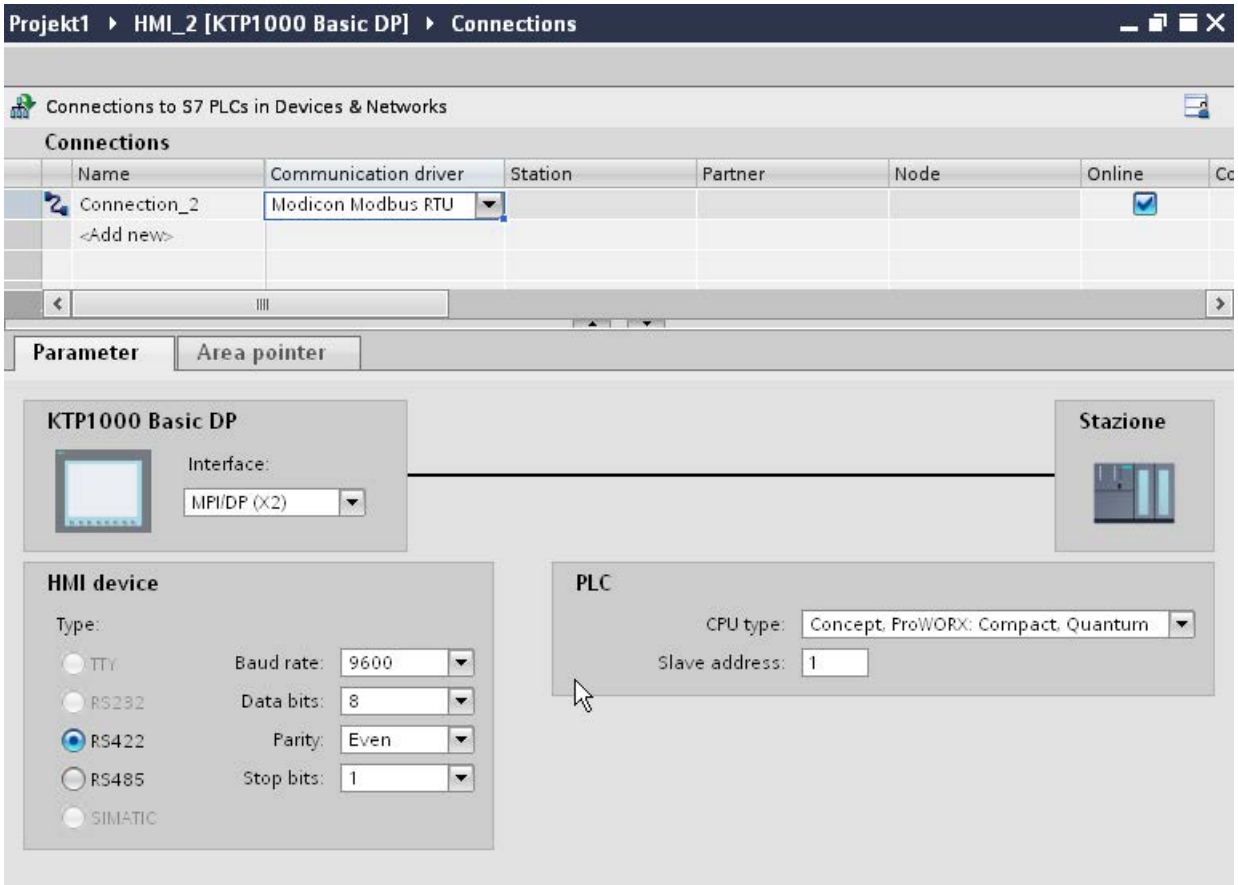
Procedure

1. Double-click the HMI device under "Devices" in the project tree.
2. Double-click the "Connections" item.

3. Double-click "<Add>" in the "Connections" editor.



4. In the "Communication drivers" column, select the "Modicon Modbus RTU" driver.



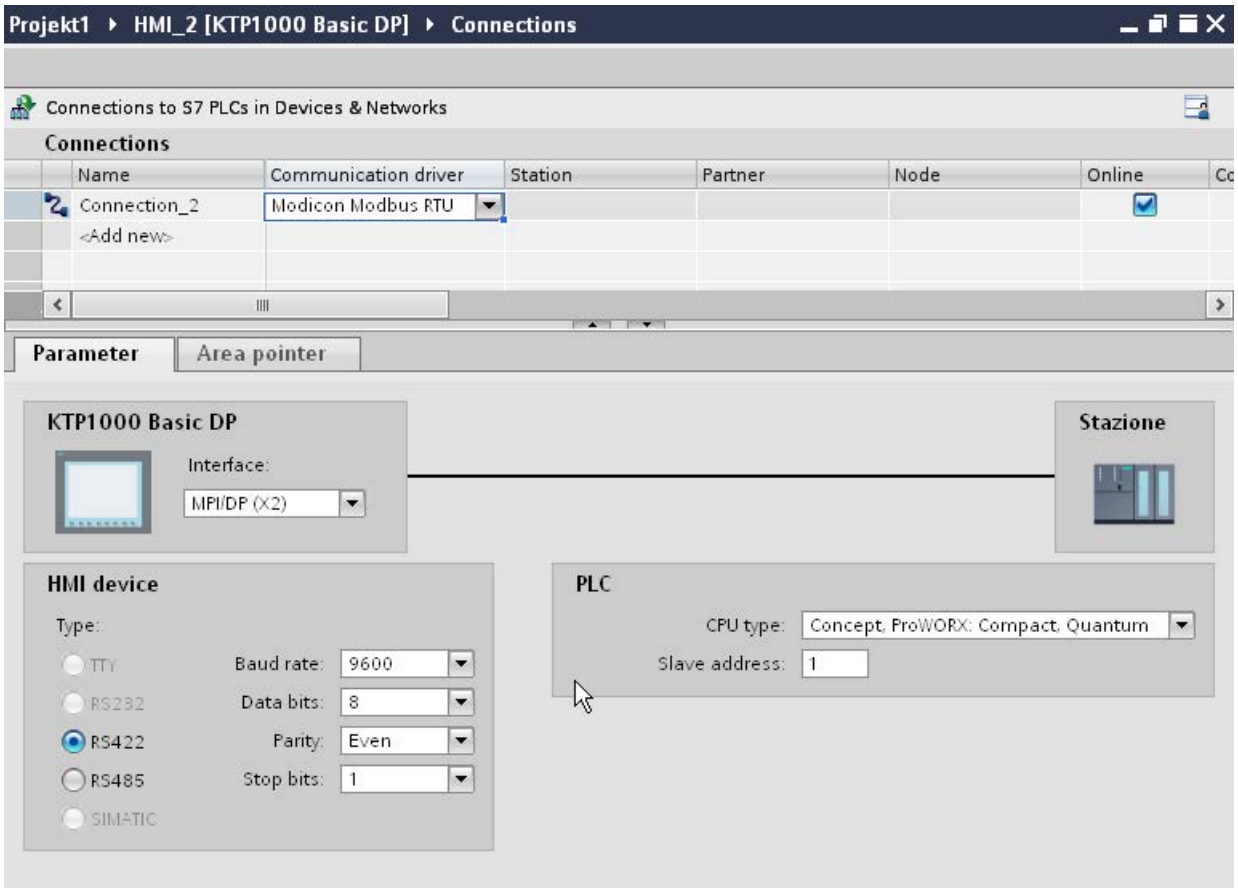
5. Select all necessary connection parameters for the interface in the Inspector window under "Parameters".

Parameters for the connection (Modicon Modbus RTU)

Parameters to be set

To set the connection parameters, such as addresses and profiles, click the connection that you have created in the "Connections" editor.

The communication partners are displayed schematically in the Inspector window under "Parameters". The "HMI device" and "PLC" areas are available for assigning parameters according to the interface used.



Parameters for the HMI device

You can select an interface for the HMI device in the Inspector window under "Parameters". Depending on the HMI device, there are several interfaces available.

- Type
  Only RS 232 is system-tested.
  No warranty is given for RS 485.

  Note

  RS 422 is only approved in combination with the RS 422-RS 232 converter.

  Order number: 6AV6 671-8XE00-0AX0

  Note

  If you use the IF1B interface, you must switch over the RS422 receive data additionally by 4 DIL-switches on the back of the HMI device.

- Baud rate
  For "Baud rate", select the transmission speed between the HMI device and Modicon PLC.
  A baud rate of 19200 or 9600 can be selected for the communication.
  A baud rate of 4800 can be selected for certain HMI devices.

- Data bits
  For "Data bits", only the value "8" can be selected.
- Parity
  For "Parity", you can choose from "None", "Even", and "Odd".
- Stop bits
  For "Stop bits", you can choose between 1 and 2 bits.

Parameters for the PLC

- CPU type
  For "CPU type", you set the Modicon PLC to which the HMI device is connected.
  You can select the following CPUs:
  – Concept, ProWORX: Compact, Quantum
- Slave address
  Under "Slave address" you set which slave address the CPU has.

Connecting HMI device to PLC

Connections via Modicon Modbus RTU

Connection

Connect the HMI device to the Modicon Modbus RTU interface of the Modicon Modbus RTU slave.

The connection of the HMI device to Modicon is limited primarily to the physical connection of the HMI device. Special blocks for the connection are not required in the PLC.

Connection cable

The following connecting cables are available to connect the HMI device to Modicon Modbus.

Interface to the HMI device	Modicon PLC		
	directly via Modbus interface (RS232) 9-pin Sub D male connector	Via MB Bridge (RS 232)	directly via Modbus interface (RS232) 8-pin RJ45 connector
RS 232, 9-pin	PP1	PP1	PP2

The cable pin assignments can be found in Section "Connecting cables for Modicon Modbus RTU".

Communication types

Approved communication types

The following communication types are system-tested and approved:

- Ppoint-to-point connection only via the RS-232 interface.
- Multipoint connection from a HMI device (Modbus-Master) with up to 4 PLCs: The HMI device must be connected with a Modbus Plus Bridge or a Compact, Momentum CPU or TSX Quantum CPU which is configured as a Modbus Plus Bridge.
- You connect the other PLCs via the Modbus Plus connection on the first PLC. The PLCs can be reached under their address via the bridge functionality of the first PLC.

  Note

  It is not possible to integrate the HMI device into a Modbus network because the HMI device is Modbus-Master.

- Integration of the HMI device into a Modbus Plus network via the "bridge mode" of the Compact, Momentum or Quantum (logical point-to-point communication of the HMI device with a Compact, Momentum or Quantum).

Restrictions

The connection of the HMI device to PLCs of other manufacturers which offer a Modicon Modbus interface is not system-tested and therefore not approved.

If you use another PLC nevertheless, observe the following information:

- These drivers only work for tags with the bit counting method typical for Modicon PLCs from left (bit1 = most significant bit) to right (bit16 = least significant bit in data type INT).
- The address offset displayed in the configuring is subtracted at protocol level in the message frame. E.g. in Holding Register 4x the offset "40001". The configured address "40006" therefore becomes address "5" in the message frame. The address (e.g. "5") transferred in the message frame is transformed to the PLC-specific address range in the different Non-Modicon PLCs.
- A reply message frame without "ExceptionCode" is expected within 500 ms.
- The following function codes are used for the respective data areas:

Reading function codes		Address range	
01	ReadCoilStatus	0x	DIGITAL_OUT
02	ReadInputStatus	1x	DIGITAL_IN
03	ReadHoldingRegisters	4x	USERDATA
04	ReadInputRegisters	3x	ANALOG_IN
20 (14Hex)	ReadGeneralReference	6x	EXTENDEDMEMORY (not for all CPUs)

Writing function codes		Address range	
06	PresetSingleRegister	4x	USERDATA Single
16 (10Hex)	PresetMultipleRegisters	4x	USERDATA Multiple
05	ForceSingleCoil	0x	DIGITAL_OUT with data type Bit
15 (0FHex)	ForceMultipleCoils	0x	DIGITAL_OUT with data type 16 bit group
21 (15Hex)	WriteGeneralReference	6x	EXTENDEDMEMORY (not for all CPUs)

Connectable PLCs

Connections can be implemented for the following Modicon Modbus PLCs:

Modicon Modbus PLC	Supported protocol	
Modicon Modbus RTU [2]	Modicon Modbus TCP/IP	
TSX Compact	x	x [1]
TSX Quantum	x	x
Momentum	x	x
Premium	-	x
Micro	-	x
M340 20x0 (without 2010)	-	x

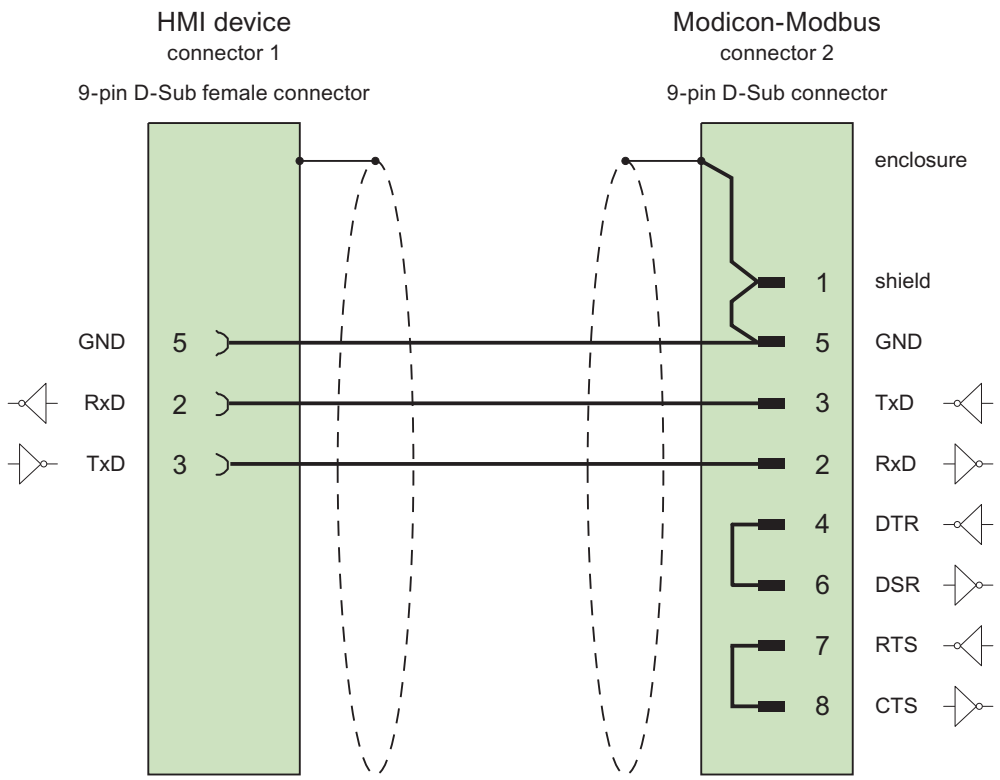
[1] Only via Ethernet TCP/IP-Modbus Plus Bridge

[2] Communication via RS 232 is tested and enabled for the PLC. In the HMI devices which only have an RS 422/485 interface, the RS 422/232 converter with the order number 6AV6 671-8XE00-0AX0 was tested and approved.

Connecting cables for Modicon Modbus RTU

Connecting cable PP1, RS-232, for Modicon

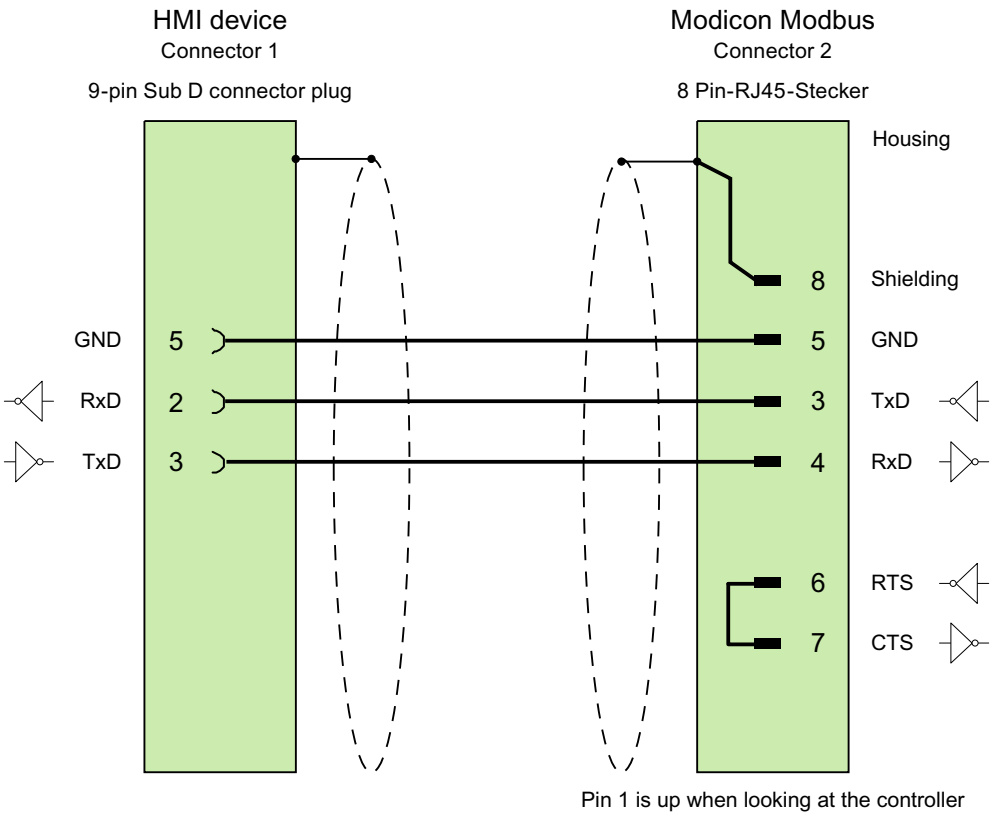
Point-to-point cable 1: PLC > PC ...



Cables: 3 x 0.14 mm^2, shielded,
max. length 15 m

Connecting cable PP2, RS-232, for Modicon

Point-to-point cable 2: PLC (TSX Compact) > PC...



Cables: 3 x 0.14 mm^2, shielded,
max. length 15 m

Performance features of communication

Permitted data types for Modicon Modbus RTU

Permitted data types

The table lists the data types that can be used when configuring tags and area pointers.

Data type	Operand type	Length
+/- Double	4x, 6x	4 bytes
+/- Int	3x, 4x, 6x	2 bytes
16-bit group	0x, 1x	2 bytes
ASCII	4x, 6x	0 to 80 characters
Bit [1)]	0x, 1x, 3x, 4x, 6x	1-bit
Double	4x, 6x	4 bytes
Float	4x, 6x	4 bytes
Int	3x, 4x, 6x	2 bytes

1) Note the following for write accesses:
For data type "Bit" with the operand types "4x" and "6x", the entire word is written back to the PLC following a change to the specified bit. There is no check to determine whether any other bits in the word have changed. As a result, the PLC only has read access to the specified word.

The usual bit counting method (16 LSB - 1 MSB) in the following CPU types is only used in the "HMI tags" editor with the selected "Bit" data type:

- Concept ProWORX: Compact, Quantum

The following bit location assignment applies:

	Left byte								Right byte							
Counting with tags	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Format for "Signed"

The placeholder "+/-" stands for the data types "Signed Int" and "Signed Double".

Supported CPU types for Modicon Modbus RTU

CPU types

The following CPU types are supported in the configuration of the Modicon Modbus RTU communication driver.

- Compact
- Momentum
- Quantum

Address areas for Modicon Modbus RTU

UnityPl7

Address areas	Data types							
	Bool	16 Bit Group	Int	+/- Int	DInt	+/- DInt	Float	ASCII
%1	%I0 - %I65535	%I65535 %I0 - %I65520	--	--	--	--	--	--
%M	%M0 - %M65535	%M65535 %M0 - %M65520	--	--	--	--	--	--
%IW	%IW0.0 - %IW65535.15	--	%IW0 - %IW65535	%IW0 - %IW65535	--	--	--	--
%MW	%MW0.0 - %MW65535.15	--	%MW0 - %MW65535	%MW0 - %MW65535	%MW0 - %MW65534	%MW0 - %MW65534	%MW0 - %MW65534	%MW0 - %MW65535

ConceptProWORX

Address areas	Data types							
	Bool	16 Bit Group	Int	+/- Int	DInt	+/- DInt	Float	ASCII
0x	0x1 - 0x65535	0x1 - 0x65520	--	--	--	--	--	--
1x	1x100001 - 1x165535	1x100001 - 1x165520	--	--	--	--	--	--
3x	3x300001.1 - 3x365535.16	--	3x300001 - 3x365535	3x300001 - 3x365535	--	--	--	--

Address areas	Data types							
	Bool	16 Bit Group	Int	+/- Int	DInt	+/- DInt	Float	ASCII
4x	4x400001.1 - 4x465535.16	--	4x400001 - 4x465535	4x400001 - 4x465535	4x400001 - 4x465534	4x400001 - 4x465534	4x400001 - 4x465534	4x400001 - 4x465535
6x	6x60000.1:1 - 6x69999.16:10	--	6x60000:1 - 6x69999:10	6x60000:1 - 6x69999:10	6x60000:1 - 6x69998:10	6x60000:1 - 6x69998:10	6x60000:1 - 6x69998:10	6x60000:1 - 6x69999:10

Commissioning components

Transferring a project to the HMI device

1. Switch the HMI device to "transfer mode".
2. Set all necessary transfer parameters.
   - Interface
   - Transfer parameters
   - Target storage location
3. Start the project transfer.
   The project is compiled automatically.
   All compilation and transfer steps are logged to a message window.

Interconnecting the PLC with the HMI device

1. Interconnect the PLC with the HMI device using a suitable cable.
2. The message "Connection to PLC .... is established" is output to the HMI device.

Optimizing the configuration

Acquisition cycle and update time

The acquisition cycles for the "Area pointers" and of the tags defined in the configuration software are decisive in terms of the update times which can actually be achieved.

The update time is equivalent to the acquisition cycle + transmission time + processing time.

Items to observe when optimizing the update times in configuration data:

- Optimize the maximum and minimum size of the data areas.
- Acquisition cycles which are too short lead to unnecessary load on overall performance. Set the acquisition cycle according to the rate of change of the process values. The rate of temperature changes at a furnace, for example, is significantly slower compared to the speed rate of an electrical drive. A time of approx. 1 second is a benchmark for the acquisition cycle.
- Avoid any gaps when entering the alarm or screen tags in a data area.
- Changes in the PLC can only be detected reliably if these are available at least within the actual acquisition cycle.

Screens

The refresh rate of screens is determined by the type and volume of data to be visualized.

Only configure short acquisition cycles for objects which actually require shorter refresh cycles. This procedure reduces update times.

Trends

The HMI device always updates all bit-triggered trends whose group bit is set in the "Trend transfer area". It resets the bits in the next cycle.

The group bit in the PLC program can only be set again after the HMI device has reset all bits.

Job mailboxes

A high rate and volume of job mailboxes transferred may lead to overload in communication between the HMI device and the PLC.

The HMI device confirms acceptance of the job mailbox by entering the value zero in the first data word of the job mailbox. The HMI device now processes the job for which it requires a certain time slice. It may take the HMI device some time to process a new job mailbox which is transferred in immediate succession to the job mailbox. The next job mailbox is only accepted if sufficient computing resources are available.

Data exchange

Area pointers for Modicon Modbus

Area pointers for connections via Modicon Modbus communication drivers

You use an area pointer to access a data area in the PLC.

For more detailed information on area pointers and their configuration, refer to Section "Data exchange using area pointers (Page 8745)".

Special considerations for connections via Modicon communication drivers

You can configure the following area pointers

Area pointers	Modicon Modbus TCP/IP	Modicon Modbus RTU
Screen number	Yes	Yes
Date/time	Yes	Yes
Date/time PLC	Yes	Yes
Coordination	Yes	Yes
Project ID	Yes	Yes
Job mailbox	Yes	Yes
Data record	Yes	Yes

Restrictions Modicon Modbus TCP/IP

The following restrictions apply for configuring area pointers.

CPU type	Data types	File types
Concept, ProWORX: Compact, Quantum, Momentum	+/- Int, Int	4x, 6x
Unity, PL7: Premium, Micro, Quantum, M340	+/- Int, Int	%MW

Modicon Modbus RTU restrictions

The following restrictions apply for configuring area pointers.

CPU type	Data types	File types
Concept, ProWORX: Compact, Quantum, Momentum	+/- Int, Int	4x, 6x

Trends

Trends

A trend is the graphical representation of one or more values from the PLC. The value is read out time-triggered for Basic Panels.

For additional information see:

Configuring trend displays for values from the PLC (Page 8147)

Time-triggered trends

The HMI device reads in the trend values cyclically at an interval specified in the configuration.

Time-triggered trends are suitable for continuous curves, such as the operating temperature of a motor.

Alarms

Configuring alarms

Configuring alarms for non-integrated connections

Several steps are necessary to configure alarms such as warnings, error messages and acknowledgement.

- Step 1: Create tags
- Step 2: Configure alarms
- Step 3: Configure acknowledgment

You can find additional information in the section:

Working with alarms (Page 8157)

Distinctive features when configuring alarms

If you are configuring connections of HMI devices to PLCs of other manufacturers, note the following distinctive features when configuring:

- Data types of the tags
- Addressing of tags
- How the bit positions are counted

Data types

For connections with a Modicon Modbus communication driver, the following data types are supported:

PLC	Permitted data types	
	Discrete alarms	Analog alarms
All Modicon series	Int, +/-Int	16 Bit Group, Int, +/-Int, Double, +/-Double, Float

Arrays and array tags cannot be used for discrete alarms.

How the bit positions are counted

For connections with a Modicon Modbus communication driver, the following counting method applies:

How the bit positions are counted	Left byte								Right byte							
In WinCC you configure:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Acknowledgment of alarms

Procedure

Create suitable tags on the PLC to acknowledge an error alarm. You assign these tags to an alarm in the "Bit messages" editor. You make the assignment in "Properties > Acknowledgment".

Distinction in terms of acknowledgment:

- Acknowledgment by the PLC
- Acknowledgment on the HMI device

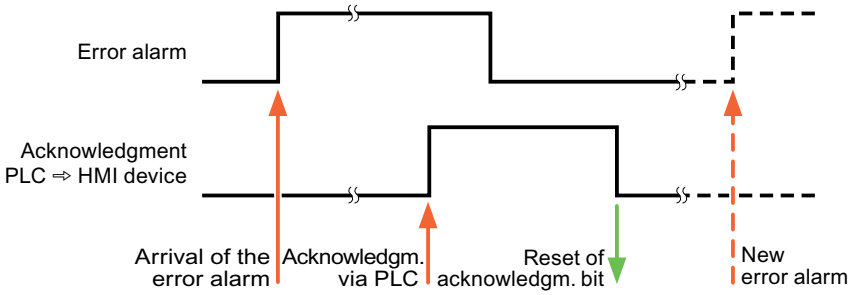
Acknowledgment by the PLC

In "PLC acknowledgment tag", configure the tag or array tag and the bit number that the HMI device uses to identify a PLC acknowledgment.

A bit set in the tag triggers acknowledgment of the assigned error alarm bit at the HMI device. This tag bit returns a function similar to acknowledgment on the HMI device which is triggered by pressing the "ACK" button, for example.

The acknowledgment bit must be located in the same tag as the bit for the error alarm.

Reset the acknowledgment bit before setting the bit in the alarm area again. The figure below shows the pulse diagram.



Acknowledgment on the HMI device

In the "HMI acknowledgment tag" area, configure the tag or array tag as well as the bit number that the HMI device writes to the PLC after acknowledgment. Make sure when you use an array tag that it is not longer than 6 words.

To always create a signal change when setting an assigned acknowledgment bit of a discrete alarm that must be acknowledged, the HMI device will reset the acknowledgment bit assigned to the alarm as soon as it detects an alarm subject to acknowledgment and write the acknowledgment tag in the PLC. There will be a certain delay between detecting the message and writing the acknowledgment tag in the PLC because the HMI device has to process the operations.

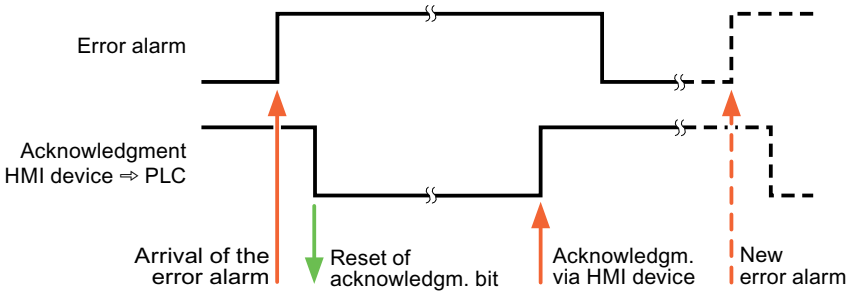
If a discrete alarm subject to acknowledgment is acknowledged by the HMI device, then the corresponding bit in the assigned acknowledgment tag will be set. The entire acknowledgment tag is then written to the PLC by the HMI device. This allows the PLC to recognize that a certain alarm message has been acknowledged at the HMI device.

Note

All alarm bits acknowledged since the last Runtime start will remain in the acknowledgment tag until a new incoming of the respective discrete alarms is detected.

This area should only be read by the PLC because the entire section of the HMI device will be overwritten once the next acknowledgment tag is written.

The figure below shows the pulse diagram.



Omron

Omron communication drivers

Introduction

This section describes the communication between an HMI device and PLCs that use Omron communication drivers.

The following communication drivers are supported:

- Omron Host Link

Data exchange

Data is exchanged by means of tags or area pointers.

- Tags
  The PLC and the HMI device use process values for data exchange. You create tags in the configuration that point to addresses in the PLC. The HMI device reads the value from the defined address, and then displays it. The operator may also enter values on the HMI device, which are then written to the address in the PLC.
- Area pointers
  Area pointers are used to exchange specific data and are only set up when these data are used.

Omron Host Link

Configuring a connection via Omron Host Link

Introduction

You configure a connection to a PLC with an Omron Host Link communication driver in the "Connections" editor of the HMI device.

Note

Connection with Omron Host Link

A connection will not automatically be established when runtime is started if you have configured a connection via Omron.

A tag which is in the valid PLC memory area must be configured in the runtime start screen.

The connection will otherwise only be established once a corresponding screen has been selected.

This tag will be accessed when runtime is started and a connection will then be established.

The interfaces are named differently depending on the HMI device.

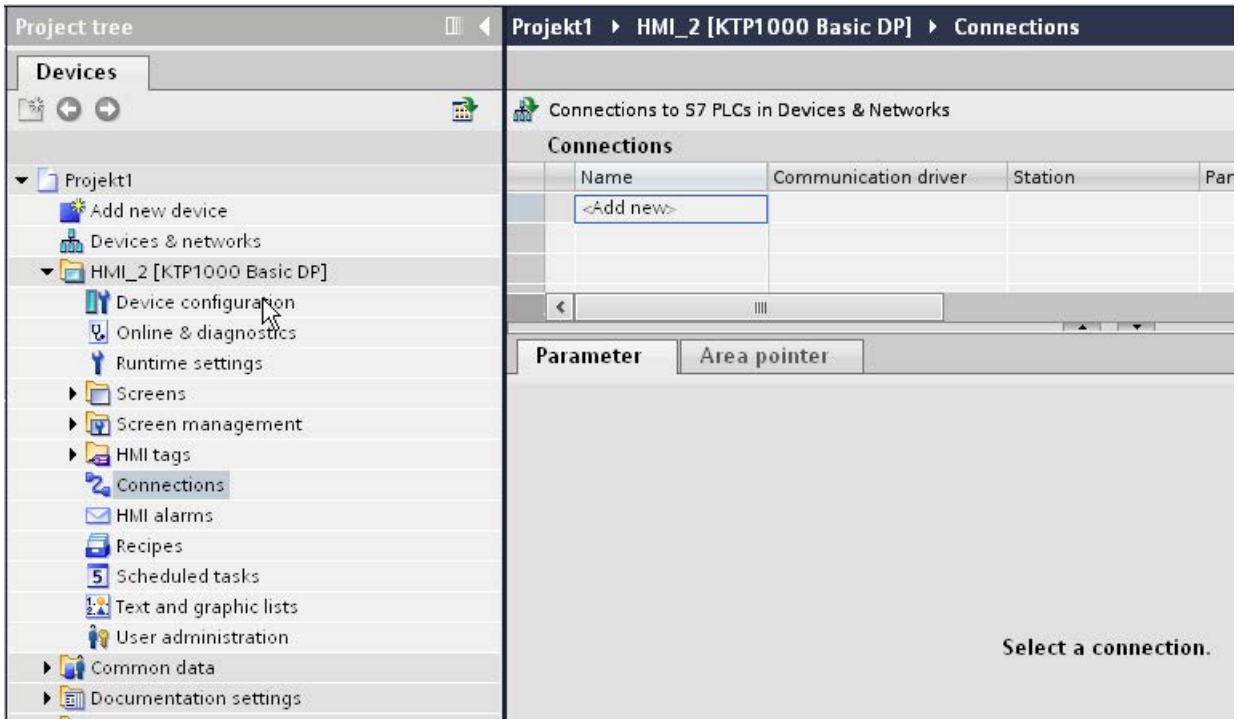
Requirements

- A project is open.
- An HMI device has been created.

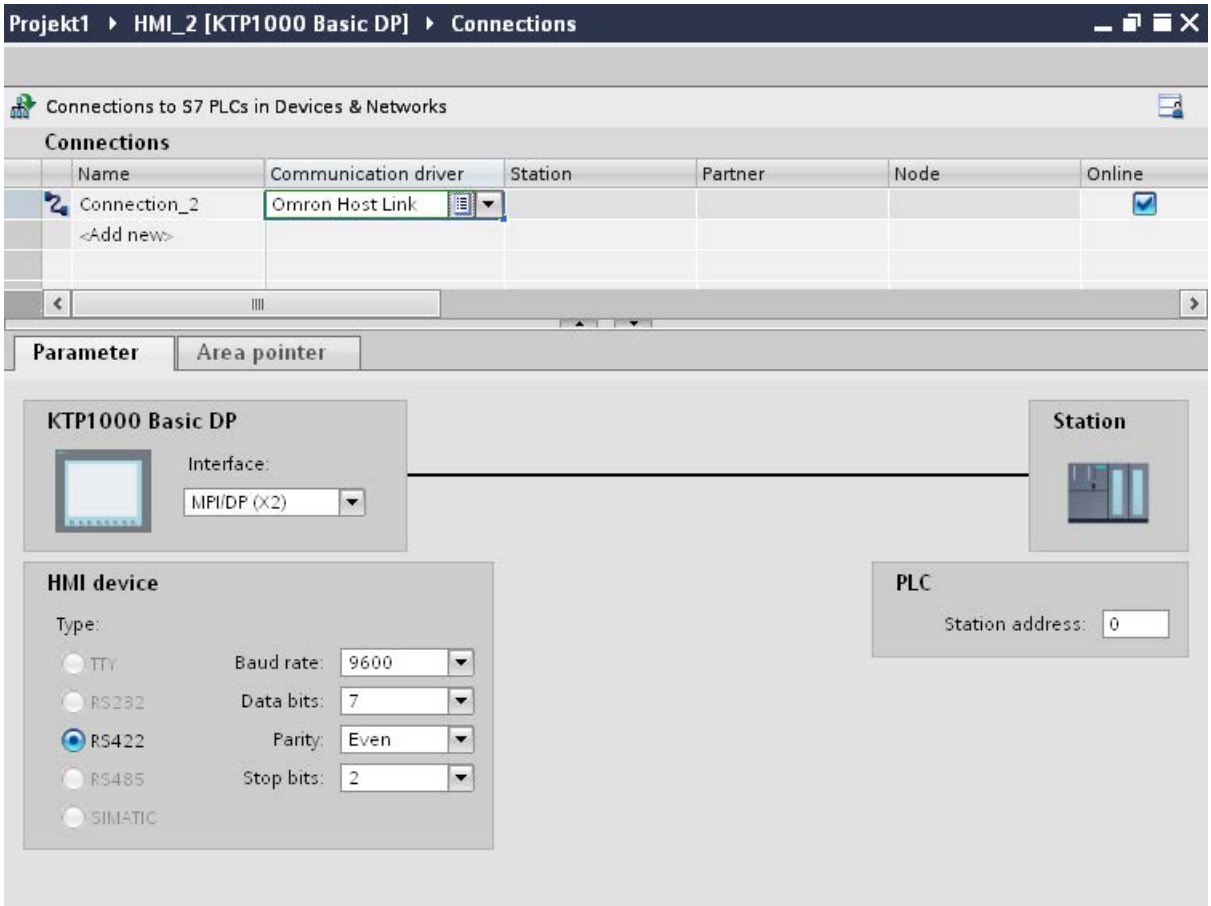
Procedure

1. Double-click the HMI device under "Devices" in the project tree.
2. Double-click the "Connections" item.

3. Double-click "<Add>" in the "Connections" editor.



4. Select the "Omron Host Link" driver in the "Communication driver" column.



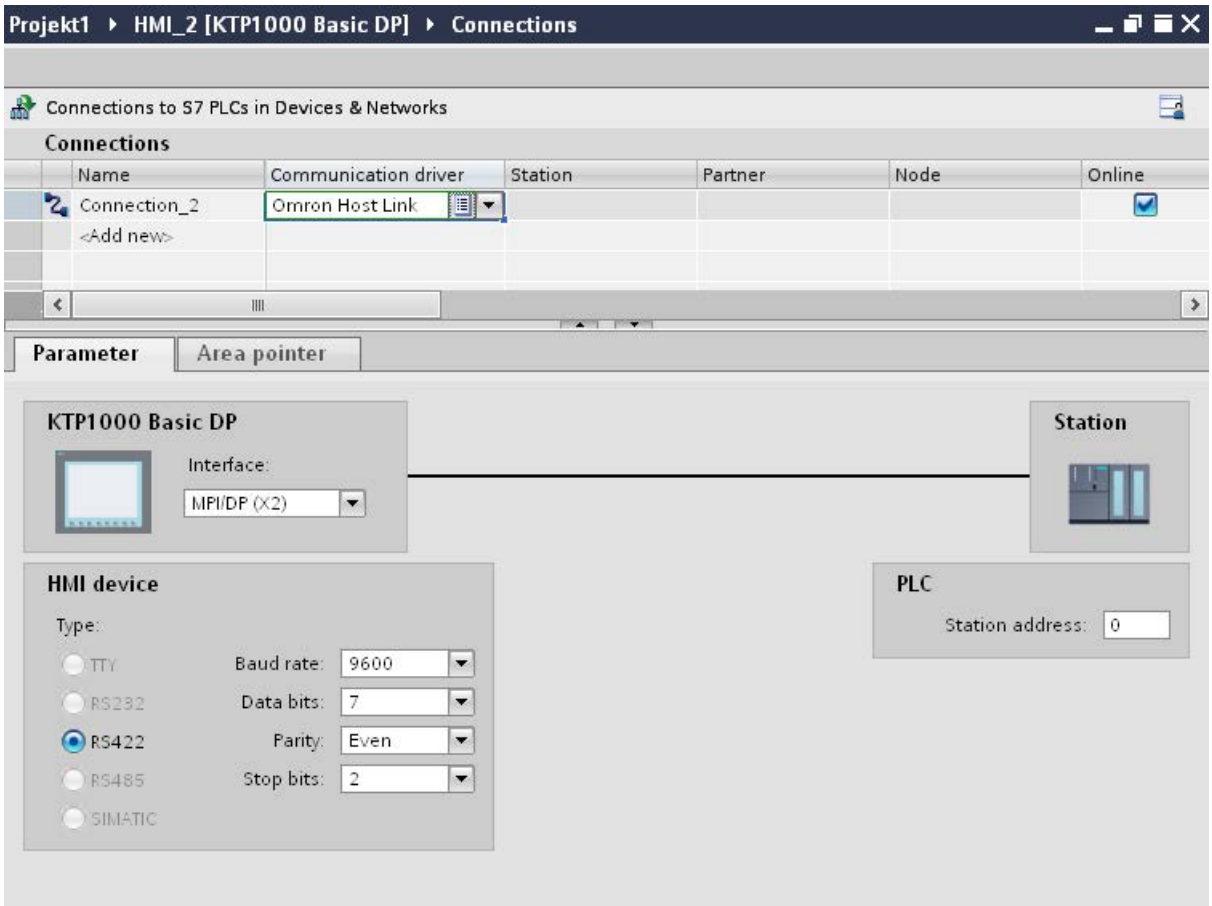
5. Select all necessary connection parameters for the interface in the inspector window under "Parameters".

Parameters for the connection (Omron Hostlink)

Parameters to be set

To set the connection parameters, such as addresses and profiles, click the connection that you have created in the "Connections" editor.

The communication partners are displayed schematically in the Inspector window under "Parameters". The "HMI device" and "PLC" areas are available for assigning parameters according to the interface used.



Parameters for the HMI device

You can select an interface for the HMI device in the Inspector window under "Parameters". Depending on the HMI device, there are several interfaces available.

- Type
  Specifies the physical connection used.
- Baud rate
  For "Baud rate", you set the transmission speed of the HMI device to OMRON. A baud rate of 19200 or 9600 can be selected for the communication.
- Data bits
  For "Data bits", you can choose between "7 bits" and "8 bits".
- Parity
  For "Parity", you can choose from "None", "Even", and "Odd".
- Stop bits
  For "Stop bits", you can choose between 1 and 2 bits.

Parameters for the PLC

- Station address
  For "Station address", set the station number of the connected PLC.

Connecting HMI device to PLC

Connections via Omron Host Link

Connection

The connection of the HMI device to an OMRON PLC is limited primarily to the physical connection of the HMI device. Special blocks for the connection are not required in the PLC.

Connection cable

The following connecting cables are available to connect the HMI device to an Omron PLC.

Interface to the HMI device	Omron PLC			
	RS232, 9-pin	RS232 I/O port	RS422, 9-pin	RS422, terminals/ pins
RS232, 9-pin	PP1	Programming cable (standard cable of Omron)	—	—
RS232 via converter	—	—	—	Multi-point cable 1
RS422, 9-pin	—	—	PP2	Multi-point cable 2

Refer to the relevant device manual to determine which HMI device interface is to be used.

Communication types

Approved communication types

The connection from a HMI device to an OMRON-CPU with the Omron Host Link protocol via RS232 and via RS 422 is system-tested and approved by Siemens AG.

This concerns the following CPU types:

- CP1x (CP1L, CP1H, CP1E)
- CJ1x (CJ1M, CJ1H, CJ1G)
- CJ2H

- CS1x (CS1G, CS1H, CS1D)
- CPM2C

Note

Only the following CPU types have been tested and released for Basic Panels, TP 177A and OP 77A:

- CP1x (CP1L, CP1H, CP1E)
- CJ1x (CJ1M, CJ1H, CJ1G)

Multipoint connection

A multipoint connection to the up to 4 approved OMRON PLCs in a RS422-four-wire connection can be implemented with communication modules on the PLCs and is system-tested and approved by Siemens AG.

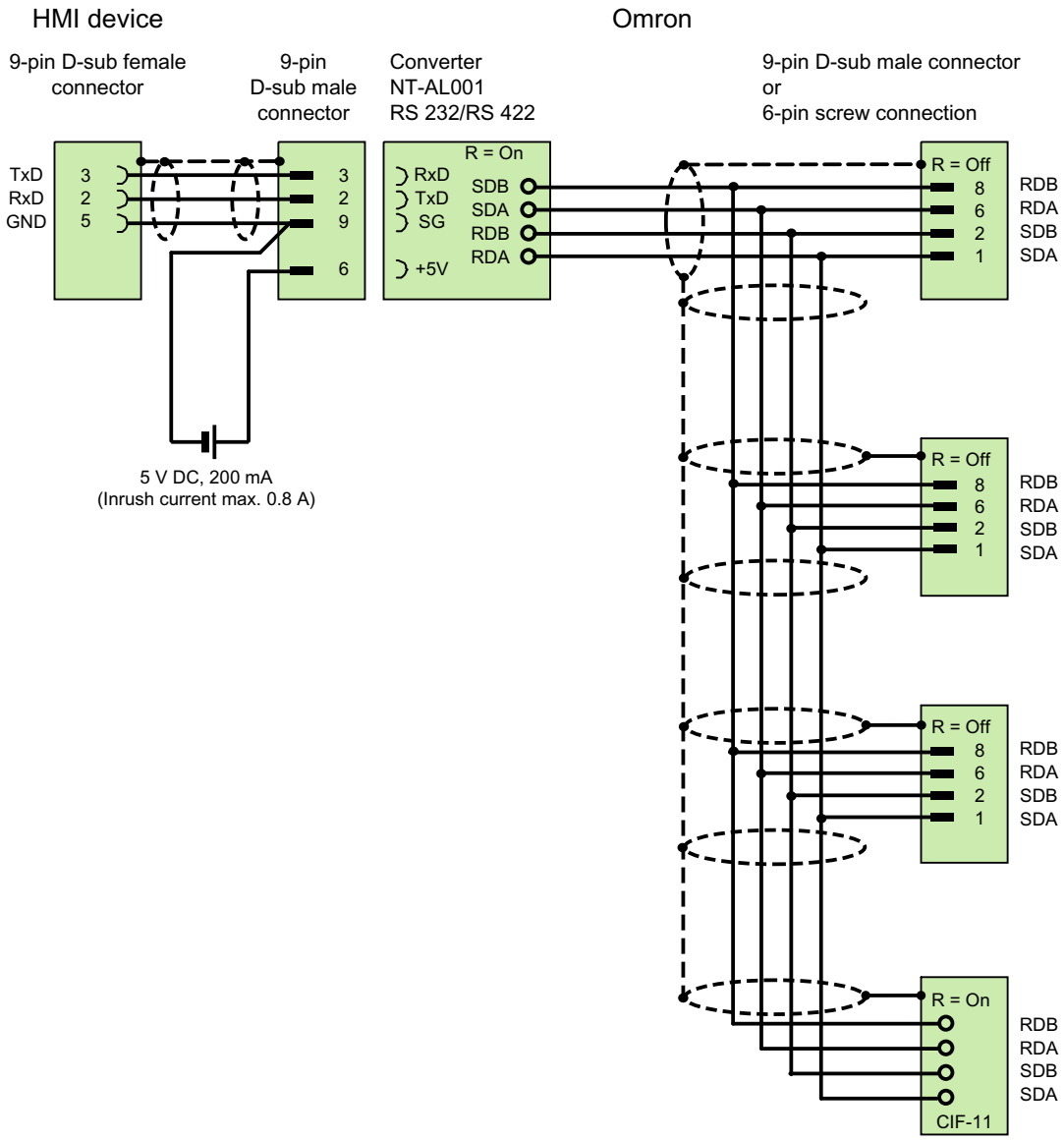
Note

The HMI device can only be operated as a master. Exactly one master is possible in the RS422-four-wire-Multidrop connection.

Connecting cable

Connecting cable MP1, RS-232, over converter, for Omron

Multipoint cable 1: MP/TP/PC > PLC

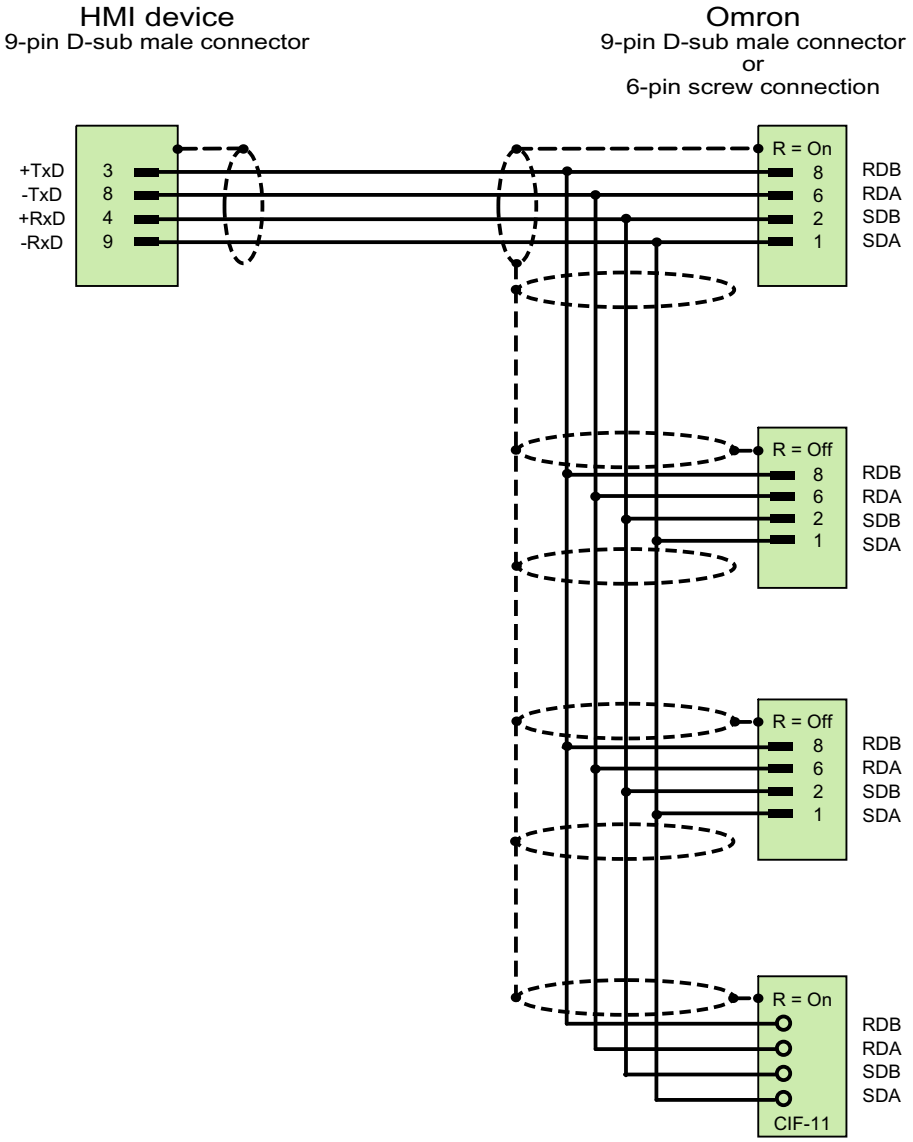


[1] Inrush current max. 0.8 A

shielded, max. length 500 m

Connecting cable MP2, RS-422, for Omron

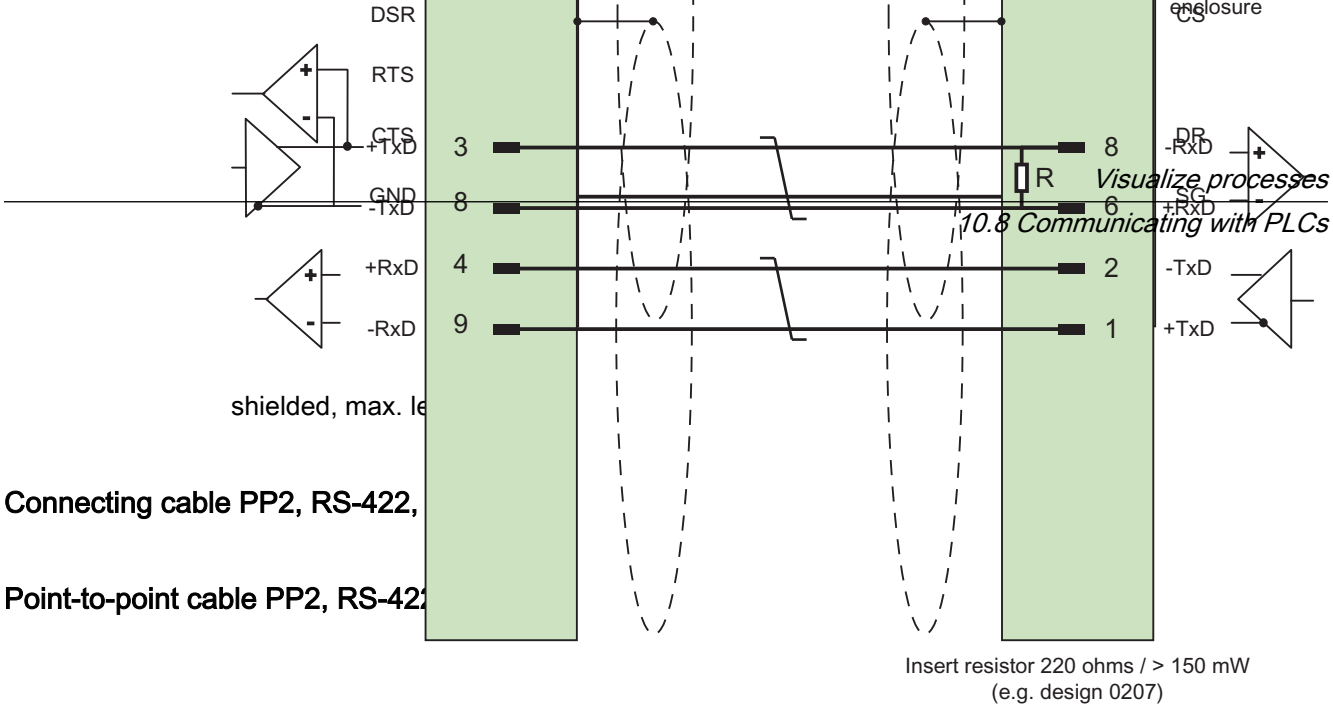
Multipoint cable 2: RS422, MP/TP/PC > SPS_



shielded, max. length 500 m

Connecting cable PP1, RS-232, for Omron

Point-to-point cable PP1, PC/TP/OP - PLC



shielded, max. le

Connecting cable PP2, RS-422,

Point-to-point cable PP2, RS-42

shielded, max. length 500 m

Performance features of communication

Permissible data types for Omron Host Link

Permitted data types

The table lists the data types that can be used when configuring tags and area pointers.

Data type	Operand type	Length
Bool	I/O, HR, AR, LR, DM, T/C bit, CPU status	1-bit
Byte	CPU type	1 byte
DInt	HR, AR, LR, DM	4 bytes
Int	I/O, HR, AR, LR, DM, T/C Val	2 bytes
Real	HR, DM	4 bytes
String	HR, AR, LR, DM	0 to 80 characters
UDInt	HR, AR, LR, DM	4 bytes
UInt	I/O, HR, AR, LR, DM, T/C Val	2 bytes

Note

Read and write operations of all data areas in the OMRON PLC can only be reliably carried out in "STOP" or "MONITOR" mode.

"I/O" refers either to the IR/SR area or the CIO area depending on the PLC series. The operand types "LR", "HR" and "AR" are not available in all PLC series.

Note

Note the following for write accesses:

For the "Bool" data type with the operand types "I/O", "HR", "AR", "LR" and "DM", the entire word is written back into the PLC when the specified bit is changed. There is no check to determine whether any other bits in the word have changed. As a result, the PLC only has read access to the specified word.

Operand type old PLC	Operand type CS and CJ PLC
CPU Status	CPU Status
I/O	CIO
HR	H Range 0-511
AR	A
LR	n/a 1)
DM	D
T/C	T/C
CPU type	CPU type

1) You do not get an error message when you read or write the LR area in the following PLCs
- CS
- CJ
- CP

Supported CPU types for Omron Host Link

CPU types

The following CPU types are supported in the configuration of the Omron Host Link communication driver.

- CP1
  - CP1L
  - CP1H
  - CP1E
- CJ1
  - CJ1M
  - CJ1H
  - CJ1G

- CJ2
  - CJ2H
- CS1
  - CS1G
  - CS1H
  - CS1D
- CPM
  - CPM2C

Addressing in Omron Host Link

Addressing of PLCs in Omron Host Link

In PLCs of the series CS, CP and CJ, the timers 0-4095 are addressed with T/C 0-2047.

The counters 0-4095 must be addressed with an offset of 2048 (T/C 2048-4095 correspond to the counters 0-2047). Counters and timers with addresses > 2047 cannot be addressed via Host Link.

Counters and timers with addresses > 2047 cannot be addressed via Host Link.

Example:

If you want to address counter C20, you must address T/C 20+2048 = T/C 2068.

Address areas for Omron Host Link

Omron

Address areas	Data types							
	Bool	Byte	UInt	Int	UDInt	DInt	Real	String
I/O	I/O 0.0 - I/O 9999.15	--	I/O 0 - I/O 9999	I/O 0 - I/O 9999	--	--	--	--
HR	HR 0.0 - HR 9999.15		HR 0 - HR 9999	HR 0 - HR 9999	HR 0 - HR 9998	HR 0 - HR 9998	HR 0 - HR 9999	HR 0 - HR 9999
AR	AR 0.0 - AR 9999.15		AR 0 - AR 9999	AR 0 - AR 9999	AR 0 - AR 9998	AR 0 - AR 9998		AR 0 - AR 9999
LR	LR 0.0 - LR 9999.15		LR 0 - LR 9999	LR 0 - LR 9999	LR 0 - LR 9998	LR 0 - LR 9998		LR 0 - LR 9999
DM	DM 0.0 - DM 9999.15		DM 0 - DM 9999	DM 0 - DM 9999	DM 0 - DM 9998	DM 0 - DM 9998	DM 0 - DM 9999	DM 0 - DM 9999
T/C Bit	T/C Bit 0 - T/C Bit 4095							

Address areas	Data types							
	Bool	Byte	UInt	Int	UDInt	DInt	Real	String
T/C Val			T/C Val 0 - T/C Val 4095	T/C Val 0 - T/C Val 4095				
CPU Status	RUN, MONITOR							
CPU type		CPU type						

Commissioning components

Transferring a project to the HMI device

1. Switch the HMI device to "transfer mode".
2. Set all necessary transfer parameters.
   - Interface
   - Transfer parameters
   - Target storage location
3. Start the project transfer.
   The project is compiled automatically.
   All compilation and transfer steps are logged to a message window.

Interconnecting the PLC with the HMI device

1. Interconnect the PLC with the HMI device using a suitable cable.
2. The message "Connection to PLC .... is established" is output to the HMI device.

Optimizing the configuration

Acquisition cycle and update time

The acquisition cycles for the "Area pointers" and of the tags defined in the configuration software are decisive in terms of the update times which can actually be achieved.

The update time is equivalent to the acquisition cycle + transmission time + processing time.

Items to observe when optimizing the update times in configuration data:

- Optimize the maximum and minimum size of the data areas.
- Acquisition cycles which are too short lead to unnecessary load on overall performance. Set the acquisition cycle according to the rate of change of the process values. The rate of temperature changes at a furnace, for example, is significantly slower compared to the speed rate of an electrical drive. A time of approx. 1 second is a benchmark for the acquisition cycle.

- Avoid any gaps when entering the alarm or screen tags in a data area.
- Changes in the PLC can only be detected reliably if these are available at least within the actual acquisition cycle.

Screens

The refresh rate of screens is determined by the type and volume of data to be visualized.

Only configure short acquisition cycles for objects which actually require shorter refresh cycles. This procedure reduces update times.

Trends

The HMI device always updates all bit-triggered trends whose group bit is set in the "Trend transfer area". It resets the bits in the next cycle.

The group bit in the PLC program can only be set again after the HMI device has reset all bits.

Job mailboxes

A high rate and volume of job mailboxes transferred may lead to overload in communication between the HMI device and the PLC.

The HMI device confirms acceptance of the job mailbox by entering the value zero in the first data word of the job mailbox. The HMI device now processes the job for which it requires a certain time slice. It may take the HMI device some time to process a new job mailbox which is transferred in immediate succession to the job mailbox. The next job mailbox is only accepted if sufficient computing resources are available.

Data exchange

Area pointers for Omron

Area pointers in connections via Omron communication drivers

You use an area pointer to access a data area in the PLC.

For more detailed information on area pointers and their configuration, refer to Section:

"Data exchange using area pointers".

Special features of connections via Omron Host Link

Area pointers can only be created in the following "File types": "DM", "I/O", "HR", "AR", and "LR".

See also

Data exchange using area pointers (Page 8745)

Trends

Trends

A trend is the graphical representation of one or more values from the PLC. The value is read out time-triggered for Basic Panels.

For additional information see:

Configuring trend displays for values from the PLC (Page 8147)

Time-triggered trends

The HMI device reads in the trend values cyclically at an interval specified in the configuration.

Time-triggered trends are suitable for continuous curves, such as the operating temperature of a motor.

Alarms

Configuring alarms

Configuring alarms for non-integrated connections

Several steps are necessary to configure alarms such as warnings, error messages and acknowledgement.

- Step 1: Create tags
- Step 2: Configure alarms
- Step 3: Configure acknowledgment

You can find additional information in the section:

Working with alarms (Page 8157)

Distinctive features when configuring alarms

If you are configuring connections of HMI devices to PLCs of other manufacturers, note the following distinctive features when configuring:

- Data types of the tags
- Addressing of tags
- How the bit positions are counted

Data types

For connections with an Omron communication driver, the following data types are supported:

PLC	Permitted data types	
	Discrete alarms	Analog alarms
CP1, CJ1, CJ2, CS1, CPM	Uint, int	UInt, Int, UDInt, DInt

How the bit positions are counted

For connections with an Omron communication driver, the following counting method applies:

How the bit positions are counted	Left byte								Right byte							
In Omron PLCs	15							8	7							0
In WinCC you configure:	15							8	7							0

Only tags for the "DM", "I/O", "HR", "AR", and "LR" file types are allowed for use as a trigger tag for discrete alarms.

Configuring discrete alarms

Use arrays for discrete alarms and append each individual alarm to one bit of the array tags themselves and not to the individual subelements.

Only tags for the "DM", "I/O", "HR", "AR", LR" areas and the "Int" and "UInt" file types are permitted for discrete alarms and arrays.

Acknowledgment of alarms

Procedure

Create suitable tags on the PLC to acknowledge an error alarm. You assign these tags to an alarm in the "Bit messages" editor. You make the assignment in "Properties > Acknowledgment".

Distinction in terms of acknowledgment:

- Acknowledgment by the PLC
- Acknowledgment on the HMI device

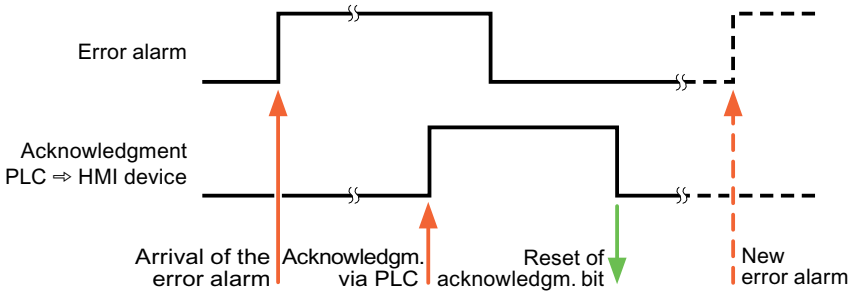
Acknowledgment by the PLC

In "PLC acknowledgment tag", configure the tag or array tag and the bit number that the HMI device uses to identify a PLC acknowledgment.

A bit set in the tag triggers acknowledgment of the assigned error alarm bit at the HMI device. This tag bit returns a function similar to acknowledgment on the HMI device which is triggered by pressing the "ACK" button, for example.

The acknowledgment bit must be located in the same tag as the bit for the error alarm.

Reset the acknowledgment bit before setting the bit in the alarm area again. The figure below shows the pulse diagram.



Acknowledgment on the HMI device

In the "HMI acknowledgment tag" area, configure the tag or array tag as well as the bit number that the HMI device writes to the PLC after acknowledgment. Make sure when you use an array tag that it is not longer than 6 words.

To always create a signal change when setting an assigned acknowledgment bit of a discrete alarm that must be acknowledged, the HMI device will reset the acknowledgment bit assigned to the alarm as soon as it detects an alarm subject to acknowledgment and write the acknowledgment tag in the PLC. There will be a certain delay between detecting the message and writing the acknowledgment tag in the PLC because the HMI device has to process the operations.

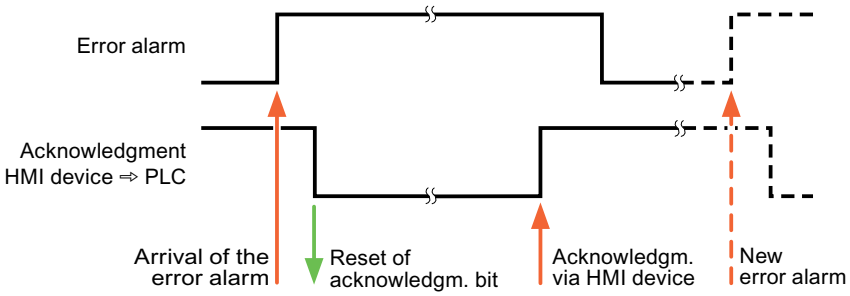
If a discrete alarm subject to acknowledgment is acknowledged by the HMI device, then the corresponding bit in the assigned acknowledgment tag will be set. The entire acknowledgment tag is then written to the PLC by the HMI device. This allows the PLC to recognize that a certain alarm message has been acknowledged at the HMI device.

Note

All alarm bits acknowledged since the last Runtime start will remain in the acknowledgment tag until a new incoming of the respective discrete alarms is detected.

This area should only be read by the PLC because the entire section of the HMI device will be overwritten once the next acknowledgment tag is written.

The figure below shows the pulse diagram.



Communication Service Packages

10.8.11.4 Data exchange using area pointers

General information on area pointers

Introduction

You use an area pointer to access a data area in the PLC. During communication, the PLC and the HMI device alternately access these data areas for read and write operations.

The PLC and the HMI device trigger defined interactions based on the evaluation of stored data.

Configuration of area pointers

Before you use the area pointer, you enable it in "Connections ▸ Area pointers". You then assign the area pointer parameters.

Parameter | Area pointer

Active	Display name	PLC tag	Access mode	Address	Length	Acquisition mode	Acquisition cycle	Comment
☐	Coordination	<Undefined>	<symbolic access>		1	Cyclic continuous	<Undefined>	
☐	Date/time	<Undefined>	<symbolic access>		6	Cyclic continuous	<Undefined>	
☐	Job mailbox	<Undefined>	<symbolic access>		4	Cyclic continuous	<Undefined>	
☐	Data record	<Undefined>	<symbolic access>		5	Cyclic continuous	<Undefined>	

Global area pointer of HMI device

Connection	Display name	PLC tag	Access mode	Address	Length	Acquisition mode	Acquisition cycle	Commer
<Undefined>	Project ID	<Undefined>	<symbolic access>		1	Cyclic continuous	<Undefined>	
<Undefined>	Screen number	<Undefined>	<symbolic access>		5	Cyclic continuous	<Undefined>	
<Undefined>	Date/time PLC	<Undefined>	<symbolic access>		6	Cyclic continuous	<Undefined>	

- Active
  Enables the area pointer.
- Pointer name
  Name of the area pointer specified by WinCC.
- PLC tag
  Here you select the PLC tag or the tag array that you have configured as the data area for the area pointer.
- Address
  No address is entered into this field because of the symbolic access.
- Length
  WinCC specifies the length of the area pointer.
- Acquisition cycle
  You specify the acquisition cycle in this field for area pointers that are read by the HMI device. Note that a very short acquisition time may have a negative impact on HMI device performance.
- Comment
  Enter a comment, for example, to describe the purpose of the area pointer.

Accessing data areas

Accessing data areas

The following table shows how HMI devices and PLCs access individual data areas for read (R) or write (W) operations.

Data area	Required for	HMI device	PLC
Screen number	Evaluation by the PLC in order to determine the active screen.	W	R
Data record	Transfer of data records with synchronization	R/W	R/W
Date/time	Transfer of the date and time from the HMI device to the PLC	W	R
Date/time PLC	Transfer of the date and time from the PLC to the HMI device	R	W
Coordination	Requesting the HMI device status in the PLC program	W	R
Project ID	Runtime checks for consistency between the WinCC project ID and the project in the PLC	R	W
Job mailbox	Triggering of HMI device functions by the PLC program	R/W	R/W

"Screen number" area pointer

Function

The HMI device saves information about the screen called on the HMI device to the "Screen number" area pointer.

This allows the transfer of the current screen contents from the HMI device to the PLC. The PLC can trigger specific reactions such as the call of a different screen.

Use

Configure and enable the area pointer in "Communication > Connections" before you put it into use. You can create only **one** instance of the "Screen number" area pointer and only on **one** PLC.

The screen number is always transferred to the PLC when a new screen is activated or when the focus within a screen changes from one screen object to another.

Structure

The area pointer is a data area in the memory of the PLC with a fixed length of 5 words.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1. word	Current screen type															
2. word	Current screen number															
3. word	Reserved															
4th word	Current field number															
5. word	Reserved															

- Current screen type
  "1" for root screen or
  "4" for permanent window
- Current screen number
  1 to 32767
- Current field number
  1 to 32767

"Date/time" area pointer

Function

This area pointer is used to transfer the date and time from the HMI device to the PLC.

The PLC writes control job "41" to the job mailbox.

When it evaluates the control job, the HMI device writes its current date and the time in the data area configured in the "Date/time" area pointer. All definitions are coded in BCD format.

The "Date/Time" area pointer when used in a project which contains multiple connections must be enabled for each configured connection.

Note

You cannot use the "Date/Time PLC" area pointer if you have configured the "Date/Time" area pointer.

The date/time data area has the following structure:

Data word	Left byte								Right byte								
	15							8	7							0	
n+0	Reserved								Hour (0 to 23)								Time
n+1	Minute (0 to 59)								Second (0 to 59)								
n+2	Reserved								Reserved								
n+3	Reserved								Weekday (1 to 7, 1=Sunday)								Date
n+4	Day (1 to 31)								Month (1 to 12)								
n+5	Year (80 to 99/0 to 29)								Reserved								

Note

When making entries in the "Year" data area, you should note that values 80 to 99 result in years 1980 through 1999, while the values 0 to 29 result in the years 2000 through 2029.

"Date/time PLC" area pointer

Function

This area pointer is used to transfer the date and time from the PLC to the HMI device. Use this area pointer if the PLC is the time master.

The PLC loads the data area of the area pointer. All definitions are coded in BCD format.

The HMI device reads the data cyclically within the configured acquisition cycle and synchronizes itself.

Note

Set an acquisition cycle of sufficient length for the Date/time area pointer in order to avoid any negative impact on HMI device performance.
Recommended: Acquisition cycle of 1 minute if your process can handle it.

"Date/Time PLC" is a global area pointer and may be configured only once per project.

Note

You cannot use the "Date/Time" area pointer if you have configured the "Date/Time PLC" area pointer.

The "Date/time PLC" data area has the following structure:

Data word	Left byte			Right byte		
	15	. . . . .	8	7	. . . . .	0
n+0	Year (80 to 99/0 to 29)			Month (1 to 12)		
n+1	Day (1 to 31)			Hour (0 to 23)		
n+2	Minute (0 to 59)			Second (0 to 59)		
n+3	Reserved			Reserved		Weekday (1 to 7, 1=Sunday)
n+4 [1)]	Reserved			Reserved		
n+5 [1)]	Reserved			Reserved		

1) The two data words must exist in the data area to ensure that the data format matches WinCC and to avoid reading false information.

Note

When making entries in the "Year" data area, you should note that values 80 to 99 result in years 1980 through 1999, while the values 0 to 29 result in the years 2000 through 2029.

"Coordination" area pointer

Function

The "Coordination" area pointer is used to implement the following functionality:

- Detecting the startup of the HMI device in the control program
- detection in the control program of the current HMI device operating mode
- detection in the control program of the HMI devices ready to communicate state

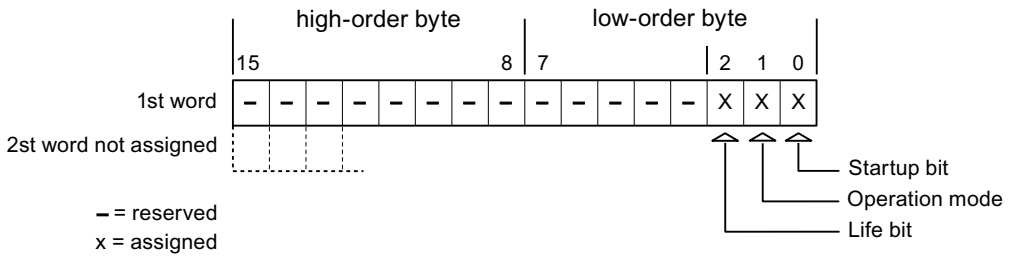
The "Coordination" area pointer has a length of one word.

Use

Note

The HMI device always writes the entire coordination area when updating the area pointer. The control program may not make changes to the coordination area for this reason.

Assignment of bits in the "Coordination" area pointer



Startup bit

The startup bit is set briefly to "0" by the HMI device during startup. It sets the bit permanently to "1" when startup is completed.

Operating mode

The operating mode bit is set to 1 as soon as the user switches the HMI device offline. The state of the operating mode bit is "0" during normal operation of the HMI device. You can determine the current operating mode of the HMI device by reading this bit.

Life bit

The HMI device inverts the life bit at intervals of approximately one second. You can check whether or not the connection to the HMI device is still up by querying this bit in the control program.

"Project ID" area pointer

Function

You can check whether the HMI device is connected to the correct PLC at the start of runtime. This check is important when operating with several HMI devices.

The HMI device compares a value stored on the PLC with the value specified in configuration. This ensures compatibility of configuration data with the control program. If discrepancy is detected, a system event is displayed on the HMI device and runtime is stopped.

Use

To use this area pointer, set up the following during the configuration:

- Define the version of configuration. Possible values between 1 and 255.
  You enter the version in the editor "Runtime settings > General" in the "Identification" area.
- Data address of the value for the version that is stored in the PLC:
  You enter the data address in the editor "Communication > Connections".

Connection failure

A connection failure to a device on which the "project ID" area pointer is configured results in all the other connections in the project being switched to "offline".

This behavior has the following prerequisites:

- You have configured several connections in a project.
- You are using the "project ID" area pointer in at least one connection.

Causes which may set connections "offline":

- The PLC is not available.
- The connection has been switched offline in the engineering system.

"PLC job" area pointer

Function

The PLC can use the job mailbox to transfer jobs to the HMI device to trigger corresponding actions on the HMI device. These functions include, for example:

- Display screen
- Set date and time

Data structure

The first word of the job mailbox contains the job number. Depending on the job mailbox, up to three parameters can be transferred.

Word	Left byte	Right byte
n+0	0	Job number
n+1	Parameter 1	
n+2	Parameter 2	
n+3	Parameter 3	

The HMI device evaluates the job mailbox if the first word of this job is not equal to zero. This means that the parameters must be entered in the job mailbox first, followed by the job number.

When the HMI device accepts the job mailbox, the first word is set to 0 again. The execution of the job mailbox is generally not completed at this point in time.

Job mailboxes

All job mailboxes and their parameters are listed below. The "No." column contains the job number of the job mailbox. Job mailboxes can only be triggered by the PLC when the HMI device is online.

Note

Please note that not all HMI devices support job mailboxes.

No.	Function	
14	**Setting the time (BCD coded)**	
	Parameter 1	Left byte: - Right byte: hours (0-23)
	Parameter 2	Left byte: minutes (0-59) Right byte: seconds (0-59)
	Parameter 3	-
15	**Setting the date (BCD coded)**	
	Parameter 1	Left byte: - Right byte: weekday (1-7: Sunday-Saturday)
	Parameter 2	Left byte: day (1-31) Right byte: month (1-12)
	Parameter 3	Left byte: year
23	**User logon**	
	Logs the user on with the name "PLC user" at the HMI device with the group number transferred in Parameter 1. The logon is possible only when the transferred group number exists in the project.	
	Parameter 1	Group number 1 to 255
	Parameter 2, 3	-
24	**User logoff**	
	Logs off the current user. (The function corresponds to the "logoff" system function)	
	Parameter 1, 2, 3	-
40	**Transfer date/time to PLC**	
	(in the S7 format DATE_AND_TIME) An interval of at least 5 seconds must be maintained between two successive jobs to prevent overload of the HMI device.	
	Parameter 1, 2, 3	-
41	**Transfer date/time to PLC**	
	(In OP/MP format) An interval of at least 5 seconds must be maintained between successive jobs in order to prevent overload of the HMI device.	

No.	Function	
14	Setting the time (BCD coded)	
	Parameter 1, 2, 3	-
46	Update tags	
	Causes the HMI device to read the current value of the tags•from the PLC whose update ID matches the value transferred in parameter 1. (Function corresponds to the "UpdateTag" system function.)	
	Parameter 1	1 - 100
49	Delete alarm buffer	
	Deletes all analog alarms and discrete alarms of the "Warnings" class from the alarm buffer.	
	Parameter 1, 2, 3	-
50	Delete alarm buffer	
	Deletes all analog alarms and discrete alarms of the "Errors" class from the alarm buffer.	
	Parameter 1, 2, 3	-
51	Screen selection [1]	
	Parameter 1	Screen number
	Parameter 2	-
	Parameter 3	Field number
69	Read data record from PLC	
	Parameter 1	Recipe number (1-999)
	Parameter 2	Data record number (1-65535)
	Parameter 3	0: Do not overwrite existing data record 1: Overwrite existing data record
70	Write data record to PLC	
	Parameter 1	Recipe number (1-999)
	Parameter 2	Data record number (1-65535)
	Parameter 3	-

[1] OP 73, OP 77A and TP 177A HMI devices also execute the "Screen selection" job mailbox if the on-screen keyboard is active.

"Data record" area pointer

"Data mailbox" area pointer

Function

When data records are transferred between the HMI device and PLC, both partners access common communications areas on the PLC.

Data transfer types

There are two ways of transferring data records between the HMI device and PLC:

- Transfer without synchronization
- Transfer with synchronization over the data record

Data records are always transferred directly. That is, the tag values are read from an address or written to an address configured for this tag directly, without redirecting the values by means of interim memory.

Initiating the transfer of data records

There are three ways of triggering the transfer:

- Operator input in the recipe view
- Job mailboxes
  The transfer of data records can also be triggered by the PLC.
- Triggering by configured functions

If the transfer of data records is triggered by a configured function or by a job mailbox, the recipe view on the HMI device remains operable. The data records are transferred in the background.

Simultaneous processing of several transfer requests is, however, not possible. In this case, the HMI device rejects the other transfer requests with a system event.

Transfer without synchronization

If you select asynchronous transfer of data records between the HMI device and PLC, there is no coordination over the common data areas. It is therefore unnecessary to set up a data area during configuration.

Asynchronous data record transfer can be a useful alternative, for example, when:

- The system is capable of excluding the risk of uncontrolled overwriting of data by the communication peer.
- The PLC does not require information about the recipe number and data record number.
- The transfer of data records is triggered by the operator of the HMI device.

Reading values

When a read job is triggered, the values are read from the PLC addresses and transferred to the HMI device.

- Triggering by the operator in the recipe view:
  The values are downloaded to the HMI device. You can then process, edit, or save these values, for example.
- Triggering by a function or job mailbox:
  The values are saved immediately to the data volume.

Writing values

When a write job is triggered, the values are written to the PLC addresses.

- Triggering by the operator in the recipe view:
  The current values are written to the PLC.
- Triggering by a function or job mailbox:
  The current values are written to the PLC from the data medium.

Transfer with synchronization

If you select synchronous transfer, both communication partners set status bits in the common data area. You can use this mechanism to prevent uncontrolled overwriting of data in either direction in your control program.

Application

Synchronous data record transfer can be a useful solution, for example, when:

- The PLC is the "active partner" in the transfer of data records.
- The PLC evaluates the information about the recipe number and data record number.
- The transfer of data records is triggered by means of a Job mailbox.

Requirements

In order to synchronize transfer of data records between the HMI device and the PLC, the following requirements must be met during configuration:

- An area pointer has been set up: "Communication > Connections" editor in "Area pointer".
- The PLC with which the HMI device synchronizes transfer of data records is specified in the recipe:
  "Recipes" editor in the inspector window the option "Coordinated transfer of data records" under "General > Synchronization > Settings"

Structure of the data area

The data area has a fixed length of 5 words. Structure of the data area:

	15		0
1. Word	Current recipe number (1 - 999)		
2. Word	Current data record number (0 - 65535)		
3. Word	Reserved		
4. Word	Status (0, 2, 4, 12)		
5. Word	Reserved		

- Status
  The status word (word 4) can adopt the following values:

Value		Meaning
Decimal	Binary	
0	0000 0000	Transfer permitted, data record free
2	0000 0010	Transferring.
4	0000 0100	Transfer completed without error
12	0000 1100	Transfer completed with error

Sequence of a transfer started by the operator in the recipe display

Reading from the PLC started by the operator in the recipe view

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe number to be read and the status "Transferring" in the data record and sets the data record number to 0.	Abort with system event.
3	The HMI device reads the values from the PLC and displays them in the recipe view. If the recipes have synchronized tags, the values from the PLC are also written to the tags.	
4	The HMI device sets the status "Transfer completed."	
5	The control program must reset the status word to zero in order to enable further transfers.	

Writing to the PLC started by the operator in the recipe view

Step	Action	
	Check: Status word = 0?	
1	Yes	No
	The HMI device enters the recipe and data record number to be written and the status "Transferring" in the data record.	Abort with system event.
2	The HMI device writes the current values to the PLC. If the recipes have synchronized tags, the changed values are synchronized between the recipe view and tags and then written to the PLC.	
3	The HMI device sets the status "Transfer completed."	
4	If required, the control program can now evaluate the transferred data.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Note

The status word may only be set by the HMI device. The PLC may only reset the status word to zero.

Note

The PLC may only evaluate the recipe and data record numbers when data inconsistency is detected if one of the conditions outlined below has been met:

- The data mailbox status is set to "Transfer completed".
- The data mailbox status is set to "Transfer completed with error".

Sequence of the transfer triggered by a PLC job

The transfer of data records between the HMI device and the PLC can be initiated by either one of these stations.

The two job mailboxes No. 69 and No. 70 are available for this type of transfer.

No. 69: Read data record from PLC ("PLC → DAT")

Job mailbox no. 69 transfers data records from the PLC to the HMI device. The job mailbox is structured as follows:

	Left byte (LB)	Right byte (RB)
Word 1	0	69
Word 2	Recipe number (1-999)	
Word 3	Data record number (1 to 65535)	
Word 4	Do not overwrite existing data record: 0 Overwrite existing data record: 1	

No. 70: Write data record to PLC ("DAT → PLC")

Job mailbox No. 70 transfers data records from the HMI device to the PLC. The job mailbox is structured as follows:

	Left byte (LB)	Right byte (RB)
Word 1	0	70
Word 2	Recipe number (1-999)	
Word 3	Data record number (1 to 65535)	
Word 4	—	

Sequence when reading from the PLC with job mailbox "PLC → DAT" (no. 69)

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the job and the status "Transferring" in the data record.	Abort without return message.
3	The HMI device reads the values from the PLC and saves these to the data record defined in the job mailbox.	
4	• If "Overwrite" was selected in the job, an existing data record is overwritten without any prompt for confirmation. The HMI device sets the status "Transfer completed." • If "Do not overwrite" was selected in the job, and the data record already exists, the HMI device aborts the job and enters 0000 1100 in the status word of the data record.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Sequence of writing to the PLC with job mailbox "DAT → PLC" (no. 70)

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the job and the status "Transferring" in the data record.	Abort without return message.
3	The HMI device fetches the values of the data record specified in the function from the data medium and writes the values to the PLC.	
4	The HMI device sets the status "Transfer completed."	
5	The control program can now evaluate the transferred data. The control program must reset the status word to zero in order to enable further transfers.	

Sequence of the transfer when triggered by a configured function

Reading from the PLC using a configured function

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the function and the status "Transferring" in the data record.	Abort with system event.
3	The HMI device reads the values from the PLC and stores them in the data record specified in the function.	

Step	Action	
4	• If "Yes" was selected for the "Overwrite" function, an existing data record is overwritten without any prompt for confirmation. The HMI device sets the status "Transfer completed." • If "No" was selected for the "Overwrite" function and the data record already exists, the HMI device aborts the job and enters 0000 1100 in the status word of the data record.	
5	The control program must reset the status word to zero in order to enable further transfers.	

Writing to the PLC by means of configured function

Step	Action	
1	Check: Status word = 0?	
	Yes	No
2	The HMI device enters the recipe and data record number specified in the function and the status "Transferring" in the data record.	Abort with system event.
3	The HMI device fetches the values of the data record specified in the function from the data medium and transfers the values to the PLC.	
4	The HMI device sets the status "Transfer completed."	
5	The control program can now evaluate the transferred data. The control program must reset the status word to zero in order to enable further transfers.	

Possible causes of error when transferring data records

Possible causes of error

The section below shows possible error causes which lead to the cancellation of data record transfer:

- Tag address not set up on the PLC
- Overwriting data records not possible

- Recipe number does not exist
- Data record number does not exist

Note

The status word may only be set by the HMI device. The PLC may only reset the status word to zero.

Note

The PLC may only evaluate the recipe and data record numbers when data inconsistency is detected if one of the conditions outlined below has been met:

- The data mailbox status is set to "Transfer completed".
- The data mailbox status is set to "Transfer completed with error".

Reaction to an aborted transfer due to errors

If the transfer of data records is aborted due to errors, the HMI device reacts as follows:

- Triggering by the operator in the recipe view
  Information in the status bar of the recipe view and output of system alarms
- Triggered by function
  Output of system alarms
- Triggering by job mailbox
  No return message on the HMI device

You can nonetheless evaluate the status of the transfer by querying the status word in the data record.

10.9 Using global functions

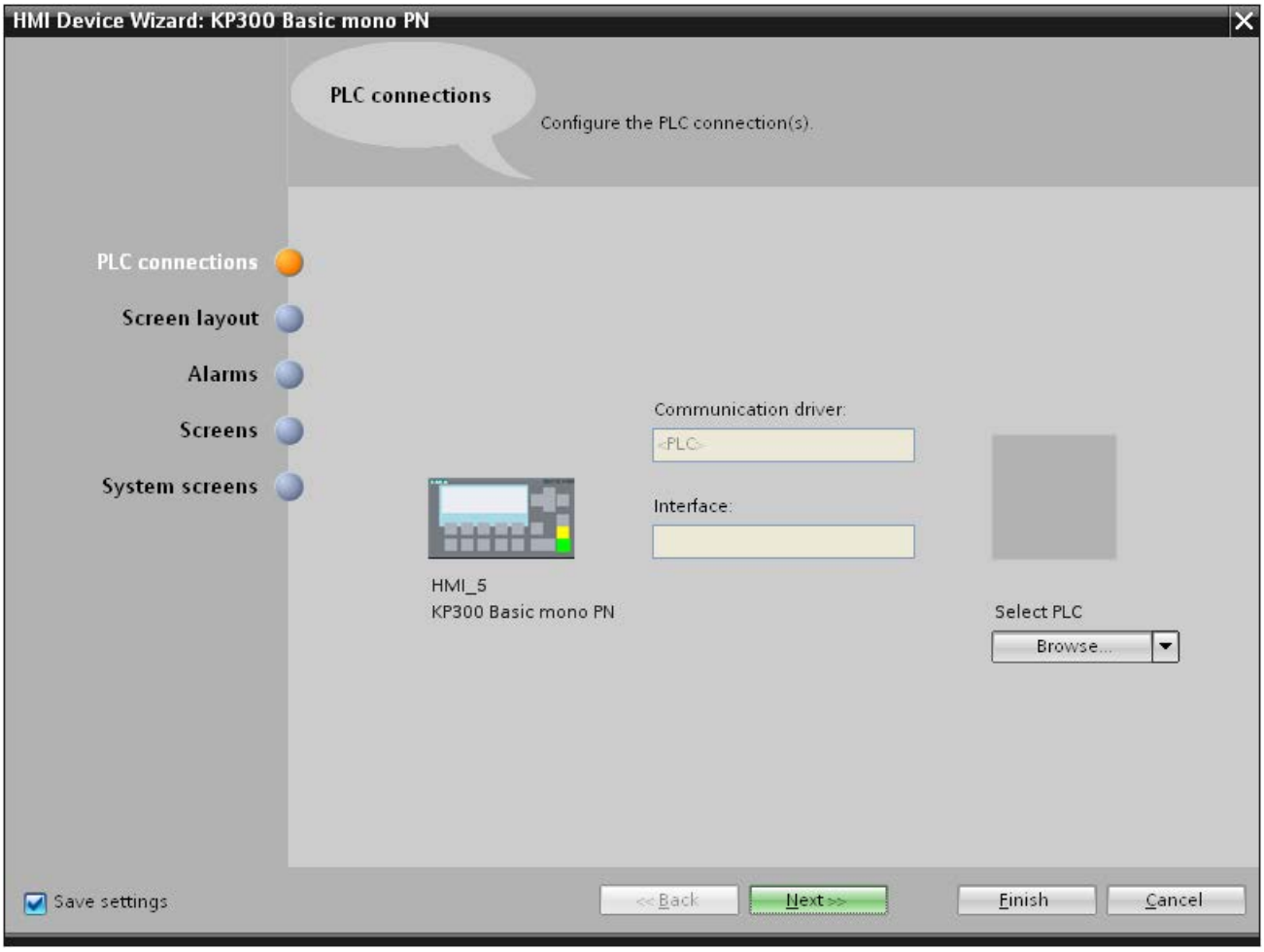
10.9.1 HMI device wizard basics

Introduction

The HMI device wizard will automatically start when you create a new HMI device in your project.

HMI device wizard

The HMI device wizard will guide you through each dialog step by step and help you set up a device. You use the HMI device wizard to specify the basic settings for your HMI device, such as screen layout and the connection to your PLC.



10.9.2 Working with libraries

10.9.2.1 Basics on libraries

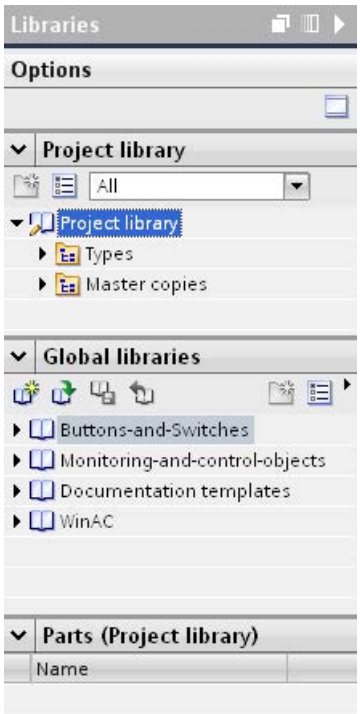
Introduction

Store all objects you need frequently in the libraries. An object that is stored in the library only has to be configured once. It can then be used repeatedly as often as required. Library objects extend the number of available screen objects and increase the effectiveness during configuration through the multiple usage of ready-to-use objects.

Your WinCC software package is supplied with comprehensive libraries that contain, for example, "Motor" or "Valve" objects. You may, however, define your own library objects.

You manage libraries in the "Libraries" task card or library view. The following libraries are available:

- Project library
- Global libraries



Note

There is a symbol library in the "Tools" task card in the "Graphics" palette.

Project library

There is one library for each project. Objects of the project library are stored alongside with the project data and are available only for the project in which the library was created. If the project is moved to another PC, any project library created in it is also moved.

To use the library object of the project library in other objects, move or copy the object into a global library.

Global libraries

A global library is saved independently of the project data in its own file with the extension *.al12.

A project can access several global libraries. A global library may be used concurrently in several projects.

When a library object is changed by a project, this library will be changed in all projects in which these libraries are open.

Library objects

A library can contain all WinCC objects. Examples:

- Complete HMI device
- Screens
- Display and control objects including tags and functions
- Graphics
- Tags
- Alarms
- Text and graphics lists
- Faceplates
- User data types

See also

Copy templates and types (Page 8765)

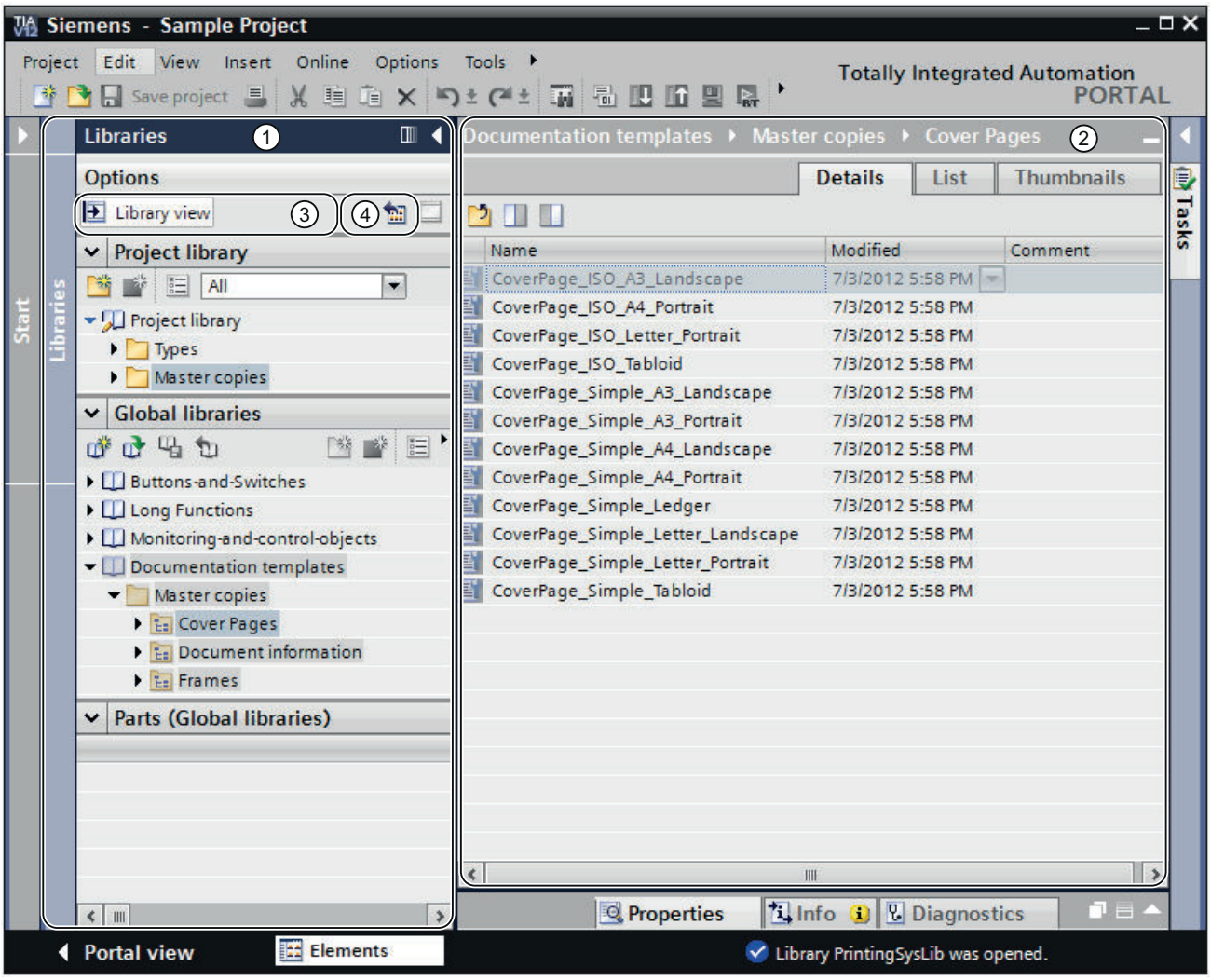
10.9.2.2 Overview of the library view

Function of the library view

The library view combines the functionality of the "Libraries" task card and the overview window. In the library view, the elements of a library are displayed in various views. In the details view, for example, you see additional properties of the individual elements. You can also edit and version the types in the library view.

Layout of the library view

The following figure shows the components of the library view:



1 Library tree
2 Library overview
3 "Library view" button
4 "Open or close library overview" button

Library tree

The library tree is similar to the "Libraries" task card, apart from a few minor differences. In contrast to the task card, there is no "Elements" palette, because the elements are displayed in the library overview. In addition, you can close the library view in the library tree, or open and close the library overview.

Library overview

The library overview corresponds to the overview window and displays the elements of the currently selected object in the library tree. You can display the elements in three different views. You can also perform the following actions in the library overview, for example:

- Copying elements
- Moving elements
- Versioning types
- Editing faceplates and HMI user data types
- Editing type instances

10.9.2.3 Copy templates and types

Introduction

Both the "Project library" and the "Global library" contain the two folders "Master copies" and "Types". You can create or use the library objects either as a master copy or a type.

Master copies

Use master copies to create independent copies of the library object.

Types

Create instances of objects of the "Types" folder and use these in your project. The instances are bound to their respective type.

Administration of the library objects

You can copy and move library objects to another library. You can only copy master copies to the "Master copies" folder or any sub-folder of "Master copies". You can also only insert types in the "Types" folder or any sub-folder of "Types".

See also

Basics on libraries (Page 8761)

State of type versions (Page 8779)

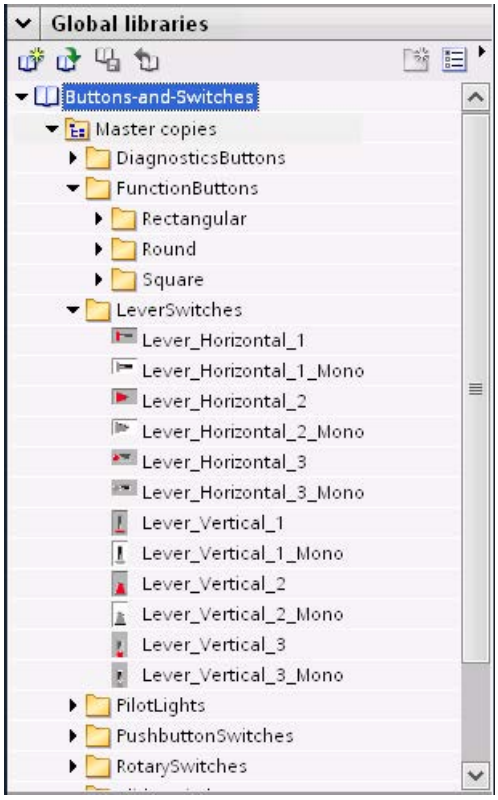
10.9.2.4 Libraries in WinCC

Introduction

The WinCC software package includes extensive libraries. Sorted by topic into folders, they contain preassembled graphic objects which you can use in screens for operation and monitoring of your plant.

Global library "Buttons and Switches"

The libraries "Buttons and Switches" offer a wide selection of buttons and switches.



The folders divide switches and buttons into categories. The "DiagnosticsButtons" folder contains the object "System diagnostics indicator", for example. You use the "System diagnostics indicator" object for system diagnostics in your plant.

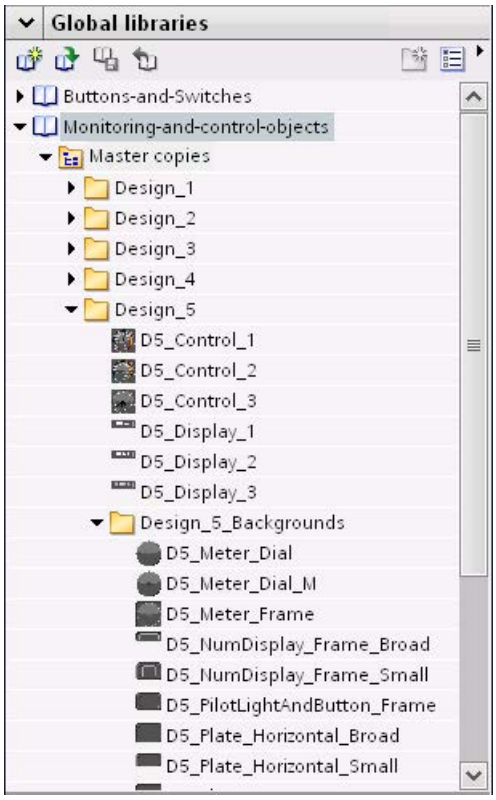
Note

You can only use the objects in the "DiagnosticsButtons" folder on Comfort Panels.

You cannot use objects which have "Switch" in the object name or in the associated folder name in Runtime Professional.

Global library "Monitoring and Control objects"

The "Monitoring and Control objects" library offers more complex display and operating objects in several designs and corresponding control lights, buttons and switches.



In addition, graphics views for the designs are stored in the "Design_Backgrounds" folder; they can be used as object backgrounds for the custom extension of the scope of the library.

Note

You cannot use objects which have "Switch" in the object name in Runtime Professional. The same applies for the object "D5_Display_3" with the date/time field it contains.

10.9.2.5 Managing libraries

Overview of the library management

Function of the library management

Master copies and types with dependencies to other library elements are subject to some functional restrictions. They can, for example, not be deleted as long as dependencies still

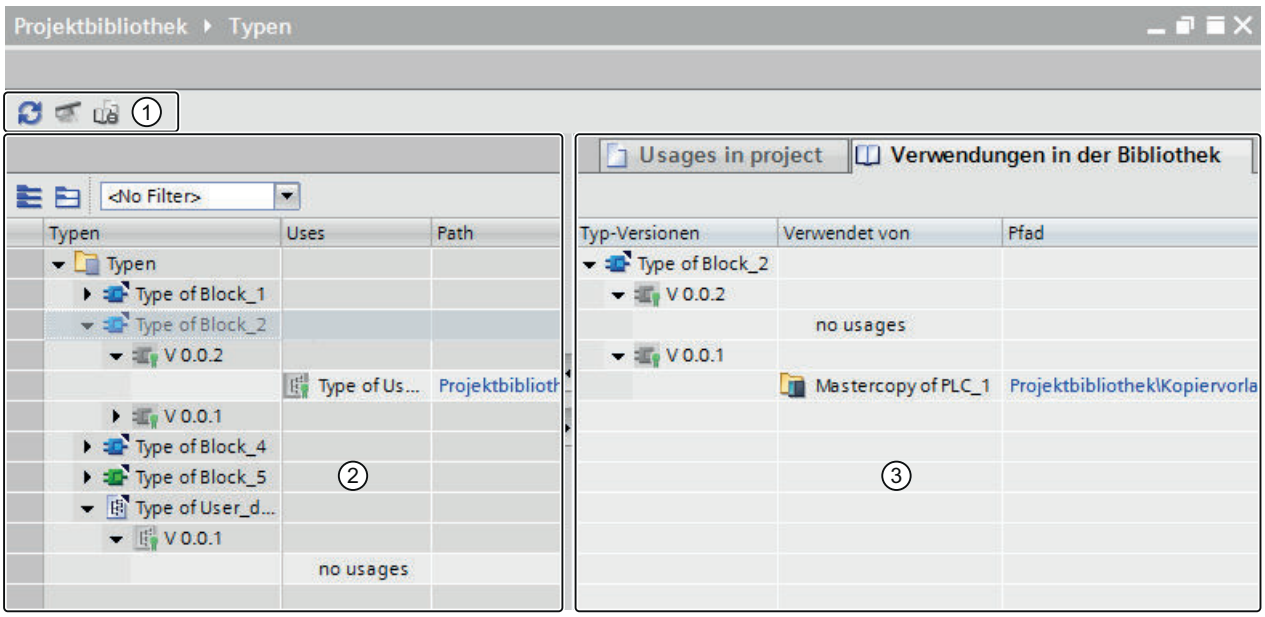
exist. This prevents other library elements from becoming useless. The library management is used to identify the dependencies and to create an overview of the work progress.

The library management offers the following functions:

- Display of the correlations of types and master copies
  If a type is referenced in other types or master copies, the correlations are displayed in the library management. You will also be able to see which library elements reference a type or a master copy.
- Display of points of use of types in the project
- Display all types that include a version with the "In testing" or "In progress" status

Layout of the library management

The following figure shows the components of the library management:



① Toolbar of the library management

② "Types" area

③ "Uses" area

Toolbar of the library management

You can perform the following tasks in the toolbar of the library management:

- Update view
  If the project was changed, you can update the view of the library management.
- Clean up library
  You can clean up the project library and global libraries. By cleaning up a library, you delete all types and type versions that are not linked to an instance in the project.
- Harmonize project
  By harmonizing a project, you adapt the names and the path structures of type uses in the project to the corresponding names and path structures of the types within a library.

"Types" area

The "Types" area displays the contents of the folder that you have selected in the library view. You can open or close all types by using the buttons in the toolbar of the "Types" area. You can also use the "Filter" drop-down list to filter the view and display all types with the "In testing" or "In progress" status only. For each type, the types that it references are displayed.

"Uses" area

The "Uses" area gives you an overview of the points of use of the selected types and master copies. The "Uses" area is divided into two tabs:

- "Uses in the project" tab
  The "Uses in the project" tab is used to show the instances of type versions and their respective point of use in the project. When you select an instance, you can show the cross references of the instance in the project in the Inspector window.
- "Uses in the library" tab
  The "Uses in the library" tab is used to show all points within the library at which a type or a master copy is used.

Opening library management

Procedure

To open the library management, follow these steps:

1. Open the library view.
2. Select a type or any folder that contains types.
3. Select the "Library management" command from the shortcut menu.

Result

The library management opens and the types are displayed with their versions.

Filtering types in the library management

Introduction

A filter function in the library management enables you to limit the displayed types. The following filters are available:

- Display all types that include a version with "in progress" status
- Display all types that have no instances in the project
- Display all types that have more than one version
- Display all released types

Requirement

At least one type has been created.

Filtering for all types with "in progress" status

1. Select the "Types" folder in the project library.
2. Select the "Library Management" command in the shortcut menu of the "Types" folder.
   The library management opens.
3. Select "Pending changes" from the "Filter" drop-down list.
   The "Types" area only displays types that have the "in progress" status.

Filtering for all types that have no instances in the project

1. Select the "Types" folder in the project library.
2. Select the "Library Management" command in the shortcut menu of the "Types" folder.
   The library management opens.
3. Select "No instances in project" from the "Filter" drop-down list.
   "Not used in project" or "No instances in the project"?
   The "Types" area only displays types that have the "in progress" status.

Filtering for types with multiple versions

1. Select the "Types" folder in the project library.
2. Select the "Library Management" command in the shortcut menu of the "Types" folder.
   The library management opens.
3. Select "Multiple versions" from the "Filter" drop-down list.
4. The "Types" area only displays types that have more than one version.

Filtering for all released types

1. Select the "Types" folder in the project library.
2. Select the "Library Management" command in the shortcut menu of the "Types" folder.
   The library management opens.
3. Select "Released types" from the "Filter" drop-down list.
   The "Types" area only displays types that have released versions.

See also

Opening library management (Page 8769)

Creating a global library

Introduction

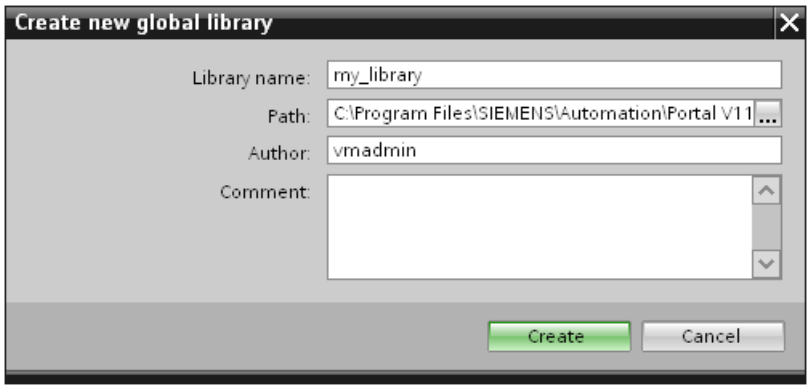
In the libraries you store the configured objects that you want to use several times in your configuration. To use objects in several projects, create a global library.

Requirement

- A project is open.
- The "Libraries" task card is displayed or the "Library view" is open.

Procedure

1. Click the icon under "Global library".
   The "Create new global library" dialog opens.



2. Enter a name.
3. Select the path where the new library is to be stored.
4. Click "Create".

Result

The new library is shown in the "Global libraries" palette. The global library contains the "Types", "Master copies" and "Common data" folders. Under the "Common data" you can find reports for the global library.

A folder with the name of the global library is created in the file system at the storage location of the global library. This actual library file is given the file name extension ".al12".

Saving a global library

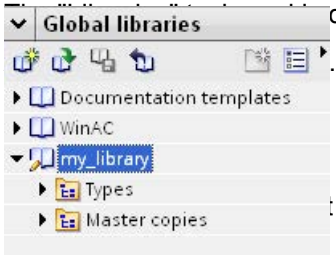
Introduction

A global library is stored in a separate file on your hard disk drive. The file contains the objects of the global library including the referenced objects. E.g. the reference of a tag which was configured on an I/O field is also saved in the library.

WinCC prompts you to save the global libraries when you close WinCC or your project without saving. You also can store the global library during configuration, without storing the entire project.

Requirement

- You have opened a project which contains at least one library.
- displayed or the library view is open.
- .



Procedure

1. you want to save.
2. Click the icon in the "Global library" palette.

You can alternatively select the "Save Library" command in the shortcut menu.

If you want to store the global library in a different folder, select "Save as" in the shortcut menu. Select the path in which you want to store the new library and enter a file name.

Result

The global libraries are saved under their current file name or a file name you have specified.

Opening a global library

Introduction

In WinCC, the global libraries are stored in separate files. You can use a global library in every project.

Requirement

- You have saved a global library.
- A project is open.
- The "Libraries" task card is displayed or the library view is open.

Procedure

1. Click the icon in the "Global library" palette.
   The "Open global library" dialog box is displayed.
2. Select the path in which the library is stored.
3. Click "Open".

Note

To have the access to a global library from multiple projects, open the global library read-only. As soon as a global library is not opened read-only, access from other projects is blocked.

Result

WinCC displays the opened global library in the "Global library" palette.

Displaying logs of global libraries

Logs listing all changes made to the global library are created when global libraries are updated. The logs are stored together with the global library and are always available once you have opened the global library.

Procedure

To open the logs of a global library, follow these steps:

1. Open the global library in the "Libraries" task card or in the library view.
2. Open "Common data > Logs" in the lower-level folder.
3. Double-click the required log.
   The log opens in the work area.

Updating a project with the contents of a library

Introduction

After you have edited several types in the project library, update all instances in the project to the most recent version of the types from the project library.

Requirement

The "Libraries" task card or the library view is open.

Procedure

1. Select the project library.
2. Select "Update > Project" from the shortcut menu. A dialog opens.
3. Select either the entire project or individual devices for the update.
4. Select the "Delete all unused versions of affected types" check box to delete all older versions of the updated types from the project library.
5. Click "OK" to confirm.

Result

All instances of the types are updated in the project to the most recent version of the selected types in the project library.

You can find a log of the update process in the project tree under "Common data".

Updating a library with the contents of another library

The following options are available for updating libraries:

- Updating a global library with types from another global library
- Updating the project library with types from a global library

Each of the following elements can be selected as source for the update:

- An entire library
- Individual folders within a library
- Individual types

Requirement

To update a global library, open the library with write rights.

Requirement

The "Libraries" task card or the library view is open.

Procedure

To update a library with the contents of a different library, follow these steps:

1. Select the entire library or a folder within the library or individual types.
2. Right-click the source and select the "Update > Library" command from the shortcut menu. The "Update library" dialog box opens.

3. Select the type of library you want to update:
   – Select "Update the project library" to update the project library with types from a global library.
   – Select "Update a global library" if you want to update a global library.
4. Optional: Select the global library you want to update from the drop-down list.
5. Select the "Delete all unused versions of affected types" check box to delete all older versions of the updated types from the project library.
6. Click "OK" to confirm.

Result

- Types not yet available in the target library are supplemented there with all their versions. More recent versions are added to the types that already exist in the target library.
  If a more recent version of a type already exists in the target library, the latest version is nevertheless copied and automatically assigned a newer version number.
- A log listing all performed changes to the target library is created for the update process. If you have updated the project library, you can find the log in the project tree under "Common data > Logs".
  If you have updated a global library, you can find the log in the "Common data > Logs" folder in the level below the global library.

10.9.2.6 Managing objects in a library

Displaying library objects

Introduction

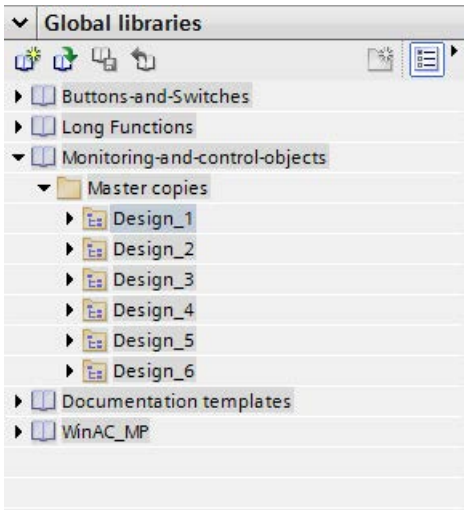
The libraries are displayed as file folders in the corresponding palette. The elements contained in the library are displayed in the file folder and in the "Elements" palette.

Requirement

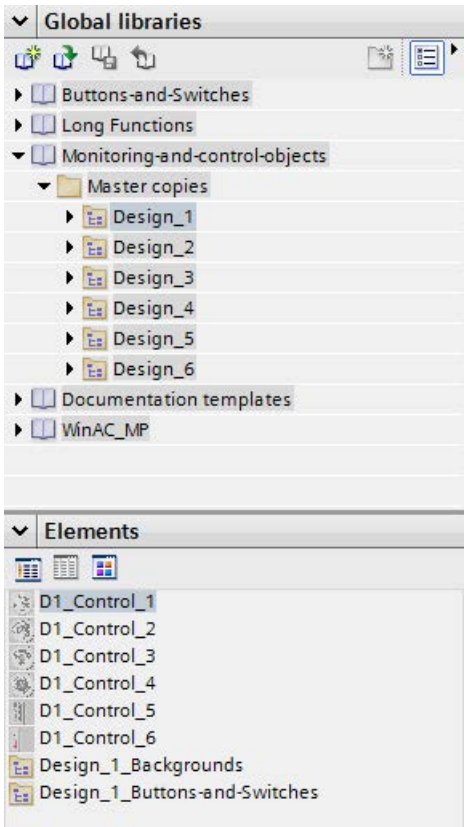
- At least one library object has been created in a library.
- The "Libraries" task card is opened.

Procedure

1. Select the library in the corresponding palette whose library objects you want to display.



2. Click .
   The contained library objects are displayed in the "Elements" palette.



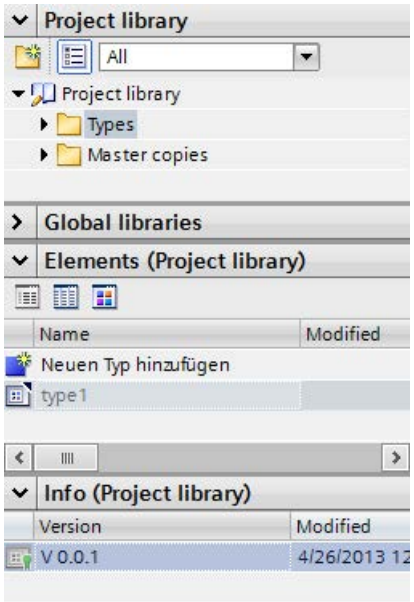
3. Click one of the following icons:

Icon	Description
	Element view in detailed mode
	Element view in list mode
	Element view in overview mode with icons

When several objects are assigned to the library with a multiple selection, only one of the objects is shown in the "Elements" palette. The individual components of this element are displayed in the "Parts" palette.

Show parts of the library objects

1. Select the library in the corresponding palette from which you want to view the components of an element.
2. Click .
3. The contained library objects are displayed in the "Elements" palette.
4. Select the element.
   The "Parts" palette shows the objects of which the element consists.



Result

The library objects are displayed in accordance with the configuration. The components of the faceplates are displayed.

Storing an object in a library

Introduction

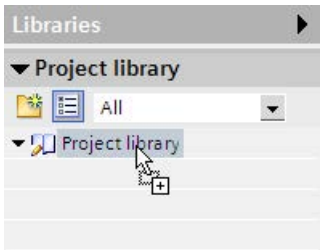
You can store all of WinCC objects, such as screens, tags, graphic objects or alarms in your libraries. You can use drag-and-drop to move the corresponding object from the work area, project window or detail view to the library. In a library you have divided into categories, you can directly add objects to a specific category.

Requirement

- The "Screens" editor is open.
- A screen object has been created in the work area of the screen.
- The created libraries are displayed.

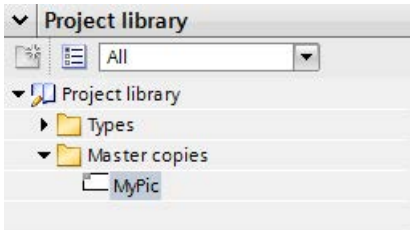
Procedure

1. Select the object in the work area of the "Screens" editor.
2. Drag-and-drop the object from the work area to the desired library.
   The mouse pointer is transformed into a crosshair with an appended object icon.



Result

The object is saved to the library for further use in multiple instances of your configuration.



Inserting a library object

Introduction

The system always assigns the inserted library object a name which consists of the name of the object type and of a consecutive number.

If the inserted object already exists, you have the option of replacing the object or of saving it under a new name.

You cannot insert library objects that are not supported by the HMI device.

Note

If you insert a screen with interconnected template from the library, the template will also be inserted. Any existing matching template is not used.

Requirement

- The "Libraries" task card is opened.
- The editor in which you want to insert the library object is open.

Procedure

1. Select a library object from the library.
2. Drag-and-drop the library object to the position in the work area where you want to insert the object.
   The library object is inserted.

Result

If the object was contained in the "Copy templates" folder, you have inserted an independent copy of the library object in the editor.

If the object was contained in the "Types" folder, you have inserted an instance of the library object in the editor.

10.9.2.7 Using types and their versions

State of type versions

Introduction

Depending on the point of use, the version of a type has different statuses.

Released version

The "released version" status is available for all types, regardless of the point of use.

If you want to edit a released version, you must first create a new test version or an "in progress" version.

Released type versions of scripts and screens can be opened and viewed at their instance.

"In progress" version

The "in progress" status is possible for the following types:

- Faceplates
- User data
- Styles

When you create a new type or a new version of a released type, the type is assigned the status "in progress".

Types with the "in progress" status can be edited in the library view without the need for a reference to an instance in the project. Upon release, the compatibility of the type is tested by a consistency check.

"In testing" version

Only versions of scripts and HMI screens have the "in testing" status.

When you create a new version of a type, the type is assigned the status "in testing".

A version with "in testing" status is linked to an instance in the project. You can set only one version to "in testing" for each type at a given time.

A version in testing may only be linked to a single instance in the project. Therefore, it is not possible to copy an instance to the clipboard, to duplicate it or to create an additional type from the instance as long as it has "in testing" status.

See also

Generating a script as a type (Page 8780)

Generating a screen as a type (Page 8781)

Create a new version of a type (Page 8782)

Copy templates and types (Page 8765)

Generating a script as a type

Requirement

- A project is open.
- An HMI device has been created and opened.
- The project tree is open.
- The "Libraries" task card is opened.

Procedure

1. Open the "Scripts" editor in the project tree.
2. Create a new script.

3. Select the script in the project tree.
4. Drag-and-drop the script into a library in the "Libraries" task card. A dialog opens.
5. Enter a name.
6. Enter a comment.

Result

You have created a type version of a script in the library. The created type is stored as a released version in the library. An instance of the type is used in the project.

To change the script create a new version of the script.

See also

State of type versions (Page 8779)

Generating a screen as a type

Requirement

A project is open.

An HMI device has been created and opened.

The project tree is open.

The "Libraries" task card is opened.

Procedure

1. Open the "Screens" editor in the project tree.
2. Create a new screen.
3. Select the screen in the project tree.
4. Drag-and-drop the screen into a library in the "Libraries" task card. A dialog opens.
5. Enter a name.
6. Enter a comment.

Result

You have created a type of a screen in the library.

The created type is stored as a released version in the library. An instance of the type is used in the project.

To change the screen create a new version of the screen.

See also

State of type versions (Page 8779)

Generating a style as a type

Introduction

To define a new style, add a new type in the project library.

Adding a new style

1. Open the "Libraries" task card.
2. Select the "Add new type" command under "Types" in the shortcut menu of the project library. A dialog opens.
3. Select "HMI style".

Result

The new style is created and displayed under its selected name in the project library. The HMI style type is assigned the status "in progress" and the version 0.0.1.

Create a new version of a type

Principle

If you create a new version of a type, the point of use of the type determines the status of the newly created version.

Requirement

The "Libraries" task card is opened.

A type has been created and released.

Procedure

1. Select the released type.
2. Select "Edit type" in the shortcut menu.

Result for types of faceplates, user data, and styles

A new version of the type is created.

The version has the status "in progress". The library view opens.

Result for types of scripts and screens

A dialog opens.

After you have selected the settings in the dialog, the version is set to the status "in testing". The instance used in the project is set to the status "in testing". The library view opens.

See also

State of type versions (Page 8779)

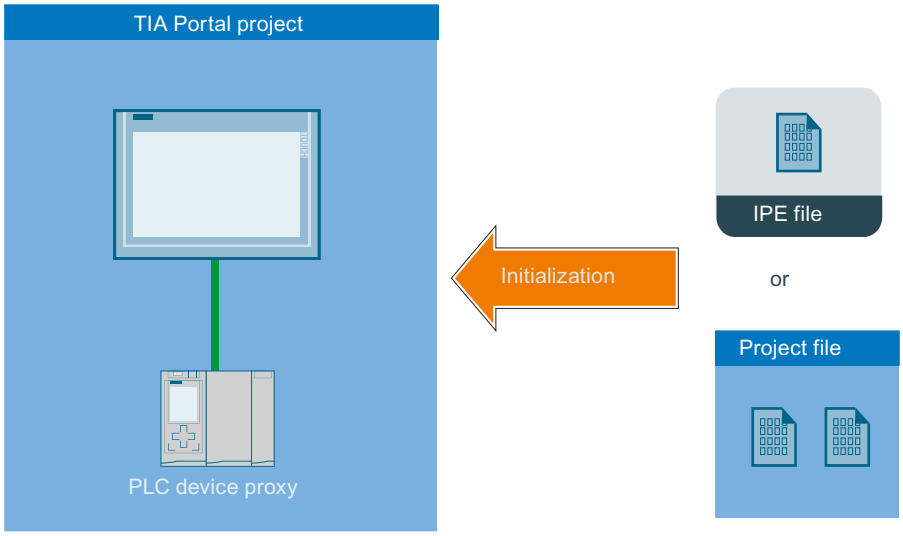
10.9.3 Using controller data from other projects

10.9.3.1 Basics of Inter Project Engineering (IPE)

What is IPE?

You can use the IPE (Inter Project Engineering) function to read in controller data from other TIA Portal projects and use it for configuring.

IPE enables you to transfer and merge the HMI configuration and PLC programming communication into various TIA Portal projects.



You can exchange controller data using IPE in two ways:

- Initializing device proxy via an IPE file (Page 8784)
- Initializing a device proxy via a project file (Page 8790)

10.9.3.2 Controller data

Introduction

In the source project of the PLC, you can select which controller data is to be stored as device-proxy data.

To this end, you create a device proxy data from a configured PLC.

Which device proxy data can be exchanged?

The following device proxy data can be exchanged via IPE:

- Program blocks
- Technology objects
- PLC tags
- PLC alarms

The following information is exchanged automatically:

- Interfaces of the PLC
- Configured communication processors and communication modules at the PLC

10.9.3.3 Initializing a device proxy via an IPE file

Initializing device proxy via an IPE file

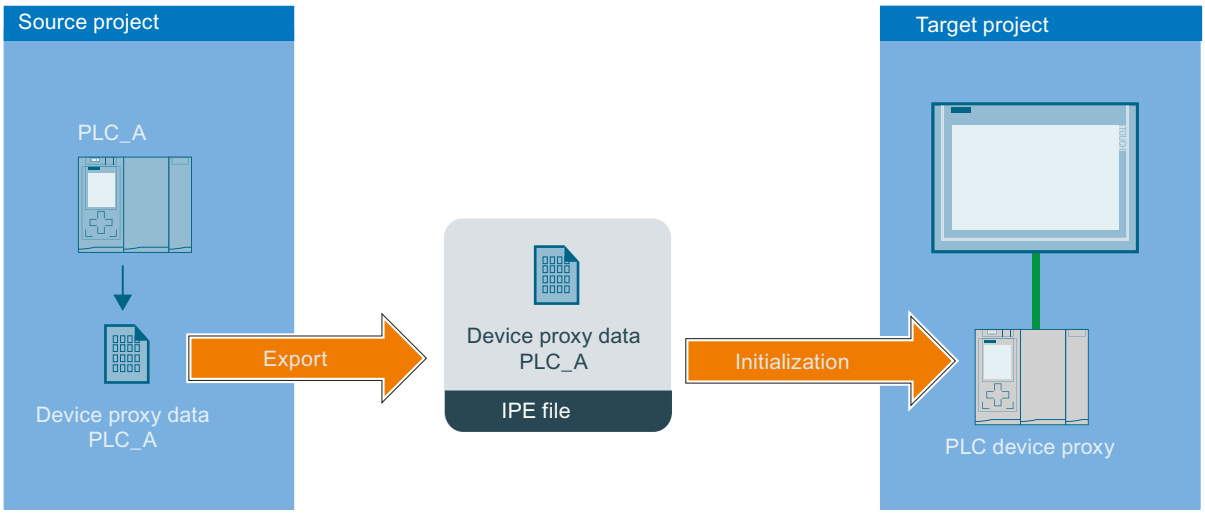
How does initialization of a device proxy via an IPE file work?

You can exchange controller data between two or more projects using IPE files.

Due to their small data volume, you can send IPE files by e-mail, for example.

You can create multiple IPE files with different device proxy data for each PLC.

A device proxy can only be filled with one device proxy data record in the target project.



- An IPE file is exported from the source project.
  The IPE file contains device proxy data of the PLC "PLC_A".
- A device proxy is created in the target project.
- In the target project, the device proxy is then initialized with the device proxy data from the IPE file.
- Following initialization, all device proxy data from the IPE file is contained in the device proxy.
- If changes are made in the source project, the device proxy of PLC_A in the target project can be updated via the new IPE file.

Initializing a device proxy via an IPE file

Introduction

You initialize device proxy data using a dialog on the device proxy in the TIA Portal.

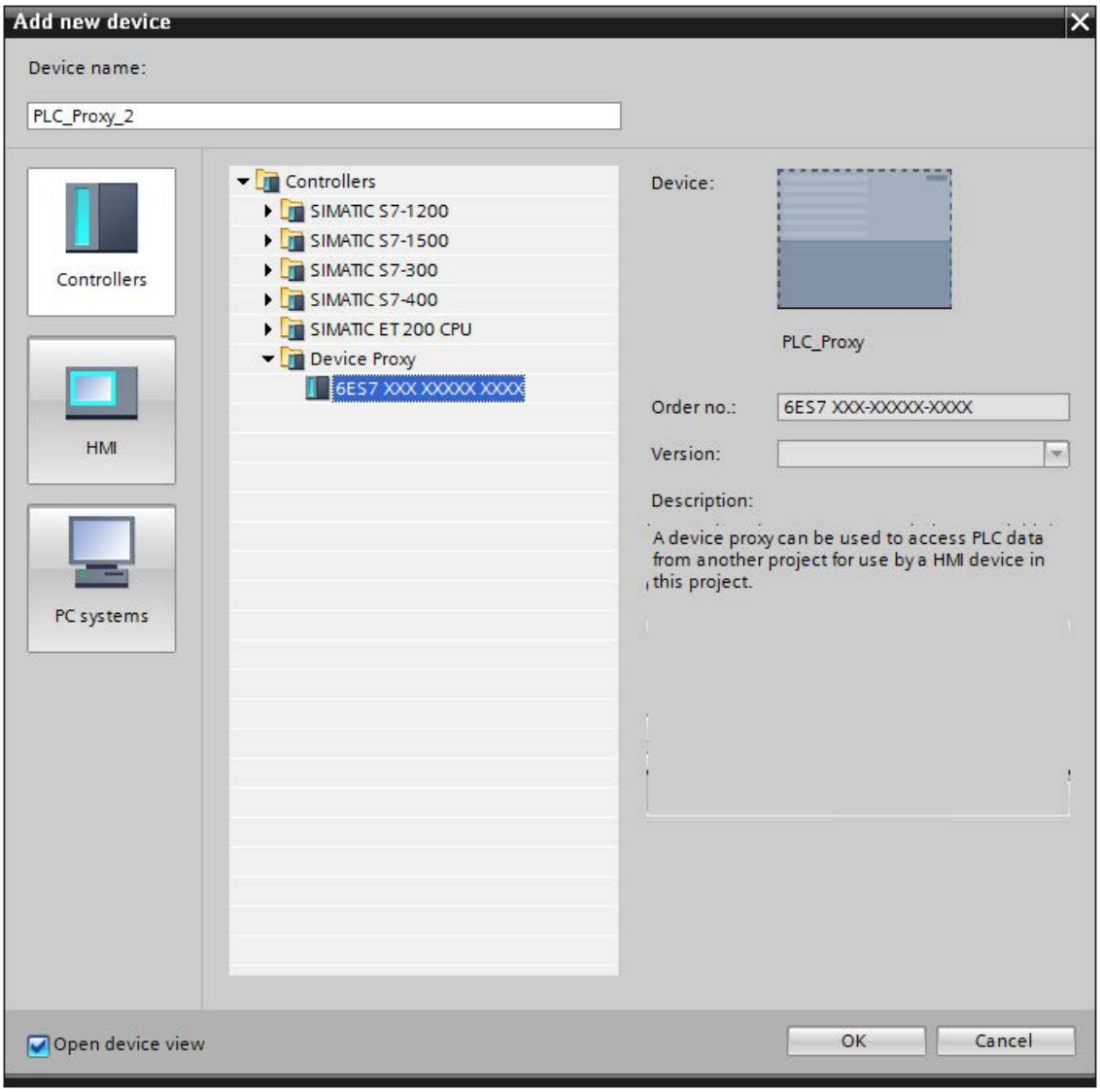
The IPE file contains the device proxy data from the source project.

Requirement

IPE file is available.

Procedure

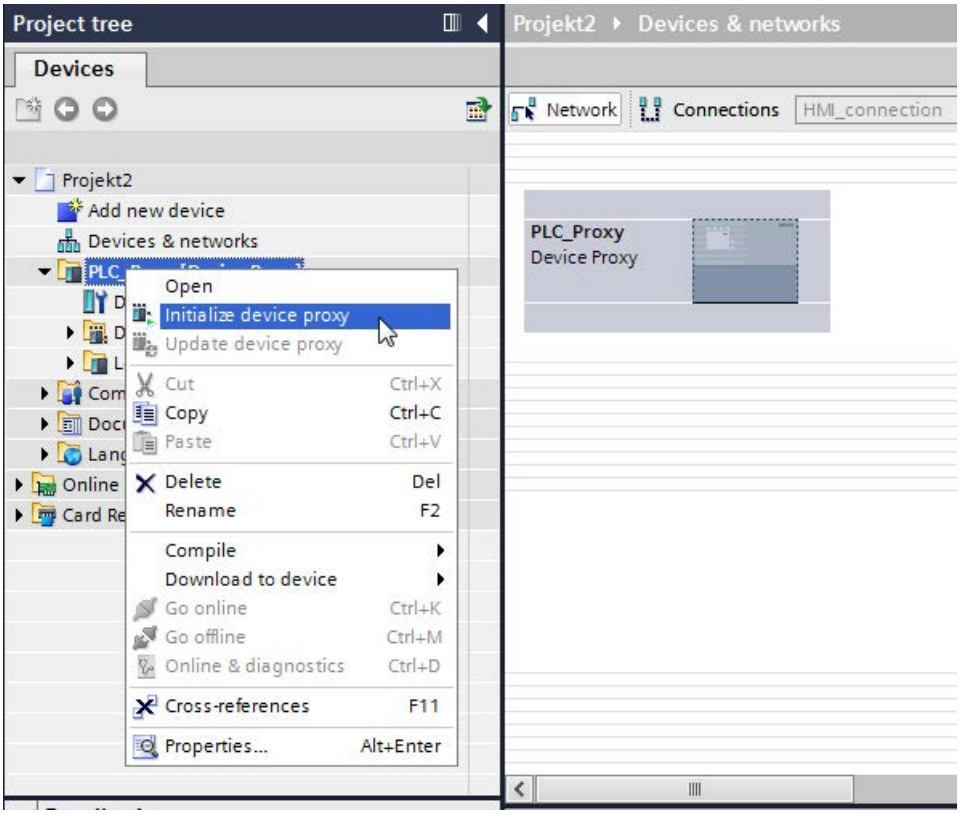
1. Double-click "Add new device" in the project tree.
2. Select the device proxy under "Controller".



A new device is created in the "Devices & Networks" editor.

3. Select the device proxy in the project tree.

4. Click "Initialize device proxy" in the shortcut menu.



5. Select the IPE file and click "Open".
   The "Initialize device proxy" dialog opens.

Note

You cannot select the device proxy data before initialization in the target project.

All device proxy data included in the IPE file is transferred.

Initialize a project file if you want to select specific device proxy data before initialization: Initializing a device proxy via a project file (Page 8790)

6. Click "OK".
   Initialization of the device proxy is started.

Result

Following initialization, all device proxy data from the IPE file is contained in the device proxy.

You can now configure an HMI connection with the device proxy and connect PLC tags from the device proxy with HMI tags, for example.

Updating a device proxy via an IPE file

Introduction

If changes were made to the device proxy data in the device proxy source project, the device proxy data can be updated in the target project.

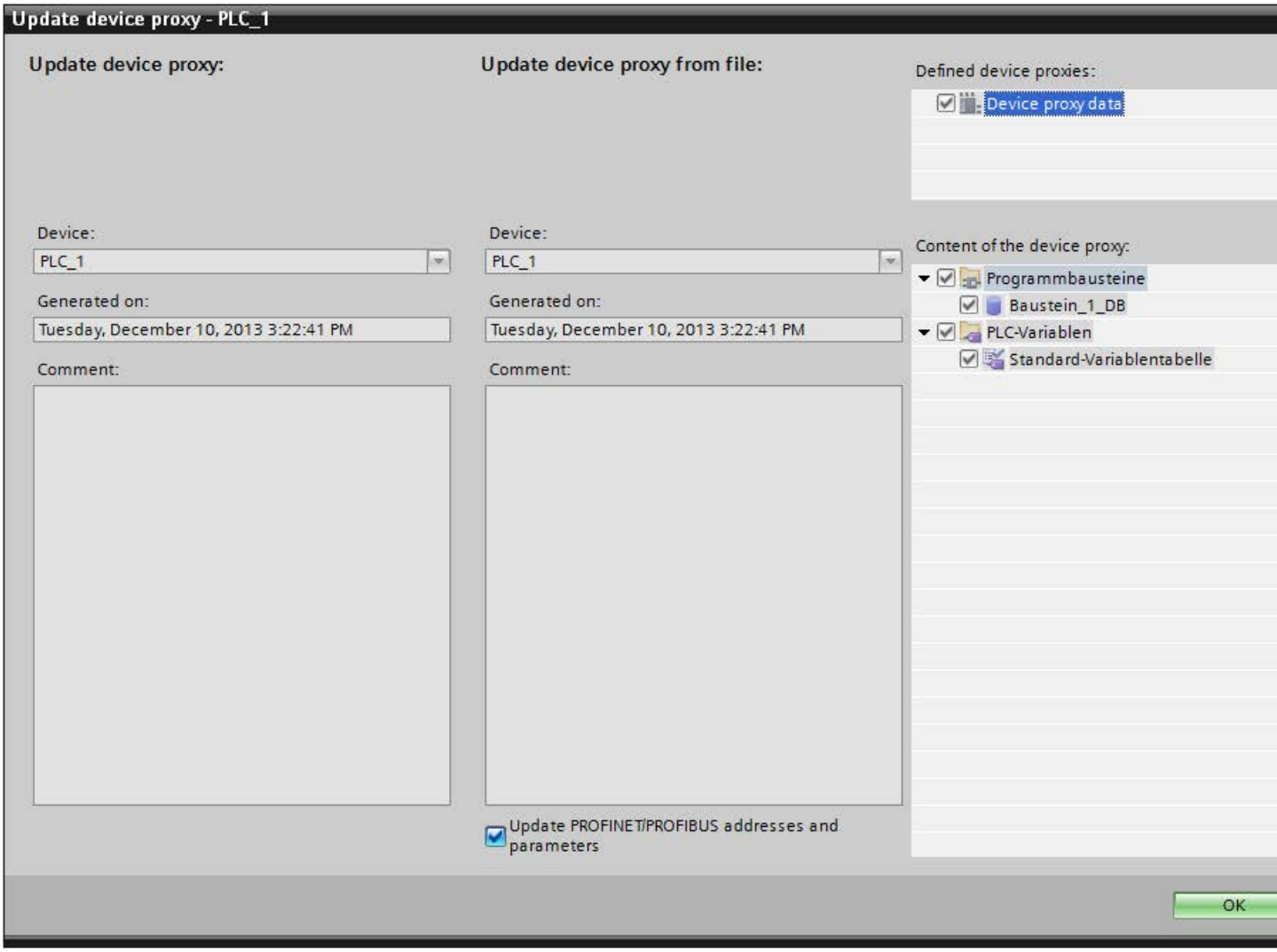
Requirement

- New IPE file contains the device proxy data from the source project.
- A device proxy has already been initialized using an IPE file in the target project.

Procedure

1. Select the device proxy in the project tree.
2. Click "Update device proxy" in the shortcut menu.

3. Select the IPE file.



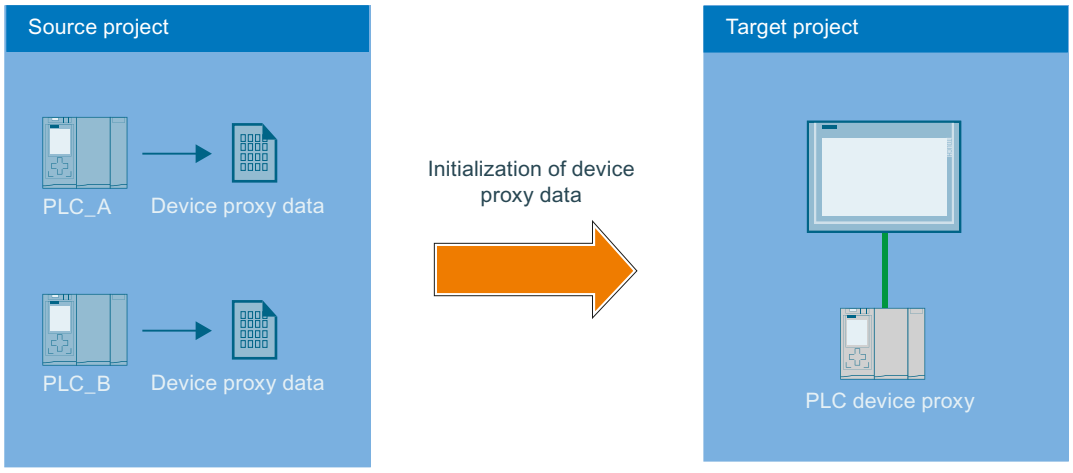
4. Select whether you wish to update the PROFINET or PROFIBUS addresses.
5. Click "OK".

10.9.3.4 Initializing a device proxy via a project file

Initializing a device proxy via a project file

How does an initialization via project file work?

You can transfer controller data using a project file in your TIA Portal project:



- There are two PLCs in the source project.
- Device proxy data objects of the two PLCs are created in the source project.
- The source project is saved.
- A device proxy is created in the target project.
- The source project (*.ap13) is first selected using the command for the initialization of device proxy in the device shortcut menu.
- The "PLC_A" and "PLC_B" controllers are available for selection and their device proxy data have already been created.
- After selection of device proxy data of "PLC_A", the data is stored in the PLC_Proxy of the target project.
- Changes from the source project can be transferred by an update via the project file.

Initializing a device proxy via a project file

Introduction

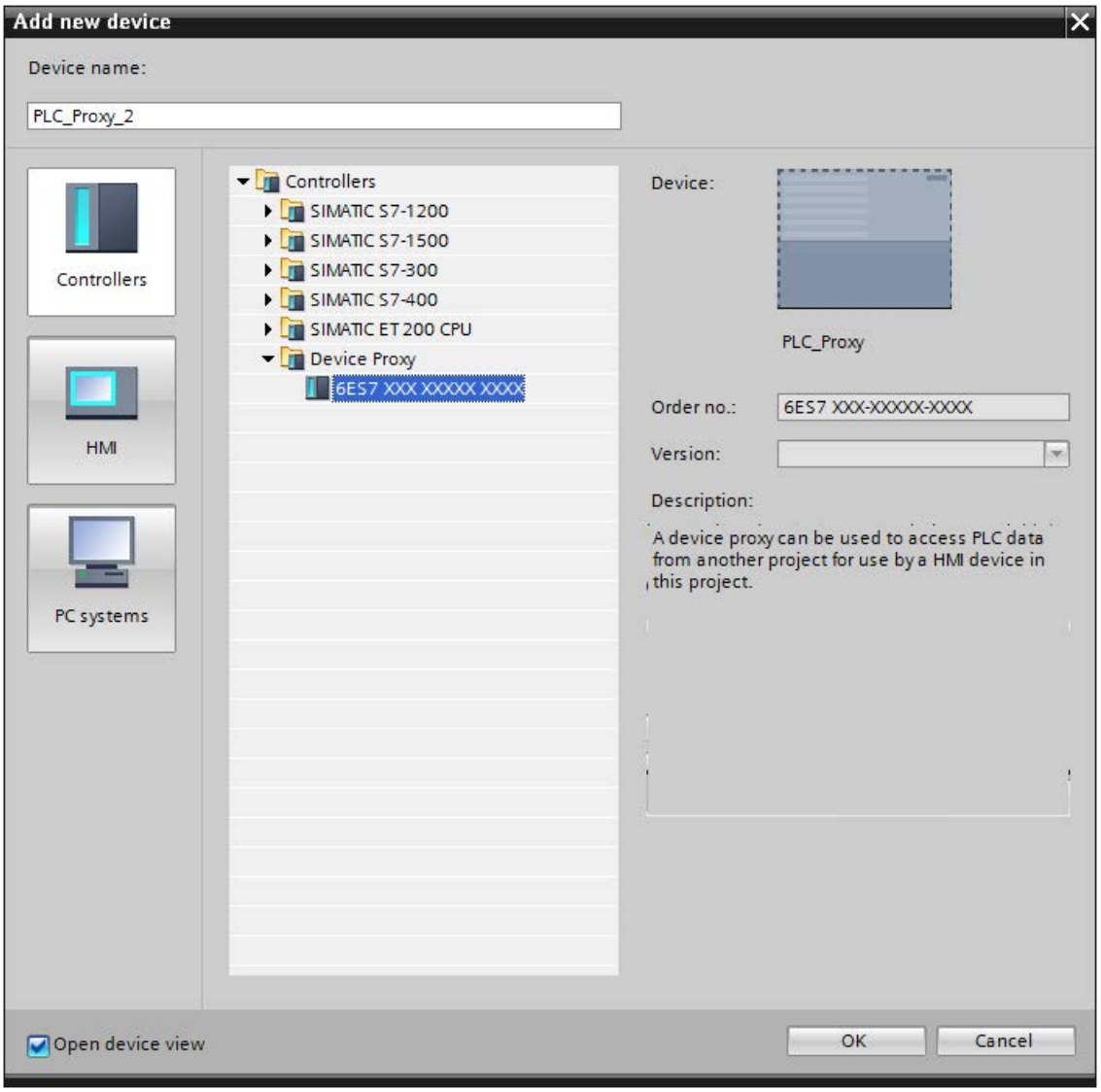
You initialize the device proxy using a project file.

Requirement

Project file (*.ap13) exists.

Procedure

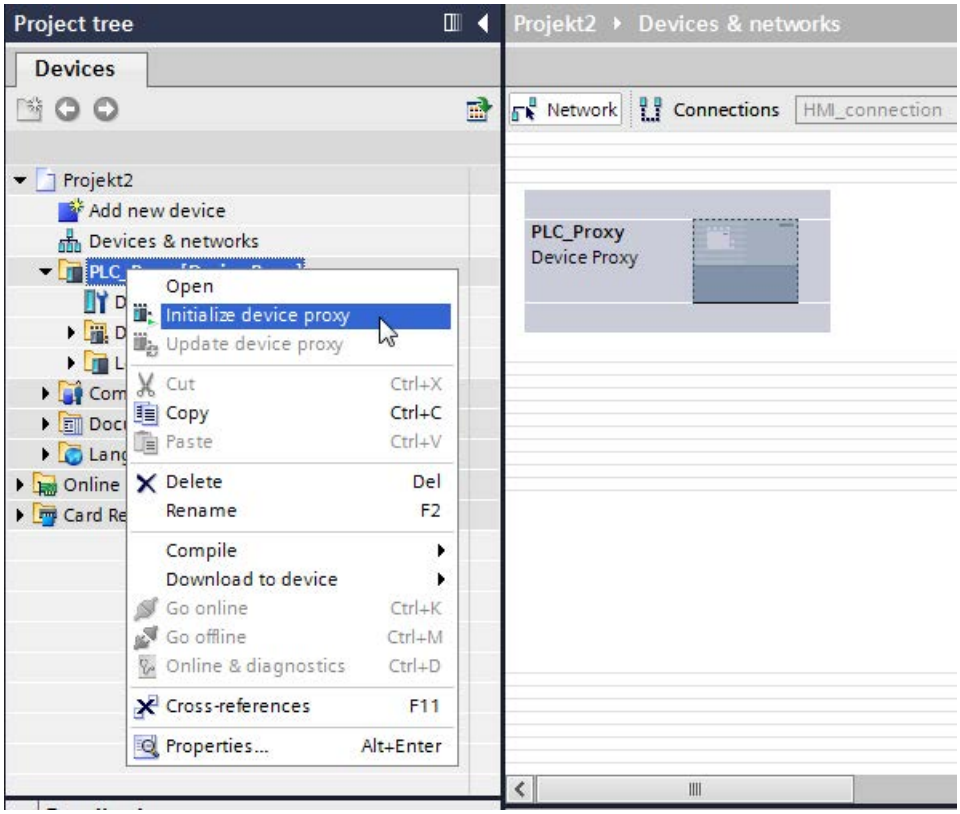
1. Double-click "Add new device" in the project tree.
2. Select the device proxy under "Controller".



A new device is created in the "Devices & Networks" editor.

3. Select the device proxy in the project tree.

4. Click "Initialize device proxy" in the shortcut menu.



5. Select the following entry in the "Open device proxy data source" dialog:
   "TIA Portal projects (*.ap13)"
6. Select a project file and click "Open".
   The "Initialize device proxy" dialog opens.
7. Select a device proxy data object already created from an available PLC to initialize the device proxy.
8. Click "OK".

Result

Following initialization, the controller data from the project file in the selected device proxy data object is stored in the device proxy.

You can now configure an HMI connection with the device proxy and connect PLC tags from the device proxy with HMI tags, for example.

Initialization via Classic projects

You can find more information on importing controller data from projects prior to TIA Portal V11 in an FAQ.

Entry ID: 73502293 (http://support.automation.siemens.com/WW/view/de/73502293)

Updating a device proxy via a project file

Introduction

If changes have been made to the controller data in the source project of the device proxy, you can update the device proxy in your TIA Portal project.

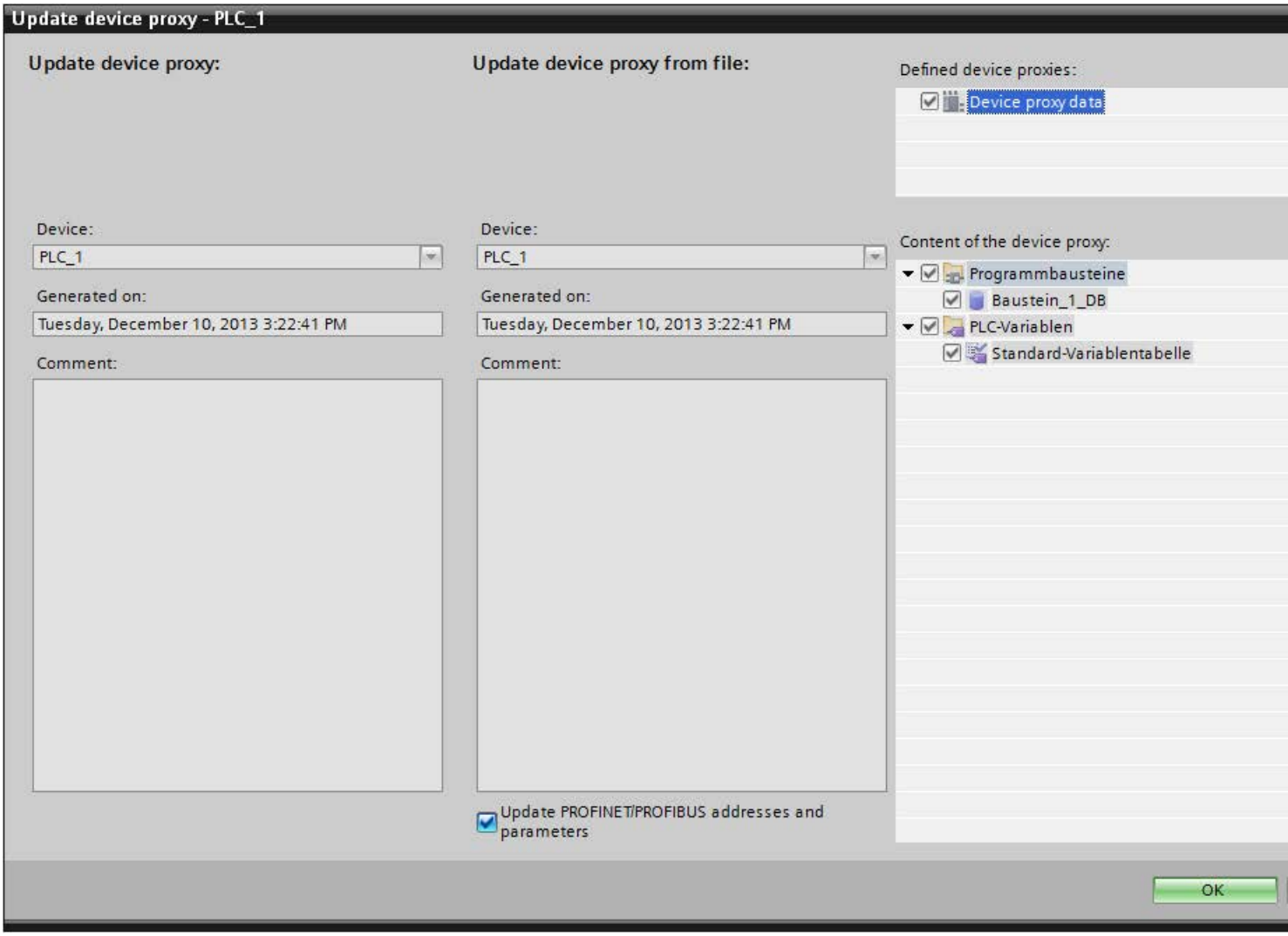
Requirement

- Project file has been generated from the source project of the device proxy.
- A device proxy that has already been initialized is available in the target project.

Procedure

1. Select the device proxy in the project tree.
2. Click "Update device proxy" in the shortcut menu.

3. Select the project file.
   The "Update device proxy" dialog opens.



4. Select a device.

Note

If you have created several device proxy data objects in the source project, you need to select one device proxy data object.

Note

Already initialized device proxies cannot be overwritten with controller data from a different PLC.

5. Select whether you wish to update the PROFINET or PROFIBUS addresses.
6. Click "OK".

10.9.4 Importing and exporting project data

10.9.4.1 Importing and exporting project data

Introduction

WinCC gives you the option of exchanging project data between the projects or copying them to external applications.

Exporting and importing between projects

You can export the following project data from a project and import them into another project.

- Recipe data records
- Alarms
- Tags
- Text lists
- Project texts

Exporting and importing reduces the workload. Instead of creating new data records, you use data already created in previous projects.

Editing the export file

The following file formats are available for export and import depending on the editor:

- *.xlsx for alarms, tags, project texts and text lists
- *.csv for recipe data records

You can edit the import file in Excel, for example.

XLSX file format

XLSX format is a file format for Excel tables based on the Open XML format. XLSX files are optimized for Microsoft Excel 2007.

You can sort the columns as required in the XLSX file.

CSV file format

CSV stands for Comma Separated Value. In this format, the columns of the table that contain the names and the value of the entry are separated by semicolons. Each table row terminates with a line break. You can also open the CSV file for editing in Excel.

Importing project data

When project data is imported, the objects in the project are created.

The syntax of the import file is checked during import. The accuracy of the values imported and dependencies between the imported values are not checked.

Any errors found in the imported data are reported when the project is compiled.

Copying in Excel format

In all spreadsheet editors you can copy the content in Excel format into the buffer of your PC. Subsequently you insert the project data in Excel format directly into any application outside the TIA Portal. To this purpose you use the corresponding command in the shortcut menu of the work area:

- If you select the command in the shortcut menu of the line header, the complete line is copied into the buffer.
- If you select the command in the shortcut menu of a cell, only the cell content is copied into the buffer.
- If you select several lines and select the command, all marked data are always copied into the buffer.

This data exchange is only possible as an export.

See also

Exporting alarms (Page 8800)

Exporting tags (Page 8807)

Exporting text lists (Page 8812)

10.9.4.2 Importing and exporting recipes

Exporting recipes

Introduction

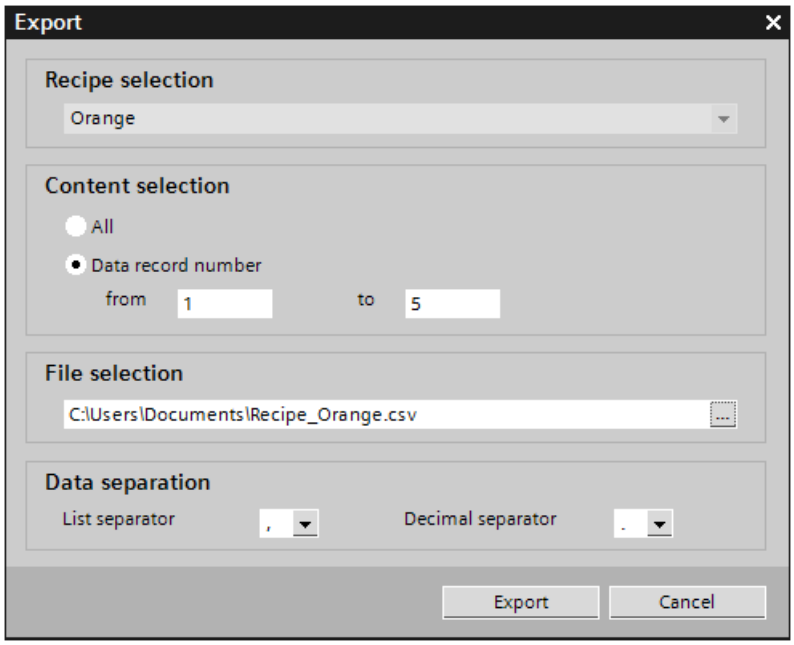
WinCC features an export function for exporting data records from recipes.

Requirements

- The WinCC project for the export is open.
- Recipes have been created in a project.
- The "Recipes" editor is open.

Exporting recipes

1. In the "Recipes" editor, select the recipe with the data records you want to export.
2. Click .
   The "Export" dialog box opens.



   The selected recipe is shown under "Recipe selection".
3. Under "Content selection", specify if all or only selected data records are to be exported.
4. Under "File selection", specify the file in which the recipe data is to be stored.
5. Specify the list separator and decimal separator under "Data separation".
6. Click "Export."
   The export will start.

Result

The exported data has been written to a CSV file. The CSV file will be stored in the specified directory.

See also

Exporting alarms (Page 8800)

Importing and exporting project data (Page 8795)

Exporting tags (Page 8807)

Exporting text lists (Page 8812)

Importing recipes

Introduction

Recipes are identified by their name. The recipe name must therefore be unique. Open the import file in a simple text editor to check that it has the correct data structure.

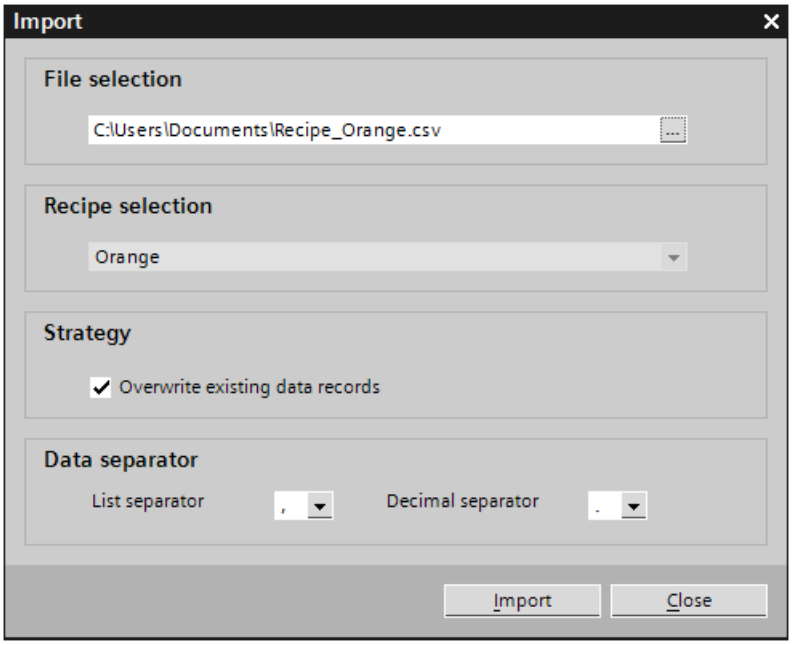
Specify whether or not existing data records should be overwritten by records with the same name during the import.

Requirements

- A CSV file containing at least one recipe has been created.
- The WinCC project for the import is open.
- The "Recipes" editor is open with at least one recipe.

Importing a recipe

1. In the "Recipes" editor, select the recipe with the data records you want to import.
2. Click .
   The "Import" dialog box opens.



   The selected recipe is shown under "Recipe selection".
3. Select the file you want to import under "File selection".
4. Under "Strategy", specify if existing data records should be overwritten by records of the same name.

5. Under "Data separation", select the list separator and the decimal separator to use in the CSV file.
6. Click "Import".
   The import will start.

Result

The data records are created in the selected recipe. Depending on the setting for "Strategy", existing data records are overwritten by records with the same name from the CSV file.

Existing data records with the same name will also be imported from the CSV file if you deactivate the "Overwrite existing data records" option.

Format of recipe data

Introduction

This section describes the required format of the file for the import of recipes. The file containing the data of the recipes must be available in "*.csv" format. :

Structure of recipe data

The structure of the import file is fixed. The following example shows the structure of a recipe containing two recipe elements, each with two data records:

```
List separator=<List separator>Decimal symbol=<Decimal
separator><List separator><Line break>
<Name of the recipe><List separator><List separator><Line break>
LANGID_<ID of the language><List separator>
<Display name, recipe element 1><List separator>
<Display name, recipe element 2><Line break>
<Number recipe><List separator>
<Recipe data record number 1><List separator>
<Recipe data record number 2><Line break>
<Tag recipe element 1><List separator>
<Recipe data record 1 value 1><List separator>
<Recipe data record 2 value 1><Line break>
<Tag recipe element 2><List separator>
<Recipe data record 1 value 2><List separator>
<Recipe data record 2 value 2><Line break>
```

ID of the language

Use the "Windows language ID" in decimal notation, e.g. "1033" for English. Additional information is available in the documentation for the Windows operating system.

See also

Exporting recipes (Page 8796)

10.9.4.3 Importing and exporting alarms

Exporting alarms

Introduction

WinCC has an export function for alarms.

Requirements

- The WinCC project for the export is open.
- Alarms have been created in the project.
- The "HMI alarms" editor is open.

Exporting alarms

1. Click the button in "Discrete alarms" or "Analog alarms".
   The "Export" dialog box opens.

2. Click the "..." button and specify in which file the data are saved.
3. Specify whether you want to export "Disrete alarms" or "Analog alarms".
4. Click "Export". The export will start.

Result

The exported data has been written to an xlsx file. The xlsx file will be stored in the specified folder.

See also

Importing alarms (Page 8801)

Format of the analog alarm data (Page 8802)

Format of the discrete alarm data (Page 8805)

Exporting recipes (Page 8796)

Exporting tags (Page 8807)

Importing and exporting project data (Page 8795)

Importing alarms

Introduction

Alarms are identified by their alarm number. The alarm numbers must be unique in the analog and discrete alarm types. Alarms with redundant alarm numbers will be overwritten. An alarm without an existing alarm number is created.

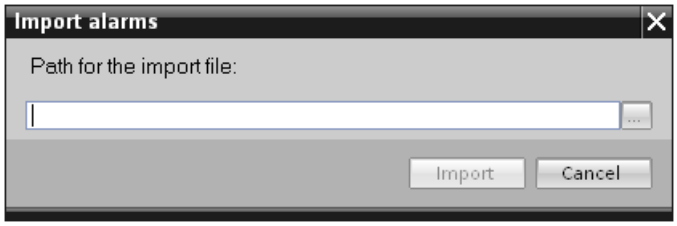
Any empty list entries for existing alarms contained in an xlsx file will be ignored for the purposes of the import. The entries of the existing alarms remain active and will not be replaced by empty ones.

Requirements

- An xlsx file with alarms has been created.
- The structure of the xlsx file meets the requirements.
- The WinCC project for the import is open.
- The "HMI alarms" editor is open.

Importing alarms

1. Click the button in "Discrete alarms" or "Analog alarms". The "Import" dialog box opens.



2. Click the "..." button and select the file that you want to import.
3. Click on the "Import" button. The import will start. A progress bar indicates the progress of the import operation.

Result

The corresponding alarms including alarm texts are created in WinCC on the basis of the import data. Alarms relating to the import operation are displayed in the output window. A log file is saved in the source directory of the import files. The log file has the same name as the respective import file but with the "*.xml" extension.

Check when importing the data whether there are any links to objects, for example, dynamic parameters such as tags.

- If an object with the same name already exists, the existing object is used.
- If no object of the same name yet exists, create an object with the relevant name or create a new link.

Note

The syntax of the import file is checked during xlsx file import. The meaning of the properties or dependencies between the properties is not checked. It is possible to assign a trigger tag of an incorrect type, such as string, to an alarm. An error will be reported during compilation.

See also

Exporting alarms (Page 8800)

Format of the analog alarm data

Introduction

This chapter describes the required format of the file for the import of analog alarms. The file containing the analog alarm data must be in "*.xlsx" format.

Structure of the alarm data

The import file in Microsoft Excel consists of a number of worksheets:

- Analog alarms(Analog alarms)
- Limits (Limits)

Each alarm is assigned a separate row in the import file. The import file with the analog alarms must be formatted as follows:

Example of the worksheet "Analog alarms"

	A	B	C	D	E
1	ID	Name	Event text [en-US], Alarm text	FieldInfo [Alarm text]	Class
2	1	Analog_alarm_1	AA1 Error-AC with maximum text length: abcdefghijklmnopqrstuvwxyz		Errors
3	2	Analog_alarm_2	AA2 Warning-AC this text should be bold		Warnings
4	3	Analog_alarm_3	AA3 SDm-AC this text should be italic		SDm
5	4	Analog_alarm_4	AA4 SDo-AC <u>this text should be underlined</u>		SDo
6	5	Analog_alarm_5	AA5 SystemAcknowledgement-AC <blink>this text should be flashing<		System_Ackn
7	25	Analog_alarm_25	Internal AA23 switchDT: Deadband mode in case of violation - value HL		AADT-Interna
8	26	Analog_alarm_26	Internal AA23 switchDT: Deadband mode in case of violation - percent		AADT-Interna
9	31	Analog_alarm_31	Internal AA23 switchDT: Low limit violation static		AADT-Interna
10	32	Analog_alarm_32	Internal AA23 switchDT: High limit violation static		AADT-Interna
11	33	Analog_alarm_33	Internal AA23 switchDT: Low limit violation dynamic		AADT-Interna
12	34	Analog_alarm_34	Internal AA23 switchDT: High limit violation dynamic		AADT-Interna
13	35	Analog_alarm_35	Internal AA23 switchDT: delay 3 seconds High limit violation		AADT-Interna
14	42	Analog_alarm_40	Internal AA23 switchDT: delay 3 seconds Low limit violation LLV		AADT-Interna
15	23	Analog_alarm_41	AA23 DT in event text: <field ref="0" /> Bool, <field ref="1" /> Byte, <fi	<ref id = 0; type = AlarmTag; T	ACAT
16	24	Analog_alarm_42	AA24 DT in event text: <field ref="0" /> Timer, <field ref="1" /> Counte	<ref id = 0; type = AlarmTag; T	ACAT

Table 10-13 Meaning of the entries

List entry	Meaning
ID	The alarm number is used to reference an alarm. The alarm number is unique. Alarms with identical alarm numbers are overwritten during import. An alarm without an existing alarm number is created.
Name	Name of the analog alarm
Event text [de-DE], Alarm text	Displays the alarm text. The field designation contains a language ID. Alarm texts must be assigned a language ID for import. An expression with a reference ID will be added to the alarm text if the text has a dynamic parameter. Example: text <field ref="0" />. Use the ID to assign dynamic parameters to alarm texts.
FeldInfo	Specifies whether the alarm text contains dynamic parameters. The settings are separated by a semicolon ";". Example of dynamic parameters: Tag: <ref id = 0; type = AlarmTag; Tag = Tag1; DisplayType = Decimal; Length = 5;> Text list: <ref id = 1; type = CommonTextList; TextList = Textlist1; Tag = tag 2; Length = 5;>
Class	The class of an alarm determines whether or not the alarm must be acknowledged. It can also be used to determine how the alarm appears when it is displayed on the HMI device. The alarm class also determines whether and where the corresponding alarm will be logged.
Group	Indicates the allocation to an alarm group. If an alarm belongs to a group with other alarms, it can be acknowledged together with these alarms of the same group in a single operation.
Trigger tag	Specifies the tag monitored for limit value violation.
Delay time value	Specifies the delay time. The alarm is not triggered until the duration of the limit value violation equals the specified delay time.

List entry	Meaning
Delay time unit	Specifies the time unit for the delay.
Report	Enables reporting of the specific alarm on a printer. True or "1" = Reporting enabled. False or "0" = Reporting disabled. Reporting must also be globally enabled in the project.
Info text [de-DE], Info text	The tooltip is an optional property of an alarm. Tooltips can contain additional information about the alarm. A tooltip will be displayed in a separate window on the HMI device when the operator presses the <HELP> key. The field designation contains a language ID.

Example of the worksheet "Limits"

	A	B	C	D	E	F	G
1	Alarm ID	Limit type	Limit value	Limit mode	Deadband mo	Deadband valu	Deadband in percent
2	1	Constant	0	Upper limit	Off	0	False
3	2	Constant	1	Upper limit	Off	0	False
4	3	Constant	2	Upper limit	Off	0	False
5	4	Constant	3	Upper limit	Off	0	False
6	5	Constant	4	Upper limit	Off	0	False
7	25	Constant	50	Upper limit	On both	5	False
8	26	Constant	50	Upper limit	On both	10	True
9	31	Constant	50	Lower limit	Off	0	False
10	32	Constant	50	Upper limit	Off	0	False
11	33	Tag	AASDTdyn	Lower limit	Off	0	False
12	34	Tag	AASDT1dyn	Upper limit	Off	0	False
13	35	Constant	50	Upper limit	Off	0	False
14	36	Constant	50	Lower limit	On both	5	False

Table 10-14 Meaning of the entries

List entry	Meaning
Alarm ID	Alarm number The alarm number is used to reference an alarm. The alarm number is unique. Alarms with identical alarm numbers are overwritten during import. An alarm without an existing alarm number is created.
Limit mode	Trigger mode Indicates the method used for monitoring the limit value.
Limit type	Specifies the limit that will be monitored. Both a tag and a constant can be used as limit value.
Limit value	Limit value Indicates the tag or constant monitored for limit violation.

List entry	Meaning
Deadband mode	Hysteresis mode Specifies whether and in which cases hysteresis will be used. For "Outgoing" For "Incoming" For "Incoming" and "Outgoing"
Deadband in percent	0 = The value specified for "Hysteresis" is considered to be absolute. 1 = The value specified for "Hysteresis" is referred to as a percentage of the limit value.
Deadband mode	Hysteresis Specifies a constant as a value of the hysteresis.

Note

"No value" in the table

Entries in the table which have the value "No value" delete the corresponding values in an existing alarm of the same name.

See also

Exporting alarms (Page 8800)

Format of the discrete alarm data

Introduction

This chapter describes the required format of the file for the import of discrete alarms. The file containing the discrete alarm data must be in "*.xlsx" format.

Structure of the alarm data

The import file in Microsoft Excel consists of the worksheets "Discrete alarms" (discrete alarms). Each alarm is assigned a separate row in the import file. Structure of the import file containing the discrete alarms:

Example of the worksheet "Discrete alarms"

	A	B	C	D	E	F
1	ID	Name	Event text [en-US], Alarm text	FieldInfo [Alarm text]	Class	Trigger t
2	1	Discrete_alarm_1	DA1 Error-AC with maximum text length: abcdefghijklmnopqrstuvwxyz äöü\na		Errors	HMI_AC
3	2	Discrete_alarm_2	DA2 Warning-AC this text should be bold		Warnings	HMI_AC
4	3	Discrete_alarm_3	DA3 SDm-AC this text should be italic		SDm	HMI_AC
5	4	Discrete_alarm_4	DA4 SDo-AC <u>this text should be underlined</u>		SDo	HMI_AC
6	5	Discrete_alarm_5	DA5 SystemAcknowledgement-AC <blink>this text should be flashing</blink		System_Ackn	HMI_AC
7	6	Discrete_alarm_6	DA6 SystemNoAcknowledgement-AC mixed test: Bold, Italic, <l		System_No_A	HMI_AC
8	7	Discrete_alarm_7	DA7 DT in event text: <field ref="0" /> Integer, <field ref="1" /> Real, <field re	<ref id = 0; type = AlarmTag; Tag = PL	ACAT	HMI_Tri
9	8	Discrete_alarm_8	DA8 DT in event text: <field ref="0" /> S5Time, <field ref="1" /> Timer, <field r	<ref id = 0; type = AlarmTag; Tag = PL	ACAT	HMI_Tri
10	11	Discrete_alarm_9	DA11 DT in event text: <field ref="0" /> Int, <field ref="1" /> Real, <field ref="2	<ref id = 0; type = AlarmTag; Tag = In	ACAT	HMI_Tri
11	12	Discrete_alarm_10	DA12 DT in event text: <field ref="0" /> UDInt, <field ref="1" /> UInt;	<ref id = 0; type = AlarmTag; Tag = U	ACAT	HMI_Tri
12	13	Discrete_alarm_11	DA13 Textformat: Integer: <field ref="0" /> decimal, <field ref="1" /> binary, <	<ref id = 0; type = AlarmTag; Tag = H	ACAT	HMI_TF

Table 10-15 Meaning of the entries

List entry	Meaning
ID	The alarm number is used to reference an alarm. The alarm number is unique. Alarms with identical alarm numbers are overwritten during import. An alarm without an existing alarm number is created.
Name	Name of the analog alarm
Event text [de-DE], Alarm text	Displays the alarm text. The field designation contains a language ID. For import, a language ID must be assigned to alarm text. An expression with a reference ID will be added to the alarm text if the text has a dynamic parameter. Example: text <field ref="0" />. Use the ID to assign dynamic parameters to alarm texts.
FeldInfo	Specifies whether the alarm text contains dynamic parameters. The settings are separated by a semicolon ";". Example of dynamic parameters: Tag: <ref id = 0; type = AlarmTag; Tag = Tag1; DisplayType = Decimal; Length = 5;> Text list: <ref id = 1; type = CommonTextList; TextList = Textlist1; Tag = tag 2; Length = 5;>
Class	The class of an alarm determines whether or not the alarm must be acknowledged. It can also be used to determine how the alarm appears when it is displayed on the HMI device. The alarm class also determines whether and where the corresponding alarm will be logged.
Group	Indicates the allocation to an alarm group. If an alarm belongs to a group with other alarms, it can be acknowledged together with these alarms of the same group in a single operation.
Trigger tag	Specifies the tag containing the bit that triggers the alarm.
Trigger bit	Specifies the number of the bit that triggers the alarm.
Acknowledge tag	Specifies the tag containing the bit that is set by the operator upon acknowledgment. Only available if the selected alarm class requires alarm acknowledgment.
Acknowledgment bit	Specifies the number of the bit that is set when the operator acknowledges the alarm.

List entry	Meaning
PLC acknowledgement tag	Specifies the tag containing the bit that acknowledges the alarm of the control program. Only available if the selected alarm class requires alarm acknowledgment.
PLC acknowledgment bit	Specifies the number of the bit that acknowledges the alarm of the control program.
Delay time value	Specifies the delay time. The alarm is not triggered until the duration of the limit value violation equals the specified delay time.
Delay time unit	Specifies the time unit for the delay.
Report	Enables reporting of the specific alarm on a printer. True or "1" = Reporting enabled. False or "0" = Reporting disabled. Reporting must also be globally enabled in the project.
Info text [de-DE], Info text	The tooltip is an optional property of an alarm. Tooltips can contain additional information about the alarm. A tooltip will be displayed in a separate window on the HMI device when the operator presses the <HELP> key. The field designation contains a language ID.

Note

"No value" in the table

Entries in the table which have the value "No value" delete the corresponding values in an existing alarm of the same name.

See also

Exporting alarms (Page 8800)

10.9.4.4 Importing and exporting tags

Exporting tags

Introduction

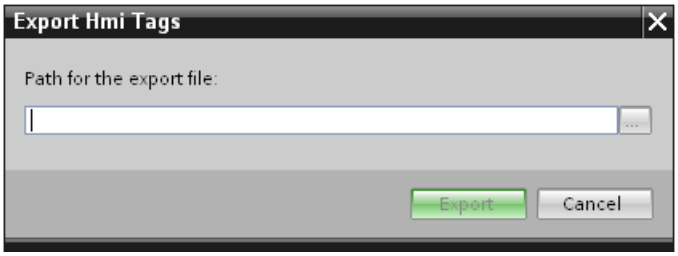
WinCC has an export function for tags.

Requirements

- The WinCC project for the export is open.
- Tags have been created in the project.
- The "HMI tags" editor is open.

Exporting tags

1. Click on the button in the "HMI Tags" tab.
   The "Export" dialog box opens.



2. Click the "..." button and specify in which file the data are saved.
3. Click "Export". The export will start.

Result

The exported data has been written to an xlsx file. The xlsx file will be stored in the specified folder.

See also

Importing tags (Page 8808)

Exporting alarms (Page 8800)

Importing and exporting project data (Page 8795)

Exporting text lists (Page 8812)

Exporting recipes (Page 8796)

Format of the tag data (Page 8810)

Importing tags

Introduction

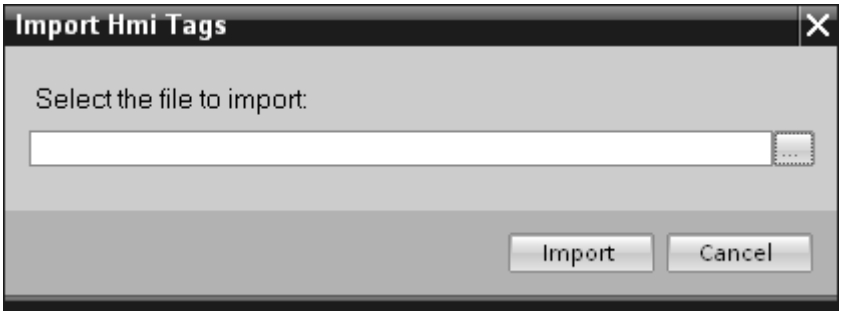
Tags are identified by the tag name. An existing tag will be overwritten with the data from the xlsx file if the tag name already exists in the project. A new tag is created if the tag does not yet exist.

Requirements

- An xlsx file with tags has been created.
- The structure of the xlsx file meets the requirements.
- The WinCC project for the import is open.

Importing tags

1. Click "HMI tags" in the project navigation.
2. Double-click "Show all tags". The "HMI tags" editor opens.
3. Click on the button. The "Import" dialog box opens.



4. Click the "..." button and select the file that you want to import.
5. Click on the "Import" button. The import will start.

Result

The relevant tags have been created in WinCC. Alarms relating to the import operation are displayed in the output window. A log file is saved in the source directory of the import files. The log file has the same name as the respective import file but with the "*.xml" extension.

Check when importing the data whether there are any links to objects, for example, dynamic parameters such as tags.

- If an object with the same name already exists, the existing object is used.
- If no object of the same name yet exists, create an object with the relevant name or create a new link.

Note

The syntax of the import file is checked during xlsx file import. The meaning of the properties or dependencies between the properties is not checked. It is possible to assign a tag a trigger tag of the wrong type, for example, string. An error will be reported during compilation.

See also

Exporting tags (Page 8807)

Format of the tag data

Introduction

This section describes the format required for the file with tag data used for imports. The tag data file must be in "*.xlsx" format.

Tag data structure

The import file in Microsoft Excel consists of a number of worksheets:

- HMI Tags (HMI tags)
- Multiplexing (multiplex tags)

Each tag is on a separate line in the import file. The import file with the tag data must have the following format:

Example of the worksheet "HMI Tags"

	A	B	C	D	E	F	G	H
1	Name	Path	Connection	PLC tag	DataType	Length	Address	Access Me
2	HMI_Int	Internal Tags	<No Value>	<No Value>	Int	2	<No Value>	<No Value
3	Mux_Tag_1	Internal Tags	<No Value>	<No Value>	Int	2	<No Value>	<No Value
4	Mux_Tag_2	Internal Tags	<No Value>	<No Value>	Int	2	<No Value>	<No Value
5	Mux_11	Internal Tags	<No Value>	<No Value>	Int	2	<No Value>	<No Value
6	Mux_21	Internal Tags	<No Value>	<No Value>	Int	2	<No Value>	<No Value
7	Mux_13	Internal Tags	<No Value>	<No Value>	Int	2	<No Value>	<No Value
8	Mux_12	Internal Tags	<No Value>	<No Value>	Int	2	<No Value>	<No Value
9	Mux_23	Internal Tags	<No Value>	<No Value>	Int	2	<No Value>	<No Value
10	Mux_22	Internal Tags	<No Value>	<No Value>	Int	2	<No Value>	<No Value
11	Mux_Tag_1_Index	Internal Tags	<No Value>	<No Value>	UInt	2	<No Value>	<No Value
12	Mux_Tag_12	Internal Tags	<No Value>	<No Value>	USInt	1	<No Value>	<No Value
13	Mux_Tag_11	Internal Tags	<No Value>	<No Value>	USInt	1	<No Value>	<No Value
14	Mux_Tag_13	Internal Tags	<No Value>	<No Value>	USInt	1	<No Value>	<No Value
15	HMI_UDInt	Internal Tags	<No Value>	<No Value>	UDInt	4	<No Value>	<No Value
16	Gauge_Process	Default tag table	<No Value>	<No Value>	Int	2	<No Value>	<No Value
17	Button_Tag_4	Default tag table	<No Value>	<No Value>	Int	2	<No Value>	<No Value
18	HMI_USInt	Internal Tags	<No Value>	<No Value>	USInt	1	<No Value>	<No Value
19	Data_block_2_PLC_DateTime_2	Default tag table	HMI_connection_1	Data_block_2.PLC_I	Date_And_Time	8	%DB28.DBX598.0	<absolute

Table 10-16 Meaning of the entries

List entry	Meaning
Name	Indicates the configured name of an HMI tag.
Path	Specifies which folders in the project tree contain the tag. The folder structure is represented by "\" : "FolderName1\FolderName2\TagName".
PLC Tag	Indicates whether the tag is linked to a PLC tag.
Connection	Indicates the name of the connection to the PLC.

List entry	Meaning
Data type	Specifies the data type of a tag. The data types allowed depend on the communication driver being used. See the "Communication" section of the documentation for additional information on the data types permitted for the various communication drivers.
Length	Specifies the length of the tag. This entry is only useful for data types with a dynamic length such as strings; it is left empty for all other data types.
Address	Specifies the tag address in the PLC. The tag address must exactly match the one used in WinCC, for example, "%DB1.DBW0". The tag address is empty for internal tags.
Multiplexing	Specifies whether multiplexing is used.
Index tag	Shows the name of the index tag for multiplexing. In Runtime, the system first reads the value of the index tag. It then accesses the tag in the corresponding place in the tag list.
StartValue	Specifies the start value of a tag.
ID tag	The update ID updates the value of a tag with the aid of a function or a PLC job. The update ID must be unique within an HMI device.
Coding	Shows the coding method.
DiplayName [de_DE]	Shows the display name of an HMI tag. The field designation contains a language ID. The field designation contains a language ID. Display names must be assigned a language ID for import. Texts are imported to the corresponding project language.
Acquisition mode	Specifies the tag acquisition mode.
Acquisition cycle	Specifies the tag acquisition cycle. The acquisition cycle must correspond exactly to the one used in WinCC. The value is not language-dependent and should therefore be the same in every language. The default value is "1 s". The acquisition cycle is undefined if the tag acquisition mode is "on demand". User-defined acquisition cycles must be created beforehand as the file will otherwise not be imported.
High High Limit type	Indicates whether the limit value "High high" is monitored by a constant, a tag or not at all.
High High Limit	Displays the limit value "High High".
High Limit type	Indicates whether the limit value "High" is monitored by a constant, a tag or not at all.
High Limit	Displays the limit value "High".
Low Limit type	Indicates whether the limit value "Low" is monitored by a constant, a tag or not at all.
Low Limit	Displays the limit value "Low".
Low Low Limit type	Indicates whether the limit value "Low Low" is monitored by a constant, a tag or not at all.
Low Low Limit	Displays the limit value "Low Low".
Linear scaling	Indicates whether linear scaling is enabled. This entry can only be used for external tags.
End value PLC	Specifies the end value of the PLC tag.
Start value PLC	Specifies the start value of the PLC tag.
End value HMI	Specifies the end value of the HMI tag.
Start value HMI	Specifies the start value of the HMI tag.

Example of the worksheet "Multiplexing"

	A	B	C
1	HMI Tag name	Multiplex Tag	Index
2	Mux_Tag_1	Mux_11	0
3	Mux_Tag_1	Mux_12	1
4	Mux_Tag_1	Mux_13	2
5	Mux_Tag_2	Mux_21	0
6	Mux_Tag_2	Mux_22	1
7	Mux_Tag_2	Mux_23	2
8	Mux_Tag_12	HMI_Array_Mux2	-1
9	Mux_Tag_11	HMI_Array_Mux1	-1
10	Mux_Tag_13	HMI_Array_Mux3	-1

Table 10-17 Meaning of the entries

List entry	Meaning
Name	Indicates the configured name of an HMI tag which uses indirect addressing. The HMI tag must be available in the "HMI Tags" worksheet.
Index	Shows the value which governs which tag is selected.
Multiplex Tag	Displays the tag from the tag list corresponding to the index value.

Note

"No value" in the table

Entries in the table which have the value "No value" delete the corresponding values in an existing tag of the same name.

See also

Exporting tags (Page 8807)

10.9.4.5 Importing and exporting text lists

Exporting text lists

Introduction

WinCC has an export function for text lists.

Requirements

- The WinCC project for the export is open.
- Text lists have been created in the project.
- The "Text & graphics lists" editor is open.

Exporting text lists

1. Click on the button in the "TextLists" tab.
   The "Export" dialog box opens.

2. Click the "..." button and specify in which file the data are saved.
3. By default, texts are exported in all defined project languages.
   If you do not want to export specific languages, disable the unnecessary languages in the dialog.
4. Click "Export". The export will start.

Result

The exported data has been written to an xlsx file. The xlsx file will be stored in the specified folder.

See also

Importing text lists (Page 8814)

Importing and exporting project data (Page 8795)

Exporting tags (Page 8807)

Exporting recipes (Page 8796)

Format of text list data (Page 8815)

Importing text lists

Introduction

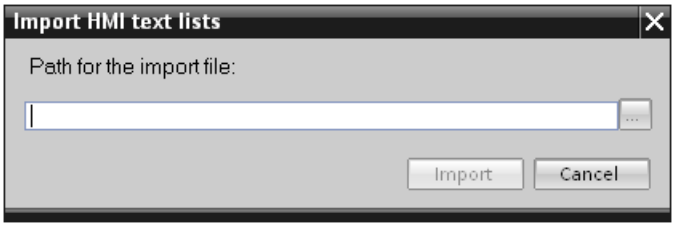
You then import text lists from an xlsx file to WinCC.

Requirements

- An xlsx file with text lists has been created.
- The structure of the xlsx file meets the requirements.
- The WinCC project for the import is open.
- The "Text & graphics lists" editor is open.

Importing text lists

1. Click on the button in the "Text lists" tab.
   The "Import" dialog box opens.



2. Select the file you want to import under "File selection".
3. Click on the "Import" button. The import will start.

Result

You have now imported the text lists. The relevant text lists have been created in WinCC. Alarms relating to the import operation are displayed in the output window. A log file is saved in the source directory of the import files. The log file has the same name as the respective import file but with the "*.xml" extension.

Check when importing the data whether there are any links to objects, for example, dynamic parameters such as tags.

- If an object with the same name already exists, the existing object is used.
- If no object of the same name yet exists, create an object with the relevant name or create a new link.

See also

Exporting text lists (Page 8812)

Format of text list data

Introduction

This section describes the format required for the file with the text lists used for imports. The text list data file must be in "*.xlsx" format.

Tag data structure

The import file in Microsoft Excel consists of two worksheets:

- TextList (Text lists)
- TextListEntry (Text list entry)

Each text list is assigned a separate line in the import file. The import file containing the data must be structured as follows:

Example of the worksheet "TextList"

	A	B	C
1	Name	ListRange	Comment
2	TLValue/Range	Decimal	
3	TLBit	Bit	
4	TLBitnumber	Binary	
5	TLall_1	Decimal	
6	TLall_2	Decimal	

Table 10-18 Meaning of the entries

List entry	Meaning
Name	Shows the name of the text list.
ListRange	Shows the text list range: Number = Bit number (0-31) Range = value/range (...-...) Bit = Bit (0;1)
Comment	Any comments on the text list. You can use up to 500 characters

Example of the worksheet "TextListEntry"

	A	B	C	D	E	F
1	Name	Parent	DefaultEntry	Value	Text	FieldInfos
2	Text_list_entry_1	TLValue/Range	TRUE	0 - 1	Default entry TLValue/Range ->	
3	Text_list_entry_2	TLValue/Range		2 - 3	TLValue/Range = 2-3 ->	
4	Text_list_entry_3	TLValue/Range		1	TLValue Range - single value = 1 ->	
5	Text_list_entry_1	TLBit		0	TLBit = 0 ->	
6	Text_list_entry_2	TLBit		1	TLBit = 1 ->	
7	Text_list_entry_1	TLBitnumber	TRUE	0	Default entry TLBitnumber; ->	
8	Text_list_entry_2	TLBitnumber		0	TLBitnumber - Bitnumber 0 is set ->	
9	Text_list_entry_3	TLBitnumber		1	TLBitnumber - Bitnumber 1 is set ->	
10	Text_list_entry_4	TLBitnumber		2	TLBitnumber - Bitnumber 2 is set ->	
11	Text_list_entry_5	TLBitnumber		3	TLBitnumber - Bitnumber 3 is set ->	
12	Text_list_entry_1	TL1	TRUE	0 - 1	Default entry TL1	
13	Text_list_entry_2	TL1		1 - 3	TL1 Value between 1 - 3 ->	
14	Text_list_entry_3	TL1		4 - 6	TL1 Value between 4 - 6 ->	
15	Text_list_entry_4	TL1		7	TL1 Single value = 7 ->	
16	Text_list_entry_1	TL2	TRUE	0 - 1	<field ref="0" /> ->	<ref id = 0; type = CommonTextList; TextList = TL1; Tag = HMI_TL1control; Length = 1
17	Text_list_entry_2	TL2		1	TL2 Single value = 1 ->	
18	Text_list_entry_3	TL2		2 - 3	TL2 Range between 2 - 3 ->	
19	Text_list_entry_1	TLMultilined	TRUE	0 - 1	Default entry TLMultilined; last row	
20	Text_list_entry_2	TLMultilined		0 - 3	TLMultilined Value between 0-3\nwith test of"\n"	
21	Text_list_entry_1	TLall_1	TRUE	0 - 1	Default entry TLall_1	
22	Text_list_entry_1	TLall_2	TRUE	0 - 1	Default entry TLall_2	

Table 10-19 Meaning of the entries

List entry	Meaning
Name	Shows the name of the text list entry.
Parent	Specifies the name of the corresponding text list.
DefaultEntry	Indicates whether the text list entry is a default entry. The default entry is always displayed when the tag has an undefined value.
Value	Specifies the tag integer values or value ranges which are assigned to the text entries in the text list.
Text	Shows the text list entry. The field designation contains a language ID. Text list entries must be assigned a language ID for import. An expression with a reference ID will be added to the text if the text list entry has a dynamic parameter. Example: text <field ref="0" />. Use the ID to assign the dynamic parameter to a text list entry.
FeldInfo	Specifies whether the text list contains dynamic parameters. The settings are separated by a semicolon ";". Example of dynamic parameters: Tag: <ref id = 0; type = CommonTagDisplayFormat; Tag = tag 1; DisplayType = Decimal; DisplayFormat = 9;> Text list: <ref id = 1; type = CommonTextList; TextList = Textliste_1; Tag = tag 2; Length = 5;> PLC tag: <ref id = 0; type = CommonControlTagDisplayFormat; DisplayType = Decimal; DisplayFormat = 9;>

See also

Exporting text lists (Page 8812)

10.9.4.6 Importing and exporting project texts

Exporting project texts

Project texts are exported for translation. Texts are exported to Office Open XML files ending in ".xlsx". These files can be edited in Microsoft Excel, for example.

You can exchange the file with the translators and import it back to the project as soon as it has been translated.

Requirements

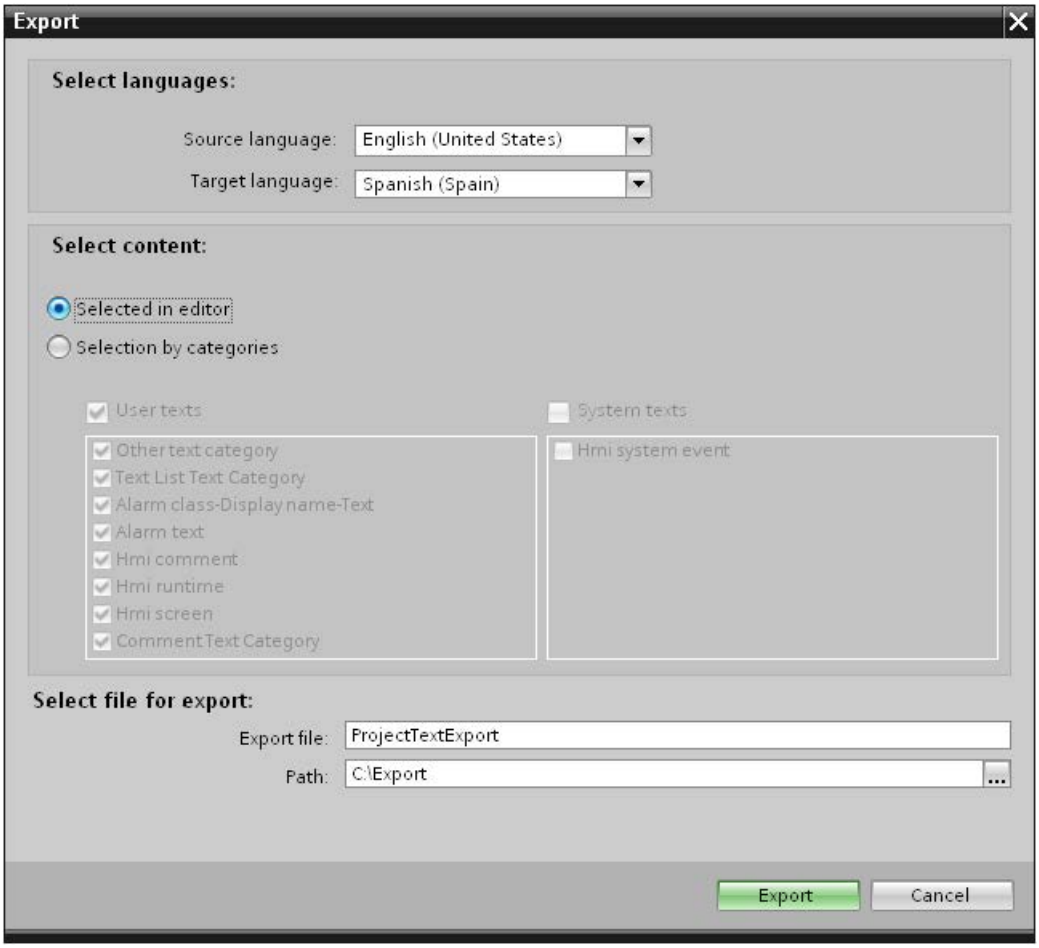
- At least two languages have been enabled in the "Project languages" editor, for example, Italian and French.

Exporting project texts

To export individual project texts, proceed as follows:

1. Click on the arrow to the left of "Languages & resources" in the project tree.
   The child elements are displayed.
2. Double-click on "Project texts". The "Project texts" editor will open.
3. Select the texts you want to export.

4. Click on the button. The "Export" dialog opens.



5. From the "Source language" drop-down list, select the language from which you wish to translate, for example Italian.
6. From the "Target language" drop-down list, select the language into which the texts are to be translated, for example, French.
7. Enter a file path and a file name for the export file in the "Export file" input field.
8. Click "Export".

Result

The texts selected in the "Project texts" editor are written to an xlsx file. The xlsx file will be stored in the specified folder.

You can alternatively select and export all project texts from categories. Select "User texts" or "System texts" in the "Export" dialog in line with the type of texts you wish to export. In this case, export can additionally be limited by categories.

Note

Project texts in library objects cannot be exported.

See also

Importing project texts (Page 8819)

Importing project texts

Edit the xlsx file or send it to a translator. Import the texts once they have been translated. The foreign languages will be imported to the relevant object in the project.

Requirements

- At least two languages have been enabled in the "Project languages" editor, for example, Italian and French.

Importing project texts

To import a project text file, proceed as follows:

1. Click on the arrow to the left of "Languages & resources" in the project tree.
   The lower-level elements will be displayed.
2. Double-click on "Project texts". The "Project texts" editor will open.
3. Click on the button. The "Import" dialog opens.
4. Select the path and file name of the import file from the "Import file" field.
5. Activate the "Import source language" check box if you have made changes to the source language in the export file and would like to overwrite the entries in the project with the changes.
6. Click on "Import".

Result

You have imported the project texts.

See also

Exporting project texts (Page 8817)

10.9.5 Using cross-references

10.9.5.1 General information about cross references

Introduction

The cross-reference list provides an overview of the use of objects within the project.

Uses of cross-references

The cross-reference list offers you the following advantages:

- When creating and changing a program, you retain an overview of the objects, tags, and alarms etc. you have used.
- From the cross-references, you can jump directly to the object location of use.
- You can learn the following when debugging:
  - The objects used in a specific screen.
  - The alarms and recipes shown in a specific display.
  - The tags used in a specific alarm or object.
- As part of the project documentation, the cross-references provide a comprehensive overview of all object, alarms, recipes, tags and screens.

10.9.5.2 Displaying the cross-reference list

Introduction

Details on the use of objects can be found in the cross-reference list. You can show cross-references for HMI devices, folders and all editors in the project tree. The detail view also lets you select individual objects of the editors.

Requirement

You have created a project.

Several objects have been created.

Procedure

1. Select the required entry in the project tree or detail view.
2. Select "Cross-references" in the shortcut menu. The cross-reference list is opened in the work area.
3. Open the "Used by" tab to display where the objects shown in the cross-reference list are used.
4. Open the "Used" tab to view the users of the objects displayed in the cross-reference list.

5. You can sort the entries in the "Object" column in ascending or descending order by clicking on the corresponding column header.
6. To go to the location of use for a specific object, click on the displayed link.

Result

The cross-reference list for the selected object is displayed in the work area.

10.9.5.3 Structure of the cross-reference list

Views of the cross-reference list

There are two views of the cross-reference list. The difference between the two views is in the objects displayed in the first column:

- Used by:
  Display of the referenced objects. Here, you can see where the object is used.
- Used:
  Display of the referencing objects. The users of the object are shown here.

The assigned tool tips provide additional information about each object.

Structure of the cross-reference list

Column	Content/meaning
Object	Name of the object that uses the lower-level objects or that is being used by the lower-level objects.
Numbers	Number of uses
Location of use	Each location of use, for example, an object or event
Property	Function of the referenced objects, for example, tag for data record or process value
Connected to	PLC tag with which the object is connected.
Type	Type of object
Path	Path of object

Depending on the installed products, additional columns or different columns are displayed for the cross-references.

Settings in the cross-reference list

You can make the following settings using the icons in the toolbar for the cross-reference list:

- Update cross-reference list
  Updates the current cross-reference list.
- Making settings for the cross-reference list
  Here, you specify whether all used, all unused, all defined or all undefined objects will be displayed. If the "Undefined objects" option is enabled, references to previously deleted objects are also displayed.
- Collapse entries
  Reduces the entries in the current cross-reference list by closing the lower-level objects.
- Expand entries
  Expands the entries in the current cross-reference list by opening the low-level objects.

Sorting in the cross-reference list

You can sort the entries in the "Object" column in ascending or descending order. Click on the column header to do this.

10.9.5.4 Displaying cross-references in the Inspector window

Introduction

The Inspector window displays cross-reference information about a selected object in the "Info > Cross-references" tab. The Inspector window displays the cross-reference information in tabular format.

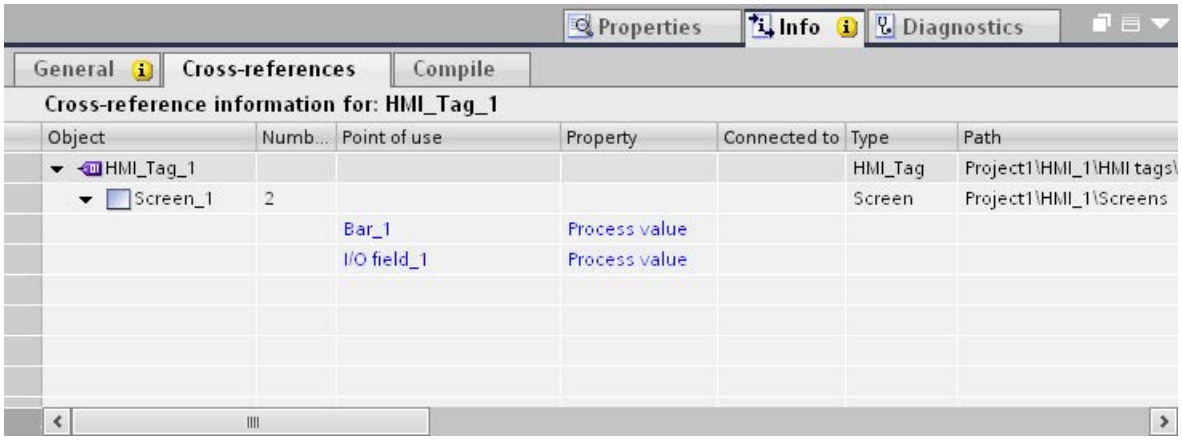
Requirement

You have created a project.

Several objects have been created.

Procedure

1. Select an object in a screen or a tabular editor.
2. Select "Cross-reference information" in the shortcut menu. The cross-references are opened in the Inspector window.



Result

The instances where and the other objects by which the selected object is being used are displayed.

The table below shows the additional information listed in the "About > Cross-reference" tab:

Column	Meaning
Object	Name of the object that uses the lower-level objects or that is being used by the lower-level objects.
Numbers	Number of uses
Location of use	Each location of use, for example an object or event
Property	Function of the referenced objects, for example tag for data record or process value
Connected to	PLC tag with which the object is connected.
Type	Type of object
Path	Path of object

Depending on the installed products, additional columns or different columns are displayed for the cross-references.

10.9.5.5 Rewiring tags in the screens

Introduction

You can change the references of tags in the properties of screen objects using the "Change object references" dialog.

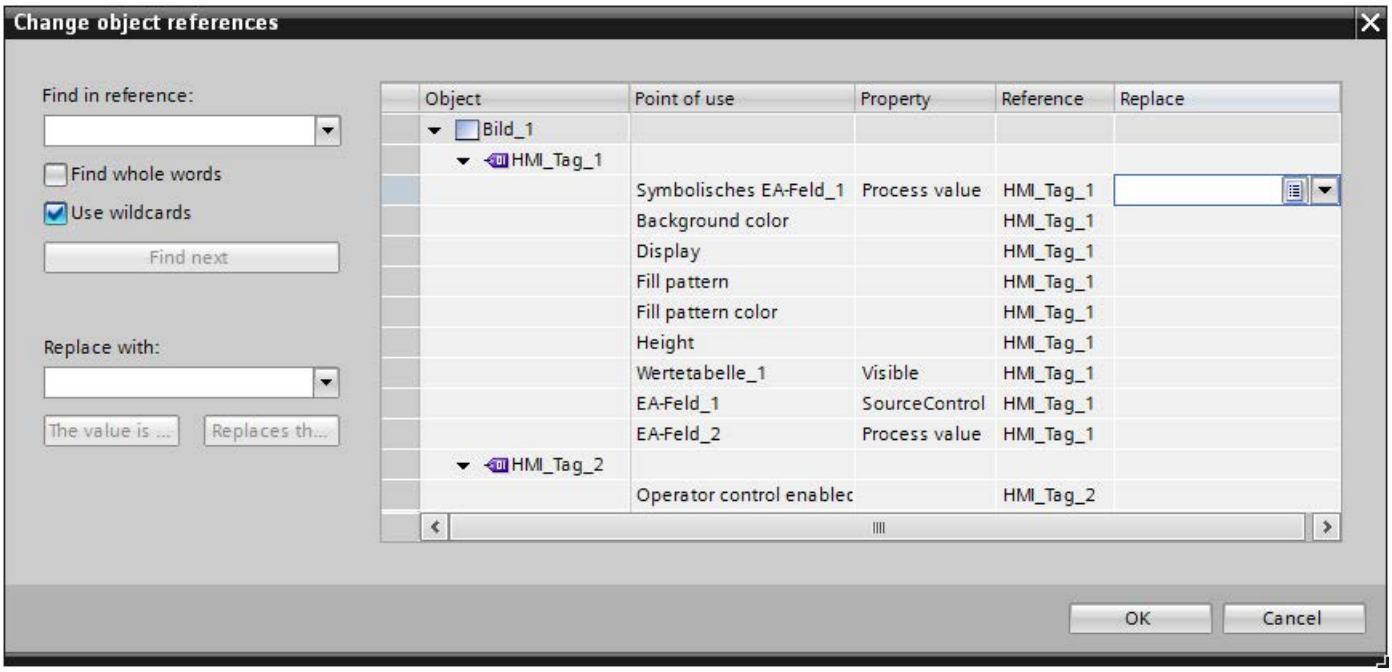
This function lets you to replace a large number of tags in various screen objects.

You search for and replace individual or multiple characters in the tag names.

You can open the "Change object references" dialog with the command in the shortcut menu. You can open the dialog for one or more screen objects or a screen.

How to replace a tag

1. Select the objects in the screen that contains a tag that you want to change.
2. Select the "Change object references" command in the shortcut menu or in the "Edit" menu. The "Change object references" dialog opens. You can see the selected screen objects and the tags used in the "Object" column.



3. Enter the name or part of the name you want to find in the "Find in reference" input box.
4. Start the search with the "Find next" button. If a reference is found, it is highlighted in the "Reference" column.
   Another click on the "Find next" button continues the search process until the list has been completely searched.
   After this, the search process starts again from the top of the list.
5. Enter the name or part of the name you want to replace for the found name in the "Replace with" input box.
6. Click the "Replace" button to replace the selected tag.
   All references found are replaced by clicking the "Replace all" button. In this way, you can replace a large number of references with another reference.
7. To select a tag from the list of all tags created in the project, click on the button in the "Replace" column.

8. To select a tag from the object list or to create a new tag, click on the button in the "Replace" column.
9. Confirm your entry with "OK".

The rewiring of the tags is applied in the project.

Additional options for searching

You can narrow down your search by enabling the following options:

- Use wildcards
  Enter * for any number of characters. Example: You want to search for all tags starting with "HMI". Enter "HMI*" in the search key box.
  Enter ? for an individual character.
- Find whole words
  To search only for full names, enable the "Find whole words" option.

Replacing arrays and user-defined tags

Variables whose names contain special characters . [ ] are shown in quotes.

Array type tags and tags of user-defined data types (UDT) contain special characters to separate tags from their elements. However, they are not shown in quotation marks. To replace an array or a UDT with an HMI tag with special characters, enter the name of the HMI tag in quotation marks in the "Replace" input box. If you type the name without quotation marks, a reference to a non-existent array or UDT is created instead of a reference to the desired HMI tag with special characters.

Correcting entries

If an incorrect entry is made in the "Replace with" input box, the existing references can be replaced by references to non-existent tags.

If you detect an incorrect entry before confirming with "OK", you can correct the error. To do this, select the "Replace" column and press the "Remove" button. Any incorrect entries in the "Reference" column are deleted.

If you confirm the dialog with "OK", the affected properties of the screen objects are marked in red. Non-existent tags are highlighted in red in the "Change cross references" dialog.

If the new tag does not yet exist, you must subsequently create the tag.

10.9.6 Managing languages

10.9.6.1 Languages in WinCC

User interface language and project languages

A distinction is drawn between two different language levels in WinCC:

- User interface language
  During configuration, the text in the WinCC menus and dialogs is displayed in the user interface language. The user interface language also affects the labeling of operating elements, the parameters of the system functions, the online help, etc.
- Project languages
  Project languages are all languages in which a project will later be used. Project languages are used to create a project in multiple languages.

The two language levels are completely independent of one another. For example, you can create English projects at any time using a German user interface and vice versa.

Project languages

The following languages are differentiated within the project languages:

- Reference language
  The reference language is the language that you use to configure the project initially. During configuration, you select one of the project languages as the reference language. You use the reference language as a template for translations. All of the texts for the project are first created in the reference language and then translated. While you are translating the texts, you can have them displayed simultaneously in the reference language.
- Editing language
  You produce translations of the texts in the editing language.
  Once you have created your project in the reference language, you can translate the texts into the remaining project languages. Select a project language respectively as an edit language and edit the texts for the appropriate language variant. You can change the editing language at any time.

  Note

  When switching the project languages, the assignment to the keys on the keyboard also changes. For some languages (for example, Spanish), the operating system does not allow you to switch to the corresponding keyboard assignment. In this case, the keyboard assignment is switched to English.

- Runtime languages
  Runtime languages are those project languages that are transferred to the HMI device. You decide which project languages to transfer to the HMI device depending on your project requirements.
  You must provide appropriate controls so that the operator can switch between languages in runtime.

See also

Language settings in the operating system (Page 8827)

Operating system settings for Asian languages (Page 8828)

Selecting the user interface language (Page 8829)

"Graphics" editor (Page 8840)

Languages in Runtime (Page 8844)

Example: Configuring a button for language switching (Page 8851)

10.9.6.2 Language settings in the operating system

Introduction

The configuration PC operating system settings influence WinCC language management in the following areas:

- Selection of project languages
- Regional format of dates, times, currency, and numbers
- Displaying ASCII characters

Project language selection

A language is not available as a project language unless it is installed in the operating system.

Regional format of dates, times, currency, and numbers

WinCC specifies a fixed date and time format in the Date - Time field for the selected project language and runtime language.

In order for dates, times, and numbers to be presented correctly in the selected editing language, this language must be set in the Regional Options in the Control Panel.

Displaying ASCII characters

With text output fields, the display of ASCII characters as of 128 depends on the set language and the operating system being used.

If the same special characters are to be displayed on different PCs, the PCs must use the same operating system and regional settings.

See also

Languages in WinCC (Page 8826)

Operating system settings for Asian languages (Page 8828)

10.9.6.3 Operating system settings for Asian languages

Settings on Western operating systems

If you want to enter Asian characters, you must activate the support for this language in the operating system.

The Input Method Editor (IME) is available in Windows for configuring Asian texts. Without this editor, you can display Asian text but not edit it. For more information on the Input Method Editor, refer to the documentation for Windows. To enter Asian characters when configuring, switch to the Asian entry method in the "Input Method Editor".

Switch the operating system to the appropriate language to have language-specific project texts, such as alarm texts, displayed in the simulator in Asian characters.

Settings on Asian operating systems

If you are configuring on an Asian operating system, you must switch to the English default input language to enter ASCII characters, for example, for object names. As the English default input language is included in the basic installation of the operating system, you do not need to install an additional input locale.

Enable language support

1. Open the system controller.
2. Select "Regional and Language Options".
3. On the "Languages" tab, activate the check box "Install files for East Asian languages".
4. Then click on "Details" under "Text Services and Input Languages". The dialog "Text Services and Input Languages" is opened.
5. On the "Settings" tab add the required default input language under the "Installed Services".
6. Select the language of the operating system in the "Language for non-Unicode programs" area in the "Advanced" tab.

See also

Languages in WinCC (Page 8826)

Language settings in the operating system (Page 8827)

10.9.6.4 Setting project languages

Selecting the user interface language

Introduction

The user interface language is used for displaying menu entries, title bars, infotexts, dialog texts and other designations in the WinCC user interface.

You can switch between the installed user interface languages during configuration. The labeling of the operating elements remains in the language you set when you added the object even if you change the user interface language.

Procedure

1. Select "Options > Settings" in the menu.
   The "Settings" dialog box is opened.
2. Select the desired user interface language under "General > General settings".

Result

WinCC will use the selected language as user interface language.

See also

Enable project languages (Page 8829)

Selecting the reference language and editing language (Page 8830)

Languages in WinCC (Page 8826)

Enable project languages

Introduction

The project languages are set in the "Project languages" editor. You define which project language is to be the reference language and which the editing language.

Enable project languages

1. Click on the arrow to the left of "Languages & resources" in the project tree.
   The lower-level elements will be displayed.
2. Double-click on "Project languages".
   The possible project languages will be displayed in the working area.
3. Enable the relevant project languages.

Note

Copying multilingual objects

The copies of multilingual objects to a different project only include text objects in the project languages which are activated in the target project. Activate all project languages in the target project to include the corresponding text objects when transferring the copy.

Disabling project languages

1. Disable the languages which are not relevant for the project.

NOTICE
If you disable a project language, all text and graphic objects you have already created in this language will be deleted from the current project.

See also

Selecting the user interface language (Page 8829)

Selecting the reference language and editing language (Page 8830)

Selecting the reference language and editing language

Introduction

The project languages are set in the "Project languages" editor. You define which project language is to be the reference language and which the editing language. You can change the editing language at any time.

Requirements

The "Project languages" editor is open.

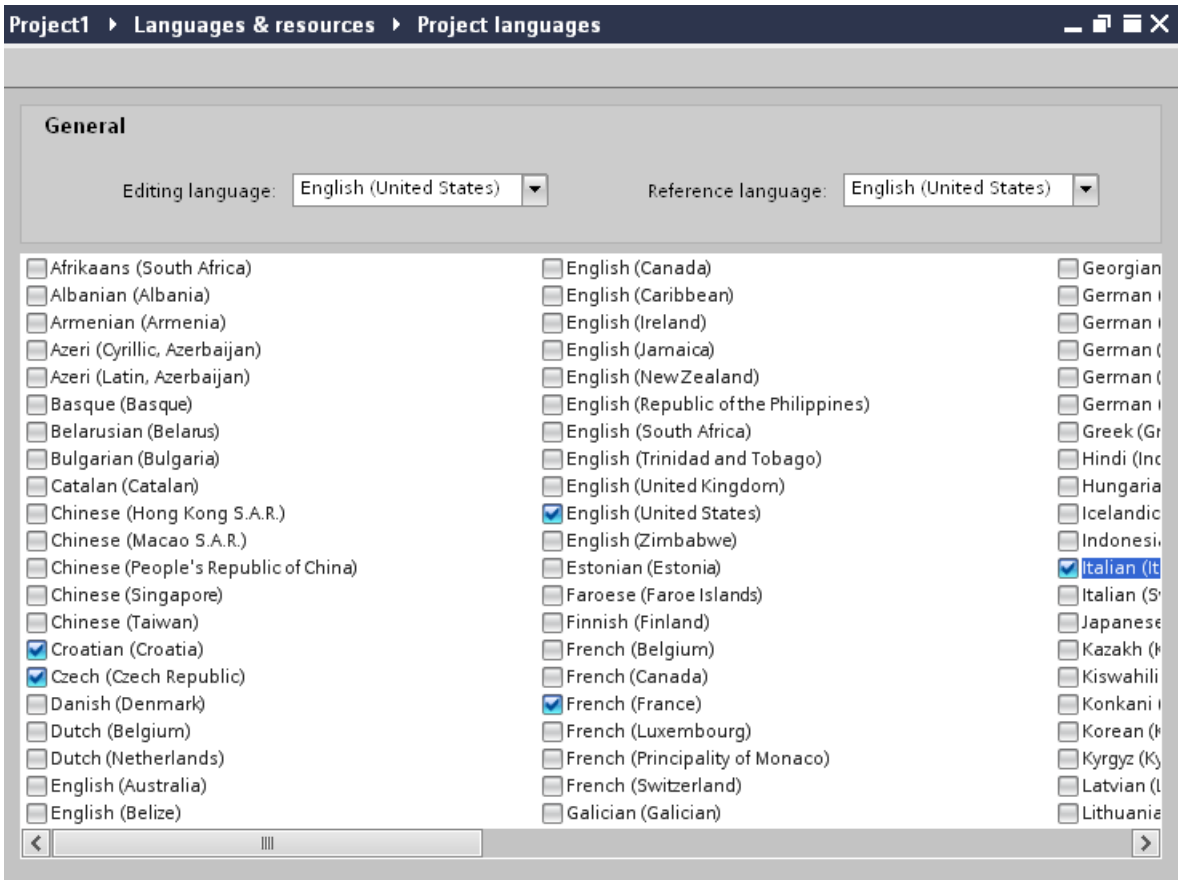
Several project languages have been activated.

Selecting the reference language and editing language

1. Click the arrow in the drop-down list in the "General > Editing language" section.
2. Click on the required language in the drop-down list, for example, German.

3. Click on the arrow in the drop-down list in the "General > Reference language" section.
4. Click on the required language in the drop-down list, for example, English.

The language selection is displayed in the list box.



Result

You have now selected the editing and reference languages.

If you change the editing language, all future text input will be stored in the new editing language.

See also

Selecting the user interface language (Page 8829)

Enable project languages (Page 8829)

10.9.6.5 Creating one project in multiple languages

Working with multiple languages

Multilingual configuration in WinCC

You can configure your projects in multiple languages using WinCC. There are various grounds for creating a project in multiple languages:

- You would like to use a project in more than one country.
  You create the project in multiple languages but when the HMI device is commissioned, only the language spoken by the operators at the respective site will be transferred to the HMI device.
- The operators of a system speak a range of different languages.
  Example: An HMI device is used in China, but the service personnel understand only English.

Translating project texts

With WinCC, you can enter project texts directly in several languages in various different editors, for example, in the "Project texts" editor. WinCC also allows you to export and import your configuration for translation purposes. This is particularly advantageous if you configure projects containing a large amount of text and want to have it translated.

Language management and translation in WinCC

The following editors are used to manage languages and translate texts in WinCC:

Editor	Short description
Project languages	Selection of project languages, editing language and reference language.
Languages and fonts	Management of runtime languages and fonts used on the HMI device.
Project texts	Central management of configured texts in all project languages.
Graphics	Graphic library for managing graphics and their language-specific versions.

See also

Project text basics (Page 8833)

Translating texts directly (Page 8834)

Translating texts using reference texts (Page 8836)

Exporting project texts (Page 8837)

Importing project texts (Page 8839)

Project text basics

Texts in different languages in the project

Texts that are output on display devices during processing are typically entered in the language in which the automation solution is programmed. Comments and the names of objects are also entered in this language.

If operators do not understand this language, they require a translation of all operator-relevant texts into a language they understand. You can therefore translate all the texts into any language. In this way, you can ensure that anyone who is subsequently confronted with the texts in the project sees the texts in his/her language of choice.

User texts and system texts

In the interests of clarity, a distinction is drawn between user texts and system texts:

- User texts are texts created by the user.
- System texts are texts created automatically and which are a product of configuration in the project.

The project texts are managed in the project text editor. This can be found in the project tree under "Languages & Resources > Project texts".

Examples of multilingual project texts

You can, for example, manage the following types of text in more than one language:

- Display texts
- Alarm texts
- Comments in tables
- Labels of screen objects
- Text lists

Translating texts

There are two ways of translating texts.

- Translating texts directly
  You can enter the translations for the individual project languages directly in the "Project texts" editor.
- Translating texts using reference texts
  You can change the editing language for shorter texts. You can enter the new texts in the editing language while the texts of the reference language are displayed.

See also

Working with multiple languages (Page 8832)

Translating texts directly (Page 8834)

Translating texts using reference texts (Page 8836)
Exporting project texts (Page 8837)
Importing project texts (Page 8839)

Translating texts directly

Translating texts

If you use several languages in your project, you can translate individual texts directly. As soon as you change the language of the software user interface, the translated texts are available in the selected language.

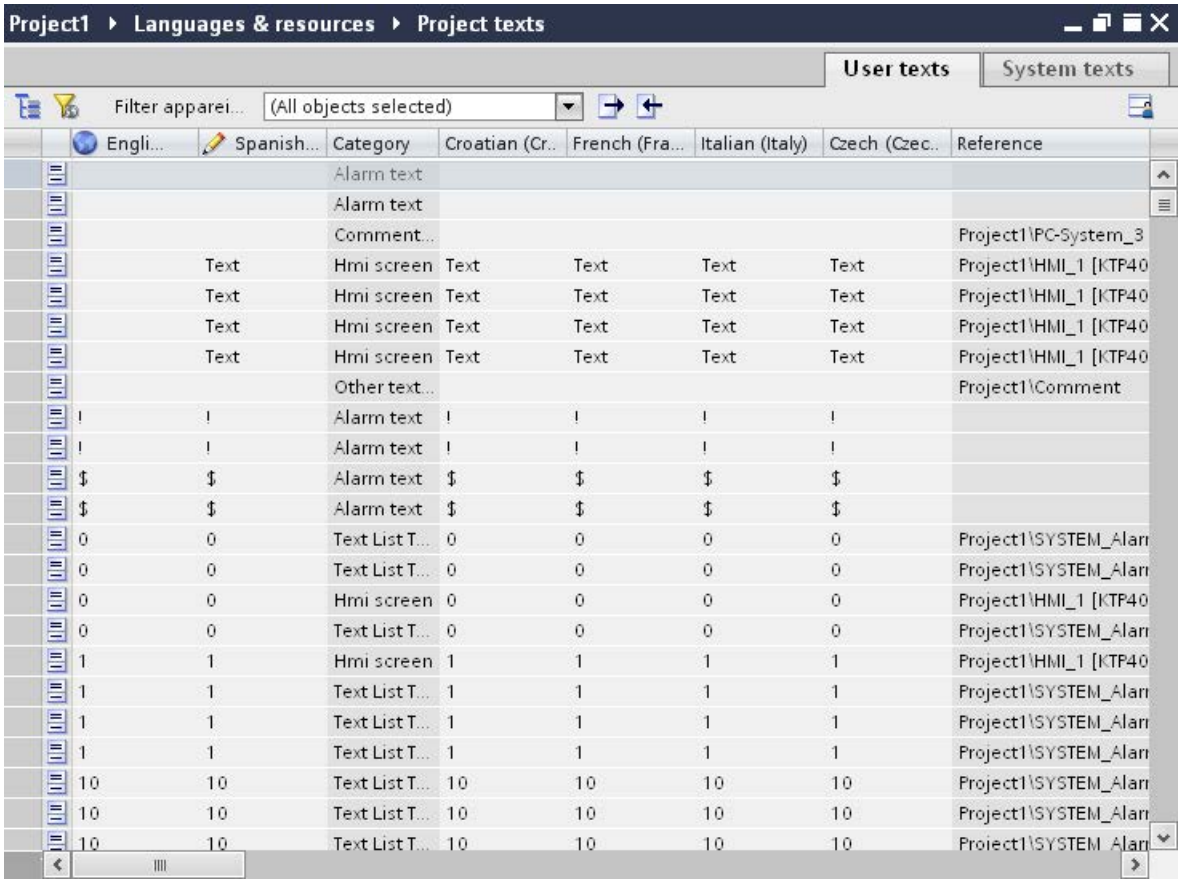
Requirements

- You are in the project view.
- A project is open.
- You have selected at least two further project languages.

Procedure

Proceed as follows to translate individual texts:

1. Click on the arrow to the left of "Languages & resources" in the project tree.
   The elements below this are displayed.
2. Double-click on "Project texts".
   A list with the texts in the project is displayed in the work area. There is a separate column for each project language.



Engli...	Spanish...	Category	Croatian (Cr..	French (Fra...	Italian (Italy)	Czech (Czec..	Reference
		Alarm text					
		Alarm text					
		Comment...					Project1\PC-System_3
	Text	Hmi screen	Text	Text	Text	Text	Project1\HMI_1 [KTP40
	Text	Hmi screen	Text	Text	Text	Text	Project1\HMI_1 [KTP40
	Text	Hmi screen	Text	Text	Text	Text	Project1\HMI_1 [KTP40
	Text	Hmi screen	Text	Text	Text	Text	Project1\HMI_1 [KTP40
		Other text...					Project1\Comment
!	!	Alarm text	!	!	!	!	
!	!	Alarm text	!	!	!	!	
$	$	Alarm text	$	$	$	$	
$	$	Alarm text	$	$	$	$	
0	0	Text List T...	0	0	0	0	Project1\SYSTEM_Alar
0	0	Text List T...	0	0	0	0	Project1\SYSTEM_Alar
0	0	Hmi screen	0	0	0	0	Project1\HMI_1 [KTP40
0	0	Text List T...	0	0	0	0	Project1\SYSTEM_Alar
1	1	Hmi screen	1	1	1	1	Project1\HMI_1 [KTP40
1	1	Text List T...	1	1	1	1	Project1\SYSTEM_Alar
1	1	Text List T...	1	1	1	1	Project1\SYSTEM_Alar
1	1	Text List T...	1	1	1	1	Project1\SYSTEM_Alar
10	10	Text List T...	10	10	10	10	Project1\SYSTEM_Alar
10	10	Text List T...	10	10	10	10	Project1\SYSTEM_Alar
10	10	Text List T...	10	10	10	10	Project1\SYSTEM_Alar

3. To group identical texts and translate them simultaneously, click on the "" button in the toolbar.
4. To hide texts that do not have a translation, click on the button in the toolbar.
5. Click on an empty column and enter the translation.

Result

You have translated individual texts in the "Project texts" editor. The texts will then be displayed in the runtime language.

See also

Working with multiple languages (Page 8832)

Exporting project texts (Page 8837)

Project text basics (Page 8833)

Importing project texts (Page 8839)

Translating texts using reference texts

Introduction

After changing the editing language, all texts are shown in input boxes in the new editing language. If there is not yet a translation available for this language, the input boxes are empty or filled with default values.

If you enter text again in an input field, this is saved in the current editing language. Following this, the texts exist in two project languages for this input field, in the previous editing language and in the current editing language. This makes it possible to create texts in several project languages.

You can display existing translations for an input box in other project languages. These serve as a comparison for text input in the current editing language and they are known as the reference language.

Requirement

There is at least one translation into a different project language for an input field.

Procedure

To display the translation of an input cell in a reference language, follow these steps:

1. Select "Tasks > Languages & resources" in the task card.
2. Select a reference language from the "Reference language" drop-down list.

Result

The reference language is preset. If you click in a text block, translations that already exist in other project languages are shown in the "Tasks > Reference text" task card.

See also

Working with multiple languages (Page 8832)

Exporting project texts (Page 8837)

Project text basics (Page 8833)

Importing project texts (Page 8839)

Exporting project texts

Project texts are exported for translation. Texts are exported to Office Open XML files ending in ".xlsx". These files can be edited in Microsoft Excel, for example.

You can exchange the file with the translators and import it back to the project as soon as it has been translated.

Requirements

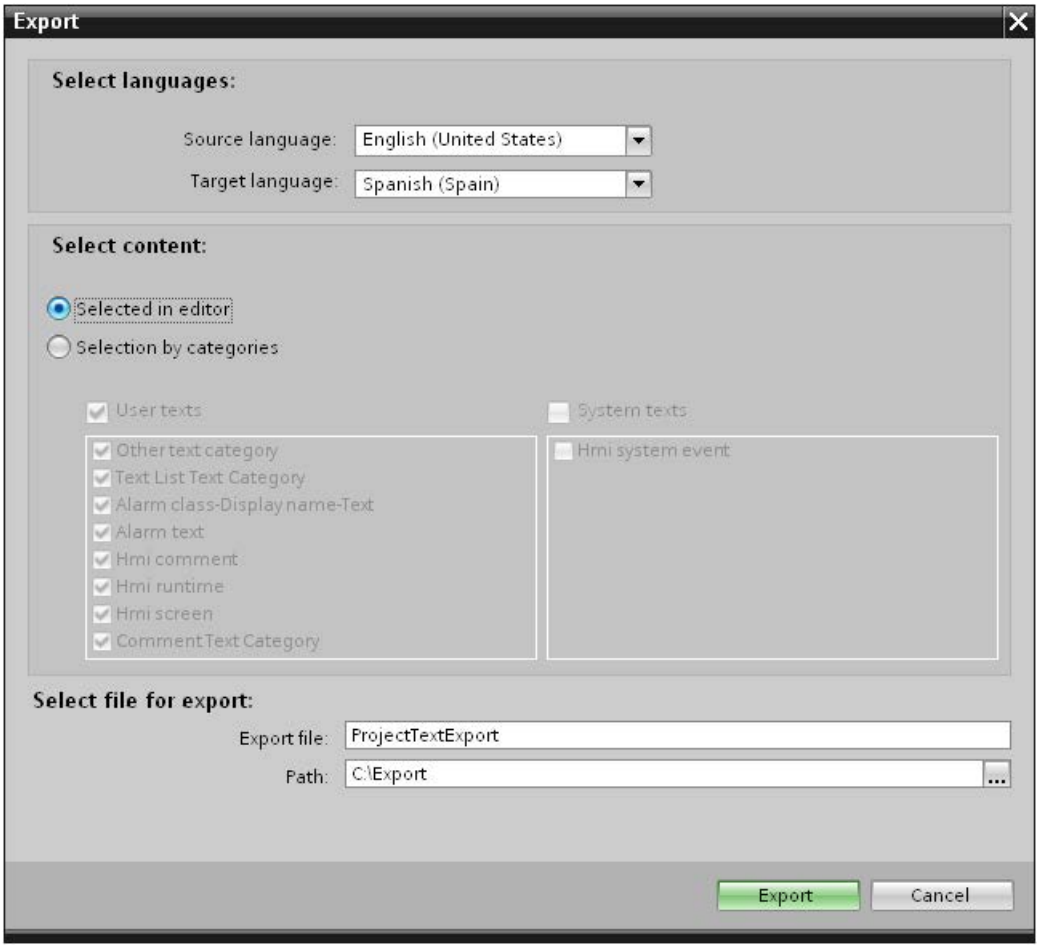
- At least two languages have been enabled in the "Project languages" editor, for example, Italian and French.

Exporting project texts

To export individual project texts, proceed as follows:

1. Click on the arrow to the left of "Languages & resources" in the project tree.
   The child elements are displayed.
2. Double-click on "Project texts". The "Project texts" editor will open.
3. Select the texts you want to export.

4. Click on the button. The "Export" dialog opens.



5. From the "Source language" drop-down list, select the language from which you wish to translate, for example Italian.
6. From the "Target language" drop-down list, select the language into which the texts are to be translated, for example, French.
7. Enter a file path and a file name for the export file in the "Export file" input field.
8. Click "Export".

Result

The texts selected in the "Project texts" editor are written to an xlsx file. The xlsx file will be stored in the specified folder.

You can alternatively select and export all project texts from categories. Select "User texts" or "System texts" in the "Export" dialog in line with the type of texts you wish to export. In this case, export can additionally be limited by categories.

Note

Project texts in library objects cannot be exported.

See also

Working with multiple languages (Page 8832)

Translating texts using reference texts (Page 8836)

Translating texts directly (Page 8834)

Project text basics (Page 8833)

Importing project texts (Page 8839)

Importing project texts

Edit the xlsx file or send it to a translator. Import the texts once they have been translated. The foreign languages will be imported to the relevant object in the project.

Requirements

- At least two languages have been enabled in the "Project languages" editor, for example, Italian and French.

Importing project texts

To import a project text file, proceed as follows:

1. Click on the arrow to the left of "Languages & resources" in the project tree.
   The lower-level elements will be displayed.
2. Double-click on "Project texts". The "Project texts" editor will open.
3. Click on the button. The "Import" dialog opens.
4. Select the path and file name of the import file from the "Import file" field.
5. Activate the "Import source language" check box if you have made changes to the source language in the export file and would like to overwrite the entries in the project with the changes.
6. Click on "Import".

Result

You have imported the project texts.

See also

Exporting project texts (Page 8837)

Working with multiple languages (Page 8832)

Project text basics (Page 8833)

Translating texts directly (Page 8834)

Translating texts using reference texts (Page 8836)

10.9.6.6 Using language-specific graphics

"Graphics" editor

Introduction

You use the "Graphics" editor to manage the configured graphic objects in different language versions. Multilingual projects sometimes also require language-specific versions of the graphics, for example, if

- the graphics contain text;
- cultural aspects play a role in the graphics.

Opening the "Graphics" editor

Double-click on "Languages and resources" in the project tree.

Work area

The work area displays all configured graphic objects in a table. There is a separate column in the table for each project language. Each column in the table contains the versions of the graphics for one particular language.

In addition, you can specify a default graphic for each graphic to be displayed whenever a language-specific graphic for a project language does not exist.

Preview

The preview shows you how the graphics will look on various devices.

See also

Storing an external image in the graphics library (Page 8842)

Storing an image in the graphics library (Page 8841)

Languages in WinCC (Page 8826)

Storing an image in the graphics library

Introduction

You use the "Graphics" editor to import graphics you want to use in screens in the "Screens" editor. It also allows you to manage language-specific versions of graphics. A preview shows the graphic displays on various HMI devices.

Requirement

- The language-dependent versions of a graphic are available.
- Multiple languages have been enabled in the "Project languages" editor.
- The "Graphics" editor is open.

Inserting graphics

1. Click "Add" in the "Graphics library" table. A dialog box opens.
2. Select the required graphic file.
3. Click "Open" in the dialog box.
   The graphic will be imported to the project and displayed in all cells in this row in the "Graphics" editor.
4. Click in the corresponding cell of a language for which a language-dependent version of this graphic exists.
5. Select "Add graphic" from the shortcut menu. A dialog box opens.
6. Select the desired graphic file and click "Open."
   The language-dependent version is inserted in the table in place of the reference language graphic.
7. Then, in the "Default graphic" column, import a graphic to be displayed in runtime for those languages for which there is no language-specific graphic.

You can also drag&drop a graphic from Windows Explorer to the relevant position in the "Graphics library" table.

Displaying graphics in the HMI device preview

1. Click on a graphic in the table.
2. Select the required HMI device under "Properties > Graphics settings > Device preview" in the Inspector window.
   The graphic will then be displayed as it will appear in runtime on the selected HMI device.

Result

The graphics added are available in the "Graphics" editor. The graphic assigned to the respective editing language will be displayed during editing. The default screen will be displayed in all editing languages for which no screen has been imported.

The screens assigned to the respective runtime language are displayed during runtime. The default screen is displayed in all runtime languages for which a screen has not been imported.

Note

If you disable a project language, all of the graphic objects you have already created in this language will be deleted from the current project.

See also

"Graphics" editor (Page 8840)

Storing an external image in the graphics library

Introduction

To display graphics that have been created in an external graphics program in your screens, you will first have to store these graphics in the graphics browser of the WinCC project.

Requirement

- Multiple languages have been enabled in the "Project languages" editor.
- The "Graphics" editor is open.
- There is a graphic in the "Graphics" editor.

Creating and adding a new graphic as an OLE object

1. Click "Add" in the "Graphics library" table. A dialog box opens.
2. Navigate to the folder in which the graphic is stored.
3. Click "Open" in the dialog box.
   The graphic will be imported to the project and displayed in all cells in this row in the "Graphics" editor.
4. Click in the corresponding cell of a language for which a language-dependent version of this graphic exists.
5. Select "Insert object" from the shortcut menu. The "Insert object" dialog box opens.

   Note

   In addition, the dialog "External application running..." will open. The dialog will not close until you exit the external application.

6. Select "Insert object > Create new" and an object type in the dialog.

7. Click "OK." The associated graphic program is opened.
8. Close the graphics program once you have created the graphic.
   The graphic will be stored in the graphic programming software standard format and added to the graphic browser.

Inserting created graphics in WinCC

1. Click in the corresponding cell of a language for which a language-dependent version of this graphic exists.
2. Select "Insert object" from the shortcut menu. The "Insert object" dialog box opens.

   Note

   In addition, the dialog "External application running..." will open. The dialog will not close until you exit the external application.

3. From the "Insert object" dialog box, select "Create from file."
4. Click the "Browse" button.
5. Navigate to the created graphic and select it.

   Note

   To import graphics files, note the following size restrictions:

   *.bmp, *.tif, *.emf, *.wmf ≤4 MB

   *.jpg, *.jpeg, *.ico, *.gif "*≤1 MB

Result

The OLE objects added are available in the "Graphics" editor.

Versions of the graphics for the current editing language are displayed in the "Screens" editor. The default graphic is displayed in all editing languages for which no screen has been imported.

The graphic is displayed in runtime in the set runtime language. The default graphic is displayed in all runtime languages for which no graphic has been imported.

You can double-click OLE objects in your library to open them for editing in the corresponding graphic editor.

See also

"Graphics" editor (Page 8840)

10.9.6.7 Languages in runtime

Languages in Runtime

Using multiple runtime languages

You can decide which project languages are to be used in runtime on a particular HMI device. The number of runtime languages that can be available at one time on the HMI device depends on the device. To enable the operator to switch between languages during runtime, you must configure a corresponding operator control.

When runtime starts, the project is displayed according to the most recent language setting. When runtime starts the first time, the language with the lowest number in the "Order for language setting" is displayed.

Setting runtime languages during configuration

In the "Languages and Fonts" editor you can specifiy:

- The project languages to be available as runtime languages for the respective HMI device.
- The order in which the languages are to be switched.

See also

Methods for language switching (Page 8844)

Enabling the runtime language (Page 8845)

Setting the runtime language order for language switching (Page 8847)

Setting the default font for a runtime language (Page 8849)

Selecting the log language (Page 8850)

Languages in WinCC (Page 8826)

Methods for language switching

Introduction

You need to configure language switching if you want to have multiple runtime languages available on the HMI device. This is necessary to enable the operator to switch between the various Runtime languages.

Methods for language switching

You can configure the following methods for language switching:

- Direct language selection
  Each language is set by means of a separate button. In this case, you create a button for each Runtime language.
- Language switching
  The operator switches the languages using a button.

Regardless of the method used, the button names must be translated into each of the languages used. You can also configure an output field that displays the current language setting.

See also

Languages in Runtime (Page 8844)

Selecting the log language (Page 8850)

Enabling the runtime language (Page 8845)

Setting the runtime language order for language switching (Page 8847)

Setting the default font for a runtime language (Page 8849)

Enabling the runtime language

Introduction

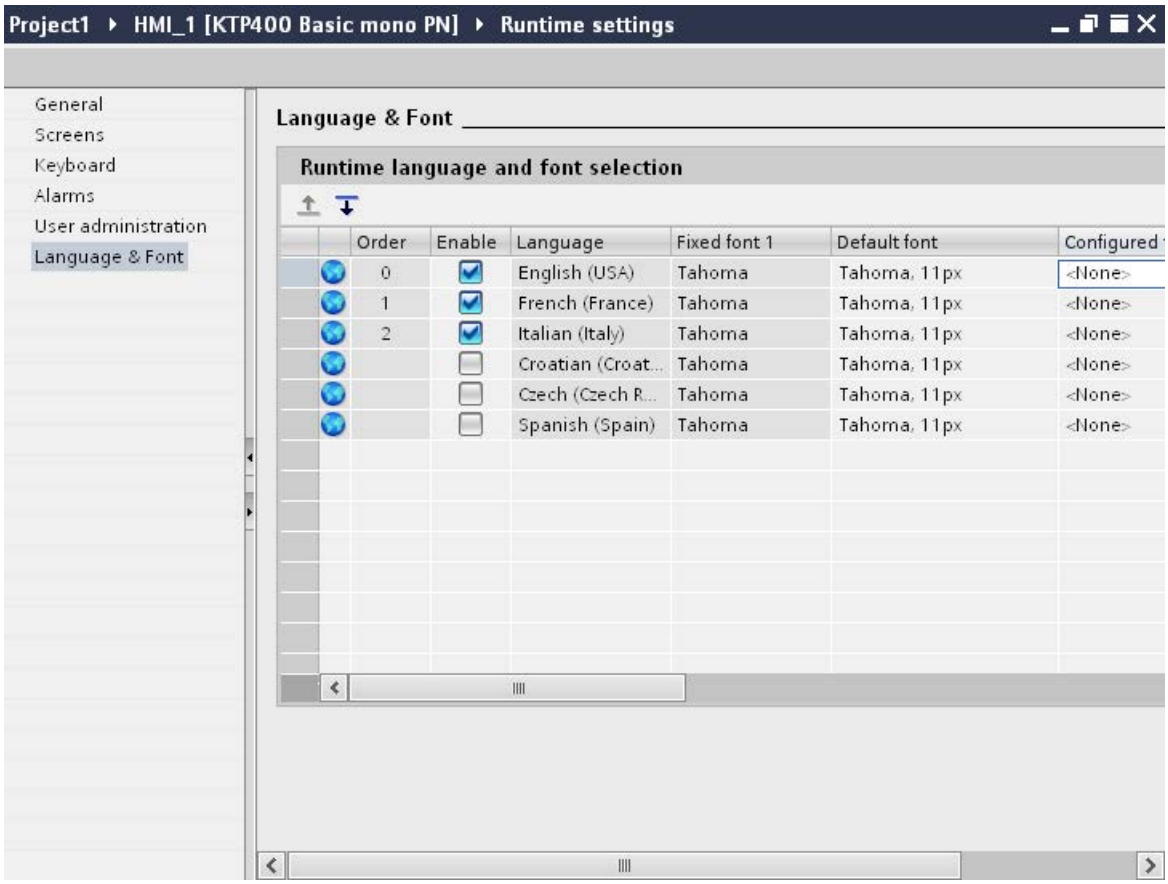
The "Language & Font" editor shows all project languages available in the project. Here you select which project languages are to be available as runtime languages on the HMI device.

Requirements

Multiple languages have been enabled in the "Project languages" editor.

Procedure

1. Double-click on "Runtime settings" in the project tree.
2. Click on "Language & Font".
3. Select the following languages:
   - English
   - Italian
   - French



Result

You have now set three runtime languages. A number is automatically assigned to each language in the "Order" column. The enabled runtime languages are transferred with the compiled project to the HMI device.

If the number of languages selected exceeds the number that can be transferred to the HMI device, the table background changes color.

See also

Languages in Runtime (Page 8844)
Selecting the log language (Page 8850)
Setting the runtime language order for language switching (Page 8847)
Methods for language switching (Page 8844)

Setting the runtime language order for language switching

Introduction

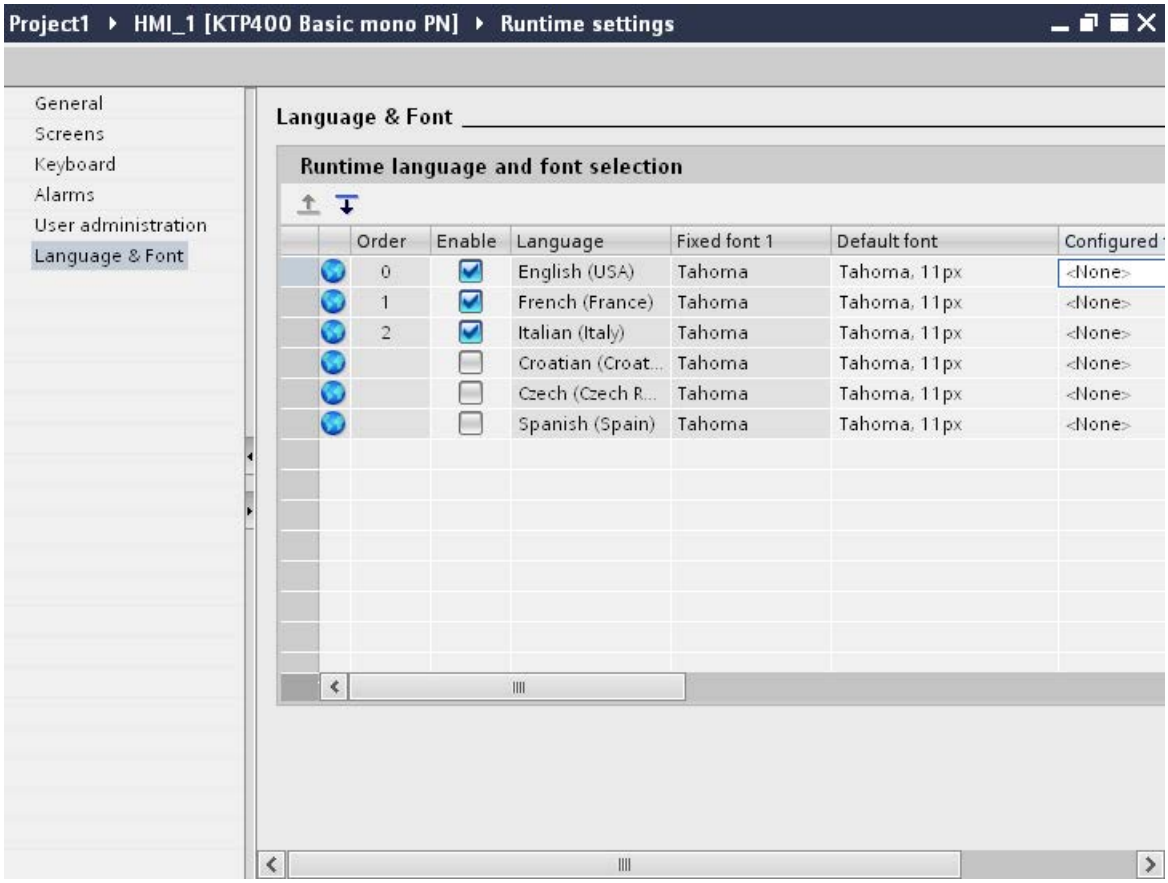
You specify the language order for runtime language switching. The first time runtime starts, the project is displayed in the language with the lowest number in the "Order" column.

Requirements

- Multiple languages have been enabled in the "Project languages" editor.
- "Language & Font" is open in the editor and three runtime languages have been set in the following order:
  1. English
  2. Italian
  3. French

Procedure

1. Select the runtime language "Italian".
2. Click the button. The runtime language "Italian" is moved down a place. The number will automatically be changed to "2" in the "Order" column.



Result

You have changed the order of runtime languages. The first time runtime starts, the project will be displayed in the language with the smallest number, in other words Chinese. If the language is switched, this will happen in numerical order.

See also

Languages in Runtime (Page 8844)

Selecting the log language (Page 8850)

Enabling the runtime language (Page 8845)

Setting the default font for a runtime language (Page 8849)

Methods for language switching (Page 8844)

Setting the default font for a runtime language

Introduction

You can specify the font used to display the texts for each runtime language on the HMI device in the "Language & Font" editor. The default font is used in all texts, such as dialog texts, for which you cannot define a specific font.

WinCC offers only fonts supported by the HMI device.

Requirements

- Multiple languages have been enabled in the "Project languages" editor.
- Three runtime languages have been enabled in the "Language & Font" editor.
  1. Chinese
  2. German
  3. French

Procedure

1. Double-click on "Runtime settings" in the project tree.
2. Click on "Language & Font". The table shows the runtime languages and fonts set.
3. Click in the "French" row in the "Default font" column.
4. Select the font to be used by default if a font cannot be selected for a given text.

Result

The project texts for the runtime language "French" are displayed on the HMI device in the selected font.

These fonts are transferred to the HMI device during a transfer operation.

The default font is also used for the representation of dialogs in the operating system of the HMI device. Select a smaller font as default if the full length of the dialog texts or headers is not displayed.

See also

Languages in Runtime (Page 8844)

Selecting the log language (Page 8850)

Setting the runtime language order for language switching (Page 8847)

Methods for language switching (Page 8844)

Selecting the log language

Introduction

In the "Runtime settings > General" editor, select the language to be used for writing to logs in runtime.

Requirements

- The languages used in your project are activated in the "Project languages" editor, for example "German" and "English" .

Procedure

1. Double-click on "Runtime settings" in the project tree.
2. Click on "Language & Font".
3. Activate the runtime languages, for example, "German" and "English".
4. Specify the "order":
   - 1 German
   - 2 English
5. Click on "Runtime settings > General".
6. Select "German" for "Logs > Log language".

Result

After loading, the project will start in the runtime language "German". The logs are now written in German. During runtime, the operator switches the runtime language to English. The logs will still to be written in German.

See also

Languages in Runtime (Page 8844)

Setting the default font for a runtime language (Page 8849)

Setting the runtime language order for language switching (Page 8847)

Methods for language switching (Page 8844)

Enabling the runtime language (Page 8845)

10.9.6.8 Example of multilingual configuration

Example: Configuring a button for language switching

Introduction

In this example, you configure a button that can be used to toggle between multiple runtime languages during runtime.

Requirements

- You have completed the "Configuring a button in multiple languages" example.
- The "Screen_1" screen is open.
- The button on the screen has been selected.

Procedure

1. In the Inspector window, select "Properties > Events > Press".
2. Click on "Add function" in the table.
3. Select the "SetLanguage" system function and the "Switch" setting.

Result

You have assigned the button the function "SetLanguage". Pressing the button during runtime will switch the runtime language. The runtime languages are switched in the order specified by the number sequence in the "Languages and fonts" editor.

See also

Example: Configuring a button in multiple languages (Page 8851)

Example: Configuring a button for language switching for each runtime language (Page 8852)

Languages in WinCC (Page 8826)

Example: Configuring a button in multiple languages

Introduction

In this example, you configure a "Sprache umschalten" button in German and "Switch language" button in English.

Requirements

- The languages "German" and "English" have been enabled in the "Project languages" editor.
- German has been set as editing and reference language.
- You have created and opened the "Screen_1" screen.
- The Inspector window is open.

Procedure

1. Drag-and-drop a button from the "Tools" task card into the screen.
   The button will be added to the screen.
2. In the Inspector window, open "Properties > Properties > General".
3. Enter the text ""Sprache umschalten" under "Text > off".
4. Press <Enter> to confirm. The button is named.
5. Open the "Tasks" task card.
6. Select "English" under "Languages & Resources > Editing language".
7. Enter the label "Switch Language" under "Properties > Properties > General > Text > Off" in the Inspector window.

Result

The button name is configured in German and English language. The button name corresponding with the current Runtime language is shown in Runtime.

See also

Example: Configuring a button for language switching (Page 8851)

Example: Configuring a button for language switching for each runtime language (Page 8852)

Example: Configuring a button for language switching for each runtime language

Introduction

In this example, you configure a "Sprache umschalten" button in German and "Switch language" button in English.

Requirements

- The following languages have been enabled in the "Project languages" editor
  - German
  - English
  - Italian
- All languages have been set as runtime languages in the "Runtime settings > Language & Font" editor.
- You have created and opened the "Screen_1" screen.
- Three buttons have been created on the screen:
  - Button_1 labelled "Deutsch"
  - Button_2 labelled "English"
  - Button_3 labelled "Italiano"
- The Inspector window is open.

Procedure

1. Select "Button_1".
2. In the Inspector window, select "Properties > Events > Press".
3. Click on <Add function> in the table.
4. Select the "SetLanguage" system function.
5. Click on the "Switch" field.
6. Click on the button.
7. Select "Runtime language". The field will be highlighted in red.
8. Select "German" from the drop-down list.
9. Repeat steps 1 - 8 for the other two buttons and select the corresponding runtime language.

Result

You have configured three buttons for switching language in runtime. Each button will switch to a different runtime language. For example, clicking on the "English" button during runtime will switch the runtime language to English.

See also

Example: Configuring a button for language switching (Page 8851)

Example: Configuring a button in multiple languages (Page 8851)

10.9.7 Replacing devices

10.9.7.1 Basics

Introduction

You can use existing configurations for new devices and optimize these configurations with little manual effort.

All data configured by you is retained in the configuration data. This means you do not need to copy individual objects of one device and paste them to another.

Principle

The following applies when you replace devices:

- Only functions supported by the new device are available. Only configuration data supported by the new device are displayed.
  This affects
  - recipes,
  - objects available on the screens,
  - available system functions,
  - available communication logs, etc.
- The number of supported objects, such as screens or tags, may be limited on the new device. If the existing objects exceed the limitations on the new device, the objects are displayed in full. The objects are, however, highlighted in color in the individual editors. An error is generated when the project data is compiled.
  Manual post processing is required when switching to a device with fewer features.
  Example: Limited number of connections
  All connections will be highlighted in color as invalid if fewer connections are supported on the new device than have been configured. Delete any excess connections.

Note

If you replace a Panel and select a PC Station as your new device, for example, WinCC Runtime Advanced will automatically be moved below the PC Station in the project tree.

See also

Device-specific functions (Page 8855)

Screen adjustment options (Page 8858)

Key assignment when replacing devices (Page 8856)

Engineering system (Page 8959)

Basic Panel (Page 8961)

10.9.7.2 Device-specific functions

Device-specific functions

Functions dependent on the device

Functions dependent on the device are implemented as follows:

- Colors
  The color is changed automatically when you switch from a device with full color display to one with a smaller color range.
  If you change the color manually and then change back again to a device with a larger range of colors, the reduced range of colors will be retained.
- Fonts
  Any configured font not available on a device will be replaced by a similar one or by the configured default font. The default font depends on the device selected.
- Character sets with different font sizes
  Avoid using too many different font sizes when configuring the following devices:
  - OP 73
  - OP 77A
  - TP 177A

  A character set is downloaded to the device for each font size. Check the Inspector window during compilation to see how much of the device memory is being used by the character sets.
- Font size
  Use small Windows fonts to display the text on devices. If you use large Windows fonts, then, depending on the size of the display, the text will not be displayed in full.
  Using font sizes of 28 pixels or more for the OP 77A and TP 177A devices will affect device performance.
  The character scope is much greater for Asian languages. The use of different font sizes therefore has serious implications on the memory requirements of all devices.
  Use the same font type for all large characters throughout the project to ensure effective and efficient configuration.
- Screens and screen objects
  If the new device supports a different resolution than the previous device when you replace a device, there are several ways to adjust the screens.
  Adjust the size of the screens to the new device in the menu under "Options > Settings > Visualization > Fit to size screen".

See also

Basics (Page 8854)

Key assignment when replacing devices

Introduction

The devices available each have different function keys. The functions configured for these keys will be mapped to the available function keys of the new device if the device is replaced.

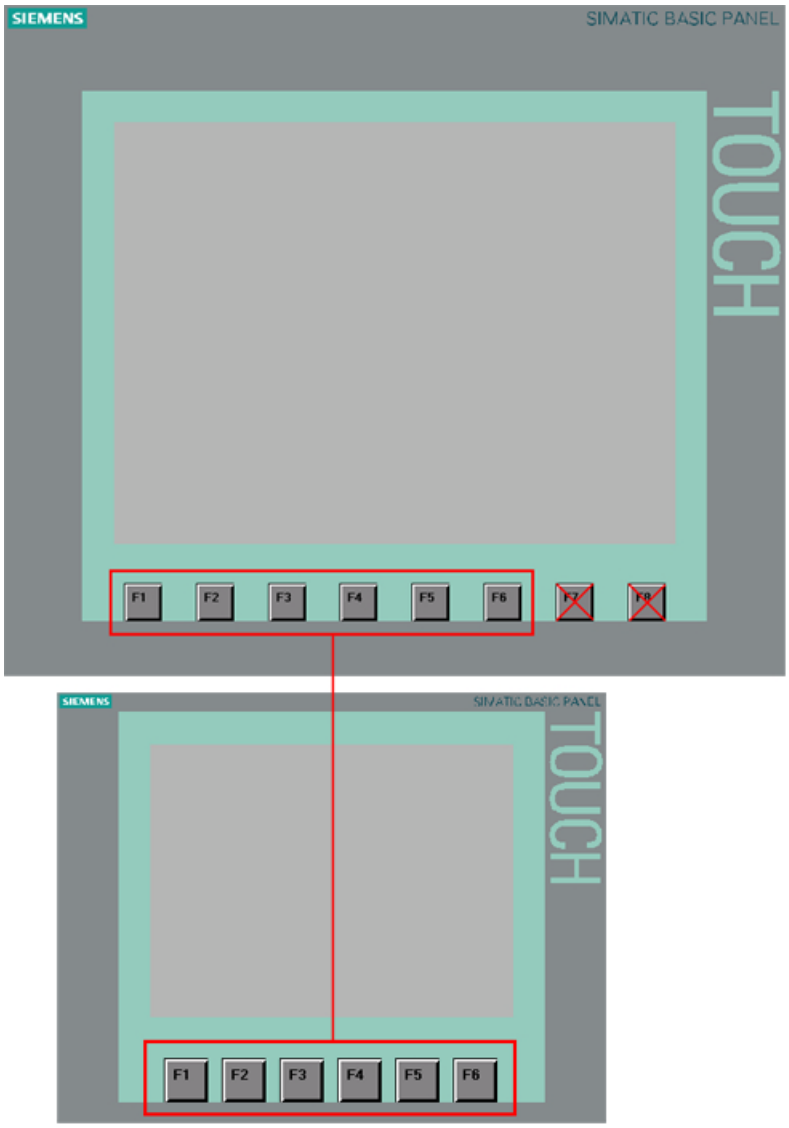
Function key mapping

The function keys below the display are mapped from left to right to the new device. If the new device has fewer keys, the keys it does not have are not mapped.

Example: Replacing a KTP1000 Basic with a KTP600 Basic

You have configured a function for F2 in KTP1000 Basic. This function is triggered by F2 following replacement with a KTP600 Basic.

If you have used F7 in a KTP1000 Basic, this function will no longer be available if the panel is replaced with a KTP600 Basic.



Mapping of control keys and cursor keys

The following keys are mapped only to the same keys of the new device:

- HELP
- ESC
- ACK
- ENTER
- PAGE UP
- PAGE DOWN
- CURSOR UP
- CURSOR DOWN

See also

Basics (Page 8854)

Screen adjustment options (Page 8858)

10.9.7.3 Adjusting screens to the new device

Screen adjustment options

Introduction

Select fit to size for screens before you replace a device. Fit to size is particularly important when switching devices with different display resolutions.

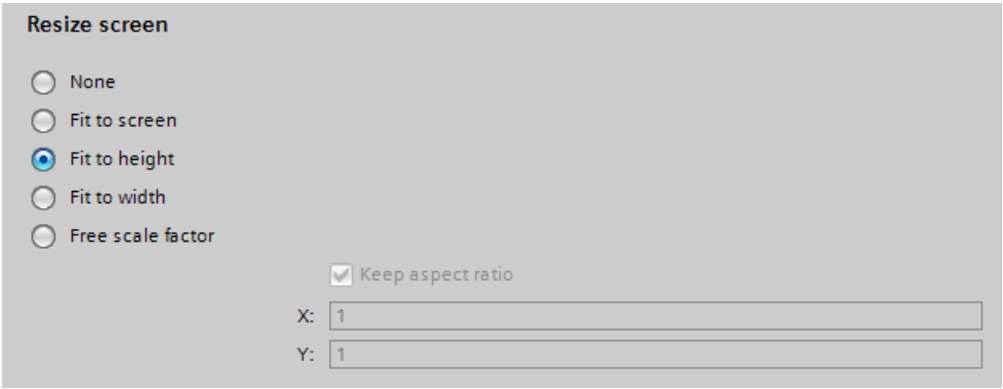
Object adjustment to content can be prevented for objects such as graphic views or text fields.

Note

The objects are distorted if you replace a device with a landscape format display with a device with a portrait format display. The difference in display format can, for example, result in object labels being cut off and content not being fitted to the object. You must therefore adjust the screens to the new device once you have replaced devices.

Screen adjustment when replacing devices

Adjust the size of the screens to the new device in the menu under "Tools > Settings > Visualization > Resize screen".



Select one of the following settings.

None (default setting)

The screens are not scaled. The objects in the screen retain their position and size. Use this setting as first test for checking of a possible exchange result because there are no rounding losses during forth and back exchange.

This option may result in objects being outside the configurable area if the display of the new device is smaller than the old one.

Fit to screen

The position and object size are adjusted to the new display size. Resizing takes place along the x-axis and the y-axis. Graphics and font size are adjusted accordingly.

Fit to height

The aspect ratio is maintained and the screens are adjusted to the height of the new device.

Use this option when you are replacing a device with display format 4:3, for example, with a device with widescreen.

Fit to width

The aspect ratio is maintained and the screens are adjusted to the width of the new device.

Use this option when you are replacing a device with widescreen, for example, with a device with display format 4:3.

Free scale factor

You select a free scale factor for screen adjustment. You can specify a factor for the x-axis and the y-axis.

Using a free scale factor of < 1 may distort the objects. Object labels may, for example, be cut off and the content may not be fitted to the object.

You must therefore adjust the screens to the new device once you have replaced devices.

Note

The aspect ratio is not adjusted for objects with a fixed aspect ratio, for example, gauge, circle. The objects are displayed on the new device with the same aspect ratio as prior to the replacement of the device.

See also

Specifying the position of screen objects (Page 8862)

Fit objects to contents (Page 8860)

Basics (Page 8854)

Key assignment when replacing devices (Page 8856)

Fit objects to contents

Introduction

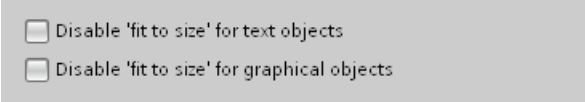
For some objects, you can specify fit to respective content in the Inspector window, for example:

- Text field: fit to text content
- I/O field: fit to text content
- Symbolic I/O field: fit to text content or to text list
- Graphic view: fit to included graphic

Fit to size for text and graphic objects

Disable automatic fit to size of the individual objects in the menu under "Options > Settings > Visualization > Resize screen and screen objects > Fit to content". This results in scaling of the objects as specified under "Options > Settings > Visualization > Resize screen and screen objects".

Select the objects which are not automatically fitted to size.



- If "Disable 'fit to size' for text objects" is activated, automatic fit to size is ignored in the text object properties.
  If you have activated "Fit screen to window height", the text field along with the other objects is scaled in accordance with the height of the new device.
- If "Disable 'fit to size' for graphic objects" is activated, automatic fit to size is ignored in the graphic object properties.
  If you have activated "Fit screen to window width", the graphics view along with the other objects is scaled in accordance with the width of the new device.

Note

The settings have no effect on screen objects whose size cannot be changed, such as alarm indicators or screen objects with a fixed aspect ratio.

"Disable 'fit to size' for text objects" and "Disable 'fit to size' for graphic objects" have no effect if:

- You have activated "Resize screen and screen objects > None".

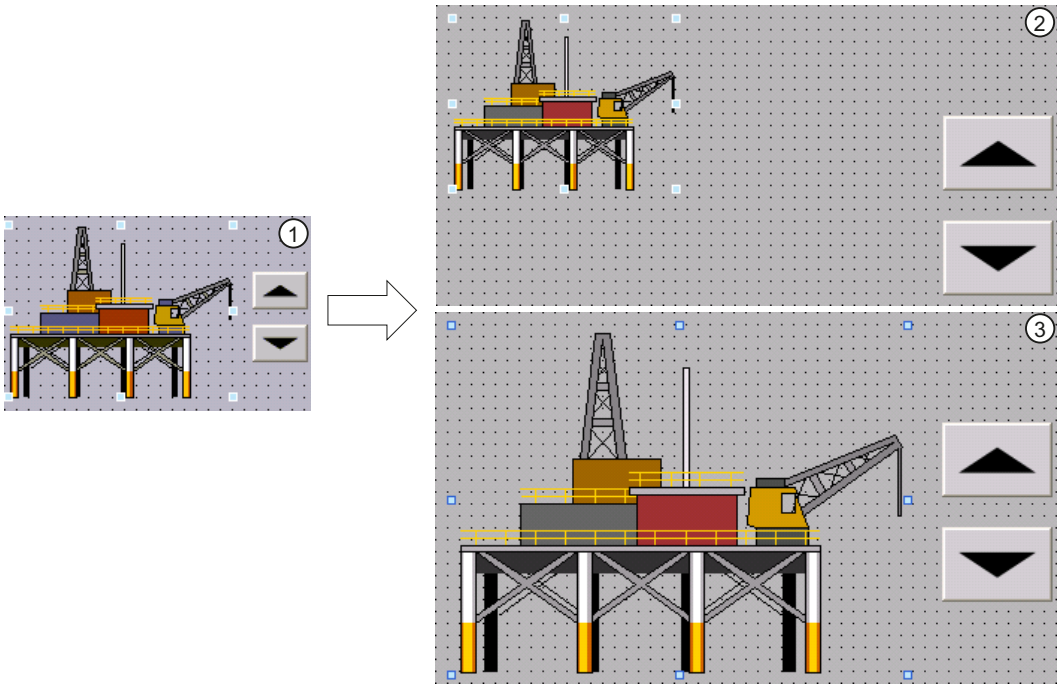
- You have activated "Fit screen to window width and height" and the new device has the same resolution as the current device.

- You have activated "Fit screen to window height" and the new device has the same resolution as the current device.

- You have activated "Fit screen to window width" and the new device has the same resolution as the current device.

Example

The figure below shows the effects of automatic sizing using a graphic object with two buttons aligned as an example:



① Initial situation:

- Two buttons are aligned on a graphic object.
- The option "Fit object size to graphic" or "Adjust object size to graphic" is activated in the object properties of the graphic object under "Display > Sizing".

② Option 1: The original properties of the graphic object are to be maintained after switching the HMI device.

- Deactivate the option "Disable 'fit to size' for graphical objects" in the settings under "Size adaptation of objects".

Effect: The graphic object retains its original size after switching the HMI device. The alignment to the buttons is lost.

③ Option 2: The graphic object is to be placed relative to the new screen resolution after switching the HMI device.

- Activate the option "Disable 'fit to size' for graphical objects" in the settings under "Size adaptation of objects".

The option "Fit graphic to object size" is activated automatically in the object properties of the graphic object. The two buttons are properly aligned on the graphic object even after switching the HMI device.

See also

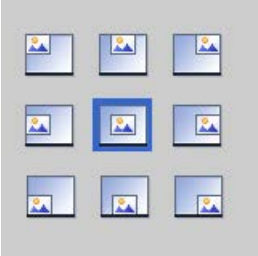
Specifying the position of screen objects (Page 8862)

Screen adjustment options (Page 8858)

Specifying the position of screen objects

Introduction

There are various ways to adjust the position of screen objects to the new device.



Select position

Adjust the position of the screen objects to the new device in the menu under "Options > Settings > Visualization > Fit to size screen > Select position".

Example

The following option aligns the objects with the top left edge.

The following object centers the objects in the middle of the screen.

See also

Fit objects to contents (Page 8860)

Screen adjustment options (Page 8858)

10.9.8 Copying between devices and editors

10.9.8.1 Basics

Basics

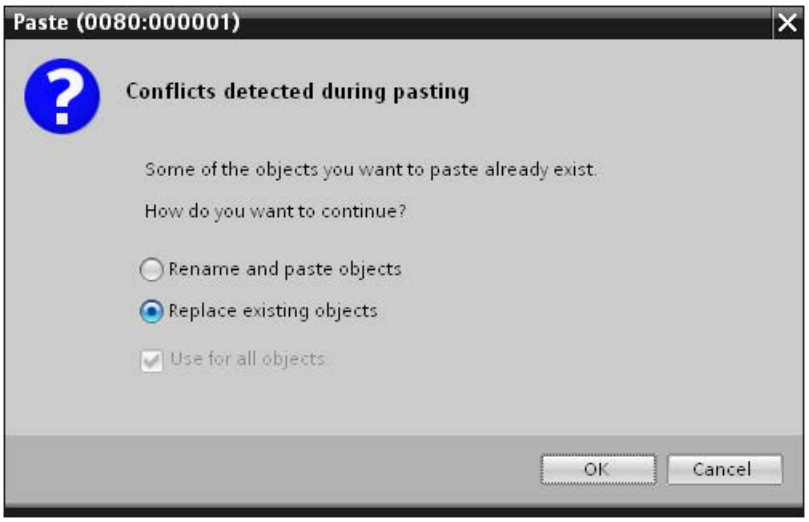
Copying and pasting within an HMI device

You can copy and paste objects, such as display objects, within an HMI device. If the object is already created in the editor, when the object name is inserted a number is automatically attached, in accordance with the following principle:

- "<Object_name>_1" is renamed to "<Object_name>_2".
- "<Object_name>_2" is renamed to "<Object_name>_3".

Copying and pasting between HMI devices

You can also copy and paste between HMI devices. If an object with the same name already exists, you have the following options:



Note

Exception to this basic rule

Copying and pasting of an alarm class that has been generated from project-wide alarm class is handled differently than with this basic rule. When the copied alarm class already exists in the target HMI device within the same project, the "Paste" command is not performed.

Copying user-defined folders

You can create user-defined folders for editors, for example, for HMI tags, screens, etc.

You can copy user-defined folders and paste them into another HMI device. The objects contained in a user-defined folder may exceed the limitations applying to the other HMI device, such as the number of supported screens. After they have been pasted, all the objects are displayed. An error is displayed when the project data is compiled.

System folders cannot be copied.

See also

Unsupported objects and functionalities (Page 8864)

Unsupported objects and functionalities

Introduction

When an object is copied, all its properties and settings are transferred to the target HMI device.

Unsupported objects

Objects that are not supported in the target HMI device cannot be pasted.

Note

When you copy a screen containing objects which are not supported by the destination HMI device, the objects remain in the background. When you copy the screen again and the new device supports the objects, they are displayed again.

Invalid objects

The following objects become invalid once they have been pasted into the target HMI device:

- Referenced objects that do not exist in the target HMI device.
- Objects with settings that are not supported in the target HMI device.
- System functions that were configured for objects and that are not supported in the target HMI device.

Invalid objects are highlighted by a color coding. Select a supported object or create a new one. If you retain an invalid object, an error will be displayed when the project data is compiled.

Colors and fonts

Colors and fonts are supported to varying degrees by HMI devices. When unsupported colors and fonts are pasted, they are replaced by supported colors and fonts. When you paste the same object back into the source HMI device, the original settings become active again.

See also

Basics (Page 8863)

10.9.8.2 Copying and pasting

Copying screens

Introduction

You copy one or more screens from the "Screens" folder and paste them into the "Screens" folder of another device.

Type and size of the displays

In the case of HMI devices with keys, the available keys are displayed automatically in the screen. When a screen is copied between HMI devices, the keys are either displayed or hidden; functions configured for function keys are not transferred.

If there is less space for the screen in the target HMI device than in the source HMI device, you can adjust the size of and the spacing between existing objects.

Automatic fit to size for objects

1. Select "Options > Settings > Visualization > Resize screen and screen objects" in the menu.
2. Activate, for example, "Fit to height".

See also

Copying recipes within an HMI device (Page 8865)

Copying objects with linked objects (Page 8866)

Linked objects copied automatically (Page 8867)

Copying recipes within an HMI device

"Recipes" Editor

You can copy recipes, recipe elements and recipe data records within each table. You copy a recipe element to another recipe.

Only WinCC Runtime Professional: You can copy a recipe view element to another recipe view. If a recipe view element of the same name already exists, a conflict dialog is displayed. You can select whether to replace or rename the recipe element. You can copy recipe elements to the first empty row of the "Recipe views" editor, "Elements" tab.

You can copy a recipe data record to another recipe, if the other recipe contains the same number of recipe elements. If the data types differ, the value will be copied to the target data record but it is assigned an error flag.

"Tags" editor

You can drag-and-drop a tag to a recipe element in the "Tag" column. The tag is linked to the recipe element. If a tag is already linked, an error message will be generated.

"Screens" editor

If you drag-and-drop a recipe to a screen, a new recipe display will be created and linked to the recipe.

See also

Copying screens (Page 8865)

Copying objects with linked objects (Page 8866)

Linked objects copied automatically (Page 8867)

Copying objects with linked objects

Introduction

An object is linked to another object in the following situations, for example:

- You specify a tag for an alarm as a trigger tag.
  The alarm is the object. The tag is the linked object.
- You specify a connection for an external tag.
  The tag is the object. The connection is the linked object.

The object is always fully inserted during copying and pasting. Whether or not the linked object is pasted depends on the command used to insert it.

Simple pasting

The linked object is not copied. The linked object is transferred and handled as follows in the target HMI device:

- If an object with the same name exists, the existing object with its settings is used.
- If no object with the same name exists, the name of the object will be displayed. The object becomes invalid.

For some objects, linked objects are pasted automatically during simple pasting.

Extended pasting

Select the "Extended paste" command in the shortcut menu to paste the linked objects as well. If objects of the same name exist in the target HMI device, you need to decide whether or not to overwrite each of these objects.

See also

Copying screens (Page 8865)

Copying recipes within an HMI device (Page 8865)

Linked objects copied automatically

Copying linked objects

The following table shows the objects for which linked objects are pasted automatically in simple pasting.

Object	Linked object
Screen	Template
Symbolic I/O field	Text list
Graphic I/O field	Graphics list
Graphic view	Graphic
Tag	Alarm
	Cycle
Recipe element	Text list
Scheduler	Triggers

See also

Copying screens (Page 8865)

Copying recipes within an HMI device (Page 8865)

Drag & drop from the details view

Introduction

You can improve configuration efficiency with just a few simple measures. Below are a few examples of efficient configuration.

Pasting objects to a screen from the details view

You can drag objects in the details view from various different editors to other editors.

Pasting a symbolic I/O field

1. Open a screen.
2. Click on the "Text and graphics lists" editor in the project tree. All existing text and graphics lists will be shown in the details view.
3. Click on a text list, for example, "Textlist1" in the Details view.
4. Drag-and-drop a text list from the Details view to a screen. A symbolic I/O field has been created and connected to the text list "Textlist1".

Pasting a graphic I/O field

1. Open a screen.
2. Click on the "Text and graphics lists" editor in the project tree. All existing text and graphics lists will be shown in the details view.
3. Click on a graphics list in the Details view, for example "Graficlist1".
4. Drag-and-drop a graphics list from the Details view to a screen. A graphic I/O field has been created and connected to the graphics list "Graficlist1".

Pasting an I/O field

1. Open a screen.
2. Click on the "HMI tags" editor in the project tree. All existing HMI tags will be shown in the Details view.
3. Click on an HMI tag in the Details view, for example "Tag1".
4. Drag-and-drop the HMI tag from the Details view to a screen. An I/O field has been created and connected to the HMI tag "Tag1".

10.9.8.3 Copying between different RT and ES versions

Introduction

You can copy and paste project data such as screens, objects or tags between projects with different WinCC versions.

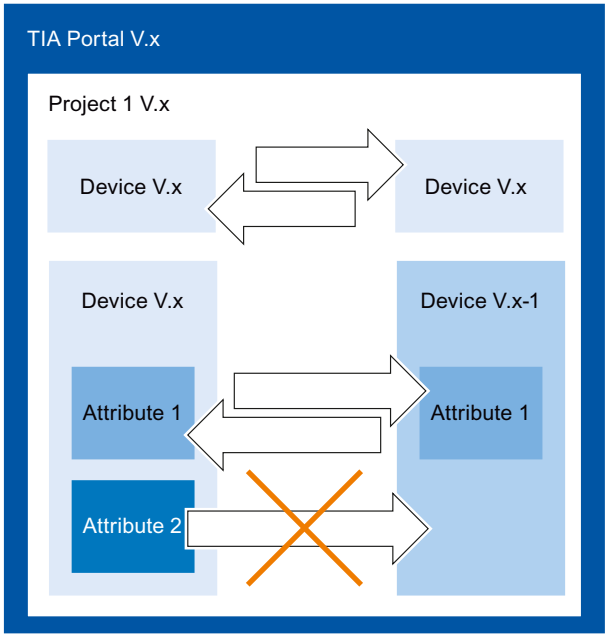
All configurations that are supported in the target version are retained when you copy data between projects of different WinCC versions. Configurations that are not supported by the target version are marked as invalid with a color code.

Copy of control between HMI devices from different device versions

WinCC supports all configurations of a previous WinCC version.

The following applies when copying within different ES versions:

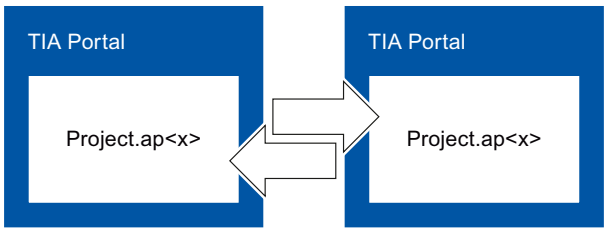
- All the configurations that are also supported in the respective RT version are retained.
- Default settings are defined for configurations that are only supported in the WinCC version of the target project.
- Configurations that are not supported by the respective RT version are marked by color as invalid or are not displayed.
  All properties and settings originally defined in the source HMI device are reactivated when you copy an object back to the source HMI device unchanged.

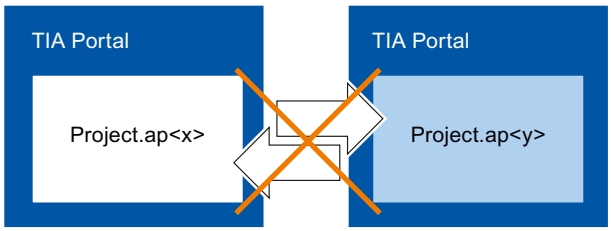


- The HMI device must be valid for the current Runtime version.

Copying between different ES versions

Open a second instance of the TIA Portal to copy between two TIA projects. You can only copy between projects that have the same ES version. The ES version of a project is indicated by the file extension *.ap<version_number>.





10.9.9 Using WinCC version compatibility

10.9.9.1 Basics on version compatibility

Introduction

Edit existing projects as follows with WinCC:

- You edit, compile and download existing projects with the range of functions of the previous version of WinCC. You can continue to edit these projects afterwards with the previous version of WinCC.
- You upgrade existing projects and use the functions of the current WinCC version.

Note

WinCC functions

While editing a project of a previous WinCC version, you can only access the functions and HMI devices of this previous WinCC version.

Versions in WinCC

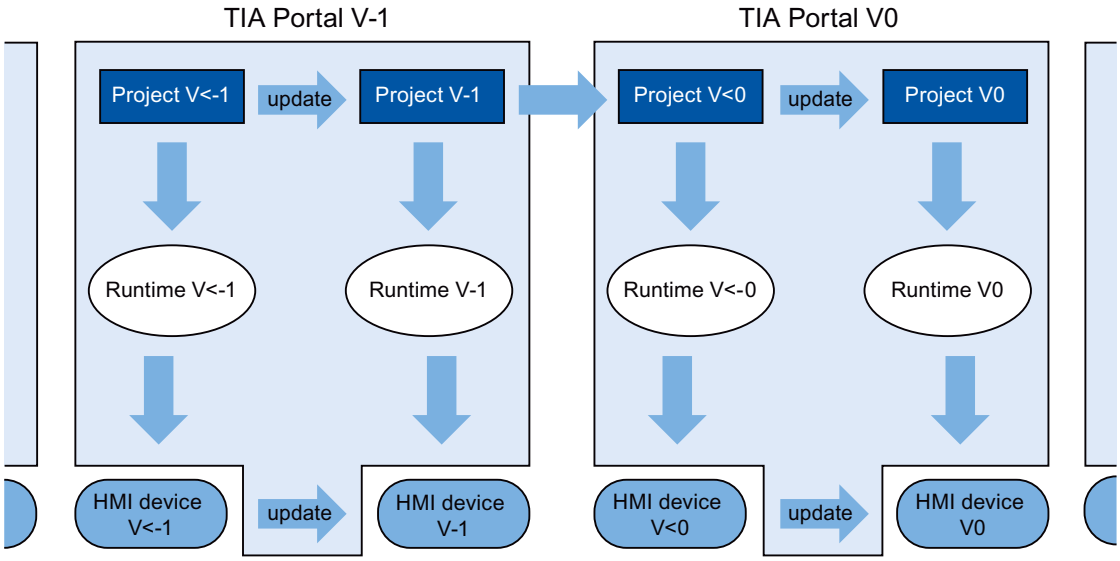
In WinCC, you work with different version types:

- WinCC version
  The WinCC version installed on the configuration PC, for example, WinCC V12.
- Project version
  Projects are created using the WinCC version that is installed on the configuration PC. While you are editing a WinCC project from a previous version in the current WinCC version, the version ID is displayed behind the project name in the project tree.

- Runtime version
  You can configure HMI devices with different runtime versions in WinCC. You specify the runtime version once for an HMI device. The device version must match the Runtime version.
- Device versions
  Depending on the used HMI device, the image is a combination of the operating system and / or runtime software. For each HMI device, WinCC provides various images which can be loaded onto the HMI device, if necessary, according to the configuration. The device version corresponds to a specific image. The device version must match the configuration.

Compatibility of WinCC versions, Runtime versions and device versions

The figure below shows the interaction of versions in the TIA Portal:



Creating projects

After you create a new project in WinCC, open and edit it in the WinCC version in which you created it.

Saving

To save a project of a predecessor WinCC version in this version again, save it in the usual way. If you manually upgrade the project to your WinCC version, please note that you will no longer be able to open the project in the previous WinCC version.

To save a project of the previous WinCC version in the current version, upgrade the project to your WinCC version. You can then no longer edit the project with a previous version of WinCC.

Compiling, simulating and loading

If you are using a project of a previous WinCC version, you can use your current version to generate Runtime data for this previous version. This also allows you to load HMI devices that are no longer compatible with your WinCC version.

Copying within projects with different WinCC versions

If objects and configurations are also available in the target version, copy these as required via the clipboard or using drag-and-drop.

Opening, editing and saving projects of a previous WinCC version

You can open and edit projects of previous WinCC versions as required. In doing so, you only utilize the functions of the previous WinCC version. Once you have completed editing, you can once again save and edit the project in the previous WinCC version.

Compiling, downloading and simulating projects of a previous WinCC version

You can compile, download and simulate projects of previous WinCC versions as required. Your current WinCC version will provide the Runtimes and device versions for the corresponding WinCC version.

10.9.9.2 Editing projects of a previous WinCC version

Introduction

WinCC provides the option of editing projects of a previous WinCC version. While editing a project of a previous WinCC version, you can only access the functions of this version. In order to use the functions of your current WinCC version for this project, upgrade the project to your WinCC version.

Note

If you upgrade a project to your WinCC version, please note that you will no longer be able to open and edit the project in the previous WinCC version.

Requirement

- A project of a previous WinCC version has been created.
- The current WinCC version is installed on the configuration PC.

Procedure

Proceed as follows to edit a project of a previous WinCC version:

1. Open the project.
2. Edit the project using the functions of the previous WinCC version.
3. Save the project.
4. Compile the project.
5. Download and simulate the project.
6. You can open the project in the previous WinCC version for further editing, if required.

Result

The modified project data can be edited further on a different configuration PC with the previous WinCC version for further processing. The Runtime project was generated and downloaded in the corresponding Runtime version.

10.9.9.3 Upgrading projects

Introduction

If the project version is older than the WinCC version, the version ID is displayed in the project navigation. Your WinCC version also contains the previous version that you can use to edit projects as required. In order to use the functions and options of your WinCC version in a project, upgrade the project to your WinCC version. Then, change your device or runtime version(s) in accordance with the new project version.

Note

WinCC version compatibility

If you upgrade a project to your WinCC version, please note that you will no longer be able to edit it with the previous WinCC version.

Requirements

- The project version is a predecessor of your WinCC version.
- You have write access to your project drive.
- The project drive provides sufficient storage capacity for another project of this size.

Procedure

Proceed as follows to upgrade a project to your WinCC version:

1. Select the project in the project navigation.
2. Select the "Upgrade project" command from the shortcut menu of the project.
   A dialog opens.
3. Click "Confirm".
   The project closes and the progress bar is displayed.

An alarm is output when the project has been upgraded.

Result

- The project has been saved on the project drive in the previous WinCC version and with the corresponding file extension.
- The project is displayed on the project drive in the current WinCC version and with the corresponding file extension.
- The project is displayed without a WinCC version ID in the project navigation.

To make use of the functions of the new WinCC version, change your device or runtime version(s) in accordance with the new project version in the next step.

10.9.9.4 Upgrading a global library

Introduction

In order to process objects of a global library in a project, the global library must be available in the product version of the project. You can upgrade each global library from an older product version to the current product version. You are prompted accordingly when you open the global library.

Requirements

- The version of the global library is a predecessor of your WinCC version.
- You have write access to your project drive.
- All types in the library have been released.

Note

Upgrading a user library

If you want to use a user library from an earlier version of WinCC, you must upgrade it. Make sure that all types of the library have been released. The library being upgraded cannot contain a type with the "in progress" status.

Procedure

Proceed as follows to upgrade a global library from TIA Portal V12.x or lower:

1. Open the global library.
   The "Upgrade global library" dialog box opens.
2. Click "OK".

A copy of the global library is created and upgraded. The copy of the global library receives the name extension "_V13". The global library opens.

Result

The global library is stored with the appropriate file extension.

See also

Basics on version compatibility (Page 8870)

Managing libraries (Page 8767)

10.9.9.5 Changing between device versions

Selection of the device version

When you configure a new HMI device, WinCC automatically selects the latest version of the device.

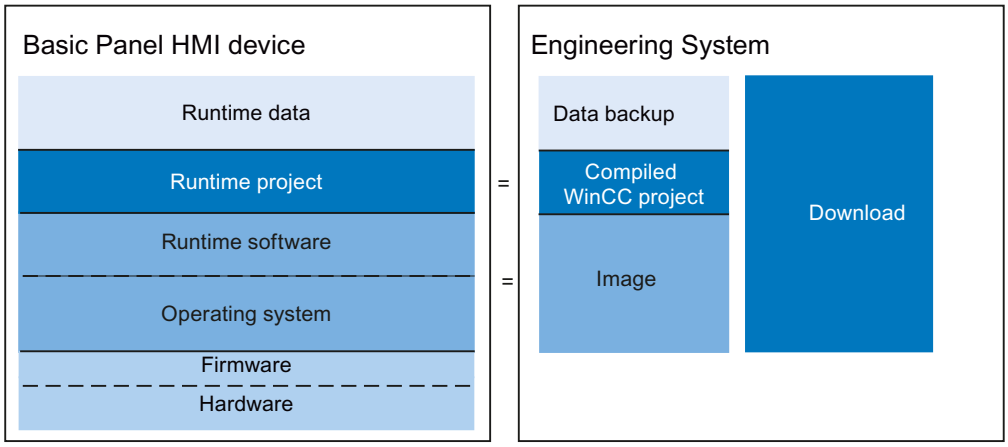
If you want to use a device version other than the one set in WinCC, transfer an image to the HMI device. WinCC provides the images required for the supported HMI devices.

Information on the device versions used in WinCC is available in the FAQs on the Internet, entry ID 21742389.

NOTICE
Changing the device version deletes all data on the HMI device. Data is deleted on the target system if you change the device version. For this reason, you should save existing Runtime data before changing the device version.

HMI device configuration

The following figure shows the software components of an HMI device:



10.9.9.6 Changing the device version

Introduction

Depending on the required Runtime version, select the suitable device version for your configuration.

Note

Selection of device versions

The selection of available devices versions depends on the version of the project.

Requirements

- A project has been created and opened.
- The project contains an HMI device.

Procedure

To change the device version, follow these steps:

1. Double-click on "Devices & Networks" in the project tree.
   The editor opens.
2. Select the required HMI device from the device view.
3. Select "Change Device/Version" in the device shortcut menu of the HMI device.
   A dialog opens.
4. Select the required HMI device.

5. Select the required device version under "Version".
6. Confirm your selection with "OK".

Result

You have changed the device version in the WinCC project.

NOTICE
Changing the device version deletes all data on the HMI device. All data is deleted from the HMI device when you change the device version and compile/download the project. You should therefore backup your Runtime data prior to the download.

10.9.10 Viewing memory card data

10.9.10.1 Basics

Introduction

WinCC provides you with the possibility of viewing data stored on your memory card. The function supports the use of memory cards of the HMI device and of the CPU.

You have the following options:

Viewing a backup (Page 8878)

Renaming and deleting backups (Page 8879)

Auto-Hotspot

Auto-Hotspot

Auto-Hotspot

See also

Viewing a backup (Page 8878)

Renaming and deleting backups (Page 8879)

10.9.10.2 Working with backups

Viewing a backup

Introduction

The backup of a Basic Panel that is stored on a memory card can also be viewed in the TIA Portal.

Requirements

- WinCC is installed.
- A memory card with a backup is available.
- The card reader is connected to the configuration PC.
- The project view is open.

Backup on the memory card in the card reader

1. Insert the memory card into the card reader.
2. Open "SIMATIC Card Reader" in the project navigation.
3. Select the card reader drive.
   The "Online Card Data" folder is displayed.
4. Open the "Online Card Data" folder
5. Click the backup to open the shortcut menu.
6. Select "Properties".

Backup on the memory card of the PLC

Proceed as follows if the backup is stored on the memory card of the PLC:

1. Connect the PLC with the configuration PC.
2. Click on the PLC in the project navigation.
3. Select "Connect online" from the shortcut menu.
   A connection to the PLC is established.
   Once the PLC is connected, the "Online Card Data" folder is displayed.

4. Open the "Online Card Data" folder.

Note

Accessing a password-protected PLC

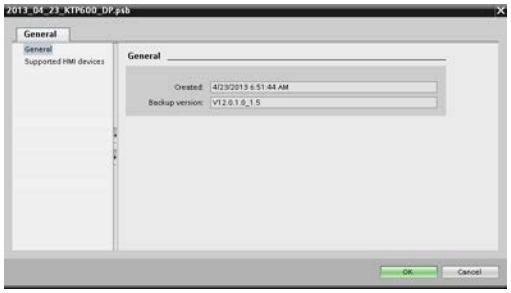
When you attempt to access a PLC that is protected by a password, you will be prompted to enter the password.

You need at least read access rights in order to view the data that is stored on the memory card.

5. Click the backup to open the shortcut menu.
6. Select "Properties".

Result

The backup properties are displayed in a separate dialog.



See also

Basics (Page 8877)

Renaming and deleting backups (Page 8879)

Renaming and deleting backups

Introduction

You may rename and delete backups from a memory card in the project navigation of the TIA Portal.

Requirements

- WinCC is installed.
- The card reader is connected to the configuration PC.
  The PLC is connected online with the configuration PC.
- A memory card with a backup is available.

- The project view is open.
- The backup is displayed in the project navigation.

Note

Accessing a password-protected PLC

When you attempt to access a PLC that is protected by a password, you will be prompted to enter the password.

You need write access rights to rename or delete memory card data.

Procedure

1. Click on the backup in the project navigation.
2. Open the shortcut menu.
3. Select "Rename" to rename the file.
4. Enter a new name.
5. Select "Delete" to delete the file.

Result

The backup file is now renamed or deleted.

See also

Basics (Page 8877)

Viewing a backup (Page 8878)

10.10 Compiling and loading

10.10.1 Compiling and loading projects

10.10.1.1 Overview of compiling and loading projects

Overview

The project is compiled in the background even as you are configuring it in WinCC. This reduces the time for final compilation. When you start compilation, you create a file that can be run on the corresponding HMI device.

If an error occurs during compilation, WinCC provides support in locating and correcting it.

Once you have corrected any problems, you download the compiled project to the HMI devices on which the project is to run.

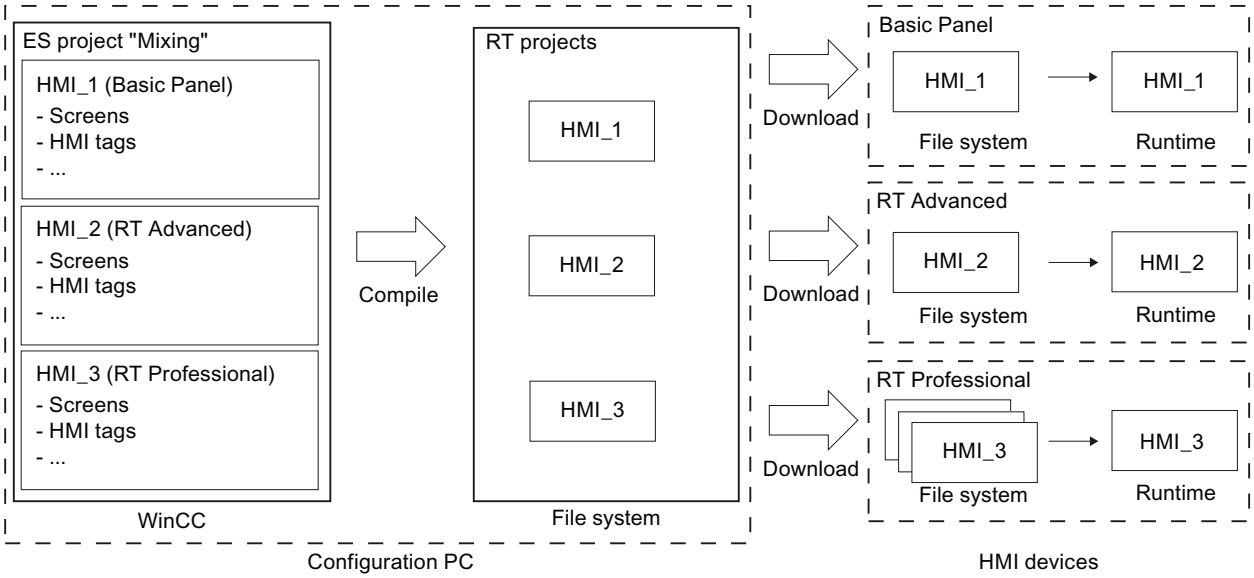
If you are using HMI tags in your project that are connected to PLC tags, you should also compile all modified S7 blocks with the command "Compile > Software" in the shortcut menu before you compile the HMI device.

Definition of terms

The term "project" has two different meanings in the contexts of compilation and loading. "Project" is the WinCC project on the configuration PC. "Project" is also the Runtime project you create by compiling the configuration data of an HMI device and download to the HMI device.

- WinCC project: contains the configuration data of one or more HMI devices
- Runtime project: contains the compiled configuration data of an HMI device

The figure below illustrates the link between WinCC projects and Runtime projects using the example of the "Compile and load" process:



Runtime version

The Runtime version depends on the image of the configured HMI device. The Runtime version of the compiled project is displayed under "Info" in the Inspector window.

10.10.1.2 Compiling a project

Introduction

The changes made to the project are compiled in the background even as you are configuring a project in WinCC. Projects are compiled automatically when you load them. This ensures that the latest version of the project is loaded at all times.

WinCC checks consistency of the project during compilation. The error locations in the project are listed in the Inspector window. You can jump directly to the source of the error from the entry in the Inspector window. Check and correct errors found.

Scope of the compilation

Configuration data is compiled in the background as soon as you start configuring an HMI device. If you compile a project manually, only the changes in the configuration made since the last compilation process are compiled in the background.

You can start complete project compilation manually at any time; this may, for example, be done to test the consistency of the configured data.

Requirement

- A project is open.

Procedure

Proceed as follows to compile a project:

1. If you want to compile several HMI devices at the same time, select all the relevant HMI devices with multiple selection in the project tree.
2. Compile the project:
   - To only compile changes in the project, select the "Compile > Software (only changes)" command from the shortcut menu of the HMI device.
   - To compile all project data, select the "Compile > Software (compile all)" command from the shortcut menu.

Result

The configuration data of all selected HMI devices is compiled. Any errors that occur during compilation are shown in the Inspector window.

10.10.1.3 Loading projects

Overview for loading of projects

Overview

Delta data of the project is automatically compiled before you download it to one or several HMI devices. This always ensures that the latest version of the project is transferred.

Loading a project to an HMI device

The following steps are completed prior to downloading:

1. The download settings are verified. The "Extended loading" dialog box is opened automatically during the initial download of a project to an HMI device. You use this dialog to define the protocol and interface or destination path for the project in accordance with the HMI device Runtime used.
   You can call the "Extended loading" dialog at any time with the menu command "Online > Advanced download to device...".
   The "Load preview" dialog opens.
2. The project is compiled. Warnings and errors during compilation are displayed in the Inspector window and in the "Load preview" dialog.
3. The "Load preview" dialog shows you the following information for each HMI device:
   - The individual steps for loading
   - If the image of the target HMI device does not match the image from the configuration, a prompt is displayed asking whether you want to now change the image.

NOTICE
Changing the image deletes all data from the HMI device. Data is deleted on the target system if you change the image. For this reason, first back up the following data, if necessary: • User administration • Recipes

   - Presettings that take effect at loading. You can change the default settings for this download process, if necessary.
   - Warning events (optional). You can download a project while ignoring the "warnings". The functionality may be restricted in runtime.
   - Error events (optional). You cannot load the project. Eliminate the errors and then reload the project.
     WinCC will open the invalid configuration in the corresponding editor if you double-click the error message in the Inspector window. Correct the errors and reload the project.

If you are using HMI tags in your project that are connected to controller tags, you should also compile all modified S7 blocks with the command "Compile > Software" in the shortcut menu before you compile the HMI device.

Loading with S7 routing

Configure the S7 routing settings in the "Devices & Networks" editor in the relevant controller. The settings depend on the device configured.

S7 routing supports the following protocols:

- MPI/PROFIBUS
- Ethernet

Transferring Runtime add-ons

Projects can contain Runtime add-ons in the form of controls or CSP (Communication Support Packages). These Runtime add-ons are automatically transferred with the project.

Loading a project

Introduction

Before a project can run on an HMI device, you must first load it to the HMI device. During loading, you must most importantly specify whether existing data on the HMI device such as "user administration" and "recipe data" is to be overwritten.

If the HMI device supports PROFINET, the name of the HMI device entered in the project navigation is used as the device name for PROFINET communication. The name is written to the HMI device during loading. If a device name for the PROFINET communication has already been entered in the HMI device, it will be overwritten.

As a general rule, only one project can be active in runtime on an HMI device. An HMI device is generally configured to exit Runtime automatically when loading is started. If this is not the case, you will have to exit Runtime manually on the HMI device.

If the image of the target HMI device does not match the image from the configuration, a prompt is displayed asking whether you want to now change the image.

NOTICE
Changing the image deletes all data from the HMI device. Data is deleted on the target system if you change the image. For this reason, first back up the following data, if necessary: • User administration • Recipes

Please refer to the documentation for the HMI device used for more detailed information on transfer settings.

Note

Ending Runtime automatically

If automatic transfer is enabled on the HMI device and a transfer is started on the configuration PC, the running project is automatically stopped.

The HMI device then switches autonomously to "Transfer" mode.

After the commissioning phase, disable the automatic transfer function to prevent the HMI device from switching inadvertently to transfer mode.

Transfer mode can cause undesired reactions in the system.

To block access to the transfer settings and thus avoid unauthorized changes, assign a password in the Control Panel.

Requirement

- You have created an HMI device in the project.
- The HMI device is connected to the configuration PC.
- Transfer mode is set in the HMI device.

Procedure

Proceed as follows to load a project:

1. To download a project simultaneously to several HMI devices, select the HMI devices by means of multiple selection in the project navigation.
2. Select the "Download to device > Software" command from the shortcut menu of the HMI device.
3. If the "Extended loading" dialog is open, configure the "Settings for loading". Make sure that the "Settings for loading" correspond to the "Transfer settings in the HMI device".
   - Select the protocol used, for example, Ethernet or HTTP.
   - Configure the relevant interface parameters on the configuration PC.
   - Make any interface-specific or protocol-specific settings required in the HMI device.
   - Click "Download".

   You can call the "Extended loading" dialog at any time with the menu command "Online > Advanced download to device...".
   The "Load preview" dialog opens. The project is compiled at the same time. The result is displayed in the "Load preview" dialog.
4. Check the displayed presettings and change them as necessary.
5. Click "Download".

Result

The project is loaded to all selected HMI devices. Any existing project is replaced. The data for user administration and / or recipes is replaced in accordance with the settings in the "Load preview" dialog.

During the download, you can keep track of the files that are transferred.

If errors or warnings occur during the download, corresponding alarms are displayed under "Info > Load" in the Inspector window.

On completion of the successful download of the project, you can execute it on the HMI device.

Note

If the transfer is interrupted, WinCC automatically ensures that no data is lost and that existing data is deleted on the HMI device only after complete transmission.

See also

Backing up and restoring data of the HMI device (Page 8895)

Updating the HMI device operating system (Page 8897)

Error messages during loading of projects (Page 8899)

Adapting the project for another HMI device (Page 8900)

Establishing a connection to the HMI device (Page 8901)

10.10.1.4 Runtime start

Starting Runtime on the HMI device

Introduction

On completion of the project download to the HMI device, you can start the project in Runtime. The project is saved in the HMI device to a file with the following extension:

- Basic Panels as well as OP 73, OP 77A and TP 177A: "*.srt"

The project settings defined in the "Runtime settings" of the HMI device are activated when the project is started in Runtime.

The programs that you can use to start projects on the HMI device are available in the Runtime installation folder.

Note

Ending Runtime automatically

If automatic transfer is enabled on the HMI device and a transfer is started on the configuration PC, the running project is automatically stopped.

The HMI device then switches autonomously to "Transfer" mode.

After the commissioning phase, disable the automatic transfer function to prevent the HMI device from switching inadvertently to transfer mode.

Transfer mode can trigger unwanted responses in the plant.

To block access to the transfer settings and thus avoid unauthorized changes, assign a password in the Control Panel.

Requirement

WinCC Runtime is installed on the HMI device.

Procedure

The "RT Loader" application is started on a panel. The project loaded is started automatically after expiration of the configured delay.

If the project does not start automatically:

1. To select the project file, click "Settings" and enter the path to the project file under "Configuration file".
2. Click "OK" and then "Start".

10.10.2 Simulating projects

10.10.2.1 Simulation basics

Introduction

You can use the simulator to test the performance of your configuration on the configuration PC. This allows you to quickly locate any logical configuration errors before productive operation.

You can start the simulator as follows:

- In the shortcut menu of the HMI device, or in a screen: "Start simulation"
- Menu command "Online > Simulation > [Start|With tag simulator|With script debugger]"
- Under "Visualization > Simulate device" in the portal view.

Requirements

The simulation/runtime component is installed on the configuration PC.

Field of application

Use the simulator to test various functions of the operator control and monitoring system, such as:

- Checking limit levels and alarm outputs
- Consistency of interrupts
- Configured interrupt simulation
- Configured warnings
- Configured error messages
- Check of status displays
- Interconnection and screen layout

See also

Simulating a project (Page 8888)

10.10.2.2 Simulating a project

Introduction

You simulate your project with one of the following two methods:

- Without a connected PLC
  You change the value of area pointers and tags in a tag simulator that is read for the simulation of WinCC Runtime.
- With a connected PLC without a running process
  You simulate your project by running it directly in Runtime. The tags and area pointers become active. This allows you to create an authentic simulation of your configured HMI device in Runtime.

Note

Simulation restrictions

You cannot simulate the following system functions:

- CalibrateTouchScreen

You cannot simulate the Media Player. A static screen appears in the simulation window instead of the Media Player.

File access via scripts is not possible for HMI devices with Windows CE.

Requirement

- Simulation without a connected PLC: Tags have been created
- Simulation with a connected PLC but no active process: A project with tags and area pointers has been created

Procedure

To simulate a project using the tag simulator, follow these steps:

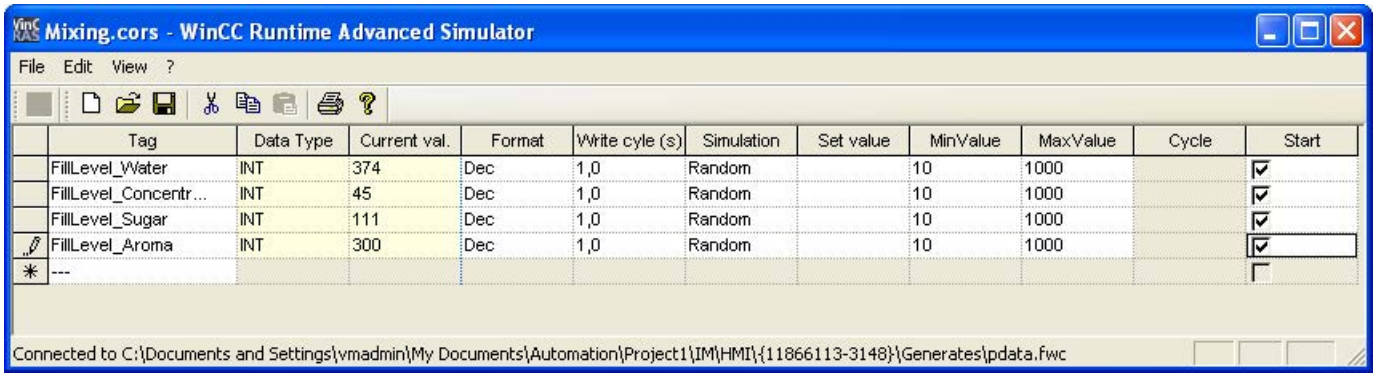
1. Open the project on the configuration PC.
2. Select the "Online > Simulation > With tag simulator" menu command.
   For initial project simulation, the simulator is started with a new, empty table. The project is opened simultaneously in Runtime.
   Toggle between the tag simulator and Runtime using the <Alt +Tab> key combination.
3. To simulate a process value, select the corresponding "tag" from the tag simulator.
   The table lists all configured tags. You can simulate up to 300 tags simultaneously.
4. Select the simulation mode in the "Simulation" column.
5. Change the value of tags and area pointers in the respective columns.
6. Activate the "Start" check box to start the simulation for this tag.
7. To save the simulation, select the menu command "File > Save" and enter a descriptive name, for example, "Mixing".
   The file name is assigned the extension "*.cors".

Result

The process values are simulated in Runtime. The tag values are created at random, or incremented, depending on the simulation mode.

To specify tag values, change the simulation mode to "<Display>" and enter a value at "Set value".

The following figure shows a tag simulator with four tags whose values can be determined at random in a range of values from 10 to 1000:



Managing simulation data

If you have saved data from a previous simulation, you can open the file at a later point in time and simulate your project again. The tags and area pointers listed in the tag simulator must still be available in the project.

Proceed as follows to open a simulation file:

1. Select the menu command "Online > Simulate Runtime > With tag simulator".
2. Select the menu command "File > Open" in the tag simulator.
3. Select the corresponding simulation file and click "Open".
   The simulator loads the stored data.

Enabling and disabling tags

Start and stop the simulation for each tag separately in order to facilitate the transition from offline to online engineering. Activate "Start" in the corresponding row.

If a tag is activated, the simulation values are calculated and transferred to the WinCC simulator.

Deleting a tag

To delete a tag from the tag simulator, follow these steps:

1. Select the cell that contains the tag name.
2. Select the "Edit > Cut" menu command.
   The tag is removed from the table.

See also

Simulation basics (Page 8887)

Working with the tag simulator (Page 8890)

10.10.2.3 Working with the tag simulator

About the tag simulator

The tag simulator has the following columns:

Column	Description
Tag	Specifies the tags for the simulation.
Data type	Shows the data type of the selected tag.
Current value	Shows the simulated value of the defined tags.
Format	Specifies the selected format in which the tag values are simulated: • Decimal (1, 2, 3, 4, ...) • Hexadecimal (03CE, 01F3, ...) • Binary (0 and 1)

Column	Description
Write cycle	Specifies the selected time interval at which the current tag values are simulated. If you enter "2", for example, the current value of the tag will be shown every 2 seconds.
Simulation	Shows the method by which the tag values are processed during simulation.
Set value	Sets the selected value for the respective tag. The simulation start with the specified value.
minValue maxValue	Specifies the value range of the tag. You set a minimum and maximum value for this range. The default values are -32768 for the minimum and 32767 for the maximum.
Period	Contains the period during which the value of the tag is repeated for the "Increment" and "Decrement" simulation modes.
Start	Starts simulation of the tag based on the previously entered information.

Simulation modes

The simulator has six different simulation modes. The configured tags are supplied with nearly realistic values during the simulation.

Simulation mode	Description
Sinusoidal	Changes the tag value to form a sinusoidal curve. The value is visualized as a periodic, non-linear function.
Random	Provides randomly generated values. The tag value is changed by means of a random function.
Increment	Increases the value of the tag continuously up to a specified maximum value. Begins again at the minimum after the maximum has been reached. The value trend corresponds to a positive saw-tooth curve.
Decrement	Reduces the value of the tag continuously down to a specified minimum value. Begins again at the maximum after the minimum has been reached. The value curve corresponds to a negative saw-tooth curve.
Shift bit	Shifts a set bit continuously by one position. The previous position is always reset. This lets you test the alarms of an HMI device, for example.
<Display>	The current tag value is displayed statically.

Example: Simulate tags with the "Shift bit" simulation mode

Proceed as follows to simulate tags with the "Shift bit" simulation mode:

1. Open the project you want to simulate.
2. Select the menu command "Online > Simulate Runtime > With tag simulator".
   The tag simulator opens.
3. In the "Tag" column, select a tag from your project.
4. Select "Bin" in the "Format" column.
5. Enter the value "1" in the "Write cycle" column.
6. Select the "Shift bit" simulation mode in the "Simulation" column.
7. Enter the value "1" in the "Set value" column.
8. Enable the tag with the "Start" check box.

Result

The simulator tests the selected tag bit-by-bit as follows:

Simulation values	Byte for alarms
Set start value	00000001
1. Simulation value	00000010
2. Simulation value	00000100
3. Simulation value	00001000
....	...

In Runtime you see if the desired alarm is output at a given value.

See also

Simulating a project (Page 8888)

10.10.2.4 Simulation restrictions

Alarms with dynamic parameters

If you use tags or text lists as external tags for alarms, the dynamic parameters of the alarms are not displayed.

Only internal tags are enabled for the simulation of alarms in the tag simulator .

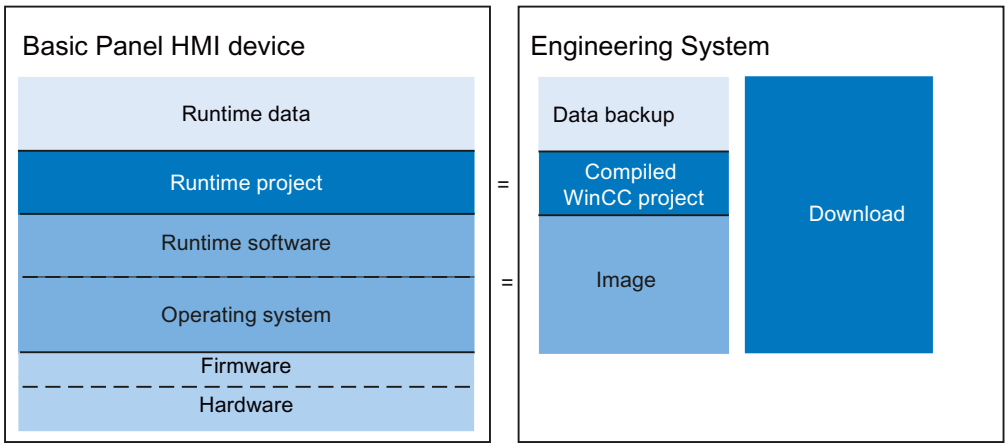
You can use PLCSim to simulate dynamic parameters.

10.10.3 Servicing the HMI device

10.10.3.1 Overview of HMI device maintenance (Basic Panels)

Structure

The following figure shows the software components of an HMI device and their relation to the engineering system.



Runtime data

The Runtime data is generated during operation of the plant and stored on the HMI device. This data includes, for example, recipes and data for the user administration. This data is overwritten during loading. If required, save this data before loading a Runtime project.

Runtime project

The Runtime project contains the compiled configuration data for an HMI device. You download the Runtime project from WinCC to the HMI device.

Runtime software and operating system

Together, the Runtime software and operating system of an HMI device form the image. Various images are available for the HMI device. All images of an HMI device are available in WinCC. Depending on the configuration, download the appropriate image along with the Runtime project to the HMI device as required.

Firmware and hardware

The HMI device is delivered with preconfigured firmware and hardware.

10.10.3.2 ProSave

Introduction

The "ProSave" service tool is installed by default when WinCC is installed. The ProSave functions are accessed in WinCC with the menu "Online > HMI Device maintenance".

Functional scope

ProSave provides all of the functions needed to transfer files to the HMI device.

- Data backup and restoration of backed-up data
- Operating system update
- Communication settings

See also

Backup of HMI data (Page 8894)

Updating the operating system (Page 8896)

Overview of HMI device maintenance (Basic Panels) (Page 8892)

10.10.3.3 Backup of HMI data

Introduction

Regular backups of HMI device data keep downtimes to a minimum, for example, when you replace a device. You simply transfer the backup data to the HMI device, restoring the original status.

Data backup with WinCC

If an HMI device is connected to the configuration PC, you can back up and restore HMI device data from the configuration PC using WinCC.

Scope of data backup

Which data is backed up and restored depends on the type of HMI device:

- Complete backup.
  Depending on the HMI device: Runtime, firmware, operating system, configuration, recipes, user administration, settings
- Recipes only
- User administration only
- Recipes as CSV file

A backup file with the extension *.psb is generated when you backup the data of an HMI device.

As a general rule, you can backup the data to any storage medium. If the HMI device is networked, you can also backup the data to a server.

Note

Scope of data backup

Data backup secures the contents of the flash memory. Note the following when performing a complete data backup and restore operation for HMI devices:

- In the event of a full restore of all data, all data previously on the device, including the operating system, is deleted irrevocably.
- If the data restoration was interrupted, execute the command "Reset to factory settings". Restart data restoration.

See also

Backing up and restoring data of the HMI device (Page 8895)

ProSave (Page 8893)

Overview of HMI device maintenance (Basic Panels) (Page 8892)

10.10.3.4 Backing up and restoring data of the HMI device

Note

Use the restore function for project data only on operating devices which were configured using the same configuration software.

Requirement

- The HMI device is connected to the configuration PC
- The HMI device is selected in the project tree.
- If a server is used for data backup: The configuration PC has access to the server

Backup of the data of the HMI device

Proceed as follows to backup the data of the HMI device:

1. Select the "Backup" command from the "Online > HMI Device maintenance" menu. The "Create backup" dialog box opens.
2. Select the type of the PG/PC interface and the target device, and click "Create". The "SIMATIC ProSave" dialog box opens.
3. Select the data to backup for the HMI device under "Data type".
4. Enter the name of the backup file under "Save as".
5. Click "Start Backup".

This starts the data backup. The backup operation takes some time, depending on the connection selected.

Restoring the data of the HMI device

Proceed as follows to restore the data of the HMI device:

1. Select the "Restore" command from the "Online > HMI Device maintenance" menu.
   The "Restore backup" dialog box opens.
2. Select the type of the PG/PC interface and the target device, and click "Load".
   The "SIMATIC ProSave" dialog box opens.
3. Enter the name of the backup file under "Save as".
   Information about the selected backup file is displayed under "File information".
4. Click "Start Restore".

This starts the restoration. This operation takes some time, depending on the connection selected.

Backup/Restore from the "Backup/Restore" dialog in the control panel of the HMI device

The "Backup/Restore" function is enabled for MMC, SD memory cards and USB mass storage devices.

See also

Backup of HMI data (Page 8894)

Overview of HMI device maintenance (Basic Panels) (Page 8892)

10.10.3.5 Updating the operating system

Introduction

If the image of an HMI device has a version that does not match the configuration, it is automatically updated, after prompting, during the loading of the project. Loading will then continue. If you decline to update the image, loading is canceled. Updating the image also updates the operating system of the HMI device.

Updating the image

Connect the HMI device to the configuration PC to update an image. If possible, you should use the interface with the highest bandwidth for this connection, such as Ethernet. Updating the image via a serial connection can take up to an hour.

"Reset to factory settings"

If the operating system on the HMI device is no longer operational, update the operating system and reset the HMI device to the factory settings.

Note

You will require the following to reset to the factory settings via Ethernet:

- the MAC address of the HMI device
- the available IP address
- the programming device/PC interface of the configuration PC set to Ethernet TCP/IP

The programming device/PC interface is configured using the control panel of the configuration PC. Select" "S7ONLINE (STEP7) -> TCP/IP" in the "Access point of the application" field.

See also

Updating the HMI device operating system (Page 8897)

ProSave (Page 8893)

Overview of HMI device maintenance (Basic Panels) (Page 8892)

10.10.3.6 Updating the HMI device operating system

When you update the operating system, the Runtime software on the HMI device is also updated and the device version is changed.

NOTICE
An operating system update deletes all the data from the HMI device The operating system update deletes all data from the target system. For this reason, you should back up the following data beforehand: • User administration • Recipes

Requirement

- The HMI device is connected to the configuration PC or the PC with ProSave.
- The HMI device is selected in the project navigation.

Updating the operating system

Proceed as follows to update the operating system:

1. Select the "Update operating system" command from the "Online > HMI device maintenance" menu.
2. Select the "Update operating system" command from the menu under "Online > HMI Device maintenance" on the configuration PC in WinCC.
   The "SIMATIC ProSave [OS-Update]" dialog opens. The path to the image is preset.
3. If required, you can select a different path for the image that you want to transfer to the HMI device.
4. Click "Update OS".

This starts the update. The update operation can take time, depending on the connection selected.

Resetting the HMI device to factory settings

To reset the HMI device to factory settings, proceed as follows:

1. Switch off power to the HMI device.
2. Select the "Update operating system" command from the menu under "Online > HMI Device maintenance" on the configuration PC in WinCC.
   The "SIMATIC ProSave [OS-Update]" dialog opens. The path to the image is preset.
3. If required, you can select a different path for the image that you want to transfer to the HMI device.
4. Activate "Reset to factory settings".
5. Click "Update OS".
6. To start "Reset to factory settings", switch on power to the HMI device again.

This operation can take time.

Result

The operating system of the HMI device is updated to the latest version.

See also

Updating the operating system (Page 8896)

Overview of HMI device maintenance (Basic Panels) (Page 8892)

10.10.4 Reference

10.10.4.1 Error messages during loading of projects

Possible problems during the download

When a project is being downloaded to the HMI device, status messages regarding the download progress are displayed in the output window.

Usually, problems arising during the download of the project to the HMI device are caused by one of the following errors:

- Incorrect download settings on the HMI device
- Incorrect HMI device type in the project
- The HMI device is not connected to the configuration PC.

The most common download failures and possible causes and remedies are listed below.

The serial download is cancelled

Possible remedy: Select a lower baud rate.

The download is cancelled due to a compatibility conflict

Possible cause	Remedy
The configuration PC is connected to the wrong device, e.g. a controller.	Check the cabling. Correct the communication parameters.

Project download fails

Possible cause	Remedy
Connection to the HMI device cannot be established (alarm in the output window)	Check the physical connection between the configuration PC and the HMI device. Check whether the HMI device is in transfer mode. Exception: Remote control
The default communication driver is not listed in the Windows Device Manager.	Check the device status of the COM connection in the properties window of the Device Manager.

Download over MPI/DP interface fails

Possible cause	Remedy
"Configured mode" is set on the CP, for example, if you are using the SIMATIC NET CD.	Set the CP to "PG mode" using the "Set PC station" application. Check the "baud rate" and "MPI address" network parameters. Download the project from WinCC to the CP. Set the CP back to "configured mode".
On the programming device/PC panel, the "S7ONLINE" access point is not set to a hardware device such as CP5611 (MPI). The cause may be the installation of "SIMATIC NET CD 7/2001".	Set the access point "S7ONLINE" on the selected device using the "PG/PC Panel" or "Set PC station" application. Check the "baud rate" and "MPI address" network parameters. Download the project from WinCC to the HMI device. Restore the "S7ONLINE" access point to the original device.

The configuration is too complex

Possible cause	Remedy
The configuration contains too many different objects or options for the HMI device selected.	Remove all objects of a type, e.g. all graphic views.

10.10.4.2 Adapting the project for another HMI device

Introduction

When you download a WinCC project to an HMI device, WinCC checks whether this is compatible with the HMI device type used in the project. If the types of HMI device do not match, you will see a message before the download starts.

The download is aborted.

Adapting the project for the HMI device

You need to adapt the project accordingly to be able to download the project to the connected HMI device.

- Add a new HMI device in the project tree. Select the correct type of HMI device from the HMI device selection.
- Copy the configured components from the previous to the new HMI device.
  Copy large amounts of components directly in the project navigation and details view.
  For example, copy the "Screens" folder to the screens folder of the new HMI device with the help of the shortcut menu.

- Use the detail view to copy entries in the project tree for which the "Copy" command is not available in the shortcut menu.
- Select the "Recipes" entry in the project tree, for example. The recipes are displayed in the detail view.
- Select the recipes in the detail view and drag them to the "Recipes" entry of the new HMI device. The recipes are copied. You can also select multiple objects in the detail view.
- Configure the components that cannot be copied, e.g. connections, area pointers, and alarms.
- Save the project at various points in time.
- Compile the full project.
- When the compilation is successfully completed, download the project to the HMI device.

Linking references

References to linked objects are included in the copying. The references are linked again once the linked objects are copied.

Example:

You copy a screen in which objects are linked to tags. The tag names are entered at the individual objects after the screen is added to the new HMI device. The tag names are marked in red because the references are open. When you then copy the tags and insert them into the new HMI device, the open references are closed. The red marking for the tag names disappears.

For complete references to connected objects in the PLC, you first need to configure a connection to the PLC.

Using the information area

When you compile the project for the HMI device, errors and warnings are displayed in the "Info" tab of the Inspector window. You can use the shortcut menu command "Go to" to go directly to the location where the error or warning can be corrected.

Work through the list of errors and warnings from top to bottom.

When the compilation is successfully completed, download the project to the HMI device.

10.10.4.3 Establishing a connection to the HMI device

Introduction

To download a WinCC project to an HMI device, a properly configured connection must be set up between the configuration PC and HMI device. The connection cannot be set up, the download is cancelled.

Setting up a connection between the configuration PC and HMI device

1. Check the cable connection between the HMI device and configuration PC.
2. Open the "Devices & Networks" editor in WinCC and start the network view.
3. Select the subnet in the network view and check the settings for the subnet.
4. Select the interface of the HMI device in the network view or device view and check the connection parameters in the Inspector window.
5. Switch on the HMI device and press the "Control Panel" button in the loader.
   The Control Panel opens.
6. Press "Transfer" twice in the Control Panel.
   The "Transfer Settings" dialog box opens.
7. Check the settings and then press "Advanced".
   The [Protocol*] Settings" dialog opens.
   *: The title of the dialog depends on the protocol used, for example, "PROFIBUS Settings".
8. Check the advanced settings and close the dialog with "OK".

Important settings

Check the connection settings and in particular the following parameters:

- Network and station addresses
- Selected transmission rate
- Master on the bus; as a general rule, only one master is permitted.

If using a configurable adapter for the connection, check the adapter settings, for example, transmission rate, master on the bus.

10.11 Operating in Runtime

10.11.1 Basics

10.11.1.1 Overview

Configuration and process control phases

HMI devices are used to operate and monitor tasks in process and production automation. The plant screens on the HMI devices provide a clear overview of active processes. The HMI device project, which includes the plant screens, is created during the configuration phase.

Transfer the project to the HMI device for the process control phase. Another requirement for the process control phase is that the HMI device must be connected online to a PLC. The process control phase, operator control, and monitoring can then be carried out during an ongoing work process.

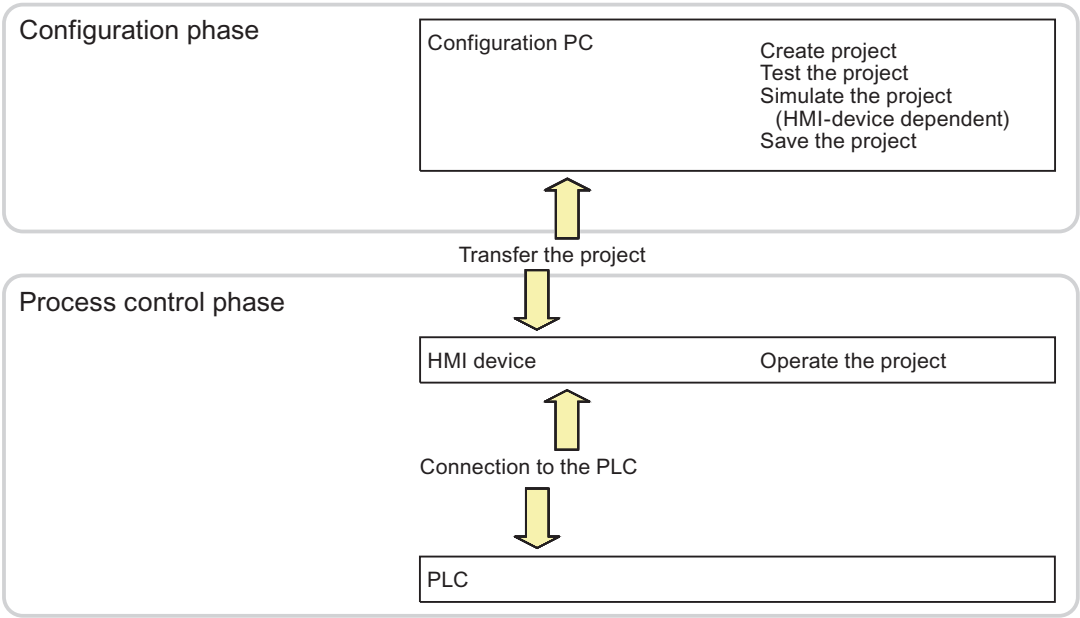


Figure 10-3 Configuration and process control phases

Downloading the project to the HMI device

The following procedures are available to download a project to an HMI device:

- Downloading from the configuration PC
- Restore the project from a PC using ProSave
  In this case, an archived project is downloaded from a PC to the HMI device. The configuration software need not be installed on this PC.

These procedures are available for commissioning and recommissioning a project.

Commissioning and Recommissioning

- When the HMI device is commissioned there is no project at first.
  The HMI device is also in this state after the operating system has been updated.
- When recommissioning, any project on the HMI device is replaced.

10.11.1.2 Tags in Runtime

Definition

Tags correspond to defined memory areas on the HMI device, to which values are written and/or from which values are read. This action can be initiated by the PLC, or by the operator at the HMI device.

10.11.1.3 System functions in Runtime

Application

System functions are used in Runtime for the following purposes:

- To control the process.
- To utilize the properties of the HMI device.
- To configure system setting on the HMI device in online mode.

Each system function in WinCC is logically linked with an object and an event. As soon as the event occurs, the system function is triggered.

System functions

These default system functions are used to implement many tasks in Runtime, such as:

- Calculations, e.g. increasing a tag value by a specific or variable amount.
- Logging functions, e.g. starting a process value log.
- Settings, e.g. PLC changes, or setting a bit in the PLC.
- Alarms, e.g after a different user logs on.

Events

The object and the selected function determine the event that can be defined as a trigger for executing a system function.

For example, the "Change value", "Low limit violated" and "Upper limit exceeded" events are associated with the "Tag" object. The "Loaded" and "Cleared" events are associated with the "Screen" object.

10.11.2 Commissioning projects

10.11.2.1 Settings in the Runtime software

Open the WinCC configuration software and make the following settings for the runtime software:

Display on the PLC

In WinCC, configure the visual representation of the generated project in runtime. The screen resolution is fixed for Basic Panels. Scroll bars will appear if the screen is larger than the configured screen resolution.

To switch off the taskbar, select the Start menu command "Settings > Taskbar". In the "Properties of the Taskbar" dialog, disable the options "Always on top" and "Auto hide".

Dialog fonts

The dialog text will be shown in the standard font. Define the default font under "Language & Font" in the "Device Settings" editor.

Disabling program switching

You lock program switching to prevent operators from calling other applications in Runtime.

Click "Window" in the "Device Settings" editor and select the option "Disable program switching". Also hide the Windows taskbar.

Note

Stop runtime

Always configure a softkey or button for calling the "StopRuntime" system function if you lock program switching. Otherwise, you will not be able to close either runtime or Windows.

Screen saver

A screensaver is no longer required for most modern screens and can, in fact, even cause damage. These monitors switch to hibernate mode as soon as the video signal has not changed for a specified time. A conventional screensaver would prevent this and thus reduce the service life of your monitor.

Note

Approved screensavers

If you do want to use a screensaver, note that only the standard Windows screensavers are approved for use in Runtime.

Make sure that the correct time zone is set on the PC on which the runtime software is installed. To set the time zone in Windows, select Start > Settings > Control Panel > Date and Time.

10.11.2.2 Loading projects

Overview

Various scenarios are possible for loading the project:

- The Runtime software is installed on the same system as the configuration software.
- The Runtime software and the configuration software are installed on different systems. The project must be loaded from the configuration computer to the target system. The HMI devices must be connected to the configuration PC for the transfer. Another requirement is that the transfer mode must match on the HMI devices and in WinCC.

Note

Security prompts may appear during the loading process, depending on the configuration. The recipe data and password list on the HMI device are overwritten following a prompt.

The configuration software and the Runtime software are installed on the same system

If the configuration software and the Runtime software are installed on the same system, proceed as follows:

1. Create and compile your project.
2. Start Runtime directly from the active configuration software. Select the "Start Runtime" command from the "Online" menu.
3. You may test and operate the project online with the controller if you have configured the corresponding communication.

The configuration software and the Runtime software are installed on different systems

If the configuration software and the Runtime software are installed on different systems, proceed as follows:

1. Create and compile your project. For additional information, refer to "Compiling a project".
2. To download the file via cable:
   Connect the HMI device to the configuration computer using a standard cable to match the desired transfer mode and then switch on the HMI device.
3. Set the HMI device to transfer mode.
   To start transfer mode, press the "Transfer" button in the Loader. You can also assign the system function "SetDeviceMode" to an operator control.
4. Load the project from the configuration computer to your target device. For further information, refer to "Loading projects".

Note

If the HMI device is a PC, you can transfer the compiled file without using the loader, for example, via Ethernet. Double-click the corresponding file on your PC to start Runtime.

10.11.2.3 Starting Runtime on the Engineering Station

Introduction

You can start your project in Runtime at the engineering station while performing the configuration in WinCC. However, this so-called online configuration is subject to certain limitations.

The project cannot be compiled in the background while Runtime is active on the Engineering Station. The delta data of the project is compiled automatically when you load the project to an HMI device after having closed Runtime. You can also start compilation manually.

When the project is started in Runtime, the settings you have stored in your project for the HMI device in the "Configuration" editor take effect.

Requirement

A project is open on the Engineering Station.

Procedure

Proceed as follows to start Runtime on the Engineering Station:

1. Select the desired HMI device in the project tree.
2. Select the menu command "Online > Start Runtime".
3. If you want to edit project data after Runtime was started on the Engineering Station, select the "Compile > Software" command in the shortcut menu of the HMI device.

The updated project is displayed in the Runtime on the Engineering Station.

10.11.2.4 Starting Runtime on the HMI device

Introduction

On completion of the project download to the HMI device, you can start the project in Runtime. The project is saved in the HMI device to a file with the following extension:

- Basic Panels as well as OP 73, OP 77A and TP 177A: "*.srt"

The project settings defined in the "Runtime settings" of the HMI device are activated when the project is started in Runtime.

The programs that you can use to start projects on the HMI device are available in the Runtime installation folder.

Note

Ending Runtime automatically

If automatic transfer is enabled on the HMI device and a transfer is started on the configuration PC, the running project is automatically stopped.

The HMI device then switches autonomously to "Transfer" mode.

After the commissioning phase, disable the automatic transfer function to prevent the HMI device from switching inadvertently to transfer mode.

Transfer mode can trigger unwanted responses in the plant.

To block access to the transfer settings and thus avoid unauthorized changes, assign a password in the Control Panel.

Requirement

WinCC Runtime is installed on the HMI device.

Procedure

The "RT Loader" application is started on a panel. The project loaded is started automatically after expiration of the configured delay.

If the project does not start automatically:

1. To select the project file, click "Settings" and enter the path to the project file under "Configuration file".
2. Click "OK" and then "Start".

10.11.2.5 Testing a project

Introduction

You have the following options for testing a WinCC project:

Testing projects on the configuration computer.

- Simulator
  The Simulator is used to test WinCC projects with internal tags and process tags. For additional information, refer to "Simulating a project".
  The simulator enables you to test the following:
  - Offline testing of a configuration without connection to a PLC.
  - Online testing of a configuration with connection to a PLC and inactive process.
  - Implementation of a demo project.

Testing projects on the HMI device

- Offline testing of the project on the HMI device
  Offline testing means that communication between the HMI device and the PLC is down for the duration of the test. You can operate the HMI device, but you cannot exchange data with the PLC and vice versa. Set the "Offline" mode on the HMI device by assigning the system function "SetDeviceMode" to an operator control.
- Online testing of the project on the HMI device
  Online testing means that communication between the HMI device and the PLC is up for the duration of the test. You control the plant using the HMI device based on the configuration. Set the "Online" mode on the HMI device by assigning the system function "SetDeviceMode" to an operator control.

Procedure

Proceed as follows to simulate a project without a PLC connection to the configuration PC:

1. Create a project as it is going to be run later with an interconnected PLC.
2. Save and compile the project.
3. Start the Simulator directly from the active configuration software. Select the menu command "Online > Simulate Runtime > With tag simulator".
   When you simulate the project for the first time, the simulator is started with a new, empty simulation table. If you have already created a simulation table for your project, it is opened. The simulation table "*.six" contains all the settings required for the simulation. For additional information, refer to "Working with the tag simulator".
4. Make any changes to the tags and area pointers of your project in the simulation table. Toggle between the simulation table and Runtime using the <ALT+TAB> key shortcut. Save the settings for the simulation using the menu command "File > Save." Enter a name to save the file.. The file name is automatically assigned the extension "*.six".

10.11.2.6 Closing a project

Introduction

You define the steps in closing Runtime in the user program:

Procedure

Exit Runtime as follows:

1. When Runtime is running, you can close it using the close symbol or the Task Manager.
2. When Runtime is running, press the relevant button to close Runtime. The close of Runtime is especially configured.

10.11.2.7 Backing up and restoring data of the HMI device

Introduction

Backup the data of an HMI device at regular intervals.

Working from the engineering station to which an HMI device is connected, you can backup and restore the data of this HMI device using WinCC.

You have the option of conveniently performing a central data backup using ProSave on a computer without WinCC installation.

Requirement

- The HMI device is interconnected with the Engineering Station or the computer is connected with ProSave.
- The HMI device whose data should be backed up or restored is selected in the project tree.
- The settings for loading are correctly set in the properties of the HMI device.
- When using a special storage medium, such as a data server: The HMI device is connected to the storage medium.

Procedure

Proceed as follows to back up the data:

1. Select the "Backup" command from the "Online > Device maintenance" menu.
2. Select the scope of the backup: "Complete backup," "Recipes," or "User administration."
3. Click "...", select the storage location in the "Select backup file" dialog, and specify a file name.
4. Click "OK."

This starts the data backup. The backup can take some time.

Procedure for restoring data

1. Select the "Restore" command from the "Online > Device maintenance" menu.
2. Click "...", and select the storage location and file in the "Open" dialog.
   The "Content" area indicates which HMI device is the origin of the data backup and the scope of the data backup.
3. Click "OK."

This starts the restoration. This process can take some time.

10.11.3 Languages in runtime

10.11.3.1 Languages in runtime

Using multiple runtime languages

You can decide which project languages are to be used in runtime on a particular HMI device. The number of runtime languages that can be available at one time on the HMI device depends on the device. To enable the operator to switch between languages during runtime, you must configure a corresponding operator control.

When runtime starts, the project is displayed according to the most recent language setting. When runtime starts the first time, the language with the lowest number in the "Language order" is displayed.

Setting runtime languages during configuration

You can specify the following in the "Language and Font" editor:

- The project languages available as runtime languages for the HMI device.
- The order in which the languages are switched.

10.11.3.2 Setting a runtime language

Introduction

The "Language & Font" editor shows all project languages available in the project. You can select the project languages to be available as runtime languages on the HMI device. In addition, you specify the order in which the languages will be switched.

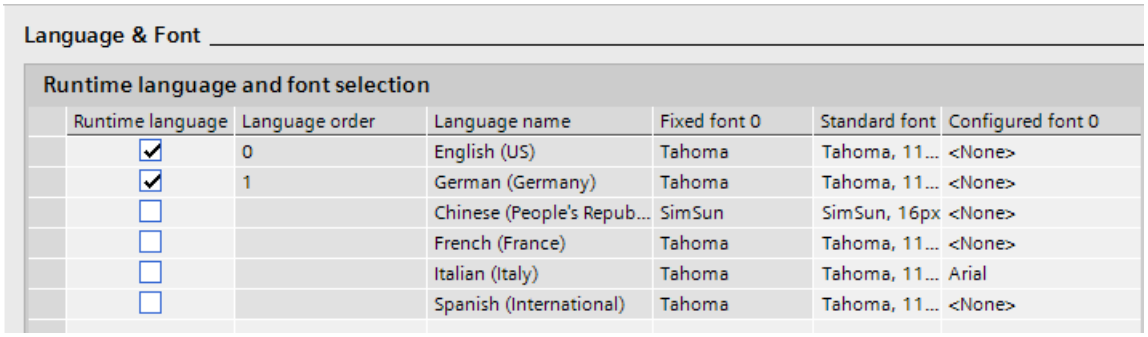
Requirement

Multiple languages are enabled in the "Project languages" editor.

Procedure

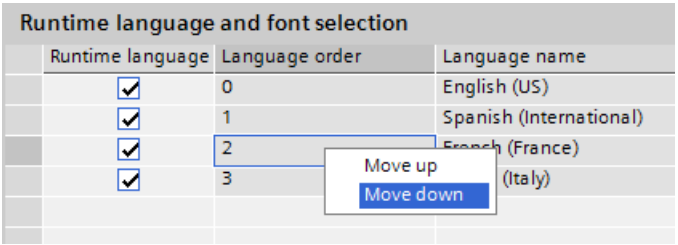
1. Open the "Language & Font" editor under "Device settings".
2. In the "Runtime language" column, select the the languages to be used when runtime first starts.
   The selected language is assigned the number "0" in the "Language order" column.

3. In the "Runtime language" column, select the language to be activated next when the language is switched.
   The selected language is assigned the number "1" in the "Language order" column.



Runtime language	Language order	Language name	Fixed font 0	Standard font	Configured font 0
☑	0	English (US)	Tahoma	Tahoma, 11...	<None>
☑	1	German (Germany)	Tahoma	Tahoma, 11...	<None>
☐		Chinese (People's Repub...	SimSun	SimSun, 16px	<None>
☐		French (France)	Tahoma	Tahoma, 11...	<None>
☐		Italian (Italy)	Tahoma	Tahoma, 11...	Arial
☐		Spanish (International)	Tahoma	Tahoma, 11...	<None>

4. Select additional languages in the order in which they are to be activated when the language is switched.
   If the number of languages selected exceeds the number that can be loaded on the HMI device, the table background changes in color.
5. If you want to change the order of a language, select the desired row and then select the shortcut menu command "Move up" or "Move down".



Result

The enabled runtime languages are transferred with the compiled project to the HMI device.

When runtime starts the first time, the project is displayed in the language with the lowest number in the "Language order."

If language switching by means of the "SetLanguage" system function has been configured, the specified number sequence determines the order in which the languages are switched.

10.11.3.3 Setting the font for a runtime language

Introduction

You can specify the font used to display the texts for each runtime language on the HMI device in the "Language & Font" editor. The default font is used in all texts if you cannot define a specific font.

WinCC offers only fonts supported by the HMI device.

The default font is also used for the representation of dialogs in the operating system of the HMI device. Select a smaller font as default if the full length of the dialog texts or headers is not displayed.

Requirement

Multiple languages are enabled in the "Project languages" editor.

Procedure

1. Open the "Language & Font" editor under "Device settings".
2. Enable the languages to be displayed on the HMI device in the "Runtime language" column. In the "Fixed font 0" column, WinCC shows the fonts used by default in runtime.
3. In the "Configured font 0" column, select another font for each language you want to have available during configuration.
   These fonts are transferred to the HMI device during a transfer operation.
4. In the "Standard font" column, select the font to be used by default if a font cannot be selected for a text.

Result

The project texts for the selected language are displayed in the selected font on the HMI device.

10.11.3.4 Configuring language switching

Introduction

You need to configure language switching if you want to have multiple runtime languages available on the HMI device. This is necessary to enable the operator to switch between the various runtime languages.

Methods for language switching

You can configure the following methods for language switching:

- Direct language selection
  Each language is set by means of a separate button. In this case, you create a button for each runtime language. Configure the "SetLanguage" system function for each font with the number of the language or the language ID as a parameter.
- Language switching
  The operator toggles through all languages using a button.
  Configure "SetLanguage" system function for the button with the "Toggle" parameter. The language is enabled in the order you have specified in the "Language & Font" editor.

Regardless of the method used, the button names must be translated into each of the languages used. You can also configure an output field that displays the current language setting.

10.11.3.5 Specific features of Asian and Eastern languages in runtime

Introduction

When configuring for Asian languages some specific features should be observed for operation in runtime.

Note

You can only use Asian fonts supported by your configuration computer during configuration.

Memory requirement for Asian character sets

The memory requirement is of course greater when using Asian languages. Therefore, pay attention to any error messages when compiling the project.

Inputting Eastern and Asian characters (not ANSI)

You cannot enter Eastern and Asian characters in runtime on the Basic Panels.

Interpretation of Asian characters

When using Sm@rtAcess and Sm@rtService, only the characters known on the HMI device can be used. To be able to use Asian characters, these must be configured in the engineering system. Additionally configured characters require additional memory on the HMI device. Pay attention to the amount of available memory on the HMI device.

Font size for Asian character sets

Use at least a font size of 10 points to display the text of projects created for Asian languages in runtime. Asian characters will become illegible if smaller font sizes are used. This also applies to the default font used in the "Language & Font" editor.

Text field length for Asian languages

Make allowances for an appropriate length of the text fields when working on multilingual projects with Asian languages. Field contents may be partially hidden, depending on the font and the font size.

1. Open the "Properties > Appearance" text box in the Inspector window.
2. Under "Fit to size", disable the "Auto-size" option.
3. Verify the proper display in runtime.

10.11.4 Operating projects

10.11.4.1 Basics

Overview of operator control over a project

All Basic HMI devices feature a touch screen. Certain Basic HMI devices feature function keys. Use the touch screen and function keys to operate the Control Panel or the project running on your HMI device.

DANGER

Incorrect operation

A project can contain certain operations that require in-depth knowledge about the specific plant on the part of the operator.

Ensure that only trained professional personnel operate the plant.

Operating the touch screen

NOTICE

Damage to the touch screen

Pointed or sharp objects can damage the plastic surface of the touch screen.

Always operate the touch screen with your fingers or with a touch pen only.

Triggering unintended actions

Touching several operator controls at the same time can trigger unintended actions.

Touch only one operator control on the screen at a time.

Operator controls are touch-sensitive symbols on the screen of the HMI device.

They are basically operated in the same way as mechanical keys. You activate operator controls by touching them with your finger.

Note

The HMI device returns a visual feedback as soon as it detects that an operator control has been touched.

The visual feedback is independent of any communication with the PLC. The visual feedback signal therefore does not indicate whether or not the relevant action is actually executed.

Examples of operator controls:

- Buttons
  Buttons can assume the following states:

- Invisible buttons
  The focus of invisible buttons is by default not indicated following selection. No optical operation feedback is provided in this case.
  The configuration engineer may, however, configure invisible buttons so that their outline appears as lines when touched. This outline remains visible until you select another operator control.
- I/O fields
  A screen keyboard appears as visual feedback after you touched an I/O field, for example, to enter a password.
  Depending on the HMI device and the configured operator control, the system displays different screen keyboards for entering numerical or alphanumerical values.
  The screen keyboard is automatically hidden again when input is complete.

Operating function keys

The function keys can be assigned global or local functions:

- Function keys with global function assignment
  A function key with global function assignment always triggers the same action on the HMI device or in the PLC, regardless of the currently displayed screen. The activation of a screen or the closing an alarm window, for example, is such an action.
- Function keys with local function assignment
  A function key with local function assignment is screen-specific and is therefore only effective within the active screen.
  The function assigned to a function key can vary from screen to screen.

The function key could be assigned only a single function within a screen only, that is, either a global or a local function. Local function assignments override global function assignments.

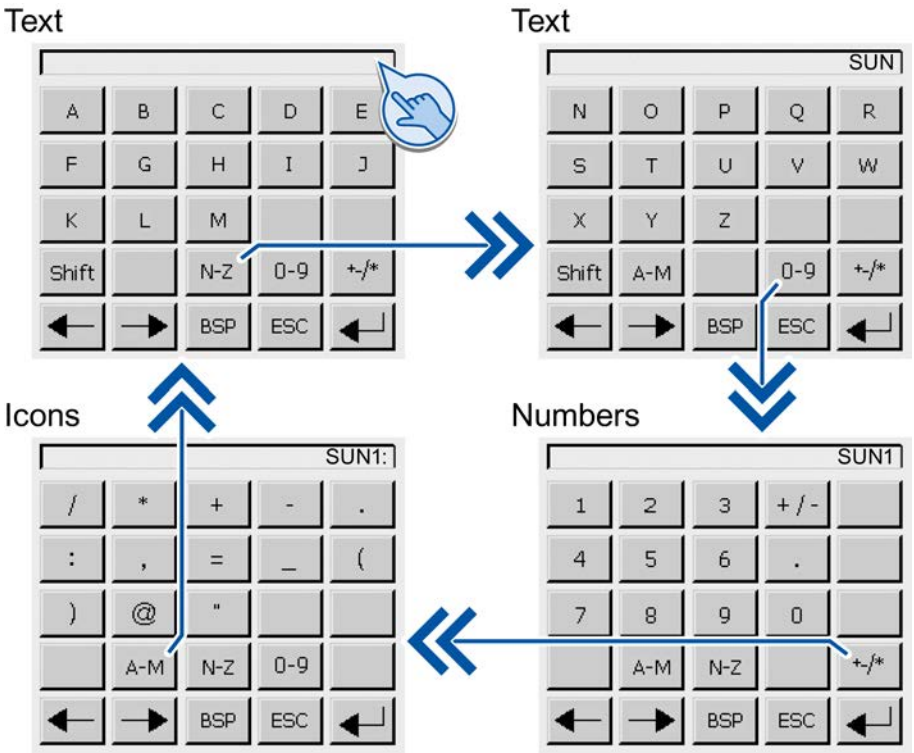
General functions of the screen keyboard

The following keys are available on the screen keyboard of all Basic HMI devices:

←	Cursor left
→	Cursor right
BSP	Deleting a character
ESC	Cancel input
↵	Confirm input
Help	Displaying infotext. This key only appears when an infotext has been configured for the operator control.

Entering data on the KTP400 Basic

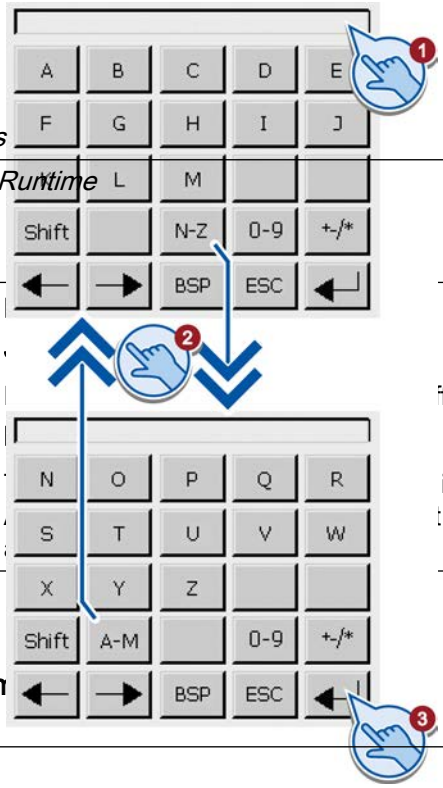
Due to the small display, the screen keyboard and the input concept of the KTP400 Basic differs compared to other Basic HMI devices.



The screen keyboard appears on the HMI device touch screen when you touch an operator control that requires input.

The screen keyboard of the KTP400 features four views. You can change the view while making entries using the buttons in the fourth row of the screen keyboard:

	Changes to the view
	Entering text, characters "A" to "M"
	Entering text, characters "N" to "Z"
	Entering numbers, "0" to "9," signed or unsigned and with or without decimal places
Shift	Entering special characters
	Entering text, shift to lower case letters

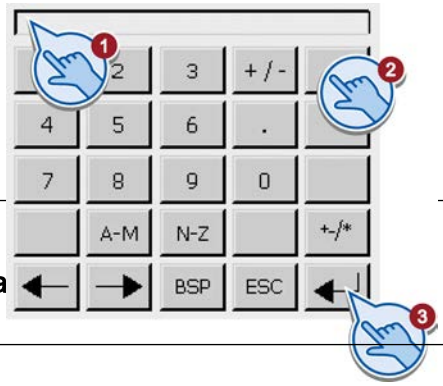


fect while the screen keyboard is open.

is monolingual.
does not have any effect on the layout of the

Entering alphanum

	1. Touch the desired operator control on the screen. The alphanumerical screen keyboard opens. 2. Enter the value. Depending on the settings, the HMI device outputs an audible signal. You can change the view of the screen keyboard using the keys <N-Z> and <A-M>. Use the <Shift> key to enter lower-case letters. 3. Press <Return> key to confirm your entries, or cancel them with <ESC>. Either action closes the screen keyboard.



Entering numerica

	1. Touch the desired operator control on the screen. The numerical screen keyboard opens. 2. Enter the value. Depending on the settings, the HMI device outputs an audible signal. You can change the view of the screen keyboard for entering numbers with hexadecimal notation using the <N-Z> and <A-M> keys. 3. Press <Return> key to confirm your entries, or cancel them with <ESC>. Either action closes the screen keyboard.

Checking numerical value limits

Tags can be assigned limit values. Any entry of a value outside this limit is rejected. If an alarm view is configured, a system event is triggered and the original value is displayed again.

Decimal places of numerical values

The configuration engineer can define the number of decimal places for a numerical text box. The number of decimal places is checked when you enter a value in this type of I/O field.

- Decimal places that exceed the limit are ignored.
- Unused decimal places are padded with "0" entries.

Entering data on the KTP600, KTP1000, TP1500 Basic

Alphanumerical screen keyboard

The screen keyboard appears on the HMI device touch screen when you touch an operator control that requires input.

Text

Numbers

Note

Job mailbox has no effect

PLC job 51 "Select screen" has no effect while the screen keyboard is open.

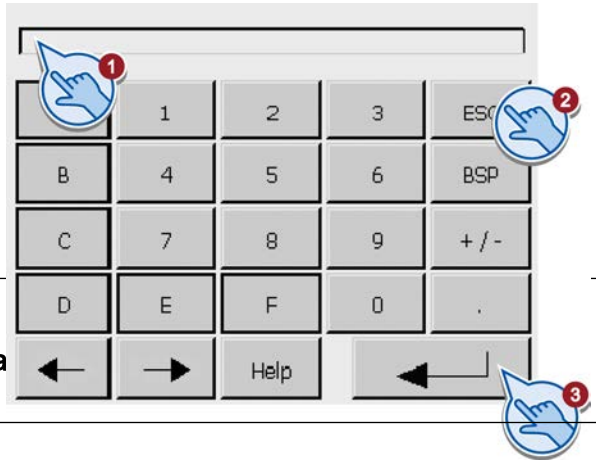
Key assignment

The alphanumerical keyboard layout is monolingual.
A language change within the project does not have any effect on the layout of the alphanumerical screen keyboard.

Entering alphanum

	1. Touch the desired operator control on the screen. The alphanumerical screen keyboard opens. 2. Enter the value. Depending on the settings, the HMI device outputs an audible signal. Use the <Shift> key to enter lower-case letters. 3. Press <Return> key to confirm your entries, or cancel them with <ESC>. Either action closes the screen keyboard.

Entering numerica



	1. Touch the desired operator control on the screen. The numerical screen keyboard opens. 2. Enter the value. Depending on the settings, the HMI device outputs an audible signal. 3. Press <Return> key to confirm your entries, or cancel them with <ESC>. Either action closes the screen keyboard.

Checking numerical value limits

Tags can be assigned limit values. Any entry of a value outside this limit is rejected. If an alarm view is configured, a system event is triggered and the original value is displayed again.

Decimal places of numerical values

The configuration engineer can define the number of decimal places for a numerical text box. The number of decimal places is checked when you enter a value in this type of I/O field.

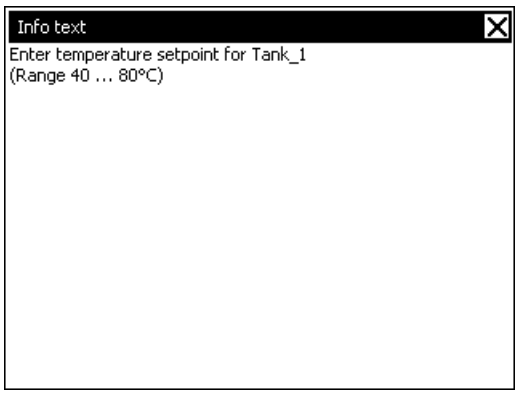
- Decimal places that exceed the limit are ignored.
- Unused decimal places are padded with "0" entries.

Display tooltip

Use

The configuration engineer uses tooltips to provide additional information and operating instructions. The configuration engineer can configure tooltips for screens and operating elements.

The tooltip of an I/O field may contain, for example, information on the value to be entered.



Procedure

Proceed as follows to open the tooltip for operator controls:

1. Touch the required operating element.
   The screen keyboard opens. The representation of the Help key indicates whether a tooltip was configured for the operating element or for the current screen.
2. Touch the Help key of the on-screen keyboard.
   The tooltip for the operating element is displayed. If there is no tooltip for the selected screen object, the tooltip for the current screen is displayed if it has been configured.
   You can scroll through the contents of long tooltips using the ▼ and ▲ buttons.

   Note

   Switching between displayed tooltips

   The configuration engineer can configure tooltips for an I/O field and the associated screen. You can switch between two tooltips by touching the tooltip window.

3. Close the displayed tooltip by pressing ☒.

Alternative procedure

Depending on your configuration, tooltips can also be called via a configured operating element.

Setting the project language

Introduction

The HMI device supports multilingual projects. You must have configured a corresponding operating element which lets you change the language setting on the HMI device during runtime.

The project always starts with the language set in the previous session.

Requirement

- The required language for the project must be available on the HMI device.
- The language switching function must be logically linked to a configured operating element such as a button.

Selecting a language

You can change project languages at any time. Language-specific objects are immediately output to the screen in the new language when you switch languages.

The following options are available for switching the language:

- A configured operating element switches from one language to the next in a list.
- A configured operating element directly sets the desired language.

10.11.4.2 Operating objects

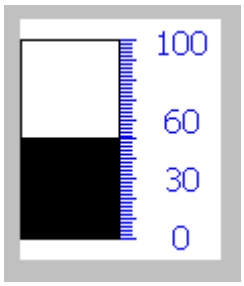
Bar

Application

The bar is a dynamic display object. The bar displays a value from the PLC as a rectangular area. The bar allows you to recognize the following at a glance:

- The distance of the current value from the configured limit values
- Whether a set point value has been reached

The bar can display values such as fill levels or batch counts.



Layout

The layout of the bar depends on the configuration:

- The bar may feature a scale of values
- The configured limit values can be indicated by lines
- Color changes can signal when a limit value has been exceeded or has not been reached

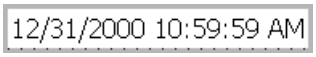
Date/time field

Overview

Application

A "Date / time box" may have the following runtime functions:

- Output of the date and time
- Combined input and output; here the operator can edit the output values so as to reset the date and time.



Layout

The appearance of the date/time field depends on the language set in the HMI device.

The date can be displayed in detail (e.g. Tuesday, 31 December 2003) or in short form (31.12.2003).

Operation

Depending on the configuration, you can operate the date/time field in the following ways:

- Standard operation: Change date and time.

Runtime behavior

When the operator ignores the syntax when entering values, or enters illegal values, the system rejects these. Instead, the original values (plus the time that has elapsed in the meantime) appears in the date/time field and a system alarm is displayed on the HMI device.

Touch and key operation

Touch operation

Proceed as follows to operate the date/time field:

1. Touch the date/time field on the touch screen of the HMI device. The on-screen keyboard is displayed automatically.
2. Enter the required value using the on-screen keyboard.
3. Press <ENTER> to confirm your entry, or cancel it with <ESC>. The on-screen keyboard is cleared automatically after you have confirmed or canceled your entries.

Key operation

Activate the date/time field, for example, with one or several TAB according to the configured tab sequence. A color frame signals the selected state of the field content.

Procedure

Proceed as follows to operate the date/time field:

1. Position the cursor using the cursor keys and enter the value.
2. Press ENTER. The object changes to the special editing mode. Only one character at any time is marked in the field.
   – Use the cursor keys ▲ / ▼ to scroll a character table.
   – Use the cursor keys ◀ / ▶ to move the cursor to the next or previous input position.
3. Confirm your entry with ENTER, or discard it with ESC.

I/O field

Overview

Application

You enter numeric or alphanumeric values in an I/O field. For example, a numeric value could be the number 80 as a temperature reference; an alphanumeric value could be the text "Service" as a user name.

Layout

The appearance of the I/O field depends on the configuration:

- Numeric I/O field
  For input of numbers in decimal, hexadecimal or binary format.
- Alphanumeric I/O field
  For input of character strings.
- I/O field for date and time
  For input of calendar dates or time information. The format depends on the set configuration.
- I/O field for password entry
  For concealed entry of a password. The entered character string is displayed with placeholders (*).

Operation

Depending on the configuration, you can operate the I/O field in the following ways:

- Standard operation: Enter a value in the I/O field.
- Event: An event is triggered when you operate the I/O field, for example, when you activate it. The processing of a function list can be configured to the event.

Runtime behavior

Limit value test of numerical values

Tags can be assigned limit values. If you enter a value that lies outside of this limit, it will not be accepted; for example, 80 with a limit value of 78. In this case the HMI device will deliver a system alarm, if an alarm window is configured. The original value is displayed again.

Decimal places for numerical values

The configuration engineer can define the number of decimal places for a numerical text box. The number of decimal places is checked when you enter a value in this type of I/O field.

- Decimal places in excess of the limit are ignored.
- Empty decimal places are filled with "0".

Hidden input

A "*" is displayed for every character during hidden input. The data format of the value entered cannot be recognized.

Behavior when switching between input fields

When you change to another input field within the same screen, and the screen keyboard appears, the "Exit field" event is not executed for the previous field unless you close the screen keyboard.

Touch and key operation

Touch operation

Proceed as follows to operate the IO field:

1. Touch the IO field on the touch screen of the HMI device. The on-screen keyboard is displayed automatically.
2. Enter the required value using the on-screen keyboard.
3. Press <ENTER> to confirm your entry, or cancel it with <ESC>.

The on-screen keyboard is cleared automatically after you have confirmed or canceled your entries.

Key operation

Activate the IO field,for example, with one or several TAB according to the tab sequence configured. A color frame signals the selection of the field content.

Procedure

Proceed as follows to operate the IO field:

1. Position the cursor using SHIFT and a cursor key.
2. This step cancels the field content selection. Enter the desired value.
3. Press ENTER. The object will change to special editing mode. Only one character at any time is marked in the field.
   - Use the cursor keys ▲ / ▼ to scroll a character table.
   - Use the cursor keys ▶ / ◀ to move the cursor to the next or previous input position.
4. Confirm your entry with ENTER, or discard it with ESC.

Note

Enable the input of hexadecimal characters "F" for numerical values by toggling the input keys to character mode using the key A-Z.

Graphic view

Application

The graphic view displays graphics.

Layout

The appearance of the graphic depends on the configuration. The graphic view, for example, adapts automatically to the size of the graphic.

Note

If you use bitmaps with the "Transparent color" setting in WinCC screens, requires high-performance display on the Panel HMI devices. Performance is enhanced by disabling the "Transparent color" setting in the properties of the respective graphic object. This restriction applies in particular when bitmaps are used as background image.

Operation

The graphic display is for display only and cannot be operated.

Graphic I/O field

Overview

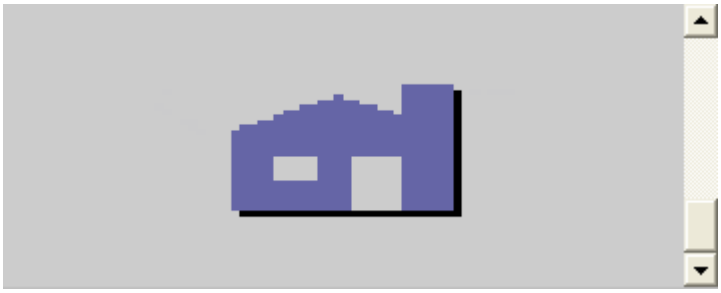
Application

A graphic IO field can have the following Runtime functions:

- Output of graphic list entries
- Combined input and output

Example of its use as output field:

To indicate the Runtime status of a valve, the graphic IO field outputs the image of a closed or open valve.



Operation

Depending on the configuration, you can operate the graphic IO field in the following ways:

- Standard operation: Select an entry from the graphic list.
- Event: An event is triggered when you operate the graphic IO field, e.g. when you activate it. The processing of a function list can be configured to the event.

Runtime behavior

If the graphic IO field displays a cactus image, you have not defined a graphic to be output for a specific value in your project.

The contents of the graphic IO field change color to show that it is now activated.

The frame in 3D is only shown graphically in an output field.

Touch and key operation

Touch operation

Touch the graphic IO field on the touch screen of the HMI device. Selection mode is now enabled.

Select the graphic object using the scroll bar.

Touch the selected graphic object to save it or discard the selection by touching a different screen object.

Key operation

How to use a graphic IO field on the keyboard device:

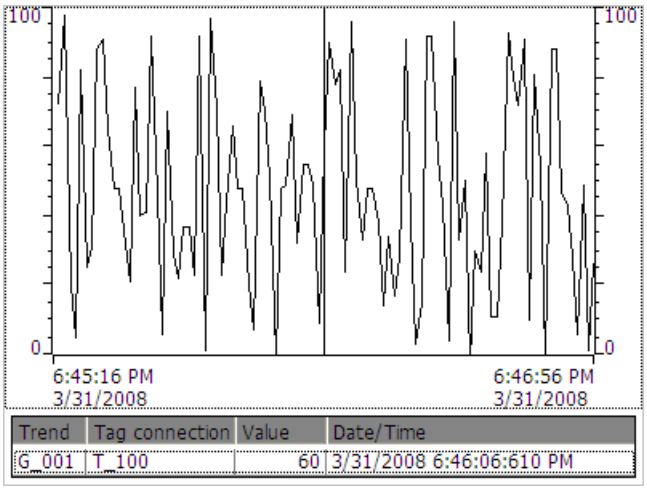
Step		Procedure	
1	Select the graphic IO field	e.g. ▶	The graphic IO field is marked.
2	Enable the selection mode	ENTER	The selection mode is now enabled.
3	Select an entry	▲ ▼ ◀ ▶	Moves the cursor by lines.
4	Accept selection or	ENTER	The entry selected is now valid. The selection mode is closed.
	Cancel selection	ESC	The original value is restored.

Trend view

Overview

Application

The trend view is a dynamic display object. The Trend view can display actual process data and process data from a log continuously when it is supported by the HMI device.



Layout

The layout of the trend view is based on the configuration. A trend view can show multiple curves simultaneously to allow the user, for example, to compare different process sequences. If the displayed process value exceeds or falls below the configured limit values, the violation of the limit can be displayed by a change of color in the curve.

A ruler can also simplify the reading of the process values from the trend view. The ruler displays the Y-value that belongs to an X-value.

Operation

Depending on the configuration you can:

- The displayed time section extends.
- The displayed time section reduces.
- Scroll back by one display width.
- Scroll forwards by one display width.
- Stop or continue the trend recording.

Operator controls

The buttons have the following functions:

Operator control	Function
	Scrolls back to the start of trend recording. The start values of the trend recording are displayed there.
	Zooms the displayed time section
	Zooms out of the displayed time section
	Moves the ruler backwards (to the left).
	Moves the ruler forward (to the right).
	Scrolls back by one display width (to the left).
	Scrolls forward by one display width (to the right).
	Shows or hides the ruler. The ruler displays the X-value associated with a Y-value.
	Stops or continues trend recording

Touch and key operation

Procedure

Touch the relevant operator controls of the trend view on the touch screen of the HMI device.

Procedure

Select the trend view with the TAB key using the configured tab sequence.

The table below lists the key shortcuts available:

Keys	Function
CTRL ENTER	Scrolls back to the start of trend recording. The start values of the trend recording are displayed there.
CTRL + +	Enlarges the time section displayed.
CTRL + −	Reduces the time section displayed.
CTRL + ALT + ◀	Moves the ruler backwards (to the left).
CTRL + ALT + ▶	Moves the ruler forward (to the right).
SHIFT + ◀	Scrolls back by one display width (to the left).
SHIFT + ▶	Scrolls forward by one display width (to the right).

Button

Overview

Application

A button is a virtual key on the screen of the HMI device that can have one or more functions.

Layout

The layout of the button depends on the button type.

- Button with text: The text shown on the button gives information regarding the status of the button.
- Button with graphic: The graphic shown on the button gives information regarding the status of the button.
- Invisible: The button is not visible during Runtime.

Operation

Depending on the configuration, you can operate the button in the following ways:

- Standard operation: Click the button.
- Event: An event is triggered when you operate the button, e.g. when you click it. The processing of a function list can be configured to the event.

Runtime behavior

The operation may be followed with a optical feedback. However, note that the optical feedback only indicates a completed operation and not whether the configured functions were actually executed.

Touch and key operation

Procedure

Touch the button on the touch screen of the HMI device.

Procedure

How to operate a button on the keyboard device:

1. Select the button using a cursor key, e.g. ▶
2. Next, press the ENTER or the ␣ key.

Switch

Overview

Application

The switch is an operating element and display object with two states: "pressed" and "released." Switches can signal the state of a system component that cannot be seen from the HMI device, e.g. a motor. You can also change the state of that system component at the HMI device.

Layout

The layout of the switch depends on the switch type.

- Switches: The switch has a slider. The position of this slider gives information about the status of the switch.
- Switch with text: The text shown on the switch gives information regarding the status of the switch.
- Switch with graphic: The graphic shown on the switch gives information regarding the status of the switch.

Operation

Depending on the configuration, you can operate the switch in the following ways:

- Standard operation: Click the switch.
- Event: An event is triggered when you operate the switch, e.g. when you click it. The processing of a function list can be configured to the event.

Runtime behavior

A switch has two stable states: When you activate the switch, it changes to the other state. The switch retains this state until the next operation.

Touch and key operation

Touch operation

Touch operation of the switch differs, depending on its type:

- If a slider is displayed for the switch:
  Drag the slider to the new position on the touch screen of the HMI device or double-click in the slider range.
- If only a text or graphic object is displayed for the switch:
  Touch the switch on the touch screen of the HMI device.

Key operation

How to operate a switch on the keyboard device:

- Select the switch using a cursor key, e.g. ▶
- Next, press the ENTER or the ␣ key.

Symbolic I/O field

Overview

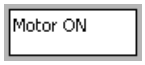
Application

A symbolic IO field can be assigned the following functions in Runtime:

- Output of text list entries
- Combined input and output

Example of its use as combination IO field:

You control a motor in Runtime by selecting the "Motor OFF" or "Motor ON" text from the text list. The motor is started or stopped in accordance with the selection. The symbolic IO field visualizes the current status of the motor.



Operation

The following options of operating the symbolic IO field are available, depending on the configuration:

- Standard operation: Select an entry from the text list.
- Event: You trigger an event by operating the symbolic IO field, e.g. by enabling it. The processing of a function list can be configured to the event.

Runtime behavior

The selection list of the symbolic IO field displays an empty text line if a corresponding entry was not defined in project data. The active state is indicated on the HMI device by changing the color of contents of the symbolic IO field.

Touch and key operation

Touch operation

Touch the symbolic IO field on the touch screen of the HMI device. The selection list displays the default entries.

If the selection list displays a scroll bar: Touch the scroll bar on the touch screen of the HMI device. Touch the scroll bar and drag to the required position on the touch screen.

Select the entry and accept the corresponding tag value by touching the entry on the screen. The selection list closes and the entry is displayed. The focus is still set on the symbolic IO field.

Key operation

How to operate a symbolic IO field on the keyboard device:

Step		Procedure	
1	Select the symbolic IO field	e.g. ▶	The symbolic IO field is marked.
2	Open selection list	ENTER	The drop down list box opens.
3	Select an entry	▲ ◀ ▼ ▶	Moves the cursor by lines.
4	Accept the selection Or	ENTER	The entry selected is now valid. The selection list is closed.
	Canceling the selection	ESC	The original value is restored. The selection list is closed.

10.11.4.3 Project security

Overview

Design of the security system

The configuration engineer can protect the operation of a project by implementing a security system.

The security system is based on authorizations, user groups and users.

If you use an operator control with access protection, the HMI device first requests that you log on. A logon screen is displayed in which you enter your user name and password. After logging on, you can operate the operator controls for which you have the necessary authorizations.

The logon dialog can be set up by the configuration engineer via an individual operator control.

In the same way, the configuration engineer can set up an operator control to log off. After logging off, objects assigned access protection can no longer be operated; to do so, log on again.

User groups and authorizations

Project-specific user groups are created by the configuration engineer. The "Administrators" and "Users" groups are included in all projects by default. User groups are assigned authorizations. Authorization required for an operation is specifically defined for each individual object and function in the project.

Users and passwords

Each user is assigned to exactly one user group.

The following people are allowed to create users and assign them passwords:

- The configuration engineer during configuration
- The administrator on the HMI device
- A user with user administration authorization on the HMI device

Irrespective of the user group, each user is allowed to change his own password.

Logoff times

A logoff time is specified in the system for each user. If the time between any two user actions, such as entering a value or changing screens, exceeds this logoff time, the user is automatically logged off. The user must then log on again to continue to operate objects assigned access protection.

Backup and restore

The user data is encrypted and saved on the HMI device to protect it from loss due to power failure.

The users, passwords, group assignments and logoff times set up on the HMI device can be backed up and restored. This prevents you having to enter all of the data again on another HMI device.

Note

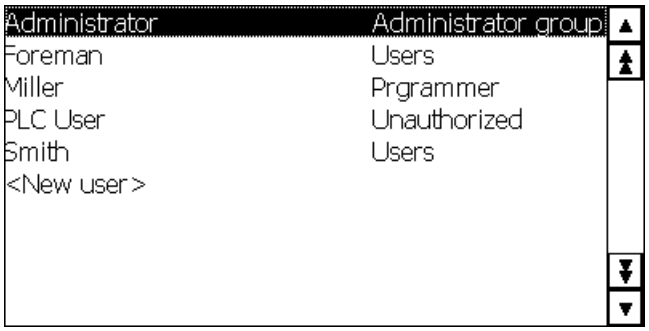
The currently valid user data is overwritten in the following cases:

- Depending on settings for a new download of the project.
- Upon restore of a backed-up project
- Upon import of the user administration via an operator control. The newly downloaded or restored user data and passwords take effect immediately.

Simple user view

Use

On HMI devices with a small display, the simple user view is used to display users on the HMI device.



Note

The "Simple user view" object cannot be operated dynamically with a script.

Layout

The appearance depends on the authorizations.

- All users on the HMI device are displayed in the User view to the administrator or to a user with administrator authorizations.
- When user administration authorization is lacking, only the personal user entry is displayed.

Operation

Depending on the configuration you can:

- Manage users, e.g. create, delete.
- Change existing user data.
- Export or import user data.

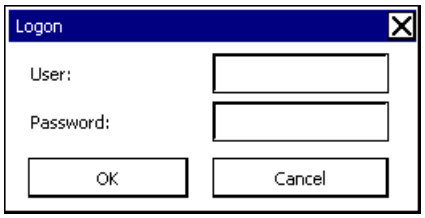
Note

On an HMI device the number is limited to 100 users and one PLC user. This restriction does not apply to PCs. On a PC, the maximum number of users is restricted by the physical memory.

User logon

Logon dialog

Use the logon dialog to log on to the security system of the HMI device. Enter your user name and password in the logon dialog.



The logon dialog opens in the following cases:

- You use an operator control with access protection.
- You press an operator control that was configured for displaying the logon dialog.
- You enable the "<ENTER>" entry in the simple user view.
- You enable a blank entry in the extended user view.
- The logon dialog will be automatically displayed when the project is started, depending on the configuration.

Requirement

The logon dialog is open.

Procedure for touch operation

Proceed as follows:

1. Enter the user name and password.
   Touch the corresponding input field. The alphanumerical screen keyboard is displayed.
2. Select "OK" to confirm logon.

Procedure for key operation

Proceed as follows:

1. Select the "User" input field within the logon dialog by pressing the TAB key.
2. Enter the user name using the system keys.
   Switch the numerical keypad to alphabet mode using the A-Z key to enter letters.
3. Use the TAB key to select the "Password" input field.
4. Enter the password using the system keys.
5. Confirm your entries with "OK".

Note

The user name is not case sensitive.

The password is case sensitive.

Result

After successful logon to the security system, you can execute functions on the HMI device which are access protected and for which you have authorization.

An alarm is output if an incorrect password has been entered and if an alarm view was configured.

User logoff

Requirement

You have logged into the security system of the HMI device.

Procedure

You have the following options for logging off:

- You press an operator control that was configured for logoff.
- You will be logged off automatically if you are not operating the project and if the logoff time has been exceeded.

You will also be automatically logged off if you enter an incorrect password.

Result

You are no longer logged into the project. In order to use an operator control with access protection, you first have to log on again.

Creating users

Requirement

- The user view is open.
- You are either authorized for user administration or you are an administrator.
- A user group has been created.

Note

Runtime users must be assigned to a user group. The user group is created in the Engineering System. The designation of the user group is language-dependent.

Note

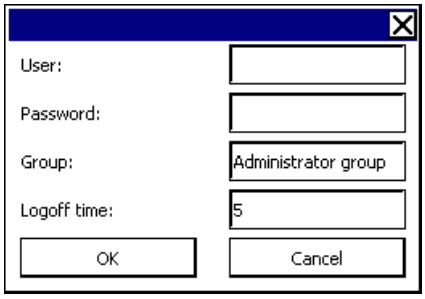
The following characters cannot be used in passwords:

- Blanks
- Special characters * ? . % / \ ' "

Creating users in the simple user view

Proceed as follows:

1. Click the "<New user>" entry in the user view. A dialog opens.



2. Enter the desired user name and password.
   Touch the corresponding text box. The alphanumerical on-screen keyboard is displayed.
3. Click on the text box of the group. A dialog opens.
4. Assign the user to a group. Select ▲ and ▼ to scroll the selection list.
5. Touch the required entry in the drop down list box.
   The selected entry is accepted as input.
6. Touch the text box "Logoff time". The on-screen keyboard is displayed.
7. Enter a logoff time between 0 and 60 minutes. The value 0 stands for "no automatic logoff."
8. Confirm your entries with "OK."

Result

The new user is created.

Changing the user

Requirement

The user view is open.

Your authorization level determines the data you can change:

- You are an administrator or a user with user administration authorization. In these cases you are allowed to change the data for all the users on the HMI device in the user view:
  - User name
  - Group assignment
  - Password
  - Logoff time
- You are a user without user management authorization. In this case you are only allowed to change your personal user data:
  - Password
  - Logoff time, if configured

Note

You can only change the logoff time and password for the "Admin" user.

You can only change the logoff time for the "PLC_User". This user is used for logging on via the PLC.

Note

Changes in the user view are effective immediately in Runtime. Changes in Runtime are not updated in the Engineering System.

When the user management is downloaded to the HMI device, all changes in the user view are overwritten.

Changing user data in the simple user view

Proceed as follows:

1. In the user view, touch the user whose user data you want to change.
2. When entering the data, use exactly the same procedure as for creating a user.

Changing user data in the advanced user view

Proceed as follows:

1. In the user view, touch the user whose user data you want to change.
2. When entering the data, use exactly the same procedure as for creating a user.

Result

User data have been changed.

Deleting a user

Requirement

- You have opened a screen that contains the user view.
- You must be logged on with administrator rights or be authorized for user management to delete users.

Note

Changes in the user view are effective immediately in Runtime. Changes in Runtime are not updated in the Engineering System.

When the user management is downloaded to the HMI device, all changes in the user view are overwritten.

Procedure for touch operation

Proceed as follows:

1. Touch the user to be deleted in the user view.
2. Delete the user name.

Procedure for key operation

Proceed as follows:

1. Select the user view using the TAB key or the cursor keys.
2. Select the user in the user view by means of cursor keys.
3. Press INS DEL to delete the user.

Result

The user has been deleted and may no longer log onto the project.

10.11.4.4 Operating alarms

Overview

Alarms

Alarms indicate events and states on the HMI device which have occurred in the plant, in the process or on the HMI device itself. A status is reported when it is received.

An alarm could trigger one of the following alarm events:

- Incoming
- Outgoing
- Acknowledge

The configuration engineer defines which alarms must be acknowledged by the user.

An alarm may contain the following information:

- Date
- Time
- Alarm text
- Event text
- Location of fault
- Status
- Alarm class
- Alarm number
- Alarm group
- Supports diagnostics

Alarm classes

Alarms are assigned to various alarm classes. The selection depends on the HMI device.

- "Warnings"
  Alarms of this class usually indicate states of a plant such as "Motor switched on." Alarms in this class do not require acknowledgment.
- "Errors"
  Alarms in this class must always be acknowledged. Alarms normally indicate critical errors within the plant such as "Motor temperature too high".
- "System"
  System alarms indicate states or events which occur on the HMI device.
  System alarms provide information on occurrences such as operator errors or communication faults.
- "Diagnosis Events"
  SIMATIC diagnostic alarms show states and events in the SIMATIC S7 controllers.

Note
Availability for specific devices

Diagnostic alarms are not available for Basic Panels.

- STEP 7 alarm classes
  The alarm classes configured in STEP 7 are also available to the HMI device.

  Note
  Availability for specific devices

  STEP 7 alarm classes are not available for Basic Panels.

- Custom alarm classes
  The properties of this alarm class must be defined in the configuration.

Alarm buffer

Alarm events are saved to an internal buffer. The size of this alarm buffer depends on the HMI device type.

Alarm report

When alarm report is enabled in the project, alarm events are output directly to the connected printer.

You can set the reporting function separately for each alarm. The system outputs "Incoming" and "Outgoing" alarm events to the printer.

The output of alarms of the "System" alarm class to a printer must be initiated by means of the corresponding alarm buffer. This outputs the complete content of the alarm buffer to the printer. To be able to initiate this print function, you need to configure a corresponding control object in the project.

Note
Availability for specific devices

Alarm reports are not available for Basic Panels.

Alarm log

Alarm events are stored in an alarm log, provided this log file is configured. The capacity of the log file is limited by the storage medium and system limits.

Note
Availability for specific devices

Alarm logs are not available for Basic Panels.

Alarm view

The alarm view shows selected alarms or alarm events from the alarm buffer or alarm log. Whether alarms have to be acknowledged or not is specified in your configuration. By means of configuration, the display can be filtered in such a way that only alarms that contain a specific character string will be shown.

Alarm window

If configured, an alarm window shows all pending alarms or alarms awaiting acknowledgment of a particular alarm class. The alarm window is displayed as soon as a new alarm occurs.

You can configure the order in which the alarms are displayed. Either the current alarm or the oldest alarm is displayed. The alarm window can also be set to indicate the exact location of the fault, including the date and time of the alarm event. By means of configuration, the display can be filtered in such a way that only alarms that contain a specific character string will be shown.

Alarm indicator

The alarm indicator is a graphic icon that is displayed on the screen when an alarm of the specified alarm class is incoming.

The alarm indicator can assume one of two states:

- Flashing: At least one unacknowledged alarm is pending.
- Static: The alarms are acknowledged but at least one of them is not yet deactivated. The number displayed indicates the number of pending alarms.

Simple alarm view, alarm window

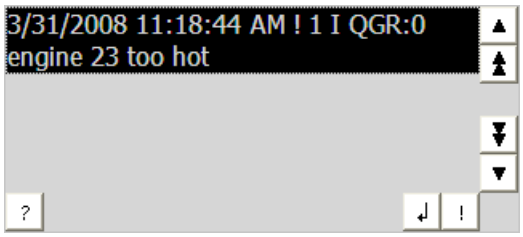
Use

The simple alarm view shows selected alarms or alarm events from the alarm buffer or alarm log. The layout and operation of the simple alarm window correspond to that of the simple alarm view.

Note

The "Single alarm view" object cannot be operated dynamically with a script.

In the Engineering System you can dynamically control the visibility of an object, for example, in the "Animations" tab of the Properties window. In Runtime, the "Simple alarm view" does not support animations. If you have configured an animation and wish to perform a consistency check of the project, for example, then an error alarm is issued in the Output window.



Layout

Depending on the configuration, in the alarm view different columns with information regarding an alarm or an alarm event are displayed.

To differentiate between the different alarm classes, the first column in the alarm view contains an icon:

Icon	Alarm class
!	"Errors"
(empty)	"Warnings"
(depends on the configuration)	Custom alarm classes
$	"System"

Operation

Depending on the configuration you can:

- Acknowledge alarms
- Edit alarms

Control elements

The buttons have the following functions:

Button	Function
!	Acknowledge alarm
↲	Edit alarm
?	Display infotext for an alarm
▶	Shows the full text of the selected alarm in a separate window, the alarm text window. In the alarm text window, you can view alarm text that requires more space than is available in the alarm view. Close the alarm text window with ✕.
▲	Scrolls one alarm up
⏫	Scrolls one page up in the alarm view

Button	Function
	Scrolls one page down in the alarm view
	Scrolls one alarm down

Layout of the operator controls

The layout of the buttons for operating the simple alarm view depends on the configured size. Therefore check if all the required buttons are available on the HMI device.

Detecting pending alarms

Introduction

You can recognize the presence of alarms which must be acknowledged by the following:

- For an HMI device with keys: The LED of the key ACK is lit.
- Depending on the configuration: An alarm indicator is displayed on screen.

The configuration determines whether an alarm has to be acknowledged or not. This is also defined by the alarm class which an alarm belongs to.

LED in the "ACK" key

An LED is integrated in the ACK key on HMI devices with keyboard. The LED is lit if there are alarms requiring acknowledgment which must still be acknowledged.

The LED goes out when you acknowledge all alarms requiring acknowledgment.

Alarm indicator

The alarm indicator is a graphic symbol indicating pending alarms or alarms requiring acknowledgment, depending on the configuration.

Figure 10-4 Alarm indicator with three pending alarms

Layout

The alarm indicator can assume one of two states:

- Flashing:
  The alarm indicator flashes as long as alarms are pending for acknowledgment. The number displayed indicates the number of pending alarms. The project engineer can configure specific functions to be executed by operating the alarm indicator.
- Static: The alarms are acknowledged but at least one of them is not yet deactivated.

Displaying dialogs

The displayed alarm indicator view is covered, for example, by the logon dialog, help dialog or alarm text windows. The alarm indicator is visible once you close these dialogs.

Display infotext for alarm

Procedure for touch operation

Proceed as follows to display the info text:

1. Touch the relevant alarm in the alarm view or in the alarm window.
   The alarm is selected.
2. Touch the ? button in the simple alarm view or in the advanced alarm view.
   If configured, the info text assigned to this alarm is displayed.
3. Close the screen for displaying the Info text by means of the ☒ button.

Procedure for key operation

Proceed as follows to display the info text:

1. Select the desired alarm in the alarm view.
2. Press the key HELP.
   If configured, the info text assigned to this alarm is displayed.
3. Close the info text by pressing the HELP key.

Acknowledge alarm

Requirement

The alarm to be acknowledged is displayed in the alarm window or the alarm view.

Procedure for touch operation

To acknowledge an alarm, proceed as follows:

1. Touch the relevant alarm in the alarm view or in the alarm window.
   The alarm is selected.
2. Touch the ! button in the simple alarm view or in the advanced alarm view.

Procedure for key operation

The alarm view and the alarm window have a tab sequence with which you can select operator controls and the last selected alarm using the keyboard.

To acknowledge an alarm, proceed as follows:

1. Select the desired alarm view or alarm window using the TAB key.
2. Select the desired alarm. Use the HOME, END, ▲ or ▼ keys accordingly.
3. Press the key ACK.

Alternative operation

Depending on the configuration, you can also acknowledge an alarm with a function key.

Result

The alarm is acknowledged. If the alarm belongs to an alarm group, all the alarms of the associated group are acknowledged.

Editing an alarm

Introduction

The configuration engineer can assign additional functions to each alarm. These functions are executed when the alarm is processed.

Note

When you edit an unacknowledged alarm, it is acknowledged automatically.

Requirement

The alarm to be edited is displayed in the alarm window or the alarm view.

Procedure for touch operation

Proceed as follows to edit an alarm:

1. Touch the relevant alarm in the alarm view or in the alarm window. The alarm is selected.
2. Touch the button in the simple alarm view or in the enhanced alarm view.

Procedure for key operation

Proceed as follows to edit an alarm:

1. Select the desired alarm view or alarm window with TAB.
2. Select the desired alarm. Use the HOME, END, ▲ or ▼ keys.
3. Continue to press the key TAB until the button is selected in the simple alarm view or in the extended alarm view.
4. Confirm your entry by pressing the key ENTER.

Result

The system executes the additional functions of the alarm. Additional information on this topic may be available in your plant documentation.

10.11.4.5 Operating recipes

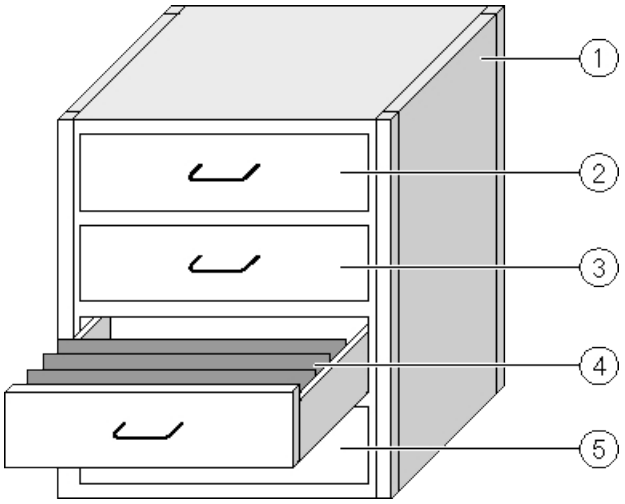
Structure of a recipe

Recipes

The recipe collection for the production of a product family can be compared to a file cabinet. A recipe which is used to manufacture a product corresponds to a drawer in a file cabinet.

Example:

In a plant for producing fruit juice, recipes are required for different flavors. There is a recipe, for example, for the flavors orange, grape, apple and cherry.



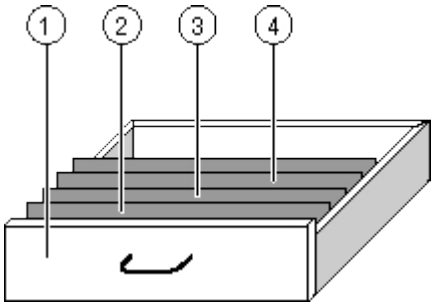
① File cabinet	Recipe collection	Recipes for a fruit juice plant
② Drawer	Recipe	Orange flavored drinks
③ Drawer	Recipe	Grape flavored drinks
④ Drawer	Recipe	Apple flavored drinks
⑤ Drawer	Recipe	Cherry flavored drinks

Recipe data records

The drawers of the file cabinet are filled with suspension folders. The suspension folders in the drawers represent records required for manufacturing various product variants.

Example:

Product variants of the flavor apple might be a soft drink, a juice or nectar, for example.



①	Drawer	Recipe	Product variants of apple flavored drinks
②	Suspension folder	Recipe data record	Apple drink
③	Suspension folder	Recipe data record	Apple nectar
④	Suspension folder	Recipe data record	Apple juice

Elements

In the figure showing the file cabinet, each suspension folder contains the same number of sheets. Each sheet in the suspension folder corresponds to an element of the recipe data record. All the records of a recipe contain the same elements. The records differ, however, in the value of the individual elements.

Example:

All drinks contain the same components: water, concentrate, sugar and flavoring. The records for soft drink, fruit juice or nectar differ, however, in the quantity of sugar used in production.

Recipes in the project

Overview

If recipes are used in a project, the following components interact:

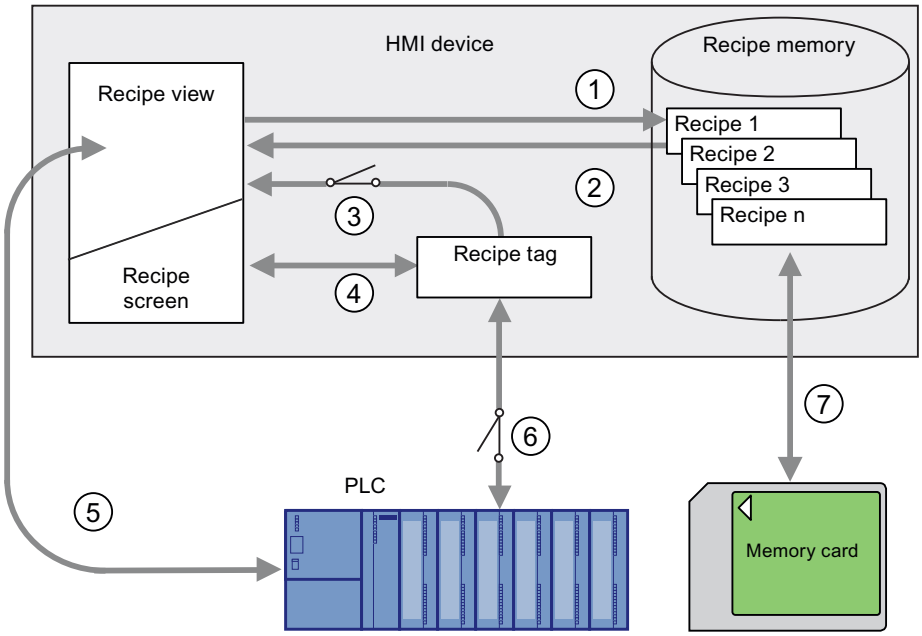
- HMI device recipe memory
  Recipes are saved in the form of data records in the HMI device recipe memory.
- Recipe view / recipe screen
  On the HMI device, recipes are displayed and edited in the recipe view or in a recipe screen.
  - The recipe data records from the internal memory of the HMI device are displayed and edited in the recipe view.
  - The values of the recipe tags are displayed and edited in the recipe screen.

    Note

    The same recipe tags can be configured in a variety of recipes. If you modify the value of a recipe tag, the synchronization changes the value of the recipe tag in all recipes.
- Recipe tags
  The recipe tags contain recipe data. When you edit recipes in a recipe screen, the recipe values are stored in recipe tags. The point at which the recipe tag values are exchanged with the PLC depends on the configuration.

Data flow

The following figure shows the data flow in a project with recipes:



① Editing, saving or deleting a recipe data record
② Display recipe data record
③ Synchronize or do not synchronize recipe tags
④ Display and edit recipe tags in the recipe screen
⑤ Write records from the recipe view to the PLC or read records from the PLC and display them in the recipe view.
⑥ Recipe tags are to the PLC online or offline
⑦ Export or import recipe data record to memory card

Simple recipe view

Layout

The simple recipe view consists of three areas:

- Recipe list
- Data record list
- Element list

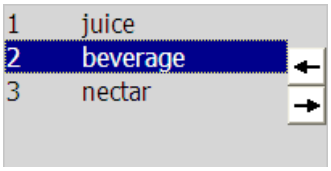


Figure 10-5 Simple recipe view - example with data record list

In the simple recipe view, each area is shown separately on the HMI device. You can use the shortcut menu to operate each of these display areas.

The simple recipe view always begins with the recipe list.

Operation

You can use the simple recipe view as follows, depending on the configuration:

- Create, change, copy or delete recipe data records
- Read recipe data records from the PLC or transfer to the PLC

Operator controls of the simple recipe view

Toggle between the display areas and the shortcut menus to operate the simple recipe views.

The table below shows the operation of the display area.

	Key	Function
Touching an entry	ENTER	The next lowest display area is opened, i.e. the data record list or the element list.
←	ESC	The previous display area opens.
→	▶	The shortcut menu of the display area opens.

The table below shows the operation of the shortcut menu:

	Key	Function
→	ESC	The menu is closed. The display area opens.
Touching the menu command	Input of the number of the menu command	The menu command is executed.

Shortcut menus of the simple recipe view

A shortcut menu can be called for each view area by pressing the button or the key. The shortcut menu lists the commands that are available in the active view area. A number is assigned to each command. You execute the command by entering its number. You can also use the system keys to execute certain commands.

The scope depends on the HMI device.

- Recipe list

No.	Menu command	Keys	Function
0	New	SHIFT + INS DEL	A new recipe data record is created for the selected recipe. If a start value is configured, it is displayed in the input field.
1	Displaying infotext	HELP	Displays the infotext configured for the simple recipe view.
2	Open	ENTER	The record list of the selected recipe opens.

- Data record list

	Menu command	Keys	Function
	New	SHIFT + INS DEL	Creates a new recipe data record. If a start value is configured, it is displayed in the input field.
	Deleting	INS DEL	The displayed record is deleted.
	Save as		The selected data record is saved under a different name. A dialog box opens where you can enter the name.
	Rename		Renames the selected data record. A dialog box opens where you can enter the name.
	Open	ENTER	The element list of the selected data record opens.
	Previous	ESC	The recipe list opens.
	To PLC		The displayed values of the selected data record are transferred from the HMI device to the PLC.
	From PLC		The recipe values from the PLC are displayed in the recipe view of the HMI device.
	Displaying infotext	HELP	Displays the infotext configured for the simple recipe view.

- Element list

	Menu command	Keys	Function
	Save		The selected record is renamed.
	To PLC		The displayed values of the selected data record are transferred from the HMI device to the PLC.
	From PLC		The recipe values from the PLC are displayed in the recipe view of the HMI device.
	Save as		The data record is saved under a new name. A dialog box opens where you can enter the name.
	Displaying infotext	HELP	Displays the infotext configured for the simple recipe view.
	Rename		Renames the selected data record. A dialog box opens where you can enter the name.
	Previous	ESC	The data record list opens.

Mouse control or touchscreen control of the simple recipe view

1. Select the desired recipe from the recipe view.
2. Click on the button.
   The shortcut menu is opened.
3. Select the desired menu command.
   The menu command is executed.
4. Alternatively, open the desired recipe in the recipe view.
   The data record list is displayed.
5. Open the desired data record. You could also use the button to open the shortcut menu and select a menu command.
   The menu command is executed.

Using the keyboard with the simple recipe view

1. Press the TAB key as many times as required to select the simple recipe view.
2. Select the desired recipe with the cursor keys.
3. Press the key.
   The shortcut menu is opened.
4. Press the cursor key as many times as required to select the menu command.
5. Confirm the menu command by pressing the key ENTER.
6. Alternatively, press the number of the desired menu command.
   The menu command is executed.

Creating a recipe data record

Introduction

Create a new recipe data record in the recipe list or in the record list. Then enter the values for the new record in the element list and save the record.

Requirement

A screen with a simple recipe view is displayed.

Procedure

Proceed as follows to create a recipe data record:

1. If the recipe list contains several recipes: Select the recipe for which you want to create a new recipe data record.
2. Open the recipe list menu.

3. Select the "0 new" menu command.
   Creates a new record.
   The element list of the new record opens.
4. Enter values for the data record elements.
   The tags of the record can be assigned default values depending on the configuration.
5. Open the menu of the element list and select the "0 Save" menu command.
6. Enter a name for the new record.
7. Confirm your entries.
   An existing data record is overwritten if you assign its number to a new data record.

Result

The new recipe data record is saved to the selected recipe.

Editing a recipe data record

Introduction

Edit the values of the recipe data records and save them to a recipe view.

Synchronization with the PLC

To display the current recipe values from the PLC in the simple recipe view, you first have to read the actual values in the element list from the PLC using menu command "2 From PLC".

Values changed in the recipe view are only activated after you transferred the modified data record to the PLC by means of menu command "1 To PLC".

Requirement

A screen with a recipe view is displayed.

Procedure

Proceed as follows to copy a recipe data record:

1. If the recipe list contains several recipes: Select the recipe which contains the relevant recipe data record.
2. Open the data record list.
3. Select the recipe data record you want to edit.
4. Open the element list.
5. Edit the element values as required.
6. Save your changes using menu command "0 Save".

Result

The edited recipe data record is saved to the selected recipe.

Deleting a recipe data record

Introduction

You can delete all the data records which are not required.

Requirement

A screen with a simple recipe view is displayed.

Procedure for touch operation

Proceed as follows to delete a new recipe data record:

1. If the recipe list contains several recipes: Select the recipe which contains the relevant recipe data record.
2. Open the data record list.
3. Select the data record you want to delete.
4. Open the menu.
5. Select the menu command "1 Delete".

Procedure for key operation

Proceed as follows to delete a new recipe data record:

1. If the recipe list contains several recipes: Select the recipe which contains the relevant recipe data record.
2. Open the data record list.
3. Select the data record you want to delete.
4. Press INS DEL.

Result

The data record is deleted.

Reading a recipe data record from the PLC

Introduction

The values of recipe elements are exchanged with the PLC via tags.

You can edit values which were saved to the recipes in the HMI device directly at plant level within the active project; this may be the case if a valve in the plant opens more than is indicated in the recipe. The values of the tags on the HMI device possibly no longer match the values in the PLC.

Read the values from the PLC and output these to the recipe view to synchronize the recipe values.

Requirement

A screen with a simple recipe view is displayed.

Procedure

To transfer a recipe data record to the PLC, proceed as follows:

1. If the recipe list contains several recipes: Select the recipe which contains the relevant recipe data record.
2. Select the element list of the recipe data record to which you want to apply the values from the PLC.
3. Open the menu.
4. Select menu command "2 From PLC".
   The values are read from the PLC.
5. Save the displayed values to the HMI device using menu command "0 Save".

Result

The values were read from the PLC, are visible on the HMI device and were saved to the selected recipe data record.

Note

Basic Panels

With Basic Panels, the "From PLC" menu command can also be configured for the data record list: In this case, you can also select the "From PLC" menu command in the data record list.

Transferring a recipe data record to the PLC

Introduction

You must transfer the values to the PLC to activate a changed recipe data record for the process.

The values displayed in the recipe view are transferred to the PLC.

Requirement

A screen with a simple recipe view is displayed.

Procedure

To transfer a recipe data record to the PLC, proceed as follows:

1. If the recipe list contains several recipes: Select the recipe which contains the relevant recipe data record.
2. Select the element list of the recipe data record whose values you want to transfer to the PLC.
3. Open the menu.
4. Select menu command "1 To PLC".

Result

The values of the recipe data record were transferred to the PLC and are active in the process.

Note

Basic Panels

With Basic Panels, the "To PLC" menu command can also be configured for the data record list: In this case, you can also select the "To PLC" menu command in the data record list.

10.12 Performance features

10.12.1 Engineering system

Engineering system

The following tables help you assess whether your project still meets the performance specifications of the engineering system.

In addition to the specified limits, allowances must also be made for restrictions imposed by main memory resources. WinCC uses up to 2 GB of RAM, depending on the operating system. It is nonetheless useful to install more than 2 GB of main memory on the PC if running many applications with high memory requirements in parallel.

Project system limits

Different system limits apply for the engineering system, depending on whether a 32-bit or 64-bit operating system is used.

	32-bit operating system	64-bit operating system
Number of HMI devices in the project	20	40
Number of HMI tags [1]	80,000	260,000
Number of logging tags	8,000	20,000
Number of blocks (faceplates, user data types)[3]	10,000	20,000
Number of screens	3,000	6,000
Number of screen objects per screen	3,200	3,200
Number of screen objects	320,000	640,000
Number of alarms [2] [3]	20,000	60,000
Number of texts [3]	300,000	600,000
Number of text lists and graphic lists [3]	10,000	20,000
Number of entries per text list	3,000	3,000
Number of languages	32	32
Number of global libraries [3]	20	20
Number of objects in the project library [3]	300,000	800,000

1) Including logging tags.
2) With an average of 3 texts and a dynamic parameter
3) Including the objects configured in the "Program PLC" area

WinCC system limits for an HMI device

	32-bit operating system	64-bit operating system
Number of HMI tags [1]	80,000	260,000
Number of logging tags	8,000	20,000
Number of logs	500	1,000
Number of screens	1000	2,000
Number of screen objects per screen	3,200	3,200
Number of screen objects	320,000	640,000
Number of function lists	30,000	60,000
Number of animations and local scripts	50,000	100,000
Number of user-defined functions	1,000	2,000
Number of tasks	500	1,000
Number of alarms [2]	20,000	60,000
Number of recipes	1,000	2,000
Number of recipe elements	10,000	20,000
Number of texts	300,000	600,000

	32-bit operating system	64-bit operating system
Number of text lists and graphic lists	1,000	2,000
Number of entries per text list	5,000	10,000
Number of users	200	200
Number of reports	300	600

1) Including logging tags.
2) With an average of 3 texts and a dynamic parameter

System limits during migration

You can migrate projects which are beyond the specified system limits in one or more areas.

An alarm will be output if migration creates a project whose limits are beyond the specified system limits. You must then adapt the project after migration to within the specified system limits to ensure safe operation in WinCC.

10.12.2 Basic Panel

Basic Panel

The following table helps you assess whether your project meets the performance features of the HMI device.

The specified maximum values are not additive. It cannot be guaranteed that configurations running on the devices at the full system limits will be functional.

In addition to the specified limits, allowances must be made for restrictions imposed by configuration memory resources.

Tags

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of tags in the project	250	500	250 (mono) 500 (color)	500	500	500
Number of PowerTags	--	--	--	--	--	--
Number of elements per array	100	100	100	100	100	100
Number of local tags	--	--	--	--	--	--
Number of structures	--	--	--	--	--	--
Number of structure elements	--	--	--	--	--	--

Alarms

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of alarm classes	32	32	32	32	32	32
Number of discrete alarms	200	200	200	200	200	200
Number of analog alarms	15	15	15	15	15	15
Length of an alarm in characters	80	80	80	80	80	80
Number of process values per alarm	8	8	8	8	8	8
Size of the alarm buffer	256	256	256	256	256	256
Number of queued alarm events	64	64	64	64	64	64

Screens

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of screens	50	50	50	50	50	50
Number of fields per screen	30	30	30	30	30	30
Number of tags per screen	30	30	30	30	30	30
Number of complex objects per screen [1)]	5	5	5	5	5	5
Number of array elements per screen [2)]	100	100	100	100	100	100

1) Complex objects include: Bars, sliders, symbol library, clock, and all objects from the Controls area.

2) Array elements contained in recipes are included in the count.

Recipes

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of recipes	5	5	5	5	5	5
Number of elements per recipe[1)]	20	20	20	20	20	20
User data length in bytes per data record	--	--	--	--	--	--
Number of data records per recipe	20	20	20	20	20	20
Reserved memory for data records in the internal Flash	40 KB	40 KB	40 KB	40 KB	40 KB	40 KB

1) If arrays are used, each array element counts as one recipe element

Logs

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of logs	--	--	--	--	--	--
Number of entries per log (including all log segments) [1]	--	--	--	--	--	--
Number of log segments	--	--	--	--	--	--
Cyclic trigger for tag logging	--	--	--	--	--	--
Number of tags that can be logged per log	--	--	--	--	--	--

1) The number of entries for the "segmented circular log" logging method is the total number for all sequential logs. The product of the number of sequential logs and the number of data records per sequential log may not exceed the system limit

Trends

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of trends	25	25	25	25	25	25

Text lists and graphics lists

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of graphics lists	100	100	100	100	100	100
Number of text lists	150	150	150	150	150	150
Number of entries per text or graphics list	30	30	30	30	30	30
Number of graphic objects	500	500	500	500	500	500
Number of text elements	500	500	500	500	500	500

Scripts

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of scripts	--	--	--	--	--	--

Communication

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of connections	4	4	4	4	4	4
Number of connections based on "SIMATIC HMI HTTP"	--	--	--	--	--	--

Help system

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of characters in a help text	500	500	500	500	500	500

Languages

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of runtime languages	5	5	5	5	5	5

Scheduler

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Time-triggered tasks [1)]	--	--	--	--	--	--

1) Event-triggered tasks are not relevant for the system limits

User administration

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Number of user groups	50	50	50	50	50	50
Number of authorizations	32	32	32	32	32	32
Number of users	50	50	50	50	50	50

Project

	KP300 Basic	KP400 Basic	KTP400 Basic	KTP600 Basic	KTP1000 Basic	TP1500 Basic
Size of the project file "*.srt"	1024 KB	1024 KB	1024 KB	1024 KB	1024 KB	1024 KB

See also

S7-1200 manual (http://support.automation.siemens.com/WW/view/en/36932465/0/en)

10.12.3 Basic Panel 2nd Generation

Basic Panel 2nd Generation

The following table helps you assess whether your project meets the performance features of the HMI device.

The specified maximum values are not additive. It cannot be guaranteed that configurations running on the devices at the full system limits will be functional.

In addition to the specified limits, allowances must be made for restrictions imposed by configuration memory resources.

Tags

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of tags in the project	800	800	800	800
Number of PowerTags	--	--	--	--
Number of elements per array	100	100	100	100
Number of local tags	--	--	--	--
Number of structures	--	--	--	--
Number of structure elements	--	--	--	--

Alarms

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of alarm classes	32	32	32	32
Number of discrete alarms	1000	1000	100	100
Number of analog alarms	25	25	25	25
Length of an alarm in characters	80	80	80	80
Number of process values per alarm	8	8	8	8

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Size of the alarm buffer	256	256	256	256
Number of queued alarm events	64	64	64	64

Screens

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of screens	250	250	250	250
Number of fields per screen	100	100	100	100
Number of tags per screen	100	100	100	100
Number of complex objects per screen [1)]	150	150	150	150
Number of recipe displays per screen	10	10	10	10
Number of trend displays per screen	8	8	8	8
Number of alarm displays per screen	20	20	20	20
Number of user displays per screen	1	1	1	1
Number of system diagnostic displays per screen	150	150	150	150
Number of array elements per screen [2)]	100	100	100	100

1) Complex objects include: Bars, sliders, symbol library, clock, and all objects from the Controls area.

2) Array elements contained in recipes are included in the count.

Recipes

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of recipes	50	50	50	50
Number of elements per recipe[1)]	100	100	100	100
User data length in KB per data record	32	32	32	32
Number of data records per recipe	100	100	100	100
Reserved memory for data records in the internal Flash	256 KB	256 KB	256 KB	256 KB

1) If arrays are used, each array element counts as one recipe element

Logs

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of logs	2	2	2	2
Number of entries per log (including all log segments) [1)]	10	10	10	10
Number of log segments	10000	10000	10000	10000

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Cyclic trigger for tag logging	400	400	400	400
Number of tags that can be logged per log	1 s	1 s	1 s	1 s

1) The number of entries for the "segmented circular log" logging method is the total number for all sequential logs. The product of the number of sequential logs and the number of data records per sequential log may not exceed the system limit

Trends

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of trends	25	25	25	25

Text lists and graphics lists

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of graphics lists	100	100	100	100
Number of text lists	300	300	300	300
Number of entries per text or graphics list	100	100	100	100
Number of graphic objects	1000	1000	1000	1000
Number of text elements	2500	2500	2500	2500

Scripts

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of scripts	--	--	--	--

Communication

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of connections	4	4	4	4
Number of connections based on "SIMATIC HMI HTTP"	--	--	--	--

Help system

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of characters in a help text	500	500	500	500

Languages

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of runtime languages	10	10	10	10

Scheduler

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Time-triggered tasks [1)]	--	--	--	--

1) Event-triggered tasks are not relevant for the system limits

User administration

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Number of user groups	50	50	50	50
Number of authorizations	32	32	32	32
Number of users	50	50	50	50

Project

	KTP400 Basic	KTP700 Basic	KTP900 Basic	KTP1200 Basic
Size of the project file "*.srt"	10 MB	10 MB	10 MB	10 MB

10.12.4 General technical data

10.12.4.1 Permitted characters

Introduction

The following table shows the restrictions that must be observed when allocating names and passwords.

Permitted characters

Names / passwords	Restriction
Names of objects	Do not use the special characters ? " / \ * < > %
Names and passwords in user view	Do not use the special characters ? " / \ § & $ %
Names of alarm logs	In the storage location file - RDB, file - CSV (ASCII) and file -TXT (Unicode) do not use the characters \ / * ? : " < > |
	For the storage location database, use only the characters a-z A-Z 0-9 _ @ # $ The characters _ @ # $ may not be used as the first character of a name
Name of HMI tags	HMI tag names may not start with the @ character
Names of screens	Do not use the special characters ? " / \ * < >
Names of AuditTrails	Do not use any special characters
Names of connections	Do not use spaces or special characters
Name of VB functions	Do not use spaces, special characters or VB keywords
Name of Windows user groups and Windows users when using SIMATIC Logon	Do not use the special characters / \

10.12.4.2 Recommended printers

Recommended printers

The current list of printers recommended for use with the HMI devices is available on the Internet at:

Link to the current printer list (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=0&lang=en&referer=%2fWW%2f&func=cslib.csinfo&siteid=csius&caller=view&extranet=standard&viewreg=WW&nodeid0=10805558&objaction=csopen)

Note

All HMI devices except for a PC and Panel PC support only one printer at their USB port, even if several ports are available.

See also

Printer list (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=0&lang=en&referer=%2fWW%2f&func=cslib.csinfo&siteid=csius&caller=view&extranet=standard&viewreg=WW&nodeid0=10805558&objaction=csopen)

10.12.4.3 Memory requirements of recipes for Basic Panels

Introduction

The following calculation of memory requirements of recipes is valid for Basic Panels, OP 77A, and TP 177A devices.

Restrictions

The HMI device provides 39 KB of memory space for recipes. This memory space may not be exceeded. The total memory space for recipes is calculated as follows: Total of all recipes + recipe with highest memory requirement.

Each recipe may not exceed a maximum memory space of 19 KB.

Calculation of memory requirements

The memory space requirement of each recipe (in KB) is calculated based on the three addends D1 + D2 + D3.

Valid is::

- D1 = number of data records x M

  Rule for M (size of a data record):
  M = 1 x number of elements of a byte + 2 x number of elements of 2 bytes + 4 x number of elements of 4 bytes + 8 x number of elements of 8 bytes + K

  Rule for K (size of the string elements):
  K = number of string elements x (string length + 1) x 2

- D2 - data record size

  D2 = 4 + number of languages x 8 + number of languages x (4 + 4 x number of data records + (length of the data record name + 1) x 2 x number of data records) + 8 + 8 x number of data records

  Or rewritten:

  D2 = 12 + 8 x number of data records + number of languages x (12 + number of data records x (4 + (length of the data record name +1) x 2))

- D3 - shared memory
  D3 = 14 + number of elements

Note

Arrays and single elements can be calculated as described above.

10.13 Migrating to WinCC in the TIA Portal

10.13.1 Overview of migration to WinCC V12

Overview of the section "Migration to WinCC in the TIA Portal"

SIMATIC WinCC offers a number of functional changes in the TIA Portal. Some functions differ from the functions that you know from familiar environments such as WinCC V7 or WinCC flexible.

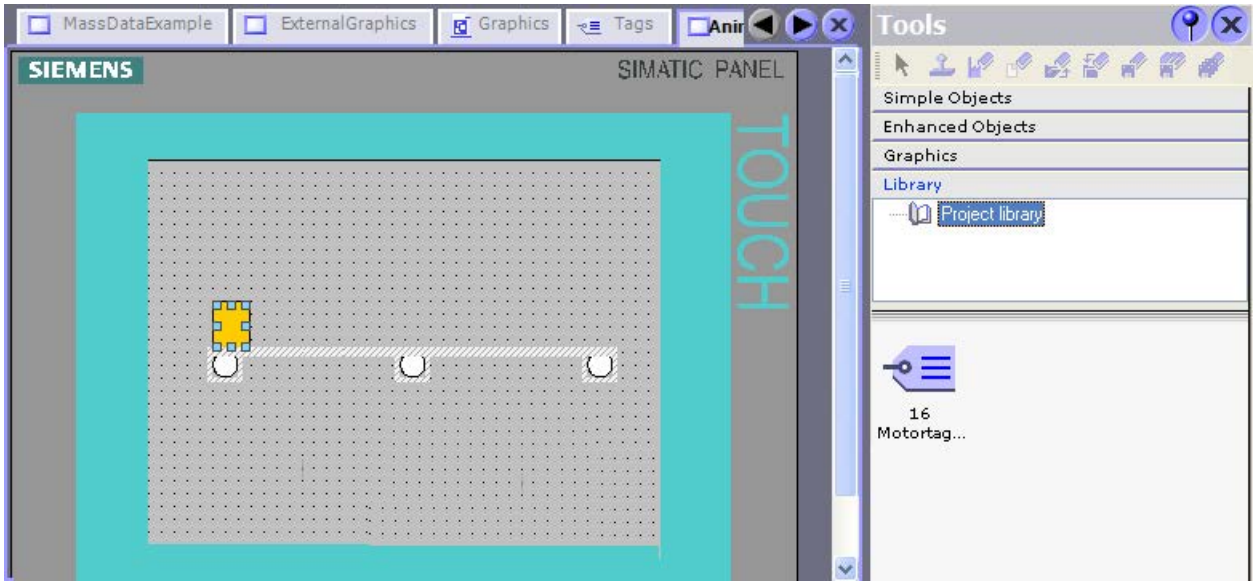
This document provides an overview of the special functions and procedures in SIMATIC WinCC in the TIA Portal.

These functions and procedures are fundamentally different from the WinCC V7 and WinCC flexible version, or have a different name.

10.13.2 Libraries

Libraries in WinCC flexible

Libraries are a collection of pre-configured screen objects. They expand the number of available screen objects and increase engineering efficiency, because library objects are always available for reuse; there is no need to reconfigure them.



WinCC flexible enables you to create two library types:

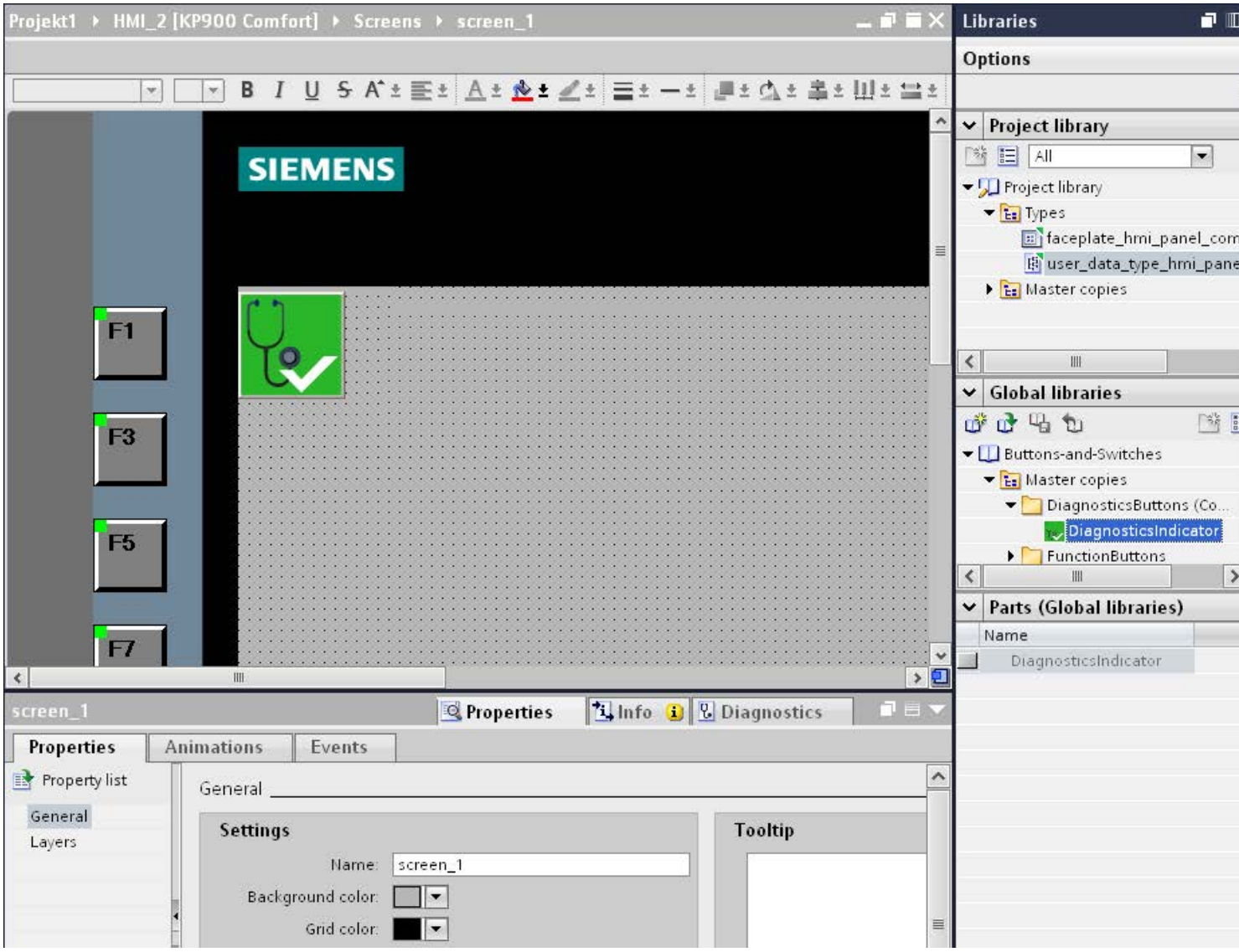
- Project library
- Global library

A library can contain all the WinCC flexible objects, such as screens, tags, graphic objects, or alarms.

How do I configure libraries with WinCC in the TIA Portal?

In WinCC, you also configure the "Project library" and the "Global library".

You can no longer store any system functions in libraries, as was the case in WinCC flexible.



Both the "Project library" and the "Global library" contain the two folders, "Copy templates" and "Types". You can create or use the library objects as a copy template, or as a type.

- Copy templates
  Use copy templates to create independent copies of the library object.
- Types
  Create instances of objects of the "Types" folder and use the instances in your project. The instances are bound to their respective type. Changes to an instance also change all other instances. Types are marked by a green triangle in the "Libraries" task card.
- Managing the library objects
  You can only copy and move library objects within the same library.

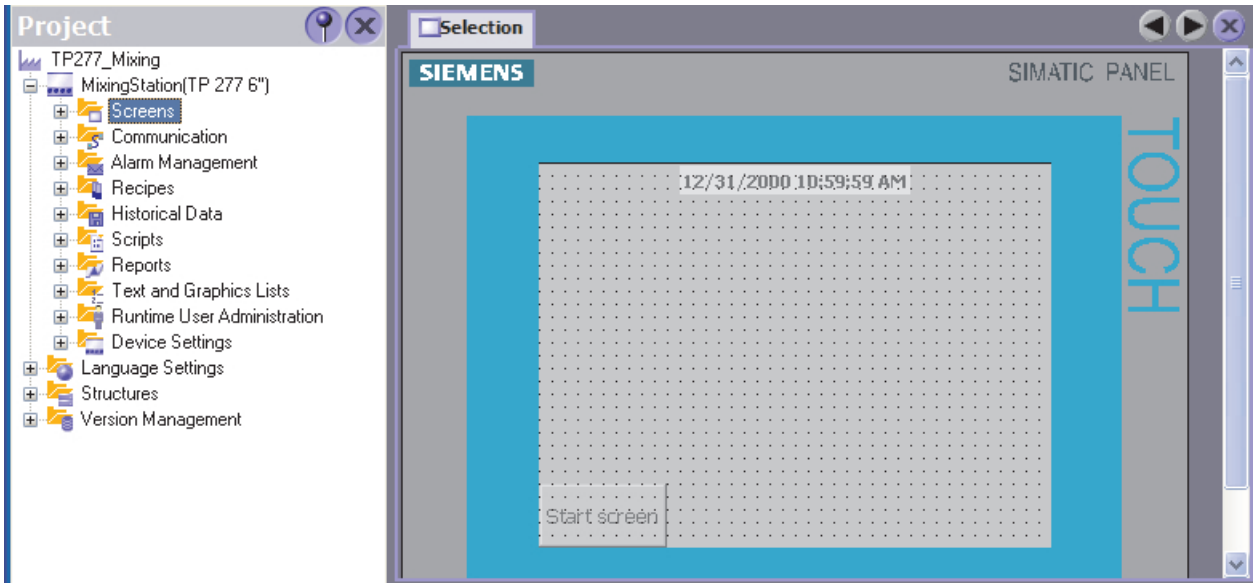
For more detailed information, see:

Libraries in WinCC (Page 8766)

10.13.3 Screens and templates

Screens and templates in WinCC flexible

In WinCC flexible, you create screens that an operator can use to control and monitor machines and plants. When you create your screens, the object templates provided support you in visualizing your plant, displaying processes and defining process values.



The project has a template for every HMI device. You can centrally configure the function keys and objects for your project in these templates.

Every screen based on this template will contain the function keys and objects that you configured in the template. Changes to an object or of a function key assignment in the template are applied to the object in all the screens, which are based on this template.

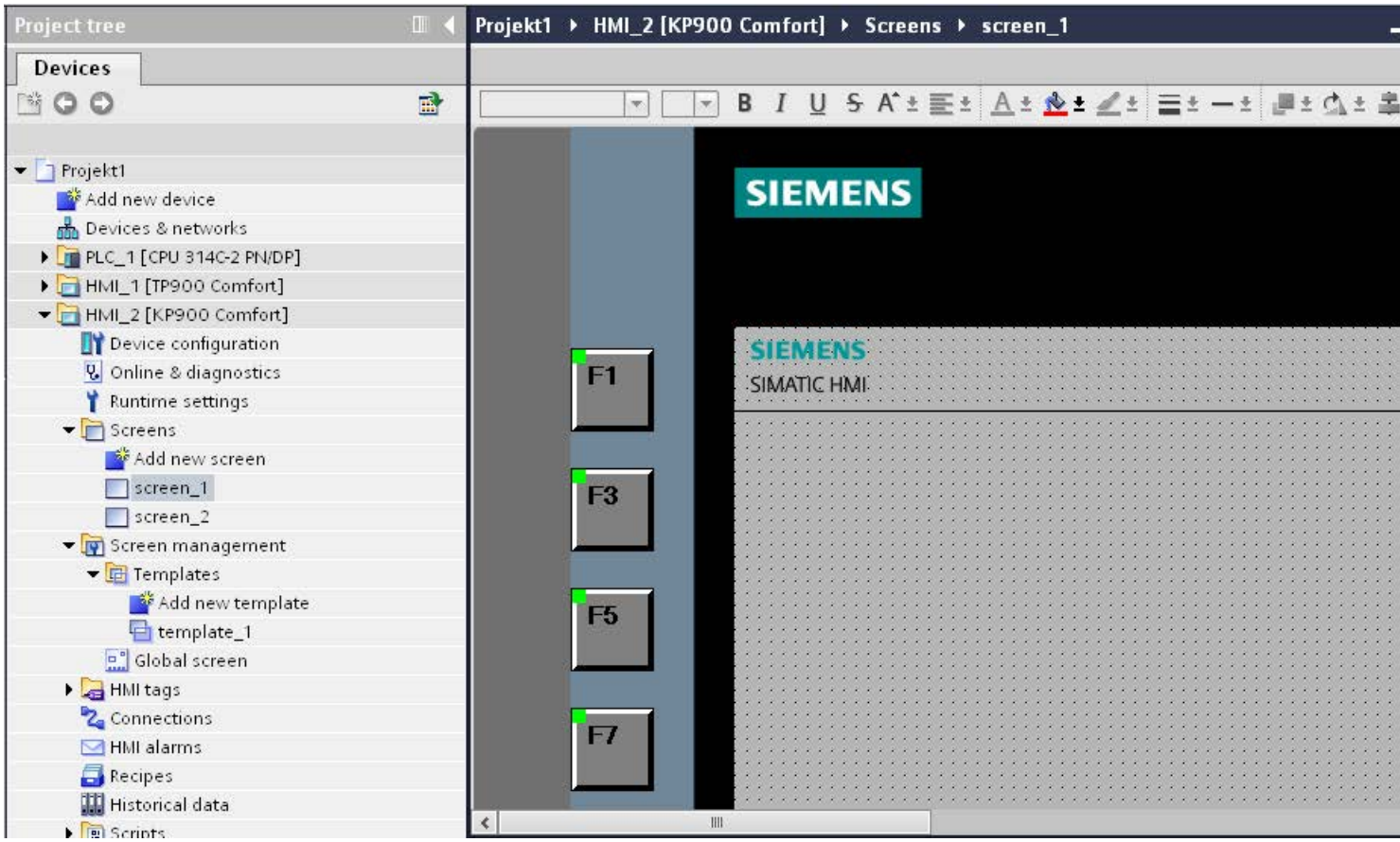
How do I configure screens and templates with WinCC in the TIA Portal?

In WinCC, you also configure "Templates" and a "Global screen" along with the "Screens".

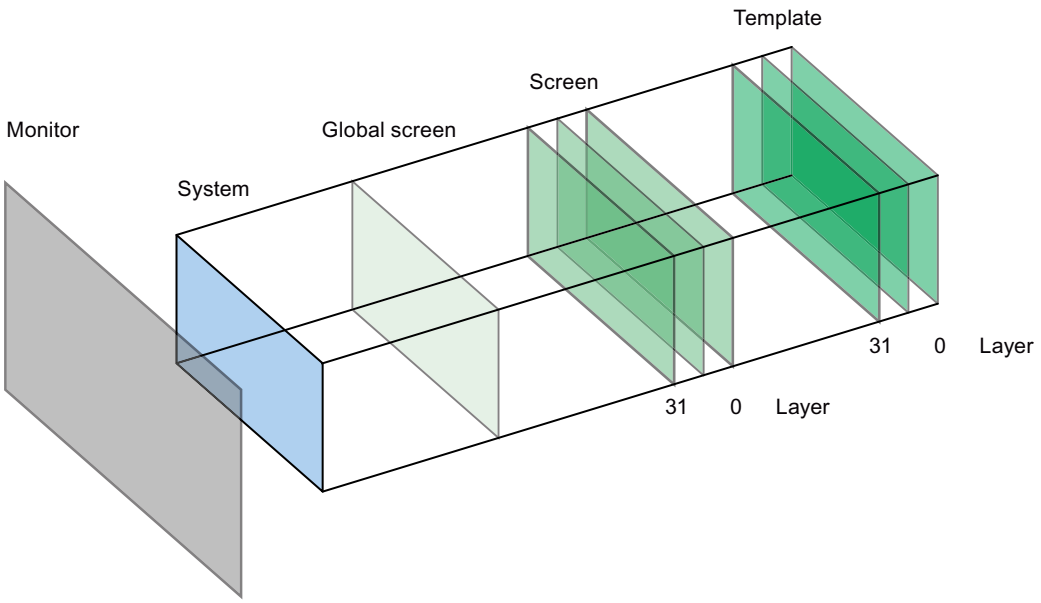
You determine functions and objects in the template which then apply to all screens based on this template. You can create multiple templates in WinCC.

In the "Global screen", define the elements which are independent of the template used for all screens of an HMI device. The "Alarm window" and "Alarm indicator" objects are available for use as global objects. For HMI devices with function keys, configure the function keys in the "Global Screen" editor.

You can also configure a "System Diagnostics Window" in the global screen of Comfort Panels.



Excluding the controls, the screens are displayed in runtime in the following order:



For more detailed information, see:

Screen basics (Page 7961)

Using technology functions 11

11.1 Counting, measurement and position input

11.1.1 Counting, measurement and position input (S7-1500)

11.1.1.1 The basics of counting, measurement and position detection

Overview of technology modules and properties

Technology modules

The table below summarizes the performance features of the technology modules for counting, measurement and position input:

Table 11-1 Technology modules for counting, measurement and position input

Property	S7-1500		ET 200SP	
	TM Count 2x24V	TM PosInput 2	TM Count 1x24V	TM PosInput 1
Number of channels	2	2	1	1
Maximum signal frequency	200 kHz	1 MHz	200 kHz	1 MHz
Maximum count frequency for incremental encoders with quadruple evaluation	800 kHz	4 MHz	800 kHz	4 MHz
Maximum counter / position value range	32 Bit	32 Bit/31 Bit	32 Bit	32 Bit/31 Bit
RS422/TTL connection to incremental and pulse encoder	—	X	—	X
24 V connection to incremental and pulse encoder	X	—	X	—
SSI absolute encoder connection	—	X	—	X
5 V encoder supply	—	X	—	—
24 V encoder supply	X	X	X	X
Number of digital inputs per channel	3	2	3	2
Number of digital outputs per channel	2	2	2	2
Gate control	X	X	X	X

Property	S7-1500		ET 200SP	
	TM Count 2x24V	TM PosInput 2	TM Count 1x24V	TM PosInput 1
Capture function	X	X	X	X
Synchronization	X	X	X	X
Comparison functions	X	X	X	X
Frequency, velocity and period measurement	X	X	X	X
Isochronous mode support	X	X	X	X
Diagnostic interrupt support	X	X	X	X
Hardware interrupt support	X	X	X	X
Configurable filter for count signals and digital inputs	X	X	X	X

Overview of applications

Introduction

The technology module is configured and assigned parameters using the configuration software.

The technology module functions are controlled and monitored via the user program.

System environment

The technology modules can be used in the following system environments:

Applications	Components required	Configuration software	In the user program
Central and distributed operation in an S7-1500 system with a corresponding ET 200 system	• S7-1500 automation system • ET 200 distributed I/O system • Technology module	STEP 7 (TIA Portal): • Device configuration with hardware configuration (HWCN) • Parameter settings with High_Speed_Counter technology object	Counting and measuring functions: High_Speed_Counter instruction for the technology object Position input with SSI absolute encoder: Direct access to technology module control and feedback interface in the I/O data
		STEP 7 (TIA Portal): Device configuration with hardware configuration (HWCN) in operating mode "Position Detection for Motion Control"	Control by means of technology object

Applications	Components required	Configuration software	In the user program
Distributed operation in an S7-300/400 system	• S7-300/400 automation system • ET 200 distributed I/O system • Technology module	STEP 7 (TIA Portal): Device configuration and parameter settings with hardware configuration (HWCN) STEP 7: Device configuration and parameter settings with HSP	Direct access to technology module control and feedback interface in the I/O data
Distributed operation in a third-party system	• Third-party automation system • ET 200 distributed I/O system • Technology module	Third-party configuration software: Device configuration and parameter settings with GSD file	Direct access to technology module control and feedback interface in the I/O data

Parameter assignment options

In an S7-1500 system, you have two options for parameter assignment and control of technology module functions:

- Configuration using the High_Speed_Counter technology object and control with the corresponding High_Speed_Counter instruction
  The technology module control and feedback interface is accessed with the technology object.
- Parameter settings with HWCN
  The control and feedback interface of the technology module is accessed using direct access to the I/O data.

Configuration via technology object

For central and distributed use in an S7-1500 system, we recommend simple configuration with graphic support using a High_Speed_Counter technology object. A detailed description of this configuration can be found in section High_Speed_Counter technology object (Page 9024) and below.

You can define "Operation with technology object" for the technology module in the device configuration: see section Operating mode (Page 9048).

You can assign the technology module and counting channel in the basic parameters of the technology object: see section Basic parameters (Page 9029).

Parameter setting via HWCN

Additional support for parameter settings via HWCN is available in the context-sensitive help for the parameters in STEP 7 (TIA Portal). A description of the control and feedback interface is available in the following sections:

Assignment of the control interface (Page 9074)

Assignment of the feedback interface (Page 9077)

Recording of count signals

Counting with incremental or pulse encoder

Counting refers to the recording and adding up of events. The counters of the technology modules capture and evaluate pulse and incremental signals. The count direction can be specified using encoder or pulse signals or through the user program.

You can control counting processes using the digital inputs of the technology module. You can switch the digital outputs exactly at defined counter values, regardless of the user program.

You can configure the response of the counters using the functionalities described below.

Counter limits

The counter limits define the counter value range used. The counter limits are configurable and can be modified during runtime using the user program.

The highest counter limit that can be set is 2147483647 (2^{31}–1). The lowest counter limit that can be set is –2147483648 (–2^{31}).

You can configure the response of the counter at the counter limits:

- Continue or stop counting upon violation of a counter limit (automatic gate stop)
- Set counter value to start value or to opposite counter limit upon violation of a counter limit

Start value

You can configure a start value within the counter limits. The start value can be modified during runtime by the user program.

The technology module can, as configured, set the current counter value to the start value upon synchronization, upon Capture function activation, upon violation of a counter limit or when the gate is opened.

Gate control

Opening and closing the hardware gate and software gate defines the period of time during which the counting signals are captured.

The hardware gate is controlled externally via a digital input of the technology module. The software gate is controlled via the user program. The hardware gate can be enabled through parameter assignment. The software gate (bit in the control interface of the cyclic I/O data) cannot be disabled.

Capture

You can configure an external reference signal edge that triggers the saving of the current counter value or position value as a Capture value. The following external signals can trigger the Capture function:

- Rising or falling edge of a digital input
- Both edges of a digital input
- Rising edge of signal N at the encoder input (for incremental and pulse encoders)

For incremental and pulse encoders, you can configure whether counting continues from the current counter value or from the start value after the Capture function.

Synchronization

You can configure an external reference signal edge to load the counter with the specified start value. The following external signals can load the counter with the start value:

- Rising or falling edge of a digital input
- Rising edge of signal N at the encoder input
- Rising edge of signal N at the encoder input depending on the level of the assigned digital input

Hysteresis

You can specify hysteresis for the comparison values, within which a digital output is prevented from switching again. An encoder can come to a standstill at a specific position, and slight movements may make the counter value fluctuate around this position. If a comparison value or a counting limit lies within this fluctuation range, the corresponding digital output will be switched on and off with corresponding frequency if hysteresis is not used. The hysteresis prevents these unwanted switching operations.

Position input with SSI absolute encoder

Description

You can use the TM PosInput technology modules with an SSI absolute encoder for position detection. The technology module reads the position value via a synchronous serial interface from the SSI absolute encoder and makes it available to the controller.

You can switch the digital outputs of the technology module exactly at defined position values, regardless of the user program. Position input with an SSI absolute encoder does not involve gate control. Due to system constraints, synchronization is not possible with an SSI absolute encoder.

To do this, select the signal type "Absolute encoder (SSI)" in the parameter setting (HWCN) of the technology module in STEP 7 (TIA Portal).

Gray-dual conversion

Gray-code and dual-code SSI absolute encoders are supported.

Capture

You can configure one or both edges of a digital input that triggers a saving of the current position value as Capture value.

Hysteresis

You can specify hysteresis for the comparison values, within which a digital output is prevented from switching again. An encoder can come to a standstill at a specific position, and slight movements may make the position value fluctuate around this position. If a comparison value or a limit lies within this fluctuation range, the corresponding digital output is switched on and off with corresponding frequency if hysteresis is not used. The hysteresis prevents these unwanted switching operations.

Range for position value

You can specify a frame length of 10 bits to 40 bits for the SSI absolute encoder. The configurable bit numbers of the LSB and the MSB of the position value in the frame define the value range. The technology module can read a position value with a maximum length of31 bits and communicate it to the controller. The position value is treated as unsigned positive value and can assume values between 0" and "$2^{(MSB-LSB+1)-1}$".

Complete SSI frame

Instead of having a measured variable returned, you can choose to have the least significant 32 bits of the current unprocessed SSI frame returned. This provides you with encoder-specific additional bits, such as error bits, in addition to the position value. If the SSI frame is shorter than 32 bits, the complete SSI frame is returned right-aligned and the top unused bits are returned with "0" in the feedback interface.

Behavior at the counting limits

Violation of a counting limit

The high counting limit is violated when the current counter value is equal to the high counting limit and another upward count pulse is received. The low counting limit is violated when the current counter value is equal to the low counting limit and another downward count pulse is received.

The appropriate status bit is set in the feedback interface in the event of limit violation:

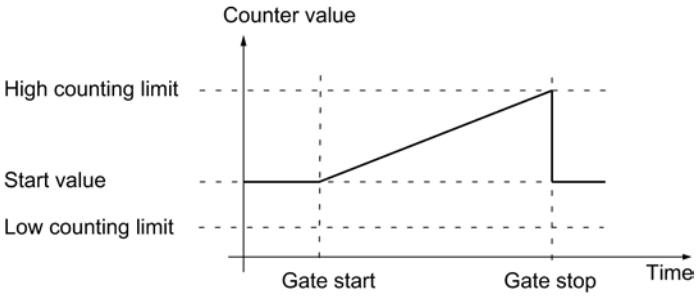
Counting limit violated	Status bit
High counting limit	EVENT_OFLW is set
Low counting limit	EVENT_UFLW is set

You can reset the status bits with RES_EVENT .

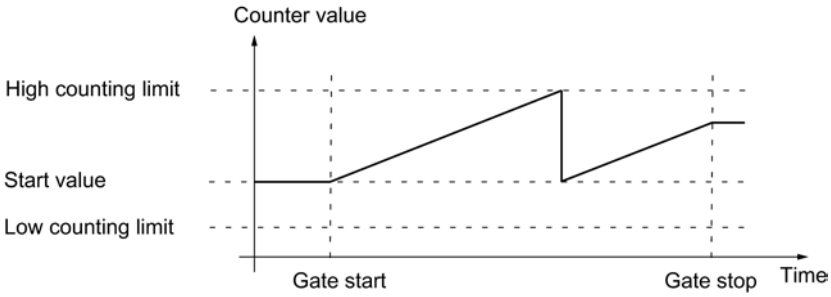
You can configure whether or not and from which counter value counting is to continue following counting limit violation.

Examples

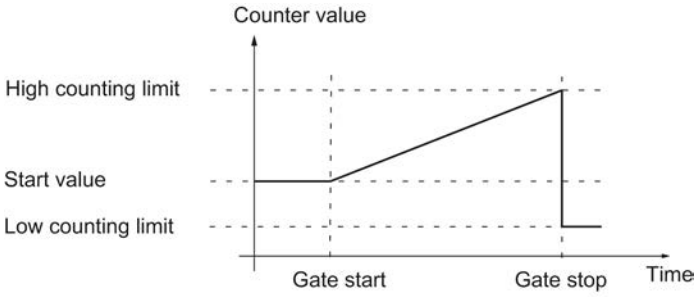
The figure below shows an example for terminating the counting process (automatic gate stop) after an overflow and setting the counter to the start value:



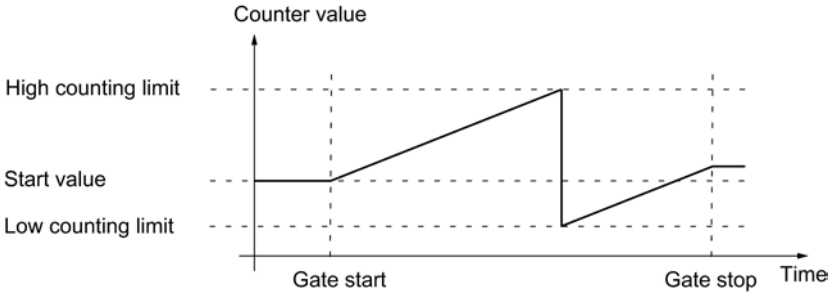
The figure below shows an example for continuing the counting process after an overflow and setting the counter to the start value:



The figure below shows an example for terminating counting after setting the counter to the opposite counting limit:



The figure below shows an example for continuing the counting process after an overflow and setting the counter to the opposite counting limit:



Gate control with incremental or pulse encoder

Many applications require counting processes to be started or stopped in accordance with other events. In such cases, counting is started and stopped using the gate function.

The technology modules have two gates for each channel. These define the resulting internal gate:

- Software gate
- Hardware gate

Software gate

The software gate of the channel is opened and closed with the control bit (Page 9074) SW_GATE. The status of the software gate is indicated by the feedback bit (Page 9077) STS_SW_GATE .

Hardware gate

The hardware gate is optional. You open and close the hardware gate by means of signals at the configured digital inputs of the channel.

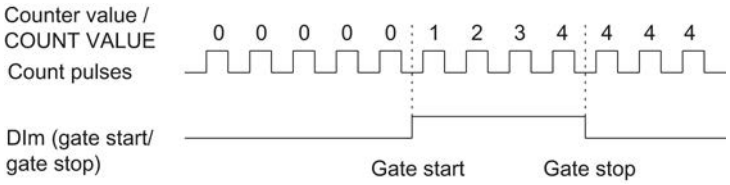
Note

The configured input filters delay the control signal of the digital input.

The status of a DIm digital input is indicated by the respective feedback bit (Page 9077) STS_DIm .

Level-triggered opening and closing of the hardware gate with a digital input

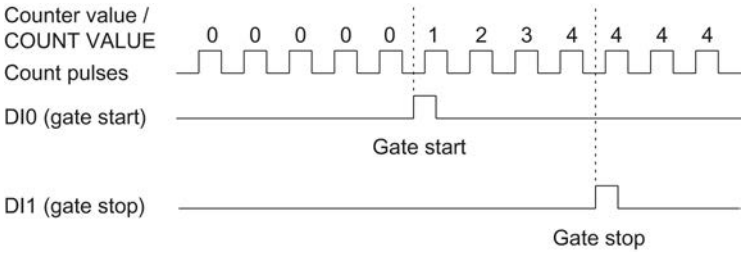
The figure below shows an example of level-triggered opening and closing with a digital input. The digital input is configured to be active with high level:



As long as the digital input is active, the hardware gate is open and the count pulses are counted. The hardware gate is closed when the digital input becomes inactive. The counter value stays constant and ignores any further count pulses.

Edge-triggered opening and closing of the hardware gate with two digital inputs

The figure below shows an example of opening and closing with two digital inputs. The two digital inputs are configured so that the rising edge of the hardware gate opens:



The hardware gate is opened with the configured edge at the digital input that is configured for opening. The hardware gate is closed with the configured edge at the digital input that is configured for closing.

Internal gate

Internal gate

The internal gate is open if the software gate is open and the hardware gate is open or has not been configured. The status of the internal gate is indicated by the feedback bit (Page 9077) STS_GATE.

If the internal gate is open, counting is started. If the internal gate is closed, all other count pulses are ignored and counting is stopped.

If you want to control a counting process with the hardware gate only, the software gate must be open. If you do not configure a hardware gate, the hardware gate is considered to be always open. In this case, you open and close the internal gate with the software gate only.

Hardware gate	Software gate	Internal gate
Open/not configured	open	open
Open/not configured	closed	closed

Hardware gate	Software gate	Internal gate
closed	open	closed
closed	closed	closed

When you configure the counter behavior, you can specify whether the counting process is to start from the start value or from the current counter value when the internal gate is opened.

The internal gate can also be automatically closed upon violation of a counting limit. The software or hardware gate must then be closed and reopened to continue counting.

Counter behavior at gate start

You have the following configuration options for counter behavior upon gate start:

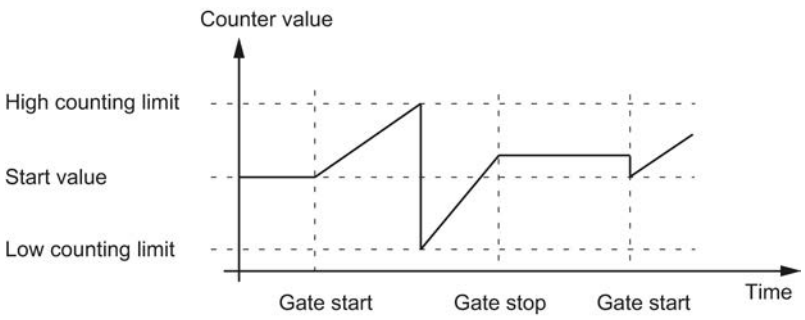
- Setting counter to start value
- Continuing with the current counter value

Setting counter to start value

Counter behavior is as follows for this configuration:

Each counting process starts with the start value when the internal gate is opened.

The figure below shows an example for continuing the counting process after counter is set to the start value:

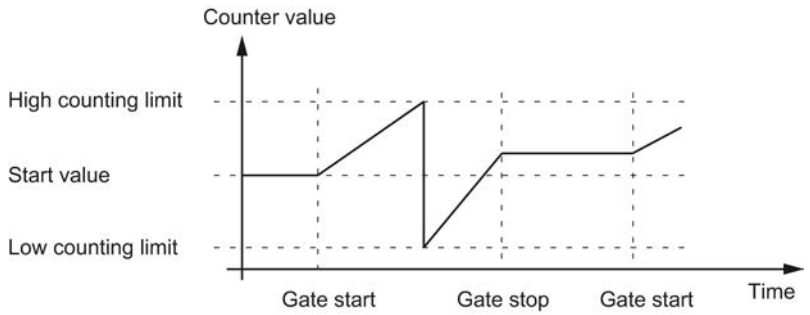


Continuing with the current counter value

Counter behavior is as follows for this configuration:

Each counting process starts from the current counter value after the internal gate is reopened.

The figure below shows an example for continuing the counting process with the current counter value:



Capture

Capture with incremental or pulse encoder

Description

The "Capture" function is used to save the current counter value with an external reference signal. You can configure the Capture function for the following reference signals:

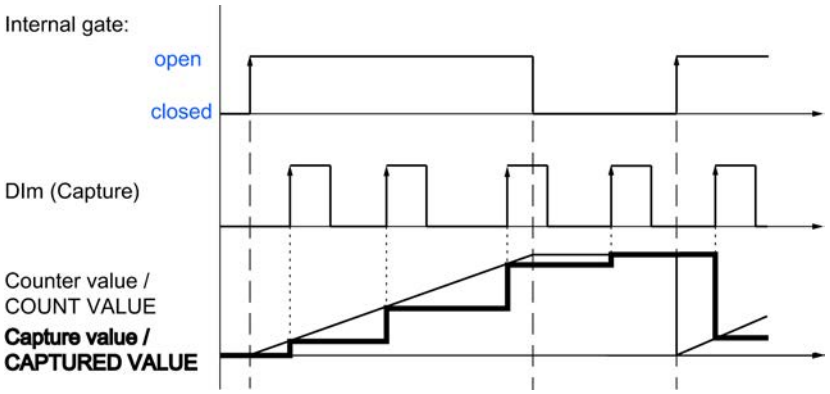
- Rising or falling edge at a digital input
- Rising and falling edge at a digital input
- Rising edge of signal N at the encoder input

Function principle

The Capture value is always the exact counter value at the time of the edge in question (delayed by the configured input filter time). The Capture function is effective regardless of the status of the internal gate. The unchanged counter value is saved when the gate is closed.

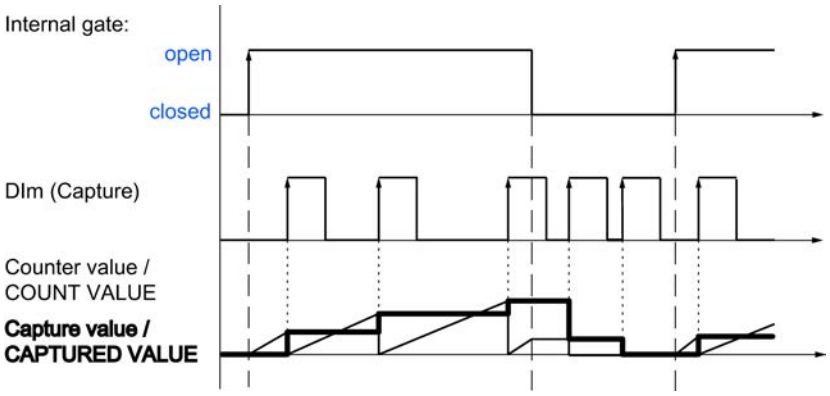
The figure below shows an example of the Capture function with the following configuration:

- Start value = 0
- Capture event upon rising edge at configured digital input
- Set counter to start value at gate start
- Continue counting after Capture event



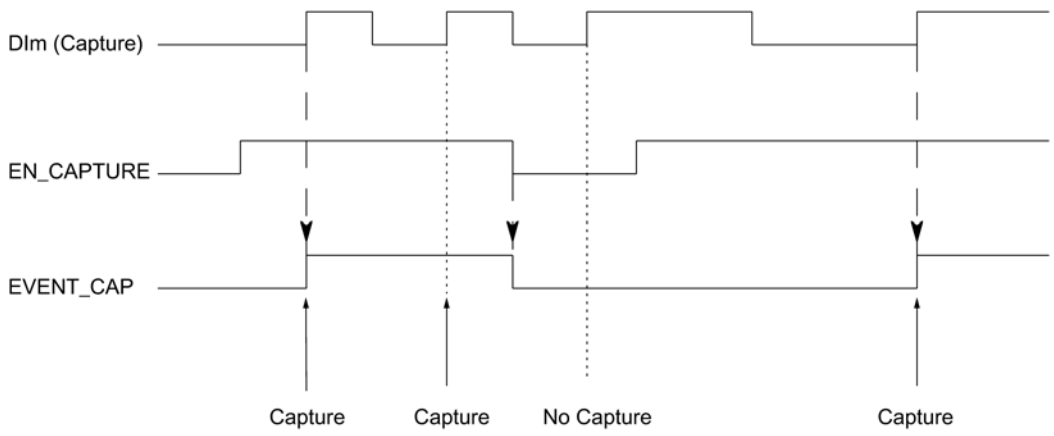
The figure below shows another example of the Capture function with the following configuration:

- Start value = 0
- Capture event upon rising edge at configured digital input
- Set counter to start value at gate start
- Reset counter value to start value after Capture event and continue counting.



The control bit (Page 9074) EN_CAPTURE is used to enable the Capture function. The feedback bit (Page 9077) EVENT_CAP indicates that a counter value has been saved as a Capture in the feedback interface. If you reset EN_CAPTURE, EVENT_CAP is also reset. The status of a digital input is indicated by the respective feedback bit (Page 9077) STS_DIm .

The figure below shows an example of the EN_CAPTURE and EVENT_CAP bits with use of the Capture function by the rising edge at a digital input:



Note

The configured input filters delay the control signal of the corresponding digital input.

The Capture function has no effect on the feedback bit STS_CNT and the LEDs UP and DN.

Hardware interrupt

You can configure a hardware interrupt for the Capture function. Individual hardware interrupts may be lost if the hardware interrupt rate is higher than the system acknowledgement rate. You can signal a lost hardware interrupt with a diagnostic interrupt.

Capture with SSI absolute encoder

Description

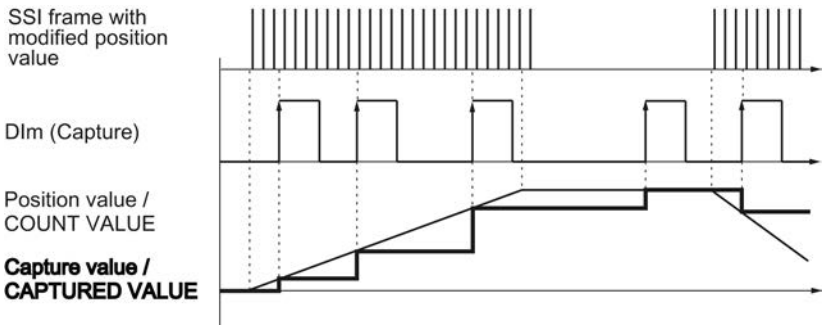
The "Capture" function is used to save the current position value using an external reference signal. You can configure the Capture function for the following reference signals:

- Rising or falling edge at a digital input
- Rising and falling edge at a digital input

Function principle

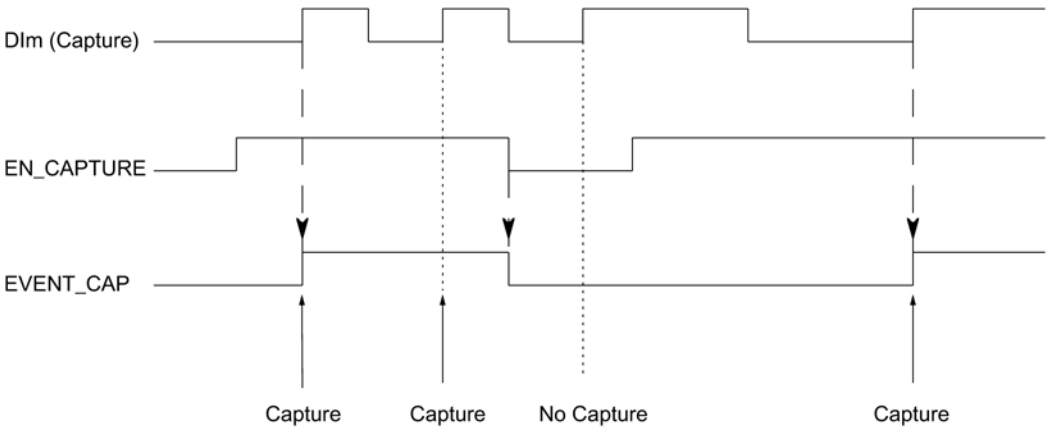
At the time of each edge, the position value of the last valid SSI frame is stored in the Capture value.

The figure below shows an example of the Capture event by a rising edge at the configured digital input:



The control bit (Page 9074) EN_CAPTURE is used to enable the Capture function. The feedback bit (Page 9077) EVENT_CAP indicates that a position value has been saved as Capture value in the feedback interface. If you reset EN_CAPTURE, EVENT_CAP is also reset. The status of a digital input is indicated by the respective feedback bit (Page 9077) STS_DIm .

The figure below shows an example of the EN_CAPTURE and EVENT_CAP bits with use of the Capture function by the rising edge at a digital input:



Note

The configured input filters delay the control signal of the corresponding digital input.

Hardware interrupt

You can configure a hardware interrupt for the Capture function. Individual hardware interrupts may be lost if the hardware interrupt rate is higher than the system acknowledgement rate. You can signal a lost hardware interrupt with a diagnostic interrupt.

Synchronization

Description

You use the "Synchronization" function to set the counter to the pre-defined start value with an external reference signal. You can configure synchronization for the following reference signals:

- Rising or falling edges at a digital input
- Rising edge of signal N at the encoder input
- Rising edge of signal N at the encoder input defined by the level of a digital input

Function principle

Synchronization always takes place exactly at the time of the reference signal. Synchronization is effective regardless of the status of the internal gate.

You use the control bit (Page 9074) EN_SYNC_UP to enable synchronization for counting in an upwards direction. Use the control bit (Page 9074) EN_SYNC_DN to enable synchronization for counting down. The feedback bit (Page 9077) EVENT_SYNC indicates that synchronization has been performed. Resetting EN_SYNC_UP or EN_SYNC_DN also resets EVENT_SYNC.

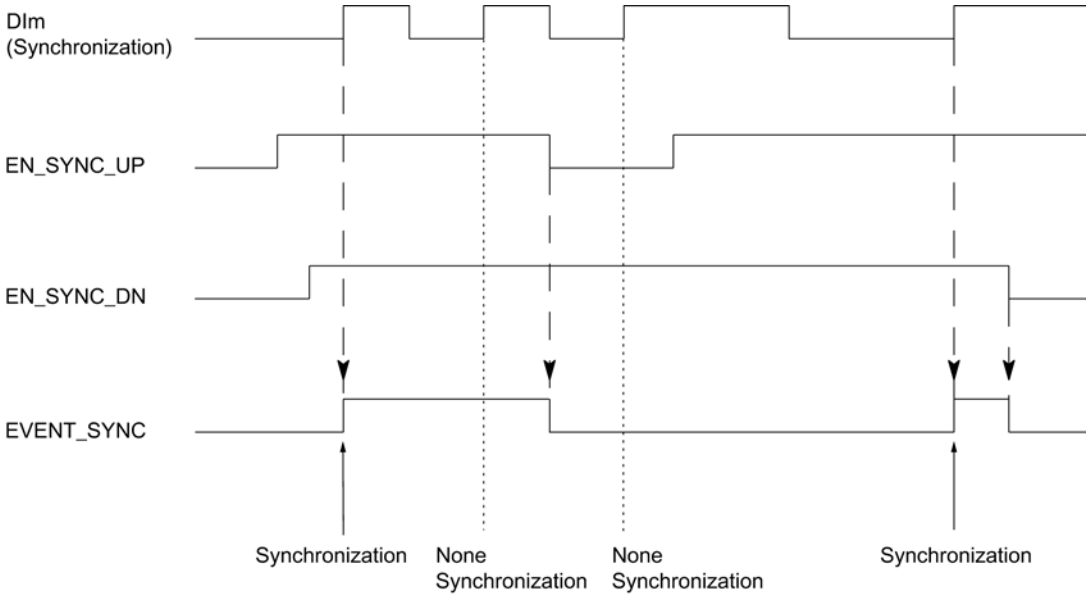
Note

The configured input filters delay the control signal of the corresponding digital input.

Synchronization has no effect on the feedback bit (Page 9077) STS_CNT and the LEDs UP and DN.

Single synchronization

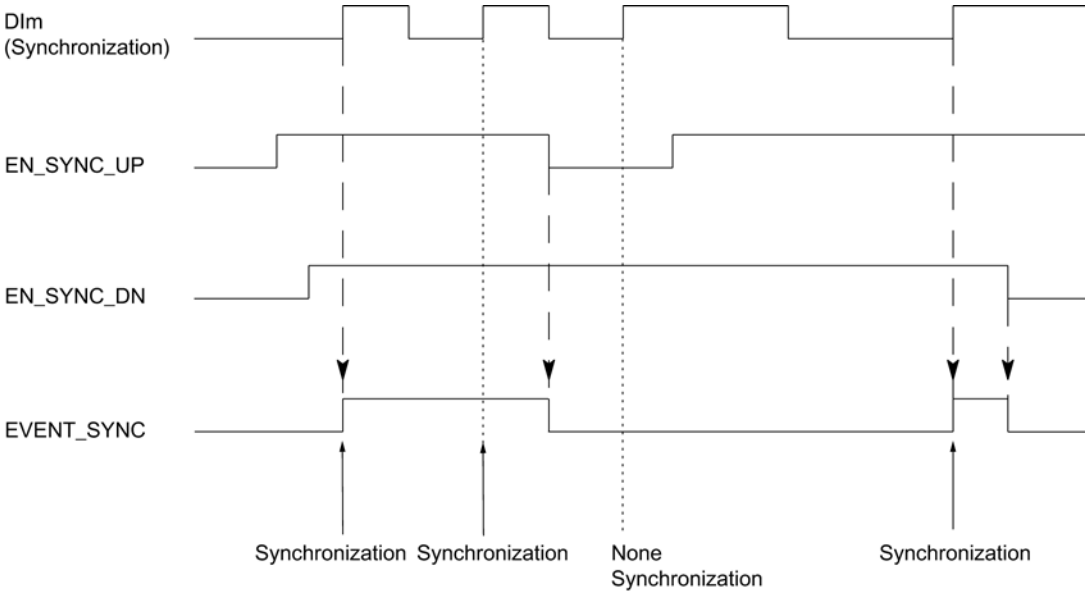
The figure below shows an example of the EN_SYNC_UP, EN_SYNC_DNand EVENT_SYNC bits with single synchronization by an edge at a digital input for count pulses in an upwards direction:



After synchronization is enabled for counting in an upwards direction, the counter is synchronized at the first rising edge at the configured digital input. The counter can only be synchronized again once the control bit (Page 9074) EN_SYNC_UP has been reset and set again.

Periodic synchronization

The figure below shows an example of the EN_SYNC_UP, EN_SYNC_DN and EVENT_SYNC bits with periodic synchronization by an edge at a digital input for count pulses in an upwards direction:



As long as synchronization for counting in an upwards direction is enabled, the counter is synchronized at each rising edge at the configured digital input.

Hardware interrupt

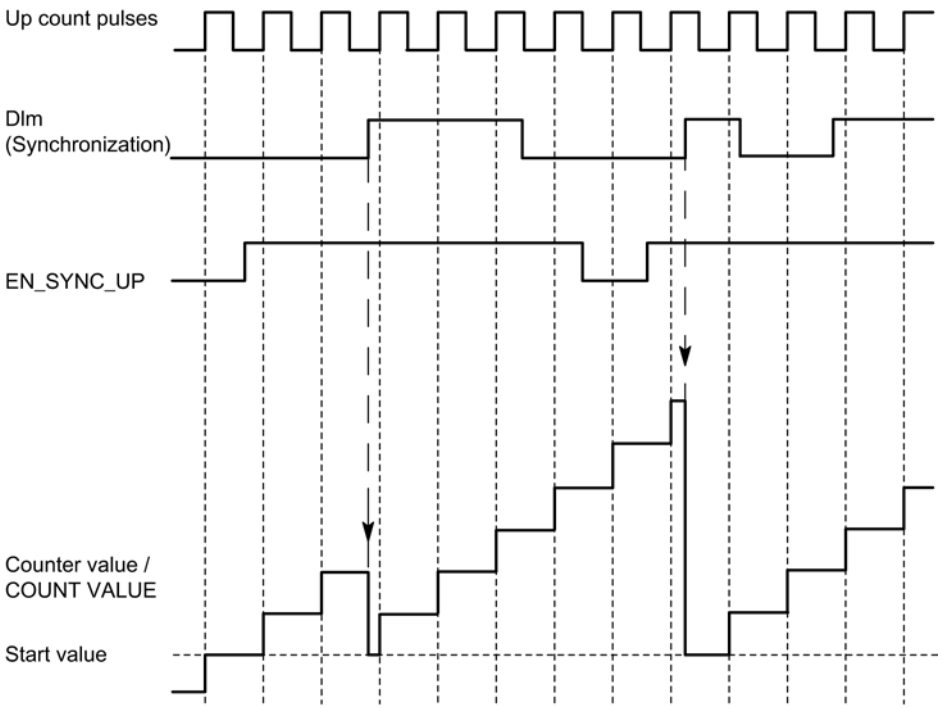
You can configure a hardware interrupt for the synchronization. The hardware interrupts will be lost if the interrupt rate is higher than the system acknowledgement rate. You can signal a lost hardware interrupt with a diagnostic interrupt.

Synchronization by digital input

You can trigger synchronization by edges at a digital input.

Single synchronization

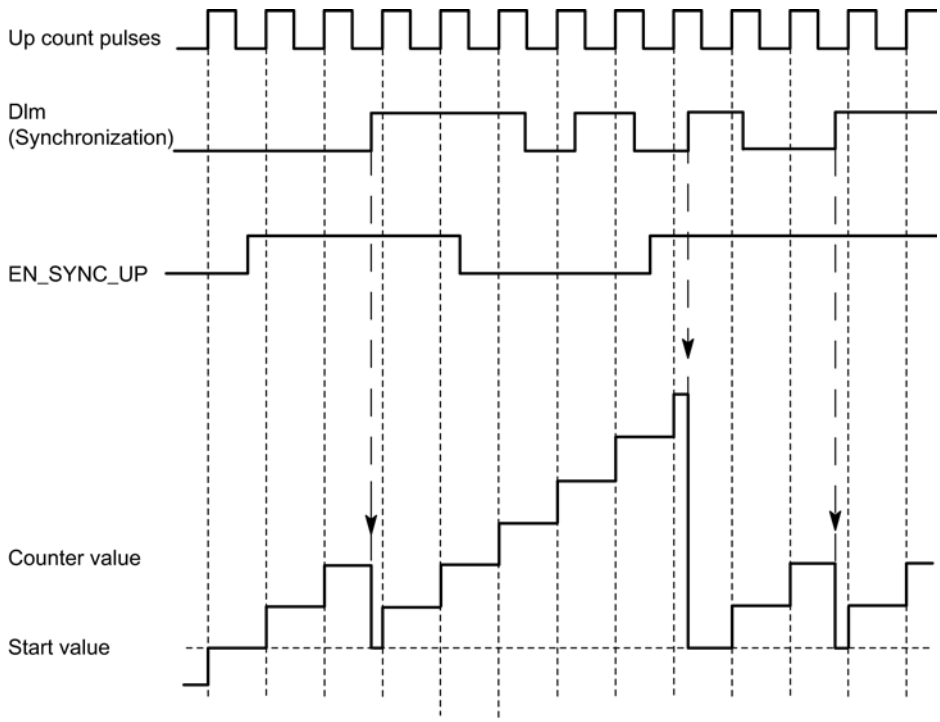
The figure below shows an example for single synchronization by an edge at a digital input:



After synchronization is enabled for counting in an upwards direction, the counter is synchronized at the first rising edge at the configured digital input. Until the control bit (Page 9074) EN_SYNC_UP has been reset and set again, any additional rising edge at the digital output is ignored. The counter can then be synchronized again.

Periodic synchronization

The figure below shows an example for periodic synchronization by an edge at a digital input:



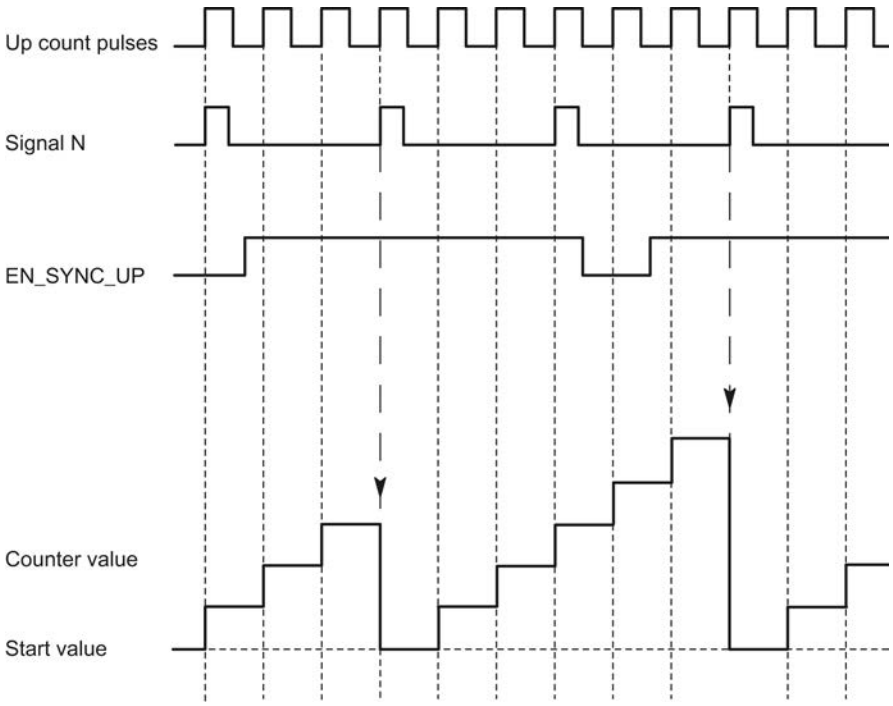
As long as synchronization for counting in an upwards direction is enabled, the counter is synchronized at each rising edge at the configured digital input.

Synchronization at signal N

You can trigger synchronization at signal N at the encoder input either directly or depending on the status of a digital input.

Single synchronization

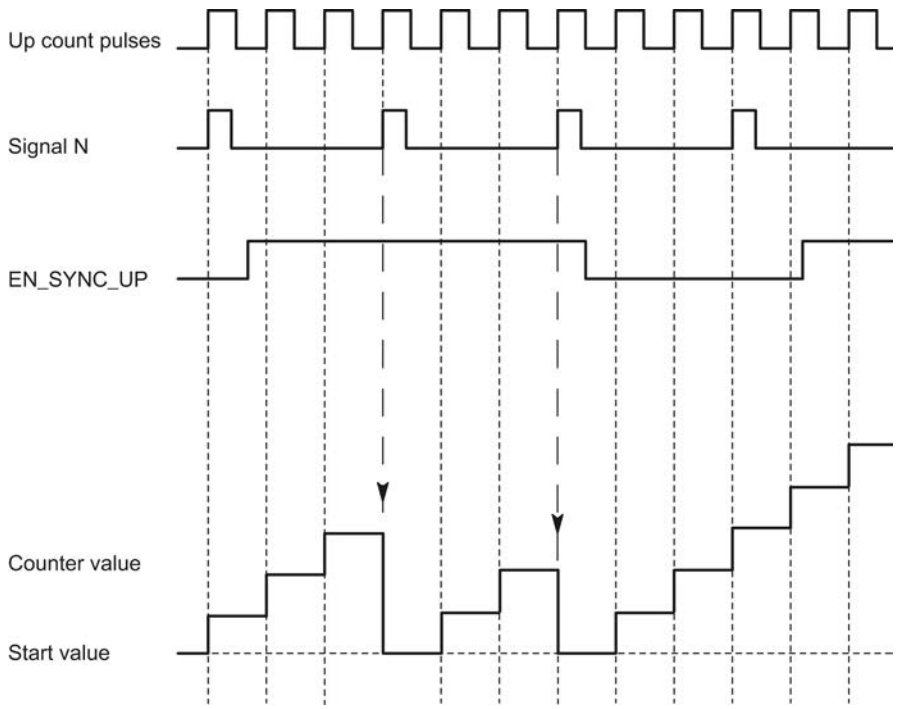
The figure below shows an example of single synchronization at signal N (not dependent on a digital input):



After synchronization is enabled for counting in an upwards direction, the counter is synchronized at the first signal N. After resetting and setting the control bit (Page 9074) EN_SYNC_UP once again, the counter can be synchronized again.

Periodic synchronization

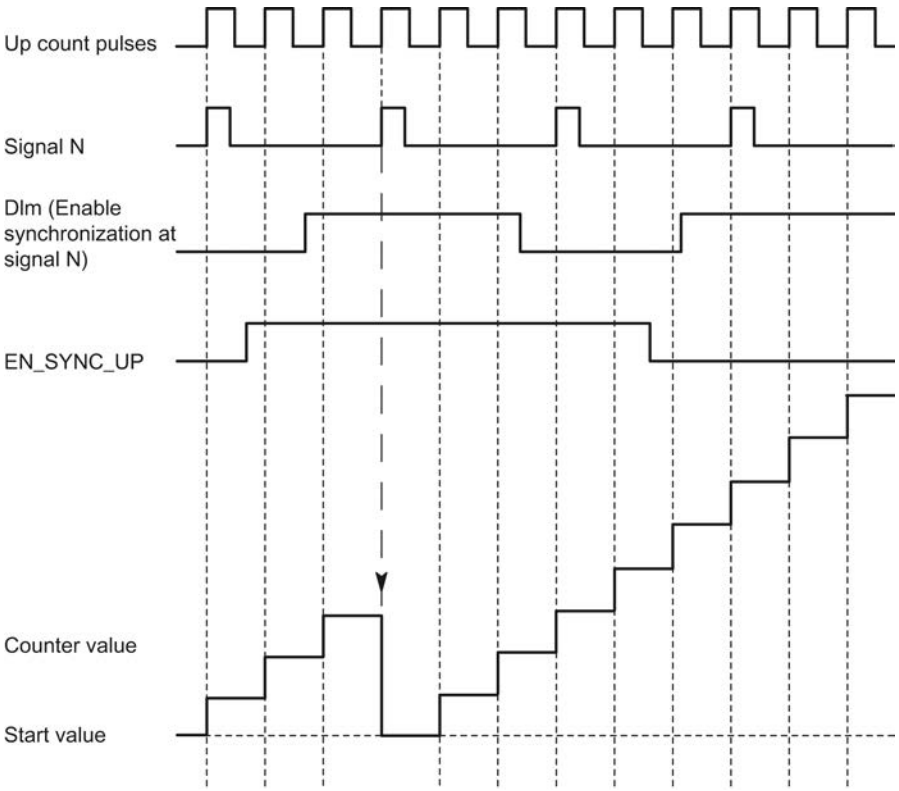
The figure below shows an example for periodic synchronization at signal N:



As long as synchronization for counting in an upwards direction is enabled, the counter is synchronized at each signal N.

Enable by a digital input

The figure below shows an example for periodic synchronization at signal N depending on the status of a digital input:



As long as synchronization for counting up is enabled and the corresponding digital input is active, the counter is synchronized at each signal N. If one of the conditions is not met, the counter is not synchronized at the signal N.

Comparison values

Comparison values and outputs

Description

You can specify two comparison values to control both digital outputs of the channel independently of the user program:

- Comparison value 0 for digital output DQ0
- Comparison value 1 for digital output DQ1

Depending on the operating mode and the encoder used, define two position, counter or measured values as comparison value. Comparison value 1 must be greater than comparison value 0. The comparison values are configurable and can be modified during runtime using the user program.

Switching digital outputs from the user program

The control bits (Page 9074) TM_CTRL_DQ0 and TM_CTRL_DQ1 are used to control use of the digital outputs.

If TM_CTRL_DQm is set to 0, you can control the relevant digital output from the user program with the control bit SET_DQm regardless of the configured technological function. If TM_CTRL_DQm is set to 1, the technological function of the controller of the respective digital output is enabled.

The status of a digital output is indicated by the respective STS_DQm feedback bit.

Switch at comparison values in Counting mode

The comparison values are compared with the current counter or position value. If the counter or position value meets the configured comparison condition and the technological function of the corresponding digital output is enabled, the digital output is set. If you configure "Between comparison value 0 and 1" for digital output DQ1, both comparison values affect DQ1.

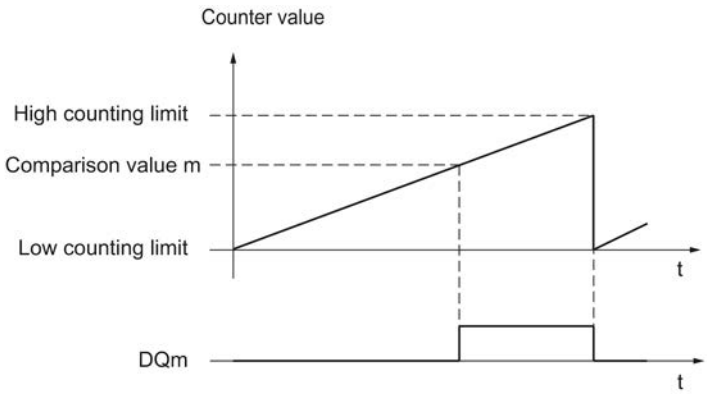
Switch at comparison values with incremental or pulse encoder

You can make switching for a digital output dependent on one of the following comparison events:

Setting between comparison value and high counter limit

The digital output is set to 1 if:

Comparison value <= counter value <= high counter limit

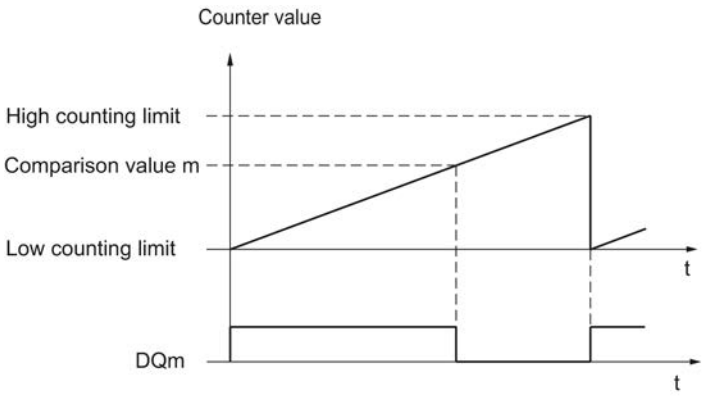


The comparison event is independent of the count direction.

Setting between comparison value and low counter limit

The digital output is set to 1 if:

Low counter limit <= counter value <= comparison value



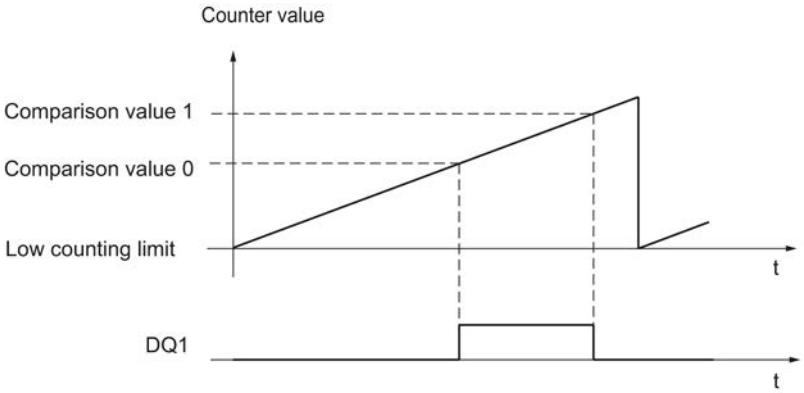
The comparison event is independent of the count direction.

Setting between comparison value 0 and comparison value 1

The comparison event can be configured for the digital output DQ1 if "Use by user program" has been configured for the digital output DQ0.

DQ1 is set to 1 if:

Comparison value 0 <= counter value <= comparison value 1



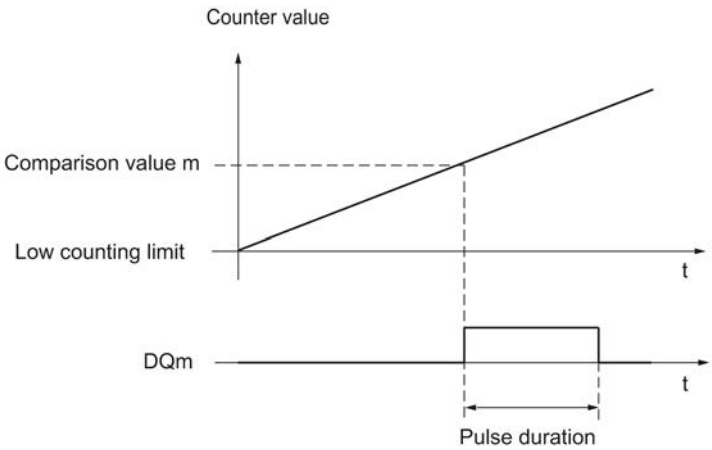
The comparison event is independent of the count direction.

Setting at comparison value for one pulse duration

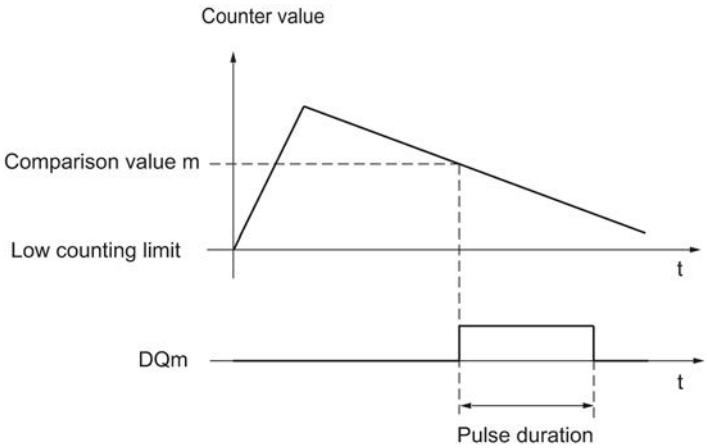
The respective digital output is set to 1 for a specified period of time when the following conditions are fulfilled:

- Counter value = comparison value
- Current count direction = configured count direction for the comparison event

The figure below shows an example of the comparison event when counting in an upwards direction:

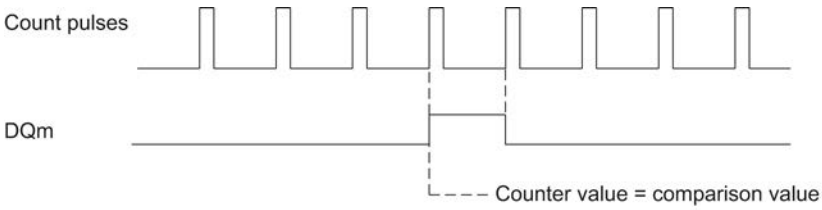


The figure below shows an example of the comparison event when counting in a downward direction:



To repeat the comparison event, the counter value must change and then correspond to the respective comparison value again.

If the pulse duration has been defined as "0" and the counter value is equal to the comparison value, the digital output is set to 1 until the next count pulse:



Note

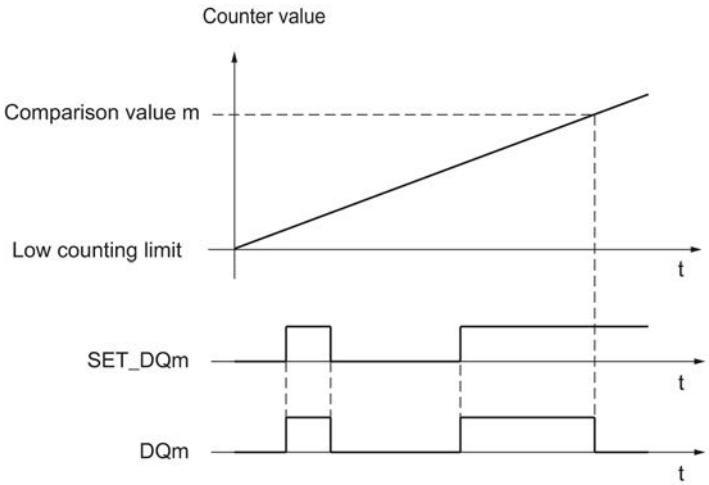
This comparison event switches the relevant digital output if a count pulse reaches the comparison value. The digital output does not switch when the counter value is set, by synchronization for example.

Setting by the user program up to comparison value

You can set the respective digital output to 1 (edge) by setting the control bit (Page 9074) SET_DQm. The respective digital output is set to 0 by any of the following events:

- Match of the counter value and the comparison value in the configured direction of the comparison event
- Reset of the corresponding SET_DQm control bit.

The figure below shows an example of the comparison event when counting in an upwards direction:



You can disable the digital output before the counter value reaches the comparison value by setting the control bit SET_DQm to 0.

Note

If the comparison value is reached in the configured counting direction, the feedback bit EVENT_CMPm is set independently of the state of the control bit SET_DQm.

The comparison event switches a digital output when a count pulse reaches the respective comparison value. The digital output does not switch when the counter value is set, by synchronization for example.

Switch at comparison values with SSI absolute encoder

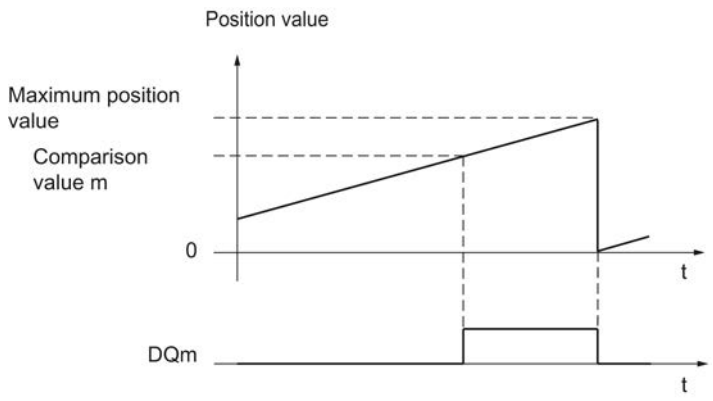
You can make switching for a digital output dependent on one of the following comparison events:

Setting between comparison value and high limit

The high limit corresponds to the maximum position value.

The digital output is set to 1 if:

Comparison value <= position value <= maximum position value



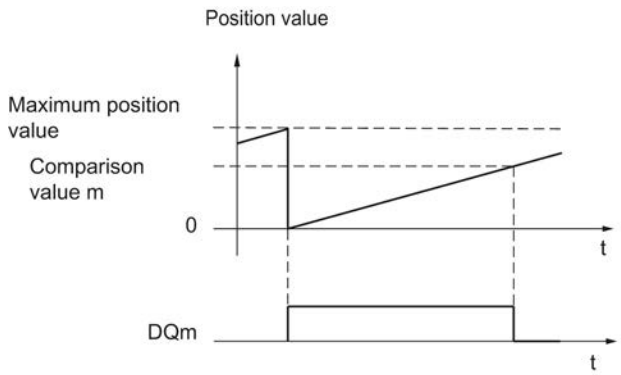
The comparison event is independent of the direction of the position value change. The maximum position value depends on the resolution of the SSI absolute encoder.

Setting between comparison value and low limit

The low limit corresponds to the position value "0".

The digital output is set to 1 if:

0 <= position value <= comparison value



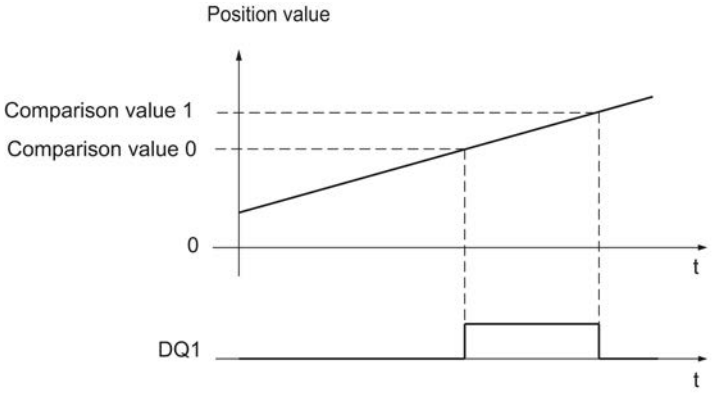
The comparison event is independent of the direction of the position value change.

Setting between comparison value 0 and comparison value 1

The comparison event can be configured for the digital output DQ1 if "Use by user program" has been configured for the digital output DQ0.

DQ1 is set to 1 if:

Comparison value 0 <= position value <= comparison value 1



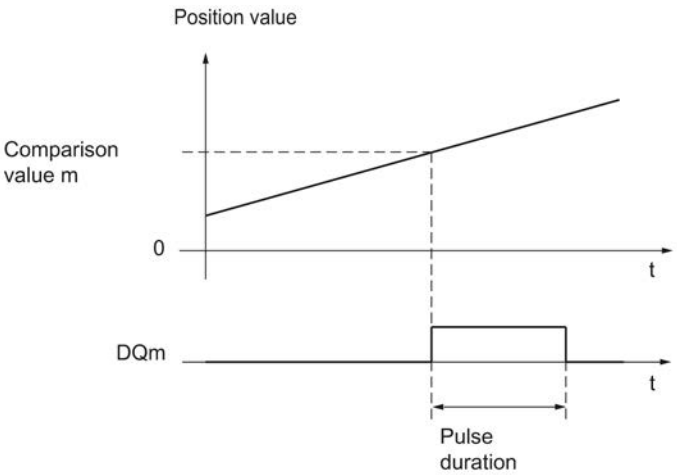
The comparison event is independent of the direction of the position value change.

Setting at comparison value for one pulse duration

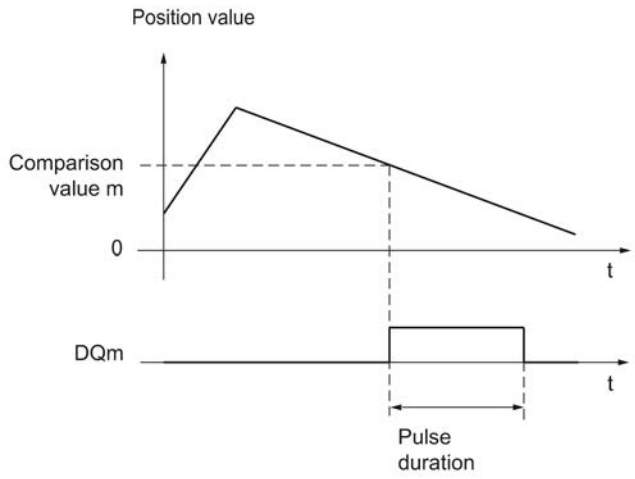
The respective digital output is set to 1 for a specified period of time when the following conditions are fulfilled:

- Matching of the position value and comparison value or crossing of the comparison value
- Current direction of the position value change = assigned direction for the comparison event

The figure below shows an example of the comparison event when counting up:



The figure below shows an example of the comparison event when counting down:



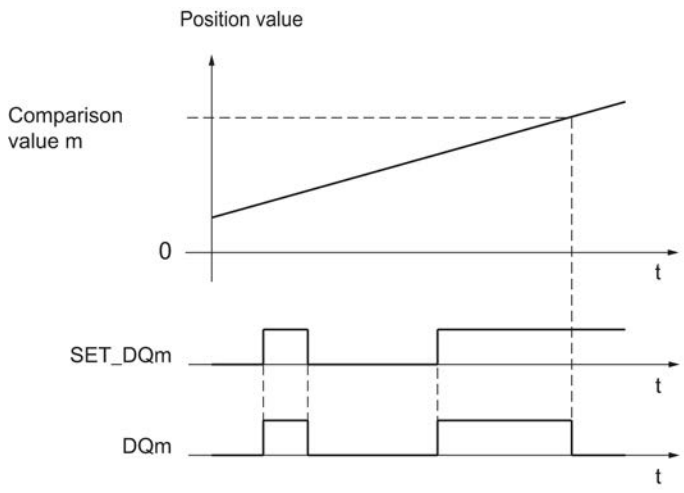
To repeat the comparison event, the position value must change and then correspond to or cross the respective comparison value again.

Setting by the user program up to comparison value

You can set each digital output to 1 (edge) by setting the control bit (Page 9074) SET_DQm. The respective digital output is set to 0 by any of the following events:

- Matching of the position value and the comparison value or crossing of the comparison value in the configured direction of the comparison event
- Resetting of the SET_DQm control bit.

The figure below shows an example of the comparison event when counting up:



You can disable the digital output before the position value corresponds to or exceeds the comparison value by setting the control bit SET_DQm to 0.

Note

If the comparison value is reached or exceeded in the assigned direction, feedback bit EVENT_CMPm is set independently of the status of control bit SET_DQm.

Switch at comparison values in Measuring mode

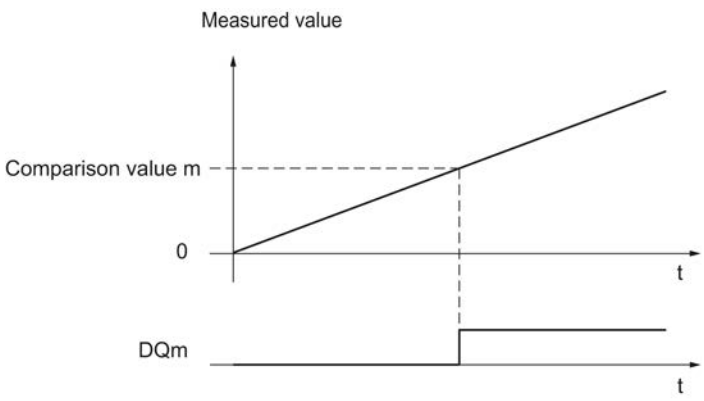
The comparison values are compared with the current measured value. If the measured value meets the configured comparison condition and the technological function of the corresponding digital output is enabled, the digital output is set. If you configure "Between comparison value 0 and 1" or "Not between comparison value 0 and 1" for digital output DQ1, both comparison values affect DQ1.

You can make switching for a digital output dependent on one of the following comparison events:

Setting above the comparison value

The digital output is set to 1 if:

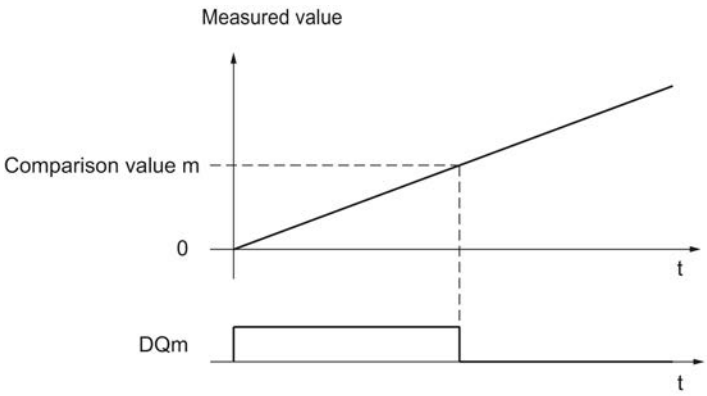
Measured value >= comparison value



Setting below the comparison value

The digital output is set to 1 if:

Measured value <= comparison value

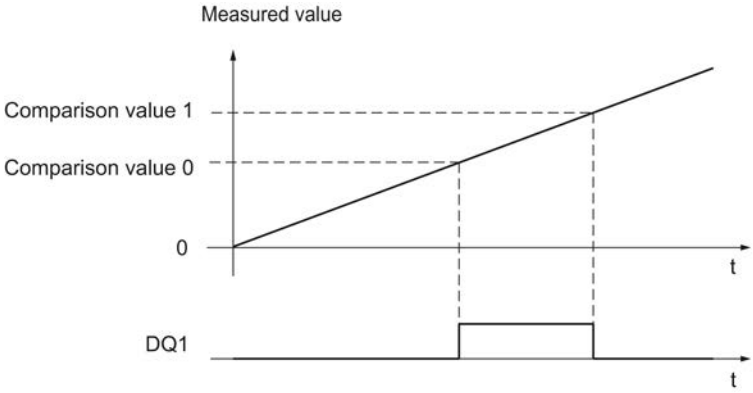


Setting between comparison value 0 and comparison value 1

The comparison event can be configured for the digital output DQ1 if "Use by user program" has been configured for the digital output DQ0.

DQ1 is set to 1 if:

Comparison value 0 <= measured value <= comparison value 1

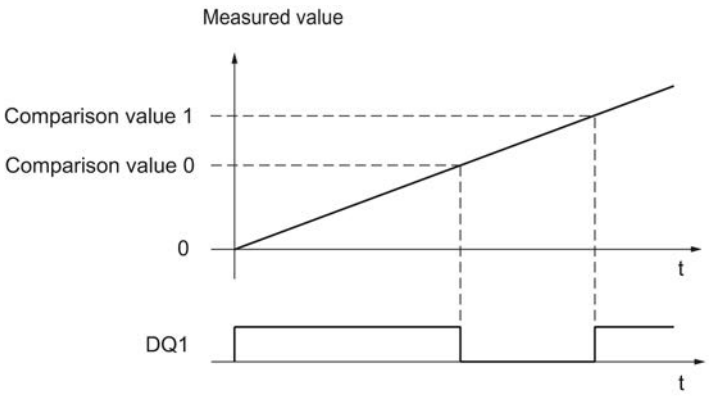


Not setting between comparison value 0 and comparison value 1

The comparison event can be configured for the digital output DQ1 if "Use by user program" has been configured for the digital output DQ0.

DQ1 is set to 1 if:

Comparison value 1 <= measured value <= comparison value 0



Measured value determination

Overview of measuring functions

The following measuring functions are available:

Measurement type (Page 9010)	Description
Frequency measurement	The mean frequency is calculated at set measuring intervals on the basis of the time profile of the count pulses or position value changes and returned in Hertz as floating point number.
Period measurement	The mean period duration is calculated at set measuring intervals on the basis of the time profile of the count pulses or position value changes and returned in seconds as floating point number.
Velocity measurement	The mean velocity is calculated at set measuring intervals on the basis of the time profile of the count pulses or position value changes and other parameters, and returned in the configured unit of measurement.

Measured values and counter values are available concurrently in the feedback interface.

Update time

You can configure the interval at which the technology module updates the measured values cyclically as the update time. Setting longer update time intervals allows uneven measured variables to be smoothed and increases measuring accuracy.

Gate control for incremental and pulse encoders

Opening and closing the internal gate defines the period of time during which the count pulses are captured. The update time is asynchronous to the opening of the gate, which means that the update time is not started when the gate is opened. After the internal gate is closed, the last measured value captured is still returned.

Measured value determination with incremental or pulse encoder

Measuring ranges

Measuring ranges

The measuring functions have the following measuring limits:

Measurement type	Low measuring range limit	High measuring range limit
Frequency measurement	0,04 Hz	800 kHz* / 4 MHz**
Period measurement	1,25 µs* / 0,25 µs**	25 s
Velocity measurement	Depending on the configured number of "increments per unit" and the "time base for velocity measurement"	

* Applies to 24 V incremental encoders and "quadruple" signal evaluation.

** Applies to RS422 incremental encoders and "quadruple" signal evaluation.

All measured values are returned as signed values. The sign indicates whether the counter value increased or decreased during the relevant time period.

Measuring intervals

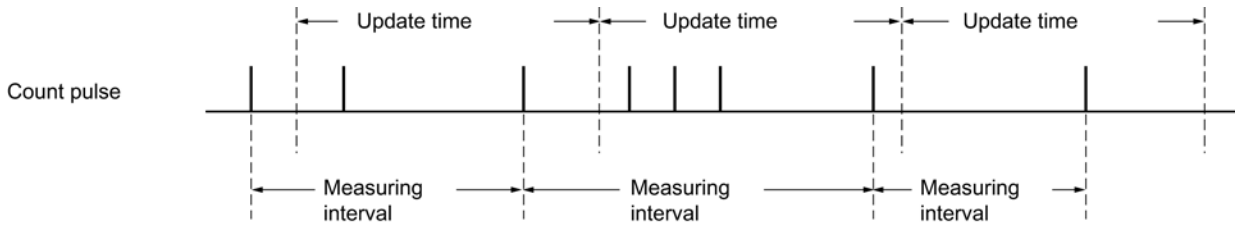
Measuring principle

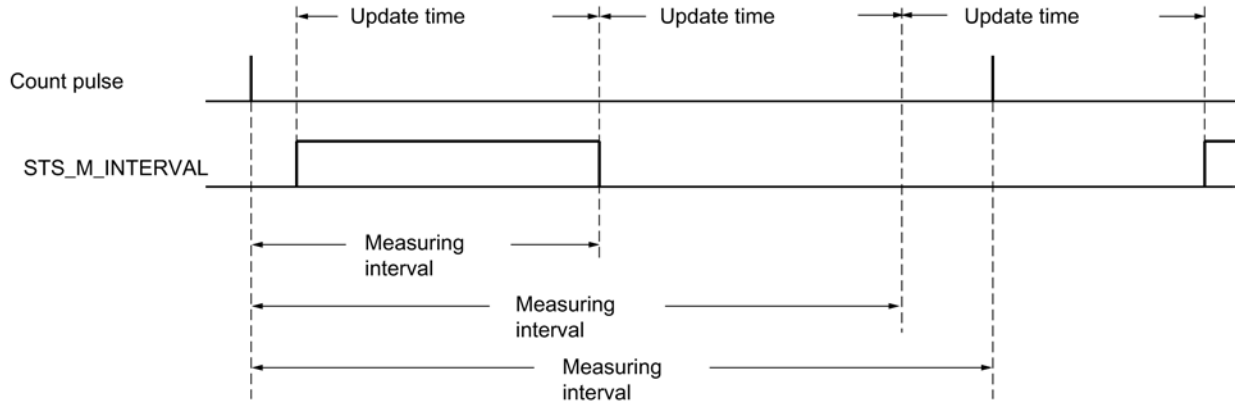
The technology module assigns a time value to each count pulse. The measuring interval is defined as the time between each last count pulse before and during the previous update time. The measuring interval and the number of pulses in the measuring interval are evaluated to calculate measured variables.

If there is no count pulse within an update time, the measuring interval is dynamically adjusted. In this case, a pulse is assumed at the end of the update time and the measuring interval is calculated as the time between that point and the last pulse which occurred. The number of pulses is then 1.

The feedback bit STS_M_INTERVAL indicates whether a count pulse occurred in the previous measuring interval. This allows for a differentiation between an assumed and an actual count pulse.

The following figures show the principle of measurement and the dynamic adjustment of the measuring interval:





Measuring types

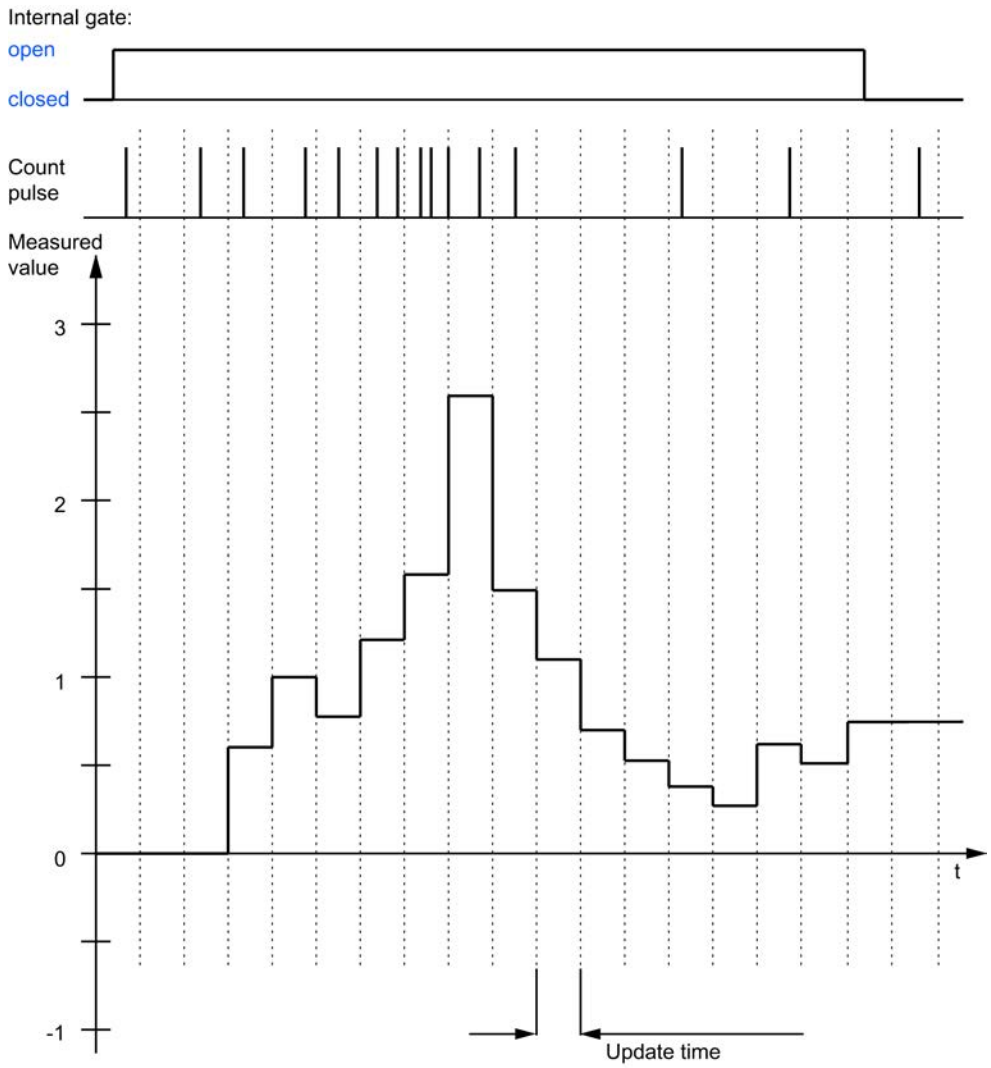
Frequency measurement

A value "0" is returned until the first measured value is available.

The measurement process begins with the first pulse detected once the internal gate has been opened. The first measured value can be calculated after the second pulse at the earliest.

The measured value is updated in the feedback interface (Page 9077) upon completion of each update time. If the internal gate is closed, measuring stops and the measured value is no longer updated.

The figures below shows an example of frequency measurement with an update time of 1 s:



Period measurement

The reciprocal of the frequency is output as the measured value for period measurement.

A value "25 s" is returned until the first measured value is available.

Velocity measurement

The normalized frequency is output as the measured value in velocity measurement. You can configure the scaling using the time basis and the number of increments that your encoder delivers per unit.

Example:

Your encoder delivers 4000 increments per meter. The velocity is to be measured in meters per minute.

In this case, you need to configure 4000 Increments per unit and a time basis of one minute.

Measured value determination with SSI absolute encoder

Measuring ranges

Measuring range SSI absolute encoder

The measuring functions have the following measuring limits:

Measurement type	Low measuring range limit	High measuring range limit
Frequency measurement	0,04 Hz	4 MHz
Period measurement	0,25 µs	25 s
Velocity measurement	Depending on the configured number of "increments per unit" and the "time base for velocity measurement"	

All measured values are returned as signed values. The sign indicates whether the position value increased or decreased during the relevant time period.

Measuring intervals

Measuring principle

The technology module assigns a time value to each SSI frame. The measuring interval is defined as the time between the last SSI frame with a change of position value before and during the previous update time. The measuring interval and the total change in position value in the measuring interval are evaluated to calculate a measured variable. The total change in position value in a measuring interval corresponds to the number of encoder increments in the same measuring interval.

If there is no change in position value within an update time, the measuring interval is dynamically adjusted. In this case, a change in position value is assumed at the end of the update time and the measuring interval is calculated as the time between that point and the last SSI frame with a change in position value. The change in position value is then 1.

The feedback bit STS_M_INTERVAL indicates whether a change in position value occurred in the previous measuring interval. This allows for a differentiation between an assumed and an actual change in position value. If the technology module cannot calculate measured values because the measuring range limit has been violated, the feedback bit STS_M_INTERVAL is not set.

Measuring types

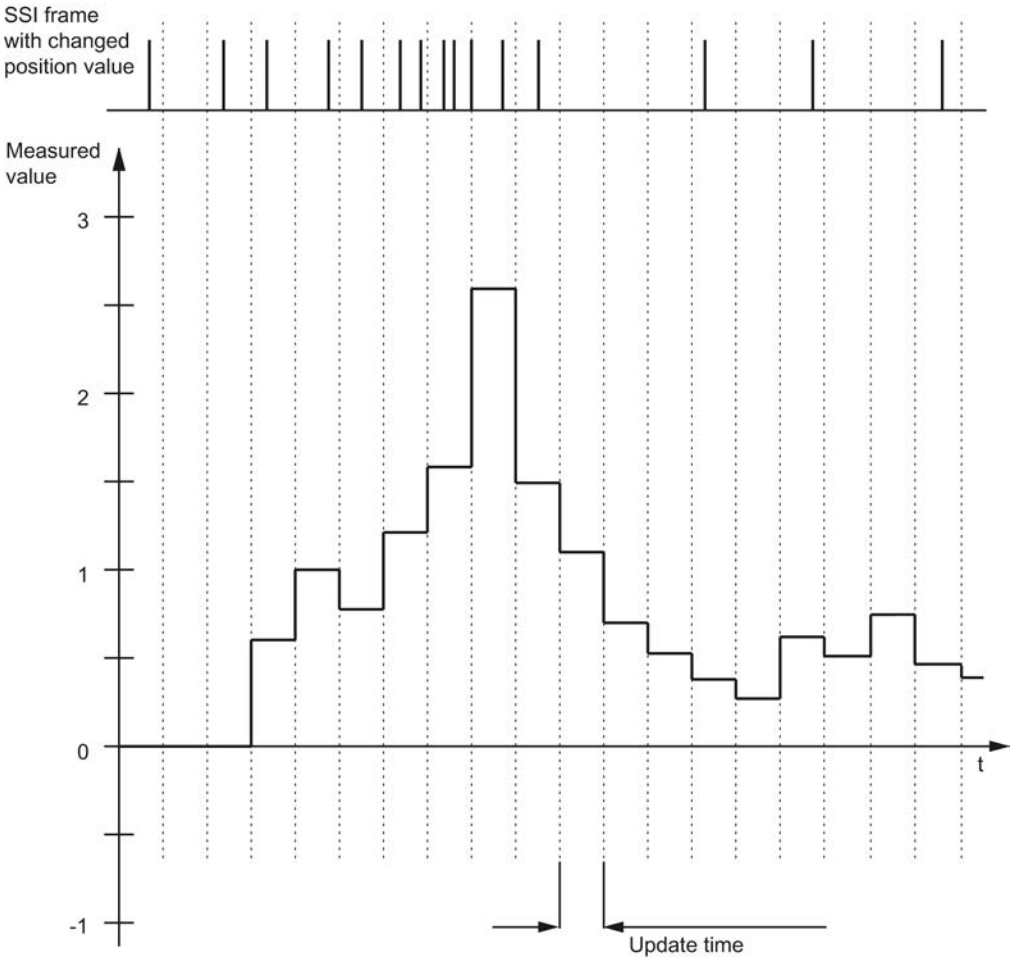
Frequency measurement

The value "0.0" is reported in the time up to the first available measured value.

The measuring process begins with the first detected change in position value. The first measured value can be calculated after the second detected change in position value at the earliest.

The measured value is updated in the feedback interface (Page 9077) upon completion of each update time.

The figures below shows an example of frequency measurement with an update time of 1 s:



Period measurement

The reciprocal of the frequency is output as the measured value for period measurement.

A value "25 s" is returned until the first measured value is available.

Velocity measurement

The normalized frequency is output as the measured value in velocity measurement. You can configure the scaling using the time basis and the number of increments that your encoder delivers per unit.

Example:

Your SSI absolute encoder operates with a resolution of 12 bits per revolution and performs 4096 increments per revolution. The velocity should be measured in revolutions per minute.

In this case, you need to configure 4096 Increments per unit and a time basis of one minute.

Hysteresis

Hysteresis with incremental or pulse encoder

Description

Hysteresis allows you to specify a range around the comparison values within which the digital outputs are not to be switched again until the counter value has gone outside this range.

Slight movements by the encoder can result in the counter value fluctuating around a certain value. If a comparison value or a counting limit lies within this fluctuation range, the corresponding digital output is switched on and off with corresponding frequency if hysteresis is not used. Hysteresis prevents this unwanted switching, and configured hardware interrupts when a compare event occurs.

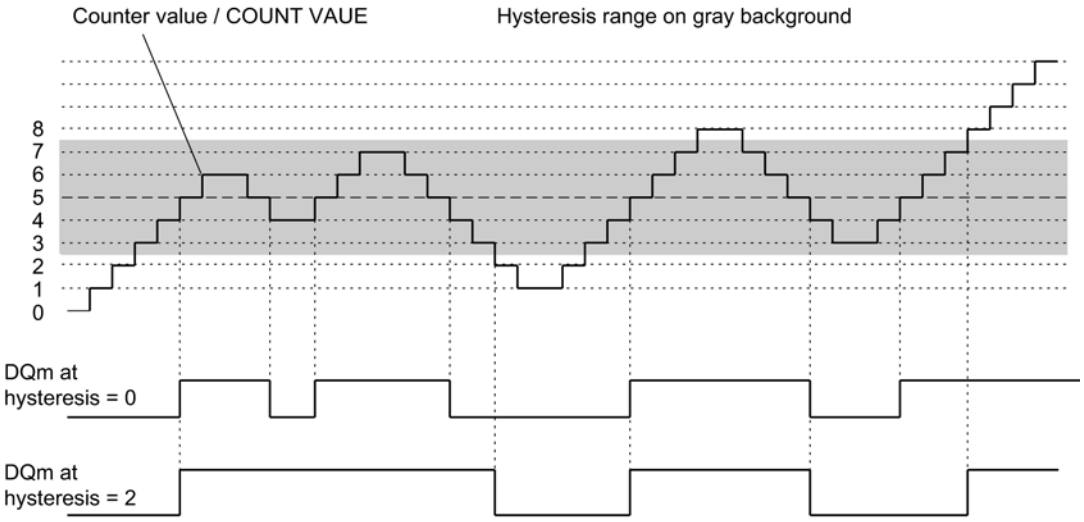
The Hysteresis becomes active when the respective comparison value is reached by a count pulse. If the counter value is set to the start value during an active Hysteresis, the Hysteresis becomes inactive.

Regardless of the hysteresis value, the hysteresis range ends at the low/high counting limits.

Function principle

The figure below shows an example for the hysteresis with the following configuration:

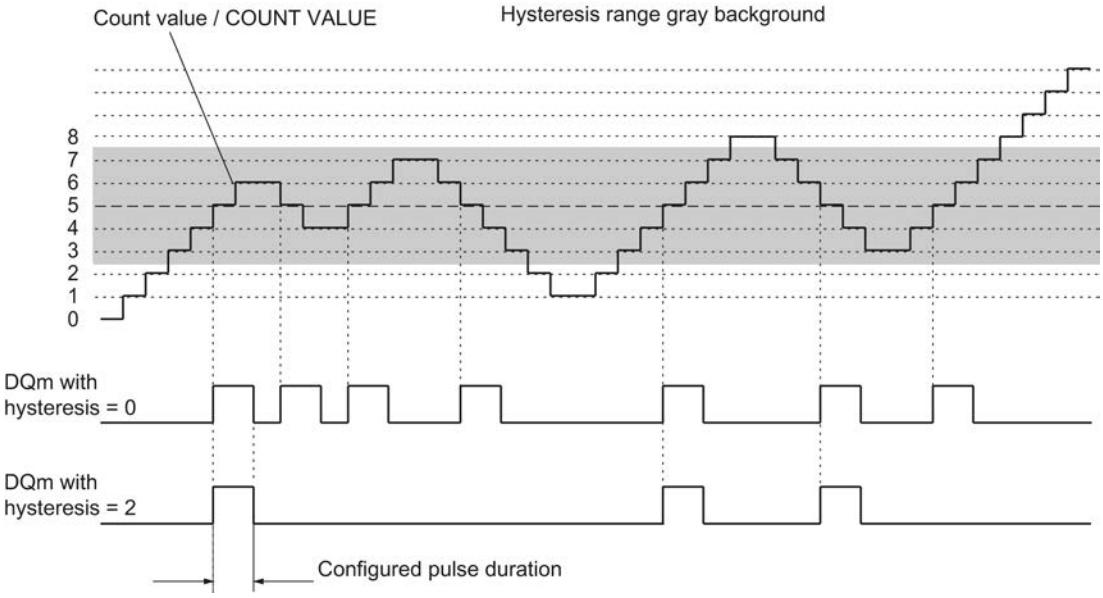
- Setting of a digital output between comparison value and high counting limit
- Comparison value = 5
- Hysteresis = 0 or 2



Hysteresis is enabled when the counter value 5 is reached. When the hysteresis is active, the comparison result remains unchanged. Hysteresis is disabled when the counter values 2 or 8 are reached.

The figure below shows an example for the hysteresis with the following configuration:

- Setting at comparison value for one pulse duration
- Comparison value = 5
- Comparison in both count directions
- Hysteresis = 0 or 2



Hysteresis with SSI absolute encoder

Description

Hysteresis allows you to specify a range around the comparison values within which the digital outputs are not to be switched again until the position value has gone outside this range.

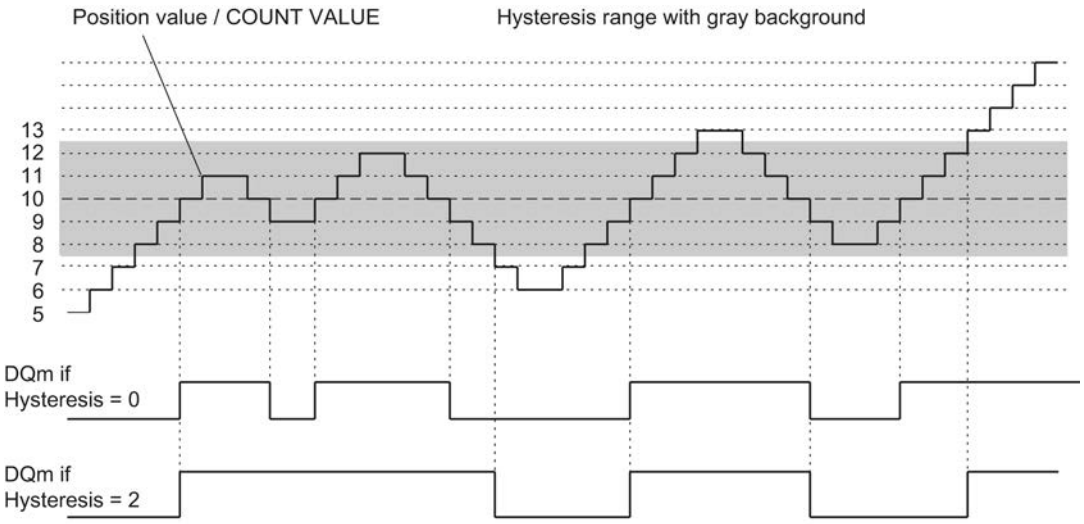
Slight movements by the encoder can result in the position value fluctuating around a certain value. If a comparison value, "0", or the respective maximum position value lies within this fluctuation range, the associated digital output is switched on and off with corresponding frequency if a hysteresis is not used. Hysteresis prevents this unwanted switching, and configured hardware interrupts when a compare event occurs.

Regardless of the hysteresis value, the hysteresis range ends at "0" and at the respective maximum position value.

Function principle

The figure below shows an example for the hysteresis with the following parameter assignment:

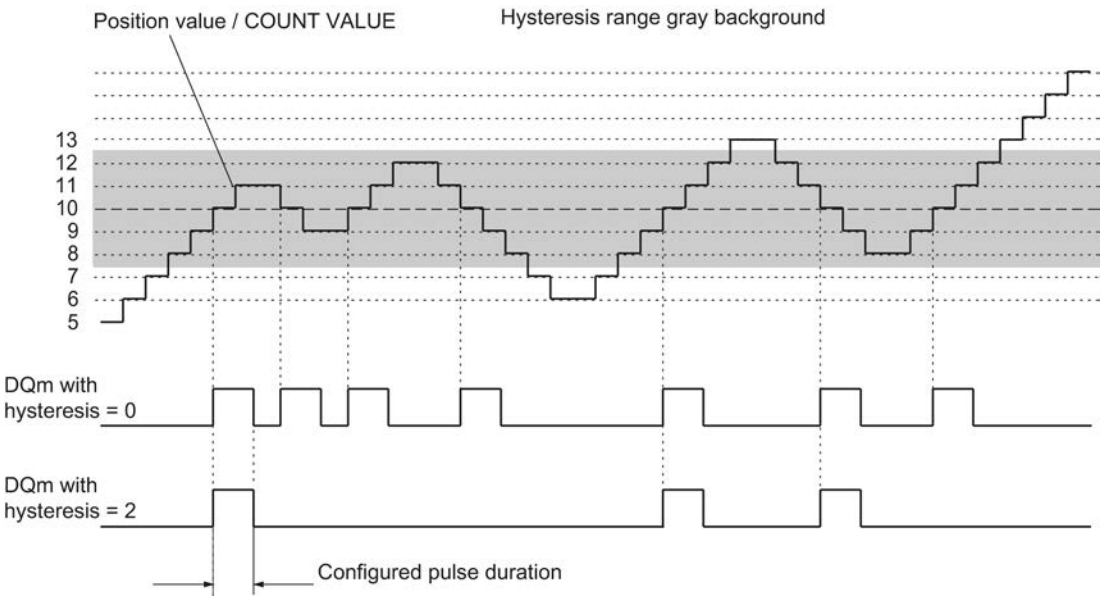
- Setting of a digital output between comparison value and high limit
- Comparison value = 10
- Hysteresis = 0 or 2



Hysteresis is enabled when the position value 10 is reached. When the hysteresis is active, the comparison result remains unchanged. Hysteresis is disabled when the position values 7 or 13 are reached.

The figure below shows an example for the hysteresis with the following parameter assignment:

- Setting at comparison value for one pulse duration
- Comparison value = 10
- Comparison in both directions of position value changes
- Hysteresis = 0 or 2



Interrupts

Hardware interrupt

The technology module can trigger a hardware interrupt in the CPU if, for example, a comparison event occurs; in the event of overflow or underflow; in the event of a zero crossing of the counter and/or of a change in count direction (direction reversal). You can specify which events are to trigger a hardware interrupt during operation.

Diagnostic interrupt

The technology module can trigger diagnostic interrupts in the event of errors. You enable the diagnostic interrupts for certain errors in the device configuration. Refer to the device manual for the technology module to learn about the events that can trigger a diagnostic interrupt during operation.

Position detection for Motion Control

Description

You can use the technology module with S7-1500 Motion Control for position detection.

In the Device configuration of the technology module in STEP 7 (TIA Portal), select "Position input for Motion Control" mode. This reduces the configuration options to the parameters that are essential. The mode automatically applies to all channels of the technology module.

When using an incremental or pulse encoder, the position input is based on the counting function of the technology module. With an SSI absolute encoder, the absolute value is transferred via a synchronous, serial interface and prepared according to the parameter assignment to be made available for S7-1500 Motion Control.

Encoder signals

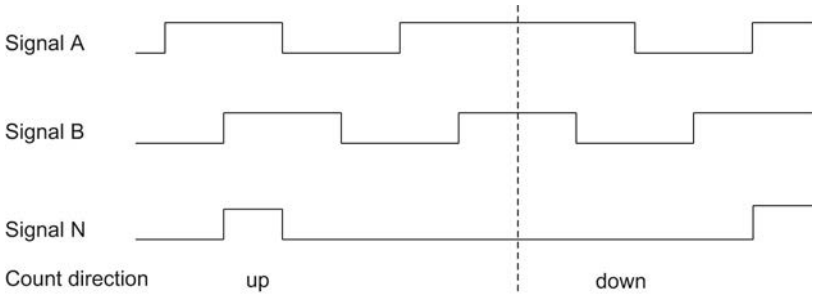
24 V and TTL count signals

24 V and TTL incremental encoder count signals

The 24 V incremental encoder returns the 24 V signals A, B, and N to the technology module. The A and B signals are phase-shifted by 90°. You can also connect incremental encoders without an N signal.

A 24 V incremental encoder uses the A and B signals for counting. If configured accordingly, the N signal is used for setting the counter to the start value or for saving the current counter value to the Capture value.

The figure below shows an example of the time profile of the signals of an 24 V incremental encoder:



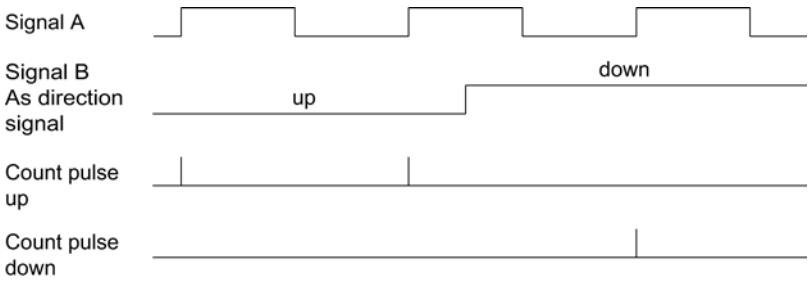
The technology module detects the count direction by evaluating the sequence of edges of the A and B signals. You can specify an inversion of the count direction.

24 V and TTL pulse encoder count signals without/with direction signal

The encoder, for example an initiator (BERO) or a light barrier, returns only a count signal that is connected to terminal A of the counter.

In addition, you can connect a signal for direction detection to terminal B of the counter. If your encoder does not return a corresponding signal, you can specify the count direction with the user program using the control interface.

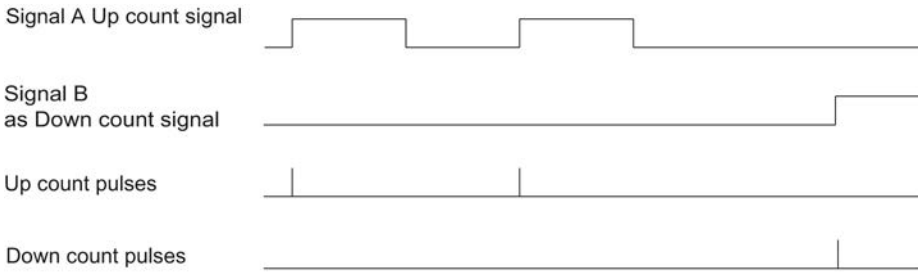
The figure below shows an example of the time profile of the signals of a 24 V pulse encoder with direction signal and the resulting count pulses:



24 V and TTL pulse encoder count signals with Up/Down count signal

The Up count signal is connected to terminal A. The Down count signal is connected to terminal B.

The figure below shows an example of the time profile of the signals of a pulse encoder with Up/Down count signal and the resulting count pulses:



Sourcing output/sinking output for 24 V count signals

You can connect the following encoders/sensors to the counter inputs:

- Sourcing output:
  The A, B, and N inputs are wired to 24VDC .
- Sinking output:
  The A, B, and N inputs are wired to ground M .
- Push-pull (sourcing and sinking output):
  The A, B, and N inputs are wired alternately to 24VDC and ground M .

Monitoring of encoder signals

The signals of push-pull 24 V sensors are monitored for wire breaks by the technology module. The technology module monitors TTL signals for offset voltages.

If you enable the diagnostic interrupt in the device configuration, the technology module triggers a diagnostic interrupt in the event of encoder signal errors.

RS422 count signals

RS422 incremental encoder count signals

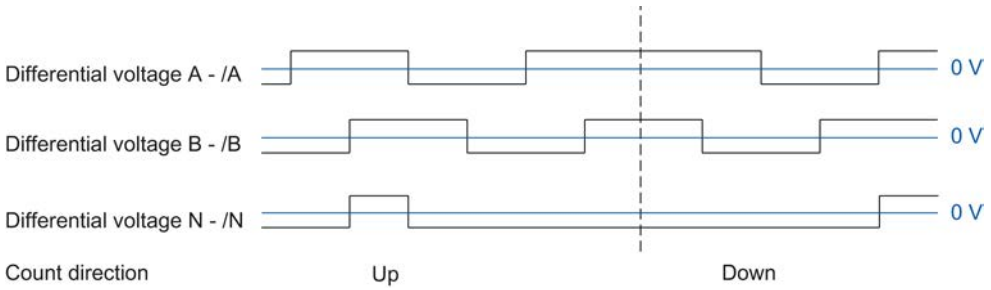
The RS422 incremental encoder sends the following differential signals to the technology module:

- A and /A
- B and /B
- N and /N

The signal information for RS422 signals is encoded in the differential voltage between A and /A, B and /B or N and /N. The A and B signals are phase-shifted by 90°. You can also connect incremental encoders without an N signal.

RS422 incremental encoders use the A and B signals for counting. If configured accordingly, the N signal is used for setting the counter to the start value or for saving the current counter value as the Capture value.

The figure below shows an example of the time profile of the signals of an RS422 incremental encoder:



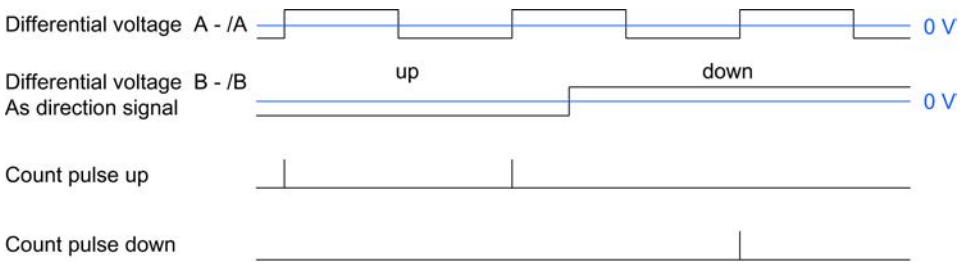
The technology module detects the count direction by evaluating the sequence of edges of the A and B signals. You can specify an inversion of the count direction.

Count signals of RS422 pulse encoders without/with direction signal

The encoder, for example a light barrier, only returns a count signal that is connected to terminal A.

You can also connect a signal for direction detection to terminal B. If your encoder does not return a corresponding signal, you can specify the count direction with the user program using the control interface.

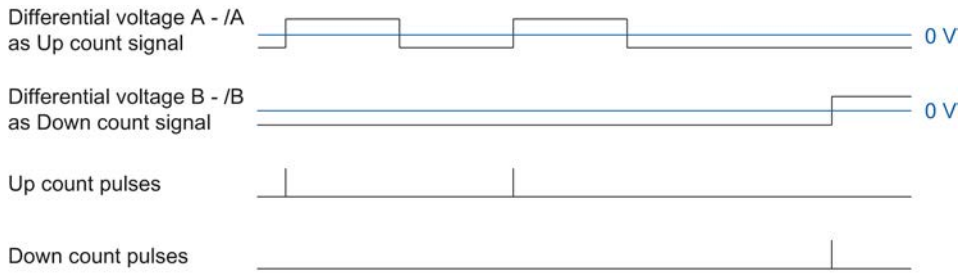
The figure below shows an example of the time profile of the signals of a RS422 pulse encoder with direction signal and the resulting count pulses:



Count signals of RS422 pulse encoders with Up/Down count signal

The Up count signal is connected to the A terminals. The Down count signal is connected to the B terminals.

The figure below shows an example of the time profile of the signals of an RS422 pulse encoder with Up/Down count signal and the resulting count pulses:



Monitoring of encoder signals

The technology module monitors RS422 signals for wire breaks, short-circuits, and offset voltages.

If you enable the diagnostic interrupt in the device configuration, the technology module triggers a diagnostic interrupt in the event of encoder signal errors.

SSI signals

Signals from SSI absolute encoders

The SSI absolute encoder and the technology module communicate via SSI data signals DAT and /DAT (D) and the SSI clock signals CLK and /CLK (C). SSI uses the RS422 signal standard. The signal information is coded in the respective differential voltage between C and /C as well as D and /D.

Monitoring of the encoder signals and the SSI frames

The technology module monitors the signals of an SSI absolute encoder for wire breaks, short-circuits, and offset voltages. The technology module also monitors SSI frames for errors.

If you enable the diagnostic interrupts in the device configuration, the technology module triggers a diagnostic interrupt in the event of encoder signal or SSI frame errors.

Signal evaluation of incremental signals

Overview

The technology module counter counts the edges of encoder signals A and B. For incremental encoders with phase-shifted signals A and B, you can select either single or multiple evaluation to improve the resolution.

You can configure the following signal evaluations:

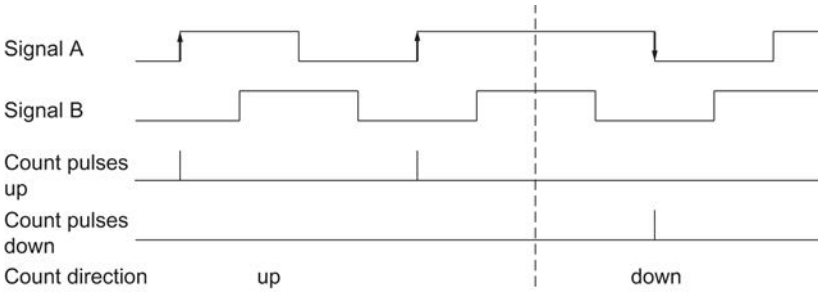
- Single evaluation (Page 9022)
- Double evaluation (Page 9022)
- Quadruple evaluation (Page 9023)

Single evaluation

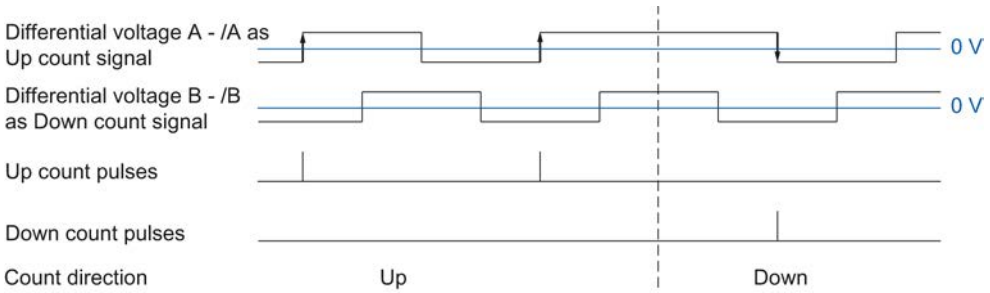
Single evaluation evaluates the rising and falling edge at signal A when signal B has a low level.

Count pulses in an upwards direction are generated with a rising edge at signal A during a low level at signal B. Count pulses in a downwards direction are generated with a falling edge at signal A during a low level of signal B.

The following figure shows an example for single evaluation of 24 V and TTL count signals:



The following figure shows an example for single evaluation of RS422 count signals:

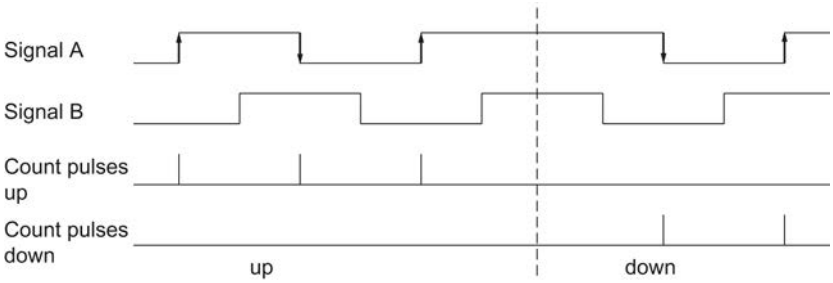


Double evaluation

With double evaluation, the rising and falling edges of signal A are evaluated.

The edge direction of signal A and the level at signal B determines whether count pulses are generated in an upward or downward direction.

The following figure shows an example for double evaluation of 24 V and TTL count signals:



The following figure shows an example for double evaluation of RS422 count signals:

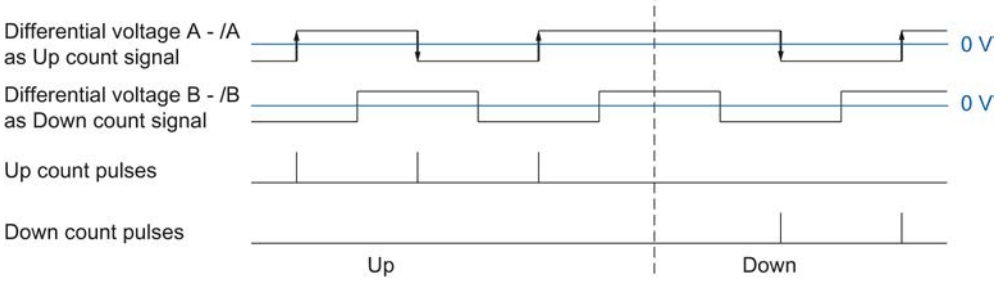


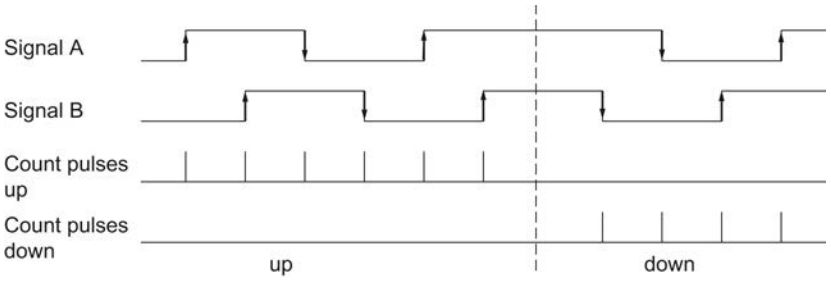
Figure 11-1 Double

Quadruple evaluation

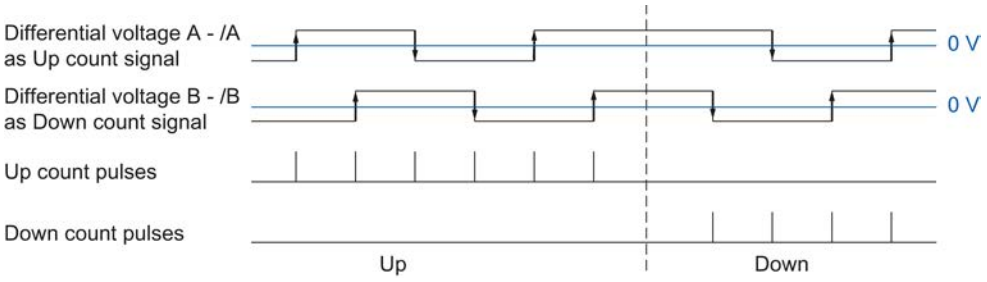
With quadruple evaluation, the rising and falling edges of signals A and B are evaluated.

The edge direction of one signal and the level of the other signal determines whether count pulses are generated in an upward or downward direction.

The figure below shows an example for quadruple evaluation of 24 V and TTL count signals:



The figure below shows an example for quadruple evaluation of RS422 count signals:



Isochronous mode

The technology module supports the "isochronous mode" system function in distributed mode. This system function enables position, counter and measured values to be recorded in a defined system cycle.

In isochronous mode, the cycle of the user program, the transmission of the input signals and processing in the technology module are synchronized. The output signals switch immediately if the relevant comparison condition is met. A change in the state of a digital input immediately affects the planned reaction of the technology module and changes the status bit of the digital input in the feedback interface.

Data processing

The data that was transmitted to the technology module in the current bus cycle via the control interface takes effect when it is processed in the internal technology module cycle. The position or counter value and the measured value as well as status bits are captured at time T_i and made available in the feedback interface for retrieval in the current bus cycle.

In isochronous mode, there is always data consistency across all bytes in the feedback interface.

11.1.1.2 Using the High_Speed_Counter

High_Speed_Counter technology object

STEP 7 (TIA Portal) helps you with the configuration of the counting and measuring functions for the technology modules with the "Technology objects" function in the project navigator ("Project > S7-1500 PLC > Technology objects"):

- You configure the High_Speed_Counter technology object in STEP 7 (TIA Portal) with the settings for the counting and measuring functions.
- The corresponding High_Speed_Counter instruction is programmed in the user program. This instruction supplies the control and feedback interface of the technology module.

The High_Speed_Counter technology object corresponds to the instance DB of the High_Speed_Counter instruction. The configuration of the counting and measuring functions is saved in the technology object. The technology object is located in the folder "PLC > Technology objects". It is implemented by an instance DB of the instruction.

The High_Speed_Counter technology object can be used for technology modules of both the S7-1500 and ET 200SP systems.

Counting operating mode

Counting mode is automatically defined if the technology object High_Speed_Counter is being used. The measured values are available concurrently.

Measuring mode

No technology object is available for the Measuring mode. The parameters of the Measuring mode are assigned using the parameter setting (HWCN) (Page 9054) of the technology module. The technology module is controlled by direct access to the control and feedback interface (Page 9074).

Position input

- The parameter assignment of the SSI absolute encoder (Page 9061) is performed using the parameter setting (HWCN) (Page 9054) of the technology module. No technology object is available for position detection using an SSI absolute encoder.
- The parameter assignment of the encoder signals for position input for Motion Control (Page 9049) is performed using the device configuration of the technology module. You can perform additional configuration for this application using the "TO_ExternalEncoder" technology object of S7-1500 Motion Control .

Overview of the configuration steps

Introduction

The overview below shows the basic procedure for configuring the counting and measuring functions of the technology module with the High_Speed_Counter technology object.

Requirement

Before you can use the High_Speed_Counter technology object, a project must be created in STEP 7 (TIA Portal) with an S7-1500 CPU.

Procedure

Proceed in the recommended sequence outlined below:

Step	Description
1	Configuring the technology module and specifying use with technology object (Page 9045)
2	Add technology object (Page 9025)
3	Working with the configuration dialog (Page 9027)
4	Call instruction in the user program (Page 9041)
5	Load to CPU
6	Commissioning the technology object (Page 9042)
7	Diagnostics of the technology object (Page 9043)

Add technology object

Add technology object in the project navigator

When a technology object is added, an instance DB is created for the instruction of this technology object. The configuration of the technology object is stored in this instance DB.

Requirement

A project with an S7-1500 CPU has been created.

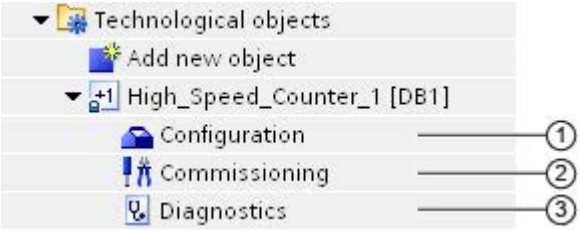
Procedure

To add a technology object, proceed as follows:

1. Open the CPU folder in the project tree.
2. Open the "Technology objects" folder.
3. Double-click on "Add new object".
   The "Add new object" dialog opens.
4. Select the "Counting and measurement" technology.
5. Select the "High_Speed_Counter" object.
6. Enter an individual name for the technology object in the "Name" text box.
7. Select the "Manual" option if you want to change the suggested data block number of the instance DB.
8. Click "Additional information" if you want to add your own information to the technology object.
9. Confirm with "OK".

Result

The new technology object has now been created and stored in the project tree in the "Technology objects" folder.



	Object	Description
①	Configuration (Page 9027)	In the configuration dialog: • Assignment of technology module and channel • Technology object parameter settings for counting and measurement functions When you change the configuration of the technology object, you must download the technology object **and** the hardware configuration to the CPU.
②	Commissioning (Page 9042)	Commissioning and function test of the technology object: Simulating parameters of the High_Speed_Counter instruction and monitoring the effects
③	Diagnostics (Page 9043)	Monitoring the counting and measuring functions

Configuring the High_Speed_Counter

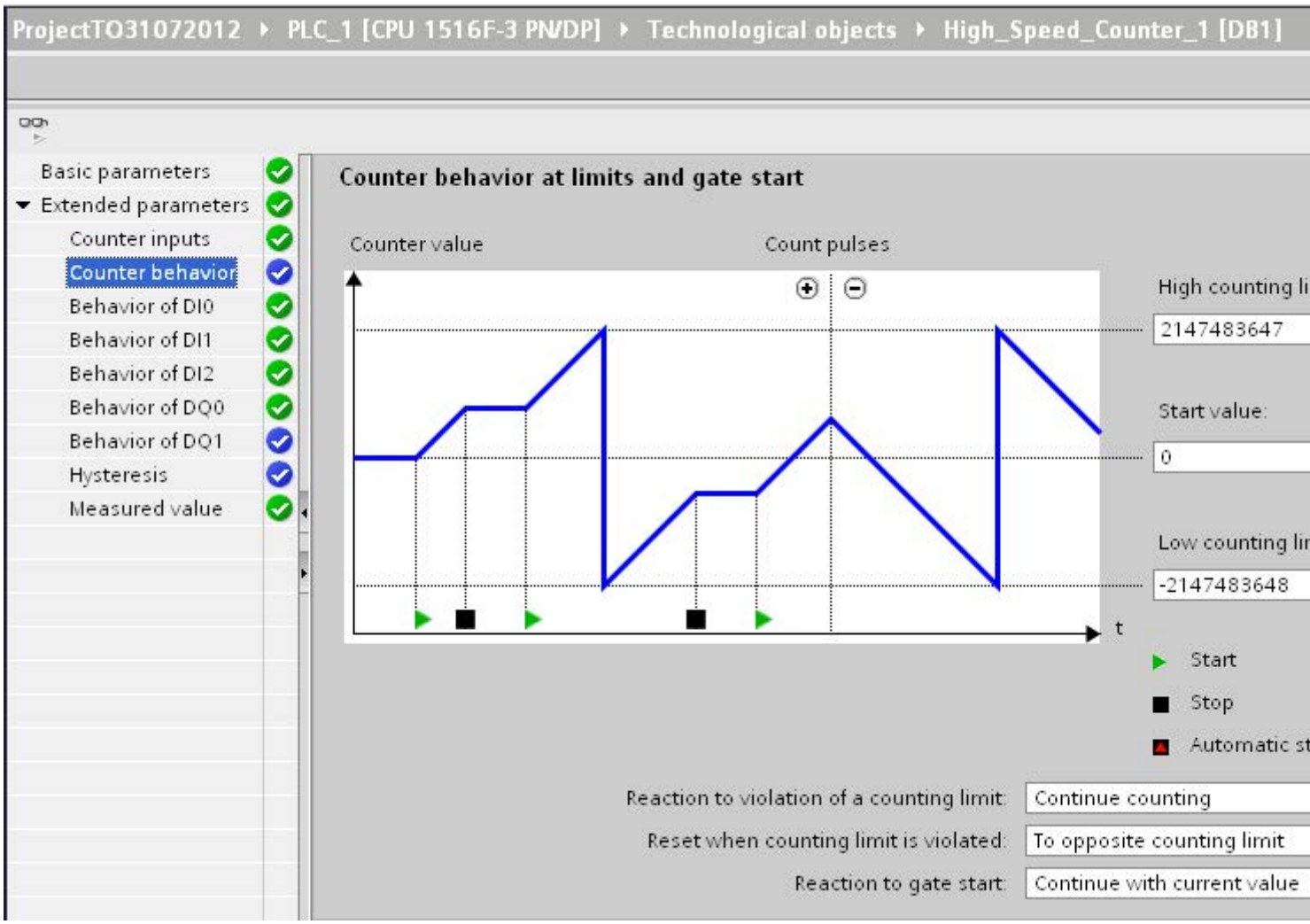
Working with the configuration dialog

You configure the properties of the technology object in the configuration window. Proceed as follows to open the configuration window of the technology object:

1. Open the "Technology objects" folder in the project tree.
2. Open the technology object in the project tree.
3. Double-click on the "Configuration" object.

The configuration is divided into the following categories:

- **Basic parameters**
  The basic parameters include the selection of the technology module and the number of the counting channel for which the technology object is configured.
- **Extended parameters**
  The extended parameters include the parameters for adapting the counting and measuring functions and for setting the characteristics of the digital inputs and digital outputs.



Configuration window icons

Icons in the area navigation of the configuration show additional details about the status of the configuration:

	The configuration contains default values and is complete. The configuration contains only default values. With these default values, you can use the technology object without additional changes.
	The configuration contains values set by the user and is complete. All text boxes of the configuration contain valid values and at least one default value was changed.
	The configuration is incomplete or incorrect At least one text box or drop-down list contains an invalid value. The corresponding field or the drop-down list is displayed on a red background. Click the roll-out error message to indicate the cause of error.

Basic parameters

You can establish the connection between the High_Speed_Counter technology object and the technology module under "Basic parameters".

Module

You select the technology module in a subsequent dialog box. You can choose any of the technology modules (central or distributed) which are configured in the S7-1500 CPU for use with a technology object from "Counting and measuring".

After selecting the technology module, you can open the device configuration associated with the technology module by clicking the"Device configuration" button.

The technology module parameter settings required for the use of the technology object are made in the "Extended parameters" of the technology object.

Channel

For a technology module with several counting channels, you can also select the number of the counting channel for which the High_Speed_Counter technology object is to be valid.

Synchronization of the parameter values

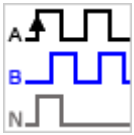
If there is an inconsistency between the parameter values under "Parameters" (HWCN) and in the technology object after assigning the count channel to the technology object, a button appears: "Overwrite technology module parameters". When you click the button, the parameter values under "Parameters" (HWCN) are overwritten (after a prompt) with the parameter values of the technology object in STEP 7 (TIA Portal). The current parameter values (read only) of the technology object are displayed under "Parameters" (HWCN).

Counter inputs

Specifying input signals/encoder type

Signal type

You can choose from the following signal types (Page 9018):

Symbol	Signal type	Meaning
	Incremental encoder (A, B phase-shifted)	An incremental encoder with phase-shifted A and B signals is connected.
	Incremental encoder (A, B, N)	An incremental encoder with phase-shifted signals A and B and a zero signal N is connected.

Symbol	Signal type	Meaning
	Pulse (A) and direction (B)	A pulse encoder (signal A) with direction signal (signal B) is connected.
	Pulse (A)	A pulse encoder (signal A) without direction signal is connected. You can specify the counting direction by means of the control interface (Page 9074).
	Count up (A), count down (B)	Signals for counting up (signal A) and down (signal B) are connected.

Invert direction

You can invert the count direction to adapt it to the process.

Inversion of direction can only be configured and become effective if signal type "Incremental encoder (A, B phase-shifted)" or "Incremental encoder (A, B, N)" is selected.

Additional parameters

Signal evaluation

By configuring signal evaluation (Page 9021), you can specify which edges of the signals are counted.

You can select from the following options:

Icon	Signal evaluation	Meaning
	Single (Page 9022) (default)	The edges of signal A are evaluated during a low level of signal B.
	Double (Page 9022)	Each edge of signal A is evaluated.
	Quadruple (Page 9023)	Each edge of signals A and B is evaluated.

The parameter can be assigned with the following signal types:

- Incremental encoder (A, B phase-shifted)
- Incremental encoder (A, B, N)

Filter frequency

By configuring the filter frequency, you suppress interferences at the counter inputs A, B and N.

The selected filter frequency is based on a pulse-break ratio of between 40:60 and 60:40. This produces a set minimum pulse/break time. Signal changes with a duration shorter than the minimum pulse/break time are suppressed.

You can select from the following filter frequencies:

Table 11-2 Filter frequency and respective minimum pulse/break time

Filter frequency	Minimum pulse/break time
100 Hz	4,0 ms
200 Hz	2,0 ms
500 Hz	800 µs
1 kHz	400 µs
2 kHz	200 µs
5 kHz	80 µs
10 kHz	40 µs
20 kHz	20 µs
50 kHz	8,0 µs
100 kHz	4,0 µs
200 kHz (default at TM Count)	2,0 µs
500 kHz*	0,8 µs
1 MHz* (default at TM PosInput)	0,4 µs

* only with TM PosInput

Sensor type

By configuring the sensor type, you specify how the counter inputs are wired for the TM Count.

You can select from the following options:

Sensor type	Meaning
Sourcing output (default)	The encoder/sensor switches the inputs A, B and N to 24VDC.
Sinking output	The encoder/sensor switches the inputs A, B and N to M.
Push-pull (sinking and sourcing output)	The encoder/sensor alternately switches the inputs A, B and N to M and 24VDC.

"Push-pull" is usually selected when incremental encoders are used. If using 2-wire sensors, such as light barriers or proximity switches, you need to select the corresponding wiring "sourcing output" or "sinking output".

The data sheet of the encoder includes information on whether your incremental encoder is a push-pull encoder.

Note

If you use a push-pull encoder, you can monitor the encoder signals for wire break.

Interface standard

You use this parameter to specify whether the encoder outputs symmetric (RS422) or asymmetric (TTL) signals for the TM PosInput.

You can select from the following options:

Interface standard	Meaning
RS422, symmetric	The encoder outputs symmetric signals according to the RS422 standard (Page 9019).
TTL (5 V), asymmetric	The encoder outputs asymmetric 5 V signals according to the TTL standard (Page 9018).

Note

The RS422 standard provides greater interference immunity than the TTL standard. If your incremental or pulse encoder supports the RS422 standard **and** the TTL standard, we recommend using the RS422 standard.

Reaction to signal N

You use this parameter to specify which reaction is triggered at signal N.

You can select from the following options:

Option	Meaning
No reaction to signal N (default)	The counter is not affected by signal N.
Synchronization at signal N (Page 8995)	The counter is set to the start value at signal N. If you select the function "Enable synchronization at signal N" for a digital input, the synchronization depends on the level at the digital input.
Capture at signal N (Page 8987)	The counter value is stored in the Capture value at signal N.

Note

You can only select the reaction at signal N if you have selected the signal type (Page 9029) "Incremental encoder (A, B, N)".

Note

If "Synchronization at signal N" is selected, you can select the function "Enable synchronization at signal N" for a digital input (Page 9035).

Frequency

This parameter is used to define the frequency of the following events:

- Synchronization at signal N
- Synchronization as function of a digital input

You can select from the following options:

Option	Meaning
Once (default)	The counter is only set at the first signal N or the first configured edge of the digital input.
Periodic	The counter is set at each signal N or each configured edge of the digital input.

Counter behavior

Counting limits and start value

High counting limit

You limit the counting range by setting the high counting limit. You can enter a value up to 2147483647 (2^{31}-1). You must enter a value that lies above the low counting limit.

The default setting is "2147483647".

Low counting limit

You limit the counting range by setting the low counting limit. You can enter a value up to -2147483648 (-2^{31}). You must enter a value below the high counting limit.

The default setting is "-2147483648".

Start value

By configuring the start value, you specify the value at which counting is to start and where it is to continue in the case of defined events. You must enter a value between the counting limits or equal to the counting limits.

The default setting is "0".

Additional information

For more information, see Behavior at the counting limits (Page 8982) and Counter behavior at gate start (Page 8986).

Counter behavior at limits and gate start

Reaction to violation of a counting limit

You can configure the following characteristics for violation of a counting limit (Page 8982):

Reaction	Meaning
Stop counting	If a counting limit is violated, counting is stopped and the internal gate is closed. To restart counting, you must also close and reopen the software/ hardware gate.
Continue counting (default)	Counting continues either with the start value or at the opposite counting limit, depending on the additional parameter assignment.

Reset when counting limit is violated

You can reset the counter to the following values when a counting limit is violated:

Reset the value	Meaning
To start value	The counter value is set to the start value.
To opposite counting limit (default)	The counter value is set to the opposite counting limit.

Reaction to gate start

You can set the following Reaction to gate start (Page 8986):

Reaction	Meaning
Set to start value	When the gate is opened, the counter value is set to the start value.
Continue with current value (default)	When the gate is opened, counting is continued with the last counter value.

Behavior of a DI

Function of a DI

Setting function of the DI

By configuring a digital input, you specify which functions the digital input triggers at switching.

You can select from the following options:

Function of a digital input	Meaning
Gate start/stop (level-triggered)	The level at the respective digital input opens and closes the hardware gate (Page 8984).
Gate start (edge-triggered)	The configured edge at the respective digital input opens the hardware gate (Page 8984).
Gate stop (edge-triggered)	The configured edge at the respective digital input closes the hardware gate (Page 8984).
Synchronization (Page 8991)	The configured edge at the respective digital input sets the counter to the start value.
Enable synchronization at signal N	The active level at the respective digital input enables synchronization of the counter at signal N (Page 8995).
Capture (Page 8987)	The configured edge at the respective digital input saves the current counter value as a Capture value.
Digital input without function	No function is assigned to the respective digital input. The signal status of the digital input can be read by the CPU by means of the feedback interface (Page 9077).

Note

With the exception of "Digital input without function", each function can only be used once, and if used for one digital input is not available to the others.

Filter time

By configuring the filter time, you suppress interferences at the digital inputs. Signals with a pulse duration below the configured filter time are suppressed.

You can select from the following filter times:

- None
- 0.05 ms
- 0.1 ms (default)
- 0.4 ms
- 0.8 ms

- 1.6 ms
- 3.2 ms
- 12.8 ms
- 20 ms

Note

If you select the option "None" or "0.05 ms", you have to use shielded cables for connecting the digital inputs.

Note

You can only configure the filter time under "Behavior of DI0" for all digital inputs together.

Function options

Certain functions require additional parameters which allow you to specify characteristics in more detail. These can be configured for each of the relevant functions.

Select level

You use this parameter to specify the level at which the digital input is active.

You can select from the following options:

Level	Meaning
Active with high level (default)	The respective digital input is active when it is set.
Active with low level	The respective digital input is active when it is not set.

The parameter can be set for the following functions of a digital input:

- Gate start/stop (level-triggered)
- Enable synchronization at signal N

Edge selection

You can use this parameter to specify the edge of the digital input at which the configured function is triggered.

The following options may be available depending on the function selected:

- At rising edge (default)
- At falling edge
- At rising and falling edge

The parameter can be set for the following functions of a digital input:

- Gate start (edge-triggered)
- Gate stop (edge-triggered)
- Synchronization
- Capture

"At rising and falling edge" can only be configured for the function "Capture".

Behavior of counter value after Capture

You can configure the following characteristics for the counter after a capture event (Page 8987):

	Meaning
Continue counting (default)	After saving the current count value as Capture value, counting is continued unchanged.
Set to start value and continue counting	After saving the current count value as Capture value, counting is continued with the start value.

See also

Function of a DI (Page 9035)

Function of a DQ

Set output

By configuring a digital output, you can specify the condition (Page 8999) upon which the digital output switches.

You can select from the following options:

Function of a digital output	Meaning
Between comparison value and high limit (default)	The respective digital output is active if the counter value is between the comparison value and the high counting limit.
Between comparison value and low counting limit	The respective digital output is set to active if the counter value is between the comparison value and the low counting limit.
Between comparison value 0 and 1	The digital output DQ1 is active if the counter value is between the comparison value 0 and the comparison value 1.
At comparison value for a pulse duration	The respective digital output is active once for the configured time and count direction when the count value reaches the comparison value.

Function of a digital output	Meaning
After set command from CPU until comparison value	When a set command is sent from the CPU, the respective digital output is active until the counter value is equal to the comparison value.
Use by user program	The respective digital output (Page 8998) can be switched by the CPU via the control interface (Page 9074).

Note

You can select the function "Between comparison value 0 and 1" if the function "Use by user program" is selected for digital output DQ0.

Note

The "At comparison value for a pulse duration" and "After set command from CPU until comparison value" functions only switch the digital output in question if a count pulse reaches the comparison value. The digital output does not switch when the counter value is set, by synchronization for example.

Comparison value 0

By configuring the comparison value (Page 8998), you can specify the counter value at which the digital output DQ0 is to switch as a result of the selected compare event.

You must enter a value which is greater than or equal to the low counting limit and smaller than comparison value 1.

The default setting is "0".

Comparison value 1

By configuring the comparison value (Page 8998), you can specify the counter value at which the digital output DQ1 is to switch as a result of the selected compare event.

You must enter a value which is greater than comparison value 0 and smaller than or equal to the high counting limit.

The default setting is "10".

Count direction

You use this parameter to specify the count direction for which the selected function is valid.

You can select from the following options:

Count direction	Meaning
In both directions (default)	The comparison and switching of the respective digital output take place regardless of the count direction.
Up	The comparison and switching of the respective digital output only takes place when the counter counts up.
Down	The comparison and switching of the respective digital output only takes place when the counter counts down.

The parameter can be configured for the following functions:

- At comparison value for a pulse duration
- After set command from CPU until comparison value

Pulse duration

By configuring the pulse duration for the function "At comparison value for a pulse duration", you specify the number of milliseconds for which the respective digital output is active.

If you enter "0" and the counter value corresponds to the comparison value, the digital output remains active until the next count pulse.

You can enter a value between 0.0 and 6553.5.

The default setting is "500.0", which is equivalent to a pulse duration of 0.5 s.

Hysteresis

Set hysteresis range

By configuring hysteresis (Page 9014), you can define a range around the comparison values. Within the hysteresis range, the digital outputs cannot switch again until the counter value is outside the range.

Regardless of the hysteresis value, the hysteresis range ends at the low/high counting limits. If you enter "0", the hysteresis is turned off. You can enter a value between 0 and 255. The default setting is "0".

Hysteresis can only be configured in Counting mode.

Specifying the measured value

Measured variable

This parameter is used to specify the measured variable (Page 9010) to be provided by the technology module.

You can select from the following options:

Measured variable	Meaning
Frequency (default)	The measured variable is a frequency. The unit is Hz.
Period	The measured variable is a period. The unit is s.
Velocity	The measured variable is a velocity. You configure the time basis for the velocity measurement and the Increments per unit separately.

Update time

By configuring the update time (Page 9009) in milliseconds, you can specify the time interval between two measured value updates.

The update time and the signal type (Page 9029) effect the accuracy of the measurement. In the case of update times of at least 100 ms, the effect of the signal type is negligible.

In the case of update times of less than 100 ms, you achieve maximum measurement accuracy using the following signal types:

- Incremental encoder (A, B phase-shifted) with Signal evaluation "Single"
- Incremental encoder (A, B, N) with Signal evaluation "Single"
- Pulse (A) and direction (B)
- Pulse (A)

In the case of other signal types, measurement accuracy depends on the encoder and cable used.

If you enter "0", the measured value is updated in each cycle. Up to three decimal places can be entered. You can enter a value between 0.000 and 25000.000. The default setting is "10.000".

Time base for velocity measurement

This parameter defines the time base on which the velocity is to be returned.

You can select from the following options:

- 1 ms
- 10 ms
- 100 ms
- 1 s
- 60 s/1 min
- The default setting is "60 s/1 min".

Increments per unit

This parameter defines the number of count pulses per relevant unit.

The number of count pulses depends on the configured signal evaluation. You can enter a value between 1 and 65535.

Example:

Your encoder delivers 4000 increments per meter. The velocity is to be measured in meters per minute. "Double" is configured as signal evaluation.

In this case, you need to assign the following parameters:

- Increments per unit: 8000
- Time base for velocity measurement: 60 s/1 min

Programming the High_Speed_Counter

High_Speed_Counter instruction

High_Speed_Counter

The High_Speed_Counter instruction is part of the High_Speed_Counter technology object. It supplies the control and feedback interface of the technology module.

The High_Speed_Counter instruction therefore forms the software interface between the CPU and the technology module. It is called cyclically from the user program to synchronize the input and output data.

The High_Speed_Counter instruction can be used for the technology modules of both S7-1500 and ET 200SP.

Additional information

Description High_Speed_Counter (Page 4920)

High_Speed_Counter input parameters (Page 4923)

High_Speed_Counter output parameters (Page 4924)

ErrorID parameter (Page 4925)

High_Speed_Counter static variables (Page 4926)

Call instruction in the user program

The High_Speed_Counter instruction can be called once for each counter in the cycle or, alternatively, in a time-controlled program. The call is not permitted in an event-controlled interrupt program.

Procedure

Proceed as follows to call the instruction in the user program:

1. Open the CPU folder in the project tree.
2. Open the "Program blocks" folder.
3. Double-click the OB for cyclic program execution.
   The block is opened in the work area.

4. In the "Instructions" window, open the "Technology" group and the "Counting and measurement" folder.
   The instructions are located in the folder.
5. Select an instruction and drag it to your OB.
   The "Call options" dialog opens.
6. Select a technology object from the "Name" list or enter the name for a new technology object.

Result

If the technology object does not exist yet, it is added. The instruction is added in the OB. The technology object is assigned to this call of the instruction.

Commissioning the High_Speed_Counter

Commissioning the technology object

The commissioning editor helps you with commissioning and the function test for the technology object. You can change specific parameters of the High_Speed_Counter instruction in CPU/IM online mode and monitor their effects.

Requirements

- There is an online connection between STEP 7 (TIA Portal) and the CPU.
- The CPU is in RUN.
- The corresponding High_Speed_Counter instruction is called cyclically from the user program.
- The parameters of the technology object are not overwritten by the user program.

Procedure

To open the commissioning editor of a technology object, follow these steps:

1. Open the "Technology objects" folder in the project tree.
2. Open the High_Speed_Counter technology object in the project tree.
3. Double-click on the "Commissioning" object.
   The functions for commissioning the High_Speed_Counter technology object are displayed.
4. Click the button "Launch display".
   The parameters (online values) of the High_Speed_Counter technology object are loaded and displayed.

Online mode

In online mode, you can modify the following parameters to test the technology object function:

- New counter value (NewCountValue)
- New high counting limit (NewUpperLimit)
- New low counting limit (NewLowerLimit)
- New comparison value 0 (NewReferenceValue0)
- New comparison value 1 (NewReferenceValue1)
- New start value (NewStartValue)
- Start and stop counter (SwGate)
- Enable Capture (CaptureEnable)
- Enable synchronization (SyncEnable)
- Acknowledgment of signaled error states (ErrorAck)
- Resetting the status flag (EventAck)

Modified values become effective as new parameters of the technology object when the text box is closed, when you press ENTER or by additionally setting the relevant set variable.

High_Speed_Counter diagnostics

Monitoring counter values, measured values, DIs and DQs

You use the diagnostic functions to monitor the counting and measuring functions.

Requirements

- There is an online connection between STEP 7 (TIA Portal) and the CPU.
- The CPU is in RUN.

Procedure

To open the display editor for the diagnostic functions, follow these steps:

1. Open the "Technology objects" folder in the project tree.
2. Open the High_Speed_Counter technology object in the project tree.
3. Double-click on the "Diagnostics" object.

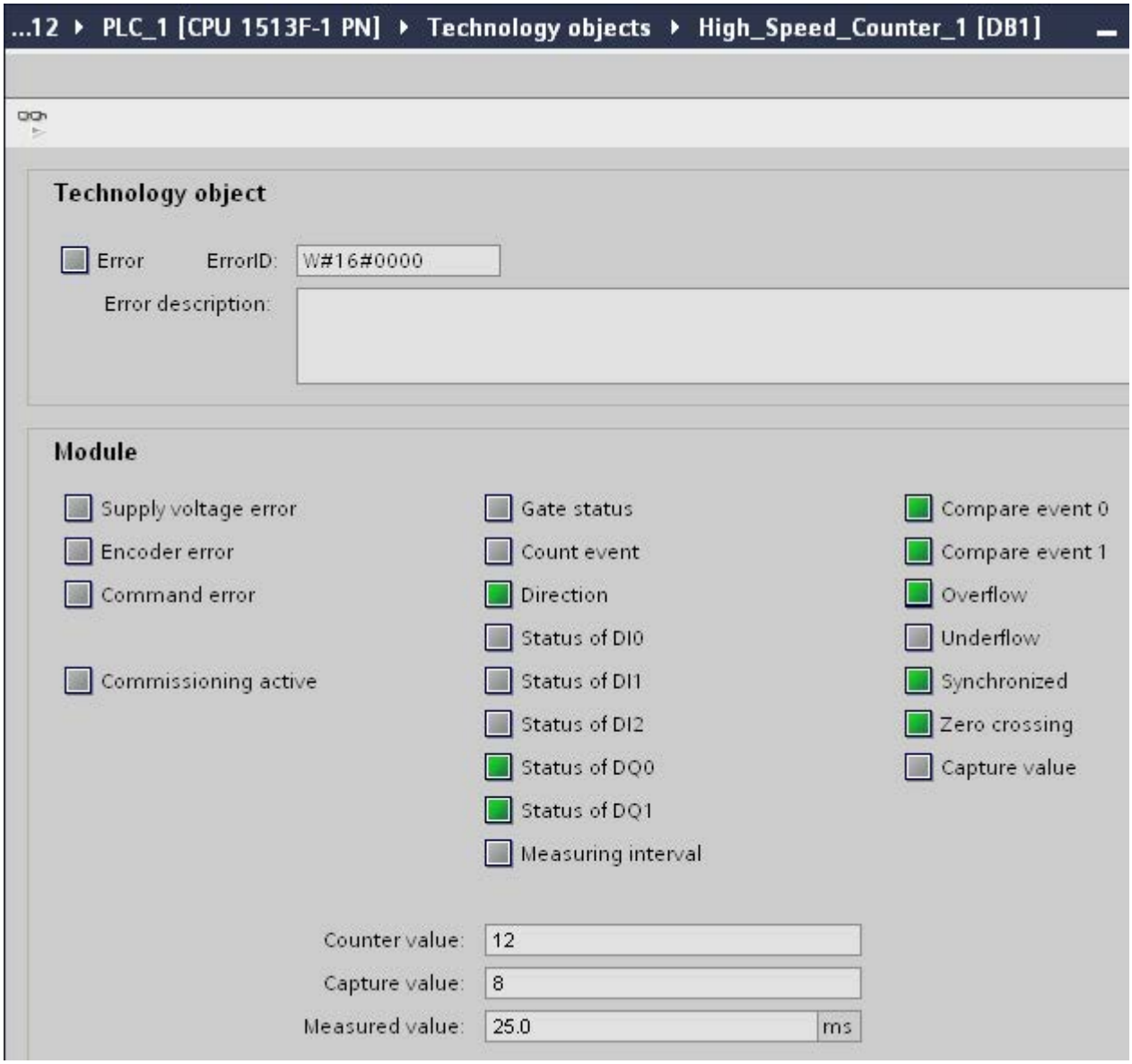
Display

The following values are read out from the technology object and displayed:

- Event display/diagnostics information
- Signal states of the digital inputs and digital outputs

- Counter value
- Capture value
- Measured value

Additional information on status displays is available in the context-sensitive help for each event in STEP 7 (TIA Portal). When the CPU is in STOP, the status display is not updated.



11.1.1.3 Using the technology module

Configuring a module

Adding a technology module to hardware configuration

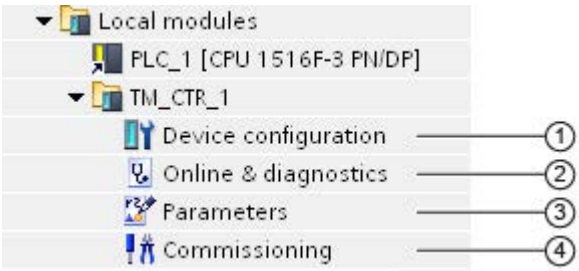
Procedure

A technology module can only be configured if you have set up a project in which the configuration can be saved.

1. Open the device configuration of the CPU or IM.
2. Select a module rack.
3. Select the technology module from the module catalog:
   "TM > Counting or Position feedback> Technology module > Order number".
4. Drag the technology module to the required slot in the module rack.

Result

The new technology module is now displayed with the following objects under "Local modules" or "Distributed I/O" in the project tree. Double-click to open the required editor.



	Object	Description
①	Device configuration (Page 9046)	In the Inspector window (per channel): • Setting the reaction to CPU STOP • Enable diagnostic interrupt • Setting the operating mode (Page 9048) • Enable hardware interrupts (Page 9051) • Setting the module addresses
②	Online & Diagnostics (Page 9071)	• Hardware diagnostics • Obtain information on technology module • Run firmware update

	Object	Description
③	Parameter (Page 9053)	Display and, if applicable, parameter settings for the counting and measuring functions in HWCN, if parameter assignment and control are not implemented using the High_Speed_Counter technology object.
④	Commissioning (Page 9072)	Commissioning and function test of the technology module: Simulate counter values and DIs in CPU or IM online mode and monitor their effects.

Basic parameters

Reaction to CPU STOP

Reaction to CPU STOP

You set the response of the technology module for each channel to CPU STOP in the basic parameters of the device configuration.

Table 11-3 Reaction of the technology module to CPU STOP depending on parameter assignment (by channel)

Basic parameters	Reaction to CPU STOP
Continue operation	The technology module remains fully functional. Incoming count pulses are processed or the actual position is read. The digital outputs continue to switch according to the parameter assignment.
Output substitute value	The technology module outputs the configured substitute values at the digital outputs until the next CPU STOP-RUN transition. The technology module is returned to its startup state after a STOP-RUN transition: The counter value is set to the Start value (with incremental encoders or pulse encoders) and the digital outputs switch according to the parameter assignment.
Keep last value	The technology module outputs the values at the digital outputs that were valid when the transition to STOP took place until the next CPU STOP-RUN transition. The technology module is returned to its startup state after a STOP-RUN transition: The counter value is set to the Start value (with incremental encoders or pulse encoders) and the digital outputs switch according to the parameter assignment.

Substitute value for DQ0

This parameter lets you specify which value the technology module is to output to the digital output DQ0 in the event of a CPU STOP under "Output substitute value".

Substitute value for DQ1

This parameter lets you specify which value the technology module is to output to the digital output DQ1 in the event of a CPU STOP under "Output substitute value".

Diagnostic interrupts

The technology module can trigger additional diagnostic interrupts when you enable the diagnostic interrupts in the basic parameters. These diagnostic interrupts are processed in an interrupt OB.

Enabling diagnostic interrupts for wire break

You use this parameter to specify whether a diagnostic interrupt is to be triggered in the event of wire breaks of the utilized signals for the following encoders:

- Push-pull 24 V encoder (Page 9018)
- RS422 encoder (Page 9019) (also monitored for short-circuits and offset voltages)
- SSI absolute encoder (Page 9021) (also monitored for short-circuits and faulty voltages)

Note

If you use an encoder with a different sensor type or interface standard, wire break cannot be detected.

Enabling additional diagnostic interrupts

You use this parameter to specify whether diagnostic interrupts are to be triggered for additional errors.

See the device manual for the technology module to find out which errors during operation can trigger a diagnostic interrupt.

Operating mode

Selecting the mode for the channel

This setting defines how the channel counting and measuring functions are to be configured and controlled.

Operating mode	Description
Operating with technology object	The channel is configured using the High_Speed_Counter technology object. Access to the control and feedback interface of the technology module is provided by the corresponding High_Speed_Counter instruction in the user program. Technology module/channel assignment to a technology object is defined at the start of technology object configuration: see Basic parameters technology object (Page 9029) Counting mode is automatically defined if the technology object High_Speed_Counter is being used.
Position input for Motion Control	The technology module is used for position input for a higher-level Motion Control controller. All channels of the technology module for Motion Control are then available. Parameter assignment is implemented via the Device configuration of the technology module. Parameter assignment of the encoder signals is implemented by means of the Module parameters (Page 9049).
Manual operation	The channel is configured using the parameter setting (HWCN) (Page 9054) of the technology module. You have direct access to the control and feedback interface of the technology module from the user program.

Selects the channel operating mode

You can use this parameter in "Manual operation" to specify the function for which the channel of the technology module is used. This defines the settings options under "Parameters" (HWCN).

Operating mode	Description
Counting (Page 8980) / Position input (Page 8981)	The main function of the technology module is counting or position input. The comparison functions and hardware interrupts work with the counter value or position value. The measured value is available concurrently.
Measuring (Page 9008)	The main function of the technology module is measuring. The comparison functions (Page 9060) and hardware interrupts for compare events work with the measured value. The counter value is available concurrently.

Module parameters (position input for Motion Control)

In the operating mode "Position input for Motion Control" mode, the parameters for the encoder signals of the channel are assigned under "Module parameters". The parameters depend on the encoder used.

Module parameters for incremental encoders and pulse encoders

If you use an incremental or pulse encoder, you need to configure the following parameters for the encoder signals of the channel.

- Signal type
- Invert direction
- Signal evaluation
- Filter frequency
- Sensor type (for TM Count)
- Interface standard (for TM PosInput)
- Signal selection for reference mark 0
- Increments per unit and
- Reference speed

You can find a description of the first six parameters in section Counter inputs (Page 9029).

Module parameters for SSI absolute encoders

If you use an TM PosInput with an SSI absolute encoder, you need to configure the following parameters for the encoder signals of the channel.

- Signal type
- Invert direction
- Frame length
- Type of code
- Transmission rate
- Monoflop time
- Parity
- LSB bit number of position setpoint
- MSB bit number of position setpoint
- Increments per unit and
- Reference speed

You can find additional information on the first new parameters in section Configure position detection with SSI absolute encoder (Page 9061).

Signal selection for reference mark 0

When S7-1500 Motion Control is used, this parameter specifies the external reference signal upon which a new reference mark is saved for the encoder position.

You can select from the following options:

Option	Meaning
DI0	The current counter value is saved as the new encoder position reference mark upon a rising edge at digital input DI0.
Signal N of incremental encoder	The current counter value is saved as the new encoder position reference mark upon a rising edge of the signal N of incremental encoder.

Increments per unit

When using an incremental encoder or pulse encoder, you specify the number of counter pulses per encoder revolution with this parameter. When you use an SSI absolute encoder, you use this parameter to specify the number of increments the encoder delivers per revolution.

The number of count pulses depends on the configured signal evaluation. You can enter a value between 1 and 65535.

Example of an incremental or pulse encoder:

Your incremental or pulse encoder delivers 2048 increments per revolution. Depending on the signal evaluation , you need to assign the following in this case:

Signal evaluation	Increments per unit
Single	2048
Double	4096
Quadruple	8192

Reference speed

When an S7-1500 Motion Control is used, the encoder transmits the process value of the speed as a percentage of the reference speed. This parameter defines the speed in rpm which is to be correspond to the value 100 % . The reference speed must be identical to the setting in the controller.

You can enter a value between 6.00 and 210000.00. The default setting is "3000.00".

Further configuring

You can perform additional configuration using the "TO_ExternalEncoder" technology object of S7-1500 Motion Control . See the S7-1500 Motion Control (http://support.automation.siemens.com/WW/view/en/59381279) function manual for further details on configuring and commissioning position input.

Hardware interrupts

You can set for each channel the events which are to trigger a hardware interrupt during operation when assigning the basic parameters of the technology module.

In an S7-1500 system, you enter a matching event name for each enabled hardware interrupt and assign a corresponding hardware interrupt OB to each hardware interrupt. If a hardware interrupt is triggered, the corresponding OB is started to evaluate the hardware interrupt data.

A hardware interrupt is triggered if the condition for changing the respective status bit or event bit in the feedback interface is fulfilled.

Lost hardware interrupt

If an event occurs which is supposed to trigger a hardware interrupt but the previous, identical event has not yet been processed, no further hardware interrupt is triggered. The hardware interrupt is lost. This may lead to the "Hardware interrupt lost" diagnostic interrupt, depending on the parameter assignment.

Hardware interrupts that can be activated

Hardware interrupt	Available in counting mode using		Available in measuring mode using		Available in position input for Motion Control mode	Description	EventType number
	Incremental or pulse encoder	SSI absolute encoder	Incremental or pulse encoder	SSI absolute encoder			
New Capture value available	Yes	Yes	No	No	No	Hardware interrupt when current counter value or position value is saved as Capture value	8
Synchronization of the counter by an external signal	Yes	No	Yes	No	No	Hardware interrupt upon synchronization of the counter by signal N or DI edge	9
Gate start	Yes	No	Yes	No	No	Hardware interrupt when internal gate opens	1
Gate stop	Yes	No	Yes	No	No	Hardware interrupt when internal gate closes	2
Overflow (high counter limit violated)	Based on counter value	No	Based on counter value	No	No	Hardware interrupt when counter value violates high counting limit	3
Underflow (low counter limit violated)	Based on counter value	No	Based on counter value	No	No	Hardware interrupt when counter value violates low counting limit	4
Direction reversal*	Based on counter value	Based on position value	Based on counter value	Based on position value	No	Hardware interrupt when counter value or position value changes direction	10

Hardware interrupt	Available in counting mode using		Available in measuring mode using		Available in position input for Motion Control mode	Description	EventType number
	Incremental or pulse encoder	SSI absolute encoder	Incremental or pulse encoder	SSI absolute encoder			
Zero pass	Based on counter value	Based on position value	Based on counter value	Based on position value	No	Hardware interrupt with zero pass of counter value or position value	7
Comparison event for DQ0 occurred	Based on counter value	Based on position value	Based on measured value	Based on measured value	No	Hardware interrupt when a comparison event for DQ0 occurs as a result of the selected comparison condition. No hardware interrupt when the change of the counter value for an incremental or pulse encoder was not caused by a count pulse	5
Comparison event for DQ1 occurred	Based on counter value	Based on position value	Based on measured value	Based on measured value	No	Hardware interrupt when a comparison event for DQ1 occurs as a result of the selected comparison condition. No hardware interrupt when the change of the counter value for an incremental or pulse encoder was not caused by a count pulse	6

* The feedback bit STS_DIR has the default value "0". A hardware interrupt is not triggered when the first change to the counter value or position value immediately after the technology module is switched on is in the down direction.

Default setting

No hardware interrupts are enabled in the default setting.

Parameter assignment of module

Parameter assignment options

Counting mode — Parameter assignment options

You have two options for technology module counting function parameter assignment and control:

- Configuration of a High_Speed_Counter technology object and control with a corresponding High_Speed_Counter instruction

  Note

  Configuration via technology object

  We recommend the simple configuration with graphic support using the High_Speed_Counter technology object. A detailed description of this configuration can be found in section Auto-Hotspot and below.

  You start technology object parameter assignment (Page 9029) by assigning the technology object the configured technology module and counting channel.

- Parameter settings via HWCN (Page 9054) and control via the control and feedback interface of the technology module

  Note

  Parameter setting via HWCN

  Additional support for parameter settings via HWCN is available in the context-sensitive help for the parameters in STEP 7 (TIA Portal). A description of the control and feedback interface is available in the following sections:

  Assignment of the control interface (Page 9074)

  Assignment of the feedback interface (Page 9077)

Measuring mode

No technology object is available for the measuring mode. The parameters (Page 9054) of the Measuring mode are assigned using the parameter setting (HWCN) (Page 9054) of the technology module.

The technology module is controlled by direct access to the control and feedback interface.

Position input with SSI absolute encoder

No technology object is available for position input with SSI absolute encoder.

- In "Manual operation", the parameter assignment (Page 9061) of the SSI absolute encoder, the digital inputs and digital outputs is implemented by means of the Parameter setting (HWCN) (Page 9054) of the technology module.
- In operating mode "Position input for Motion Control", the parameter assignment of the SSI absolute encoder is implemented by means of the module parameters (Page 9049) during the device configuration of the technology module.

Feedback from the technology module is returned by means of the feedback interface.

Opening the parameter settings (HWCN)

Opening via the project tree

Proceed as follows:

1. Open the "Local modules" or "Distributed I/O" folder in the project tree.
2. Open the technology module in the project tree.
3. Double-click on the "Parameters" object.

Opening from the device view

Proceed as follows:

1. Open the device configuration of the CPU or IM.
2. Select the device view.
3. Right-click on the technology module and select "Parameters".

Measuring mode

Overview

In Measuring mode, the following applies:

- Counter inputs
  - Specifying input signals/encoder type (Page 9055)
  - Additional parameters (Page 9055)
- Specifying the measured value (Page 9039)
- Behavior of a DI
  - Function of a DI (Page 9058)
  - Function options (Page 9059)

- Behavior of a DQ
  - Function of a DQ (Page 9060)
- Counter behavior (Page 9033)

Counter inputs: Specifying input signals/encoder type (Measuring mode)

Signal type

You can choose from the following signal types:

Signal type	Meaning
Incremental encoder (A, B phase-shifted)	An incremental encoder with phase-shifted A and B signals is connected.
Incremental encoder (A, B, N)	An incremental encoder with phase-shifted signals A and B and a zero signal N is connected.
Pulse (A) and direction (B)	A pulse encoder (signal A) with direction signal (signal B) is connected.
Pulse (A)	A pulse encoder (signal A) without direction signal is connected. You can specify the counting direction by means of the control interface (Page 9074).
Count up (A), count down (B)	Signals for counting up (signal A) and down (signal B) are connected.
Absolute encoder (SSI)	An SSI absolute encoder is connected (only applicable if TM PosInput is being used): see Configure position detection with SSI absolute encoder (Page 9061)

Invert direction

You can invert the count direction or position values to adapt them to the process.

Inversion can be configured and become effective for the following signal types:

- Incremental encoder (A, B phase-shifted)
- Incremental encoder (A, B, N)
- Absolute encoder (SSI)

Counter inputs: Additional parameters (Measuring mode)

Signal evaluation

By configuring signal evaluation (Page 9021), you can specify which edges of the signals are counted.

You can select from the following options:

Signal evaluation	Meaning
Single (default)	The edges of signal A are evaluated during a low level of signal B.
Double	Each edge of signal A is evaluated.
Quadruple	Each edge of signals A and B is evaluated.

The parameter can be assigned with the following signal types:

- Incremental encoder (A, B phase-shifted)
- Incremental encoder (A, B, N)

Filter frequency

By configuring the filter frequency, you suppress interferences at the counter inputs A, B and N.

The selected filter frequency is based on a pulse-break ratio of between 40:60 and 60:40. This produces a set minimum pulse/break time. Signal changes with a duration shorter than the minimum pulse/break time are suppressed.

You can select from the following filter frequencies:

Table 11-4 Filter frequency and respective minimum pulse/break time

Filter frequency	Minimum pulse/break time
100 Hz	4.0 ms
200 Hz	2.0 ms
500 Hz	800 µs
1 kHz	400 µs
2 kHz	200 µs
5 kHz	80 µs
10 kHz	40 µs
20 kHz	20 µs
50 kHz	8.0 µs
100 kHz	4.0 µs
200 kHz (default at TM Count)	2.0 µs
500 kHz*	0.8 µs
1 MHz* (default for TM PosInput)	0.4 µs

* only with TM PosInput

Sensor type

By configuring the sensor type, you specify how the counter inputs are switched for the TM Count.

You can select from the following options:

Sensor type	Meaning
Sourcing output (default)	The encoder/sensor activates the inputs A, B and N after 24VDC.
Sinking output	The encoder/sensor activates the inputs A, B and N after M.
Push-pull (sinking and sourcing output)	The encoder/sensor alternately activates the inputs A, B and N after M and 24VDC.

"Push-pull" is usually selected when incremental encoders are used. If using 2-wire sensors, such as light barriers or proximity switches, you need to select the corresponding wiring "sourcing output" or "sinking output".

The data sheet of the encoder includes information on whether your incremental encoder is a push-pull encoder.

Note

If you use a push-pull incremental encoder, you can monitor the encoder signals for wire break.

Reaction to signal N

You use this parameter to specify which reaction is triggered at signal N.

You can select from the following options:

Option	Meaning
No reaction to signal N (default)	The counter is not affected by signal N.
Synchronization at signal N (Page 8995)	The counter is set to the start value at signal N. If you select the function "Enable synchronization at signal N" for a digital input, the synchronization depends on the level at the digital input.

Note

You can only select the reaction at signal N if you have selected the signal type (Page 9055) "Incremental encoder (A, B, N)".

Note

If "Synchronization at signal N" is selected, you can select the function "Enable synchronization at signal N" for a digital input (Page 9058).

Frequency

This parameter is used to define the frequency of the following events:

- Synchronization at signal N
- Synchronization as function of a digital input

You can select from the following options:

Option	Meaning
Once (default)	The counter is only set at the first signal N or the first configured edge of the digital input.
Periodic	The counter is set at each signal N or each configured edge of the digital input.

Behavior of a DI: Function of a DI (Measuring mode)

Setting function of the DI

By configuring a digital input, you specify which functions the digital input triggers at switching.

You can select from the following options:

Function of a digital input	Meaning
Gate start/stop (level-triggered)	The level at the respective digital input opens and closes the hardware gate (Page 8984).
Gate start (edge-triggered)	The configured edge at the respective digital input opens the hardware gate (Page 8984).
Gate stop (edge-triggered)	The configured edge at the respective digital input closes the hardware gate (Page 8984).
Synchronization (Page 8991)	The configured edge at the respective digital input sets the counter to the start value.
Enable synchronization at signal N	The active level at the respective digital input enables synchronization of the counter at signal N (Page 8995).
Digital input without function	No function is assigned to the respective digital input. The signal status of the digital input can be read by the CPU by means of the feedback interface (Page 9077).

Note

With the exception of "Digital input without function", each function can only be used once, and if used for one digital input is not available to the others.

Filter time

By configuring the filter time, you suppress interferences at the digital inputs. Signals with a pulse duration below the configured filter time are suppressed.

You can select from the following filter times:

- None
- 0.05 ms
- 0.1 ms (default)
- 0.4 ms
- 0.8 ms
- 1.6 ms
- 3.2 ms
- 12.8 ms
- 20 ms

Note

If you select the option "None" or "0.05 ms", you have to use shielded cables for connecting the digital inputs.

Note

You can only configure the filter time under "Behavior of DI0" for all digital inputs together.

Behavior of a DI: Function options (Measuring mode)

Certain functions require additional parameters which allow you to specify characteristics in more detail. These can be configured for each of the relevant functions.

Select level

You use this parameter to specify the level at which the digital input is active.

You can select from the following options:

Level	Meaning
Active with high level (default)	The respective digital input is active when it is set.
Active with low level	The respective digital input is active when it is not set.

The parameter can be set for the following functions of a digital input:

- Gate start/stop (level-triggered)
- Enable synchronization at signal N

Edge selection

You can use this parameter to specify the edge of the digital input at which the configured function is triggered.

You can select from the following options:

- At rising edge (default)
- At falling edge

The parameter can be set for the following functions of a digital input:

- Gate start (edge-triggered)
- Gate stop (edge-triggered)
- Synchronization

Function of a DQ (Measuring operating mode)

Set output

By configuring a digital output, you can specify the condition (Page 9006) upon which the digital input switches.

You can select from the following options:

Function of a digital output	Meaning
Measured value >= comparison value (default)	The respective digital output is active if the measured value is greater than or equal to the comparison value.
Measured value <= comparison value	The respective digital output is active if the measured value is smaller than or equal to the comparison value.
Between comparison value 0 and 1	The digital output DQ1 is active if the measured value is between the comparison value 0 and the comparison value 1.
Not between comparison value 0 and 1	The digital output DQ1 is active if the measured value is less than comparison value 0 or greater than comparison value 1.
Use by user program	The respective digital output can be switched by the CPU via the control interface (Page 9074).

Note

You can select the function "Between comparison value 0 and 1" and "Not between comparison value 0 and 1" if the function "Use by user program" is selected for digital output DQ0.

Comparison value 0

By configuring the comparison value (Page 9012), you can specify the measured value at which the digital output DQ0 switches as a result of the selected compare event.

You must enter a value that is smaller than comparison value 1. Up to six decimal places can be entered.

The default setting is "0.000000". The unit of the comparison value depends on the measured variable.

Comparison value 1

By configuring the comparison value (Page 9012), you can specify the measured value at which the digital output DQ1 switches as a result of the selected compare event.

You must enter a value that is greater than comparison value 0. Up to six decimal places can be entered.

The default setting is "10.000000". The unit of the comparison value depends on the measured variable.

Configure position detection with SSI absolute encoder

Overview

When using an SSI absolute encoder, assign the parameters as follows:

- Counter inputs
  - Specifying input signals/encoder type (Page 9062)
  - Additional parameters (Page 9062)
- Behavior of a DI
  - Function of a DI in Counting mode (Page 9064)
  - Function of a DI in Measuring mode (Page 9065)
  - Function options (Page 9065)
- Behavior of a DQ
  - Function of a DQ in Counting mode (Page 9066)
  - Function of a DQ in Measuring mode (Page 9067)
- Hysteresis (Page 9067)
- Specifying the measured value (Page 9068)

You can find two examples of the SSI frame format in the technology module at Examples of the frame format (Page 9069).

Counter inputs: Specifying input signals/encoder type (SSI)

Signal type

If an SSI absolute encoder (Page 8981) is connected, select the signal type "Absolute encoder (SSI)".

Invert direction

You can invert the values delivered by the SSI absolute encoder to adapt them to the process.

Counter inputs: Additional parameters (SSI)

Frame length

By configuring the frame length, you can define the number of bits for an SSI frame. You can find the frame length of your SSI absolute encoder in the data sheet of the encoder. A parity bit does not count in the frame length.

A frame length of between 10 bits and 40 bits is permitted. The default setting is "13 Bit".

Type of code

You use the parameter assignment of the code type to specify whether the encoder supplies Dual code or Gray code.

You can select from the following options:

Type of code	Meaning
Gray (default)	The gray-coded position value returned by the SSI absolute encoder is converted to dual code.
Dual	The value returned by the SSI absolute encoder is not converted.

Transmission rate

By configuring the transmission rate, you can specify the rate of data transmission between the SSI absolute encoder and the technology module. You can select from several options between 125 kHz and 2 MHz. The default setting is "125 kHz".

The maximum transmission rate depends on the cable length and the technical specifications of the SSI absolute encoder. For additional information, please refer to the encoder description.

Monoflop time

By configuring the monoflop time, you can specify the break time between two SSI frames.

The configured monoflop time must be greater than the monoflop time of the SSI absolute encoder used. You can find this value in the technical specifications of the SSI absolute encoder.

You can select from the following options:

- Automatic (default)
- 16 µs
- 32 µs
- 48 µs
- 64 µs

Note

If you use the system function "Isochronous mode", the "Automatic" option corresponds to a Monoflop time of 64 µs.

Parity

When you configure the parity, you specify if a parity bit is to be transferred to the SSI absolute encoder.

If, for example, a 25-bit encoder with parity is assigned, the technology module reads 26 bits. The bit following the LSB (least significant bit) in the SSI frame is evaluated as a parity bit. A parity error is reported in the feedback interface (Page 9077) by means of the ENC_ERROR bit.

LSB bit number of position setpoint

This parameter is used to specify the bit number of the LSB (least significant bit) of the position value in the frame of the SSI absolute encoder. This restricts the resolution of the SSI absolute encoder. The value must be less than the bit number of the MSB of the position value. The difference between the bit numbers of the MSB and the LSB of the position value must be less than 31.

The default setting is "0".

MSB bit number of position setpoint

This parameter is used to specify the bit number of the MSB (most significant bit) of the position value in the frame of the SSI absolute encoder. This restricts the resolution of the SSI absolute encoder. The value must be less than the frame length and higher than the bit number of the LSB of the position value. The difference between the bit numbers of the MSB and the LSB of the position value must be less than 31.

The default setting is "12".

Note

If you have selected the Code type "Gray", only the values in the range between LSB and MSB of the position value are converted to dual code.

Behavior of a DI: Function of a DI (SSI, Measuring mode)

Setting function of the DI

By configuring a digital input, you specify which functions the digital input triggers at switching.

You can select from the following options:

Function of a digital input	Meaning
Capture (Page 8989)	The configured edge at the respective digital input saves the current position value as a Capture value. The function can only be used for a digital input.
Digital input without function	No function is assigned to the respective digital input. The signal status of the digital input can be read by the CPU by means of the feedback interface (Page 9077).

Filter time

By configuring the filter time, you suppress interferences at the digital inputs. Signals with a pulse duration below the configured filter time are suppressed.

You can select from the following filter times:

- None
- 0.05 ms
- 0.1 ms (default)
- 0.4 ms
- 0.8 ms
- 1.6 ms
- 3.2 ms
- 12.8 ms
- 20 ms

Note

If you select the "None" or "0.05 ms" option, you have to use shielded cables for connection of the digital inputs.

Note

You can only configure the filter time under "Behavior of DI0" for all digital inputs together.

Behavior of a DI: Function of a DI (SSI, Measuring mode)

Setting function of the DI

"Digital input without function" is assigned to the respective digital input. The signal status of the digital input can be read by the CPU by means of the feedback interface (Page 9077).

Filter time

By configuring the filter time, you suppress interferences at the digital inputs. Signals with a pulse duration below the configured filter time are suppressed.

You can select from the following filter times:

- None
- 0.05 ms
- 0.1 ms (default)
- 0.4 ms
- 0.8 ms
- 1.6 ms
- 3.2 ms
- 12.8 ms
- 20 ms

Note

If you select the "None" or "0.05 ms" option, you have to use shielded cables for connection of the digital inputs.

Note

You can only configure the filter time under "Behavior of DI0" for all digital inputs together.

Behavior of a DI: Function options (SSI)

Edge selection

You can use this parameter to specify the edge of the digital input at which the configured function is triggered for the "Capture" function.

You can select from the following options:

- At rising edge (default)
- At falling edge
- At rising and falling edge

Behavior of a DQ: Function of a DQ (SSI, Measuring mode)

Set output

By configuring a digital output, you can specify the condition (Page 9003) upon which the digital output switches.

You can select from the following options:

Function of a digital output	Meaning
Between comparison value and high limit (default)	The respective digital output is active if the position value is between the comparison value and the high limit.
Between comparison value and low counter limit	The respective digital output is active if the position value is between "0" and the comparison value.
Between comparison value 0 and 1	The digital output DQ1 is active if the position value is between comparison value 0 and comparison value 1.
At comparison value for a pulse duration	The respective digital output is active once for the assigned time and direction of the position value change when the position value is equal to the comparison value or has fallen below or exceeded it.
After set command from CPU until comparison value	When a set command is sent from the CPU, the respective digital output is active for the assigned direction of the position value change until the position value is equal to the comparison value or has fallen below or exceeded it.
Use by user program	The respective digital output can be switched by the CPU via the control interface (Page 9074).

Note

You can select the function "Between comparison value 0 and 1" if the function "Use by user program" is selected for digital output DQ0.

Comparison value 0

By configuring the comparison value, you specify the position value at which the digital output DQ0 is to switch as a result of the selected comparison event.

You must enter a value that is less than comparison value 1.

The default setting is "0".

Comparison value 1

By configuring the comparison value, you specify the position value at which the digital output DQ1 is to switch as a result of the selected comparison event.

You must enter a value that is greater than comparison value 0.

The default setting is "10".

Count direction

You use this parameter to specify the direction of position value change for which the selected function is valid.

You can select from the following options:

Direction of position value change	Meaning
In both directions (default)	The comparison and switching of the respective digital output is carried out regardless of whether the position value becomes larger or smaller.
Up	The comparison and switching of the respective digital output only takes place when the position value becomes larger.
Down	The comparison and switching of the respective digital output only takes place when the position value becomes smaller.

The parameter can be configured for the following functions:

- At comparison value for a pulse duration
- After set command from CPU until comparison value

Pulse duration

By configuring the pulse duration for the function "At comparison value for a pulse duration", you specify the number of milliseconds for which the respective digital output is active.

You can enter a value between 0.5 and 6553.5 s.

The default setting is "500.0", which is equivalent to a pulse duration of 0.5 s.

Behavior of a DQ: Function of a DQ (SSI, Measuring mode)

In measuring mode, you specify the parameters described in the section Function of a DQ (Measuring operating mode) (Page 9060) for the digital outputs of the channel.

Hysteresis (SSI)

Set hysteresis range

By configuring the hysteresis (Page 9015), you can define a range around the comparison values. Within the hysteresis range, the digital outputs cannot switch again until the position value has left this range once.

If you enter "0", the hysteresis is turned off. You can enter a value between 0 and 255. The default setting is "0".

Hysteresis can only be configured in Counting mode.

Specifying measured value (SSI)

Measured variable

This parameter is used to specify the measured variable (Page 9012) to be provided by the technology module.

You can select from the following options:

Measured variable	Meaning
Frequency (default)	The measured variable is a frequency. The unit is Hz. The incremental changes of the position value are measured.
Period	The measured variable is a period of time corresponding to the reciprocal value of the frequency. The unit is s.
Velocity	The measured variable is a velocity. You configure the time basis for the velocity measurement and the Increments per unit separately.
Complete SSI frame	The first 32 bits of the SSI frame are returned instead of a measured variable. Extra bits that do not belong to the actual position information are supplied in this case. A configured inversion of the direction is ignored.

Update time

By configuring the update time (Page 9012) in milliseconds, you can specify the time interval between two measured value updates.

The update time influences the accuracy of the measurement.

If you enter "0", the measured value is updated in each cycle. Up to three decimal places can be entered. You can enter a value between 0.000 and 25000.000. The default setting is "10.000".

Time base for velocity measurement

This parameter defines the time base on which the velocity is to be returned.

You can select from the following options:

- 1 ms
- 10 ms
- 100 ms
- 1 s
- 60 s/1 min

The default setting is "60 s/1 min".

Increments per unit

This parameter defines the number of increments the SSI absolute encoder delivers per revolution.

You can enter a value between 1 and 65535.

Example:

Your absolute encoder operates with a resolution of 12 bits per revolution and performs 4096 increments per revolution. The velocity should be measured in revolutions per minute.

In this case, you need to assign the following parameters:

- Increments per unit: 4096
- Time base for velocity measurement: 60 s/1 min

Examples of the frame format

Example 1

In this example the SSI absolute encoder has the following specification:

- The SSI frame has a length of 25 bits.
- The MSB of the position value is bit 24.
- The LSB of the position value is bit 0.
- The position value is Gray coded.
- A parity bit is not available.

The frame has the following format:

Cycle:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Bit:	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MG	MG	MG	MG	MG	MG	MG	MG	MG	MG	MG	MG	SG	SG	SG	SG	SG	SG	SG	SG	SG	SG	MG	SG	SG
	MSB																								LSB

Position value

MG Multiturn bit as Gray code

SG Singleturn bit as Gray code

Complete SSI frame

If you want to obtain the unedited SSI frame instead of a measured variable, it is returned right-justified in the feedback interface:

Cycle:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Bit:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	0	MG	MG	MG	MG	MG	MG	MG	MG	MG	MG	MG	MG	SG	SG	SG	SG	SG	SG	SG	SG	SG	SG	SG	SG	SG

MG Multiturn bit as Gray code

SG Singleturn bit as Gray code

Feedback value position value

The position value supplied in Gray code is converted into binary code by the technology module and returned right-justified in the feedback interface:

Cycle:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Bit:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	0	MD	MD	MD	MD	MD	MD	MD	MD	MD	MD	MD	MD	SD	SD	SD	SD	SD	SD	SD	SD	SD	SD	SD	SD	SD

MD	Multiturn bit as binary code
SD	Singleturn bit as binary code

Example 2

In this example the SSI absolute encoder has the following specification:

- The SSI frame has a length of34 bits.
- The MSB of the position value is bit 33.
- The LSB of the position value is bit 6.
- The position value is Gray coded.
- The SSI frame has two special bits.
- A parity bit is available. A parity bit does not count in the frame length.

The frame has the following format:

Cycle:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
Bit:	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	*	3	2	1	0
	MG	MG	MG	MG	MG	MG	MG	MG	MG	MG	MG	SG	SG	SG	SG	SG	SG	SG	SG	SG	SG	SG	MG	SG	SG	SG	SG	SG	S1	S2	P	0	0	0	0
	MSB																											LSB							

Position value

MG	Multiturn bit as Gray code
SG	Singleturn bit as Gray code
S1	Status bit 1
S2	Status bit 2
P	Parity bit

Complete SSI frame

If you want to obtain the unedited SSI frame instead of a measured variable, the technology module reads the least significant 32 bit as maximum from the encoder. The technology module returns the bit following the LSB as parity bit. In this example the technology module therefore reads in the least significant 31 bit of the SSI frame.

The returned complete SSI frame has the following format:

Cycle:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Bit:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MG	MG	MG	MG	MG	MG	MG	MG	SG	SG	SG	SG	SG	SG	SG	SG	SG	SG	SG	SG	SG	SG	MG	SG	SG	P	S1	S2	0	0	0	0

MG Multiturn bit as Gray code

SG Singleturn bit as Gray code

Position value

The position value supplied in Gray code is converted into binary code by the technology module and returned right-justified in the feedback interface:

Cycle:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Bit:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	MD	MD	MD	MD	MD	MD	MD	MD	MD	MD	MD	SD	SD	SD	SD	SD	SD	SD	MD	SD	SD	SD	SD	SD	SD	SD	SD	SD

MD Multiturn bit as binary code

SD Singleturn bit as binary code

Online & diagnostics module

Displaying and evaluating diagnostics

The online and diagnostics view enables hardware diagnostics. You can also

- Obtain information on the technology module (e.g., Firmware version and serial number)
- Execute a firmware update if required

Procedure

To open the display editor for the diagnostic functions, follow these steps:

1. Open the "Local modules" or "Distributed I/O" folder in the project tree.
2. Open the technology module in the project tree.
3. Double-click on the "Online & diagnostics" object.
4. Select the required display in the diagnostics navigation.

Additional information

Additional information on the diagnostic alarms and possible remedies can be found in the technology module device manual.

Note

Position input for Motion Control

In the operating mode "Position input for Motion Control", channel diagnostics is not available for the technology module.

Commissioning the module

The commissioning editor assists you in the commissioning and the function test of the technology module.

You can simulate the count signals and the control of digital input states without process wiring. This allows you to monitor and test the characteristics of the user program technology module during counting.

Note

The digital output functions and hardware interrupts which have been configured are triggered in line with the simulated counter value.

Note

The commissioning editor is not used in the operating mode "Position input for Motion Control".

See also

Assignment of the feedback interface (Page 9077)

Monitoring counter values, measured values, DIs and DQs

Requirements

- There is an online connection between STEP 7 (TIA Portal) and the CPU.
- The CPU is in RUN.

Procedure

To open the commissioning editor of a technology module, follow these steps:

1. Open the "Local modules" or "Distributed I/O" folder in the project tree.
2. Open the technology module in the project tree.
3. Double-click on the "Commissioning" object.

The commissioning editor automatically switches to online mode. The CPU operator panel is displayed on the right in the "Test" task card to control the CPU mode.

Monitor values

The following values are read from the technology module cyclically and displayed under "Monitor values":

- Counter value
- Measured value

- Capture value
- Error code

The following values are read from the technology module once and displayed under "Monitor values":

- Comparison value 0
- Comparison value 1
- High counting limit
- Low counting limit
- Start value

You can toggle between the decimal and hexadecimal format of the displayed values in the "Display format" column. The display format "Floating point number" is also available for the measured value.

Monitor status flags

The status information from the technology module is read and displayed online. When the CPU is in STOP, the status display is not updated.

Monitor DIs & DQs

The technology module digital input and digital output signal states are displayed under "Monitor/control DI & DQ". The function currently configured is displayed in the "Comment" column. You can manually change the values with the control panel above the table and thus simulate the counting and measuring functions: see Manually controlling counter values, DIs and DQs (Page 9073)

Manually controlling counter values, DIs and DQs

With manual control, you can modify the technology module digital input states online and specify a counter value without process wiring. You can use a simulation function and specify a simulation velocity to monitor the effects on the measuring and counting functions. The digital outputs are switched in accordance with the simulated counter value, and the corresponding hardware interrupts are triggered.

Requirements

- There is an online connection between STEP 7 (TIA Portal) and the CPU.
- The CPU is in RUN.
- The commissioning window "Monitor/control DI & DQ" for the technology module is open. The current values of the technology module are displayed.

Manual control

WARNING

Warning of personal injury and material damage

Incorrect manual control can cause serious injury and damage plants and equipment:

- Check all entries before you switch to manual control and before you apply changed values during manual control.
- Manual control may only be carried out by trained personnel.

Basic procedure - manually controlling counter values, DIs and DQs

1. Activate manual control in the operator panel using the "Activate" button.
2. Enter new values in the empty "Control value" column.
   The check box to the right is automatically selected. The yellow warning triangle indicates that this value has not been transmitted to the technology module yet.
3. Enter a value for the "Velocity for simulation". For a TM Count the limits of the value range are ±800000.00. For a TM PosInput, the limits of the value range are ±4000000.00.
   The value determines the speed of the simulated count signals.
4. Apply the modified values with the selected check box by clicking the Change" button.
   The yellow warning triangle goes out. The changed values are updated in the technology module.
5. Control the simulation of the counting or measuring function with the "Start" and "Stop" buttons.
6. You can monitor the displays under "Monitor values", "Monitor status flags" and "Monitor/ control DIs & DQs" and check how your user program responds.
7. If necessary, you can adapt the values in the column "Control value" by repeating steps 2 to 4.

Control and feedback interface

Information on using the control and feedback interface can be found under Overview of application options (Page 8978).

Assignment of the control interface

The user program uses the control interface to influence the behavior of the technology module.

Control interface per channel

The following table shows control interface assignment:

Offset to the start address	Parameter	Meaning				
Bytes 0 to 3	Slot 0	Load value (significance of the value is specified in LD_SLOT_0)				
Bytes 4 to 7	Slot 1	Load value (significance of the value is specified in LD_SLOT_1)				
Byte 8	LD_SLOT_0*	Specifies the significance of the value in Slot 0				
		Bit 3	**Bit 2**	**Bit 1**	**Bit 0**	
		0	0	0	0	No action, idle
		0	0	0	1	Load count value (with incremental or pulse encoder)
		0	0	1	0	Reserve
		0	0	1	1	Load start value (with incremental or pulse encoder)
		0	1	0	0	Load comparison value 0
		0	1	0	1	Load comparison value 1
		0	1	1	0	Load low counter limit (with incremental or pulse encoder)
		0	1	1	1	Load high counter limit (with incremental or pulse encoder)
		1	0	0	0	Reserve
		to				
		1	1	1	1	
	LD_SLOT_1*	Specifies the significance of the value in Slot 1				
		Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	
		0	0	0	0	No action, idle
		0	0	0	1	Load count value (with incremental or pulse encoder)
		0	0	1	0	Reserve
		0	0	1	1	Load start value (with incremental or pulse encoder)
		0	1	0	0	Load comparison value 0
		0	1	0	1	Load comparison value 1
		0	1	1	0	Load low counting limit (with incremental or pulse encoder)
		0	1	1	1	Load high counting limit (with incremental or pulse encoder)
		1	0	0	0	Reserve
		to				
		1	1	1	1	
Byte 9	EN_CAPTURE	Bit 7: Capture function enable				
	EN_SYNC_DN	Bit 6: Enable downward synchronization (with incremental or pulse encoder)				
	EN_SYNC_UP	Bit 5: Enable upward synchronization (with incremental or pulse encoder)				
	SET_DQ1	Bit 4: Set DQ1				

Offset to the start address	Parameter	Meaning
	SET_DQ0	Bit 3: Set DQ0
	TM_CTRL_DQ1	Bit 2: enable technological function DQ1
	TM_CTRL_DQ0	Bit 1: Enable technological function DQ0
	SW_GATE	Bit 0: Software gate (with incremental or pulse encoder)
Byte 10	SET_DIR	Bit 7: Count direction (for encoders without direction signal)
	–	Bits 2 to 6: Reserve; bits must be set to 0
	RES_EVENT	Bit 1: Reset of saved events
	RES_ERROR	Bit 0: Reset of saved error states
Byte 11	–	Bits 0 to 7: Reserve; bits must be set to 0

* If values are loaded simultaneously via LD_SLOT_0 and LD_SLOT_1, the first value is taken internally from Slot 0 and then the value from Slot 1 is taken. This may lead to unexpected intermediate states.

Notes on the control bits

Control bit	Notes
EN_CAPTURE	Use this bit to enable the Capture function. Resetting this bit resets a set EVENT_CAP in the feedback interface.
EN_SYNC_DN	Use this bit to enable the synchronization of the counter when counting in downward direction with an incremental encoder or pulse encoder. Resetting this bit resets a set EVENT_SYNC in the feedback interface.
EN_SYNC_UP	Use this bit to enable the synchronization of the counter when counting in upward direction with an incremental encoder or pulse encoder. Resetting this bit resets a set EVENT_SYNC in the feedback interface.
RES_EVENT	Use this bit to trigger the reset of the saved events in the EVENT_ZERO, EVENT_OFLW, EVENT_UFLW, EVENT_CMP0, EVENT_CMP1 feedback bits.
RES_ERROR	Use this bit to trigger the reset of the saved error states LD_ERROR and ENC_ERROR .
SET_DIR	Use this bit to specify the count direction for signal type "Pulse (A)". 0 means: Up 1 means: Down
SET_DQ0	Use this bit to set digital output DQ0 when TM_CTRL_DQ0 is set to 0. In the case of the function "After set command from CPU until comparison value",SET_DQ0 is effective regardless of TM_CTRL_DQ0 as long as the counter value is not equal to the comparison value.
SET_DQ1	Use this bit to set digital output DQ1 when TM_CTRL_DQ1 is set to 0. In the case of the function "After set command from CPU until comparison value",SET_DQ1 is effective regardless of TM_CTRL_DQ1 as long as the counter value is not equal to the comparison value.
SW_GATE	Use this bit to open and close the software gate when using an incremental encoder or pulse encoder. Together, the software gate and the hardware gate form the internal gate. The technology module only counts when the internal gate is open. 0 means: Software gate closed 1 means: Software gate open

Control bit	Notes
TM_CTRL_DQ0	Use this bit to enable the technological function of digital output DQ0. 0 means: SET_DQ0 defines the state of DQ0 1 means: assigned function defines the state of DQ0
TM_CTRL_DQ1	Use this bit to enable the technological function of digital output DQ1. 0 means: SET_DQ1 defines the state of DQ1 1 means: assigned function defines the state of DQ1

Assignment of the feedback interface

The user program receives current values and status information from the technology module via the feedback interface.

Feedback interface per channel

The following table shows the assignment of the feedback interface:

Offset to the start address	Parameter	Meaning
Bytes 0 to 3	COUNT VALUE	Current counter value or position value
Bytes 4 to 7	CAPTURED VALUE	The last acquired Capture value
Bytes 8 to 11	MEASURED VALUE	Current measured value or complete SSI frame
Byte 12	–	Bits 3 to 7: Reserve; set to 0
	LD_ERROR	Bit 2: Error when loading via control interface
	ENC_ERROR	Bit 1: Faulty encoder signal or SSI frame
	POWER_ERROR	Bit 0: S7-1500 module: Incorrect supply voltage L+ / ET 200SP module: Supply voltage too low L+
Byte 13	–	Bits 6 to 7: Reserve; set to 0
	STS_SW_GATE	Bit 5: Software gate status (with incremental or pulse encoder)
	STS_READY	Bit 4: Technology module started up and configured
	LD_STS_SLOT_1	Bit 3: Load request for Slot 1 detected and executed (toggling)
	LD_STS_SLOT_0	Bit 2: Load request for Slot 0 detected and executed (toggling)
	RES_EVENT_ACK	Bit 1: Reset of event bits active
	–	Bit 0: Reserve; set to 0
Byte 14	STS_DI2 (TM Count)	Bit 7: TM Count: Status DI2 / TM PosInput: Reserve; set to 0
	STS_DI1	Bit 6: Status DI1
	STS_DI0	Bit 5: Status DI0
	STS_DQ1	Bit 4: Status DQ1
	STS_DQ0	Bit 3: Status DQ0
	STS_GATE	Bit 2: Internal gate status (with incremental or pulse encoder)
	STS_CNT	Bit 1: Count pulse or position value change detected within the last ca. 0.5 s
	STS_DIR	Bit 0: Direction of last counter value or position value change
Byte 15	STS_M_INTERVAL	Bit 7: Count pulse or position value change detected in previous measurement interval
	EVENT_CAP	Bit 6: Capture event has occurred

Offset to the start address	Parameter	Meaning
	EVENT_SYNC	Bit 5: Synchronization has occurred (with incremental or pulse encoder)
	EVENT_CMP1	Bit 4: Comparison event for DQ1 has occurred
	EVENT_CMP0	Bit 3: Comparison event for DQ0 has occurred
	EVENT_OFLW	Bit 2: An overflow has occurred
	EVENT_UFLW	Bit 1: An underflow has occurred
	EVENT_ZERO	Bit 0: A zero pass has occurred

Notes on the feedback bits

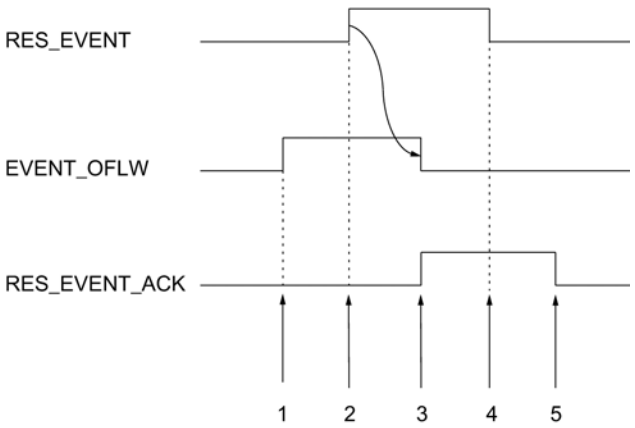
Feedback bit	Notes
ENC_ERROR	This bit indicates that one of the following errors has occurred at the encoder signals (retentive) for the respective technology module: TM Count: • Wire break of digital input A, B, or N (with push-pull encoder) • Invalid transition of A/B signals (with incremental encoder) TM PosInput: • Invalid transition of A/B signals (with incremental encoder) • RS422/TTL error • SSI encoder error or SSI frame error (with SSI absolute encoder) If you have enabled the diagnostic interrupts, the respective diagnostic interrupt is triggered in the event of encoder signal errors. For information on the meaning of the diagnostic interrupts, refer to the manual for the respective technology module. The bit is reset once you have acknowledged the error with RES_ERROR .
EVENT_CAP	This bit indicates that a Capture event has occurred and a counter value has been saved in CAPTURED VALUE . You reset the status by resetting EN_CAPTURE .
EVENT_CMP0	This bit indicates the saved status that a comparison event has occurred for the digital output DQ0. You reset the status by acknowledgment with RES_EVENT. The EVENT_CMP0 bit is not set when you set the counter value to the start value.
EVENT_CMP1	This bit indicates the saved status that a comparison event has occurred for the digital output DQ1. You reset the status by acknowledgment with RES_EVENT. The EVENT_CMP1 bit is not set when you set the counter value to the start value.
EVENT_OFLW	This bit indicates the saved state which shows that there was a counter value overflow. You reset the status by acknowledgment with RES_EVENT.
EVENT_SYNC	This bit indicates the saved status for which the counter was loaded with the start value by an external reference signal (synchronization) when using an incremental or pulse encoder. You reset the state by resetting EN_SYNC_UP or EN_SYNC_DN .
EVENT_UFLW	This bit indicates the saved state which shows that there was a counter value underflow. You reset the status by acknowledgment with RES_EVENT.
EVENT_ZERO	This bit indicates the saved status with which the counter value or position value experienced a zero crossing. You reset the status by acknowledgment with RES_EVENT.

Feedback bit	Notes
LD_ERROR	This bit indicates that an error occurred (latching) during loading via the control interface. The loaded values were not applied. When using an incremental or pulse encoder, one of the following conditions is not fulfilled: • Low counting limit <= counter value <= high counting limit • Low counting limit <= start value <= high counting limit • Low counting limit <= comparison value 0/1 <= high counting limit • Comparison value 0 < comparison value 1 When using an SSI absolute encoder, one of the following conditions is not fulfilled: • 0 <= position value <= maximum position value • 0 <= comparison value 0/1 <= maximum position value • Comparison value 0 < comparison value 1 The bit is reset once you have acknowledged the error with RES_ERROR .
LD_STS_SLOT_0	This bit indicates by a status change (toggling) that the load request for Slot 0 (LD_SLOT_0) has been detected and executed.
LD_STS_SLOT_1	This bit indicates by a status change (toggling) that the load request for Slot 1 (LD_SLOT_1) has been detected and executed.
POWER_ERROR	For an S7-1500 technology module, this bit indicates that the supply voltage L+ is not available or too low or that the front plug is not plugged in. For an ET 200SP technology module, this bit indicates that the supply voltage L+ is too low. If the supply voltage L+ is available at a sufficient level once again, POWER_ERROR is automatically set to 0.
RES_EVENT_ACK	This bit indicates that the reset of event bit EVENT_SYNC, EVENT_CMP0, EVENT_CMP1, EVENT_OFLW, EVENT_UFLW, EVENT_ZERO is active.
STS_CNT	This bit indicates that at least one count pulse or a position value change has occurred in the last ca. 0.5 s.
STS_DI0	This bit indicates the status of digital input DI0.
STS_DI1	This bit indicates the status of digital input DI1.
STS_DI2	This bit indicates the status of digital input DI2 of the TM Count .
STS_DIR	This bit indicates the count direction of the last count pulse or the direction of the last position value change. 0 means: Down 1 means: Up
STS_DQ0	This bit indicates the status of digital output DQ0.
STS_DQ1	This bit indicates the status of digital output DQ1.
STS_GATE	This bit indicates the status of the internal gate when using an incremental or pulse encoder. 0 means: Gate closed 1 means: Gate open
STS_M_INTERVAL	This bit indicates that at least one count pulse or a position value change was detected in the previous measurement interval.
STS_READY	This bit indicates that the technology module supplies valid user data. The technology module has been started up and configured.
STS_SW_GATE	This bit indicates the status of the software gate. 0 means: Gate closed 1 means: Gate open

Complete acknowledgment principle

Saving bits are acknowledged according to the complete acknowledgment principle.

The figure below shows an example of the sequence of the complete acknowledgment principle in the event of an overflow:



① The EVENT_OFLW feedback bit is set as a saving event upon overflow.
② You set the RES_EVENT control bit to trigger EVENT_OFLW reset.
③ The RES_EVENT_ACK feedback bit is set when reset of EVENT_OFLW is detected.
④ You then rest the control bit RES_EVENT .
⑤ The RES_EVENT_ACK feedback bit is reset.

11.1.2 Counting and measurement (S7-300, S7-400)

11.1.2.1 Counting and measuring

Counting and measuring basics

What is counting and measuring?

Counting and measuring

Counting is the capturing and totalizing of events. The counter modules detect and analyze encoder signals (pulses).

The pulses are detected in real time in measuring modes and used to calculate the relevant measured values (e.g. frequency).

Counting basics

Counting ranges and count limits

The module counts within various counting limits in the two counting ranges "0 to +32 bits" and "-31 to +31 bits". An high limit violated or an low limit violated is detected at the respective count limits.

In the "-31 to +31 bits" counting range, the counter value is represented in 2's complement.

Counting range		High limit violated	Low limit violated
0 to +32 bits [1]	0 to 4,294,967,295 0 to FFFF FFFFH	When the counter value changes from 4 294 967 295 to 0	When the counter value changes from 0 to 4 294 967 295
-31 to +31 bits	-2,147,483,648 to 2,147,483,647 8000 0000H to 7FFF FFFFH	When the counter value changes from +2 147 483 647 to –2 147 483 648	When the counter value changes from -2,147,483,648 to 2,147,483,647
[1] This counting range is only available with FM x50-1. You can only specify and evaluate hexadecimal values.			

Load value

The load value is the counter value from which the module starts the counting process.

You can assign a load value LOAD_VAL to the module during operation. This will overwrite the initial value.

You can assign this load value **directly**. It will then be accepted directly by the module as a new counter value and loaded in preparation.

You can only load the load value **in preparation**. A load value loaded in preparation is accepted by the module as the new counter value in response to the following events:

- In all Count modes
  - The counting process is started by the interrupting software or hardware gate (the counter continues counting from the current counter value when the software or hardware gate is started if the interrupting gate function is activated; the load value is not applied).
  - Synchronization (via corresponding DI)
  - Latch/Retrigger (via corresponding DI)
- In the "Count once" and " Count periodically" modes also:
  - When the set high count limit is reached for main counting direction up.
  - Reaching 0 when counting down.

Gate control

You can use the hardware gate and software gate to control i.e. start and stop the module's counting processes.

Main counting direction

When you set a main counting direction (up or down), you can limit the maximum count range to a smaller range by setting the high count limit. The set count range is then between 0 and the set high count limit. This can be used to create incrementing or decrementing counting applications, for example. The set main counting direction has no effect on the direction evaluation when the count pulses are detected.

Starting Counts According to Parameterization

Value	Main counting direction	Start value
Load value	None Up Down	0 0 Parameterized high count limit
Count value	None Up Down	0 0 Parameterized high count limit
Comparison value 1 and 2	None Up Down	0 0 Parameterized high count limit
Latch value	None Up Down	0 0 Parameterized high count limit

Commands for the Counting Modes

You can apply commands to the module counting process during operation:

Command	Description
Open and close gate	The counting process starts when a gate opens and ends when it closes.
Set counter [1])	The counter will be set to the load value via various signals (DI, zero mark).
Latch/Retrigger [2])	Saves the counter level and loads the counter with the load value in response to a positive edge at DI Start.
Latching [2])	Saves the counter level in response to a positive edge at DI.Start.
Measuring times between two edges[3])	Measures the times between two immediately successive edges at the DI Start digital input.
[1]) only with FM 350-1 and FM 450-1 [2]) only with FM 350-1, ET 200S 1Count24V and ET 200S 1Count5V [3]) only with FM 350-1	

Isochronous mode

In isochrone mode, the module accepts control bits and control values from the control interface in each PROFIBUS / PROFINET cycle and returns its response to them within the same cycle.

In each PROFIBUS / PROFINET cycle, the module transfers the counter and latch value that were valid at time T_i and the status bits valid at time T_i.

A counter value influenced by hardware input signals will only be transferred in the same cycle if the input signal occurs before time T_i.

Count modes

Overview of the count modes

You select module functionality by setting a default operating mode. The table below gives an overview of the counting modes.

Operating mode	Description
Count continuously (Page 9083) (with or without software or hardware gate)	The module counts continuously from the current counter value.
Count once (Page 9084) (with software gate or hardware gate)	When the gate opens, the module counts from the load value to the count limit.
Count periodically (Page 9085) (with software gate or hardware gate)	When the gate opens, the module counts between the load value and the count limit.

Count continuously

Description

In this operating mode, the module counts continuously from the counter value:

- If the counter reaches the high limit when counting up and a further count pulse is received, it jumps to the low count limit and continues to count from there without any pulse losses.
- If the counter reaches the low limit when counting down and a further count pulse is received, it jumps to the high limit and continues to count from there without any pulse losses.

The following applies to the ±31 bit counting range:

- The high count limit is set to +2,147,483,647 (2^{31} - 1).
- The low count limit is set to -2,147,483,648 (-2^{31}).

The following applies to the 32 bit counting range:

- The high count limit is set at +4,294,967,295 (2^{32} - 1).
- The low count limit is set to 0 (zero).

Behavior at the Count Limits

If the counter reaches the high or low count limit and a further count pulse is received, then the counter is set to the other count limit. An appropriate status bit is set in the feedback interface and if applicable in the DB:

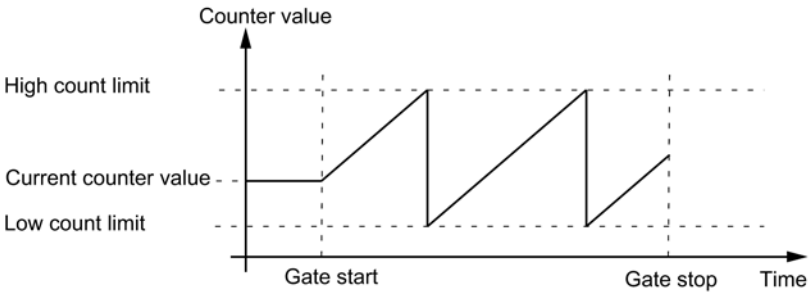
Count limit reached	Status bit
High count limit	STS_OFLW is set
Low count limit	STS_UFLW is set

Selecting gate control

This mode allows you to select gate control. The following options are available to you:

- Without gate (default)
- Software gate (Page 9102)
- Hardware gate (Page 9102), level-controlled or edge-controlled

The figure below shows an example of continuous counting with gate control.



Count once

Description

In this operating mode, the module counts once in the parameterized main counting direction and then stops the counting process automatically.

Behavior at the Count Limits

If the counter has reached the high or low count limit and a further count pulse is received, the counter is set

- to the other count limit when counting without a main count direction (Page 9086)
- to the load value when counting with a main count direction (Page 9086).

The gate is then closed and the counting process terminated, even if the parameter / the control bitSW_GATE is still set or the hardware gate is still open. The appropriate status bit is set in the feedback interface / in the DB:

Count limit reached	Status bit
High count limit	STS_OFLW is set
Low count limit	STS_UFLW is set

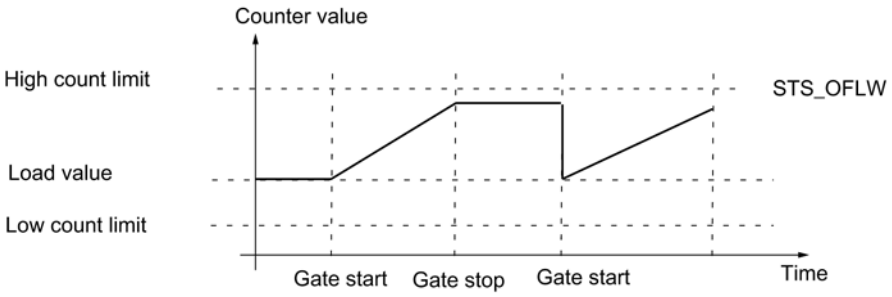
If you want to restart the counter, you must reset the parameter / the control bit SW_GATE and/or reopen the hardware gate. The counting process is then continued from the load value.

Selecting gate control

This mode allows you to select gate control. The following options are available to you:

- Software gate (Page 9102)
- Hardware gate (Page 9102), level-controlled or edge-controlled

The figure below shows an example of continuous counting with load value and gate control.



Count periodically

Description

In this operating mode, the module counts periodically when the gate is open and does not stop counting at the count limits.

Behavior at the Count Limits

If the counter

- has reached one of the count limits when counting without a main count direction (Page 9086)
- has reached the high count limit when counting with the main count direction up (Page 9086)
- has reached the low count limit when counting with main count direction down (Page 9086)

The gate is then closed and the count is terminated even if the SW_GATE parameter is still set or the hardware gate is still open. The appropriate status bit is set in the feedback interface / in the DB.

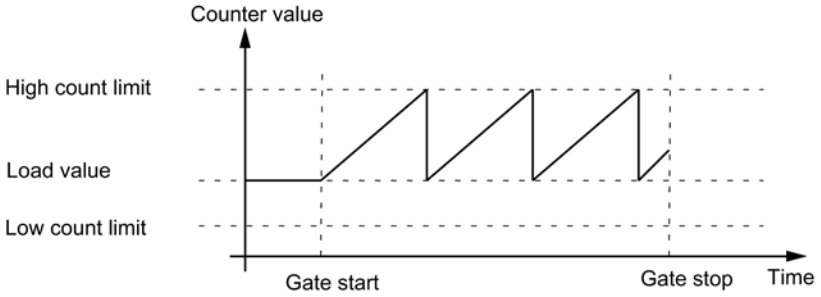
Count limit reached	Status bit
High count limit	STS_OFLW is set
Low count limit	STS_UFLW is set

Selecting gate control

This mode allows you to select gate control. The following options are available to you:

- Software gate (Page 9102)
- Hardware gate (Page 9102), level-controlled or edge-controlled

The figure below shows an example of count periodically with load value and gate control.



Main count direction

A main count direction can only be selected in the "Count once", " Count periodically" and "Dosing" modes.

You can parameterize the following behavior:

- None (no main count direction)
- Up
- Down

Count once without main count direction

When the gate is opened in "Count once" mode without a main count direction, the module will count up or down from the load value until one of the count limits is exceeded.

If one of the count limits is exceeded,

- the gate is closed,
- the STS_OFLW or STS_UFLW bit is set in the feedback interface,
- the counter is set to the other count limit.

The count limits are fixed at the maximum count range.

The STS_ZERO and/or STS_ND bit is set if the counter value is zero (not in FM 350-2).

You must open the gate again to restart the count.

Count Periodically without Main Counting Direction

When the gate is opened in " Count periodically" mode without a main count direction, the module counts up or down from the load value until one of the count limits is exceeded.

If one of the count limits is exceeded

- the STS_OFLW or STS_UFLW bit is set in the feedback interface,
- the counter is set to the load value from which it resumes counting.

The count limits are fixed at the maximum count range.

The STS_ZERO and/or STS_ND bit is set if the counter value is zero (not in FM 350-2).

The counting is resumed when the gate is closed.

Count once with main count direction up

When the gate is opened in "Count once" or "Dosing" mode with the main count direction up, the module counts up or down from the load value until the high count limit is exceeded.

If the high count limit is exceeded,

- the gate is closed,
- the STS_OFLW bit in the feedback interface is set,
- the counter is set to the load value.

The high count limit can be set. The load value has a starting count and can be changed.

You must open the gate again to restart the count.

Count Periodically with Main Counting Direction Up

When the gate is opened in " Count periodically" mode with main count direction up, the module counts up or down from the load value until the high count limit is exceeded.

If the high count limit is exceeded,

- the STS_OFLW bit in the feedback interface is set,
- the counter is set to the load value from which it resumes counting.

The high count limit can be set. The load value has a starting count and can be changed.

The counting is resumed when the gate is closed.

Count once with main count direction down

When the gate is opened in "Count once" or "Dosing" mode with main count direction down, the module counts up or down from the load value until the low count limit is violated.

If the low count limit is under-run,

- the gate is closed,
- the STS_UFLW bit in the feedback interface is set,
- the counter is set to the load value.

The low count limit is fixed at 0; the load value has an initial value and can be changed.

You must open the gate again to restart the count.

Count Periodically with Main Counting Direction Down

When the gate is opened in " Count periodically" mode with main count direction down, the module counts up or down from the load value until the low count limit is violated.

If the low count limit is under-run,

- the STS_UFLW bit in the feedback interface is set,
- the counter is set to the load value from which it resumes counting.

The low count limit is fixed at 0; the load value has an initial value and can be changed.

The counting is resumed when the gate is closed.

Measuring basics

Measuring ranges

The modules have the following measuring ranges in measuring modes:

Operating mode	Measuring range
Frequency measurement	0.1 Hz [1] to 650 000 Hz (depending on the module)
Rotational speed measurement	Up to 25,000 /min
Period measurement	Up to 120 s

[1] The low limit of 0.1 Hz is defined by the maximum update time of 10 s. The resolution of the value is 0.001 Hz.

Update time / Integration time

The module updates the measured values cyclically. Set the "Measuring time" parameter as factor n to calculate the update time / integration time.

Calculation of the update time / integration time:

Boundary conditions		Update time / Integration time	Value range of n	
			n_{min}	n_{max}
Non-isochronous mode	Any T_{DP}	n × 10 ms	1	1000 [3]
Isochronous mode [2]	T_{DP} < 10 ms	n × T_{DP}	(10 ms/T_{DP} [ms]) + 1 [1]	1000 [3]
	T_{DP} ≥ 10 ms	n x T_{DP}	1	10000 [3] ms/T_{DP} [ms] [1]

[1] Any decimal places that result from dividing by T_{DP} are omitted.

It is prohibited to exceed these limits. If these limits are violated, the module will generate a parameter assignment error and will not go into isochrone mode.

[2] not with FM 350-2

[3] for period measurement: 120000

The measured value is updated in the feedback interface and the STS_CMP1 or STS_COMP1 bit is set at the end of update time / integration time.

Measuring principle

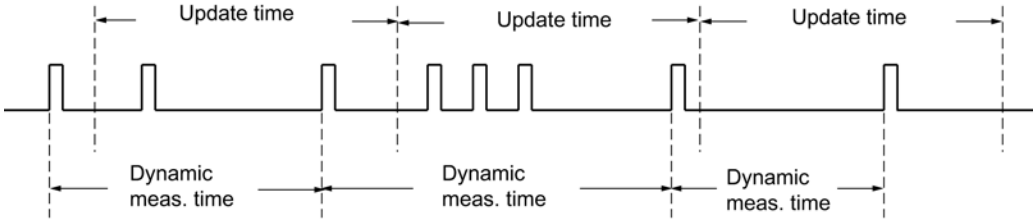
The module counts each positive edge of a pulse and allocates it a time value in µs.

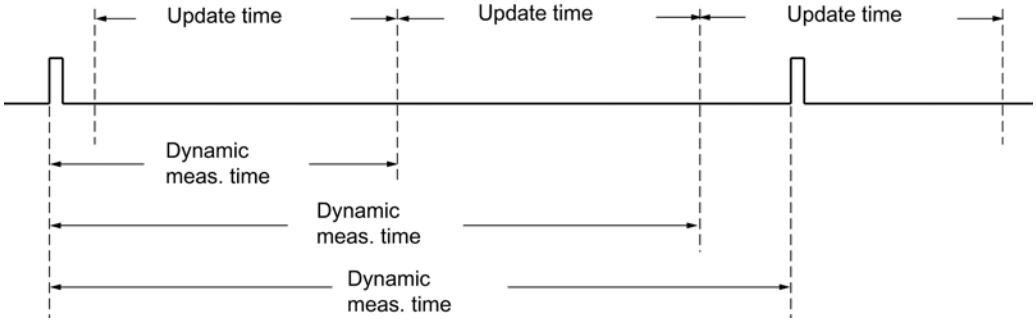
The dynamic measuring time is defined as the difference between two time values.

At a pulse sequence with one or several pulses per update interval:

Dynamic measuring time = Time value of last pulse in the current update time interval
minus
time value of the last pulse in the previous update interval

If no pulses are received within the next update intervals after the dynamic measuring time is calculated, the measuring time is extended by these update intervals. If the value "1 pulse per dynamic measuring time" is less than the last measured value, it will be output as the new value. The figure below shows the measuring principle.





Measurement sequence

The module measures continuously. You define a specific update time in the parameter settings.

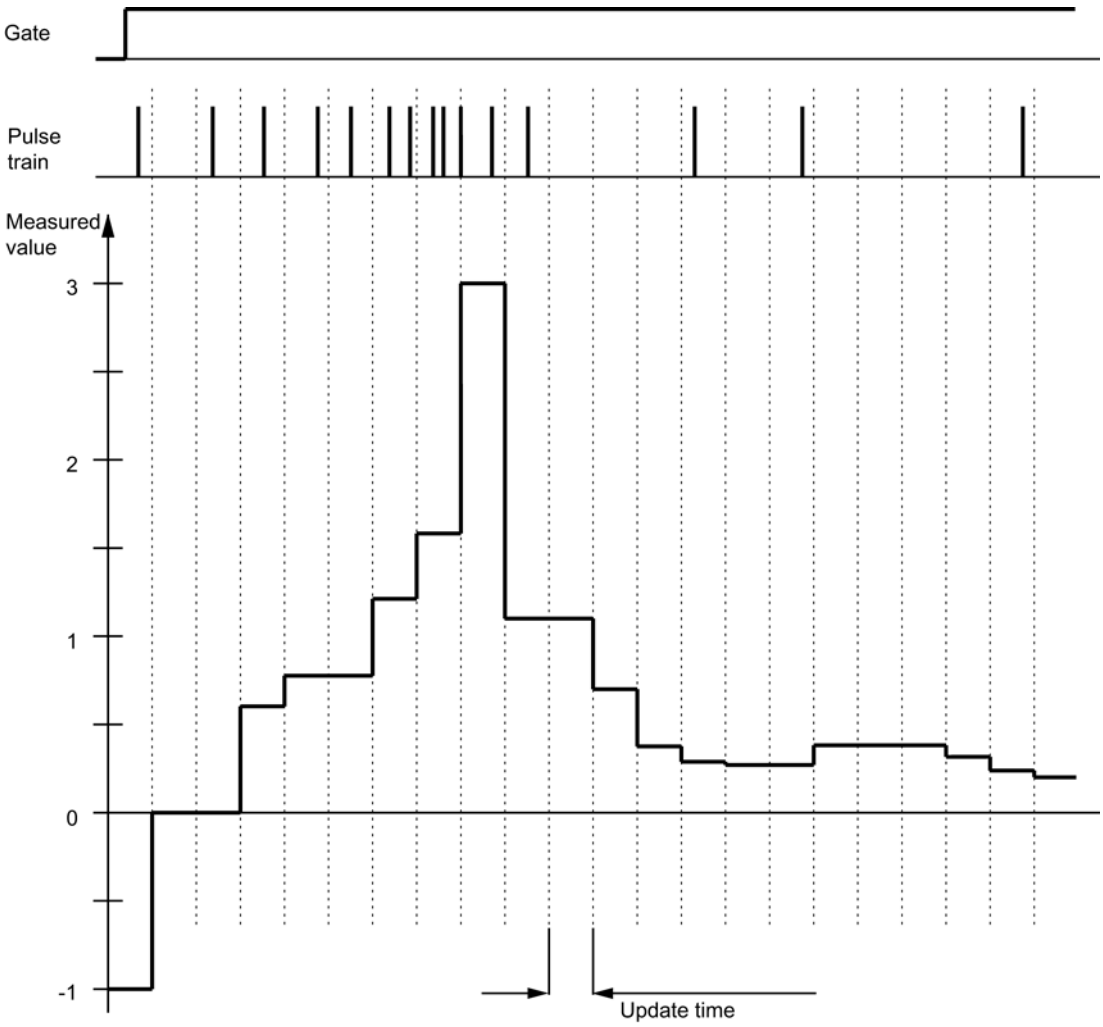
The module returns the value "-1" until the first update time has expired. The first update time starts when the gate has opened.

When the gate has opened, continuous measurement starts at the first pulse of the pulse sequence to measure. The first measured value can be calculated after the second pulse at the earliest.

On each expiration of the update time, a measured value is output at the checkback interface (frequency, period or rpm.) The end of measurement is reported at the STS_CMP1 or STS_COMP1 (for FM 350-2) status bits. This bit is reset by the RES_ZERO and STS_RES_ZERO or RES_STS and RES_STS_A bits by applying the complete acknowledgement principle.

If the direction of rotation is reversed during an update time, the measured value for this measurement period will be undefined. You can respond to any irregularities in the process by evaluating the feedback bits for direction.

The following figure illustrates the principle of continuous measurement taking frequency measurement as an example.



Limit Monitoring

The limit values must meet each of the following conditions:

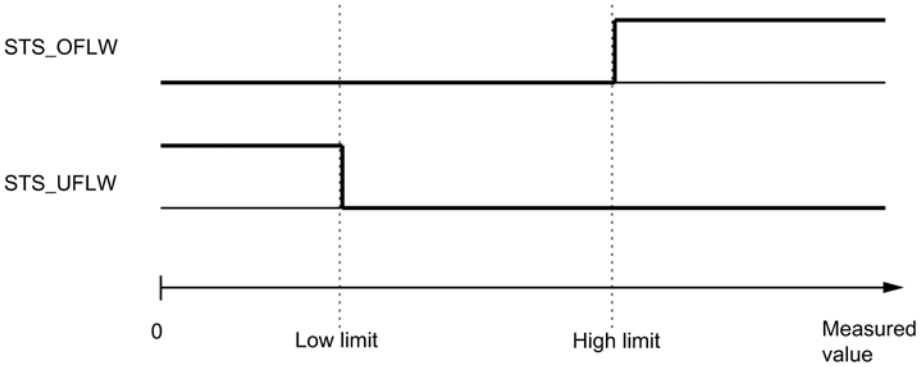
- Low limit value ≥ low measuring range limit
- High limit value ≥ high measuring range limit
- High limit value ≥ (low limit value + 1)

Upon expiry of the update time, the module compares the measured value (frequency, speed or period) with the low and high limit values parameterized. The result of the comparison is as follows.

If the measured value...	...the following bit will be set in the status:
is greater than the high limit value	STS_OFLW
is smaller than the low limit value	STS_UFLW

A hardware interrupt may also be generated.

The image below shows the value of the status bits for limit monitoring in measuring modes.



Reset the STS_OFLW and STS_UFLW bits by setting the RES_ZERO and STS_RES_ZERO or RES_STS bits by applying the complete acknowledgement principle. The module sets the status bits again if it detects high limit violated of the measured value after the acknowledgement bits were set.

If parameters are assigned accordingly, you also can use limit monitoring to set output DOn.

Gate control

You can use the hardware gate and software gate to control i.e. start and stop the module's measuring processes.

Commands in the measuring modes

You can apply commands to the module measuring process during operation:

Designation	Description
Open and close gate	You start the measurement by opening a gate, and stop it by closing this gate.

Isochronous mode

When operating in isochrone mode, the module accepts the control signals output by the control interface at time T_o in each PROFIBUS / PROFINET cycle. As a result, all control operations are executed in isochronous mode and take effect at the time T_o. The response to the control signals is reported within the same PROFIBUS / PROFINET cycle.

The module returns a measured value and the status bits at time T_i in each PROFIBUS / PROFINET cycle.

Each measurement starts and ends at the time T_i.

Note

As you define the update time as an integer multiple of 10 ms for operation in non-isochrone mode and in integer multiples of the bus cycle time for operation in isochrone mode, you should also always adapt the update time parameter when you toggle these modes if you wish to retain the actual update time.

Measuring modes

Overview of measuring modes

You select module functionality by setting a default operating mode. The table below gives an overview of measuring modes.

Operating mode	Description
Frequency measurement (Page 9093)	The module counts the pulses it receives within a dynamic measuring time [1] and uses them to calculate the frequency.
Rotational speed measurement (Page 9094)	The module counts pulses received from a tachometer generator within a dynamic measuring time [1] and calculates the velocity based on this value and the number of pulses per encoder revolution.
Period measurement (Page 9096)	The module displays the dynamic measuring time [1] as a period. If the period is less than the update time, a mean value is calculated for the period.

[1] Dynamic measuring time means: if no 2 edges are detected by the measuring signal before the end of the parameterized measuring time, an estimated measured value will be calculated with the dynamic measuring time. It is assumed that a pulse was sent at the end of the update time. This gives you a constant measured value.

Note

FM 450-1 does not support any measuring modes.

Frequency measurement

Description

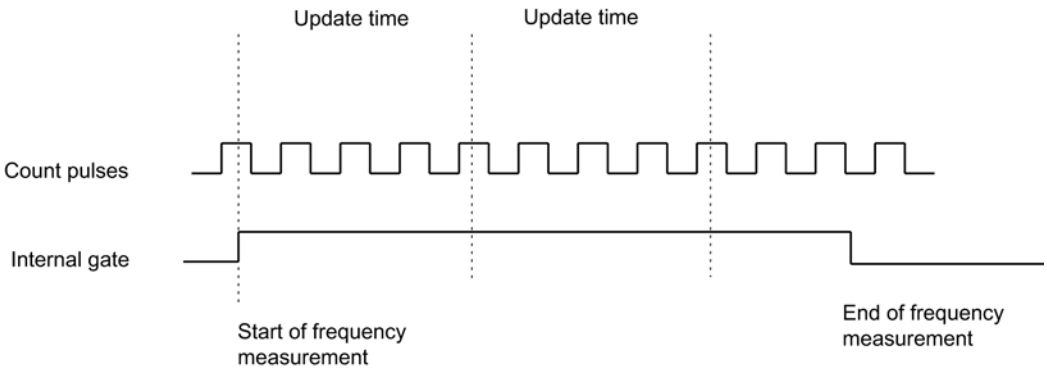
In this operating mode, the module counts the pulses it receives within a dynamic measuring time or a specified integration time. The frequency is then calculated with the edge time stamps.

Selecting gate control

This mode allows you to select gate control. The following options are available to you:

- Software gate (Page 9102)
- Hardware gate (Page 9102), level-controlled or edge-controlled

The figure below shows an example of frequency measurement with gate control.



Possible Measuring Ranges with Error Indication

The table below sets out the measuring ranges and absolute errors for continuous frequency measurements.

Frequency f	Absolute error	
	FM 350-1 ET 200S 1Count5V	ET 200S 1Count24V
0.1 Hz	± 0.001 Hz	
1 Hz	± 0.001 Hz	
10 Hz	± 0.003 Hz	
100 Hz	± 0.02 Hz	
1,000 Hz	± 0.18 Hz	
10,000 Hz	± 1.8 Hz	
100,000 Hz	± 18 Hz	
500,000 Hz	± 90 Hz	–

The table below sets out the measuring ranges and absolute errors for continuous frequency measurements with integration time.

Integration time	f_{min} ± absolute error	f_{max} ± absolute error	
		ET 200S 1Count24V	ET 200S 1Count5V
10 s	0.1 Hz ± 0.001 Hz	100,000 Hz ± 18 Hz	500,000 Hz ± 90 Hz
1 s	1 Hz ± 0.001 Hz	100,000 Hz ± 11 Hz	500,000 Hz ± 55 Hz
0.1 s	10 Hz ± 0.002 Hz	100,000 Hz ± 10 Hz	500,000 Hz ± 52 Hz
0.01 s	100 Hz ± 0.013 Hz	100,000 Hz ± 13 Hz	500,000 Hz ± 63 Hz

Values that can be changed during operation

The following values can be changed during operation:

- Low limit
- High limit
- Update time (measuring time) [1)]
- Function of the digital output [1)]

[1)] not with FM 350-2

Result

The end of a frequency measurement is reported at the STS_CMP1 or STS_COMP1 (for FM 350-2) status bit. The value of the calculated frequency is displayed in units of 10^{-3} Hz. You can read the frequency value measured from the feedback interface.

Rotational speed measurement

Description

In this operating mode, the module counts the pulses it receives from a tachometer generator within a dynamic measuring time or a specified integration time and measures the time interval between the edges. On this basis it then calculates the velocity with the pulses per encoder revolution.

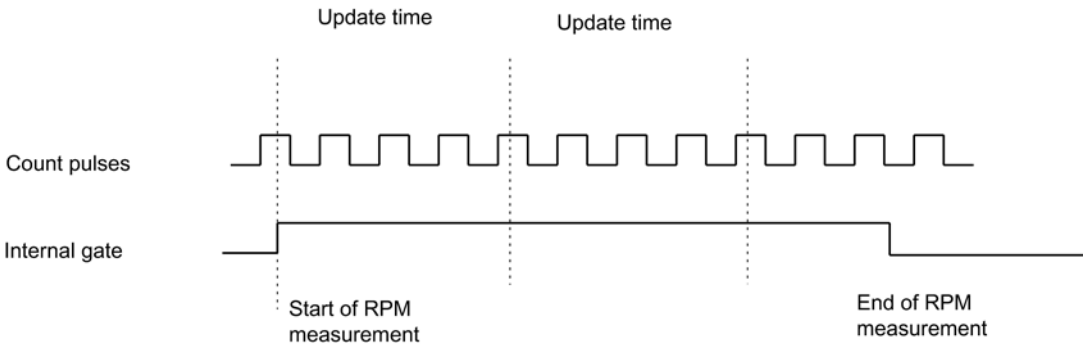
For "Rotational speed measurement" mode, you must also assign the pulses per encoder revolution.

Selecting gate control

This mode allows you to select gate control. The following options are available to you:

- Software gate (Page 9102)
- Hardware gate (Page 9102), level-controlled or edge-controlled

The figure below shows an example of rotational speed measurement with gate control.



Possible measuring ranges with error indication

The table below sets out the measuring ranges and absolute errors for continuous rotational speed measurement when the number of pulses per encoder rotation = 60.

Speed n	Absolute error
1 /min	± 0.04 /min
10 /min	± 0.04 /min
100 /min	± 0.05 /min
1,000 /min	± 0.21 /min
10,000 /min	± 1.82 /min
25,000 /min	± 4.50 /min

The table below sets out the measuring ranges and absolute errors for rotational speed measurement with integration time when the number of pulses per encoder rotation = 60.

Integration time	n_{min} ± absolute error	n_{max} ± absolute error
10 s	1 /min ± 0.03 /min	25000 /min ± 4.5 /min
1 s	1 /min ± 0.03 /min	25000 /min ± 2.8 /min
0.1 s	10 /min ± 0.03 /min	25000 /min ± 2.6 /min
0.01 s	100 /min ± 0.04 /min	25000 /min ± 3.2 /min

Values that can be changed during operation

The following values can be changed during operation:

- Low limit
- High limit
- Update time (measuring time) [1)]
- Function of the digital output [1)]

[1)] not with FM 350-2

Result

The end of a rotational speed measurement is reported at the STS_CMP1 or STS_COMP1 (for FM 350-2) status bit. The value of the velocity calculated is displayed in 10^{-3} /min. You can read the velocity measured from the feedback interface.

Period measurement

Description

In this operating mode, the module indicates the dynamic measuring time as a period. If the period is less than the update time, a mean value is calculated for the period.

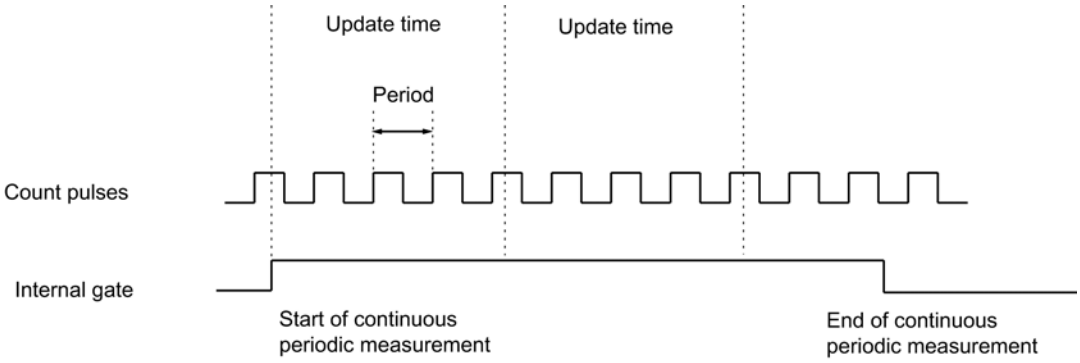
In "Period measurement with integration time" mode, 1Count24V measures the time between two positive edges of the count signal by counting the pulses of an internal quartz-accurate reference frequency (16 MHz) within a specified integration time.

Selecting gate control

This mode allows you to select gate control. The following options are available to you:

- Software gate (Page 9102)
- Hardware gate (Page 9102), level-controlled or edge-controlled

The figure below shows an example of period measurement with gate control.



Possible measuring ranges with error indication

Continuous period measurement:

- The absolute error for continuous period measurement with a **resolution of 1 µs** is equal to 0.1‰ of the measured value.
- The table below sets out the measuring ranges and absolute error for continuous period measurements with a **resolution of 1/16 µs**.

Period T_{min} ± Absolute error
1/16 µs x (160 ± 1)
1/16 µs x (1,600 ± 1)
1/16 µs x (16,000 ± 3)
1/16 µs x (160,000 ± 20)
1/16 µs × (1,600,000 ± 160)
1/16 µs × (16,000,000 ± 1,600)
1/16 µs × (160,000,000 ± 16,000)
1/16 µs × (1,600,000,000 ± 160,000)

Period measurements with integration time:

- The table below sets out the measuring ranges and absolute error for period measurements with integration time with a **resolution of 1 µs**.

Integration time	T_{min} ± absolute error	T ± absolute error
100 s	1 µs × (10 ± 0)	1 µs × (100,000,000 ± 10,000)
10 s	1 µs × (10 ± 0)	1 µs × (10,000,000 ± 1,000)
1 s	1 µs × (10 ± 0)	1 µs × (1,000,000 ± 100)
0.1 s	1 µs × (10 ± 0)	1 µs × (100,000 ± 10)
0.01 s	1 µs × (10 ± 0)	1 µs × (10,000 ± 1)

- The table below sets out the measuring ranges and absolute error for period measurements with integration time with a **resolution of 1/16 µs**.

Integration time	T_{min} ± absolute error	T ± absolute error
100 s	1/16 µs × (160 ± 0)	1/16 µs × (1,600,000,000 ± 160,000)
10 s	1/16 µs × (160 ± 0)	1/16 µs × (160,000,000 ± 16,000)
1 s	1/16 µs × (160 ± 0)	1/16 µs × (16,000,000 ± 1,600)
0.1 s	1/16 µs × (160 ± 0)	1/16 µs × (1,600,000 ± 160)
0.01 s	1/16 µs × (160 ± 0)	1/16 µs × (160,000 ± 16)

Values that can be changed during operation

The following values can be changed during operation:

- Low limit
- High limit
- Update time (measuring time) [1)]
- Function of the digital output [1)]

[1] not with FM 350-2

Result

The end of a period measurement is reported at the STS_CMP1 or STS_COMP1 (for FM 350-2) status bit. The value of the calculated period is displayed in 1 µs or 1/16 µs. You can read the period measured from the feedback interface.

Dosing (only for FM 350-2)

Description

In this mode, four counter channels are grouped together on the FM 350-2 to form one dosing channel. When the gate is open, FM 350-2 counts once in the main count direction:

- In the main count direction "up" between 0 and the parameterized high count limit.
  If you set the main count direction as "up," the start value is 0 and you specify the end value.
- In the main count direction "down" between the parameterized high count limit and 0.
  If you set the main count direction as "down," you specify the start value and the end value is 0.

Selecting gate control

This mode allows you to select gate control. The following options are available to you:

- Software gate (Page 9102)
- Software gate (Page 9102) and HW gate (Page 9102)
  You can add a hardware gate to the software gate. The two gates act together like a logical AND operation, meaning the FM 350-2 only counts when both gates are open.

Behavior at the Count Limits, Software Gate

Main count direction up: If the counter receives a further count pulse after reaching the "High count limit - 1", the module sets the counter to 0, closes the internal gate and terminates the count even if the SW_GATE0, 4 bit is still set. The relevant status bit STS_OFLW0, 4 is set. The high count limit itself is therefore never reached.

Main count direction down: If the counter receives a further count pulse after reaching the value "1", the module sets the counter to the high count limit, closes the internal gate and terminates the count even if the SW_GATE0, 4 bit is still set. The relevant status bit STS_UFLW0, 4 is set. The value "0" is therefore never reached.

If you want to start the counter again, you must reset the SW_GATE0, 4 bit and then set it again.

Behavior at the Count Limits, Hardware Gate

Main count direction up: If the counter receives a further count pulse after reaching the "High count limit - 1", the module sets the counter to 0, closes the internal gate and terminates the

count even if the SW_GATE0, 4 bit and the I0, I4 input are still set. The high count limit itself is therefore never reached. The relevant status bit STS_OFLW0, 4 is set.

Main count direction down: If the counter receives a further count pulse after reaching the value "1", the module sets the counter to the high count limit, closes the internal gate and terminates the count even if the SW_GATE0, 4 bit and the I0, I4 input are still set. The value "0" is therefore never reached. The relevant status bit STS_UFLW0, 4 is set.

If you want to start the counter again, you must reset the I0, I4 input and then set it again. You can only start a new count with the hardware gate.

Comparison values

You can configure four comparison values within the parameterized counting range for each dosing channel of the FM 350-2. You can also configure whether and under what conditions a digital output should be set and/or a hardware interrupt triggered in conjunction with a comparison value. You can set the following conditions for this:

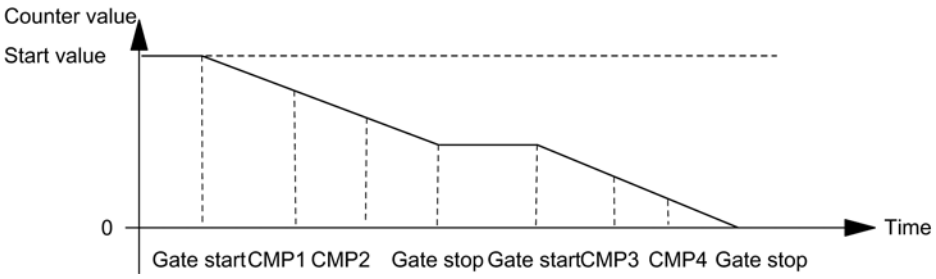
- A hardware interrupt is triggered if the current counter value matches a comparison value.
- A digital output is set if the current counter value is greater than or equal to a comparison value.
- A digital output is set if the current counter value is less than or equal to a comparison value.

A digital output is only set if you have enabled the corresponding output with the CTRL_DQ0…7 bit.

Note

Outputs can only be set while the CPU is in RUN mode if the gate is open.

The figure below shows an example of dosing in the main count direction "down" with comparison values.



Load Value in Preparation

You can define a load value within the parameterized counting range while the CPU is in RUN mode. This value is used by the counter as the new start value every time after the following events:

- When the high count limit is reached for main count direction up.
- Reaching 0 when counting down
- Canceling of the count process by a software gate or a hardware gate (when the count process is interrupted the load value is not used)

The load value is then the new initial value from which the next and all subsequent dosing processes begin. The set output and interrupt behavior remains the same.

Load Value Directly

You can change the current counter value while the CPU is in RUN mode. The new count value is used by the counter directly as the dosing value.

Value Range for Load Values

The value range for load values depends on the set main count direction. The range is:

- 0 to high count limit - 2 in main count direction "up"
- High count limit to 2 in main count direction down

Gate control

Gate functions

Counting with Gate Functions

Many applications require that the count be started or stopped at a defined time depending on other events. The counting process in counter modules is started and stopped like this using a gate function. When the gate opens, the counter receives count pulses and the count starts. When the gate is closed, the counter can no longer receive count pulses and the count stops.

Software Gate and Hardware Gate

The counter modules have two gates:

- A software gate (Page 9102) which is controlled by the input parameter / control bit SW_GATE.
- A hardware gate (Page 9102) which is controlled by signals at the module's digital inputs.

Please also see the information on gate function in isochrone mode (Page 9104).

Internal Gate

The internal gate is the logic AND operation combining a hardware gate and a software gate. The status at the internal gate is indicated by the STS_GATE bit.

If no hardware gate was assigned, only the setting of the software gate is relevant. The count process is activated, interrupted, resumed, and canceled via the internal gate. The internal gate can also be closed by events dependent on the counter value in the operating modes "Count once" and "Dosing".

Hardware gate	Software gate	Internal gate	Count process
open	open	open	active
open	closed	closed	inactive
closed	open	closed	inactive
closed	closed	closed	inactive

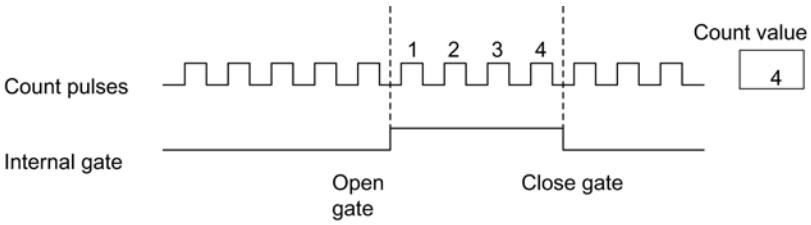
When configuring the hardware and software gates, you can specify whether the internal gate is to cancel or interrupt counting.

- When cancelled (Page 9103), the count restarts at its initial value after gate stop and gate start.
- When interrupted (Page 9103), the count is resumed from the last current count value following gate stop and gate start.

Example

The gate is opened and the count pulses are counted by setting the gate signal. If the gate signal is taken away, the gate will be closed and the count pulses no longer detected by the counter. The count value remains constant.

The diagram below shows the opening and closing of a gate and pulse counting.



Stopping the count using the gate stop function

You can also stop the count in the FM x50-1 modules by setting the gate stop function, regardless of the signals set at or status of the software gate. You do this by setting the GATE_STP input parameter.

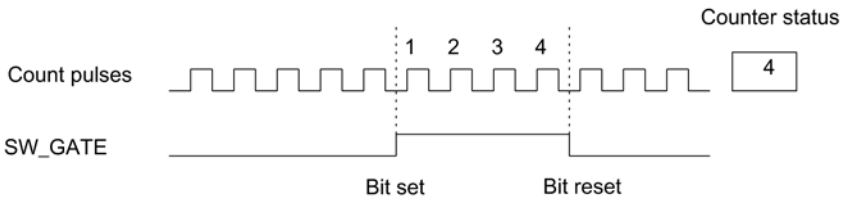
When you reset this parameter, you can only reopen the gate by setting a positive edge at the digital input "DI start" (hardware gate) or by setting the input parameter SW_GATE (software gate) again.

Software gate

You open and close the software gate with the input parameter / control bit SW_GATE.

Opening and closing the software gate

The image below shows how the software gate is opened and closed. No hardware gates should be configured.



The software gate is:

- opened by setting the input parameter / control bit SW_GATE.
- closed by resetting the input parameter / control bit SW_GATE.

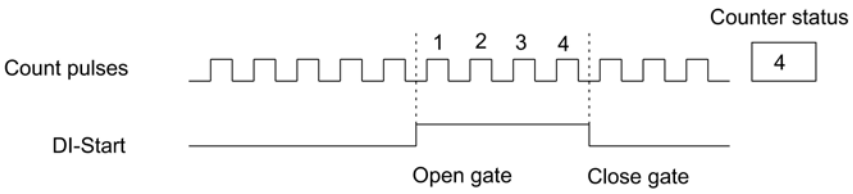
You can re-open the closed gate by resetting input parameter SW_GATE.

Hardware gate

You open and close the hardware gate by applying the relevant signals to or removing the signals from the digital inputs.

The software gate must be open.

Level-triggered opening and closing of the HW gate



Count pulses will be received and counted by the counter as long as the digital input is set. Resetting the digital input closes the gate. The counter stops and ignores any further count pulses.

If the gate is closed as a result of high limit violated or low limit violated, you open it again by a reset > set cycle at the digital input.

The level-controlled hardware gate is activated by the first positive edge at the digital input after you assign parameters.

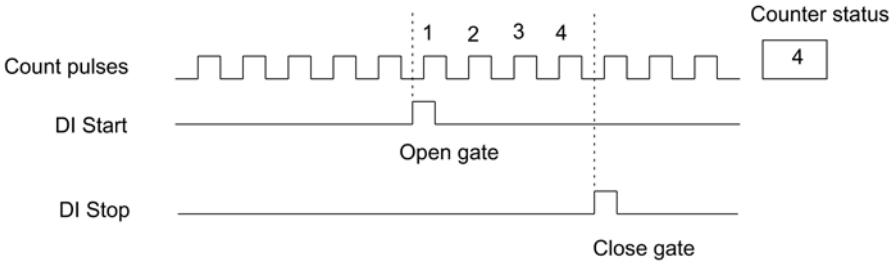
ET 200S 1Count24Vand1Count5V:

1Count24V / 1Count5V has a hardware gate, which is controlled by the digital input on the module. You configure the hardware gate as a function of the digital input (Function DI

"Hardware Gate"). The gate is opened in response to a positive edge on the digital input and closed in response to a negative edge.

Edge-triggered opening and closing of the hardware gate

Edge-controlled opening and closing of the hardware gate is only possible for FM x50-1.



An edge-controlled hardware gate is opened by a positive edge at digital input "DI start". The gate is closed by a positive edge at digital input "DI stop".

If a positive edges is set in parallel at both inputs, an open gate will be closed, whereas a closed gate remains closed. A positive edge at digital input "DI start" cannot open the gate if digital input "DI stop" is set.

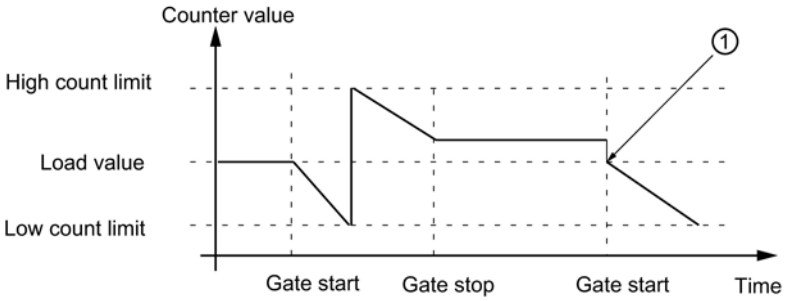
Cancel and interrupt function of the gate

In your gate function parameters, you can define whether the gate should cancel or interrupt the count.

Cancel gate function

When the "cancel" gate function is active, the count stops when the gate closes and is restarted at the load value when the gate is restarted (time ① in the figure below).

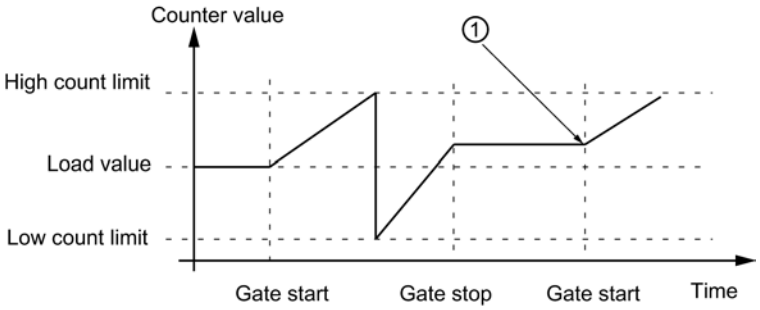
The figure below shows an example of continuous counting down with the "cancel" gate function.



"Interrupt" gate function

When the "interrupt" gate function is active, the count stops when the gate closes, and resumes at the last current counter value when the gate is restarted (time ① in the figure below).

The figure below shows an example of continuous counting up with the "interrupt" gate function.

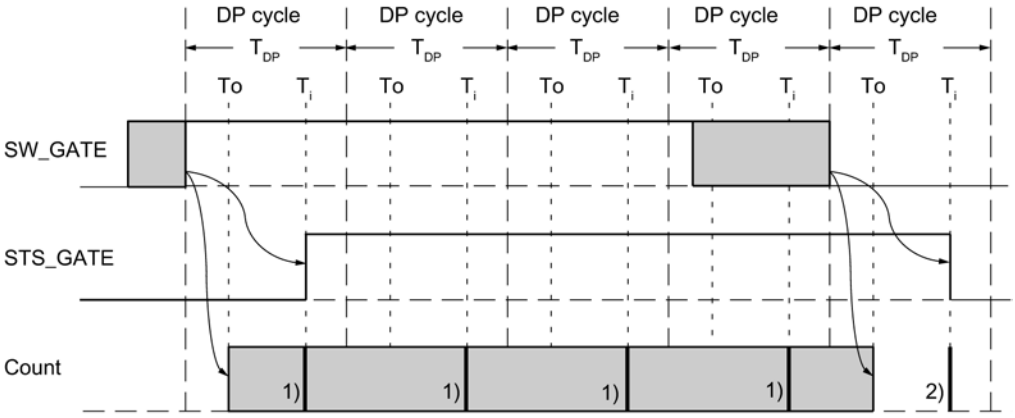


Gate control in isochronous mode

Gate control operates as follows in isochrone mode:

Software gate control

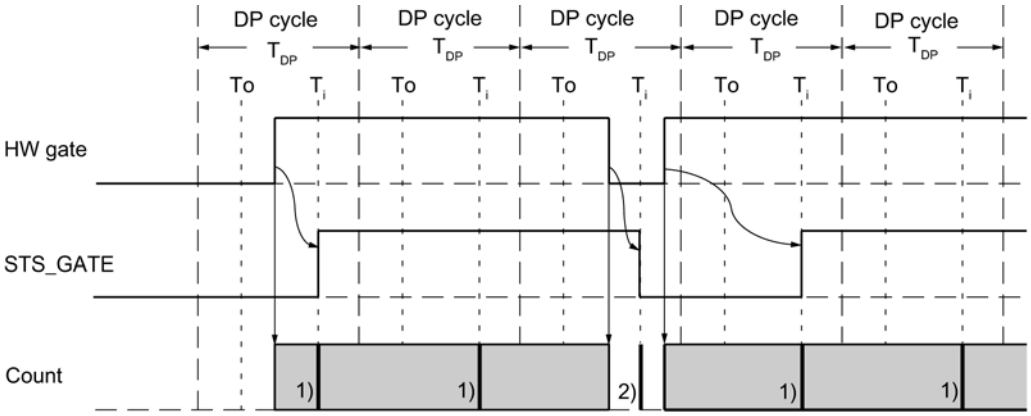
The software gate is controlled by setting and resetting the SW_GATE control bit. After the control bit has changed, the count starts and ends at time T_o in the next PROFIBUS DP cycle:



1) Provision of actual count value
2) Provision of the count value that was valid at the end of the count

Hardware gate control

In hardware gate control mode, the count starts or stops instantaneously when the hardware gate opens or closes:



1) Provision of actual count value
2) Provision of the count value that was valid at the end of the count

Encoder signals

Encoders which can be connected

The counter module can process square wave signals generated by incremental encoders or pulse generators.

- **Incremental encoders** scan a grating to generate square wave electrical pulses. They differ in terms of pulse amplitude and number of signals. Incremental encoders output two tracks out of phase by 90°.
- **Pulse generators** such as light barriers and initiators (BEROs) only supply a square wave signal with a specific voltage level.

5 V differential signal

5 V incremental encoder count signals

The 5 V incremental encoder sends the following RS 422 differential signal to the module:

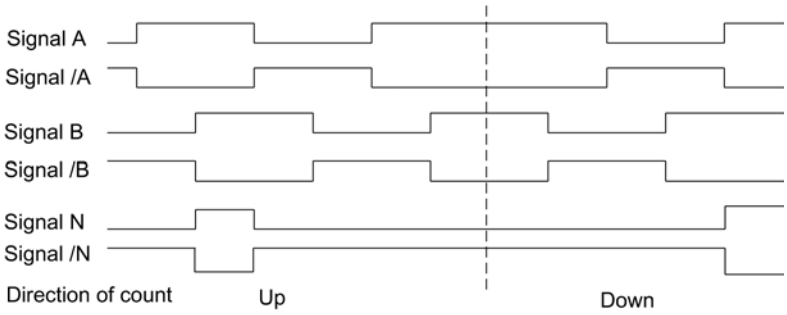
- A and /A
- B and /B
- N and /N

The signals /A, /B and /N are the inverted signals of A, B and N. Signals A and B are phase-shifted by 90°.

5 V incremental encoders use tracks A and B for counting. Track N is used to initialize the counter with the load value, if programmed accordingly.

Encoders which return inverted signals for each signal sent are symmetrical encoders.

The diagram below shows the time profile of the encoder signals of a 5 V incremental encoder.



The module detects the count direction by evaluating the ratio of signals A and B.

Changing the count direction

You can change the count direction using the "Count direction normal" and "Count direction inverted" parameters without having to modify the wiring.

Monitoring encoder signals

The module monitors the cable connection, and detects wire-break or short-circuit.

You can define which of the three signal pairs to include in monitoring in your program. There is no need to wire any unused signal pairs, if you have disabled the corresponding diagnostics functions in the program (monitoring.)

An error at all three signals indicates a defective encoder, a short circuit in the encoder supply or that no encoder is connected.

When programming is completed, and the module detects an error, the error information will be written to the diagnostics data records DS0 and DS1. This situation my lead to a diagnostics interrupt if programmed accordingly.

24 V signals

24 V incremental encoder count signals

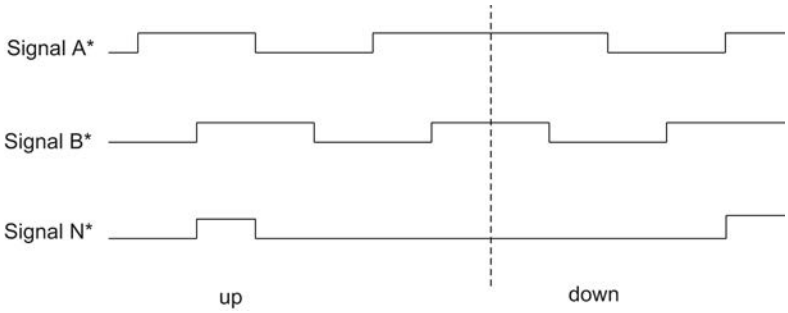
The 24 V incremental encoder returns the 24 V signals A*, B* and N* to the module. The A* and B* signals are phase-shifted by 90°.

24-V signals are marked with an asterisk "*" character.

24 V incremental encoders use tracks A* and B* for counting. Track N* is used to initialize the counter with the load value, if programmed accordingly.

Encoders which do not return inverted signals asymmetrical encoders.

The diagram below shows the time profile of the encoder signals of a 24 V incremental encoder.



The module detects the count direction by evaluating the ratio of signals A* and B*.

You can configure the inputs of 24 V encoder signals for the connection of Sourcing output/ push pull or Sinking output to the counter. For further information, refer to the encoder manual.

Changing the count direction

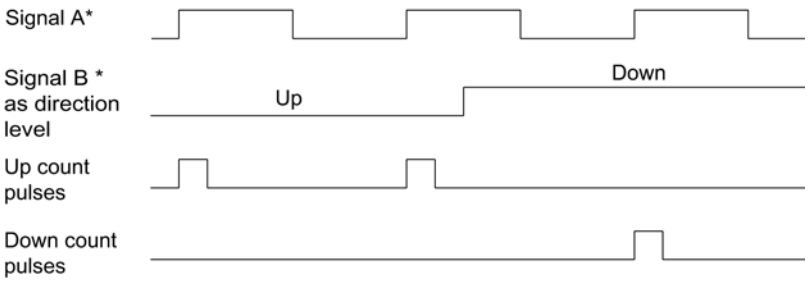
You can change the count direction using the "Count direction normal" and "Count direction inverted" parameters without having to modify the wiring.

Count signals from 24 V pulse generators without/with direction level

Encoders such as proximity switches (BERO) or light barriers return only a count signal which you wire to terminal A* of the front connector.

in additional, you can wire a signal for direction detection to terminal B* of the relevant counter. If your encoder does not return a corresponding signal, you can wire a corresponding ID signal you generate within the user program, or use a corresponding process signal.

The diagram below shows the time profile of the encoder signals of a 24 V pulse generator with direction level and the resultant count pulses.



Monitoring encoder signals

The 24 V count signals are not monitored to detect wire breaks or short circuits.

Signal evaluation

Overview

The counter module counter counts the signal edges. It usually evaluates the edge at A (A*) (single evaluation). Options in the program of increasing the resolution:

- Single evaluation (Page 9108)
- Double evaluation (Page 9108)
- Quadruple evaluation (Page 9109)

Multiple evaluation is supported only:

- for incremental 5 V encoders with signals A and B out of phase by 90°.
- for incremental 24 V encoders with signals A* and B* out of phase by 90°.

Default setting

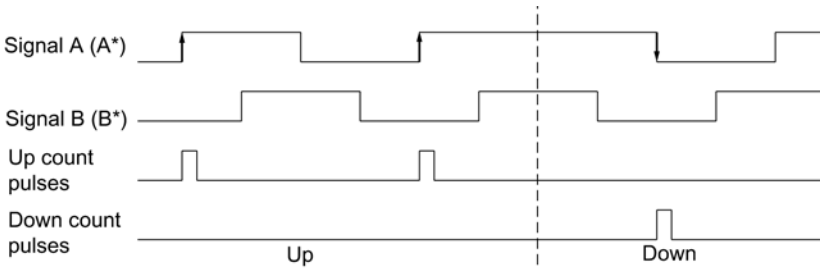
The default setting is "Single evaluation".

Single evaluation

In this mode, the module evaluates only one edge of signal A.

- "Up" count pulses are recorded at the negative edge at A and a low level at B.
- Down count pulses are recorded if the edge at A falls or the level at B is low.

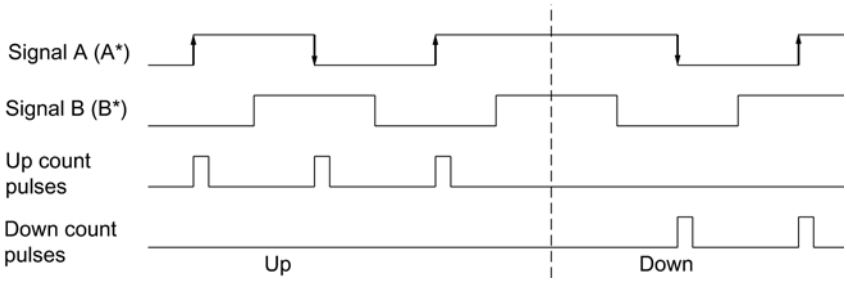
The diagram below illustrates the single evaluation of the signals.



Double evaluation

Double evaluation refers to the evaluation of the positive and negative edges of signal A. The level of signal B determines whether up or down count pulses are generated.

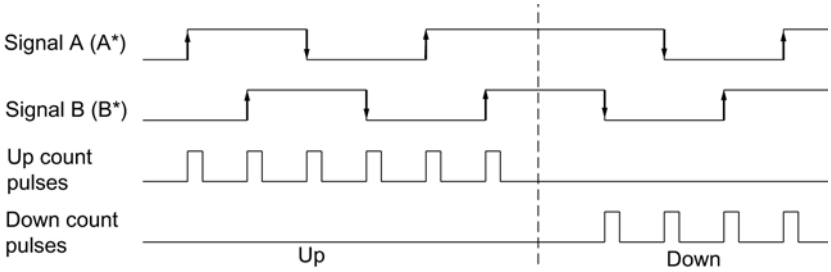
The diagram below shows double evaluation of signals.



Quadruple evaluation

Quadruple evaluation refers to the evaluation of the positive and negative edges of signals A and B. The level of signals A and B determines whether up or down count pulses are generated.

The diagram below illustrates the quadruple evaluation of signals.



Using FM 350-1

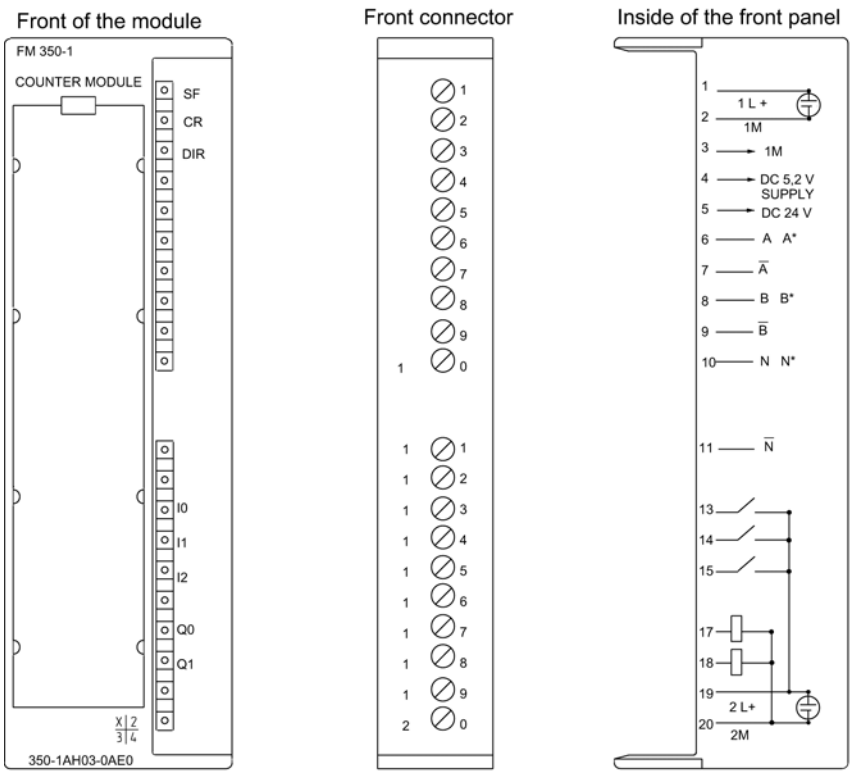
FM 350-1 front connector terminal assignment

Front connector

You connect the following to the 20-pin front connector:

- Count signals
- Digital inputs
- Digital outputs
- Encoder power supply
- Auxiliary voltage and load voltage

The figure below shows the front of the module, the front connector and the inside of the front panel with terminal assignment.



Pin assignments of the front connector

The following table shows the terminal assignment for the front connector of the FM 350-1.

Connection	Name	Input/ output	Function			
Auxiliary voltage						
1	1L+	Input	24 V auxiliary voltage			
2	1M	Input	Auxiliary voltage ground			
Encoder supply and count signals			**5 V encoder RS 422, symmetrical**	**24 V encoder, asymmetrical**	**24 V pulse generator with direction level**	**24 V initiator**
3	1M	Output	Encoder supply ground			
4	5.2 V DC	Output	Encoder power supply 5.2 V			
5	24 V DC	Output	Encoder power supply 24 V			
6	A A*	Input	Encoder signal A	Encoder signal A *		
7	/A	Input	Encoder signal /A	—		
8	B B*	Input	Encoder signal B	Encoder signal B *	Directional signal	—
9	/B	Input	Encoder signal /B	—		
10	N N*	Input	Encoder signal N	Encoder signal N *	—	
11	/N	Input	Encoder signal /N	—		
12	—	—	—			
Digital Inputs and Digital Outputs						
13	I0	Input	Digital input DI Start			

Connection	Name	Input/ output	Function
14	I1	Input	Digital input DI Stop
15	I2	Input	Digital input DI Set
16	—	—	—
17	Q0	Output	Digital output DO0
18	Q1	Output	Digital output DO1
Load voltage			
19	2L+	Input	24 V load voltage
20	2M	Input	Load voltage ground for the digital inputs and outputs

Note

The circuits for the counter inputs (encoder supply, encoder signals) are non-isolated from the ground of the CPU. You should therefore connect terminal 2 (1M) to the CPU ground using a low-resistance connection. The lack of such a connection can result in malfunction or a defect in the FM 350-1.

If you supply the encoders externally, you must also connect the ground of this external voltage with the ground of the CPU.

You will find more information on construction and wiring in the manual (http://support.automation.siemens.com/WW/view/en/1086726) for the FM 350-1 counter module.

Auxiliary voltage 1L+/1M

Connect a direct voltage of 24 V to the 1L+ and 1M terminals for the 5 V and 24 V encoder voltage supply.

An integrated diode protects the module from a reversal in the polarity of the auxiliary voltage.

The module monitors the connection of the auxiliary voltage.

5.2V DC encoder power supply

The module generates a voltage of 5.2 V from the auxiliary voltage 1L+/1M at a maximum current of 300 mA, supplied at the "DC5.2V" connection for the short circuit proof supply of a 5 V encoder. The encoder supply is monitored for short-circuit.

24 V DC encoder supply

1L+/1M is provided at output "DC24V" for 24V short-circuit proof encoder voltage supply. The encoder supply is monitored for short-circuit.

5 V encoder signals A and /A, B and /B, N and /N

You can connect incremental encoders with 5 V differential signal to the front connector in accordance with RS 422, i.e. incremental encoders with the differential signal A and /A, B and /B, N and /N.

The signals A and /A, B and /B, N and /N are connected via the correspondingly labeled terminals.

The signals N and /N are only connected if you want to set the counter to the zero mark of the encoder.

The inputs are not isolated from the S7-300 bus.

24 V encoder signals A*, B* and N*

24 V signals are identified with the letters A*, B* and N*.

You can connect three different encoder types to each counter:

- Incremental encoder with 24 V signals:
  The signals A*, B* and N* are connected via the correspondingly labeled terminals.
- Pulse encoders without direction level:
  The signal is connected to terminal A*.
- Pulse encoders with direction level:
  The count signal is connected to terminal A*. The direction level is connected to terminal B*.

The inputs are not isolated from the S7-300 bus.

Input filter for 24 V encoder signals

To suppress faults, you can configure input filters (RC elements) with a uniform filtering time for the 24 V inputs A*, B* and N*. The following two input filters are available.

Characteristics	Input Filter 1 (default)	Input Filter 2
Typical input delay	1 µs	15 µs
Maximum count frequency	200 kHz	20 kHz
Minimum pulse width of the count signals	2.5 µs	25 µs

Digital Inputs DI Start, DI Stop and DI Set

You can use digital inputs DI Start and DI Stop for the gate control of the counter. The gate control can be either level-controlled or edge-controlled.

Digital input DI Set is used to set the counter to the load value.

The digital inputs are operated with a nominal voltage of 24 V.

The digital inputs are isolated from the bus of the S7-300 and the counter inputs.

Input Filters for Digital Inputs

For the purpose of suppressing interference, you can parameterize input filters DI-Start, DI-Stop and DI-Set (RC elements) with a uniform filter time for digital inputs. The following two input filters are available:

Characteristics	Input Filter 1 (default)	Input Filter 2
Typical input delay	1 µs	15 µs
Minimum pulse width of the input signals	2.5 µs	25 µs

Digital Outputs DO0 and DO1

The FM 350-1 has two digital outputs, DO0 and DO1, for directly triggering control processes.

The digital outputs are supplied via the load voltage 2L+.

The digital outputs are isolated from the bus of the S7-300 and the counter inputs.

The digital outputs are Sourcing outputs and can carry load current of 0.5 A. They are protected from overload and short-circuit.

Note

Relays and contactors can be connected directly without external circuitry.

The time response of the digital outputs depends on parameter assignment.

2L+/2M load voltage

For supplying digital outputs DO0 and DO1, a load voltage of 24 V must be supplied to the module via terminals 2L+ and 2M.

An integrated diode protects the module from polarity reversal of the load voltage.

The load voltage 2L+/2M is not monitored by the FM 350-1.

Configuring FM 350-1

Configuring FM 350-1

Operating modes

Selecting an operating mode

Description

You define counter module functionality by setting an operating mode.

Operating mode	Description
Count continuously (Page 9083) (with or without gate)	The FM 350-1 counts continuously from the current counter value.
Count once (Page 9084) (with hardware or software gate)	When the gate opens, the FM 350-1 counts from the load value to the count limit.
Count periodically (Page 9085) (with hardware or software gate)	When the gate opens, the FM 350-1 counts from the load value between the count limits.
Frequency measurement (Page 9093) (with hardware or software gate)	The FM 350-1 counts the pulses that occur during a dynamic measuring time.
Rotational speed measurement (Page 9094) (with hardware or software gate)	The FM 350-1 counts pulses received from a tachometer generator within a dynamic measuring time and calculates the velocity based on this value and the number of pulses per encoder revolution.
Period measurement (Page 9096) (with hardware or software gate)	The FM 350-1 indicates the dynamic measuring time as a period. The mean value of the period is calculated if its duration is less than the update time.

The default setting is the "Count continuously" operating mode.

Specifying the high count limit

Description

By configuring an high count limit together with a main count direction (Page 9119) (up or down), you can reduce the maximum counting range. The parameterized counting range is then between 0 and the high count limit parameterized.

The high count limit can be any value between > 1 and the high count range limit.

Specifying measuring time

Description

If you have selected a measuring mode, the module will update the measured values on a cyclic basis. The update time is the product of the measuring time x 10ms. You set the measuring time as the factor n.

Detailed descriptions of the value ranges for the measuring time can be found under "Measuring basics (Page 9088)".

Selecting gate control

Description

You define the count process in the selected operating mode by setting a gate control.

- without gate
  All gates are open.
- Software gate (Page 9102)
  The software gate is opened and closed by the input parameter / control bit SW_GATE.
- Hardware gate (Page 9102), level-controlled or edge-controlled
  The hardware gate is opened and closed by the corresponding signals at the digital inputs.
- Latching (Page 9115)
  You can save (latch) the counter values by generating edges at the digital input "DI start".
- Latch/Retrigger (Page 9117)
  You can save (latch) the counter values by generating edges at the digital input "DI start". The counter is set to the load value each time the values are saved.

The default setting is "Without gate".

Latching

Description

If you select "Latch" for the gate control, you can save (latch) the counter values with edges at the "DI start" digital input. This does not change the counter value.

Requirements

You must use the software gate (Page 9102) and select the edge for latching (Page 9120).

The minimum interval between latch edges is 1 ms. Values can be lost if the time interval between edges is less than this value.

Function principle

The counter and latch values are assigned an initial value. The initial value is 0 for operating modes without a main count direction and for main count direction up. For main count direction down, the initial value is the parameterized high count limit.

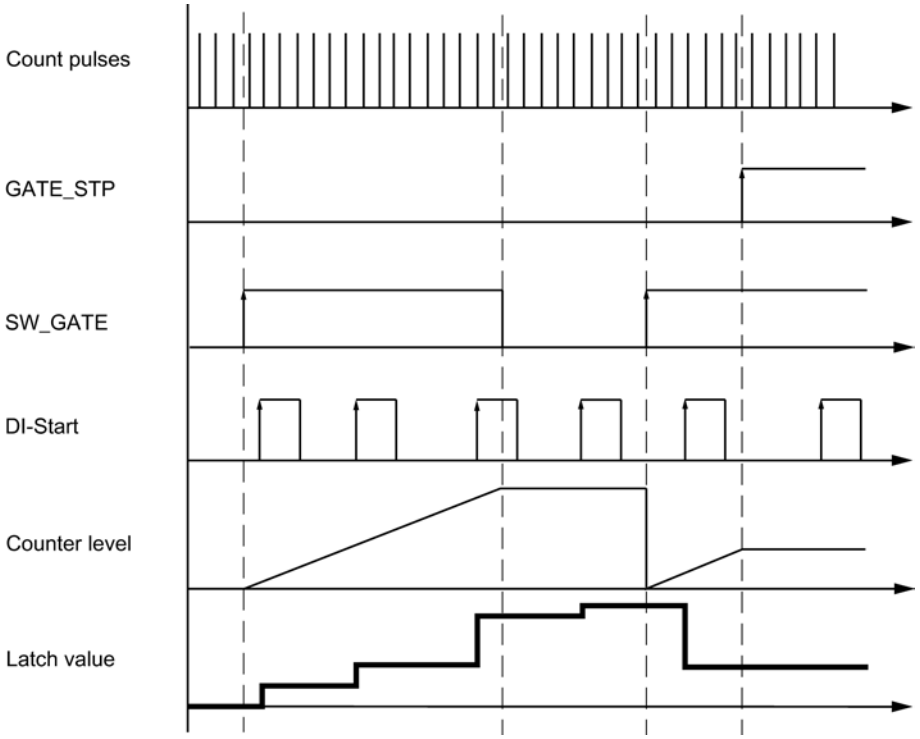
The counting function is started when the software gate is opened. The counter starts at the load value.

The latch value is always exactly equivalent to the counter value at the time the edge trigger is generated.

The "DI start" value is indicated in the DB at status bit STS_STA.

The latch value is indicated by LATCH_LOAD in the DB.

The diagram below shows an example of "Latching" when load value = 0 and there is a positive edge at "DI start".



Interrupting and exiting

Closing the software gate by resetting SW_GATE will cancel counting.
The unchanged counter value can be resaved with edges at "DI start" even if the software gate is closed.

However, the count will be cancelled if you close the software gate by executing GATE_STP of instruction CNT_CTL1. "DI start" can in this case no longer be used to save any counter values.

Hardware interrupt upon "Latching"

A hardware interrupt can be triggered each time a counter value is saved by "Latching". As a consequence, you may need to increase the interval between the edges. The hardware interrupts will be lost if the interrupt rate is higher than the system acknowledgement rate. This situation is reported by a diagnostic interrupt.

Latch/Retrigger

Description

If you select "Latch/Retrigger" gate control, you can save (latch) the counter values with edges at the "DI start" digital input. After each latching operation, the counter will be initialized with the load value, and resumes the count starting at the load value (retrigger.)

Requirements

You must use the software gate (Page 9102) and select the edge for latching (Page 9120) for "Latch/Retrigger".

The minimum interval between latch edges is 1 ms. Values can be lost if the time interval between edges is less than this value.

Function principle

The counter function is enabled by opening the SW gate.

The counter and latch values are assigned an initial value. These values do not change by opening the SW gate. The initial value is 0 for operating modes without a main count direction and for main count direction up. For main count direction down, the initial value is the parameterized high count limit.

The first edge at input "DI start" starts the count with the load value.

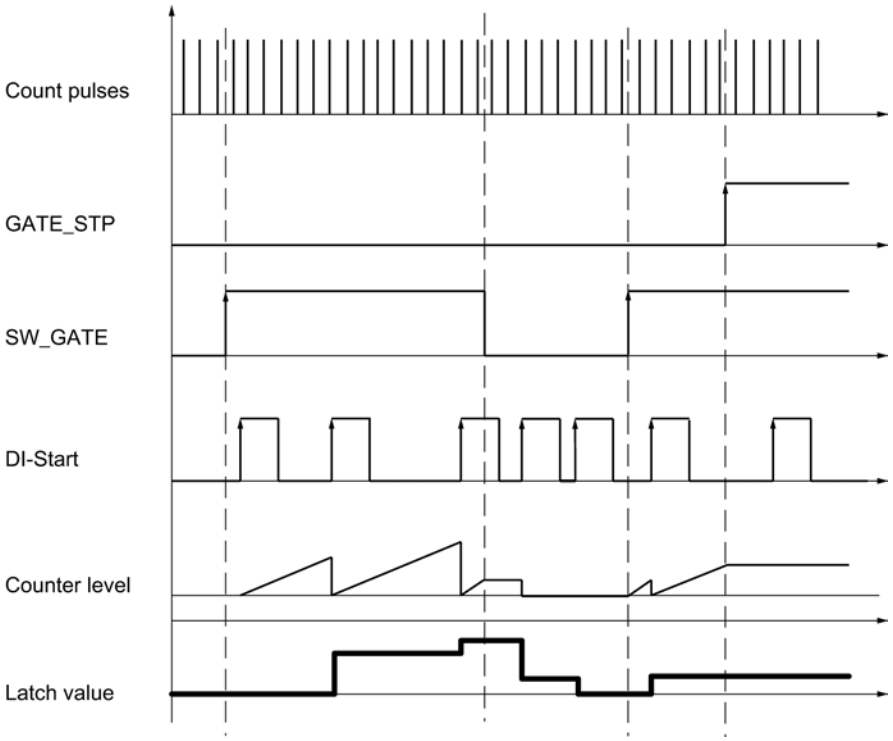
The load value is loaded again at each further edge at the input "DI start".

The latch value is always exactly equivalent to the counter value at the time the edge trigger is generated.

The "DI start" value is indicated in the DB at status bit STS_STA.

The latch value is indicated by LATCH_LOAD in the DB.

The diagram below shows an example of "Latch/Retrigger" when load value = 0 and there is a positive edge at "DI start".



Measuring times between two edges (Page 9130) using the internal time base 1 MHz (Page 9123) is a special application for the "Latch/Retrigger" gate control.

Interrupting and exiting

Closing the software gate by resetting SW_GATE will interrupt counting, i.e. the count will continue when the software gate is reopened.
The unchanged counter value can be resaved with edges at "DI start" even if the software gate is closed.

However, the count will be cancelled if you close the software gate by executing GATE_STP of instruction CNT_CTL1. "DI start" can in this case no longer be used to save any counter values.

Hardware interrupt upon "Latch/Retrigger"

A hardware interrupt may be triggered each time a counter value is saved by executing "Latch/Retrigger". As a consequence, you may need to increase the interval between the edges. The hardware interrupts will be lost if the interrupt rate is higher than the system acknowledgement rate. This situation is reported by a diagnostic interrupt.

Selecting the counting range

Description

There is a 32-bit wide count register on the module. With the Count range, you select whether the module is to count only in the positive range or whether the 32nd bit is interpreted as a

sign bit thus allowing negative numbers to be represented. You can only select a count range if no main counting direction is set.

Select the desired counting range.

For more information, please see "Counting ranges and count limits (Page 9081)".

Violation of the high limit, low limit and zero pass

A bit is set in the DB of the CNT_CTL1 instruction for both count limits when the high or low limit is violated.

A bit is also set in the DB in the "-31 to +31-bits" counting range in the event of a zero pass.

Depending on the count direction, violation of the high or low limit is also indicated in the "0 to +32 bits" counting range upon zero pass.

Event	Status bit in DB
Violation of the high limit	STS_OFLW is set.
Violation of the low limit	STS_UFLW is set.
Zero pass	STS_ZERO is set.

Initiating Hardware Interrupts

You can also signal the violation of the high limit, low limit and the zero pass events using hardware interrupts.

Selecting the gate function

Description

When configuring the hardware and software gates, you can specify whether the internal gate is to cancel or interrupt counting.

- Cancel counting (Page 9103): The counting process restarts from the beginning after gate stop and restart.
- Interrupt counting (Page 9103): The counting process starts from the last current counter value after gate stop and restart.

Selecting main count direction

Description

By configuring a main count direction (Page 9086) (up or down), you can reduce the maximum counting range by setting an high count limit (Page 9114). The parameterized counting range is then between 0 and the high count limit parameterized.

A main count direction can only be selected in the "Count once" and " Count periodically" modes.

You can select the following behavior:

- None (no main count direction)
- Up
- Down

Selecting an edge for latching

Selecting edges

You can select the edge at the digital input "DI start" at which the counter values are to be saved (latched). The following behavior can be configured:

- Latching at a positive edge at "DI-Start"
- Latching at a negative edge at "DI-Start"
- Latching at both edges at "DI-Start"

Selecting period resolution

Description

The value of the calculated period is displayed in 1 µs or 1/16 µs.

Select the required unit of resolution.

Possible Measuring Ranges with Error Indication

- The absolute error for a **resolution of 1 µs** is equal to 0.1‰ of the measured value.
- The table below sets out the measuring ranges and errors for a**resolution of 1/16 µs**:

Period T_{min} ± Absolute error
1/16 µs x (160 ± 1)
1/16 µs x (1,600 ± 1)
1/16 µs x (16,000 ± 3)
1/16 µs x (160,000 ± 20)
The absolute error for a period of 100ms or more is equal to 0.1‰ of the measured value.

Encoder

Selecting the encoder signal type

Selecting the signal type

1. Select the type of signal for the encoder used.
   In line with your selection and the operating mode, any option not possible for this specific encoder will be disabled. A disabled option is identified by grayed font in the corresponding selection window.
2. Select the options to be used with the selected encoder.

You can select the signal type from one of the following encoders:

- Encoder: 5 V incremental (Page 9121) (5 V differential signal)
- Encoder: 24 V differential signal (Page 9122) (24 V track A+B phase shifted)
- Encoder: 24 V pulse and direction (Page 9122)
- Encoder: 24 V initiator (Page 9123)
- Internal time base 1 MHz (Page 9123)

Encoder: 5 V incremental

5 V incremental encoder

Select the signal type "5V differential signal" to use a 5 V incremental encoder with differential signal A, /A, B, /B and N, /N.

The following options are available for this encoder:

Parameter	Option	Meaning
Signal evaluation (Page 9108)	Single	Only one edge of signal A is evaluated.
	Double	Both edges of signal A are evaluated.
	Quadruple	Both edges of signals A and B are evaluated.
Count direction (Page 9105)	Normal	To adapt the count direction to the process
	Inverted	To adapt the count direction to the process
Max. count frequency	–	The maximum count frequency for this encoder is 500 kHz.
Encoder inputs	–	Not relevant
Monitoring (Page 9105)	A+B+N	All signal pairs are monitored for short-circuit and wire break.
	A+B	Signal pairs A and B are monitored.
	A	Only signal pair A is monitored.
	None	No signal pair is monitored.

Encoder: 24 V incremental

24 V incremental encoder

Select signal type "24 V track A+B phase shifted" to use a 24 V incremental encoder with signals A*, B* and N*.

The following options are available for this encoder:

Parameter	Option	Meaning
Signal evaluation (Page 9108)	Single	Only one edge of signal A is evaluated.
	Double	Both edges of signal A are evaluated.
	Quadruple	Both edges of signals A and B are evaluated.
Count direction (Page 9106)	Normal	To adapt the count direction to the process
	Inverted	To adapt the count direction to the process
Max. count frequency (Page 9124)	200 kHz	A filter with 1 µs delay is active at the inputs.
	20 kHz	A filter with 15 µs delay is active at the inputs.
Encoder inputs (Page 9123)	Sinking output	The encoder switches to 0 V (ground).
	Sourcing output/ push pull	The encoder switches to 24 V.
Monitoring	–	Not relevant

Encoder: 24 V pulse and direction

24 V pulse generator

Select signal type "24 V pulse and direction" to use a 24 V pulse generator with count signal A* and direction signal B*.

The following options are available for this encoder:

Parameter	Option	Meaning
Signal evaluation	–	Not relevant
Count direction	–	Not relevant
Max. count frequency (Page 9124)	200 kHz	A filter with 1 µs delay is active at the inputs.
	20 kHz	A filter with 15 µs delay is active at the inputs.
Encoder inputs (Page 9123)	Sinking output	The encoder switches to 0 V (ground).
	Sourcing output/ push pull	The encoder switches to 24 V.
Monitoring	–	Not relevant

Encoder: 24 V initiator

24 V initiator

Select the signal type "24 V initiator" to use a 24 V pulse generator with count signal A*.

The following options are available for this encoder:

Parameter	Option	Meaning
Signal evaluation	–	Not relevant
Count direction	–	Not relevant
Max. count frequency (Page 9124)	200 kHz	A filter with 1 µs delay is active at the inputs.
	20 kHz	A filter with 15 µs delay is active at the inputs.
Encoder inputs (Page 9123)	Sinking output	The encoder switches to 0 V (ground).
	Sourcing output/ push pull	The encoder switches to 24 V.
Monitoring	–	Not relevant

Internal timebase 1 MHz

Internal time base 1 MHz

Select the signal type "Internal time base 1 MHz " to use the internal 1 MHz time base instead of an external encoder.

This option allows you to measure the intervals between two successive edges at the DI start digital input, for example.

See also

Measuring times between two edges (Page 9130)

Encoder input parameter assignment

You have selected the "24 V pulse and direction" for the signal type (Page 9121) in parameter assignment.

Encoder input parameter assignment

The count direction is defined by configuring the encoder inputs as follows. The table shows a change in count direction based on parameter assignment.

Parameter assignment	Terminal B*	Count direction
Sinking output	not wired	Down
	Short-circuited to ground	Up

Parameter assignment	Terminal B*	Count direction
Sourcing output/push pull	not wired	Up
	24 V connected	Down

You cannot reverse the direction of these count signals by inverting the B* signal.

Note

As all signals are added, this type of evaluation may cause the count value to "drift off" at the edge if the count signal oscillates.

Input filters for 24 V counter inputs

Input filters for 24 V counter inputs

To suppress faults, you can configure input filters with a uniform filtering time for the 24 V counter inputs A*, B* and N*. Input filters available:

Features	Input filter 1 (default setting)	Input filter 2
Typical input delay	1 µs	15 µs
Maximum count frequency	200 kHz	20 kHz
Minimum pulse width of the count signals	2.5 µs	25 µs

Entering pulses per encoder revolution

Pulses per encoder revolution

For "Rotational speed measurement (Page 9094)" mode, you must also configure the pulses per encoder revolution.

The value for the number of pulses per encoder revolution can be found on the rating plate, or in the technical data of your encoder.

Inputs

Selecting gate signals

Controlling the hardware gate

The following options are available for hardware gate control:

- Level-controlled hardware gate (with level at "DI start")
- Edge-controlled hardware gate (with edges at "DI start", "DI stop")

For additional information on this topic, please refer to "Hardware gate (Page 9102)".

Selecting minimum pulse width

Minimum pulse width

To suppress faults, you can configure input filters (RC elements) with a uniform filtering time for the digital inputs.

Two input filters are available with the following data:

Features	Input filter 1 (default setting)	Input filter 2
Typical input delay	1 µs	15 µs
Minimum pulse width of input signals	2.5 µs	25 µs

Setting the counter (DI set)

Setting the counter

Overview

If you want to start the count from a specific value, the so-called load value, you must define the signal that is to be used to set the counter to the load value. You can set the counter:

- With the L_DIRECT or L_PREPAR input parameter of the CNT_CTL1 instruction
- With an external signal, using either the "DI set" input or "DI set" in combination with the zero mark of the encoder

Load value

Any number within the counting range can be specified as the load value.

The load value is interpreted according to the selected counting range. If, for example, you specify FFFF FFFF H as the load value, this will be interpreted as 4 294 967 295 in the "0 to +32 bits" counting range and as -1 in the count range "-31 to +31 bits".

Enter the load value in the DB of the CNT_CTL1 instruction and transfer it to the module with CNT_CTL1.

The following ranges of values apply for the load value:

Value range for load values	Main count direction		
	None	Up	Down
Low limit	Maximum low count limit	$-2^{31} + 1$	2
High limit	Maximum high count limit	Parameterized high count limit - 2	$2^{31} - 1$

Setting the counter with an external signal

You can select two different external signals with which to set the counter to the load value:

- "DI set" (Page 9126) only
- "DI set" and zero mark (Page 9128) of the encoder

You use the encoder zero mark if you want to synchronize the counter to a specific counter value at a specific point in your process. This improves the precision of the counting process.

The counter is set independently of the counting mode.

When the counter has been set with an external signal, bit STS_SYNC will be set in the DB.

Note

Synchronizing a counter with the zero mark only makes sense if the gate is open.

If you have only enabled one count direction when setting a counter with an external signal, you must note that the current count direction will be saved (frozen) when the gate is closed. This allows the counter to be synchronized against the enabled count direction.

Setting the counter with "DI set"

Setting the counter with "DI set"

The counter can be loaded with the load value by means of a rising edge at "DI set".

You can define the response of the FM 350-1 to a rising edge at DI set with variables ENSET_UP and ENSET_DN in the DB of the CNT_CTL1 instruction and by assigning parameters.

Parameter	Behavior of the FM 350-1
ENSET_UP set	The counter is only set if counting up.
ENSET_DN set	The counter is only set if counting down.
ENSET_UP and ENSET_DN set	The counter is set for counting up and counting down.
"Single setting of counter" parameter assignment	The counter is set only in response to the first rising edge at "DI set". If the counter is to be set again, you must first set ENSET_UP or ENSET_DN again. The counter will then be set again at the next rising edge at "DI set".
"Multiple setting of counter" parameter assignment	The counter is set in response to each rising edge at "DI set" as long as ENSET_UP and/or ENSET_DN are set.

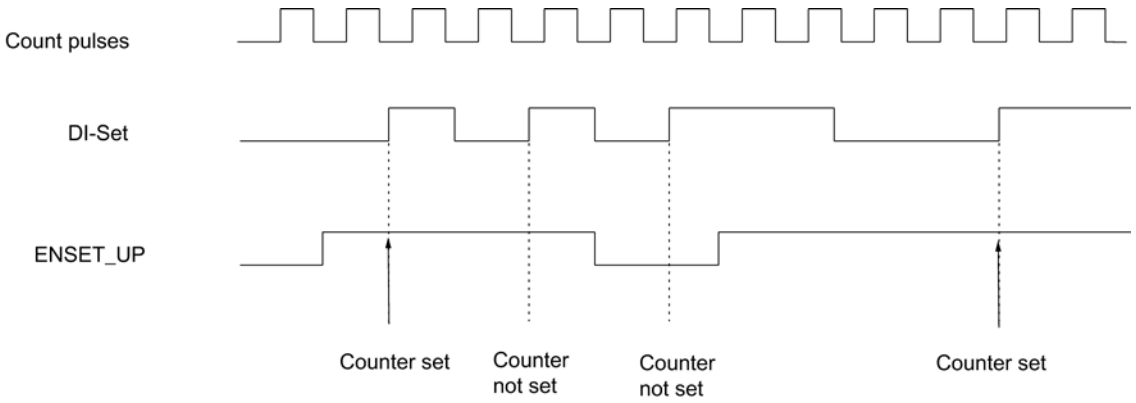
Note

It is imperative that you set one of the two variables ENSET_UP and ENSET_DN so the counter can be set by the digital input "DI set".

Single setting with "DI set"

The figure below shows single setting of the counter with digital input "DI Set". In the case shown, only ENSET_UP is set, i.e. the counter is set when counting up.

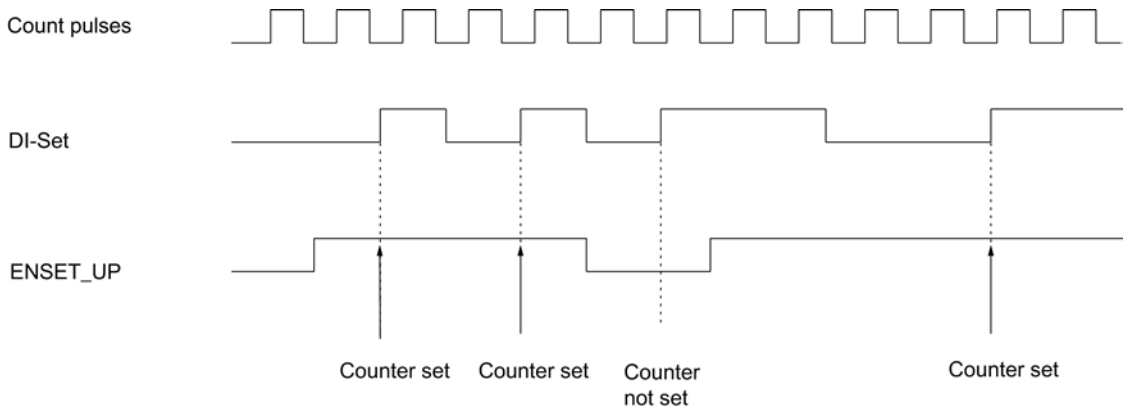
The counter is set in response to the first rising edge at digital input "DI set" provided that ENSET_UP is also set. If you want to set the counter again, you must first reset ENSET_UP and then set it again. The next rising edge at digital input "DI set" will then result in the counter being set.



Multiple setting with "DI Set"

The figure below shows multiple setting of the counter with "DI set". In the case shown, only ENSET_UP is set, i.e. the counter is set when counting up.

The counter is set at every rising edge at "DI set" provided that ENSET_UP is also set. If you reset ENSET_UP, the counter will not be set by "DI set". Only once you have set ENSET_UP again will the next positive edge at "DI set" set the counter.



Setting the counter with "DI set" and zero mark

Setting the counter with "DI set" and zero mark

If you assign parameters so that the counter is set by the encoder zero mark ("Evaluate zero mark for setting"), the counter will be set to the load value at the rising edge of the zero mark.

Setting is performed only if the "DI set" input is also set at the time of the rising edge of the zero mark.

You can define the response of the FM 350-1 in the event of a zero mark rising edge with the variables ENSET_UP and ENSET_DN in the DB of the CNT_CTL1 instruction and by assigning parameters.

Input parameters	Behavior of the FM 350-1
ENSET_UP set	The counter is only set if counting up.
ENSET_DN set	The counter is only set if counting down.
ENSET_UP and ENSET_DN set	The counter is set for counting up and counting down.
"Single setting of counter" parameter assignment	The counter is only set in response to the first rising edge of the zero mark. If the counter is to be set again, you must first set ENSET_UP or ENSET_DN again (edge evaluation). The counter is then set again in response to the next rising edge of the zero mark.
"Multiple setting of counter" parameter assignment	The counter is set at every rising edge of the zero mark as long as ENSET_UP and/or ENSET_DN are set.

Note

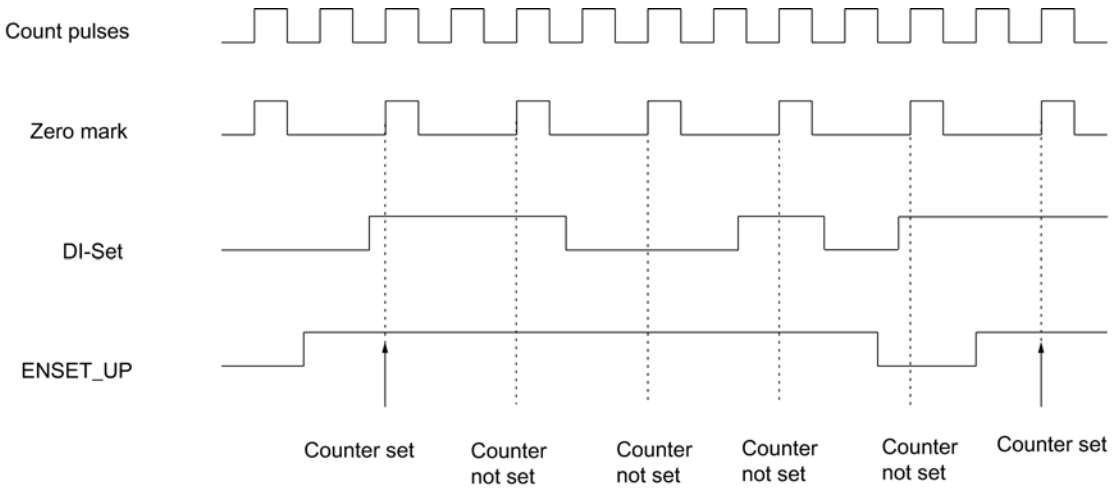
It is imperative that you set one of the two variables ENSET_UP and ENSET_DN so that the counter can be set with the zero mark.

Single setting with "DI set" and zero mark

The figure below shows single setting of the counter with the zero mark. In the case shown, only ENSET_UP is set, i.e. the counter is set when counting up.

The counter is set upon the first rising edge of the zero mark, provided that ENSET_UP and "DI set" are also set.

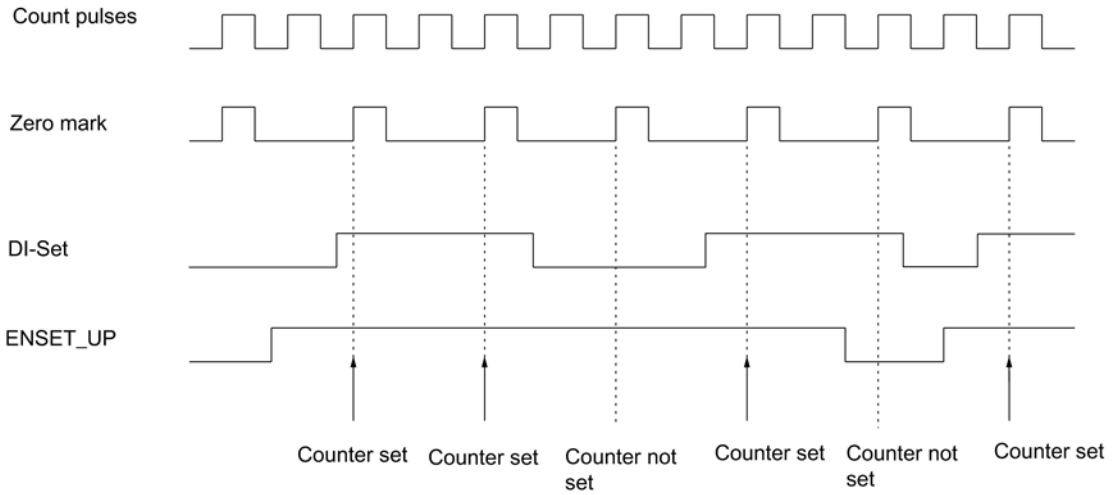
If you want to set the counter again, you must first reset ENSET_UP and then set it again. If "DI set" is not set, setting is performed upon the first zero mark after "DI set" is set. If "DI set" is set, setting is performed with the next zero mark.



Multiple setting with "DI set" and zero mark

The figure below shows multiple setting of the counter with the zero mark. In the case shown, only ENSET_UP is set, i.e. the counter is set when counting up.

The counter is set with every rising edge of the zero mark provided that ENSET_UP and "DI set" are set.



Measuring times between two edges

Introduction

You can measure the time intervals between two successive edges at the "DI start" digital input.

Requirements

Ensure the following conditions are met before measurement:

- Do not connect any encoders to the FM 350-1.
- Configure a counting mode (Page 9114) under operating modes.
- Configure the "Latch/Retrigger" (Page 9115) gate control under operating modes.
- Configure the signal type "Internal time base 1 MHz" (Page 9121) under encoder.

Selecting edges

To measure the time between two immediately successive ...	... Parameterize
Rising edges at "DI start"	Latch with positive edge
Falling edges at "DI start"	Latch with negative edge
Any edges at "DI start"	Latch with both edges

See also: Selecting an edge for latching (Page 9120)

Function principle

The FM 350-1 uses an internal time base 1 MHz to measure times. The time measurement starts upon the first edge at "DI start". At every subsequent edge at "DI start", the time in µs that has elapsed since the last edge was received is saved as the latch value LATCH_LOAD in the feedback interface.

Outputs

Defining behavior of the outputs in counting modes

Comparison values and outputs

Description

You can store two comparison values (Comparison value 1 and 2) in the FM 350-1. These comparison values are assigned to the two digital outputs (Comparison value 1: DO0,

Comparison value 2: DO1). The comparison values are compared with the current counter value. When the counter value reaches a comparison value, the relevant output can be set.

Both comparison values are assigned to DO1 in "Switch at comparison values (Page 9136)".

Comparison values 1 and 2

You enter the two comparison values in the DB of the CNT_CTL1 instruction (CMP_V1, CMP_V2) and transfer them to the FM 350-1 by setting the bits T_CMP_V1 or T_CMP_V2. This does not affect counting.

The comparison values must be within the limits of the selected counting range. The comparison value is interpreted according to the selected counting range. If you specify FFFF FFFF H as the comparison value, for example, this value will be interpreted as 4,294,967,295 in the "0 to +32 bits" counting range and as -1 in the"-31 to +31 bits" counting range.

The following ranges of values are permitted for the comparison values:

Value ranges for comparison values	Main count direction		
	None	Up	Down
Low limit	Maximum low count limit	-2^{31}	1
High limit	Maximum high count limit	Parameterized high count limit -1	$2^{31} - 1$

Enabling the outputs

Before the outputs can be set, you must first enable them by setting the relevant bits in the DB. If you reset one of these bits, the associated output will be switched off immediately even if you have configured a pulse duration for it.

The following bits must be set to enable the outputs:

Output	... is enabled by enable bit
DO0	CTRL_DO0
DO1	CTRL_DO1

Setting and resetting the outputs

If you have configured "No comparison" for output behavior, you can set and reset enabled outputs using the corresponding bits in the DB.

Outputs are set and reset with the following bits:

Output	... is set by	... is reset by
DO0	SET_DO0 = 1	SET_DO0 = 0
DO1	SET_DO1 = 1	SET_DO1 = 0

Behavior of the outputs in counting modes

Specifying the behavior of the outputs

Behavior of the DO0 and DO1 outputs

Select the required behavior for the output in question: Activate the corresponding option button:

- Substitute value "1" at CPU STOP (Page 9133)
- No comparison (Page 9133)
- On when counter value >= comparison value (Page 9133)
- On when counter value <= comparison value (Page 9134)
- Active on reaching the comparison value for pulse duration up (Page 9135)
- Active on reaching the comparison value for pulse duration down (Page 9135)
- Active on reaching the comparison value for pulse duration up/down (Page 9134)
- Switch at comparison values (Page 9136) (only possible for output DO1)

Note

To activate the settings, enter values for comparison values 1 and 2 in the DB with the counter data and transfer them to the module with the CNT_CTL1 instruction; you must also set the CTRL_DO0 and CTRL_DO1 bits in the DB in order to enable the outputs. See also "Comparison values and outputs (Page 9130)".

Output default setting

The default setting for the outputs is deactivated.

Behavior of the digital outputs in isochrone mode

In isochrone mode, the outputs DO0 and DO1 switch as soon as the comparison conditions are fulfilled, and are thus independent of the PROFIBUS DP cycle.

Exception:

If you have configured "No comparison" for the behavior of the digital outputs and, after enabling with CTRL_DO0 or CTRL_DO1, you activate the outputs with the control signals SET_DO0 or SET_DO1, the outputs will be set and reset at the time T_o.

Substitute value "1" at CPU STOP

Substitute value "1" at CPU STOP

If you selected the "Substitute value" option under "Reaction to CPU STOP" in the module's basic parameters and the CPU goes into STOP mode, the module will output the substitute values you configured in this dialog at the digital outputs. Check the checkbox for DO0, for example, and output DO0 will be set at CPU STOP.

The substitute values are retained at the digital outputs after the CPU transition from STOP to RUN until the next value is input.

The outputs are reset if you change the "Reaction to CPU STOP" basic parameter.

⚠ CAUTION
Danger of damage to equipment. Parameterized substitute values are always output when the CPU goes into STOP mode even if the gate is closed and even at outputs that are not enabled. Only configure substitute values which will not lead to dangerous states in the system if output.

No comparison

No comparison

The output remains deactivated and is not affected by comparison value, zero pass, high limit violated or low limit violated events. The output DOx may be used as a purely digital output. When enabled, it can be set and reset with the SET_DOx bit.

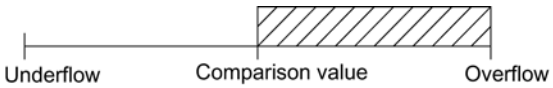
On when counter value >= comparison value

On when counter value >= comparison value

The output is set to 1 if the counter is in the range between the comparison value and high limit violated.

Setting the counter to a value between the comparison value and high limit violated sets the output to 1.

The hatched area in the graphic below means that the output is set in this range.



Please note the basic conditions for the behavior of the outputs (Page 9137).

See also

Effect of the hysteresis (Page 9138)

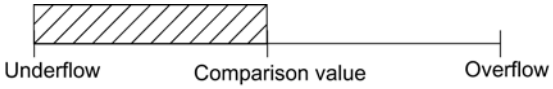
On when counter value <= comparison value

On when counter value <= comparison value

The output is set to 1 if the counter is in the range between the comparison value and low limit violated.

Setting the counter to a value between the comparison value and low limit violated sets the ouput to 1.

The hatched area in the graphic below means that the output is set in this range.



Please note the basic conditions for the behavior of the outputs (Page 9137).

See also

Effect of the hysteresis (Page 9138)

Active on reaching the comparison value for pulse duration up/down

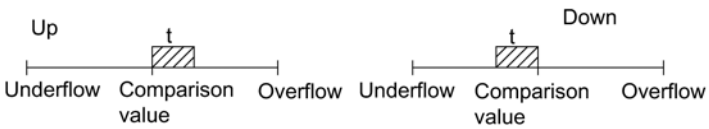
Active on reaching the comparison value for pulse duration up/down

The output is set to 1 for the duration of the pulse when the counter has reached the comparison value independently of the count direction.

Requirements:

- Main count direction: "None"

The hatched area in the graphic below means that the output is set in this range (t = pulse duration).



Active on reaching the comparison value for pulse duration up

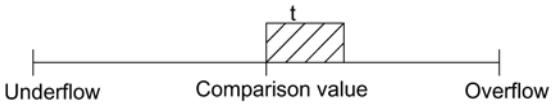
Active on reaching the comparison value for pulse duration up

The output is set to 1 for the pulse duration when the counter reaches the comparison value when counting up.

This requires one of the following:

- Main count direction: "None"
- Main count direction: "Up"

The hatched area in the graphic below means that the output is set in this range (t = pulse duration).



Please note the basic conditions for the behavior of the outputs (Page 9137).

See also

Effect of the hysteresis (Page 9138)

Active on reaching the comparison value for pulse duration down

Active on reaching the comparison value for pulse duration down

The output is set to 1 for the pulse duration when the counter reaches the comparison value when counting down.

This requires one of the following:

- Main count direction: "None"
- Main count direction: "Down"

The hatched area in the graphic below means that the output is set in this range (t = pulse duration).

Please note the basic conditions for the behavior of the outputs (Page 9137).

See also

Effect of the hysteresis (Page 9138)

Switch at comparison values

Switch at comparison values

Output DO1 switches at two comparison values if the following conditions are met:

- You have configured "No comparison" for DO0.
- You have configured "Switching at comparison values" for DO1.
- You have loaded the two comparison values CMP_V1 and CMP_V2 .
- You have enabled output DO1 with CRTL_DO1.

The following table shows when DO1 is switched on or off:

Comparison values CMP_V1 and CMP_V2	DO1 is switched on when	DO1 is switched off when
CMP_V1 < CMP_V2	CMP_V1 ≤ Counter value ≤ CMP_V2	Counter value < CMP_V1 or counter value > CMP_V2
CMP_V1 = CMP_V2	CMP_V1 = Counter value = CMP_V2	CMP_V1 ≠ Counter value ≠ CMP_V2
CMP_V1 > CMP_V2	Counter value < CMP_V2 or counter value > CMP_V1	CMP_V2 ≤ Counter value ≤ CMP_V1

The result of the comparison is indicated by the STS_COMP2 status bit.

You cannot acknowledge and reset the STS_COMP2 status bit until the comparison condition is no longer met.

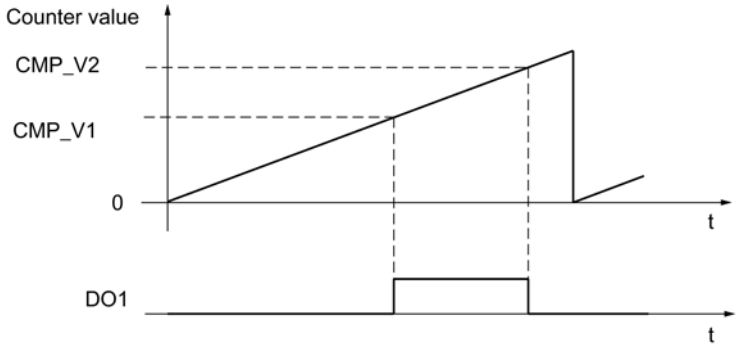
The status at output DO1 is indicated by the STS_CMP2 status bit.

There is no hysteresis in the case of this output behavior.

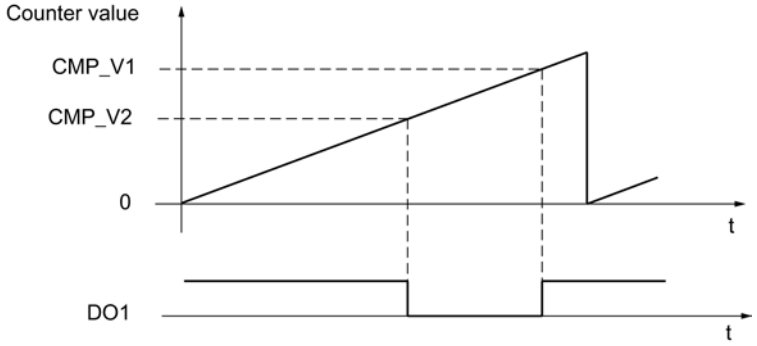
It is not possible to control the DO1 output with the SET_DO1 control bit in the case of this output behavior.

The figures below show the status of output DO1 as determined by the counter value and comparison values:

- CMP_V2 > CMP_V1 at the start of counting



- CMP_V2 < CMP_V1 at the start of counting



Basic conditions for the behavior of the digital outputs

Basic conditions for the behavior of the digital outputs

You must observe the following basic conditions when configuring the behavior of the digital outputs:

If...	then...
... ...you want to configure an output for "On when counter value >= comparison value or <= comparison value",	... ...you must ensure that the time between these events is greater than the minimum switching time of the outputs (switching time: 300 µs); the control pulses at the outputs will otherwise be lost. If the counter value reaches the relevant comparison value again while the output is still active, no new pulse will be initiated. A further pulse can only be initiated when the output is no longer active.
... you want to configure an output "On when counter value >= comparison value",	... you must not enable a hardware interrupt on "Reaching associated comparison value up or down".
... you want to configure an output "On when counter value <= comparison value",	... you must not enable a hardware interrupt on "Reaching associated comparison value up or down".

If...	then...
... you want to configure an output that is "Active on reaching the comparison value for pulse duration up",	... you must not enable a hardware interrupt on "Associated comparison value down is reached".
... ...you want to configure an output "Active on reaching comparison value for pulse duration down",	... ...you must not enable a hardware interrupt on "Reaching the relevant comparison value up".

Effect of the hysteresis

Function of hysteresis with parameter assignment "On when counter value >= comparison value or <= comparison value"

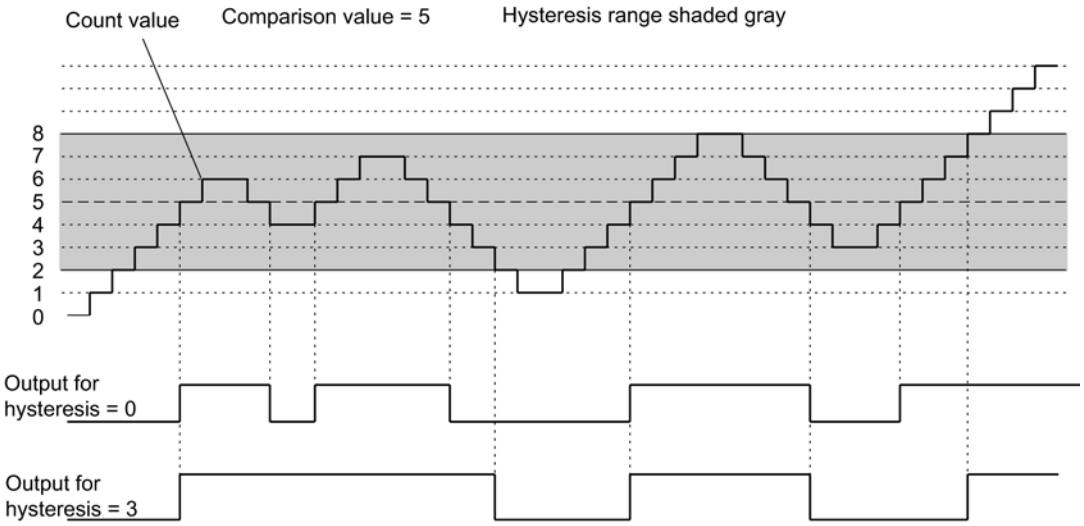
The example in the figure below demonstrates the effect of hysteresis. The figure shows the differences in the behavior of an output when hysteresis of 0 (= switched off) is parameterized as opposed to hysteresis of 3. In the example, the comparison value = 5

Parameter assignment in this example is as follows:

- Main count direction: "Up"
- Output response: "On when counter value >= comparison value"

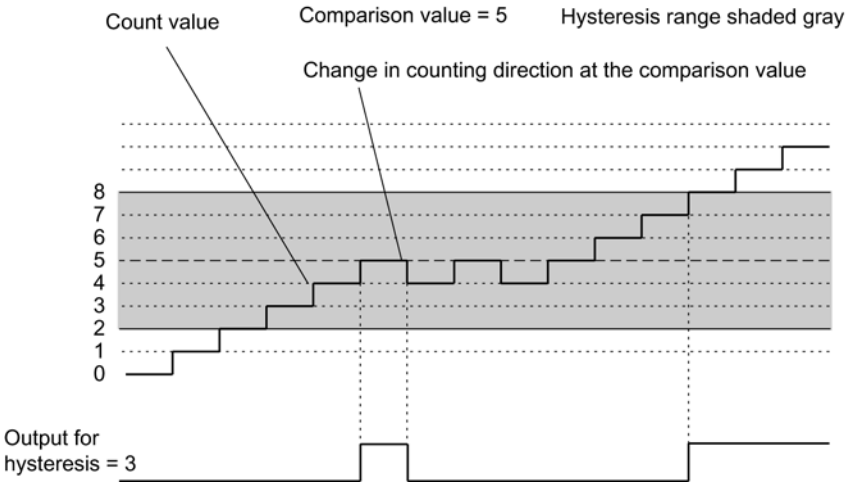
The hysteresis is set when the comparison condition (counter value = 5) is fulfilled. When the hysteresis is active, the comparison result remains unchanged.

If the count value overshoots/undershoots the range of the hysteresis (at counter value 2 or 8 in the example) the hysteresis ceases to be active. The comparator switches again according to its comparison conditions, i.e., in the example at counter value 5.



Note

If the counter value is equal to the comparison value, and the hysteresis is active, FM 350-1 resets the output when the count direction changes at the comparison value (see the diagram below).



Effect of hysteresis with parameter assignment "Active on reaching the comparison value for pulse duration up or down"

The example in the figure below demonstrates the effect of hysteresis. The figure shows the differences in the behavior of an output when hysteresis of 0 (= switched off) is parameterized as opposed to hysteresis of 3. In the example, the comparison value = 5

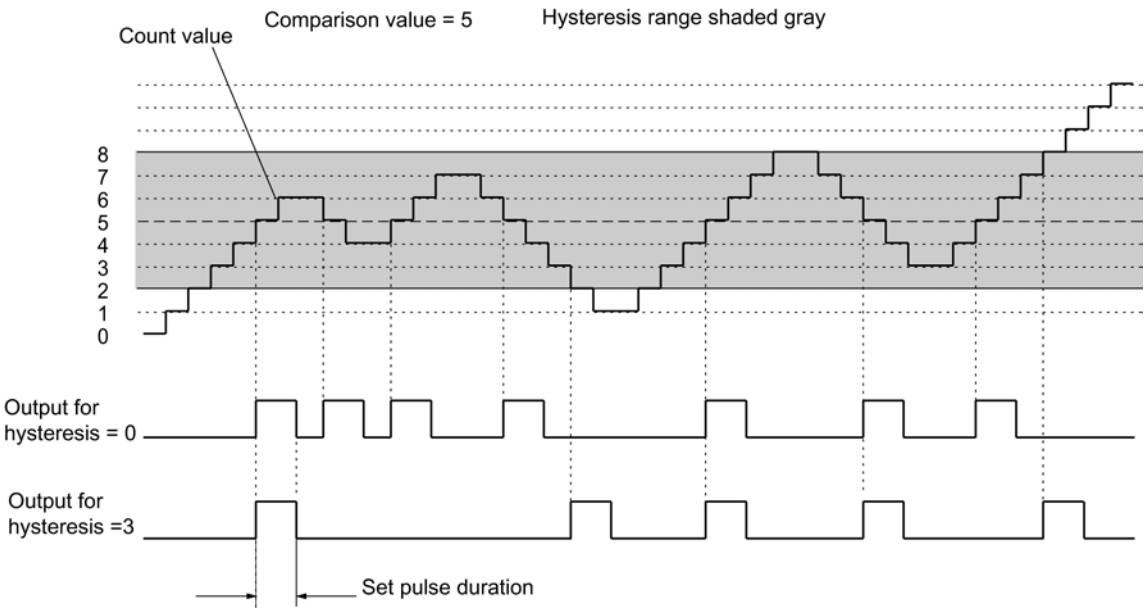
Parameter assignment in this example is as follows:

- Main count direction: "None"
- Output response: "Active on reaching the comparison value for pulse duration up"
- Pulse duration > 0

The hysteresis is set when the comparison condition (counter value = 5) is fulfilled, and a pulse of the programmed duration is output.

If the counter value leaves the hysteresis range, the hysteresis ceases to be active.

When hysteresis is enabled, the FM 350-1 stores the count direction. A pulse is output if the signal overshoots hysteresis range in opposite direction of the direction saved previously.



Defining behavior of the outputs in measuring modes

Limits and outputs

Low and high limit

You can save an high and a low limit for frequency, speed and period measurements and activate digital output DO0 if one of these limits is exceeded. You can configure these limits and change them via the control interface.

The following value ranges are possible for the low and high limits:

Measuring mode	Low limit (LL)	High limit
Frequency measurement (5 V encoders)	0 to 499,999,999 × 10^{-3} Hz	LL+1 to 500,000,000 × 10^{-3} Hz
Frequency measurement (24 V encoders)	0 to 199,999,999 × 10^{-3} Hz	LL+1 to 200,000,000 × 10^{-3} Hz
Rotational speed	0 to 24,999,999 × 10^{-3} /min	LL+1 to 25,000,000 × 10^{-3} /min
Period measurement (resolution = 1 µs)	0 to 119,999,999 µs	LL+1 to 120,000,000 µs
Period measurement (resolution = 1/16 µs)	0 to 1,919,999,999 µs	LL+1 to 1,920,000,000 µs

Enabling the outputs

Before the outputs can be set, you must first enable them by setting the relevant bits in the DB. If you reset one of these bits, the associated output will be switched off immediately.

Output	... is enabled by enable bit
DO0	CTRL_DO0
DO1	CTRL_DO1

Behavior of the outputs in measuring modes

Behavior of the digital outputs

Digital output DO0

You can select one of four possible responses to reaching limits for digital output DO0. The various options are shown in the table below.

Digital output DO0 parameter assignment	Behavior of digital output DO0	Switching time	
		Isochrone mode	Non-isochrone mode
No comparison	Not affected by limit monitoring. If output DO0 is already set, it can be reset by changing the parameter to "No comparison". You can use output DO0 freely as a digital output and set and reset it with the control signal SET_DO0 if you have enabled it with the control signal CTRL_DO0.	At time T_o.	Immediately after setting or resetting the output
Outside limits	DO0 is set in each of the following cases: • measured value < low limit • measured value > high limit	At the end of the update time at time T_i	At the end of the update time
Violating low limit	DO0 is set if • measured value < low limit		
Exceeding high limit	DO0 is set if • measured value > high limit		

Select how you want output DO0 to respond. Activate the corresponding option button.

Digital output DO1

You can use output DO1 freely as a digital output and set and reset it with the control signal SET_DO1 provided it has been enabled.

DO1 is not affected by limit monitoring.

In isochrone mode, DO1 switches at the end of the update time at time T_o.

Specifying pulse duration

Pulse duration

You can set a pulse duration in order to adapt to the actuators (contactors, actuators, etc) used in your process. The pulse duration indicates how long outputs DO0 and DO1 are active on reaching a comparison value.

The pulse duration only affects the main count direction when "Main count direction up" or "Main count direction down" has been assigned.

The pulse duration can affect both main count directions if a main count direction has not been parameterized.

The pulse duration starts when the output is set. The pulse duration inaccuracy is less than 1 ms.

A value between 0 and 500 ms may be set for the pulse duration. This value applies to both outputs.

If the pulse duration = 0 ms, the output is set when the comparison value is reached and is reset at the next count pulse.

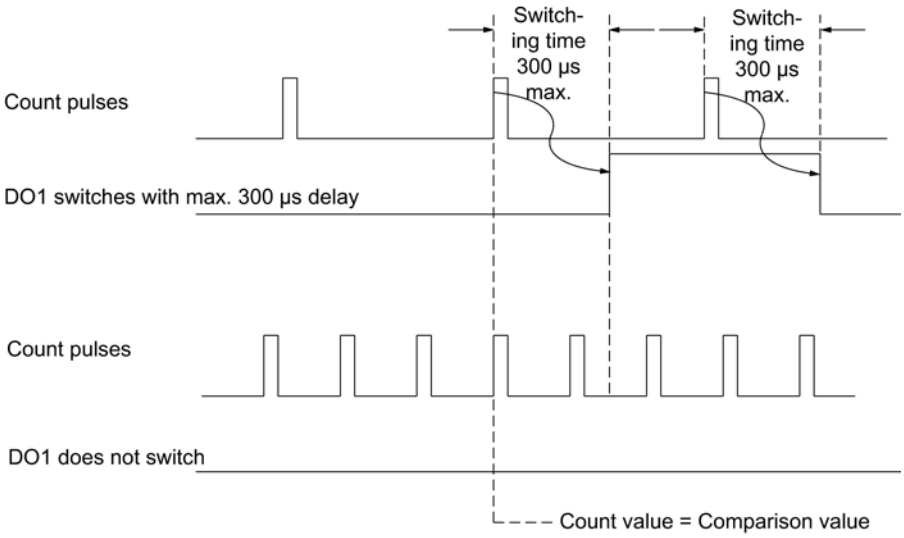
The value 0 is set by default for the pulse duration.

Note

If you specify pulse duration = 0 ms, the output will remain active as long as the counter value equals the comparison value (see top half of figure below).

Control pulses may be lost at the outputs if the time interval between the count pulses is less than the switching times of the digital outputs (up to 300 µs) (see bottom half of figure below).

You should therefore make sure that the interval between the count pulses is greater than the switching times of the digital outputs.



Setting hysteresis

Function principle of the hysteresis

An encoder can stop at a certain position, and then "oscillate" around this position. This state causes the counter level to fluctuate around a certain value. If there is a comparison value within this fluctuation range, for example, the associated output would be switched on and off in time with these fluctuations. The FM 350-1 has configurable hysteresis to prevent this switching in response to small fluctuations.

A value between 0 and 255 can be set for hysteresis.

The effect of hysteresis is as follows in accordance with the value parameterized:

Hysteresis	Effect
Hysteresis value n = 0, 1	The hysteresis has no effect (switched off) The output reacts to the slightest change in the counter value.
2 ≤ Hysteresis value n ≤ 255	The hysteresis takes effect. The output does not react until the counter value is offset by n units to the comparison value.

The hysteresis applies to both high limit violated and low limit violated.

Enable hardware interrupts

Enabling hardware interrupt trigger

Introduction

The FM 350-1 allows you to set which events are to trigger a hardware interrupt during operation.

What is a Hardware Interrupt?

If you want to set a response to a specific event independently of the CPU cycle, the FM 350-1 can initiate a hardware interrupt. The CPU interrupts the cyclic program on receiving the interrupts and executes the hardware interrupt OB.

Enabling the Hardware Interrupt

Hardware interrupts are enabled in two steps:

1. When configuring the hardware, you enable the interrupts for the module in the basic parameters and choose whether the module is to trigger diagnostic and/or hardware interrupts.
2. You then select the relevant events under "Enable hardware interrupts":
   - Events to trigger a hardware interrupt in counting modes (Page 9144)
   - Events to trigger a hardware interrupt in measuring modes (Page 9145)

Hardware Interrupt OB, OB 40

If a hardware interrupt occurs, the user program is interrupted, the data is transferred from the module to the start information of OB40 (Page 9145) and OB40 is called. The hardware interrupt is acknowledged by exiting OB 40.

If no OB 40 is programmed, the CPU goes into STOP mode. If you then switch back to RUN, the hardware interrupt requirements are deleted.

Lost hardware interrupt

If an event occurs that is to initiate a hardware interrupt and the same previous event has not yet been acknowledged, no further hardware interrupt is initiated; the hardware interrupt is lost. This may lead to the "Hardware interrupt lost" diagnostic interrupt, depending on the parameterization.

Default setting

No hardware interrupts are enabled in the default setting.

Events to trigger a hardware interrupt in counting modes

Which events can trigger a hardware interrupt in counting modes?

The following events can trigger a hardware interrupt in counting modes during operation of the FM 350-1:

- Opening of the gate (in the operating modes with hardware or software gate)
- Closing of the gate (in the operating modes with hardware or software gate)
- Violation of the high limit
- Violation of the low limit
- Zero pass
- Reaching comparison value 1 or 2 in the up direction
- Reaching Comparison value 1 or 2 in the backward direction

- Setting the counter (DI set)
- Latching

Select the check box for the relevant events.

You can select any number of events for triggering a hardware interrupt. Observe the basic conditions for output behavior (Page 9137) applicable to hardware interrupts on reaching a comparison value.

Events to trigger a hardware interrupt in measuring modes

Which events can trigger a hardware interrupt in measuring modes?

The following events can trigger a hardware interrupt in measuring modes during operation of the FM 350-1:

- Opening of the gate (in modes with hardware or software gate)
- Closing of the gate (in operating modes with hardware or software gate)
- Violating low limit
- Exceeding high limit
- End of measurement

Select the check box for the relevant events.

You can select any number of events for triggering a hardware interrupt.

OB 40 start information

OB 40 start information

The temporary variable OB40_POINT_ADDR is described in the start information of OB 40.

The OB40_POINT_ADDR variable (bytes 8 to 11) comprises 4 bytes. The information about the event which has triggered the hardware interrupt is entered in bytes 8 and 9.

The table shows which bits are set for which interrupt. Any bits not listed are irrelevant and are zero.

Byte	Bit	Meaning: Interrupt in the case of...
8	0	Opening the gate
	1	Closing the gate
	2	Violation of the high limit (for count mode)
		Measured value outside limits (in measuring mode)
	3	Violation of the low limit (for count mode)
		End of measurement (at Measure mode)
	4	Reaching comparison value 1 in the up direction
	5	Reaching comparison value 1 in the down direction
	6	Reaching comparison value 2 in the up direction

Byte	Bit	Meaning: Interrupt in the case of...
	7	Reaching comparison value 2 in the down direction
9	0	Zero pass
	5	Setting the counter (DI set) (synchronization)
	7	Latching

Control and feedback interface of the FM 350-1

Overview

If you want to operate the FM 350-1 without instructions, you can control and monitor the FM 350-1 directly via the control and feedback interface (user data interface).

The user data interface begins with the start address of the module and is 16 bytes long.

You can read the feedback interface with the load commands.

You can write to the control interface with the transfer commands.

You cannot mix load/transfer commands and the programming with instructions.

Control interface for the count modes

Control interface for the count modes

Parameter LOAD_VAL (bytes 0 to 3) has two meanings:

- If you set the L_DIRECT or L_PREPAR bit, LOAD_VAL is interpreted as a load value.
- If you set the C_DOPARA bit, you can define the behavior of outputs DO0 and DO1 in byte 0; bytes 1 and 2 are interpreted as hysteresis and pulse duration.

The table below shows the assignment of the FM 350-1 control interface (outputs) for the count modes.

Offset to the start address	Parameters	Meaning				
Bytes 0 to 3	LOAD_VAL	Load value; direct and preparatory loading with bit L_DIRECT Load value; preparatory loading with bit L_PREPAR				
Byte 0		Characteristics of the DO0 and DO1 outputs, hysteresis and the pulse duration; defined with the C_DOPARA bit				
		Bit 3	Bit 2	Bit 1	Bit 0	Behavior of output DO0
		x	0	0	0	No comparison
		x	0	0	1	On when counter value >= comparison value
		x	0	1	0	On when counter value <= comparison value
		x	0	1	1	Active on reaching the comparison value for pulse duration up/down
		x	1	0	0	Active on reaching the comparison value for pulse duration up

Offset to the start address	Parameters	Meaning				
		x	1	0	1	Active on reaching the comparison value for pulse duration down
		Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Behavior of output DO1**
		x	0	0	0	No comparison
		x	0	0	1	On when counter value >= comparison value
		x	0	1	0	On when counter value <= comparison value
		x	0	1	1	Active on reaching the comparison value for pulse duration up/down
		x	1	0	0	Active on reaching the comparison value for pulse duration up
		x	1	0	1	Active on reaching the comparison value for pulse duration down
		x	1	1	0	Switch to Comparison value
		x = irrelevant				
Byte 1		Hysteresis (value range 0 to 255)				
Byte 2		Pulse duration (value range 0 to 250)				
Byte 3		Reserve = 0				
Bytes 4 to 7	CMP_V1	Comparison value 1; load with the T_CMP_V1 bit				
Bytes 8 to 11	CMP_V2	Comparison value 2; load with the T_CMP_V2 bit				
Byte 12	– NEUSTQ – – OT_ERR_A – – –	Bit 7: Reserve = 0 Bit 6: Restart acknowledgement Bit 5: Reserve = 0 Bit 4: Reserve = 0 Bit 3: Operator error acknowledgement Bit 2: Reserve = 0 Bit 1: Reserve = 0 Bit 0: Reserve = 0				
Byte 13	– – – – SW_GATE GATE_STP ENSET_DN ENSET_UP	Bit 7: Reserve = 0 Bit 6: Reserve = 0 Bit 5: Reserve = 0 Bit 4: Reserve = 0 Bit 3: SW gate control bit Bit 2: General gate stop Bit 1: Enable synchronization down Bit 0: Enable synchronization up				
Byte 14	– – – – SET_DO1 SET_DO0 CTRL_DO1 CTRL_DO0	Bit 7: Reserve = 0 Bit 6: Reserve = 0 Bit 5: Reserve = 0 Bit 4: Reserve = 0 Bit 3: Control bit DO1 Bit 2: Control bit DO0 Bit 1: Enable DO1 Bit 0: Enable DO0				

Offset to the start address	Parameters	Meaning
Byte 15	–	Bit 7: Reserve = 0
	C_DOPARA[1]	Bit 6: Change function of DO0/DO1, hysteresis or pulse duration
	RES_ZERO	Bit 5: Reset status bits for zero pass, violation of high and low limit, and comparator
		Bit 4: Reset synchronization status bit
	RES_SYNC	Bit 3: Load comparison value 2
	T_CMP_V2[2]	Bit 2: Load comparison value 1
	T_CMP_V1[2]	Bit 1: Load counter in preparation
	L_PREPAR[2]	Bit 0: Direct and preparatory loading of counter
	L_DIRECT[2]	
[1] Do not set this bit at the same time as bits 0, 1, 2 or 3 of byte 15. [2] Do not set this bit at the same time as bit 6 of byte 15.		

Explanation of the control bits for the count modes

Control bit	Explanation
C_DOPARA	Set this bit to change the function and behavior of DO0 and DO1, and the hysteresis and pulse duration. The values of bytes 0 to 2 are accepted as a new function, hysteresis and pulse duration at DO0 and DO1. Transfer the old values if you want to discard all changes.
CTRL_DO0	Enable DO0 Set this bit to enable output DO0.
CTRL_DO1	Enable DO1 Set this bit to enable output DO1.
ENSET_DN	Set this bit to enable loading of the counter for up counts.
ENSET_UP	Set this bit to enable loading of the counter for down counts.
GATE_STP	Set this bit to close the internal gate.
L_DIRECT	Set this bit to enable direct and preparatory loading of the counter.
L_PREPAR	Set this bit to enable preparatory loading of the counter.
NEUSTQ	Set this bit to acknowledge a start of the FM 350-1. After its restart, FM 350-1 does not recognize any control or data input unless this bit is set. The NEUSTQ bit is set by the CNT_CNTL1 instruction as soon as the feedback signal FM_NEUST is set and the feedback signal is FM_NEUSTQ = 0. It is reset by the CNT_CNTL1 instruction when the FM_NEUST bit is reset by the FM 350-1 and the FM_NEUSTQ bit has been set by the FM 350-1. If you do not use the CNT_CNTL1 instruction, you need to coordinate the restart via the user program.
OT_ERR_A	Set this bit to acknowledge an operator error. For detailed information on operator errors, read the feedback interface before you acknowledge the error. The error message is no longer valid after its acknowledgement.
RES_SYNC	Use this bit to reset and acknowledge the feedback bit STS_SYNC and thereby enable loading of the counter by synchronization input DI-Set.
RES_ZERO	Use this bit to reset the feedback bits STS_ZERO, STS_OFLW, STS_UFLW, STS_COMP1 and STS_COMP2.

Control bit	Explanation
SET_DO0	Provided you have set the "No comparison" output behavior and the CRTL_DO0 enable bit is set, you can use this bit to toggle the digital output DO0 on and off.
SET_DO1	Provided you have set the "No comparison" output behavior and the CRTL_DO1 enable bit is set, you can use this bit to toggle the digital output DO1 on and off.
SW_GATE	Set/reset this bit to open/close the SW gate.
T_CMP_V1	Set this bit to load the value of bytes 4 to 7 to comparison value 1.
T_CMP_V2	Set this bit to load the value of bytes 8 to 11 to comparison value 2.

Checkback interface for count modes

Feedback interface for count modes

The table below shows the assignment of the FM 350-1 feedback interface (inputs) for the count modes.

Offset to the start address	Parameters	Meaning
Bytes 0 to 3	LATCH_LOAD	Load value which can be returned, or stored counter value for the latch function at the digital input
Bytes 4 to 7	ACT_CNTV	Counter value
Bytes 8 to 9	DA_ERR_W	Data error
Byte 10	OT_ERR_B	Operator error
Byte 11	PARA FM_NEUST FM_NEUSTQ DATA_ERR OT_ERR DIAG – –	Bit 7: Parameter assignment done Bit 6: Restart request Bit 5: Restart acknowledgement done Bit 4: Data error Bit 3: Operator error Bit 2: Diagnostics event Bit 1: Reserve = 0 Bit 0: Reserve = 0
Byte 12	–	Reserve = 0
Byte 13	STS_SW_GATE STS_GATE STS_SYNC STS_UFLW STS_OFLW STS_ZERO STS_DIR STS_RUN	Bit 7: Software gate status Bit 6: Gate status Bit 5: Synchronization Bit 4: Violation of the low limit Bit 3: Violation of the high limit Bit 2: Zero pass Bit 1: Direction bit Bit 0: Counter active

Offset to the start address	Parameters	Meaning
Byte 14	STS_COMP2 STS_COMP1 STS_CMP2 STS_CMP1 STS_STP STS_STA STS_LATCH STS_SET	Bit 7: Latched state of comparator 2 Bit 6: Latched state of comparator 1 Bit 5: Status at output DO1 Bit 4: Status at output DO0 Bit 3: Status at digital input DI-Stop Bit 2: Status at digital input DI-Start Bit 1: New latch value for isochronous mode Bit 0: Status at digital input DI-Set
Byte 15	– STS_C_DOPARA STS_RES_ZERO STS_RES_SYNC STS_T_CMP_V2 STS_T_CMP_V1 STS_L_PREPAR STS_L_DIRECT	Bit 7: Reserve = 0 Bit 6: Change function of DO0/DO1, hysteresis or pulse duration Bit 5: Reset status bit for zero pass, violation of high and low limit, or comparator Bit 4: Reset synchronization status bit Bit 3: Load comparison value 2 Bit 2: Load comparison value 1 Bit 1: Load counter in preparation Bit 0: Direct and preparatory loading of counter

Description of the feedback bits for the count modes

Feedback bit	Explanation
DATA_ERR	This bit indicates a faulty data (parameter assignment error) entry in the feedback interface.
DIAG	The bit will be set if diagnostics data record DS1 was updated to signal a diagnostics event. The bit will be reset when data record DS1 has been read. If no diagnostics interrupt is enabled, this bit may be used as init bit for the DIAG_INF instruction embedded in OB1.
FM_NEUST	FM 350-1 sets this bit each time it performs a restart or detects a system startup, regardless of whether the system starts up automatically or manually. The FM_NEUST bit is reset at the next positive edge at the NEUSTQ bit. FM 350-1 then accepts control commands and allows the input and output of values.
FM_NEUSTQ	FM 350-1 clears this bit each time it performs a restart or detects a system startup, regardless of whether the system starts up automatically or manually. It is set after you reset the FM_NEUST bit.
OT_ERR	This bit will be set after an operator error was logged at the feedback interface. It is reset when the OT_ERR_A bit is set. As long as OT_ERR bit is set, the function does not report any additional operator errors.
PARA	This bit will be set when the module parameters are free of errors. The parameter data record on the module does not contain any errors. This bit is only set when the FM_NEUSTQ bit is set. From this moment on, the values in the feedback interface are valid and up to date.
STS_C_DOPARA	Bit used to acknowledge simultaneous changes of the behavior of DO0 and DO1, and of the hysteresis and pulse duration. Transfer the old values if you want to discard all changes.
STS_CMP1	Status at output DO0
STS_CMP2	Status at output DO1
STS_T_CMP_V1	Bit used to acknowledge loading of comparison value 1
STS_T_CMP_V2	Bit used to acknowledge loading of comparison value 2

Feedback bit	Explanation
STS_COMP1	This bit indicates the stored status that output DO0 was set. That also applies if output DO0 was not enabled with CTRL_DO0. The saved status is reset by acknowledgment with RES_ZERO.
STS_COMP2	This bit indicates the stored status that output DO1 was set. That also applies if output DO1 was not enabled with CTRL_DO1. The saved status is reset by acknowledgment with RES_ZERO.
STS_DIR	This bit indicates the count direction of the counter: 0 = Up (LED DIR is off) 1 = Down (LED DIR is lit)
STS_GATE	This bit indicates the gate status. 0 = Gate closed 1 = Gate open
STS_LATCH	In isochronous mode, this bit indicates whether at least one new latch value was saved between the one to last Ti and the last Ti. If the bit is set, the last latch value is in LATCH_LOAD. The bit will not be set if no new latch value was saved. The bit is not set in non-isochronous mode.
STS_L_DIRECT	Acknowledgement bit for direct and preparatory loading of the counter and load value.
STS_L_PREPAR	Acknowledgement bit for preparatory loading of the load value.
STS_OFLW	This bit indicates a high limit violation. The saved status is reset by acknowledgment with RES_ZERO.
STS_RES_SYNC	Resets the STS_SYNC feedback bit.
STS_RES_ZERO	Acknowledgement bit for resetting saved states in the feedback bits STS_ZERO, STS_OFLW, STS_UFLW, STS_COMP1 and STS_COMP2
STS_RUN	This bit corresponds to counter bit 2^0. 0 = LED CR is off 1 = LED CR is lit
STS_SET	Status at digital input DI-Set
STS_STA	Status at digital input DI-Start
STS_STP	Status at digital input DI-Stop
STS_UFLW	This bit indicates a low limit violation. The saved status is reset by acknowledgment with RES_ZERO.
STS_SYNC	This bit indicates the saved state, which shows that the counter was loaded by an event at DI-Set (synchronization). The saved status is reset by acknowledgment with RES_SYNC.
STS_ZERO	This bit indicates the saved state, which shows that the counter value has passed a zero pass. The saved status is reset by acknowledgment with RES_ZERO.

Control interface for measuring modes

Control interface for measuring modes

Parameter LOAD_VAL (bytes 0 to 3) has two meanings:

- If you set the L_PREPAR bit, LOAD_VAL is interpreted as a low limit.
- You set the C_DOPARA bit in byte 0 to define the behavior at output DO0.

The table below shows the assignment of the FM 350-1 control interface (outputs) for measuring modes.

Offset to the start address	Parameter	Assignment			
Bytes 0 to 3	LOAD_VAL	Load low limit with the L_PREPAR bit			
Byte 0		Define behavior of output DO0 with the C_DOPARA bit			
		Bits 2 to 7	Bit 1	Bit 0	Behavior of output DO0
		Irrelevant	0	0	No comparison
		Irrelevant	0	1	Outside limits
		Irrelevant	1	0	Violating low limit
		Irrelevant	1	1	Exceeding high limit
Byte 1		Reserve = 0			
Byte 2		Reserve = 0			
Byte 3		Reserve = 0			
Bytes 4 to 7	CMP_V1	High limit; load with the T_CMP_V1 bit			
Bytes 8 to 9	CMP_V2	Update time; load with the T_CMP_V2 bit			
Bytes 10 to 11	–	–			
Byte 12	– NEUSTQ – – OT_ERR_A – – –	Bit 7: Reserve = 0 Bit 6: Restart acknowledgement Bit 5: Reserve = 0 Bit 4: Reserve = 0 Bit 3: Operator error acknowledgement Bit 2: Reserve = 0 Bit 1: Reserve = 0 Bit 0: Reserve = 0			
Byte 13	– – – – SW_GATE GATE_STP – –	Bit 7: Reserve = 0 Bit 6: Reserve = 0 Bit 5: Reserve = 0 Bit 4: Reserve = 0 Bit 3: SW gate control bit Bit 2: General gate stop Bit 1: Reserve = 0 Bit 0: Reserve = 0			
Byte 14	– – – – SET_DO1 SET_DO0 CTRL_DO1 CTRL_DO0	Bit 7: Reserve = 0 Bit 6: Reserve = 0 Bit 5: Reserve = 0 Bit 4: Reserve = 0 Bit 3: Control bit DO1 Bit 2: Control bit DO0 Bit 1: Enable DO1 Bit 0: Enable DO0			

Offset to the start address	Parameter	Assignment
Byte 15	–	Bit 7: Reserve = 0
	C_DOPARA[1]	Bit 6: Change function DO0
	RES_ZERO	Bit 5: Reset status bits for violation of high limit, low limit and end of measurement
	–	Bit 4: Reserve = 0
	T_CMP_V2[2]	Bit 3: Change refresh time
	T_CMP_V1[2]	Bit 2: Load high limit
	L_PREPAR[2]	Bit 1: Load low limit
	–	Bit 0: –
[1] Do not set this bit at the same time as bit 1, 2 or 3 of byte 15. [2] Do not set this bit at the same time as bit 6 of byte 15.		

Description of the control bits for measuring modes

Control bit	Explanation
C_DOPARA	Set this bit to change the behavior and function of DO0. The values at byte 0 are accepted as new function of DO0. Transfer the old values if you want to discard all changes.
CTRL_DO0	Enable DO0 Set this bit to enable output DO0.
CTRL_DO1	Enable DO1 Set this bit to enable output DO1.
GATE_STP	Set this bit to close the internal gate.
L_PREPAR	Set this bit to load the low limit.
NEUSTQ	Set this bit to acknowledge a start of the FM 350-1. After its restart, FM 350-1 does not recognize any control or data input unless this bit is set. The NEUSTQ bit is set by the CNT_CNTL1 instruction as soon as the feedback signal FM_NEUST is set and the feedback signal is FM_NEUSTQ = 0. It is reset by the CNT_CNTL1 instruction when the FM_NEUST bit is reset by the FM 350-1 and the FM_NEUSTQ bit has been set by the FM 350-1. If you do not use the CNT_CNTL1 instruction, you need to coordinate the restart via the user program.
OT_ERR_A	Set this bit to acknowledge an operator error. For detailed information on operator errors, read the feedback interface before you acknowledge the error. The error message is no longer valid after its acknowledgement.
RES_ZERO	Use this bit to reset the feedback bits STS_OFLW, STS_UFLW and STS_COMP1.
SET_DO0	Provided you have set the "No comparison" output behavior and the CRTL_DO0 enable bit is set, you can use this bit to toggle the digital output DO0 on and off.
SET_DO1	Provided you have set the "No comparison" output behavior and the CRTL_DO1 enable bit is set, you can use this bit to toggle the digital output DO1 on and off.
SW_GATE	Set/reset this bit to open/close the SW gate.
T_CMP_V1	Set this bit to load the high limit.
T_CMP_V2	Set this bit to load the refresh interval time.

Checkback interface for the measuring modes

Feedback interface for the measuring modes

The table below shows the assignment of the FM 350-1 feedback interface (inputs) for measuring modes.

Offset to the start address	Parameter	Assignment
Bytes 0 to 3	LATCH_LOAD	Measured value
Bytes 4 to 7	ACT_CNTV	Counter value
Bytes 8 to 9	DA_ERR_W	Data error
Byte 10	OT_ERR_B	Operator error
Byte 11	PARA FM_NEUST FM_NEUSTQ DATA_ERR OT_ERR DIAG – –	Bit 7: Parameter assignment done Bit 6: Restart request Bit 5: Restart acknowledgement done Bit 4: Data error Bit 3: Operator error Bit 2: Diagnostics event Bit 1: Reserve = 0 Bit 0: Reserve = 0
Byte 12		Reserve = 0
Byte 13	– STS_GATE – STS_UFLW STS_OFLW STS_COMP1 STS_DIR STS_RUN	Bit 7: Reserve = 0 Bit 6: Gate status Bit 5: Reserve = 0 Bit 4: Low limit violated Bit 3: High limit violated Bit 2: End of measurement Bit 1: Direction bit Bit 0: Counter active

Offset to the start address	Parameter	Assignment
Byte 14	– – STS_CMP2 STS_CMP1 STS_STP STS_STA – STS_SET	Bit 7: Reserve = 0 Bit 6: Reserve = 0 Bit 5: Status at output DO1 Bit 4: Status at output DO0 Bit 3: Status at digital input DI-Stop Bit 2: Status at digital input DI-Start Bit 1: Reserve = 0 Bit 0: Status at digital input DI-Set
Byte 15	– STS_C_DOPARA STS_RES_ZERO – STS_T_CMP_V2 STS_T_CMP_V1 STS_L_PREPAR –	Bit 7: Reserve = 0 Bit 6: Change function DO0 Bit 5: Reset end of measurement status bit Bit 4: Reserve = 0 Bit 3: Change refresh time Bit 2: Load high limit Bit 1: Load low limit Bit 0: Reserve = 0

Description of the acknowledgement bits for the measuring modes

Feedback bit	Explanation
DATA_ERR	This bit indicates that a data error was entered in the feedback interface.
DIAG	The bit will be set if diagnostics data record DS1 was updated to signal a diagnostics event. The bit will be reset when data record DS1 has been read. If no diagnostics interrupt is enabled, this bit may be used as init bit for the DIAG_INF instruction embedded in OB1.
FM_NEUST	FM 350-1 sets this bit each time it performs a restart or detects a system startup, regardless of whether the system starts up automatically or manually. The FM_NEUST bit is reset at the next positive edge at the NEUSTQ bit. FM 350-1 then accepts control commands and allows the input and output of values.
FM_NEUSTQ	FM 350-1 clears this bit each time it performs a restart or detects a system startup, regardless of whether the system starts up automatically or manually. It is set after you reset the FM_NEUST bit.
OT_ERR	This bit will be set after an operator error was logged at the feedback interface. It is reset when the OT_ERR_A bit is set. As long as OT_ERR bit is set, the function does not report any additional operating errors.
PARA	This bit will be set when the module parameters are free of errors. The parameter data record on the module does not contain any errors. This bit is only set when the FM_NEUSTQ bit is set. From this moment on, the values in the feedback interface are valid and up to date.
STS_C_DOPARA	Bit used to acknowledge simultaneous changes of the behavior of DO0 and DO1, and of the hysteresis and pulse duration. Transfer the old values if you want to discard all changes.
STS_CMP1	Status at output DO0
STS_CMP2	Status at output DO1
STS_CMP_T_VAL1	Acknowledgment bit for loading the high value
STS_CMP_T_VAL2	Acknowledgement bit for loading of the refresh time

Feedback bit	Explanation
STS_DIR	This bit indicates the count direction of the counter: 0 = Up (LED DIR is off) 1 = Down (LED DIR is lit)
STS_GATE	This bit indicates the gate status. 0 = Gate closed 1 = Gate open
STS_L_PREPAR	Acknowledgment bit for loading the low value
STS_OFLW	This bit indicates the saved status, which shows that a measured value is greater than the high limit. The saved status is reset by acknowledgment with RES_ZERO.
STS_RES_ZERO	Acknowledgement bit for resetting saved states in the feedback bits STS_OFLW, STS_UFLW and STS_COMP1
STS_RUN	This bit corresponds to counter bit 2^0. 0 = LED CR is off 1 = LED CR is lit
STS_SET	Status at digital input DI-Set
STS_STA	Status at digital input DI-Start
STS_STP	Status at digital input DI-Stop
STS_UFLW	This bit indicates the saved status, which shows that a measured value is less than the low limit. The saved status is reset by acknowledgment with RES_ZERO.

Using FM 350-1 from the user program

Integrating the FM 350-1 into a user program

Instructions are available for integrating the FM 350-1 into a user program. These enable you to handle the desired functions with ease.

The instructions CNT_CTRL, CNT_CTL1 and CNT_CTL2 can be called in the CPU user program. They ensure communication between the CPU and the FM 350-1.

All data associated with a channel of the module are in the DB of the instruction CNT_CTRL, CNT_CTL1 or CNT_CTL2 (counter DB (Page 7101)). Use the command "Program blocks > Add new block" to add a CNT_CHANTYPE1 type data block to your project.

The DIAG_INF instruction is also available for FM 350-1. This operation allows you to transfer the diagnostic information to the counter DB.

The instruction CNT_CTL2 is only used in isochrone mode.

See also

CNT_CTRL (Page 7096)

DIAG_INF (Page 7097)

CNT_CTL1 (Page 7098)

CNT_CTL2 (Page 7100)

FM 350-1: Online & Diagnostics

Current module information

Default diagnostics

FM 350-1 group error LED fault display

Where is the Fault Indicated?

If the red group error LED lights up, a fault has occurred either on the module (internal fault) or at the cable connections (external fault).

Which Errors Are Displayed?

The table below shows the error types displayed by the group error LED.

Type of error	Cause of the error	Correction
Internal faults	Error in EPROM test	Module replacement
	RAM test error	Module replacement
	Watchdog tripped	Module replacement
	Lost hardware interrupt	Increase time between the interrupt causes
	Module parameterization missing	Assign parameters and transfer
External errors	Coding plug incorrectly connected	Correct the position of the coding plug
	Auxiliary voltage 1L+/1M not connected or DC24V encoder supply short-circuited	Correct connection
	DC5.2V encoder supply short-circuited or overloaded	Correct connection
	Fault in 5V encoder signals (wire break, short circuit, cable missing)	Correct connection
	Module parameterization does not match the position of the coding plug	Correct parameterization and transfer, or reconnect the coding plug

Initiating a Diagnostics Interrupt

All errors except an EPROM test error can trigger a diagnostic interrupt provided you have enabled the diagnostic interrupt in the basic parameters during parameter assignment. The diagnostic data records DS0 and DS1 (Page 9159) will show you which fault has caused the LED to light up.

Channel-specific diagnostics

Triggering diagnostics interrupts

Definition of a diagnostics interrupt

You can determine reactions to internal or external errors in the user program by configuring a diagnostic interrupt which interrupts cyclic program execution on the CPU and calls diagnostic interrupt OB.

Events which can trigger a diagnostics interrupt

The following events can trigger a diagnostic interrupt:

- Faults in the external auxiliary 1L+/1M voltage.
- 5.2 V DC encoder supply short circuit or overload
- No module configuration
- Faulty module parameters
- Time monitoring (watchdog) triggered
- RAM defective
- Hardware interrupt lost
- Signal A error (wire break, short-circuit, cable missing)
- Signal B error (wire break, short-circuit, cable missing)
- Signal N error (wire break, short-circuit, cable missing)
- Coding plug incorrectly connected

Enabling diagnostics interrupts

When configuring the hardware, you enable the interrupts for the module in the basic parameters and choose whether the module is to trigger diagnostic and/or hardware interrupts.

Reactions to a diagnostics interrupt

When an event occurs that could trigger a diagnostic interrupt:

- The diagnostics information is stored in the diagnostic data records DS0 and DS1 (Page 9159) in the module.
- The group error LED is lit.
  The group error LED does dark when the error is eliminated.
- The diagnostic interrupt OB (e.g. OB 82) is called.

- The diagnostic data record DS0 is entered in the start information of the diagnostic interrupt OB.
- The count continues unchanged if a diagnostic interrupt OB has been programmed. If no diagnostic interrupt OB has been programmed, the CPU goes to STOP.

Default setting

No diagnostic interrupts are enabled in the default setting.

Diagnostics data record DS0 and DS1

Diagnostic data record DS0 and DS1

The information showing the event which has triggered a diagnostics interrupt is written to the diagnostics data records DS0 and DS1. The diagnostic data record DS0 comprises four bytes. DS1 comprises 16 bytes; the first four bytes are identical to DS0.

Reading data records from the module

Diagnostic data record DS0 (Page 9159) is automatically transferred to the start information when the diagnostic OB is called. These four bytes are written to the local date (bytes 8 to 11) of the diagnostic interrupt OB.

You can read out the diagnostic data record DS1 (Page 9160) (and thus also the content of the DS0) from the module with the instruction DIAG_INF. However, this is only useful if DS0 reports a channel error.

Assignments of diagnostics data record DS0

Assignments of diagnostics data record DS0 in the start information

The following table shows the assignment of the diagnostic data record DS0 in the start information. Any bits not listed are insignificant and are zero.

Byte	Bit	Meaning	Comments	Event no.
0	0	Module in error state	Set upon all diagnostics events	8:x:00
	1	Internal error	Set at all internal error events: • RAM test error • Time monitoring (watchdog) triggered • Lost hardware interrupt	8:x:01
	2	External error	Set at all external error events: • Auxiliary voltage 1L+/1M not connected or short-circuit in the 5.2 V DC encoder supply. • 5.2 V DC encoder supply short circuit or overload • 5 V signal faults • Parameter error	8:x:02

Byte	Bit	Meaning	Comments	Event no.
	3	Channel error	See DS1, byte 4 (Page 9160) for a more detailed breakdown	8:x:03
	4	Failure of the external auxiliary voltage	Check the voltage	8:x.04
	6	No configuration	Assign parameters	8:x:06
	7	Parameter error	See Data errors (Page 9160) for a more detailed breakdown	8:x:07
1	0...3	Type class	Always assigned the value 8	
	4	Channel information	Always assigned the value 1	
2	3	Time monitoring (watchdog) triggered	Module defective, or heavy interference	8:x:33
3	3	RAM defective	Module defective, or heavy interference	8:x:43
	6	Hardware interrupt lost	Check the configuration. A hardware interrupt was detected and can not be reported, because the same event is not yet acknowledged by the CPU	8:x:46

Assignment of diagnostic data record DS1

Assignment of diagnostic data record DS1

Diagnostic data record DS1 consists of 16 bytes. The first 4 bytes are identical to diagnostic data record DS0. The following table shows the assignment of the remaining bytes. Any bits not listed are insignificant and are zero. The instruction DIAG_INF writes this data record to the DB of the instruction CNT_CTL1, starting at DW54.

Byte	Bit	Meaning	Comments	Event no.
4	0...6	Channel type	Always assigned the value 76H	
	7	Further channel types	Always assigned zero value	
5	0...7	Diagnostics data length	Always assigned the value 10H	
6	0...7	Number of channels	Always assigned the value 1	
7	0	Channel error vector	Assigned 1 on channel error	
8	0	Signal A error		8:x:B0
	1	Signal B error		8:x:B1
	2	Signal N error		8:x:B2
	4	Error at the 5.2-V encoder supply		8:x:B4
	5...7	Reserved		
9 ... 15		Reserved		

Data error

Data error events

FM 350-1 checks all new parameters transferred to the module. The module reports any errors returned in this check.

Where data errors are indicated

Instruction CNT_CTL1 enters the data errors and the error numbers in the CNT_CTL1 DB. You can access this data word in the user program using the DA_ERR_W variable.

What data errors are possible?

The table below lists the data errors and their error numbers and meaning.

No.	Meaning
0	No error
200	Coding plug in wrong position or missing
201	The position of the coding plug does not match the programmed encoder
202	Invalid diagnostics value of the signal pair
203	Incorrect value for signal evaluation
204	Incorrect 24 V count signal input filter value
205	Invalid value at digital input filters
206	Reversal of direction not allowed
207	Incorrect configuration of the reaction of DO0
208	Incorrect configuration of the reaction of DO1
209	Pulse duration out of limits
211	Wrong operating mode selected
212	No gate or both gates defined
213	Faulty parameters of main count direction
214	Count high limit violated
215	A different count direction was selected for configuring output behavior "Active on reaching the comparison value for pulse duration up or down" in parameter assignment for the hardware interrupt "Reaching the comparison value up or down". The directions set must match.
216	Gate control interrupts are only possible in modes with gate control.
217	Interrupt on "Reaching comparison values" is not valid for output behavior "On when counter value >= comparison value" or "On when counter value <= comparison value".
218	Interrupt triggering at the zero transition is not allowed
219	Incorrect coding of the "Latch" setting
220	Faulty gate control parameters
221	Low limit or load value violated
222	High limit or comparison value 1 exceeded
223	Update time or comparison value 2 exceeded
224	Pulses per encoder revolution out of limits

How to acknowledge data errors

Correct the parameter values according to specifications. Transfer the corrected parameter set to the FM 350-1 again. It will recheck the parameters and delete the data error in the DB.

Operator error

Operator errors events

Operator errors develop as a result of improper operation of the module caused by the incorrect input of control signals.

Objects which indicate operator errors

The instruction CNT_CTL1 enters the operator error numbers in the DB. Instruction CNT_CTL1 sets output parameter OT_ERR to indicate an operator error has occurred. You can access this data byte in your program using the OT_ERR_B variable.

What operator errors are possible?

The table below lists the operator errors and their error numbers and meaning.

No.	Meaning
0	No error
1	Operating mode can not be started using the SW gate
2	Operating mode can not be canceled
4	Only allowed when the CPU is in STOP
5	Only the parameter assignment control bit may be set
6	Illegal job
10	Low limit or load value violated
11	High limit or comparison value 1 exceeded
12	Update time or comparison value 2 exceeded
20	Incorrect configuration of the reaction of DO0
21	Incorrect configuration of the reaction of DO1
22	Pulse duration out of limits
90	See CNT_CTL2 (Page 7100)
91	See CNT_CTL2 (Page 7100)

How to acknowledge operator errors

You acknowledge the error with the input parameter OT_ERR_A of instruction CNT_CTL1 or CNT_CTL2.

Using FM 350-2

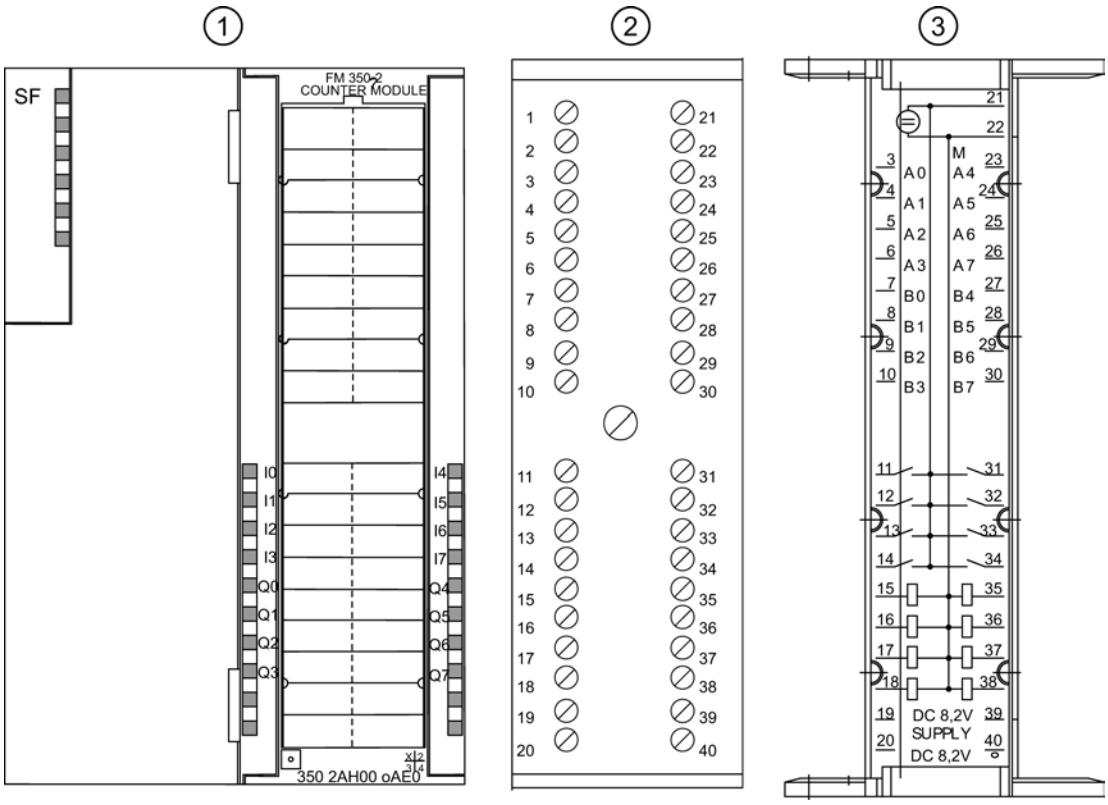
FM 350-2 front connector terminal assignment

Front connector

You connect the following to the 40-pin front connector:

- Count signals
- Digital inputs
- Digital outputs
- Encoder supply and module power supply

The figure below shows the front of the module, the front connector and the inside of the front panel with terminal assignment.



① Front of the module
② Front connectors
③ Inside of the front panel cover

Front Connector Assignments

The following table shows the terminal assignment for the front connector of the FM 350-2.

Terminal	Name	Input/output	Function
1	-	-	Not connected
2	-	-	Not connected
3	A0	Input	Channel 0 count input NAMUR / BERO
4	A1	Input	Channel 1 count input NAMUR / BERO
5	A2	Input	Channel 2 count input NAMUR / BERO
6	A3	Input	Channel 3 count input NAMUR / BERO
7	B0	Input	Channel 0 direction input BERO
8	B1	Input	Channel 1 direction input BERO
9	B2	Input	Channel 2 direction input BERO
10	B3	Input	Channel 3 direction input BERO
11	I0	Input	Channel 0 hardware gate input BERO
12	I1	Input	Channel 1 hardware gate input BERO
13	I2	Input	Channel 2 hardware gate input BERO
14	I3	Input	Channel 3 hardware gate input BERO
15	Q0	Output	Channel 0 digital output 0.5 A
16	Q1	Output	Channel 1 digital output 0.5 A
17	Q2	Output	Channel 2 digital output 0.5 A
18	Q3	Output	Channel 3 digital output 0.5 A
19	P8V2	Output	8.2 V NAMUR encoder supply
20	P8V2	Output	8.2 V NAMUR encoder supply
21	L+	Input	24 V module supply
22	M	Input	Ground module supply
23	A4	Input	Channel 4 count input NAMUR / BERO
24	A5	Input	Channel 5 count input NAMUR / BERO
25	A6	Input	Channel 6 count input NAMUR / BERO
26	A7	Input	Channel 7 count input NAMUR / BERO
27	B4	Input	Channel 4 direction input BERO
28	B5	Input	Channel 5 direction input BERO
29	B6	Input	Channel 6 direction input BERO
30	B7	Input	Channel 7 direction input BERO
31	I4	Input	Channel 4 hardware gate input BERO
32	I5	Input	Channel 5 hardware gate input BERO
33	I6	Input	Channel 6 hardware gate input BERO
34	I7	Input	Channel 7 hardware gate input BERO
35	Q4	Output	Channel 4 digital output 0.5 A
36	Q5	Output	Channel 5 digital output 0.5 A
37	Q6	Output	Channel 6 digital output 0.5 A
38	Q7	Output	Channel 7 digital output 0.5 A

Terminal	Name	Input/output	Function
39	P8V2	Output	8.2 V NAMUR encoder supply
40	P8V2	Output	8.2 V NAMUR encoder supply

Note

The circuits for the counter inputs (encoder supply, encoder signals) are isolated from the ground of the CPU.

All inputs are not isolated from each other but are isolated from the S7 300 bus.

You will find more information on structure and wiring in the Commissioning and Hardware Installation Manual (http://support.automation.siemens.com/WW/view/en/1105178) for the FM 350-2 counter module.

24 V voltage supply

Connect a direct voltage of 24 V to the L+ and M terminals for the FM 350-2 voltage supply.

8.2 V DC encoder power supply

The module generates a voltage of 8.2 V (max. 200 mA) from the 24 V supply voltage. This voltage is available at the terminals P8V2 (pins 19, 20, 39, and 40) for the voltage supply to the NAMUR encoders and is resistant to short circuits.

The encoder supply is monitored for 8.2 V.

Encoder Signals A0 to A7, B0 to B7

You can connect four different types of encoder:

- NAMUR encoders pursuant to DIN 19234 (with diagnostics):
  The signals are wired to terminals A0 to A7.
- 24 V incremental encoder:
  The signals A0/B0 to A7/B7 are connected via the terminals so labeled.
- 24 V pulse encoder with direction level:
  The count signals are wired to terminals A0 to A7. The directional signals are wired to terminals B0 to B7.
- 24 V initiators:
  The signals are connected to terminals A0 to A7.

Note

An external 24 V DC voltage supply line must be used for the 24 V encoder supply.

Digital Inputs I0 to I7 (Hardware Gates)

You can use the digital inputs I0 to I7 for the gate control of the counter.

One digital input is available for each count channel with which you can start and stop the corresponding counter.

The digital inputs are operated at a rated voltage of 24 V.

Digital Outputs Q0 to Q7

The FM 350-2 can trigger control processes directly through digital outputs Q0 to Q7.

One digital output is available per counter.

The digital outputs are powered by the FM 350-2 24 V voltage supply.

The digital outputs are current-sourcing switches and support a load current of 0.5 A. These outputs are protected against overload and short circuit-proof.

Note

Relays and contactors can be connected directly without external circuitry.

Configuring FM 350-2

Configuring FM 350-2

Specifying channels

Selecting channel settings

Channel setting

This is the field in which you set the function of the module channels:

- Single counter: Each channel in the selected group operates as a separate counter.
- Dosing counter: 4 counter channels are combined to form one dosing channel. This channel can control up to 4 dosing units.

You can make the following selections:

- Channels 0 to 7 as single counters
- Channel 0 as dosingcounter, channels 4..7 as single counters
- Channels 0..3 as single counters, channel 4 as dosingcounter
- Channel 0 as dosing counter and channel 4 as dosing counter

Enabling counter input for NAMUR encoders

NAMUR encoder

This is the field in which you define the counter inputs to which a NAMUR encoder is to be connected.

You can make the following selections:

- Counter input channel 0 to 3 NAMUR
  and/or
- Counter input channel 4 to 7 NAMUR

See also: "NAMUR signals (Page 9167)" and "Enabling hardware monitoring for NAMUR encoders (Page 9174)"

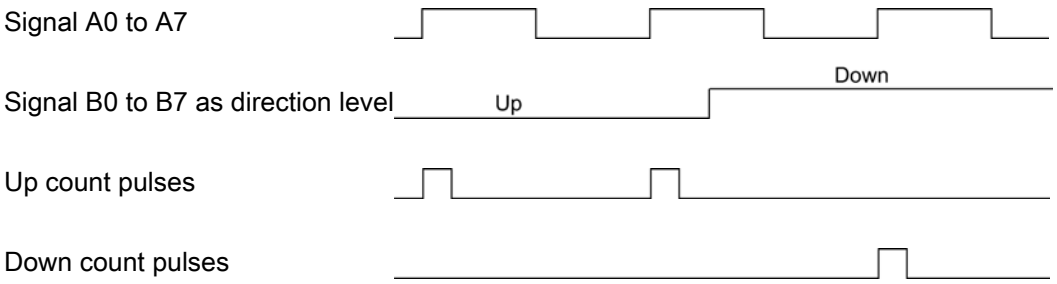
NAMUR Signals

NAMUR encoder

The encoder supplies one count signal in accordance with DIN 19234 and this must be connected to terminal A0 to A7 of the front connector.

You can wire a directional signal (24 V) to terminal B0 to B7 of the relevant counter. If your encoder does not return a corresponding signal, you can also generate and interconnect a relevant identifier within the S7 system, or use a suitable process signal.

The diagram below shows the time profile of a NAMUR encoder with directional level.



You can enable inputs A0 to A3, A4 to 7, or all inputs for NAMUR encoder connections.

How are the Signals Monitored?

You can assign parameters for the FM 350-2 so the module monitors A0 to A7 for wire break and short circuit. This also indirectly results in the 8.2 V encoder supply being monitored. This is only possible with NAMUR encoders.

If a wire break or short circuit is detected, the group error LED lights up, you can also assign a diagnostic interrupt to be triggered.

CAUTION
Danger of property damage. Using another encoder on a FM 350-2 channel which was parameterized for the connection of a NAMUR encoder can damage the module. You should only connect NAMUR encoders to FM 350-2 channels parameterized for the connection of a NAMUR encoder.

Selecting feedback interface assignment

Feedback interface assignment, bytes 8 to 15

This is the field in which you select how bytes 8 to 15 of the feedback interface are assigned.

The following options are available for variables USER_STAT_WORD_0 to USER_STAT_WORD_3:

- Data type
  - WORD
  - DWORD
- Channel
  - Channels 0 to 7
- Value
  - Counter value
  - Measured value
- Word
  - Either the low word or the high word of a value or both can be stored. You have direct access to these values.

Mapping the count and measurement values of all channels in the expanded user data

Mapping of all count and measurement values in the user data image

Under certain conditions, all count and measurement values can be mapped in the user data image (use of expanded user data).

The following basic conditions apply to use of expanded user data:

- Configure the module "6ES7350-2AH01-0AE0 E" from the module catalog.
- The check box "Use extended user data interface" must be selected.

- The extended user data can only be used on a distributed basis behind an IM 153-2 V5.0 (order number 6ES7153-2BA01-0XB0 V5.0) or higher.
- 40 bytes of input data are now available.
- The update cycle of the extended user data differs from the update cycle of the first 16 bytes of input data. The update cycle of the extended user data is approximately 100 to 200 ms depending on the PROFIBUS DP cycle time.

The structure of the first 16 bytes in the user data is the same as before in each operating mode. The additional 24 bytes are defined by the operating mode.

Depending on the operating mode, the values (count or measurement value) of channels 2 to 7 are stored in the expanded user data. These values are preassigned to "0". The input data are occupied with 0 as long as valid values have not been detected.

Structure of the extended user data as defined by the operating mode:

Byte	Counting modes	Measuring modes	Dosing mode (two dosing counters)
0..15	Same as previous – structure can be influenced via parameter assignment	Same as previous – structure can be influenced via parameter assignment	Same as previous – structure can be influenced via parameter assignment
16..19	Count value channel 2	Measurement value channel 2	0
20..23	Count value channel 3	Measurement value channel 3	0
24..27	Count value channel 4	Measurement value channel 4	Dosing counter channel 4
28..31	Count value channel 5	Measurement value channel 5	0
32..35	Count value channel 6	Measurement value channel 6	0
36..39	Count value channel 7	Measurement value channel 7	0

Copying channels

Copying channels

You can copy the parameters of one channel to one or more other channels.

1. Select the channel whose parameters are to be copied (source channel).
2. Select the channels to which the parameters are to be copied (target channel).
3. Press the "Copy" button.

Channel n operating modes

Selecting an operating mode

Description

You define counter module functionality by specifying an operating mode.

Mode	Description
Count continuously (Page 9083) (with or without gate)	The counter module counts from the current counter reading continuously.
Count once (Page 9084) (with hardware or software gate)	When the gate opens, the counter module counts from zero up to the parameterized high count limit or from the parameterized high count limit to zero.
Count periodically (Page 9085) (with hardware or software gate)	The counter module counts between zero and the parameterized high count value when the gate is opened.
Frequency measurement (Page 9093) (with hardware or software gate)	The counter module detects the incoming pulses in a parameterized time window of n x 10 ms and uses this data to determine the current frequency.
Rotational speed measurement (Page 9094) (with hardware or software gate)	The counter module detects the incoming pulses in a parameterized time window of n x 10 ms and determines the speed on the basis of the parameterized number of pulses per encoder revolution.
Period measurement (Page 9096) (with hardware or software gate)	The counter module detects the incoming pulses in a parameterized time window of n x 10 ms and uses this data to determine the period.
Dosing (Page 9098) (with hardware or software gate)	In this mode, four count channels are grouped together to form a dosingchannel. This mode functions in exactly the same way as the "Count once" mode.

The default setting is the "Count continuously" operating mode.

Specifying the high count limit

Description

By configuring an high count limit together with a main count direction (up or down), you can reduce the maximum counting range. The parameterized counting range is then between 0 and the high count limit parameterized.

The high count limit can be any value between > 1 and the high count range limit.

Specifying measuring time

Description

If you have selected a measuring mode, the module will update the measured values on a cyclic basis. The integration time is the product of the measuring time x 10ms. You set the measuring time as the factor n.

Detailed descriptions of the value ranges for the measuring time can be found under "Measuring basics (Page 9088)".

Using a hardware gate

Using a hardware gate

You can decide when or not the channel is to count with a hardware gate.

If counting is to be done with a hardware gate, check the check box "Use hardware gate".

Selecting gate function

Description

When configuring the hardware and software gates, you can specify whether the internal gate is to cancel or interrupt counting.

- Cancel counting (Page 9103): The counting process restarts from the beginning after gate stop and restart.
- Interrupt counting (Page 9103): The counting process starts from the last counter value after gate stop and restart.

Selecting main count direction

Description

By configuring a main count direction (Page 9086) (up or down), you can limit the maximum counting range by setting an high count limit. The parameterized counting range is then between 0 and the high count limit parameterized.

A main count direction can only be selected in the "Count once", " Count periodically" and "Dosing" modes.

You can select the following behavior:

- Up
- Down

Channel n encoder

Selecting the encoder signal type

Selecting the signal type

1. Select the type of signal for the encoder used.
   In line with your selection and the operating mode, any option not possible for this specific encoder will be disabled. A disabled option is identified by grayed font in the corresponding selection window.
2. Select the options to be used with the selected encoder.

You can select the signal type for one of the following encoders:

- Encoder: 24 V differential signal (Page 9172) (24 V track A+B phase shifted)
- Encoder: 24 V pulse and direction (Page 9172)

Encoder: 24 V incremental

24 V incremental encoder

Select signal type "24 V track A+B phase shifted" to use a 24 V incremental encoder with signals A* and B*.

The following options are available for this encoder:

Parameter	Option	Meaning
Signal evaluation (Page 9108)	Single	Only one edge of signal A is evaluated.
	Double	Both edges of signal A are evaluated.
	Quadruple	Both edges of signal A, and B are evaluated.
Count direction (Page 9173)	Normal	To adapt the count direction to the process
	Inverted	To adapt the count direction to the process

Encoder: 24 V pulse and direction

24 V pulse generator

Select signal type "24 V pulse and direction" to use a 24 V pulse generator with count signal A* and direction signal B*.

The following options are available for this encoder:

Parameter	Option	Meaning
Signal evaluation	–	Not relevant
Count direction (Page 9173)	Inverted	To adapt the count direction to the process
	Normal	To adapt the count direction to the process

Selecting the count direction

Changing the count direction

You can change the count direction using the "Count direction normal" and "Count direction inverted" parameters without having to modify the wiring.

Assigning the parameter "Main count direction down (Page 9086)" does **not** automatically reverse the count direction.

Entering pulses per encoder revolution

Pulses per encoder revolution

For "Rotational speed measurement (Page 9094)" mode, you must also configure the pulses per encoder revolution.

The value for the number of pulses per encoder revolution can be found on the rating plate, or in the technical data of your encoder.

Setting hysteresis

Function principle of the hysteresis

An encoder can stop at a certain position and then "oscillate" around this position. This state causes the counter level to fluctuate around a specific value. If there is a comparison value within this fluctuation range, for example, the associated output is switched on and off in accordance with the rhythm of the fluctuations. The FM 350-2 has configurable hysteresis to prevent this switching in response to small fluctuations.

Values between 0 and 255 can be set for hysteresis.

The effect of hysteresis is as follows in accordance with the value parameterized:

Hysteresis	Effect
Hysteresis value n = 0, 1	The hysteresis has no effect (switched off) The output reacts to the slightest change in the counter value.
2 ≤ Hysteresis value n ≤ 255	The hysteresis takes effect. The output does not react until the counter value is offset by n units to the comparison value.

The hysteresis applies to both high limit violated and low limit violated.

Enabling hardware monitoring for NAMUR encoders

How are the signals from NAMUR encoders monitored?

FM 350-2 can monitor A0 to A7 for wire break and short circuit. This also indirectly results in the 8.2 V encoder supply being monitored.

Select the check box "Hardware monitoring for NAMUR encoders" if FM 350-2 is to monitor the encoder signals.

Channel n outputs

Comparison values and outputs

Description

You can store eight comparison values on the FM 350-2, each for one of the eight digital outputs (comparison value 0: Q0, comparison value 1: Q1, etc.). The relevant output can be set depending on the counter level and the comparison value.

You can assign four comparison values for each dosing channel of the FM 350-2 in "Dosing (Page 9098)" mode.

Comparison values

The comparison values are entered during parameter assignment.

When the CPU is in RUN mode you can enter the comparison values in the data block of the CNT2_WR / CNT2WRPN instruction (CMP_VAL0…7) and transfer them to the FM 350-2 using jobs 30 to 37 or 60 to 62. The count is not affected by this.

The comparison values must lie within the count range of the respective operating mode; the count range limits cannot be used.

If you want to set the comparison value to the initial or final value in an operating mode with a main count direction, the digital outputs will behave as follows:

Parameterized behavior of digital outputs	Response of assigned digital outputs
On when counter value >= comparison value	The output is always switched on irrespective of the counter value
On when counter value <= comparison value	The output is always switched off irrespective of the counter value

Enabling the Outputs

Before the outputs can be set, you must first enable them by setting the relevant bits in the counter DB. If you reset one of these bits, the associated output will be switched off

immediately. The bits are transferred between the data block and the module by the CNT2_CTR instruction.

Output	...is enabled by
Q0…7	CTRL_DQ0…7

Output switch-off

The outputs are switched off by the following events regardless of the parameter assignment:

- Module watchdog triggered (internal error)
- Removal of the enable bits (CTRL_DQ0...7 for Q0...7 in the DB)

Controlling the outputs

Irrespective of output switching via the comparators, you can set and reset the outputs when they are enabled by CTRL_DQ0…7 using the SET_DQ0…7 control signals.

The following applies to the relationship between controlling and switching: Controlling has priority over switching using the comparators. This means:

- The output can be used as a digital output if switching using the comparator is not set for the output.
- You can continue to control the outputs using SET_DQ0…7 if you have selected a compare function for the outputs. You can use this setup in the user program to simulate the effect of the compare functions:
  - The output is set with the positive edge of SET_DQ0…7.
  - A SET_DQ0…7 negative edge resets the output.

  Note that the comparators are still active and can set or reset the output if the comparison result changes.

  Note

  An output set with SET_DQ0…7 cannot be reset with the comparator.

Behavior of the outputs in counting modes

Specifying the behavior of the outputs

Behavior of the outputs DQ0 to DQ7

Select the required behavior for the output in question. Activate the corresponding option button:

- No comparison (Page 9176)
- On when counter value >= comparison value (Page 9176)
- On when counter value <= comparison value (Page 9177)

Note

To activate the settings, enter values for comparison values in the DB with the counter data and transfer them to the module with the instruction CNT2_WR / CNT2WRPN; you must also set the CTRL_DQ0…7 bits in the DB in order to enable the outputs. See also "Comparison values and outputs (Page 9174)".

Note

You cannot configure the outputs in "Frequency Measurement", "Rotational Speed Measurement" or "Period Duration Measurement" mode.

Output default setting

The default setting for the outputs is deactivated.

No comparison

No comparison

The output remains deactivated and is not influenced by the events comparison value, zero pass, jump from final value to initial value (high limit violated) or jump from initial value to final value (low limit violated).

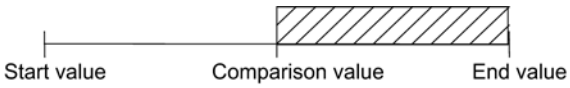
On when counter value >= comparison value

On when counter value >= comparison value

The output is set to 1 if the counter is in the range between the comparison value and the final value.

Setting the counter to a value between the comparison value and final value sets the output to 1.

The hatched area in the graphic below means that the output is set in this range.



Constraint:

The following constraint applies if you want to configure an output "On when counter value >= comparison value": The time between reaching the comparison value and the initial or final value must be longer than the outputs' minimum switching time (switching time: 300 µs). The control pulses at the outputs will otherwise be lost.

See also

Effect of the hysteresis (Page 9177)

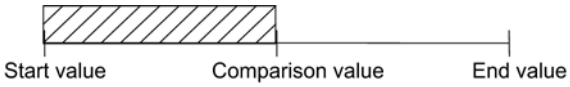
On when counter value <= comparison value

On when counter value <= comparison value

The output is set to 1 if the counter is in the range between the comparison value and the initial value.

Setting the counter to a value between the comparison value and initial value sets the output to 1.

The hatched area in the graphic below means that the output is set in this range.



Constraint:

The following constraint applies if you want to configure an output "On when counter value <= comparison value": The time between reaching the comparison value and the initial or final value must be longer than the outputs' minimum switching time (switching time: 300 µs). The control pulses at the outputs will otherwise be lost.

See also

Effect of the hysteresis (Page 9177)

Effect of the hysteresis

Function of hysteresis with parameter assignment "On when counter value >= comparison value or <= comparison value"

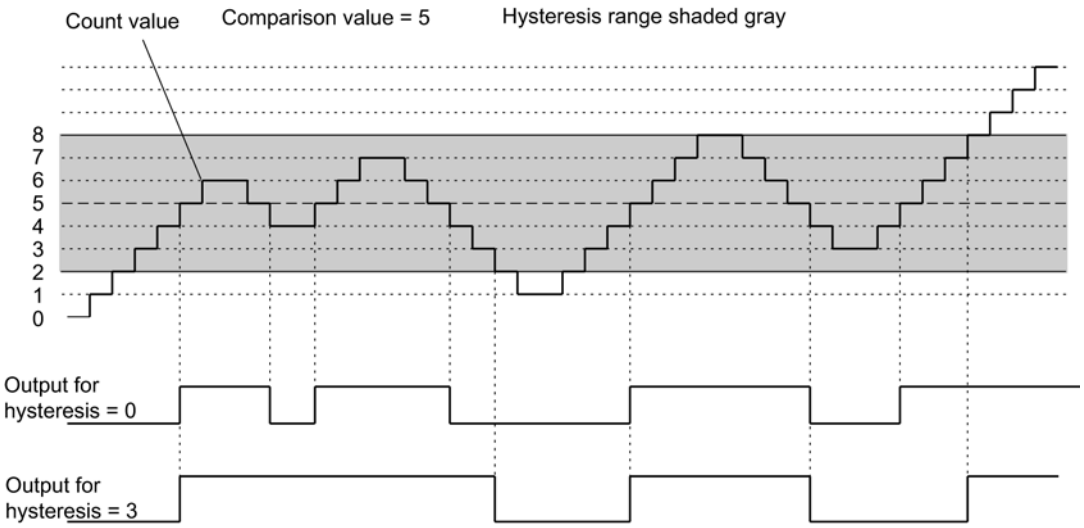
The example in the figure below demonstrates the effect of hysteresis. The diagram shows the differences between the behavior of an output at a set hysteresis of 0 (= switched off) and a hysteresis of 3. In the example, the comparison value = 5

Parameter assignment in this example is as follows:

- Main count direction: "Up"
- Output response: "On when counter value >= comparison value"

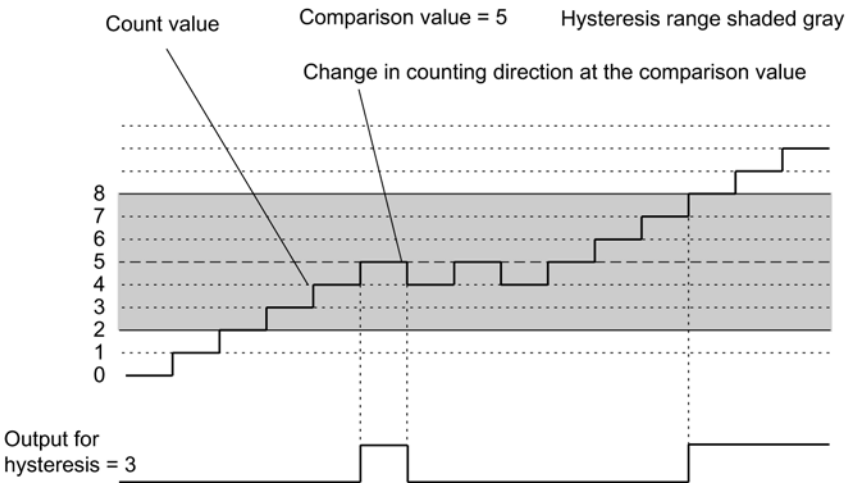
The hysteresis is set when the comparison condition (counter value = 5) is fulfilled. When the hysteresis is active, the comparison result remains unchanged.

If the count value overshoots/undershoots the range of the hysteresis (at counter value 2 or 8 in the example) the hysteresis ceases to be active. The comparator switches again according to its comparison conditions, i.e., in the example at counter value 5.



Note

If the counter value is equal to the comparison value, and hysteresis is active, FM 350-2 resets the output when the count direction changes at the comparison value (see the diagram below).



Enabling substitute value '1' at CPU STOP

Substitute value "1" at CPU STOP

If you selected the "Substitute value" option under "Reaction to CPU STOP" in the module's basic parameters and the CPU goes into STOP, the module will return the parameterized substitute values at the digital outputs. Check the checkbox for DQ0, for example, and the output Q0 will be set at CPU STOP.

You can set up to four substitute values in "Dosing" mode and one in each of the other counting modes.

After the CPU transition from STOP to RUN, the substitute values are retained at the digital outputs until the next value is input.

The outputs are reset if you change the "Reaction to CPU STOP" basic parameter.

⚠ CAUTION
Danger of damage to equipment. Parameterized substitute values are always output when the CPU goes into STOP mode even if the gate is closed and even at outputs that are not enabled. Only configure substitute values which will not lead to a dangerous system status if output.

Channel n enable hardware interrupts

Enabling hardware interrupt trigger

Introduction

The FM 350-2 allows you to set which events are to trigger a hardware interrupt during operation.

What is a hardware interrupt?

If you want to set a response to a specific event independently of the CPU cycle, the FM 350-2 can initiate a hardware interrupt. The CPU then interrupts the cyclical program and processes the hardware interrupt OB.

Enabling the hardware interrupt

Hardware interrupts are enabled in two steps:

1. When configuring the hardware, you enable the interrupts for the module in the basic parameters and choose whether the module is to trigger diagnostic and/or hardware interrupts.
2. You then select the relevant events under "Enable hardware interrupts":
   - Events to trigger a hardware interrupt in counting modes (Page 9181)
   - Events to trigger a hardware interrupt in measuring modes (Page 9181)

Hardware Interrupt OB, e.g. OB40

If a hardware interrupt occurs, the user program is interrupted, the data is transferred from the module to the start information of OB40 (Page 9181) and OB40 is called. The hardware interrupt is acknowledged by exiting OB 40.

If no hardware interrupt has been programmed, the CPU goes into STOP mode. If you then switch back to RUN, the hardware interrupt requirements will be deleted.

Lost hardware interrupt

If an event occurs which is supposed to trigger a hardware interrupt but there is an identical, previous event which has not yet been acknowledged, no further hardware interrupt will be triggered; the hardware interrupt will be lost. This may lead to the "Hardware interrupt lost" diagnostic interrupt depending on parameter assignment.

If there is less that 2 ms between two events which should trigger a hardware interrupt, the second hardware interrupt will be lost without a diagnostic interrupt being triggered.

Default setting

No hardware interrupts are enabled in the default setting.

Events to trigger a hardware interrupt in counting modes

Which events can trigger a hardware interrupt in counting modes?

The following events can trigger a hardware interrupt in counting modes during operation of the FM 350-2:

- The hardware gate opening
- The hardware gate closing
- High and low limit violation
- Reaching a comparison value
  Enter the comparison value in the text box. The comparison values must lie within the count range of the respective operating mode; the count range limits cannot be used.

Select the check box for the relevant events.

You can select a number of different events to trigger a hardware interrupt depending on the operating mode set. You must observe the relevant basic conditions for hardware interrupts on reaching a comparison value (for "On when counter value >= comparison value (Page 9176)" and "On when counter value <= comparison value (Page 9177)").

Events to trigger a hardware interrupt in measuring modes

Which events can trigger a hardware interrupt in measuring modes?

The following events can trigger a hardware interrupt in measuring modes during operation of the FM 350-2:

- The hardware gate opening
- The hardware gate closing
- End of measurement
- Violating low limit
  Enter the low limit in the text box. Information on the value ranges for the low limits can be found under "Measuring basics (Page 9088)".
- Exceeding high limit
  Enter the high limit in the text box. Information on the value ranges for the high limits can be found under "Measuring basics (Page 9088)".

Select the check box for the relevant events.

You can select a number of different events to trigger a hardware interrupt depending on the operating mode set.

OB 40 start information

OB 40 start information

The temporary variable OB40_POINT_ADDR is described in the start information of OB 40.

The OB40_POINT_ADDR variable (bytes 8 to 11) comprises 4 bytes. The information on the event that triggered the hardware interrupt is entered in these bytes.

The table below shows which bits are set for which interrupt. Any bits not listed are irrelevant and are zero.

Operating mode: Count continuously, count once, count periodically

Channel	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	8					Comparison value reached	High and low limit violation	Hardware gate closed	Hardware gate opened
1		Comparison value reached	High and low limit violation	Hardware gate closed	Hardware gate opened				
2 to 7	9 to 11	See byte 8							

Operating mode: Frequency measurement, rotational speed measurement, period measurement

Channel	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	8					High limit violated / low limit violated	End of measurement	Hardware gate closed	Hardware gate opened
1		High limit violated / low limit violated	End of measurement	Hardware gate closed	Hardware gate opened				
2 to 7	9 to 11	See byte 8							

Operating mode: Dosing

Channel	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	8	–	High and low limit violation	Comparison value 4 reached	Comparison value 3 reached	Comparison value 2 reached	Comparison value 1 reached	Hardware gate closed	Hardware gate opened
4	10	See byte 8							

Commissioning the FM 350-2

Using FM 350-2 from the user program

Integrating the FM 350-2 into a user program

Instructions are available for integrating the FM 350-2 into a user program. These enable you to handle the desired functions with ease.

The instructions CNT2WRPN, CNT2RDPN and CNT2_CTR can be called in the CPU user program. They ensure communication between the CPU and the FM 350-2.

All data associated with a channel of the module are in the DB of the instruction CNT2_CTR, CNT2WRPN or CNT2RDPN (counter DB (Page 7113)). Use the command "Program blocks > Add new block" to add a CNT2_CHANTYPE type data block to your project.

The DIAG_RD instruction is also available for FM 350-2. This operation allows you to transfer the diagnostic information to the counter DB.

See also

CNT2WRPN (Page 7110)

CNT2RDPN (Page 7111)

CNT2_CTR (Page 7112)

DIAG_RD (Page 7112)

FM 350-2: Online & Diagnostics

Current module information

Default diagnostics

FM 350-2 group error LED fault display

Where the error is indicated

If the red group error LED lights up, either a fault has occurred in the module (internal fault) or cable connections (external fault), or there is a parameter assignment error.

Which Errors are Indicated?

The table below shows the error types displayed by the group error LED.

Type of Error	Cause of error	Remedy
Internal errors	Watchdog tripped	Module replacement
	Lost hardware interrupt	acknowledged by hardware interrupt processing
External errors	No module parameters	Assign parameters and transfer
	Parameter assignment error	
External channel errors	Encoder supply short-circuited or overloaded	Correct connection
	Fault in NAMUR encoder signals (wire break, short circuit, cable missing)	Correct connection

Channel-specific diagnostics

Triggering diagnostics interrupts

What is a diagnostic interrupt?

You can determine reactions to internal or external errors in the user program by configuring a diagnostic interrupt which interrupts the CPU cyclic program and calls the diagnostic interrupt OB 82.

What events can trigger a diagnostic interrupt?

The following events can trigger a diagnostic interrupt:

- Channel error in parameterized channel
- No module parameter assignment
- Error in module parameter assignment
- Watchdog reacts
- Hardware interrupt lost
- Signal line monitoring NAMUR encoder reports error
- Fault in 8.2 V NAMUR encoder supply

Enabling a diagnostic interrupt

When configuring the hardware, you enable the interrupts for the module in the basic parameters and choose whether the module is to trigger diagnostic and/or hardware interrupts.

Responses to a diagnostic interrupt

When an event occurs that could trigger a diagnostic interrupt:

- The diagnostics information is stored in the diagnostic data records DS0 and DS1 (Page 9185) in the module.
- The group error LED lights up.
  The group error LED goes out when the error is eliminated.
- The diagnostic interrupt OB (OB 82) is called.
- The diagnostic data record DS0 is entered in the start information of the diagnostic interrupt OB.
- The count continues unchanged if an OB 82 has been programmed. If no OB 82 has been programmed, the CPU will go into STOP mode.

Default setting

No diagnostic interrupts are enabled in the default setting.

Diagnostic data record DS0 and DS1

Diagnostic data record DS0 and DS1

The information about which event triggered a diagnostic interrupt is stored in the diagnostic data records DS0 and DS1. The diagnostic data record DS0 comprises four bytes. DS1 comprises 16 bytes; the first four bytes are identical to DS0.

Reading a data record from the module

Diagnostic data record DS0 (Page 9185) is automatically transferred to the start information when the diagnostic OB is called. These four bytes are written to the local date (bytes 8 to 11) of OB 82.

You can read out the diagnostic data record DS1 (Page 9186) (and thus also the content of the DS0) from the module with the instruction DIAG_RD. This only makes sense if an error is reported in a channel in the DS0.

Assignment of diagnostic data record DS0

Assignment of the diagnostic data record DS0 and the start information

The following table shows the assignment of the diagnostic data record DS0 in the start information. Any bits not listed are irrelevant and are zero.

Byte	Bit	Meaning	Note	Event no.
0	0	Module fault	Is set for each diagnostic event	8:x:00
	1	Internal error	Is set for every internal error	8:x:01
	2	External error	Set for all external errors	8:x:02
	3	Error in one channel	See DS1, byte 4 (Page 9186) for further details	8:x:03
	6	No parameter assignment	Assign parameters	8:x:06
	7	Parameter assignment faults	See Data errors (Page 9187) for a more detailed breakdown	8:x:07
1	0 ... 3	Type class	Is always assigned the value 8	
	4	Channel information	Is always assigned the value 1	
2	3	Watchdog reacts	Module defective, or heavy interference	8:x:33
3	6	Hardware interrupt lost	Check the configuration. A hardware interrupt has been detected and cannot be reported because an identical event has not yet been acknowledged by the CPU.	8:x:46

Assignment of diagnostic data record DS1

Assignment of diagnostic data record DS1

Diagnostic data record DS1 consists of 16 bytes. The first 4 bytes are identical to diagnostic data record DS0. The following table shows the assignment of the remaining bytes. Any bits not listed are irrelevant and are zero. DIAG_RD writes this data record to the DB of the instruction CNT2_CTR, starting at DW212.

Byte	Bit	Meaning	Note	Event no.
4	0 ... 6	Channel type	Always assigned the value 76H	
	7	Other channel types	Is always assigned the value 0	
5	0 ... 7	Diagnostic information length	Is always assigned the value 8	
6	0 ... 7	Number of channels	Is always assigned the value 8	
7	0	Channel fault vector	Channel bit	
8	4	Fault in the 8.2 V encoder supply	NAMUR encoder	8:x:94
(channel 0)	6	NAMUR encoder signal line	Short circuit / wire break	8:x:96
9	4	Fault in the 8.2 V encoder supply	NAMUR encoder	8:x:94
(channel 1)	6	NAMUR encoder signal line	Short circuit / wire break	8:x:96
10	4	Fault in the 8.2 V encoder supply	NAMUR encoder	8:x:94
(channel 2)	6	NAMUR encoder signal line	Short circuit / wire break	8:x:96
11	4	Fault in the 8.2 V encoder supply	NAMUR encoder	8:x:94
(channel 3)	6	NAMUR encoder signal line	Short circuit / wire break	8:x:96
12	4	Fault in the 8.2 V encoder supply	NAMUR encoder	8:x:94
(channel 4)	6	NAMUR encoder signal line	Short circuit / wire break	8:x:96
13	4	Fault in the 8.2 V encoder supply	NAMUR encoder	8:x:94
(channel 5)	6	NAMUR encoder signal line	Short circuit / wire break	8:x:96
14	4	Fault in the 8.2 V encoder supply	NAMUR encoder	8:x:94
(channel 6)	6	NAMUR encoder signal line	Short circuit / wire break	8:x:96
15	4	Fault in the 8.2 V encoder supply	NAMUR encoder	8:x:94
(channel 7)	6	NAMUR encoder signal line	Short circuit / wire break	8:x:96

Data error

Where do data errors occur?

Checks are performed when jobs are given to the module by the programming device or by means of the CNT2WRPN (Page 7110) instruction. The module reports any errors returned in this check.

Incorrect jobs are not accepted by the module.

Where are data errors displayed?

Data errors are displayed in the parameter assignment interface.

The CHECKBACK_SIGNALS, DATA_ERR = 1 bit is set in the counter DB if an error is located when jobs are checked.

How are data errors acknowledged?

Correct the jobs in accordance with the specifications. Transfer the corrected jobs to the FM 350-2 again.

FM 350-2 diagnostics buffer

Entries in the FM 350-2 diagnostics buffer are displayed in the parameter assignment interface.

Error codes in the diagnostics buffer

Error codes

The error numbers and corresponding error descriptions are listed below. An error with an error ID is entered in the diagnostics buffer in the following form:

16#Fx83 yyzz

Where:

- yy = ID for the error class, for example 04 for a channel parameter assignment error.
- zz = Error number, for example 18 for the error "invalid main count direction"

Channel parameter assignment error	Error number	Description
04	01	Job not allowed, because the channel is being used for proportioning
04	18	Illegal main count direction
04	19	Bad count range
04	1A	Comparison value is not in the achievable count range
04	1B	Illegal output response
04	1C	Hardware interrupt enable not possible

Channel parameter assignment error	Error number	Description
04	1D	Invalid parameter assignment for hardware monitoring
04	1E	Invalid parameter assignment for hardware gate
04	1F	Signal selection not possible
04	20	Bad signal selection
04	21	Value for hysteresis too high
04	22	Substitute value illegal
04	23	Reaction on closing gate illegal
04	24	Reserved parameter set to value other than 0
04	25	Reserved parameter set to value other than 0
04	26	Reserved parameter set to value other than 0
04	27	Bad encoder selection
04	28	Counter input diagnostics only possible with NAMUR encoder
04	29	Value to load not in permitted count range
04	2B	The counting range cannot be extended
04	2C	Comparison value 1 is not in the count range
04	2D	Comparison value 2 is not in the count range
04	2E	Comparison value 3 is not in the count range
04	2F	Comparison value 4 is not in the count range
04	30	Invalid low limit for frequency or rotational speed
04	31	Invalid high limit for frequency or rotational speed
04	32	Low limit >= high limit
04	33	Illegal integration time
04	34	Pulses per encoder rotation illegal
04	35	Invalid low limit for period
04	36	Invalid high limit for period

Module parameters assignment error	Error number	Description
05	0A	Illegal interrupt generation
05	0B	Illegal interrupt selection
05	0C	Illegal reaction with active output disable
05	0D	Reserved parameter set to value other than 0
05	18	Time stamp illegal
05	19	Illegal encoder signal selection
05	1A	Illegal channel mode
05	1B	Invalid distribution of CHECKBACK_SIGNALS.USER_STAT in the data block

Data error	Error number	Description
06	18	Illegal main count direction
06	19	Bad count range

Data error	Error number	Description
06	1A	Comparison value is not in the achievable count range
06	1B	Illegal output response
06	1C	Hardware interrupts cannot be enabled
06	1D	Invalid parameter assignment for hardware monitoring
06	1E	Invalid parameter assignment for hardware gate
06	1F	Signal selection not possible
06	20	Bad signal selection
06	21	Value for hysteresis too high
06	22	Substitute value illegal
06	23	Reaction on closing gate illegal
06	24	Reserved parameter set to value other than 0
06	25	Reserved parameter set to value other than 0
06	26	Reserved parameter set to value other than 0
06	27	Bad encoder selection
06	28	Counter input diagnostics only possible with NAMUR encoder
06	29	Value to load not in permitted count range
06	2B	The counting range cannot be extended
06	2C	Comparison value 1 is not in the count range
06	2D	Comparison value 2 is not in the count range
06	2E	Comparison value 3 is not in the count range
06	2F	Comparison value 4 is not in the count range
06	30	Invalid low limit for frequency or rotational speed
06	31	Invalid high limit for frequency or rotational speed
06	32	Low limit >= high limit
06	33	Illegal integration time
06	34	Pulses per encoder rotation illegal
06	35	Invalid low limit for period
06	36	Invalid high limit for period

Message	Error number	Description
0F	01	End of module start-up
0F	02	No parameters on the module
0F	03	Module has the default parameter settings
0F	04	Module parameters set
0F	05	PG mode is set
0F	06	PG mode is deselected
0F	07	PG mode has failed
0F	08	PG mode is set, no job from the CPU permitted
0F	09	Bad job sequence

Diagnostic error	Error number	Description
80	04	Missing external auxiliary supply
80	05	Front connector missing
80	06	Module parameters not set
80	07	Bad parameters on module
80	30	Wrong/missing user module
80	31	Communication error
80	32	STOP mode
80	33	Time monitoring responded
80	34	Internal module power supply failed
80	35	Battery 1 flat
80	36	Total backup failed
80	41	Processor failure
80	42	EPROM fault
80	43	RAM error
80	44	ADC/DAC fault
80	45	Fuse fault
80	46	Hardware interrupt was lost
80	90	Bad encoder signal A on channel
80	91	Bad encoder signal B on channel
80	92	Bad encoder signal N on channel
80	93	Bad value between channels
80	94	Fault in 5.2 V or NAMUR encoder supply in channel
80	95	Fault in the 24 V encoder supply
80	96	NAMUR encoder short circuit/wire break at channel

Using FM 450-1

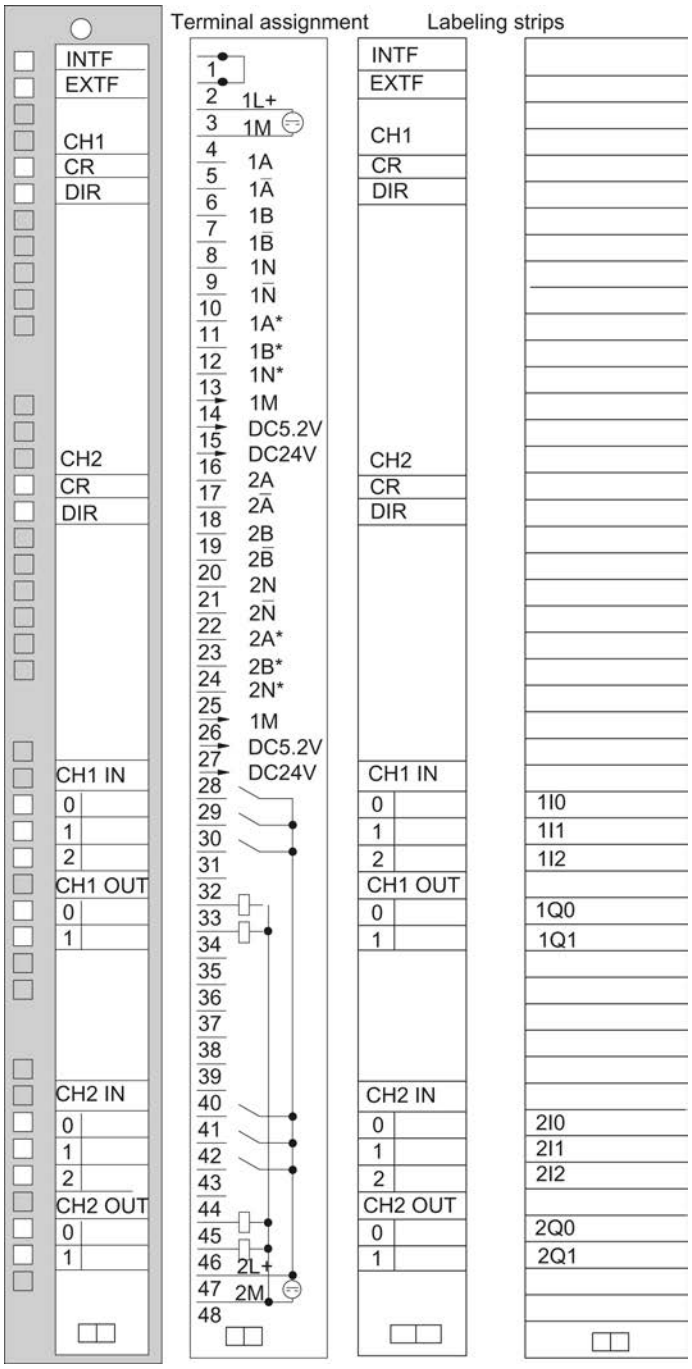
FM 450-1 front connector terminal assignment

Front connector

You connect the following to the 48-pin front connector:

- Count signals
- Digital inputs
- Digital outputs
- Encoder power supply
- Auxiliary voltage and load voltage

The following illustration shows the front of the front connector, the strip printed with terminal assignment and the labeling strips.


Terminal assignment
Labeling strips
INTF
EXTF
CH1
CR
DIR
CH2
CR
DIR
CH1 IN
0
1
2
CH1 OUT
0
1
CH2 IN
0
1
2
CH2 OUT
0
1
1
2 1L+
3 1M
4 1A
5 1Ā
6 1B
7 1B̄
8 1N
9 1N̄
10 1A*
11 1B*
12 1N*
13 1M
14 DC5.2V
15 DC24V
16 2A
17 2Ā
18 2B
19 2B̄
20 2N
21 2N̄
22 2A*
23 2B*
24 2N*
25 1M
26 DC5.2V
27 DC24V
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46 2L+
47 2M
48
INTF
EXTF
CH1
CR
DIR
CH2
CR
DIR
CH1 IN
0
1
2
CH1 OUT
0
1
CH2 IN
0
1
2
CH2 OUT
0
1
1I0
1I1
1I2
1Q0
1Q1
2I0
2I1
2I2
2Q0
2Q1

Terminal assignment of the front connector

The following table shows the terminal assignment for the front connector of the FM 450-1.

Terminal	Name	Input/ output	Function			
			5 V encoder RS 422, symmetrical	24 V encoder, asymmetrical	24 V pulse generator with direction level	24 V initiator
1			-			
2			-			
3	1L+	Input	24 V auxiliary voltage supply for encoders			
4	1M	Input	Auxiliary voltage ground to supply encoders			
Counter 1						
5	1 A	Input	Encoder signal A	-		
6	1 /A	Input	Encoder signal /A	-		
7	1 B	Input	Encoder signal B	-		
8	1 /B	Input	Encoder signal /B	-		
9	1 N	Input	Encoder signal N	-		
10	1 /N	Input	Encoder signal /N	-		
11	1 A*	Input	-	Encoder signal A*		
12	1 B*	Input	-	Encoder signal B*	Directional signal	-
13	1 N*	Input	-	Encoder signal N*	-	
14	1M	Output	Ground for encoder power supply			
15	5.2 V DC	Output	Encoder power supply 5.2 V	-		
16	24 V DC	Output	-	Encoder power supply 24 V		
Counter 2						
17	2 A	Input	Encoder signal A	-		
18	2 /A	Input	Encoder signal /A	-		
19	2 B	Input	Encoder signal B	-		
20	2 /B	Input	Encoder signal /B	-		
21	2 N	Input	Encoder signal N	-		
22	2 /N	Input	Encoder signal /N	-		
23	2 A*	Input	-	Encoder signal A*		
24	2 B*	Input	-	Encoder signal B*, directional signal		
25	2 N*	Input	-	Encoder signal N*		
26	1M	Output	Ground for encoder power supply			
27	5.2 V DC	Output	Encoder power supply 5.2 V	-		
28	24 V DC	Output	-	Encoder power supply 24 V		
Counter 1						
29	1I0	Input	Digital input 1I0			
30	1I1	Input	Digital input 1I1			
31	1I2	Input	Digital input 1I2 (set counter)			
32			-			

Terminal	Name	Input/ output	Function			
			5 V encoder RS 422, symmetrical	24 V encoder, asymmetrical	24 V pulse generator with direction level	24 V initiator
33	1Q0	Output	Digital output 1Q0			
34	1Q1	Output	Digital output 1Q1			
35			-			
36			-			
37			-			
38			-			
39			-			
40			-			
Counter 2						
41	2I0	Input	Digital input 2I0			
42	2I1	Input	Digital input 2I1			
43	2I2	Input	Digital input 2I2 (set counter)			
44			-			
45	2Q0	Output	Digital output 2Q0			
46	2Q1	Output	Digital output 2Q1			
47	2L+	Input	24 V load voltage for the digital inputs and outputs			
48	2M	Input	Load voltage ground for digital inputs and outputs			

Note

The circuits for the counter inputs (encoder supply, encoder signals) are non-isolated from the ground of the CPU. You should therefore connect terminal 4 (1M) to the CPU ground using a low-resistance connection.

If you supply the encoders with external voltage, you must also connect the mass of this external voltage supply to the ground of the CPU.

You will find more information on construction and wiring in the manual (http://support.automation.siemens.com/WW/view/en/1118412) for the FM 450-1 counter module.

Auxiliary voltage 1L+, 1M

Connect a direct voltage of 24 V to the 1L+ and 1M terminals for the 5 V and 24 V encoder voltage supply.

An integrated diode protects the module from a reversal in the polarity of the auxiliary voltage.

The module monitors whether the auxiliary voltage is connected.

5.2 V DC encoder power supply

The module generates a voltage of 5.2 V from the auxiliary voltage 1L+/1M at a maximum current of 300 mA per count channel; this is provided at the corresponding "DC5.2V" connection for the short-circuit proof supply of a 5 V encoder.

24 V DC encoder power supply

1L+/1M is provided at output "DC24V" for 24V short-circuit proof encoder voltage supply. The encoder power supply is short-circuit checked.

5 V encoder signals A and /A, B and /B, N and /N

You can connect incremental encoders with 5 V differential signal to each counter in accordance with RS 422, i.e. incremental encoders with the differential signal A and /A, B and /B, N and /N.

The A and /A, B and /B, N and /N signals are connected via the correspondingly labeled terminals

The signals N and /M are only connected if you wish to set the counter to the zero mark of the encoder.

The inputs are not isolated from the S7-400 bus.

24 V encoder signals A*, B* and N*

24 V signals are identified with the letters A*, B* and N*.

You can connect three different types of encoders to each counter:

- Incremental encoder with 24 V signals:
  The signals A*, B* and N* are connected via the correspondingly labeled pins.
- Pulse encoders without directional level:
  The signal is connected to terminal A*
- Pulse encoders with directional level:
  The count signal is connected to the terminal A*. The directional level is connected to terminal B*.

The inputs are not isolated from the S7-400 bus.

Input filter for 24 V encoder signals

To suppress faults, you can configure input filters (RC elements) with a uniform filtering time for the 24 V inputs A*, B* and N*. The following two input filters are available for each counter:

Characteristics	Input filter 1 (default setting)	Input filter 2
Typical input delay	1 µs	15 µs
Maximum count frequency	200 kHz	20 kHz
Minimum pulse width of the count signals	2.5 µs	25 µs

Digital inputs

You can use digital inputs 1I0 and 1I1 to control the gates of counter 1.
Digital inputs 2I0 and 2I1 are used to control the gates of counter 2.
The gates can be both level-controlled and edge-controlled.

The 1I2 digital input is for setting counter 1 to the load value.
Digital input 2I2 is for setting counter 2 to the load value.

The digital inputs are operated with a 24 V nominal voltage.

The digital inputs are isolated from the bus of the S7-400 and the counter inputs.

Input filter for digital inputs

To suppress faults you can assign parameters to the input filters (RC elements) with a uniform filtering time for the digital inputs 1I0, 1I1 and 1I2 or 2I0, 2I1 and 2I2. The following two input filters are available:

Characteristics	Input filter 1 (default setting)	Input filter 2
Typical input delay	1 µs	15 µs
Minimum pulse width of input signals	2.5 µs	25 µs

Digital outputs

FM 450-1 has the digital outputs 1Q0 and 1Q1 (for counter 1) and 2Q0 and 2Q1 (for counter 2) for directly triggering control processes.

The digital outputs are supplied by load voltage 2L+.

The digital outputs are isolated from the bus of the S7-400 and the counter inputs.

The digital outputs are Sourcing outputs and can carry load current of 0.5 A. They are protected against overload and short circuit.

Note

The direct connection of relays and cutouts can be executed without external wiring

The time response of the digital outputs depends on parameter assignment.

2L+/2M load voltage

To ensure the power supply for digital outputs 1Q0 and 1Q1 or 2Q0 and 2Q1, you must connect a load voltage of 24 V to the module terminals 2L+ and 2M.

An integrated diode protects the module from reversing the polarity of the load voltage.

The load voltage 2L+/2M is not monitored by the FM 450-1.

Configuring FM 450-1

Configuring FM 450-1

Specify channel

Copy channel

You can copy the parameters of one channel to another.

1. Select the channel whose parameters are to be copied (source channel).
2. Select the channel to which the parameters are to be copied (target channel).
3. Press the "Copy" button.

Channel n operating modes

Selecting an operating mode

Description

You define counter module functionality by specifying an operating mode.

Operating mode	Description
Count continuously (Page 9083) (with or without gate)	The FM 450-1 counts continuously from the current counter value.
Count once (Page 9084) (with hardware or software gate)	When the gate opens, the FM 450-1 counts from the load value to the count limit.
Count periodically (Page 9085) (with hardware or software gate)	When the gate opens, the FM 450-1 counts from the load value between the count limits.

The default setting is the "Count continuously" operating mode.

Selecting the Gate Function

Description

You define the count process in the selected operating mode by specifying a gate control.

- without gate
  All gates are open.
- Software gate (Page 9102)
  The software gate is opened and closed by the input parameter / control bit SW_GATE.
- Hardware gate (Page 9102), level-controlled or edge-controlled
  The hardware gate is opened and closed by the corresponding signals at the digital inputs.

The default setting is "Without gate".

Selecting the counting range

Description

There is a 32-bit wide count register on the module. With the Count range, you select whether the module is to count only in the positive range or whether the 32nd bit is interpreted as a sign bit thus allowing negative numbers to be represented.

Select the desired counting range.

For more information, please see "Counting ranges and count limits (Page 9081)".

Violation of the high limit, low limit and zero pass

A bit is set in the DB of the CNT_CTRL instruction for both count limits when the high or low limit is violated.

A bit is also set in the DB in the "-31 to +31-bits" counting range in the event of a zero pass.

Depending on the count direction, violation of the high or low limit is also indicated in the "0 to +32 bits" counting range upon zero pass.

Event	Status bit in DB
Violation of the high limit	STS_OFLW is set.
Violation of the low limit	STS_UFLW is set.
Zero pass	STS_ZERO is set.

Initiating Hardware Interrupts

You can also signal the violation of the high limit and low limit and the zero pass events using hardware interrupts.

Channel n encoder

Selecting the encoder signal type

Selecting the signal type

1. Select the type of signal for the encoder used.
   In line with your selection and the operating mode, any option not possible for this specific encoder will be disabled. A disabled option is identified by grayed font in the corresponding selection window.
2. Select the options to be used with the selected encoder.

You can select the signal type for one of the following encoders:

- Encoder: 5 V incremental (Page 9198) (5 V differential signal)
- Encoder: 24 V differential signal (Page 9198) (24 V track A+B phase shifted)
- Encoder: 24 V pulse and direction (Page 9199)
- Encoder: 24 V initiator (Page 9199)

Encoder: 5 V differential signal

5 V incremental encoder

Select the signal type "5V differential signal" to use a 5 V differential encoder with differential signal A, /A, B, /B and N, /N.

The following options are available for this encoder:

Parameter	Option	Meaning
Signal evaluation (Page 9108)	Single	Only one edge of signal A is evaluated.
	Double	Both edges of signal A are evaluated.
	Quadruple	Both edges of signals A and B are evaluated.
Count direction (Page 9105)	Normal	To adapt the count direction to the process
	Inverted	To adapt the count direction to the process
Max. count frequency	–	The maximum count frequency for this encoder is 500 kHz.
Encoder inputs	–	Not relevant
Monitoring (Page 9105)	A+B+N	All signal pairs are monitored for short-circuit and wire break.
	A+B	Signal pairs A and B are monitored.
	A	Only signal pair A is monitored.
	None	No signal pair is monitored.

Encoder: 24 V incremental

24 V incremental encoder

Select signal type "24 V track A+B phase shifted" to use a 24 V incremental encoder with signals A*, B* and N*.

The following options are available for this encoder:

Parameter	Option	Meaning
Signal evaluation (Page 9108)	Single	Only one edge of signal A is evaluated.
	Double	Both edges of signal A are evaluated.
	Quadruple	Both edges of signals A and B are evaluated.
Count direction (Page 9106)	Normal	To adapt the count direction to the process
	Inverted	To adapt the count direction to the process
Max. count frequency (Page 9200)	200 kHz	A filter with 1 µs delay is active at the inputs.
	20 kHz	A filter with 15 µs delay is active at the inputs.

Parameter	Option	Meaning
Encoder inputs (Page 9200)	Sinking output	The encoder switches to 0 V (ground).
	Sourcing output/ push pull	The encoder switches to 24 V.
Monitoring	–	Not relevant

Encoder: 24 V pulse and direction

24 V pulse generator

Select signal type "24 V pulse and direction" to use a 24 V pulse generator with count signal A* and direction signal B*.

The following options are available for this encoder:

Parameter	Option	Meaning
Signal evaluation	–	Not relevant
Count direction	–	Not relevant
Max. count frequency (Page 9200)	200 kHz	A filter with 1 µs delay is active at the inputs.
	20 kHz	A filter with 15 µs delay is active at the inputs.
Encoder inputs (Page 9200)	Sinking output	The encoder switches to 0 V (ground).
	Sourcing output/ push pull	The encoder switches to 24 V.
Monitoring	–	Not relevant

Encoder: 24 V initiator

24 V initiator

Select the signal type "24 V initiator" to use a 24 V pulse generator with count signal A*.

The following options are available for this encoder:

Parameter	Option	Meaning
Signal evaluation	–	Not relevant
Count direction	–	Not relevant
Max. count frequency (Page 9200)	200 kHz	A filter with 1 µs delay is active at the inputs.
	20 kHz	A filter with 15 µs delay is active at the inputs.
Encoder inputs (Page 9200)	Sinking output	The encoder switches to 0 V (ground).
	Sourcing output/ push pull	The encoder switches to 24 V.
Monitoring	–	Not relevant

Encoder input parameter assignment

Encoder input parameter assignment

The count direction is specified by means of the encoder output parameter assignment. The table shows a change in count direction based on parameter assignment.

Parameter assignment	Terminal B*	Count direction
Sinking output	not wired	Down
	Short-circuited to ground	Up
Sourcing output/push pull	not wired	Up
	24 V connected	Down

Set the "24 V pulse and direction" parameter when selecting the encoder.

You cannot reverse the direction of these count signals by inverting the B* signal.

Note

As all signals are added, this type of evaluation may cause the count value to "drift off" at the edge if the count signal oscillates.

Input filters for 24 V counter inputs

Input filters for 24 V counter inputs

For the purpose of suppressing interference, you can configure input filters with a uniform filter time for the 24 V counter inputs A*, B* and N* and for the digital inputs. Input filters available:

Characteristics	Input filter 1 (default setting)	Input filter 2
Typical input delay	1 µs	15 µs
Maximum count frequency	200 kHz	20 kHz
Minimum pulse width of the count signals	2.5 µs	25 µs

Channel n inputs

Selecting gate signals

Controlling the hardware gate

The following options are available for hardware gate control:

- Level-controlled hardware gate (with level at input 1I0 or 2I0)
- Level-controlled hardware gate (for opening with edge at input 1I0 or 2I0 and closing with edge at input 1I1 or 2I1)

For additional information on this topic, please refer to "Hardware gate (Page 9102)".

Selecting minimum pulse width

Minimum pulse width

To suppress faults, you can configure input filters (RC elements) with a uniform filtering time for the 24 V inputs A*, B* and N* and for the digital inputs.

Two input filters are available with the following data:

Characteristics	Input Filter 1 (default setting)	Input Filter 2
Typical input delay	1 µs	15 µs
Maximum count frequency	200 kHz	20 kHz
Minimum pulse width of the count signals	2.5 µs	25 µs

Setting the counter

Setting the counter

Overview

If you want to start the count from a specific value (the load value), you must parameterize the signal that is to be used to set this counter to the load value. You can set the counter as follows:

- With the L_DIRECT input parameter of the CNT_CTRL instruction
- With an external signal, using either the 1I2 (2I2) input or the input in combination with the encoder zero mark

Load value

Any number within the limits of the count range can be set for the count range.

The load value is interpreted according to the selected counting range. If, for example, you specify FFFF FFFF H as the load value, this will be interpreted as 4 294 967 295 in the "0 to +32 bits" counting range and as -1 in the counting range "-31 to +31 bits".

Enter the load value in the DB of instruction CNT_CTRL.

The following ranges of values are permitted for the load value:

Range of values for load values	
Low limit	Maximum low count limit
High limit	Maximum high count limit

Setting the Counter with an External Signal

You can select two different external signals with which you can set a counter to the load value:

- Input 1I2 (2I2) (Page 9202) only
- Input 1I2 (2I2) and the zero mark (Page 9204) of the encoder

You use the zero mark of the encoder if you want to synchronize the counter to a specific counter status at a specific point in your process. This improves the precision of the counting process.

The counter is set independently of the Count mode.

When the counter has been set with an external signal, bit STS_SYNC will be set in the DB.

Note

Synchronization of a counter with zero mark only makes sense if the gate is open.

If you have only enabled one count direction when setting a counter with an external signal, you must note that when the gate is closed the current count direction is saved (frozen). This enables synchronization of the counter against the enabled count

Setting the counter with input 1I2 (2I2)

Setting the counter with input 1I2 (2I2)

The counter can be loaded with the load value by means of a rising edge at input 1I2 (2I2) (DI set).

You can define the behavior of the FM 450-1 with a rising edge at input 1I2 (2I2) with the variables ENSET_UP and ENSET_DN in the DB of the CNT_CTRL instruction and by assigning parameters.

Parameter	Behavior of the FM 450-1
ENSET_UP set	The counter is only set if counting up.
ENSET_DN set	The counter is only set if counting down.

Parameter	Behavior of the FM 450-1
ENSET_UP and ENSET_DN set	The counter is set when counting up and counting down.
Parameterization "Single setting of counter"	The counter is set only on the first rising edge at input 1I2 (2I2). If the counter is to be set again, you must first set ENSET_UP or ENSET_DN again. The counter is then set again on the next rising edge at input 1I2 (2I2).
Parameterization "Multiple setting of counter"	The counter is set at every rising edge at input 1I2 (2I2) as long as ENSET_UP and/or ENSET_DN is set.

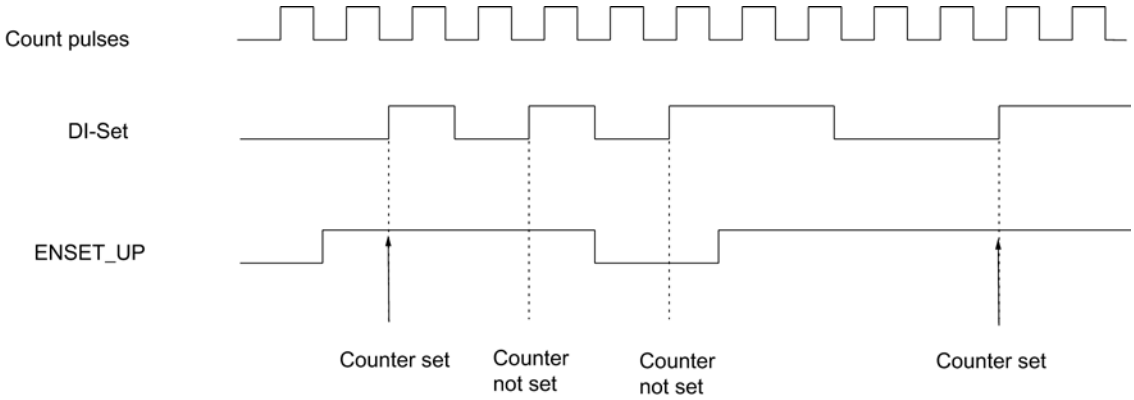
Note

It is imperative that you set one of the two variables ENSET_UP and/or ENSET_DN so that the relevant counter can be set via the input 1I2 (2I2).

Set once with input 1I2 (2I2)

The figure below shows setting counter 1 once with input 1I2. The same procedure applies for setting counter 2 with input 2I2. In the case shown, only ENSET_UP is set, i.e. the counter is set when counting up.

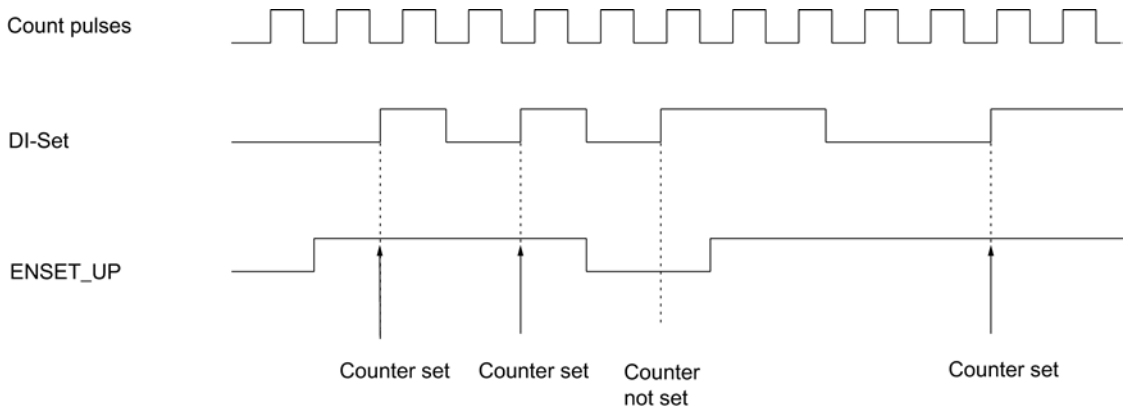
Counter 1 is set with the first rising edge at input 1I2 as long as ENSET_UP is also set. If you want to set the counter again, you must first reset ENSET_UP and then set it again. The next rising edge at input 1I2 will then result in the counter being set.



Setting repeatedly with input 1I2 (2I2)

The figure below shows the repeated setting of counter 1 with input 1I2. The same procedure applies for setting counter 2 with input 2I2. In the case shown, only ENSET_UP is set, i.e. the counter is set when counting up.

Counter 1 is set with every rising edge at input 1I2 as long as ENSET_UP is also set. If you reset ENSET_UP, the counter will not be set with a rising edge at input 1I2. Only once you have set ENSET_UP again will the next rising edge at input 1I2 set the counter.



Setting the counter with input 1I2 (2I2) and the zero mark

Setting the counter with input 1I2 (2I2) and the zero mark

If you assign parameters so that the counter is set by the encoder zero mark ("Evaluate zero mark for setting"), the counter will be set to the load value at the rising edge of the zero mark.

Setting is performed only if the 1I2 (2I2) input is also set at the time of the rising edge of the zero mark.

You can define the behavior of the FM 450-1 in the event of a zero mark rising edge with the variables ENSET_UP and ENSET_DN in the DB of the CNT_CTRL instruction and by assigning parameters.

Input parameters	Behavior of the FM 450-1
ENSET_UP set	The counter is only set if counting up.
ENSET_DN set	The counter is only set if counting down.
ENSET_UP and ENSET_DN set	The counter is set when counting up and counting down.
Parameterization "Single setting of counter"	The counter is set only at the first rising of the zero mark. If the counter is to be set again, you must first set ENSET_UP or ENSET_DN again (edge evaluation). The counter is then set again with the next rising edge of the zero mark.
Parameterization "Multiple setting of counter"	The counter is set at every rising edge of the zero mark as long as ENSET_UP and/or ENSET_DN are set.

Note

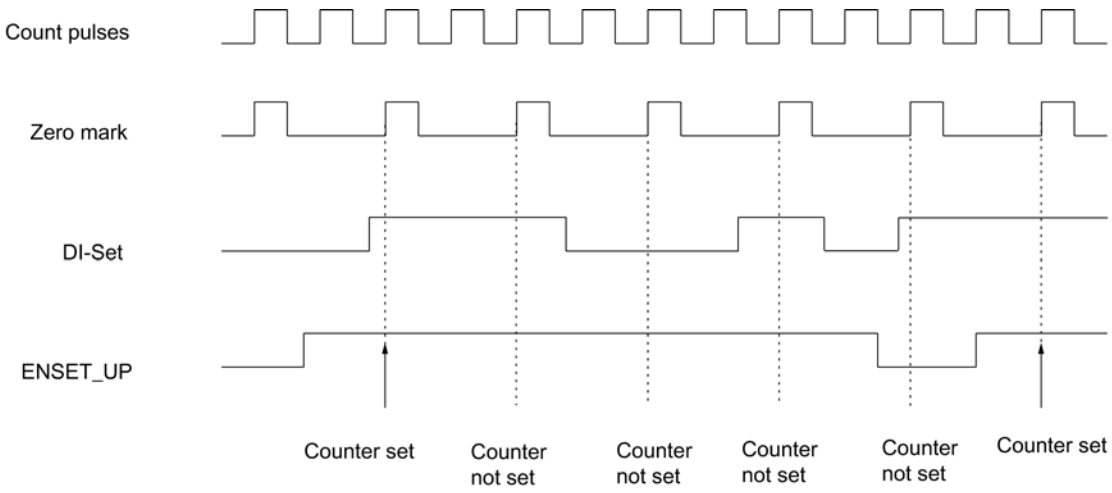
It is imperative that you set one of the two variables ENSET_UP and/or ENSET_DN and input 1I2 (2I2) so that the counter can be set with the zero mark.

Set once with input 1I2 (2I2) and the zero mark

The figure below shows setting the counter 1 once with the zero mark. The same procedure applies for setting counter 2 with input 2I2. In the case shown, only ENSET_UP is set, i.e. the counter is set when counting up.

Counter 1 is set on the first rising edge of the zero mark provided that ENSET_UP and input 1I2 are also set.

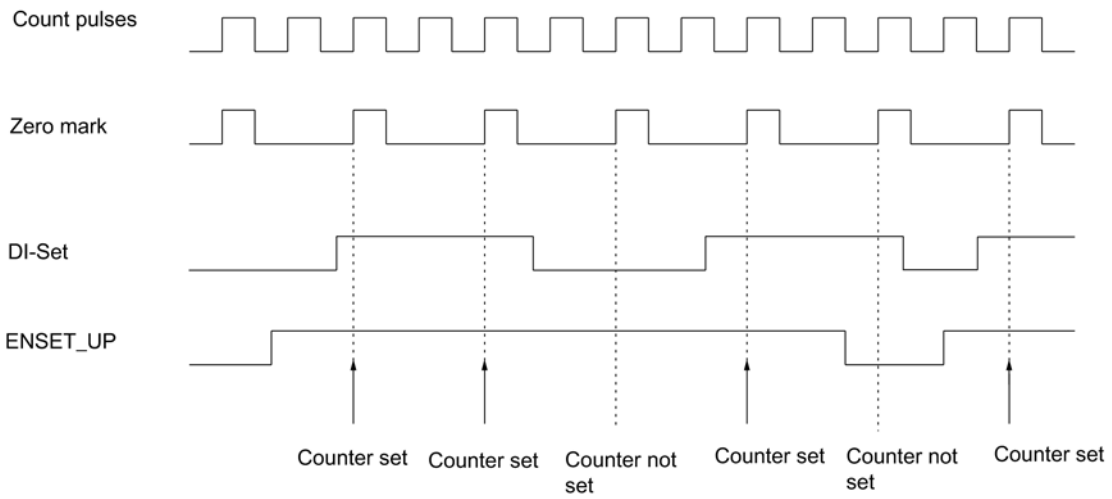
If you want to set counter 1 again, you must first reset ENSET_UP and then set it again. If input 1I2 is not set, the counter will be set upon the first zero mark after it is. If input 1I2 is set, setting will be carried out with the next zero mark.



Repeated setting with input 1I2 (2I2) and the zero mark

The figure below shows the repeated setting of counter 1 with the zero mark. The same procedure applies for setting counter 2 with input 2I2. In the case shown, only ENSET_UP is set, i.e. the counter is set when counting up.

The counter is set with every rising edge of the zero mark provided that ENSET_UP and input 1I2 are set.



Channel n outputs

Specifying pulse duration

Pulse duration

You can set a pulse duration in order to adapt to the actuators (contactors, actuators, etc) used in your process. The pulse duration indicates how long outputs Q0 and Q1 are active when a comparison value is reached.

The pulse duration specifies the minimum period for which the output is to be set. This setting is only effective if you preselect the output behavior accordingly. The pulse duration setting is not effective if the output is to be activated at a value between the comparison value and high limit violated or low limit violated.

The pulse duration starts when the output is set. The pulse duration inaccuracy is less than 1 ms.

A value between 0 and 500 ms may be set for the pulse duration. This value applies to both outputs together.

If the pulse duration = 0 ms, the output is set when the comparison value is reached and is reset at the next count pulse.

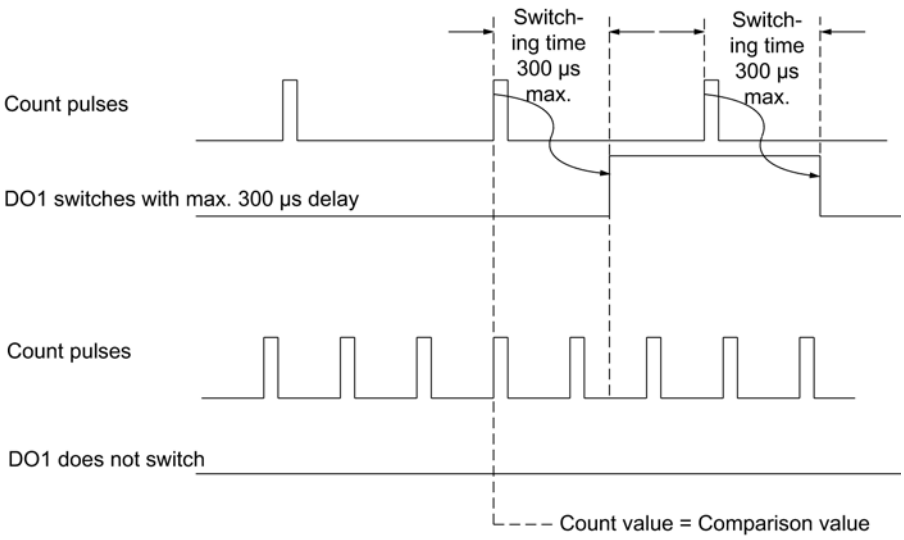
The value 0 is set by default for the pulse duration.

Note

If you specify a pulse duration of 0 ms, the output will be activated for as long as the counter value is equal to the comparison value (see top of figure below).

Control pulses may be lost at the outputs if the time interval between the count pulses is less than the digital output switching times (up to 300 µs) (see bottom half of figure below).

You should therefore make sure that the interval between the count pulses is greater than the switching times of the digital outputs.



Specifying the behavior of the outputs

Comparison values and outputs

Description

You can store two comparison values (comparison value 1 and 2) for the counters in the FM 450-1; these are assigned to the two relevant digital outputs (comparison value°1: 1Q0 or 2Q0, comparison value 2: 1Q1 or 2Q1). The comparison values are compared with the current counter value. When the counter value reaches a comparison value, the relevant output can be set.

Comparison Values 1 and 2

You enter the two comparison values in the DB of the CNT_CTRL instruction (CMP_V1, CMP_V2) and transfer them to the FM 450-1 by setting the bits T_CMP_V1 or T_CMP_V2. This does not affect the count.

The comparison values must be within the limits of the selected count range. The comparison value is interpreted according to the selected count range. If you specify FFFF FFFF H as the comparison value, for example, this value will be interpreted as 4,294,967,295 in the "0 to +32 bits" counting range and as -1 in the"-31 to +31 bits" counting range.

The valid value ranges for the comparison values are as follows:

Range of values for comparison values	
Low limit	Maximum low count limit
High limit	Maximum high count limit

Enabling the Outputs

Before the outputs can be controlled, you must first enable them by setting the relevant bits in the DB. If you reset one of these bits the associated output is switched off immediately even if you have parameterized a pulse duration for them.

The following bits must be set before you can enable the outputs:

Output	... Is enabled by enable bit
Q0	CTRL_DQ0
Q1	CTRL_DQ1

Behavior of the outputs in counting modes

Specifying the behavior of the outputs

Behavior of outputs Q0 and Q1

Select the required behavior for the output in question. Activate the corresponding option button:

- No comparison (Page 9208)
- On when counter value >= comparison value (Page 9209)
- On when counter value <= comparison value (Page 9209)
- Active on reaching the comparison value for pulse duration up (Page 9210)
- Active on reaching the comparison value for pulse duration down (Page 9210)
- Active on reaching the comparison value for pulse duration up/down (Page 9209)

Note

To activate the settings, enter values for comparison values 1 and 2 in the DB with the counter data and transfer them to the module with the CNT_CTRL instruction; you must also set the CTRL_DQ0 and CTRL_DQ1 bits in the DB in order to enable the outputs. See also "Comparison values and outputs (Page 9207)".

No comparison

No comparison

The output remains deactivated and is not affected by comparison value, zero pass, high limit violated or low limit violated. The output Qx may be used as a purely digital output.

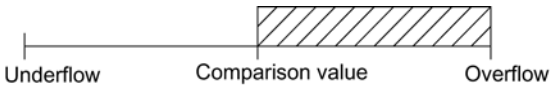
On when counter value >= comparison value

On when counter value >= comparison value

The output is set to 1 if the counter is in the range between the comparison value and high limit violated.

Setting the counter to a value between the comparison value and high limit violated sets the output to 1.

The hatched area in the graphic below means that the output is set in this range.



Please note the basic conditions for the behavior of the outputs (Page 9211).

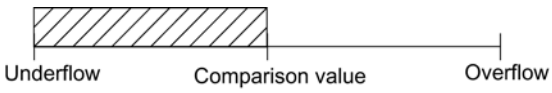
On when counter value <= comparison value

On when counter value <= comparison value

The output is set to 1 if the counter is in the range between the comparison value and low limit violated.

Setting the counter to a value between the comparison value and low limit violated activates the output at 1.

The hatched area in the graphic below means that the output is set in this range.



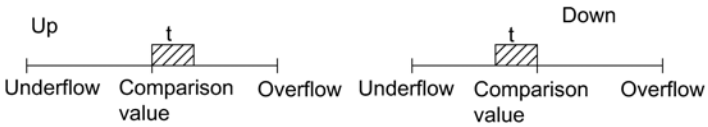
Please note the basic conditions for the behavior of the outputs (Page 9211).

Active on reaching the comparison value for pulse duration up/down

Active on reaching the comparison value for pulse duration up/down

The output is set to 1 for the duration of the pulse when the counter has reached the comparison value irrespective of the count direction.

The hatched area in the graphic below means that the output is set in this range (t = pulse duration).

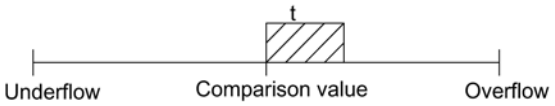


Active on reaching the comparison value for pulse duration up

Active on reaching the comparison value for pulse duration up

The output is set to 1 for the pulse duration when the counter reaches the comparison value when counting up.

The hatched area in the graphic below means that the output is set in this range (t = pulse duration).



Please note the basic conditions for the behavior of the outputs (Page 9211).

Active on reaching the comparison value for pulse duration down

Active on reaching the comparison value for pulse duration down

The output is set to 1 for the pulse duration when the counter reaches the comparison value when counting down.

The hatched area in the graphic below means that the output is set in this range (t = pulse duration).

Please note the basic conditions for the behavior of the outputs (Page 9211).

Basic conditions for the behavior of the digital outputs

Marginal Conditions for the Behavior of the Digital Outputs

You must observe the following basic conditions when configuring the behavior of the digital outputs:

If...	Then...
... ...you want to configure an output "On when counter value >= comparison value or <= comparison value",	... ...You must ensure that the time between these events is greater than the minimum switching time of the outputs (switching time: 300 µs); the control pulses at the outputs will otherwise be lost. No new pulse will be triggered if the counter value reaches the relevant comparison value again while the output is still active. A further pulse can only be triggered when the output is no longer active.
... you want to configure an output that is "Active on reaching the comparison value for pulse duration up",	... ...You must not enable a hardware interrupt on `Reaching the relevant comparison value down'.
... you want to configure an output that is "Active on reaching the comparison value for pulse duration down",	... ...You must not enable a hardware interrupt on `Reaching the (relevant) comparison value up'.

Channel n enable hardware interrupts

Enabling hardware interrupt trigger

Introduction

You can set which events are to trigger a hardware interrupt for the FM 450-1 counters.

What is a Hardware Interrupt?

If you want there to be a response to a specific event irrespective of the CPU cycle, each FM 450-1 counter can trigger a hardware interrupt. The CPU interrupts the cyclic program on receiving the interrupts and executes the hardware interrupt OB.

Enabling the Hardware Interrupt

Hardware interrupts are enabled in two steps:

1. When configuring the hardware, you enable the interrupts for the module in the basic parameters and choose whether the module is to trigger diagnostic and/or hardware interrupts.
2. You then select the relevant events (Page 9212) under "Enable hardware interrupts".

Process interrupt OB, OB 4x

If a hardware interrupt occurs, the user program is interrupted, the data is transferred from the module to the start information of OB 4x (Page 9212) and OB 4x is called. The process interrupt is acknowledged by exiting the OB 4x.

If no OB 4x is programmed, the CPU goes into STOP mode. If you then switch back to RUN, the hardware interrupt requirements are deleted.

Lost hardware interrupt

If an event occurs that is to initiate a hardware interrupt and the same previous event has not yet been acknowledged, no further hardware interrupt is initiated; the hardware interrupt is lost. This may lead to the "Hardware interrupt lost" diagnostic interrupt, depending on the parameterization.

Default setting

No hardware interrupts are enabled in the default setting.

Events which can trigger a hardware interrupt

Which events can trigger a hardware interrupt?

The following events can trigger a hardware interrupt (independently of each other for each of the two counters) during operation of the FM 450-1:

- Opening of the gate (in the operating modes with hardware or software gate)
- Closing of the gate (in the operating modes with hardware or software gate)
- Violation of the high limit
- Violation of the low limit
- Zero pass
- Reaching comparison value 1 or 2 in the up direction
- Reaching comparison value 1 or 2 in the down direction
- Setting the counter (DI set)

Select the check box for the relevant events.

You can select any number of events for triggering a hardware interrupt. Observe the basic conditions for output behavior (Page 9211) applicable to hardware interrupts on reaching a comparison value.

OB 4x start information

OB 4x start information

The temporary variable OB4x_POINT_ADDR is described in the OB 4x start information.

The OB4x_POINT_ADDR variable (bytes 8 to 11) comprises 4 bytes. The information on the event that triggered the hardware interrupt is entered in these 4 bytes.

The table shows which bits are set for which interrupt. Any bits not listed are irrelevant and are zero.

Byte	Bit	Meaning: Interrupt in the case of...
Counter 1		
8	0	Opening the gate
	1	Closing the gate
	2	Violation of the high limit
	3	Violation of the low limit
	4	Reaching comparison value 1 in the up direction
	5	Reaching comparison value 1 in the down direction
	6	Reaching comparison value 2 in the up direction
	7	Reaching comparison value 2 in the down direction
9	0	Zero pass
	5	Setting the counter (DI set) (synchronization)
Counter 2		
10	0	Opening the gate
	1	Closing the gate
	2	Violation of the high limit
	3	Violation of the low limit
	4	Reaching comparison value 1 in the forward direction
	5	Reaching comparison value 1 in the backward direction
	6	Reaching comparison value 2 in the forward direction
	7	Reaching comparison value 2 in the backward direction
11	0	Zero pass
	5	Setting the counter (DI set) (synchronization)

Using FM 450-1 from the user program

Integrating the FM 450-1 into a user program

Instructions are available for integrating the FM 450-1 into a user program. These enable you to handle the desired functions with ease.

The instruction CNT_CTRL can be called in the CPU user program. It ensures communication between the CPU and the FM 450-1.

All data associated with a channel of the module are in the DB of the instruction CNT_CTRL (counter DB (Page 7101)). Use the command "Program blocks > Add new block" to add a CNT_CHANTYPE1 type data block to your project.

The DIAG_INF instruction is also available for FM 450-1. This operation allows you to transfer the diagnostic information to the counter DB.

See also

CNT_CTRL (Page 7096)
DIAG_INF (Page 7097)

FM 450-1: Online & Diagnostics

Current module information

Default diagnostics

FM 450-1 diagnostic LED fault display

Where are the faults displayed?

Faults are indicated by the two red diagnostics LEDs:

- The INTF diagnostics LED displays internal faults in the module.
- The EXTF diagnostics LED displays external faults in the cable connections.

Which fauts are displayed?

The table below shows the error types displayed by the diagnostic LEDs.

Fault type/LED	Cause of fault	Correction
Internal fault Diagnostics LEDINTF	Error in EPROM test	Module replacement
	RAM test error	Module replacement
	Watchdog tripped	Module replacement
	Lost hardware interrupt	Is acknowledged by processing the process interrupt.
External fault Diagnostics LEDEXTF	Auxiliary voltage 1L+/1M not connected or DC24V encoder supply short-circuited	Correct connection
	DC5.2V encoder supply short-circuited or overloaded	Correct connection
	Fault in 5V encoder signals (wire break, short circuit, cable missing)	Correct connection

Initiation of a diagnostics interrupt

All errors except an EPROM test error can trigger a diagnostic interrupt provided you have enabled the diagnostic interrupt in the basic parameters during parameter assignment. The diagnostic data records DS0 and DS1 (Page 9216) will show you which error has caused the LED to light up.

Channel-specific diagnostics

Triggering diagnostics interrupts

Definition of a diagnostics interrupt

You can determine reactions to internal or external errors in the user program, by programming a diagnostics interrupt which interrupts cyclic program execution on the CPU, and triggers a call of diagnostics interrupt OB 82.

Events which can trigger a diagnostics interrupt

The following events can trigger a diagnostic interrupt:

- Faults in the external auxiliary 1L+/1M voltage.
- Short-circuit or overload at the 5.2 V DC encoder supply
- No module configuration
- Faulty module parameters
- Time monitoring (watchdog) triggered
- RAM defective
- Hardware interrupt lost
- Signal A error (wire break, short-circuit, cable missing)
- Signal B error (wire break, short-circuit, cable missing)
- Signal N error (wire break, short-circuit, cable missing)

Enabling diagnostics interrupts

When configuring the hardware, you enable the interrupts for the module in the basic parameters and choose whether the module is to trigger diagnostic and/or hardware interrupts.

Reactions to a diagnostics interrupt

When an event occurs that could trigger a diagnostic interrupt:

- The diagnostics information is stored in the diagnostic data records DS0 and DS1 (Page 9216) in the module.
- The group error LED is lit.
  The group error LED does dark when the error is eliminated.
- Call of the diagnostics interrupt OB 82.
- The diagnostic data record DS0 is entered in the start information of the diagnostic interrupt OB.
- The count continues unchanged if an OB 82 has been programmed. If no OB 82 has been programmed, the CPU will go into STOP mode.

Default setting

No diagnostic interrupts are enabled in the default setting.

Diagnostics data record DS0 and DS1

Diagnostics data record DS0 and DS1

The information showing the event which has triggered a diagnostics interrupt is written to the diagnostics data records DS0 and DS1. The diagnostic data record DS0 comprises four bytes. DS1 comprises 16 bytes; the first four bytes are identical to DS0.

Reading data records from the module

Diagnostic data record DS0 (Page 9216) is automatically transferred to the start information when the diagnostic OB is called. These four bytes are written to the local date (bytes 8 to 11) of OB 82.

You can read the diagnostic data record DS1 (Page 9217) (and thus also the content of the DS0) from the module with the DIAG_INF (Page 7097) instruction. However, this is only useful if DS0 reports a channel error.

Assignments of diagnostics data record DS0

Assignments of diagnostics data record DS0 in the start information

The following table shows the assignment of the diagnostic data record DS0 in the start information. Any bits not listed are insignificant, and zero.

Byte	Bit	Meaning	Remarks	Event no.
0	0	Module in error state	Set upon all diagnostics events	8:x:00
	1	Internal error	Set at all internal error events: • RAM test error • Time monitoring (watchdog) triggered • Lost hardware interrupt	8:x:01
	2	External error	Set at all external error events: • Auxiliary voltage 1L+/1M not connected, or short-circuit at the 5.2 V DC encoder supply. • Short-circuit or overload at the 5.2 V DC encoder supply • 5 V signal faults • Parameter error	8:x:02
	3	Channel error	Is always assigned 1 for internal and external faults.	8:x:03
	4	Failure of the external auxiliary voltage	Check the voltage	8:x.04
	7	Parameter error	See Data errors (Page 9218) for a more detailed breakdown	8:x:07

Byte	Bit	Meaning	Remarks	Event no.
1	0...3	Type class	Always assigned the value 8	
	4	Channel information	Always assigned the value 1	
2	3	Time monitoring (watchdog) triggered	Module defective, or heavy interference	8:x:33
3	3	RAM defective	Module defective, or heavy interference	8:x:43
	6	Hardware interrupt lost	Check the configuration. A hardware interrupt was detected and can not be reported, because the same event is not yet acknowledged by the CPU	8:x:46

Assignment of diagnostic data record DS1

Assignment of diagnostic data record DS1

Diagnostic data record DS1 consists of 12 bytes. The first 4 bytes are identical to diagnostic data record DS0. The following table shows the assignment of the remaining bytes. Any bits not listed are irrelevant and are zero. The DIAG_INF (Page 7097) instruction writes this data record to the DB of the CNT_CTRL instruction starting at DW54.

Byte	Bit	Meaning	Remarks	Event no.
4	0...6	Channel type	Always assigned the value 76H	
	7	Further channel types	Always assigned zero value	
5	0...7	Diagnostics data length	Always assigned CH	
6	0...7	Number of channels	Is always assigned 2	
7	0…1	Channel error vector	Bit 0= 1: Fault on channel 1 Bit 1= 1: Fault on channel 2	
8	0	Channel 1 signal A defective		8:x:B0
	1	Channel 1 signal B defective		8:x:B1
	2	Channel 1 signal N defective		8:x:B2
	4	Channel 1 5.2 V encoder supply defective		8:x:B4
	5...7	Reserved		
9		Reserved		
10	0	Channel 2 signal A defective		8:x:B0
	1	Channel 2 signal B defective		8:x:B1
	2	Channel 2 signal N defective		8:x:B2
	4	Channel 2 5.2 V encoder supply defective		8:x:B4
	5...7	Reserved		
11		Reserved		

Data error

Data error events

FM 450-1 checks all new parameters transferred to the module. The module reports any errors returned in this check.

Where are data errors displayed?

The CNT_CTRL (Page 7096) instruction enters the data errors with the error numbers in the CNT_CTRL DB. You can access this data word in the user program using the DA_ERR_W variable.

Possible data errors

The table below lists the data errors and their error numbers and meaning.

No.	Meaning
0	No error
201	Incorrect receiver selection value
202	Invalid diagnostics value of the signal pair
203	Incorrect value for signal evaluation
204	Incorrect 24 V count signal input filter value
205	Invalid value at digital input filters
206	Reversal of direction not allowed
207	Incorrect 1Q0 (2Q0) behavior configuration
208	Incorrect 1Q1 (2Q1) behavior configuration
209	Pulse duration out of limits
211	Wrong operating mode selected
212	No gate or both gates defined
215	A different count direction was selected for configuring output behavior "Active on reaching the comparison value for pulse duration up or down" in parameter assignment for the hardware interrupt "Reaching the comparison value up or down". The directions set must match.
216	Gate control interrupts are only possible in modes with gate control.
217	Interrupt on "Reaching comparison values" is not valid for output behavior "On when counter value >= comparison value" or "On when counter value <= comparison value".

How are data errors acknowledged?

Correct the parameter values according to specifications. Transfer the corrected parameter set to the FM 450-1 again. It will recheck the parameters and delete the data error in the DB.

Operator error

Operator errors events

Operator errors occur as a result of improper operation of the module caused by the incorrect input of control signals.

Objects which indicate operator errors

The CNT_CTRL (Page 7096) instruction enters the operator error numbers in the DB. Instruction CNT_CTRL indicates an operator error has occurred when output parameter OT_ERR is set. You can access this data word in your program using the OT_ERR_B variable.

Possible operator errors

The table below lists the operator errors and their error numbers and meaning.

No.	Meaning
0	No error
1	Operating mode can not be started using the SW gate
2	Operating mode can not be canceled
4	Only allowed when the CPU is in STOP

How to acknowledge operator errors

You acknowledge the error with the OT_ERR_A input parameter of the CNT_CTRL (Page 7096) instruction.

Using ET 200S 1Count24V

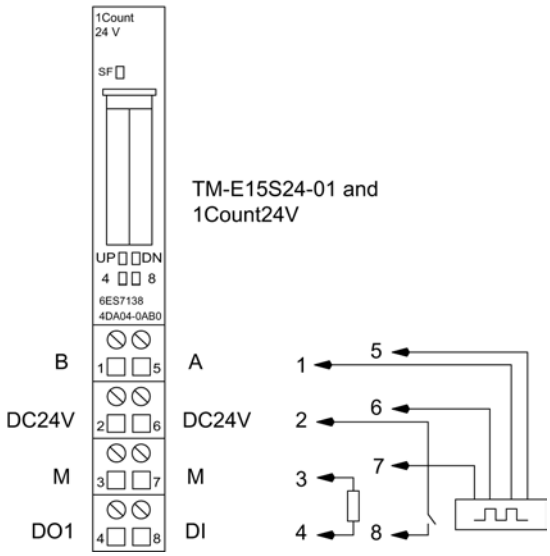
1Count24V terminal assignment

Wiring Rules

The cables (terminals 1 and 5 and terminals 2 and 8) must be shielded. The shield must be supported at both ends. To do this, use the shield support (see the appendix to the ET 200S Distributed I/O System (http://support.automation.siemens.com/WW/view/en/1144348) operating instructions).

Terminal assignment of the 1Count24V

The following table shows the terminal assignment for 1Count24V.

View	Terminal Assignment	Remarks
		B: Direction input or track B A: Pulse input or track A 24V DC: Sensor supply M: Chassis ground DI: Digital input DO1: Digital output

Pulse generator connection

The following table shows the pulse generator connections to the terminal module and the possible count directions.

Encoder Type	Connection	Count Direction
Pulse generator without direction indicator	24 V count pulses at terminal 5 (A)	Up
Pulse generator with direction indicator	24 V count pulses at terminal 5 (A) and 24 V direction at terminal 1 (B)	Up, down
Pulse generator with 2 tracks that are 90° out of phase	Track A terminal 5 (A) and track B terminal 1 (B)	Up, down

Configuring and assigning parameters to 1Count24V

Operation

Selecting an operating mode

Operating mode

You define 1Count24V functionality by setting an operating mode.

Operating mode	Description
Counting (Page 9221)	Preselection for counting modes
Measuring (Page 9245)	Preselection for measuring modes
Position feedback (Page 9262)	In this mode, 1Count24V counts continuously from the load value: • If another count pulse is received after 1Count24V reaches the high count limit when counting up, it will jump to the low count limit and continue counting from there without losing a pulse. • If another count pulse is received after 1Count24V reaches the low count limit when counting down, it will jump to the high count limit and continue counting from there without losing a pulse.
Fast mode (Page 9274) (supported as of firmware version V2.0)	In this mode, 1Count24V counts continuously from the initial value: • If another count pulse is received after 1Count24V reaches the maximum value that can be represented by 25 bits when counting up (all counter bits are set), the count value will jump to "0" and resume counting from there without losing a pulse. • If 1Count24V reaches the value "0" when counting down and another count pulse is then received, the count value will jump to the maximum value that can be represented by 25 bits (all counter bits set) and resume counting from there without losing a pulse. This operating mode can be used for position feedback in particularly short isochrone cycles.

Counting parameters

Enabling group diagnostics

Group diagnostics

Channel-specific diagnostics will be carried out if you enable group diagnostics in your parameter assignment.

Selecting reaction to CPU/Master STOP

Setting response to CPU/Master STOP

You can configure the response of the 1Count24V to failure of the higher-level controller.

Parameter	Status of the 1Count24V at CPU/Master STOP	What happens when new parameters are assigned?
Turn off DO1	The current mode is cancelled. The gate is closed and the digital output disabled. Comparison values 1 and 2 the load value are reset. The high and low limit, function and behavior of the digital outputs and the integration time are transferred from the parameters.	The changed parameters are accepted and become effective.
Continue working mode [1]	The current mode will continue working. The gate and digital output status will not change.	The gate is closed. The current mode is cancelled. The digital output is disabled. The changed parameters are accepted and become effective.
DO 1 substitute a value	The current mode is cancelled. The gate is closed and the substitute value set for the digital output is activated. Comparison values 1 and 2 and the load value are reset. The high and low limit, function and behavior of the digital outputs and the integration time are transferred from the parameters. If "Pulse when comparison value is reached" has been set for the output, the only available pulse duration substitute value will be 1.	The changed parameters are accepted and become effective.
DO 1 keep last value	The current mode is cancelled. The gate is closed and the status of the digital output is maintained. Comparison values 1 and 2 and the load value are reset. The high and low limit, function and behavior of the digital outputs and the integration time are transferred from the parameters.	The changed parameters are accepted and become effective.

[1] The CPU/Master must not clear the outputs if the mode is to continue working upon transition from CPU/Master STOP to RUN (startup).
Possible solution: In the part of the user program that is executed during startup, set the SW_GATE control bit and transfer the values to 1Count24V.

Leaving the Assigned State

Under what conditions does the 1Count24V leave the assigned state?

The CPU or master must be in RUN mode and you must make a change at the control interface (Page 9241).

Automatic New Parameter Assignment

New parameters are assigned to the ET 200S station by your CPU/ DP master:

- Upon power on of the CPU/DP master
- Upon IM 151-1 / IM 151-1 FO power on
- After failure of the DP transmission
- Once altered parameters or configuration of the ET 200S station have been loaded to the CPU/DP master
- When the 1Count24V is plugged
- Upon power on or inserting of the appropriate power module

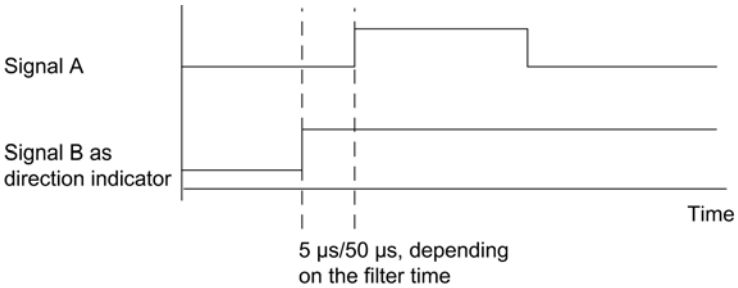
Signal evaluation

Selecting count and direction evaluation

Signal evaluation A*, B*

Signal evaluation by means of A*, B* allows you to count directionally. Different evaluation modes are possible depending on the parameter assignment:

- Pulse and Direction (Page 9224)
  In the case of 24 V pulse generators with a direction level, there must be an interval of at least 5 µs/50 µs (depending on the input filter set) between the direction signal (B*) and the count signal (A*).
  The following figure shows the interval between direction signal and count signal.



- Rotary Encoder (Page 9108)
  If you connect a 24 V rotary encoder with two tracks that are 90 degrees out of phase at the count and direction inputs, you can set single evaluation (Page 9108) in all counting modes. You can alternatively set dual (Page 9108) or quadruple evaluation (Page 9109). You can invert the count direction (Page 9225) at direction input B* in all evaluation modes. A range of different sensors can be connected to the encoder inputs (Sourcing output/push pull or Sinking output).

Note

If you have selected the "Sinking output" setting with 1Count24V for the "Encoder inputs" parameter, you must use the M-switching sensors.

Note

The count frequency of 100 kHz is the maximum frequency of the A* and B* signals. The maximum frequency for count pulses is therefore 200 kHz in double evaluation and 400 kHz in quadruple evaluation.

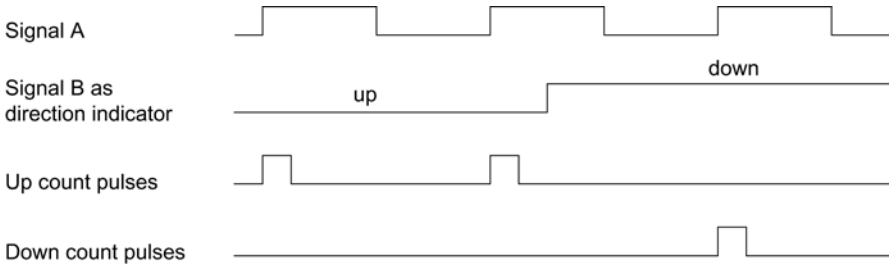
"Pulse and direction" evaluation

Pulse and direction

The level at direction input B* is used as the direction setting in this type of evaluation.

An unwired input corresponds to the "Up" count direction if you have selected "Pulse and direction" for the "Signal evaluation" parameter.

The diagram below shows the signals of a 24 V pulse generator with direction level.



Selecting filters for input signals

Input filter

You can activate a filter to filter signals at count input A*, direction input B* and digital input DI in line with the minimum pulse duration or the maximum signal frequency.

You can select a minimum pulse width of 2.5 µs or 25 µs for each input.

Selecting encoder inputs

Encoder inputs

A range of different sensors can be connected to the encoder inputs (Sourcing output/push pull or Sinking output).

Note

If you have selected the "Sinking output" setting with for the "Encoder inputs" parameter, you must use the M-switching sensors.

Selecting the count direction

Count direction

To adapt the count direction to the process, you can select:

- Normal count direction:
- Inverted count direction

Behavior of the outputs in counting modes

Comparison values and outputs

Introduction

The 1Count24V allows you to store two comparison values, which are assigned to the digital outputs. The outputs can be activated, depending on the counter status and the comparison values.

Digital outputs

The 1Count24V has a real digital output and a virtual digital output that exists only as a status bit in the feedback interface.

Parameters can be assigned for both outputs ("Function DO1" and "Function DO2" parameters).

You can change the function and the behavior of the digital outputs during operation. The new function takes effect immediately.

Controlling the outputs and comparators simultaneously

You can continue to control the outputs using SET_DO1 or SET_DO2 if you have selected a compare function for the outputs. This allows you to simulate the effect of the compare functions in your control program:

- The output is set with the positive edge of SET_DO1 or SET_DO2.
  However, if the "Pulse on reaching the comparison value" function is selected, only one pulse with the specified duration will be output. SET_DO1 and SET_DO2 have no effect when pulse duration = 0.
  The SET_DO1 control bit cannot be used with output behavior "Switch at comparison values".
- A SET_DO1 or SET_DO2 negative edge resets the output.

Please note that the comparators are still active and can set or reset the output if the comparison result changes.

Note

An output set by SET_DO1 or SET_DO2 cannot be reset by the comparator.

Loading comparison values

You transfer the comparison values to the 1Count24V. This does not affect the count.

Valid range for the two comparison values

Main count direction: None	Main count direction: Up	Main count direction: Down
Low count limit to high count limit	–2147483648 to high count limit -1	1 to 2147483647

Changing the function and behavior of digital outputs

You can change the functions and behavior of the outputs during operation using the control interface (Page 9241). The 1Count24V clears the outputs and accepts the values as follows:

- **Function of digital outputs DO1 and DO2:** If you change this function so that the comparison condition is satisfied, the output will not be set until after the next count pulse. However, 1Count24V will not make any changes at the output if hysteresis (Page 9231) is enabled.
- **Hysteresis:** Active hysteresis (Page 9231) remains active after a change. The new hysteresis range will be applied the next time the comparison value is reached.
- **Pulse duration:** The new pulse duration (Page 9234) will take effect with the next pulse.

Selecting output behavior

Behavior of outputs DO1 and DO2

Select the behavior of the output in question in the drop down list:

- Output (Page 9228)
- On when counter value ≥ comparison value (Page 9228)
- On when counter value ≤ comparison value (Page 9228)
- Pulse on reaching the comparison value (Page 9228)
- Switch at comparison values (Page 9229) (DO1 only)

Setting or Modifying the Function and Behavior of the Digital Output DO1

When setting or modifying the behavior of DO1, you must take all assignable interdependencies into account. Failure to do so will generate a parameter assignment error or a loading error.

Boundary conditions:

If you assign "Switch at comparison values" for DO1, you must:

- Set hysteresis = 0, and
- also configure "DO2 function" as "Output".

Output

Output

You can switch the outputs on and off with the control bits (Page 9241)SET_DO1 and SET_DO2.

The control bits (Page 9241)CTRL_DO1 or CTRL_DO2 must be set.

You can query the status of the outputs with the status bits STS_DO1 and STS_DO2 in the feedback interface (Page 9243).

The status bits (Page 9243)STS_CMP1 and STS_CMP2 indicate that the relevant output is or was switched on. These status bits retain their status until they are acknowledged. If the output is still switched, the corresponding bit is set again immediately. These status bits are also set when the control bit SET_DO1 or SET_DO2 is operated without DO1 or DO2 being enabled.

Isochrone mode

Output DO1 is switched at time T_o in isochrone mode. The status of the virtual output DO2 is signaled at time T_i.

On when counter value ≤ comparison value or when counter value ≥ comparison value

Counter status ≤ Comparison Value and Counter Status ≥ Comparison Value

If the comparison conditions are fulfilled, the respective comparator switches on the output. The status of the output (Page 9243) is indicated by STS_DO1 and STS_DO2.

The control bits (Page 9241)CTRL_DO1 or CTRL_DO2 must be set.

The result of the comparison is indicated by the status bits (Page 9243)STS_CMP1 or STS_CMP2. You cannot acknowledge and thus reset these bits until the comparison conditions are no longer fulfilled.

Isochronous mode

In isochrone mode, too, output DO1 is switched as soon as the comparison condition is fulfilled and is therefore independent of the bus cycle. The status of the virtual output DO2 is reported at time T_i.

Pulse on reaching the comparison value

Pulse on reaching the comparison value

If the counter status reaches the comparison value, the comparator switches on the respective digital output for the assigned pulse duration.

The control bit (Page 9241)CTRL_DO1 or CTRL_DO2 must be set.

The status of the status bits (Page 9243)STS_DO1 and STS_DO2 is always the same as that of the corresponding digital output.

The comparison result is indicated by the status bit (Page 9243)STS_CMP1 or STS_CMP2 and cannot be reset by acknowledgment until the pulse duration has elapsed.

If a main count direction is assigned, the comparator switches only when the comparison value in the main count direction is reached.

If a main count direction is not assigned, the comparator switches when the comparison value is reached from either direction.

If the digital output is set by control bit (Page 9241)SET_DO1 or SET_DO2, it will be reset when the pulse duration has elapsed.

Isochronous mode

In isochrone mode, too, output DO1 is switched as soon as the comparison condition is fulfilled and is therefore independent of the bus cycle. The status of virtual output DO2 is reported at time T_i.

Switch at comparison values

Switch at comparison values

The comparator switches the output when the following conditions are met:

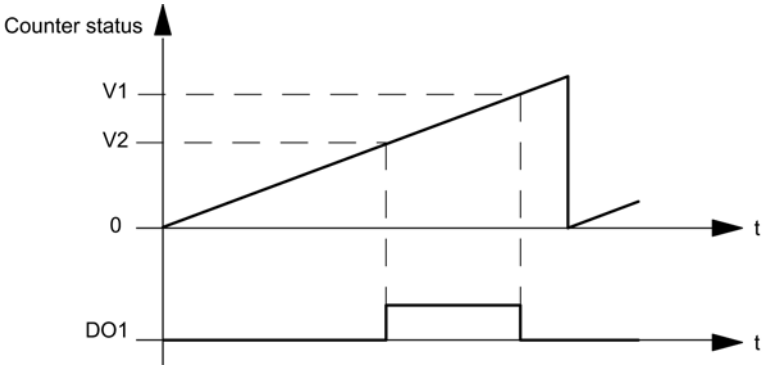
- The two comparison values must be loaded with the load functions (Page 9241)CMP_VAL1 and CMP_VAL2 and
- The output DO1 (Page 9241) must be enabled with CRTL_DO1 after the comparison values are loaded.

The following table shows you when the DO1 is switched on or off:

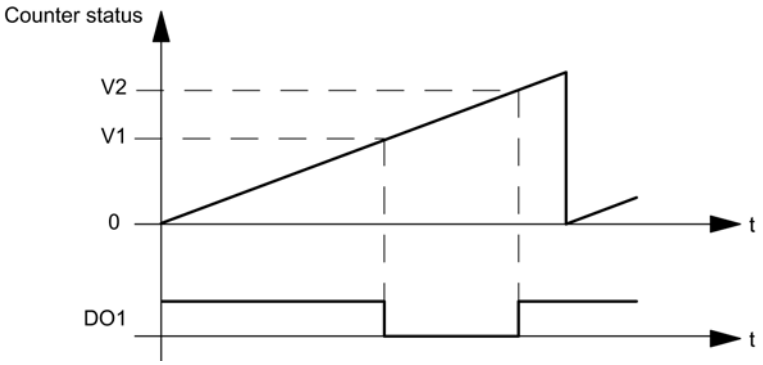
	DO1 is switched on when	DO1 Is switched off when
V2 < V1 (see Figure below)	V2 ≤ counter status ≤ V1	V2 > counter status or counter status > V1
V2 = V1	V2 = counter status = V1	V2 ≠ counter status ≠ V1
V2 > V1 (see Figure below)	V1 > counter status or counter status > V2	V1 ≤ counter status ≤ V2

The figures below show the status of output DO1 as determined by the counter value and comparison values:

- V2 < V1 at the start of counting



- V2 > V1 at the start of counting



The result of the comparison is shown by the status bit (Page 9243)STS_CMP1. You can only acknowledge and thus reset this bit when the comparison condition is no longer fulfilled.

There is no hysteresis in the case of this output behavior.

It is not possible to control the DO1 output with control bit (Page 9241)SET_DO1 if this is the output behavior.

Isochronous mode

In isochrone mode, too, output DO1 is switched as soon as the comparison condition is fulfilled and is therefore independent of the bus cycle.

Selecting substitute value DO1

Substitute value DO1

This is where you select the substitute value which is to be output at digital output DO1 if you have selected the parameter "DO1substitute a value" under "Reaction to CPU/Master STOP (Page 9222)".

Activating DO1 diagnostics

DO1 diagnostics

"DO1 diagnostics" (wire break, short circuit) are possible only with pulse lengths of > 90 ms at digital output DO1.

Select the check box for diagnostics at output DO1.

Setting hysteresis

Hysteresis

An encoder can remain at a particular position and then fluctuate around this position. This state causes the counter status to fluctuate around a particular value. If there is a comparison value in this fluctuation range, for example, the associated output is switched on and off in accordance with the rhythm of the fluctuations. The 1Count24V is equipped with programmable hysteresis to prevent this switching in response to small fluctuations. You can assign a range between 0 and 255 (0 means: hysteresis switched off).

Hysteresis also works with high limit violated and low limit violated.

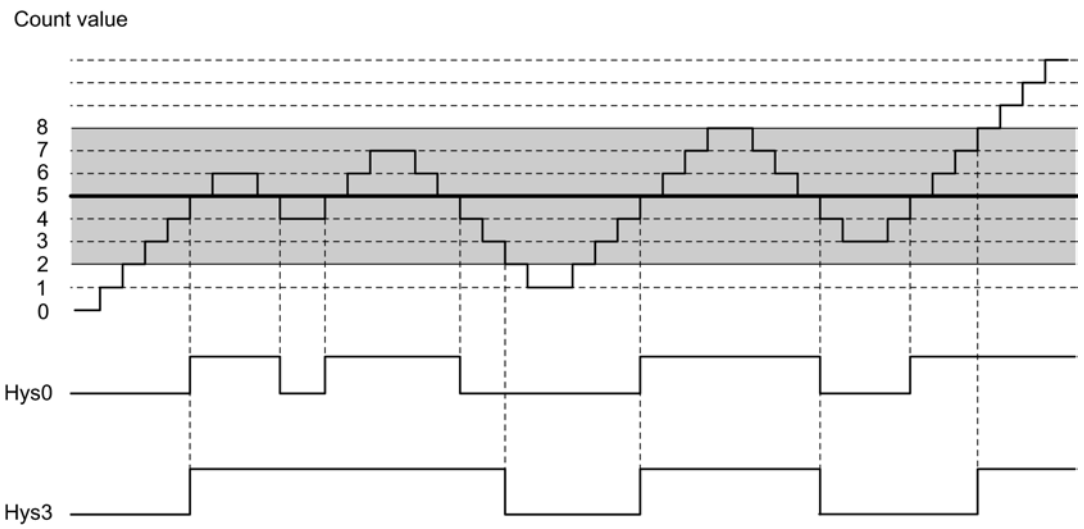
Method of Operation with Counter Status ≤ Comparison Value and Counter Status ≥ Comparison Value

The example in the figure below demonstrates the effect of hysteresis. The figure shows the differences in the behavior of an output when hysteresis of 0 (= switched off) is parameterized as opposed to hysteresis of 3. In the example, the comparison value is 5.

The following settings are assigned for the counter: "Main count direction" = "Up" and "Turn on at counter status≥ comparison value".

When the comparison condition is met, hysteresis becomes active. While the hysteresis is active, the comparison result remains unchanged.

Hysteresis will no longer be active if the counter value is outside the hysteresis range. The comparator switches again according to its comparison conditions.



Note

If the count direction changes on the comparison value when hysteresis is active, the output is reset.

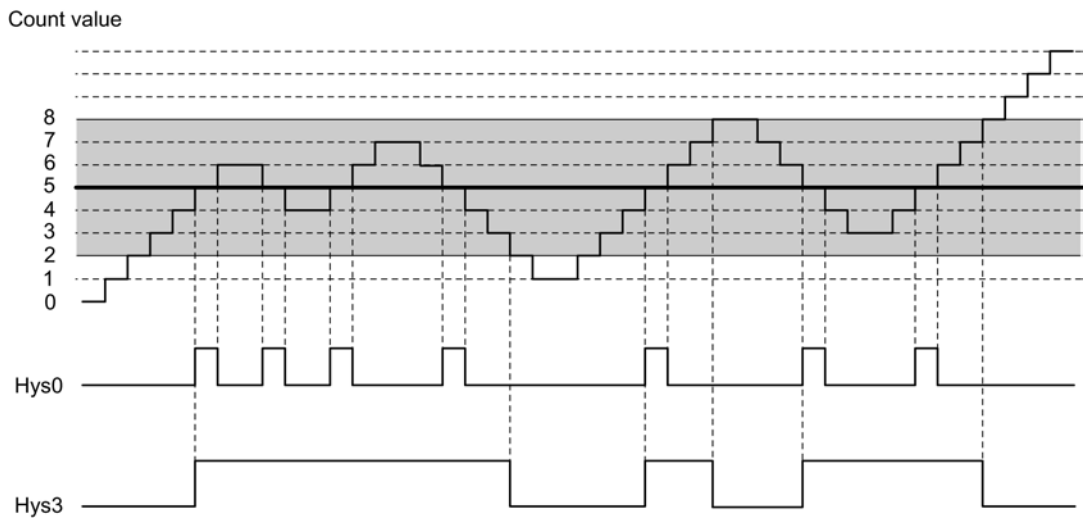
Method of Operation when the Comparison Value Is Reached and the Pulse Duration = 0

The example in the figure below demonstrates the effect of hysteresis. The figure shows the various responses of an output with a hysteresis of 0 (= switched off) as opposed to hysteresis of 3. In the example, the comparison value is 5.

The following settings are assigned for the counter: "Pulse on reaching the comparison value", "No main count direction" and "Pulse duration = 0".

When the comparison conditions are met, hysteresis becomes active. While the hysteresis is active, the comparison result remains unchanged.

If the counter value goes outside the hysteresis range, hysteresis is no longer active. The comparator deletes the result of the comparison.



Method of Operation when the Comparison Value Is Reached, Output Pulse Duration

The example in the figure below demonstrates the effect of hysteresis. The figure shows the various responses of an output with a hysteresis of 0 (= switched off) as opposed to hysteresis of 3. In the example, the comparison value is 5.

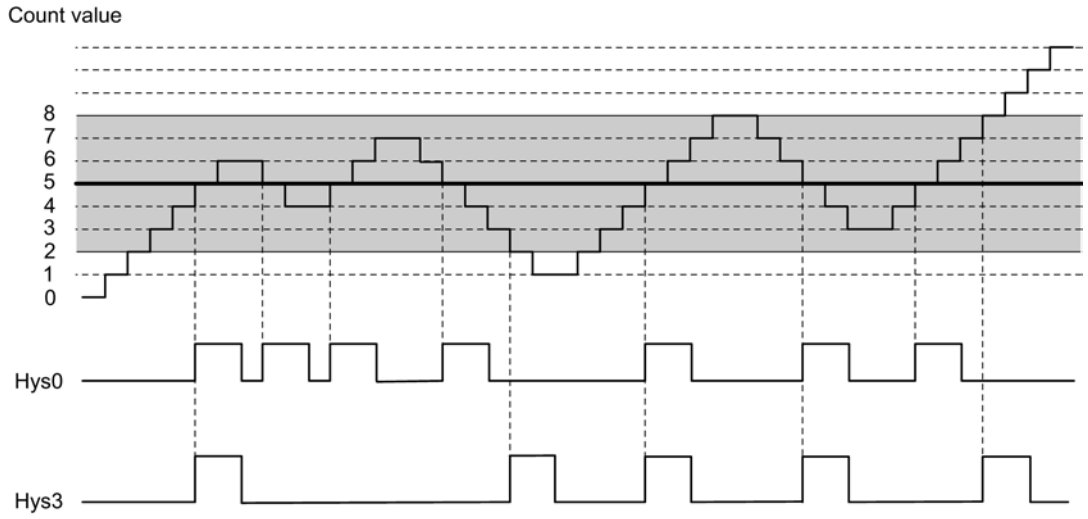
The following settings are assigned for the counter: "Pulse on reaching the comparison value", "No main count direction" and "pulse duration > 0".

When the comparison conditions have been met, hysteresis becomes active and a pulse of the assigned duration is output.

If the counter value goes outside the hysteresis range, hysteresis is no longer active.

When hysteresis is enabled, the 1Count24V stores the count direction.

If the hysteresis range is exited in a different direction to the one stored, a pulse is output.



Specifying pulse duration

Pulse duration

The pulse duration begins when the respective digital output is set. The pulse duration is accurate to at least 2 ms.

The pulse duration can be set to suit the actuators used. The pulse duration specifies how long the output is to be set for. The pulse duration can be preselected between 0 ms and 510 ms in increments of 2 ms.

If the pulse duration = 0, the output is set until the comparison condition is no longer fulfilled. Note that the count pulse times must be greater than the minimum switching times of the digital output.

Isochronous mode

In isochrone mode, too, output DO1 is switched as soon as the comparison condition is fulfilled and is therefore independent of the bus cycle. The status of virtual output DO2 is reported at time T_i.

Selecting counting mode

Description

You define 1Count24V functionality by setting a counting mode.

Operating mode	Description
Count continuously (Page 9083) (with or without gate)	1Count24V counts continuously from the current counter value.
Count once (Page 9084) (with hardware or software gate)	When the gate opens, 1Count24V counts from the load value to the count limit.
Count periodically (Page 9085) (with hardware or software gate)	When the gate opens, 1Count24V counts from the load value between the count limits.

The default setting is the "Count continuously" operating mode.

Selecting main count direction

Description

By configuring a main count direction (Page 9086) (up or down), you can reduce the maximum counting range by setting an high count limit (Page 9240). The parameterized counting range is then between 0 and the high count limit parameterized.

A main count direction can only be selected in the "Count once" and " Count periodically" modes.

You can select the following behavior:

- None (no main count direction)
- Up
- Down

Selecting the gate function

Description

When configuring the hardware and software gates, you can specify whether the internal gate is to cancel or interrupt counting.

- Cancel counting (Page 9103): The counting process restarts from the beginning after gate stop and restart.
- Interrupt counting (Page 9103): The counting process starts from the last current counter value after gate stop and restart.

DI function

Selecting DI function

Description

You select the digital input function with "DI function":

- Input
  The digital input operates as a "normal" input.
- Hardware gate (Page 9102), level-controlled
  The hardware gate is opened and closed by signals at the digital input.
- Latch/Retrigger with positive edge (Page 9236)
  Counter values are saved (latched) with positive edges at the digital input. The counter is set to the load value each time the values are saved.
- Synchronization with positive edge (Page 9237)
  The counter is set to the load value with positive edges at the digital input.
- Latching with positive edge (Page 9239)
  Counter values are saved (latched) with positive edges at the digital input.

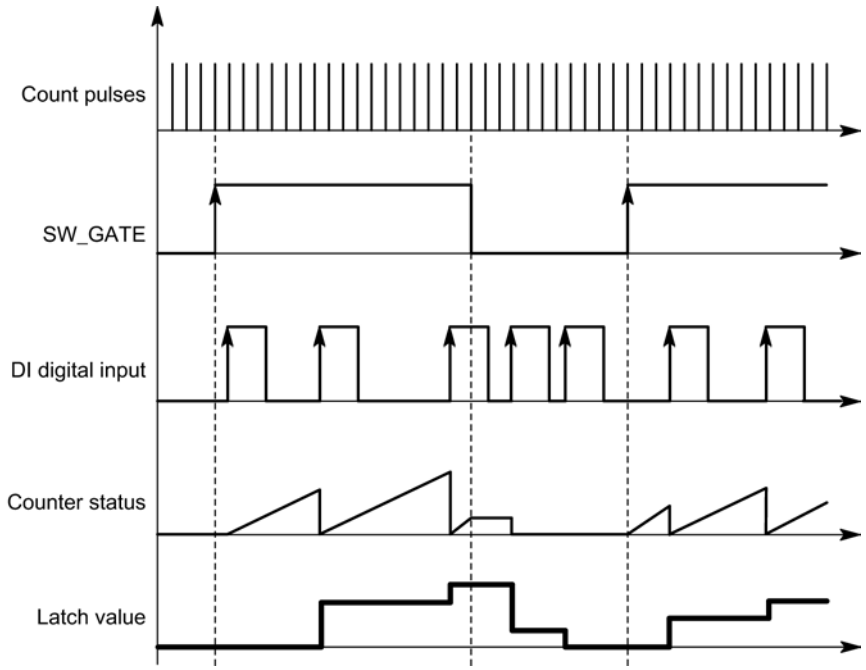
The default setting is "Input".

Latch/Retrigger with positive edge

Latch/Retrigger

This function stores the current internal counter status of the 1Count24V and retriggers counting when there is a positive edge on the digital input. This means that the current internal counter status at the time of the positive edge is stored (latch value), and the 1Count24V is then loaded again with the load value, from which counting resumes.

The following figure shows latch/retrigger when load value = 0.



The counting mode must be enabled with the SW gate before the function can be executed. It is started with the first positive edge on the digital input.

The saved rather than the current counter value is displayed in the feedback interface (Page 9243). The STS_DI bit indicates the status of the latch and retrigger signal.

The latch value is pre-assigned its RESET status:

- "0" for main count direction "None" and "Up"
- "High count limit parameters assigned" for main count direction "Down"

It is not changed when the SW gate is opened.

Direct loading of the counter does not cause the indicated stored counter status to be changed.

If you close the SW gate, counting is only interrupted; this means that when you open the SW gate again, counting is continued. The digital input DI remains active even when the SW gate is closed.

Counting is also latched and triggered in isochronous mode with each edge on the digital input. The counter status that was valid at the time of the last edge before T_i is displayed in the feedback interface.

Extended feedback interface

If the 1Count24V is inserted behind an IM 151 that supports the reading and writing of broader user data interfaces, the current counter value can be read from bytes 8 to 11 of the feedback interface (Page 9243).

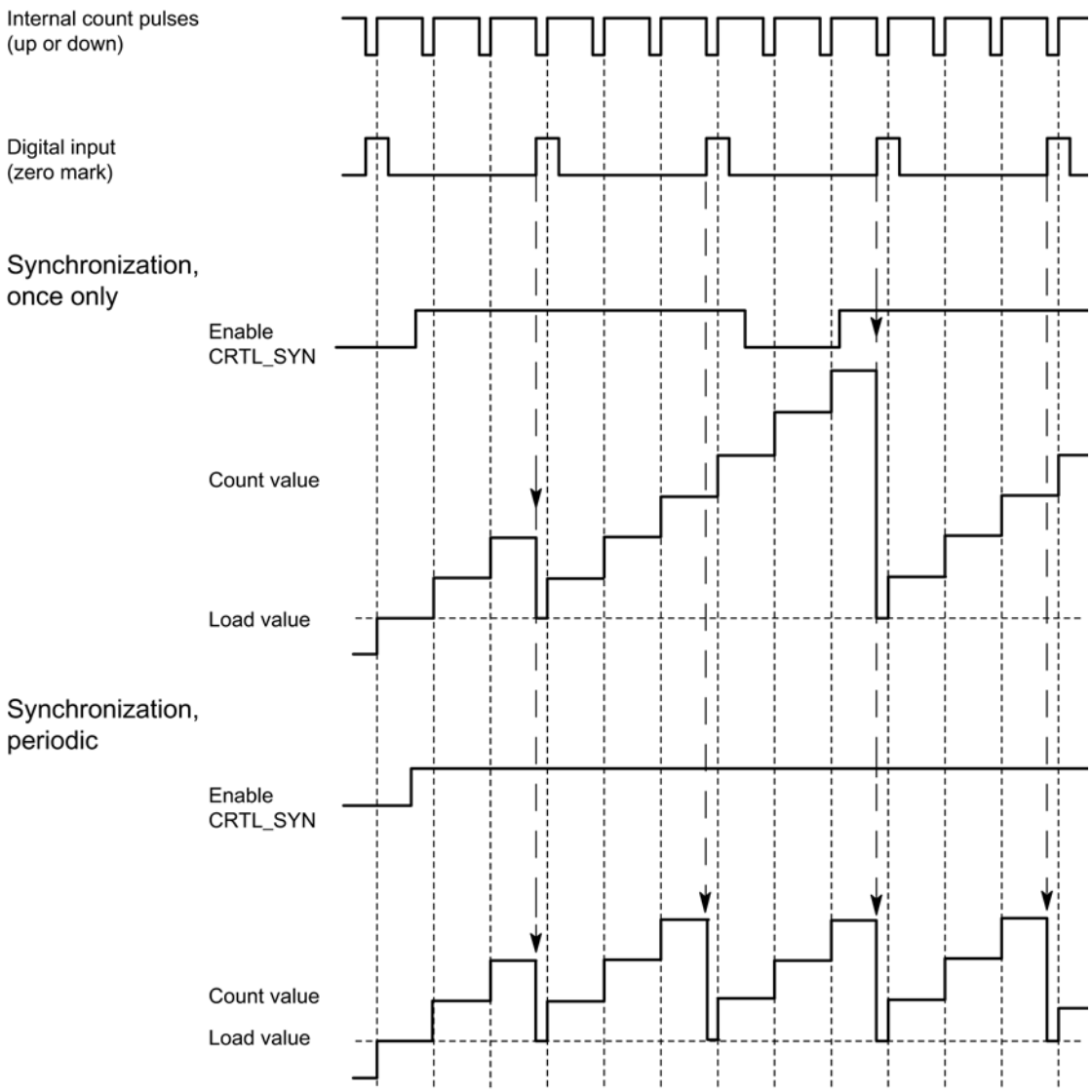
Synchronization with positive edge

Synchronization

If you have configured "Synchronization with positive edge", the rising edge of a reference signal at the input will set 1Count24V to the load value.

You can select between once-only and periodic synchronization ("Synchronization" parameter).

The figure below shows the counter values in once-only and periodic synchronization.



The following conditions apply:

- The counting mode must have been started with the SW gate.
- The "Enable synchronization" (CTRL_SYN) control bit (Page 9241) must be set.
- Synchronize once only: If the enable bit is set, the first edge loads the 1Count24V with the load value.
- Synchronization periodically: If the enable bit is set, the first and each subsequent edge loads the 1Count24V with the load value.
- The feedback bit (Page 9243)STS_SYN is set after successful synchronization. It must be reset by the control bit (Page 9241)RES_STS.
- The signal of a bounce-free switch or the zero mark of a rotary encoder can serve as the reference signal.
- The feedback bit (Page 9243)STS_DI indicates the level of the reference signal.

In isochrone mode, the set feedback bit (Page 9243)STS_SYN indicates that the rising edge at the digital input was between time T_i in the current cycle and time T_i in the previous cycle.

Latching with positive edge

Latching

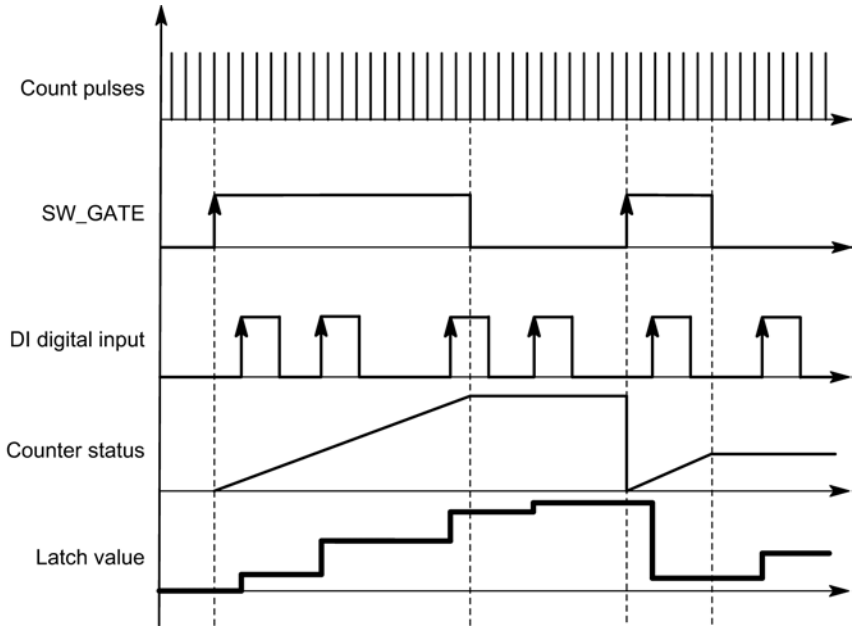
Counter value and latch value are pre-assigned their RESET statuses:

- "0" for main count direction "None" and "Up"
- "High count limit parameters assigned" for main count direction "Down"

The counting function is started when the SW gate is opened. 1Count24V begins at the load value:

- "0" for main count direction "None" and "Up"
- "High count limit parameters assigned" for main count direction "Down"

The following figure shows latching when load value = 0.



The latch value is always the exact counter status at the time of the positive edge on the digital input DI.

The saved rather than the current counter value is displayed in the feedback interface (Page 9243). The STS_DI bit indicates the level of the latch signal.

Direct loading of the counter does not cause the indicated stored counter status to be changed.

In isochronous mode, the counter status that was latched at the time of the last positive edge before T_i is displayed in the feedback interface.

When you close the SW gate, the effect is either canceling or interrupting, depending on the parameter assignment. The digital input DI remains active even when the software gate is closed.

Extended feedback interface

If the 1Count24V is inserted behind an IM 151 that supports the reading and writing of broader user data interfaces, the current counter value can be read from bytes 8 to 11 of the feedback interface (Page 9243).

Selecting hardware gate input signal

Hardware gate input signal

The level of the digital input for the "Function DI = HW gate" configuration with the parameter "Input signal HW gate" can be inverted.

The feedback bit (Page 9243)STS_DI indicates the level of the digital input.

Selecting synchronization

Synchronization

If you have selected "Synchronization with positive edge" at the "DI function (Page 9235)" parameter, this is where you can select either once-only or periodic synchronization:

- Once-only: If the enable bit is set, the first edge loads the 1Count24V with the load value.
- Periodic: If the enable bit is set, the first and each subsequent edge loads 1Count24V with the load value.

Specifying the high count limit

Description

By configuring an high count limit together with a main count direction (Page 9086) (up or down), you can reduce the maximum counting range. The parameterized counting range is then between 0 and the high count limit parameterized.

The high count limit can be any value between > 1 and the high count range limit.

Control and feedback interface for counting modes

Control interface for counting modes

Note

The following control interface data is consistent for 1Count24V:

- Bytes 0 ... 3
- Bytes 4 ... 7
- Bytes 8 ... 11 (extended user data interface)

Control interface assignment

The table below shows the assignment of the 1Count24V control interface (outputs) for counting modes.

Address		Assignment			
Bytes 0 to 3		**Load value direct, preparatory, comparison value 1 or 2**			
	Byte 0	**Behavior of DO1, DO2 of 1Count24V**			
		Bit 2	Bit 1	Bit 0	Function DO1
		0	0	0	Output
		0	0	1	Turn on at counter status ≥ comparison value
		0	1	0	Turn on at counter status ≤ comparison value
		0	1	1	Pulse on reaching the comparison value
		1	0	0	Switch at comparison values
		1	0	1	blocked
		1	1	0	blocked
		1	1	1	blocked
			Bit 5	Bit 4	Function DO2
			0	0	Output
			0	1	Turn on when counter status ≥ comparison value
			1	0	Turn on when counter status ≤ comparison value
			1	1	Pulse on reaching the comparison value
		Bits 3, 6, and 7: Reserved = 0			
	Bytes 1 to 3	Byte 1	Hysteresis DO1, DO2 (value range 0 to 255)		
		Byte 2	Pulse duration DO1, DO2 (value range 0 to 255)		
		Byte 3	Reserved = 0		
					Name
Byte 4		Bit 7	Error acknowledgment		EXTF_ACK
		Bit 6	Enable DO2		CTRL_DO2
		Bit 5	Control bit DO2		SET_DO2
		Bit 4	Enable DO1		CTRL_DO1
		Bit 3	Control bit DO1		SET_DO1

Address	Assignment		
	Bit 2	Trigger status bits reset	RES_STS
	Bit 1	Enable synchronization	CTRL_SYN
	Bit 0	Software gate control bit	SW_GATE
Byte 5	Bit 7	Reserved = 0	
	Bit 6	Reserved = 0	
	Bit 5	Reserved = 0	
	Bit 4	Change function and behavior of DO1, DO2	C_DOPARAM
	Bit 3	Load comparison value 2	CMP_VAL2
	Bit 2	Load comparison value 1	CMP_VAL1
	Bit 1	Load counter in preparation	LOAD_PREPARE
	Bit 0	Load counter direct	LOAD_VAL
Bytes 6 to 7	Reserved = 0 [1]		

[1] Not used for extended user interface

Notes on the Control Bits

Bit	Explanation
C_DOPARAM	Change function and behavior of DO1, DO2 The values from bytes 0 to 2 are applied as new function, hysteresis, and pulse duration of DO1, DO2. This may result in the following error: The conditions for the "Switch at comparison values" behavior are not fulfilled.
CMP_VAL1	Load comparison value 1 The value from bytes 0 to 3 is transferred to comparison value 1 with the control bit "Load comparison value 1" (CMP_VAL1).
CMP_VAL2	Load comparison value 2 The value from bytes 0 to 3 is transferred to comparison value 2 with the control bit "Load comparison value 2" (CMP_VAL2).
CTRL_DO1	Enable DO1 You use this bit to enable the DO1 output.
CTRL_DO2	Enable DO2 You use this bit to enable the DO2 output.
CTRL_SYN	You use this bit to enable synchronization.
EXTF_ACK	Error acknowledgment The error bits must be acknowledged with the EXTF_ACK control bit after the cause has been eliminated.
LOAD_PREPARE	Load counter in preparation The value from bytes 0 to 3 is applied as the load value.
LOAD_VAL	The value from bytes 0 to 3 is loaded directly as the new counter value.
RES_STS	Trigger status bits reset The status bits are reset by the acknowledgment process between the RES_STS bit and the RES_STS_A bit.
SET_DO1	Control bit DO1 Switches the DO1 digital output on and off when CRTL_DO1 is set.

Bit	Explanation
SET_DO2	Control bit DO2 Switches the DO2 digital output on and off when CRTL_DO2 is set.
SW_GATE	Software gate control bit The software gate is opened/closed via the control interface with the SW_GATE bit.

Feedback interface for counting modes

Note

The following feedback interface data is consistent for 1Count24V:

- Bytes 0 ... 3
- Bytes 4 ... 7
- Bytes 8 ... 11 (extended user data interface)

Feedback interface assignment

The table below shows the assignment of the 1Count24V feedback interface (inputs) for counting modes.

Address	Assignment		Name
Bytes 0 to 3	Count value or stored counter value in the case of the latch function on the digital input		
Byte 4	Bit 7	Encoder supply short circuit	ERR_24V
	Bit 6	Short circuit / wire break / overtemperature	ERR_DO1
	Bit 5	Parameter assignment error	ERR_PARA
	Bit 4	Reserved = 0	
	Bit 3	Reserved = 0	
	Bit 2	Resetting of status bits active	RES_STS_A
	Bit 1	Load function error	ERR_LOAD
	Bit 0	Load function is running	STS_LOAD
Byte 5	Bit 7	Status down	STS_C_DN
	Bit 6	Status up	STS_C_UP
	Bit 5	Reserved = 0	
	Bit 4	DO2 status	STS_DO2
	Bit 3	DO1 status	STS_DO1
	Bit 2	Reserved = 0	
	Bit 1	DI status	STS_DI
	Bit 0	Internal gate status	STS_GATE
Byte 6	Bit 7	Zero-crossing in the count range when counting without a main count direction	STS_ND
	Bit 6	Low count limit	STS_UFLW
	Bit 5	High count limit	STS_OFLW
	Bit 4	Comparator 2 status	STS_CMP2
	Bit 3	Comparator 1 status	STS_CMP1

Address	Assignment		Name
	Bit 2	Reserved = 0	
	Bit 1	Reserved = 0	
	Bit 0	Synchronization status	STS_SYN
Byte 7	Reserved = 0		
Bytes 8 to 11	Counter value [1]		

[1] Extended user data interface

Notes on the Feedback Bits

Bit	Explanation
ERR_24V	Encoder supply short circuit The error bit must be acknowledged with the control bit (Page 9241)EXTF_ACK Diagnostic alarm if parameterized.
ERR_DO1	Short circuit/wire break/overtemperature due to overload at output DO1 The error bit must be acknowledged with the control bit (Page 9241)EXTF_ACK Diagnostic alarm if parameterized.
ERR_LOAD	Load function error The LOAD_VAL, LOAD_PREPARE, CMP_VAL1, CMP_VAL2 and C_DOPARAMcontrol bits (Page 9241) cannot be set simultaneously during transfer. As when you load an incorrect value (which is not accepted), this results in the ERR_LOAD status bit being set.
ERR_PARA	Errors in assigned module parameters.
RES_STS_A	Resetting of status bits active
STS_C_DN	Status down
STS_C_UP	Status up
STS_CMP1	Comparator 1 status The STS_CMP1 status bit indicates that the output is or was switched on. It must be acknowledged with the control bit (Page 9241)RES_STS. If the status bit is acknowledged when the output is still switched on, the bit is set again immediately. This bit is also set if the SET_DO1 control bit is used when DO1 is not enabled.
STS_CMP2	Comparator 2 status The STS_CMP2 status bit indicates that the output is or was switched on. It must be acknowledged with the control bit (Page 9241)RES_STS. If the status bit is acknowledged when the output is still switched on, the bit is set again immediately. This bit is also set if the SET_DO2 control bit is operated when DO2 is not enabled.
STS_DI	DI status The status of the DI is indicated in all modes by the STS_DI bit in the feedback interface.
STS_DO1	DO1 status The STS_DO1 status bit indicates the status of the DO1 digital output.
STS_DO2	DO2 status The STS_DO2 status bit indicates the status of the virtual DO2 digital output.
STS_GATE	Internal gate status: Counting.
STS_LOAD	Load function is running
STS_ND	Zero-crossing in the count range when counting without a main count direction. The bit must be reset by the control bit (Page 9241)RES_STS.

Bit	Explanation
STS_OFLW STS_UFLW	High count limit violated Low count limit violated Both bits must be reset.
STS_SYN	Synchronization status The STS_SYN bit is set after successful synchronization. It must be reset by the control bit (Page 9241)RES_STS.

Measurement parameters

Enabling group diagnostics

Group diagnostics

Channel-specific diagnostics will be carried out if you enable group diagnostics in your parameter assignment.

Selecting reaction to CPU/Master STOP

Setting response to CPU/Master STOP

You can configure the response of the 1Count24V to failure of the higher-level controller.

Parameter	Status of the 1Count24V at CPU/Master STOP	What happens when new parameters are assigned?
Turn off DO1	The current mode is cancelled. The gate is closed and the digital output disabled. Comparison values 1 and 2 the load value are reset. The high and low limit, function and behavior of the digital outputs and the integration time are transferred from the parameters.	The changed parameters are accepted and become effective.
Continue working mode [1)]	The current mode will continue working. The gate and digital output status will not change.	The gate is closed. The current mode is cancelled. The digital output is disabled. The changed parameters are accepted and become effective.

Parameter	Status of the 1Count24V at CPU/Master STOP	What happens when new parameters are assigned?
DO 1 substitute a value	The current mode is cancelled. The gate is closed and the substitute value set for the digital output is activated. Comparison values 1 and 2 and the load value are reset. The high and low limit, function and behavior of the digital outputs and the integration time are transferred from the parameters. If "Pulse when comparison value is reached" has been set for the output, the only available pulse duration substitute value will be 1.	The changed parameters are accepted and become effective.
DO 1 keep last value	The current mode is cancelled. The gate is closed and the status of the digital output is maintained. Comparison values 1 and 2 and the load value are reset. The high and low limit, function and behavior of the digital outputs and the integration time are transferred from the parameters.	The changed parameters are accepted and become effective.

1) The CPU/Master must not clear the outputs if the mode is to continue working upon transition from CPU/Master STOP to RUN (startup).
Possible solution: In the part of the user program that is executed during startup, set the SW_GATE control bit and transfer the values to 1Count24V.

Leaving the Assigned State

Under what conditions does the 1Count24V leave the assigned state?

The CPU or master must be in RUN mode and you must make a change at the control interface (Page 9259).

Automatic New Parameter Assignment

New parameters are assigned to the ET 200S station by your CPU/ DP master:

- Upon power on of the CPU/DP master
- Upon IM 151-1 / IM 151-1 FO power on
- After failure of the DP transmission
- Once altered parameters or configuration of the ET 200S station have been loaded to the CPU/DP master
- When the 1Count24V is plugged
- Upon power on or inserting of the appropriate power module

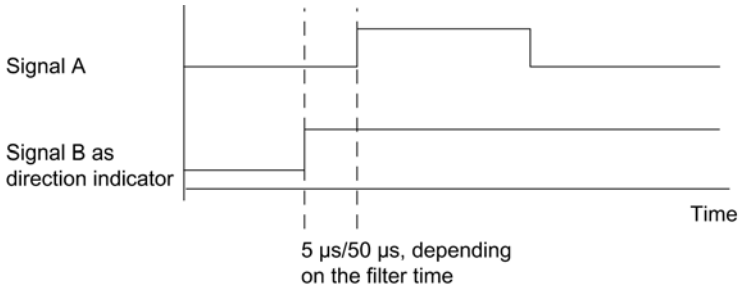
Signal evaluation

Selecting count and direction evaluation

Signal evaluation A*, B*

Signal evaluation by means of A*, B* allows direction-specific counting. Different evaluation modes are possible depending on the parameter assignment:

- Pulse and Direction (Page 9247)
  In the case of 24 V pulse generators with a direction level, there must be an interval of at least 5 µs/50 µs (depending on the input filter set) between the direction signal (B*) and the count signal (A*).
  The following figure shows the interval between direction signal and count signal.



- Rotary Encoder
  If you connect a 24 V rotary encoder with two tracks that are 90 degrees out of phase at the count and direction inputs, you can configure single evaluation (Page 9108) in all measuring modes.
  You can invert the count direction (Page 9253) at direction input B*.
  A range of different sensors can be connected to the encoder inputs (Sourcing output/push pull or Sinking output).

Note

If you have selected the "Sinking output" setting with 1Count24V for the "Encoder inputs" parameter, you must use the M-switching sensors.

Note

The count frequency of 100 kHz is the maximum frequency of the A* and B* signals.

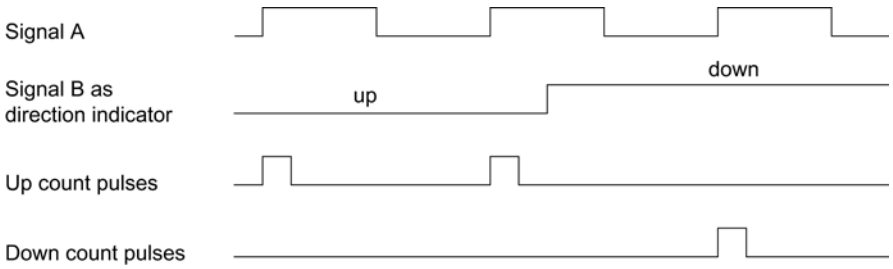
"Pulse and direction" evaluation

Pulse and direction

The level at direction input B* is used as the direction setting in this type of evaluation.

An unwired input corresponds to the "Up" count direction if you have selected "Pulse and direction" for the "Signal evaluation" parameter.

The diagram below shows the signals of a 24 V pulse generator with direction level.



Measuring methods

Overview

Measuring methods

You can choose from the following measuring methods for the "Measuring procedure" parameter:

- With integration time (Page 9248)
- Continuous (Page 9249)

Measurement with integration time

Measurement with integration time

The measurement is carried out during the assigned integration time. When the integration time elapses, the measured value is updated.

The end of a measurement is indicated by the STS_CMP1status bit (Page 9260). This bit is reset by the control bit (Page 9259)RES_STS in the control interface.

If there were not at least two rising edges in the assigned integration time, 0 is returned as the measured value.

A value of -1 is returned up until the end of the first integration time.

You can change the integration time for the next measurement during operation.

Direction Reversal

If the direction of rotation is reversed during integration time, the measured value for this measurement period will be undefined. Evaluating the STS_C_UP and STS_C_DNfeedback bits (Page 9260) (direction evaluation) allows you to respond to any process irregularities.

See also

Integration Time and Update Time in Isochronous Mode (Page 9252)

Continuous measurement

Measuring Principle

The 1Count24V counts each positive edge of a pulse and assigns it a time value in µs.

The update time indicates the time interval at which the measured value is updated by the module in the feedback interface.

Rule for pulse sequences with one or several pulses per update time:

Dynamic measuring time =Time of last pulse in the current update time interval
minus
Time of last pulse in the previous update time interval

When the update time has elapsed, a new measured value is calculated and output with the dynamic measuring time.

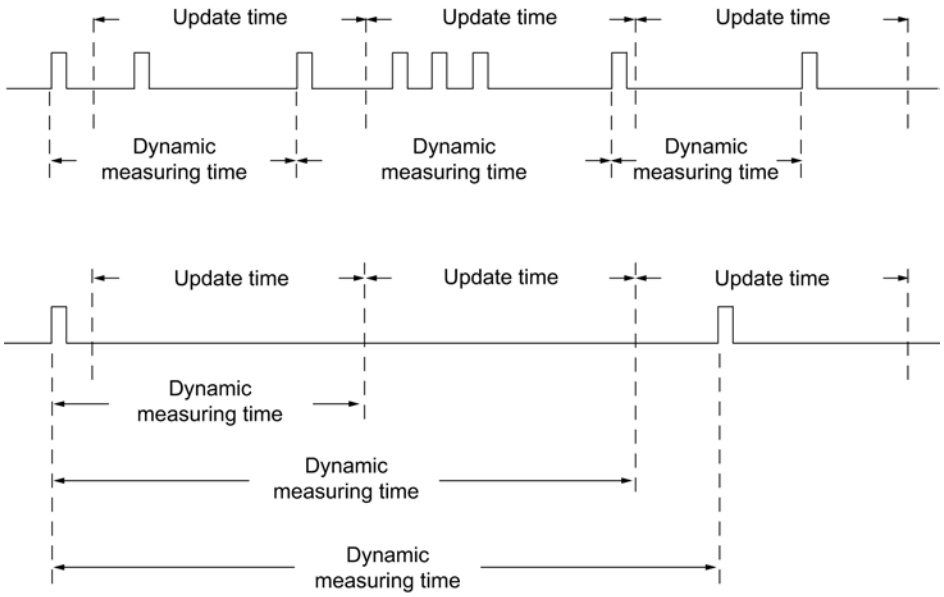
The following dynamic measuring time will result if the current update time does not contain a pulse:

Dynamic measuring time =Time of current, elapsed update time
minus
Time of last pulse

When the update time has elapsed, an estimated measured value is calculated with the dynamic measuring time under the assumption that a pulse occurred at the end of the update time.

If the "1 Pulse per dynamic measuring time" estimated measured value is less than the last measured value during the frequency and speed measurement, this estimated measured value is output as the new measured value. With the period measurement, the dynamic measuring time is output as the estimated period if the dynamic measuring time is greater than the last measured period.

The figure below shows the measuring principle.



The 1Count24V measures continuously. When assigning parameters, you specify an update time.

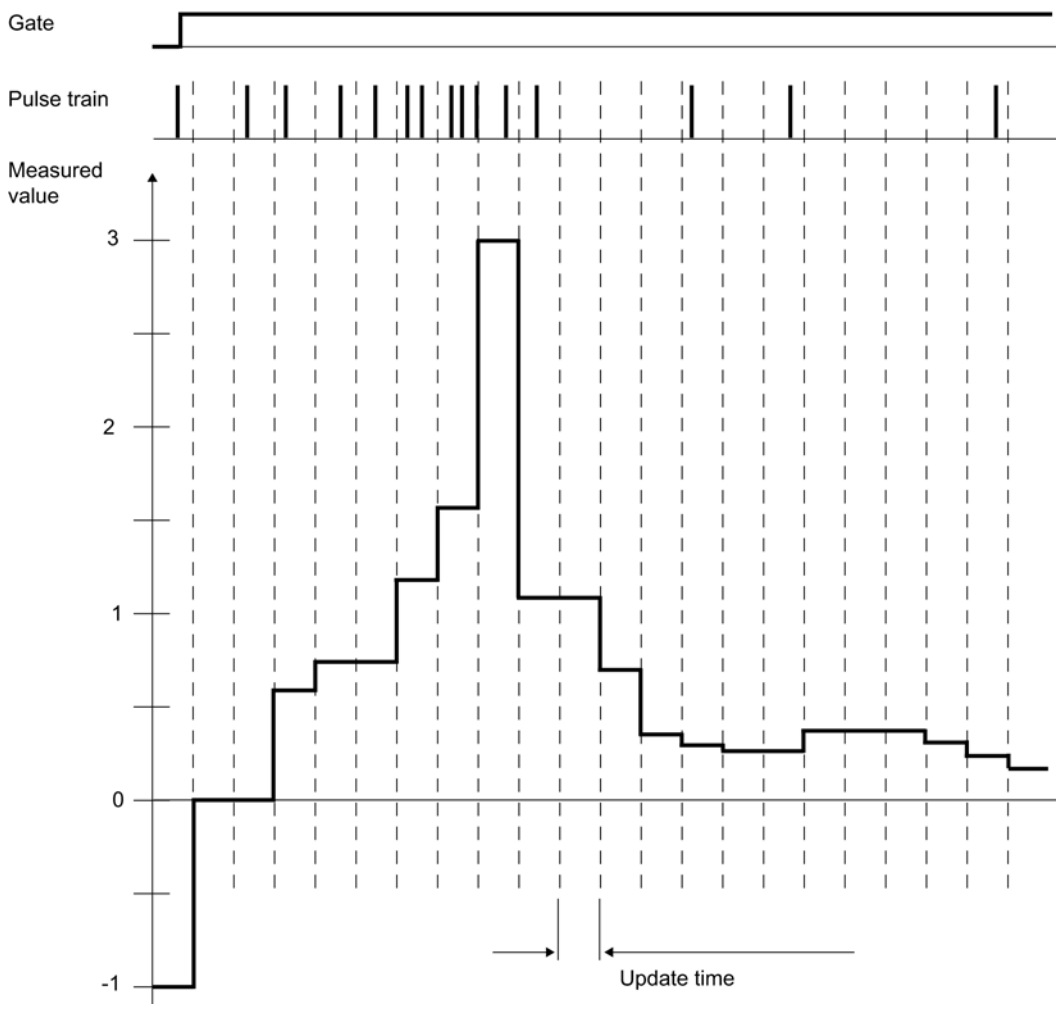
During the time until the end of the first elapsed update time, a value of "-1" is returned.

The continuous measurement begins after the gate is opened with the first pulse of the pulse train to be measured. The first measured value can be calculated after the second pulse, at the earliest.

A measured value (frequency, period, or speed) is output in the feedback interface each time the update time elapses. The end of a measurement is indicated by the STS_CMP1status bit (Page 9260). This bit is reset with the RES_STS and RES_STS_A bits by applying the complete acknowledgement principle.

If the direction of rotation is reversed during an update time, the measured value for this measurement period will be undefined. Evaluating the STS_C_DN and STS_C_UPfeedback bits (Page 9260) (direction evaluation) enables you to respond to any process irregularities.

The following figure illustrates the principle of continuous measurement taking frequency measurement as an example.



Gate Control

To control the 1Count24V, you have to use the gate functions.

Isochronous mode

In isochronous mode, the 1Count24V accepts control bits and control values from the control interface in each bus cycle and reports back the response in the same cycle.

In each cycle, the 1Count24V transfers a measured value and the status bits that were valid at time T_i.

The measurement starts and ends at time T_i.

See also

Integration Time and Update Time in Isochronous Mode (Page 9252)

Integration Time and Update Time in Isochronous Mode

Integration time and update time in isochrone mode

If the integration time/update time lasts several T_{DP} cycles, the new measured value will be indicated in the user program by the STS_CMP1status bit (Page 9260) (measurement completed) of the feedback interface. This enables monitoring of the measuring operation or a synchronization with the measuring operation. It takes 4 T_{DP} cycles, however, for this alarm to be acknowledged. The minimum integration time/update time in this case is (4 x T_{DP}).

You do not need to continually evaluate status bit STS_CMP1 if the application can tolerate a jitter in the integration time/update time of T_{DP} and a measured value that remains constant for several cycles. Integration times/update times of (1 x T_{DP}) to (3 x T_{DP}) are then possible.

A loss of isochrone operation in the last T_{DP} cycle of the integration time will increase the integration time/update time by one T_{DP} cycle. This does not corrupt the measured value.

Note

The value range limits for the integration time/update time may not be exceeded (see tables for the individual measuring modes).

Violation of the value range limits will result in a parameter assignment error and 1Count24V will not go into isochrone mode.

Note

You must always adjust the integration time/update time parameter if you want to retain the integration time/update time length when you change the configuration from non-isochrone to isochrone mode and vice versa.

Selecting filters for input signals

Input filter

You can activate a filter to filter signals at count input A*, direction input B* and digital input DI in line with the minimum pulse duration or the maximum signal frequency.

You can select a minimum pulse width of 2.5 µs or 25 µs for each input.

Selecting encoder inputs

Encoder inputs

A range of different sensors can be connected to the encoder inputs (Sourcing output/push pull or Sinking output).

Note

If you have selected the "Sinking output" setting with for the "Encoder inputs" parameter, you must use the M-switching sensors.

Selecting the count direction

Count direction

To adapt the count direction to the process, you can select:

- Normal count direction:
- Inverted count direction

Behavior of the output in measuring modes

Selecting behavior of output DO1

Behavior of the Output in Measuring Modes

You can assign parameters for the digital output of the 1Count24V.

You can store a high and a low limit for frequency measurement, rotational speed measurement or period measurement. If the limits are violated, digital output DO1 is activated. These limit values can be assigned and changed with the load function.

You can change the function and the behavior of the digital output during operation. The new function takes effect immediately. In isochronous mode it always takes effect at time T_i.

You can choose between the following functions in the drop-down list box.

- Output (Page 9254)
- Outside limits (Page 9254) (limit monitoring)
- Violating low limit (Page 9254) (limit monitoring)
- Exceeding high limit (Page 9254) (limit monitoring)

Output

Output

If you want to switch the output on or off, you must enable it with the CTRL_DO1control bit (Page 9259).

You can switch the output on and off with the SET_DO1control bit (Page 9259).

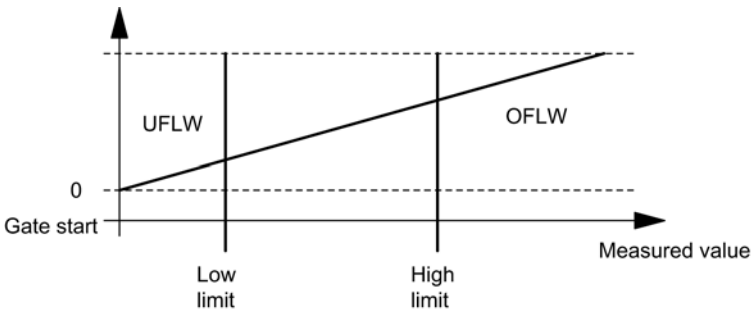
You can query the status of the output with the STS_DO1status bit (Page 9260) in the feedback interface.

In isochronous mode, the output is switched at time T_o.

Limit-Value Monitoring

Limit monitoring

The figure below illustrates limit monitoring.



When the integration time has elapsed, the measured value obtained (frequency, rotational speed or period) is compared with the assigned limits.

The STS_UFLW = 1 bit will be set in the feedback (Page 9260) interface if the current measured value is below the parameterized low limit (measured value < low limit).

The STS_OFLW = 1 bit will be set in the feedback (Page 9260) interface if the current measured value is above the parameterized high limit (measured value > high limit).

You must acknowledge these bits with the control bit (Page 9259)RES_STS.

The corresponding status bit will be set again if the measured value still or once again violates the limits after acknowledgment.

Setting the low limit to 0 switches off dynamic monitoring of "Violating low limit".

The table below shows how the enabled digital output DO1 can be set by limit monitoring in accordance with parameter assignment:

"Function DO1" parameter	DO1 is Set ...
Outside limits	Measured value < low limit OR measured value > high limit
Violating low limit	Measured value < low limit
Exceeding high limit	Measured value > high limit

In isochronous mode the output is switched at the end of measurement at time T_i.

Selecting substitute value DO1

Substitute value DO1

This is where you select the substitute value which is to be output at digital output DO1 if you have selected the parameter "DO1substitute a value" under "Reaction to CPU/Master STOP (Page 9245)".

Activating DO1 diagnostics

DO1 diagnostics

"DO1 diagnostics" (wire break, short circuit) are possible only with pulse lengths of > 90 ms at digital output DO1.

Select the check box for diagnostics at output DO1.

Selecting a measuring mode

Description

You define 1Count24V functionality by specifying a measuring mode.

	Description	
Operating mode	Measurement with integration time	Continuous measurement
Frequency measurement (Page 9093) (with hardware or software gate)	1Count24V counts the pulses that occur during a specified integration time.	1Count24V counts the pulses that occur during a dynamic measurement time.
Rotational speed measurement (Page 9094) (with hardware or software gate)	1Count24V counts the pulses received from a tachometer generator within a specified integration time and calculates the speed on the basis of this value and the number of pulses per encoder revolution.	1Count24V counts the pulses received from a tachometer generator within a dynamic measuring time and calculates the speed on the basis of this value and the number of pulses per encoder revolution.
Period measurement (Page 9096) (with hardware or software gate)	1Count24V measures the time between two positive edges of the count signal by counting the pulses of an internal quartz accurate reference frequency (16 MHz) over a specified integration time.	1Count24V indicates the dynamic measuring time as a period. The mean value of the period is calculated if its duration is less than the update time.

The default setting is "Frequency measurement" mode.

Selecting period resolution

Description

The value of the calculated period is displayed in 1 µs or 1/16 µs.

Select the required unit of resolution.

Possible Measuring Ranges with Error Indication

- Period measurement with integration time
  - Measuring ranges and errors at a resolution of 1 µs:

Integration time	T_{min} ± absolute error	T_{min} ± absolute error
100 s	1 µs* (10 ± 0)	1 µs* (100,000,000 ± 10,000)
10 s	1 µs* (10 ± 0)	1 µs* (10,000,000 ± 1,000)
1 s	1 µs* (10 ± 0)	1 µs* (1,000,000 ± 100)
0.1 s	1 µs* (10 ± 0)	1 µs* (100,000 ± 10)
0.01 s	1 µs* (10 ± 0)	1 µs* (10,000 ± 1)

  - Measuring ranges and errors at a resolution of 1/16 µs:

Integration time	T_{min} ± absolute error	T_{min} ± absolute error
100 s	1/16 µs* (160 ± 0)	1/16 µs* (1,600,000,000 ± 160,000)
10 s	1/16 µs* (160 ± 0)	1/16 µs* (160,000,000 ± 16,000)
1 s	1/16 µs* (160 ± 0)	1/16 µs* (16,000,000 ± 1,600)
0.1 s	1/16 µs* (160 ± 0)	1/16 µs* (1,600,000 ± 160)
0.01 s	1/16 µs* (160 ± 0)	1/16 µs* (160,000 ± 16)

- Continuous period measurement:
  - Measuring ranges and errors at a resolution of 1 µs:

Period T_{min} ± Absolute error	Period T_{min} ± Absolute error
1 µs* (10 ± 0)	1 µs* (100,000 ± 10)
1 µs* (100 ± 0)	1 µs* (1 000 000 ± 100)
1 µs* (1 000 ± 0)	1 µs* (10 000 000 ± 1 002)
1 µs* (10 000 ± 1)	1 µs* (100,000,000 ± 10,020)

  - Measuring ranges and errors at a resolution of 1/16 µs:

Period T_{min} ± Absolute error	Period T_{min} ± Absolute error
1/16 µs* (160 ± 1)	1/16 µs* (1,600,000 ± 160)
1/16 µs* (1,600 ± 1)	1/16 µs* (16,000,000 ± 1,600)
1/16 µs* (16 000 ± 3)	1/16 µs* (160 000 000 ± 16 000)
1/16 µs* (160 000 ± 20)	1/16 µs* (1,600,000,000 ± 160,000)

Selecting DI function

Description

You select the digital input function with "DI function":

- Input
  The digital input operates as a "normal" input.
- Hardware gate (Page 9102), level-controlled
  The hardware gate is opened and closed by signals at the digital input.

The default setting is "Input".

Selecting hardware gate input signal

Hardware gate input signal

The level of the digital input for the "Function DI = HW gate" configuration with the parameter "Input signal HW gate" can be inverted.

The feedback bit (Page 9260)STS_DI indicates the level of the digital input.

Setting low and high limit

Low and high limit

You can save a high and low limit for frequency measurement, speed measurement and period measurement. If these limits are exceeded, digital output DO1 (Page 9253) will be activated. You can configure these limits and change them using the load function.

The following value ranges are possible for the low and high limits:

Measuring mode	Low limit (LL)	High limit
Frequency measurement	0 to 99,999,999 × 10^{-3} Hz	LL+1 to 100,000,000 × 10^{-3} Hz
Rotational speed measurement	0 to 24,999,999 × 10^{-3} /min	LL+1 to 25,000,000 × 10^{-3} /min
Period measurement (resolution = 1 µs)	0 to 119,999,999 µs	LL+1 to 120,000,000 µs
Period measurement (resolution = 1/16 µs)	0 to 1,919,999,999 µs	LL+1 to 1,920,000,000 µs

Specifying measuring time

Description

If you have selected a measuring mode, the module will update the measured values on a cyclic basis. The update time/integration time is the product of measuring time x 10ms. You set the measuring time as the factor n.

Detailed descriptions of the value ranges for the measuring time can be found under "Measuring basics (Page 9088)".

Entering pulses per encoder revolution

Pulses per encoder revolution

For "Rotational speed measurement (Page 9094)" mode, you must also configure the pulses per encoder revolution.

The value for the number of pulses per encoder revolution can be found on the rating plate, or in the technical data of your encoder.

Control and feedback interface for measuring modes

Control interface for measuring modes

Note

The following control interface data is consistent for 1Count24V:

- Bytes 0 ... 3
- Bytes 4 ... 7
- Bytes 8 ... 11 (extended user data interface)

Control interface assignment

The table below shows the assignment of the 1Count24V control interface (outputs) for measuring modes.

Address	Assignment				
Bytes 0 to 3	**Low or high limit**				
	Function of DO1				
	Byte 0	Bit 1	Bit 0	Function DO1	
		0	0	Output	
		0	1	Outside limits	
		1	0	Violating low limit	
		1	1	Exceeding high limit	
	Bytes 1 to 3	Reserved = 0			
	Integration time				
	Bytes 0, 1	Integration time [n × 10 ms] (Value range 1 to 1000/12000)			
	Bytes 2, 3	Reserved = 0			
					Name
Byte 4	Bit 7	Error acknowledgment			EXTF_ACK
	Bit 6	Reserved = 0			
	Bit 5	Reserved = 0			
	Bit 4	Enable DO1			CTRL_DO1
	Bit 3	Control bit DO1			SET_DO1
	Bit 2	Trigger status bits reset			RES_STS
	Bit 1	Reserved = 0			
	Bit 0	Software gate control bit			SW_GATE
Byte 5	Bit 7	Reserved = 0			
	Bit 6	Reserved = 0			

Address	Assignment		
	Bit 5	Reserved = 0	
	Bit 4	Change function of DO1	C_DOPARAM
	Bit 3	Reserved = 0	
	Bit 2	Change integration time	C_INTTIME
	Bit 1	Load high limit	LOAD_PREPARE
	Bit 0	Load low limit	LOAD_VAL
Bytes 6 to 7	Reserved = 0 [1]		

[1] Not used for extended user interface

Notes on the control bits

Bit	Explanation
C_DOPARAM	Change function of DO1 The value from byte 0 is adopted as the new function of DO1.
C_INTTIME	Change integration time The value from bytes 0 and 1 is adopted as the new integration time for the next measurement.
CTRL_DO1	Enable DO1 You use this bit to enable the DO1 output.
EXTF_ACK	Error acknowledgment The error bits must be acknowledged with the EXTF_ACK control bit after the cause has been eliminated.
LOAD_PREPARE	Load high limit The value from bytes 0 to 3 is applied as the new high limit.
LOAD_VAL	Load low limit The value from bytes 0 to 3 is applied as the new low limit.
RES_STS	Trigger status bits reset The status bits are reset by the acknowledgment process between the RES_STS bit and the RES_STS_A bit.
SET_DO1	Control bit DO1 Switches the DO1 digital output on and off when CRTL_DO1 is set.
SW_GATE	Software gate control bit The software gate is opened/closed via the control interface with the SW_GATE bit.

Feedback interface for measuring modes

Note

The following feedback interface data is consistent for 1Count24V:

- Bytes 0 ... 3
- Bytes 4 ... 7
- Bytes 8 ... 11 (extended user data interface)

Feedback interface assignment

The table below shows the assignment of the 1Count24V feedback interface (inputs) for measuring modes.

Address	Assignment		Name
Bytes 0 to 3	Measured value		
Byte 4	Bit 7	Encoder supply short circuit	ERR_24V
	Bit 6	Short circuit / wire break / overtemperature	ERR_DO1
	Bit 5	Parameter assignment error	ERR_PARA
	Bit 4	Reserved = 0	
	Bit 3	Reserved = 0	
	Bit 2	Resetting of status bits active	RES_STS_A
	Bit 1	Load function error	ERR_LOAD
	Bit 0	Load function is running	STS_LOAD
Byte 5	Bit 7	Status down	STS_C_DN
	Bit 6	Status up	STS_C_UP
	Bit 5	Reserved = 0	
	Bit 4	Reserved = 0	
	Bit 3	DO1 status	STS_DO1
	Bit 2	Reserved = 0	
	Bit 1	DI status	STS_DI
	Bit 0	Internal gate status	STS_GATE
Byte 6	Bit 7	Reserved = 0	
	Bit 6	Low limit, measuring range	STS_UFLW
	Bit 5	High limit, measuring range	STS_OFLW
	Bit 4	Reserved = 0	
	Bit 3	Measurement completed	STS_CMP1
	Bit 2	Reserved = 0	
	Bit 1	Reserved = 0	
	Bit 0	Reserved = 0	
Byte 7	Reserve = 0		
Bytes 8 to 11	Counter value [1]		

[1] Extended user data interface

Notes on the feedback bits

Bit	Explanation
ERR_24V	Encoder supply short circuit The error bit must be acknowledged with the control bit (Page 9259)EXTF_ACK. Diagnostic alarm if parameterized.
ERR_DO1	Short circuit/wire break/overtemperature at output DO1 The error bit must be acknowledged with the control bit (Page 9259)EXTF_ACK. Diagnostic alarm if parameterized.

Bit	Explanation
ERR_LOAD	Load function error The LOAD_VAL, LOAD_PREPARE, C_DOPARAM and C_INTTIMEcontrol bits (Page 9259) cannot be set simultaneously during transfer. As when you load an incorrect value (which is not accepted), this results in the ERR_LOAD status bit being set.
ERR_PARA	Errors in assigned module parameters.
RES_STS_A	Resetting of status bits active
STS_C_DN	Status down
STS_C_UP	Status up
STS_CMP1	Measurement completed The measured value is updated each time a time interval (update time/integration time) elapses. **Measurement with integration time:** The end of a measurement (after the interval has elapsed) is indicated by the STS_CMP1 status bit. **Continuous measurement:** The end of the measurement is signaled by the status bit STS_CMP1 at the end of the update time if a measured value is output. The bit remains at "0" if an estimated measured value is output. This bit is reset by the control bit (Page 9259)RES_STS in the control interface.
STS_DI	DI status The status of the DI is indicated in all modes by the STS_DI bit in the feedback interface.
STS_DO1	DO1 status
STS_GATE	Internal gate status: Measuring.
STS_LOAD	Load function is running
STS_OFLW STS_UFLW	High limit violated Low limit violated Both bits must be reset.

Position feedback parameters

Enabling group diagnostics

Group diagnostics

Channel-specific diagnostics will be carried out if you enable group diagnostics in your parameter assignment.

Selecting reaction to CPU/Master STOP

Setting response to CPU/Master STOP

You can configure the response of the 1Count24V to failure of the higher-level controller.

Parameter	Status of the 1Count24V at CPU/Master STOP	What happens when new parameters are assigned?
Turn off DO1	The current mode is cancelled. The gate is closed and the digital output disabled. Comparison values 1 and 2 the load value are reset. The high and low limit, function and behavior of the digital outputs and the integration time are transferred from the parameters.	The changed parameters are accepted and become effective.
Continue working mode [1]	The current mode continues working. The gate and digital output status does not change.	The gate is closed. The current mode is cancelled. The digital output is disabled. The changed parameters are accepted and become effective.

[1] The CPU/Master must not clear the outputs if the mode is to continue working on transition from CPU/Master STOP to RUN (startup).
Possible solution: In the part of the user program that is executed during startup, set the SW_GATE control bit and transfer the values to 1Count24V.

Exiting parameterized status

Under what conditions does the 1Count24V exit the parameterized status?

The CPU or master must be in RUN mode and you must make a change at the control interface (Page 9270).

Automatic new parameter assignment

New parameters are assigned to the ET 200S station by your CPU/ DP master:

- Upon CPU/DP master power on
- Upon IM 151-1 / IM 151-1 FO power on
- After failure of DP transmission
- Once altered parameters or configuration of the ET 200S station have been loaded to the CPU/DP master
- When the 1Count24V is inserted
- Upon power on or insertion of the corresponding power module

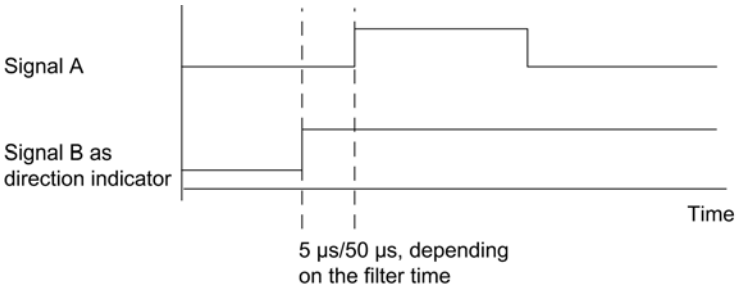
Signal evaluation

Selecting count and direction evaluation

Signal evaluation A*, B*

Signal evaluation by means of A*, B* allows you to count directionally. Different evaluation modes are possible depending on the parameter assignment:

- Pulse and Direction (Page 9264)
  In the case of 24 V pulse generators with a direction level, there must be an interval of at least 5 µs/50 µs (depending on the input filter set) between the direction signal (B*) and the count signal (A*).
  The following figure shows the interval between direction signal and count signal.



- Rotary Encoder (Page 9108)
  If you connect a 24 V rotary encoder with two tracks that are 90 degrees out of phase at the count and direction inputs, you can set single evaluation (Page 9108) in all counting modes. You can alternatively set dual (Page 9108) or quadruple evaluation (Page 9109). You can invert the count direction (Page 9265) at direction input B* in all evaluation modes. A range of different sensors can be connected to the encoder inputs (Sourcing output/push pull or Sinking output).

Note

If you have selected the "Sinking output" setting with 1Count24V for the "Encoder inputs" parameter, you must use the M-switching sensors.

Note

The count frequency of 100 kHz is the maximum frequency of the A* and B* signals. The maximum frequency for count pulses is therefore 200 kHz in double evaluation and 400 kHz in quadruple evaluation.

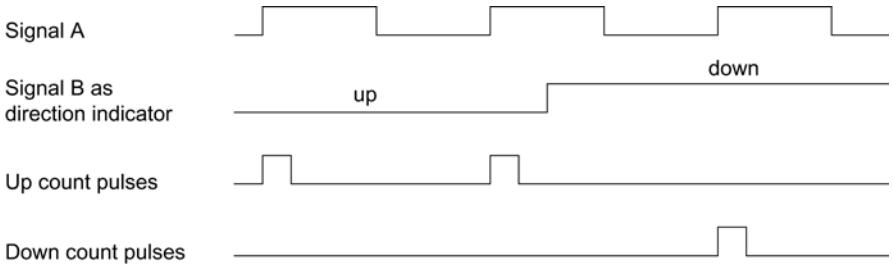
"Pulse and direction" evaluation

Pulse and direction

The level at direction input B* is used as the direction setting in this type of evaluation.

An unwired input corresponds to the "Up" count direction if you have selected "Pulse and direction" for the "Signal evaluation" parameter.

The diagram below shows the signals of a 24 V pulse generator with direction level.



Selecting filters for input signals

Input filter

You can activate a filter to filter signals at count input A*, direction input B* and digital input DI in line with the minimum pulse duration or the maximum signal frequency.

You can select a minimum pulse width of 2.5 µs or 25 µs for each input.

Selecting encoder inputs

Encoder inputs

A range of different sensors can be connected to the encoder inputs (Sourcing output/push pull or Sinking output).

Note

If you have selected the "Sinking output" setting with for the "Encoder inputs" parameter, you must use the M-switching sensors.

Selecting the count direction

Count direction

To adapt the count direction to the process, you can select:

- Normal count direction:
- Inverted count direction

Selecting the gate function

Description

When configuring the hardware and software gates, you can specify whether the internal gate is to cancel or interrupt counting.

- Cancel counting (Page 9103): The counting process restarts from the beginning after gate stop and restart.
- Interrupt counting (Page 9103): The counting process starts from the last current counter value after gate stop and restart.

DI function

Selecting DI function

Description

You select the digital input function with "DI function":

- Input
  The digital input operates as a "normal" input.
- Hardware gate (Page 9102), level-controlled
  The hardware gate is opened and closed by signals at the digital input.
- Latch/Retrigger with positive edge (Page 9266)
  Counter values are saved (latched) with positive edges at the digital input. The counter is set to the load value each time the values are saved.
- Synchronization with positive edge (Page 9267)
  The counter is set to the load value with positive edges at the digital input.
- Latching with positive edge (Page 9269)
  Counter values are saved (latched) with positive edges at the digital input.

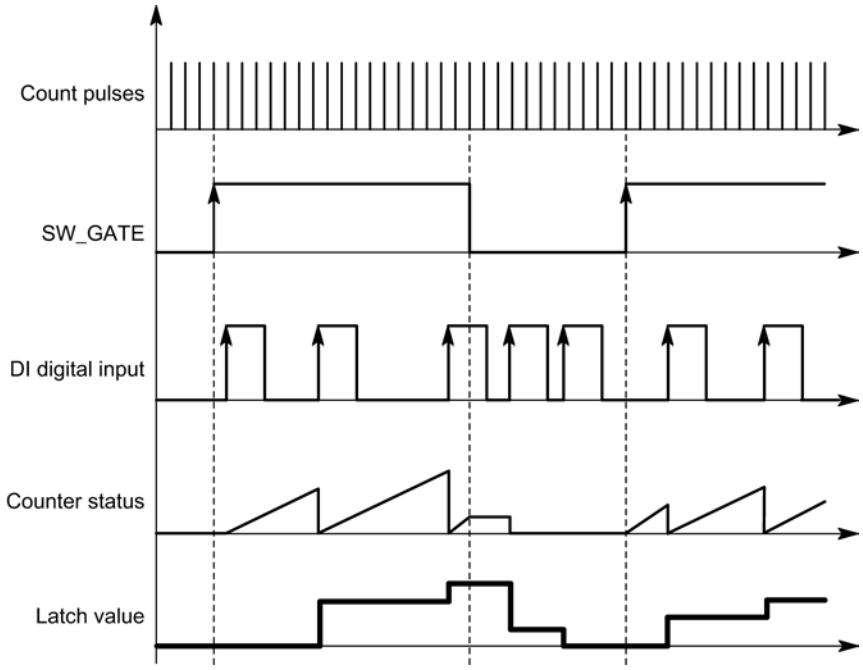
The default setting is "Input".

Latch/Retrigger with positive edge

Latch/Retrigger

This function saves the current internal counter value of the 1Count24V and retriggers counting when there is a positive edge at the digital input. This means that the current internal counter value at the time of the positive edge is stored (latch value), and the 1Count24V is then loaded again with the load value, from which counting resumes.

The following figure shows latch/retrigger when load value = 0.



The counting mode must be enabled with the software gate before the function can be executed. It is started with the first positive edge at the digital input.

The saved rather than the current counter value is displayed in the feedback interface (Page 9271). The STS_DI bit indicates the status of the latch and retrigger signal.

The latch value is preassigned its RESET status ("0"). It is not changed when the software gate is opened.

Directly loading the counter does not alter the saved counter value displayed.

Closing the software gate only interrupts counting; this means that counting resumes when you open the software gate again. The digital input DI remains active even when the software gate is closed.

Counting is also latched and triggered in isochrone mode with each edge at the digital input. The counter status that was valid at the time of the last edge before T_i is displayed in the feedback interface.

Extended feedback interface

If the 1Count24V is inserted behind an IM 151 that supports the reading and writing of broader user data interfaces, the current count value can be read from bytes 8 to 11 of the feedback interface (Page 9271).

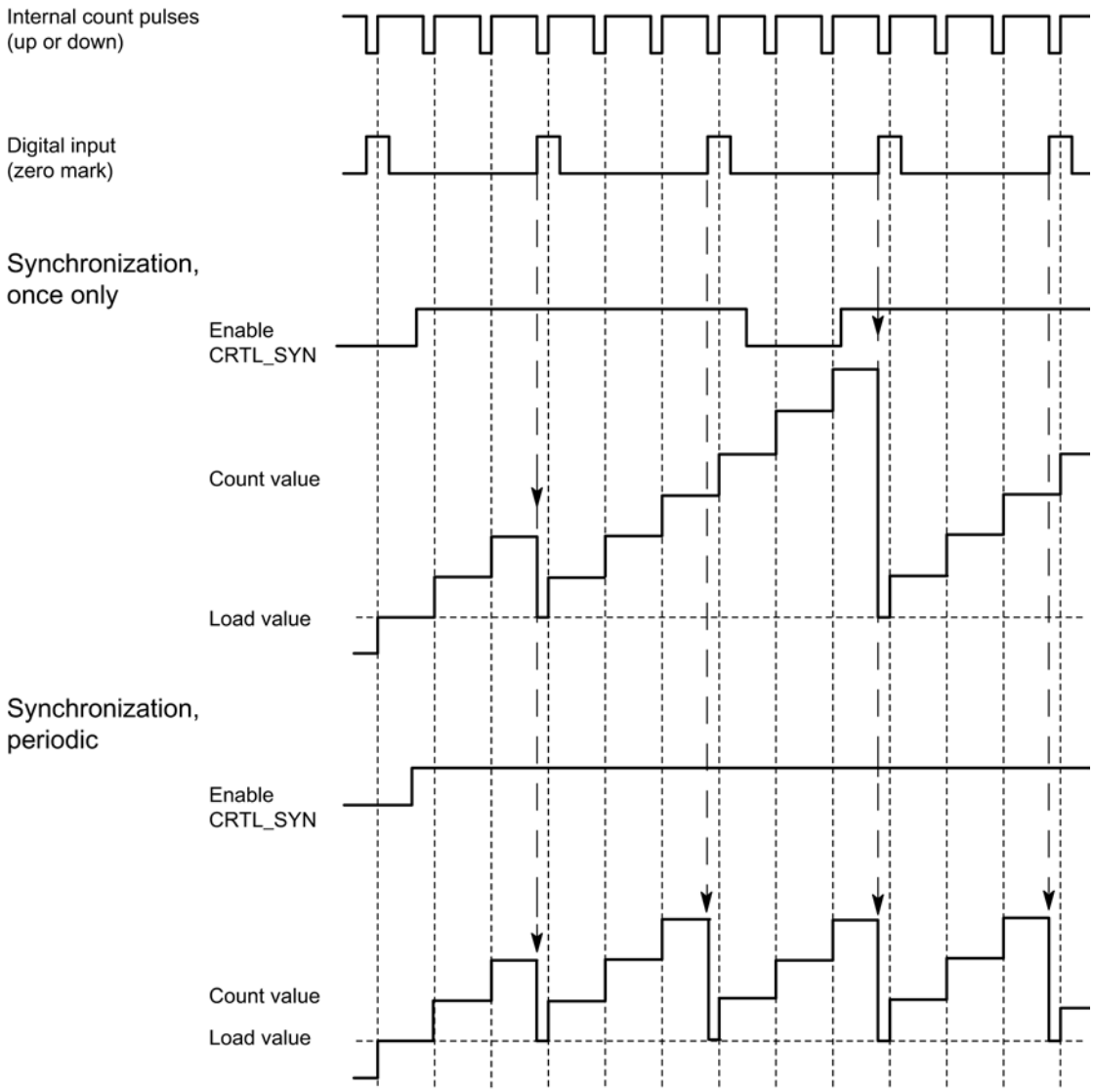
Synchronization with positive edge

Synchronization

If you have configured "Synchronization with positive edge", the rising edge of a reference signal at the input will set 1Count24V to the load value.

You can select between once-only and periodic synchronization ("Synchronization" parameter).

The figure below shows the counter values in once-only and periodic synchronization.



The following conditions apply:

- The counting mode must have been started with the SW gate.
- The "Enable synchronization" (CTRL_SYN) control bit (Page 9270) must be set.
- Synchronize once only: If the enable bit is set, the first edge loads the 1Count24V with the load value.
- Synchronization periodically: If the enable bit is set, the first and each subsequent edge loads the 1Count24V with the load value.
- The feedback bit (Page 9271)STS_SYN is set after successful synchronization. It must be reset by the control bit (Page 9270)RES_STS.

- The signal of a bounce-free switch or the zero mark of a rotary encoder can serve as the reference signal.
- The feedback bit (Page 9271)STS_DI indicates the level of the reference signal.

In isochrone mode, the set feedback bit (Page 9271)STS_SYN indicates that the rising edge at the digital input was between time T_i in the current cycle and time T_i in the previous cycle.

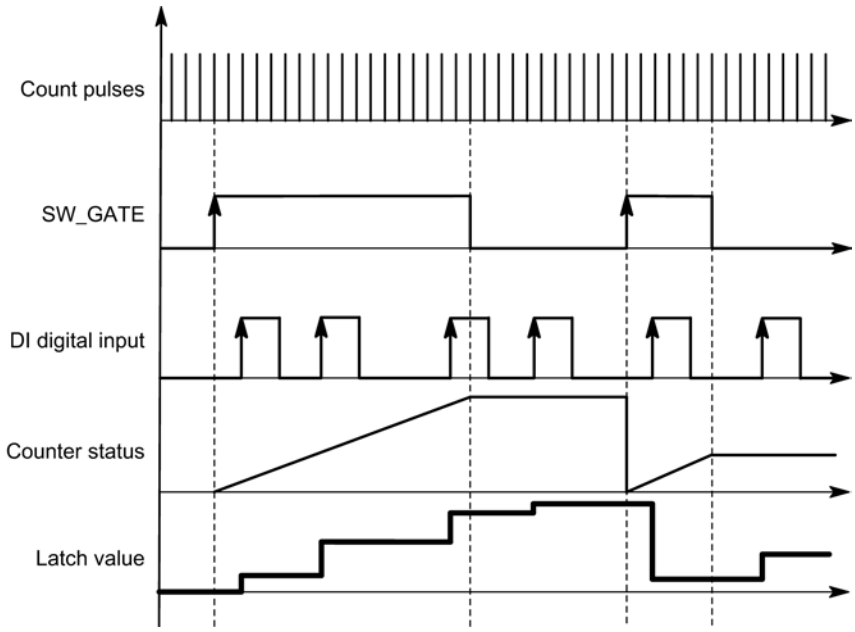
Latching with positive edge

Latching

Counter value and latch value are pre-assigned their RESET statuses ("0"):

The counting function is started when the software gate is opened. 1Count24V begins at the load value ("0").

The following figure shows latching when load value = 0.



The latch value is always the exact counter value at the time of the positive edge at the digital input DI.

The saved rather than the current counter value is displayed in the feedback interface (Page 9271). The STS_DI bit indicates the level of the latch signal.

Directly loading the counter does not alter the saved counter value displayed.

In isochrone mode, the counter status that was latched at the time of the last positive edge before T_i is displayed in the feedback interface.

Closing the software gate will either cancel or interrupt depending on the parameter assignment. The digital input DI remains active even when the software gate is closed.

Extended feedback interface

If the 1Count24V is inserted behind an IM 151 that supports the reading and writing of broader user data interfaces, the current count value can be read from bytes 8 to 11 of the feedback interface (Page 9271).

Selecting hardware gate input signal

Hardware gate input signal

The level of the digital input for the "Function DI = HW gate" configuration with the parameter "Input signal HW gate" can be inverted.

The feedback bit (Page 9271)STS_DI indicates the level of the digital input.

Selecting synchronization

Synchronization

If you have selected "Synchronization with positive edge" at the "DI function (Page 9266)" parameter, this is where you can select either once-only or periodic synchronization:

- Once-only: If the enable bit is set, the first edge loads the 1Count24V with the load value.
- Periodic: If the enable bit is set, the first and each subsequent edge loads 1Count24V with the load value.

Control and feedback interface for position feedback

Position feedback control interface

Note

The following control interface data is consistent for 1Count24V:

- Bytes 0 ... 3
- Bytes 4 ... 7
- Bytes 8 ... 11 (extended user data interface)

Control interface assignment

The table below shows the assignment of the 1Count24V control interface (outputs) for position feedback.

Address	Assignment		Name
Bytes 0 to 3	Load value directly, in preparation		
Byte 4	Bit 7	Error acknowledgment	EXTF_ACK

Address	Assignment		Name
	Bit 6	Reserved = 0	
	Bit 5	Reserved = 0	
	Bit 4	Reserved = 0	
	Bit 3	Reserved = 0	
	Bit 2	Trigger status bits reset	RES_STS
	Bit 1	Enable synchronization	CTRL_SYN
	Bit 0	Software gate control bit	SW_GATE
Byte 5	Bit 7	Reserved = 0	
	Bit 6	Reserved = 0	
	Bit 5	Reserved = 0	
	Bit 4	Reserved = 0	
	Bit 3	Reserved = 0	
	Bit 2	Reserved = 0	
	Bit 1	Load counter in preparation	LOAD_PREPARE
	Bit 0	Load counter directly	LOAD_VAL
Bytes 6 to 7	Reserved = 0 [1]		

[1] Not used for extended user interface

Notes on the control bits

Bit	Explanation
CTRL_SYN	You use this to enable synchronization.
EXTF_ACK	Error acknowledgment The error bits must be acknowledged with the EXTF_ACK control bit after the cause has been eliminated.
LOAD_PREPARE	Load counter in preparation The value from bytes 0 to 3 is applied as the load value.
LOAD_VAL	The value from bytes 0 to 3 is loaded directly as the new counter value.
RES_STS	Trigger status bits reset The status bits are reset by the acknowledgment process between the RES_STS bit and the RES_STS_A bit.
SW_GATE	The software gate is opened/closed via the control interface with the SW_GATE bit.

Position feedback feedback interface

Note

The following feedback interface data is consistent for 1Count24V:

- Bytes 0 ... 3
- Bytes 4 ... 7
- Bytes 8 ... 11 (extended user data interface)

Feedback interface assignment

The table below shows the assignment of the 1Count24V feedback interface (inputs) for position feedback.

Address	Assignment		Name
Bytes 0 to 3	Counter value or stored counter value in the case of the latch function at the digital input		
Byte 4	Bit 7	Encoder supply short circuit	ERR_24V
	Bit 6	Reserved = 0	
	Bit 5	Parameter assignment error	ERR_PARA
	Bit 4	Reserved = 0	
	Bit 3	Reserved = 0	
	Bit 2	Resetting of status bits active	RES_STS_A
	Bit 1	Load function error	ERR_LOAD
	Bit 0	Load function is running	STS_LOAD
Byte 5	Bit 7	Status down	STS_C_DN
	Bit 6	Status up	STS_C_UP
	Bit 5	Reserved = 0	
	Bit 4	Reserved = 0	
	Bit 3	Reserved = 0	
	Bit 2	Reserved = 0	
	Bit 1	DI status	STS_DI
	Bit 0	Internal gate status	STS_GATE
Byte 6	Bit 7	Zero pass	STS_ND
	Bit 6	Low count limit	STS_UFLW
	Bit 5	High count limit	STS_OFLW
	Bit 4	Reserved = 0	
	Bit 3	Reserved = 0	
	Bit 2	Reserved = 0	
	Bit 1	Reserved = 0	
	Bit 0	Synchronization status	STS_SYN
Byte 7	Reserved = 0		
Bytes 8 to 11	Counter value [1]		

[1] Extended user data interface

Notes on the feedback bits

Bit	Explanation
ERR_24V	Encoder supply short circuit The error bit must be acknowledged with the control bit (Page 9270)EXTF_ACK. Diagnostic alarm if parameterized.
ERR_LOAD	Load function error TheLOAD_VAL and LOAD_PREPAREcontrol bits (Page 9270) cannot be set simultaneously during transfer. As when you load an incorrect value (which is not accepted), this results in the ERR_LOAD status bit being set.
ERR_PARA	Errors in assigned module parameters.
RES_STS_A	Resetting of status bits active
STS_C_DN	Status down
STS_C_UP	Status up
STS_DI	DI status The status of the DI is indicated in all modes by the STS_DI bit in the feedback interface.
STS_GATE	Internal gate status: Counting.
STS_LOAD	Load function is running
STS_ND	Zero pass in the counting range during counting without a main count direction. The bit must be reset by the control bit (Page 9270)RES_STS.
STS_OFLW STS_UFLW	High count limit exceeded Low count limit violated Both bits must be reset.
STS_SYN	Synchronization status: The STS_SYN bit is set after successful synchronization. It must be reset by the control bit (Page 9270)RES_STS.

Fast mode parameters

Selecting reaction to CPU/Master STOP

Setting response to CPU/Master STOP

You can configure the response of the 1Count24V to failure of the higher-level controller.

Parameter	Status of the 1Count24V at CPU/Master STOP	What happens when new parameters are assigned?
Turn off DO1	The current mode is cancelled. The gate is closed and the digital output disabled. Comparison values 1 and 2 the load value are reset. The high and low limit, function and behavior of the digital outputs and the integration time are transferred from the parameters.	The changed parameters are accepted and become effective.
Continue working mode [1]	The current mode continues working. The gate and digital output status does not change.	The gate is closed. The current mode is cancelled. The digital output is disabled. The changed parameters are accepted and become effective.

[1] The CPU/Master must not clear the outputs if the mode is to continue working on transition from CPU/Master STOP to RUN (startup).
Possible solution: In the part of the user program that is executed during startup, set the SW_GATE control bit and transfer the values to 1Count24V.

Automatic new parameter assignment

New parameters are assigned to the ET 200S station by your CPU/ DP master:

- Upon CPU/DP master power on
- Upon IM 151-1 / IM 151-1 FO power on
- After failure of DP transmission
- Once altered parameters or configuration of the ET 200S station have been loaded to the CPU/DP master
- When the 1Count24V is inserted
- Upon power on or insertion of the corresponding power module

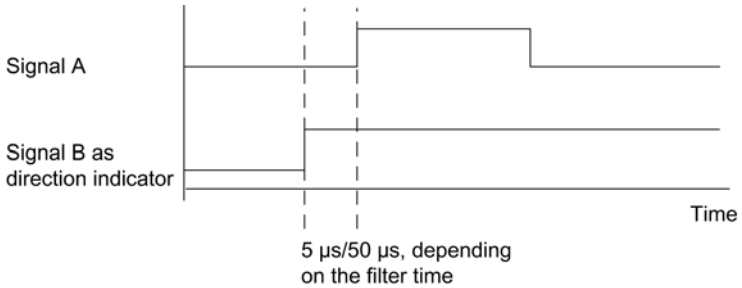
Signal evaluation

Selecting count and direction evaluation

Signal evaluation A*, B*

Signal evaluation by means of A*, B* allows you to count directionally. Different evaluation modes are possible depending on the parameter assignment:

- Pulse and Direction (Page 9275)
  In the case of 24 V pulse generators with a direction level, there must be an interval of at least 5 µs/50 µs (depending on the input filter set) between the direction signal (B*) and the count signal (A*).
  The following figure shows the interval between direction signal and count signal.



- Rotary Encoder (Page 9108)
  If you connect a 24 V rotary encoder with two tracks that are 90 degrees out of phase at the count and direction inputs, you can set single evaluation (Page 9108) in all counting modes. You can alternatively set dual (Page 9108) or quadruple evaluation (Page 9109). You can invert the count direction (Page 9276) at direction input B* in all evaluation modes. A range of different sensors can be connected to the encoder inputs (Sourcing output/push pull or Sinking output).

Note

If you have selected the "Sinking output" setting with 1Count24V for the "Encoder inputs" parameter, you must use the M-switching sensors.

Note

The count frequency of 100 kHz is the maximum frequency of the A* and B* signals. The maximum frequency for count pulses is therefore 200 kHz in double evaluation and 400 kHz in quadruple evaluation.

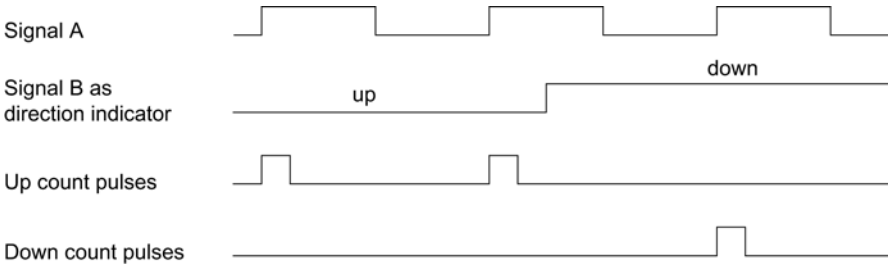
"Pulse and direction" evaluation

Pulse and direction

The level at direction input B* is used as the direction setting in this type of evaluation.

An unwired input corresponds to the "Up" count direction if you have selected "Pulse and direction" for the "Signal evaluation" parameter.

The diagram below shows the signals of a 24 V pulse generator with direction level.



Selecting filters for input signals

Input filter

You can activate a filter to filter signals at count input A*, direction input B* and digital input DI in line with the minimum pulse duration or the maximum signal frequency.

You can select a minimum pulse width of 2.5 µs or 25 µs for each input.

Selecting encoder inputs

Encoder inputs

A range of different sensors can be connected to the encoder inputs (Sourcing output/push pull or Sinking output).

Note

If you have selected the "Sinking output" setting with for the "Encoder inputs" parameter, you must use the M-switching sensors.

Selecting the count direction

Count direction

To adapt the count direction to the process, you can select:

- Normal count direction:
- Inverted count direction

Selecting the gate function

Description

When configuring the hardware and software gates, you can specify whether the internal gate is to cancel or interrupt counting.

- Cancel counting (Page 9103): The counting process restarts from the beginning after gate stop and restart.
- Interrupt counting (Page 9103): The counting process starts from the last current counter value after gate stop and restart.

DI function

Selecting DI function

Description

You select the digital input function with "DI function":

- Input
  The digital input operates as a "normal" input.
- Hardware gate (Page 9102), level-controlled
  The hardware gate is opened and closed by signals at the digital input.
- Synchronization with positive edge (Page 9277)
  The counter is set to the load value with positive edges at the digital input.

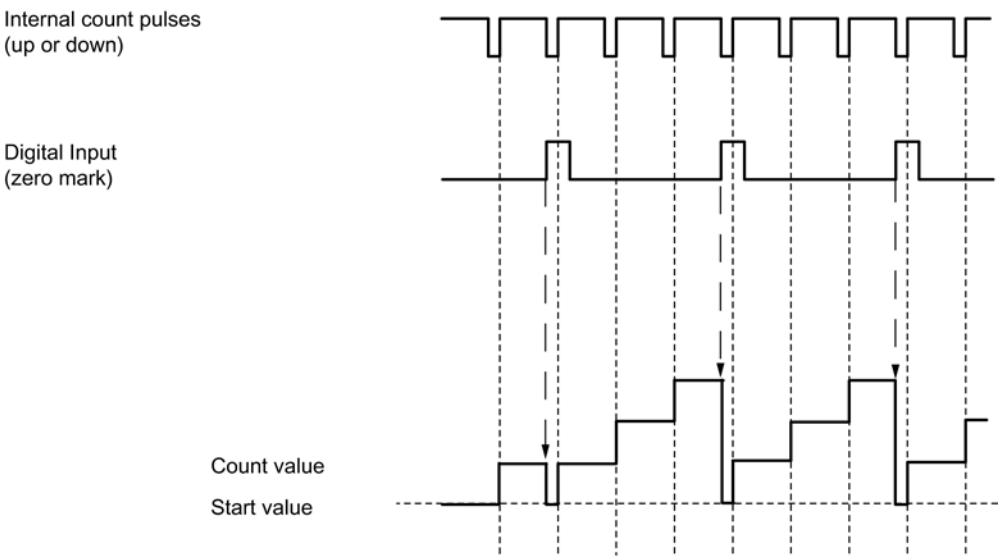
The default setting is "Input".

Synchronization with positive edge

Synchronization

If you have configured "Synchronization with positive edge", the positive edge of a reference signal at the input will set 1Count24V to the initial value (= load value).

The figure below shows the counter values for synchronization with a positive edge.



The following conditions apply:

- "Fast mode" must be active (hardware gate).
  - When synchronization is activated, the first edge and each additional edge loads 1Count24V with the initial value (=load value).
- The signal of a bounce-free switch or the zero mark of a rotary encoder can serve as the reference signal.
- The STS_DI feedback bit indicates the level of the reference signal.

Selecting hardware gate input signal

Hardware gate input signal

The level of the digital input for the "Function DI = HW gate" configuration with the parameter "Input signal HW gate" can be inverted.

The feedback bit (Page 9279)STS_DI indicates the level of the digital input.

Specifying load value

Load value

You can specify a load value for 1Count24V.

This load value is applied directly as the initial value.

Valid value range: -16777216 to +16777215

Fast mode feedback interface

Note

The following feedback interface data is consistent for 1Count24V:

- Bytes 0 ... 3

Feedback interface assignment

The table below shows the assignment of the 1Count24V feedback interface (inputs) for fast mode.

Address	Assignment		Name
Bytes 0 to 3	Bit 31	Lifebeat	LZ
	Bit 30	Isochronous mode applied	STS_TIC
	Bit 29	Parameter assignment error	ERR_PARA
	Bit 28	Group error	EXTF
	Bit 27	DI status	STS_DI
	Bit 26	Status of direction up/down	STS_DIR
	Bit 25	Internal gate status	STS_GATE
	Bits 0 to 24	Count value	

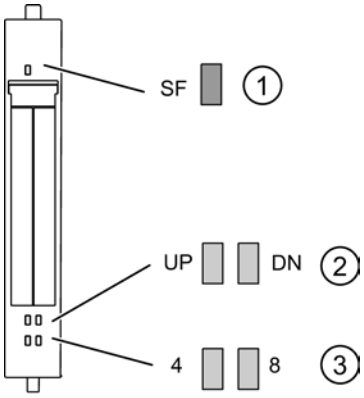
Notes on the Feedback Bits

Bit	Explanation
ERR_PARA	Errors in assigned module parameters.
EXTF	Group error Possible cause: • Encoder supply short circuit EXTF is reset when the causes of the errors have been eliminated.
LZ	The lifebeat is toggled each time the feedback interface is updated, i.e. the last value sent is inverted.
STS_DI	The bit displays the status of the digital input DI.
STS_DIR	Direction status: • for encoder value change from larger to smaller encoder positions (including zero pass) → "1 " • for encoder value change from smaller to larger encoder positions (including zero pass) → "0 "
STS_GATE	Internal gate status: Counting.
STS_TIC	Isochronous mode (if assigned) was applied.

1Count24V diagnostics

Diagnostics using the LED display

LED display on the 1Count24V



① Group error (red)
② Status display for counting direction (green)
③ Status display for digital input/digital output (green)

Status and error displays by means of LEDs on the 1Count24V

The table below shows the status and fault displays on the 1Count24V.

Event (LED)					Cause	Remedy
SF	UP	DN	4	8		
On					No configuration or incorrect module plugged in. There is a diagnostic message.	Check the parameter assignment. Evaluate the diagnostics data.
	On				Status of the low-order bits of the counter, if the counter counts up.	
		On			Status of the inverse low-order bits of the counter, if the counter counts down.	
			On		DO (direct control, comparator output) activated.	
				On	DI (HW gate, synchronization, latch) activated.	

Error types

For information on the structure of the channel-related diagnostics, refer to the manual on the interface module used in your ET 200S station.

1Count24V error types

The following table shows the 1Count24V error types.

Error type		Meaning	Remedy
1_D	00001: Short circuit	Short circuit of the sensor supply or the actuator.	Check the wiring to the sensor. Correct the process wiring.
5_D	00101: Overtemperature	The digital output is overloaded.	Correct the process wiring.
6_D	00110: Open circuit	Line to the actuator is interrupted.	Correct the process wiring.
9_D	01001: Error	Internal module error occurred.	Replace the module.
		Load voltage from the power module is too low.	Correct the process wiring. Check the load voltage.
16_D	10000: Parameter assignment error	Parameters have not been assigned to the module.	Correct the parameter assignment.

Using ET 200S 1Count5V

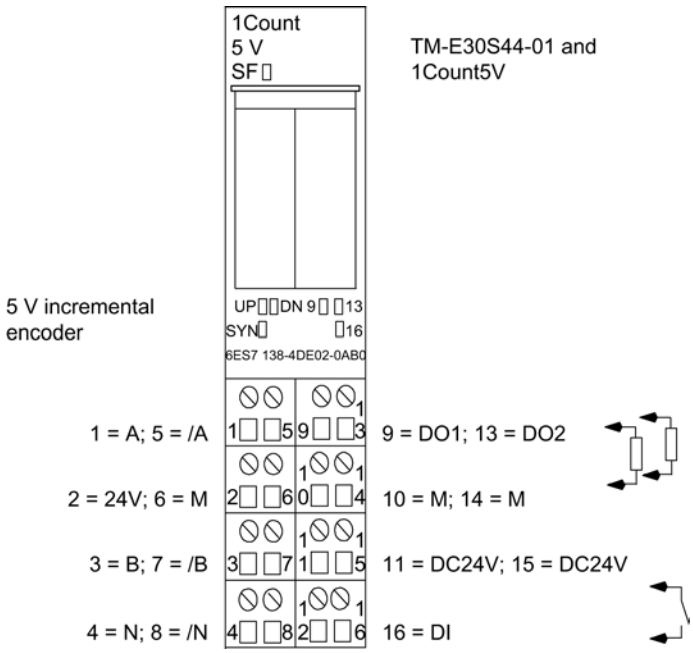
1Count5V terminal assignment

Wiring Requirements

The cables (terminals 1 and 8 and terminals 15 and 16) must be shielded. The shield must be supported at both ends. To do this, use the shield support (see the appendix to the ET 200S Distributed I/O System (http://support.automation.siemens.com/WW/view/en/1144348) operating instructions).

Terminal assignment of the 1Count5V

The following table shows the terminal assignment for 1Count5V.

View	Terminal Assignment	Remarks
		A, /A: Track A B, /B: Track B N, /N: Track N 24V DC: Sensor supply M: Chassis ground DI: Digital input DO1: Digital output DO2: Digital output

Configuring and assigning parameters to 1Count5V

Operation

Selecting an operating mode

Operating mode

You define 1Count5V functionality by specifying an operating mode.

Operating mode	Description
Counting (Page 9283)	Preselection for counting modes
Measuring (Page 9306)	Preselection for measuring modes

Operating mode	Description
Position feedback (Page 9321)	In this mode, 1Count5V counts continuously from the load value: • If another count pulse is received after 1Count5V reaches the high count limit when counting up, it will jump to the low count limit and continue counting from there without losing a pulse. • If another count pulse is received after 1Count5V reaches the low count limit when counting down, it will jump to the high count limit and continue counting from there without losing a pulse.
Fast mode (Page 9332) (supported as of firmware version V2.0)	In this mode, 1Count5V counts continuously from the initial value: • If another count pulse is received after 1Count5V reaches the maximum value that can be represented by 25 bits when counting up (all counter bits are set), the count value will jump to "0" and resume counting from there without losing a pulse. • If 1Count5V reaches the value "0" when counting down and another count pulse is then received, the count value will jump to the maximum value that can be represented by 25 bits (all counter bits set) and resume counting from there without losing a pulse. This operating mode can be used for position feedback in particularly short isochrone cycles.

Counting parameters

Enabling group diagnostics

Group diagnostics

Channel-specific diagnostics will be carried out if you enable group diagnostics in your parameter assignment.

Selecting reaction to CPU/Master STOP

Setting response to CPU/Master STOP

You can configure the response of the 1Count5V to failure of the higher-level controller.

Parameter	Status of the 1Count5V at CPU/Master STOP	What happens when new parameters are assigned?
Turn off DO	The current mode is cancelled. The gate is closed and the digital output disabled. Comparison values 1 and 2 the load value are reset. The high and low limit, function and behavior of the digital outputs and the integration time are transferred from the parameters.	The changed parameters are accepted and become effective.
Continue working mode [1)]	The current mode continues working. The gate and digital output status does not change.	The gate is closed. The current mode is cancelled. The digital output is disabled. The changed parameters are accepted and become effective.

Parameter	Status of the 1Count5V at CPU/Master STOP	What happens when new parameters are assigned?
DO substitute a value	The current mode is cancelled. The gate is closed and the substitute value set for the digital output is activated. Comparison values 1 and 2 and the load value are reset. The high and low limit, function and behavior of the digital outputs and the integration time are transferred from the parameters. If "Pulse when comparison value is reached" has been set for the output, the only available pulse duration substitute value will be 1.	The changed parameters are accepted and become effective.
DO keep last value	The current mode is cancelled. The gate is closed and the status of the digital output is maintained. Comparison values 1 and 2 and the load value are reset. The high and low limit, function and behavior of the digital outputs and the integration time are transferred from the parameters.	The changed parameters are accepted and become effective.

1) The CPU/Master must not clear the outputs if the mode is to continue working on transition from CPU/Master STOP to RUN (startup).
Possible solution: In the part of the user program that is executed during startup, set the SW_GATE control bit and transfer the values to 1Count5V.

Exiting parameterized status

Under what conditions does the 1Count5V exit the parameterized status?

The CPU or master must be in RUN mode and you must make a change at the control interface (Page 9301).

Automatic new parameter assignment

New parameters are assigned to the ET 200S station by your CPU/ DP master:

- Upon CPU/DP master power on
- Upon IM 151-1 / IM 151-1 FO power on
- After failure of DP transmission
- Once altered parameters or configuration of the ET 200S station have been loaded to the CPU/DP master
- When the 1Count5V is inserted
- Upon power on or insertion of the corresponding power module

Selecting signal evaluation

Signal Evaluation A, B

Signal evaluation by means of A, B allows you to count directionally. Different evaluation modes are possible depending on the parameter assignment:

- Rotary Encoder (Page 9108)
  The 1Count5V can count the edges of the signals. Normally, only the edges at A are evaluated (single evaluation). To obtain a higher resolution, you can select whether the signals are to undergo single (Page 9108), double (Page 9108) or quadruple (Page 9109) evaluation when assigning parameters ("Signal evaluation" parameter).
  Multiple evaluation is only possible with asymmetric incremental encoders with A and B signals that are 90 degrees out of phase.

Note

500 kHz for count frequency is the maximum frequency of the A and B signals. This means a maximum frequency of 1 MHz for the count pulses in double evaluation and a maximum of 2 MHz in quadruple evaluation.

Activating signal pair diagnostics

Signal pair monitoring

Signal pairs A and /A, B and /B, and N and /N can be monitored.

Activate the corresponding check box.

Note

You cannot activate "N, /N diagnostics" until you have activated "A, /A and B, /B diagnostics".

Selecting the count direction

Count direction

To adapt the count direction to the process, you can select:

- Normal count direction:
- Inverted count direction

Behavior of the outputs in counting modes

Comparison values and outputs

Introduction

The 1Count5V allows you to store two comparison values, which are assigned to the digital outputs. The outputs can be activated in line with counter value and comparison values.

Digital outputs

1Count5V has two digital outputs.

Parameters can be assigned for both outputs ("Function DO1" and "Function DO2" parameters).

You can change the function and the behavior of the digital outputs during operation.

Controlling the outputs and comparators simultaneously

You can continue to control the outputs using SET_DO1 or SET_DO2 if you have selected a compare function for the outputs. This allows you to simulate the effect of the compare functions in your control program:

- The output is set with the positive edge of SET_DO1 or SET_DO2.
  However, if the "Pulse on Reaching the Comparison Value" function is selected, only one pulse with the specified duration is output. When pulse duration = 0, the output can be set with SET_DO1 or SET_DO2 as long as the counter value is at the comparison value or hysteresis is active.
  The SET_DO1 control bit cannot be used with output behavior "Switch at comparison values".
- A SET_DO1 or SET_DO2 negative edge resets the output.

Note that the comparators are still active and can set or reset the output if the comparison result changes.

Note

An output set by SET_DO1 or SET_DO2 cannot be reset (by the comparator) at the comparison value.

Loading Comparison Values

You transfer the comparison values to the 1Count5V. The counting is not affected by this.

Valid Range for the Two Comparison Values

Main count direction: None	Main count direction: Up	Main count direction: Down
Low count limit to high count limit	-2147483648 to high count limit -1	1 to 2147483647

Modifying the Function and Behavior of Digital Outputs

You can change the functions and behavior of the outputs during operation using the control interface (Page 9301). The 1Count5V deletes the outputs and accepts the values as follows:

- **Function of digital outputs DO1 and DO2:** If you change this function so that the comparison condition is satisfied, the output will not be set until after the next count pulse. However, 1Count5V will not make any changes at the output if hysteresis (Page 9291) is enabled.
- **Hysteresis:** Active hysteresis (Page 9291) remains active after a change. The new hysteresis range will be applied the next time the comparison value is reached.
- **Pulse duration:** The new pulse duration (Page 9294) will take effect with the next pulse.

Selecting output behavior

Behavior of outputs DO1 and DO2

Select the behavior of the output in question in the drop down list:

- Output (Page 9288)
- On when counter value ≥ comparison value (Page 9288)
- On when counter value ≤ comparison value (Page 9288)
- Pulse on reaching the comparison value (Page 9288)
- Switch at comparison values (Page 9289) (DO1 only)

Setting or Modifying the Function and Behavior of the Digital Output DO1

When setting or modifying the behavior of DO1, you must take all assignable interdependencies into account. Failure to do so will generate a parameter assignment error or a loading error.

Boundary conditions:

If you assign "Switch at comparison values" for DO1, you must:

- Set hysteresis = 0, and
- also configure "DO2 function" as "Output".

Output

Output

You can switch the outputs on and off with the control bits (Page 9301)SET_DO1 and SET_DO2.

The control bits (Page 9301)CTRL_DO1 or CTRL_DO2 must be set.

You can query the status of the outputs with the status bits STS_DO1 and STS_DO2 in the feedback interface (Page 9304).

The status bits (Page 9304)STS_CMP1 and STS_CMP2 indicate that the relevant output is or was switched on. These status bits retain their status until they are acknowledged. If the output is still activated, the corresponding bit will be set again immediately. These status bits are also set when the control bit SET_DO1 or SET_DO2 is operated without DO1 or DO2 being enabled.

Isochrone mode

Outputs DO1 and DO2 are switched at time T_o in isochrone mode.

On when counter value ≤ comparison value or when counter value ≥ comparison value

Counter value ≤ comparison value and Counter value ≥ comparison value

If the comparison conditions are fulfilled, the respective comparator will switch on the output. The status of the output (Page 9304) is indicated by STS_DO1 and STS_DO2.

The control bits (Page 9301)CTRL_DO1 or CTRL_DO2 must be set.

The result of the comparison is indicated by the status bits (Page 9304)STS_CMP1 or STS_CMP2. You cannot acknowledge and thus reset these bits until the comparison conditions are no longer fulfilled.

Isochrone mode

In isochrone mode, too, the DO1 and DO2 outputs are switched as soon as the comparison condition is fulfilled and are therefore independent of the bus cycle.

Pulse on reaching the comparison value

Pulse on reaching the comparison value

If the counter status reaches the comparison value, the comparator switches on the respective digital output for the assigned pulse duration.

The control bit (Page 9301)CTRL_DO1 or CTRL_DO2 must be set.

The status of the status bits (Page 9304)STS_DO1 and STS_DO2 is always the same as that of the corresponding digital output.

The comparison result is indicated by the status bit (Page 9304)STS_CMP1 or STS_CMP2 and cannot be reset by acknowledgment until the pulse duration has elapsed.

If a main count direction is assigned, the comparator will only switch when the comparison value in the main count direction is reached.

If a main count direction is not assigned, the comparator switches when the comparison value is reached from either direction.

If the digital output is set by control bit (Page 9301)SET_DO1 or SET_DO2, it will be reset when the pulse duration has elapsed.

Isochrone mode

In isochrone mode, too, the DO1 and DO2 outputs are switched as soon as the comparison condition is fulfilled and are therefore independent of the bus cycle.

Switch at comparison values

Switch at comparison values

The comparator switches the output when the following conditions are met:

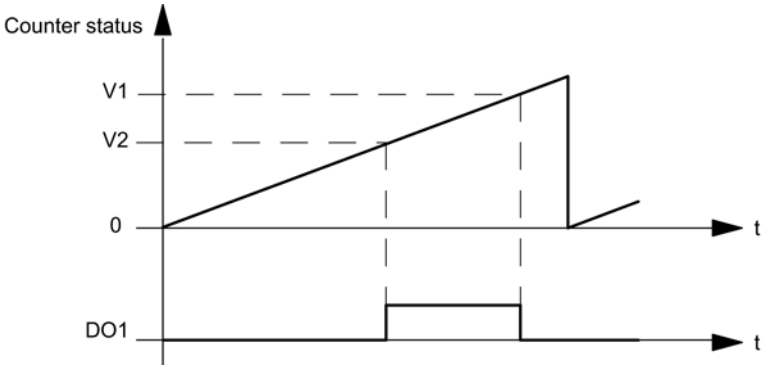
- The two comparison values must be loaded with the load functions (Page 9301)CMP_VAL1 and CMP_VAL2 and
- The output DO1 (Page 9301) must be enabled with CRTL_DO1 after the comparison values are loaded.

The following table shows you when the DO1 is switched on or off:

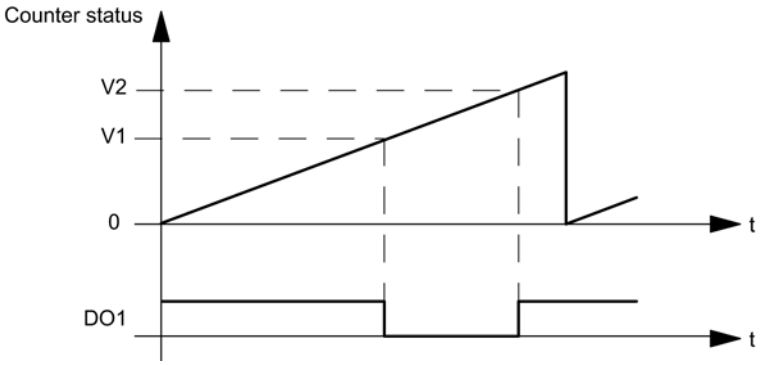
	DO1 is switched on when	DO1 Is switched off when
V2 < V1 (see Figure below)	V2 ≤ counter status ≤ V1	V2 > counter status or counter status > V1
V2 = V1	V2 = counter status = V1	V2 ≠ counter status ≠ V1
V2 > V1 (see Figure below)	V1 > counter status or counter status > V2	V1 ≤ counter status ≤ V2

The figures below show the status of output DO1 as determined by the counter value and comparison values:

- V2 < V1 at the start of counting



- V2 > V1 at the start of counting



The result of the comparison is shown by the status bit (Page 9304)STS_CMP1. You can only acknowledge and thus reset this bit when the comparison condition is no longer fulfilled.

There is no hysteresis in the case of this output behavior.

It is not possible to control the DO1 output with control bit (Page 9301)SET_DO1 if this is the output behavior.

Isochronous mode

In isochrone mode, too, output DO1 is switched as soon as the comparison condition is fulfilled and is therefore independent of the bus cycle.

Selecting substitute value DOx

Substitute values DO1 and DO2

This is where you select the substitute value which is to be output at digital output DO1 or DO2 if you have selected the parameter "DO substitute a value" under "Reaction to CPU/Master STOP (Page 9283)".

Activating DOx diagnostics

DO1 and DO2 diagnostics

"DO1 diagnostics" or "DO2 diagnostics" (wire break, short circuit) are possible only with pulse lengths of > 90 ms at digital output DO1 or DO2.

Select the check box for diagnostics at output DO1 or DO2.

Setting hysteresis

Hysteresis

An encoder can remain at a particular position and then fluctuate around this position. This state causes the counter status to fluctuate around a particular value. If there is a comparison value in this fluctuation range, for example, the associated output is switched on and off in accordance with the rhythm of the fluctuations. The 1Count5V is equipped with programmable hysteresis to prevent this switching in response to small fluctuations. You can assign a range between 0 and 255 (0 means: hysteresis switched off).

Hysteresis also works with high limit violated and low limit violated.

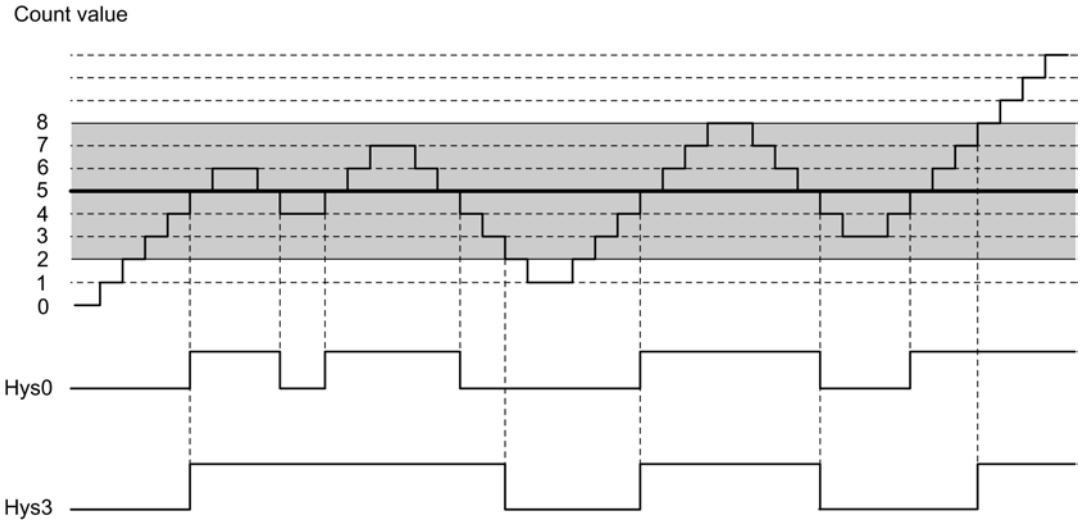
Effect when counter value ≤ comparison value and counter value ≥ comparison value

The example in the figure below demonstrates the effect of hysteresis. The figure shows the various responses of an output with a hysteresis of 0 (= switched off) as opposed to a hysteresis of 3. In the example, the comparison value is 5.

The following settings are assigned for the counter: "Main count direction" = "Up" and "Turn on at counter status≥ comparison value".

When the comparison condition is met, hysteresis becomes active. While the hysteresis is active, the comparison result remains unchanged.

Hysteresis will no longer be active if the counter value is outside the hysteresis range. The comparator switches again in line with its comparison conditions.



Note

If the count direction changes at the comparison value when hysteresis is active, the output will be reset.

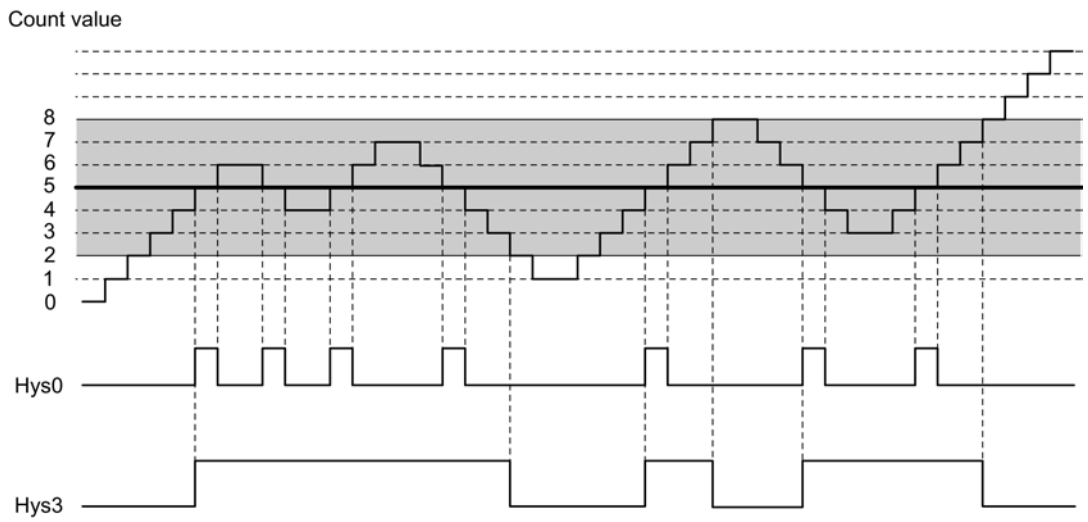
Effect when comparison value is reached and pulse duration = 0

The example in the figure below demonstrates the effect of hysteresis. The figure shows the various responses of an output with a hysteresis of 0 (= switched off) as opposed to hysteresis of 3. In the example, the comparison value is 5.

The following settings are assigned for the counter: "Pulse on reaching the comparison value", "No main count direction" and "Pulse duration = 0".

When the comparison conditions are met, hysteresis becomes active. While the hysteresis is active, the comparison result remains unchanged.

Hysteresis will no longer be active if the counter value is outside the hysteresis range. The comparator deletes the result of the comparison.



Effect when comparison value is reached, output pulse duration

The example in the figure below demonstrates the effect of hysteresis. The figure shows the various responses of an output with a hysteresis of 0 (= switched off) as opposed to hysteresis of 3. In the example, the comparison value is 5.

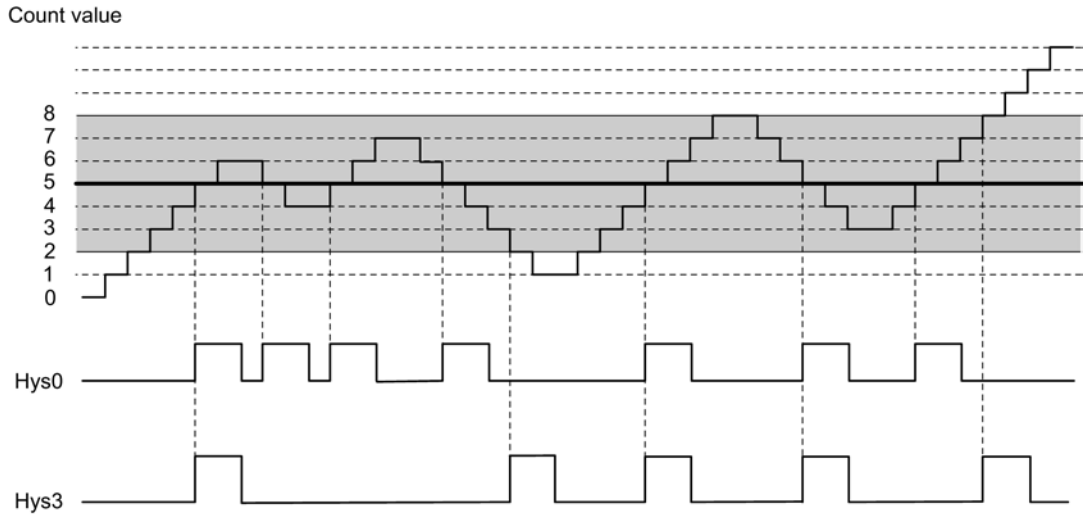
The following settings are assigned for the counter: "Pulse on reaching the comparison value", "No main count direction" and "Pulse duration > 0".

When the comparison conditions have been met, hysteresis becomes active and a pulse of the assigned duration is output.

Hysteresis will no longer be active if the counter value is outside the hysteresis range.

When hysteresis is enabled, 1Count5V stores the count direction.

A pulse is output if the hysteresis range is exited in a different direction to the one stored.



Specifying pulse duration

Pulse duration

The pulse duration begins when the respective digital output is set. The pulse duration is accurate to at least 2 ms.

The pulse duration can be set to suit the actuators used. The pulse duration specifies how long the output is to be set for. The pulse duration can be preselected in increments of 2 ms between 0 ms and 510 ms.

If the pulse duration = 0, the output will be set until the comparison condition is no longer fulfilled. Note that the count pulse times must be greater than the minimum switching times of the digital output.

Isochrone mode

In isochrone mode, too, the DO1 and DO2 outputs are switched as soon as the comparison condition is fulfilled and are therefore independent of the bus cycle.

Selecting counting mode

Description

You define 1Count5V functionality by setting a counting mode.

Operating mode	Description
Count continuously (Page 9083) (with or without gate)	1Count5V counts continuously from the current counter value.
Count once (Page 9084) (with hardware or software gate)	When the gate opens, 1Count5V counts from the load value to the count limit.
Count periodically (Page 9085) (with hardware or software gate)	When the gate opens, 1Count5V counts from the load value between the count limits.

The default setting is the "Count continuously" operating mode.

Selecting main count direction

Description

By configuring a main count direction (Page 9086) (up or down), you can reduce the maximum counting range by setting an high count limit (Page 9301). The parameterized counting range is then between 0 and the high count limit parameterized.

A main count direction can only be selected in the "Count once" and " Count periodically" modes.

You can select the following behavior:

- None (no main count direction)
- Up
- Down

Selecting the gate function

Description

When configuring the hardware and software gates, you can specify whether the internal gate is to cancel or interrupt counting.

- Cancel counting (Page 9103): The counting process restarts from the beginning after gate stop and restart.
- Interrupt counting (Page 9103): The counting process starts from the last current counter value after gate stop and restart.

DI function

Selecting DI function

Description

You select the digital input function with "DI function":

- Input
  The digital input operates as a "normal" input.
- Hardware gate (Page 9102), level-controlled
  The hardware gate is opened and closed by signals at the digital input.
- Latch/Retrigger with positive edge (Page 9296)
  Counter values are saved (latched) with positive edges at the digital input. The counter is set to the load value each time the values are saved.
- Synchronization with positive edge (Page 9297)
  The counter is set to the load value with positive edges at the digital input.
- Latching with positive edge (Page 9299)
  Counter values are saved (latched) with positive edges at the digital input.
- Hardware enable for synchronization (Page 9300)
  The digital input acts as a hardware enable. The encoder zero mark sets the counter to the load value when the hardware enable is active.

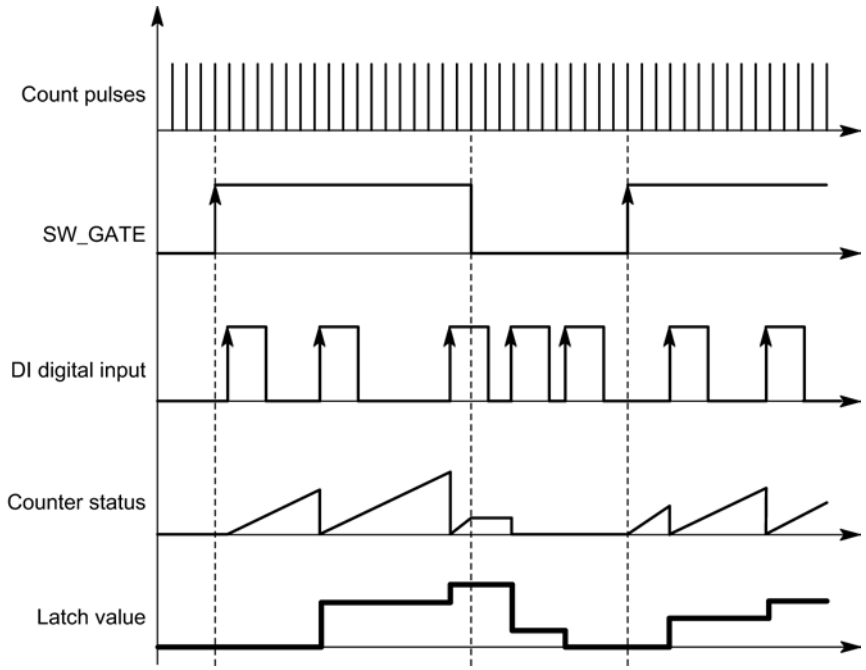
The default setting is "Input".

Latch/Retrigger with positive edge

Latch/Retrigger

This function saves the current internal counter value of the 1Count5V and retriggers counting when there is a positive edge at the digital input. This means that the current internal counter value at the time of the positive edge is stored (latch value), and the 1Count5V is then loaded again with the load value, from which counting resumes.

The following figure shows latch/retrigger when load value = 0.



The counting mode must be enabled with the software gate before the function can be executed. It is started with the first positive edge at the digital input.

The saved rather than the current counter value is displayed in the feedback interface (Page 9304). The STS_DI bit indicates the status of the latch and retrigger signal.

The latch value is pre-assigned its RESET status:

- "0" for main count direction "None" and "Up"
- "High count limit parameters assigned" for main count direction "Down"

It is not changed when the software gate is opened.

Directly loading the counter does not alter the saved counter value displayed.

Closing the software gate only interrupts counting; this means that counting resumes when you open the software gate again. The digital input DI remains active even when the software gate is closed.

Counting is also latched and triggered in isochrone mode with each edge at the digital input. The counter status that was valid at the time of the last edge before T_i is displayed in the feedback interface.

Extended feedback interface

If the 1Count5V is inserted behind an IM 151 that supports the reading and writing of broader user data interfaces, the current count value can be read from bytes 8 to 11 of the feedback interface (Page 9304).

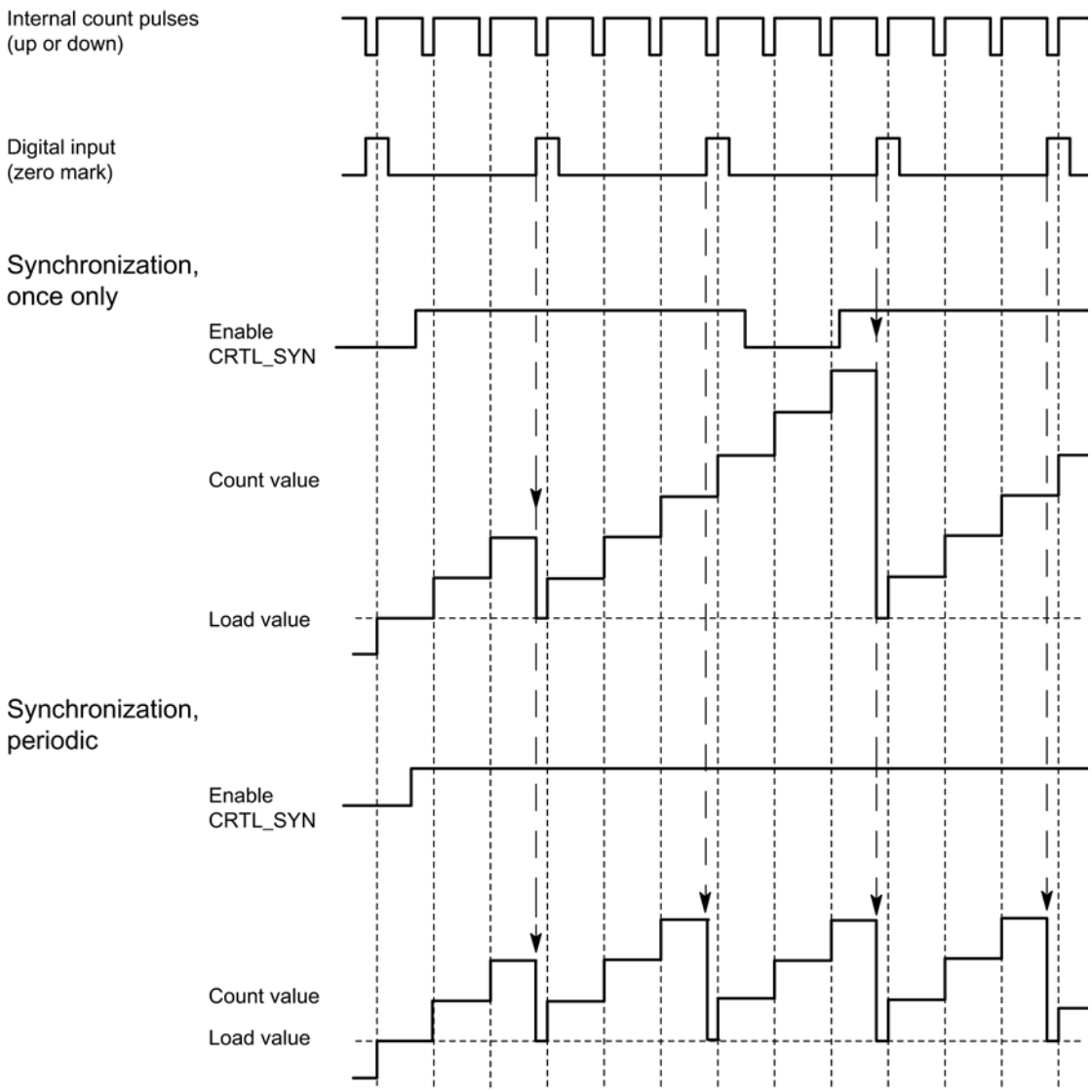
Synchronization with positive edge

Synchronization

If you have configured "Synchronization with positive edge", the rising edge of a reference signal at the input will set 1Count5V to the load value.

You can choose between once only and periodic synchronization ("Synchronization" parameter).

The figure below shows the counter values in once only and periodic synchronization.



The following conditions apply:

- The counting mode must have been started with the software gate.
- The "Enable synchronization" (CTRL_SYN) control bit (Page 9301) must be set.
- Synchronize once only: If the enable bit is set, the first edge loads 1Count5V with the load value.
- Synchronization periodically: If the enable bit is set, the first and each subsequent edge loads the 1Count5V with the load value.
- After successful synchronization, the STS_SYNfeedback bit (Page 9304) is set and the SYN LED comes on. The RES_STScontrol bit (Page 9301) resets the feedback bit and switches off the LED.
- The signal of a bounce-free switch can serve as the reference signal.
- The feedback bit (Page 9304)STS_DI indicates the level of the reference signal.

In isochrone mode, the set feedback bit (Page 9304)STS_SYN indicates that the rising edge at the digital input was between time T_i in the current cycle and time T_i in the previous cycle.

Latching with positive edge

Latching

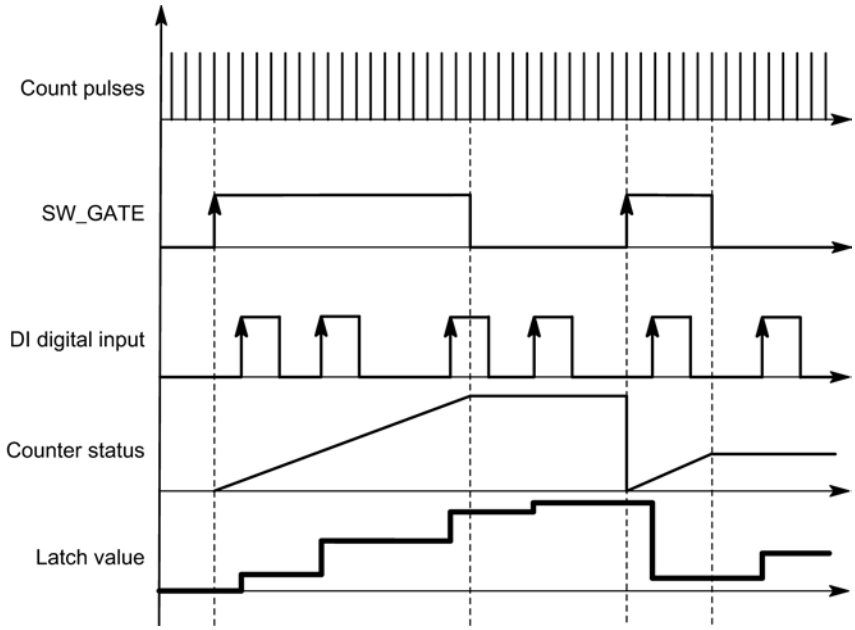
Counter value and latch value are pre-assigned their RESET statuses:

- "0" for main count direction "None" and "Up"
- "High count limit parameters assigned" for main count direction "Down"

The counting function is started when the software gate is opened. 1Count5V begins at the load value:

- "0" for main count direction "None" and "Up"
- "High count limit parameters assigned" for main count direction "Down"

The following figure shows latching when load value = 0.



The latch value is always the exact counter value at the time of the positive edge at the digital input DI.

The saved rather than the current counter value is displayed in the feedback interface (Page 9304). The STS_DI bit indicates the level of the latch signal.

Directly loading the counter does not alter the saved counter value displayed.

In isochrone mode, the counter status that was latched at the time of the last positive edge before T_i is displayed in the feedback interface.

Closing the software gate will either cancel or interrupt depending on the parameter assignment. The digital input DI remains active even when the software gate is closed.

Extended feedback interface

If the 1Count5V is inserted behind an IM 151 that supports the reading and writing of broader user data interfaces, the current count value can be read from bytes 8 to 11 of the feedback interface (Page 9304).

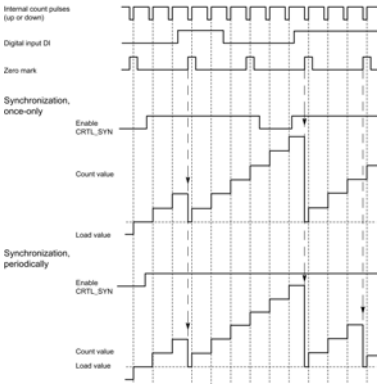
Hardware enable for synchronization

Synchronization with DI and Zero Mark

The DI will act as the hardware enable if you have configured "Hardware enable for synchronization". When the HW enable is active, the 1Count5V is loaded with the load value by the zero mark of the encoder.

You can choose between once only and periodic synchronization ("Synchronization (Page 9301)" parameter).

The figure below shows the counter values in once only and periodic synchronization.



The following conditions apply:

- The counting mode must have been started with the software gate.
- The "Enable synchronization" (CTRL_SYN) control bit (Page 9301) must be set.
- Synchronize once only: If the enable bit is set and the HW enable is present, the first zero mark loads the 1Count5V with the load value.
- Synchronization periodically: If the enable bit is set and the HW enable is present, the first and each subsequent zero mark will load 1Count5V with the load value.
- After successful synchronization, the STS_SYNfeedback bit (Page 9304) is set and the SYN LED comes on. The RES_STScontrol bit (Page 9301) resets the feedback bit and switches off the LED.
- The signal of a bounce-free switch can serve as the reference signal.
- The feedback bit (Page 9304)STS_DI indicates the level of the reference signal.

In isochrone mode, the set feedback bit (Page 9304)STS_SYN indicates that the positive edge on the digital input was between time T_i in the current cycle and time T_i in the previous cycle.

Selecting hardware gate input signal

Hardware gate input signal

The level of the digital input for the "Function DI = HW gate" configuration with the parameter "Input signal HW gate" can be inverted.

The feedback bit (Page 9304)STS_DI indicates the level of the digital input.

Selecting synchronization

Synchronization

This is where you can select either once only or periodic synchronization if you have selected "Synchronization with positive edge" at the "DI function (Page 9295)" parameter:

- Once: If the enable bit is set, the first edge loads the 1Count5V with the load value.
- Periodic: If the enable bit is set, the first and each subsequent edge loads 1Count5V with the load value.

Specifying the high count limit

Description

By configuring an high count limit together with a main count direction (Page 9086) (up or down), you can reduce the maximum counting range. The parameterized counting range is then between 0 and the high count limit parameterized.

The high count limit can be any value between > 1 and the high count range limit.

Control and feedback interface for counting modes

Control interface for counting modes

Note

The following control interface data is consistent for 1Count5V:

- Bytes 0 ... 3
- Bytes 4 ... 7
- Bytes 8 ... 11 (extended user data interface)

Control interface assignment

The table below shows the assignment of the 1Count5V control interface (outputs) for counting modes.

Address		Assignment			
Bytes 0 to 3		**Load value direct, in preparation, comparison value 1 or 2**			
	Byte 0	**Behavior of DO1, DO2 of 1Count5V**			
		Bit 2	Bit 1	Bit 0	Function DO1
		0	0	0	Output
		0	0	1	On when counter value ≥ comparison value
		0	1	0	On when counter value ≤ comparison value
		0	1	1	Pulse on reaching the comparison value
		1	0	0	Switch at comparison values
		1	0	1	disabled
		1	1	0	disabled
		1	1	1	disabled
			Bit 5	Bit 4	Function DO2
			0	0	Output
			0	1	On when counter value ≥ comparison value
			1	0	On when counter value ≤ comparison value
			1	1	Pulse on reaching the comparison value
		Bits 3, 6, and 7: Reserve = 0			
	Bytes 1 to 3	Byte 1	Hysteresis DO1, DO2 (value range 0 to 255)		
		Byte 2	Pulse duration DO1, DO2 (value range 0 to 255)		
		Byte 3	Reserved = 0		
					Name
Byte 4		Bit 7	Error acknowledgment		EXTF_ACK
		Bit 6	Enable DO2		CTRL_DO2
		Bit 5	Control bit DO2		SET_DO2
		Bit 4	Enable DO1		CTRL_DO1
		Bit 3	Control bit DO1		SET_DO1
		Bit 2	Trigger status bits reset		RES_STS
		Bit 1	Enable synchronization		CTRL_SYN
		Bit 0	Software gate control bit		SW_GATE
Byte 5		Bit 7	Reserved = 0		
		Bit 6	Reserved = 0		
		Bit 5	Reserved = 0		
		Bit 4	Change function and behavior of DO1, DO2		C_DOPARAM
		Bit 3	Load comparison value 2		CMP_VAL2
		Bit 2	Load comparison value 1		CMP_VAL1
		Bit 1	Load counter in preparation		LOAD_PREPARE

Address	Assignment		
	Bit 0	Load counter directly	LOAD_VAL
Bytes 6 to 7	Reserved = 0 [1]		

[1] Not used for extended user interface

Notes on the control bits

Bit	Explanation
C_DOPARAM	Change function and behavior of DO1, DO2 The values from bytes 0 to 2 are applied as new function, hysteresis, and pulse duration of DO1, DO2. This may result in the following error: The conditions for the "Switch at comparison values" behavior are not fulfilled.
CMP_VAL1	Load comparison value 1 The value from bytes 0 to 3 is transferred to comparison value 1 with the control bit "Load comparison value 1" (CMP_VAL1).
CMP_VAL2	Load comparison value 2 The value from bytes 0 to 3 is transferred to comparison value 2 with the control bit "Load comparison value 2" (CMP_VAL2).
CTRL_DO1	Enable DO1 You use this bit to enable the DO1 output.
CTRL_DO2	Enable DO2 You use this bit to enable the DO2 output.
CTRL_SYN	You use this to enable synchronization.
EXTF_ACK	Error acknowledgment The error bits must be acknowledged with the EXTF_ACK control bit after the cause has been eliminated.
LOAD_PREPARE	Load counter in preparation The value from bytes 0 to 3 is applied as the load value.
LOAD_VAL	The value from bytes 0 to 3 is loaded directly as the new counter value.
RES_STS	Trigger status bits reset The status bits are reset by the acknowledgment process between the RES_STS bit and the RES_STS_A bit.
SET_DO1	Control bit DO1 Switches the DO1 digital output on and off when CRTL_DO1 is set.
SET_DO2	Control bit DO2 Switches the DO2 digital output on and off when CRTL_DO2 is set.
SW_GATE	Software gate control bit The software gate is opened/closed via the control interface with the SW_GATE bit.

Feedback interface for counting modes

Note

The following feedback interface data is consistent for 1Count5V:

- Bytes 0 ... 3
- Bytes 4 ... 7
- Bytes 8 ... 11 (extended user data interface)

Feedback interface assignment

The table below shows the assignment of the 1Count5V feedback interface (inputs) for counting modes.

Address	Assignment		Name
Bytes 0 to 3	Counter value or stored counter value in the case of the latch function on the digital input		
Byte 4	Bit 7	Short circuit of the encoder supply	ERR_24V
	Bit 6	Short circuit / wire break / overtemperature	ERR_DO1
	Bit 5	Parameter assignment error	ERR_PARA
	Bit 4	Short circuit / wire break / overtemperature	ERR_DO2
	Bit 3	Short circuit / wire break / encoder signal	ERR_ENCODER
	Bit 2	Resetting of status bits active	RES_STS_A
	Bit 1	Load function error	ERR_LOAD
	Bit 0	Load function is running	STS_LOAD
Byte 5	Bit 7	Down direction status	STS_C_DN
	Bit 6	Up direction status	STS_C_UP
	Bit 5	Reserved = 0	
	Bit 4	DO2 status	STS_DO2
	Bit 3	DO1 status	STS_DO1
	Bit 2	Reserved = 0	
	Bit 1	DI status	STS_DI
	Bit 0	Internal gate status	STS_GATE
Byte 6	Bit 7	Zero-crossing in the count range when counting without a main count direction	STS_ND
	Bit 6	Low count limit	STS_UFLW
	Bit 5	High count limit	STS_OFLW
	Bit 4	Comparator 2 status	STS_CMP2
	Bit 3	Comparator 1 status	STS_CMP1
	Bit 2	Reserved = 0	
	Bit 1	Reserved = 0	
	Bit 0	Synchronization status	STS_SYN
Byte 7	Reserved = 0		
Bytes 8 to 11	Counter value [1]		

[1] Extended user data interface

Notes on the Feedback Bits

Bit	Explanation
ERR_24V	Encoder supply short circuit The error bit must be acknowledged by the EXTF_ACKcontrol bit (Page 9301). Diagnostic alarm, if parameterized.
ERR_DO1	Short circuit/wire break/overtemperature due to overload at output DO1 The error bit must be acknowledged by the EXTF_ACKcontrol bit (Page 9301). Diagnostic alarm, if parameterized.
ERR_DO2	Short circuit/wire break/overtemperature at output DO2 The error bit must be acknowledged by the EXTF_ACKcontrol bit (Page 9301). Diagnostic alarm, if parameterized.
ERR_ENCODER	Short circuit / wire break in 5 V encoder signal The error bit must be acknowledged by the EXTF_ACKcontrol bit (Page 9301). Diagnostic alarm, if parameterized.
ERR_LOAD	Load function error The LOAD_VAL, LOAD_PREPARE, CMP_VAL1, CMP_VAL2 and C_DOPARAMcontrol bits (Page 9301) cannot be set simultaneously during transfer. As when you load an incorrect value (which is not accepted), this results in the ERR_LOAD status bit being set.
ERR_PARA	Errors in assigned module parameters.
RES_STS_A	Resetting of status bits active
STS_C_DN	Status down
STS_C_UP	Status up
STS_CMP1	Comparator 1 status The STS_CMP1 status bit indicates that the output is or was switched on. It must be acknowledged with the control bit (Page 9301)RES_STS. If the status bit is acknowledged when the output is still switched on, the bit is set again immediately. This bit is also set if the SET_DO1 control bit is used when DO1 is not enabled.
STS_CMP2	Comparator 2 status The STS_CMP2 status bit indicates that the output is or was switched on. It must be acknowledged with the control bit (Page 9301)RES_STS. If the status bit is acknowledged when the output is still switched on, the bit is set again immediately. This bit is also set if the SET_DO2 control bit is operated when DO2 is not enabled.
STS_DI	DI status The status of the DI is indicated in all modes by the STS_DI bit in the feedback interface.
STS_DO1	DO1 status The STS_DO1 status bit indicates the status of the DO1 digital output.
STS_DO2	DO2 status The STS_DO2 status bit indicates the status of the DO2 digital output.
STS_GATE	Internal gate status: Counting
STS_LOAD	Load function is running
STS_ND	Zero-crossing in the count range when counting without a main count direction. The bit must be reset by the control bit (Page 9301)RES_STS.

Bit	Explanation
STS_OFLW STS_UFLW	High count limit violated Low count limit violated Both bits must be reset.
STS_SYN	Synchronization status The STS_SYN bit is set after successful synchronization. It must be reset by the control bit (Page 9301)RES_STS.

Measurement parameters

Enabling group diagnostics

Group diagnostics

Channel-specific diagnostics will be carried out if you enable group diagnostics in your parameter assignment.

Selecting reaction to CPU/Master STOP

Setting response to CPU/Master STOP

You can configure the response of the 1Count5V to failure of the higher-level controller.

Parameter	Status of the 1Count5V at CPU/Master STOP	What happens when new parameters are assigned?
Turn off DO	The current mode is cancelled. The gate is closed and the digital output disabled. Comparison values 1 and 2 the load value are reset. The high and low limit, function and behavior of the digital outputs and the integration time are transferred from the parameters.	The changed parameters are accepted and become effective.
Continue working mode [1)]	The current mode continues working. The gate and digital output status does not change.	The gate is closed. The current mode is cancelled. The digital output is disabled. The changed parameters are accepted and become effective.

Parameter	Status of the 1Count5V at CPU/Master STOP	What happens when new parameters are assigned?
DO substitute a value	The current mode is cancelled. The gate is closed and the substitute value set for the digital output is activated. Comparison values 1 and 2 and the load value are reset. The high and low limit, function and behavior of the digital outputs and the integration time are transferred from the parameters. If "Pulse when comparison value is reached" has been set for the output, the only available pulse duration substitute value will be 1.	The changed parameters are accepted and become effective.
DO keep last value	The current mode is cancelled. The gate is closed and the status of the digital output is maintained. Comparison values 1 and 2 and the load value are reset. The high and low limit, function and behavior of the digital outputs and the integration time are transferred from the parameters.	The changed parameters are accepted and become effective.

1) The CPU/Master must not clear the outputs if the mode is to continue working on transition from CPU/Master STOP to RUN (startup).
Possible solution: In the part of the user program that is executed during startup, set the SW_GATE control bit and transfer the values to 1Count5V.

Exiting parameterized status

Under what conditions does the 1Count5V exit the parameterized status?

The CPU or master must be in RUN mode and you must make a change at the control interface (Page 9317).

Automatic new parameter assignment

New parameters are assigned to the ET 200S station by your CPU/ DP master:

- Upon CPU/DP master power on
- Upon IM 151-1 / IM 151-1 FO power on
- After failure of DP transmission
- Once altered parameters or configuration of the ET 200S station have been loaded to the CPU/DP master
- When the 1Count5V is inserted
- Upon power on or insertion of the corresponding power module

Measuring methods

Overview

Measuring methods

You can choose from the following measuring methods for the "Measuring procedure" parameter:

- With integration time (Page 9308)
- Continuous (Page 9308)

Measurement with integration time

Measurement with integration time

The measurement is carried out during the assigned integration time. When the integration time elapses, the measured value is updated.

The end of a measurement is indicated by the STS_CMP1status bit (Page 9319). This bit is reset by the control bit (Page 9317)RES_STS in the control interface.

If there were not at least two rising edges in the assigned integration time, 0 is returned as the measured value.

A value of -1 is returned up until the end of the first integration time.

You can change the integration time for the next measurement during operation.

Direction Reversal

If the direction of rotation is reversed during integration time, the measured value for this measurement period will be undefined. Evaluating the STS_C_UP and STS_C_DNfeedback bits (Page 9319) (direction evaluation) allows you to respond to any process irregularities.

See also

Integration Time and Update Time in Isochronous Mode (Page 9311)

Continuous measurement

Measuring Principle

The 1Count5V counts each positive edge of a pulse and assigns it a time value in µs.

The update time indicates the time interval at which the measured value is updated by the module in the feedback interface.

Rule for pulse sequences with one or several pulses per update time:

Dynamic measuring time =Time of last pulse in the current update time interval
minus
Time of last pulse in the previous update time interval

When the update time has elapsed, a new measured value is calculated and output with the dynamic measuring time.

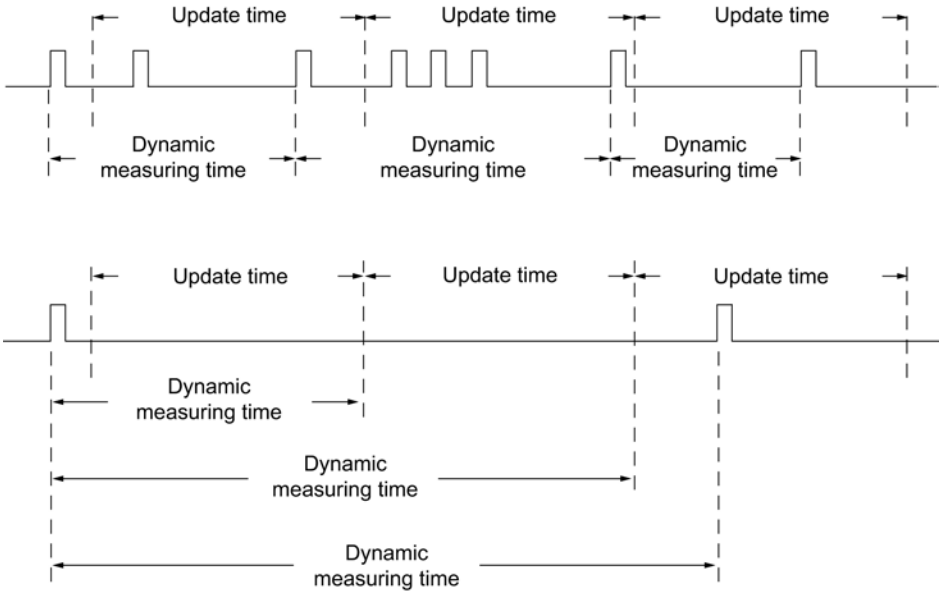
The following dynamic measuring time will result if the current update time does not contain a pulse:

Dynamic measuring time =Time of current, elapsed update time
minus
Time of last pulse

When the update time has elapsed, an estimated measured value is calculated with the dynamic measuring time under the assumption that a pulse occurred at the end of the update time.

If the "1 Pulse per dynamic measuring time" estimated measured value is less than the last measured value during the frequency and speed measurement, this estimated measured value is output as the new measured value. With the period measurement, the dynamic measuring time is output as the estimated period if the dynamic measuring time is greater than the last measured period.

The figure below shows the measuring principle.



1Count5V measures continuously. When assigning parameters, you specify an update time.

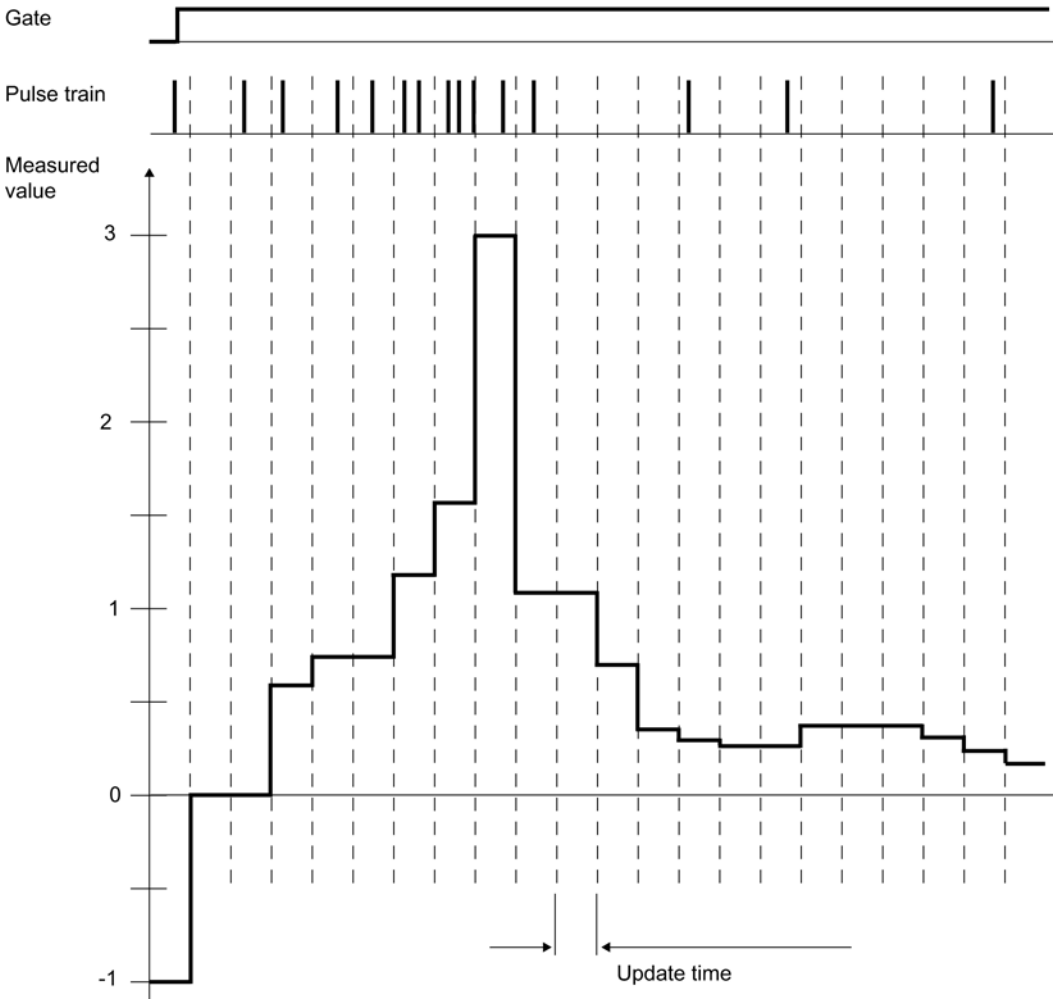
During the time until the end of the first elapsed update time, a value of "-1" is returned.

The continuous measurement begins after the gate is opened with the first pulse of the pulse train to be measured. The first measured value can be calculated after the second pulse, at the earliest.

A measured value (frequency, period, or speed) is output in the feedback interface each time the update time elapses. The end of a measurement is indicated by the STS_CMP1status bit (Page 9319). This bit is reset with the RES_STS and RES_STS_A bits by applying the complete acknowledgement principle.

If the direction of rotation is reversed during an update time, the measured value for this measurement period will be undefined. Evaluating the STS_C_DN and STS_C_UPfeedback bits (Page 9319) (direction evaluation) enables you to respond to any process irregularities.

The following figure illustrates the principle of continuous measurement taking frequency measurement as an example.



Gate Control

To control the 1Count5V, you have to use the gate functions.

Isochronous Mode

In isochrone mode, the 1Count5V accepts control bits and control values from the control interface in each bus cycle and reports back the response in the same cycle.

The 1Count5V transfers a measured value and the status bits that were valid at time T_i in each cycle.

The measurement starts and ends at time T_i.

See also

Integration Time and Update Time in Isochronous Mode (Page 9311)

Integration Time and Update Time in Isochronous Mode

Integration time and update time in isochrone mode

If the integration time/update time lasts several T_{DP} cycles, the new measured value will be indicated in the user program by the STS_CMP1status bit (Page 9319) (measurement completed) of the feedback interface. This enables monitoring of the measuring operation or a synchronization with the measuring operation. It takes 4 T_{DP} cycles, however, for this alarm to be acknowledged. The minimum integration time/update time in this case is (4 x T_{DP}).

You do not need to continually evaluate status bit STS_CMP1 if the application can tolerate a jitter in the integration time/update time of T_{DP} and a measured value that remains constant for several cycles. Integration times/update times of (1 x T_{DP}) to (3 x T_{DP}) are then possible.

A loss of isochrone operation in the last T_{DP} cycle of the integration time will increase the integration time/update time by one T_{DP} cycle. This does not corrupt the measured value.

Note

The value range limits for the integration time/update time may not be exceeded (see tables for the individual measuring modes).

Violation of the value range limits will result in a parameter assignment error and 1Count5V will not go into isochrone mode.

Note

You must always adjust the integration time/update time parameter if you want to retain the integration time/update time length when you change the configuration from non-isochrone to isochrone mode and vice versa.

Activating signal pair diagnostics

Signal pair monitoring

Signal pairs A and /A, and B and /B can be monitored.

Activate the check box.

Selecting the count direction

Count direction

To adapt the count direction to the process, you can select:

- Normal count direction:
- Inverted count direction

Behavior of the outputs in measuring modes

Selecting behavior of output DO1

Behavior of the Output in Measuring Modes

You can assign parameters for the digital output of the 1Count5V.

You can store a high and a low limit for frequency measurement, rotational speed measurement or period measurement. If the limits are violated, digital output DO1 is activated. These limit values can be assigned and changed with the load function.

You can change the function and the behavior of the digital output during operation. The new function takes effect immediately. In isochronous mode it always takes effect at time T_i.

You can choose between the following functions in the drop-down list box.

- Output (Page 9312)
- Outside limits (Page 9313) (limit monitoring)
- Violating low limit (Page 9313) (limit monitoring)
- Exceeding high limit (Page 9313) (limit monitoring)

Output

Output

If you want to switch the output on or off, you must enable it with the CTRL_DO1 or CTRL_DO2control bit (Page 9317).

You can switch the outputs on and off with the control bits (Page 9317)SET_DO1 and SET_DO2.

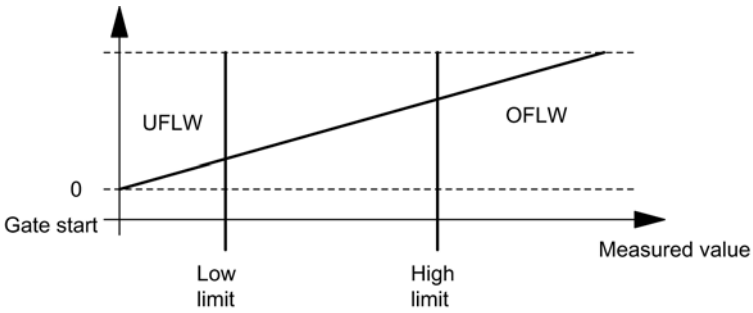
You can query the status of the output with the STS_DO1 or STS_DO2status bit (Page 9319) in the feedback interface.

The outputs are switched at time T_o in isochronous mode.

Limit-Value Monitoring

Limit monitoring

The figure below illustrates limit monitoring.



When the integration time has elapsed, the measured value obtained (frequency, rotational speed or period) is compared with the assigned limits.

The STS_UFLW = 1 bit will be set in the feedback (Page 9319) interface if the current measured value is below the parameterized low limit (measured value < low limit).

The STS_OFLW = 1 bit will be set in the feedback (Page 9319) interface if the current measured value is above the parameterized high limit (measured value > high limit).

You must acknowledge these bits with the control bit (Page 9317)RES_STS.

The corresponding status bit will be set again if the measured value still or once again violates the limits after acknowledgment.

Setting the low limit to 0 switches off dynamic monitoring of "Violating low limit".

The table below shows how the enabled digital output DO1 can be set by limit monitoring in accordance with parameter assignment:

"Function DO1" parameter	DO1 is Set ...
Outside limits	Measured value < low limit OR measured value > high limit
Violating low limit	Measured value < low limit
Exceeding high limit	Measured value > high limit

In isochronous mode the output is switched at the end of measurement at time T_i.

Selecting substitute value DOx

Substitute values DO1 and DO2

This is where you select the substitute value which is to be output at digital output DO1 or DO2 if you have selected the parameter "DO substitute a value" under "Reaction to CPU/Master STOP (Page 9306)".

Activating DOx diagnostics

DO1 and DO2 diagnostics

"DO1 diagnostics" or "DO2 diagnostics" (wire break, short circuit) are possible only with pulse lengths of > 90 ms at digital output DO1 or DO2.

Select the check box for diagnostics at output DO1 or DO2.

Selecting a measuring mode

Description

You define 1Count5V functionality by specifying a measuring mode.

	Description	
Operating mode	Measurement with integration time	Continuous measurement
Frequency measurement (Page 9093) (with hardware or software gate)	1Count5V counts the pulses that occur during a specified integration time.	1Count5V counts the pulses that occur during a dynamic measurement time.
Rotational speed measurement (Page 9094) (with hardware or software gate)	1Count5V counts the pulses received from a tachometer generator within a specified integration time and calculates the speed on the basis of this value and the number of pulses per encoder revolution.	1Count5V counts the pulses received from a tachometer generator within a dynamic measuring time and calculates the speed on the basis of this value and the number of pulses per encoder revolution.
Period measurement (Page 9096) (with hardware or software gate)	1Count5V measures the time between two positive edges of the count signal by counting the pulses of an internal quartz accurate reference frequency (16 MHz) over a specified integration time.	1Count5V indicates the dynamic measuring time as a period. The mean value of the period is calculated if its duration is less than the update time.

The default setting is "Frequency measurement" mode.

Selecting period resolution

Description

The value of the calculated period is displayed in 1 µs or 1/16 µs.

Select the required unit of resolution.

Possible Measuring Ranges with Error Indication

- Period measurement with integration time
  - Measuring ranges and errors at a resolution of 1 µs:

Integration time	T_{min} ± absolute error	T_{min} ± absolute error
100 s	1 µs* (10 ± 0)	1 µs* (100,000,000 ± 10,000)
10 s	1 µs* (10 ± 0)	1 µs* (10,000,000 ± 1,000)
1 s	1 µs* (10 ± 0)	1 µs* (1,000,000 ± 100)
0.1 s	1 µs* (10 ± 0)	1 µs* (100,000 ± 10)
0.01 s	1 µs* (10 ± 0)	1 µs* (10,000 ± 1)

  - Measuring ranges and errors at a resolution of 1/16 µs:

Integration time	T_{min} ± absolute error	T_{min} ± absolute error
100 s	1/16 µs* (160 ± 0)	1/16 µs* (1,600,000,000 ± 160,000)
10 s	1/16 µs* (160 ± 0)	1/16 µs* (160,000,000 ± 16,000)
1 s	1/16 µs* (160 ± 0)	1/16 µs* (16,000,000 ± 1,600)
0.1 s	1/16 µs* (160 ± 0)	1/16 µs* (1,600,000 ± 160)
0.01 s	1/16 µs* (160 ± 0)	1/16 µs* (160,000 ± 16)

- Continuous period measurement:
  - Measuring ranges and errors at a resolution of 1 µs:

Period T_{min} ± Absolute error	Period T_{min} ± Absolute error
1 µs* (10 ± 0)	1 µs* (100,000 ± 10)
1 µs* (100 ± 0)	1 µs* (1 000 000 ± 100)
1 µs* (1 000 ± 0)	1 µs* (10 000 000 ± 1 002)
1 µs* (10 000 ± 1)	1 µs* (100,000,000 ± 10,020)

  - Measuring ranges and errors at a resolution of 1/16 µs:

Period T_{min} ± Absolute error	Period T_{min} ± Absolute error
1/16 µs* (160 ± 1)	1/16 µs* (1,600,000 ± 160)
1/16 µs* (1,600 ± 1)	1/16 µs* (16,000,000 ± 1,600)
1/16 µs* (16 000 ± 3)	1/16 µs* (160 000 000 ± 16 000)
1/16 µs* (160 000 ± 20)	1/16 µs* (1,600,000,000 ± 160,000)

Selecting DI function

Description

You select the digital input function with "DI function":

- Input
  The digital input operates as a "normal" input.
- Hardware gate (Page 9102), level-controlled
  The hardware gate is opened and closed by signals at the digital input.

The default setting is "Input".

Selecting hardware gate input signal

Hardware gate input signal

The level of the digital input for the "Function DI = HW gate" configuration with the parameter "Input signal HW gate" can be inverted.

The feedback bit (Page 9319)STS_DI indicates the level of the digital input.

Setting low and high limit

Low and high limit

You can save a high and low limit for frequency measurement, speed measurement and period measurement. If these limits are exceeded, digital output DO1 (Page 9312) will be activated. You can configure these limits and change them using the load function.

The following value ranges are possible for the low and high limits:

Measuring mode	Low limit (LL)	High limit
Frequency measurement	0 to 99,999,999 × 10^{-3} Hz	LL+1 to 100,000,000 × 10^{-3} Hz
Rotational speed measurement	0 to 24,999,999 × 10^{-3} /min	LL+1 to 25,000,000 × 10^{-3} /min
Period measurement (resolution = 1 µs)	0 to 119,999,999 µs	LL+1 to 120,000,000 µs
Period measurement (resolution = 1/16 µs)	0 to 1,919,999,999 µs	LL+1 to 1,920,000,000 µs

Specifying measuring time

Description

If you have selected a measuring mode, the module will update the measured values on a cyclic basis. The update time/integration time is the product of measuring time x 10ms. You set the measuring time as the factor n.

Detailed descriptions of the value ranges for the measuring time can be found under "Measuring basics (Page 9088)".

Entering pulses per encoder revolution

Pulses per encoder revolution

For "Rotational speed measurement (Page 9094)" mode, you must also configure the pulses per encoder revolution.

The value for the number of pulses per encoder revolution can be found on the rating plate, or in the technical data of your encoder.

Control and feedback interface for measuring modes

Control interface for measuring modes

Note

The following control interface data is consistent for 1Count5V:

- Bytes 0 ... 3
- Bytes 4 ... 7
- Bytes 8 ... 11 (extended user data interface)

Control interface assignment

The table below shows the assignment of the 1Count5V control interface (outputs) for measuring modes.

Address	Assignment			
Bytes 0 to 3	**Low or high limit**			
	Function of DO1			
	Byte 0	Bit 1	Bit 0	Function DO1
		0	0	Output
		0	1	Outside limits
		1	0	Violating low limit
		1	1	Exceeding high limit
	Bytes 1 to 3	Reserved = 0		
	Integration time/update time			
	Bytes 0, 1	Integration time [n × 10 ms] (Value range 1 to 1000/12000)		
	Bytes 2, 3	Reserved = 0		
				Name
Byte 4	Bit 7	Error acknowledgment		EXTF_ACK
	Bit 6	Enable DO2		CTRL_DO2
	Bit 5	Control bit DO2		SET_DO2
	Bit 4	Enable DO1		CTRL_DO1
	Bit 3	Control bit DO1		SET_DO1
	Bit 2	Start resetting of status bit		RES_STS
	Bit 1	Reserved = 0		
	Bit 0	SW gate control bit		SW_GATE
Byte 5	Bit 7	Reserved = 0		
	Bit 6	Reserved = 0		

Address	Assignment		
	Bit 5	Reserved = 0	
	Bit 4	Change function of DO1	C_DOPARAM
	Bit 3	Reserved = 0	
	Bit 2	Change integration time	C_INTTIME
	Bit 1	Load high limit	LOAD_PREPARE
	Bit 0	Load low limit	LOAD_VAL
Bytes 6 to 7	Reserved = 0 [1]		

[1] Not used for extended user interface

Notes on the Control Bits

Bit	Explanation
C_DOPARAM	Change function of DO1 The value from byte 0 is adopted as the new function of DO1.
C_INTTIME	Change integration time The value from bytes 0 and 1 is adopted as the new integration time for the next measurement.
CTRL_DO1	Enable DO1 You use this bit to enable the DO1 output.
CTRL_DO2	Enable DO2 You use this bit to enable the DO2 output.
EXTF_ACK	Error acknowledgment The error bits must be acknowledged with the EXTF_ACK control bit after the cause has been eliminated.
LOAD_PREPARE	Load high limit The value from bytes 0 to 3 is applied as the new high limit.
LOAD_VAL	Load low limit The value from bytes 0 to 3 is applied as the new low limit.
RES_STS	Trigger status bits reset The status bits are reset by the acknowledgment process between the RES_STS bit and the RES_STS_A bit.
SET_DO1	Control bit DO1 Switches the DO1 digital output on and off when CRTL_DO1 is set.
SET_DO2	Control bit DO2 Switches the DO2 digital output on and off when CRTL_DO2 is set.
SW_GATE	Software gate control bit The software gate is opened/closed via the control interface with the SW_GATE bit.

Feedback interface for measuring modes

Note

The following feedback interface data is consistent for 1Count5V:

- Bytes 0 ... 3
- Bytes 4 ... 7
- Bytes 8 ... 11 (extended user data interface)

Feedback interface assignment

The table below shows the assignment of the 1Count5V feedback interface (inputs) for measuring modes.

Address	Assignment		Name
Bytes 0 to 3	Measured value		
Byte 4	Bit 7	Encoder supply short circuit	ERR_24V
	Bit 6	Short circuit / wire break / overtemperature	ERR_DO1
	Bit 5	Parameter assignment error	ERR_PARA
	Bit 4	Short circuit / wire break / overtemperature	ERR_DO2
	Bit 3	Short circuit / wire break / encoder signal	ERR_ENCODER
	Bit 2	Resetting of status bits active	RES_STS_A
	Bit 1	Load function error	ERR_LOAD
	Bit 0	Load function is running	STS_LOAD
Byte 5	Bit 7	Status down	STS_C_DN
	Bit 6	Status up	STS_C_UP
	Bit 5	Reserved = 0	
	Bit 4	DO2 status	STS_DO2
	Bit 3	DO1 status	STS_DO1
	Bit 2	Reserved = 0	
	Bit 1	DI status	STS_DI
	Bit 0	Internal gate status	STS_GATE
Byte 6	Bit 70	Reserved = 0	
	Bit 6	Low limit, measuring range	STS_UFLW
	Bit 5	High limit, measuring range	STS_OFLW
	Bit 4	Reserved = 0	
	Bit 3	Measurement completed	STS_CMP1
	Bit 2	Reserved = 0	
	Bit 1	Reserved = 0	
	Bit 0	Reserved = 0	
Byte 7	Reserved = 0		
Bytes 8 to 11	Counter value [1]		

[1] Extended user data interface

Notes on the feedback bits

Bit	Explanation
ERR_24V	Encoder supply short circuit The error bit must be acknowledged with the control bit (Page 9317)EXTF_ACK. Diagnostic alarm if parameterized.
ERR_DO1	Short circuit/wire break/overtemperature at output DO1 The error bit must be acknowledged with the control bit (Page 9317)EXTF_ACK. Diagnostic alarm if parameterized.
ERR_DO2	Short circuit/wire break/overtemperature at output DO2 The error bit must be acknowledged with the control bit (Page 9317)EXTF_ACK. Diagnostic alarm if parameterized.
ERR_ENCODER	Short circuit / wire break / encoder signal The error bit must be acknowledged with the control bit (Page 9317)EXTF_ACK Diagnostic alarm if parameterized.
ERR_LOAD	Load function error The LOAD_VAL, LOAD_PREPARE, C_DOPARAM and C_INTTIMEcontrol bits (Page 9317) cannot be set simultaneously during transfer. As when you load an incorrect value (which is not accepted), this results in the ERR_LOAD status bit being set.
ERR_PARA	Errors in assigned module parameters.
RES_STS_A	Resetting of status bits active
STS_C_DN	Status down
STS_C_UP	Status up
STS_CMP1	Measurement completed The measured value is updated each time a time interval (update time/integration time) elapses. **Measurement with integration time:** The end of a measurement (after the interval has elapsed) is indicated by the STS_CMP1 status bit. **Continuous measurement:** The end of the measurement is signaled by the status bit STS_CMP1 at the end of the update time if a measured value is output. The bit remains at "0" if an estimated measured value is output. This bit is reset by the control bit (Page 9317)RES_STS in the control interface.
STS_DI	DI status The status of the DI is indicated in all modes by the STS_DI bit in the feedback interface.
STS_DO1	DO1 status
STS_DO2	DO2 status
STS_GATE	Internal gate status: Measuring.
STS_LOAD	Load function is running
STS_OFLW STS_UFLW	High limit violated Low limit violated Both bits must be reset.

Position feedback parameters

Enabling group diagnostics

Group diagnostics

Channel-specific diagnostics will be carried out if you enable group diagnostics in your parameter assignment.

Selecting reaction to CPU/Master STOP

Setting response to CPU/Master STOP

You can configure the response of the 1Count5V to failure of the higher-level controller.

Parameter	Status of the 1Count5V at CPU/Master STOP	What happens when new parameters are assigned?
Turn off DO	The current mode is cancelled. The gate is closed and the digital output disabled. Comparison values 1 and 2 the load value are reset. The high and low limit, function and behavior of the digital outputs and the integration time are transferred from the parameters.	The changed parameters are accepted and become effective.
Continue working mode [1]	The current mode continues working. The gate and digital output status does not change.	The gate is closed. The current mode is cancelled. The digital output is disabled. The changed parameters are accepted and become effective.

[1] The CPU/Master must not clear the outputs if the mode is to continue working on transition from CPU/Master STOP to RUN (startup).
Possible solution: In the part of the user program that is executed during startup, set the SW_GATE control bit and transfer the values to 1Count5V.

Leaving the Assigned State

Under what conditions does the 1Count5V exit the parameterized status?

The CPU or master must be in RUN mode and you must make a change at the control interface (Page 9328).

Automatic New Parameter Assignment

New parameters are assigned to the ET 200S station by your CPU/ DP master:

- Upon power on of the CPU/DP master
- Upon IM 151-1 / IM 151-1 FO power on
- After failure of DP transmission

- Once altered parameters or configuration of the ET 200S station have been loaded to the CPU/DP master
- When the 1Count5V is inserted
- Upon power on or inserting of the appropriate power module

Selecting signal evaluation

Signal Evaluation A, B

Signal evaluation by means of A, B allows you to count directionally. Different evaluation modes are possible depending on the parameter assignment:

- Rotary Encoder (Page 9108)
  The 1Count5V can count the edges of the signals. Normally, only the edges at A are evaluated (single evaluation). To obtain a higher resolution, you can select whether the signals are to undergo single (Page 9108), double (Page 9108) or quadruple (Page 9109) evaluation when assigning parameters ("Signal evaluation" parameter).
  Multiple evaluation is only possible with asymmetric incremental encoders with A and B signals that are 90 degrees out of phase.

  Note

  500 kHz for count frequency is the maximum frequency of the A and B signals. This means a maximum frequency of 1 MHz for the count pulses in double evaluation and a maximum of 2 MHz in quadruple evaluation.

Activating signal pair diagnostics

Signal pair monitoring

Signal pairs A and /A, B and /B, and N and /N can be monitored.

Activate the corresponding check box.

Note

You cannot activate "N, /N diagnostics" until you have activated "A, /A and B, /B diagnostics".

Selecting the count direction

Count direction

To adapt the count direction to the process, you can select:

- Normal count direction:
- Inverted count direction

Selecting the gate function

Description

When configuring the hardware and software gates, you can specify whether the internal gate is to cancel or interrupt counting.

- Cancel counting (Page 9103): The counting process restarts from the beginning after gate stop and restart.
- Interrupt counting (Page 9103): The counting process starts from the last current counter value after gate stop and restart.

DI function

Selecting DI function

Description

You select the digital input function with "DI function":

- Input
  The digital input operates as a "normal" input.
- Hardware gate (Page 9102), level-controlled
  The hardware gate is opened and closed by signals at the digital input.
- Latch/Retrigger with positive edge (Page 9323)
  Counter values are saved (latched) with positive edges at the digital input. The counter is set to the load value each time the values are saved.
- Synchronization with positive edge (Page 9324)
  The counter is set to the load value with positive edges at the digital input.
- Latching with positive edge (Page 9326)
  Counter values are saved (latched) with positive edges at the digital input.
- Hardware enable for synchronization (Page 9327)
  The digital input acts as a hardware enable. The encoder zero mark sets the counter to the load value when the hardware enable is active.

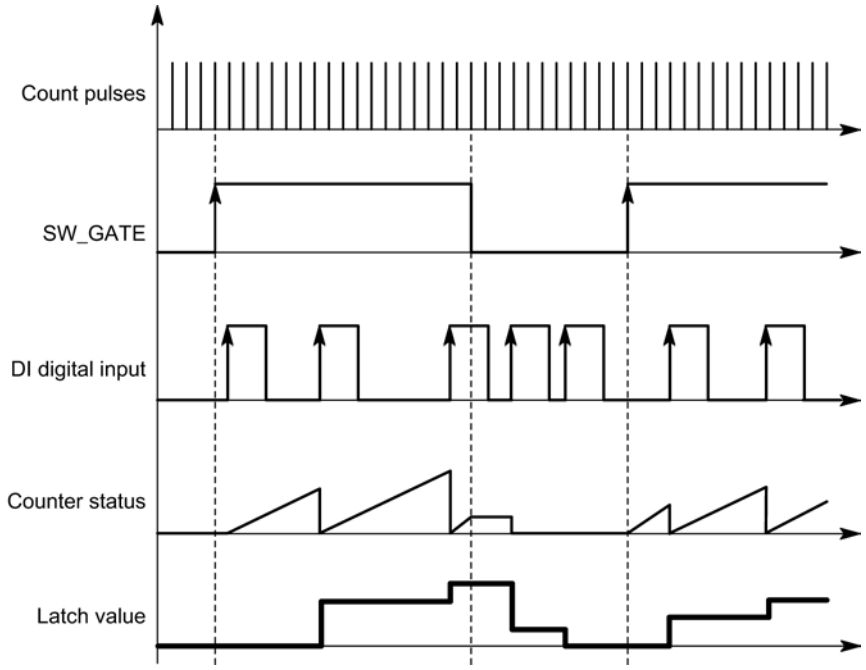
The default setting is "Input".

Latch/Retrigger with positive edge

Latch/Retrigger

This function saves the current internal counter value of the 1Count5V and retriggers counting when there is a positive edge at the digital input. This means that the current internal counter value at the time of the positive edge is stored (latch value), and the 1Count5V is then loaded again with the load value, from which counting resumes.

The following figure shows latch/retrigger when load value = 0.



The counting mode must be enabled with the SW gate before the function can be executed. It is started with the first positive edge on the digital input.

The saved rather than the current counter value is displayed in the feedback interface (Page 9329). The STS_DI bit indicates the status of the latch and retrigger signal.

The latch value is preassigned its RESET status ("0"). It is not changed when the SW gate is opened.

Direct loading of the counter does not cause the indicated stored counter status to be changed.

If you close the SW gate, counting is only interrupted; this means that when you open the SW gate again, counting is continued. The digital input DI remains active even when the SW gate is closed.

Counting is also latched and triggered in isochronous mode with each edge on the digital input. The counter status that was valid at the time of the last edge before T_i is displayed in the feedback interface.

Extended feedback interface

If the 1Count5V is inserted behind an IM 151 that supports the reading and writing of broader user data interfaces, the current count value can be read from bytes 8 to 11 of the feedback interface (Page 9329).

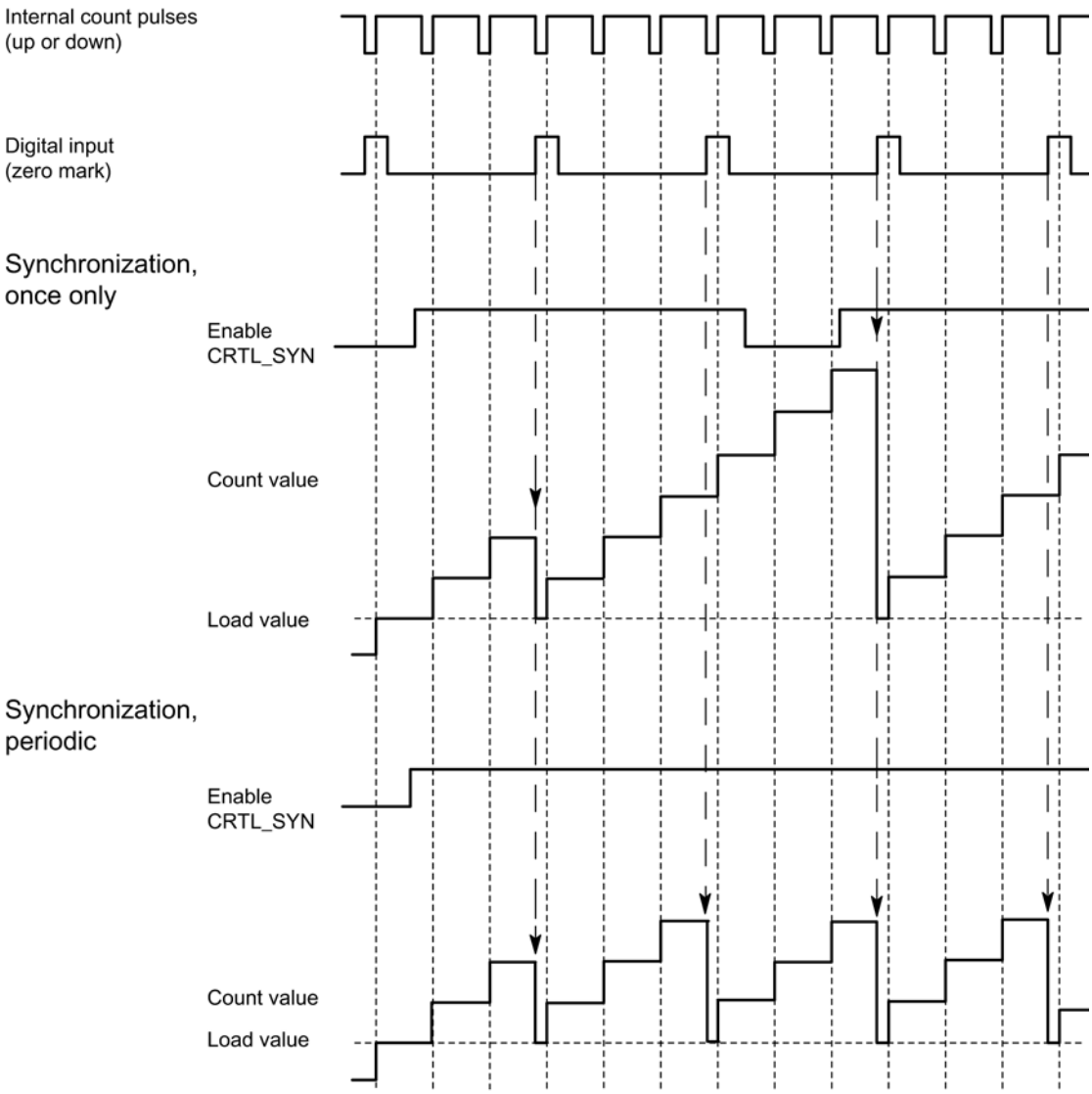
Synchronization with positive edge

Synchronization

If you have configured "Synchronization with positive edge", the rising edge of a reference signal at the input will set 1Count5V to the load value.

You can choose between once only and periodic synchronization ("Synchronization" parameter).

The figure below shows the counter values in once only and periodic synchronization.



The following conditions apply:

- The counting mode must have been started with the software gate.
- The "Enable synchronization" (CTRL_SYN) control bit (Page 9328) must be set.
- Synchronize once only: If the enable bit is set, the first edge loads 1Count5V with the load value.
- Synchronization periodically: If the enable bit is set, the first and each subsequent edge loads the 1Count5V with the load value.
- After successful synchronization, the STS_SYNfeedback bit (Page 9329) is set and the SYN LED comes on. The RES_STScontrol bit (Page 9328) resets the feedback bit and switches off the LED.

- The signal of a bounce-free switch can serve as the reference signal.
- The feedback bit (Page 9329)STS_DI indicates the level of the reference signal.

In isochrone mode, the set feedback bit (Page 9329)STS_SYN indicates that the rising edge at the digital input was between time T_i in the current cycle and time T_i in the previous cycle.

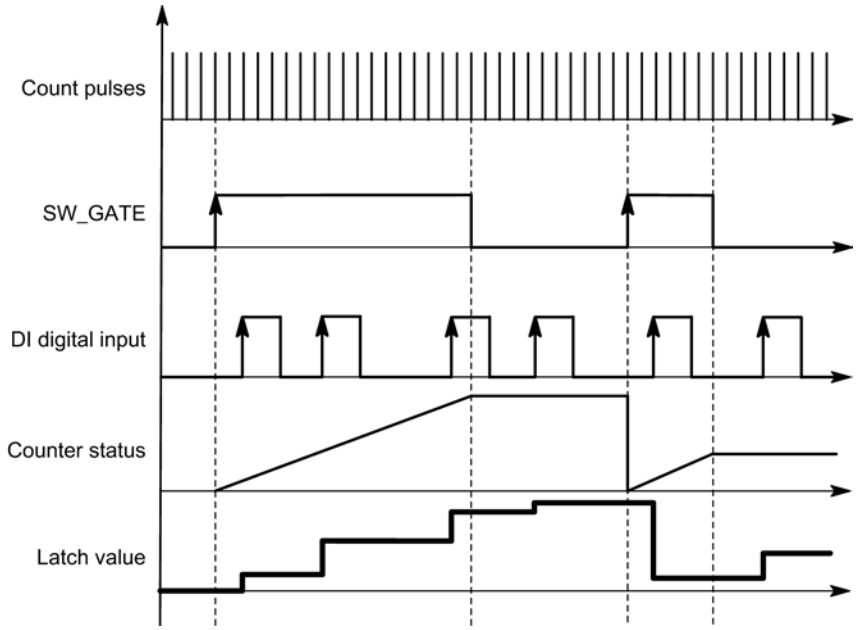
Latching with positive edge

Latching

Counter value and latch value are pre-assigned their RESET statuses ("0").

The counting function is started when the SW gate is opened. 1Count5V begins at the load value ("0").

The following figure shows latching when load value = 0.



The latch value is always the exact counter status at the time of the positive edge on the digital input DI.

The saved rather than the current counter value is displayed in the feedback interface (Page 9329). The STS_DI bit indicates the level of the latch signal.

Directly loading the counter does not alter the saved counter value displayed.

In isochronous mode, the counter status that was latched at the time of the last positive edge before T_i is displayed in the feedback interface.

When you close the SW gate, the effect is either canceling or interrupting, depending on the parameter assignment. The digital input DI remains active even when the software gate is closed.

Extended feedback interface

If the 1Count5V is inserted behind an IM 151 that supports the reading and writing of broader user data interfaces, the current count value can be read from bytes 8 to 11 of the feedback interface (Page 9329).

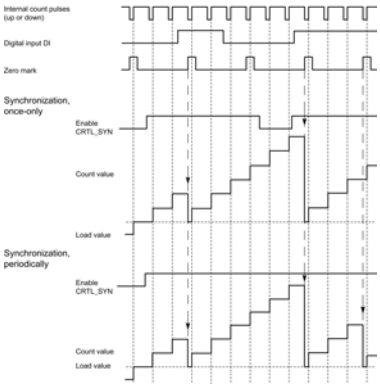
Hardware enable for synchronization

Synchronization with DI and Zero Mark

The DI will act as the hardware enable if you have configured "Hardware enable for synchronization". When the HW enable is active, the 1Count5V is loaded with the load value by the zero mark of the encoder.

You can choose between once only and periodic synchronization ("Synchronization (Page 9328)" parameter).

The figure below shows the counter values in once only and periodic synchronization.



The following conditions apply:

- The counting mode must have been started with the software gate.
- The "Enable synchronization" (CTRL_SYN) control bit (Page 9328) must be set.
- Synchronize once only: If the enable bit is set and the HW enable is present, the first zero mark loads the 1Count5V with the load value.
- Synchronization periodically: If the enable bit is set and the HW enable is present, the first and each subsequent zero mark will load 1Count5V with the load value.
- After successful synchronization, the STS_SYNfeedback bit (Page 9329) is set and the SYN LED comes on. The RES_STScontrol bit (Page 9328) resets the feedback bit and switches off the LED.
- The signal of a bounce-free switch can serve as the reference signal.
- The feedback bit (Page 9329)STS_DI indicates the level of the reference signal.

In isochrone mode, the set feedback bit (Page 9329)STS_SYN indicates that the positive edge on the digital input was between time T_i in the current cycle and time T_i in the previous cycle.

Selecting hardware gate input signal

Hardware gate input signal

The level of the digital input for the "Function DI = HW gate" configuration with the parameter "Input signal HW gate" can be inverted.

The feedback bit (Page 9329)STS_DI indicates the level of the digital input.

Selecting synchronization

Synchronization

This is where you can select either once only or periodic synchronization if you have selected "Synchronization with positive edge" at the "DI function (Page 9323)" parameter:

- Once: If the enable bit is set, the first edge loads the 1Count5V with the load value.
- Periodic: If the enable bit is set, the first and each subsequent edge loads 1Count5V with the load value.

Control and feedback interface for position feedback

Position feedback control interface

Note

The following control interface data is consistent for 1Count5V:

- Bytes 0 ... 3
- Bytes 4 ... 7
- Bytes 8 ... 11 (extended user data interface)

Control interface assignment

The table below shows the assignment of the 1Count5V control interface (outputs) for position feedback.

Address	Assignment		Name
Bytes 0 to 3	Load value directly, in preparation		
Byte 4	Bit 7	Error acknowledgment	EXTF_ACK
	Bit 6	Reserved = 0	
	Bit 5	Reserved = 0	
	Bit 4	Reserved = 0	
	Bit 3	Reserved = 0	
	Bit 2	Trigger status bits reset	RES_STS

Address	Assignment		Name
	Bit 1	Enable synchronization	CTRL_SYN
	Bit 0	Software gate control bit	SW_GATE
Byte 5	Bit 7	Reserved = 0	
	Bit 6	Reserved = 0	
	Bit 5	Reserved = 0	
	Bit 4	Reserved = 0	
	Bit 3	Reserved = 0	
	Bit 2	Reserved = 0	
	Bit 1	Load counter in preparation	LOAD_PREPARE
	Bit 0	Load counter direct	LOAD_VAL
Bytes 6 to 7	Reserved = 0 [1]		

[1] Not used for extended user interface

Notes on the control bits

Bit	Explanation
CTRL_SYN	You use this bit to enable synchronization.
EXTF_ACK	Error acknowledgment The error bits must be acknowledged with the EXTF_ACK control bit after the cause has been eliminated.
LOAD_PREPARE	Load counter in preparation The value from bytes 0 to 3 is applied as the load value.
LOAD_VAL	The value from bytes 0 to 3 is loaded directly as the new count value.
RES_STS	Trigger status bits reset The status bits are reset by the acknowledgment process between the RES_STS bit and the RES_STS_A bit.
SW_GATE	Software gate control bit The software gate is opened/closed via the control interface with the SW_GATE bit.

Position feedback feedback interface

Note

The following feedback interface data is consistent for 1Count5V:

- Bytes 0 ... 3
- Bytes 4 ... 7
- Bytes 8 ... 11 (extended user data interface)

Feedback interface assignment

The table below shows the assignment of the 1Count5V feedback interface (inputs) for position feedback.

Address	Assignment		Name
Bytes 0 to 3	Counter value or stored counter value in the case of the latch function at the digital input		
Byte 4	Bit 7	Encoder supply short circuit	ERR_24V
	Bit 6	Reserved = 0	
	Bit 5	Parameter assignment error	ERR_PARA
	Bit 4	Reserved = 0	
	Bit 3	Reserved = 0	
	Bit 2	Resetting of status bits active	RES_STS_A
	Bit 1	Load function error	ERR_LOAD
	Bit 0	Load function is running	STS_LOAD
Byte 5	Bit 7	Status down	STS_C_DN
	Bit 6	Status up	STS_C_UP
	Bit 5	Reserved = 0	
	Bit 4	Reserved = 0	
	Bit 3	Reserved = 0	
	Bit 2	Reserved = 0	
	Bit 1	DI status	STS_DI
	Bit 0	Internal gate status	STS_GATE
Byte 6	Bit 7	Zero crossing	STS_ND
	Bit 6	Low count limit	STS_UFLW
	Bit 5	High count limit	STS_OFLW
	Bit 4	Reserved = 0	
	Bit 3	Reserved = 0	
	Bit 2	Reserved = 0	
	Bit 1	Reserved = 0	
	Bit 0	Synchronization status	STS_SYN
Byte 7	Reserved = 0		
Bytes 8 to 11	Counter value [1]		

[1] Extended user data interface

Notes on the feedback bits

Bit	Explanation
ERR_24V	Encoder supply short circuit The error bit must be acknowledged with the control bit (Page 9328)EXTF_ACK. Diagnostic alarm if parameterized.
ERR_LOAD	Load function error The LOAD_VAL and LOAD_PREPAREcontrol bits (Page 9328) cannot be set simultaneously during transfer. As when you load an incorrect value (which is not accepted), this results in the ERR_LOAD status bit being set.
ERR_PARA	Errors in assigned module parameters.
RES_STS_A	Resetting of status bits active
STS_C_DN	Status down
STS_C_UP	Status up
STS_DI	DI status The status of the DI is indicated in all modes by the STS_DI bit in the feedback interface.
STS_GATE	Internal gate status: Counting.
STS_LOAD	Load function is running
STS_ND	Zero crossing in the count range during counting without a main count direction. The bit must be reset by the control bit (Page 9328)RES_STS.
STS_OFLW STS_UFLW	High count limit violated Low count limit violated Both bits must be reset.
STS_SYN	Synchronization status The STS_SYN bit is set after successful synchronization. It must be reset by the control bit (Page 9328)RES_STS.

Fast mode parameters

Selecting reaction to CPU/Master STOP

Setting response to CPU/Master STOP

You can configure the response of the 1Count5V to failure of the higher-level controller.

Parameter	Status of the 1Count5V at CPU/Master STOP	What happens when new parameters are assigned?
Turn off DO	The current mode is cancelled. The gate is closed and the digital output disabled. Comparison values 1 and 2 the load value are reset. The high and low limit, function and behavior of the digital outputs and the integration time are transferred from the parameters.	The changed parameters are accepted and become effective.
Continue working mode [1]	The current mode continues working. The gate and digital output status does not change.	The gate is closed. The current mode is cancelled. The digital output is disabled. The changed parameters are accepted and become effective.

[1] The CPU/Master must not clear the outputs if the mode is to continue working on transition from CPU/Master STOP to RUN (startup).
Possible solution: In the part of the user program that is executed during startup, set the SW_GATE control bit and transfer the values to 1Count5V.

Automatic new parameter assignment

New parameters are assigned to the ET 200S station by your CPU/ DP master:

- Upon power on of the CPU/DP master
- Upon IM 151-1 / IM 151-1 FO power on
- After failure of DP transmission
- Once altered parameters or configuration of the ET 200S station have been loaded to the CPU/DP master
- When the 1Count5V is inserted
- Upon power on or inserting of the appropriate power module

Selecting signal evaluation

Signal Evaluation A, B

Signal evaluation by means of A, B allows you to count directionally. Different evaluation modes are possible depending on the parameter assignment:

- Rotary Encoder (Page 9108)
  The 1Count5V can count the edges of the signals. Normally, only the edges at A are evaluated (single evaluation). To obtain a higher resolution, you can select whether the signals are to undergo single (Page 9108), double (Page 9108) or quadruple (Page 9109) evaluation when assigning parameters ("Signal evaluation" parameter).
  Multiple evaluation is only possible with asymmetric incremental encoders with A and B signals that are 90 degrees out of phase.

Note

500 kHz for count frequency is the maximum frequency of the A and B signals. This means a maximum frequency of 1 MHz for the count pulses in double evaluation and a maximum of 2 MHz in quadruple evaluation.

Activating signal pair diagnostics

Signal pair monitoring

Signal pairs A and /A, B and /B, and N and /N can be monitored.

Activate the corresponding check box.

Note

You cannot activate "N, /N diagnostics" until you have activated "A, /A and B, /B diagnostics".

Selecting the count direction

Count direction

To adapt the count direction to the process, you can select:

- Normal count direction:
- Inverted count direction

Selecting the gate function

Description

When configuring the hardware and software gates, you can specify whether the internal gate is to cancel or interrupt counting.

- Cancel counting (Page 9103): The counting process restarts from the beginning after gate stop and restart.
- Interrupt counting (Page 9103): The counting process starts from the last current counter value after gate stop and restart.

DI function

Selecting DI function

Description

You select the digital input function with "DI function":

- Input
  The digital input operates as a "normal" input.
- Hardware gate (Page 9102), level-controlled
  The hardware gate is opened and closed by signals at the digital input.
- Synchronization with positive edge (Page 9334)
  The counter is set to the load value with positive edges at the digital input.
- Hardware enable for synchronization (Page 9335)
  The digital input acts as a hardware enable. The encoder zero mark sets the counter to the load value when the hardware enable is active.

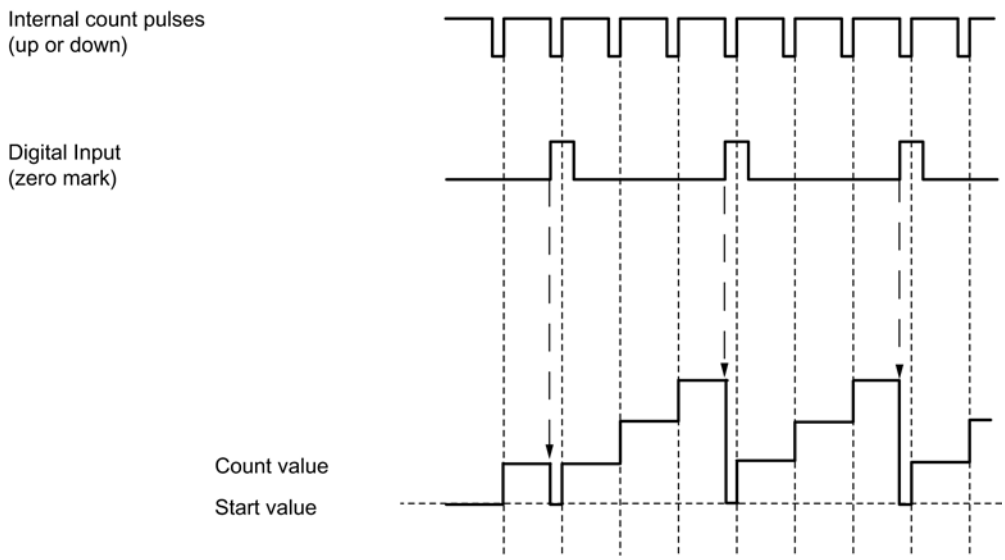
The default setting is "Input".

Synchronization with positive edge

Synchronization

If you have configured "Synchronization with positive edge", the positive edge of a reference signal at the input will set 1Count5V to the initial value (= load value).

The figure below shows the counter values for synchronization with a positive edge.



The following conditions apply:

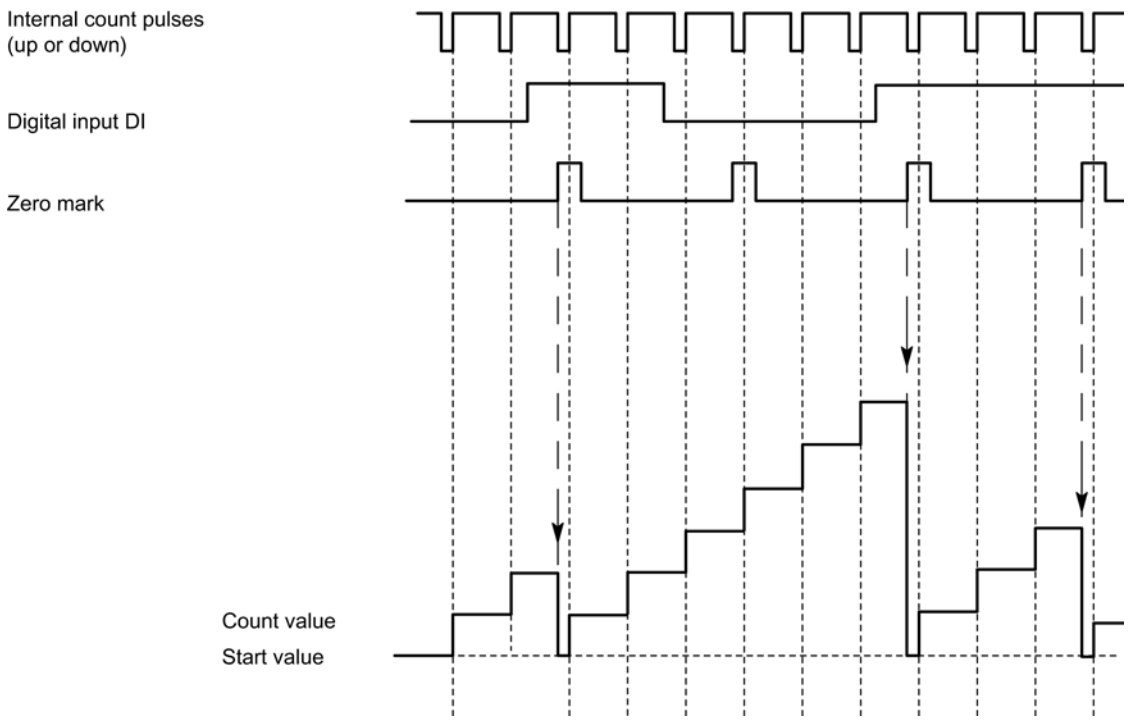
- "Fast mode" must be active (hardware gate).
  - When synchronization is activated, the first edge and each additional edge loads 1Count5V with the initial value (=load value).
- The signal of a bounce-free switch or the zero mark of a rotary encoder can serve as the reference signal.
- The STS_DI feedback bit indicates the level of the reference signal.

Hardware enable for synchronization

Synchronization with DI and Zero Mark

The DI will act as the hardware enable if you have configured "Hardware enable for synchronization". When the HW enable is active, 1Count5V is loaded with the initial value (load value) by the zero mark of the encoder.

The figure below shows the counter status following hardware enable at the DI and synchronization with a positive edge.



The following conditions apply:

- The counting mode must have been started with the SW gate.
- The "Enable synchronization" (CTRL_SYN) control bit must be set.
- After successful synchronization, the STS_SYNfeedback bit (Page 9337) is set and the SYN LED comes on. The RES_STS control bit resets the feedback bit and switches the LED off.
- The signal of a bounce-free switch can serve as the reference signal.
- The feedback bit (Page 9337)STS_DI indicates the level of the reference signal.

In isochronous mode, the set STS_SYN feedback bit indicates that the positive edge on the digital input was between time T_i in the current cycle and time T_i in the previous cycle.

Selecting hardware gate input signal

Hardware gate input signal

The level of the digital input for the "Function DI = HW gate" configuration with the parameter "Input signal HW gate" can be inverted.

The feedback bit (Page 9337)STS_DI indicates the level of the digital input.

Specifying load value

Load value

You can specify a load value for 1Count5V.

This load value is applied directly as the initial value.

Valid value range: -16777216 to +16777215

Fast mode feedback interface

Note

The following feedback interface data is consistent for 1Count5V:

- Bytes 0 ... 3

Feedback interface assignment

The table below shows the assignment of the 1Count5V feedback interface (inputs) for fast mode.

Address	Assignment		Name
Bytes 0 to 3	Bit 31	Lifebeat	LZ
	Bit 30	Isochronous mode applied	STS_TIC
	Bit 29	Parameter Assignment Error	ERR_PARA
	Bit 28	Group error • Encoder supply short circuit • Short circuit / wire break / encoder signal	EXTF
	Bit 27	DI status	STS_DI
	Bit 26	Status of direction up/down	STS_DIR
	Bit 25	Internal gate status	STS_GATE
	Bits 0 to 24	Count value	

Notes on the Feedback Bits

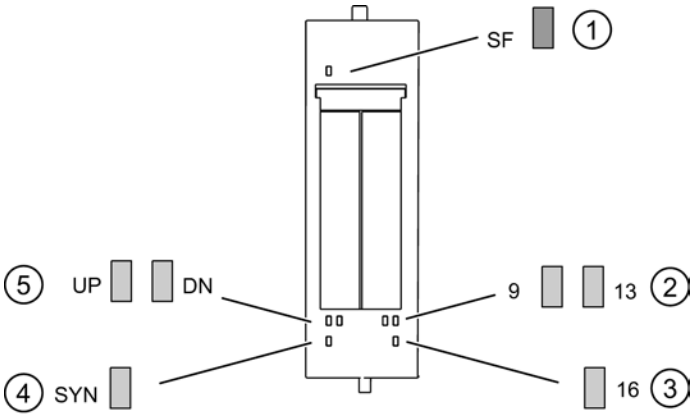
Bit	Explanation
ERR_PARA	Errors in assigned module parameters.
EXTF	Group error Possible causes: • Encoder supply short circuit • Short circuit or encoder signal wire break EXTF is reset when the causes of the errors have been eliminated.
LZ	The lifebeat is toggled each time the feedback interface is updated, i.e. the last value sent is inverted.

Bit	Explanation
STS_DI	The bit displays the status of the digital input DI.
STS_DIR	Direction status: • for encoder value change from larger to smaller encoder positions (including zero pass) → "1 " • for encoder value change from smaller to larger encoder positions (including zero pass) → "0 "
STS_GATE	Internal gate status: Counting.
STS_TIC	Isochronous mode (if assigned) was applied.

1Count5V diagnostics

Diagnostics using the LED display

LED display on the 1Count5V



① Group error (red)
② Status display for the digital output (green)
③ Status display for digital input (green)
④ Status display for synchronization (green)
⑤ Status display for counting direction (green)

Status and error displays by means of LEDs on the 1Count5V

The table below shows the status and fault displays on the 1Count5V.

Event (LED)							Cause	Remedy
SF	UP	DN	SYN	9	13	16		
On							No parameter assignment. There is a diagnostic message.	Check the parameter assignment. Evaluate the diagnostics data.
	On						Status of the low-order bits of the counter, if the counter counts up.	

Event (LED)							Cause	Remedy
SF	UP	DN	SYN	9	13	16		
		On					Status of the inverse low-order bits of the counter, if the counter counts down.	
			On				Synchronization is performed (only in counting modes; image of STS_SYN acknowledgement bit).	
				On			DO 1 is activated.	
					On		DO 2 is activated.	
						On	DI is activated.	

Error types

For information on the structure of the channel-related diagnostics, refer to the manual on the interface module used in your ET 200S station.

1Count5V error types

The following table shows the 1Count5V error types.

Error class		Meaning	Remedy
1_D	00001: Short circuit	Short circuit of the sensor supply or the actuator.	Check the wiring to the sensor. Correct the process wiring.
5_D	00101: Overtemperature	The digital output is overloaded.	Correct the process wiring.
6_D	00110: Open circuit	Line to the actuator is interrupted.	Correct the process wiring.
9_D	01001: Error	Internal module error occurred.	Replace the module.
16_D	10000: Parameter assignment error	Parameters have not been assigned to the module.	Correct the parameter assignment.
26_D	11010: External error	Wire break/short circuit of the 5 V sensor signals: A, /A, B, /B, N, /N,	Correct the parameter assignment.

11.2 Cam control (S7-300, S7-400)

11.2.1 Cam control

11.2.1.1 Cam control basics

What is cam control?

Cam control

Cam controllers are implemented to activate position-dependent or time-dependent functions. They are far superior to mechanical cam controllers, in particular due to their high flexibility; changes can for example be implemented during operation using software.

A cam generates position-dependent switching signals depending on the position values of the axis or an external encoder. You distinguish between position-based and time-based cams.

A cam is defined by the following:

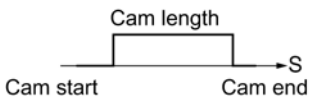
- Initial position and final position (position-based cams (Page 9340))
- Initial position and ON period (time-based cams (Page 9340))

Properties of the cam types

Cam types

You can assign each cam for operation as a position-based cam or time-based cam.

The following table sets outs the properties of the two cam types:

	Position-based cams	Time-based cams
Representation		
Parameter assignment	The following parameters are assigned: • Cam start • Cam end • Effective direction • Lead time	The following parameters are assigned: • Cam start • Activation time • Effective direction • Lead time

	Position-based cams	Time-based cams
Effective direction	Two effective directions are supported: • positive: The cam is activated at the cam start, if the axis is moving in direction of increasing actual values. • negative: The cam is activated at the cam end, if the axis is moving in direction of decreasing actual values. You can set both effective directions in parallel.	Two effective directions are supported: • positive: The cam is activated at the cam start, if the axis is moving in direction of increasing actual values. • negative: The cam is activated at the cam start if the axis is moving in direction of decreasing actual values. You can set both effective directions in parallel.
Enabling	The cam is activated: • at the cam start, if the axis is moving in positive direction, and a positive effective direction is set. • at the cam end if the axis is moving in a negative direction and a negative effective direction is set. • when the actual value lies within the cam range.	The cam is activated: • at the cam start if the axis is moving in positive direction and a positive effective direction is set or the axis is moving in a negative direction and a negative effective direction is set. After it has been activated, the full cam activation time elapses. even if the direction of motion of the axis changes after the cam is activated. The cam is **not** retriggered if its start position is passed once again within the cam activation time.
Deactivating	The cam is deactivated if: • the assigned distance has been traveled • its effective direction is opposite to the direction of the axis motion, and you have not assigned a hysteresis • the actual value is out of the cam range.	The cam is deactivated on expiration of the assigned time.
Path length	The path length of the cam is defined by the cam start and cam end. The cam start and cam end belong to the active section of the cam.	The path length of the cam is determined by the axis velocity within the cam activation time.
ON period	The on period of the cam is determined by the speed at which the axis travels across the path length of the cam.	The on period of the cam is assigned with the activation time.

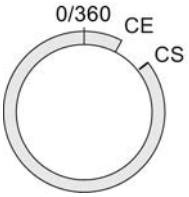
Direction recognition

The direction of the axis motion is determined as follows:

- At each pulse of the incremental encoder.
- With each error-free frame of an SSI encoder.

Inverse cam

An inverse cam can briefly be generated with "Change cam edge" if the new cam start is greater than the old cam end. The table below shows the effect of an inverse cam on a linear and on a rotary axis.

Inverse cam with a linear axis	Inverse cam with a rotary axis
	
The cam start is greater than the cam end	The cam start is more positive than the cam end
With both types of axis there must always be a minimum clearance of 4 pulses between the cam start and the cam end.	

Normal tracks and track results

Cam tracks

The 32 tracks can be used to control up to 32 different switching actions. You can evaluate the tracks based on the checkback signals.

The first 13 tracks (track 0 to 12) are each assigned a digital output (Q0 to Q12) of FM 352. This output can be used for the direct control of a connected contactor, for example.
The first 16 tracks (track 0 to 15) in the FM 452 are each assigned a digital output (Q0 to Q15).

Track result

The system provides up to 128 cams which can be assigned to user-specific tracks.

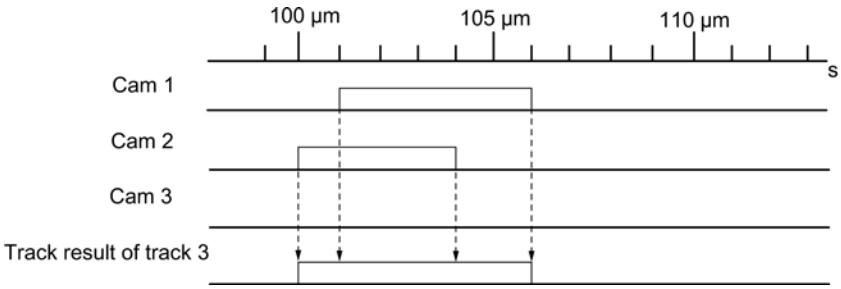
Each track can be assigned several cams. The track result is a logic OR operation derived from all cam values of this track.

Example of a track result

During parameter assignment, you define the following cams for track 3:

Cam	Cam start	Cam end
1	101 µm	106 µm
2	100 µm	104 µm

The figure below shows the track result.



Track enable

You must activate the tracks used in order to transfer the track results of tracks 0 to 12 as track signals to the digital outputs Q0 to Q12 of FM 352. The FM 452 has tracks 0 to 15.

External enable of a track

You can assign an external enable signal for track 3 in the machine data. Track signal 3 is then logically linked to digital input I3 by an AND operation before it can switch digital output Q3 of the FM 352.

Digital output Q3 is thus only set if the following conditions are satisfied:

- The corresponding track is enabled.
- At least one cam on this track is active (track result = 1).
- The corresponding digital input I3 was set by an external event.

Tracks 3 to 10 are available for external enabling in the FM 452.

Setting the track signals

The track signals 0 to 12 (according to digital outputs Q0 to Q12) can be set by the cam control system, or by the CPU. Track signals 0 to 15 can be set in the FM 452.

Special tracks

Definition

You can program tracks 0 to 2 for operation as special tracks:

- Track 0 or 1: Counter cam track
- Track 2: Brake cam track
  Input I0 must be evaluated in order to enable the track.

Requirements

Requirements of working with special tracks:

- You programmed the cams for the track
- Cam processing is enabled
- The corresponding track is enabled
- The track is programmed as special track

Counter cam track

A counter cam track counts the status transitions of the track results on this track.

Define a counter high counter value for the counter cam track and then start the counter function.

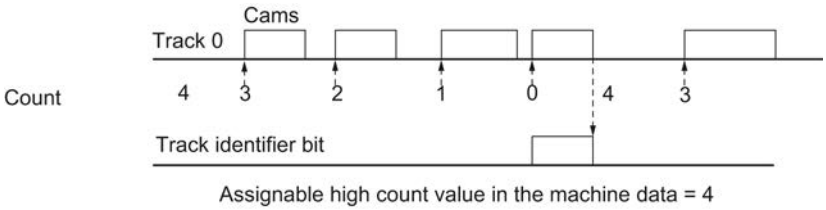
The counter value of the relevant track decrements by the count of 1 at each positive edge of the track result signal.

The track flag bit = 0 as long as the value of the counter cam track is not equal to zero.

The controller sets the track flag bit and, if assigned parameters accordingly, the track signal (Page 9347) when the counter value = 0.

It resets the track flag bit and sets the default value of the counter at the next negative edge of the track result signal (all cams on this track are disabled).

The figure below shows how a counter cam track is switched.



Brake cam track

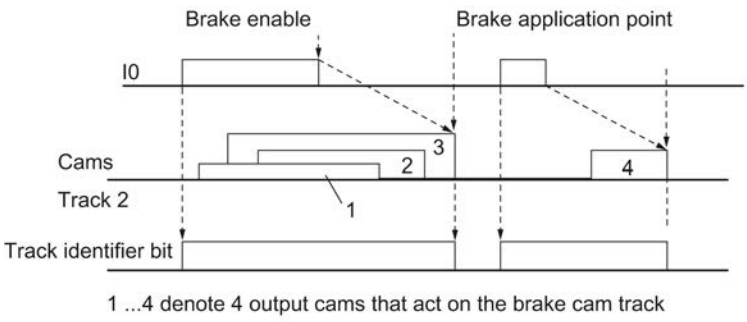
In order to use track 2 as a brake cam track, interconnect digital input I0.

A positive edge of the I0 signal sets the track flag bit.

The track flag bit is reset again when:

- there is no longer a "1" signal at I0 **and** afterwards
- the controller has detected a negative edge at the track 2 result signal.

The figure below shows how a brake cam track is switched.



In the example, the track flag bit is reset by a negative edge at cam 3 or 4.

Hysteresis

Definition

Mechanical imbalance at the axis may cause fluctuation of the actual position value. If the actual position value is offset by one edge of a cam, or within an active cam with only one effective direction, this cam's activation would be cycled on and off continuously. A hysteresis prevents this flutter.

A hysteresis setting is dependent on the actual value, and applies globally to all cams. It is enabled when a direction reversal is detected. A hysteresis will always take effect, regardless of whether or not a cam is set at the current axis position.

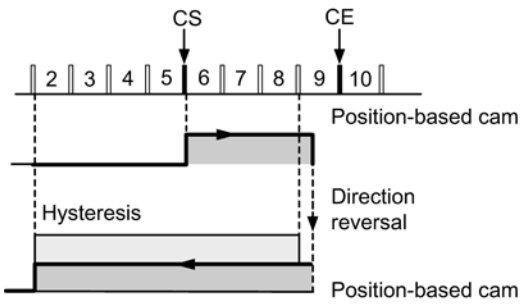
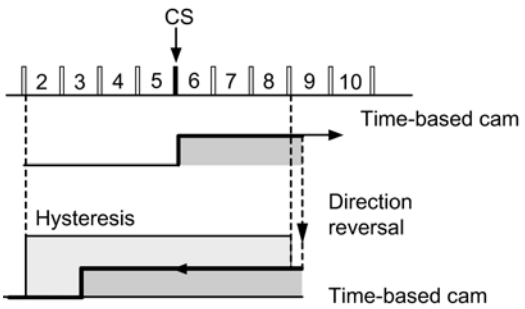
Rules for the hysteresis range

Rules applicable to the hysteresis range:

- The hysteresis will always be set when a directional reversal is detected.
- The indication of the actual value remains constant within the hysteresis.
- The direction is not redefined within the hysteresis.
- A position-based cam is neither activated nor deactivated within the hysteresis.
- A time-based cam is not activated within the hysteresis. An active time-based cam is deactivated on expiration of the assigned activation (not only on reaching the hysteresis limit).
- When the value is out of the hysteresis range, the FM x52 sets:
  - the actual position value
  - the current direction of motion of the axis
  - the current states of all cams
- The hysteresis range applies to all cams.

Direction reversal of a cam with hysteresis

The table below gives an example of the response to cam direction reversal. A distinction must be made between the reaction of position- and time-based cams. The effective direction of the cam is **positive**.

Position-based cam	Time-based cam
	
The hysteresis will be set when a directional reversal is detected. The cam is deactivated immediately when the the hysteresis range is violated.	The cam always remains active for the duration of the assigned activation time.
Cam Hysteresis	

Dynamic adjustment

Introduction

The dynamic adjustment is used to compensate delay times of the connected control elements.

Lead time

You can program a delay time and assign it as derivative-action time to each cam. You can assign one derivative-action time to each cam. The derivative-action time applies to the cam start and end position.

Correction distance

The actuation distance of a cam is calculated continuously based on the current velocity and derivative-action time. The entire cam is shifted in direction of the actual value by this value. The programmed range is the "static range," and the range calculated based on the derivative-action time represents the "dynamic range."

Actuation distance = lead time x actual velocity of the axis

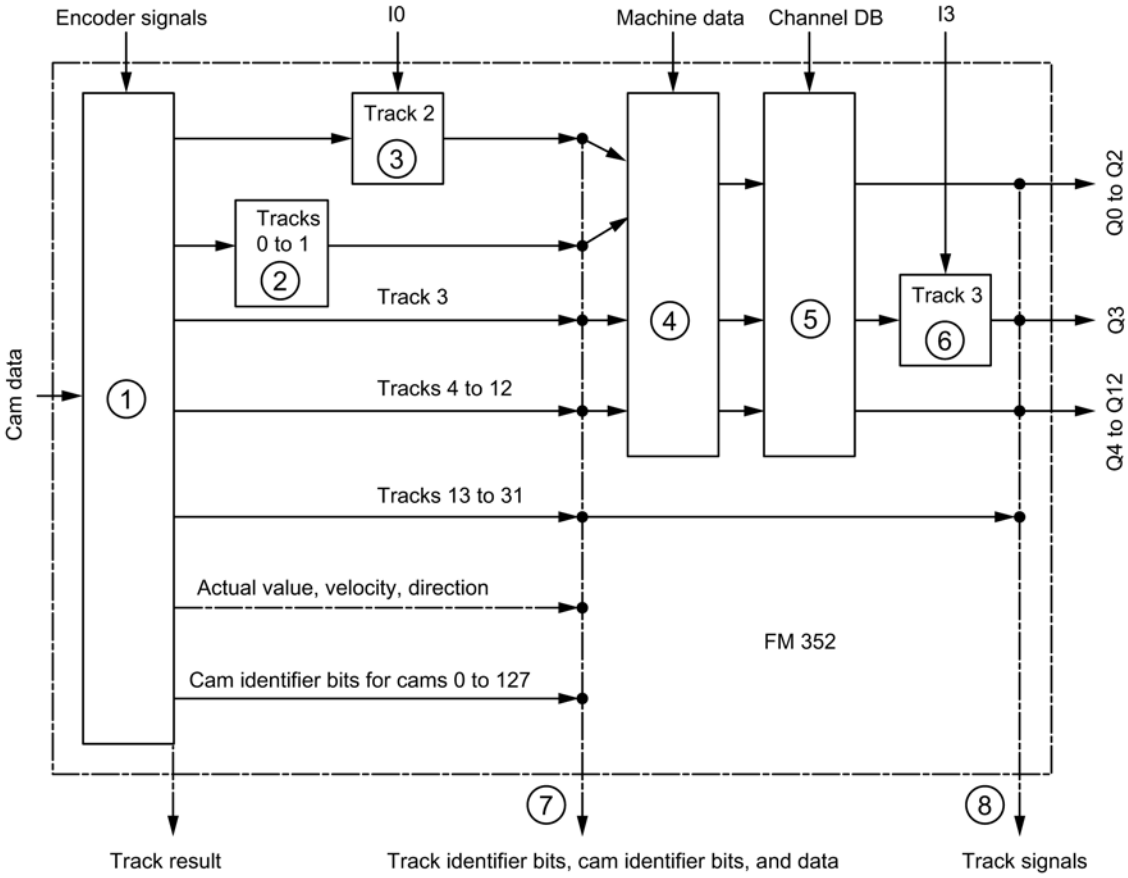
FM x52 calculates the actuation distance of all cams within 1/4 of the longest parameterized lead time.

An extremely high derivative-action time of a cam reduces the dynamic performance of cam processing.

Interfaces of the cam controller

Overview

Taking FM 352 as an example, the schematic diagram below sets out the major interfaces of the cam controller to illustrate the connection between data, inputs and outputs. Differences to the FM 452 are set out in the table below.



No.	Description
①	The cam processing functions of FM x52 calculate the cam identifier bits on the basis of the switching conditions and actual value. The module also determines the track results according to the assignment of the cams to the tracks.
②	If you assigned track 0 or 1 as a counter cam track, the track result of the cam controller (point 1) is logically linked to the counter result in order to produce the track identifier bit. The track identifier bit is otherwise equivalent to the track result.
③	If you assigned track 2 as a brake cam track, the track result of the cam controller (point 1) is logically linked to input I0 to produce the track identifier bit. The track identifier bit is otherwise equivalent to the track result.

No.	Description
④	Using machine data, you can control whether the previously determined track identifier bits of tracks 0 to 12 (in FM 352) or 0 to 15 (in FM 452) of the cam controller are transferred to the channel DB, or whether they are set directly by the track enable signal (TRACK_EN).
⑤	You enable the track signals of tracks 0 to 12 (or 0 to 15) by setting TRACK_EN and the count function by setting CNTC0_EN / CNTC1_EN.
⑥	FM 352: The track signal of track 3 and digital input I3 can be logically linked by an AND operation provided you have set this option in the machine data (EN_IN_I3).
	FM 452: The track signals of tracks 3 to 10 can be logically linked to digital inputs I3 to I10 with an AND operation if you have set this option in the machine data (EN_IN_I3 to EN_IN_I10).
⑦	All track and cam identifier bits can be read at this location (i.e. before being logically linked with machine and channel data) using the ACTPOS_EN or CAMOUT_EN job. At tracks 3 to 31, the track identifier bit is equivalent to the track result (point 1).
⑧	FM 352: The track signals of tracks 0 to 12 are available in the checkback signals after having been logically linked with the machine data (e.g. axis type, software limit switch, distance per encoder revolution...) and channel data. The track signals of tracks 13 to 31 and the track identifier bits of point 7 are identical. The track signals of tracks 0 to 12 are also available at the digital outputs Q0 to Q12.
	FM 452: The track signals of tracks 0 to 15 are available in the checkback signals after having been logically linked with the machine and channel data. The track signals of tracks 16 to 31 and the track identifier bits of point 7 are identical. The track signals of tracks 0 to 15 are also available at the digital outputs Q0 to Q15.

Encoder signals

Supported encoders

Connectable incremental encoders

Incremental encoders (Page 9350) with two pulses 90° out of phase with or without zero marks are supported:

- Encoders with asymmetrical 24 V output signals
  - Limit frequency = 50 kHz
  - Cable up to 100 m long, shielded
- Encoders with symmetrical output signals with 5 V differential interface in accordance with RS 422
  - Limit frequency = 1 MHz
  - With 5 V voltage supply: Cable up to 32 m long, shielded
  - With 24 V voltage supply: Cable up to 100 m long, shielded

Note

If the 5 V encoder does not output a zero mark signal and wire-break monitoring is enabled, you must interconnect the zero mark signal inputs N and /N externally so that the inputs have different signal levels (for example, N to 5 V, /N to ground).

Supported initiators

The FM x52 supports the following initiators (Page 9352):

- Initiators with 24 V signal level (proximity switches)
- Limit frequency = 50 kHz
- Cable up to 100 m long

Absolute encoders supported

Absolute encoders (Page 9352) with serial interface are supported. Position data are transferred synchronously using the SSI protocol (**S**ynchronous**S**erial**I**nterface). The FM x52 only supports the GRAY code. Due to the arrangement of the data bits in the transferred frames, the data formats "fir tree", "half fir tree" and "right-justified" are used.

Absolute encoders are divided into the categories:

- Single-turn encoder
  The total range of single-turn encoders is scaled to one revolution.
- Multiturn encoder
  The total range of multiturn encoders is scaled to several revolutions.

Encoder type	Frame length / type
Single-turn encoder	13-bit half fir tree
Single-turn encoder	13-bit right-justified
Single-turn encoder	25-bit right-justified
Multiturn encoder	25-bit fir tree
Multiturn encoder	25-bit right-justified
Listen in	Fir tree
Listen in	Right-justified
Special setting:	
Multiturn encoder in single-turn mode	25-bit half fir tree

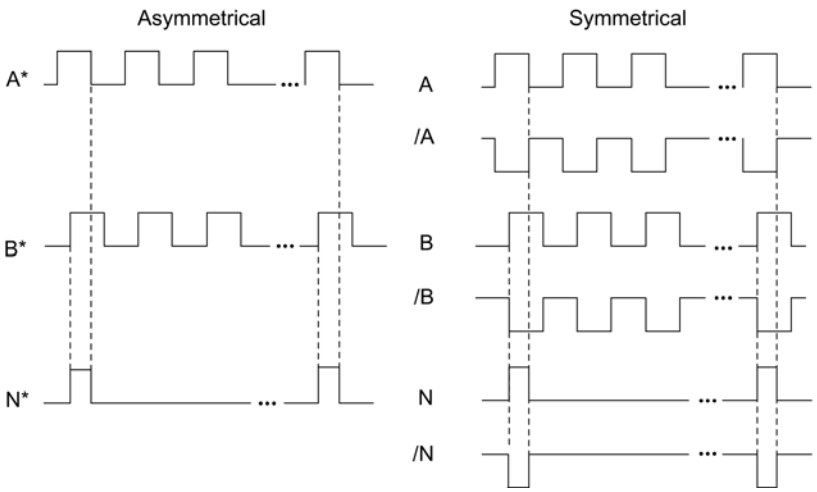
The transmission rate for data transmission depends on the cable length:

- Max. 125 kHz for 320 m of cable, shielded
- Max. 250 kHz for 160 m of cable, shielded
- Max. 500 kHz for 60 m of cable, shielded
- Max. 1 MHz for 20 m of cable, shielded

Incremental encoder

Signal shapes

The following figure illustrates the signal forms of encoders with asymmetrical and symmetrical output signals.

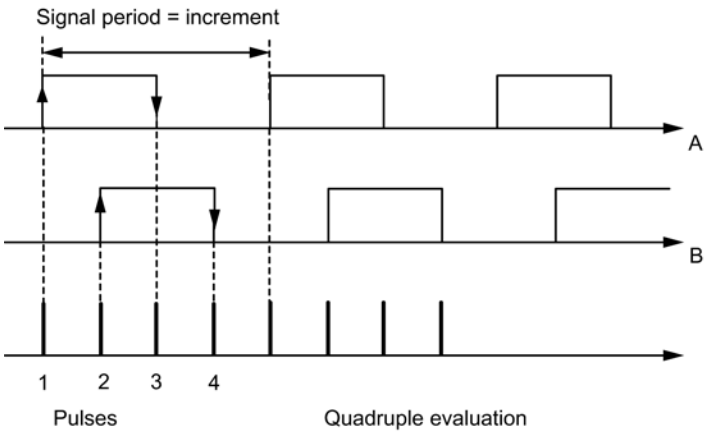


Signal evaluation

Increments

An increment identifies a signal period of the encoders signals A and B. This value is specified in the technical data and/or on the rating plate of the encoder.

The figure below shows the ratio of increments to pulses.



Pulses

FM x52 evaluates all 4 edges of signals A and B in each increment (quadruple evaluation).

1 increment (encoder default) = 4 pulses (FM evaluation)

Reaction times

FM x52 has the following response times for connected incremental encoders:

- Minimum reaction time = cam cycle time + switching time of the connected switching elements
- Maximum reaction time = 2 x cam cycle time + switching time of the connected switching elements

Example of reaction times

Example of the min./max. reaction time with 16 cams:

- Cam cycle time: approx. 20 µs
- Switching time of the hardware: approx. 150 µs

Minimum reaction time = 20 µs + 150 µs = 170 µs

Maximum reaction time = 2 x 20 µs + 150 µs = 190 µs

Note

You can compensate the reaction time by assigning the cam parameters accordingly or using dynamic adjustment.

Flat gain

The flat gain is equivalent to the difference between the min./max. reaction time. For incremental encoders this is:

Flat gain = cam cycle time

Note

If the switching time of the FM x52 hardware and of the connected switching elements can be disregarded, reliable cam activation is always ensured if the cam is longer than the distance covered within the cam cycle time.

Proximity switches

Definition

Initiators are simple switches which output pulse-shaped signals, and do not return a directional signal. You define the direction with the machine data "Initiator direction".

CAUTION

Risk of material damage!

Incorrect direction settings can cause serious errors in the system (for example, faulty control of a unit of equipment).

Check the direction settings in the commissioning phase, and whenever you replace an initiator.

Signal evaluation

With an initiator, the rising edge of signal A* is counted.

Absolute encoder (SSI)

Absolute encoder pulse evaluation

1 increment (encoder default) = 1 pulse (FM evaluation)

Listen in

"Listen in" means: An absolute encoder is operated in parallel in two modules (for example FM x51 and FM x52). The FM x51 positioning module is the master and clocks the absolute encoder; the FM x52 electronic cam controller acts as the slave listening in to the signals of the SSI frame.

Set "Increments/Encoder revolution" and "Number of revolutions" to the master setting. The "Transmission rate" is not relevant. Depending on the encoder type select "Listen in" or "Listen in Right-Justified" for "Frame Length".

Reaction times

The FM x52 has the following reaction times for absolute encoders:

- Minimum reaction time = frame run time + cam cycle time + switching time of the connected switching elements
- Maximum reaction time = 2 x frame runtime + monoflop time + 2 x cam cycle time + switching time of the connected switching elements
- For programmable absolute encoders:
  Maximum reaction time = frame runtime + monoflop time + 2 x cam cycle time + switching time of the connected switching elements +1/max. step sequence rate

Monoflop time

The following limit values apply to the monoflop time:

- Minimum monoflop time: > 15 µs
- Maximum monoflop time: < 64 µs

Encoders with values outside the limits shown here are not permitted.

Frame run time

The frame run times depend on the transmission rate:

Transmission rate	Frame run time for 13 bits	Frame run time for 25 bits
0.125 MHz	112 µs	208 µs
0.250 MHz	56 µs	104 µs
0.500 MHz	28 µs	52 µs
1000 MHz	14 µs	26 µs

Example of Reaction Times

The following example shows how to calculate the minimum and maximum reaction times. In the example a programmable encoder is not used.

- Cam cycle time: approx. 20 µs for max. 16 cams
- Switching time of the hardware: approx. 150 µs
- Frame run time: 26 µs at a transmission rate of 1 MHz (25-bit frame)
- Monoflop time: 20 µs (depends on the encoder: typical 20 µs to 40 µs

Maximum reaction time = 26 µs + 20 µs + 150 µs = 196 µs

Maximum reaction time = 2 x 26 µs + 20 µs + 2 x 20 µs + 150 µs = 262 µs

Note

You can compensate the reaction time by assigning the cam parameters accordingly or by using dynamic adjustment.

Flat gain

The flat gain is equivalent to the difference between the minimum and maximum reaction times.

For an absolute encoder, it is as follows:

- Flat gain = cam cycle time + frame run time + monoflop time

With programmable absolute encoders, it is as follows:

- Flat gain = cam cycle time + frame run time + monoflop time
  + 1/max. step sequence frequency

Note

If the switching time of the FM x52 hardware and of the connected switching elements can be disregarded, reliable cam activation is always ensured if the cam is longer than the distance covered within the cam cycle time.

11.2.1.2 Functions of the FM x52

Writing and enabling machine data

Writing and activating machine data

You can use the machine data to adapt the FM x52 to the axis and the encoder.

You can change parameters during operation using the user program.

All parameters are stored in the parameter DB:

- Machine data are stored in the parameter DB at addresses 3.1 to 104.0.

Enter the parameter DB number in the corresponding channel DB.

You can enter the parameters with the DB Editor, or indeed simply and easily enter them in the "System of units", "Axis", "Encoder" and "Tracks" dialogs and write them to the parameter DB with the "Export" function.

You can use the "Import" function to import parameters from an existing parameter DB to the dialogs.

Initial parameter assignment

If the module does not yet contain any machine data (feedback signal PARA = 0), proceed as follows for initial parameter assignment without the parameter assignment dialogs:

1. Enter the new values in the parameter DB.
2. Download the parameter DB to the CPU.
3. Set the following trigger bit in the channel DB:
   – Write machine data (MDWR_EN = 1)
4. Call the CAM_CTRL (Page 7157) or CAM_CTRL_452 (Page 7160) instruction in the cyclic user program.

Modifying machine data

To modify existing machine data (feedback signal PARA = 1) using the user program, proceed as follows:

1. Enter the new values in the parameter DB.
2. Set the trigger bits at the channel DB:
   – Write machine data (MDWR_EN = 1)
   – Activate machine data (MD_EN = 1)
3. Call the CAM_CTRL (Page 7157) or CAM_CTRL_452 (Page 7160) instruction in the cyclic user program.
4. Check whether the previous cam data and the modified machine data are compatible.
5. Always rewrite the cam data of the parameterized cams, irrespective of whether or not they have been changed (CAM1WR_EN … CAM8WR_EN).

If you set the trigger bits simultaneously, the CAM_CTRL or CAM_CTRL_452 instruction ensures the jobs are processed in the correct order.

Otherwise, you must always change the machine data in the following order:

1. Write machine data
2. Activate machine data

Data used in the channel DB

Address	Name	Type	Start value	Comment
35.0	MDWR_EN	BOOL	FALSE	1 = write machine data
35.1	MD_EN	BOOL	FALSE	1 = enable machine data

Note

If any parameters relevant for synchronization were modified, the synchronization will be canceled when the machine data are activated. The functions will also be reset and all machine and cam data deleted from the module.

Parameters relevant to synchronization:

- Axis type
- End of rotary axis
- Signal type
- Distance per encoder revolution
- Increments per encoder revolution
- Number of revolutions
- Reference point coordinate
- Absolute encoder adjustment
- Type of reference point retriggering
- Direction adaptation
- Number of cams
- Software limit switch start and end

Reading machine data

Reading machine data

To read actual machine data from the module:

1. Set the following trigger bit in the channel DB:
   – Reading machine data (MDRD_EN = 1)
2. Call the CAM_CTRL (Page 7157) or CAM_CTRL_452 (Page 7160) instruction in the cyclic user program.

This saves the current machine data to the parameter DB on the CPU.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
37.1	MDRD_EN	BOOL	FALSE	1 = read machine data

Writing cam data

Writing cam data

Cam data define the type and function principle of the cams and their assignment to the tracks.

Cam data are stored in the parameter DB, starting at address 108.0. These data are grouped in packets, each consisting of 16 cams.

Cam data are active immediately after having been written.

To write cam data without the parameter assignment dialogs, proceed as follows:

1. Enter the new values in the parameter DB.
2. Download the parameter DB to the CPU.
3. Set the trigger bits in the channel DB (CAM1WR_EN … CAM8WR_EN)
4. Call the CAM_CTRL (Page 7157) or CAM_CTRL_452 (Page 7160) instruction in the cyclic user program.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
35.3	CAM1WR_EN	BOOL	FALSE	1 = write cam data 1 (cams 0 to 15)
35.4	CAM2WR_EN	BOOL	FALSE	1 = write cam data 2 (cams 16 to 31)
35.5	CAM3WR_EN	BOOL	FALSE	1 = write cam data 3 (cams 32 to 47)
35.6	CAM4WR_EN	BOOL	FALSE	1 = write cam data 4 (cams 48 to 63)
35.7	CAM5WR_EN	BOOL	FALSE	1 = write cam data 5 (cams 64 to 79)
36.0	CAM6WR_EN	BOOL	FALSE	1 = write cam data 6 (cams 80 to 95)
36.1	CAM7WR_EN	BOOL	FALSE	1 = write cam data 7 (cams 96 to 111)
36.2	CAM8WR_EN	BOOL	FALSE	1 = write cam data 8 (cams 112 to 127)

Reading cam data

Reading cam data

To read actual cam data from the module:

1. Set the following trigger bit in the channel DB:
   – Read cam data (CAM1RD_EN … CAM8RD_EN)
2. Call the CAM_CTRL (Page 7157) or CAM_CTRL_452 (Page 7160) instruction in the cyclic user program.

This saves the actual cam data to the parameter DB on the CPU.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
37.2	CAM1RD_EN	BOOL	FALSE	1 = read cam data 1 (cams 0 to 15)
37.3	CAM2RD_EN	BOOL	FALSE	1 = read cam data 2 (cams 16 to 31)
37.4	CAM3RD_EN	BOOL	FALSE	1 = read cam data 3 (cams 32 to 47)
37.5	CAM4RD_EN	BOOL	FALSE	1 = read cam data 4 (cams 48 to 63)

Address	Name	Type	Initial value	Comment
37.6	CAM5RD_EN	BOOL	FALSE	1 = read cam data 5 (cams 64 to 79)
37.7	CAM6RD_EN	BOOL	FALSE	1 = read cam data 6 (cams 80 to 95)
38.0	CAM7RD_EN	BOOL	FALSE	1 = read cam data 7 (cams 96 to 111)
38.1	CAM8RD_EN	BOOL	FALSE	1 = read cam data 8 (cams 112 to 127)

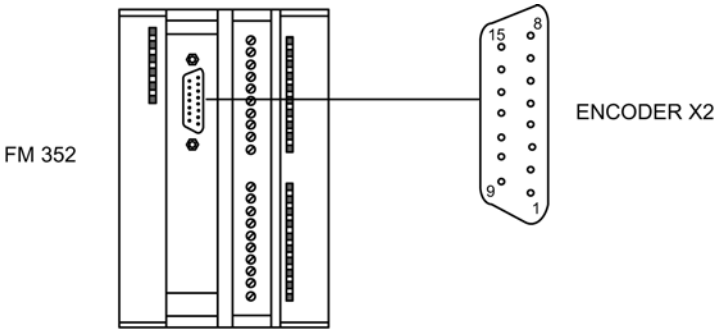
11.2.1.3 Using FM 352

Terminal assignment of FM 352 front connector

Description of the encoder interface

Location of the sub D socket

The figure below shows the position and designation of the connector on the module. You can connect an initiator, incremental encoder or absolute encoder (SSI) to the sub D socket.



Pin assignment of the encoder interface (X2)

Pin	Name	Proximity switch	Incremental encoder	Absolute encoder
1	A*	Encoder signal A (24 V)		---
2	CLS	---	---	SSI shift clock
3	/CLS	---	---	SSI shift clock inverse
4	B*	---	Encoder signal B (24 V)	---
5	24 VDC	Encoder supply 24 V		
6	5.2 VDC	---	Encoder supply 5.2 V	
7	M	Ground		
8	N*	---	Zero mark signal (24 V)	---
9	RE	---	Current sourcing/current sinking[2)]	---
10	N	---	Zero mark signal (5 V)	---

Pin	Name	Proximity switch	Incremental encoder	Absolute encoder
11	/N	---	Zero mark signal inverse (5 V)	---
12	/B / /CLI [1]	---	Encoder signal B inverse (5 V)	SSI shift clock inverse
13	B / CLI [1]	---	Encoder signal B (5 V)	SSI shift clock
14	/A / /DAT	---	Encoder signal A inverse (5 V)	SSI data inverse
15	A / DAT	---	Encoder signal A (5 V)	SSI data

[1] In listen-in mode

[2] **Note:** If you are connecting an incremental encoder from a third party in series-mode switching (current sourcing/current sinking), the following must be observed:

- Current sourcing: Connect RE (9) with ground (7).
- Current sinking: Connect RE (9) with +24 V (5).

Terminal assignment of the front connector

Front connector

Connect the following to the 20-pin front connector:

- the digital outputs and switching elements (inputs)
- the supply voltage of the encoders and digital outputs

Terminal assignment of front connector (X1)

The table below shows the terminal assignment of the front connector of FM 352.

Wiring	Name	Function
1	L+	24 V DC encoder supply, and digital outputs
2	M	Ground of the encoder supply and digital outputs
3	I 0	Brake enable
4	I 1	Length measurement/ edge detection/ set actual value on-the-fly
5	I 2	Home position switch
6	I 3	Enable track signal 3
7	Q 0	Digital output 0
8	Q 1	Digital output 1
9	Q 2	Digital output 2
10	Q 3	Digital output 3
11	Q 4	Digital output 4
12	Q 5	Digital output 5
13	Q 6	Digital output 6
14	Q 7	Digital output 7
15	Q 8	Digital output 8
16	Q 9	Digital output 9

Wiring	Name	Function
17	Q 10	Digital output 10
18	Q 11	Digital output 11
19	Q 12	Digital output 12
20	–	–

Additional information about design and wiring can be found in the Operating Instructions (http://support.automation.siemens.com/WW/view/en/2103044) of the electronic cam controller FM 352.

Auxiliary voltage for encoders and digital outputs (L+, M)

The 24 VDC auxiliary voltage of the encoders and digital outputs is monitored:

- for wire-break of the 24 V feed line
- for power failure

This supplies 24 V DC and 5.2 V DC at the encoder interface (sub D socket X2) to the various types of encoders.

Load current supplies

The DC load power supply must meet the following requirements:

Only a safe, isolated extra-low voltage of ≤ 60 V DC may be used as the load current supply. Safe isolation can be implemented, for example, by adhering to the specifications

- VDE 0100 Part 410 / HD 384-4-41 / IEC 364-4-41
  (as protective extra low voltage) and/or
- VDE 0805 / EN 60950 / IEC 950
  (as safety low voltage SELV) or
- VDE 0106 Part 101.

4 digital inputs (I0 to I3)

You can connect bounce-free switches (24 V current sourcing) or non-contact sensors (2- or 3-wire proximity switches) to the 4 digital inputs.

The digital inputs are not monitored for short circuits or wire break and have a non-isolated connection to the module chassis.

13 digital outputs (Q0 to Q12)

The state (on/off) of tracks 0 to 12 is output at 13 digital outputs. The digital outputs have a non-isolated connection to the module chassis.

Loads supported:

- Operating voltage 24 V
- Current load 0.5 A / short-circuit proof

The state of each output can be read off from the respective LED.

Cam control functions

Influence of functions on the switching characteristics of time-based cams

Actual value changes

A time-based cam can be skipped by the following functions that change the actual value:

- Set actual value
- Set actual value on-the-fly
- Zero offset
- Retrigger reference point

Activating a time-based cam

If you skip the start of a time-based cam with one of the above functions, this cam will be activated provided the actual direction of the axis motion is the same as the effective direction assigned for the cam. The programmed cam activation time starts.

Note

If the axis is at a standstill, the direction of movement is influenced by fluctuations of the actual value.

Set a hysteresis higher than the fluctuation of the actual value signal in order to suppress flutter when the axis is at a standstill.

This retains the last determined direction of movement while the axis is at a standstill.

WARNING

Risk of injury and material damage.

Functions which modify the actual value at rotary axes may cause the unwanted activation of time-based cams.

You should always set the "invalid" option for the time-based cams of a rotary axis if you want to influence the actual value using the functions mentioned above.

Set Actual Value/Set Actual Value on-the-fly/Cancel Set Actual Value

Definition

Use the "Set process value" and "Set process value on-the-fly" functions to assign a new coordinate to the current encoder value. This shifts the coordinate system by the value: ACT_{new} - $ACT_{current}$

Whereby:

- ACT_{new} is the default value
- $ACT_{current}$ is the process value at the time of execution

Determining new coordinates

Calculate all your default positions in the shifted coordinate system based on the following formula:

$Coordinate_{new} = Coordinate_{old} + (ACT_{new} - ACT_{current})$

Requirements

- The axis must be synchronized.
- With "set process value on-the-fly": Digital input I1 must be interconnected.

Function sequence

1. Enter the coordinate for the process value, or for the process value to set on-the-fly in the channel DB.
   - Linear axis:
     You must select the specified process value so that the software limit switches are still within the valid traversing range after the function is called.
     The absolute value of the offset derived from $(ACT_{new} - ACT_{current})$ must be less than or equal to the absolute value of the permitted traversing range (maximum 100 m or 1000 m).
   - Rotary axis:
     Rule for the defined process value:
     0 ≤ process value < end of rotary axis
2. Set the appropriate trigger bits at the channel DB.
3. Call the CAM_CTRL (Page 7157) instruction.
   "Set process values" is executed immediately.
   "Set process value on-the-fly" is executed at the next positive edge at digital input I1. The FVAL_DONE bit is set.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
36.4	AVAL_EN	BOOL	FALSE	1 = set process value
36.5	FVAL_EN	BOOL	FALSE	1 = set process value on-the-fly
90.0	AVAL	DINT	L#0	Process value coordinate
94.0	FVAL	DINT	L#0	Coordinate for on-the-fly process value
25.5	FVAL_DONE	BOOL	FALSE	1 = set process value on-the-fly executed

Effects of the function

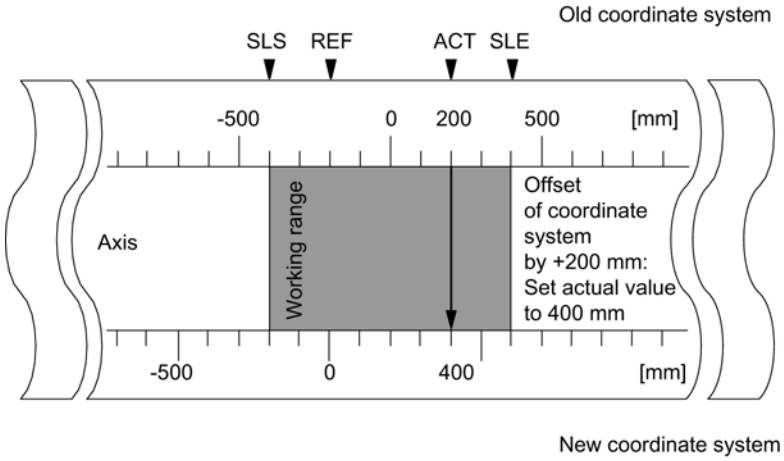
The example "set process value" to 400 mm (at position 200 mm) demonstrates how this function shifts the coordinate system. Resultant effects:

- The position of the working range is **not** physically shifted.
- The individual points (such as the software limit switches) are assigned new coordinate values.
- The cams retain their coordinate values, and are therefore located at a different physical position.
- If the axis is synchronized and cam processing is enabled, the positional process value may skip cam edges or entire cams as a result of this function.
- Status changes of the cam which would normally trigger an interrupt may be lost.

Note

Make allowances for the switching characteristics of time-based cams (Page 9361).

The table below shows how "Set process value" / "Set process value on-the-fly" shifts the coordinate system.

Set process value	SLS [mm]	REF [mm]	ACT [mm]	SLE [mm]
	-400	-200	200	400
	-200	0	400	600

Cancelling the function

The "Undo set process value" function can be used to reset the coordinate shift caused by "Set process value" or "Set process value on-the-fly".

Once "Set process value on-the-fly" has been triggered, it can no longer be deleted before execution by a positive edge at input I1. However, it can be overwritten by a new "Set process value on-the-fly".

These functions are reset when a module is started up.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
35.2	AVALREM_EN	BOOL	FALSE	1: Cancel process value setting

Possible causes of error

"Set process value on-the-fly" and "Retrigger reference point" may not be executed in parallel.

The "Set actual value on-the-fly" function can signal an operator error if the function would cause a software limit switch to exceed the valid traversing range with a positive edge at I1. This operating error is signaled by a diagnostic interrupt, and written to the diagnostic buffer.

Set zero offset

Definition

The "Zero offset" function lets you shift the zero point in the coordinate system by a defined value. The sign determines the offset direction.

Determining a new coordinate

All values of the shifted coordinate system can be calculated according to the following formula:

$\text{Coordinate}_{new} = \text{Coordinate}_{old} - (\text{ZPO}_{new} - \text{ZPO}_{old})$

ZPO_{old} identifies any existing zero offset. If no zero offset was active prior to the call, set a 0 value at ZPO_{old}.

This equation allows you to calculate the coordinates for the software limit switches, for example.

Function sequence

1. Write the zero offset value to the channel DB.
   - Linear axis:
     The zero offset must be selected so that the software limit switches remain within the permissible numerical range after the function is called.
   - Rotary axis:
     Rule for the zero offset:
     value of zero offset ≤ end of the rotary axis.
2. Set the relevant trigger bit.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
36.6	ZOFF_EN	BOOL	FALSE	1 = set zero offset
86.0	ZOFF	DINT	L#0	Zero offset

Effects on a linear axis

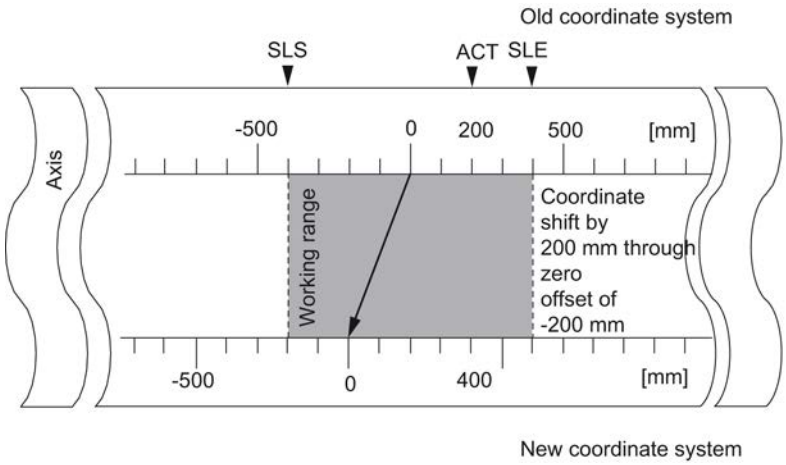
The example of a zero offset of -200 mm demonstrates that this function shifts the coordinate system in positive direction. Resultant effects:

- The working range is **not** physically shifted.
- The individual points (such as the software limit switches) are assigned new coordinate values.
- The cams retain their coordinate values, and are therefore located at a different physical position.
- If the axis is synchronized and cam processing is enabled, the positional process value may skip cam edges or entire cams as a result of this function.
- Status changes of the cam which would normally trigger an interrupt may be lost.

Note

Make allowances for the switching characteristics of time-based cams (Page 9361).

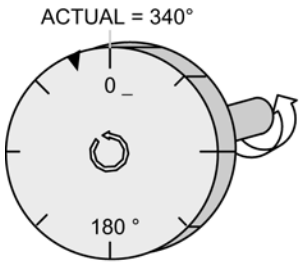
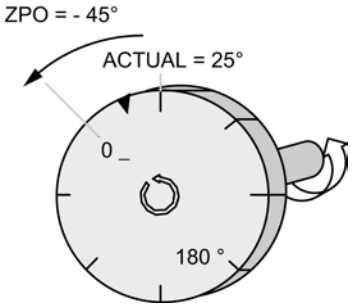
The following table shows how the coordinate system is offset by "Zero offset".

Zero offset	SLS [mm]	REF [mm]	ACT [mm]	SLE [mm]
	-400	-200	200	400
	-200	0	400	600

Effects on a rotary axis

The example of a -45° zero offset demonstrates how this function rotates the coordinate system.

The following table shows how the coordinate system is rotated by "Zero offset".

Tool when ACT = 340°	Tool when ACT = 25°
	
The zero mark rotates by -45 degrees. All axis points are assigned new coordinate values.	

Including a **ZPO_{old} = 0**, the result is a new value of 385°

As process value resumes at 0 at the end of the rotary axis operating in positive directional rotation, the calculation returns an process value of 25°:

$Coordinate_{new}$ = $Coordinate_{old}$ - (ZPO_{new} - ZPO_{old}) - end of rotary axis

The **end of rotary axis** value only needs to be subtracted when

$Coordinate_{old}$ - (ZPO_{new} - NPV_{old}) is greater than the end of rotary axis.

Loss of synchronization

If synchronization is lost due to an error, or reset due to "retrigger reference point", a zero offset is **retained**.

Cancelling function

A zero offset of 0 resets any existing zero offset.

Set reference point

Definition

The "set reference point" function is used to synchronize the axis. The function shifts the working area. All shifts generated by a zero offset or "set actual value" are retained.

The function projects the operating range on the axis. Different value settings will thus move the working range to different positions within the physical range of the axis.

Requirements

Cam processing must be disabled.

Function sequence

1. Write the value for the reference point coordinate to the channel DB.
   - Linear axis:
     The reference point coordinates must not lie outside the software limit switches. This also applies to the reference point coordinate in a shifted coordinate system.
   - Rotary axis:
     Rule for the reference point coordinate:
     0 ≤ reference point coordinate < end of rotary axis
2. Set the relevant trigger bit.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
36.3	REFPT_EN	BOOL	FALSE	1 = set reference point coordinate
98.0	REFPT	DINT	L#0	Reference point coordinate
25.0	SYNC	BOOL	FALSE	1 = axis synchronized

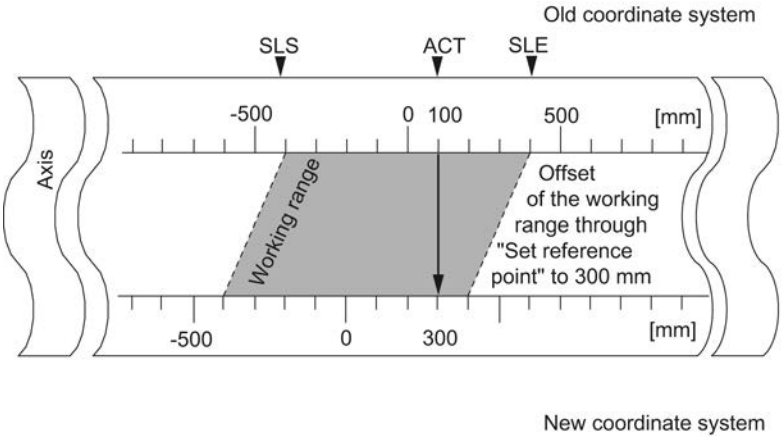
Effects of the function

The example "Set reference point" to 300 mm demonstrates how this function shifts the axis operating range.

This has the following effects:

- The actual position is set to the value of the reference point coordinate.
- The working range is physically shifted on the axis.
- The various points retain their original coordinates, but are now at new physical positions.
- The SYNC bit is set in the checkback signals.

The following table shows how "Set reference point" shifts the operating range on the axis.

Set reference point	SLS [mm]	REF [mm]	ACT [mm]	SLE [mm]
	-400	-200	100	400
	-400	-200	300	400

Special features of the absolute encoder

This function is necessary for absolute encoder adjustment (Page 9390).

Change cam edges

Definition

The "change cam edges" function can be used to change the cam start and, for position-based cams, the end of a specific position-based cam in runtime.

Requirements

The cam you want to change must be valid.

Function sequence

1. Enter the cam number in the channel DB.
2. At a position-based cam:
   Enter the cam start and cam end in the channel DB.
   For a time-based cam:
   Enter the cam start value in the channel DB.
3. Set the relevant trigger bit.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
36.7	CH01CAM_EN	BOOL	FALSE	1 = write function for cam edges (1 cam)
102.0	CAM_NO	INT	0	Cam number

Address	Name	Type	Initial value	Comment
104.0	CAM_START	DINT	L#0	Cam start
108.0	CAM_END	DINT	L#0	Cam end

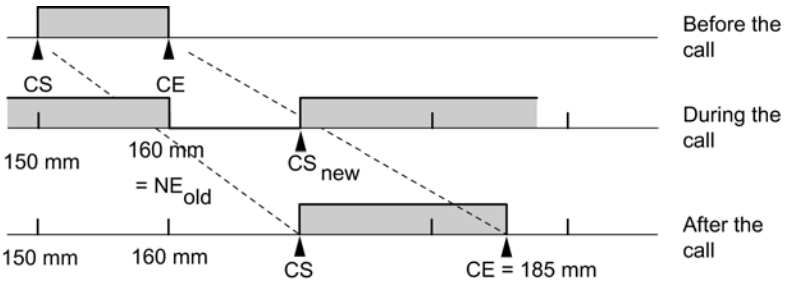
Effects of the function

The FM x52 shifts first the on edge and then the off edge of the cam. This sequence does not depend on the direction in which the cam is shifted.

Exception:

The sequence described above may briefly generate an inverse cam (Page 9340) if the new cam start is greater than the old cam end.

The figure below shows the each stage in cam edge change.



Note

If a hardware interrupt has been enabled for this cam, FM x52 can trigger one or two hardware interrupts when it detects the inverse cam depending on parameter assignment.

Changing the on and/or off edge can result in a cam edge or the entire cam being skipped.

Make allowances for the switching characteristics of time-based cams (Page 9361).

Cam status changes that would normally trigger a hardware interrupt can be lost.

Reading modified values

You can read out the modified values by calling one of the jobs CAM1RD_EN to CAM8RD_EN.

Cancelling the function

The modified values are lost when you restart the module.

Perform "Fast Cam Parameter Change"

Definition

The "Fast cam parameter change" function can be used to modify a group of up to 16 cams in runtime.

Requirements

The cams you want to modify must be valid.

Function sequence

1. Enter the number of cams to be modified in the channel DB.
2. Enter the number of the first cam to be modified in the channel DB.
3. Set the trigger bits for the required modifications.
4. Declare the new values at the channel DB.
5. Repeat steps 2 to 4 for each cam to be modified.
6. Set the relevant trigger bit in the channel DB.

Data used in the channel DB

Address Absolute	Name	Type	Initial value	Comment
37.0	CH16CAM_EN	BOOL	FALSE	1 = write fast cam parameter change function (16 cams)
176.0	C_QTY	BYTE	B#16#0	Number of cams to modify
177.0	DIS_CHECK	BOOL	FALSE	1 = disable data check

Address relative	Name	Type	Initial value	Comment
+0.0	CAM_NO	BYTE	B#16#0	Number of the cam to modify
+1.0	C_EFFDIR	BOOL	FALSE	1 = change the effective direction of the cam
+1.1	C_CBEGIN	BOOL	FALSE	1 = change the cam start to the value CBEGIN
+1.2	C_CEND	BOOL	FALSE	1 = change the cam end / activation time to the value CEND
+1.3	C_LTIME	BOOL	FALSE	1 = change the lead time to the value LTIME
+1.4	CAM_OFF	BOOL	FALSE	1 = deactivate the cam during the cam modification
+1.5	EFFDIR_P	BOOL	FALSE	1 = positive effective direction (plus)
+1.6	EFFDIR_M	BOOL	FALSE	1 = negative effective direction (minus)
+2.0	CBEGIN	DINT	L#0	New cam start

Address relative	Name	Type	Initial value	Comment
+6.0	CEND	DINT	L#0	New cam end / new activation time
+10.0	LTIME	INT	L#0	New lead time

Deactivating a cam when making modifications

To maintain consistency, always deactivate the cam (CAM_OFF) when modifying its start and end settings.

Data check by the module

Use the DIS_CHECK (channel DB) parameter to specify whether or not to disable the validation of transferred data by FM x52. If you disable the data check, you must take steps to verify that only permitted values are transferred. If you transfer illegal values without a check, the module can behave unexpectedly.

- FALSE: The module checks all data to be transferred
- TRUE: The check of cam parameter data is disabled. This allows for faster activation of the data to be changed on FM x52.
  Regardless of this function, the module always checks whether
  - the axis is assigned
  - the number of cams to be changed (C_QTY) is valid
  - the cam (cam number) to change is valid.

The data are only enabled on the module after having been checked and found faultless.

Any faulty data are rejected.

Effects of the function

Note

Make allowances for the switching characteristics of time-based cams (Page 9361).

Reading modified values

You can read the modified values by calling one of the jobs CAM1RD_EN to CAM8RD_EN.

Cancelling the function

The modified values are lost when you restart the module.

Executing "Length measurement" and "Edge detection"

Definition

You can establish the length of a workpiece using the "Length measurement" and "Edge detection" functions.

Length measurement and edge detection are active and remain active until you disable these functions or select a different measuring method. The CAM_CTRL (Page 7157) instruction enables length measurement if you select both measuring methods in parallel.

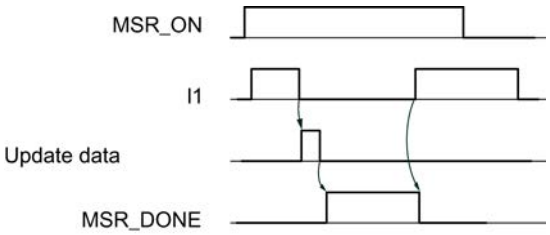
Requirement

A **bounce-free** switch must be connected to input I1.

Function sequence

The FM 352 updates the data on the module at different times depending on the type of measurement. FM 352 reports each update at a parameter on the feedback interface.

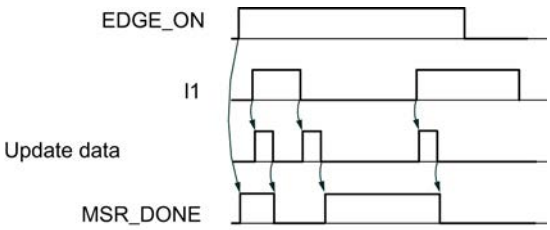
Length measurement



1. Set the function switch for "length measurement".
2. A positive edge at input I1 starts the length measurement.
3. The negative edge at input I1 stops the current measurement. The FM 352 updates the initial value, final value and length data.
4. When parameter MSR_DONE is set, FM 352 reports the data update. The parameter indicates completion of the measurement. The results of the measurement can be read out.
5. The start of the next measurement with a rising edge at I1 resets the MSR_DONE parameter.

The FM 352 does not update the data if the function is disabled during length measurement. The MSR_DONE parameter remains reset.

Edge detection



1. Enter a value for the minimum edge distance in the parameter DB. Write and enable machine data.
2. Set the "edge detection" function switch. This sets the MSR_DONE parameter.
3. The positive edge at input I1 starts edge detection. The results of the measurement are updated and can be read out, and the start value of the measurement is entered; the end value and length assume the value -1.
4. After the update, FM 352 reports the change by resetting the MSR_DONE parameter.
5. The negative edge at input I1 stops the current measurement. The FM 352 updates the data for the end measurement and length values.
6. After the update, FM 352 reports the change by setting the MSR_DONE parameter. The results of the measurement can be read out.
7. The start of the next measurement with a rising edge at I1 resets the MSR_DONE parameter.

The FM 352 does not update the data if the function is disabled during edge detection. The MSR_DONE parameter remains reset.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
25.1	MSR_DONE	BOOL	FALSE	1 = length measurement completed
34.0	EDGE_ON	BOOL	FALSE	1 = edge detection on
34.2	MSR_ON	BOOL	FALSE	1 = length measurement on
38.2	MSRRD_EN	BOOL	FALSE	1 = read measured values
112.0	BEG_VAL	DINT	L#0	Initial value
116.0	END_VAL	DINT	L#0	End value
120.0	LEN_VAL	DINT	L#0	Length

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
4.0	EDGEDIST	DINT	L#0	**Minimum edge distance for edge detection** Valid value range: • 0 ... 1 000 000 000 µm

The minimum edge distance is used to define a range after the start of a measurement is detected with an edge detection. The measurement will be discarded if the measuring operation ends within this range.

The start of the measurement is not signaled unless the minimum edge distance has been traveled.

Conditions of length measurement

- The CPU program requires an interval of sufficient length between the off and on edges at input I1 in order to be able to evaluate the result of the measurement before a new measurement is triggered.
- The minimum interval between the positive and negative edges at input I1 and between the negative edge and the next positive edge at input I1 must be greater than 2 ms.

Faulty measurement

The FM 352 returns the length value -1 in the event of errors in length measurement / edge detection.

A "length measurement" or "edge detection" may only perform up to 126 zero passes in one direction. The zero pass represents the rotary axis transition from the end of rotary axis value to 0 and vice versa. The FM 352 reports incorrect "length measurement" or "edge detection" if it detects more than 126 zero passes in one direction, regardless of whether or not it then detects any zero passes in the opposite direction.

A length measurement is also considered faulty if:

- the length measured at a rotary axis is greater than 2^{31}.
- the on and off edges are detected simultaneously by the FM 352 (for example due to switch bounce).

This event can be signaled as a hardware interrupt.

Shift of the coordinate system during a length measurement

Conditions under which a shift of the coordinate system will influence the measured length:

- You are using an incremental encoder or an initiator or are operating the FM 352 in simulation mode.
- You are executing a "set reference point" or "retrigger reference point" function while a length measurement is active.

Example

To utilize the above mentioned influences on the measured length:

Your system always develops slip when you perform a length measurement.

The "Retrigger reference point" function can be used to correct this slip to allow correct length measurement values to be output.

Retrigger reference point

Definition

The "Retrigger reference point" function can be used to synchronize the axis as a reaction to a recurring external event.

The function remains active until you deactivate it.

Requirements

- You are using an incremental encoder or an initiator.
- The external event may represent the zero mark signal of an incremental encoder or reference point switch at input I2.

Function sequence

1. Set the value for the reference point coordinate at the parameter DB.
2. Set the type of "retrigger reference point" at the parameter DB.
   Options:
   – Evaluation of the zero mark of the encoder only (RETR_TYPE = 7).
   – Evaluation of the reference point switch only (RETR_TYPE = 6).
   – Evaluation of the zero mark signal
     in positive direction: Evaluation of the first rising edge of the zero mark after passing the reference point switch in positive direction (RETR_TYPE = 0).
     in negative direction: Evaluation of the first falling edge of the zero mark after passing the reference point switch in negative direction (RETR_TYPE = 1).
3. Write and enable machine data.
4. Set the function switch in the channel DB.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
34.3	REFTR_ON	BOOL	FALSE	1 = retrigger reference point
25.0	SYNC	BOOL	FALSE	1 = axis is synchronized

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
44.0	REFPT	DINT	L#0	Reference point coordinate
52.0	RETR_TYPE	DINT	L#0	Reference point retrigger mode

Effects of the function

- FM x52 evaluates the zero mark and reference point switch in line with the axis direction of motion.
  - It evaluates the positive edges when the axis moves in positive direction.
  - It evaluates the negative edges when the axis moves in negative direction.
- It sets the actual position to the value of the reference point coordinate.
- The working range is physically shifted on the axis.
- The various points retain their original values, but are now at new physical positions.
- Cam status changes that would normally trigger a hardware interrupt can be lost.
- The SYNC bit in the checkback signals is set.

Note

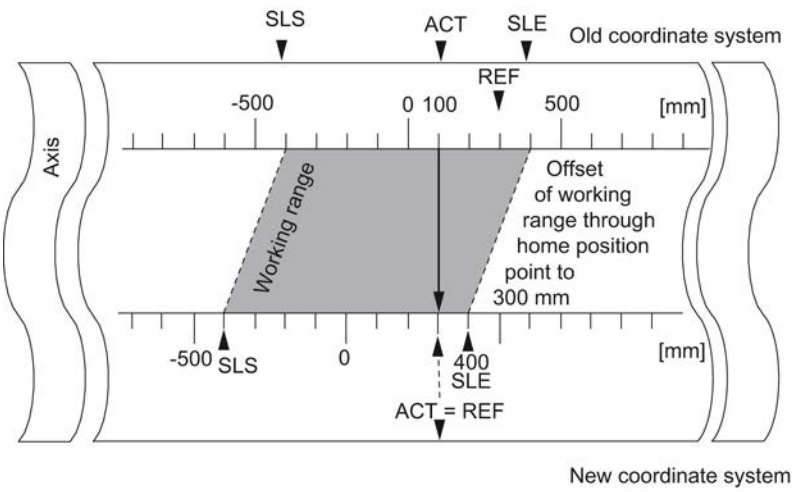
Make allowances for the switching characteristics of time-based cams (Page 9361).

Example

Rules for the example:

- The module evaluates the positive edges of the reference point switch and zero mark signals (axis moving in positive direction).
- Value of the reference point coordinate = 300 mm.
- No zero offset is active at the time of execution.

The following table shows how "Retrigger reference point" shifts the operating range on the axis.

Retrigger reference point	SLS [mm]	REF [mm]	ACT [mm]	SLE [mm]
	-400	300	100	400
	-400	300	300	400

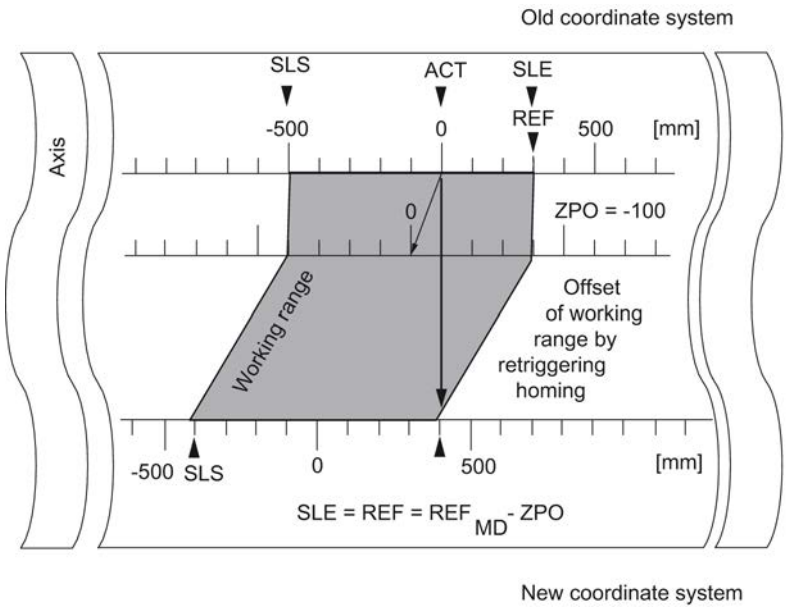
Inclusion of a zero offset

Any active zero offset is included in the "Retrigger reference point" function. The reference point coordinate setting is thus calculated according to the formula:

Ref = Ref_{MD} - Zero offset

Ref_{MD} is the value of the reference point coordinate store in machine data.

The following table shows how "Retrigger reference point" with zero offset shifts the operating range on the axis.

Retrigger reference point	SLS [mm]	REF [mm]	ACT [mm]	SLE [mm]
	-500	300	0	300
	-400	400	100	400
	-400	400	400	400

Deactivate software limit switch

Definition

Use the "Deactivate software limit switch" function to disable monitoring of the software limit switches at a linear axis.

The function remains active until you deactivate it. This re-activates the software limit switches originally parameterized.

Function sequence

Set the function switch in the channel DB.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
34.4	SSW_OFF	BOOL	FALSE	1 = Software limit switches deactivated

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
64.0	SSW_STRT	DINT	L#-1000000000	Software limit switch start
68.0	SSW_END	DINT	L#1000000000	Software limit switch end

Effects of the function

- simulation
  - Simulation mode stops when the axis overruns a software limit switch.
  - You can resume simulation mode by disabling software limit switch monitoring. The axis moves in the defined direction.
- Zero offset when monitoring is disabled
  If a zero offset is set in which software limit switches are located within traversing range limits, the actual value may still be outside the valid number range.
- Cams located outside the parameterized software limit switches can be switched.

CAUTION

Risk of material damage!

If you reduce the traversing range by setting software limit switches to protect your system, you risk damage to your equipment when these switches are disabled.

In the planning and engineering phases for your plant, you should be certain that the drive can travel in the entire physical traversing range.

Executing "Simulation"

Definition

The "Simulation" function allows you to activate the cam controller without connected encoders.

Function sequence

1. Set the simulation velocity at the parameter DB.
2. Write and enable machine data.
3. Set either a positive or negative simulation direction at the channel DB.
4. Set the function switch in the channel DB.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
15.2	DIR_M	BOOL	FALSE	1 = simulation in negative direction
15.3	DIR_P	BOOL	FALSE	1 = simulation in positive direction
34.1	SIM_ON	BOOL	FALSE	1 = simulation on

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
84.0	SIM_SPD	DINT	L#0	Simulation speed

Effects when simulation mode is activated

- Encoder signals will be ignored.
- The encoder input monitoring functions are disabled.
- All encoder errors reported will be reset.
- FM x52 simulates axis motion at a constant simulation velocity.
- Cam processing is disabled while simulation mode is active. However, you can then enable these operations again. Synchronism will be retained in this case.
- Starting at the current actual value, the actual position value changes dynamically based on the simulation velocity and direction.

Effects when simulation mode is deactivated

- Cam processing will be disabled.
- The synchronization of an incremental encoder or initiator will be cleared.. The actual value will be reset to the value of the reference point coordinate.
- The module reports the actual position value which corresponds with the absolute encoder value. The module then evaluates the encoder signals as defined at the machine parameters.

Limit values

The minimum and maximum simulation velocity depend on the resolution (Page 9394).

Velocity

There may be differences between the set velocity and the actual velocity (Page 9394) at which the module operates.

Read "count values of counter cam tracks"

Definition

The "Counter cam track counter values" function is used to read out the current counter values.

Function sequence

1. Specify the counter cam tracks and the high count values in the machine data.
2. Write and enable machine data.
3. Enable the count function.
4. The module sets the count value to its high limit.
5. The count value decrements by the count of 1 at each positive edge of the track result signal.
6. Set the trigger bit at the channel DB to read the count values.
7. The module writes both count values to the channel DB. The module outputs a 0 value for tracks not assigned as a counter cam track.
8. The module sets track identifier bit = 1 at the counter cam track when the count value = 0.
9. It resets the track identifier bit = 0 at the next negative edge of the track result signal, and resets the counter to its high limit.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
15.5	CNTC0_EN	BOOL	FALSE	1 = enable count function of counter cam track 0
15.6	CNTC1_EN	BOOL	FALSE	1 = enable count function of counter cam track 1
38.3	CNTTRC_EN	BOOL	FALSE	1 = read count values of counter cam tracks
124.0	CNT_TRC0	INT	0	Current count value of counter cam track 0
126.0	CNT_TRC1	INT	0	Current count value of counter cam track 1

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
99.0	SPEC_TRC0	BOOL	FALSE	1 = track 0 is counter cam track
99.1	SPEC_TRC1	BOOL	FALSE	1 = track 1 is counter cam track
100.0	CNT_LIM0	DINT	L#2	High count value of counter cam track 0
104.0	CNT_LIM1	DINT	L#2	High count value of counter cam track 1

Read out "Position and track data"

Definition

The "Position and track data" function can be used to read the current position, the velocity and the track identifier bits. The track identifier bits are recorded before being logically linked to machine and channel data.

The algorithm implemented in the FM x52 calculates velocity changes greater than 1 pulse/ 4 ms. The indicated velocity includes this inaccuracy, and is thus unsuitable in particular for closed-loop control. The internal velocity value used for dynamic cam control offers a higher precision.

Function sequence

1. Set the trigger bit in the channel DB.
2. The data are stored in the channel DB.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
38.4	ACTPOS_EN	BOOL	FALSE	1 = read position and track data
128.0	ACTPOS	DINT	L#0	Current position
132.0	ACTSPD	DINT	L#0	Current velocity
136.0	TRACK_ID	DWORD	DW#16#0	Track identifier bits of tracks 0 to 31

Read out "Encoder data"

Definition

The "Encoder data" function can be used to read out current encoder data and the value for absolute encoder adjustment.

Requirements

You can read out the value for absolute encoder adjustment (Page 9390) after you have implemented the "Set reference point" (Page 9366) function.

Function sequence

1. Set the trigger bit in the channel DB.
2. The data are stored in the channel DB.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
38.5	ENCVAL_EN	BOOL	FALSE	1 = read encoder values
140.0	ENCVAL	DINT	L#0	Encoder value / counter value (internal representation)
144.0	ZEROVAL	DINT	L#0	Counter value at the last zero mark (internal representation)
148.0	ENC_ADJ	DINT	L#0	Absolute encoder adjustment

Read out "Cam and track data"

Definition

The "Cam and track data" setting can be used to read the current cam and track identifier bits and the position. The track identifier bits are recorded before being logically linked to machine and channel data.

Function sequence

1. Enter type ID = 1 at the FM_TYPE parameter of the channel DB. This allows you to read 24 bytes of cam and track data.
   If you enter type ID = 0, only the cam identifier bits (16 bytes) will be read.
2. The data are stored in the channel DB.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
12.0	FM_TYPE	BOOL	FALSE	0 = FM 352 to V4.0 1 = FM 352 V5.0 or higher and FM 452
38.6	CAMOUT_EN	BOOL	FALSE	1 = read cam and track data
152.0	CAM_00_31	DWORD	DW#16#0	Cam identifier bits for cams 0 to 31
156.0	CAM_32_63	DWORD	DW#16#0	Cam identifier bits for cams 32 to 63
160.0	CAM_64_95	DWORD	DW#16#0	Cam identifier bits for cams 64 to 95
164.0	CAM_96_127	DWORD	DW#16#0	Cam identifier bits for cams 96 to 127
168.0	TRACK_ID1	DWORD	DW#16#0	Track identifier bits of tracks 0 to 31
172.0	ACTPOS1	DINT	L#0	Current position

Setting "Control signals for the cam controller"

Definition

The "Control signals for the cam controller" function can be used to enable cam processing and the tracks.

Function sequence

1. Set the required bits at the channel DB.
2. The data are transferred to the module each time the CAM_CTRL (Page 7157) instruction is called.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
15.4	CAM_EN	BOOL	FALSE	1 = enable cam processing
16.0	TRACK_EN	WORD	W#16#0	Enable cam tracks 0 to 12 Bit 0 = track 0

Effects

Cam processing will be started or stopped based on the enable status.

The track identifier bits of enabled tracks are transferred to the track signals and digital outputs.

Querying "Checkback signals for the cam controller"

Definition

The "Checkback signals for the cam controller" function provides information about the current status of the cam controller and track signals. Consistency between the reported position and track signals is not guaranteed.

Function sequence

The data is stored in the channel DB each time the CAM_CTRL (Page 7157) instruction is called.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
23.4	CAM_ACT	BOOL	FALSE	1 = cam processing busy
26.0	ACT_POS	DINT	L#0	Current axis position
30.0	TRACK_OUT	DWORD	DW#16#0	Current track signals of tracks 0 to 31 Bit 0 = track 0

Querying the "Checkback signals for diagnostics"

Definition

The "Checkback signals for diagnostics" function is used to report diagnostics events.

Function sequence

1. If the module enters a new event in the diagnostic buffer, it sets the DIAG bit in the feedback interface. Error (Page 9425) events are logged in the diagnostics buffer.
2. The module sets the DATA_ERR bit in the feedback interface when it detects incorrect data in a write job. The cause of the error is logged to the diagnostic buffer.
3. The feedback interface stores the CAM_CTRL (Page 7157) instruction in the channel DB.
4. The module resets the DIAG bit to 0 after the diagnostic buffer has been read by the CAM_DIAG (Page 7158) instruction.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
22.2	DIAG	BOOL	FALSE	1 = diagnostic buffer modified
22.4	DATA_ERR	BOOL	FALSE	1 = data error

Configuring FM 352

Configuring FM 352

Measuring system

Selecting the system of units

Selecting the system of units

For the input and output of the data, you can select from the following systems of units:

- Millimeter (default setting)
- Inches
- Degrees
- Pulses

Note

If you change the system of units in the "Measuring system" dialog, the values will be converted to the new system. This may lead to rounding errors.

If you change the system of units via the machine date, the values are not automatically converted.

Any change of the system of units from or to "pulses" disables cam processing and prevents axis synchronization.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
8.0	UNITS	DINT	L#1	**Measuring system** Valid values: • 1 = 10^{-3} mm • 2 = 10^{-4} inch • 3 = 10^{-4} degrees • 4 = 10^{-2} degrees • 5 = pulses • 6 = 10^{-3} degrees

Standard system of units in the online help

To specify the limits, we use the mm system of units. To define the limits in other systems of units, convert the values as shown below:

To convert....		you calculate...
mm → inch		Limit value (inches) = limit value (mm) × 0.1 [1]
mm → degrees	10^{-4} (4 decimal places)	Limit value (degrees) = limit value (mm) x 0.1
	10^{-3} (3 decimal places)	Limit value (degrees) = limit value (mm) x 1
	10^{-2} (2 decimal places)	Limit value(degrees) = limit value (mm) x 10
mm → pulse		Limit value (pulses) = limit value (mm) x 1000

[1] When set to the maximum value, the number of decimal places affects the number of digits before the decimal point. In the inch measurement system, 4 decimal places are used; you can therefore enter a maximum of 100,000.0000 inches. The millimeter measurement system uses 3 decimal places; a maximum of 1,000,000.000 mm can be entered.

Axis

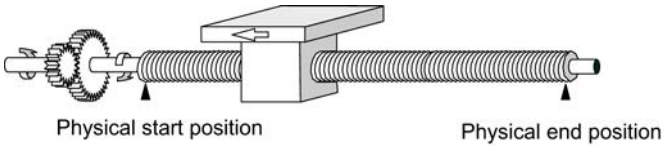
Specifying axis type

Selecting axis type

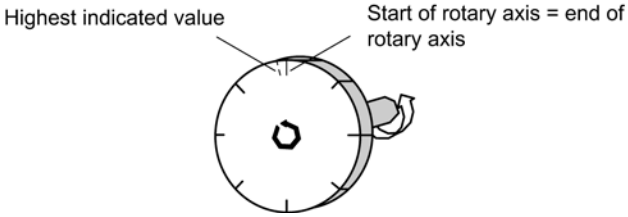
Axis types

The cam controller supports the following axis types:

- Linear axis
  A linear axis is an axis which has a limited physical traversing range.



- Rotary axis
  The rotary axis is an axis whose traversing range is not restricted by mechanical stops.



Data used in the parameter DB

Address	Name	Type	Initial value	Comment
12.0	AXIS_TYPE	DINT	L#0	**Axis type** Valid values: • 0 = linear axis • 1 = Rotary axis

Specifying software limit switches

Software limit switch

Software limit switches only exist for the linear axis.

The operating range is the range limited by the software limit switches. The FM x52 can monitor the operating range limits.

The software limit switch start (SLS) must always be smaller than the software limit switch end (SLE).

The software limit switches are active when FM x52 is synchronized.

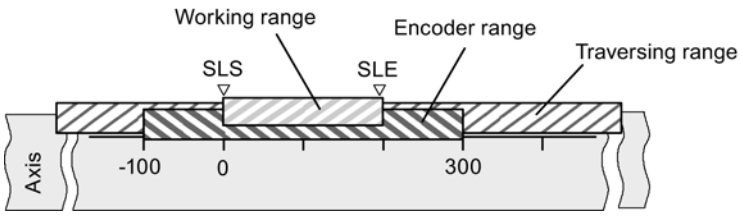
- When using an incremental encoder or initiator:
  The axis is initially not synchronized after each FM x52 startup. The parameterized software limit switches are not monitored until the axis is synchronized.
- When using an absolute encoder (SSI):
  The FM x52 is synchronized once it has received a complete frame without errors. The software limit switches are monitored from this time on. The absolute encoder you use must cover at least the operating range including the software limit switches.

Relationship between the operating range, encoder range and traversing range

- The operating range is the range which you specify for your job using the software limit switches.
- The encoder range represents the range explicitly covered by the encoder. For a linear axis, the module assigns the encoder range symmetrically across across the operating range. In other words, the module shifts the encoder range in order to equalize the distances between the software limit switches and the ends of the encoder range (see figure below).
- This traversing range is the value range the FM x52 can process. The traversing range is dependent on the resolution.

The following applies: Traversing range ≥ encoder range ≥ operating range

The following figure shows the relationship between software limit switches, the operating range, the encoder range and the traversing range.



Data used in the parameter DB

Address	Name	Type	Initial value	Comment
64.0	SSW_STRT	DINT	L# -100 000 000	**Software limit switch start**
68.0	SSW_END	DINT	L# 100 000 000	**Software limit switch end** Valid value range: • -1,000,000,000 µm to 1,000,000,000 µm

Specifying end of rotary axis

End of rotary axis

The "End of rotary axis" value is the theoretical maximum actual value of the axis. However, the theoretical maximum value is never indicated, because it also represents the physical start position of the rotary axis (= zero).

The highest rotary axis value displayed is:

End of rotary axis [µm] - resolution [µm / pulse] x 1 [pulse]

Example: End of rotary axis = 1000 mm

The displayed value jumps:

- from 999 mm to 0 mm when the rotational direction is positive.
- from 0 mm to 999 mm when rotational direction is negative.

Rotary axis with absolute encoders

The rotary range (0 to end of rotary axis) of a rotary axis with an absolute encoder must correspond exactly to the total number of steps of the absolute encoder:

$$\text{Rotary axis end[µm]} = \text{Number of revolutions(encoder)} * \frac{\text{steps(encoder)[pulse]}}{\text{Revolution}} * \text{RES}[\frac{\text{µm}}{\text{Pulse}}]$$

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
16.0	ENDROTAX	DINT	L#100000	**End of rotary axis** Valid value range: • 1 µm to +1,000,000,000 µm

Specifying synchronization

Selecting "Retrigger reference point" type

Retrigger reference point

By setting the "Retrigger reference point (Page 9375)" type, you define the conditions for axis synchronization in operation with an incremental encoder or an initiator.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
52.0	RETR_TYPE	DINT	L#0	**Type of reference point retrigger** Valid values: • 0 = reference point switch and zero mark direction + • 1 = reference point switch and zero mark direction - • 6 = reference point switch only • 7 = zero mark only

Specifying reference point coordinate

Reference point coordinate

- **Incremental encoder and initiator:**
  Using the "Retrigger reference point" function switch and a synchronization event defined by the "Retrigger reference point" type (Page 9388), the reference point coordinate is assigned to this event.
- **Absolute encoder (SSI)**
  A parameterized axis with absolute encoder is always synchronized after transmission of the first error-free SSI frame.
  For more information, refer to the description of absolute encoder adjustment (Page 9390) which explains the interaction of the absolute encoder adjustment with other data.

The value of the reference point coordinate must lie within the operating range:

- **Linear axis**
  Including the software limit switches.
- **Rotary axis**
  Greater than or equal to 0 and less than the "end of rotary axis" value (0 ≤ reference point coordinate < end of rotary axis).

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
44.0	REFPT	DINT	L#0	**Reference point coordinate** Valid value range: • -1,000,000,000 µm to +1,000,000,000 µm

Specifying absolute encoder adjustment

Determining the absolute encoder adjustment

Definition

Absolute encoder adjustment is only necessary for an absolute encoder.

The absolute encoder adjustment and reference point coordinate map the encoder's range of values to the axis coordinate system. The encoder value corresponding to the reference point coordinate on the axis is determined. The value must be less than the total number of steps of the absolute encoder.

Additional information can be found under "Absolute encoder adjustment procedure (Page 9390)" and in an example (Page 9391).

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
48.0	ENC_ADJ	DINT	L#0	**Absolute encoder adjustment** Valid value range: • 0 to (2^{25} - 1)

Data used in the channel DB

Address	Name	Type	Initial value	Comment
98.0	REFPT	DINT	L#0	**Reference point coordinate** Valid value range: • -1,000,000,000 µm to +1,000,000,000 µm

Absolute encoder adjustment procedure

Determining the correct absolute encoder adjustment

After initial parameter assignment, further steps must be taken to ensure that a correct relationship is created between the encoder and coordinate system. The procedure shown in the next section is based on the use of the parameter assignment dialogs.

1. Move the axis to a defined, reproducible point to which a unique coordinate is assigned. This can, for example, be the "software limit switch end".
2. Call the "Set reference point (Page 9366)" setting using the coordinate defined in point 1. FM x52 now determines an encoder value for the reference point coordinate entered in the channel DB (REFPT in the channel DB), namely the absolute encoder adjustment. You can read this value with the "Encoder data (Page 9381)" setting.
3. Enter the value read in the "absolute encoder adjustment" field.

4. Save your parameter assignment to the corresponding parameter DB using the export function.
5. Download the data in the HWCN to the CPU.
6. Restart the CPU to activate the data.

Note

You perform this adjustment once during commissioning. Following parameter assignment, the FM x52 is synchronized during start-up as soon as a complete, faultless encoder frame is received after start-up.

Otherwise: Mechanical adjustment of the encoder

A correct relationship between the coordinate system and the encoder can be achieved as follows:

1. Traverse the axis to a defined, reproducible point to which a unique coordinate is assigned. This can, for example, be the "software limit switch start".
2. Enter this coordinate value in the machine data as "reference point coordinate".
3. Read the encoder value displayed at this position with the "Encoder data (Page 9381)" setting.
4. Enter this value in the machine data as the "absolute encoder adjustment".

After the parameter assignment a correct actual value will always be shown.

Instead of steps 3 and 4, you can also set the encoder to zero using "Reset", if available. Then enter the value "0" in the machine data as the absolute encoder adjustment.

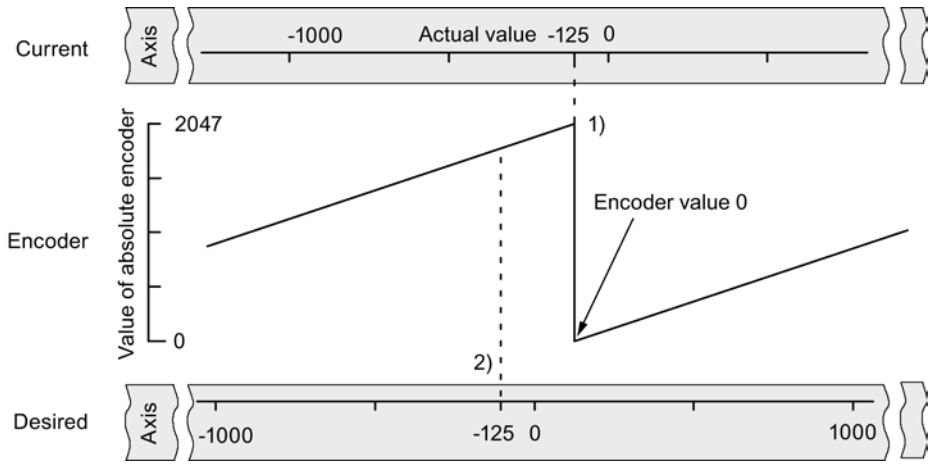
Example of absolute encoder adjustment

Example of an absolute encoder adjustment

The following assumptions apply to the example:

- Reference point coordinate = -125 mm
- Operating range from SSW_STRT = -1000 mm to SSW_END = 1000 mm
- Absolute encoder adjustment = 0
- Encoder range = 2048 increments with a resolution of 1 mm/pulse
- The absolute encoder used cannot be adjusted mechanically with precision and it also is not possible to set the encoder value specifically.

The following figure shows an example of absolute encoder adjustment.



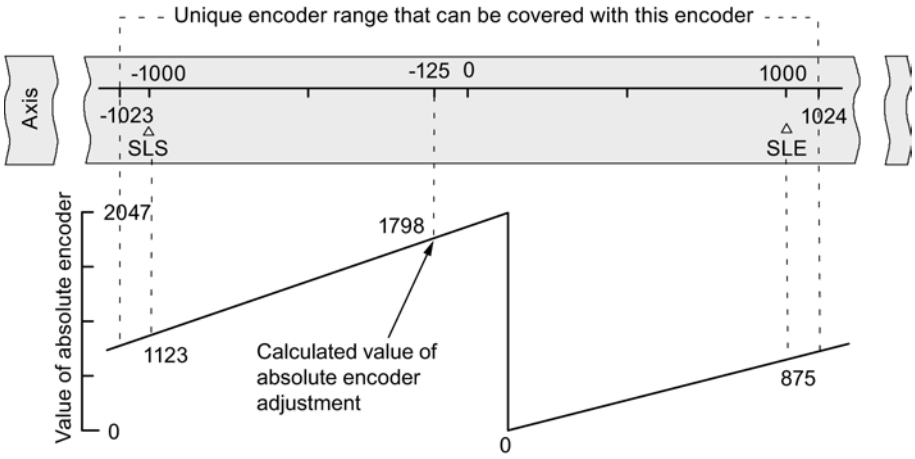
1) Assignment of the coordinate system to encoder values, based on the absolute encoder adjustment. The encoder value 0 corresponds to the actual value -125.

2) Required assignment of the coordinate system to the encoder. At this position the coordinate should be -125.

Result in accordance with "Set reference point"

After "Set reference point" the relationship looks as follows:

The reference point coordinate on the axis (-125) is assigned to the encoder value (1798) determined from absolute encoder adjustment. This assignment is set out in the figure below.



The encoder supplies 2048 unique values. The operating range is defined by the software limit switches. The selected resolution of 1 mm per pulse means the encoder can, however, cover a greater operating range than the one set by the software limit switches.

With the set resolution, the operating range is already covered with 2001 values. Therefore, in the example, 47 pulses "remain" and these are symmetrically positioned around the operating range.

Setting hysteresis

Hysteresis

The hysteresis (Page 9345) value range depends on the resolution (Page 9398).

Maximum input value:

- For linear axes: max. input value < ¼ of the operating range
- For rotary axes: max. input value < ¼ of the rotary axis range

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
80.0	HYS	DINT	L#0	**Hysteresis** Valid value range: • 0 to 65 535 [Imp] x resolution [$^{\mu m}/_{Imp}$]

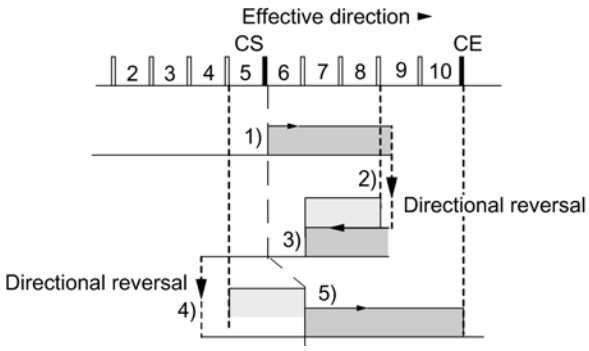
Position-based cams and hysteresis

A position-based cam is activated when the following conditions are met:

- The detected actual value lies within the position-based cam.
- Hysteresis is not active.

Switching points may vary depending on the position at which the change in direction takes place.

The figure below shows the switching characteristics of a position-based cam with hysteresis.



	Cam
	Hysteresis
1)	Cam is activated, as the cam start is crossed in the positive effective direction.
2)	Cam remains active due to hysteresis, irrespective of changes of direction.
3)	Cam is deactivated when hysteresis ends.
4)	Cam remains deactivated after a direction change.
5)	Cam is activated again at the end of hysteresis.

Note

When the direction is changed, hysteresis cannot reduce a cam actuation time shorter than the hysteresis.

Time-based cams and hysteresis

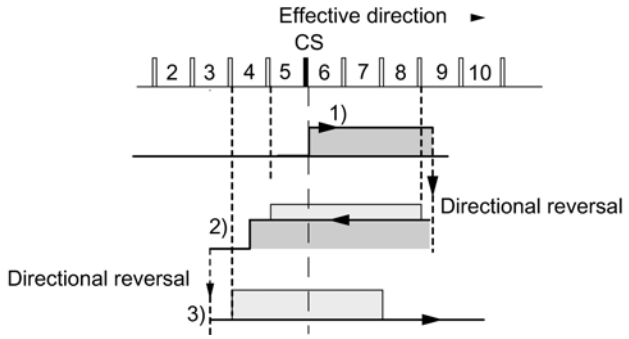
A time-based cam is activated when the following conditions are met:

- The cam start is crossed in the effective direction.
- Hysteresis is not active.

Note

The hysteresis will hide a time-based cam if its range between the reversal point and the cam start is less than the hysteresis.

The figure below shows the switching characteristics of a time-based cam with hysteresis. The figure shows a time-based cam that is **not** activated again.



	Cam
	Hysteresis
1)	Cam is activated, as the cam start is crossed in the positive effective direction.
2)	Cam remains active after the change of direction until the activation time has expired, regardless of hysteresis.
3)	Cam start is hidden by the hysteresis; the cam is not activated.

Specifying simulation speed

Simulation speed

This machine data determines the simulation speed. Please also see the information under "Influence of settings on the switching characteristics of time-based cams (Page 9361)".

The actual simulation speed V_{sim} may deviate from the setting at $V_{sim, V}$, and is calculated using the following formula:

$$V_{Sim} = \frac{6*10exp7*RES}{\text{Integer}\left(\frac{6*10exp7*RES}{V_{Sim, V}}\right)}$$

In this equation:

- V_{sim}: Simulation speed set by FM 352. Unit: $^{\mu m}/_{min}$
- $V_{sim, V}$: simulation speed set in the machine parameters. Unit: $^{\mu m}/_{min}$
- RES: Resolution derived from encoder data. Unit: $^{\mu m}/_{pulse}$
- Integer ( ): Of this expression, only the decimal integer is used in subsequent calculation. This expression must lie within the range of 2 to 65536 in all calculations.

The actual simulation speed changes abruptly as a result of correlations (see above formula).

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
84.0	SIM_SPD	DINT	L#0	**Simulation speed** Valid value range: • 0 = standstill • 5×10^8 = highest setting supported by the module Within this range, the simulation speed depends on the resolution (Page 9398): 1000 x resolution ≤ simulation velocity ≤ 3 x 10^7 x resolution

Specifying minimum edge-to-edge distance

Minimum edge-to-edge distance

This machine data is used to define a range after detection of the start of a measurement for edge detection. The measurement is discarded if the end of the measurement is within this range.

The start of measurement is not reported unless the "minimum edge-to-edge distance" has been traveled.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
4.0	EDGEDIST	DINT	L#0	**Minimum edge-to-edge distance** Valid value range: • 0 ... 1,000,000,000 µm

Encoder

Selecting the signal type

Signal type

The encoder returns position data to the module for evaluation and conversion to an process value based on the resolution.

You can select the signal type from one of the following encoders (Page 9348):

- 5 V differential signal (Page 9350)
  Select the signal type "5 V differential signal" to use a 5 V incremental encoder with differential signal A, /A, B, /B and N, /N.
- 24 V track A+B phase shifted (Page 9350)
  Select signal type "24 V track A+B phase shifted" to use a 24 V incremental encoder with signals A*, B* and N*.
- SSI absolute (Page 9352)
  Select the "SSI absolute" signal type if you use an absolute encoder with a serial interface.
- 24 V initiator (Page 9352)
  Select the signal type "24 V initiator" to use a 24 V initiator with count signal A*.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
20.0	ENC_TYPE	DINT	L#1	**Signal type** Valid values: • 1 = 5 V, incremental • 2 = 24 V, incremental • 3 = SSI 13-bit frame length • 4 = SSI 25-bit frame length • 5 = listen in • 6 = 24 V initiator up (positive) • 7 = 24 V initiator down (negative) • 8 = SSI 13-bit (right-justified) • 9 = SSI 25-bit (right-justified) • 10 = listen in (right-justified)

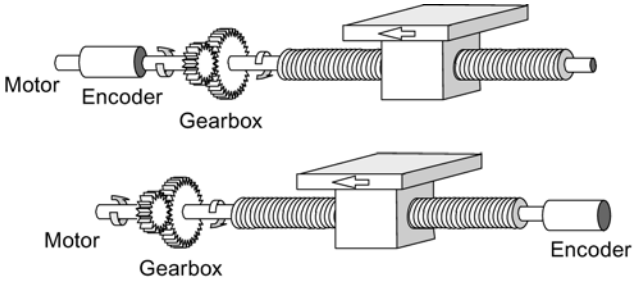
Specifying resolution

Specify distance per encoder revolution

Distance per encoder revolution

You inform the FM x52 about the distance covered by the drive system per encoder revolution using the "Distance per encoder revolution" machine data.

The "Distance per encoder revolution" value depends on the structure of the axis and the way in which the encoder is mounted. You must take all transmission elements, such as couplings and gears, into account. The following figure shows an example of two different encoder positions.



The relationship between the "Distance per encoder revolution" and "Increments per encoder revolution" machine data is described under resolution (Page 9398).

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
24.0	DISP_REV	DINT	L#80000	**Distance per encoder revolution** Valid value range: • 1 µm to 1,000,000,000 µm

Specifying increments per encoder revolution

Increments per encoder revolution

The "Increments per encoder revolution" machine data defines the number of increments an encoder returns per revolution. The FM x52 calculates the resolution based on this value and the "Distance per encoder revolution" machine data.

Incremental encoder

Any value within the range of values can be entered (Page 9350). The module evaluates the increments four times.

Initiator

Any value within the range of values is allowed.

Absolute encoder

Different encoder types have different limits:

Encoder type	Frame length / type	Value range	Can be used as linear axis
Single-turn encoder	13-bit half fir tree	64 ... 8192 in powers of 2	
Single-turn encoder	13-bit right-justified	64 ... 8192 all values	X
Single-turn encoder	25-bit right-justified	64 ... 2^{25} all values	X
Multiturn encoder	25-bit fir tree	64 ... 8192 in powers of 2	
Multiturn encoder	25-bit right-justified	64 ... 2^{24} all values	
Listen-in	Fir tree	64 ... 8192 in powers of 2	
Listen-in	Right justified	64 ... 2^{25} all values	X
Special setting: Multiturn encoder in single-turn mode	25-bit half fir tree	64 ... 8192 in powers of 2	

Note

The number of encoder pulses is the product of the "Increments per encoder revolution" and the "Number of revolutions" (see "Resolution (Page 9398)").

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
32.0	INC_REV	DINT	L#500	**Increments per encoder revolution** Valid value range: • 1 to 2^{25} Note: This entry is irrelevant when you use "pulse" units.

Resolution

Resolution

The resolution reflects the accuracy of cam processing. It also determines the maximum possible traversing range.

The resolution (RES) is calculated as shown in the table below:

	Incremental encoder	Absolute encoder / initiator
Input values	• Distance per encoder revolution • Increments per encoder revolution: – Pulse evaluation: quadruple – 1 increment = 4 pulses	• Distance per encoder revolution • Increments per encoder revolution. – 1 increment = 1 pulse
Calculation	$RES = \frac{\frac{\text{Distance}}{\text{Encoder revolution}}}{\frac{\text{Pulses}}{\text{Encoder revolution}}}$	

Note

Resolution in the "pulse" system of units is always 1.

All position values are rounded to the integer multiple of the resolution. This allows you to distinguish between the set and used values.

Range of values of the resolution

Convert the range of values for the resolution according to the defined physical units. The resolution must be kept within this range when setting the "distance per encoder revolution" and "increments per encoder revolution" values.

The chosen system of units determines the resolution value range:

Measuring system	Representation in...	Range of values of the resolution
mm	10^{-3} mm	$0,1 \times 10^{-3}$ mm to 1000×10^{-3} mm/pulse
inch	10^{-4} inch	$0,1 \times 10^{-4}$ inches to 1000×10^{-4} inches/pulse
degrees	10^{-4} degrees 10^{-3} degrees 10^{-2} degrees	$0,1 \times 10^{-4}$ degrees to 1000×10^{-4} degrees/pulse $0,1 \times 10^{-3}$ degrees to 1000×10^{-3} degrees/pulse $0,1 \times 10^{-2}$ degrees to 1000×10^{-2} degrees/pulse
Pulses	1 pulse	1

Example

- An incremental encoder has the following data:
  - Increments per encoder revolution: 5000
  - Distance per encoder revolution: 1000 mm
  - 1 increment = 4 pulses

  Resultant resolution (quadruple evaluation):

$$\text{Resolution} = \frac{1000\ \text{mm}}{5000\ \text{increments}} = 0.2000\ \frac{\text{mm}}{\text{Increment}} = 0.2000\ \frac{\text{mm}}{4\ \text{pulses}}$$

$$= 0.0500\ \frac{\text{mm}}{\text{Pulse}}$$

- An SSI encoder has the following data:
  - Increments per encoder revolution: 4096
  - Distance per encoder revolution: 1000 mm
  - 1 increment = 1 pulse

  Resultant resolution:

$$\text{Resolution} = \frac{1000\ \text{mm}}{4096\ \text{increments}} = 0.2441\ \frac{\text{mm}}{\text{Increment}} = 0.2441\ \frac{\text{mm}}{\text{Pulse}}$$

Traversing range dependency on resolution

The traversing range is restricted by the numerical representation in FM x52. This numerical representation varies depending on the resolution. You should thus make sure that your specifications are always within the permitted limits.

The table below sets out the maximum traversing range:

Resolution (RES) is in the range	Maximum traversing range
0.1 µm/pulse ≦ RES < 1 µm/pulse	-10^8 µm to 10^8 µm (-100 m to +100 m)
1 µm/pulse ≦ RES ≦ 1000 µm/pulse	-10^9 µm to 10^9 µm (-1000 m to +1000 m)

Dependency between the velocity and resolution

The velocity indicated may vary within the following limits, depending on the resolution (data relate to mm units):

- from 1 µm/min to 90 m/min, at a resolution < 1 µm/pulse
- from 1 µm/min to 900 m/min, at a resolution ≥ 1 µm/pulse

The velocity value is calculated and filtered by the module at intervals of 4 ms.

Its minimum inaccuracy of one pulse/4 ms rules it out for closed-loop control.

Selecting the count direction

Count direction

You can adapt the position feedback detection to the direction of axis motion using the "Count direction" machine data. You must also take into account the rotational direction of all transmission elements (for example, couplings and gears).

- Normal = incremental count pulses (incremental encoder) or encoder values (absolute encoder) represent increasing positional actual values
- Inverted = incremental count pulses (incremental encoder) or encoder values (absolute encoder) represent decreasing positional actual values

Note:
You cannot implement a rate time in combination with an absolute encoder (SSI) and inverted parameterized count direction.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
59.0	CNT_DIR	BOOL	FALSE	**Count direction** Valid values: • 0 = normal • 1 = inverted

Selecting monitoring

Monitoring

Wire break

When its monitoring function is enabled, the FM x52 monitors the A, /A, B, /B, N, and /N signals of an incremental encoder. The monitoring function detects:

- Wire break
- Short circuits in individual cables
  For incremental encoders without a zero mark, you must either:
  - Disable the wire-break monitoring function
  - Interconnect the N and /N signals externally (Page 9348)
- Edge-to-edge distance of the count pulses
- Encoder supply failure

Frame error

You cannot switch off the monitoring of frame errors with absolute encoders (SSI). It monitors:

- the frame for start and stop bit errors
- the monoflop time of the connected encoder

Missing pulses (incremental encoder)

An incremental encoder must always return the same number of increments between two successive zero marks.

FM x52 checks whether the zero mark of an incremental encoder coincides with the correct encoder value.

Disable missing pulse monitoring for encoders without zero marks.

Also disable wire-break monitoring, or interconnect the zero mark inputs N and /N externally.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
				Monitoring functions
63.0	MON_WIRE	BOOL	TRUE	1 = wire break
63.1	MON_FRAME	BOOL	TRUE	1 = frame error (must always be 1)
63.2	MON_PULSE	BOOL	TRUE	1 = missing pulses

Selecting data for absolute encoder (SSI)

Number of encoder revolutions

Number of encoder revolutions

The machine data "Number of encoder revolutions" only applies to absolute encoders. You use it to define the maximum possible number of revolutions for this encoder. For additional information on this topic, please refer to "Absolute encoders (SSI) (Page 9352)".

- Single-turn encoder
  Only the value 1 is possible.
- Multiturn encoder
  - Multiturn encoder / listen-in (fir tree): 2 ... 4096 in powers of 2
  - Multiturn encoder / listen-in (right-justified): 1 ... 4096 all values
- Linear scale
  You can also connect a linear scale. To do so, enter the value "1".

The total number of steps of the encoder is not machine data. It is calculated as follows:

Overall number of steps = increments per encoder revolution x number of revolutions

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
36.0	NO_REV	DINT	L#1024	**Number of encoder revolutions** Valid value range: • 1 (single-turn encoder) • 1 to 4096 (multiturn encoder)

Transmission rate

Transmission rate

You define the data transmission rate between the SSI encoder and the FM x52 with the "transmission rate" machine data.

The maximum transmission rate depends on the cable length:

- 320 m → 125 kHz
- 160 m → 250 kHz
- 63 m → 500 kHz
- 20 m → 1,000 kHz

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
40.0	BAUDRATE	DINT	L#0	**Transmission rate** Valid values: • 0 = 125 kHz • 1 = 250 kHz • 2 = 500 kHz • 3 = 1000 kHz

Frame length

Frame length

You use "Frame length" to define the clock frame output by the FM x52 for absolute encoders.

Selecting "listen-in" will disable the FM x52 cycle. The FM x52 can then listen in on other SSI frames with a 13 or 25-bit frame. The transmission rate is determined by the cycle clock rate of the master module.

Data used in the parameter DB

The frame length selected here is stored in the parameter DB under the ENC_TYPE (Page 9396) date in the parameter DB.

Initiator direction

Initiator direction

You define the direction of a connected initiator with "Initiator direction":

- Positive: = up (incremental count pulses)
- Negative: = down (downwards count pulses)

Tracks

Selecting track output control mode

Track output control

The "Track output control" machine data defines the control mode for track outputs 0 to 12. Tracks can be activated by:

- The cam controller: Track signals are activated and deactivated by FM 352 cam processing.
- CPU: The track signals represent the corresponding values of the track enables in the channel DB.

This allows the explicit activation of track outputs in the user program.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
90.0	TRACK_OUT	WORD	W#16#0	**Activation of track outputs** Valid values: • 0 = cam controller • 1 = CPU Bit number = track number Bits 13 to 15 must be 0.

Selecting special tracks

Special tracks

You can parameterize tracks 0, 1 and 2 for use as special tracks.

- Track 0 and track 1 as counter cam track (Page 9343)
- Track 2 as brake cam track (Page 9343)

Data used in the parameter DB

Address	Name	Type	Start value	Comment
				Special tracks
99.0	SPEC_TRC0	BOOL	FALSE	1 = track 0 is counter cam track
99.1	SPEC_TRC1	BOOL	FALSE	1 = track 1 is counter cam track
99.2	SPEC_TRC2	BOOL	FALSE	1 = track 2 is brake cam track

Specifying high count value for counter cam track

High count value for counter cam track

With this machine data, you define the high count value of the assigned counter cam track.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
100.0	CNT_LIM0	DINT	L#2	**High count value of counter cam track** (track 0)
104.0	CNT_LIM1	DINT	L#2	**High count value of counter cam track** (track 1) Valid value range: • 2 … 65535

Selecting enable input

Enable input

The track signal Q3 is activated when all the following conditions have been met:

- The track is enabled.
- The relevant external enable input I3 is set.
- The track result is 1.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
95.0	EN_IN_I3	BOOL	FALSE	Enable input 1 = track signal track 3 AND enable input I3 Bits 95.1 to 95.7 must be 0.

Cams

Number of cams

Number of cams

The number of cams determines the maximum number of programmable cams and therefore the cam cycle time.

Number of cams	Cam cycle time
16 cams	20.48 µs
32 cams	40.96 µs
64 cams	81.92 µs
128 cams	163.84 µs

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
76.0	C_QTY	DINT	UDT3: L#0 UDT4: L#1 UDT5: L#2 UDT6: L#3	Number of cams 0 = max. 16 cams 1 = max. 32 cams 2 = max. 64 cams 3 = max. 128 cams

Cam data

Cam data

Cam data

Cam data describe the properties of a cam, the assignment of each cam to a track, and the switching characteristic of the cam.

The cam data below is set separately for each cam:

- Validity (Page 9407)
- Track number (Page 9407)
- Cam type (Page 9408)
- Cam start and cam end of position-based cams (Page 9408)
- Cam start and activation time of time-based cams (Page 9409)
- Lead time (Page 9410)
- Effective direction (Page 9412)
- Hardware interrupt (Page 9414)

The following applies to cam processing in general:

- The module interprets and processes only the cams with "valid" settings.
- Cams 0 to 7 support hardware interrupts.
- The number of assignable cams depends on the number of cams available.

Cam validity

Cam "valid"

All cams to be processed by the cam controller must be set as "valid".

Note

Unused cams should always be set to "invalid" CAMVALID = FALSE).

Data used in the parameter DB

Address relative	Name	Type	Initial value	Comment
+0.0	CAMVALID	BOOL	FALSE	1 = cam "valid"

Specifying a cam track number

Track number

You define the track controlled by the cam by setting the track number.

Data used in the parameter DB

Address Relative	Name	Type	Initial value	Comment
+1.0	TRACK_NO	BYTE	B#16#0	**Track number** Valid value range: • 0 to 31

Selecting cam type

Cam type

Select the required cam type.

- Position-based cam
- Time-based cam

Data used in the parameter DB

Address Relative	Name	Type	Initial value	Comment
+0.3	CAM_TYPE	BOOL	FALSE	**Cam type** Valid values: • 0 = position-based cam • 1 = time-based cam

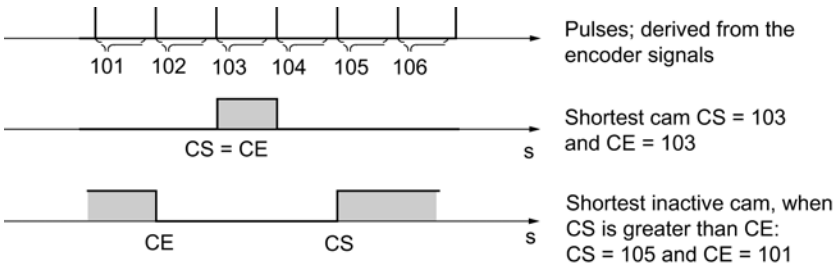
Defining cam start and cam end for position-based cams

Cam start (CS) / Cam end (CE) of position-based cams

The cam start and cam end of a position-based cam define the cam length.

Minimum length of a position-based cam

The figure below shows the shortest cam with axis motion in positive direction



The following general rules apply:

- The inactive part of a cam must always have a minimum interval of 4 pulses between the cam end (CE) and cam start (CS).
- If CE = CS, the cam is activated for the duration of one pulse.

Data used in the parameter DB

Address Relative	Name	Type	Initial value	Comment Of position-based cams
+2.0 +6.0	CBEGIN CEND	DINT DINT	L#-100000000 L#100000000	Cam start (CS) Cam end (CE) Valid value range: • -1,000,000,000 µm to 1,000,000,000 µm

Specifying cam start and activation time for time-based cams

Cam start (CS) / Cam end (CE) ≙ activation time of time-based cams

With a time-based cam, you must specify a cam start and an activation time in place of the cam end. Define the activation time with a resolution of 100 µs. The time runs starting with the activation of the cam.

Conditions of setting default times:

- 0 µs: A cam with an activation time of 0 µs will never be activated
- 0 µs < t ≤ 400 µs: The FM x52 sets a minimum cam activation time of approx. 330 µs.
- t > 400 µs: FM x52 calculates the actual activation time t_{act} based on the default activation time t_{def} using this formula:

$$t_{act} = \text{integer} \left(\frac{t_{def}}{\text{Cam cycle time}} \right) * \text{Cam cycle time}$$

The maximum error is always less than the cam cycle time.

Data used in the parameter DB

Address Relative	Name	Type	Initial value	Comment of time cams
+2.0 +6.0	CBEGIN CEND	DINT DINT	L#-100000000 L#100000000	**Cam start (CS)** **Cam end (CE) ≙ activation time** Valid cam start value range: • -1,000,000,000 µm to 1,000,000,000 µm Valid cam activation time value range: • (0 to 13421) x 100 µs with up to 16 cams • (0 to 26843) x 100 µs with up to 32 cams • (0 to 53686) x 100 µs with up to 64 cams • (0 to 65535) x 100 µs with up to 128 cams

Specifying lead time

Lead time

You can compensate any delays caused by the connected switchgear by setting a lead time. Define a lead time with a resolution of 100 µs. You can assign one lead time to each cam. The lead time applies to the cam start and cam end.

Actuation distance

The actuation distance of a cam is calculated continuously based on the current velocity and lead time. The entire cam is shifted in direction of the actual value by this value. The assigned range is the "static range", and the range calculated based on the lead time represents the "dynamic range".

Actuation distance = lead time x current velocity

FM x52 calculates the actuation distance of all cams within ¼ of the longest parameterized lead time. A very high lead time setting for a cam reduces the frequency of the dynamic adjustment calculation.

Actual lead time

To calculate the actual lead time:

1. Determine the cam cycle time: This is the time in which the FM x52 fully processes all cams. It depends on the number of cams parameterized.
2. Calculate the actual lead time based on the following formula:

$$\text{Lead time}_{act} = \text{integer}\left(\frac{\text{Lead time}_{d}}{\text{Cam cycle time} * 4}\right) * \text{Cam cycle time} * 4$$

   The identifiers have the following meanings:
   - Lead time $_{act}$ is the lead time set by the FM x52
   - Lead time $_{d}$ is your default setting.
   - Integer ( ) means that the values after the decimal point are ignored in the calculation of the parenthesis.

The maximum error of the lead time$_{act}$ is always less than [cam cycle time] x 4.

Example:

The following values are defined:

- Number of cams: Maximum 32 cams
- Cam cycle time: 40.96 µs
- Lead time$_{d}$ = 1000 µs

Result: An actual lead time of 983 µs.

Note

The actual is always less than the assigned lead time. It may be 0 even though the parameterized lead time ≥100 µs.

The actuation distance of a rotary axis must be less than the rotary axis range and the inactive part of the cam. This must be guaranteed for all velocities.

Note

You may not implement a lead time in combination with an absolute encoder (SSI) and inverted parameterized count direction.

Data used in the parameter DB

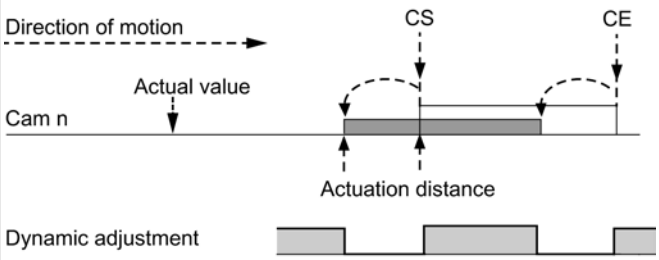
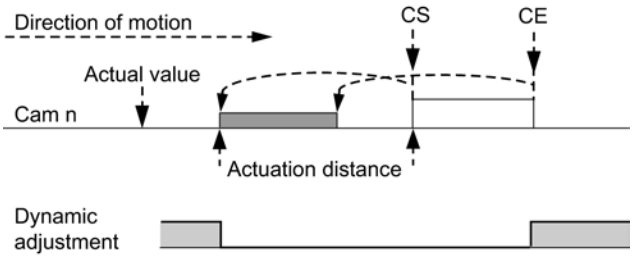
Address Relative	Name	Type	Initial value	Comment
+ 10.0	LTIME	INT	0	**Lead time** Valid value range: • (0 to 53686) x 100 µs with max. 16 cams • (0 to 65535) × 100 µs with max. 32, 64 or 128 cams

Dynamic cam adjustment

There are two distinct situations relating to the cam range:

1. The static and dynamic range of the cam overlap.
2. The static and dynamic range of the cam do not overlap.

The table below shows the calculation of dynamic adjustment in each of the two cases.

Dynamic adjustment	Description
	If the dynamic range of the cam overlaps its static range: • The cam is activated when its dynamic range is reached. At the same time, calculation of a new dynamic adjustment is disabled. • When the actual value reaches the static range of the cam, the calculation of a new dynamic adjustment is enabled again. A change in velocity affects the cam end. • When the cam is deactivated at the end of the dynamic range, dynamic adjustment will be disabled again until the end of the static range of the cam.
	The following applies if the cam's dynamic and static ranges do not overlap: • The cam is activated when its dynamic range is reached. At the same time the calculation of a new dynamic adjustment is disabled. • At the end of the static range of the cam, dynamic adjustment is enabled again.
Dynamic range Static range A **new** dynamic adjustment is possible	

Note

When the direction of rotation changes, calculation of the dynamic adjustment is enabled again.

Selecting effective direction

Effective direction

Two effective directions are supported:

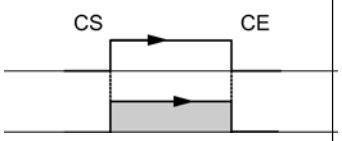
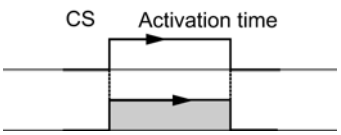
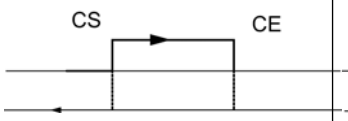
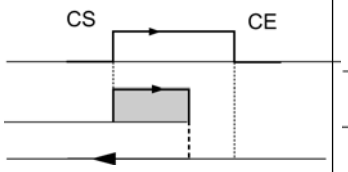
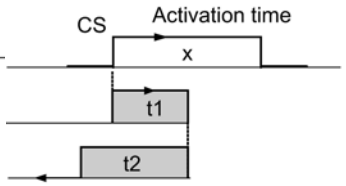
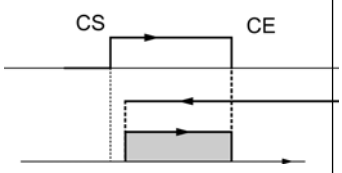
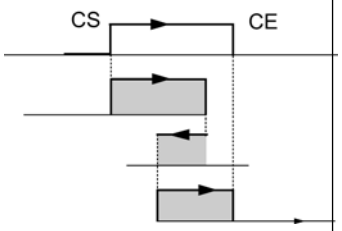
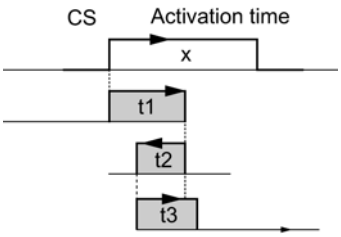
- Positive: The cam is activated at the cam start if the axis is moving in the direction of increasing process values.
- Negative: The cam is activated at the cam end if the axis is moving in the direction of decreasing process values.

You may set both effective directions in parallel.

Selecting "None" disables the cam.

Switching characteristics of cams based on their effective direction

The table below shows the switching characteristics of position-based and time-based cams as determined by the effective direction. A positive effective direction is always assumed, with the exception of example 5.

No.	Description	Position-based cams	Time-based cams
1	A cam is overrun in the effective direction		
2	A cam is traversed in reversed effective direction		
3	A cam is approached in the effective direction; the direction of motion of the axis is reversed while the cam is active		 Activation time x = t1 + t2
4	A cam is approached in reversed effective direction; the axis reverses its direction of movement on the cam in effective direction		Cam not switched
5	A cam is approached from any direction, and exited into any direction; **both directions** are set as effective direction		 Activation time x = t1 + t2 + t3
parameterized cam switched cam			

Data used in the parameter DB

Address Relative	Name	Type	Initial value	Comment
+0.1	EFFDIR_P	BOOL	TRUE	1 = positive effective direction (plus)
+0.2	EFFDIR_M	BOOL	TRUE	1 = negative effective direction (minus)

Selecting hardware interrupt for cam

Hardware interrupt when cam switches on / off

You can specify whether hardware interrupts are to be generated when cams 0 to 7 are enabled, disabled or enabled and disabled.

Data used in the parameter DB

Address Relative	Name	Type	Initial value	Comment
+0.4	PI_SW_ON	BOOL	FALSE	1 = hardware interrupt on activation
+0.5	PI_SW_OFF	BOOL	FALSE	1 = hardware interrupt on deactivation

Enable hardware interrupts

Enabling hardware interrupts

Enabling hardware interrupts

This machine data is used to enable hardware interrupts.

You can set when hardware interrupts are to be generated:

- Cam on/off
  In the cam data (Page 9414), you can specify whether hardware interrupts are to be generated when cams 0 to 7 are activated and/or deactivated.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
3.2	PI_CAM	BOOL	FALSE	1 = enable hardware interrupt: Cam on/off

Evaluation of a hardware interrupt

If FM 352 triggers a hardware interrupt, interrupt information is available in the OB4x_POINT_ADDR tags of the hardware interrupt OB.

The following table shows the content of the double word OB4x_POINT_ADDR.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	Cam	0	0
2	Cam 7 on	Cam 7 off	Cam 6 on	Cam 6 off	Cam 5 on	Cam 5 off	Cam 4 on	Cam 4 off
3	Cam 3 on	Cam 3 off	Cam 2 on	Cam 2 off	Cam 1 on	Cam 1 off	Cam 0 on	Cam 0 off

You can see the cause of the interrupt in byte 1.

Cam: Evaluate bytes 2 and 3 according to the table.

Lost hardware interrupts

If the hardware interrupt OB is still busy processing a hardware interrupt, the module registers all subsequent hardware interrupt events. If an event occurs again before the hardware interrupt could be triggered, the module triggers a "hardware interrupt lost" diagnostic interrupt.

Control and feedback interface of the FM 352

Application

Very fast access to checkback and control signals is required in special applications or in an alarm level. You can obtain those data directly by reading the IO areas of the module.

To coordinate startup following each module startup, for example, after plugging a module or after a STOP-RUN transition of the CPU, you must call the CAM_CTRL (Page 7157) instruction until the end of the startup is indicated by RETVAL = 0.

Note

For direct access to FM 352 data, always use the non-internal data and method described in this section. Otherwise, your user program will encounter difficulties accessing the module.

Reading checkback signals via direct access

The byte addresses are defined with an offset to the output address of the module. The names of the parameters correspond to the names in the channel DB (Page 7164).

In STL, you access the data using the PIB (read 1 byte) and PID (read 4 bytes) commands.

Address	Bit number							
	7	6	5	4	3	2	1	0
Byte 0	PARA	Internal	Internal	DATA_ERR	Internal	DIAG	Internal	Internal

Address	Bit number							
Byte 1	0	0	0	CAM_ACT	0	0	0	0
Byte 2	Internal							
Byte 3	0	0	FVAL_DONE	HYS	GO_P	GO_M	MSR_DONE	SYNC
Byte 4	ACT_POS							
Byte 5								
Byte 6								
Byte 7								
Byte 8	TRACK_OUT							
Byte 9								
Byte 10								
Byte 11								

Direct write access to control signals

The byte addresses are defined with an offset to the input address of the module. The names of the parameters correspond to the names in the channel DB (Page 7164).

In STL, you access the data using the PQB (write 1 byte) and PQW (write 2 bytes) commands.

Address	Bit number							
	7	6	5	4	3	2	1	0
Byte 0	Internal							
Byte 1	0	CNTC1_EN	CNTC0_EN	CAM_EN	DIR_P	DIR_M	0	0
Byte 2	TRACK_EN							
Byte 3								

Example: Position process value ACT_POS

The start address of the module is 512.

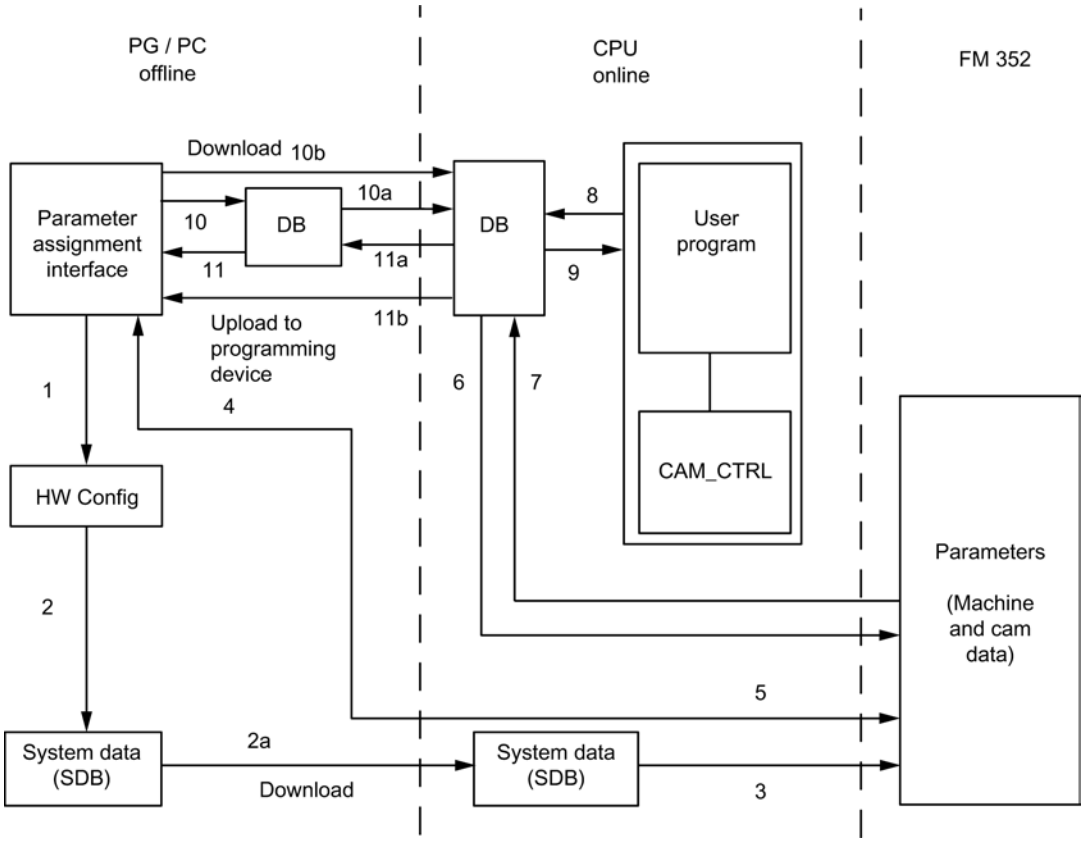
STL

```
L PID 516      //Read the position process value (ACT_POS) with direct access:
               //Start address of the module + 4
```

Parameter transmission paths

The term parameter refers to the following machine and cam data.

The following figure shows the parameter transmission paths of the FM 352.



1 Save the parameters in the parameter assignment interface.

2 Saving and compiling the HW configuration, and download to the CPU.

3 The CPU writes the parameters to the module during system parameter assignment.

4 Upload the module parameters to the PG by selecting the "PLC > Upload to PG" command.

5 Download the parameters from the parameter assignment interface to the module with the "PLC > Download" command.

6 Write parameters to the module using jobs in the user program.

7 Read parameters from the module using jobs in the user program.

8 Save parameters from the user program to the online DB.

9 Read in parameters from the online DB to the user program.

10 Export parameters from the parameter assignment interface to the DB (offline or online DB); an offline DB must then be downloaded to the CPU.

11 Import parameters from an online or offline DB into the parameter assignment interface.

Some use cases for the transfer of parameters

Use case	Steps
You edit the parameters on the parameter assignment interface. The parameter should then be assigned automatically to the module during startup.	Perform steps 1, 2, and 3.
You edit parameters in the parameter configuration interface during commissioning in test mode.	Perform steps 4 and 5
The parameters edited during commissioning should be downloaded automatically during startup.	Perform steps 1, 2, and 3.
Create the parameters using the parameter assignment interface. When it starts up, the module should be assigned parameters only by the user program using data blocks.	Perform steps 10 and 6.
You need a comfortable means of creating data records for recipes.	Perform step 10.
Create the parameters using the parameter assignment interface. These should be available to the user program for temporary modifications.	Perform steps 1, 2, and 3 for automatic parameter assignment.
	Perform steps 10, 7 to enable access by the user program.
You modify existing parameters (exclusively) using the user program.	Perform steps 7, 9, 8, and 6.
You want to view the data modified by the user program on the parameter assignment interface.	Perform step 11.
Parameters modified by the user program should also be loaded automatically during startup.	Perform steps 6, 11, 1, 2, and 3.

Commissioning the FM 352

Using FM 352 from the user program

Integrating the FM 352 into a user program

Instructions are available for integrating the FM 352 into a user program. These enable you to handle the desired functions with ease.

The instructions CAM_INIT and CAM_CTRL can be called in the CPU user program. They ensure communication between the CPU and the FM 352. The channel DB (Page 7164) is created automatically as the instance DB for the CAM_CTRL instruction when you insert the CAM_CTRL instruction into the project.

All the parameter data associated with the channel of the module (machine data, cam data) is located in the parameter DB (Page 7171) for the module. Use the command "Program blocks > Add new block" to add a CAM_PxxxTYPE type data block to your project. Depending on the quantity structure, you need the following DB types:

- Up to 16 cams: CAM_P016TYPE
- Up to 32 cams: CAM_P032TYPE
- Up to 64 cams: CAM_P064TYPE
- Up to 128 cams: CAM_P128TYPE

The instruction CAM_DIAG is also available for FM 352. This instruction allows you to transfer the diagnostic information to the diagnostic DB. The diagnostic DB (Page 7174) is created automatically as the instance DB for the CAM_DIAG instruction when you insert the CAM_DIAG instruction into the project.

Commissioning the module

The module can be initially tested using the "Commissioning" dialog, even when no user program is available. Note the following warning in particular in this case.

WARNING

Injury and material damage could occur.

To prevent personal injury and material damage, observe the following points:

- Install an emergency stop switch in the vicinity of the computer.
  This is the only way you can ensure that the plant can be safely shut down in the event of computer or software failure.
- Install hardware limit switches, which act directly on the power units of all drives.
- Make sure that no one has access to the area of the plant in which moving parts are present.
- When the FM 352 is controlled and monitored simultaneously by your program and the "Commissioning" dialog, conflicts can occur, the effects of which are not clear. Therefore, always set the CPU to STOP mode or disable your program when you work with the "Commissioning" dialog.

See also

CAM_INIT (Page 7156)

CAM_CTRL (Page 7157)

CAM_DIAG (Page 7158)

FM 352: Online & Diagnostics

Current module information

Default diagnostics

Error display through group error LED of FM 352

Where is the error displayed?

If the red group error LED is lit, an error occurred either at the module (internal error) or at the cable ports (external error). The LED is lit, even with errors that occur briefly, for at least 3 seconds.

Which errors are displayed?

The table below shows the error types indicated by the group error LED.

LEDs	Meaning	Notes
SF	Group error for internal and external errors	This LED indicates the following error states of FM 352: • Hardware interrupt lost • Watchdog expired • FM 352 not configured • FM 352 incorrectly configured • No external 24 V auxiliary supply • Front connector missing • Encoder wire break • Process error • Absolute encoder frame error • Incremental encoder pulse missing or zero mark signal missing To remove the error, see Error classes (Page 9425).

Channel-specific diagnostics

Triggering diagnostic interrupts

Interrupt processing

The FM 352 can trigger hardware and diagnostic interrupts. You process these interrupts in an interrupt OB. If an interrupt is triggered without the corresponding OB being loaded, the CPU switches to STOP.

You enable the processing of diagnostic interrupts as follows:

1. Select the module in HWCN
2. Select "Edit" > Object Properties > Basic Parameters" to enable the diagnostic interrupt.
3. Save and compile the configuration.
4. Download the configuration to the CPU.

Overview of diagnostic interrupts

Events and errors which trigger a diagnostic interrupt:

- Operating error
- Machine data error
- Incorrect cam data
- Diagnostic errors

These errors are explained in detail in "Error classes (Page 9425)".

Module response to an error with diagnostic interrupt

- Cam processing will be disabled.
- The synchronization will be cleared by the following diagnostic interrupts:
  - Front connector missing, no external auxiliary voltage
  - Zero mark error detected, cable fault (5 V encoder signals)
  - Traversing range exceeded (indicated by an operating error)
  - Set process value cannot be executed (indicated by an operating error).
- With one exception, control signals are no longer processed.
  Exception:
  When the software limit switches are overrun, a change of direction is still possible in simulation mode.
- Function switch and job processing continues.

Module detects an error ("incoming")

A diagnostic interrupt is an "incoming" event if at least one error is pending. If any errors are pending clearance, the queued errors are reported again as "incoming" events.

Sequence:

1. The FM 352 detects one or more errors and triggers a diagnostic interrupt. The error LED lights up. The error event is logged to the diagnostic buffer.
2. The CPU operating system calls the diagnostic interrupt OB, e.g. OB 82.
3. You can now evaluate the start information of OB82.
4. The OB82_MOD_ADDR parameter indicates which module triggered the interrupt.
5. For additional information, call the CAM_DIAG (Page 7158) instruction.

The module detects the transition to an error-free state ("outgoing")

A diagnostic interrupt is only registered "outgoing" if all errors on the module are cleared.

Sequence:

1. The FM 352 detects that all errors have been eliminated and triggers a diagnostic interrupt. The error LED goes out. The diagnostic buffer remains unchanged.
2. The CPU operating system calls the diagnostic interrupt OB, e.g. OB 82.
3. The OB82_MOD_ADDR parameter indicates which module triggered the interrupt.
4. Evaluate the OB82_MDL_DEFECT bit.
   When this bit is "0", no errors are present on the module. You can close the evaluation session at this point.

Diagnostic interrupt control by CPU states

- Diagnostic interrupts are blocked by FM 352 when the CPU is in STOP mode.
- If not all of the queued errors are eliminated while the CPU is in STOP mode, the FM 352 will report the remaining errors as "incoming" events again at the next CPU transition to RUN.
- If all queued errors are eliminated when the CPU is in STOP mode, the error-free state of the FM 352 will **not** be reported by a diagnostic interrupt after the transition to RUN mode.

Error evaluation in the user program

Fault reaction in the user program

The user program allows a targeted response to errors. The following options are available:

- Return values (RETVAL) of the integrated instructions as a group display for errors that occurred during execution of the instructions.
- Error bits _ERR of the jobs as a group display for errors that occurred while a job was being processed.
- The DATA_ERR error bit as a group display for an error that the FM 352 has detected during a write job.
- The error identifier in JOB_ERR for the cause of an error in communication between instruction and FM 352.
- The CAM_DIAG (Page 7158) instruction for reading the diagnostic buffer of the FM 352. This is where you find out the causes of errors in jobs and asynchronous events (operator errors, diagnostic errors).
- Diagnostic interrupts for fast reaction to events.

Diagnostics buffer of the module

Diagnostic events

The module´s diagnostic buffer contains a maximum of 4 diagnostic events and is organized as a ring buffer.

A diagnostic event is written in the buffer if an "incoming" (error) message is detected. This can be an alarm, a synchronous error (data error) or even an asynchronous error (operational errors and diagnostics errors). From a cause of error several entries can occur as follow-up errors. Outgoing messages do not create any entries in the diagnostic buffer.

For each diagnostic event the following is given:

- Status (always incoming)
- Internal error
- External error
- Error class
- Error number
- Channel number
- Cam number (for cam data errors)

When a diagnostic event is written in the diagnostic buffer, the checkback signal DIAG = 1 is set.

Using the CAM_DIAG (Page 7158) instruction, the diagnostic buffer can be transferred as a whole to a data block (diagnostic DB) or displayed by the error evaluation dialog of the configuration software. If the diagnostic buffer is read, the module sets the checkback signal DIAG = 0.

Note

If the diagnostic buffer is read simultaneously from the CAM_DIAG instruction and the error evaluation dialog, it is possible that a new diagnostic event not be detected by the program.

Data and Structure of the Diagnostic DB

Data and Structure of the Diagnostic DB

Note

Do not modify any data not listed in this table.

The following table shows the structure of the diagnostic DB:

Address	Name	Type	Initial value	Comment
0.0	MOD_ADDR (Enter!)	INT	0	Module address
256.0	JOB_ERR	INT	0	Communication errors
258.0	JOBBUSY	BOOL	FALSE	1 = job active
258.1	DIAGRD_EN	BOOL	FALSE	1 = read diagnostic buffer unconditional
260.0	DIAG_CNT	INT	0	Number of valid entries in the list
262.0	DIAG[1]	STRUCT		Diagnostic data latest entry
272.0	DIAG[2]	STRUCT		Diagnostic data second entry
282.0	DIAG[3]	STRUCT		Diagnostic data third entry
292.0	DIAG[4]	STRUCT		Diagnostic data oldest entry

The diagram below shows the structure of a DIAG[n] diagnostic entry.

Address	Name	Type	Initial value	Comment
+0.0	STATE	BOOL	FALSE	0 = outgoing event 1 = incoming event
+0.1	INTF	BOOL	FALSE	1 = Internal error
+0.2	EXTF	BOOL	FALSE	1 = external error
+2.0	FCL	INT	0	Error class: 1: Operating error 4: Data error 5: Machine data error 7: Cam data error 15: Messages 128: Diagnostic errors
+4.0	FNO	INT	0	Error number 0 to 255
+6.0	CH_NO	INT	0	Channel number (always 1)
+8.0	CAMNO	INT	0	Cam numbers 0 to 127 with error class = cam data error

List of JOB_ERR messages

JOB_ERR alarms

JOB_ERR (Hex)	JOB_ERR (Dec)	JOB_ERR (Int)	Meaning
80A0	32928	-32608	Negative acknowledgment when reading from module. Module removed during read operation or module defective.
80A1	32929	-32607	Negative acknowledgment when writing to module. Module removed during write operation or module defective.
80A2 [1)]	32930	-32606	Protocol error on Layer 2 (data transfer interrupted within PROFINET / PROFIBUS DP, for example, due to wire break, missing connector, parameter assignment error, etc.)
80A3 [1)]	32931	-32605	Protocol error at user interface/user (data transfer interrupted within PROFINET / PROFIBUS DP, for example, due to wire break, missing connector, parameter assignment error, etc.)

JOB_ERR (Hex)	JOB_ERR (Dec)	JOB_ERR (Int)	Meaning
80A4 [1]	32932	-32604	Communication problem on K bus
80B1	32945	-32591	Specified length wrong. FM_TYPE parameter in the channel DB is not correctly set for the module being used.
80B2	32946	-32590	The configured slot is empty.
80B3	32947	-32589	Actual module type does not match configured module type.
80C0 [1]	32960	-32576	The module has not yet prepared the data to be read.
80C1 [1]	32961	-32575	The data of a write job of the same type have not yet been processed on the module.
80C2 [1]	32962	-32574	The module is currently processing the maximum number of jobs.
80C3 [1]	32963	-32573	Required resources (memory etc.) currently in use.
80C4 [1]	32964	-32572	Communication errors
80C5 [1]	32965	-32571	Distributed I/Os not available.
80C6 [1]	32966	-32570	Priority class abort (restart or background)
8522	34082	-31454	Channel DB or parameter DB too short. The data cannot be read from the DB. (write job)
8532	34098	-31438	DB number of the parameter DB too high (write job)
853A	34106	-31430	Parameter DB does not exist. (write job)
8544	34116	-31420	Error in nth (n > 1) read access to a DB after error occurred. (write job)
8723	34595	-30941	Channel DB or parameter DB too short. The data cannot be written to the DB. (read job)
8730	34608	-30928	Parameter DB on the CPU write-protected. The data cannot be written to the DB (read job).
8732	34610	-30926	DB number of the parameter DB too high (read job)
873A	34618	-30918	Parameter DB does not exist. (read job)
8745	34629	-30907	Error in nth (n > 1) write access to a DB after error occurred. (read job)

[1] Errors 80A2 to 80A4 and 80Cx are temporary, i.e., they can be rectified automatically after a waiting time. Messages in the form 7xxx indicate temporary states in communication.

Error classes

Operating error

Error class 1: Operating error

Operating errors are detected asynchronous to operator input/commands.

The following table shows the possible operating errors and their respective causes:

No.	Meaning	Diagnostic interrupt
1	Software limit switch start overrun	Yes
2	Software limit switch end overrun	Yes
3	Traversing range start passed	Yes

No.	Meaning		Diagnostic interrupt
4	**Traversing range end passed**		Yes
13	**Set actual value on-the-fly cannot be executed**		Yes
	Cause	The software limit switches are outside the traversing range after the actual value on the fly is set (-100 m to +100 m or -1000 m to +1000 m). The shift resulting from set actual value / set actual value on-the-fly is more than ±100 m or ± 1000 m.	
	Effect	Axis not synchronized.	

Data error

Error class 4: Data error

Data errors are detected synchronous to operator input/commands.

The following table shows the possible data errors and their respective causes:

No.	Meaning		Diagnostic interrupt
10	**Zero offset error**		No
	Cause	The zero offset is greater than ±100 m or ±1000 m. The software limit switches are outside the traversing range after zero offset (-100 m to +100 m or -1000 m to +1000 m). Rotary axis: The zero offset amount is greater than the end of the rotary axis.	
11	**Faulty actual value setting**		No
	Cause	Linear axis: The coordinate lies outside the current (possibly shifted) software limit switches. Rotary axis: The coordinate is < 0 or greater than the end of rotary axis.	
12	**Reference point error**		No
	Cause	Linear axis: The coordinate lies outside the current (possibly shifted) software limit switches. Rotary axis: The coordinate is < 0 or greater than the end of rotary axis.	
20	**Enable machine data not permitted**		No
	Cause	There are no new (error-free) machine data on the module.	
21	**Set actual value on-the-fly not permitted**		No
	Cause	An attempt was made to execute "set actual value on-the-fly" while "retrigger reference point" was active.	
27	**Illegal bit-coded setting**		No
	Cause	Unused and, in this case, unwritten bits are not 0. An attempt was made to select "length measurement" and "edge detection" at the same time.	
28	**Retrigger reference point is not permitted**		No

No.	Meaning		Diagnostic interrupt
	Cause	An attempt was made to execute "retrigger reference point" while "set actual value on-the-fly" was active. An attempt was made to execute "retrigger reference point" with an SSI encoder.	
29	Illegal bit-coded command		No
	Cause	Unused and, in this case, unwritten bits are not 0.	
30	Incorrect lead time		No
31	Incorrect cam number		No
	Cause	Cam is invalid. The cam number is not in the range 0 to 127.	
32	Incorrect cam start		No
	Cause	The cam start is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m). Rotary axis: The cam start is < 0 or greater than the end of the rotary axis.	
33	Incorrect cam end / incorrect activation time		No
	Cause	The cam end is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m). Rotary axis: The cam end is < 0 or greater than the end of the rotary axis. The cam is not activated at least for the duration of one pulse. With an inverse cam, there are not at least 4 pulses between the cam start and cam end.	
34	Cancel set actual value not possible		No
	Cause	The actual position value would be outside the working range with an SSI encoder and linear axis after making the setting.	
35	Incorrect actual value specified by "set actual value" / "set actual value on-the-fly"		No
	Cause	The specified actual value is outside the permitted numeric range of ±100 m or ±1000 m. The software limit switches would be outside the traversing range (-100m to +100 m or -1000 m to +1000 m) were this setting to be applied. The offset derived from set actual value / set actual value on-the-fly would be more than ±100 m or ±1000 m.	
107	Parameters not set for the axis		No
	Cause	There are no machine data on the axis. There are no machine data activated on the axis.	
108	Axis not synchronized		No
	Cause	One of the settings "set actual value" or "set actual value on-the-fly" was initiated although the axis is not synchronized.	
109	Cam processing running.		No
110	Incorrect number of cams to be modified		No

Machine data error

Error class 5: Machine data error

The diagnostic interrupt is only triggered if an error is detected in the system data block (SDB).

The following table shows the possible machine data errors and their respective causes:

No.	Meaning		Diagnostic interrupt
5	**Error in hardware interrupt setting**		Yes
	Cause	You have attempted to select a hardware interrupt that the module does not support.	
6	**Incorrect minimum edge distance**		Yes
	Cause	You have entered a value < 0 or > 10^9 µm as the minimum edge-to-edge distance.	
8	**Incorrect axis type**		Yes
	Cause	You have specified neither 0 nor 1 as the axis type	
9	**Incorrect end of rotary axis**		Yes
	Cause	The value for the end of the rotary axis is outside the valid range from 1 to 10^9 µm or 1 to 10^8 µm (depending on the resolution).	
10	**Incorrect encoder type**		Yes
	Cause	The value for the encoder type is outside the valid range of 1 to 10	
11	**Incorrect distance per encoder revolution**		Yes
	Cause	The value for distance/encoder revolution is outside the permitted range of 1 to 10^9 µm (regardless of the resolution).	
13	**Incorrect number of increments per encoder revolution** (see "Increments per encoder revolution (Page 9397)")		Yes
14	**Incorrect number of revolutions** (see "Number of encoder revolutions (Page 9402)")		Yes
15	**Incorrect transmission rate**		Yes
	Cause	You have specified a transmission rate outside the permitted range of 0 to 3.	
16	**Incorrect reference point coordinate**		Yes
	Cause	The coordinate is outside the range of -100 m to +100 m or -1000 m to +1000 m, depending on the resolution. Linear axis: The coordinate is outside the working range. Rotary axis: The coordinate is greater than the end of the rotary axis or < 0.	
17	**Incorrect absolute value adjustment**		Yes
	Cause	SSI encoder: The value of the absolute encoder adjustment is not in the encoder range (increments per encoder revolution x number of revolutions - 1).	
18	**Incorrect "Retrigger reference point" type**		Yes
	Cause	You have specified a value other than 0, 1, 6 and 7.	
19	**Incorrect direction adaptation**		Yes
	Cause	You have specified a value other than 0 and 1.	
20	**Hardware monitoring not possible**		Yes
	Cause	You have set frame error monitoring to "FALSE" in the parameter DB.	
21	**Incorrect software limit switch start**		Yes

No.	Meaning		Diagnostic interrupt
	Cause	Linear axis: The software limit switch start is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m, depending on the resolution). Linear axis: The software limit switch start (including any existing zero offset) is less than -100 m or -1000 m (depending on the resolution).	
22	**Incorrect software limit switch end**		Yes
	Cause	Linear axis: The software limit switch end is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m, depending on the resolution) or is less than the software limit switch start. The software limit switch end (including any existing zero offset) is greater than +100 m or +1000 m (depending on the resolution).	
144	**Incorrect number of cams**		Yes
	Cause	You have specified a value other than 0 to 3 for the number of cams.	
145	**Incorrect hysteresis**		Yes
	Cause	The hysteresis is outside the range 0 to 65535 × resolution. Hysteresis is greater than ¼ × operating range or ¼ × rotary axis range.	
146	**Incorrect simulation velocity**		Yes
	Cause	The simulation speed is outside the range 1000 × RES to 3 × 10^7 × RES, or is greater than 5 × 10^8 µm/min. The simulation velocity cannot be set internally.	
147	**Incorrect track**		Yes
	Cause	Control of a track outside 0 to 15 (bits 0 to 15) was selected.	
148	**Incorrect selection of enable inputs**		Yes
	Cause	You wanted to activate a track outside 3 to 10 (bits 0 to 7) using an external signal.	
149	**Incorrect selection of special tracks**		Yes
	Cause	You wanted to define a track outside 0, 1 and 2 (bits 0, 1 and 2) as a special track.	
150	**Incorrect high count value track 0**		Yes
	Cause	You have specified a count value < 2 or > 65535 as the high count value.	
151	**Incorrect high count value track 1**		Yes
	Cause	You have specified a count value < 2 or > 65535 as the high count value.	
200	**Incorrect resolution**		Yes
	Cause	You have specified a resolution < 0.1 µm/pulse or >1000 µm/pulse. You have specified a distance per encoder revolution and a number of pulses per encoder revolution that results in a resolution of < 0.1 or > 1000.	
201	**Position encoder does not match the working range/rotary axis range**		Yes
	Cause	SSI position encoder and rotary axis: The encoder does not exactly cover the rotary axis range. Linear axis: The position encoder does not cover at least the operating range (including software limit switches).	

Cam data error

Error class 7: Cam data error

The diagnostic interrupt is only triggered if an error is detected in the system data block (SDB).

The following table shows the possible cam data errors and their respective causes.

No.	Meaning		Diagnostic interrupt
1	**Illegal hardware interrupt**		Yes
	Cause	You attempted to define a process interrupt for a cam with a cam number > 7.	
2	**Incorrect track number**		Yes
	Cause	The track number is outside the range from 0 to 31.	
3	**Incorrect cam start**		Yes
	Cause	The cam start is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m). Rotary axis: The cam start is < 0 or greater than the end of the rotary axis.	
4	**Incorrect cam end**		Yes
	Cause	The cam end is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m). The cam is at least one pulse long. Rotary axis: The cam end is < 0 or greater than the end of the rotary axis. There are not at least 4 pulses between the cam start and cam end for an inverse cam.	
5	**Incorrect activation time**		Yes
	Cause	The activation time is < 0 µs. The maximum value (Page 9409) depends on the number of cams.	
6	**Incorrect lead time**[1]		Yes
	Cause	The lead time is < 0 µs. The maximum value (Page 9410) depends on the number of cams.	
50	**Too many cam sets**		Yes
	Cause	You wanted to enter more cam records than is possible with this number of cams.	
51	**Axis in operation**		Yes
	Cause	You wanted to enter cam records while the cam controller is active.	
52	**Parameters not set for the axis**		Yes
	Cause	You want to enter cam data although not machine data are active yet.	

[1] The error message can also be issued if you have assigned the parameter "inverted" as the count direction with an absolute encoder (SSI).

Alarms

Error class 15: Alarms

The following table shows the possible alarms and their respective causes:

No.	Meaning		Diagnostic interrupt
1	**Start of parameter assignment**		No
	Cause	The module has detected parameter assignment via a system data block.	
2	**End of parameter assignment**		No
	Cause	The module has processed the parameter assignment via a system data block error-free.	

Diagnostic errors

Error class 128: Diagnostic errors

The following table shows the possible diagnostic errors and their respective causes, effects and remedies.

No.	Meaning		Diagnostic interrupt
4	**External auxiliary voltage missing**		Yes
	Cause	External auxiliary 24 V voltage is not connected or has failed, front connector missing	
	Effect	See "Triggering diagnostic interrupts (Page 9420)": • Cam processing is disabled • Track outputs are deactivated • With incremental encoders, synchronization is canceled • FM 352 has not been configured (checkback signal PARA = 0).	
	Correction	Make sure that the 24 V connection is correct (if the 24 V connection is correct, then the module is defective)	
51	**Watchdog expired**		Yes
	Cause	• Strong interference on FM 352 • Error in the FM 352	
	Effect	• Module is reset • If, after resetting the module, no module fault is detected, the module is ready for operation again • The module reports the expired watchdog with "incoming" and "outgoing"	
	Correction	• Eliminate the interference • Contact the relevant sales department which will require details of the exact circumstances leading to the error • Replace the FM 352	
52	**Internal module power supply failed**		Yes

No.	Meaning		Diagnostic interrupt
	Cause	Error in the FM 352	
	Effect	• Module is reset • If, after resetting the module, no module fault is detected, the module is ready for operation again	
	Correction	Replace the FM 352	
70	**Hardware interrupt lost**		Yes
	Cause	A hardware interrupt event detected by FM 352 cannot be reported because an identical event has not yet been processed by the user program/ CPU.	
	Effect	• Cam processing is disabled • Track outputs are deactivated • Synchronization is cancelled in incremental encoders	
	Correction	• Integrate OB 40 into the user program • Check bus connection of the module • Deactivate hardware interrupt • Adapt your hardware and software to suit your process requirements (for example, faster CPU, optimize user program)	
144	**Encoder wire break**		Yes
	Cause	• Encoder cable cut or not plugged in • Encoder has no transverse signals • Incorrect pin assignment • Cable length too long • Encoder signals short-circuited	
	Effect	• Cam processing is disabled • Track outputs are deactivated • Synchronization is cancelled in incremental encoders	
	Correction	• Check the encoder cable • Keep within encoder specifications • Monitoring can be temporarily hidden at the responsibility of the operator using the parameter assignment dialog • Keep to the module's technical data	
145	**Absolute encoder frame error**		Yes
	Cause	The frame traffic between FM 352 and the absolute encoder (SSI) is incorrect or has been interrupted: • Encoder cable cut or not plugged in • Incorrect encoder type • Encoder incorrectly set (programmable encoders) • Frame length incorrectly specified • Encoder returns incorrect values (encoder defective) • Interference on measuring system cable • Transmission rate selected too high	
	Effect	• Cam processing is disabled • Track outputs are deactivated • The last correct actual value remains unchanged till the end of the next correct SSI transfer	

No.	Meaning		Diagnostic interrupt
	Correction	• Check the encoder cable • Check the encoder • Check the frame traffic between the encoder and FM 352	
146	**Missing pulses, incremental encoder**		Yes
	Cause	• Encoder monitoring has detected missing pulses • Number of increments per encoder revolution is incorrectly entered • Encoder defective: does not return the defined number of pulses • Bad or missing zero marker • Interference on the encoder cable	
	Effect	• Cam processing is disabled • Track outputs are deactivated • Synchronization is canceled	
	Correction	• Enter the correct number of increments/encoder revolution • Check the encoder and its cable • Comply with shielding and grounding regulations • Monitoring can be temporarily hidden at the responsibility of the operator using the parameter assignment dialog	

11.2.1.4 Using FM 452

Terminal assignment of the front connector of FM 452

Front connector

Connect the following to the 48-pin front connector:

- the encoder
- the digital outputs and inputs
- the supply voltage of the encoders and digital outputs

Terminal assignment of front connector (X1)

The table below shows the terminal assignment of the front connector of FM 452.

Wiring	Name	Proximity switch	Incremental encoder	Absolute encoder
1		–		
2		–		
3	1L+	Auxiliary supply 24 VDC		
4	A / DAT	–	Encoder signal A (5 V)	SSI data
5	/A / /DAT	–	Encoder signal A inverse (5 V)	SSI data inverse

Wiring	Name	Proximity switch	Incremental encoder	Absolute encoder
6	B / CLI [1]	–	Encoder signal B (5 V)	SSI shift clock input [1]
7	/B / /CLI [1]	–	Encoder signal B inverse (5 V)	SSI shift clock input inverse [1]
8	N	–	Zero mark signal (5 V)	–
9	/N	–	Zero mark signal inverse (5 V)	–
10	CLS [2]	–	–	SSI shift clock output
11	/CLS [2]	–	–	SSI shift clock output inverse
12	A*	Encoder signal A (24 V)		–
13	B*	–	Encoder signal B (24 V)	–
14	N*	–	Zero mark signal (24 V)	–
15	Q0	Digital output 0		
16	Q1	Digital output 1		
17	Q2	Digital output 2		
18	Q3	Digital output 3		
19	Q4	Digital output 4		
20	Q5	Digital output 5		
21	Q6	Digital output 6		
22	Q7	Digital output 7		
23	5.2 VDC	–	Encoder supply (5.2 V)	
24	24 VDC	Encoder supply (24 V)		
25	M [3]	Encoder ground		
26	2L+	Auxiliary supply 24 VDC		
27	RE	–	Current sourcing/current sinking[4]	–
28	Q8	Digital output 8		
29	Q9	Digital output 9		
30	Q10	Digital output 10		
31	Q11	Digital output 11		
32	Q12	Digital output 12		
33	Q13	Digital output 13		
34	Q14	Digital output 14		
35	Q15	Digital output 15		
36	3L+	Auxiliary supply 24 VDC		
37	I0	Brake cam track enable		
38	I1	Length measurement/ edge detection/ set actual value on-the-fly		
39	I2	Reference point switch		
40	I3	Enable track signal 3		
41	I4	Enable track signal 4		
42	I5	Enable track signal 5		
43	I6	Enable track signal 6		
44	I7	Enable track signal 7		
45	I8	Enable track signal 8		

Wiring	Name	Proximity switch	Incremental encoder	Absolute encoder
46	I9	Enable track signal 9		
47	I10	Enable track signal 10		
48	M [3]	Ground for auxiliary voltages		

[1] In listen-in mode

[2] In master mode

[3] Ground connections wired on module

[4] **Note:** If you are connecting an incremental encoder from a third party in series-mode switching (current sourcing/current sinking), the following must be observed:

- Sourcing output: Connect RE (27) with ground (25).
- Sinking output: Connect RE (27) with +24 V (24).

Additional information about design and wiring can be found in the Operating Instructions (http://support.automation.siemens.com/WW/view/en/1123332) of the electronic cam controller FM 452.

Auxiliary voltage for sensors and digital outputs (1L+, 2L+, 3L+)

The 24 VDC auxiliary voltage of the encoders and digital outputs is monitored:

- for wire-break of the 24 V feed line
- for power failure

This means that 24 V DC (terminal 24) and 5.2 V DC (terminal 23) are available at the front connector (X1) for the different types of encoders.

Load current supplies

The DC load current supply must meet the following requirements:

Only a safe, isolated extra-low voltage of ≤ 60 V DC may be used as the load current supply. Safe isolation can be implemented, for example, by adhering to the specifications

- VDE 0100 Part 410 / HD 384-4-41 / IEC 364-4-41
  (as protective extra low voltage) and/or
- VDE 0805 / EN 60950 / IEC 950
  (as safety low voltage SELV) or
- VDE 0106 Part 101.

11 digital inputs (I0 to I10)

You can connect bounce-free switches (24 V current sourcing) or non-contact sensors (2 or 3-wire proximity switches) via 11 digital inputs.

The digital inputs are not monitored for short circuits or wire break and have a non-isolated connection to the module chassis.

The state of each input can be read off from the respective LED.

16 digital outputs (Q0 to Q15)

The state (on/off) of tracks 0 to 15 is output at 16 digital outputs. The digital outputs have a non-isolated connection to the module chassis.

Loads supported:

- Operating voltage 24 V
- Current load 0.5 A / short-circuit proof

The associated LED indicates the state of each output.

Cam control functions

Influence of functions on the switching characteristics of time-based cams

Actual value changes

A time-based cam can be skipped by the following functions that change the actual value:

- Set actual value
- Set actual value on-the-fly
- Zero offset
- Retrigger reference point

Activating a time-based cam

If you skip the start of a time-based cam with one of the above functions, this cam will be activated provided the actual direction of the axis motion is the same as the effective direction assigned for the cam. The programmed cam activation time starts.

Note

If the axis is at a standstill, the direction of movement is influenced by fluctuations of the actual value.

Set a hysteresis higher than the fluctuation of the actual value signal in order to suppress flutter when the axis is at a standstill.

This retains the last determined direction of movement while the axis is at a standstill.

WARNING

Risk of injury and material damage.

Functions which modify the actual value at rotary axes may cause the unwanted activation of time-based cams.

You should always set the "invalid" option for the time-based cams of a rotary axis if you want to influence the actual value using the functions mentioned above.

Set Actual Value/Set Actual Value on-the-fly/Cancel Set Actual Value

Definition

Use the "Set actual value" and "Set actual value on-the-fly" functions to assign a new coordinate to the current encoder value. This shifts the coordinate system by the value: ACT_{new} - $ACT_{current}$

Whereby:

- ACT_{new} is the default value
- $ACT_{current}$ is the process value at the time of execution

Determining new coordinates

Calculate all your default positions in the shifted coordinate system based on the following formula:

$$Coordinate_{new} = Coordinate_{old} + (ACT_{new} - ACT_{current})$$

Requirements

- The axis must be synchronized.
- With "Set process value on-the-fly": Digital input I1 must be interconnected.

Function sequence

1. Enter the coordinate for the process value, or for the process value to set on-the-fly in the channel DB.
   - Linear axis:
     You must select the specified process value so that the software limit switches are still within the valid traversing range after the function is called.
     The absolute value of the offset derived from $(ACT_{new} - ACT_{current})$ must be less than or equal to the absolute value of the permitted traversing range (maximum 100 m or 1000 m).
   - Rotary axis:
     Rule for the defined process value:
     0 ≤ process value < end of rotary axis
2. Set the appropriate trigger bits at the channel DB.
3. Call the CAM_CTRL (Page 7157) or CAM_CTRL_452 (Page 7160) instruction.
   "Set process values" is executed immediately.
   "Set process value on-the-fly" is executed at the next positive edge at digital input I1. The FVAL_DONE bit is set.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
36.4	AVAL_EN	BOOL	FALSE	1 = set process value
36.5	FVAL_EN	BOOL	FALSE	1 = set process value on-the-fly
90.0	AVAL	DINT	L#0	Process value coordinate
94.0	FVAL	DINT	L#0	Coordinate for on-the-fly process value
25.5	FVAL_DONE	BOOL	FALSE	1 = set process value on-the-fly executed

Effects of the function

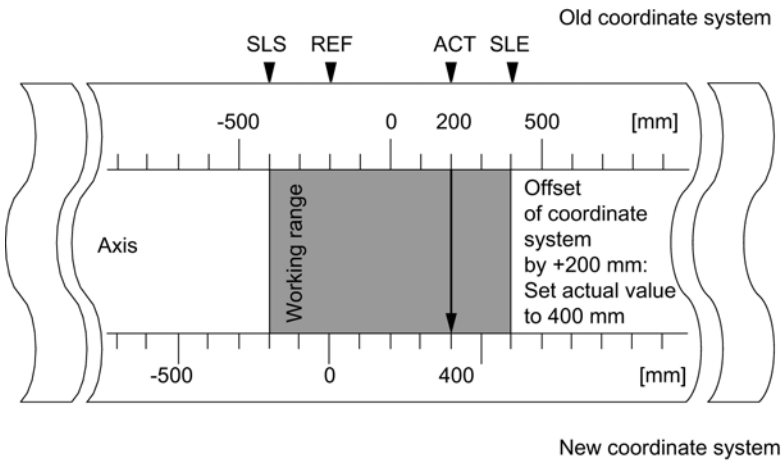
The example "set process value" to 400 mm (at position 200 mm) demonstrates how this function shifts the coordinate system. Resultant effects:

- The position of the working range is **not** physically shifted.
- The individual points (such as the software limit switches) are assigned new coordinate values.
- The cams retain their coordinate values, and are therefore located at a different physical position.
- If the axis is synchronized and cam processing is enabled, the position process value may skip cam edges or entire cams as a result of this function.
- Status changes of the cam which would normally trigger an interrupt may be lost.

Note

Make allowances for the switching characteristics of time-based cams (Page 9436).

The table below shows how "Set process value" / "Set process value on-the-fly" shifts the coordinate system.

Set process value	SLS [mm]	REF [mm]	ACT [mm]	SLE [mm]
	-400	-200	200	400
	-200	0	400	600

Cancelling the function

The "Undo set process value" function can be used to reset the coordinate shift caused by "Set process value" or "Set process value on-the-fly".

Once "Set process value on-the-fly" has been triggered, it can no longer be deleted before execution by a positive edge at input I1. However, it can be overwritten by a new "Set process value on-the-fly".

These functions are reset when a module is started up.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
35.2	AVALREM_EN	BOOL	FALSE	1: Undo process value setting

Possible causes of error

"Set process value on-the-fly" and "Retrigger reference point" may not be executed in parallel.

The "Set process value on-the-fly" function can signal an operator error if the function would cause a software limit switch to exceed the valid traversing range with a positive edge at I1. This operating error is signaled by a diagnostic interrupt, and written to the diagnostic buffer.

Set zero offset

Definition

The "Zero offset" function lets you shift the zero point in the coordinate system by a defined value. The sign determines the offset direction.

Determining a new coordinate

All values of the shifted coordinate system can be calculated according to the following formula:

$\text{Coordinate}_{new} = \text{Coordinate}_{old} - (\text{ZPO}_{new} - \text{ZPO}_{old})$

ZPO_{old} identifies any existing zero offset. If no zero offset was active prior to the call, set a 0 value at ZPO_{old}.

This equation allows you to calculate the coordinates for the software limit switches, for example.

Function sequence

1. Write the zero offset value to the channel DB.
   - Linear axis:
     The zero offset must be selected so that the software limit switches remain within the permissible numerical range after the function is called.
   - Rotary axis:
     Rule for the zero offset:
     value of zero offset ≤ end of the rotary axis.
2. Set the relevant trigger bit.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
36.6	ZOFF_EN	BOOL	FALSE	1 = set zero offset
86.0	ZOFF	DINT	L#0	Zero offset

Effects on a linear axis

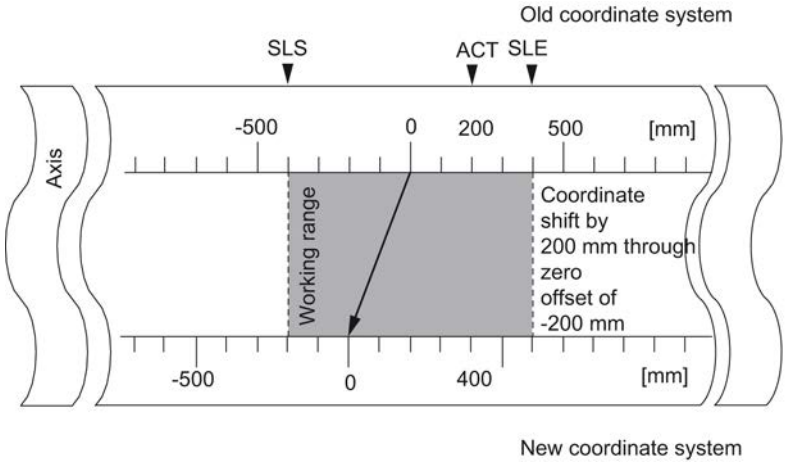
The example of a zero offset of -200 mm demonstrates that this function shifts the coordinate system in positive direction. Resultant effects:

- The working range is **not** physically shifted.
- The individual points (such as the software limit switches) are assigned new coordinate values.
- The cams retain their coordinate values, and are therefore located at a different physical position.
- If the axis is synchronized and cam processing is enabled, the positional process value may skip cam edges or entire cams as a result of this function.
- Status changes of the cam which would normally trigger an interrupt may be lost.

Note

Make allowances for the switching characteristics of time-based cams (Page 9436).

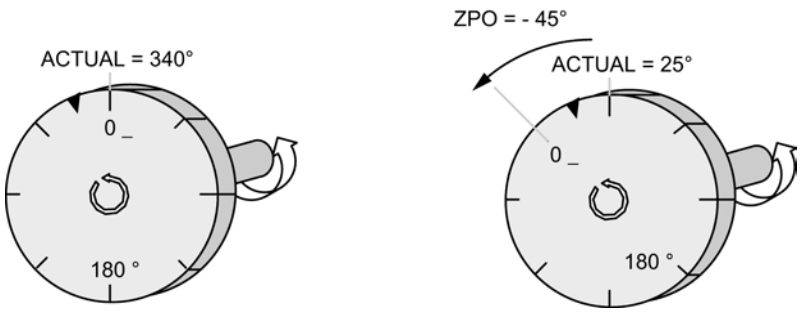
The following table shows how the coordinate system is offset by "Zero offset".

Zero offset	SLS [mm]	REF [mm]	ACT [mm]	SLE [mm]
	-400	-200	200	400
	-200	0	400	600

Effects on a rotary axis

The example of a -45° zero offset demonstrates how this function rotates the coordinate system.

The following table shows how the coordinate system is rotated by "Zero offset".

Tool when ACT = 340°	Tool when ACT = 25°
	
The zero mark rotates by -45 degrees. All axis points are assigned new coordinate values.	

Including a **ZPO_{old} = 0**, the result is a new value of 385°

As process value resumes at 0 at the end of the rotary axis operating in positive directional rotation, the calculation returns an process value of 25°:

$Coordinate_{new}$ = $Coordinate_{old}$ - (ZPO_{new} - ZPO_{old}) - end of rotary axis

The **end of rotary axis** value only needs to be subtracted when

Coordinate$_{old}$ - (**ZPO**$_{new}$ - **NPV**$_{old}$) is greater than the end of rotary axis.

Loss of synchronization

If synchronization is lost due to an error, or reset due to "retrigger reference point", a zero offset is **retained**.

Cancelling function

A zero offset of 0 resets any existing zero offset.

Set reference point

Definition

The "set reference point" function is used to synchronize the axis. The function shifts the working area. All shifts generated by a zero offset or "set actual value" are retained.

The function projects the operating range on the axis. Different value settings will thus move the working range to different positions within the physical range of the axis.

Requirements

Cam processing must be disabled.

Function sequence

1. Write the value for the reference point coordinate to the channel DB.
   - Linear axis:
     The reference point coordinates must not lie outside the software limit switches. This also applies to the reference point coordinate in a shifted coordinate system.
   - Rotary axis:
     Rule for the reference point coordinate:
     0 ≤ reference point coordinate < end of rotary axis
2. Set the relevant trigger bit.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
36.3	REFPT_EN	BOOL	FALSE	1 = set reference point coordinate
98.0	REFPT	DINT	L#0	Reference point coordinate
25.0	SYNC	BOOL	FALSE	1 = axis synchronized

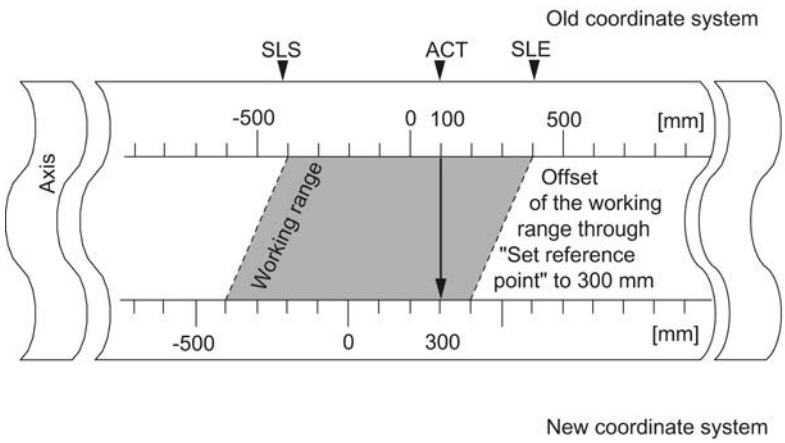
Effects of the function

The example "Set reference point" to 300 mm demonstrates how this function shifts the axis operating range.

This has the following effects:

- The actual position is set to the value of the reference point coordinate.
- The working range is physically shifted on the axis.
- The various points retain their original coordinates, but are now at new physical positions.
- The SYNC bit is set in the checkback signals.

The following table shows how "Set reference point" shifts the operating range on the axis.

Set reference point	SLS [mm]	REF [mm]	ACT [mm]	SLE [mm]
	-400	-200	100	400
	-400	-200	300	400

Special features of the absolute encoder

This function is necessary for absolute encoder adjustment (Page 9465).

Change cam edges

Definition

The "change cam edges" function can be used to change the cam start and, for position-based cams, the end of a specific position-based cam in runtime.

Requirements

The cam you want to change must be valid.

Function sequence

1. Enter the cam number in the channel DB.
2. At a position-based cam:
   Enter the cam start and cam end in the channel DB.
   For a time-based cam:
   Enter the cam start value in the channel DB.
3. Set the relevant trigger bit.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
36.7	CH01CAM_EN	BOOL	FALSE	1 = write function for cam edges (1 cam)
102.0	CAM_NO	INT	0	Cam number
104.0	CAM_START	DINT	L#0	Cam start
108.0	CAM_END	DINT	L#0	Cam end

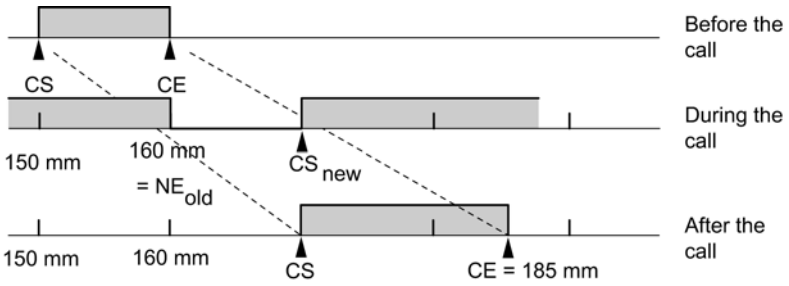
Effects of the function

The FM x52 shifts first the on edge and then the off edge of the cam. This sequence does not depend on the direction in which the cam is shifted.

Exception:

The sequence described above may briefly generate an inverse cam (Page 9340) if the new cam start is greater than the old cam end.

The figure below shows the each stage in cam edge change.



Note

If a hardware interrupt has been enabled for this cam, FM x52 can trigger one or two hardware interrupts when it detects the inverse cam depending on parameter assignment.

Changing the on and/or off edge can result in a cam edge or the entire cam being skipped.

Make allowances for the switching characteristics of time-based cams (Page 9436).

Cam status changes that would normally trigger a hardware interrupt can be lost.

Reading modified values

You can read out the modified values by calling one of the jobs CAM1RD_EN to CAM8RD_EN.

Cancelling the function

The modified values are lost when you restart the module.

Perform "Fast Cam Parameter Change"

Definition

The "Fast cam parameter change" function can be used to modify a group of up to 16 cams in runtime.

Requirements

The cams you want to modify must be valid.

Function sequence

1. Enter the number of cams to be modified in the channel DB.
2. Enter the number of the first cam to be modified in the channel DB.
3. Set the trigger bits for the required modifications.
4. Declare the new values at the channel DB.
5. Repeat steps 2 to 4 for each cam to be modified.
6. Set the relevant trigger bit in the channel DB.

Data used in the channel DB

Address Absolute	Name	Type	Initial value	Comment
37.0	CH16CAM_EN	BOOL	FALSE	1 = write fast cam parameter change function (16 cams)
176.0	C_QTY	BYTE	B#16#0	Number of cams to modify
177.0	DIS_CHECK	BOOL	FALSE	1 = disable data check

Address relative	Name	Type	Initial value	Comment
+0.0	CAM_NO	BYTE	B#16#0	Number of the cam to modify
+1.0	C_EFFDIR	BOOL	FALSE	1 = change the effective direction of the cam
+1.1	C_CBEGIN	BOOL	FALSE	1 = change the cam start to the value CBEGIN

Address relative	Name	Type	Initial value	Comment
+1.2	C_CEND	BOOL	FALSE	1 = change the cam end / activation time to the value CEND
+1.3	C_LTIME	BOOL	FALSE	1 = change the lead time to the value LTIME
+1.4	CAM_OFF	BOOL	FALSE	1 = deactivate the cam during the cam modification
+1.5	EFFDIR_P	BOOL	FALSE	1 = positive effective direction (plus)
+1.6	EFFDIR_M	BOOL	FALSE	1 = negative effective direction (minus)
+2.0	CBEGIN	DINT	L#0	New cam start
+6.0	CEND	DINT	L#0	New cam end / new activation time
+10.0	LTIME	INT	L#0	New lead time

Deactivating a cam when making modifications

To maintain consistency, always deactivate the cam (CAM_OFF) when modifying its start and end settings.

Data check by the module

Use the DIS_CHECK (channel DB) parameter to specify whether or not to disable the validation of transferred data by FM x52. If you disable the data check, you must take steps to verify that only permitted values are transferred. If you transfer illegal values without a check, the module can behave unexpectedly.

- FALSE: The module checks all data to be transferred
- TRUE: The check of cam parameter data is disabled. This allows for faster activation of the data to be changed on FM x52.
  Regardless of this function, the module always checks whether
  - the axis is assigned
  - the number of cams to be changed (C_QTY) is valid
  - the cam (cam number) to change is valid.

The data are only enabled on the module after having been checked and found faultless.

Any faulty data are rejected.

Effects of the function

Note

Make allowances for the switching characteristics of time-based cams (Page 9436).

Reading modified values

You can read the modified values by calling one of the jobs CAM1RD_EN to CAM8RD_EN.

Cancelling the function

The modified values are lost when you restart the module.

Executing "Length measurement" and "Edge detection"

Definition

You can establish the length of a workpiece using the "Length measurement" and "Edge detection" functions.

Length measurement and edge detection are active and remain active until you disable these functions or select a different measuring method. The CAM_CTRL (Page 7157) or CAM_CTRL_452 (Page 7160) instruction enables length measurement if you select both measuring methods in parallel.

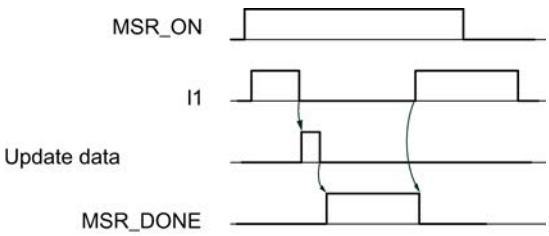
Requirement

A **bounce-free** switch must be connected to input I1.

Function sequence

The FM 452 updates the data on the module at different times Depending on the type of measurement. FM 452 reports each update at a parameter on the feedback interface.

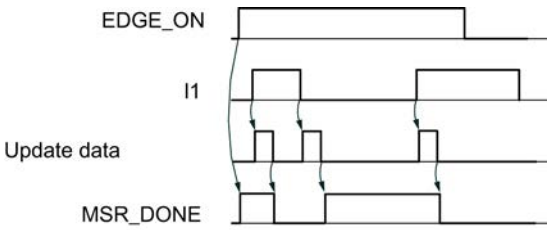
Length measurement



1. Set the function switch for "length measurement".
2. A positive edge at input I1 starts the length measurement.
3. The negative edge at input I1 stops the current measurement. The FM 452 updates the initial value, final value and length data.
4. When parameter MSR_DONE is set, FM 452 reports the data update. The parameter indicates completion of the measurement. The results of the measurement can be read out.
5. The start of the next measurement with a rising edge at I1 resets the MSR_DONE parameter.

The FM 452 does not update the data if the function is disabled during length measurement. The MSR_DONE parameter remains reset.

Edge detection



1. Enter a value for the minimum edge distance in the parameter DB. Write and enable machine data.
2. Set the "edge detection" function switch. This sets the MSR_DONE parameter.
3. The positive edge at input I1 starts edge detection. The results of the measurement are updated and can be read out, and the start value of the measurement is entered; the end value and length assume the value -1.
4. After the update, FM 452 reports the change by resetting the MSR_DONE parameter.
5. The negative edge at input I1 stops the current measurement. The FM 452 updates the data for the end measurement and length values.
6. After the update, FM 452 reports the change by setting the MSR_DONE parameter. The results of the measurement can be read out.
7. The start of the next measurement with a rising edge at I1 resets the MSR_DONE parameter.

The FM 452 does not update the data if the function is disabled during edge detection. The MSR_DONE parameter remains reset.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
25.1	MSR_DONE	BOOL	FALSE	1 = length measurement completed
34.0	EDGE_ON	BOOL	FALSE	1 = edge detection on
34.2	MSR_ON	BOOL	FALSE	1 = length measurement on
38.2	MSRRD_EN	BOOL	FALSE	1 = read measured values
112.0	BEG_VAL	DINT	L#0	Initial value
116.0	END_VAL	DINT	L#0	End value
120.0	LEN_VAL	DINT	L#0	Length

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
4.0	EDGEDIST	DINT	L#0	**Minimum edge distance for edge detection** Valid value range: • 0 ... 1,000,000,000 µm

The minimum edge distance is used to define a range after the start of a measurement is detected with an edge detection. The measurement will be discarded if the measuring operation ends within this range.

The start of the measurement is not signaled unless the minimum edge distance has been traveled.

Conditions of length measurement

- The CPU program requires an interval of sufficient length between the off and on edges at input I1 in order to be able to evaluate the result of the measurement before a new measurement is triggered.
- The minimum interval between the positive and negative edges at input I1 and between the negative edge and the next positive edge at input I1 must be greater than 2 ms.

hardware interrupts

A hardware interrupt can report the start and end of a measurement in the FM 452.

Faulty measurement

The FM 452 returns the length value -1 in the event of errors in length measurement / edge detection.

A "length measurement" or "edge detection" may only perform up to 126 zero passes in one direction. The zero pass represents the rotary axis transition from the end of rotary axis value to 0 and vice versa. The FM 452 reports incorrect "length measurement" or "edge detection" if it detects more than 126 zero passes in one direction, regardless of whether or not it then detects any zero passes in the opposite direction.

A length measurement is also considered faulty if:

- the length measured at a rotary axis is greater than 2^{31}.
- the on and off edges are detected simultaneously by the FM 452 (for example due to switch bounce).

This event can be signaled as a hardware interrupt.

Shift of the coordinate system during a length measurement

Conditions under which a shift of the coordinate system will influence the measured length:

- You are using an incremental encoder or an initiator or are operating the FM 452 in simulation mode.
- You are executing a "set reference point" or "retrigger reference point" function while a length measurement is active.

Example

To utilize the above mentioned influences on the measured length:

Your system always develops slip when you perform a length measurement.

The "Retrigger reference point" function can be used to correct this slip to allow correct length measurement values to be output.

Retrigger reference point

Definition

The "Retrigger reference point" function can be used to synchronize the axis as a reaction to a recurring external event.

The function remains active until you deactivate it.

Requirements

- You are using an incremental encoder or an initiator.
- The external event may represent the zero mark signal of an incremental encoder or reference point switch at input I2.

Function sequence

1. Set the value for the reference point coordinate at the parameter DB.
2. Set the type of "retrigger reference point" at the parameter DB.
   Options:
   – Evaluation of the zero mark of the encoder only (RETR_TYPE = 7).
   – Evaluation of the reference point switch only (RETR_TYPE = 6).
   – Evaluation of the zero mark signal
     in positive direction: Evaluation of the first rising edge of the zero mark after passing the reference point switch in positive direction (RETR_TYPE = 0).
     in negative direction: Evaluation of the first falling edge of the zero mark after passing the reference point switch in negative direction (RETR_TYPE = 1).
3. Write and enable machine data.
4. Set the function switch in the channel DB.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
34.3	REFTR_ON	BOOL	FALSE	1 = retrigger reference point
25.0	SYNC	BOOL	FALSE	1 = axis is synchronized

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
44.0	REFPT	DINT	L#0	Reference point coordinate
52.0	RETR_TYPE	DINT	L#0	Reference point retrigger mode

Effects of the function

- FM x52 evaluates the zero mark and reference point switch in line with the axis direction of motion.
  - It evaluates the positive edges when the axis moves in positive direction.
  - It evaluates the negative edges when the axis moves in negative direction.
- It sets the actual position to the value of the reference point coordinate.
- The working range is physically shifted on the axis.
- The various points retain their original values, but are now at new physical positions.
- Cam status changes that would normally trigger a hardware interrupt can be lost.
- The SYNC bit in the checkback signals is set.

Note

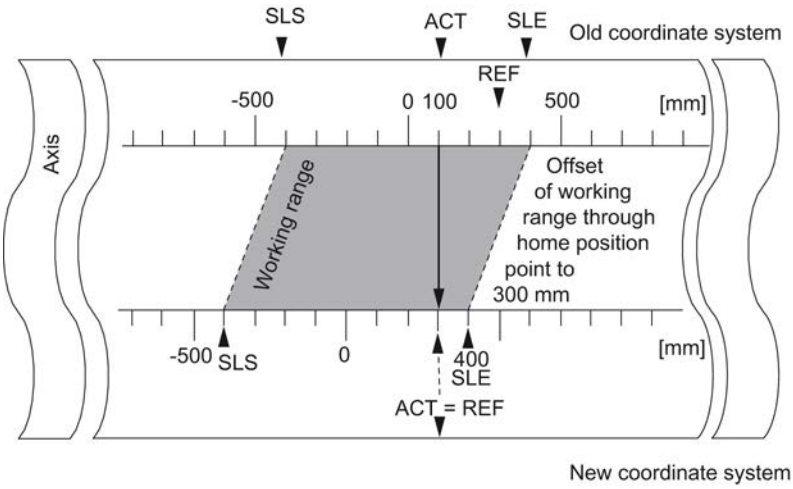
Make allowances for the switching characteristics of time-based cams (Page 9436).

Example

Rules for the example:

- The module evaluates the positive edges of the reference point switch and zero mark signals (axis moving in positive direction).
- Value of the reference point coordinate = 300 mm.
- No zero offset is active at the time of execution.

The following table shows how "Retrigger reference point" shifts the operating range on the axis.

Retrigger reference point	SLS [mm]	REF [mm]	ACT [mm]	SLE [mm]
	-400	300	100	400
	-400	300	300	400

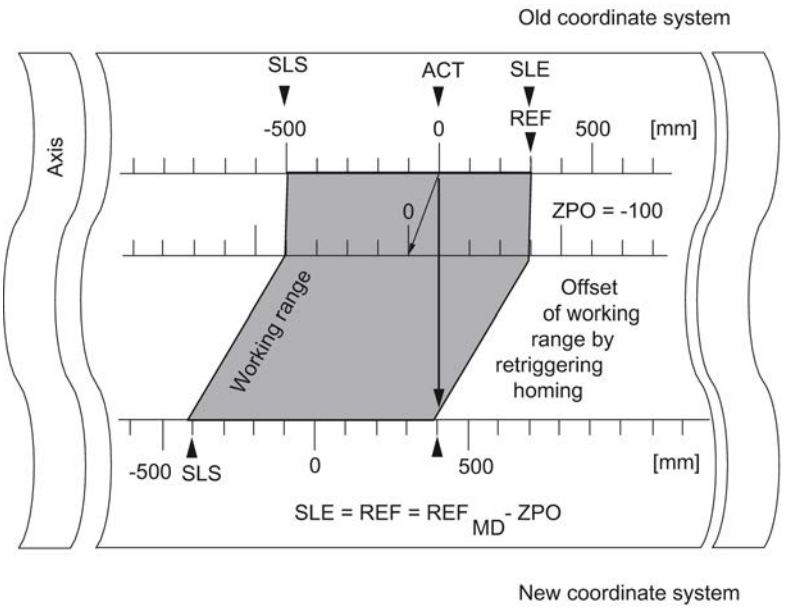
Inclusion of a zero offset

Any active zero offset is included in the "Retrigger reference point" function. The reference point coordinate setting is thus calculated according to the formula:

Ref = Ref_{MD} - Zero offset

Ref_{MD} is the value of the reference point coordinate store in machine data.

The following table shows how "Retrigger reference point" with zero offset shifts the operating range on the axis.

Retrigger reference point	SLS [mm]	REF [mm]	ACT [mm]	SLE [mm]
	-500	300	0	300
	-400	400	100	400
	-400	400	400	400

Deactivate software limit switch

Definition

Use the "Deactivate software limit switch" function to disable monitoring of the software limit switches at a linear axis.

The function remains active until you deactivate it. This re-activates the software limit switches originally parameterized.

Function sequence

Set the function switch in the channel DB.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
34.4	SSW_OFF	BOOL	FALSE	1 = Software limit switches deactivated

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
64.0	SSW_STRT	DINT	L#-1000000000	Software limit switch start
68.0	SSW_END	DINT	L#1000000000	Software limit switch end

Effects of the function

- simulation
  - Simulation mode stops when the axis overruns a software limit switch.
  - You can resume simulation mode by disabling software limit switch monitoring. The axis moves in the defined direction.
- Zero offset when monitoring is disabled
  If a zero offset is set in which software limit switches are located within traversing range limits, the actual value may still be outside the valid number range.
- Cams located outside the parameterized software limit switches can be switched.

CAUTION

Risk of material damage!

If you reduce the traversing range by setting software limit switches to protect your system, you risk damage to your equipment when these switches are disabled.

In the planning and engineering phases for your plant, you should be certain that the drive can travel in the entire physical traversing range.

Executing "Simulation"

Definition

The "Simulation" function allows you to activate the cam controller without connected encoders.

Function sequence

1. Set the simulation velocity at the parameter DB.
2. Write and enable machine data.
3. Set either a positive or negative simulation direction at the channel DB.
4. Set the function switch in the channel DB.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
15.2	DIR_M	BOOL	FALSE	1 = simulation in negative direction
15.3	DIR_P	BOOL	FALSE	1 = simulation in positive direction
34.1	SIM_ON	BOOL	FALSE	1 = simulation on

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
84.0	SIM_SPD	DINT	L#0	Simulation speed

Effects when simulation mode is activated

- Encoder signals will be ignored.
- The encoder input monitoring functions are disabled.
- All encoder errors reported will be reset.
- FM x52 simulates axis motion at a constant simulation velocity.
- Cam processing is disabled while simulation mode is active. However, you can then enable these operations again. Synchronism will be retained in this case.
- Starting at the current actual value, the actual position value changes dynamically based on the simulation velocity and direction.

Effects when simulation mode is deactivated

- Cam processing will be disabled.
- The synchronization of an incremental encoder or initiator will be cleared.. The actual value will be reset to the value of the reference point coordinate.
- The module reports the actual position value which corresponds with the absolute encoder value. The module then evaluates the encoder signals as defined at the machine parameters.

Limit values

The minimum and maximum simulation velocity depend on the resolution (Page 9470).

Velocity

There may be differences between the set velocity and the actual velocity (Page 9470) at which the module operates.

Read "count values of counter cam tracks"

Definition

The "Counter cam track counter values" function is used to read out the current counter values.

Function sequence

1. Specify the counter cam tracks and the high count values in the machine data.
2. Write and enable machine data.
3. Enable the count function.
4. The module sets the count value to its high limit.
5. The count value decrements by the count of 1 at each positive edge of the track result signal.
6. Set the trigger bit at the channel DB to read the count values.
7. The module writes both count values to the channel DB. The module outputs a 0 value for tracks not assigned as a counter cam track.
8. The module sets track identifier bit = 1 at the counter cam track when the count value = 0.
9. It resets the track identifier bit = 0 at the next negative edge of the track result signal, and resets the counter to its high limit.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
15.5	CNTC0_EN	BOOL	FALSE	1 = enable count function of counter cam track 0
15.6	CNTC1_EN	BOOL	FALSE	1 = enable count function of counter cam track 1
38.3	CNTTRC_EN	BOOL	FALSE	1 = read count values of counter cam tracks
124.0	CNT_TRC0	INT	0	Current count value of counter cam track 0
126.0	CNT_TRC1	INT	0	Current count value of counter cam track 1

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
99.0	SPEC_TRC0	BOOL	FALSE	1 = track 0 is counter cam track
99.1	SPEC_TRC1	BOOL	FALSE	1 = track 1 is counter cam track
100.0	CNT_LIM0	DINT	L#2	High count value of counter cam track 0
104.0	CNT_LIM1	DINT	L#2	High count value of counter cam track 1

Read out "Position and track data"

Definition

The "Position and track data" function can be used to read the current position, the velocity and the track identifier bits. The track identifier bits are recorded before being logically linked to machine and channel data.

The algorithm implemented in the FM x52 calculates velocity changes greater than 1 pulse/ 4 ms. The indicated velocity includes this inaccuracy, and is thus unsuitable in particular for closed-loop control. The internal velocity value used for dynamic cam control offers a higher precision.

Function sequence

1. Set the trigger bit in the channel DB.
2. The data are stored in the channel DB.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
38.4	ACTPOS_EN	BOOL	FALSE	1 = read position and track data
128.0	ACTPOS	DINT	L#0	Current position
132.0	ACTSPD	DINT	L#0	Current velocity
136.0	TRACK_ID	DWORD	DW#16#0	Track identifier bits of tracks 0 to 31

Read out "Encoder data"

Definition

The "Encoder data" function can be used to read out current encoder data and the value for absolute encoder adjustment.

Requirements

You can read out the value for absolute encoder adjustment (Page 9465) after you have implemented the "Set reference point" (Page 9442) function.

Function sequence

1. Set the trigger bit in the channel DB.
2. The data are stored in the channel DB.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
38.5	ENCVAL_EN	BOOL	FALSE	1 = read encoder values
140.0	ENCVAL	DINT	L#0	Encoder value / counter value (internal representation)
144.0	ZEROVAL	DINT	L#0	Counter value at the last zero mark (internal representation)
148.0	ENC_ADJ	DINT	L#0	Absolute encoder adjustment

Read out "Cam and track data"

Definition

The "Cam and track data" setting can be used to read the current cam and track identifier bits and the position. The track identifier bits are recorded before being logically linked to machine and channel data.

Function sequence

1. Enter type ID = 1 at the FM_TYPE parameter of the channel DB. This allows you to read 24 bytes of cam and track data.
   If you enter type ID = 0, only the cam identifier bits (16 bytes) will be read.
2. The data are stored in the channel DB.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
12.0	FM_TYPE	BOOL	FALSE	0 = FM 352 to V4.0 1 = FM 352 V5.0 or higher and FM 452
38.6	CAMOUT_EN	BOOL	FALSE	1 = read cam and track data
152.0	CAM_00_31	DWORD	DW#16#0	Cam identifier bits for cams 0 to 31
156.0	CAM_32_63	DWORD	DW#16#0	Cam identifier bits for cams 32 to 63
160.0	CAM_64_95	DWORD	DW#16#0	Cam identifier bits for cams 64 to 95
164.0	CAM_96_127	DWORD	DW#16#0	Cam identifier bits for cams 96 to 127
168.0	TRACK_ID1	DWORD	DW#16#0	Track identifier bits of tracks 0 to 31
172.0	ACTPOS1	DINT	L#0	Current position

Setting "Control signals for the cam controller"

Definition

The "Control signals for the cam controller" function can be used to enable cam processing and the tracks.

Function sequence

1. Set the required bits at the channel DB.
2. The data are transferred to the module each time the CAM_CTRL (Page 7157) or CAM_CTRL_452 (Page 7160) instruction is called.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
15.4	CAM_EN	BOOL	FALSE	1 = enable cam processing
16.0	TRACK_EN	WORD	W#16#0	Enable cam tracks 0 to 12 Bit 0 = track 0

Effects

Cam processing will be started or stopped based on the enable status.

The track identifier bits of enabled tracks are transferred to the track signals and digital outputs.

Querying "Checkback signals for the cam controller"

Definition

The "Checkback signals for the cam controller" function provides information about the current status of the cam controller and track signals. Consistency between the reported position and track signals is not guaranteed.

Function sequence

The data is stored in the channel DB each time the CAM_CTRL (Page 7157) or CAM_CTRL_452 (Page 7160) instruction is called.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
23.4	CAM_ACT	BOOL	FALSE	1 = cam processing busy
26.0	ACT_POS	DINT	L#0	Current axis position
30.0	TRACK_OUT	DWORD	DW#16#0	Current track signals of tracks 0 to 31 Bit 0 = track 0

Querying the "Checkback signals for diagnostics"

Definition

The "checkback signals for diagnostics" setting is used to report diagnostics events.

Programming steps

1. If the module enters a new event in the diagnostic buffer, it sets the DIAG bit in the feedback interface. Error (Page 9503) events are logged in the diagnostics buffer.
2. The module sets the DATA_ERR bit in the feedback interface when it detects incorrect data in a write job. The cause of the error is logged to the diagnostic buffer.
3. The feedback interface stores the CAM_CTRL (Page 7157) or CAM_CTRL_452 (Page 7160) instruction in the channel DB.
4. The module resets the DIAG bit to 0 after the diagnostic buffer has been read by the CAM_DIAG (Page 7158) or CAM_DIAG_452 (Page 7162) instruction.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
22.2	DIAG	BOOL	FALSE	1 = diagnostic buffer modified
22.4	DATA_ERR	BOOL	FALSE	1 = data error

Configuring FM 452

Configuring FM 452

Measuring system

Selecting the system of units

Selecting the system of units

For the input and output of the data, you can select from the following systems of units:

- Millimeter (default setting)
- Inches
- Degrees
- Pulses

Note

If you change the system of units in the "Measuring system" dialog, the values will be converted to the new system. This may lead to rounding errors.

If you change the system of units via the machine date, the values are not automatically converted.

Any change of the system of units from or to "pulses" disables cam processing and prevents axis synchronization.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
8.0	UNITS	DINT	L#1	**Measuring system** Valid values: • 1 = 10^{-3} mm • 2 = 10^{-4} inch • 3 = 10^{-4} degrees • 4 = 10^{-2} degrees • 5 = pulses • 6 = 10^{-3} degrees

Standard system of units in the online help

To specify the limits, we use the mm system of units. To define the limits in other systems of units, convert the values as shown below:

To convert....		you calculate...
mm → inch		Limit value (inches) = limit value (mm) × 0.1 [1]
mm → degrees	10^{-4} (4 decimal places)	Limit value (degrees) = limit value (mm) x 0.1
	10^{-3} (3 decimal places)	Limit value (degrees) = limit value (mm) x 1
	10^{-2} (2 decimal places)	Limit value(degrees) = limit value (mm) x 10
mm → pulse		Limit value (pulses) = limit value (mm) x 1000

[1] When set to the maximum value, the number of decimal places affects the number of digits before the decimal point. In the inch measurement system, 4 decimal places are used; you can therefore enter a maximum of 100,000.0000 inches. The millimeter measurement system uses 3 decimal places; a maximum of 1,000,000.000 mm can be entered.

Axis

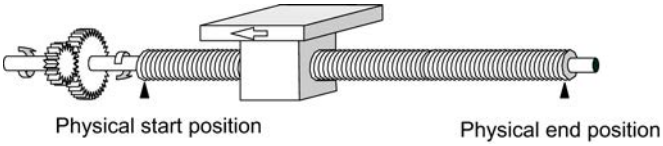
Specifying axis type

Selecting axis type

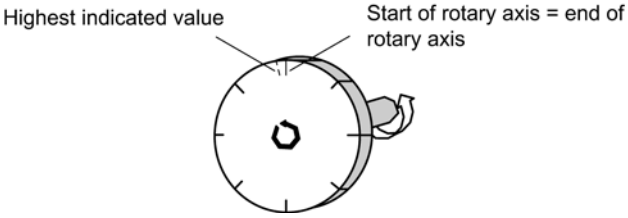
Axis types

The cam controller supports the following axis types:

- Linear axis
  A linear axis is an axis which has a limited physical traversing range.



- Rotary axis
  The rotary axis is an axis whose traversing range is not restricted by mechanical stops.



Data used in the parameter DB

Address	Name	Type	Initial value	Comment
12.0	AXIS_TYPE	DINT	L#0	**Axis type** Valid values: • 0 = linear axis • 1 = Rotary axis

Specifying software limit switches

Software limit switch

Software limit switches only exist for the linear axis.

The operating range is the range limited by the software limit switches. The FM x52 can monitor the operating range limits.

The software limit switch start (SLS) must always be smaller than the software limit switch end (SLE).

The software limit switches are active when FM x52 is synchronized.

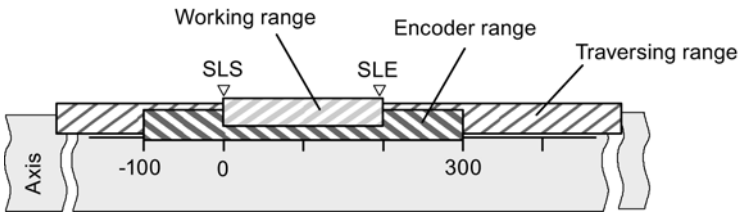
- When using an incremental encoder or initiator:
  The axis is initially not synchronized after each FM x52 startup. The parameterized software limit switches are not monitored until the axis is synchronized.
- When using an absolute encoder (SSI):
  The FM x52 is synchronized once it has received a complete frame without errors. The software limit switches are monitored from this time on. The absolute encoder you use must cover at least the operating range including the software limit switches.

Relationship between the operating range, encoder range and traversing range

- The operating range is the range which you specify for your job using the software limit switches.
- The encoder range represents the range explicitly covered by the encoder. For a linear axis, the module assigns the encoder range symmetrically across across the operating range. In other words, the module shifts the encoder range in order to equalize the distances between the software limit switches and the ends of the encoder range (see figure below).
- This traversing range is the value range the FM x52 can process. The traversing range is dependent on the resolution.

The following applies: Traversing range ≥ encoder range ≥ operating range

The following figure shows the relationship between software limit switches, the operating range, the encoder range and the traversing range.



Data used in the parameter DB

Address	Name	Type	Initial value	Comment
64.0	SSW_STRT	DINT	L# -100 000 000	**Software limit switch start**
68.0	SSW_END	DINT	L# 100 000 000	**Software limit switch end** Valid value range: • -1,000,000,000 µm to 1,000,000,000 µm

Specifying end of rotary axis

End of rotary axis

The "End of rotary axis" value is the theoretical maximum actual value of the axis. However, the theoretical maximum value is never indicated, because it also represents the physical start position of the rotary axis (= zero).

The highest rotary axis value displayed is:

End of rotary axis [µm] - resolution [µm / pulse] x 1 [pulse]

Example: End of rotary axis = 1000 mm

The displayed value jumps:

- from 999 mm to 0 mm when the rotational direction is positive.
- from 0 mm to 999 mm when rotational direction is negative.

Rotary axis with absolute encoders

The rotary range (0 to end of rotary axis) of a rotary axis with an absolute encoder must correspond exactly to the total number of steps of the absolute encoder:

$$\text{Rotary axis end[µm]} = \text{Number of revolutions(encoder)} * \frac{\text{steps(encoder)[pulse]}}{\text{Revolution}} * \text{RES}[\frac{\text{µm}}{\text{Pulse}}]$$

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
16.0	ENDROTAX	DINT	L#100000	**End of rotary axis** Valid value range: • 1 µm to +1,000,000,000 µm

Specifying synchronization

Selecting "Retrigger reference point" type

Retrigger reference point

By setting the "Retrigger reference point (Page 9450)" type, you define the conditions for axis synchronization in operation with an incremental encoder or an initiator.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
52.0	RETR_TYPE	DINT	L#0	**Type of reference point retrigger** Valid values: • 0 = reference point switch and zero mark direction + • 1 = reference point switch and zero mark direction - • 6 = reference point switch only • 7 = zero mark only

Specifying reference point coordinate

Reference point coordinate

- **Incremental encoder and initiator:**
  Using the "Retrigger reference point" function switch and a synchronization event defined by the "Retrigger reference point" type (Page 9464), the reference point coordinate is assigned to this event.
- **Absolute encoder (SSI)**
  A parameterized axis with absolute encoder is always synchronized after transmission of the first error-free SSI frame.
  For more information, refer to the description of absolute encoder adjustment (Page 9465) which explains the interaction of the absolute encoder adjustment with other data.

The value of the reference point coordinate must lie within the operating range:

- Linear axis
  Including the software limit switches.
- Rotary axis
  Greater than or equal to 0 and less than the "end of rotary axis" value (0 ≤ reference point coordinate < end of rotary axis).

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
44.0	REFPT	DINT	L#0	Reference point coordinate Valid value range: • -1,000,000,000 µm to +1,000,000,000 µm

Specifying absolute encoder adjustment

Determining the absolute encoder adjustment

Definition

Absolute encoder adjustment is only necessary for an absolute encoder.

The absolute encoder adjustment and reference point coordinate map the encoder's range of values to the axis coordinate system. The encoder value corresponding to the reference point coordinate on the axis is determined. The value must be less than the total number of steps of the absolute encoder.

Additional information can be found under "Absolute encoder adjustment procedure (Page 9466)" and in an example (Page 9467).

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
48.0	ENC_ADJ	DINT	L#0	Absolute encoder adjustment Valid value range: • 0 to (2^{25} - 1)

Data used in the channel DB

Address	Name	Type	Initial value	Comment
98.0	REFPT	DINT	L#0	Reference point coordinate Valid value range: • -1,000,000,000 µm to +1,000,000,000 µm

Absolute encoder adjustment procedure

Determining the correct absolute encoder adjustment

After initial parameter assignment, further steps must be taken to ensure that a correct relationship is created between the encoder and coordinate system. The procedure shown in the next section is based on the use of the parameter assignment dialogs.

1. Move the axis to a defined, reproducible point to which a unique coordinate is assigned. This can, for example, be the "software limit switch end".
2. Call the "Set reference point (Page 9442)" setting using the coordinate defined in point 1. FM x52 now determines an encoder value for the reference point coordinate entered in the channel DB (REFPT in the channel DB), namely the absolute encoder adjustment. You can read this value with the "Encoder data (Page 9457)" setting.
3. Enter the value read in the "absolute encoder adjustment" field.
4. Save your parameter assignment to the corresponding parameter DB using the export function.
5. Download the data in the HWCN to the CPU.
6. Restart the CPU to activate the data.

Note

You perform this adjustment once during commissioning. Following parameter assignment, the FM x52 is synchronized during start-up as soon as a complete, faultless encoder frame is received after start-up.

Otherwise: Mechanical adjustment of the encoder

A correct relationship between the coordinate system and the encoder can be achieved as follows:

1. Traverse the axis to a defined, reproducible point to which a unique coordinate is assigned. This can, for example, be the "software limit switch start".
2. Enter this coordinate value in the machine data as "reference point coordinate".
3. Read the encoder value displayed at this position with the "Encoder data (Page 9457)" setting.
4. Enter this value in the machine data as the "absolute encoder adjustment".

After the parameter assignment a correct actual value will always be shown.

Instead of steps 3 and 4, you can also set the encoder to zero using "Reset", if available. Then enter the value "0" in the machine data as the absolute encoder adjustment.

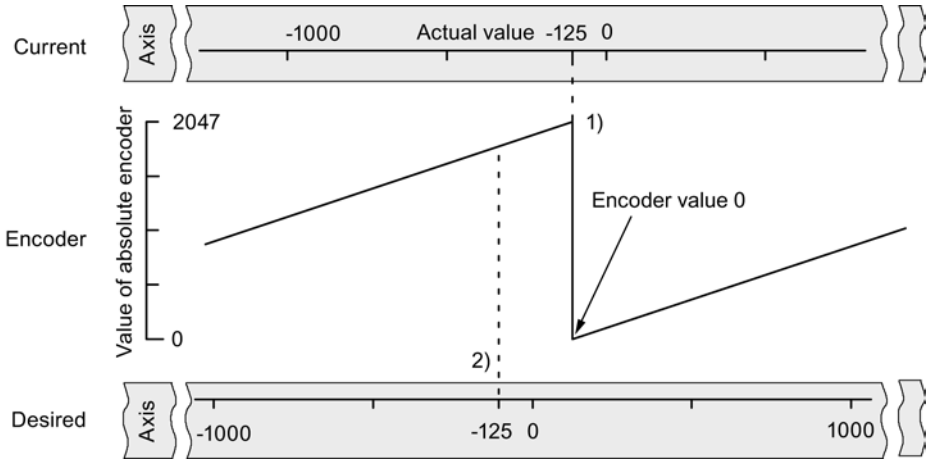
Example of absolute encoder adjustment

Example of an absolute encoder adjustment

The following assumptions apply to the example:

- Reference point coordinate = -125 mm
- Operating range from SSW_STRT = -1000 mm to SSW_END = 1000 mm
- Absolute encoder adjustment = 0
- Encoder range = 2048 increments with a resolution of 1 mm/pulse
- The absolute encoder used cannot be adjusted mechanically with precision and it also is not possible to set the encoder value specifically.

The following figure shows an example of absolute encoder adjustment.

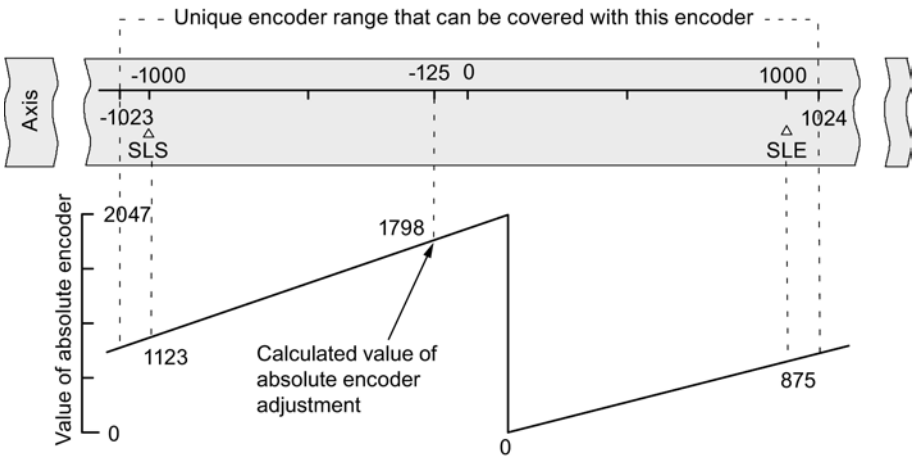


1) Assignment of the coordinate system to encoder values, based on the absolute encoder adjustment. The encoder value 0 corresponds to the actual value -125.
2) Required assignment of the coordinate system to the encoder. At this position the coordinate should be -125.

Result in accordance with "Set reference point"

After "Set reference point" the relationship looks as follows:

The reference point coordinate on the axis (-125) is assigned to the encoder value (1798) determined from absolute encoder adjustment. This assignment is set out in the figure below.



The encoder supplies 2048 unique values. The operating range is defined by the software limit switches. The selected resolution of 1 mm per pulse means the encoder can, however, cover a greater operating range than the one set by the software limit switches.

With the set resolution, the operating range is already covered with 2001 values. Therefore, in the example, 47 pulses "remain" and these are symmetrically positioned around the operating range.

Setting hysteresis

Hysteresis

The hysteresis (Page 9345) value range depends on the resolution (Page 9474).

Maximum input value:

- For linear axes: max. input value < ¼ of the operating range
- For rotary axes: max. input value < ¼ of the rotary axis range

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
80.0	HYS	DINT	L#0	**Hysteresis** Valid value range: • 0 to 65 535 [Imp] x resolution [$^{\mu m}/_{Imp}$]

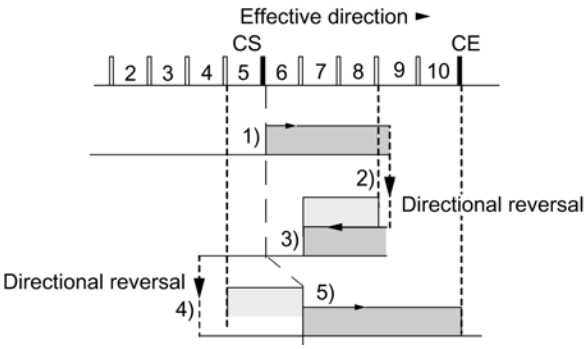
Position-based cams and hysteresis

A position-based cam is activated when the following conditions are met:

- The detected actual value lies within the position-based cam.
- Hysteresis is not active.

Switching points may vary depending on the position at which the change in direction takes place.

The figure below shows the switching characteristics of a position-based cam with hysteresis.



	Cam
	Hysteresis
1)	Cam is activated, as the cam start is crossed in the positive effective direction.
2)	Cam remains active due to hysteresis, irrespective of changes of direction.
3)	Cam is deactivated when hysteresis ends.
4)	Cam remains deactivated after a direction change.
5)	Cam is activated again at the end of hysteresis.

Note

When the direction is changed, hysteresis cannot reduce a cam actuation time shorter than the hysteresis.

Time-based cams and hysteresis

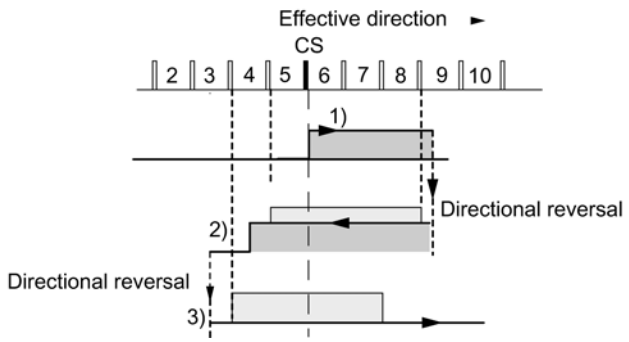
A time-based cam is activated when the following conditions are met:

- The cam start is crossed in the effective direction.
- Hysteresis is not active.

Note

The hysteresis will hide a time-based cam if its range between the reversal point and the cam start is less than the hysteresis.

The figure below shows the switching characteristics of a time-based cam with hysteresis. The figure shows a time-based cam that is **not** activated again.



	Cam
	Hysteresis
1)	Cam is activated, as the cam start is crossed in the positive effective direction.
2)	Cam remains active after the change of direction until the activation time has expired, regardless of hysteresis.
3)	Cam start is hidden by the hysteresis; the cam is not activated.

Specifying simulation speed

Simulation speed

This machine data determines the simulation speed. Please also see the information under "Influence of settings on the switching characteristics of time-based cams (Page 9436)".

The actual simulation speed V_{sim} may deviate from the setting at $V_{sim, V}$, and is calculated using the following formula:

$$V_{Sim} = \frac{6*10exp7*RES}{\text{Integer}\left(\frac{6*10exp7*RES}{V_{Sim, V}}\right)}$$

In this equation:

- V_{sim}: Simulation speed set by FM 352. Unit: $^{\mu m}/_{min}$
- $V_{sim, V}$: simulation speed set in the machine parameters. Unit: $^{\mu m}/_{min}$
- RES: Resolution derived from encoder data. Unit: $^{\mu m}/_{pulse}$
- Integer ( ): Of this expression, only the decimal integer is used in subsequent calculation. This expression must lie within the range of 2 to 65536 in all calculations.

The actual simulation speed changes abruptly as a result of correlations (see above formula).

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
84.0	SIM_SPD	DINT	L#0	**Simulation speed** Valid value range: • 0 = standstill • 5×10^8 = highest setting supported by the module Within this range, the simulation speed depends on the resolution (Page 9474): 1000 x resolution ≤ simulation velocity ≤ 3 x 10^7 x resolution

Specifying minimum edge-to-edge distance

Minimum edge-to-edge distance

This machine data is used to define a range after detection of the start of a measurement for edge detection. The measurement is discarded if the end of the measurement is within this range.

The start of measurement is not reported unless the "minimum edge-to-edge distance" has been traveled.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
4.0	EDGEDIST	DINT	L#0	**Minimum edge-to-edge distance** Valid value range: • 0 ... 1,000,000,000 µm

Encoder

Selecting the signal type

Signal type

The encoder returns position data to the module for evaluation and conversion to an process value based on the resolution.

You can select the signal type from one of the following encoders (Page 9348):

- 5 V differential signal (Page 9350)
  Select the signal type "5 V differential signal" to use a 5 V incremental encoder with differential signal A, /A, B, /B and N, /N.
- 24 V track A+B phase shifted (Page 9350)
  Select signal type "24 V track A+B phase shifted" to use a 24 V incremental encoder with signals A*, B* and N*.
- SSI absolute (Page 9352)
  Select the "SSI absolute" signal type if you use an absolute encoder with a serial interface.
- 24 V initiator (Page 9352)
  Select the signal type "24 V initiator" to use a 24 V initiator with count signal A*.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
20.0	ENC_TYPE	DINT	L#1	**Signal type** Valid values: • 1 = 5 V, incremental • 2 = 24 V, incremental • 3 = SSI 13-bit frame length • 4 = SSI 25-bit frame length • 5 = listen in • 6 = 24 V initiator up (positive) • 7 = 24 V initiator down (negative) • 8 = SSI 13-bit (right-justified) • 9 = SSI 25-bit (right-justified) • 10 = listen in (right-justified)

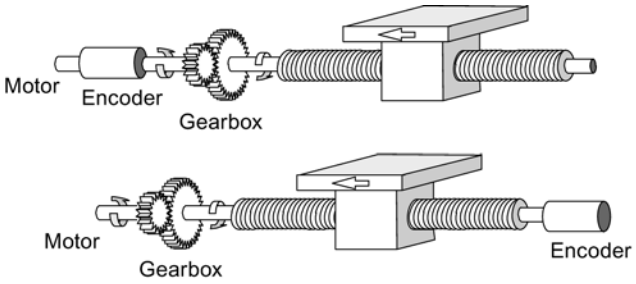
Specifying resolution

Specify distance per encoder revolution

Distance per encoder revolution

You inform the FM x52 about the distance covered by the drive system per encoder revolution using the "Distance per encoder revolution" machine data.

The "Distance per encoder revolution" value depends on the structure of the axis and the way in which the encoder is mounted. You must take all transmission elements, such as couplings and gears, into account. The following figure shows an example of two different encoder positions.



The relationship between the "Distance per encoder revolution" and "Increments per encoder revolution" machine data is described under resolution (Page 9474).

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
24.0	DISP_REV	DINT	L#80000	**Distance per encoder revolution** Valid value range: • 1 µm to 1,000,000,000 µm

Specifying increments per encoder revolution

Increments per encoder revolution

The "Increments per encoder revolution" machine data defines the number of increments an encoder returns per revolution. The FM x52 calculates the resolution based on this value and the "Distance per encoder revolution" machine data.

Incremental encoder

Any value within the range of values can be entered (Page 9350). The module evaluates the increments four times.

Initiator

Any value within the range of values is allowed.

Absolute encoder

Different encoder types have different limits:

Encoder type	Frame length / type	Value range	Can be used as linear axis
Single-turn encoder	13-bit half fir tree	64 ... 8192 in powers of 2	
Single-turn encoder	13-bit right-justified	64 ... 8192 all values	X
Single-turn encoder	25-bit right-justified	64 ... 2^{25} all values	X
Multiturn encoder	25-bit fir tree	64 ... 8192 in powers of 2	
Multiturn encoder	25-bit right-justified	64 ... 2^{24} all values	
Listen-in	Fir tree	64 ... 8192 in powers of 2	

Encoder type	Frame length / type	Value range	Can be used as linear axis
Listen-in	Right justified	64 ... 2^{25} all values	X
Special setting: Multiturn encoder in single-turn mode	25-bit half fir tree	64 ... 8192 in powers of 2	

Note

The number of encoder pulses is the product of the "Increments per encoder revolution" and the "Number of revolutions" (see "Resolution (Page 9474)").

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
32.0	INC_REV	DINT	L#500	**Increments per encoder revolution** Valid value range: • 1 to 2^{25} Note: This entry is irrelevant when you use "pulse" units.

Resolution

Resolution

The resolution reflects the accuracy of cam processing. It also determines the maximum possible traversing range.

The resolution (RES) is calculated as shown in the table below:

	Incremental encoder	Absolute encoder / initiator
Input values	• Distance per encoder revolution • Increments per encoder revolution: – Pulse evaluation: quadruple – 1 increment = 4 pulses	• Distance per encoder revolution • Increments per encoder revolution. – 1 increment = 1 pulse
Calculation	$RES = \frac{\frac{Distance}{Encoder\ revolution}}{\frac{Pulses}{Encoder\ revolution}}$	

Note

Resolution in the "pulse" system of units is always 1.

All position values are rounded to the integer multiple of the resolution. This allows you to distinguish between the set and used values.

Range of values of the resolution

Convert the range of values for the resolution according to the defined physical units. The resolution must be kept within this range when setting the "distance per encoder revolution" and "increments per encoder revolution" values.

The chosen system of units determines the resolution value range:

Measuring system	Representation in...	Range of values of the resolution
mm	10^{-3} mm	0,1 × 10^{-3} mm to 1000 x 10^{-3} mm/pulse
inch	10^{-4} inch	0,1 × 10^{-4} inches to 1000 x 10^{-4} inches/pulse
degrees	10^{-4} degrees 10^{-3} degrees 10^{-2} degrees	0,1 × 10^{-4} degrees to 1000 x 10^{-4} degrees/pulse 0,1 × 10^{-3} degrees to 1000 x 10^{-3} degrees/pulse 0,1 × 10^{-2} degrees to 1000 x 10^{-2} degrees/pulse
Pulses	1 pulse	1

Example

- An incremental encoder has the following data:
  - Increments per encoder revolution: 5000
  - Distance per encoder revolution: 1000 mm
  - 1 increment = 4 pulses

  Resultant resolution (quadruple evaluation):

$$\text{Resolution} = \frac{1000\ \text{mm}}{5000\ \text{increments}} = 0.2000\ \frac{\text{mm}}{\text{Increment}} = 0.2000\ \frac{\text{mm}}{4\ \text{pulses}}$$

$$= 0.0500\ \frac{\text{mm}}{\text{Pulse}}$$

- An SSI encoder has the following data:
  - Increments per encoder revolution: 4096
  - Distance per encoder revolution: 1000 mm
  - 1 increment = 1 pulse

  Resultant resolution:

$$\text{Resolution} = \frac{1000\ \text{mm}}{4096\ \text{increments}} = 0.2441\ \frac{\text{mm}}{\text{Increment}} = 0.2441\ \frac{\text{mm}}{\text{Pulse}}$$

Traversing range dependency on resolution

The traversing range is restricted by the numerical representation in FM x52. This numerical representation varies depending on the resolution. You should thus make sure that your specifications are always within the permitted limits.

The table below sets out the maximum traversing range:

Resolution (RES) is in the range	Maximum traversing range
0.1 µm/pulse ≦ RES < 1 µm/pulse	-10^8 µm to 10^8 µm (-100 m to +100 m)
1 µm/pulse ≦ RES ≦ 1000 µm/pulse	-10^9 µm to 10^9 µm (-1000 m to +1000 m)

Dependency between the velocity and resolution

The velocity indicated may vary within the following limits, depending on the resolution (data relate to mm units):

- from 1 µm/min to 90 m/min, at a resolution < 1 µm/pulse
- from 1 µm/min to 900 m/min, at a resolution ≥ 1 µm/pulse

The velocity value is calculated and filtered by the module at intervals of 4 ms.

Its minimum inaccuracy of one pulse/4 ms rules it out for closed-loop control.

Selecting the count direction

Count direction

You can adapt the position feedback detection to the direction of axis motion using the "Count direction" machine data. You must also take into account the rotational direction of all transmission elements (for example, couplings and gears).

- Normal = incremental count pulses (incremental encoder) or encoder values (absolute encoder) represent increasing positional actual values
- Inverted = incremental count pulses (incremental encoder) or encoder values (absolute encoder) represent decreasing positional actual values

Note:
You cannot implement a rate time in combination with an absolute encoder (SSI) and inverted parameterized count direction.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
59.0	CNT_DIR	BOOL	FALSE	Count direction Valid values: • 0 = normal • 1 = inverted

Selecting monitoring

Monitoring

Wire break

When its monitoring function is enabled, the FM x52 monitors the A, /A, B, /B, N, and /N signals of an incremental encoder. The monitoring function detects:

- Wire break
- Short circuits in individual cables
  For incremental encoders without a zero mark, you must either:
  - Disable the wire-break monitoring function
  - Interconnect the N and /N signals externally (Page 9348)
- Edge-to-edge distance of the count pulses
- Encoder supply failure

Frame error

You cannot switch off the monitoring of frame errors with absolute encoders (SSI). It monitors:

- the frame for start and stop bit errors
- the monoflop time of the connected encoder

Missing pulses (incremental encoder)

An incremental encoder must always return the same number of increments between two successive zero marks.

FM x52 checks whether the zero mark of an incremental encoder coincides with the correct encoder value.

Disable missing pulse monitoring for encoders without zero marks.

Also disable wire-break monitoring, or interconnect the zero mark inputs N and /N externally.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
				Monitoring functions
63.0	MON_WIRE	BOOL	TRUE	1 = wire break
63.1	MON_FRAME	BOOL	TRUE	1 = frame error (must always be 1)
63.2	MON_PULSE	BOOL	TRUE	1 = missing pulses

Selecting data for absolute encoder (SSI)

Number of encoder revolutions

Number of encoder revolutions

The machine data "Number of encoder revolutions" only applies to absolute encoders. You use it to define the maximum possible number of revolutions for this encoder. For additional information on this topic, please refer to "Absolute encoders (SSI) (Page 9352)".

- **Single-turn encoder**
  Only the value 1 is possible.
- **Multiturn encoder**
  - Multiturn encoder / listen-in (fir tree): 2 ... 4096 in powers of 2
  - Multiturn encoder / listen-in (right-justified): 1 ... 4096 all values
- **Linear scale**
  You can also connect a linear scale. To do so, enter the value "1".

The **total number of steps of the encoder** is not machine data. It is calculated as follows:

Overall number of steps = increments per encoder revolution x number of revolutions

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
36.0	NO_REV	DINT	L#1024	**Number of encoder revolutions** Valid value range: • 1 (single-turn encoder) • 1 to 4096 (multiturn encoder)

Transmission rate

Transmission rate

You define the data transmission rate between the SSI encoder and the FM x52 with the "transmission rate" machine data.

The maximum transmission rate depends on the cable length:

- 320 m → 125 kHz
- 160 m → 250 kHz
- 63 m → 500 kHz
- 20 m → 1,000 kHz

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
40.0	BAUDRATE	DINT	L#0	**Transmission rate** Valid values: • 0 = 125 kHz • 1 = 250 kHz • 2 = 500 kHz • 3 = 1000 kHz

Frame length

Frame length

You use "Frame length" to define the clock frame output by the FM x52 for absolute encoders.

Selecting "listen-in" will disable the FM x52 cycle. The FM x52 can then listen in on other SSI frames with a 13 or 25-bit frame. The transmission rate is determined by the cycle clock rate of the master module.

Data used in the parameter DB

The frame length selected here is stored in the parameter DB under the ENC_TYPE (Page 9471) date in the parameter DB.

Initiator direction

Initiator direction

You define the direction of a connected initiator with "Initiator direction":

- Positive: = up (incremental count pulses)
- Negative: = down (downwards count pulses)

Tracks

Selecting track output control mode

Track output control

The "Track output control" machine data defines the control mode for track outputs 0 to 15. Tracks can be activated by:

- The cam controller: Track signals are activated and deactivated by FM 452 cam processing.
- CPU: The track signals represent the corresponding values of the track enables in the channel DB.

This allows the explicit activation of track outputs in the user program.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
90.0	TRACK_OUT	WORD	W#16#0	**Track output control** Valid values: • 0 = cam controller • 1 = CPU Bit number = track number

Selecting special tracks

Special tracks

You can parameterize tracks 0, 1 and 2 for use as special tracks.

- Track 0 and track 1 as counter cam track (Page 9343)
- Track 2 as brake cam track (Page 9343)

Data used in the parameter DB

Address	Name	Type	Start value	Comment
				Special tracks
99.0	SPEC_TRC0	BOOL	FALSE	1 = track 0 is counter cam track
99.1	SPEC_TRC1	BOOL	FALSE	1 = track 1 is counter cam track
99.2	SPEC_TRC2	BOOL	FALSE	1 = track 2 is brake cam track

Specifying high count value for counter cam track

High count value for counter cam track

With this machine data, you define the high count value of the assigned counter cam track.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
100.0	CNT_LIM0	DINT	L#2	**High count value of counter cam track** (track 0)
104.0	CNT_LIM1	DINT	L#2	**High count value of counter cam track** (track 1) Valid value range: • 2 … 65535

Selecting enable input

Enable input

Track signal Q3 to Q10 is activated if the following conditions are met:

- The track is enabled.
- The relevant external enable input I3 to I10 has been set.
- The track result is 1.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
				Enable input
95.0	EN_IN_I3	BOOL	FALSE	1 = track signal track 3 AND enable input I3
…	…	…	…	…
95.7	EN_IN_I10	BOOL	FALSE	1 = track signal track 10 AND enable input I10

Cams

Number of cams

Number of cams

The number of cams determines the maximum number of programmable cams and therefore the cam cycle time.

Number of cams	Cam cycle time
16 cams	20.48 µs
32 cams	40.96 µs
64 cams	81.92 µs
128 cams	163.84 µs

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
76.0	C_QTY	DINT	UDT3: L#0 UDT4: L#1 UDT5: L#2 UDT6: L#3	Number of cams 0 = max. 16 cams 1 = max. 32 cams 2 = max. 64 cams 3 = max. 128 cams

Cam data

Cam data

Cam data

Cam data describe the properties of a cam, the assignment of each cam to a track, and the switching characteristic of the cam.

The cam data below is set separately for each cam:

- Validity (Page 9483)
- Track number (Page 9483)
- Cam type (Page 9484)
- Cam start and cam end of position-based cams (Page 9484)
- Cam start and activation time of time-based cams (Page 9485)
- Lead time (Page 9486)

- Effective direction (Page 9488)
- Hardware interrupt (Page 9490)

The following applies to cam processing in general:

- The module interprets and processes only the cams with "valid" settings.
- Cams 0 to 7 support hardware interrupts.
- The number of assignable cams depends on the number of cams available.

Cam validity

Cam "valid"

All cams to be processed by the cam controller must be set as "valid".

Note

Unused cams should always be set to "invalid" CAMVALID = FALSE).

Data used in the parameter DB

Address relative	Name	Type	Initial value	Comment
+0.0	CAMVALID	BOOL	FALSE	1 = cam "valid"

Specifying a cam track number

Track number

You define the track controlled by the cam by setting the track number.

Data used in the parameter DB

Address Relative	Name	Type	Initial value	Comment
+1.0	TRACK_NO	BYTE	B#16#0	**Track number** Valid value range: • 0 to 31

Selecting cam type

Cam type

Select the required cam type.

- Position-based cam
- Time-based cam

Data used in the parameter DB

Address Relative	Name	Type	Initial value	Comment
+0.3	CAM_TYPE	BOOL	FALSE	**Cam type** Valid values: • 0 = position-based cam • 1 = time-based cam

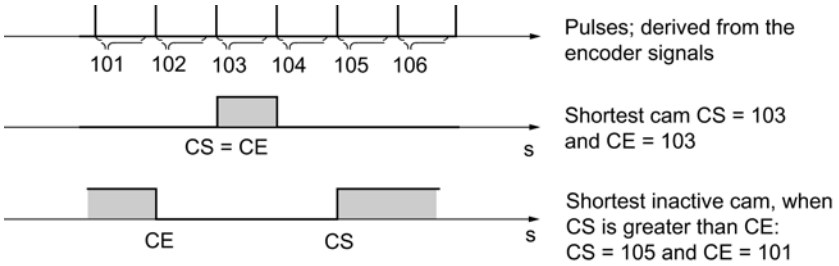
Defining cam start and cam end for position-based cams

Cam start (CS) / Cam end (CE) of position-based cams

The cam start and cam end of a position-based cam define the cam length.

Minimum length of a position-based cam

The figure below shows the shortest cam with axis motion in positive direction



The following general rules apply:

- The inactive part of a cam must always have a minimum interval of 4 pulses between the cam end (CE) and cam start (CS).
- If CE = CS, the cam is activated for the duration of one pulse.

Data used in the parameter DB

Address Relative	Name	Type	Initial value	Comment Of position-based cams
+2.0 +6.0	CBEGIN CEND	DINT DINT	L#-100000000 L#100000000	**Cam start (CS)** **Cam end (CE)** Valid value range: • -1,000,000,000 µm to 1,000,000,000 µm

Specifying cam start and activation time for time-based cams

Cam start (CS) / Cam end (CE) ≙ activation time of time-based cams

With a time-based cam, you must specify a cam start and an activation time in place of the cam end. Define the activation time with a resolution of 100 µs. The time runs starting with the activation of the cam.

Conditions of setting default times:

- 0 µs: A cam with an activation time of 0 µs will never be activated
- 0 µs < t ≤ 400 µs: The FM x52 sets a minimum cam activation time of approx. 330 µs.
- t > 400 µs: FM x52 calculates the actual activation time t_{act} based on the default activation time t_{def} using this formula:

$$t_{act} = \text{integer} \left(\frac{t_{def}}{\text{Cam cycle time}} \right) * \text{Cam cycle time}$$

The maximum error is always less than the cam cycle time.

Data used in the parameter DB

Address Relative	Name	Type	Initial value	Comment of time cams
+2.0 +6.0	CBEGIN CEND	DINT DINT	L#-100000000 L#100000000	**Cam start (CS)** **Cam end (CE) ≙ activation time** Valid cam start value range: • -1,000,000,000 µm to 1,000,000,000 µm Valid cam activation time value range: • (0 to 13421) x 100 µs with up to 16 cams • (0 to 26843) x 100 µs with up to 32 cams • (0 to 53686) x 100 µs with up to 64 cams • (0 to 65535) x 100 µs with up to 128 cams

Specifying lead time

Lead time

You can compensate any delays caused by the connected switchgear by setting a lead time. Define a lead time with a resolution of 100 µs. You can assign one lead time to each cam. The lead time applies to the cam start and cam end.

Actuation distance

The actuation distance of a cam is calculated continuously based on the current velocity and lead time. The entire cam is shifted in direction of the actual value by this value. The assigned range is the "static range", and the range calculated based on the lead time represents the "dynamic range".

Actuation distance = lead time x current velocity

FM x52 calculates the actuation distance of all cams within ¼ of the longest parameterized lead time. A very high lead time setting for a cam reduces the frequency of the dynamic adjustment calculation.

Actual lead time

To calculate the actual lead time:

1. Determine the cam cycle time: This is the time in which the FM x52 fully processes all cams. It depends on the number of cams parameterized.
2. Calculate the actual lead time based on the following formula:

$$\text{Lead time}_{act} = \text{integer}\left(\frac{\text{Lead time}_{d}}{\text{Cam cycle time} * 4}\right) * \text{Cam cycle time} * 4$$

   The identifiers have the following meanings:

   – Lead time $_{act}$ is the lead time set by the FM x52
   – Lead time $_{d}$ is your default setting.
   – Integer ( ) means that the values after the decimal point are ignored in the calculation of the parenthesis.

The maximum error of the lead time$_{act}$ is always less than [cam cycle time] x 4.

Example:

The following values are defined:

- Number of cams: Maximum 32 cams
- Cam cycle time: 40.96 µs
- Lead time$_{d}$ = 1000 µs

Result: An actual lead time of 983 µs.

Note

The actual is always less than the assigned lead time. It may be 0 even though the parameterized lead time ≥100 µs.

The actuation distance of a rotary axis must be less than the rotary axis range and the inactive part of the cam. This must be guaranteed for all velocities.

Note

You may not implement a lead time in combination with an absolute encoder (SSI) and inverted parameterized count direction.

Data used in the parameter DB

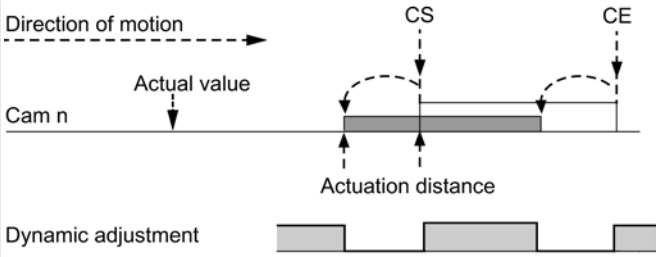
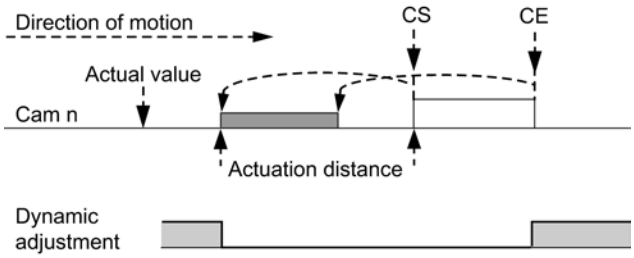
Address Relative	Name	Type	Initial value	Comment
+ 10.0	LTIME	INT	0	**Lead time** Valid value range: • (0 to 53686) x 100 µs with max. 16 cams • (0 to 65535) × 100 µs with max. 32, 64 or 128 cams

Dynamic cam adjustment

There are two distinct situations relating to the cam range:

1. The static and dynamic range of the cam overlap.
2. The static and dynamic range of the cam do not overlap.

The table below shows the calculation of dynamic adjustment in each of the two cases.

Dynamic adjustment	Description
	If the dynamic range of the cam overlaps its static range: • The cam is activated when its dynamic range is reached. At the same time, calculation of a new dynamic adjustment is disabled. • When the actual value reaches the static range of the cam, the calculation of a new dynamic adjustment is enabled again. A change in velocity affects the cam end. • When the cam is deactivated at the end of the dynamic range, dynamic adjustment will be disabled again until the end of the static range of the cam.
	The following applies if the cam's dynamic and static ranges do not overlap: • The cam is activated when its dynamic range is reached. At the same time the calculation of a new dynamic adjustment is disabled. • At the end of the static range of the cam, dynamic adjustment is enabled again.
Dynamic range Static range A **new** dynamic adjustment is possible	

Note

When the direction of rotation changes, calculation of the dynamic adjustment is enabled again.

Selecting effective direction

Effective direction

Two effective directions are supported:

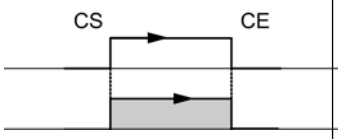
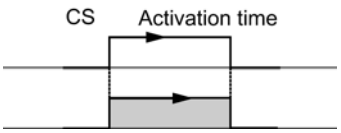
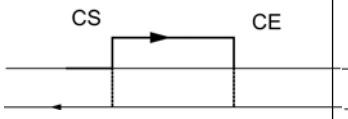
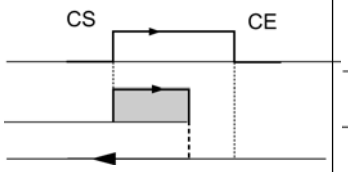
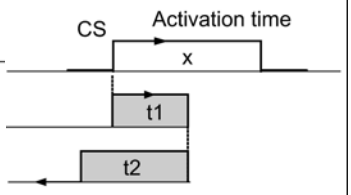
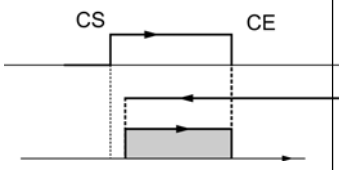
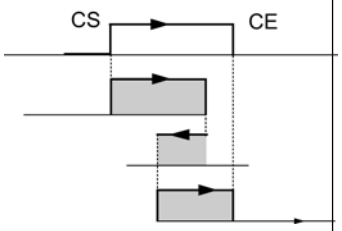
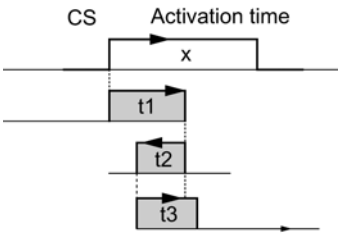
- Positive: The cam is activated at the cam start if the axis is moving in the direction of increasing process values.
- Negative: The cam is activated at the cam end if the axis is moving in the direction of decreasing process values.

You may set both effective directions in parallel.

Selecting "None" disables the cam.

Switching characteristics of cams based on their effective direction

The table below shows the switching characteristics of position-based and time-based cams as determined by the effective direction. A positive effective direction is always assumed, with the exception of example 5.

No.	Description	Position-based cams	Time-based cams
1	A cam is overrun in the effective direction		
2	A cam is traversed in reversed effective direction		
3	A cam is approached in the effective direction; the direction of motion of the axis is reversed while the cam is active		 Activation time x = t1 + t2
4	A cam is approached in reversed effective direction; the axis reverses its direction of movement on the cam in effective direction		Cam not switched
5	A cam is approached from any direction, and exited into any direction; **both directions** are set as effective direction		 Activation time x = t1 + t2 + t3
parameterized cam switched cam			

Data used in the parameter DB

Address Relative	Name	Type	Initial value	Comment
+0.1	EFFDIR_P	BOOL	TRUE	1 = positive effective direction (plus)
+0.2	EFFDIR_M	BOOL	TRUE	1 = negative effective direction (minus)

Selecting hardware interrupt for cam

Hardware interrupt when cam switches on / off

You can specify whether hardware interrupts are to be generated when cams 0 to 7 are enabled, disabled or enabled and disabled.

Data used in the parameter DB

Address Relative	Name	Type	Initial value	Comment
+0.4	PI_SW_ON	BOOL	FALSE	1 = hardware interrupt on activation
+0.5	PI_SW_OFF	BOOL	FALSE	1 = hardware interrupt on deactivation

Enable hardware interrupts

Enabling hardware interrupts

Enabling hardware interrupts

Hardware interrupts are enabled in the machine data.

You can set when hardware interrupts are to be generated:

- Start of measurement
  A rising edge at digital input I1 can trigger a hardware interrupt in the "Edge detection (Page 9447)" setting.
- End of measurement
  With both the "Edge detection (Page 9447)" and "Length measurement (Page 9447)" setting, a falling edge at digital input I1 can trigger a hardware interrupt.
- Cam on/off
  In the cam data (Page 9490), you can specify whether hardware interrupts are to be generated when cams 0 to 7 are activated and/or deactivated.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
3.1	PI_MEND	BOOL	FALSE	1 = enable process interrupt: End of measurement
3.2	PI_CAM	BOOL	FALSE	1 = enable process interrupt: Cam on/off
3.5	PI_MSTRT	BOOL	FALSE	1 = enable process interrupt: Start of measurement

Evaluation of a hardware interrupt

If FM 452 triggers a hardware interrupt, interrupt information is available in the OB4x_POINT_ADDR tags of the hardware interrupt OB.

The following table shows the content of the double word OB4x_POINT_ADDR.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	0	0	0	0	0
1	0	0	Start of measure ment	0	0	Cams	End of measure ment	0
2	Cam 7 on	Cam 7 off	Cam 6 on	Cam 6 off	Cam 5 on	Cam 5 off	Cam 4 on	Cam 4 off
3	Cam 3 on	Cam 3 off	Cam 2 on	Cam 2 off	Cam 1 on	Cam 1 off	Cam 0 on	Cam 0 off

Byte 1 identifies the cause of the interrupt:

- Cam:
  Evaluate bytes 2 and 3 according to the table.
- Start / end of measurement:
  Use the CAM_MSRM_452 (Page 7163) instruction to read the current measured values from the module.

Lost hardware interrupts

If the hardware interrupt OB is still busy processing a hardware interrupt, the module registers all subsequent hardware interrupt events. If an event occurs again before the hardware interrupt could be triggered, the module triggers a "hardware interrupt lost" diagnostic interrupt.

Control and feedback interface of the FM 452

Application

Very fast access to checkback and control signals is required in special applications or in an alarm level. You can obtain those data directly by reading the IO areas of the module.

To coordinate startup following each module startup, for example, after plugging a module or after a STOP-RUN transition of the CPU, you must call the CAM_CTRL (Page 7157) instruction until the end of the startup is indicated by RETVAL = 0.

Note

For direct access to FM 452 data, always use the non-internal data and method described in this section. Otherwise, your user program will encounter difficulties accessing the module.

Reading checkback signals via direct access

The byte addresses are defined with an offset to the output address of the module. The names of the parameters correspond to the names in the channel DB (Page 7164).

In STL, you access the data using the PIB (read 1 byte) and PID (read 4 bytes) commands.

Address	Bit number							
	7	6	5	4	3	2	1	0
Byte 0	PARA	Internal	Internal	DATA_ERR	Internal	DIAG	Internal	Internal
Byte 1	0	0	0	CAM_ACT	0	0	0	0
Byte 2	Internal							
Byte 3	0	0	FVAL_DONE	HYS	GO_P	GO_M	MSR_DONE	SYNC
Byte 4	ACT_POS							
Byte 5								
Byte 6								
Byte 7								
Byte 8	TRACK_OUT							
Byte 9								
Byte 10								
Byte 11								

Direct write access to control signals

The byte addresses are defined with an offset to the input address of the module. The names of the parameters correspond to the names in the channel DB (Page 7164).

In STL, you access the data using the PQB (write 1 byte) and PQW (write 2 bytes) commands.

Address	Bit number							
	7	6	5	4	3	2	1	0
Byte 0	Internal							
Byte 1	0	CNTC1_EN	CNTC0_EN	CAM_EN	DIR_P	DIR_M	0	0
Byte 2	TRACK_EN							
Byte 3								

Example: Position process value ACT_POS

The start address of the module is 512.

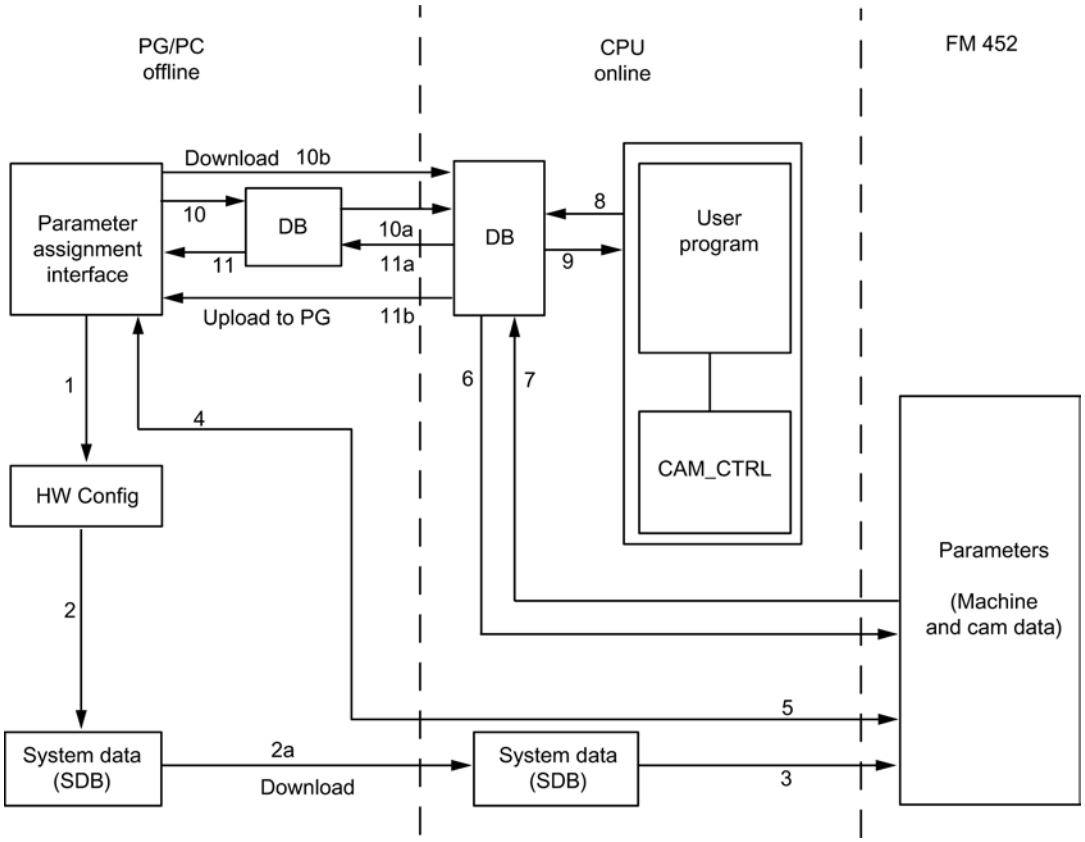
STL

```
L PID 516      //Read the position process value (ACT_POS) with direct access:
               //Module start address + 4
```

Parameter transmission paths

The term parameter refers to the following machine and cam data.

The following figure shows the parameter transmission paths of the FM 452.



1 Save the parameters in the parameter assignment interface.

2 Saving and compiling the HW configuration, and download to the CPU.

3 The CPU writes the parameters to the module during system parameter assignment.

4 Upload the module parameters to the PG by selecting the "PLC > Upload to PG" command.

5 Download the parameters from the parameter assignment interface to the module with the "PLC > Download" command.

6 Write parameters to the module using jobs in the user program.

7 Read parameters from the module using jobs in the user program.

8 Save parameters from the user program to the online DB.

9 Read in parameters from the online DB to the user program.

10 Export parameters from the parameter assignment interface to the DB (offline or online DB); an offline DB must then be downloaded to the CPU.

11 Import parameters from an online or offline DB into the parameter assignment interface.

Some use cases for the transfer of parameters

Use case	Steps
You edit the parameters on the parameter assignment interface. The parameter should then be assigned automatically to the module during startup.	Perform steps 1, 2, and 3.
You edit parameters in the parameter configuration interface during commissioning in test mode.	Perform steps 4 and 5
The parameters edited during commissioning should be downloaded automatically during startup.	Perform steps 1, 2, and 3.
Create the parameters using the parameter assignment interface. When it starts up, the module should be assigned parameters only by the user program using data blocks.	Perform steps 10 and 6.
You need a comfortable means of creating data records for recipes.	Perform step 10.
Create the parameters using the parameter assignment interface. These should be available to the user program for temporary modifications.	Perform steps 1, 2, and 3 for automatic parameter assignment.
	Perform steps 10, 7 to enable access by the user program.
You modify existing parameters (exclusively) using the user program.	Perform steps 7, 9, 8, and 6.
You want to view the data modified by the user program on the parameter assignment interface.	Perform step 11.
Parameters modified by the user program should also be loaded automatically during startup.	Perform steps 6, 11, 1, 2, and 3.

Commissioning the FM 452

Using FM 452 from the user program

Integrating the FM 452 into a user program

Instructions are available for integrating the FM 452 into a user program. These enable you to handle the desired functions with ease.

You call the CAM_INIT and CAM_CTRL or CAM_CTRL_452 instructions in the CPU user program. They ensure communication between the CPU and the FM 452.

The channel DB (Page 7164) is created automatically as the instance DB for the CAM_CTRL instruction when you insert the CAM_CTRL instruction into the project.
When you insert the CAM_CTRL_452 instruction into the project, use the command "Program blocks > Add new block" to add a CAM_CHANTYPE type data block to your project as the channel DB.

All the parameter data associated with the channel of the module (machine data, cam data) is located in the parameter DB (Page 7171) for the module. Use the command "Program blocks > Add new block" to add a CAM_PxxxTYPE type data block to your project. Depending on the quantity structure, you need the following DB types:

- Up to 16 cams: CAM_P016TYPE
- Up to 32 cams: CAM_P032TYPE
- Up to 64 cams: CAM_P064TYPE
- Up to 128 cams: CAM_P128TYPE

The instruction CAM_MSRM_452 allows you to evaluate data from FM 452 length measurement or edge detection immediately in the hardware interrupt OB.

The CAM_DIAG or CAM_DIAG_452 instruction is also available for FM 452. This instruction lets you transfer the diagnostic information to the diagnostic DB.

The diagnostic DB (Page 7174) is created automatically as the instance DB for the CAM_DIAG instruction when you insert the CAM_DIAG instruction into the project.
When you insert the CAM_DIAG_452 instruction into the project, use the command "Program blocks > Add new block" to add a CAM_DIAGTYPE type data block to your project as the diagnostic DB.

Commissioning the module

The module can be initially tested using the "Commissioning" dialog, even when no user program is available. Note the following warning in particular in this case.

WARNING

Injury and material damage could occur.

To prevent personal injury and material damage, observe the following points:

- Install an emergency stop switch in the vicinity of the computer.
  This is the only way you can ensure that the plant can be safely shut down in the event of computer or software failure.
- Install hardware limit switches, which act directly on the power units of all drives.
- Make sure that no one has access to the area of the plant in which moving parts are present.
- When the FM 452 is controlled and monitored simultaneously by your program and the "Commissioning" dialog, conflicts can occur, the effects of which are not clear. Therefore, always set the CPU to STOP mode or disable your program when you work with the "Commissioning" dialog.

See also

CAM_INIT (Page 7156)

CAM_CTRL (Page 7157)

CAM_DIAG (Page 7158)

CAM_CTRL_452 (Page 7160)

CAM_DIAG_452 (Page 7162)

CAM_MSRM_452 (Page 7163)

FM 452: Online & Diagnostics

Current module information

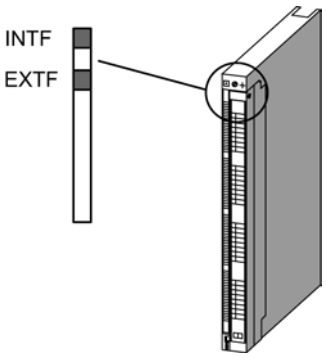
Default diagnostics

Error display through error LEDs of FM 452

Where is the error displayed?

The red error LEDs on the module display different error states. The LEDs are lit, even with errors that occur briefly, for at least 3 seconds.

The following figure shows the location of the error LEDs on FM 452.



Which errors are displayed?

The table below shows the error types indicated by the error LEDs.

LED	Meaning	Explanations
INTF	Group error for internal error	This LED indicates the following error states of FM 452: • Hardware interrupt lost • Watchdog expired • FM 452 not configured • FM 452 incorrectly configured To remove the error, see Error classes (Page 9503).
EXTF	Group error for external error	This LED indicates the following error states: • No external 24 V auxiliary supply • Front connector missing • Encoder wire break • Operating error • Absolute encoder frame error • Incremental encoder pulse missing or zero mark signal missing
INTF + EXTF	Module defective	All outputs are disabled. The module must be replaced.

Channel-specific diagnostics

Triggering diagnostic interrupts

Interrupt processing

The FM 452 can trigger hardware and diagnostic interrupts. You process these interrupts in an interrupt OB. If an interrupt is triggered without the corresponding OB being loaded, the CPU switches to STOP.

You enable the processing of diagnostic interrupts as follows:

1. Select the module in HWCN
2. Select "Edit" > Object Properties > Basic Parameters" to enable the diagnostic interrupt.
3. Save and compile the configuration.
4. Download the configuration to the CPU.

Overview of diagnostic interrupts

Events and errors which trigger a diagnostic interrupt:

- Operating error
- Machine data error

- Incorrect cam data
- Diagnostic errors

These errors are explained in detail in "Error classes (Page 9503)".

Module response to an error with diagnostic interrupt

- Cam processing will be disabled.
- The synchronization will be cleared by the following diagnostic interrupts:
  - Front connector missing, no external auxiliary voltage
  - Zero mark error detected, cable fault (5 V encoder signals)
  - Traversing range exceeded (indicated by an operating error)
  - Set process value cannot be executed (indicated by an operating error).
- With one exception, control signals are no longer processed.
  Exception:
  When the software limit switches are overrun, a change of direction is still possible in simulation mode.
- Function switch and job processing continues.

Module detects an error ("incoming")

A diagnostic interrupt is an "incoming" event if at least one error is pending. If any errors are pending clearance, the queued errors are reported again as "incoming" events.

Sequence:

1. The FM 452 detects one or more errors and triggers a diagnostic interrupt. The error LED lights up. The error event is logged to the diagnostic buffer.
2. The CPU operating system calls the diagnostic interrupt OB, e.g. OB 82.
3. You can now evaluate the start information of OB82.
4. The OB82_MOD_ADDR parameter indicates which module triggered the interrupt.
5. For additional information, call the CAM_DIAG (Page 7158) or CAM_DIAG_452 (Page 7162) instruction.

The module detects the transition to an error-free state ("outgoing")

A diagnostic interrupt is only registered "outgoing" if all errors on the module are cleared.

Sequence:

1. The FM 452 detects that all errors have been eliminated and triggers a diagnostic interrupt. The error LED goes out. The diagnostic buffer remains unchanged.
2. The CPU operating system calls the diagnostic interrupt OB, e.g. OB 82.

3. The OB82_MOD_ADDR parameter indicates which module triggered the interrupt.
4. Evaluate the OB82_MDL_DEFECT bit.
   When this bit is "0", no errors are present on the module. You can close the evaluation session at this point.

Diagnostic interrupt control by CPU states

- Diagnostic interrupts are blocked by FM 452 when the CPU is in STOP mode.
- If not all of the queued errors are eliminated while the CPU is in STOP mode, the FM 452 will report the remaining errors as "incoming" events again at the next CPU transition to RUN.
- If all queued errors are eliminated when the CPU is in STOP mode, the error-free state of the FM 452 will **not** be reported by a diagnostic interrupt after the transition to RUN mode.

Error evaluation in the user program

Fault reaction in the user program

The user program allows a targeted response to errors. The following options are available:

- Return values (RET_VAL) of the integrated instructions as a group display for errors that occurred during execution of the instructions.
- Error bits _ERR of the jobs as a group display for errors that occurred while a job was being processed.
- The DATA_ERR error bit as a group display for an error that the FM 452 has detected during a write job.
- The error identifier in JOB_ERR for the cause of an error in communication between instruction and FM 452.
- The CAM_DIAG (Page 7158) or CAM_DIAG_452 (Page 7162) instruction for reading the diagnostic buffer of the FM 452. This is where you find out the causes of errors in jobs and asynchronous events (operator errors, diagnostic errors).
- Diagnostic interrupts for fast reaction to events.

Diagnostics buffer of the module

Diagnostic events

The module´s diagnostic buffer contains a maximum of 4 diagnostic events and is organized as a ring buffer.

A diagnostic event is written in the buffer if an "incoming" (error) message is detected. This can be an alarm, a synchronous error (data error) or even an asynchronous error (operational errors and diagnostics errors). From a cause of error several entries can occur as follow-up errors. Outgoing messages do not create any entries in the diagnostic buffer.

For each diagnostic event the following is given:

- Status (always incoming)
- Internal error
- External error
- Error class
- Error number
- Channel number
- Cam number (for cam data errors)

When a diagnostic event is written in the diagnostic buffer, the checkback signal DIAG = 1 is set.

With the CAM_DIAG (Page 7158) or CAM_DIAG_452 (Page 7162) instruction, the diagnostic buffer can be transferred as a whole into a data block (diagnostic DB) or displayed via the error evaluation dialog of the configuration software. If the diagnostic buffer is read, the module sets the checkback signal DIAG = 0.

Note

If the diagnostic buffer is read simultaneously from the CAM_DIAG or CAM_DIAG_452 instruction and the error evaluation dialog, it is possible that a new diagnostic event is not detected by the program.

Data and Structure of the Diagnostic DB

Data and Structure of the Diagnostic DB

Note

Do not modify any data not listed in this table.

The following table shows the structure of the diagnostic DB:

Address	Name	Type	Initial value	Comment
0.0	MOD_ADDR (Enter!)	INT	0	Module address
256.0	JOB_ERR	INT	0	Communication errors
258.0	JOBBUSY	BOOL	FALSE	1 = job active
258.1	DIAGRD_EN	BOOL	FALSE	1 = read diagnostic buffer unconditional
260.0	DIAG_CNT	INT	0	Number of valid entries in the list
262.0	DIAG[1]	STRUCT		Diagnostic data latest entry
272.0	DIAG[2]	STRUCT		Diagnostic data second entry
282.0	DIAG[3]	STRUCT		Diagnostic data third entry
292.0	DIAG[4]	STRUCT		Diagnostic data oldest entry

The diagram below shows the structure of a DIAG[n] diagnostic entry.

Address	Name	Type	Initial value	Comment
+0.0	STATE	BOOL	FALSE	0 = outgoing event 1 = incoming event
+0.1	INTF	BOOL	FALSE	1 = Internal error
+0.2	EXTF	BOOL	FALSE	1 = external error
+2.0	FCL	INT	0	Error class: 1: Operating error 4: Data error 5: Machine data error 7: Cam data error 15: Messages 128: Diagnostic errors
+4.0	FNO	INT	0	Error number 0 to 255
+6.0	CH_NO	INT	0	Channel number (always 1)
+8.0	CAMNO	INT	0	Cam numbers 0 to 127 with error class = cam data error

List of JOB_ERR messages

JOB_ERR alarms

JOB_ERR (Hex)	JOB_ERR (Dec)	JOB_ERR (Int)	Meaning
80A0	32928	-32608	Negative acknowledgment when reading from module. Module removed during read operation or module defective.
80A1	32929	-32607	Negative acknowledgment when writing to module. Module removed during write operation or module defective.
80A2 [1)]	32930	-32606	Protocol error on Layer 2 (data transfer interrupted within PROFINET / PROFIBUS DP, for example, due to wire break, missing connector, parameter assignment error, etc.)
80A3 [1)]	32931	-32605	Protocol error at user interface/user (data transfer interrupted within PROFINET / PROFIBUS DP, for example, due to wire break, missing connector, parameter assignment error, etc.)
80A4 [1)]	32932	-32604	Communication problem on K bus
80B1	32945	-32591	Specified length wrong. FM_TYPE parameter in the channel DB is not correctly set for the module being used.
80B2	32946	-32590	The configured slot is empty.
80B3	32947	-32589	Actual module type does not match configured module type.
80C0 [1)]	32960	-32576	The module has not yet prepared the data to be read.
80C1 [1)]	32961	-32575	The data of a write job of the same type have not yet been processed on the module.
80C2 [1)]	32962	-32574	The module is currently processing the maximum number of jobs.
80C3 [1)]	32963	-32573	Required resources (memory etc.) currently in use.
80C4 [1)]	32964	-32572	Communication errors
80C5 [1)]	32965	-32571	Distributed I/Os not available.

JOB_ERR (Hex)	JOB_ERR (Dec)	JOB_ERR (Int)	Meaning
80C6 [1]	32966	-32570	Priority class abort (restart or background)
8522	34082	-31454	Channel DB or parameter DB too short. The data cannot be read from the DB. (write job)
8532	34098	-31438	DB number of the parameter DB too high (write job)
853A	34106	-31430	Parameter DB does not exist. (write job)
8544	34116	-31420	Error in nth (n > 1) read access to a DB after error occurred. (write job)
8723	34595	-30941	Channel DB or parameter DB too short. The data cannot be written to the DB. (read job)
8730	34608	-30928	Parameter DB on the CPU write-protected. The data cannot be written to the DB (read job).
8732	34610	-30926	DB number of the parameter DB too high (read job)
873A	34618	-30918	Parameter DB does not exist. (read job)
8745	34629	-30907	Error in nth (n > 1) write access to a DB after error occurred. (read job)

[1] Errors 80A2 to 80A4 and 80Cx are temporary, i.e., they can be rectified automatically after a waiting time. Messages in the form 7xxx indicate temporary states in communication.

Error classes

Operating error

Error class 1: Operating error

Operating errors are detected asynchronous to operator input/commands.

The following table shows the possible operating errors and their respective causes:

No.	Meaning		Diagnostic interrupt
1	**Software limit switch start overrun**		Yes
2	**Software limit switch end overrun**		Yes
3	**Traversing range start passed**		Yes
4	**Traversing range end passed**		Yes
13	**Set actual value on-the-fly cannot be executed**		Yes
	Cause	The software limit switches are outside the traversing range after the actual value on the fly is set (-100 m to +100 m or -1000 m to +1000 m). The shift resulting from set actual value / set actual value on-the-fly is more than ±100 m or ± 1000 m.	
	Effect	Axis not synchronized.	

Data error

Error class 4: Data error

Data errors are detected synchronous to operator input/commands.

The following table shows the possible data errors and their respective causes:

No.	Meaning		Diagnostic interrupt
10	**Zero offset error**		No
	Cause	The zero offset is greater than ±100 m or ±1000 m. The software limit switches are outside the traversing range after zero offset (-100 m to +100 m or -1000 m to +1000 m). Rotary axis: The zero offset amount is greater than the end of the rotary axis.	
11	**Faulty actual value setting**		No
	Cause	Linear axis: The coordinate lies outside the current (possibly shifted) software limit switches. Rotary axis: The coordinate is < 0 or greater than the end of rotary axis.	
12	**Reference point error**		No
	Cause	Linear axis: The coordinate lies outside the current (possibly shifted) software limit switches. Rotary axis: The coordinate is < 0 or greater than the end of rotary axis.	
20	**Enable machine data not permitted**		No
	Cause	There are no new (error-free) machine data on the module.	
21	**Set actual value on-the-fly not permitted**		No
	Cause	An attempt was made to execute "set actual value on-the-fly" while "retrigger reference point" was active.	
27	**Illegal bit-coded setting**		No
	Cause	Unused and, in this case, unwritten bits are not 0. An attempt was made to select "length measurement" and "edge detection" at the same time.	
28	**Retrigger reference point is not permitted**		No
	Cause	An attempt was made to execute "retrigger reference point" while "set actual value on-the-fly" was active. An attempt was made to execute "retrigger reference point" with an SSI encoder.	
29	**Illegal bit-coded command**		No
	Cause	Unused and, in this case, unwritten bits are not 0.	
30	**Incorrect lead time**		No
31	**Incorrect cam number**		No
	Cause	Cam is invalid. The cam number is not in the range 0 to 127.	
32	**Incorrect cam start**		No

No.	Meaning		Diagnostic interrupt
	Cause	The cam start is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m). Rotary axis: The cam start is < 0 or greater than the end of the rotary axis.	
33	Incorrect cam end / incorrect activation time		No
	Cause	The cam end is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m). Rotary axis: The cam end is < 0 or greater than the end of the rotary axis. The cam is not activated at least for the duration of one pulse. With an inverse cam, there are not at least 4 pulses between the cam start and cam end.	
34	Cancel set actual value not possible		No
	Cause	The actual position value would be outside the working range with an SSI encoder and linear axis after making the setting.	
35	Incorrect actual value specified by "set actual value" / "set actual value on-the-fly"		No
	Cause	The specified actual value is outside the permitted numeric range of ±100 m or ±1000 m. The software limit switches would be outside the traversing range (-100m to +100 m or -1000 m to +1000 m) were this setting to be applied. The offset derived from set actual value / set actual value on-the-fly would be more than ±100 m or ±1000 m.	
107	Parameters not set for the axis		No
	Cause	There are no machine data on the axis. There are no machine data activated on the axis.	
108	Axis not synchronized		No
	Cause	One of the settings "set actual value" or "set actual value on-the-fly" was initiated although the axis is not synchronized.	
109	Cam processing running.		No
110	Incorrect number of cams to be modified		No

Machine data error

Error class 5: Machine data error

The diagnostic interrupt is only triggered if an error is detected in the system data block (SDB).

The following table shows the possible machine data errors and their respective causes:

No.	Meaning		Diagnostic interrupt
5	Error in hardware interrupt setting		Yes
	Cause	You have attempted to select a hardware interrupt that the module does not support.	
6	Incorrect minimum edge distance		Yes
	Cause	You have entered a value < 0 or > 10^9 µm as the minimum edge-to-edge distance.	

No.	Meaning		Diagnostic interrupt
8	**Incorrect axis type**		Yes
	Cause	You have specified neither 0 nor 1 as the axis type	
9	**Incorrect end of rotary axis**		Yes
	Cause	The value for the end of the rotary axis is outside the valid range from 1 to 10^9 µm or 1 to 10^8 µm (depending on the resolution).	
10	**Incorrect encoder type**		Yes
	Cause	The value for the encoder type is outside the valid range of 1 to 10	
11	**Incorrect distance per encoder revolution**		Yes
	Cause	The value for distance/encoder revolution is outside the permitted range of 1 to 10^9 µm (regardless of the resolution).	
13	**Incorrect number of increments per encoder revolution** (see "Increments per encoder revolution (Page 9473)")		Yes
14	**Incorrect number of revolutions** (see "Number of encoder revolutions (Page 9478)")		Yes
15	**Incorrect transmission rate**		Yes
	Cause	You have specified a transmission rate outside the permitted range of 0 to 3.	
16	**Incorrect reference point coordinate**		Yes
	Cause	The coordinate is outside the range of -100 m to +100 m or -1000 m to +1000 m, depending on the resolution. Linear axis: The coordinate is outside the working range. Rotary axis: The coordinate is greater than the end of the rotary axis or < 0.	
17	**Incorrect absolute value adjustment**		Yes
	Cause	SSI encoder: The value of the absolute encoder adjustment is not in the encoder range (increments per encoder revolution x number of revolutions - 1).	
18	**Incorrect "Retrigger reference point" type**		Yes
	Cause	You have specified a value other than 0, 1, 6 and 7.	
19	**Incorrect direction adaptation**		Yes
	Cause	You have specified a value other than 0 and 1.	
20	**Hardware monitoring not possible**		Yes
	Cause	You have set frame error monitoring to "FALSE" in the parameter DB.	
21	**Incorrect software limit switch start**		Yes
	Cause	Linear axis: The software limit switch start is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m, depending on the resolution). Linear axis: The software limit switch start (including any existing zero offset) is less than -100 m or -1000 m (depending on the resolution).	
22	**Incorrect software limit switch end**		Yes
	Cause	Linear axis: The software limit switch end is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m, depending on the resolution) or is less than the software limit switch start. The software limit switch end (including any existing zero offset) is greater than +100 m or +1000 m (depending on the resolution).	
144	**Incorrect number of cams**		Yes
	Cause	You have specified a value other than 0 to 3 for the number of cams.	

No.	Meaning		Diagnostic interrupt
145	**Incorrect hysteresis**		Yes
	Cause	The hysteresis is outside the range 0 to 65535 × resolution. Hysteresis is greater than ¼ × operating range or ¼ × rotary axis range.	
146	**Incorrect simulation velocity**		Yes
	Cause	The simulation speed is outside the range 1000 × RES to 3 × 10^7 × RES, or is greater than 5 × 10^8 µm/min. The simulation velocity cannot be set internally.	
147	**Incorrect track**		Yes
	Cause	Control of a track outside 0 to 15 (bits 0 to 15) was selected.	
148	**Incorrect selection of enable inputs**		Yes
	Cause	You wanted to activate a track outside 3 to 10 (bits 0 to 7) using an external signal.	
149	**Incorrect selection of special tracks**		Yes
	Cause	You wanted to define a track outside 0, 1 and 2 (bits 0, 1 and 2) as a special track.	
150	**Incorrect high count value track 0**		Yes
	Cause	You have specified a count value < 2 or > 65535 as the high count value.	
151	**Incorrect high count value track 1**		Yes
	Cause	You have specified a count value < 2 or > 65535 as the high count value.	
200	**Incorrect resolution**		Yes
	Cause	You have specified a resolution < 0.1 µm/pulse or >1000 µm/pulse. You have specified a distance per encoder revolution and a number of pulses per encoder revolution that results in a resolution of < 0.1 or > 1000.	
201	**Position encoder does not match the working range/rotary axis range**		Yes
	Cause	SSI position encoder and rotary axis: The encoder does not exactly cover the rotary axis range. Linear axis: The position encoder does not cover at least the operating range (including software limit switches).	

Cam data error

Error class 7: Cam data error

The diagnostic interrupt is only triggered if an error is detected in the system data block (SDB).

The following table shows the possible cam data errors and their respective causes.

No.	Meaning		Diagnostic interrupt
1	**Illegal hardware interrupt**		Yes
	Cause	You attempted to define a process interrupt for a cam with a cam number > 7.	
2	**Incorrect track number**		Yes
	Cause	The track number is outside the range from 0 to 31.	

No.	Meaning		Diagnostic interrupt
3	**Incorrect cam start**		Yes
	Cause	The cam start is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m). Rotary axis: The cam start is < 0 or greater than the end of the rotary axis.	
4	**Incorrect cam end**		Yes
	Cause	The cam end is outside the traversing range (-100 m to +100 m or -1000 m to +1000 m). The cam is at least one pulse long. Rotary axis: The cam end is < 0 or greater than the end of the rotary axis. There are not at least 4 pulses between the cam start and cam end for an inverse cam.	
5	**Incorrect activation time**		Yes
	Cause	The activation time is < 0 µs. The maximum value (Page 9485) depends on the number of cams.	
6	**Incorrect lead time**[1]		Yes
	Cause	The lead time is < 0 µs. The maximum value (Page 9486) depends on the number of cams.	
50	**Too many cam sets**		Yes
	Cause	You wanted to enter more cam records than is possible with this number of cams.	
51	**Axis in operation**		Yes
	Cause	You wanted to enter cam records while the cam controller is active.	
52	**Parameters not set for the axis**		Yes
	Cause	You want to enter cam data although not machine data are active yet.	

[1] The error message can also be issued if you have assigned the parameter "inverted" as the count direction with an absolute encoder (SSI).

Alarms

Error class 15: Alarms

The following table shows the possible alarms and their respective causes:

No.	Meaning		Diagnostic interrupt
1	**Start of parameter assignment**		No
	Cause	The module has detected parameter assignment via a system data block.	
2	**End of parameter assignment**		No
	Cause	The module has processed the parameter assignment via a system data block error-free.	

Diagnostic errors

Error class 128: Diagnostic errors

The following table shows the possible diagnostic errors and their respective causes, effects and remedies.

No.	Meaning		Diagnostic interrupt
4	**External auxiliary voltage missing**		Yes
	Cause	External auxiliary 24 V voltage is not connected or has failed, front connector missing	
	Effect	See "Triggering diagnostic interrupts (Page 9498)": • Cam processing will be disabled • Track outputs are deactivated • With incremental encoders, synchronization is canceled • FM 452 has not been configured (checkback signal PARA = 0).	
	Correction	Make sure that the 24 V connection is correct (if the 24 V connection is correct, then the module is defective)	
5	**No front connector**		Yes
	Cause	Front connector not inserted	
	Effect	• No external 24 V auxiliary voltage • Module not ready for operation	
	Correction	Insert the front connector	
51	**Watchdog expired**		Yes
	Cause	• Strong interference on FM 452 • Error in the FM 452	
	Effect	• Module is reset • If, after resetting the module, no module fault is detected, the module is ready for operation again • The module reports the expired watchdog with "incoming" and "outgoing"	
	Correction	• Eliminate the interference • Contact the relevant sales department which will require details of the exact circumstances leading to the error • Replace the FM 452	
52	**Internal module power supply failed**		Yes
	Cause	Error in the FM 452	
	Effect	• Module is reset • If, after resetting the module, no module fault is detected, the module is ready for operation again	
	Correction	Replace the FM 452	
70	**Hardware interrupt lost**		Yes
	Cause	A hardware interrupt event detected by FM 452 cannot be reported because an identical event has not yet been processed by the user program/ CPU.	

No.	Meaning		Diagnostic interrupt
	Effect	• Cam processing will be disabled • Track outputs are deactivated • With incremental encoders, synchronization is canceled	
	Correction	• Integrate OB 40 into the user program • Check bus connection of the module • Deactivate hardware interrupt • Adapt your hardware and software to suit your process requirements (for example, faster CPU, optimize user program)	
144	**Encoder wire break**		Yes
	Cause	• Encoder cable cut or not plugged in • Encoder has no transverse signals • Incorrect pin assignment • Cable length too long • Encoder signals short-circuited	
	Effect	• Cam processing will be disabled • Track outputs are deactivated • With incremental encoders, synchronization is canceled	
	Correction	• Check the encoder cable • Keep within encoder specifications • Monitoring can be temporarily hidden at the responsibility of the operator using the parameter assignment dialog • Keep to the module's technical data	
145	**Absolute encoder frame error**		Yes
	Cause	The frame traffic between FM 452 and the absolute encoder (SSI) is incorrect or has been interrupted: • Encoder cable cut or not plugged in • Incorrect encoder type • Encoder incorrectly set (programmable encoders) • Frame length incorrectly specified • Encoder returns incorrect values (encoder defective) • Interference on measuring system cable • Transmission rate selected too high	
	Effect	• Cam processing will be disabled • Track outputs are deactivated • The last correct actual value remains unchanged till the end of the next correct SSI transfer	
	Correction	• Check the encoder cable • Check the encoder • Check the frame traffic between the encoder and FM 452	
146	**Missing pulses, incremental encoder**		Yes

No.	Meaning		Diagnostic interrupt
	Cause	• Encoder monitoring has detected missing pulses • Number of increments per encoder revolution is incorrectly entered • Encoder defective: does not return the defined number of pulses • Bad or missing zero marker • Interference on the encoder cable	
	Effect	• Cam processing will be disabled • Track outputs are deactivated • Synchronization is canceled	
	Correction	• Enter the correct number of increments/encoder revolution • Check the encoder and its cable • Comply with shielding and grounding regulations • Monitoring can be temporarily hidden at the responsibility of the operator using the parameter assignment dialog	

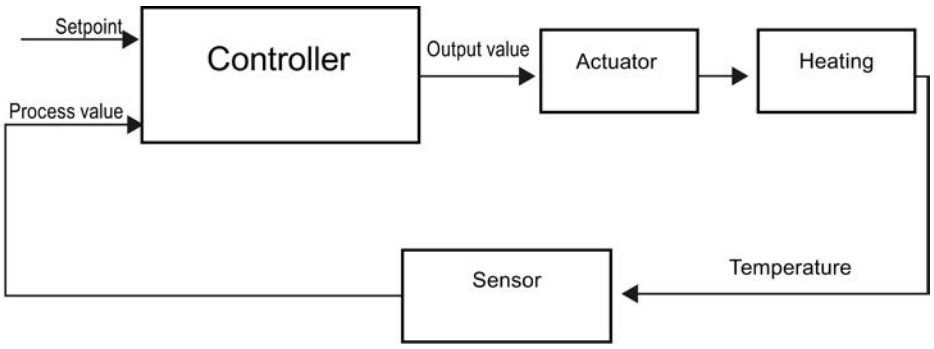
11.3 PID control

11.3.1 Principles for control

11.3.1.1 Controlled system and actuators

Controlled system

Room temperature control by means of a heating system is a simple example of a controlled system. A sensor measures the room temperature and transfers the value to a controller. The controller compares the current room temperature with a setpoint and calculates an output value (manipulated variable) for heating control.



A properly set PID controller reaches this setpoint as quickly as possible and then holds it a constant value. After a change in the output value, the process value often changes only with a time delay. The controller has to compensate for this response.

Actuators

The actuator is an element of the controlled system and is influenced by the controller. Its function modifies mass and energy flows.

The table below provides an overview of actuator applications.

Application	Actuator
Liquid and gaseous mass flow	Valve, shutter, gate valve
Solid mass flow, e.g., bulk material	Articulated baffle, conveyor, vibrator channel
Flow of electrical power	Switching contact, contactor, relay, thyristor
	Variable resistor, variable transformer, transistor

Actuators are distinguished as follows:

- Proportional actuators with constant actuating signal
  These elements set degrees of opening, angular positions or positions in proportion to the output value. The output value has an analog effect on the process within the control range. Actuators in this group include spring-loaded pneumatic drives, as well as motorized drives with position feedback for which a position control system is formed.
  An continuous controller, such as PID_Compact, generates the output value.
- Proportional actuators with pulse-width modulated signal
  These actuators are used to generate the output of pulses with a length proportional to the output value within the sampling time intervals. The actuator - e.g. a heating resistor or cooling apparatus - is switched on in isochronous mode for durations that differ depending on the output value.
  The actuating signal can assume unipolar "On" or "Off" states, or represent bipolar states such as "open/close", "forward/backward", "accelerate/brake".
  The output value is generated by a two-step controller such as PID_Compact with pulse-width modulation.
- Actuators with integral action and three-step actuating signal
  Actuators are frequently operated by motors with an on period that is proportional to the actuator travel of the choke element. This includes elements such as valves, shutters, and gate valves. In spite of their different design, all of these actuators follow the effect of an integral action at the input of the controlled system.
  A step controller, such as PID_3Step. generates the output value.

11.3.1.2 Controlled systems

The properties of a controlled system can hardly be influenced as these are determined by the technical requirements of the process and machinery. Acceptable control results can only be achieved by selecting a suitable controller type for the specific controlled system and adapting the controller to the time response of the controlled system. Therefore, it is is indispensable for the configuration of the proportional, integral and derivative actions of the controller to have precise knowledge of the type and parameters of the controlled system.

Controlled system types

Controlled systems are classified based on their time response to step changes of the output value.

We distinguish between the following controlled systems:

- Self-regulating controlled systems
  - Proportional-action controlled systems
  - PT1 controlled systems
  - PT2 controlled systems
- Non-self-regulating controlled systems
- Controlled systems with and without dead time

Self-regulating controlled systems

Proportional-action controlled systems

In proportional-action controlled systems, the process value follows the output value almost immediately. The ratio between the process value and output value is defined by the proportional Gain of the controlled system.

Examples:

- Gate valve in a piping system
- Voltage dividers
- Step-down function in hydraulic systems

PT1 controlled systems

In a PT1 controlled system, the process value initially changes in proportion to the change of the output value. The rate of change of the process value is reduced as a function of the time until the end value is reached, i.e., it is delayed.

Examples:

- Spring damping system
- Charge of RC elements
- Water container that is heated with steam.

The time constants are often identical for heating and cooling processes, or for charging and discharge characteristics. With different time constants, controlling is clearly more complex.

PT2 controlled systems

In a PT2 controlled system, the process value does not immediately follow a step change of the output value, i.e., it increases in proportion to the positive rate of rise and then approaches the setpoint at a decreasing rate of rise. The controlled system shows a proportional response characteristic with second order delay element.

Examples:

- Pressure control
- Flow rate control
- Temperature control

Non-self-regulating controlled systems

Non-self-regulating controlled systems have an integral response. The process value approaches an infinite maximum value.

Example:

- Liquid flow into a container

Controlled systems with dead time

A dead time always represents the runtime or transport time that has to expire before a change to the system input can be measured at the system output.

In controlled systems with dead time, the process value change is delayed by the amount of the dead time.

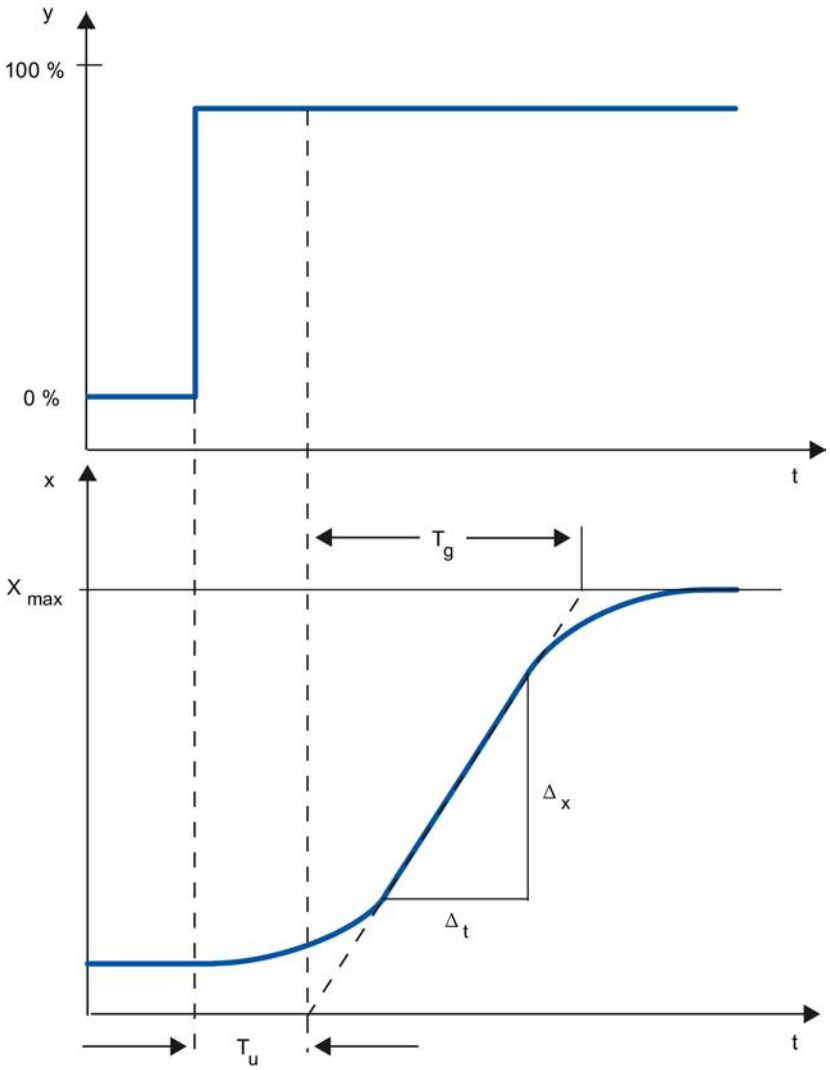
Example:

Conveyor

11.3.1.3 Characteristic values of the control section

Determining the time response from the step response

Time response of the controlled system can be determined based on the time characteristic of process value x following a step change of output value y. Most controlled systems are self-regulating controlled systems.



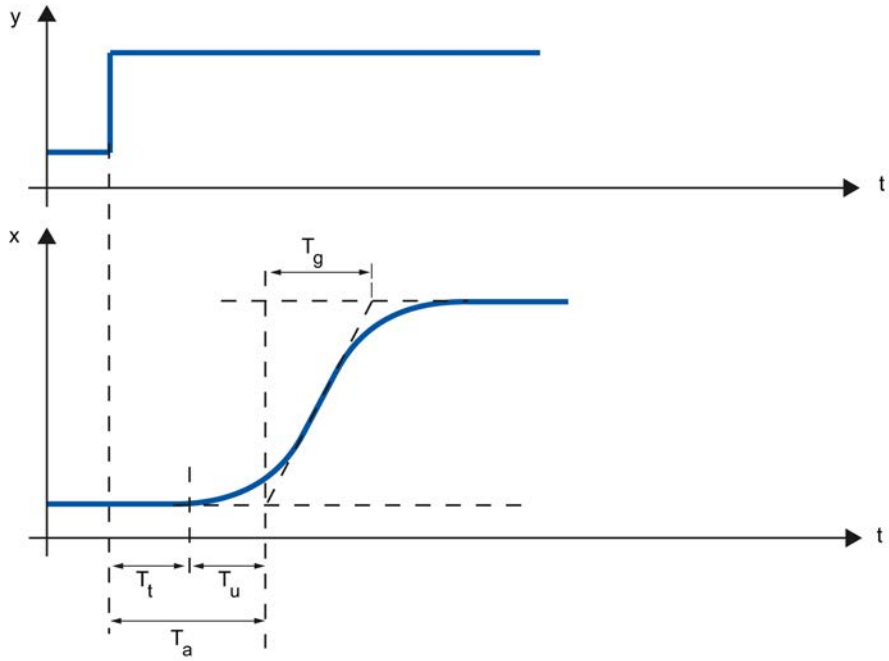
The time response can be determined by approximation using the variables Delay time T_u, Recovery time T_g and Maximum value X_{max}. The variables are determined by applying tangents to the maximum value and the inflection point of the step response. In many situations, it is not possible to record the response characteristic up to the maximum value because the process value cannot exceed specific values. In this case, the rate of rise v_{max} is used to identify the controlled system ($v_{max} = \Delta_x/\Delta_t$).

The controllability of the controlled system can be estimated based on the ratio T_u/T_g, or $T_u \times v_{max}/X_{max}$. Rule:

Process type	T_u / T_g	Suitability of the controlled system for controlling
I	< 0,1	can be controlled well
II	0.1 to 0.3	can still be controlled
III	> 0,3	difficult to control

Influence of the dead time on the controllability of a controlled system

A controlled system with dead time and recovery reacts as follows to a jump of the output value.



T_t	Dead time
T_u	Delay time
T_g	Recovery time
y	Output value
x	Process value

The controllability of a self-regulating controlled system with dead time is determined by the ratio of T_t to T_g. T_t must be small compared to T_g. Rule:

$T_t/T_g \leq 1$

Response rate of controlled systems

Controlled systems can be judged on the basis of the following values:

T_u < 0.5 min, T_g < 5 min = fast controlled system

T_u > 0.5 min, T_g > 5 min = slow controlled system

Parameters of certain controlled systems

Physical quantity	Controlled system	Delay time T_u	Recovery time T_g	Rate of rise v_{max}
Temperature	Small electrically heated furnace	0.5 to 1 min	5 to 15 min	Up to 60 K/min.
	Large electrically heated annealing furnace	1 to 5 min	10 to 20 min	Up to 20 K/min.
	Large gas-heated annealing furnace	0.2 to 5 min	3 to 60 min	1 to 30 K/min
	Distillation tower	1 to 7 min	40 to 60 min	0.1 to 0.5° C/s
	Autoclaves (2.5 m^3)	0.5 to 0.7 min	10 to 20 min	Not specified
	High-pressure autoclaves	12 to 15 min	200 to 300 min	Not specified
	Steam superheater	30 s to 2.5 min	1 to 4 min	2°C/s
	Injection molding machines	0.5 to 3 min	3 to 30 min	5 to 20 K/min
	Extruders	1 to 6 min	5 to 60 min	
	Packaging machines	0.5 to 4 min	3 to 40 min	2 to 35 K/min
	Room heating	1 to 5 min	10 to 60 min	1° C/min
Flow rate	Pipeline with gas	0 to 5 s	0.2 to 10 s	Not relevant
	Pipeline with liquid	None	None	
Pressure	Gas pipeline	None	0.1 s	Not relevant
	Drum boiler with gas or oil firing	None	150 s	Not relevant
	Drum boiler with impact grinding mills	1 to 2 min	2 to 5 min	Not relevant
Vessel level	Drum boiler	0.6 to 1 min	Not specified	0.1 to 0.3 cm/s
Speed	Small electric drive	None	0.2 to 10 s	Not relevant
	Large electric drive	None	5 to 40 s	Not relevant
	Steam turbine	None	Not specified	50 min^{-1}
Voltage	Small generators	None	1 to 5 s	Not relevant
	Large generators	None	5 to 10 s	Not relevant

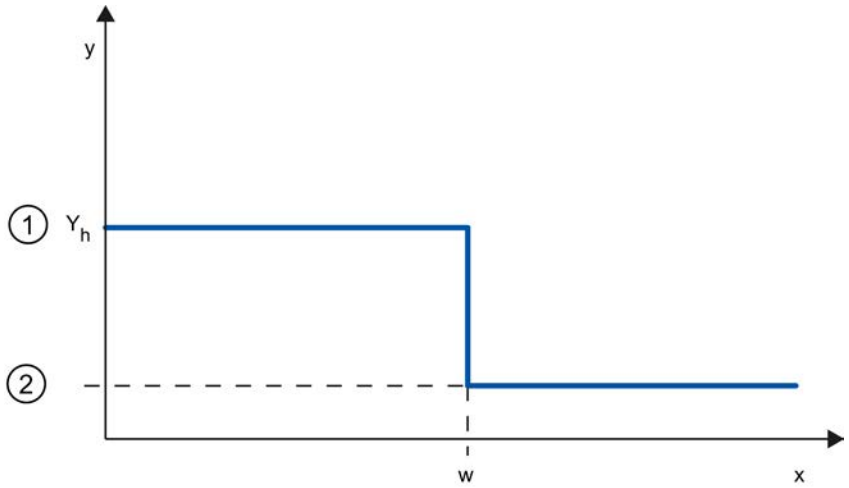
11.3.1.4 Pulse controller

Two-step controllers without feedback

Two-step controllers have the state "ON" and "OFF" as the switching function. This corresponds to 100% or 0% output. This behavior generates a sustained oscillation of process value x around setpoint w.

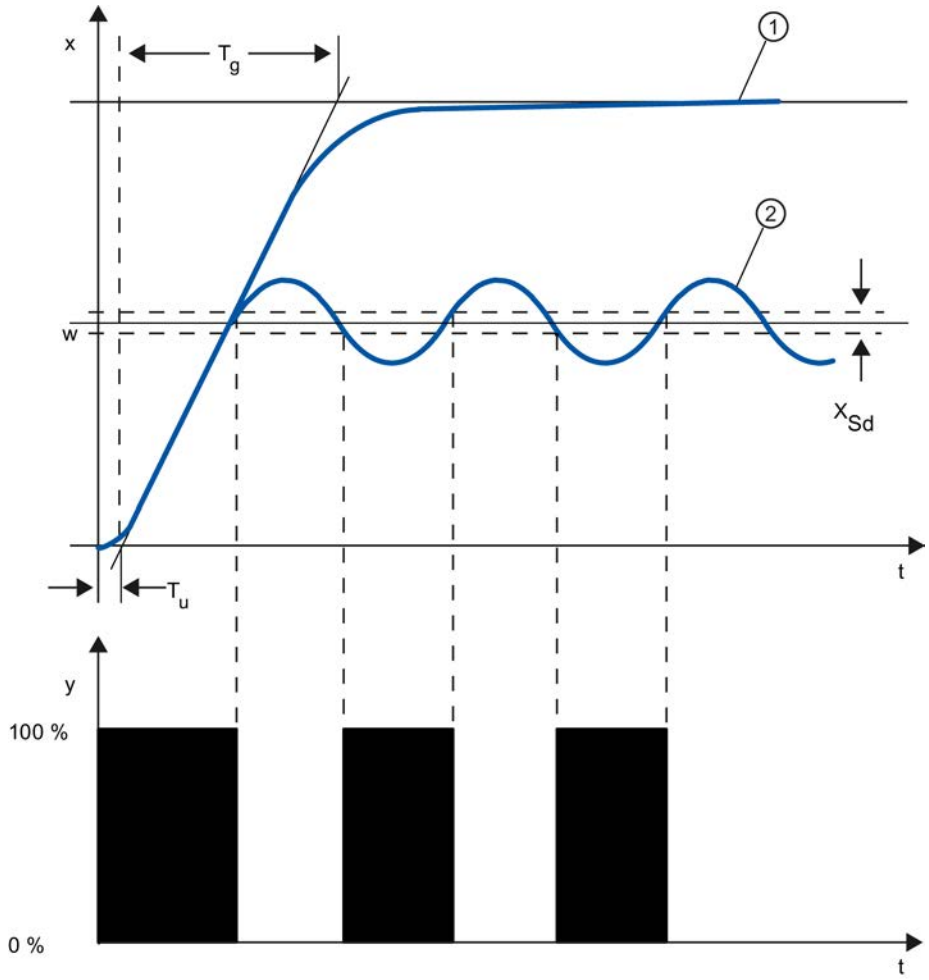
The amplitude and duration of the oscillation increase in proportion to the ratio between the delay time T_u and recovery time T_g of the controlled system. These controllers are used mainly for simple temperature control systems (such as electrically directly heated furnaces) or as limit-value signaling units.

The following diagram shows the characteristic of a two-step controller



①	ON
②	OFF
Y_h	Control range
w	Setpoint

The following diagram shows the control function of a two-step controller



①	Response characteristic without controller
②	Response characteristic with two-step controller
T_u	Delay time
T_g	Recovery time
X_{Sd}	Switching difference

Two-step controllers with feedback

The behavior of two-step controllers in the case of controlled systems with larger delay times, such as furnaces where the functional space is separated from the heating, can be improved by the use of electronic feedback.

The feedback is used to increase the switching frequency of the controller, which reduces the amplitude of the process value. In addition, the control-action results can be improved substantially in dynamic operation. The limit for the switching frequency is set by the output level. It should not exceed 1 to 5 switches per minute at mechanical actuators, such as relays and contactors. In the case of voltage and current outputs with downstream thyristor or Triac controllers high switching frequencies can be selected that exceed the limit frequency of the controlled system by far.

Since the switching pulses can no longer be determined at the output of the controlled system, results comparable with those of continuous controllers are obtained.

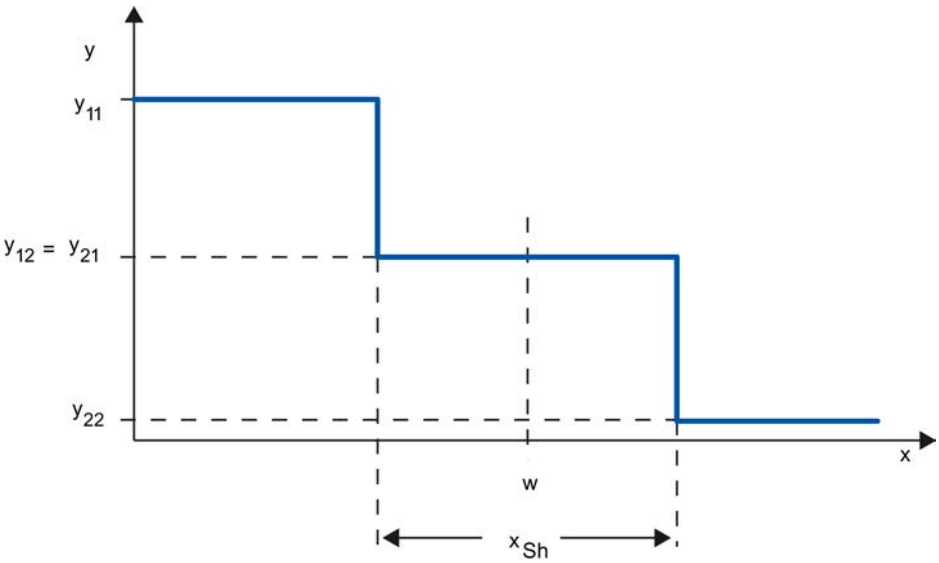
The output value is generated by pulse-width modulation of the output value of a continuous controller.

Two-step controllers with feedback are used for temperature control in furnaces, at processing machines in the plastics, textile, paper, rubber and foodstuff industries as well as for heating and cooling devices.

Three-step controllers

Three-step controllers are used for heating / cooling. These controllers have two switching points as their output. The control-action results are optimized through electronic feedback structures. Fields of applications for such controllers are heating, low-temperature, climatic chambers and tool heating units for plastic-processing machines.

The following diagram shows the characteristic of a three-step controller

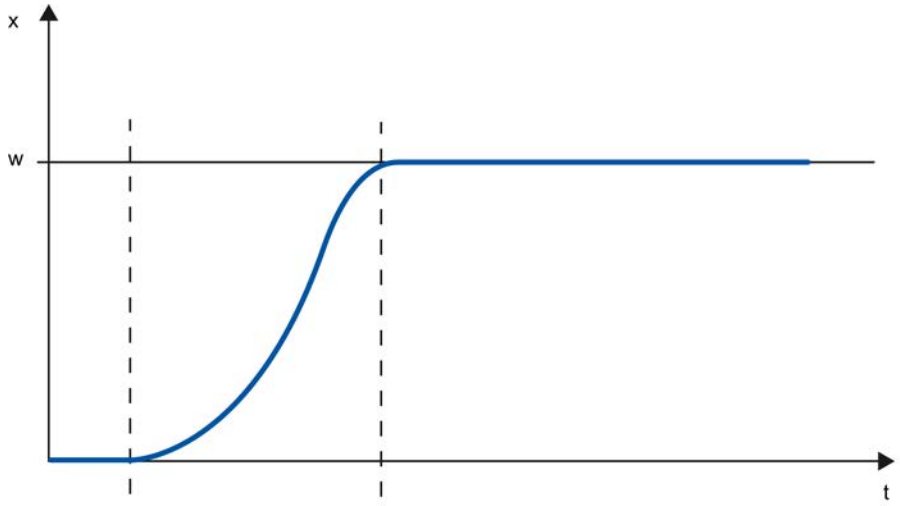


y	Output value, e.g. y11 = 100% heating y12 = 0% heating y21 = 0% cooling y22 = 100% cooling
x	Physical quantity of the process value, e.g., temperature in° C
w	Setpoint
x_{Sh}	Distance between Switching Point 1 and Switching Point 2

11.3.1.5 Response to setpoint changes and disturbances

Response to setpoint changes

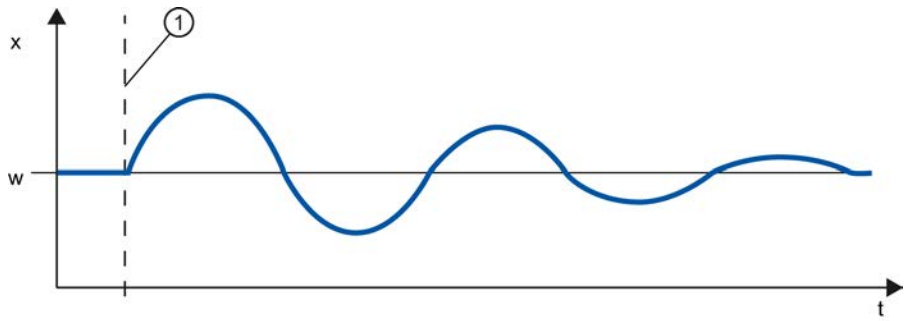
The process value should follow a setpoint change as quickly as possible. The response to setpoint changes is improved by minimizing fluctuation of the process value and the time required to reach the new setpoint.



x	Process value
w	Setpoint

Response to disturbances

The setpoint is influenced by disturbance variables. The controller has to eliminate the resulting control deviations in the shortest time possible. The response to disturbances is improved by minimizing fluctuation of the process value and the time required to reach the new setpoint.



x	Process value
w	Setpoint
①	Influencing a disturbance variable

Disturbance variables are corrected by a controller with integral action. A persistent disturbance variable does not reduce control quality because the control deviation is relatively constant. Dynamic disturbance variables have a more significant impact on control quality because of control deviation fluctuation. The control deviation is eliminated again only by means of the slow acting integral action.

A measurable disturbance variable can be included in the controlled system. This inclusion would significantly accelerated the response of the controller.

11.3.1.6 Control Response at Different Feedback Structures

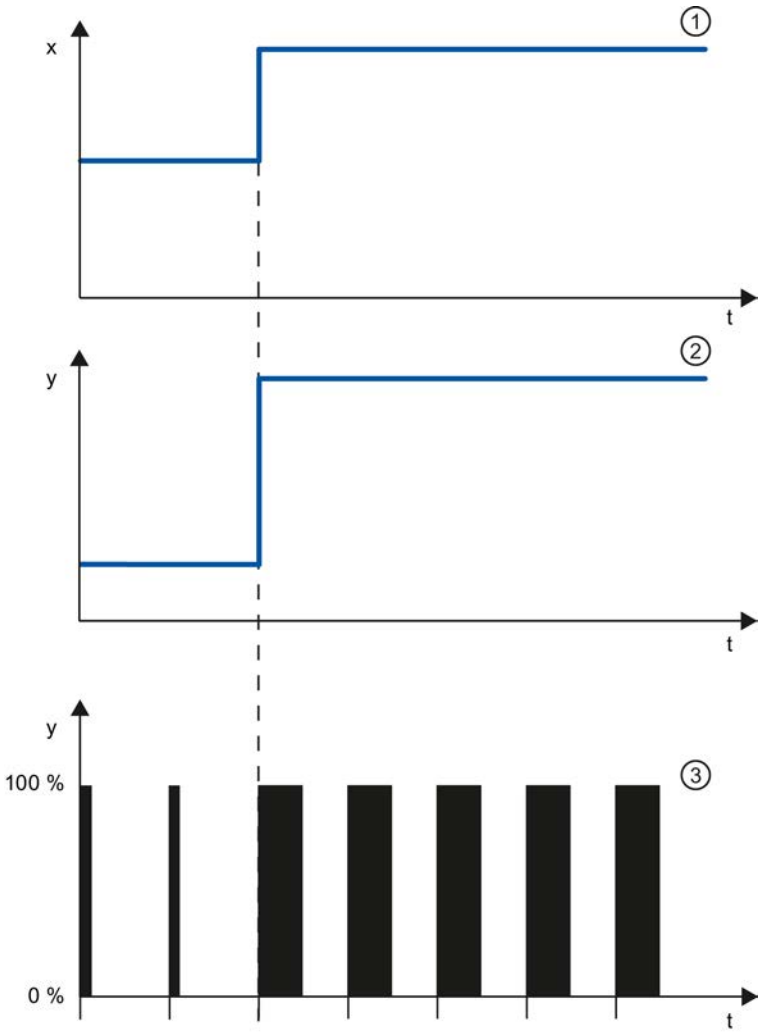
Control behavior of controllers

A precise adaptation of the controller to the time response of the controlled system is decisive for the controller's precise settling to the setpoint and optimum response to disturbance variables.

The feedback circuit can have a proportional action (P), proportional-derivative action (PD), proportional-integral action (PI), or proportional-integral-derivative action (PID).

If step functions are to be triggered by control deviations, the step responses of the controllers differ depending on their type.

Step response of a proportional action controller



① Control deviation
② Output value of a continuous controller
③ Output value of a pulse controller

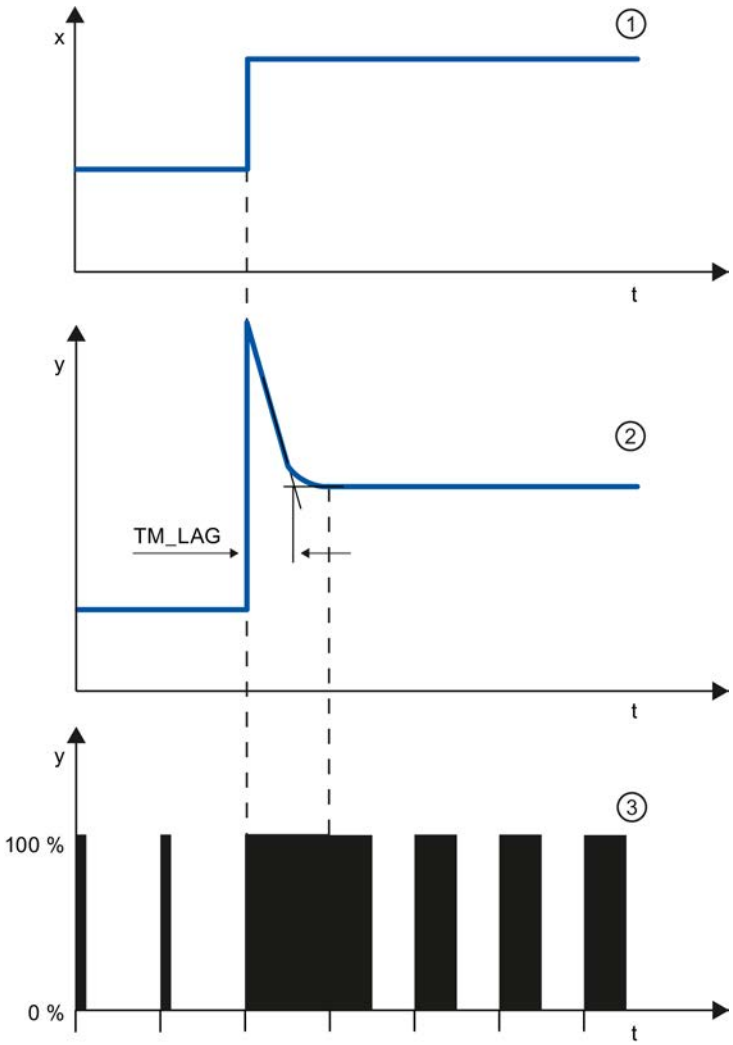
Equation for proportional action controller

Output value and control deviation are directly proportional, meaning:

Output value = proportional gain × control deviation

y = GAIN × x

Step response of a PD-action controller



①	Control deviation
②	Output value of a continuous controller
③	Output value of a pulse controller
TM_LAG	Delay of the Derivative action

Equation for PD-action controller

The following applies for the step response of the PD-action controller in the time range:

$$y = GAIN \cdot X_W \cdot \left(1 + \frac{TD}{TM_LAG} \cdot e^{\frac{-t}{TM_LAG}} \right)$$

t = time interval since the step of the control deviation

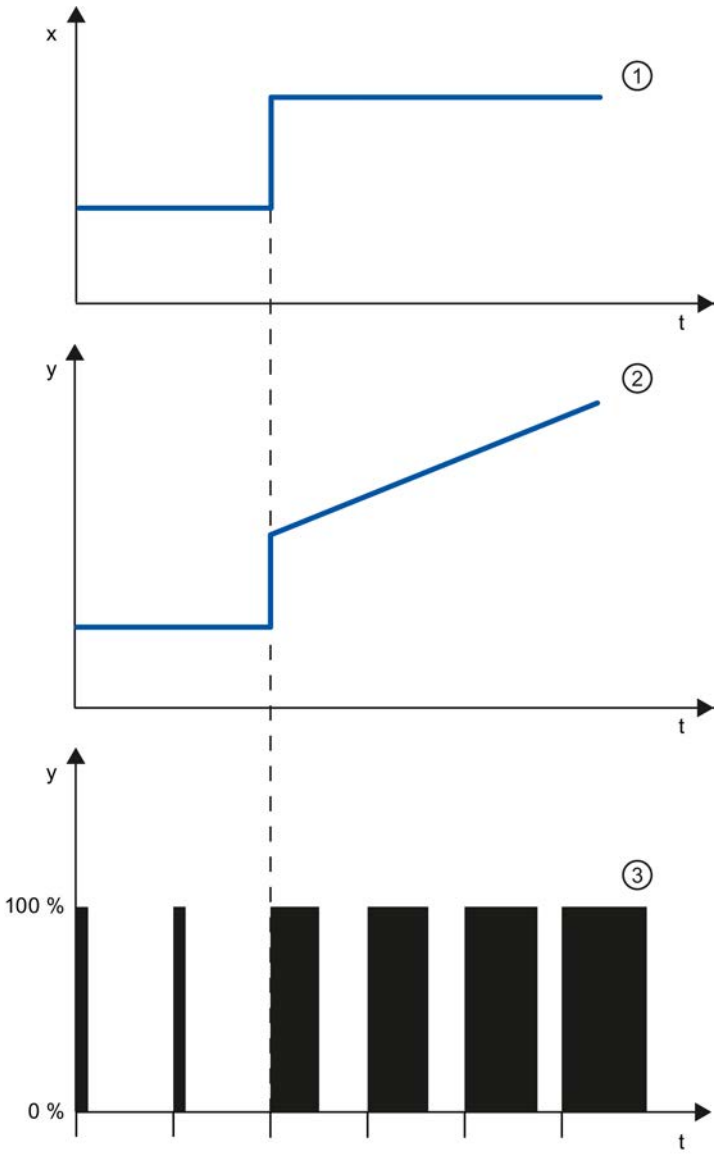
The derivative action generates a output value as a function of the rate of change of the process value. A derivative action by itself is not suitable for controlling because the output value only

follows a step of the process value. As long as the process value remains constant, the output value will no longer change.

The response to disturbances of the derivative action is improved in combination with a proportional action. Disturbances are not corrected completely. The good dynamic response is advantageous. A well attenuated, non-oscillating response is achieved during approach and setpoint change.

A controller with derivative action is not appropriate if a controlled system has pulsing measured quantities, for example, in the case of pressure or flow control systems.

Step response of a PI-action controller



① Control deviation

② Output value of a continuous controller

③ Output value of a pulse controller

An integral action in the controller adds the control deviation as a function of the time. This means that the controller corrects the system until the control deviation is eliminated. A sustained control deviation is generated at controllers with proportional action only. This effect can be eliminated by means of an integral action in the controller.

In practical experience, a combination of the proportional, integral and derivative actions is ideal, depending on the requirements placed on the control response. The time response of the individual components can be described by the controller parameters proportional gain GAIN, integral action time TI (integral action), and derivative action time TD (derivative action).

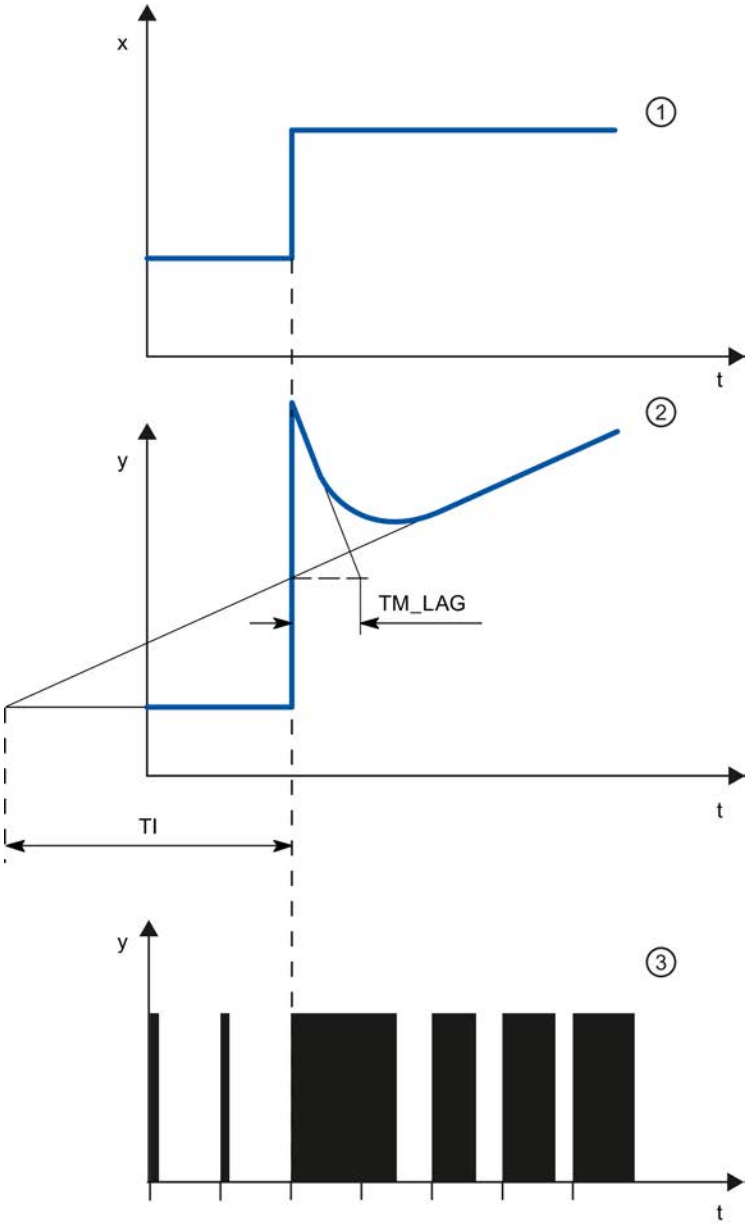
Equation for PI-action controller

The following applies for the step response of the PI-action controller in the time range:

$$y = GAIN \cdot X_W \cdot \left(1 + \frac{1}{TI \cdot t}\right)$$

t = time interval since the step of the control deviation

Step response of a PID controller



①	Control deviation
②	Output value of a continuous controller
③	Output value of a pulse controller
TM_LAG	Delay of the Derivative action
T_i	Integral action time

Equation for PID controller

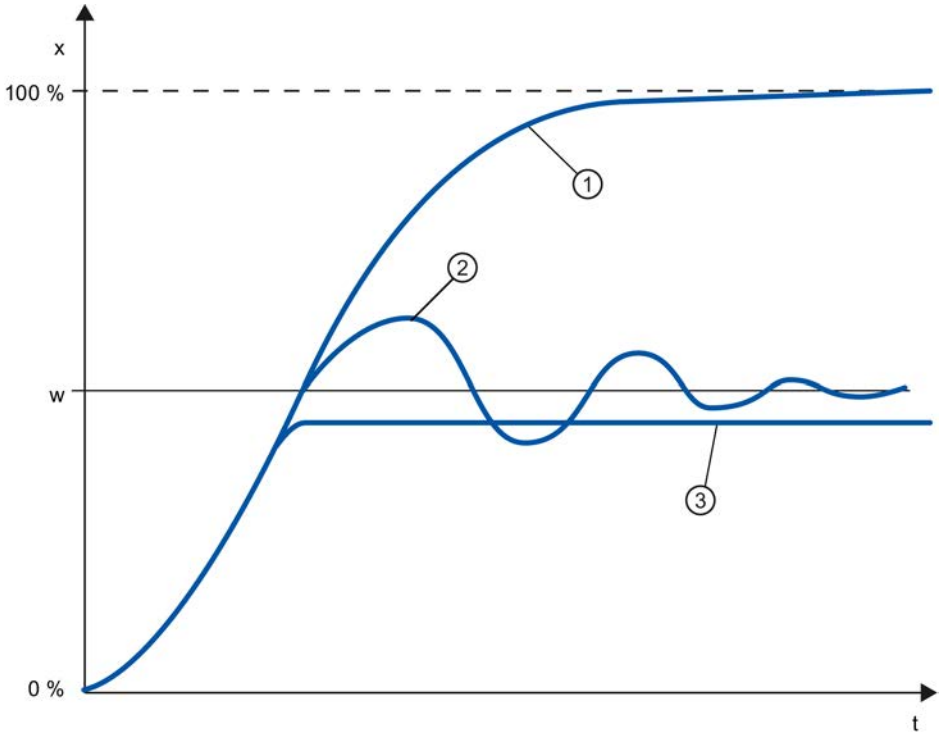
The following applies for the step response of the PID controller in the time range:

$$y = GAIN \cdot X_w \cdot \left(1 + \frac{1}{TI \cdot t} + \frac{TD}{TM_LAG} \cdot e^{\frac{-t}{TM_LAG}}\right)$$

t = time interval since the step of the control deviation

Response of a controlled system with different controller structures

Most of the controller systems occurring in process engineering can be controlled by means of a controller with PI-action response. In the case of slow controlled system with a large dead time, for example temperature control systems, the control result can be improved by means of a controller with PID action.



①	No controller
②	PID controller
③	PD-action controller
w	Setpoint
x	Process value

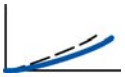
Controllers with PI and PID action have the advantage that the process value does not have any deviation from the setpoint value after settling. The process value oscillates over the setpoint during approach.

11.3.1.7 Selection of the controller structure for specified controlled systems

Selection of the Suitable Controller Structures

To achieve optimum control results, select a controller structure that is suitable for the controlled system and that you can adapt to the controlled system within specific limits.

The table below provides an overview of suitable combinations of a controller structure and controlled system.

Controlled system		Controller structure			
		P	PD	PI	PID
	With dead time only	Unsuitable	Unsuitable	Suitable	Unsuitable
	PT1 with dead time	Unsuitable	Unsuitable	Well suited	Well suited
	PT2 with dead time	Unsuitable	Suited conditionally	Well suited	Well suited
	Higher order	Unsuitable	Unsuitable	Suited conditionally	Well suited
	Not self-regulating	Well suited	Well suited	Well suited	Well suited

The table below provides an overview of suitable combinations of a controller structure and physical quantity.

Physical quantity	Controller structure			
	P	PD	PI	PID
	Sustained control deviation		No sustained control deviation	
Temperature	For low performance requirements and proportional action controlled systems with T_u/ Tg < 0,1	Well suited	The most suitable controller structures for high performance requirements (except for specially adapted special controllers)	
Pressure	Suitable, if the delay time is inconsiderable	Unsuitable	The most suitable controller structures for high performance requirements (except for specially adapted special controllers)	
Flow rate	Unsuitable, because required GAIN range is usually too large	Unsuitable	Suitable, but integral action controller alone often better	Hardly required

11.3.1.8 PID parameter settings

Rule of Thumb for the Parameter Setting

Controller structure	Setting
P	GAIN ≈ v_{max} × T_u [° C]
PI	GAIN ≈ 1.2 × v_{max} × T_u [° C]
PD	GAIN ≈ 0.83 × v_{max} × T_u [° C] TD ≈ 0.25 × v_{max} × T_u [min] TM_LAG ≈ 0.5 × TD[min]
PID	GAIN ≈ 0.83 × v_{max} × T_u [° C] TI ≈ 2 × T_u [min] TD ≈ 0.4 × T_u [min] TM_LAG ≈ 0.5 × TD[min]
PD/PID	GAIN ≈ 0.4 × v_{max} × T_u [° C] TI ≈ 2 × T_u [min] TD ≈ 0.4 × T_u [min] TM_LAG ≈ 0.5 × TD[min]

Instead of $v_{max} = \Delta_x / \Delta_t$, you can use X_{max} / T_g.

In the case of controllers with PID structure the setting of the integral action time and differential-action time is usually coupled with each other.

The ratio TI / TD lies between 4 and 5 and is optimal for most controlled systems.

Non-observance of the differential-action time TD is uncritical at PD controllers.

In the case of PI and PID controllers, control oscillations occur if the integral action time TI has been select by more than half too small.

An integral action time that is too large slows down the settling times of disturbances. One cannot expect that the control loops operate "optimally" after the first parameter settings. Experience shows that adjusting is always necessary, when a system exists that is "difficult to control" with $T_u / T_g > 0.3$.

11.3.2 Configuring a software controller

11.3.2.1 Overview of software controller

For the configuration of a software controller, you need an instruction with the control algorithm and a technology object. The technology object for a software controller corresponds with the instance DB of the instruction. The configuration of the controller is saved in the technology object. In contrast to the instance DBs of other instructions, technology objects are not stored for the program resources, but rather under CPU > Technology objects.

Technology objects and instructions

CPU	Library	Instruction	Technology object	Description
S7-1200	Compact PID	PID_Compact V1.X	PID_Compact V1.X	Universal PID controller with integrated tuning
S7-1200		PID_3Step V1.X	PID_3Step V1.X	PID controller with integrated tuning for valves
S7-1500 S7-1200 V4.x		PID_Compact V2.X	PID_Compact V2.X	Universal PID controller with integrated tuning
S7-1500 S7-1200 V4.x		PID_3Step V2.X	PID_3Step V2.X	PID controller with integrated tuning for valves
S7-1500/300/400	PID basic functions	CONT_C	CONT_C	Continuous controller
S7-1500/300/400		CONT_S	CONT_S	Step controller for actuators with integrating behavior
S7-1500/300/400		PULSEGEN	-	Pulse generator for actuators with proportional behavior
S7-1500/300/400		TCONT_CP	TCONT_CP	Continuous temperature controller with pulse generator
S7-1500/300/400		TCONT_S	TCONT_S	Temperature controller for actuators with integrating behavior
S7-300/400	PID Self Tuner	TUN_EC	TUN_EC	Optimization of a continuous controller
S7-300/400		TUN_ES	TUN_ES	Optimization of a step controller
S7-300/400	Standard PID Control (PID Professional optional package)	PID_CP	PID_CP	Continuous controller with pulse generator
S7-300/400		PID_ES	PID_ES	Step controller for actuators with integrating behavior
S7-300/400		LP_SCHED	-	Distribute controller calls
S7-300/400	Modular PID Control (PID Professional optional package)	A_DEAD_B	-	Filter interfering signal from control deviation
S7-300/400		CRP_IN	-	Scale analog input signal
S7-300/400		CRP_OUT	-	Scale analog output signal
S7-300/400		DEAD_T	-	Delay output of input signal
S7-300/400		DEADBAND	-	Suppress small fluctuations to the process value
S7-300/400		DIF	-	Differentiate input signals over time
S7-300/400		ERR_MON		Monitor control deviation
S7-300/400		INTEG	-	Integrate input signals over time
S7-300/400		LAG1ST	-	First-order delay element
S7-300/400		LAG2ND	-	Second-order delay element
S7-300/400		LIMALARM	-	Report limit values
S7-300/400		LIMITER	-	Limiting the manipulated variable
S7-300/400		LMNGEN_C	-	Determine manipulated variable for continuous controller
S7-300/400		LMNGEN_S	-	Determine manipulated variable for step controller
S7-300/400		NONLIN	-	Linearize encoder signal

CPU	Library	Instruction	Technology object	Description
S7-300/400		NORM	-	Scale process value physically
S7-300/400		OVERRIDE	-	Switch manipulated variable from 2 PID controllers to 1 actuator
S7-300/400		PARA_CTL	-	Switch parameter sets
S7-300/400		PID	-	PID algorithm
S7-300/400		PUSLEGEN_M	-	Generate pulse for proportional actuators
S7-300/400		RMP_SOAK	-	Specify setpoint according to ramp / soak
S7-300/400		ROC_LIM	-	Limit rate of change
S7-300/400		SCALE_M	-	Scale process value
S7-300/400		SP_GEN	-	Specify setpoint manually
S7-300/400		SPLT_RAN	-	Split manipulated variable range
S7-300/400		SWITCH	-	Switch analog values
S7-300/400		LP_SCHED_M	-	Distribute controller calls

11.3.2.2 Steps for the configuration of a software controller

All SW-controllers are configured according to the same scheme:

Step	Description
1	Add technology object (Page 9533)
2	Configure technology object (Page 9534)
3	Call instruction in the user program (Page 9535)
4	Download technology object to device (Page 9536)
5	Commission software controller (Page 9537)
6	Save optimized PID parameters in the project (Page 9538)
7	Comparing values (Page 9539)
8	Display instances of a technology object (Page 9558)

11.3.2.3 Add technology objects

Add technology object in the project navigator

When a technology object is added, an instance DB is created for the instruction of this technology object. The configuration of the technology object is stored in this instance DB.

Requirement

A project with a CPU has been created.

Procedure

To add a technology object, proceed as follows:

1. Open the CPU folder in the project tree.
2. Open the "Technology objects" folder.
3. Double-click "Add new object".
   The "Add new object" dialog box opens.
4. Click on the "PID" button.
   All available PID-controllers for this CPU are displayed.
5. Select the instruction for the technology object, for example, PID_Compact.
6. Enter an individual name for the technology object in the "Name" input field.
7. Select the "Manual" option if you want to change the suggested data block number of the instance DB.
8. Click "Further information" if you want to add own information to the technology object.
9. Confirm with "OK".

Result

The new technology object has been created and stored in the project tree in the "Technology objects" folder. The technology object is used if the instruction for this technology object is called in a cyclic interrupt OB.

Note

You can select the "Add new and open" check box at the bottom of the dialog box. This opens the configuration of the technology object after adding has been completed.

11.3.2.4 Configure technology objects

The properties of a technology object on a S7-1200 CPU can be configured in two ways.

- In the Inspector window of the programming editor
- In the configuration editor

The properties of a technology object on a S7-300/400 CPU can only be configured in the configuration editor.

Inspector window of the programming editor

In the Inspector window of the programming editor you can only configure the parameters required for operation.

The offline values of the parameters are also shown in online mode. You can only change the online values in the commissioning window.

To open the Inspector window of the technology object, follow these steps:

1. Open the "Program blocks" folder in the project tree.
2. Double click the block (cyclic interrupt OB) in which you open the instruction of the SW-controller.
   The block is opened in the work area.
3. Click on the instruction of the SW-controller.
4. In the Inspector window, select the "Properties" and "Configuration" tabs consecutively.

Configuration window

For each technology object, there is a specific configuration window in which you can configure all properties.

To open the configuration window of the technology object, follow these steps:

1. Open the "Technology objects" folder in the project tree.
2. Open the technology object in the project tree.
3. Double-click the "Configuration" object.

Symbols

Icons in the area navigation of the configuration and in the Inspector window show additional details about the completeness of the configuration:

	The configuration contains default values and is complete. The configuration exclusively contains default values. With these default values the use of the technology object is possible without further changes.
	The configuration contains values defined by the user and is complete All input fields of the configuration contain valid values and at least one default setting was changed.
	The configuration is incomplete or faulty At least one input field or a collapsible list contains no or one invalid value. The corresponding field or the drop-down list box has a red background. When clicked the roll-out error message indicates the cause of the error.

The properties of a technology object are described in detail in the chapter for the technology object.

11.3.2.5 Call instruction in the user program

The instruction of the software controller must be called in a cyclic interrupt OB. The sampling time of the software controller is determined by the interval between the calls in the cyclic interrupt OB.

Requirement

The cyclic interrupt OB is created and the cycle time of the cyclic interrupt OB is correctly configured.

Procedure

Proceed as follows to call the instruction in the user program:

1. Open the CPU folder in the project tree.
2. Open the "Program blocks" folder.
3. Double-click the cyclic interrupt OB.
   The block is opened in the work area.
4. Open the "Technology" group in the "Instructions" window and the "PID Control" folder.
   The folder contains all instructions for software controllers that can be configured on the CPU.
5. Select the instruction and drag it to your cyclic interrupt OB.
   The "Call options" dialog box opens.
6. Select a technology object or type the name for a new technology object from the "Name" list.

Result

If the technology object does not exist yet, it is added. The instruction is added in the cyclic interrupt OB. The technology object is assigned to this call of the instruction.

11.3.2.6 Downloading technology objects to device

A new or modified configuration of the technology object must be downloaded to the CPU for the online mode. The following characteristics apply when downloading retentive data:

- Software (changes only)
  - S7-1200, S7-1500:
    Retentive data is retained.
  - S7-300/400:
    Retentive data is updated immediately. CPU does not change to Stop.
- Download PLC program to device and reset
  - S7-1200, S7-1500:
    Retentive data is updated at the next change from Stop to RUN. The PLC program can only be downloaded completely.
  - S7-300/400:
    Retentive data is updated at the next change from Stop to RUN.

Downloading retentive data to an S7-1200 or S7-1500 CPU

Note

The download and reset of the PLC program during ongoing system operation can result in serious damages or injuries in the case of malfunctions or program errors.

Make sure that dangerous states cannot occur before you download and reset the PLC program.

Proceed as follows to download the retentive data:

1. Select the entry of the CPU in the project tree.
2. Select the command "Download and reset PLC program" from the "Online" menu.
   - If you have not established an online connection yet, the "Extended download" dialog opens. In this case, set all required parameters for the connection and click "Download".
   - If the online connection has been defined, the project data is compiled, if necessary, and the dialog "Load preview" opens. This dialog displays messages and recommends actions necessary for download.
3. Check the messages.
   As soon as download is possible, the "Download" button becomes active.
4. Click on "Download".
   The complete PLC program is downloaded and the "Load results" dialog opens. This dialog displays the status and the actions after the download.
5. If the modules are to restart immediately after the download, select the check box "Start all".
6. Close the dialog "Download results" with "Finish".

Result

The complete PLC program is downloaded to the device. Blocks that only exist online in the device are deleted. By downloading all affected blocks and by deleting any blocks in the device that are not required, you avoid inconsistencies between the blocks in the user program.

The messages under "Info > General" in the Inspector window indicate whether the download was successful.

11.3.2.7 Commissioning software controller

Procedure

To open the "Commissioning" work area of the technology object, follow these steps:

1. Open the "Technology objects" folder in the project tree.
2. Open the technology object in the project tree.
3. Double-click the "Commissioning" object.

The commissioning functions are specific for each controller and are described there.

11.3.2.8 Save optimized PID parameter in the project

The software controller is optimized in the CPU. Through this, the values in the instance-DB on the CPU no longer agree with those in the project.

To update the PID parameter in the project with the optimized PID parameters, proceed as follows:

Requirement

- An online connection to the CPU is established and the CPU is in "RUN" mode.
- The functions of the commissioning window have been enabled by means of the "Start" button.

Procedure

1. Open the CPU folder in the project tree.
2. Open the "Technology objects" folder.
3. Open a technology object.
4. Double click on "Commissioning".
5. Click on the icon "Upload PID parameters".
6. Save the project.

Result

The currently active PID parameters are stored in the project data. When reloading the project data in the CPU, the optimized parameters are used.

11.3.2.9 Comparing values

Comparison display and boundary conditions

The "Compare values" function provides the following options:

- Comparison of configured start values of the project with the start values in the CPU and the actual values
- Direct editing of actual values and the start values of the project
- Immediate detection and display of input errors with suggested corrections
- Backup of actual values in the project
- Transfer of start values of the project to the CPU as actual values

Icons and operator controls

The following icons and operator controls are available:

Icon	Function
	Start value PLC matches the configured Start value project
	Start value PLC does not match the configured Start value project
	The comparison of the Start value PLC with the configured Start value project cannot be performed
	At least one of the two comparison values has a process-related or syntax error.
	Transfers actual values to the offline project
	Transfers updated start values in the project to the CPU (initialize setting values)
	Opens the "Compare values" dialog

Boundary conditions

The "Compare values" function is available for S7-1200 and S7-1500 without limitations.

The following limitation applies to S7-300 and S7-400:

In monitoring mode, an S7-300/S7-400 cannot transfer the start values to the CPU. These values cannot be displayed online with "Compare values".

The actual values of the technology object are displayed and can be changed directly.

Comparing values

The procedure is shown in the following using "PID Parameters" as an example.

Requirements

- A project with a software controller is configured.
- The project is downloaded to the CPU.
- The configuration dialog is open in the project navigator.

Procedure

1. Open the desired software controller in the project navigation.
2. Double-click the "Configuration" object.
3. Navigate within the configuration window to the "PID Parameters" dialog.
4. Click the icon to activate monitoring mode.
   The icons and operator controls (Page 9538) of the "Compare values" function are shown behind the parameters.

5. Click the desired parameter in the input box and change the parameter values manually by entering them directly.
   - If the background of the input box is gray, this value is a read-only value and cannot be changed.
   - To change the values in the "PID Parameters" dialog, enable manual entry by selecting the "Enable manual entry" check box beforehand.
6. Click the icon to open the dialog for the start values.
   This dialog indicates two values of the parameter:
   - Start value in CPU: The start value in the CPU is shown in the top part.
   - Start value in the project: The configured start value in the project is shown in the bottom part.
7. Enter the desired value in the input box for the project.

Error detection

The input of incorrect values is detected. Corrections are suggested in this case.

If you enter a value with incorrect syntax, a rollout containing the corresponding error message opens below the parameter. The incorrect value is not applied.

If you enter a value that is incorrect for the process, a dialog opens containing the error message and a suggested correction:

- Click "No" to accept this suggested correction and correct your input.
- Click "OK" to apply the incorrect value.

NOTICE
Malfunctions of the controller Values incorrect for the process can result in controller malfunctions.

Backing up actual values

Click the icon to transfer the actual controller values to the start values of your configured project.

Transferring project values to the CPU

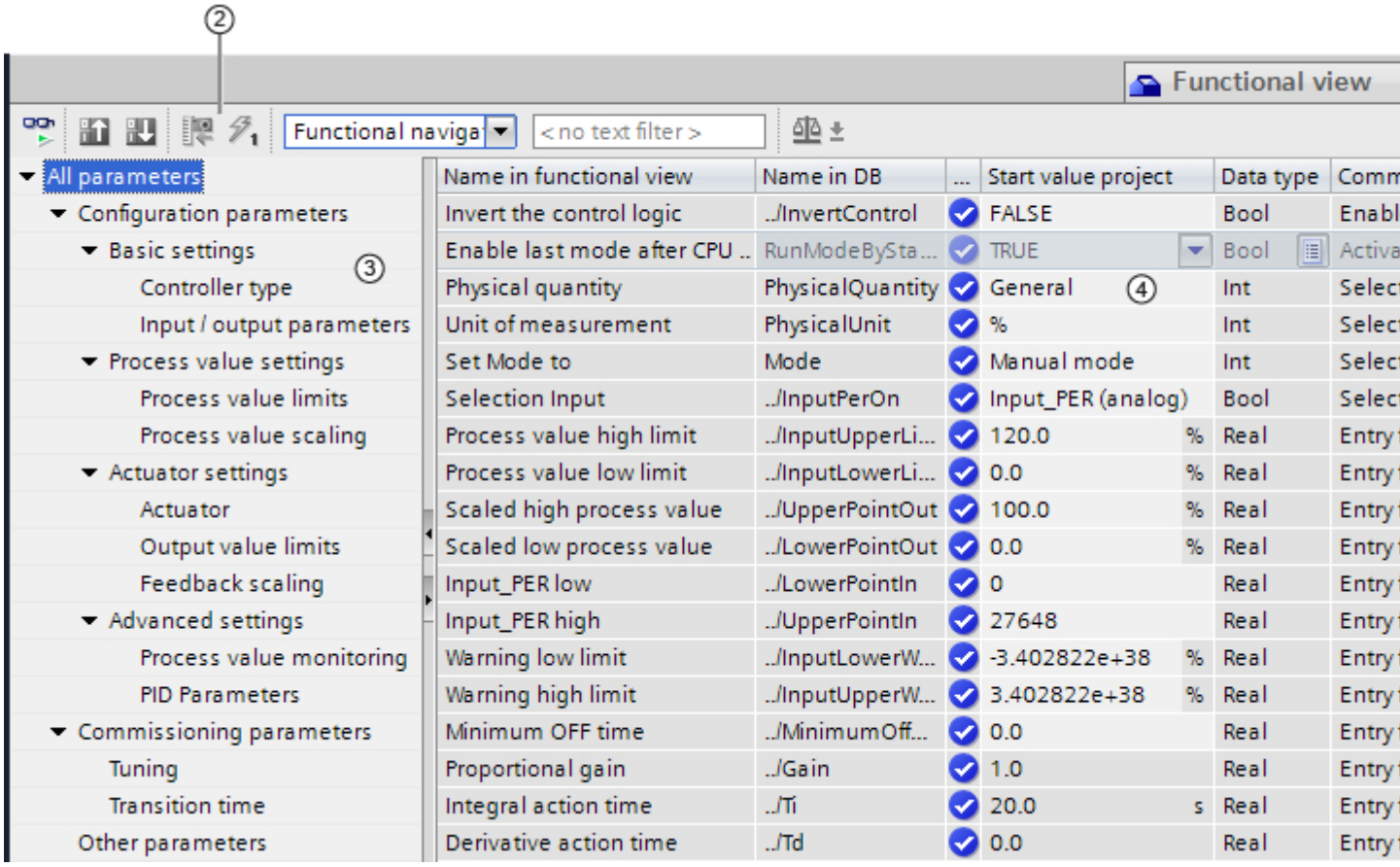
Click the icon to transfer the configured values of your project to the CPU.

> **CAUTION**
>
> **Prevent personal injury and property damage!**
>
> Downloading and resetting of the user program while the plant is operating may result in significant property damage and severe personal injuries in the event of malfunctions or program errors.
>
> Make sure that dangerous states cannot occur before you download and reset the user program.

11.3.2.10 Parameter view

Introduction to the parameter view

The Parameter view provides you with a general overview of all relevant parameters of a technology object. You obtain an overview of the parameter settings and can easily change them in offline and online mode.



① "Parameter view" tab

② Toolbar (Page 9543)

③ Navigation (Page 9544)

④ Parameter table (Page 9544)

Function scope

The following functions are available for analyzing the parameters of the technology objects and for enabling targeted monitoring and modification.

Display functions:

- Display of parameter values in offline and online mode
- Display of status information of the parameters
- Display of value deviations and option for direct correction
- Display of configuration errors
- Display of value changes as a result of parameter dependencies
- Display of all memory values of a parameter: Start value PLC, Start value project, Monitor value
- Display of the parameter comparison of the memory values of a parameter

Operator control functions:

- Navigation for quickly changing between the parameters and parameter structures.
- Text filter for faster searches for particular parameters.
- Sorting function for customizing the order of parameters and parameter groups to requirements.
- Memory function for backing up structural settings of the Parameter view.
- Monitoring and modifying of parameter values online.
- Function for saving a snapshot of parameter values of the CPU in order to capture momentary situations and to respond to them.
- Function for applying a snapshot of parameter values as start values.
- Download of modified start values to the CPU.
- Comparison functions for comparing parameter values with one another.

Validity

The Parameter view is available for the following technology objects:

- PID_Compact
- PID_3Step
- CONT_C (S7-1500 only)
- CONT_S (S7-1500 only)

- TCONT_CP (S7-1500 only)
- TCONT_S (S7-1500 only)
- TO_Axis_PTO (S7-1200 Motion Control)
- TO_Positioning_Axis (S7-1200 Motion Control)
- TO_CommandTable_PTO (S7-1200 Motion Control)
- TO_CommandTable (S7-1200 Motion Control)

Structure of the parameter view

Toolbar

The following functions can be selected in the toolbar of the parameter view.

Icon	Function	Explanation
	Monitor all	Starts the monitoring of visible parameters in the active Parameter view (online mode).
	Create snapshot of monitor values and accept setpoints of this snapshot as start values	Applies the current monitor values to the “Snapshot” column and updates the start values in the project. Only in online mode for PID_Compact and PID_3Step.
	Initialize setpoints	Transfers the start values updated in the project to the CPU. Only in online mode for PID_Compact and PID_3Step.
	Create snapshot of monitor values	Applies the current monitor values to the “Snapshot” column. Only in online mode.
	Modify all selected parameters immediately and once	This command is executed once and as quickly as possible without reference to any particular point in the user program. Only in online mode.
Functional naviga	Select navigation structure	Toggles between functional navigation and data navigation.
< no text filter >	Text filter...	After entry of a character string: Display of all parameters containing the specified string in one of the currently visible columns.
	Selection of compare values	Selection of parameter values that are to be compared with one another in online mode (Start value project, Start value PLC, Snapshot) Only in online mode.
	Save window settings	Saves your display settings for the Parameter view (e.g., selected navigation structure, activated table columns, etc.)

Navigation

Within the "Parameter view" tab, the following alternative navigation structures can be selected.

Navigation		Explanation
Functional navigation	All parameters Configuration parameters Commissioning parameters Other parameters	In the functional navigation, the structure of the parameters is based on the structure in the configuration dialog ("Functional view" tab), commissioning dialog, and diagnostics dialog. The last group "Other parameters" contains all other parameters of the technology object.
Data navigation	All parameters Input Output InOut Static Other parameters	In the data navigation, the structure of the parameters is based on the structure in the instance DB. The last group "Other parameters" contains the parameters that are not contained in the instance DB.

You can use the "Select navigation structure" drop-down list to toggle the navigation structure.

Parameter table

The table below shows the meaning of the individual columns of the parameter table. You can show or hide the columns as required.

- Column "Offline" = X: Column is visible in offline mode.
- Column "Online" = X: Column is visible in online mode (online connection to the CPU).

Column	Explanation	Offline	Online
Name in functional view	Name of the parameter in the functional view. The display field is empty for parameters that are not configured via the technology object.	X	X
Full name in DB	Complete path of the parameter in the instance DB. The display field is empty for parameters that are not contained in the instance DB.	X	X
Name in DB	Name of the parameter in the instance DB. If the parameter is part of a structure or UDT, the prefix ". ./" is added. The display field is empty for parameters that are not contained in the instance DB.	X	X
Status of configuration	Display of the completeness of the configuration using status symbols. see Status of configuration (offline) (Page 9552)	X	
Compare result	Result of the "Compare values" function. This column is shown if there is an online connection and the "Monitor all" button is selected. see Comparing values (Page 9538)		X

Column	Explanation	Offline	Online
Start value project	Configured start value in the project. Error indication if entered values have a syntax or process-related error.	X	X
Default value	Value that is pre-assigned to the parameter. The display field is empty for parameters that are not contained in the instance DB.	X	X
Snapshot	Snapshot of the current values in the CPU (monitor values). Error indication if values have a process-related error.	X	X
Start value PLC	Start value in the CPU. This column is shown if there is an online connection and the "Monitor all" button is selected. Error indication if values have a process-related error.		X
Monitor value	Current value in the CPU. This column is shown if there is an online connection and the "Monitor all" button is selected. Error indication if values have a process-related error.		X
Modify value	Value that is to be used to change the monitor valuet. This column is shown if there is an online connection and the "Monitor all" button is selected. Error indication if entered values have a syntax or process-related error.		X
Selection for transmission	Selection of the Modify values that are to be transmitted using the "Modify all selected parameters immediately and once" button. This column is displayed together with the "Modify value" column.		X
Minimum value	Minimum process-related value of the parameter. If the minimum value is dependent on other parameters, it is defined: • Offline: By the Start value project. • Online: By the Monitor values.	X	X
Maximum value	Maximum process-related value of the parameter. If the maximum value is dependent on other parameters, it is defined: • Offline: By the Start value project. • Online: By the Monitor values.	X	X
Setpoint	Designates the parameter as a setpoint. These parameters can be initialized online.	X	X
Data type	Data type of the parameter. The display field is empty for parameters that are not contained in the instance DB.	X	X
Retain	Designates the value as a retentive value. The values of retentive parameters are retained even after the voltage supply is switched off.	X	X
Accessible from HMI	Indicates whether the HMI can access this parameter during runtime.	X	X
Visible in HMI	Indicates whether the parameter is visible in the selection list of the HMI by default.	X	X
Comment	Brief description of the parameter.	X	X

Opening the parameter view

Requirement

The technology object has been added in the project tree, i.e., the associated instance DB of the instruction has been created.

Procedure

1. Open the "Technology objects" folder in the project tree.
2. Open the technology object in the project tree.
3. Double-click the "Configuration" object.
4. Select the "Parameter view" tab in the top right corner.

Result

The Parameter view opens. Each displayed parameter is represented by one row in the parameter table.

The displayable parameter properties (table columns) vary depending on whether you are working with the Parameter view in offline or online mode.

In addition, you can selectively display and hide individual table columns.

See also

Default setting of the parameter view (Page 9546)

Default setting of the parameter view

Default settings

To enable you to work efficiently with the Parameter view, you can customize the parameter display and save your settings.

The following customizations are possible and can be saved:

- Show and hide columns
- Change column width
- Change order of the columns
- Toggle navigation
- Select parameter group in the navigation
- Selection of compare values

Show and hide columns

To show or hide columns in the parameter table, follow these steps:

1. Position the cursor in the header of the parameter table.
2. Select the "Show/Hide" command in the shortcut menu.
   The selection of available columns is displayed.
3. To show a column, select the check box for the column.
4. To hide a column, clear the check box for the column.

or

1. Position the cursor in the header of the parameter table.
2. Select the "Show all columns" command in the shortcut menu if all columns of the offline or online mode are to be displayed.

Some columns can only be displayed in online mode: see Parameter table (Page 9544).

Change column width

To customize the width of a column so that all texts in the rows can be read, follow these steps:

1. Position the cursor in the header of the parameter table to the right of the column to be customized until the shape of the cursor changes to a cross.
2. Then double-click this location.

or

1. Open the shortcut menu on the header of the parameter table.
2. Click
   - "Optimize column width" or
   - "Optimize width of all columns".

If the column width setting is too narrow, the complete content of individual fields are shown if you hover the cursor briefly over the relevant field.

Change order of the columns

The columns of the parameter table can be arranged in any way.

To change the order of the columns, follow these steps:

1. Click on the column header and use a drag-and-drop operation to move it to the desired location.
   When you release the mouse button, the column is anchored to the new position.

Toggle navigation

To toggle the display form of the parameters, follow these steps:

1. Select the desired navigation in the “Select navigation structure” drop-down list.
   - Data navigation
   - Functional navigation

See also Navigation (Page 9544).

Select parameter group in the navigation

Within the selected navigation, you choose between the “All parameters” display or the display of a subordinate parameter group of your choice.

1. Click the desired parameter group in the navigation.
   The parameter table only displays the parameters of the parameter group.

Selection of compare values (online)

To set the compare values for the “Compare values” function, follow these steps:

1. Select the desired compare values in the “Selection of compare values” drop-down list.
   - Start value project / Start value PLC
   - Start value project / Snapshot
   - Start value PLC / Snapshot

The “Start value project / Start value PLC” option is set by default.

Save default setting of the Parameter view

To save the above customizations of the Parameter view, follow these steps:

1. Customize the Parameter view according to your requirements.
2. Click the “Save window settings” button at the top right of the Parameter view.

Working with the parameter view

Overview

The following table provides an overview of the functions of the Parameter view in online and offline mode described in the following.

- Column "Offline" = X: This function is possible in offline mode.
- Column "Online" = X: This function is possible in online mode.

Function/action	Offline	Online
Filtering the parameter table (Page 9549)	X	X
Sorting the parameter table (Page 9550)	X	X
Transferring parameter data to other editors (Page 9550)	X	X
Indicating errors (Page 9550)	X	X
Editing start values in the project (Page 9551)	X	X
Status of configuration (offline) (Page 9552)	X	
Monitoring values online in the parameter view (Page 9553)		X
Create snapshot of monitor values (Page 9553)		X
Modifying values (Page 9554)		X
Comparing values (Page 9555)		X
Applying values from the online program as start values (Page 9556)		X
Initializing setpoints in the online program (Page 9557)		X

Filtering the parameter table

You can filter the parameters in the parameter table in the following ways:

- With the text filter
- With the subgroups of the navigation

Both filter methods can be used simultaneously.

With the text filter

Texts that are visible in the parameter table can be filtered. This means only texts in displayed parameter rows and columns can be filtered.

1. Enter the desired character string for filtering in the “Text filter...” input box.
   The parameter table displays only the parameters containing the character string.

The text filtering is reset.

- When another parameter group is selected in the navigation.
- When navigation is changed from data navigation to functional navigation, or vice versa.

With the subgroups of the navigation

1. Click the desired parameter group in the navigation, e.g., "Static".
   The parameter table only shows the static parameters. You can select further subgroups for some groups of the navigation.
2. Click “All parameters” in the navigation if all parameters are to be shown again.

Sorting the parameter table

The values of the parameters are arranged in rows. The parameter table can be sorted by any displayed column.

- In columns containing numerical values, sorting is based on the magnitude of the numerical value.
- In text columns, sorting is alphabetical.

Sorting by column

1. Position the cursor in the header cell of the desired column.
   The background of this cell turns blue.
2. Click the column header.

Result

The entire parameter table is sorted by the selected column. A triangle with tip facing up appears in the column header.

Clicking the column header again changes the sorting as follows:

- Symbol "▲": Parameter table is sorted in ascending order.
- Symbol "▼": Parameter table is sorted in descending order.
- No symbol: The sorting is removed again. The parameter table assumes the default display.

The "../" prefix in the "Name in DB" column is ignored when sorting.

Transferring parameter data to other editors

After selecting an entire parameter row of the parameter table, you can use the following:

- Drag-and-drop
- <Ctrl+C>/<Ctrl+V>
- Copy/Paste via shortcut menu

Transfer parameters to the following editors of the TIA Portal:

- Program editor
- Watch table
- Signal table for trace function

The parameter is inserted with its full name: See information in "Full name in DB" column.

Indicating errors

Error indication

Parameter assignment errors that result in compilation errors (e.g., limit violation) are indicated in the Parameter view.

Every time a value is input in the Parameter view, a check is made for process-related and syntax errors and the result is indicated.

Bad values are indicated by:

- Red error symbol in the "Status of configuration" (offline mode) or "Compare result" (online mode, depending on the selected comparison type) columns

and/or

- Table field with red background
  If you click the bad field, a roll-out error message appears with information of the permissible value range or the required syntax (format)

Compilation error

From the error message of the compiler, you can directly open the Parameter view (functional navigation) containing the parameter causing the error in situations where the parameter is not displayed in the configuration dialog.

Editing start values in the project

With the Parameter view, you can edit the start values in the project in offline mode and online mode.

- You make value changes in the “Start value project” column of the parameter table.
- In the “Status of configuration” column of the parameter table, the progress of the configuration is indicated by the familiar status symbols from the configuration dialog of the technology object.

Boundary conditions

- If other parameters depend on the parameter whose start value was changed, the start value of the dependent parameters are also adapted.
- If a parameter of a technology object is not editable, it is also not editable in the parameter view. The ability to edit a parameter can also depend on the values of other parameters.

Defining new start values

To define start values for parameters in the Parameter view, follow these steps:

1. Open the Parameter view of the technology object.
2. Enter the desired start values in the "Start value project" column. The value must match the data type of the parameter and must not exceed the value range of the parameter. The limits of the value range can be seen in the “Maximum value” and “Minimum value” columns.

The "Status of configuration" column indicates the progress of the configuration with colored symbols.

See also Status of configuration (offline) (Page 9552)

Following adaptation of the start values and downloading of the technology object to the CPU, the parameters take the defined value at startup if they are not declared as retentive (“Retain” column).

Error indication

When a start value is input, a check is made for process-related and syntax errors and the result is indicated.

Bad start values are indicated by:

- Red error symbol in the "Status of configuration" (offline mode) or "Compare result" (online mode, depending on the selected comparison type) columns

and/or

- Red background in the “Start value project” field
  If you click on the bad field, a roll-out error message appears with information of the permissible value range or the necessary syntax (format)

Correcting bad start values

1. Correct bad start values using information from the roll-out error message.
   Red error symbol, red field background, and roll-out error message are no longer displayed.

The project cannot be successfully compiled unless the start values are error-free.

Status of configuration (offline)

The status of the configuration is indicated by icons:

- In the “Status of configuration” column in the parameter table
- In the navigation structure of the functional navigation and data navigation

Symbol in “Status of configuration” column

Symbol	Meaning
	The start value of the parameter corresponds to the default value and is valid. A start value has not yet been defined by the user.
	The start value of the parameter contains a value defined by the user. The start value is different than the default value. The start value is error-free and valid.
	The start value of the parameter is invalid (syntax or process-related error). The input box has a red background. When clicked, the roll-out error message indicates the cause of the error.
	Only for S7-1200 Motion Control: The start value of the parameter is valid but contains warnings. The input box has a yellow background.

Symbol in the navigation

The symbols in the navigation indicate the progress of the configuration in the same way as in the configuration dialog of the technology object.

See Configure technology objects (Page 9534)

Monitoring values online in the parameter view

You can monitor the values currently taken by the parameters of the technology object in the CPU (monitor values) directly in the Parameter view.

Requirements

- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").
- The Parameter view of the technology object is open.

Procedure

1. Start the monitoring by clicking .
   As soon as the Parameter view is online, the following columns are additionally displayed:
   - Compare result
   - Start value PLC
   - Monitor value
   - Modify value
   - Selection for transmission

   The "Monitor value" column shows the current parameter values on the CPU.
   Meaning of the additional columns: see Parameter table (Page 9544)
2. Stop the monitoring by clicking again.

Display

All columns that are only available online have an orange background:

- Values in light-orange cells can be changed.
- Values in cells with a dark orange background cannot be changed.

Create snapshot of monitor values

You can back up the current values of the technology object on the CPU (monitor values) and display them in the Parameter view.

Requirements

- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").
- The Parameter view of the technology object is open.
- The “Monitor all” button is selected.

Procedure

To show the current parameter values, follow these steps:

1. In the Parameter view, click the “Create snapshot of monitor values" icon .

Result

The current monitor values are transferred once to the "Snapshot" column of the parameter table.

You can analyze the values "frozen" in this way while the monitor values continue to be updated in the "Monitor values" column.

Modifying values

With the Parameter view, you can modify values of the technology object in the CPU.

You can assign values to the parameter once (Modify value) and modify them immediately. The modify request is executed as quickly as possible without reference to any particular point in the user program.

DANGER

Danger when modifying:

Changing the parameter values while the plant is operating may result in severe damage to property and personal injury in the event of malfunctions or program errors.

Make sure that dangerous states cannot occur before you use the "Modify" function.

Requirements

- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").
- The Parameter view of the technology object is open.

- The “Monitor all” button is selected.
- The parameter can be modified (associated field in the "Modify value" column has a light-orange background).

Procedure

To modify parameters immediately, follow these steps:

1. Enter the desired modify values in the “Modify values” column of the parameter table.
2. Check whether the check box for modifying is selected in the "Select for transmission" column.
   The modify values and associated check boxes of dependent parameters are automatically adapted at the same time.
3. Click the “Modify all selected parameters immediately and once” icon .

The selected parameters are modified once and immediately with the specified values and can be monitored in the "Modify values" column. The check boxes for modifying in the "Selection for transmission" column are automatically cleared after the modify request is complete.

Error indication

When a start value is input, a check is made immediately for process-related and syntax errors and the result is indicated.

Bad start values are indicated by:

- Red background in the “Modify value” field

and

- If you click the bad field, a roll-out error message appears with information of the permissible value range or the necessary syntax (format)

Bad modify values

- Modify values with process-related errors can be transmitted.
- Modify values with syntax errors **cannot** be transmitted.

Comparing values

You can use comparison functions to compare the following memory values of a parameter:

- Start value project
- Start value PLC
- Snapshot

Requirements

- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").
- The Parameter view of the technology object is open.
- The “Monitor all” button is selected.

Procedure

To compare the start values on the various target systems, follow these steps:

1. Click the "Selection of compare values" icon .
   A selection list containing the comparison options opens:
   – Start value project - Start value PLC (default setting)
   – Start value project - Snapshot
   – Start value PLC - Snapshot
2. Select the desired comparison option.
   The selected comparison option is executed as follows:
   – A scales symbol appears in the header cells of the two columns selected for comparison.
   – Symbols are used in the "Compare result" column to indicate the result of the comparison of the selected columns.

Symbol in "Compare result" column

Symbol	Meaning
	The compare values are equal and error-free.
	The compare values are not equal and error-free.
	At least one of the two compare values has a process-related or syntax error.
	The comparison cannot be performed. At least one of the two compare values is not available (e.g., snapshot).

Symbol in the navigation

The symbols are shown in the same way in the navigation if the comparison result applies to at least one of the parameters below the displayed navigation structure.

Applying values from the online program as start values

In order to apply optimized values from the CPU to the project as start values, you create a snapshot of the monitor values. Values of the snapshot marked as a "Setpoint" are then applied to the project as start values.

Requirements

- The technology object is of type "PID_Compact" or "PID_3Step".
- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").
- The Parameter view of the technology object is open.
- The “Monitor all” button is selected.

Procedure

To apply optimized values from the CPU, follow these steps:

1. Click the "Create snapshot of monitor values and accept setpoints of this snapshot as start values" icon .

Result

The current monitor values are applied to the "Snapshot" column and their setpoints are copied to the "Start value project" column as new start values.

Note

Applying values of individual parameters

You can also apply the values of individual parameters that are not marked as a setpoint from the "Snapshot" column to the "Start values project" column. To do so, copy the values and insert them into the "Start value project" column using the "Copy" and "Paste" commands in the shortcut menu.

Initializing setpoints in the online program

You can initialize all parameters that are marked as a "Setpoint" in the Parameter view with new values in the CPU in one step. In so doing, the start values are downloaded from the project to the CPU. The CPU remains in "RUN" mode.

To avoid data loss on the CPU during a cold restart or warm restart, you must also download the technology object to the CPU.

DANGER

Danger when changing parameter values

Changing the parameter values while the plant is operating may result in severe damage to property and personal injury in the event of malfunctions or program errors.

Make sure that dangerous states cannot occur before you reinitialize the setpoints.

Requirements

- The technology object is of type "PID_Compact" or "PID_3Step".
- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").
- The Parameter view of the technology object is open.
- The “Monitor all” button is selected.
- The parameters marked as a "Setpoint" have a "Start value project" that is free of process-related and syntax errors

Procedure

To initialize all setpoints, follow these steps:

1. Enter the desired values in the "Start value project" column.
   Ensure that the start values are free of process-related and syntax errors.
2. Click the "Initialize setpoints" icon .

Result

The setpoints in the CPU are initialized with the start values from the project.

11.3.2.11 Display instance DB of a technology object.

An instance DB, in which the parameter and static variables are saved, is created for each technology object.

Procedure

To display the instance DB of a technology object, proceed as follows:

1. Open the CPU folder in the project tree.
2. Open the "Technology objects" folder.
3. Highlight a technology object.
4. Select the command "Open DB editor" in the shortcut menu.

11.3.3 PID control (S7-1200, S7-1500)

11.3.3.1 Using PID_Compact

Technology object PID_Compact

The technology object PID_Compact provides a continuous PID controller with integrated optimization. You can alternatively configure a pulse controller. Both manual and automatic mode are possible.

PID-Compact continuously acquires the measured process value within a control loop and compares it with the required setpoint. From the resulting control deviation, the instruction PID_Compact calculates an output value by which the process value is adapted as quickly and stable as possible to the setpoint. The output value for the PID controller consists of three actions:

- P action
  The proportional action of the output value increases in proportion to the control deviation.
- I action
  The integral action of the output value increases until the control deviation has been balanced.
- D action
  The derivative action increases with the rate of change of control deviation. The process value is corrected to the setpoint as quickly as possible. The derivative action will be reduced again if the rate of change of control deviation drops.

The instruction PID_Compact calculates the proportional, integral and derivative parameters for your controlled system during pretuning. Fine tuning can be used to tune the parameters further. You do not need to manually determine the parameters.

Additional information

- Overview of software controller (Page 9531)
- Add technology objects (Page 9533)
- Configure technology objects (Page 9534)
- Configuring PID_Compact V2 (Page 9560)
- Configuring PID_Compact V1 (Page 9575)

PID_Compact V2

Configuring PID_Compact V2

Basic settings

Introduction

Configure the following properties of the "PID_Compact" technology object under "Basic settings" in the Inspector window or in the configuration window:

- Physical quantity
- Control logic
- Start-up behavior after reset
- Setpoint (only in the Inspector window)
- Process value (only in the Inspector window)
- Output value (only in the Inspector window)

Setpoint, process value and output value

You can only configure the setpoint, process value and output value in the Inspector window of the programming editor. Select the source for each value:

- Instance DB
  The value saved in the instance DB is used.
  Value must be updated in the instance DB by the user program.
  There should be no value at the instruction.
  Change via HMI possible.
- Instruction
  The value connected to the instruction is used.
  The value is written to the instance DB each time the instruction is called.
  No change via HMI possible.

Controller type

Physical quantity

Select the physical quantity and unit of measurement for setpoint, process value, and disturbance variable in the "Controller type" group. Setpoint, process value, and disturbance variable is displayed in this unit of measurement.

Control logic

An increase of the output value is generally intended to cause an increase in the process value. This is referred to as a normal control logic.

PID_Compact does not work with negative proportional gain. Select the check box "Invert control logic" to reduce the process value with a higher output value.

Examples

- Opening the drain valve will reduce the level of a container's contents.
- Increasing cooling will reduce the temperature.

Startup characteristics

1. To switch to "Inactive" mode after CPU restart, clear the "Activate Mode after CPU restart" check box.
   To switch to the operating mode saved in the Mode parameter after CPU restart, select the "Activate Mode after CPU restart" check box.
2. In the "Set Mode to" drop-down list, select the mode that is to be enabled after a complete download to the device.
   After a complete download to the device, PID_Compact starts in the selected operating mode. With each additional restart, PID_Compact starts in the mode that was last saved in Mode.

Example

You have selected the "Activate Mode after CPU restart" check box and the entry "Pretuning" in the "Set Mode to" list. After a complete download to the device, PID_Compact starts in the "Pretuning" mode. If pretuning is still active, PID_Compact starts in "Pretuning" mode again after restart of the CPU. If pretuning was successfully completed and automatic mode is active, PID_Compact starts in "Automatic mode" after restart of the CPU.

Setpoint

Procedure

Proceed as follows to define a fixed setpoint:

1. Select "Instance DB".
2. Enter a setpoint, e.g. 80° C.
3. Delete any entry in the instruction.

Proceed as follows to define a variable setpoint:

1. Select "Instruction".
2. Enter the name of the REAL variable in which the setpoint is saved.
   Program-controlled assignment of various values to the REAL variable is possible, for example for the time controlled change of the setpoint.

Process value

PID_Compact will scale the value of the analog input to the physical quantity if you use the analog input value directly.

You will need to write a program for processing if you wish first to process the analog input value. The process value is, for example, not directly proportional to the value at the analog input. The processed process value must be in floating point format.

Procedure

Proceed as follows to use the analog input value without processing:

1. Select the entry "Input_PER" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the address of the analog input.

Proceed as follows to use the processed process value in floating point format:

1. Select the entry "Input" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the name of the variable in which the processed process value is saved.

Output value

PID_Compact offers three output values. Your actuator will determine which output value you use.

- Output_PER
  The actuator is triggered via an analog output and controlled with a continuous signal, e.g. 0...10V, 4...20mA.
- Output
  The output value needs to be processed by the user program, for example because of nonlinear actuator response.
- Output_PWM
  The actuator is controlled via a digital output. Pulse width modulation creates minimum ON and minimum OFF times.

Procedure

Proceed as follows to use the analog output value:

1. Select the entry "Output_PER (analog)" in the drop-down list "Output".
2. Select "Instruction".
3. Enter the address of the analog output.

Proceed as follows to process the output value using the user program:

1. Select the entry "Output" in the drop-down list "Output".
2. Select "Instance DB".
   The calculated output value is saved in the instance data block.
3. For the preparation of the output value, use the output parameter Output.
4. Transfer the processed output value to the actuator via a digital or analog CPU output.

Proceed as follows to use the digital output value:

1. Select the entry "Output_PWM" in the drop-down list "Output".
2. Select "Instruction".
3. Enter the address of the digital output.

Process value settings

Scaling the process value

If you have configured the use of Input_PER in the basic setting, you must convert the value of the analog input to the physical quantity of the process value. The current configuration is displayed in the Input_PER display.

Input_PER will be scaled using a low and high value pair if the process value is directly proportional to the value of the analog input.

Procedure

To scale the process value, follow these steps:

1. Enter the low pair of values in the "Scaled low process value" and "Low" input fields.
2. Enter the high pair of values in the "Scaled high process value" and "High" input boxes.

Default settings for the value pairs are stored in the hardware configuration. To use the value pairs from the hardware configuration, follow these steps:

1. Select the PID_Compact instruction in the programming editor.
2. Interconnect Input_PER with an analog input in the basic settings.
3. Click the "Automatic setting" button in the process value settings.

The existing values will be overwritten with the values from the hardware configuration.

Process value limits

You must specify an appropriate absolute high limit and low limit for the process value as limit values for your controlled system. As soon as the process value violates these limits, an error occurs (ErrorBits = 0001h). Tuning is canceled when the process value limits are violated. You can configure how PID_Compact reacts to an error in automatic mode in the output value settings.

Advanced settings

Monitoring process value

Configure a warning high and low limit for the process value in the "Process value monitoring" configuration window. If one of the warning limits is exceeded or undershot during operation, a warning will be displayed at the PID_Compact instruction:

- At the InputWarning_H output parameter if the warning high limit has been exceeded
- At the InputWarning_L output parameter if the warning low limit has been undershot

The warning limits must be within the process value high and low limits.

The process value high and low limits will be used if you do not enter values.

Example

Process value high limit = 98 °C; warning high limit = 90 °C

Warning low limit = 10 °C; process value low limit = 0 °C

PID_Compact will respond as follows:

Process value	InputWarning_H	InputWarning_L	ErrorBits	Operating mode
> 98 °C	TRUE	FALSE	0001h	Inactive or Substitute output value with error monitoring
≤ 98 °C and > 90 °C	TRUE	FALSE	0000h	Automatic mode
≤ 90 °C and ≥ 10 °C	FALSE	FALSE	0000h	Automatic mode
< 10 °C and ≥ 0 °C	FALSE	TRUE	0000h	Automatic mode
< 0 °C	FALSE	TRUE	0001h	Inactive or Substitute output value with error monitoring

In the output value settings, you can specify the reaction of PID_Compact when the process value high limit or low limit is violated.

See also

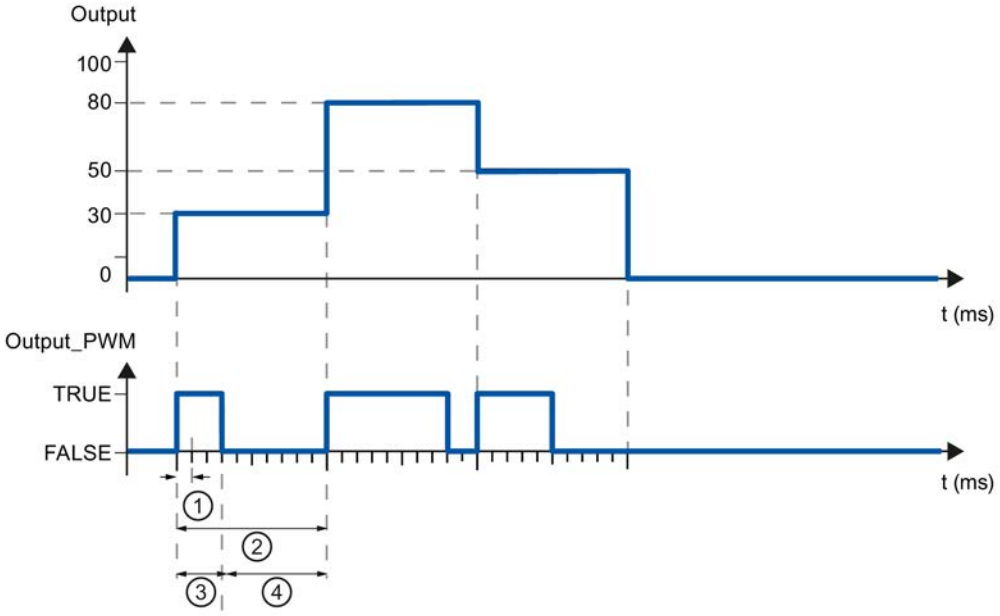
Parameters State and Mode V2 (Page 4947)

PWM limits

The value at the output parameter Output is transformed into a pulse sequence that is output at output parameter Output_PWM by means of a pulse width modulation. Output is calculated in the PID algorithm sampling time, Output_PWM is output in the PID_Compact sampling time.

The PID algorithm sampling time is determined during pretuning or fine tuning. If manually setting the PID parameters, you will also need to configure the PID algorithm sampling time. The PID_Compact sampling time is equivalent to the cycle time of the calling OB.

The pulse duration is proportional to the value at Output and is always an integer multiple of the PID_Compact sampling time.



① PID_Compact sampling time

② PID algorithm sampling time

③ Pulse duration

④ Break time

The "Minimum ON time" and the "Minimum OFF time" are rounded to an integer multiple of the PID_Compact sampling time.

A pulse or a break is never shorter than the minimum ON or OFF time. The inaccuracies this causes are added up and compensated in the next cycle.

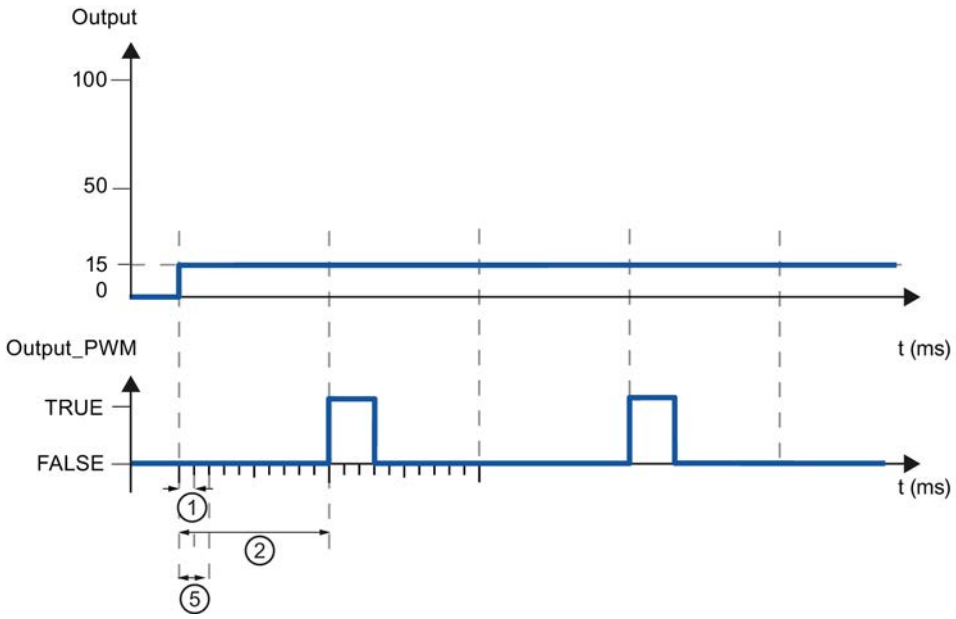
Example

PID_Compact sampling time = 100 ms

PID algorithm sampling time = 1000 ms

Minimum ON time = 200 ms

Output is a constant 15%. The smallest pulse that PID_Compact can output is 20%. In the first cycle, no pulse is output. In the second cycle, the pulse not output in the first cycle is added to the pulse of the second cycle.



① PID_Compact sampling time
② PID algorithm sampling time
⑤ Minimum ON time

In order to minimize operation frequency and conserve the actuator, extend the minimum ON and OFF times.

If you are using "Output" or "Output_PER", you must configure the value 0.0 for the minimum ON and OFF times.

Note

The minimum ON and OFF times only affect the output parameter Output_PWM and are not used for any pulse generators integrated in the CPU.

Output value

Output value limits

In the "Output value limits" configuration window, configure the absolute limits of your output value in percent. Absolute output value limits are not violated in neither manual mode nor automatic mode. If an output value outside the limits is specified in manual mode, the effective value is limited in the CPU to the configured limits.

The output value limits must match the control logic.

The valid output value limit values depend on the Output used.

Output	-100.0 to 100.0%
Output_PER	-100.0 to 100.0%
Output_PWM	0.0 to 100.0%

Reaction to error

NOTICE
Your system may be damaged. If you output "Current value while error pending " or "Substitute output value while error pending" in the event of an error, PID_Compact remains in automatic mode. This may cause a violation of the process value limits and damage your system. It is essential to configure how your controlled system reacts in the event of an error to protect your system from damage.

PID_Compact is preset so that the controller stays active in most cases in the event of an error. If errors occur frequently in controller mode, this default reaction has a negative effect on the control response. In this case, check the Errorbits parameter and eliminate the cause of the error.

PID_Compact generates a programmable output value in response to an error:

- Zero (inactive)
  PID_Compact outputs 0.0 as output value for all errors and switches to "Inactive" mode. The controller is only reactivated by a falling edge at Reset or a rising edge at ModeActivate.
- Current value while error is pending
  If the following errors occur in **automatic mode**, PID_Compact returns to automatic mode as soon as the errors are no longer pending.
  If one or more of the following errors occur, PID_Compact stays in automatic mode:
  - 0001h: The "Input" parameter is outside the process value limits.
  - 0800h: Sampling time error
  - 40000h: Invalid value at Disturbance parameter.

  If one or more of the following errors occur in **automatic mode**, PID_Compact switches to "Substitute output value with error monitoring" mode and outputs the last valid output value:
  - 0002h: Invalid value at Input_PER parameter.
  - 0200h: Invalid value at Input parameter.
  - 0400h: Calculation of output value failed.
  - 1000h: Invalid value at Setpoint parameter.

  If an error occurs in **manual mode**, PID_Compact continues using the manual value as the output value. If the manual value is invalid, the substitute output value is used. If the manual value and substitute output value are invalid, the output value low limit is used.
  If the following error occurs during a **pretuning or fine tuning**, PID_Compact remains in active mode:
  - 0020h: Pretuning is not permitted during fine tuning.

  When any other error occurs, PID_Compact cancels the tuning and switches to the mode from which tuning was started.
  As soon as no errors are pending, PID_Compact returns to automatic mode.
- Substitute output value while error is pending
  PID_Compact outputs the substitute output value.
  If the following error occurs, PID_Compact stays in "Substitute output value with error monitoring" mode and outputs the output value low limit:
  - 20000h: Invalid value at SubstituteOutput tag.

  For all other errors, PID_Compact reacts as described for "Current value while error is pending".

See also

Parameters State and Mode V2 (Page 4947)

PID parameters

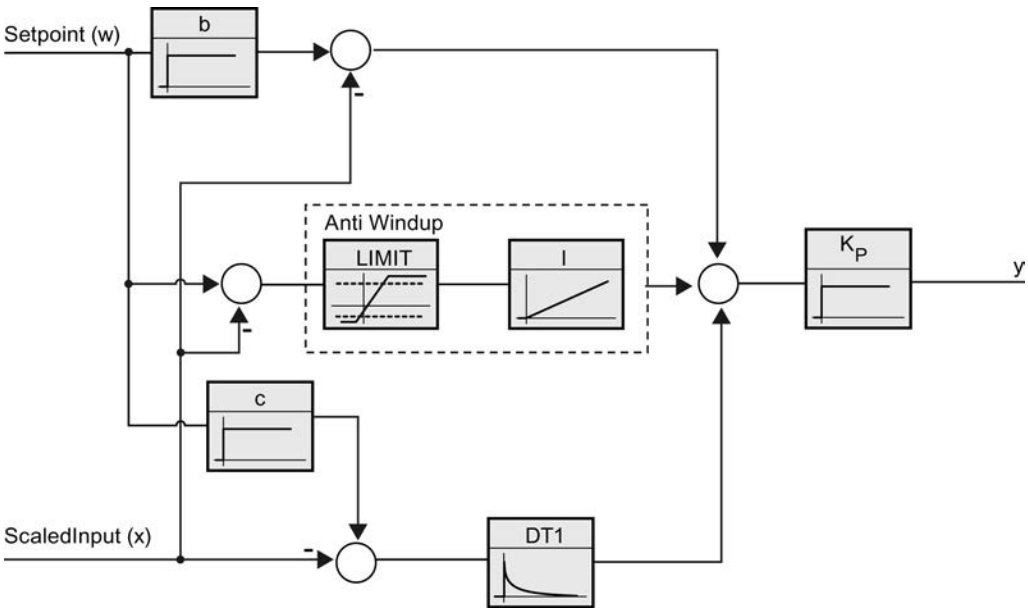
The PID parameters are displayed in the "PID Parameters" configuration window. The PID parameters will be adapted to your controlled system during controller tuning. You do not need to enter the PID parameters manually.

The PID algorithm operates according to the following equation:

$$y = K_p \left[(b \cdot w - x) + \frac{1}{T_I \cdot s}(w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1}(c \cdot w - x) \right]$$

Symbol	Description
y	Output value of the PID algorithm
K_p	Proportional gain
s	Laplace operator
b	Proportional action weighting
w	Setpoint
x	Process value
T_I	Integral action time
a	Derivative delay coefficient (derivative delay T1 = a × T_D)
T_D	Derivative action time
c	Derivative action weighting

The diagram below illustrates the integration of the parameters into the PID algorithm:



All PID parameters are retentive. If you enter the PID parameters manually, you must completely download PID_Compact.

Downloading technology objects to device (Page 9536)

Proportional gain

The value specifies the proportional gain of the controller. PID_Compact does not work with a negative proportional gain. Control logic is inverted under Basic settings > Controller type.

Integral action time

The integral action time determines the time behavior of the integral action. The integral action is deactivated with integral action time = 0.0.

Derivative action time

The derivative action time determines the time behavior of the derivative action. Derivative action is deactivated with derivative action time = 0.0.

Derivative delay coefficient

The derivative delay coefficient delays the effect of the derivative action.

Derivative delay = derivative action time × derivative delay coefficient

- 0.0: Derivative action is effective for one cycle only and therefore almost not effective.
- 0.5: This value has proved useful in practice for controlled systems with **one** dominant time constant.
- > 1.0: The greater the coefficient, the longer the effect of the derivative action is delayed.

Proportional action weighting

The proportional action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Proportional action for setpoint change is fully effective
- 0.0: Proportional action for setpoint change is not effective

The proportional action is always fully effective when the process value is changed.

Derivative action weighting

The derivative action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Derivative action is fully effective upon setpoint change
- 0.0: Derivative action is not effective upon setpoint change

The derivative action is always fully effective when the process value is changed.

PID algorithm sampling time

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is calculated during tuning and rounded to a multiple of the cycle time. All other functions of PID_Compact are executed at every call.

If you use Output_PWM, the accuracy of the output signal is determined by the ratio of the PID algorithm sampling time to the cycle time of the OB. The PID algorithm sampling time

corresponds to the time period of the pulse width modulation. The cycle time should be at least 10 times the PID algorithm sampling time.

Rule for tuning

Select whether PI or PID parameters are to be calculated in the "Controller structure" drop-down list.

- **PID**
  Calculates PID parameters during pretuning and fine tuning.
- **PI**
  Calculates PI parameters during pretuning and fine tuning.
- **User-defined**
  The drop-down list displays "User-defined" if you have configured different controller structures for pretuning and fine tuning via a user program.

Commissioning PID_Compact V2

Pretuning

The pretuning determines the process response to a jump change of the output value and searches for the point of inflection. The PID parameters are calculated from the maximum rate of rise and dead time of the controlled system. You obtain the best PID parameters when you perform pretuning and fine tuning.

The more stable the process value is, the easier it is to calculate the PID parameters and the more precise the result will be. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher compared to the noise. This is most likely the case in operating modes "Inactive" and "manual mode". The PID parameters are backed up before being recalculated.

Requirement

- The "PID_Compact" instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- Reset = FALSE
- PID_Compact is in one of the following modes: "Inactive", "Manual mode", or "Automatic mode".
- The setpoint and the process value lie within the configured limits (see "Process value monitoring" configuration).
- The difference between setpoint and process value is greater than 30% of the difference between process value high limit and process value low limit.
- The distance between the setpoint and the process value is > 50% of the setpoint.

Procedure

To perform pretuning, follow these steps:

1. Double-click the "PID_Compact > Commissioning" entry in the project tree.
2. Select the entry "Pretuning" in the "Tuning mode" drop-down list.
3. Click the "Start" icon.
   - An online connection will be established.
   - Value recording is started.
   - Pretuning is started.
   - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

   Note

   Click the "Stop" icon when the progress bar has reached 100% and it can be assumed the controller tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

Result

If pretuning was performed without an error message, the PID parameters have been tuned. PID_Compact switches to automatic mode and uses the tuned parameters. The tuned PID parameters will be retained during power OFF and a restart of the CPU.

If pretuning is not possible, PID_Compact responds with the configured reaction to errors.

See also

Parameters State and Mode V2 (Page 4947)

Fine tuning

Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are tuned for the operating point from the amplitude and frequency of this oscillation. All PID parameters are recalculated from the results. PID parameters from fine tuning usually have better master control and disturbance characteristics than PID parameters from pretuning. You obtain the best PID parameters when you perform pretuning and fine tuning.

PID_Compact automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value. The PID parameters are backed up before being recalculated.

Requirement

- The PID_Compact instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- Reset = FALSE

- The setpoint and the process value lie within the configured limits.
- The control loop has stabilized at the operating point. The operating point is reached when the process value corresponds to the setpoint.
- No disturbances are expected.
- PID_Compact is in one of the following operating modes: Inactive, automatic mode, or manual mode.

Process depends on initial situation

Fine tuning can be started from the following operating modes: "Inactive", "automatic mode", or "manual mode". Fine tuning proceeds as follows when started from:

- Automatic mode
  Start fine tuning from automatic mode if you wish to improve the existing PID parameters through tuning.
  PID_Compact controls the system using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start.
- Inactive or manual mode
  If the requirements for pretuning are met, pretuning is started. The determined PID parameters will be used for control until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start. If pretuning is not possible, PID_Compact responds with the configured reaction to errors.
  An attempt is made to reach the setpoint with the minimum or maximum output value if the process value for pretuning is already too near the setpoint. This can produce increased overshoot.

Procedure

To perform fine tuning, follow these steps:

1. Select the entry "Fine tuning" in the "Tuning mode" drop-down list.
2. Click the "Start" icon.
   - An online connection will be established.
   - Value recording is started.
   - The process of fine tuning is started.
   - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

Note

Click the "Stop" icon in the "Tuning mode" group when the progress bar has reached 100% and it is to be assumed that tuning is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

Result

If no errors occurred during fine tuning, the PID parameters have been tuned. PID_Compact switches to automatic mode and uses the tuned parameters. The tuned PID parameters will be retained during power OFF and a restart of the CPU.

If errors occurred during "fine tuning", PID_Compact responds with the configured response to errors.

See also

Parameters State and Mode V2 (Page 4947)

"Manual" mode

The following section describes how you can use the "manual mode" operating mode in the commissioning window of the "PID_Compact" technology object. Manual mode is also possible when an error is pending.

Requirement

- The "PID_Compact" instruction is called in a cyclic interrupt OB.
- An online connection to the CPU has been established and the CPU is in the "RUN" mode.

Procedure

Use "Manual mode" in the commissioning window if you want to test the controlled system by specifying a manual value. To define a manual value, follow these steps:

1. Click the "Start" icon.
2. Select the "Manual mode" check box in the "Online status of controller" area.
   PID_Compact operates in manual mode. The most recent current output value remains in effect.
3. Enter the manual value in the "Output" field as a % value.
4. Click the icon.

Result

The manual value is written to the CPU and immediately goes into effect.

Clear the "Manual mode" check box if the output value is to be specified again by the PID controller. The switchover to automatic mode is bumpless.

See also

Parameters State and Mode V2 (Page 4947)

PID_Compact V1

Configuring PID_Compact V1

Basic settings

Introduction

Configure the following properties of the "PID_Compact" technology object under "Basic settings" in the Inspector window or in the configuration window:

- Physical quantity
- Control logic
- Start-up behavior after reset
- Setpoint (only in the Inspector window)
- Process value (only in the Inspector window)
- Output value (only in the Inspector window)

Setpoint, process value and output value

You can only configure the setpoint, process value and output value in the Inspector window of the programming editor. Select the source for each value:

- Instance DB
  The value saved in the instance DB is used.
  Value must be updated in the instance DB by the user program.
  There should be no value at the instruction.
  Change via HMI possible.
- Instruction
  The value connected to the instruction is used.
  The value is written to the instance DB each time the instruction is called.
  No change via HMI possible.

Controller type

Physical quantity

Select the unit of measurement and physical quantity for the setpoint and process value in the "Controller type" group. The setpoint and process value will be displayed in this unit.

Control logic

An increase of the output value is generally intended to cause an increase in the process value. This is referred to as a normal control logic.

PID_Compact does not work with negative proportional gain. Select the check box "Invert control logic" to reduce the process value with a higher output value.

Examples

- Opening the drain valve will reduce the level of a container's contents.
- Increasing cooling will reduce the temperature.

Start-up behavior after reset

To change straight to the last active mode after restarting the CPU, select the "Enable last mode after CPU restart" check box.

PID_Compact will remain in "Inactive" mode if the check box is cleared.

Setpoint

Procedure

Proceed as follows to define a fixed setpoint:

1. Select "Instance DB".
2. Enter a setpoint, e.g. 80° C.
3. Delete any entry in the instruction.

Proceed as follows to define a variable setpoint:

1. Select "Instruction".
2. Enter the name of the REAL variable in which the setpoint is saved.
   Program-controlled assignment of various values to the REAL variable is possible, for example for the time controlled change of the setpoint.

Process value

PID_Compact will scale the value of the analog input to the physical quantity if you use the analog input value directly.

You will need to write a program for processing if you wish first to process the analog input value. The process value is, for example, not directly proportional to the value at the analog input. The processed process value must be in floating point format.

Procedure

Proceed as follows to use the analog input value without processing:

1. Select the entry "Input_PER" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the address of the analog input.

Proceed as follows to use the processed process value in floating point format:

1. Select the entry "Input" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the name of the variable in which the processed process value is saved.

Output value

PID_Compact offers three output values. Your actuator will determine which output value you use.

- Output_PER
  The actuator is triggered via an analog output and controlled with a continuous signal, e.g. 0...10V, 4...20mA.
- Output
  The output value needs to be processed by the user program, for example because of nonlinear actuator response.
- Output_PWM
  The actuator is controlled via a digital output. Pulse width modulation creates minimum ON and minimum OFF times.

Procedure

Proceed as follows to use the analog output value:

1. Select the entry "Output_PER (analog)" in the drop-down list "Output".
2. Select "Instruction".
3. Enter the address of the analog output.

Proceed as follows to process the output value using the user program:

1. Select the entry "Output" in the drop-down list "Output".
2. Select "Instance DB".
   The calculated output value is saved in the instance data block.
3. For the preparation of the output value, use the output parameter Output.
4. Transfer the processed output value to the actuator via a digital or analog CPU output.

Proceed as follows to use the digital output value:

1. Select the entry "Output_PWM" in the drop-down list "Output".
2. Select "Instruction".
3. Enter the address of the digital output.

Process value settings

Configure the scaling of your process value and specify the process value absolute limits In the "Process value settings" configuration window.

Scaling the process value

If you have configured the use of Input_PER in the basic settings, you will need to convert the value of the analog input into the physical quantity of the process value. The current configuration will be displayed in the Input_PER display.

Input_PER will be scaled using a low and high value pair if the process value is directly proportional to the value of the analog input.

1. Enter the low pair of values in the "Scaled low process value" and "Low" input fields.
2. Enter the high pair of values in the "Scaled high process value" and "High" input boxes.

Default settings for the value pairs are saved in the hardware configuration. Proceed as follows to use the value pairs from the hardware configuration:

1. Select the instruction PID_Compact in the programming editor.
2. Connect Input_PER with an analog input in the basic settings.
3. Click on the "Automatic setting" button in the process value settings.

The existing values will be overwritten with the values from the hardware configuration.

Monitoring process value

Specify the absolute high and low limit of the process value. As soon as these limits are violated during operation, the controller switches off and the output value is set to 0%. You must enter reasonable limits for your controlled system. Reasonable limits are important during optimization to obtain optimal PID parameters.

The default for the "High limit process value" is 120 %. At the I/O input, the process value can be a maximum of 18% higher than the standard range (overrange). An error is no longer reported for a violation of the "High limit process value". Only a wire-break and a short-circuit are recognized and the PID_Compact switches to "Inactive" mode.

⚠ WARNING
If you set very high process value limits (for example $-3.4*10^{38}$...$+3.4*10^{38}$), process value monitoring will be disabled. Your system may then be damaged if an error occurs.

See also

Monitoring process value (Page 9579)

PWM limits (Page 9579)

Output value limits (Page 9581)

PID parameters (Page 9582)

Advanced settings

Monitoring process value

Configure a warning high and low limit for the process value in the "Process value monitoring" configuration window. If one of the warning limits is exceeded or undershot during operation, a warning will be displayed at the PID_Compact instruction:

- At the InputWarning_H output parameter if the warning high limit has been exceeded
- At the InputWarning_L output parameter if the warning low limit has been undershot

The warning limits must be within the process value high and low limits.

The process value high and low limits will be used if you do not enter values.

Example

Process value high limit = 98° C; warning high limit = 90° C

Warning low limit = 10° C; process value low limit = 0° C

PID_Compact will respond as follows:

Process value	InputWarning_H	InputWarning_L	Operating mode
> 98° C	TRUE	FALSE	Inactive
≤ 98° C and > 90° C	TRUE	FALSE	Automatic mode
≤ 90° C and ≥ 10° C	FALSE	FALSE	Automatic mode
< 10° C and ≥ 0° C	FALSE	TRUE	Automatic mode
< 0° C	FALSE	TRUE	Inactive

See also

Process value settings (Page 9577)

PWM limits (Page 9579)

Output value limits (Page 9581)

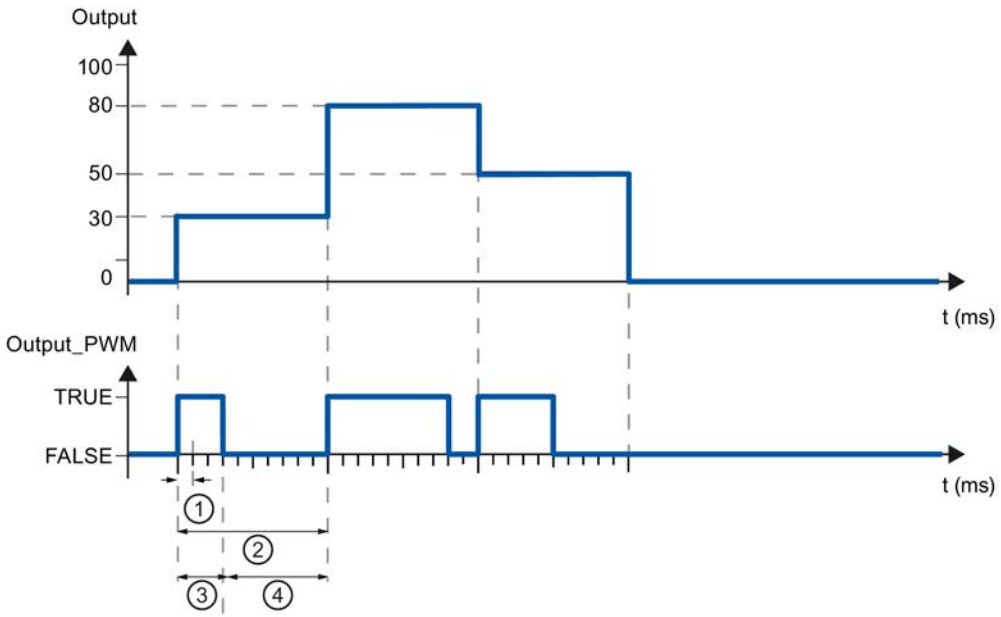
PID parameters (Page 9582)

PWM limits

The value at the output parameter Output is transformed into a pulse sequence that is output at output parameter Output_PWM by means of a pulse width modulation. Output is calculated in the PID algorithm sampling time, Output_PWM is output in the PID_Compact sampling time.

The PID algorithm sampling time is determined during pretuning or fine tuning. If manually setting the PID parameters, you will also need to configure the PID algorithm sampling time. The PID_Compact sampling time is equivalent to the cycle time of the calling OB.

The pulse duration is proportional to the value at Output and is always an integer multiple of the PID_Compact sampling time.



① PID_Compact sampling time

② PID algorithm sampling time

③ Pulse duration

④ Break time

The "Minimum ON time" and the "Minimum OFF time" are rounded to an integer multiple of the PID_Compact sampling time.

A pulse or a break is never shorter than the minimum ON or OFF time. The inaccuracies this causes are added up and compensated in the next cycle.

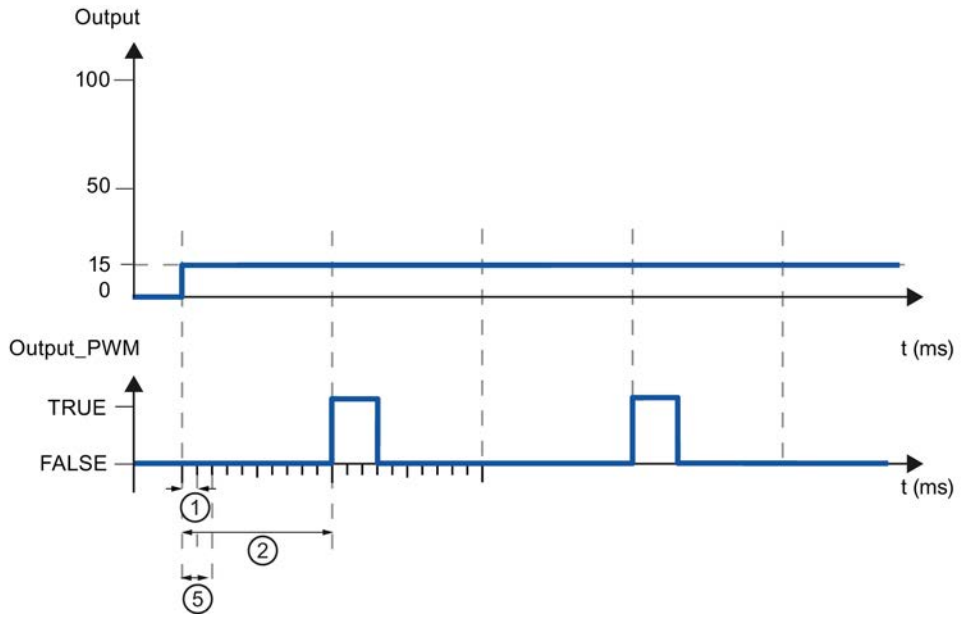
Example

PID_Compact sampling time = 100 ms

PID algorithm sampling time = 1000 ms

Minimum ON time = 200 ms

Output is a constant 15%. The smallest pulse that PID_Compact can output is 20%. In the first cycle, no pulse is output. In the second cycle, the pulse not output in the first cycle is added to the pulse of the second cycle.



① PID_Compact sampling time

② PID algorithm sampling time

⑤ Minimum ON time

In order to minimize operation frequency and conserve the actuator, extend the minimum ON and OFF times.

If you are using "Output" or "Output_PER", you must configure the value 0.0 for the minimum ON and OFF times.

Note

The minimum ON and OFF times only affect the output parameter Output_PWM and are not used for any pulse generators integrated in the CPU.

See also

Process value settings (Page 9577)

Monitoring process value (Page 9579)

Output value limits (Page 9581)

PID parameters (Page 9582)

Output value limits

In the "Output value limits" configuration window, configure the absolute limits of your output value in percent. Absolute output value limits are not violated in neither manual mode nor in automatic mode. If a output value outside the limits is specified in manual mode, the effective value is limited in the CPU to the configured limits.

The valid output value limit values depend on the Output used.

Output	-100.0 to 100.0
Output_PER	-100.0 to 100.0
Output_PWM	0.0 to 100.0

PID_Compact sets the output value to 0.0 if an error occurs. 0.0 must therefore always be within the output value limits. You will need to add an offset to Output and Output_PER in the user program if you want an output value low limit of greater than 0.0.

See also

Process value settings (Page 9577)

Monitoring process value (Page 9579)

PWM limits (Page 9579)

PID parameters (Page 9582)

PID parameters

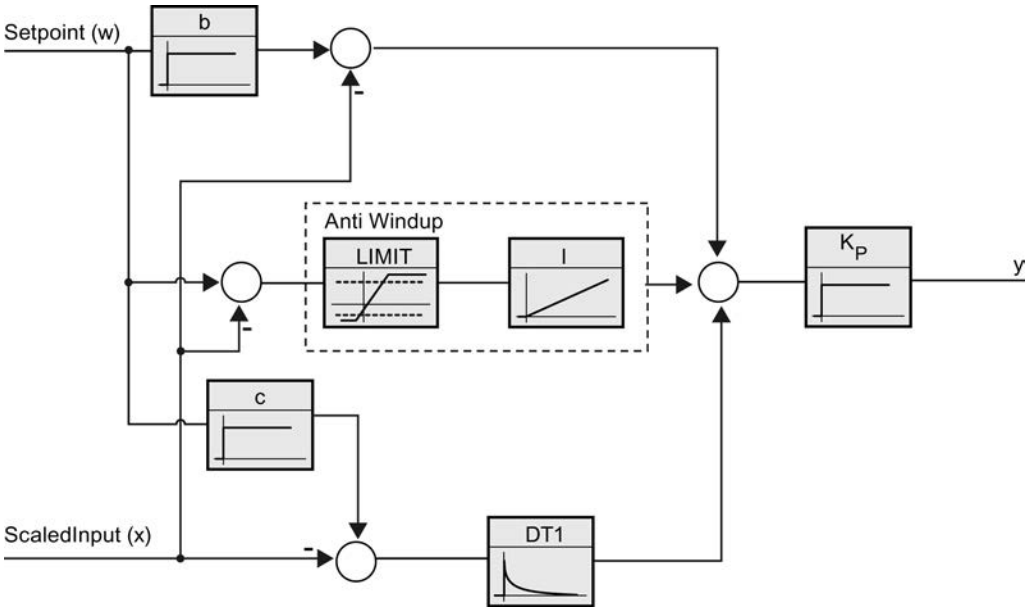
The PID parameters are displayed in the "PID Parameters" configuration window. The PID parameters will be adapted to your controlled system during controller tuning. You do not need to enter the PID parameters manually.

The PID algorithm operates according to the following equation:

$$y = K_p \left[(b \cdot w - x) + \frac{1}{T_I \cdot s} (w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1} (c \cdot w - x) \right]$$

Symbol	Description
y	Output value of the PID algorithm
K_p	Proportional gain
s	Laplace operator
b	Proportional action weighting
w	Setpoint
x	Process value
T_I	Integral action time
a	Derivative delay coefficient (derivative delay T1 = a × T_D)
T_D	Derivative action time
c	Derivative action weighting

The diagram below illustrates the integration of the parameters into the PID algorithm:



All PID parameters are retentive. If you enter the PID parameters manually, you must completely download PID_Compact.

Auto-Hotspot

Proportional gain

The value specifies the proportional gain of the controller. PID_Compact does not work with a negative proportional gain. Control logic is inverted under Basic settings > Controller type.

Integral action time

The integral action time determines the time behavior of the integral action. The integral action is deactivated with integral action time = 0.0.

Derivative action time

The derivative action time determines the time behavior of the derivative action. Derivative action is deactivated with derivative action time = 0.0.

Derivative delay coefficient

The derivative delay coefficient delays the effect of the derivative action.

Derivative delay = derivative action time × derivative delay coefficient

- 0.0: Derivative action is effective for one cycle only and therefore almost not effective.
- 0.5: This value has proved useful in practice for controlled systems with **one** dominant time constant.
- > 1.0: The greater the coefficient, the longer the effect of the derivative action is delayed.

Proportional action weighting

The proportional action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Proportional action for setpoint change is fully effective
- 0.0: Proportional action for setpoint change is not effective

The proportional action is always fully effective when the process value is changed.

Derivative action weighting

The derivative action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Derivative action is fully effective upon setpoint change
- 0.0: Derivative action is not effective upon setpoint change

The derivative action is always fully effective when the process value is changed.

PID algorithm sampling time

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is calculated during tuning and rounded to a multiple of the cycle time. All other functions of PID_Compact are executed at every call.

If you use Output_PWM, the accuracy of the output signal is determined by the ratio of the PID algorithm sampling time to the cycle time of the OB. The PID algorithm sampling time corresponds to the time period of the pulse width modulation. The cycle time should be at least 10 times the PID algorithm sampling time.

Rule for tuning

Select whether PI or PID parameters are to be calculated in the "Controller structure" drop-down list.

- PID
  Calculates PID parameters during pretuning and fine tuning.
- PI
  Calculates PI parameters during pretuning and fine tuning.
- User-defined
  The drop-down list displays "User-defined" if you have configured different controller structures for pretuning and fine tuning via a user program.

See also

Downloading technology objects to device (Page 9536)

Commissioning PID_Compact V1

Commissioning

The commissioning window helps you commission the PID controller. You can monitor the values for the setpoint, process value and output value along the time axis in the trend view. The following functions are supported in the commissioning window:

- Controller pretuning
- Controller fine tuning
  Use fine tuning for fine adjustments to the PID parameters.
- Monitoring the current closed-loop control in the trend view
- Testing the controlled system by specifying a manual output value

All functions require an online connection to the CPU to have been established.

Basic handling

- Select the desired sampling time in the "Sampling time" drop-down list.
  All values in the commissioning window are updated in the selected update time.
- Click the "Start" icon in the measuring group if you want to use the commissioning functions. Value recording is started. The current values for the setpoint, process value and output value are entered in the trend view. Operation of the commissioning window is enabled.
- Click the "Stop" icon if you want to end the commissioning functions.
  The values recorded in the trend view can continue to be analyzed.

Closing the commissioning window will terminate recording in the trend view and delete the recorded values.

See also

Pretuning (Page 9585)

Fine tuning (Page 9587)

"Manual" mode (Page 9588)

Pretuning

The pretuning determines the process response to a jump change of the output value and searches for the point of inflection. The tuned PID parameters are calculated as a function of the maximum slope and dead time of the controlled system.

The more stable the process value is, the easier it is to calculate the PID parameters and the more precise the result will be. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher compared to the noise. The PID parameters are backed up before being recalculated.

Requirement

- The "PID_Compact" instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- PID_Compact is in "inactive" or "manual" mode.
- The setpoint may not be changed during controller tuning. PID_Compact will otherwise be deactivated.
- The setpoint and the process value lie within the configured limits (see "Process value monitoring" configuration).
- The difference between setpoint and process value is greater than 30% of the difference between process value high limit and process value low limit.
- The distance between the setpoint and the process value is > 50% of the setpoint.

Procedure

To perform pretuning, follow these steps:

1. Double-click the "PID_Compact > Commissioning" entry in the project tree.
2. Select the entry "Pretuning" in the "Tuning mode" drop-down list.
3. Click the "Start" icon.
   - An online connection will be established.
   - Value recording is started.
   - Pretuning is started.
   - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

   Note

   Click the "Stop" icon when the progress bar has reached 100% and it is to be assumed the controller tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

Result

If pretuning was performed without an error message, the PID parameters have been tuned. PID_Compact switches to automatic mode and uses the tuned parameters. The tuned PID parameters will be retained during power OFF and a restart of the CPU.

If pretuning is not possible, PID_Compact will change to "Inactive" mode.

See also

Parameters State and sRet.i_Mode V1 (Page 4965)

Commissioning (Page 9585)

Fine tuning (Page 9587)

"Manual" mode (Page 9588)

Fine tuning

Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are optimized for the operating point from the amplitude and frequency of this oscillation. All PID parameters are recalculated on the basis of the findings. PID parameters from fine tuning usually have better master control and disturbance behavior than PID parameters from pretuning.

PID_Compact automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value. The PID parameters are backed up before being recalculated.

Requirement

- The PID_Compact instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- The setpoint and the process value lie within the configured limits (see "Process value monitoring" configuration).
- The control loop has stabilized at the operating point. The operating point is reached when the process value corresponds to the setpoint.
- No disturbances are expected.
- The setpoint may not be changed during controller tuning.
- PID_Compact is in inactive mode, automatic mode or manual mode.

Process depends on initial situation

Fine tuning can be started in "inactive", "automatic" or "manual" mode. Fine tuning proceeds as follows when started in:

- Automatic mode
  Start fine tuning in automatic mode if you wish to improve the existing PID parameters using controller tuning.
  PID_Comact will regulate using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start.
- Inactive or manual mode
  If the requirements for pretuning are met, pretuning is started. The PID parameters established will be used for adjustment until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start. If pretuning is not possible, PID_Compact will change to "Inactive" mode.
  An attempt is made to reach the setpoint with a minimum or maximum output value if the process value for pretuning is already too near the setpoint. This can produce increased overshoot.

Procedure

Proceed as follows to carry out "fine tuning":

1. Select the entry "Fine tuning" in the "Tuning mode" drop-down list.
2. Click the "Start" icon.
   - An online connection will be established.
   - Value recording is started.
   - The process of fine tuning is started.
   - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

Note

Click the "Stop" icon in the "Tuning mode" group when the progress bar has reached 100% and it is to be assumed the controller tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

Result

The PID parameters will have been optimized if fine tuning has been executed without errors. PID_Compact changes to automatic mode and uses the optimized parameters. The optimized PID parameters will be retained during power OFF and a restart of the CPU.

If errors occurred during "fine tuning", PID_Compact will change to "inactive" mode.

See also

Parameters State and sRet.i_Mode V1 (Page 4965)

Commissioning (Page 9585)

Pretuning (Page 9585)

"Manual" mode (Page 9588)

"Manual" mode

The following section describes how you can use the "Manual" operating mode in the commissioning window of the "PID Compact" technology object.

Requirement

- The "PID_Compact" instruction is called in a cyclic interrupt OB.
- An online connection to the CPU has been established and the CPU is in the "RUN" mode.
- The functions of the commissioning window have been enabled via the "Start" icon.

Procedure

Use "Manual mode" in the commissioning window if you want to test the process by specifying a manual value. To define a manual value, proceed as follows:

1. Select the check box "Manual mode" in the "Online status of the controller" area. PID_Compact operates in manual mode. The most recent current output value remains in effect.
2. Enter the manual value in the "Output" field as a % value.
3. Click the control icon .

Result

The manual value is written to the CPU and immediately goes into effect.

Note

PID_Compact continues to monitor the process value. If the process value limits are exceeded, PID_Compact is deactivated.

Clear the "Manual mode" check box if the output value is to be specified again by the PID controller. The change to automatic mode is bumpless.

See also

Parameters State and sRet.i_Mode V1 (Page 4965)

Commissioning (Page 9585)

Pretuning (Page 9585)

Fine tuning (Page 9587)

11.3.3.2 Using PID_3Step

Technology object PID_3Step

The technology object PID_3Step provides a PID controller with tuning for valves or actuators with integral response.

You can configure the following controllers:

- Three-point step controller with position feedback
- Three-point step controller without position feedback
- Valve controller with analog output value

PID_3Step continuously acquires the measured process value within a control loop and compares it with the setpoint. From the resulting control deviation, PID_3Step calculates an output value through which the process value reaches the setpoint as quickly and steadily as possible. The output value for the PID controller consists of three actions:

- P action
  The proportional action of the output value increases in proportion to the control deviation.
- I action
  The integral action of the output value increases until the control deviation has been balanced.
- D action
  The derivative action increases with the rate of change of control deviation. The process value is corrected to the setpoint as quickly as possible. The derivative action will be reduced again if the rate of change of control deviation drops.

The instruction PID_3Step calculates the proportional, integral and derivative parameters for your controlled system during pretuning. Fine tuning can be used to tune the parameters further. You do not need to manually determine the parameters.

Additional information

- Overview of software controller (Page 9531)
- Add technology objects (Page 9533)
- Configure technology objects (Page 9534)
- Configuring PID_3Step V2 (Page 9590)
- Configuring PID_3Step V1 (Page 9607)

PID_3Step V2

Configuring PID_3Step V2

Basic settings

Introduction

Configure the following properties of the "PID_3Step" technology object under "Basic settings" in the Inspector window or in the configuration window:

- Physical quantity
- Control logic
- Start-up behavior after reset
- Setpoint (only in the Inspector window)
- Process value (only in the Inspector window)
- Output value (only in the Inspector window)
- Position feedback (only in the Inspector window)

Setpoint, process value, output value and position feedback

You can only configure the setpoint, process value, output value and position feedback in the Inspector window of the programming editor. Select the source for each value:

- Instance DB
  The value saved in the instance DB is used.
  Value must be updated in the instance DB by the user program.
  There should be no value at the instruction.
  Change via HMI possible.
- Instruction
  The value connected to the instruction is used.
  The value is written to the instance DB each time the instruction is called.
  No change via HMI possible.

Controller type

Physical quantity

Select the physical quantity and unit of measurement for setpoint, process value, and disturbance variable in the "Controller type" group. Setpoint, process value, and disturbance variable is displayed in this unit of measurement.

Control logic

An increase of the output value is generally intended to cause an increase in the process value. This is referred to as a normal control logic.

PID_3Step does not work with negative proportional gain. Select the check box "Invert control logic" to reduce the process value with a higher output value.

Examples

- Opening the drain valve will reduce the level of a container's contents.
- Increasing cooling will reduce the temperature.

Startup characteristics

1. To switch to "Inactive" mode after CPU restart, clear the "Activate Mode after CPU restart" check box.
   To switch to the operating mode saved in the Mode parameter after CPU restart, select the "Activate Mode after CPU restart" check box.
2. In the "Set Mode to" drop-down list, select the mode that is to be enabled after a complete download to the device.
   After a complete download to the device, PID_3Step starts in the selected operating mode.
   With each additional restart, PID_3Step starts in the mode that was last saved in Mode.

Example

You have selected the "Activate Mode after CPU restart" check box and the entry "Pretuning" in the "Set Mode to" list. After a complete download to the device, PID_3Step starts in the "Pretuning" mode. If pretuning is still active, PID_3Step starts in "Pretuning" mode again after

restart of the CPU. If pretuning was successfully completed and automatic mode is active, PID_3Step starts in "Automatic mode" after restart of the CPU.

Setpoint

Procedure

Proceed as follows to define a fixed setpoint:

1. Select "Instance DB".
2. Enter a setpoint, e.g. 80° C.
3. Delete any entry in the instruction.

Proceed as follows to define a variable setpoint:

1. Select "Instruction".
2. Enter the name of the REAL variable in which the setpoint is saved.
   Program-controlled assignment of various values to the REAL variable is possible, for example for the time controlled change of the setpoint.

Process value

PID_3Step will scale the value of the analog input to the physical quantity if you use the analog input value directly.

You will need to write a program for processing if you wish first to process the analog input value. The process value is, for example, not directly proportional to the value at the analog input. The processed process value must be in floating point format.

Procedure

Proceed as follows to use the analog input value without processing:

1. Select the entry "Input_PER" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the address of the analog input.

Proceed as follows to use the processed process value in floating point format:

1. Select the entry "Input" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the name of the variable in which the processed process value is saved.

Position feedback

Position feedback configuration depends upon the actuator used.

- Actuator without position feedback
- Actuator with digital endstop signals

- Actuator with analog position feedback
- Actuator with analog position feedback and endstop signals

Actuator without position feedback

Proceed as follows to configure PID_3Step for an actuator without position feedback:

1. Select the entry "No Feedback" in the drop-down list "Feedback".

Actuator with digital endstop signals

Proceed as follows to configure PID_3Step for an actuator with endstop signals:

1. Select the entry "No Feedback" in the drop-down list "Feedback".
2. Activate the "Actuator endstop signals" check box.
3. Select "Instruction" as source for Actuator_H and Actuator_L.
4. Enter the addresses of the digital inputs for Actuator_H and Actuator_L.

Actuator with analog position feedback

Proceed as follows to configure PID_3Step for an actuator with analog position feedback:

1. Select the entry "Feedback" or "Feedback_PER" in the drop-down list "Feedback".
   - Use the analog input value for Feedback_PER. Configure Feedback_PER scaling in the actuator settings.
   - Process the analog input value for Feedback using your user program.
2. Select "Instruction" as source.
3. Enter the address of the analog input or the variable of your user program.

Actuator with analog position feedback and endstop signals

Proceed as follows to configure PID_3Step for an actuator with analog position feedback and endstop signals:

1. Select the entry "Feedback" or "Feedback_PER" in the drop-down list "Feedback".
2. Select "Instruction" as source.
3. Enter the address of the analog input or the variable of your user program.
4. Activate the "Actuator endstop signals" check box.
5. Select "Instruction" as source for Actuator_H and Actuator_L.
6. Enter the addresses of the digital inputs for Actuator_H and Actuator_L.

Output value

PID_3Step offers an analog output value (Output_PER) and digital output values (Output_UP, Output_DN). Your actuator will determine which output value you use.

- Output_PER
  The actuator is triggered via an analog output and controlled with a continuous signal, e.g. 0...10V, 4...20mA.
- Output_UP, Output_DN
  The actuator is controlled via two digital outputs.

Procedure

Proceed as follows to use the analog output value:

1. Select the entry "Output (analog)" in the drop-down list "Output".
2. Select "Instruction".
3. Enter the address of the analog output.

Proceed as follows to use the digital output value:

1. Select the entry "Output (digital)" in the drop-down list "Output".
2. Select "Instruction" for Output_UP and Output_DN.
3. Enter the addresses of the digital outputs.

Proceed as follows to process the output value using the user program:

1. Select the entry corresponding to the actuator in the drop-down list "Output".
2. Select "Instruction".
3. Enter the name of the variable you are using to process the output value.
4. Transfer the processed output value to the actuator by means of an analog or digital CPU output.

Process value settings

Scaling the process value

If you have configured the use of Input_PER in the basic setting, you must convert the value of the analog input to the physical quantity of the process value. The current configuration is displayed in the Input_PER display.

Input_PER will be scaled using a low and high value pair if the process value is directly proportional to the value of the analog input.

Procedure

To scale the process value, follow these steps:

1. Enter the low pair of values in the "Scaled low process value" and "Low" text boxs.
2. Enter the high pair of values in the "Scaled high process value" and "High" input boxes.

Default settings for the value pairs are stored in the hardware configuration. To use the value pairs from the hardware configuration, follow these steps:

1. Select the PID_3Step instruction in the programming editor.
2. Interconnect Input_PER with an analog input in the basic settings.
3. Click the "Automatic setting" button in the process value settings.

The existing values will be overwritten with the values from the hardware configuration.

Process value limits

You must specify an appropriate absolute high limit and low limit for the process value as limit values for your controlled system. As soon as the process value violates these limits, an error occurs (ErrorBits = 0001h). Tuning is canceled when the process value limits are violated. You can specify how PID_3Step responds to errors in automatic mode in the actuator settings.

Actuator settings

Actuator

Actuator-specific times

Configure the motor transition time and the minimum ON and OFF times to prevent damage to the actuator. You can find the specifications in the actuator data sheet.

The motor transition time is the time in seconds the motor requires to move the actuator from the closed to the opened state. You can measure the motor transition time during commissioning.

The motor transition time is retentive. If you enter the motor transition time manually, you must completely download PID_3Step.

Downloading technology objects to device (Page 9536)

If you are using "Output_UP" or "Output_DN", you can reduce the switching frequency with the minimum on and minimum OFF time.

The on or off times calculated are totaled in automatic mode and only become effective when the sum is greater than or equal to the minimum on or OFF time.

Manual_UP = TRUE or Manual_DN = TRUE in manual mode operates the actuator for at least the minimum ON or OFF time.

Reaction to error

PID_3Step is preset so that the controller stays active in most cases in the event of an error. If errors occur frequently in controller mode, this default reaction has a negative effect on the

control response. In this case, check the Errorbits parameter and eliminate the cause of the error.

NOTICE
Your system may be damaged. If you output "Current value while error pending" or "Substitute output value while error pending" in the event of an error, PID_3Step remains in automatic mode even if the process value limits are violated. This may damage your system. It is essential to configure how your controlled system reacts in the event of an error to protect your system from damage.

PID_3Step generates a programmable output value in response to an error:

- Current value
  PID_3Step is switched off and no longer modifies the actuator position.
- Current value for error while error is pending
  The controller functions of PID_3Step are switched off and the position of the actuator is no longer changed.
  If the following errors occur in automatic mode, PID_3Step returns to automatic mode as soon as the errors are no longer pending.
  - 0002h: Invalid value at Input_PER parameter.
  - 0200h: Invalid value at Input parameter.
  - 0400h: Calculation of output value failed.
  - 1000h: Invalid value at Setpoint parameter.
  - 2000h: Invalid value at Feedback_PER parameter.
  - 4000h: Invalid value at Feedback parameter.
  - 8000h: Error during digital position feedback.
  - 20000h: Invalid value at SavePosition tag.

  If one or more of the following errors occur, PID_3Step stays in automatic mode:
  - 0001h: The Input parameter is outside the process value limits.
  - 0800h: Sampling time error
  - 40000h: Invalid value at Disturbance parameter.

  PID_3Step remains in manual mode if an error occurs in manual mode.
  If an error occurs during tuning or transition time measurement, PID_3Step switches to the mode in which tuning or transition time measurement was started. Only in the event of the following error is tuning not aborted:
  - 0020h: Pretuning is not permitted during fine tuning.

- Substitute output value
  PID_3Step moves the actuator to the substitute output value and then switches off.
- Substitute output value while error is pending
  PID_3Step moves the actuator to the substitute output value. When the substitute output value is reached, PID_3Step reacts as it does with "Current value for while error is pending".

Enter the substitute output value in "%".

Only substitute output values 0% and 100% can be approached precisely in the case of actuators without analog position feedback. A substitute output value not equal to 0% or 100% is approached via an internally simulated position feedback. This procedure does not, however, allow the exact approach of substitute output value.

All substitute output values can be approached precisely with actuators with analog position feedback.

Scaling position feedback

Scaling position feedback

If you have configured the use of Feedback_PER in the basic settings, you will need to convert the value of the analog input into %. The current configuration will be displayed in the "Feedback" display.

Feedback_PER is scaled using a low and high value pair.

1. Enter the low pair of values in the "Low endstop" and "Low" input boxes.
2. Enter the high pair of values in the "High endstop" and "High" input boxes.

"Low endstop" must be less than "High endstop"; "Low" must be less than "High".

The valid values for "High endstop" and "Low endstop" depend upon:

- No Feedback, Feedback, Feedback_PER
- Output (analog), Output (digital)

Output	Feedback	Low endstop	High endstop
Output (digital)	No Feedback	Cannot be set (0.0%)	Cannot be set (100.0%)
Output (digital)	Feedback	-100.0% or 0.0%	0.0% or +100.0%
Output (digital)	Feedback_PER	-100.0% or 0.0%	0.0% or +100.0%
Output (analog)	No Feedback	Cannot be set (0.0%)	Cannot be set (100.0%)
Output (analog)	Feedback	-100.0% or 0.0%	0.0% or +100.0%
Output (analog)	Feedback_PER	-100.0% or 0.0%	0.0% or +100.0%

Output value limits

Limiting the output value

You can exceed or undershoot the output value limits during the transition time measurement and with mode = 10. The output value is limited to these values in all other modes.

Enter the absolute output value limits in the "Output value high limit" and "Output value low limit" input boxes. The output value limits must be within "Low endstop" and "High endstop".

If no Feedback is available and Output (digital) is set, you cannot limit the output value. Output_UP and Output_DN are then reset upon Actuator_H = TRUE or Actuator_L = TRUE. If no endstop signals are available, Output_UP and Output_DN are reset after a travel time of 150% of the motor actuating time.

Advanced settings

Monitoring process value

Configure a warning high and low limit for the process value in the "Process value monitoring" configuration window. If one of the warning limits is exceeded or undershot during operation, a warning will be displayed at the PID_3Step instruction:

- At the InputWarning_H output parameter if the warning high limit has been exceeded
- At the InputWarning_L output parameter if the warning low limit has been undershot

The warning limits must be within the process value high and low limits.

The process value high and low limits will be used if you do not enter values.

Example

Process value high limit = 98° C; warning high limit = 90° C

Warning low limit = 10° C; process value low limit = 0° C

PID_3Step will respond as follows:

Process value	InputWarning_H	InputWarning_L	ErrorBits	Operating mode
> 98° C	TRUE	FALSE	0001h	As configured
≤ 98° C and > 90° C	TRUE	FALSE	0000h	Automatic mode
≤ 90° C and ≥ 10° C	FALSE	FALSE	0000h	Automatic mode
< 10° C and ≥ 0° C	FALSE	TRUE	0000h	Automatic mode
< 0° C	FALSE	TRUE	0001h	As configured

In the actuator settings, you can configure the response of PID_3Step when the process value high limit or low limit is violated.

PID parameters

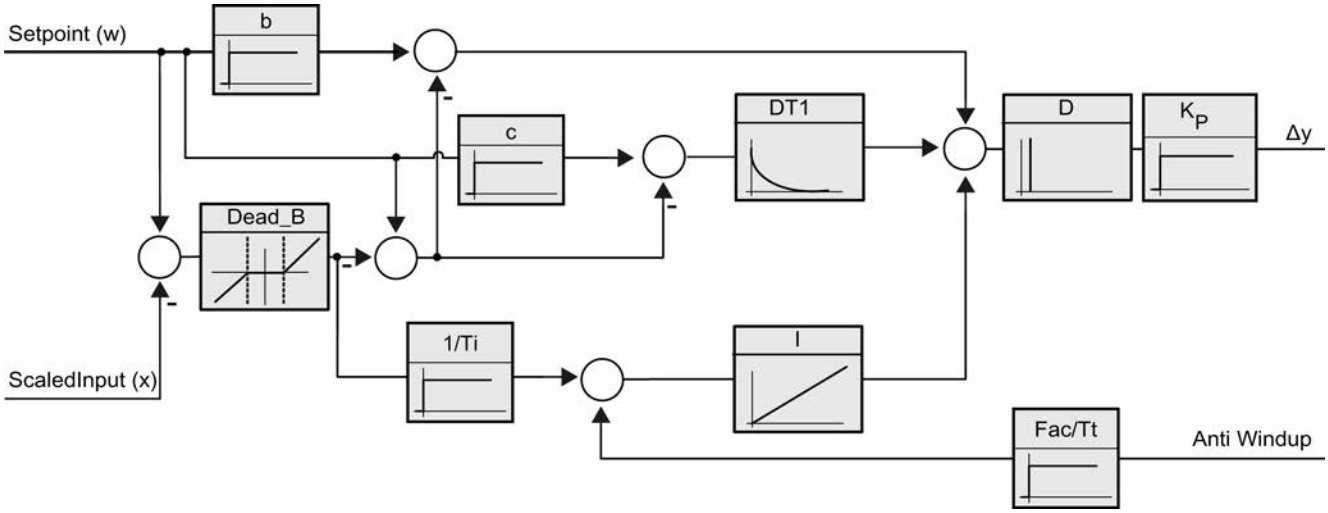
The PID parameters are displayed in the "PID Parameters" configuration window. The PID parameters will be adapted to your controlled system during controller tuning. You do not need to enter the PID parameters manually.

The PID algorithm operates according to the following equation:

$$\Delta y = K_p \cdot s \cdot \left[(b \cdot w - x) + \frac{1}{T_I \cdot s}(w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1}(c \cdot w - x) \right]$$

Symbol	Description
Δy	Output value of the PID algorithm
K_p	Proportional gain
s	Laplace operator
b	Proportional action weighting
w	Setpoint
x	Process value
T_I	Integral action time
a	Derivative delay coefficient (derivative delay T1 = a × T_D)
T_D	Derivative action time
c	Derivative action weighting

The diagram below illustrates the integration of the parameters into the PID algorithm:



All PID parameters are retentive. If you enter the PID parameters manually, you must completely download PID_3Step.

Downloading technology objects to device (Page 9536)

Proportional gain

The value specifies the proportional gain of the controller. PID_3Step does not work with a negative proportional gain. Control logic is inverted under Basic settings > Controller type.

Integral action time

The integral action time determines the time behavior of the integral action. The integral action is deactivated with integral action time = 0.0.

Derivative action time

The derivative action time determines the time behavior of the derivative action. Derivative action is deactivated with derivative action time = 0.0.

Derivative delay coefficient

The derivative delay coefficient delays the effect of the derivative action.

Derivative delay = derivative action time × derivative delay coefficient

- 0.0: Derivative action is effective for one cycle only and therefore almost not effective.
- 0.5: This value has proved useful in practice for controlled systems with **one** dominant time constant.
- > 1.0: The greater the coefficient, the longer the effect of the derivative action is delayed.

Proportional action weighting

The proportional action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Proportional action for setpoint change is fully effective
- 0.0: Proportional action for setpoint change is not effective

The proportional action is always fully effective when the process value is changed.

Derivative action weighting

The derivative action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Derivative action is fully effective upon setpoint change
- 0.0: Derivative action is not effective upon setpoint change

The derivative action is always fully effective when the process value is changed.

PID algorithm sampling time

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is calculated during tuning and rounded to a multiple of the PID_3Step sampling time. All other functions of PID_3Step are executed at every call.

Deadband width

The deadband suppresses the noise component in the steady controller state. The deadband width specifies the size of the deadband. The deadband is off if the deadband width is 0.0.

Commissioning PID_3Step V2

Pretuning

The pretuning determines the process response to a pulse of the output value and searches for the point of inflection. The tuned PID parameters are calculated as a function of the maximum slope and dead time of the controlled system. You obtain the best PID parameters when you perform pretuning and fine tuning.

The more stable the process value is, the easier it is to calculate the PID parameters and the more precise the result will be. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher compared to the noise. This is most likely the case in operating modes "Inactive" and "manual mode". The PID parameters are backed up before being recalculated.

The setpoint is frozen during pretuning.

Requirement

- The PID_3Step instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- Reset = FALSE
- The motor transition time has been configured or measured.
- PID_3Step is in one of the following modes: "Inactive", "Manual mode", or "Automatic mode".
- The setpoint and the process value lie within the configured limits (see "Process value settings" configuration).

Procedure

To perform pretuning, follow these steps:

1. Double-click the "PID_3Step > Commissioning" entry in the project tree.
2. Select the entry "Pretuning" in the "Tuning mode" drop-down list in the working area "Tuning".
3. Click the "Start" icon.
   - An online connection will be established.
   - Value recording is started.
   - Pretuning is started.
   - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

Note

Click the "Stop" icon when the progress bar has reached 100% and it is to be assumed the controller tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

Result

If pretuning was performed without an error message, the PID parameters have been tuned. PID_3Step switches to automatic mode and uses the tuned parameters. The tuned PID parameters will be retained during power OFF and a restart of the CPU.

If pretuning is not possible, PID_3Step responds with the configured reaction to errors.

Fine tuning

Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are tuned for the operating point from the amplitude and frequency of this oscillation. All PID parameters are recalculated from the results. PID parameters from fine tuning usually have better master control and disturbance characteristics than PID parameters from pretuning. You obtain the best PID parameters when you perform pretuning and fine tuning.

PID_3Step automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value. The PID parameters are backed up before being recalculated.

The setpoint is frozen during fine tuning.

Requirement

- The PID_3Step instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- Reset = FALSE
- The motor transition time has been configured or measured.

- The setpoint and the process value lie within the configured limits (see "Process value settings" configuration).
- The control loop has stabilized at the operating point. The operating point is reached when the process value corresponds to the setpoint.
- No disturbances are expected.
- PID_3Step is in inactive mode, automatic mode or manual mode.

Process depends on initial situation

Fine tuning proceeds as follows when started from:

- Automatic mode
  Start fine tuning from automatic mode if you wish to improve the existing PID parameters through tuning.
  PID_3Step controls the system using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start.
- Inactive or manual mode
  Pretuning is always started first. The determined PID parameters will be used for control until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start.

Procedure

To perform fine tuning, follow these steps:

1. Select the entry "Fine tuning" in the "Tuning mode" drop-down list.
2. Click the "Start" icon.
   - An online connection will be established.
   - Value recording is started.
   - The process of fine tuning is started.
   - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

Note

Click the "Stop" icon in the "Tuning mode" group when the progress bar has reached 100% and it is to be assumed the controller tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

Result

If no errors occurred during fine tuning, the PID parameters have been tuned. PID_3Step switches to automatic mode and uses the tuned parameters. The tuned PID parameters will be retained during power OFF and a restart of the CPU.

If errors occurred during fine tuning, PID_3Step responds with the configured response to errors.

Commissioning with manual PID parameters

Requirement

- The PID_3Step instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- Reset = FALSE
- The motor transition time has been configured or measured.
- PID_3Step is in "inactive" mode.
- The setpoint and the process value lie within the configured limits (see "Process value settings" configuration).

Procedure

Proceed as follows to commission PID_3Step with manual PID parameters:

1. Double-click on "PID_3Step > Configuration" in the project tree.
2. Click on "Advanced settings > PID Parameters" in the configuration window.
3. Select the check box "Enable direct input".
4. Enter the PID parameters.
5. Double-click the "PID_3Step > Commissioning" entry in the project tree.
6. Establish an online connection to the CPU.
7. Load the PID parameters to the CPU.
8. Click the "Start PID_3Step" icon.

Result

PID_3Step changes to automatic mode and controls using the current PID parameters.

See also

PID parameters (Page 9598)

Measuring the motor transition time

Introduction

PID_3Step requires the motor transition time to be as accurate as possible for good controller results. The data in the actuator documentation contains average values for this type of actuator. The value for the specific actuator used may differ.

You can measure the motor transition time during commissioning if you are using actuators with position feedback or endstop signals. The output value limits are not taken into consideration during the motor transition time measurement. The actuator can travel to the high or the low endstop.

The motor transition time cannot be measured if neither position feedback nor endstop signals are available.

Actuators with analog position feedback

Proceed as follows to measure motor transition time with position feedback:

Requirement

- Feedback or Feedback_PER has been selected in the basic settings and the signal has been connected.
- An online connection to the CPU has been established.

1. Select the "Use position feedback" check box.
2. Enter the position to which the actuator is to be moved in the "Target position" input field. The current position feedback (starting position) will be displayed. The difference between "Target position" and "Position feedback" must be at least 50% of the valid output value range.
3. Click the "Start" icon.

Result

The actuator is moved from the starting position to the target position. Time measurement starts immediately and ends when the actuator reaches the target position. The motor transition time is calculated according to the following equation:

Motor transition time = (output value high limit – output value low limit) × Measuring time / AMOUNT (target position – starting position).

The progress and status of transition time measurement are displayed. The transition time measured is saved in the instance data block on the CPU and displayed in the "Measured transition time" field. When the transition time measurement is ended and ActivateRecoverMode = TRUE, PID_3Step switches to the operating mode from which the

transition time measurement was started. If the transition time measurement is ended and ActivateRecoverMode = FALSE, PID_3Step changes to "Inactive" mode.

Note

Click on the icon "Upload measured transition time" to load the motor transition time measured to the project.

Actuators with endstop signals

Proceed as follows to measure the transition time of actuators with endstop signals:

Requirement

- The "Endstop signals" check box in the basic settings has been selected and Actuator_H and Actuator_L are connected.
- An online connection to the CPU has been established.

Proceed as follows to measure motor transition time with endstop signals:

1. Select the "Use actuator endstop signals" check box.
2. Select the direction in which the actuator is to be moved.
   - Open - Close - Open
     The actuator is moved first to the high endstop, then to the low endstop and then back to the high endstop.
   - Close - Open - Close
     The actuator is moved first to the low endstop, then to the high endstop and then back to the low endstop.
3. Click the "Start" icon.

Result

The actuator is moved in the selected direction. Time measurement will start once the actuator has reached the first endstop and will end when the actuator reaches this endstop for the second time. The motor transition time is equal to the time measured divided by two.

The progress and status of transition time measurement are displayed. The transition time measured is saved in the instance data block on the CPU and displayed in the "Measured transition time" field. When the transition time measurement is ended and ActivateRecoverMode = TRUE, PID_3Step switches to the operating mode from which the transition time measurement was started. If the transition time measurement is ended and ActivateRecoverMode = FALSE, PID_3Step changes to "Inactive" mode.

Cancelling transition time measurement

PID_3Step switches to "Inactive" mode if you cancel transition time measurement by pressing the Stop button.

PID_3Step V1

Configuring PID_3Step V1

Basic settings

Introduction

Configure the following properties of the "PID_3Step" technology object under "Basic settings" in the Inspector window or in the configuration window:

- Physical quantity
- Control logic
- Start-up behavior after reset
- Setpoint (only in the Inspector window)
- Process value (only in the Inspector window)
- Output value (only in the Inspector window)
- Position feedback (only in the Inspector window)

Setpoint, process value, output value and position feedback

You can only configure the setpoint, process value, output value and position feedback in the Inspector window of the programming editor. Select the source for each value:

- Instance DB
  The value saved in the instance DB is used.
  Value must be updated in the instance DB by the user program.
  There should be no value at the instruction.
  Change via HMI possible.
- Instruction
  The value connected to the instruction is used.
  The value is written to the instance DB each time the instruction is called.
  No change via HMI possible.

Controller type

Physical quantity

Select the unit of measurement and physical quantity for the setpoint and process value in the "Controller type" group. The setpoint and process value will be displayed in this unit.

Control logic

An increase of the output value is generally intended to cause an increase in the process value. This is referred to as a normal control logic.

PID_3Step does not work with negative proportional gain. Select the check box "Invert control logic" to reduce the process value with a higher output value.

Examples

- Opening the drain valve will reduce the level of a container's contents.
- Increasing cooling will reduce the temperature.

Start-up behavior after reset

To change straight to the last active mode after restarting the CPU, select the "Enable last mode after CPU restart" check box.

PID_3Step will remain in "Inactive" mode if the check box is cleared.

Setpoint

Procedure

Proceed as follows to define a fixed setpoint:

1. Select "Instance DB".
2. Enter a setpoint, e.g. 80° C.
3. Delete any entry in the instruction.

Proceed as follows to define a variable setpoint:

1. Select "Instruction".
2. Enter the name of the REAL variable in which the setpoint is saved.
   Program-controlled assignment of various values to the REAL variable is possible, for example for the time controlled change of the setpoint.

Process value

PID_3Step will scale the value of the analog input to the physical quantity if you use the analog input value directly.

You will need to write a program for processing if you wish first to process the analog input value. The process value is, for example, not directly proportional to the value at the analog input. The processed process value must be in floating point format.

Procedure

Proceed as follows to use the analog input value without processing:

1. Select the entry "Input_PER" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the address of the analog input.

Proceed as follows to use the processed process value in floating point format:

1. Select the entry "Input" in the drop-down list "Input".
2. Select "Instruction" as source.
3. Enter the name of the variable in which the processed process value is saved.

Position feedback

Position feedback configuration depends upon the actuator used.

- Actuator without position feedback
- Actuator with digital endstop signals
- Actuator with analog position feedback
- Actuator with analog position feedback and endstop signals

Actuator without position feedback

Proceed as follows to configure PID_3Step for an actuator without position feedback:

1. Select the entry "No Feedback" in the drop-down list "Feedback".

Actuator with digital endstop signals

Proceed as follows to configure PID_3Step for an actuator with endstop signals:

1. Select the entry "No Feedback" in the drop-down list "Feedback".
2. Activate the "Actuator endstop signals" check box.
3. Select "Instruction" as source for Actuator_H and Actuator_L.
4. Enter the addresses of the digital inputs for Actuator_H and Actuator_L.

Actuator with analog position feedback

Proceed as follows to configure PID_3Step for an actuator with analog position feedback:

1. Select the entry "Feedback" or "Feedback_PER" in the drop-down list "Feedback".
   - Use the analog input value for Feedback_PER. Configure Feedback_PER scaling in the actuator settings.
   - Process the analog input value for Feedback using your user program.
2. Select "Instruction" as source.
3. Enter the address of the analog input or the variable of your user program.

Actuator with analog position feedback and endstop signals

Proceed as follows to configure PID_3Step for an actuator with analog position feedback and endstop signals:

1. Select the entry "Feedback" or "Feedback_PER" in the drop-down list "Feedback".
2. Select "Instruction" as source.

3. Enter the address of the analog input or the variable of your user program.
4. Activate the "Actuator endstop signals" check box.
5. Select "Instruction" as source for Actuator_H and Actuator_L.
6. Enter the addresses of the digital inputs for Actuator_H and Actuator_L.

Output value

PID_3Step offers an analog output value (Output_PER) and digital output values (Output_UP, Output_DN). Your actuator will determine which output value you use.

- Output_PER
  The actuator is triggered via an analog output and controlled with a continuous signal, e.g. 0...10V, 4...20mA.
- Output_UP, Output_DN
  The actuator is controlled via two digital outputs.

Procedure

Proceed as follows to use the analog output value:

1. Select the entry "Output (analog)" in the drop-down list "Output".
2. Select "Instruction".
3. Enter the address of the analog output.

Proceed as follows to use the digital output value:

1. Select the entry "Output (digital)" in the drop-down list "Output".
2. Select "Instruction" for Output_UP and Output_DN.
3. Enter the addresses of the digital outputs.

Proceed as follows to process the output value using the user program:

1. Select the entry corresponding to the actuator in the drop-down list "Output".
2. Select "Instruction".
3. Enter the name of the variable you are using to process the output value.
4. Transfer the processed output value to the actuator by means of an analog or digital CPU output.

Process value settings

Configure the scaling of your process value and specify the process value absolute limits In the "Process value settings" configuration window.

Scaling the process value

If you have configured the use of Input_PER in the basic settings, you will need to convert the value of the analog input into the physical quantity of the process value. The current configuration will be displayed in the Input_PER display.

Input_PER will be scaled using a low and high value pair if the process value is directly proportional to the value of the analog input.

1. Enter the low pair of values in the "Scaled low process value" and "Low" input fields.
2. Enter the high pair of values in the "Scaled high process value" and "High" input boxes.

Default settings for the value pairs are saved in the hardware configuration. Proceed as follows to use the value pairs from the hardware configuration:

1. Select the instruction PID_3Step in the programming editor.
2. Connect Input_PER to an analog input in the basic settings.
3. Click on the "Automatic setting" button in the process value settings.
   The existing values will be overwritten with the values from the hardware configuration.

Monitoring process value

Specify the absolute high and low limit of the process value. You must enter reasonable limits for your controlled system. Reasonable limits are important during optimization to obtain optimal PID parameters. The default for the "High limit process value" is 120 %. At the I/O input, the process value can be a maximum of 18% higher than the standard range (overrange). This setting ensures that an error is no longer signaled due to a violation of the "Process value high limit". Only a wire-break and a short-circuit are recognized and PID_3Step reacts according to the configured reaction to error.

NOTICE
Your system may be damaged. If you set very high process value limits (for example $-3.4*10^{38}$...$+3.4*10^{38}$), process value monitoring will be disabled. Your system may then be damaged if an error occurs. You need to configure useful process value limits for your controlled system.

Actuator settings

Actuator-specific times

Configure the motor transition time and the minimum ON and OFF times to prevent damage to the actuator. You can find the specifications in the actuator data sheet.

The motor transition time is the time in seconds the motor requires to move the actuator from the closed to the opened state. The maximum time that the actuator is moved in one direction is 110% of the motor transition time. You can measure the motor transition time during commissioning.

If you are using "Output_UP" or "Output_DN", you can reduce the switching frequency with the minimum on and minimum OFF time.

The on or off times calculated are totaled in automatic mode and only become effective when the sum is greater than or equal to the minimum on or OFF time.

A rising edge at Manual_UP or Manual_DN in manual mode will operate the actuator for at least the minimum on or OFF time.

Reaction to error

PID_3Step is preset so that the controller stays active in most cases in the event of an error. If errors occur frequently in controller mode, this default reaction has a negative effect on the control response. In this case, check the Errorbits parameter and eliminate the cause of the error.

PID_3Step generates a programmable output value in response to an error:

- Current value
  PID_3Step is switched off and no longer modifies the actuator position.
- Current value for error while error is pending
  The controller functions of PID_3Step are switched off and the position of the actuator is no longer changed.
  If the following errors occur in automatic mode, PID_3Step returns to automatic mode as soon as the errors are no longer pending.
  - 0002h: Invalid value at Input_PER parameter.
  - 0200h: Invalid value at Input parameter.
  - 0800h: Sampling time error
  - 1000h: Invalid value at Setpoint parameter.
  - 2000h: Invalid value at Feedback_PER parameter.
  - 4000h: Invalid value at Feedback parameter.
  - 8000h: Error during digital position feedback.

  If one of these error occurs in manual mode, PID_3Step remains in manual mode.
  If an error occurs during the tuning or transition time measurement, PID_3Step is switched off.
- Substitute output value
  PID_3Step moves the actuator to the substitute output value and then switches off.
- Substitute output value while error is pending
  PID_3Step moves the actuator to the substitute output value. When the substitute output value is reached, PID_3Step reacts as it does with "Current value for while error is pending".

Enter the substitute output value in "%".

Only substitute output values 0% and 100% can be approached precisely in the case of actuators without analog position feedback. The actuator is moved in one direction at 110% of the motor transition time to ensure the high or low endstop is reached. There endstop signals take priority. A substitute output value not equal to 0% or 100% is approached via an internally simulated position feedback. This procedure does not, however, allow the exact approach of substitute output value.

All substitute output values can be approached precisely with actuators with analog position feedback.

Scaling position feedback

If you have configured the use of Feedback_PER in the basic settings, you will need to convert the value of the analog input into %. The current configuration will be displayed in the "Feedback" display.

Feedback_PER is scaled using a low and high value pair.

1. Enter the low pair of values in the "Low endstop" and "Low" input boxes.
2. Enter the high pair of values in the "High endstop" and "High" input boxes.

"Low endstop" must be less than "High endstop"; "Low" must be less than "High".

The valid values for "High endstop" and "Low endstop" depend upon:

- No Feedback, Feedback, Feedback_PER
- Output (analog), Output (digital)

Output	Feedback	Low endstop	High endstop
Output (digital)	No Feedback	Cannot be set (0.0%)	Cannot be set (100.0%)
Output (digital)	Feedback	-100.0% or 0.0%	0.0% or +100.0%
Output (digital)	Feedback_PER	-100.0% or 0.0%	0.0% or +100.0%
Output (analog)	No Feedback	Cannot be set (0.0%)	Cannot be set (100.0%)
Output (analog)	Feedback	-100.0% or 0.0%	0.0% or +100.0%
Output (analog)	Feedback_PER	-100.0% or 0.0%	0.0% or +100.0%

Limiting the output value

You can only exceed or undershoot the output value limits during the transition time measurement. The output value is limited to these values in all other modes.

Enter the absolute output value limits in the "Output value high limit" and "Output value low limit" input boxes. The output value limits must be within "Low endstop" and "High endstop".

If no Feedback is available and Output (digital) is set, you cannot limit the output value. The digital outputs are reset with Actuator_H = TRUE or Actuator_L = TRUE, or after a travel time amounting to 110% of the motor transition time.

Advanced settings

Monitoring process value

Configure a warning high and low limit for the process value in the "Process value monitoring" configuration window. If one of the warning limits is exceeded or undershot during operation, a warning will be displayed at the PID_3Step instruction:

- At the InputWarning_H output parameter if the warning high limit has been exceeded
- At the InputWarning_L output parameter if the warning low limit has been undershot

The warning limits must be within the process value high and low limits.

The process value high and low limits will be used if you do not enter values.

Example

Process value high limit = 98° C; warning high limit = 90° C

Warning low limit = 10° C; process value low limit = 0° C

PID_3Step will respond as follows:

Process value	InputWarning_H	InputWarning_L	Operating mode
> 98° C	TRUE	FALSE	Inactive
≤ 98° C and > 90° C	TRUE	FALSE	Automatic mode
≤ 90° C and ≥ 10° C	FALSE	FALSE	Automatic mode
< 10° C and ≥ 0° C	FALSE	TRUE	Automatic mode
< 0° C	FALSE	TRUE	Inactive

PID parameters

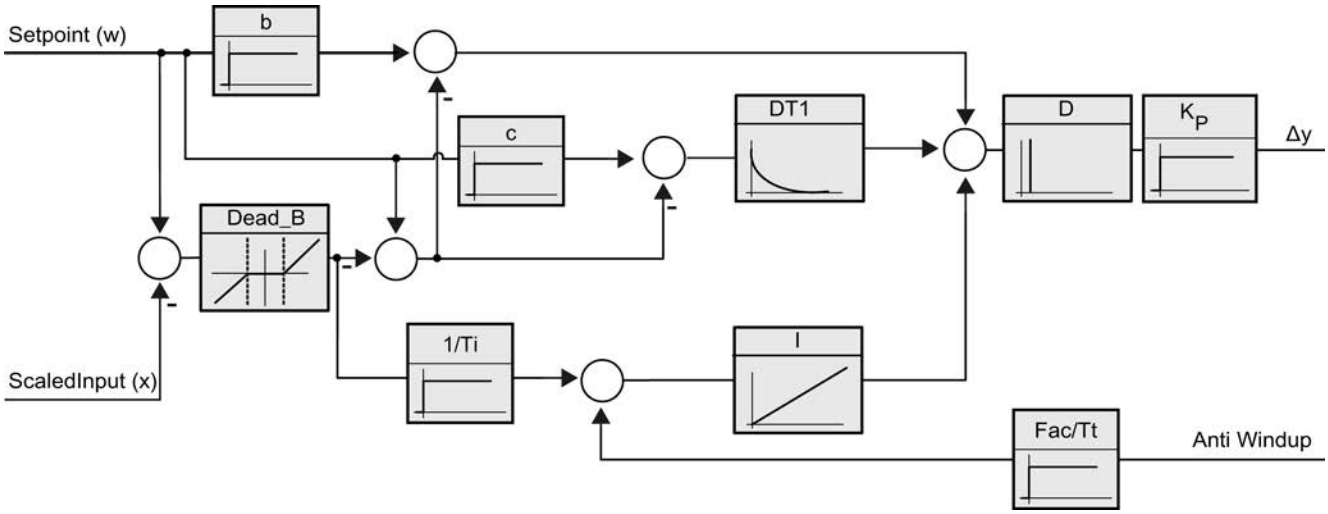
The PID parameters are displayed in the "PID Parameters" configuration window. The PID parameters will be adapted to your controlled system during controller tuning. You do not need to enter the PID parameters manually.

The PID algorithm operates according to the following equation:

$$\Delta y = K_p \cdot s \cdot \left[(b \cdot w - x) + \frac{1}{T_I \cdot s}(w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1}(c \cdot w - x) \right]$$

Symbol	Description
Δy	Output value of the PID algorithm
K_p	Proportional gain
s	Laplace operator
b	Proportional action weighting
w	Setpoint
x	Process value
T_I	Integral action time
a	Derivative delay coefficient (derivative delay T1 = a × T_D)
T_D	Derivative action time
c	Derivative action weighting

The diagram below illustrates the integration of the parameters into the PID algorithm:



All PID parameters are retentive. If you enter the PID parameters manually, you must completely download PID_3Step.

Auto-Hotspot

Proportional gain

The value specifies the proportional gain of the controller. PID_3Step does not work with a negative proportional gain. Control logic is inverted under Basic settings > Controller type.

Integral action time

The integral action time determines the time behavior of the integral action. The integral action is deactivated with integral action time = 0.0.

Derivative action time

The derivative action time determines the time behavior of the derivative action. Derivative action is deactivated with derivative action time = 0.0.

Derivative delay coefficient

The derivative delay coefficient delays the effect of the derivative action.

Derivative delay = derivative action time × derivative delay coefficient

- 0.0: Derivative action is effective for one cycle only and therefore almost not effective.
- 0.5: This value has proved useful in practice for controlled systems with **one** dominant time constant.
- > 1.0: The greater the coefficient, the longer the effect of the derivative action is delayed.

Proportional action weighting

The proportional action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Proportional action for setpoint change is fully effective
- 0.0: Proportional action for setpoint change is not effective

The proportional action is always fully effective when the process value is changed.

Derivative action weighting

The derivative action may weaken with changes to the setpoint.

Values from 0.0 to 1.0 are applicable.

- 1.0: Derivative action is fully effective upon setpoint change
- 0.0: Derivative action is not effective upon setpoint change

The derivative action is always fully effective when the process value is changed.

PID algorithm sampling time

The controlled system needs a certain amount of time to respond to changes in the output value. It is therefore not advisable to calculate the output value in every cycle. The sampling time of the PID algorithm represents the time between two calculations of the output value. It is calculated during tuning and rounded to a multiple of the PID_3Step sampling time. All other functions of PID_3Step are executed at every call.

Deadband width

The deadband suppresses the noise component in the steady controller state. The deadband width specifies the size of the deadband. The deadband is off if the deadband width is 0.0.

See also

Downloading technology objects to device (Page 9536)

Commissioning PID_3Step V1

Commissioning

You can monitor the setpoint, process value and output value over time in the "Tuning" working area. The following commissioning functions are supported in the curve plotter:

- Controller pretuning
- Controller fine tuning
- Monitoring the current closed-loop control in the trend view

All functions require an online connection to the CPU to have been established.

Basic handling

- Select the desired sampling time in the "Sampling time" drop-down list.
  All values in the tuning working area are updated in the selected update time.
- Click the "Start" icon in the measuring group if you want to use the commissioning functions. Value recording is started. The current values for the setpoint, process value and output value are entered in the trend view. Operation of the commissioning window is enabled.
- Click the "Stop" icon if you want to end the commissioning functions.
  The values recorded in the trend view can continue to be analyzed.
- Closing the commissioning window will terminate recording in the trend view and delete the recorded values.

Pretuning

The pretuning determines the process response to a pulse of the output value and searches for the point of inflection. The tuned PID parameters are calculated as a function of the maximum slope and dead time of the controlled system.

The more stable the process value is, the easier it is to calculate the PID parameters and the more precise the result will be. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher compared to the noise. The PID parameters are backed up before being recalculated.

The setpoint is frozen during pretuning.

Requirement

- The PID_3Step instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- PID_3Step is in "inactive" or "manual" mode.
- The setpoint and the process value lie within the configured limits (see "Process value settings" configuration).

Procedure

To perform pretuning, follow these steps:

1. Double-click the "PID_3Step > Commissioning" entry in the project tree.
2. Select the entry "Pretuning" in the "Tuning mode" drop-down list in the working area "Tuning".
3. Click the "Start" icon.
   - An online connection will be established.
   - Value recording is started.
   - Pretuning is started.
   - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

   Note

   Click the "Stop" icon when the progress bar has reached 100% and it is to be assumed the controller tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

Result

If pretuning was performed without an error message, the PID parameters have been tuned. PID_3Step switches to automatic mode and uses the tuned parameters. The tuned PID parameters will be retained during power OFF and a restart of the CPU.

If pretuning is not possible, PID_3Step changes to "Inactive" mode.

Fine tuning

Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are optimized for the operating point from the amplitude and frequency of this oscillation. All PID parameters are recalculated on the basis of the findings. PID parameters from fine tuning usually have better master control and disturbance behavior than PID parameters from pretuning.

PID_3Step automatically attempts to generate an oscillation greater than the noise of the process value. Fine tuning is only minimally influenced by the stability of the process value. The PID parameters are backed up before being recalculated.

The setpoint is frozen during fine tuning.

Requirement

- The PID_3Step instruction is called in a cyclic interrupt OB.
- ManualEnable = FALSE
- The motor transition time has been configured or measured.
- The setpoint and the process value lie within the configured limits (see "Process value settings" configuration).

- The control loop has stabilized at the operating point. The operating point is reached when the process value corresponds to the setpoint.
- No disturbances are expected.
- PID_3Step is in inactive mode, automatic mode or manual mode.

Process depends on initial situation

Fine tuning proceeds as follows when started in:

- Automatic mode
  Start fine tuning in automatic mode if you wish to improve the existing PID parameters using controller tuning.
  PID_3Step will regulate using the existing PID parameters until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start.
- Inactive or manual mode
  Pretuning is always started first. The PID parameters established will be used for adjustment until the control loop has stabilized and the requirements for fine tuning have been met. Only then will fine tuning start.

Procedure

Proceed as follows to carry out "fine tuning":

1. Select the entry "Fine tuning" in the "Tuning mode" drop-down list.
2. Click the "Start" icon.
   - An online connection will be established.
   - Value recording is started.
   - The process of fine tuning is started.
   - The "Status" field displays the current steps and any errors that may have occurred. The progress bar indicates the progress of the current step.

   Note

   Click the "Stop" icon in the "Tuning mode" group when the progress bar has reached 100% and it is to be assumed the controller tuning function is blocked. Check the configuration of the technology object and, if necessary, restart controller tuning.

Result

The PID parameters will have been optimized if fine tuning has been executed without errors. PID_3Step changes to automatic mode and uses the optimized parameters. The optimized PID parameters will be retained during power OFF and a restart of the CPU.

If errors occurred during fine tuning, PID_3Step will change to "inactive" mode.

Commissioning with manual PID parameters

Procedure

Proceed as follows to commission PID_3Step with manual PID parameters:

1. Double-click on "PID_3Step > Configuration" in the project tree.
2. Click on "Advanced settings > PID Parameters" in the configuration window.
3. Select the check box "Enable direct input".
4. Enter the PID parameters.
5. Double-click on "PID_3Step > Commissioning" in the project tree.
6. Establish an online connection to the CPU.
7. Load the PID parameters to the CPU.
8. Click on the "Activate controller" icon.

Result

PID_3Step changes to automatic mode and controls using the current PID parameters.

Measuring the motor transition time

Introduction

PID_3Step requires the motor transition time to be as accurate as possible for good controller results. The data in the actuator documentation contains average values for this type of actuator. The value for the specific actuator used may differ.

You can measure the motor transition time during commissioning if you are using actuators with position feedback or endstop signals. The output value limits are not taken into consideration during the motor transition time measurement. The actuator can travel to the high or the low endstop.

The motor transition time cannot be measured if neither position feedback nor endstop signals are available.

Actuators with analog position feedback

Proceed as follows to measure motor transition time with position feedback:

Requirement

- Feedback or Feedback_PER has been selected in the basic settings and the signal has been connected.
- An online connection to the CPU has been established.

1. Select the "Use position feedback" check box.
2. Enter the position to which the actuator is to be moved in the "Target position" input field. The current position feedback (starting position) will be displayed. The difference between "Target position" and "Position feedback" must be at least 50% of the valid output value range.
3. Click the "Start transition time measurement" icon.

Result

The actuator is moved from the starting position to the target position. Time measurement starts immediately and ends when the actuator reaches the target position. The motor transition time is calculated according to the following equation:

Motor transition time = (output value high limit – output value low limit) × Measuring time / AMOUNT (target position – starting position).

The progress and status of transition time measurement are displayed. The transition time measured is saved in the instance data block on the CPU and displayed in the "Measured transition time" field. PID_3Step will change to "Inactive" mode once transition time measurement is complete.

Note

Click on the icon "Upload measured transition time" to load the motor transition time measured to the project.

Actuators with endstop signals

Proceed as follows to measure the transition time of actuators with endstop signals:

Requirement

- The "Endstop signals" check box in the basic settings has been selected and Actuator_H and Actuator_L are connected.
- An online connection to the CPU has been established.

Proceed as follows to measure motor transition time with endstop signals:

1. Select the "Use actuator endstop signals" check box.
2. Select the direction in which the actuator is to be moved.
   - Open - Close - Open
     The actuator is moved first to the high endstop, then to the low endstop and then back to the high endstop.
   - Close - Open - Close
     The actuator is moved first to the low endstop, then to the high endstop and then back to the low endstop.
3. Click the "Start transition time measurement" icon.

Result

The actuator is moved in the selected direction. Time measurement will start once the actuator has reached the first endstop and will end when the actuator reaches this endstop for the second time. The motor transition time is equal to the time measured divided by two.

The progress and status of transition time measurement are displayed. The transition time measured is saved in the instance data block on the CPU and displayed in the "Measured transition time" field. PID_3Step will change to "Inactive" mode once transition time measurement is complete.

Cancelling transition time measurement

PID_3Step will change to "Inactive" mode immediately if you cancel transition time measurement. The actuator will stop being moved. You can reactive PID-3Step in the curve plotter.

11.3.4 PID control (S7-300, S7-400, S7-1500)

11.3.4.1 Using PID basic functions

CONT_C

Technology object CONT_C

The technology object CONT_C provides a continual PID-controller for automatic and manual mode. It corresponds to the instance data block of the instruction CONT_C. You can configure a pulse controller using the PULSEGEN instruction.

The proportional, integral (INT) and differential components (DIF) are switched parallel to each other and can be turned on and off individually. With this, P-, I, PI-, PD- and PID-controller can be set.

S7-1500
All parameters and tags of the technology object are retentive and can only be changed during download to the device if you completely download CONT_C.

See also

Overview of software controller (Page 9531)

Add technology objects (Page 9533)

Configure technology objects (Page 9534)

CONT_C (Page 6793)

Downloading technology objects to device (Page 9536)

Configure controller difference CONT_C

Use process value periphery

To use the process value in the periphery format at the PV_PER input parameter, follow these steps:

1. Select the "Enable I/O" check box.
2. If the process value is available as a physical size, enter the factor and offset for the scaling in percent.
   The process value is then determined according to the following formula:
   PV = PV_PER × PV_FAC + PV_OFF

Use internal process values

To use the process value in the floating-point format at the PV_IN input parameter, follow these steps:

1. Clear the "Enable I/O" check box.

Control deviation

Set a dead zone range under the following requirement:

- The process value signal is noisy.
- The controller gain is high.
- The derivative action is activated.

The noise component of the process value causes strong deviations of the output value in this case. The dead zone suppresses the noise component in the steady controller state. The dead zone range specifies the size of the dead zone. With a dead zone range of 0.0, the dead zone is turned off.

See also

How CONT_C works (Page 6794)

Configure the controller algorithm CONT_C

General

To determine which components of the control algorithm are activated, proceed as follows:

1. Select an entry from the "Controller structure" list.
   You can only specify required parameters for the selected controller structure.

Proportional action

1. If the controller structure contains a proportional action, enter the "proportional gain".

Integral action

1. If the controller structure contains an integral action, enter the integral action time.
2. To give the integral action an initialization value, select the check box "Initialize integral action" and enter the initialization value.
3. In order to permanently set the integral action to this initialization value, select the "Integral action hold" check box.

Derivative action

1. If the controller structure contains a derivative action, enter the derivative action time, the derivative action weighting and the delay time.

See also

How CONT_C works (Page 6794)

Configure the output value CONT_C

General

You can set CONT_C in the manual or automatic mode.

1. To set a manual manipulated value, activate the option "Activate manual mode" option check box.
   You can specify a manual manipulated value on the input parameter MAN.

Manipulated value limits

The manipulated value is limited at the top and bottom so that it can only accept valid values. You cannot turn off the limitation. Exceeding the limits is displayed through the output parameters QLMN_HLM and QLMN_LLM.

1. Enter a value for the high and low manipulated value limits.
   If the manipulated value is a physical size, the units for the high and low manipulated value limits must match.

Scaling

The manipulated value can be scaled for output as a floating point and periphery value through a factor and an offset according to the following formula.

Scaled manipulated value = manipulated value x factor + offset

Default is a factor of 1.0 and an offset of 0.0.

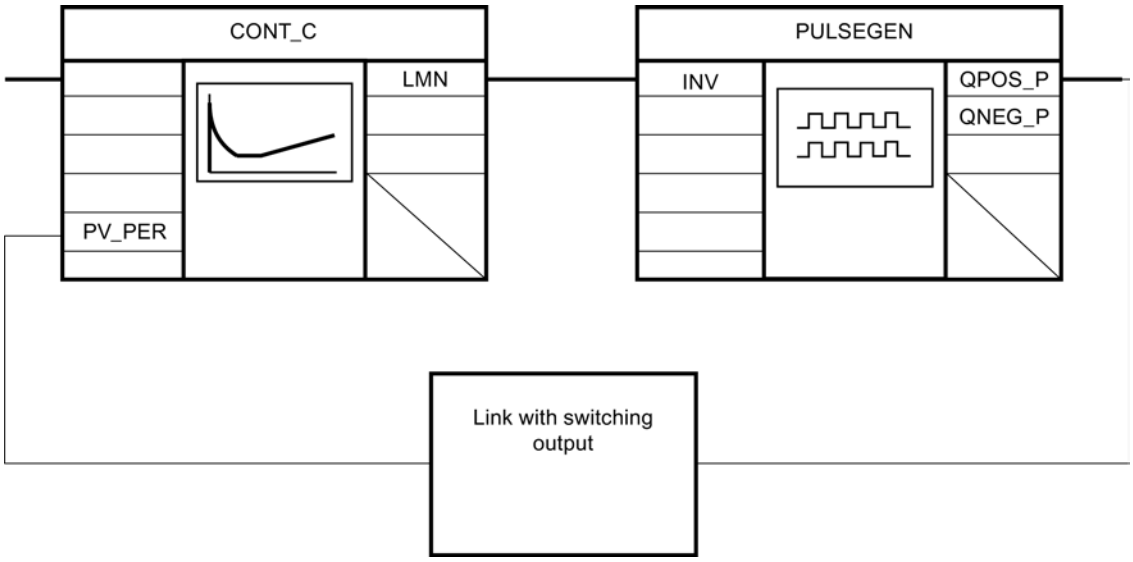
1. Enter a value for the factor and offset.

See also

How CONT_C works (Page 6794)

Programming a pulse controller

With the continuous controller CONT_C and the pulse shaper PULSEGEN, you can implement a fixed setpoint controller with a switching output for proportional actuators. The following figure shows the signal flow of the control loop.



The continuous controller CONT_C forms the output value LMN that is converted by the pulse shaper PULSEGEN into pulse/break signals QPOS_P or QNEG_P.

See also

PULSEGEN (Page 6803)

Commissioning CONT_C

Requirements

- The instruction and the technology object are loaded on the CPU.

Procedure

In order to manually determine the optimal PID parameter, proceed as follows:

1. Click the "Start" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value and output value are recorded.
2. Enter new PID parameters in the "P", "I", "D" and "Delay time" fields.
3. Click on the icon "Send parameter to CPU" in the "Tuning" group.
4. Select the "Change setpoint" check box in the "Current values" group.
5. Enter a new setpoint and click in the "Current Values" group on the icon .

6. Clear the "Manual mode" check box.
   The controller works with the new PID parameters and controls the new setpoint.
7. Check the quality of the PID parameter to check the curve points.
8. Repeat steps 2 to 6 until you are satisfied with the controller results.

CONT_S

Technology object CONT_S

The technology object CONT_S provides a step controller for actuators with integrating behavior and is used to control technical temperature processes with binary output value output signals. The technology object corresponds to the instance data block of the CONT_S instruction. The operating principle is based on the PI control algorithm of the sampling controller. The step controller operates without a position feedback signal. Both manual and automatic mode are possible.

S7-1500
All parameters and tags of the technology object are retentive and can only be changed during download to the device if you completely download CONT_S.

See also

Overview of software controller (Page 9531)

Add technology objects (Page 9533)

Configure technology objects (Page 9534)

CONT_S (Page 6799)

Downloading technology objects to device (Page 9536)

Configure controller difference CONT_S

Use process value periphery

To use the process value in the periphery format at the PV_PER input parameter, follow these steps:

1. Select the "Enable I/O" check box.
2. If the process value is available as a physical quantity, enter the factor and offset for the scaling in percent.
   The process value is then determined according to the following formula:
   PV = PV_PER × PV_FAC + PV_OFF

Use internal process values

To use the process value in the floating-point format at the PV_IN input parameter, follow these steps:

1. Clear the "Enable I/O" check box.

Control deviation

Set a deadband range under the following requirement:

- The process value signal is noisy.
- The controller gain is high.
- The derivative action is activated.

The noise component of the process value causes strong deviations of the manipulated variable in this case. The deadband suppresses the noise component in the steady controller state. The deadband range specifies the size of the deadband. With a deadband range of 0.0, the deadband is turned off.

See also

Mode of operation CONT_S (Page 6799)

Configuring control algorithm CONT_S

PID algorithm

1. Enter the "proportional amplification" for the P-component.
2. Enter the integration time for the time behavior of the I-component.
   With an integration time of 0.0, the I-component is switched off.

See also

Mode of operation CONT_S (Page 6799)

Configure manipulated value CONT_S

General

You can set CONT_S in the manual or automatic mode.

1. To set a manual manipulated value, activate the "Activate manual mode" option check box.
   Enter a manual manipulated value for the input parameters LMNUP and LMNDN.

Pulse generator

1. Enter the minimum impulse duration and minimum pause duration.
   The values must be greater than or equal to the cycle time for the input parameter CYCLE.
   The frequency of operation is reduced through this.
2. Enter the motor setting time.
   The value must be greater than or equal to the cycle time of the input parameter CYCLE.

See also

Mode of operation CONT_S (Page 6799)

Commissioning CONT_S

Requirements

- The instruction and the technology object have been loaded to the CPU.

Procedure

To manually determine the optimal PID parameters, proceed as follows:

1. Click the "Start" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value and output value are recorded.
2. In the fields "P" and "I", enter a new proportional value and a new integration time.
3. Click on the icon "Send parameter to CPU" in the "Tuning" group.
4. Select the "Change setpoint" check box in the "Current values" group.
5. Enter a new setpoint and click in the "Current Values" group on the icon .
6. Clear the "Manual mode" check box.
   The controller works with the new parameters and controls the new setpoint.
7. Check the quality of the PID parameter to check the curve points.
8. Repeat steps 2 to 6 until you are satisfied with the controller results.

TCONT_CP

Technology object TCONT_CP

The technology object TCONT_CP provides a continual temperature controller with pulse generator. It corresponds to the instance data block of the instruction TCONT_CP. The operation is based on the PID control algorithm of the sampling controller. Both manual and automatic mode are possible.

The instruction TCONT_CP calculates the proportional, integral and derivative parameters for your controlled system during pretuning. "Fine tuning" can be used to tune the parameters further. You can also enter the PID parameters manually.

S7-1500
All parameters and tags of the technology object are retentive and can only be changed during download to the device if you completely download TCONT_CP.

See also

Overview of software controller (Page 9531)

Add technology objects (Page 9533)

Configure technology objects (Page 9534)

TCONT_CP (Page 6812)

Downloading technology objects to device (Page 9536)

Configure TCONT_CP

Controller difference

Use process value periphery

To use the input parameter PV_PER, proceed as follows:

1. Select the entry "Periphery" from the "Source" list.
2. Select the "sensor type".
   Depending on the sensor type, the process value is scaled according to different formulas.
   - Standard
     Thermoelements; PT100/NI100
     PV = 0.1 × PV_PER × PV_FAC + PV_OFFS
   - Cooling;
     PT100/NI100
     PV = 0.01 × PV_PER × PV_FAC + PV_OFFS
   - Current/voltage
     PV = 100/27648 × PV_PER × PV_FAC + PV_OFFS
3. Enter the factor and offset for the scaling of the process value periphery.

Use internal process values

To use the input parameter PV_IN, proceed as follows:

1. Select the entry "Internal" from the "Source" list.

Control deviation

Set a deadband range under the following requirement:

- The process value signal is noisy.
- The controller gain is high.
- The derivative action is activated.

The noise component of the process value causes strong deviations of the manipulated variable in this case. The deadband suppresses the noise component in the steady controller state. The deadband range specifies the size of the deadband. With a deadband range of 0.0, the deadband is turned off.

See also

Mode of operation TCONT_CP (Page 6813)

Controlling algorithm

General

1. Enter the "Sampling time PID algorithm".
   A controller sampling time should not exceed 10 % of the determined integratl action time of the controller (TI).
2. If the controller structure contains a proportional action, enter the "proportional gain".
   A negative proportional gain inverts the rule meaning.

Proportional action

For changes of the setpoint, it may lead to overshooting of the proportional action. Through the weighting of the proportional action, you can select how strongly the proportional action

should react when setpoint changes are made. The weakening of the proportional action is reached through a compensation of the integral action.

1. To weaken the proportional action for setpoint changes, enter a "Proportional action weighting".
   - 1.0: Proportional action for setpoint change is fully effective
   - 0.0: Proportional action for setpoint change is not effective

Integral action

With a limitation of the manipulated value, the integral action is stopped. With a control deviation that moves the integral action in the direction of an internal setting range, the integral action is released again.

1. If the controller structure contains an integral action, enter the "integral action time".
   With an integral action time of 0.0, the integral action is switched off.
2. To give the integral action an initialization value, select the "Initialize integral action" check box and enter the "Initialization value".
   Upon restart or COM_RST = TRUE, the integral action is set to this value.

Derivative action

1. If the controller structure contains a derivative action, enter the derivative action time (TD) and the coefficients DT1 (D_F).
   With switched derivative action, the following equation should be maintained:
   TD = 0.5 × CYCLE× D_F.
   The delay time is calculated from this according to the formula:
   delay time = TD/D_F

Set PD-controller with operating point

1. Enter the integral action time 0.0.
2. Activate the "Initialize integral action" check box.
3. Enter the operating point as the initialization value.

Set P-controller with operating point

1. Set a PD-controller with an operating point.
2. Enter the derivative action time 0.0.
   The derivative action is disabled.

Control zone

The control zone limits the value range of the control deviation. If the control deviation is outside of this value range, the manipulated value limits are used.

With an occurrence in the control zone, the derivative action leads to a very quick reduction of the manipulated variable. Thus, the control zone only makes sense for switched on derivative

actions. Without control zone, only the reducing proportional action would reduce the manipulated value. The control zone leads to a quick oscillation without over/under shooting if the emitted minimum or maximum manipulated values are removed from the manipulated value required for the new operating point.

1. Activate the "Activate" check box in the "control zone" group.
2. Enter a setpoint value in the "Width" input field from which the process value may deviate above or below.

See also

Mode of operation TCONT_CP (Page 6813)

Manipulated value continual controller

Manipulated value limits

The manipulated value is limited at the top and bottom so that it can only accept valid values. You cannot turn off the limitation. Exceeding the limits is displayed through the output parameters QLMN_HLM and QLMN_LLM.

1. Enter a value for the high and low manipulated value limits.

Scaling

The manipulated value can be scaled for output as a floating point and periphery value through a factor and an offset according to the following formula.

Scaled manipulated value = manipulated value x factor + offset

Default is a factor of 1.0 and an offset of 0.0.

1. Enter a value for the factor and offset.

Pulse generator

The pulse generator must be turned on for a continual controller.

1. Disable the "Activate" option check box in the "Pulse generator" group.

See also

Mode of operation TCONT_CP (Page 6813)

Manipulated value pulse controller

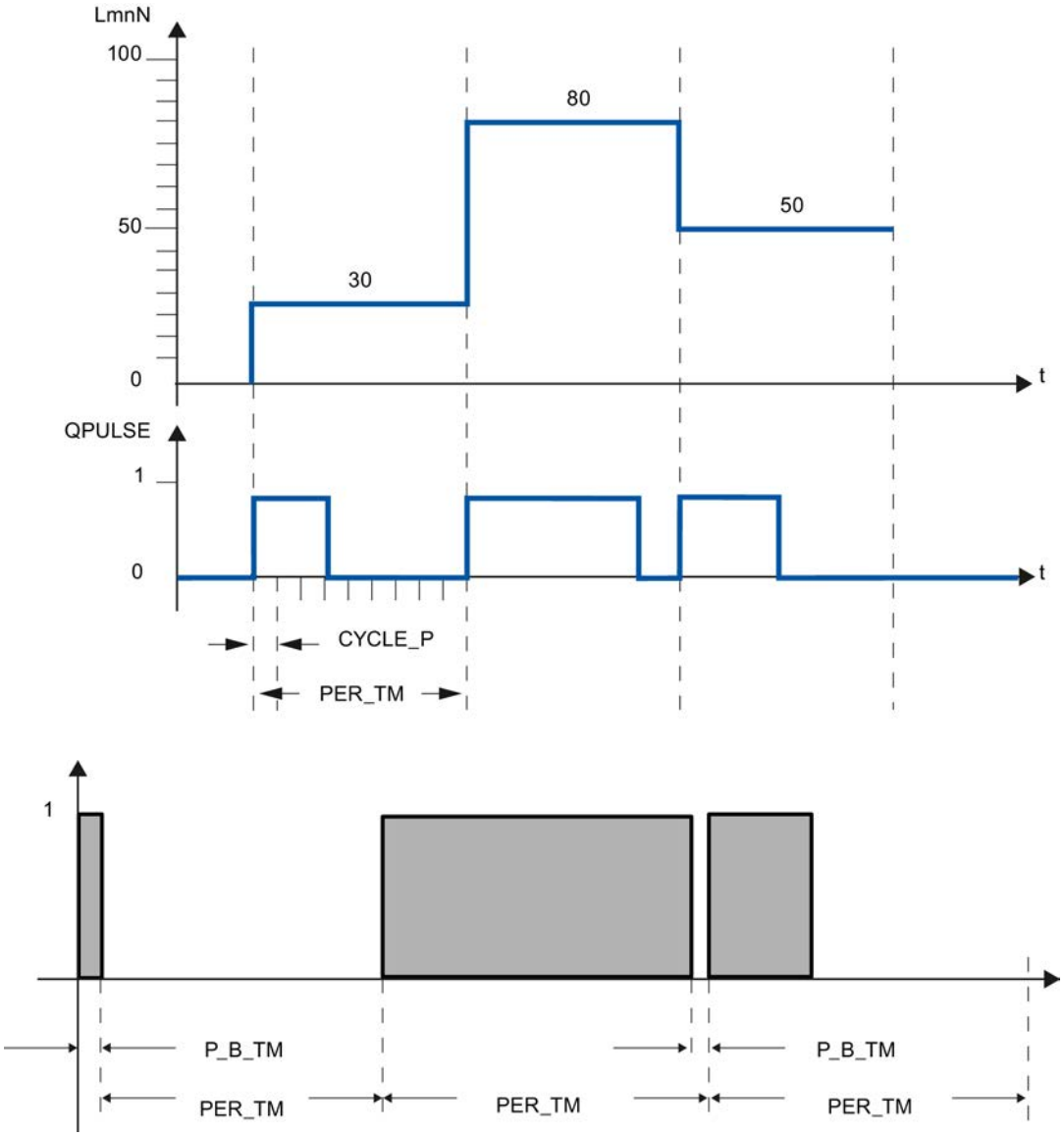
Pulse generator

The analog manipulated value (LmnN) can be emitted through pulse-duration modulation on the output parameter QPULSE as an impulse sequence.

To use the pulse generator, proceed as follows:

1. Activate the "Activate" option check box in the "pulse generator" group.
2. Enter the "sampling time pulse generator", the "minimum impulse/break duration" and the "period duration".

The following graphics clarify the connection between the "sampling pulse generator" (CYCLE_P), the "minimum impulse/break duration" (P_B_TM) and the "period duration" (PER_TM):



Sampling time pulse generator

The sampling time pulse generator must agree with the time tact of the cyclic interrupt OB being called. The duration of the created impulse is always a whole number factor of this value.

For an adequately precise manipulated value resolution, the following relationship should apply:
CYCLE_P ≤ PER_TM/50

Minimum impulse/break duration

Through the minimum impulse/break duration, short on or off times on the actuator are avoided. An impulse smaller than P_B_TM is suppressed.

Recommended are values P_B_TM ≤ 0.1 × PER_TM.

Period duration

The period duration should not exceed 20% of the determined integration time of the controller (TI):
PER_TM ≤ TI/5

Example for the effect of the parameter CYCLE_P, CYCLE and PER_TM:

Period duration PER_TM = 10 s

Sampling time PID-algorithm CYCLE = 1 s

Sampling time pulse generator CYCLE_P = 100 ms.

Every second, a new manipulated value, every 100 ms the comparison of the manipulated value occurs with the previously emitted impulse length and break length.

- If an impulse is emitted, there are 2 possibilities:
  - The calculated manipulated value is larger than the previous impulse length/PER_TM. Then the impulse is extended.
  - The calculated manipulated value is less than or equal to the previous impulse length/ PER_TM. Then no impulse signal will be emitted.
- If no impulse is emitted, there are also 2 possibilities:
  - The value (100 % - calculated manipulated value) is greater than the previous break length / PER_TM. Then the break is extended.
  - The value (100 % - calculated manipulated value) is less than or equal to the previous break length / PER_TM. Then an impulse signal will be emitted.

See also

Mode of operation TCONT_CP (Page 6813)

Operating principle of the pulse generator (Page 6822)

Commissioning TCONT_CP

Optimization of TCONT_CP

Application possibilities

The controller optimization for heating or cooling processes from process type I is applicable. But you can use the block for processes with higher levels like process type II or III.

The PI/PID parameters are automatically determined and set. The controller draft is designed for an optimal disruption behavior The "precise" parameters resulting from this lead to overshooting of 10% to 40% of the jump height for setpoint jump heights.

Phases of controller optimization

For the controller optimization, individual phases are run through, which you can read on the parameter PHASE .

PHASE = 0

No tuning is running. TCONT_CP works in automatic or manual mode.

During PHASE = 0, you can make sure that the controlled system fulfills the requirements for an optimization.

At the end of the optimization, TCONT_CP changes back into PHASE = 0.

PHASE = 1

TCONT_CP is prepared for optimization. PHASE = 1 may only be started if the requirements for an optimization are fulfilled.

During PHASE = 1, the following values are determined:

- Process value noise NOISE_PV
- Initial slope PVDT0
- Average of the manipulated variable
- Sampling time PID algorithm CYCLE
- Sampling time pulse generator CYCLE_P

PHASE = 2

In phase 2, the process value attempts to detect the point of inflection with a constant manipulated variable. This method prevents the point of inflection from being found too early as a result of process variable noise.

With the pulse controller, the process variable is averaged over N pulse cycles and then made available to the controller stage. There is a further averaging of the process variable in the controller stage: Initially, this averaging is inactive; in other words, averaging always takes

place over 1 cycle. As long as the noise exceeds a certain level, the number of cycles is doubled.

The period and amplitude of the noise are calculated. The search for the point of inflection is canceled and phase 2 is exited only when the gradient is always smaller than the maximum rise during the estimated period. TU and T_P_INF are, however, calculated at the actual point of inflection.

Tuning, however, is only ended when the following two conditions are met:

1. The process value is more than 2*NOISE_PV away from the point of inflection.
2. The process value has exceeded the point of inflection by 20%.

Note

When exciting the process using a setpoint step change, tuning is ended at the latest when the process value exceeds 75% of the setpoint step change (SP_INT-PV0) (see below).

PHASE = 3, 4, 5

The phases 3, 4 and 5 last 1 cycle each.

In Phase 3, the valid PI/PID parameters are saved before the optimization and the process parameter is calculated.

In Phase 4, the new PI/PID parameters are calculated.

In Phase 5, the new manipulated variable is calculated and the controlled system is given.

PHASE = 7

The process type is inspected in Phase 7, because TCONT_CP always changes to automatic mode after optimization. The automatic mode starts with LMN = LMN0 + 0.75*TUN_DLMN as a manipulated variable. The testing of the process type occurs **in the automatic mode** with the recently recalculated controller parameters and ends at the latest 0.35*TA (equilibrium time) after the point of inflection. If the process order deviates strongly from the estimated value, the controller parameters are newly calculated and STATUS_D is counted up by 1, otherwise, the controller parameters remain unchanged.

Then the optimization mode is complete and TCONT_CP is back in PHASE = 0. At the STATUS_H parameter, you can identify whether the tuning was successfully completed.

Premature cancellation of the optimization

In Phase 1, 2 or 3, you can cancel the optimization by resetting TUN_ON = FALSE without calculating new parameters. The controller starts in the automatic mode with LMN = LMN0 + TUN_DLMN. If the controller was in manual mode before the tuning, the old manual manipulated variable is output.

If the tuning is canceled in Phase 4, 5 or 7 with TUN_ON = FALSE, the determined controlled parameters are contained until then.

Requirements for an optimization

Transient response

The process must have a stable, asymptotic transient response with time lag.

The process value must settle to steady state after a step change of the manipulated variable. This therefore excludes processes that already show an oscillating response without control, as well as processes with no recovery (integrator in the control system).

WARNING

This may result in death, severe injury or considerable property damage.

During an tuning, the parameter MAN_ON is ineffective. Through this, the output value or process value may take on undesired - even extreme - values.

The output value is defined through the tuning. To cancel the tuning, you first have to set TUN_ON = FALSE. This makes MAN_ON effective again.

Guaranteeing a stationary initial state (phase 0)

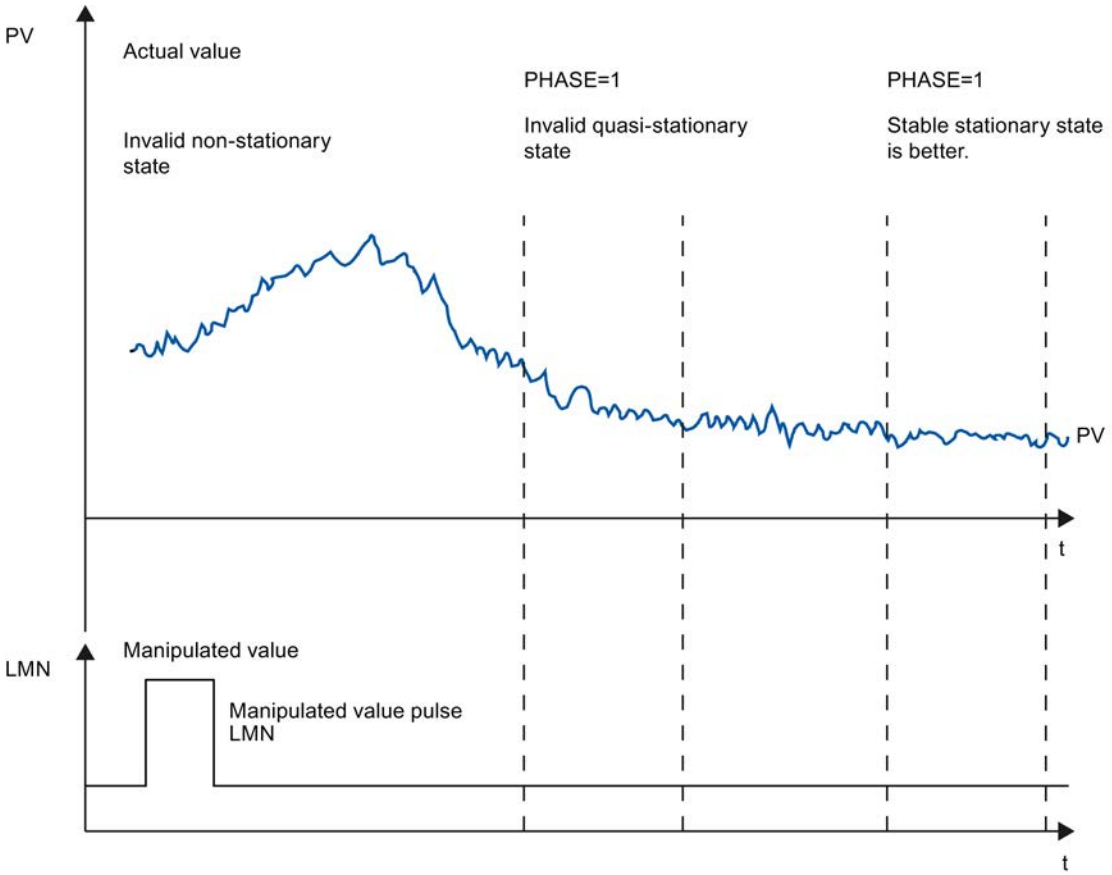
With lower-frequency oscillations of the process value, for example, due to incorrect controller parameters, the controller must be put in manual mode before the tuning is started and wait for the oscillation to stop. Alternatively, you could switch to a "soft" set PI controller (small loop gain large integration time).

Now you have to wait until the stationary state is reached, this means, until the process value and output value have a steady state. It is also permissible to have an asymptotic transient oscillation or slow drifting of the process value (stationary state, see the following image). The output value must be constant or fluctuate by a constant average.

Note

Avoid changing the manipulated variable shortly before starting the tuning. A change of the manipulated variable can occur in an unintended manner through the establishment of the test conditions (for example, closing an oven door)! If this does happen, you have to at least wait until the process value has an asymptotic transient oscillation in a stationary state again. Better controller parameters can be reached if you wait until the transient effect has completely subsided.

In the following image, the transient oscillation is illustrated in the stationary state:



Linearity and operating range

The process response must be linear across the operating range. Non-linear response occurs, for example, when an aggregation state changes. Tuning must take place in a linear part of the operating range.

This means, during tuning and normal control operation non-linear effects within the operating range must be insignificant. It is, however possible to retune the process when the operating point changes, providing tuning is repeated in the close vicinity of the new operating point and non-linearity does not occur during tuning.

If a specific static non-linearity (e.g., valve characteristics) is known, it is always advisable to compensate this with a polyline to linearize the process response.

Disturbance in temperature processes

Disturbances such as the transfer of heat to neighboring zones must not affect the overall temperature process too much. For example, when optimizing the zones of an extruder, all zones must be heated simultaneously.

Possibilities for optimization

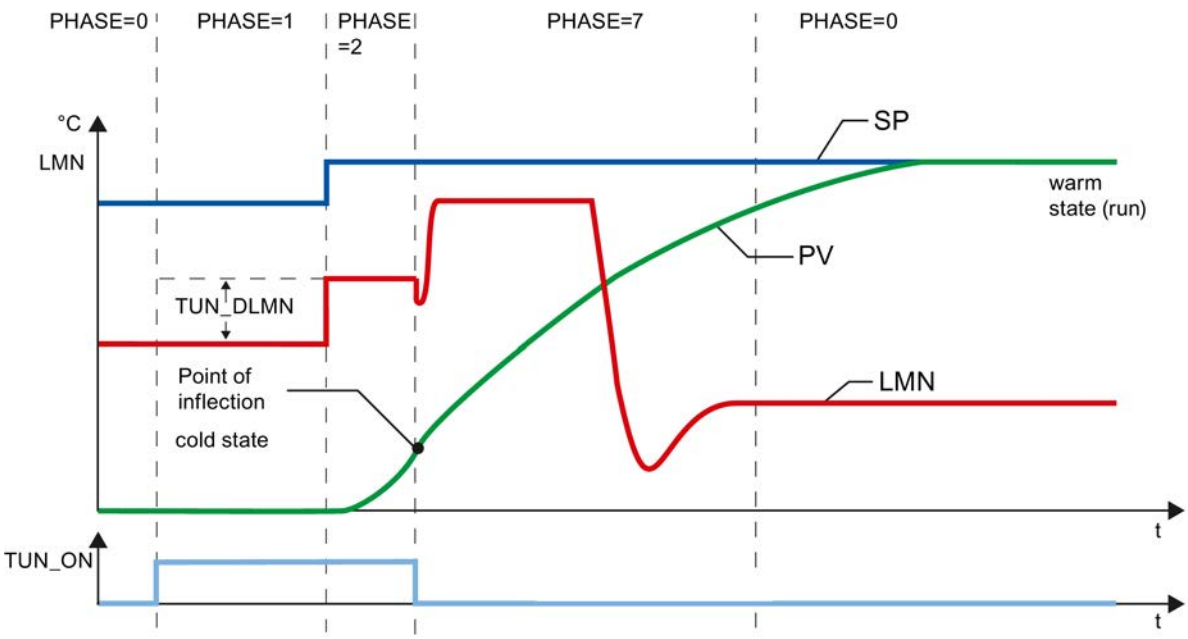
The following possibilities for tuning exist:

- Pretuning
- Fine tuning
- Manual fine-tuning in control mode

Pretuning

During this tuning, the working point is approached from the cold state through a setpoint jump.

With TUN_ON = TRUE, you can establish the tuning readiness. The controller switches from PHASE = 0 to PHASE = 1.



The tuning manipulated variable (LMN0 + TUN_DLMN) is activated by a setpoint change (transition phase 1 -> 2). The setpoint is not effective until the inflection point has been reached (automatic mode is not enabled until this point is reached).

The user is responsible for defining the output excitation delta (TUN_DLMN) according to the permitted process value change. The sign of TUN_DLMN must be set depending on the intended process value change (take into account the direction in which the control is operating).

The setpoint step change and TUN_DLMN must be suitably matched. If the value of TUN_DLMN is too high, there is a risk that the point of inflection will not be found before 75% of the setpoint step change is reached.

TUN_DLMN must nonetheless be high enough to ensure that the process value reaches at least 22 % of the setpoint step change. Otherwise, the process will remain in tuning mode (phase 2).

Remedy: Reduce the setpoint value during the inflection point search.

Note

If processes are extremely sluggish, it is advisable during tuning to specify a target setpoint that is somewhat lower than the desired operating point and to monitor the status bits and PV closely (risk of overshooting).

Tuning only in the linear range:

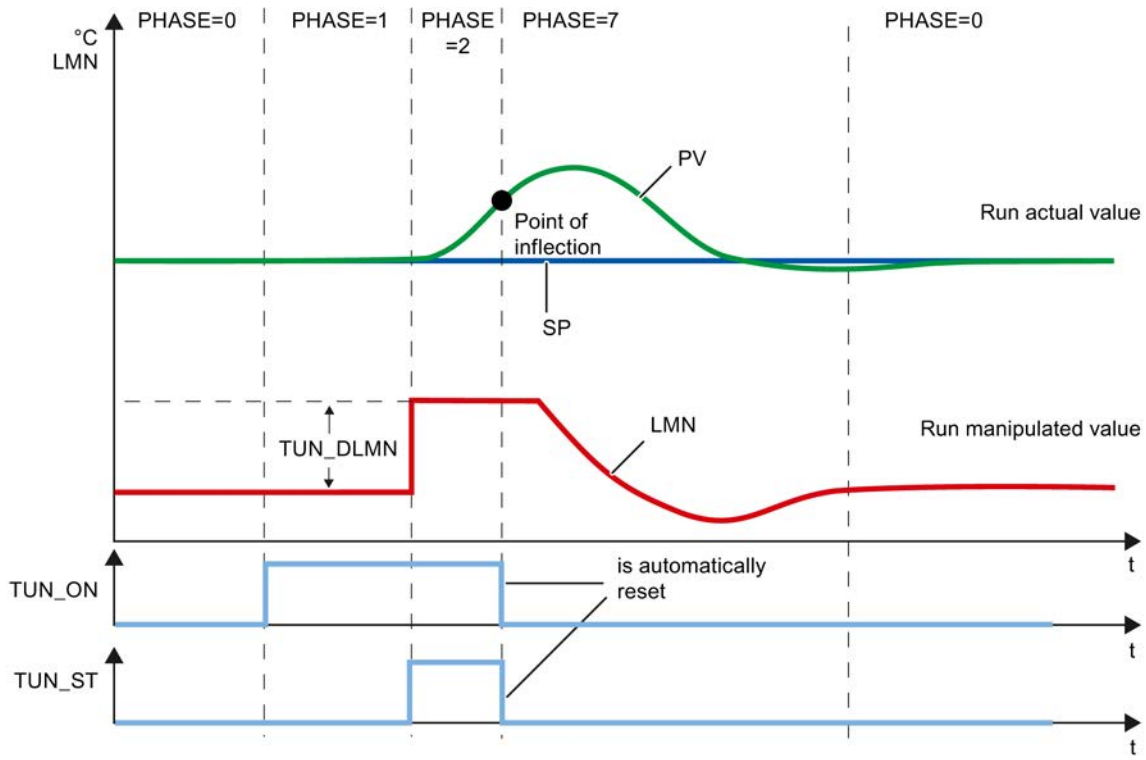
The signals of certain processes (e.g., zinc or magnesium smelters) will pass a non-linear area at the approach of the operating range (change in the state of aggregation).

By selecting a suitable setpoint step change, tuning can be limited to the linear range. When the process value has passed 75% of the setpoint step change (SP_INT-PV0), tuning is ended.

At the same time, TUN_DLMN should be reduced to the extent that the point of inflection is guaranteed to be found before 75% of the setpoint step change is reached.

Fine tuning

During this tuning, the process with a constant setpoint is activated through a output value jump.



The tuning manipulated variable (LMN0 + TUN_DLMN) is activated by setting the start bit TUN_ST (transition from phase 1 -> 2). When you modify the setpoint value, the new value

will not take effect until the point of inflection has been reached (automatic mode will not be enabled until this point has been reached).

The user is responsible for defining the output excitation delta (TUN_DLMN) according to the permitted process value change. The sign of TUN_DLMN must be set depending on the intended process value change (take into account the direction in which the control is operating).

NOTICE
Safety off at 75% is not available when you excite the process via TUN_ST. Tuning is ended when the point of inflection is reached. However, in noisy processes the point of inflection may be significantly exceeded.

Manual fine-tuning in control mode

The following measures can be employed to achieve an overshoot-free setpoint response:

- Adapting the control zone
- Optimize command action
- Attenuation of control parameters
- Modifying control parameters

Tuning result

The left cipher of STATUS_H displays the tuning status

STATUS_H	Result
0	Default, i.e., new controller parameters have not (yet) been found.
10000	Suitable control parameters found.
2xxxx	Control parameters have been found via estimated values; check the control response or check the STATUS_H diagnostic message and repeat controller tuning.
3xxxx	An operator error has occurred; check the STATUS_H diagnostic message and repeat controller tuning.

The CYCLE and CYCLE_P sampling times were already checked in phase 1.

The following controller parameters are updated on TCONT_CP:

- P (proportional GAIN)
- I (integration time TI)
- D (derivative time TD)
- Weighting of the proportional action PFAC_SP
- Coefficient DT1 (D_F)
- Control zone on/off CONZ_ON
- Control zone width CON_ZONE

The control zone is only activated if the process type is suitable (process type I and II) and a PID controller is used (CONZ_ON = TRUE).

Depending on PID_ON, control is implemented either with a PI or a PID controller. The old controller parameters are saved and can be retrieved with UNDO_PAR. A PI parameter record and a PID parameter record are saved additionally in the PI_CON and PID_CON structures. Using LOAD_PID and making a suitable setting for PID_ON, it is also possible to switch later between the tuned PI or PID parameters.

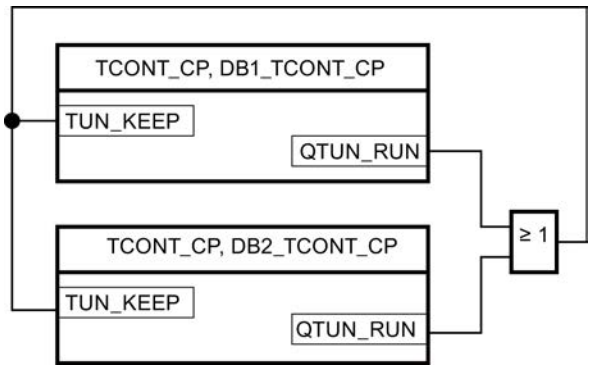
Parallel tuning of controller channels

Adjacent zones (strong heat coupling)

If two or more controllers are controlling the temperature, on a plate, for example (in other words, there are two heaters and two measured process values with strong heat coupling), proceed as follows:

1. OR the two outputs QTUN_RUN.
2. Interconnect each TUN_KEEP input with the output of the OR element.
3. Start both controllers by specifying a setpoint step change at the same time or by setting TUN_ST at the same time.

The following schematic illustrates the parallel tuning of controller channels.



Advantage:

Both controllers output LMN0 + TUN_DLMN until both controllers have left phase 2. This prevents the controller that completes tuning first from falsifying the tuning result of the other controller due to the change in its manipulated variable.

NOTICE
Reaching 75% of the setpoint step change causes an exiting of phase 2 and resetting of output QTUN_RUN. However, automatic mode does not start until TUN_KEEP is also 0.

Adjacent zones (weak heat coupling)

In general terms, tuning should be carried out to reflect the way in which the controller will operate subsequently. If zones are operated together during production such that the temperature differences between the zones remain the same, the temperature of the adjacent zones ought to be increased accordingly during tuning.

Differences in temperature at the beginning of the tuning are irrelevant since they will be compensated by the initial heating (-> initial rise = 0).

Fault descriptions and corrective measures

Compensating operator errors

Operator error	STATUS and action	Comment
TUN_ON and setpoint step change or TUN_ST are set simultaneously	Transition to phase 1; however, tuning is not started. • SP_INT = SP_{old} or • TUN_ST = FALSE	The setpoint change is canceled. This prevents the controller from settling to the new setpoint value and from leaving the stationary operating point unnecessarily.
Effective TUN_DLMN < 5% (end of phase 1)	STATUS_H = 30002 • Transition to phase 0 • TUN_ON = FALSE • SP = SP_{old}	Tuning is canceled. The setpoint change is canceled. This prevents the controller from settling to the new setpoint value and from leaving the stationary operating point unnecessarily.

Point of inflection not reached (only if excited by setpoint step change)

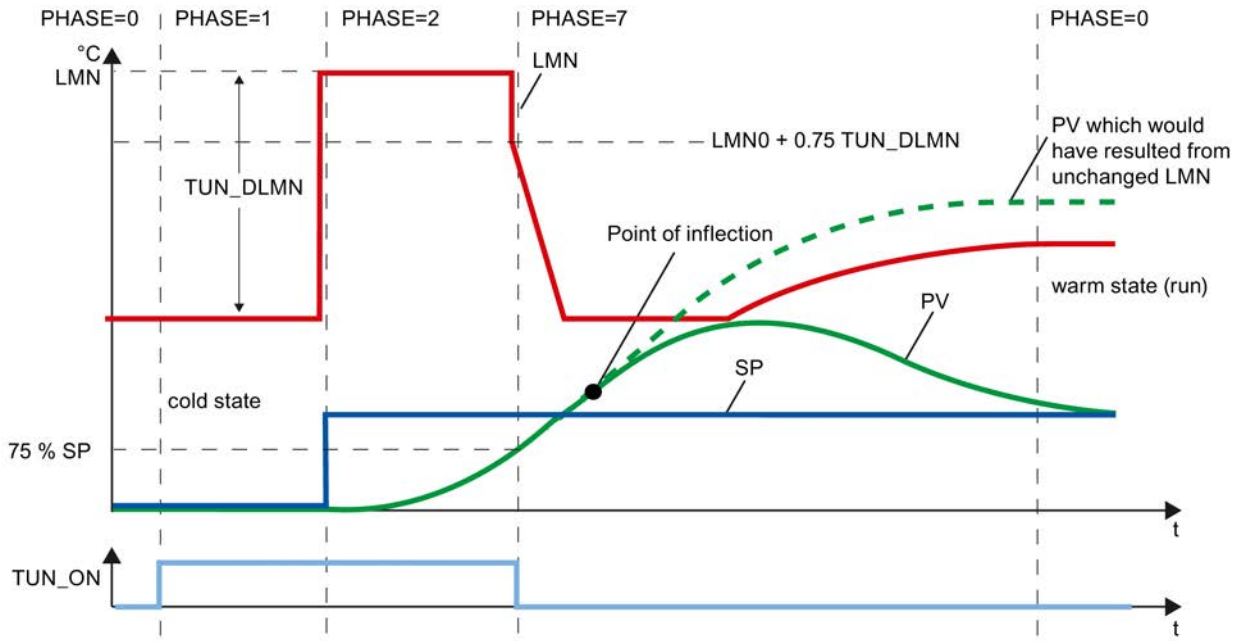
At the latest, tuning is ended when the process value has passed 75% of the setpoint step change (SP_INT-PV0). This is signaled by "inflection point not reached" in STATUS_H (2xx2x).

The currently valid setpoint always applies. By reducing the setpoint, it is possible to achieve an earlier end of the tuning function.

In typical temperature processes, cancelation of tuning at 75% of the setpoint step change is normally adequate to prevent overshoot. However, **caution** is advised, particularly in processes with a greater delay (TU/TA > 0.1, process type III). If manipulated variable excitation is too strong compared to the setpoint step change, the process value can overshoot heavily (up to a factor of 3).

In higher-order processes, if the point of inflection is still a long way off after reaching 75% of the setpoint step change, there will be significant overshoot. In addition, the controller parameters are too stringent. In this case, you should reduce the controller parameters or repeat the attempt.

The following schematic illustrates the overshoot of the process variable when the excitation is too strong (process type III):



In typical temperature processes, cancelation shortly before reaching the point of inflection is not critical in terms of the controller parameters.

If you repeat the attempt, reduce TUN_DLMN or increase the setpoint step change.

Principle: The value of the manipulated variable used for tuning must be suitable for the setpoint step change.

Error estimating the delay time or order

The delay time (STATUS_H = 2x1xx or 2x3xx) or order (STATUS_H = 21xxx or 22xxx) were not acquired correctly. Operation continues with an estimate that can lead to non-optimum controller parameters.

Repeat the tuning procedure and ensure that disturbances do not occur at the process value.

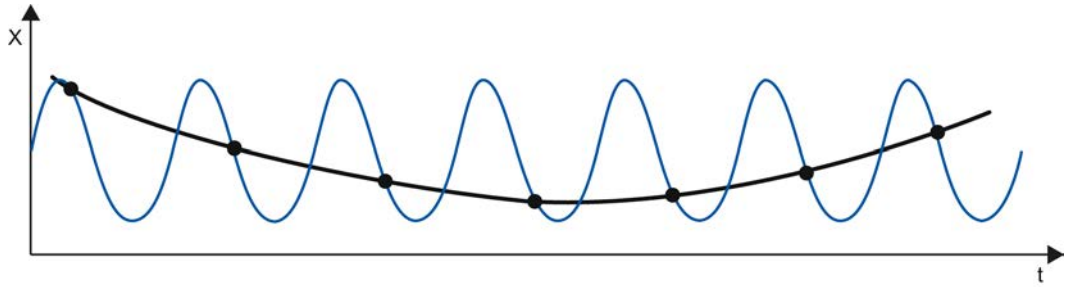
Note

The special case of a PT1-only process is also indicated by STATUS_H = 2x1xx (TU <= 3*CYCLE). In this case, it is not necessary to repeat the attempt. Reduce the controller parameters if the control oscillates.

Quality of measuring signals (measurement noise, low-frequency interference)

The results of tuning can be distorted by measurement noise or by low-frequency interference. Note the following:

- If you encounter measurement noise, set the sampling frequency higher rather than lower. During one noise period, the process value should be sampled at least twice. In pulse mode, integrated mean value filtering can be helpful. This assumes, however, that the process variable PV is transferred to the instruction in the fast pulse cycle. The degree of noise should not exceed 5% of the useful signal change.
- High-frequency interference cannot be filtered out by TCONT_CP. This should be filtered earlier in the measuring sensor to prevent the aliasing effect.
  The following schematic illustrates the aliasing effect when the sampling time is too long:



- With low-frequency interference, it is relatively easy to ensure an adequately high sampling rate. However, the TCONT_CP must then generate a uniform measuring signal by having a large interval in the mean value filtering. Mean value filtering must extend over at least two noise periods. Internally in the block, this soon results in higher sampling times such that the accuracy of the tuning is adversely affected. Adequate accuracy is guaranteed with at least 40 noise periods up to the point of inflection.
  Possible remedy when repeating the attempt:
  Increase TUN_DLMN.

Overshoot

Overshoot can occur in the following situations:

Situation	Cause	Remedy
End of tuning	• Excitation by a too high manipulated value change compared with the setpoint step change (see above). • PI controller activated by PID_ON = FALSE.	• Increase the setpoint step change or reduce the manipulated value step change. • If the process permits a PID controller, start tuning with PID_ON = TRUE.
Tuning in phase 7	Initially, less aggressive controller parameters were determined (process type III); these can lead to an overshoot in phase 7.	-
Control mode	PI controller with PFAC_SP = 1.0 for process type I.	If the process permits a PID controller, start tuning with PID_ON = TRUE.

Performing pretuning

Requirements

- The instruction and the technology object are loaded on the CPU.

Procedure

To manually determine the optimum PID parameters for initial commissioning, follow these steps:

1. Click the "Start" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value and output value are recorded.
2. Select "Pretuning" from the "Mode" drop-down list.
   TCONT_CP is ready for tuning.
3. In the "Output value jump" field, specify how much the output value should be increased.
4. Enter a setpoint in the "Setpoint" field. The output value jump only takes effect when another setpoint is entered.
5. Click the ▶ "Start tuning" icon.
   The pretuning starts. The status of the tuning is displayed.

Performing fine tuning

Requirements

- The instruction and the technology object are loaded on the CPU.

Procedure

To determine the optimal PID parameters at the operating point, follow these steps:

1. Click the "Start" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value and output value are recorded.
2. Select "Fine tuning" from the "Mode" drop-down list.
   TCONT_CP is ready for tuning.
3. In the "Output value jump" field, specify how much the output value should be increased.
4. Click the ▶ "Start tuning" icon.
   Fine tuning starts. The status of the tuning is displayed.

Cancelling pretuning or fine tuning

To cancel pretuning or fine tuning, click on the ■ icon, "Stop tuning".

If the PID parameters have not yet been calculated and stored, TCONT_CP starts in automatic mode LMN = LMN0 + TUN_DLMN. If the controller was in manual mode before the tuning, the old manual manipulated variable is output.

If the calculated PID parameters have already been saved, TCONT_CP starts in automatic mode and works with the previously determined PID parameters.

Manual fine-tuning in control mode

The following measures can be employed to achieve an overshoot-free setpoint response:

Adapting the control zone

During tuning, "TCONT_CP" determines a control zone CON_ZONE and activated if the process type is suitable (process type I and II) and a PID controller is used (CONZ_ON = TRUE). In control mode, you can modify the control zone or switch it off completely (with CONZ_ON = FALSE).

Note

Activating the control zone with higher-order processes (process type III) does not normally provide any benefit since the control zone is then larger than the control range that can be achieved with a 100% manipulated variable. There is also no advantage in activating the control zone for PI controllers.

Before you switch on the control zone manually, make sure that the control zone is not too narrow. If the control zone is set too narrow, oscillations occur in the manipulated variable and the process value.

Continuous attenuation of the control response with PFAC_SP

The control response can be attenuated with the PFAC_SP parameter. This parameter specifies the percentage of proportional component that is effective for setpoint step changes.

Regardless of the process type, PFAC_SP is set to a default value of 0.8 by the tuning function; you can later modify this value if required. To limit overshoot during setpoint step changes (with otherwise correct controller parameters) to approximately 2%, the following values are adequate for PFAC_SP:

	Process type I	Process type II	Process type III
	Typical temperature process	Intermediate range	Higher-order temperature process
PI	0.8	0.82	0.8
PID	0.6	0.75	0.96

Adjust the default factor (0.8) in the following situations, in particular:

- Process type I with PID (0.8 →0.6): Setpoint step changes within the control zone still lead to approximately 18% overshoot with PFAC_SP = 0.8.
- Process type III with PID (0.8 →0.96): Setpoint step changes with PFAC_SP = 0.8 are attenuated too strongly. This leads to a significantly slower response time.

Attenuation of control parameters

When a closed-loop control circuit oscillates or if overshoot occurs after setpoint step changes, you can reduce the controller GAIN (e.g., to 80% of the original value) and increase integral time TI (e.g., to 150% of the original value). If the analog output value of the continuous controller is converted to binary actuating signals by a pulse shaper, quantization noise may cause minor permanent oscillation. You can eliminate this by increasing the controller deadband DEADB_W.

Modifying control parameters

Proceed as follows to modify control parameters:

1. Save the current parameters with SAVE_PAR.
2. Modify the parameters.
3. Test the control response.

If the new parameter settings are worse than the old ones, retrieve the old parameters with UNDO_PAR.

Performing fine tuning manually

Requirements

- The instruction and the technology object have been loaded to the CPU.

Procedure

To manually determine the optimal PID parameters, proceed as follows:

1. Click the "Start" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value and output value are recorded.
2. Select "Manual" from the "Mode" drop-down list.
3. Enter the new PID parameters.
4. Click on the icon "Send parameter to CPU" in the "Tuning" group.
5. Select the "Change setpoint" check box in the "Current values" group.
6. Enter a new setpoint and click in the "Current Values" group on the icon .
7. Clear the "Manual mode" check box.
   The controller works with the new PID parameters and controls the new setpoint.

8. Check the quality of the PID parameter to check the curve points.
9. Repeat steps 3 to 8 until you are satisfied with the controller results.

TCONT_S

Technology object TCONT_S

The technology object TCONT_S provides a step controller for actuators with integrating behavior and is used to control technical temperature processes with binary output value output signals. The technology object corresponds to the instance data block of the TCONT_S instruction. The operating principle is based on the PI control algorithm of the sampling controller. The step controller operates without a position feedback signal. Both manual and automatic mode are possible.

S7-1500
All parameters and tags of the technology object are retentive and can only be changed during download to the device if you completely download TCONT_S.

See also

Overview of software controller (Page 9531)

Add technology objects (Page 9533)

Configure technology objects (Page 9534)

TCONT_S (Page 6833)

Downloading technology objects to device (Page 9536)

Configure controller difference TCONT_S

Use process value periphery

To use the input parameter PV_PER, proceed as follows:

1. Select the entry "Periphery" from the "Source" list.
2. Select the "sensor type".
   Depending on the sensor type, the process value is scaled according to different formulas.
   - Standard
     Thermoelements; PT100/NI100
     PV = 0.1 × PV_PER × PV_FAC + PV_OFFS
   - Cooling;
     PT100/NI100
     PV = 0.01 × PV_PER × PV_FAC + PV_OFFS
   - Current/voltage
     PV = 100/27648 × PV_PER × PV_FAC + PV_OFFS
3. Enter the factor and offset for the scaling of the process value periphery.

Use internal process values

To use the input parameter PV_IN, proceed as follows:

1. Select the entry "Internal" from the "Source" list.

Control deviation

Set a dead zone range under the following requirement:

- The process value signal is noisy.
- The controller gain is high.
- The derivative action is activated.

The noise component of the process value causes strong deviations of the output value in this case. The dead zone suppresses the noise component in the steady controller state. The dead zone range specifies the size of the dead zone. With a dead zone range of 0.0, the dead zone is turned off.

See also

Mode of operation TCONT_S (Page 6834)

Configure controller algorithm TCONT_S

General

1. Enter the "Sampling time PID algorithm".
   A controller sampling time should not exceed 10 % of the determined integral action time of the controller (TI).
2. If the controller structure contains a proportional action, enter the "proportional gain".
   A negative proportional gain inverts the rule meaning.

Proportional action

For changes of the setpoint, it may lead to overshooting of the proportional action. Through the weighting of the proportional action, you can select how strongly the proportional action

should react when setpoint changes are made. The weakening of the proportional action is reached through a compensation of the integral action.

1. To weaken the proportional action for setpoint changes, enter a "Proportional action weighting".
   - 1.0: Proportional action for setpoint change is fully effective
   - 0.0: Proportional action for setpoint change is not effective

Integral action

1. If the controller structure contains an integral action, enter the "integral action time". With an integral action time of 0.0, the integral action is switched off.

See also

Mode of operation TCONT_S (Page 6834)

Configure manipulated value TCONT_S

Pulse generator

1. Enter the minimum impulse duration and minimum pause duration.
   The values must be greater than or equal to the cycle time for the input parameter CYCLE. The frequency of operation is reduced through this.
2. Enter the motor setting time.
   The value must be greater than or equal to the cycle time of the input parameter CYCLE.

See also

Mode of operation TCONT_S (Page 6834)

Commissioning TCONT_S

Requirements

- The instruction and the technology object have been loaded to the CPU.

Procedure

To manually determine the optimal PID parameters, proceed as follows:

1. Click the "Start" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value and output value are recorded.
2. Enter new PID parameters in the "P", "I" and weighting proportional action fields.
3. Click on the icon "Send parameter to CPU" in the "Tuning" group.

4. Select the "Change setpoint" check box in the "Current values" group.
5. Enter a new setpoint and click in the "Current Values" group on the icon .
6. Clear the "Manual mode" check box.
   The controller works with the new parameters and controls the new setpoint.
7. Check the quality of the PID parameter to check the curve points.
8. Repeat steps 2 to 6 until you are satisfied with the controller results.

11.3.5 PID control (S7-300, S7-400)

11.3.5.1 Using standard PID control

Using PID_CP

Technology object PID_CP

The technology object PID_CP provides a continuous PID controller for automatic and manual mode. It corresponds to the instance data block of the instruction PID_CP. You can configure the following controllers:

- Fixed setpoint control with continuous P, PI, PD, PID controllers
- Fixed setpoint control with disturbance variable selection
- Cascade control
- Two-loop ratio control
- Blending control

You configure the following functions of the technology object PID_CP:

- Scaling the process value signal
- Suppression of high-frequency vibrations in the process value branch by smoothing (time delay) of the process value signal.
- Linearization of the quadratic functions of the process value (flow-rate control with differential pressure encoders)
- Calling of "own functions" in the process value branch
- Monitoring the process value for violation of alarm and warning limits
- Monitoring the rate of change of the process value
- Adjustment of the setpoint by ramp/soak
- Scaling an external setpoint signal with ratio controllers
- Calling of "own functions" in the setpoint branch
- Limiting the rate of change of the setpoint
- Limiting the absolute values of the setpoint

- Filtering the control deviation signal with a deadband
- Monitoring the control deviation for violation of alarm and warning limits
- The proportional, integral, and derivative actions can be activated separately.
- Option of proportional and derivative effect in the controller feedback path.
- Selecting a disturbance variable
- Control of the manipulated variable in manual mode from a PG or HMI)
- Calling "own functions" in the manipulated value branch
- Limiting the rate of change of the manipulated variable
- Limiting the absolute values of the manipulated variable
- Scaling the manipulated variable
- Pulse width modulation for pulse generator

Control deviation generation

Process value

Scaling (PV_NORM)

The "I/O process value ON" check box is used to specify which analog input variables are scaled to the physical unit of the process value.

- Activated: Input variable is the I/O process value (PV_PER)
- Disabled: Input variable is the internal process value (PV_IN)

In order to uniquely specify the scaling levels, you define the following four parameters:

- High input (NM_PIHR)
- Low input (NM_PILR)
- High output (NM_PVHR)
- Low output (NM_PVLR)

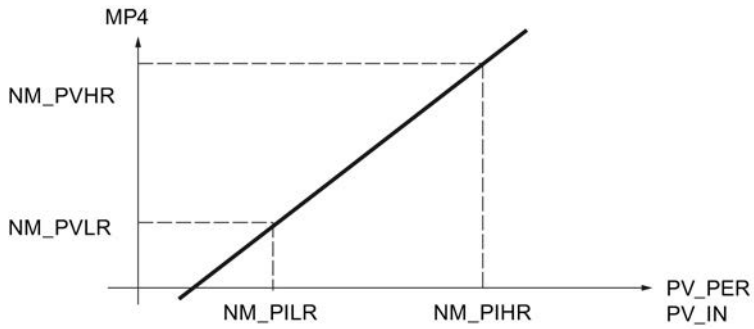
The parameters are preset depending on the sensor type.

The process value is calculated from the respective input value (PV_PER or PV_IN) according to the following formula:

MP4 = (PV_PER - NM_PILR) × (NM_PVHR - NM_PVLR) / (NM_PIHR - NM_PILR) + NM_PVLR

or

MP4 = (PV_IN - NM_PILR) × (NM_PVHR - NM_PVLR) / (NM_PIHR - NM_PILR) + NM_PVLR



The process value is output in the static variable MP4.

No values are limited and the parameters are not checked. If you enter the same value for input high and input low, a division by zero may occur based on the formulas above. The instruction will not detect this.

Filter functions

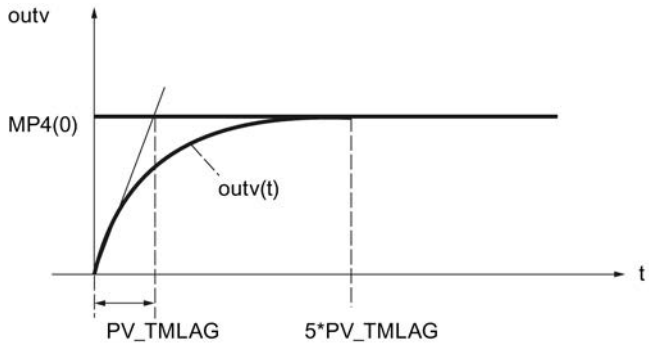
Smooth controlled variable (LAG1ST)

In order to smooth process signals with superimposed rapid variations (noise), a first-order delay element is applied in the process value branch. This function stabilizes the analog controlled variable to a greater or lesser degree, depending on the time constant (PV_TMLAG). In this way, disturbances are effectively suppressed. As a result of this, however, the time constant of the overall control loop also increases, i.e., the closed-loop control becomes more sluggish.

The stabilizing effect is achieved by a first-order delay algorithm.

The step response within the time range is:

$outv(t) = MP4(0)\ (1 - e^{-t/PV_TMLAG})$



MP4(0)	The height of the process value step at the input
outv(t)	The output variable
PV_TMLAG	The time lag constant
t	Time

Smoothing of the process value

If smoothing of the process value (LAG1STON = FALSE) is not activated, then the I/O input (PV_PER) or the internal input (PV_IN) is switched through to the process value branch without a delay.

Time lag

The PV_TMLAG input parameter specifies the time lag for the process value. Any value within the value range is permitted.

If PV_TMLAG ≤ 0.5 * CYCLE, there is no longer any lag in effect.

A sampling time (CYCLE) of less than one-fifth of the time lag is required in order to achieve a lag response approaching that of the analog response.

Square root (SQRT)

The "Square root" function (SQRT) is used to linearize the controlled variable. If the process value supplied by an encoder is a physical quantity and there is a quadratic relation between this quantity and the measured process variable, the characteristic of the controlled variable must be linearized prior to any further processing in the controller.

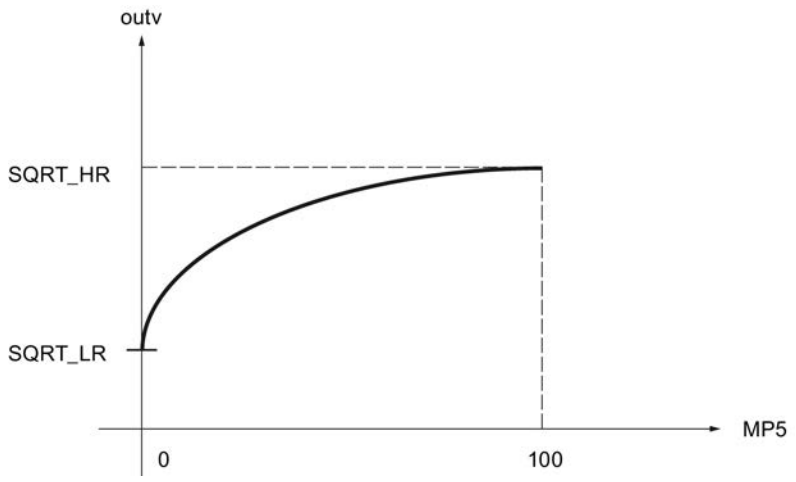
The measured signal must always be linearized by the square-root function if flow measurements are made using metering orifices, venturi tubes, or the like. The measured pressure differential is then proportional to the square of the flow.

The SQRT_ON = TRUE input signal is used to activate the square root function in the process value branch. The square root function algorithm has the following form:

outv = SQRT (MP5) × (SQRT_HR – SQRT_LR) / 10.0 + SQRT_LR

Here, outv is the scaled output value of the square root function (physical flow).

This formula stipulates that the input value of the square root is scaled to the number range 0 to 100. The parameters for the "Measuring range high" and "Measuring range low" must be set to 100.0 and 0.0 in the scaling of the process value. The square root of this value yields a measuring range from 0 to 10. This range of numbers will be scaled using the high limit (SQRT_HR) and the low limit of the physical measurement range (SQRT_LR).



Example of scaling

The input value PV_IN of the controller is the differential pressure in mbar:

Start of measuring range NM_PILR	End of measuring range NM_PIHR	Sample value for PV_IN
20.0 mbar	200.0 mbar	150.0 mbar

The scaled differential pressure is calculated with the help of the scaling function PV_NORM, whereby NM_PVHR = 100.0 and NM_PVLR = 0.0 are selected:

MP4 = (PV_IN - NM_PILR) * (NM_PVHR - NM_PVLR) /
(NM_PIHR - NM_PILR) + NM_PVLR
= (PV_IN - 20.0 mbar) * (100.0 - 0.0) /
(200.0 mbar - 20.0 mbar) + 0.0
= (PV_IN - 20.0 mbar) * 100 / 180.0 mbar

Initial value of MP4	End value of MP4	Sample value for MP4
0.0	100.0	72.222

Since no smoothing is used in this example, the following applies: MP5 = MP4.

For the square root from the scaled differential pressure MP5, the result is:

Initial value according to the square root	End value according to the square root	Sample value for SQRT(MP5)
0.0	10.0	8.498

For the scaled output value outv of the square root function (physical flow-rate) with SQRT_HR = 20000.0 m^3/h und SQRT_LR = 0.0 m3/h, there follows:

outv = SQRT(MP5) * (SQRT_HR - SQRT_LR) / 10.0 + SQRT_LR
= SQRT(MP5) * (20000.0 m^3/h - 0.0 m3/h) / 10.0 + 0.0 m3/h
= 2000.0 m^3/h * SQRT(MP5)

Start of measuring range outv	End of measuring range outv	Sample value for outv
0.0 m^3/h	20000.0 m^3/h	16996.732 m^3/h

FC call in the process value branch (PVFC)

Insertion of a user-specific function in the process value branch allows the controlled variable to undergo signal processing, e.g., signal delay or linearization, prior to any further processing in the controller.

Calling a function

If you want to insert a user-specific function into the process value branch, enable "Enable FC call" (PVFC_ON). The number of the utilized function is entered in the "FC number" input field (PVFC_NBR). The controller calls the function. In so doing, existing input/output parameters of the user function are not supplied. You have to program the data transfer in your function yourself.

You can monitor the output value of the function at the static variable MP6.

Note

No check is made to determine whether a function exists. If the function does not exist, the CPU switches to STOP with an internal system error.

Alarm

Monitor process value for limit values (PV_ALARM)

If the values of the process variables (e.g., speed, pressure, filling level, temperature, etc.) exceed or fall below certain values in closed-loop controls, an illegal process status or plant status can occur. In such cases, monitoring of the process value for limit values (PV_ALARM) is used in order to monitor the process value for violation of the permissible operating range. Limit violations are detected and signaled so that an appropriate reaction can be initiated.

Note

The monitoring of the process value for limit values cannot be deactivated. The violation of the configured limits is always signaled.

Monitoring of the process value for limit values (PV_ALARM) monitors the controlled variable PV(t) for four assignable limits in two tolerance bands. If the limits are reached or exceeded, the function signals a “warning” at the first limit and an “alarm” at the second limit.

Limit values

The numeric values of the limits are set in the input parameters for “warning” and “alarm”. If the process value (PV) exceeds or falls below these limits, the associated output message bits are set for the high alarm (QPVH_ALM), the high warning (QPVH_WRN), the low warning (QPVL_WRN), and the low alarm (QPVL_ALM).

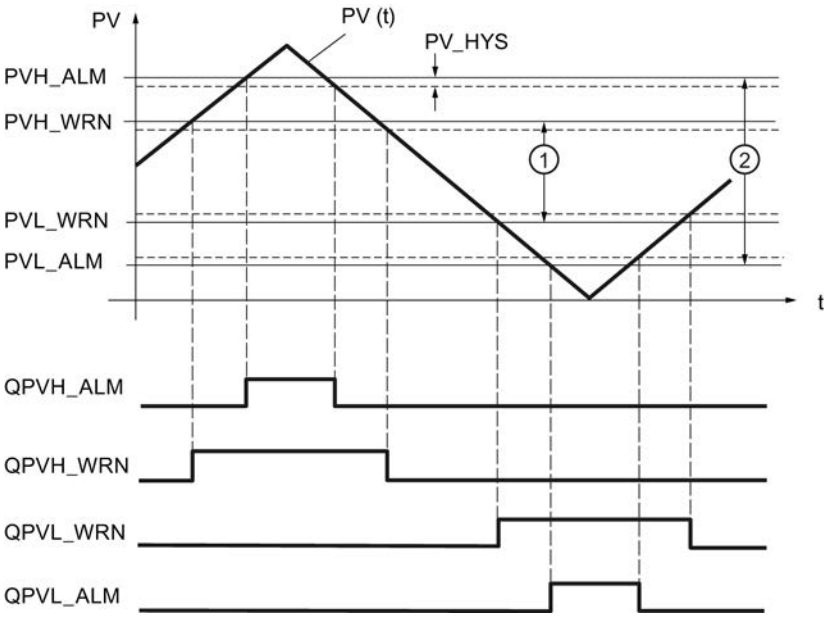
During a restart, the message bits are reset.

The message bits are set as follows:

QPVH_ALM	QPVH_WRN	QPVL_WRN	QPVL_ALM	If:	And:
TRUE	TRUE	FALSE	FALSE	PV ↗ PV ↘	PV ≥ PVH_ALM PV ≥ PVH_ALM - PV_HYS
FALSE	TRUE	FALSE	FALSE	PV ↗ PV ↘	PV ≥ PVH_WRN PV ≥ PVH_WRN - PV_HYS
FALSE	FALSE	TRUE	FALSE	PV ↘ PV ↗	PV ≤ PVL_WRN PV ≤ PVL_WRN + PV_HYS
FALSE	FALSE	TRUE	TRUE	PV ↘ PV ↗	PV ≤ PVL_ALM PV ≤ PVL_ALM + PV_HYS

The following has to apply so that the block operates properly:

High alarm < High warning < Low warning < Low alarm



①	1. Tolerance band
②	2. Tolerance band

Hysteresis

In order to prevent "flickering" of the message bits when the input variable undergoes slight changes or in the event of rounding errors, a hysteresis PV_HYS is used. The controlled variable must violate the hysteresis before the message bits are reset.

Monitor rate of change of the process value (ROCALARM)

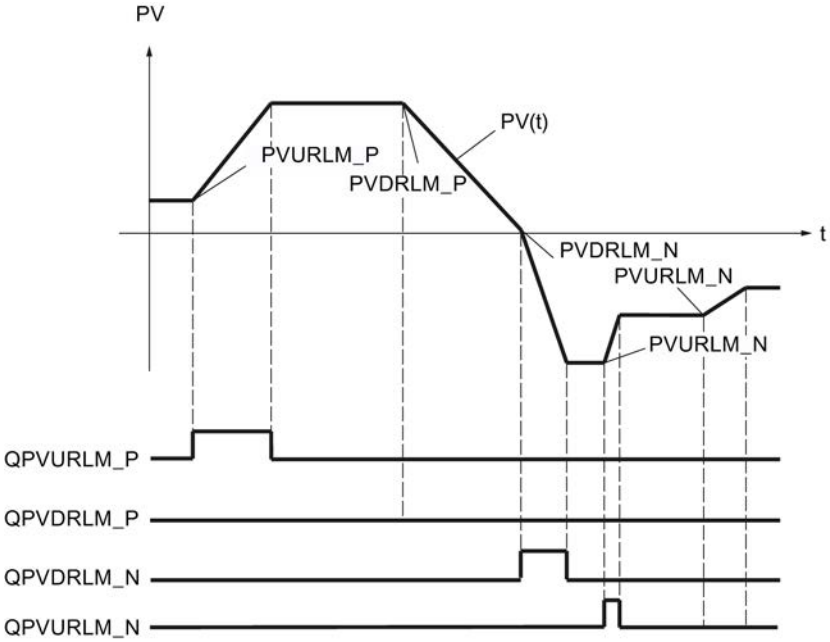
If the change in the process variable (e.g., speed, pressure, filling level, temperature, etc.) is too great in closed-loop controls, an illegal process status or plant status can occur. In such cases, monitoring the rate of change of the process value (ROCALARM) is used to monitor the process value for violations of a permissible rate of change. Limit violations are detected and signaled so that an appropriate reaction can be initiated.

Note

Monitoring of the rate of change of the process value (ROCALARM) cannot be deactivated. The violation of the configured limits is always signaled.

The ROCALARM function monitors the controlled variable PV(t) for permissible rate-of-change limits that are assignable according to sign.

The numeric values of the limit slopes are set at the input parameters for "Maximum increase" and "Maximum decrease" in the positive and negative range of the controlled variable. The slopes refer to a rising slope or falling slope in percent per second. If the rate of change of the controlled variable violates these limits, the associated output message bit QPVURLMP ... QPVDRLMN is set.



The ramp parameters are identified in accordance with the following scheme:

Parameter	PV change
Maximum increase in the positive range (PVURLM_P)	PV > 0 and |PV| rising
Maximum decrease in the positive range (PVDRLM_P)	PV > 0 and |PV| falling
Maximum increase in the negative range (PVURLM_N)	PV > 0 and |PV| rising
Maximum decrease in the negative range (PVDRLM_N)	PV < 0 and |PV| falling

Setpoint

Specifying a setpoint

Select a source for the setpoint. The following options are available:

- Internal setpoint
  The input parameter SP_INT is used as setpoint
- Setpoint generator
  The input parameter SP_INT is used as setpoint and can be incremented or decremented using the static variables SPUP and SPDN.
- Ramp/soak
  The input parameter SP_INT is used as setpoint and can be changed according to a specified ramp/soak.
- External setpoint
  The input parameter SP_EXT is used as setpoint for blending, ratio and cascade controllers. The external setpoint can be scaled.

The effective setpoint is shown at the output parameter SP.

Setpoint generator (SP_GEN)

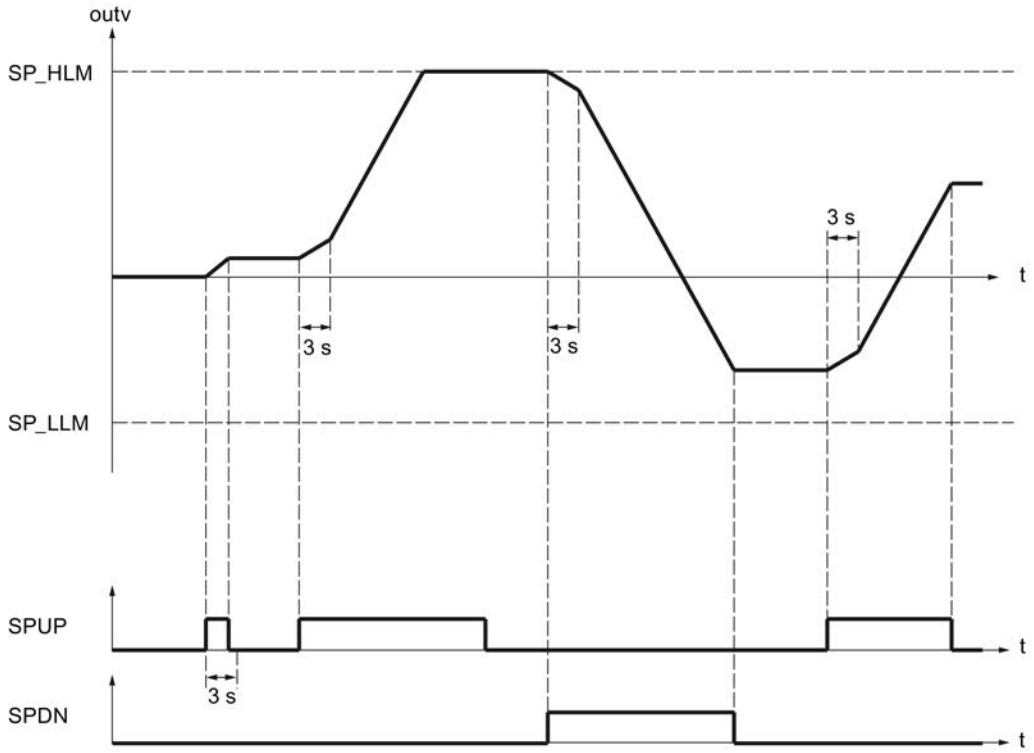
The internal setpoint SP_INT can be incremented or decremented in steps using the SPUP and SPDN switches. The modified value can be monitored at MP1. The configured setpoint limits are taken into account.

Effect of the switches SPUP and SPDN

For changes involving small steps, the controller should have a maximum sampling time of 100 ms.

The rate of change of the output variable depends on how long the SPUP or SPDN switch is actuated and on the selected limits. The following relationship applies:

- During the first 3 s after setting SPUP or SPDN:
  doutv/dt = (SP_HLM - SP_LLM) / 100 s
- Afterwards:
  doutv/dt = (SP_HLM - SP_LLM) / 10 s



For a sampling time of 100 ms and a setpoint range of -100.0 to 100.0, the setpoint, for example, changes by 0.2 per cycle during the first 3 seconds. If SPUP is TRUE for more than 3 seconds, the rate of changes increases to ten times the value, in this case therefore to 2 per cycle.

Startup and operating principles of the setpoint generator

- During a complete restart, output outv is reset to 0.0.
- If you activate the setpoint generator (SPGEN_ON = TRUE), the signal value SPFC_IN is output initially at output outv. Therefore, the switchover to the setpoint generator from another operating mode is always smooth. Provided the SPUP and SPDN (up/down keys) are not activated, the signal value SPFC_IN is retained at the output.

Parameters of the SP_GEN function

Output parameter outv is an implicit parameter. It can be monitored with the configuration tool via measuring point MP1.

Parameter	Meaning	Permitted value range
SPFC_IN	FC input setpoint	Engineering value range
SP_INT	Internal setpoint	Engineering value range

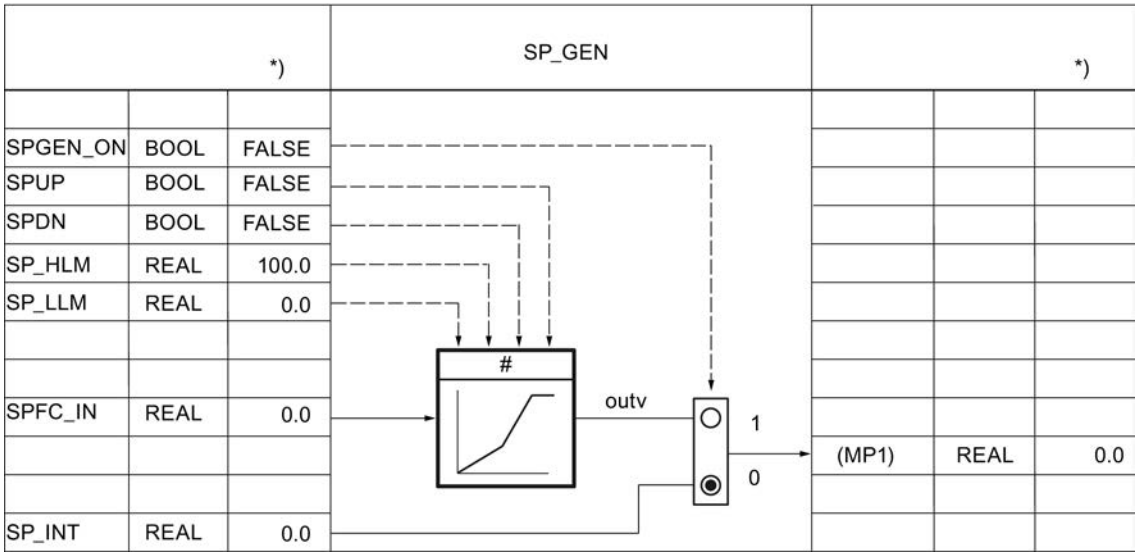


Figure 11-2 Function scheme and parameters of the setpoint generator

Ramp/soak (RMP_SOAK)

If you want the setpoint SP_INT to change autonomously depending on the time, for example, during the control of processes after a time-controlled temperature program, this is possible by configuring a ramp/soak and activating the ramp/soak function. The setpoint change is formed from the level sections with a maximum of 256 time slices.

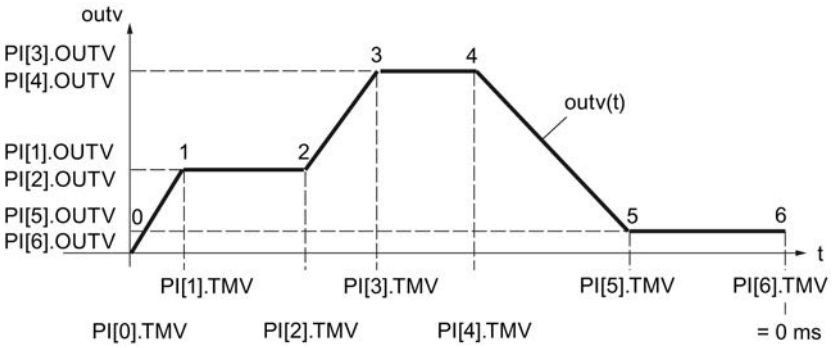
Note

The block function does not check whether a global DB with the number 'DB_NBR' exists or whether the parameter 'Number of time slices' NBR_PTS corresponds to the data block length. If the parameter assignment is incorrect, the CPU goes to **STOP** and issues an internal system error.

How the ramp/soak works

The ramp/soak supplies the output variable outv according to a function that runs to a defined ramp/soak. The ramp/soak process is defined by specifying a series of time slices in a global

data block RMP_SOAK with time values PI[i].TMV and the associated output values PI[i].OUTV. The following figure shows a ramp/soak with 6 time slices.

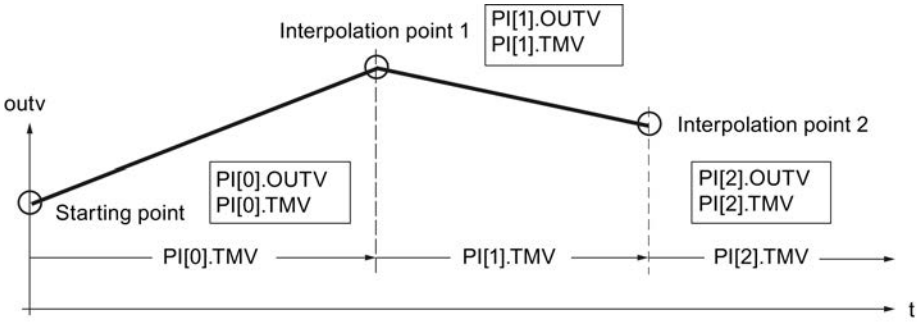


The time value for the last time slice n is PI[n].TMV = 0 ms. The processing time of a ramp/soak is counted down from the initial value to zero.

PI[i].TMV specifies the intervals between time slices. Between the time slices, interpolation is according to the following function:

$$outv(t) = PI[n + 1].\ OUTV - \frac{RS_TM}{PI[n].TMV}\ (PI[n + 1].OUTV - PI[n].OUTV)$$

$0 \leq n \leq (NBR_PTS - 1)$



Note

Note

During the interpolation of the ramp/soak between the time slices, pauses can sometimes occur at the initial value if the sampling time CYCLE is very small compared to the time between the time slices PI[n].TMV. Due to the computational accuracy of the CPU, the ramp/soak function cannot linearize all flat ramp/soak profiles. If the ramp/soak profile is too flat, the initial value pauses for a time at the respective time slice and continues, after a certain time, to integrate with the minimum rate of rise to the next time slice.

To correct or avoid error: Shorten the time between the time slices by inserting additional time slices. The output ramp/soak profile thus approximates the desired flat ramp/soak profile.

Ramp/soak function ON

The ramp/soak function is activated by changing RMPSK_ON from FALSE to TRUE. The ramp/soak is terminated after the last time slice has been reached. When the function is started again by the operator, RMPSK_ON first has to be set to 'FALSE' and then back to 'TRUE' .

During a **restart**, the output outv is reset to 0.0 and the total time or total remaining time is determined. During the transition to normal operation, the ramp/soak profile is processed immediately from the starting point according to the set operating mode. If this is not desired, the parameter RMPSK_ON must be set to FALSE in the restart OB.

Ramp/soak functions

The instruction has the following functions:

- Activate ramp/soak for single execution
- Pre-assign fixed value to the output
- Cyclic repetition on
- Hold ramp/soak
- Continue ramp/soak
- Update the total processing time and remaining total time

The following table applies for the significance of the control inputs when setting a desired operating mode:

Operating mode	RMPSK_ ON	DFOUT_ ON	RMP_ HOLD	CONT_ ON	CYC_ ON	TUPDT_ ON	Output signal OUTV
Ramp/soak on	TRUE	FALSE	FALSE	**)	FALSE	**)	OUTV(t) End value is held after the end of processing.
Set output	TRUE	TRUE	**)	**)	**)	**)	DF_OUTV
Cyclic repetition on	TRUE	FALSE	FALSE	**)	TRUE	**)	OUTV(t) After end: Automatic start
Hold ramp/soak	TRUE	FALSE	TRUE	FALSE	**)	**)	Current value of OUTV(t) is held *)
Continue ramp/soak	TRUE	FALSE	TRUE	TRUE	**)	**)	OUTV (old) *)
			FALSE				Operation is continued with newly defined values.
Updating the total time and remaining total time	**)	**)	**)	**)	**)	FALSE	Does not influence OUTV
						TRUE	Does not influence OUTV

*) The ramp/soak profile does not have the form configured by the user until the next time slice.

**) The respective operating mode set is executed irrespective of the significance of the control signals in the fields marked with **)

Set output of ramp/soak

If you enable this function (DFRMP_ON = TRUE), the output value of the ramp/soak is set to signal value SP_INT or output value SP_GEN.

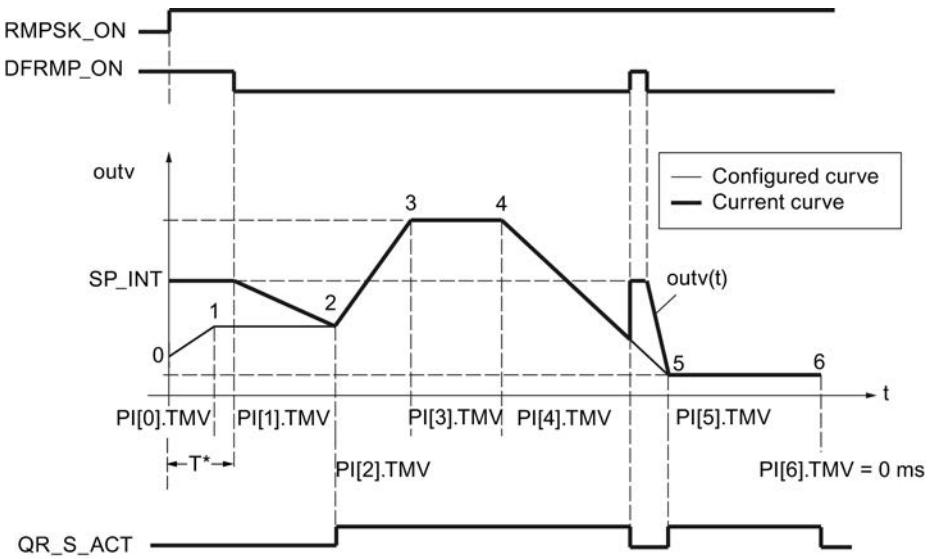
Note

Preassignment with the internal setpoint has an effect only if the ramp/soak function is enabled (RMPSK_ON = TRUE).

After the switchover to DFRMP_ON = FALSE, outv moves linearly from the specified setpoint (e.g., SP_INT) to the output value of the current time slice number PI[NBR_ATMS].OUTV.

The internal time processing continues running even when a fixed setpoint (RMPSK_ON = TRUE and DFRMP_ON = TRUE) is switched through.

On the start of the ramp/soak profile (RMPSK_ON = TRUE), the fixed setpoint SP_INT continues to be output until DFRMP_ON changes from TRUE to FALSE after the duration of T*.



After this moment, the time PI[0].TMV and a part of the time PI[1].TMV has expired. OUTV moves from SP_INT to PI[2].OUTV, i.e., from the setpoint to time slice 2.

The configured ramp/soak profile is output only starting from time slice 2. The conformity of the configured ramp/soak profile to the current ramp/soak profile is displayed by setting QR_S_ACT = TRUE. On a rising edge at DFRMP_ON during processing of the ramp/soak profile, the output value OUTV jumps to SP_INT without a delay.

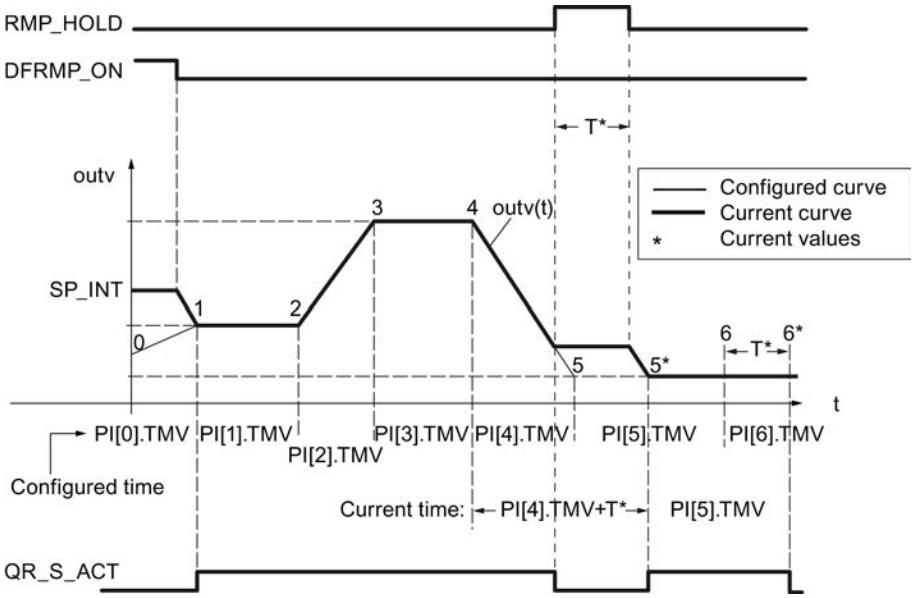
Cyclic repetition on

If "Cyclic repetition" operating mode (CYC_ON = TRUE) is activated, the ramp/soak function automatically returns to the starting point after the last time slice value has been output and begins a new run.

There is no interpolation between the last time slice and the starting point. For a smooth transition the following must be true: PI[NBR_PTS].OUTV = PI[0].OUTV.

Hold ramp/soak

When RMP_HOLD = TRUE, the value of the output variable (including the time processing) is frozen. When RMP_HOLD = FALSE is reset, the function resumes at the interruption point PI[x].TMV.



The processing time of the ramp/soak profile is extended by the amount of the time T*. The ramp/soak profile has the configured course from the starting point until the rising edge at RMP_HOLD and from time slice 5* to time slice 6*, i.e., output signal QR_S_ACT has the value TRUE.

If switch CONT_ON is set, the held ramp/soak function continues processing the ramp/soak profile at preassigned point TM_CONT.

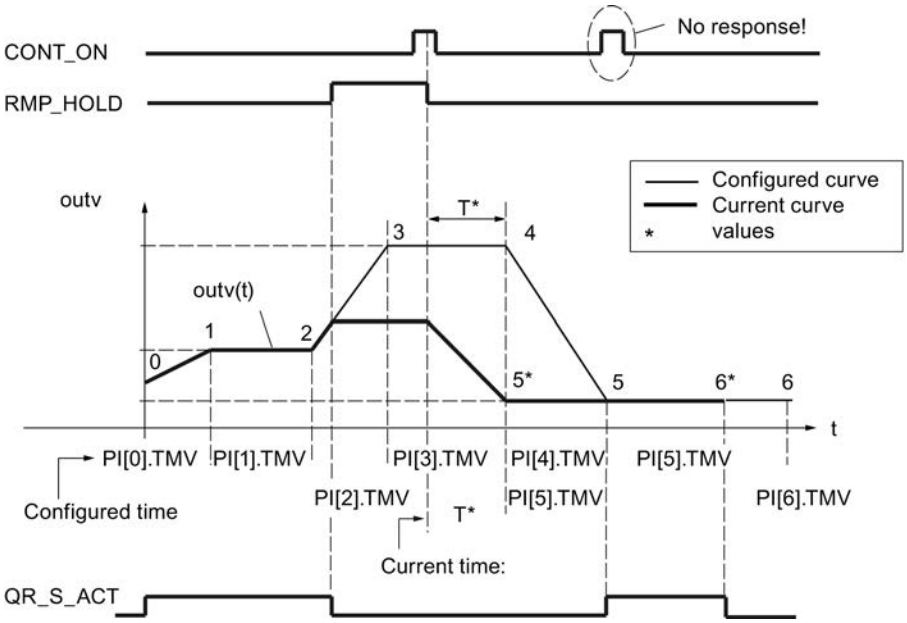
Continue ramp/soak

With CONT_ON, a held ramp/soak profile is restarted at a defined point. If the ramp/soak function is not held, CONT_ON has no effect.

If the control input for continuing (CONT_ON = TRUE) is set, the held ramp/soak function is continued at time slice TM_SNBR. TM_CONT determines the remaining time that the ramp/soak function requires to reach time slice TM_SNBR.

The following figure shows the effect of RMP_HOLD and CONT_ON when the configuration is as follows:

Continue at time slice 5 (TM_SNBR = 5) and continue in T* s: TM_CONT = T*



Time slices 3 and 4 are omitted in this processing cycle of the ramp/soak function. On a signal change of RMP_HOLD from TRUE to FALSE, the configured curve progression is only reached again starting from time slice 5.

Output QR_S_ACT is only set when the ramp/soak function executes the ramp/soak profile assigned by the user.

Updating the total time and remaining total time

The current time slice number NBR_ATMS, the current remaining time until the time slice is reached (RS_TM), the total time T_TM, and the total remaining time until the end of the ramp/soak profile (RT_TM) are updated in every cycle.

In the case of online changes of PI[n].TMV, the total time and the total remaining time of the ramp/soak profile change. Since the calculation of T_TM and RT_TM with many time slices significantly increases the processing time of the function block, the calculation is only carried out after a restart or at TUPDT_ON = TRUE. The time intervals PI[0 ... NBR_PTS].TMV between the individual time slices are added up and indicated at the total time T_TM and total remaining time RT_TM outputs.

Please note that relatively high CPU runtimes are required to determine the total times.

Procedure

To configure the ramp/soak function, proceed as follows:

1. Select "Ramp/soak" from the "Setpoint source" drop-down list.
2. Create a global data block RMP_SOAK.
3. Define the required parameters for the ramp/soak.
4. Enter the number of the data block RMP_SOAK in the "DB number" input field.

5. Enable or disable the additional functions of the ramp/soak.
6. If you have enabled the Continue ramp/soak function, enter at which time slice and when the ramp/sock is to be continued.

See also

Data block RMP_SOAK (Page 6908)

Scaling

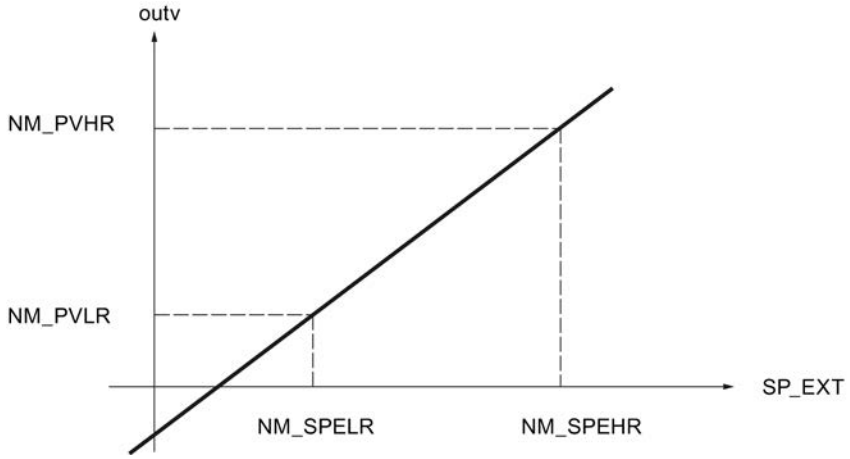
Scaling of an external setpoint

If the external setpoint is not expressed in the physical unit of the process value (e.g., expressed as a percent in the case of cascade control), then this value and its setting range must be scaled to the physical unit of the process value. The "Scaling in setpoint branch" function is used for this.

The SP_NORM function scales an analog input variable. The analog external setpoint is transformed to output variable outv using the scaling level. Output value OUTV is accessible in the configuration tool via measuring point MP2.

The output value of the function has an effect when control input SPEXT_ON = TRUE. In order to uniquely specify the scaling levels, you define the following four parameters:

- High input: SP_EXT: NM_SPEHR
- Low input: SP_EXT: NM_SPELR
- High output: NM_PVHR (You specify this value in the scaling function of the process value.)
- Low output: NM_PVLR (You specify this value in the scaling function of the process value.)



Output value outv is calculated from the respective input value SP_EXT according to the following formula:

outv = (SP_EXT – NM_SPELR) x (NM_PVHR – NM_PVLR) / (NM_SPEHR – NM_SPELR) + NM_PVLR

For the special case in which a square root function is activated in the process value branch, the scaling values of the square root function (SQRT_HR and SQRT_LR) apply as the high and low limits of output value outv.

In this case, output value outv of the scaling function is calculated from the respective input value SP_EXT according to the following formula:

outv = (SP_EXT – NM_SPELR) x (SQRT_HR – SQRT_LR) / (NM_SPEHR – NM_SPELR) + SQRT_LR

There is no internal value limiting in the function. The parameters are not checked. If you enter the same value for NM_SPEHR and NM_SPELR, a division by zero is possible based on the formulas above. The instruction will not detect this.

Combination/ratio factor

With ratio control, the setpoints of the control loop are coordinated with one another in such a way that the set ratio matches the actual ratio of the process values.

With blending control, the desired quantities of the individual components are set. The total of the blending factors FAC must be 1.

Procedure

To configure the scaling, proceed as follows:

1. Enter the values for the scaling in the input fields "High input" and "Low input".
2. Enter a blending/ratio factor (FAC).

Limiting functions

FC call in the setpoint branch

Insertion of a user-specific function in the setpoint branch allows an externally specified reference variable to undergo signal processing (e.g., signal delay or linearization) prior to application in the controller.

If you activate the FC call in the setpoint branch, a user-specific function is called. The controller calls the FC. In so doing, existing input/output parameters of the user FC are not initialized. Therefore, you must use S7-STL to program the data transfer in the user FC. You can monitor the prepared setpoint at MP3.

The value of SPFC_ON then determines whether a user-programmed function in the form of a standard FC (e.g., a characteristic curve) will be inserted at this point in the setpoint channel or whether the setpoint will continue to be processed without any influence of this type.

Note

The statement does not check to determine if a function exists. If the function does not exist, the CPU switches to STOP with an internal system error.

Procedure

To call a function in the setpoint branch, proceed as follows:

1. Select the "Enable FC call" check box.
2. In the "FC" input field, enter the number of the function to be called.

Limiting of the rate of change of the setpoint

Ramp functions are used in the setpoint branch in situations where the process does not tolerate abrupt changes in the actuating signal, since an abrupt setpoint change generally also causes an abrupt change in the manipulated variable of the controller. These abrupt manipulated variable changes must be avoided, for example, when a gearbox is connected in between an adjustable-speed motor and the load to be driven and an excessively rapid increase of the motor speed would result in overloading of the gearbox.

The SP_ROC function limits the rate of change of the setpoints processed in the controller. This is carried out separately for the rising slope and falling slope and separately for the positive and negative range of the reference variable.

The limiting for each slope of the ramp function in the positive range and in the negative range of the reference variable is entered in the four inputs SPURLM_P, SPDRLM_P, SPURLM_N, and SPDRLM_N. The slopes refer to a rising or falling slope per second. Faster setpoint changes are ramped down to these maximum rates.

For example, if SPURLM_P is assigned the value 10.0 [engineering value range/s], the following values are added to the "old value" of outv in each sampling cycle for as long as inv > outv:

Sampling time	1 s	$outv_{old}$ + 10
	100 ms	$outv_{old}$ + 1
	10 ms	$outv_{old}$ + 0.1

The following figure shows an example of how the signal processing functions. The step functions at input inv(t) become ramp functions at output outv(t).

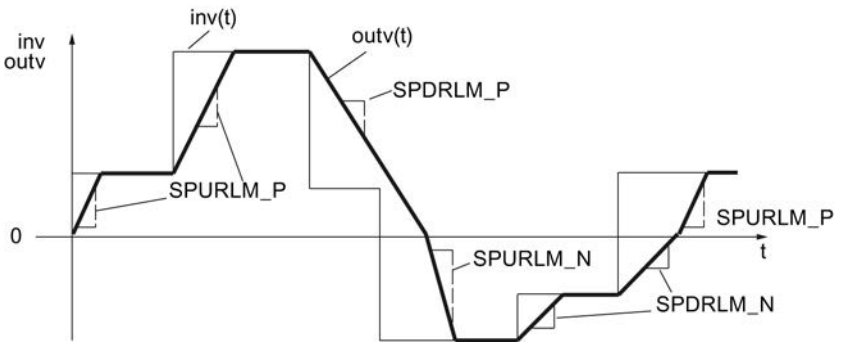


Figure 11-3 Limiting of the rate of change of the reference variable SP(t)

When slope limits are reached, this is not signaled.

Procedure

To limit the rate of change, proceed as follows:

1. Activate the "Limit rate of change" check box.
2. Specify the following values for the positive range as well as for the negative range:
   - Rising slope for setpoint value
   - Falling slope for setpoint value

Limiting of the absolute value of the reference variable

The setting range of the reference variable determines the range in which the controlled variable can fluctuate, i.e., the range in which the process varies within the scope of permissible status values.

To avoid a critical or impermissible process status, the setting range of the reference variable is subject to high and low limits (SP_LIMIT) in the setpoint branch of standard PID control.

The SP_LIMIT function limits the setpoint SP at the assignable low and high values (SP_LLM and SP_HLM) as long as input variable INV is outside these limits. Because the function cannot be deactivated, the specification of a low limit and high limit must be taken into consideration in the configuration.

The numeric values of the limits are set in the input parameters for the low and high limits. In the event of violations by input variable inv(t), the associated displays are output via the message outputs.

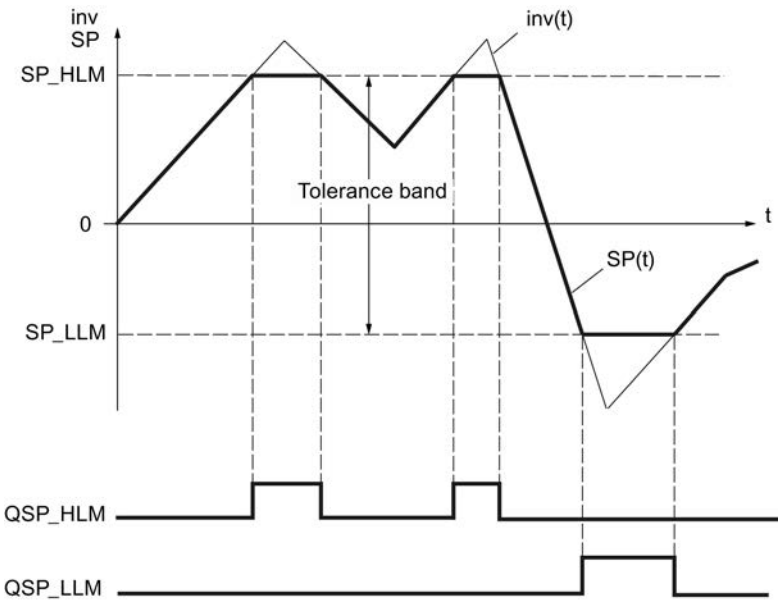


Figure 11-4 Absolute value limiting of reference variable SP(t)

During a restart, the message bits are reset.

The message bits are set as follows:

SP =PVH_ALM	QSP_HLM	QSP_LLM	If:
SP_HLM	TRUE	FALSE	inv ≥ SP_HLM
SP_LLM	FALSE	TRUE	inv ≤ SP_LLM
inv	FALSE	FALSE	SP_HLM < inv < SP_LLM

Procedure

To limit the absolute value of the reference variable, proceed as follows:

1. Enter the high limit for the absolute setpoint (SP_HLM) in the "High limit" input field.
2. Enter the low limit for the absolute setpoint (SP_LLM) in the "Low limit" input field.

Control deviation

Deadband

In an optimally tuned controller, when a high-frequency interfering signal is superimposed on the control variable or reference variable, the noise portion will also have an effect on the controller output. This can result in strong fluctuations of the manipulated variable, for example, if the controller gain is high and derivative action is enabled. Due to the increased switching frequency (step controller), this causes increased wear of the actuator.

The function reduces the noise portion in the control deviation signal while the controller is in steady-state and thus reduces undesired oscillating of the controller output.

The deadband can be disabled.

Signal filtering through deadband function

The DEADBAND function suppresses small fluctuations of the input variable around a specified zero point within an adjustable range. Outside this fluctuation range, the control deviation ER rises or falls proportional to the input variable. The width of the deadband can be specified with the DEADB_W parameter. Only positive values are permitted for the deadband width.

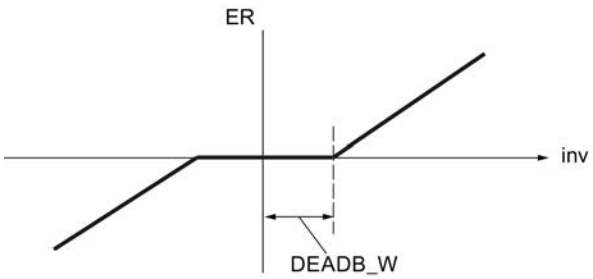
If the input variable is within the deadband, the value 0 (control deviation = 0) is output at the output. Only when the input variable exits the deadband will the output rise or fall by the same values as input variable inv. This results in corruption of the transmitted signal outside the deadband, too. However, this is accepted in order to avoid step changes at the deadband limits (see figure below). The corruption corresponds to the value DEADB_W and can therefore be controlled easily.

The DEADBAND function operates according to the following functions:

(ER) = inv + DEADB_W for inv < - DEADB_W

(ER) = 0 for -DEADB_W ≤ inv ≤ + DEADB_W

(ER) = inv - DEADB_W for inv > + DEADB_W



Effects of the signal filtering can be monitored at the output parameter ER.

Alarm

Excessive deviations of the controlled variable from the specified setpoint can cause an undesired process status. In such cases, the ER_ALARM function is used to monitor the control deviation for violation of the permissible operating range. ER_ALARM detects and signals any limit violations that occur so that an appropriate reaction can be initiated.

Note

The monitoring of the control deviation for limit values cannot be disabled. The violation of the configured limits is always signaled.

The ER_ALARM function monitors the size of control deviation ER(t) for four assignable limits in two tolerance bands. If the limits are reached or exceeded, the function signals a “warning” at the first limit and an “alarm” at the second limit.

Limit values

The numeric values of the limits are set in the input parameters for “warning” and “alarm”. If the control deviation (ER) violates these limits, the associated output message bits QERN_ALM ... QERP_ALM are set.

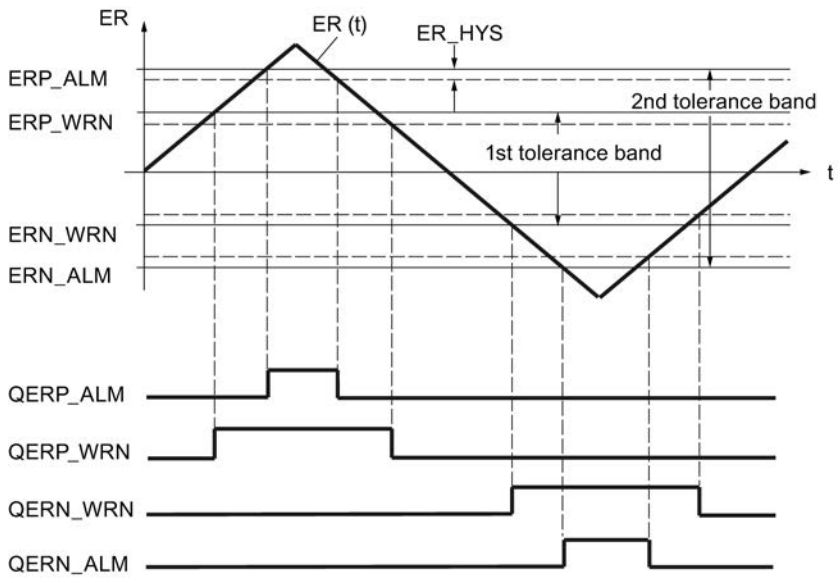
During a restart, the message bits are reset.

The message bits are set as follows:

QERP_ALM	QERP_WRN	QERN_WRN	QERN_ALM	If:	And:
TRUE	TRUE	FALSE	FALSE	ER ↗ ER ↘	ER ≥ ERP_ALM ER ≥ ERP_ALM - ER_HYS
FALSE	TRUE	FALSE	FALSE	ER ↗ ER ↘	ER ≥ ERP_WRN ER ≥ ERP_WRN - ER_HYS
FALSE	FALSE	TRUE	FALSE	ER ↘ ER ↗	ER ≤ ERP_WRN ER ≤ ERP_WRN + ER_HYS
FALSE	FALSE	TRUE	TRUE	ER ↘ ER ↗	ER ≤ ERP_ALM ER ≤ ERP_ALM + ER_HYS

The following has to apply so that the block operates properly:

High alarm < High warning < Low warning < Low alarm



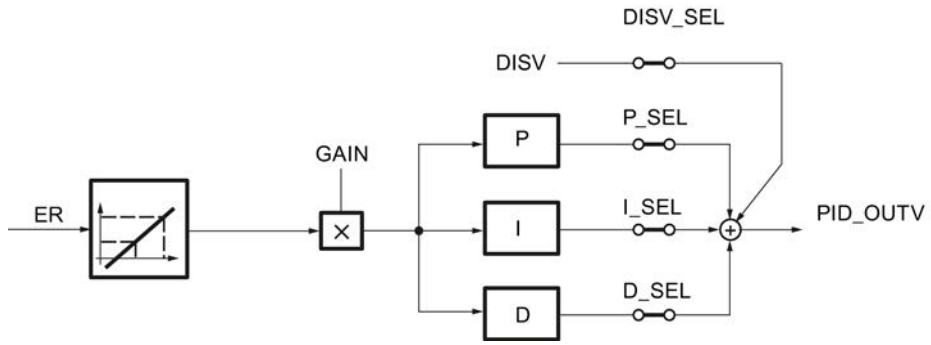
Hysteresis

In order to avoid "flickering" of the message bits when the input variable undergoes slight changes or in the event of rounding errors, a hysteresis ER_HYS is used. The control deviation must violate the hysteresis before the message bits are reset.

Controller

Control algorithm

The manipulated variable of the continuous controller is calculated within the configured sampling time cycle on the basis of the control deviation in the PID position algorithm. The controller is implemented in a purely parallel structure. The proportional, integral, and derivative actions can each be disabled individually.

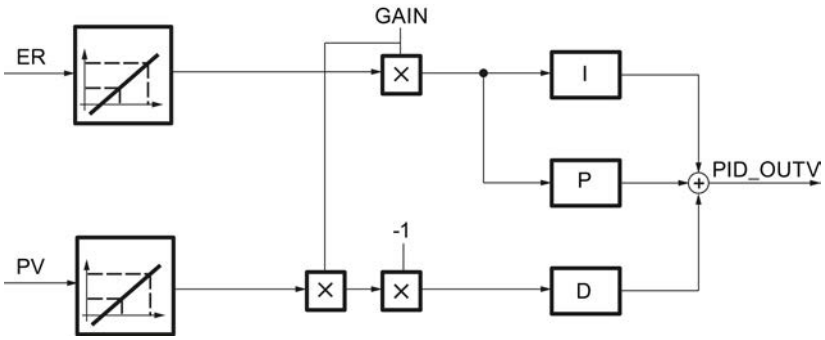


Feedforward control

A disturbance variable DISV can be added in the output signal of the controller (PID_OUTV). The variable is switched-in or switched-off in the PID window of the configuration tool using the DISV_SEL structure selector or through "Disturbance variable on".

Proportional and derivative action in the feedback path

In the parallel structure, each action in the control algorithm receives the control deviation as input signal. Via the proportional action and derivative action, the manipulated variable is influenced directly by setpoint steps. However, a different controller structure in which the formation of the proportional and the derivative actions is moved to the feedback path ensures a smooth response of the manipulated variable to setpoint step changes. In this structure, the integral action processes the control deviation as the input signal. Only the negative controlled variable (factor = - -1) is applied to the proportional action and the derivative action.



Scaling the input variable ER and PV

The input variables ER and PV of the PID controller are scaled to the range 0 to 100 prior to the controller processing according to the following formulas:

- with square-root function disabled (SQRT_ON = FALSE):
  - ER_{scaled} = ER * 100.0 / (NM_PVHR – NM_PVLR)
  - PV_{scaled} = (PV - NM_PVLR) * 100.0 / (NM_PVHR – NM_PVLR)
- with square-root function enabled (SQRT_ON = TRUE):
  - ER_{scaled} = ER * 100.0 / (SQRT_HR – SQRT_LR)
  - PV_{scaled} = (PV - SQRT_LR) * 100.0 / (SQRT_HR – SQRT_LR)

This scaling is performed so that the proportional gain GAIN of the PID controller can be entered without dimensions. If the high limit and low limit of the physical measuring range change (for example, from bar to mbar), the gain factor does not then have to be changed.

The scaled input variables ER_{scaled} and PV_{scaled} cannot be monitored.

See also

Controller structures (Page 9677)

Proportional component (Page 9681)

Derivative component (Page 9684)

Integral action (Page 9681)

Controller structures

Use and parameter assignment of the PID controller

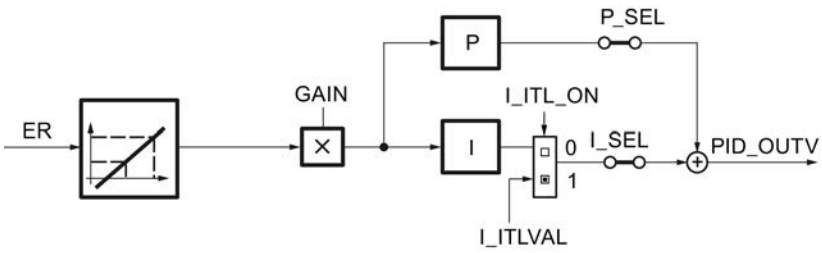
The majority of the controlled systems occurring in the process industry can be controlled with the PI controller function/PID controller function of standard PID control. Only in special cases are additional methods and measures needed for a particular closed-loop control.

However, a major problem in the field continues to be how to assign the PI controller/PID controller parameters, i.e., how to determine the "correct" settings for the controller parameters. The quality of this parameter assignment is critical in order for the PID control to meet the requirements of the task at hand. It requires either a lot of practical experience, special expertise, or a great amount of time.

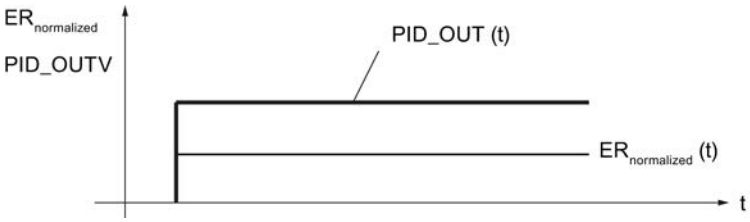
P control

The integral and derivative actions are disabled in a proportional-action controller (P-controller). (I_SEL and D_SEL = FALSE). This means that when control deviation ER = 0, output signal OUTV = 0. If an operating point ≠ 0 is intended, i.e., a numeric value will be set for the output signal when the control deviation is zero, the integral branch must be activated (see the following figure).

In the integral action, an operating point ≠ 0 can be specified for the P-controller via a corresponding setting of initialization value I_ITLVAL. Set switches 'I_ITL_ON' and 'I_SEL' = TRUE for this purpose.



The step response of the P-controller within the time range is:
PID_OUTV (t) = I_ITLVAL + GAIN * ER_{scaled} (t)



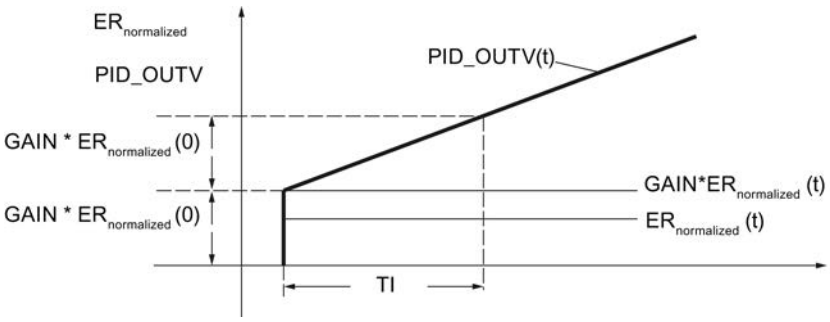
PID_OUTV(t)	Manipulated variable when controller is in automatic mode
I_ITLVAL	Operating point of the P-controller
GAIN	Controller gain
$ER_{scaled}(t)$	Scaled control deviation

PI control

The derivative action is disabled in the proportional-integral controller (PI controller). (D_SEL = FALSE). A PI controller adjusts output variable PID_OUTV using the integral action until control deviation ER = 0. However, this only applies if the output variable does not exceed the limits of the control range.

The step response within the time range is:

$$PID_OUTV(t) = GAIN * ER_{normalized}(0)\left(1 + \frac{1}{TI} * t\right)$$



PID_OUTV(t)	Manipulated variable when controller is in automatic mode
GAIN	Controller gain
$ER_{scaled}(0)$	Step height of the scaled control deviation
TI	Integral-action time constant

PD control

The integral action is disabled in the proportional-derivative controller (PD-controller) (I–SEL = FALSE). This means that when control deviation ER = 0, output signal OUTV = 0. If an operating point ≠ 0 is intended, i.e., a numeric value will be set for the output signal when the control deviation is zero, the integral action branch must be activated (see also the block diagram of the P control).

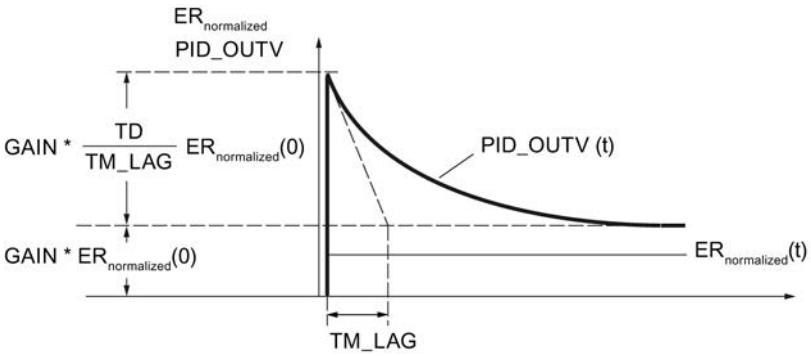
In the integral action, an operating point ≠ 0 can be specified for the PD-controller via a corresponding setting of initialization value I_ITLVAL. The switches 'I_ITL_ON' and 'I_SEL' must be set to TRUE for this purpose.

The PD-controller maps input variable ER(t) proportionally to the output signal and adds to this the derivative action generated by differentiation of ER(t); the derivative action is calculated with double precision according to the trapezoid rule (Padé approximation). The time response is determined by the derivative time constant TD.

For signal smoothing and suppression of interfering signals, a first-order delay (adjustable time: TM_LAG) is integrated into the algorithm for generating the derivative action. A small value for TM_LAG is usually sufficient to achieve the desired result. If TM_LAG ≤ CYCLE/2 is configured, the time delay is deactivated.

The step response within the time range is:

$$PID_OUTV(t) = GAIN * ER_{normalized}(0)\left(1 + \frac{TD}{TM_LAG} * e^{-\frac{t}{TM_LAG}}\right)$$



PID_OUTV(t)	Manipulated variable when controller is in automatic mode
GAIN	Controller gain
$ER_{scaled}(0)$	Step height of the scaled control deviation
TD	Derivative time
TM_LAG	Time delay

PID control

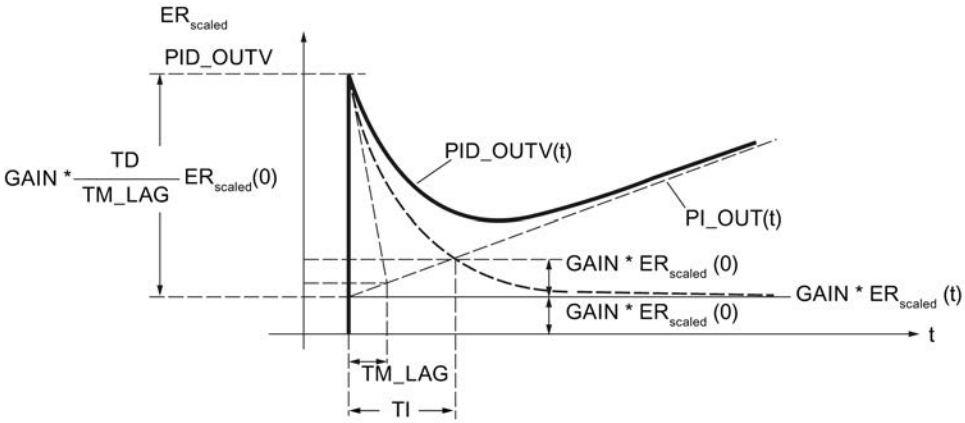
In the proportional-integral-derivative controller (PID-controller), the proportional action, integral action, and derivative action are enabled (P–SEL = TRUE, I–SEL = TRUE, D_SEL = TRUE). A PID controller adjusts output variable PID_OUTV using the integral action until control deviation ER = 0. However, this only applies if the output variable does not exceed the limits of the control range. If manipulated variable limits are exceeded, the integral action maintains the value that was reached at the limit (anti-reset windup).

The PID-controller maps scaled input variable $ER_{scaled}(t)$ proportionally to the output signal and adds to this the components generated by differentiation and integration of $ER_{scaled}(t)$; these components are calculated with double precision according to the trapezoid rule (Padé approximation). The time response is determined by the derivative-action time TD and integral-action time TI.

For signal smoothing and suppression of interfering signals, a first-order delay (adjustable time constant: TM_LAG) is integrated into the algorithm for generating the derivative action. A small value for TM_LAG is usually sufficient to achieve the desired result. If TM_LAG ≤ CYCLE/2 is configured, the time delay is deactivated.

The step response within the time range is:

$$PID_OUTV(t) = GAIN * ER_{normalized}(0)\left(1 + \frac{1}{TI} * t + \frac{TD}{TM_LAG} * e^{-\frac{t}{TM_LAG}}\right)$$



PID_OUTV(t)	Manipulated variable when controller is in automatic mode
GAIN	Controller gain
$ER_{scaled}(0)$	Step height of the scaled control deviation
TI	Integral-action time constant
TD	Derivative time
TM_LAG	Time delay

Note

If TD is changed, you should also adjust TM_LAG accordingly.

Recommendation: 5 ≤ (TM / TM_LAG) ≤ 10

See also

Control algorithm (Page 9675)
Proportional component (Page 9681)
Derivative component (Page 9684)
Integral action (Page 9681)

Proportional component

Proportional gain

The sign of this proportional gain determines the effective direction of the continuous controller.

- Positive proportional gain
  increasing control deviation → rising manipulated variable
- Negative proportional gain
  increasing control deviation → falling manipulated variable

Applying proportional action in feedback path

If the proportional action is applied to the feedback path, only the process value has an effect on the proportional action. Therefore, the manipulated variable does not undergo a step change due to the proportional action during a changeover from manual to automatic (also in the case of a setpoint step-change); the changeover is bumpless.

See also

Control algorithm (Page 9675)

Integral action

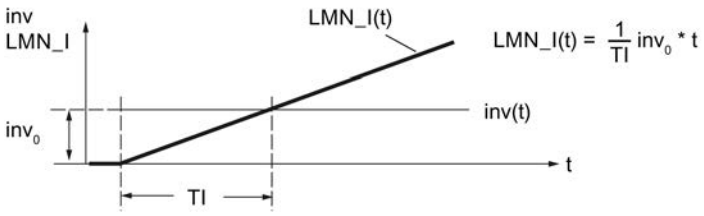
Function of the integral action

The integral action ensures that the control deviation can go to 0 for any manipulated variables by resetting the operating point.

The integral action generates an output signal whose rate of change is proportional to the change in the absolute value of the input variable. The time response is determined by the integral action time TI.

The step response to the input step inv_0 is:

$$OUTV(t) = \frac{1}{TI} \int inv(t)\, dt$$



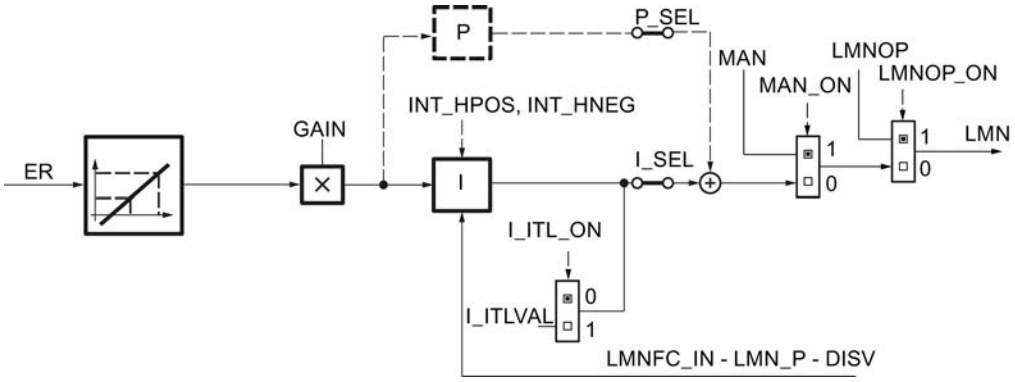
LMN_I(t)	The output variable of the integral action
inv_0	The step height at the input of the integral action
TI	Integral action time

The integral action can be monitored at the static tag LMN_I.

In the deactivated state (I_SEL = FALSE), the integral action, i.e., the internal memory and the static tag LMN_I, is set to zero.

The output and the memory of the integral action are limited by the absolute limits of the manipulated variables LMN_HLM and LMN_LLM (Anti Reset Wind–up).

If the actuating signal is influenced in manual mode, i.e., if MAN_ON, LMNOP_ON, or CAS_ON = TRUE, the internal memory value of the integral action is corrected to the value LMNFC_IN - LMN_P - DISV.



Smooth changeover from manual to automatic

In manual mode, the integral action of the controller is corrected in such a manner that the controller starts with an appropriate manipulated variable upon switchover to automatic mode.

- Smooth changeover enabled (SMOO_CHG = TRUE)
  The integral action is set in manual mode in such a way that the manipulated variable initially remains unchanged on the switchover from manual to automatic.
  In manual mode, the derivative action is set to zero. The switchover to automatic is performed without manipulated variable step.
  A pending system deviation will be slowly corrected.
- Smooth changeover disabled (SMOO_CHG = FALSE)
  The manipulated variable makes a step from the actual position during the switchover from manual to automatic. The height of this step is determined by the proportional action. The step height of the proportional action corresponds to the manipulated variable change on a setpoint step change from the current process value to the current setpoint.
  In manual mode, the derivative action is set to a value that corresponds to the pending control deviation. The switchover to automatic mode is performed with a manipulated variable step; the control deviation is corrected faster.
  However, if the proportional action is applied to the feedback path, only the process value has an effect on the proportional action. Therefore, the manipulated variable does not undergo a step change due to the proportional action during a switchover from manual to automatic; the changeover is smooth. The same applies to the derivative action, if this was moved to the feedback path (DFDB_SEL = TRUE).
  The pending control deviation is corrected faster. This is desirable, for example, for temperature control processes.

Permissible ranges for the integral action time TI and CYCLE

The limited accuracy of the REAL numbers calculated in the CPU means that the following can occur during integration: If the selected sampling time CYCLE of the control is too short in comparison with the integral action time TI and the input value inv of the integral action is small compared to its output value OUTV, the integral action does not respond and remains at its current output value.

This effect can be avoided if the following sizing rule is observed during configuration:

$CYCLE > 10^{-4} \times TI$

As a result, the integral action still responds to changes in the input values in the range of millionths of one part per thousand of the current output variable:

$inv > 10^{-10} \times OUTV$

In order for the response characteristic of the integrator algorithm to conform to the analog response, the sampling time should be less than 20% of the specified integral action time or TI should be at least five times the value of the specified sampling time:

$CYCLE < 0.2 \times TI$

The algorithm permits sampling time values up to $CYCLE \leq 0.5 * TI$.

Initializing the integral action

If initialization of the integral action is enabled, the initialization value is switched through to the static tag LMN_I at a restart of the CPU or COM_RST = TRUE. On the transition to normal operation, the integral action starts the integration of its input variable from the initialization value due to the resetting of the initialization.

Hold integral action

The integral action can be blocked in positive or negative direction by the static tags INT_HPOS and INT_HNEG. This can be useful for controller cascades. If, for example, the manipulated variable of the follow-up controller approaches the high limit, a further increase in the manipulated variable of the master controller can be prevented by its integral action. You program this response using the following instructions, for example, in STL:

```
A    "Follow-up controller".QLMN_HLM
=    "Master controller".INT_HPOS
A    "Follow-up controller".QLMN_ LLM
=    "Master controller".INT_HNEG
```

See also

Control algorithm (Page 9675)

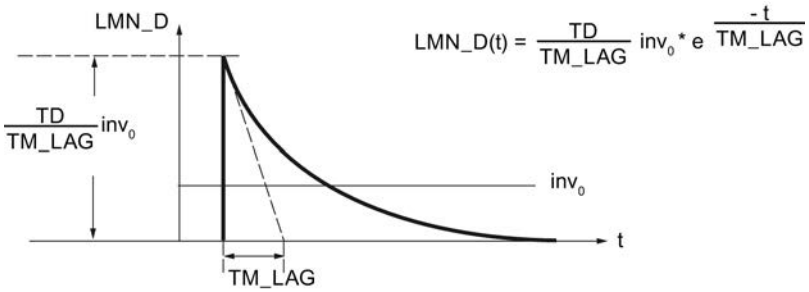
Controller structures (Page 9677)

Derivative component

Function of the derivative action

The derivative action generates an output signal whose size changes proportionally to the rate of change of the input variable. The time response is determined by the derivative time TD and the time delay TM_LAG.

The step response to the input step inv_0 is:



LMN_D(t)	The output variable of the derivative action
inv_0	The step height at the input of the derivative action
TD	Derivative-action time
TM_LAG	Time delay

The derivative action can be monitored at the static tag LMN_D.

In the disabled state (D_SEL = FALSE), the derivative action, i.e., the internal memory and the static tag LMN_D, is set to zero.

Apply derivative action in feedback path

If the derivative action is applied to the feedback path, only the process value has an effect on the derivative action. Consequently – as in a setpoint jump – the manipulated variable does not undergo a step change due to the derivative action during a changeover from manual to automatic; the switchover is smooth.

Permitted values for derivative time and time delay.

To calculate the derivative action correctly, the following conditions must be satisfied for the derivative time and time delay:

- TD ≥ CYCLE and
- TM_LAG ≥ 0.5 × CYCLE

If TD < CYCLE, the derivative action operates as though the TD has the value CYCLE.

If TM_LAG < 0.5 × CYCLE, the derivative action operates without delay. An input step is then multiplied with the factor TD/CYCLE and this value is applied as "needle pulse" at the output. In other words, LMN_D is reset to zero in the subsequent processing cycle.

Coefficient DT1

The coefficient DT1 is calculated as follows:

Coefficient DT1 = TD/TM_LAG

Time delay

For signal smoothing and suppression of interfering signals, a first-order delay (adjustable time constant: TM_LAG) is integrated into the algorithm for generating the derivative action.

A small value for the time delay is usually sufficient to achieve the desired result. If the assigned time delay is less than half the controller sampling time, the delay is deactivated.

Procedure

To configure the derivative action, proceed as follows:

1. To apply the derivative action to the feedback path, select the corresponding check box.
2. Enter the derivative time in seconds.
3. Enter the time delay in seconds.

See also

Control algorithm (Page 9675)

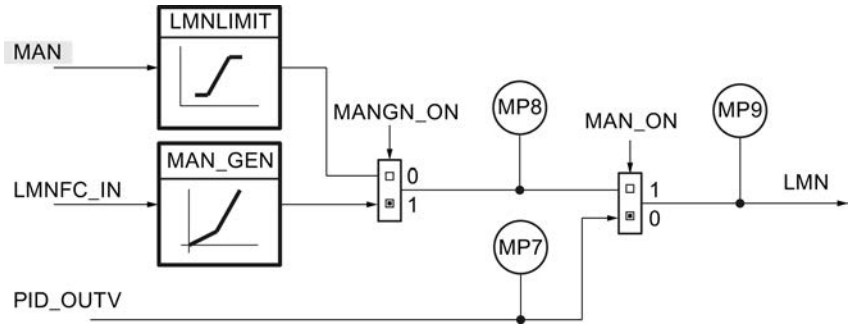
Manipulated variable

Manual mode

In addition to "automatic" mode, in which the output of the PID algorithm (PID_OUTV) is switched through to the output, there are two other operating modes in which the actuating signal can be influenced manually: "Manual mode without switch operation" and "Manual mode using more/less switch" (MAN_GEN).

The MAN parameter can be used to switch through an external manipulated variable or to specify it from the user program. Input value MAN is limited by the manipulated variable limits (LMN_HLM (high) and LMN_LLM (low)).

The structure of the manual value generation and manual value application emerges from the following figure. When MAN_GEN is activated from another mode, the current manipulated variable in effect is applied at the output of MAN_GEN. The changeover to the manual value generator is thus always bumpless.



Activating automatic mode

If manual mode is disabled (MAN_ON = FALSE), the manipulated variable of the PID algorithm is switched through to the output. In manual mode (MAN_ON = TRUE), the integral action of the controller is corrected such that the controller begins with an appropriate manipulated variable on changeover to automatic mode. The output of the PID algorithm is stored in measuring point MP7.

In automatic mode, the manual value MAN of the manipulated variable (minus the derivative action) is corrected. On changeover to manual mode, the manipulated variable is kept at the last calculated value. It can be changed via operator input.

1. Clear the "Activate manual mode" check box.

Activate manual mode with manual value

In this mode (MANGN_ON = FALSE and MAN_ON = TRUE), the manual output value is entered as an absolute value in the MAN input. The manual output value is displayed at measuring point MP8.

1. Select the "Activate manual mode" check box.
2. Choose the "Manual value" option.

Activate the manual value generator

The manual value MAN can be incremented or decremented in steps using the MANUP and MANDN switches. The modified value can be monitored at MP8. The configured manipulated variable limits are taken into account.

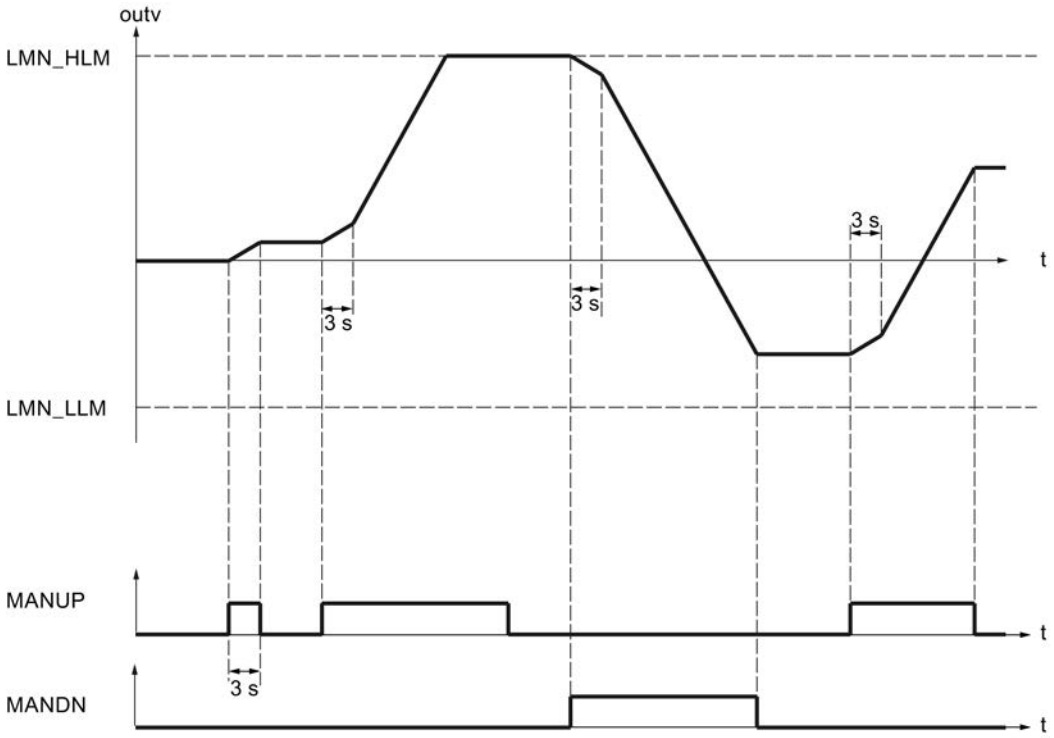
1. Select the "Activate manual mode" check box.
2. Select the "Manual value generator" option.

Effect of the switches MANUP and MANDN

For changes involving small steps, the controller should have a maximum sampling time of 100 ms.

The rate of change of the manipulated variable depends on how long the MANUP or MANDN switch is actuated and on the selected limits. The following relationship applies:

- During the first 3 s after setting MANUP or MANDN:
  doutv/dt = (LMN_HLM - LMN_LLM) / 100 s
- Afterwards:
  doutv/dt = (LMN_HLM - LMN_LLM) / 10 s



For a sampling time of 100 ms and a manipulated variable range of -100.0 to 100.0, the manipulated variable, for example, changes by 0.2 per cycle during the first 3 seconds. If MANUP is TRUE for more than 3 seconds, the rate of changes increases to ten times the value, in this case therefore to 2 per cycle.

The manual value generator is activated bumpless.

Operating modes

The following table shows the possible operating modes of the continuous controller with the required values of the switches.

Operating mode	MAN_ON	MANGN_ON
Automatic mode	FALSE	Not relevant
Manual mode with manual value	TRUE	FALSE
Manual mode with manual value generator	TRUE	TRUE

Limiting functions

Calling a function

If you insert a user-specific function into the manipulated variable branch, the manipulated variable PID_OUTV generated in the controller can undergo signal processing (e.g., a signal delay) prior to application to the output of the controller.

The controller calls the function. In so doing, existing input/output parameters of the user function are not supplied. You have to program the data transfer in your function yourself.

Note

A check is not performed to determine whether a function exists. If the function does not exist, the CPU switches to STOP with an internal system error.

1. Select the "Enable FC call" check box.
2. Enter the number of the function in the "FC" input field.

Limiting the rate of change

Ramp functions will be used in the manipulated variable branch if the process does not tolerate abrupt changes in the controlled system input signal. These abrupt manipulated variable changes must be avoided, for example, when a gearbox is connected in between an adjustable-speed motor and the load to be driven and an excessively rapid increase of the motor speed would result in overloading of the gearbox.

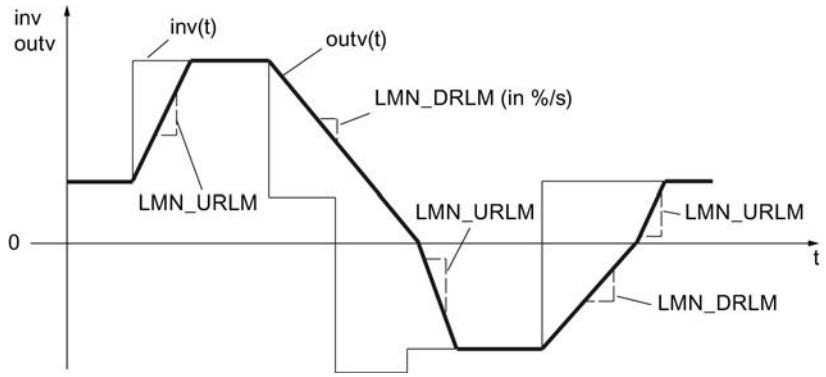
The rate of change of the manipulated variable at the output of the controller is limited separately for the rising and falling slope. Two ramps with rising or falling values can be assigned for the entire value range, when seen from the zero point. The function is enabled if LMNRC_ON = TRUE.

The limiting for each slope of the ramp function in the positive range and in the negative range of the manipulated variable is entered in the two static variables LMN_URLM and LMN_DRLM. The slopes refer to a rising slope or falling slope in percent per second. Faster manipulated variable changes are ramped down to these maximum rates.

For example, if 'LMN_URLM' is assigned the value 10.0 [%/s], the following values are added to the "old value" of outv in each sampling cycle for as long as |inv| > |outv|:

Sampling time	1 s	→	$outv_{old}$ + 10%
	100 ms	→	$outv_{old}$ + 1%
	10 ms	→	$outv_{old}$ + 0.1 %

The figure below shows how the signal processing works: The step functions at input inv (t) become ramp functions at output outv (t).



When slope limits are reached, this is not signaled.

The ramp parameters are identified in accordance with the following scheme:

Parameter	Ramp
LMN_URLM	|outv| rising
LMN_DRLM	|outv| falling

1. Activate the "Limit rate of change" check box.
2. Enter the maximum increase of the manipulated variable in the "Maximum increase" field. Faster manipulated variable changes will be ramped down to this value.
3. Enter the maximum decrease of the manipulated variable in the "Maximum decrease" field. Faster manipulated variable changes will be ramped down to this value.

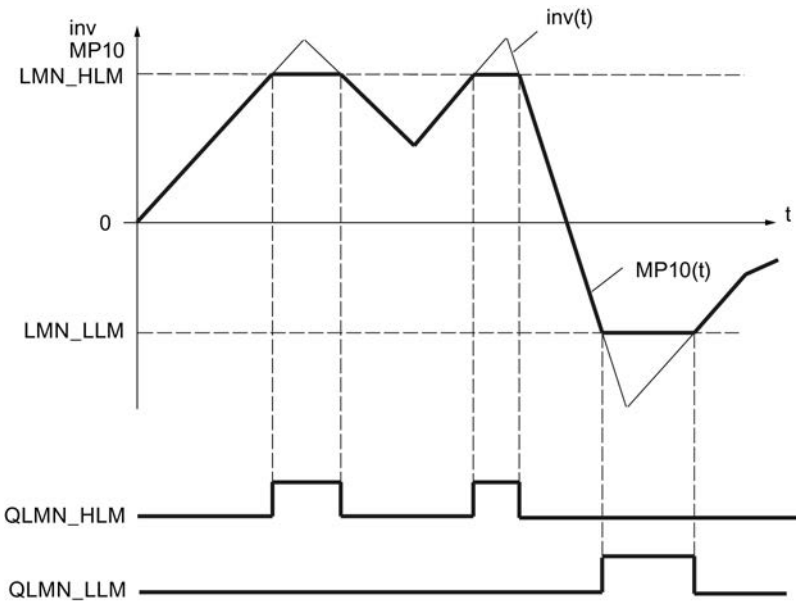
Limiting the absolute value of the manipulated variable

The operating range, i.e., the range in which the actuator can vary within the scope of permissible manipulated variables, is determined by the setting range of the manipulated variable. Since the limits for allowed manipulated variables usually do not coincide with the 0% or 100% limit of the correcting range, depending on the type of actuator, you can configure the high and low limit of the manipulated variable.

To avoid an impermissible status in the particular process, high and low limits for the setting range of the manipulated variable will be set in the manipulated variable branch of the closed-loop control.

The manipulated variable LMN(t) is limited to the assignable high limit (LMN_HLM) and low limit (LMN_LLM). However, the input variable inv must be outside these limits. Because the function cannot be deactivated, the specification of a low limit and high limit must be taken into consideration in the configuration.

In the event of violations by input variable inv(t), the associated displays are output via the message outputs.



Startup and operating principles

- During a restart, all message outputs are set to zero.
- The limiting operates according to the following relationships:

LMN =	QLMN_HLM	QLMN_LLM	If:
LMN_HLM	TRUE	FALSE	inv ≥ LMN_HLM
LMN_LLM	FALSE	TRUE	inv ≤ LMN_LLM
inv	FALSE	FALSE	LMN_HLM < inv < LMN_LLM

The effective manipulated value of the controller is displayed at the output, i.e. at the parameter LMN or at the measuring point MP10.

Procedure

To limit the absolute value of the manipulated variable, proceed as follows:

1. In the "High limit" input field, enter the value for the maximum permissible value.
2. In the "Low limit" input field, enter the value for the minimum permissible value.

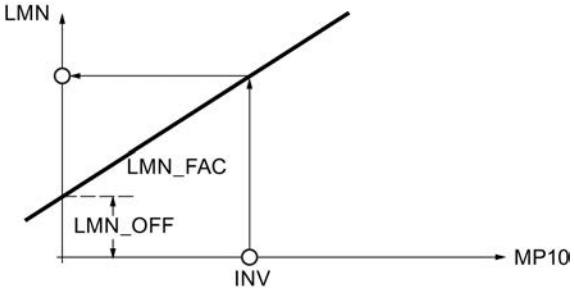
Scaling

If a physical quantity is required for the manipulated variable at the controlled process input, the floating-point values in the range from 0 to 100% must be unscaled to the physical range (e.g., 150 to 3000 rpm) of the manipulated variable.

Two parameters are defined for the unique specification of the scaling levels:

- The factor for the slope (LMN_FAC)
- The offset of the scaling level at the zero point (LMN_OFF)

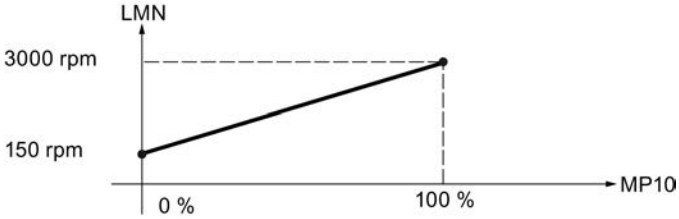
Internal percentage values (in REAL format) ⇒ external physical values



The scaling value is calculated from the respective input value MP10 according to the following function:

LMN = MP10 * LMN_FAC + LMN_OFF

Example



$$LMNFAC = \frac{(3000 - 150)\ rpm}{(100 - 0)\%} = 28.5\ \frac{Rpm}{\%}$$

LMNOFF = 150 rpm

Values are not limited internally in the function and the parameters are not checked.

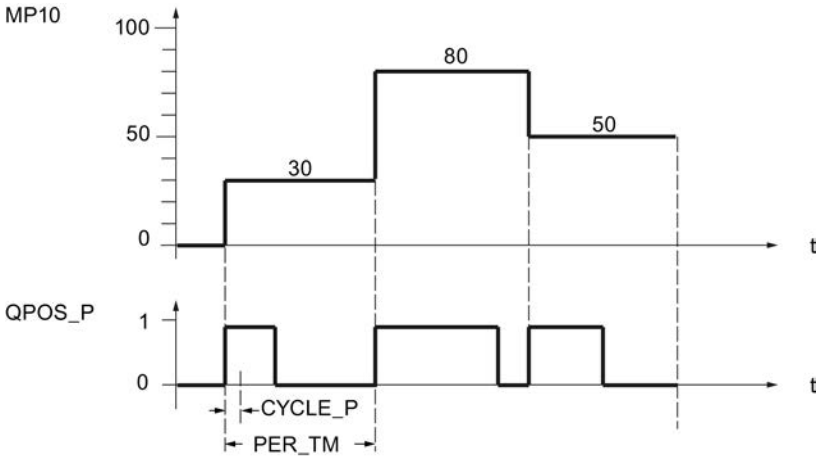
Performing scaling

1. Enter the slope of the scaling level in the "Factor" input field.
2. Enter the offset of the scaling level with respect to the zero point in the "Offset" input field.

Pulse generator

Application

The pulse generator transforms the input variable “Manipulated variable of PID controller at measuring point MP10" by modulating the pulse width into a pulse train with a constant period. The duration of a pulse per period is proportional to the input variable.



An input variable of 30% and 10 calls of the pulse generator per period therefore means:

- "One" at the QPOS output for the first three calls (30% of 10 calls).
- "Zero" at the QPOS output for seven further calls (70% of 10 calls).

Operating modes of controller with pulse output

Depending on the parameter assignment of the pulse generator, PID controllers with a three-step response or with a bipolar or unipolar two-step output can be configured. The following table illustrates the setting of the switch combinations for the possible operating modes.

Operating mode	Switch		
	MAN_ON	STEP3_ON	ST2BI_ON
Three-step control	FALSE	TRUE	Any
Two-step control with bipolar Correcting range (-100% to 100%)	FALSE	FALSE	TRUE
Two-step control with unipolar Correcting range (0% to 100%)	FALSE	FALSE	FALSE
Manual mode	TRUE	Any	Any

Three-step control

In "Three-step control" mode, three control signal states can be generated, e.g., depending on the actuator and process: More – OFF – Less, Forward – Stop – Backward, Heat – OFF – Cool, etc. The status values of the binary output signals QPOS_P and QNEG_P are assigned

to the respective operating states of the actuator, based on the requirements of the process being controlled. The table shows two examples.

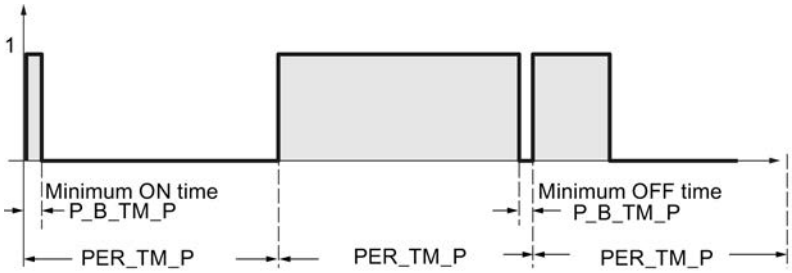
	Heat Forward	Off Stop	Cool Backward
QPOS_P	TRUE	FALSE	FALSE
QNEG_P	FALSE	FALSE	TRUE

Selecting sizes for the minimum pulse time or minimum break time P_B_TM_P can prevent very short activation or deactivation times, which reduce the operating life of switch elements and actuating devices considerably. In so doing, a response value for the pulse output is superimposed on the proportional output characteristic curve used to calculate the pulse duration.

Note

Small absolute values of the "Manipulated variable of the PID controller at measuring point MP10" input variable that would generate a pulse duration less than P_B_TM_P are suppressed. For large input values that would generate a pulse duration longer than PER_TM_P – P_B_TM_P, the pulse duration is set to 100% or -100%.

Set values P_B_TM ≤ 0.1 * PER_TM are recommended.

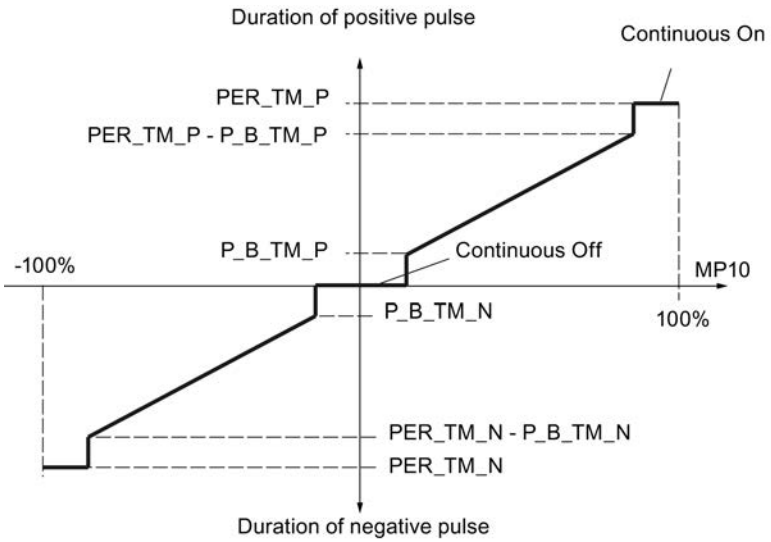


The duration of the positive or negative pulses are calculated by multiplying the "Manipulated variable of PID controller at measuring point MP10" input variable (in %) by the period:

$$\text{Pulse duration} = \frac{\text{MP10}}{100} * \text{PER_TM_P[s]}$$

The suppression of the minimum pulse time or minimum break time causes bend points near the start and end of the conversion characteristic curve.

The above statements also apply to P_B_TM_N and PER_TM_N.



Three-step control asymmetrical

Using the ratio factor RATIOFAC, the ratio of the duration of positive to negative pulses can be changed. In a thermal process, for example, this would allow different process time constants for heating and cooling to be included.

Assuming the absolute value of the "Manipulated variable of the PID controller at measuring point MP10" is the same, a shorter pulse duration at the negative pulse output than for the positive pulse requires a ratio factor setting less than 1:

Pos. pulse > Neg. pulse: RATIOFAC < 1

Pulse duration negative: $\frac{MP10}{100} * PER_TM_N * RATIOFAC$

Pulse duration positive: $\frac{MP10}{100} * PER_TM_N$

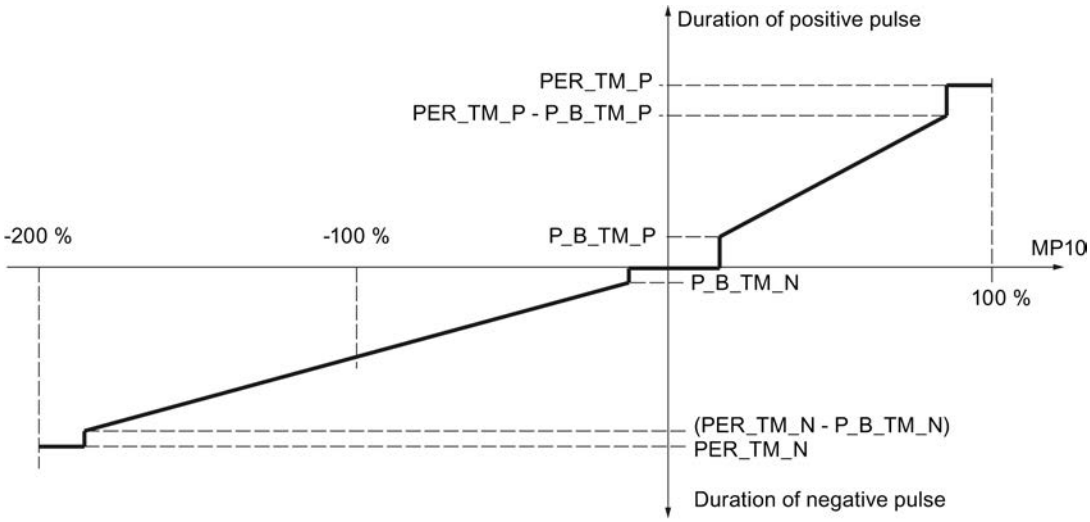


Figure 11-5 Asymmetrical characteristic curve of the three-step controller (ratio factor = 0.5)

On the other hand, for the same absolute value |MP10|, a shorter pulse duration at the positive pulse output compared to the negative pulse requires a ratio factor setting greater than 1:

Pos. pulse < Neg. pulse: RATIOFAC > 1

Pulse duration negative: $\frac{MP10}{100} * PER_TM_N$

Pulse duration positive: $\frac{MP10 * PER_TM_P}{100 * RATIOFAC}$

Mathematically, this means that when RATIOFAC < 1, the response value for negative pulses is multiplied by the ratio factor and when RATIOFAC > 1, the response value for positive pulses is divided by the ratio factor.

Note

For asymmetrical three-step control RATIOFAC ≠ 1, you must adapt the manipulated variable limits according to the following formulas:

RATIOFAC < 1:

LMN_HLM = 100

LMN_LLM = –100 * (1 / RATIOFAC)

RATIOFAC > 1:

LMN_HLM = 100 * RATIOFAC

LMN_LLM = –100

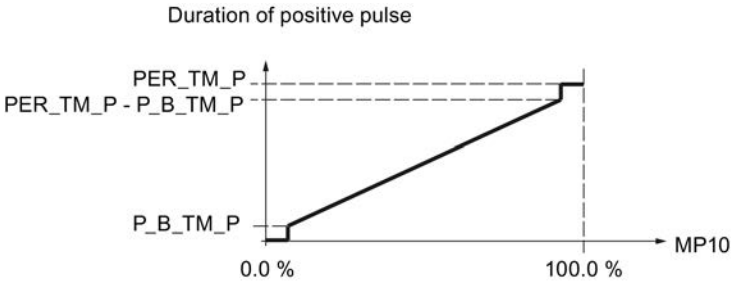
Examples:

RATIOFAC	=	1	RATIOFAC	=	0.5	RATIOFAC	=	2.0
LMN_HLM	=	100	LMN_HLM	=	100	LMN_HLM	=	200
LMN_LLM	=	–100	LMN_LLM	=	–200	LMN_LLM	=	–100

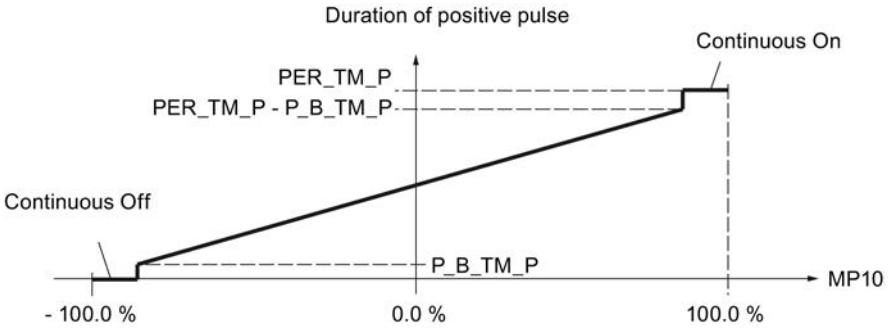
Two-step control

In two-step control, only the positive pulse output QPOS_P is connected to the affected on/off actuator. Depending on the correcting range used, MP10 = -100.0% to +100.0% or MP10 = 0.0% to 100.0%, the two-step controller has a bipolar or unipolar correcting range.

In unipolar operating mode, MP10 may assume values between 0.0% and 100.0% only.



In bipolar operating mode, MP10 may assume values between -100.0% and 100.0%.



The negated output signal is available at QNEG_P in case the interconnection of the two-step controller in the control loop requires a logically inverted binary signal for the actuating pulses.

	On	Off
QPOS_P	TRUE	FALSE
QNEG_P	FALSE	TRUE

Controller sampling time CYCLE and pulse pattern width CYCLE_P

If you have activated the pulse generator stage (PULSE_ON = TRUE), you must specify the cycle clock of the calling cyclic interrupt OB in the CYCLE_P input. The duration of the generated pulse is always an integer multiple of this value.

At the CYCLE input, you specify the sampling time for the remaining control functions of the PID_CP. PID_CP determines the time scaling and processes the control functions with the sampling time CYCLE.

CYCLE must be an integer multiple of CYCLE_P. If you do not adhere to this condition, PID_CP rounds up the sampling time for the control functions to an integer multiple of CYCLE_P. The time-dependent functions (e.g., smoothing, integration, differentiation) will then not work correctly to some extent.

The value selected for CYCLE can be less than the period PER_TM_P or PER_TM_N. This is useful if the longest possible period is desired to protect the actuators, but at the same time very short sampling times are required due to a fast process.

The same guideline applies here to the selection of the sampling time CYCLE as for continuous controllers without a pulse generator stage: the value of CYCLE should not be less than approximately 10% of the dominating process time constants of the controlled system.

Example of the effect of the CYCLE_P, CYCLE and PER_TM_P or PER_TM_N parameters:

- PER_TM_P = 10 s, CYCLE = 1 s, CYCLE_P = 100 ms.

A new manipulated variable is calculated every second, and the manipulated variable is compared with the previously output pulse length or break length every 100 ms.

If a pulse is output and calculate manipulated variable is greater than the previous pulse length/PER_TM_P, the pulse is extended. Otherwise, no more pulse signals are output. If no pulse is output and (100% - calculated manipulated variable) is greater than the previous break length/PER_TM_P, the break is extended. Otherwise, a pulse signal is output.

As a result of a special pulse generation procedure, an increase or decrease in the manipulated variable during the period causes the output pulse to be extended or shortened. In this case (CYCLE < PER_TM_P), if the configured period is so long that it would cause the process value to fluctuate, the effective period of PID_CP is automatically reduced to an appropriate value.

Accuracy of pulse generation

The period PER_TM is not identical with the processing cycle of the pulse generator. A period is comprised of several processing cycles of the pulse generator. The number of calls of the pulse generator per period is a measure for the accuracy of the pulse width.

The smaller the sampling time CYCLE_P is compared to the period duration PER_TM_P (or PER_TM_N), the more precise the pulse width modulation is. To achieve sufficiently accurate control, the following condition should be met:

$$\text{CYCLE_P} \leq \frac{\text{PER_TM}}{20 \ldots 50}$$

Realization of very short sampling times

A fast process requires very small sampling times (for example, 10 ms). Due to the program runtime in this case, it is not appropriate to process the control functions in the same cyclic interrupt OB where the pulse output is calculated. You then transfer the processing of the control functions either to the OB 1 or to a slower cyclic interrupt OB. The processing of the control functions in OB 1 is only recommended if the cycle time of OB 1 is significantly less than the sampling time CYCLE of the controller.

Use the SELECT parameter to specify which program section is to be processed. The following table provides a quick overview of the parameter assignment of the SELECT input parameter:

SELECT	Block functionality used	Underlying method
0	Control functions and pulse output	Control functions and pulse output in one and the same block
1	Call in OB 1 (control functions)	Control functions in OB 1, pulse output in fast cyclic interrupt OB
2	Call in cyclic interrupt OB (pulse output)	
3	Call in slow cyclic interrupt OB (control functions)	Control functions in slow cyclic interrupt OB, pulse output in fast cyclic interrupt OB
2	Call in fast cyclic interrupt OB (pulse output)	

The methods for achieving very short pulse pattern widths indicated in the table above are described in more detail below.

- Control functions in OB 1, pulse output in cyclic interrupt OB
  When called with SELECT = 2, the pulse output is calculated and a test is performed to determine whether the sampling time configured in CYCLE has already elapsed since the last time the control functions were processed. If the sampling time has elapsed, PID_CP writes the value TRUE in the QC_ACT variable in the instance DB.
  When called with SELECT = 1, the QC_ACT variable is evaluated in the instance DB: If the value of QC_ACT is FALSE, the block is stopped immediately; thus, only a very brief execution time is required. If the value of QC_ACT is TRUE, the control functions are executed once and the QC_ACT is then reset.
  As a result of this procedure, the exact sampling time for the control functions cannot be adhered to. It fluctuates by the amount of the execution time of OB 1 (including all interruptions). This process is therefore only suitable when the execution time of OB 1 is less than the sampling time CYCLE.
- Control functions in slow cyclic interrupt OB, pulse output in fast cyclic interrupt OB
  When called with SELECT = 2, the pulse output is always calculated.
  When called with SELECT = 3, the control functions are always processed.

Note

To prevent having to program the PID_CP instruction call twice, along with its many formal operands, it is advisable to program this call once in an FC that also has a SELECT parameter as an input parameter. Interconnect this input parameter to PID_CP.SELECT. You then call only this FC in OB 1 or in the cyclic interrupt OB.

This procedure is easy to modify and saves program memory.

Commissioning PID_CP

Performing tuning manually

Requirements

- The instruction and the technology object have been loaded to the CPU.
- MAN_ON = FALSE

Procedure

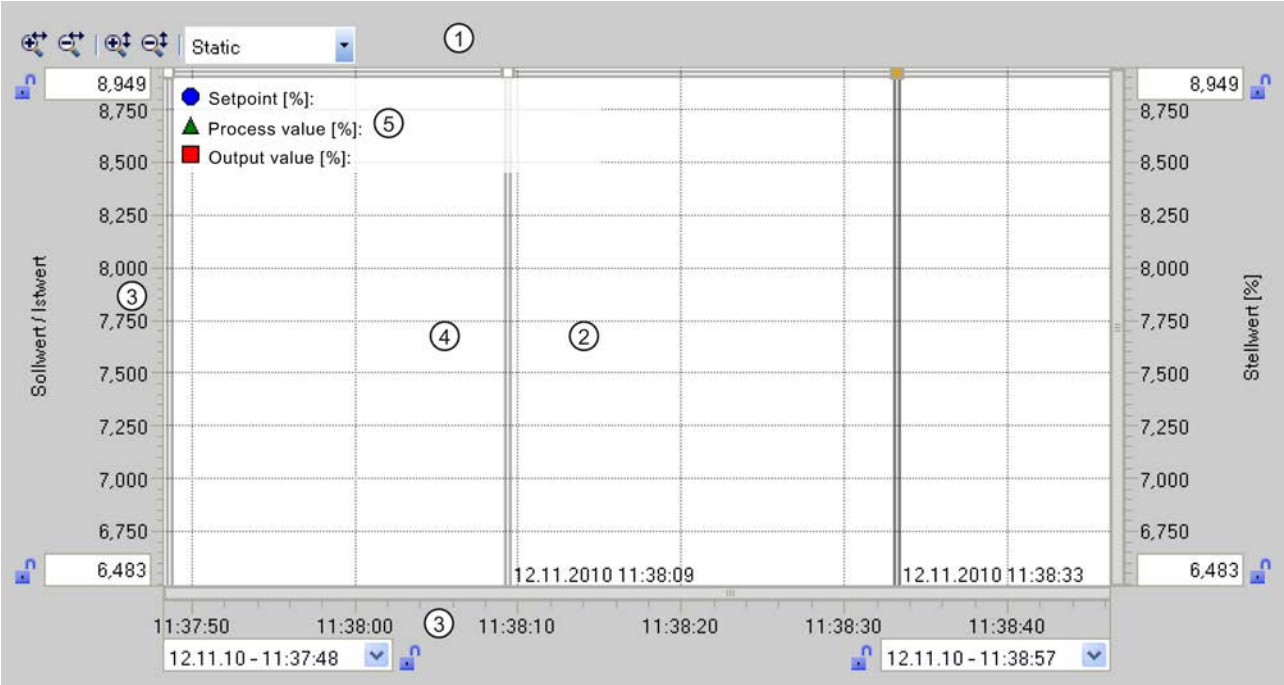
In order to manually determine the optimal PID parameters, proceed as follows:

1. Click the "Measurement on" icon.
   An online connection will be established, if one is not already present. The current values for the setpoint, process value and manipulated variable are recorded.
2. Enter new PID parameters in the "P", "I", "D" and "Time delay" fields.
3. Click on the icon "Send parameter to CPU" in the "Tuning" group.
4. Select the "Change setpoint" check box in the "Current values" group.
5. Enter a new setpoint and click in the "Current Values" group on the icon .
   If the setpoint's rate of change is limited, the changeover to the new setpoint will be bumpless
6. Clear the "Manual mode" check box, if necessary.
   The controller works with the new PID parameters and controls the new setpoint.
7. Check the quality of the PID parameter based on the curve points.
8. Repeat steps 3 to 8 until you are satisfied with the controller results.

Using the trend view

The trend view is used for graphic illustration of the setpoint, process value and output value. The values of the trend view are updated in the selected updating cycle. Start the recording of the trend values by clicking "Measurement on" and stop it by clicking "Measurement off".

Elements of the trend view



①	Selection of the display mode
②	Trend view
③	Area for moving and scaling the axes
④	Ruler
⑤	Legend with the trend values at the ruler

Selection of the display mode

The following display modes can be selected for displaying the trend recording:

- Strip (continuous display)
  New trend values are entered at the right-hand trend view. Previous trend values are scrolled to the left. The time axis cannot be moved.
- Scope (jumping area display)
  New trend values are entered within the trend view from left to right. When the right-hand margin of the trend view is reached, the monitoring area is moved one view width to the right. The time axis can be moved within the limits of the monitoring area.
- Sweep (rotating display)
  New trend values are displayed in the trend view in accordance with a rotating display. The trend values are entered from left to right. The trend values of the last rotating display are overwritten at the writing position. The time axis cannot be moved.
- Static (static area display)
  Writing of the trends is interrupted. Recording of the new trend values is carried out in the background. The time axis can be moved across the entire previous recording period.

The ratio of the visible area to the recorded area is shown in the ratio display:

The width of the ratio display represents the complete duration of the recording. The yellow section of the display represents that part of the recording that is visible in the trend view. The grayed section of the display represents that part of the recording that is **not** visible in the trend view.

In the "Static" display mode, you can move the visible range of the ratio display along the axis by dragging it with the mouse.

Trend view

The trends for the setpoint, process value and output value are displayed in the trend view. In addition to different colors, the trends are identified by symbols (refer to legend).

Moving and scaling the axes

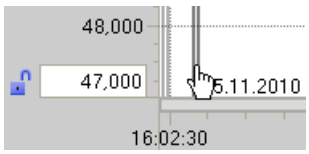
The axes for the setpoint and process value, for the output value and the time axis can be moved and scaled individually. The right and left mouse buttons are assigned alternating functions. You can use the following icons and mouse actions:

	Moving the setpoint / process value axes, or the output value axis up or down. The axis cannot be moved if the high or the low scaling value is blocked.
	Moving the time axis to the right or left. The axis cannot be moved if the left or right scaling value is blocked.
	Stretch and compress the setpoint , process value axes, or the output value axis. • The scaling of the axes is stretched or compressed symmetrically if none of the scaling values is blocked. • Blocked scaling values are retained when you stretch or compress an axis.
	Stretching and compressing the time axis • The scaling of the axis is stretched or compressed symmetrically if none of the scaling values is blocked. • Blocked scaling values are retained when you stretch or compress an axis.
	Stretching and compressing the setpoint / process value axes, or the output value axis. The low scaling value is not changed as a result of stretching or compressing.
	Stretch and compress the setpoint / process value or the output value axis. The high scaling value is not changed during stretching or compressing.
	Stretching and compressing the time axis. The right scaling value is not changed as a result of stretching or compressing.
	Stretching and compressing the time axis. The left scaling value is not changed as a result of stretching or compressing.
50,000	Enter a scaling value.
50,000	The current scaling value can be blocked using the padlock icon. Only one value of a respective axis can be blocked.
	Double-clicking in the trend view optimizes the scaling and position of the setpoint, process value and output value in the trend view.
	Double-clicking in the axis area restores the default position and scaling of the axis.

Working with rulers

Use one or several rulers to analyze discrete values of the trend profile.

The park positions for vertical rulers are found on the left edge of the curve area, for horizontal rulers on the high edge.



Move the mouse to the left edge of the trend area and watch out for the transition of the mouse pointer. Drag a vertical ruler onto the measuring trend that you want to analyze.

The trend values on the vertical ruler are displayed on the left of the ruler. The time of the ruler position is displayed at the base of the ruler.

The trend values of the active ruler are displayed in the legend. If several rulers are dragged to the trend area, the respectively last ruler is active. The active ruler is indicated by the correspondingly colored symbol.

However, the values of horizontal rulers cannot be displayed.

You can reactivate an inactive ruler by clicking it.

Use the shortcut ALT+Click to remove rulers which are no longer required.

Commissioning with the PID Self Tuner

You can commission the PID_CP instruction and determine the optimum PID parameters with the PID Self Tuner.

Procedure

To commission a continuous controller, proceed as follows:

1. Call the instruction PID_CP in a cyclic interrupt OB and interconnect the input and output parameters.
2. Call the instruction TUN_EC in the same cyclic interrupt OB.

3. Interconnect the instruction TUN_EC as follows with PID_CP:
   - TUN_EC.PV with PID_CP.PV
   - TUN_EC.LMN with PID_CP.LMN
   - TUN_EC.SP_OUT with PID_CP.SP_IN or PID_CP.SP_EXT
   - TUN_EC.GAIN with PID_CP.GAIN
   - TUN_EC.TI with PID_CP.TI
   - TUN_EC.TD with PID_CP.TD
   - TUN_EC.TM_LAG with PID_CP.TM_LAG
   - If you are using a three-step controller, TUN_EC.RATIOFAC with PID_CP.RATIOFAC
   - TUN_EC.QMAN_ON with PID_CP.MAN_ON
4. Click the "" icon on the instruction TUN_EC.
   The commissioning editor for TUN_EC opens. The operation of the commissioning editor is described with the PID Self Tuner.

Using PID_ES

Technology object PID_ES

The PID_ES technology object provides a step controller for automatic and manual mode. It corresponds to the instance data block of the instruction PID_ES. You can configure the following controllers:

- Fixed setpoint control with P, PI, PD, PID step controllers
- Step controllers with and without position feedback
- Fixed setpoint control with disturbance variable selection
- Cascade control with step controller in the follow-up circuit
- Two-loop ratio control
- Blending control

You configure the following functions of the technology object PID_ES:

- Scaling the process value signal
- Suppression of high-frequency vibrations in the process value branch by smoothing (time delay) of the process value signal.
- Linearization of the quadratic functions of the process value (flow-rate control with differential pressure encoders)
- Calling of "own functions" in the process value branch
- Monitoring the process value for violation of alarm and warning limits
- Monitoring the rate of change of the process value
- Adjustment of the setpoint by ramp/soak
- Scaling an external setpoint signal with ratio controllers

- Calling of "own functions" in the setpoint branch
- Limiting the rate of change of the setpoint
- Limiting the absolute values of the setpoint
- Filtering the control deviation signal with a deadband
- Monitoring the control deviation for violation of alarm and warning limits
- The proportional, integral, and derivative actions can be activated separately.
- Option of proportional and derivative effect in the controller feedback path.
- Selecting a disturbance variable
- Enabling position feedback
- Control of the manipulated variable in manual mode from a PG or HMI)
- Limiting the absolute values of the manipulated variable
- Taking into account actuator-specific times when generating the pulse

Control deviation generation

Process value

Scaling (PV_NORM)

The "I/O process value ON" check box is used to specify which analog input variables are scaled to the physical unit of the process value.

- Activated: Input variable is the I/O process value (PV_PER)
- Disabled: Input variable is the internal process value (PV_IN)

In order to uniquely specify the scaling levels, you define the following four parameters:

- High input (NM_PIHR)
- Low input (NM_PILR)
- High output (NM_PVHR)
- Low output (NM_PVLR)

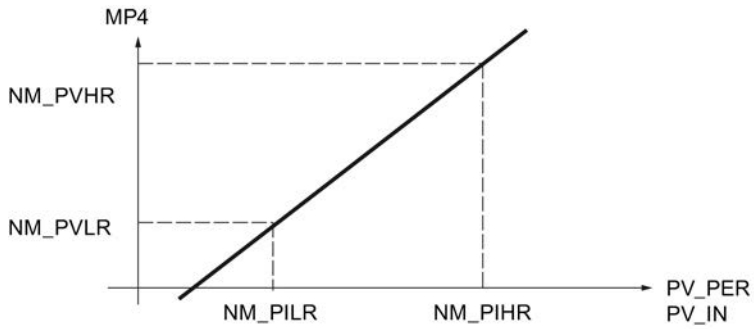
The parameters are preset depending on the sensor type.

The process value is calculated from the respective input value (PV_PER or PV_IN) according to the following formula:

MP4 = (PV_PER - NM_PILR) × (NM_PVHR - NM_PVLR) / (NM_PIHR - NM_PILR) + NM_PVLR

or

MP4 = (PV_IN - NM_PILR) × (NM_PVHR - NM_PVLR) / (NM_PIHR - NM_PILR) + NM_PVLR



The process value is output in the static variable MP4.

No values are limited and the parameters are not checked. If you enter the same value for input high and input low, a division by zero may occur based on the formulas above. The instruction will not detect this.

Filter functions

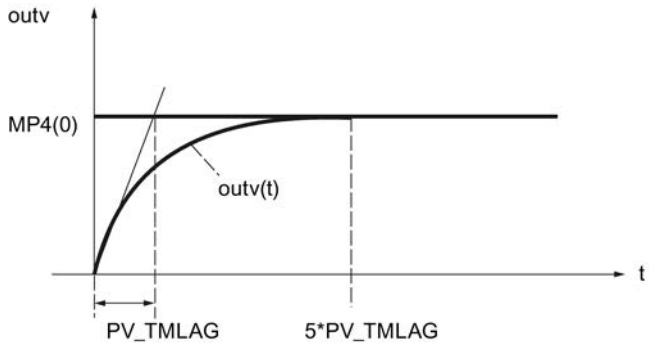
Smooth controlled variable (LAG1ST)

In order to smooth process signals with superimposed rapid variations (noise), a first-order delay element is applied in the process value branch. This function stabilizes the analog controlled variable to a greater or lesser degree, depending on the time constant (PV_TMLAG). In this way, disturbances are effectively suppressed. As a result of this, however, the time constant of the overall control loop also increases, i.e., the closed-loop control becomes more sluggish.

The stabilizing effect is achieved by a first-order delay algorithm.

The step response within the time range is:

$outv(t) = MP4(0)\ (1 - e^{-t/PV_TMLAG})$



MP4(0)	The height of the process value step at the input
outv(t)	The output variable
PV_TMLAG	The time lag constant
t	Time

Smoothing of the process value

If smoothing of the process value (LAG1STON = FALSE) is not activated, then the I/O input (PV_PER) or the internal input (PV_IN) is switched through to the process value branch without a delay.

Time lag

The PV_TMLAG input parameter specifies the time lag for the process value. Any value within the value range is permitted.

If PV_TMLAG ≤ 0.5 * CYCLE, there is no longer any lag in effect.

A sampling time (CYCLE) of less than one-fifth of the time lag is required in order to achieve a lag response approaching that of the analog response.

Square root (SQRT)

The "Square root" function (SQRT) is used to linearize the controlled variable. If the process value supplied by an encoder is a physical quantity and there is a quadratic relation between this quantity and the measured process variable, the characteristic of the controlled variable must be linearized prior to any further processing in the controller.

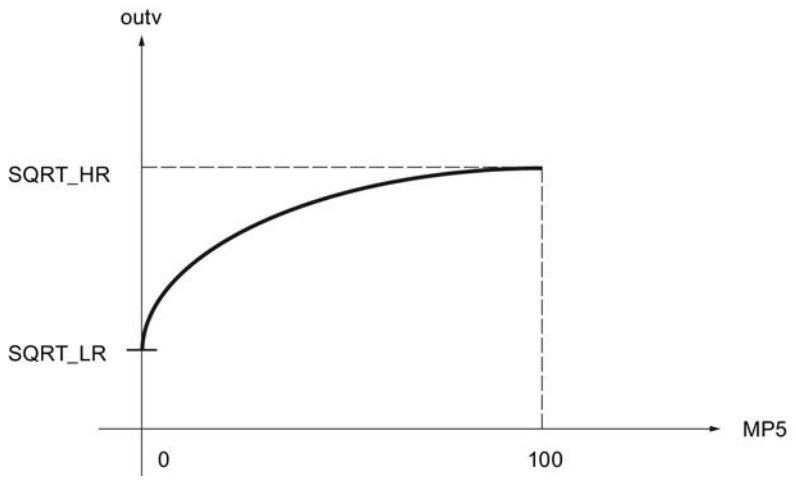
The measured signal must always be linearized by the square-root function if flow measurements are made using metering orifices, venturi tubes, or the like. The measured pressure differential is then proportional to the square of the flow.

The SQRT_ON = TRUE input signal is used to activate the square root function in the process value branch. The square root function algorithm has the following form:

outv = SQRT (MP5) × (SQRT_HR – SQRT_LR) / 10.0 + SQRT_LR

Here, outv is the scaled output value of the square root function (physical flow).

This formula stipulates that the input value of the square root is scaled to the number range 0 to 100. The parameters for the "Measuring range high" and "Measuring range low" must be set to 100.0 and 0.0 in the scaling of the process value. The square root of this value yields a measuring range from 0 to 10. This range of numbers will be scaled using the high limit (SQRT_HR) and the low limit of the physical measurement range (SQRT_LR).



Example of scaling

The input value PV_IN of the controller is the differential pressure in mbar:

Start of measuring range NM_PILR	End of measuring range NM_PIHR	Sample value for PV_IN
20.0 mbar	200.0 mbar	150.0 mbar

The scaled differential pressure is calculated with the help of the scaling function PV_NORM, whereby NM_PVHR = 100.0 and NM_PVLR = 0.0 are selected:

MP4 = (PV_IN - NM_PILR) * (NM_PVHR - NM_PVLR) /
(NM_PIHR - NM_PILR) + NM_PVLR
= (PV_IN - 20.0 mbar) * (100.0 - 0.0) /
(200.0 mbar - 20.0 mbar) + 0.0
= (PV_IN - 20.0 mbar) * 100 / 180.0 mbar

Initial value of MP4	End value of MP4	Sample value for MP4
0.0	100.0	72.222

Since no smoothing is used in this example, the following applies: MP5 = MP4.

For the square root from the scaled differential pressure MP5, the result is:

Initial value according to the square root	End value according to the square root	Sample value for SQRT(MP5)
0.0	10.0	8.498

For the scaled output value outv of the square root function (physical flow-rate) with SQRT_HR = 20000.0 m^3/h und SQRT_LR = 0.0 m3/h, there follows:

outv = SQRT(MP5) * (SQRT_HR - SQRT_LR) / 10.0 + SQRT_LR
= SQRT(MP5) * (20000.0 m^3/h - 0.0 m3/h) / 10.0 + 0.0 m3/h
= 2000.0 m^3/h * SQRT(MP5)

Start of measuring range outv	End of measuring range outv	Sample value for outv
0.0 m^3/h	20000.0 m^3/h	16996.732 m^3/h

FC call in the process value branch (PVFC)

Insertion of a user-specific function in the process value branch allows the controlled variable to undergo signal processing, e.g., signal delay or linearization, prior to any further processing in the controller.

Calling a function

If you want to insert a user-specific function into the process value branch, enable "Enable FC call" (PVFC_ON). The number of the utilized function is entered in the "FC number" input field (PVFC_NBR). The controller calls the function. In so doing, existing input/output parameters of the user function are not supplied. You have to program the data transfer in your function yourself.

You can monitor the output value of the function at the static variable MP6.

Note

No check is made to determine whether a function exists. If the function does not exist, the CPU switches to STOP with an internal system error.

Alarm

Monitor process value for limit values (PV_ALARM)

If the values of the process variables (e.g., speed, pressure, filling level, temperature, etc.) exceed or fall below certain values in closed-loop controls, an illegal process status or plant status can occur. In such cases, monitoring of the process value for limit values (PV_ALARM) is used in order to monitor the process value for violation of the permissible operating range. Limit violations are detected and signaled so that an appropriate reaction can be initiated.

Note

The monitoring of the process value for limit values cannot be deactivated. The violation of the configured limits is always signaled.

Monitoring of the process value for limit values (PV_ALARM) monitors the controlled variable PV(t) for four assignable limits in two tolerance bands. If the limits are reached or exceeded, the function signals a “warning” at the first limit and an “alarm” at the second limit.

Limit values

The numeric values of the limits are set in the input parameters for “warning” and “alarm”. If the process value (PV) exceeds or falls below these limits, the associated output message bits are set for the high alarm (QPVH_ALM), the high warning (QPVH_WRN), the low warning (QPVL_WRN), and the low alarm (QPVL_ALM).

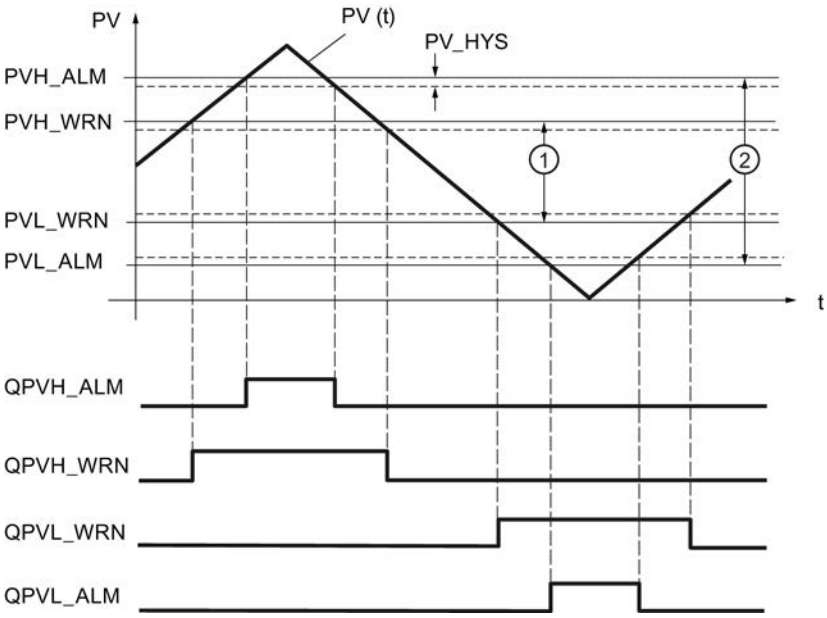
During a restart, the message bits are reset.

The message bits are set as follows:

QPVH_ALM	QPVH_WRN	QPVL_WRN	QPVL_ALM	If:	And:
TRUE	TRUE	FALSE	FALSE	PV ↗ PV ↘	PV ≥ PVH_ALM PV ≥ PVH_ALM - PV_HYS
FALSE	TRUE	FALSE	FALSE	PV ↗ PV ↘	PV ≥ PVH_WRN PV ≥ PVH_WRN - PV_HYS
FALSE	FALSE	TRUE	FALSE	PV ↘ PV ↗	PV ≤ PVL_WRN PV ≤ PVL_WRN + PV_HYS
FALSE	FALSE	TRUE	TRUE	PV ↘ PV ↗	PV ≤ PVL_ALM PV ≤ PVL_ALM + PV_HYS

The following has to apply so that the block operates properly:

High alarm < High warning < Low warning < Low alarm



①	1. Tolerance band
②	2. Tolerance band

Hysteresis

In order to prevent "flickering" of the message bits when the input variable undergoes slight changes or in the event of rounding errors, a hysteresis PV_HYS is used. The controlled variable must violate the hysteresis before the message bits are reset.

Monitor rate of change of the process value (ROCALARM)

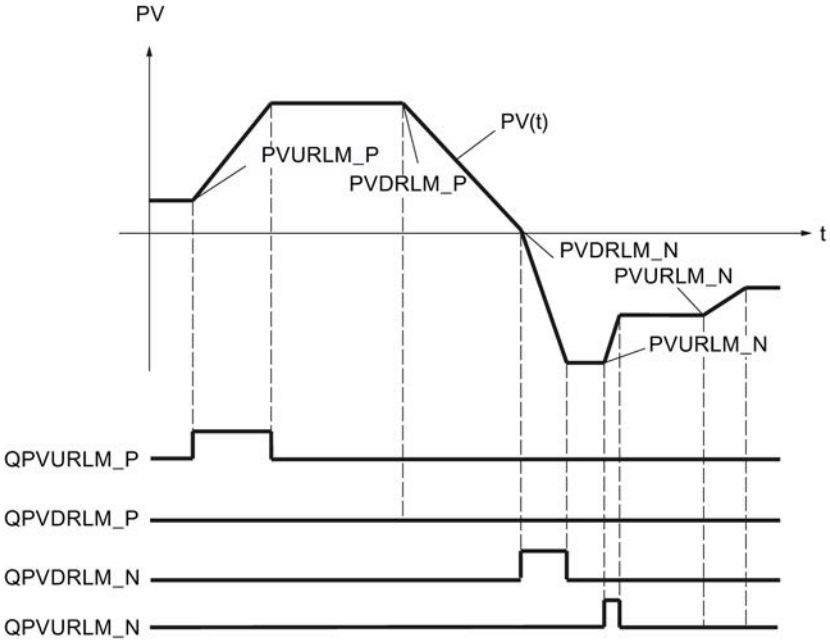
If the change in the process variable (e.g., speed, pressure, filling level, temperature, etc.) is too great in closed-loop controls, an illegal process status or plant status can occur. In such cases, monitoring the rate of change of the process value (ROCALARM) is used to monitor the process value for violations of a permissible rate of change. Limit violations are detected and signaled so that an appropriate reaction can be initiated.

Note

Monitoring of the rate of change of the process value (ROCALARM) cannot be deactivated. The violation of the configured limits is always signaled.

The ROCALARM function monitors the controlled variable PV(t) for permissible rate-of-change limits that are assignable according to sign.

The numeric values of the limit slopes are set at the input parameters for "Maximum increase" and "Maximum decrease" in the positive and negative range of the controlled variable. The slopes refer to a rising slope or falling slope in percent per second. If the rate of change of the controlled variable violates these limits, the associated output message bit QPVURLMP ... QPVDRLMN is set.



The ramp parameters are identified in accordance with the following scheme:

Parameter	PV change
Maximum increase in the positive range (PVURLM_P)	PV > 0 and |PV| rising
Maximum decrease in the positive range (PVDRLM_P)	PV > 0 and |PV| falling
Maximum increase in the negative range (PVURLM_N)	PV > 0 and |PV| rising
Maximum decrease in the negative range (PVDRLM_N)	PV < 0 and |PV| falling

Setpoint

Specifying a setpoint

Select a source for the setpoint. The following options are available:

- Internal setpoint
  The input parameter SP_INT is used as setpoint
- Setpoint generator
  The input parameter SP_INT is used as setpoint and can be incremented or decremented using the static variables SPUP and SPDN.
- Ramp/soak
  The input parameter SP_INT is used as setpoint and can be changed according to a specified ramp/soak.
- External setpoint
  The input parameter SP_EXT is used as setpoint for blending, ratio and cascade controllers. The external setpoint can be scaled.

The effective setpoint is shown at the output parameter SP.

See also

Ramp/soak (RMP_SOAK) (Page 9713)

Setpoint generator (SP_GEN)

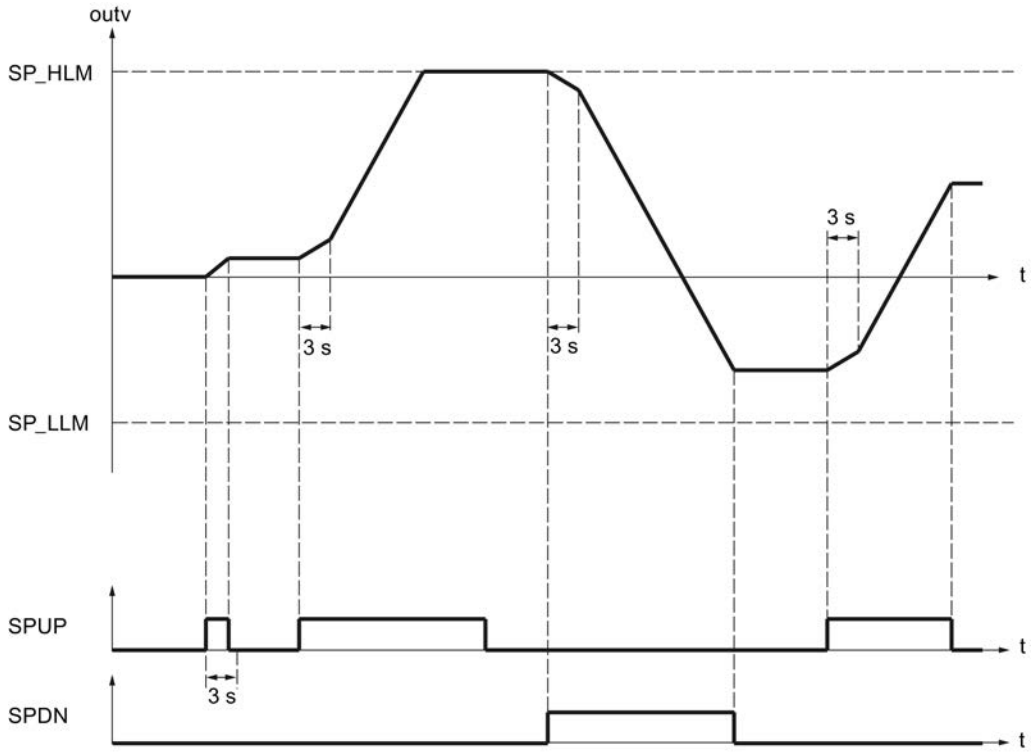
The internal setpoint SP_INT can be incremented or decremented in steps using the SPUP and SPDN switches. The modified value can be monitored at MP1. The configured setpoint limits are taken into account.

Effect of the switches SPUP and SPDN

For changes involving small steps, the controller should have a maximum sampling time of 100 ms.

The rate of change of the output variable depends on how long the SPUP or SPDN switch is actuated and on the selected limits. The following relationship applies:

- During the first 3 s after setting SPUP or SPDN:
  doutv/dt = (SP_HLM - SP_LLM) / 100 s
- Afterwards:
  doutv/dt = (SP_HLM - SP_LLM) / 10 s



For a sampling time of 100 ms and a setpoint range of -100.0 to 100.0, the setpoint, for example, changes by 0.2 per cycle during the first 3 seconds. If SPUP is TRUE for more than 3 seconds, the rate of changes increases to ten times the value, in this case therefore to 2 per cycle.

Startup and operating principles of the setpoint generator

- During a complete restart, output outv is reset to 0.0.
- If you activate the setpoint generator (SPGEN_ON = TRUE), the signal value SPFC_IN is output initially at output outv. Therefore, the switchover to the setpoint generator from another operating mode is always smooth. Provided the SPUP and SPDN (up/down keys) are not activated, the signal value SPFC_IN is retained at the output.

Parameters of the SP_GEN function

Output parameter outv is an implicit parameter. It can be monitored with the configuration tool via measuring point MP1.

Parameter	Meaning	Permitted value range
SPFC_IN	FC input setpoint	Engineering value range
SP_INT	Internal setpoint	Engineering value range

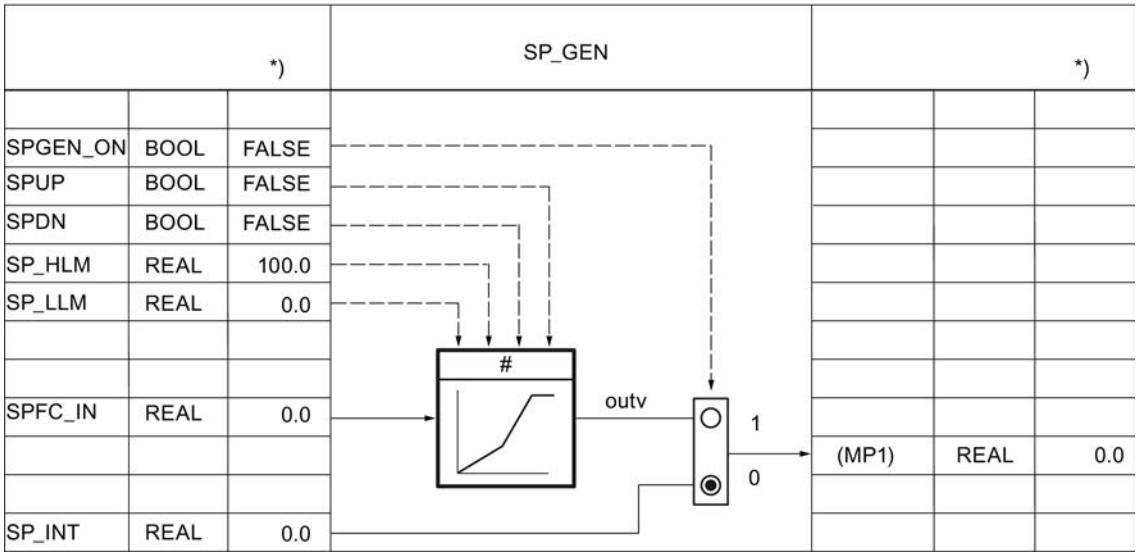


Figure 11-6 Function scheme and parameters of the setpoint generator

Ramp/soak (RMP_SOAK)

If you want the setpoint SP_INT to change autonomously depending on the time, for example, during the control of processes after a time-controlled temperature program, this is possible by configuring a ramp/soak and activating the ramp/soak function. The setpoint change is formed from the level sections with a maximum of 256 time slices.

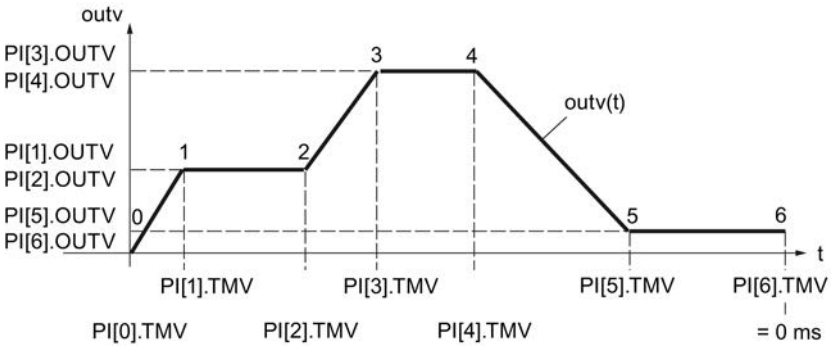
Note

The block function does not check whether a global DB with the number 'DB_NBR' exists or whether the parameter 'Number of time slices' NBR_PTS corresponds to the data block length. If the parameter assignment is incorrect, the CPU goes to **STOP** and issues an internal system error.

How the ramp/soak works

The ramp/soak supplies the output variable outv according to a function that runs to a defined ramp/soak. The ramp/soak process is defined by specifying a series of time slices in a global

data block RMP_SOAK with time values PI[i].TMV and the associated output values PI[i].OUTV. The following figure shows a ramp/soak with 6 time slices.

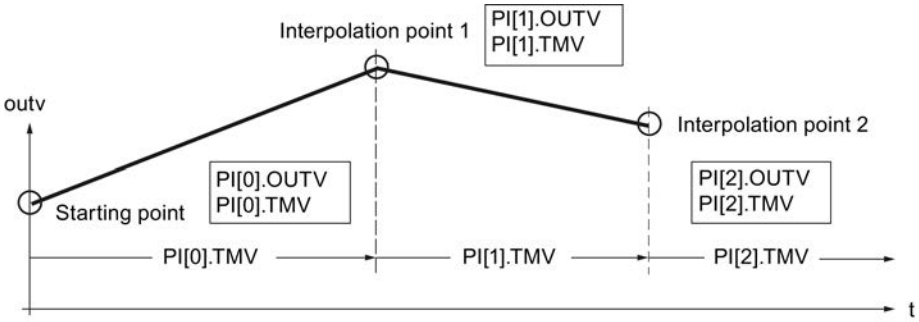


The time value for the last time slice n is PI[n].TMV = 0 ms. The processing time of a ramp/soak is counted down from the initial value to zero.

PI[i].TMV specifies the intervals between time slices. Between the time slices, interpolation is according to the following function:

$$outv(t) = PI[n + 1].\ OUTV - \frac{RS_TM}{PI[n].TMV}\ (PI[n + 1].OUTV - PI[n].OUTV)$$

$0 \leq n \leq (NBR_PTS - 1)$



Note

Note

During the interpolation of the ramp/soak between the time slices, pauses can sometimes occur at the initial value if the sampling time CYCLE is very small compared to the time between the time slices PI[n].TMV. Due to the computational accuracy of the CPU, the ramp/soak function cannot linearize all flat ramp/soak profiles. If the ramp/soak profile is too flat, the initial value pauses for a time at the respective time slice and continues, after a certain time, to integrate with the minimum rate of rise to the next time slice.

To correct or avoid error: Shorten the time between the time slices by inserting additional time slices. The output ramp/soak profile thus approximates the desired flat ramp/soak profile.

Ramp/soak function ON

The ramp/soak function is activated by changing RMPSK_ON from FALSE to TRUE. The ramp/soak is terminated after the last time slice has been reached. When the function is started again by the operator, RMPSK_ON first has to be set to 'FALSE' and then back to 'TRUE' .

During a **restart**, the output outv is reset to 0.0 and the total time or total remaining time is determined. During the transition to normal operation, the ramp/soak profile is processed immediately from the starting point according to the set operating mode. If this is not desired, the parameter RMPSK_ON must be set to FALSE in the restart OB.

Ramp/soak functions

The instruction has the following functions:

- Activate ramp/soak for single execution
- Pre-assign fixed value to the output
- Cyclic repetition on
- Hold ramp/soak
- Continue ramp/soak
- Update the total processing time and remaining total time

The following table applies for the significance of the control inputs when setting a desired operating mode:

Operating mode	RMPSK_ ON	DFOUT_ ON	RMP_ HOLD	CONT_ ON	CYC_ ON	TUPDT_ ON	Output signal OUTV
Ramp/soak on	TRUE	FALSE	FALSE	**)	FALSE	**)	OUTV(t) End value is held after the end of processing.
Set output	TRUE	TRUE	**)	**)	**)	**)	DF_OUTV
Cyclic repetition on	TRUE	FALSE	FALSE	**)	TRUE	**)	OUTV(t) After end: Automatic start
Hold ramp/soak	TRUE	FALSE	TRUE	FALSE	**)	**)	Current value of OUTV(t) is held *)
Continue ramp/soak	TRUE	FALSE	TRUE	TRUE	**)	**)	OUTV (old) *)
			FALSE				Operation is continued with newly defined values.
Updating the total time and remaining total time	**)	**)	**)	**)	**)	FALSE	Does not influence OUTV
						TRUE	Does not influence OUTV

*) The ramp/soak profile does not have the form configured by the user until the next time slice.

**) The respective operating mode set is executed irrespective of the significance of the control signals in the fields marked with **)

Set output of ramp/soak

If you enable this function (DFRMP_ON = TRUE), the output value of the ramp/soak is set to signal value SP_INT or output value SP_GEN.

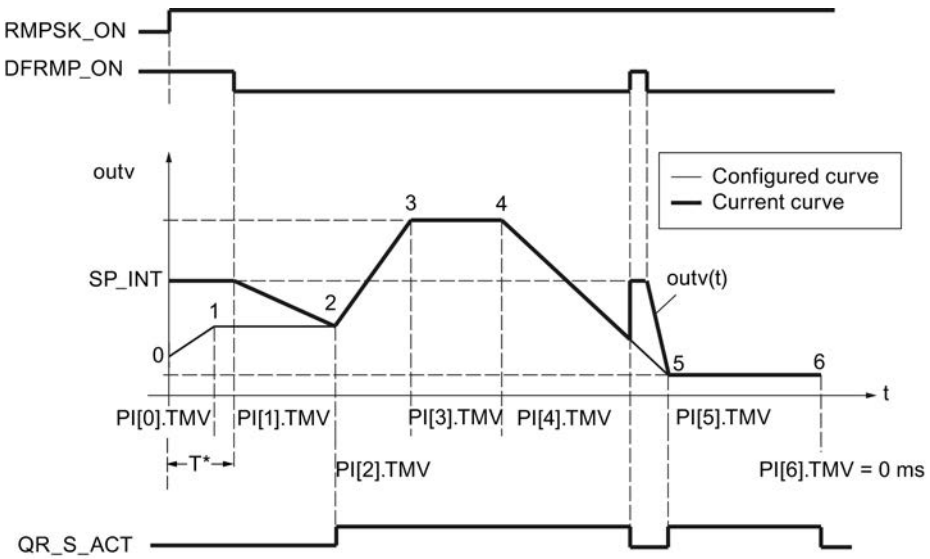
Note

Preassignment with the internal setpoint has an effect only if the ramp/soak function is enabled (RMPSK_ON = TRUE).

After the switchover to DFRMP_ON = FALSE, outv moves linearly from the specified setpoint (e.g., SP_INT) to the output value of the current time slice number PI[NBR_ATMS].OUTV.

The internal time processing continues running even when a fixed setpoint (RMPSK_ON = TRUE and DFRMP_ON = TRUE) is switched through.

On the start of the ramp/soak profile (RMPSK_ON = TRUE), the fixed setpoint SP_INT continues to be output until DFRMP_ON changes from TRUE to FALSE after the duration of T*.



After this moment, the time PI[0].TMV and a part of the time PI[1].TMV has expired. OUTV moves from SP_INT to PI[2].OUTV, i.e., from the setpoint to time slice 2.

The configured ramp/soak profile is output only starting from time slice 2. The conformity of the configured ramp/soak profile to the current ramp/soak profile is displayed by setting QR_S_ACT = TRUE. On a rising edge at DFRMP_ON during processing of the ramp/soak profile, the output value OUTV jumps to SP_INT without a delay.

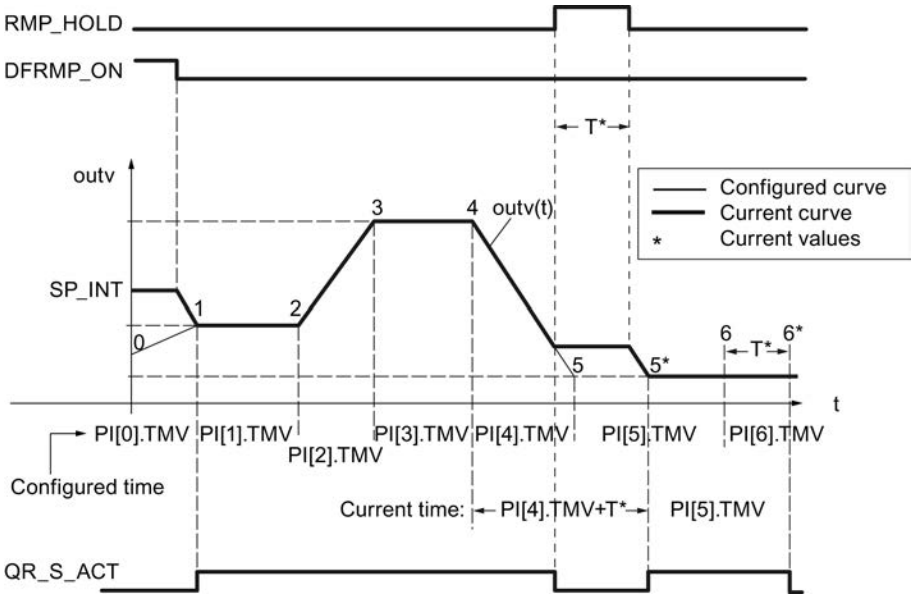
Cyclic repetition on

If "Cyclic repetition" operating mode (CYC_ON = TRUE) is activated, the ramp/soak function automatically returns to the starting point after the last time slice value has been output and begins a new run.

There is no interpolation between the last time slice and the starting point. For a smooth transition the following must be true: PI[NBR_PTS].OUTV = PI[0].OUTV.

Hold ramp/soak

When RMP_HOLD = TRUE, the value of the output variable (including the time processing) is frozen. When RMP_HOLD = FALSE is reset, the function resumes at the interruption point PI[x].TMV.



The processing time of the ramp/soak profile is extended by the amount of the time T*. The ramp/soak profile has the configured course from the starting point until the rising edge at RMP_HOLD and from time slice 5* to time slice 6*, i.e., output signal QR_S_ACT has the value TRUE.

If switch CONT_ON is set, the held ramp/soak function continues processing the ramp/soak profile at preassigned point TM_CONT.

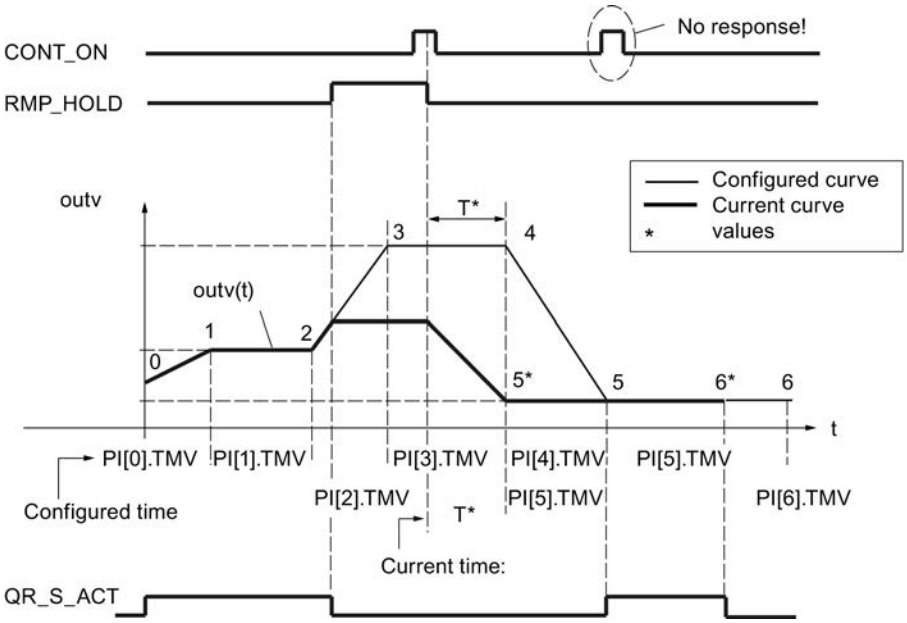
Continue ramp/soak

With CONT_ON, a held ramp/soak profile is restarted at a defined point. If the ramp/soak function is not held, CONT_ON has no effect.

If the control input for continuing (CONT_ON = TRUE) is set, the held ramp/soak function is continued at time slice TM_SNBR. TM_CONT determines the remaining time that the ramp/soak function requires to reach time slice TM_SNBR.

The following figure shows the effect of RMP_HOLD and CONT_ON when the configuration is as follows:

Continue at time slice 5 (TM_SNBR = 5) and continue in T* s: TM_CONT = T*



Time slices 3 and 4 are omitted in this processing cycle of the ramp/soak function. On a signal change of RMP_HOLD from TRUE to FALSE, the configured curve progression is only reached again starting from time slice 5.

Output QR_S_ACT is only set when the ramp/soak function executes the ramp/soak profile assigned by the user.

Updating the total time and remaining total time

The current time slice number NBR_ATMS, the current remaining time until the time slice is reached (RS_TM), the total time T_TM, and the total remaining time until the end of the ramp/soak profile (RT_TM) are updated in every cycle.

In the case of online changes of PI[n].TMV, the total time and the total remaining time of the ramp/soak profile change. Since the calculation of T_TM and RT_TM with many time slices significantly increases the processing time of the function block, the calculation is only carried out after a restart or at TUPDT_ON = TRUE. The time intervals PI[0 ... NBR_PTS].TMV between the individual time slices are added up and indicated at the total time T_TM and total remaining time RT_TM outputs.

Please note that relatively high CPU runtimes are required to determine the total times.

Procedure

To configure the ramp/soak function, proceed as follows:

1. Select "Ramp/soak" from the "Setpoint source" drop-down list.
2. Create a global data block RMP_SOAK.
3. Define the required parameters for the ramp/soak.
4. Enter the number of the data block RMP_SOAK in the "DB number" input field.

5. Enable or disable the additional functions of the ramp/soak.
6. If you have enabled the Continue ramp/soak function, enter at which time slice and when the ramp/sock is to be continued.

See also

Data block RMP_SOAK (Page 6935)

Scaling

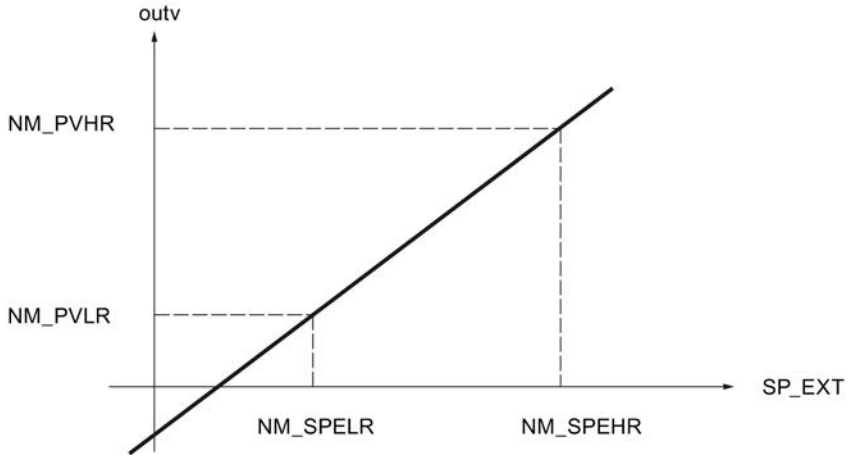
Scaling of an external setpoint

If the external setpoint is not expressed in the physical unit of the process value (e.g., expressed as a percent in the case of cascade control), then this value and its setting range must be scaled to the physical unit of the process value. The "Scaling in setpoint branch" function is used for this.

The SP_NORM function scales an analog input variable. The analog external setpoint is transformed to output variable outv using the scaling level. Output value OUTV is accessible in the configuration tool via measuring point MP2.

The output value of the function has an effect when control input SPEXT_ON = TRUE. In order to uniquely specify the scaling levels, you define the following four parameters:

- High input: SP_EXT: NM_SPEHR
- Low input: SP_EXT: NM_SPELR
- High output: NM_PVHR (You specify this value in the scaling function of the process value.)
- Low output: NM_PVLR (You specify this value in the scaling function of the process value.)



Output value outv is calculated from the respective input value SP_EXT according to the following formula:

outv = (SP_EXT – NM_SPELR) x (NM_PVHR – NM_PVLR) / (NM_SPEHR – NM_SPELR) + NM_PVLR

For the special case in which a square root function is activated in the process value branch, the scaling values of the square root function (SQRT_HR and SQRT_LR) apply as the high and low limits of output value outv.

In this case, output value outv of the scaling function is calculated from the respective input value SP_EXT according to the following formula:

outv = (SP_EXT – NM_SPELR) x (SQRT_HR – SQRT_LR) / (NM_SPEHR – NM_SPELR) + SQRT_LR

There is no internal value limiting in the function. The parameters are not checked. If you enter the same value for NM_SPEHR and NM_SPELR, a division by zero is possible based on the formulas above. The instruction will not detect this.

Combination/ratio factor

With ratio control, the setpoints of the control loop are coordinated with one another in such a way that the set ratio matches the actual ratio of the process values.

With blending control, the desired quantities of the individual components are set. The total of the blending factors FAC must be 1.

Procedure

To configure the scaling, proceed as follows:

1. Enter the values for the scaling in the input fields "High input" and "Low input".
2. Enter a blending/ratio factor (FAC).

Limiting functions

FC call in the setpoint branch

Insertion of a user-specific function in the setpoint branch allows an externally specified reference variable to undergo signal processing (e.g., signal delay or linearization) prior to application in the controller.

If you activate the FC call in the setpoint branch, a user-specific function is called. The controller calls the FC. In so doing, existing input/output parameters of the user FC are not initialized. Therefore, you must use S7-STL to program the data transfer in the user FC. You can monitor the prepared setpoint at MP3.

The value of SPFC_ON then determines whether a user-programmed function in the form of a standard FC (e.g., a characteristic curve) will be inserted at this point in the setpoint channel or whether the setpoint will continue to be processed without any influence of this type.

Note

The statement does not check to determine if a function exists. If the function does not exist, the CPU switches to STOP with an internal system error.

Procedure

To call a function in the setpoint branch, proceed as follows:

1. Select the "Enable FC call" check box.
2. In the "FC" input field, enter the number of the function to be called.

Limiting of the rate of change of the setpoint

Ramp functions are used in the setpoint branch in situations where the process does not tolerate abrupt changes in the actuating signal, since an abrupt setpoint change generally also causes an abrupt change in the manipulated variable of the controller. These abrupt manipulated variable changes must be avoided, for example, when a gearbox is connected in between an adjustable-speed motor and the load to be driven and an excessively rapid increase of the motor speed would result in overloading of the gearbox.

The SP_ROC function limits the rate of change of the setpoints processed in the controller. This is carried out separately for the rising slope and falling slope and separately for the positive and negative range of the reference variable.

The limiting for each slope of the ramp function in the positive range and in the negative range of the reference variable is entered in the four inputs SPURLM_P, SPDRLM_P, SPURLM_N, and SPDRLM_N. The slopes refer to a rising or falling slope per second. Faster setpoint changes are ramped down to these maximum rates.

For example, if SPURLM_P is assigned the value 10.0 [engineering value range/s], the following values are added to the "old value" of outv in each sampling cycle for as long as inv > outv:

Sampling time	1 s	$outv_{old}$ + 10
	100 ms	$outv_{old}$ + 1
	10 ms	$outv_{old}$ + 0.1

The following figure shows an example of how the signal processing functions. The step functions at input inv(t) become ramp functions at output outv(t).

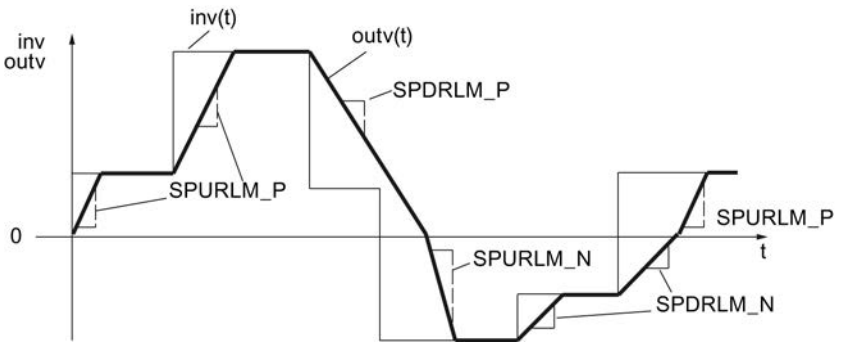


Figure 11-7 Limiting of the rate of change of the reference variable SP(t)

When slope limits are reached, this is not signaled.

Procedure

To limit the rate of change, proceed as follows:

1. Activate the "Limit rate of change" check box.
2. Specify the following values for the positive range as well as for the negative range:
   - Rising slope for setpoint value
   - Falling slope for setpoint value

Limiting of the absolute value of the reference variable

The setting range of the reference variable determines the range in which the controlled variable can fluctuate, i.e., the range in which the process varies within the scope of permissible status values.

To avoid a critical or impermissible process status, the setting range of the reference variable is subject to high and low limits (SP_LIMIT) in the setpoint branch of standard PID control.

The SP_LIMIT function limits the setpoint SP at the assignable low and high values (SP_LLM and SP_HLM) as long as input variable INV is outside these limits. Because the function cannot be deactivated, the specification of a low limit and high limit must be taken into consideration in the configuration.

The numeric values of the limits are set in the input parameters for the low and high limits. In the event of violations by input variable inv(t), the associated displays are output via the message outputs.

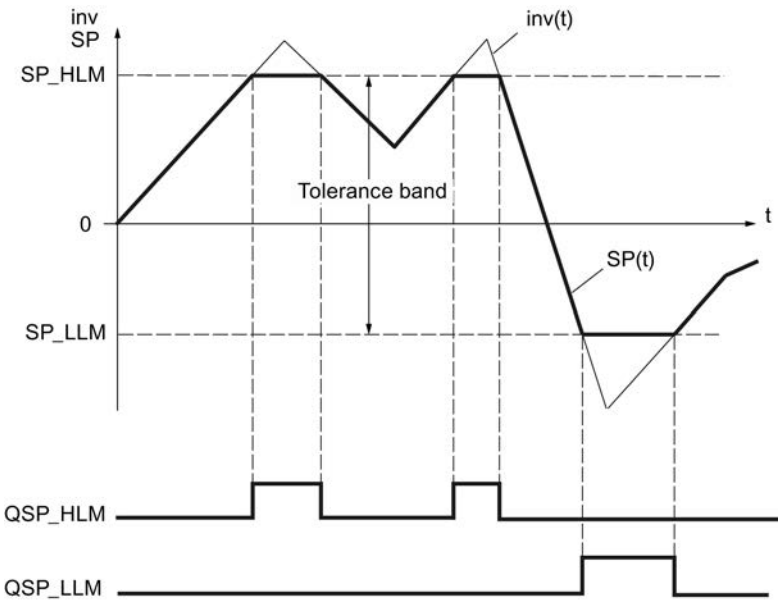


Figure 11-8 Absolute value limiting of reference variable SP(t)

During a restart, the message bits are reset.

The message bits are set as follows:

SP =PVH_ALM	QSP_HLM	QSP_LLM	If:
SP_HLM	TRUE	FALSE	inv ≥ SP_HLM
SP_LLM	FALSE	TRUE	inv ≤ SP_LLM
inv	FALSE	FALSE	SP_HLM < inv < SP_LLM

Procedure

To limit the absolute value of the reference variable, proceed as follows:

1. Enter the high limit for the absolute setpoint (SP_HLM) in the "High limit" input field.
2. Enter the low limit for the absolute setpoint (SP_LLM) in the "Low limit" input field.

Control deviation

Deadband

In an optimally tuned controller, when a high-frequency interfering signal is superimposed on the control variable or reference variable, the noise portion will also have an effect on the controller output. This can result in strong fluctuations of the manipulated variable, for example, if the controller gain is high and derivative action is enabled. Due to the increased switching frequency (step controller), this causes increased wear of the actuator.

The function reduces the noise portion in the control deviation signal while the controller is in steady-state and thus reduces undesired oscillating of the controller output.

The deadband can be disabled.

Signal filtering through deadband function

The DEADBAND function suppresses small fluctuations of the input variable around a specified zero point within an adjustable range. Outside this fluctuation range, the control deviation ER rises or falls proportional to the input variable. The width of the deadband can be specified with the DEADB_W parameter. Only positive values are permitted for the deadband width.

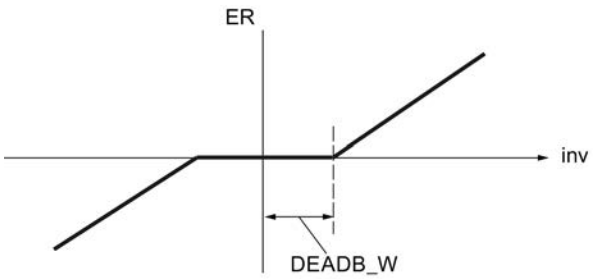
If the input variable is within the deadband, the value 0 (control deviation = 0) is output at the output. Only when the input variable exits the deadband will the output rise or fall by the same values as input variable inv. This results in corruption of the transmitted signal outside the deadband, too. However, this is accepted in order to avoid step changes at the deadband limits (see figure below). The corruption corresponds to the value DEADB_W and can therefore be controlled easily.

The DEADBAND function operates according to the following functions:

(ER) = inv + DEADB_W for inv < - DEADB_W

(ER) = 0 for -DEADB_W ≤ inv ≤ + DEADB_W

(ER) = inv - DEADB_W for inv > + DEADB_W



Effects of the signal filtering can be monitored at the output parameter ER.

Alarm

Excessive deviations of the controlled variable from the specified setpoint can cause an undesired process status. In such cases, the ER_ALARM function is used to monitor the control deviation for violation of the permissible operating range. ER_ALARM detects and signals any limit violations that occur so that an appropriate reaction can be initiated.

Note

The monitoring of the control deviation for limit values cannot be disabled. The violation of the configured limits is always signaled.

The ER_ALARM function monitors the size of control deviation ER(t) for four assignable limits in two tolerance bands. If the limits are reached or exceeded, the function signals a “warning” at the first limit and an “alarm” at the second limit.

Limit values

The numeric values of the limits are set in the input parameters for “warning” and “alarm”. If the control deviation (ER) violates these limits, the associated output message bits QERN_ALM ... QERP_ALM are set.

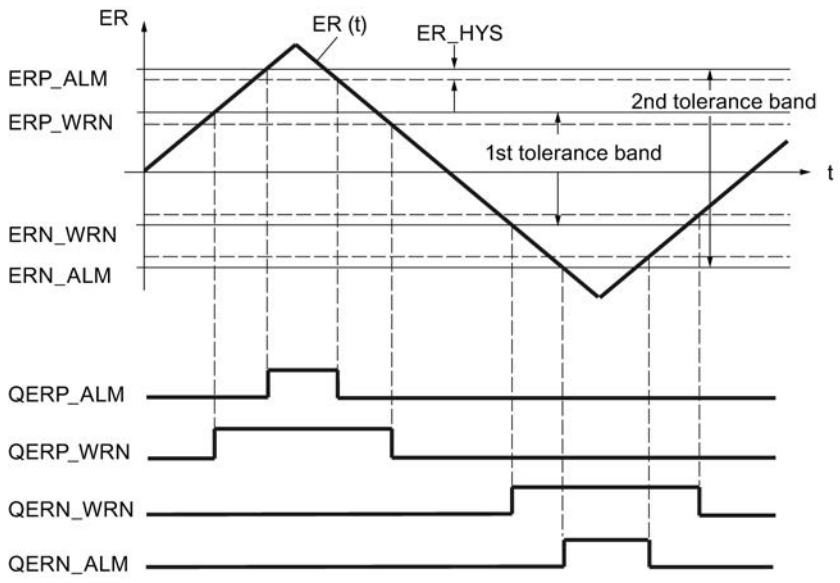
During a restart, the message bits are reset.

The message bits are set as follows:

QERP_ALM	QERP_WRN	QERN_WRN	QERN_ALM	If:	And:
TRUE	TRUE	FALSE	FALSE	ER ↗ ER ↘	ER ≥ ERP_ALM ER ≥ ERP_ALM - ER_HYS
FALSE	TRUE	FALSE	FALSE	ER ↗ ER ↘	ER ≥ ERP_WRN ER ≥ ERP_WRN - ER_HYS
FALSE	FALSE	TRUE	FALSE	ER ↘ ER ↗	ER ≤ ERP_WRN ER ≤ ERP_WRN + ER_HYS
FALSE	FALSE	TRUE	TRUE	ER ↘ ER ↗	ER ≤ ERP_ALM ER ≤ ERP_ALM + ER_HYS

The following has to apply so that the block operates properly:

High alarm < High warning < Low warning < Low alarm



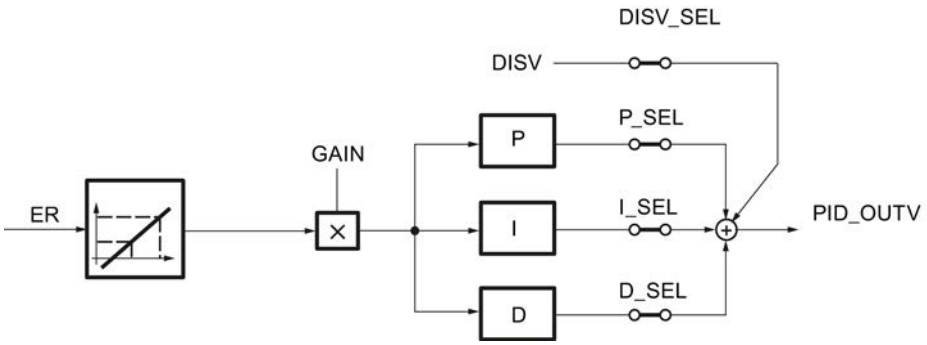
Hysteresis

In order to avoid "flickering" of the message bits when the input variable undergoes slight changes or in the event of rounding errors, a hysteresis ER_HYS is used. The control deviation must violate the hysteresis before the message bits are reset.

Controller

Control algorithm

The manipulated variable of the continuous controller is calculated within the configured sampling time cycle on the basis of the control deviation in the PID position algorithm. The controller is implemented in a purely parallel structure. The proportional, integral, and derivative actions can each be disabled individually.

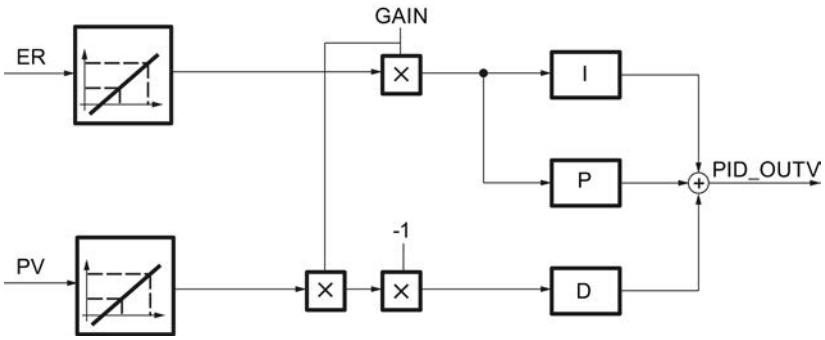


Feedforward control

A disturbance variable DISV can be added in the output signal of the controller (PID_OUTV). The variable is switched-in or switched-off in the PID window of the configuration tool using the DISV_SEL structure selector or through "Disturbance variable on".

Proportional and derivative action in the feedback path

In the parallel structure, each action in the control algorithm receives the control deviation as input signal. Via the proportional action and derivative action, the manipulated variable is influenced directly by setpoint steps. However, a different controller structure in which the formation of the proportional and the derivative actions is moved to the feedback path ensures a smooth response of the manipulated variable to setpoint step changes. In this structure, the integral action processes the control deviation as the input signal. Only the negative controlled variable (factor = - -1) is applied to the proportional action and the derivative action.



Scaling the input variable ER and PV

The input variables ER and PV of the PID controller are scaled to the range 0 to 100 prior to the controller processing according to the following formulas:

- with square-root function disabled (SQRT_ON = FALSE):
  - ER_{scaled} = ER * 100.0 / (NM_PVHR – NM_PVLR)
  - PV_{scaled} = (PV - NM_PVLR) * 100.0 / (NM_PVHR – NM_PVLR)
- with square-root function enabled (SQRT_ON = TRUE):
  - ER_{scaled} = ER * 100.0 / (SQRT_HR – SQRT_LR)
  - PV_{scaled} = (PV - SQRT_LR) * 100.0 / (SQRT_HR – SQRT_LR)

This scaling is performed so that the proportional gain GAIN of the PID controller can be entered without dimensions. If the high limit and low limit of the physical measuring range change (for example, from bar to mbar), the gain factor does not then have to be changed.

The scaled input variables ER_{scaled} and PV_{scaled} cannot be monitored.

See also

Controller structures (Page 9727)

Proportional component (Page 9730)

Derivative component (Page 9734)

Integral action (Page 9731)

Controller structures

Use and parameter assignment of the PID controller

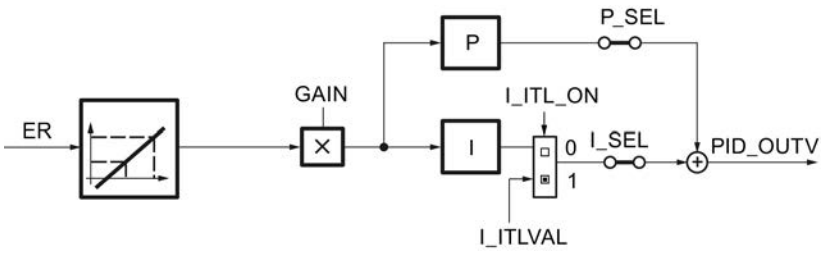
The majority of the controlled systems occurring in the process industry can be controlled with the PI controller function/PID controller function of standard PID control. Only in special cases are additional methods and measures needed for a particular closed-loop control.

However, a major problem in the field continues to be how to assign the PI controller/PID controller parameters, i.e., how to determine the "correct" settings for the controller parameters. The quality of this parameter assignment is critical in order for the PID control to meet the requirements of the task at hand. It requires either a lot of practical experience, special expertise, or a great amount of time.

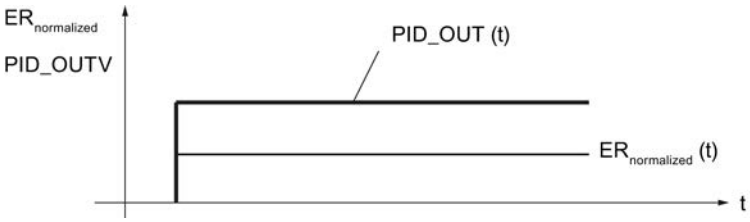
P control

The integral and derivative actions are disabled in a proportional-action controller (P-controller). (I_SEL and D_SEL = FALSE). This means that when control deviation ER = 0, output signal OUTV = 0. If an operating point ≠ 0 is intended, i.e., a numeric value will be set for the output signal when the control deviation is zero, the integral branch must be activated (see the following figure).

In the integral action, an operating point ≠ 0 can be specified for the P-controller via a corresponding setting of initialization value I_ITLVAL. Set switches 'I_ITL_ON' and 'I_SEL' = TRUE for this purpose.



The step response of the P-controller within the time range is:
PID_OUTV (t) = I_ITLVAL + GAIN * ER_{scaled} (t)



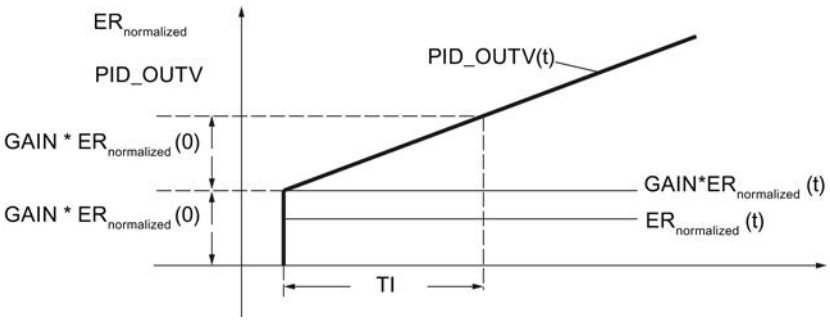
PID_OUTV(t)	Manipulated variable when controller is in automatic mode
I_ITLVAL	Operating point of the P-controller
GAIN	Controller gain
$ER_{scaled}(t)$	Scaled control deviation

PI control

The derivative action is disabled in the proportional-integral controller (PI controller). (D_SEL = FALSE). A PI controller adjusts output variable PID_OUTV using the integral action until control deviation ER = 0. However, this only applies if the output variable does not exceed the limits of the control range.

The step response within the time range is:

$$PID_OUTV(t) = GAIN * ER_{normalized}(0)\left(1 + \frac{1}{TI} * t\right)$$



PID_OUTV(t)	Manipulated variable when controller is in automatic mode
GAIN	Controller gain
$ER_{scaled}(0)$	Step height of the scaled control deviation
TI	Integral-action time constant

PD control

The integral action is disabled in the proportional-derivative controller (PD-controller) (I–SEL = FALSE). This means that when control deviation ER = 0, output signal OUTV = 0. If an operating point ≠ 0 is intended, i.e., a numeric value will be set for the output signal when the control deviation is zero, the integral action branch must be activated (see also the block diagram of the P control).

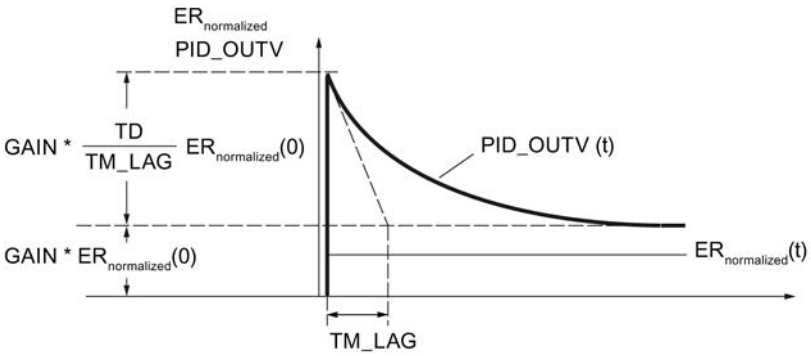
In the integral action, an operating point ≠ 0 can be specified for the PD-controller via a corresponding setting of initialization value I_ITLVAL. The switches 'I_ITL_ON' and 'I_SEL' must be set to TRUE for this purpose.

The PD-controller maps input variable ER(t) proportionally to the output signal and adds to this the derivative action generated by differentiation of ER(t); the derivative action is calculated with double precision according to the trapezoid rule (Padé approximation). The time response is determined by the derivative time constant TD.

For signal smoothing and suppression of interfering signals, a first-order delay (adjustable time: TM_LAG) is integrated into the algorithm for generating the derivative action. A small value for TM_LAG is usually sufficient to achieve the desired result. If TM_LAG ≤ CYCLE/2 is configured, the time delay is deactivated.

The step response within the time range is:

$$PID_OUTV(t) = GAIN * ER_{normalized}(0)\left(1 + \frac{TD}{TM_LAG} * e^{-\frac{t}{TM_LAG}}\right)$$



PID_OUTV(t)	Manipulated variable when controller is in automatic mode
GAIN	Controller gain
$ER_{scaled}(0)$	Step height of the scaled control deviation
TD	Derivative time
TM_LAG	Time delay

PID control

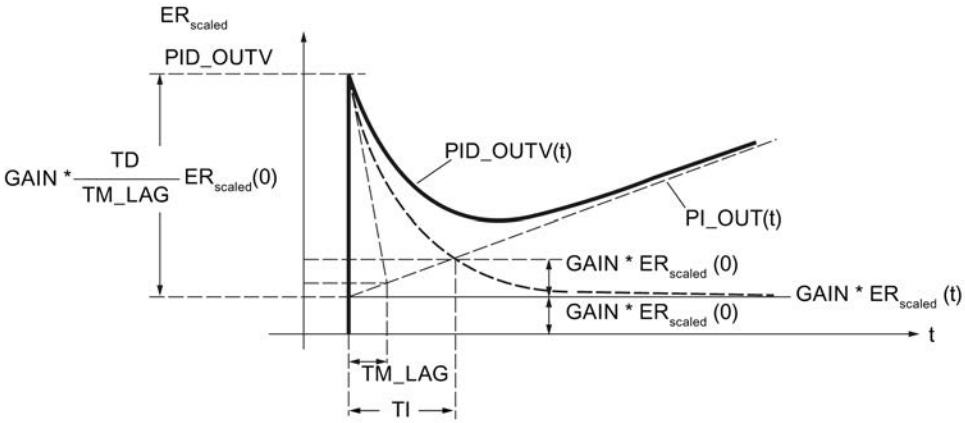
In the proportional-integral-derivative controller (PID-controller), the proportional action, integral action, and derivative action are enabled (P–SEL = TRUE, I–SEL = TRUE, D_SEL = TRUE). A PID controller adjusts output variable PID_OUTV using the integral action until control deviation ER = 0. However, this only applies if the output variable does not exceed the limits of the control range. If manipulated variable limits are exceeded, the integral action maintains the value that was reached at the limit (anti-reset windup).

The PID-controller maps scaled input variable $ER_{scaled}(t)$ proportionally to the output signal and adds to this the components generated by differentiation and integration of $ER_{scaled}(t)$; these components are calculated with double precision according to the trapezoid rule (Padé approximation). The time response is determined by the derivative-action time TD and integral-action time TI.

For signal smoothing and suppression of interfering signals, a first-order delay (adjustable time constant: TM_LAG) is integrated into the algorithm for generating the derivative action. A small value for TM_LAG is usually sufficient to achieve the desired result. If TM_LAG ≤ CYCLE/2 is configured, the time delay is deactivated.

The step response within the time range is:

$$PID_OUTV(t) = GAIN * ER_{normalized}(0) \left(1 + \frac{1}{TI} * t + \frac{TD}{TM_LAG} * e^{-\frac{t}{TM_LAG}}\right)$$



PID_OUTV(t)	Manipulated variable when controller is in automatic mode
GAIN	Controller gain
$ER_{scaled}(0)$	Step height of the scaled control deviation
TI	Integral-action time constant
TD	Derivative time
TM_LAG	Time delay

Note

If TD is changed, you should also adjust TM_LAG accordingly.

Recommendation: 5 ≤ (TM / TM_LAG) ≤ 10

See also

Control algorithm (Page 9725)

Proportional component (Page 9730)

Derivative component (Page 9734)

Proportional component

Proportional gain

The sign of this proportional gain determines the effective direction of the continuous controller.

- Positive proportional gain
  increasing control deviation → rising manipulated variable
- Negative proportional gain
  increasing control deviation → falling manipulated variable

Applying proportional action in feedback path

If the proportional action is applied to the feedback path, only the process value has an effect on the proportional action. Therefore, the manipulated variable does not undergo a step change due to the proportional action during a changeover from manual to automatic (also in the case of a setpoint step-change); the changeover is bumpless.

See also

Control algorithm (Page 9725)

Integral action

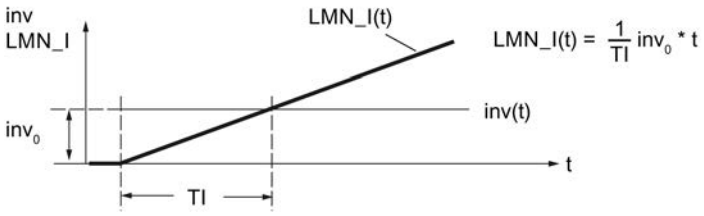
Function of the integral action

The integral action ensures that the control deviation can go to 0 for any manipulated variables by resetting the operating point.

The integral action generates an output signal whose rate of change is proportional to the change in the absolute value of the input variable. The time response is determined by the integral action time TI.

The response characteristic in the time range is:

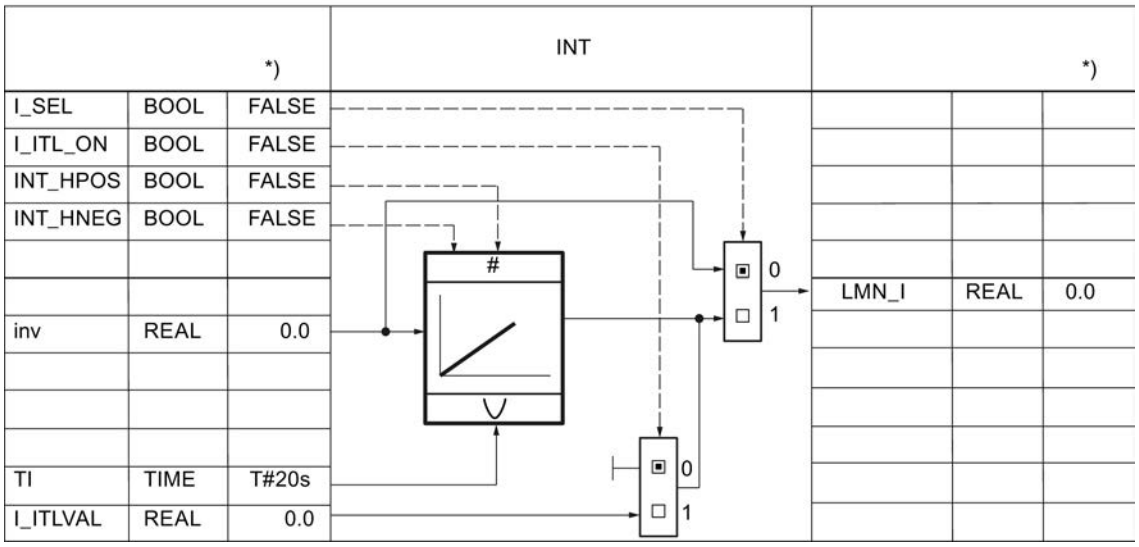
$$OUTV(t) = \frac{1}{TI} \int inv(t)\, dt$$



LMN_I(t)	The output variable of the integral action
inv_0	The step height at the input of the integral action
TI	Integral action time

The integral action can be monitored at the static tag LMN_I.

Parameter	Meaning	Permitted value range
TI	Integral action time	≥ 5 * CYCLE
I_ITLVAL	Initialization value for I action	-100.0 to +100.0 [%]

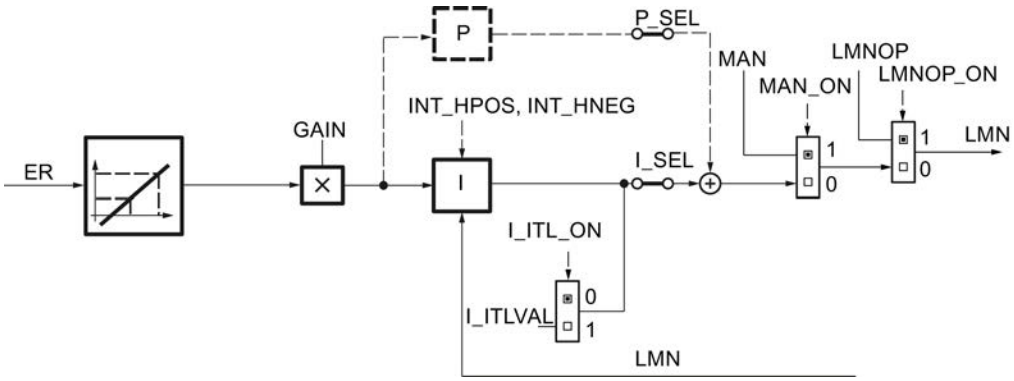


*) Default for creation of new technology object

In the deactivated state (I_SEL = FALSE), the integral action, i.e., the internal memory and the static tag LMN_I, is set to zero.

The output and the memory of the integral action are limited by the absolute limits of the manipulated variables LMN_HLM and LMN_LLM (Anti Reset Wind–up).

If the actuating signal is influenced in manual mode, i.e., if MAN_ON, LMNOP_ON or CAS_ON = TRUE, the internal memory value of the integral action is corrected to the value LMN.



Changeover from manual to automatic mode

The integral action is set in manual mode in such a manner that the manual-to-automatic transition causes the manipulated variable to move from the current position. The height of the step change is determined by the proportional action. The step height of the proportional action corresponds to the manipulated variable change on a setpoint step change from the current process value to the current setpoint.

However, if the proportional action is applied to the feedback path, only the process value has an effect on the proportional action. Therefore, the manipulated variable does not undergo a step change due to the proportional action during a switchover from manual to automatic; the changeover is smooth.

Permissible ranges for the integral action time TI and CYCLE

The limited accuracy of the REAL numbers calculated in the CPU means that the following can occur during integration: If the selected sampling time CYCLE of the control is too short in comparison with the integral action time TI and the input value inv of the integral action is small compared to its output value OUTV, the integral action does not respond and remains at its current output value.

This effect can be avoided if the following sizing rule is observed during configuration:

$CYCLE > 10^{-4} \times TI$

As a result, the integral action still responds to changes in the input values in the range of millionths of one part per thousand of the current output variable:

$inv > 10^{-10} \times OUTV$

In order for the response characteristic of the integrator algorithm to conform to the analog response, the sampling time should be less than 20% of the specified integral action time or TI should be at least five times the value of the specified sampling time:

$CYCLE < 0.2 \times TI$

The algorithm permits sampling time values up to $CYCLE \leq 0.5 * TI$.

Initializing the integral action

If initialization of the integral action is enabled, the initialization value is switched through to the static tag LMN_I at a restart of the CPU or COM_RST = TRUE. On the transition to normal operation, the integral action starts the integration of its input variable from the initialization value due to the resetting of the initialization.

Hold integral action

The integral action can be blocked in positive or negative direction by the static tags INT_HPOS and INT_HNEG. This can be useful for controller cascades. If, for example, the manipulated variable of the follow-up controller approaches the high limit, a further increase in the manipulated variable of the master controller can be prevented by its integral action. You program this response using the following instructions, for example, in STL:

```
A    "Follow-up controller".QLMN_HLM
=    "Master controller".INT_HPOS
A    "Follow-up controller".QLMN_ LLM
=    "Master controller".INT_HNEG
```

See also

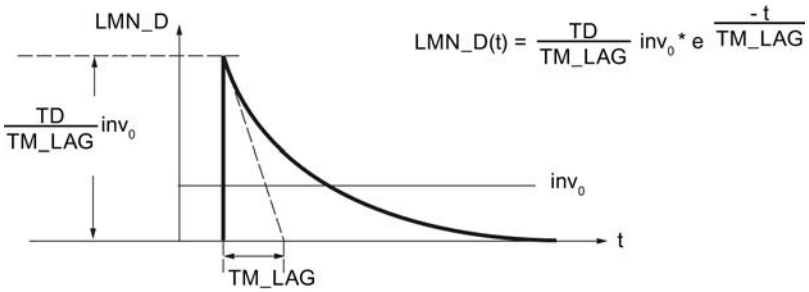
Control algorithm (Page 9725)

Derivative component

Function of the derivative action

The derivative action generates an output signal whose size changes proportionally to the rate of change of the input variable. The time response is determined by the derivative time TD and the time delay TM_LAG.

The step response to the input step inv_0 is:



LMN_D(t)	The output variable of the derivative action
inv_0	The step height at the input of the derivative action
TD	Derivative-action time
TM_LAG	Time delay

The derivative action can be monitored at the static tag LMN_D.

In the disabled state (D_SEL = FALSE), the derivative action, i.e., the internal memory and the static tag LMN_D, is set to zero.

Apply derivative action in feedback path

If the derivative action is applied to the feedback path, only the process value has an effect on the derivative action. Consequently – as in a setpoint jump – the manipulated variable does not undergo a step change due to the derivative action during a changeover from manual to automatic; the switchover is smooth.

Permitted values for derivative time and time delay.

To calculate the derivative action correctly, the following conditions must be satisfied for the derivative time and time delay:

- TD ≥ CYCLE and
- TM_LAG ≥ 0.5 × CYCLE

If TD < CYCLE, the derivative action operates as though the TD has the value CYCLE.

If TM_LAG < 0.5 × CYCLE, the derivative action operates without delay. An input step is then multiplied with the factor TD/CYCLE and this value is applied as "needle pulse" at the output. In other words, LMN_D is reset to zero in the subsequent processing cycle.

Coefficient DT1

The coefficient DT1 is calculated as follows:

Coefficient DT1 = TD/TM_LAG

Time delay

For signal smoothing and suppression of interfering signals, a first-order delay (adjustable time constant: TM_LAG) is integrated into the algorithm for generating the derivative action.

A small value for the time delay is usually sufficient to achieve the desired result. If the assigned time delay is less than half the controller sampling time, the delay is deactivated.

Procedure

To configure the derivative action, proceed as follows:

1. To apply the derivative action to the feedback path, select the corresponding check box.
2. Enter the derivative time in seconds.
3. Enter the time delay in seconds.

See also

Control algorithm (Page 9725)

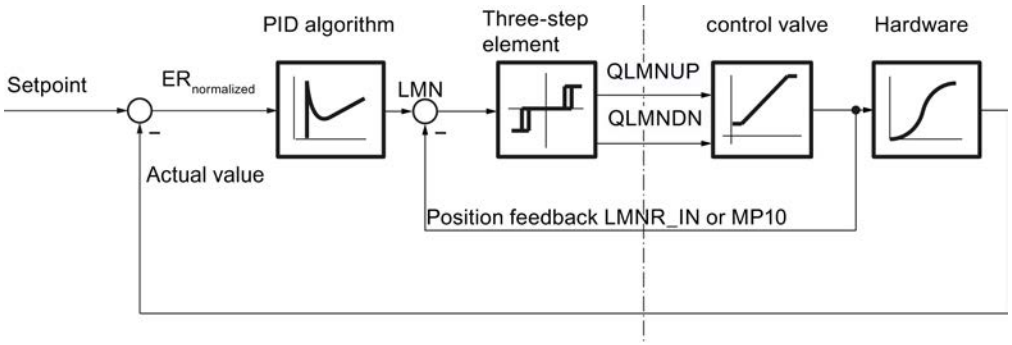
Manipulated variable

Step controller with position feedback

Function chart of the step controller with position feedback in the control loop

The functional principles of the step controller with position feedback are based on the PID control algorithm and are supplemented by the function elements for generating the binary output signal from the analog actuating signal of the controller.

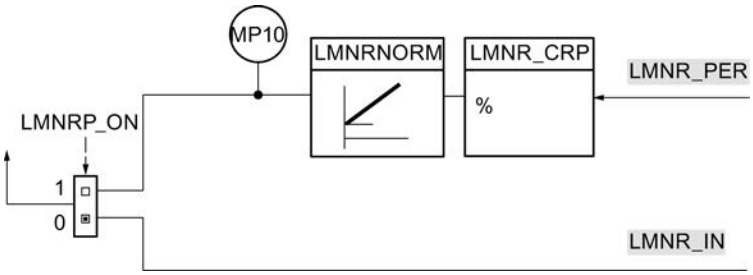
Depending on the sign, the three-step element hereby converts deviations between manipulated variable and position feedback into positive or negative pulses of the output signal in order, for example, to apply these to a motor-operated valve drive. Basically, this amounts to a cascade control with a lower-level position control loop.



Selecting position feedback

For applying the position feedback to the comparator in the manipulated variable branch of the step controller, inputs with appropriate signal processing are available, depending on the format of the value to be processed. LMNR_PER can be used to apply signals in the SIMATIC I/O format, and LMNR_IN can be used to apply signals in floating-point format.

The associated internal value in the percent range is stored at measuring point MP10.



If the position feedback is provided by an analog input module, the numerical value of the position feedback at LMNR_PER must be converted to a floating-point value scaled to a percentage.

In this process, no check is carried out for positive/negative overflow, reaching of the overrange/underrange, or wire break.

The following table provides an overview of ranges and numerical values before and after processing.

I/O value LMNR_PER	Output value as a %
32767	118.515
27648	100.000
1	0.003617
0	0.000
- 1	- 0.003617
- 27648	- 100.000
- 32768	- 118.519

Procedure

Note

It is not permissible to enable or disable position feedback in online mode. In other words, LMNR_ON may not be changed online.

To activate the position feedback, proceed as follows:

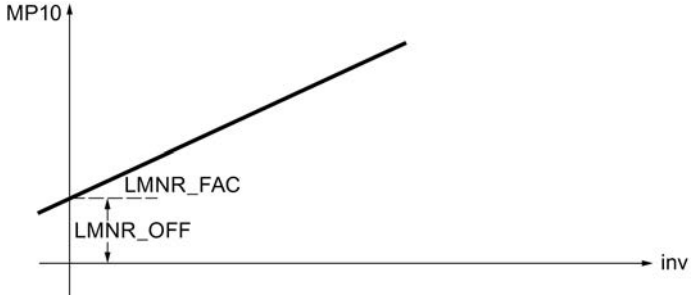
1. Select the "Enable" check box in the "Select position feedback" area.
2. Choose the "Position feedback signal IO" option in order to apply signals in the SIMATIC I/O format.
   Choose the "Position feedback" option in order to apply signals in floating-point format.

Scaling position feedback

If the position feedback is available as a physical variable (e.g. 240 to 800 mm or in the angle from 0° to 60°), the position feedback in floating-point format (%) must be scaled to the physical variable required for the further processing of the required internal floating-point value.

The following parameters must be defined for the unique specification of the scaling levels:

- The factor (for the slope): LMNR_FAC
- The offset of the scaling level at the zero point: LMNR_OFF



The scaled position feedback is calculated according to the following formula from inv (LMNR_PER) and output at MP10:

MP10 = inv * LMNR_FAC + LMNR_OFF

The scaling is effective if the I/O position feedback is enabled (LMNRP_ON = TRUE). No values are limited internally in the function. There is no parameter check.

Procedure

To scale the signal of the position feedback, proceed as follows:

1. Enter the slope of the scaling level in the "Factor" input field.
2. Enter the offset of the scaling level with respect to the zero point in the "Offset" input field.

Step controller without position feedback

Structure and function of the step controller

The step controller without position feedback consists of two parts: The PD controller part, which operates with continuous signals, and a second part in which the binary actuating signals are generated from the difference between the PD action and the feedback.

The integral action of the step controller without position feedback is calculated in the feedback path. The feedback signal compared with the controller output of the PD controller is generated in this process by integrating the following two signal components:

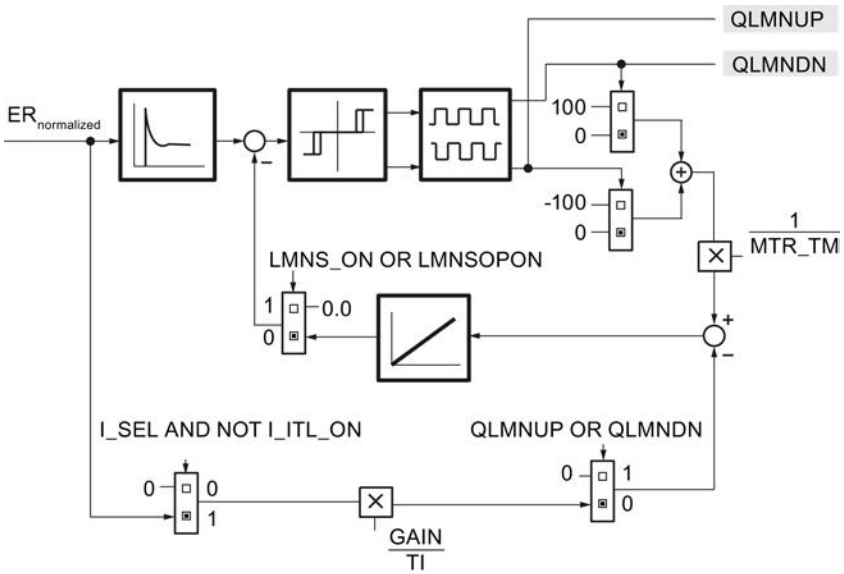
- Signal component for simulated position feedback
  ± 100/MTR_TM
- Signal component for the integral action
  ER_{scaled}*GAIN/TI

The feedback signal is thus the difference between simulated position feedback and integral action.

Depending on the sign, the three-step element converts deviations between manipulated variable and feedback signal into positive or negative pulses of the output in order, for example, to apply these to a motor-operated valve drive.

In steady state, the output of the integral action and the PD action go to zero. Because the input of the three-step element also goes to zero, the binary actuating signals QLMNUP and QLMNDN remain at FALSE.

The I action of the PID algorithm is deactivated. In the step controller without position feedback, it is not intended for the I action to be preassigned or frozen. In this case, there is no manual mode option using the MAN parameter because of the lack of information regarding the actuator position.



Addressing the endstop signals of the actuator (LMNR_HS/LMNR_LS) causes the manipulated variable LMNUP or LMNDN to be blocked.

If the actuator returns no endstop signals, the input parameters LMNR_HS and LMNR_LS = FALSE must be set.

Note

If end stop signals are not available, the controller cannot detect whether a mechanical end stop has been reached. For example, it is possible that the controller will output signals for opening the valve even though the valve is already at its upper end stop.

Manual mode

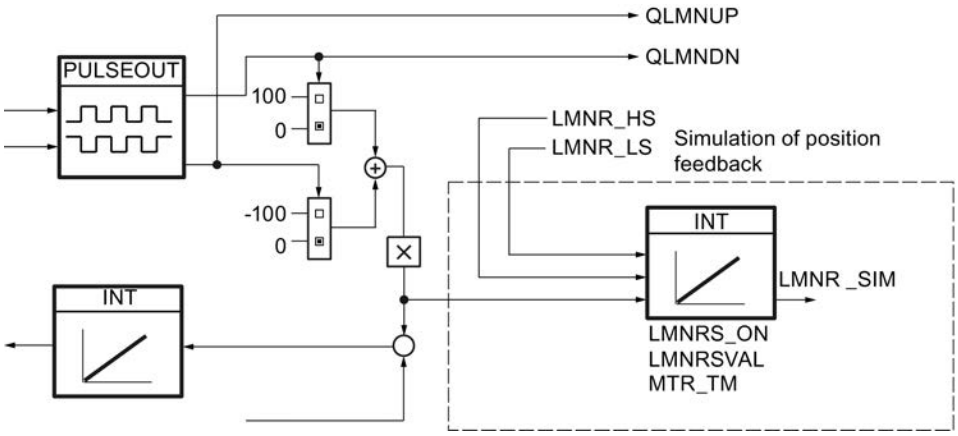
The step controller without position feedback has no manual mode via the parameter MAN or the manual output variable generator MAN_GEN because there is no information on the position of the actuator. Manual mode is only possibly by direct switching of the actuating signal (LMNS_ON = TRUE).

Simulation of position feedback

If a measurable value for the position feedback is not available, it can also be simulated (LMNRS_ON = TRUE).

The position feedback is simulated by an integrator with motor transition time MTR_TM as the integration time constant. When LMNRS_ON = FALSE, the start value of the LMNRSVAL parameter is output at the output of the LMNR_ SIM integrator. On switchover to LMNRS_ON = TRUE, the simulation starts up with this start value.

If LMNR_HS = TRUE, the integration is held up, and if LMNR_LS = TRUE, it is held down. The simulated position feedback is not adapted to the end positions.



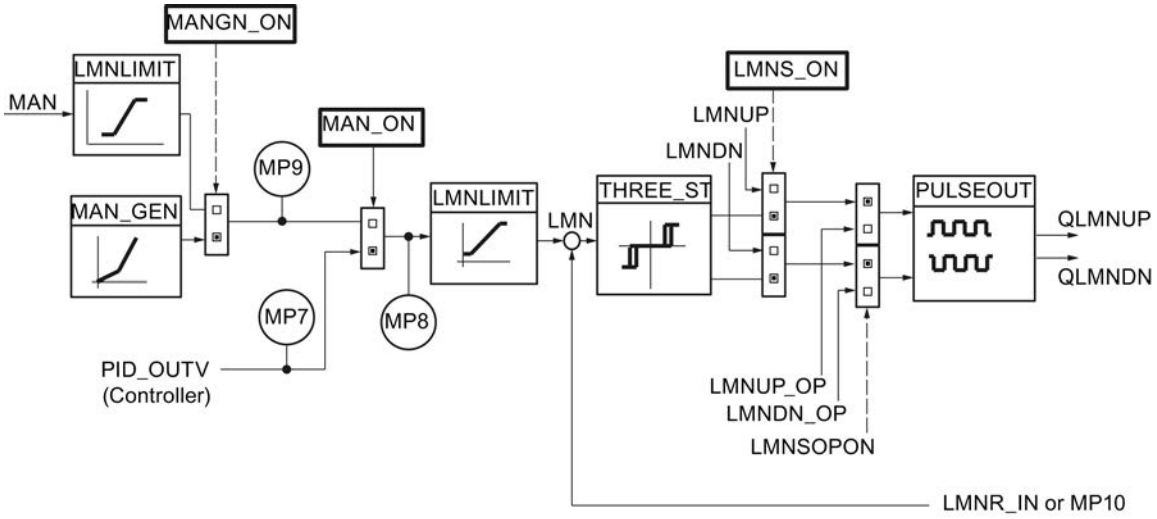
Note

The position feedback is merely simulated. It does not have to match the actual position of the actuator. If a real position feedback is available, it should always be used.

Manual mode

Manual mode

The structure of the manual value generation and manual value selection is shown in the next figure.



With step controllers, there are three operating modes in which the manipulated variable can be influenced:

- Manual value via the input parameter MAN
  The MAN parameter (-100.0% to 100.0%) allows you to activate influencing by an external manipulated variable in the form of an absolute value or to specify it in the user program. This is possible only when position feedback is enabled.
- Manual mode via more/less switch of the manual value generator (MANUP, MANDN).
  If MAN_GEN is activated from another operating mode, the manipulated variable MP9 is applied. The changeover to the manual value generator is thus always bumpless. The manual output variable can be increased or decreased within the LMN_HLM and LMN_LLM limits.
  This is possible only when position feedback is enabled.
- Manual mode by means of direct connection of the actuating signals
  Manual mode by direct connection of the actuating signals LMNUP and LMNDN always has priority.
  When LMNS_ON = FALSE is reset, the switch to any other activated operating mode will be "bumpless".

Automatic mode

If manual mode is disabled (MAN_ON = FALSE), the manipulated variable of the PID algorithm is connected to the three-step element. The changeover from manual mode to automatic mode causes a step change in the manipulated variable LMN. However, this is not a disadvantage because the integrating actuator means that the process is controlled according to a ramp function. The output of the PID algorithm is stored in measuring point MP7.

Manual value follow-up in automatic mode

If the I/O position feedback is enabled (LMNRP_ON = TRUE), the input parameter MAN of the position of the actuator (MP10) is corrected.

If position feedback is enabled (LMNRP_ON = FALSE), the input parameter MAN of the position of the actuator (LMNR_IN) is corrected.

Therefore, on switchover to manual mode, the manipulated variable remains at the value that corresponds to the actuator position. It can now be changed by operator input.

Activate manual mode with manual value

In this mode (MANGN_ON = FALSE and MAN_ON = TRUE), the manual output value is entered as an absolute value in the MAN input. The manual output value is displayed at measuring point MP8.

1. Select the "Activate manual mode" check box.
2. Choose the "Manual value" option.

Activate the manual value generator

The manual value MAN can be incremented or decremented in steps using the MANUP and MANDN switches. The modified value can be monitored at MP8. The configured manipulated variable limits are taken into account.

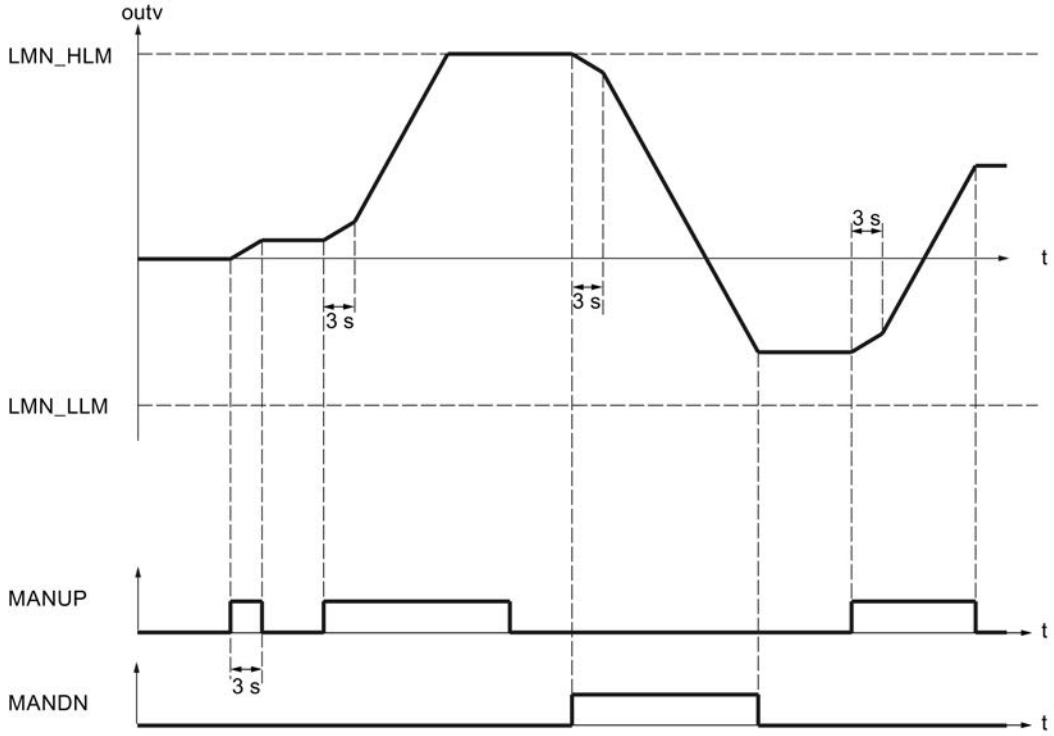
1. Select the "Activate manual mode" check box.
2. Select the "Manual value generator" option.

Effect of the switches MANUP and MANDN

For changes involving small steps, the controller should have a maximum sampling time of 100 ms.

The rate of change of the manipulated variable depends on how long the MANUP or MANDN switch is actuated and on the selected limits. The following relationship applies:

- During the first 3 s after setting MANUP or MANDN:
  doutv/dt = (LMN_HLM - LMN_LLM) / 100 s
- Afterwards:
  doutv/dt = (LMN_HLM - LMN_LLM) / 10 s



For a sampling time of 100 ms and a manipulated variable range of -100.0 to 100.0, the manipulated variable, for example, changes by 0.2 per cycle during the first 3 seconds. If MANUP is TRUE for more than 3 seconds, the rate of changes increases to ten times the value, in this case therefore to 2 per cycle.

The manual value generator is activated bumpless.

Switching the operating modes

The following table shows the possible operating modes of the step controller with the required values of the structure-defining switches.

Table 11-5 Operating modes of step controller

	MAN_ON	MNGN_ON	LMNS_ON
Automatic mode	FALSE	Any	FALSE
Manual mode without switch operation	TRUE	FALSE	FALSE
Manual mode with manual value generator	TRUE	TRUE	FALSE
Manual mode using actuating signals	Any	Any	TRUE

Limiting functions

With step controllers with position feedback, you can limit the manipulated variable.

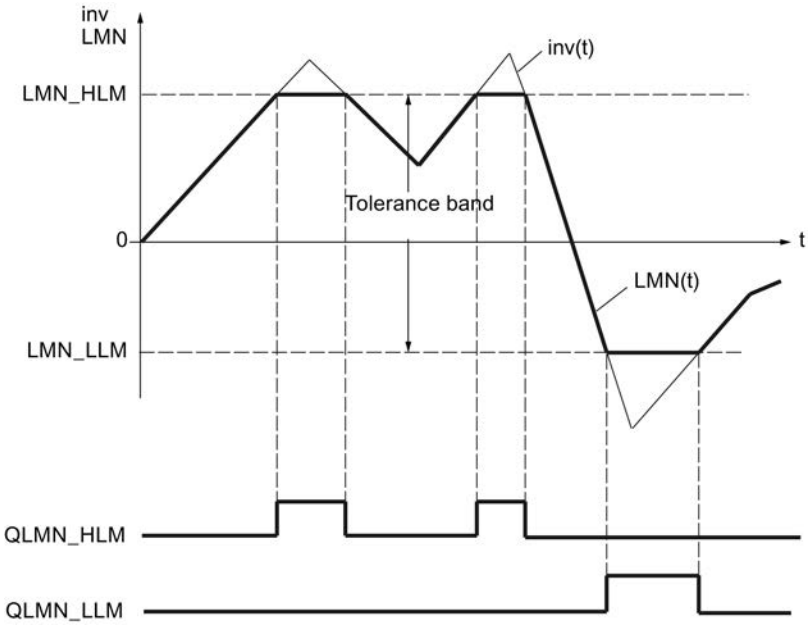
Limiting the absolute value of the manipulated variable

The operating range, i.e., the range in which the actuator can vary within the scope of permissible manipulated variables, is determined by the setting range of the manipulated variable. Since the limits for allowed manipulated variables usually do not coincide with the 0% or 100% limit of the correcting range, depending on the type of actuator, you can configure the high and low limit of the manipulated variable.

To avoid an impermissible status in the particular process, high and low limits for the setting range of the manipulated variable will be set in the manipulated variable branch of the closed-loop control.

The LMNLIMIT function limits manipulated variable LMN(t) to the low and high values LMN_HLM and LMN_LLM. These values can be specified. However, input variable inv must be outside these limits. Because the function cannot be deactivated, the specification of a low limit and high limit must always be taken into consideration in the configuration.

The numeric values of the limits (in percent) are set in the input parameters for the low and high limits. In the event of violations by input variable inv(t), the associated displays are output via the message outputs.



Startup and operating principles

- During a restart, all message outputs are set to zero.
- The limiting operates according to the following relationships:

LMN =	QLMN_HLM	QLMN_LLM	If:
LMN_HLM	TRUE	FALSE	inv ≥ LMN_HLM
LMN_LLM	FALSE	TRUE	inv ≤ LMN_LLM
inv	FALSE	FALSE	LMN_HLM < inv < LMN_LLM

The effective manipulated value of the controller is displayed at the output, i.e. at the parameter LMN or at the measuring point MP10.

Procedure

To limit the absolute value of the manipulated variable, proceed as follows:

1. In the "High limit" input field, enter the value for the maximum permissible value.
2. In the "Low limit" input field, enter the value for the minimum permissible value.

Pulse generator

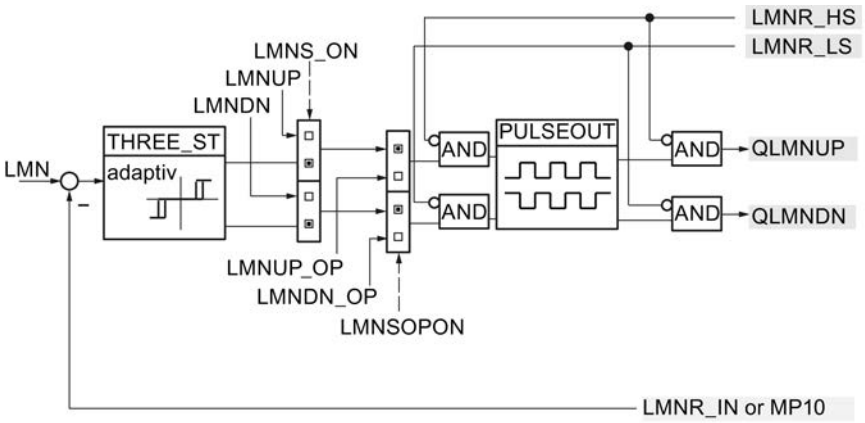
The pulse generator ensures that a minimum on and off time is adhered to in order to protect the actuators. Addressing the endstop signals of the actuator (LMNR_HS/LMNR_LS) causes the manipulated variable LMNUP or LMNDN to be blocked. This applies to automatic and manual mode.

If both signal switches are set for manipulated variable signal input (LMNUP = LMNDN = TRUE and LMNUP_OP = LMNDN_OP = TRUE), outputs QLMNUP and QLMNDN always return FALSE.

It is not possible to change directly from "Manipulated variable signal up" (QLMNUP = TRUE, QLMNDN = FALSE) to "Manipulated variable signal down" (QLMNUP = FALSE, QLMNDN = TRUE). The pulse generator inserts a cycle with QLMNUP = QLMNDN = FALSE.

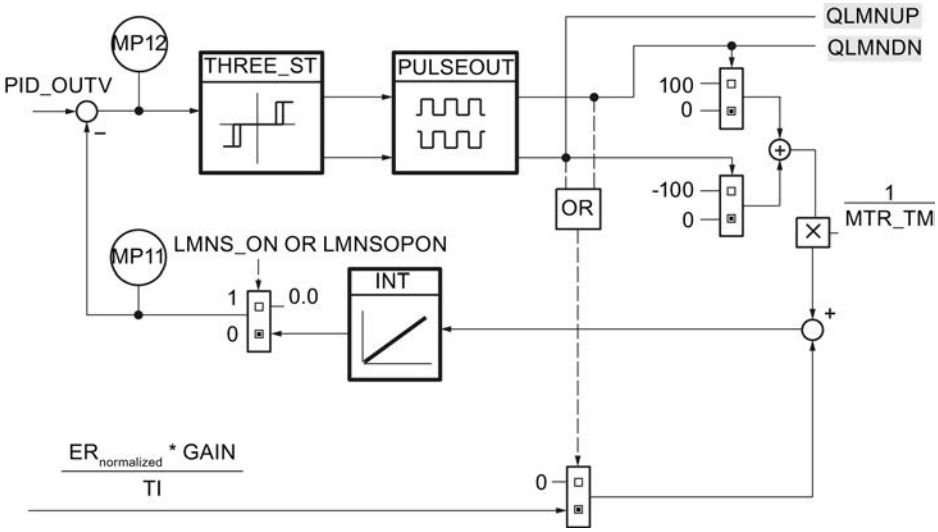
Input signal INV with step controllers with position feedback

The difference between manipulated variable LMN and position feedback LMNR is switched-through to the three-step element with hysteresis THREE_ST.



Input signal INV with step controllers without position feedback

The difference between the PD action of the controller and the feedback value (MP11) is switched to the three-step element with hysteresis THREE_ST.



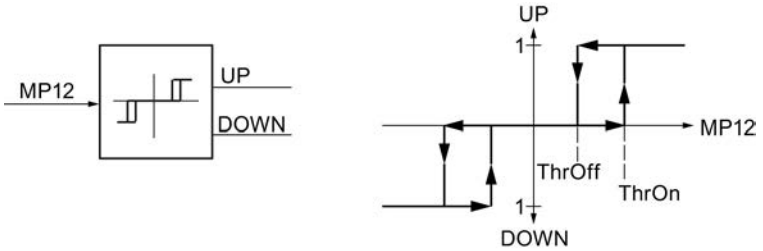
Influence of motor transition time on the three-step element

The three-step element generates two binary signals which are TRUE or FALSE depending on the value and sign of the differential value at the input.

The three-step element THREE_ST responds to the input signal INV according to the following relationships and hereby adopts in each case one of the three possible combinations of its output signals UP/DOWN:

UP	DOWN	Input combination
TRUE	FALSE	|INV| ≥ ThrOn or INV > ThrOff and UP_{old} = TRUE
FALSE	TRUE	|INV| ≤ -ThrOn or INV < -ThrOff and $DOWN_{old}$ = TRUE
FALSE	FALSE	|INV| ≤ ThrOff

ThrOn = Switch-on threshold, ThrOff = Switch-off threshold



The switch-off threshold ThrOff must be greater than the change in the position feedback or the position change after one pulse duration. This value is dependent on the transition time of the motor MTR_TM and is calculated as follows:

$$ThrOff = 0.5 * \frac{110}{MTR_TM} * CYCLE$$

MTR_TM must be an integer multiple of CYCLE.

Note

An excessively long motor transition time setting (10% over the actual transition time) will cause the manipulated variable signals QLMNUP and QLMNDN to constantly switch on and off.

Adaptation of the switch-on threshold ThrOn

To reduce the switching frequency when larger control deviations are corrected, the switch-on threshold ThrOn is adapted automatically during operation, while ThrOff remains constant. ThrOn is limited to:

$$Min\ ThrOn = \frac{100}{MTR_TM} * MAX\ (PULSE_TM, CYCLE)$$

$$Max\ ThrOn = 10$$

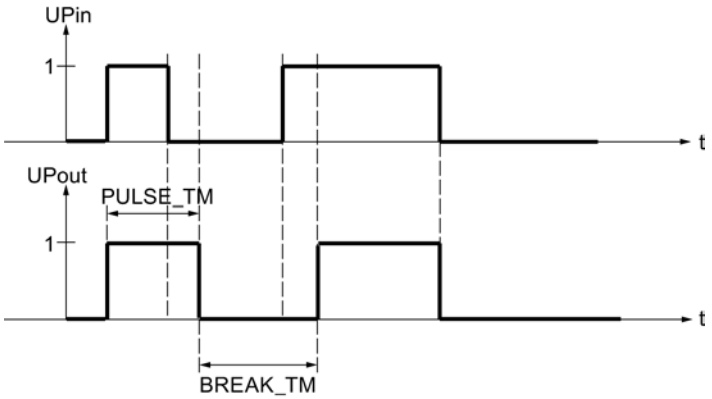
The adaptation of the switch-on threshold is disabled in pure P, D, and PD controllers. For example:

ThrOn = Min ThrOn

Minimum ON time and minimum OFF time.

The pulse generator ensures that a minimum on and off time is adhered to when the output pulses are switched on and off.

Parameters for a minimum ON time PULSE_TM and a minimum OFF time BREAK_TM can therefore be assigned to protect the actuator. The duration of the output pulse QLMNUP or QLMNDN is always greater than the minimum ON time and the break between two pulses always greater than the minimum OFF time. Using the UP signal as an example, the following figure shows how this works.



Procedure

To configure the pulse generator, proceed as follows:

1. Enter the transition time of the motor in the "Motor transition time" input field.
2. Enter the "Minimum ON time".
3. Enter the "Minimum OFF time".

Commissioning PID_ES

Performing tuning manually

Requirements

- The instruction and the technology object have been loaded to the CPU.
- MAN_ON = FALSE

Procedure

In order to manually determine the optimal PID parameters, proceed as follows:

1. Click the "Measurement on" icon.
   An online connection will be established, if one is not already present. The current values for the setpoint, process value and manipulated variable are recorded.
2. Enter new PID parameters in the "P", "I", "D" and "Time delay" fields.
3. Click on the icon "Send parameter to CPU" in the "Tuning" group.
4. Select the "Change setpoint" check box in the "Current values" group.
5. Enter a new setpoint and click in the "Current Values" group on the icon .
   If the setpoint's rate of change is limited, the changeover to the new setpoint will be bumpless
6. Clear the "Manual mode" check box, if necessary.
   The controller works with the new PID parameters and controls the new setpoint.
7. Check the quality of the PID parameter based on the curve points.
8. Repeat steps 3 to 8 until you are satisfied with the controller results.

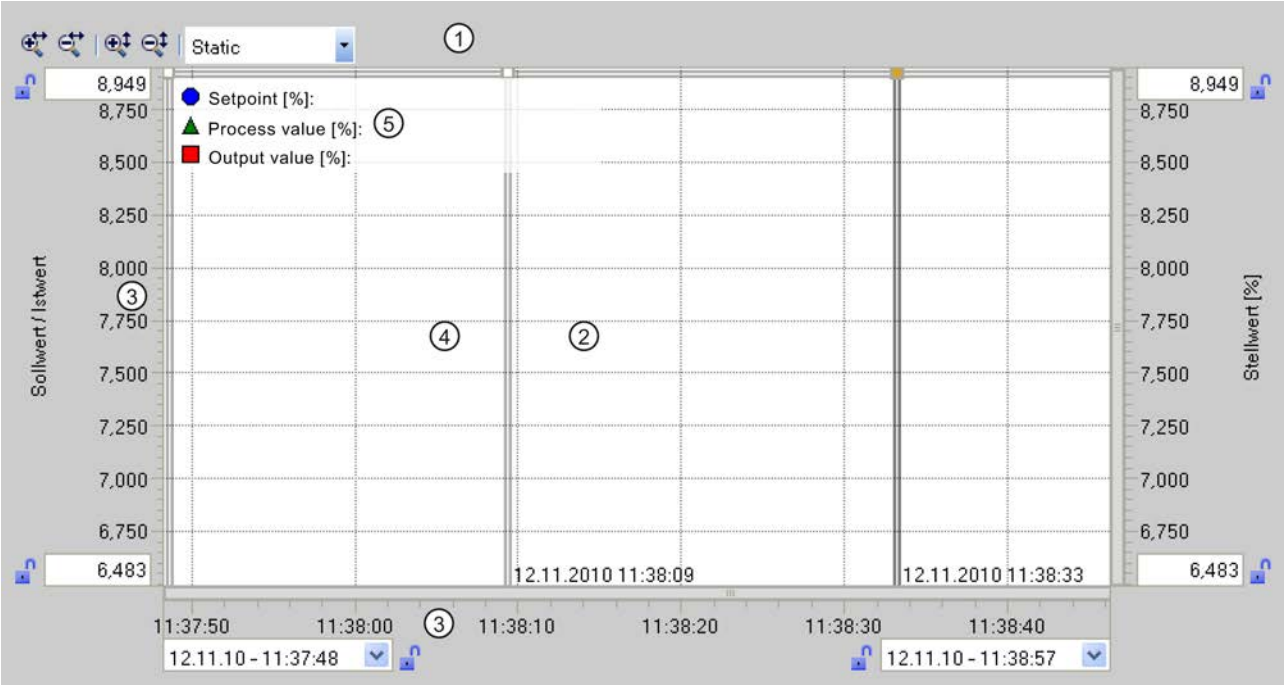
See also

Using the trend view (Page 9747)

Using the trend view

The trend view is used for graphic illustration of the setpoint, process value and output value. The values of the trend view are updated in the selected updating cycle. Start the recording of the trend values by clicking "Measurement on" and stop it by clicking "Measurement off".

Elements of the trend view



① Selection of the display mode
② Trend view
③ Area for moving and scaling the axes
④ Ruler
⑤ Legend with the trend values at the ruler

Selection of the display mode

The following display modes can be selected for displaying the trend recording:

- Strip (continuous display)
  New trend values are entered at the right-hand trend view. Previous trend values are scrolled to the left. The time axis cannot be moved.
- Scope (jumping area display)
  New trend values are entered within the trend view from left to right. When the right-hand margin of the trend view is reached, the monitoring area is moved one view width to the right. The time axis can be moved within the limits of the monitoring area.
- Sweep (rotating display)
  New trend values are displayed in the trend view in accordance with a rotating display. The trend values are entered from left to right. The trend values of the last rotating display are overwritten at the writing position. The time axis cannot be moved.
- Static (static area display)
  Writing of the trends is interrupted. Recording of the new trend values is carried out in the background. The time axis can be moved across the entire previous recording period.

The ratio of the visible area to the recorded area is shown in the ratio display:

The width of the ratio display represents the complete duration of the recording. The yellow section of the display represents that part of the recording that is visible in the trend view. The grayed section of the display represents that part of the recording that is **not** visible in the trend view.

In the "Static" display mode, you can move the visible range of the ratio display along the axis by dragging it with the mouse.

Trend view

The trends for the setpoint, process value and output value are displayed in the trend view. In addition to different colors, the trends are identified by symbols (refer to legend).

Moving and scaling the axes

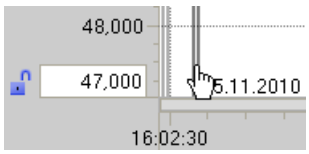
The axes for the setpoint and process value, for the output value and the time axis can be moved and scaled individually. The right and left mouse buttons are assigned alternating functions. You can use the following icons and mouse actions:

	Moving the setpoint / process value axes, or the output value axis up or down. The axis cannot be moved if the high or the low scaling value is blocked.
	Moving the time axis to the right or left. The axis cannot be moved if the left or right scaling value is blocked.
	Stretch and compress the setpoint , process value axes, or the output value axis. • The scaling of the axes is stretched or compressed symmetrically if none of the scaling values is blocked. • Blocked scaling values are retained when you stretch or compress an axis.
	Stretching and compressing the time axis • The scaling of the axis is stretched or compressed symmetrically if none of the scaling values is blocked. • Blocked scaling values are retained when you stretch or compress an axis.
	Stretching and compressing the setpoint / process value axes, or the output value axis. The low scaling value is not changed as a result of stretching or compressing.
	Stretch and compress the setpoint / process value or the output value axis. The high scaling value is not changed during stretching or compressing.
	Stretching and compressing the time axis. The right scaling value is not changed as a result of stretching or compressing.
	Stretching and compressing the time axis. The left scaling value is not changed as a result of stretching or compressing.
50,000	Enter a scaling value.
50,000	The current scaling value can be blocked using the padlock icon. Only one value of a respective axis can be blocked.
	Double-clicking in the trend view optimizes the scaling and position of the setpoint, process value and output value in the trend view.
	Double-clicking in the axis area restores the default position and scaling of the axis.

Working with rulers

Use one or several rulers to analyze discrete values of the trend profile.

The park positions for vertical rulers are found on the left edge of the curve area, for horizontal rulers on the high edge.



Move the mouse to the left edge of the trend area and watch out for the transition of the mouse pointer. Drag a vertical ruler onto the measuring trend that you want to analyze.

The trend values on the vertical ruler are displayed on the left of the ruler. The time of the ruler position is displayed at the base of the ruler.

The trend values of the active ruler are displayed in the legend. If several rulers are dragged to the trend area, the respectively last ruler is active. The active ruler is indicated by the correspondingly colored symbol.

However, the values of horizontal rulers cannot be displayed.

You can reactivate an inactive ruler by clicking it.

Use the shortcut ALT+Click to remove rulers which are no longer required.

Commissioning with the PID Self Tuner

You can commission the PID_ES instruction and determine the optimum PID parameters with the PID Self Tuner.

Procedure

To commission a step controller, proceed as follows:

1. Call the instruction PID_ES in a cyclic interrupt OB and interconnect the input and output parameters.
2. Call the instruction TUN_ES in the same cyclic interrupt OB.

3. Interconnect the instruction TUN_ES as follows with PID_ES:
   - TUN_ES.PV with PID_ES.PV
   - If you are using a position feedback, TUN_ES.LMNR with PID_ES.LMNR_IN
   - TUN_ES.C_LMNUP and TUN_ES.C_LMNDN with PID_ES.QLMN_UP and PID_ES.QLMNDN
   - TUN_ES.LMNR_HS with PID_ES.LMNR_HS
   - TUN_ES.LMNR_ON with PID_ES.LMNR_ON
   - TUN_ES.LMNS_ON with PID_ES.LMNS_ON
   - TUN_ES.SP_OUT with PID_ES.SP_IN or PID_ES.SP_EXT
   - TUN_ES.GAIN with PID_ES.GAIN
   - TUN_ES.TI with PID_ES.TI
   - TUN_ES.TD with PID_ES.TD
   - TUN_ES.TM_LAG with PID_ES.TM_LAG
   - TUN_ES.MTR_TM with PID_ES.MTR_TM
   - If you are using a deadband, TUN_ES.DEADB_W with PID_ES.DEADB_W
   - TUN_ES.QLMNUP and TUN_ES.QLMNDN with PID_ES.LMNUP and PID_ES.LMNDN
4. Click the "" icon on the instruction TUN_ES.
   The commissioning editor for TUN_ES opens. The operation of the commissioning editor is described with the PID Self Tuner.

11.3.5.2 Using modular PID control

Scope of Modular PID Control functions

In many control tasks, the classic PID controller which controls the processes is not the only key element. Complex signal processing is also required.

Modular PID Control includes instructions with functions for processing the process value and setpoint and calculating and tuning the manipulated variable.

The following controllers are possible:

- Continual controller
- Step controller for actuators with integrating behavior
- Pulse controller

You can implement controllers for different applications by connecting individual instructions.

- Fixed value controller
- Cascade controller
- Ratio controller
- Mixing controller
- Split range controller

- Override controller
- Controller with disturbance variable processing
- Multi-variable controller

Both slow (temperature, filling levels etc.) and fast controlled systems (flow, pressure, etc.) can be controlled.

You can call multiple control loops equidistantly but with different sampling intervals using the instruction LP_SCHED_M. You alter the sampling time of the control loop in accordance with the slowness of the controlled system.

This ensures the even distribution of the CPU load.

Use the PID Self-Tuner to commission and fine-tune the PID parameters.

Basic steps for setting up a controller FB

Basic procedure for programming a controller

You program a controller as the function block (controller FB), for example with STL. Your controller FB must have at least the following parameters:

- Input parameter COM_RST
- Input parameter CYCLE
- Input parameter for setpoint
- Input parameter for process value
- Output parameter for the manipulated variable
- In/out parameter for the manual output value (continual controller)

Depending on controller requirements, you can also set the following parameters

- Input parameter for position feedback (step controller)
- Input parameter for disturbance variable
- Output parameter for the processed process value
- Output parameter for the processed setpoint
- Output parameter for limit violation

The details of how to set the inputs and outputs depend on your specific situation, for example on whether the process value is in I/O or floating-point format, on the type of position feedback, etc.

Call the Modular PID Control instructions in the controller FB and connect the input and output parameters. Call the controller FB you have created, for example in OB 100 or in a cyclic interrupt OB.

The following steps are needed to program a controller with Modular PID Control.

Step	Description
1.	Processing setpoints (Page 9753)
2.	Processing the process value (Page 9755)
3.	Generating and processing the control deviation (Page 9757)
4.	Applying the disturbance variable (Page 9759) (optional)
5.	Calculating the manipulated variable • Creating a continual controller (Page 9762) • Creating a pulse controller (Page 9763) • Creating a step controller (Page 9767)
6.	Processing position feedback (Page 9768)

See also

Processing setpoints (Page 9753)

Processing the process value (Page 9755)

Generating and processing the control deviation (Page 9757)

Applying the disturbance variable (Page 9759)

Creating a continual controller (Page 9762)

Creating a pulse controller (Page 9763)

Creating a step controller (Page 9767)

Processing position feedback (Page 9768)

Programming the controller FB

Processing setpoints

Instructions for processing the setpoint

You can use the following instructions:

- SP_GEN: Enter setpoint manually
- ROC_LIM: Limit rate of change
- LAG1ST: Smooth setpoint
- NONLIN: Linearize setpoint
- LIMITER: Limit setpoint
- RMP_SOAK: Specify setpoint according to ramp/soak
- SCALE_M: Scale setpoint
- NORM: Normalize setpoint

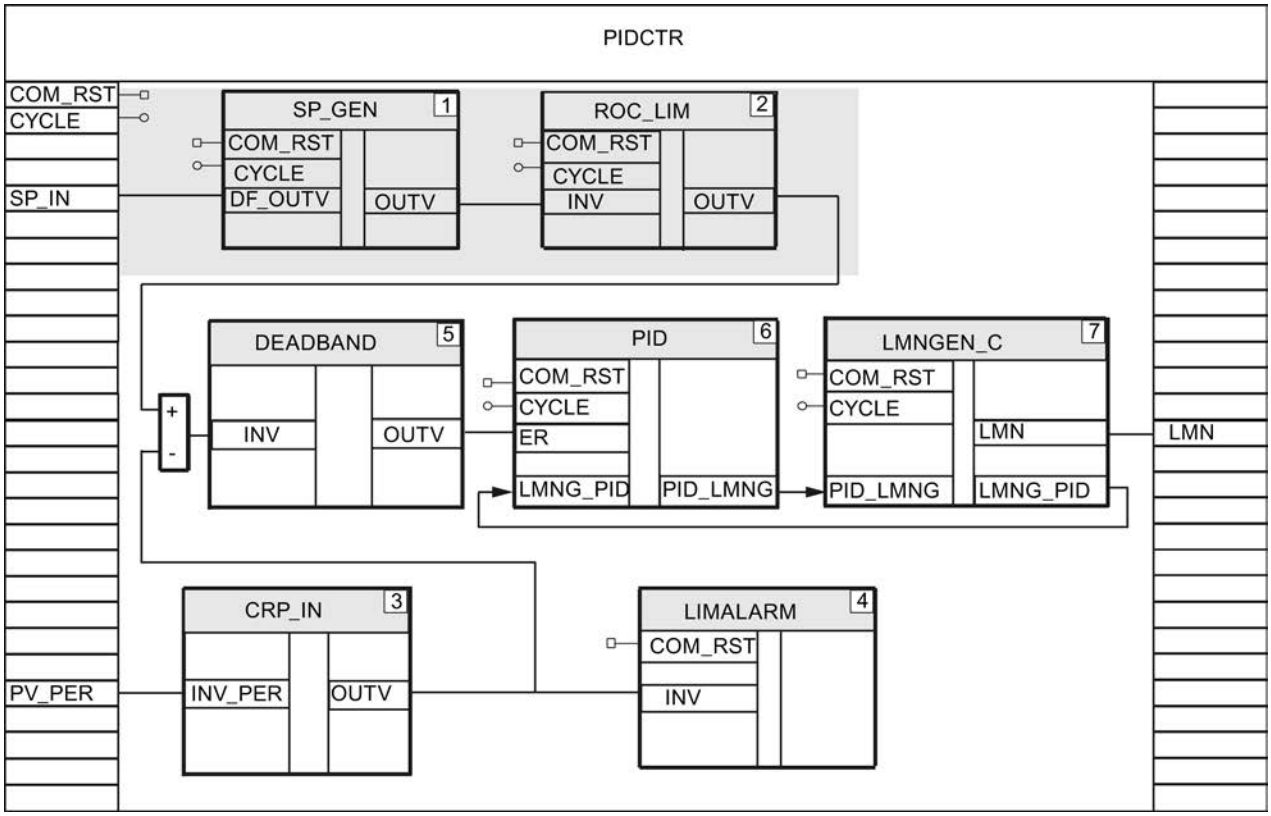
Example of setpoint processing

It should be possible to increase or decrease a specified setpoint manually. The rate of change is limited to prevent jumps in the setpoint.

Proceed as follows to process the setpoint as described in the example:

1. At the controller FB, in this case PIDCTR, declare an input parameter SP_IN of the data type "Real".
2. Connect SP_IN to a variable containing the required setpoint.
3. Call the instruction SP_GEN as a multi-instance in the controller FB and connect SP_IN to SP_GEN.DF_OUTV.
4. Assign parameters for the high and low setpoint limit in the SP_GEN_Instance structure of the multi-instance.
5. Call the instruction ROC_LIM as a multi-instance in the controller FB and connect SP_GEN.OUTV to ROC_LIM.INV via a temporary variable.
6. Assign parameters for the maximum rates of change in the ROC_LIM_Instance structure of the multi-instance.
7. To generate the control deviation, subtract the processed process value from ROC_LIM.OUTV or connect ROC_LIM.OUTV to ERR_MON.SP.

The diagram below shows how the Modular PID Control instructions are called and connected in the controller FB PIDCTR.



See also

Processing the process value (Page 9755)
Description SP_GEN (Page 7022)
Description ROC_LIM (Page 7014)
Description LAG1ST (Page 6963)
Description NONLIN (Page 6986)
Description LIMITER (Page 6972)
Description RMP_SOAK (Page 7007)
Description SCALE_M (Page 7021)
Description NORM (Page 6990)

Processing the process value

Instructions for processing the process value

You can use the following instructions:

- CRP_IN: Scale I/O value
- LAG1ST: Smooth process value
- LIMITER: Limit process value
- LIMALARM: Monitor process value
- NONLIN: Linearize process value
- SCALE_M: Scale process value
- NORM: Normalize process value

Example of process value processing

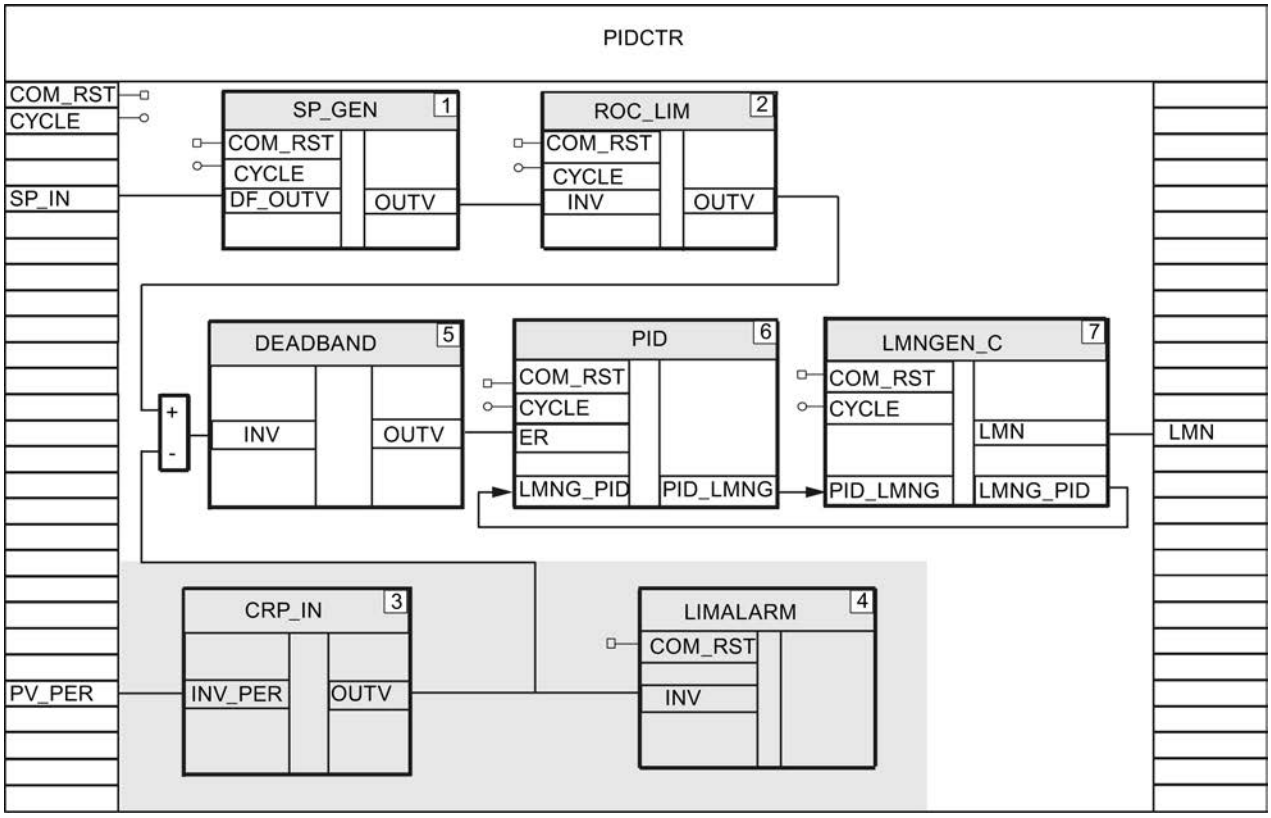
The process value is in I/O format and is to be monitored to ensure compliance with the process value limits.

Proceed as follows to process the process value as described in the example:

1. At the controller FB, in this case PIDCTR, declare an input parameter PV_PER of the data type "Word".
2. Connect PV_PER to the I/O input at which the process value is measured.
3. Call the instruction CRP_IN as a multi-instance in the controller FB, for example with the name CRP_IN_Instance_PV.
4. Connect PV_PER to CRP_IN.INV_PER.
5. Assign parameters for factor and offset for scaling in the CRP_IN_Instance_PV structure of the multi-instance.

6. Call the instruction LIMALARM as a multi-instance in the controller FB and connect CRP_IN.OUTV to LIMALARM.INV via a temporary variable.
7. Assign parameters for the process value limits in the LIMALARM_Instance structure of the multi-instance.

The diagram below shows how the Modular PID Control instructions are called and connected in the controller FB PIDCTR.



See also

Processing setpoints (Page 9753)

Generating and processing the control deviation (Page 9757)

Description CRP_IN (Page 6948)

Description LAG1ST (Page 6963)

Description LIMITER (Page 6972)

Description LIMALARM (Page 6970)

Description NONLIN (Page 6986)

Description SCALE_M (Page 7021)

Description NORM (Page 6990)

Generating and processing the control deviation

Establishing and limiting the control deviation

There are two ways to establish the control deviation. Either program the subtraction of the process value from the setpoint, or use the instruction ERR_MON. The instruction ERR_MON already includes control deviation monitoring.

Instructions for processing the control deviation

You can use the following instructions:

- A_DEAD_B: Route control deviation through adaptive deadband
- DEADBAND: Route control deviation through deadband

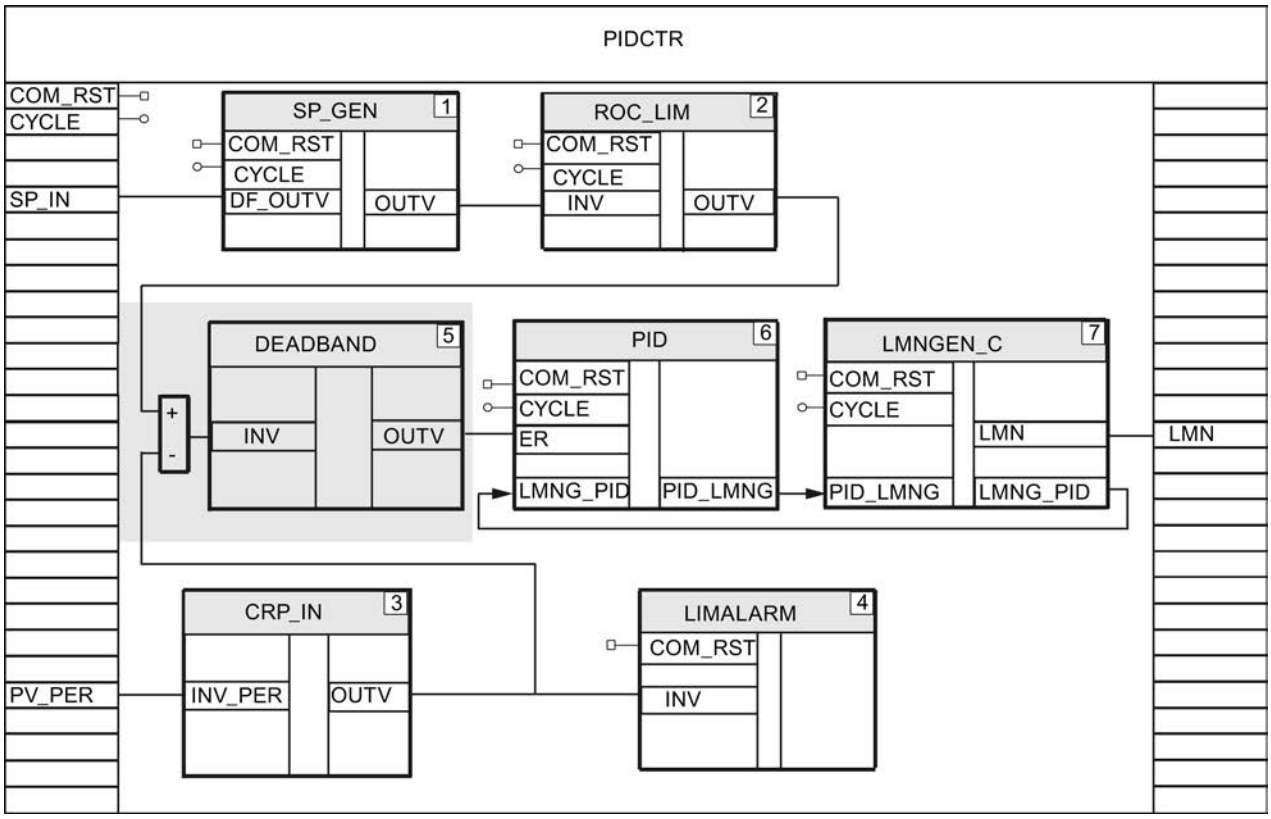
Example of subtraction

The control deviation is produced by subtraction and routed through a deadband.

Proceed as follows to determine the control deviation as described in the example:

1. Call the instruction DEADBAND as multi-instance and connect DEADBAND.INV to the difference between setpoint and process value, e.g. ROC_LIM.OUTV - CRP_IN.OUTV.
2. Assign parameters for the deadband width and a deadband displacement in the DEADBAND_Instance structure of the multi-instance.
   The processed control deviation is output at DEADBAND.OUTV.

The diagram below shows how the Modular PID Control instructions are called and connected in the controller FB PIDCTR.



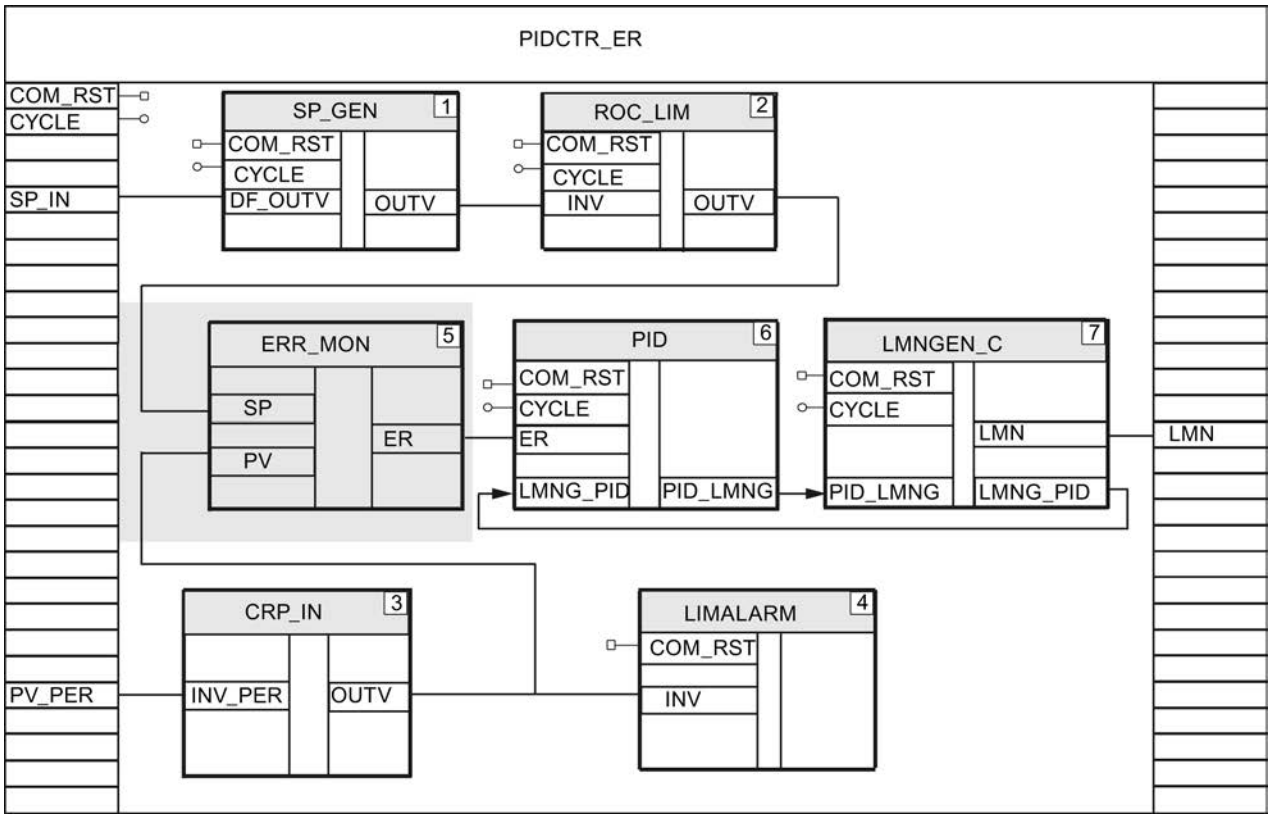
Example of the use of ERR_MON

The control deviation is calculated and monitored with the instruction ERR_MON.

Proceed as follows to determine the control deviation as described in the example:

1. Call the ERR_MON instruction as a multi-instance.
2. Using temporary variables, connect the processed setpoint to ERR_MON.SP, e.g. ROC_LIM.OUTV, and the processed process value to ERR_MON.PV, e.g. CRP_IN.OUTV. The control deviation is output at ERR_MON.ER.
3. Assign parameters for control deviation monitoring in the ERR_MON_Instance structure of the multi-instance.

The diagram below shows how the Modular PID Control instructions are called and connected in the controller FB PIDCTR_ER.



See also

Processing the process value (Page 9755)
Applying the disturbance variable (Page 9759)
Description A_DEAD_B (Page 6944)
Description DEADBAND (Page 6953)
Description ERR_MON (Page 6957)

Applying the disturbance variable

Instructions for processing the disturbance variable

You can use the following instructions:

- CRP_IN: Scale I/O value
- DIF: Respond to rapid changes
- LAG1ST: Filter disturbances
- NON_LIN: Linearize disturbance variable
- SCALE_M: Scale disturbance variable
- NORM: Normalize disturbance variable

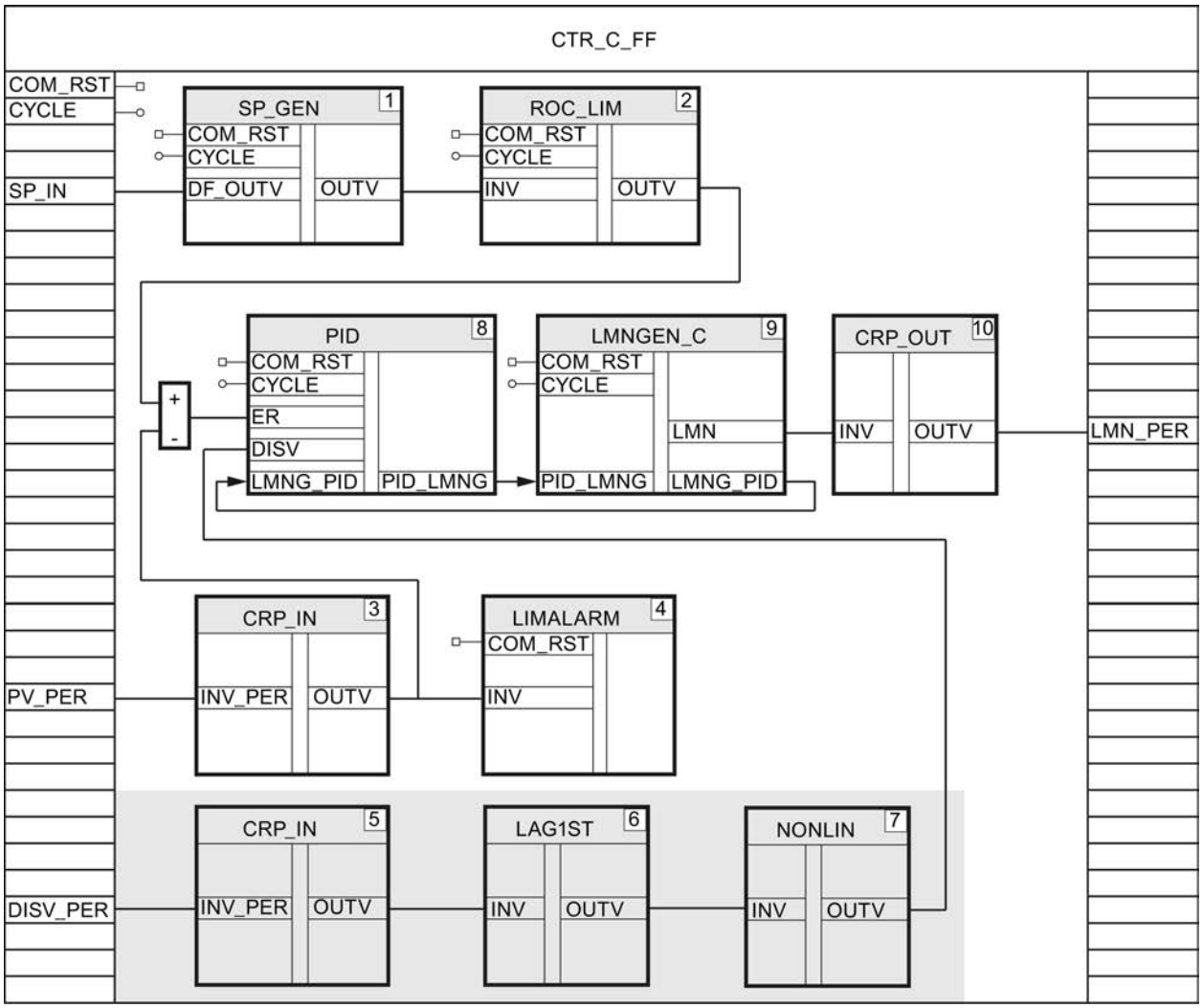
Example of disturbance variable processing

The disturbance variable is in I/O format. The signal is to be smoothed and linearized with a characteristic curve.

Proceed as follows to process the disturbance variable as described in the example:

1. At the controller FB, declare an input parameter DISV_PER of the data type "Word".
2. Connect DISV_PER to the I/O input at which the disturbance variable is measured.
3. Call the instruction CRP_IN as a multi-instance in the controller FB, for example with the name CRP_IN_Instance_DISV.
4. Connect DISV_PER to CRP_IN.INV_PER.
5. Assign parameters for factor and offset for scaling in the CRP_IN_Instance_DISV structure of the multi-instance.
6. Call the instruction LAG1ST as a multi-instance in the controller FB and connect CRP_IN.OUTV to LAG1ST.INV via a temporary variable.
7. Assign parameters for the time delay in the LAG1ST_Instance structure of the multi-instance.
8. Call the instruction NONLIN as a multi-instance in the controller FB and connect LAG1ST.OUTV to NONLIN.INV via a temporary variable.
9. Create a global data block DB_NBR_NONLIN in which to save the time slices of the characteristic curve.
10. Assign parameters for the number of the global data block with the characteristic curve in the NONLIN_Instance structure of the multi-instance.
11. The processed disturbance variable is output at NONLIN.OUTV.

The diagram below shows how the Modular PID Control instructions are called and connected in the controller FB CTR_C_FF.



See also

Generating and processing the control deviation (Page 9757)

Creating a continual controller (Page 9762)

Creating a pulse controller (Page 9763)

Creating a step controller (Page 9767)

Description CRP_IN (Page 6948)

Description DIF (Page 6955)

Description LAG1ST (Page 6963)

Description NONLIN (Page 6986)

Description SCALE_M (Page 7021)

Description NORM (Page 6990)

Creating a continual controller

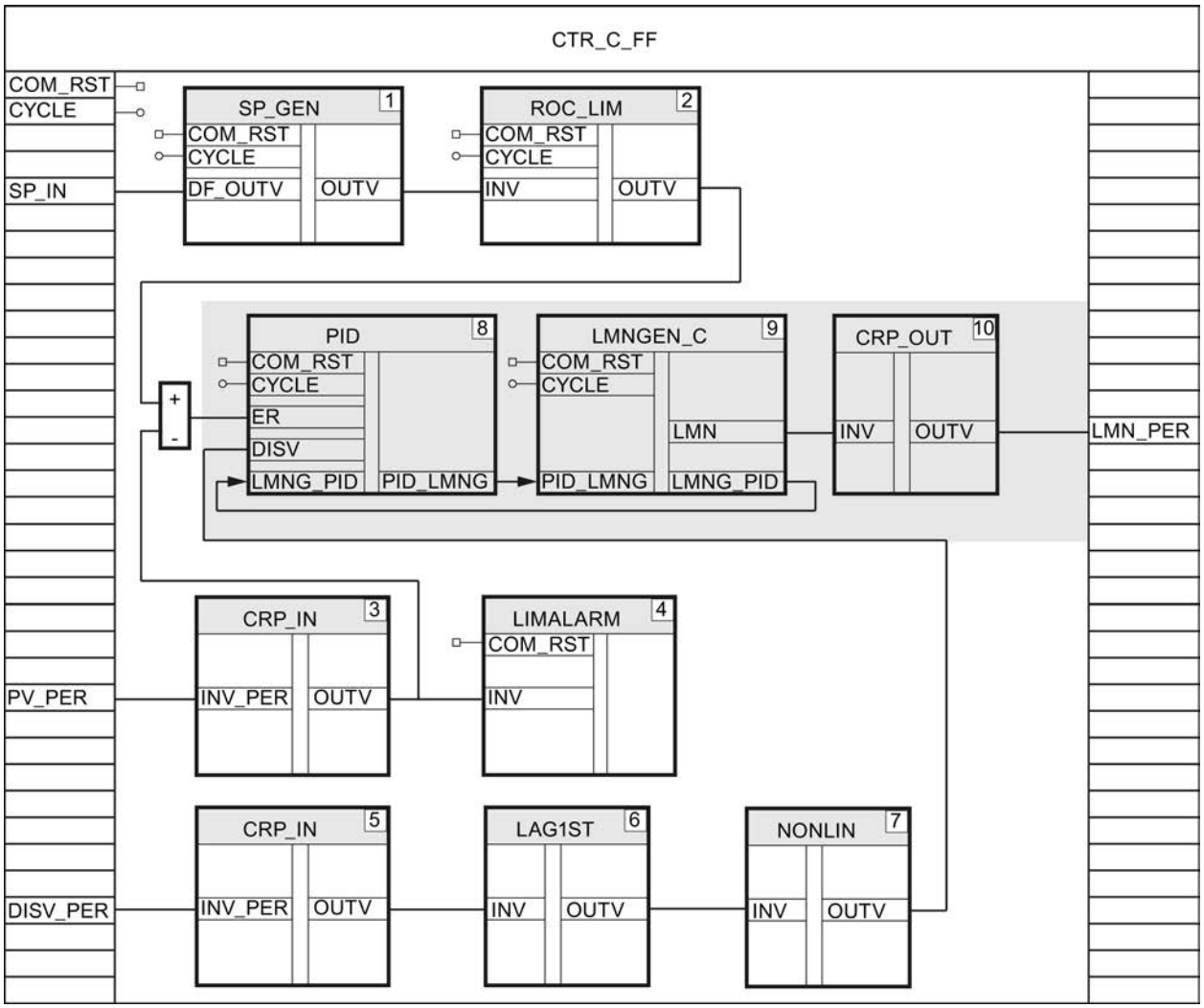
You will need the instructions PID, LMNGEN_C and CRP_OUT for a continual controller with an analog manipulated variable.

Procedure

Follow these steps to program a continual controller with an analog manipulated variable:

1. At the controller FB, declare an in/out parameter MAN of the data type "Real".
   You will require MAN to commission the controller with the PID Self-Tuner.
2. Call the instructions PID, LMNGEN_C and CRP_OUT as multi-instances in the controller FB.
3. At the controller FB, declare an in/out parameter MAN of the data type "Real" and connect MAN to LMNGEN_C.MAN.
4. Assign the parameters for the response of the control algorithm, e.g. whether the derivative action is used, in the PID_Instance structure of the controller DB.
5. Assign parameters for the manipulated value and rate of change limits, for example, in the LMNGEN_C_Instance structure of the multi-instance.
6. Assign parameters for the factor and offset for converting the floating-point value to I/O format in the CRP_OUT_Instance structure of the multi-instance.
7. Connect PID.ER to the control deviation via a temporary variable, e.g. ROC_LIM.OUTV - CRP_IN.OUTV or ERR_MON.ER.
8. If you are connecting a disturbance variable, connect PID.DISV to the processed signal of the disturbance variable, e.g. NONLIN.OUTV, using a temporary variable.
9. Connect the output parameter PID.PID_LMNG to the input parameter LMNGEN_C.PID_LMNG and the output parameter LMNGEN_C.LMNG_PID to the input parameter PID.LMNG_PID with the instruction BLKMOV.
10. Connect LMNGEN_C.LMN to CRP_OUT.INV using a temporary variable.
11. At the controller FB, declare an output parameter LMN_PER of the data type "Word" and connect CRP_OUT.OUTV to LMN_PER.
12. Connect LMN_PER to the analog output of the CPU which controls the actuator.

The diagram below shows how the Modular PID Control instructions are called and connected in the controller FB CTR_C_FF .



See also

Applying the disturbance variable (Page 9759)

Description PID (Page 6995)

Description LMNGEN_C (Page 6974)

Description CRP_OUT (Page 6950)

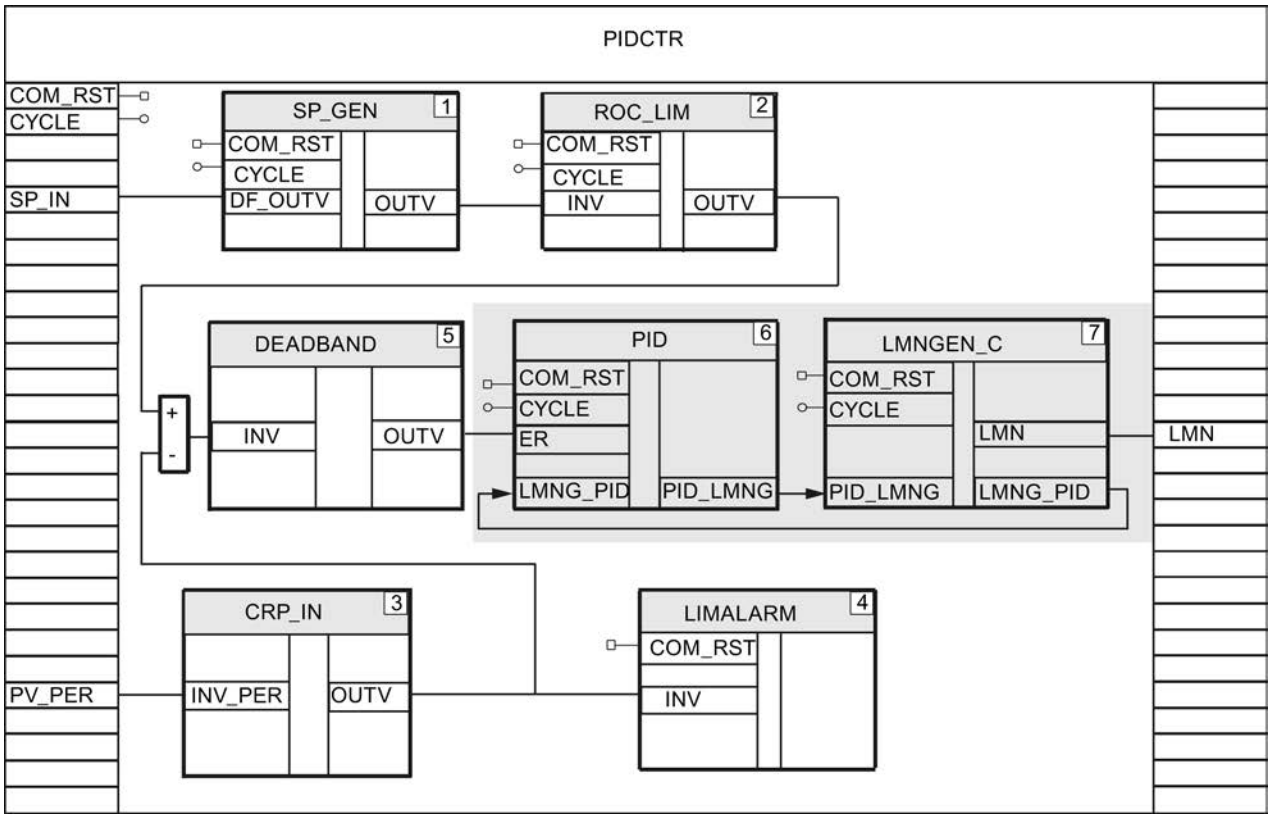
Creating a pulse controller

For a pulse controller, you generate a continuous manipulated variable in floating-point format with the instructions PID and LMNGEN_C. On the basis of this continuous manipulated variable, you then generate a pulse train with pulse width modulation using the instruction PULSEGEN_M.

Generating a continuous manipulated variable in floating-point format

1. At the controller FB, in this case PIDCTR, declare an output parameter LMN of the data type "Real".
2. Call the instructions PID and LMNGEN_C as multi-instances in the controller FB.
3. Connect and assign parameters for PID and LMNGEN_C as described for the continual controller.
4. Connect LMNGEN_C.LMN to the output parameter LMN of the controller FB.

The diagram below shows how the Modular PID Control instructions are called and connected in the controller FB PIDCTR.



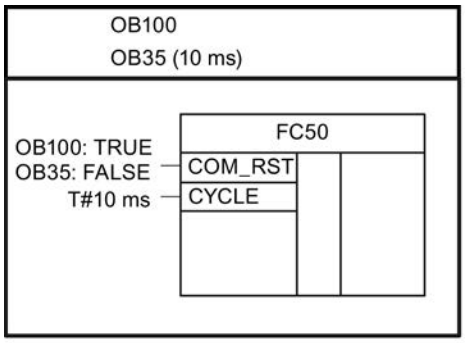
Generating a pulse train

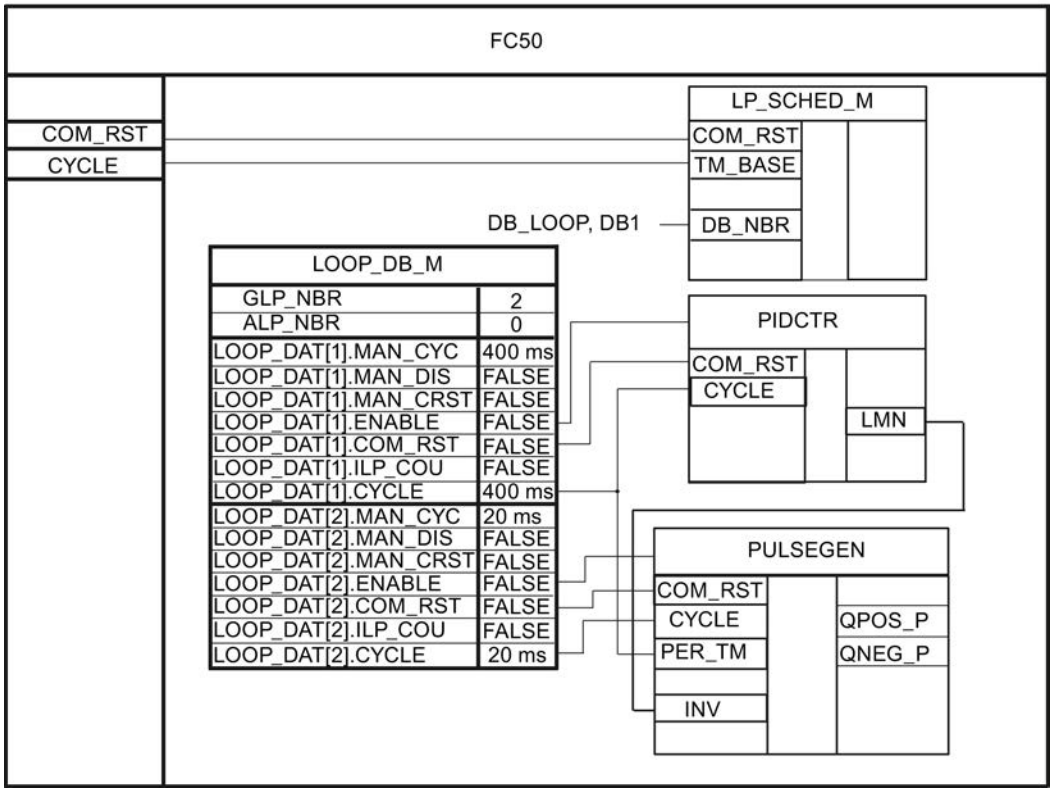
Follow these steps to generate a pulse train from the continuous manipulated variable:

1. Create a function, e.g. FC50, with the input parameters COM_RST and CYCLE.
2. Create a global data block LOOP_DB_M in which to assign parameters for the cycle times for calling the controller FB and the pulse generator PULSEGEN_M.
   The ratio of cycle time of the controller FB call to PULSEGEN_M should be ≥ 20.
3. Call the LP_SCHED_M instruction in the function.
4. Connect LP_SCHED_M.DB_NBR to the global data block LOOP_DB_M.
5. Call your controller FB, e.g. PIDCTR, if LOOP_DB_M.LOOP_DAT[1].ENABLE = TRUE.

6. Connect LOOP_DB_M.LOOP_DAT[1].COM_RST and LOOP_DB_M.LOOP_DAT[1].CYCLE to the input parameters COM_RST and CYCLE of the controller FB.
7. Set LOOP_DB_M.LOOP_DAT[1].ENABLE = FALSE.
8. Call the instruction PULSEGEN_M if LOOP_DB_M.LOOP_DAT[2].ENABLE = TRUE.
9. Connect LOOP_DB_M.LOOP_DAT[2].COM_RST and LOOP_DB_M.LOOP_DAT[2].CYCLE to PULSEGEN_M.COM_RST and PULSEGEN_M.CYCLE.
10. Connect PULSEGEN_M.INV to the output parameter LMN of your controller FB.
11. Connect PULSEGEN_M.PER_TM to the input parameter CYCLE of your controller FB.
12. Set LOOP_DB_M.LOOP_DAT[2].ENABLE = FALSE.

13. Connect PULSEGEN_M.QPOS_P and, if necessary, PULSEGEN_M.QNEG_P to the digital outputs of the CPU which control the actuator.
14. Call the function in OB 100 and in the cyclic interrupt OB.





See also

Applying the disturbance variable (Page 9759)

Description PID (Page 6995)

Description LMNGEN_C (Page 6974)

Description PULSEGEN_M (Page 6999)

Description LP_SCHED_M (Page 7028)

Creating a step controller

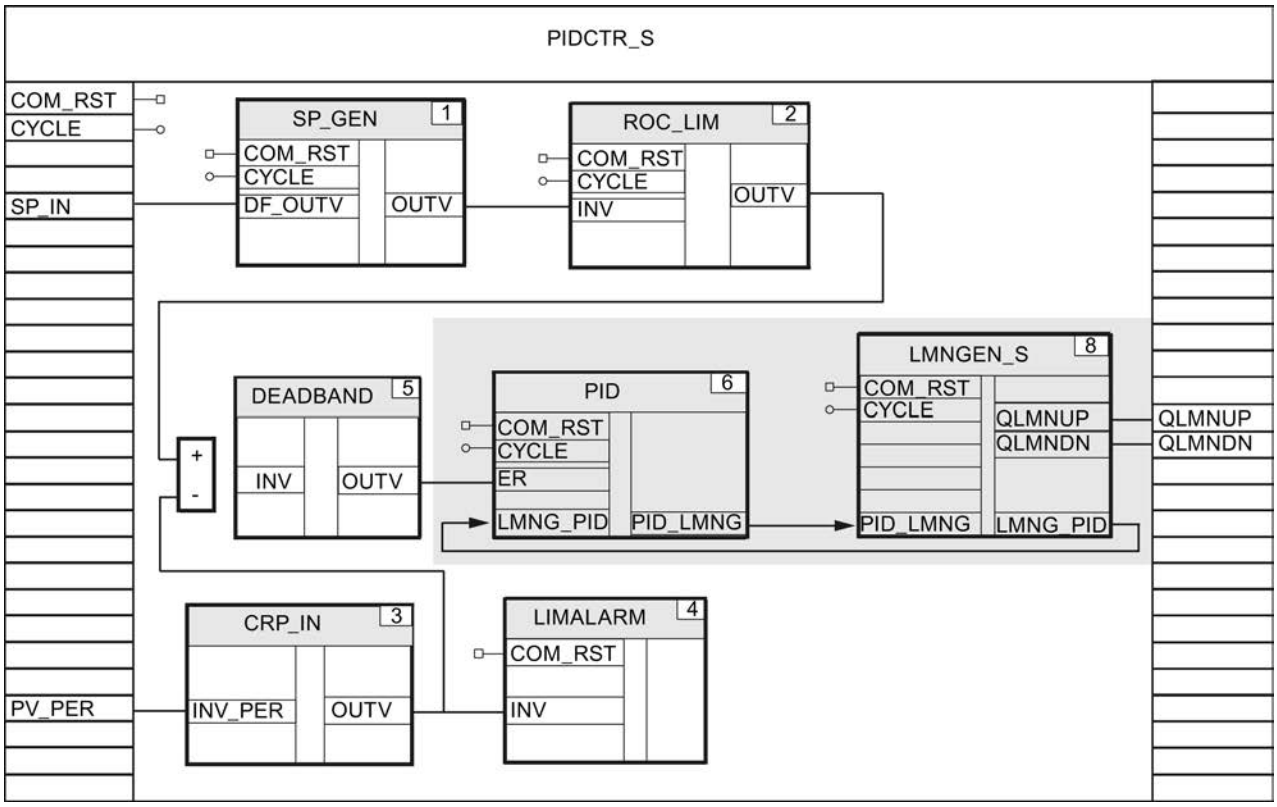
You will require the instructions PID and LMNGEN_S for a step controller.

Procedure

Proceed as follows to program a step controller:

1. Call the instructions PID and LMNGEN_S as multi-instances in the controller FB.
2. Assign the parameters for the response of the control algorithm, e.g. whether the derivative action is used, in the PID_Instance structure of the multi-instance.
3. Assign parameters in the LMNGEN_S_Instance structure of the multi-instance, for example specify that no position feedback is to be used, to LMNR_ON = FALSE.
4. Connect PID.ER to the control deviation via a temporary variable, e.g. DEADBAND.OUTV or ERR_MON.ER.
5. If you are connecting a disturbance variable, connect PID.DISV to the processed signal of the disturbance variable, e.g. CRP_IN.OUTV, using a temporary variable.
6. Connect the output parameter PID.PID_LMNG to the input parameter LMNGEN_S.PID_LMNG and the output parameter LMNGEN_S.LMNG_PID to the input parameter PID.LMNG_PID with the instruction BLKMOV.
7. At the controller FB, declare the output parameters QLMNUP and QLMNDN of the data type "Bool".
8. Connect LNMGEN_S.QLMNUP and LMNGEN_S. QLMNDN to these output parameters.
9. Connect the output parameters of the controller FB to the digital outputs of the CPU which control the actuator.

The diagram below shows how the Modular PID Control instructions are called and connected in the controller FB PIDCTR_S.



See also

Applying the disturbance variable (Page 9759)

Processing position feedback (Page 9768)

Description PID (Page 6995)

Description LMNGEN_S (Page 6979)

Processing position feedback

Position feedback configuration depends upon the actuator used.

- Actuator with analog position feedback
- Actuator with digital endstop signals

Requirement

You have already programmed a step controller.

Creating a step controller with analog position feedback

The following instructions are useful for processing analog position feedback.

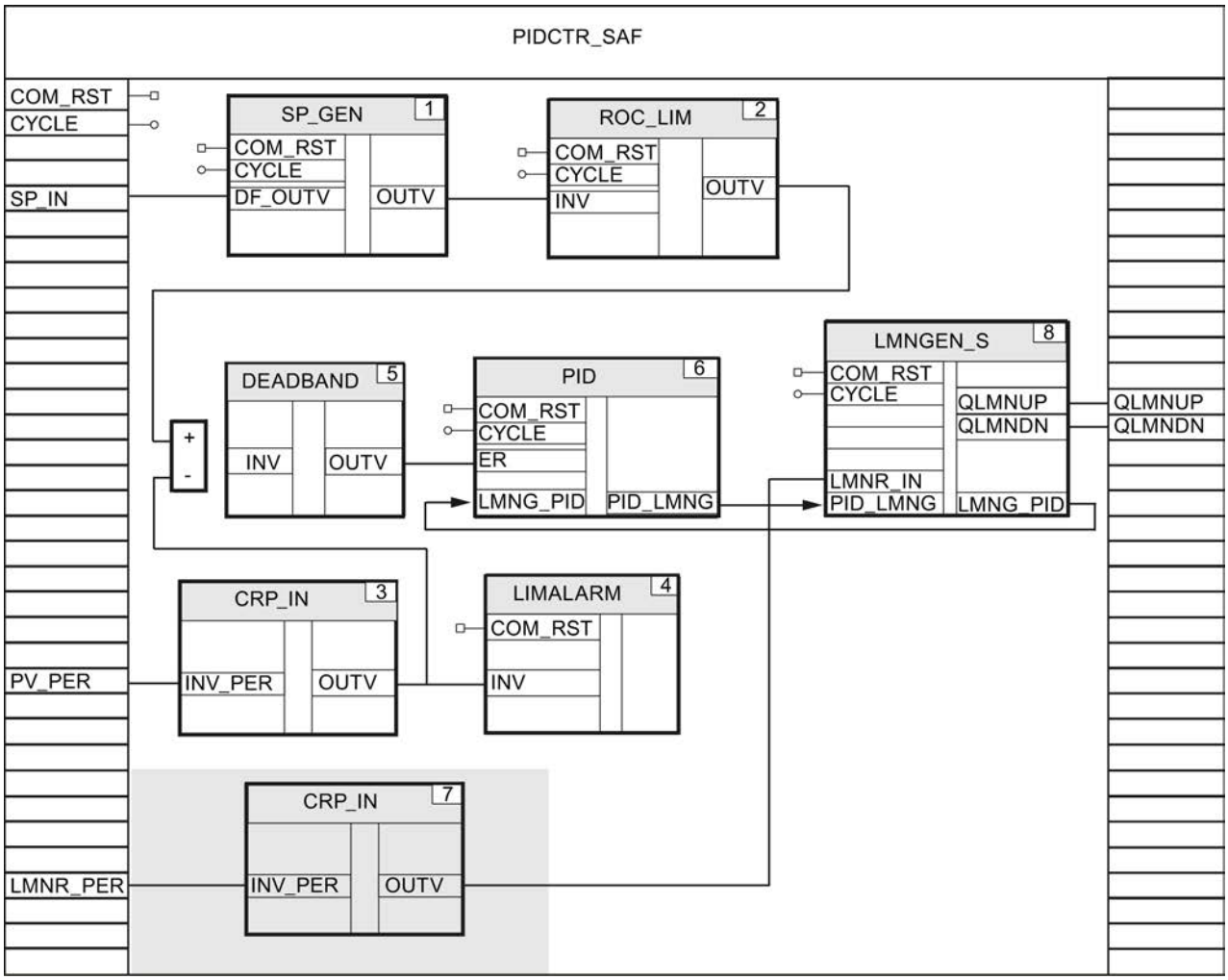
- CRP_IN: Scale I/O value
- NONLIN: Linearize position feedback
- SCALE_M: Scale position feedback
- NORM: Normalize position feedback

Example of analog position feedback

An actuator provides position feedback in I/O format; this should to be factored into the manipulated variable calculation.

1. At the controller FB, declare an input parameter LMNR_PER of the data type "Word".
2. Connect LMNR_PER to the I/O input.
3. Call the instruction CRP_IN as a multi-instance in the controller FB, for example with the name CRP_IN_Instance_LMNR.
4. Connect LMNR_PER to CRP_IN.INV_PER.
5. Assign parameters for factor and offset for scaling in the CRP_IN_Instance_LMNR structure of the multi-instance.
6. Connect CRP_IN.OUTV to LMNGEN_S.LMNR_IN using a temporary variable.
7. Activate position feedback with LMNR_ON = TRUE in the LMNGEN_S_Instance structure of the multi-instance.

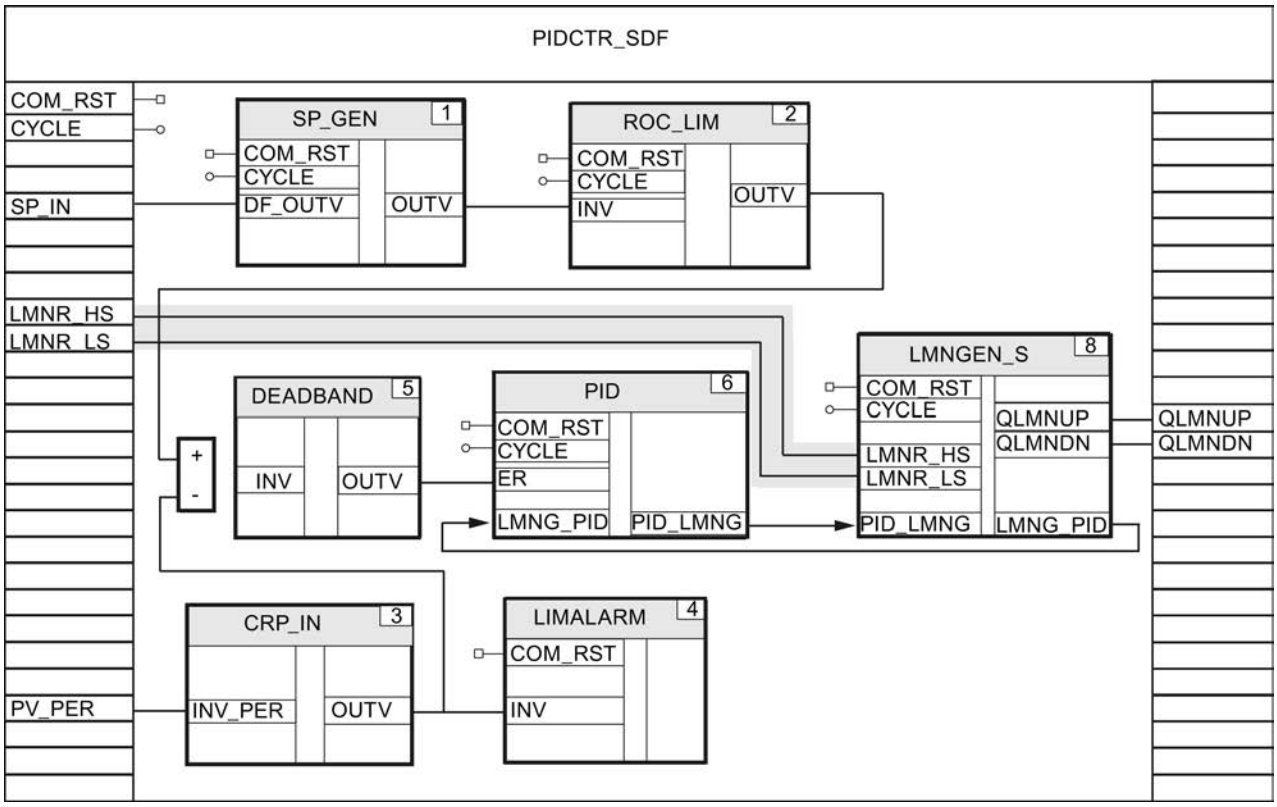
The diagram below shows how the Modular PID Control instructions are called and connected in the controller FB PIDCTR_SAF.



Creating a step controller with digital position feedback

1. At the controller FB, declare the input parameters LMNR_HS and LMNR_LS of the data type "Bool".
2. Connect the input parameters LMNR_HS and LMNR_LS to LMNGEN_S.LMNR_HS and LMNGEN_S.LMNR_LS.
3. Activate position feedback with LMNR_ON = TRUE in the LMNGEN_S_Instance structure.

The diagram below shows how the Modular PID Control instructions are called and connected in the controller FB PIDCTR_SDF.



See also

Creating a step controller (Page 9767)

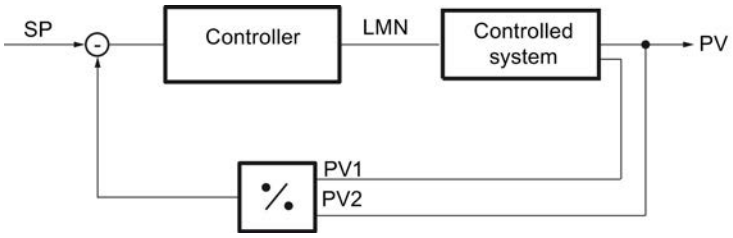
Description CRP_IN (Page 6948)

Description NONLIN (Page 6986)

Complex controller structures

Single-loop ratio controller

The diagram below shows a typical control loop of a single-loop ratio controller.



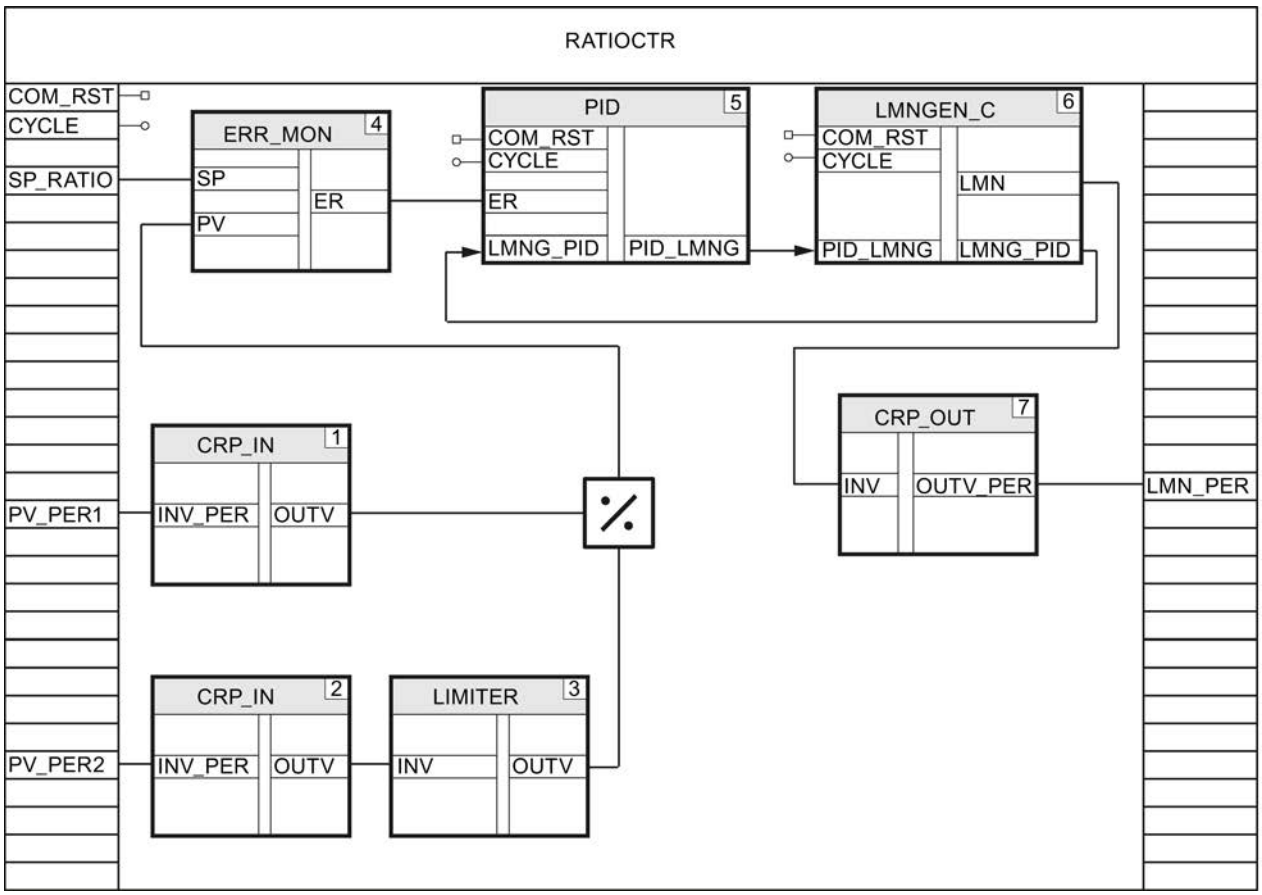
The ratio between two process values is adjusted to a defined setpoint with a PID algorithm.

Procedure

Proceed as follows to create a single-loop ratio controller:

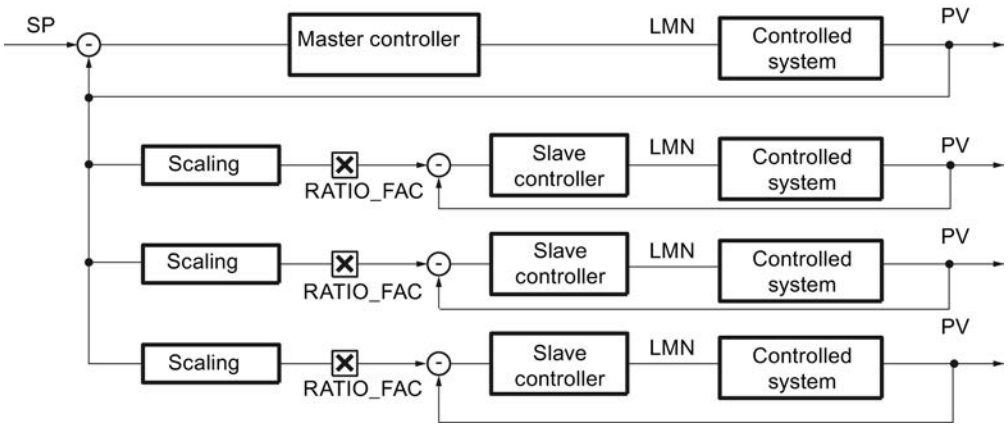
1. Program process value processing as outlined below in your controller FB.
2. To avoid division by zero, you must limit the value range for the process value PV_PER2 to Values > Zero with the instruction LIMITER.

The diagram below shows a controller FB, in this case RATIOCTR, for a single-loop ratio controller with a continual manipulated variable.



Multi-loop ratio controller

The diagram below shows a typical control loop for a multi-loop ratio controller.



Programming a control loop

Follow these steps to program this example with Modular PID Control:

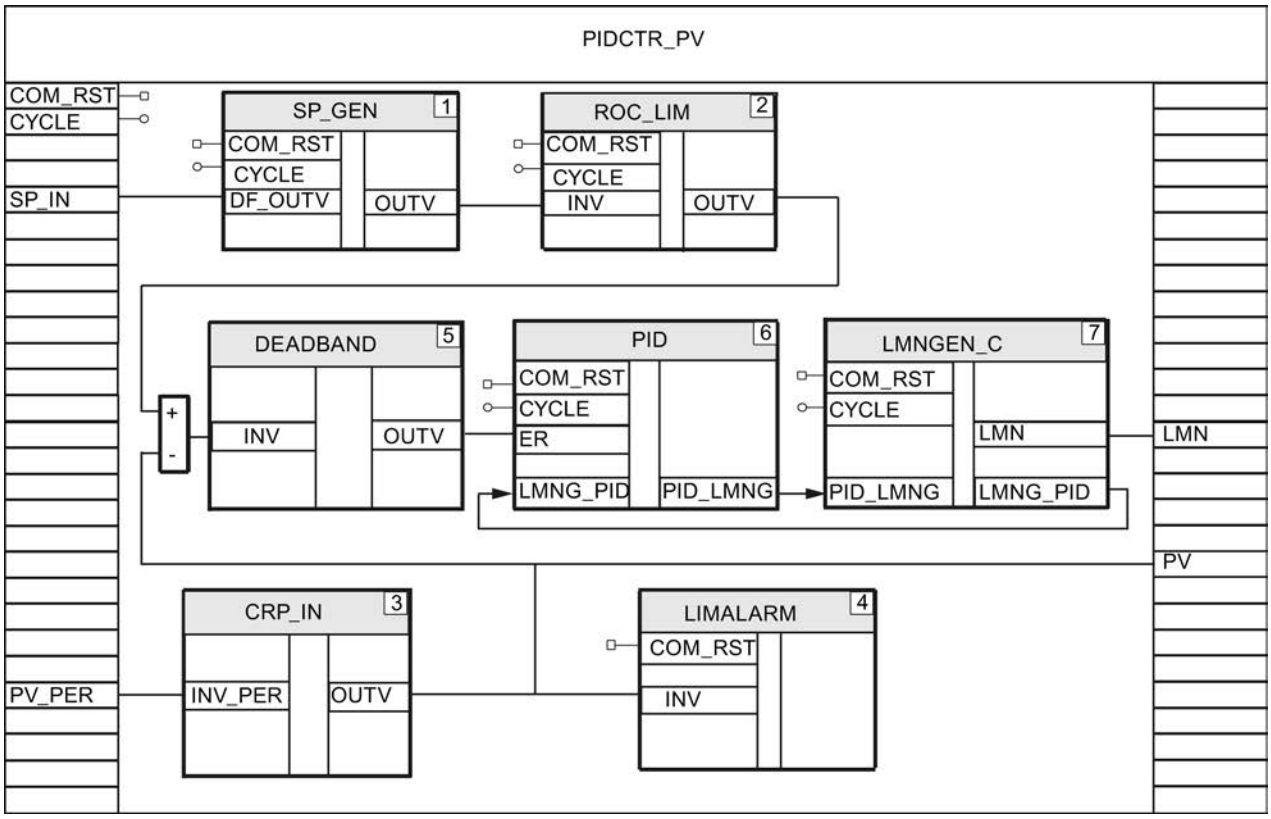
1. Program a controller FB for the master controller, e.g. PIDCTR_PV.
2. Program one or more controller FBs for the slave controllers, e.g. RB_CTR_S.
3. Program a function in which you can call and connect the four controllers.

Programming a master controller

Proceed as follows to program the master controller:

1. Program the master controller in line with your requirements as described in chapter Programming the controller FB (Page 9753).
2. At the controller FB, in this case PIDCTR_PV, declare an output parameter PV of the data type "Real" at which you output the processed process value of the master controller.

The diagram below shows the master controller interconnection, using the controller FB PIDCTR_PV as an example.

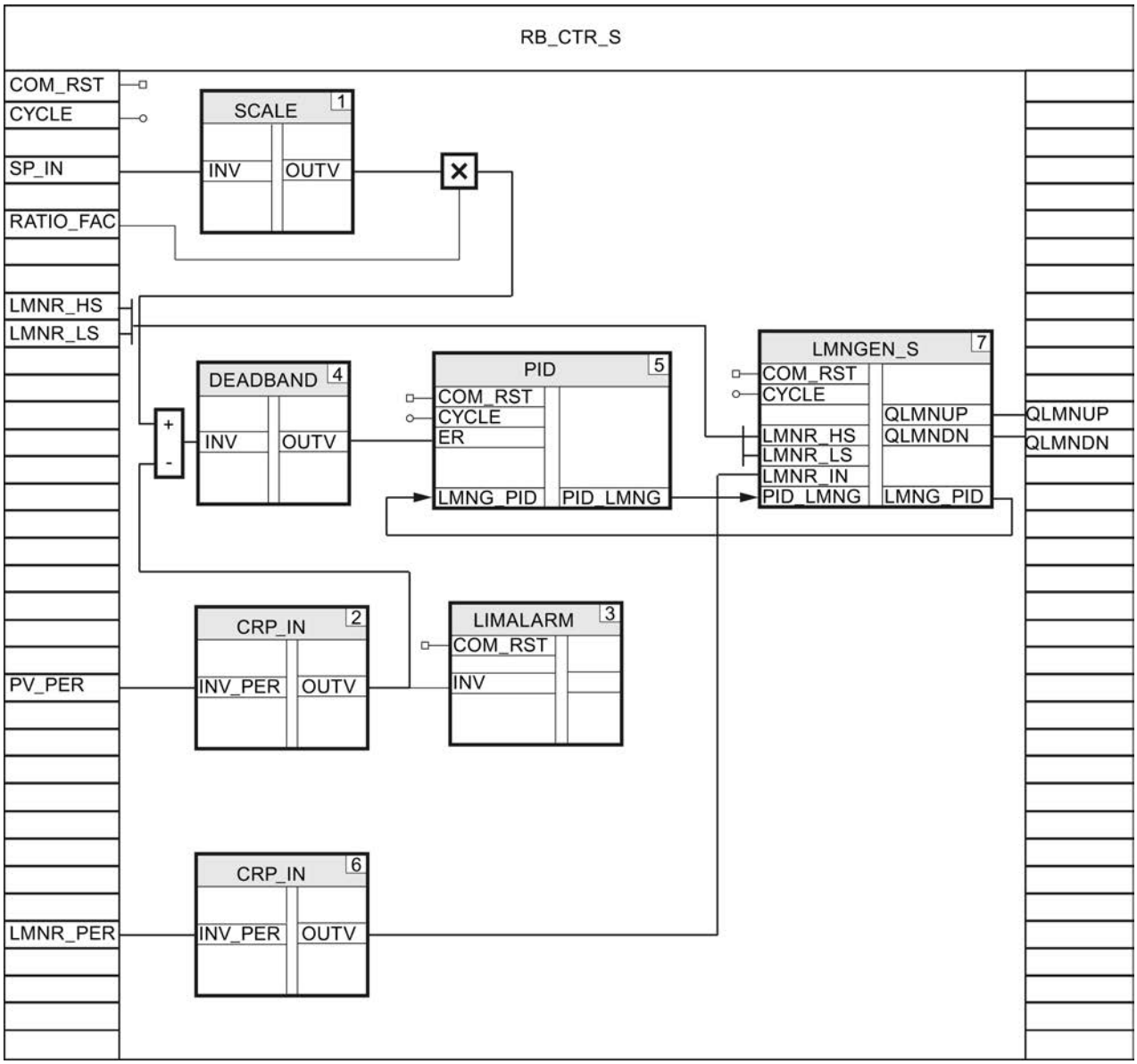


Programming slave controllers

Proceed as follows to program the slave controllers:

1. Program the slave controllers in line with your requirements as described in chapter Programming the controller FB (Page 9753).
2. At the controller FB, in this case RB_CTR_S, declare an input parameter RATIO_FAC of the data type "Int". The processed setpoint, e.g. SCALE.OUTV, of the slave controller is multiplied by this value.

The diagram below shows the slave controller interconnection, using the controller FB RB_CTR_S as an example.



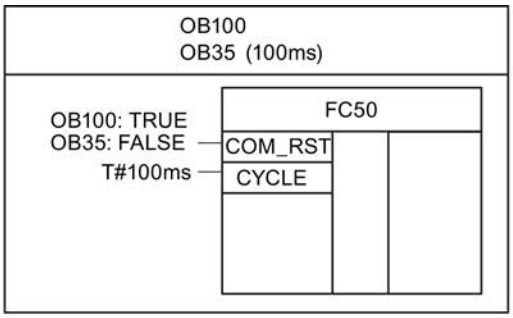
Calling and connecting controllers

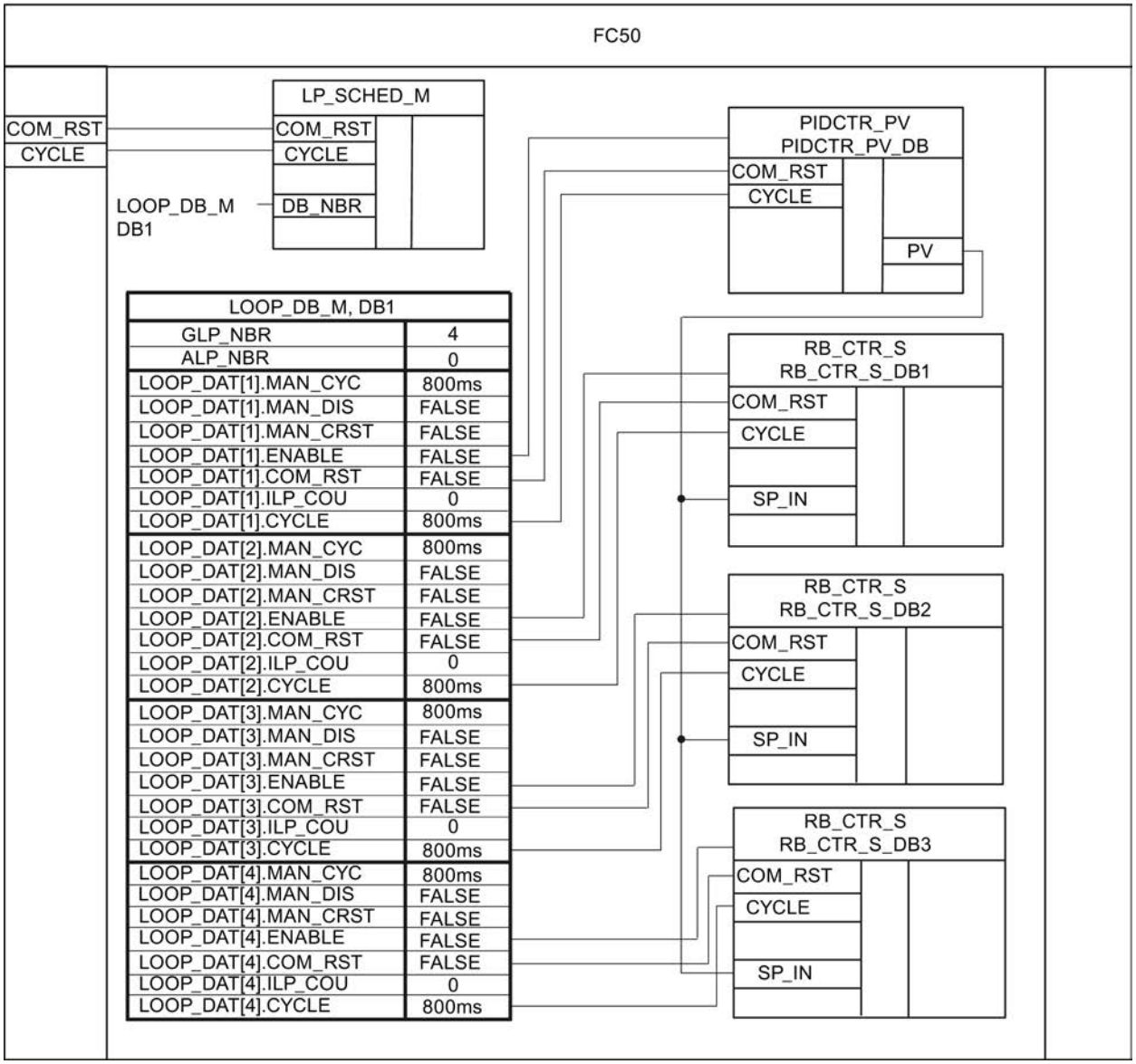
Proceed as follows to call and connect master and slave controllers:

1. Create a function, e.g. FC50,, with the input parameters COM_RST and CYCLE.
2. Call the LP_SCHED_M instruction in the function with the instance data block LP_SCHED_M_DB.
3. Create a global data block DB_LOOP_M in which to assign parameters for the number of controllers and the cycle times for calling the master controller and slave controller.
   The cycle times of the slave controllers must be shorter than or equal to the cycle time of the master controller.
4. Call the controller FB for the master controller, in this case PIDCTR_PV, if LOOP_DB_M.LOOP_DAT[1].ENABLE = TRUE.

5. Connect LOOP_DB_M.LOOP_DAT[1].COM_RST and LOOP_DB_M.LOOP_DAT[1].CYCLE to the input parameters COM_RST and CYCLE of the master controller.
6. Set LOOP_DB_M.LOOP_DAT[1].ENABLE = FALSE.
7. Call the controller FB for the first slave controller, in this case RB_CTR_S, if LOOP_DB_M.LOOP_DAT[2].ENABLE = TRUE.
8. Connect LOOP_DB_M.LOOP_DAT[2].COM_RST and LOOP_DB_M.LOOP_DAT[2].CYCLE to COM_RST and CYCLE of the slave controller.
9. Connect the output parameter PV of the master controller to the input parameter SP_IN of the slave controller.
10. Set LOOP_DB_M.LOOP_DAT[2].ENABLE = FALSE.
11. Repeat the last four steps for the two other slave controllers. Increase the array counter for each controller.
12. Connect the manipulated variables to the outputs of the CPU.
13. Call the function in OB 100 and in the cyclic interrupt OB.

The diagram below shows controller calls and interconnection, using the function FC50 as an example.

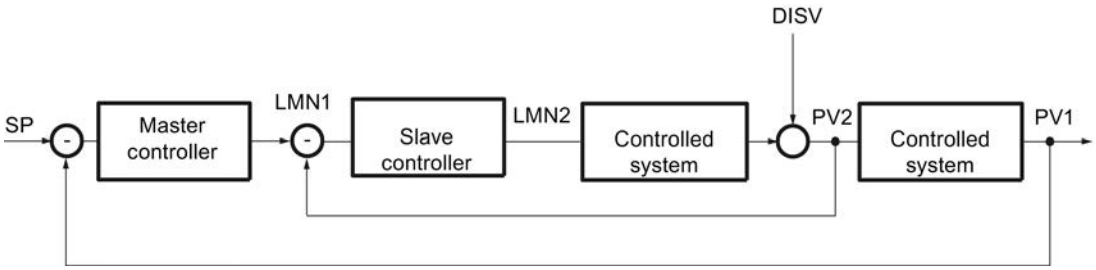




LOOP_DB_M, DB1	
GLP_NBR	4
ALP_NBR	0
LOOP_DAT[1].MAN_CYC	800ms
LOOP_DAT[1].MAN_DIS	FALSE
LOOP_DAT[1].MAN_CRST	FALSE
LOOP_DAT[1].ENABLE	FALSE
LOOP_DAT[1].COM_RST	FALSE
LOOP_DAT[1].ILP_COU	0
LOOP_DAT[1].CYCLE	800ms
LOOP_DAT[2].MAN_CYC	800ms
LOOP_DAT[2].MAN_DIS	FALSE
LOOP_DAT[2].MAN_CRST	FALSE
LOOP_DAT[2].ENABLE	FALSE
LOOP_DAT[2].COM_RST	FALSE
LOOP_DAT[2].ILP_COU	0
LOOP_DAT[2].CYCLE	800ms
LOOP_DAT[3].MAN_CYC	800ms
LOOP_DAT[3].MAN_DIS	FALSE
LOOP_DAT[3].MAN_CRST	FALSE
LOOP_DAT[3].ENABLE	FALSE
LOOP_DAT[3].COM_RST	FALSE
LOOP_DAT[3].ILP_COU	0
LOOP_DAT[3].CYCLE	800ms
LOOP_DAT[4].MAN_CYC	800ms
LOOP_DAT[4].MAN_DIS	FALSE
LOOP_DAT[4].MAN_CRST	FALSE
LOOP_DAT[4].ENABLE	FALSE
LOOP_DAT[4].COM_RST	FALSE
LOOP_DAT[4].ILP_COU	0
LOOP_DAT[4].CYCLE	800ms

Cascade controller

The diagram below shows a typical control loop for a cascade controller.

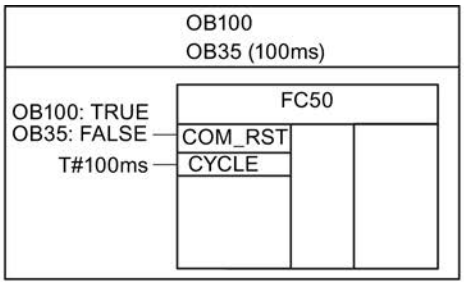


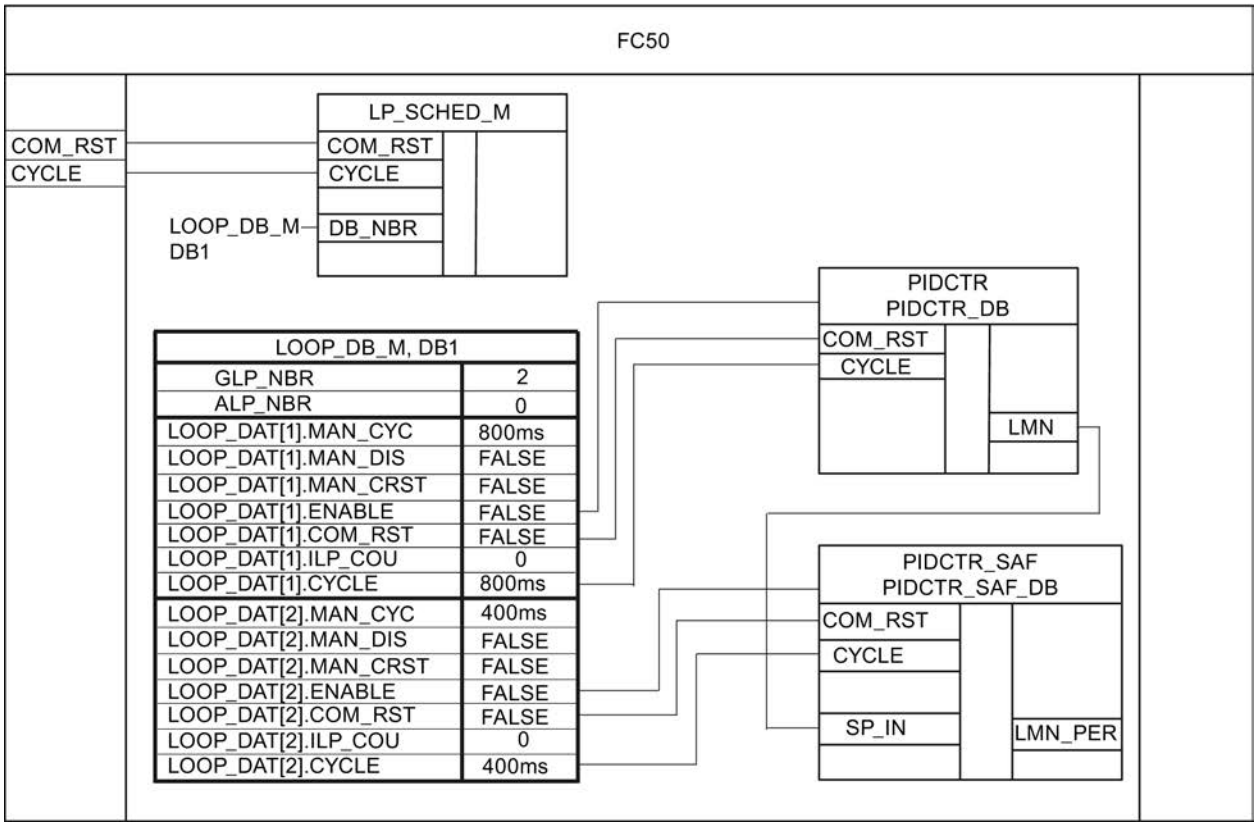
Programming a control loop

Follow these steps to program this example with Modular PID Control:

1. Program a controller FB for the master controller, e.g. PIDCTR.
2. Program a controller FB for the slave controller, e.g. PIDCTR_SAF.
3. Program a function in which you can call and connect the two controllers.
4. Call the LP_SCHED_M instruction in the function with the instance data block LP_SCHED_M_DB.
5. Create a global data block LOOP_DB_M in which to assign parameters for the number of controllers and the cycle times for calling the master controller and slave controller.
   The cycle time of the slave controller must be less than or equal to the cycle time of the master controller.
6. Call the controller FB for the master controller if LOOP_DB_M.LOOP_DAT[1].ENABLE = TRUE.
7. Connect LOOP_DB_M.LOOP_DAT[1].COM_RST and LOOP_DB_M.LOOP_DAT[1].CYCLE to the input parameters COM_RST and CYCLE of the master controller.
8. Set LOOP_DB_M.LOOP_DAT[1].ENABLE = FALSE.
9. Call the controller FB for the slave controller if LOOP_DB_M.LOOP_DAT[2].ENABLE = TRUE.
10. Connect LOOP_DB_M.LOOP_DAT[2].COM_RST and LOOP_DB_M.LOOP_DAT[2].CYCLE to COM_RST and CYCLE of the slave controller.
11. Connect the output parameter LMN of the master controller to the input parameter SP_IN of the slave controller.
12. Set LOOP_DB_M.LOOP_DAT[2].ENABLE = FALSE.
13. Connect the manipulated variable of the slave controller to the analog output of the CPU which controls the actuator.
14. Call the function in OB 100 and in the cyclic interrupt OB.

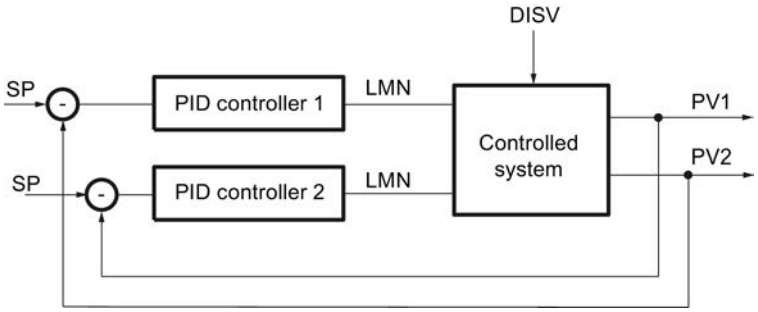
The diagram below shows controller calls and interconnection, using the function FC50 as an example.


OB100
OB35 (100ms)
FC50
OB100: TRUE
OB35: FALSE
T#100ms
COM_RST
CYCLE


FC50
COM_RST
CYCLE
LP_SCHED_M
COM_RST
CYCLE
LOOP_DB_M
DB1
DB_NBR
PIDCTR
PIDCTR_DB
COM_RST
CYCLE
LMN
LOOP_DB_M, DB1
GLP_NBR 2
ALP_NBR 0
LOOP_DAT[1].MAN_CYC 800ms
LOOP_DAT[1].MAN_DIS FALSE
LOOP_DAT[1].MAN_CRST FALSE
LOOP_DAT[1].ENABLE FALSE
LOOP_DAT[1].COM_RST FALSE
LOOP_DAT[1].ILP_COU 0
LOOP_DAT[1].CYCLE 800ms
LOOP_DAT[2].MAN_CYC 400ms
LOOP_DAT[2].MAN_DIS FALSE
LOOP_DAT[2].MAN_CRST FALSE
LOOP_DAT[2].ENABLE FALSE
LOOP_DAT[2].COM_RST FALSE
LOOP_DAT[2].ILP_COU 0
LOOP_DAT[2].CYCLE 400ms
PIDCTR_SAF
PIDCTR_SAF_DB
COM_RST
CYCLE
SP_IN
LMN_PER

Multi-variable controller

The diagram below shows a typical control loop for a multi-variable controller. Two PID controllers with continuous output are switched to the process.

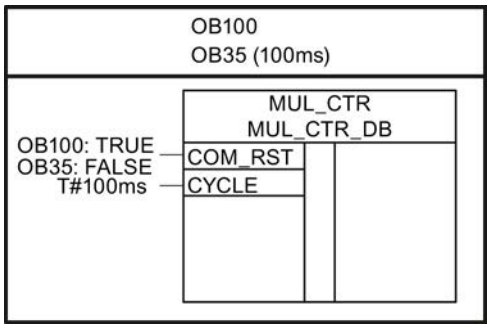


Programming a control loop

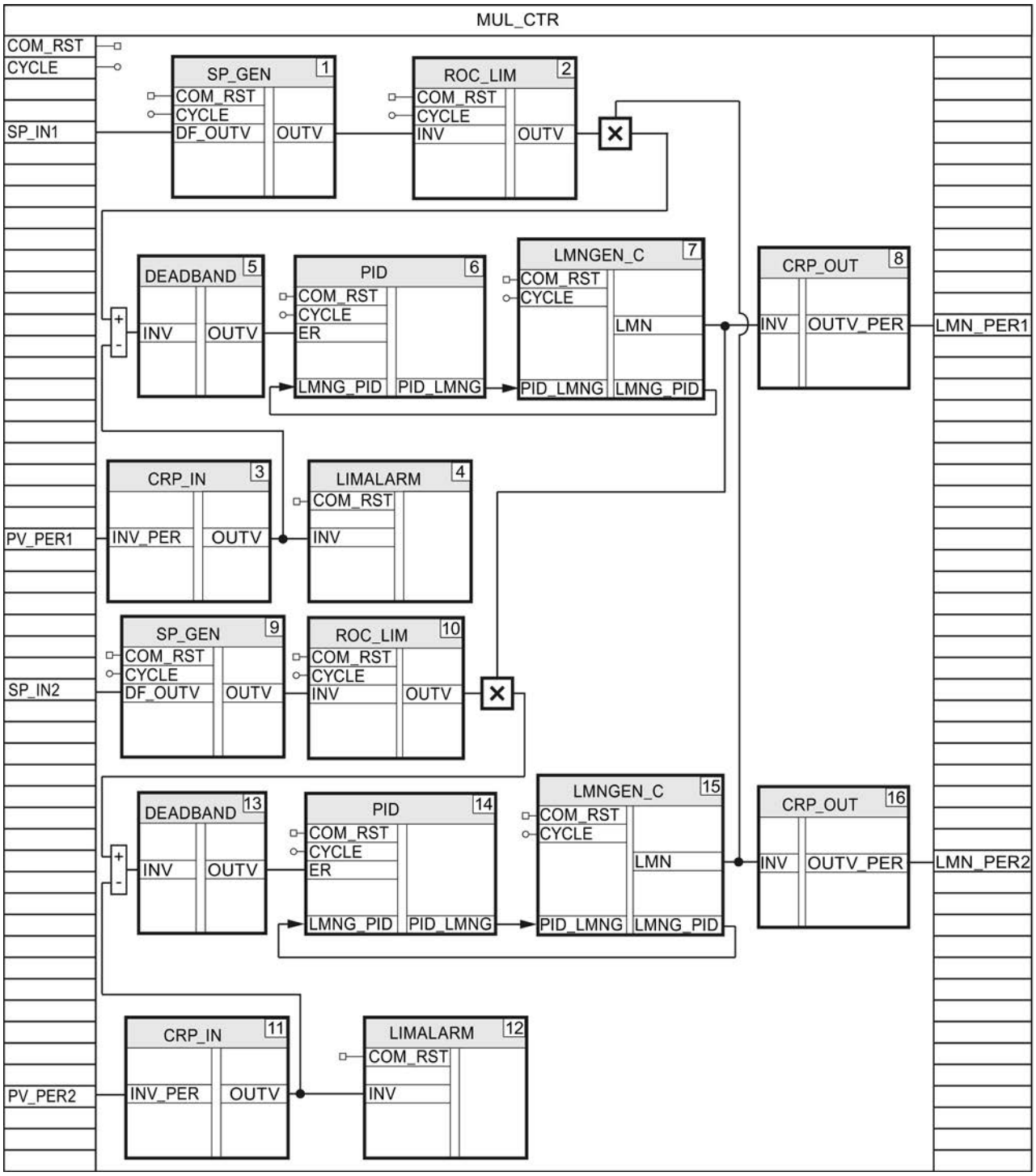
Follow these steps to program this example with Modular PID Control:

1. At the controller FB, in this case MUL_CTR, declare the following input parameters:
   - COM_RST of the data type "Bool"
   - CYCLE of the data type "Time"
   - SP_IN1 and SP_IN2of the data type "Real"
   - PV_PER1 and PV_PER2 of the data type "Word"
2. At the controller FB, declare the output parameters LMN_PER1 and LMN_PER2 of the data type "Word".
3. Program setpoint and process value processing and calculation of the manipulated variable for both PID controllers.
4. To calculate the control deviation of the first PID controller, subtract the processed process value PV_PER1 from the product of the setpoint SP_IN1 and the manipulated variable of the second PID controller.
5. To calculate the control deviation of the second PID controller, subtract the processed process value PV_PER2 from the product of the setpointSP_IN2 and the manipulated variable of the first PID controller.
6. Connect the manipulated variables LMN_PER1 and LMN_PER2 to two analog outputs which control the actuators.
7. Call the controller FB in OB 100 and in the cyclic interrupt OB.

The diagram below shows the call, using the controller FB MUL_CTR as an example.



The diagram below shows the controller interconnection, using the controller FB MUL_CTR as an example.


MUL_CTR
COM_RST
CYCLE
SP_IN1
SP_GEN 1
COM_RST
CYCLE
DF_OUTV
OUTV
ROC_LIM 2
COM_RST
CYCLE
INV
OUTV
DEADBAND 5
INV
OUTV
PID 6
COM_RST
CYCLE
ER
LMNG_PID
PID_LMNG
LMNGEN_C 7
COM_RST
CYCLE
LMN
PID_LMNG
LMNG_PID
CRP_OUT 8
INV
OUTV_PER
LMN_PER1
CRP_IN 3
INV_PER
OUTV
PV_PER1
LIMALARM 4
COM_RST
INV
SP_GEN 9
COM_RST
CYCLE
DF_OUTV
OUTV
SP_IN2
ROC_LIM 10
COM_RST
CYCLE
INV
OUTV
DEADBAND 13
INV
OUTV
PID 14
COM_RST
CYCLE
ER
LMNG_PID
PID_LMNG
LMNGEN_C 15
COM_RST
CYCLE
LMN
PID_LMNG
LMNG_PID
CRP_OUT 16
INV
OUTV_PER
LMN_PER2
CRP_IN 11
INV_PER
OUTV
PV_PER2
LIMALARM 12
COM_RST
INV

Commissioning the controller FB

Commissioning the continual controller

You can commission your controller FB with the PID Self-Tuner and determine the optimum PID parameters.

Procedure

Proceed as follows to commission a continual controller or a pulse controller:

1. Call the controller FB in a cyclic interrupt OB and connect the input and output parameters.
2. Call the instruction TUN_EC in the same cyclic interrupt OB.
3. Connect the instruction TUN_EC to your controller FB as follows:
   - TUN_EC.PV to the processed process value, e.g. the static variable CRP_IN_Instance_PV.OUTV or the output parameter PV
   - TUN_EC.LMN to the manipulated variable in real format, e.g. the static variable LMNGEN_C_Instance.LMN
   - TUN_EC.MAN to the in/out parameter for the manual value, e.g. MAN.
   - TUN_EC.SP_OUT to the input parameter for the setpoint, e.g. SP_IN
   - TUN_EC.GAIN to the proportional gain, e.g. PID_Instance.GAIN
   - TUN_EC.TI to the integral action time, e.g. PID_Instance.TI
   - TUN_EC.TD to the derivative action time, e.g. PID_Instance.TD
   - TUN_EC.TM_LAG to the time delay, e.g. PID_Instance.TM_LAG
4. Click the "" icon in the TUN_EC instruction.
   The commissioning editor for TUN_EC opens. The operation of the commissioning editor is explained under PID Self-Tuner.

Commissioning the step controller

You can commission your controller FB with the PID Self-Tuner and determine the optimum PID parameters.

Procedure

Follow the steps below to commission a step controller:

1. Call the controller FB in a cyclic interrupt OB and connect the input and output parameters.
2. Call the instruction TUN_ES in the same cyclic interrupt OB.

3. Connect the instruction TUN_ES to your controller FB as follows:
   - TUN_ES.PV to the processed process value, e.g. the static variable CRP_IN_Instance _PV.OUTV or the output parameter PV
   - TUN_ES.LMNR_IN to the processed position feedback, e.g. LMNGEN_S_instance.LMNR_IN
   - TUN_ES.C_LMNUP and TUN_ES.C_LMNDN to the manipulated variable signals, e.g. the output parameters QLMN_UP and QLMNDN
   - TUN_ES.LMNR_HS to the high endstop signal, e.g. LMNGEN_S_Instance.LMNR_HS
   - TUN_ES.LMNR_ON to the position feedback switch, e.g. LMNGEN_S_Instance.LMNR_ON
   - TUN_ES.SP_OUT to the input parameter for the setpoint, e.g. SP_IN
   - TUN_ES.GAIN to the proportional gain, e.g. PID_Instance.GAIN
   - TUN_ES.TI to the integral action time, e.g. PID_Instance.TI
   - TUN_ES.TD to the derivative action time, e.g. PID_Instance.TD
   - TUN_ES.TM_LAG to the time delay, e.g. PID_Instance.TM_LAG
   - TUN_ES.MTR_TM to the motor transition time, e.g. LMNGEN_S_Instance.MTR_TM
   - If you are using a deadband, TUN_ES.DEADB_W to the deadband width, e.g. DEADBAND_Instance.OUTV
   - TUN_ES.QLMNUP and TUN_ES.QLMNDN to the manipulated variable signals LMNGEN_S_Instance.LMNUP and LMNGEN_S_Instance.LMNDN
4. Click the "" icon in the TUN_ES instruction.
   The commissioning editor for TUN_ES opens. The operation of the commissioning editor is explained under PID Self-Tuner.

11.3.5.3 Using PID Self Tuner

Tuning continuous controllers and pulse controllers

Interconnecting TUN_EC with controller

The TUN_EC instruction is used to tune a continuous controller or a pulse controller. Depending on the controller, you interconnect TUN_EC with one of the following instructions:

Controller	Instruction
FM 355 C, FM 355 S pulse controller	PID_FM
FM 455 C, FM 455 S pulse controller	PID_FM
PID Basic Functions	CONT_C with or without PULSEGEN
Standard PID Control (PID Professional optional package)	PID_CP
Modular PID Control (PID Professional optional package)	PID LMNGEN_C

Interconnect the input and output parameters of the TUN_EC instruction with the instruction of the controller as follows.

TUN_EC	PID_FM	CONT_C	PID_CP
PV	PV	PV	PV
LMN	LMN	LMN	LMN
NORM_FAC	--	--	(NM_PVHR-NM_PVLR)*0.01
WRITE_DIS	OR operation from LOAD_OP and LOAD_PAR	--	--
MAN_OUT	LMN_RE	MAN	MAN
SP_OUT	SP_RE	SP_INT	SP_INT
GAIN	GAIN	GAIN	GAIN
TI	TI	TI	TI
TD	TD	TD	TD
TM_LAG	TM_LAG	TM_LAG	TM_LAG
RATIOFAC	--	--	RATIOFAC
PHASE	--	--	PHASE
QMAN_ON	LMN_REON	MAN_ON	MAN_ON
QI_SEL	--	I_SEL	I_SEL
QD_SEL	--	D_SEL	D_SEL
QWRITE	LOAD_PAR and LOAD_OP	--	--

See also

Mode of operation TUN_EC (Page 6845)

TUN_EC (Page 6842)

Tuning options

Pretuning

The PID parameters are tuned during the heating process. During this tuning, process identification with subsequent controller design is carried out.

You can request tuning with ADAPT1ST.

When you set ADAPT1ST = TRUE, then ADAPT_ON = TRUE is set, which forces pretuning to be performed. For pretuning, you specify the manipulated variable jump at the LHLM_TUN parameter.

The ADAPT1ST parameter is only required if serious adaptation errors resulted in process parameters that make the automatically calculated manipulated variable jump unsuitable for a re-adaptation.

When ADAPT1ST = TRUE , the initialization routine of the TUN_EC instruction is run. The following pre-assignments are made:

- The system switches to PI controller. The parameters of PI_CON are transferred to the controller for this purpose. If the PI_CON.GAIN and PI_CON.TI parameters both have the value 0.0, they are pre-assigned with PI_CON.GAIN = 1.5 and PI_CON.TI = 3600.0 s.
- PROCESS.GAIN = 999.0
- RATIOFAC = 1.0
- STATUS_H = 0; STATUS_D = 0; STATUS_C = 0

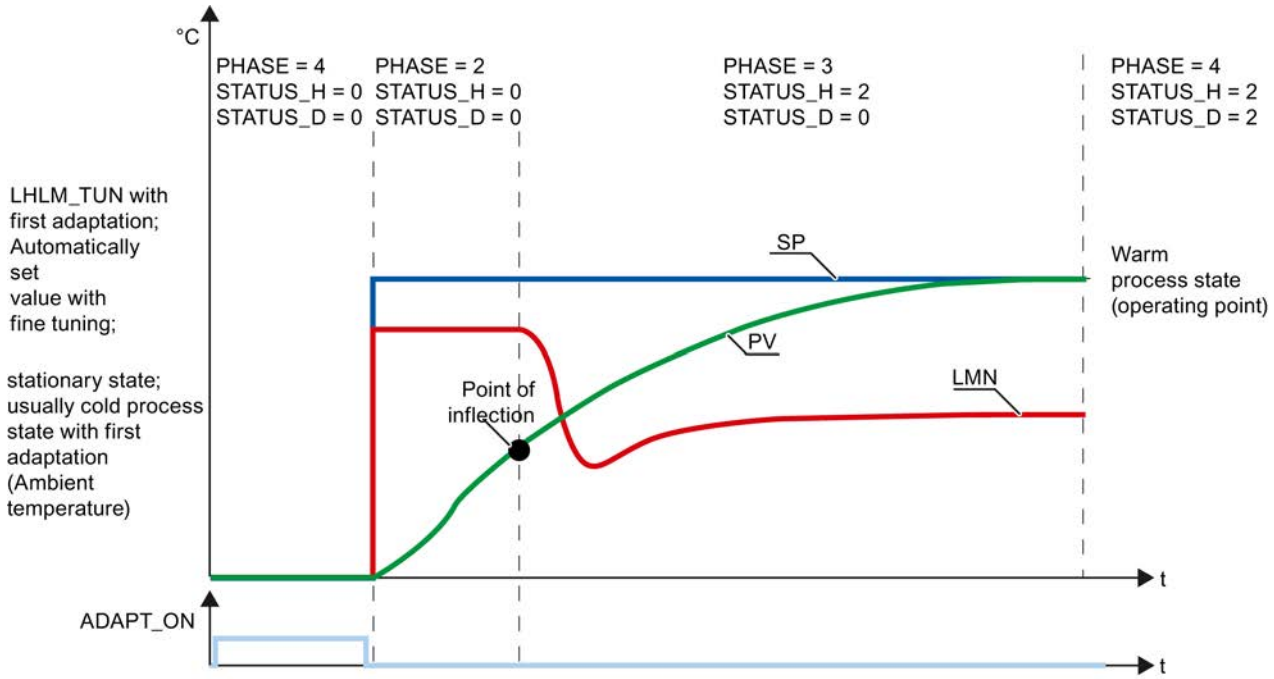
Note

Unless explicitly required, do **not** call TUN_EC during a warm restart of the CPU in its initialization routine as is usual for other instructions. If you do, all process parameters will be lost.

Tuning is started with ADAPT1ST = TRUE and a subsequent setpoint jump ≥ MIN_STEP in the positive direction.

As a rule, the setpoint jump changes from the setpoint of the cold process (ambient temperature) to a point close to the operating point during pretuning. Any setpoint can be used as the initial state as long as it is ensured that the process is in a steady state. You are not permitted to specify any further setpoint jumps during tuning.

Pretuning trend characteristics



Fine tuning

Fine tuning is possible only after pretuning has been carried out correctly. With ADAPT_ON = TRUE, TUN_EC decides whether it will perform pretuning or fine tuning. If information about the process is already available, TUN_EC automatically calculates a manipulated variable

jump and performs fine tuning. If information on the process is not available, pretuning is carried out.

Starting tuning

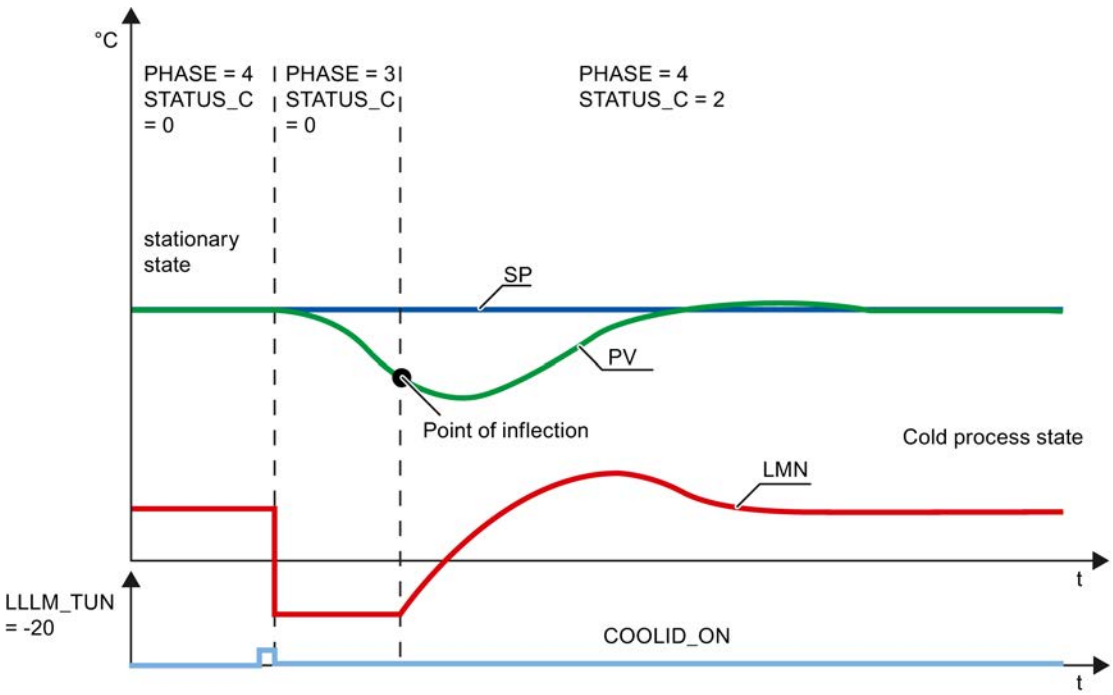
You can start tuning from either automatic mode or manual mode:

- **Starting tuning from automatic mode**
  In automatic mode, set ADAPT_ON or ADAPT1ST = TRUE and specify a setpoint jump ≥ MIN_STEP.
- **Starting tuning from manual mode**
  In manual mode, set ADAPT_ON or ADAPT1ST = TRUE and specify a setpoint jump ≥ MIN_STEP. You must then reset MAN_ON = FALSE in order for the setpoint jump to become effective and tuning to begin.

Cooling tuning

Cooling tuning is possible only after pretuning has been carried out correctly. It is started in PHASE 4. When COOLID_ON = TRUE is set, the controller is set to manual mode and LLLM_TUN is output: MAN_OUT = LLLM_TUN. TUN_EC switches to PHASE = 3, and COOLID_ON is reset again immediately. As soon as a point of inflection in the process value trend is found, TUN_EC switches back to closed-loop control (PHASE = 4). TUN_EC determines a ratio factor RATIOFAC, whose value is forwarded to the pulse generator of the controller.

Cooling tuning trend characteristics



Note

- The COOLID_ON start bit can be set at the same time as ADAPT1ST or ADAPT_ON, or during tuning of the controller parameters. This ensures that cooling tuning can be started immediately after tuning of the controller parameters for heating has been completed.
- In the case of a plant with several thermally coupled temperature zones (e.g., plastics machines), cooling tuning should not be started until all (!) zones have finished pretuning; otherwise, the pretuning results are invalid.

Canceling tuning

To cancel the tuning before the setpoint jump, you must set ADAPT_ON = FALSE . To cancel tuning after the setpoint jump (PHASE 2 or 3) or cancel cooling tuning, you must switch to manual mode (MAN_ON = TRUE).

By setting ADAPT_ON, ADAPT1ST or COOLID_ON during active tuning (PHASE 2 or 3), the tuning is not canceled but is instead completed. Because ADAPT_ON, ADAPT1ST or COOLID_ON is not reset, tuning is restarted once the start conditions are met.

Tuning phases

Individual phases are run through during controller tuning. You can read these phases at the PHASE parameter.

PHASE = 0

When an instance DB for TUN_EC is created, the PHASE parameter is preassigned with zero and MAN_ON with TRUE. For this reason, the system switches to manual mode (PHASE = 7) after the first cycle. You can start the pretuning directly from manual mode (ADAPT_ON = TRUE) or switch to automatic mode. When you switch to automatic mode MAN_ON = FALSE for the first time, the parameters of PI_CON are applied. The default values for GAIN and TI are: GAIN = 1.5, TI = 1h.

PHASE = 2

As soon as you specify a setpoint jump ≥ MIN_STEP in the positive direction toward the operating point of the warm process state, LHLM_TUN is assigned to MAN_OUT in the case of pretuning, while a value calculated by TUN_EC is assigned to MAN_OUT in the case of fine tuning. QMAN_ON = TRUE is also set. Both values are transferred to the PID controller. The PID controller is controlled with them in manual mode. MIN_STEP should be greater than 10% of the operating range of the setpoint and the process value.

PHASE = 3

When the point of inflection of the jump response is recognized (STATUS_H = 2) or when the process value has reached 70% of the setpoint jump height (STATUS_H = 3), a PI controller with conservative settings is designed. The controller gain GAIN is limited to the range from 0.2 to 20.0. As of this point, the controller immediately starts working as a PI controller and

attempts to bring the controlled system to a steady-state condition. TUN_EC stores the newly determined PI controller parameters in GAIN, TI, and PI_CON. The previous values of the PI controller are saved in PI_CON_OLD. If it takes a long time to reach steady-state condition (creeping transient response during temperature processes), you can set STEADY = TRUE to initiate controller design with the current data when steady-state condition is almost reached. You can also set STEADY = TRUE to re-initiate the controller design with the current data at a later stage in PHASE = 3 or PHASE = 4. This will often provide some additional improvement to the controller design.

If an overshoot occurs or if no point of inflection is found, the manipulated variable jump size LHLM_TUN selected for pretuning may have been too large. This does not necessarily result in poor controller tuning. For the next pretuning, the value you select for LHLM_TUN should be approximately 20% smaller.

When TUN_EC has recognized steady-state condition or when time 30 × TI (TI: integral action time of the PI controller set in PHASE = 3) has elapsed since the setpoint jump, the controller design is improved and the system switches to PHASE = 4. TUN_ES stores the newly determined PI/PID controller parameters in GAIN, TI, TD, and PI_CON or PID_CON. The previous values of the PID controller are saved in PID_CON_OLD. If possible, a PID controller and a PI controller are designed. If PID_ON = TRUE, the PID controller is activated. Otherwise, the PI controller is activated. GAIN is limited in such a way that the loop gain of the open loop (product of GAIN and process gain) is less than 80.0.

PHASE = 4

In this phase, the controller controls using its tuned parameters. From this state, you can start cooling tuning, fine tuning, or pretuning.

You can switch bumplessly between the PI controller and PID controller at any time.

See also

Mode of operation TUN_EC (Page 6845)

Performing pretuning

Requirements

- The instruction and the technology object are loaded on the CPU.
- TUN_EC is interconnected with a continuous controller or a pulse controller.
- MAN_ON = FALSE
- The controller is in automatic mode.
- The process is in steady-state condition.

Procedure

To determine the optimal PID parameters during initial commissioning, follow these steps:

1. Click the "Start" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value, and manipulated variable are recorded.
2. Select "Pretuning" from the "Mode" drop-down list.
   TUN_EC is ready for tuning.
3. Choose a controller structure.
4. In the "Manipulated variable" field, enter a manipulated variable to be used until the transition to PHASE = 3.
5. Enter a setpoint in the "Setpoint" field. The setpoint jump must be ≥ MIN_STEP. The new manipulated variable only takes effect when another setpoint is entered.
6. Click the "Start tuning" icon.
   Setpoint and manipulated variable are transferred to the controller. The pretuning starts. The status of the tuning is displayed.

See also

Mode of operation TUN_EC (Page 6845)

Performing fine tuning

Requirements

- The instruction and the technology object are loaded on the CPU.
- TUN_EC is interconnected with a continuous controller or a pulse controller.
- MAN_ON = FALSE
- The controller is in automatic mode.
- The process is in steady-state condition.

Procedure

To determine the optimal PID parameters at the operating point, follow these steps:

1. Click the "Start" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value, and manipulated variable are recorded.
2. Select "Fine tuning" from the "Mode" list.
   TUN_EC is ready for tuning.
3. Choose a controller structure.

4. Enter a setpoint in the "Setpoint" field. The setpoint jump must be ≥ MIN_STEP.
5. Click the "Start tuning" icon.
   The new setpoint is sent to the controller. TUN-EC calculates a manipulated variable jump. Fine tuning starts. The status of the tuning is displayed.

See also

Mode of operation TUN_EC (Page 6845)

Performing cold tuning

During cooling tuning, the ratio factor is determined for closed-loop controls with two opposing actuators (e.g. heating/cooling).

Requirements

- Pretuning has been carried out correctly.
- The instruction and the technology object are loaded on the CPU.
- TUN_EC is interconnected with a continuous controller or a pulse controller.
- MAN_ON = FALSE
- The controller is in automatic mode.
- The process is in steady-state condition.

Procedure

To determine the ratio factor, follow these steps:

1. Click the "Start" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value, and manipulated variable are recorded.
2. Select "Cooling tuning" from the "Mode" drop-down list.
   TUN_EC is ready for tuning.
3. Choose a controller structure.
4. In the "Manipulated variable" field, enter a manipulated variable that will be used until the point of inflection is found.
5. Click the "Start tuning" icon.
   Cooling tuning starts. The status of the tuning is displayed.

Performing tuning manually

You can manually change the PID parameters if oscillations occur or overshoots following setpoint jump changes are present in the closed control loop after pretuning or fine tuning. For example, you can reset the controller gain GAIN to 80% of the original value and the integral action time TI to 150% of the original setting. In a pulse controller, small continuous oscillations may occur, which you can eliminate by expanding the dead zone DEADB_W.

Requirements

- The instruction and the technology object are loaded on the CPU.

Procedure

To manually determine the optimal PID parameters, follow these steps:

1. Click the "Start" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value, and manipulated variable are recorded.
2. Select "Manual" from the "Mode" list.
3. Choose a controller structure.
4. Enter new PID parameters in the "P", "I" and "D" fields.
5. Click the "Send parameters to the controller" icon in the "Tuning" group.
6. Select the "Change setpoint" check box in the "Current values" group.
7. Enter a new setpoint and click the icon in the "Current values" group.
8. Clear the "Manual mode" check box, if necessary.
   The controller works with the new PID parameters and controls to the new setpoint.
9. Check the quality of the PID parameters based on the trend characteristics.
10. Repeat steps 3 to 9 until you are satisfied with the controller results.

See also

Mode of operation TUN_EC (Page 6845)

Tuning step controllers

Interconnecting TUN_ES with controller

The TUN_ES instruction is used to tune a step controller. Depending on the controller, you interconnect TUN_ES with one of the following instructions:

Controller	Instruction
FM 355 S step controller	PID_FM
FM 455 S step controller	PID_FM
PID Basic Functions	CONT_S TCONT_S
Standard PID Control (optional package)	PID_ES
Modular PID Control (optional package)	PID LMNGEN_S

Interconnect the input and output parameters of the TUN_ES instruction with the instruction of the controller.

TUN_ES	PID_FM	CONT_S	TCONT_S	PID_ES
PV	PV	PV	PV	PV
LMNR	LMN_A	--	--	LMNR_IN
NORM_FAC	--	--	--	(NM_PVHR-NM_PVLR)*0.01
C_LMNUP	QLMNUP	QLMNUP	QLMNUP	QLMNUP
C_LMNDN	QLMNDN	QLMNDN	QLMNDN	QLMNDN
LMNR_HS	QLMNR_HS	LMNR_HS	LMNR_HS	LMNR_HS
LMNR_ON	QLMNR_ON	--	--	LMNR_ON
WRITE_DIS	OR operation from LOAD_OP and LOAD_PAR	--	--	--
MAN_OUT	LMN_RE	--	--	MAN
SP_OUT	SP_RE	SP_INT	SP_INT	SP_INT or SP_EXT
GAIN	GAIN	GAIN	GAIN	GAIN
TI	TI	TI	TI	TI
TD	TD	--	--	TD
TM_LAG	TM_LAG	--	--	TM_LAG
MTR_TM	MTR_TM	MTR_TM	MTR_TM	MTR_TM
DEADB_W	DEADB_W	DEADB_W	DEADB_W	DEADB_W
PHASE	--	--	--	PHASE
QMAN_ON	LMN_REON	--	--	MAN_ON
QLMNS_ON	LMNSOPON	LMNS_ON	LMNS_ON	LMNS_ON
QLMNUP	LMNUP_OP	LMNUP	LMNUP	LMNUP
QLMNDN	LMNDN_OP	LMNDN	LMNDN	LMNDN
QI_SEL	--	--	--	I_SEL
QD_SEL	--	--	--	D_SEL
QWRITE	LOAD_PAR and LOAD_OP	--	--	--

See also

Mode of operation TUN_ES (Page 6864)

TUN_ES (Page 6861)

Tuning options

Pretuning

The PID parameters are tuned during the heating process. During this tuning, process identification with subsequent controller design is carried out.

You can request pretuning with ADAPT1ST.

If you set ADAPT1ST = TRUE, then ADAPT_ON = TRUE and the valve are closed (PHASE = 1).

When ADAPT1ST = TRUE , the initialization routine of the TUN_ES instruction is run. The following pre-assignments are made:

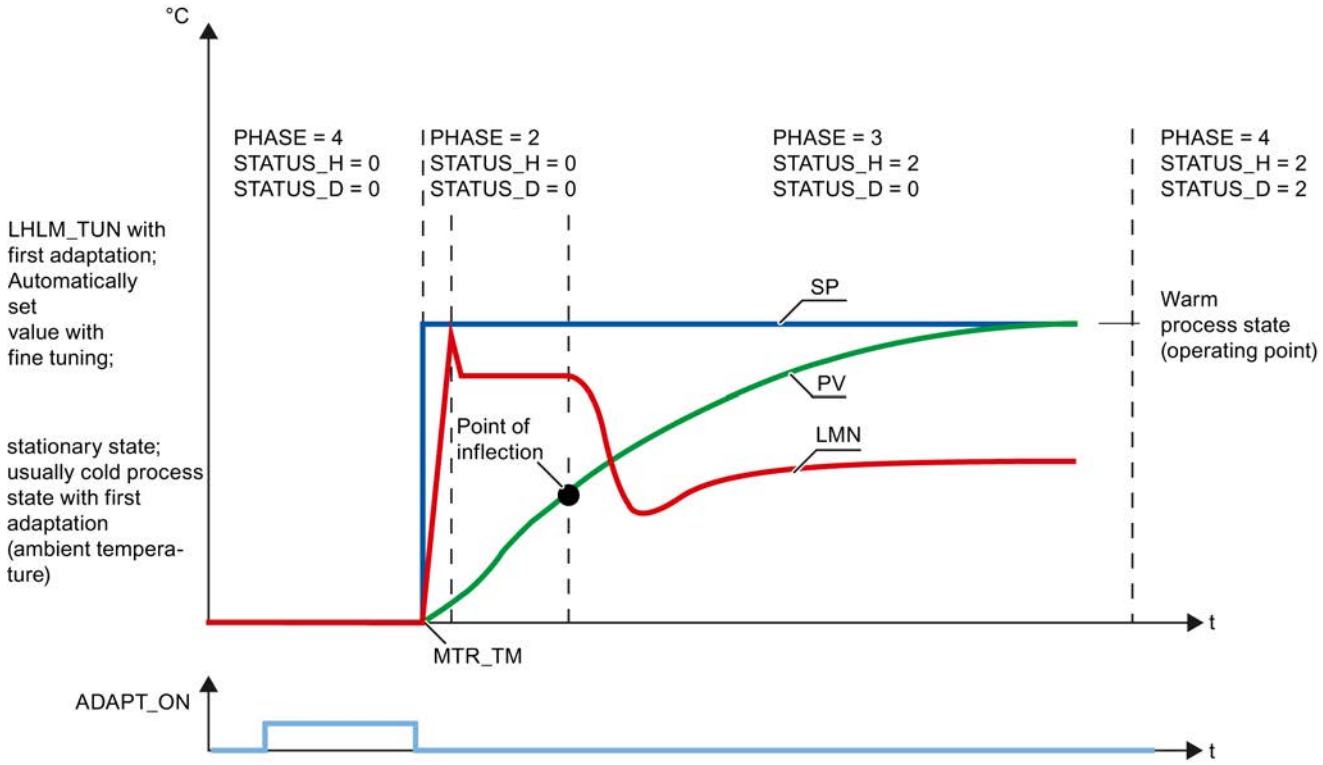
- The system switches to PI controller. The parameters of PI_CON are transferred to the controller for this purpose. If the PI_CON.GAIN and PI_CON.TI parameters both have the value 0.0, they are pre-assigned with PI_CON.GAIN = 1.5 and PI_CON.TI = 3600.0 s.
- PROCESS.GAIN = 999.0
- STATUS_H = 0; STATUS_D = 0

Note

Unless explicitly required, do **not** call TUN_ES during a warm restart of the CPU in its initialization routine as is usual for other instructions. If you do, all process parameters will be lost.

After a setpoint jump ≥ MIN_STEP, the motor transition time is measured after which pretuning is carried out. For process excitation during pretuning, TUN_ES uses the LHLM_TUN parameter for the manipulated variable jump. In the case of step controllers with position feedback, the value LHLM_TUN is approached directly. In the case of step controllers without position feedback, the control valve is opened completely and then moved back toward the closed position until the fraction of the motor transition time specified by (100 - LHLM_TUN) has elapsed.

Pretuning with measurement of motor transition time for step controller without position feedback



Fine tuning

You can use ADAPT_ON to start the tuning only for step controllers with position feedback. TUN_ES uses old process values to determine whether pretuning without measurement of the motor transition time or fine tuning will be performed. Tuning is started with ADAPT_ON = TRUE and a subsequent setpoint jump ≥ MIN_STEP in the positive direction. TUN_ES then immediately enters PHASE = 2 and performs a manipulated variable jump to LHLM_TUN in the case of pretuning or to an automatically calculated manipulated variable in the case of fine tuning.

As a rule, the setpoint jump changes from the setpoint of the cold process (ambient temperature) to a point close to the operating point during pretuning. Any setpoint can be used as the initial state as long as it is ensured that it is a steady state. You are not permitted to specify any further setpoint jumps during tuning.

Starting tuning

You can start tuning from either automatic mode or manual mode:

- Starting tuning from automatic mode
  In automatic mode, set ADAPT_ON or ADAPT1ST = TRUE and specify a setpoint jump ≥ MIN_STEP.
- Starting tuning from manual mode
  In manual mode, set ADAPT_ON or ADAPT1ST = TRUE and specify a setpoint jump ≥ MIN_STEP. You must then reset MAN_ON = FALSE in order for the setpoint jump to become effective and tuning to begin.

Canceling tuning

To cancel the tuning before the setpoint jump, you must set ADAPT_ON = FALSE . To cancel the tuning after the setpoint jump (PHASE 2 or 3), you must switch to manual mode (MAN_ON = TRUE or LMNS_ON = TRUE).

By setting ADAPT_ON or ADAPT1ST during active tuning (PHASE 2 or 3), the tuning is not canceled but is instead completed. Because ADAPT_ON or ADAPT1ST is not reset, tuning is restarted once the start conditions are met.

Tuning phases

Individual phases are run through during controller tuning. You can read these phases at the PHASE parameter.

PHASE = 0

When an instance DB for TUN_ES is created, the PHASE parameter is preassigned with zero and LMNS_ON with TRUE. For this reason, the system switches to manual mode (PHASE = 7) after the first cycle. You can start the pretuning directly from manual mode (ADAPT_ON = TRUE) or switch to automatic mode. When you switch to automatic mode MAN_ON = FALSE for the first time, the parameters of PI_CON are applied. The default values for GAIN and TI are: GAIN = 1.5, TI = 1h.

PHASE = 1

After tuning has been started with ADAPT1ST = TRUE, TUN_ES closes the control valve (manipulated variable = 0 or LMNS_ON = LMNDN = TRUE) and waits for a positive setpoint jump ≥ MIN_STEP.

PHASE = 2

As soon as you specify a setpoint jump ≥ MIN_STEP in the positive direction toward the operating point of the warm process state, LHLM_TUN is assigned to MAN_OUT in the case of pretuning and a value calculated by TUN_ES is assigned to MAN_OUT in the case of fine tuning. MIN_STEP should be greater than 10% of the operating range of the setpoint and process value.

In the case of step control with position feedback (LMNR_ON=TRUE), the value TRUE is assigned to QMAN_ON. The controller controls the valve to the value of MAN_OUT, and TUN_ES calculates the motor transition time MTR_TM.

In the case of step control without position feedback, QLMNS_ON and QLMNUP are set to TRUE, and the valve is moved to the high endstop position. When the high endstop is reached (LMNR_HS =TRUE), TUN_ES calculates the motor transition time and forwards it to the controller. The valve is then moved toward the closed position until the value of MAN_OUT is reached (QLMNDN = TRUE).

PHASE = 3

When the point of inflection of the jump response is recognized (STATUS_H = 2) or when the process value has reached 70% of the setpoint jump height (STATUS_H = 3), a PI controller with conservative settings is designed. The controller gain GAIN is limited to the range from 0.2 to 20.0. As of this point, the controller immediately starts working as a PI controller and attempts to bring the controlled system to a steady-state condition. TUN_ES stores the newly determined PI controller parameters in GAIN, TI, and PI_CON. The previous values of the PI controller are saved in PI_CON_OLD. If it takes a long time to reach steady-state condition (creeping transient response during temperature processes), you can set STEADY = TRUE to initiate controller design with the current data when steady-state condition is almost reached. You can also set STEADY = TRUE to re-initiate the controller design with the current data at a later stage in PHASE = 3 or PHASE = 4. This will often provide some additional improvement to the controller design.

If an overshoot occurs or if no point of inflection is found, the manipulated variable jump size LHLM_TUN selected for pretuning may have been too large. This does not necessarily result in poor controller tuning. For the next pretuning, the value you select for LHLM_TUN should be approximately 20% smaller.

When TUN_ES has recognized steady-state condition or when time 30 × TI (TI: integral action time of the PI controller set in PHASE = 3) has elapsed since the setpoint jump, the controller design is improved and the system switches to PHASE = 4. TUN_ES stores the newly determined PI/PID controller parameters in GAIN, TI, TD, and PI_CON or PID_CON. The previous values of the PID controller are saved in PID_CON_OLD. If possible, a PID controller and a PI controller are designed. If PID_ON = TRUE, the PID controller is activated, otherwise the PI controller is activated. For complicated processes, TUN_ES always designs a PI controller, irrespective of PID_ON. GAIN is limited in such a way that the loop gain of the open loop (product of GAIN and process gain) is less than 40.0.

PHASE = 4

In this phase, the controller controls using its tuned parameters. From this state, you can start a fine tuning or a pretuning.

You can switch bumplessly between the PI controller and PID controller at any time.

See also

Scope of TUN_ES (Page 6862)

Mode of operation TUN_ES (Page 6864)

Performing pretuning

The controller parameters are pretuned during the heating process. During this tuning, process identification with subsequent controller design is carried out.

Requirements

- The instruction and the technology object are loaded on the CPU.
- TUN_ES is interconnected with a step controller.
- MAN_ON = FALSE
- The controller is in automatic mode.
- The process is in steady-state condition.

Procedure

To determine the optimal PID parameters during initial commissioning, follow these steps:

1. Click the "Start" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value, and manipulated variable are recorded.
2. Select "Pretuning" from the "Mode" drop-down list.
   TUN_ES is ready for tuning.
3. Choose a controller structure.
4. In the "Manipulated variable" field, enter a manipulated variable to be used until the transition to PHASE = 3.
5. Enter a new setpoint. The setpoint jump must be ≥ MIN_STEP.
6. Click the "Start tuning" icon.
   The pretuning starts. The status of the tuning is displayed.

See also

Mode of operation TUN_ES (Page 6864)

Performing fine tuning

For step controllers with position feedback, you can start fine tuning.
The controller can be fine tuned online during a positive setpoint jump when there is a change from one operating point to another or when there are changes in the process response.

Requirements

- The instruction and the technology object are loaded on the CPU.
- TUN_ES is interconnected with a step controller.
- MAN_ON = FALSE
- LMNR_ON = TRUE for the TUN_ES instruction and the controller.
- The controller is in automatic mode.
- The process is in steady-state condition.

Procedure

To determine the optimal PID parameters during initial commissioning, follow these steps:

1. Click the "Start" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value, and manipulated variable are recorded.
2. Select "Fine tuning" from the "Mode" list.
   TUN_ES is ready for tuning.
3. Choose a controller structure.
4. Enter a new setpoint. The setpoint jump must be ≥ MIN_STEP.
5. Click the "Start tuning" icon.
   Fine tuning starts. The manipulated variable is determined by TUN_ES. The status of the tuning is displayed.

Performing tuning manually

You can manually change the PID parameters if oscillations occur or overshoots following setpoint jump changes are present in the closed control loop after pretuning or fine tuning. For example, you can reset the controller gain GAIN to 80% of the original value and the integral action time TI to 150% of the original setting.

Requirements

- The instruction and the technology object are loaded on the CPU.
- TUN_ES is interconnected with a step controller.
- MAN_ON = FALSE
- The controller is in automatic mode.
- The process is in steady-state condition.

Procedure

To manually determine the optimal PID parameters, follow these steps:

1. Click the "Start" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value, and manipulated variable are recorded.
2. Select "Manual" from the "Mode" list.
3. Choose a controller structure.
4. Enter new PID parameters in the "P", "I" and "D" fields.
5. Click the "Send parameters to the controller" icon in the "Tuning" group.
6. Select the "Change setpoint" check box in the "Current values" group.
7. Enter a new setpoint and click the icon in the "Current values" group.
8. Clear the "Manual mode" check box, if necessary.
   The controller works with the new PID parameters and controls to the new setpoint.
9. Check the quality of the PID parameters based on the trend characteristics.
10. Repeat steps 3 to 9 until you are satisfied with the controller results.

See also

Mode of operation TUN_ES (Page 6864)

11.3.5.4 Basics on controller modules

Inputs

Introduction

You can connect different types of sensors to the analog inputs. The sensor input signals are then processed to suit your requirements.

The digital inputs can be used to set different operating modes for the module.

The structure of analog and digital inputs is identical on continuous and step controllers.

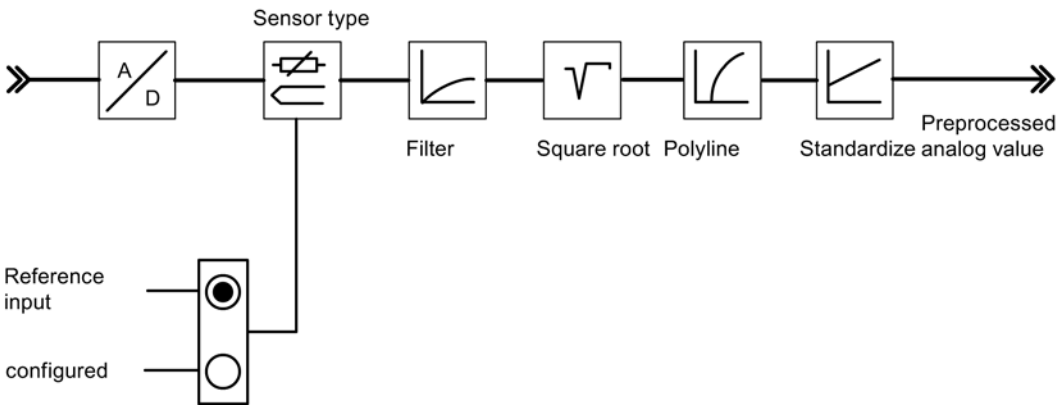
Number of inputs

Input	FM 355	FM 355-2	FM 455
Analog inputs for analog value processing	4	4	16
Reference junction input for the compensation of thermocouples	1	1	1
Digital inputs	8	8	16

Analog inputs

Function blocks of an analog input

The following figure shows how analog values are prepared:



Adaptation to sensors

You can set up the analog input parameters to suit different sensors. The following settings are available:

- Analog input is not processed (e.g., unused input)
  With this setting, all PID parameters are set to zero, the controller channel is not processed, and a manipulated variable is not calculated.
- Current sensor, 0 mA to 20 mA
- Current sensor, 4 mA to 20 mA
- Voltage sensor, 0 V to 10 V
- Pt 100, –200 °C to +850 °C
- Pt 100, –200 °C to +556 ºC (double resolution)
- Pt 100, –200 °C to +130 ºC (quadruple resolution)
- Thermocouples type B, J, K, R, and S (analog input set to ±80 mV)
- Free thermocouple (analog input set to ±80 mV)

You assign the parameters for analog inputs in the "Analog input" screen form.

Toggling between Celsius / Fahrenheit

Temperatures can be measured in °C or °F units.

Adaptation to the line frequency

Input signal processing can be adapted to the line frequency in order to suppress interference during the measurement of analog signals. The following settings are available:

- 50 Hz mode
- 60 Hz mode

Reference junction

If you set up a thermocouple as sensor on an analog input, you can connect a Pt 100 to the reference junction input of the module in order to compensate for the reference junction temperature at the thermocouple elements. You can also configure a static reference junction temperature.

Alternatively, with the FM 355, you can assign parameters for an internal compensation for thermocouples J, K and E. An internal temperature sensor measures the reference junction temperature directly in the module.

When using the reference junction input, the sampling time of each controller extends by the conversion time for the reference junction input.

Analog value processing

The analog value processing offers various assignable options for preparing input signals. The following table offers an overview of these parameters and their assignable values.

Parameter	Assignable values	Note
Resolution	• 12 bits • 14 bits	Conversion time 20 ms (50 Hz) Conversion time $16^2/_3$ ms (60 Hz) Conversion time 100 ms
Filters	• ON / OFF • Time constant in s	Filter of the first order whose time response is defined by the time constant.
Square root	• ON / OFF	For the linearization of sensor signals, where the process value is available as physical quantity with quadratic relation with the measured process variable.
Polyline	• ON / OFF • 13 interpolation points can be selected in – mA with current input – mV with voltage input	To linearize encoder characteristics
Standardization	• High value • Low value	For conversion of the input signal to a different unit of measurement by means of linear interpolation between the start value (low) and end value (high)

Note

Standardization/polyline: The conversion of the unit mA or mV into a unit of measurement takes place either via the polyline or - if this is not enabled - via scaling. The polyline is used for linearization of a free thermocouple, or for any other linearization.

Copying parameters

Using the analog input of the FMs, you can transfer the parameters of one input to one or more other inputs. In doing so, all assigned parameters of the output channel are transferred to the destination channels.

Digital Inputs

Operating modes

The digital inputs are used to change the operating mode of individual controller channels.

The direction of control action of the digital inputs can be configured. The following settings are available for the digital inputs:

- Active high
- Active low, or open circuit

You can select the following operating modes:

- Changeover to manipulated value setpoint by calling FB PID_FM
- Changeover to following mode (manipulated value setpoint via analog input)
- Changeover to safety manipulated value

For the step controller, you can also define the following signals via digital inputs:

- Feedback: Actuating device at high limit stop
- Feedback: Actuating device at low limit stop

Parallel tuning

The "Parallel tuning" function is only available for FM 355-2.

Adjacent zones (heavy thermal coupling)

If using two or several temperature controls of a module to control the temperature, for example, of a plate (e.g. two heaters and two measured process values with heavy thermal coupling), you can tune these in parallel. Proceed as follows:

1. Define the channel group in zone A or zone B.
2. Start tuning in one of these channels. Tuning of the other channels will then be started automatically.

Note

However, for excitation by means of setpoint step, you have to ensure that the setpoint steps are set simultaneously. When using the wizard, you can specify the PID_ON and TUN_DLMN/ TUN_CLMN parameters only for one channel. Specify the parameters for the other channels by means of the corresponding instance DBs of FB FMT_PID before you start tuning.

Cancelling tuning

Tuning of all channels is cancelled if an error occurs in one of the channels (phase 0, or end of phase 1).

The user can reset tuning by setting TUN_ON=FALSE at any one of the participating channels.

When a channel goes into safety mode, tuning of all other participating channels is also disabled.

The wizard displays only the status information of the selected channel. For all other channels, the causes of error are found in the corresponding instance DBs.

Advantage

Each participating controller will output its tuning manipulated variable until all controllers have exited phase 2. This prevents the controller that completes tuning first from corrupting the tuning results of the other controllers due to the change of its manipulated variable.

NOTICE
When 75% of the setpoint step is reached at one of the participating channels, tuning is not terminated (risk of overshoot). Automatic mode is not started unless all participating controllers have completed tuning.

Adjacent zones (weak thermal coupling)

As a general rule, tuning should be carried out to reflect the way in which the control will later work. When zones are operated in parallel during production so that temperature differences between the zones remain the same, the temperature level of the adjacent zones should also be raised accordingly during tuning.

Temperature differences at the start of testing are irrelevant since they will be compensated for in the initial heating process (initial rise = 0).

Error signal

Copying channels

The FMs support the transfer of a controller channel to one or several other channels. All output channel settings are transferred to the target channels.

Controller types

The error signal is derived from the same structure in all controller modules.

1. The signal for the actual value is conditioned to the effective actual value.
2. The setpoint signal is conditioned to the effective setpoint.
3. The effective setpoint is deducted from the effective actual value.
4. The error signal is transferred to the control algorithm.

You can select the signal sources for the setpoint and actual value and use the controller modules as:

- Fixed setpoint and cascade controllers
- Three-component controllers
- Ratio/blending controllers

Fixed setpoint and cascade controllers

The setpoint value of a master controller is set as the manipulated value of a slave controller in a cascade control.

Three-component controllers

Three-component controllers have three actual value inputs. For a blending control, the actual values A, B, and C are added to produce the total quantity PV.

- Blending controls
  A blending control is always slave controller. The associated master controller is a three-component controller.
  The manipulated variable of the master controller is interconnectd via the actual value D input. The blending factor is defined by the value at the setpoint input of the controller.

Ratio controllers

A ratio controller is always a slave controller. The associated master controller is a fixed-point controller.

The actual value of the master controller is selected as actual value D. The ratio factor is defined by the value at the setpoint input. If a controller output is activated as ratio factor, the setpoint value is converted from "0 .. 100%" to the value range "low threshold ... high threshold" using the high and low thresholds (normalized).

Blending controls

A blending control is always slave controller. The associated master controller is a three-component controller.

The manipulated variable of the master controller is activated as process value D. The blending factor is defined by the value at the setpoint input of the controller.

The manipulated variable of the total quantity controller is set to a range of values from 0% and 100%. The slave controller converts this variable to the range of actual value A at actual value input D. The range of actual value A is defined by the "high" and "low" normalization values of the selected analog input.

When the manipulated variable of a slave controller has reached the limiting threshold, or the setpoint gradient of a slave controller is limited by the ramp function in the setpoint value branch, the integral component of the master controller is blocked depending on the direction (anti-reset windup) until the cause of limiting has been eliminated in the slave controller.

Signal selection for setpoint value, actual value, D-action input and disturbance variable

Signal selection

You can activate different signal sources for the setpoint, actual value, value at the D input (derivative action input), and disturbance variable at each controller channel. The following table provides an overview of the signal selection options.

Table 11-6 Signal selection for setpoints, actual values, D input, and disturbance variables.

Affected values	Signal sources available
Setpoint	• A value defined by the function block in the user program • The conditioned analog value of an analog input • The manipulated value (LMN, LMN_A or LMN_B) of another controller channel (when cascading controllers)
Actual values A, B, and C	• Zero • The conditioned analog value of an analog input (the actual values B and C can also be evaluated by factors)
Actual value D	• Zero • The conditioned analog value of an analog input • Setpoint of another controller channel
Value for D input (only relevant for PD or PID controllers)	• The error signal following the deadband of the internal controller channel • The conditioned analog value of an analog input • The negated effective actual value of the controller channel
Disturbance variable	• Zero • The conditioned analog value of an analog input

Preparation setpoint

The conditioning of a setpoint to the effective process value can be influenced by the following functions:

Setpoint source

The setpoint can be defined by:

- an instruction
- an analog input
- the conditioned output value of another controller channel in the module
  This source for the output value is selected for a blending controller, or slave controller in a cascade control.
- the conditioned output value A of another controller channel in the module (FM 355-2 only)
- the conditioned output value B of another controller channel in the module (FM 355-2 only)

Safety setpoint

When the CPU fails or with a CPU stop, the module can set a safety setpoint.

Enter a safety setpoint and set "Setpoint = Safety setpoint" as behavior to CPU stop.

Reaction to CPU failure

The controller module can set the following setpoints if the CPU fails or with a CPU stop:

- Setpoint = last valid setpoint
  If the "from instruction" is set as the setpoint selection, the setpoint will remain constant at the last value set after CPU failure or CPU stop.
  If the setpoint is defined by the output value of an FM controller, or by an analog input, the setpoint will change in line with the activated value.
- Setpoint = safety setpoint
  If you entered a safety setpoint, the module corrects the setpoint to this value.

Reaction to module start (FM 355 and FM 455 only)

The controller module can correct the values to the following setpoints during the start of module:

- Setpoint = last valid setpoint
- Setpoint = safety setpoint

Ramp

You can limit the setpoint rate of change by selecting a ramp up time that covers the physical start to the end value.

Setpoint limits

If the setpoint is defined by the function block, or by a conditioned analog value of an analog input, the setpoint is limited to configurable high and low limits.

If the conditioned output value of another controller channel has been activated as setpoint for the ratio controller, this value is used as multiplication factor for process value D. The setpoint, given as a percentage at the input, can in this case be scaled using the high and low limits.

If the output value of another controller channel is used as setpoint for a fixed setpoint or cascade controller, this value can be scaled to a physical value with the help of the normalization constants of the process value channel selected.

Multiplication

Process value A is used as controlled variable, while process value D is used as ratio variable for the "ratio controller" type. The setpoint input acts as the ratio factor. It is conditioned to form an effective setpoint by multiplication with process value D and adding an adjustable offset. If process value D is disabled, only the offset is added to the setpoint.

Preparation actual value

Actual value conditioning to form the effective actual value depends on the controller structure.

Effective actual value for fixed setpoint and cascade controllers

The effective actual value is equivalent to actual value A that can be derived from an analog input source. Actual values B, C, and D are ignored.

Effective actual value for three-component controllers

The effective actual value is calculated by adding the actual values A, B, and C plus an adjustable offset. The actual values B and C can also be evaluated by factors. Actual value D is ignored.

Effective actual value in ratio controllers

Select the following actual values for the ratio controller

- Actual value A: The conditioned signal of an analog input
- Actual value D: Actual value of the master controller.

The ratio factor is defined as setpoint.

If a controller output has been activated as ratio factor FAC, the setpoint is converted from "0 ... 100%" to the value range "low threshold ... high threshold" using the low and high thresholds (scaled).

Effective actual value in blending controllers

Select the following actual values for the blending controller

- Actual values A, B, and C The conditioned signals of the associated analog inputs.
- Actual value D: Actual value of the master controller.

The ratio factors of the individual controlled systems are defined as setpoint.

The manipulated variable of the total quantity controller is set in the range of values from "0% to 100%". The slave controller converts this variable at input D to the range of values between "low threshold ... high threshold" of actual value A (scaled).

The manipulated variable of the master controller is interconnected via "Actual value D" input of the slave controller. Blending factor FAC is defined by the value set at the controller's setpoint input.

Interrupt

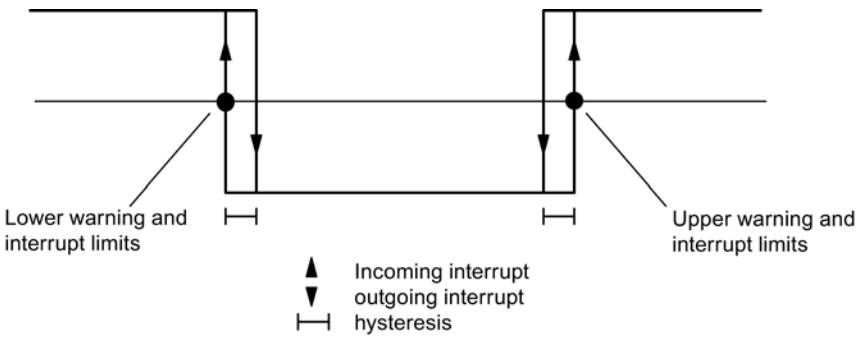
Limit monitoring

The controller module contains a limit monitoring function. This enables monitoring of the high and low limits

- of the control deviation
- of the effective process value

for high and low limits warnings and alarms. You can set up a hysteresis for these limits.

The following figure shows the hysteresis for warning and alarm limit:



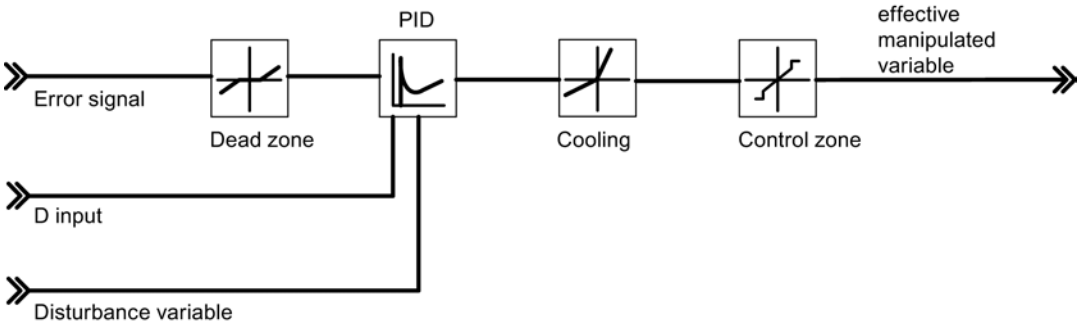
Control algorithm

FM 355-2 control algorithm

Components of the control algorithm

Continuous controllers (FM 355-2 C) and pulse controllers (FM 355-2 S) have the same control algorithm structure. The cooling and control zone buttons cannot be selected for the step controller (FM 355-2 S).

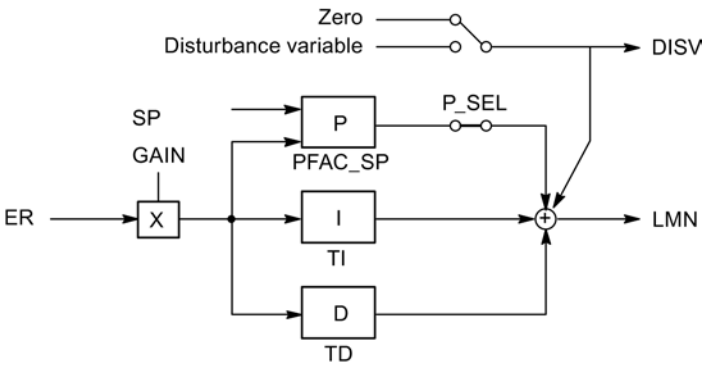
The following figure shows the block diagram of the control algorithm for continuous and pulse controllers.



Control algorithm: PID in parallel structure

During the cycle of the configured sampling time, the controller's manipulated variable is calculated based on the control deviation of the PID position algorithm. The controller is implemented in a purely parallel structure. The proportional, integral and derivative actions can each be disabled separately. To disable the integral and derivative actions, set the respective parameter TI or TD to zero.

The following figure shows the control algorithm of the FM 355-2 (parallel structure):



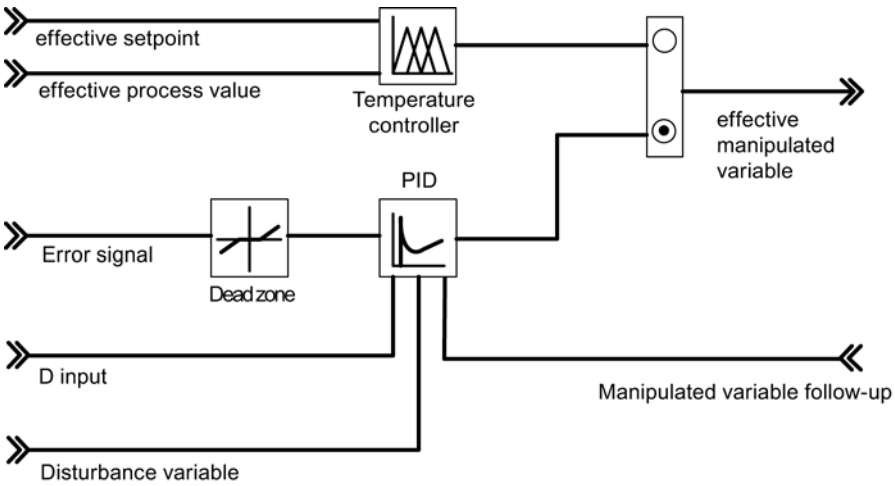
FM 355 and FM 455 control algorithm

Components of the control algorithm

You can choose between the following operating modes of the control algorithm:

- Temperature controller (self-tuning fuzzy controller)
- PID controller

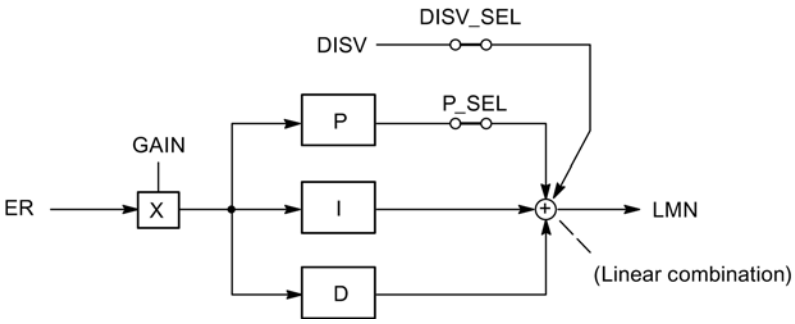
C and S controllers have the same control algorithm structure. The following figure shows the block diagram of the control algorithm:



Control algorithm and controller structure

The manipulated variable of the continuous controller is calculated within the configured sampling time on the basis of the control deviation in the PID positioning algorithm. The controller is designed with a purely parallel structure (see figure below). The proportional, integral and derivative actions can each be disabled separately. To disable the integral and derivative actions, set the respective parameter TI or TD to zero.

The following figure shows the control algorithm of the FM 355 (parallel structure):



Selecting the operating mode

Selecting the operating mode

You can choose between different operating modes if the "PID controller" entry has been set as controller type:

- P, for implementing a proportional controller
- PI, for implementing a PI controller
- PD, for implementing a PD controller
- PID, for implementing a PID controller

- I, for implementing an integrating controller
- ID, for implementing an ID controller
- D, for implementing a D controller

Bumpless change from manual to automatic mode

If you selected “Bumpless change from manual to automatic mode” (not available for step controllers), the integrator is tracked in manual mode so that the manipulated variable does not take a step through the proportional and derivative actions during the manual to automatic changeover. An error signal is corrected only slowly by means of the integral action. If bumpless manual to automatic changeover is not selected, the manipulated variable performs a step that corresponds with the current error signal during the change from manual to automatic mode. This function allows for the fast correction of active control error.

Note

A step controller is always subject to pulse action at the changeover from manual to automatic mode. The existing error signal and GAIN leads to a jump in the internal manipulated variable. The integral effect of the actuator, however, results in a ramp-shaped excitation of the process.

Disturbance variable compensation

An additional disturbance variable DISV can be added to the controller’s output signal. This is activated and deactivated in the “System deviation” window of the configuration tool using the switch “Signal selection, controller disturbance variable”.

Dead zone

Purpose of the deadband

A deadband is interconnected upstream of the PID controller. In settled controller state, the deadband suppresses the noise component in the error signal that could be generated by superimposition of a disturbance signal of higher frequency on the control or reference variable. This function prevents unwanted oscillation of the controller output.

Deadband width

The deadband width can be configured. If the error signal is within the set dead band width, the value 0 (error signal = 0) is returned at the deadband output. The output value does not follow the input variable until the input variable moves outside the sensitivity range.

This results in corruption of the transferred signal even outside the deadband. However, this is accepted in order to avoid jumps at the limits of the deadband. The corruption corresponds to the value of the deadband width and can therefore be controlled easily.

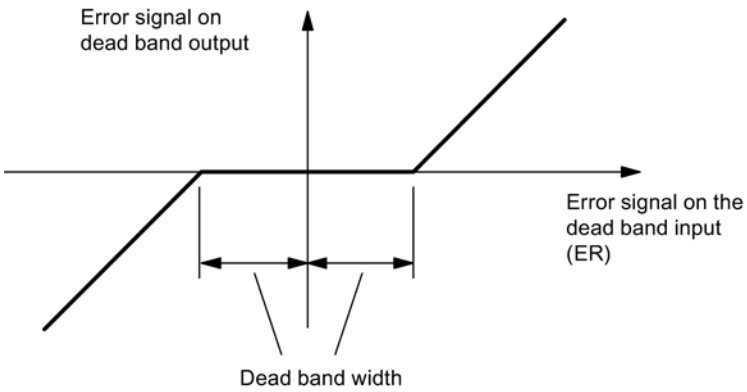
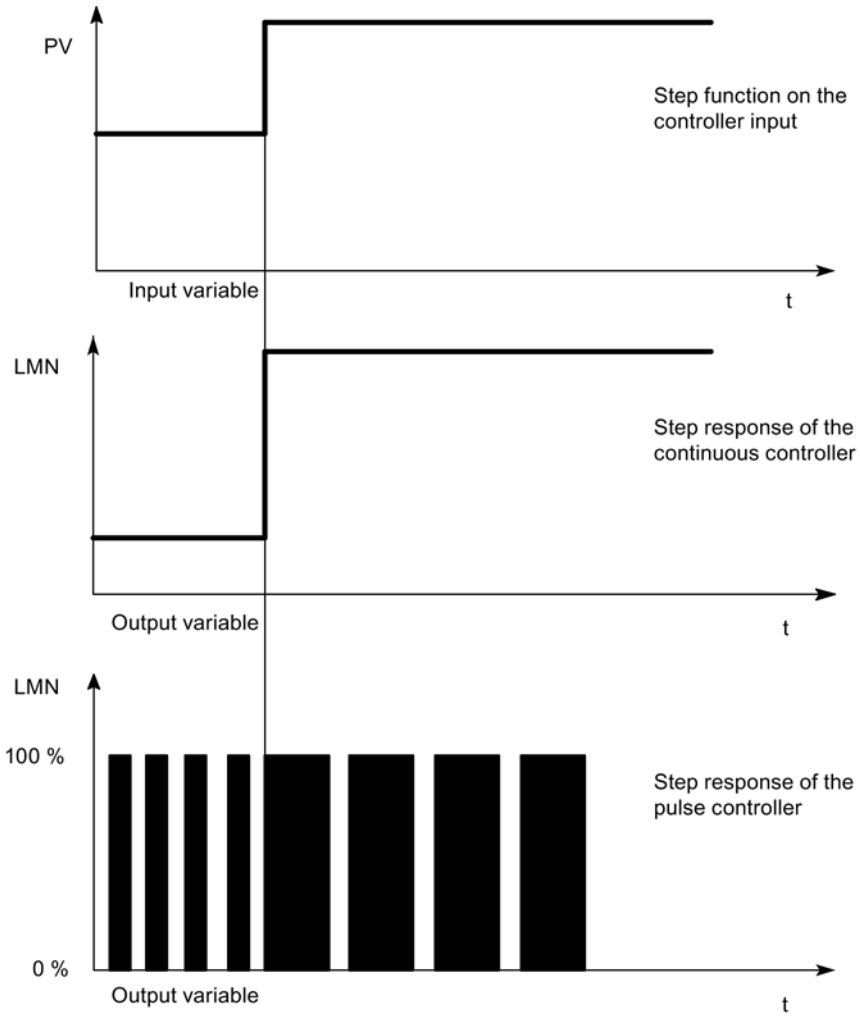


Figure 11-9 Deadband

Proportional control

Proportional controller

The following figure shows the step response of a proportional controller:



Formula for proportional controllers

The output and input variables are directly proportional, i.e.:

Change to the output variable = proportional coefficient x change to input variable

LMN = GAIN x ER

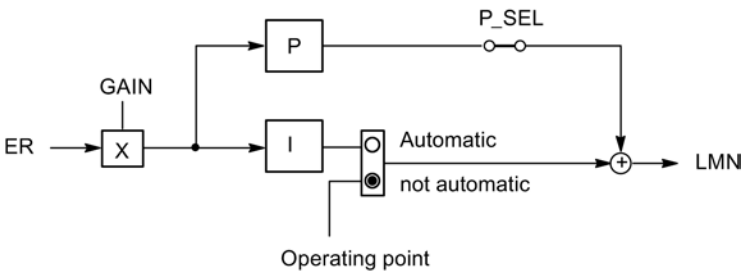
Proportional control

The integral and derivative actions are disabled in a proportional controller. This means that the manipulated variable is also "0" when control deviation ER = 0. If an operating point ≠0, in

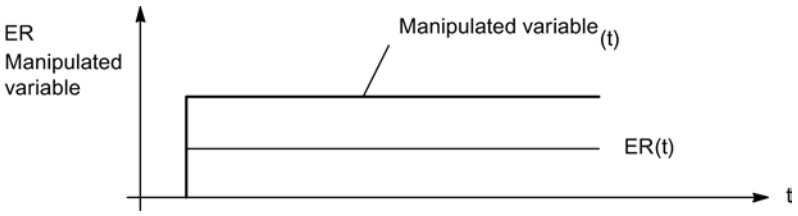
other words, a numeric value is to be set for the manipulated variable when control deviation = 0, the following is possible via the operating point:

- Automatic operating point: The controller automatically sets the operating point to the value of the current (manual) manipulated variable when you switch from manual to automatic mode.
- Operating point not automatic: You can assign the operating point parameters.
- Example: Operating point AP = 5% results in a manipulated variable of 5% when control deviation ER = 0.

The following figure shows a proportional controller with operating point setting by means of an integral element:



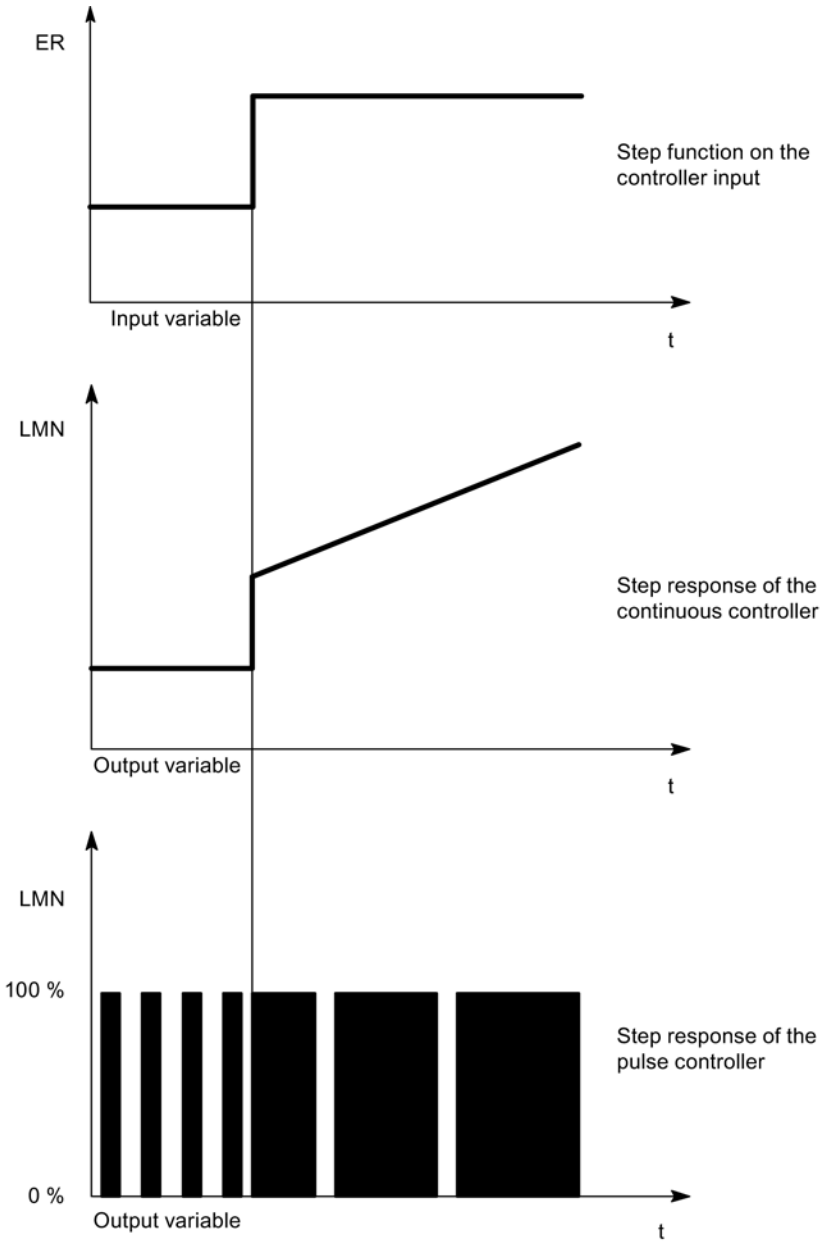
The following figure shows the step response of the proportional controller:



PI control

PI controller

The following figure shows the step response of a PI controller:



Integral control elements have the integral of the input variable as the output variable, i.e. the controller adds up the deviation from the setpoint value over time. This means that the controller continues correction until the deviation from the setpoint is eliminated. In practice, depending on the demands on the controlling, a combination of the different timing elements is ideal. The

timing of the individual elements can be described by the controller parameters Proportional band GAIN, Reset time TI (integral action) and Rate time TD (derivative action) .

Formula for PI controllers

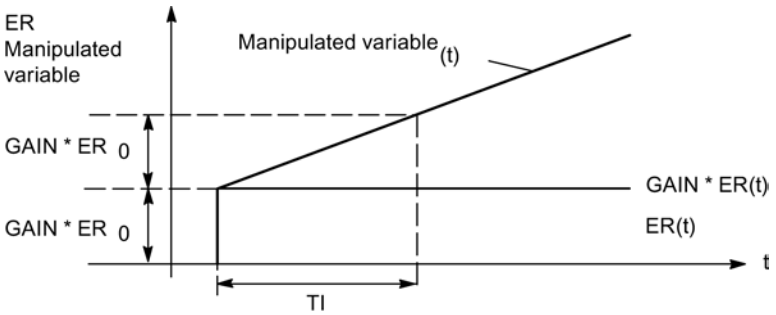
The following applies for the step response of the PI controller in the time range:

$$LMN = GAIN \times ER \times \left(1 + \frac{t}{TI}\right)$$

t = time interval since the input variable step

PI control

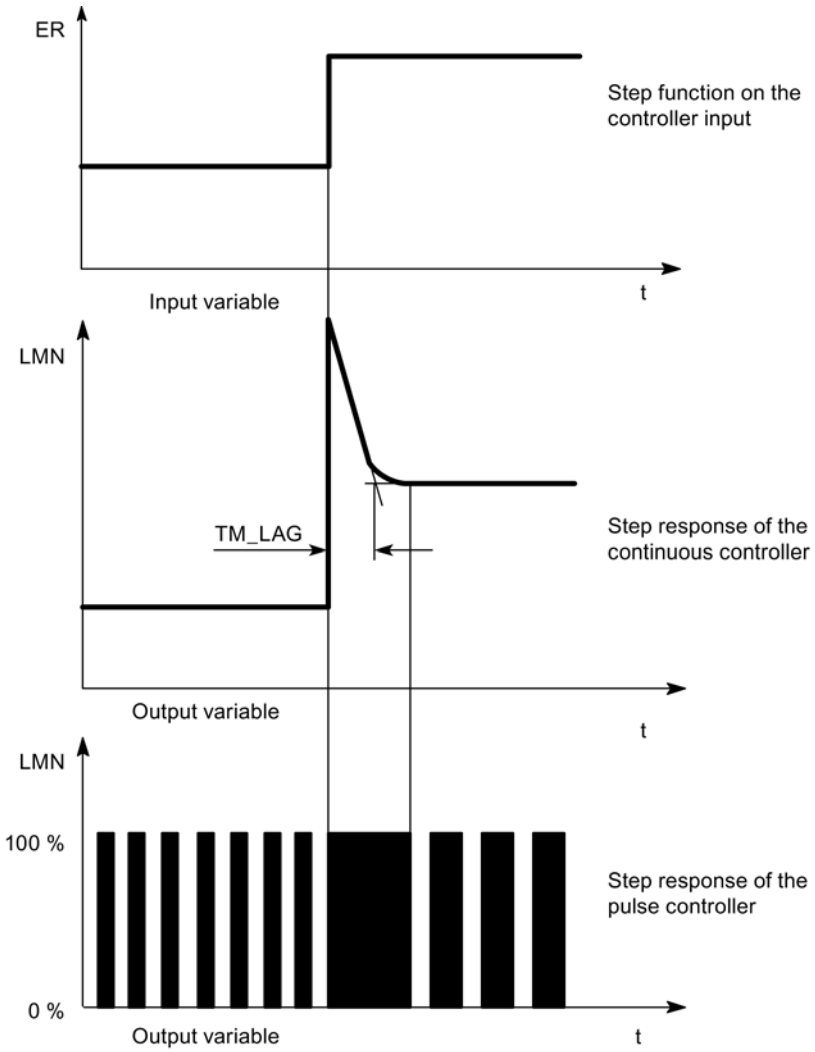
The derivative action is disabled in the PI controller (TD=0.0). A PI controller corrects the output variable by means of integration until control deviation ER = 0. However, this only applies if the output variable does not exceed the limits of the output range. The integral action retains the value reached at the output value limit if these limits are exceeded (anti-reset windup).



PD control

PD controller

The following figure shows the step response of a PD controller:



The derivative control elements are the only ones unsuitable for controlling, because they no longer output actuating commands when the input variable has settled to a static value.

In combination with proportional control elements, the derivative action is used to generate a control pulse based on the rate of change at the controlled variable. If a disturbance variable z is affecting the controlled system, the PD controller sets a different control deviation as a result of the changed degree of correction. Disturbance factors are not fully corrected. Of advantage is the good dynamic response. A well attenuated, non-oscillating transfer is achieved during startup and when changing the reference input variable. However, a controller with derivative action is not appropriate if a control system has oscillating measured variables, e.g. in pressure or flow control systems.

Formula for PD controllers

The following applies for the step response of the PD controller in the time range:

$$LMN = GAIN \times ER \times \left(1 + \frac{TD}{TM_LAG} \times e^{\frac{-t}{TM_LAG}}\right)$$

t = time interval since the input variable step

PD control

The integral action is disabled in the PD controller (TI=0.0). This means also that output signal = 0 when control deviation ER = 0. If an operating point ≠0, in other words, a numeric value is to be set for the manipulated variable when control deviation = 0, the following is possible via the operating point:

- Automatic operating point: The controller automatically sets the operating point to the value of the current (manual) manipulated variable when you switch from manual to automatic mode.
- Operating point not automatic: You can assign the operating point parameters.

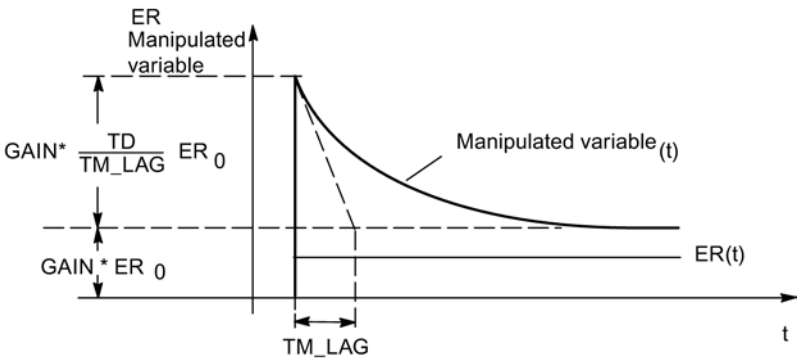
The PD controller maps input variable ER(t) proportionally to the output signal and adds the derivative action formed by the differentiation of ER(t); the derivative action is calculated with double precision according to the trapezoid rule (Padé approximation). The time response (strength of the derivative action or actuating area) is determined by the derivative action time TD (rate time).

For signal smoothing and interference suppression, the derivative action is implemented with a time lag of the first order (adjustable time constant: TM_LAG). A small value for TM_LAG is usually sufficient to achieve the desired result.

The higher the derivative factor D_F,

- the smaller the effective time constant TD/D_F of the delay,
- the higher the maximum initial manipulated variable,
- and the better the control action;
- the higher the noise sensitivity.

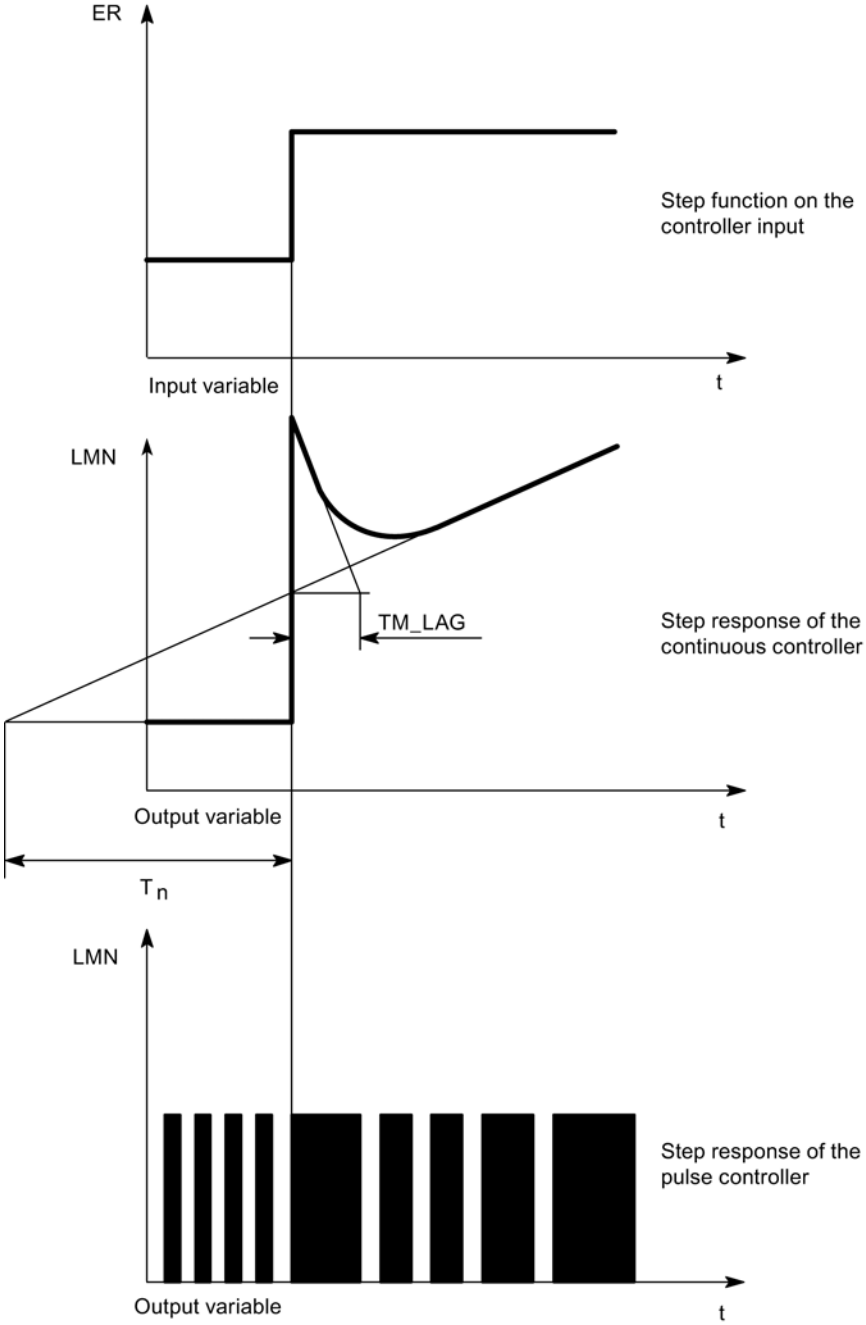
D_F is limited to the value range 5.0 through 10.0.



PID control

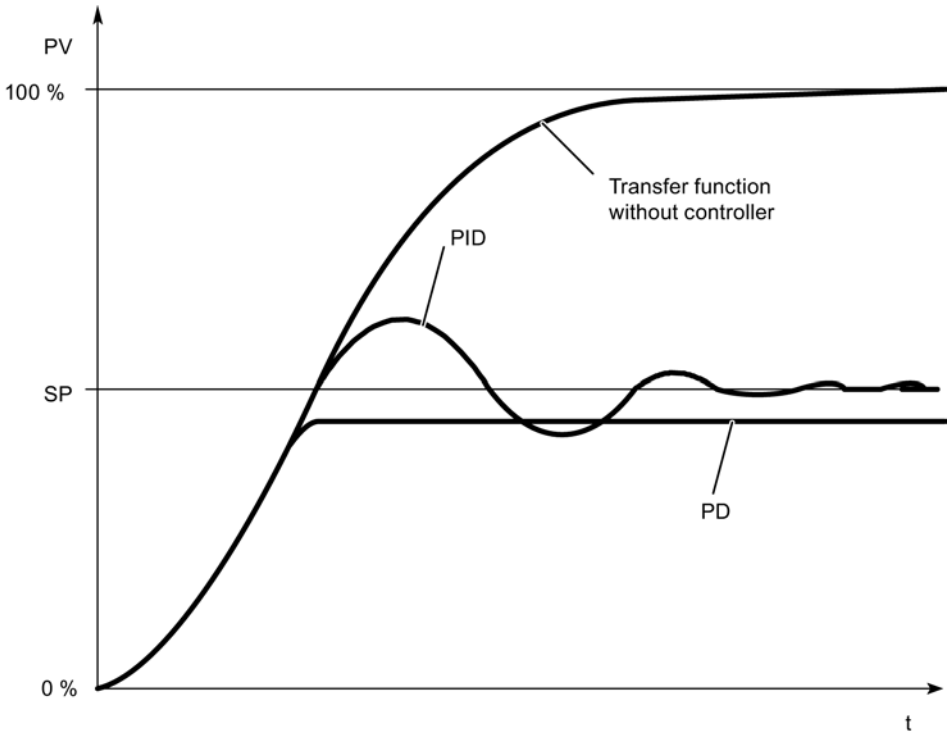
PID controller

The following figure shows the step response of a PID controller:



Most control systems in process engineering can be managed with a controller with PI component. In the case of slow control systems with long dead times, for example temperature control systems, the control result can be improved with a controller with PID action.

The following figure shows the step response for various proportional controllers:



Controllers with PI and PID action have the advantage that the process variable does not deviate from the setpoint value after settling. The controlled variable oscillates above the setpoint value upon start-up.

Formula for PID controllers

The following applies for the step response of the PI controller in the time range:

$$LMN = GAIN \times ER \times \left(1 + \frac{t}{TI} + \frac{TD}{TM_LAG} \times e^{\frac{-t}{TM_LAG}}\right)$$

t = time interval since the input variable step

PID control

The P, I and D actions are switched on in PID controllers. A PID controller adjusts the output variable via the integral action until the control deviation ER = 0. However, this only applies if the output variable does not exceed the limits of the output range. The integrator maintains

the value it has at the point where the limits of the output value are exceeded (Anti-Reset-Windup).

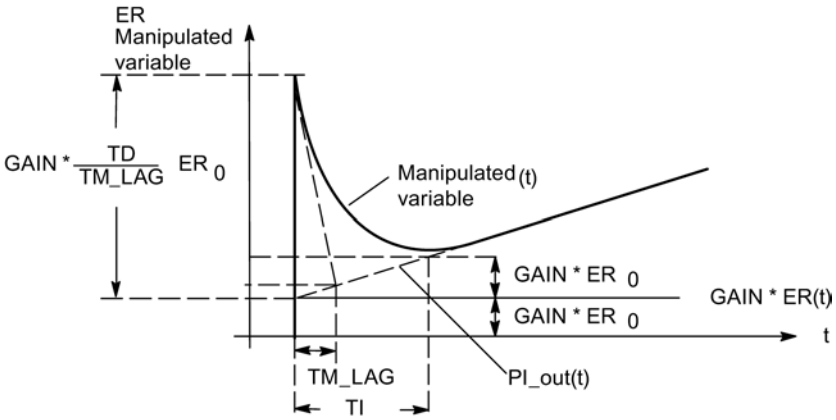
The PID controller proportionally maps the input variable ER(t) to the output signal and then adds the actions generated by the differentiation and integration of ER(t). The time action is determined by the derivative action time TD (rate time) and the integration time TI (reset time).

The derivative action is realized with a delay circuit of the first order for signal smoothing and interference suppression.

The higher the derivative factor D_F,

- the smaller the effective time constant TD/D_F of the delay,
- the higher the maximum initial manipulated variable,
- and the better the control action;
- the higher the noise sensitivity.

D_F is limited to the value range 5.0 through 10.0.



I control

Integral controller

You can deactivate the proportional action in order to realize a pure integral-action control. This can also be done using the P_SEL parameter of the PID_FM function block.

Proportional action

Attenuation of the proportional action in the event of setpoint changes

You can avoid process value overshoot or excessive manipulated variable amplitude with attenuation of the proportional action using the parameter “Proportional factor at setpoint

change” (PFAC_SP). Using PFAC_SP, you can select continuously between 0.0 and 1.0 to decide the effect of the proportional action when the setpoint changes:

- PFAC_SP=1.0: Proportional action for setpoint change is fully effective
- PFAC_SP=0.0: Proportional action for setpoint change is not effective

There are two ways of limiting the rate of setpoint change in the FM355-2:

- Activate ramp (> 0.0s)
- Setpoint change factor < 1.0

Use only one of the two limits. If both limits are activated at the same time, a setpoint jump will produce a change in the manipulated variable in the opposite direction to the setpoint change (step response).

Specific features of the step controller

A value of PFAC_SP < 1.0 can reduce overshoot if the motor transition time MTR_TM is small compared to the recovery time TA and if the ratio TU/TA < 0.2. Should MTR_TM reach 20 % of TA, only a slight improvement can be achieved.

Inversion of the controller effect

You can enable controller inversion, that is, conversion from

- increasing control deviation = rising manipulated variable
  to
- rising control deviation = falling manipulated variable

by setting a negative proportional gain (GAIN). The sign of this parameter value determines the effective direction of the controller.

Derivative action

Derivative action in feedback path

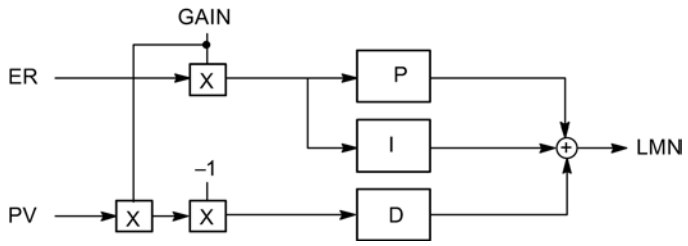
When the setpoint changes, you can avoid pulse-shaped peaks of the derivative action of the manipulated variable by moving the derivative action into the feedback path.

In this structure only the negative control variable (Factor = -1) is fed forward to the derivative action. Changeover for the derivative action is carried out in the feedback in the “Control deviation” window via the “Derivative input controller” switch by selecting the negated effective process value as the input signal.

Note

If you move the derivative action to the feedback path, you should also reduce the value of PFAC_SP, otherwise you would increase overshoot of the process value.

The following figure shows the control algorithm with derivative action in the feedback branch:



Proportional/Derivative action

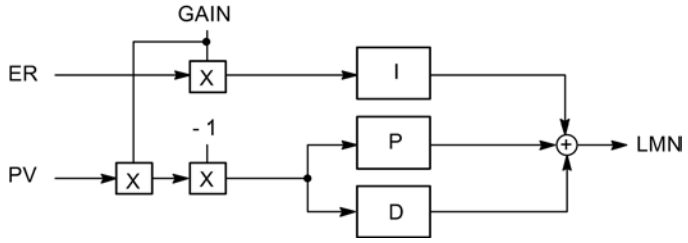
P/D action in the feedback

In the parallel structure, each action in the control algorithm receives the control deviation as input signal. In this structure, setpoint steps act directly on the controller. The manipulated variable is influenced directly by the proportional and derivative actions by setpoint steps.

However, a different controller structure, in which the formation of the proportional and derivative action is moved to the feedback, ensures a smooth response of the manipulated variable to setpoint step changes (see figure below).

In this structure, the integral action processes the control deviation as the input signal. Only the **negative** controlled variable (factor = - 1) is applied to the proportional and derivative actions. Changeover for the derivative action is carried out in the feedback in the “Control deviation” window via the “Derivative input controller” switch by selecting the negated effective process value as the input signal.

The following figure shows the control algorithm with proportional and derivative action in the feedback branch:



Fuzzy temperature controller module

Temperature controller

The temperature controller is a self-regulating fuzzy controller, which, after identifying the control section, works using control parameters it has calculated itself.

The following temperature controller settings are possible:

- Cooling controller
- Heating controller
- Aggressivity
  The aggressivity parameter can be used to influence the speed of the transient response.

Possible aggressivity values:	
-1 ≤ Aggressivity < 0 :	Slower transient response than determined by identification
Aggressivity = 0	Transient response as determined via identification
0 < Aggressivity ≤ 1	Faster transient response than determined by identification

Controller output

Differences between controllers

The FM 355, FM 355-2 and FM 455 controllers are basically divided into two types: step controllers and continuous controllers. Step controllers are identified by an added S, such as FM 355 S, FM 355-2 S and FM 455 S, whereas continuous controllers are identified with a C, such as FM 355 C, FM 355-2 C and FM 455 C. The way the controller modules work do not differ, an FM 355 S and an FM 355 have identical algorithms. The differences lie in the controller output.

Step controller

The continuous step controller has digital outputs. Therefore, a control valve can only be addressed by the signals "open" or "closed". If you wish to control a control valve for 50% of the capacity, for example, this can only be realized by setting "open" and "closed" to an equal length.

You can find more detailed information on step controllers at the following locations:

- Digital controller outputs (Page 9825)
- Step controller without position feedback (Page 9827)
- Step controller with position feedback (Page 9827)
- Pulse controller (Page 9827)

Continual controller

The continuous step controller has analog outputs. A control valve can not only be "open" or "closed", it can also be opened by specific percentages. If you wish to control a control valve for 50% of the capacity, for example, the controller module outputs the value "50%" and the door opens halfway.

You can find more detailed information on continuous controllers at the following locations:

- Controller analog outputs (Page 9825)
- Continuous controller (Page 9826)

Controller analog outputs

Analog outputs

You can assign parameters for the following functions for each analog output:

- Signal source
- Signal type

Selection of source for the signal at the analog outputs

Selection of the signal source allows you to define which signal value is to be output at the analog output in question.

The following signal sources can be assigned:

- The value zero
- The conditioned analog value of one of the analog inputs
- Manipulated value A of one of the controller channels
- Manipulated value B of one of the controller channels

The FM 355 and FM 355-2 each have four analog inputs and four controller channels.

The FM 455 has 16 analog inputs and 16 controller channels.

Signal Type at the analog outputs

You can determine the signal type for each analog output.

The following signal types can be assigned:

- Current output 0 ... 20 mA
- Current output 4 ... 20 mA
- Voltage output 0 ... 10 V
- Voltage output -10 ... 10 V

Digital controller outputs

Digital controller outputs

The digital outputs of the controller are used to control integrating and non-integrating actuators.

You can assign the following functions to each digital output:

- Output type
- Signal selection
- Signal processing

Signal output type at the digital output

You can define the signal type for each digital output.

The following signal types are available:

- Step controller w/o position feedback input
- Step controller with position feedback input
- Pulse controller

Signal selection

A range of values can be assigned for signal selection depending on the signal type at the output:

- External manipulated variable (Page 9828)
- Compensation (Page 9828)
- Position feedback input (Page 9829)
- Safety manipulated value (Page 9830)

Signal processing

A range of values can be assigned for signal processing depending on the signal type at the output:

- Output signal limit (Page 9831)
- Split range (Page 9831)
- Pulse shaper (Page 9834)

Continuous controller

Signal type at the digital outputs

The following parameters can only be set for an analog module.

Available parameters

- Activateexternal manipulated variable (Page 9828)
- Activatecompensation (Page 9828)
- Setsafety manipulated value (Page 9830)
- Limit manipulated value (Page 9831)
- Enable two actuators (Page 9831)
- Select output signal and signal type (Page 9834)

Step controller without position feedback

Signal type at the digital outputs

The following parameters can only be set for a digital module.

Available parameters

- Activateexternal manipulated variable (Page 9828)
- Setsafety manipulated value (Page 9830)
- Pulse shaper (Page 9834)

Step controller with position feedback

Signal type at the digital outputs

The following parameters can only be set for a digital module.

Available parameters

- Activateexternal manipulated variable (Page 9828)
- Activatecompensation (Page 9828)
- Setsafety manipulated value (Page 9830)
- Limit manipulated value (Page 9831)
- Setposition feedback (Page 9829)
- Pulse shaper (Page 9834)

Pulse controller

Signal type at the digital outputs

The following parameters can only be set for a digital module.

Available parameters

- Activateexternal manipulated variable (Page 9828)
- Activatecompensation (Page 9828)
- Setsafety manipulated value (Page 9830)
- Limit manipulated value (Page 9831)
- Conditioning analog signal for conversion to a digital signal (Page 9831)
- Converting analog signal to digital signal (Page 9800)

Functions of the controller output

External manipulated variable

Overview

There are various interconnection possibilities for the manipulated value, the compensation input and the safety manipulated value at the controller output (manipulated value changeover).

A limit ensures that the manipulated value cannot assume invalid values for the process.

Switching external manipulated value

The changeover between external manipulated value and effective manipulated value from the controller can alternatively take place using

- A binary value from the function block
- A signal from a logical OR link of a digital value from a function block and a digital input.

Compensation

Overview

Manipulated value compensation prevents a jump in the manipulated value during the changeover from manual to automatic mode.

The manipulated value remains unchanged during the changeover from manual to automatic mode. Manipulated value compensation is not active if a purely proportional controller with a fixed operating point has been implemented (”automatic” is not checked in the PID controller screen).

Compensation input

The following alternative settings are available:

- The value at the compensation input = zero
- The compensation input is the conditioned analog value of an analog input

Switching to compensation

You can also change from manipulated value to compensation input using

- A binary value from the function block
- A signal from a logical OR link of a digital value from a function block and a digital input.

Position feedback

Overview

A position feedback input is only available for step controllers.

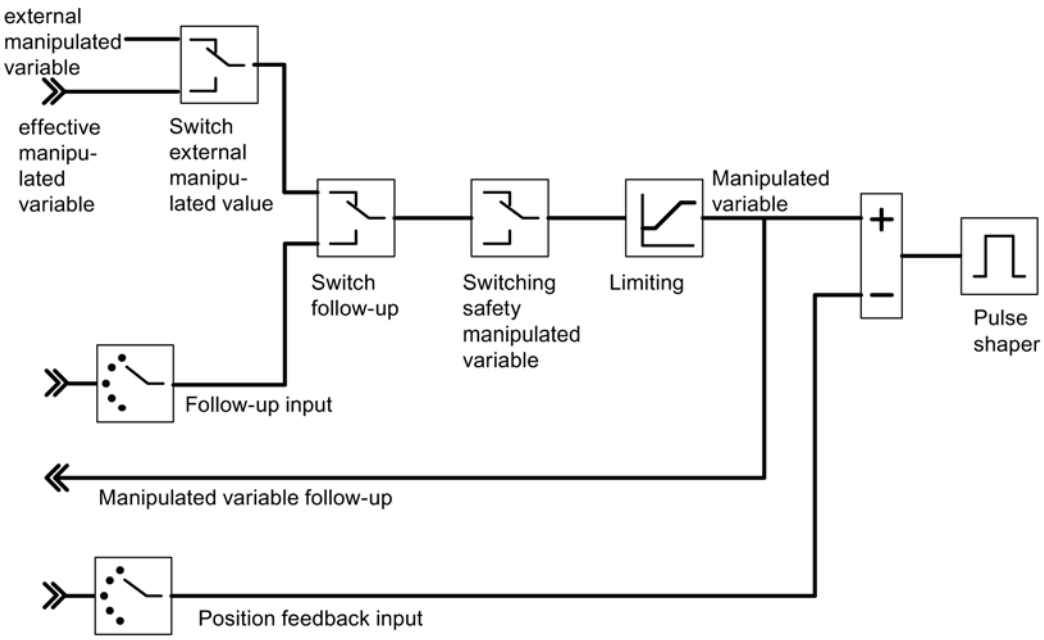
Position feedback input

The following alternative settings are available:

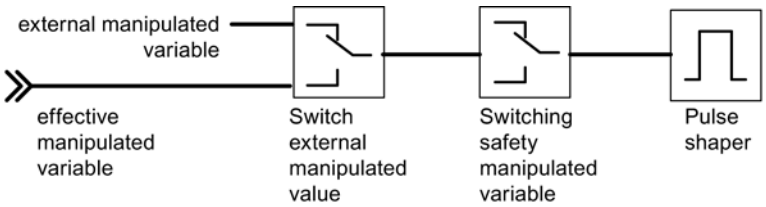
- The position feedback input value = zero
- The position feedback input is the conditioned analog value of an analog input

Controller output of the step controller

The following figure shows the principle of the S controller output (step controller operating mode with position feedback):



The following figure shows the principle of the S controller output (step controller operating mode without position feedback):



In a step controller without analog position feedback, the external output value and the safety output value act as follows:

If a value between 40.0% and 60.0% is specified, no binary output is set and the actuating device remains unchanged.

If a value > 60.0% is specified, "Actuating signal high" is output until the feedback "Actuating device at high limit" is triggered.

If a value < 40.0% is specified, "Actuating signal low" is output until the feedback "Actuating device at low limit" is triggered.

Safety manipulated value

Overview

There are various interconnection possibilities for the manipulated value, the compensation input and the safety manipulated value at the controller output (manipulated value changeover).

Safety manipulated value switching

- Establishing the safety manipulated value
- Alternative reaction of the function module upon start-up:
  - The function module switches to closed-loop operation
  - The safety manipulated value is output as the manipulated value
- Changeover to the safety manipulated value is also possible with
  - A binary value from the function block
  - A signal from a logical OR link of a digital value from a function block and a digital input.
- Reaction to measuring transducer fault of actual value A:
  - The operating mode of the controller remains unchanged when "Closed-loop control operation" is set
  - It changes to the safety manipulated value when "manipulated value = safety manipulated value" is set
- Reaction to analog input measuring transducer fault:
  - The operating mode of the controller remains unchanged when "Closed-loop control operation" is set
  - It changes to the safety manipulated value when "manipulated value = safety manipulated value" is set

Manipulated value limitation

Overview

A limit ensures that the manipulated value cannot assume invalid values for the process.

Manipulated value limit

- High and low limit (cannot be switched off)

Split range

Overview

The split range function generates two differently standardized output signals, output value A and output value B, from the output value as input signal. This allows two valves, for example, to be controlled with one output value.

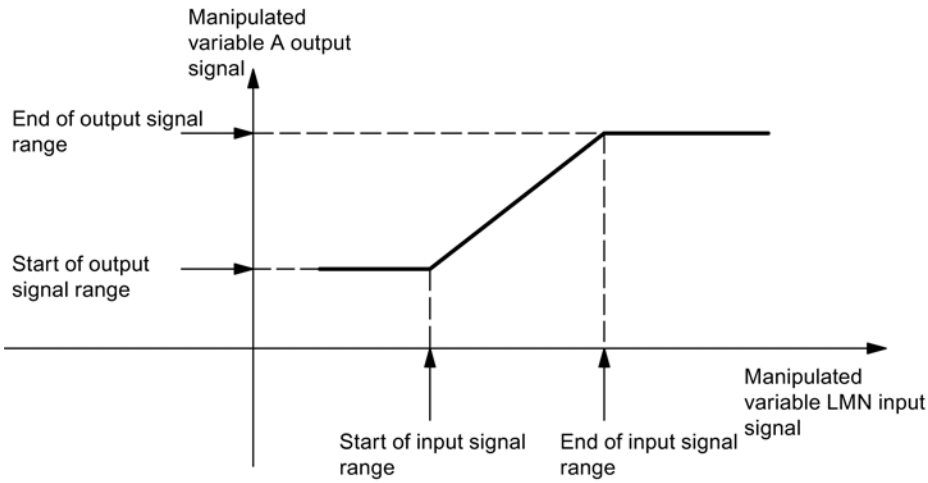
Formation of the split range output value

- In / out (C controllers only)
- Initial and final value - input signal
- Initial and final value of the output signal

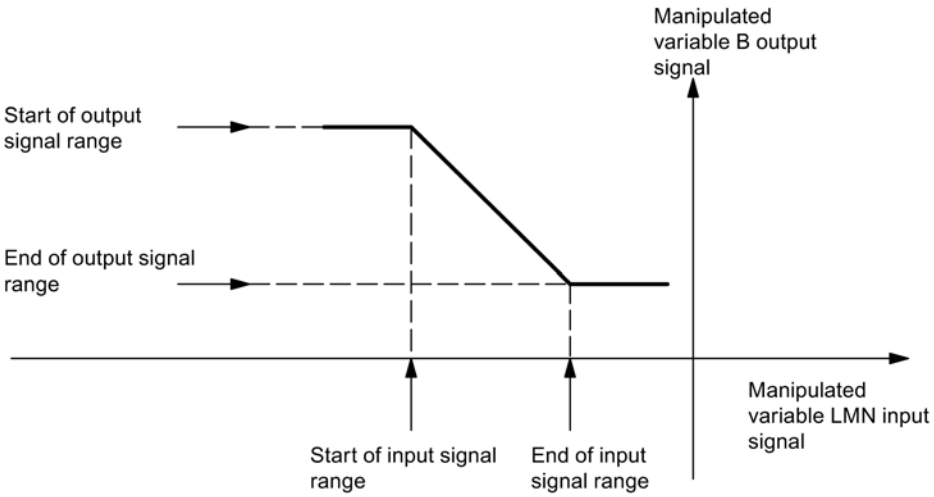
Split range

The splitrange function allows two control valves to be controlled with one manipulated variable. The split range function generates the two output signals output value A and output value B from the output value LMN as input signal.

The following figure shows the effect of the parameters for the output of output value A:



The following figure shows the effect of the parameters for the output of output value B:



The start of the input signal area must be less than the end of the input signal area.

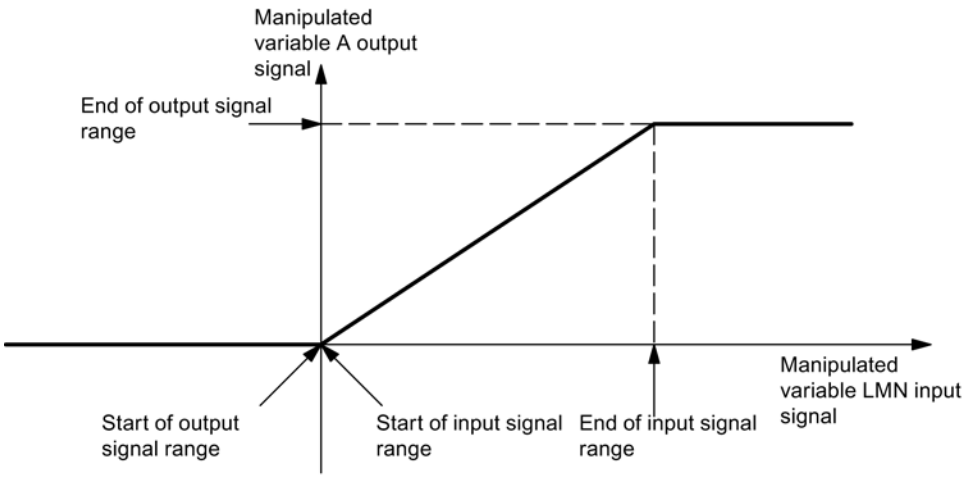
Split range / pulse shaper

The split range function is the preparation of the analog signal for conversion to a binary signal.

In the case of a **two-step controller** (e. g. a heating controller), only the output value A is of relevance. The conversion of the output value to the output value A is shown in the following figure. Conversion to a binary output signal is carried out so that the ratio of pulse length to period duration corresponds to the output value A at the assigned digital output.

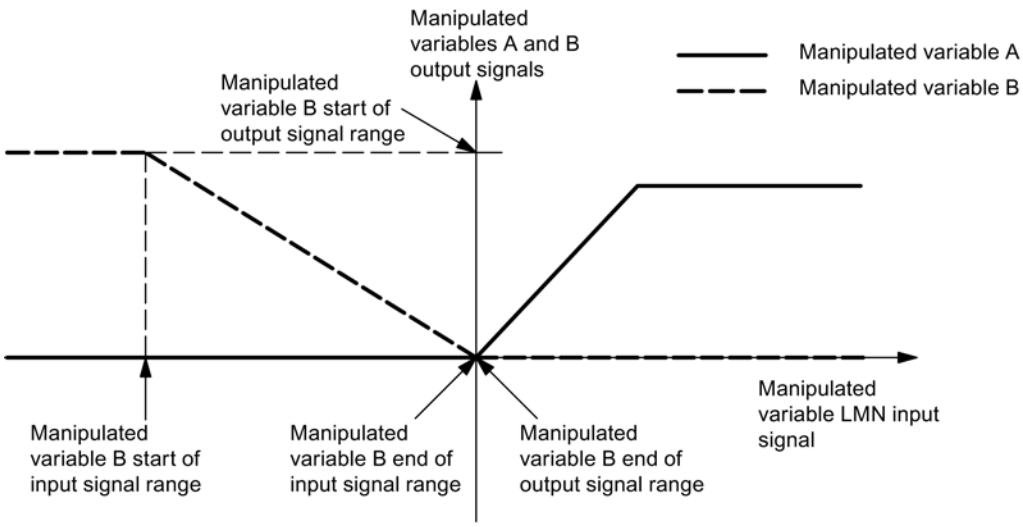
For example, a output value A of 40% and a period of 60 seconds result in a pulse length of 24 seconds and a pause duration of 36 seconds.

The figure below shows the split range function on a two-step controller:



The statements above apply for the output value A in the case of a **three-step controller** (for example, a heating and cooling controller). The second signal for controlling the cooling is generated via output value B. The conversion of the output value to the output values A and B is shown in the figure below. The conversion to a binary output signal is carried out so that the ratio of pulse length to period duration corresponds to the output values A and B at the assigned digital outputs.

The figure below shows the split range function on a three-step controller:



Pulse shaper of the step controller

Overview

The pulse shaper is only available for step controllers (FM 355 S, FM 355-2 S and FM 455). The followings settings can be configured.

Pulse shaper (step controllers only)

- Motor control time
- Minimum duration of pulse
- Minimum pause time

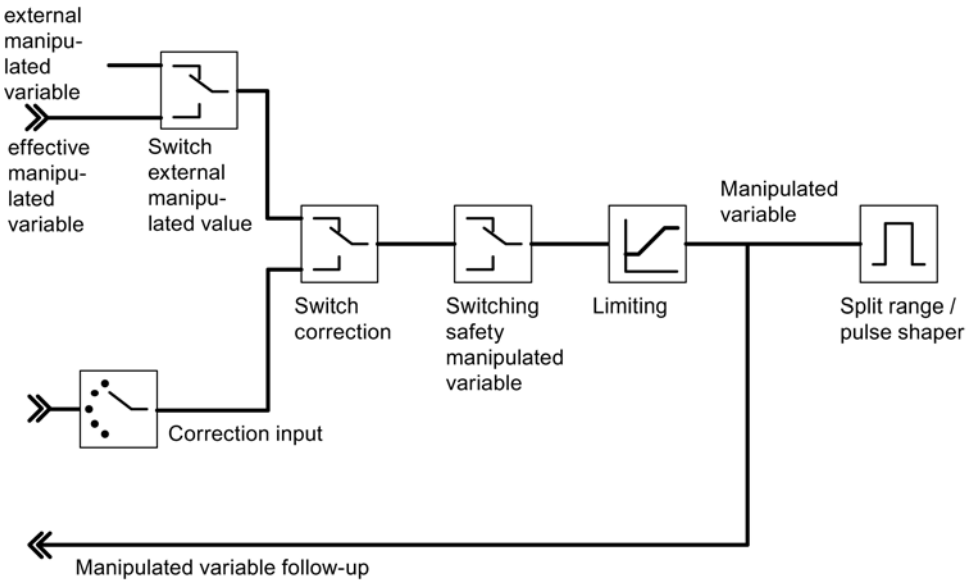
Select output signal and signal type

Analog output

You can select which signal is to be output for each channel at the analog output. This is usually the output value A of a controller. However, you can also select the output value B of a controller or indeed an analog input value. The latter can be used for the linearization of an analog value. This allows, for example, the signal supplied by a thermocouple to be linearized and converted to a range of 0 V to 10 V.

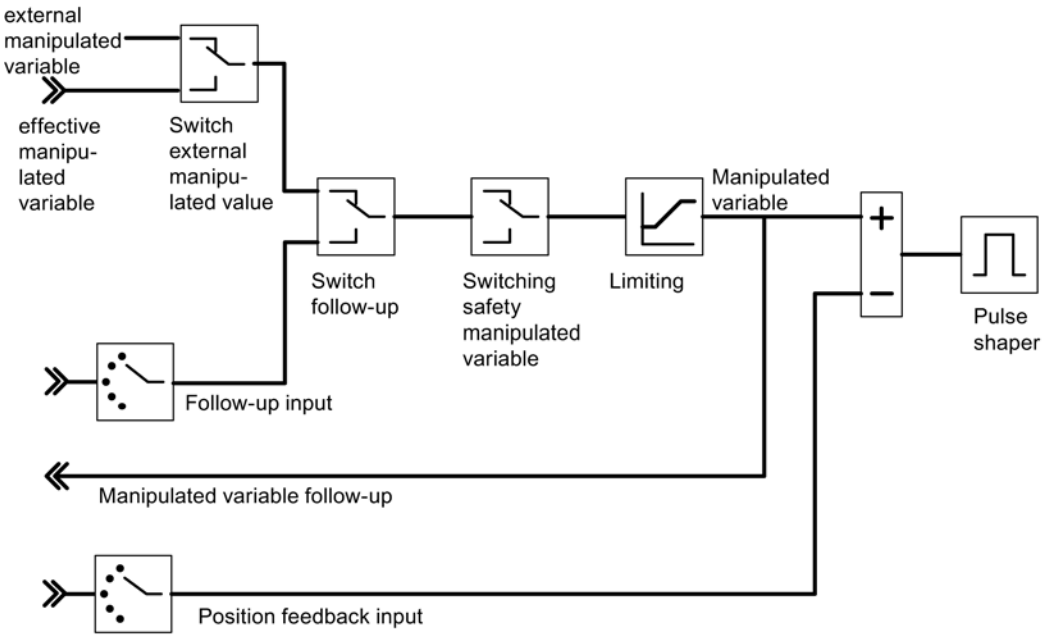
Controller output of the pulse controller

The following figure shows the controller output of the step controller (pulse control mode):

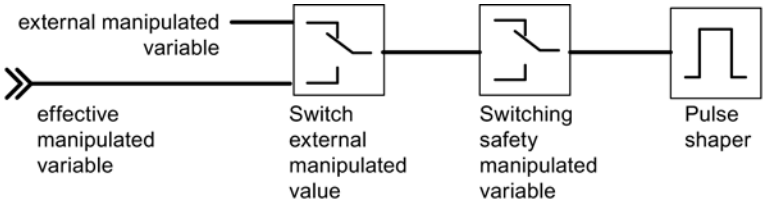


Controller output of the step controller

The following figure shows the controller output of the step controller (step control with position feedback mode):



The following figure shows the controller output of the step controller (step control without position feedback mode):



In a step controller without analog position feedback, the external output value and the safety output value act as follows:

If a value between 40.0% and 60.0% is specified, no binary output is set and the actuating device remains unchanged.

If a value > 60.0% is specified, "Actuating signal high" is output until the feedback "Actuating device at high limit" is triggered.

If a value < 40.0% is specified, "Actuating signal low" is output until the feedback "Actuating device at low limit" is triggered.

11.3.5.5 Using FM 355

Configuring FM 355

Configuring the hardware

Procedure

Configuration assumes that you have set up a project in which you can save the parameter assignment. The following briefly explains the most important steps.

1. Start the TIA portal and open the device configuration.
2. Select one module rack.
3. Select one controller module from your module catalog.
4. Move the controller module to the appropriate place on the module rack.
5. From the configuration table, note the input address of the module, for example 272.
   The value that you read is displayed in decimal format.

Parameter assignment FM 355

Module parameters

Procedure

When assigning parameters you set the module parameters.

1. Double click in the device view under the "Parameter setting" module. Select the "Properties" tab in the inspector window.
   Result: The properties are displayed in the inspector window.
2. Set the basic settings of the module and the other parameters.
3. Set the further block settings in the project view.
4. Save the entered parameters with "Project > Save".
5. Translate the project with "Edit > Translate".
6. Transfer the parameter data with the CPU in STOP mode by selecting "Online > Load in device..."
   Result: The data is located in the CPU´s memory and will be directly transferred from there to the module.
7. Carry out a CPU start-up.

Analog input parameter

Evaluation of the analog input

Procedure

To select a type of sensor, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Analog Input > Input (No.) > Evaluation of Analog Input".
3. Set the following parameter:
   - Resolution of analog input
   - Analog input sensor type
     The "wire-break monitoring" options is open for different sensor types. Depending on the requirements, the option can be activated.
   - reference junction temperature
4. Save the entered parameters with "Project > Save".

Filter functions

Procedure

To set a filter function, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Analog Input > Input (No.) > Filter functions".
3. Set the following parameter:
   - To filter according to a time constant, activate the "Signal filter" option.
     If the "Signal filter" option is selected, the filter time constant can be set.
   - Filter time constant
     Can only be set if the "Signal filter" option is activated.
   - To linearize the transmitter signal, activate the "Square root filter" option.
   - To linearize the transmitter signal characteristic, activate the "Polygon filter" option.
4. Save the entered parameters with "Project > Save".

Signal filter

Filter 1. order, whose time behavior is set through the time constants.

Square root filter

Serves for the linearization of the transmitter signals where the actual value exists as a physical size, which is in a quadratic connection with the measured process size.

Polygon filter

Serves for the linearization of transmitter characteristics.

The 13 optional supporting points can be selected in:

- mA for current input
- mV for voltage input

Scaling

Procedure

To change the input signal into a physical unit, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Analog Input > Input (No.) > Scaling".
3. Set the following parameter:
   - High value
   - Low value
4. Save the entered parameters with "Project > Save".

Scaling

For the transfer of the input signal into a different physical unit through linear interpolation between the initial value (low value) and final value (high value).

Controller difference

Realizing fixed value or cascade controller

Fixed value or cascade controller

For the fixed value or cascade controller structures, the effective actual value is identical with the actual value A.

Select controller type

To realize a fixed value or cascade controller, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.)".
3. Select the option "Fixed value or cascade controller" in the "Controller type" area.
4. Save the entered parameters with "Project > Save".

Prepare setpoint

To prepare the setpoint, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller difference".
3. Set the corresponding parameters in the different areas.
   - Setpoint source
     Select the source of the setpoint.
   - Safety setpoint
     Set the safety setpoint here. Also set the reaction to a CPU-failure and the start-up behavior.
   - Setpoint ramp
     Set the start-up time.
   - Setpoint limit
     Select the high and low setpoint limits.
4. Save the entered parameters with "Project > Save".

Select signal source for actual value A

To select the signal source for the actual value A, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller difference".
3. Select the corresponding source in the "Manipulated value" area. The following selections are available for selection:
   - Zero
   - Prepared analog input
4. Save the entered parameters with "Project > Save".

Realizing three component controller

Three component controller

For the three component controller structure, the effective actual value is formed by adding the three actual values A, B and C and through adding an adjustable offset. The actual values B and C can be additionally evaluated through factors.

Select controller type

Proceed as follows to implement a three-term controller:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.)".

3. Select the option "Three component controller" in the "Controller type" area.
4. Save the entered parameters with "Project > Save".

Prepare setpoint

To set how the setpoint should be prepared, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller difference".
3. In the "Setpoint source" area, select from which source the setpoint should be referenced.
4. Enter a value in the "safety setpoint" area that should serve as a safety value in the case of a CPU-error and during start-up.
5. Select the behavior of the controller module that should apply as "reaction to CPU error" and for "start-up behavior". The selection consists of:
   – Setpoint = last valid setpoint: upon the occurrence of the event, the setpoint from the last valid setpoint will be assigned. The last valid setpoint remains constant.
   – Setpoint = safety setpoint: upon the occurrence of the event, the setpoint will be assigned the safety setpoint. In this way, a secured emergency mode can be realized.
6. In the "Setpoint ramp" area, set the rise time that should be held with a change of the setpoint in order to reduce an overshooting.
7. In the "setpoint limit" area, set both an high and low limit for the setpoint.
8. Save the entered parameters with "Project > Save".

Select signal source for actual value A, B and C

To set the signal sources for the actual values A, B and C, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller difference".
3. In the "Manipulated value" area, select the respective signal source for each actual value. Here you can select if you want to use no signal ("zero") as a source or if you want to use prepared analog inputs.
4. Save the entered parameters with "Project > Save".

Enter factors for actual values B and C and offset

To set the signal sources for the actual values A, B and C, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller difference".
3. In the "Add" area, set the required weight factor for the actual values B and C.
4. Set the required "offset" that should be added to the actual values.
5. Save the entered parameters with "Project > Save".

Realizing ratio or mixture controller

Select controller type

Proceed as follows to implement a ratio or blending controller:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.)".
3. Select the option "Ratio or Mixture controller" in the "Controller type" area.
4. Save the entered parameters with "Project > Save".

Prepare setpoint

To set how the setpoint should be prepared, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller difference".
3. In the "Setpoint source" area, select from which source the setpoint should be referenced.
4. Enter a value in the "safety setpoint" area that should serve as a safety value in the case of a CPU-error and during start-up.
5. Select the behavior of the controller module that should apply as "reaction to CPU error" and for "start-up behavior". The selection consists of:
   - Setpoint = last valid setpoint: upon the occurrence of the event, the setpoint from the last valid setpoint will be assigned. The last valid setpoint remains constant.
   - Setpoint = safety setpoint: upon the occurrence of the event, the setpoint will be assigned the safety setpoint. In this way, a secured emergency mode can be realized.
6. In the "Setpoint ramp" area, set the rise time that should be held with a change of the setpoint in order to reduce an overshooting.
7. In the "setpoint limit" area, set both an high and low limit for the setpoint.
8. Save the entered parameters with "Project > Save".

Select signal source for actual value A and D

To set the signal sources for the actual values A and D, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller difference".
3. In the "Manipulated value" area, select the respective signal source for each actual value. Here you can select if you want to use no signal ("zero") as a source or if you want to use prepared analog inputs, for the actual value D you can select the manipulated value from one of the controllers.
4. Save the entered parameters with "Project > Save".

Enter multiplier

To set the signal sources for the actual values A and D, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller difference".
3. Select the desired offset in the "Multiply" area.
4. Save the entered parameters with "Project > Save".

For ratio controllers, the actual value A is used as the controller size, the actual value D as the ratio size. The setpoint input serves as the ratio factor. Through multiplication with the actual value D and addition of an adjustable offset, an effective setpoint is prepared. If the actual value D is turned off, then only the offset will be added to the setpoint.

Setting alarm generation

Procedure

Proceed as follows to set interrupt generation:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller difference".
3. Select the required options in the "alarm generation" area:
   - At "Limit monitor for", select for which signal the alarm should be set.
   - Under "Alarm high limit" you can set from which value the alarm should be activated upon reaching the high limit.
   - Under "Warning high limit" you can set from which value an alarm should be activated upon reaching the high limit.
   - Under "Warning low limit" you can set from which value an alarm should be activated upon reaching the low limit.
   - Under "Alarm low limit" you can set from which value the alarm should be activated upon reaching the low limit.
   - To avoid flickering on the monitoring displays, a hysteresis can be set under "Hysteresis".
4. Save the entered parameters with "Project > Save".

Controlling algorithm

Realizing P-control

Procedure

To realize a P-control, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller algorithm".
3. Select the option "PID-controller" in the "Type" area.
4. Select the option "P" in the "General" area under "Mode".
   All options that can be set for a P-control are activated, all other options are disabled.
5. Save the entered parameters with "Project > Save".

Realizing PI-control

Procedure

To realize a PI-control, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller algorithm".
3. Select the option "PID-controller" in the "Type" area.
4. Select the option "PI" in the "General" area under "Mode".
   All options that can be set for a PI-control are activated, all other options are disabled.
5. Save the entered parameters with "Project > Save".

Realizing I-control

Procedure

To realize a I-control, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller algorithm".
3. Select the option "PID-controller" in the "Type" area.
4. Select the option "I" in the "General" area under "Mode".
   All options that can be set for a I-control are activated, all other options are disabled.
5. Save the entered parameters with "Project > Save".

Realizing PID-control

Procedure

To realize a PID-control, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller algorithm".
3. Select the option "PID-controller" in the "Type" area.
4. Select the option "PID" in the "General" area under "Mode".
   All options that can be set for a PID-control are activated, all other options are disabled.
5. Save the entered parameters with "Project > Save".

Realizing PD-control

Procedure

To realize a PD-control, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller algorithm".
3. Select the option "PID-controller" in the "Type" area.
4. Select the option "PD" in the "General" area under "Mode".
   All options that can be set for a PD-control are activated, all other options are disabled.
5. Save the entered parameters with "Project > Save".

Realizing fuzzy temperature controller

Procedure

To implement a fuzzy temperature control, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller algorithm".
3. Select the option "Fuzzy temperature controller module" in the "Type" area.
   All options that can be set for a fuzzy temperature controller module are activated, all other options are disabled.
4. Save the entered parameters with "Project > Save".

Suppressing the noise of the controller difference

Procedure

To suppress the sound of the controller difference, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller algorithm".
3. Set the "Dead zone width" setting in the "General" area.
4. Save the entered parameters with "Project > Save".

Turning on disturbance factor

Procedure

To switch on the disturbance variable, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller algorithm".
3. In the "General" area, select the source for the disturbance variable from the "Disturbance variable source" list.
   Select the entry "Zero" in order not to switch on a disturbance variable.
4. Under "Disturbance variable factor", enter a value with which you want to weigh the disturbance variable.
   A value < 0 weakens the disturbance variable, a value > 0 strengthens the disturbance variable. With the value 1, the disturbance variable remains unchanged.
5. Save the entered parameters with "Project > Save".

Place P-component in feedback path

Procedure

To realize a fixed value or cascade controller, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller algorithm".
3. Select the option "In return" in the "P-component" area.
   If the option is activated, the P-component is placed in return.
4. Save the entered parameters with "Project > Save".

Weaken D-component for setpoint changes

Select controller type

To realize a fixed value or cascade controller, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller algorithm".
3. In the "D-component" area, select the source for the D-component with the "D-component" option.
4. Set the factor for the D-component in the "Difference factor" area.
5. Save the entered parameters with "Project > Save".

Reverse controller effect

Principle

Reversing the controller from the assignment "increasing controller difference = increasing manipulated variable" to "increasing controller difference = decreasing manipulated variable" can be done by setting a negative proportion value (GAIN).

Procedure

To reverse the controller effect, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller algorithm".
3. Set a proportion value with the reverse prefix in the "General" area.
   If a positive proportion value is specified and the controller effect should be reversed, set a negative proportional value.
4. Save the entered parameters with "Project > Save".

Controller output

Implementing a continuous controller

A continuous controller can be created only with an analog module.

Procedure

1. Switching external manipulated value (Page 9848)
2. Setting safety manipulated value (Page 9849)
3. Switching tracing (Page 9848)
4. Limiting manipulated value (Page 9849)

5. Controlling two actuators - split range (Page 9850)
6. Selecting output signal and signal type (Page 9851)

Realizing a step controller without position feedback

A step controller without position feedback can be realized only with a digital module.

Procedure

1. Select controller output step controller without position feedback
2. Switching external manipulated value (Page 9848)
3. Setting safety manipulated value (Page 9849)
4. Pulse shaper step controller (Page 9850)

Realizing step-by-step controller with position feedback

A step-by-step controller with position feedback can only be realized with a digital module.

Procedure

1. Select controller output step-by-step controller with position feedback
2. Switching external manipulated value (Page 9848)
3. Switching tracing (Page 9848)
4. Setting safety manipulated value (Page 9849)
5. Limiting manipulated value (Page 9849)
6. Setting position feedback (Page 9848)
7. Pulse shaper step controller (Page 9850)

Realizing pulse controller

An pulse controller can only be realized with a digital module.

Procedure

1. Select controller output pulse controller
2. Switching external manipulated value (Page 9848)
3. Switching tracing (Page 9848)
4. Setting safety manipulated value (Page 9849)
5. Limiting manipulated value (Page 9849)
6. Prepare analog signal for conversion to a digital signal
7. Converting analog signal into a digital signal (Page 9851)

Setting functions of the controller output

Switching external manipulated value

Procedure

To switch the external manipulated value, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller output".
3. In the "External Manipulated value" area, select the option through which the external manipulated value should be activated. Here you can choose from the following options:
   – Activation of function block.
   – Activation of a function block or from one of the digital inputs.
4. Save the entered parameters with "Project > Save".

Switching tracing

Procedure

To realize a fixed value or cascade controller, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller output".
3. In the "Tracing" area, select the option through which the external manipulated value should be activated. Here you can choose from the following options:
   – Activation of function block.
   – Activation of a function block or from one of the digital inputs.
4. In the "tracing" area, select the source of the signal for the tracing. Here you can choose from the following options:
   – Zero, no source is used.
   – One of the analog inputs.
5. Save the entered parameters with "Project > Save".

Setting position feedback

Procedure

To set the position feedback, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Digital output > Output (No.)".

3. In the "Position feedback input", select the "Source" option in order to select the signal for the position feedback.
   - Zero: no position feedback.
   - One of the prepared analog inputs.
4. Save the entered parameters with "Project > Save".

Setting safety manipulated value

Procedure

To set the safety manipulated value, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller output".
3. In the "Safety manipulated value", set the value that should be used as the safety manipulated value.
4. Under "Activation", select the option through which the external manipulated value should be activated. Here you can choose from the following options:
   - Activation of function block.
   - Activation of a function block or from one of the digital inputs.
5. Under "Reaction upon startup of the module", select the option that should be executed upon starting up the module. Here you can choose from the following options:
   - Control mode: sets the module immediately into control mode upon startup.
   - Safety manipulated value: initially, the safety manipulated value is controlled.
6. Under "Reaction upon fault of sensor actual value A", select the option that should be executed if there is a fault of sensor actual value A.
7. Under "Reaction upon transducer disruption of an input", select the option that should be executed if there is a transducer disruption of an input.
8. Save the entered parameters with "Project > Save".

Limiting manipulated value

Procedure

To limit the manipulated value, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller output".

3. In the "Output signal limit", set the following value:
   - High value: illustrates the high limit for the output signal.
   - Low value: illustrates the low limit for the output signal.
4. Save the entered parameters with "Project > Save".

Controlling two actuators - split range

Procedure

To control two actuators, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Controller > Channel (No.) > Controller output".
3. In the "Split Range" area, select the "Activate" option to control two actuators.
4. Set the following value for the input and output signals from A and also for the input and output signals from B:
   - Low input signal.
   - High input signal.
   - Low output signal.
   - High output signal.
5. Save the entered parameters with "Project > Save".

Pulse shaper step controller

Procedure

To set the pulse shaper, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Digital output > Output (No.)".
3. Set the following parameter in the "Pulse shaper" area:
   - Motor manipulating time.
   - Minimum impulse duration.
   - Minimum break duration.
4. Select the fitting option for the high and low value as to where the value should be taken over from:
   - From the function block.
   - From the function block or from one of the digital inputs.
5. Save the entered parameters with "Project > Save".

Converting analog signal into a digital signal

Procedure

To convert an analog signal into a digital signal, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Digital output > Channel (No.)".
3. Set the following parameter in the "Pulse shaper" area:
   - Minimum impulse duration.
   - Minimum break duration.
4. Save the entered parameters with "Project > Save".

Selecting output signal and signal type

Procedure

To select the output signal and the signal type, proceed as follows:

1. Double click in the device view under the "Parameter setting" module.
2. Navigate in the project view to "Analog output > Output (No.)".
3. Under the "Source" option, select which signal should be used as the output signal. You can choose from the following:
   - Zero: no source for the output signal.
   - One of the analog inputs.
   - One of the manipulated values for controller A or B.
4. Under the "Signal type" option, select the type in which the output signal should be emitted.
5. Save the entered parameters with "Project > Save".

Programming FM 355

Commissioning the FM 355

Creating and supplying instance DBs

Procedure

To meet the requirements for commissioning, follow these steps:

1. Create a user program that calls the PID_FM instruction cyclically in a cyclic interrupt OB.
2. Set READ_VAR = TRUE cyclically in the user program.

3. Create a single-instance DB of the PID_FM instruction for each controller channel.
4. Enter the module address for each instance DB in the PID_FM.MOD_ADDR parameter. Use the start address that you specified when you configured your hardware as the module address of the FM 355.
5. Enter the number of the controller channel (1 ... 4) in the PID_FM.CHANNEL parameter.
6. If you use the PID_PAR instruction, enter the number of the controller channel in the PID_PAR.CHANNEL parameter.
7. Check to ensure that the "Sensor type analog input" of this channel is selected.
   If the analog input is not processed, all PID parameters are set to zero and a manipulated variable is not calculated.

Note

Multi-instance DBs

If you use multi-instance DBs in your project, you must commission the FM via a watch table.

See also

Description PID_FM (Page 7185)

Description PID_PAR (Page 7212)

Commissioning the controller

Introduction

You tune the PID parameters as follows:

- PID/PI controllers (step controller, continuous controller, pulse controller): Manually
- Fuzzy temperature controller: Automatically
- Step controller: Automatically with instruction TUN_ES
- Continuous controller: Automatically with instruction TUN_EC
- Pulse controller: Automatically with instruction TUN_EC

Procedure

Proceed as follows to tune a controller:

1. Put the CPU in RUN mode.
2. If you are using a step controller, measure the motor actuating time.
3. Tune the PID parameters with one of the options listed above.
4. Monitor the control loop with the trend view.

See also

Measuring the motor transition time (Page 9859)
Description TUN_EC (Page 6861)
Description TUN_EC (Page 6842)

Tuning PID parameters manually

Requirement

You have selected the "PID controller" control algorithm during parameter assignment.

Procedure

To set a PID/PI controller manually, proceed as follows:

1. Select the instance DB of the channel you want to tune.
2. Click the "Start" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value, and manipulated variable are recorded.
3. Enter values for the proportional action, integral action, derivative action and delay.
4. Click the icon "Send PID parameters to FM" in the "Tuning" group.
5. Select the "Change setpoint" check box in the "Current values" group.
6. Enter a new setpoint.
7. Clear the check box "Manual mode", if necessary, and click the icon in the "Current values" group.
   The controller works with the new parameters and controls the new setpoint.
8. Check the quality of the PID parameters based on the trend characteristics.
9. Repeat steps 3 to 8 until you are satisfied with the controller results.

Fuzzy system identification

Requirements on process with fuzzy temperature controller

Requirement

The process should fulfill the following requirements in order to achieve optimal control using the fuzzy temperature controller:

- At bath heatings the liquid to be heated has to be mixed thoroughly.
- In case of bath-in-bath control systems both liquids have to be mixed thoroughly. At the same time good heat transitions between all the heat-transferring media have to be ensured. In the case of materials with poor heat-transferring properties large transfer surfaces should ensure good heat transportation.

- In case of room temperature control systems thorough mixing (for example with fans) has to be ensured.
- The controlling system gain may not exceed the factor of 3.
- The delay time may not exceed 3% of the recovery time.
- The temperature to be controlled should change by a maximum of 1‰ of the specified maximum temperature at the maximum manipulated value output within the sampling time of the controller.

Classification of the Controlled Systems

A controlled system or a process to be controlled is characterized by parameters such as the heat output, the heating mass or the heating capacity of the medium to be heated. With regard to the fuzzy controller a difference is made between "critical" and "non-critical" temperature controlled systems as follows: The control system becomes increasingly critical:

- The greater the heat output,
- the greater the heating capacity of the heating,
- The lower the heating capacity of the medium to be heated,
- the greater the heat transition resistance,
- the smaller the heat transfer surface.

After a manipulated value step change has been applied to the controlled system, it reacts with a step response. The controlled system can also be classified on the basis of this step response: The control system becomes increasingly critical the greater the ratio t_u / t_g is and the greater the controlled system gain is. With t_u / t_g < 1/10 you have a non-critical control section.

Process of fuzzy system identification

Process of parameter tuning with fuzzy temperature controller

Parameter optimization is based on a self-optimizing fuzzy controller.

The identification of the controlled system begins with a monitoring phase during which no heat output takes place. The duration of the monitoring phase is as follows:

- Monitoring phase continuous controller: Approx. 1 min
- Monitoring phase step controller: Approx. 1 min + 1/2 x actuating time of final control element

This time is used in order to determine temperature trends in the heating medium. Afterwards the maximum heating output of 100% is output. This is visible at the LMN output parameter in the instance DB of the PID_FM instruction.

The range of the first 4% of the temperature increase of the setpoint value range is used for the identification, whereby information about the process response is obtained at 1% and 4% respectively of the temperature increase on the basis of the time that has passed.

The identification is complete when the heating output becomes less than 100%. This behavior can be used, for example, to deactivate the identification via the user program. After the identification phase the controller continues to operate with the determined parameters.

If the setpoint value step change is not adjusted and if the heating output remains permanently at zero, the identification has been terminated unsuccessfully, meaning that the controller cannot control the connected controlled system.

Controller behavior with different control sections

Problems do not arise in case of an "uncritical" controlled system, neither during identification nor during controlling.

The identification of a controlled system that is "too critical" is cancelled. Controlling of an identified "critical" controlled system is carried out very "carefully" or slowly.

Performing fuzzy system identification

Requirement

You have selected the "Fuzzy controller" control algorithm during parameter assignment.

You have to ensure as far as possible that the controlled system has settled (no heating-up or cooling-down process) or is changing slowly and monotonously before the identification is carried out. The criterion can be that the temperature change is to approach a straight during a period of one minute. At faster processes this requirement is particularly relevant.

Since the manipulated variable zero is output for approx. one minute by the controller at the beginning of identification, the temperature to be controlled has to lie near the ambient temperature.

Starting fuzzy system identification

Follow these steps to tune a fuzzy controller:

1. Select the instance DB of the channel you want to tune.
2. Click the "Start" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value, and manipulated variable are recorded.
3. Enter a new setpoint (SP_OP) in the "Tuning" group.
   The new setpoint must meet the following conditions:
   - Setpoint step change > 5%
   - New setpoint > process value + setpoint high limit x 0.12
4. Click "Fuzzy process identification" in the "Tuning" group.

Result

The new setpoint (SP_OP), SP_OP_ON = TRUE and FUZID_ON = TRUE are sent to the FM. Starting the fuzzy system identification. The IDSTATUS parameter of the CH_DIAG instruction provides information on the identification state. You see in the status display field if a fuzzy system identification is activated or running.

Deactivating fuzzy system identification

As long as the bit FUZID_ON = TRUE, the next identification is started whenever the setpoint step change is sufficiently large.

Proceed as follows to turn off the tuning mode:

1. Enter FUZID_ON in a watch table.
2. Set FUZID_ON = FALSE.

Note

Deactivation of the tuning mode FUZID_ON = FALSE before identification has been completed does not stop the identification. An identification process that has been started continues to run under all circumstances – with the exception of a negative setpoint value step change.

Repeating Fuzzy system identification

A renewed identification is carried out with each additional setpoint step change ≥12% unless the identification has been deactivated again with FUZID_ON = FALSE.

It is also possible to restart the identification by reducing and then increasing the setpoint by 12> %.

Canceling fuzzy system identification

Identification can be canceled in the following cases:

- By the controller if a "critical" controlled system is identified. After canceling the controller is in the error state. This state is indicated by the fact that the manipulated variable is reset permanently by the controller. This is also not changed by deactivating the optimization. The error state is deleted by starting a new identification.
  Information about the state of the identification is available through the IDSTATUS parameter of the CH_DIAG FB. More information on this can be found at the end of this section under "Controller State Information".
- By the operator by generating a negative setpoint value step change, whereby the setpoint value must lie below that of Condition 2 (see above "How To Start identification").

Proceed as follows to cancel tuning:

1. Enter a new setpoint that is less than the current process value and meets the same conditions as for the start of fuzzy system identification.
2. Click "Fuzzy process identification" in the "Tuning" group.

See also

Parameter IDSTATUS (Page 7210)

Downloading tuned parameters in device

Introduction

You can proceed as follows to prevent the parameters on the FM from being overwritten with the system data the next time you start up the CPU:

- You download the tuned PID parameters from the instance DB of the PID_FM to the FM after each startup of the FM with LOAD_PAR=TRUE.
- Download the PID parameters to the SDB.

Note

The PI, PID and SAVE parameter records cannot be stored in the SDB.

Procedure

Proceed as follows to download the parameters to the SDB:

1. Click on the icon "Transfer online data to an offline project".
   The PID parameters are saved in the parameter setting of the FM.
2. Stop CPU.
3. Click "Download to device".

Working with the trend view

Monitoring

1. Double click in the device view under the "Commissioning" module.
2. Navigate in the project view to "Trend view".
3. Select the instance DB you wish to monitor. The channel to which the instance DB has been assigned will be displayed.
4. Specify the update cycle in which the trend view is to record.
5. Click on "Measurement on" to start measuring.
6. Click on "Measurement off" to stop measuring.

Procedure

To adjust the scale, proceed as follows:

1. Double click on "Commissioning" in the device view of the module.
2. Navigate to "Trend view" in the configuration view.

3. To change the time axis:
   - Click on the arrow-button next to the date to open a calendar. Select a date from the calendar. To lock the selection, click on the blue lock. This possibility exists for the beginning and end dates.
   - Click on the time scale. The mouse pointer will be illustrated as a left/right arrow. Press the mouse and drag it to the left or right.
     If the beginning or end time is locked, the time scaling will change.
     If neither the beginning nor the end time is locked, the specified period will change, the scaling remains the same.
4. To change the value axis:
   - Click on the input field for the value displayed above or below. Enter your desired value here. To lock the selection, click on the blue lock. This possibility exists for the high and low value.
   - Click on the value scale. The mouse pointer will be illustrated as an up/down arrow. Press the mouse and drag it up or down.
     If the high or low value is locked, the scaling of the value axis will change.
     If neither the high nor the low value is locked, the specified range will change, the scaling remains the same.

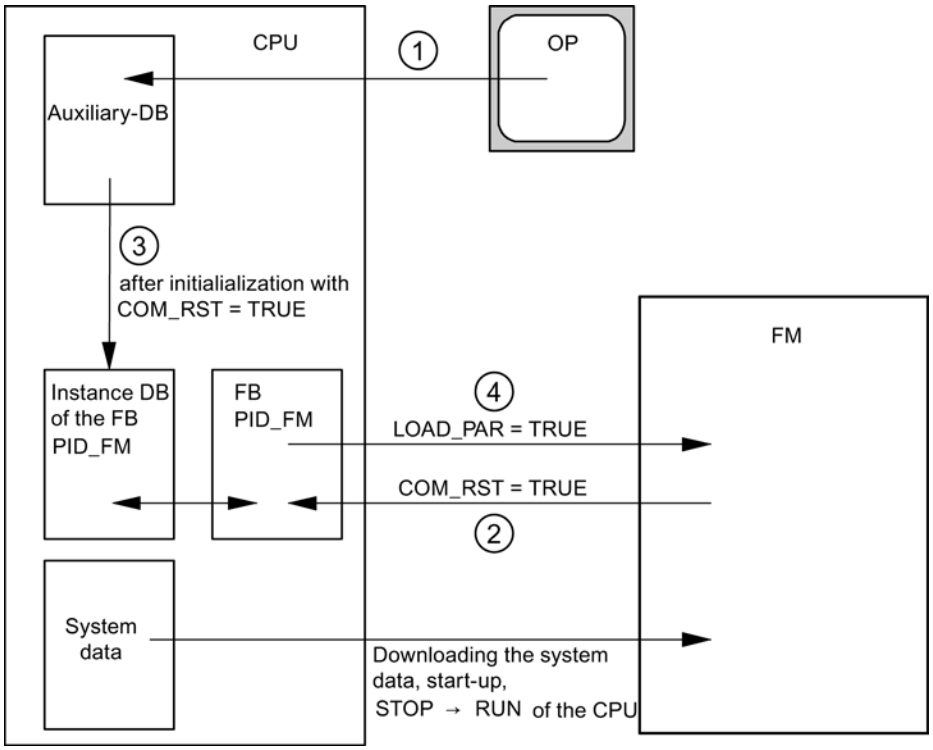
Changing parameters with the HMI

Procedure

Proceed as follows to change the controller parameters of the instruction PID_FM with the HMI:

1. Write the parameter you want to change from the HMI to an auxiliary DB on the CPU (see ①).
2. Set PID_FM.COM_RST to TRUE.
   The instruction performs an initialization run. The parameters are synchronized between FM and CPU (see ②).
3. Transfer the parameters you want to changed from the auxiliary DB to the instance DB of the instruction PID_FM (see ③).
4. Set PID_FM.LOAD_PAR to TRUE.
   The changed parameters are transferred to the FM (see ④).

It is necessary to store the parameters in an auxiliary DB because PID_FM after the start of the CPU with COM_RST = TRUE reads those parameters from the module that the CPU has previously transferred from the system data to the FM.



See also

Mode of operation PID_FM (Page 7187)

Measuring the motor transition time

Introduction

The motor transition time has to be as accurate as possible for good controller results. The data in the actuator documentation contains average values for this type of actuator. The value for the specific actuator used may differ.

You can measure the motor transition time during commissioning if you are using actuators with position feedback or endstop signals. The motor transition time cannot be measured if neither position feedback nor endstop signals are available.

Requirement

- The configured motor transition time must be half the size of the real transition time. The actuator is moved for a maximum of twice the transition time in one direction.
- The position feedback must be interconnected for actuators with analog position feedback.
- The endstop signals must be interconnected for actuators with endstop signals.

Actuators with analog position feedback

Proceed as follows to measure motor transition time with position feedback:

1. Select the instance DB for which you want to measure the motor transition time.
2. Select the "Use position feedback" check box.
3. Enter the position to which the actuator is to be moved in the "Target position" input field. The current position feedback (starting position) will be displayed. Click the "Start transition time measurement" icon.

Result

The actuator is moved from the starting position to the target position. Time measurement starts immediately and ends when the actuator reaches the target position. The motor transition time is calculated according to the following equation:

Motor transition time = (output value high limit – output value low limit) × Measuring time / AMOUNT (target position – starting position).

The progress and status of transition time measurement are displayed. The measured transition time is displayed in the field "Measured transition time". Once transition time measurement is complete, the manual mode of the transition signals is switched on (LMNSOPON = TRUE). The actuator will stop being moved.

Actuators with endstop signals

Proceed as follows to measure motor transition time with endstop signals:

1. Select the instance DB for which you want to measure the motor transition time.
2. Select the "Use actuator endstop signals" check box.
3. Click the "Start transition time measurement" icon.

Result

The actuator is moved first to the low endstop, then to the high endstop and then back to the low endstop. Time measurement will start once the actuator has reached the lower endstop and will end when the actuator reaches the lower endstop for the second time. The motor transition time is equal to the time measured divided by two.

The progress and status of transition time measurement are displayed. The measured transition time is displayed in the field "Measured transition time". Once transition time measurement is complete, the manual mode of the transition signals is switched on (LMNSOPON = TRUE). The actuator will stop being moved.

Loading measured transition time

1. Click "Transfer online data to an offline project" to save the measured motor transition time locally in the project.
   The measured transition time is saved in the parameter setting of the FM.
2. Click "Download to device" to activate all changes the next time you start the CPU.

Cancelling transition time measurement

When you cancel the transition time measurement, the manual mode of the actuator signals is switched on (LMNSOPON = TRUE) and the manual mode with manipulated value specification is switched off (LMNOP_ON = FALSE). The actuator will stop being moved.

See also

In/out parameters PID_FM (Page 7192)

FM 355: Diagnostics with LED display

Error display from the group error LED

Introduction

If the red group error LED lights up, there is either an error on the module (internal error) or in the line connections (external error).

If the yellow LED flashes, then the firmware has been deleted. This status can only occur in the case of faulty hardware or if the loading procedure of the firmware is aborted.

Which errors are displayed?

The following errors are displayed by the group error LED lighting up:

Type of error	Diagnostic message	Possible cause	Correction
Internal errors	Module defective	Hardware error	Replace the module
	Time watchdog tripped	Hardware error	Replace the module
	EEPROM content is invalid	Failure of the supply voltage when configuring	Reconfigure module
External errors	Incorrect parameters in module	Incorrect parameters have been transferred to the module	Reassign the module parameters
	Errors with the analog inputs or analog outputs	Analog input hardware error	Replace module
		Analog input wire break	Remedy wire break
		Analog input measuring range violation (underrange)	Check measuring signal
		Analog input measuring range violation (overrange)	Check the measuring signal
		Analog output wire break	Remedy wire break
		Analog output short circuit	Eliminate short circuit
	Missing external auxiliary supply	24 V supply missing	Restore 24 V supply

Diagnostic interrupt in the case of errors

All errors can trigger a diagnostic interrupt if you have enabled the diagnostic interrupt in the respective parameterization screen. From the diagnostic data records DS0 and DS1 you can see which errors have caused the LED to light up. The assignment of the diagnostic data records DS0 and DS1 is described in the next section.

Triggering diagnostic interrupts

What is a diagnostic interrupt?

If the user program reacts to an internal or external error, you can configure a diagnostic interrupt that interrupts the cyclic program of the CPU and calls the diagnostic interrupt OB, OB 82.

Which events can trigger a diagnostic interrupt?

The list shows which events can trigger a diagnostic interrupt:

- The module has not been configured or is incorrectly configured
- Module defective
- Wire break with analog inputs (only 4 to 20 mA)
- High limit violated and low limit violated with analog inputs
- Load break and short circuit with analog outputs

Default setting

The diagnostic interrupt is blocked by default.

Enabling the diagnostic interrupt

In the "Basic parameters" screen you can block or enable the diagnostic interrupt for the module.

Responses to an interrupt-triggering event

The following happens when an event occurs that could trigger a diagnostic interrupt:

- The diagnostic information is stored in the diagnostic records DS0 and DS1 on the module.
- The group error LED lights up.
- The diagnostic interrupt OB (OB 82) is called.
- The diagnostic record DS0 is entered in the start information of the diagnostic interrupt OB.
- If there is no hardware fault, the module continues to control.

If no OB 82 is programmed, the CPU goes to STOP.

Diagnostic data record DS0 and DS1

The information as to which event has triggered a diagnostic interrupt is stored in the diagnostic data records DS0 and DS1. The diagnostic data record DS0 contains four bytes, the DS1 16 bytes, whereby the first four bytes are identical to those of the DS0.

Diagnostic data records DS0 and DS1

Read data record from the module

The diagnostic data record DS0 is automatically transferred to the start information when the diagnostic OB is called. There these four bytes are stored in the local data (byte 8 - 11) of the OB 82.

The diagnostic data record DS1 (and hence also the content of the DS0) can be read from the module using the "CH_DIAG" instruction. This only makes sense if an error is reported in a channel in the DS0.

The "CH_DIAG" instruction must be called in the same OB as the FB "PID_FM". This is achieved by the following measure: Set a bit while the OB 82 is being executed. Query this bit in the OB in which the FB "PID_FM" is called and then call the "CH_DIAG" instruction with a set bit.

How does the diagnosis text appear in the diagnostics buffer?

If you want to enter the diagnostics message in the diagnostics buffer, call the "WR_REC" instruction "Enter user-specific message in the diagnostics buffer" in the user program. The event number of the respective diagnostics message is specified in the input parameter EVENTN. The interrupt is entered in the diagnostics buffer defined as x = 1 for incoming and x = 0 for outgoing. The diagnostics buffer contains the relevant diagnostics text in the "Meaning" column as well as the time of the entry.

Assignments of the Diagnostics Record DS0 in the Start Information

The following table shows the assignments of the diagnostics record DS0 in the start information. All unlisted bits are not significant and are set to zero.

Table 11-7 Assignments of diagnostics record DS0

Byte	Bit	Meaning	Remark	Event No.
0	0	Module malfunction	Is set at every diagnostics event	8:x:00
	1	Internal fault	Is set at all internal faults: • Watchdog time-out • EEPROM contents invalid. Module starts up without controlling and waits for renewed parameter configuration by the CPU. • EPROM error • ADC/DAC fault • Analog input, hardware fault	8:x:01

Byte	Bit	Meaning	Remark	Event No.
	2	External error	Is set at all external errors: • Missing external auxiliary voltage • Parameter configuration faulty • Analog input, wire breakage (only range 4 to 20 mA) • Analog input, measuring range violation (underrange) • Analog input, measuring range violation (overrange) • Analog output, wire breakage • Analog output, short circuit	8:x:02
	3	Fault in a channel	See DS1, from byte 7, for further breakdown	8:x:03
	4	Missing external auxiliary voltage	24 V power supply of the FM 355 failed	8:x:04
	6	EEPROM contents invalid	Failure of the supply voltage during a writing process to the EEPROM. The module starts up with default parameters.	8:x:03
	7	Parameter configuration faulty	The module cannot use a parameter. Reason: Parameter unknown or impermissible combination of parameters. Refer to the menu **PLC > Parameter Assignment Error**	8:x:07
1	0 ... 3	Module class	Always has 8 assigned	–
	4	Channel-specific diagnostics	Is set if the module can supply additional channel information and if a channel error exists (refer to DS 1 Byte 7 to 12)	–
2	3	Watchdog time-out	Hardware fault	8:x:33
3	2	EPROM error	Module defective	8:x:42
	4	ADC/DAC fault	Module defective	8:x:44

Assignments of Diagnostics Record DS1

The diagnostics record DS1 consists of 16 bytes. The first 4 bytes are identical with diagnostics record DS0. The following table shows the assignment of the remaining bytes. All unlisted bits are not significant and are set to zero.

Table 11-8 Assignment of Bytes 4 to 12 of the diagnostics record DS1

Byte	Bit	Meaning	Remark	Event No.
4	0 ... 7	Channel type	Always has 75H assigned	–
5	0 ... 7	Length of the diagnostic information	Always has 8 assigned	–
6	0 ... 7	Number of channels	Always has 5 assigned (4 controllers + 1 reference channel)	–
7	0 ... 7	Channel error vector	One bit is assigned to each channel	–

Byte	Bit	Meaning	Remark		Event No.
8	0	Analog input, hardware fault	Channel-specific diagnostics channel 1		8:x:B0
	1	Unused			8:x:B1
	2	Analog input wire break (only area 4 to 20 mA)			8:x:B2
	3	Unused			8:x:B3
	4	Analog input below measuring range			8:x:B4
	5	Analog input, measuring range violation (overrange)			8:x:B5
	6	Analog output, wire breakage		Only for current output of continuous controller	8:x:B6
	7	Analog output, short circuit		Only with the current output of the C controller	8:x:B7
9	0 ... 7	See Byte 8	Channel-specific diagnostics channel 2		See above
10	0 ... 7	See Byte 8	Channel-specific diagnostics channel 3		See above
11	0 ... 7	See Byte 8	Channel-specific diagnostics channel 4		See above
12	0 ... 5	See Byte 8	Diagnostics for reference channel		See above

Additional information

For more information on the "RET_VALU" parameter, refer to List of RET_VALU messages (Page 9866) and the "Extended instructions" section of the help system.

Measuring transducer fault

Faults at Measuring Transducers

The following measuring transducer faults can be recognized by the controller module:

- Measuring range violation (underrange)
- Measuring range violation (overrange)
- Wiring breakage (not at all measuring ranges)

If one of these faults occurs, the group error bit "External error" is set in the diagnostics record DS0 and the channel-specific error bits in the diagnostics record DS1 (refer to the tables in the previous section). When these faults disappear, the corresponding bits are reset.

The following table shows at which limits in the individual measuring ranges the error bits are set and reset:

Measuring range	Error bit measuring range violation (underrange) at ...	Error bit measuring range violation (overrange) at ...	Error bit wire breakage indication
	DS1: Byte 10 to 26, Bit 4	DS1: Byte 10 to 26, Bit 5	DS1: Byte 10 to 26, Bit 2
0 to 20 mA	< –3.5 mA	> 23.5 mA	–
4 to 20 mA	Error bit = 1 at < 3.6 mA Error bit = 0 at < 3.8 mA	> 22.8 mA	Error bit = 1 at < 3.6 mA Error bit = 0 at > 3.8 mA
0 V to 10 V	<–1.175 V	> 11.75 V	–
Pt 100 (–200 to 850°C) (–328 to 1562°F)	< 30.82 mV	> 650.46 mV	–
Pt 100 (–200 to 556°C) (–328 to 1032°F)	< 30.82 mV	> 499.06 mV	–
Pt 100 (–200 to 130°C) (–328 to 264°F)	< 30.82 mV	> 254.12 mV	–
Thermocouple Type B	< 0 mV	> 13.81 mV	–
Thermocouple Type J	< –8.1 mV	> 69.54 mV	–
Thermocouple Type K	< –6.45 mV	> 54.88 mV	–
Thermocouple Type R	< –0.23 mV	> 21.11 mV	–
Thermocouple Type S	< –0.24 mV	> 18.7 mV	–
Free thermocouple	< Low input value of the polyline	< High input value of the polyline	–

List of RET_VALU messages

RET_VALU messages

JOB_ERR (Hex)	JOB_ERR (Dec)	JOB_ERR (Int)	Meaning
7000	28672	-32624	First call with REQ=0: no data transmission active; BUSY has the value 0.
7001	28673	-32624	First call with REQ=1: data transmission initiated; BUSY has the value 1.
7002	28674	-32624	Interim call (REQ irrelevant). Data transfer already active; BUSY has the value 1.
8090	32912	-32624	Specified logical base address invalid: There is no assignment in the SDB1/SDB2x, or it is not a base address.
80A0	32928	-32608	Negative acknowledgment when reading from the module. Module was removed during the read operation or the module is defective.
80A1	32929	-32607	Negative acknowledgment when writing to the module. Module was removed during the write operation or the module is defective.
80A2	32930	-32606	Protocol error at layer 2

JOB_ERR (Hex)	JOB_ERR (Dec)	JOB_ERR (Int)	Meaning
80A3	32931	-32605	Protocol error involving user interface/user
80A4	32932	-32604	Communication bus error
80B1	32945	-32591	Incorrect length specification. FM_TYPE parameter in channel DB not set correctly for the module in use.
80B2	32946	-32590	The configured slot is not being used.
80B3	32947	-32589	Actual module type is not match configured module type.
80C0	32960	-32576	Module data not ready for reading.
80C1	32961	-32575	Data of a write job of the same type have not yet been processed by the module.
80C2	32962	-32574	The module is currently processing the maximum possible number of jobs.
80C3	32963	-32573	Required resources (memory etc.) currently occupied.
80C4	32964	-32572	Communication error
80C5	32965	-32571	Distributed I/O not available.
80C6	32966	-32570	Priority class abort (restart or background).
8522	34082	-31454	Channel DB or parameter DB too short. The data cannot be read off the DB. (Write job)
8532	34098	-31438	DB number of the parameter DB too high (Write job)
853A	34106	-31430	Parameter DB not present. (write job)
8544	34116	-31420	Error at n-th (n > 1) read access to a DB after an error has occurred. (write job)
8723	34595	-30941	Channel DB or parameter DB too short. The data cannot be written to the DB. (Read job)
8730	34608	-30928	Parameter DB in the CPU is write protected. The data cannot be written to the DB (read job)
8732	34610	-30926	DB number of the parameter DB too high (Read job)
873A	34618	-30918	Parameter DB does not exist. (read job)
8745	34629	-30907	Error at n-th (n > 1) write access to a DB after an error has occurred. (Read job)
80ff	33023	-32513	Incorrect index specification with block FMT_PAR
Errors 80A2..80A4 and 80Cx are temporary, i.e., they can be rectified automatically after a waiting time. Messages of the 7xxx form indicate temporary operating states of communication.			

Example projects for FM 355

11.3.5.6 Using FM 355-2

Configuring FM 355-2

Configuring the hardware

Procedure

Configuration assumes that you have set up a project in which you can save the parameter assignment. The following briefly explains only the most important steps.

1. Start the TIA portal and open the device configuration.
2. Select one module rack.
3. Select one controller module from your module catalog.
4. Move the controller module to the appropriate place on the module rack.
5. From the configuration table, note the input address of the module, for example 272.
   The value that you read is displayed in decimal format.

Parameter assignment FM 355-2

Module parameters

Procedure

When assigning parameters you set the module parameters.

1. Double-click on the "Parameter setting" in the Device view of the module. In the Inspector window, select the "Properties" tab.
   Result: The properties are displayed in the inspector window.
2. Configure the basic parameters and the remaining parameters of the module.
3. Configure the remaining device parameters in the Design view.
4. Save the parameters entered with "Project> Save".
5. Translate the project with "Edit > Translate".
6. Load the configuration data with the CPU in STOP mode by selecting "Online > Device load..."
   Result: The data stored in the memory of the CPU are then transferred directly to the module.
7. Carry out a CPU start-up.

Analog input parameter

Evaluation of analog inputs

Procedure

To select a sensor type, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Digital input > input [No]> Evaluation of analog inputs".
3. Enter the following parameters:
   - Sensor type analog input
     For different sensor types the option "open-circuit monitoring" will be enabled. The option may be activated depending on requirements.
   - Junction temperature
4. Save the parameters entered with "Project> Save".

Filter functions

Procedure

To set the filter functions, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Analog input > input [No]> Filter functions".
3. Enter the following parameters:
   - To filter on a time constant you should enable the "signal filter" option.
     If the "signal filter" option is selected, the time constant filter can be set.
   - Filter time constant
     Can only be set if the "signal filter" option is enabled.
   - To linearize the encoder signal you should activate the "square root filter" option.
   - To linearize the encoder signal characteristics you should activate the "polygon filter" option.
4. Save the parameters entered with "Project> Save".

Signal filter

Filter 1. order, whose time behavior is set through the time constants.

Square root filter

Used to linearize encoder signals in which the actual value is present as a physical quantity, which is square in relation to the measured process variable.

Polygon filter

Used to linearize encoder characteristics.

The 13 selectable points are selectable in:

- mA at current input
- mV with voltage input

Scaling

Procedure

To convert the input signal into a physical unit, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Analog input > input [No]> Scaling".
3. Enter the following parameters:
   - High value
   - Low value
4. Save the parameters entered with "Project> Save".

Scaling

To convert the input signal into another physical unit through linear interpolation between the initial value (low value) and the ending value (high value).

Deviation

Implementing a constant or a cascade controller

Constant or cascade controller

In the controller structure of constant or cascade controllers the effective current value is equal to the actual value A.

Selecting the controller type

To implement a constant or cascade controller, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]".
3. In the "Encoder type" section select the "Constant or Cascade controller".
4. Save the parameters entered with "Project> Save".

Processing setpoints

To edit setpoints, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> Channel [No]> Deviation".
3. Set in the different areas the corresponding parameters:
   - Setpoint source
     Select the source of the setpoint.
   - Safety setpoint
     Set your safety setpoints here. Set also the response to a CPU fault and the startup behavior.
   - Setpoint ramp
     Set the start-up time.
   - Setpoint limit
     Select the high and low setpoint limits.
4. Save the parameters entered with "Project> Save".

Selecting a signal source for actual value A

To select the signal source for the actual value A, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> Channel [No]> Deviation".
3. In the "Control value" section select the corresponding source. The following choices are available:
   - Zero
   - Processed analog input
4. Save the parameters entered with "Project> Save".

Implementing three component controllers

Three component controller

In the controller structure of a three component controller the effective actual value is obtained by summing the three actual values of A, B and C and then adding an adjustable offset. The actual values of B and C can also be evaluated via factors.

Selecting the controller type

Proceed as follows to implement a three-term controller:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]".

3. In the "Controller type" section select the "Constant or Cascade controller" option.
4. Save the parameters entered with "Project> Save".

Processing setpoints

To adjust the setpoint processing mode, proceed as follows:

1. Double-click on "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> Channel [No]> Deviation".
3. In the "Setpoint source" section select the source from which the setpoint shall be obtained.
4. In the "Safety setpoint" section enter a value to be utilized upon CPU fault and start-up as a safety value.
5. Select the behavior of the controller module considering the values of the "Reaction to CPU errors" and the "Start-up behavior" options. You may select amongst the following options:
   – Setpoint = last valid setpoint: the setpoint will be assigned the last valid setpoint upon occurrence of the event. The last valid setpoint will remain unchanged.
   – Setpoint = safety setpoint: the setpoint will be assigned the safety setpoint upon the occurrence of the event. Thus, a secured emergency operation can be performed.
6. Set in the "setpoint ramp" the rise time which shall be met upon a setpoint change to reduce overshoot.
7. In the "Setpoint limits" section set both the high and a low limits for the setpoint.
8. Save the parameters entered with "Project > Save".

Selecting a signal source for actual values A, B. and C

To set the signal sources for the actual values of A, B and C, proceed as follows:

1. Double-click on "Parameter setting" in the device view of the module.
2. Navigate in the configuration view to "Controller > Channel [No] > System deviation".
3. In the "Control value" section select the signal source for each actual value. Here you can select whether you use no signal ("zero") as a source or one of the processed analog inputs.
4. Save the parameters entered with "Project > Save".

Registering the factors for actual values B and C and the offset value

Proceed as follows to set the signal sources for the actual values A, B and C:

1. Double-click on "Parameter setting" in the device view of the module.
2. Navigate in the configuration view to "Controller > Channel [No] > System deviation".
3. In the"Sum" section set the necessary weight factors for the actual values B and C.
4. Set the required "offset" value that should be added to the actual values.
5. Save the parameters entered with "Project > Save".

Implementing ratio or mixture controllers

Selecting the controller type

Proceed as follows to implement a ratio or blending controller:

1. Double-click on "Parameter setting" in the device view of the module.
2. Navigate in the configuration view to "Controller> channel [No]".
3. In the "Encoder type" section select the "Ratio or Mixture controller".
4. Save the parameters entered with "Project > Save".

Processing setpoints

Proceed as follows to adjust the setpoint processing mode:

1. Double-click on "Parameter setting" in the device view of the module.
2. Navigate in the configuration view to "Controller > Channel [No] > System deviation".
3. In the "Setpoint source" section, select the source from which the setpoint is to be obtained.
4. In the "Safety setpoint" section, enter a value to be utilized upon CPU fault and start up as a safety value.
5. Select the behavior of the controller module for "Reaction to CPU errors" and "Start-up behavior". You may select from amongst the following options:
   - Setpoint = last valid setpoint: the setpoint will be assigned the last valid setpoint upon occurrence of the event. The last valid setpoint will remain unchanged.
   - Setpoint = safety setpoint: the setpoint will be assigned the safety setpoint upon the occurrence of the event. This enables secured emergency operation.
6. In the "setpoint ramp" section, set the rise time to be adhered to upon a setpoint change to reduce overshoot.
7. In the "Setpoint limits" section, set both an high and a low limit for the setpoint.
8. Save the parameters entered with "Project > Save".

Selecting the signal source for actual values A and D

To set the signal sources for the actual values of A and D, proceed as follows:

1. Double-click on "Parameter setting" in the device view of the module.
2. Navigate in the configuration view to "Controller > Channel [No] > System deviation".
3. Select the signal source for each actual value in the "Manipulated value" section. You can choose whether using a no signal ("zero") as a source or one the processed analog inputs. For the actual value for D, you may also select the controller control value.
4. Save the parameters entered with "Project > Save".

Registering Multipliers

Proceed as follows to set the signal sources for actual values A and D:

1. Double-click on "Parameter setting" in the device view of the module.
2. Navigate in the configuration view to "Controller > Channel [No] > System deviation".
3. In the "Multiply" section select the desired offset.
4. Save the parameters entered with "Project > Save".

For ratio controllers the actual value A is used as control variable while the actual value D stores the ratio magnitude. The setpoint input value is used as a ratio factor. Multiplying it by the actual value of D and adding the adjustable offset will yield the effective setpoint value. If actual value D is off only the offset should be added to the setpoint.

Setting the alarm generation

Procedure

Proceed as follows to set interrupt generation:

1. Double-click on "Parameter setting" in the device view of the module.
2. Navigate in the configuration view to "Controller > Channel [No] > System deviation".
3. In the "Alarm generation" section select the necessary options:
   - In the "Limit monitor for" section enter the signal for which the alarm should be set.
   - In "Alarm high limit", set the high limit beyond which the alarm should be triggered.
   - In "Warning high limit", set the value beyond which the "approaching high limit" warning should be issued.
   - In "Warning low limit", set the value below which the "approaching low limit" warning should be issued.
   - In "Alarm low limit", set the low limit below which the alarm should be triggered.
   - To avoid flickering of the monitor you may set the hysteresis parameter in "hysteresis".
4. Save the parameters entered with "Project > Save".

Control algorithms

Implementing a P control

Procedure

To implement a P control, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "Type" section select the "PID controller" option.

4. In the "General" section under "Operation mode" select the "P" option.
   All configurable options for a P control will be activated and all other options disabled.
5. Save the parameters entered with "Project> Save".

Implementing a PI control

Procedure

To implement a PI control, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "Type" section select the "PID controller" option.
4. In the "General" section under "Operation mode" select the "PI" option.
   All configurable options for a PI control will be activated and all other options disabled.
5. Save the parameters entered with "Project> Save".

Implementing an I control

Procedure

To implement an I control, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "Type" section select the "PID controller" option.
4. In the "General" section under "Operation mode" select the "I" option.
   All configurable options for an I control will be activated and all other options disabled.
5. Save the parameters entered with "Project> Save".

Implementing a PID control

Procedure

To implement a PID control, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "Type" section select the "PID controller" option.
4. In the "General" section under "Operation mode" select the "PID" option.
   All configurable options for a PID control will be activated and all other options disabled.
5. Save the parameters entered with "Project> Save".

Implementing a PD control

Procedure

To implement a PD control, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "Type" section select the "PID controller" option.
4. In the "General" section under "Operation mode" select the "PD" option.
   All configurable options for a PD control will be activated and all other options disabled.
5. Save the parameters entered with "Project> Save".

Implementing a Fuzzy temperature controller

Procedure

To implement a P control, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "Type" section select the "Fuzzy temperature controller module" option.
   All configurable options for a fuzzy temperature controller module will be activated and all other options disabled.
4. Save the parameters entered with "Project> Save".

Suppressing the deviation noise component

Procedure

To suppress the noise component in the deviation, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "General" section, set the "Dead zone width" parameter.
4. Save the parameters entered with "Project> Save".

Switching disturbances

Procedure

To switch a disturbance, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. From the "Disturbance source" list in the "General" section select the disturbance source. Select the entry "zero" to perform no disturbance switching.
4. Enter the "Disturbance factor" value you wish to use as a disturbance weight factor. A value <0 weakens the disturbance whereas a value> 0 increases the disturbance. A value of 1 keeps the disturbance unchanged.
5. Save the parameters entered with "Project> Save".

Setting the feedback P portion

Procedure

To implement a constant or cascade controller, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "P portion" section select the "In feedback" option. If the option is enabled, the P portion is set in feedback.
4. Save the parameters entered with "Project> Save".

Attenuating the D portion for setpoint changes

Selecting the controller type

To implement a constant or cascade controller, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "D portion" section enter the D portion source in the "D portion" option.
4. In the "Differentiation factor" section, set the D portion factor.
5. Save the parameters entered with "Project> Save".

Reversing the controller effect

Principle

Controller reversal may be achieved by setting a negative proportional gain thus moving from "increasing deviation value = increasing correction value" to "increasing deviation value = decreasing correction value".

Procedure

To reverse the controller effect, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "General" section, set a proportional gain of opposite sign.
   As an example, if the current setting shows a positive proportional gain and you wish to reverse the controller effect, set a negative proportional gain.
4. Save the parameters entered with "Project> Save".

Controller output

Implementing a continuous controller

A continuous controller can be created only with an analog module.

Procedure

1. Changing an external control value (Page 9879)
2. Setting the safety control value (Page 9881)
3. Changing the tracking (Page 9880)
4. Setting control value limits (Page 9881)
5. Controlling two actuators - split range (Page 9882)
6. Selecting the output signal and the signal type (Page 9883)

Step controller without position feedback

A step controller without position feedback can be implemented only with a digital module.

Procedure

1. Select controller output step controller without position feedback
2. Changing an external control value (Page 9879)

3. Setting the safety control value (Page 9881)
4. Using the pulse shaper step controller (Page 9882)

Implementing a step controller with position feedback

A step controller with position feedback can be implemented only with a digital module.

Procedure

1. Select controller output step controller with position feedback
2. Changing an external control value (Page 9879)
3. Changing the tracking (Page 9880)
4. Setting the safety control value (Page 9881)
5. Setting control value limits (Page 9881)
6. Setting position feedback (Page 9880)
7. Using the pulse shaper step controller (Page 9882)

Implementing a pulse controller

A pulse controller can be implemented only with a digital module.

Procedure

1. Select controller output pulse controller
2. Changing an external control value (Page 9879)
3. Changing the tracking (Page 9880)
4. Setting the safety control value (Page 9881)
5. Setting control value limits (Page 9881)
6. Prepare the analog-to-digital signal conversion
7. Converting analog into digital signal (Page 9883)

Parameterize the functions of the controller output

Changing an external control value

Procedure

For changing the external control variable, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> controller output".

3. In the "External control variable" select the option to activate the external control variable. You may choose from the following options:
   - Activation of the function module.
   - Activation of a function module or one of the digital inputs.
4. Save the parameters entered with "Project> Save".

Changing the tracking

Procedure

To change the tracking proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> controller output".
3. In the "Tracking" section select the option to activate the external control variable. You may choose from the following options:
   - Activation of the function module.
   - Activation of a function module or one of the digital inputs.
4. In the "Tracking" section select the source of the signal you wish to track. You may choose from the following options:
   - Zero, while no source is used.
   - One of the analog inputs.
5. Save the parameters entered with "Project> Save".

Setting position feedback

Procedure

To set the position feedback, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to 'Digital Output> output [No.] ".
3. In the "Position feedback input" section select the "Source" option to choose the signal for position feedback.
   - Zero: no position feedback.
   - One of the processed analog inputs.
4. Save the parameters entered with "Project> Save".

Setting the safety control value

Procedure

To set the safety control value, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> controller output".
3. In the "safety control value" section select the value that should be used as a safety control value.
4. Under "Activation" select the option to activate the external control variable. You may choose from the following options:
   – Activation of the function module.
   – Activation of a function module or one of the digital inputs.
5. Under "Action on module start" select the option which should be executed as the module starts. You may choose from the following options:
   – Control mode: sets the module immediately into control mode upon startup.
   – Safety manipulated value: initially, the safety manipulated value is controlled.
6. Under "Action on controller interference of the actual value A" select the option which should be executed as a controller interference of the actual value A occurs.
7. Under "Action on input controller interference" select the option which should be executed as an input controller interference occurs.
8. Save the parameters entered with "Project> Save".

Setting control value limits

Procedure

To set control value limits you should proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> controller output".
3. In the "Output signal limit" enter the following values:
   – High value: illustrates the high limit for the output signal.
   – Low value: illustrates the low limit for the output signal.
4. Save the parameters entered with "Project> Save".

Controlling two actuators - split range

Procedure

To control two actuators you should proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> controller output".
3. In the "Split range" section select the option "Enable" to effectively control two actuators.
4. Set the following values for both the input and output signals A and for the input and output signals B:
   - Low input signal.
   - High input signal.
   - Low output signal.
   - High output signal.
5. Save the parameters entered with "Project> Save".

Using the pulse shaper step controller

Procedure

To configure the pulse shaper you should proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to 'Digital Output> output [No.] ".
3. In the "Pulse shaper" section, enter the following parameters:
   - Engine operating time.
   - Minimum pulse duration.
   - Minimum break duration.
4. Select the best acceptable option for the high and the low values:
   - From the function module.
   - From the function module or one of the digital inputs.
5. Save the parameters entered with "Project> Save".

Converting analog into digital signal

Procedure

To convert an analog signal into a digital signal you should proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate to the "Digital output> channel [No.]" in the design view.
3. In the "Pulse shaper" area, enter the following parameters:
   - Minimum pulse duration.
   - Minimum break duration.
4. Save the parameters entered with "Project> Save".

Selecting the output signal and the signal type

Procedure

To select the output signal and the signal type you should proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate to the "Digital output> channel [No]" in the design view.
3. Under the option "Source" select the signal you wish to use as output signal. You may choose among the following options:
   - Zero: no source for the output signal.
   - One of the analog inputs.
   - One of the control values for controller A or B.
4. Under the option "Signal type" select the type for the output signal you wish to use.
5. Save the parameters entered with "Project> Save".

Comissioning FM 355-2

Creating and supplying instance DB

Creating and initializing an instance DB

1. Create a user program that calls the FMT_PID instruction cyclically.
2. Set READ_OUT = TRUE cyclically in the user program.
3. Create an instance DB of the FMT_PID instruction for each controller channel.

4. Enter the module address for each instance DB in the MOD_ADDR parameter.
   Use the start address that you specified when you configured your hardware as the module address of the FM 355-2.
5. Enter the channel number of the controller channel (0, 1, 2 or 3) in the FMT_PID.CHANNEL parameter.
1. If you use the FMT_PAR and FMT_TUN instructions, enter the channel number of the controller channel in the FMT_PAR.CHANNEL and FMT_TUN.CHANNEL parameters.
2. Check to ensure that the "Sensor type analog input" of this channel is selected.
   If the analog input is not processed, all PID parameters are set to zero and a manipulated variable is not calculated.

Note

Multi-instance DBs

If you use multi-instance DBs in your project, you must commission the FM via a watch table.

See also

FMT_PID (Page 7226)

FMT_TUN (Page 7254)

FMT_PAR (Page 7247)

Commissioning the controller

Introduction

You tune the PID parameters as follows:

- Pretuning
- Fine tuning heating
- Fine tuning cooling
- Manual fine tuning

Procedure

Proceed as follows to tune a controller:

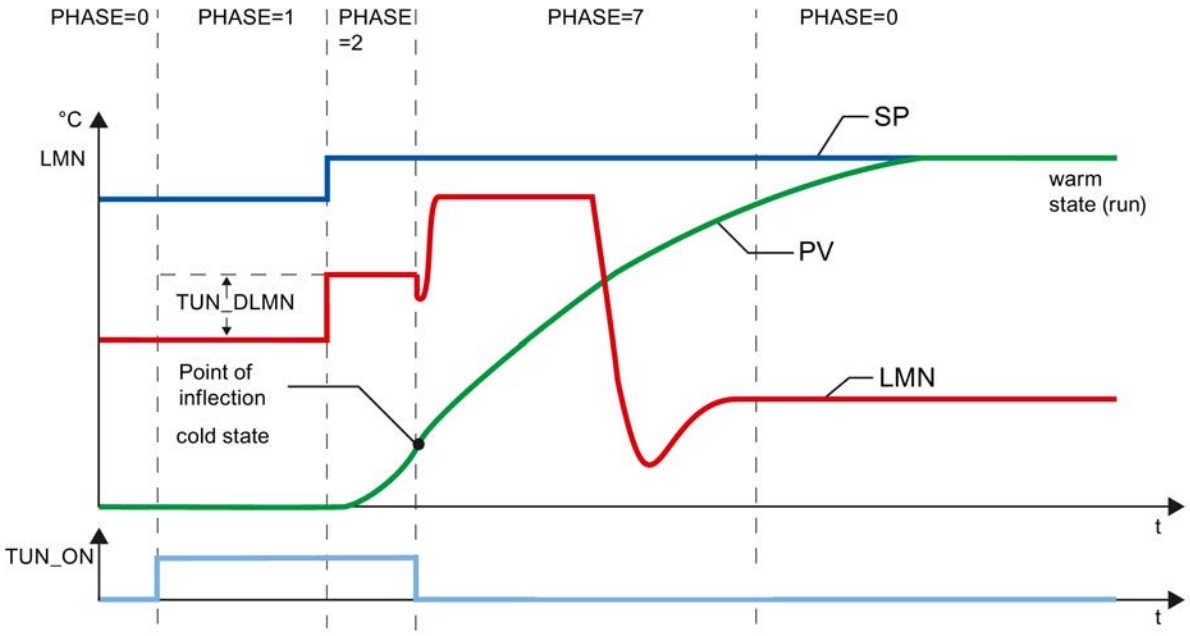
1. Put the CPU in RUN mode.
2. If using a step controller, measure the motor actuating time.
3. Perform controller tuning.
4. Monitor the control loop with the trend view.

Options of controller tuning

Pretuning

During this tuning, the working point is approached from the cold state through a setpoint jump.

With TUN_ON = TRUE, you can establish the tuning readiness. The controller switches from PHASE = 0 to PHASE = 1.



The tuning manipulated variable (LMN0 + TUN_DLMN) is activated by a setpoint change (transition PHASE 1 -> 2). The setpoint is not effective until the inflection point has been reached (automatic mode is not enabled until this point is reached).

The user is responsible for defining the output excitation delta (TUN_DLMN) according to the permitted process value change. The sign of TUN_DLMN must be set depending on the intended process value change (take into account the direction in which the control is operating).

The setpoint step change and TUN_DLMN must be suitably matched. If the value of TUN_DLMN is too high, there is a risk that the point of inflection will not be found before 75% of the setpoint step change is reached.

TUN_DLMN must still be high enough to ensure that the process value reaches at least 22% of the setpoint step change. Otherwise, the process will remain in tuning mode (phase 2).

Solution: Reduce the setpoint value during the inflection point search.

Note

If processes are extremely sluggish, it is advisable to specify a target setpoint that is somewhat lower than the desired operating point during tuning and to monitor the status bits and PV closely (risk of overshooting).

Tuning only in the linear range:

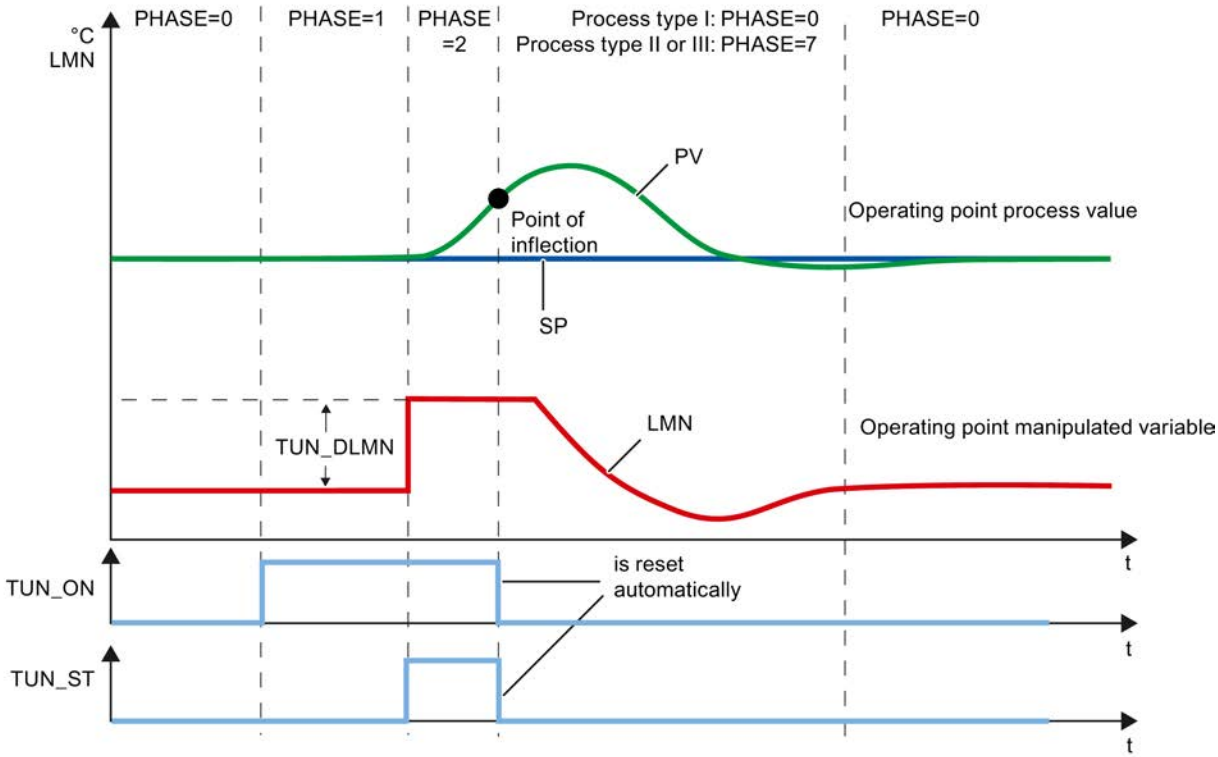
The signals of certain processes (e.g., zinc or magnesium smelters) will pass a non-linear area at the approach of the operating range (change in the state of aggregation).

By selecting a suitable setpoint step change, tuning can be limited to the linear range. When the process value has passed 75% of the setpoint step change (SP_INT-PV0), tuning is ended.

At the same time, TUN_DLMN should be reduced to the extent that the point of inflection is guaranteed to be found before 75% of the setpoint step change is reached.

Fine tuning heating

During this tuning, the process with a constant setpoint is activated through a output value jump.



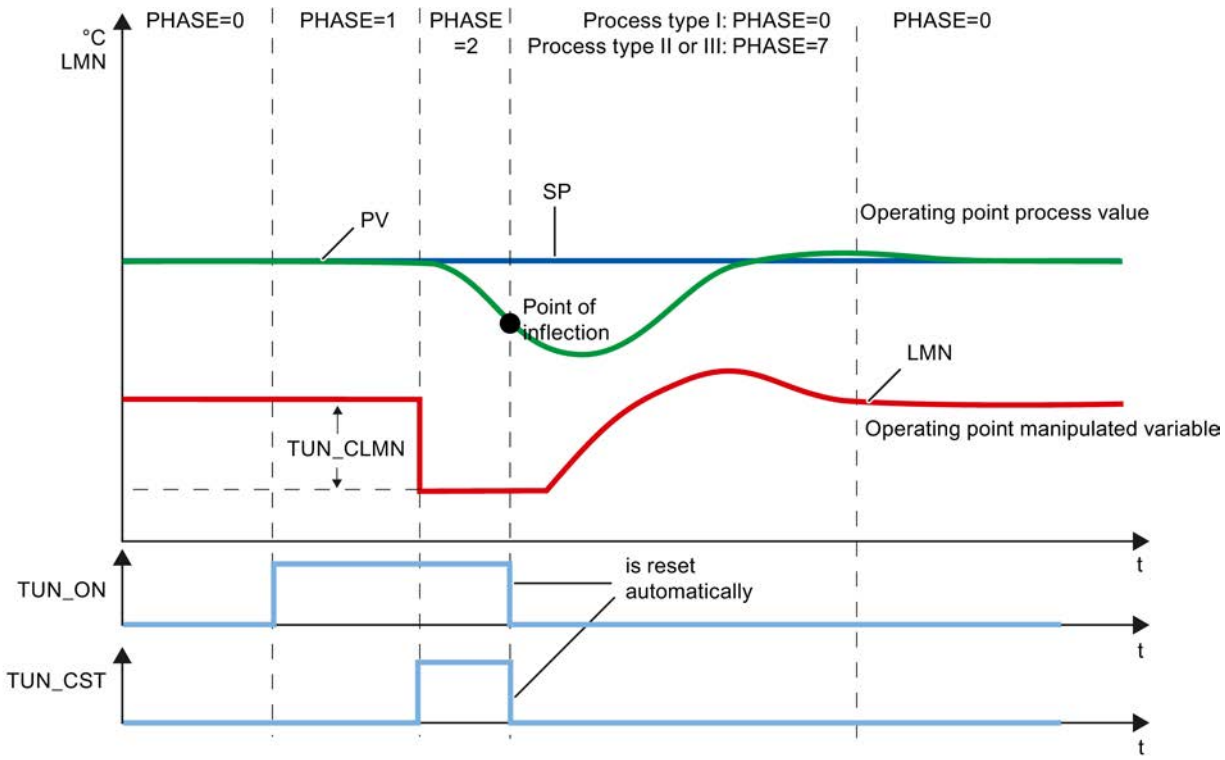
The tuning manipulated variable (LMN0 + TUN_DLMN) is activated by setting the start bit TUN_ST (transition from phase 1 -> 2). When you modify the setpoint value, the new value will not take effect until the point of inflection has been reached (automatic mode will not be enabled until this point has been reached).

The user is responsible for defining the output excitation delta (TUN_DLMN) according to the permitted process value change. The sign of TUN_DLMN must be set depending on the intended process value change (take into account the direction in which the control is operating).

NOTICE
Safety off at 75% is not available when you excite the process via TUN_ST. Tuning is ended when the point of inflection is reached. However, in noisy processes the point of inflection may be significantly exceeded.

Fine tuning cooling

During this tuning, the process with a constant setpoint is activated through a output value jump.



The tuning manipulated variable (LMN0 + TUN_CLMN) is activated by setting the start bit TUN_CST (transition from phase 1 -> 2). The previously determined manipulated value LMN0 is kept constant during this process and TUN_CLMN is activated directly in the split-range function so that only the cooling power is changed.TUN_CST is reset immediately.

When you modify the setpoint during phase 2, the new setpoint will not take effect until the point of inflection has been reached (automatic mode will not be enabled until this point has been reached).

The user is responsible for defining the output excitation delta (TUN_CLMN) according to the permitted process value change. The sign of TUN_CLMN must be set depending on the intended process value change (take into account the direction in which the control is operating). Note that a negative TUN_CLMN causes an increase in cooling power. But you can also use TUN_CLMN > 0.0 to activate a setpoint jump by reducing the cooling power.

NOTICE
Safety off at 75% is not available when you excite the process via TUN_CST. Tuning is ended when the point of inflection is reached. However, in noisy processes the point of inflection may be significantly exceeded.

Split-range function

During optimization, the gradient of the split-range function is added to the process. If you want to modify the gradient of the split-range function after optimization, you must adapt the controller's GAIN or RATIOFAC parameter accordingly.

Manual fine-tuning in control mode

The following measures can be employed to achieve an overshoot-free setpoint response:

- Adapt control zone
- Optimize command action
- Attenuation of control parameters
- Control parameter modification

Automatic tuning

Process requirements

Transient response

The process must have a stable, asymptotic transient response with time lag.

The actual value must settle to steady state after a jump of the manipulated variable. This therefore excludes processes that already show an oscillating response without control, as well as processes with no compensation (integrator in the control system).

Guaranteeing a stationary initial state (phase 0)

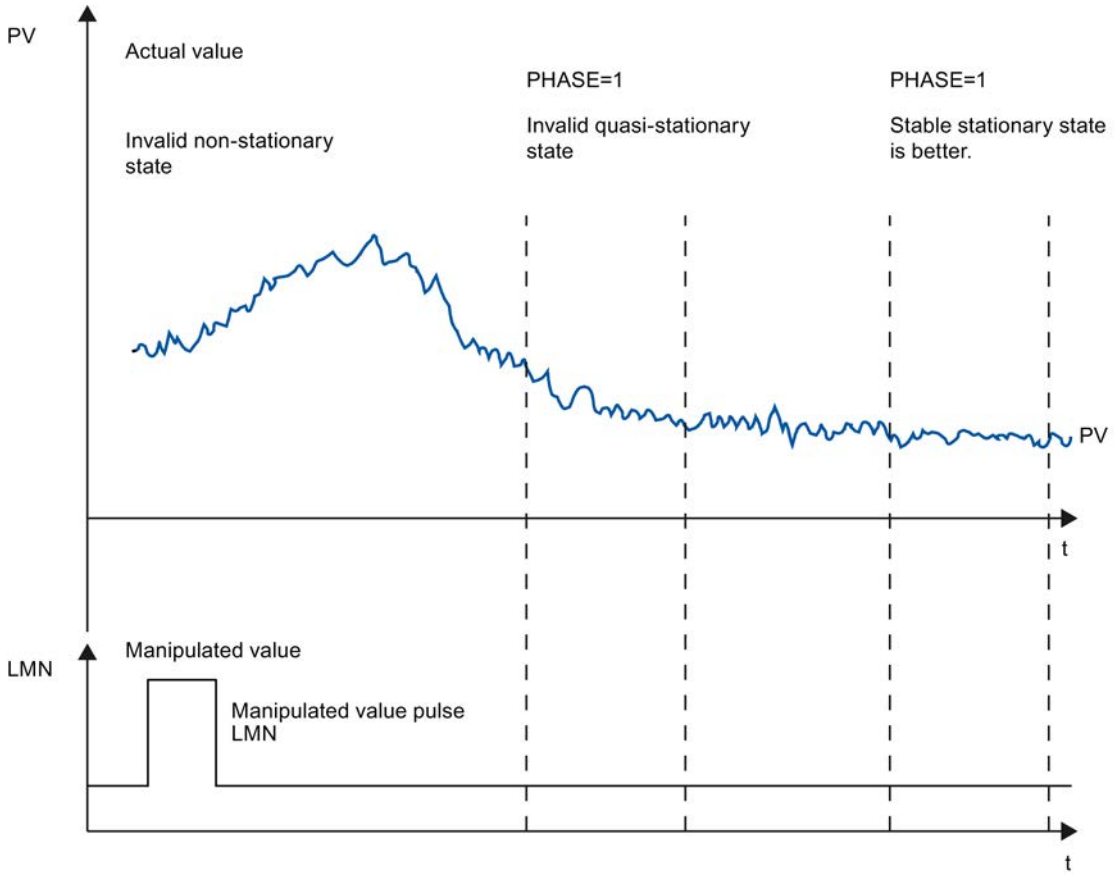
With lower-frequency oscillations of the actual value, for example, due to incorrect controller parameters, the controller must be put in manual mode before the optimization is started and wait for the oscillation to stop. You can also switch to a PI controller with small proportional gain and high integration time.

Now you have to wait until the stationary state is reached, this means, until the process value and output value have a steady state. It is also permissible to have an asymptotic transient oscillation or slow drifting of the process value (stationary state, see the following image). The output value must be constant or fluctuate by a constant average.

Note

Avoid changing the manipulated variable shortly before starting the tuning. A change of the manipulated variable can occur in an unintended manner through the establishment of the test conditions (for example, closing an oven door)! If this does happen, you have to at least wait until the process value has an asymptotic transient oscillation in a stationary state again. Better controller parameters can be reached if you wait until the transient effect has completely subsided.

In the following image, the transient oscillation is illustrated in the stationary state:



Linearity and operating range

The process response must be linear across the operating range. Non-linear response occurs, for example, when a state of aggregation changes. Optimization must take place in a linear part of the operating range.

This means, during optimization and normal control operation non-linear effects within the operating range must be insignificant. It is, however possible to retune the process when the operating point changes, providing optimization is repeated in the close vicinity of the new working point and non-linearity does not occur during the optimization.

If a specific static non-linearity (e.g. valve characteristics) is known, it is always advisable to compensate this with a polyline to linearize the process response.

Disturbance in temperature processes

Disturbances such as the transfer of heat to neighboring zones must not affect the overall temperature process too much. For example, when optimizing the zones of an extruder, all zones must be heated simultaneously.

Process of controller tuning

PI/PID controller parameters

The FM 355-2 auto-tuning feature automatically sets the PI/PID controller parameters. You can tune the heating and cooling processes as well as split-range processes with two counteracting final controlling elements (e.g. the final controlling elements for heating and cooling processes).

Phases of controller tuning

For the controller tuning, individual phases are run through, which you can read on the parameter PHASE .

PHASE = 0

Tuning is not running. The FM 355-2 works in automatic or manual mode.

During PHASE = 0, you can make sure that the controlled system fulfills the requirements for tuning.

At the end of tuning, FM 355-2 switches back into PHASE = 0.

PHASE = 1

FM 355-2 is ready for tuning. PHASE = 1 may only be started if the requirements for tuning are fulfilled.

During PHASE = 1, the following values are determined:

- Process value noise NOISE_PV
- Initial slope PVDT0
- Average of the manipulated variable

PHASE = 2

In phase 2, the process value attempts to detect the point of inflection with a constant manipulated variable. This method prevents the point of inflection from being found too early as a result of process variable noise.

Initially, this averaging is inactive; in other words, averaging always takes place over one cycle. As long as the noise exceeds a certain level, the number of cycles is doubled.

The period and amplitude of the noise are calculated. The search for the point of inflection is canceled and phase 2 is exited only when the gradient is always smaller than the maximum rise during the estimated period. TU and T_P_INF are, however, calculated at the actual point of inflection.

Tuning, however, is only ended when the following two conditions are met:

1. The process value is more than 2*NOISE_PV away from the point of inflection.
2. The process value has exceeded the point of inflection by 20%.

Note

When exciting the process using a setpoint step change, tuning is ended at the latest when the process value exceeds 75% of the setpoint step change (SP_INT-PV0) (see below).

PHASE = 3, 4, 5, 6

The phases 3, 4, 5 and 6 last 1 cycle each.

In Phase 3, the valid PI/PID parameters are saved before the optimization and the process parameter is calculated.

In Phase 4, the new PI/PID parameters are calculated.

In phases 5 and 6, the new manipulated variable is calculated and the controlled system is given.

If a controlled system of type I was identified, tuning mode stops and tuning is once again in phase 0. The controller now starts with LMN = LMN0 + 0.75*TUN_DLMN always in automatic mode (even if you controlled in manual mode before you started tuning).

If a controlled system of type II or III was identified, tuning changes to phase 7.

PHASE = 7

There is a risk with typical temperature systems (process type I) that the point of inflection is detected too early as a result of noise. The lower point of inflection time T_P_INF may possibly result in the detection of a process type II or III.

There is a check in phase 7 to see if the process type is correct. This testing occurs in the automatic mode with the recently recalculated controller parameters and ends at the latest 6*TA (equilibrium time) after the point of inflection. If process type I is detected, the controller parameters are recalculated (STATUS_D = 122) otherwise they remain the same.

The checking of the process type is cancelled, if the process value during tuning at the operating point reaches the process value once again prior to tuning PV0.

If phase 7 is canceled with TUN_ON = FALSE, the already determined controller parameters are maintained!

Note

It is generally possible to start the heat tuning during cooling. The order check in phase 7 will not take place in this case.

The reverse case (starting cooler tuning during heating) is not critical, because a cooler tuning never runs through phase 7!

Premature cancellation of the optimization

In Phase 1, 2 or 3, you can cancel the optimization by resetting TUN_ON = FALSE without calculating new parameters. The controller starts in automatic mode with LMN = LMN0. If the controller was in manual mode before the tuning, the old manual manipulated variable is output.

If tuning is canceled in Phase 4, 5, 6 or 7 with TUN_ON = FALSE, the controller parameters determined until then are maintained.

Tuning result

Optimization result

The left number of STATUS_H and STATUS_C displays the tuning state.

STATUS_H STATUS_C	Result
0	Default or new controller parameters have not yet been found.
10000	Suitable controller parameters were found
2xxxx	Control parameters have been found via estimated values; check the control response or check the STATUS_H/C diagnostic message and repeat controller optimization.
3xxxx	An operator input error has occurred; check the STATUS_H/C diagnostic message and repeat controller optimization.

After the recognition of the inflection point in phase 6, the following controller parameters are updated on the FM 355-2 and at the online instance DB of instruction FMT_PID:

- Factor for the attenuation of the proportional component PFAC_SP = 0.8
- Controller GAIN
- Integration time TI (limited to ≥0.5 s)
- Derivative time TD (limited to ≥1.0 s)
- Derivative factor D_F = 5.0
- Control zone on/off CONZ_ON=TRUE/FALSE
- Control zone width CON_ZONE=250/GAIN
- P_SEL = TRUE (even if it was previously a controller with integral action only)

If RATIOFAC<>0.0, then CON_ZONE is multiplied by the factor 1.5.

If a value TD<1.0 s is calculated in the optimization process, only a PI controller is determined and PID_ON as well as the PID parameters will be set to zero.

The control zone is only enabled for matching process types (process type I and II) and PID controllers (CONZ_ON = TRUE).

Depending on PID_ON, control is implemented either with a PI or a PID controller. The old controller parameters are saved and can be retrieved with UNDO_PAR. A PI and a PID parameter record is saved additionally. Using LOAD_PID and making a suitable setting for PID_ON, it is also possible to switch later between the tuned PI or PID parameters.

A previously split structure (derivative component in the feedback path) will be maintained.

Note

Verify correct operation of your controller parameters immediately after the controller optimization process is completed.

Split-range function

During optimization, the gradient of the split-range function is added to the process. If you want to modify the gradient of the split-range function after optimization, you must adapt the controller's GAIN parameter accordingly.

Fault descriptions and corrective measures

Compensating operator errors

Tuning is canceled in case of all errors listed in this table.

Errors	STATUS and actions	Comment
TUN_ON and TUN_ST or TUN_CST are set simultaneously	STATUS_H/C = 30001 • Transition to phase 0 • TUN_ON = FALSE	Cannot occur with assistant, because TUN_ST and TUN_CST are set to FALSE in the first template.
Sign of setpoint change does not match process value change.	End tuning at the point of inflection Safety off at 75% ineffective	
|TUN_DLMN| or |Effective TUN_DLMN| < 5 % (end of phase 1) Special case STATUS_H=30004: Effective manipulated value difference limited by split–range limits and not by manipulated value limits	STATUS_H/C = 30002/30004 • Transition in phase 99 (Phase 0 in a parallel optimization process) • Output LMN = LMN0	Either you configured a value for TUN_DLMN that is too low, or the manipulated variable was near to a control limit prior to start of tuning. If STATUS_H=30004: Also take into account that, for example, with LMN_A < 5% a heating tuning with negative TUN_DLMN is not possible (reason: the cooling power must not be adjusted). In this case, you must prevent the controller from settling to the new setpoint and from unnecessary leaving the stationary operating point (not possible with parallel tuning process). You should now proceed as follows: • Withdrawing setpoint • TUN_ON=FALSE • Correcting TUN_DLMN • Restart tuning
Only for step controllers without analog position feedback: |Effective TUN_DLMN| < 5 % (start of phase 2)	STATUS_H = 30002 • Transition to phase 0 • TUN_ON = FALSE	Error occurs if |TUN_DLMN| >= 5 % but manipulated value was near to a control limit prior to start of tuning.

Errors	STATUS and actions	Comment
If TUN_CST: |TUN_CLMN| or |Effective TUN_CLMN| < 5 % (end of phase 1) Special case STATUS_H=30004: Effective manipulated value difference limited by split–range limits and not by manipulated value limits.	STATUS_C = 30002/30004 • Transition to phase 0 • TUN_ON = FALSE	Either you configured a value for TUN_CLMN that is too small or the manipulated variable was near to a control limit prior to start of tuning.
If TUN_CST: Special case of "|Effective TUN_CLMN| < 5 % (end of phase 1)": TUN_CLMN <= -5%, but LMN_LLM > -5%.	STATUS_C = 30003 • Transition to phase 0 • TUN_ON = FALSE	The low limit LMN_LLM is too high (e.g. 0%) and therefore prevents the output of cooling power.
TUN_CST but without previous heating tuning	STATUS_C = 30008 • Transition to phase 0 • TUN_ON = FALSE	
Safety mode	STATUS_C = 30009 • Transition to phase 0 • TUN_ON = FALSE	If FM 355-2 enters safety mode in PHASE > 2, it is displayed by STATUS_C=30009.

Point of inflection not reached (only if excited by setpoint step change)

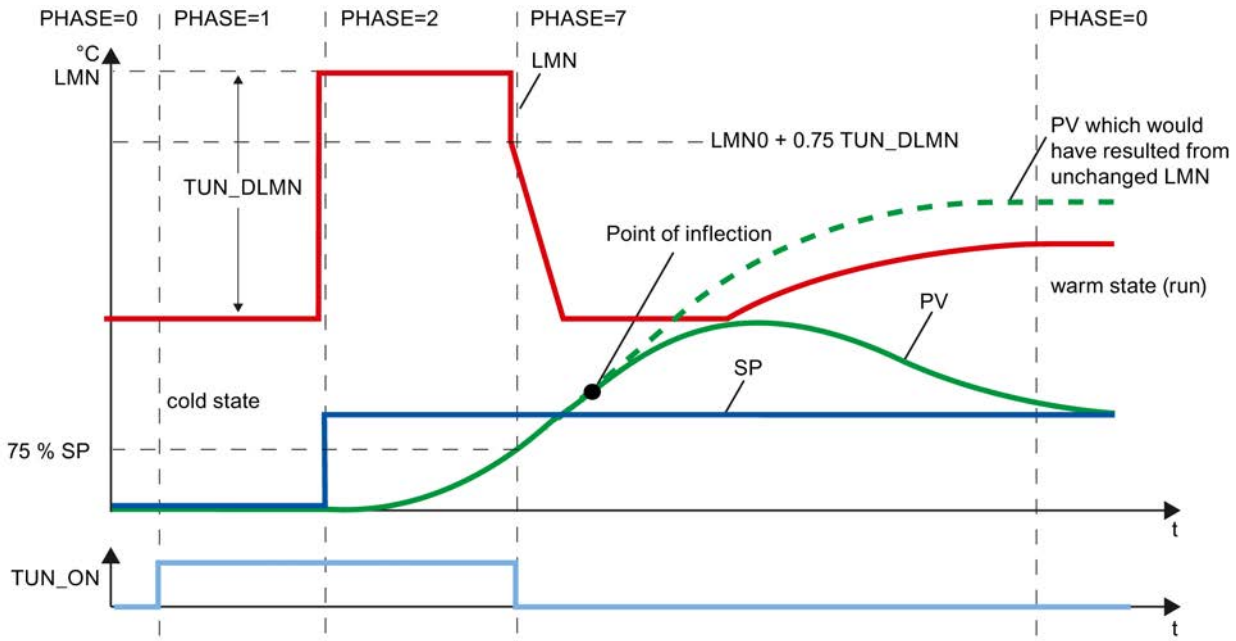
At the latest, tuning is ended when the process value has passed 75% of the setpoint step change (SP_INT-PV0). This is signaled by "inflection point not reached" in STATUS_H/ STATUS_C (2xx2x).

The currently valid setpoint always applies. By reducing the setpoint, it is possible to achieve an earlier end of the tuning function.

In typical temperature processes, cancelation of tuning at 75% of the setpoint step change is normally adequate to prevent overshoot. However, **caution** is advised, particularly in processes with a greater delay (TU/TA > 0.1, process type III). If manipulated variable excitation is too strong compared to the setpoint step change, the process value can overshoot heavily (up to a factor of 3).

In higher-order processes, if the point of inflection is still a long way off after reaching 75% of the setpoint step change, there will be significant overshoot. In addition, the controller parameters are too stringent. In this case, you should reduce the controller parameters or repeat the attempt.

The following schematic illustrates the overshoot of the process variable when the excitation is too strong (process type III):



In typical temperature processes, cancelation shortly before reaching the point of inflection is not critical in terms of the controller parameters.

If you repeat the attempt, reduce TUN_DLMN or TUN_CLMN or increase the setpoint step change.

Principle: The value of the manipulated variable used for tuning must be suitable for the setpoint step change.

Error estimating the delay time or order

The delay time (STATUS_H/STATUS_C = 2x1xx, 2x2xx or 2x3xx) or order (STATUS_H/STATUS_C = 21xxx or 22xxx) were not acquired correctly. Operation continues with an estimate that can lead to non-optimum controller parameters.

Repeat the tuning procedure and ensure that disturbances do not occur at the process value.

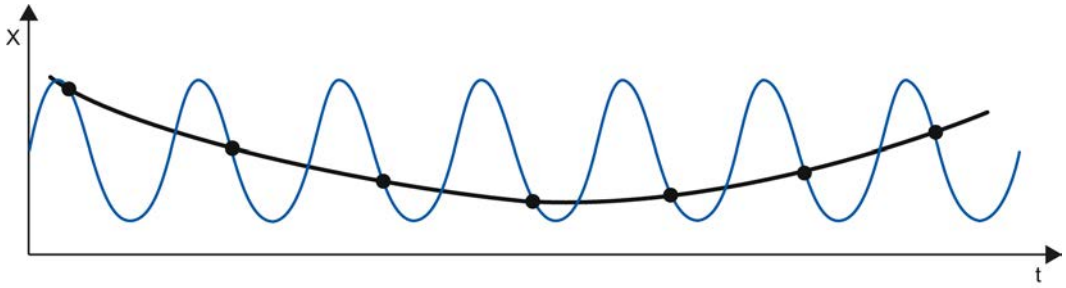
Note

The special case of a PT1-only process is also indicated by STATUS_H/STATUS_C = 2x2xx (TU < CYCLE). In this case, it is not necessary to repeat the attempt. Reduce the controller parameters if the control oscillates.

Quality of measuring signals (measurement noise, low-frequency interference)

The results of tuning can be distorted by measurement noise or by low-frequency interference. Note the following:

- During one noise period, the process value should be sampled at least twice. The degree of noise should not exceed 5% of the useful signal change.
- High-frequency interference cannot be filtered out by a software block. This should be filtered earlier in the measuring sensor to prevent the aliasing effect.
  The following schematic illustrates the aliasing effect when the sampling time is too long:



- In case of low-frequency interferences, we assume that the sampling time of the FM 355-2 is small enough. However, the FM 355-2 must then generate a uniform measuring signal by having a large interval in the mean value filtering. Mean value filtering must extend over at least two noise periods. Internally in the block, this soon results in higher sampling times such that the accuracy of the tuning is adversely affected. Adequate accuracy is guaranteed with at least 40 noise periods up to the point of inflection. Possible remedy when repeating the attempt: Increase TUN_DLMN/TUN_CLMN.

Overshoot

Overshoot can occur in the following situations:

Situation	Cause	Remedy
End of tuning	• Excitation by a too high manipulated value change compared with the setpoint step change (see above). • PI controller activated by PID_ON = FALSE.	• Increase the setpoint step change or reduce the manipulated value step change. • If the process permits a PID controller, start tuning with PID_ON = TRUE.
Tuning in phase 7	Initially, less aggressive controller parameters were determined (process type III); these can lead to an overshoot in phase 7.	-
Control mode	PI controller with PFAC_SP = 1.0 for process type I.	If the process permits a PID controller, start tuning with PID_ON = TRUE.

Tuning with step controller

Introduction

The general information on controller optimization applies.

Peculiarities of step controllers

- Step controllers on the FM 355-2 operate without control zone.
- Phase 7 is not executed.
- No cooling optimization.

Note

The motor transition time is not determined by tuning and must be measured or set before you start tuning.

Controller design

The motor actuating time MTR_TM should be as small as possible compared to the inflection point time T_P_INF and the equivalent dead time TU.

Controllers with a softer action are automatically generated for longer motor actuating times.

The higher the process excitation TUN_DLMN, the higher is the influence of the motor actuating time on the controller design.

PI or PID parameters

PI parameters are attenuated by 25% compared to a controller designed for closed-loop controllers and pulse controllers. PID parameters are also determined (but without the 25% safety margin). The PID parameters should only be used if the motor actuating time is not too high compared to the process parameters and when the load on the final controlling element stays within the hysteresis due to the derivative component.

Step controller without position feedback

The output of Open/Close instructions is stopped right at the start of phase 1 (quasi manual mode).

The mean value of the manipulated variable in phase 1 (LMN0) will not be calculated.

At the start of phase 2 and during the time MTR_TM * TUN_DLMN / 100, an Open instruction (or with negative TUN_DLMN, a close instruction) is output.

When an end stop signal is triggered during the pulse action in phase 2, the effective TUN_DLMN is calculated as: 100 * time / MTR_TM.

When the value of the effective TUN_DLMN < 5%, the error message STATUS_H=30002 is output and optimization is canceled. This error can only occur if the final controlling element is unexpectedly close to a limit.

End or cancellation of optimization

The controller starts with LMN = LMN0 + TUN_DLMN.

Performing pretuning

Requirement

- The requirements on the process are met.
- An instance DB has been created for the channel and is supplied.
- "Instruction" is configured as source for the setpoint.
- The FM 355-2 is not in safety mode (SAFE_ON = FALSE). An ongoing tuning is cancelled when safety mode is switched on.

Procedure

WARNING

This may result in death, severe injury or considerable property damage.

During an tuning, the parameter LMN_REON is ineffective. Even tracking circuits that are derived from alarm limits, for example, do not have an effect. Through this, the manipulated value or process value may take on undesired - even extreme - values.

The manipulated value is defined through the tuning. To cancel the tuning, you first have to set TUN_ON = FALSE. This will make LMN_REON effective once again.

To manually determine the optimum PID parameters for initial commissioning, follow these steps:

1. Select the instance DB of the channel you want to tune.
2. Click the "Measurement on" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value and output value are recorded.
3. Select "Pretuning" from the "Mode" drop-down list.
   The FM 355-2 is ready for tuning (PHASE = 1).
4. Wait until the steady state has been reached.
5. In the "Output value jump" field, specify how much the output value should be increased.

6. Enter a setpoint in the "Setpoint" field. The output value jump only takes effect when another setpoint is entered.
7. Click the "Start tuning" icon.
   The pretuning starts. The status of the tuning is displayed.

Note

During phase 1 a tuning process can also be triggered by minor setpoint changes (e.g. measurement noise at an analog input). In this case, tuning is terminated very quickly (wrong controller parameters; risk of instability).

Performing fine tuning for heating

Requirement

- The requirements on the process are met.
- An instance DB has been created for the channel and is supplied.
- "Instruction" is configured as source for the setpoint.
- The FM 355-2 is not in safety mode (SAFE_ON = FALSE). An ongoing tuning is cancelled when safety mode is switched on.

Procedure

To determine the optimal PID parameters at the operating point, follow these steps:

1. Select the instance DB of the channel you want to tune.
2. Click the "Measurement on" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value and output value are recorded.
3. Select "Fine tuning heating" from the "Mode" drop-down list.
   The FM 355-2 is ready for tuning (PHASE = 1).
4. In the "Output value jump" field, specify how much the output value should be increased.
5. Click the "Start tuning" icon.
   Fine tuning starts. The status of the tuning is displayed.

Performing cooling fine tuning

Introduction

For controllers operating with two counteracting final actuators (heating and cooling actuator), FM 355-2 determines the process gain ratio RATIOFAC (heating/cooling gain) after a manipulated variable jump using the cooling actuator. Even the control zone width CON_ZONE is recalculated. The other controller parameters remain the same.

Requirement

- "Instruction" is configured as source for the setpoint.
- The FM 355-2 is not in safety mode (SAFE_ON = FALSE). An ongoing tuning is cancelled when safety mode is switched on.
- A heating fine tuning of the channel has been successfully completed. Heating tuning has to be repeated after a voltage failure of the FM 355-2.
- In a plant with several thermally interconnected temperature zones (e.g. plastics machines) the heating fine tuning has been successfully completed for all zones and the steady state has been reached.
- Use manipulated variable A for heating and manipulated variable B for cooling.
- RATIOFAC is effective at LMN<0.0, which means the split-range function must be defined as follows: A for LMN>=0 .0 and B for LMN<0.0.
- To start cooling fine tuning during heating, the heating and cooling signals have to become effective in the process at the same time.

Procedure

To determine the optimal PID parameters at the operating point, follow these steps:

1. Select the instance DB of the channel you want to tune.
2. Click the "Measurement on" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value and output value are recorded.
3. Select "Fine tuning cooling" from the "Mode" drop-down list.
   The FM 355-2 is ready for tuning (PHASE = 1).
4. In the "Manipulated variable jump" field, specify how much the manipulated variable should be increased or decreased.
5. Click the "Start tuning" icon.
   Fine tuning starts. The status of the tuning is displayed.

Result

As soon as a point of inflection is detected in the process value curve, the FM 355-2 switches back to control mode (PHASE 0).

Contrary to heat tuning, the old controller parameters are not saved (the old RATIOFAC was already backed up during heat tuning) and phase 4 will not run through.

STATUS_D remains the same and still refers to heat tuning. Even the PI and PID parameter sets of the last heat tuning will be maintained so that you can still load them after cooling tuning with LOAD_PID.

Parallel tuning of controller channels

Neighboring zones with strong temperature coupling effect

When two or multiple controllers of a module are used to control the temperature, for example, on a plate (e.g. two heaters and two measured actual values with strong temperature coupling), you have the option of parallel controller tuning.

Procedure

Follow these steps to tune several channels at once:

1. Configure the channel group in Zone A or Zone B.
2. Start tuning in one of these channels. Tuning of the other channels will then be started automatically.

Note

The user must ensure that the setpoint jumps are set concurrently when the process is excited via setpoint jump. You can use the parameters PID_ON and TUN_DLMN/ TUN_CLMN for one channel only. You must specify the parameters for the other channels in the corresponding instance DBs of the FMT_PID instruction before you start tuning.

Tuning is aborted

Tuning of all channels is aborted if an error occurs in one of the channels (Phase 0 or end of phase 1).

The user can reset tuning by setting TUN_ON=FALSE at any of the participating channels.

When a channel goes into safety mode, tuning of all other participating channels is also switched off.

Only the status information of the selected channel is displayed. For all other channels, the causes of error are found in the corresponding instance DBs.

Advantage

Each participating controller will output its tuning manipulated variable until all controllers have exited phase 2. This avoids that the controller that completes tuning first corrupts the tuning result of the other controller due to the change in its manipulated variable.

CAUTION

When 75% of the setpoint jump is reached at one of the participating channels, tuning is not terminated (risk of overshoot). Automatic mode is only started after all participating controllers have terminated tuning.

Neighboring zones with weak temperature coupling effect

As a general rule, tuning should be carried out to reflect the way in which the control will work later. When zones are operated in parallel during production so that the temperature differences between the zones remain the same, the temperature level of the neighboring zones ought to be increased accordingly during tuning.

Temperature differences at the beginning of a trial are irrelevant since they will be compensated in the initial heating process (→initial rise = 0).

Manual fine tuning in control mode

Introduction

The following measures can be employed in order to achieve a non-overshooting response to setpoint changes:

Adapt control zone

During tuning, the FM 355-2 determines the control zone width CON_ZONE and, with an appropriate process type (process type I and II), a PID controller is activated: CONZ_ON = TRUE. During control mode, you can modify the control zone or switch it off completely (with CONZ_ON = FALSE).

No control zone for process type III, PI controller, step controller

Activating the control zone with higher order processes (process type III) does not normally bring any benefit since the control zone is then larger than the control range that can be achieved with a 100% manipulated variable. There is also no advantage in activating the control zone for PI controllers.

Note

Before you switch on the control zone manually, make sure that the control zone width is not too small. This means, the manipulated variable and actual value will oscillate if the control zone width is too small.

Weakening control response continuously with PFAC_SP

The control zone offers the best dynamic option of achieving a control response that is free of overshoot. Use it wherever possible, so for PID controllers for process types I and II.

For PI controllers, process type III or setpoint jumps within the control zone you can weaken the control response with parameter PFAC_SP. This parameter specifies the amount of P action that is affective for setpoint jumps.

Regardless of the process type, PFAC_SP is set to a default value of 0.8 by the tuning function; you can later modify this value if required. To limit overshoot during setpoint jumps (with otherwise correct controller parameters) to approximately 2%, the following values are adequate for PFAC_SP:

	Process type I	Process type II	Process type III
	Typical temperature process	Intermediate range	Higher order temperature process
PI	0.8	0.82	0.8
PID	0.6	0.75	0.96

Adapt the default factor (0.8) especially under the following situations:

- Process type I with PID (0.8 →0.6): With PFAC_SP = 0.8, setpoint jumps will still lead to approx. 18% overshoot.
- Process type III with PID (0.8 →0.96): With PFAC_SP = 0.8, the setpoint jumps is attenuated too heavily. This leads to a significant loss of tuning time.

Attenuation of control parameters

When a closed-loop control circuit oscillates or if overshoot occurs after setpoint jumps, you can reduce the controller's GAIN (e.g. to 80% of the original value) and increase the integration time TI (e.g. to 150% of the original value). If the analog manipulated variable of the closed-loop controller is converted to binary actuating signals by a pulse shaper, quantization noise may cause minor permanent oscillation. You can eliminate this effect by increasing the dead band width DEADB_W.

Tuning PID parameters manually

Requirement

- The requirements on the process are met.
- An instance DB has been created for the channel and is supplied.
- "Instruction" is configured as source for the setpoint.

Procedure

To set a controller manually, proceed as follows:

1. Select the instance DB of the channel you want to tune.
2. Click the "Measurement on" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value and output value are recorded.
3. Enter the respective values for the following parameters:
   - Proportional action
   - Proportional action weighting with setpoint jumps
   - Integral action
   - Derivative action
   - Coefficient DT1
   - Heating / cooling ratio

4. Click the icon "Send PID parameter to FM" in the "Tuning" group.
5. Select the "Change setpoint" check box in the "Current values" group.
6. Enter a new setpoint.
7. Deselect the check box "Manual mode", if necessary, and click the icon in the "Current Values" group.
   The controller works with the new parameters and controls the new setpoint.
8. Check the quality of the PID parameter to check the curve points.
9. Repeat steps 3 to 8 until you are satisfied with the controller results.

Note

You can use FMT_PID.OP.UNDO_PAR = TRUE to set the old controller parameters again.

Downloading tuned parameters in device

Introduction

You can proceed as follows to prevent the parameters on the FM from being overwritten with the system data the next time you start up the CPU:

- You download the tuned PID parameters from the instance DB of the FMT_PID to the FM after each startup of the FM 355-2 with LOAD_PAR=TRUE.
- Download the PID parameters to the SDB.

Note

The PI, PID and SAVE parameter records cannot be stored in the SDB.

Procedure

Proceed as follows to download the parameters to the SDB:

1. Click on the icon "Transfer online data to an offline project".
   The PID parameters are saved in the parameter setting of the FM.
2. Stop CPU.
3. Click "Download to device".

Saving and retrieving controller parameters

Back-up file records

The FM 355-2 features an effective parameter record system as well as a backed up parameter record system. Save and retrieval operations are performed by means of SAVE_PAR or UNDO_PAR in OP structure of the FMT_PID instruction.

For example, prior to a manual change you can save the current parameters via SAVE_PAR = TRUE. You can retrieve the last saved controller parameters via UNDO_PAR = TRUE and enable them for the controller. SAVE_PAR or UNDO_PAR is reset by FMT_PID at the end of this action.

If you want to know the values in the current parameter record or in the backed up parameter record, you can retrieve this information via FMT_PID.READ_PAR = TRUE or FMT_TUN.READ_OUT = TRUE:

Current parameters in the PAR structure of FMT_PID:	Parameters included in the backup of the FMT_TUN output parameters:
PFAC_SP	SAV_PFAC
GAIN	SAV_GAIN
TI	SAV_TI
TD	SAV_TD
D_F	SAV_D_F
CON_ZONE	SAV_CONZ
RATIOFAC	SAV_RATI
CONZ_ON	SAV_CZON
P_SEL	SAV_PSEL

Note

At the end of controller tuning, the saved parameters are overwritten with parameter data that were valid prior to tuning (Exception: this intermediate saving operation is discarded in a cooling tuning process).

Change between PI and PID parameters

After tuning, the PI and PID parameters are saved on the FM355-2. You load these parameter records by setting FMT_PID.OP.LOAD_PID. If PID_ON in the structure PAR = TRUE, the PID parameter record is copied to the effective controller parameters; otherwise, the PI parameter record is copied. LOAD_PID is reset by FMT_PID at the end of this action. Of course, the value of PID_ON is retained, as it is not an operator control parameter.

If you want to know which values are stored in the PI or PID parameter record, you can view these data via FMT_TUN. Read out READ_OUT = TRUE:

	PI parameter record	PID parameter record
Controller GAIN	PI_GAIN	PID_GAIN
Integration time	PI_TI	PID_TI
Derivative time	0.0	PID_TD

Note

Please note the following:

- Controller parameters are only written back with UNDO_PAR or LOAD_PID if the controller gain (SAV_GAIN, PID_GAIN or PI_GAIN) is not equal to zero:
- This strategy takes into account the situation that no tuning has yet been made or that PID parameters are missing. For example, if PID_ON was TRUE and PID_GAIN = 0.0, PID_ON is set to FALSE and the PI parameters will be copied.
- D_F, PFAC_SP are set to default values by the tuning. The user can later modify them. LOAD_PID does not modify these parameters.
- With LOAD_PID, the control zone is always recalculated (CON_ZONE = 250/GAIN or 375/GAIN if RATIOFAC <> 0.0), irrespective of the status CONZ_ON = FALSE.

Pulse-free changeover with LOAD_PID and UNDO_PAR

The changeover is performed so that the sum of the proportional action and integral action stays equal. A changeover from PID to PI triggers a jump of the manipulated variable, as the derivative action is disabled. Initially, the changeover from PI to PID is pulse-free. After a few cycles, however, the derivative action will lead to a fast change.

Measuring motor actuating time

Introduction

The motor transition time has to be as accurate as possible for good controller results. The data in the actuator documentation contains average values for this type of actuator. The value for the specific actuator used may differ.

You can measure the motor transition time during commissioning if you are using actuators with position feedback or endstop signals. The motor transition time cannot be measured if neither position feedback nor endstop signals are available.

Requirement

- The configured motor transition time must be half the size of the real transition time. The actuator is moved for a maximum of twice the transition time in one direction.
- The position feedback must be interconnected for actuators with analog position feedback.
- The endstop signals must be interconnected for actuators with endstop signals.

Actuators with analog position feedback

Proceed as follows to measure motor transition time with position feedback:

1. Select the instance DB for which you want to measure the motor transition time.
2. Select the "Use position feedback" check box.
3. Enter the position to which the actuator is to be moved in the "Target position" input field. The current position feedback (starting position) will be displayed. Click the "Start transition time measurement" icon.

Result

The actuator is moved from the starting position to the target position. Time measurement starts immediately and ends when the actuator reaches the target position. The motor transition time is calculated according to the following equation:

Motor transition time = (output value high limit – output value low limit) × Measuring time / AMOUNT (target position – starting position).

The progress and status of transition time measurement are displayed. The measured transition time is displayed in the field "Measured transition time". Once transition time measurement is complete, the manual mode of the transition signals is switched on (OP.LMNS_ON = TRUE). The actuator will stop being moved.

Actuators with endstop signals

Proceed as follows to measure motor transition time with endstop signals:

1. Select the instance DB for which you want to measure the motor transition time.
2. Select the "Use actuator endstop signals" check box.
3. Click the "Start transition time measurement" icon.

Result

The actuator is moved first to the low endstop, then to the high endstop and then back to the low endstop. Time measurement will start once the actuator has reached the lower endstop and will end when the actuator reaches the lower endstop for the second time. The motor transition time is equal to the time measured divided by two.

The progress and status of transition time measurement are displayed. The measured transition time is displayed in the field "Measured transition time". Once transition time measurement is complete, the manual mode of the transition signals is switched on (OP.LMNS_ON = TRUE). The actuator will stop being moved.

Loading measured transition time

1. Click "Transfer online data to an offline project" to save the measured motor transition time locally in the project.
   The measured transition time is saved in the parameter setting of the FM.
2. Click "Download to device" to activate all changes the next time you start the CPU.

Cancelling transition time measurement

When you cancel the transition time measurement, the manual mode of the actuator signals is switched on (OP.LMNS_ON = TRUE) and the manual mode with manipulated value specification is switched off (OP.LMN_REON = FALSE). The actuator will stop being moved.

Working with the trend view

Monitoring

1. Double click on "Commissioning" in the device view of the module.
2. Navigate to "Trend view" in the configuration view.
3. Select the instance DB you wish to monitor. The channel to which the instance DB has been assigned will be displayed.
4. Specify the update cycle in which the trend view is to record.
5. Click on "Measurement on" to start measuring.
6. Click on "Measurement off" to stop measuring.

Procedure

Proceed as follows to adjust the scale:

1. Double click on "Commissioning" in the device view of the module.
2. Navigate to "Trend view" in the configuration view.
3. To change the time axis:
   - Click on the arrow button next to the date to open a calendar. Select a date from the calendar. To lock the selection, click on the blue lock. This can be done for both start and end dates.
   - Click on the time scale. The cursor will appear as a left/right arrow. Hold down the cursor and drag it to the left or right.
     Time scaling will change if the start or end time is locked.
     If neither the start nor end time is locked, the specified period will change and the scaling remain the same.
4. To change the value axis:
   - Click on the input field for the high and low value displayed. Enter the desired value here. To lock the selection, click on the blue lock. This option is available for both the high and low value.
   - Click on the value scale. The cursor will appear as an up/down arrow. Hold down the cursor and drag it up or down.
     If the high or low value is locked, the scaling of the value axis will change.
     If neither the high nor the low value is locked, the specified range will change and the scaling remain the same.

FM 355-2: Diagnostics with LED display

Error display from the group error LED

Introduction

If the red group error LED lights up, there is either an error on the module (internal error) or in the line connections (external error).

If the yellow LED flashes, then the firmware has been deleted. This status can only occur in the case of faulty hardware or if the loading procedure of the firmware is aborted.

Which errors are displayed?

The following errors are displayed by the group error LED lighting up:

Type of error	Diagnostic message	Possible cause	Correction
Internal errors	Module defective	Hardware error	Replace the module
	Time watchdog tripped	Hardware error	Replace the module
	EEPROM content is invalid	Failure of the supply voltage when configuring	Reconfigure module
External errors	Incorrect parameters in module	Incorrect parameters have been transferred to the module	Reassign the module parameters
	Errors with the analog inputs or analog outputs	Analog input hardware error	Replace module
		Analog input wire break	Remedy wire break
		Analog input measuring range violation (underrange)	Check measuring signal
		Analog input measuring range violation (overrange)	Check the measuring signal
		Analog output wire break	Remedy wire break
		Analog output short circuit	Eliminate short circuit
	Missing external auxiliary supply	24 V supply missing	Restore 24 V supply

Diagnostic interrupt in the case of errors

All errors can trigger a diagnostic interrupt if you have enabled the diagnostic interrupt in the respective parameterization screen. From the diagnostic data records DS0 and DS1 you can see which errors have caused the LED to light up. The assignment of the diagnostic data records DS0 and DS1 is described in the next section.

Triggering diagnostic interrupts

What is a diagnostic interrupt?

If the user program is to react to an internal fault or external error, you can configure a diagnostic interrupt. The cyclical CPU program is interrupted and the diagnostic interrupt OB, OB 82 is called.

Which events may trigger a diagnostic interrupt?

The list shows you which events can trigger a diagnostic interrupt:

- Module configuration missing or faulty
- Module defective
- Wire break when analog inputs are made (4 up to 20 mA only)
- High limit violated and low limit violated during analog inputs
- Load break or short-circuit during analog outputs

Enabling the diagnostic interrupt

You can enable or disable the diagnostic interrupt for the module in the basic parameter tab.

The default setting for the diagnostic interrupt is disabled.

Reactions to an event triggering an interrupt

If an event occurs that can trigger a diagnostic interrupt, the following occurs:

- The diagnostics information is written to the diagnostics data records DS0 and DS1 on the module.
- The group error LED lights up.
- Diagnostic interrupt OB is called (OB 82).
- Diagnostic data record DS0 is written to the start information of diagnostic interrupt OB.
- If there is no hardware defect, the module resumes control operation.

If no OB 82 is programmed, the CPU goes into STOP.

Diagnostics data record DS0 and DS1

The information about which event triggered a diagnostic interrupt is written to the diagnostic data records DS0 and DS1. The length of the diagnostic data record DS0 is four bytes; the record length of DS1 is 16 bytes of which the first 4 bytes are identical with DS0.

Diagnostic data records DS0 and DS1

Read data record from the module

The diagnostic data record DS0 is automatically transferred to the start information when the diagnostic OB is called. There these four bytes are stored in the local data (byte 8 - 11) of the OB 82.

The diagnostic data record DS1 (and hence also the content of the DS0) can be read from the module by means of FB 55 "FMT_DS1". This only makes sense if an error is reported in a channel in the DS0.

The FB 55 "FMT_DS1" must be called in the same OB as an FB FMT_PID that might be present (e.g. OB 35). You can do this as follows: While OB 82 is being processed, set the READ_DS1 bit. The FMT_DS1 in OB 35 then reads the diagnostic data record DS1.

How does the diagnosis text appear in the diagnostics buffer?

If you want to enter the diagnostics message in the diagnostics buffer, call the "WR_REC" instruction "Enter user-specific message in the diagnostics buffer" in the user program. The event number of the respective diagnosis message is specified on the input parameter EVENTN. The interrupt is entered in the diagnostics buffer defined as x = 1 for incoming and x = 0 for outgoing. In addition to the time of the entry, the appropriate diagnosis text is displayed in the "Meaning" column.

The assignment of the diagnostic data records DS0 and the Start information

The following table shows the assignment of the diagnostic data record DS0 in the Start information. Any bits not listed have no meaning and are zero.

Table 11-9 Assignment of diagnostic data record DS0

Byte	Bit	Meaning	Remark	Event no.
0	0	Module fault	Is set at each diagnostics event	8:x:00
	1	Internal error	Is set at every internal error: • Watchdog timeout • EPROM error • ADC/DAC error • Analog input hardware error	8:x:01
	2	External error	Is set at every external error: • Missing external auxiliary voltage • Faulty parameter assignment • Analog input wire break (only in range 4 to 20 mA) • Analog input below measuring range • Analog input measuring range exceeded • Analog output load break • Analog output short circuit	8:x:02
	3	Error in one channel	See DS1, from byte 7 for further breakdown	8:x:03

Byte	Bit	Meaning	Remark	Event no.
	4	Missing external auxiliary voltage	24 V supply for FM 355-2 has failed	8:x:04
	6	Unused	-	-
	7	Faulty parameter assignment	The module is unable to use a parameter. Reason: Parameters unknown or illegal combination of parameters. See menu PLC > Parameter assignment error display.	8:x:07
1	0 ... 3	Module class	8 is always assigned.	-
	4	Channel-specific diagnosis	If set, the module can provide additional channel information and a channel error exists (see DS1, byte 7 to 12)	-
2	3	Time watchdog responds	Hardware fault	8:x:33
3	2	EPROM error	Module defective	8:x:42
	4	ADC/DAC error	Module defective	8:x:44

Diagnostic data record DS1

The length of diagnostic data record DS1 is 16 bytes. The first 4 bytes are identical to diagnostic data record DS0. The table below shows the assignment of the remaining bytes. Any bits not listed have no meaning and are zero.

Diagnostic data record DS1 of the FM 355-2

Table 11-10 Assignments of bytes 4 to 12 of diagnostic data record DS1 of the FM 355-2

Byte	Bit	Meaning	Remark		Event no.
4	0 ... 7	Channel type	75H is always assigned.		-
5	0 ... 7	Length of the diagnostics information	8 is always assigned.		-
6	0 ... 7	Number of channels	5 always assigned (4 controllers + 1 reference channel)		-
7	0 ... 4	Channel error vector	One bit is assigned to each channel (0...3; 4 for reference channel).		-
8	0	Analog input hardware error	Channel-specific diagnostics channel 0		8:x:B0
	1	Unused			8:x:B1
	2	Analog input wire break (only in range 4 to 20 mA)			8:x:B2
	3	Unused			8:x:B3
	4	Analog input below measuring range			8:x:B4
	5	Analog input measuring range exceeded			8:x:B5
	6	Analog output wire breakage		Only with the current output of the C controller	8:x:B6

Byte	Bit	Meaning	Remark		Event no.
	7	Analog output short circuit		Only with the voltage output of the C controller	8:x:B7
9	0 ... 7	see Byte 8	Channel-specific diagnostics channel 1		See above
10	0 ... 7	see Byte 8	Channel-specific diagnostics channel 2		See above
11	0 ... 7	see Byte 8	Channel-specific diagnostics channel 3		See above
12	0 ... 5	see Byte 8	Diagnostics for reference channel		See above

Additional information

For more information on the "RET_VALU" parameter, refer to List of RET_VALU messages (Page 9915) and the "Extended instructions" section of the help system.

Measuring transducer fault

Measuring transducer faults

The FM 355-2 can identify the following measuring transducer faults:

- Measuring range violation (underrange)
- Measuring range violation (overrange)
- Wire break (not for all measuring ranges)

If one of these faults occur, the group error bit "external error" is set in diagnostic data record DS0 and channel-specific error bits are set in diagnostic data record DS1 (see table above). When these faults disappear, the respective bits are reset.

The following table shows the specific measuring range limits at which the error bits are set or reset:

Measuring range	Error bit measuring range violation (underrange) at...	Error bit measuring range violation (overrange) at...	Error bit wire break display
	DS1: Byte 10 to 26, bit 4	DS1: Byte 10 to 26, bit 5	DS1: Byte 10 to 26, bit 2
0 to 20 mA	< - 3.5 mA	> 23.5 mA	-
4 to 20 mA	Error bit = 1 at < 3.6 mA Error bit = 0 at > 3.8 mA	> 22.8 mA	Error bit = 1 at < 3.6 mA Error bit = 0 at > 3.8 mA
0 to 10 V	<- 1.175 V	> 11.75 V	-
Pt 100 (-200 to 850 °C) (-328 to 1562 °F)	< 30.82 mV	> 650.46 mV	-

Measuring range	Error bit measuring range violation (underrange) at...	Error bit measuring range violation (overrange) at...	Error bit wire break display
	DS1: Byte 10 to 26, bit 4	DS1: Byte 10 to 26, bit 5	DS1: Byte 10 to 26, bit 2
Pt 100 (-200 to 556 °C) (-328 to 1032 °F)	< 30.82 mV	> 499.06 mV	-
Pt 100 (-200 to 130 °C) (-328 to 264 °F)	< 30.82 mV	> 254.12 mV	-
Type B thermocouple	< 0 mV	> 13.81 mV	-
Type E thermocouple (-270.0 to 1,000.03 °C)	< -9.84 mV	> 76.36 mV	-
Type J thermocouple	< -8.1 mV	> 69.54 mV	-
Type K thermocouple	< -6.45 mV	> 54.88 mV	-
Type R thermocouple	< -0.23 mV	> 21.11 mV	-
Type S thermocouple	< -0.24 mV	> 18.7 mV	-
Free thermocouple	< low input value of the polyline	> high input value of the polyline	-

List of RET_VALU messages

RET_VALU messages

JOB_ERR (Hex)	JOB_ERR (Dec)	JOB_ERR (Int)	Meaning
7000	28672	-32624	First call with REQ=0: no data transmission active; BUSY has the value 0.
7001	28673	-32624	First call with REQ=1: data transmission initiated; BUSY has the value 1.
7002	28674	-32624	Interim call (REQ irrelevant). Data transmission already active; BUSY has the value 1.
8090	32912	-32624	Specified logical base address invalid: There is no assignment in the SDB1/SDB2x, or it is not a base address.
80A0	32928	-32608	Negative acknowledgment when reading from the module. Module was removed during the read operation or the module is defective.
80A1	32929	-32607	Negative acknowledgment when writing to the module. Module was removed during the write operation or the module is defective.
80A2	32930	-32606	DP protocol error at layer 2
80A3	32931	-32605	DP protocol error in user interface/user
80A4	32932	-32604	Communication bus error
80B1	32945	-32591	Incorrect length specification. FM_TYPE parameter in channel DB not set correctly for the module in use.
80B2	32946	-32590	The configured slot is not being used.
80B3	32947	-32589	Actual module type is not match configured module type.

JOB_ERR (Hex)	JOB_ERR (Dec)	JOB_ERR (Int)	Meaning
80C0	32960	-32576	Module data not ready for reading.
80C1	32961	-32575	Data of a write job of the same type have not yet been processed by the module.
80C2	32962	-32574	The module is currently processing the maximum possible number of jobs.
80C3	32963	-32573	Required resources (memory etc.) currently occupied.
80C4	32964	-32572	Communication error
80C5	32965	-32571	Distributed I/O not available.
80C6	32966	-32570	Priority class abort (restart or background).
8522	34082	-31454	Channel DB or parameter DB too short. The data cannot be read off the DB. (Write job)
8532	34098	-31438	DB number of the parameter DBs too high. (Write job)
853A	34106	-31430	Parameter DB not present. (Write job)
8544	34116	-31420	Error at n-th (n > 1) read access to a DB after an error has occurred. (Write job)
8723	34595	-30941	Channel DB or parameter DB too short. The data cannot be written to the DB. (Read job)
8730	34608	-30928	Parameter DB in the CPU is write protected. The data cannot be written to the DB (read job)
8732	34610	-30926	DB number of the parameter DBs too high (Read job)
873A	34618	-30918	Parameter DB not present. (Read job)
8745	34629	-30907	Error at n-th (n > 1) write access to a DB after an error has occurred. (Read job)
80ff	33023	-32513	Incorrect index specification with block FMT_PAR
Errors 80A2..80A4 and 80Cx are temporary, i.e. after a waiting period they can be eliminated without any action on your part. Messages of the 7xxx form indicate temporary operating states of communication.			

11.3.5.7 Using FM 455

Configuring FM 455

Configuring the hardware

Procedure

Configuration assumes that you have set up a project in which you can save the parameter assignment. The following briefly explains the most important steps.

1. Start the TIA portal and open the device configuration.
2. Select one module rack.
3. Select one controller module from your module catalog.

4. Move the controller module to the appropriate place on the module rack.
5. From the configuration table, note the input address of the module, for example 512.
   The value that you read is displayed in decimal format.

Parameter assignment FM 455

Module parameters

Procedure

When assigning parameters you set the module parameters.

1. Double-click on the "Parameter setting" in the device view of the module. In the Inspector window, select the "Properties" tab.
   Result: The properties are displayed in the inspector window.
2. Configure the basic parameters and the remaining parameters of the module.
3. Configure the remaining device parameters in the Design view.
4. Save the parameters entered with "Project> Save".
5. Translate the project with "Edit > Translate".
6. Load the configuration data with the CPU in STOP mode by selecting "Online > Device load..."
   Result: The data stored in the memory of the CPU are then transferred directly to the module.
7. Carry out a CPU start-up.

Analog input parameter

Evaluation of analog inputs

Procedure

To select a sensor type, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Digital input > input [No]> Evaluation of analog inputs".
3. Enter the following parameters:
   - Analog input resolution
   - Sensor type analog input
     For different sensor types the option "open-circuit monitoring" will be enabled. The option may be activated depending on requirements.
   - Junction temperature
4. Save the parameters entered with "Project> Save".

Filter functions

Procedure

To set the filter functions, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Analog input > input [No]> Filter functions".
3. Enter the following parameters:
   - To filter on a time constant you should enable the "Signal filter" option.
     If the "signal filter" option is selected, the time constant filter can be set.
   - Filter time constant
     Can only be set if the "signal filter" option is enabled.
   - To linearize the encoder signal you should activate the "square root filter" option.
   - To linearize the encoder signal characteristics you should activate the "polygon filter" option.
4. Save the parameters entered with "Project> Save".

Signal filter

Filter 1. order, whose time behavior is set through the time constants.

Square root filter

Used to linearize encoder signals in which the actual value is present as a physical quantity, which is square in relation to the measured process variable.

Polygon filter

Used to linearize encoder characteristics.

The 13 selectable points are selectable in:

- mA at current input
- mV with voltage input

Scaling

Procedure

To convert the input signal into a physical unit, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Analog input > input [No]> Scaling".

3. Enter the following parameters:
   - High value
   - Low value
4. Save the parameters entered with "Project> Save".

Scaling

To convert the input signal into another physical unit through linear interpolation between the initial value (low value) and the ending value (high value).

Deviation

Implementing a constant or a cascade controller

Constant or cascade controller

In the controller structure of constant or cascade controllers the effective current value is equal to the actual value A.

Selecting the controller type

To implement a constant or cascade controller, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]".
3. In the "Encoder type" section select the "Constant or cascade controller".
4. Save the parameters entered with "Project> Save".

Processing setpoints

To edit setpoints, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> Channel [No]> Deviation".

3. Set in the different areas the corresponding parameters:
   - Setpoint source
     Select the source of the setpoint.
   - Safety setpoint
     Set your safety setpoints here. Set also the response to a CPU fault and the startup behavior.
   - Setpoint ramp
     Set the start-up time.
   - Setpoint limit
     Select the high and low setpoint limits.
4. Save the parameters entered with "Project> Save".

Selecting a signal source for actual value A

To select the signal source for the actual value A, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> Channel [No]> Deviation".
3. In the "Control value" section select the corresponding source. The following choices are available:
   - Zero
   - Processed analog input
4. Save the parameters entered with "Project> Save".

Implementing three component controllers

Three-term controller

In the controller structure of a three-term controller, the effective actual value is obtained by totaling the three actual values A, B and C and then adding an adjustable offset. The actual values B and C can also be evaluated using factors.

Selecting the controller type

Proceed as follows to implement a three-term controller:

1. Double-click on "Parameter setting" in the device view of the module.
2. Navigate in the configuration view to "Controller> channel [No]".
3. In the "Controller type" section, select the "Three-term controller" option.
4. Save the parameters entered with "Project > Save".

Processing setpoints

Proceed as follows to adjust the setpoint processing mode:

1. Double-click on "Parameter setting" in the device view of the module.
2. Navigate in the configuration view to "Controller > Channel [No] > System deviation".
3. In the "Setpoint source" section, select the source from which the setpoint is to be obtained.
4. In the "Safety setpoint" section, enter a value to be utilized upon CPU fault and start up as a safety value.
5. Select the behavior of the controller module for "Reaction to CPU errors" and "Start-up behavior". You may select from amongst the following options:
   - Setpoint = last valid setpoint: the setpoint will be assigned the last valid setpoint upon occurrence of the event. The last valid setpoint will remain unchanged.
   - Setpoint = safety setpoint: the setpoint will be assigned the safety setpoint upon the occurrence of the event. This enables secured emergency operation.
6. In the "setpoint ramp" section, set the rise time to be adhered to upon a setpoint change to reduce overshoot.
7. In the "Setpoint limits" section, set both an high and a low limit for the setpoint.
8. Save the parameters entered with "Project > Save".

Selecting a signal source for actual values A, B and C

Proceed as follows to set the signal sources for the actual values A, B and C:

1. Double-click on "Parameter setting" in the device view of the module.
2. Navigate in the configuration view to "Controller > Channel [No] > System deviation".
3. Select the signal source for each actual value in the "Manipulated value" section. You can select whether you use no signal ("zero") as a source or one of the processed analog inputs.
4. Save the parameters entered with "Project > Save".

Determining factors for actual values B and C and the offset value

Proceed as follows to set the signal sources for the actual values A, B and C:

1. Double-click on "Parameter setting" in the device view of the module.
2. Navigate in the configuration view to "Controller > Channel [No] > System deviation".
3. In the"Total" section, set the necessary weighting factor for the actual values B and C.
4. Set the required "offset" value to be added to the actual values.
5. Save the parameters entered with "Project > Save".

Implementing ratio or mixture controllers

Selecting the controller type

Proceed as follows to implement a ratio or blending controller:

1. Double-click on "Parameter setting" in the device view of the module.
2. Navigate in the configuration view to "Controller> channel [No]".
3. Select the "Ratio or blending controller" in the "Controller type" section.
4. Save the parameters entered with "Project > Save".

Processing setpoints

Proceed as follows to adjust the setpoint processing mode:

1. Double-click on "Parameter setting" in the device view of the module.
2. Navigate in the configuration view to "Controller > Channel [No] > System deviation".
3. In the "Setpoint source" section, select the source from which the setpoint is to be obtained.
4. In the "Safety setpoint" section, enter a value to be utilized upon CPU fault and start up as a safety value.
5. Select the behavior of the controller module for "Reaction to CPU errors" and "Start-up behavior". You may select from amongst the following options:
   - Setpoint = last valid setpoint: the setpoint will be assigned the last valid setpoint upon occurrence of the event. The last valid setpoint will remain unchanged.
   - Setpoint = safety setpoint: the setpoint will be assigned the safety setpoint upon the occurrence of the event. This enables secured emergency operation.
6. In the "setpoint ramp" section, set the rise time to be adhered to upon a setpoint change to reduce overshoot.
7. In the "Setpoint limits" section, set both an high and a low limit for the setpoint.
8. Save the parameters entered with "Project > Save".

Selecting the signal source for actual values A and D

Proceed as follows to set the signal sources for actual values A and D:

1. Double-click on "Parameter setting" in the device view of the module.
2. Navigate in the configuration view to "Controller > Channel [No] > System deviation".
3. Select the signal source for each actual value in the "Manipulated value" section. You can choose whether to use no signal ("zero") as a source or one of the processed analog inputs. For the actual value D, you may also select the manipulated value of one of the controllers.
4. Save the parameters entered with "Project > Save".

Determining multipliers

Proceed as follows to set the signal sources for actual values A and D:

1. Double-click on "Parameter setting" in the device view of the module.
2. Navigate in the configuration view to "Controller > Channel [No] > System deviation".
3. Select the desired offset in the "Multiply" section.
4. Save the parameters entered with "Project > Save".

For ratio controllers, the actual value A is used as control variable while the actual value D stores the ratio magnitude. The setpoint input value is used as a ratio factor. Multiplying it by the actual value of D and adding the adjustable offset will yield the effective setpoint value. If actual value D is off, only the offset will be added to the setpoint.

Setting the alarm generation

Procedure

Proceed as follows to set interrupt generation:

1. Double-click on "Parameter setting" in the device view of the module.
2. Navigate in the configuration view to "Controller > Channel [No] > System deviation".
3. Select the necessary options in the "Generate interrupt" section:
   - Enter the signal for which the alarm is to be set in the "Limit monitor for" section.
   - In "Alarm high limit", set the high limit beyond which the alarm is to be triggered.
   - In "Warning high limit", set the value beyond which the "approaching high limit" warning is to be issued.
   - In "Warning low limit", set the value below which the "approaching low limit" warning is to be issued.
   - In "Alarm low limit", set the low limit below which the alarm is to be triggered.
   - You can set hysteresis under "Hysteresis" to prevent the monitor flickering.
4. Save the parameters entered with "Project > Save".

Control algorithms

Implementing a P control

Procedure

To implement a P control, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "Type" section select the "PID controller" option.

4. In the "General" section under "Operation mode" select the "P" option.
   All configurable options for a P control will be activated and all other options disabled.
5. Save the parameters entered with "Project> Save".

Implementing a PI control

Procedure

To implement a PI control, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "Type" section select the "PID controller" option.
4. In the "General" section under "Operation mode" select the "PI" option.
   All configurable options for a PI control will be activated and all other options disabled.
5. Save the parameters entered with "Project> Save".

Implementing an I control

Procedure

To implement an I control, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "Type" section select the "PID controller" option.
4. In the "General" section under "Operation mode" select the "I" option.
   All configurable options for an I control will be activated and all other options disabled.
5. Save the parameters entered with "Project> Save".

Implementing a PID control

Procedure

To implement a PID control, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "Type" section select the "PID controller" option.
4. In the "General" section under "Operation mode" select the "PID" option.
   All configurable options for a PID control will be activated and all other options disabled.
5. Save the parameters entered with "Project> Save".

Implementing a PD control

Procedure

To implement a PD control, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "Type" section select the "PID controller" option.
4. In the "General" section under "Operation mode" select the "PD" option.
   All configurable options for a PD control will be activated and all other options disabled.
5. Save the parameters entered with "Project> Save".

Implementing a Fuzzy temperature controller

Procedure

To implement a fuzzy temperature control, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "Type" section select the "Fuzzy temperature controller module" option.
   All configurable options for a fuzzy temperature controller module will be activated and all other options disabled.
4. Save the parameters entered with "Project> Save".

Suppressing the deviation noise component

Procedure

To suppress the noise component in the deviation, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "General" section, set the "Dead zone width" parameter.
4. Save the parameters entered with "Project> Save".

Switching disturbances

Procedure

To switch a disturbance, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. From the "Disturbance source" list in the "General" section select the disturbance source. Select the entry "zero" to perform no disturbance switching.
4. Enter the "Disturbance factor" value you wish to use as a disturbance weight factor. A value <0 weakens the disturbance whereas a value> 0 increases the disturbance. A value of 1 keeps the disturbance unchanged.
5. Save the parameters entered with "Project> Save".

Setting the feedback P portion

Procedure

To implement a constant or cascade controller, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "P portion" section select the "In feedback" option. If the option is enabled, the P portion is set in feedback.
4. Save the parameters entered with "Project> Save".

Attenuating the D portion for setpoint changes

Selecting the controller type

To implement a constant or cascade controller, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "D portion" section enter the D portion source in the "D portion" option.
4. In the "Differentiation factor" section, set the D portion factor.
5. Save the parameters entered with "Project> Save".

Reversing the controller effect

Principle

Controller reversal may be achieved by setting a negative proportional gain thus moving from "increasing deviation value = increasing correction value" to "increasing deviation value = decreasing correction value".

Procedure

To reverse the controller effect, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> Controller algorithm".
3. In the "General" section, set a proportional gain of opposite sign.
   As an example, if the current setting shows a positive proportional gain and you wish to reverse the controller effect, set a negative proportional gain.
4. Save the parameters entered with "Project> Save".

Controller output

Implementing a continuous controller

A continuous controller can be created only with an analog module.

Procedure

1. Changing an external control value (Page 9928)
2. Setting the safety control value (Page 9930)
3. Changing the tracking (Page 9929)
4. Setting control value limits (Page 9930)
5. Controlling two actuators - split range (Page 9931)
6. Selecting the output signal and the signal type (Page 9932)

Realizing a step controller without position feedback

A step controller without position feedback can be realized only with a digital module.

Procedure

1. Select controller output step controller without position feedback
2. Changing an external control value (Page 9928)

3. Setting the safety control value (Page 9930)
4. Using the pulse shaper step controller (Page 9931)

Realizing a step controller with position feedback

A step controller with position feedback can be realized only with a digital module.

Procedure

1. Select controller output step controller with position feedback
2. Changing an external control value (Page 9928)
3. Changing the tracking (Page 9929)
4. Setting the safety control value (Page 9930)
5. Setting control value limits (Page 9930)
6. Setting position feedback (Page 9929)
7. Using the pulse shaper step controller (Page 9931)

Realizinga pulse controller

A pulse controller can be realized only with a digital module.

Procedure

1. Select controller output pulse controller
2. Changing an external control value (Page 9928)
3. Changing the tracking (Page 9929)
4. Setting the safety control value (Page 9930)
5. Setting control value limits (Page 9930)
6. Prepare the analog-to-digital signal conversion
7. Converting analog into digital signal (Page 9932)

Assigning parameters for the functions of the controller output

Changing an external control value

Procedure

For changing the external control variable, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> controller output".

3. In the "External control variable" select the option to activate the external control variable. You may choose from the following options:
   - Activation of the function module.
   - Activation of a function module or one of the digital inputs.
4. Save the parameters entered with "Project> Save".

Changing the tracking

Procedure

To change the tracking proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> controller output".
3. In the "Tracking" section select the option to activate the external control variable. You may choose from the following options:
   - Activation of the function module.
   - Activation of a function module or one of the digital inputs.
4. In the "Tracking" section select the source of the signal you wish to track. You may choose from the following options:
   - Zero, while no source is used.
   - One of the analog inputs.
5. Save the parameters entered with "Project> Save".

Setting position feedback

Procedure

To set the position feedback, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to 'Digital Output> output [No.] ".
3. In the "Position feedback input" section select the "Source" option to choose the signal for position feedback.
   - Zero: no position feedback.
   - One of the processed analog inputs.
4. Save the parameters entered with "Project> Save".

Setting the safety control value

Procedure

To set the safety control value, proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> controller output".
3. In the "safety control value" section select the value that should be used as a safety control value.
4. Under "Activation" select the option to activate the external control variable. You may choose from the following options:
   – Activation of the function module.
   – Activation of a function module or one of the digital inputs.
5. Under "Action on module start" select the option which should be executed as the module starts. You may choose from the following options:
   – Control mode: sets the module immediately into control mode upon startup.
   – Safety manipulated value: initially, the safety manipulated value is controlled.
6. Under "Action on controller interference of the actual value A" select the option which should be executed as a controller interference of the actual value A occurs.
7. Under "Action on input controller interference" select the option which should be executed as an input controller interference occurs.
8. Save the parameters entered with "Project> Save".

Setting control value limits

Procedure

To set control value limits you should proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> controller output".
3. In the "Output signal limit" enter the following values:
   – High value: illustrates the high limit for the output signal.
   – Low value: illustrates the low limit for the output signal.
4. Save the parameters entered with "Project> Save".

Controlling two actuators - split range

Procedure

To control two actuators you should proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to "Controller> channel [No]> controller output".
3. In the "Split range" section select the option "Enable" to effectively control two actuators.
4. Set the following values for both the input and output signals A and for the input and output signals B:
   - Low input signal.
   - High input signal.
   - Low output signal.
   - High output signal.
5. Save the parameters entered with "Project> Save".

Using the pulse shaper step controller

Procedure

To configure the pulse shaper you should proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate in the design view to 'Digital Output> output [No.] ".
3. In the "Pulse shaper" area, enter the following parameters:
   - Engine operating time.
   - Minimum pulse duration.
   - Minimum break duration.
4. Select the best acceptable option for the high and the low values:
   - From the function module.
   - From the function module or one of the digital inputs.
5. Save the parameters entered with "Project> Save".

Converting analog into digital signal

Procedure

To convert an analog signal into a digital signal you should proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate to the "Digital output> channel [No.]" in the design view.
3. In the "Pulse shaper" area, enter the following parameters:
   – Minimum pulse duration.
   – Minimum break duration.
4. Save the parameters entered with "Project> Save".

Selecting the output signal and the signal type

Procedure

To select the output signal and the signal type you should proceed as follows:

1. Double-click on the "Parameter setting" in the device view of the module.
2. Navigate to the "Digital output> channel [No]" in the design view.
3. Under the option "Source" select the signal you wish to use as output signal. You may choose among the following options:
   – Zero: no source for the output signal.
   – One of the analog inputs.
   – One of the control values for controller A or B.
4. Under the option "Signal type" select the type for the output signal you wish to use.
5. Save the parameters entered with "Project> Save".

Programming FM 455

Commissioning FM 455

Creating and supplying instance DBs

Procedure

To meet the requirements for commissioning, follow these steps:

1. Create a user program that calls up the PID_FM_455 instruction cyclically in a cyclic interrupt OB.
2. Set READ_VAR = TRUE cyclically in the user program.

3. Create an individual instance DB of the PID_FM_455 instruction for each controller channel.
4. Enter the module address for each instance DB in the PID_FM_455.MOD_ADDR parameter.
   Use the start address that you specified when configuring your hardware as the module address of the FM 455.
5. Enter the number of the controller channel (1 to 16) in the PID_FM_455.CHANNEL parameter.
6. If you use the PID_PAR_455 instruction, enter the number of the controller channel in the PID_PAR_455.CHANNEL parameter.
7. Check to ensure that the "Sensor type analog input" of this channel is selected.
   If the analog input is not processed, all PID parameters are set to zero and a manipulated variable is not calculated.

Note

Multi-instance DBs

If you use multi-instance DBs in your project, you must commission the FM via a watch table.

See also

Description PID_FM_455 (Page 7266)

Description PID_PAR_455 (Page 7292)

Commissioning the controller

Introduction

You tune the PID parameters as follows:

- PID/PI controllers (step controller, continuous controller, pulse controller): Manually
- Fuzzy temperature controller: Automatically
- Step controller: Automatically with instruction TUN_ES
- Continuous controller: Automatically with instruction TUN_EC
- Pulse controller: Automatically with instruction TUN_EC

Procedure

Proceed as follows to tune a controller:

1. Put the CPU in RUN mode.
2. If you are using a step controller, measure the motor actuating time.
3. Tune the PID parameters with one of the options listed above.
4. Monitor the control loop with the trend view.

See also

Description TUN_EC (Page 6861)

Description TUN_EC (Page 6842)

Measuring the motor transition time (Page 9940)

Tuning PID parameters manually

Requirement

You have selected the "PID controller" control algorithm during parameter assignment.

Procedure

To set a PID/PI controller manually, proceed as follows:

1. Select the instance DB of the channel you want to tune.
2. Click the "Start" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value, and manipulated variable are recorded.
3. Enter values for the proportional action, integral action, derivative action and delay.
4. Click the icon "Send PID parameters to FM" in the "Tuning" group.
5. Select the "Change setpoint" check box in the "Current values" group.
6. Enter a new setpoint.
7. Clear the check box "Manual mode", if necessary, and click the icon in the "Current values" group.
   The controller works with the new parameters and controls the new setpoint.
8. Check the quality of the PID parameters based on the trend characteristics.
9. Repeat steps 3 to 8 until you are satisfied with the controller results.

Fuzzy system identification

Requirements on process with fuzzy temperature controller

Requirement

The process should fulfill the following requirements in order to achieve optimal control using the fuzzy temperature controller:

- At bath heatings the liquid to be heated has to be mixed thoroughly.
- In case of bath-in-bath control systems both liquids have to be mixed thoroughly. At the same time good heat transitions between all the heat-transferring media have to be ensured. In the case of materials with poor heat-transferring properties large transfer surfaces should ensure good heat transportation.

- In case of room temperature control systems thorough mixing (for example with fans) has to be ensured.
- The controlling system gain may not exceed the factor of 3.
- The delay time may not exceed 3% of the recovery time.
- The temperature to be controlled should change by a maximum of 1‰ of the specified maximum temperature at the maximum manipulated value output within the sampling time of the controller.

Classification of the Controlled Systems

A controlled system or a process to be controlled is characterized by parameters such as the heat output, the heating mass or the heating capacity of the medium to be heated. With regard to the fuzzy controller a difference is made between "critical" and "non-critical" temperature controlled systems as follows: The control system becomes increasingly critical:

- The greater the heat output,
- the greater the heating capacity of the heating,
- The lower the heating capacity of the medium to be heated,
- the greater the heat transition resistance,
- the smaller the heat transfer surface.

After a manipulated value step change has been applied to the controlled system, it reacts with a step response. The controlled system can also be classified on the basis of this step response: The control system becomes increasingly critical the greater the ratio t_u / t_g is and the greater the controlled system gain is. With t_u / t_g < 1/10 you have a non-critical control section.

Process of fuzzy system identification

Process of parameter tuning with fuzzy temperature controller

Parameter optimization is based on a self-optimizing fuzzy controller.

The identification of the controlled system begins with a monitoring phase during which no heat output takes place. The duration of the monitoring phase is as follows:

- Monitoring phase continuous controller: Approx. 1 min
- Monitoring phase step controller: Approx. 1 min + 1/2 x actuating time of final control element

This time is used in order to determine temperature trends in the heating medium. Afterwards the maximum heating output of 100% is output. This is visible at the LMN output parameter in the instance DB of the PID_FM instruction.

The range of the first 4% of the temperature increase of the setpoint value range is used for the identification, whereby information about the process response is obtained at 1% and 4% respectively of the temperature increase on the basis of the time that has passed.

The identification is complete when the heating output becomes less than 100%. This behavior can be used, for example, to deactivate the identification via the user program. After the identification phase the controller continues to operate with the determined parameters.

If the setpoint value step change is not adjusted and if the heating output remains permanently at zero, the identification has been terminated unsuccessfully, meaning that the controller cannot control the connected controlled system.

Controller behavior with different control sections

Problems do not arise in case of an "uncritical" controlled system, neither during identification nor during controlling.

The identification of a controlled system that is "too critical" is cancelled. Controlling of an identified "critical" controlled system is carried out very "carefully" or slowly.

Performing fuzzy system identification

Requirement

You have selected the "Fuzzy controller" control algorithm during parameter assignment.

You have to ensure as far as possible that the controlled system has settled (no heating-up or cooling-down process) or is changing slowly and monotonously before the identification is carried out. The criterion can be that the temperature change is to approach a straight during a period of one minute. At faster processes this requirement is particularly relevant.

Since the manipulated variable zero is output for approx. one minute by the controller at the beginning of identification, the temperature to be controlled has to lie near the ambient temperature.

Starting fuzzy system identification

Follow these steps to tune a fuzzy controller:

1. Select the instance DB of the channel you want to tune.
2. Click the "Start" icon.
   If there is no online connection, this will be established. The current values for the setpoint, process value, and manipulated variable are recorded.
3. Enter a new setpoint (SP_OP) in the "Tuning" group.
   The new setpoint must meet the following conditions:
   – Setpoint step change > 5%
   – New setpoint > process value + setpoint high limit x 0.12
4. Click ▶ "Fuzzy process identification" in the "Tuning" group.

Result

The new setpoint (SP_OP), SP_OP_ON = TRUE and FUZID_ON = TRUE are sent to the FM. Starting the fuzzy system identification. The IDSTATUS parameter of the CH_DIAG instruction provides information on the identification state. You see in the status display field if a fuzzy system identification is activated or running.

Deactivating fuzzy system identification

As long as the bit FUZID_ON = TRUE, the next identification is started whenever the setpoint step change is sufficiently large.

Proceed as follows to turn off the tuning mode:

1. Enter FUZID_ON in a watch table.
2. Set FUZID_ON = FALSE.

Note

Deactivation of the tuning mode FUZID_ON = FALSE before identification has been completed does not stop the identification. An identification process that has been started continues to run under all circumstances – with the exception of a negative setpoint value step change.

Repeating Fuzzy system identification

A renewed identification is carried out with each additional setpoint step change ≥12% unless the identification has been deactivated again with FUZID_ON = FALSE.

It is also possible to restart the identification by reducing and then increasing the setpoint by 12> %.

Canceling fuzzy system identification

Identification can be canceled in the following cases:

- By the controller if a "critical" controlled system is identified. After canceling the controller is in the error state. This state is indicated by the fact that the manipulated variable is reset permanently by the controller. This is also not changed by deactivating the optimization. The error state is deleted by starting a new identification.
  Information about the state of the identification is available through the IDSTATUS parameter of the CH_DIAG FB. More information on this can be found at the end of this section under "Controller State Information".
- By the operator by generating a negative setpoint value step change, whereby the setpoint value must lie below that of Condition 2 (see above "How To Start identification").

Proceed as follows to cancel tuning:

1. Enter a new setpoint that is less than the current process value and meets the same conditions as for the start of fuzzy system identification.
2. Click "Fuzzy process identification" in the "Tuning" group.

See also

Parameter IDSTATUS (Page 7290)

Downloading tuned parameters in device

Introduction

You can proceed as follows to prevent the parameters on the FM from being overwritten with the system data the next time you start up the CPU:

- You download the tuned PID parameters from the instance DB of the PID_FM to the FM after each startup of the FM with LOAD_PAR=TRUE.
- Download the PID parameters to the SDB.

Note

The PI, PID and SAVE parameter records cannot be stored in the SDB.

Procedure

Proceed as follows to download the parameters to the SDB:

1. Click on the icon "Transfer online data to an offline project".
   The PID parameters are saved in the parameter setting of the FM.
2. Stop CPU.
3. Click "Download to device".

Working with the trend view

Monitoring

1. Double click in the device view under the "Commissioning" module.
2. Navigate in the project view to "Trend view".
3. Select the instance DB you wish to monitor. The channel to which the instance DB has been assigned will be displayed.
4. Specify the update cycle in which the trend view is to record.
5. Click on "Measurement on" to start measuring.
6. Click on "Measurement off" to stop measuring.

Procedure

To adjust the scale, proceed as follows:

1. Double click on "Commissioning" in the device view of the module.
2. Navigate to "Trend view" in the configuration view.

3. To change the time axis:
   - Click on the arrow-button next to the date to open a calendar. Select a date from the calendar. To lock the selection, click on the blue lock. This possibility exists for the beginning and end dates.
   - Click on the time scale. The mouse pointer will be illustrated as a left/right arrow. Press the mouse and drag it to the left or right.
     If the beginning or end time is locked, the time scaling will change.
     If neither the beginning nor the end time is locked, the specified period will change, the scaling remains the same.
4. To change the value axis:
   - Click on the input field for the value displayed above or below. Enter your desired value here. To lock the selection, click on the blue lock. This possibility exists for the high and low value.
   - Click on the value scale. The mouse pointer will be illustrated as an up/down arrow. Press the mouse and drag it up or down.
     If the high or low value is locked, the scaling of the value axis will change.
     If neither the high nor the low value is locked, the specified range will change, the scaling remains the same.

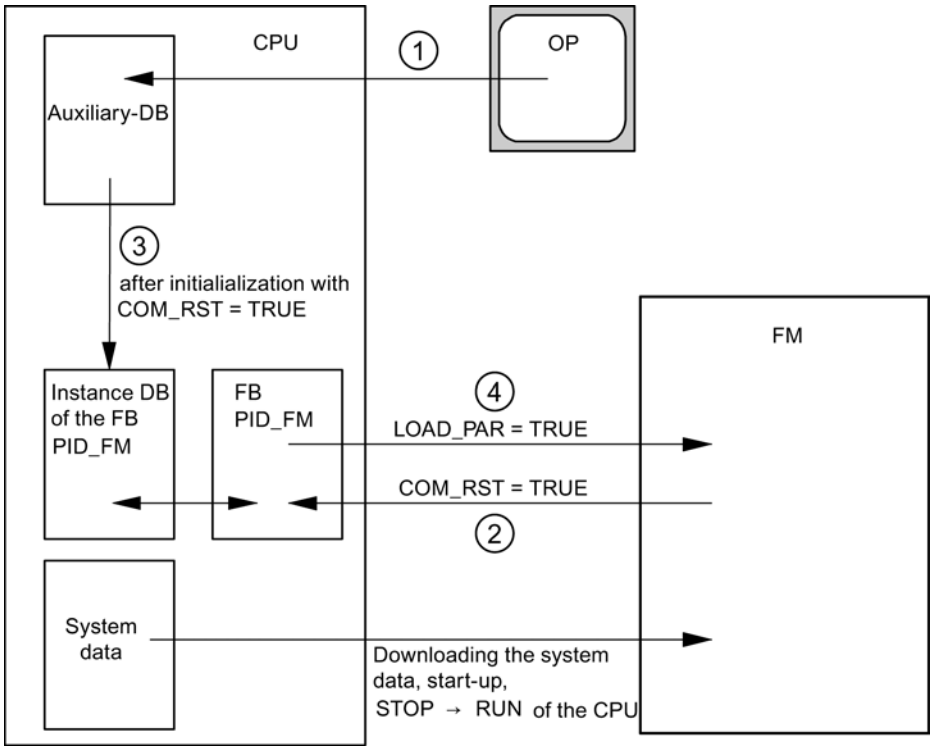
Changing parameters with the HMI

Procedure

Proceed as follows to change the controller parameters of the instruction PID_FM with the HMI:

1. Write the parameter you want to change from the HMI to an auxiliary DB on the CPU (see ①).
2. Set PID_FM.COM_RST to TRUE.
   The instruction performs an initialization run. The parameters are synchronized between FM and CPU (see ②).
3. Transfer the parameters you want to changed from the auxiliary DB to the instance DB of the instruction PID_FM (see ③).
4. Set PID_FM.LOAD_PAR to TRUE.
   The changed parameters are transferred to the FM (see ④).

It is necessary to store the parameters in an auxiliary DB because PID_FM after the start of the CPU with COM_RST = TRUE reads those parameters from the module that the CPU has previously transferred from the system data to the FM.



See also

Mode of operation PID_FM_455 (Page 7267)

Measuring the motor transition time

Introduction

The motor transition time has to be as accurate as possible for good controller results. The data in the actuator documentation contains average values for this type of actuator. The value for the specific actuator used may differ.

You can measure the motor transition time during commissioning if you are using actuators with position feedback or endstop signals. The motor transition time cannot be measured if neither position feedback nor endstop signals are available.

Requirement

- The configured motor transition time must be half the size of the real transition time. The actuator is moved for a maximum of twice the transition time in one direction.
- The position feedback must be interconnected for actuators with analog position feedback.
- The endstop signals must be interconnected for actuators with endstop signals.

Actuators with analog position feedback

Proceed as follows to measure motor transition time with position feedback:

1. Select the instance DB for which you want to measure the motor transition time.
2. Select the "Use position feedback" check box.
3. Enter the position to which the actuator is to be moved in the "Target position" input field. The current position feedback (starting position) will be displayed. Click the "Start transition time measurement" icon.

Result

The actuator is moved from the starting position to the target position. Time measurement starts immediately and ends when the actuator reaches the target position. The motor transition time is calculated according to the following equation:

Motor transition time = (output value high limit – output value low limit) × Measuring time / AMOUNT (target position – starting position).

The progress and status of transition time measurement are displayed. The measured transition time is displayed in the field "Measured transition time". Once transition time measurement is complete, the manual mode of the transition signals is switched on (LMNSOPON = TRUE). The actuator will stop being moved.

Actuators with endstop signals

Proceed as follows to measure motor transition time with endstop signals:

1. Select the instance DB for which you want to measure the motor transition time.
2. Select the "Use actuator endstop signals" check box.
3. Click the "Start transition time measurement" icon.

Result

The actuator is moved first to the low endstop, then to the high endstop and then back to the low endstop. Time measurement will start once the actuator has reached the lower endstop and will end when the actuator reaches the lower endstop for the second time. The motor transition time is equal to the time measured divided by two.

The progress and status of transition time measurement are displayed. The measured transition time is displayed in the field "Measured transition time". Once transition time measurement is complete, the manual mode of the transition signals is switched on (LMNSOPON = TRUE). The actuator will stop being moved.

Loading measured transition time

1. Click "Transfer online data to an offline project" to save the measured motor transition time locally in the project.
   The measured transition time is saved in the parameter setting of the FM.
2. Click "Download to device" to activate all changes the next time you start the CPU.

Cancelling transition time measurement

When you cancel the transition time measurement, the manual mode of the actuator signals is switched on (LMNSOPON = TRUE) and the manual mode with manipulated value specification is switched off (LMNOP_ON = FALSE). The actuator will stop being moved.

See also

In/out parameters PID_FM_455 (Page 7272)

FM 455: Diagnostics with LED display

Error Display via the Error LEDs

Where are the errors displayed?

The red LED INTF indicates an internal fault on the module.

The red LED EXTF indicates an external fault, for example fault on the connections.

If the yellow LED flashes, then the firmware has been deleted. This status can only occur in the case of faulty hardware or if the loading procedure of the firmware is aborted.

Which errors are displayed?

The following faults are indicated by the error LEDs lighting up:

Type of error	Diagnostic message	Possible cause	Correction
Internal errors	Module defective	Hardware error	Replace module
	Time monitoring (watchdog) tripped	Hardware error	Replace module
	EEPROM content is invalid	Failure of the supply voltage when configuring	Reconfigure module
External errors	Incorrect parameters in module	Incorrect parameters have been transferred to the module	Reconfigure module
	Errors with the analog inputs or analog outputs	Analog input hardware error	Replace module
		Analog input, wire breakage	Remedy wire break
		Analog input measuring range violation (underrange)	Check the measuring signal
		Analog input measuring range violation (overrange)	Check the measuring signal
		Analog output wire break	Remedy wire break
		Analog output short circuit	Eliminate the short-circuit
	Missing external auxiliary supply	Left-hand front connector not plugged 24 V supply missing	Plug in the left-hand front connector Restore 24 V supply

Diagnostic interrupt in the case of errors

All errors can trigger a diagnostic interrupt if you have enabled the diagnostic interrupt in the respective parameterization screen. From the diagnostic data records DS0 and DS1 you can see which errors have caused the LED to light up. The assignment of the diagnostic data records DS0 and DS1 is described in chapter "Triggering diagnostic interrupts (Page 9943)".

Triggering diagnostic interrupts

What is a diagnostic interrupt?

If a user program is to respond to an internal or external fault, you can set a diagnostics interrupt parameter that interrupts the cyclic program of the CPU and calls the diagnostics interrupt OB, OB8 82.

Which events may trigger a diagnostic interrupt?

The list shows which events can trigger a diagnostic interrupt:

- The module has not been configured or is incorrectly configured
- Module defective
- Wire break in analog inputs (4 to 20 mA)
- High limit violated and low limit violated with analog inputs
- Load break and short circuit at analog outputs
- Wire break at thermocouples and Pt 100

Default setting

The diagnostic interrupt is blocked by default.

Enabling the diagnostic interrupt

You disable or enable the diagnostics interrupt for the module in the “Basic parameters” dialog box.

Responses to an interrupt-triggering event

The following happens when an event occurs that could trigger a diagnostic interrupt:

- The diagnostic information is stored in diagnostic data records DS0 and DS1.
- One or both error LEDs light up.
- The diagnostic interrupt OB (OB 82) is called.
- The diagnostic record DS0 is entered in the start information of the diagnostic interrupt OB.
- If there is no hardware fault, the module continues to control.

If no OB 82 is programmed, the CPU goes to STOP.

Diagnostic data record DS0 and DS1

Introduction

The information showing the event which has triggered a diagnostics interrupt is written to the diagnostics data records DS0 and DS1. The diagnostics data record DS0 comprises four bytes. DS1 comprises 27 bytes with the first four bytes being identical to DS0.

Read data record from the module

The diagnostic data record DS0 is automatically transferred to the start information when the diagnostic OB is called. There these four bytes are stored in the local data (byte 8 - 11) of the OB 82.

The diagnostic data record DS1 (and hence also the content of the DS0) can be read from the module using the "CH_DIAG_455" instruction. This only makes sense if an error is reported in a channel in the DS0.

How does the diagnostics text appear in the diagnostics buffer?

If you want to enter the diagnostics message in the diagnostics buffer, call the "WR_REC" instruction "Enter user-specific message in the diagnostics buffer" in the user program. The event number of the respective diagnostics report is specified on the input parameter EVENTN. The interrupt is entered in the diagnostics buffer with x = 1 as incoming and x = 0 as outgoing. Alongside the time of the entry, the diagnostics buffer also shows the respective diagnostics text that is specified in the "Meaning" column.

Assignment of the diagnostic data record DS0 in the start information

The following table shows the assignment of the diagnostic data record DS0 in the start information. Bits that are not listed are of no significance and are zero.

Table 11-11 Assignment of the diagnostic data record DS0

Byte	Bit	Meaning	Note	Event no.
0	0	BG fault	Is set for every diagnostic event	8:x:00
	1	Internal error	If set with all internal errors: • Time watchdog tripped • EEPROM content invalid; module continues without controlling and waits for a new configuration by the CPU • EPROM fault • ADC/DAC fault • Analog input hardware error	8:x:01

Byte	Bit	Meaning	Note	Event no.
	2	External error	Is set with all external errors: • Missing external auxiliary supply • Faulty parameters • Analog input wire break (only area 4 to 20 mA) • Analog input measuring range violation (underrange) • Analog input measuring range violation (overrange) • Analog output wire break • Analog output short circuit	8:x:02
	3	Error in a channel	Refer to DS 1, as of byte 7 for a further breakdown	8:x:03
	4	Missing external auxiliary supply	24 V supply of the FM 455 failed	8:x:04
	6	EEPROM content is invalid	Missing supply voltage during a write operation in EEPROM. The module starts with default parameters.	8:x:03
	7	Faulty parameters	The module cannot make use of a parameter. Reason: Parameter unknown or invalid combination of parameters. See menu **Target system > Parameterization error display**	8:x:07
1	0 ... 3	Module class	Is always occupied with 8	–
	4	Channel-specific diagnosis	Is set when the module can provide additional channel information and a channel error exists (see DS 1 byte 7 to 12)	–
2	0	User module incorrect / missing	The position (A, B, C, D) of the coding key does not agree with the parameter assignment in the FM 455	8:x:30
	3	Time monitoring (watchdog) tripped	Hardware error	8:x:33
3	2	EPROM fault	Module defective	8:x:42
	4	ADC/DAC fault	Module defective	8:x:44

Diagnostics Data Record DS 1 of the FM 455

The length of diagnostic data record DS1 is 27 bytes. The first 4 bytes are identical with the diagnostic data record DS0. The following table illustrates the assignment of the remaining bytes. Bits that are not listed are of no significance and are zero.

Table 11-12 Assignment of the Bytes 4 to 12 of the Diagnostics Data Record DS1 of the FM 455

Byte	Bit	Meaning	Note	Event no.
4	0 ... 7	Channel type	Is always occupied with 75H	–
5	0 ... 7	Length of diagnostics information	Is always occupied with 8	–

Byte	Bit	Meaning	Note		Event no.
6	0 ... 7	Number of channels	17 always assigned (16 controllers + 1 reference channel)		–
7	0 … 7	Channel error vector	One bit each is assigned to channels 1 to 8		-
8	0 … 7	Channel error vector	One bit each is assigned to channels 9 to 16		-
9	0	Error on the reference channel			-
10	0	Analog input hardware error	Channel-specific diagnostics channel 1	–	8:x:B0
	1	Parameterization error		The position (A, B, C, D) of the coding key does not agree with the parameter assignment in the FM 455	8:x:B1
	2	Analog input wire break (only area 4 to 20 mA)			8:x:B2
	3	Unused			8:x:B3
	4	Analog input measurement range low limit violated[1]			8:x:B4
	5	Analog input measurement range high limit violated[1]			8:x:B5
	6	Analog output wire break		Only for current output of continuous controller	8:x:B6
	7	Analog output short circuit		Only with the current output of the C controller	8:x:B7
11	0 ... 7	See Byte 10	Channel-specific diagnostics - channel 2		See above
12	0 ... 7	See Byte 10	Channel-specific diagnostics - channel 3		See above
13	0 ... 7	See Byte 10	Channel-specific diagnostics - channel 4		See above
14	0 ... 7	See Byte 10	Channel-specific diagnostics channel 5		See above
15	0 ... 7	See Byte 10	Channel-specific diagnostics channel 6		See above
16	0 ... 7	See Byte 10	Channel-specific diagnostics channel 7		See above
17	0 ... 7	See Byte 10	Channel-specific diagnostics channel 8		See above
18	0 ... 7	See Byte 10	Channel-specific diagnostics channel 9		See above
19	0 ... 7	See Byte 10	Channel-specific diagnostics channel 10		See above
20	0 ... 7	See Byte 10	Channel-specific diagnostics channel 11		See above
21	0 ... 7	See Byte 10	Channel-specific diagnostics channel 12		See above
22	0 ... 7	See Byte 10	Channel-specific diagnostics channel 13		See above
23	0 ... 7	See Byte 10	Channel-specific diagnostics channel 14		See above
24	0 ... 7	See Byte 10	Channel-specific diagnostics channel 15		See above
25	0 ... 7	See Byte 10	Channel-specific diagnostics channel 16		See above

Byte	Bit	Meaning	Note	Event no.
26	0 ... 7	See Byte 10	Diagnosis for reference channel	See above
[1] refer to the chapter "Measuring transducer fault (Page 9947)"				

What you should particularly note

The FM 455 is supplied with voltage solely via the left front connector.

In the following cases, the CPU therefore recognizes the "Module removed / cannot respond":

- if the left front connector of the FM 455 is not connected
- if there is no 24 V supply voltage on the left front connector

Note

If, in the diagnostics buffer of the CPU, the entry "Module removed / cannot respond" is entered, check to see if the left front connector is connected and if the 24 V supply voltage of the FM 455 exists.

Additional information

For more information on the "RET_VALU" parameter, refer to List of RET_VALU messages (Page 9948) and the "Extended instructions" section of the help system.

Measuring transducer fault

Measuring transducer fault

The following faults in measuring transducers can be recognized by the FM 455:

- Measuring range violation (underrange)
- Measuring range violation (overrange)
- Wire break (not with all measuring ranges)

Whenever one of these faults occurs, the group error ”External error” is set in the diagnostics data record DS0 and channel-specific error bits are set in the diagnostics data record DS1 (refer to the tables in the previous section). When these errors disappear, the respective bits are reset.

The following table shows at which limits in the individual measuring ranges the error bits are set or reset.

Measuring range	Error bit Low Measuring Range Limit violated at ...	Error bit Measuring Range high limit violated at ...	Error bit Wire Break Display
	DS1: Byte 10 to 26, bit 4	DS1: Byte 10 to 26, bit 5	DS1: Byte 10 to 26, bit 2
0 to 20 mA	< - 3.5 mA	> 23.5 mA	–
4 to 20 mA	Error bit = 1 at < 3.6 mA Error bit = 0 at < 3.8 mA	> 22.8 mA	Error bit = 1 at < 3.6 mA Error bit = 0 at < 3.8 mA
0 V to 10 V	<- 1.175 V	> 11.75 V	–
Pt 100 (-200 to 850 °C) (-328 to 1562 °F)	< 30.82 mV	> 650.46 mV	Yes, parameter can be set, Display: 650.46 mV
Pt 100 (-200 to 556 °C) (-328 to 1032 °F)	< 30.82 mV	> 499.06 mV	Yes, parameter can be set, Display: 499.06 mV
Pt 100 (-200 to 130 °C) (-328 to 264 °F)	< 30.82 mV	> 254.12 mV	Yes, parameter can be set, Display: 254.12 mV
Thermocouple element type B	< 0 mV	> 13.81 mV	Yes, parameter can be set, the last value will be stored.
Thermocouple element type J	< - 8.1 mV	> 69.54 mV	Yes, parameter can be set, the last value will be stored.
Thermocouple element type K	< - 6.45 mV	> 54.88 mV	Yes, parameter can be set, the last value will be stored.
Thermocouple element type R	< - 0.23 mV	> 21.11 mV	Yes, parameter can be set, the last value will be stored.
Thermocouple element type S	< - 0.24 mV	> 18.7 mV	Yes, parameter can be set, the last value will be stored.
Free thermocouple element	< low input value of the polyline	> high input value of the polyline	Yes, parameter can be set, the last value will be stored.

List of RET_VALU messages

RET_VALU messages

JOB_ERR (Hex)	JOB_ERR (Dec)	JOB_ERR (Int)	Meaning
7000	28672	-32624	First call with REQ=0: no data transmission active; BUSY has the value 0.
7001	28673	-32624	First call with REQ=1: data transmission initiated; BUSY has the value 1.
7002	28674	-32624	Interim call (REQ irrelevant). Data transfer already active; BUSY has the value 1.
8090	32912	-32624	Specified logical base address invalid: There is no assignment in the SDB1/SDB2x, or it is not a base address.

JOB_ERR (Hex)	JOB_ERR (Dec)	JOB_ERR (Int)	Meaning
80A0	32928	-32608	Negative acknowledgment when reading from the module. Module was removed during the read operation or the module is defective.
80A1	32929	-32607	Negative acknowledgment when writing to the module. Module was removed during the write operation or the module is defective.
80A2	32930	-32606	Protocol error at layer 2
80A3	32931	-32605	Protocol error involving user interface/user
80A4	32932	-32604	Communication bus error
80B1	32945	-32591	Incorrect length specification. FM_TYPE parameter in channel DB not set correctly for the module in use.
80B2	32946	-32590	The configured slot is not being used.
80B3	32947	-32589	Actual module type is not match configured module type.
80C0	32960	-32576	Module data not ready for reading.
80C1	32961	-32575	Data of a write job of the same type have not yet been processed by the module.
80C2	32962	-32574	The module is currently processing the maximum possible number of jobs.
80C3	32963	-32573	Required resources (memory etc.) currently occupied.
80C4	32964	-32572	Communication error
80C5	32965	-32571	Distributed I/O not available.
80C6	32966	-32570	Priority class abort (restart or background).
8522	34082	-31454	Channel DB or parameter DB too short. The data cannot be read off the DB. (Write job)
8532	34098	-31438	DB number of the parameter DB too high (Write job)
853A	34106	-31430	Parameter DB not present. (write job)
8544	34116	-31420	Error at n-th (n > 1) read access to a DB after an error has occurred. (write job)
8723	34595	-30941	Channel DB or parameter DB too short. The data cannot be written to the DB. (Read job)
8730	34608	-30928	Parameter DB in the CPU is write protected. The data cannot be written to the DB (read job)
8732	34610	-30926	DB number of the parameter DB too high (Read job)
873A	34618	-30918	Parameter DB does not exist. (read job)
8745	34629	-30907	Error at n-th (n > 1) write access to a DB after an error has occurred. (Read job)
80ff	33023	-32513	Incorrect index specification with block FMT_PAR
Errors 80A2..80A4 and 80Cx are temporary, i.e., they can be rectified automatically after a waiting time. Messages of the 7xxx form indicate temporary operating states of communication.			

Example projects for FM 455

11.3.5.8 Using ET 200S 2PULSE

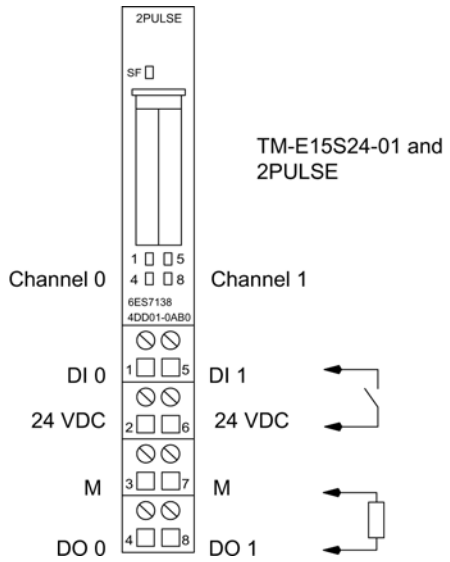
2PULSE terminal assignment

Wiring Rules

The cables (terminals 1 and 2 and terminals 5 and 6) must be shielded. The shield must be supported at both ends. To do this, use the shield contact (see the appendix of the ET 200S Distributed I/O System (http://support.automation.siemens.com/WW/view/en/1144348) operating instructions).

Terminal assignment of the 2PULSE

The following table shows the terminal assignment for 2PULSE.

View	Terminal Assignment	Meaning
		Channel 0: Terminal 1 to 4 Channel 1: Terminal 5 to 8 24V DC: Encoder supply M: Chassis ground DI: Input Signal DO: Output signal (max. 2 A per channel)

Operating modes and functions

Overview

Principle

The 2PULSE has two channels. You can select a separate mode for each channel. The operating mode is specified during the configuration. The mode that has been assigned parameters can then no longer be changed with your control program.

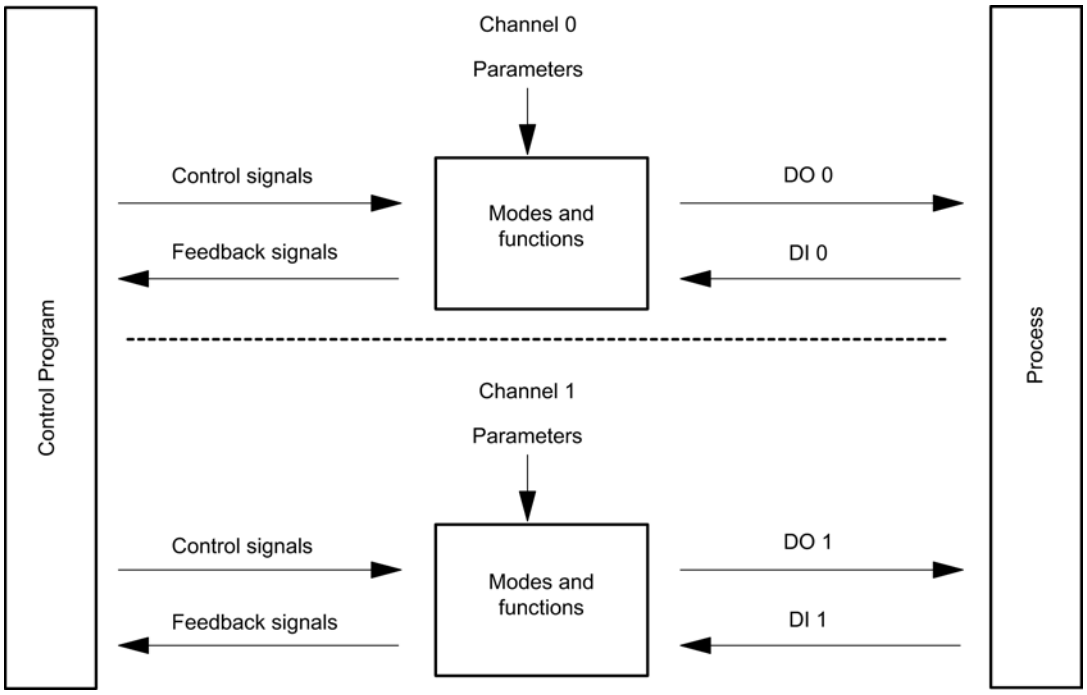
You can select various modes for each channel:

- Pulse output (Page 9952)
- Pulse width modulation (Page 9958)
- Pulse train (Page 9968)
- On/Off delay (Page 9974)
- Frequency output (Page 9982) (as of 6ES7138-4DD01-0AB0)

In addition to the set mode, the 2PULSE also has the following functions:

- Current measurement (Page 9988) in the "Pulse width modulation" and "Pulse train" modes (as of 6ES7138-4DD01-0AB0)
- Direct control of the DO digital output (Page 9989) by means of your control program; can be controlled separately for each channel.
- Error detection/diagnostics (Page 9990); the 2PULSE recognizes the errors for each channel separately.
- Parallel connection of both channels (Page 9992) to achieve a higher output current (as of 6ES7138-4DD01-0AB0).
- Reaction to CPU/Master STOP (Page 9994); the 2PULSE recognizes the CPU/Master STOP for both channels and responds in accordance with your parameter assignment.

The following figure shows the operating principle of the 2PULSE.



Interfaces to the Control Program and Process

To run the modes and execute functions, the 2PULSE has as an interface to the process of a digital input and a digital output for each channel (DI 0, DO 0 for Channel 0 and DI 1, DO1 for Channel 1).

You can modify and monitor the modes and functions with your control program using control signals (Page 9999) and feedback signals (Page 10002).

Parameters are assigned to the various modes.

You will find the following in the descriptions on operating modes and functions:

- The relevant parameters
- The control and feedback signals

The description of the modes and functions applies to both channels and the channels are therefore not referred to separately in the description.

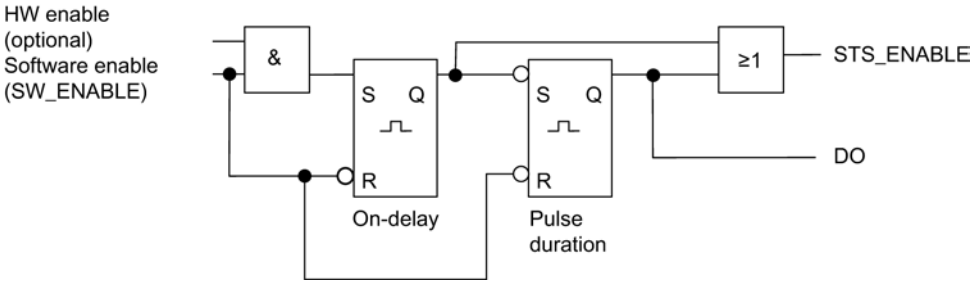
"Pulse output" operating mode

"Pulse output" operating mode

Pulse output

For the pulse duration you set, the 2PULSE outputs a pulse at the DO digital output (output sequence) on expiration of the set on-delay (Page 9997).

The following figure shows the block diagram for the operating mode "Pulse output".



Starting the output sequence

You must always issue the enable for the output sequence by means of the software enable (Page 9999) (SW_ENABLE 0 → 1; MANUAL_DO = 0) in your control program.

The feedback bit (Page 10002)ACK_SW_ENABLE indicates the software enable pending at the 2PULSE.

You can also set the DI digital input of the 2PULSE as a hardware enable (HW enable) with the "DI function (Page 9997)" parameter.

If you want to work with the software enable and hardware enable at the same time, when the software enable has been issued, the output sequence starts at the first positive edge of the hardware enable. Further positive edges of the hardware enable during the current output sequence are ignored by the 2PULSE. When the software enable has been issued, a positive edge of the hardware enable is enough to start the next output sequence.

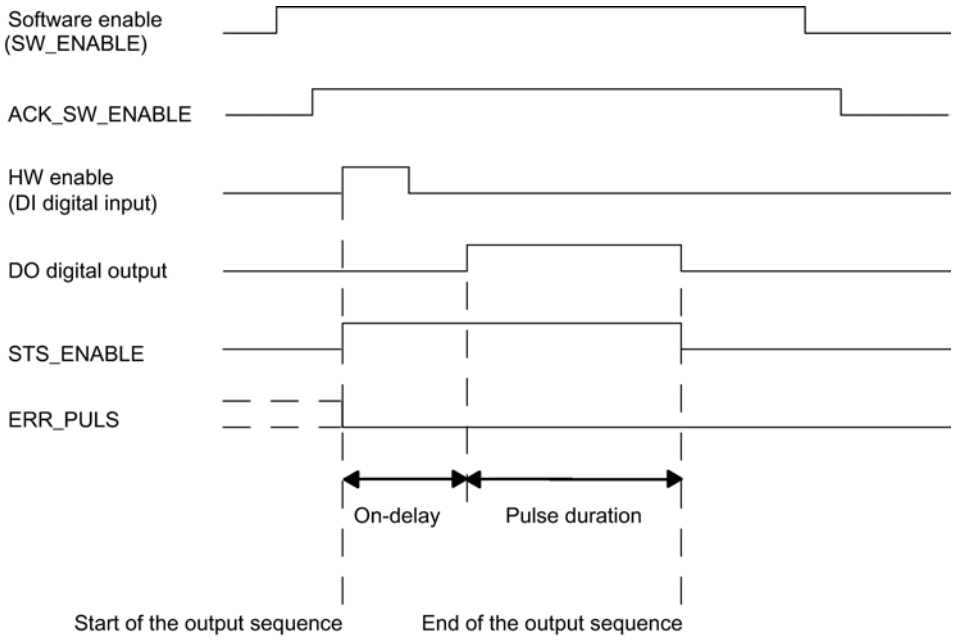
When the enable is issued (positive edge), the on-delay is started and the STS_ENABLE set. On expiration of the on-delay, the pulse is output with the set pulse duration. The output sequence (Page 9987) finishes with the end of the pulse; STS_ENABLE is cleared.

If you make a prohibited change to the pulse duration during runtime, the signal (Page 10002)ERR_PULS indicates a pulse output error. You will then have to restart the output sequence.

The next time the output sequence starts, the 2PULSE deletes the ERR_PULS feedback bit.

Pulse Diagram

The following figure shows an output sequence for the operating mode "Pulse output".



Canceling the output sequence

Deleting the software enable (SW_ENABLE = 0) during the on-delay or the pulse output cancels the output sequence, and STS_ENABLE and the DO digital output are canceled.

You will then have to restart the output sequence.

Truth table

The table below shows how the 2PULSE responds to certain signals.

Software enable SW_ENABLE	Hardware enable (digital input DI)	DO Digital Output	STS_ENABLE	Output Sequence
1	0 → 1	0, if on-delay > 0 1, if on-delay = 0	0 → 1	Start
0 → 1	Not used	0, if on-delay > 0 1, if on-delay = 0	0 → 1	Starting
0	Any status	0	0	Terminate
1	Any status	Previous status remains		-
0 → 1	0	0	0	-

Setting Times Using a Time Base

By means of the time base (Page 9997) that can be assigned parameters, you can select the resolution and range of the pulse duration and the on-delay.

Time base = 0.1 ms:	You can set times from 0.2 ms to 6.5535 s with a resolution of 0.1 ms.
Time base = 1 ms:	You can set times from 1 ms to 65.535 s with a resolution of 1 ms.

Setting and Changing the Pulse Duration

Set the pulse duration (Page 9998) directly in your control program as a numerical value between 0 and 65535.

Pulse duration = Time basis x Specified numerical value

If you change the pulse duration when an output sequence is running, the time already output will be subtracted from the new pulse duration and the pulse will continue to be output.

Reducing the Pulse Duration

If you have reduced the pulse duration to a time that is shorter than the time already output, the output sequence is terminated, STS_ENABLE and the DO digital output are deleted, and the ERR_PULS status bit is set. At the next output sequence, the ERR_PULS status bit is cleared.

Set the on-delay and change when using the short control interface

You specify the on-delay (Page 9997) as a value between 0 and 65535 in the parameters.

Configured on-delay = Time base x specified numerical value

Using the factor for the on-delay, you can adjust the time that has been assigned parameters in your control program. Set the factor between 0 and 255, with a weighting of 0.1:

On-delay = Factor x 0.1 x configured on-delay

If you change the on-delay factor during the output sequence, the new on-delay is activated at the next output sequence.

Isochronous mode (as of 6ES7138-4DD01-0AB0)

In the "Pulse output" operating mode, isochronous mode does not have any influence on the functionality.

Parameters of the "Pulse output" operating mode

Parameters of the "Pulse output" operating mode

The table below shows the parameters of the operating mode "Pulse output".

Parameter	Meaning	Value range	Default
Operating mode (Page 9996)	Set the "Pulse output" operating mode.	• Pulse output • Pulse width modulation • Pulse train • On/Off-delay • Frequency output	Pulse width modulation
Time base (Page 9997)	Using the time base, select the resolution and range of the pulse duration and the on-delay.	• 0.1 ms • 1 ms	0.1 ms
Function DI (Page 9997)	You can use the DI digital input as an input or as a hardware enable.	• Input • HW enable	Input
On-delay (Page 9997) [1]	The time from the start of the output sequence to the output of the pulse. You can change the on-delay in your control program using the "On-delay factor".	0 to 65535	0
[1] Only if the short control interface (Page 9999) is used If the long control interface is used, enter this value directly in the control interface.			

Control and feedback signals of the "Pulse output" operating mode

Control and feedback signals of the "Pulse output" operating mode

The table below shows the control and feedback signals of the operating mode "Pulse output".

Control and feedback signals	Meaning	Value range	Channel 0 address	Channel 1 address
Control signals when the short control interface is used (8 bytes)				
Software enable SW_ENABLE	Starting and canceling of the output sequence.	0 = SW_ENABLE canceled 1 = SW_ENABLE set 0→1 = start of output sequence; may be dependent on the hardware enable	Byte 2: Bit 0	Byte 6: Bit 0
Pulse duration	The time that is set for the DO digital output on expiration of the on-delay.	With time base 0.1 ms: 2 to 65535 With time base 1 ms: 1 to 65535 If you violate the low limit of the range, the 2PULSE will not output a pulse.	Word 0	Word 4

Control and feedback signals	Meaning	Value range	Channel 0 address	Channel 1 address
On-delay factor	The on-delay that has been assigned parameters can be changed before the start of the output sequence: On-delay = Factor x 0.1 x configured on-delay	0 to 255 If the on-delay <0.2 ms or if factor = 0, the effective on-delay is = 0. If there is an on-delay > 65.535 s, the on-delay is limited to 65.535 s.	Byte 3	Byte 7
Control signals when the long control interface is used (12 bytes) (as of 6ES7138-4DD01-0AB0)				
Software enable (SW_ENABLE)	Starting and canceling of the output sequence.	0 = SW_ENABLE canceled 1 = SW_ENABLE set 0→1 = start of output sequence; may be dependent on the hardware enable	Byte 4: Bit 0	Byte 10: Bit 0
Pulse duration	The time that is set for the DO digital output on expiration of the on-delay.	With time base 0.1 ms: 2 to 65535 With time base 1 ms: 1 to 65535 If you violate the low limit of the range, the 2PULSE will not output a pulse.	Word 0	Word 6
On-delay	The on-delay can be changed before the start of the output sequence.	0 to 65535 On-delay = Time basis x Specified numerical value	Word 2	Word 8
Feedback signals				
STS_ENABLE	Indicates an output sequence is running.	0 = Pulse output blocked 1 = Pulse output running	Byte 0: Bit 0	Byte 4: Bit 0
STS_DO	Indicates the signal level at the digital output DO. Note the update rate.	0 = Signal 0 on digital output DO 1 = Signal 1 on digital output DO	Byte 0: Bit 1	Byte 4: Bit 1
STS_DI	Indicates the signal level at digital input DI.	0 = Signal 0 on digital input DI 1 = Signal 1 on digital input DI	Byte 0: Bit 2	Byte 4: Bit 2
ACK_SW_ENABLE	Indicates the status of SW_ENABLE.	0 = SW_ENABLE cleared. 1 = SW_ENABLE set	Byte 0: Bit 3	Byte 4: Bit 3
ERR_PULS	Indicates a pulse output error.	0 = no pulse output error 1 = pulse output error	Byte 0: Bit 4	Byte 4: Bit 4
SEQ_CNT (as of 6ES7138-4DD01-0AB0)	Is incremented after completion of an output sequence.	Without HW enable: 0 … 1 With HW enable: 0 … 15	Byte 1: Bit 0 to 3	Byte 5: Bit 0 … 3
Reserved	–	0	Word 2	Word 6

Input and output signals of "Pulse output" operating mode

Input and output signals of "Pulse output" operating mode

The table below shows the input and output signals of the operating mode "Pulse output".

Input and output signal	Meaning	Value range	Channel 0 terminal	Channel 1 terminal
Input signal				
HW enable	You can select the HW enable with the "Function DI" parameter. The signal of the digital input DI is then interpreted by the 2PULSE at the start of the output sequence.	0 = HW enable cleared 1 = HW enable issued 0 → 1 = Start of the output sequence; dependent on the software enable (SW_ENABLE)	1	5
Output signal				
Pulse at the DO digital output	A pulse is output at the DO digital output for the set pulse duration.	0 = no pulse 1 = pulse	4	8

"Pulse width modulation" operating mode

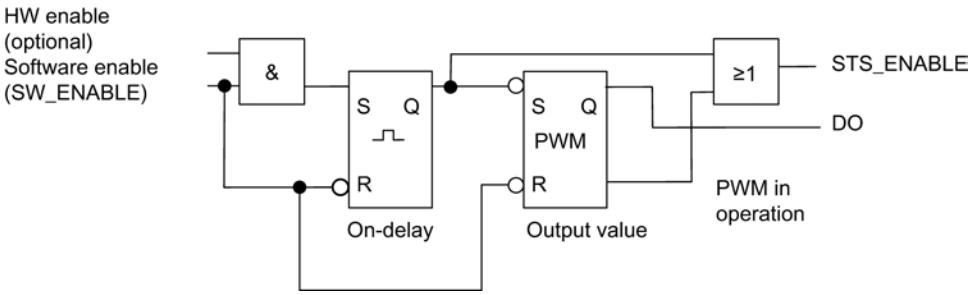
"Pulse width modulation" operating mode

Pulse width modulation

You specify an output value for the 2PULSE. The 2PULSE generates continuous pulses on this basis. The output value determines the pulse/interpulse ratio within a period (pulse-width modulation). The period can be adjusted.

The pulse train is output on expiration of the assigned on-delay (Page 9997) on the DO digital output of the 2PULSE (output sequence).

The following figure shows the block diagram for the operating mode "Pulse-width modulation".



Starting the output sequence

You must always issue the enable for the output sequence by means of the software enable (Page 9999) (SW_ENABLE 0 → 1; MANUAL_DO = 0) in your control program.

The feedback bit (Page 10002)ACK_SW_ENABLE indicates the software enable pending at the 2PULSE.

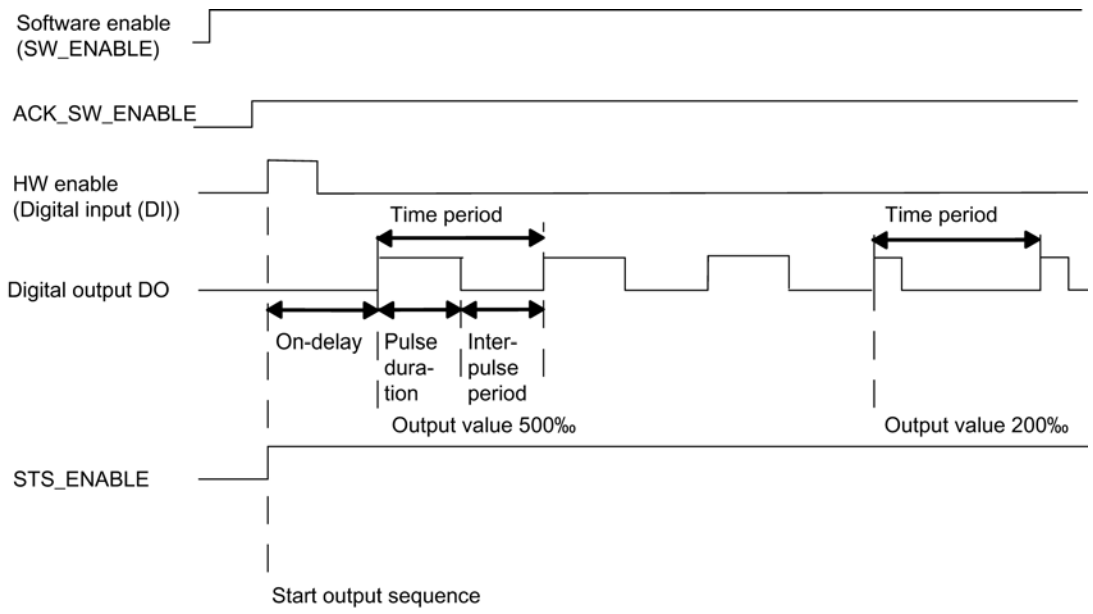
You can also set the digital input DI of the 2PULSE as a HW enable with the "Function DI (Page 9997)" parameter. In isochronous mode (as of 6ES7138-4DD01-0AB0), configuration of a hardware enable has no effect.

If you want to work with the software enable and hardware enable at the same time, when the software enable has been issued, the output sequence starts with the first positive edge of the hardware enable. Further positive edges of the hardware enable during the current output sequence are ignored by the 2PULSE.

When the enable is issued (positive edge), the on-delay is started and the STS_ENABLE set. The pulse train is output on expiration of the on-delay. The output sequence runs continuously as long as SW_ENABLE is set.

Pulse Diagram

The following figure shows an output sequence for the operating mode "Pulse-width modulation".



Canceling the output sequence

Deleting the software enable (SW_ENABLE = 0) during the on-delay or the pulse output cancels the output sequence, and STS_ENABLE and the DO digital output are canceled.

You will then have to restart the output sequence.

Truth table

The table below shows how the 2PULSE responds to certain signals.

Software enable SW_ENABLE	Hardware enable (digital input DI)	Digital output DO	STS_ENABLE	Output sequence
1	0 → 1	0, if on-delay > 0 1, if on-delay = 0	0 → 1	Starting
0 → 1	Not used	0, if on-delay > 0 1, if on-delay = 0	0 → 1	Starting
0	Any status	0	0	Canceling
1	Any status	Previous status remains		-
0 → 1	0	0	0	-

Minimum Pulse Duration and Minimum Interpulse Period

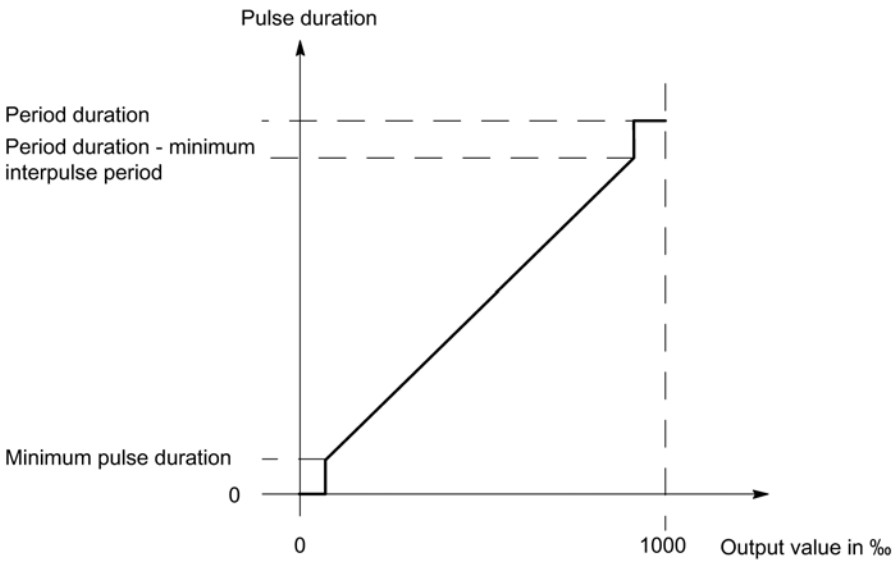
The minimum pulse duration and minimum interpulse period are superimposed on the proportional output characteristic.

You assign the minimum pulse duration and minimum interpulse period using the "Minimum pulse duration (Page 9998)" parameter; they always have the same value.

A pulse duration calculated by the 2PULSE that is shorter than the minimum pulse duration is suppressed.

A pulse duration calculated by the 2PULSE that is longer than the period minus the minimum interpulse period is set at 1000‰.

The following figure shows the modulation of the pulse duration.



Setting Times Using a Time Base

You use the assignable time base (Page 9997) to select the resolution and range of the period, the minimum pulse duration and the on-delay.

Time base = 0.1 ms:	You can set times from 0.2 ms to 6.5535 s with a resolution of 0.1 ms.
Time base = 1 ms:	You can set times from 1 ms to 65.535 s with a resolution of 1 ms.

Setting and Changing the Output Value

You select the range of the output value with the "Output format (Page 9997)" parameter.

- Output format "Per mill": Value range between 0 and 1000
  The 2PULSE uses this specified output value to calculate the pulse duration:
  Pulse duration = (output value/1000) x time period.
- "S7 analog output" output format: Value range between 0 and 27648
  The 2PULSE uses this specified output value to calculate the pulse duration:
  Pulse duration = (output value/27648) x time period.

You set the output value directly using your control program. The new output value is applied at the next rising edge of the output.

Setting the time period

Depending on the configuration, the time period is specified as a parameter value or in the control interface. The table below shows when a specific value is used.

For isochronous mode (as of 6ES7138-4DD01-0AB0), you have to specify the time period when parameters are assigned. In contrast to non-isochronous mode, you cannot change the time period later during runtime.

	Configuration with ...	
	Short control interface	Long control interface
Isochronous mode	Time period = "Time period" parameter value The value from the control interface is ignored.	
Non-isochronous mode	Time period = "Time period" parameter value × "Time period factor" from control interface	Time period = "Time period" setpoint from control interface The "Time period" parameter value is ignored.

Time period at configuration with short control interface

You specify the time period (Page 9998) in the parameters as a numerical value depending on the time base:

- With time base 1 ms: from 1 to 65535
- With time base 0.1 ms: from 2 to 65535

Configured time period = Time base x specified numerical value

Using the factor for the period duration, you can adjust the time that has been assigned parameters in your control program. Set the factor between 0 and 255, with a weighting of 0.1:

Time period = Factor x 0.1 x configured time period

The new setpoints for the time period are applied with the next rising edge of the output.

Note

If the time period changes in runtime, a brief minor increase in the time period can result (approx. 10 µs).

Note

A change in the time period using the control interface has no effect in isochronous mode (as of 6ES7138-4DD01-0AB0).

Setting the Minimum Pulse Duration and Minimum Interpulse Period

You specify the minimum pulse duration and the minimum interpulse period as a numerical value between 0 and 65535 using the "Minimum pulse duration (Page 9998)" parameter:

Configured minimum pulse duration/minimum pause between pulses = Time base x specified numerical value

Setting the On-Delay

You specify the on-delay (Page 9997) as a value between 0 and 65535 in the parameters.

Configured on-delay = Time base x specified numerical value

When the long control interface is used, the factor for the on-delay can be used to adjust the configured time in your control program. Set the factor between 0 and 255, with a weighting of 0.1:

On-delay = Factor x 0.1 x configured on-delay

Current measurement (as of 6ES7138-4DD01-0AB0)

An important application for current measurement in the "Pulse width modulation" operating mode is the current regulation of a proportional valve. This allows you to compensate for thermal effects at the valve, for example.

Additional information is available under "Function: Current measurement (as of 6ES7138-4DD01-0AB0) (Page 9988)".

Isochronous mode in the operating mode "Pulse width modulation"

Isochronous mode

Note

Isochronous mode is available with 2PULSE as of order number 6ES7138-4DD01-0AB0.

The following particular aspects apply to isochronous mode in the "Pulse width modulation" operating mode:

- The output always begins without an on-delay, a configured on-delay is ignored.
- The output always begins after the software enable has been issued, a configured hardware enable is ignored.
- A change of the time period during runtime is not possible. Corresponding specifications via the control interface are ignored.

In isochronous mode, the output sequence is synchronized with the moment T_o. The time period is coordinated to the DP cycle. It is possible that your defined time period cannot be implemented exactly.

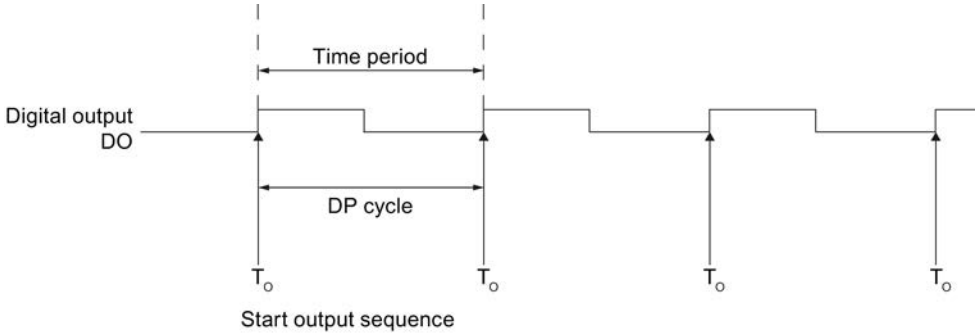
The configured value of the time period is adjusted by the 2PULSE to the DP cycle based on a calculation algorithm. The calculation is performed to minimize the difference between the specified and calculated time period. In the most unfavorable case, the deviation amounts to half the DP cycle. The table below shows examples.

DP cycle time T_{DP} [1]	Specified time period $T_{Setpoint}$	Calculated time period T_{Actual} [1]
10 ms	5 ms	5 ms
10 ms	2 ms	2 ms
10 ms	6 ms	5 ms
10 ms	16 ms	20 ms

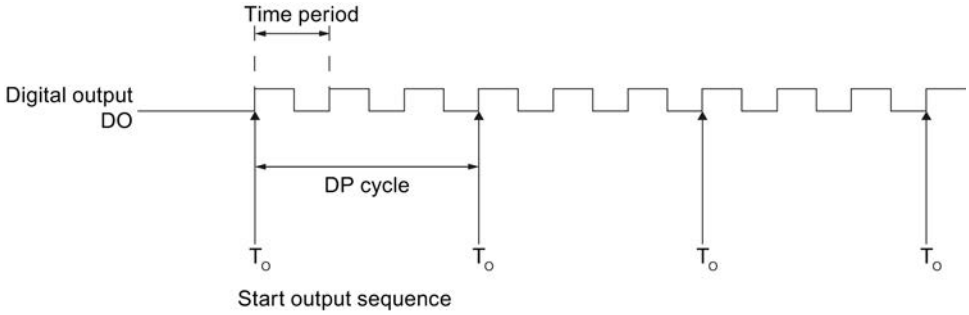
[1] The calculated time period and the DP cycle time always have an integer ratio.

The correlation between the and the DP cycle is represented graphically below. The pulse-pause ratio is shown in the examples with 50% each. For additional information, refer to the Isochronous Mode (http://support.automation.siemens.com/WW/view/en/15218045) function manual.

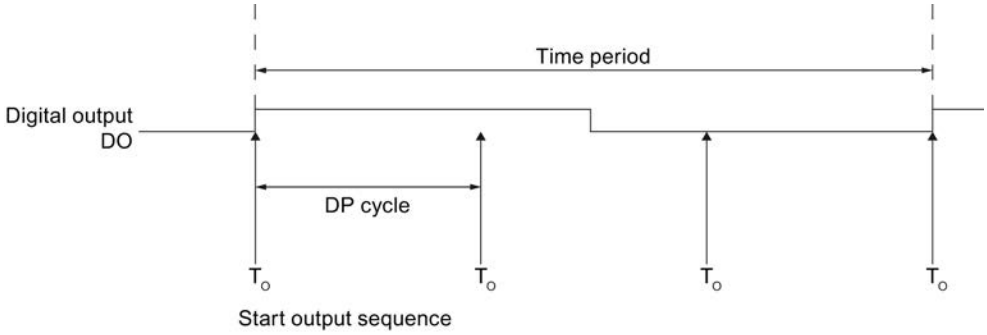
- The time period is equal to the DP cycle



- The time period is less than the DP cycle



- The time period is greater than the DP cycle



Parameters of the "Pulse width modulation" operating mode

Parameters of the "Pulse width modulation" operating mode

The table below shows the parameters of the operating mode "Pulse width modulation".

Parameter	Meaning	Value Range	Default
Operating mode (Page 9996)	Set the "Pulse width modulation" operating mode.	• Pulse output • Pulse-width modulation • Pulse train • On/Off-delay • Frequency output	Pulse width modulation
Time base (Page 9997)	Use the time base to select the resolution and the value range of the time period, the minimum pulse duration, and the on-delay.	• 0.1 ms • 1 ms	0.1 ms
Output format (Page 9997)	Choose between the Per mil, and the S7 analog output formats, depending on which output value resolution you need.	• Per mill • S7 analog output	Per mill
Function DI (Page 9997)	You can use the digital input DI as an input or as a HW enable.	• Input • HW enable	Input
On-delay (Page 9997)	The time from the start of the output sequence to the output of the pulse train.	0 to 65535	0
Minimum pulse duration (Page 9998)	Minimum pulse duration and minimum interpulse period: Enter the response time of the actuator connected on your DO digital output.	0 to 65535	0
Time period (Page 9998) [1]	Setpoint of the time period When specifying the time period take into account the set minimum pulse duration and the response time of your actuator that is connected to the digital output DO. When the short control interface is used, you can use the "Time period factor" to change the time period in your control program in non-isochronous mode.	With time base 0.1 ms: 2 to 65535 With time base 1 ms: 1 to 65535	20000 → 2 s

[1] Read the information provided above on "Setting the time period (Page 9958)" and on "Isochronous mode (Page 9963)".

Control and feedback signals of the "Pulse width modulation" operating mode

Control and feedback signals of the "Pulse width modulation" operating mode

The table below shows the control and feedback signals of the operating mode "Pulse width modulation".

Control and feedback signals	Meaning	Value range	Channel 0 address	Channel 1 address
Control signals when the short control interface is used (8 bytes)				
Software enable (SW_ENABLE)	Starting and canceling of the output sequence.	0 = SW_ENABLE canceled 1 = SW_ENABLE set 0→1 = start of output sequence; may be dependent on the hardware enable	Byte 2: Bit 0	Byte 6: Bit 0
Output value	The value that is output in pulse-width modulated format on the digital output DO.	Depending on the output format: • Per mill: 0 ... 1000 • S7 analog output: 0 to 27648 If you enter an output value > 1000 or 27648, the 2PULSE limits this to 1000 or 27648.	Word 0	Word 4
Period duration factor	You can change the assigned period: Time period = Factor x 0.1 x configured time period	Factor: 0 to 255 Period duration: 2 × Minimum pulse duration up to 65.635 s. If a period duration of < 2 x minimum pulse duration occurs or < 2 µs or if Factor = 0, the effective period = 2 x minimum pulse duration, but at least 200 µs. If there is a period > 65.535 s, it is limited to 65.535 s.	Byte 3	Byte 7
Control signals when the long control interface is used (12 bytes) (as of 6ES7138-4DD01-0AB0)				
Software enable (SW_ENABLE)	Starting and canceling of the output sequence.	0 = SW_ENABLE canceled 1 = SW_ENABLE set 0→1 = start of output sequence; may be dependent on the hardware enable	Byte 4: Bit 0	Byte 10: Bit 0
Output value	The value that is output in pulse-width modulated format on the digital output DO.	Depending on the output format: • Per mill: 0 ... 1000 • S7 analog output: 0 to 27648 If you enter an output value > 1000 or 27648, the 2PULSE limits this to 1000 or 27648.	Word 0	Word 6
Time period [1)]	Setpoint of the time period in non-isochronous mode	With time base 0.1 ms: 2 to 65535 With time base 1 ms: 1 to 65535	Word 2	Word 8

Control and feedback signals	Meaning	Value range	Channel 0 address	Channel 1 address
On-delay factor	The on-delay that has been assigned parameters can be changed before the start of the output sequence: On-delay = Factor x 0.1 x configured on-delay	0 to 255 If the on-delay <0.2 ms or if factor = 0, the effective on-delay is = 0. If there is an on-delay > 65.535 s, the on-delay is limited to 65.535 s.	Byte 5	Byte 11
Feedback signals				
STS_ENABLE	Indicates an output sequence is running.	0 = pulse output blocked 1 = pulse output running	Byte 0: Bit 0	Byte 4: Bit 0
STS_DO	Indicates the signal level at the digital output DO. Note the update rate.	0 = Signal 0 on digital output DO 1 = Signal 1 on digital output DO	Byte 0: Bit 1	Byte 4: Bit 1
STS_DI	Indicates the signal level at digital input DI.	0 = Signal 0 on digital input DI 1 = Signal 1 on digital input DI	Byte 0: Bit 2	Byte 4: Bit 2
ACK_SW_ENABLE	Indicates the status of SW_ENABLE.	0 = SW_ENABLE cleared. 1 = SW_ENABLE set	Byte 0: Bit 3	Byte 4: Bit 3
SEQ_CNT (as of 6ES7138-4DD01-0AB 0)	No function in this operating mode.	–	Byte 1: Bit 0 to 3	Byte 5: Bit 0 … 3
Current measurement (as of 6ES7138-4DD01-0AB 0)	Current at the DO as a SIMATIC S7 analog value.	0 to 27648 (overrange up to 32767)	Word 2	Word 6
[1] Read the information provided on "Setting the time period (Page 9958)".				

Input and output signals of the "Pulse width modulation" operating mode

Input and output signals of the "Pulse width modulation" operating mode

The table below shows the input and output signals of the operating mode "Pulse width modulation".

Input and Output Signals	Meaning	Value range	Channel 0 terminal	Channel 1 terminal
Input signal				
HW enable	You can select the HW enable with the "Function DI" parameter. The signal of the digital input DI is then interpreted by the 2PULSE at the start of the output sequence.	0 = HW enable cleared 1 = HW enable issued 0 → 1 = Start of the output sequence; dependent on the software enable	1	5
Output signal				
Pulse train on the digital output DO	The pulse train is output on the digital output DO.	0 = no pulse 1 = pulse	4	8

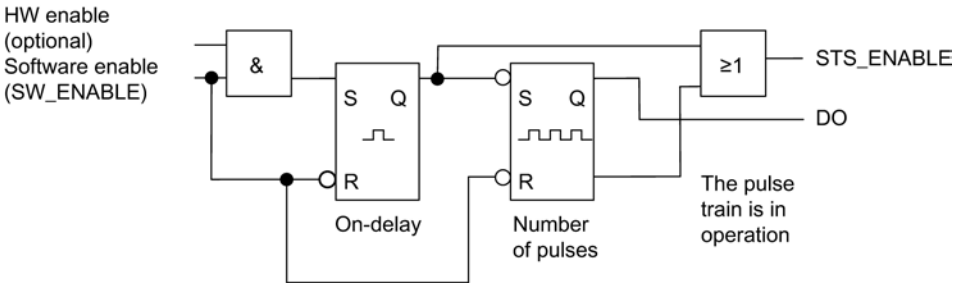
"Pulse train" operating mode

"Pulse train" operating mode

Definition

The 2PULSE outputs the number of pulses you specified as a pulse train at the DO digital output on expiration of the set on-delay (Page 9997) (output sequence). The period duration and pulse duration of the pulses can be adjusted.

The following figure shows the block diagram for the operating mode "Pulse train".



Starting the Output Sequence

You must always issue the enable for the output sequence by means of the software enable (Page 9999) (SW_ENABLE 0 → 1; MANUAL_DO = 0) in your control program. The feedback bit (Page 10002)ACK_SW_ENABLE indicates the software enable pending at the 2PULSE.

You can also set the digital input DI of the 2PULSE as a HW enable with the "Function DI (Page 9997)" parameter.

If you want to work with the software enable and hardware enable at the same time, when the software enable has been issued, the output sequence starts with the first positive edge of the hardware enable. Further positive edges of the hardware enable during the current output sequence are ignored by the 2PULSE. When the software enable has been issued, a positive edge of the hardware enable is enough to start the next output sequence.

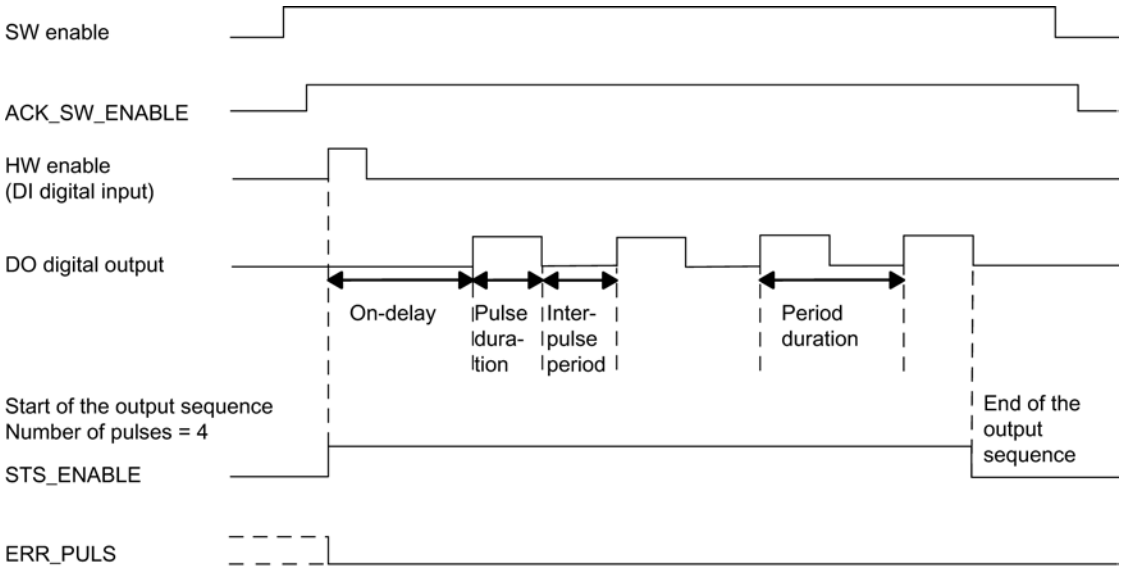
When the enable is issued (positive edge), the on-delay is started and the STS_ENABLE set. On expiration of the on-delay, the pulse train is output with the set number of pulses. The output sequence (Page 9987) finishes as soon as the last pulse has been output; STS_ENABLE is cleared.

If you make a prohibited change to the number of pulses during runtime, the signal (Page 10002)ERR_PULS indicates a pulse output error.

At the next output sequence, the 2PULSE deletes the ERR_PULS feedback bit.

Pulse diagram

The following figure shows an output sequence for the operating mode "Pulse train".



Canceling the output sequence

Deleting the software enable during the on-delay or the pulse train terminates the output sequence, and STS_ENABLE and the DO digital output are deleted.

You will then have to restart the output sequence.

Truth table

The table below shows how the 2PULSE responds to certain signals.

Software enable SW_ENABLE	Hardware enable (digital input DI)	Digital output DO	STS_ENABLE	Output sequence
1	0 → 1	0, if on-delay > 0 1, if on-delay = 0	0 → 1	Starting
0 → 1	Not used	0, if on-delay > 0 1, if on-delay = 0	0 → 1	Starting
0	Any status	0	0	Canceling
1	Any status	Previous status remains		-
0 → 1	0	0	0	-

Setting Times Using a Time Base

By means of the time base (Page 9997) that can be assigned parameters, you can select the resolution and range of the period, pulse duration and on-delay.

Time base = 0.1 ms:	You can set times from 0.2 ms to 6.5535 s with a resolution of 0.1 ms.
Time base = 1 ms:	You can set times from 1 ms to 65.535 s with a resolution of 1 ms.

Setting and Changing the Number of Pulses

Set the number of pulses (Page 9999) directly as a numerical value between 0 and 65535 with your control program.

If you change the number of pulses on expiration of the on-delay, the new value takes effect immediately:

- If you have increased the number of pulses, the new, higher number of pulses is output.
- If you have reduced the number of pulses, and if the lower number of pulses has already been output, the output sequence is terminated, STS_ENABLE and the DO digital output are deleted, and ERR_PULS is set. At the next output sequence, ERR_PULS is cleared.

Set the time period and change when using the short control interface

Set the period duration (Page 9998) as a value between 2 and 65535 in the parameters:

Configured time period = Time base x specified numerical value

Using the factor for the period, you can adjust the assigned time in your control program. Set the factor between 0 and 255, with a weighting of 0.1:

Time period = Factor x 0.1 x configured time period

If you change the factor during the output sequence, the new period duration will take effect at the start of the next output sequence.

Setting the Pulse Duration

Set the pulse duration (Page 9998) as a numerical value between 1 and 65535 with the "Pulse duration" parameter:

Configured pulse duration = Time base x Specified numerical value

When the long control interface is used, you can set the ratio of the pulse duration to the interpulse period:

Pulse-pause ratio = (Setpoint / 255) × time period

Setting the On-Delay

You specify the on-delay (Page 9997) as a value between 0 and 65535 in the parameters.

Configured on-delay = Time base x specified numerical value

Isochronous mode (as of 6ES7138-4DD01-0AB0)

In the "Pulse train" operating mode, isochronous mode does not have any influence on the functionality.

Current measurement (as of 6ES7138-4DD01-0AB0)

Additional information is available under "Function: Current measurement (as of 6ES7138-4DD01-0AB0) (Page 9988)".

Parameters of the "Pulse train" operating mode

Parameters of the "Pulse train" operating mode

The table below shows the parameters of the operating mode "Pulse train".

Parameter	Meaning	Value range	Default
Operating mode (Page 9996)	Set the "Pulse train" operating mode.	• Pulse output • Pulse width modulation • Pulse train • On/Off-delay • Frequency output	Pulse width modulation
Time base (Page 9997)	Using the time base, select the resolution and range of the period duration, pulse duration, and on-delay.	• 0.1 ms • 1 ms	0.1 ms
Function DI (Page 9997)	You can use the digital input DI as an input or as a HW enable.	• Input • HW enable	Input
On-delay (Page 9997)	The time from the start of the output sequence to the output of the pulse train.	0 to 65535	0
Pulse duration (Page 9998) [1)]	Pulse duration: Enter the response time of the actuator connected on your DO digital output.	With time base 0.1 ms: 2 to 65535 With time base 1 ms: 1 to 65535 If you violate the low limit of the range, the 2PULSE sets the pulse duration to 0.2 ms or 1 ms.	10000 → 1 s

Parameter	Meaning	Value range	Default
Time period (Page 9998) [1]	The period duration should always be a multiple of the response time of the actuator connected to the DO digital output. Define the period duration according to the required repetition rate of the pulses. You can change the time period in your control program using the "Time period factor".	With time base 0.1 ms: 2 to 65535 With time base 1 ms: 1 to 65535	20000 → 2 s
[1] Only if the short control interface (Page 9999) is used If the long control interface is used, enter this value directly in the control interface.			

Control and feedback signals of the "Pulse train" operating mode

Control and feedback signals of the "Pulse train" operating mode

The table below shows the control and feedback signals of the operating mode "Pulse train".

Control and feedback signals	Meaning	Value range	Channel 0 address	Channel 1 address
Control signals when the short control interface is used (8 bytes)				
Software enable (SW_ENABLE)	Starting and canceling of the output sequence.	0 = SW_ENABLE cleared. 1 = SW_ENABLE set 0 → 1 = Start of the output sequence; may be dependent on the HW enable	Byte 2: Bit 0	Byte 6: Bit 0
Number of pulses	Number of pulses that are output at the DO digital output on expiration of the on-delay.	0 to 65535 If the number of pulses is 0, the 2PULSE does not output any pulses. The output sequence is terminated with ERR_PULS = 1.	Word 0	Word 4
Period duration factor	The on-delay that can be assigned parameters can be changed before the start of the output sequence: Time period = Factor x 0.1 x configured time period	Factor: 0 to 255 Period duration: > Pulse duration up to 65.535 s If there is a period duration > 65.535 s, it is set to 65.535 s. If a period duration ≤ pulse duration, it is set to a pulse duration of + 0.2 ms (or pulse duration + 1 ms).	Byte 3	Byte 7
Control signals when the long control interface is used (12 bytes) (as of 6ES7138-4DD01-0AB0)				

Control and feedback signals	Meaning	Value range	Channel 0 address	Channel 1 address
Software enable (SW_ENABLE)	Starting and canceling of the output sequence.	0 = SW_ENABLE cleared. 1 = SW_ENABLE set 0 → 1 = Start of the output sequence; may be dependent on the HW enable	Byte 4: Bit 0	Byte 10: Bit 0
Number of pulses	Number of pulses that are output at the DO digital output on expiration of the on-delay.	0 to 65535 If the number of pulses is 0, the 2PULSE does not output any pulses. The output sequence is terminated with ERR_PULS = 1.	Word 0	Word 6
Period duration	Setpoint of the time period	With time base 0.1 ms: 2 to 65535 With time base 1 ms: 1 to 65535	Word 2	Word 8
Pulse-pause ratio	You can set the ratio of the pulse duration to the interpulse period	0 to 255 E.g the value of 128 corresponds to a ratio of 50:50.	Byte 5	Byte 11
Feedback Signals				
STS_ENABLE	Indicates an output sequence is running.	0 = pulse output blocked 1 = pulse output running	Byte 0: Bit 0	Byte 4: Bit 0
STS_DO	Indicates the signal level at the digital output DO. Note the update rate.	0 = Signal 0 on digital output DO 1 = Signal 1 on digital output DO	Byte 0: Bit 1	Byte 4: Bit 1
STS_DI	Indicates the signal level at digital input DI.	0 = Signal 0 on digital input DI 1 = Signal 1 on digital input DI	Byte 0: Bit 2	Byte 4: Bit 2
ACK_SW_ENABLE	Indicates the status of SW_ENABLE.	0 = SW_ENABLE cleared. 1 = SW_ENABLE set	Byte 0: Bit 3	Byte 4: Bit 3
ERR_PULS	Indicates a pulse output error.	0 = no pulse output error 1 = pulse output error	Byte 0: Bit 4	Byte 4: Bit 4
SEQ_CNT (as of 6ES7138-4DD01-0AB 0)	Is incremented after completion of an output sequence.	Without HW enable: 0 … 1 With HW enable: 0 … 15	Byte 1: Bit 0 … 3	Byte 5: Bit 0 … 3
Current measurement (as of 6ES7138-4DD01-0AB 0)	Current at the DO as a SIMATIC S7 analog value.	0 … 27648 (overrange up to 32767)	Word 2	Word 6

Input and output signals of "Pulse train" operating mode

Input and output signals of "Pulse train" operating mode

The table below shows the input and output signals of the operating mode "Pulse train".

Input and Output Signals	Meaning	Value range	Channel 0 terminal	Channel 1 terminal
Input signal				
HW enable	You can select the HW enable with the "Function DI" parameter. The signal of the digital input DI is then interpreted by the 2PULSE at the start.	0 = HW enable cleared 1 = HW enable issued 0 → 1 = Start of the output sequence; dependent on the software enable (SW_ENABLE)	1	5
Output Signal				
Pulse train at the DO digital output	The preset number of pulses is output at the DO digital output.	0 = no pulse 1 = pulse	4	8

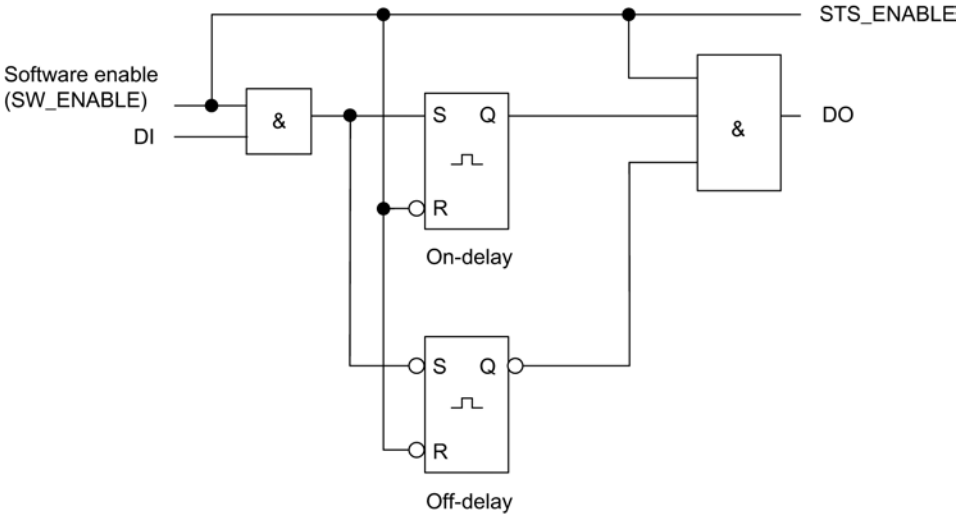
"On/Off-delay" operating mode

"On/Off-delay" operating mode

Definition

The signal pending at the digital input DI is output with an on/off-delay at the digital output DO by the 2PULSE.

The following figure shows the block diagram for the operating mode "On/Off-delay".



Output Sequence Enable

You must always issue the enable for the output sequence by means of the software enable (Page 9999) (SW_ENABLE 0 → 1; MANUAL_DO = 0) in your control program, STS_ENABLE is then set. The feedback bit (Page 10002)ACK_SW_ENABLE indicates the software enable pending at the 2PULSE.

The positive edge on the DI digital input (0 → 1) starts the on-delay, and the DO digital output (Page 9987) is set on expiration of the on-delay.

The negative edge on the DI digital input (1 → 0) starts the off-delay, and the DO digital output (Page 9987) is deleted on expiration of the off-delay.

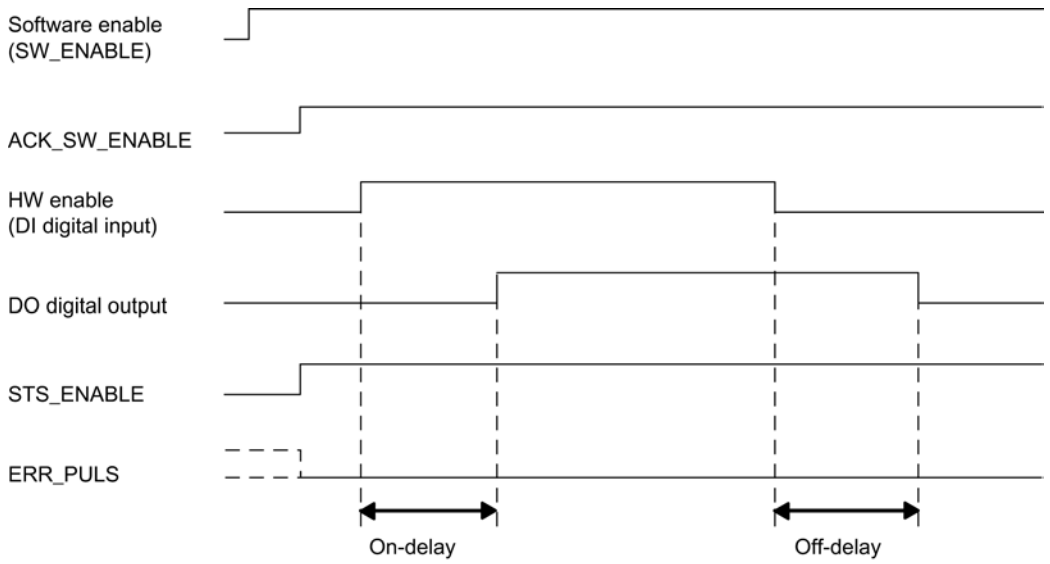
If the 2PULSE recognizes a pulse duration or interpulse period that is too short, this is displayed by the pulse output error (Page 10002)ERR_PULS.

At the next edge at the DI digital input, the 2PULSE deletes the ERR_PULS feedback bit.

Pulse diagram

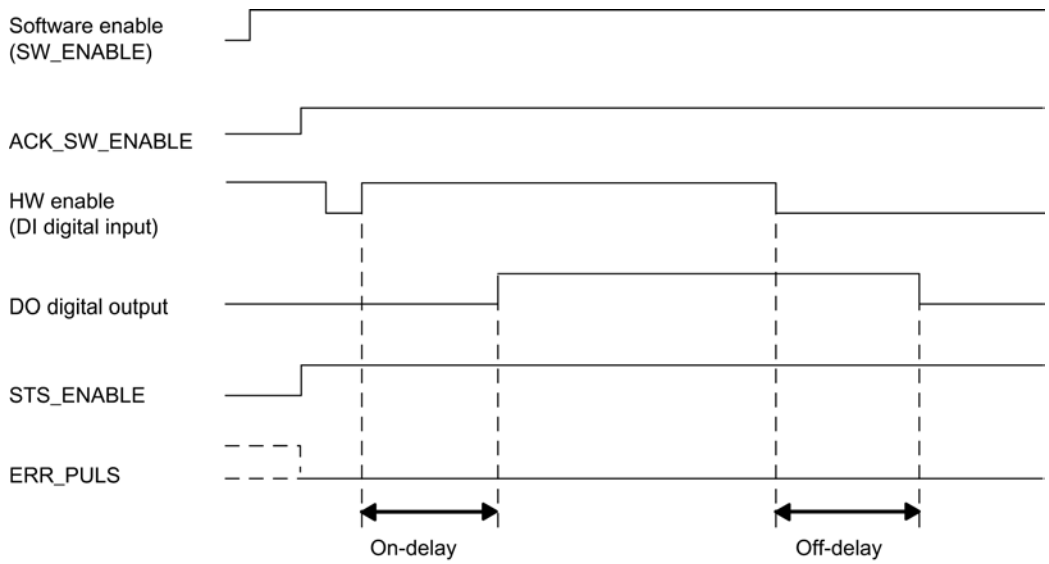
SW_ENABLE is set, while DI = 0:

The following figure shows an output sequence in the operating mode "On/Off-delay" if DI = 0 at the start.



SW_ENABLE is set, while DI = 1:

If SW_ENABLE is set while DI = 1, the first edge at DI (falling edge) is ignored. The following figure shows the corresponding output sequence.



Canceling the output sequence

Clearing the software enable (SW_ENABLE = 0) during the output sequence causes the output sequence to be aborted, along with STS_ENABLE and the digital output.

Truth table

The table below shows how the 2PULSE responds to certain signals.

Software enable SW_ENABLE	Digital input DI	Digital output DO	STS_ENABLE	Output sequence
1	0 → 1	0, if on-delay > 0 1, if on-delay = 0	1	Starting
1	1 → 0	1, if off-delay > 0 0, if off-delay = 0	1	Starting
0	Any status	0	0	Canceling
1	Any status	Previous status remains	1	-
0 → 1	0	0	1	-

Minimum Pulse Duration/Minimum Interpulse Period of the Digital Output DO

The minimum pulse duration/minimum interpulse period of the digital output DO amounts to 100 µs.

Make sure you take this into consideration when you set the on/off-delay and the pulse duration/interpulse period of the digital input DI; otherwise, the response at the digital output DO is not defined.

The Pulse Duration of the Digital Input DI Is Too Short

The 2PULSE detects a pulse that is too short on the negative edge on the digital input DI if:

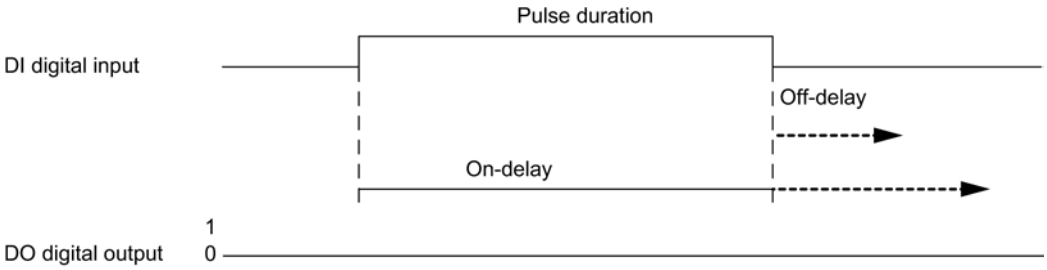
Pulse duration + Off-delay ≤ On-delay.

Response of the 2PULSE to a pulse duration that is too short:

- ERR_PULS is set
- The current on-delay is deleted.
- The Off-delay is not started.
- The signal level at the digital output DO remains at 0.

ERR_PULS is deleted at the next positive edge on the digital input DI.

The following figure shows the behavior if the pulse duration is too short.



The Interpulse Period of the Digital Input DI Is Too Short

The 2PULSE detects an interpulse period that is too short on the positive edge on the digital input DI if:

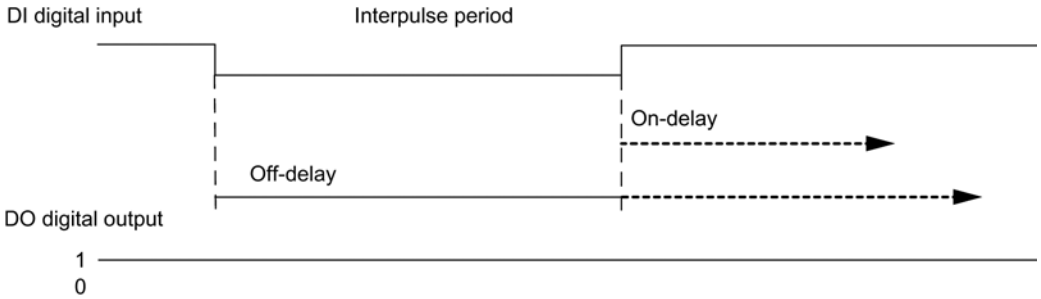
Interpulse period + on-delay ≤ Off-delay.

Response of the 2PULSE to a interpulse period that is too short:

- ERR_PULS is set
- The current Off-delay is deleted.
- The on-delay is not started.
- The signal level at the digital output DO remains at 1.

ERR_PULS is deleted with the next negative edge on the digital input DI.

The following figure shows the behavior if the interpulse period is too short.



Retriggering the current on-delay

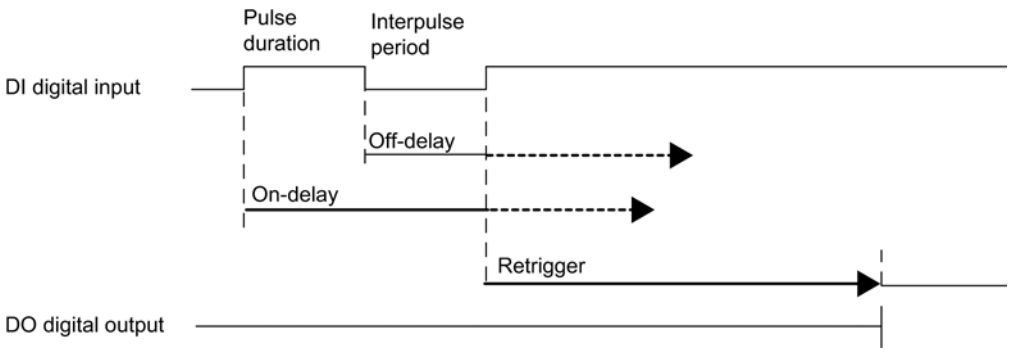
The 2PULSE starts a new on-delay on the positive edge on the digital input DI if:

On-delay > pulse duration + interpulse period

This deletes the current Off-delay.

The digital output DO is only set if signal level 1 is present on the digital input DI longer than the on-delay. This enables you to filter rapid pulse trains.

The following figure shows the retriggering of the current on-delay.



Retriggering the current Off-delay

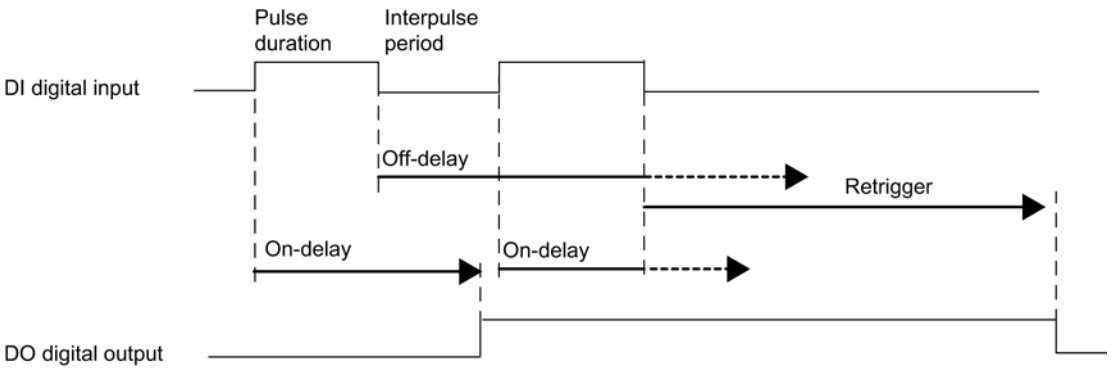
The 2PULSE starts a new off-delay on the negative edge on the digital input DI if:

Off-delay > pulse duration + interpulse period.

This deletes the current on-delay.

The digital output DO is only deleted if signal level 0 is present on the digital input DI longer than the off-delay.

The following figure shows the retriggering of the current off-delay.



Setting Times Using a Time Base

Use the assigned time base (Page 9997) to select the resolution and the value range of the on-delay and the off-delay.

Time base = 0.1 ms:	You can set times from 0.2 ms to 6.5535 s with a resolution of 0.1 ms.
Time base = 1 ms:	You can set times from 1 ms to 65.535 s with a resolution of 1 ms.

Set the on-delay and change when using the short control interface

You specify the on-delay (Page 9997) as a value between 0 and 65535 in the parameters.

Configured on-delay = Time base x specified numerical value

Using the factor for the on-delay, you can adjust the assigned time in your control program. Set the factor between 0 and 255, with a weighting of 0.1:

On-delay = Factor x 0.1 x configured on-delay

If you change the on-delay factor, the new on-delay is activated with the next positive edge on the digital input DI.

Setting and changing the off-delay

Set the off-delay directly as a numerical value between 0 and 65535 in your control program (Page 9999).

Off-delay = Time base x Specified numerical value

If you change the off-delay factor, the new Off-delay is activated with the next negative edge on the digital input DI.

Isochronous mode (as of 6ES7138-4DD01-0AB0)

In the "On/Off-delay" operating mode, isochronous mode does not have any influence on the functionality.

Parameters for the "On/Off-delay" operating mode

Parameters for the "On/Off-delay" operating mode

The table below shows the parameters of the operating mode "On/Off-delay".

Parameter	Meaning	Value range	Default
Operating mode (Page 9996)	Set the "On/Off-delay" operating mode.	• Pulse output • Pulse width modulation • Pulse train • On/Off-delay • Frequency output	Pulse width modulation
Time base (Page 9997)	Use the time base to select the resolution and the value range of the on-delay and Off-delay.	• 0.1 ms • 1 ms	0.1 ms
On-delay (Page 9997) [1]	The time between a positive edge of digital input DI and its output on the digital output DO. You can change the on-delay with your control program with "On-delay factor".	0 to 65535	0
[1] Only if the short control interface (Page 9999) is used If the long control interface is used, enter this value directly in the control interface.			

Control and feedback signals of the "On/Off-delay" operating mode

Control and feedback signals of the "On/Off-delay" operating mode

The table below shows the control and feedback signals of the operating mode "On/Off-delay".

Control and feedback signals	Meaning	Value range	Channel 0 address	Channel 1 address
Control signals when the short control interface is used (8 bytes)				
Software enable (SW_ENABLE)	You must always issue the software enable in your control program. If you cancel the software enable, the current output sequence will be canceled.	0 = SW_ENABLE cleared. 1 = SW_ENABLE set	Byte 2: Bit 0	Byte 6: Bit 0
Off-delay	The time between a negative edge of the digital input DI and its output on the digital output DO.	With time base 0.1 ms: 2 to 65535 With time base 1 ms: 1 to 65535 If you violate the low limit of the range, the off-delay will not function.	Word 0	Word 4

Control and feedback signals	Meaning	Value range	Channel 0 address	Channel 1 address
On-delay factor	You can change the assigned on-delay: On-delay = Factor x 0.1 x configured on-delay	Factor: 0 to 255 On-delay: 0.2 ms to 65.535 s If the on-delay is < 0.2 ms or if the factor = 0, the effective on-delay = 0. If the on-delay is > 65.535 s, the on-delay is limited to 65.535 s.	Byte 3	Byte 7
Control signals when the long control interface is used (12 bytes) (as of 6ES7138-4DD01-0AB0)				
Software enable (SW_ENABLE)	You must always issue the software enable in your control program. If you cancel the software enable, the current output sequence will be canceled.	0 = SW_ENABLE cleared. 1 = SW_ENABLE set	Byte 4: Bit 0	Byte 10: Bit 0
Off-delay	The time between a negative edge of the digital input DI and its output on the digital output DO.	With time base 0.1 ms: 2 to 65535 With time base 1 ms: 1 to 65535 If you violate the low limit of the range, the off-delay will not function.	Word 0	Word 6
On-delay	The time between a positive edge of the digital input DI and its output on the digital output DO.	With time base 0.1 ms: 2 to 65535 With time base 1 ms: 1 to 65535 If you violate the low limit of the range, the on-delay will not function.	Word 2	Word 8
Feedback Signals				
STS_ENABLE	Indicates the status of the software enable (SW_ENABLE).	0 = software enable blocked 1 = software enable issued	Byte 0: Bit 0	Byte 4: Bit 0
STS_DO	Indicates the signal level at the digital output DO. Note the update rate.	0 = Signal 0 on digital output DO 1 = Signal 1 on digital output DO	Byte 0: Bit 1	Byte 4: Bit 1
STS_DI	Indicates the signal level at digital input DI.	0 = Signal 0 on digital input DI 1 = Signal 1 on digital input DI	Byte 0: Bit 2	Byte 4: Bit 2
ACK_SW_ENABLE	Indicates the status of SW_ENABLE.	0 = SW_ENABLE cleared. 1 = SW_ENABLE set	Byte 0: Bit 3	Byte 4: Bit 3
ERR_PULS	Indicates a pulse output error if the pulse duration or interpulse period is too short.	0 = no pulse output error 1 = pulse output error	Byte 0: Bit 4	Byte 4: Bit 4
SEQ_CNT (as of 6ES7138-4DD01-0AB0)	Is incremented at DO with each edge (positive and negative).	0 … 15	Byte 1: Bit 0 … 3	Byte 5: Bit 0 … 3
Reserved	–	0	Word 2	Word 6

Input and output signals of the "On/Off-delay" operating mode

Input and output signals of the "On/Off-delay" operating mode

The table below shows the input and output signals of the operating mode "On/Off-delay".

Input and Output Signals	Meaning	Value range	Channel 0 terminal	Channel 1 terminal
Input signal				
Digital input DI	The signal of the digital input DI is output with an on/off-delay on digital output DO by the 2PULSE.	0 = no pulse 1 = pulse	1	5
Output Signal				
Pulse on the digital output DO	The signal of the digital input DI is output with an on/off-delay on digital output DO by the 2PULSE.	0 = no signal 1 = signal	4	8

"Frequency output" mode (as of 6ES7138-4DD01-0AB0)

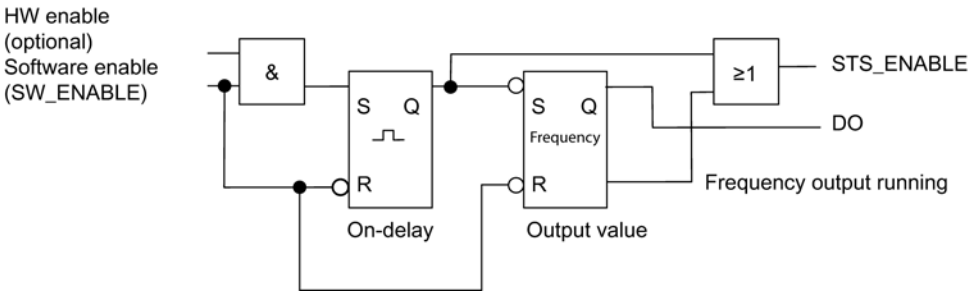
"Frequency output" operating mode

Definition

A square wave signal with a specified frequency is output at the digital output of the 2PULSE.

The output sequence is started on expiration of the configured on-delay (Page 9997) on the DO digital output of the 2PULSE.

The following figure shows the block diagram for the operating mode "Frequency output".



Starting the Output Sequence

You must always issue the enable for the output sequence by means of the software enable (Page 9999) (SW_ENABLE 0 → 1; MANUAL_DO = 0) in your control program.

The feedback bit (Page 10002)ACK_SW_ENABLE indicates the software enable pending at the 2PULSE.

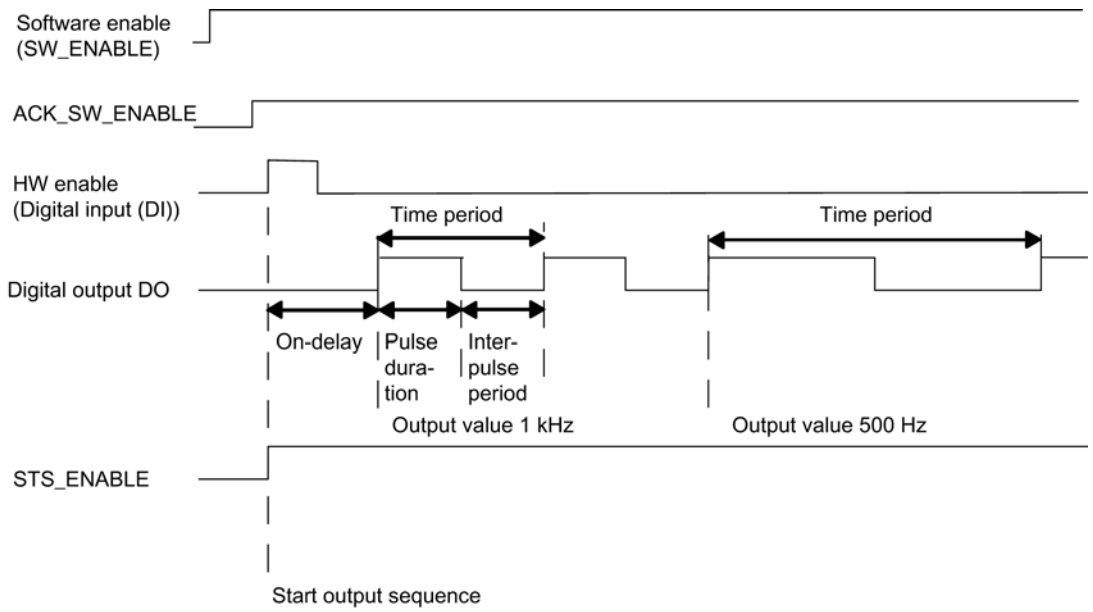
You can also set the digital input DI of the 2PULSE as a HW enable with the "Function DI (Page 9997)" parameter.

If you want to work with the software enable and hardware enable at the same time, when the software enable has been issued, the output sequence starts with the first positive edge of the hardware enable. Further positive edges of the hardware enable during the current output sequence are ignored by the 2PULSE.

When the enable is issued (positive edge), the on-delay is started and the STS_ENABLE set. The frequency output is started on expiration of the on-delay time. The output sequence runs continuously as long as SW_ENABLE is set.

Pulse diagram

The following figure shows an output sequence for the operating mode "Frequency output".



Canceling the output sequence

Deleting the software enable (SW_ENABLE = 0) during the on-delay or the frequency output cancels the output sequence, and STS_ENABLE and the DO digital output are canceled.

You will then have to restart the output sequence.

Truth Table

The table below shows how the 2PULSE responds to certain signals.

Software enable SW_ENABLE	Hardware enable (digital input DI)	Digital output DO	STS_ENABLE	Output sequence
1	0 → 1	0, if on-delay > 0 1, if on-delay = 0	0 → 1	Starting
0 → 1	Not used	0, if on-delay > 0 1, if on-delay = 0	0 → 1	Starting
0	Any status	0	0	Canceling
1	Any status	Previous status remains		-
0 → 1	0	0	0	-

Setting Times Using a Time Base

Use the configured time base (Page 9997) to select the resolution and the value range of the on-delay.

Time base = 0.1 ms:	You can set times from 0.2 ms to 6.5535 s with a resolution of 0.1 ms.
Time base = 1 ms:	You can set times from 1 ms to 65.535 s with a resolution of 1 ms.

Setting and Changing the Output Value

You select the range of the output value with the "Output format (Page 9997)" parameter.

- If your output value is between 1 Hz and 5000 Hz, select the output format "1 Hz".
- If your output value is a SIMATIC S7 analog value (between 0 and 27648), select the "S7 analog output" output format. With "S7 analog output" you attain a higher accuracy at the setpoint specification.
  The 2PULSE calculates the frequency according to the following equation:
  Frequency = (Setpoint / 27648) × 5 kHz

You set the output value directly using your control program. The new frequency value is applied at the next rising edge of the output.

Output frequency accuracy

The configured output frequency is output with the following accuracy at the digital output DO:

- 0.01 Hz at output frequencies of 0.1 Hz to 800 Hz
- 0.1 Hz at output frequencies > 800 Hz to 2000 Hz
- 0.5 Hz at output frequencies > 2000 Hz to 5000 Hz

Set the on-delay when using the short control interface

You specify the on-delay (Page 9997) as a value between 0 and 65535 in the parameters.

Configured on-delay = Time base x specified numerical value

When the short control interface is used, the factor for the on-delay can be used to adjust the configured time in your control program. Set the factor between 0 and 255, with a weighting of 0.1:

On-delay = Factor x 0.1 x configured on-delay

Isochronous mode

In the "Frequency output" operating mode, isochronous mode does not have any influence on the functionality.

Parameters of the "Frequency output" operating mode

Parameters of the "Frequency output" operating mode

The table below shows the parameters of the operating mode "Frequency output".

Parameter	Meaning	Value range	Default
Operating mode (Page 9996)	Set the "Frequency output" operating mode.	• Pulse output • Pulse width modulation • Pulse train • On/Off-delay • Frequency output	Pulse width modulation
Time base (Page 9997)	Use the time base to select the resolution and the value range of the on-delay.	• 0.1 ms • 1 ms	0.1 ms
Output format (Page 9997)	Choose between 1 Hz and the S7 analog output format, depending on which output frequency resolution you need.	• 1 Hz • S7 analog output	1 Hz
Function DI (Page 9997)	You can use the digital input DI as an input or as a HW enable.	• Input • HW enable	Input
On-delay (Page 9997) [1]	The time from the start of the output sequence to the output of the pulse train. You can change the on-delay in your control program using the "On-delay factor".	0 to 65535	0
[1] Only if the short control interface (Page 9999) is used If the long control interface is used, enter this value directly in the control interface.			

Control and feedback signals of the "Frequency output" operating mode

Control and feedback signals of the "Frequency output" operating mode

The table below shows the control and feedback signals of the operating mode "Frequency output".

Control and feedback signals	Meaning	Value range	Channel 0 address	Channel 1 address
Control signals when the short control interface is used (8 bytes)				
Software enable (SW_ENABLE)	Starting and canceling of the output sequence.	0 = SW_ENABLE canceled 1 = SW_ENABLE set 0→1 = start of output sequence; may be dependent on the hardware enable	Byte 2: Bit 0	Byte 6: Bit 0
Output value	The value that is output in frequency format on the digital output DO.	Depending on the output format: • 1 Hz: 0 ... 5000 • S7 analog output: 0 to 27648 If you enter an output value > 5000 or 27648, the 2PULSE limits this to 5000 or 27648.	Word 0	Word 4
On-delay factor	The on-delay that has been assigned parameters can be changed before the start of the output sequence: On-delay = Factor x 0.1 x configured on-delay	0 to 255 If the on-delay <0.2 ms or if factor = 0, the effective on-delay is = 0. If there is an on-delay > 65.535 s, the on-delay is limited to 65.535 s.	Byte 3	Byte 7
Control signals when the long control interface is used (12 bytes)				
Software enable (SW_ENABLE)	Starting and canceling of the output sequence.	0 = SW_ENABLE canceled 1 = SW_ENABLE set 0→1 = start of output sequence; may be dependent on the hardware enable	Byte 4: Bit 0	Byte 10: Bit 0
Output value	The value that is output in frequency format on the digital output DO.	Depending on the output format: • 1 Hz: 0 ... 5000 • S7 analog output: 0 to 27648 If you enter an output value > 5000 or 27648, the 2PULSE limits this to 5000 or 27648.	Word 0	Word 6
On-delay	The on-delay can be changed before the start of the output sequence.	0 to 65535 On-delay = Time basis x Specified numerical value	Word 2	Word 8
Feedback Signals				
STS_ENABLE	Indicates an output sequence is running.	0 = pulse output blocked 1 = pulse output running	Byte 0: Bit 0	Byte 4: Bit 0
STS_DO	Indicates the signal level at the digital output DO. Note the update rate.	0 = Signal 0 on digital output DO 1 = Signal 1 on digital output DO	Byte 0: Bit 1	Byte 4: Bit 1

Control and feedback signals	Meaning	Value range	Channel 0 address	Channel 1 address
STS_DI	Indicates the signal level at digital input DI.	0 = Signal 0 on digital input DI 1 = Signal 1 on digital input DI	Byte 0: Bit 2	Byte 4: Bit 2
ACK_SW_ENABLE	Indicates the status of SW_ENABLE.	0 = SW_ENABLE cleared. 1 = SW_ENABLE set	Byte 0: Bit 3	Byte 4: Bit 3
SEQ_CNT	No function in this operating mode.	–	Byte 1: Bit 0 … 3	Byte 5: Bit 0 … 3
Reserved	–	0	Word 2	Word 6

Input and output signals of the "Frequency output" operating mode

Input and output signals of the "Frequency output" operating mode

The table below shows the input and output signals of the operating mode "Frequency output".

Input and Output Signals	Meaning	Value range	Channel 0 terminal	Channel 1 terminal
Input signal				
HW enable	You can select the HW enable with the "Function DI" parameter. The signal of the digital input DI is then interpreted by the 2PULSE at the start of the output sequence.	0 = HW enable cleared 1 = HW enable issued 0 → 1 = Start of the output sequence; dependent on the software enable	1	5
Output Signal				
Frequency on the digital output DO	The frequency is output on the digital output DO.	0 = no pulse 1 = pulse	4	8

Sequence counter (as of 6ES7138-4DD01-0AB0)

The 2PULSE has a sequence counter for each channel with which the completed output sequences are counted.
You can monitor the completion of an output sequence with the sequence counter SEC_CNT in the feedback interface.

The counter has a width of 4 bits. After an high limit violated it jumps back to 0.

Function of the sequence counter

The sequence counter has the following function in the individual operating modes:

- Pulse output (Page 9968) and pulse train (Page 9952)
  - Without hardware enable the counter is set to 1 after completion of the output sequence.
  - With hardware enable, the counter is incremented after every completed output sequence.
- On/Off-delay (Page 9974)
  The counter is incremented at DO with each edge (positive and negative).
- Pulse width modulation and frequency output
  The counter does not have any function.

When the software enable (SW_ENABLE = 0) is created, the counter is reset to 0.

Application options

The sequence counter can be used in particular for:

- Detecting (counting) very short pulses
- Counting output sequences in connection with the hardware enable

Function: Current measurement (as of 6ES7138-4DD01-0AB0)

The measurement of current can be set in a variety of applications for control and diagnostics purposes. You can read the measured value of the current from the feedback interface.

Principle of operation

A measurement of current only takes place during the pulse output in the "Pulse width modulation (Page 9958)" and "Pulse train (Page 9968)" operating modes. Without a pulse output (at STS_ENABLE = 0), 0 is supplied as the measured value. The measured value is valid at STS_ENABLE = 1.

In the "Pulse output", On/Off-delay" and "Frequency output" operating modes, 0 is always measured as the measured value of the current.

The returned measured value of the current is a mean value of measured values that is recorded during the duration of a period. At period durations < 5 ms, the interval between two individual measuring points amounts to a maximum of 40 µs. When longer periods are involved, 128 measuring points per period are always recorded. This results in correspondingly longer intervals between the measuring points.

In isochronous mode, a new measured value of the current is always made available in the feedback interface at the moment T_i. In non-isochronous mode, a new measured value of the current is made available about every 500 µs.

Note

In order for the current measurement to work correctly, no freewheeling diode may be applied to the load.

Measuring range and measured value

The measuring range is set fixed to a rated current of 2 A.

The measured value of the current is supplied as a SIMATIC S7 analog value in the feedback interface (Page 10002).

- A current of 2 A corresponds to the SIMATIC S7 analog value of 27648.
- With a parallel connection, a current of 4 A corresponds to the SIMATIC S7 analog value of 27648.

Measurements of current are possible up to a SIMATIC S7 analog value of 32767. This corresponds to a current of 2.4 A. Currents above 2 A may occur only briefly.

Accuracy

The accuracy of the measurement of current amounts to 3% of the measuring range.

Note

This accuracy during current measurement can only be achieved if the time period is not changed during the measurement.

Function: Direct control of the digital output DO

Definition

You can directly control the digital output DO of the 2PULSE to test the actuator you have connected. To do this, you have to select the function from your control program (Page 9999) with the MANUAL_DO control bit set and SW_ENABLE control bit cleared. The function

- Supports you in commissioning a plant
- Allows you to utilize an unused channel of the 2PULSE as DO

After you have selected the function, the feedback bits (Page 10002)STS_ENABLE and ERR_PULS are cleared by the 2PULSE, and an active output sequence is aborted. The function takes precedence over the currently selected operating mode.

You specify the status of the digital output DO with the control bit (Page 9999)SET_DO.

When you clear the MANUAL_DO control bit, you deselect the "Direct control of the digital output DO" function. This deletes the digital output DO. You can then restart an output sequence with the rising edge of SW_ENABLE.

Control and Feedback Signals/Output Signal

Signals	Meaning	Value range	When the short control interface is used (8 bytes)		When the long control interface is used (12 bytes)	
			Channel 0 address	Channel 1 address	Channel 0 address	Channel 1 address
Control signals						
SW_ENABLE	To select the function, the control bit must be deleted.	0 = SW_ENABLE cleared. 1 = SW_ENABLE set	Byte 2: Bit 0	Byte 6: Bit 0	Byte 4: Bit 0	Byte 10: Bit 0
MANUAL_DO	You can select and deselect the function with the control bit.	0 = direct control of the DO not selected. 1 = direct control of the DO selected.	Byte 2: Bit 1	Byte 6: Bit 1	Byte 4: Bit 1	Byte 10: Bit 1
SET_DO	You use the control bit to set the status of the digital output DO.	0 = signal 0 on digital output DO 1 = signal 1 on digital output DO	Byte 2: Bit 2	Byte 6: Bit 2	Byte 4: Bit 2	Byte 10: Bit 2
Feedback signals						
STS_ENABLE	Deleted after the function has been selected.	0 = pulse output blocked 1 = pulse output running	Byte 0: Bit 0	Byte 4: Bit 0	Byte 0: Bit 0	Byte 4: Bit 0
STS_DO	Indicates the signal level at the digital output DO. Note the update rate.	0 = Signal 0 on digital output DO 1 = Signal 1 on digital output DO	Byte 0: Bit 1	Byte 4: Bit 1	Byte 0: Bit 1	Byte 4: Bit 1
STS_ DI	Indicates the signal level at digital input DI.	0 = Signal 0 on digital input DI 1 = Signal 1 on digital input DI	Byte 0: Bit 2	Byte 4: Bit 2	Byte 0: Bit 2	Byte 4: Bit 2
Output Signal						
Digital output DO	The status preset with the SET_DO control bit is output on the digital output DO.	0 = No signal 1 = Signal	4	8	4	8

Function: Error detection / diagnostics

Parameter Assignment ErrorERR_PARA

If the 2PULSE cannot identify the parameters as its own, it generates a parameter assignment error (Page 10002). The two channels are then not assigned parameters.

The 2PULSE slot you configure must match the setup.

Pulse output error ERR_PULS

The 2PULSE detects a channel-specific pulse output error in the operating modes "Pulse output (Page 9952)", "On/off-delay (Page 9974)", and "Pulse train (Page 9968)".

You can find information about causes and responses in the respective operating mode descriptions and under "Feedback interface of 2PULSE (Page 10002)".

The pulse output error detected is displayed for the relevant channel with the feedback bit (Page 10002)ERR_PULS.

Short circuit of the encoder supply ERR_24V

The 2PULSE detects a short circuit in the encoder supply that it makes available at terminals 2 and 6.

The short circuit error detected is displayed for the two channels with the feedback bit (Page 10002)ERR_24V.

A drop in the 24 V supply voltage can also result in a short circuit message.

Short circuit digital output ERR_DO

The 2PULSE detects a short circuit on the digital output of the channel.

The detected short circuit is displayed for the relevant channel with the feedback bit (Page 10002)ERR_DO.

Note

If a short circuit occurs, the digital output is able to briefly supply a current substantially higher than the rated value.

Diagnostic Message

If a short circuit occurs in the encoder supply or in the digital output, the 2PULSE generates a diagnostic message for the connected CPU/master. You must enable the "Group diagnostics (Page 9995)" parameter in this case.

Note

Diagnostic messages provide you the information that an error has occurred in one or both channels of the 2PULSE. You can evaluate the diagnostics with STEP 7.

The diagnostic record DS1 only contains the information for Channel 0. However, a diagnostic interrupt is also triggered at errors in Channel 1. The feedback signals provide detailed information about the occurring errors from both channels. Evaluate the feedback interface (Page 10002) in new user programs (see below).

Parameters

Parameter	Meaning	Value Range	Default
Group diagnostics (Page 9995)	When group diagnostics is enabled, the 2PULSE generates a diagnostic message for the CPU/master.	Disable/enable	Disable
Diagnostics DO (Page 9996)	The 2PULSE detects a short circuit of the digital outputs when "Diagnostics DO" = On.	Off/on	On

Feedback Signals

Feedback Signals	Meaning	Value Range	Channel 0 address	Channel 1 address
ERR_PARA	Indicates a parameter assignment error.	0 = no parameter assignment error 1 = parameter assignment error	Byte 0: Bit 5	Byte 4: Bit 5
ERR_PULS	Indicates a pulse output error.	0 = no pulse output error 1 = pulse output error	Byte 0: Bit 4	Byte 4: Bit 4
ERR_24V	Indicates a short circuit of the encoder supply.	0 = no encoder supply short circuit 1 = encoder supply short circuit	Byte 0: Bit 7	Byte 4: Bit 7
ERR_DO	Indicates a short circuit of the digital output DO.	0 = no digital output short circuit 1 = digital output short circuit	Byte 0: Bit 6	Byte 4: Bit 6

Parallel connection of both channels (as of 6ES7138-4DD01-0AB0)

Definition

In order to achieve higher output current, you can connect both channels of the 2PULSE in parallel. You then obtain a rated current of 4 A. This allows you to also operate actuators with a higher power requirement directly at the 2PULSE.

Parameter assignment and operation

You can enable parallel connection of the two channels during the configuration by selecting the check box "Parallel connection Channel 0 and 1 (Page 9995)".

Once enabled, the 2PULSE operates like a single-channel module. All the parameter assignments and operations via the control interface (Page 9999) have to be executed with Channel 0. Channel 1 cannot be configured and operated.

Feedback signals (Page 10002) are also only provided for Channel 0.

Measurement of current

During the measurement of current, the measured values from both channels are averaged once more. This results in a measuring range up to 4 A.

The measured value of the current is supplied as a SIMATIC S7 analog value in the feedback interface. A current of 4 A corresponds here to the SIMATIC S7 analog value of 27648.

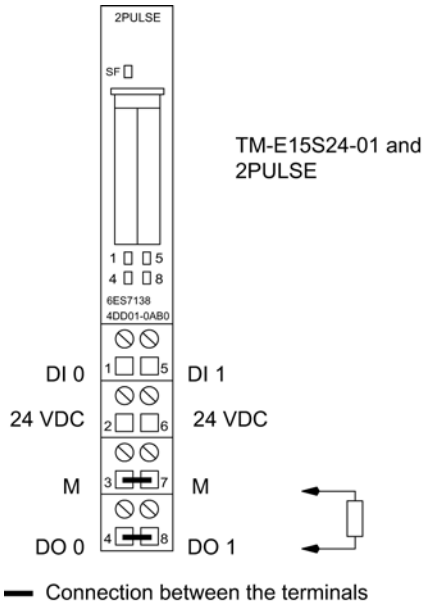
The accuracy of the current measurement also amounts to 3% of the measuring range in case of parallel connection.

Pin assignment for the parallel connection

The following terminals have to be connected at the terminal module for the parallel connection of both channels of the 2PULSE:

- Terminal 3 and Terminal 7
- Terminal 4 and Terminal 8

You then connect the load (e.g. a actuator) directly to Terminals 3 and 4 or to Terminals 7 and 8. The following figure shows a connection layout.



Reaction to CPU/Master STOP

Definition

You can configure the reaction (Page 9995) of the 2PULSE to failure of the higher-level controller for the two channels together.

Reaction to CPU/Master STOP	Channel-specific response and status of the 2PULSE
Turn off DO	Delete the DO digital output Delete STS_ENABLE and Terminate the current output sequence
Continue working mode	The DO digital output remains unchanged STS_ENABLE remains unchanged and The current output sequence is continued
DO substitute a value	Output of the channel-specific, substitute value that has been assigned parameters of the DO digital output Delete STS_ENABLE and Terminate the current output sequence
DO keeps last value	The DO digital output remains unchanged Delete STS_ENABLE and Terminate the current output sequence

Startup

To start a new output sequence after CPU/Master STOP and a set ACK_SW_ENABLE, first delete SW_ENABLE. Keep deleting until ACK_SW_ENABLE is also deleted.

If the mode is to continue during a change from CPU/Master STOP to RUN (startup), the CPU/Master cannot clear the outputs. **Possible solution:** In the part of the user program that is executed during startup, set the "Software enable" (SW_ENABLE = 1) control bit.

Modified Parameter Assignment

The status assumed by the 2PULSE at CPU/Master STOP remains even in the case of parameter assignment or configuration of the ET 200S station. This occurs, for example, at POWER ON of the CPU/master or the IM 151 or at the resumption of DP transfer.

In "Continue working mode", however, and after loading changed parameters or configuration of the ET 200S station to the CPU/master, the 2PULSE terminates the process. As a result, the 2PULSE does the following:

- Deletes the DO digital output.
- Deletes STS_ENABLE and
- Terminates the current output sequence.

Configuring and assigning parameters to 2PULSE

Parameters

Selecting an operating mode

Operating mode

Select the operating mode of the 2PULSE in accordance with your requirements:

- 2 Pulse: 8 bytes output data – compatible to predecessor module
  The control interface uses fewer addresses in the process image.
- 2 Pulse ext. control interface: 12 bytes output data – improved controlling possibilities
  You can enter the setpoints more conveniently.

Enabling group diagnostics

Group diagnostics

Group diagnostics are enabled in the respective check box.

When group diagnostics is enabled, the 2PULSE will generate a diagnostic message for the CPU/master.

Selecting reaction to CPU/Master STOP

Reaction to CPU/Master STOP

Select the desired response to CPU/Master STOP from the dropdown list:

- Turn off DO
- Continue working mode
- DO substitute a value
- DO keep last value

You will find further information on the response and mode of the 2PULSE under "Reaction to CPU/Master STOP (Page 9994)".

Activating parallel connection Channel 0 and 1

Parallel connection channel 0 and 1 (as of 6ES7138-4DD01-0AB0)

If you require a rated current of more than 2 A, you can connect the two channels of the 2PULSE in parallel. This is done by selecting the check box "Parallel connection Channel 0 and 1".

Additional information is available under "Parallel connection of both channels (as of 6ES7138-4DD01-0AB0) (Page 9992)".

Channel n

Activating DO diagnostics

Diagnostics DO

2PULSE will detect a short at the digital outputs if you have selected the "DO diagnostics" checkbox.

Selecting DO substitute value

Substitute value DO

Select the required substitute value for the digital output:

- 0
- 1

The parameter "Substitute value DO" may only have a value not equal to zero if the parameter "Reaction to CPU/Master STOP (Page 9995)" has been set to "DO substitute a value".

Selecting an operating mode

Operating mode

Select the required operating mode from the drop-down list:

- Pulse output (Page 9952)
- "Pulse width modulation" (Page 9958)
- Pulse train (Page 9968)
- On/Off delay (Page 9974)
- Frequency output (Page 9982) (as of 6ES7138-4DD01-0AB0)

Selecting the output format

Output format

The output format is relevant for operating modes "Pulse width modulation(PWM)" and "Frequency output" (as of 6ES7138-4DD01-0AB0).

- You can select the following output formats from the drop-down list in "Pulse width modulation (PWM) (Page 9958)" mode:
  - Per mill
  - S7 analog output
- You can select the following output formats from the drop-down list in "Frequency output (Page 9982)" mode:
  - 1 Hz
  - S7 analog output

Selecting a time base

Time base

Use the time base to select the resolution and value range for all predefined times:

- 0.1 ms
  You can set times from 0.2 ms to 6.5535 s with a resolution of 0.1 ms.
- 1 ms
  You can set times from 1 ms to 65.535 s with a resolution of 1 ms.

Selecting DI function

Function DI

You can use the DI digital input as an input or as a hardware enable.

Select the required function for the digital input from the drop-down list:

- Input
- Hardware enable

Specifying on delay

On-delay

The on delay is the time from the start of the output sequence to pulse output.

Enter the required on delay in the input field.

Valid value range: 0 … 65535

Specifying minimum pulse duration

Minimum pulse duration

Minimum pulse duration is relevant for operating modes "Pulse width modulation (PWM) (Page 9958)" and "Frequency output (Page 9982)" (as of 6ES7138-4DD01-0AB0).

You assign the minimum pulse duration and minimum break time using the "Minimum pulse duration" parameter; they always have the same value.

You can, for example, enter the response time of the actuator connected to your DO digital output here.

Valid value range: 0 … 65535

Specifying pulse duration

Pulse duration

Pulse duration is relevant for "Pulse train (Page 9968)" mode.

You can, for example, enter the response time of the actuator connected to your DO digital output here.

Valid value range:

- With time base 1 ms: from 1 ... 65535
- With time base 0.1 ms: from 2 ... 65535

Specifying period

Time period

The period is relevant for "Pulse width modulation (PWM) (Page 9958)" and "Pulse train (Page 9968)" operating modes.

When specifying the time period, take into account the set minimum pulse duration / pulse duration and the response time of the actuator connected to the digital output DO. The period should always be a multiple of the response time of the actuator connected to the DO digital output.

You specify the period as a numerical value in accordance with the time base:

- With time base 1 ms: from 1 ... 65535
- With time base 0.1 ms: from 2 ... 65535

Configured period = Time base x specified numerical value

2PULSE control and feedback interface

2PULSE control assignment

Note

The following control interface data is consistent for 2PULSE:

- Bytes 0 ... 3
- Bytes 4 ... 7

Assignment of the short control interface (8 bytes)

The following table shows the assignment of the short control interface (outputs) of 2PULSE.

Address		Assignment
Channel 0	Channel 1	
Word 0	Word 4	Depending on the mode • Pulse output: Pulse duration • Pulse-width modulation: Output value • Pulse train: Number of pulses • On/Off-delay: Off-delay • Frequency output: Output frequency (as of 6ES7138-4DD01-0AB0)
Byte 2	Byte 6	Bit 7: Reserve = 0 Bit 6: Reserve = 0 Bit 5: Reserve = 0 Bit 4: Reserve = 0 Bit 3: Reserve = 0 Bit 2: SET_DO Bit 1: MANUAL_DO Bit 0: SW_ENABLE
Byte 3	Byte 7	Depending on mode • Pulse output: On-delay factor • Pulse width modulation: Period duration factor • Pulse train: Period duration factor • On/Off-delay: On-delay factor • Frequency output: On-delay factor (as of 6ES7138-4DD01-0AB0)

Assignment of the long control interface (12 bytes) (as of 6ES7138-4DD01-0AB0)

The following table shows the assignment of the long control interface (outputs) of 2PULSE.

Address		Assignment
Channel 0	Channel 1	
Word 0	Word 6	Depending on mode • Pulse output: Pulse duration • Pulse width modulation: Output value • Pulse train: Number of pulses • On/Off-delay: Off-delay • Frequency output: Output frequency
Word 2	Word 8	Depending on mode • Pulse output: On-delay • Pulse width modulation: Period • Pulse train: Period • On/Off-delay: On-delay • Frequency output: On-delay
Byte 4	Byte 10	Bit 7: Reserve = 0 Bit 6: Reserve = 0 Bit 5: Reserve = 0 Bit 4: Reserve = 0 Bit 3: Reserve = 0 Bit 2: SET_DO Bit 1: MANUAL_DO Bit 0: SW_ENABLE
Byte 5	Byte 11	Depending on mode • Pulse output: Not used • Pulse width modulation: On-delay factor • Pulse train: Pulse-pause ratio • On/Off-delay: Not used • Frequency output: Not used

Notes on the Control Signals

Control Signal	Explanation
"Pulse output" operating mode:	
• Pulse duration [1) 2)]	The time that is set for the DO digital output on expiration of the on-delay.
• On-delay factor [1)]	You can change the configured on-delay before the start of the output sequence: On-delay = Factor × 0.1 × configured on-delay
• On-delay [2)]	The time from the start of the output sequence to the output of the pulse train.
"Pulse width modulation" operating mode:	

Control Signal	Explanation
• Output value [1] [2]	Value that is output with pulse-width modulation at the DO digital output on expiration of the on-delay.
• Time period factor [1]	You can change the configured period duration: Period duration = Factor × 0.1 × configured period duration
• Time period [2]	When specifying the time period take into account the set minimum pulse duration and the response time of your actuator that is connected to the digital output DO.
• On-delay factor [2]	You can change the configured on-delay before the start of the output sequence: On-delay = Factor × 0.1 × configured on-delay
"Pulse train" operating mode:	
• Number of pulses [1] [2]	Number of pulses that are output at the DO digital output on expiration of the on-delay.
• Period factor [1]	You can change the configured time period before the start of the output sequence: Time period = Factor x 0.1 x configured time period
• Period [2]	When specifying the period, take into account the set minimum pulse duration and the response time of the actuator connected to the digital output DO.
• Pulse-pause ratio [2]	You can set the ratio of pulse duration to break time: Pulse/Break ratio = (Setpoint / 255) × Period
"On/Off-delay" operating mode:	
• Off-delay [1] [2]	The time between a negative edge of the digital input DI and its output on the digital output DO.
• On-delay factor [1]	You can change the configured on-delay before the start of the output sequence: On-delay = Factor × 0.1 × configured on-delay
• On-delay [2]	The time between a positive edge of the digital input DI and its output on the digital output DO.
"Frequency output" mode(as of 6ES7138-4DD01-0AB0)**:**	
• Output frequency [1] [2]	Frequency that is output at the DO digital output after expiration of the on-delay.
• On-delay factor [1]	You can change the configured on-delay before the start of the output sequence: On-delay = Factor × 0.1 × configured on-delay
• On-delay [2]	The time from the start of the output sequence to the output of the frequency.
Direct control of the digital output	
• MANUAL_DO	You use the control bit to select and deselect the "Direct control of the digital output DO" function.
• SET_DO	You use the control bit to set the status of the digital output DO.
Software enable	
• SW_ENABLE	You must always issue the software enable in your control program. If you do not use a hardware enable, the output sequence will be started by the positive edge of the software enable. If you delete the software enable, the current output sequence will be terminated.

[1] When using the short control interface

[2] When using the long control interface

2PULSE feedback interface

Note

The following data of the feedback interface are consistent for the 2PULSE:

- Bytes 0 ... 3
- Bytes 4 ... 7

Assignment of the Feedback Interface

The following table shows the assignment of the feedback interface (inputs) of 2PULSE.

Address		Assignment
Channel 0	Channel 1	
Byte 0	Byte 4	Bit 7: ERR_24V Bit 6: ERR_DO Bit 5: ERR_PARA Bit 4: ERR_PULS Bit 3: ACK_SW_ENABLE Bit 2: STS_DI Bit 1: STS_DO Bit 0: STS_ENABLE
Byte 1	Byte 5	Bit 0 to Bit 3: Sequence counter SEQ_CNT (as of 6ES7138-4DD01-0AB0) Bit 4 to Bit 7: Reserve = 0
Word 2	Word 6	Current measurement (as of 6ES7138-4DD01-0AB0)

Note

In line with the IM 151 interface module used and with the configuration, a feedback value (substitute value) of either 16#7FFF7FFF or 16#00000000 will be output if the module fails.

Notes on the feedback signals

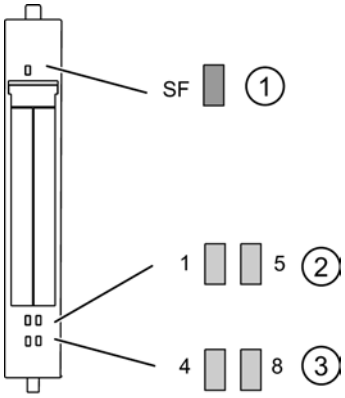
Checkback signal	Explanation
ACK_SW_ENABLE	Indicates the status of the software enable pending at the 2PULSE.
ERR_24V	Indicates a short circuit or drop of the encoder supply.
ERR_DO	Indicates a short circuit at the digital output.
ERR_PARA	Indicates a parameter assignment error.

Checkback signal	Explanation
ERR_PULS	"Pulse output" operating mode: Indicates a pulse output error. If the pulse duration is reduced after expiration of the on-delay so the time is less than the time already output, this is detected by the 2PULSE. The 2PULSE deletes the feedback bit ERR_PULS next time the output sequence starts. "Pulse train" operating mode: Indicates a pulse output error. If the number of pulses is reduced after expiration of the on-delay and the smaller number of pulses is already output, this is detected by the 2PULSE. The 2PULSE deletes the feedback bit ERR_PULS next time the output sequence starts. "On/Off-delay" mode: Indicates a pulse output error if the pulse duration or interpulse period is too short. The 2PULSE deletes the ERR_PULS feedback bit at the next positive edge of the software enable or at the next edge at the DI digital input.
STS_ DI	Indicates the signal level at the digital input DI.
STS_DO	Indicates the signal level at the digital output DO.
STS_ENABLE	"Pulse output" operating mode: Is set at the start of the output sequence until the pulse duration expires. If you delete the software enable (SW_ENABLE) or the 2PULSE detects a pulse output error (ERR_PULS), STS_ENABLE is deleted. "Pulse-width modulation" operating mode: Set at start of output sequence. If you delete the software enable (STS_ENABLE), SW_ENABLE is deleted. "Pulse train" operating mode: Is set at the start of the output sequence until the output of the last pulse. If you delete the software enable (SW_ENABLE) or the 2PULSE detects a pulse output error (ERR_PULS), STS_ENABLE will be deleted. "On/Off-delay" mode: Indicates the status of the software enable (SW_ENABLE) detected by the 2PULSE. "Frequency output" mode (as of 6ES7138-4DD01-0AB0): Set at start of output sequence. If you delete the software enable (STS_ENABLE), SW_ENABLE will be deleted.
SEQ_CNT (as of 6ES7138-4DD01-0AB0)	Counts the completed output sequences in the "Pulse output", "Pulse train" and "On/Off-delay" operating modes. No function in the "Pulse width modulation" and "Frequency output" operating modes.
Current measurement (as of 6ES7138-4DD01-0AB0)	Current at the DO as a SIMATIC S7 analog value

2PULSE diagnostics

Diagnostics using the LED display

LED display at the 2PULSE



① Group error (red)
② Status display for digital input (green)
③ Status display for the digital output (green)

Status and error displays by means of LEDs on the 2PULSE

The table below shows the status and fault displays on the 2PULSE.

Event (LED)					Cause	Remedy
SF	1	5	4	8		
On					No parameter assignment. There is a diagnostic message.	Check the error bits types in the feedback interface. Evaluate the diagnostics data.
	On				Input DI0 on Channel 0 activated.	
		On			Input DI1 on Channel 1 activated.	
			On		Output DO0 on Channel 0 activated.	
				On	Output DO1 on Channel 1 activated.	

Error types

For information on the structure of the channel-related diagnostics, refer to the manual on the interface module used in your ET 200S station.

2PULSE error types

The following table shows the 2PULSE error types.

Error class		Meaning	Remedy
1_D	00001: Short circuit	Short circuit of the sensor supply or the actuator. A drop in the 24 V supply voltage can also result in this error message.	Check the wiring to the switches and actuators. Correct the process wiring.
9_D	01001: Error	Internal module error occurred.	Replace the module.
16_D	10000: Parameter assignment error	Parameters have not been assigned to the module.	Correct the parameter assignment.

A simple error evaluation is also possible through the corresponding bits in the feedback interface (Page 10002).

11.4 Motion Control

11.4.1 Motion Control (S7-1200, S7-1500)

11.4.1.1 Using S7-1200 Motion Control

Introduction

Motion functionality of the CPU S7-1200

The TIA Portal, together with the "Motion Control" functionality of the CPU S7-1200, supports you in controlling stepper motors and servo motors with pulse interface:

- You configure the positioning axis and command table technology objects in the TIA Portal. The CPU S7-1200 uses these technology objects to control the pulse generator outputs that control the drives.
- In the user program you control the axis by means of Motion Control instructions and initiate motion commands of your drive.

You can find a multi-media introduction on the Internet (http://www.automation.siemens.com/mcms/topics/en/simatic/simatic-technology/integrated-functions/simatic-s7-1200/Pages/Default.aspx).

See also

Hardware components for motion control (Page 10006)

Integration of the positioning axis technology object (Page 10026)

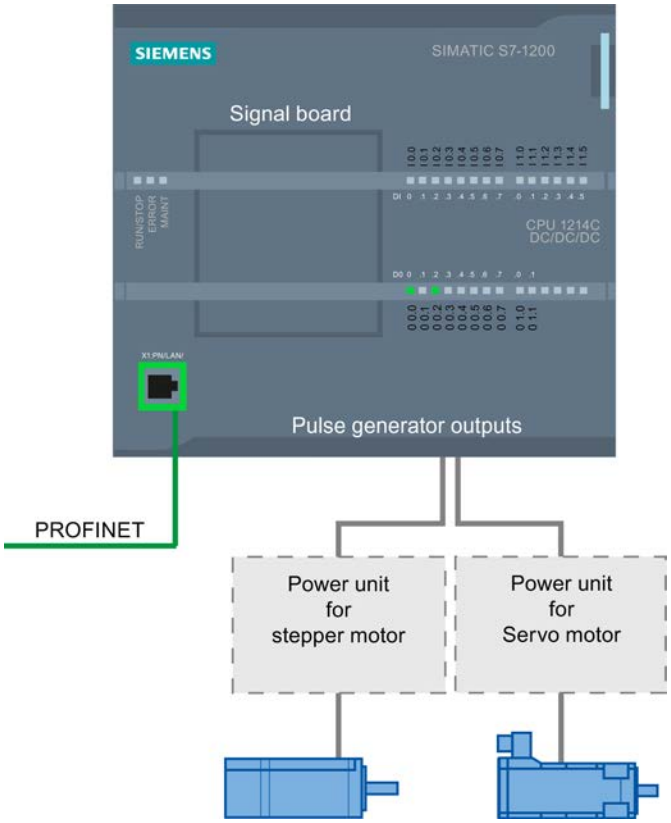
Tools of the positioning axis technology object (Page 10029)

Use of the command table technology object (Page 10055)

Command table technology object tools (Page 10055)

Hardware components for motion control

The representation below shows the basic hardware configuration for a motion control application with the CPU S7-1200.



CPU S7-1200:

CPU S7-1200 combines the functionality of a programmable logic controller with motion control functionality for operation of stepper motors and servo motors with pulse interface. The motion control functionality takes over the control and monitoring of the drives.

The DC/DC/DC variants of the CPU S7-1200 have on-board outputs for direct control of drives. The relay variants of the CPU require one of the signal boards described below to control a drive.

Signal board

You add further inputs and outputs to the CPU with the signal boards. The digital outputs can be used as pulse generator outputs for controlling drives as required.

In CPUs with relay outputs, the pulse signal cannot be output on the on-board outputs because the relays do not support the necessary switching frequencies. A signal board with digital outputs must be used to enable you to work with the PTO (Pulse Train Output) on these CPUs.

PROFINET

Use the PROFINET interface to establish the online connection between the CPU S7-1200 and the programming device. In addition to the online functions of the CPU, additional commissioning and diagnostic functions are available for motion control.

Ordering information for CPU Firmware V4

The order information listed below applies to the currently installed product phase (without any installed Hardware Support Packages) of the TIA Portal.

Name	MLFB (order number)
CPU 1211C DC/DC/DC	6ES7211-1AE40-0XB0
CPU 1211C AC/DC/RLY	6ES7211-1BE40-0XB0
CPU 1211C DC/DC/RLY	6ES7211-1HE40-0XB0
CPU 1212C DC/DC/DC	6ES7212-1AE40-0XB0
CPU 1212C AC/DC/RLY	6ES7212-1BE40-0XB0
CPU 1212C DC/DC/RLY	6ES7212-1HE40-0XB0
CPU 1214C DC/DC/DC	6ES7214-1AG40-0XB0
CPU 1214C AC/DC/RLY	6ES7214-1BG40-0XB0
CPU 1214C DC/DC/RLY	6ES7214-1HG40-0XB0
CPU 1214FC DC/DC/DC	6ES7214-1AF40-0XB0
CPU 1214FC DC/DC/RLY	6ES7214-1HF40-0XB0
CPU 1215C DC/DC/DC	6ES7215-1AG40-0XB0
CPU 1215C AC/DC/RLY	6ES7215-1BG40-0XB0
CPU 1215C DC/DC/RLY	6ES7215-1HG40-0XB0
CPU 1215FC DC/DC/DC	6ES7215-1AF40-0XB0
CPU 1215FC DC/DC/RLY	6ES7215-1HF40-0XB0
CPU 1217C DC/DC/DC	6ES7217-1AG40-0XB0
Signal board DI2/DO2 x DC24V 20kHz	6ES7223-0BD30-0XB0
Signal board DI2/DO2 x DC24V 200kHz	6ES7223-3BD30-0XB0
Signal board DO4 x DC24V 200kHz	6ES7222-1BD30-0XB0
Signal board DI2/DO2 x DC5V 200kHz	6ES7223-3AD30-0XB0
Signal board DO4 x DC5V 200kHz	6ES7222-1AD30-0XB0

Use a Hardware Support Package (HSP) to install new hardware components. The hardware component will then be available in the hardware catalog.

See also

Motion functionality of the CPU S7-1200 (Page 10005)

CPU outputs relevant for motion control (technology version V4) (Page 10008)

Basics for working with S7-1200 Motion Control

CPU outputs relevant for motion control (technology version V4)

The number of usable drives depends on the number of PTOs (pulse train outputs) and the number of available pulse generator outputs.

The following tables provide information about the relevant dependencies:

Maximum number of PTOs

4 PTOs are available for each CPU with technology version V4. This means a maximum of 4 drives can be controlled.

Signal type of the PTO

Depending on the signal type of the PTO, 1-2 pulse generator outputs are required per PTO (drive):

Signal type	Number of pulse generator outputs
Pulse A and direction B (direction output disabled) *)	1
Pulse A and direction B *)	2
Count up A, count down B	2
A/B phase offset	2
A/B phase offset - quadruple	2

*) The direction output must be on-board or on a signal board.

Usable pulse generator outputs and limit frequencies

The relay variants of the CPUs can only access the pulse generator outputs of a signal board.

Depending on the CPU and signal board, the following pulse signal generator outputs can be used with the following limit frequencies:

On-board	Q0.0	Q0.1	Q0.2	Q0.3	Q0.4	Q0.5	Q0.6	Q0.7	Q1.0	Q1.1
CPU 1211 (DC/DC/DC)	100 kHz	100 kHz	100 kHz	100 kHz	-	-	-	-	-	-
CPU 1212 (DC/DC/DC)	100 kHz	100 kHz	100 kHz	100 kHz	30 kHz	30 kHz	-	-	-	-
CPU 1214(F) (DC/DC/DC)	100 kHz	100 kHz	100 kHz	100 kHz	30 kHz	30 kHz	30 kHz	30 kHz	30 kHz	30 kHz

On-board	Q0.0	Q0.1	Q0.2	Q0.3	Q0.4	Q0.5	Q0.6	Q0.7	Q1.0	Q1.1
CPU 1215(F) (DC/DC/DC)	100 kHz	100 kHz	100 kHz	100 kHz	30 kHz	30 kHz	30 kHz	30 kHz	30 kHz	30 kHz
CPU 1217 (DC/DC/DC)	1 MHz	1 MHz	1 MHz	1 MHz	100 kHz	100 kHz	100 kHz	100 kHz	100 kHz	100 kHz
Signal board	**Qx.0**	**Qx.1**	**Qx.2**	**Qx.3**	-	-	-	-	-	-
Signal board DI2/DO2 x DC24V 20kHz	20 kHz	20 kHz	-	-	-	-	-	-	-	-
Signal board DI2/DO2 x DC24V 200kHz	200 kHz	200 kHz	-	-	-	-	-	-	-	-
Signal board DO4 x DC24V 200kHz	200 kHz	200 kHz	200 kHz	200 kHz	-	-	-	-	-	-
Signal board DI2/DO2 x DC5V 200kHz	200 kHz	200 kHz	-	-	-	-	-	-	-	-
Signal board DO4 x DC5V 200kHz	200 kHz	200 kHz	200 kHz	200 kHz	-	-	-	-	-	-

The low limit frequency is always 1Hz.

The pulse generator outputs can be freely assigned to the PTOs.

Note

If pulse generator outputs with different limit frequencies are used in accordance with the signal type, the low limit frequency is used in each case.

Signal type "Pulse A and direction B" is an exception. With this type of signal, the limit frequency of the pulse generator output is always used.

Note

Access to pulse generator outputs via the process image

The firmware takes control via the corresponding pulse generator and direction outputs if the PTO (Pulse Train Output) is selected and assigned to an axis.

With this takeover of the control function, the connection between the process image and I/O output is also disconnected. Although the user has the option of writing the process image of pulse generator and direction outputs via the user program or watch table, this is not transferred to the I/O output. Accordingly, it is also not possible to monitor the I/O output via the user program or watch table. The information read reflects the value of the process image and does not match the real status of the I/O output.

For all other CPU outputs that are not used permanently by the CPU firmware, the status of the I/O output can be controlled or monitored via the process image, as usual.

Outputs for drive signals

For motion control, you can optionally parameterize a drive interface for "Drive enabled" and "Drive ready".

When using the drive interface the digital output for the drive enable and the digital input for "drive ready" can be freely selected.

Acceleration/deceleration limits

The following limits apply to acceleration and deceleration:

Acceleration/deceleration	Value
Minimum acceleration/deceleration	5.0E-3 pulses/s^2
Maximum acceleration/deceleration	9.5E+9 pulses/s^2

Jerk limits

The following limits apply to the jerk:

Jerk	Value
Minimum jerk	4.0E-3 pulses/s^3
Maximum jerk	1.0E+10 pulses/s^3

See also

CPU outputs relevant for motion control (technology version V1...3) (Page 10162)

How the pulse interface works (Page 10010)

Relationship between the signal type and the direction of travel (Page 10011)

Hardware and software limit switches (Page 10015)

Jerk limit (Page 10016)

Homing (Page 10017)

Hardware components for motion control (Page 10006)

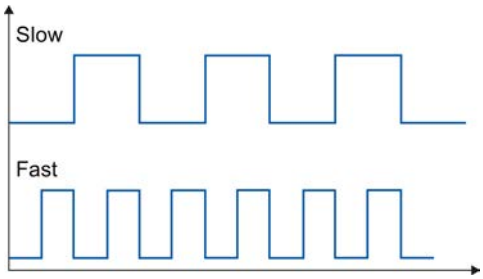
Integration of the positioning axis technology object (Page 10026)

Tools of the positioning axis technology object (Page 10029)

How the pulse interface works

Depending on the settings of the stepper motor, each pulse affects the movement of the stepper motor by a specific angle. If the stepper motor is set to 1000 pulses per revolution, for example, it moves 0.36° per pulse.

The speed of the stepper motor is determined by the number of pulses per time unit.



(The statements made here also apply to servo motors with pulse interface.)

See also

CPU outputs relevant for motion control (technology version V4) (Page 10008)

Relationship between the signal type and the direction of travel (Page 10011)

Hardware and software limit switches (Page 10015)

Jerk limit (Page 10016)

Homing (Page 10017)

Integration of the positioning axis technology object (Page 10026)

Tools of the positioning axis technology object (Page 10029)

Relationship between the signal type and the direction of travel

The CPU outputs the velocity and direction of travel via two outputs.

The relationships between the configuration and direction of travel differ depending on the selected signal type. You can configure the following signal types in the axis configuration under "Basic parameters > General":

- "PTO (pulse A and direction B)"
- "PTO (count up A, count down B)" (as of V4)
- "PTO (A/B phase offset)" (as of V4)
- "PTO (A/B phase offset - quadruple)" (as of V4)

You configure the direction under "Extended Parameters > Mechanics" in the axis configuration. If you select the "Invert direction" option, the direction logic described below for the respective signal type is inverted.

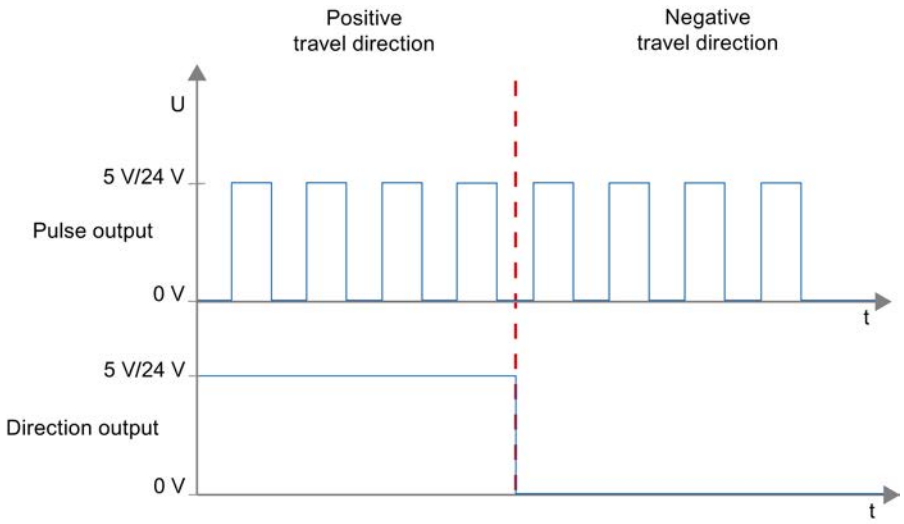
PTO (pulse A and direction B)

The pulse output pulses and the direction output level are evaluated for this signal type.

The pulses are output via the pulse output of the CPU. The direction output of the CPU specifies the direction of rotation of the drive:

- 5 V/24 V at direction output ⇒ positive direction of rotation
- 0 V at direction output ⇒ negative direction of rotation

The specified voltage depends on the hardware used. The indicated values do not apply to the differential outputs of CPU 1217.

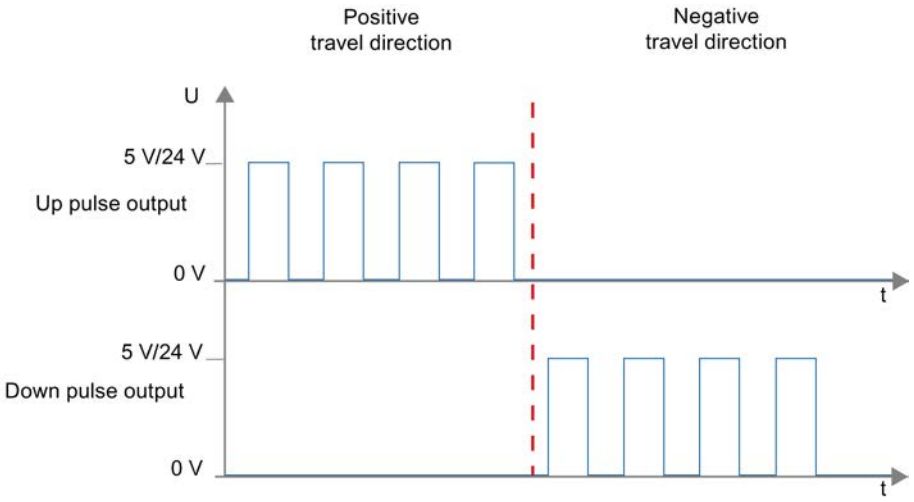


PTO (count up A, count down B) (as of V4)

The pulses of an output are evaluated for this signal type.

The pulse for the positive direction is output via the "Up pulse output" The pulse for the negative direction is output via the "Down pulse output"

The specified voltage depends on the hardware used. The values indicated do not apply to the differential outputs of CPU 1217.



PTO (A/B phase offset) (as of V4)

The positive edges of one output in each case are evaluated for this signal type.

The pulse is output via the "Phase A" output and phase-shifted via the "Phase B" output. The phase shifting between the outputs defines the direction of rotation:

- Phase A leads phase B by 90° ⇒ positive direction of rotation
- Phase B leads phase A by 90° ⇒ negative direction of rotation

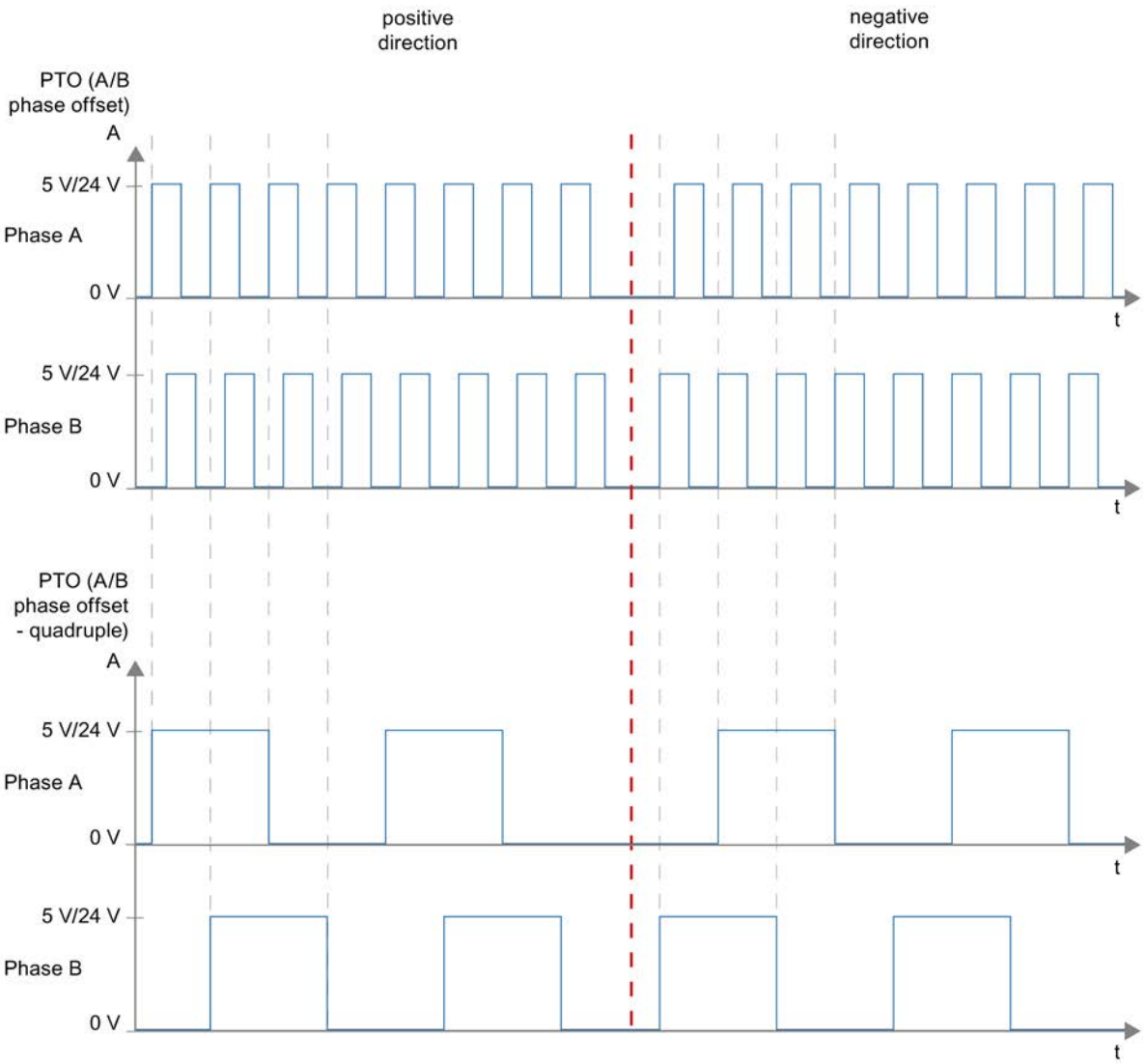
PTO (A/B phase offset - quadruple) (as of V4)

The positive and negative edges of both outputs are evaluated for this signal type. A pulse period has four edges with two phases (A and B). The pulse frequency at the output is therefore reduced to a quarter.

The pulse is output via the "Phase A" output and phase-shifted via the "Phase B" output. The phase shifting between the outputs defines the direction of rotation:

- Phase A leads phase B by 90° ⇒ positive direction of rotation
- Phase B leads phase A by 90° ⇒ negative direction of rotation

The specified voltage depends on the hardware used. The indicated values do not apply to the differential outputs of CPU 1217.



Invert direction

If you select the "Invert rotation signal" option, the direction logic is inverted:

- **PTO (pulse A and direction B)**
  - 0 V at direction output (low level) ⇒ positive direction of rotation
  - 5 V/24 V at direction output (high level) ⇒ negative direction of rotation

  The specified voltage depends on the hardware used. The voltages indicated do not apply to the differential outputs of CPU 1217.
- **PTO (count up A, count down B)**
  The outputs "Pulse output down" and "Pulse output up" are swapped.

- PTO (A/B phase offset)
  The "Phase A" and "Phase B" outputs are swapped.
- "PTO (A/B phase offset - quadruple)
  The "Phase A" and "Phase B" outputs are swapped.

See also

CPU outputs relevant for motion control (technology version V4) (Page 10008)

How the pulse interface works (Page 10010)

Hardware and software limit switches (Page 10015)

Jerk limit (Page 10016)

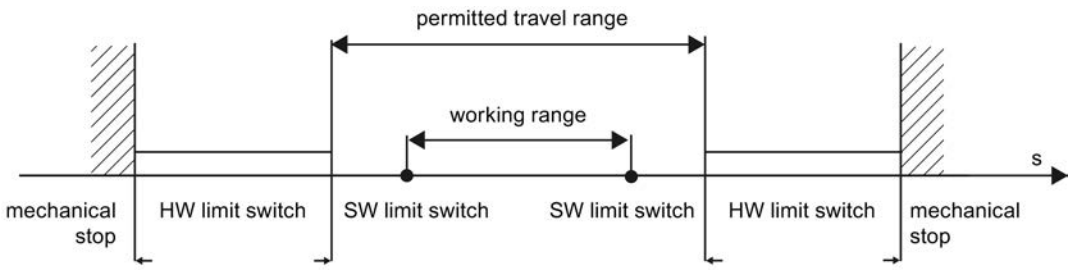
Homing (Page 10017)

Integration of the positioning axis technology object (Page 10026)

Tools of the positioning axis technology object (Page 10029)

Hardware and software limit switches

Use the hardware and software limit switches to limit the "permitted traversing range" and the "working range" of your positioning axis technology object. The relationships are shown in the following diagram:



Hardware limit switches are limit switches that limit the maximum "permitted traversing range" of the axis. Hardware limit switches are physical switching elements that must be connected to interrupt-capable inputs of the CPU.

Software limit switches limit the "working range" of the axis. They should fall inside the hardware limit switches relative to the traversing range. Since the positions of the software limit switches can be flexibly set, the working range of the axis can be adapted on an individual basis, depending on the current traversing profile. In contrast to hardware limit switches, software limit switches are implemented exclusively via the software and do not require their own switching elements.

Hardware and software limit switches must be activated prior to use in the configuration or in the user program.. Software limit switches are only active after homing the axis.

See also

CPU outputs relevant for motion control (technology version V4) (Page 10008)

How the pulse interface works (Page 10010)

Relationship between the signal type and the direction of travel (Page 10011)

Jerk limit (Page 10016)

Homing (Page 10017)

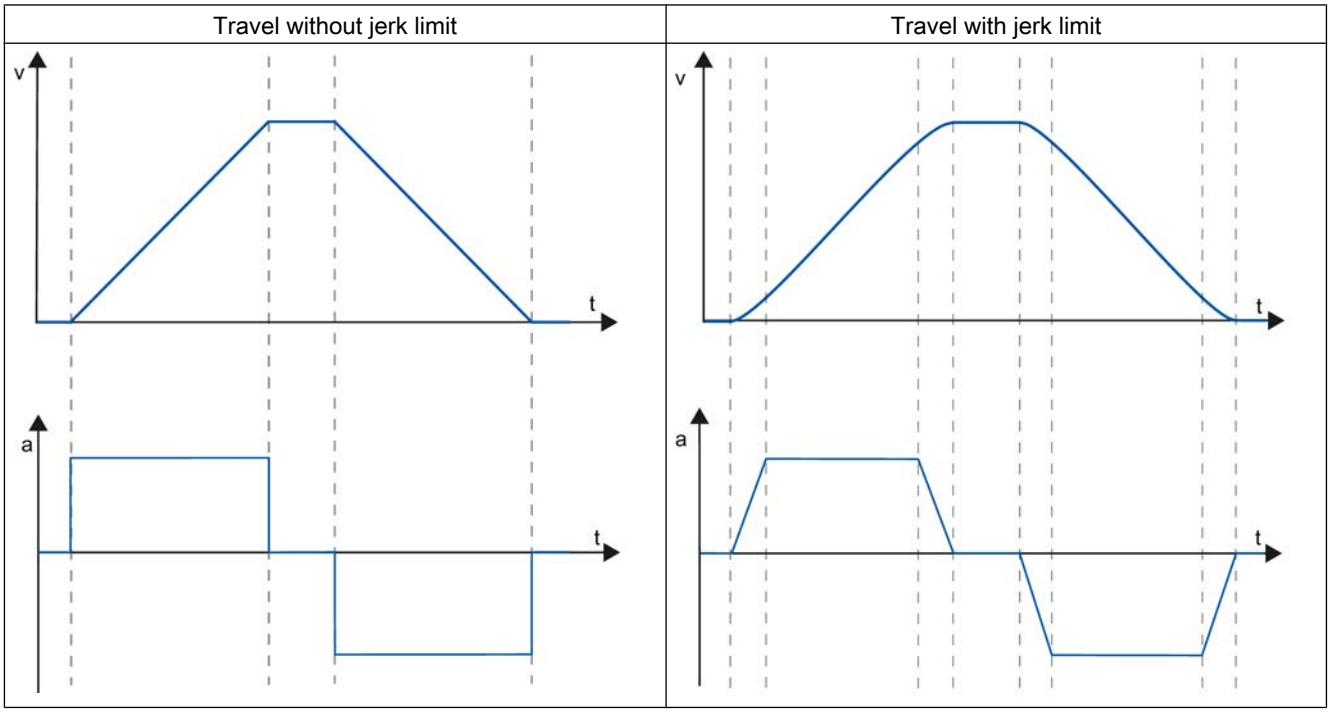
Integration of the positioning axis technology object (Page 10026)

Tools of the positioning axis technology object (Page 10029)

Position limits (Page 10037)

Jerk limit

With the jerk limit you can reduce the stresses on your mechanics during an acceleration and deceleration ramp. The acceleration and deceleration value is not changed abruptly when the jerk limiter is active; it is gradually increased and decreased. The figure below shows the velocity and acceleration curve without and with jerk limit.



The jerk limit gives a "smoothed" velocity profile of the axis motion. This ensures, for example, soft starting and braking of a conveyor belt.

See also

Behavior of the axis when using the jerk limit (Page 10046)

CPU outputs relevant for motion control (technology version V4) (Page 10008)

How the pulse interface works (Page 10010)

Relationship between the signal type and the direction of travel (Page 10011)

Hardware and software limit switches (Page 10015)

Homing (Page 10017)

Integration of the positioning axis technology object (Page 10026)

Tools of the positioning axis technology object (Page 10029)

Homing

Homing means matching the axis coordinates of the technology object to the real, physical location of the drive. For position-controlled axes the entries and displays for the position refer exactly to these axis coordinates. Therefore, agreement between the axis coordinates and the real situation is extremely important. This step is necessary to ensure that the absolute target position of the axis is also achieved exactly with the drive.

In the S7-1200 CPU, axis homing is implemented with the motion control instruction, "MC_Home". The following homing modes exist

Homing modes

- Active homing
  In active homing mode, the motion control instruction "MC_Home" performs the required reference point approach. When the homing switch is detected, the axis is homed according to the configuration. Active traversing motions are aborted.
- Passive homing
  During passive homing, the motion control instruction "MC_Home" does not carry out any homing motion. The traversing motion required for this step must be implemented by the user via other motion control instructions. When the homing switch is detected, the axis is homed according to the configuration. Active traversing motions are not aborted upon start of passive homing.
- Direct homing absolute
  The axis position is set regardless of the homing switch. Active traversing motions are not aborted. The value of input parameter "Position" of motion control instruction "MC_Home" is set immediately as the reference point of the axis.
- Direct homing relative
  The axis position is set regardless of the homing switch. Active traversing motions are not aborted. The following statement applies to the axis position after homing:
  New axis position = current axis position + value of parameter "Position" of instruction "MC_Home".

See also

CPU outputs relevant for motion control (technology version V4) (Page 10008)

How the pulse interface works (Page 10010)

Relationship between the signal type and the direction of travel (Page 10011)

Hardware and software limit switches (Page 10015)

Jerk limit (Page 10016)

Integration of the positioning axis technology object (Page 10026)

Tools of the positioning axis technology object (Page 10029)

Homing (positioning axis technology object as of V2) (Page 10048)

Guidelines on use of motion control

The guidelines described here present the basic procedure for using motion control with the CPU S7-1200.

Requirements

To use the positioning axis technology object, a project with a CPU S7-1200 must be created.

Procedure

Follow the steps below in the order given to use motion control with the CPU S7-1200. Use the following links for this purpose:

1. Adding a positioning axis technology object (Page 10031)
2. Working with the configuration dialog (Page 10031)
3. Download to CPU (Page 10074)
4. Function test of the axis in the commissioning window (Page 10075)
5. Programming (Page 10096)
6. Diagnostics of the axis control (Page 10114)

Using versions

Overview of versions

The relationship between the relevant versions for S7-1200 Motion Control can be found in the following table:

Technology version

You can check the currently selected technology version in the "Instructions" task card in folder "Technology > Motion Control > S7-1200 Motion Control" and in the "Add new object" dialog.

Select the technology version in the "Instructions" task card in folder "Technology > Motion Control > S7-1200 Motion Control".

If a technology object with an alternative version is added in the "Add new object" dialog, the technology version will also be changed.

Note

The selection of an alternative technology version will also affect the Motion Control Instructions version (task card).

The technology objects and Motion Control instructions will only be converted to the selected version upon compilation or "Load to device".

Version of the technology object

The version of a technology object can be checked in the inspector window under "Properties > General > Information" in the "Version" box.

Motion Control instruction version

The Motion Control instruction version can be checked in the inspector window under "Properties > General > Information" in the "Version" box.

If the Motion Control instruction version used is not in line with the following compatibility list, the relevant Motion Control instructions will be highlighted in the program editor.

Compatibility list

Technology		CPU	Technology object	Motion Control Instruction
V1.0		V1.0, V2.0, V2.1, V2.2, V3.0	Axis V1.0	MC_Power V1.0 MC_Reset V1.0 MC_Home V1.0 MC_Halt V1.0 MC_MoveAbsolute V1.0 MC_MoveRelative V1.0 MC_MoveVelocity V1.0 MC_MoveJog V1.0
V2.0	Innovations: • Jerk limit • Command table • MC_ChangeDynamic	V2.1, V2.2, V3.0	Axis V2.0, Command table V2.0	MC_Power V2.0 MC_Reset V2.0 MC_Home V2.0 MC_Halt V2.0 MC_MoveAbsolute V2.0 MC_MoveRelative V2.0 MC_MoveVelocity V2.0 MC_MoveJog V2.0 MC_CommandTable V2.0 MC_ChangeDynamic V2.0

Technology		CPU	Technology object	Motion Control Instruction
V3.0	Innovation: Load in RUN operating mode	V2.2, V3.0, V4.0	Axis V3.0, Command table V3.0	MC_Power V3.0 MC_Reset V3.0 MC_Home V3.0 MC_Stop V3.0 MC_MoveAbsolute V3.0 MC_MoveRelative V3.0 MC_MoveVelocity V3.0 MC_MoveJog V3.0 MC_CommandTable V3.0 MC_ChangeDynamic V3.0
V4.0	Innovations: • MC_ReadParam • MC_WriteParam • Standardization of S7-1200 and S7-1500 Motion Control technology data blocks.	V4.0	Positioning axis V4.0, command table V4.0	MC_Power V4.0 MC_Reset V4.0 MC_Home V4.0 MC_Halt V4.0 MC_MoveAbsolute V4.0 MC_MoveRelative V4.0 MC_MoveVelocity V4.0 MC_MoveJog V4.0 MC_CommandTable V4.0 MC_ChangeDynamic V4.0 MC_ReadParam V4.0 MC_WriteParam V4.0

See also

Changing a technology version (Page 10020)

Compatibility list of tags (Page 10021)

Status of limit switch (Page 10024)

Changing a technology version

Before you can access all the benefits of a new technology version, you may need to set / modify the technology version for existing projects.

Note

Compatibility of the technology object tags

If you are switching from V1...3 and V4, please see the compatibility list (Page 10021) if you are using tags of the technology object in your user program, monitoring tables, etc.

Setting/changing a technology version

To set or change the technology version, follow these steps:

1. Open the programming editor (e.g., by opening the OB1).
2. In the "Instructions" task card, open the desired technology version in the "Technology > Motion Control > S7-1200 Motion Control" folder.
3. Save and compile the project. Pay attention to any error information that is displayed during compilation. Deal with the causes of the errors indicated.
4. Check the configuration of the technology objects.
5. If necessary, adapt the tag names in the following objects in line with the compatibility list.

- User program
- Monitoring tables
- Force tables
- HMI configuration
- Trace configuration

See also

Overview of versions (Page 10018)

Status of limit switch (Page 10024)

Compatibility list of tags

The technology data blocks for S7-1200 Motion Control and S7-1500 Motion Control have been standardized within the framework of the V4 technology. This has resulted in new tags and tag names for the V4 positioning axis and V4 command table technology objects.

Note the following tables if you have used tags of the technology objects in the user program and you want to convert the project from V1...3 to V4 (or vice versa).

The tags listed in the "Automatic conversion V1...3 to V4" column are converted automatically when the project is compiled. Tag names in monitoring and force tables or the HMI or trace configuration are not converted.

The following tags are new or have been adapted and may have to be corrected in the user program, monitoring tables, etc.:

Config tags (positioning axis)

Tag name V1.0 to V3.0	Tag name V4.0	Automatic conversion V1..3 to V4
<Axis name>.Config.DynamicDefaults.Acceleration	<Axis name>.DynamicDefaults.Acceleration	Yes
<Axis name>.Config.DynamicDefaults.Deceleration	<Axis name>.DynamicDefaults.Deceleration	Yes
<Axis name>.Config.DynamicDefaults.EmergencyDeceleration	<Axis name>.DynamicDefaults.EmergencyDeceleration	Yes
<Axis name>.Config.DynamicDefaults.Jerk	<Axis name>.DynamicDefaults.Jerk	Yes
<Axis name>.Config.DynamicDefaults.JerkActive	Not available The jerk is activated if the configured jerk is > 0.004 pulse/s^3.	No
<Axis name>.Config.DynamicLimits.MaxVelocity	<Axis name>.DynamicLimits.MaxVelocity	Yes
<Axis name>.Config.DynamicLimits.MinVelocity	<Axis name>.DynamicLimits.MinVelocity	Yes
<Axis name>.Config.General.LengthUnit	<Axis name>.Units.LengthUnit	Yes
<Axis name>.Config.Homing.AutoReversal	<Axis name>.Homing.AutoReversal	Yes
<Axis name>.Config.Homing.Direction	<Axis name>.Homing.ApproachDirection	Yes
<Axis name>.Config.Homing.FastVelocity	<Axis name>.Homing.ApproachVelocity	Yes
<Axis name>.Config.Homing.Offset	<Axis name>.Sensor[1].ActiveHoming.HomePositionOffset	Yes
<Axis name>.Config.Homing.SideActiveHoming	<Axis name>.Sensor[1].ActiveHoming.SideInput	Yes
<Axis name>.Config.Homing.SidePassiveHoming	<Axis name>.Sensor[1].PassiveHoming.SideInput	Yes
<Axis name>.Config.Homing.SlowVelocity	<Axis name>.Homing.ReferencingVelocity	Yes
<Axis name>.Config.Homing.SwitchedLevel	<Axis name>.Sensor[1].ActiveHoming.SwitchLevel <Axis name>.Sensor[1].PassiveHoming.SwitchLevel	No
<Axis name>.Config.Mechanics.InverseDirection	<Axis name>.Actor.InverseDirection	Yes
<Axis name>.Config.Mechanics.LeadScrew	<Axis name>.Mechanics.LeadScrew	Yes
<Axis name>.Config.Mechanics.PulsesPerDriveRevolution	<Axis name>.Actor.DriveParameter.PulsesPerDriveRevolution	Yes
<Axis name>.Config.PositionLimits_HW.Active	<Axis name>.PositionLimitsHW.Active	Yes
<Axis name>.Config.PositionLimits_HW.MaxSwitchedLevel	<Axis name>.PositionLimitsHW.MaxSwitchLevel	Yes
<Axis name>.Config.PositionLimits_HW.MinSwitchedLevel	<Axis name>.PositionLimitsHW.MinSwitchLevel	Yes

Tag name V1.0 to V3.0	Tag name V4.0	Automatic conversion V1..3 to V4
<Axis name>.Config.PositionLimits_SW.Active	<Axis name>.PositionLimitsSW.Active	Yes
<Axis name>.Config.PositionLimits_SW.MaxPosition	<Axis name>.PositionLimitsSW.MaxPosition	Yes
<Axis name>.Config.PositionLimits_SW.MinPosition	<Axis name>.PositionLimitsSW.MinPosition	Yes
Not available	<Axis name>.Actor.DirectionMode	No
Not available	<Axis name>.Actor.Type	No
Not available	<Axis name>.Sensor[1].ActiveHoming.Mode	No
Not available	<Axis name>.Sensor[1].PassiveHoming.Mode	No

ErrorBits tags (positioning axis)

Tag name V1.0 to V3.0	Tag name V4.0	Automatic conversion V1..3 to V4
<Axis name>.ErrorBits.HwLimitMax	<Axis name>.ErrorBits.HWLimit (Note the new status bits and the section Status of the limit switch (Page 10024).)	No
<Axis name>.ErrorBits.HwLimitMin		
<Axis name>.ErrorBits.SwLimitMaxExceeded	<Axis name>.ErrorBits.SWLimit (Note the new status bits and the section Status of the limit switch (Page 10024).)	No
<Axis name>.ErrorBits.SwLimitMaxReached		
<Axis name>.ErrorBits.SwLimitMinExceeded		
<Axis name>.ErrorBits.SwLimitMinReached		
Not available	<Axis name>.ErrorBits.DirectionFault	No

MotionStatus tags (positioning axis)

Tag name V1.0 to V3.0	Tag name V4.0	Automatic conversion V1..3 to V4
<Axis name>.MotionStatus.Distance	<Axis name>.StatusPositioning.Distance	Yes
<Axis name>.MotionStatus.Position	<Axis name>.Position	Yes
<Axis name>.MotionStatus.TargetPosition	<Axis name>.StatusPositioning.TargetPosition	Yes
<Axis name>.MotionStatus.Velocity	<Axis name>.Velocity	Yes

StatusBits tags (positioning axis)

Tag name V1.0 to V3.0	Tag name V4.0	Automatic conversion V1..3 to V4
<Axis name>.StatusBits.Homing	<Axis name>.StatusBits.HomingCommand	Yes
<Axis name>.StatusBits.SpeedCommand	<Axis name>.StatusBits.VelocityCommand	Yes
Not available	<Axis name>.StatusBits.HWLimitMaxActive	No
Not available	<Axis name>.StatusBits.HWLimitMinActive	No
Not available	<Axis name>.StatusBits.SWLimitMaxActive	No
Not available	<Axis name>.StatusBits.SWLimitMinActive	No

Tags (command table)

Tag name V1.0 to V3.0	Tag name V4.0	Automatic conversion V1..3 to V4
<Command table>.Config.Command[n].Position	<Command table>.Command[n].Position	Yes
<Command table>.Config.Command[n].Velocity	<Command table>.Command[n].Velocity	Yes
<Command table>.Config.Command[n].Duration	<Command table>.Command[n].Duration	Yes
<Command table>.Config.Command[n].NextStep	<Command table>.Command[n].NextStep	Yes
<Command table>.Config.Command[n].StepCode	<Command table>.Command[n].StepCode	Yes

See also

Overview of versions (Page 10018)

Changing a technology version (Page 10020)

Tags of the positioning axis technology object as of V4 (Page 10142)

Status of limit switch

The status and error bits for the display of the reached limit switch have been adapted in version V4.

In order to replicate the behavior of the error bits of versions V1...3, use the following logical operators:

V1...3	V4
<Axis name>.ErrorBits.HwLimitMin	<Axis name>.ErrorBits.HWLimit AND <Axis name>.StatusBits.HWLimitMinActive
<Axis name>.ErrorBits.HwLimitMax	<Axis name>.ErrorBits.HWLimit AND <Axis name>.StatusBits.HWLimitMaxActive
<Axis name>.ErrorBits.SwLimitMinReached	<Axis name>.ErrorBits.SWLimit AND (<Axis name>.Position = <Axis name>.PositioningLimits_SW.MinPosition)
<Axis name>.ErrorBits.SwLimitMinExceeded	<Axis name>.ErrorBits.SWLimit AND (<Axis name>.Position < <Axis name>.PositioningLimits_SW.MinPosition)
<Axis name>.ErrorBits.SwLimitMaxReached	<Axis name>.ErrorBits.SWLimit AND (<Axis name>.Position = <Axis name>.PositioningLimits_SW.MaxPosition)
<Axis name>.ErrorBits.SwLimitMaxExceeded	<Axis name>.ErrorBits.SWLimit AND (<Axis name>.Position > <Axis name>.PositioningLimits_SW.MaxPosition)

See also

Overview of versions (Page 10018)

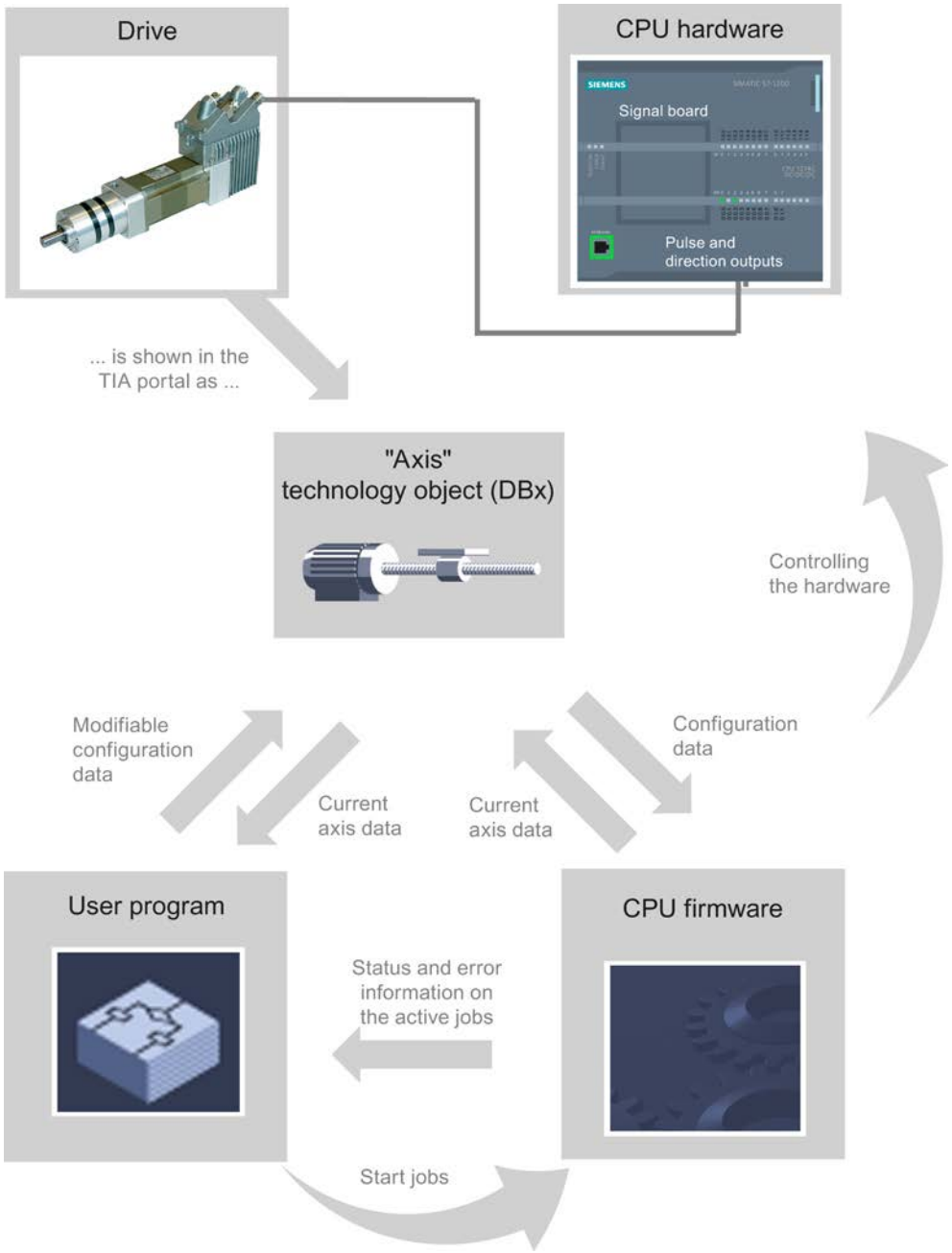
Changing a technology version (Page 10020)

Compatibility list of tags (Page 10021)

Positioning axis technology object

Integration of the positioning axis technology object

The following representation shows the relationships between the hardware and software components that are implemented when using the positioning axis technology object:



CPU hardware

The physical drive is controlled and monitored by the CPU hardware.

Drive

The drive represents the unit of power unit and motor. Stepper motors or servo motors with a pulse interface may be used.

Positioning axis technology object

The physical drive including mechanics is mapped in the TIA Portal as a positioning axis technology object. To do this, configure the positioning axis technology object with the following parameters:

- Selection of the PTOs (Pulse Train Output) to be used and configuration of the drive interface
- Parameter for mechanics and gear transmission of the drive (or the machine or system)
- Parameter for position monitoring, for dynamic parameters and for homing

The configuration of the positioning axis technology object is saved in the technology object (data block). This data block also forms the interface between the user program and the CPU firmware. The current axis data is saved in the data block of the technology object at the runtime of the user program.

User program

You start Motion Control instructions jobs in the CPU firmware with the user program. The following jobs for controlling the axis are possible:

- Enable and disable axis
- Position axis absolutely
- Position axis relatively
- Move axis with velocity set point
- Run axis commands as motion sequence (technology as of V2)
- Moving axes in jog mode
- Stop axis
- Reference axis; set reference point
- Change dynamic settings of axis
- Continuously read motion data of the axis
- Write tag of axis
- Acknowledge error

You determine the command parameters with the input parameters of the Motion Control instructions and the axis configuration. The output parameters of the instruction give you up to date information about the status and any errors of the command.

Before starting a command for the axis, you must enable the axis with the Motion Control instruction "MC_Power".

You can read out configuration data and current axis data with the tags of the technology object. You can change individual, changeable tags of the technology object (e.g. the current acceleration) from the user program.

You can also change the dynamic settings of the axis with the Motion Control instruction "MC_ChangeDynamic" and write additional configuration data with "MC_WriteParam". You can read the current motion status of the axis with the Motion Control instruction "MC_ReadParam".

CPU firmware

The motion control jobs started in the user program are processed in the CPU firmware. When using the axis control table, Motion Control jobs are triggered by operating the axis control table. The CPU firmware performs the following jobs depending on the configuration:

- Calculate the exact motion profile for motion jobs and emergency stop situations
- Control the drive enable and the pulse and direction signal
- Monitor the drive and the hardware and software limit switches
- Up to date feedback of status and error information to the Motion Control instructions in the user program
- Writing of current axis data into the data block of the technology object

See also

Tags of the positioning axis technology object as of V4 (Page 10142)

CPU outputs relevant for motion control (technology version V4) (Page 10008)

Relationship between the signal type and the direction of travel (Page 10011)

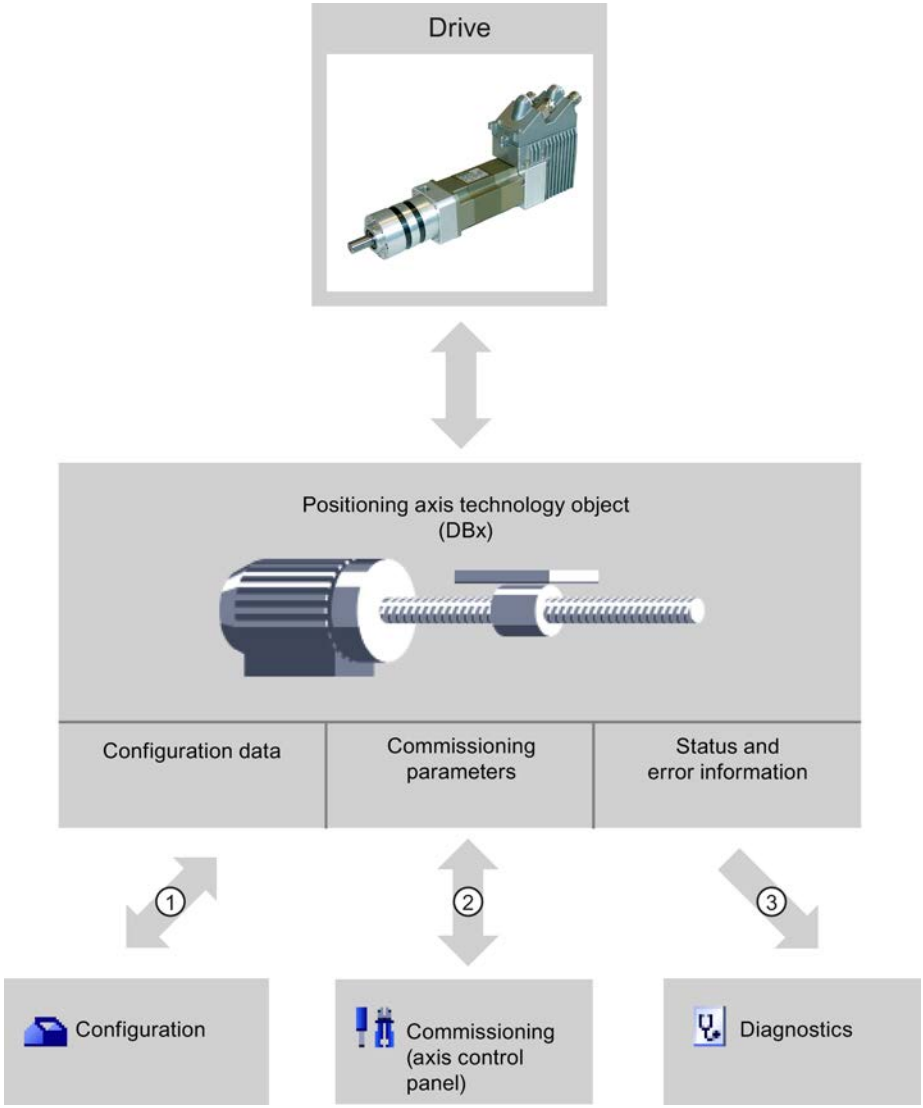
Tools of the positioning axis technology object (Page 10029)

Hardware and software limit switches (Page 10015)

Homing (Page 10017)

Tools of the positioning axis technology object

The TIA Portal provides the "Configuration", "Commissioning", and "Diagnostics" tools for the positioning axis technology object. The following representation shows the interaction of the three tools with the technology object and the drive:



①	Reading and writing of configuration data of the technology object;
②	Drive control via the technology object; Reading axis status for display on the control panel
③	Readout of the current status and error information of the technology object.

Configuration

Use the "Configuration" tool to configure the following properties of the positioning axis technology object:

- Selection of the PTO to be used and configuration of the drive interface
- Properties of the mechanics and the transmission ratio of the drive (or machine or system)
- Properties for position monitoring, dynamics, and homing

Save the configuration in the data block of the technology object.

Commissioning

Use the "Commissioning" tool to test the function of your axis without having to create a user program. When the tool is started, the axis control table will be displayed. The following commands are available on the axis control table:

- Enabling and disabling the axis
- Move axis in jog mode
- Position axis in absolute and relative terms
- Home axis
- Acknowledge errors

The dynamic values can be adjusted accordingly for the motion commands. The axis control table also shows the current axis status.

Diagnostics

Use the "Diagnostics" tool to keep track of the current status and error information for the axis and drive.

See also

CPU outputs relevant for motion control (technology version V4) (Page 10008)

Relationship between the signal type and the direction of travel (Page 10011)

Integration of the positioning axis technology object (Page 10026)

Hardware and software limit switches (Page 10015)

Homing (Page 10017)

Configuring the positioning axis technology object (Page 10031)

Commissioning the axis - Axis control panel (Page 10075)

Axis - Diagnostics (Page 10114)

Adding a positioning axis technology object

Requirements

A project with a CPU S7-1200 has been created.

Procedure

To add a positioning axis technology object in the project tree, follow these steps:

1. Open the "CPU > Technology objects" folder in the project tree.
2. Double-click the "Add new object" command.
   The "Add new object" dialog opens.
3. Select the "Motion Control" technology.
4. Open the "Motion Contro > S7-1200 Motion Control" folder.
5. Select the desired technology version in the "Version" column.
6. Select the "TO_PositioningAxis" object.
7. Enter the name of the axis in the "Name" input box.
8. To change the automatically assigned data block number, select the "Manual" option.
9. To display additional information about the technology object, click "Additional information".
10. Confirm your entry with "OK".

Result

The new technology object is created and saved to the "Technology objects" folder in the project tree.

See also

Guidelines on use of motion control (Page 10018)

Configuring the positioning axis technology object

Working with the configuration dialog

You configure the properties of the technology object in the configuration window. Proceed as follows to open the configuration window of the technology object:

1. Open the group of the required technology object in the project tree.
2. Double-click the "Configuration" object.

The configuration is divided into the following categories:

- **Basic parameters**
  The basic parameters contain all the parameters which must be configured for a functioning axis.
- **Extended parameters**
  The advanced parameters include parameters to adapt to your drive or your plant.

Configuration window icons

Icons in the area navigation of the configuration show additional details about the status of the configuration:

	The configuration contains default values and is complete. The configuration contains only default values. With these default values you can use the technology object without additional changes.
	The configuration contains values set by the user and is complete. All input boxess of the configuration contain valid values and at least one preset value has changed.
	The configuration is incomplete or incorrect At least one input box or drop-down list contains an invalid value. The corresponding box or the drop-down list is displayed on a red background. Click the roll-out error message to indicate the cause of error.
	The configuration is valid but contains warnings For example, only one hardware limit switch is configured. Depending on the plant, the lacking configuration of a hardware limit switch may result in a hazard. The corresponding box or the drop-down list is displayed on a yellow background.

See also

Guidelines on use of motion control (Page 10018)

Basic parameters (Page 10033)

Extended parameters (Page 10035)

Compare values

The "Compare values" function provides the following options:

- Comparison of configured start values of the project with the start values in the CPU and the actual values
- Direct editing of actual values and the start values of the project
- Immediate detection and display of input errors with suggested corrections
- Backup of actual values in the project
- Transfer of start values of the project to the CPU as actual values

Icons and operator controls

The following icons and operator controls are available:

Icon	Function
	Start value in CPU matches the configured Start value in the project
	Start value in CPU does not match the configured Start value in the project
	The comparison of the Start value in CPU with the configured Start value in the project cannot be performed
	Backs up the actual values in the project
	Transfers updated start values in the project to the CPU (initialize setting values)
	Opens the "Compare values" dialog

Basic parameters

Configuration - General (positioning axis technology object as of V4)

Configure the basic properties of the positioning axis technology object in the "General" configuration window.

Axis name

Define the name of the axis or the name of the positioning axis technology object in this box. The technology object is listed under this name in the project navigation.

Hardware interface

The pulses are output to the power unit of the drive by fixed assigned digital outputs.

In CPUs with relay outputs, the pulse signal cannot be output on these outputs because the relays do not support the necessary switching frequencies. A signal board with digital outputs must be used to enable you to work with the PTO (Pulse Train Output) on these CPUs.

Note

The PTO requires the functionality of a fast counter (HSC). An internal HSC is used for this, the count of which cannot be evaluated.

Selecting the pulse generator

In the drop-down list, select the PTO (Pulse Train Output) which is to provide the pulses for controlling the stepper motors or servo motors with pulse interface. If the pulse generators and high-speed counters are not used elsewhere in the device configuration, the hardware

interface can be configured automatically. In this case, the PTO selected in the drop-down list is displayed with a white background.

"Device configuration" button

This button takes you to the settings for the pulse options in the device configuration of the CPU.

Signal type

Select the desired signal type in the drop-down list. The following signal types are available:

- **PTO (pulse A and direction B)**
  A pulse output and a direction output are used for controlling the stepper motor.
- **PTO (count up A, count down B)**
  One pulse output each for motion in positive direction and negative direction is used for controlling the stepper motor.
- **PTO (A/B phase offset)**
  Both pulse outputs for Phase A and for Phase B run at the same frequency.
  The period of the pulse outputs is evaluated at the drive end as a step.
  The phase offset between Phase A and Phase B determines the direction of the motion.
- **PTO (A/B phase offset - quadruple)**
  Both pulse outputs for Phase A and for Phase B run at the same frequency.
  All positive edges and all negative edges of Phase A and Phase B are evaluated as a step at the drive end.
  The phase offset between Phase A and Phase B determines the direction of the motion.

Pulse output (signal type "PTO (pulse A and direction B)")

Select the desired output for the pulse output in this box.

You can select the output using a symbolic address or assign an absolute address.

Activate direction output (signal type "PTO (pulse A and direction B)")

You can deactivate and activate the direction output in "pulse and direction" mode. This option enables you to limit the motion direction.

Direction output (signal type "PTO (pulse A and direction B)")

Select the desired output for the direction output in this box.

You can select the output using a symbolic address or assign an absolute address.

Pulse output up (signal type "PTO (count up A, count down B)")

Select the desired pulse output for motions in positive direction in this box.

You can select the output using a symbolic address or assign an absolute address.

Pulse output down (signal type "PTO (count up A, count down B)")

Select the desired pulse output for motions in negative direction in this box.

You can select the output using a symbolic address or assign an absolute address.

Phase A (signal types "PTO (A/B phase offset)" and "PTO (A/B phase offset - quadruple)")

Select the desired pulse output for Phase A signals in this box.

You can select the output using a symbolic address or assign an absolute address.

Phase B (signal types "PTO (A/B phase offset)" and "PTO (A/B phase offset - quadruple)")

Select the desired pulse output for Phase B signals in this box.

You can select the output using a symbolic address or assign an absolute address.

User unit

Select the desired unit for the dimension system of the axis in the dropdown list. The selected unit is used for further configuration of the positioning axis technology object and for displaying the current axis data.

The values at the input parameters (Position, Distance, Velocity, ...) of the Motion Control instructions also refer to this unit.

Note

Later changing of the dimension system may not be converted correctly in all the configuration windows of the technology object. In this case check the configuration of all axis parameters.

The values of the input parameters of the Motion Control instructions may have to be adapted to the new unit of measurement in the user program.

See also

CPU outputs relevant for motion control (technology version V4) (Page 10008)

Relationship between the signal type and the direction of travel (Page 10011)

Configuration - General ("Axis" technology object V1...3) (Page 10166)

Extended parameters

Configuration - Drive interface

Configure the output for drive enable and the input for the "Drive ready" feedback signal of the drive in the "Drive signals" configuration window.

Drive enable is controlled by Motion Control instruction "MC_Power" and enables power to the drive. The signal is provided to the drive via the output to be configured.

The drive signals "Drive ready" to the CPU if it is ready to start executing travel after receipt of drive enable. The "Drive ready" signal is reported back to the CPU via the input to be configured.

If the drive does not have any interfaces of this type, you will not have to configure the parameters. In this case, select the value TRUE for the ready input.

See also

Configuration - Mechanics (Page 10036)

Position limits (Page 10037)

Dynamics (Page 10041)

Homing (positioning axis technology object as of V2) (Page 10048)

Configuration - Mechanics

Configure the mechanical properties of the drive in the "Mechanics" configuration window.

Increments per motor revolution

Configure the number of pulses required for one revolution of the motor in this box.

Limits (independent of the selected unit of measurement):

- 0 < Pulse per motor revolution ≤ 2147483647

Distance per revolution

In this box, configure the load distance per motor revolution covered by the mechanical system of your unit.

Limits (independent of the selected unit of measurement):

- 0.0 < Distance per revolution ≤ 1.0e12

Permitted direction of rotation (technology version as of V4)

Configure this box to determine whether the mechanics of your system are to move in both directions or only in the positive or negative direction.

If you have not activated the direction output of the pulse generator in the "PTO (pulse A and direction B)" mode, the selection is limited to the positive or negative direction.

Invert direction

You can use the "Invert direction" check box to adapt the control system to the direction logic of the drive.

The direction logic is inverted according to the selected mode of the pulse generator:

- PTO (pulse A and direction B)
  - 0 V at direction output ⇒ positive direction of rotation
  - 5 V/24 V at direction output ⇒ negative direction of rotation

  The specified voltage depends on the hardware used. The indicated values do not apply to the differential outputs of CPU 1217.
- PTO (count up A, count down B)
  The outputs "Pulse output down" and "Pulse output up" are swapped.
- PTO (A/B phase offset)
  The "Phase A" and "Phase B" outputs are swapped.
- "PTO (A/B phase offset - quadruple)
  The "Phase A" and "Phase B" outputs are swapped.

See also

Configuration - Drive interface (Page 10035)

Position limits (Page 10037)

Dynamics (Page 10041)

Homing (positioning axis technology object as of V2) (Page 10048)

Relationship between the signal type and the direction of travel (Page 10011)

Position limits

Requirements for hardware limit switches

Use only hardware limit switches that remain permanently switched after being approached. This switching status may only be revoked after a return to the valid travel range.

See also

Configuration - Position limits (Page 10037)

Behavior of axis when position limits is tripped (Page 10039)

Changing the position limits configuration in the user program (Page 10040)

Configuration - Position limits

Configure the hardware and software limit switches of the axis in the "Position limits" configuration window.

Enable hardware limit switch

Activate the function of the low and upper hardware limit switch with this check box. The hardware limit switches can be used for purposes of direction reversal during a homing procedure. For details, refer to the configuration description for homing.

Lower / upper HW limit switch input

Select the digital input for the low or upper hardware limit switch from the drop-down list. The input must be interrupt-capable. The digital onboard CPU inputs and the digital inputs of a plugged signal board can be selected as inputs for the HW limit switches.

Note

The digital inputs are set to a filter time of 6.4 ms by default. If these are used as hardware limit switches, undesired decelerations may occur. If this occurs, reduce the filter time for the relevant digital inputs.

The filter time can be set under "Input filter" in the device configuration of the digital inputs.

Active level

In the drop-down list, select the signal level available at the CPU when the hardware limit switch is approached.

- "Lower level" selected
  0 V (FALSE) at CPU input corresponds to hardware limit switch approached
- "High level" selected
  5 V / 24 V (TRUE) at the CPU input = hardware limit switch approached (the actual voltage depends on the hardware used)

Enable software limit switch

Activate the function of the low and upper software limit switch with this check box.

Note

The enabled software limit switch only affects a homed axis.

High and lower software limit switch

Enter the position value of the low and upper software limit switch in these boxes.

Limits (independent of the selected unit of measurement):

- -1.0e12 ≤ lower software limit switch ≤ 1.0e12
- -1.0e12 ≤ upper software limit switch ≤ 1.0e12

The value of the upper software limit switch must be greater than or equal to the value of the lower software limit switch.

See also

Requirements for hardware limit switches (Page 10037)

Behavior of axis when position limits is tripped (Page 10039)

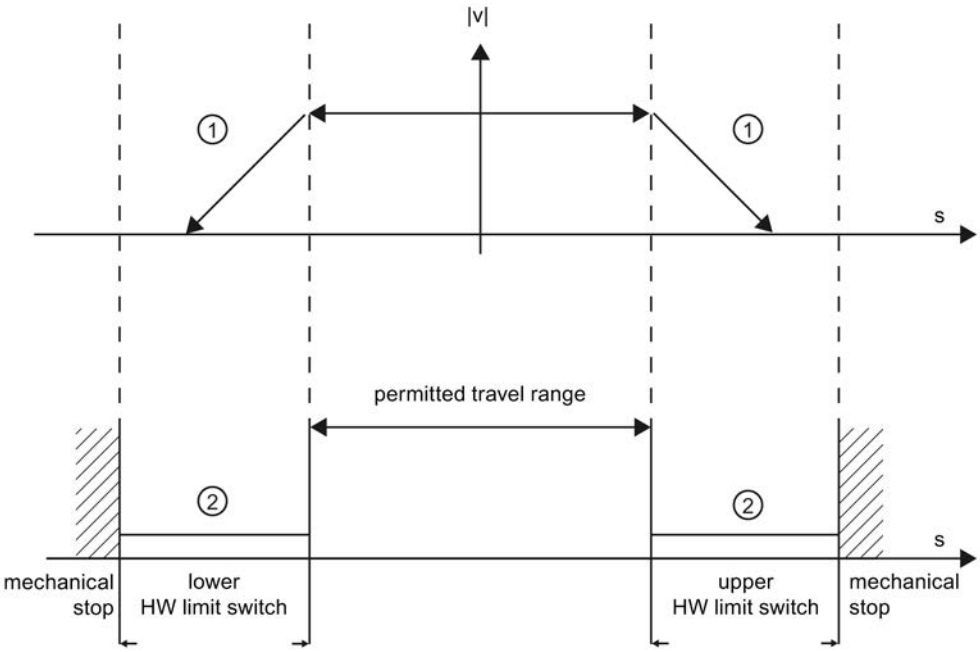
Changing the position limits configuration in the user program (Page 10040)

Configuration - Homing - Active (Page 10049)

Behavior of axis when position limits is tripped

Behavior of axis when hardware limit switches are approached

When the hardware limit switches are approached, the axis brakes to a standstill at the configured emergency stop deceleration. The specified emergency stop deceleration must be sufficient to reliably stop the axis before the mechanical stop. The following diagram presents the behavior of the axis after it approaches the hardware limit switches:



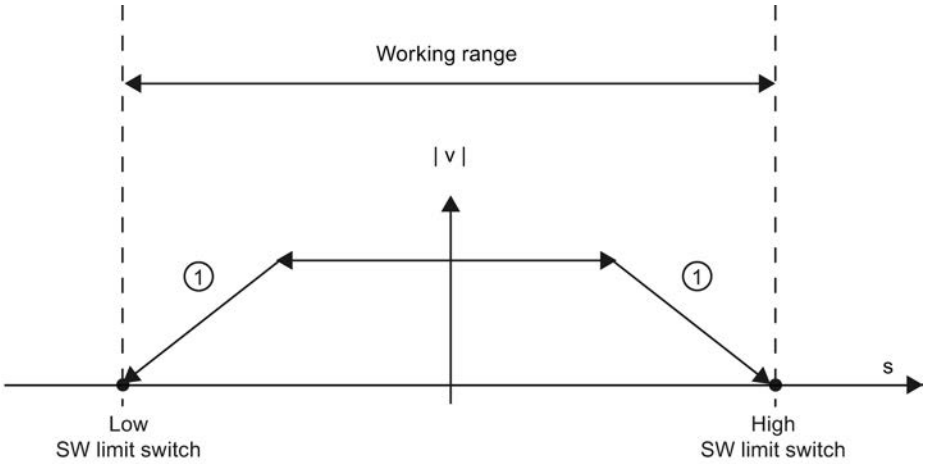
①	The axis brakes to a standstill at the configured emergency stop deceleration.
②	Range in which the HW limit switches signal the status "approached".

The "HW limit switch approached" error is displayed in the motion control instruction to be initiated, in "MC_Power", and in the technology object tags. Instructions for eliminating errors can be found in the Appendix under "List of ErrorIDs and ErrorInfos".

Behavior of axis when software limit switches are reached

If software limit switches are activated, an active motion is stopped at the position of the software limit switch. The axis is braked at the configured deceleration.

The following diagram presents the behavior of the axis until it reaches the software limit switches:



①	The axis brakes to a standstill at the configured deceleration.

The "SW limit switch approached" error is displayed in the initiating Motion Control instruction, at "MC_Power", and in the technology object tags. Instructions for eliminating errors can be found in the Appendix under "List of ErrorIDs and ErrorInfos".

You can learn about the circumstances that can lead to exceeding the software limit switches in the sections "Software limit switches in conjunction with a homing operation (Page 10125)" and "Software limit switches in conjunction with dynamic changes (Page 10130)".

Use additional hardware limit switches if a mechanical endstop is located after the software limit switches and there is a risk of mechanical damage.

See also

Requirements for hardware limit switches (Page 10037)

Configuration - Position limits (Page 10037)

Changing the position limits configuration in the user program (Page 10040)

Changing the position limits configuration in the user program

You can change the following configuration parameters during runtime of the user program in the CPU:

Hardware limit switches

You can also activate and deactivate the hardware limit switches during runtime of the user program. Use the following technology object tag for this purpose:

- <Axis name>.PositionLimitsHW.Active

Refer to the description of the technology object tags (Page 10142) in the appendix for information on when changes to the configuration parameter take effect.

Software limit switches

You can also activate and deactivate the software limit switches and change their position values during runtime of the user program. Use the following technology object tags for this purpose:

- <Axis name>.PositionLimitsSW.Active
  for activating and deactivating the software limit switches
- <Axis name>.PositionLimitsSW.MinPosition
  for changing the position of the lower software limit switch
- <Axis name>.PositionLimitsSW.MaxPosition
  for changing the position of the upper software limit switch

Refer to the description of technology object tags in the appendix for information on when changes to the configuration parameters take effect.

See also

Compatibility list of tags (Page 10021)

MC_ChangeDynamic: Change dynamic settings of axis V4 (Page 4851)

Requirements for hardware limit switches (Page 10037)

Configuration - Position limits (Page 10037)

Behavior of axis when position limits is tripped (Page 10039)

Dynamics

Configuration - General dynamics

Configure the maximum velocity, the start/stop velocity, the acceleration and deceleration, and the jerk limit (positioning axis technology object as of V2) of the axis in the "General dynamics" configuration window.

Velocity limiting unit

Select the unit of measurement with which you want to set the velocity limits in the dropdown list. The unit set here depends on the unit of measurement set under "Configuration - General" and serves only for easier input.

Maximum velocity / Start/stop velocity

Define the maximum permissible velocity and the start/stop velocity of the axis in these boxes. The start/stop velocity is the minimum permissible velocity of the axis.

Limit values:

The limits indicated below refer to the "Pulses/s" unit of measurement:

- Positioning axis technology object V4
  - 1 ≤ start/stop velocity ≤ 20000 (signal board 20 kHz)
    1 ≤ start/stop velocity ≤ 200000 (signal board 200 kHz)
    1 ≤ start/stop velocity ≤ 100000 (on-board CPU outputs 100 kHz)
    1 ≤ start/stop velocity ≤ 30000 (on-board CPU outputs 30 kHz)
    1 ≤ start/stop velocity ≤ 1000000 (on-board CPU outputs 1 MHz CPU 1217)
  - 1 ≤ maximum velocity ≤ 20000 (signal board 20 kHz)
    1 ≤ maximum velocity ≤ 200000 (signal board 200 kHz)
    1 ≤ maximum velocity ≤ 100000 (on-board CPU outputs 100 kHz)
    1 ≤ maximum velocity ≤ 30000 (on-board CPU outputs 30 kHz)
    1 ≤ maximum velocity ≤ 1000000 (on-board CPU outputs 1 MHz CPU 1217)

You can learn about the limits for the technology object positioning axis < V4 in the appendix Outputs of the CPU relevant for Motion Control (technology version V1...3) (Page 10162).

The value of the maximum velocity must be greater or equal to the value of the start/stop velocity.

The limits for other units of measurement must be converted by the user to conform to the given mechanics.

Acceleration / Delay - Ramp-up time / Ramp-down time

Set the desired acceleration in the "Ramp-up time" or "Acceleration" boxes. The desired deceleration can be set in the "Deceleration time" or "Deceleration" boxes.

The relation between the ramp-up time and acceleration and the deceleration time and deceleration is shown in the following equations:

$$\text{Rampup time} = \frac{\text{Maximum velocity} - \text{Start/stop velocity}}{\text{Acceleration}}$$

$$\text{Deceleration time} = \frac{\text{Maximum velocity} - \text{Start/stop velocity}}{\text{Deceleration}}$$

Motion jobs started in the user program are performed with the selected acceleration / deceleration.

The limits for acceleration and deceleration can be found in section Hardware components for motion control (Page 10006).

Note

Changes to the velocity limits ("start/stop velocity" and "maximum velocity") influence the acceleration and deceleration values of the axis. The ramp-up and deceleration times are retained.

Activate the jerk limit (positioning axis technology object as of V2)

Activate the jerk limit with this check box.

Note

If an error occurs, the axis decelerates with the configured emergency stop deceleration. An activated jerk limit is not considered here.

Smoothing time / jerk (positioning axis technology object as of V2)

You can input the parameters of the jerk limit in the "Smoothing time" box or alternatively in the "Jerk" box.

- Set the desired jerk for acceleration and deceleration ramp in the "Jerk" box.
- Set the desired smoothing time for the acceleration ramp in the "Rounding time" box.

Note

Smoothing time V2...3

The set smoothing time visible in the configuration only applies to the acceleration ramp.

If the values for acceleration and deceleration differ, the smoothing time of the deceleration ramp is calculated according to the jerk of the acceleration ramp and used. (See also Behavior of the axis when using the jerk limit (Page 10046)

The smoothing time of the deceleration is adapted as follows:

- **Acceleration > deceleration**
  The smoothing time used for the deceleration ramp is shorter than that for the acceleration ramp.
- **Acceleration < deceleration**
  The smoothing time used for the deceleration ramp is greater than that for the acceleration ramp.
- **Acceleration = deceleration**
  The smoothing times of the acceleration and deceleration ramp are equal.

The relation between smoothing times and jerk is shown in the following equation:

$$\text{Rounding off time (acceleration ramp)} = \frac{\text{Acceleration}}{\text{Jerk}}$$

$$\text{Rounding off time (deceleration ramp)} = \frac{\text{Deceleration}}{\text{Jerk}}$$

Motion jobs started in the user program are performed with the selected jerk.

The limits for the jerk can be found in section Hardware components for motion control (Page 10006).

See also

Behavior of the axis when using the jerk limit (Page 10046)

Hardware components for motion control (Page 10006)

Configuration - Dynamics emergency stop (Page 10044)

Changing the configuration of dynamics in the user program (Page 10047)

Configuration - Dynamics emergency stop

Configure the emergency stop deceleration of the axis in the "Dynamics emergency stop" configuration window. In the event of an error, and when disabling the axis, the axis is brought to a standstill with this deceleration using the Motion Control instruction "MC_Power" (input parameter StopMode = 0 or 2).

Velocity limits

The velocity values configured in the "General dynamics" configuration window are once again displayed in this information area.

Deceleration

Set the deceleration value for emergency stop in the "Emergency stop deceleration" or "Emergency stop ramp-down time" field.

The relation between emergency stop deceleration time and emergency stop deceleration is shown in the following equation:

$$\text{Emergency stop ramp-down} = \frac{\text{Maximum velocity} - \text{Start/stop velocity}}{\text{Emergency stop deceleration}}$$

The specified emergency stop deceleration must be sufficient to bring the axis to a standstill in a timely manner in the event of an emergency (for example, when the hardware limit switch is approached prior to reaching the mechanical endstop).

The configured maximum velocity of the axis must be used as a basis for selecting the emergency stop deceleration.

Limit values:

The limits indicated below refer to the "Pulses/s^2" unit of measurement:

- As of CPU firmware V3
  0.005 ≤ emergency stop deceleration ≤ 9.5E9
- CPU Firmware V1...2
  0.28 ≤ emergency stop deceleration ≤ 9.5E9

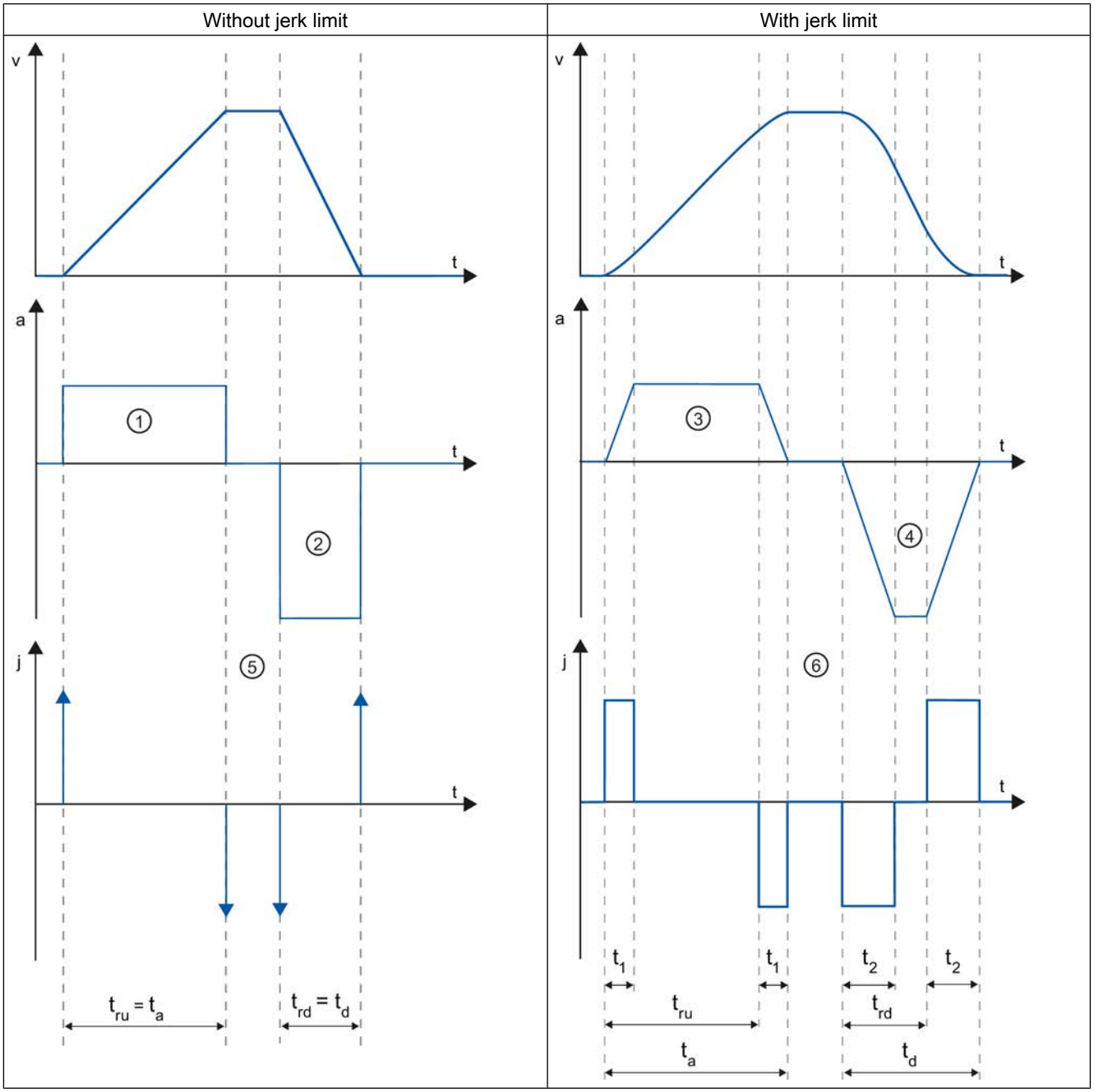
The limits for other units of measurement must be converted to conform to the given mechanics.

See also

Configuration - General dynamics (Page 10041)
Changing the configuration of dynamics in the user program (Page 10047)

Behavior of the axis when using the jerk limit

Axis acceleration and deceleration is not stopped abruptly when the jerk limit is activated; it is adjusted gently according to the set step or smoothing time. The diagram below details the behavior of the axis with and without activated jerk limit:



t	Time axis
v	Velocity
a	Acceleration

j	Jerk
t_{ru}	Rampup time
t_a	Time taken for the axis to accelerate
t_{rd}	Deceleration time
t_d	Time taken for the axis to decelerate
t_1	Smoothing time of the acceleration ramp
t_2	Smoothing time of the deceleration ramp

The example shows travel in which the deceleration value ② is twice the acceleration value ①. The resulting ramp-down time t_{rd} is therefore only half the length of the ramp-up time t_{ru}.

Acceleration ① and deceleration ② change abruptly without a jerk limit. Acceleration ① and deceleration ② change gradually with activated jerk limiter. As the jerk applies to entire motion, the rate is the same for the increase and decrease in acceleration and deceleration.

The step value j becomes infinitely high ⑤ as soon as the change is made without jerk limit. The step is limited to the configured value ⑥ when the jerk limit is activated.

The smoothing time t_1 given in the configuration applies to the acceleration ramp. The deceleration ramp smoothing time t_2 is calculated from the configured jerk value and the configured deceleration.

See also

Configuration - General dynamics (Page 10041)

Changing the configuration of dynamics in the user program

You can change the following configuration parameters during runtime of the user program in the CPU:

Acceleration and deceleration

You can also change the values for acceleration and deceleration during runtime of the user program. Use the following technology object tags for this purpose:

- <Axis name>.DynamicDefaults.Acceleration
  for changing acceleration
- <Axis name>.DynamicDefaults.Deceleration
  for changing deceleration

Refer to the description of the technology object tags (Page 10142) in the appendix for information on when changes to the configuration parameters take effect.

Emergency stop deceleration

You can also change the value for the emergency stop deceleration during runtime of the user program. Use the following technology object tag for this purpose:

- <Axis name>.DynamicDefaults.EmergencyDeceleration

Refer to the description of the technology object tags in the appendix for information on when changes to the configuration parameter take effect.

Note

After changes to this parameter, it may be necessary to adapt the positions of the hardware limit switches and other safety-relevant settings.

Jerk limit

You can also activate and deactivate the jerk limit at runtime of the user program and change the value for the jerk. To do this, use the technology object tag <Axis name>.DynamicDefaults.Jerk.

If you enter a value > 0.004 pulse/s^3 for the jerk, the jerk limit is activated with the specified value.

If you enter a value = 0.0 for the jerk, the jerk limit is deactivated.

Refer to the description of the technology object tags in the appendix for information on when changes to the configuration parameter take effect.

See also

Changing configuration of dynamic values in user program ("Axis" technology object V1...3) (Page 10168)

Compatibility list of tags (Page 10021)

MC_ChangeDynamic: Change dynamic settings of axis V4 (Page 4851)

Configuration - General dynamics (Page 10041)

Configuration - Dynamics emergency stop (Page 10044)

Homing (positioning axis technology object as of V2)

Configuration - Homing - Passive

Configure the necessary parameters for passive homing in the "Homing - Passive" configuration window.

The movement for passive homing must be triggered by the user (e.g. using an axis motion command). Passive homing is started using Motion Control instruction "MC_Home" with input parameter "Mode" = 2.

Homing switch input

Select the digital input for the homing switch from the drop-down list. The input must be interrupt-capable. The on-board CPU inputs and the inputs of an inserted signal board can be selected as inputs for the homing switch.

Note

The digital inputs are set to a filter time of 6.4 ms by default.

When the digital inputs are used as a homing switch, this can result in undesired decelerations and thus inaccuracies. Depending on the homing velocity and extent of the homing switch, the home position may not be detected. The filter time can be set under "Input filter" in the device configuration of the digital inputs.

The specified filter time must be less than the duration of the input signal at the homing switch.

Active level

In the drop-list, select the level of the homing switch that is to be used for homing.

Side of the homing switch

This is where you select whether the axis is to be homed on the low or high end of the homing switch.

Home position

The position configured in the Motion Control instruction "MC_Home" is used as the home position.

Note

If passive homing is carried out without an axis motion command (axis at a standstill), homing will be executed upon the next rising or falling edge at the homing switch.

Configuration - Homing - Active

Configure the necessary parameters for active homing in the "Active homing" configuration window. Active homing is started using Motion Control instruction "MC_Home" with input parameter "Mode" = 3.

Homing switch input

Select the digital input for the homing switch from the drop-down list. The input must be interrupt-capable. The on-board CPU inputs and the inputs of an inserted signal board can be selected as inputs for the homing switch.

Note

The digital inputs are set to a filter time of 6.4 ms by default.

When the digital inputs are used as a homing switch, this can result in undesired decelerations and thus inaccuracies. Depending on the homing velocity and extent of the homing switch, the home position may not be detected. The filter time can be set under "Input filter" in the device configuration of the digital inputs.

The specified filter time must be less than the duration of the input signal at the homing switch.

Active level

In the drop-list, select the level of the homing switch that is to be used for homing.

Allowing direction reversal at the hardware limit switch

Activate the check box to use the hardware limit switch as a reversing cam for the homing procedure. The hardware limit switches must be enabled for the reversal of direction (at least the hardware limit switch in the direction of approach must be configured).

If the hardware limit switch is reached during active homing, the axis brakes at the configured deceleration (not with the emergency stop deceleration) and reverses direction. The homing switch is then sensed in reverse direction.

If the direction reversal is not active and the axis reaches the hardware limit switch during active homing, the homing procedure is aborted with an error and the axis is braked at the emergency stop deceleration.

Note

If possible, use one of the following measures to ensure that the machine does not travel to a mechanical endstop in the event of a direction reversal:

- Keep the approach velocity low.
- Increase the configured acceleration/deceleration.
- Increase the distance between the hardware limit switch and the mechanical endstop.

Approach/homing direction

With the direction selection, you determine the approach direction used during active homing to search for the homing switch, as well as the homing direction. The homing direction specifies the travel direction the axis uses to approach the configured end of the homing switch to carry out the homing operation.

Side of the homing switch

This is where you select whether the axis is to be homed on the low or high end of the homing switch.

Approach velocity

In this box, specify the velocity at which the homing switch is to be searched for during the homing procedure.

Limits (independent of the selected unit of measurement):

- Start/stop velocity ≤ approach velocity ≤ maximum velocity

Homing velocity

In this box, specify the velocity at which the homing switch is to be approached for homing.

Limits (independent of the selected unit of measurement):

- Start/stop velocity ≤ Homing velocity ≤ Maximum velocity

Home position offset

If the desired home position deviates from the position of the homing switch, the home position offset can be specified in this box.

If the value does not equal 0, the axis executes the following actions following homing at the homing switch:

1. Move the axis at the homing velocity by the value of the home position offset
2. Upon reaching the "home position offset", the axis is at the home position that was specified in input parameter "Position" of the "MC_Home" Motion Control instruction.

Limits (independent of the selected unit of measurement):

- -1.0e12 ≤ home position offset ≤ 1.0e12

Home position

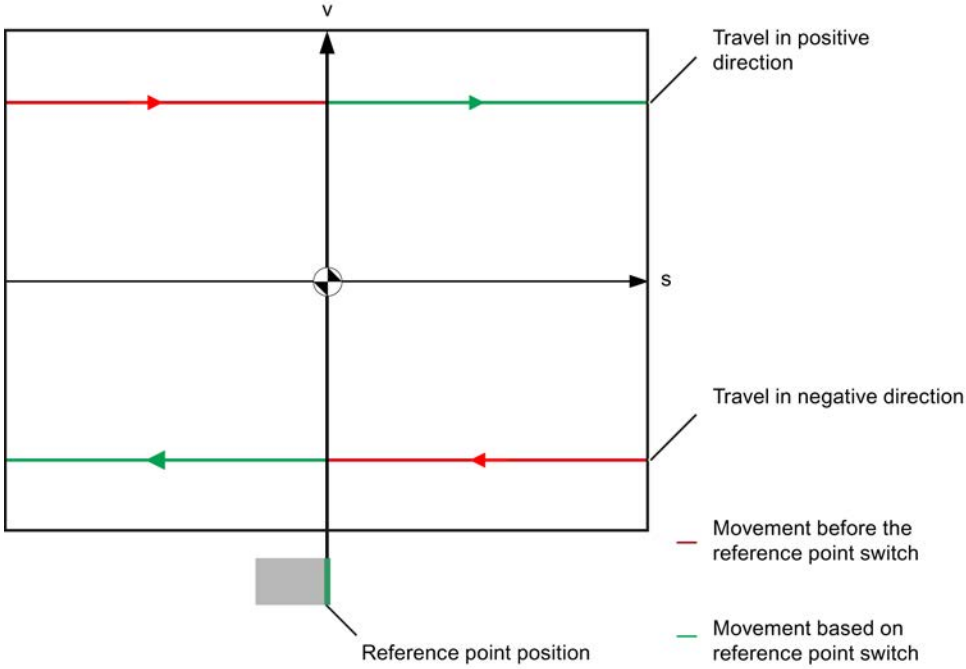
The position configured in the Motion Control instruction "MC_Home" is used as the home position.

Sequence - Passive homing

Passive homing is started with Motion Control instruction "MC_Home" (input parameter Mode = 2). Input parameter "Position" specifies the absolute reference point position.

The diagram below shows an example of a characteristic curve for passive homing with the following configuration parameters:

- "Reference point switch side" = "High side"



Movement towards reference point switch (red section of curve)

The Motion Control instruction "MC_Home" does not itself carry out any homing motion when passive homing is started. The travel required for reaching the reference point switch must be implemented by the user via other motion control instructions such as "MC_MoveRelative". The tag <axis name>.StatusBits.HomingDone remains TRUE during passive homing if the axis has already been homed.

Axis homing (transition from red to green section of curve)

The axis is homed when the configured side of the reference point switch is reached. The current position of the axis is set to the reference point position. This is specified at the "Position" parameter of the "MC_Home" Motion Control instruction. The variable <axis name>.StatusBits.HomingDone will be set to "TRUE" if the axis has not been been homed before. The travel previously started is not cancelled.

Movement beyond reference point switch (green section of curve)

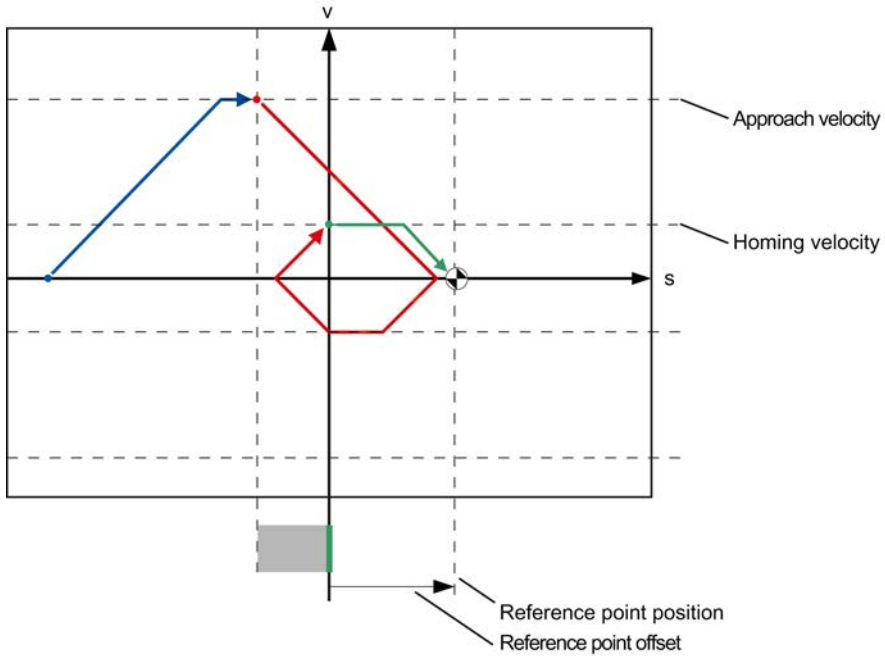
Following homing at the reference point switch, the axis continues and completes the previously started travel with the corrected axis position.

Sequence - Active homing

You start active homing with motion control instruction "MC_Home" (input parameter Mode = 3). The "Position" input parameter specifies the absolute home position. Alternatively, you can start active homing on the axis command table for test purposes.

The diagram below shows an example of a characteristic curve for an active reference point approach with the following configuration parameters:

- "Approach/homing direction" = "Positive direction"
- "Side of the homing switch" = "Top side"
- Value of "home position offset" > 0



Search for homing switch (blue curve section)

When active homing starts, the axis accelerates to the configured "approach velocity" and searches at this velocity for the homing switch. The tag <axis name>.StatusBits.HomingDone is set to FALSE.

Reference point approach (red curve section)

When the homing switch is detected, the axis in this example brakes and reverses, to be homed to the configured side of the homing switch at the configured homing velocity. Homing causes the tag <axis name>.StatusBits.HomingDone to change to TRUE.

Travel to home position offset (green curve segment)

After homing, the axis moves at the homing velocity along the path to the home position offset. There the axis is at the homing point position that was specified in input parameter "Position" of the "MC_Home" Motion Control instruction.

See also

Configuration - Homing - General (Axis technology object V2...3) (Page 10172)

Changing the homing configuration in the user program

With positioning axis technology object as of V2, you can change the following configuration parameters during runtime of the user program in the CPU:

Passive homing

You can change the end of the homing switch for passive homing during the user program runtime. Use the following technology object tag for this purpose:

- <Axis name>.Sensor[1].PassiveHoming.SideInput
  for changing the end of the homing switch

Refer to the description of the technology object tags (Page 10142) in the appendix for information on when changes to the configuration parameter take effect.

Active homing

You can change the direction of approach, the end of the homing switch, the approach velocity, the homing velocity, and the home position offset for active homing during the program runtime of the user program. Use the following technology object tags for this purpose:

- <Axis name>.Homing.AutoReversal
  for changing the auto reverse at the HW limit switch
- <Axis name>.Homing.ApproachDirection
  for changing the approach/homing direction
- <Axis name>.Sensor[1].ActiveHoming.SideInput
  for changing the end of the homing switch
- <Axis name>.Homing.ApproachVelocity
  for changing the approach velocity
- <Axis name>.Homing.ReferencingVelocity
  for changing the homing velocity
- <Axis name>.Sensor[1].ActiveHoming.HomePositionOffset
  for changing the home position offset

Refer to the description of the technology object tags in the appendix for information on when changes to the configuration parameter take effect.

See also

Compatibility list of tags (Page 10021)

MC_ChangeDynamic: Change dynamic settings of axis V4 (Page 4851)

Technology object command table

Use of the command table technology object

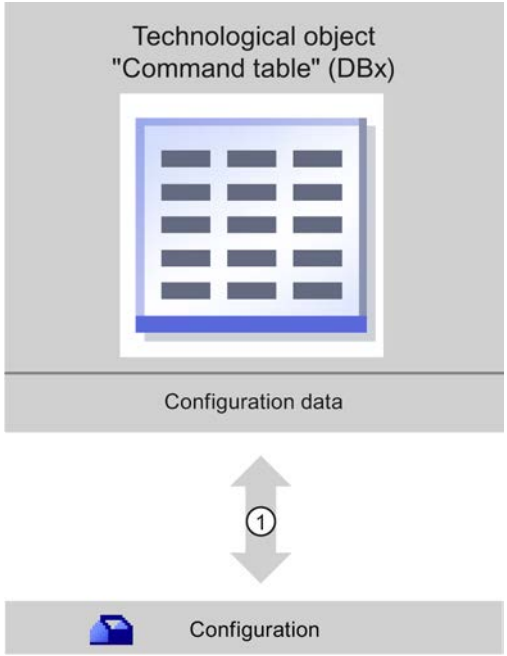
The Motion Control instruction "Command table" allows you to combine multiple individual axis control commands in one movement sequence. The technology object can be used with technology version as of V2.

You configure the movement sequence as a table in a configuration dialog.

The motion profile of the movement sequences can be checked on a graph before the project is loaded to the CPU. The command tables created are then linked to an axis and used in the user program with the "MC_CommandTable" Motion Control instruction. You can process part or all of the command table.

Command table technology object tools

The "Configuration" tool is provided in the TIA Portal for the "Command Table" technology object. The representation below shows the interaction of the tool with the technology object:



①	Writing and reading the configuration of the technology object

Configuration

Configure the following properties of the "Command Table" technology object with the "Configuration" tool:

- You can create one or more movement sequences by configuring individual jobs.
- You can configure the graphic display to check your movement sequence using an axis already configured or a configurable default axis.

The movement sequence data are saved in the data block of the technology object.

Adding the technological object command table

Requirements

- A project with a CPU S7-1200 has been created.
- The CPU firmware version is V2.1 or higher

Procedure

Proceed as follows to add a "Command table" technology object in the project tree:

1. Open the "CPU > Technology objects" folder in the project tree.
2. Double-click the "Add new object" command.
   The "Add new object" dialog opens.
3. Select the "Motion Control" technology.
4. Open the "Motion Contro > S7-1200 Motion Control" folder.
5. Select the desired technology version in the "Version" column.
6. Select the "TO_CommandTable" object.
7. Enter the name of the command table in the "Name" input box.
8. To change the automatically assigned data block number, select the "Manual" option.
9. To display additional information about the technology object, click "Additional information".
10. Confirm your entry with "OK".

Result

The new technology object is created and saved to the "Technology objects" folder in the project tree.

Configuring the command table technology object

Working with the configuration dialog

You configure the properties of the technology object in the configuration window. Proceed as follows to open the configuration window of the technology object:

1. Open the group of the required technology object in the project tree.
2. Double-click the "Configuration" object.

The configuration is divided into the following categories:

- **Basic parameters**
  The basic parameters contain all parameters which must be configured for a functional command table.
- **Extended parameters**
  The extended parameters contain the parameters of the default axis or display the parameter values of the axis selected.

Configuration window icons

Icons in the area navigation of the configuration show additional details about the status of the configuration:

	The configuration contains default values and is complete. The configuration contains only default values. With these default values, you can use the technology object without additional changes.
	The configuration contains values set by the user and is complete. All input fields of the configuration contain valid values and at least one preset value has changed.
	The configuration is incomplete or incorrect At least one input field or drop-down list contains an invalid value. The corresponding field or the drop-down list is displayed on a red background. Click the roll-out error message to display the cause of the error.
	The configuration contains mutually incompatible parameter values The configuration contains parameter values that contradict each other either in size or logic. The corresponding field or the drop-down list is displayed on a yellow background.

See also

Guidelines on use of motion control (Page 10018)

Basic parameters (Page 10058)

Extended parameters (Page 10071)

Compare values

The "Compare values" function provides the following options:

- Comparison of configured start values of the project with the start values in the CPU and the actual values
- Direct editing of actual values and the start values of the project
- Immediate detection and display of input errors with suggested corrections
- Backup of actual values in the project
- Transfer of start values of the project to the CPU as actual values

Icons and operator controls

The following icons and operator controls are available:

Icon	Function
	Start value in CPU matches the configured Start value in the project
	Start value in CPU does not match the configured Start value in the project
	The comparison of the Start value in CPU with the configured Start value in the project cannot be performed
	Backs up the actual values in the project
	Transfers updated start values in the project to the CPU (initialize setting values)
	Opens the "Compare values" dialog

Basic parameters

Configuration - General

Configure the name of the technology object in the "General" configuration window.

Name

Define the name of the command table or the name of the "Command table" technology object in this field. The technology object will be listed under this name in the project tree.

See also

Configuration - Command table (Page 10059)

Shortcut menu commands - Command table (Page 10062)

Working with the trend diagram (Page 10063)

Shortcut menu commands - Curve chart (Page 10067)

Transition from "Complete command" to "Blend motion" (Page 10068)

Changing the command table configuration in the user program (Page 10070)

Configuration - Command table

Create the desired movement sequence in the "Command Table" configuration window and check the result against the graphic view in the trend diagram.

Note

Small deviations are possible between the time behavior and position in the trend shown and the real movement of the axis. Movements in response to software limit switches being reached are not shown.

Activate warnings

Activate the display of warnings in the command table with this checkbox.

Use axis parameters of

From the drop-down list, select which axis parameters are to be used for selecting the graphic view of and checking the movement sequence. Select "Default axis" if you have yet to add an axis to the "Technology object" folder or wish to use value which have not been configured in any of the available axes. You configure the properties of the default axis under "Advanced parameters".

The axis parameters of the axis selected at the "Axis" parameter are used to process the command table in the user program.

Column: Step

Shows the step number of the command.

Column: Command type

In this column, select the command types which are to be used for processing the command table. Up to 32 commands can be entered. The commands will be processed in sequence. You can choose between the following entries and command types:

- **Empty**
  The entry serves as a placeholder for any commands to be added. The empty entry is ignored when the command table is processed.
- **Halt**
  Stop axis
  (the command only takes effect after a "Velocity set point" command)
- **Positioning Relative**
  Position axis relatively

- **Positioning Absolute**
  Position axis absolutely
- **Velocity set point**
  Move axis at set velocity
- **Wait**
  Waits until the given period is over. Wait does not stop active traversing.
- **Separator**
  Adds a Separator line above the selected line. The Separator line acts as a range limit for the graphic display of the trend view.
  Use the Separator lines if you wish to process parts of the command table.

Column: Position

Enter the position or travel path for the selected command in this column:

- **Command "Positioning Relative"**
  The command will move the axis by the the given travel path.
- **Command "Positioning Absolute"**
  The command will move the axis by the the given position.
- **Separator**
  The value given specifies the start position for the graphic display.

Limit values (independent of the selected user unit):

- -1.0e12 ≤ position / distance ≤ -1.0e-12
- 1.0e-12 ≤ position / distance ≤ 1.0e12
- Position / travel path = 0.0

Column: Velocity

In this column, you enter the velocity for the selected command:

- **Command "Positioning Relative"**
  The command will move the axis at the the given velocity.
  The given velocity will not be reached if the travel path selected is not large enough.
- **Command "Positioning Absolute"**
  The command will move the axis at the the given velocity.
  The given velocity will not be reached if the target position is too close to the starting position.
- **Command " Velocity set point"**
  The command will move the axis at the the given velocity.
  The given velocity will not be reached during the command if too short a runtime is selected.

Limit values (independent of the selected user unit):

- For the commands: "Positioning Relative" and "Positioning Absolute"
  - 1.0e-12 ≤ velocity ≤ 1.0e12
- For the command: "Velocity set point"
  - -1.0e12 ≤ velocity ≤ -1.0e-12
  - 1.0e-12 ≤ velocity ≤ 1.0e12
  - Velocity = 0.0

Column: Duration

Enter the duration of the selected command in this column:

- **Command " Velocity set point"**
  The command will move the axis for the specified duration. The duration includes both the acceleration phase and the constant travel phase. The next command will be processed once the duration is over.
- **Command "Wait"**
  Waits until the given duration is over.

Limit values (independent of the selected user unit):

- 0.001s ≤ duration ≤ 64800s

Column: Next step

Select the mode of transition to the next step from the drop-down list:

- **Complete command**
  The command will be completed. The next command will be processed immediately.
- **Blend motion**
  The motion of the current command will be blended with the motion of the following command. The transition mode "Blend motion" is available with command types "Positioning Relative" and "Positioning Absolute".
  Motion will be blended with motions of the following command types:
  - Positioning Relative
  - Positioning Absolute
  - Velocity set point

No blending occurs with other command types.

For the exact behavior of the axis when a command is appended or overlapped, see: Transition from "Complete command" to "Blend motion" (Page 10068)

Column: Step code

Enter a numerical value / bit pattern in this column which is to be output at the "StepCode" output parameter of the "MC_CommandTable" Motion Control instruction while the command is being processed.

Limit values:

- 0 ≤ code number ≤ 65535

See also

Configuration - General (Page 10058)

Shortcut menu commands - Command table (Page 10062)

Working with the trend diagram (Page 10063)

Shortcut menu commands - Curve chart (Page 10067)

Transition from "Complete command" to "Blend motion" (Page 10068)

Changing the command table configuration in the user program (Page 10070)

Shortcut menu commands - Command table

The following shortcut menu commands are available in the command table:

Insert empty line

Adds an empty line above the selected line.

This shortcut menu command can only be executed if there are enough empty lines at the end of the command table.

Add empty line

Adds an empty line below the selected line.

This shortcut menu command can only be executed if there are enough empty lines at the end of the command table.

Insert separator line

Adds a separator line above the selected line.

You cannot have two consecutive separator lines.

Add separator line

Adds a separator line below the selected line.

You cannot have two consecutive separator lines, nor can you add a separator line at the end of the command table.

Cut

Removes the selected lines or content of the selected cell and saves them/it in the clipboard.

Selected lines will be deleted and the subsequent lines of the command table shifted up.

Copy

Copies the selected lines or content of the selected cell and saves them/it in the clipboard.

Paste

- Selected lines:
  Pastes the lines from the clipboard into the table above the selected line.
- Selected cell:
  Pastes the content of the clipboard into the selected line.

This shortcut menu command can only be executed if there are enough empty lines at the end of the command table.

Replace

Replaces the selected lines with the lines in the clipboard.

Delete

Deletes the selected lines. The lines below in the command table shift up.

See also

Configuration - General (Page 10058)

Configuration - Command table (Page 10059)

Working with the trend diagram (Page 10063)

Shortcut menu commands - Curve chart (Page 10067)

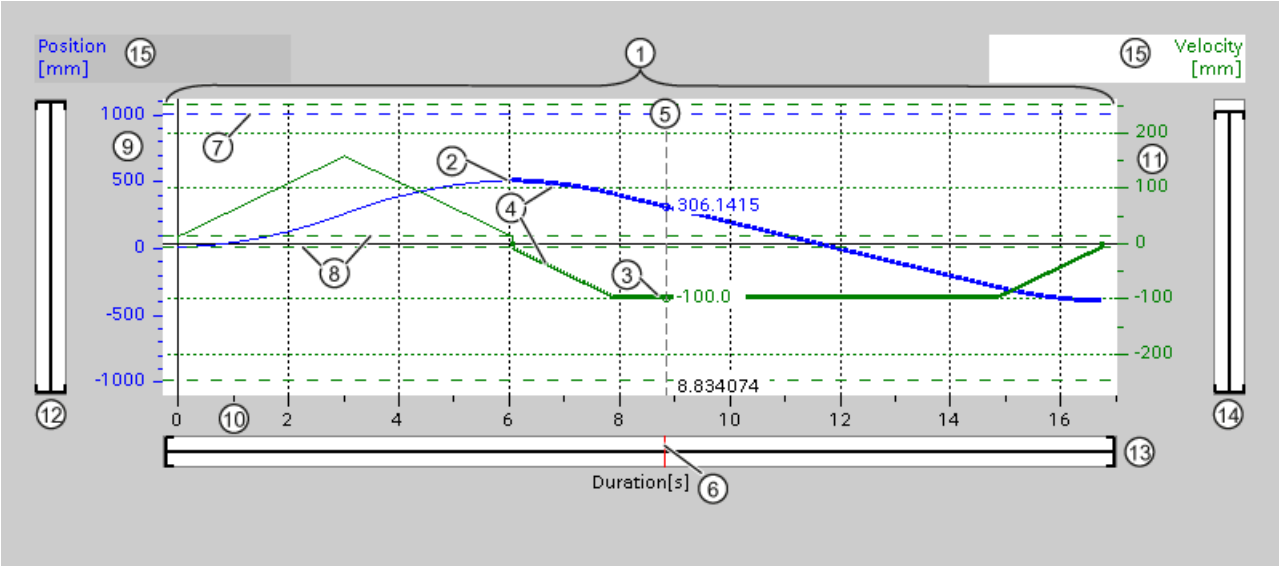
Transition from "Complete command" to "Blend motion" (Page 10068)

Changing the command table configuration in the user program (Page 10070)

Working with the trend diagram

The following tools and information are available in the trend view.

Trend view and components



①	Trend view
②	Position curve
③	Velocity curve
④	Curve section of a selected command
⑤	Ruler
⑥	Ruler position marking
⑦	Software limit switch position
⑧	Start/stop velocity
⑨	Position axis scale range
⑩	Time axis scale range
⑪	Velocity axis scale range
⑫	Scroll bar, position axis
⑬	Scroll bar time axis
⑭	Scroll bar, velocity axis
⑮	Selecting the grid

Selecting separator sections

If the command table consists of multiple sections separated by separators, you can select these sections in the trend view by selecting a command in the section.

Selecting commands

Commands can be selected in the trend view and in the command table:

- Click on a point on the velocity or position curve in the trend view. The corresponding command will be highlighted in the command table.
- Select a command in the command table.
  The corresponding section of curve will be highlighted.

Selecting the visible range of the trend view

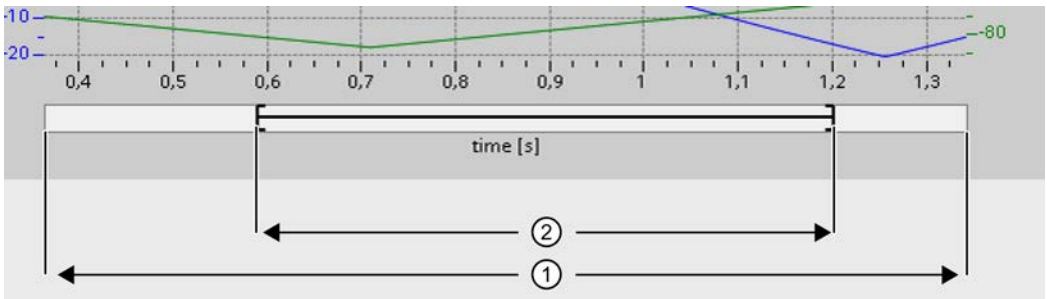
Follow the steps below to adjust the section of the trend view to be displayed:

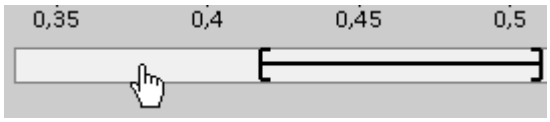
Select the scaling in the shortcut menu:

- Scale to curves:
  Scales the axes so the position and velocity curves are visible.
- Scale to curves and limits:
  Scales the axes so the position and velocity curves, the positions of the activated software limit switches and the minimum and maximum velocity limits are visible.

The view selected will be marked in the shortcut menu with a tick.

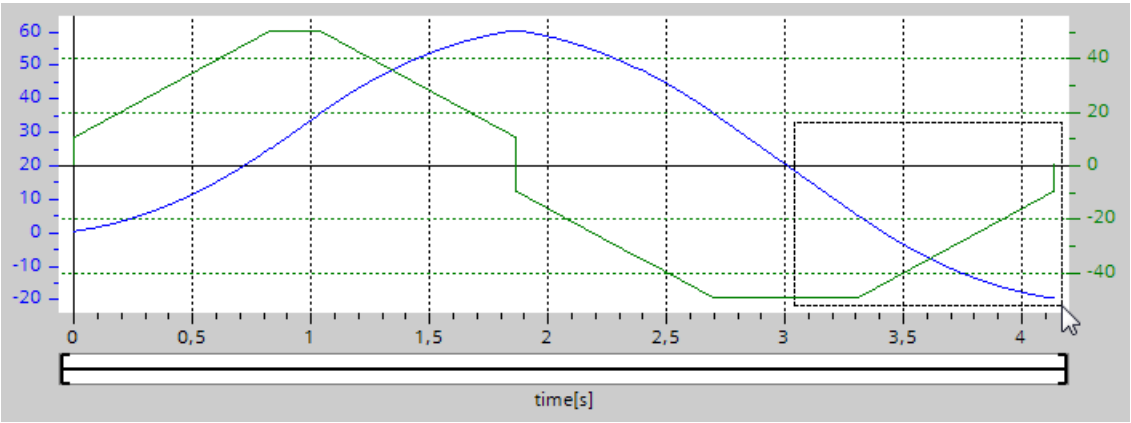
Selecting the section to be shown within the range:



①	Range which the curve values and / or limits are within. (see Selecting in the shortcut menu)
②	Selected range to be shown in the trend window. You set the range with the margin cursor at the right-hand and left-hand margin. You set the position within range ① with the drag cursor. You can also define the position by clicking in range ①. 

Selecting the section to be shown with the mouse:

Drag a section of the trend view by clicking and dragging with the mouse. The section of curve selected will be enlarged once you release the mouse.



Undoing the last change to the section:

Select the shortcut command "Undo zoom" to undo the last change to the section.

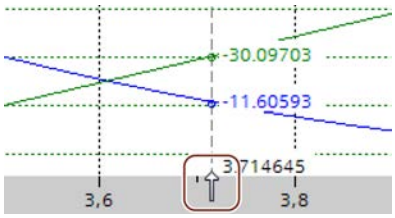
Synchronizing the grid

Click on the axis scales to select whether the grid is to be synchronized with the position axis or velocity axis.

Reading off curve values from the ruler

Activate the ruler using the shortcut menu command "Show ruler".

You can move the ruler to any point on the curves using the ruler cursor.



See also

Configuration - General (Page 10058)

Configuration - Command table (Page 10059)

Shortcut menu commands - Command table (Page 10062)

Shortcut menu commands - Curve chart (Page 10067)

Transition from "Complete command" to "Blend motion" (Page 10068)

Changing the command table configuration in the user program (Page 10070)

Shortcut menu commands - Curve chart

The following shortcut menu commands are available in the curve window:

Zoom 100%

Selects a zoom factor which will show 100% of the curve values and / or limits.

Undo zoom

Undoes the last zoom change.

Scale to curves

Scales the axes so the position and velocity curves are visible.

Scale to curves and limits

Scales the axes so the position and velocity curves, the positions of the activated software limit switches and the minimum and maximum velocity limits are visible.

Show velocity limits

Shows the lines of the velocity limits.

Show software limit switches

Shows the lines of the software limit switches.

Show ruler

Fades the ruler in / out

Use the ruler when you want to see the individual values of the curves.

See also

Configuration - General (Page 10058)

Configuration - Command table (Page 10059)

Shortcut menu commands - Command table (Page 10062)

Working with the trend diagram (Page 10063)

Transition from "Complete command" to "Blend motion" (Page 10068)

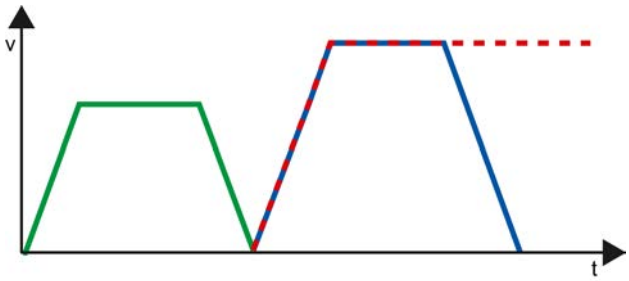
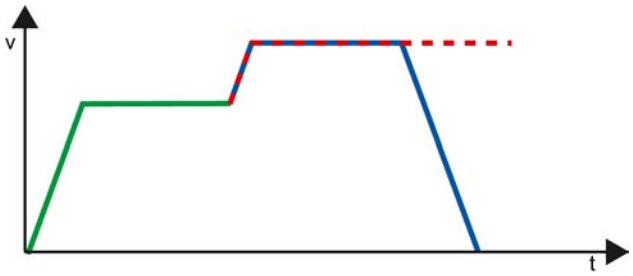
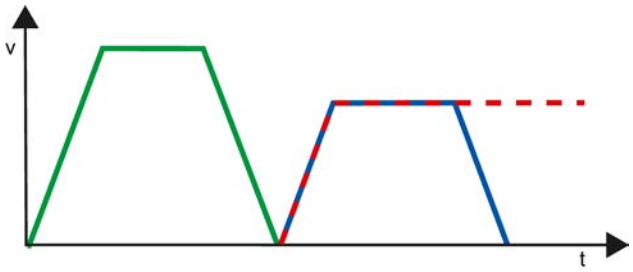
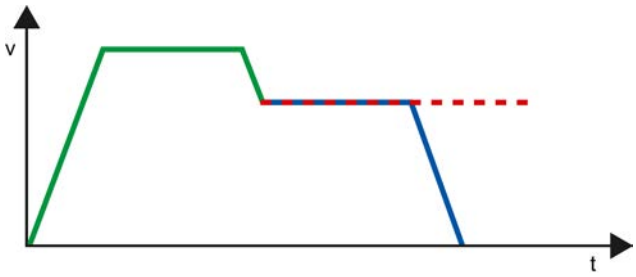
Changing the command table configuration in the user program (Page 10070)

Transition from "Complete command" to "Blend motion"

The charts below show the transition between movements in various different transition modes in the "Next step" column:

Motion transition with preceding positioning jobs

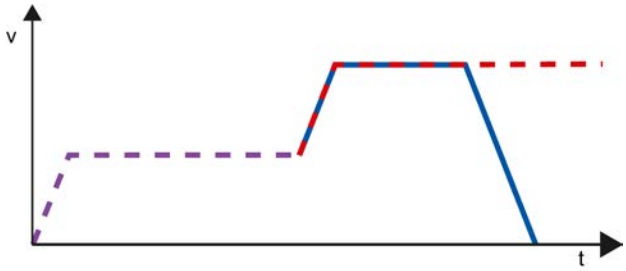
The following diagrams show a command sequence with two motion tasks. The first command is for positioning (green). The second command is for velocity (red) or positioning (blue):

Complete job	Blend motion
Transition from lower to higher velocity  A job with high velocity is appended to a previous positioning job. The positioning job terminates at its target position at velocity "0". The second job starts from standstill.	Transition from lower to higher velocity  A job with high velocity is overlapped with a previous positioning job. The first positioning job terminates without standstill at its target position. The second job starts with the new velocity.
Transition from higher to lower velocity  A job with low velocity is appended to a previous positioning job. The positioning job terminates at its target position at velocity "0". The second job starts from standstill.	Transition from higher to lower velocity  A job with low velocity is overlapped with a previous positioning job. The first positioning job terminates without standstill at its target position. The first job starts with the new velocity.

	1. Job "Positioning Relative" or "Positioning Absolute"
	2. Job "Velocity set point"
	2. Job "Positioning Relative" or "Positioning Absolute"

Motion transition with preceding velocity jobs

The following diagrams show a command sequence with two motion tasks. The first command is for velocity (violet). The second command is for velocity (red) or positioning (blue):

Transition from lower to higher velocity	Transition from higher to lower velocity
	
A job with a high velocity is appended to a previous velocity job. The first velocity job ends after the defined runtime. The second job starts with the new velocity.	A command with lower velocity is blended with a previous velocity command. The second job starts with the new velocity.

	1. Job "Velocity set point"
	2. Job "Velocity set point"
	2. Job "Positioning Relative" or "Positioning Absolute"

See also

Configuration - General (Page 10058)

Configuration - Command table (Page 10059)

Shortcut menu commands - Command table (Page 10062)

Working with the trend diagram (Page 10063)

Shortcut menu commands - Curve chart (Page 10067)

Changing the command table configuration in the user program (Page 10070)

Changing the command table configuration in the user program

You can change the following configuration parameters during runtime of the user program in the CPU:

Commands and corresponding values

You can also change the parameters of the command table during the runtime of the user program. Use the following technology object tags for this purpose:

- <Table name>.Command[1..32].Type
  for changing the command type
- <Table name>.Command[1..32].Position
  for changing the position/travel distance

- <Table name>.Command[1..32].Velocity
  for changing the velocity
- <Table name>.Command[1..32].Duration
  for changing the duration
- <Table name>.Command[1..32].NextStep
  for changing the parameter "Next step"
- <Table name>.Command[1..32].StepCode
  for changing the step code

Refer to the description of the technology object tags (Page 10161) in the appendix for information on when changes to the configuration parameters take effect.

See also

Compatibility list of tags (Page 10021)

Configuration - General (Page 10058)

Configuration - Command table (Page 10059)

Shortcut menu commands - Command table (Page 10062)

Working with the trend diagram (Page 10063)

Shortcut menu commands - Curve chart (Page 10067)

Transition from "Complete command" to "Blend motion" (Page 10068)

Extended parameters

Chart parameters

Configuration - General

Configure the basic properties of the chart view of the "Command table" technology object in the "General" configuration window.

Note

If the default axis has been selected under "Use axis parameters of", the unit of measurement can be edited. If a configured axis has been selected, the unit of measurement for this axis will be displayed.

Use axis parameters of

From the drop-down list, select which axis parameters are to be used for selecting the graphic view of and checking the movement sequence. Select "Default axis" if you have yet to add an axis to the "Technology object" folder or wish to use values which have not been configured in any of the available axes.

The axis parameters of the axis selected at the "Axis" parameter will be used to process the command table in the user program.

Unit of measurement

Enter the unit of measurement for the default axis in this field. If a preconfigured axis has been selected under "Use axis parameters of", the unit of measurement configured in these parameter will be displayed.

Configuration - Dynamics

Configure the acceleration and deceleration and the jerk limit for the default axis in the "Dynamics" configuration window.

Note

If the default axis has been selected under "Use axis parameters of", the following fields can be edited. If a configured axis has been selected, the values of this axis will be displayed.

Acceleration / deceleration

Set the desired acceleration of the default axis in the "Acceleration" field. The desired deceleration can be set in the "Deceleration" field.

Motion jobs configured in the command table will be calculated with the selected acceleration / deceleration.

Limit values:

- 1.0e-12 ≤ acceleration ≤ 1.0e12
- 1.0e-12 ≤ deceleration ≤ 1.0e12

Activate jerk limit

Activate the jerk limit with this checkbox.

Step

Set the desired step for ramping up and ramping down in the "Step" field.

Motion jobs configured in the command table will be calculated with the selected step.

Limit values:

- 1.0e-12 ≤ jerk ≤ 1.0e12

Configuration - Limit values

Configure the maximum velocity, the start/stop velocity and the software limit switches of the default axis in the "Limits" configuration window.

Note

If the default axis has been selected under "Use axis parameters of", the following fields can be edited. If a configured axis has been selected, the values of this axis will be displayed.

Maximum velocity / Start/stop velocity

Define the maximum permissible velocity and the start/stop velocity of the default axis in these fields. The start/stop velocity is the minimum permissible velocity of the default axis.

Limit values:

- 1.0e-12 ≤ start/stop velocity ≤ 1.0e12
  Start/stop velocity = 0.0
- 1.0e-12 ≤ maximum velocity ≤ 1.0e12
  Maximum velocity = 0.0

The value of the maximum velocity must be greater or equal to the value of the start/stop velocity.

Enable software limit switches

Activate the function of the low and high software limit switch with this checkbox. Movements in response to software limit switches being reached are not shown in the trend view.

Low / high software limit switch

Enter the position value of the low and high software limit switches in these fields.

Limits:

- -1.0e12 ≤ low software limit switch ≤ -1.0e-12
  1.0e-12 ≤ low software limit switch ≤ 1.0e12
  Low software limit switch = 0.0
- -1.0e12 ≤ high software limit switch ≤ -1.0e-12
  1.0e-12 ≤ high software limit switch ≤ 1.0e12
  High software limit switch = 0.0

The value of the high software limit switch must be greater than or equal to the value of the low software limit switch.

Download to CPU

When loading to the CPU S7-1200 always ensure that the project files are consistent after the online and offline loading. It is not possible to load single blocks. When selecting single blocks, all new and modified blocks are always loaded.

The following object groups can be loaded to the CPU:

Context menu command "Download to device"	Description
All	Download all new and modified blocks and a new or modified hardware configuration
Hardware configuration	Download a new or modified hardware configuration
Software	Download all new and modified blocks
Software (all blocks)	Download all blocks

The data of the Motion Control technology objects are saved in the data blocks. The conditions for downloading of "blocks" thus apply when loading a new or modified technology object.

CAUTION

Possible malfunctions of the axis when loading without hardware configuration

The hardware configuration is modified when the following modifications are made to the axis configuration:

- Modification of the pulse generator (PTO)
- Modification of the HW limit switch address
- Modification of the homing switch address

If the modified configuration of the axis is loaded with the context menu commands "Software" or "Software (all blocks)" without downloading the hardware configuration, this can lead to malfunctions of the axis.

Ensure that the current hardware configuration is downloaded to the CPU under the conditions listed below.

Download in CPU S7-1200 RUN operating mode (from firmware version V2.2)

For CPU S7-1200 from firmware version V2.2, when loading in CPU RUN operating mode it is checked whether it is possible to load without stopping the CPU.

The following conditions apply when loading data blocks in RUN operating mode:

	Download to load memory	Download to work memory
Data block modified values	Yes	No
Data block modified structure	Yes (as of firmware V4)	Yes (as of firmware V4) • When downloading with reinitialization • For tags in reserve for downloading without reinitialization
	No (firmware V2.2...3)	No (firmware V2.2...3)

	Download to load memory	Download to work memory
New data block	Yes	Yes
Data block deleted	Yes	Yes

Also note the following when deleting data blocks and downloading data blocks with reinitialization:

- The axis must be locked when downloading a positioning axis technology object.
- When downloading a command table technology object, no MC_CommandTable command with this command table must be active (parameter "Busy" = FALSE).
- When downloading an MC_Power instance data block, no MC_Power instruction must be active (parameter "Busy" = FALSE).

From technology version V3.0, Motion Control technology objects (data blocks) can also be downloaded in CPU RUN operating mode.

Technology objects after V3.0 cannot be downloaded in CPU RUN operating mode.

Select one of the actions described below to download the modified version of a Motion Control technology object (from version V3.0) to the work memory:

- Technology object axis and command table
  Change the CPU operating mode from STOP to RUN.
- Technology object axis
  Disable the axis and execute a "Restart" using the Motion Control instruction "MC_Reset".
- Technology object command table
  Ensure that the command table is not being used. Download the data block of the command table to the work memory using the extended instruction "READ_DBL".

See also

Guidelines on use of motion control (Page 10018)

MC_Reset: Acknowledge fault V4 (Page 4827)

Commissioning the axis - Axis control panel

Use the axis command table to move the axis in manual mode, to optimize the axis settings, and to test your system.

The axis control table can only be used if an online connection to the CPU is established.

Note

Response times of the axis control panel

The response time during axis control table operation depends on the communication load of the CPU. Close all other online windows of the TIA Portal to minimize the response time.

"Manual control" button

Click "Manual control" to move the axis in manual control mode. Start by disabling the axis in the user program using Motion Control instruction "MC_Power". In "Manual control" mode, the axis control table takes over control priority for the axis functions. The user program has no influence on the axis functions until manual control is ended.

> **CAUTION**
>
> **Additional axes in automatic mode**
>
> The Manual control is active for one axis only. If additional axes are in automatic mode, dangerous situations may arise as a result.
>
> In this happens, disable all other axes.

"Automatic mode" button

Click "Automatic mode" to end the "Manual control" mode. The axis control table passes back the control priority and the axis can be controlled by the user program again. The axis must be re-enabled in the user program and homed, if required.

Complete all active traversing motions before switching to automatic control; otherwise, the axis will be braked with the emergency stop deceleration.

"Enable" button

Click "Enable" to enable the axis in "Manual control" mode. When the axis is enabled, the axis control panel functions can be used.

If the axis cannot be enabled because certain conditions are not met, note the error message in the "Error message" box. Information on eliminating errors is available in the appendix under "List of ErrorIDs and ErrorInfos". After the error has been corrected, enable the axis again.

"Disable" button

Click "Disable" if you want to temporarily disable the axis in "Manual control" mode.

"Command" area

Operation in the "Command" area is only possible if the axis is enabled. You can select one of the following command inputs:

- Jogging
  This command is equivalent to Motion Control command "MC_MoveJog" in the user program.
- Positioning
  This command is equivalent to the Motion Control commands "MC_MoveAbsolute" and "MC_MoveRelative" in the user program. The axis must be homed for absolute positioning.
- Homing
  This command is equivalent to Motion Control command "MC_Home" in the user program.
  - The "Set reference point" button corresponds to Mode = 0 (direct homing absolute)
  - The "Active homing" button corresponds to Mode = 3 (active homing)

  For active homing, the homing switch must be configured in the axis configuration.
  The values for approach velocity, homing velocity, and reference position offset are taken from the axis configuration unchanged.

Depending on the selection, the relevant boxes for entry of setpoints and the buttons for starting the command are displayed.

Jerk limitation can be activated and deactivated using the "Activate jerk limit" button. By default, the jerk is applied with 10% of the configured value. This value can be changed as required.

"Axis status" area

If "Manual control" mode is activated, the current axis status and drive status are shown in the "Axis status" area. The current position and velocity of the axis are displayed at "Process values".

Click "Acknowledge" to acknowledge all cleared errors.

The "Info message" box displays advanced information about the status of the axis.

"Current values" area

This area displays the current position and velocity of the axis.

Error message

The "Error message" box shows the current error. In "Manual control" mode, the error entry can be deleted by pressing the "Acknowledge" button once the error is eliminated.

Note

Initial values for velocity, acceleration / deceleration and jerk

For safety reasons, the "Velocity", "Acceleration/deceleration" and "Jerk" parameters are initialized with values equivalent to only 10% of the configured values when the axis command table is activated. The "Jerk" parameter is only used for technology object "Axis" V2.0 and higher.

The values in the configuration view displayed when you select "Extended parameters > Dynamics > General" are used for initialization.

The "Velocity" parameter on the control panel is derived from the "Maximum velocity" and the "Acceleration/deceleration" parameters from "Acceleration" in the configuration.

The "Velocity", "Acceleration/deceleration" and "Jerk" parameters can be changed in the axis command table; this does not affect the values in the configuration.

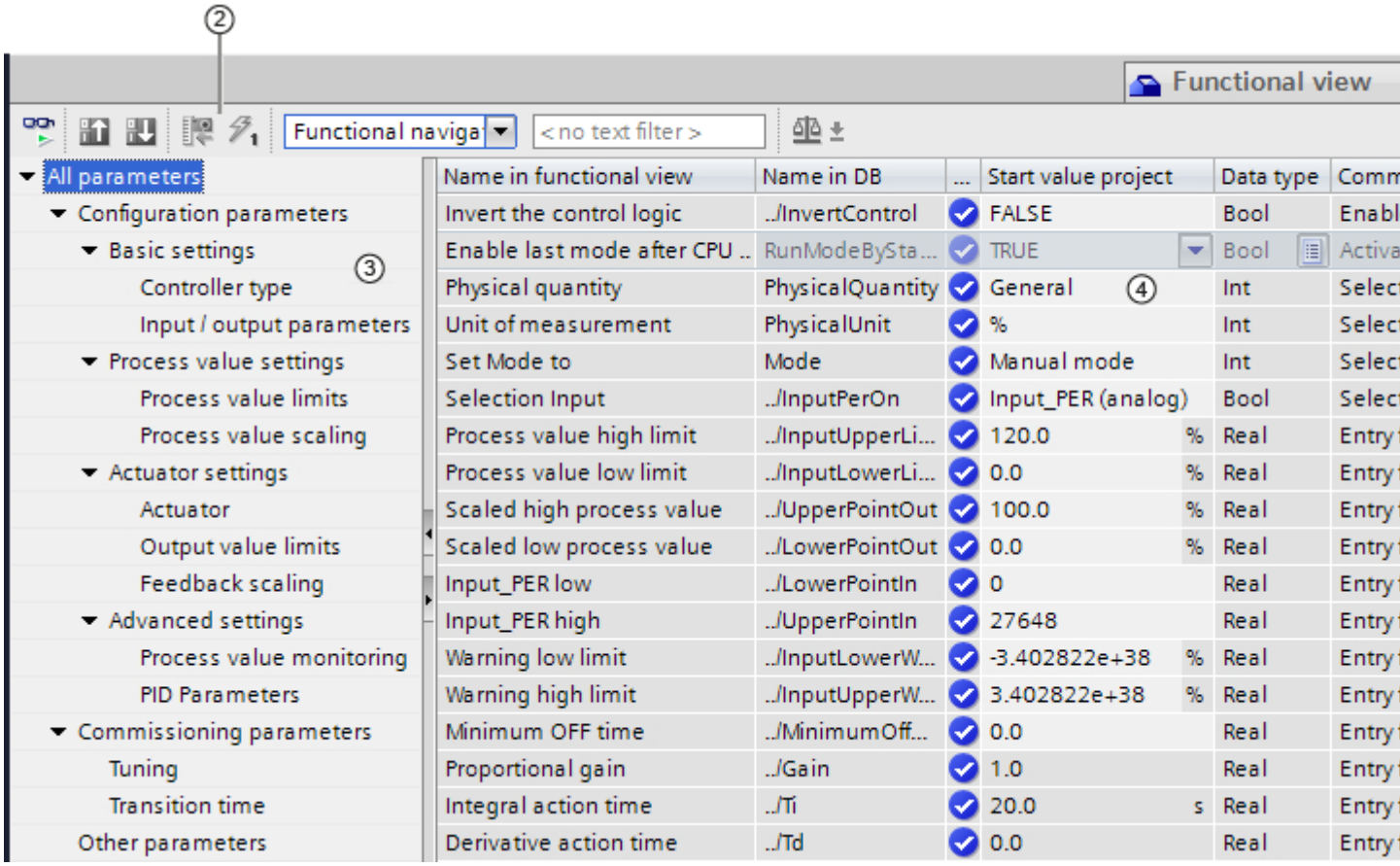
See also

Guidelines on use of motion control (Page 10018)

Parameter view

Introduction to the parameter view

The Parameter view provides you with a general overview of all relevant parameters of a technology object. You obtain an overview of the parameter settings and can easily change them in offline and online mode.



① "Parameter view" tab

② Toolbar (Page 9543)

③ Navigation (Page 9544)

④ Parameter table (Page 9544)

Function scope

The following functions are available for analyzing the parameters of the technology objects and for enabling targeted monitoring and modification.

Display functions:

- Display of parameter values in offline and online mode
- Display of status information of the parameters

- Display of value deviations and option for direct correction
- Display of configuration errors
- Display of value changes as a result of parameter dependencies
- Display of all memory values of a parameter: Start value PLC, Start value project, Monitor value
- Display of the parameter comparison of the memory values of a parameter

Operator control functions:

- Navigation for quickly changing between the parameters and parameter structures.
- Text filter for faster searches for particular parameters.
- Sorting function for customizing the order of parameters and parameter groups to requirements.
- Memory function for backing up structural settings of the Parameter view.
- Monitoring and modifying of parameter values online.
- Function for saving a snapshot of parameter values of the CPU in order to capture momentary situations and to respond to them.
- Function for applying a snapshot of parameter values as start values.
- Download of modified start values to the CPU.
- Comparison functions for comparing parameter values with one another.

Validity

The Parameter view is available for the following technology objects:

- PID_Compact
- PID_3Step
- CONT_C (S7-1500 only)
- CONT_S (S7-1500 only)
- TCONT_CP (S7-1500 only)
- TCONT_S (S7-1500 only)
- TO_Axis_PTO (S7-1200 Motion Control)
- TO_Positioning_Axis (S7-1200 Motion Control)
- TO_CommandTable_PTO (S7-1200 Motion Control)
- TO_CommandTable (S7-1200 Motion Control)

Structure of the parameter view

Toolbar

The following functions can be selected in the toolbar of the parameter view.

Icon	Function	Explanation
	Monitor all	Starts the monitoring of visible parameters in the active Parameter view (online mode).
	Create snapshot of monitor values and accept setpoints of this snapshot as start values	Applies the current monitor values to the “Snapshot” column and updates the start values in the project. Only in online mode for PID_Compact and PID_3Step.
	Initialize setpoints	Transfers the start values updated in the project to the CPU. Only in online mode for PID_Compact and PID_3Step.
	Create snapshot of monitor values	Applies the current monitor values to the “Snapshot” column. Only in online mode.
	Modify all selected parameters immediately and once	This command is executed once and as quickly as possible without reference to any particular point in the user program. Only in online mode.
Functional naviga	Select navigation structure	Toggles between functional navigation and data navigation.
< no text filter >	Text filter...	After entry of a character string: Display of all parameters containing the specified string in one of the currently visible columns.
	Selection of compare values	Selection of parameter values that are to be compared with one another in online mode (Start value project, Start value PLC, Snapshot) Only in online mode.
	Save window settings	Saves your display settings for the Parameter view (e.g., selected navigation structure, activated table columns, etc.)

Navigation

Within the "Parameter view" tab, the following alternative navigation structures can be selected.

Navigation		Explanation
Functional navigation	All parameters Configuration parameters Commissioning parameters Other parameters	In the functional navigation, the structure of the parameters is based on the structure in the configuration dialog ("Functional view" tab), commissioning dialog, and diagnostics dialog. The last group "Other parameters" contains all other parameters of the technology object.
Data navigation	All parameters Input Output InOut Static Other parameters	In the data navigation, the structure of the parameters is based on the structure in the instance DB. The last group "Other parameters" contains the parameters that are not contained in the instance DB.

You can use the "Select navigation structure" drop-down list to toggle the navigation structure.

Parameter table

The table below shows the meaning of the individual columns of the parameter table. You can show or hide the columns as required.

- Column "Offline" = X: Column is visible in offline mode.
- Column "Online" = X: Column is visible in online mode (online connection to the CPU).

Column	Explanation	Offline	Online
Name in functional view	Name of the parameter in the functional view. The display field is empty for parameters that are not configured via the technology object.	X	X
Full name in DB	Complete path of the parameter in the instance DB. The display field is empty for parameters that are not contained in the instance DB.	X	X
Name in DB	Name of the parameter in the instance DB. If the parameter is part of a structure or UDT, the prefix ". ./" is added. The display field is empty for parameters that are not contained in the instance DB.	X	X
Status of configuration	Display of the completeness of the configuration using status symbols. see Status of configuration (offline) (Page 9552)	X	
Compare result	Result of the "Compare values" function. This column is shown if there is an online connection and the "Monitor all" button is selected. see Comparing values (Page 9538)		X

Column	Explanation	Offline	Online
Start value project	Configured start value in the project. Error indication if entered values have a syntax or process-related error.	X	X
Default value	Value that is pre-assigned to the parameter. The display field is empty for parameters that are not contained in the instance DB.	X	X
Snapshot	Snapshot of the current values in the CPU (monitor values). Error indication if values have a process-related error.	X	X
Start value PLC	Start value in the CPU. This column is shown if there is an online connection and the "Monitor all" button is selected. Error indication if values have a process-related error.		X
Monitor value	Current value in the CPU. This column is shown if there is an online connection and the "Monitor all" button is selected. Error indication if values have a process-related error.		X
Modify value	Value that is to be used to change the monitor valuet. This column is shown if there is an online connection and the "Monitor all" button is selected. Error indication if entered values have a syntax or process-related error.		X
Selection for transmission	Selection of the Modify values that are to be transmitted using the "Modify all selected parameters immediately and once" button. This column is displayed together with the "Modify value" column.		X
Minimum value	Minimum process-related value of the parameter. If the minimum value is dependent on other parameters, it is defined: • Offline: By the Start value project. • Online: By the Monitor values.	X	X
Maximum value	Maximum process-related value of the parameter. If the maximum value is dependent on other parameters, it is defined: • Offline: By the Start value project. • Online: By the Monitor values.	X	X
Setpoint	Designates the parameter as a setpoint. These parameters can be initialized online.	X	X
Data type	Data type of the parameter. The display field is empty for parameters that are not contained in the instance DB.	X	X
Retain	Designates the value as a retentive value. The values of retentive parameters are retained even after the voltage supply is switched off.	X	X
Accessible from HMI	Indicates whether the HMI can access this parameter during runtime.	X	X
Visible in HMI	Indicates whether the parameter is visible in the selection list of the HMI by default.	X	X
Comment	Brief description of the parameter.	X	X

Opening the parameter view

Requirement

The technology object has been added in the project tree, i.e., the associated instance DB of the instruction has been created.

Procedure

1. Open the "Technology objects" folder in the project tree.
2. Open the technology object in the project tree.
3. Double-click the "Configuration" object.
4. Select the "Parameter view" tab in the top right corner.

Result

The Parameter view opens. Each displayed parameter is represented by one row in the parameter table.

The displayable parameter properties (table columns) vary depending on whether you are working with the Parameter view in offline or online mode.

In addition, you can selectively display and hide individual table columns.

See also

Default setting of the parameter view (Page 9546)

Default setting of the parameter view

Default settings

To enable you to work efficiently with the Parameter view, you can customize the parameter display and save your settings.

The following customizations are possible and can be saved:

- Show and hide columns
- Change column width
- Change order of the columns
- Toggle navigation
- Select parameter group in the navigation
- Selection of compare values

Show and hide columns

To show or hide columns in the parameter table, follow these steps:

1. Position the cursor in the header of the parameter table.
2. Select the "Show/Hide" command in the shortcut menu.
   The selection of available columns is displayed.
3. To show a column, select the check box for the column.
4. To hide a column, clear the check box for the column.

or

1. Position the cursor in the header of the parameter table.
2. Select the "Show all columns" command in the shortcut menu if all columns of the offline or online mode are to be displayed.

Some columns can only be displayed in online mode: see Parameter table (Page 9544).

Change column width

To customize the width of a column so that all texts in the rows can be read, follow these steps:

1. Position the cursor in the header of the parameter table to the right of the column to be customized until the shape of the cursor changes to a cross.
2. Then double-click this location.

or

1. Open the shortcut menu on the header of the parameter table.
2. Click
   – "Optimize column width" or
   – "Optimize width of all columns".

If the column width setting is too narrow, the complete content of individual fields are shown if you hover the cursor briefly over the relevant field.

Change order of the columns

The columns of the parameter table can be arranged in any way.

To change the order of the columns, follow these steps:

1. Click on the column header and use a drag-and-drop operation to move it to the desired location.
   When you release the mouse button, the column is anchored to the new position.

Toggle navigation

To toggle the display form of the parameters, follow these steps:

1. Select the desired navigation in the “Select navigation structure” drop-down list.
   - Data navigation
   - Functional navigation

See also Navigation (Page 9544).

Select parameter group in the navigation

Within the selected navigation, you choose between the “All parameters” display or the display of a subordinate parameter group of your choice.

1. Click the desired parameter group in the navigation.
   The parameter table only displays the parameters of the parameter group.

Selection of compare values (online)

To set the compare values for the “Compare values” function, follow these steps:

1. Select the desired compare values in the “Selection of compare values” drop-down list.
   - Start value project / Start value PLC
   - Start value project / Snapshot
   - Start value PLC / Snapshot

The “Start value project / Start value PLC” option is set by default.

Save default setting of the Parameter view

To save the above customizations of the Parameter view, follow these steps:

1. Customize the Parameter view according to your requirements.
2. Click the “Save window settings” button at the top right of the Parameter view.

Working with the parameter view

Overview

The following table provides an overview of the functions of the Parameter view in online and offline mode described in the following.

- Column "Offline" = X: This function is possible in offline mode.
- Column "Online" = X: This function is possible in online mode.

Function/action	Offline	Online
Filtering the parameter table (Page 9549)	X	X
Sorting the parameter table (Page 9550)	X	X
Transferring parameter data to other editors (Page 9550)	X	X
Indicating errors (Page 9550)	X	X
Editing start values in the project (Page 9551)	X	X
Status of configuration (offline) (Page 9552)	X	
Monitoring values online in the parameter view (Page 9553)		X
Create snapshot of monitor values (Page 9553)		X
Modifying values (Page 9554)		X
Comparing values (Page 9555)		X
Applying values from the online program as start values (Page 9556)		X
Initializing setpoints in the online program (Page 9557)		X

Filtering the parameter table

You can filter the parameters in the parameter table in the following ways:

- With the text filter
- With the subgroups of the navigation

Both filter methods can be used simultaneously.

With the text filter

Texts that are visible in the parameter table can be filtered. This means only texts in displayed parameter rows and columns can be filtered.

1. Enter the desired character string for filtering in the “Text filter...” input box.
   The parameter table displays only the parameters containing the character string.

The text filtering is reset.

- When another parameter group is selected in the navigation.
- When navigation is changed from data navigation to functional navigation, or vice versa.

With the subgroups of the navigation

1. Click the desired parameter group in the navigation, e.g., "Static".
   The parameter table only shows the static parameters. You can select further subgroups for some groups of the navigation.
2. Click “All parameters” in the navigation if all parameters are to be shown again.

Sorting the parameter table

The values of the parameters are arranged in rows. The parameter table can be sorted by any displayed column.

- In columns containing numerical values, sorting is based on the magnitude of the numerical value.
- In text columns, sorting is alphabetical.

Sorting by column

1. Position the cursor in the header cell of the desired column.
   The background of this cell turns blue.
2. Click the column header.

Result

The entire parameter table is sorted by the selected column. A triangle with tip facing up appears in the column header.

Clicking the column header again changes the sorting as follows:

- Symbol "▲": Parameter table is sorted in ascending order.
- Symbol "▼": Parameter table is sorted in descending order.
- No symbol: The sorting is removed again. The parameter table assumes the default display.

The "../" prefix in the "Name in DB" column is ignored when sorting.

Transferring parameter data to other editors

After selecting an entire parameter row of the parameter table, you can use the following:

- Drag-and-drop
- <Ctrl+C>/<Ctrl+V>
- Copy/Paste via shortcut menu

Transfer parameters to the following editors of the TIA Portal:

- Program editor
- Watch table
- Signal table for trace function

The parameter is inserted with its full name: See information in "Full name in DB" column.

Indicating errors

Error indication

Parameter assignment errors that result in compilation errors (e.g., limit violation) are indicated in the Parameter view.

Every time a value is input in the Parameter view, a check is made for process-related and syntax errors and the result is indicated.

Bad values are indicated by:

- Red error symbol in the "Status of configuration" (offline mode) or "Compare result" (online mode, depending on the selected comparison type) columns

and/or

- Table field with red background
  If you click the bad field, a roll-out error message appears with information of the permissible value range or the required syntax (format)

Compilation error

From the error message of the compiler, you can directly open the Parameter view (functional navigation) containing the parameter causing the error in situations where the parameter is not displayed in the configuration dialog.

Editing start values in the project

With the Parameter view, you can edit the start values in the project in offline mode and online mode.

- You make value changes in the “Start value project” column of the parameter table.
- In the “Status of configuration” column of the parameter table, the progress of the configuration is indicated by the familiar status symbols from the configuration dialog of the technology object.

Boundary conditions

- If other parameters depend on the parameter whose start value was changed, the start value of the dependent parameters are also adapted.
- If a parameter of a technology object is not editable, it is also not editable in the parameter view. The ability to edit a parameter can also depend on the values of other parameters.

Defining new start values

To define start values for parameters in the Parameter view, follow these steps:

1. Open the Parameter view of the technology object.
2. Enter the desired start values in the "Start value project" column. The value must match the data type of the parameter and must not exceed the value range of the parameter. The limits of the value range can be seen in the “Maximum value” and “Minimum value” columns.

The "Status of configuration" column indicates the progress of the configuration with colored symbols.

See also Status of configuration (offline) (Page 9552)

Following adaptation of the start values and downloading of the technology object to the CPU, the parameters take the defined value at startup if they are not declared as retentive (“Retain” column).

Error indication

When a start value is input, a check is made for process-related and syntax errors and the result is indicated.

Bad start values are indicated by:

- Red error symbol in the "Status of configuration" (offline mode) or "Compare result" (online mode, depending on the selected comparison type) columns

and/or

- Red background in the “Start value project” field
  If you click on the bad field, a roll-out error message appears with information of the permissible value range or the necessary syntax (format)

Correcting bad start values

1. Correct bad start values using information from the roll-out error message.
   Red error symbol, red field background, and roll-out error message are no longer displayed.

The project cannot be successfully compiled unless the start values are error-free.

Status of configuration (offline)

The status of the configuration is indicated by icons:

- In the “Status of configuration” column in the parameter table
- In the navigation structure of the functional navigation and data navigation

Symbol in “Status of configuration” column

Symbol	Meaning
	The start value of the parameter corresponds to the default value and is valid. A start value has not yet been defined by the user.
	The start value of the parameter contains a value defined by the user. The start value is different than the default value. The start value is error-free and valid.
	The start value of the parameter is invalid (syntax or process-related error). The input box has a red background. When clicked, the roll-out error message indicates the cause of the error.
	Only for S7-1200 Motion Control: The start value of the parameter is valid but contains warnings. The input box has a yellow background.

Symbol in the navigation

The symbols in the navigation indicate the progress of the configuration in the same way as in the configuration dialog of the technology object.

See Configure technology objects (Page 9534)

Monitoring values online in the parameter view

You can monitor the values currently taken by the parameters of the technology object in the CPU (monitor values) directly in the Parameter view.

Requirements

- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").
- The Parameter view of the technology object is open.

Procedure

1. Start the monitoring by clicking .
   As soon as the Parameter view is online, the following columns are additionally displayed:
   - Compare result
   - Start value PLC
   - Monitor value
   - Modify value
   - Selection for transmission

   The "Monitor value" column shows the current parameter values on the CPU.
   Meaning of the additional columns: see Parameter table (Page 9544)
2. Stop the monitoring by clicking again.

Display

All columns that are only available online have an orange background:

- Values in light-orange cells can be changed.
- Values in cells with a dark orange background cannot be changed.

Create snapshot of monitor values

You can back up the current values of the technology object on the CPU (monitor values) and display them in the Parameter view.

Requirements

- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").
- The Parameter view of the technology object is open.
- The “Monitor all” button is selected.

Procedure

To show the current parameter values, follow these steps:

1. In the Parameter view, click the “Create snapshot of monitor values" icon .

Result

The current monitor values are transferred once to the "Snapshot" column of the parameter table.

You can analyze the values "frozen" in this way while the monitor values continue to be updated in the "Monitor values" column.

Modifying values

With the Parameter view, you can modify values of the technology object in the CPU.

You can assign values to the parameter once (Modify value) and modify them immediately. The modify request is executed as quickly as possible without reference to any particular point in the user program.

DANGER

Danger when modifying:

Changing the parameter values while the plant is operating may result in severe damage to property and personal injury in the event of malfunctions or program errors.

Make sure that dangerous states cannot occur before you use the "Modify" function.

Requirements

- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").
- The Parameter view of the technology object is open.

- The “Monitor all” button is selected.
- The parameter can be modified (associated field in the "Modify value" column has a light-orange background).

Procedure

To modify parameters immediately, follow these steps:

1. Enter the desired modify values in the “Modify values” column of the parameter table.
2. Check whether the check box for modifying is selected in the "Select for transmission" column.
   The modify values and associated check boxes of dependent parameters are automatically adapted at the same time.
3. Click the “Modify all selected parameters immediately and once” icon .

The selected parameters are modified once and immediately with the specified values and can be monitored in the "Modify values" column. The check boxes for modifying in the "Selection for transmission" column are automatically cleared after the modify request is complete.

Error indication

When a start value is input, a check is made immediately for process-related and syntax errors and the result is indicated.

Bad start values are indicated by:

- Red background in the “Modify value” field

and

- If you click the bad field, a roll-out error message appears with information of the permissible value range or the necessary syntax (format)

Bad modify values

- Modify values with process-related errors can be transmitted.
- Modify values with syntax errors **cannot** be transmitted.

Comparing values

You can use comparison functions to compare the following memory values of a parameter:

- Start value project
- Start value PLC
- Snapshot

Requirements

- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").
- The Parameter view of the technology object is open.
- The “Monitor all” button is selected.

Procedure

To compare the start values on the various target systems, follow these steps:

1. Click the "Selection of compare values" icon .
   A selection list containing the comparison options opens:
   – Start value project - Start value PLC (default setting)
   – Start value project - Snapshot
   – Start value PLC - Snapshot
2. Select the desired comparison option.
   The selected comparison option is executed as follows:
   – A scales symbol appears in the header cells of the two columns selected for comparison.
   – Symbols are used in the "Compare result" column to indicate the result of the comparison of the selected columns.

Symbol in "Compare result" column

Symbol	Meaning
	The compare values are equal and error-free.
	The compare values are not equal and error-free.
	At least one of the two compare values has a process-related or syntax error.
	The comparison cannot be performed. At least one of the two compare values is not available (e.g., snapshot).

Symbol in the navigation

The symbols are shown in the same way in the navigation if the comparison result applies to at least one of the parameters below the displayed navigation structure.

Applying values from the online program as start values

In order to apply optimized values from the CPU to the project as start values, you create a snapshot of the monitor values. Values of the snapshot marked as a "Setpoint" are then applied to the project as start values.

Requirements

- The technology object is of type "PID_Compact" or "PID_3Step".
- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").
- The Parameter view of the technology object is open.
- The “Monitor all” button is selected.

Procedure

To apply optimized values from the CPU, follow these steps:

1. Click the "Create snapshot of monitor values and accept setpoints of this snapshot as start values" icon .

Result

The current monitor values are applied to the "Snapshot" column and their setpoints are copied to the "Start value project" column as new start values.

Note

Applying values of individual parameters

You can also apply the values of individual parameters that are not marked as a setpoint from the "Snapshot" column to the "Start values project" column. To do so, copy the values and insert them into the "Start value project" column using the "Copy" and "Paste" commands in the shortcut menu.

Initializing setpoints in the online program

You can initialize all parameters that are marked as a "Setpoint" in the Parameter view with new values in the CPU in one step. In so doing, the start values are downloaded from the project to the CPU. The CPU remains in "RUN" mode.

To avoid data loss on the CPU during a cold restart or warm restart, you must also download the technology object to the CPU.

DANGER

Danger when changing parameter values

Changing the parameter values while the plant is operating may result in severe damage to property and personal injury in the event of malfunctions or program errors.

Make sure that dangerous states cannot occur before you reinitialize the setpoints.

Requirements

- The technology object is of type "PID_Compact" or "PID_3Step".
- There is an online connection.
- The technology object is downloaded to the CPU.
- The program execution is active (CPU in "RUN").
- The Parameter view of the technology object is open.
- The “Monitor all” button is selected.
- The parameters marked as a "Setpoint" have a "Start value project" that is free of process-related and syntax errors

Procedure

To initialize all setpoints, follow these steps:

1. Enter the desired values in the "Start value project" column.
   Ensure that the start values are free of process-related and syntax errors.
2. Click the "Initialize setpoints" icon .

Result

The setpoints in the CPU are initialized with the start values from the project.

Programming

Overview of the Motion Control statements

You control the axis with the user program using Motion Control instructions. The instructions start Motion Control commands that execute the desired functions.

The status of the Motion Control commands and any errors that occur during their execution can be obtained from the output parameters of the Motion Control instructions. The following Motion Control instructions are available:

- MC_Power: Enable, disable axis V4 (Page 4823)
- MC_Reset: Acknowledge fault V4 (Page 4827)
- MC_Home: Home axes, set reference point V4 (Page 4829)
- MC_Halt: Halt axes V4 (Page 4832)
- MC_MoveAbsolute: Absolute positioning of axes V4 (Page 4835)
- MC_MoveRelative: Relative positioning of axes V4 (Page 4838)
- MC_MoveVelocity: Move axes at preset rotational speed V4 (Page 4842)
- MC_MoveJog: Move axes in jog mode V4 (Page 4846)
- MC_CommandTable: Run axis commands as motion sequence V4 (Page 4849)
- MC_ChangeDynamic: Change dynamic settings of axis V4 (Page 4851)

- MC_ReadParam: Continuously read motion data of a positioning axis V4 (Page 4853)
- MC_WriteParam: Write tag of positioning axis V4 (Page 4855)

See also

Creating a user program (Page 10097)

Programming notes (Page 10100)

Behavior of the Motion Control commands after POWER OFF and restart (Page 10102)

Monitoring active commands (Page 10102)

Error displays of the Motion Control statements (Page 10113)

Creating a user program

In the section below you learn how to create a user program with the basic configuration for controlling your axis. All available axis functions are controlled using the Motion Control instructions to be inserted.

Requirement

- The technology object has been created and configured without errors.

Before creating and testing the user program, it is advisable to test the function of the axis and the corresponding parts of the system with the axis command table.

Procedure

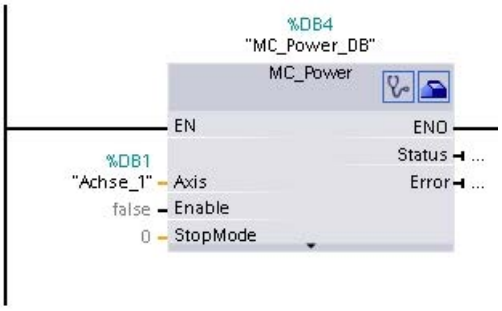
Proceed as follows to create the user program in accordance with the principles described below:

1. In the project tree, double-click your code block (the code block must be called in the cyclic program).
   The code block is opened in the programming editor and all available instructions are displayed.
2. Open the "Technology" category and the "Motion Control" and "S7-1200 Motion Control" folders.
3. Use a drag-and-drop operation to move the "MC_Power" instruction to the desired network of the code block.
   The dialog box for defining the instance DB opens.
4. In the next dialog box, select from the following alternatives:
   Single instance
   Click "Single instance" and select whether you want to define the name and number of the instance DB automatically or manually.
   Multi-instance
   Click "Multi-instance" and select whether you want to define the name of the multi-instance automatically or manually.

5. Click "OK".
   The Motion Control instruction "MC_Power" is inserted into the network.

Parameters marked with "<???>" must be initialized; all other parameters are assigned default values.
Parameters displayed in black are required for use of the Motion Control instruction.

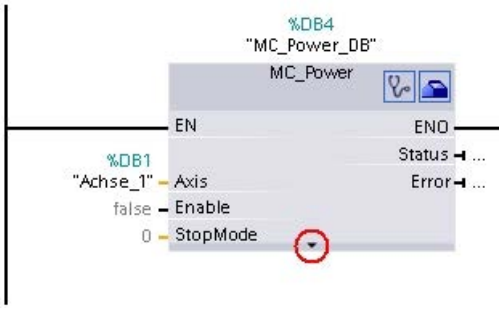
6. Select technology object in the project tree and drag-and-drop it on <???>.



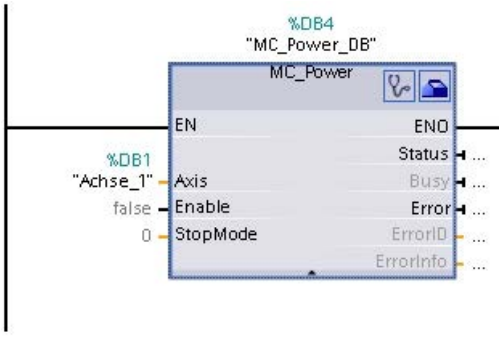
Following selection of the technology object data block, the following buttons are available:

Click the stethoscope icon if you want to open the diagnostics dialog for the technology object.

Click the toolbox icon if you want to open the configuration view of the technology object.



Click the arrow down icon to view additional parameters of the Motion Control instruction.



The grayed-out parameters now visible can be used optionally.

7. Add your choice of Motion Control instructions in accordance with steps 3 to 6.

Result

You have created the basic configuration for axis control in the user program.

Initialize the input parameters of Motion Control instructions in other parts of the user program to initiate the desired jobs for the "Axis" technology object.

Evaluate the output parameters of the Motion Control instructions and the tags of the data block to track the initiated jobs and the status of the axis.

Refer to the detailed description for details on the parameters of Motion Control instructions.

See also

Overview of the Motion Control statements (Page 10096)

Programming notes (Page 10100)

Behavior of the Motion Control commands after POWER OFF and restart (Page 10102)

Monitoring active commands (Page 10102)

Error displays of the Motion Control statements (Page 10113)

Programming notes

When creating your user program, note the following information:

- Cyclic call of utilized motion control instructions
  The current status of command execution is available via the output parameters of the motion control instruction. The status is updated with every call of the motion control instruction. Therefore, make sure that the utilized motion control instructions are called cyclically.
- Transfer of parameter values of a motion control instruction
  The parameter values pending for the input parameters are transferred with a positive edge at input parameter "Execute" when the block is called.
  The motion control command is started with these parameter values. Parameter values that are subsequently changed for the motion control instruction are not transferred until the next start of the motion control command.
  Exceptions to this are input parameters "StopMode" of motion control instruction "MC_Power" and "Velocity" of motion control instruction "MC_MoveJog". A change in the input parameter is also applied when "Enable" = TRUE, or "JogForward" and "JogBackward". .

- **Programming under consideration of the status information**
  In a stepwise execution of motion control jobs, make sure to wait for the active command to finish before starting a new command. Use the status messages of the motion control instruction and the "StatusBits" tag of the technology object to check for completion of the active command.
  In the examples below, observe the indicated sequence. Failure to observe the sequence will display an axis or command error.
  - **Axis enable with motion control instruction "MC_Power"**
    You must enable the axis before it can take on motion jobs. Use an AND operation of tag <Axis name>.StatusBits.Enable = TRUE with output parameter Status = TRUE of motion control instruction "MC_Power" to verify that the axis is enabled.
  - **Acknowledge error with motion control instruction "MC_Reset"**
    Prior to starting a motion control command, errors requiring acknowledgement must be acknowledged with "MC_Reset". Eliminate the cause of the error and acknowledge the error with motion control instruction "MC_Reset". Verify that the error has been successfully acknowledged before initiating a new command. For this purpose, use an AND operation of tag <Axis name>.StatusBits.Error = FALSE with output parameter Done = TRUE of motion control instruction "MC_Reset".
  - **Home axis with motion control instruction "MC_Home"**
    Before you can start an MC_MoveAbsolute command, the axis must be homed. Use an AND operation of tag <Axis name>.StatusBits.HomingDone = TRUE with output parameter Done = TRUE of motion control instruction "MC_Home" to verify that the axis has been homed.
- **Override of motion control command processing**
  Motion control jobs for moving an axis can also be executed as overriding jobs.
  If a new motion control command is started for an axis while another motion control command is active, the active command is overridden by the new command before the existing command is completely executed. The overridden command signals this using CommandAborted = TRUE in the motion control instruction. It is possible to override an active MC_MoveRelative command with a MC_MoveAbsolute command.
- **Avoiding multiple use of the same instance**
  All relevant information of a motion control command is stored in its instance.
  Do not start a new command using this instance, if you want to track the status of the current command. Use different instances if you want to track the commands separately. If the same instance is used for multiple motion control commands, the status and error information of the individual commands will overwrite each other.
- **Call of motion control instructions in different priority classes (run levels)**
  Motion Control instructions with the same instance may not be called in different priority classes without interlocking. To learn how to call locked motion control instructions, refer to "Tracking commands from higher priority classes (run levels) (Page 10123)".

See also

Overview of the Motion Control statements (Page 10096)

Creating a user program (Page 10097)

Behavior of the Motion Control commands after POWER OFF and restart (Page 10102)

Monitoring active commands (Page 10102)

Error displays of the Motion Control statements (Page 10113)

Tracking jobs from higher priority classes (execution levels) (Page 10123)

Behavior of the Motion Control commands after POWER OFF and restart

A POWER OFF or CPU-STOP aborts all active motion control jobs. All CPU outputs, including pulse and direction outputs, are reset.

After a subsequent POWER ON or CPU restart (CPU RUN), the technology objects and the motion control jobs will be reinitialized.

All actual data of the technology objects as well as all status and error information of the previously active motion control jobs are reset to their initial values.

Before the axis can be reused, it must be enabled again using the Motion Control instruction "MC_Power". If homing is required, the axis must be homed again with Motion Control instruction "MC_Home".

See also

Overview of the Motion Control statements (Page 10096)

Creating a user program (Page 10097)

Programming notes (Page 10100)

Monitoring active commands (Page 10102)

Error displays of the Motion Control statements (Page 10113)

Monitoring active commands

Monitoring active commands

There are three typical groups for tracking active Motion Control commands:

- **Motion control instructions with output parameter "Done"**
- **Motion control instruction "MC_MoveVelocity"**
- **Motion control instruction "MC_MoveJog"**

Motion control instructions with "Done" output parameter

Motion control instructions with the output parameter "Done" are started via input parameter "Execute" and have a defined conclusion (for example, with Motion Control instruction "MC_Home": Homing was successful). The command is complete and the axis is at a standstill.

The commands of the following Motion Control instructions have a defined conclusion:

- MC_Reset
- MC_Home
- MC_Halt

- MC_MoveAbsolute
- MC_MoveRelative
- MC_CommandTable (technology object as of V2)
- MC_ChangeDynamic (technology object as of V2)
- MC_WriteParam (as of technology object V4)
- MC_ReadParam (as of technology object V4)

The output parameter "Done" indicates the value TRUE, if the command has been successfully completed.

The output parameters "Busy", "CommandAborted", and "Error" signal that the command is still being processed, has been aborted or an error is pending. The Motion Control instruction "MC_Reset" cannot be aborted and thus has no "CommandAborted" output parameter. The Motion Control instruction "MC_ChangeDynamic" is completed immediately and therefore has no "Busy" or "CommandAborted" output parameters.

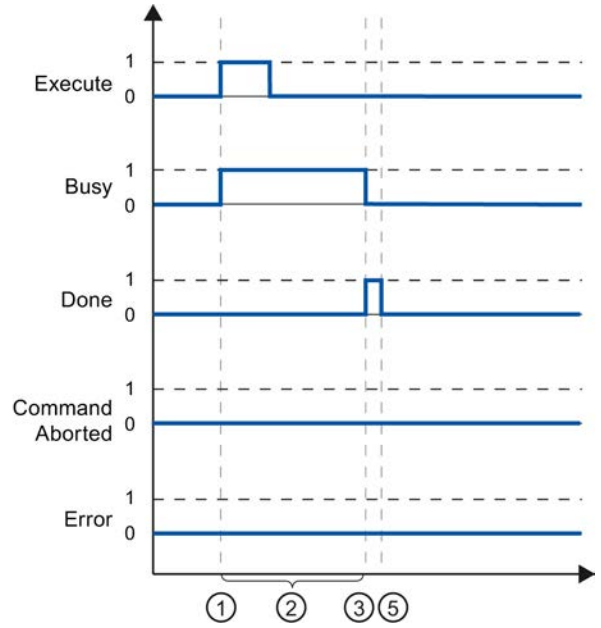
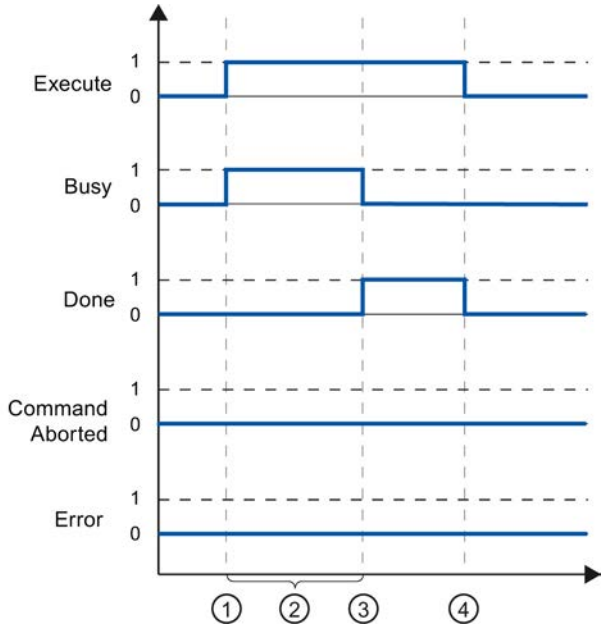
During execution of the Motion Control command, the output parameter "Busy" indicates the value TRUE. If the command has been completed, aborted, or stopped by an error, the output parameter "Busy" changes its value to FALSE. This change occurs regardless of the signal at input parameter "Execute".

Output parameters "Done", "CommandAborted", and "Error" indicate the value TRUE for at least one cycle. These status messages are latched while input parameter "Execute" is set to TRUE.

The behavior of the status bits is presented below for various example situations:

Complete execution of command

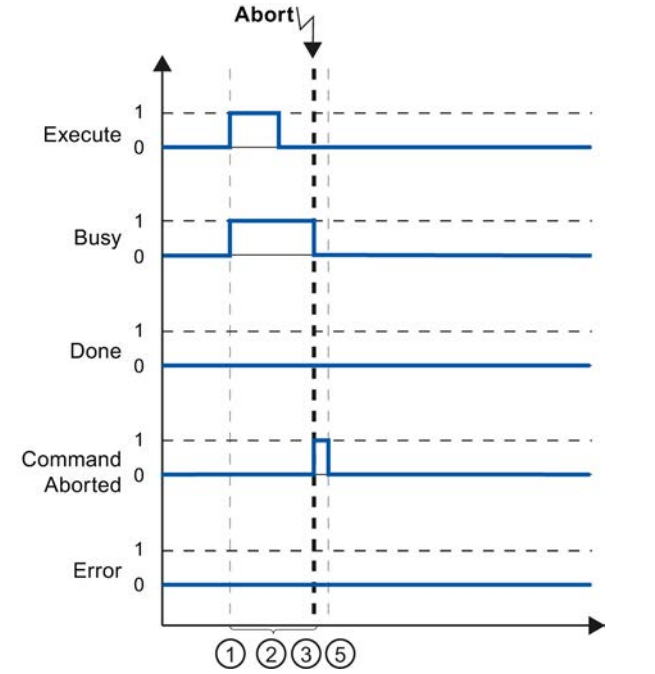
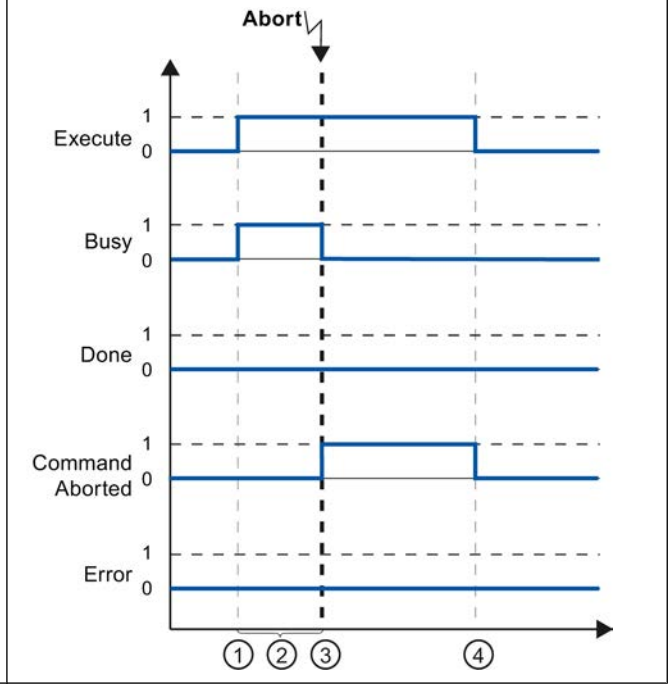
If the Motion Control command has been completely executed by the time of its conclusion, this is indicated by the value TRUE in output parameter "Done". The signal status of the input parameter "Execute" influences the display duration at the output parameter "Done":

"Execute" changes its value to FALSE **during processing** of the command	"Execute" changes its value to FALSE **after completion** of the command
	

①	The command is started with a positive edge at the input parameter "Execute". Depending on the programming, "Execute" can still be reset to the value FALSE during the command, or the value TRUE can be retained until after completion of the command.
②	While the command is active, the output parameter "Busy" indicates the value TRUE.
③	With conclusion of the command (for example, for Motion Control instruction "MC_Home": Homing was successful), output parameter "Busy" changes to FALSE and "Done" to TRUE.
④	If "Execute" retains the value TRUE until after completion of the command, then "Done" also remains TRUE and changes its value to FALSE together with "Execute".
⑤	If "Execute" has been set to FALSE before the command is complete, "Done" indicates the value TRUE for only one execution cycle.

Abort command

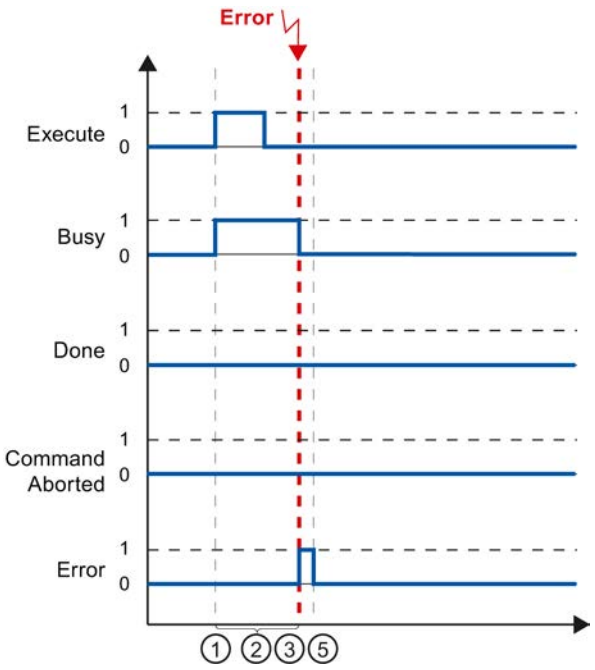
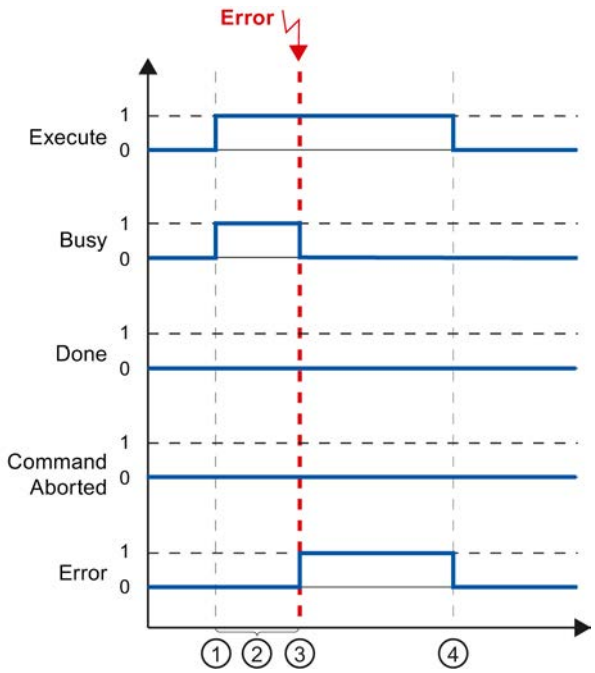
If the Motion Control command is aborted during execution, this is indicated by the value TRUE in output parameter "CommandAborted". The signal status of the input parameter "Execute" influences the display duration at the output parameter "CommandAborted":

"Execute" changes its value to FALSE **before** the command is aborted.	"Execute" changes its value to FALSE **after** the command is aborted.
	

①	The command is started with a positive edge at the input parameter "Execute". Depending on the programming, "Execute" can still be reset to the value FALSE during the command, or the value TRUE can be retained until after completion of the command.
②	While the command is active, the output parameter "Busy" indicates the value TRUE.
③	During command execution, the command is aborted by another Motion Control command. If the command is aborted, output parameter "Busy" changes to FALSE and "CommandAborted" to TRUE.
④	If "Execute" retains the value TRUE until after the command is aborted, then "CommandAborted" also remains TRUE and changes its value to FALSE together with "Execute".
⑤	If "Execute" has been set to FALSE before the command is aborted, "CommandAborted" indicates the value TRUE for only one execution cycle.

Error during command execution

If an error occurs during execution of the Motion Control command, this is indicated by the value TRUE in the output parameter "Error". The signal status of the input parameter "Execute" influences the display duration at the output parameter "Error":

"Execute" changes its value to FALSE **before** the error occurs	"Execute" changes its value to FALSE **after** the error occurs
	

①	The command is started with a positive edge at the input parameter "Execute". Depending on the programming, "Execute" can still be reset to the value FALSE during the command, or the value TRUE can be retained until after completion of the command.
②	While the command is active, the output parameter "Busy" indicates the value TRUE.
③	An error occurred during command execution. When the error occurs, the output parameter "Busy" changes to FALSE and "Error" to TRUE.
④	If "Execute" retains the value TRUE until after the error occurs, then "Error" also remains TRUE and only changes its value to FALSE together with "Execute".
⑤	If "Execute" has been set to FALSE before the error occurs, "Error" indicates the value TRUE for only one execution cycle.

Motion control instruction MC_MoveVelocity

A "MC_MoveVelocity" command is started with a positive edge at the "Execute" parameter. The command objective is fulfilled when the assigned velocity is reached and the axis travels at constant velocity. When the assigned velocity is reached and maintained, this is indicated in the "InVelocity" parameter with the value TRUE.

The motion of the axis can, for example, be stopped with an "MC_Halt" command.

The output parameters "Busy", "CommandAborted", and "Error" signal that the command is still being processed, has been aborted or an error is pending.

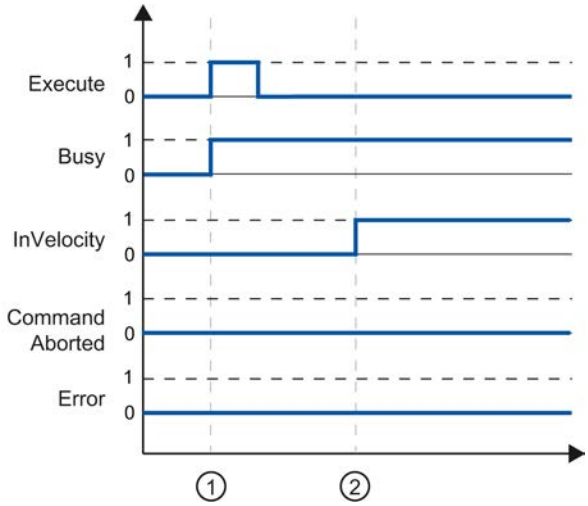
During execution of the Motion Control command, the output parameter "Busy" indicates the value TRUE. If the command has been stopped by another command or by an error, the output parameter "Busy" changes its value to FALSE. This change occurs regardless of the signal at the input parameter "Execute".

The output parameters "CommandAborted" and "Error" show the value TRUE for at least one cycle. These status messages are latched while input parameter "Execute" is set to TRUE.

The behavior of the status bits is presented below for various example situations:

The parameterized velocity is reached

If the Motion Control command has been executed by the time the parameterized velocity is reached, this is indicated by the value TRUE in output parameter "InVelocity". The parameter "Execute" has no effect on the indication duration in the "InVelocity" parameter.

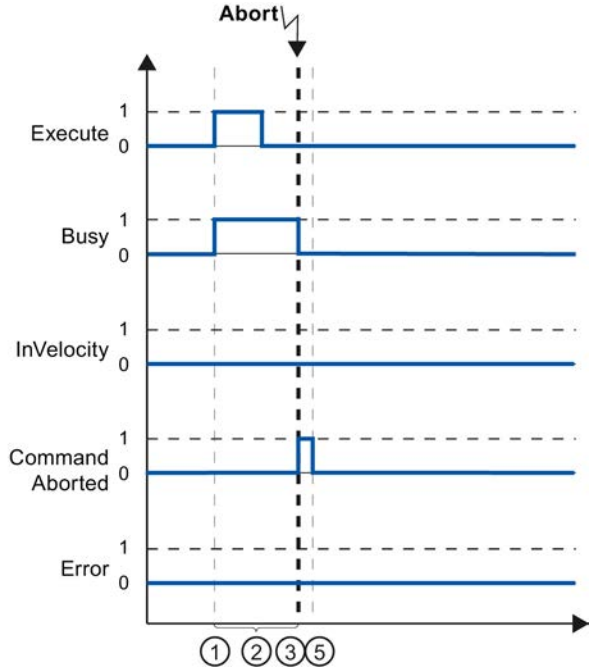
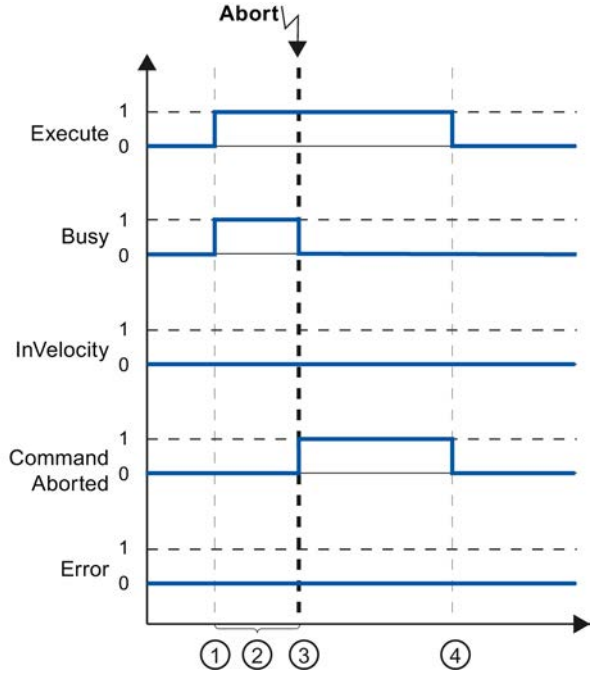


①	The job is started with a positive edge at the "Execute" parameter. Depending on the programming, "Execute" can be reset to the FALSE value before or after the parameterized velocity has been reached. While the job is active, the parameter "Busy" shows the value TRUE.
②	When the assigned velocity is reached, the "InVelocity" parameter changes to TRUE. The "Busy" and "InVelocity" parameters retain the TRUE value until the "MC_MoveVelocity" command is overridden by another Motion Control command or stopped by an error.

The command is aborted prior to reaching the parameterized velocity

If the Motion Control command is aborted before the parameterized velocity is reached, this is indicated by the value TRUE in output parameter "CommandAborted". The signal status of

input parameter "Execute" influences the display duration at the output parameter "CommandAborted".

"Execute" changes its value to FALSE **before** the command is aborted.	"Execute" changes its value to FALSE **after** the command is aborted.
	

①	The command is started with a positive edge at the input parameter "Execute". Depending on the programming, "Execute" can still be reset to the value FALSE during the command, or the value TRUE can be retained until after the command is aborted.
②	While the command is active, the output parameter "Busy" indicates the value TRUE.
③	During command execution, the command is aborted by another Motion Control command. If the command is aborted, output parameter "Busy" changes to FALSE and "CommandAborted" to TRUE.
④	If "Execute" retains the value TRUE until after the command is aborted, then "CommandAborted" also remains TRUE and changes its status to FALSE together with "Execute".
⑤	If "Execute" has been reset to FALSE before the command is aborted, "CommandAborted" indicates the value TRUE for only one execution cycle.

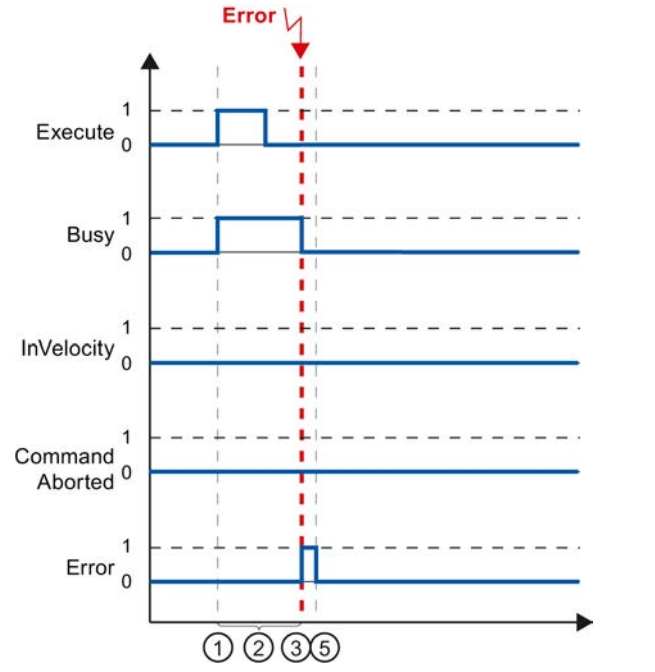
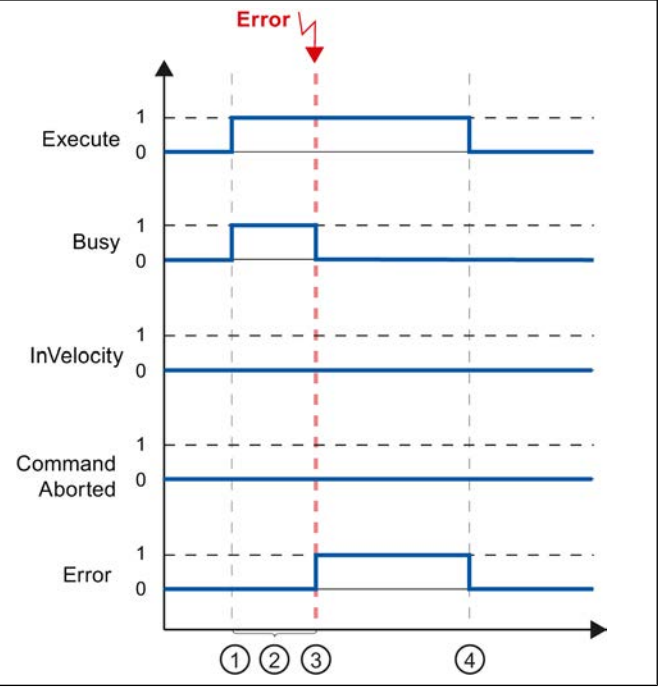
Note

Under the following conditions, an abort is not indicated in output parameter "CommandAborted":

The parameterized velocity has been reached, input parameter "Execute" has the value FALSE, and a new Motion Control command is initiated.

An error has occurred prior to reaching the parameterized velocity

If an error occurs during execution of the Motion Control command before the parameterized velocity has been reached, this is indicated by the value TRUE in the output parameter "Error". The signal status of the input parameter "Execute" influences the display duration at the output parameter "Error":

"Execute" changes its value to FALSE **before** the error occurs	"Execute" changes its value to FALSE **after** the error occurs
	

①	The command is started with a positive edge at the input parameter "Execute". Depending on the programming, "Execute" can still be reset to the value FALSE during the command, or the value TRUE can be retained until after the error has occurred.
②	While the command is active, the output parameter "Busy" indicates the value TRUE.
③	An error occurred during command execution. When the error occurs, the output parameter "Busy" changes to FALSE and "Error" to TRUE.
④	If "Execute" retains the value TRUE until after the error has occurred, then "Error" also remains TRUE and only changes its status to FALSE together with "Execute".
⑤	If "Execute" has been reset to FALSE before the error occurs, "Error" indicates the value TRUE for only one execution cycle.

Note

Under the following conditions, an error is not indicated in output parameter "Error":

The parameterized velocity has been reached, input parameter "Execute" has the value FALSE, and an axis error occurs (software limit switch is approached, for example).

The error of the axis is only indicated in the "MC_Power" Motion Control instruction.

Motion control instruction MC_MoveJog

The commands of Motion Control instruction "MC_MoveJog" implement a jog operation.

The motion control commands "MC_MoveJog" do not have a defined end. The command objective is fulfilled when the parameterized velocity is reached for the first time and the axis travels at constant velocity. When the parameterized velocity is reached, this is indicated by the value TRUE in output parameter "InVelocity".

The order is complete when input parameter "JogForward" or "JogBackward" has been set to the value FALSE and the axis has come to a standstill.

The output parameters "Busy", "CommandAborted", and "Error" signal that the command is still being processed, has been aborted or an error is pending.

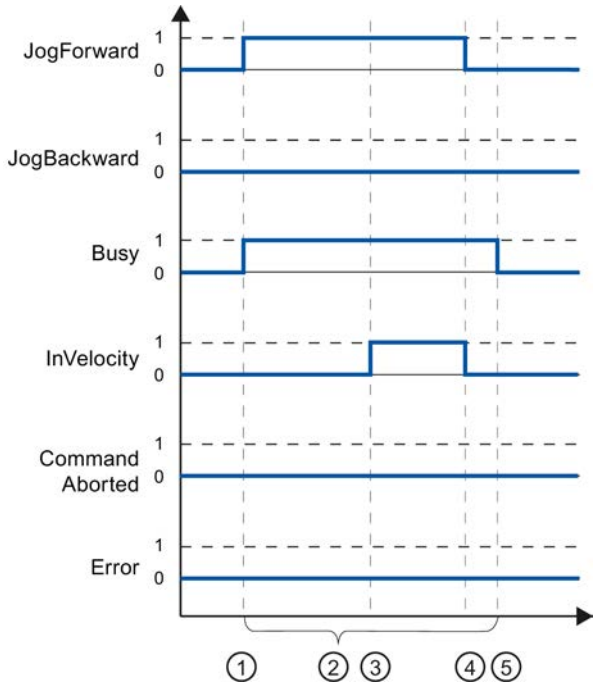
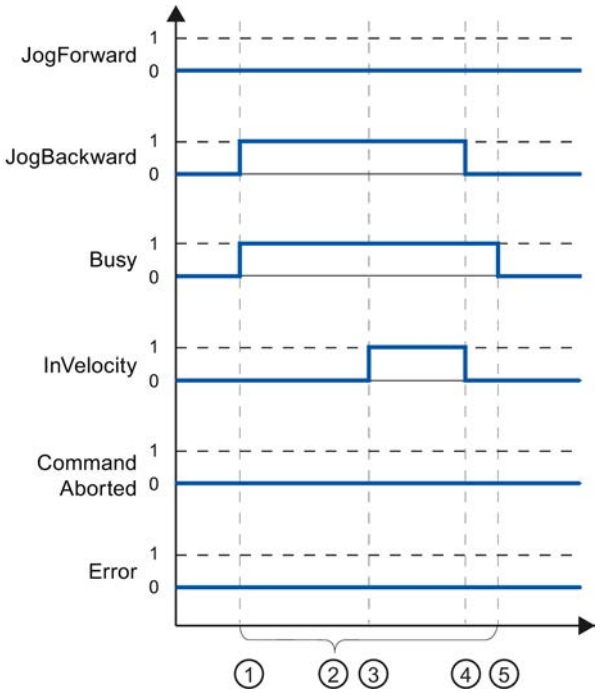
During processing of the motion control command, the output parameter "Busy" indicates the value TRUE. If the command has been completed, aborted, or stopped by an error, the output parameter "Busy" changes its value to FALSE.

The output parameter "InVelocity" indicates the status TRUE, as long as the axis is moving at the parameterized velocity. The output parameters "CommandAborted" and "Error" indicate the status for at least one cycle. These status messages are latched as long as either input parameter "JogForward" or "JogBackward" is set to TRUE.

The behavior of the status bits is presented below for various example situations:

The parameterized velocity is reached and maintained

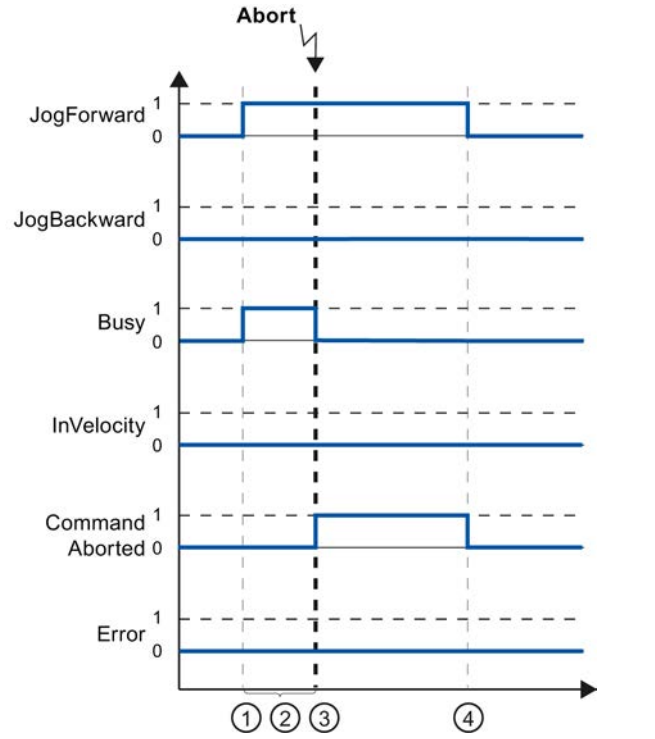
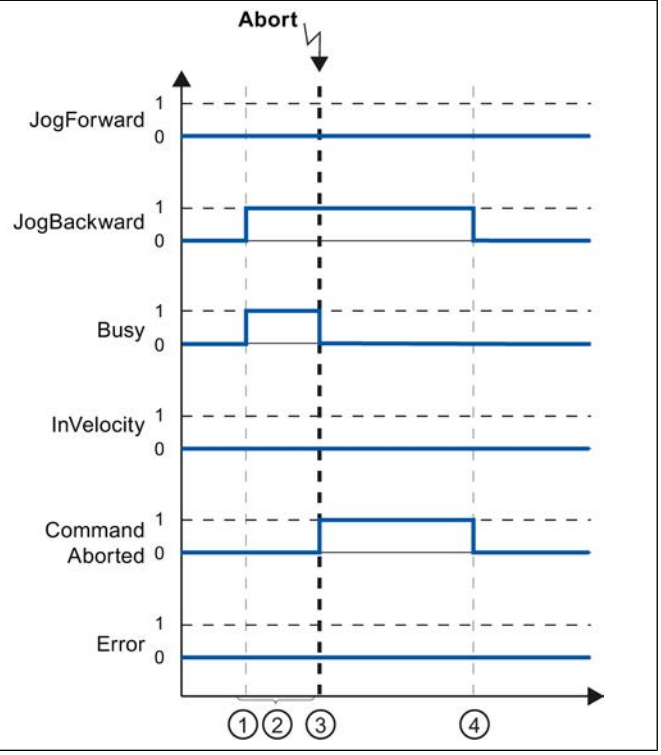
If the motion control command has been executed by the time the parameterized velocity is reached, this is indicated by the value TRUE in output parameter "InVelocity".

Jog mode is controlled by input parameter "JogForward"	Jog mode is controlled by input parameter "JogBackward".
	

①	The command is started with a positive edge at the input parameter "JogForward" or "JogBackward".
②	While the command is active, the output parameter "Busy" indicates the value TRUE.
③	When the parameterized velocity is reached, the output parameter "InVelocity" changes to TRUE.
④	When the input parameter "JogForward" or "JogBackward" is reset to the value FALSE, the axis motion ends. The axis starts to decelerate. As a result, the axis no longer moves at constant velocity and the output parameter "InVelocity" changes its status to FALSE.
⑤	If the axis has come to a standstill, the motion control command is complete and the output parameter "Busy" changes its value to FALSE.

The command is aborted during execution

If the motion control command is aborted during execution, this is indicated by the value TRUE in output parameter "CommandAborted". The behavior is independent of whether or not the parameterized velocity has been reached.

Jog mode is controlled by input parameter "JogForward".	Jog mode is controlled by input parameter "JogBackward".
	

①	The command is started with a positive edge at the input parameter "JogForward" or "JogBackward".
②	While the command is active, the output parameter "Busy" indicates the value TRUE.
③	During command execution, the command is aborted by another motion control command. If the command is aborted, output parameter "Busy" changes to FALSE and "CommandAborted" to TRUE.
④	When the input parameter "JogForward" or "JogBackward" is reset to the value FALSE, the output parameter "CommandAborted" changes its value to FALSE.

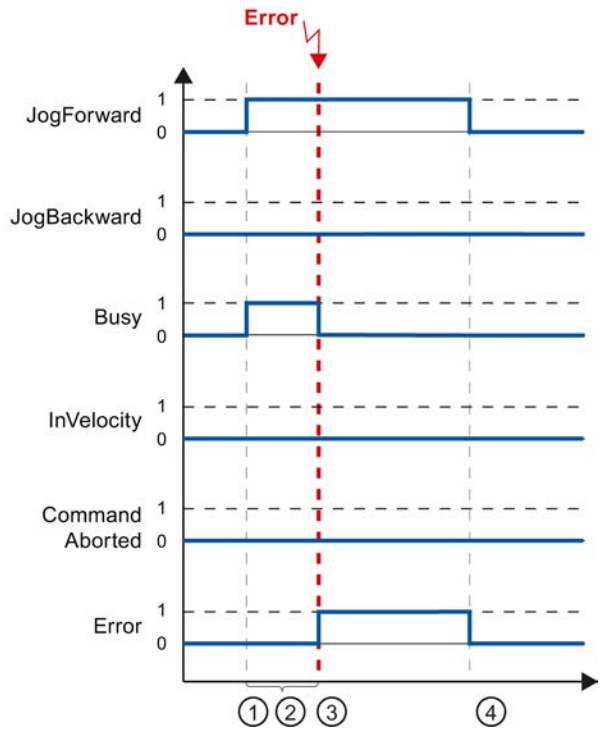
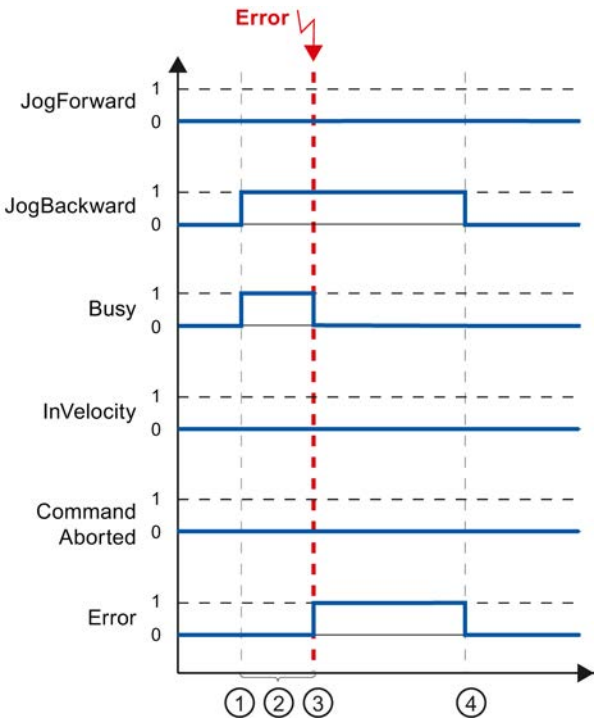
Note

The command abort is indicated in the output parameter "CommandAborted" for only one execution cycle, if all conditions below are met:

The input parameters "JogForward" and "JogBackward" have the value FALSE (but the axis is still decelerating) and a new motion control command is initiated.

An error has occurred during command execution

If an error occurs during execution of the motion control command, this is indicated by the value TRUE in output parameter "Error". The behavior is independent of whether or not the parameterized velocity has been reached.

Jog mode is controlled by input parameter "JogForward".	Jog mode is controlled by input parameter "JogBackward".
	

①	The command is started with a positive edge at the input parameter "JogForward" or "JogBackward".
②	While the command is active, the output parameter "Busy" indicates the value TRUE.

③	An error occurred during command execution. When the error occurs, the output parameter "Busy" changes to FALSE and "Error" to TRUE.
④	When the input parameter "JogForward" or "JogBackward" is reset to the value FALSE, the output parameter "Error" changes its value to FALSE.

Note

An error occurrence is indicated in the output parameter "Error" for only one execution cycle, if all the conditions below are met:

The input parameters "JogForward" and "JogBackward" have the value FALSE (but the axis is still decelerating) and a new error occurs (software limit switch is approached, for example).

Error displays of the Motion Control statements

The Motion Control instructions indicate any errors in motion control commands and the technology object at the output parameters "Error", "ErrorID" and "ErroInfo" of the Motion Control instructions.

Error display at output parameters "Error", "ErrorID" and "ErrorInfo"

If the output parameter "Error" indicates the value TRUE, the complete command, or portions thereof, could not be executed. The cause of the error is indicated by the value in output parameter "ErrorID". Detailed information about the cause of the error is returned by the value in output parameter ErrorInfo. We distinguish between the following error classes for error indication:

- **Operating error with axis stop (for example, "HW limit switch was approached")**
  Operating errors with axis stop are errors that occur during runtime of the user program. If the axis is in motion, it is stopped with the configured deceleration or emergency stop deceleration, depending on the error. The errors are indicated in the error-triggering Motion Control instruction and in the Motion Control instruction "MC_Power".
- **Operating error without axis stop (for example, "Axis is not homed")**
  Operating errors without axis stop are errors that occur during runtime of the user program. If the axis is in motion, the motion is continued. The errors are only indicated in the Motion Control instruction which triggers the error.
- **Configuration error in Motion Control instruction (for example "Incorrect value in parameter "Velocity")**
  Parameterization errors occur when incorrect information is specified in the input parameters of Motion Control instructions. If the axis is in motion, the motion is continued. The errors are only indicated in the Motion Control instruction which triggers the error.

- **Configuration error on technology object "Axis" (for example, "Value for "Acceleration" is invalid")**
  A configuration error exists if one or more parameters are incorrectly configured in the axis configuration or if editable configuration data have been modified incorrectly during runtime of the program. An axis in motion is stopped with the configured emergency stop deceleration. The error is indicated in the error-triggering Motion Control instruction and in Motion Control instruction "MC_Power".
- **Configuration error on technology object "Command table" (for example "Value for "Velocity" is invalid")**
  There is a configuration error if one or more parameters are incorrectly set in the axis command table or if programmable configuration data have been modified incorrectly during runtime of the program. If the axis is in motion, the motion is continued. The errors are only indicated in the "MC_CommandTable" Motion Control instruction.
- **Internal error**
  When an internal error occurs, the axis is stopped. The errors are indicated in the error-triggering Motion Control instruction and, in some cases, in the Motion Control instruction "MC_Power".

A detailed description of the ErrorIDs and ErrorInfos, as well as their remedies, is available in the Appendix (Page 10132).

See also

Overview of the Motion Control statements (Page 10096)

Creating a user program (Page 10097)

Programming notes (Page 10100)

Behavior of the Motion Control commands after POWER OFF and restart (Page 10102)

Monitoring active commands (Page 10102)

List of ErrorIDs and ErrorInfos (technology objects as of V4) (Page 10132)

Axis - Diagnostics

Status and error bits (technology objects as of V4)

You use the "Status and error bits" diagnostic function to monitor the most important status and error messages for the axis in the TIA Portal. The diagnostic function display is available in online mode in "Manual control" mode and in "Automatic control" when the axis is active. The status error messages have the following meaning:

Status messages

Status message - Axis	Description
Enabled	The axis is enabled and ready to be controlled via Motion Control commands. (Tag of technology object: <Axis name>.StatusBits.Enable)
Homed	The axis is homed and is capable of executing absolute positioning commands of Motion Control instruction "MC_MoveAbsolute". The axis does not have to be homed for relative positioning. Special situations: • During active homing, the status is FALSE. • If a homed axis undergoes passive homing, the status is set to TRUE during passive homing. (Tag of technology object: <Axis name>.StatusBits.HomingDone)
Axis error	An error has occurred in the "Axis" technology object. Additional information about the error is available in automatic control at the ErrorID and ErrorInfo parameters of the Motion Control instructions. In manual mode, the "Error message" box of the axis control panel displays detailed information about the cause of error. (Tag of technology object: <Axis name>.StatusBits.Error)
Control panel active	The "Manual control" mode was enabled in the axis control panel. The axis control panel has control priority over the "Axis" technology object. The axis cannot be controlled from the user program. (Tag of technology object: <Axis name>.StatusBits.ControlPanelActive)
Restart necessary	A modified configuration of the axis was downloaded to the load memory in CPU RUN operating mode. To download the modified configuration to the work memory, you need to restart the axis. Use the Motion Control instruction MC_Reset to do this. (Tag of technology object: <Axis name>.StatusBits.RestartRequired)

Status message - Drive	Description
Ready	The drive is ready for operation. (Tag of technology object: <Axis name>.StatusBits.DriveReady)
Drive error	The drive has reported an error due to loss of its "Drive ready" signal. (Tag of technology object: <Axis name>.ErrorBits.DriveFault)

Status message - Motion	Description
Standstill	The axis is at a standstill. (Tag of technology object: <Axis name>.StatusBits.StandStill)
Acceleration	The axis accelerates. (Tag of technology object: <Axis name>.StatusBits.Accelerating)
Constant velocity	The axis travels at constant velocity. (Tag of technology object: <Axis name>.StatusBits.ConstantVelocity)
Deceleration	The axis decelerates (slows down). (Tag of technology object: <Axis name>.StatusBits.Decelerating)

Status message - Motion type	Description
Positioning	The axis executes a positioning command of the Motion Control instruction "MC_MoveAbsolute", "MC_MoveRelative" or the axis control panel. (Tag of technology object: <Axis name>.StatusBits.PositioningCommand)
Travel with velocity specification	The axis executes a command with velocity specification of the Motion Control instruction "MC_MoveVelocity", "MC_MoveJog" or the axis control panel. (Tag of technology object: <Axis name>.StatusBits.VelocityCommand)
Homing	The axis executes a homing command of the Motion Control instruction "MC_Home" or the axis control panel. (Tag of technology object: <Axis name>.StatusBits.HomingCommand)
Command table active	The axis is controlled by Motion Control instruction "MC_CommandTable". (Tag of technology object: <Axis name>.StatusBits.CommandTableActive)

Limit switch status messages

Limit switch status message	Description
Lower SW limit switch has been reached	A software limit switch was reached or exceeded. (Tag of technology object: <Axis name>.StatusBits.SWLimitMinActive)
Upper SW limit switch has been reached	A hardware limit switch was reached or exceeded. (Tag of technology object: <Axis name>.StatusBits.SWLimitMaxActive)
Lower HW limit switch was reached	The lower hardware limit switch was reached or exceeded. (Tag of technology object: <Axis name>.StatusBits.HWLimitMinActive)
Upper HW limit switch was reached	The upper hardware limit switch was reached or exceeded. (Tag of technology object: <Axis name>.StatusBits.HWLimitMaxActive)

Error messages

Error message	Description
SW limit switch has been reached	A software limit switch was reached or exceeded. (Tag of technology object: <Axis name>.ErrorBits.SWLimit)
HW limit switch has been reached	A hardware limit switch was reached or exceeded. (Tag of technology object: <Axis name>.ErrorBits.HWLimit)
Impermissible motion direction	The motion direction of the command does not match the configured motion direction. (Tag of technology object: <Axis name>.ErrorBits.DirectionFault)
PTO already in use	A second axis is using the same PTO (Pulse Train Output) and HSC (High Speed Counter) and is enabled with "MC_Power". (Tag of technology object: <Axis name>.ErrorBits.HWUsed)
Configuration error	The "Axis" technology object was incorrectly configured or editable configuration data were modified incorrectly during runtime of the user program. (Tag of technology object: <Axis name>.ErrorBits.ConfigFault)
Internal error	An internal error has occurred. (Tag of technology object: <Axis name>.ErrorBits.SystemFault)

The output window below shows the first reported and still unacknowledged error.

See also

StatusBits tags as of V4 (Page 10155)

ErrorBits tags as of V4 (Page 10159)

Diagnostics - Status and error bits ("Axis" technology object V1...3) (Page 10174)

Compatibility list of tags (Page 10021)

Motion status (Page 10117)

Motion status

Use the "Motion status" diagnostic function to monitor the motion status of the axis in the TIA Portal. The diagnostic function display is available in online mode in "Manual control" mode and in "Automatic control" when the axis is active. The displayed status information has the following meaning:

Status	Description
Current position	The "Current position" box indicates the current axis position. If the axis is not homed, the value indicates the position value relative to the enable position of the axis. (Tag of technology object: <Axis name>.Position)
Current velocity	The "Current velocity" box indicates the current axis velocity. (Tag of technology object: <Axis name>.Velocity)
Target position	The "Target position" box indicates the current target position of an active positioning command or of the axis command table. The value of the "Target position" is only valid during execution of a positioning command. (Tag of technology object: <Axis name>.StatusPositioning.TargetPosition)
Remaining traversing distance	The "Remaining traversing distance" box indicates the traversing distance currently remaining for an active positioning command or the axis command table. The "Remaining traversing distance" value is only valid during execution of a positioning command. (Tag of technology object: <Axis name>.StatusPositioning.Distance)

See also

Position tag as of V4 (Page 10142)

Velocity tag as of V4 (Page 10142)

StatusPositioning tags as of V4 (Page 10154)

Compatibility list of tags (Page 10021)

Status and error bits (technology objects as of V4) (Page 10114)

Dynamics settings

Use the "Dynamics settings" diagnostic function to monitor the dynamic limits of the axis in the TIA Portal. The diagnostic function display is available in online mode in "Manual control" mode

and in "Automatic control" when the axis is active. The displayed status information has the following meaning:

Dynamic limit	Description
Acceleration	The "Acceleration" box indicates the currently configured acceleration of the axis. (Tag of technology object: <Axis name>.DynamicDefaults.Acceleration)
Deceleration	The "Deceleration" box indicates the currently configured deceleration of the axis. (Tag of technology object: <Axis name>.DynamicDefaults.Deceleration)
Emergency stop deceleration	The "Emergency stop deceleration" box indicates the currently configured emergency stop deceleration of the axis. (Tag of technology object: <Axis name>.DynamicDefaults.EmergencyDeceleration)
Jerk (axis technology object as of V2)	The "Velocity" box indicates the current axis step velocity configured. (Tag of technology object: <Axis name>.DynamicDefaults.Jerk)

See also

DynamicDefaults tags as of V4 (Page 10149)

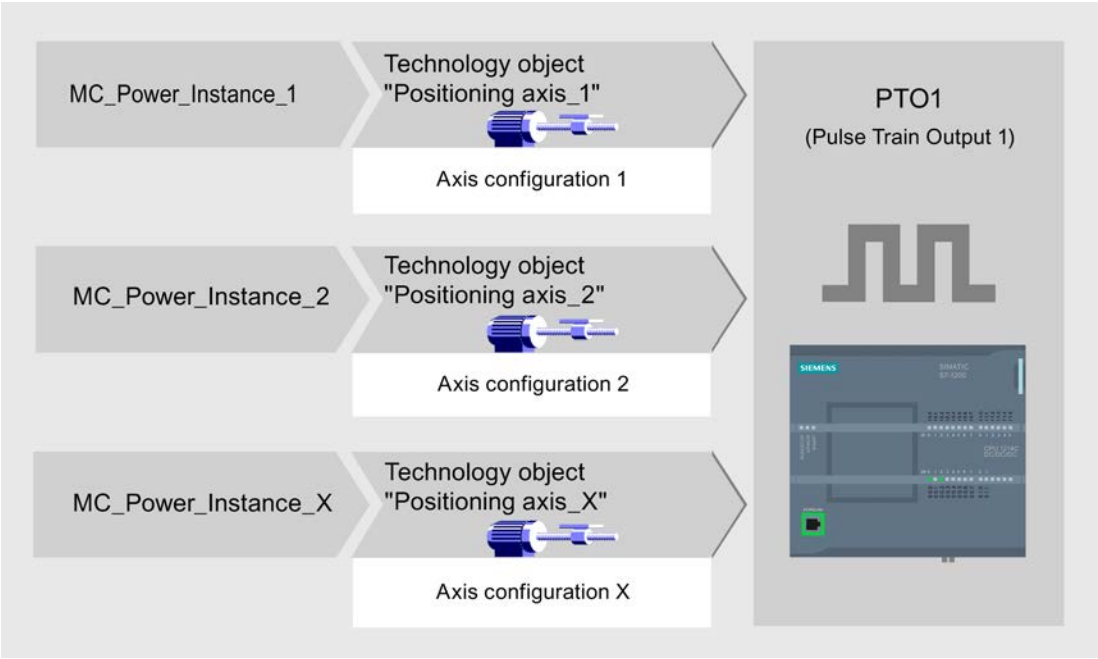
Compatibility list of tags (Page 10021)

Appendix

Using multiple axes with the same PTO

Use the Motion Control functionality of the CPU S7-1200 to run multiple positioning axis technology objects with the same PTO (Pulse Train Output) and thus with the same CPU outputs. This is appropriate, for example, if different axis configurations are to be used for different production sequences via one PTO. As described below, it is possible to switch

between these axis configurations as often as necessary. The following diagram presents the basic functional relationships:

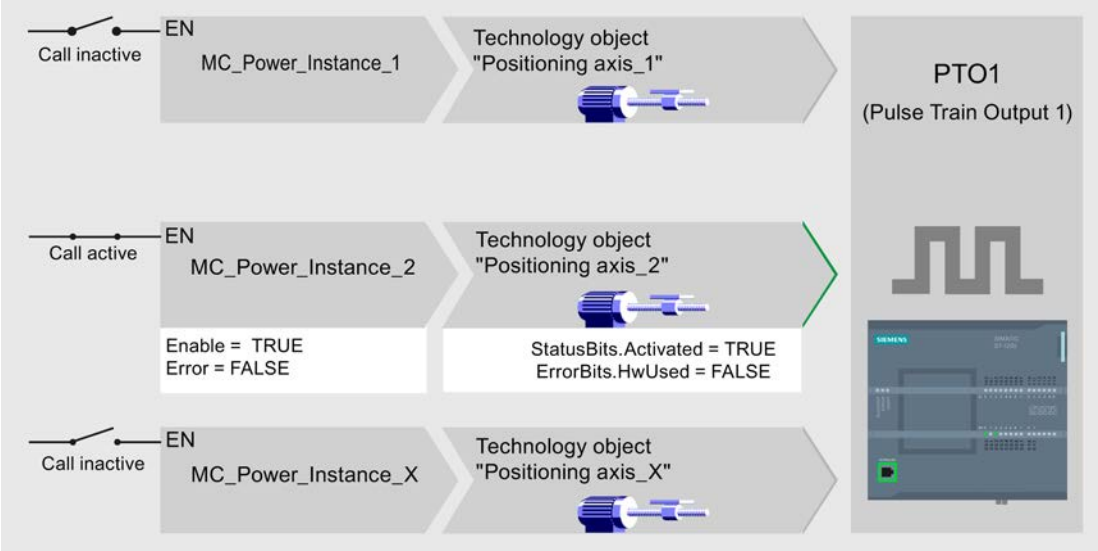


In this example, several positioning axis technology objects, each with its own axis configuration, use the same PTO. Each axis must be called in the user program with a separate call of Motion Control instruction "MC_Power" with a separate instance data block. Only one axis at a time may use the PTO. The axis that is currently using the PTO indicates this with tag <Axis name>.StatusBits.Activated = TRUE.

Switchover of the positioning axis technology object

The program scheme described below shows you how to switch between different technology objects and, thus, between different axis configurations. To use the same PTO with multiple axes without error indications, only the Motion Control instructions of the axis currently being used may be called.

The following diagram presents this principle using Motion Control instruction "MC_Power" as an example:



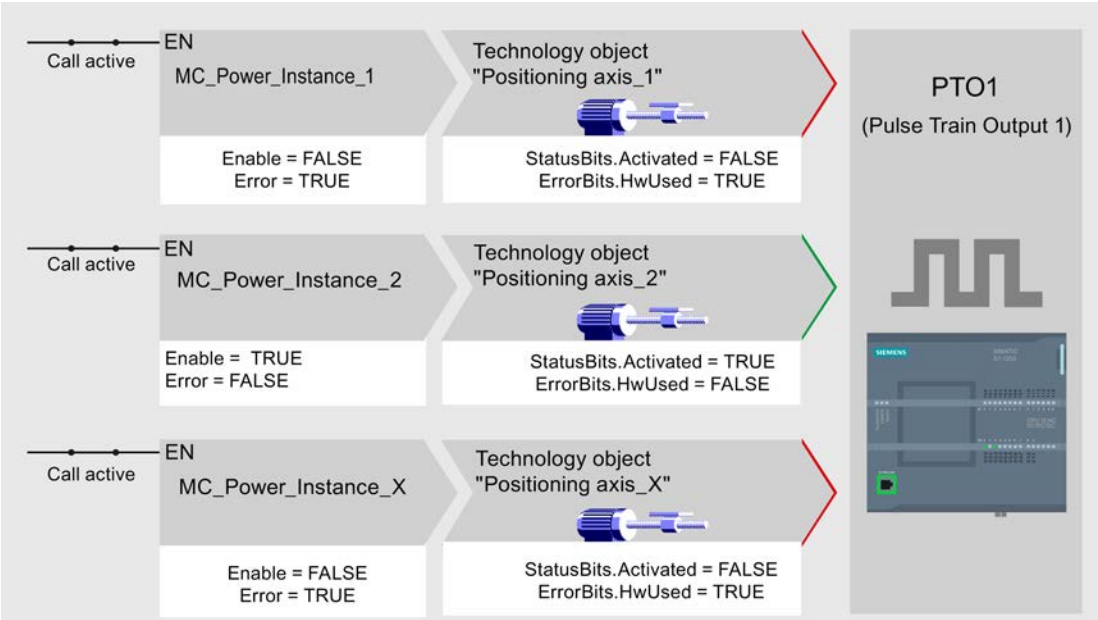
The tags of the activated axis ("Positioning axis_2" here) show the following typical indicators in the user program:

- <Axis name>.StatusBits.Activated = TRUE
- <Axis name>.ErrorBits.HWUsed = FALSE

To switch from one positioning axis technology object to another, follow the steps described below. In the example, a switch is made from "Positioning axis_2" to "Positioning axis_1":

1. End any active traversing motions of activated "Positioning axis_2"
2. Disable "Positioning axis_2" with the associated Motion Control instruction "MC_Power" using input parameter Enable = FALSE
3. To verify that "Positioning axis_2" has been disabled, use an AND operation of output parameter Status = FALSE of Motion Control instruction "MC_Power" and technology object tag <Axis name>.StatusBits.Enable = FALSE.
4. Deactivate the conditional call of the Motion Control instructions for "Positioning axis_2".
5. Activate the conditional call of the Motion Control instruction for "Positioning axis_1". At the first call of the corresponding Motion Control instruction "MC_Power", "Positioning axis_2" is deactivated and "Positioning axis_1" is activated.
6. Enable "Positioning axis_1" with the associated Motion Control instruction "MC_Power" using input parameter Enable = TRUE.
7. To verify that "Positioning axis_1" has been enabled, use an AND operation of output parameter Status = TRUE of Motion Control instruction "MC_Power" and technology object tag <Axis name>.StatusBits.Enable = TRUE.

It is also always possible to cyclically call all Motion Control instructions of all axes working with a single PTO.



When an axis is enabled (here "Positioning axis_2"), this axis becomes active.

In contrast to the conditional call, the Motion Control instructions of the deactivated axes (here "Positioning axis_1" and "Positioning axis_x") indicate errors. The tags of these axes indicate the status <Axis name>.StatusBits.Activated = FALSE and <Axis name>.ErrorBits.HWUsed = TRUE.

Use the conditional call of the Motion Control instructions if you want to implement the user program without error indicators.

See also

Tags of the positioning axis technology object as of V4 (Page 10142)

Using multiple drives with the same PTO (Page 10122)

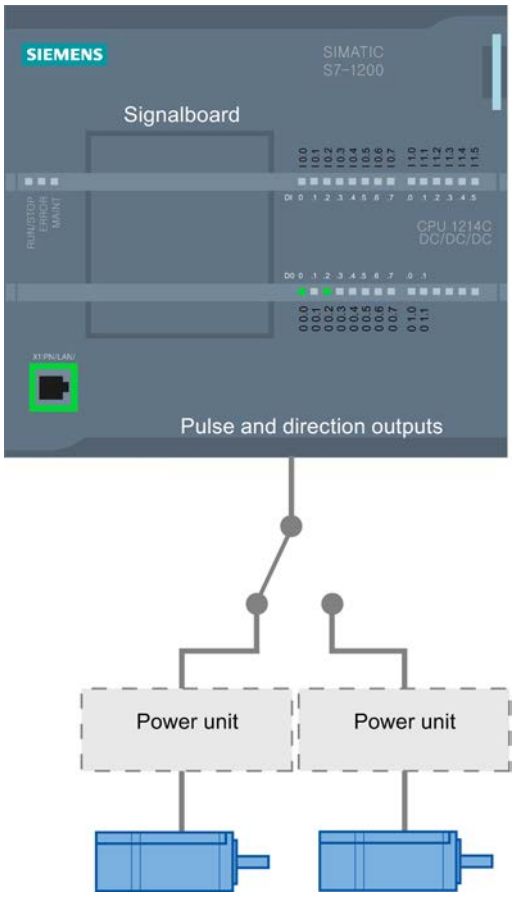
Tracking jobs from higher priority classes (execution levels) (Page 10123)

Special cases for use of software limit switches (Page 10125)

List of ErrorIDs and ErrorInfos (technology objects as of V4) (Page 10132)

Using multiple drives with the same PTO

If multiple drives are to be used, they can be run with a common PTO (Pulse Train Output) using changeover. The following diagram represents the basic circuit design:



The changeover between drives can be controlled, if required, by the user program via a digital output. If different axis configurations are required for the different drives, a changeover between these configurations is required for the PTO. For additional information on this topic, refer to "Using multiple axes with the same PTO (Page 10118)".

See also

Tags of the positioning axis technology object as of V4 (Page 10142)

Using multiple axes with the same PTO (Page 10118)

Tracking jobs from higher priority classes (execution levels) (Page 10123)

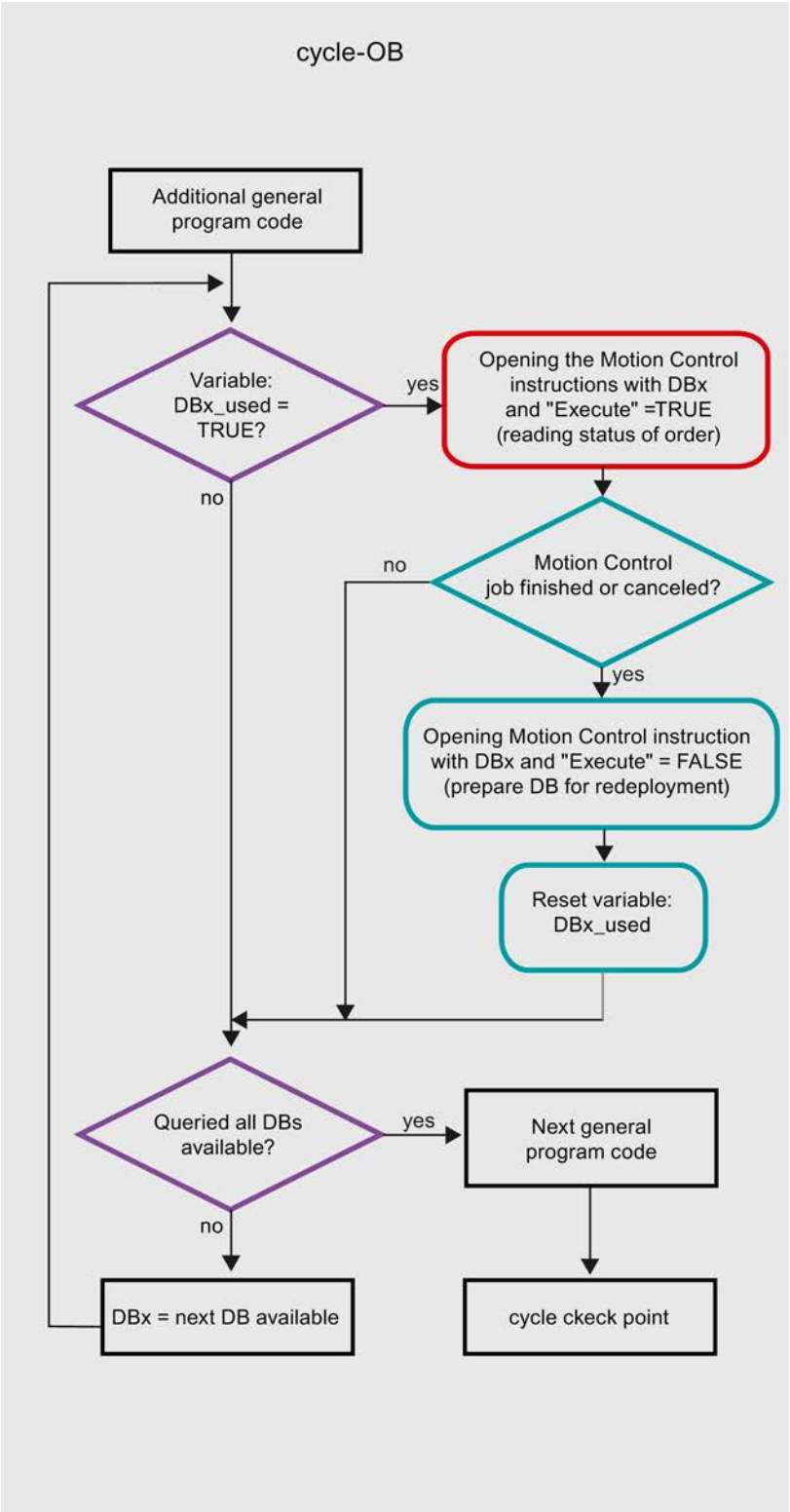
Special cases for use of software limit switches (Page 10125)

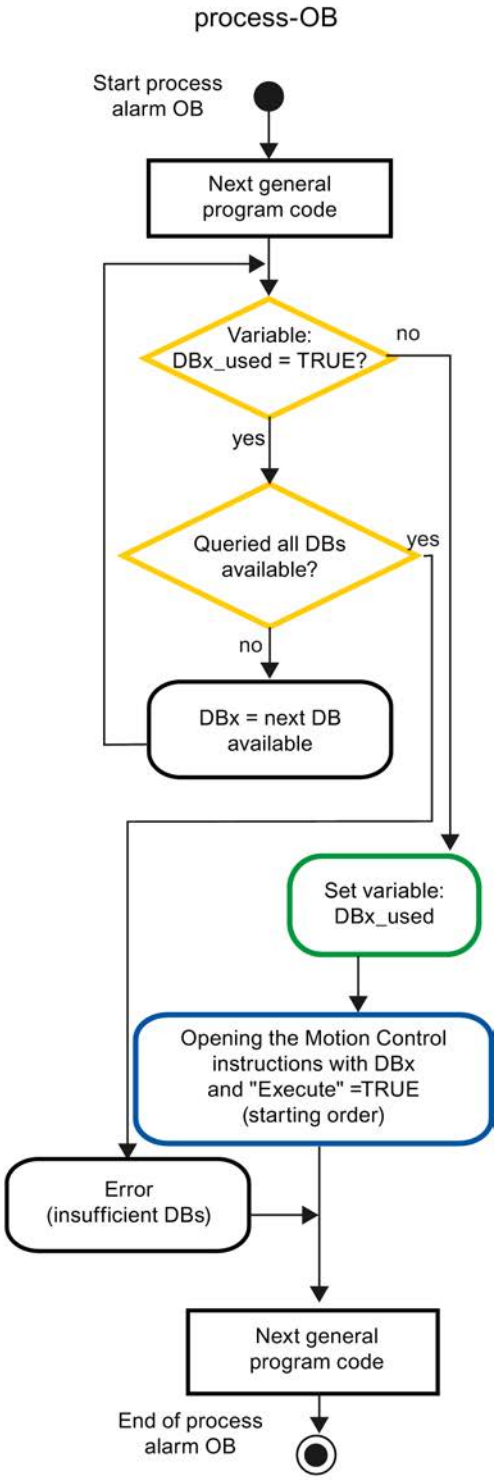
List of ErrorIDs and ErrorInfos (technology objects as of V4) (Page 10132)

Tracking jobs from higher priority classes (execution levels)

Depending on the application, it may be necessary to start Motion Control commands (for example, interrupt-controlled) in a higher priority class (execution level).

The Motion Control instructions must be called at short intervals for status monitoring. Motion Control commands cannot be sufficiently closely monitored if the higher priority Motion Control commands are called only once or at too great an interval. Tracking in such cases can be carried out in the cycle OB. An instance data block that is not currently being utilized must be available for each start of a higher priority Motion Control command. Refer to the following flow chart to see how you start Motion Control commands in a higher priority class (for example, hardware interrupt OB) and continue tracking in the program cycle OB:





Depending on the frequency of the Motion Control commands you want to start, you will have to generate a sufficient number of instance data blocks. Users determine which instance data block is currently used in the DBx_used tags.

Start of Motion Control command in the hardware interrupt OB

Binary queries of the DBx_used tags (orange) are used to find an instance data block not currently in use. If such an instance data block is found, the utilized instance data block is marked as "used" (green) and the Motion Control command is started with this instance data block (blue).

Any other program sections of the hardware interrupt OB are then executed, followed by a return to the program cycle OB.

Tracking of started Motion Control commands in the program cycle OB

All instance data blocks available in the cycle OB are checked to determine if they are currently in use by means of the DBx_used tag (violet).

If an instance data block is in use (Motion Control command is being processed), the Motion Control instruction with this instance data block and input parameter Execute = TRUE is called to read out the status messages (red).

If the command is complete or has been aborted, the following actions are taken next (blue green):

- Call of Motion Control instruction with input parameter Execute = FALSE
- Resetting the DBx_used tag

This completes the command tracking, and the instance data block is now available for use again.

See also

Tags of the positioning axis technology object as of V4 (Page 10142)

Using multiple axes with the same PTO (Page 10118)

Using multiple drives with the same PTO (Page 10122)

Special cases for use of software limit switches (Page 10125)

List of ErrorIDs and ErrorInfos (technology objects as of V4) (Page 10132)

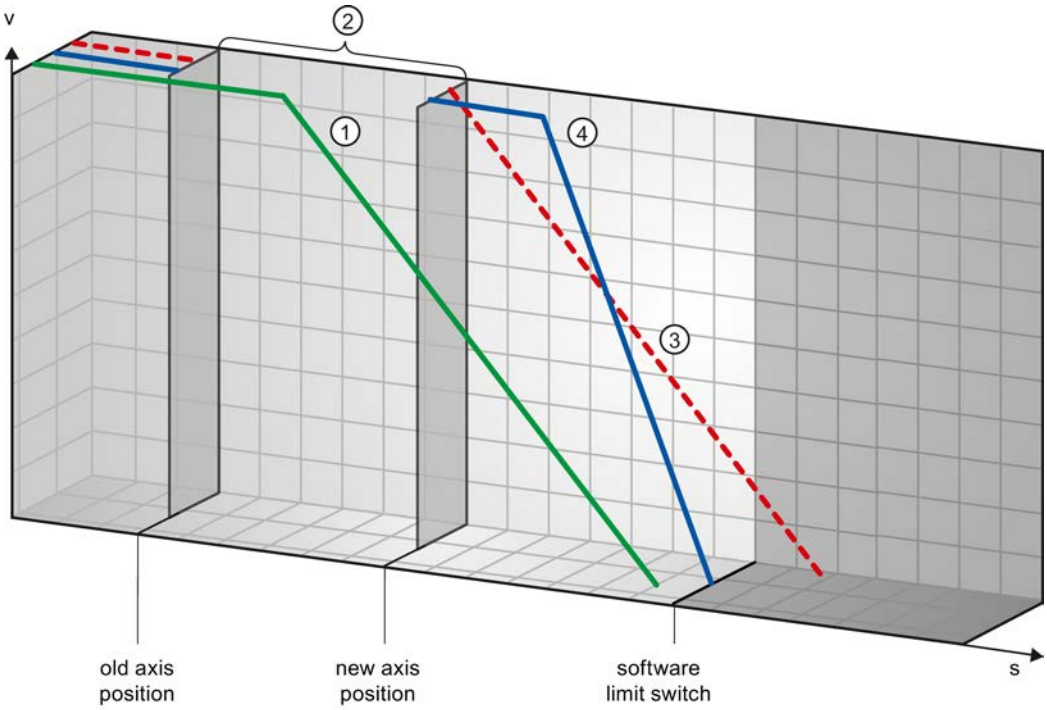
Special cases for use of software limit switches

Software limit switches in conjunction with a homing operation

Due to unfavorably parameterized homing jobs, the braking action of the axis may be influenced at the software limit switch. Take the following examples into consideration when developing your program.

Example 1:

During a travel command, a homing job (for example, Set reference point) offsets the current axis position in the direction of the software limit switch. It is still possible to bring the axis to a standstill before reaching the software limit switch:

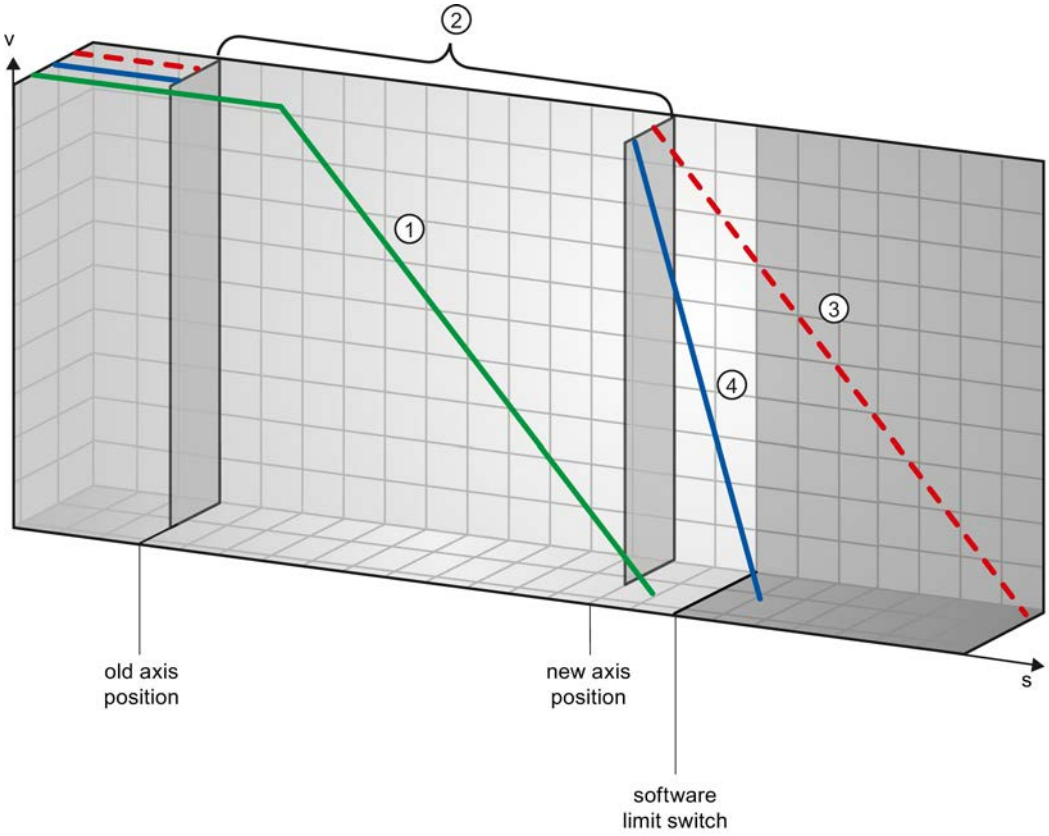


①	The green curve shows the motion **without** the homing job. The axis brakes at the configured deceleration and comes to a standstill at a position before the software limit switch.
②	A new axis position is set as a result of the homing job. The area between the old and new axis position is thus "skipped".
③	Based on the new axis position, the axis would theoretically be stopped with the configured deceleration at a position after the software limit switch (red curve).
④	Because braking with the configured deceleration is no longer sufficient, the axis actually follows the blue curve. Following constant motion, the axis brakes at the configured emergency stop deceleration and comes to a standstill at the position of the software limit switch.

Example 2:

During a travel command, a homing job (for example, Set reference point) offsets the current axis position in the direction of the software limit switch. In contrast to example 1, it is no longer

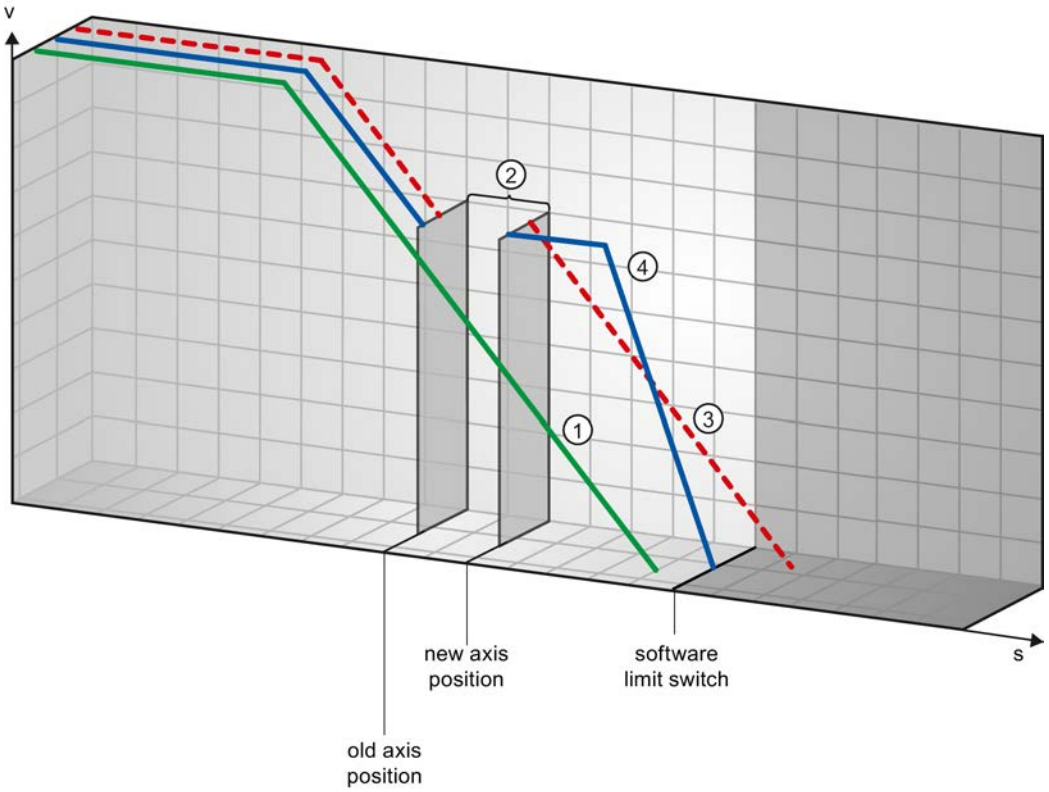
possible to bring the axis to a standstill before reaching the software limit switch. The axis overruns the position of the software limit switch.



①	The green curve shows the motion **without** the homing job. The axis brakes at the configured deceleration and comes to a standstill at a position before the software limit switch.
②	A new axis position is set as a result of the homing job. The area between the old and new axis position is thus "skipped".
③	Based on the new axis position, the axis would theoretically be stopped with the configured deceleration at a position well after the software limit switch (red curve).
④	Because braking with the configured deceleration is no longer sufficient, the axis actually follows the blue curve. The axis brakes with the configured emergency stop deceleration. However, the emergency stop deceleration is not sufficient to stop the axis at the position of the software limit switch. The position of the software limit switch is overrun.

Example 3:

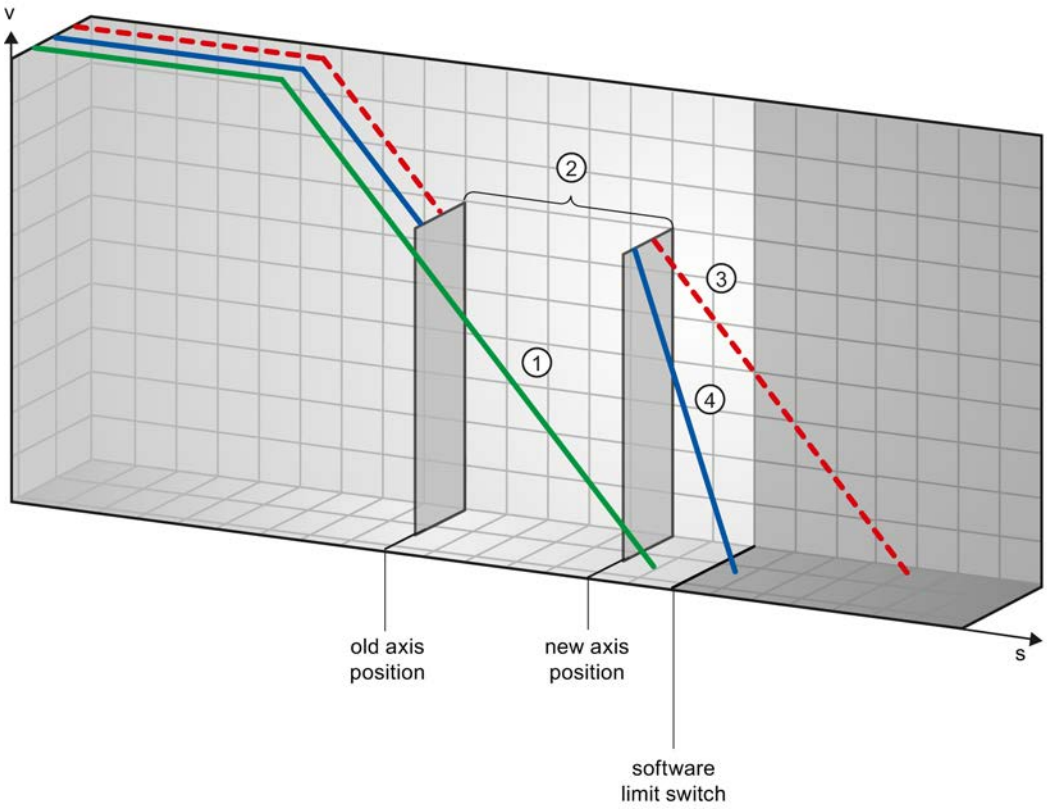
During a braking operation, a homing job (for example, Set reference point) offsets the current axis position in the direction of the software limit switch. It is still possible to bring the axis to a standstill before reaching the software limit switch:



①	The green curve shows the motion **without** the homing job. The axis brakes at the configured deceleration and comes to a standstill at a position before the software limit switch.
②	A new axis position is set as a result of the homing job. The area between the old and new axis position is thus "skipped".
③	Based on the new axis position, the axis would theoretically be stopped with the configured deceleration at a position after the software limit switch (red curve).
④	Because braking with the configured deceleration is no longer sufficient, the axis actually follows the blue curve. Following constant motion, the axis brakes at the configured emergency stop deceleration and comes to a standstill at the position of the software limit switch.

Example 4:

During a braking operation, a homing job (for example, Set reference point) offsets the current axis position in the direction of the software limit switch. In contrast to example 3, it is no longer possible to bring the axis to a standstill before reaching the software limit switch. The axis overruns the position of the software limit switch.



①	The green curve shows the motion **without** the homing job. The axis brakes at the configured deceleration and comes to a standstill at a position before the software limit switch.
②	A new axis position is set as a result of the homing job. The area between the old and new axis position is thus "skipped".
③	Based on the new axis position, the axis would theoretically be stopped with the configured deceleration at a position well after the software limit switch (red curve).
④	Because braking with the configured deceleration is no longer sufficient, the axis actually follows the blue curve. The axis brakes with the configured emergency stop deceleration. However, the emergency stop deceleration is not sufficient to stop the axis at the position of the software limit switch. The position of the software limit switch is overrun.

See also

Software limit switches and software limit switch position changes. (Page 10129)

Software limit switches in conjunction with dynamic changes (Page 10130)

Behavior of axis when position limits is tripped (Page 10039)

Software limit switches and software limit switch position changes.

An incorrect change in the position of the software limit switch during the runtime of the user program can abruptly reduce the distance between the current axis position and the position of the software limit switch.

The axis response is similar to that described in Software limit switches in conjunction with a homing operation (Page 10125).

See also

Software limit switches in conjunction with a homing operation (Page 10125)

Software limit switches in conjunction with dynamic changes (Page 10130)

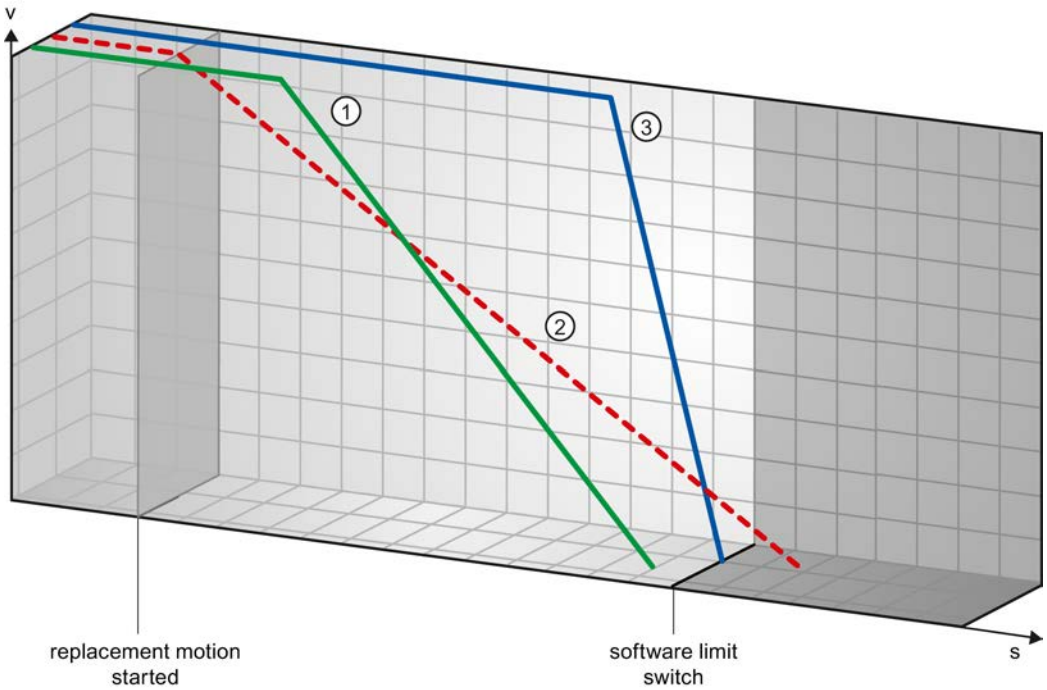
Behavior of axis when position limits is tripped (Page 10039)

Software limit switches in conjunction with dynamic changes

It is possible to influence the deceleration of the axis in the area of the software limit switches in conjunction with overriding motion commands. This applies when the overriding motion command is started with a lower deceleration (tag <Axis name>.DynamicDefaults.Deceleration). Take the following examples into consideration when developing your program.

Example 1:

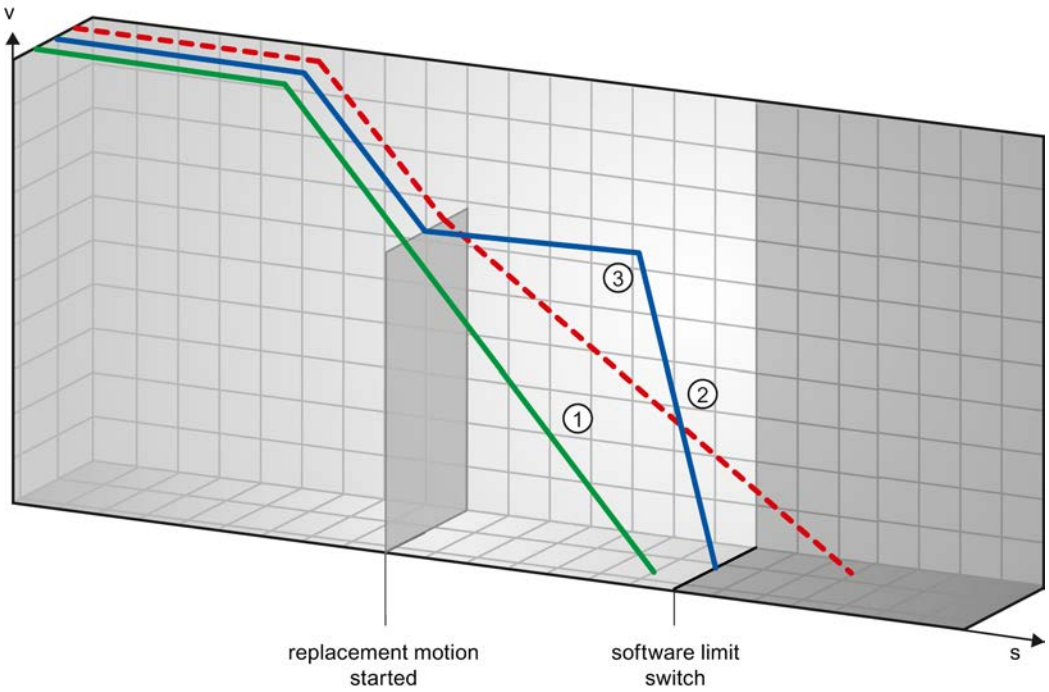
During axis motion, an active motion command is overridden by another motion command with a lower deceleration:



①	The green curve shows the motion of an active command **without** this command being overridden. The axis brakes at the configured deceleration and comes to a standstill at a position before the software limit switch.
②	Based on the overriding motion command with lower deceleration, the axis would theoretically be stopped with the configured deceleration at a position after the software limit switch (red curve).
③	Because braking with the configured deceleration of the overriding motion command is no longer sufficient, the axis actually follows the blue curve. Following a constant motion, the axis brakes at the emergency stop deceleration and comes to a standstill at the position of the software limit switch.

Example 2:

During braking of the axis, an active motion command is overridden by another motion command with a lower deceleration:



①	The green curve shows the motion of an active command **without** this command being overridden. The axis brakes at the configured deceleration and comes to a standstill at a position before the software limit switch.
②	Based on the overriding motion command with lower deceleration, the axis would theoretically be stopped at a position well after the software limit switch (red curve).
③	Because braking with the configured deceleration of the overriding motion command is no longer sufficient, the axis actually follows the blue curve. Following a constant motion, the axis brakes at the emergency stop deceleration and comes to a standstill at the position of the software limit switch.

See also

Software limit switches in conjunction with a homing operation (Page 10125)

Software limit switches and software limit switch position changes. (Page 10129)

Behavior of axis when position limits is tripped (Page 10039)

Reducing velocity for a short positioning duration

The CPU can reduce the velocity of a positioning command when the planned positioning duration is < 2 ms.

The velocity of command execution will then be reduced for the entire duration. The reduced velocity (pulses per s) is calculated as follows:

- Reduced velocity = Number of pulses to be output * 500Hz

Velocity is **not** reduced if the planned positioning duration is >= 2 ms.

Dynamic adjustment of start/stop velocity

The configuration of your velocity limits (start/stop velocity and maximum velocity), the dynamic values (acceleration, deceleration and jerk) and the target speed of the traversing command may under certain circumstances result in the start/stop velocity being dynamically adjusted by the CPU.

This is for example the case when, due to a low configured start/stop velocity, the time required for the first pulses would be longer than that possible for the entire acceleration. The first pulse is in these cases output at a greater velocity than the configured start/stop velocity. The subsequent pulses are also dynamically adjusted to ensure the acceleration process can be completed in the specified time.

Ensure in the event of any pulse loss that the hardware (drive) you use is adjusted to this situation, or change the dynamic settings of your axis to avoid dynamic adjustment of the start/stop velocity.

List of ErrorIDs and ErrorInfos (technology objects as of V4)

The following table lists all ErrorIDs and ErrorInfos that can be indicated in Motion Control instructions. In addition to the cause of the error, remedies for eliminating the error are also listed:

Operating error with axis stop

ErrorID	ErrorInfo	Description	Remedy
16#8000		**Drive error, loss of "Drive ready"**	
	16#0001	-	Acknowledge error with instruction "MC_Reset"; provide drive signal; restart command, if necessary
16#8001		**Lower SW limit switch has been tripped**	

ErrorID	ErrorInfo	Description	Remedy
	16#000E	The position of the lower SW limit switch was reached with the currently configured deceleration	Acknowledge the error with instruction "MC_Reset"; use a motion command to move the axis in the positive direction out of the range of the SW limit switch
	16#000F	The position of the lower SW limit switch was reached with the emergency stop deceleration	
	16#0010	The position of the lower SW limit switch was exceeded with the emergency stop deceleration	
16#8002		Upper SW limit switch has been tripped	
	16#000E	The position of the upper SW limit switch was reached with the currently configured deceleration	Acknowledge the error with instruction "MC_Reset"; use a motion command to move the axis in the negative direction out of the range of the SW limit switch
	16#000F	The position of the upper SW limit switch was reached with the emergency stop deceleration	
	16#0010	The position of the upper SW limit switch was exceeded with the emergency stop deceleration	
16#8003		Lower HW limit switch was reached	
	16#000E	The lower HW limit switch was reached. The axis was stopped with the emergency stop deceleration. (During an active homing procedure, the homing switch was not found)	Acknowledge the error for an enabled axis with instruction "MC_Reset"; use a motion command to move the axis in the positive direction out of the range of the HW limit switch.
16#8004		Upper HW limit switch was reached	
	16#000E	The upper HW limit switch has been reached. The axis was stopped with the emergency stop deceleration. (During an active homing procedure, the homing switch was not found)	Acknowledge the error for an enabled axis with instruction "MC_Reset"; use a motion command to move the axis in the negative direction out of the range of the HW limit switch.
16#8005		PTO/HSC are already being used by another axis	
	16#0001	-	The axis was configured incorrectly: Correct the configuration of the PTO (Pulse Train Output) / HSC (High Speed Counter) and download it to the controller
			More than one axis is to run with one PTO: Another axis is using the PTO / HSC. If the current axis is to assume the control, the other axis must be disabled with "MC_Power" Enable = FALSE. (see also Using multiple axes with the same PTO (Page 10118))
16#8006		A communication error in the control panel has occurred	
	16#0012	A timeout has occurred	Check the cable connection and press the "Manual control" button again.
16#8007		The axis cannot be enabled.	
	16#0025	Restarting	Wait until the axis restart is complete.
	16#0026	Executing loading process in RUN mode	Wait until the loading process is complete.
16#8008		Impermissible motion direction	

ErrorID	ErrorInfo	Description	Remedy
	16#002E	The selected motion direction is not allowed.	• Adjust the motion direction and restart the command. • Adjust the allowed direction of rotation in the technology object configuration under "Extended parameters > Mechanics" Restart the command.
	16#002F	A reversing motion is not possible with the selected motion direction.	

Operating error without axis stop

ErrorID	ErrorInfo	Description	Remedy
16#8200		Axis is not enabled	
	16#0001	-	Enable the axis; restart the command
16#8201		Axis has already been enabled by another "MC_Power" instance	
	16#0001	-	Enable the axis with only one "MC_Power" instance
16#8202		The maximum number of simultaneously active Motion Control commands has been exceeded (maximum of 200 commands for all motion control technology objects)	
	16#0001	-	Reduce the number of simultaneously active commands; restart the command A command is active if parameter "Busy" = TRUE in the Motion Control instruction.
16#8203		Axis is currently operated in "Manual control" (axis control panel)	
	16#0001	-	Exit "Manual control"; restart the command
16#8204		Axis is not homed	
	16#0001	-	Home the axis with instruction "MC_Home"; restart the command
16#8205		The axis is currently controlled by the user program (the error is only displayed in the axis control panel)	
	16#0013	The axis is enabled in the user program.	Disable axis with instruction "MC_Power" and select "Manual control" again in the axis control panel
16#8206		Technology object not activated yet	
	16#0001	-	Enable the axis with instruction "MC_Power" Enable = TRUE or enable the axis in the axis control panel.
16#8207		Command rejected	
	16#0016	Active homing is running; another homing method cannot be started.	Wait for active homing to finish or abort the active homing with a motion command, for example, "MC_Halt".
	16#0018	The axis cannot be moved with a command table while it is being actively or passively homed.	Wait until direct or passive homing is complete.
	16#0019	The axis cannot be actively or passively homed while a command table is being processed.	Wait for command table to finish or abort the command table with a motion command, for example, "MC_Halt".
16#8208		Difference between maximum and start/stop velocity is invalid	
	16#0002	Number format of value is invalid	Correct the value; restart the command

ErrorID	ErrorInfo	Description	Remedy
	16#000A	Value is less than or equal to 0	
16#8209		**Invalid acceleration for technology object "Axis"**	
	16#0002	Number format of value is invalid	Correct the value; restart the command
	16#000A	Value is less than or equal to 0	
16#820A		**It is not possible to restart the axis**	
	16#0013	The axis is enabled in the user program.	Disable the axis with the "MC_Power" instruction; restart again
	16#0027	The axis is currently being operated in "Manual control" (axis control panel)	Exit "Manual control"; restart again
16#820B		**It is not possible to execute the command table**	
	16#0026	Executing loading process in RUN mode	Wait until the loading process is complete.

Block parameter error

ErrorID	ErrorInfo	Description	Remedy
16#8400		**Invalid value at parameter "Position" of the Motion Control instruction**	
	16#0002	Number format of value is invalid	Correct the value; restart the command
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#0006	The value is outside the number range (less than $-1e^{12}$)	
16#8401		**Invalid value at parameter "Distance" of the Motion Control instruction**	
	16#0002	Number format of value is invalid	Correct the value; restart the command
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#0006	The value is outside the number range (less than $-1e^{12}$)	
16#8402		**Invalid value at parameter "Velocity" of the Motion Control instruction**	
	16#0002	Number format of value is invalid	Correct the value; restart the command
	16#0008	Value is greater than the configured maximum velocity	
	16#0009	Value is less than the configured start/stop velocity	
	16#0024	Value is less than 0	
16#8403		**Invalid value at parameter "Direction" of the Motion Control instruction**	
	16#0011	The selection value is invalid	Correct the selection value; restart the command
16#8404		**Invalid value at parameter "Mode" of the Motion Control instruction**	
	16#0011	The selection value is invalid	Correct the selection value; restart the command
	16#0015	Active/passive homing is not configured	Correct the configuration and download it to the controller; enable the axis and restart the command

ErrorID	ErrorInfo	Description	Remedy
	16#0017	The direction reversal is activated at the hardware limit switch, despite the fact that the hardware limit switches are disabled	• Activate the HW limit switch using the tag <Axis name>.PositionLimitsHW.Active = TRUE, restart the command • Correct the configuration and download it to the controller; enable the axis and restart the command
16#8405		Invalid value at parameter "StopMode" of the Motion Control instruction	
	16#0011	The selection value is invalid	Correct the selection value; enable the axis again
16#8406		Simultaneous forward and backward jogging is not allowed	
	16#0001	-	Take steps to ensure that parameters "JogForward" and "JogBackward" do not have signal status TRUE simultaneously; restart the command.
16#8407		Switching to another axis with instruction "MC_Power" is only permitted after disabling the active axis.	
	16#0001	-	Disable the active axis; it is then possible to switch to the other axis and enable it.
16#8408		Invalid value at parameter "Axis" of the Motion Control instruction	
	16#001A	The specified value does not match the required technology object version	Correct the value; restart the command
	16#001B	The specified value does not match the required technology object type	
	16#001C	The specified value is not a Motion Control technology data block	
16#8409		Invalid value at parameter "CommandTable" of the Motion Control instruction	
	16#001A	The specified value does not match the required technology object version	Correct the value; restart the command
	16#001B	The specified value does not match the required technology object type	
	16#001C	The specified value is not a Motion Control technology data block	
16#840A		Invalid value at parameter "StartStep" of the Motion Control instruction	
	16#000A	Value is less than or equal to 0	Correct the value; restart the command
	16#001D	The start step is greater than the end step	
	16#001E	Value is greater than 32	
16#840B		Invalid value at parameter "EndStep" of the Motion Control instruction	
	16#000A	Value is less than or equal to 0	Correct the value; restart the command
	16#001E	Value is greater than 32	
16#840C		Invalid value at parameter "RampUpTime" of the Motion Control instruction	
	16#0002	Number format of value is invalid	Correct the value; restart the command
	16#000A	Value is less than or equal to 0	
16#840D		Invalid value at parameter "RampDownTime" of the Motion Control instruction	
	16#0002	Number format of value is invalid	Correct the value; restart the command
	16#000A	Value is less than or equal to 0	
16#840E		Invalid value at parameter "EmergencyRampTime" of the Motion Control instruction	
	16#0002	Number format of value is invalid	Correct the value; restart the command
	16#000A	Value is less than or equal to 0	

ErrorID	ErrorInfo	Description	Remedy
16#840F		Invalid value at parameter "JerkTime" of the Motion Control instruction	
	16#0002	Number format of value is invalid	Correct the value; restart the command
	16#000A	Value is less than or equal to 0	
16#8410		Invalid value at parameter "Parameter" of the Motion Control instruction	
	16#0028	Data types of VARIANT pointer "Parameter" and "Value" do not match.	Use a suitable data type; restart command
	16#0029	VARIANT pointer "Parameter" does not point to a data block of the technology object.	Correct the VARIANT pointer; restart the command
	16#002A	The value at the VARIANT pointer "Parameter" cannot be read.	Correct the VARIANT pointer; restart the command
	16#002B	The value at the VARIANT pointer "Parameter" cannot be written.	Correct the VARIANT pointer or value; restart the command
	16#002C	The axis is not disabled.	Disable the axis; restart the command
16#8411		Invalid value at parameter "Value" of the Motion Control instruction	
	16#0002	Number format of value is invalid	Correct the value; restart the command

Configuration error of the axis

ErrorID	ErrorInfo	Description	Remedy
16#8600		Parameter assignment of pulse generator (PTO is invalid	
	16#000B	The address is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0014	The selected hardware is used by another application	
16#8601		Parameter assignment of the high-speed counter (HSC) is invalid	
	16#000B	The address is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0014	The selected hardware is used by another application	
16#8602		Invalid parameter assignment of "Enable output"	
	16#000B	The address is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8603		Invalid parameter assignment of "Ready input"	
	16#000B	The address is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8604		Invalid "Pulses per motor revolution" value	
	16#000A	Value is less than or equal to zero	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8605		Invalid "Distance per revolution" value	
	16#0002	Number format of value is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#000A	Value is less than or equal to zero	
16#8606		Invalid "Start/stop velocity" value	
	16#0002	Number format of value is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0003	Value exceeds the upper hardware limit	

ErrorID	ErrorInfo	Description	Remedy
	16#0004	Value is less than the lower hardware limit	
	16#0007	The start/stop velocity is greater than the maximum velocity	
16#8607		Invalid "maximum velocity" value	
	16#0002	Number format of value is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0003	Value exceeds the upper hardware limit	
	16#0004	Value is less than the lower hardware limit	
16#8608		Invalid "Acceleration" value	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0003	Value exceeds the upper hardware limit	
	16#0004	Value is less than the lower hardware limit	
16#8609		Invalid "Deceleration" value	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0003	Value exceeds the upper hardware limit	
	16#0004	Value is less than the lower hardware limit	
16#860A		Invalid "Emergency stop deceleration" value	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0003	Value exceeds the upper hardware limit	
	16#0004	Value is less than the lower hardware limit	
16#860B		Value for position of the lower SW limit switch is invalid	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#0006	The value is outside the number range (less than $-1e^{12}$)	
	16#0030	The position value of the lower software limit switch is greater than that of the upper software limit switch	
16#860C		Value for position of the upper SW limit switch is invalid	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#0006	The value is outside the number range (less than $-1e^{12}$)	
16#860D		Invalid address of the lower HW limit switch	
	16#000B	Invalid address	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#000C	The address of the falling edge is invalid	
	16#000D	The address of the rising edge is invalid	

ErrorID	ErrorInfo	Description	Remedy
16#860E		Invalid address of the upper HW limit switch	
	16#000B	Invalid address	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#000C	The address of the falling edge is invalid	
	16#000D	The address of the rising edge is invalid	
16#860F		Invalid "home position offset" value	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#0006	The value is outside the number range (less than $-1e^{12}$)	
16#8610		Invalid "approach velocity" value	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0008	The velocity is greater than the maximum velocity	
	16#0009	The velocity is less than the start/stop velocity	
16#8611		Invalid "Homing velocity" value	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0008	The velocity is greater than the maximum velocity	
	16#0009	The velocity is less than the start/stop velocity	
16#8612		Invalid address of the homing switch	
	16#000B	Invalid address	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#000C	The address of the falling edge is invalid	
	16#000D	The address of the rising edge is invalid	
16#8613		During active homing, direction reversal at the hardware limit switch is activated although the hardware limit switches are not configured	
	16#0001	-	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8614		Invalid "Jerk" value	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#001F	Value is greater than the maximum jerk	
	16#0020	Value is less than the minimum jerk	
16#8615		Value for "Unit of measurement" is invalid	
	16#0011	The selection value is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8616		Address of homing switch is invalid (passive homing as of V4)	

ErrorID	ErrorInfo	Description	Remedy
	16#0011	The selection value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8617		Value of tag <Axis name>.Sensor.Sensor[1].ActiveHoming.Mode is invalid	
	16#0011	The selection value is invalid (Valid value: 2 = Homing via digital input)	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8618		Value of tag <Axis name>.Sensor.Sensor[1].PassiveHoming.Mode is invalid	
	16#0011	The selection value is invalid (Valid value: 2 = Homing via digital input)	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8619		Value of tag <Axis name>.Actor.Type is invalid	
	16#0011	The selection value is invalid (Valid value: 2 = Connection via pulse interface)	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#861A		Value for "Permitted direction of rotation" is invalid	
	16#0011	The selection value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#002D	"Both directions" not allowed when direction output is deactivated	

Configuration error of the command table

ErrorID	ErrorInfo	Description	Remedy
16#8700		Value for "Command type" in the command table is invalid	
	16#0001	-	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and restart the command, if necessary
16#8701		Value for "Position / travel path" in the command table is invalid	

ErrorID	ErrorInfo	Description	Remedy
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and restart the command, if necessary
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#0006	The value is outside the number range (less than $-1e^{12}$)	
16#8702		Value for "Velocity" in the command table is invalid	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and restart the command, if necessary
	16#0008	Value is greater than the configured maximum velocity	
	16#0009	Value is less than the configured start/stop velocity	
16#8703		Value for "Duration" in the command table is invalid	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and restart the command, if necessary
	16#0021	Value is greater than 64800 s	
	16#0022	Value is less than 0.001 s	
16#8704		Value for "Next step" in the command table is invalid	
	16#0011	The selection value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and restart the command, if necessary
	16#0023	The command transition is not permitted for this command	

Internal errors

ErrorID	ErrorInfo	Description	Remedy
16#8FFF		Internal error	
	16#F0**	-	POWER OFF and POWER ON the CPU If this does not work, contact Customer Support. Have the following information ready: • ErrorID • ErrorInfo • Diagnostic buffer entries

See also

Tags of the positioning axis technology object as of V4 (Page 10142)

ErrorIDs and ErrorInfos (Page 10177)

Error displays of the Motion Control statements (Page 10113)

Using multiple axes with the same PTO (Page 10118)

Using multiple drives with the same PTO (Page 10122)

Tracking jobs from higher priority classes (execution levels) (Page 10123)
Special cases for use of software limit switches (Page 10125)

Tags of the positioning axis technology object as of V4

Position tag as of V4

Legend

Data type	Data type of tag	
Start value	Start value of tag	
Access	Access to the tag in the user program:	
	RCCP	The tag can be read in the user program and is updated at each cycle checkpoint.
	RP	The tag can be read with the Motion Control instruction "MC_ReadParam". The current value of the corresponding tag is determined at the start of the command.
Effective	Specifies when a change to the tag takes effect.	
HMI	The tag can be used in an HMI system (according to the access to the tag in the user program).	

<Axis name>.Position				
Current position of the axis (indicated in the configured unit of measurement) If the axis is not homed, the tag indicates the position value relative to the enable position of the axis.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.0	RCCP, RP	-	X

See also

Motion status (Page 10117)
Tag of the axis technology object V1...3 (Page 10190)

Velocity tag as of V4

Legend

Data type	Data type of tag	
Start value	Start value of tag	
Access	Access to the tag in the user program:	
	RCCP	The tag can be read in the user program and is updated at each cycle checkpoint.
	RP	The tag can be read with the Motion Control instruction "MC_ReadParam". The current value of the corresponding tag is determined at the start of the command.
Effective	Specifies when a change to the tag takes effect.	
HMI	The tag can be used in an HMI system (according to the access to the tag in the user program).	

<Axis name>.Velocity				
Current velocity of the axis (indicated in the configured unit of measurement)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.0	RCCP, RP	-	X

See also

Motion status (Page 10117)

Tag of the axis technology object V1...3 (Page 10190)

Actor tags as of V4

Legend

Data type	Data type of tag	
Start value	Start value of tag	
Access	Access to the tag in the user program:	
	R	The tag can be read in the user program.
	WP	If the axis is disabled (MC_Power.Status = FALSE), the tag can be written with the Motion Control instruction "MC_WriteParam".
	-	The tag cannot be used in the user program.
Effective	Specifies when a change to the tag takes effect:	
	2	When axis is enabled
HMI	The tag can be used in an HMI system (according to the access to the tag in the user program).	

<Axis name>.Actor.Type				
Drive connection The drive is connected via a pulse interface.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
DInt	2	R	-	X

<Axis name>.Actor.InverseDirection				
Invert direction FALSE = The direction is not inverted. TRUE = The direction is inverted.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X
		WP	2	-

<Axis name>.Actor.DirectionMode				
Permitted direction of rotation 0 = Both directions 1 = Positive direction 2 = Negative direction				
Data type	Start value	Access	Effective	HMI
Int	0	R	-	X
		WP	2	-

<Axis name>.Actor.Interface.EnableDriveOutput				
Enable output (internal parameter)				
Data type	Start value	Access	Effective	HMI
VREF	-	-	-	-

<Axis name>.Actor.Interface.DriveReadyInput				
Ready input (internal parameter)				
Data type	Start value	Access	Effective	HMI
VREF	-	-	-	-

<Axis name>.Actor.Interface.PTO				
Pulse output (internal parameter)				
Data type	Start value	Access	Effective	HMI
DWord	0	-	-	-

<Axis name>.Actor.DriveParameter.PulsesPerDriveRevolution				
Pulses per motor revolution				
Data type	Start value	Access	Effective	HMI
DInt	1000	R	-	X
		WP	2	-

See also

Tag of the axis technology object V1...3 (Page 10190)

Sensor[1] tags as of V4

ActiveHoming tags as of V4

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to the tag in the user program:	
	RW	The tag can be read and written in the user program. The tag can be written with Motion Control instruction "MC_WriteParam".
	R	The tag can be read in the user program.
	WP	If the axis is disabled (MC_Power.Status = FALSE), the tag can be written with the Motion Control instruction "MC_WriteParam".
	-	The tag cannot be used in the user program.
Effective	Specifies when a change to the tag takes effect:	
	1	When axis is activated (<Axis name>.StatusBits.Activated tag changes from FALSE → TRUE), disabled, or enabled
	2	When axis is enabled
	8	When an active homing command is started
HMI	The tag can be used in an HMI system (according to the access to the tag in the user program).	

<Axis name>.Sensor[1].ActiveHoming.Mode				
Active homing mode 2 = Homing via digital input				
Data type	**Start value**	**Access**	**Effective**	**HMI**
DInt	2	R	-	X
		WP	2	-

<Axis name>.Sensor[1].ActiveHoming.SideInput				
End of the homing switch to which the axis is homed during active homing FALSE = Lower end TRUE = Upper end				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RW	1, 8	X

<Axis name>.Sensor[1].ActiveHoming.DigitalInputAddress				
Symbolic input address of the homing switch (internal parameter)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
VREF	-	-	-	-

<Axis name>.Sensor[1].ActiveHoming.HomePositionOffset				
Home position offset (active homing) (indicated in the configured unit of measurement)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.0	RW	1, 8	X

<Axis name>.Sensor[1].ActiveHoming.SwitchLevel				
Selection of signal level that is present at the CPU input when the homing switch is reached FALSE = Low level (Low-active) TRUE = High level (High-active)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	TRUE	RW	1, 8	X

See also

Tag of the axis technology object V1...3 (Page 10190)

PassiveHoming tags as of V4

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to the tag in the user program:	
	RW	The tag can be read and written in the user program. The tag can be written with Motion Control instruction "MC_WriteParam".
	R	The tag can be read in the user program.
	WP	If the axis is disabled (MC_Power.Status = FALSE), the tag can be written with the Motion Control instruction "MC_WriteParam".
	-	The tag cannot be used in the user program.
Effective	Specifies when a change to the tag takes effect:	
	1	When axis is activated (<Axis name>.StatusBits.Activated tag changes from FALSE -> TRUE), disabled, or enabled
	2	When axis is enabled
	7	When a passive homing command is started
HMI	The tag can be used in an HMI system (according to the access to the tag in the user program).	

<Axis name>.Sensor[1].PassiveHoming.Mode				
Passive homing mode 2 = Homing via digital input				
Data type	**Start value**	**Access**	**Effective**	**HMI**
DInt	2	R	-	X
		WP	2	-

<Axis name>.Sensor[1].PassiveHoming.SideInput				
End of the homing switch to which the axis is homed during passive homing FALSE = Lower end TRUE = Upper end				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RW	1, 7	X

<Axis name>.Sensor[1].PassiveHoming.DigitalInputAddress				
Symbolic input address of the homing switch (internal parameter)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
VREF	-	-	-	-

<Axis name>.Sensor[1].PassiveHoming.SwitchLevel				
Selection of level that is present at the CPU input when the homing switch is reached FALSE = Low level (Low-active) TRUE = High level (High-active)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	TRUE	RW	1, 7	X

See also

Tag of the axis technology object V1...3 (Page 10190)

Units tag as of V4

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to the tag in the user program:	
	R	The tag can be read in the user program.
	WP	If the axis is disabled (MC_Power.Status = FALSE), the tag can be written with the Motion Control instruction "MC_WriteParam".
Effective	Specifies when a change to the tag takes effect:	
	2	When axis is enabled
HMI	The tag can be used in an HMI system (according to the access to the tag in the user program).	

<Axis name>.Units.LengthUnit				
Configured unit of measurement of the parameter -1 = "Pulse" 1005 = "°" (degrees) 1013 = "mm" 1010 =: "m" 1018 = "ft" 1019 = "in"				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Int	1013	R	-	X
		WP	2	-

See also

Tag of the axis technology object V1...3 (Page 10190)

Mechanics tag as of V4

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to the tag in the user program:	
	R	The tag can be read in the user program.
	WP	If the axis is disabled (MC_Power.Status = FALSE), the tag can be written with the Motion Control instruction "MC_WriteParam".
Effective	Specifies when a change to the tag takes effect:	
	2	When axis is enabled
HMI	The tag can be used in an HMI system (according to the access to the tag in the user program).	

<Axis name>.Mechanics.LeadScrew				
Distance per revolution (indicated in the configured unit of measurement)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	10.0	R	-	X
		WP	2	-

See also

Tag of the axis technology object V1...3 (Page 10190)

DynamicLimits tags as of V4

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to the tag in the user program:	
	R	The tag can be read in the user program.
	WP	If the axis is disabled (MC_Power.Status = FALSE), the tag can be written with the Motion Control instruction "MC_WriteParam".
Effective	Specifies when a change to the tag takes effect:	
	2	When axis is enabled
HMI	The tag can be used in an HMI system (according to the access to the tag in the user program).	

<Axis name>.DynamicLimits.MaxVelocity				
Maximum velocity of the axis (indicated in the configured unit of measurement)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	250.0	R	-	X
		WP	2	-

<Axis name>.DynamicLimits.MinVelocity				
Start/stop velocity of the axis (indicated in the configured unit of measurement)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	10.0	R	-	X
		WP	2	-

See also

Tag of the axis technology object V1...3 (Page 10190)

DynamicDefaults tags as of V4

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to the tag in the user program:	
	RW	The tag can be read and written in the user program. The tag can be written with Motion Control instruction "MC_WriteParam".
Effective	Specifies when a change to the tag takes effect:	

	1	When axis is activated (<Axis name>.StatusBits.Activated tag changes from FALSE -> TRUE), disabled, or enabled
	5	The next time an MC_MoveAbsolute, MC_MoveRelative, MC_MoveVelocity, MC_MoveJog, MC_Halt, MC_CommandTable, or active MC_Home command is started (Mode = 3).
	6	When a MC_MoveJog command is stopped
HMI	The tag can be used in an HMI system (according to the access to the tag in the user program).	

<Axis name>.DynamicDefaults.Acceleration				
Acceleration of the axis (indicated in the configured unit of measurement)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	48.0	RW	1, 5, 6	X

<Axis name>.DynamicDefaults.Deceleration				
Deceleration of the axis (indicated in the configured unit of measurement)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	48.0	RW	1, 5, 6	X

<Axis name>.DynamicDefaults.EmergencyDeceleration				
Emergency stop deceleration of the axis (indicated in the configured unit of measurement)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	120.0	RW	1, 5, 6	X

<Axis name>.DynamicDefaults.Jerk				
Jerk during acceleration and deceleration ramp of the axis (indicated in the configured unit of measurement) The jerk is activated if the configured jerk is greater than 0.00004 mm/s².				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	192.0	RW	1, 5	X

See also

Tag of the axis technology object V1...3 (Page 10190)

PositionLimitsSW tags as of V4

Legend

Data type	Data type of tag
Start value	Start value of tag The start value can be overwritten by the axis configuration.

Access	Access to the tag in the user program:	
	RW	The tag can be read and written in the user program. The tag can be written with Motion Control instruction "MC_WriteParam".
Effective	Specifies when a change to the tag takes effect:	
	1	When axis is activated (<Axis name>.StatusBits.Activated tag changes from FALSE -> TRUE), disabled, or enabled
	5	The next time an MC_MoveAbsolute, MC_MoveRelative, MC_MoveVelocity, MC_MoveJog, MC_Halt, MC_CommandTable, or active MC_Home command is started (Mode = 3).
	6	When a MC_MoveJog command is stopped
HMI	The tag can be used in an HMI system (according to the access to the tag in the user program).	

<Axis name>.PositionLimitsSW.Active				
Activation of the software limit switches FALSE = The software limit switches are deactivated. TRUE = The software limit switches are activated.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RW	1, 5, 6	X

<Axis name>.PositionLimitsSW.MinPosition				
Position of the lower software limit switch (indicated in the configured unit of measurement)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	-10000.0	RW	1, 5, 6	X

<Axis name>.PositionLimitsSW.MaxPosition				
Position of the upper software limit switch (indicated in the configured unit of measurement)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	10000.0	RW	1, 5, 6	X

See also

Tag of the axis technology object V1...3 (Page 10190)

PositionLimitsHW tags as of V4

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to tag in the user program:	
	RW	The tag can be read and written in the user program. The tag can be written with Motion Control instruction "MC_WriteParam".

	R	The tag can be read in the user program.
	WP	If the axis is disabled (MC_Power.Status = FALSE), the tag can be written with Motion Control instruction "MC_WriteParam".
	-	The tag cannot be used in the user program.
Effective	Specifies when a change to the tag takes effect:	
	1	When axis is activated (<Axis name>.StatusBits.Activated tag changes from FALSE -> TRUE), disabled, or enabled
	2	When axis is enabled
	5	The next time an MC_MoveAbsolute, MC_MoveRelative, MC_MoveVelocity, MC_MoveJog, MC_Halt, MC_CommandTable, or active MC_Home command is started (Mode = 3).
	6	When a MC_MoveJog command is stopped
HMI	The tag can be used in an HMI system (according to the access to the tag in the user program).	

<Axis name>.PositionLimitsHW.Active				
Activation of the hardware limit switches FALSE = The hardware limit switches are deactivated. TRUE = The hardware limit switches are activated.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RW	1, 5, 6	X

<Axis name>.PositionLimitsHW.MinSwitchLevel				
Selection of signal level that is present at the CPU input when the lower hardware limit switch is reached FALSE = Low level (Low-active) TRUE = High level (High-active)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X
		WP	2	-

<Axis name>.PositionLimitsHW.MinSwitchAddress				
Symbolic input address of the lower hardware limit switch (internal parameter)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
VREF	-	-	-	-

<Axis name>.PositionLimitsHW.MaxSwitchLevel				
Selection of signal level that is present at the CPU input when the upper hardware limit switch is reached FALSE = Low level (Low-active) TRUE = High level (High-active)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X
		WP	2	-

<Axis name>.PositionLimitsHW.MaxSwitchAddress				
Input address of the upper hardware limit switch (internal parameter)				
Data type	Start value	Access	Effective	HMI
VREF	-	-	-	-

See also

Tag of the axis technology object V1...3 (Page 10190)

Homing tags as of V4

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to the tag in the user program:	
	RW	The tag can be read and written in the user program. The tag can be written with Motion Control instruction "MC_WriteParam".
Effective	Specifies when a change to the tag takes effect:	
	1	When axis is activated (<Axis name>.StatusBits.Activated tag changes from FALSE → TRUE), disabled, or enabled
	8	When an active homing command is started
HMI	The tag can be used in an HMI system (according to the access to the tag in the user program).	

<Axis name>.Homing.AutoReversal				
Activation of auto reverse at the hardware limit switch during active homing FALSE = Auto reversal at the hardware limit switch is deactivated. TRUE = Auto reversal at the hardware limit switch is activated.				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	RW	1, 8	X

<Axis name>.Homing.ApproachDirection				
Approach and homing direction of axis during active homing FALSE = Negative approach direction for finding the homing switch and negative homing direction TRUE = Positive approach direction for finding the homing switch and positive homing direction				
Data type	Start value	Access	Effective	HMI
Bool	TRUE	RW	1, 8	X

<Axis name>.Homing.ApproachVelocity				
Approach velocity of the axis during active homing (indicated in the configured unit of measurement)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	200.0	RW	1, 8	X

<Axis name>.Homing.ReferencingVelocity				
Homing velocity of the axis during active homing (indicated in the configured unit of measurement)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	40.0	RW	1, 8	X

See also

Tag of the axis technology object V1...3 (Page 10190)

StatusPositioning tags as of V4

Legend

Data type	Data type of tag	
Start value	Start value of tag	
Access	Access to the tag in the user program:	
	RCCP	The tag can be read in the user program and is updated at each cycle checkpoint.
	RP	The tag can be read with the Motion Control instruction "MC_ReadParam". The current value of the corresponding tag is determined at the start of the command.
Effective	Specifies when a change to the tag takes effect.	
HMI	The tag can be used in an HMI system.	

<Axis name>.StatusPositioning.Distance				
Current distance of axis from target position (indicated in the configured unit of measurement) The value of the tag is only valid during execution of a positioning command with "MC_MoveAbsolute", "MC_MoveRelative", or the axis control panel.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.0	RCCP, RP	-	X

<Axis name>.StatusPositioning.TargetPosition				
Target position of the axis (indicated in the configured unit of measurement) The value of the tag is only valid during execution of a positioning command with "MC_MoveAbsolute", "MC_MoveRelative", or the axis control panel.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.0	RCCP, RP	-	X

See also

Motion status (Page 10117)

Tag of the axis technology object V1...3 (Page 10190)

StatusBits tags as of V4

Legend

Data type	Data type of tag	
Start value	Start value of tag	
Access	Access to the tag in the user program:	
	RCCP	The tag can be read in the user program and is updated at each cycle checkpoint.
Effective	Specifies when a change to the tag takes effect.	
HMI	The tag can be used in an HMI system.	

<Axis name>.StatusBits.Activated				
Activation of the axis FALSE = The axis is not activated. TRUE = The axis is activated. It is connected to the assigned PTO (Pulse Train Output). The data of the technology data block will be updated cyclically.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP	-	X

<Axis name>.StatusBits.Enable				
Enable status of the axis FALSE = The axis is not enabled. TRUE = The axis is enabled and ready to accept Motion Control commands.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP	-	X

<Axis name>.StatusBits.HomingDone				
Homing status of the axis FALSE = The axis is not homed. TRUE = The axis is homed and is capable of executing absolute positioning commands. The axis does not have to be homed for relative positioning. The status is FALSE during active homing. The status remains TRUE during passive homing if the axis was already homed beforehand.				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	RCCP	-	X

<Axis name>.StatusBits.Done				
Command execution on the axis FALSE = A Motion Control command is active on the axis. TRUE = A Motion Control command is not active on the axis.				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	RCCP	-	X

<Axis name>.StatusBits.Error				
Error status on the axis FALSE = No error is active on the axis. TRUE = An error occurred on the axis. Additional information about the error is available in automatic control at the "ErrorID" and "ErrorInfo" parameters of the Motion Control instructions. In manual mode, the "Error message" box of the axis control panel displays detailed information about the cause of error.				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	RCCP	-	X

<Axis name>.StatusBits.StandStill				
Standstill of the axis FALSE = The axis is in motion. TRUE = The axis is at a standstill.				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	RCCP	-	X

<Axis name>.StatusBits.PositioningCommand				
Execution of a positioning command FALSE = A positioning command is not active on the axis. TRUE = The axis is executing a positioning command of the "MC_MoveRelative" or MC_MoveAbsolute Motion Control instruction.				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	RCCP	-	X

<Axis name>.StatusBits.VelocityCommand				
Execution of a command with velocity specification FALSE = A command with velocity specification is not active on the axis. TRUE = The axis is executing a motion command with velocity specification of the "MC_MoveVelocity" or MC_MoveJog Motion Control instruction.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP	-	X

<Axis name>.StatusBits.HomingCommand				
Execution of a homing command FALSE = A homing command is not active on the axis. TRUE = The axis is executing a homing command of the "MC_Home" Motion Control instruction.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP	-	X

<Axis name>.StatusBits.CommandTableActive				
Execution of a command table FALSE = A command table is not active on the axis. TRUE = The axis is controlled by "MC_CommandTable" Motion Control instruction.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP	-	X

<Axis name>.StatusBits.ConstantVelocity				
Constant velocity FALSE = The axis is accelerating, decelerating, or at a standstill. TRUE = The setpoint velocity has been reached. The axis is moving at constant velocity.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP	-	X

<Axis name>.StatusBits.Accelerating				
Acceleration process FALSE = The axis is decelerating, moving at constant velocity, or at a standstill. TRUE = The axis is accelerating.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP	-	X

<Axis name>.StatusBits.Decelerating				
Deceleration process FALSE = The axis is accelerating, moving at constant velocity, or at a standstill. TRUE = The axis is decelerating.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP	-	X

<Axis name>.StatusBits.ControlPanelActive				
Activation status of the axis control panel FALSE = "Automatic mode" is activated. The user program has control priority over the axis. TRUE = The "Manual control" mode was enabled in the axis control panel. The axis control panel has control priority over the axis. The axis cannot be controlled from the user program.				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	RCCP	-	X

<Axis name>.StatusBits.DriveReady				
Operational status of the drive FALSE = The drive is not ready. Setpoints will not be executed. TRUE = The drive is ready. Setpoints can be executed.				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	RCCP	-	X

<Axis name>.StatusBits.RestartRequired				
Restart of axis required FALSE = A restart of the axis is not required. TRUE = Values were modified in the load memory. To download the values to the work memory while the CPU is in RUN mode, the axis must be restarted. Use the MC_Reset Motion Control instruction to do this.				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	RCCP	-	X

<Axis name>.StatusBits.SWLimitMinActive				
Status of the lower software limit switch FALSE = The axis is kept within its configured working range. TRUE = The lower software limit switch was reached or exceeded.				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	RCCP	-	X

<Axis name>.StatusBits.SWLimitMaxActive				
Status of the upper software limit switch FALSE = The axis is kept within its configured working range. TRUE = The upper software limit switch was reached or exceeded.				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	RCCP	-	X

<Axis name>.StatusBits.HWLimitMinActive				
Status of the lower hardware limit switch FALSE = The axis is kept within its configured permitted traversing range. TRUE = The lower hardware limit switch was reached or exceeded.				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	RCCP	-	X

<Axis name>.StatusBits.HWLimitMaxActive				
Status of the upper hardware limit switch FALSE = The axis is kept within its configured permitted traversing range. TRUE = The upper hardware limit switch was reached or exceeded.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP	-	X

See also

Status and error bits (technology objects as of V4) (Page 10114)
Tag of the axis technology object V1...3 (Page 10190)

ErrorBits tags as of V4

Legend

Data type	Data type of tag	
Start value	Start value of tag	
Access	Access to the tag in the user program:	
	RCCP	The tag can be read in the user program and is updated at each cycle checkpoint.
Effective	Specifies when a change to the tag takes effect.	
HMI	The tag can be used in an HMI system.	

<Axis name>.ErrorBits.SystemFault				
Internal system error				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP	-	X

<Axis name>.ErrorBits.ConfigFault				
Incorrect configuration of the axis				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP	-	X

<Axis name>.ErrorBits.DriveFault				
Error in the drive. Loss of the "Drive ready" signal.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP	-	X

<Axis name>.ErrorBits.SWLimit				
Software limit switch reached or exceeded				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RCCP	-	X

<Axis name>.ErrorBits.HWLimit				
Hardware limit switch reached or exceeded				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	RCCP	-	X

<Axis name>.ErrorBits.DirectionFault				
Impermissible motion direction				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	RCCP	-	X

<Axis name>.ErrorBits.HWUsed				
Another axis is using the same PTO (Pulse Train Output) and is enabled with "MC_Power".				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	RCCP	-	X

See also

Status and error bits (technology objects as of V4) (Page 10114)

Tag of the axis technology object V1...3 (Page 10190)

ControlPanel tags as of V4

The "ControlPanel" tags do not contain any user-relevant data. These tags cannot be accessed in the user program.

See also

Tag of the axis technology object V1...3 (Page 10190)

Internal tags as of V4

The "Internal" tags do not contain any user-relevant data. These tags cannot be accessed in the user program.

See also

Tag of the axis technology object V1...3 (Page 10190)

Update of the technology object tags

The status and error information of the axis indicated in the technology object tags is updated at each cycle control point.

Changes to the values of configuration tags do not take effect immediately. For information on the conditions under which a change takes effect, refer to the detailed description of the relevant tag.

Tags of the command table technology object as of V4

Command[1...32] tags as of V4

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the configuration of the command table.	
Access	Access to the tag in the user program:	
	RW	The tag can be read and written in the user program.
	R	The tag can be read in the user program.
	-	The tag cannot be used in the user program.
Effective	Specifies when a change to the tag takes effect.	
HMI	The tag can be used in an HMI system.	

<Command table>.Command[n].Type				
Command type • 0 = "Empty" command • 2 = "Hold" command • 5 = "Relative positioning" command • 6 = "Absolute positioning" command • 7 = "Velocity setpoint" command • 151 = "Wait" command				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Int	0	RW	-	X

<Command table>.Command[n].Position				
Target position/traversing distance of the command				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.0	RW	-	X

<Axis name>.Command[n].Velocity				
Command velocity				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.0	RW	-	X

<Command table>.Command[n].Duration				
Command duration				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.0	RW	-	X

<Command table>.Command[n].NextStep				
Mode for the transition to the next command • 0 = "Complete command" • 1 = "Blend movement"				
Data type	Start value	Access	Effective	HMI
Int	0	RW	-	X

<Command table>.Command[n].StepCode				
Command step code				
Data type	Start value	Access	Effective	HMI
Word	0	RW	-	X

See also

Tags of the command table technology object V1...3 (Page 10205)

Versions V1...3

CPU outputs relevant for motion control (technology version V1...3)

The number of usable drives depends on the CPU, the number of PTOs (pulse train outputs) and the number of available pulse generator outputs.

The following tables provide information about the relevant dependencies:

Maximum number of PTOs

The maximum number of controllable PTOs (drives) depends on the order number of CPU:

Order number CPU	Number of PTOs
xxxxxxx-1xx**30**-xxxx	2
xxxxxxx-1xx**31**-xxxx	4

The maximum number of controllable PTOs (drives) applies regardless of the use of a signal board.

Usable pulse generator outputs

The CPU provides one pulse output and one direction output for controlling a stepper motor drive or a servo motor drive with pulse interface. The pulse output provides the drive with the pulses required for motor motion. The direction output controls the travel direction of the drive.

Pulse and direction outputs are permanently assigned to one another and form a pulse generator output. On-board CPU outputs or outputs of a signal board can be used as pulse generator outputs. You select between on-board CPU outputs and outputs of the signal board during device configuration under Pulse generators (PTO/PWM) on the "Properties" tab.

The following table shows the number of usable drives per CPU or signal board:

CPU		On-board	Signal board				
			DI2/DO2 x DC24V 20kHz	DI2/DO2 x DC24V 200kHz	DO4 x DC24V 200kHz	DI2/DO2 x DC5V 200kHz	DO4 x DC5V 200kHz
CPU 1211C, CPU 1212C, CPU 1214C (MLFB - order number xxxxxxx-1xx30-xxxx)	DC/DC/DC	2	2	2	2	2	2
	AC/DC/RLY	-	1	1	2	1	2
	DC/DC/RLY	-	1	1	2	1	2
CPU 1211C (MLFB - order number xxxxxxx-1xx31-xxxx)	DC/DC/DC	2	3	3	4	3	4
	AC/DC/RLY	-	1	1	2	1	2
	DC/DC/RLY	-	1	1	2	1	2
CPU 1212C (MLFB - order number xxxxxxx-1xx31-xxxx)	DC/DC/DC	3	4	4	4	4	4
	AC/DC/RLY	-	1	1	2	1	2
	DC/DC/RLY	-	1	1	2	1	2
CPU 1214C (MLFB - order number xxxxxxx-1xx31-xxxx)	DC/DC/DC	4	4	4	4	4	4
	AC/DC/RLY	-	1	1	2	1	2
	DC/DC/RLY	-	1	1	2	1	2
CPU 1215C	DC/DC/DC	4	4	4	4	4	4
	AC/DC/RLY	-	1	1	2	1	2
	DC/DC/RLY	-	1	1	2	1	2

The following table shows the address assignment of the pulse and direction outputs:

CPU S7-1200:		Outputs PTO1 *)		Outputs PTO2 **)		Outputs PTO3 *)		Outputs PTO4 **)	
		Pls.	Dir.	Pls.	Dir.	Pls.	Dir.	Pls.	Dir.
CPU 1211C, CPU 1212C, CPU 1214C, CPU1215C (DC/DC/DC)	CPU	Ax.0	Ax.1	Ax.2	Ax.3	Ax.4	Ax.5	Ax.6	Ax.7
	Signal board	Ay.0	Ay.1	Ay.2	Ay.3	Ay.0	Ay.1	Ay.2	Ay.3
CPU 1211C, CPU 1212C, CPU 1214C, CPU 1215C	CPU	-	-	-	-	-	-	-	-

CPU S7-1200:		Outputs PTO1 *)		Outputs PTO2 **)		Outputs PTO3 *)		Outputs PTO4 **)	
		Pls.	Dir.	Pls.	Dir.	Pls.	Dir.	Pls.	Dir.
(AC/DC/RLY)	Signal board	Ay.0	Ay.1	Ay.2	Ay.3	Ay.0	Ay.1	Ay.2	Ay.3
CPU 1211C, CPU 1212C, CPU 1214C, CPU 1215C (DC/DC/RLY)	CPU	-	-	-	-	-	-	-	-
	Signal board	Ay.0	Ay.1	Ay.2	Ay.3	Ay.0	Ay.1	Ay.2	Ay.3

x = Initial byte address of on-board CPU outputs (default value = 0)

y = Initial byte address of signal board outputs (default value = 4)

* If a DC/DC/DC CPU variant is used together with a DI2/DO2 signal board, the signals of the PTO1/3 can be generated via the on-board CPU outputs or via the signal board.

** If a DC/DC/DC CPU variant is used together with a DO4 signal board, the signals for PTO1/3 or PTO2/4 can be generated via the on-board CPU outputs or via the signal board.

PTO3 and PTO4 are only available for CPUs with the order numbers xxxxxxx-1xx**31**-xxxx.

Note

Access to pulse generator outputs via the process image

The firmware takes control via the corresponding pulse generator outputs if the PTO (Pulse Train Output) is selected and assigned to an axis.

With this takeover of the control function, the connection between the process image and I/O output is also disconnected. Although the user has the option of writing the process image of pulse generator outputs via the user program or watch table, this is not transferred to the I/O output. Accordingly, it is also not possible to monitor the I/O output via the user program or watch table. The information read reflects the value of the process image and does not match the real status of the I/O output.

For all other CPU outputs that are not used permanently by the CPU firmware, the status of the I/O output can be controlled or monitored via the process image, as usual.

Limit frequencies of pulse outputs

The following limit frequencies apply to the pulse outputs:

Pulse output	Limit frequencies for technology object positioning axis V1	Limiting frequencies of the technology object positioning axis V2/V3 with CPU < V3.0	Limiting frequencies of the technology object positioning axis V2/V3 with CPU V3.0
On-board (MLFB - order number xxxxxxx-1xx**30**-xxxx)	2 Hz ≤ f ≤ 100 kHz	2 Hz ≤ f ≤ 100 kHz	1 Hz ≤ f ≤ 100 kHz
On-board (MLFB - order number xxxxxxx-1xx**31**xxxx)	2 Hz ≤ f ≤ 100 kHz (PTO 1+2) 2 Hz ≤ f ≤ 20 kHz (PTO 3+4)	2 Hz ≤ f ≤ 100 kHz (PTO 1+2) 2 Hz ≤ f ≤ 20 kHz (PTO 3+4)	1 Hz ≤ f ≤ 100 kHz (PTO 1+2) 1 Hz ≤ f ≤ 20 kHz (PTO 3+4)
Signal board DI2/DO2 x DC24V 20kHz	2 Hz ≤ f ≤ 20 kHz	2 Hz ≤ f ≤ 20 kHz	1 Hz ≤ f ≤ 20 kHz
Signal board DI2/DO2 x DC24V 200kHz	2 Hz ≤ f ≤ 100 kHz	2 Hz ≤ f ≤ 200 kHz	1 Hz ≤ f ≤ 200 kHz
Signal board DO4 x DC24V 200kHz	2 Hz ≤ f ≤ 100 kHz	2 Hz ≤ f ≤ 200 kHz	1 Hz ≤ f ≤ 200 kHz
Signal board DI2/DO2 x DC5V 200kHz	2 Hz ≤ f ≤ 100 kHz	2 Hz ≤ f ≤ 200 kHz	1 Hz ≤ f ≤ 200 kHz
Signal board DO4 x DC5V 200kHz	2 Hz ≤ f ≤ 100 kHz	2 Hz ≤ f ≤ 200 kHz	1 Hz ≤ f ≤ 200 kHz

Drive signals

For motion control, you can optionally parameterize a drive interface for "Drive enabled" and "Drive ready". When using the drive interface the digital output for the drive enable and the digital input for "drive ready" can be freely selected.

Acceleration/deceleration limits

The following limits apply to acceleration and deceleration:

Acceleration/deceleration	Value (CPU < V3.0)	Value (CPU V3.0)
Minimum acceleration/deceleration	2.8E-1 pulses/s^2	5.0E-3 pulses/s^2
Maximum acceleration/deceleration	9.5E+9 pulses/s^2	9.5E+9 pulses/s^2

Jerk limits

The following limits apply to the jerk:

Jerk	Value (CPU < V3.0)	Value (CPU V3.0)
Minimum jerk	4.0E-2 pulses/s^3	4.0E-3 pulses/s^3
Maximum jerk	1.5E+8 pulses/s^3	1.0E+10 pulses/s^3

See also

CPU outputs relevant for motion control (technology version V4) (Page 10008)

Configuration dialogs

Configuration - General ("Axis" technology object V1...3)

Configure the basic properties of the "Axis" technology object in the "General" configuration window.

Axis name:

Define the name of the axis or the name of the "Axis" technology object in this box. The technology object is listed under this name in the project navigation.

Hardware interface

The pulses are output to the power unit of the drive by fixed assigned digital outputs.

In CPUs with relay outputs, the pulse signal cannot be output on these outputs because the relays do not support the necessary switching frequencies. A signal board with digital outputs must be used to enable you to work with the PTO (Pulse Train Output) on these CPUs.

Note

The PTO requires the functionality of a fast counter (HSC). An HSC is used for this purpose with CPU version <V3.0. The HSC is then no longer available to the user. An internal HSC is used for this with CPU version ≥ V3.0.

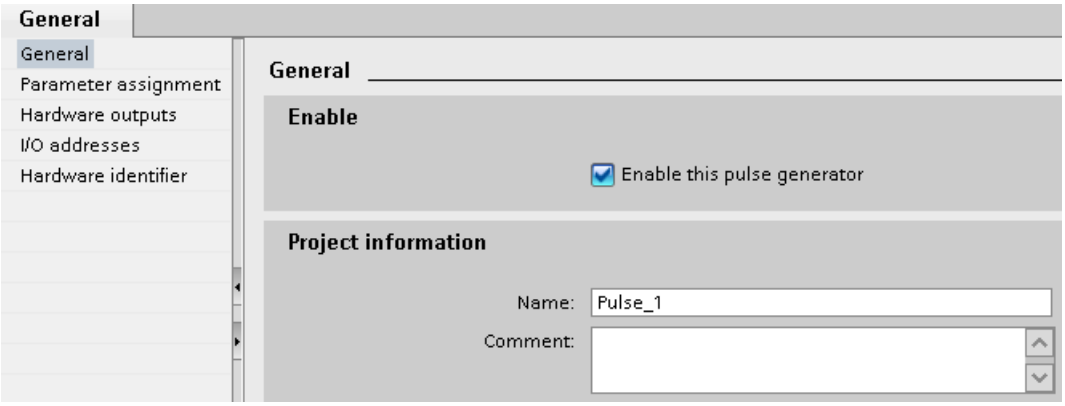
The count can not be evaluated from its input address.

The assignment between PTO and HSC is fixed. When the user activates PTO1, it is connected to the HSC1. If the PTO2 is activated, this is connected with the HSC2.

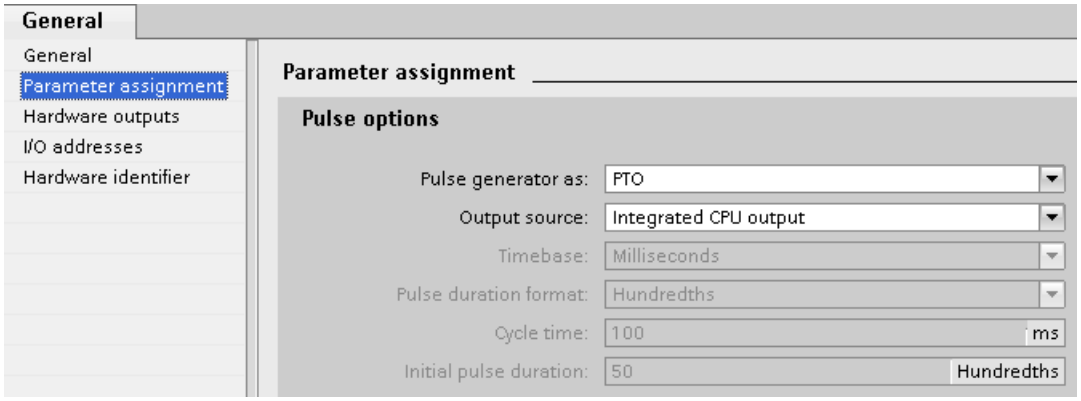
In the "Pulse generator selection" drop-down list, select the PTO (Pulse Train Output) which is to provide the pulses for controlling the stepper motors or servo motors with pulse interface. If the pulse generators and high-speed counters are not used elsewhere in the device configuration, the hardware interface can be configured automatically. In this case, the PTO selected in the drop-down list is displayed with a white background. The interfaces used are listed in the output boxes "Output source", "Pulse output", "Direction output" and "Assigned fast counter".

Proceed as follows if you wish to change the interfaces or if the PTO could not be automatically configured (entry in the "Pulse generator selection" drop-down list is highlighted in red):

1. Click on the "Device configuration" button.
   The pulse generator device configuration opens.
   Enlarge the property window of the device configuration if the configuration of the pulse generator is not visible.



2. Select the "Enable this pulse generator" check box.
3. Select the "Parameter assignment" entry in the area navigation.
   The "Parameter assignment" opens.



4. In the "Pulse generator as:" dropdown list select the "PTO" entry.
5. In the "Output source:" dropdown list select the "Integrated CPU output" or "Signal board output" entry. The "Signal board output" entry can only be selected for PTO1 or for PTO1 and PTO2 depending on the installed signal board. For more detailed information, see section: CPU outputs relevant for motion control (technology version V4) (Page 10008)
6. Go back to the axis configuration.
   Unless the corresponding fast counter has already been used elsewhere, the PTO boxes of the "General" axis configuration are not shaded red. If this is not the case, correct the configuration based on the error messages.

User unit

Select the desired unit for the dimension system of the axis in the dropdown list. The selected unit is used for additional configuration of the "Axis" technology object and for displaying the current axis data.

The values at the input parameters (Position, Distance, Velocity, ...) of the Motion Control instructions also refer to this unit.

Note

Later changing of the dimension system may not be converted correctly in all the configuration windows of the technology object. In this case check the configuration of all axis parameters.

The values of the input parameters of the Motion Control instructions may have to be adapted to the new unit of measurement in the user program.

See also

Configuration - General (positioning axis technology object as of V4) (Page 10033)

Changing configuration of dynamic values in user program ("Axis" technology object V1...3)

You can change the following configuration parameters during runtime of the user program in the CPU:

Acceleration and deceleration

You can also change the values for acceleration and deceleration during runtime of the user program. Use the following technology object tags for this purpose:

- <Axis name>.Config.DynamicDefaults.Acceleration
  for changing acceleration
- <Axis name>.Config.DynamicDefaults.Deceleration
  for changing deceleration

Refer to the description of the technology object tags (Page 10190) in the appendix for information on when changes to the configuration parameters take effect.

Emergency stop deceleration

You can also change the value for the emergency stop deceleration during runtime of the user program. Use the following technology object tag for this purpose:

- <Axis name>.Config.DynamicDefaults.EmergencyDeceleration

Refer to the description of the technology object tags in the appendix for information on when changes to the configuration parameter take effect.

NOTICE
After changes to this parameter, it may be necessary to adapt the positions of the hardware limit switches and other safety-relevant settings.

Jerk limit (as of technology object "Axis" V2.0)

You can also activate and deactivate the jerk limit at runtime of the user program and change the value for the jerk. Use the following technology object tags for this purpose:

- <Axis name>.Config.DynamicDefaults.JerkActive
  for activating and deactivating the jerk limit
- <Axis name>.Config.DynamicDefaults.Jerk
  for changing the jerk

Refer to the description of the technology object tags in the appendix for information on when changes to the configuration parameter take effect.

See also

Changing the configuration of dynamics in the user program (Page 10047)

Configuration - Homing ("Axis" technology object V1)

Configure the parameters for active and passive homing in the "Homing" configuration window. The homing method is set using the "Mode" input parameter of the Motion Control instruction. Here, Mode = 2 means passive homing and Mode = 3 means active homing.

Reference point switch input

Select the digital input for the reference point switch from the drop-down list. The input must be interrupt-capable. The onboard CPU inputs and the inputs of an inserted signal board can be selected as inputs for the reference point switch.

Note

The digital inputs are set to a filter time of 6.4 ms by default.

When the digital inputs are used as a reference point switch, this can result in undesired decelerations and thus inaccuracies. Depending on the homing velocity and extent of the reference point switch, the reference point may not be detected. The filter time can be set under "Input filter" in the device configuration of the digital inputs.

The specified filter time must be less than the duration of the input signal at the reference point switch.

Permitting direction reversal after reaching the HW limit switch (active homing only)

Activate the check box to use the hardware limit switch as a reversing cam for the homing procedure. The hardware limit switches must be activated for direction reversal. If the CPU firmware V1.0 is used, both hardware limit switches must be configured. If CPU firmware as of V2.0 is used, only the hardware limit switch in the approach direction must be configured.

If the hardware limit switch is reached during active homing, the axis brakes at the configured deceleration (not with the emergency stop deceleration) and reverses direction. The reference point switch is then sensed in reverse direction.

If the direction reversal is not active and the axis reaches the hardware limit switch during active homing, the homing procedure is aborted with an error and the axis is braked at the emergency stop deceleration.

Note

Use one of the following measures to ensure that the machine does not travel to a mechanical endstop in the event of a direction reversal:

- Keep the approach velocity low
- Increase the configured acceleration/deceleration
- Increase the distance between hardware limit switch and mechanical stop

Approach / homing direction (active and passive homing)

With the direction selection, you determine the "approach direction" used during active homing to search for the reference point switch, as well as the homing direction. The homing direction specifies the travel direction the axis uses to approach the configured end of the reference point switch to carry out the homing operation.

Refer to the table under "Reference point switches" for the effect of the approach direction setting on passive homing.

Side of the reference point switch (active and passive homing)

- **Active homing**
  This is where you select whether the axis is homed on the low or high end of the reference point switch.

  Note

  Depending on the start position of the axis and the configuration of the homing parameters, the homing procedure sequence can differ from the diagram in the configuration window.

- **Passive homing**
  With passive homing, the traversing motions for purposes of homing must be implemented by the user via motion commands. The end of the reference point switch on which homing occurs depends on the following factors:
  - "Approach direction" configuration
  - "Reference point switch" configuration
  - Current travel direction during passive homing

  The table below presents details on the effect of factors:

Influencing factors:			Result:
Configuration Approach direction	**Configuration Reference point switch**	**Current travel direction**	**Homing on Reference point switch**
Positive	"Bottom side"	Positive direction	**Top side**
		Negative direction	**Bottom side**

Influencing factors:			Result:
Configuration Approach direction	Configuration Reference point switch	Current travel direction	Homing on Reference point switch
Positive	"Top side"	Positive direction	**Bottom side**
		Negative direction	**Top side**
Negative	"Bottom side"	Positive direction	**Bottom side**
		Negative direction	**Top side**
Negative	"Top side"	Positive direction	**Top side**
		Negative direction	**Bottom side**

Velocity (active homing only)

In this box, specify the velocity at which the reference point switch is to be searched for during the homing procedure.

Limits (independent of the selected unit of measurement):

- Start/stop velocity ≤ approach velocity ≤ maximum velocity

Homing velocity (active homing only)

In this box, specify the velocity at which the axis approaches the reference point switch for homing.

Limits (independent of the selected unit of measurement):

- Start/stop velocity ≤ Homing velocity ≤ Maximum velocity

Home position offset (active homing only)

If the desired home position deviates from the position of the reference point switch, the home position offset can be specified in this box.

If the value does not equal 0, the axis executes the following actions following homing at the reference point switch:

1. Move the axis at the homing velocity by the value of the home position offset
2. Upon reaching the "home position offset", the axis is at the home position that was specified in input parameter "Position" of the "MC_Home" Motion Control instruction.

Limits (independent of the selected unit of measurement):

- -1.0e12 ≤ home position offset ≤ 1.0e12

Home position

The position configured in the Motion Control instruction "MC_Home" is used as the home position.

Configuration - Homing ("Axis" technology object V2...3)

Configuration - Homing - General (Axis technology object V2...3)

Configure the reference point switch input for active and passive homing in the "Homing - General" configuration window.

Reference point switch input

Select the digital input for the homing switch from the drop-down list. The input must be interrupt-capable. The on-board CPU inputs and the inputs of an inserted signal board can be selected as inputs for the homing switch.

Note

The digital inputs are set to a filter time of 6.4 ms by default.

When the digital inputs are used as a homing switch, this can result in undesired decelerations and thus inaccuracies. Depending on the homing velocity and extent of the homing switch, the home position may not be detected. The filter time can be set under "Input filter" in the device configuration of the digital inputs.

The specified filter time must be less than the duration of the input signal at the homing switch.

Active level

In the drop-list, select the level of the homing switch that is to be used for homing.

See also

Sequence - Active homing (Page 10053)

Configuration - Homing - Passive (Axis technology object V2...3)

Configure the necessary parameters for passive homing in the "Homing - Passive" configuration window.

The movement for passive homing must be triggered by the user (e.g. using an axis motion command). Passive homing is started using Motion Control instruction "MC_Home" with input parameter "Mode" = 2.

Side of the homing switch

This is where you select whether the axis is to be homed on the low or high end of the homing switch.

Home position

The position configured in the Motion Control instruction "MC_Home" is used as the home position.

Note

If passive homing is carried out without an axis motion command (axis at a standstill), homing will be executed upon the next rising or falling edge at the homing switch.

Configuration - Homing - Active (Axis technology object V2...3)

Configure the necessary parameters for active homing in the "Active homing" configuration window. Active homing is started using Motion Control instruction "MC_Home" with input parameter "Mode" = 3.

Permit auto reverse at the hardware limit switch

Activate the check box to use the hardware limit switch as a reversing cam for the homing procedure. The hardware limit switches must be enabled for the reversal of direction (at least the hardware limit switch in the direction of approach must be configured).

If the hardware limit switch is reached during active homing, the axis brakes at the configured deceleration (not with the emergency stop deceleration) and reverses direction. The homing switch is then sensed in reverse direction.

If the direction reversal is not active and the axis reaches the hardware limit switch during active homing, the homing procedure is aborted with an error and the axis is braked at the emergency stop deceleration.

Note

If possible, use one of the following measures to ensure that the machine does not travel to a mechanical endstop in the event of a direction reversal:

- Keep the approach velocity low.
- Increase the configured acceleration/deceleration.
- Increase the distance between the hardware limit switch and the mechanical endstop.

Approach/homing direction

With the direction selection, you determine the approach direction used during active homing to search for the homing switch, as well as the homing direction. The homing direction specifies the travel direction the axis uses to approach the configured end of the homing switch to carry out the homing operation.

Side of the homing switch

This is where you select whether the axis is to be homed on the low or high end of the homing switch.

Velocity

In this box, specify the velocity at which the homing switch is to be searched for during the homing procedure.

Limits (independent of the selected unit of measurement):

- Start/stop velocity ≤ approach velocity ≤ maximum velocity

Homing velocity

In this box, specify the velocity at which the reference point switch is to be approached for homing.

Limits (independent of the selected unit of measurement):

- Start/stop velocity ≤ Homing velocity ≤ Maximum velocity

Home position offset

If the desired home position deviates from the position of the homing switch, the home position offset can be specified in this box.

If the value does not equal 0, the axis executes the following actions following homing at the homing switch:

1. Move the axis at the homing velocity by the value of the home position offset
2. Upon reaching the "home position offset", the axis is at the home position that was specified in input parameter "Position" of the "MC_Home" Motion Control instruction.

Limits (independent of the selected unit of measurement):

- -1.0e12 ≤ home position offset ≤ 1.0e12

Home position

The position configured in the Motion Control instruction "MC_Home" is used as the home position.

Diagnostics - Status and error bits ("Axis" technology object V1...3)

You use the "Status and error bits" diagnostic function to monitor the most important status and error messages for the axis in the TIA Portal. The diagnostic function display is available in online mode in "Manual control" mode and in "Automatic control" when the axis is active. The status error messages have the following meaning:

Status of the axis

Status	Description
Enabled	The axis is enabled and ready to be controlled via Motion Control commands. (Tag of technology object: <Axis name>.StatusBits.Enable)
Homed	The axis is homed and is capable of executing absolute positioning commands of Motion Control instruction "MC_MoveAbsolute". The axis does not have to be homed for relative positioning. Special situations: • During active homing, the status is FALSE. • If a homed axis undergoes passive homing, the status is set to TRUE during passive homing. (Tag of technology object: <Axis name>.StatusBits.HomingDone)
Axis error	An error has occurred in the "Axis" technology object. Additional information about the error is available in automatic control at the ErrorID and ErrorInfo parameters of the Motion Control instructions. In manual mode, the "Error message" box of the axis control panel displays detailed information about the cause of error. (Tag of technology object: <Axis name>.StatusBits.Error)
Axis control panel enabled	The "Manual control" mode was enabled in the axis control panel. The axis control panel has control priority over the "Axis" technology object. The axis cannot be controlled from the user program. (Tag of technology object: <Axis name>.StatusBits.ControlPanelActive)
Restart necessary	A modified configuration of the axis was downloaded to the load memory in CPU RUN operating mode. To download the modified configuration to the work memory, you need to restart the axis. Use the Motion Control instruction MC_Reset to do this.

Drive status

Status	Description
Drive ready	The drive is ready for operation. (Tag of technology object: <Axis name>.StatusBits.DriveReady)
Drive error	The drive has reported an error due to loss of its "Drive ready" signal. (Tag of technology object: <Axis name>.ErrorBits.DriveFault)

Status of the axis motion

Status	Description
Standstill	The axis is at a standstill. (Tag of technology object: <Axis name>.StatusBits.StandStill)
Acceleration	The axis accelerates. (Tag of technology object: <Axis name>.StatusBits.Acceleration)
Constant velocity	The axis travels at constant velocity. (Tag of technology object: <Axis name>.StatusBits.ConstantVelocity)
Deceleration	The axis decelerates (slows down). (Tag of technology object: <Axis name>.StatusBits.Deceleration)

Status of the motion mode

Status	Description
Positioning	The axis executes a positioning command of the Motion Control instruction "MC_MoveAbsolute", "MC_MoveRelative" or the axis control panel. (Tag of technology object: <Axis name>.StatusBits.PositioningCommand)
Travel with velocity specification	The axis executes a command with velocity specification of the Motion Control instruction "MC_MoveVelocity", "MC_MoveJog" or the axis control panel. (Tag of technology object: <Axis name>.StatusBits.SpeedCommand)
Homing	The axis executes a homing command of the Motion Control instruction "MC_Home" or the axis control panel. (Tag of technology object: <Axis name>.StatusBits.Homing)
Command table active (as of technology object Axis V2.0)	The axis is controlled by Motion Control instruction "MC_CommandTable". (Tag of technology object: <Axis name>.StatusBits.CommandTableActive)

Error messages

Error	Description
Lower SW limit switch was reached	The lower software limit switch has been reached. (Tag of technology object: <Axis name>.ErrorBits.SwLimitMinReached)
Lower SW limit switch was exceeded	The lower software limit switch has been exceeded. (Tag of technology object: <Axis name>.ErrorBits.SwLimitMinExceeded)
Upper SW limit switch was reached	The upper software limit switch has been reached. (Tag of technology object: <Axis name>.ErrorBits.SwLimitMaxReached)
Upper SW limit switch was exceeded	The upper software limit switch has been exceeded. (Tag of technology object: <Axis name>.ErrorBits.SwLimitMaxExceeded)
Lower HW limit switch was reached	The lower hardware limit switch has been reached. (Tag of technology object: <Axis name>.ErrorBits.HwLimitMin)
Upper HW limit switch was reached	The upper hardware limit switch has been reached. (Tag of technology object: <Axis name>.ErrorBits.HwLimitMax)
PTO and HSC already in use	A second axis is using the same PTO (Pulse Train Output) and HSC (High Speed Counter) and is enabled with "MC_Power". (Tag of technology object: <Axis name>.ErrorBits.HwUsed)
Configuration error	The "Axis" technology object was incorrectly configured or editable configuration data were modified incorrectly during runtime of the user program. (Tag of technology object: <Axis name>.ErrorBits.ConfigFault)
Internal error	An internal error has occurred. (Tag of technology object: <Axis name>.ErrorBits.SystemFault)

See also

Status and error bits (technology objects as of V4) (Page 10114)

ErrorIDs and ErrorInfos

List of ErrorIDs and ErrorInfos (technology objects V2...3)

The following table lists all ErrorIDs and ErrorInfos that can be indicated in Motion Control instructions. In addition to the cause of the error, remedies for eliminating the error are also listed:

Operating error with axis stop

ErrorID	ErrorInfo	Description	Remedy
16#8000		Drive error, loss of "Drive ready"	
	16#0001	-	Acknowledge error with instruction "MC_Reset"; provide drive signal; restart command, if necessary
16#8001		Lower SW limit switch has been tripped	
	16#000E	The position of the lower SW limit switch was reached with the currently configured deceleration	Acknowledge the error with instruction "MC_Reset"; use a motion command to move the axis in the positive direction out of the range of the SW limit switch
	16#000F	The position of the lower SW limit switch was reached with the emergency stop deceleration	
	16#0010	The position of the lower SW limit switch was exceeded with the emergency stop deceleration	
16#8002		Upper SW limit switch has been tripped	
	16#000E	The position of the upper SW limit switch was reached with the currently configured deceleration	Acknowledge the error with instruction "MC_Reset"; use a motion command to move the axis in the negative direction out of the range of the SW limit switch
	16#000F	The position of the upper SW limit switch was reached with the emergency stop deceleration	
	16#0010	The position of the upper SW limit switch was exceeded with the emergency stop deceleration	
16#8003		Lower HW limit switch was reached	
	16#000E	The lower HW limit switch was reached. The axis was stopped with the emergency stop deceleration. (During an active homing procedure, the reference point switch was not found)	Acknowledge the error for an enabled axis with instruction "MC_Reset"; use a motion command to move the axis in the positive direction out of the range of the HW limit switch.
16#8004		Upper HW limit switch was reached	
	16#000E	The upper HW limit switch has been reached. The axis was stopped with the emergency stop deceleration. (During an active homing procedure, the reference point switch was not found)	Acknowledge the error for an enabled axis with instruction "MC_Reset"; use a motion command to move the axis in the negative direction out of the range of the HW limit switch.
16#8005		PTO/HSC are already being used by another axis	
	16#0001	-	The axis was configured incorrectly: Correct the configuration of the PTO (Pulse Train Output) / HSC (High Speed Counter) and download it to the controller

ErrorID	ErrorInfo	Description	Remedy
			More than one axis is to run with one PTO: Another axis is using the PTO / HSC. If the current axis is to assume the control, the other axis must be disabled with "MC_Power" Enable = FALSE. (see also Using multiple axes with the same PTO (Page 10118))
16#8006		**A communication error in the control panel has occurred**	
	16#0012	A timeout has occurred	Check the cable connection and press the "Manual control" button again.
16#8007		**The axis cannot be enabled.**	
	16#0025	Restarting	Wait until the axis restart is complete.
	16#0026	Executing loading process in RUN mode	Wait until the loading process is complete.

Operating error without axis stop

ErrorID	ErrorInfo	Description	Remedy
16#8200		**Axis is not enabled**	
	16#0001	-	Enable the axis; restart the command
16#8201		**Axis has already been enabled by another "MC_Power" instance**	
	16#0001	-	Enable the axis with only one "MC_Power" instance
16#8202		**The maximum number of simultaneously active Motion Control commands has been exceeded (maximum of 200 commands for all motion control technology objects)**	
	16#0001	-	Reduce the number of simultaneously active commands; restart the command A command is active if parameter "Busy" = TRUE in the Motion Control instruction.
16#8203		**Axis is currently operated in "Manual control" (axis control panel)**	
	16#0001	-	Exit "Manual control"; restart the command
16#8204		**Axis is not homed**	
	16#0001	-	Home the axis with instruction "MC_Home"; restart the command
16#8205		**The axis is currently controlled by the user program (the error is only displayed in the axis control panel)**	
	16#0013	The axis is enabled in the user program.	Disable axis with instruction "MC_Power" and select "Manual control" again in the axis control panel
16#8206		**Technology object not activated yet**	
	16#0001	-	Enable the axis with instruction "MC_Power" Enable = TRUE or enable the axis in the axis control panel.
16#8207		**Command rejected**	
	16#0016	Active homing is running; another homing method cannot be started.	Wait for active homing to finish or abort the active homing with a motion command, for example, "MC_Halt".

ErrorID	ErrorInfo	Description	Remedy
	16#0018	The axis cannot be moved with a command table while it is being actively or passively homed.	Wait until direct or passive homing is complete.
	16#0019	The axis cannot be actively or passively homed while a command table is being processed.	Wait for command table to finish or abort the command table with a motion command, for example, "MC_Halt".
16#8208		Difference between maximum and start/stop velocity is invalid	
	16#0002	Number format of value is invalid	Correct the value; restart the command
	16#000A	Value is less than or equal to 0	
16#8209		Invalid acceleration for technology object "Axis"	
	16#0002	Number format of value is invalid	Correct the value; restart the command
	16#000A	Value is less than or equal to 0	
16#820A		It is not possible to restart the axis	
	16#0013	The axis is enabled in the user program.	Disable the axis with the "MC_Power" instruction; restart again
	16#0027	The axis is currently being operated in "Manual control" (axis control panel)	Exit "Manual control"; restart again
16#820B		It is not possible to execute the command table	
	16#0026	Executing loading process in RUN mode	Wait until the loading process is complete.

Block parameter error

ErrorID	ErrorInfo	Description	Remedy
16#8400		Invalid value at parameter "Position" of the Motion Control instruction	
	16#0002	Number format of value is invalid	Correct the value; restart the command
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#0006	The value is outside the number range (less than $-1e^{12}$)	
16#8401		Invalid value at parameter "Distance" of the Motion Control instruction	
	16#0002	Number format of value is invalid	Correct the value; restart the command
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#0006	The value is outside the number range (less than $-1e^{12}$)	
16#8402		Invalid value at parameter "Velocity" of the Motion Control instruction	
	16#0002	Number format of value is invalid	Correct the value; restart the command
	16#0008	Value is greater than the configured maximum velocity	
	16#0009	Value is less than the configured start/stop velocity	
	16#0024	Value is less than 0	
16#8403		Invalid value at parameter "Direction" of the Motion Control instruction	
	16#0011	The selection value is invalid	Correct the selection value; restart the command
16#8404		Invalid value at parameter "Mode" of the Motion Control instruction	

ErrorID	ErrorInfo	Description	Remedy
	16#0011	The selection value is invalid	Correct the selection value; restart the command
	16#0015	Active/passive homing is not configured	Correct the configuration and download it to the controller; enable the axis and restart the command
	16#0017	The direction reversal is activated at the hardware limit switch, despite the fact that the hardware limit switches are disabled	• Activate the hardware limit switch using the tag <axis>.Config.PositionLimits_HW.Active = TRUE, restart the command • Correct the configuration and download it to the controller; enable the axis and restart the command
16#8405		Invalid value at parameter "StopMode" of the Motion Control instruction	
	16#0011	The selection value is invalid	Correct the selection value; enable the axis again
16#8406		Simultaneous forward and backward jogging is not allowed	
	16#0001	-	Take steps to ensure that parameters "JogForward" and "JogBackward" do not have signal status TRUE simultaneously; restart the command.
16#8407		Switching to another axis with instruction "MC_Power" is only permitted after disabling the active axis.	
	16#0001	-	Disable the active axis; it is then possible to switch to the other axis and enable it.
16#8408		Invalid value at parameter "Axis" of the Motion Control instruction	
	16#001A	The specified value does not match the required technology object version	Correct the value; restart the command
	16#001B	The specified value does not match the required technology object type	
	16#001C	The specified value is not a Motion Control technology data block	
16#8409		Invalid value at parameter "CommandTable" of the Motion Control instruction	
	16#001A	The specified value does not match the required technology object version	Correct the value; restart the command
	16#001B	The specified value does not match the required technology object type	
	16#001C	The specified value is not a Motion Control technology data block	
16#840A		Invalid value at parameter "StartStep" of the Motion Control instruction	
	16#000A	Value is less than or equal to 0	Correct the value; restart the command
	16#001D	The start step is greater than the end step	
	16#001E	Value is greater than 32	
16#840B		Invalid value at parameter "EndStep" of the Motion Control instruction	
	16#000A	Value is less than or equal to 0	Correct the value; restart the command
	16#001E	Value is greater than 32	
16#840C		Invalid value at parameter "RampUpTime" of the Motion Control instruction	
	16#0002	Number format of value is invalid	Correct the value; restart the command
	16#000A	Value is less than or equal to 0	
16#840D		Invalid value at parameter "RampDownTime" of the Motion Control instruction	
	16#0002	Number format of value is invalid	Correct the value; restart the command
	16#000A	Value is less than or equal to 0	

ErrorID	ErrorInfo	Description	Remedy
16#840E		Invalid value at parameter "EmergencyRampTime" of the Motion Control instruction	
	16#0002	Number format of value is invalid	Correct the value; restart the command
	16#000A	Value is less than or equal to 0	
16#840F		Invalid value at parameter "JerkTime" of the Motion Control instruction	
	16#0002	Number format of value is invalid	Correct the value; restart the command
	16#000A	Value is less than or equal to 0	

Configuration error of the axis

ErrorID	ErrorInfo	Description	Remedy
16#8600		Parameter assignment of pulse generator (PTO is invalid	
	16#000B	The address is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0014	The selected hardware is used by another application	
16#8601		Parameter assignment of the high-speed counter (HSC) is invalid	
	16#000B	The address is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0014	The selected hardware is used by another application	
16#8602		Invalid parameter assignment of "Enable output"	
	16#000B	The address is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8603		Invalid parameter assignment of "Ready input"	
	16#000B	The address is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8604		Invalid "Pulses per motor revolution" value	
	16#000A	Value is less than or equal to zero	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
16#8605		Invalid "Distance per revolution" value	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#000A	Value is less than or equal to zero	
16#8606		Invalid "Start/stop velocity" value	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0003	Value exceeds the upper hardware limit	
	16#0004	Value is less than the lower hardware limit	
	16#0007	The start/stop velocity is greater than the maximum velocity	
16#8607		Invalid "maximum velocity" value	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#0003	Value exceeds the upper hardware limit	

ErrorID	ErrorInfo	Description	Remedy
	16#0004	Value is less than the lower hardware limit	
16#8608		Invalid "Acceleration" value	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0003	Value exceeds the upper hardware limit	
	16#0004	Value is less than the lower hardware limit	
16#8609		Invalid "Deceleration" value	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0003	Value exceeds the upper hardware limit	
	16#0004	Value is less than the lower hardware limit	
16#860A		Invalid "Emergency stop deceleration" value	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0003	Value exceeds the upper hardware limit	
	16#0004	Value is less than the lower hardware limit	
16#860B		Value for position of the lower SW limit switch is invalid	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#0006	The value is outside the number range (less than $-1e^{12}$)	
	16#0007	The position value of the lower software limit switch is greater than that of the upper software limit switch	
16#860C		Value for position of the upper SW limit switch is invalid	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#0006	The value is outside the number range (less than $-1e^{12}$)	
16#860D		Invalid address of the lower HW limit switch	
	16#000C	The address of the falling edge is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#000D	The address of the rising edge is invalid	
16#860E		Invalid address of the upper HW limit switch	
	16#000C	The address of the falling edge is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#000D	The address of the rising edge is invalid	
16#860F		Invalid "home position offset" value	

ErrorID	ErrorInfo	Description	Remedy
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#0006	The value is outside the number range (less than $-1e^{12}$)	
16#8610		Invalid "approach velocity" value	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0008	The velocity is greater than the maximum velocity	
	16#0009	The velocity is less than the start/stop velocity	
16#8611		Invalid "Homing velocity" value	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0008	The velocity is greater than the maximum velocity	
	16#0009	The velocity is less than the start/stop velocity	
16#8612		Invalid address of the reference point switch	
	16#000C	The address of the falling edge is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"
	16#000D	The address of the rising edge is invalid	
16#8613		During active homing, direction reversal at the hardware limit switch is activated although the hardware limit switches are not configured	
	16#0001	-	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
16#8614		Invalid "Jerk" value	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#001F	Value is greater than the maximum jerk	
	16#0020	Value is less than the minimum jerk	
16#8615		Value for "Unit of measurement" is invalid	
	16#0011	The selection value is invalid	Download error-free configuration to the controller; enable the axis again with instruction "MC_Power"

Configuration error of the command table

ErrorID	ErrorInfo	Description	Remedy
16#8700		**Value for "Command type" in the command table is invalid**	
	16#0001	-	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and restart the command, if necessary
16#8701		**Value for "Position / travel path" in the command table is invalid**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and restart the command, if necessary
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#0006	The value is outside the number range (less than $-1e^{12}$	
16#8702		**Value for "Velocity" in the command table is invalid**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and restart the command, if necessary
	16#0008	Value is greater than the configured maximum velocity	
	16#0009	Value is less than the configured start/stop velocity	
16#8703		**Value for "Duration" in the command table is invalid**	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and restart the command, if necessary
	16#0021	Value is greater than 64800 s	
	16#0022	Value is less than 0.001 s	
16#8704		**Value for "Next step" in the command table is invalid**	
	16#0011	The selection value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online and restart the command, if necessary
	16#0023	The command transition is not permitted for this command	

Internal errors

ErrorID	ErrorInfo	Description	Remedy
16#8FFF		**Internal error**	
	16#F0**	-	POWER OFF and POWER ON the CPU If this does not work, contact Customer Support. Have the following information ready: • ErrorID • ErrorInfo • Diagnostic buffer entries

See also

List of ErrorIDs and ErrorInfos (technology objects as of V4) (Page 10132)

Using multiple axes with the same PTO (Page 10118)

List of ErrorIDs and ErrorInfos (technology objects V1)

The following table lists all ErrorIDs and ErrorInfos that can be indicated in Motion Control instructions. In addition to the cause of the error, remedies for eliminating the error are also listed:

Operating error with axis stop

ErrorID	ErrorInfo	Description	Remedy
16#8000		Drive error, loss of "Drive ready"	
	16#0001	-	Acknowledge error with instruction "MC_Reset"; provide drive signal; restart command, if necessary
16#8001		Lower SW limit switch has been tripped	
	16#000E	The position of the lower SW limit switch was reached with the currently configured deceleration	Acknowledge the error with instruction "MC_Reset"; use a motion command to move the axis in the positive direction out of the range of the SW limit switch
	16#000F	The position of the lower SW limit switch was reached with the emergency stop deceleration	
	16#0010	The position of the lower SW limit switch was exceeded with the emergency stop deceleration	
16#8002		Upper SW limit switch has been tripped	
	16#000E	The position of the upper SW limit switch was reached with the currently configured deceleration	Acknowledge the error with instruction "MC_Reset"; use a motion command to move the axis in the negative direction out of the range of the SW limit switch
	16#000F	The position of the upper SW limit switch was reached with the emergency stop deceleration	
	16#0010	The position of the upper SW limit switch was exceeded with the emergency stop deceleration	
16#8003		Lower HW limit switch was reached	
	16#000E	The lower HW limit switch was reached. The axis was stopped with the emergency stop deceleration. (During an active homing procedure, the reference point switch was not found)	Acknowledge the error for an enabled axis with instruction "MC_Reset"; use a motion command to move the axis in the positive direction out of the range of the HW limit switch.
16#8004		Upper HW limit switch was reached	
	16#000E	The upper HW limit switch has been reached. The axis was stopped with the emergency stop deceleration. (During an active homing procedure, the reference point switch was not found)	Acknowledge the error for an enabled axis with instruction "MC_Reset"; use a motion command to move the axis in the negative direction out of the range of the HW limit switch.
16#8005		PTO/HSC are already being used by another axis	

ErrorID	ErrorInfo	Description	Remedy
	16#0001	-	**The axis was configured incorrectly:** Correct the configuration of the PTO (Pulse Train Output) / HSC (High Speed Counter) and download it to the controller
			More than one axis is to run with one PTO: Another axis is using the PTO / HSC. If the current axis is to assume the control, the other axis must be disabled with "MC_Power" Enable = FALSE. (see also Using multiple axes with the same PTO (Page 10118))

Operating error without axis stop

ErrorID	ErrorInfo	Description	Remedy
16#8200		**Axis is not enabled**	
	16#0001	-	Enable the axis; restart the command
16#8201		**Axis has already been enabled by another "MC_Power" instance**	
	16#0001	-	Enable the axis with only one "MC_Power" instruction
16#8202		**The maximum number of simultaneously active Motion Control commands was exceeded (maximum of 200 commands for all motion control technology objects)**	
	16#0001	-	Reduce the number of simultaneously active commands; restart the command A command is active if parameter "Busy" = TRUE in the Motion Control instruction.
16#8203		**Axis is currently operated in "Manual control" (axis control panel)**	
	16#0001	-	Exit "Manual control"; restart the command
16#8204		**Axis is not homed**	
	16#0001	-	Home the axis with instruction "MC_Home"; restart the command
16#8205		**The axis is currently controlled by the user program (the error is only displayed in the axis control panel)**	
	16#0001	-	Disable axis with instruction "MC_Power" and select "Manual control" again in the axis control panel
16#8206		**Technology object Axis not yet enabled**	
	16#0001	-	Enable the axis with instruction "MC_Power" Enable = TRUE or enable the axis in the axis control panel.
16#8207		**Command rejected**	
	16#0016	Active homing is running; another homing method cannot be started.	Wait for active homing to finish or abort the active homing with a motion command, for example, "MC_Halt". The other homing type can then be started.

Block parameter error

ErrorID	ErrorInfo	Description	Remedy
16#8400		**Invalid value at parameter "Position" of the Motion Control instruction**	
	16#0002	Number format of value is invalid	Correct the "position" value; restart the command
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#0006	The value is outside the number range (less than $-1e^{12}$)	
16#8401		**Invalid value at parameter "Distance" of the Motion Control instruction**	
	16#0002	Number format of value is invalid	Correct the "Distance" value; restart the command
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#0006	The value is outside the number range (less than $-1e^{12}$)	
16#8402		**Invalid value at parameter "Velocity" of the Motion Control instruction**	
	16#0002	Number format of value is invalid	Correct the "Velocity" value; restart the command
	16#0008	Velocity is greater than the maximum velocity	
	16#0009	Velocity is less than the start/stop velocity	
16#8403		**Invalid value at parameter "Direction" of the Motion Control instruction**	
	16#0011	Invalid selection value	Correct the selection value; restart the command
16#8404		**Invalid value at parameter "Mode" of the Motion Control instruction**	
	16#0011	Invalid selection value	Correct the selection value; restart the command
	16#0015	Active/passive homing is not configured	Correct the configuration and download it to the controller; enable the axis and restart the command
	16#0017	Axis reversal is activated at the HW limit switch, despite the fact that the hardware limit switches are disabled	• Activate the hardware limit switch using the tag <axis>.Config.PositionLimits_HW.Active = TRUE, restart the command • Correct the configuration and download it to the controller; enable the axis and restart the command
16#8405		**Invalid value at parameter "StopMode" of the Motion Control instruction**	
	16#0011	Invalid selection value	Correct the selection value; enable the axis again
16#8406		**Simultaneous forward and backward jogging is not allowed**	
	16#0001	-	Take steps to ensure that parameters "JogForward" and "JogBackward" do not have signal status TRUE simultaneously; restart the command.
16#8407		**Switching the axis with Motion Control instruction "MC_Power" is only permitted after disabling the axis.**	
	16#0001	-	Disable the active axis; it is then possible to switch to the other axis and enable it.

Configuration error

ErrorID	ErrorInfo	Description	Remedy
16#8600		Parameter assignment of pulse generator (PTO is invalid	
	16#000B	Address is invalid	Correct the configuration of the PTO (Pulse Train Output) and download it to the controller
16#8601		Parameter assignment of the high-speed counter (HSC) is invalid	
	16#000B	Address is invalid	Correct the configuration of the HSC (High Speed Counter) and download it to the controller
16#8602		Invalid parameter assignment of "Enable output"	
	16#000D	Address is invalid	Correct the configuration and download it to the controller
16#8603		Invalid parameter assignment of "Ready input"	
	16#000D	Address is invalid	Correct the configuration and download it to the controller
16#8604		Invalid "Pulses per motor revolution" value	
	16#000A	Value is less than or equal to zero	Correct the configuration and download it to the controller
16#8605		Invalid "Distance per revolution" value	
	16#0002	Number format of value is invalid	Correct the configuration and download it to the controller
	16#000A	Value is less than or equal to zero	
16#8606		Invalid "Start/stop velocity" value	
	16#0002	Number format of value is invalid	Correct the configuration and download it to the controller
	16#0003	Value exceeds the hardware limit	
	16#0004	Value is less than the hardware limit	
	16#0007	The start/stop velocity is greater than the maximum velocity	
16#8607		Invalid "Maximum velocity" value	
	16#0002	Number format of value is invalid	Correct the configuration and download it to the controller
	16#0003	Value exceeds the hardware limit	
	16#0004	Value is less than the hardware limit	
16#8608		Invalid "Acceleration" value	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0003	Value exceeds the hardware limit	
	16#0004	Value is less than the hardware limit	
16#8609		Invalid "Deceleration" value	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0003	Value exceeds the hardware limit	
	16#0004	Value is less than the hardware limit	
16#860A		Invalid "Emergency stop deceleration" value	

ErrorID	ErrorInfo	Description	Remedy
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0003	Value exceeds the hardware limit	
	16#0004	Value is less than the hardware limit	
16#860B		Value for position of the lower SW limit switch is invalid	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#0006	The value is outside the number range (less than $-1e^{12}$)	
	16#0007	The position value of the lower SW limit switch is greater than that of the upper SW limit switch	
16#860C		Value for position of the upper SW limit switch is invalid	
	16#0002	Number format of value is invalid	• Download error-free configuration to the controller; enable the axis again with instruction "MC_Power" • Correct the incorrect value online; acknowledge error with instruction "MC_Reset" and restart the command, if necessary
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#0006	The value is outside the number range (less than $-1e^{12}$)	
16#860D		Invalid address of the lower HW limit switch	
	16#000C	Address of falling edge is invalid	Correct the configuration and download it to the controller
	16#000D	Address of rising edge is invalid	
16#860E		Invalid address of the upper HW limit switch	
	16#000C	Address of falling edge is invalid	Correct the configuration and download it to the controller
	16#000D	Address of rising edge is invalid	
16#860F		Invalid "home position offset" value	
	16#0002	Number format of value is invalid	Correct the configuration and download it to the controller
	16#0005	The value is outside the number range (greater than $1e^{12}$)	
	16#0006	The value is outside the number range (less than $-1e^{12}$)	
16#8610		Invalid "approach velocity" value	
	16#0002	Number format of value is invalid	Correct the configuration and download it to the controller
	16#0008	Velocity is greater than the maximum velocity	
	16#0009	Velocity is less than the start/stop velocity	
16#8611		Invalid "Homing velocity" value	
	16#0002	Number format of value is invalid	Correct the configuration and download it to the controller
	16#0008	Velocity is greater than the maximum velocity	
	16#0009	Velocity is less than the start/stop velocity	
16#8612		Invalid address of the reference point switch	
	16#000C	Address of falling edge is invalid	Correct the configuration and download it to the controller
	16#000D	Address of rising edge is invalid	

ErrorID	ErrorInfo	Description	Remedy
16#8613		**During active homing, direction reversal at the hardware limit switch is activated although the hardware limit switches are not configured**	
	16#0001	-	Correct the configuration and download it to the controller

Internal errors

ErrorID	ErrorInfo	Description	Remedy
16#8FFF		**Internal error**	
	16#F0**	-	POWER OFF and POWER ON the CPU If this does not work, contact Customer Support. Have the following information ready: • ErrorID • ErrorInfo • Diagnostic buffer entries

See also

List of ErrorIDs and ErrorInfos (technology objects as of V4) (Page 10132)

Using multiple axes with the same PTO (Page 10118)

Tag of the axis technology object V1...3

Config tags V1...3

Config.General tags V1...3

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to tag in the user program:	
	RW	The tag can be read and written in the user program.
	R	The tag can be read in the user program.
	-	The tag cannot be used in the user program.
Effective	Specifies when a change to the tag takes effect.	
HMI	The tag can be used in an HMI system.	

<Axis name>.Config.General.PTO				
Tag cannot be evaluated in the user program.				
Data type	Start value	Access	Effective	HMI
DWORD	DW#16#00000000	-	-	-

<Axis name>.Config.General.HSC				
Tag cannot be evaluated in the user program.				
Data type	Start value	Access	Effective	HMI
DWORD	DW#16#00000000	-	-	-

<Axis name>.Config.General.LengthUnit ("Axis" technology object as of V2.0)				
The unit of measurement for the parameter selected in the configuration: • 1013 = "mm" • 1010 =: "m" • 1019 = "in" • 1018 = "ft" • 1005 = "°" (degrees) • -1 = "Pulse"				
Data type	Start value	Access	Effective	HMI
Int	1013	R	-	X

See also

Tags of the positioning axis technology object as of V4 (Page 10142)

Config.DriveInterface tags V1...3

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to tag in the user program:	
	RW	The tag can be read and written in the user program.
	R	The tag can be read in the user program.
	-	The tag cannot be used in the user program.
Effective	Specifies when a change to the tag takes effect.	
HMI	The tag can be used in an HMI system.	

<Axis name>.Config.DriveInterface.EnableOutput				
Tags cannot be evaluated in the user program.				
Data type	Start value	Access	Effective	HMI
-	-	-	-	-

<Axis name>.Config.DriveInterface.ReadyInput				
Tags cannot be evaluated in the user program.				
Data type	Start value	Access	Effective	HMI
-	-	-	-	-

See also

Tags of the positioning axis technology object as of V4 (Page 10142)

Config.Mechanics tags V1...3

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to tag in the user program:	
	RW	The tag can be read and written in the user program.
	R	The tag can be read in the user program.
	-	The tag cannot be used in the user program.
Effective	Specifies when a change to the tag takes effect.	
HMI	The tag can be used in an HMI system.	

<Axis name>.Config.Mechanics.PulsesPerDriveRevolution				
Increments per motor revolution				
Data type	Start value	Access	Effective	HMI
DInt	L#1000	R	-	X

<Axis name>.Config.Mechanics.LeadScrew				
Distance per revolution (specified in the configured unit of measurement)				
Data type	Start value	Access	Effective	HMI
Real	1.0E+001	R	-	X

<Axis name>.Config.Mechanics.InverseDirection				
Invert direction				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	R	-	X

See also

Tags of the positioning axis technology object as of V4 (Page 10142)

Config.DynamicLimits tags V1...3

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to tag in the user program:	
	RW	The tag can be read and written in the user program.
	R	The tag can be read in the user program.
	-	The tag cannot be used in the user program.
Effective	Specifies when a change to the tag takes effect.	
HMI	The tag can be used in an HMI system.	

<Axis name>.Config.DynamicLimits.MinVelocity				
Start/stop velocity of axis (specified in the configured unit of measurement)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	1.0E+001	R	-	X

<Axis name>.Config.DynamicLimits.MaxVelocity				
Maximum velocity of axis (specified in the configured unit of measurement)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	2.5E+002	R	-	X

See also

Tags of the positioning axis technology object as of V4 (Page 10142)

Config.DynamicDefaults tags V1...3

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to tag in the user program:	
	RW	The tag can be read and written in the user program.
	R	The tag can be read in the user program.
	-	The tag cannot be used in the user program.
Effective	Specifies when a change to the tag takes effect.	
	1	When axis is activated (<Axis name>.StatusBits.Activated tag changes from FALSE -> TRUE), disabled, or enabled
	2	When axis is enabled

	5	The next time an MC_MoveAbsolute, MC_MoveRelative, MC_MoveVelocity, MC_MoveJog, MC_Halt, MC_CommandTable or active MC_Home command is started (Mode = 3).
	6	When a MC_MoveJog command is stopped
HMI	The tag can be used in an HMI system.	

<Axis name>.Config.DynamicDefaults.Acceleration					
Acceleration of axis (specified in the configured dimension unit)					
Data type	**Start value**	**Access**	**Effective**		**HMI**
Real	4.8E+001	RW	5	CPU Firmware V1.0	X
			1, 5, 6	CPU firmware as of V2.0	

<Axis name>.Config.DynamicDefaults.Deceleration					
Deceleration of axis (specified in the configured dimension unit)					
Data type	**Start value**	**Access**	**Effective**		**HMI**
Real	4.8E+001	RW	5, 6	CPU Firmware V1.0	X
			1, 5, 6	CPU firmware as of V2.0	

<Axis name>.Config.DynamicDefaults.EmergencyDeceleration					
Emergency stop deceleration of axis (specified in the configured dimension unit)					
Data type	**Start value**	**Access**	**Effective**		**HMI**
Real	1.2E+002	RW	2, 5, 6	CPU Firmware V1.0	X
			1, 5, 6	CPU firmware as of V2.0	

<Axis name>.Config.DynamicDefaults.JerkActive ("Axis" technology object as of V2.0)				
TRUE = the jerk limit is activated				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	RW	1, 5	X

<Axis name>.Config.DynamicDefaults.Jerk ("Axis" technology object as of V2.0)				
Jerk during axis acceleration and deceleration ramp (specified in the configured unit of measurement)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	1.92E+002	RW	1, 5	X

See also

Tags of the positioning axis technology object as of V4 (Page 10142)

Config.PositionLimits_SW tags V1...3

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to tag in the user program:	
	RW	The tag can be read and written in the user program.
	R	The tag can be read in the user program.
	-	The tag cannot be used in the user program.
Effective	Specifies when a change to the tag takes effect.	
	1	When axis is activated (<Axis name>.StatusBits.Activated tag changes from FALSE -> TRUE), disabled, or enabled
	4	Upon the next start of a Motion Control command after a standstill of the axis. The axis standstill can be checked with tag <Axis name>. StatusBits.Standstill.
	5	The next time an MC_MoveAbsolute, MC_MoveRelative, MC_MoveVelocity, MC_MoveJog, MC_Halt, MC_CommandTable or active MC_Home command is started (Mode = 3).
HMI	The tag can be used in an HMI system.	

<Axis name>.Config.PositionLimits_SW.Active					
TRUE = The software limit switches are activated					
Data type	**Start value**	**Access**	**Effective**		**HMI**
Bool	FALSE	RW	4	CPU Firmware V1.0	X
			1, 5, 6	CPU firmware as of V2.0	

<Axis name>.Config.PositionLimits_SW.MinPosition					
Position of lower software limit switch (specified in the configured unit of measurement)					
Data type	**Start value**	**Access**	**Effective**		**HMI**
Real	-1.0E+004	RW	4	CPU Firmware V1.0	X
			1, 5, 6	CPU firmware as of V2.0	

<Axis name>.Config.PositionLimits_SW.MaxPosition					
Position of upper software limit switch (specified in the configured unit of measurement)					
Data type	**Start value**	**Access**	**Effective**		**HMI**
Real	1.0E+004	RW	4	CPU Firmware V1.0	X
			1, 5, 6	CPU firmware as of V2.0	

See also

Tags of the positioning axis technology object as of V4 (Page 10142)

Config.PositionLimits_HW tags V1...3

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to tag in the user program:	
	RW	The tag can be read and written in the user program.
	R	The tag can be read in the user program.
	-	The tag cannot be used in the user program.
Effective	Specifies when a change to the tag takes effect.	
	1	When axis is activated (<Axis name>.StatusBits.Activated tag changes from FALSE -> TRUE), disabled, or enabled
	3	After axis enable (the axis must have previously been at a standstill). The axis standstill can be checked with tag <Axis name>. StatusBits.Standstill.
	4	Upon the next start of a Motion Control command after a standstill of the axis. The axis standstill can be checked with tag <Axis name>. StatusBits.Standstill.
	5	The next time an MC_MoveAbsolute, MC_MoveRelative, MC_MoveVelocity, MC_MoveJog, MC_Halt, MC_CommandTable or active MC_Home command is started (Mode = 3).
HMI	The tag can be used in an HMI system.	

<Axis name>.Config.PositionLimits_HW.Active					
TRUE = The hardware limit switches are active.					
Data type	**Start value**	**Access**	**Effective**		**HMI**
Bool	FALSE	RW	3, 4	CPU Firmware V1.0	X
			1, 5, 6	CPU firmware as of V2.0	

<Axis name>.Config.PositionLimits_HW.MinSwitchedLevel				
TRUE = 24 V at CPU input corresponds to lower hardware limit switch approached FALSE = 0 V at CPU input corresponds to lower hardware limit switch approached				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.Config.PositionLimits_HW.MinFallingEvent				
Tag cannot be evaluated in the user program.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
DWord	DW#16#00000000	-	-	-

<Axis name>.Config.PositionLimits_HW.MinRisingEvent				
Tag cannot be evaluated in the user program.				
Data type	Start value	Access	Effective	HMI
DWord	DW#16#00000000	-	-	-

<Axis name>.Config.PositionLimits_HW.MaxSwitchedLevel				
TRUE = 24 V at CPU input corresponds to upper hardware limit switch approached FALSE = 0 V at CPU input corresponds to upper hardware limit switch approached				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	R	-	X

<Axis name>.Config.PositionLimits_HW.MaxFallingEvent				
Tag cannot be evaluated in the user program.				
Data type	Start value	Access	Effective	HMI
DWord	DW#16#00000000	-	-	-

<Axis name>.Config.PositionLimits_HW.MaxRisingEvent				
Tag cannot be evaluated in the user program.				
Data type	Start value	Access	Effective	HMI
DWord	DW#16#00000000	-	-	-

See also

Tags of the positioning axis technology object as of V4 (Page 10142)

Config.Homing tags V1...3

Legend

Data type	Data type of tag	
Start value	Start value of tag The start value can be overwritten by the axis configuration.	
Access	Access to tag in the user program:	
	RW	The tag can be read and written in the user program.
	R	The tag can be read in the user program.
	-	The tag cannot be used in the user program.
Effective	Specifies when a change to the tag takes effect.	
	1	When axis is activated (<Axis name>.StatusBits.Activated tag changes from FALSE -> TRUE), disabled, or enabled
	7	When a passive homing command is started
	8	When an active homing command is started
HMI	The tag can be used in an HMI system.	

<Axis name>.Config.Homing.AutoReversal					
TRUE = Direction reversal at hardware limit switch enabled (active homing) FALSE = Direction reversal at hardware limit switch disabled (active homing)					
Data type	Start value	Access	Effective		HMI
Bool	TRUE	R	-	Technology object "Axis" V1.0	X
		RW	1, 8	Technology object "Axis" V2.0	

<Axis name>.Config.Homing.Direction					
TRUE = Positive approach direction to search for reference point switch and positive homing direction (active homing) FALSE = Negative approach direction to search for reference point switch and positive homing direction (active homing)					
Data type	Start value	Access	Effective		HMI
Bool	TRUE	R	-	Technology object "Axis" V1.0	X
		RW	1, 8	Technology object "Axis" V2.0	

<Axis name>.Config.Homing.SideActiveHoming ("Axis" technology object as of V2.0)				
TRUE = Homing on high end of the reference point switch (active homing) TRUE = Homing on lower end of the reference point switch (active homing)				
Data type	Start value	Access	Effective	HMI
Bool	TRUE	RW	1, 8	X

<Axis name>.Config.Homing.SidePassiveHoming ("Axis" technology object as of V2.0)				
TRUE = Homing on high end of the reference point switch (passive homing) TRUE = Homing on lower end of the reference point switch (passive homing)				
Data type	Start value	Access	Effective	HMI
Bool	TRUE	RW	1, 7	X

<Axis name>.Config.Homing.RisingEdge (as of technology object "Axis" V1.0)				
TRUE = Homing with negative signal edge of the reference point switch (active homing) FALSE = Homing with positive signal edge of the reference point switch (active homing) For information on the effect of the tag on passive homing, refer to the description in "Configuration - Homing".				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	R	-	X

<Axis name>.Config.Homing.Offset				
Home position offset /specified in the configured unit of measurement (active homing)				
Data type	Start value	Access	Effective	HMI

<Axis name>.Config.Homing.Offset					
Real	0.0	R RW	-	Technology object "Axis" V1.0	X
			1, 8	Technology object "Axis" V2.0	

<Axis name>.Config.Homing.FastVelocity					
Approach velocity / specified in the configured unit of measurement (active homing)					
Data type	**Start value**	**Access**	**Effective**		**HMI**
Real	2.0E+002	R	-	Technology object "Axis" V1.0	X
		RW	1, 8	Technology object "Axis" V2.0	

<Axis name>.Config.Homing.SlowVelocity					
Homing velocity / specified in the configured unit of measurement (active homing)					
Data type	**Start value**	**Access**	**Effective**		**HMI**
Real	4.0E+001	R	-	Technology object "Axis" V1.0	X
		RW	1, 8	Technology object "Axis" V2.0	

<Axis name>.Config.Homing.FallingEvent				
Tag cannot be evaluated in the user program.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
DWord	DW#16#00000000	-	-	-

<Axis name>.Config.Homing.RisingEvent				
Tag cannot be evaluated in the user program.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
DWord	DW#16#00000000	-	-	

See also

Tags of the positioning axis technology object as of V4 (Page 10142)

MotionStatus tags V1...3

Legend

Data type	Data type of tag	
Start value	Start value of tag	
Access	Access to tag in the user program:	
	RW	The tag can be read and written in the user program.

	R	The tag can be read in the user program.
	-	The tag cannot be used in the user program.
Effective	Specifies when a change to the tag takes effect.	
HMI	The tag can be used in an HMI system.	

<Axis name>.MotionStatus.Position				
Current position of the axis (specified in the configured unit of measurement) If the axis is not homed, the tag indicates the position value relative to the enable position of the axis.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.0	R	-	X

<Axis name>.MotionStatus.Velocity				
Current velocity of the axis (specified in the configured unit of measurement)				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.0	R	-	X

<Axis name>.MotionStatus.Distance				
Current distance to the target position of the axis (specified in the configured unit of measurement) The value of the tag is only valid during execution of a positioning command with "MC_MoveAbsolute" or "MC_MoveRelative" or of the axis command table.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.0	R	-	X

<Axis name>.MotionStatus.TargetPosition				
Target position of axis (specified in the configured unit of measurement) The value of the tag is only valid during execution of a positioning command with "MC_MoveAbsolute" or "MC_MoveRelative" or of the axis command table.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.0	R	-	X

See also

Motion status (Page 10117)

Tags of the positioning axis technology object as of V4 (Page 10142)

StatusBits tags V1...3

Legend

Data type	Data type of tag	
Start value	Start value of tag	
Access	Access to tag in the user program:	
	RW	The tag can be read and written in the user program.
	R	The tag can be read in the user program.

	-	The tag cannot be used in the user program.
Effective	Specifies when a change to the tag takes effect.	
HMI	The tag can be used in an HMI system.	

<Axis name>.StatusBits.Activated				
TRUE = The axis is activated. It is connected to the assigned PTO (Pulse Train Output). The data of the technology data block will be updated cyclically.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.StatusBits.Enable				
TRUE = The axis is enabled and ready to take on Motion Control commands.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.StatusBits.HomingDone				
TRUE = The axis is homed and is capable of executing absolute positioning commands. The axis does not have to be homed for relative positioning. The status is FALSE during active homing. The status remains TRUE during passive homing if the axis has already been homed.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.StatusBits.Done				
TRUE = No Motion Control command is active on the axis.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.StatusBits.Error				
TRUE = An error occurred in the axis technology object. Detailed information about the error is available in automatic mode in the "ErrorID" and "ErrorInfo" parameters of the Motion Control instructions. In manual mode, the "Error message" box of the axis control panel displays detailed information about the cause of error.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.StatusBits.StandStill				
TRUE = The axis is at a standstill.				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Bool	FALSE	R	-	X

<Axis name>.StatusBits.PositioningCommand				
TRUE = The axis is executing a positioning command.				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	R	-	X

<Axis name>.StatusBits.SpeedCommand				
TRUE = The axis is executing a travel command at predefined velocity.				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	R	-	X

<Axis name>.StatusBits.Homing				
TRUE = The axis is executing a homing command of the "MC_Home" Motion Control instruction or axis command table.				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	R	-	X

<Axis name>.StatusBits.CommandTableActive				
TRUE = The axis is controlled by Motion Control instruction "MC_CommandTable".				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	R	-	X

<Axis name>.StatusBits.ConstantVelocity				
TRUE = The axis travels at constant velocity.				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	R	-	X

<Axis name>.StatusBits.Acceleration				
TRUE = The axis accelerates.				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	R	-	X

<Axis name>.StatusBits.Deceleration				
TRUE = The axis decelerates (slows down).				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	R	-	X

<Axis name>.StatusBits.ControlPanelActive				
TRUE = The "Manual control" mode has been enabled in the axis command table. The axis command table has control priority over the "Axis" technology object. The axis cannot be controlled from the user program.				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	R	-	X

<Axis name>.StatusBits.DriveReady				
TRUE = The drive is ready.				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	R	-	X

<Axis name>.StatusBits.RestartRequired				
TRUE = Values were modified in the load memory. To download the values in the CPU RUN operating mode to the work memory, you need to restart the axis. Use the Motion Control instruction MC_Reset to do this.				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	R	-	X

See also

Status and error bits (technology objects as of V4) (Page 10114)

Tags of the positioning axis technology object as of V4 (Page 10142)

ErrorBits tags V1...3

Legend

Data type	Data type of tag	
Start value	Start value of tag	
Access	Access to tag in the user program:	
	RW	The tag can be read and written in the user program.
	R	The tag can be read in the user program.
	-	The tag cannot be used in the user program.
Effective	Specifies when a change to the tag takes effect.	
HMI	The tag can be used in an HMI system.	

<Axis name>.ErrorBits.SystemFault				
TRUE = Internal system error.				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	R	-	X

<Axis name>.ErrorBits.ConfigFault				
TRUE = Incorrect configuration of axis.				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	R	-	X

<Axis name>.ErrorBits.DriveFault				
TRUE = The drive has reported an error due to loss of its "Drive ready" signal.				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	R	-	X

<Axis name>.ErrorBits.SwLimitMinReached				
TRUE = The lower software limit switch has been reached.				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	R	-	X

<Axis name>.ErrorBits.SwLimitMinExceeded				
TRUE = The lower software limit switch has been exceeded.				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	R	-	X

<Axis name>.ErrorBits.SwLimitMaxReached				
TRUE = The upper software limit switch has been reached.				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	R	-	X

<Axis name>.ErrorBits.SwLimitMaxExceeded				
TRUE = The upper software limit switch has been exceeded.				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	R	-	X

<Axis name>.ErrorBits.HwLimitMin				
TRUE = The lower hardware limit switch has been approached.				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	R	-	X

<Axis name>.ErrorBits.HwLimitMax				
TRUE = The upper hardware limit switch has been approached.				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	R	-	X

<Axis name>.ErrorBits.HwUsed				
TRUE = A second axis is using the same PTO (Pulse Train Output) and is enabled with "MC_Power".				
Data type	Start value	Access	Effective	HMI
Bool	FALSE	R	-	X

See also

Status and error bits (technology objects as of V4) (Page 10114)

Tags of the positioning axis technology object as of V4 (Page 10142)

Internal tags V1...3

The "Internal" tags contain no user-relevant data; these tags cannot be accessed in the user program.

See also

Tags of the positioning axis technology object as of V4 (Page 10142)

ControlPanel tags V1...3

The "ControlPanel" tags contain no user-relevant data; these tags cannot be accessed in the user program.

See also

Tags of the positioning axis technology object as of V4 (Page 10142)

Update of the technology object tags

The status and error information of the axis indicated in the technology object tags is updated at each cycle control point.

The change in values of editable configuration tags does not take effect immediately. For information on the conditions under which a change takes effect, refer to the detailed description of the relevant tag.

Tags of the command table technology object V1...3

Config.Command.Command[1 ... 32] tags V1...3

Legend

Data type	Data type of the tag	
Start value	Start value of tag The start value can be overwritten by the configuration of the command table.	
Access	Access to the tag in the user program:	
	RW	The tag can be read and written in the user program.
	R	The tag can be read in the user program.
	-	The tag cannot be used in the user program.

Effective	Specifies when a change in the tag takes effect.
HMI	The tag can be used in an HMI system.

<Command table>.Config.Command.Command[x].Type				
Command type • 0 = "Empty" command • 2 = "Hold" command • 5 = "Relative positioning" command • 6 = "Absolute positioning" command • 7 = "Velocity setpoint" command • 151 = "Wait" command				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Int	0	RW	-	X

<Command table>. Config.Command.Command[x].Position				
Command target position / travel path				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real		RW	-	X

<Command table>. Config.Command.Command[x].Velocity				
Command velocity				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.0	RW	-	X

<Command table>. Config.Command.Command[x].Duration				
Command duration				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Real	0.0	RW	-	X

<Command table>. Config.Command.Command[x].BufferMode				
Value for command "Next step" • 0 = "Complete command" • 1 = "Blend movement"				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Int	0	RW	-	X

<Command table>. Config.Command.Command[x].StepCode				
Command step code				
Data type	**Start value**	**Access**	**Effective**	**HMI**
Word	0	RW	-	X

See also

Tags of the command table technology object as of V4 (Page 10161)

11.4.1.2 Using S7-1500 Motion Control

Introduction

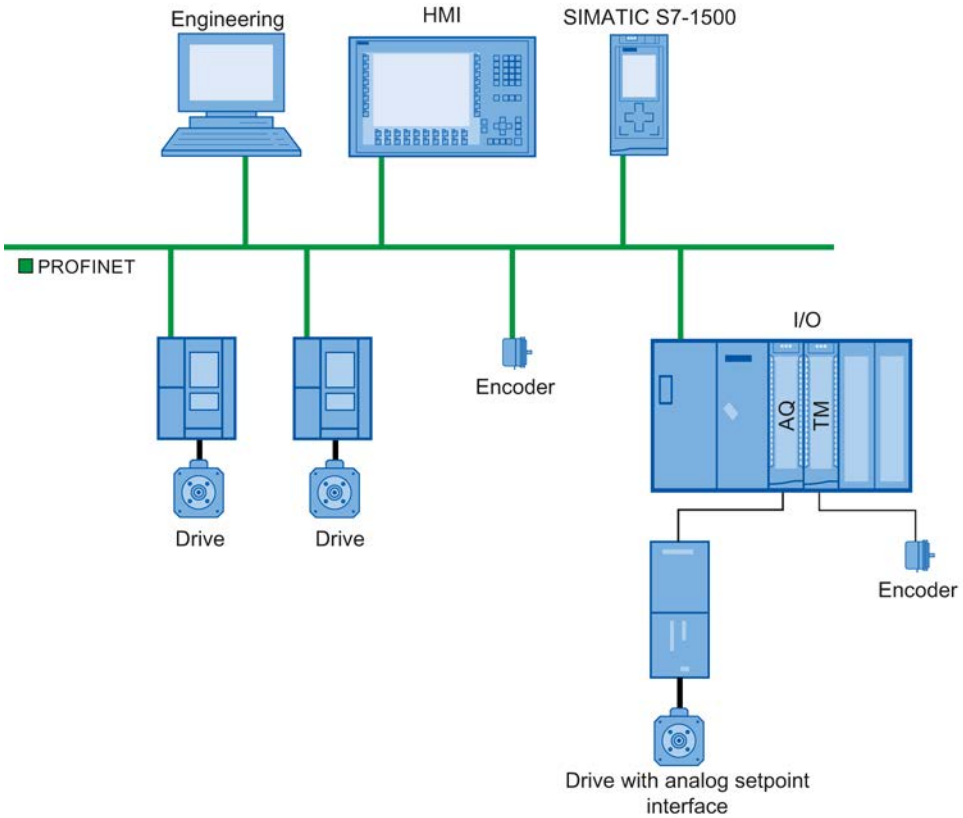
Motion Control Functionality of the CPU S7-1500

The Motion Control functionality supports the controlled positioning and movement of axes, and is a component of each CPU S7-1500.

Configuration options

You can connect drives and encoders using PROFIBUS DP and PROFINET IO. You can connect drives with analog setpoint interfaces using an analog output (AQ). You can additionally read encoders in by means of a technology module (TM).

The following figure shows an example configuration:

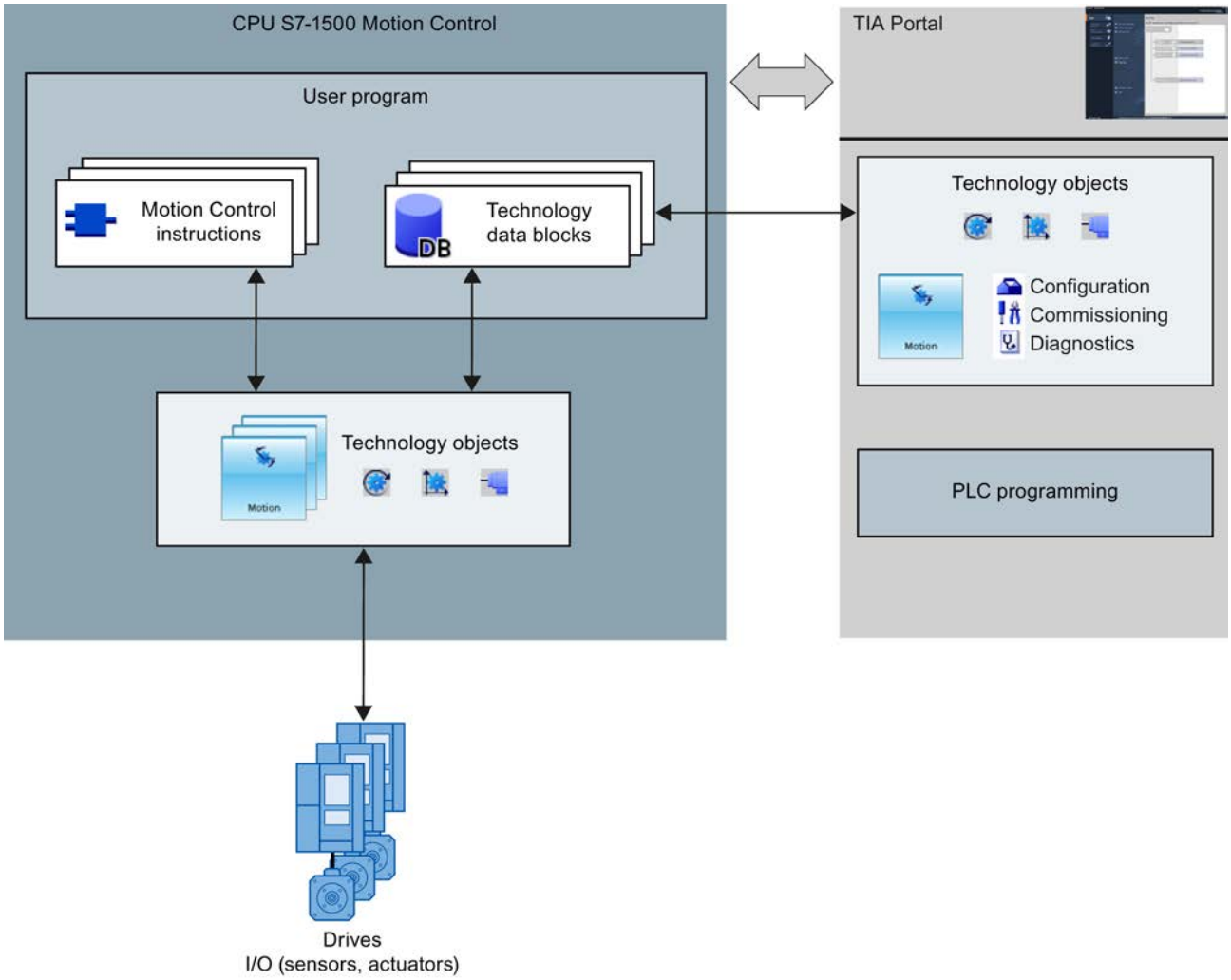


Principle of operation of Motion Control

Overview

With the TIA Portal you create a project, configure technology objects, and load the configuration into the CPU. The processing of the Motion Control functionality occurs in the CPU.
You control the technology objects by using the Motion Control instructions in your user program.
The TIA portal provides additional functions for commissioning, optimization (Page 10338), and diagnostics (Page 10349).

The following figure shows schematically the user interfaces and the integration of Motion Control into the CPU S7-1500. The concepts are then briefly explained:



TIA Portal

The TIA Portal supports you in the planning and commissioning of Motion Control functionality:

- Integrating and configuring hardware
- Creating and configuring technology objects
- Creating the user program
- Diagnostics

Motion Control instructions

With the Motion Control instructions you perform the desired functionality in the technology objects. The Motion Control instructions are available in the TIA Portal under "Instructions > Technology > Motion Control > S7-1500 Motion Control".

The Motion Control instructions conform to PLCopen (version 2.0).

Technology objects

Technology objects represent each of the real objects (e.g. a drive) in the controller. You can call the functions of the technology objects by means of Motion Control instructions in your user program. The technology objects provide open- and closed-loop control of the movement of the real objects, and report status information (e.g. the current position).

The configuration of the technology objects reflects the properties of the real object. The configuration data are stored in a technology data block.

The following technology objects are available for Motion Control:

- **Speed-controlled axis technology object**
  The speed-controlled axis technology object ("SpeedAxis") permits the specification of the speed for a drive. The movement of the axis can be programmed with Motion Control instructions.
- **Positioning axis technology object**
  The positioning axis technology object ("PosAxis") permits the position-controlled positioning of a drive. Positioning jobs can be assigned to the axis with Motion Control instructions in the user program.
- **External encoder technology object**
  The external encoder technology object ("ExternalEncoder") detects a position, and makes it available to the controller. The detected position can be evaluated in the user program.

Scale

For information on the number of technology objects that may be used, refer to the technical specifications of the utilized CPU.

Technology data block

The technology data block represents the technology object and contains all configuration data, setpoint and actual values, and status information of the technology object. The technology data block is automatically created when the technology object is created. In your user program, you can access the technology data block's data.

User program

The Motion Control instructions and the technology data block represent the programming interfaces for the technology objects. Using the Motion Control instructions, your user program can initiate and track Motion Control jobs in technology objects. The technology data block represents the technology object.

Drives and encoders

Drives permit the movement of the axis. They are integrated into the hardware configuration as slaves.

When you perform a Motion Control job in your user program, the technology object takes over the control of the drive and the reading in of values from position encoders.

Drives and encoders are connected by means of PROFIdrive frames. The following interfaces are possible:

- PROFIBUS DP
- PROFINET IO
- Technology module (TM)

For an analog drive connection, the setpoint can be specified via an analog output with an enabling signal. Analog inputs and outputs are made available by means of corresponding IO modules.

A drive is also called an actuator, and an encoder is also called a sensor.

Basics

Functions

You can perform the Motion Control function by means of Motion Control instructions in your user program or the TIA portal (under Commissioning).

The following table shows the functions that are supported by technology objects:

Function	Speed-controlled axis	Positioning axis	External encoder
Motion Control instructions (user program)			
"MC_Power (Page 4885)" Enable, disable technology objects	X	X	X
"MC_Home (Page 4890)" Homing technology objects, setting home position	-	X	X
"MC_MoveJog (Page 4894)" Moving axes in jog mode	X	X	-
"MC_MoveVelocity (Page 4899)" Moving axes with the specified speed	X	X	-
"MC_MoveRelative (Page 4904)" Position axis relatively	-	X	-
"MC_MoveAbsolute (Page 4908)" Position axis absolutely	-	X	-
"MC_Halt (Page 4912)" Stop axis	X	X	-
"MC_Reset (Page 4916)" Acknowledging alarms, restarting technology objects	X	X	X
TIA Portal			
"Axis control panel (Page 10340)" Moving and homing axes using the TIA portal	X	X	-
"Optimization (Page 10344)" Optimization of position control ("Kv" factor)	-	X	-

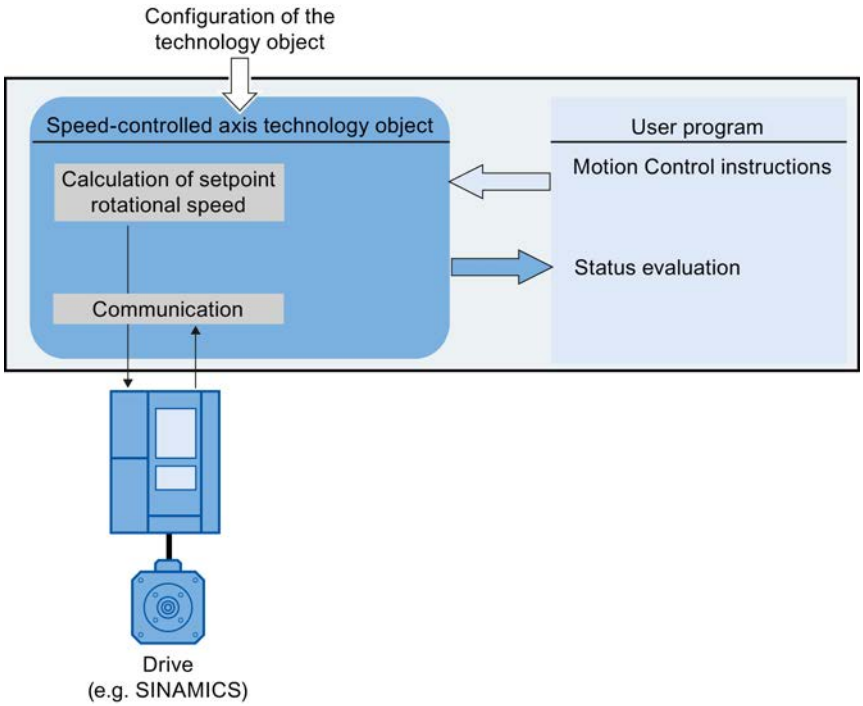
Speed-controlled axis technology object

The speed-controlled axis technology object calculates speed setpoints, taking account of the specified dynamics, and outputs them to the drive. All movements of the speed-controlled axis occur under speed control. The system takes account of an existing load gear.

A drive is assigned to each speed-controlled axis by means of a PROFIdrive message frame, or by means of an analog setpoint interface.

The speed is specified in revolutions per unit of time.

The following figure shows the basic principle of operation of the speed-controlled axis technology object:



Positioning axis technology object

The positioning axis technology object calculates position setpoints, taking account of the specified dynamics, and outputs corresponding speed control setpoints to the drive. All movements of the positioning axis occur under position control. For absolute positioning, the physical position must be known to the positioning axis technology object.

Each positioning axis is assigned a drive by means of a PROFIdrive message frame or by means of an analog setpoint interface, and an encoder by means of a PROFIdrive message frame.

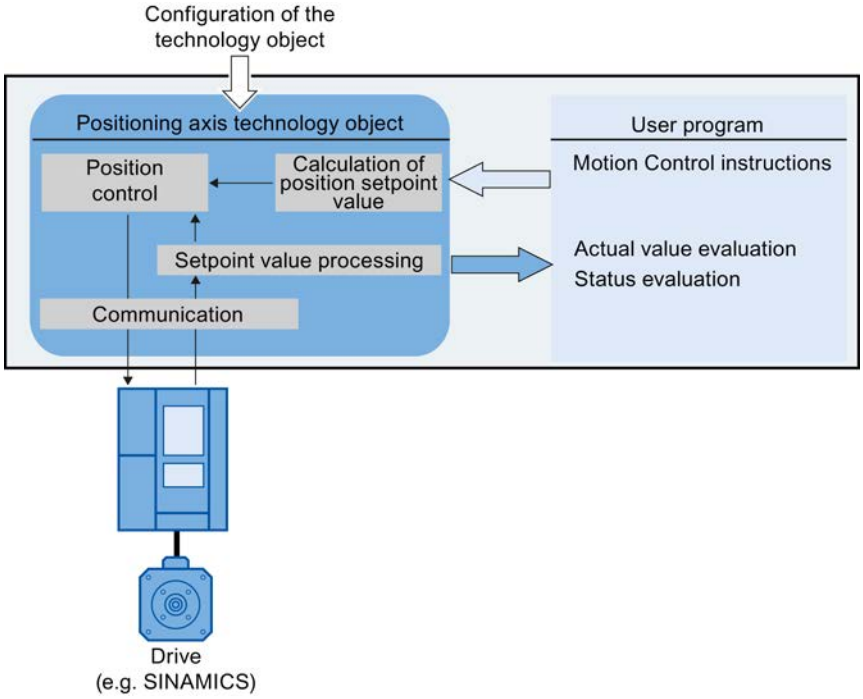
The relationship between the encoder values and a defined position is created by assigning parameters for the mechanical properties and the encoder settings, as well as a homing

process. The technology object can also perform movements without a position relationship, and relative position movements, even without being in a homed status.

Note

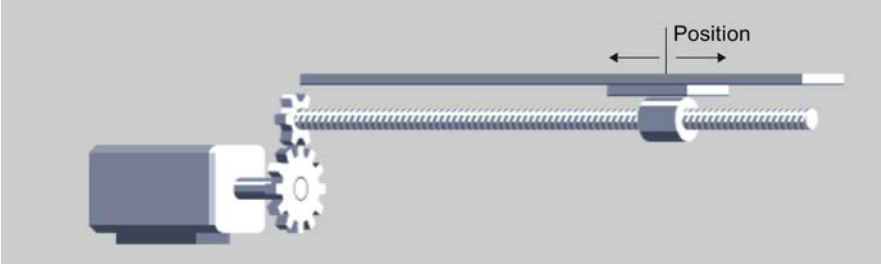
The positioning axis technology object and external encoder are independent of each other, and cannot be coupled to each other. The actual position of an external encoder cannot influence a positioning axis in its position control.

The following figure shows the basic principle of operation of the positioning axis technology object:



Depending on the execution of the mechanics, a positioning axis is implemented as a linear axis or rotational axis:

- Linear axis



For linear axes, the position of the axis is specified as a linear measure, e.g. millimeters (mm).

- Rotary axis

For rotary axes, the position of the axis is specified as an angular measure, e.g. degrees (°).

See also

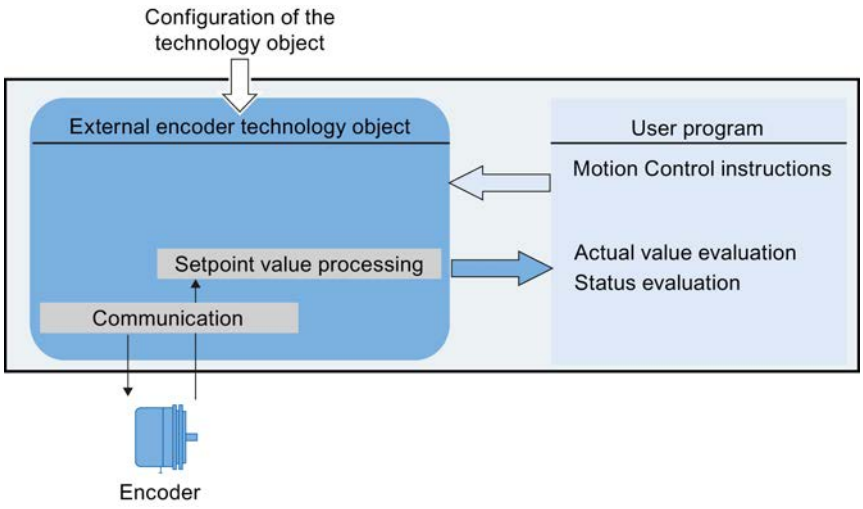
Modulo setting (Page 10215)

External encoder technology object

The external encoder technology object detects a position, and makes it available to the controller.

The relationship between the encoder values and a defined position is created by assigning parameters for the mechanical properties and the encoder settings, as well as a homing process.

The following figure shows the basic principle of operation of the external encoder technology object:



Specification of the position occurs according to the selected system of units:

- Linear system of units
  The position is specified as a linear measure, e.g. millimeters (mm).
- Rotary system of units
  The position is specified as an angular measure, e.g. degrees (°).

Modulo setting

The positioning axis technology object and external encoder can be configured with the "modulo" setting.

When "modulo" is enabled, the position value of the technology object is represented by means of a recurring modulo range.

The modulo range is defined by the initial modulo value and the modulo length.

As an example, the position value of a rotary axis with an initial value = 0° and length = 360° can be represented in the modulo range from 0° to 359.9°. In this example, the resolution of the encoder is assumed to have a value of 0.1° / encoder step.

Units of measure

The following table shows the supported units of measure for distance and velocity:

Distance	Velocity
nm, µm, mm, m, km	mm/s, mm/min, mm/h, m/s, m/min, m/h, km/min, km/h
in, ft, mi	in/s, in/min, ft/s, ft/min, mi/h
°, rad	°/s, °/min, rad/s, rad/min

- The acceleration is correspondingly configured as a distance/s².
- Jerk is correspondingly configured as a distance/s³.
- The speed is configured as revolutions per unit of time: 1/s, 1/min, 1/h

Note

When setting or changing the units of measurement, note the effect on the depiction and the user program:

- Depiction in the technology data block
- Supply to the parameters in the user program
- Input and display of the position in the TIA Portal

All information and displays are output in the selected unit of measure.

Drive and encoder connection

Brief description

A speed-controlled axis is assigned a drive.
A positioning axis is assigned a drive and an encoder.
An external encoder is assigned an encoder value.

The setpoint value at the drive is specified either with PROFIdrive message frames, or with an analog output.

The following connection options are available for an encoder:

- Encoder to drive
- Encoder on technology module
- PROFIdrive encoder directly to PROFIBUS DP / PROFINET IO

The encoder value is transmitted exclusively via PROFIdrive message frames.

PROFIdrive

PROFIdrive is the standardized standard profile for drive technology in the connection of drives and encoders via PROFIBUS DP and PROFINET IO.

Drives that support the PROFIdrive profile are connected according to the PROFIdrive standard.

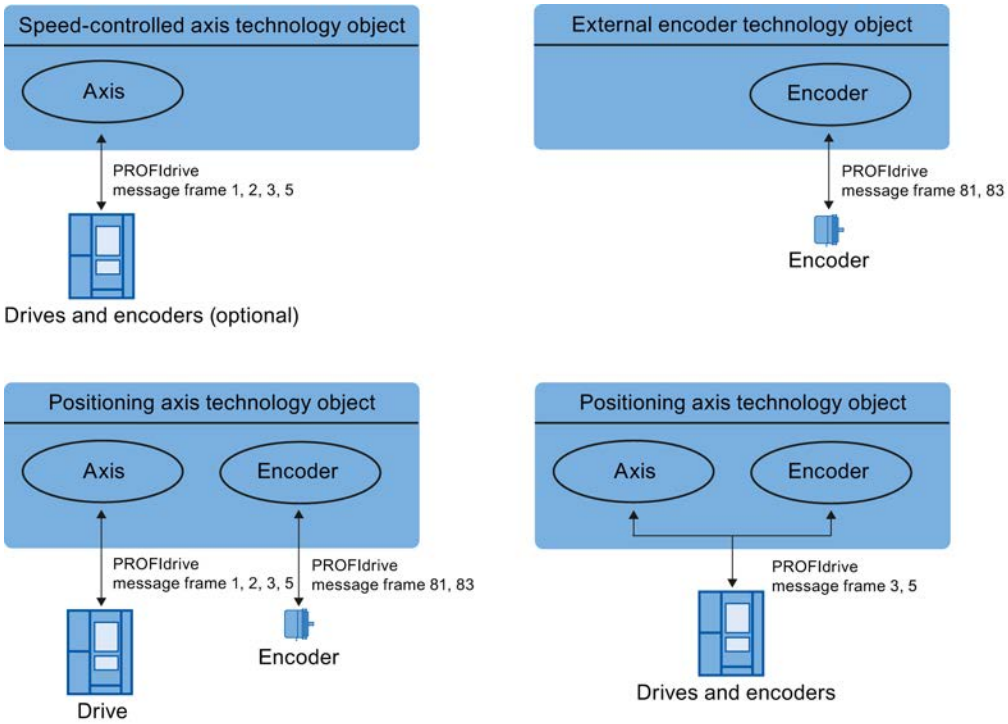
Communication between controller and drive/encoder is by means of various PROFIdrive message frames. Each of the message frames has a standardized structure. Depending on the application, you can select the applicable message frame. Control words and status words as well as setpoints and actual values are transmitted in the PROFIdrive message frames.

The PROFIdrive profile likewise supports the "Dynamic Servo Control" (DSC) control concept. DSC uses rapid position control in the drive. This can be used to solve highly dynamic positioning jobs.

Frames

The transmission of the encoder value occurs either in a frame together with the setpoint (frame 3 or frame 5), or in a separate encoder frame (frame 81 or frame 83).

The following figure represents the relationship between the technology objects and the drives / encoders:



Explanation of the figure:

- The setpoint of a speed axis is transmitted to a drive via PROFIdrive frame 1, 2, 3 or 5.
- The encoder value of an external encoder is transmitted via PROFIdrive frame 81 or 83.
- The setpoint of a positioning axis is transmitted to a drive via PROFIdrive frame 1, 2, 3 or 5.
- Encoder value of the positioning axis
  The encoder value can be transmitted in the following PROFIdrive frames:
  - Transmission in the same PROFIdrive frame, in which the setpoint is also transmitted. For example, with PROFIdrive frame 3 or 5
  - Transmission in PROFIdrive frame 81 or 83.

Frame types

The following table shows the supported PROFIdrive frame types for the assignment of drives and encoders:

Frame	Brief description
Standard frames	
1	• 16 bit speed setpoint (NSET) • 16 bit actual speed value (NACT),
2	• 32 bit speed setpoint (NSET) • 32 bit actual speed (NACT) • Signs of life
3	• 32 bit speed setpoint (NSET) • 32 bit actual speed (NACT) • Encoder actual value • Signs of life
5	• 32 bit speed setpoint (NSET) • 32 bit actual speed (NACT) • Dynamic Servo Control (DSC) • Encoder actual value • Signs of life
Standard frames encoder	
81	• Encoder actual value • Signs of life
83	• 32 bit actual speed (NACT) • Encoder actual value • Signs of life

When connecting by means of a PROFIdrive frame, the drives and encoders are handled and switched on in accordance with the PROFIdrive profile.

Setting reference values

The reference values for the drive connection and encoder connection must be set identically in the controller, and in the drive or encoder.

The setpoint speed setpoint NSET and the actual speed value NACT are transmitted in the PROFIdrive frame as a percentage value relative to the reference speed. The reference value for the speed must be set identically in the controller and in the drive.

The resolution of the actual value in the PROFIdrive frame must likewise be set identically in the controller and in the drive or encoder.

Frame settings

The controller settings are set in the TIA Portal under "Technology object > Configuration > Hardware interface > Data transmission".

The settings for drive and encoder are set in the configuration for the respective hardware.

The following table contrasts the controller settings and corresponding PROFIdrive parameters (<TO> stands for the applicable technology object):

Setting in the TIA Portal	Controller tag in the technology data block	PROFIdrive parameter
Frame	<TO>.Actor.Interface.Telegram	P922
Reference speed in [rpm]	<TO>.Actor.DriveParameter.ReferenceSpeed	(SINAMICS drives: P2000)
Maximum speed of the motor in [rpm]	<TO>.Actor.DriveParameter.MaxSpeed	(SINAMICS drives: P1082)
Encoder frame	<TO>.Sensor[n].Interface.Telegram	P922
Encoder system (linear or rotary encoder)	<TO>.Sensor[n].System	P979[1] Bit0
Encoder type 0: incremental 1: absolute 2: cyclic absolute	<TO>.Sensor[n].Type	P979[5]
Resolution (linear encoder) The grid spacing is specified on the nameplate of the sensor as a separation distance of the marks on the linear measuring system.	<TO>.Sensor[n].Parameter.Resolution	P979[2]
Increments per revolution (rotary encoder)	<TO>.Sensor[n].Parameter.StepsPerRevolution	P979[2]
Number of bits for fine resolution XIST1 (cyclic actual encoder value, linear or rotary encoder)	<TO>.Sensor[n].Parameter.FineResolutionXist1	P979[3]
Number of bits for fine resolution XIST2 (absolute encoder value, linear or rotary encoder)	<TO>.Sensor[n].Parameter.FineResolutionXist2	P979[4]
Differentiable encoder revolutions (rotary absolute value encoder)	<TO>.Sensor[n].Parameter.DeterminableRevolutions	P979[5]

Tags

The following technology object tags are relevant for the connection of drives and encoders:

Drive frame	
<TO>.Actor.Interface.Telegram	Frame number
<TO>.Actor.DriveParameter.Reference Speed	Reference velocity / reference speed for the velocity / speed (NSET), which is transmitted as a percentage value
<TO>.Actor.DriveParameter.MaxSpee d	Maximum value for the setpoint speed of the drive (N-set) (PROFIdrive: MaxSpeed ≤ 2 × ReferenceSpeed Analog setpoint: MaxSpeed ≤ 1.17 × ReferenceSpeed)
Encoder frame	
<TO>.Sensor[n].Interface.Telegram	Frame number
<TO>.Sensor[n].System	Encoder system linear or rotary
<TO>.Sensor[n].Type	Encoder type, incremental, absolute or cyclically absolute

<TO>.Sensor[n].Parameter.StepsPerRevolution	Increments per revolution for rotary encoder
<TO>.Sensor[n].Parameter.DeterminableRevolutions	Number of differentiable encoder revolutions for a multi-turn absolute value encoder
<TO>.Sensor[n].Parameter.Resolution	Resolution for linear encoder The grid spacing corresponds to the interval between two marks.
Fine resolution	
<TO>.Sensor[n].Parameter.FineResolutionXist1	Number of bits for fine resolution XIST1 (cyclic actual encoder value)
<TO>.Sensor[n].Parameter.FineResolutionXist2	Number of bits for fine resolution XIST2 (absolute value of the encoder)

Actual values

Brief description

For position-controlled motion and positioning, the controller must know the actual position value.

The actual position value is provided by a PROFIdrive frame. The actual value is updated after a one-off transition of the operating mode from STOP to RUN.

The actual values are represented incrementally or absolutely in the PROFIdrive frame, and standardized in the controller to the technological unit. Homing is used to convert the actual value to the physical position of the axis, or of the external encoder.

The controller supports the following types of actual values:

- Incremental actual value
- Absolute actual value with the setting absolute (measuring range > traversing range of the axis)
- Absolute actual value with the setting absolute (measuring range < traversing range of the axis)

Incremental actual value

The actual value in the PROFIdrive frame is based on an incremental value.

After POWER ON, position zero is displayed. The actual value is updated after a one-off transition of the operating mode from STOP to RUN. The relationship between the technology object and the mechanical position must be recreated by means of homing (Page 10223).

Absolute actual value

The actual value in the PROFIdrive frame is based on an absolute value.

After POWER ON, position zero is displayed. The actual value is updated after a one-off transition of the operating mode from STOP to RUN. The supplied absolute value is assigned to the associated mechanical axis position by means of absolute value adjustment

(Page 10238). The absolute value adjustment must be performed once. The absolute value offset is retentively saved beyond the switching on/off of the controller.

Differentiation of absolute values:

- The measuring range of the encoder is larger than the traversing range of the axis: Absolute value with absolute setting
- The measuring range of the encoder is smaller than the traversing range of the axis: Absolute value with cyclically absolute setting

Absolute actual value with the setting absolute (measuring range > traversing range)

The axis position results directly from the actual encoder value. The traversing range must be within an encoder measuring range. This means that the zero point of the encoder must not be located in the traversing range.

When the controller is switched on, the axis position is determined from the absolute encoder value.

Absolute actual value with the setting absolute (measuring range < traversing range)

The encoder supplies an absolute value within its measuring range. The controller includes the traversed measuring ranges and thus determines the correct axis position beyond the measuring range.

When the controller is switched off, the traversed measuring ranges are saved in the retentive memory area of the controller.
At the next power-on, the stored overflows are taken into account in the calculation of the actual position value.

NOTICE
Movements of the axis while the controller is switched off can skew the actual value If the axis or the encoder is moved by more than half of the encoder measuring range while the controller is switched off, then the actual value in the controller is no longer in accord with the mechanical axis position.

Tags

The tags named in the Homing (Page 10239) section are relevant for adapting actual values.

Mechanics

Brief description

For the application's view onto the position of the technology object, it is decisive, whether the position is represented as a unit of length (linear axis) or as a rotary unit (rotary axis).

Examples of units of length: mm, m, km

Examples of rotary units: °, rad

For the determination of the physical position from an actual encoder value, the system must know the various properties and configurations of the mechanics.

Positioning axis

The following configuration options for mechanics are supported:

- Load gear
- Leadscrew (linear axes only)
- Encoder configurations:
  - On the motor side (before the load gear)
  - On the load side (after the load gear and as applicable the leadscrew)
  - External (e.g. odometer)
- Inversion of the setpoint
- Inversion of the setpoint

External encoder

The following configuration options for mechanics are supported:

- Measuring gearbox (for rotary encoders)
- Leadscrew (only with linear system of units and rotatory encoders)
- Inversion of the setpoint

Speed-controlled axis

The following configuration options for mechanics are supported:

- Load gear
- Inversion of the setpoint

Tags

The following technology object tags are relevant for the setting of the mechanics:

Type of motion	
<TO>.Properties.MotionType	Indication of linear or rotary motion 0: Linear motion 1: Rotary motion
Load gear	
<TO>.LoadGear.Numerator	Load gear counter
<TO>.LoadGear.Denominator	Load gear denominator
Leadscrew pitch	

<TO>.Mechanics.LeadScrew	Leadscrew pitch
Type of encoder mounting	
<TO>.Sensor[n].MountingMode	Type of encoder mounting
<TO>.Sensor[n].Parameter.Distance PerRevolution	Load distance per encoder revolution with an externally mounted encoder
Inversion	
<TO>.Actor.InverseDirection	Setpoint inversion
<TO>.Sensor[n].InverseDirection	Actual value inversion
Modulo	
<TO>.Modulo.Enable	Enable modulo
<TO>.Modulo.Length	Modulo length
<TO>.Modulo.StartValue	Modulo start value

Homing

Brief description

With homing, you create the relationship between the position in the technology object and the mechanical position. The position value in the technology object is assigned to a homing mark at the same time. This homing mark represents a known mechanical position.

With incremental actual values this process is called homing; with absolute actual values it is called absolute value adjustment.

Homing is a requirement for the indication of the correct position in the technology object, and for absolute positioning.

Homing is enabled with the Motion Control instruction "MC_Home", and is enabled for a single homing process each time.

Homing status

The technology object tag <TO>.StatusWord.HomingDone indicates whether the technology object has been homed to an axis or external encoder.

Type of homing

Homing can occur by means of an independent movement for homing (active homing), by means of a reference mark during an application-initiated movement (passive homing), or by means of direct position assignment.

A distinction is made between the following types of homing:

- **Active homing**
  Active homing initiates a homing movement and performs the necessary approach to the reference mark. When the reference mark is detected, the actual position is set to the configured value. It is possible to specify an offset from the home position.
  When active homing starts, current traversing movements are aborted. The offset is automatically traversed after the approach to home position.
- **Passive homing**
  The homing job does not perform its own homing motion. When the reference mark is detected during an application-initiated motion, the actual position is set to the configured value.
  Passive homing is also called homing on the fly.
- **Direct homing**
  The homing job directly sets the actual position to the configured value, or offsets it by this amount.
- **Absolute value adjustment**
  Absolute value adjustment adjusts the position of the technology object to the existing absolute actual value.

Homing mode

Depending on the type of reference mark and of the reference mark search, a distinction is made among the following homing modes (Page 10225):

- **Homing with zero mark via PROFIdrive frame and proximity switch**
- **Homing with zero mark via PROFIdrive frame**
- **Homing with digital input**

Terms

Homing mark

A homing mark is an input signal, on whose occurrence a known mechanical position can be assigned to the actual values.

A homing mark can be:

- **A zero mark**
  The zero mark of an incremental encoder or an external zero mark is used as a homing mark.
  The zero mark is detected at the drive module or encoder module, and transmitted in the PROFIdrive frame. Perform the setting and evaluation as an encoder zero mark or external zero mark at the drive module and sensor module.
- **An edge at the digital input**
  The falling or rising edge at a digital input is used as a homing mark.

Proximity switch

If there are several zero marks in the traversing range, the proximity switch permits the selection of a specific zero mark before or after the proximity switch.

Homing mark position

This is the position assigned to the homing mark.

The homing mark position corresponds to the home position minus the home position offset.

Home position

At the end of the active homing motion, the axis arrives at the home position.

Home position offset

The difference between the homing mark position and the home position is the home position offset.

An offset between homing mark position and home position only has an effect during active homing. The offset is traversed after the synchronization of the axis via the Motion Control instruction "MC_Home". For axes with the modulo setting, the home position offset is always traversed using the direction setting for the shortest path.

Direction reversal at the hardware limit switch (reversing cam)

Hardware limit switches can be used as reversing cams in active homing. If the homing mark is not detected or was approached from the wrong side, then the motion continues after the reversing cam in the opposite direction.

Homing mode

Various homing modes are available for the positioning axis technology object and the external encoder technology object. The homing mode is set in the configuration.

Homing with zero mark via PROFIdrive message frame and proximity switch

The system checks for when the proximity switch is reached. After the proximity switch is reached and is left again in the assigned homing direction, zero mark detection is enabled via the PROFIdrive message frame.

When the zero mark is reached in the pre-selected direction, then the actual position of the technology object is set to the reference mark position.

Homing with zero mark via PROFIdrive message frame

The system enables zero mark detection, as soon as the actual value of the technology object moves in the assigned homing direction.

When the zero mark is reached in the specified homing direction, the actual position of the technology object is set to the reference mark position.

Homing with digital input

The system checks the state of the digital input, as soon as the actual value of the axis or encoder moves in the assigned homing direction.

When the reference mark is reached (setting of the digital input) in the specified homing direction, the actual position of the technology object is set to the reference mark position.

Note

The digital inputs must be placed into the process image partition "PIP OB Servo".

The filter time of the digital inputs must be set smaller than the duration of the input signal at the reference point switch.

See also

Active homing for SINAMICS drives with external zero mark (Page 10415)

Active homing with zero mark and proximity switch

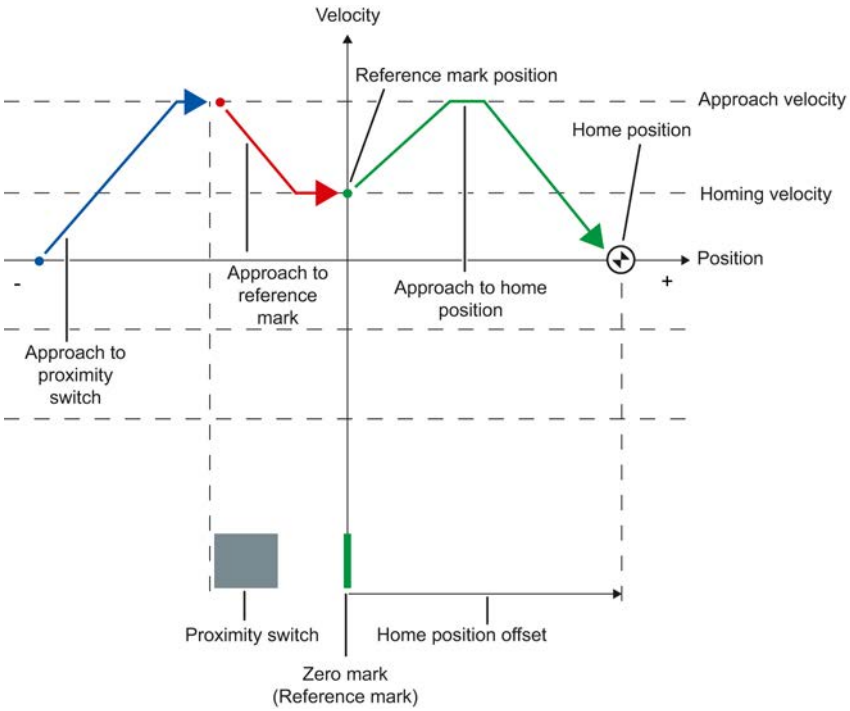
The following examples show homing motions in the positive and negative directions.

Example of homing in the positive direction

The approach to the reference mark and the home position occurs in the positive direction.

The following figure shows the homing motion with the following settings:

- Active homing with zero mark and proximity switch
- Approach in the positive direction
- Homing in the positive direction
- Positive home position offset

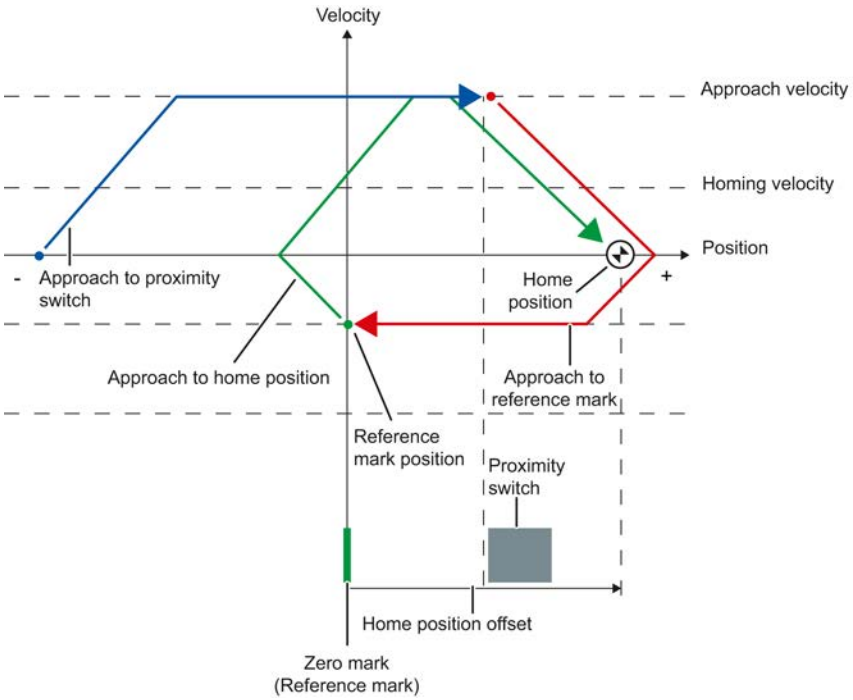


Example of homing in the negative direction

The move to the reference mark occurs in the negative direction by means of a direction reversal during the homing process. The move to the home position causes another direction reversal and occurs in the positive direction.

The following figure shows the homing motion with the following settings:

- Active homing with zero mark and proximity switch
- Approach in the positive direction
- Homing in the negative direction
- Positive home position offset



Motion sequence

The motion occurs in the following sequence:

1. **Start of active homing via the Motion Control instruction "MC_Home"**
2. **Approach to the proximity switch**
3. **Detection of the proximity switch in the homing direction, while moving at homing velocity**
4. **Departure from the proximity switch, and approach to the reference mark**
   With the departure from the proximity switch, the detection of the reference mark is enabled.

5. **Detection of the reference mark**
   When the reference mark is detected, the position of the technology object is set depending on the configured mode.
   - Parameter "Mode" to "MC_Home" = 4
     Position = value in parameter "Position" minus
     <TO>.Sensor[n].ActiveHoming.HomePositionOffset
   - Parameter "Mode" to "MC_Home" = 5
     Position = value in the tags <TO>.Homing.HomePosition minus
     <TO>.Sensor[n].ActiveHoming.HomePositionOffset
6. **Approach to the home position**
   - Parameter "Mode" to "MC_Home" = 4
     The axis moves to the position that is specified in the "Position" parameter.
   - Parameter "Mode" to "MC_Home" = 5
     The axis moves to the position that is specified in the <TO>.Homing.HomePosition tag.

   Note

   If the velocity on the span from the detection of the proximity switch to the zero mark cannot be reduced to the homing velocity, then homing occurs at the velocity that exists when the zero mark is traversed.

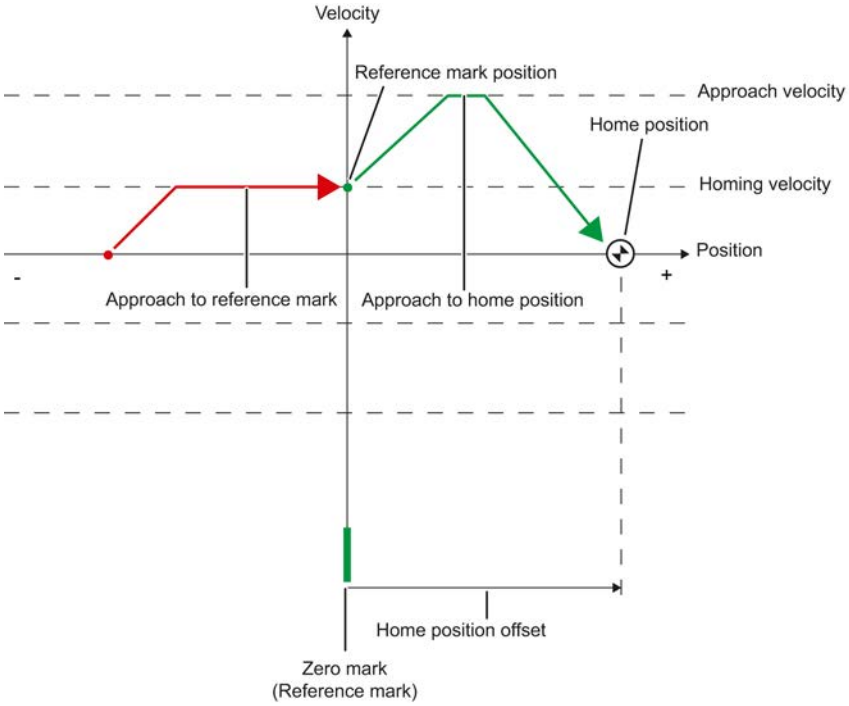
See also

Active homing for SINAMICS drives with external zero mark (Page 10415)

Active homing with zero mark

The following figure shows an example of the homing motion with the following settings:

- Active homing with zero mark
- Homing in the positive direction
- Positive home position offset



Motion sequence

The motion occurs in the following sequence:

1. **Start of active homing via the Motion Control instruction "MC_Home".**
2. **Move to the reference mark in the homing direction with the homing velocity**

3. **Detection of the reference mark**
   When the reference mark is detected, the position of the axis or encoder is set depending on the configured mode.
   - Parameter "Mode" to "MC_Home" = 4
     Position = value in parameter "Position" minus
     <TO>.Sensor[n].ActiveHoming.HomePositionOffset
   - Parameter "Mode" to "MC_Home" = 5
     Position = value in the tag <TO>.Homing.HomePosition minus
     <TO>.Sensor[n].ActiveHoming.HomePositionOffset
4. **Approach to the home position**
   - Parameter "Mode" to "MC_Home" = 4
     The axis moves to the position that is specified in the "Position" parameter.
   - Parameter "Mode" to "MC_Home" = 5
     The axis moves to the position that is specified in the <TO>.Homing.HomePosition tag.

See also

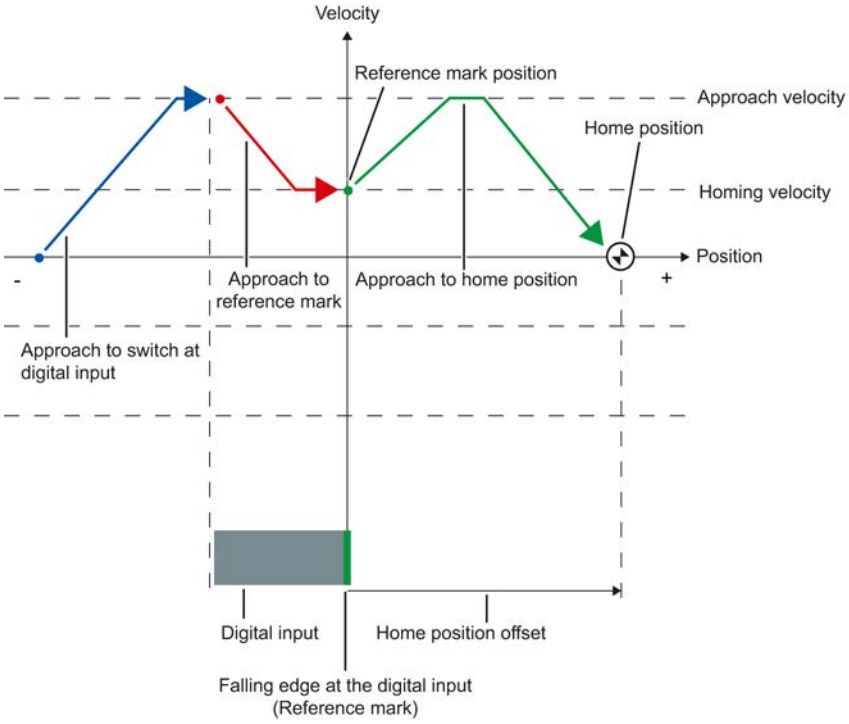
Active homing for SINAMICS drives with external zero mark (Page 10415)

Active homing with digital input

The following figure shows an example of the homing motion with the following settings:

- Active homing with digital input
- Approach in the positive direction

- Reference mark on the positive side of the digital input
- Positive home position offset



Motion sequence

The motion occurs in the following sequence:

1. **Start of active homing via the Motion Control instruction "MC_Home"**
2. **Detection of the rising edge at the digital input, while moving at homing velocity**
3. **Approach to the reference mark**

4. **Detection of the reference mark**
   In the example, the falling edge of the switch at the digital input represents the reference mark.
   When the reference mark is detected, the position of the axis or encoder is set depending on the configured mode.
   - Parameter "Mode" to "MC_Home" = 4
     Position = value in parameter "Position" minus
     <TO>.Sensor[n].ActiveHoming.HomePositionOffset
   - Parameter "Mode" to "MC_Home" = 5
     Position = value in the tag <TO>.Homing.HomePosition minus
     <TO>.Sensor[n].ActiveHoming.HomePositionOffset
5. **Approach to the home position**
   - Parameter "Mode" to "MC_Home" = 4
     The axis moves to the position that is specified in the "Position" parameter.
   - Parameter "Mode" to "MC_Home" = 5
     The axis moves to the position that is specified in the <TO>.Homing.HomePositiontag.

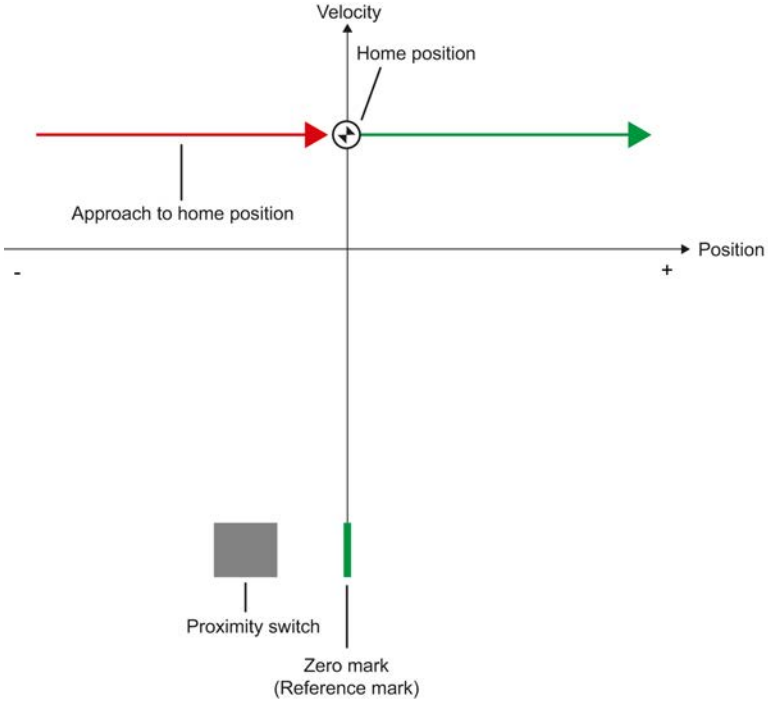
Note

If the velocity on the span from the detection of the rising edge to the falling edge cannot be reduced to the homing velocity, then homing occurs at the velocity that exists when the reference mark is traversed.

Passive homing with zero mark and proximity switch

The following figure shows an example of the homing motion with the following settings:

- Passive homing with zero mark and proximity switch
- Homing in the positive direction



Motion sequence

The motion occurs in the following sequence:

1. **Enablement of passive homing via the Motion Control instruction "MC_Home".**
2. **Motion due to a Motion Control job from the application**
   The detection of the proximity switch and of the reference mark is enabled when the actual position value of the axis or encoder moves in the assigned homing direction.
3. **Detection of the proximity switch**

4. **Departure from the proximity switch**
   The departure from the proximity switch enables the detection of the reference mark.
5. **Detection of the reference mark**
   When the reference mark is detected, the position of the axis or encoder is set depending on the configured mode.
   – Parameter "Mode" to "MC_Home" = 2
     Position = value in parameter "Position"
   – Parameter "Mode" to "MC_Home" = 3
     Position = value in tag <TO>.Homing.HomePosition

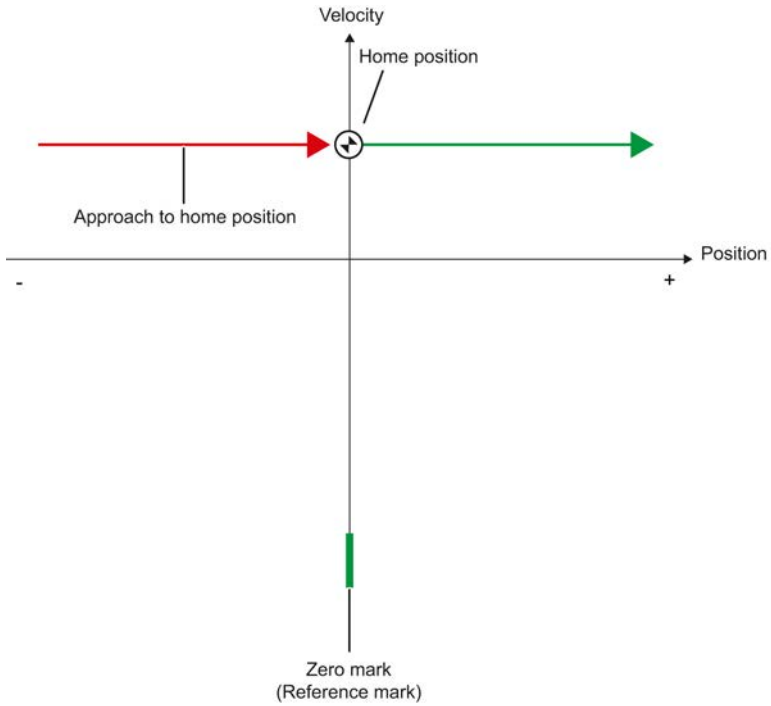
Note

If the direction of motion changes after departure from the proximity switch and before detection of the reference mark, then the proximity switch must be detected again. The Motion Control instruction "MC_Home" remains enabled.

Passive homing with zero mark

The following figure shows an example of the homing motion with the following settings:

- Passive homing with zero mark
- Homing in the positive direction



Motion sequence

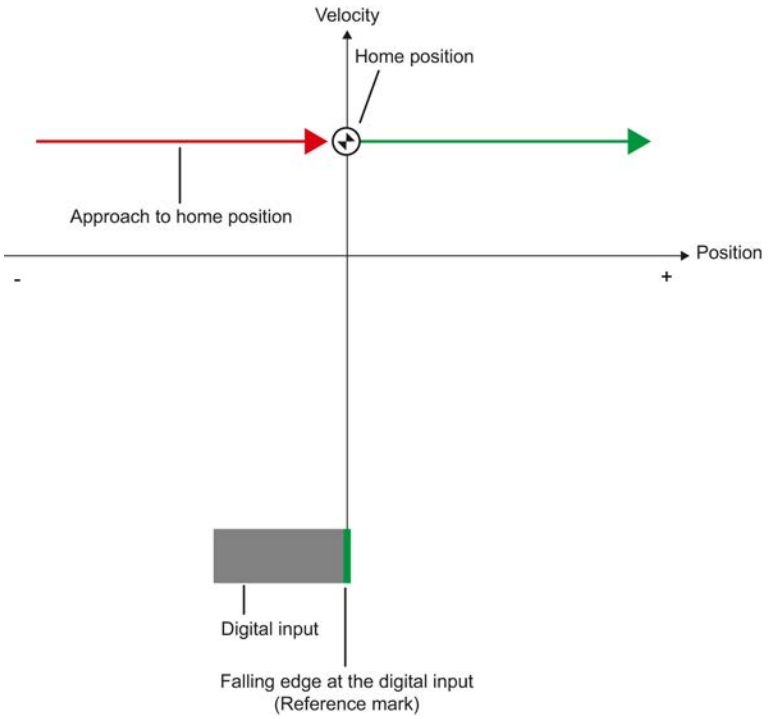
The motion occurs in the following sequence:

1. **Enablement of passive homing via the Motion Control instruction "MC_Home".**
2. **Motion due to a Motion Control job from the application**
   The detection of the reference mark is enabled when the actual position value of the axis or encoder moves in the assigned homing direction.
3. **Detection of the reference mark**
   When the reference mark is detected, the position of the axis or encoder is set depending on the configured mode.
   – Parameter "Mode" to "MC_Home" = 2
     Position = value in parameter "Position"
   – Parameter "Mode" to "MC_Home" = 3
     Position = value in tag <TO>.Homing.HomePosition

Passive homing with digital input

The following figure shows an example of the homing motion with the following settings:

- Passive homing with digital input
- Homing in the positive direction
- Reference mark on the positive side of the digital input



Motion sequence

The motion occurs in the following sequence:

1. **Enablement of passive homing via the Motion Control instruction "MC_Home".**
2. **Motion due to a Motion Control job from the application**
   The detection of the reference mark at the digital input is enabled when the actual position value of the axis or encoder moves in the assigned homing direction.
3. **Detection of the reference mark**
   In the example, the falling edge of the switch at the digital input represents the reference mark.
   When the reference mark is detected, the position of the axis or encoder is set depending on the configured mode.
   - Parameter "Mode" to "MC_Home" = 2
     Position = value in parameter "Position"
   - Parameter "Mode" to "MC_Home" = 3
     Position = value in tag <TO>.Homing.HomePosition

Direction reversal at the hardware limit switch (reversing cam)

During active homing, the hardware limit switch can optionally be used as a reversing cam. If the reference mark is not detected or the motion was not in the homing direction, then the motion continues after the reversing cam in the opposite direction with the approach velocity.

When the hardware limit switch is reached, the dynamic defaults become effective. Deceleration with the emergency stop deceleration does not occur.

NOTICE
Avoid moving to a mechanical endstop Ensure by one of the following measures, that in a direction reversal the machine does not move to a mechanical endstop. • Keep the approach velocity low. • Increase the configured acceleration / deceleration. • Increase the offset between the hardware limit switch and the mechanical endstop.

Direct homing

Depending on the configured mode, the position of the positioning axis technology object or external encoder can be absolutely or relatively set with "MC_Home".

Set position absolutely

Proceed as follows to set the position absolutely:

1. In the Motion Control instruction "MC_Home", enter the absolute position in the "Position" parameter.
2. Call the Motion Control instruction "MC_Home" with parameter "Mode" = 0.

The position is set to the value specified in the "Position" parameter.

Set position relatively

Proceed as follows to set the position relatively:

1. In the Motion Control instruction "MC_Home", enter the relative position in the "Position" parameter.
2. Call the Motion Control instruction "MC_Home" with parameter "Mode" = 1.

The position is set to the current position plus the value specified in the "Position" parameter.

Absolute value adjustment

In absolute value adjustment, Motion Control determines an absolute value offset, that is retentively stored on the CPU.

Depending on the configured mode, the position of the axis or the encoder is absolutely or relatively set in the "MC_Home" Motion Control instruction.

- Parameter "Mode" = 7 (absolute specification of position)
  Position = value in parameter "Position"
- Parameter "Mode" = 6 (relative specification of position)
  Position = current position + value in parameter "Position"

Resetting the "Homed" status

Incremental encoder

In the following cases, the "Homed" status is reset, and the technology object must be rehomed.

- Errors in the sensor system / encoder failure
- Initiation of active homing with the Motion Control instruction "MC_Home" with "Mode" = 4, 5 (after successful completion of the homing process, the status "Homed" is set again.)
- Initiation of passive homing with the Motion Control instruction "MC_Home" with "Mode" = 2, 3 (after successful completion of the homing process, the status "Homed" is set again.)
- Replacement of the CPU
- Replacement of the SIMATIC Memory Card
- POWER OFF
- Memory reset
- Modification of the encoder configuration
- Restart of the technology object

- Restoration of the CPU factory settings
- Transfer of a different project into the controller

Absolute value encoder

In the following cases, the "Homed" status is reset, and the technology object must be rehomed.

- Errors in the sensor system / encoder failure
- Replacement of the CPU
- Modification of the encoder configuration
- Restoration of the CPU factory settings
- Transfer of a different project into the controller

Resetting the memory of the CPU or upgrading a project does not require another absolute value adjustment.

Tags

The following technology object tags are relevant for homing:

Status indicators	
<TO>StatusWord.HomingCommand	Homing command active
<TO>StatusWord.HomingDone	Technology object is homed
<TO>ErrorWord.HomingFault	Error occurred during homing
Approach to the proximity switch	
<TO>.Homing.ApproachDirection	Start direction or approach direction for the approach to the proximity switch
<TO>.Homing.ApproachVelocity	Velocity for the approach to the proximity switch
Approach to the reference mark	
<TO>.Sensor[n].ActiveHoming.Direction	Homing direction
<TO>.Homing.ReferencingVelocity	Velocity for the approach to the reference mark
Approach to home position	
<TO>.Homing.ApproachVelocity	Velocity for the approach to the home position
Positions	
<TO>.Homing.AutoReversal	Reversal at the hardware limit switches
<TO>.Homing.HomePosition	Home position
<TO>.StatusSensor[n].AbsEncoderOffset	Calculated offset after the absolute value adjustment
Parameters for active homing	
<TO>.Sensor[n].ActiveHoming.Mode	Homing mode
<TO>.Sensor[n].ActiveHoming.SideInput	Side of the digital input
<TO>.Sensor[n].ActiveHoming.Direction	Homing direction or approach direction
<TO>.Sensor[n].ActiveHoming.DigitalInputAddress	Byte number of the I/O address of the digital input

<TO>.Sensor[n].ActiveHoming.DigitalInputBitNumber	Bit number of the I/O address of the digital input
<TO>.Sensor[n].ActiveHoming.HomePositionOffset	Offset of the reference mark from the home position
Parameters for passive homing	
<TO>.Sensor[n].PassiveHoming.Mode	Homing mode
<TO>.Sensor[n].PassiveHoming.SideInput	Side of the digital input
<TO>.Sensor[n].PassiveHoming.Direction	Homing direction or approach direction
<TO>.Sensor[n].PassiveHoming.DigitalInputAddress	Byte number of the I/O address of the digital input
<TO>.Sensor[n].PassiveHoming.DigitalInputBitNumber	Bit number of the I/O address of the digital input

Note

Evaluation of the bits in StatusWord, ErrorWord and WarningWord

Read the information provided in section Evaluate StatusWord, ErrorWord and WarningWord (Page 10317).

Control

Brief description

The position controller of the positioning axis is a closed-loop P controller with pre-control of velocity.

If the drive supports Dynamic Servo Control (DSC), then a closed-loop position controller in the drive can optionally be used.

Dynamic Servo Control (DSC)

In drives that support Dynamic Servo Control (DSC), you can optionally use the closed-loop position controller in the drive. The position controller in the drive is usually implemented with a rapid speed-control cycle. This improves the control performance for digitally coupled drives. The communication times between controller and drive are automatically taken into account.

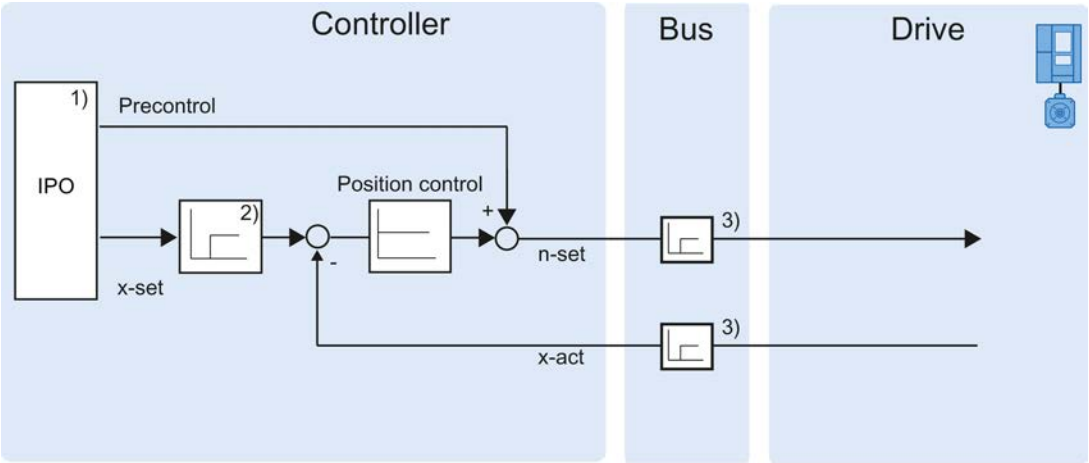
For DSC, the drive's encoder must be used and the drive must support standard frame 5.

See also

Frames (Page 10217)

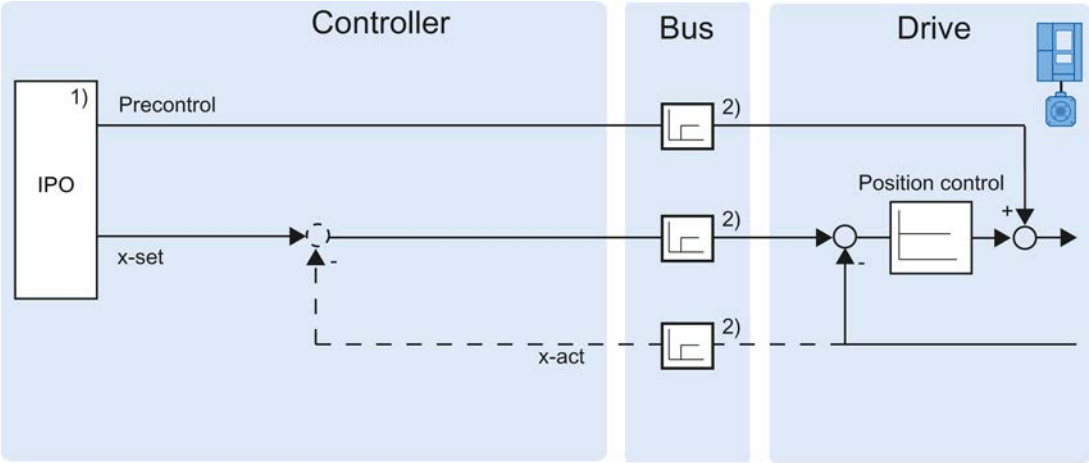
Control structure

The following figure shows the effective control structure without DSC:



1) Interpolator with Motion Control
2) Internal consideration of signal delays
3) Controller-drive communication

The following figure shows the effective control structure with DSC:



1) Interpolator with Motion Control
2) Controller-drive communication

Tags

The following technology object tags are relevant for control:

Parameter	
<TO>.PositionControl.Kv	Proportional gain in the position control

<TO>.PositionControl.Kpc	Velocity pre-control in the position control
<TO>.PositionControl.EnableDSC	Enabling DSC

Position-related monitoring

Brief description

The following functions are available in the positioning axis technology object for monitoring positioning and motion:

- Positioning monitoring (Page 10242)
  The actual position value must reach a positioning window within a specified time, and remain in this positioning window for a minimum dwell time.
- Following error monitoring (Page 10243)
  The following error is monitored based on a velocity-dependent following error limit. The permissible maximum following error depends on the setpoint velocity.

If monitored conditions are violated, then technology alarms are output. The technology object responds in accordance with the alarm response.

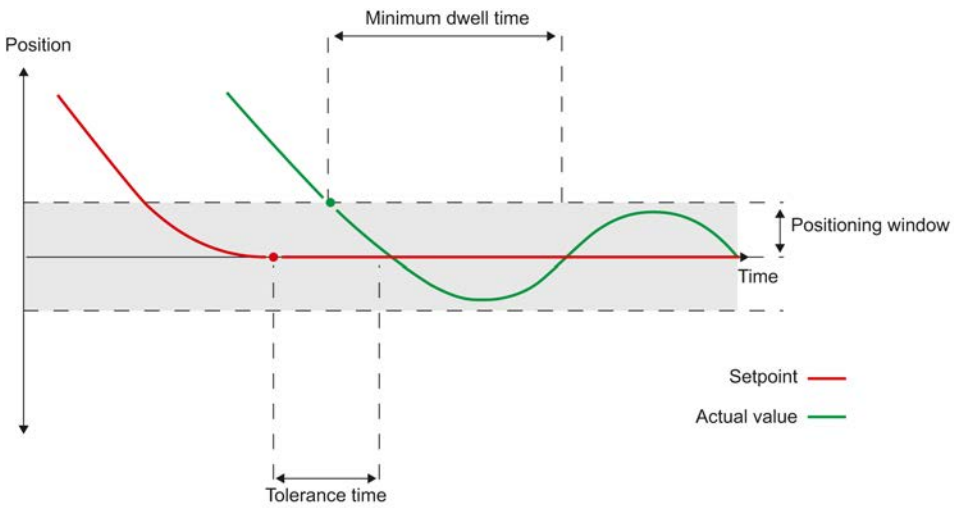
Positioning monitoring

Positioning monitoring monitors the behavior of the actual position at the end of the setpoint calculation.

As soon as the setpoint velocity reaches the value zero, the actual position value must be located within a tolerance time in the positioning window. The setpoint must not exit the positioning window during the minimum dwell time.

If the actual position value at the end of a positioning motion is reached within the tolerance time and remains in the positioning window for the minimum dwell time, then <TO>.StatusWord.Done is set in the technology data block. This completes a Motion Control job.

The following figure shows the chronological sequence and the positioning window:



Positioning monitoring does not make any distinction between how the setpoint interpolation was completed. The end of setpoint interpolation can for example be reached as follows:

- by the setpoint reaching the target position
- by position controlled stopping during the motion, via the Motion Control instruction "MC_Halt"

Violation of positioning monitoring

In the following cases, technology alarm 541 is output by the positioning monitoring, and the technology object is disabled (alarm response: Remove enable).

- The actual value does not reach the positioning window during the tolerance time.
- The actual value exits the positioning window during the minimum dwell time.

Following error monitoring

The following error in the positioning axis technology object is monitored based on a velocity-dependent following error limit. The permissible following error depends on the setpoint velocity.

A constant permissible following error can be specified for velocities less than an adjustable velocity low limit.

Above this lower velocity limit, the permissible following error increases in proportion to the setpoint velocity. The maximum permissible following error may be reached at the maximum velocity.

Calculation of the following error

The following error is the difference between the position setpoint and the actual position value. The transmission times of the setpoint to the drive, and of the actual position value to the controller, are taken into account in the calculation of the following error, i.e., subtracted out.

Warning limit

A warning limit can be specified for the following error. The warning limit is input as a percentage value and operates relative to the current permissible following error. If the warning limit of the following error is reached, then technology alarm 522 is output. This is a warning and contains no alarm response.

Exceedance of the permissible following error

If the permissible following error is exceeded, then technology alarm 521 is output, and the technology object is disabled (alarm response: Remove enable).

Tags

The following technology object tags are relevant for positioning monitoring:

Status indicators	
<TO>.StatusWord.Standstill	Is set to the value TRUE , when the actual velocity value goes below the velocity threshold, and does not exit it within the minimum dwell time. The standstill signal is only present at the positioning axis.
<TO>.StatusWord.Done	**Positioning axis** Is set to the value TRUE, when the actual velocity value reaches the positioning window within the tolerance time, and remains for the minimum dwell time in the window. **Speed axis** Is set to TRUE, when the motion is completed and thus the speed setpoint is equal to zero.
<TO>.ErrorWord.PositioningFault	A positioning error has occurred.
Positions and times	
<TO>.PositioningMonitoring.ToleranceTime	Maximum permissible time until positioning window is reached The time is started with the end of the setpoint interpolation.
<TO>.PositioningMonitoring.MinDwellTime	Minimum dwell time in the positioning window
<TO>.PositioningMonitoring.Window	Positioning window
Standstill signal	
<TO>.StandstillSignal.VelocityThreshold	Velocity threshold for the standstill signal
<TO>.StandstillSignal.MinDwellTime	Minimum dwell time below the velocity threshold

The following technology object tags are relevant for following error monitoring:

Status indicators	
<TO>.StatusPositioning.FollowingError	Current following error
<TO>.ErrorWord.FollowingErrorFault	Status indication, that the following error is too large
<TO>.WarningWord.FollowingErrorWarning	Status indication, that the following error warning limit has been reached
Control bits	
<TO>.FollowingError.EnableMonitoring	Enabling / disabling following error monitoring
Limit values	
<TO>.FollowingError.MinVelocity	Lower setpoint velocity for the characteristic curve of the maximum following error
<TO>.FollowingError.MinValue	Permissible following error below the <TO>.FollowingError.MinVelocity

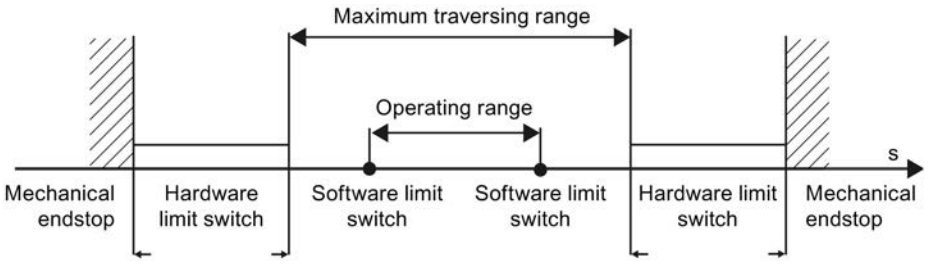
<TO>.FollowingError.MaxValue	Maximum permissible following error at maximum axis velocity
<TO>.FollowingError.WarningLevel	Warning limit as a percentage value relative to the maximum permissible following error (velocity-dependent in accordance with the characteristic curve)

Traversing range limitation

Brief description

Hardware and software limit switches limit the permissible traversing range and operating range of the positioning axis. Before use, they must be enabled in the configuration or in the user program.

The following figure shows the relationship between operating range, maximum traversing range and the limit switches:



Hardware limit switches

Hardware limit switches are limit position switches that limit the maximum permissible traversing range of the axis.

Select the positions of the hardware limit switches so that there is adequate braking distance for the axis when needed. The axis should come to a standstill before a mechanical endstop.

Approaching the hardware limit switches

In the monitoring of range limitation, no distinction is made, whether the switches are approached or overshot.

If a hardware limit switch is approached, technology alarm 531 is output, and the technology object is disabled (alarm response: remove enable).

Exception

If the hardware limit switches are used as reversing cams or reference cams, then the monitoring of the hardware limit switches has no effect.

Freeing the axis

The position of the axis when the hardware limit switch is detected is stored internally on the CPU. The status of the approached hardware limit switch is reset only after the hardware limit switch is left and the axis is once again in the maximum traversing range.

To be able to traverse the axis again after reaching the hardware limit switch and to reset the status of the hardware limit switch, follow the steps below:

1. To allow movement away from the limit switch, acknowledge the technology alarm.
2. Traverse the axis away from the hardware limit switch until it leaves it.
   The axis must then be within the maximum traversing range.
   If you traverse the axis in the direction of the limit switch before the axis has left it, the monitoring will be triggered again.

The following diagram shows the response of the status word when reaching the hardware limit switch and when releasing the axis again:

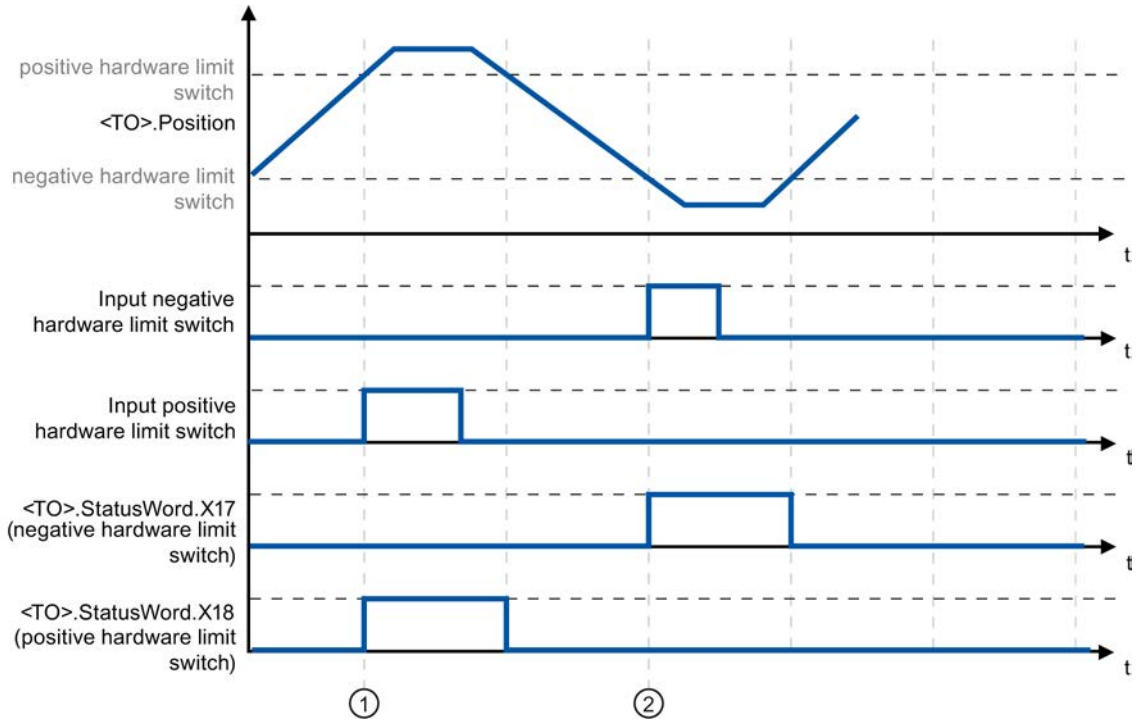


Figure 11-10 Status message HW-ES

At the time ① the position of the axis when the positive hardware limit switch is detected is stored internally on the CPU. To reset the status of the hardware limit switch, the axis must be pulled back from this position.

At the time ② the position of the axis when the negative hardware limit switch is detected is stored internally on the CPU. To reset the status of the hardware limit switch, the axis must pass this position.

Software limit switch

The operating range of the axis is limited with software limit switches. Relative to the traversing range, always position the software limit switches within the hardware limit switches. Since the positions of the software limit switches can be flexibly configured, the operating range of the axis can be individually adapted in accordance with the current velocity profile.

Software limit switches are only effective when there is a valid actual value after homing the technology object. The monitoring of the software limit switches is relative to the setpoint.

Modulo enabled

With modulo enabled, the modulo position is monitored.

The software limit switches are enabled or disabled using the tags in the technology data block. If the positions of both software limit switches are outside the modulo range, then the monitoring has no effect. It is not checked, whether the positions of the software limit switches are within the modulo range.

Approaching the software limit switches

At the beginning of a positioning motion, it is not checked, whether the software limit switches will be approached by approaching the specified target position.

If the software limit switches are approached, then technology alarm 533 is output, and the axis is stopped with the maximum dynamic values (alarm response: Stop with maximum dynamic values). The technology object remains enabled.

Overshooting the software limit switches

If a software limit switch is overshot, technology alarm 534 is output, and the technology object is disabled (alarm response: remove enable).

Retracting

Proceed as follows to retract the axis after violation of the software limit switch:

1. Acknowledge the technology alarm.
2. Move the axis in the free travel direction, until the software limit switch is exited.
   If you move opposite to the free travel direction before exiting the software limit switch, then the monitor with be triggered again.

Tags

The following technology object tags are relevant for software limit switches:

Status indicators	
<TO>.StatusWord.SWLimitMinActive	Negative software limit switch is enabled
<TO>.StatusWord.SWLimitMaxActive	Positive software limit switch is enabled
<TO>.ErrorWord.SWLimit	An alarm is pending, that a software limit switch was violated

Control bits	
<TO>.PositonLimits_SW.Active	Enables / disables the monitoring of the software limit switches
Position values	
<TO>.PositonLimits_SW.MinPosition	Position of the negative software limit switch
<TO>.PositonLimits_SW.MaxPosition	Position of the positive software limit switch

The following technology object tags are relevant for hardware limit switches:

Status indicators	
<TO>.StatusWord.HWLimitMinActive	Negative hardware limit switch is enabled
<TO>.StatusWord.HWLimitMaxActive	Positive hardware limit switch is enabled
<TO>.ErrorWord.HWLimit	An alarm is queued; a hardware limit switch was violated
Control bits	
<TO>.PositonLimits_HW.Active	Enables / disables the monitoring of the hardware limit switches
Parameter	
<TO>.PositonLimits_HW.MinSwitchLevel	Level selection for enablement of the lower hardware limit switch: FALSE: At low level, the signal is enabled TRUE: At high level, the signal is enabled
<TO>.PositonLimits_HW.MinSwitchAddress	Byte number of the I/O address of the hardware limit switch for the lower or minimum position
<TO>.PositonLimits_HW.MinSwitchBitNumber	Bit number of the I/O address of the hardware limit switch for the lower or minimum position
<TO>.PositonLimits_HW.MaxSwitchLevel	Level selection for enablement of the upper hardware limit switch: FALSE: At low level, the signal is enabled TRUE: At high level, the signal is enabled
<TO>.PositonLimits_HW.MaxSwitchAddress	Byte number of the I/O address of the hardware limit switch for the upper or maximum position
<TO>.PositonLimits_HW.MaxSwitchBitNumber	Bit number of the I/O address of the hardware limit switch for the upper or maximum position

Motion control and limits for dynamics

Brief description

Motion control of the axis occurs by means of velocity profiles (Page 10249). The velocity profiles are calculated in accordance with the specifications for dynamics. A velocity profile defines the behavior of the axis during approach, braking and changes in velocity. During positioning a velocity profile is calculated, that moves the axis to the target point.

Maximum values for velocity, acceleration and jerk result from the properties of the drive and the mechanics. These maximum values can be configured in the limits for dynamics. The limits for dynamics are in effect as limits for every motion generated by means of the technology object.

The configurable emergency stop deceleration (Page 10250) is triggered by the Motion Control instruction MC_Power or by a technology alarm.

The jerk limit reduces the mechanical load during an acceleration ramp or deceleration ramp. A "smoothed" velocity profile results.

Velocity profile

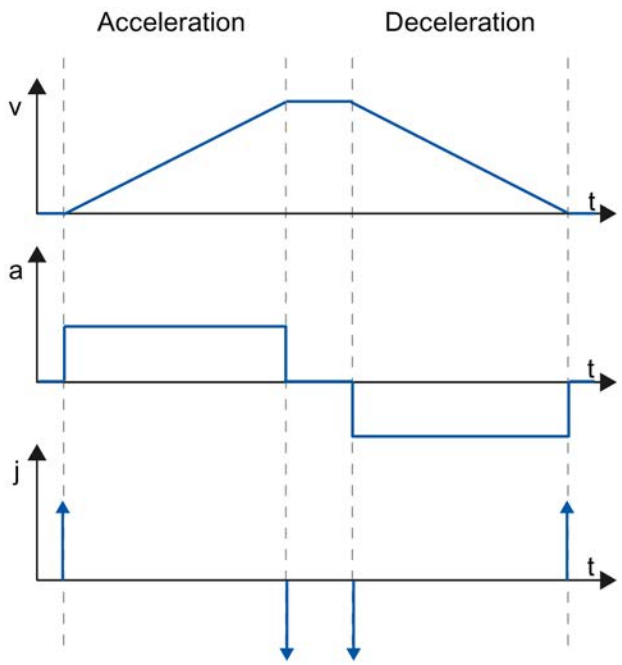
A velocity profile without jerk limit and a velocity profile with jerk limit are both supported for Motion Control of the axis.

The dynamic values for the motion are specified in the Motion Control job. Alternatively the dynamic default values can be used. The defaults and the limits for velocity, acceleration, deceleration and jerk are set in the configuration.

To influence velocity, a velocity override can override the current traversing velocity.

Velocity profile without jerk limit

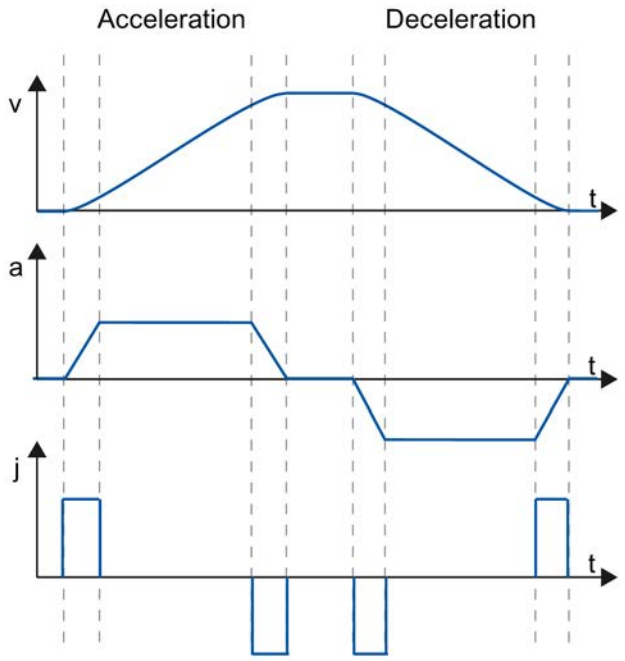
The following figure shows velocity, acceleration and jerk:



v - velocity
a - acceleration
j - jerk
t - time

Velocity profile with jerk limit

The following figure shows velocity, acceleration and jerk:



v - velocity
a - acceleration
j - jerk
t - time

A velocity profile with jerk limit is employed for a continuous acceleration and deceleration sequence. The jerk can be specified.

Emergency stop deceleration

When stopping with the emergency stop ramp, the axis is braked to a standstill without a jerk limit, using the configured emergency stop deceleration.

In the following cases the configured emergency stop deceleration is in effect:

- For an emergency stop ramp that has been enabled via the Motion Control instruction "MC_Power" with parameter "StopMode" = 0.
- For a technology alarm with the local alarm response "Stop with emergency stop ramp".

This emergency stop deceleration can be set greater than the maximum deceleration. If the emergency stop deceleration is set lower than this, it may occur that the axis does not stop until after the limit switch in the case of "Stop at software limit switch" and the occurrence of a technology alarm with the local alarm response "Stop with emergency stop ramp".

Tags

The following technology object tags are relevant for motion control:

Status	
<TO>.StatusWord	Status indicators for an active motion
<TO>.Position	Setpoint position
<TO>.Velocity	Setpoint velocity / setpoint speed
<TO>.ActualPosition	Actual position
<TO>.ActualVelocity	Actual velocity
<TO>.ActualSpeed	Actual speed of the motor (with analog setpoint = 0.0)
<TO>.Acceleration	Setpoint acceleration
<TO>.ActualAcceleration	Actual acceleration
Override	
<TO>.Override.Velocity	Velocity override
Dynamic limit values	
<TO>.DynamicLimits.MaxVelocity	Dynamic limit for maximum velocity
<TO>.DynamicLimits.MaxAcceleration	Dynamic limit for maximum acceleration
<TO>.DynamicLimits.MaxDeceleration	Dynamic limit for maximum deceleration
<TO>.DynamicLimits.MaxJerk	Dynamic limit for maximum jerk
Defaults for the dynamics	
<TO>.DynamicDefaults.Velocity	Default velocity
<TO>.DynamicDefaults.Acceleration	Default acceleration
<TO>.DynamicDefaults. Deceleration	Default deceleration
<TO>.DynamicDefaults.Jerk	Default jerk
<TO>.DynamicDefaults. EmergencyDeceleration	Emergency stop deceleration

Operational sequence

Organization Blocks for Motion Control

Description

When you create a technology object, organization blocks are automatically created for processing the technology objects. The motion control functionality of the technology objects creates its own execution level, and is called according to the application cycle.

The following blocks are created:

- **MC-Servo [OB91]**
  Calculation of the Position Controller
- **MC-Interpolator [OB92]**
  Evaluation of the motion control instructions, generation of setpoints and monitoring functionality

The organization blocks are protected (know-how protection). The program code cannot be viewed or changed.

The frequency relationship of the two organization blocks to one another is always 1:1. MC-Servo [OB91] is always executed before MC-Interpolator [OB92].

You can set the application cycle and the priority of the organization blocks according to your requirements for control quality and system load.

Application cycle

You can set the application cycle in which the MC-Servo [OB91] is called in the properties of the organization block in "General > Cycle Time":

- Synchronous to the bus
  MC-Servo [OB91] is called synchronously with or at a reduced ratio to a bus system. You set the send clock in the properties of the selected bus system. In the "Distributed I/O" drop-down list, you can select the following bus systems:
  - Isochronous PROFIBUS DP
  - Isochronous PROFINET IO

  You cannot call the MC-Servo [OB91] synchronously with a bus system that is connected to the CPU via a communications processor/communications module (CP/CM).
- Cyclic
  The MC-Servo [OB91] is called cyclically with the specified application cycle.

The selected application cycle must be long enough to be able to process the technology objects in one cycle. If the processing time of the technology objects is longer than the application cycle, overflows (Page 10253) will occur.

You can check the runtime of MC-Servo [OB91] and MC-Interpolator [OB92] with the expanded instruction "RT_INFO".

Clock reduction (CPU V1.5 and higher)

You can reduce the application cycle of the MC-Servo [OB91] relative to the send clock of an isochronous PROFINET IO system. You can set an integer multiple of the send clock as the factor. Values up to 14 times the send clock (maximum 32 ms) are possible for the application cycle.

If you call a isochronous interrupt OB and the MC-Servo [OB91] synchronously with the same PROFINET IO system, set the same application cycle for both organization blocks.

Priority

You can configure the priority of the organization blocks as needed in their properties under "General > Properties > Priority":

- MC Servo [OB91]
  Priority 17 to 31 (default value 25)
- MC Interpolator [OB92]
  Priority 16 to 30 (default value 24)

The priority of MC Servo [OB91] must be at least one higher than the priority of the MC Interpolator [OB92].

Process image partition "OB Servo PIP"

The process image partition "OB Servo PIP" is made available in isochronous mode for Motion Control. All drives and encoders used by Motion Control are assigned to this process image partition.

Additionally, you should assign all I/O modules used by Motion Control to this process image partition (e.g. hardware limit switches). The assignment results in chronologically synchronous processing with the technology object.

The input process image partition is also updated in STOP mode.

Process image partition in the user program

As of CPU version V1.5, you can access the process image partition "TPA OB Servo" in your user program using symbolic addressing. This makes it possible to evaluate the process image partition using the trace and logic analyzer function.

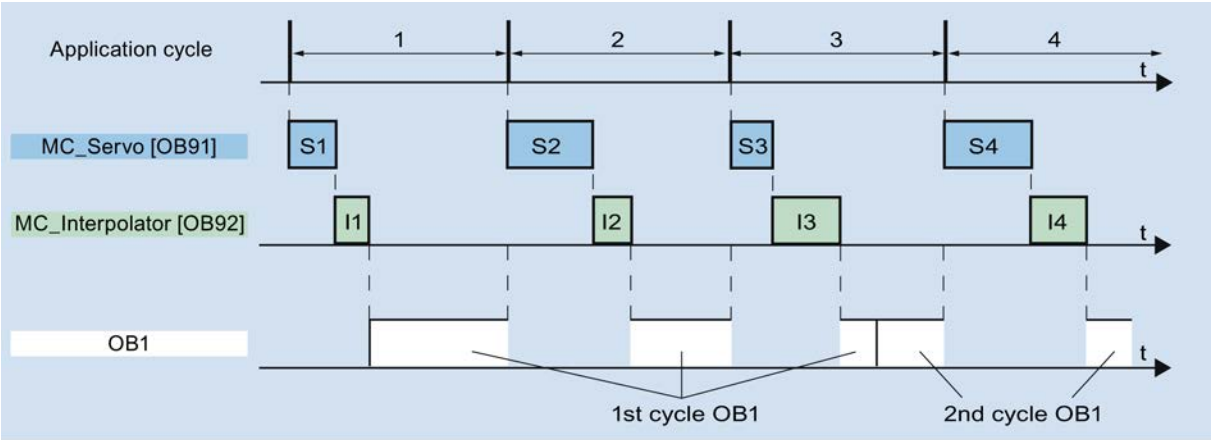
Operational Sequence and Timeouts

When processing the motion control functionality, the organization blocks MC-Servo [OB91] and MC-Interpolator [OB92] are called and processed in each application cycle. The remaining cycle time is available for the processing of your user program.

For error-free program execution, keep to the following rules:

- In each application cycle, MC-Servo [OB91] must be started and executed completely.
- In every application cycle, the relevant MC-Interpolator [OB92] must at least be started.

The following figure shows an example of the error-free operational sequence for the processing of organization block OB1:

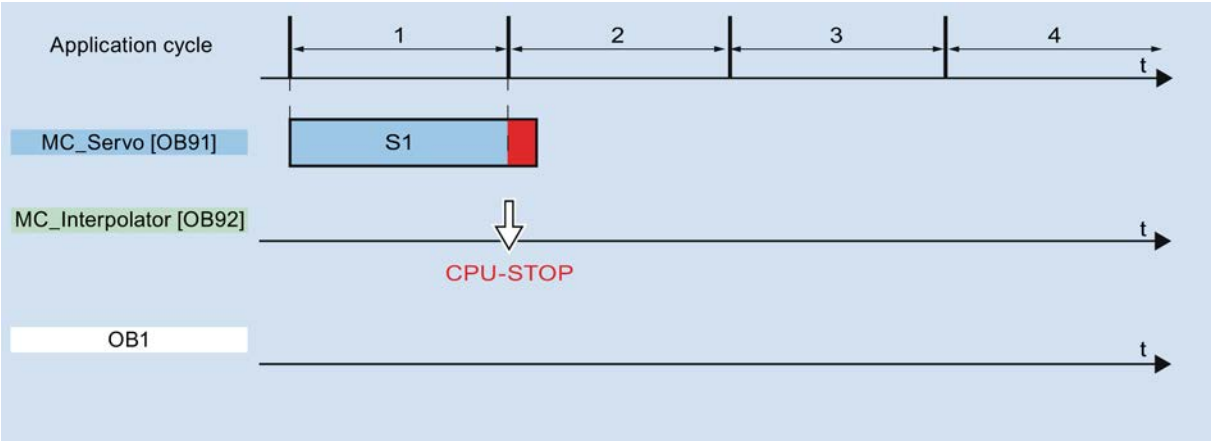


Overflows

If the set application cycle is not adhered to, for example because the application cycle is too short, overflows can occur.

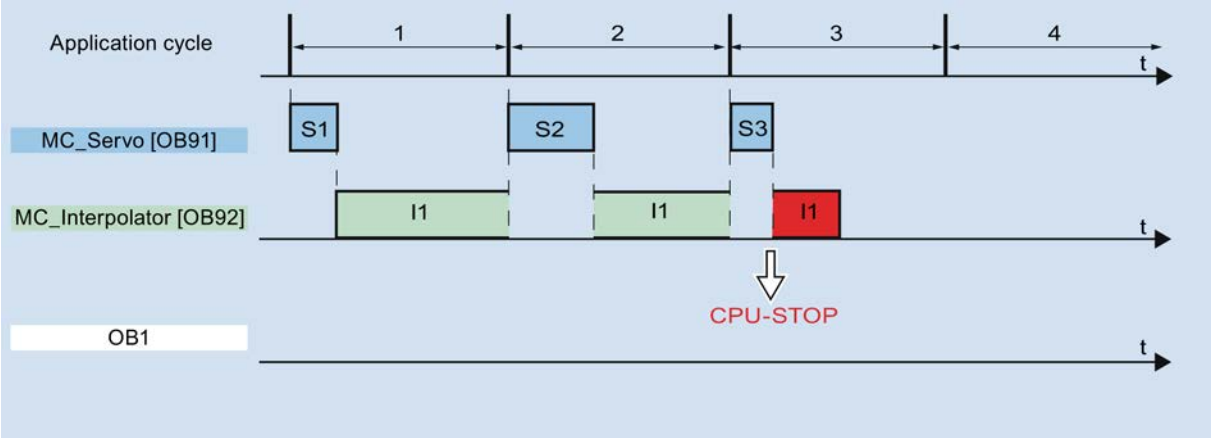
The CPU will not tolerate overflow of MC-Servo [OB91]. An overflow will cause the CPU to change to STOP mode.

The following figure shows the operational sequence if there is an overflow of MC-Servo [OB91]:



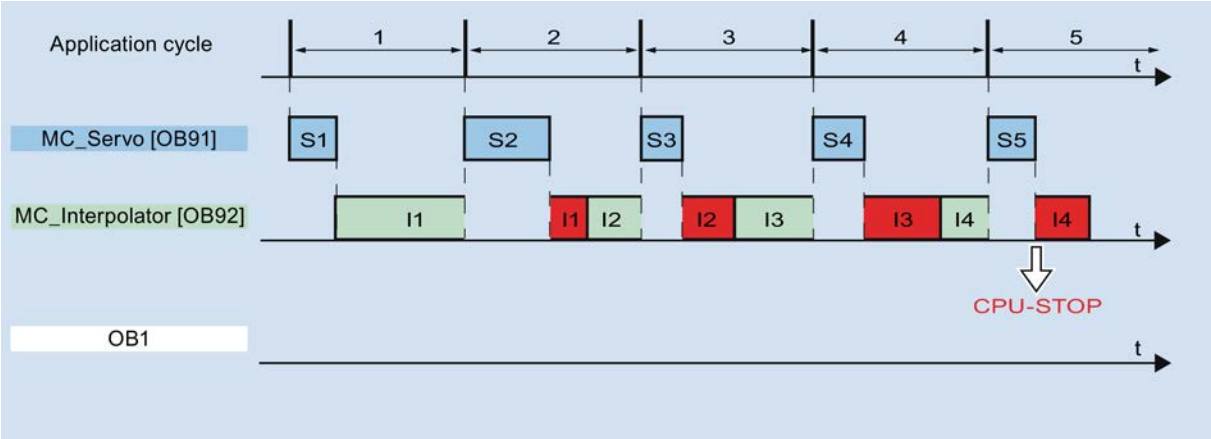
The execution of an MC-Interpolator [OB92] may only be interrupted by an MC-Servo [OB91] call. If more interruptions occur, the CPU switches to STOP mode.

The following figure shows the sequence when an MC-Interpolator [OB92] is interrupted over two time slices:



The CPU tolerates a maximum of three consecutive overflows of MC-Interpolator [OB92]. If more overflows occur, the CPU switches to STOP mode.

The following figure shows the sequence if there are four consecutive individual overflows of MC-Interpolator [OB92]:

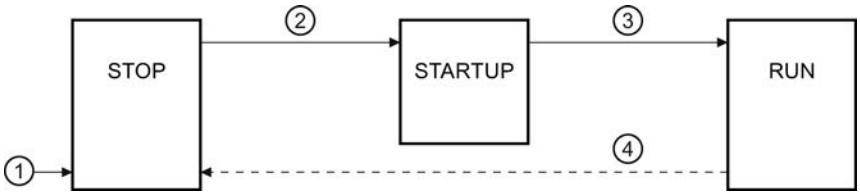


Operating modes

This section examines the behavior of Motion Control in each operating mode, and in the transitions between operating modes. A general description of the operating modes can be found in system manual S7-1500.

Operating modes and transitions

The CPU has three operating modes: STOP, STARTUP and RUN. The following figure shows the operating modes and the operating mode transitions:



STOP mode

In STOP mode the user program is not processed and al process outputs are disabled. Thus no Motion Control jobs are executed.

The technology data blocks are updated.

STARTUP mode

Before the CPU starts processing of the cyclical user program, the startup OBs are processed one time.

In STARTUP mode, the process outputs are disabled. Motion Control jobs are rejected.

The technology data blocks are updated.

RUN mode

The user program is processed in RUN mode.

In RUN mode, the programmed Motion Control jobs are cyclically called and processed.

The technology data blocks are updated.

Operating mode transitions

The following table shows the behavior of Motion Control in the transitions between the operating modes:

No.	Operating mode transition	Behavior
①	POWER ON → STOP	The CPU performs a restart of the technology objects. The technology objects are reinitialized with the values from the load memory.
②	STOP → STARTUP	Not relevant to Motion Control.
③	STARTUP → RUN	The process outputs are enabled.
④	RUN → STOP	When the CPU changes to RUN mode after STOP mode, all technology objects are disabled in accordance with the alarm response "remove enablement". Running Motion Control jobs are terminated.

Guidelines

Guidelines on use of motion control

The guidelines described here present the basic procedure for using Motion Control with the CPU S7-1500. These guidelines serve as recommendations.

Requirements

- A project with a CPU S7-1500 has been created.

Procedure

Proceed as follows to use Motion Control with the CPU S7-1500:

1. Add technology object (Page 10264)
2. Working with the configuration editor (Page 10265)
3. Programming (Page 10314)
4. Downloading to CPU (Page 10337)
5. Functional test in the Commissioning window (Page 10338)
6. Diagnostics (Page 10349)

Configuring

Adding and configuring drives in the device configuration

Add and configure PROFINET IO drives

Adding and configuring a PROFINET IO drive is described below using a **SINAMICS S120 drive**. Adding and configuring other PROFINET IO drives may differ from the description in certain respects.

Requirements

- The SIMATIC S7-1500 device is created in the project.
- The desired drive can be selected in the hardware catalog.

If the drive is not available in the Hardware Catalog, then it must be installed in the "Extras" menu as a device description file (GSD).

Adding a drive and frame in the device configuration

1. Open the device configuration and change to the network view.
2. In the hardware catalog, open the folder "Additional Field Devices > PROFINET IO > Drives > Siemens AG > SINAMICS".

3. Select the desired drive with the desired version, then drag it to the network view.
4. Assign the drive to the PROFINET interface of the PLC.
5. Open the drive in the device view.
6. Drag a Drive Object (DO) and a standard frame from the hardware catalog to the slot of the device overview of the drive.

Depending on the version of the SINAMICS S120 drive, select "DO with standard telegram X", or "DO Servo" and a "Standard telegram X" for the frame.

For more information on suitable frames, refer to the section Frames (Page 10217).

Repeat step 6, if you want to add another drive and another standard frame.

Enable isochronous operation of the drive in the device configuration

PROFINET drives can always be operated in isochronous mode or non-isochronous mode. Isochronous mode, however, increases the quality of the position control of the drive.

Proceed as follows if you want to control the drive in isochronous mode:

1. Open the device view in the drive.
2. In the properties dialog, select the tab "PROFINET Interface [X1] > Advanced options > Isochronous mode".
3. Select the "Isochronous mode" check box in this tab.

The entry for the frame also has to be selected for isochronous mode in the detailed view.

Interconnecting the port of the PLC with the port of the drive

1. Open the topology view in the device configuration.
2. Interconnect the port of the PLC with the port of the drive.

Configuring the PLC as the sync master and setting isochronous mode

1. Select the device view of the PLC.
2. In the properties dialog, select the tab "PROFINET Interface [X1] > Advanced options > Realtime settings > Synchronization".
3. Select "Sync master" from the "Synchronization role" drop-down list.
4. Click the "Domain settings" button.
5. Open the "Domain Management > Sync Domains" tab and set the desired "Send clock" (isochronous clock).

Select drive in the configuration of the technology object

1. Add a new axis technology object, or open the configuration of an existing axis.
2. Open the configuration "Hardware interface > Drive".

3. Select "PROFIdrive" from the "Drive type" drop-down list.
4. Select the Drive Object of the PROFINET drive from the "Drive" list.

For information on how to add a technology object, refer to the section Add technology object (Page 10264).

Result

The technology object is connected to the drive and the "MC Servo" organization block can be checked / configured.

The frame of the configured drive is assigned to the "PIP OB Servo" process image.

Checking/configuring the properties of the MC Servo

1. Open the "Program blocks" folder in the project tree.
2. Select the "MC Servo" organization block.
3. Select the "Properties" command in the shortcut menu.
4. Select the "Cycle time" entry in the area navigation.
5. The option "Synchronous to the bus" must be selected in the dialog box.
6. A "PROFINET IO_System" must be selected in the "Distributed I/O" drop-down list.
7. The application cycle of "MC-Servo" must correspond to the send clock of the bus or be reduced by an integral factor relative to the send clock of the bus.

Result

The PROFINET IO drive is now configured so that it can be controlled in the PROFINET IO network in isochronous mode.

The properties of the SINAMICS drive must be configured according to the configuration of the axis with a separate STARTER program.

Checking isochronous mode on the drive

If the configuration sequence described above is not adhered to during configuration of the axis, and drive-specific error occurs when the project is compiled, the setting for isochronous mode on the drive must be checked.

1. Open the device view in the drive.
2. Select standard message item in the device overview.
3. Select the properties dialog "General > I/O Addresses".
4. The following settings apply for the input and output addresses:
   - "Isochronous mode" is enabled.
   - "MC Servo" must be select the "Organization block".
   - "PIP OB Servo" must be select the "Process image".

See also

Frames (Page 10217)

Add technology object (Page 10264)

Add and configure PROFIBUS DP drives

Adding and configuring a PROFIBUS IO drive is described below using a **SINAMICS S120**. Adding and configuring other PROFIBUS drives may differ from the description in certain respects.

Requirements

- The SIMATIC S7-1500 device is created in the project.
- You have a basic knowledge of how to configure PROFIBUS DP networks.
- The desired drive can be selected in the hardware catalog.

If the drive is not available in the Hardware Catalog, then it must be installed in the "Extras" menu as a device description file (GSD).

Adding a drive and frame in the device configuration

1. Open the device configuration and change to the network view.
2. In the hardware catalog, open the folder "Additional Field Devices > PROFIBUS DP > Drives > Siemens AG > SINAMICS".
3. Select the folder of the desired drive with the desired version, then drag the drive object to the network view.
4. Assign the drive to the PROFIBUS interface of the PLC.
5. Open the drive in the device view.
6. Drag a standard frame from the hardware catalog to the slot of the device overview of the drive.

For more information on suitable frames, refer to the section Frames (Page 10217).

Use the "Axis disconnector" in the hardware catalog if you want to add another drive and another standard frame in the device overview.

Enable isochronous operation of the drive in the device configuration

PROFIBUS drives can always be operated in cyclic mode or isochronous mode. Isochronous mode, however, increases the quality of the position control of the drive.

Proceed as follows if you want to control the drive in isochronous mode:

1. Open the device view in the drive.
2. In the properties dialog, select the tab "General > Isochronous Mode".
3. Select the "Synchronize DP slave to constant DP bus cycle time" check box .

Setting isochronous mode

1. Select the network view.
2. Select the DP master system.
3. In the properties dialog, select the tab "General > Constant bus cycle time".
4. Select the desired "Constant DP bus cycle times".

Select drive in the configuration of the technology object

1. Add a new axis technology object, or open the configuration of an existing axis.
2. Open the configuration "Hardware interface > Drive".
3. Select "PROFIdrive" from the "Drive type" drop-down list.
4. Select the standard frame of the PROFIBUS drive from the "Drive" list.

For information on how to add a technology object, refer to the section Add technology object (Page 10264).

Result

The technology object is connected to the drive and the "MC Servo" organization block can be checked/configured.

The frame of the configured drive is assigned to the "PIP OB Servo" process image.

Checking/configuring the properties of the MC Servo

1. Open the "Program blocks" folder in the project tree.
2. Select the "MC Servo" organization block.
3. Select the "Properties" command in the shortcut menu.
4. Select the "Cycle time" entry in the area navigation.
5. The option "Synchronous to the bus" must be selected in the dialog box.
6. A "PROFINET IO system" must be selected in the "Distributed I/O" drop-down list.
7. The application cycle of "MC-Servo" must correspond to the send clock of the bus or be reduced by an integral factor relative to the send clock of the bus.

You can select a drive connected to the CPU via a communications processor/communications module (CP/CM) in the configuration of the technology object. You cannot select the DP master system of the CP/CM as the source clock for MC-Servo [OB91].

Result

The PROFIBUS DP drive is now configured so that it can be controlled in the PROFIBUS network in isochronous mode.

The properties of the SINAMICS drive must be configured according to the configuration of the axis with a separate STARTER program.

Checking isochronous mode on the drive

If the configuration sequence described above is not adhered to during configuration of the axis, and drive-specific error occurs when the project is compiled, isochronous mode can be checked on the drive.

1. Open the device view in the drive.
2. Select standard message item in the device overview.
3. Select the properties dialog "General > I/O Addresses".
4. The following settings apply for the input and output addresses:
   – "MC Servo" must be select the "Organization block".
   – "PIP OB Servo" must be select the "Process image".

See also

Add technology object (Page 10264)

Frames (Page 10217)

Adding and configuring drives with analog connections

The following describes how to add and configure a drive with an analog drive connection. A technology module in the rack of the PLC is used as an example of the connection of an incremental encoder.

Requirements

The SIMATIC S7-1500 device is created in the project.

Adding and configuring an analog output module in the device configuration

1. Open the device configuration of the PLC.
2. Select an analog output module from the hardware catalog and drag the module to the rack of the PLC.
3. Select the analog output module in the device view.
4. Open the "General" tab in the properties dialog and select there "Name of the Analog Output Module > I/O Addresses".
5. Enter the desired start address.
6. In the properties dialog, select the tab "General > IO Tags".
7. Enter the tag name for the desired analog output.

Adding and configuring a technology module

1. Switch to the device view of the PLC.
2. In the hardware catalog, open the folder "TM > Count > TM Count 2X24V".

3. Drag the counter module to a free slot in the rack.
4. Select the technology module in the device view.
5. In the properties dialog, open the "General" tab and select there "Count 2x24V > Basic Parameters > Channel X > Operating Mode" of the channel to be used.
6. Select "Position detection for motion control" option for "Selection of the operating mode for the channel".
7. Under "Module parameters", adapt the parameters of the incremental encoder (steps per revolution = increments per revolution).
8. Under "Reaction to CPU STOP", select the item "Continue working".

Selecting the drive and encoder in the configuration of the technology object

1. Add a new "Position axis" technology object, or open the configuration of an existing position axis.
2. Open the configuration "Hardware interface > Drive".
3. Select "Analog drive connection" from the "Drive type" drop-down list.
4. Select the previously defined tag name of the analog output from the "Output" list.
5. Open the configuration "Hardware interface > Encoder".
6. Under "Select encoder coupling" select "Encoder on technology module (TM)".
7. Select the channel of the incremental encoder from the "Technology module" list under "local modules".

You can learn how to add a technology object in section Add technology object (Page 10264).

Result

The analog drive connection and the encoder connection are configured.

The analog addresses and the addresses of the TM module are assigned to the process image "PIP OB Servo".

Checking the encoder connection / drive connection

The encoder data are applied to the position control cycle clock. If in doubt, check the following settings:

1. Switch to the device view of the PLC.
2. Select the module technology.
3. Open the properties dialog "Basic Parameters > I/O Addresses".
4. The following settings apply for the input and output addresses:
   - "MC Servo" must be select the "Organization block".
   - "PIP OB Servo" must be select the "Process image".
5. Select the analog module.

6. Open the properties dialog "Name of the Analog Module > I/O Addresses".
7. The settings from Step 4 apply for the input and output addresses

See also

Add technology object (Page 10264)

Add technology object

To add a technology object in the project navigator, proceed as follows:

Requirements

A project with a CPU S7-1500 has been created.

Procedure

1. Open the CPU's folder in the project navigator.
2. Open the "Technology Objects" folder.
3. Double-click "Add new object".
   The "Add new object" dialog box opens.
4. Select the "Motion Control" technology.
5. Open the folder "Motion Control" > "SIMATIC S7-1500".
6. If you want to add an axis from an older version, then click on the Version entry and select an alternative version of the technology.
7. Select the "TO_SpeedAxis" object for a speed-controlled axis, "TO_PositioningAxis" for a positioning axis, or "TO_ExternalEncoder" for an external encoder.
8. In the "Name" field, adapt the name of the axis to your requirements.
9. Select the "Manual" option if you want to change the suggested data block number.
10. Click on "Further Information" if you want to add your own information to the technology object.
11. Click on the "OK" button if you want to add the technology object.
    Click on the "Cancel" button if you want to discard the entries.
    Activate the "Add new and open" check box if you want to open the configuration after adding the technology object.

Result

The new technology object was created and saved in the project navigator in the folder "Technology objects".

If the organization blocks "MC Servo" and "MC Interpolator" had not yet been added, they have now been added.

Working with the configuration editor

You configure the properties of a technology object in the configuration window. To open the configuration window of the technology object in the project view, follow these steps:

1. Open the device "Technology objects" group in the project navigator.
2. Select the technology object and double-click on "Configuration".

The configuration is divided into categories which depend on the object type, for example: Basic parameters, hardware interface, extended parameters.

Configuration editor icons

Icons in the area navigation of the configuration show additional details about the status of the configuration:

	The configuration contains default values and is complete. The configuration contains only default values. With these default values you can use the technology object without additional changes.
	The configuration contains values set by the user and is complete. All input fields of the configuration contain valid values and at least one preset value has changed.
	The configuration is incomplete or incorrect At least one input field or drop-down list contains an invalid value. The corresponding field or the drop-down list is displayed on a red background. Click the field shows you the roll-out error message that indicates the cause of error.

Compare values

The "Compare values" function provides the following options:

- Comparison of configured start values of the project with the start values in the CPU and the actual values
- Direct editing of actual values and the start values of the project
- Immediate detection and display of input errors with suggested corrections
- Backup of actual values in the project
- Transfer of start values of the project to the CPU as actual values

Icons and operator controls

The following icons and operator controls are available:

Icon	Function
	Start value in CPU matches the configured Start value in the project
	Start value in CPU does not match the configured Start value in the project
	The comparison of the Start value in CPU with the configured Start value in the project cannot be performed
	Backs up the actual values in the project

Icon	Function
	Transfers updated start values in the project to the CPU (initialize setting values)
	Opens the "Compare values" dialog

Configuring the Positioning Axis technology object

Configuration - Basic Parameters

Axis name

Define the name of the positioning axis in this field. The technology object is listed in the project navigation under this name. The tags of the positioning axis can be used in the user program under this name.

Axis type

In this selection, configure whether the axis should perform linear or rotary motions.

User unit

In the drop-down list, select the desired measurement system for the position and velocity of the axis.

Modulo

Select the check box "Enable modulo", if you want to employ a recurring unit of measure for the axis (e.g. 0-360° for an axis of the "rotary" axis type).

- Start value
  In this field, define the position at which the modulo range should begin (e.g. 360° for an axis of the "rotary" axis type).
- Length
  In this field, define the length of the modulo range (e.g. 360° for an axis of the "rotary" axis type).

Hardware interface

Configuration - Drive

In the "Drive" configuration window, configure which drive type and which drive you want to use.

Drive type

In the drop-down list, select whether you want to deploy a PROFIdrive drive or a drive with an analog drive connection.

PROFIdrive drives are connected to the controller by means of a digital communication system (PROFINET or PROFIBUS). The communication occurs via PROFIdrive frames.

Drives with an analog drive connection receive the setpoint speed via an analog output signal (e.g. from -10 V to +10 V) from the PLC.

Drive (drive type: PROFIdrive)

In the "Drive" field, select an already configured PROFIdrive drive / slot. If a PROFIdrive drive was selected, then it can be configured using the "Device configuration" button.

Switch to the device configuration, and add a PROFIdrive drive in the network view, in the event that no PROFIdrive drive can be selected.

Output (drive type: Analog drive connection)

In the "Output" field, select the PLC tag of the analog output over which the drive is to be controlled. If an output was selected, then it can be configured using the "Device configuration" button.

In order to be able to select an output, an analog output module must have been added in the device configuration, and the PLC tag name for the analog output must be defined.

Activate enable output (drive type: Analog drive connection)

In the "Enable output" field, select the PLC tag of the digital output to enable the drive. With the enable output, the speed controller in the drive is enabled, or disabled.

In order to be able to select an enable output, a digital output module must have been added in the device configuration, and the PLC tag name for the digital output must be defined.

Note

If you do not use an enable output, the drive cannot be immediately disabled on the part of the system due to error responses or monitoring functions. A controlled stop of the drive is not guaranteed.

Enable ready input (drive type: Analog drive connection)

In the "Ready input" field, select the PLC tag of the digital input with which the drive is to report its operational readiness to the technology object. The power unit is switched on and the analog setpoint speed input is enabled.

In order to be able to select a ready input, a digital input module must have been added in the device configuration, and the PLC tag name for the digital input must be defined.

Note

The enable output and the ready input can be separately enabled.

The following conditions apply to the enabled ready input:

- The axis is not enabled (MC_Power Status=TRUE), until a signal is pending at the ready input.
- If the signal at the ready input goes away for an enabled axis, then the axis is disabled with an error.
- If the axis is disabled with the instruction MC_Power (Enable= FALSE), then the axis is disabled even with a pending signal at the ready input.

Configuration - Encoder

For position control, positioning axes require an actual position value in the form of an encoder position. The encoder position is transmitted to the controller by means of a PROFIdrive frame. In the "Encoder" configuration window, configure the encoder coupling with which the encoder is to be connected.

Select encoder coupling:

In this area, select how the encoder will be connected in keeping with the graphic representation.

- **Connection to drive (not with analog drive connection)**
  Select this option, if the encoder is connected to the drive. The configuration of the encoder occurs via the configuration of the PROFIdrive drive. The drive evaluates the encoder signals and sends them to the controller in a PROFIdrive frame.
- **Connection via technology module (TM)**
  Select this option, if the encoder is connected to a technology module (TM).
  - **Technology module:**
    In the "Technology module" field, select a technology module that has already been configured, and the channel to be used. Only technology modules set to the "Position detection for motion control" mode are displayed for selection.
    If no technology module is available for selection, change to the device configuration and add a technology module. If you have selected a technology module, you can change to the configuration of the technology module using the "Device configuration" button.
    You can operate the technology module centrally on an S7-1500 PLC, or distributed on a distributed I/O. To connect an incremental encoder use the technology module TM Count2x24V or TM PosInput2. To connect an absolute encoder use the technology module TM PosInput2.
- **Connection via PROFINET/PROFIBUS (PROFIdrive)**
  Select this option, if you are deploying a PROFIdrive compatible encoder.
  - **Encoder selection:**
    In the "Encoder selection" field, select an already configured encoder on PROFINET/PROFIBUS. If an encoder was selected, then it can be configured using the "Device configuration" button.
    Switch to the device configuration in the network view, and add an encoder, in the event that no encoder can be selected.

Configuration - Data exchange

Configuration - Data exchange

In the "Data exchange" configuration window, you can configure the data exchange of the drive and the encoder. The configuration varies according to the drive type and encoder coupling:

Analog driver connection - Encoder to technology module (TM) (Page 10270)

Analog driver connection - PROFIdrive encoder on PROFINET/PROFIBUS (Page 10272)

PROFIdrive Drive - Encoder to Drive (Page 10274)

PROFIdrive drive - Encoder on the technology module (TM) (Page 10276)

PROFIdrive drive - PROFIdrive encoder on PROFINET/PROFIBUS (Page 10278)

Analog driver connection - Encoder to technology module (TM)

Data exchange with the drive

In this area, you can configure the data exchange with the drive.

- Frame:
  Since the analog drive is controlled via an analog signal, the selection of the frame type is omitted.
- Reference speed:
  The reference speed of the drive is the speed, with which the drive spins when there is an output of 100% at the analog output. The reference speed must be configured in the drive, and transferred into the configuration of the technology object.
  The analog value that is output at 100% depends on the type of the analog output. As an example, with an analog output with +/- 10 V, the value 10 V is output at 100%.
  Analog outputs can be overloaded by about 17%. This means that an analog output can be operated in the range from -117% to 117%, insofar as the drive permits this.
- Maximum speed
  In this field, specify the maximum speed of the drive.
- Invert direction
  Select the check box if the rotation direction of the drive should be inverted.

Data exchange with encoder

In this area, you can configure how the encoder data is to be evaluated. The specifications must match those in the device configuration.

- Frame:
  The encoder data is transmitted with the drive frame. Therefore, no frame selection is available.
- Encoder type:
  Depending on the selected encoder type, configure the following parameters:

Encoder type	Rotary incremental
Steps per revolution:	In this field, configure the number of steps that the encoder resolves per revolution.
Bits for fine resolution in incremental actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).

Encoder type	Rotary absolute
Steps per revolution:	In this field, configure the number of steps that the encoder resolves per revolution.
Number of revolutions:	In this field, configure the number of revolutions that the absolute value encoder can detect.

Encoder type	Rotary absolute
Bits for fine resolution in incremental actual value (Gn_XIST1)	In this field, configure the number of reserved bits for fine resolution within the incremental actual value (Gn_XIST1).
Bits for the multiplication factor of the absolute value of the fine resolution (Gn_XIST2)	In this field, configure the number of reserved bits for the multiplication factor of the absolute value of the fine resolution (Gn_XIST2).

Encoder type	Rotary cyclic absolute
Steps per revolution:	In this field, configure the number of steps that the encoder resolves per revolution.
Number of revolutions:	In this field, configure the number of revolutions that the absolute value encoder can detect.
Bits for fine resolution in incremental actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).
Bits for the multiplication factor of the absolute value of the fine resolution (Gn_XIST2)	In this field, configure the number of reserved bits for the multiplication factor of the absolute value of the fine resolution (Gn_XIST2).

Encoder type	Linear incremental
Distance between two increments:	In this field, you can configure the distance between two steps of the encoder.
Bits for fine resolution in incremental actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).

Encoder type	Linear absolute
Distance between two increments:	In this field, you can configure the distance between two steps of the encoder.
Bits for fine resolution in incremental actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).
Bits for the multiplication factor of the absolute value of the fine resolution (Gn_XIST2)	In this field, configure the number of reserved bits for the multiplication factor of the absolute value of the fine resolution (Gn_XIST2).

Invert direction

Select this check box if you would like to invert the actual value of the encoder.

Analog driver connection - PROFIdrive encoder on PROFINET/PROFIBUS

Data exchange with the drive

In this area, you can configure the data exchange with the drive.

- Frame:
  Since the analog drive is controlled via an analog signal, the selection of the frame type is omitted.
- Reference speed:
  The reference speed of the drive is the speed, with which the drive spins when there is an output of 100% at the analog output. The reference speed must be configured in the drive, and transferred into the configuration of the technology object.
  The analog value that is output at 100% depends on the type of the analog output. As an example, with an analog output with +/- 10 V, the value 10 V is output at 100%.
  Analog outputs can be overloaded by about 17%. This means that an analog output can be operated in the range from -117% to 117%, insofar as the drive permits this.
- Maximum speed
  In this field, specify the maximum speed of the drive.
- Invert direction
  Select the check box if the rotation direction of the drive should be inverted.

Data exchange with encoder

In this area, you can configure the encoder frame and the criteria for how the encoder data is to be evaluated. The specifications must match those in the device configuration.

- Frame:
  In the drop-down list, select the frame of the encoder. The specification must match the setting in the device configuration.
- Encoder type:
  Depending on the selected encoder type, configure the following parameters:

Encoder type	Rotary incremental
Steps per revolution:	In this field, configure the number of steps that the encoder resolves per revolution.
Bits for fine resolution in incremental actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).

Encoder type	Rotary absolute
Steps per revolution:	In this field, configure the number of steps that the encoder resolves per revolution.
Number of revolutions:	In this field, configure the number of revolutions that the absolute value encoder can detect.

Encoder type	Rotary absolute
Bits for fine resolution in incremental actual value (Gn_XIST1)	In this field, configure the number of reserved bits for fine resolution within the incremental actual value (Gn_XIST1).
Bits for fine resolution in the absolute actual value (Gn_XIST2)	In this field, configure the number of reserved bits for the multiplication factor of the absolute value of the fine resolution (Gn_XIST2).

Encoder type	Rotary cyclic absolute
Steps per revolution:	In this field, configure the number of steps that the encoder resolves per revolution.
Number of revolutions:	In this field, configure the number of revolutions that the absolute value encoder can detect.
Bits for fine resolution in incremental actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).
Bits for fine resolution in the absolute actual value (Gn_XIST2)	In this field, configure the number of bits for fine resolution within the absolute actual value (Gn_XIST2).

Encoder type	Linear incremental
Distance between two increments:	In this field, you can configure the distance between two steps of the encoder.
Bits for fine resolution in incremental actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).

Encoder type	Linear absolute
Distance between two increments:	In this field, you can configure the distance between two steps of the encoder.
Bits for fine resolution in incremental actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).
Bits for fine resolution in the absolute actual value (Gn_XIST2)	In this field, configure the number of bits for fine resolution within the absolute actual value (Gn_XIST2).

Invert direction

Select this check box if you would like to invert the actual value of the encoder.

PROFIdrive Drive - Encoder to Drive

Data exchange with the drive

In this area, you can configure the data exchange with the drive.

- **Frame:**
  In the drop-down list, select the frame for the drive. The specification must match the setting in the device configuration.
- **Reference speed:**
  In this field, configure the reference speed of the drive in accordance with the manufacturer's specifications. The specification of the drive's speed occurs as a percentage of the reference speed in the range from -200% to 200%.
- **Maximum speed**
  In this field, specify the maximum speed of the drive.
- **Invert direction**
  Select the check box if the rotation direction of the drive should be inverted.

Data exchange with encoder

In this area, you can configure how the encoder data is to be evaluated. The specifications must match those in the device configuration.

- **Frame:**
  The encoder data is transmitted with the drive frame. Therefore, no frame selection is available.
- **Encoder type:**
  Depending on the selected encoder type, configure the following parameters:

Encoder type	Rotary incremental
Steps per revolution:	In this field, configure the number of steps that the encoder resolves per revolution.
Bits for fine resolution in incremental actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).

Encoder type	Rotary absolute
Steps per revolution:	In this field, configure the number of steps that the encoder resolves per revolution.
Number of revolutions:	In this field, configure the number of revolutions that the absolute value encoder can detect.
Bits for fine resolution in incremental actual value (Gn_XIST1)	In this field, configure the number of reserved bits for fine resolution within the incremental actual value (Gn_XIST1).
Bits for the multiplication factor of the absolute value of the fine resolution (Gn_XIST2)	In this field, configure the number of reserved bits for the multiplication factor of the absolute value of the fine resolution (Gn_XIST2).

Encoder type	Rotary cyclic absolute
Steps per revolution:	In this field, configure the number of steps that the encoder resolves per revolution.
Number of revolutions:	In this field, configure the number of revolutions that the absolute value encoder can detect.
Bits for fine resolution in incremental actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).
Bits for the multiplication factor of the absolute value of the fine resolution (Gn_XIST2)	In this field, configure the number of reserved bits for the multiplication factor of the absolute value of the fine resolution (Gn_XIST2).

Encoder type	Linear incremental
Distance between two increments:	In this field, you can configure the distance between two steps of the encoder.
Bits for fine resolution in incremental actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).

Encoder type	Linear absolute
Distance between two increments:	In this field, you can configure the distance between two steps of the encoder.
Bits for fine resolution in incremental actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).
Bits for the multiplication factor of the absolute value of the fine resolution (Gn_XIST2)	In this field, configure the number of reserved bits for the multiplication factor of the absolute value of the fine resolution (Gn_XIST2).

Invert direction

Select this check box if you would like to invert the actual value of the encoder.

PROFIdrive drive - Encoder on the technology module (TM)

Data exchange with the drive

In this area, you can configure the data exchange with the drive.

- Frame:
  In the drop-down list, select the frame for the drive. The specification must match the setting in the device configuration.
- Reference speed:
  In this field, configure the reference speed of the drive in accordance with the manufacturer's specifications. The specification of the drive's speed occurs as a percentage of the reference speed in the range from -200% to 200%.
- Maximum speed
  In this field, specify the maximum speed of the drive.
- Invert direction
  Select the check box if the rotation direction of the drive should be inverted.

Data exchange with encoder

In this area, you can configure how the encoder data is to be evaluated. The specifications must match those in the device configuration.

- Frame:
  The encoder data is transmitted with the drive frame. Therefore, no frame selection is available.
- Encoder type:
  Depending on the selected encoder type, configure the following parameters:

Encoder type	Rotary incremental
Steps per revolution:	In this field, configure the number of steps that the encoder resolves per revolution.
Bits for fine resolution in incremental actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).

Encoder type	Rotary absolute
Steps per revolution:	In this field, configure the number of steps that the encoder resolves per revolution.
Number of revolutions:	In this field, configure the number of revolutions that the absolute value encoder can detect.
Bits for fine resolution in incremental actual value (Gn_XIST1)	In this field, configure the number of reserved bits for fine resolution within the incremental actual value (Gn_XIST1).
Bits for the multiplication factor of the absolute value of the fine resolution (Gn_XIST2)	In this field, configure the number of reserved bits for the multiplication factor of the absolute value of the fine resolution (Gn_XIST2).

Encoder type	Rotary cyclic absolute
Steps per revolution:	In this field, configure the number of steps that the encoder resolves per revolution.
Number of revolutions:	In this field, configure the number of revolutions that the absolute value encoder can detect.
Bits for fine resolution in incremental actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).
Bits for the multiplication factor of the absolute value of the fine resolution (Gn_XIST2)	In this field, configure the number of reserved bits for the multiplication factor of the absolute value of the fine resolution (Gn_XIST2).

Encoder type	Linear incremental
Distance between two increments:	In this field, you can configure the distance between two steps of the encoder.
Bits for fine resolution in incremental actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).

Encoder type	Linear absolute
Distance between two increments:	In this field, you can configure the distance between two steps of the encoder.
Bits for fine resolution in incremental actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).
Bits for the multiplication factor of the absolute value of the fine resolution (Gn_XIST2)	In this field, configure the number of reserved bits for the multiplication factor of the absolute value of the fine resolution (Gn_XIST2).

Invert direction

Select this check box if you would like to invert the actual value of the encoder.

PROFIdrive drive - PROFIdrive encoder on PROFINET/PROFIBUS

Data exchange with the drive

In this area, you can configure the data exchange with the drive.

- **Frame:**
  In the drop-down list, select the frame for the drive. The specification must match the setting in the device configuration.
- **Reference speed:**
  In this field, configure the reference speed of the drive in accordance with the manufacturer's specifications. The specification of the drive's speed occurs as a percentage of the reference speed in the range from -200% to 200%.
- **Maximum speed**
  In this field, specify the maximum speed of the drive.
- **Invert direction**
  Select the check box if the rotation direction of the drive should be inverted.

Data exchange with encoder

In this area, you can configure the encoder frame and the criteria for how the encoder data is to be evaluated. The specifications must match those in the device configuration.

- **Frame:**
  In the drop-down list, select the frame of the encoder. The specification must match the setting in the device configuration.
- **Encoder type:**
  Depending on the selected encoder type, configure the following parameters:

Encoder type	Rotary incremental
Steps per revolution:	In this field, configure the number of steps that the encoder resolves per revolution.
Bits for fine resolution in incremental actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).

Encoder type	Rotary absolute
Steps per revolution:	In this field, configure the number of steps that the encoder resolves per revolution.
Number of revolutions:	In this field, configure the number of revolutions that the absolute value encoder can detect.
Bits for fine resolution in incremental actual value (Gn_XIST1)	In this field, configure the number of reserved bits for fine resolution within the incremental actual value (Gn_XIST1).
Bits for fine resolution in the absolute actual value (GN_XIST2)	In this field, configure the number of reserved bits for fine resolution within the absolute actual value (Gn_XIST2).

Encoder type	Rotary cyclic absolute
Steps per revolution:	In this field, configure the number of steps that the encoder resolves per revolution.
Number of revolutions:	In this field, configure the number of revolutions that the absolute value encoder can detect.
Bits for fine resolution in incremental actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).
Bits for fine resolution in the absolute actual value (GN_XIST2)	In this field, configure the number of reserved bits for fine resolution within the absolute actual value (Gn_XIST2).

Encoder type	Linear incremental
Distance between two increments:	In this field, you can configure the distance between two steps of the encoder.
Bits for fine resolution in incremental actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).

Encoder type	Linear absolute
Distance between two increments:	In this field, you can configure the distance between two steps of the encoder.
Bits for fine resolution in incremental actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).
Bits for fine resolution in the absolute actual value (GN_XIST2)	In this field, configure the number of reserved bits for fine resolution within the absolute actual value (Gn_XIST2).

Invert direction

Select this check box if you would like to invert the actual value of the encoder.

Extended Parameters

Configuration - Mechanics

Configuration - Mechanics

In the "Mechanics" configuration window, you can configure the mounting type of the encoder, and the adaptation of the encoder value to the mechanical conditions.

Type of encoder mounting

In the drop-down list, select how the encoder is mounted on the mechanism. The configuration varies depending on the axis type and the type of encoder mounting.

Axis type: Linear

Linear - On the motor shaft (Page 10280)

Linear - On the motor side (Page 10281)

Linear - External Measuring System (Page 10281)

Axis type: Rotary

Rotary - On the motor shaft (Page 10282)

Rotary - On the load side (Page 10282)

Rotary - External Measuring System (Page 10283)

Linear - On the motor shaft

The encoder is firmly mechanically connected with the motor shaft. Motor and encoder form a unit.

Load gear

- **Consider load gear / leadscrew pitch**
  Activate the check box, if you want to perform the configuration based on the gear ratio and the leadscrew pitch.
  Deactivate the check box, if you want to perform the configuration based on the motor revolution and the resulting distance.
- **Number of motor revolutions / number of load revolutions**
  The gear ratio of the load gear is specified as the ratio between motor revolutions and load revolutions. Specify here an integral number of motor revolutions and the resulting number of load revolutions.

Position parameter

- **Leadscrew pitch ("Consider load gear / leadscrew pitch" enabled)**
  In this field, configure the distance by which the load is moved when the leadscrew makes one revolution.
- **Load motion per motor revolution**
  In this field, configure the load distance for one motor revolution ("Consider load gear / leadscrew pitch" disabled):

Linear - On the motor side

The encoder is mechanically connected with the load side of the gear.

Load gear

- **Number of motor revolutions / number of load revolutions**
  The gear ratio of the load gear is specified as the ratio between motor revolutions and load revolutions. Specify here an integral number of motor revolutions and the resulting number of load revolutions.
  Select the same values for the number of motor revolutions and load revolutions, if no load gear is present.

Position parameter

- **Leadscrew pitch**
  In this field, configure the distance by which the load is moved when the leadscrew makes one revolution.

Linear - External Measuring System

An external measuring system provides the position values of the linear load motion.

Load gear

- **Number of motor revolutions / number of load revolutions**
  The gear ratio of the load gear is specified as the ratio between motor revolutions and load revolutions. Specify here an integral number of motor revolutions and the resulting number of load revolutions.
  Select the same values for the number of motor revolutions and load revolutions, if no load gear is present.

Position parameter

- **Load motion per motor revolution**
  In this field, configure the distance by which the load is moved when the motor makes one revolution.
- **Distance per encoder revolution**
  In this field, configure the distance recorded by the external measuring system per encoder revolution.

Rotary - On the motor shaft

The encoder is firmly mechanically connected with the motor shaft. Motor and encoder form a unit.

Load gear

- **Number of motor revolutions / number of load revolutions**
  The gear ratio of the load gear is specified as the ratio between motor revolutions and load revolutions. Specify here an integral number of motor revolutions and the resulting number of load revolutions.
  Select the same values for the number of motor revolutions and load revolutions, if no load gear is present.

Rotary - On the load side

The encoder is mechanically connected with the load side of the gear.

Load gear

- **Number of motor revolutions / number of load revolutions**
  The gear ratio of the load gear is specified as the ratio between motor revolutions and load revolutions. Specify here an integral number of motor revolutions and the resulting number of load revolutions.
  Select the same values for the number of motor revolutions and load revolutions, if no load gear is present.

Rotary - External Measuring System

An external measuring system provides the position values of the rotary load motion.

Load gear

- **Number of motor revolutions / number of load revolutions**
  The gear ratio of the load gear is specified as the ratio between motor revolutions and load revolutions. Specify here an integral number of motor revolutions and the resulting number of load revolutions.
  Select the same values for the number of motor revolutions and load revolutions, if no load gear is present.

Position parameter

- **External measuring system**
  In this field, configure the value indicated by the external measuring system per load revolution.

Configuration - Position monitoring

Configure the hardware and software limit switches of the axis in the "Position monitoring" configuration window.

Enable hardware limit switches

The check box activates the function of the negative and positive hardware limit switches. The negative hardware limit switch is located on the side in the negative direction of travel, and the positive hardware limit switch on the side in the positive direction of travel.

With enabled hardware limit switches, the drive is disabled when the hardware limit switch is approached or overshot. The drive brakes with the braking ramp configured in the drive.

Exception: If a hardware limit switch is overshot during an active home position approach with activated direction reversal at the hardware limit switch, then the axis stops with the configured maximum deceleration, and continues the home position approach in the opposite direction.

Note

Only use hardware limit switches that remain permanently switched after the approach. This switch state may only be canceled after the return into the permitted traversing range.

The digital inputs of the hardware limit switch are evaluated by default in cyclical data exchange. In the settings for the input module under "I/O addresses", select the entry "MC-Servo" for "Organization block" and the entry "PIP OB servo" for "Process image" if the hardware limit switches are to be evaluated in the position control cycle of the drive.

Negative / positive HW limit switch input

In these fields, select the PLC tags of the digital input for the negative and positive HW limit switches.

In order to be able to select an input, a digital input module must have been added in the device configuration, and the PLC tag name for the digital input must be defined.

CAUTION

During installation of hardware limit switches, attention must be paid to the filter times of the digital inputs

Based on the time for a servo clock cycle and the filter time of the digital inputs, the resulting delay times must be taken into account.

The filter time is configurable in individual digital input modules in the device configuration.

The digital inputs are set to a filter time of 6.4 ms by default. If these are used as hardware limit switches, undesired decelerations may occur. If this occurs, reduce the filter time for the relevant digital inputs.

The filter time can be set under "Input filter" in the device configuration of the digital inputs.

Signal level selection

Select the triggering signal level ("Lower level" / "Upper level") of the hardware limit switch. At "Lower level", the input signal is FALSE after the axis has reached or passed the hardware limit switch. At "Upper level", the input signal is TRUE after the axis has reached or passed the hardware limit switch.

Enable software limit switch

This check box activates the upper and lower software limit switches. An active motion is stopped after the axis has reached a position of the software limit switch. The technological object reports a fault. After acknowledging the error, the axis can again be moved in the direction of its working range.

Note

Enabled software limit switches only affect a homed axis.

Lower / upper SW limit switch input

Configure the operating range of the axis using the positions of the lower and upper software limit switches.

Configuration - Dynamic limits

In the "Dynamic limits" configuration window, configure the maximum values for velocity, acceleration, deceleration and jerk of the axis.

Maximum velocity

In this field, define the maximum permitted velocity of the axis.

Maximum acceleration / maximum deceleration - ramp-up time / ramp-down time

Set the desired acceleration in the "Ramp-up time" or "Acceleration" fields. The desired deceleration can be set in the "Ramp-down time" or "Deceleration" fields.

The relationship between ramp-up time and acceleration, and ramp-down time and deceleration can be seen in the following equations:

$$\text{Ramp-up time} = \frac{\text{Maximum velocity}}{\text{Acceleration}}$$

$$\text{Ramp-down time} = \frac{\text{Maximum velocity}}{\text{Deceleration}}$$

Note

A change in the maximum velocity influences the acceleration and deceleration values of the axis. The ramp-up and ramp-down times are retained.

The "maximum deceleration" for active homing with change of direction at the hardware limit switch must be set sufficiently large, to brake the axis before reaching the mechanical endstop.

Minimum smoothing time / maximum jerk

You can enter the jerk limit parameters in the "Minimum smoothing time" box, or alternatively in the "Maximum jerk" box:

- Set the desired jerk for the acceleration ramp and the deceleration ramp in the "Maximum jerk" box. The value 0 means that the jerk is not limited.
- Set the required smoothing time for the acceleration ramp in the "Minimum smoothing time" box.

Note

The configured smoothing time displayed in the configuration, applies only to the acceleration ramp.

If the values of the acceleration and deceleration differ, the smoothing time of the deceleration ramp is calculated and used according to the jerk of the acceleration ramp.

The smoothing time of the deceleration is adapted as follows:

- **Acceleration > Deceleration**
  A shorter smoothing time is used for the deceleration ramp compared with the acceleration ramp.
- **Acceleration < Deceleration**
  A longer smoothing time is used for the deceleration ramp compared with the acceleration ramp.
- **Acceleration = Deceleration**
  The smoothing times of the acceleration ramp and the deceleration ramp are the same.

If an error occurs, the axis decelerates with the configured emergency stop deceleration. A configured jerk limit is not taken into account for this.

The following equations show the relationship between the smoothing times and the jerk:

$$\text{Rounding off time (acceleration ramp)} = \frac{\text{Acceleration}}{\text{Jerk}}$$

$$\text{Rounding off time (deceleration ramp)} = \frac{\text{Deceleration}}{\text{Jerk}}$$

Traversing jobs initiated in the user program are executed with the selected jerk.

Configuration - Dynamic Defaults

In the "Dynamic defaults" configuration window, configure the default values for velocity, acceleration, deceleration and jerk of the axis.

The default values have an effect, when values < 0 are specified in Motion Control instructions for the "Velocity", "Acceleration", "Deceleration" or "Jerk" parameters. The default values can be applied separately for each of the parameters just listed.

The default values for acceleration and deceleration additionally have an effect on the traversing motions of active homing.

Velocity

In this field, define the default value for the velocity of the axis.

Acceleration / deceleration - ramp-up time / ramp-down time

Set the desired default value for acceleration in the "Ramp-up time" or "Acceleration" fields. The desired deceleration can be set in the "Ramp-down time" or "Deceleration" fields.

The relationship between ramp-up time and acceleration, and ramp-down time and deceleration can be seen in the following equations:

$$\text{Ramp-up time} = \frac{\text{Velocity}}{\text{Acceleration}}$$

$$\text{Ramp-down time} = \frac{\text{Velocity}}{\text{Deceleration}}$$

Note

A change in the velocity influences the acceleration and deceleration values of the axis. The ramp-up and ramp-down times are retained.

Smoothing time / jerk

You can enter the jerk limit parameters in the "Smoothing time" box, or alternatively in the "Jerk" box:

- Set the desired jerk for the acceleration ramp and the deceleration ramp in the "Jerk" box. The value 0 means that the jerk is disabled.
- Set the required smoothing time for the acceleration ramp in the "Smoothing time" box.

Note

The configured smoothing time displayed in the configuration, applies only to the acceleration ramp.

If the values of the acceleration and deceleration differ, the smoothing time of the deceleration ramp is calculated and used according to the jerk of the acceleration ramp.

The smoothing time of the deceleration is adapted as follows:

- **Acceleration > Deceleration**
  A shorter smoothing time is used for the deceleration ramp compared with the acceleration ramp.
- **Acceleration < Deceleration**
  A longer smoothing time is used for the deceleration ramp compared with the acceleration ramp.
- **Acceleration = Deceleration**
  The smoothing times of the acceleration ramp and the deceleration ramp are the same.

If an error occurs, the axis decelerates with the configured emergency stop deceleration. A configured jerk limit is not taken into account for this.

The following equations show the relationship between the smoothing times and the jerk:

$$\text{Rounding off time (acceleration ramp)} = \frac{\text{Acceleration}}{\text{Jerk}}$$

$$\text{Rounding off time (deceleration ramp)} = \frac{\text{Deceleration}}{\text{Jerk}}$$

Traversing jobs initiated in the user program are executed with the selected jerk.

Configuration - Emergency stop

In the "Emergency stop" configuration window, you can configure the emergency stop deceleration of the axis. In the event of an error, and when disabling the axis, the axis is brought to a standstill with this deceleration using the Motion Control instruction "MC_Power" (input parameter StopMode = 0).

Emergency stop deceleration

Configure the deceleration value for emergency stop in the "Emergency stop deceleration" field or the "Emergency stop ramp-down time" field.

The relationship between emergency stop ramp-down time and emergency stop deceleration can be seen in the following equation:

$$\text{Emergency stop ramp-down time} = \frac{\text{Maximum velocity}}{\text{Emergency stop deceleration}}$$

The configuration of the emergency stop deceleration is related to the configured maximum velocity of the axis. If the maximum velocity of the axis changes, then the value of the emergency stop deceleration also changes (the emergency stop ramp-down time remains unchanged).

Homing

Homing means matching the position value of a technology object to the real, physical location of the drive. Absolute positions on the axis can only be approached with a homed axis.

In S7-1500 Motion Control, axis homing is implemented with the Motion Control instruction, "MC_Home". A distinction is made between the following operation modes of the Motion Control instruction:

Operation modes of the Motion Control instruction "MC_Home"

- Active homing
  In active homing mode, the Motion Control instruction "MC_Home" performs the configured home position approach. Active traversing motions are aborted. When the reference mark is detected, the position of the axis is set according to the configuration.
- Passive homing
  During passive homing, the motion control instruction "MC_Home" does not carry out any homing motion. The traversing motion required for this step must be implemented by the user via other motion control instructions. Active traversing motions are not aborted upon start of passive homing. When the reference mark is detected, the axis is set according to the configuration.
- Direct homing absolute
  The axis position is set regardless of the homing switch. Active traversing motions are not aborted. The value of input parameter "Position" of motion control instruction "MC_Home" is set immediately as the reference point of the axis.
- Direct homing relative
  The axis position is set regardless of the homing switch. Active traversing motions are not aborted. The following statement applies to the axis position after homing:
  New axis position = current axis position + value of parameter "Position" of instruction "MC_Home".

Active homing

Configuration - Active homing

Configure the parameters for active homing in the "Active Homing" configuration window. "Active Homing" is performed using the Motion Control instruction "MC_Home" Mode = 4 and 5.

Select the Homing Mode

Select from among the following homing modes:

Zero mark via PROFIdrive frame and proximity switch (Page 10290)

Zero mark via PROFIdrive frame (Page 10291)

Reference mark via digital input (Page 10292)

Zero mark via PROFIdrive frame and proximity switch

Enable direction reversal at the hardware limit switch

Select this check box to use the hardware limit switch as a reversing cam for the home position approach. After the axis has reached the hardware limit switch during active homing, it is ramped down at the configured maximum deceleration rate and the direction is then reversed. The proximity switch is then sought in the reverse direction. If this function is not enabled and the axis reaches the hardware limit switch during active homing, then the drive is disabled and braked with the ramp configured in the drive.

Approach direction

Select the approach direction for seeking the proximity switch.

"Positive" is the approach direction in the direction of positive position values; "negative" in the direction of negative position values.

Homing direction

Select the direction in which the zero mark should be approached for homing.

Approach velocity

In this field, specify the velocity at which the proximity switch is sought during the home position approach. A possibly configured home position offset is traversed with the same velocity.

Homing velocity

In this field, specify the velocity at which the axis approaches the zero mark for homing. For zero mark detection, the proximity switch must be exited.

Home position offset

In the case of a differing zero mark position and home position, enter the corresponding home position offset in this field. The axis approaches the home position at approach velocity.

Home position

In this field, configure the absolute coordinate of the home position. The home position configured here is in effect, when the Motion Control instruction "MC_Home" is executed with Mode = 5.

Zero mark via PROFIdrive frame

Enable direction reversal at the hardware limit switch

Select this check box to use the hardware limit switch as a reversing cam for the home position approach. After the axis has reached the hardware limit switch during active homing, it is ramped down at the configured maximum deceleration rate and the direction is then reversed. The zero mark is then sought in the reverse direction. If this function is not enabled and the axis reaches the hardware limit switch during active homing, then the drive is disabled and braked with the ramp configured in the drive.

Homing direction

Select the direction in which the next zero mark should be approached for homing.

"Positive" is the homing direction in the direction of positive position values; "negative" in the direction of negative position values.

Approach velocity

In the "Zero mark via PROFIdrive frame" homing mode, the approach velocity for traversing the home position offset is used.

Homing velocity

In this field, specify the velocity at which the axis approaches the zero mark for homing.

Home position offset

In the case of a differing zero mark position and home position, enter the corresponding home position offset in this field. The axis approaches the home position at approach velocity.

Home position

In this field, configure the absolute coordinate of the home position. The home position configured here is in effect, when the Motion Control instruction "MC_Home" is executed with Mode = 5.

Reference mark via digital input

If a digital input is used as a reference mark, then the accuracy of the homing process is not as high as with hardware-supported homing using zero marks.

You can improve the accuracy by using a low homing velocity and a timer-supported input module.

Pay attention to the setting of short filter times at the digital input as well.

Digital input

In this field, select the PLC tag of the digital input, which should act as a reference mark (reference cam).

In order to be able to select an input, a digital input module must have been added in the device configuration, and the PLC tag name for the digital input must be defined.

Enable direction reversal at the hardware limit switch

Activate this check box to use the hardware limit switch as a reversing cam for the home position approach. After the axis has reached the hardware limit switch during active homing, it is ramped down at the configured maximum deceleration rate and then reversed. The home position switch is then sensed in reverse direction. If this function is not enabled and the axis reaches the hardware limit switch during active homing, then the drive is disabled and braked with the ramp configured in the drive.

Approach direction

Select the approach direction for seeking the reference mark.

"Positive" is the approach direction in the direction of positive position values; "negative" in the direction of negative position values.

Homing direction

Select the direction in which the reference mark should be approached for homing.

Reference mark

Select which switch position of the "digital input" is to be used as reference mark.

When a "digital input" is overshot, two switch edges are generated, which are spatially separated. The selection of the positive or negative side ensures that the reference mark is always evaluated based on the same mechanical position.

The positive side is the switch position with a greater position value; the negative side is the switch position with the lesser position value.

The selection of the side is independent of the approach direction, and independent of whether it causes a rising or falling edge.

Approach velocity

In this field, specify the velocity at which the "digital input" should be sought during the home position approach. A possibly configured home position offset is traversed with the same velocity.

Homing velocity

In this field, specify the velocity at which the axis approaches the home position for homing.

Home position offset

In the case of a differing switch position and home position, enter the corresponding home position offset in this field. The axis approaches the home position at approach velocity.

Home position

In this field, configure the absolute coordinate of the home position. The home position configured here is in effect, when the Motion Control instruction "MC_Home" is executed with Mode = 5.

Passive homing

Configuration - Passive homing

Configure the parameters for passive homing in the "Passive Homing" (homing on the fly) configuration window. The homing function "Passive Homing" is performed using the Motion Control instruction "MC_Home" Mode = 2 and 3.

Select the Homing Mode

Select from among the following homing modes:

Zero mark via PROFIdrive frame and proximity switch (Page 10293)

Zero mark via PROFIdrive frame (Page 10294)

Reference mark via digital input (Page 10294)

Zero mark via PROFIdrive frame and proximity switch

Homing direction

Select the direction in which the zero mark should be approached for homing. The next zero mark after exiting the proximity switch is used.

The following options are available:

- Positive
  Axis moves in the direction of higher position values.
- Negative
  Axis moves in the direction of lower position values.
- Current
  The currently effective approach direction is used for homing.

Home position

In this field, configure the absolute coordinate of the home position. The home position configured here is in effect, when the Motion Control instruction "MC_Home" is executed with Mode = 3.

Zero mark via PROFIdrive frame

Homing direction

Select the direction in which the next zero mark should be approached for homing. The following options are available:

- Positive
  Axis moves in the direction of higher position values.
- Negative
  Axis moves in the direction of lower position values.
- Current
  The currently effective approach direction is used for homing.

Home position

In this field, configure the absolute coordinate of the home position. The home position configured here is in effect, when the Motion Control instruction "MC_Home" is executed with Mode = 3.

Reference mark via digital input

Digital input

In this dialog field, select a digital input, which should act as a reference mark (reference cam).

Homing direction

Select the direction in which the reference mark should be approached for homing.

The following options are available:

- Positive
  Axis moves in the direction of higher position values.
- Negative
  Axis moves in the direction of lower position values.
- Current
  The currently effective approach direction is used for homing.

Reference mark

Select which switch position of the "digital input" is to be used as reference mark.

When a "digital input" is overshot, two switch edges are generated, which are spatially separated. The selection of the positive or negative side ensures that the reference mark is always evaluated based on the same mechanical position.

The positive side is the switch position with a greater position value; the negative side is the switch position with the lesser position value.

The selection of the side is independent of the approach direction, and independent of whether it causes a rising or falling edge.

Home position

In this field, configure the absolute coordinate of the home position. The home position configured here is in effect, when the Motion Control instruction "MC_Home" is executed with Mode = 3.

Position monitoring

Configuration - Positioning monitoring

In the "Positioning Monitoring" configuration window, configure the criteria for monitoring the target position.

Positioning window:

In this field, configure the size of the positioning window. If the axis is located within this window, then the position is considered to be "reached".

Positioning tolerance time:

In this field, configure the tolerance time, in which the position value must reach the positioning window.

Minimum dwell time in the positioning window:

In this field, configure the minimum dwell time. The current position value must be located in the positioning window for at least the "minimum dwell time".

If one of the criteria is violated, then the axis is stopped and a positioning alarm is displayed.

Configuration - Following error

In the "Following Error" configuration window, configure the permissible deviation of the actual position of the axis from the setpoint position. The following error can be dynamically adapted to the current velocity of the axis.

Enable following error monitoring

Activate this check box, if you want to enable following error monitoring. When following error monitoring is enabled, the axis will be stopped in the error range (orange); in the warning range an alarm will be displayed.

When following error monitoring is disabled, the configured limits have no effect.

Maximum following error:

In this field, configure the following error that is permissible at maximum velocity.

Warning level:

In this field, configure a percentage of the current following error limit, above which a following error warning should be output.

Example: The current maximum following error is 100 mm; the warning level is configured at 90%. If the current following error exceeds a value of 90 mm, then a following error warning will be output.

Following error:

In this field, configure the permissible following error for low velocities (without dynamic adaptation).

Beginning of dynamic adaptation:

In this field, configure the velocity above which the following error should be dynamically adapted. Above this velocity the following error will be adapted up to the maximum following error at the maximum velocity.

Configuration - Standstill signal

In the "Standstill Signal" configuration window, configure the criteria for standstill detection.

Standstill window:

In this field, configure the size of the standstill window. For standstill to be indicated, the velocity of the axis must be within this window.

Minimum dwell time in the standstill window:

In this field, configure the minimum dwell time in the standstill window. The velocity of the axis must also be in the standstill window for the specified duration.

If both criteria are met, then the standstill of the axis is indicated.

Configuration - Control loop

In the "Control loop" configuration window, configure the gain Kv of the position control loop.

The Kv factor affects the following parameters:

- Positioning accuracy and stop control
- Uniformity of motion
- Positioning time

The better the mechanical conditions of the axis are (high stiffness), the higher the Kv factor can be configured. This reduces the following error, and a higher dynamic response is achieved.

For basic information, refer to the section " Auto-Hotspot ".

Precontrol

In this box, configure the percentage velocity precontrol of the position control loop.

Gain

In the input field, enter the gain Kv of the control loop.

Dynamic Servo Control (DSC):

For position-controlled axes (positioning axes), the position control can occur either in the CPU or in the drive, provided the drive supports the Dynamic Servo Control (DSC) control process. Select your preferred control process:

- **Position and speed control in the drive (DSC enabled)**
- **Position and speed control in the PLC**

Note

Dynamic Servo Control (DSC) is only possible with standard frame 5.

See also

Function and structure of the optimization (Page 10344)

Configuring the Speed-Control Axis technology object

Configuration - Basic Parameters

Configure the basic properties of the technology object in the "Basic Parameters" configuration window.

Axis name

Define the name of the speed axis in this field. The technology object is listed in the project navigation under this name. The tags of the speed axis can be used in the user program under this name.

User unit

Select the measuring system for speed from the drop-down list box.

Hardware interface

Configuration - Drive

In the "Drive" configuration window, configure which drive type and which drive you want to use.

Drive type

In the drop-down list, select whether you want to deploy a PROFIdrive drive or a drive with an analog drive connection.

PROFIdrive drives are connected to the controller by means of a digital communication system (PROFINET or PROFIBUS). The communication occurs via PROFIdrive frames.

Drives with an analog drive connection receive the setpoint speed via an analog output signal (e.g. from -10 V to +10 V) from the PLC.

Drive (drive type: PROFIdrive)

In the "Drive" field, select an already configured PROFIdrive drive / slot. If a PROFIdrive drive was selected, then it can be configured using the "Device configuration" button.

Switch to the device configuration, and add a PROFIdrive drive in the network view, in the event that no PROFIdrive drive can be selected.

Output (drive type: Analog drive connection)

In the "Output" field, select the PLC tag of the analog output over which the drive is to be controlled. If an output was selected, then it can be configured using the "Device configuration" button.

In order to be able to select an output, an analog output module must have been added in the device configuration, and the PLC tag name for the analog output must be defined.

Activate enable output (drive type: Analog drive connection)

In the "Enable output" field, select the PLC tag of the digital output to enable the drive. With the enable output, the speed controller in the drive is enabled, or disabled.

In order to be able to select an enable output, a digital output module must have been added in the device configuration, and the PLC tag name for the digital output must be defined.

Note

If you do not use an enable output, the drive cannot be immediately disabled on the part of the system due to error responses or monitoring functions. A controlled stop of the drive is not guaranteed.

Enable ready input (drive type: Analog drive connection)

In the "Ready input" field, select the PLC tag of the digital input with which the drive is to report its operational readiness to the technology object. The power unit is switched on and the analog setpoint speed input is enabled.

In order to be able to select a ready input, a digital input module must have been added in the device configuration, and the PLC tag name for the digital input must be defined.

Note

The enable output and the ready input can be separately enabled.

The following conditions apply to the enabled ready input:

- The axis is not enabled (MC_Power Status=TRUE), until a signal is pending at the ready input.
- If the signal at the ready input goes away for an enabled axis, then the axis is disabled with an error.
- If the axis is disabled with the instruction MC_Power (Enable= FALSE), then the axis is disabled even with a pending signal at the ready input.

See also

Configuration - Data exchange (Page 10299)

Configuration - Data exchange

Data transmission to the drive

In this area, you can configure the data transmission to the drive.

Message frame type (drive type PROFIdrive)

In the drop-down list, select the message frame to the drive. The data must match the configuration in the device configuration.

Reference speed (drive type PROFIdrive)

In this field, configure the reference speed of the drive in accordance with the manufacturer's specifications. The specification of the drive's speed occurs as a percentage of the reference speed in the range from -200% to 200%.

Reference speed (drive type analog drive connection)

The reference speed of the drive is the speed, with which the drive spins when there is an output of 100% at the analog output. The reference speed must be configured in the drive, and transferred into the configuration of the technology object.

The analog value that is output at 100% depends on the type of the analog output. As an example, with an analog output with +/- 10 V, the value 10 V is output at 100%.

Analog outputs can be overloaded by about 17%. This means that an analog output can be operated in the range from -117% to 117%, insofar as the drive permits this.

Maximum speed

In this field, specify the maximum speed of the drive.

Inverting the direction

Enable the check box if the rotation direction of the drive should be inverted.

Extended Parameters

Configuration - Mechanics

Configure the connection of the load to the drive in the "Mechanics" configuration window.

Use gearbox parameters for calculation

Activate this check box, if you want to take account of a load gear when calculating the load speed.

Number of motor revolutions / number of load revolutions

The gear ratio of the load gear is specified as the ratio between motor revolutions and load revolutions. Specify here an integral number of motor revolutions and the resulting number of load revolutions.

Configuration - Dynamic limits

In the "Dynamic limits" configuration window, configure the maximum values for speed, acceleration, deceleration and jerk of the axis.

Maximum speed

In this field, define the maximum permitted speed of the axis.

Maximum acceleration / maximum deceleration - ramp-up time / ramp-down time

Set the desired acceleration in the "Ramp-up time" or "Acceleration" fields. The desired deceleration can be set in the "Ramp-down time" or "Deceleration" fields.

The relationship between ramp-up time and acceleration, and ramp-down time and deceleration can be seen in the following equations:

$$\text{Ramp-up time} = \frac{\text{Maximum velocity}}{\text{Acceleration}}$$

$$\text{Ramp-down time} = \frac{\text{Maximum speed}}{\text{Deceleration}}$$

Note

A change in the maximum speed influences the acceleration and deceleration values of the axis. The ramp-up and ramp-down times are retained.

Minimum smoothing time / maximum jerk

You can enter the jerk limit parameters in the "Minimum smoothing time" box, or alternatively in the "Maximum jerk" box:

- Set the desired jerk for the acceleration ramp and the deceleration ramp in the "Maximum jerk" box. The value 0 means that the jerk is not limited.
- Set the required smoothing time for the acceleration ramp in the "Minimum smoothing time" box.

Note

The configured smoothing time displayed in the configuration, applies only to the acceleration ramp.

If the values of the acceleration and deceleration differ, the smoothing time of the deceleration ramp is calculated and used according to the jerk of the acceleration ramp.

The smoothing time of the deceleration is adapted as follows:

- **Acceleration > Deceleration**
  A shorter smoothing time is used for the deceleration ramp compared with the acceleration ramp.
- **Acceleration < Deceleration**
  A longer smoothing time is used for the deceleration ramp compared with the acceleration ramp.
- **Acceleration = Deceleration**
  The smoothing times of the acceleration ramp and the deceleration ramp are the same.

If an error occurs, the axis decelerates with the configured emergency stop deceleration. A configured jerk limit is not taken into account for this.

The following equations show the relationship between the smoothing times and the jerk:

$$\text{Rounding off time (acceleration ramp)} = \frac{\text{Acceleration}}{\text{Jerk}}$$

$$\text{Rounding off time (deceleration ramp)} = \frac{\text{Deceleration}}{\text{Jerk}}$$

Traversing jobs initiated in the user program are executed with the selected jerk.

Configuration - Dynamic Defaults

In the "Dynamic defaults" configuration window, configure the default values for speed, acceleration, deceleration and jerk of the axis.

The default values have an effect, when values < 0 are specified in Motion Control instructions for the "Velocity", "Acceleration", "Deceleration" or "Jerk" parameters. The default values can be applied separately for each of the parameters just listed.

Velocity

In this field, define the default value for the speed of the axis.

Acceleration / deceleration - ramp-up time / ramp-down time

Set the desired default value for acceleration in the "Ramp-up time" or "Acceleration" fields. The desired deceleration can be set in the "Ramp-down time" or "Deceleration" fields.

The relationship between ramp-up time and acceleration, and ramp-down time and deceleration can be seen in the following equations:

$$\text{Ramp-up time} = \frac{\text{Speed}}{\text{Acceleration}}$$

$$\text{Ramp-down time} = \frac{\text{Speed}}{\text{Deceleration}}$$

Note

A change in the velocity influences the acceleration and deceleration values of the axis. The ramp-up and ramp-down times are retained.

Smoothing time / jerk

You can enter the jerk limit parameters in the "Smoothing time" box, or alternatively in the "Jerk" box:

- Set the desired jerk for the acceleration ramp and the deceleration ramp in the "Jerk" box. The value 0 means that the jerk is disabled.
- Set the required smoothing time for the acceleration ramp in the "Smoothing time" box.

Note

The configured smoothing time displayed in the configuration, applies only to the acceleration ramp.

If the values of the acceleration and deceleration differ, the smoothing time of the deceleration ramp is calculated and used according to the jerk of the acceleration ramp.

The smoothing time of the deceleration is adapted as follows:

- **Acceleration > Deceleration**
  A shorter smoothing time is used for the deceleration ramp compared with the acceleration ramp.
- **Acceleration < Deceleration**
  A longer smoothing time is used for the deceleration ramp compared with the acceleration ramp.
- **Acceleration = Deceleration**
  The smoothing times of the acceleration ramp and the deceleration ramp are the same.

If an error occurs, the axis decelerates with the configured emergency stop deceleration. A configured jerk limit is not taken into account for this.

The following equations show the relationship between the smoothing times and the jerk:

$$\text{Rounding off time (acceleration ramp)} = \frac{\text{Acceleration}}{\text{Jerk}}$$

$$\text{Rounding off time (deceleration ramp)} = \frac{\text{Deceleration}}{\text{Jerk}}$$

Traversing jobs initiated in the user program are executed with the selected jerk.

Configuration - Emergency stop

In the "Emergency stop ramp" configuration window, you can configure the emergency stop deceleration of the axis. In the event of an error, and when disabling the axis, the axis is brought to a standstill with this deceleration using the Motion Control instruction "MC_Power" (input parameter StopMode = 0).

Emergency stop deceleration

Configure the deceleration value for emergency stop in the "Emergency stop deceleration" field or the "Emergency stop ramp-down time" field.

The relationship between emergency stop ramp-down time and emergency stop deceleration can be seen in the following equation:

$$\text{Emergency stop ramp-down time} = \frac{\text{Maximum speed}}{\text{Emergency stop deceleration}}$$

The configuration of the emergency stop deceleration is related to the configured maximum speed of the axis. If the maximum speed of the axis changes, then the value of the emergency stop deceleration also changes (the emergency stop ramp-down time remains unchanged).

Configuring the External Encoder technology object

Configuration - Basic Parameters

Configure the basic properties of the technology object in the "Basic Parameters" configuration window.

Name of the external encoder

Define the name of the external encoder in this field. The technology object is listed in the project navigation under this name. The tags of the external encoder can be used in the user program under this name.

Type of the external encoder

In this selection, configure whether the external encoder records linear or rotary motions.

User unit

In the drop-down list, select the desired measurement system for the position and velocity of the external encoder.

Modulo

Select the check box "Activate modulo", if you want to employ a recurring unit of measure for the external encoder (e.g. 0-360° for an external encoder of the "rotary" type).

- Start value
  In this field, define at which position the modulo range should begin (e.g. 360° for an external encoder of the "rotary" type).
- Length
  In this field, define the length of the modulo range (e.g. 360° for an external encoder of the "rotary" type).

Hardware interface

Configuration - Encoder

The external encoder records the position of an externally controlled drive. The encoder required for this purpose communicates the encoder position to the controller by means of a PROFIdrive frame. In the "Encoder" configuration window, configure the encoder coupling with which the encoder is to be connected.

Select encoder coupling:

In this area, select how the encoder will be connected in keeping with the graphic representation.

- Connection via technology module (TM)
  Select this option, if the encoder is connected to a technology module (TM).
  - Technology module:
    In the "Technology module" field, select a technology module that has already been configured, and the channel to be used. Only technology modules set to the "Position detection for motion control" mode are displayed for selection.
    If no technology module is available for selection, change to the device configuration and add a technology module. If you have selected a technology module, you can change to the configuration of the technology module using the "Device configuration" button.
    You can operate the technology module centrally on an S7-1500 PLC, or distributed on a distributed I/O. To connect an incremental encoder use the technology module TM Count2x24V or TM PosInput2. To connect an absolute encoder use the technology module TM PosInput2.
- Connection via PROFINET/PROFIBUS (PROFIdrive)
  Select this option, if you are deploying a PROFIdrive compatible encoder.
  - Encoder selection:
    In the "Encoder selection" field, select an already configured encoder on PROFINET/PROFIBUS. If an encoder was selected, then it can be configured using the "Device configuration" button.
    Switch to the device configuration in the network view, and add an encoder, in the event that no encoder can be selected.

Configuration - Data exchange

Configuration - Data exchange

In the "Data transmission" configuration window, configure the data transmission to the encoder. The configuration varies according to the encoder coupling:

Encoder to technology module (Page 10307)

Encoder to PROFINET/PROFIBUS (Page 10308)

Encoder to technology module

Encoder message frame

In this area, you can configure the encoder message frame and the criteria for how the encoder data are to be evaluated. The data must match the data in the device configuration.

Message frame

In the drop-down list for the technology module, select the message frame that you have configured in the technology module.

Encoder type

In this area, you can configure how the encoder data are to be evaluated. The data must match the data in the device configuration.

Encoder type	Rotary incremental
Steps per revolution:	In this field, configure the number of steps that the encoder resolves per revolution.
Bits for fine resolution in the incremental actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).

Encoder type	Rotary absolute
Steps per revolution:	In this field, configure the number of steps that the encoder resolves per revolution.
Number of revolutions:	In this field, configure the number of revolutions that the absolute value encoder can detect.
Bits for fine resolution in the incremental actual value (Gn_XIST1)	In this field, configure the number of reserved bits for fine resolution within the incremental actual value (Gn_XIST1).
Bits for the multiplication factor of the absolute value of the fine resolution (Gn_XIST2)	In this field, configure the number of reserved bits for the multiplication factor of the absolute value of the fine resolution (Gn_XIST2).

Encoder type	Rotary cyclic absolute
Steps per revolution:	In this field, configure the number of steps that the encoder resolves per revolution.
Number of revolutions:	In this field, configure the number of revolutions that the absolute value encoder can detect.

Encoder type	Rotary cyclic absolute
Bits for fine resolution in the incremental actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).
Bits for the multiplication factor of the absolute value of the fine resolution (Gn_XIST2)	In this field, configure the number of reserved bits for the multiplication factor of the absolute value of the fine resolution (Gn_XIST2).

Encoder type	Linear incremental
Distance between two increments:	In this field, you can configure the distance between two steps of the encoder.
Bits for fine resolution in the incremental actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).

Encoder type	Linear absolute
Distance between two increments:	In this field, you can configure the distance between two steps of the encoder.
Bits for fine resolution in the incremental actual value (Gn_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (Gn_XIST1).
Bits for the multiplication factor of the absolute value of the fine resolution (Gn_XIST2)	In this field, configure the number of reserved bits for the multiplication factor of the absolute value of the fine resolution (Gn_XIST2).

Inverting the direction

Select this check box if you would like to invert the actual value of the encoder.

See also

Configuration - Data exchange (Page 10306)

Encoder to PROFINET/PROFIBUS (Page 10308)

Encoder to PROFINET/PROFIBUS

Encoder frame

In this area, you can configure the encoder frame and the criteria for how the encoder data is to be evaluated. The specifications must match those in the device configuration.

Data transmission

In the drop-down list, select the frame of the encoder. The specification must match the setting in the device configuration.

Encoder type

Depending on the selected encoder type, configure the following parameters:

Encoder type	Rotary incremental
Steps per revolution:	In this field, configure the number of steps that the encoder resolves per revolution.
Bits for fine resolution in incremental actual value (GN_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (GN_XIST1).

Encoder type	Rotary absolute
Steps per revolution:	In this field, configure the number of steps that the encoder resolves per revolution.
Number of revolutions:	In this field, configure the number of revolutions that the absolute value encoder can detect.
Bits for fine resolution in incremental actual value (GN_XIST1)	In this field, configure the number of reserved bits for fine resolution within the incremental actual value (GN_XIST1).
Bits for fine resolution in the absolute actual value (GN_XIST2)	In this field, configure the number of reserved bits for fine resolution within the absolute actual value (GN_XIST2).

Encoder type	Rotary cyclic absolute
Steps per revolution:	In this field, configure the number of steps that the encoder resolves per revolution.
Number of revolutions:	In this field, configure the number of revolutions that the absolute value encoder can detect.
Bits for fine resolution in incremental actual value (GN_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (GN_XIST1).
Bits for fine resolution in the absolute actual value (GN_XIST2)	In this field, configure the number of reserved bits for fine resolution within the absolute actual value (GN_XIST2).

Encoder type	Linear incremental
Distance between two increments:	In this field, you can configure the distance between two steps of the encoder.
Bits for fine resolution in incremental actual value (GN_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (GN_XIST1).

Encoder type	Linear absolute
Distance between two increments:	In this field, you can configure the distance between two steps of the encoder.
Bits for fine resolution in incremental actual value (GN_XIST1)	In this field, configure the number of bits for fine resolution within the incremental actual value (GN_XIST1).
Bits for fine resolution in the absolute actual value (GN_XIST2)	In this field, configure the number of reserved bits for fine resolution within the absolute actual value (GN_XIST2).

Invert direction

Select this check box if you would like to invert the actual value of the encoder.

See also

Configuration - Data exchange (Page 10306)

Encoder to technology module (Page 10307)

Extended Parameters

Configuration - Mechanics

Configuration - Mechanics

Configure the encoder parameters for the position of the externally controlled drive in the "Mechanics" configuration window.

The configuration varies according to the type of encoder:

Linear (Page 10311)

Rotary (Page 10311)

Linear

Distance between two increments

In this field you can configure the distance between two steps of the incremental encoder or absolute value encoder.

See also

Configuration - Mechanics (Page 10310)

Rotary (Page 10311)

Rotary

Use gearbox parameters for calculation

Activate this check box, if you want to take account of a load gear when calculating the actual position. Deactivate this check box, if you want to configure the distance per encoder revolution.

Measuring gearbox

- Number of encoder revolutions / number of load revolutions
  The gear ratio of the measuring gearbox is specified as the ratio between encoder revolutions and load revolutions. Specify here an integral number of load revolutions and the resulting number of encoder revolutions.
  Select the same values for the number of motor revolutions and load revolutions, if no load gear is present.

Position parameter

- Distance per encoder revolution
  In this field, configure the value indicated by the external measuring system per encoder revolution.

See also

Configuration - Mechanics (Page 10310)

Linear (Page 10311)

Homing

Configuration - Homing

Configure the parameters for homing the external encoder in the "Homing" configuration window. Homing is performed using the Motion Control instruction "MC_Home" Mode = 2 and 3.

Select the Homing Mode

Select from among the following homing modes:

Zero mark via PROFIdrive frame and proximity switch (Page 10312)

Zero mark via PROFIdrive frame (Page 10313)

Reference mark via digital input (Page 10313)

Zero mark via PROFIdrive frame and proximity switch

Homing direction

Select the direction in which the zero mark should be approached for homing. The next zero mark after exiting the proximity switch is used.

The following options are available:

- Positive
  Axis moves in the direction of higher position values.
- Negative
  Axis moves in the direction of lower position values.
- Current
  The currently effective approach direction is used for homing.

Home position

In this field, configure the absolute coordinate of the home position. The home position configured here is in effect, when the Motion Control instruction "MC_Home" is executed with Mode = 3.

See also

Configuration - Homing (Page 10312)

Zero mark via PROFIdrive frame (Page 10313)

Reference mark via digital input (Page 10313)

Zero mark via PROFIdrive frame

Homing direction

Select the direction in which the next zero mark should be approached for homing. The following options are available:

- **Positive**
  Axis moves in the direction of higher position values.
- **Negative**
  Axis moves in the direction of lower position values.
- **Current**
  The currently effective approach direction is used for homing.

Home position

In this field, configure the absolute coordinate of the home position. The home position configured here is in effect, when the Motion Control instruction "MC_Home" is executed with Mode = 3.

See also

Configuration - Homing (Page 10312)

Zero mark via PROFIdrive frame and proximity switch (Page 10312)

Reference mark via digital input (Page 10313)

Reference mark via digital input

Digital input

In this dialog field, select a digital input, which should act as a reference mark (reference cam).

Homing direction

Select the direction in which the reference mark should be approached for homing.

The following options are available:

- **Positive**
  Axis moves in the direction of higher position values.
- **Negative**
  Axis moves in the direction of lower position values.
- **Current**
  The currently effective approach direction is used for homing.

Reference mark

Select which switch position of the "digital input" is to be used as reference mark.

When a "digital input" is overshot, two switch edges are generated, which are spatially separated. The selection of the positive or negative side ensures that the reference mark is always evaluated based on the same mechanical position.

The positive side is the switch position with a greater position value; the negative side is the switch position with the lesser position value.

The selection of the side is independent of the approach direction, and independent of whether it causes a rising or falling edge.

Home position

In this field, configure the absolute coordinate of the home position. The home position configured here is in effect, when the Motion Control instruction "MC_Home" is executed with Mode = 3.

See also

Configuration - Homing (Page 10312)

Zero mark via PROFIdrive frame (Page 10313)

Zero mark via PROFIdrive frame and proximity switch (Page 10312)

Configuration - Standstill signal

In the "Standstill Signal" configuration window, configure the criteria for standstill detection.

Standstill window:

In this field, configure the size of the standstill window. For standstill to be indicated, the velocity of the external encoder must be within this window.

Minimum dwell time in the standstill window:

In this field, configure the minimum dwell time in the standstill window. The velocity of the external encoder must also be in the standstill window for the specified duration.

If both criteria are met, then the standstill of the external encoder is indicated.

Programming

Introduction

The "Programming" section contains general information on supplying and evaluating the Motion Control instructions and on technology data blocks.

An overview of the Motion Control instructions can be found in the Functions (Page 10211) section.

You can transmit jobs to the technology object by means of Motion Control instructions in the user program. You define the job using the input parameters of the Motion Control instructions. The current job status is indicated in the output parameters.

The technology data block is available to you as an additional interface to the technology object. Via your user program, you can access the technology data block's data.

Technology data block

Introduction

The technology data block represents the technology object and contains all configuration data, setpoint and actual values, and status information of the technology object. The technology data block is automatically created when the technology object is created. In your user program you can access the technology data block's data.

A listing and description of the tags can be found in the Appendix (Page 10365).

Evaluating the technology data block

Description

Access to data in the technology data block occurs in accordance with the access to standard data blocks.

Reading values from the technology data block

In your user program you can read actual values (e.g. current position) and status information, or detect error messages in the technology object. When you program a query in your user program (e.g. current velocity), the value is directly read from the technology object.

Reading values from the technology data block takes longer than for other data blocks. If you use these tags several times in a single cycle of your user program, it is recommended to copy the tag values to local tags, and use the local tags in your program.

Writing values to the technology data block

By means of the configuration of the technology object in the TIA Portal, the corresponding data are written to the technology data block. After they have been loaded into the CPU, these data are stored in the CPU on the SIMATIC Memory Card (load memory).

- **Required changes in the technology data block**
  In the following cases, it may be necessary for the user program to write values to the technology data block:
  - Adaptation of the configuration of the technology object (e.g. dynamic limits, software limit switches)
  - Use of overrides
  - Adaptation of position control (e.g. "Kv" parameter)
- **Effectiveness of changes in the technology data block**
  Changes to values in the technology data block by the user program can take effect at various points in time. The relevant property of the individual tags can be found in their descriptions in the Appendix (Page 10365).
  - **Effective immediately:**
    LREAL data type
    (e.g. <TO>.Override.Velocity)
    Value changes take effect immediately when writing these tags. The technology object performs a range check on the written value, and immediately starts using the new value. If range limits are violated when writing, the technology object automatically corrects the values. If the value is below the range, then the value is set to the lower limit of the range; if the range is exceeded, then the value is set to the upper limit of the range. You write changes using direct assignments. The changes are retained until the next POWER OFF of the CPU or restart of the technology object.
    DINT/BOOL data types
    (e.g. <TO>.Senso[n].ActiveHoming.Direction)
    Changes are only permitted in the defined value range. Value changes outside the value range are not applied. If you enter invalid values, the programming error OB (OB 121) is started. You write changes using direct assignments. The changes are retained until the next POWER OFF of the CPU or restart of the technology object.
  - **Effective after restart:**
    (e.g. <TO>.Homing.AutoReversal)
    Since restart-relevant tags have dependencies on other tags, value changes cannot be applied at any arbitrary time. The changes are only used after reinitialization (restart) of the technology object. During a restart the technology object is reinitialized with the data in load memory. You therefore write changes to the starting value in load memory, using the extended instruction "WRIT_DBL" (write to data block in load memory).
    You trigger the restart in your user program using the Motion Control instruction "MC_Reset" with parameter "Restart" = TRUE. Additional information on a restart can be found in the Restarting Technology Objects (Page 10336) section.

Note

Save changes with "WRIT_DBL"

Changes to immediately effective tags are lost on POWER OFF of the CPU, or restart of the technology object.

If changes in the technology data block should also be retained after POWER OFF of the CPU, or restart of the technology object, you must write the changes to the starting value in load memory using the extended instruction "WRIT_DBL".

Note

Using the "READ_DBL" and "WRIT_DBL" data block functions

The "READ_DBL" and "WRIT_DBL" data block functions can only be used on a single tag in conjunction with the tags of the technology object. The "READ_DBL" and "WRIT_DBL" data block functions cannot be applied to data structures of the technology object.

Evaluation of data in the isochronous interrupt OB

If you want to write the technology data block's data isochronously from a Motion Control clock cycle, there is the option of evaluating these data in the isochronous interrupt OB.

For this purpose, assign the same clock cycle source (Page 10251) to the isochronous interrupt OB, as to the MC_Servo [OB91]. By this means, the isochronous interrupt OB is called in each Motion Control clock cycle.

In the isochronous interrupt OB, read the necessary data into global tags. Use these global tags for further programming.

Evaluate StatusWord, ErrorWord and WarningWord

To be able to use individual status and error information from the "StatusWord", "ErrorWord" and "WarningWord" data double words, the evaluation can be performed as described below. For consistent evaluation, you should avoid using bit addressing to access these data double words in the technology data block. Access to an individual bit in the technology data block only lasts as long as the access to the entire data word.

When required, copy the required data double word into a tag of a data structure and query the individual bits of the tag.

A description of the data double words and the allocation of the individual bits can be found in the Appendix:

- StatusWord (Page 10378)
- ErrorWord (Page 10380)
- WarningWord (Page 10382)

Requirements

The technology object has been created.

Procedure

To evaluate the individual bits in the data word "StatusWord", proceed as follows:

1. Crate a global data structure. Name the data structure, e.g. as "Status".
2. Create a double word (DWORD) in the data structure "Status". Name the data structure, e.g. as "Temp".
3. Create 32 Boolean tags in the "Status" data structure. You can obtain a clearer overview by naming Booleanhe Boolean tags as defined for the bits in the technology DB (e.g. name the fifth Boolean tag "HomingDone").
4. Copy the tag <TO>.StatusWord as needed from the technology data block to the double word "Temp" in your data structure.
5. Copy the bits of dword "Temp" to the corresponding Boolean tags by means of bit access.
6. Use the Boolean tags to query the status bits.

Evaluate the "ErrorWord" and "WarningWord" data words as specified in steps 1 to 6.

Example

The following example shows how you can read and save the fifth bit "HomingDone" of the data word "StatusWord":

SCL

```
Status.Temp        :=    <TO>.StatusWord;                    Copy status word
Status.HomingDone  :=    Status.Temp.X5;                     Copy individual
                                                             bits per bit
                                                             access
```

STL

```
L    <TO>.StatusWord                                         Copy status word
T    Status.Temp
U    Status.Temp.X5                                          Copy individual
                                                             bits per bit
                                                             access
=    Status.HomingDone
```

Change restart-relevant data

In order to change restart-relevant data in the technology data block, write to the starting values of the tags in load memory using the extended instruction "WRIT_DBL". In order for the changes to be applied, a restart of the technology object must be performed.

Whether changes to the value of a tag are relevant for the restart can be found in the description of the tags in the Appendix (Page 10365).

Requirements

The technology object has been created.

Procedure

To change restart-relevant data, proceed as follows:

1. Create a data block and fill it with the restart-relevant values, that you want to change in the technology data block. In doing so, the data types must match the tags to be changed.
2. Write the values from your data block to the starting values of the tags of the technology data block in load memory, using the extended instruction "WRIT_DBL".
   If restart-relevant data were changed, this will be indicated in the <TO>.StatusWord.OnlineStartValuesChanged tag of the technology object.
3. Perform a restart of the technology object using the Motion Control instruction "MC_Reset" with parameter "Restart" = TRUE.

After the restart of the technology object, the new value is transferred into the technology data block in work memory, and is effective.

Motion Control instructions

Motion Control instruction parameters

Description

When creating your user program, take account of the following explanations of the Motion Control instruction parameters.

Reference to the technology object

The specification of the technology object to the Motion Control instruction occurs as follows:

- **"Axis" parameter**
  In the "Axis" input parameter of a Motion Control instruction, a reference to the technology object is specified, which the corresponding job should use.

Job start and receipt of the input parameters of a Motion Control instruction

When jobs are started, and when changed parameter values are received, a distinction is made between the following Motion Control instructions:

- **Motion Control instructions with "Execute" parameter**
  With a rising edge at the "Execute" parameter, the job is started and the values pending in the input parameters are applied.
  Subsequently changed parameter values are not applied until the next job start.
  Resetting the "Execute" parameter does not end the job, but does have an effect on the indication duration of the job status. As long as "Execute" is set to TRUE, the output parameters will be updated. If "Execute" is reset before the completion of a job, the parameters "Done", "Error" and "CommandAborted" are correspondingly set for only one call cycle.
- **Motion Control instructions with "Enable" parameter**
  The job is started by setting the "Enable" parameter.
  As long as "Enable" = TRUE. the job remains enabled and changed parameter values are immediately applied.
  The job is ended by resetting the "Enable" parameter.
  The input parameters "JogForward" and "JogBackward" of the "MC_MoveJog" Motion Control instruction correspond in their behavior to the "Enable" parameter.

Job status

The following output parameters indicate the status of the job processing:

- **Motion Control instructions with "Done" parameter**
  The normal completion of a job is indicated with parameter "Done" = TRUE.
- **Motion Control instructions without "Done" parameter**
  The achievement of the job target is indicated by means of other parameters (e.g. "Status", "InVelocity"). For more information, refer to the section " Tracking running jobs (Page 10326).
- **"Busy" parameter**
  As long as a job is being processed, the parameter "Busy" shows the value TRUE. If a job was ended or aborted, "Busy" shows the value FALSE.
- **"CommandAborted" parameter**
  If a job was aborted by another job, CommandAborted shows the value "TRUE".
- **"Error" parameter**
  If an error occurs in the Motion Control instruction, the "Error" parameter shows the value TRUE. The ErrorID parameter shows the cause of the error.

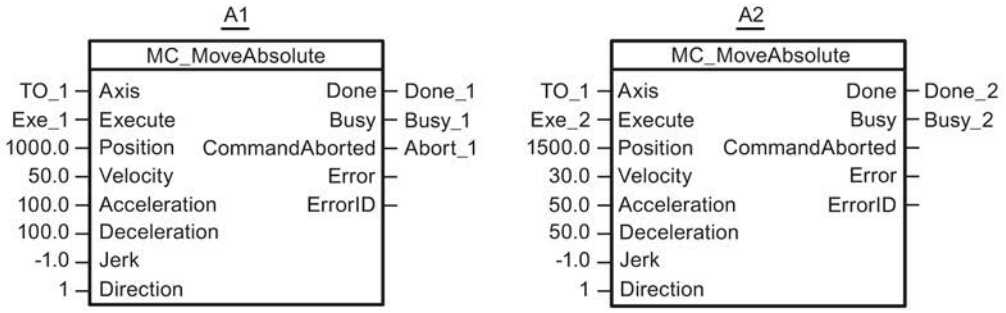
As long as the "Execute" or "Enable" parameter is set to TRUE, the output parameters will be updated. Otherwise the parameters "Done", "Error" and CommandAborted" are correspondingly only set for one cycle.

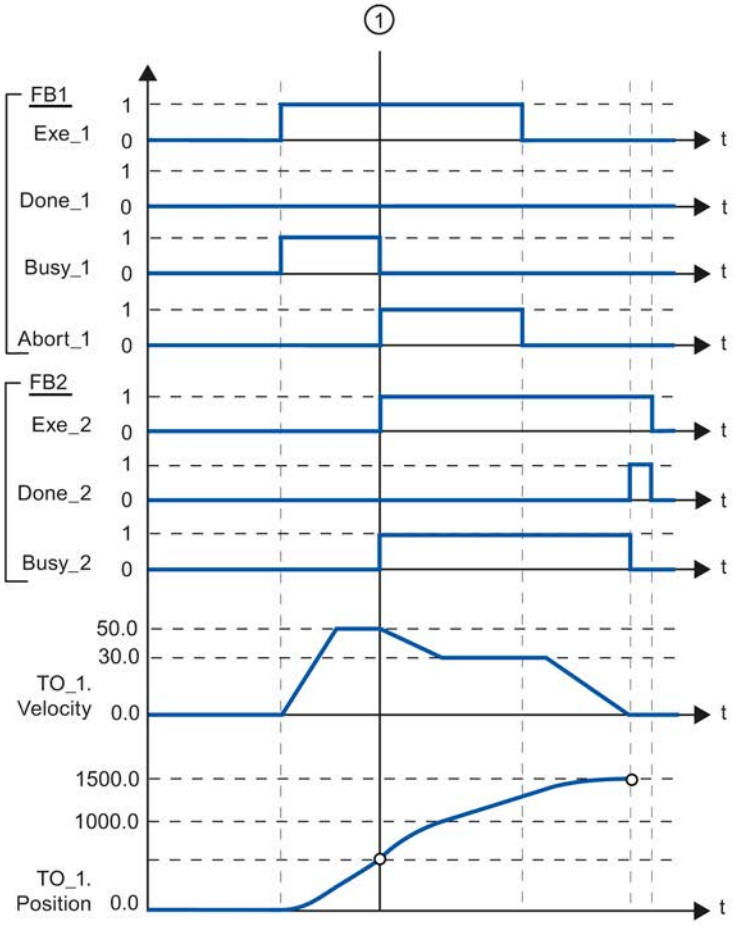
Termination of active jobs

A running Motion Control job is terminated by the initiation of a new Motion Control job. In the process, the current dynamic setpoints (acceleration, deceleration, jerk, velocity) are set to the values of the overriding job.

Example of parameter behavior

The behavior of the parameters of Motion Control instructions is shown by way of the example of two "MC_MoveAbsolute" jobs.





Using "Exe_1", an "MC_MoveAbsolute" job (A1) with target position 1000.0 is initiated. "Busy_1" is set to TRUE. The axis is accelerated to the specified velocity and moved to the target position (see TO_1.Velocity and TO_1.Position). Before the target position is reached, the job is overridden at time ① by another "MC_MoveAbsolute" job (A2). The abort is signaled via "Abort_1", and "Busy_1" is set to FALSE. The axis is braked to the velocity specified in A2,

and moved to the new target position 1500.0. When the axis reaches the target position, this is signaled via "Done_2".

Add Motion Control instructions

You add Motion Control instructions to a program block in the same way as other instructions. You control all available functions of the technology object using the Motion Control instructions.

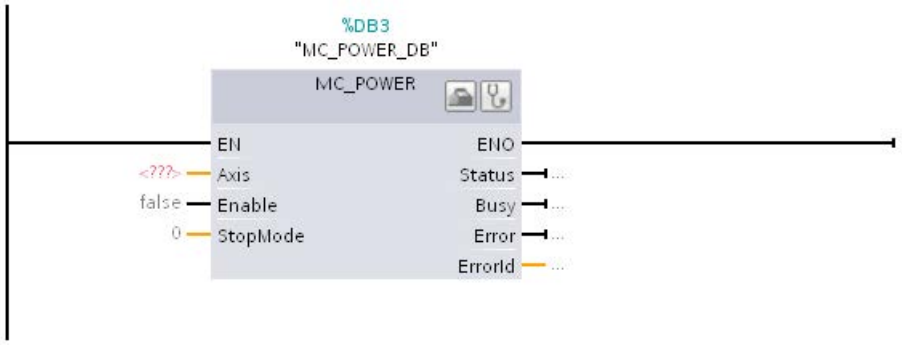
Requirements

The technology object was created.

Procedure

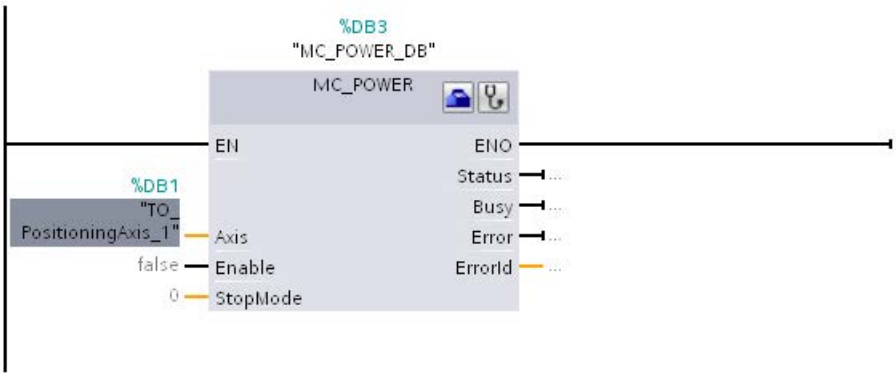
To add Motion Control instructions in your user program, proceed as follows:

1. Double click your program block in the Project Navigator (the program block must be called in the cyclical program).
   The program block is opened in the programming editor, and the available instructions are displayed.
2. In the "Instructions" task card, open the "Technology > Motion Control > S7-1500 Motion Control" folder.
3. Drag and drop the Motion Control instruction, e.g. "MC_Power", into the desired segment of the program block.
   The "Call options" dialog opens.
4. In the dialog, specify a name and a number for the instance data block of the Motion Control instruction.
5. Click "OK".
   The Motion Control instruction "MC_Power" is added to the segment.



The instance data block is automatically created under "Program Blocks > System Blocks > Program Resources".

6. Input parameters without a default value (e.g. "Axis"), must be supplied. Mark the technology object in the Project Navigator, and drag and drop it to <...> in the "Axis" parameter.



After specifying the technology object in the "Axis" parameter, the following buttons are available to you:
To open the configuration of the technology object, click on the toolbox icon.
To open the diagnostics of the technology object, click on the stethoscope icon.

7. Add additional Motion Control instructions in accordance with steps 3 through 6.

See also

Tracking active jobs (Page 10326)

Tags of the positioning axis technology object (Page 10365)

Starting Motion Control jobs

Description

Motion Control jobs are started by setting the "Execute" or "Enable" parameter of the Motion Control instruction. The call of the Motion Control instructions for a technology object should occur in an execution level.

When executing Motion Control jobs, you should also take note of the status of the technology object.

Starting Motion Control jobs should be performed in the following steps:

1. Query the status of the technology object.
2. Initiate new job for the technology object.
3. Check job status.

These steps are explained using the example of a job for absolute positioning.

1. Query the status of the technology object

Make sure that the technology object is in the appropriate status to perform the desired job:

- **Has the technology object been released?**
  To execute motion jobs, the technology object must be enabled.
  Enabling is performed using the Motion Control instruction "MC_Power".
  The "MC_Power.Status" parameter must show the value TRUE.
- **Is a technology alarm pending?**
  To perform motion jobs, no technology alarms may be pending. After resolving the error, acknowledge any possibly pending alarms using the Motion Control instruction "MC_Reset".
  A list of the technology alarms and alarm responses can be found in the appendix, Technology Alarms (Page 10403).
- **Has the technology object been homed?**
  In order to perform a job for absolute positioning, the positioning axis technology object must be homed. The referencing occurs via the Motion Control instruction "MC_Home".
  The "<TO>.StatusWord.HomingDone " tag of the technology object must show the value TRUE .

2. Initiate new job for the technology object

Initiate a new job.

In the "Position" parameter of the "MC_MoveAbsolute" instruction, specify the position to which the axis should be moved. Start the job with a positive edge at the "Execute" parameter.

3. Check job status

Parameter "Done" of the Motion Control instruction indicates successful completion of a job (target reached, in this case).

If an error is detected, the "Error" parameter of the Motion Control instruction is set to TRUE, and the job is rejected.

You can program an error handling routine for the Motion Control job. For this purpose, evaluate the error indicated in the "Error" parameter. The cause of the error is indicated in the ErrorID parameter. After resolving the cause of the error, restart the job.

Output of the "Error" = TRUE and "ErrorID" = 16#8001 status information during job processing indicates that a technology alarm was triggered.

A list of the ErrorIDs can be found in the appendix, Error Detection (Page 10405).

Additional information

An option for the evaluation of the individual status bits, error bits, and warning bits can be found in the Evaluating StatusWord, ErrorWord and WarningWord (Page 10317) section.

Tracking active jobs

Introduction

The current status of the job processing is made available via the output parameters of the Motion Control instruction. These parameters are updated with each call of the Motion Control instruction.

When tracking jobs, a distinction is made between three groups:

- Motion Control instructions with "Done" parameter (Page 10326)
- Motion Control instruction "MC_MoveVelocity" (Page 10329)
- Motion Control instruction "MC_MoveJog" (Page 10332)

Motion Control instructions with "Done" parameter

Description

Jobs for Motion Control instructions with the "Done" parameter are started with a positive edge at the "Execute" parameter. If the job was completed without errors and without interruption by another job (e.g. "MC_ MoveAbsolute": target position reached), then the "Done" parameter shows the value TRUE.

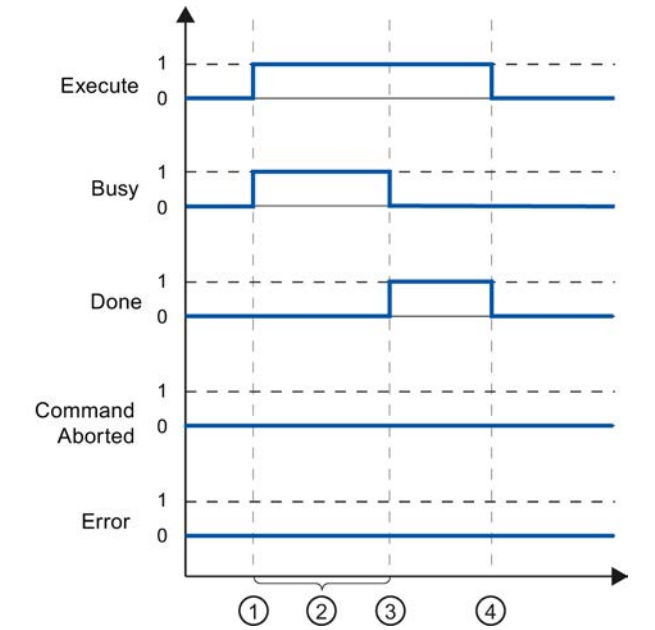
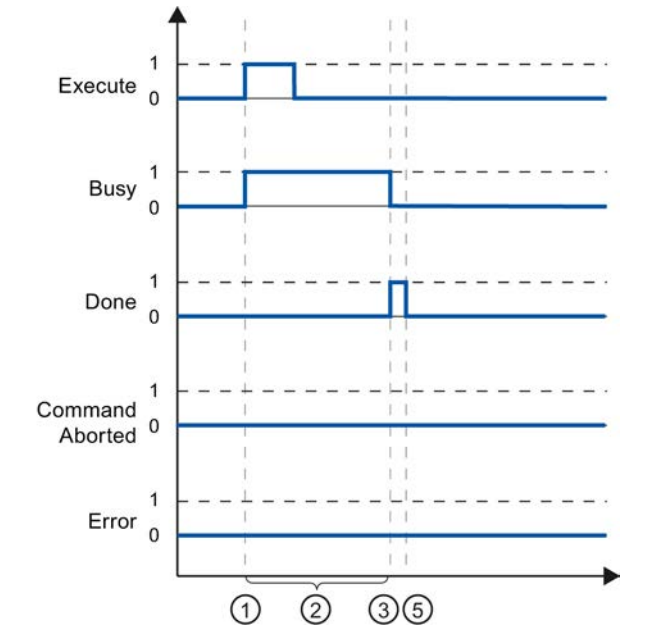
The following Motion Control instructions have a "Done" parameter:

- MC_Home
- MC_MoveRelative
- MC_MoveAbsolute
- MC_Halt
- MC_Reset

The behavior of the parameters is presented by way of example for various situations:

Complete execution of job

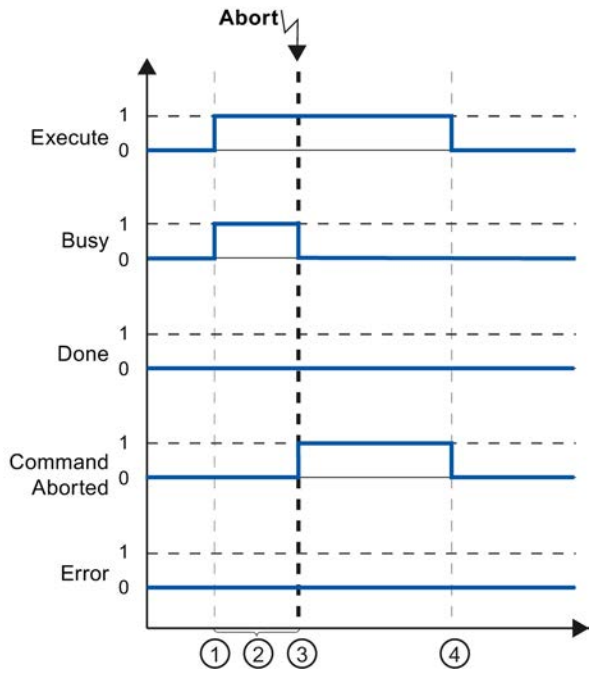
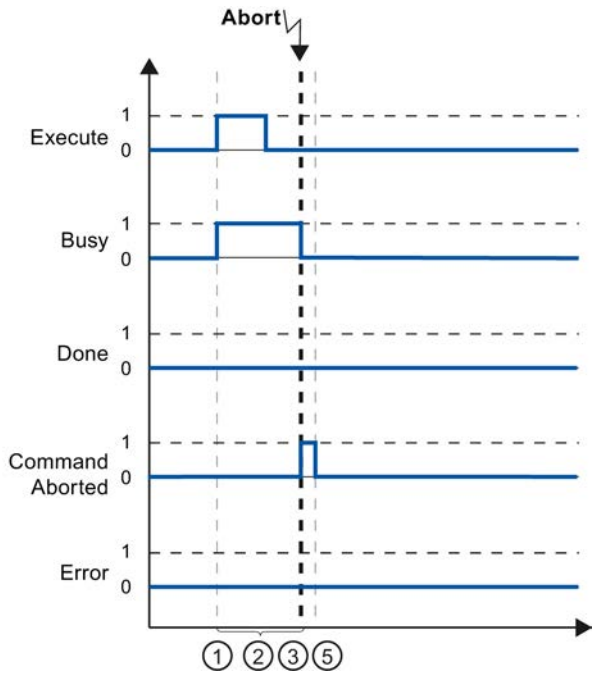
If the Motion Control job has been fully processed to the point of completion, then this is indicated in the "Done" parameter with the value TRUE. The signal status of the "Execute" parameter influences the indication duration in the "Done" parameter:

You set "Execute" **after completion** of the job to FALSE.	You set "Execute" **during processing** of the job to FALSE.
	

①	The job is started with a positive edge at the "Execute" parameter. Depending on the program, "Execute" can be reset to the FALSE value while the job is still busy, or retain the TRUE value until the job has been completed.
②	While the job is active, the "Busy" parameter shows the value TRUE.
③	When the job is complete (for example, for Motion Control instruction "MC_MoveAbsolute": target position reached) the "Busy" parameter changes to FALSE, and the "Done" parameter to TRUE.
④	As long as the "Execute" parameter retains the value TRUE after completion of the job, the "Done" parameter also retains the value TRUE.
⑤	If the "Execute" parameter has been set to FALSE before the job is complete, then the "Done" parameter only shows the value TRUE for one execution cycle.

Job termination

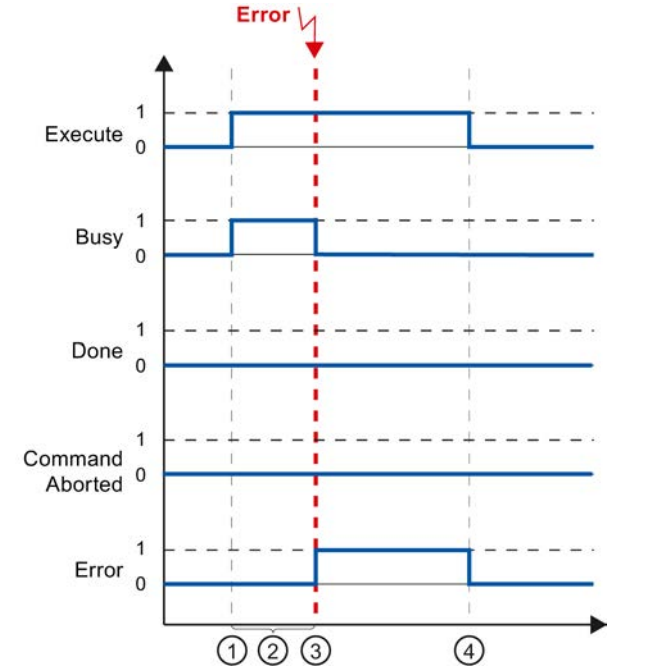
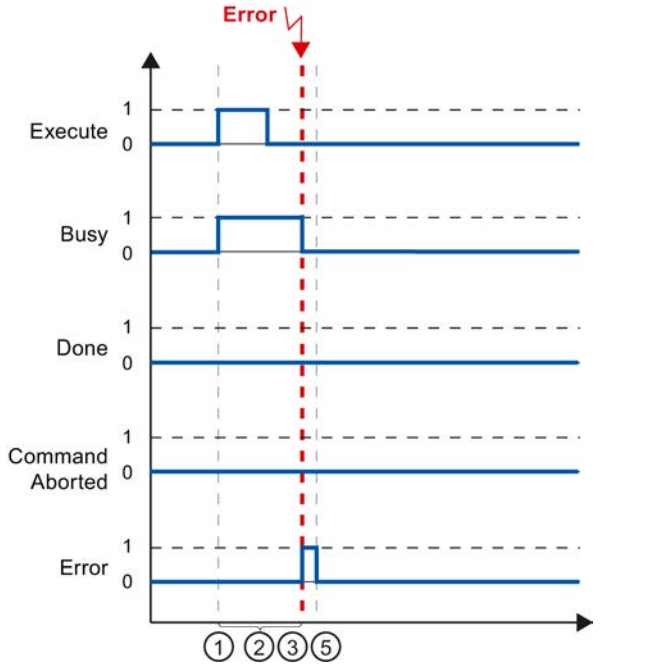
If the Motion Control job is aborted during processing by another job, then this is indicated in the "CommandAborted" parameter with the value TRUE. The signal status of the "Execute" parameter influences the indication duration in the "CommandAborted" parameter:

You set "Execute" **after** the job is aborted to FALSE.	You set "Execute" **before** the job is aborted to FALSE.
	

①	The job is started with a positive edge at the "Execute" parameter. Depending on the program, "Execute" can be reset to the FALSE value while the job is still busy, or retain the TRUE value until the job has been completed.
②	While the job is active, the parameter "Busy" shows the value TRUE.
③	During job execution, the job is aborted by another Motion Control job. When the job is aborted, the "Busy" parameter changes to "FALSE" and "CommandAborted" changes to TRUE.
④	As long as the "Execute" parameter retains the value TRUE after completion of the job, the "CommandAborted" parameter also retains the value TRUE.
⑤	If the "Execute" parameter has been set to FALSE before the job is aborted, then the "CommandAborted" parameter only shows the value TRUE for one execution cycle.

Error during job execution

If an error occurs during execution of the Motion Control job, then this is indicated in the "Error" parameter with the value TRUE. The signal status of the "Execute" parameter influences the indication duration in the "Error" parameter:

You set "Execute" **after** the occurrence of the error to FALSE.	You set "Execute" **before** the occurrence of the error to FALSE.
	

①	The job is started with a positive edge at the "Execute" parameter. Depending on the program, "Execute" can be reset to the FALSE value while the job is still busy, or retain the TRUE value until the job has been completed.
②	While the job is active, the "Busy" parameter shows the value TRUE.
③	An error occurs during the execution of the job. When the error occurs, the "Busy" parameter changes to FALSE, and the "Error" parameter to TRUE.
④	As long as the "Execute" parameter retains the value TRUE after the occurrence of the error, the "Error" parameter also retains the value TRUE.
⑤	If the "Execute" parameter has been set to FALSE before the occurrence of the error, then the "Error" parameter only shows the value TRUE for one execution cycle.

Motion Control instruction "MC_MoveVelocity"

Description

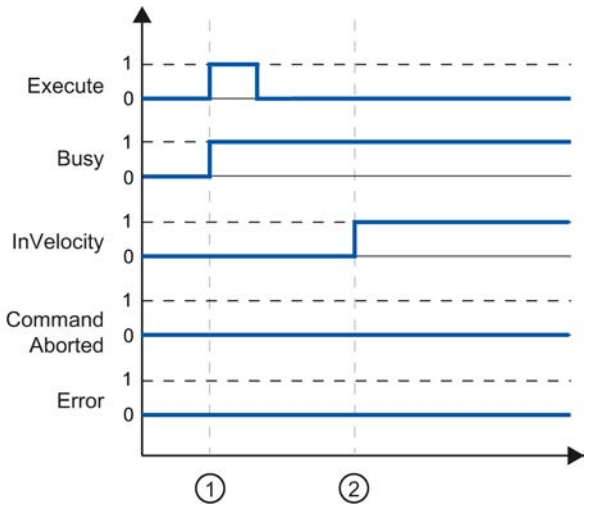
An "MC_MoveVelocity" job is started with a positive edge at the "Execute" parameter. The job objective is fulfilled when the assigned velocity is reached and the axis travels at constant velocity. When the assigned velocity is reached and maintained, this is indicated in the "InVelocity" parameter with the value TRUE.

The motion of the axis can, for example, be stopped with an "MC_Halt" job.

The behavior of the parameters is presented below for various example situations:

The assigned velocity is reached and maintained

The achievement of the assigned velocity is indicated in the "InVelocity" parameter with the value TRUE. The "Execute" has no effect on the indication duration in the "InVelocity" parameter.

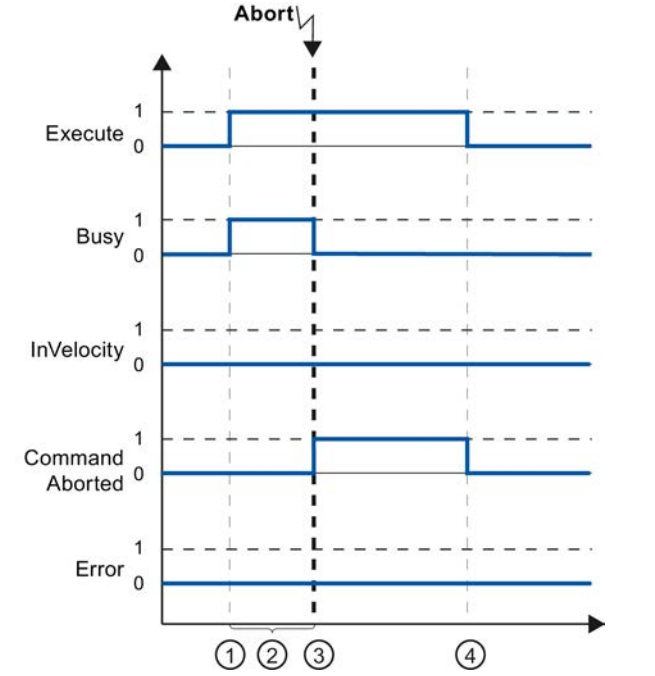
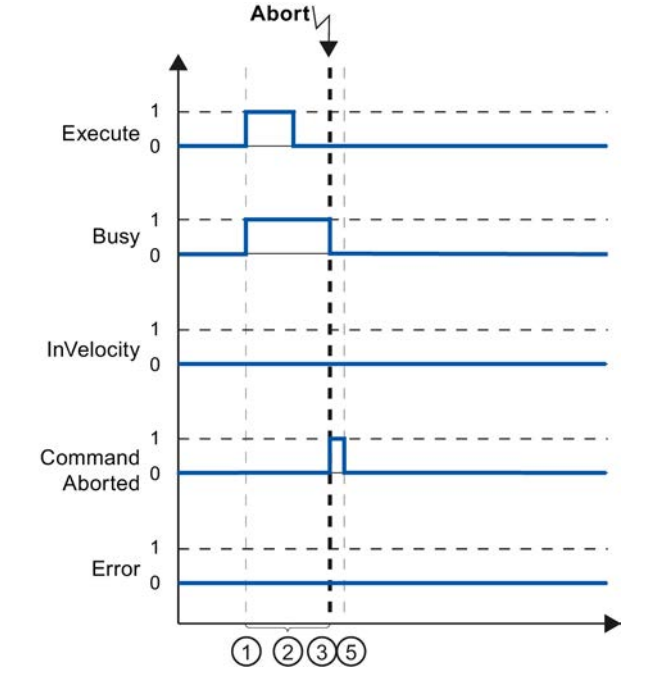


①	The job is started with a positive edge at the "Execute" parameter. Depending on the program, "Execute" can be reset to the FALSE value before or after the parameterized velocity has been reached. While the job is active, the parameter "Busy" shows the value TRUE.
②	When the assigned velocity is reached, the "InVelocity" parameter changes to TRUE. The "Busy" and "InVelocity" parameters retain the TRUE value until a different Motion Control job overrides the "MC_MoveVelocity" job.

The job is aborted prior to reaching the assigned velocity

If the Motion Control job is aborted by another job before reaching the assigned velocity, then this is indicated in the "CommandAborted" parameter with the value TRUE. The signal status

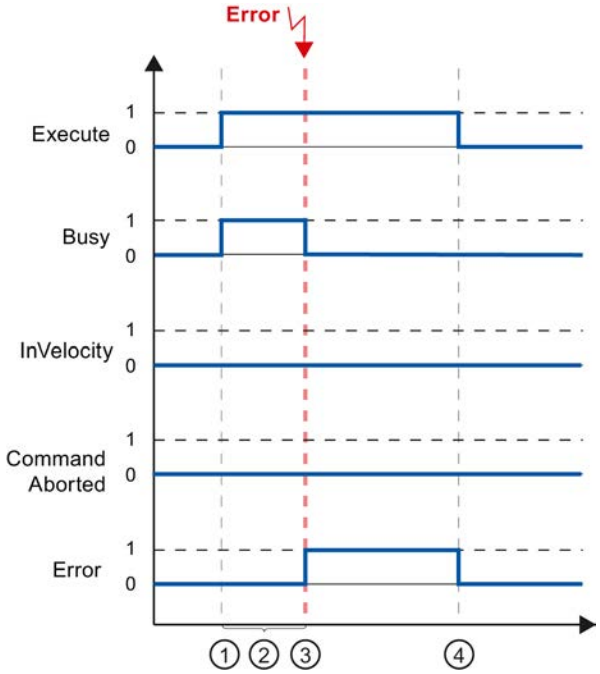
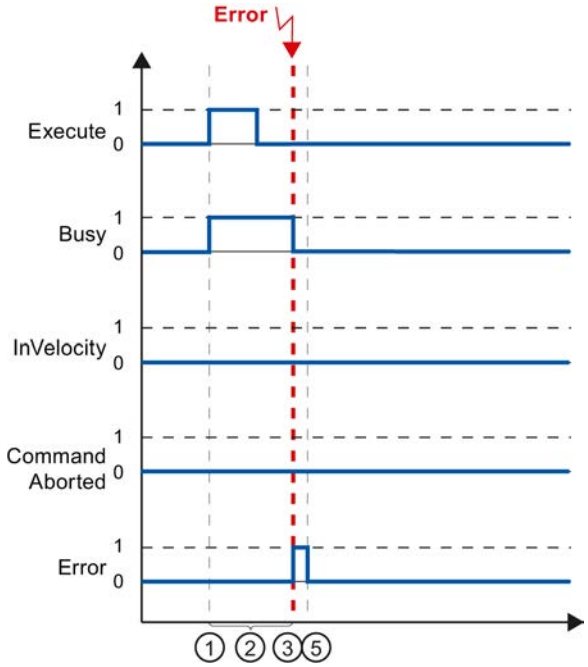
of the "Execute" parameter influences the indication duration in the "CommandAborted" parameter.

You set "Execute" **after** the job is aborted to FALSE.	You set "Execute" **before** the job is aborted to FALSE.
	

①	The job is started with a positive edge at the "Execute" parameter. Depending on the program, "Execute" can be reset to the FALSE value while the job is still busy, or retain the TRUE value until the job is aborted.
②	While the job is active, the "Busy" parameter shows the value TRUE.
③	During job execution, the job is aborted by another Motion Control job. When the job is aborted, the "Busy" parameter changes to "FALSE" and "CommandAborted" changes to TRUE.
④	As long as the "Execute" parameter retains the value TRUE after completion of the job, the "CommandAborted" parameter also retains the value TRUE.
⑤	If the "Execute" parameter has been set to FALSE before the job is aborted, then the "CommandAborted" parameter only shows the value TRUE for one execution cycle.

An error has occurred prior to reaching the assigned velocity

If an error occurs during execution of the Motion Control job before the assigned velocity has been reached, this is indicated in the "Error" parameter with the value TRUE. The signal status of the "Execute" parameter influences the indication duration in the "Error" parameter.

You set "Execute" **after** the occurrence of the error to FALSE.	You set "Execute" **before** the occurrence of the error to FALSE.
	

①	The job is started with a positive edge at the "Execute" parameter. Depending on the program, "Execute" can be reset to the FALSE value while the job is still busy, or retain the TRUE value until the error has occurred.
②	While the job is active, the "Busy" parameter shows the value TRUE.
③	An error occurs during the execution of the job. When the error occurs, the "Busy" parameter changes to FALSE and the "Error" parameter to TRUE.
④	As long as the "Execute" parameter retains the value TRUE after completion of the job, the "Error" parameter also retains the value TRUE.
⑤	If the "Execute" parameter has been set to FALSE before the job is aborted, then the "Error" parameter only shows the value TRUE for one execution cycle.

Motion Control instruction "MC_MoveJog"

Description

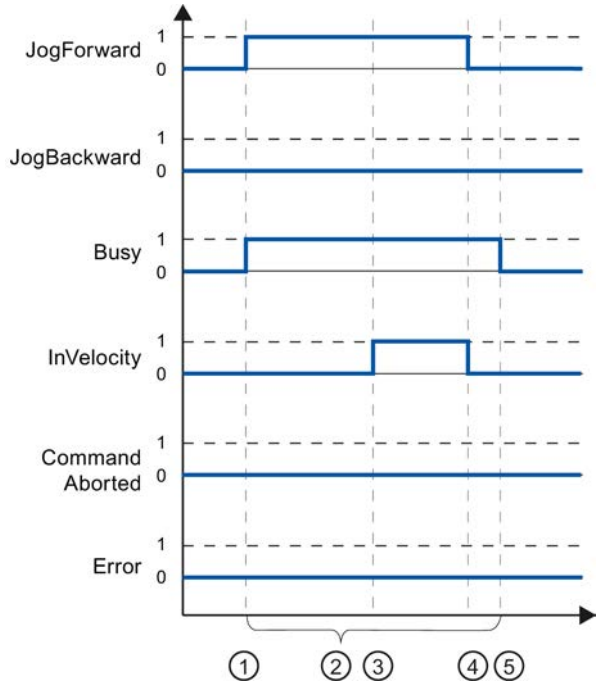
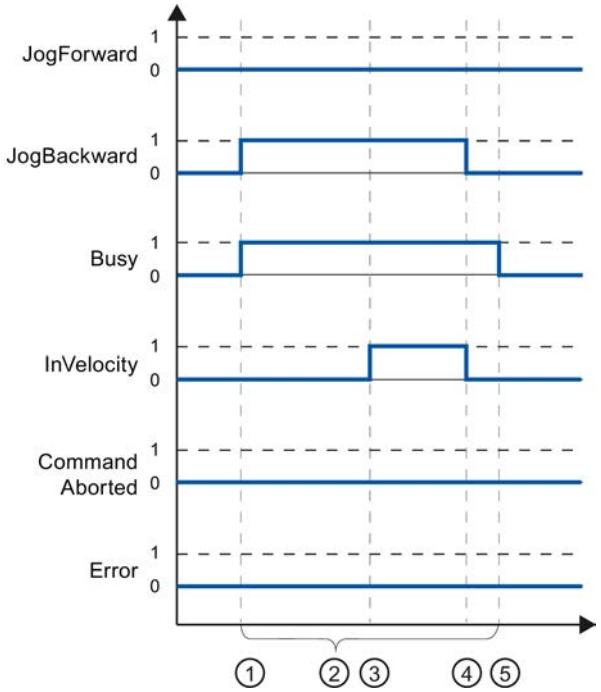
An "MC_MoveJog" job is started by setting the "JogForward" or "JogBackward" parameter. The job objective is fulfilled when the assigned velocity is reached and the axis travels at constant velocity. When the assigned velocity is reached and maintained, this is indicated in the "InVelocity" parameter with the value TRUE.

The job is complete when the "JogForward" or "JogBackward" parameter has been set to the value FALSE and the axis has come to a standstill.

The behavior of the parameters is presented below for various example situations:

The assigned velocity is reached and maintained

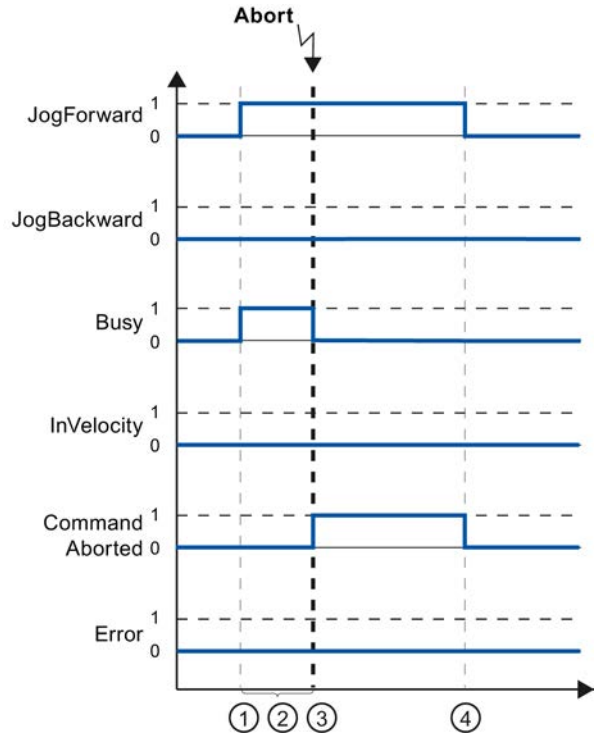
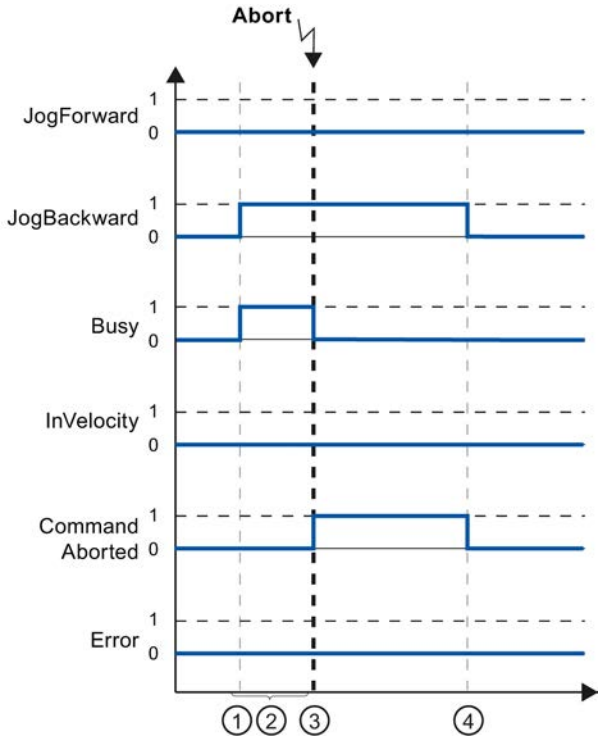
If the Motion Control job has been performed up to the point of reaching the assigned velocity, then this is indicated in the "InVelocity" parameter with the value TRUE.

Jog mode is controlled by the "JogForward" parameter.	Jog mode is controlled by the "JogBackward" parameter.
	

①	The job is started by setting the "JogForward" or "JogBackward" parameter.
②	While the job is active, the "Busy" parameter shows the value TRUE.
③	When the assigned velocity is reached, the "InVelocity" parameter changes to TRUE.
④	When the "JogForward" or "JogBackward" parameter is reset, the motion of the axis ends. The axis decelerates. The "InVelocity" parameter changes to FALSE.
⑤	If the axis has come to a standstill, then the Motion Control job is complete and the "Busy" parameter changes to FALSE.

The job is aborted during execution

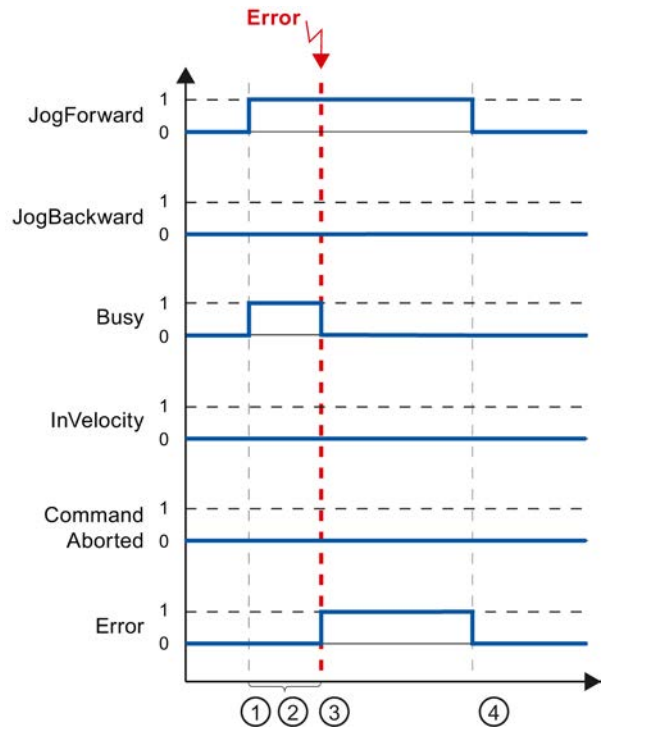
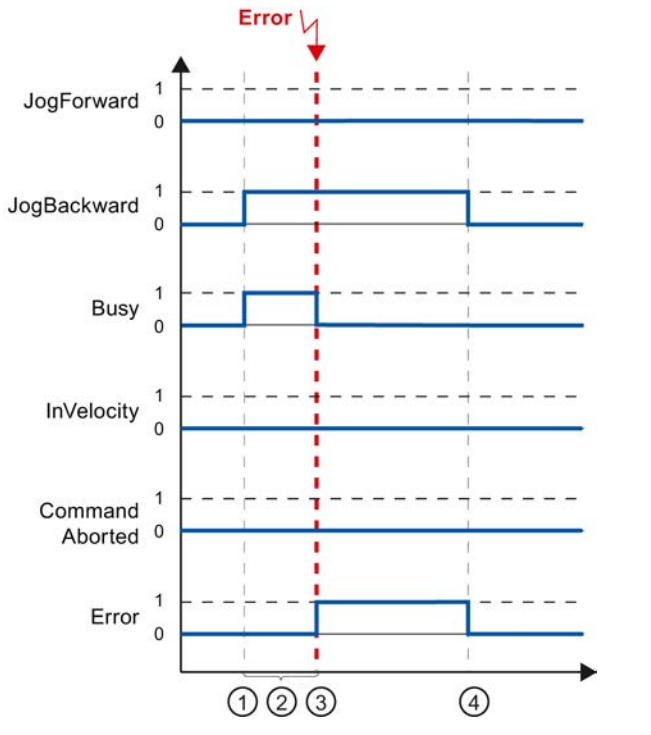
If the Motion Control job is aborted during processing by another job, then this is indicated in the "CommandAborted" parameter with the value TRUE. The behavior of the "CommandAborted" parameter is independent of reaching the assigned velocity.

Jog mode is controlled by the "JogForward" parameter.	Jog mode is controlled by the "JogBackward" parameter.
	

①	The job is started by setting the "JogForward" or "JogBackward" parameter.
②	While the job is processing, the "Busy" parameter shows the value TRUE.
③	During job execution, the job is aborted by another Motion Control job. When the job is aborted, the "Busy" parameter changes to "FALSE" and "CommandAborted" changes to TRUE.
④	When the "JogForward" or "JogBackward" parameter is reset, the "CommandAborted" parameter likewise changes to FALSE.

An error occurs during the execution of the job

If an error occurs during execution of the Motion Control job, this is indicated in the "Error" parameter with the value TRUE. The behavior of the "Error" parameter is independent of reaching the assigned velocity.

Jog mode is controlled by the "JogForward" parameter.	Jog mode is controlled by the "JogBackward" parameter.
	

①	The job is started by setting the "JogForward" or "JogBackward" parameter.
②	While the job is active, the "Busy" parameter shows the value TRUE.
③	An error occurs during the execution of the job. When the error occurs, the "Busy" parameter changes to "FALSE", and "Error" changes to TRUE.
④	When the "JogForward" or "JogBackward" is reset to the value FALSE, the "Error" parameter likewise changes to FALSE.

Ending Motion Control jobs

When ending a job, a distinction is made between error-free completion of the job, and the termination of a motion.

Completion of the job

The completion of a Motion Control job is indicated as described in the Tracking running jobs (Page 10326) section.

Termination of a motion

If a motion must be aborted, you can perform the following measures:

- "Execute MC_Halt (Page 4912)"
  To abort a motion and stop the axis, you can use the "MC_Halt" instruction.
- "Disable MC_Power (Page 4885)"
  In an emergency, you can stop the axis using an emergency stop ramp. For this purpose, set the "Enable" parameter of the "MC_Power" instruction to FALSE. The axis is decelerated according to the selected "StopMode" and all jobs at the technology object are aborted.

Restart of technology objects

Description

After the CPU is switched on, or after technology objects are downloaded into the CPU, the system automatically initializes the technology objects with the start values from the technology data blocks. If restart-relevant changes are detected during a reload into the CPU, a restart of the technology object is automatically performed.

If restart-relevant data have been changed in RUN mode by the user program, then the technology object must be reinitialized by the user in order for the changes to be used.

If changes in the technology data block should also be retained after the restart of the technology object, then you must write the changes to the start value in load memory using the extended instruction "WRIT_DBL".

Restart necessary

A necessary TO restart is indicated at "Technology object > Diagnostics > Status and error bits > Axis status or Encoder status > Online start value changed", as well as in the tag of technology object <TO>.StatusWord.OnlineStartValuesChanged.

Restarting a technology object

A restart of the technology object is triggered by the user by means of the "MC_Reset" Motion Control instruction, with parameter "Restart" = TRUE.

During a restart, all configuration data of the technology object are loaded from load memory into work memory. In the process, the actual values in the technology data block are overwritten.

Note the following during a restart of the technology object:

- A restart resets the "Referenced" status of a technology object with incremental actual values (<TO>.StatusWord.HominDone).
- While a restart is being performed, the technology object cannot perform any jobs. An active restart will be indicated under "Technology object > Diagnostics > Status and error bits > Axis status or Encoder status > Restart active", and in the <TO>.StatusWord.RestartActive tag of the technology object.

- Motion Control jobs are rejected during a restart with the "Error" = TRUE and "ErrorID" = 16#800D parameters (job not executable, because a restart is active).
- While a restart is being executed, you cannot access the technology data block.

See also

Change restart-relevant data (Page 10318)

Downloading to CPU

Description

When downloading to the CPU S7-1500, it is always verified that the project files are consistent online and offline after the download.

The data of the technology objects are saved in technology data blocks. The conditions for downloading blocks thus apply when loading new or modified technology objects.

You can load the following object groups into the CPU:

Context menu command "Download to device"	Description
Hardware and software	Download all new and modified blocks, and a new or modified hardware configuration
Hardware configuration	Download a new or modified hardware configuration
Software (changes only)	Download all new and modified blocks.
Software (all blocks)	Download all blocks.

Load in RUN mode

When loading in the CPU's RUN mode, it is checked whether a load without restart of the technology objects is possible.

If restart-relevant configuration values were changed, then a restart of the technology object is automatically performed after the load into the CPU.

Loading a technology object is only possible if the technology object is disabled.

You **cannot** download the following changes to the CPU in RUN mode:

- Changes to the MC-Servo clock speeds
- Changes to the hardware interface of the technology object in "Technology object > Configuration > Hardware interface"

Commissioning

Introduction

The following guidelines describe the steps that you should note when commissioning the Motion Control-specific components of your equipment.

The commissioning of other components of your automation system depends on the particular equipment configuration. Commissioning (not Motion Control) is described in the Automation System S7-1500 (http://support.automation.siemens.com/WW/view/en/59191792) system manual.

Commissioning guidelines

These guidelines serve as recommendations for commissioning equipment with Motion Control. The procedure is described using the example of a positioning axis technology object.

Requirements

- The configuration of the following components is complete:
  - CPU
  - BUS communication
  - Drives
  - Technology objects
- The user program has been created.
- The wiring of the CPU and of the associated I/O is complete.
- The commissioning and optimization of the drive is complete.

Procedure

Proceed as follows to commission the Motion Control-specific components of your equipment:

Step	Action to be performed	Supported by TIA Portal
Turn on CPU	Turn on the power supply and the CPU.	-
"Disable" position controller	Set the gain (Kv factor) of the position control loop to zero. (This setting avoids unwanted drive movements that may be caused by incorrect parameterization of the position control loop.)	"Technology object > Configuration > Extended parameters > Control loop"
Load project into the CPU	Bring the CPU to the STOP mode. Download your project to the CPU (load hardware and software).	• "Toolbar > Stop CPU" • "Toolbar > Download to device"
Create online connection to the CPU	Select the "Receive messages" check box under "Online & Diagnostics > Online Access". Configure the interface of the TIA Portal, and create an online connection with the CPU.	• Device configuration • "Online & Diagnostics > Online Access"

Step	Action to be performed	Supported by TIA Portal
Disable Motion Control specific user program	In order to avoid conflicts with the axis control panel, lock the enabling of technology objects in your user program (MC_Power.Enable = FALSE).	• PLC programming • Motion Control instructions
Evaluating pending messages	Evaluate the message display in the inspector window. Resolve the causes of pending technology alarms. Acknowledge the technology alarms (Page 10350).	"Inspector window > Diagnostics > Message display"
Check hardware limit switches	Click the hardware limit switches. Check for correct message display (technology alarm 531). Acknowledge the technology alarm.	"Inspector window > Diagnostics > Message display"
Check the connection and configuration of the drive (setpoint)	Bring the CPU into the RUN mode. Open the Axis control panel (Page 10340) and take over control. Perform the following steps: • Enable the technology object. ⇒ The drive must turn itself on, and where applicable release the brake. The position is held. • Move the axis in jog mode at low velocity in the positive direction. ⇒ The drive must move. The actual position value must increase (positive direction). • Disable the technology object. ⇒ The drive must turn itself off, and where applicable apply the brake.	"Technology object > Commissioning > Axis control panel"
Check the connection and configuration of the encoder (actual value)	Check the scaling of the actual values (rotation direction, distance evaluation, and resolution of the encoder) ⇒ The change in the actual mechanical position must match the change in the actual values.	• "Technology object > Diagnostics > PROFIdrive frame" • "Technology object > Commissioning > Axis control panel"
Checking the reference speed	Traverse the axis in jog mode at low velocity in the positive direction. ⇒ The displayed current velocity must match the velocity setpoint. If the displayed current velocity deviates significantly from the velocity setpoint, adjust the reference speed.	• "Technology object > Hardware interface > Data exchange" • "Technology object > Commissioning > Axis control panel"
Optimize position controller	Use the Optimization commissioning function to optimize the gain (Kv) of the position control loop. For this purpose, adapt following error limits as needed.	"Technology object > Commissioning > Optimization"
Transfer the gain Kv to the project.	Enter the gain Kv that you determined by means of the optimization function in your configuration data. Load your project into the CPU.	"Technology object > Configuration > Extended parameters > Control loop"
Enable Motion Control specific user program	Unlock the enabling technology objects lock in your user program (MC_Power.Enable = TRUE).	• PLC programming • Motion Control instructions
Check the functioning of the user program	Check the programmed functions of your user program.	• Watch and force tables • Online and diagnostic functions
End of commissioning for a positioning axis technology object	To commission additional technology objects, perform the corresponding steps again.	See above.

See also

Function and structure of the optimization (Page 10344)

Axis control panel

Function and structure of the axis control panel

Description

You traverse individual axes with the axis control panel.

A user program is necessary for the operation of the axis control panel. With the axis control panel, you take over master control for a technology object and control the movements of the axis.

WARNING

Uncontrolled axis motions

During operation with the axis control panel, the axis can execute uncontrolled motions (e.g. due to erroneous configuration of the drive, or of the technology object).

Therefore, perform the following protective measures before operation with the axis control panel:

- Ensure that the EMERGENCY OFF switch is within the reach of the operator.
- Enable the hardware limit switches.
- Enable the software limit switches.
- Ensure that following error monitoring is enabled.

The axis control panel of the speed axis and positioning axis can be found in the Project Navigator under "Technology object > Commissioning".

The axis control panel is divided into the following areas:

- Master control
- Axis
- Operating mode
- Modify
- Axis status
- Current values

The following table lists the elements of the axis control panel:

Area	Element	Description
Master control		In the "Master control" area, you take over master control of the technology object, or return it to your user program.
	"Activate" button	With the "Activate" button, you set up an online connection to the CPU and take over master control for the selected technology object. • To take over master control, the technology object must be disabled in the user program. • If the online connection to the CPU is lost during operation with the axis control panel, then after the sign of life monitoring has elapsed, the axis will be stopped with maximum deceleration. In this case, an error message is displayed ("ErrorID" = 16#8013) and the master control is passed back to the user program. If the master control of the axis control panel is lost repeatedly without a direct error message, the online connection to the CPU may be impaired because the communication load is too high. In this case, the alarm "Error during commissioning. Sign-of-life failure between controller and TIA Portal" is shown in the archive of the alarm display. To eliminate this error, adapt the sign of life monitoring in the tag of the technology object <TO>.ControlPanel.Input.TimeOut. The default value of the tag is 2000 ms. You can change the value of the tag directly. The change takes effect the next time the master control is activated. • Until master control is returned, the user program has no influence on the functions of the technology object. Motion Control jobs from the user program to the technology object are rejected with the error ("ErrorID" = 16#8012: axis control panel enabled). • When master control is taken over, the configuration of the technology object is adopted. Changes to the configuration of the technology object do not become effective until master control has been returned. For this reason, make any necessary changes before master control is taken over. • If master control has been taken over for the technology object, the axis control panel is blocked for access by another installation of the TIA Portal (Team Engineering as of CPU V1.5).
	"Deactivate" button	With the "Deactivate" button, you return master control to your user program.
Axis		In the "Axis" area, enable or disable the technology object.
	"Enable" button	With the "Enable" button, you enable the selected technology object
	"Disable" button	With the "Disable" button, you disable the selected technology object
Operating mode		Select the required mode in the "Operating mode" drop-down list.
Modify		The "Control" area displays the parameters for traversing with the axis control panel according to the selected operating mode.
	Position	Position to which the axis is homed. (Homing and set homing point modes only)
	Distance	Distance the axis is traversed. (Relative positioning mode only)
	Target position	Position to which the axis is traversed. (Absolute positioning mode only)
	Velocity/ Velocity setpoint	Velocity or speed at which the axis is traversed. Default: 10% of the default value (Velocity specification/Default speed, Jog and Positioning modes only)
	Acceleration	Acceleration with which the axis is traversed. Default: 10% of the default value

Area	Element	Description
	Deceleration	Deceleration with which the axis is traversed. Default: 10% of the default value
	Jerk	Jerk with which the axis is traversed. Default: 100% of default value
	"Start" button	With the "Start" button, you start a job according to the selected operating mode.
	"Forward" button	With the "Forward" button, you start a motion in the positive direction according to the selected operating mode.
	"Backward" button	With the "Backward" button, you start a motion in the negative direction according to the selected operating mode.
	"Stop" button	With the "Stop" button, you cancel a job or stop the axis.
	"Set homing point" button	With the "Set homing point" button, you set the actual position to the value specified in "Position". (Set homing point mode only)
Axis status		The "Axis status" area displays the status of the axis and the status of the drive.
	Drive ready	The drive is ready to execute setpoints.
	Error	An error has occurred at the technology object.
	Enabled	The technology object is enabled. The axis can be controlled with Motion Control instructions.
	Homed	The technology object is homed.
	More	With the "More" link, you display the "Technology object > Diagnostics > Status and error bits" dialog.
	Active errors	The error that occurred most recently is displayed in the "Active errors" text box.
	"Confirm" button	With the "Confirm" button, you acknowledge pending errors.
Current values		The "Current values" area shows the actual values of the axis.
	Position	Actual position of the axis
	Velocity	Actual velocity of the axis

Note

No transfer of the parameters

The configured parameter values are discarded when master control is returned. Transfer the values as needed into your configuration.

If you change configuration values with the axis control panel during operation, they have no effect on the operation of the axis control panel.

Operating mode

The following table shows the operating modes of the axis control panel:

Operating mode	Description
Homing	This function corresponds to active homing. The parameters for homing (Page 10223) must be configured. Homing is not possible with an absolute encoder. The technology object is not referenced when this mode is used with an absolute encoder.
Homing set position	This function corresponds to direct homing (absolute). With the "Homing set position" button, you set the actual position to the value specified in "Position" and the "Homed" status is set.
Jog	Motion commands occur by means of jogging. With the "Forward" or "Backward" button you start motion in the positive or negative direction. The motion runs for as long as you hold down the left mouse button.
Velocity specification/ speed setpoint	The axis is moved at the specified velocity or speed until you stop the movement. The motion commands are performed according to the setpoints assigned under "Controller".
Positioning relative	The positioning is executed as a controlled, relative traversing motion according to the defaults assigned under "Controller".
Positioning absolute	The positioning is executed as a controlled, absolute traversing motion according to the defaults assigned under "Controller". If you have enabled the "Modulo" setting of the technology object, the buttons "Forward" and "Backward" are shown in the "Controller" area. The axis is positioned within the modulo range. Position settings outside the modular range are recalculated to the modulo range. If you have not enabled the "Modulo" setting of the technology object, only the "Start" button is shown in the "Controller" area. You can directly approach the entered position.

Using the axis control panel

Requirements

- The CPU must be in RUN mode.
- The project has been created and downloaded to the CPU.
- The technology object is disabled by your user program (MC_Power.Enable = FALSE).
- The axis control panel for the technology object is not used by another installation of the TIA Portal (Team Engineering as of CPU V1.5).

Procedure

Proceed as follows to enable control the axis using the axis control panel:

1. Click "Activate" in the "Master control" area to assume master control over the technology object and to set up an online connection to the CPU.
   A warning message is displayed. Click "OK" to confirm.
2. In the "Axis" area, click the "Enable" button to enable the technology object.
3. In the drop-down list in the "Operation mode" area, select the desired function of the axis control panel.

4. In the "Control" area, specify the parameter values for your job.
5. Click the "Start", "Forward" or "Backward" button to start the job.
6. Click the "Stop" button to stop the job.
7. Repeat steps 3 through 6 for additional jobs.
8. In the "Axis" area, click the "Disable" button to disable the technology object.
9. In the "Master control" area, click the "Deactivate" button to return master control to your user program.

Optimization

Function and structure of the optimization

Description

The "Optimization" function supports you in determining the optimal gain (Kv factor) for the position control of the axis. The axis velocity profile is recorded by means of Trace function to this effect for the duration of a configurable positioning movement. Then you can evaluate the recording, and adapt the gain accordingly.

The "Optimization" function for the positioning axis technology object can be found in the Project Navigator under "Technology object > Commissioning".

The "Optimization" dialog is divided into the following areas:

- Master control
- Axis
- Optimize gain setting
- Trace

The following table lists the elements of optimization:

Area	Element	Description
Master control		In the "Master control" area, you take over master control of the technology object, or return it to your user program.
	"Activate" button	With the "Activate" button, you set up an online connection to the CPU and take over master control for the selected technology object. • To take over master control, the technology object must be disabled in the user program. • If the online connection to the CPU is lost during operation with the axis control panel, then after the sign of life monitoring has elapsed, the axis will be stopped with maximum deceleration. In this case, an error message is displayed ("ErrorID" = 16#8013) and the master control is passed back to the user program. If the master control of the axis control panel is lost repeatedly without a direct error message, the online connection to the CPU may be impaired because the communication load is too high. In this case, the alarm "Error during commissioning. Sign-of-life failure between controller and TIA Portal" is shown in the archive of the alarm display. To eliminate this error, adapt the sign of life monitoring in the tag of the technology object <TO>.ControlPanel.Input.TimeOut. The default value of the tag is 2000 ms. You can change the value of the tag directly. The change takes effect the next time the master control is activated. • Until master control is returned, the user program has no influence on the functions of the technology object. Motion Control jobs from the user program to the technology object are rejected with the error ("ErrorID" = 16#8012: axis control panel enabled). • When master control is taken over, the configuration of the technology object is adopted. Changes to the configuration of the technology object do not become effective until master control has been returned. For this reason, make any necessary changes before master control is taken over. • If master control has been taken over for the technology object, the axis control panel is blocked for access by another installation of the TIA Portal (Team Engineering as of CPU V1.5).
	"Deactivate" button	With the "Deactivate" button, you return master control to your user program.
Axis		In the "Axis" area, enable or disable the technology object.
	"Enable" button	With the "Enable" button, you enable the selected technology object
	"Disable" button	With the "Disable" button, you disable the selected technology object
Optimize gain setting		In the "Optimize gain setting" area, you make the settings for optimization of the gain.
	Gain	Current velocity gain (Kv) With the symbol beside the box, you open a list of values. The list of values contains the following gain values: • Online actual value • Online start value • Start value project In the input field for the actual value, enter the new value for the gain. The new value is applied by clicking the "Forward" or "Backward" button.
	Distance	Travel distance for a test step
	Measurement duration	Time for a test step

Area	Element	Description
	"Adapt dynamic values" checkbox	With the "Adapt dynamic values" check box, you enable adaptation of the acceleration and the maximum velocity.
	Acceleration	Presetting of the acceleration for a test step
	Maximum velocity	Presetting of the maximum velocity for a test step
	"Forward" button	With the "Forward" button, you start a test step for optimization in the positive direction.
	"Backward" button	With the "Backward" button, you start a test step for optimization in the negative direction.
Trace		The Trace function is displayed In the lower area of the "Optimization" dialog. With each test step a trace recording of the required parameters is automatically started and displayed after completion of the test step. After returning master control, the trace recording is deleted. You will find a full description of the trace function in the section on using the trace and logic analyzer function.

Note

No transfer of the parameters

The configured parameter values are discarded after master control is returned.

Transfer the values as needed into your configuration. You can apply the gain value in your configuration using the "Project start value" value.

See also

Control (Page 10240)

Optimize position controller

Requirements

- The CPU must be in RUN mode.
- The project has been created and downloaded to the CPU.
- The technology object is disabled by your user program (MC_Power.Enable = FALSE).
- The axis control panel for the technology object is not used by another installation of the TIA Portal (Team Engineering as of CPU V1.5).

Optimizing position control gain (Kv)

Proceed as follows to optimize the gain (Kv):

1. In the "Master control" area, click the "Activate" button to activate master control for the technology object, and to establish an online connection to the CPU.
   A warning message is displayed. Click "OK" to confirm.
2. In the "Axis" area, click the "Enable" button to enable the technology object.
3. As appropriate, configure values for the distance, duration, and dynamics of a test step.
4. Enter a start value for the gain. Start with a low value.
5. Click the "Forward" or "Start" button to start a test step for the optimization.
   For the specified duration, a setpoint is output according to the specified distance. The axis moves by the specified distance. A trace recording of the motion (setpoint and actual values) is created automatically.
   You will find a full description of the trace function in the section on using the trace and logic analyzer function.

   Note

   Adapt following error limits

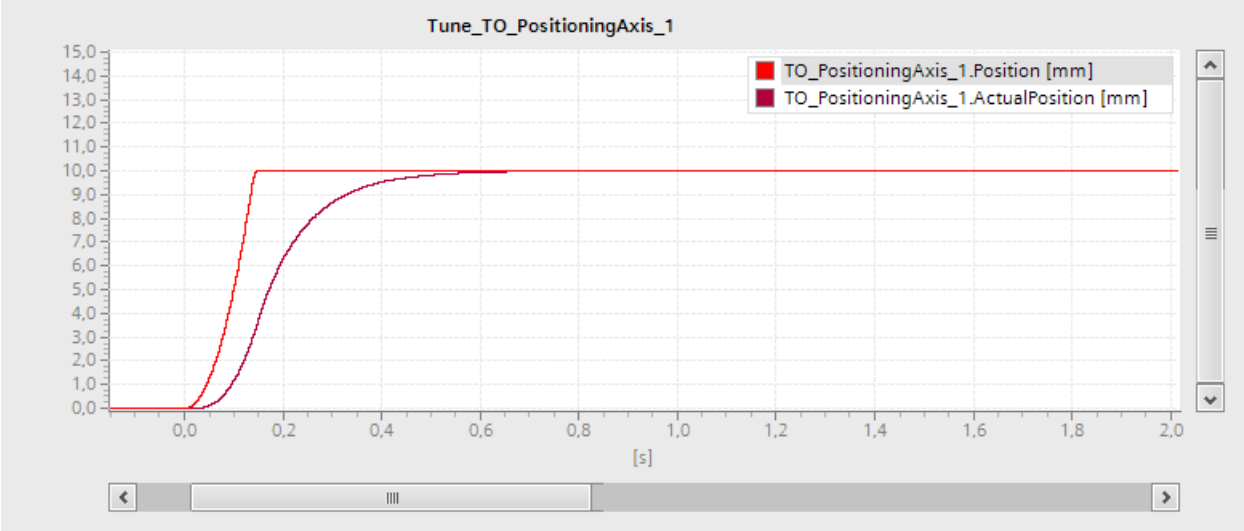
   If error messages from following error monitoring are repeatedly displayed during optimization, temporarily adapt the following error limits.

6. Evaluate the curve of the trace recording. Adapt the gain incrementally. Click the "Forward" or "Backward" button after each value you enter. This applies the value and start a new movement and trace recording each time.

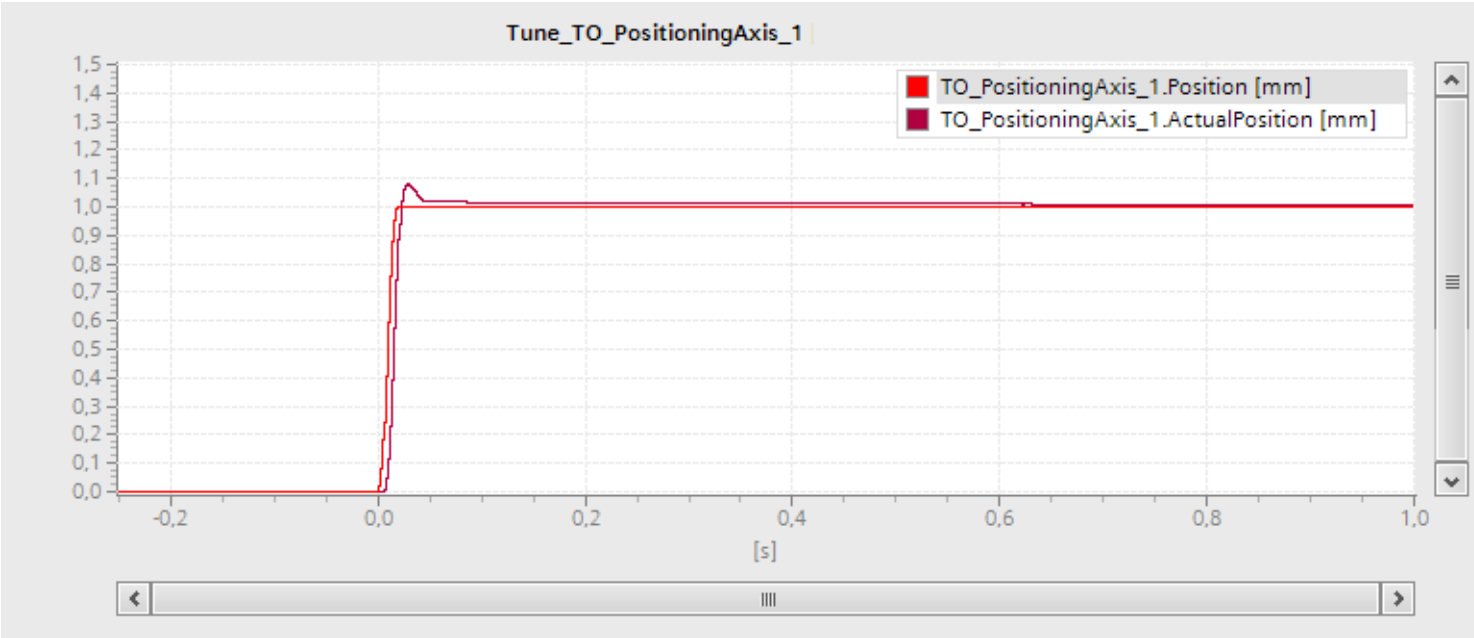
When adjusting the gain, pay attention to the following properties of the curve:

- The curve shows a brief compensation time.
- The curve does not show any motion reversal of the actual position.
- When approaching the position setpoint, no overshoot occurs.
- The curve shows a stable overall behavior (oscillation-free curve).

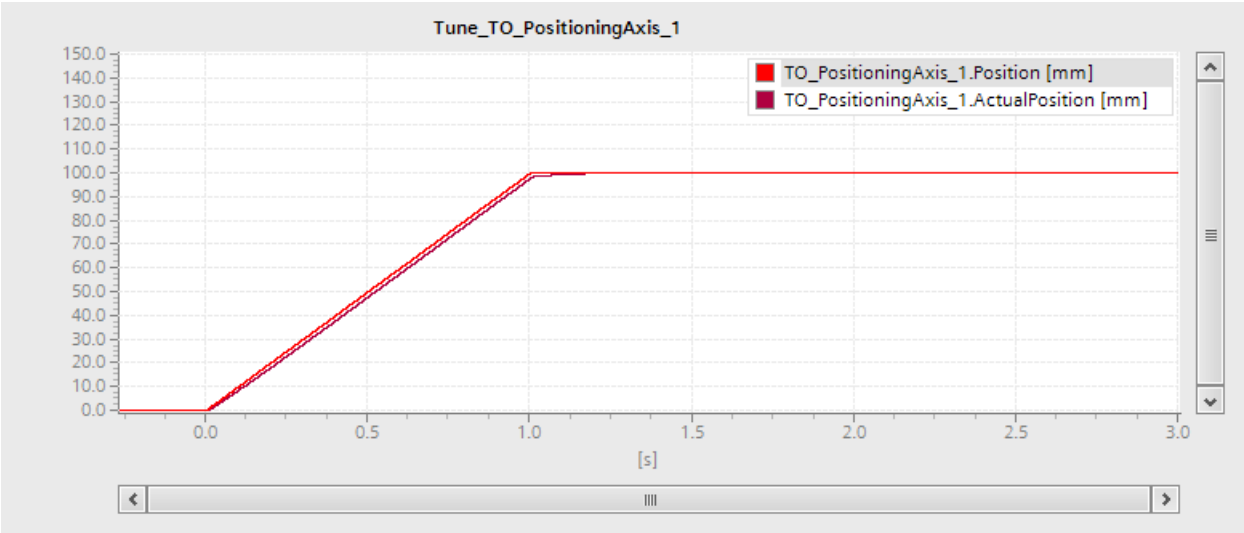
The following trace recording shows a curve with a long settling time:



The following trace recording shows a curve with overshoot when approaching the setpoint:



The following trace recording shows a curve in which the gain is optimal and the overall response is steady:



Transferring the position control gain (Kv) to the project

Proceed as follows to transfer the determined gain (Kv) into your project:

1. Click on the icon next to the "Gain" box.
   A list of values is displayed.
2. Enter the measured gain value in the "Project start value" box of the value list.
3. In the "Axis" area, click the "Disable" button to disable the technology object.
4. In the "Master control" area, click the "Deactivate" button to return master control to your user program.
5. Load your project into the CPU.

Diagnostics

Introduction

The Diagnostics chapter is limited to describing the diagnostic concept for Motion Control, and describing the Diagnostics view of the individual technology objects in the TIA Portal.

For more information about system diagnostics with the S7-1500 CPU, refer to the "System diagnostics" (http://support.automation.siemens.com/WW/view/en/59192926) function manual

Diagnostic concept

The diagnostic concept encompasses alarms and associated messages, as well as error messages in the Motion Control instructions. The TIA Portal also supports you with consistency checks during configuration of the technology objects, and during the creation of your user program.

All alarms during operation (from the CPU, technology, hardware etc.) are displayed in the Inspector window of the TIA Portal. Diagnostic information that relates to technology objects (technology alarms, status information) are additionally displayed in the Diagnostics window of the respective technology object.

During Motion Control, if an error occurs at a technology object (e.g. approaching a hardware limit switch), then a technology alarm (Page 10350) is triggered, and a corresponding message is displayed in the TIA Portal as well as on HMI devices.

In your user program, technology alarms are generally signaled via error bits in the technology data block. The program always displays the technology alarm that has top priority. In order to simplify error evaluation, the "Error" and "ErrorID" parameters of the Motion Control instructions also indicate that a technology alarm is pending.

Program errors (Page 10353) can occur during parameter assignment or during the processing sequence of the Motion Control instructions (e.g. invalid parameter specification when calling the instruction, initiation of a job without enablement via MC_Power). Errors in Motion Control instructions are indicated when the instructions are called, by means of the "Error" and "ErrorID" parameters.

Technology alarms

Description

If an error occurs at a technology object (e.g. approaching a hardware limit switch), then a technology alarm is triggered and indicated. The impact of a technology alarm on the technology object is specified by the alarm response.

Alarm classes

Technology alarms are divided into three classes:

- Warning (can be acknowledged)
  The processing of Motion Control jobs continues. The current motion of the axis can be influenced, e.g. by limitation of the current dynamic values to the configured limit values.
- Alarm (must be acknowledged)
  Motion jobs are aborted in accordance with the alarm response. In order to continue job processing after resolving the cause of error, you must acknowledge the alarms.
- Fatal error
  Motion jobs are aborted in accordance with the alarm response.
  To be able to use the technology object again after eliminating the problem, you will need to restart the technology object (Page 10336).

Display of technology alarms

A technology alarm is displayed in the following locations:

- TIA Portal
  - "Technology object > Diagnostics > Status and error bits"
    Display of pending technology alarms for each technology object.
  - "Technology object > Commissioning > Axis control panel"
    Display of the last pending technology alarm for each technology object.
  - "Inspector window > Diagnostics > Message display"
    Select the "Receive messages" check box under "Online & Diagnostics > Online Access" in order to display technology alarms via the message display.
    With an online connection to the CPU, the pending technology alarms for all technology objects are displayed. Additionally, the archive view is available to you.
    The message display can also be enabled at a connected HMI, and displayed.
  - "CPU > Online & diagnostics"
    Display of the technology alarms that have been recorded in the diagnostics buffer.
- User program
  - Tags <TO>.ErrorDetail.Number and <TO>.ErrorDetail.Reaction
    Indication of the number and the response for the last technology alarm that occurred.
  - Tag <TO>.StatusWord
    Via the "Error" bit it is indicated, that a technology alarm is pending.
  - Tag <TO>.ErrorWord
    Indication of alarms and fatal errors.
  - Tag <TO>.WarningWord
    Indication of warnings.
  - Parameter "Error" and "ErrorID"
    In a Motion Control instruction, the parameters "Error" = TRUE and "ErrorID" = 16#8001 indicate that a technology alarm is pending.
- Display of the CPU
  In order to show technology alarms on the CPU display, make the following setting when loading to the CPU:
  In the "Load preview" dialog, select the action "Consistent download" for the "Text libraries" entry.

Alarm response

A technology alarm always contains an alarm response, which describes the impact on the technology object. The alarm response is specified by the system.

The following list shows the possible alarm responses in order of ascending priority:

- No response (warnings only)
  <TO>.ErrorDetail.Reaction = 0
  The processing of Motion Control jobs continues. The current motion of the axis can be influenced, e.g. by limitation of the current dynamic values to the configured limit values.
- Stop with current dynamic values
  <TO>.ErrorDetail.Reaction = 1
  Active motion commands are aborted. The axis is braked with the dynamic values that are pending in the Motion Control instruction, and brought to a standstill.
- Stop with maximum dynamic values
  <TO>.ErrorDetail.Reaction = 2
  Active motion commands are aborted. The axis is braked with the dynamic values configured under "Technology object > Extended parameters > Dynamic limits", and brought to a standstill. The configured maximum jerk is taken into account at the same time.
  Stop with emergency stop ramp
  <TO>.ErrorDetail.Reaction = 3
  Active motion commands are aborted. The axis is braked with the emergency stop deceleration configured under "Technology object > Extended parameters > Emergency stop ramp", without any jerk limit, and brought to a standstill.
- Remove enable
  <TO>.ErrorDetail.Reaction = 4
  The setpoint zero is output, and the enablement is removed. The axis is braked depending on the configuration in the drive, and is brought to a standstill.

Acknowledging technology alarms

You can acknowledge technology alarms as follows:

- TIA Portal
  - "Technology object > Diagnostics > Status and error bits"
    Click "Confirm" to acknowledge all alarms and warnings pending for the selected technology object.
  - "Technology object > Commissioning > Axis control panel"
    Click "Confirm" to acknowledge all alarms and warnings pending for the selected technology object.
  - "Inspector window > Diagnostics > Message display"
    You can acknowledge the alarms and warnings for all technology objects either individually, or all at once.
    At an HMI with enabled message display, you can likewise acknowledge the alarms and warnings for all technology objects either individually, or all at once.
- User program
  Acknowledge pending alarms at a technology object using the Motion Control instruction "MC_Reset".

Additional information

A list of the technology alarms and alarm responses can be found in the appendix, Technology Alarms (Page 10403).

Errors in Motion Control instructions

Description

Errors in Motion Control instructions (e.g. invalid parameter value setting) are indicated at the "Error" and "ErrorID" output parameters.

Under the following conditions, "Error" = TRUE and "ErrorID" = 16#8xxx are indicated at the Motion Control instruction:

- Invalid status of the technology object, which prevents the execution of the job.
- Invalid parameter assignment of the Motion Control instruction, which prevents the execution of the job.
- As a result of the alarm response for an error at the technology object.

Error indication

If an error occurs in a Motion Control instruction, the "Error" parameter shows the value TRUE. The cause of the error can be found in the value of the ErrorID" parameter.

Jobs at the technology object are rejected when "Error" = TRUE. Running jobs are not influenced by rejected jobs.

If "Error" = TRUE and "ErrorID" = 16#8001 are indicated during the job processing, then a technology alarm has occurred. In this case, evaluate the indication of the technology alarm.

If "Error" = TRUE is displayed during the execution of a "MC_MoveJog" job, the axis is braked and brought to a standstill. In this case, the deceleration configured with the "MC_MoveJog" instruction takes effect.

Acknowledge errors

Acknowledging errors in Motion Control instructions is not required.

Restart a job after resolving the error.

Additional information

A list of the ErrorIDs can be found in the appendix, Error Detection (Page 10405).

Speed-controlled axis technology object

Status and error bits

Description

You use the "Technology object > Diagnostics > Status and error bits" diagnostic function in the TIA Portal to monitor the most important status and error messages for the technology object. The Diagnostics function is available in online operation.

The meaning of the status and error messages is described in the following tables. The associated technology object tag is given in parentheses.

Axis status

The following table shows the possible axis status values:

Status	Description
Enabled	The technology object is enabled. The axis can be controlled with Motion Control instructions. (<TO>.StatusWord.Enable)
Error	An error has occurred at the technology object. Detailed information about the error is available in the "Error" area, and in the <TO>.ErrorDetail.Number and <TO>.ErrorDetail.Reaction tags of the technology object. (<TO>.StatusWord.Error)
Restart active	The technology object is being reinitialized. (<TO>.StatusWord.RestartActive)
Axis control panel enabled	The axis control panel is enabled. The axis control panel has master control over the technology object. The axis cannot be controlled from the user program. (<TO>.StatusWord.ControlPanelActive)
Drive switched on	The drive is ready to execute setpoints. (<TO>.StatusDrive.InOperation)
Online start value changed	Data relevant for the restart has been changed. The changes will not be applied until the technology object is restarted. (<TO>.StatusWord.OnlineStartValuesChanged)

Motion status

The following table shows the possible axis motion status values:

Status	Description
Done (no active job)	No motion job is active at the technology object. (<TO>.StatusWord.Done)
Jog	The axis is being moved by a jog mode job from the Motion Control instruction "MC_MoveJog", or from the axis control panel. (<TO>.StatusWord.JogCommand)
Speed preset	The axis is traversed by means of a job with speed preset in Motion Control instruction "MC_MoveVelocity", or using the axis control panel. (<TO>.StatusWord.VelocityCommand)
Constant speed	The axis is traversing at a constant speed or is at a standstill. (<TO>.StatusWord.ConstantVelocity)
Accelerating	The axis is being accelerated. (<TO>.StatusWord.Accelerating)
Decelerating	The axis is being decelerated. (<TO>.StatusWord.Decelerating)

Warnings

The following table shows the possible warnings:

Warning	Description
Configuration adapted	One or several configuration parameters are adjusted internally at a certain time. (<TO>.WarningWord.ConfigurationFault)
Job rejected	Command cannot be executed. A Motion Control command cannot be executed, because necessary preconditions are not met. (<TO>.WarningWord.CommandNotAccepted)
Dynamic response limiting	Dynamic values were limited to the dynamic limits. (<TO>.WarningWord.DynamicError)

Errors

The following table shows the possible errors:

Errors	Description
System	An internal system error has occurred. (<TO>.ErrorWord.SystemFault)
Configuration	Configuration error One or more configuration parameters are inconsistent or invalid. The technology object was incorrectly configured, or editable configuration data were incorrectly modified during runtime of the user program. (<TO>.ErrorWord.ConfigurationFault)
User program	Error in the user program in a Motion Control instruction, or in its utilization. (<TO>.ErrorWord.UserFault)
Drive	Error in the drive. (<TO>.ErrorWord.DriveFault)
Data transmission	Error in communication with a connected device. (<TO>.ErrorWord.CommunicationFault)
I/O	Error accessing a logical address. (<TO>.ErrorWord.PeripheralError)
Job rejected	Command cannot be executed. A Motion Control instruction cannot be executed because the necessary conditions have not been fulfilled (e.g. technology object not homed). (<TO>.ErrorWord.CommandNotAccepted)
Dynamic response limiting	Dynamic values were limited to the dynamic limits. (<TO>.ErrorWord.DynamicError)

Confirm

Click "Confirm" to acknowledge alarms and warnings pending.

Additional information

An option for the evaluation of the individual status bits can be found in the Evaluating StatusWord, ErrorWord and WarningWord (Page 10317) section.

Motion status

Description

You use the "Technology object > Diagnostics > Motion status" diagnostic function in the TIA Portal to monitor the motion status of the axis. The Diagnostics function is available in online operation.

"Current values" area

The following table shows the meaning of the status data:

Status	Description
Speed setpoint	Speed setpoint of the axis (<TO>.Velocity)
Speed override	Speed setpoint correction as percentage The speed setpoint specified in motion control instructions or set by the axis control panel are superimposed with an override signal and corrected as a percentage. Valid speed correction values range from 0.0 % to 200.0 %. (<TO>.Override.Velocity)

"Dynamic limits" area

This area displays the limit values for the dynamic parameters.

The following table shows the meaning of the status data:

Status	Description
Speed	Configured maximum speed (<TO>.DynamicLimits.MaxVelocity)
Acceleration	Configured maximum acceleration (<TO>.DynamicLimits.MaxAcceleration)
Deceleration	Configured maximum deceleration (<TO>.DynamicLimits.MaxDeceleration)

PROFIdrive frame

Description

The "Technology object > Diagnostics > PROFIdrive frame" diagnostics function is used in the TIA Portal to monitor the PROFIdrive frame that the drive returns to the controller. The Diagnostics function is available in online operation.

"Drive" area

This area displays the following parameters contained in the PROFIdrive frame that the drive returns to the controller:

- Status words "SW1" and "SW2"
- The setpoint speed that was output to the drive (NSET)
- The actual speed that was signaled from the drive (NACT)

Positioning axis technology object

Status and error bits

Description

You use the "Technology object > Diagnostics > Status and error bits" diagnostic function in the TIA Portal to monitor the most important status and error messages for the technology object. The Diagnostics function is available in online operation.

The meaning of the status and error messages is described in the following tables. The associated technology object tag is given in parentheses.

Axis status

The following table shows the possible axis status values:

Status	Description
Enabled	The technology object is enabled. The axis can be controlled with Motion Control instructions. (<TO>.StatusWord.Enable)
Homed	The technology object is homed. The relationship between the position at the technology object and the mechanical position was successfully created. (<TO>.StatusWord.HomingDone)
Error	An error has occurred at the technology object. Detailed information about the error is available in the "Error" area, and in the <TO>.ErrorDetail.Number and <TO>.ErrorDetail.Reaction tags of the technology object. (<TO>.StatusWord.Error)
Restart active	The technology object is being reinitialized. (<TO>.StatusWord.RestartActive)

Status	Description
Axis control panel enabled	The axis control panel is enabled. The axis control panel has master control over the technology object. The axis cannot be controlled from the user program. (<TO>.StatusWord.ControlPanelActive)
Drive switched on	The drive is ready to execute setpoints. (<TO>.StatusDrive.InOperation)
Encoder valid	The encoder values are valid. (<TO>.StatusSensor[n].State)
Online start value changed	Data relevant for the restart has been changed. The changes will not be applied until after reinitialization (restart) of the technology object. (<TO>.StatusWord.OnlineStartValuesChanged)

Status limit switch

The following table shows possible enablements of the software and hardware limit switches:

Limit switch	Description
Negative software limit switch approached	The negative software limit switch was approached. (<TO>.StatusWord.SWLimitMinActive)
Positive software limit switch approached	The positive software limit switch was approached. (<TO>.StatusWord.SWLimitMaxActive)
Negative hardware limit switch approached	The negative hardware limit switch was approached. (<TO>.StatusWord.HWLimitMinActive)
Positive hardware limit switch approached	The positive hardware limit switch was approached or passed. (<TO>.StatusWord.HWLimitMaxActive)

Motion status

The following table shows the possible axis motion status values:

Status	Description
Done (no active job)	No motion job is active at the technology object. (<TO>.StatusWord.Done)
Homing job	The technology object is executing a homing job from the Motion Control instruction "MC_Home", or from the axis control panel. (<TO>.StatusWord.HomingCommand)
Jog	The axis is being moved by a jog mode job from the Motion Control instruction "MC_MoveJog", or from the axis control panel. (<TO>.StatusWord.JogCommand)
Velocity setpoint	A job traverses the axis at the speed setpoint specified in Motion Control instruction "MC_MoveVelocity" or at the axis control panel. (<TO>.StatusWord.VelocityCommand)
Positioning job	The axis is being moved by a positioning job from the Motion Control instructions "MC_MoveAbsolute", "MC_MoveRelative", or from the axis control panel. (<TO>.StatusWord.PositioningCommand)

Status	Description
Constant velocity	The axis is traversing at a constant velocity or is at a standstill. (<TO>.StatusWord.ConstantVelocity)
Standstill	The axis is at a standstill. (<TO>.StatusWord.StandStill)
Accelerating	The axis is being accelerated. (<TO>.StatusWord.Accelerating)
Decelerating	The axis is being decelerated. (<TO>.StatusWord.Decelerating)

Warnings

The following table shows the possible warnings:

Warning	Description
Configuration adapted	One or several configuration parameters are adjusted internally at a certain time. (<TO>.WarningWord.ConfigurationFault)
Job rejected	Command cannot be executed. A Motion Control command cannot be executed, because necessary preconditions are not met. (<TO>.WarningWord.CommandNotAccepted)
Dynamic response limiting	Dynamic values were limited to the dynamic limits. (<TO>.WarningWord.DynamicError)

Errors

The following table shows the possible errors:

Errors	Description
System	An internal system error has occurred. (<TO>.ErrorWord.SystemFault)
Configuration	Configuration error One or more configuration parameters are inconsistent or invalid. The technology object was incorrectly configured, or editable configuration data were incorrectly modified during runtime of the user program. (<TO>.ErrorWord.ConfigurationFault)
User program	Error in the user program in a Motion Control instruction, or in its utilization. (<TO>.ErrorWord.UserFault)
Drive	Error in the drive. (<TO>.ErrorWord.DriveFault)
Encoder	Error in the encoder system. (<TO>.ErrorWord.SensorFault)
Data transmission	Error in communication with a connected device. (<TO>.ErrorWord.CommunicationFault)

Errors	Description
I/O	Error accessing a logical address. (<TO>.ErrorWord.PeripheralError)
Job rejected	Command cannot be executed. A Motion Control instruction cannot be executed because the necessary conditions have not been fulfilled (e.g. technology object not homed). (<TO>.ErrorWord.CommandNotAccepted)
Homing	Error during a homing process. (<TO>.ErrorWord.HomingFault)
Positioning	The axis was not correctly positioned at the end of a positioning motion. (<TO>.ErrorWord.PositioningFault)
Dynamic response limiting	Dynamic values were limited to the dynamic limits. (<TO>.ErrorWord.DynamicError)
Following error	The maximum permitted following error was exceeded. (<TO>.ErrorWord.FollowingErrorFault)
SW limit switch	A software limit switch has been reached. (<TO>.ErrorWord.SwLimit)
Hardware limit switch	A hardware limit switch has been reached or overshot. (<TO>.ErrorWord.HWLimit)

Confirm

Click "Confirm" to acknowledge alarms and warnings pending.

Additional information

An option for the evaluation of the individual status bits can be found in the Evaluating StatusWord, ErrorWord and WarningWord (Page 10317) section.

Motion status

Description

You use the "Technology object > Diagnostics > Motion status" diagnostic function in the TIA Portal to monitor the motion status of the axis. The Diagnostics function is available in online operation.

"Current values" area

The following table shows the meaning of the status data:

Status	Description
Target position	Current target position of an active positioning job The target position value is only valid while a positioning job is busy. (<TO>.StatusPositioning.TargetPosition)
Actual position	Actual position of the axis If the technology object is not homed, then the value is displayed relative to the position that existed when the technology object was enabled. (<TO>.ActualPosition)
Actual velocity	Actual velocity of the axis (<TO>.ActualVelocity)
Velocity override	Percentage correction of the velocity override The velocity setpoint specified in motion control instructions or set by the axis control panel are superimposed with an override signal and corrected as a percentage. Valid velocity correction values range from 0.0 % to 200.0 %. (<TO>.Override.Velocity)

"Dynamic limits" area

This area displays the limit values for the dynamic parameters.

The following table shows the meaning of the status data:

Status	Description
Velocity	Configured maximum velocity (<TO>.DynamicLimits.MaxVelocity)
Acceleration	Configured maximum acceleration (<TO>.DynamicLimits.MaxAcceleration)
Deceleration	Configured maximum deceleration (<TO>.DynamicLimits.MaxDeceleration)

PROFIdrive frame

Description

The "Technology object > Diagnostics > PROFIdrive frame" diagnostics function is used in the TIA Portal to monitor the PROFIdrive frame returned by the drive and encoder. The display of the Diagnostics function is available in online operation.

"Drive" area

This area displays the following parameters contained in the PROFIdrive frame that the drive returns to the controller:

- Status words "SW1" and "SW2"
- The setpoint speed that was output to the drive (NSET)
- The actual speed that was signaled from the drive (NACT)

"Encoder" area

This area displays the following parameters contained in the PROFIdrive frame that the encoder returns to the controller:

- Status word "G1_ZSW"
- The actual position value "G1_XIST1" (cyclic actual encoder value)
- The actual position value "G1_XIST2" (absolute value of the encoder)

External encoder technology object

Status and error bits

Description

You use the "Technology object > Diagnostics > Status and error bits" diagnostic function in the TIA Portal to monitor the most important status and error messages for the technology object. The Diagnostics function is available in online operation.

The meaning of the status and error messages is described in the following tables. The associated technology object tag is given in parentheses.

Encoder status

The following table shows the possible external encoder status values:

Status	Description
Enabled	The technology object is enabled. (<TO>.StatusWord.Enable)
Homed	The technology object is homed. The relationship between the position at the technology object and the mechanical position was successfully created. (<TO>.StatusWord.HomingDone)
Error	An error has occurred at the technology object. Detailed information about the error is available in the "Error" area, and in the <TO>.ErrorDetail.Number and <TO>.ErrorDetail.Reaction tags of the technology object. (<TO>.StatusWord.Error)
Restart active	The technology object is being reinitialized. (<TO>.StatusWord.RestartActive)

Status	Description
Encoder valid	The encoder values are valid. (<TO>.StatusSensor[n].State)
Online start value changed	Data relevant for the restart has been changed. The changes will not be applied until the technology object is restarted. (<TO>.StatusWord.OnlineStartValuesChanged)

Motion status

The following table shows the possible axis motion status values:

Status	Description
Done (no active job)	No active Motion Control job at the TO. (Enable by "MC_Power" job excepted) (<TO>.StatusWord.Done)
Homing job	The technology object is executing a homing job from the Motion Control instruction "MC_Home". (<TO>.StatusWord.HomingCommand)

Errors

The following table shows the possible errors:

Errors	Description
System	An internal system error has occurred. (<TO>.ErrorWord.SystemFault)
Configuration	Configuration error One or more configuration parameters are inconsistent or invalid. The technology object was incorrectly configured, or editable configuration data were incorrectly modified during runtime of the user program. (<TO>.ErrorWord.ConfigurationFault)
User program	Error in the user program in a Motion Control instruction, or in its utilization. (<TO>.ErrorWord.UserFault)
Encoder	Error in the encoder system. (<TO>.ErrorWord.SensorFault)
Data transmission	Missing or faulty communication. (<TO>.ErrorWord.CommunicationFault)

Confirm

Click "Confirm" to acknowledge alarms and warnings pending.

Additional information

An option for the evaluation of the individual status bits can be found in the Evaluating StatusWord, ErrorWord and WarningWord (Page 10317) section.

Motion status

Description

You use the "Technology object > Diagnostics > Motion status" diagnostic function in the TIA Portal to monitor the encoder values. The Diagnostics function is available in online operation.

"Current values" area

The following table shows the meaning of the status data:

Status	Description
Actual position	Actual position of the axis If the technology object is not homed, then the value is displayed relative to the position that existed when the technology object was enabled. (<TO>.ActualPosition)
Actual velocity	Actual velocity of the axis (<TO>.ActualVelocity)

PROFIdrive frame

Description

The "Technology object > Diagnostics > PROFIdrive interface" diagnostic function is used in the TIA Portal to monitor the PROFIdrive frame of the encoder. The display of the diagnostics function is available in the online mode TO.

"Encoder" area

This area displays the following parameters contained in the PROFIdrive frame that the encoder returns to the controller:

- Status word "G1_ZSW"
- The actual position value "G1_XIST1" (cyclic actual encoder value)
- The actual position value "G1_XIST2" (absolute value of the encoder)

Appendix

Tags of the positioning axis technology object

Legend

Tag	Name of the tag	
Data type	Data type of the tag	
Values	Value range of the tag - minimum value to maximum value If no specific value is shown, the value range limits of the relevant data type apply or the information under "Description".	
W	Effectiveness of changes in the technology data block	
	DIR	Immediate: Changes to the value are made with a direct assignment and are immediately effective.
	RES	Restart: Changes to the start value in load memory are made using the extended instruction "WRIT_DBL" (write to DB in load memory). Changes will not take effect until after restart of the technology object.
	RON	Read only: The tag cannot and must not be changed.
Description	Description of the tag	

L - Linear specification R - Rotary specification

Access to the tags is with "<TO>.<tag name>". The placeholder <TO> represents the name of the technology object.

Actual values and setpoints (positioning axis)

The following tags indicate the setpoint and actual values of the technology object.

Tags

Legend (Page 10365)

Tag	Data type	Values	W	Description
Position	LREAL	-	RON	Setpoint position
Velocity	LREAL	-	RON	Setpoint velocity / setpoint speed
ActualPosition	LREAL	-	RON	Actual position
ActualVelocity	LREAL	-	RON	Actual velocity
ActualSpeed	LREAL	-	RON	Actual speed of the motor (with analog setpoint = 0.0)
Acceleration	LREAL	-	RON	Setpoint acceleration
ActualAcceleration	LREAL	-	RON	Actual acceleration

Actor tags (positioning axis)

The tag structure <TO>.Actor.<tag name> contains the controller-side configuration for the drive.

Tags

Legend (Page 10365)

Tag			Data type	Values	W	Description
Actor.			STRUCT			
	Type		DINT	-	RON	Drive connection 1: PROFIdrive frame 0: Analog output
	InverseDirection		BOOL	-	RES	Inversion of the setpoint FALSE: no TRUE: yes
	Interface.		STRUCT			
		AddressIn	UDINT	0 to 65535	RON	Input address for the PROFIdrive frame
		AdressOut	UDINT	0 to 65535	RON	Output address for the PROFIdrive frame or the analog setpoint
		Telegram	UDINT	-	RON	PROFIdrive frame of the drive 1: NSET (16 bit), NACT (16 bit) 2: NSET (32 bit), NACT (32 bit), signs of life 3: NSET (32 bit), NACT (32 bit), actual encoder value, signs of life 5: NSETP (32-bit), NACT (32-bit), actual encoder value, sign-of-life, DSC (NSET - setpoint speed, NACT - actual speed value)
		EnableDriveOutput	BOOL	-	RON	"Enable output" for analog drives FALSE: disabled TRUE: enabled
		EnableDriveOutputAddress	UDINT	0 to 65535	RON	Byte number for the "enable output"
		EnableDriveOutputBitNumber	UDINT	0 to 7	RON	Bit number for the "enabling output"
		DriveReadyInput	BOOL	-	RON	"Ready input" for analog drives The analog drive signals its readiness to receive setpoint speeds. FALSE: disabled TRUE: enabled
		DriveReadyInputAddress	UDINT	0 to 65535	RON	Byte number for the "ready input"
		DriveReadyInputBitNumber	UDINT	0 to 7	RON	Bit number for the "ready input"
	DriveParameter.					

Tag			Data type	Values	W	Description
		ReferenceSpeed	LREAL	0.0 to 1.0E12	RES	Reference value (100%) for the setpoint speed of the drive (N-set) The setpoint speed is transmitted in the PROFIdrive frame as a standardized value from -200% to 200% of the "ReferenceSpeed". For setpoint specification via an analog output, the analog output can be operated in the range from -117% to 117%, insofar as the drive permits this.
		MaxSpeed	LREAL	0.0 to 1.0E12	RES	Maximum value for the setpoint speed of the drive (N-set) (PROFIdrive: MaxSpeed ≤ 2 × ReferenceSpeed Analog setpoint: MaxSpeed ≤ 1.17 × ReferenceSpeed)

See also

Evaluating the technology data block (Page 10315)

Sensor[n] tags (positioning axis)

The tag structure <TO>.Sensor[n].<tag name> contains the controller end configuration for the encoder, and the configuration for active and passive homing.

Tags

Legend (Page 10365)

Tag			Data type	Values	W	Description
Sensor[n].			ARRAY [1..1] OF STRUCT			
	Type		DINT	-	RES	Encoder type 0: Incremental 1: Absolute 2: Cyclically absolute
	InverseDirection		BOOL	-	RES	Inversion of the actual value FALSE: no TRUE: yes
	System		DINT	-	RES	Encoder system 0: Linear encoder 1: Rotary encoder
	MountingMode		DINT	-	RES	Mounting type of the encoder 0: Incremental 1: Absolute 2: External
	Interface.					
		AddressIn	UDINT	0 to 65535	RON	Input address for the PROFIdrive frame

Tag			Data type	Values	W	Description
		AddressOut	UDINT	0 to 65535	RON	Output address for the PROFIdrive frame
		Telegram	UDINT	-	RON	PROFIdrive frame of the encoder 3: Actual values via drive frame (NSET (32 bit), NACT (32 bit), actual encoder value, signs of life) 5: Actual values from drive frame (NSETP (32-bit), NACT (32-bit), actual encoder value, sign-of-life, DSC) 81: Actual values via your own encoder telegram (actual encoder value, signs of life) 83: Actual values via your own encoder telegram (NACT (32 bit), actual encoder value, signs of life) For the technology object of external encoders, only frames 81 and 83 are valid. (NSET - setpoint speed, NACT - actual speed value)
	Parameter.					
		Resolution	LREAL	1.0E-12 to 1.0E12	RES	Resolution of a linear encoder (offset between two encoder pulses)
		StepsPerRevolution	UDINT	1 to 8388608	RES	Increments per rotary encoder revolution
		FineResolutionXist1	UDINT	0 to 31	RES	Number of bits for fine resolution Gn_XIST1 (cyclic actual encoder value)
		FineResolutionXist2	UDINT	0 to 31	RES	Number of bits for fine resolution Gn_XIST2 (absolute value of the encoder)
		DeterminableRevolutions	UDINT	0 to 8388608	RES	Number of differentiable encoder revolutions for a multi-turn absolute value encoder (For a single-turn absolute value encoder = 1; for an incremental encoder = 0)
		DistancePerRevolution	LREAL	0.0 to 1.0E12	RES	Load distance per revolution of an externally mounted encoder
	ActiveHoming.		STRUCT			
		Mode	DINT	-	RES	Homing mode 0: Homing mark only 1: Homing mark and proximity switch 2: Digital input
		SideInput	BOOL	-	DIR	Side of the digital input for active homing: FALSE: Negative side TRUE: Positive side
		Direction	DINT	-	DIR	Homing direction / homing mark approach direction 1: Positive homing direction 2: Negative homing direction
		HomePositionOffset	LREAL	-1.0E12 to 1.0E12	DIR	Home position offset
		DigitalInputAddress	UDINT	0 to 65535	RON	Byte number of the digital input
		DigitalInputBitNumber	UDINT	0 to 7	RON	Bit number of the digital input
	PassiveHoming.		STRUCT			

Tag			Data type	Values	W	Description
		Mode	DINT	-	RES	Homing mode 0: Homing mark only 1: Homing mark and proximity switch 2: Digital input
		SideInput	BOOL	-	DIR	Side of the digital input for passive homing: FALSE: Negative side TRUE: Positive side
		Direction	DINT	-	DIR	Homing direction / homing mark approach direction 0: Current homing direction 1: Positive homing direction 2: Negative homing direction
		DigitalInputAddres s	UDINT	0 to 65535	RON	Byte number of the digital input
		DigitalInputBitNum ber	UDINT	0 to 7	RON	Bit number of the digital input

See also

Evaluating the technology data block (Page 10315)

LoadGear tags (positioning axis)

The tag structure <TO>.LoadGear.<tag name> contains the configuration for the load gear.

Tags

Legend (Page 10365)

Tag		Data type	Value range	W	Description
LoadGear.		STRUCT			
	Numerator	UDINT	1 to 4294967295	RES	Load gear counter
	Denominator	UDINT	1 to 4294967295	RES	Load gear denominator

Mechanics tags (positioning axis)

The tag structure <TO>.Mechanics.<tag name> contains the configuration of the mechanics.

Tags

Legend (Page 10365)

Tag		Data type	Value range	W	Description
Mechanics.					
	LeadScrew	LREAL	0.0 to 1.0E12	RES	Leadscrew pitch

Properties tags (positioning axis)

The tag structure <TO>.Properties.<tag name> contains the configuration for the axis or type of motion.

Tags

Legend (Page 10365)

Tag		Data type	Value range	W	Description
Properties.					
	MotionType	DINT	-	RON	Indication of axis or motion type: 0: Linear axis or motion 1: Rotary axis or motion

Modulo tags (positioning axis)

The tag structure <TO>.Modulo.<tag name> contains the modulo configuration.

Tags

Legend (Page 10365)

Tag		Data type	Values	W	Description
Modulo.		STRUCT			
	Enable	BOOL	-	RES	FALSE: Modulo conversion disabled TRUE: Modulo conversion enabled When modulo conversion is enabled, a check is made for modulo length > 0.0
	Length	LREAL	0.001 to 1.0E12	RES	Modulo length
	StartValue	LREAL	-1.0E12 to 1.0E12	RES	Modulo start value

See also

Evaluating the technology data block (Page 10315)

DynamicLimits tags (positioning axis)

The tag structure <TO>.DynamicLimits.<tag name> contains the configuration of the dynamic limits. During Motion Control, no dynamic values greater than the dynamic limits are permitted. If you have specified greater values in a Motion Control instruction, then motion is performed using the dynamic limits, and a warning is indicated (alarm 501 to 503 - Dynamic values were limited).

Tags

Legend (Page 10365)

Tag		Data type	Values	W	Description
DynamicLimits.		STRUCT			
	MaxVelocity	LREAL	0.0 to 1.0E12	RES	Maximum permissible velocity of the axis
	MaxAcceleration	LREAL	0.0 to 1.0E12	DIR	Maximum permissible acceleration of the axis
	MaxDeceleration	LREAL	0.0 to 1.0E12	DIR	Maximum permissible deceleration of the axis
	MaxJerk	LREAL	0.0 to 1.0E12	DIR	Maximum permissible jerk on the axis

See also

Evaluating the technology data block (Page 10315)

DynamicDefaults tags (positioning axis)

The tag structure <TO>.DynamicDefaults.<tag name> contains the configuration of the dynamic defaults. These settings will be used, if you specify a dynamic value less than 0.0 in a Motion Control instruction (exceptions: MC_MoveJog.Velocity, MC_MoveVelocity.Velocity). Changes to the dynamic defaults will be applied with the next rising edge at the "Execute" parameter of a Motion Control instruction.

Tags

Legend (Page 10365)

Tag		Data type	Values	W	Description
DynamicDefaults.		STRUCT			
	Velocity	LREAL	0.0 to 1.0E12	DIR	Default velocity
	Acceleration	LREAL	0.0 to 1.0E12	DIR	Default acceleration
	Deceleration	LREAL	0.0 to 1.0E12	DIR	Default deceleration
	Jerk	LREAL	0.0 to 1.0E12	DIR	Default jerk
	EmergencyDeceleration	LREAL	0.0 to 1.0E12	DIR	Emergency stop deceleration

See also

Evaluating the technology data block (Page 10315)

PositionLimits_SW tags (positioning axis)

The tag structure <TO>.PositionLimits_SW.<tag name> contains the configuration for position monitoring with software limit switches. Software limit switches are used to limit the operating range of a positioning axis.

Tags

Legend (Page 10365)

Tag		Data type	Values	W	Description
PositionLimits_SW.		STRUCT			
	Active	BOOL	-	DIR	FALSE: Monitoring disabled TRUE: Monitoring enabled
	MinPosition	LREAL	-1.0E12 to 1.0E12	DIR	Position of negative software limit switches
	MaxPosition	LREAL	-1.0E12 to 1.0E12	DIR	Position of positive software limit switches ("MaxPosition" > "MinPosition")

See also

Evaluating the technology data block (Page 10315)

PositionLimits_HW tags (positioning axis)

The tag structure <TO>.PositionLimits_HW.<tag name> contains the configuration for position monitoring with hardware limit switches. Hardware limit switches are used to limit the traversing range of a positioning axis.

Tags

Legend (Page 10365)

Tag		Data type	Values	W	Description
PositionLimits_HW.		STRUCT			
	Active	BOOL	-	RES	FALSE: Monitoring disabled TRUE: Monitoring enabled With "Active", both (negative and positive) hardware limit switches are enabled or disabled.
	MinSwitchLevel	BOOL	-	RES	Level selection for enabling of the negative hardware limit switch: FALSE: Lower level (low-enabled) TRUE: Upper level (high-enabled)
	MinSwitchAddress	UDINT	0 to 65535	RON	Byte number for the negative hardware limit switch
	MinSwitchBitNumber	UDINT	0 to 7	RON	Bit number for the negative hardware limit switch

Tag		Data type	Values	W	Description
	MaxSwitchLevel	BOOL	-	RES	Level selection for enabling of the positive hardware limit switch: FALSE: Lower level (low-enabled) TRUE: Upper level (high-enabled)
	MaxSwitchAddress	UDINT	0 to 65535	RON	Byte number for the positive hardware limit switch
	MaxSwitchBitNumber	UDINT	0 to 7	RON	Bit number for the positive hardware limit switch

See also

Evaluating the technology data block (Page 10315)

Homing tags (positioning axis)

The tag structure <TO>.Homing.<tag name> contains the configuration for homing the TO.

Tags

Legend (Page 10365)

Tag		Data type	Values	W	Description
Homing.		STRUCT			
	AutoReversal	BOOL	-	RES	Reversal at the hardware limit switches FALSE: no TRUE: yes
	ApproachDirection	BOOL	-	DIR	Direction of approach to the homing position switch FALSE: Negative side TRUE: Positive direction
	ApproachVelocity	LREAL	L: 0.0 to 10000.0 mm/s R: 0.0 to 360000.0°/s	DIR	Approach velocity Velocity during active homing, with which the axis moves toward the proximity switch and toward home position.
	ReferencingVelocity	LREAL	L: 0.0 to 1000.0 mm/s R: 0.0 to 36000.0°/s	DIR	Homing velocity Velocity during active homing, with which the axis is moved toward home position.
	HomePosition	LREAL	-1.0E12 to 1.0E12	DIR	Home position

See also

Evaluating the technology data block (Page 10315)

Override tags (positioning axis)

The tag structure <TO>.Override.<tag name> contains the configuration for the override parameters. The override parameters are used to apply a correction percentage to default values. An override change takes effect immediately, and is performed with the dynamic settings in effect in the Motion Control instruction.

Tags

Legend (Page 10365)

Tag		Data type	Values	W	Description
Override.		STRUCT			
	Velocity	LREAL	0.0 to 200.0%	DIR	Velocity or speed override Percentage correction of the velocity/speed

See also

Evaluating the technology data block (Page 10315)

PositionControl tags (positioning axis)

The tag structure <TO>.PositionControl.<tag name> contains the settings for position control.

Tags

Legend (Page 10365)

Tag		Data type	Values	W	Description
PositionControl.					
	Kv	LREAL	0.0 to 2147480.0	DIR	Proportional gain in the position control ("Kv" > 0.0)
	Kpc	LREAL	0.0 to 150.0%	DIR	Velocity pre-control in the position control Recommended setting: • Isochronous drive connection via PROFIdrive: 100.0% • Non-isochronous drive connection via PROFIdrive: 0.0 to 100.0% • Analog drive connection: 0.0 to 100.0%
	EnableDSC	BOOL	-	RES	Dynamic Servo Control (DSC) FALSE: DSC disabled TRUE: DSC enabled DSC is only possible if frame 5 is used.

See also

Evaluating the technology data block (Page 10315)

FollowingError tags (positioning axis)

The tag structure <TO>.FollowingError.<tag name> contains the configuration for following error monitoring.

If the permissible following error is exceeded, then technology alarm 521 is output, and the technology object is disabled (alarm response: remove enable).

When the warning level is reached, a warning is output (technology alarm 522).

Tags

Legend (Page 10365)

Tag		Data type	Values	W	Description
FollowingError.		STRUCT			
	EnableMonitoring	BOOL	-	RES	FALSE: Following error monitoring disabled TRUE: Following error monitoring enabled
	MinValue	LREAL	0.0 to 1.0E12	DIR	Permissible following error at velocities below the value of "MinVelocity".
	MaxValue	LREAL	0.0 to 1.0E12	DIR	Maximum permissible following error, which may be reached at the maximum velocity.
	MinVelocity	LREAL	0.0 to 1.0E12	DIR	"MinValue" is permissible below this velocity, and is held constant.
	WarningLevel	LREAL	0.0 to 100.0	DIR	Warning level: Percentage value relative to the valid maximum following error.

See also

Evaluating the technology data block (Page 10315)

PositioningMonitoring tags (positioning axis)

The tag structure <TO>.PositioningMonitoring.<tag name> contains the configuration for position monitoring at the end of a positioning motion.

If the actual position value at the end of a positioning motion is reached within the tolerance time and remains in the positioning window for the minimum dwell time, then <TO>.StatusWord.Done is set in the technology data block. This completes a Motion Control job.

If the tolerance time is exceeded, then technology alarm 541 "Positioning monitoring" with supplemental value 1: "Target area not reached within the positioning tolerance time" will be indicated.

If the minimum dwell time is not met, then technology alarm 541 "Positioning monitoring" with supplemental value 2: "Target area exited within the minimum dwell time" will be indicated.

Tags

Legend (Page 10365)

Tag		Data type	Values	W	Description
PositioningMonitoring.		STRUCT			
	ToleranceTime	LREAL	0.0 to 1.0E12	DIR	Tolerance time Maximum permitted duration from reaching velocity setpoint zero, until entrance into the positioning window.
	MinDwellTime	LREAL	0.0 to 1.0E12	DIR	Minimum dwell time in the positioning window:
	Window	LREAL	0.0 to 1.0E12	DIR	Positioning window

See also

Evaluating the technology data block (Page 10315)

StandstillSignal tags (positioning axis)

The tag structure <TO>.StandstillSignal.<tag name> contains the configuration for the standstill signal.

If the actual velocity value is below the velocity threshold, and does not exceed it during the minimum dwell time, then the standstill signal <TO>.StatusWord.Standstill is set.

Tags

Legend (Page 10365)

Tag		Data type	Values	W	Description
StandstillSignal.		STRUCT			Configuration for the standstill signal
	VelocityThreshold	LREAL	0.0 to 1.0E12	DIR	Velocity threshold If velocity is below this threshold, then the minimum dwell time begins.
	MinDwellTime	LREAL	0.0 to 1.0E12	DIR	Minimum dwell time

See also

Evaluating the technology data block (Page 10315)

StatusPositioning tags (positioning axis)

The tag structure <TO>.StatusPositioning.<tag name> indicates the status of a positioning motion.

Tags

Legend (Page 10365)

Tag		Data type	Values	W	Description
StatusPositioning.		STRUCT			
	Distance	LREAL	-	RON	Distance to the target position
	TargetPosition	LREAL	-	RON	Target position
	FollowingError	LREAL	-	RON	Current following error

See also

Evaluating the technology data block (Page 10315)

StatusDrive tags (positioning axis)

The tag structure <TO>.StatusDrive.<tag name> indicates the status of the drive.

Tags

Legend (Page 10365)

Tag		Data type	Values	W	Description
StatusDrive.		STRUCT			
	InOperation	BOOL	-	RON	Operational status of the drive FALSE: Drive not ready. Setpoints will not be executed. TRUE: Drive ready. Setpoints can be executed.
	CommunicationOK	BOOL	-	RON	Cyclic BUS communication between controller and drive FALSE: Not established TRUE: Established
	Error	BOOL	-	RON	FALSE: No error at the drive TRUE: Error at the drive

See also

Evaluating the technology data block (Page 10315)

StatusSensor[n] tags (positioning axis)

The tag structure <TO>.StatusSensor[n].<tag name> indicates the status of the measuring system.

Tags

Legend (Page 10365)

Tag		Data type	Values	W	Description
StatusSensor[n].		ARRAY [1..1] OF STRUCT			
	State	DINT	-	RON	Status of the actual encoder value: 0: "NOT_VALID (not valid) 1: "Waiting_FOR_VALID (waiting for valid status) 2: "VALID (valid)
	CommunicationOK	BOOL	-	RON	Cyclic BUS communication between controller and encoder FALSE: Not established TRUE: Established
	Error	BOOL	-	RON	FALSE: No error in the measuring system TRUE: Error in the measuring system.
	AbsEncoderOffset	LREAL	-	RON	Home point offset to the value of an absolute value encoder. The value will be retentively stored in the CPU.

See also

Evaluating the technology data block (Page 10315)

StatusWord tag (positioning axis)

The <TO>.StatusWord tag contains the status information of the technology object.

Notes on the evaluation of the individual bits (e.g. bit 5 "HomingDone") can be found in the section Evaluating StatusWord, ErrorWord and WarningWord (Page 10317) section.

Tag

Legend (Page 10365)

Tag	Data type	Values	W	Description
Status Word	DWORD	-	RON	Status data of the technology object
Bit 0	-	-	-	"Enable" Enable status 0: Technology object disabled 1: Technology object enabled
Bit 1	-	-	-	"Error" 0: No error present 1: Error present

Tag	Data type	Values	W	Description
Bit 2	-	-	-	"RestartActive" 0: No "Restart" active 1: "Restart" active. The technology object is being reinitialized.
Bit 3	-	-	-	"OnlineStartValuesChanged" 0: "Restart" tag unchanged 1: Change to "Restart" tag. For the changes to be applied, the technology object must be reinitialized.
Bit 4	-	-	-	"ControlPanelActive" Axis control panel 0: disabled 1: enabled
Bit 5	-	-	-	"HomingDone" Homing status 0: Technology object not homed 1: Technology object homed
Bit 6	-	-	-	"Done" 0: Motion job in process, or axis control panel enabled 1: No motion job in process and axis control panel disabled
Bit 7	-	-	-	"Standstill" Standstill signal 0: Axis in motion 1: Axis at a standstill
Bit 8	-	-	-	"PositioningCommand" 0: No positioning job active 1: Positioning job active ("MC_MoveReleative", "MC_MoveAbsolute")
Bit 9	-	-	-	"JogCommand" 0: No "MC_MoveJog" frame active 1: "MC_MoveJog" job active
Bit 10	-	-	-	"VelocityCommand" 0: No "MC_MoveVelocity" job active 1: "MC_MoveVelocity" job active
Bit 11	-	-	-	"HomingCommand" 0: No "MC_Home" job in process 1: "MC_Home" job in process
Bit 12	-	-	-	"ConstantVelocity" 0: The axis accelerates or decelerates. 1: Velocity setpoint reached. The axis is traversing at this constant velocity or is at a standstill.
Bit 13	-	-	-	"Accelerating" 0: No acceleration process active 1: Acceleration process active
Bit 14	-	-	-	"Decelerating" 0: No deceleration process active 1: Deceleration process active

Tag	Data type	Values	W	Description
Bit 15	-	-	-	"SWLimitMinActive" 0: Negative software limit switch not reached 1: Negative software limit switch reached or exceeded
Bit 16	-	-	-	"SWLimitMaxActive" 0: Positive software limit switch not reached 1: Positive software limit switch reached or exceeded
Bit 17	-	-	-	"HWLimitMinActive" 0: Negative hardware limit switch not reached 1: Negative Hardware limit switch reached or exceeded
Bit 18	-	-	-	"HWLimitMaxActive" 0: Positive hardware limit switch not reached 1: Positive hardware limit switch reached or exceeded
Bit 19	-	-	-	"ErrorStopActive" Stop as a result of alarm response 0: not active 1: active
Bit 20	-	-	-	"MC_PowerStopActive" Stop as a result of disablement via "MC_Power" 0: not active 1: active
Bit 21... Bit 31	-	-	-	Reserved

See also

Evaluating the technology data block (Page 10315)

ErrorWord tag (positioning axis)

The <TO>.ErrorWord tag indicates errors at the technology object (technology alarms).

Notes on the evaluation of the individual bits (e.g. bit 3, CommandNotAccepted) can be found in the Evaluating StatusWord, ErrorWord and WarningWord (Page 10317) section.

Tag

Legend (Page 10365)

Tag	Data type	Values	W	Description
Error Word	DWORD	-	RON	
Bit 0	-	-	-	"SystemFault" System error

Tag	Data type	Values	W	Description
Bit 1	-	-	-	"ConfigurationFault" Configuration error One or more configuration parameters are inconsistent or invalid.
Bit 2	-	-	-	"UserFault" Error in the user program in a motion control instruction, or in its use
Bit 3	-	-	-	"CommandNotAccepted" Command cannot be executed. A Motion Control command cannot be executed, because necessary preconditions are not met.
Bit 4	-	-	-	"DriveFault" Error in the drive
Bit 5	-	-	-	"SensorFault" Error in the encoder system.
Bit 6	-	-	-	"DynamicError" Specified dynamic values are limited to permissible values.
Bit 7	-	-	-	"CommunicationFault" Communication error Missing or faulty communication.
Bit 8	-	-	-	"SWLimit" Software limit switch reached or overshot.
Bit 9	-	-	-	"HWLimit" Hardware limit switch reached or overshot.
Bit 10	-	-	-	"HomingFault" Error during homing process The homing cannot be completed.
Bit 11	-	-	-	"FollowingErrorFault" Following error limits exceeded
Bit 12	-	-	-	"PositioningFault" Positioning error
Bit 13	-	-	-	"PeripheralError" Error accessing a logical address
Bit 14... Bit 31	-	-	-	Reserved

See also

Evaluating the technology data block (Page 10315)

ErrorDetail tags (positioning axis)

The tag structure <TO>.ErrorDetail.<tag name> contains the interrupt number and the effective local interrupt response to the technology interrupt that is currently pending at the technology object.

A list of the technology alarms and alarm responses can be found in the appendix, Technology alarms (Page 10403).

Tags

Legend (Page 10365)

Tag		Data type	Values	W	Description
ErrorDetail.		STRUCT			
	Number	UDINT	-	RON	Alarm number
	Reaction	DINT	0 to 4	RON	Effective alarm response 0: No response 1: Stop with current dynamic values 2 Stop with maximum dynamic values 3: Stop with emergency stop ramp 4: Remove enable For the external encoder technology object, only alarm responses 0 and 4 are possible.

See also

Evaluating the technology data block (Page 10315)

Technology alarms (Page 10350)

WarningWord tag (positioning axis)

The <TO>.WarningWord tag indicates pending warnings at the technology object.

Notes on the evaluation of the individual bits (e.g. bit 11, "FollowingErrorWarning") can be found in the Evaluating StatusWord, ErrorWord and WarningWord (Page 10317) section.

Tag

Legend (Page 10365)

Tag	Data type	Values	W	Description
WarningWord	DWORD	-	RON	
Bit 0	-	-	-	Reserved
Bit 1	-	-	-	"ConfigurationFault" Configuration error One or several configuration parameters are adjusted internally.
Bit 2	-	-	-	Reserved

Tag	Data type	Values	W	Description
Bit 3	-	-	-	"CommandNotAccepted" Command cannot be executed. A Motion Control command cannot be executed, because necessary preconditions are not met.
Bit 4	-	-	-	Reserved
Bit 5	-	-	-	Reserved
Bit 6	-	-	-	"DynamicError" Specified dynamic values are limited to permissible values.
Bit 7	-	-	-	Reserved
Bit 8	-	-	-	Reserved
Bit 9	-	-	-	Reserved
Bit 10	-	-	-	Reserved
Bit 11	-	-	-	"FollowingErrorWarning" Warning level of the following error monitoring reached / exceeded
Bit 12	-	-	-	Reserved
Bit 13	-	-	-	Reserved
Bit 14... Bit 31	-	-	-	Reserved

See also

Evaluating the technology data block (Page 10315)

ControlPanel tags (positioning axis)

The tag structure <TO>.ControlPanel.<tag name> contains no application-relevant data. This tag structure is internally used.

One exception is the tag <TO>.ControlPanel.Input.TimeOut for adapting the sign-of-life monitoring.

Tags

Legend (Page 10365)

Tag				Data type	Values	W	Description
ControlPanel.				STRUCT			
	Input.			STRUCT			
		Command.		ARRAY [1..2] OF STRUCT			
			ReqCounter	UDINT	-	DIR	-
			Type	UDINT	-	DIR	-
			Position	LREAL	-	DIR	-

Tag				Data type	Values	W	Description
			Velocity	LREAL	-	DIR	-
			Acceleration	LREAL	-	DIR	-
			Deceleration	LREAL	-	DIR	-
			Jerk	LREAL	-	DIR	-
			Param	LREAL	-	DIR	-
		TimeOut		LREAL	-	DIR	Sign-of-life monitoring Preset value: 2000 ms
		EsLifeSign		UDINT	-	DIR	-
	Output.			STRUCT			
		Command.		ARRAY [1..2] OF STRUCT			
			AckCounter	UDINT	-	RON	-
			Error	BOOL	-	RON	-
			ErrorID	UDINT	-	RON	-
			Done	BOOL	-	RON	-
			Aborted	BOOL	-	RON	-
		RTLifeSign		UDINT	-	RON	-

See also

Evaluating the technology data block (Page 10315)

InternalTOTrace tags (positioning axis)

The tag structure <TO>.InternalTOTrace.<tag name> contains no application-relevant data. This tag structure is internally used.

Tags

Legend (Page 10365)

Tag		Data type	Values	W	Description
InternalTOTrace.		ARRAY [1..4] OF STRUCT			
	Id	DINT	-	DIR	-
	Value	LREAL	-	RON	-

See also

Evaluating the technology data block (Page 10315)

Tags of the technology object SpeedAxis

Legend

Tag	Name of the tag	
Data type	Data type of the tag	
Values	Value range of the tag - minimum value to maximum value If no specific value is shown, the value range limits of the relevant data type apply or the information under "Description".	
W	Effectiveness of changes in the technology data block	
	DIR	Immediate: Changes to the value are made with a direct assignment and are immediately effective.
	RES	Restart: Changes to the start value in load memory are made using the extended instruction "WRIT_DBL" (write to DB in load memory). Changes will not take effect until after restart of the technology object.
	RON	Read only: The tag cannot and must not be changed.
Description	Description of the tag	

L - Linear specification R - Rotary specification

Access to the tags is with "<TO>.<tag name>". The placeholder <TO> represents the name of the technology object.

Actual values and setpoints (speed axis)

The following tags indicate the setpoint and actual values of the technology object.

Tags

Legend (Page 10385)

Tag	Data type	Values	W	Description
Velocity	LREAL	-	RON	Setpoint velocity / setpoint speed
ActualSpeed	LREAL	-	RON	Actual speed of the motor (with analog setpoint = 0.0)
Acceleration	LREAL	-	RON	Setpoint acceleration

Actor tags (speed axis)

The tag structure <TO>.Actor.<tag name> contains the controller-side configuration for the drive.

Tags

Legend (Page 10385)

Tag			Data type	Values	W	Description
Actor.			STRUCT			
	Type		DINT	-	RON	Drive connection 1: PROFIdrive frame 0: Analog output
	InverseDirection		BOOL	-	RES	Inversion of the setpoint FALSE: no TRUE: yes
	Interface.		STRUCT			
		AddressIn	UDINT	0 to 65535	RON	Input address for the PROFIdrive frame
		AdressOut	UDINT	0 to 65535	RON	Output address for the PROFIdrive frame or the analog setpoint
		Telegram	UDINT	-	RON	PROFIdrive frame of the drive 1: NSET (16 bit), NACT (16 bit) 2: NSET (32 bit), NACT (32 bit), signs of life 3: NSET (32 bit), NACT (32 bit), actual encoder value, signs of life 5: NSETP (32-bit), NACT (32-bit), actual encoder value, sign-of-life, DSC (NSET - setpoint speed, NACT - actual speed value)
		EnableDriveOutput	BOOL	-	RON	"Enable output" for analog drives FALSE: disabled TRUE: enabled
		EnableDriveOutput Address	UDINT	0 to 65535	RON	Byte number for the "enable output"
		EnableDriveOutput BitNumber	UDINT	0 to 7	RON	Bit number for the "enabling output"
		DriveReadyInput	BOOL	-	RON	"Ready input" for analog drives The analog drive signals its readiness to receive setpoint speeds. FALSE: disabled TRUE: enabled
		DriveReadyInputAddress	UDINT	0 to 65535	RON	Byte number for the "ready input"
		DriveReadyInputBitNumber	UDINT	0 to 7	RON	Bit number for the "ready input"
	DriveParameter.					

Tag			Data type	Values	W	Description
		ReferenceSpeed	LREAL	0.0 to 1.0E12	RES	Reference value (100%) for the setpoint speed of the drive (N-set) The setpoint speed is transmitted in the PROFIdrive frame as a standardized value from -200% to 200% of the "ReferenceSpeed". For setpoint specification via an analog output, the analog output can be operated in the range from -117% to 117%, insofar as the drive permits this.
		MaxSpeed	LREAL	0.0 to 1.0E12	RES	Maximum value for the setpoint speed of the drive (N-set) (PROFIdrive: MaxSpeed ≤ 2 × ReferenceSpeed Analog setpoint: MaxSpeed ≤ 1.17 × ReferenceSpeed)

LoadGear tags (speed axis)

The tag structure <TO>.LoadGear.<tag name> contains the configuration for the load gear.

Tags

Legend (Page 10385)

Tag		Data type	Value range	W	Description
LoadGear.		STRUCT			
	Numerator	UDINT	1 to 4294967295	RES	Load gear counter
	Denominator	UDINT	1 to 4294967295	RES	Load gear denominator

DynamicLimits tags (speed axis)

The tag structure <TO>.DynamicLimits.<tag name> contains the configuration of the dynamic limits. During Motion Control, no dynamic values greater than the dynamic limits are permitted. If you have specified greater values in a Motion Control instruction, then motion is performed using the dynamic limits, and a warning is indicated (alarm 501 to 503 - Dynamic values were limited).

Tags

Legend (Page 10385)

Tag		Data type	Values	W	Description
DynamicLimits.		STRUCT			
	MaxVelocity	LREAL	0.0 to 1.0E12	RES	Maximum permissible velocity of the axis
	MaxAcceleration	LREAL	0.0 to 1.0E12	DIR	Maximum permissible acceleration of the axis

Tag		Data type	Values	W	Description
	MaxDeceleration	LREAL	0.0 to 1.0E12	DIR	Maximum permissible deceleration of the axis
	MaxJerk	LREAL	0.0 to 1.0E12	DIR	Maximum permissible jerk on the axis

DynamicDefaults tags (speed axis)

The tag structure <TO>.DynamicDefaults.<tag name> contains the configuration of the dynamic defaults. These settings will be used, if you specify a dynamic value less than 0.0 in a Motion Control instruction (exceptions: MC_MoveJog.Velocity, MC_MoveVelocity.Velocity). Changes to the dynamic defaults will be applied with the next rising edge at the "Execute" parameter of a Motion Control instruction.

Tags

Legend (Page 10385)

Tag		Data type	Values	W	Description
DynamicDefaults.		STRUCT			
	Velocity	LREAL	0.0 to 1.0E12	DIR	Default velocity
	Acceleration	LREAL	0.0 to 1.0E12	DIR	Default acceleration
	Deceleration	LREAL	0.0 to 1.0E12	DIR	Default deceleration
	Jerk	LREAL	0.0 to 1.0E12	DIR	Default jerk
	EmergencyDeceleration	LREAL	0.0 to 1.0E12	DIR	Emergency stop deceleration

Override tags (speed axis)

The tag structure <TO>.Override.<tag name> contains the configuration for the override parameters. The override parameters are used to apply a correction percentage to default values. An override change takes effect immediately, and is performed with the dynamic settings in effect in the Motion Control instruction.

Tags

Legend (Page 10385)

Tag		Data type	Values	W	Description
Override.		STRUCT			
	Velocity	LREAL	0.0 to 200.0%	DIR	Velocity or speed override Percentage correction of the velocity/speed

StatusDrive tags (speed axis)

The tag structure <TO>.StatusDrive.<tag name> indicates the status of the drive.

Tags

Legend (Page 10385)

Tag		Data type	Values	W	Description
StatusDrive.		STRUCT			
	InOperation	BOOL	-	RON	Operational status of the drive FALSE: Drive not ready. Setpoints will not be executed. TRUE: Drive ready. Setpoints can be executed.
	CommunicationOK	BOOL	-	RON	Cyclic BUS communication between controller and drive FALSE: Not established TRUE: Established
	Error	BOOL	-	RON	FALSE: No error at the drive TRUE: Error at the drive

StatusWord tag (speed axis)

The <TO>.StatusWord tag contains the status information of the technology object.

Notes on the evaluation of the individual bits (e.g. bit 5 "HomingDone") can be found in the section Evaluating StatusWord, ErrorWord and WarningWord (Page 10317) section.

Tag

Legend (Page 10385)

Tag	Data type	Values	W	Description
Status Word	DWORD	-	RON	Status data of the technology object
Bit 0	-	-	-	"Enable" Enable status 0: Technology object disabled 1: Technology object enabled
Bit 1	-	-	-	"Error" 0: No error present 1: Error present
Bit 2	-	-	-	"RestartActive" 0: No "Restart" active 1: "Restart" active. The technology object is being reinitialized.
Bit 3	-	-	-	"OnlineStartValuesChanged" 0: "Restart" tag unchanged 1: Change to "Restart" tag. For the changes to be applied, the technology object must be reinitialized.

Tag	Data type	Values	W	Description
Bit 4	-	-	-	"ControlPanelActive" Axis control panel 0: disabled 1: enabled
Bit 5	-	-	-	Reserved
Bit 6	-	-	-	"Done" 0: Motion job in process, or axis control panel enabled 1: No motion job in process and axis control panel disabled
Bit 7	-	-	-	Reserved
Bit 8	-	-	-	Reserved
Bit 9	-	-	-	"JogCommand" 0: No "MC_MoveJog" frame active 1: "MC_MoveJog" job active
Bit 10	-	-	-	"VelocityCommand" 0: No "MC_MoveVelocity" job active 1: "MC_MoveVelocity" job active
Bit 11	-	-	-	Reserved
Bit 12	-	-	-	"ConstantVelocity" 0: The axis accelerates or decelerates. 1: Velocity setpoint reached. The axis is traversing at this constant velocity or is at a standstill.
Bit 13	-	-	-	"Accelerating" 0: No acceleration process active 1: Acceleration process active
Bit 14	-	-	-	"Decelerating" 0: No deceleration process active 1: Deceleration process active
Bit 15	-	-	-	Reserved
Bit 16	-	-	-	Reserved
Bit 17	-	-	-	Reserved
Bit 18	-	-	-	Reserved
Bit 19	-	-	-	"ErrorStopActive" Stop as a result of alarm response 0: not active 1: active
Bit 20	-	-	-	"MC_PowerStopActive" Stop as a result of disablement via "MC_Power" 0: not active 1: active
Bit 21... Bit 31	-	-	-	Reserved

ErrorWord tag (speed axis)

The <TO>.ErrorWord tag indicates errors at the technology object (technology alarms).

Notes on the evaluation of the individual bits (e.g. bit 3 "CommandNotAccepted") can be found in the section Evaluating StatusWord, ErrorWord and WarningWord (Page 10317) section.

Tag

Legend (Page 10385)

Tag	Data type	Values	W	Description
Error Word	DWORD	-	RON	
Bit 0	-	-	-	"SystemFault" System error
Bit 1	-	-	-	"ConfigurationFault" Configuration error One or more configuration parameters are inconsistent or invalid.
Bit 2	-	-	-	"UserFault" Error in the user program in a motion control instruction, or in its use
Bit 3	-	-	-	"CommandNotAccepted" Command cannot be executed. A Motion Control command cannot be executed, because necessary preconditions are not met.
Bit 4	-	-	-	"DriveFault" Error in the drive
Bit 5	-	-	-	Reserved
Bit 6	-	-	-	"DynamicError" Specified dynamic values are limited to permissible values.
Bit 7	-	-	-	"CommunicationFault" Communication error Missing or faulty communication.
Bit 8	-	-	-	Reserved
Bit 9	-	-	-	Reserved
Bit 10	-	-	-	Reserved
Bit 11	-	-	-	Reserved
Bit 12	-	-	-	Reserved
Bit 13	-	-	-	"PeripheralError" Error accessing a logical address
Bit 14... Bit 31	-	-	-	Reserved

ErrorDetail tags (speed axis)

The tag structure <TO>.ErrorDetail.<tag name> contains the interrupt number and the effective local interrupt response to the technology interrupt that is currently pending at the technology object.

A list of the technology alarms and interrupt responses can be found in the appendix, Technology alarms (Page 10403).

Tags

Legend (Page 10385)

Tag		Data type	Values	W	Description
ErrorDetail.		STRUCT			
	Number	UDINT	-	RON	Alarm number
	Reaction	DINT	0 to 4	RON	Effective alarm response 0: No response 1: Stop with current dynamic values 2 Stop with maximum dynamic values 3: Stop with emergency stop ramp 4: Remove enable For the external encoder technology object, only alarm responses 0 and 4 are possible.

WarningWord tag (speed axis)

The <TO>.WarningWord tag indicates pending warnings at the technology object.

Notes on the evaluation of the individual bits (e.g. bit 11, "FollowingErrorWarning") can be found in the Evaluating StatusWord, ErrorWord and WarningWord (Page 10317) section.

Tag

Legend (Page 10385)

Tag	Data type	Values	W	Description
WarningWord	DWORD	-	RON	
Bit 0	-	-	-	Reserved
Bit 1	-	-	-	"ConfigurationFault" Configuration error One or several configuration parameters are adjusted internally.
Bit 2	-	-	-	Reserved
Bit 3	-	-	-	"CommandNotAccepted" Command cannot be executed. A Motion Control command cannot be executed, because necessary preconditions are not met.
Bit 4	-	-	-	Reserved
Bit 5	-	-	-	Reserved
Bit 6	-	-	-	"DynamicError" Specified dynamic values are limited to permissible values.

Tag	Data type	Values	W	Description
Bit 7	-	-	-	Reserved
Bit 8	-	-	-	Reserved
Bit 9	-	-	-	Reserved
Bit 10	-	-	-	Reserved
Bit 11	-	-	-	Reserved
Bit 12	-	-	-	Reserved
Bit 13	-	-	-	Reserved
Bit 14... Bit 31	-	-	-	Reserved

ControlPanel tags (speed axis)

The tag structure <TO>.ControlPanel.<tag name> contains no application-relevant data. This tag structure is internally used.

One exception is the tag <TO>.ControlPanel.Input.TimeOut for adapting the sign-of-life monitoring.

Tags

Legend (Page 10385)

Tag				Data type	Values	W	Description
ControlPanel.				STRUCT			
	Input.			STRUCT			
		Command.		ARRAY [1..2] OF STRUCT			
			ReqCounter	UDINT	-	DIR	-
			Type	UDINT	-	DIR	-
			Position	LREAL	-	DIR	-
			Velocity	LREAL	-	DIR	-
			Acceleration	LREAL	-	DIR	-
			Deceleration	LREAL	-	DIR	-
			Jerk	LREAL	-	DIR	-
			Param	LREAL	-	DIR	-
		TimeOut		LREAL	-	DIR	Sign-of-life monitoring Preset value: 2000 ms
		EsLifeSign		UDINT	-	DIR	-
	Output.			STRUCT			
		Command.		ARRAY [1..2] OF STRUCT			
			AckCounter	UDINT	-	RON	-

Tag				Data type	Values	W	Description
			Error	BOOL	-	RON	-
			ErrorID	UDINT	-	RON	-
			Done	BOOL	-	RON	-
			Aborted	BOOL	-	RON	-
		RTLifeSign		UDINT	-	RON	-

InternalTOTrace tags (speed axis)

The tag structure <TO>.InternalTOTrace.<tag name> contains no application-relevant data. This tag structure is internally used.

Tags

Legend (Page 10385)

Tag		Data type	Values	W	Description
InternalTOTrace.		ARRAY [1..4] OF STRUCT			
	Id	DINT	-	DIR	-
	Value	LREAL	-	RON	-

Tags of the technology object external encoder

Legend

Tag	Name of the tag	
Data type	Data type of the tag	
Values	Value range of the tag - minimum value to maximum value If no specific value is shown, the value range limits of the relevant data type apply or the information under "Description".	
W	Effectiveness of changes in the technology data block	
	DIR	Immediate: Changes to the value are made with a direct assignment and are immediately effective.
	RES	Restart: Changes to the start value in load memory are made using the extended instruction "WRIT_DBL" (write to DB in load memory). Changes will not take effect until after restart of the technology object.
	RON	Read only: The tag cannot and must not be changed.
Description	Description of the tag	

L - Linear specification R - Rotary specification

Access to the tags is with "<TO>.<tag name>". The placeholder <TO> represents the name of the technology object.

Actual values and setpoints (external encoder)

The following tags indicate the setpoint and actual values of the technology object.

Tags

Legend (Page 10394)

Tag	Data type	Values	W	Description
ActualPosition	LREAL	-	RON	Actual position
ActualVelocity	LREAL	-	RON	Actual velocity
ActualAcceleration	LREAL	-	RON	Actual acceleration

Sensor tags (external encoder)

The tag structure <TO>.Sensor.<tag name> contains the CPU-side configuration for the encoder, and the configuration for active and passive homing.

Tags

Legend (Page 10394)

Tag			Data type	Values	W	Description
Sensor.			STRUCT			
	Type		DINT	-	RES	Encoder type 0: Incremental 1: Absolute 2: Cyclically absolute
	InverseDirection		BOOL	-	RES	Inversion of the actual value FALSE: no TRUE: yes
	System		DINT	-	RES	Encoder system 0: Linear encoder 1: Rotary encoder
	MountingMode		DINT	-	RES	Mounting type of the encoder 0: Incremental 1: Absolute 2: External
	Interface.					
		AddressIn	UDINT	0 to 65535	RON	Input address for the PROFIdrive frame
		AddressOut	UDINT	0 to 65535	RON	Output address for the PROFIdrive frame

Tag			Data type	Values	W	Description
		Telegram	UDINT	-	RON	PROFIdrive frame of the encoder 3: Actual values via drive frame (NSET (32 bit), NACT (32 bit), actual encoder value, signs of life) 5: Actual values from drive frame (NSETP (32-bit), NACT (32-bit), actual encoder value, sign-of-life, DSC) 81: Actual values via your own encoder telegram (actual encoder value, signs of life) 83: Actual values via your own encoder telegram (NACT (32 bit), actual encoder value, signs of life) For the technology object of external encoders, only frames 81 and 83 are valid. (NSET - setpoint speed, NACT - actual speed value)
	Parameter.					
		Resolution	LREAL	1.0E-12 to 1.0E12	RES	Resolution of a linear encoder (offset between two encoder pulses)
		StepsPerRevolution	UDINT	1 to 8388608	RES	Increments per rotary encoder revolution
		FineResolutionXist1	UDINT	0 to 31	RES	Number of bits for fine resolution Gn_XIST1 (cyclic actual encoder value)
		FineResolutionXist2	UDINT	0 to 31	RES	Number of bits for fine resolution Gn_XIST2 (absolute value of the encoder)
		DeterminableRevolutions	UDINT	0 to 8388608	RES	Number of differentiable encoder revolutions for a multi-turn absolute value encoder (For a single-turn absolute value encoder = 1; for an incremental encoder = 0)
		DistancePerRevolution	LREAL	0.0 to 1.0E12	RES	Load distance per revolution of an externally mounted encoder
	PassiveHoming.		STRUCT			
		Mode	DINT	-	RES	Homing mode 0: Homing mark only 1: Homing mark and proximity switch 2: Digital input
		SideInput	BOOL	-	DIR	Side of the digital input for passive homing: FALSE: Negative side TRUE: Positive side
		Direction	DINT	-	DIR	Homing direction / homing mark approach direction 0: Current homing direction 1: Positive homing direction 2: Negative homing direction
		DigitalInputAddress	UDINT	0 to 65535	RON	Byte number of the digital input
		DigitalInputBitNumber	UDINT	0 to 7	RON	Bit number of the digital input

Mechanics tags (external encoder)

The tag structure <TO>.Mechanics.<tag name> contains the configuration of the mechanics.

Tags

Legend (Page 10394)

Tag		Data type	Value range	W	Description
Mechanics.					
	LeadScrew	LREAL	0.0 to 1.0E12	RES	Leadscrew pitch

LoadGear tags (external encoder)

The tag structure <TO>.LoadGear.<tag name> contains the configuration for the load gear.

Tags

Legend (Page 10394)

Tag		Data type	Value range	W	Description
LoadGear.		STRUCT			
	Numerator	UDINT	1 to 4294967295	RES	Load gear counter
	Denominator	UDINT	1 to 4294967295	RES	Load gear denominator

Properties tags (external encoder)

The tag structure <TO>.Properties.<tag name> contains the configuration for the axis or type of motion.

Tags

Legend (Page 10394)

Tag		Data type	Value range	W	Description
Properties.					
	MotionType	DINT	-	RON	Indication of axis or motion type: 0: Linear axis or motion 1: Rotary axis or motion

Modulo tags (external encoder)

The tag structure <TO>.Modulo.<tag name> contains the modulo configuration.

Tags

Legend (Page 10394)

Tag		Data type	Values	W	Description
Modulo.		STRUCT			
	Enable	BOOL	-	RES	FALSE: Modulo conversion disabled TRUE: Modulo conversion enabled When modulo conversion is enabled, a check is made for modulo length > 0.0
	Length	LREAL	0.001 to 1.0E12	RES	Modulo length
	StartValue	LREAL	-1.0E12 to 1.0E12	RES	Modulo start value

Homing tags (external encoder)

The tag structure <TO>.Homing.<tag name> contains the configuration for homing the TO.

Tags

Legend (Page 10394)

Tag		Data type	Values	W	Description
Homing.		STRUCT			
	HomePosition	LREAL	-1.0E12 to 1.0E12	DIR	Home position

StatusSensor tags (external encoder)

The tag structure <TO>.StatusSensor.<tag name> indicates the status of the measuring system.

Tags

Legend (Page 10394)

Tag		Data type	Values	W	Description
StatusSensor.		STRUCT			
	State	DINT	-	RON	Status of the actual encoder value: 0: "NOT_VALID (not valid) 1: "Waiting_FOR_VALID (waiting for valid status) 2: "VALID (valid)
	CommunicationOK	BOOL	-	RON	Cyclic BUS communication between controller and encoder FALSE: Not established TRUE: Established

Tag		Data type	Values	W	Description
	Error	BOOL	-	RON	FALSE: No error in the measuring system TRUE: Error in the measuring system.
	AbsEncoderOffset	LREAL	-	RON	Home point offset to the value of an absolute value encoder. The value will be retentively stored in the CPU.

StatusWord tag (external encoder)

The <TO>.StatusWord tag contains the status information of the technology object.

Notes on the evaluation of the individual bits (e.g. bit 5 "HomingDone") can be found in the section Evaluating StatusWord, ErrorWord and WarningWord (Page 10317) section.

Tag

Legend (Page 10394)

Tag	Data type	Values	W	Description
Status Word	DWORD	-	RON	Status data of the technology object
Bit 0	-	-	-	"Enable" Enable status 0: Technology object disabled 1: Technology object enabled
Bit 1	-	-	-	"Error" 0: No error present 1: Error present
Bit 2	-	-	-	"RestartActive" 0: No "Restart" active 1: "Restart" active. The technology object is being reinitialized.
Bit 3	-	-	-	"OnlineStartValuesChanged" 0: "Restart" tag unchanged 1: Change to "Restart" tag. For the changes to be applied, the technology object must be reinitialized.
Bit 4	-	-	-	Reserved
Bit 5	-	-	-	"HomingDone" Homing status 0: Technology object not homed 1: Technology object homed
Bit 6	-	-	-	"Done" 0: Motion job in process, or axis control panel enabled 1: No motion job in process and axis control panel disabled
Bit 7	-	-	-	Reserved
Bit 8	-		-	Reserved
Bit 9	-	-	-	Reserved
Bit 10	-	-	-	Reserved

Tag	Data type	Values	W	Description
Bit 11	-	-	-	"HomingCommand" 0: No "MC_Home" job in process 1: "MC_Home" job in process
Bit 12	-	-	-	Reserved
Bit 13	-	-	-	Reserved
Bit 14	-	-	-	Reserved
Bit 15	-	-	-	Reserved
Bit 16	-	-	-	Reserved
Bit 17	-	-	-	Reserved
Bit 18	-	-	-	Reserved
Bit 19	-	-	-	Reserved
Bit 20	-	-	-	Reserved
Bit 21... Bit 31	-	-	-	Reserved

ErrorWord tag (external encoder)

The <TO>.ErrorWord tag indicates errors at the technology object (technology alarms).

Notes on the evaluation of the individual bits (e.g. bit 3 "CommandNotAccepted") can be found in the section Evaluating StatusWord, ErrorWord and WarningWord (Page 10317).

Tag

Legend (Page 10394)

Tag	Data type	Values	W	Description
Error Word	DWORD	-	RON	
Bit 0	-	-	-	"SystemFault" System error
Bit 1	-	-	-	"ConfigurationFault" Configuration error One or more configuration parameters are inconsistent or invalid.
Bit 2	-	-	-	"UserFault" Error in the user program in a motion control instruction, or in its use
Bit 3	-	-	-	"CommandNotAccepted" Command cannot be executed. A Motion Control command cannot be executed, because necessary preconditions are not met.
Bit 4	-	-	-	Reserved
Bit 5	-	-	-	"SensorFault" Error in the encoder system.
Bit 6	-	-	-	Reserved

Tag	Data type	Values	W	Description
Bit 7	-	-	-	"CommunicationFault" Communication error Missing or faulty communication.
Bit 8	-	-	-	Reserved
Bit 9	-	-	-	Reserved
Bit 10	-	-	-	"HomingFault" Error during homing process The homing cannot be completed.
Bit 11	-	-	-	Reserved
Bit 12	-	-	-	Reserved
Bit 13	-	-	-	"PeripheralError" Error accessing a logical address
Bit 14... Bit 31	-	-	-	Reserved

ErrorDetail tags (external encoder)

The tag structure <TO>.ErrorDetail.<tag name> contains the interrupt number and the effective local interrupt response to the technology interrupt that is currently pending at the technology object.

A list of the technology alarms and interrupt responses can be found in the appendix, Technology alarms (Page 10403).

Tags

Legend (Page 10394)

Tag		Data type	Values	W	Description
ErrorDetail.		STRUCT			
	Number	UDINT	-	RON	Alarm number
	Reaction	DINT	0 to 4	RON	Effective alarm response 0: No response 1: Stop with current dynamic values 2 Stop with maximum dynamic values 3: Stop with emergency stop ramp 4: Remove enable For the external encoder technology object, only alarm responses 0 and 4 are possible.

WarningWord tag (external encoder)

The <TO>.WarningWord tag indicates pending warnings at the technology object.

Notes on the evaluation of the individual bits (e.g. bit 11, "FollowingErrorWarning") can be found in the Evaluating StatusWord, ErrorWord and WarningWord (Page 10317) section.

Tag

Legend (Page 10394)

Tag	Data type	Values	W	Description
WarningWord	DWORD	-	RON	
Bit 0	-	-	-	Reserved
Bit 1	-	-	-	"ConfigurationFault" Configuration error One or several configuration parameters are adjusted internally.
Bit 2	-	-	-	Reserved
Bit 3	-	-	-	"CommandNotAccepted" Command cannot be executed. A Motion Control command cannot be executed, because necessary preconditions are not met.
Bit 4	-	-	-	Reserved
Bit 5	-	-	-	Reserved
Bit 6	-	-	-	Reserved
Bit 7	-	-	-	Reserved
Bit 8	-	-	-	Reserved
Bit 9	-	-	-	Reserved
Bit 10	-	-	-	Reserved
Bit 11	-	-	-	Reserved
Bit 12	-	-	-	Reserved
Bit 13	-	-	-	Reserved
Bit 14... Bit 31	-	-	-	Reserved

InternalTOTrace tags (external encoder)

The tag structure <TO>.InternalTOTrace.<tag name> contains no application-relevant data. This tag structure is internally used.

Tags

Legend (Page 10394)

Tag		Data type	Values	W	Description
InternalTOTrace.		ARRAY [1..4] OF STRUCT			
	Id	DINT	-	DIR	-
	Value	LREAL	-	RON	-

Technology alarms

The following table shows an overview of the technology alarms and the corresponding alarm responses. When a technology alarm occurs, evaluate the entire indicated alarm text, in order to find the precise cause.

You can find the FAQs with remedies for technology alarms on the Internet (http://support.automation.siemens.com/WW/view/en/66958052).

Legend

No.	Number of the technology alarm (corresponds to <TO>.ErrorDetail.Number)
Response	Effective alarm response (corresponds to <TO>.ErrorDetail.Reaction)
Error bit	Bit that is set in <TO>.ErrorWord when the technology alarm occurs
Warning bit	Bit that is set in <TO>.WarningWord when the technology alarm occurs
Restart	To acknowledge the technology alarm, the technology object must be reinitialized (restart).
Alarm text	Indicated alarm test (limited)

List of the technology alarms

No.	Response	Error bit	Warning bit	Restart	Alarm text
101	Remove enable	ConfigurationFault	-	X	Error in the configuration.
104	Stop with maximum dynamic values	ConfigurationFault	-	-	Error in the specification of software limit switches,
105	Remove enable	ConfigurationFault	-	X	Configuration mode (drive/actuator).
106	Remove enable	ConfigurationFault	-	X	Configuration error in the drive actuator driver.
107	Remove enable	ConfigurationFault	-	X	Configuration error encoder.
108	Remove enable	ConfigurationFault	-	X	Configuration error encoder driver.
109	Remove enable	ConfigurationFault	-	-	Configuration error.
110	No response	-	ConfigurationFault	-	Configuration is being internally adapted.
201	Remove enable	SystemFault	-	-	Internal error.
202	No response	SystemFault	-	-	Internal configuration error.
203	Remove enable	SystemFault	-	-	Internal error. Please contact customer service.

No.	Response	Error bit	Warning bit	Restart	Alarm text
204	Remove enable	SystemFault	-	-	Commissioning error.
304	Stop with emergency stop ramp	UserFault	-	-	Limit value for velocity is zero.
305	Stop with emergency stop ramp	UserFault	-	-	Limit value for acceleration/deceleration is zero.
306	Stop with emergency stop ramp	UserFault	-	-	Limit value for jerk is zero.
307	Stop with maximum dynamic values	UserFault	-	-	The numeric value range of the position was reached.
308	Remove enable	UserFault	-	-	The numeric value range of the position was exceeded.
321	Stop with emergency stop ramp	CommandNot Accepted	-	-	Axis is not homed.
322	No response	-	CommandNot Accepted	-	Restart not performed.
323	No response	-	CommandNot Accepted	-	MC_Home was not executed because this would result in the numeric value range being exceeded.
341	Remove enable	HomingFault	-	-	Error in homing data.
342	Stop with emergency stop ramp	HomingFault	-	-	Reference cam / encoder zero mark was not found.
401	Remove enable	PeripheralError	-	-	Error accessing logical address.
411	Remove enable	SensorFault	-	-	Encoder at the logical address disrupted.
412	Remove enable	SensorFault	-	-	Violation of the range limit for the incremental actual value.
421	Remove enable	DriveFault	-	-	Drive/actuator at the logical address disrupted.
431	Remove enable	Communication Fault	-	-	Communication with device disrupted.
501	No response	-	DynamicError	-	Programmed velocity limited.
502	No response	-	DynamicError	-	Programmed acceleration/deceleration limited.
503	No response	-	DynamicError	-	Programmed jerk limited.
504	No response	-	DynamicError	-	Speed setpoint monitoring active.

No.	Response	Error bit	Warning bit	Restart	Alarm text
521	Remove enable	FollowingError Fault	-	-	Window of the following error monitoring exceeded.
522	No response		FollowingError Fault		Warning level of the following error monitoring exceeded
531	Remove enable	HWLimit	-	-	• Positive HW limit switch reached. • Negative HW limit switch reached. • Incorrect traverse direction with active hardware limit switch
531	Remove enable	HWLimit	-	X	• Hardware limit switch polarity is reversed, axis cannot be freed • Both hardware limit switches are active, axis cannot be freed
533	Stop with maximum dynamic values	SWLimit	-	-	Software limit switch reached.
534	Remove enable	SWLimit	-	-	Software limit switch passed.
541	Remove enable	PostioningFault	-	-	Positioning monitoring error.

See also

Technology alarms (Page 10350)

Error ID

Errors in Motion Control instructions are reported by means of "Error" and "ErrorID" parameters.

Under the following conditions, "Error" = TRUE and "ErrorID" = 16#8xxx are indicated at the Motion Control instruction:

- Invalid status of the technology object, which prevents the execution of the job.
- Invalid parameter assignment of the Motion Control instruction, which prevents the execution of the job.
- As a result of the alarm response for an error at the technology object.

ErrorIDs and remedies

The following table lists all "ErrorIDs" that can be indicated in Motion Control instructions. Along with the cause of the error, the table lists corresponding troubleshooting routines:

ErrorID	Description	Remedy
16#0000	No error	-
16#8001	A technology alarm (TO error) occurred while processing the Motion Control instruction.	In the technology data block, an error message is output at the "ErrorDetail.Number" tag. A list of the technology alarms and alarm responses can be found in the appendix, Technology Alarms (Page 10403).
16#8002	Invalid value in the "Axis" parameter.	Check the value in the "Axis" parameter
16#8003	Invalid velocity	Set a valid velocity value at parameter "Velocity".
16#8004	Invalid acceleration	Set a valid acceleration value at parameter "Acceleration".
16#8005	Invalid deceleration	Set a valid deceleration value at parameter "Deceleration".
16#8006	Invalid jerk	Set a valid jerk value at parameter "Jerk".
16#8007	Invalid direction	Set a valid directional value at parameter "Direction".
	When the axis is at a standstill, "Direction" = 4 (current direction) is not permitted.	
	Invalid input Both the "JogForward" and "JogBackward" parameters set to TRUE at the same time. The axis decelerates with the most recently valid deceleration.	Reset both the "JogForward" parameter and the "JogBackward" parameter.
16#8008	Invalid distance specification	Set a valid distance value at parameter "Distance".
16#8009	Invalid position specification	Set a valid position value at parameter "Position".
16#800A	Invalid operating mode	Set a valid operating mode at parameter "Mode".
16#800B	Invalid stop mode	Set a valid stop mode at parameter "StopMode".
16#800C	Only one instance of the "MC_Power" instruction is allowed per technology object.	A technology object was specified in several instances of the "MC_Power" instruction, in the "Axis" parameter. Ensure that the technology object is only specified in one "MC_Power" instruction.
16#800D	Job not permitted in current mode. "Restart" is being executed.	Wait until the "Restart" of the technology object is complete.
16#800E	If the axis is activated, a "Restart" is not possible.	Before a "Restart", deactivate the technology object with "MC_Power.Enable"FALSE.
16#800F	The technology object is disabled. The job cannot be executed.	Enable the technology object with "MC_Power.Enable" = TRUE. Restart the job.
16#8010	Invalid homing mode for incremental encoder.	Absolute value adjustment is not possible with an incremental encoder ("Mode" = 6, 7). Start a homing process for an incremental encoder using parameter "Mode" = 0, 1, 2, 3, 4, 5, 8.
16#8011	Invalid homing mode for absolute encoder.	Passive and active homing ("Mode" = 2, 3, 4, 5) are not possible for an absolute value encoder. Start a homing process for an absolute encoder using parameter "Mode" = 0, 1, 6, 7.

ErrorID	Description	Remedy
16#8012	Command cannot be executed, because the axis control panel is enabled.	Return master control to the user program. Restart the job.
16#8013	The online connection between the CPU and the TIA Portal is down.	Check the online connection to the CPU.
16#8014	No internal job memory available.	The maximum number of motion control jobs has been reached. Reduce the number of jobs to be executed (parameter "Execute" = FALSE).
16#8015	Error acknowledgement with MC_Reset" not possible. Error in the configuration of the technology object.	Check the configuration of the technology object.
16#8016	The actual values are not valid.	The actual values must be valid in order to execute a MC_Home job. Check the status of the actual values. The " <TO>.StatusSensor[n].State" tag of the technology object must show the value 2 (valid).
16#8FFF	Unspecified error	Please contact the Siemens agent at your local agency or branch. You will find information on whom to contact at: Contact information for Industry Automation and Drive Technologies (http://www.siemens.com/automation/partner)

See also

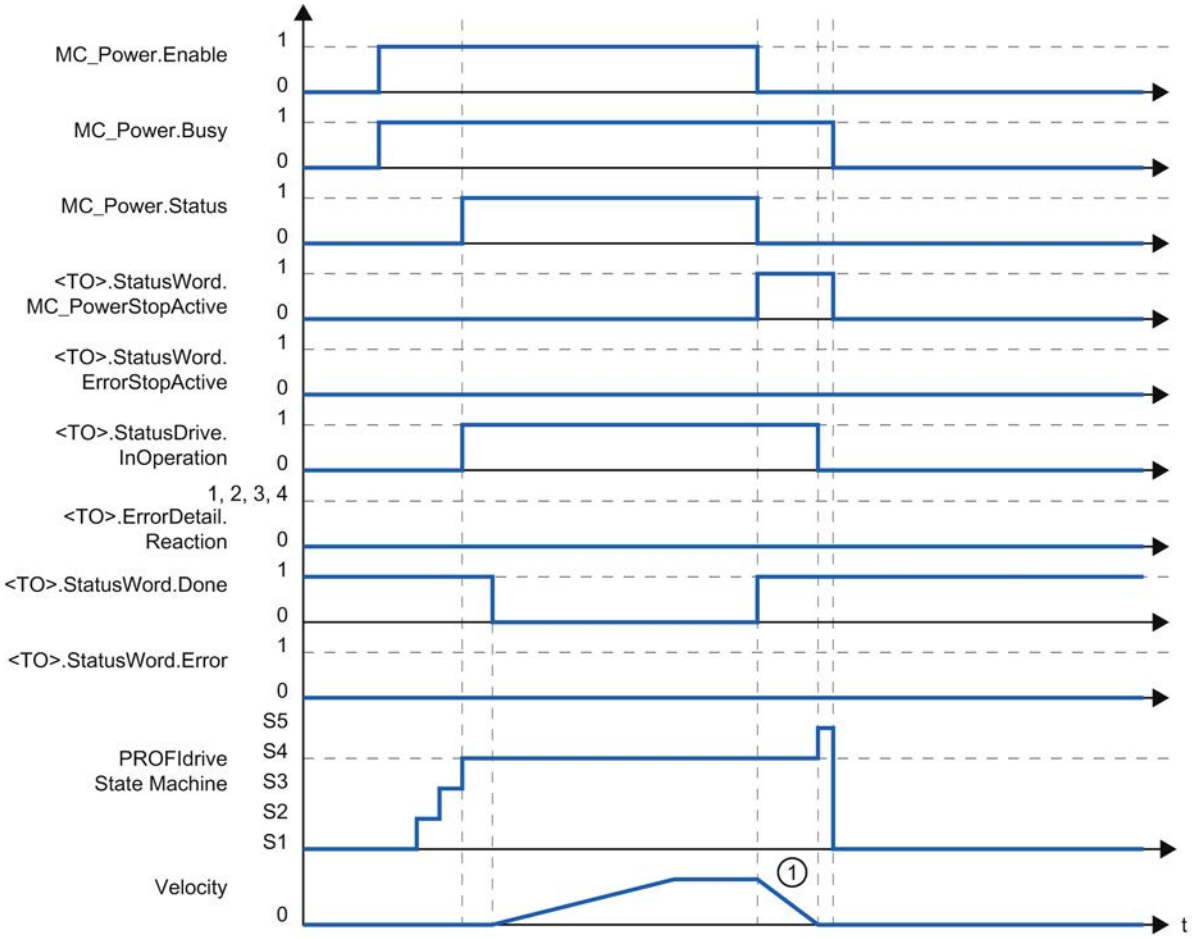
Errors in Motion Control instructions (Page 10353)

MC_Power function chart

Drive connection via PROFIdrive

StopMode 0

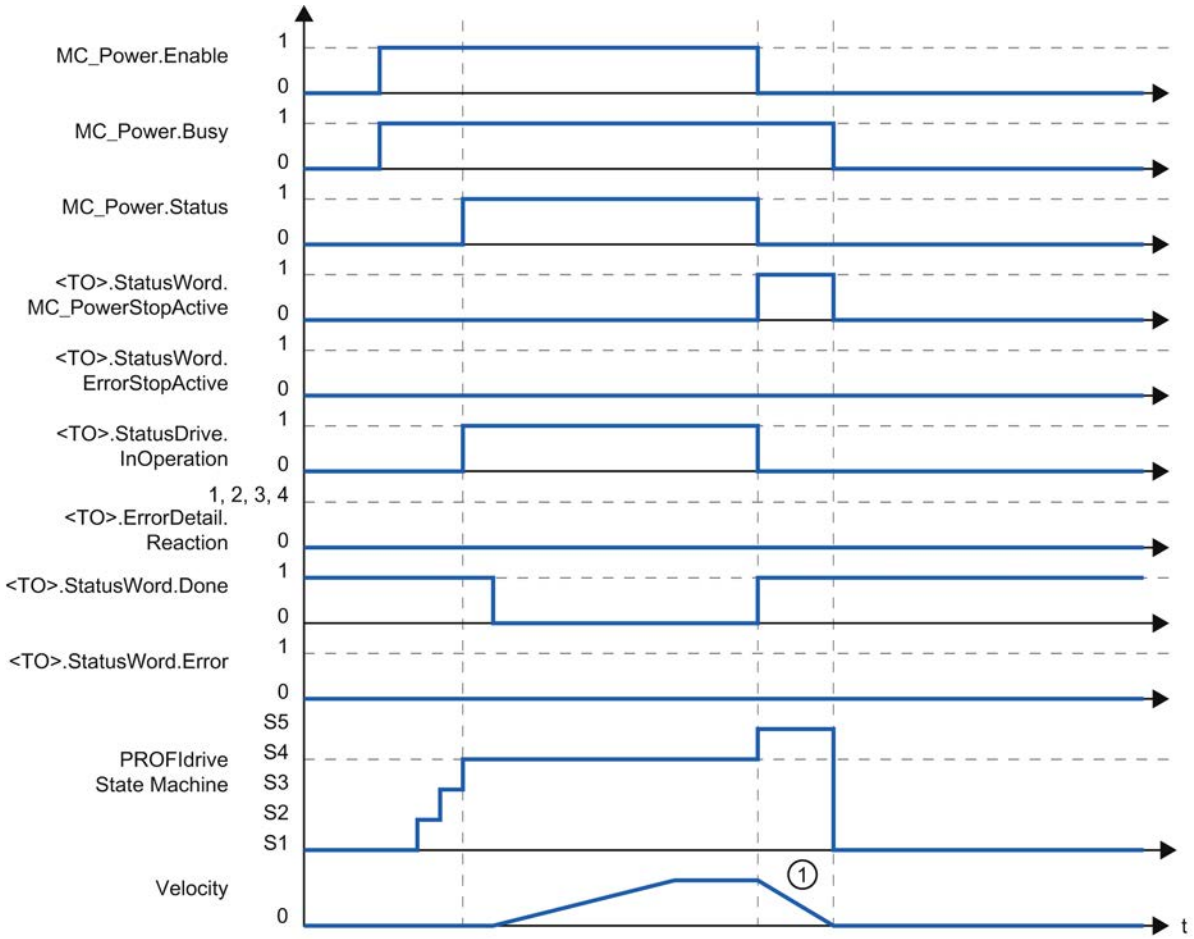
Function chart: Enabling a technology object and disabling with "StopMode" = 0



① The axis decelerates with the configured emergency stop deceleration.

StopMode 1

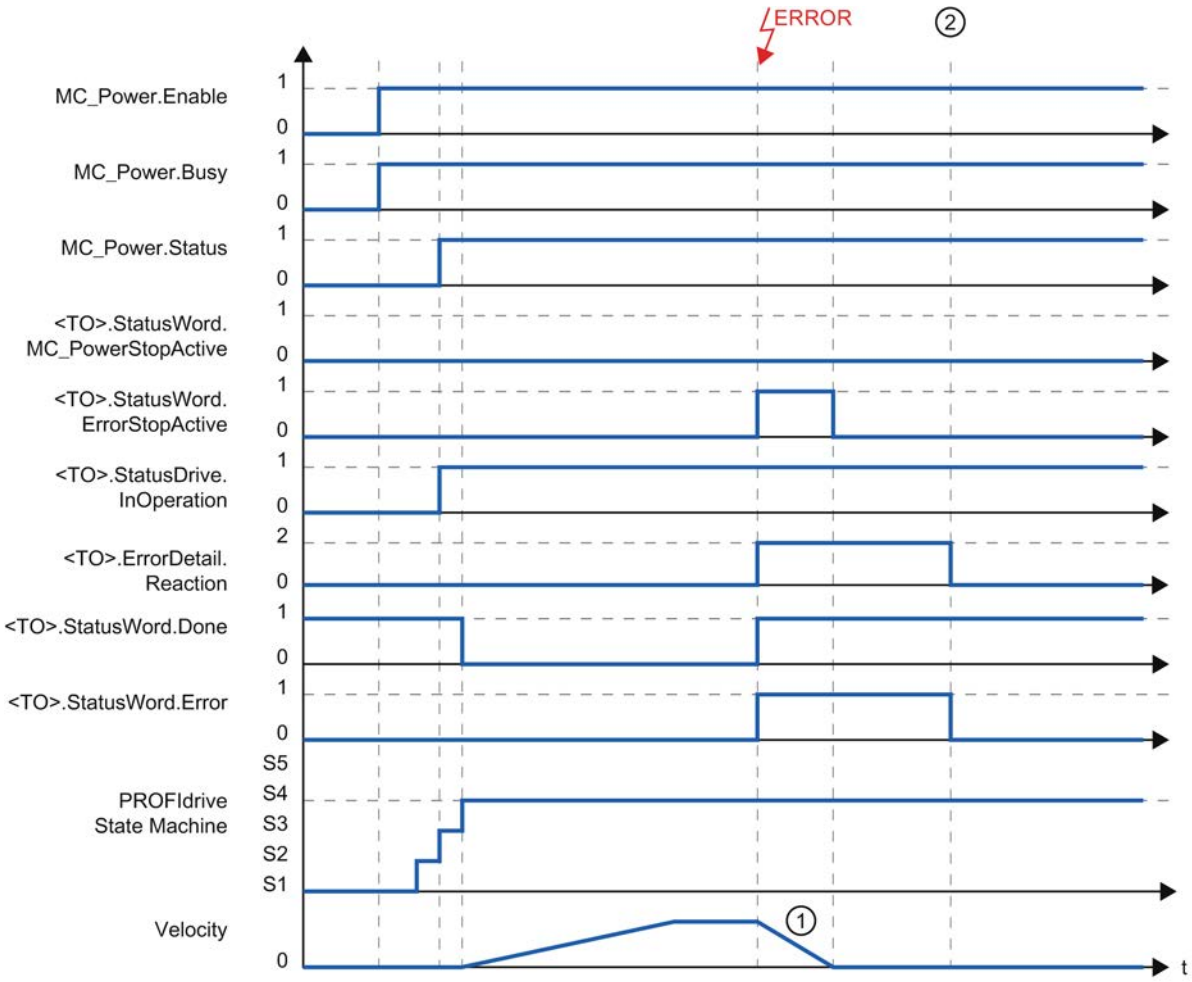
Function chart: Enabling a technology object and disabling with "StopMode" = 1



① The deceleration ramp depends on the configuration in the drive.

Alarm response "Stop with maximum dynamic values"

Function chart: Enabling a technology object and occurrence of a technology alarm with alarm response "Stop with maximum dynamic values"

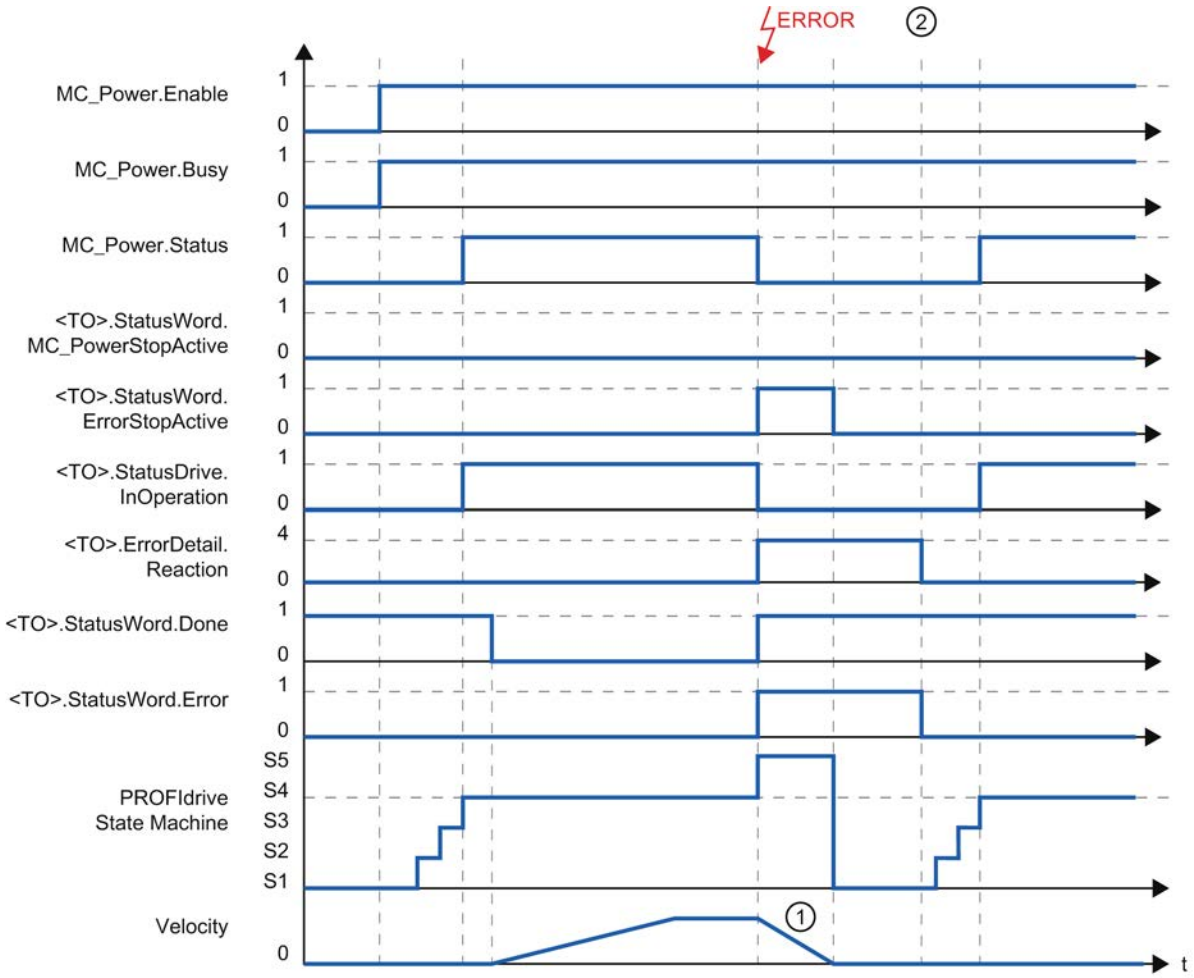


① The axis decelerates with the configured maximum deceleration.

② The technology alarm is acknowledged.

Alarm response "Remove enable"

Function chart: Enabling a technology object and occurrence of a technology alarm with alarm response "Remove enable"



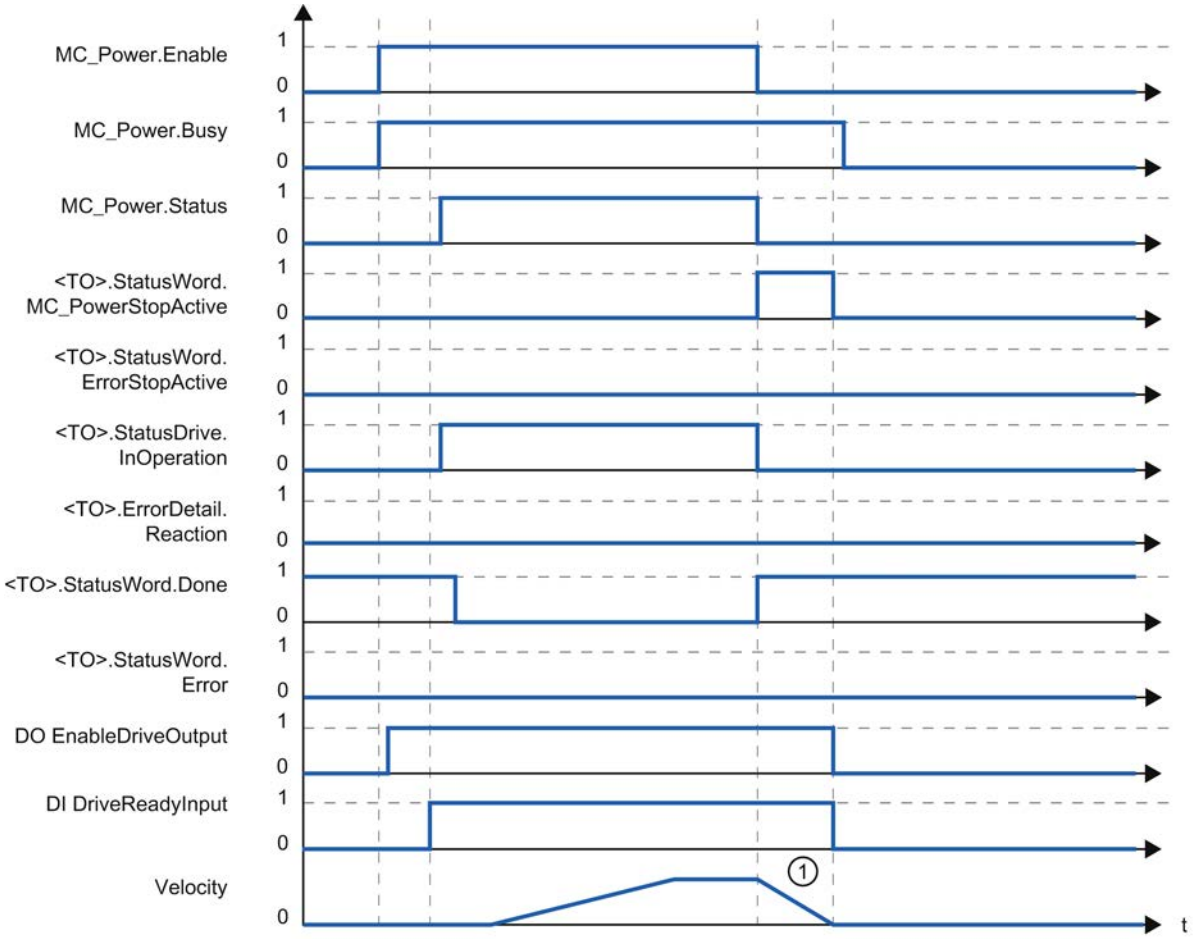
① The deceleration ramp depends on the configuration in the drive.

② The technology alarm is acknowledged at time ②.

Analog drive connection

StopMode 0

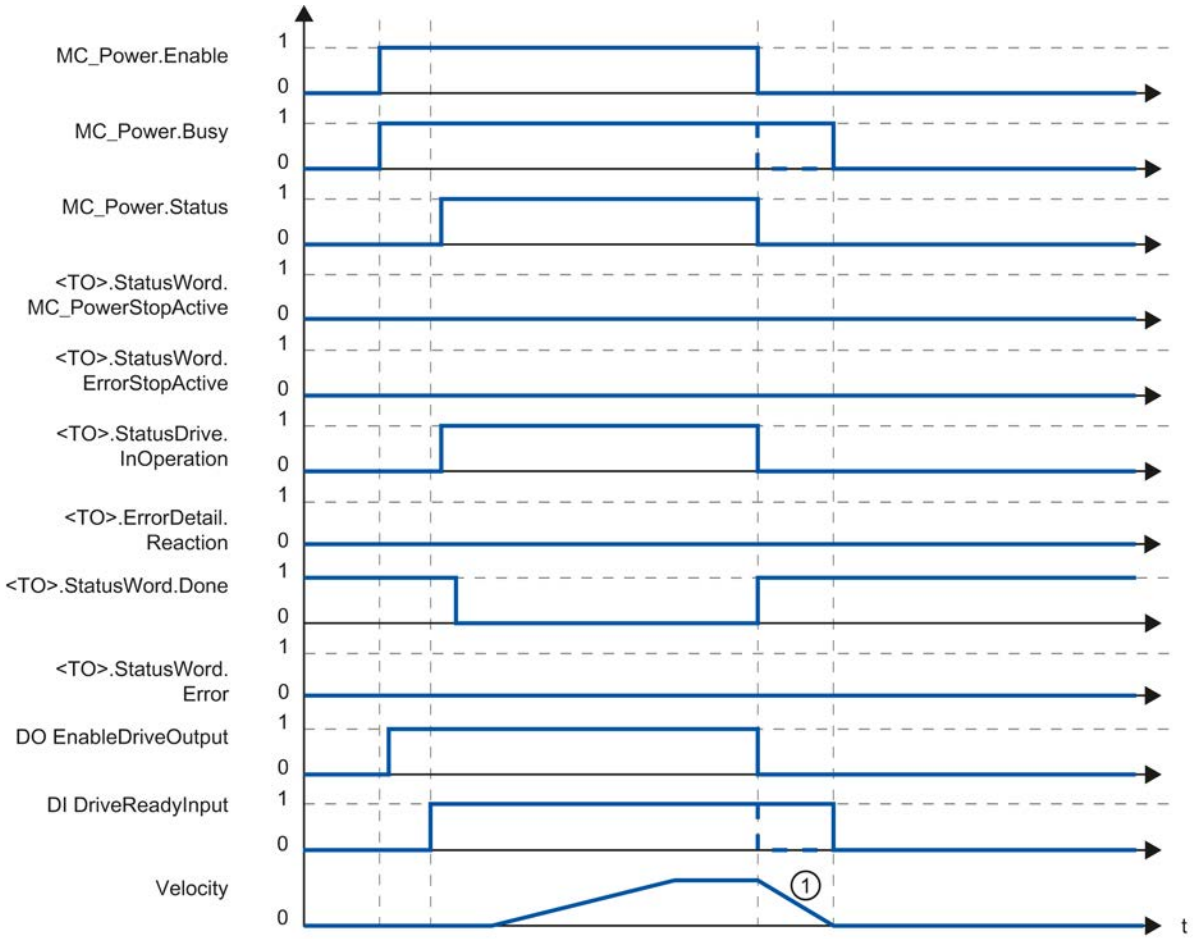
Function chart: Enabling a technology object and disabling with "StopMode" = 0



① The axis decelerates with the configured emergency stop deceleration.

StopMode 1

Function chart: Enabling a technology object and disabling with "StopMode" = 1

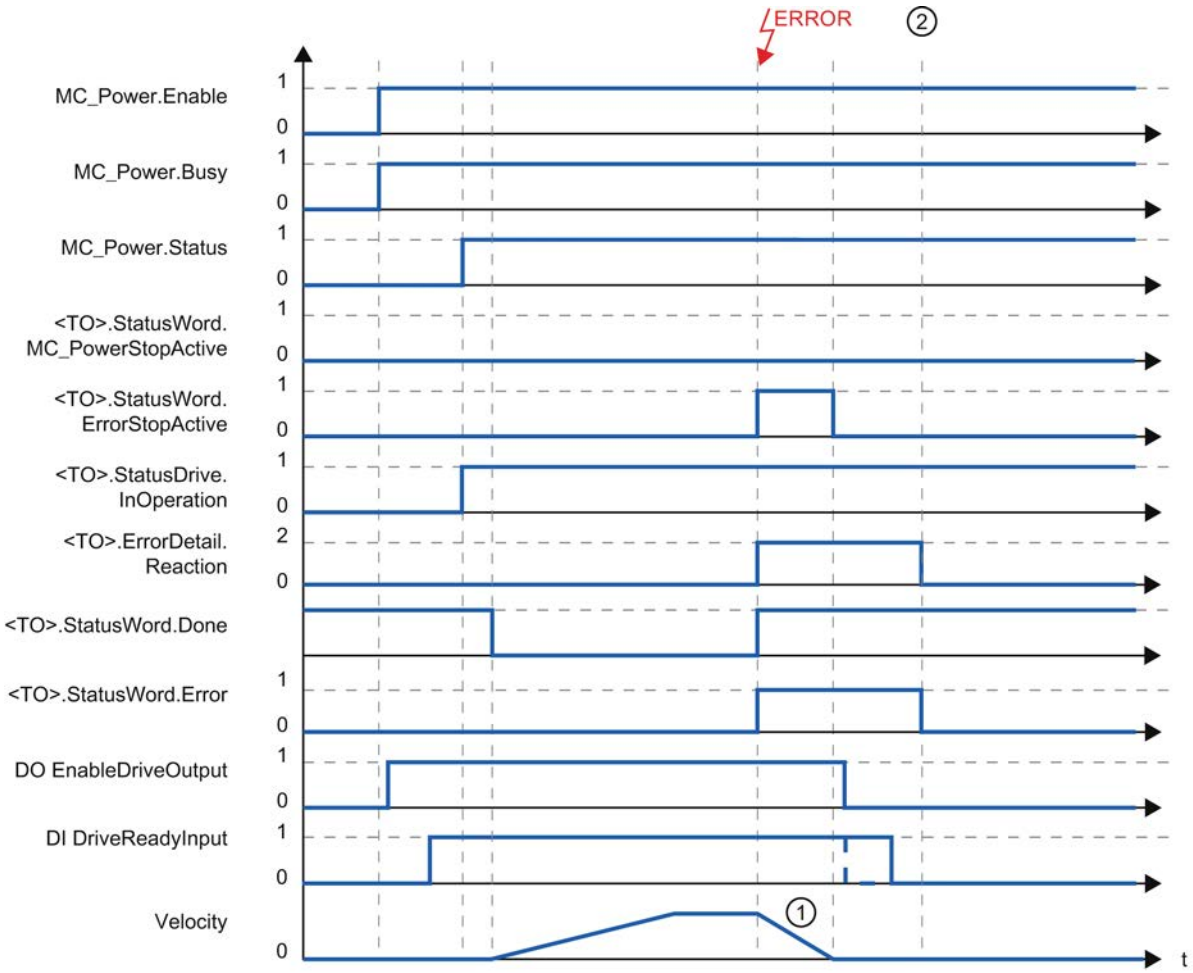


① The deceleration ramp depends on the configuration in the drive.

The behavior of the ready signal of the drive "DI DriveReadyInput" is manufacturer-specific.

Alarm response "Stop with maximum dynamic values"

Function chart: Enabling a technology object and occurrence of a technology alarm with alarm response "Stop with maximum dynamic values"



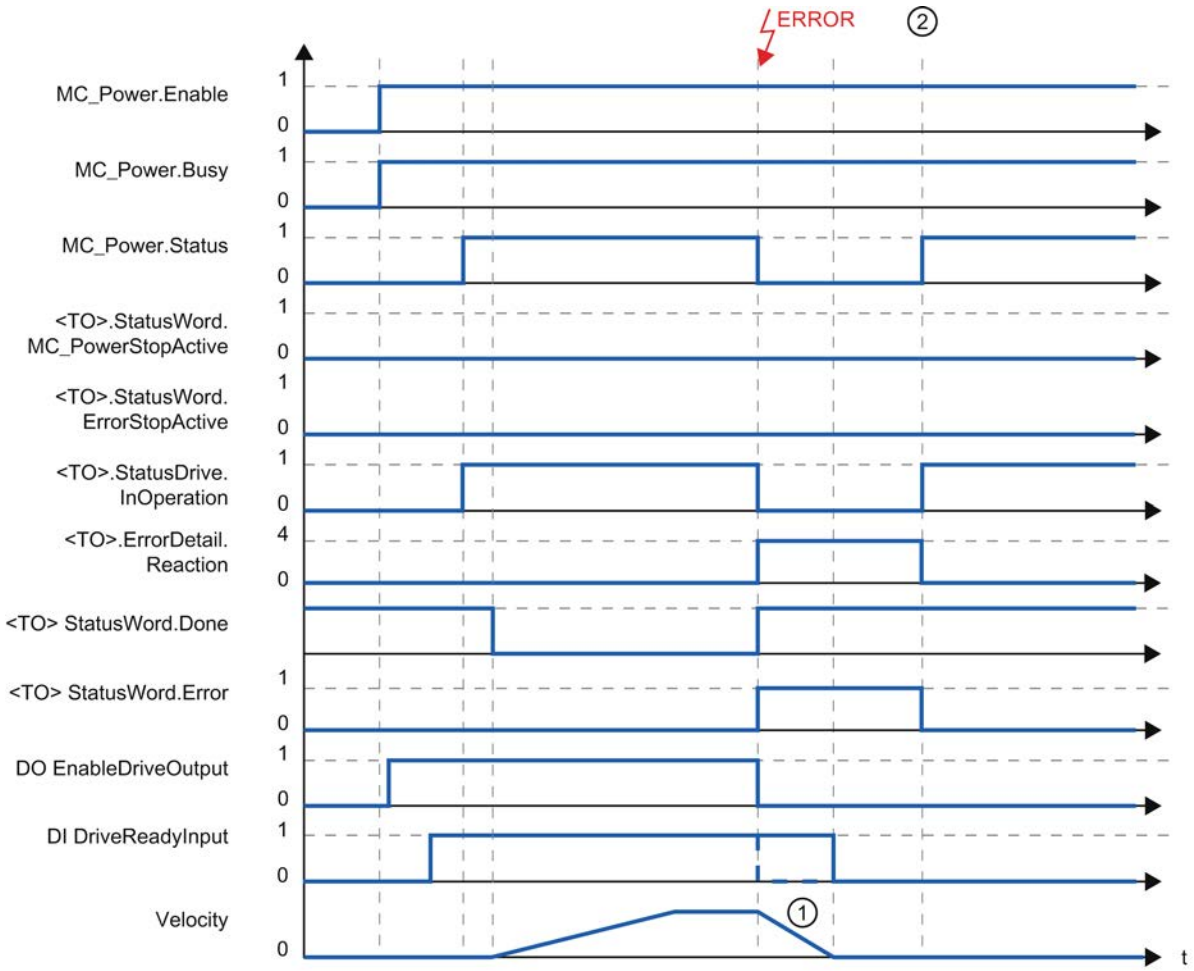
① The axis decelerates with the configured maximum deceleration.

② The technology alarm is acknowledged at time ②.

The behavior of the ready signal of the drive "DI DriveReadyInput" is manufacturer-specific.

Alarm response "Remove enable"

Function chart: Enabling a technology object and occurrence of a technology alarm with alarm response "Remove enable"



① The deceleration ramp depends on the configuration in the drive.

② The technology alarm is acknowledged at time ②.

The behavior of the ready signal of the drive "DI DriveReadyInput" is manufacturer-specific.

SINAMICS drives

Active homing for SINAMICS drives with external zero mark

For SINAMICS drives with external zero mark, synchronization during active homing must always occur on the left side of the external zero mark's signal. That is to say, with a positive direction of travel synchronization is done on a positive edge, and with a negative direction of travel synchronization is done on a negative edge.

By inverting the signal, synchronization can also be done on the right sight of the signal of the external zero mark. The inversion can be configured in the drive using SINAMICS parameter P490.

Homing to an encoder zero mark or an external zero mark is configured in SINAMICS parameter P495.

11.4.2 Motion Control (S7-300, S7-400)

11.4.2.1 Position feedback basics

What is position feedback?

Position feedback

Position feedback is the detection of actual values scaled to units of length.

It differs depending on the encoder used:

- Incremental encoder
  The module counts the incremental encoder signals. A reference point approach is required before position feedback to create a reference point for the encoder position. The counter can, for example, be set to "0" at a reference mark.
  Following the reference point approach (with synchronized axis), the counter value is a measurement of the current position.
- Absolute encoder
  No reference point approach is required. The signal output represents a fixed position.

The values determined can be processed directly in the user program. A direct response to encoder values from motion systems is therefore possible.

Encoder value acquisition

Description

The absolute encoder transfers its encoder values to the module in frames. The transmission of frames is initiated by the module.

- In non-isochrone mode, the encoder values are acquired freewheeling.
- In isochrone mode, the encoder values are acquired in synchronism with the bus cycle at each T_i.

Freewheeling encoder value acquisition

The module initiates the transmission of a frame each time the parameterized monoflop time elapses.

The module processes the acquired encoder value asynchronously to these freewheeling frames in the cycle of the update rate.

Freewheeling acquisition therefore returns encoder values of different ages. The difference between the maximum and minimum age is the jitter.

Isochrone encoder value acquisition

Isochrone encoder value acquisition is automatically set if the bus system and I/O modules are synchronized.

The module initiates a frame transfer in each bus cycle at T_i.

The module processes the transmitted encoder value isochronously to the bus cycle.

Gray/dual converter

Gray code/dual code converter

The "Gray code/dual code converter allows you to define how the codes returned by the encoder are processed:

- When "Gray code" is set, the encoder value returned by the absolute encoder in gray code will be converted to dual code.
- When "Dual code" is set, the values returned by the encoder will remain unchanged.

Note

If you have selected the "Gray code" setting, the module will always convert the total encoder value (13 to 25 bits). Any leading special bits will therefore influence the encoder value and the appended bits may be corrupted.

Encoder value and scaling

Encoder value and scaling

The encoder value transferred contains the encoder position of the absolute encoder. In addition to the encoder position, the encoder transfers additional bits located before and after the encoder position, depending on the encoder used.

The module requires the following information if it is to determine the encoder position:

- Encoder type
- Number of adjusted bits
- Increments per encoder revolution

Scaling specifies the position of the encoder value at the feedback interface.

- You specify that trailing, irrelevant bits in the encoder value are to be discounted by entering either "Number of adjusted bits = 1, 2 ... 12" (for SM 338) or "Scaling on" (for ET 200S 1SSI). The encoder value is then positioned in the address area, right-aligned (see example below).
- Entering "Number of adjusted bits = 0" (for SM 338) or "Scaling of" (for ET 200S 1SSI) will mean adjusted bits are retained and available for evaluation.
  This can be useful is you are using an absolute encoder which transfers information in the adjusted bits (see manufacturer's specifications) and you wish to evaluate this information. Please also read the Gray code/dual code converter (Page 10417) information.

Example of encoder value scaling

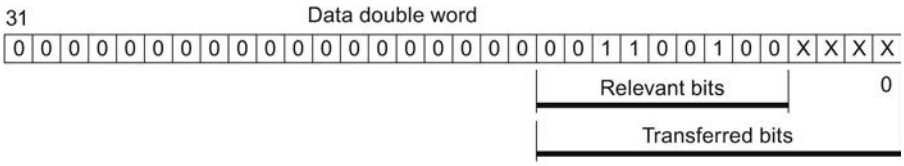
Requirements:

You are using a single-turn encoder with 2^9 steps (corresponds to 9 bits) = 512 increments/ encoder revolution (resolution/360°) with the following parameter assignment:

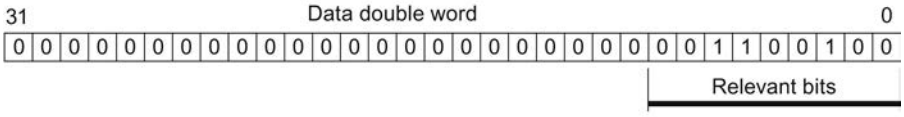
- Encoder type: SSI-13 bit
- Number of adjusted bits: 4 places
- Increments per encoder revolution: 512

The following image shows the location of the relevant bits before and after scaling.

Prior to normalizing: cyclically measured encoder value 100



After normalizing: Encoder value 100



Result: bits 0 to 3 (4 digits, marked with "x") were omitted.

11.4.2.2 Using SM 338

Terminal assignment of the front connector of SM 338

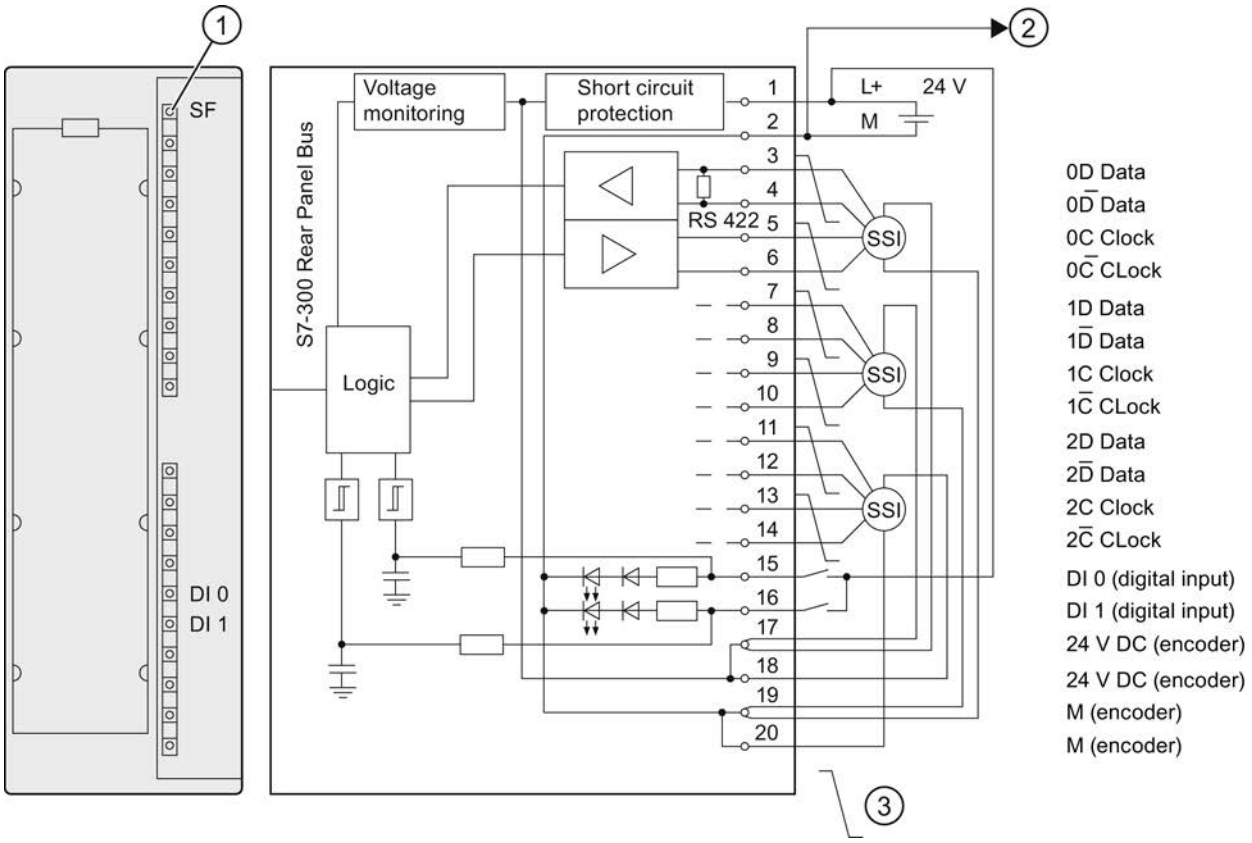
Wiring rules

Important rules to observe when wiring the module:

- Ground of the encoder supply is connected to CPU ground potential. Thus, establish a low impedance connection between pin 2 of SM 338 (M) and CPU ground.
- Always use shielded twisted-pair cable to wire encoder signals (pins 3 to 14.) Connect both ends of the shielding.
  You use the shield contact element for the shield contact at the SM 338 (order number: 6ES7390-5AA00-0AA0).
- Connect an external power supply if the maximum output current (900 mA) of the encoder supply is exceeded.

Terminal assignment of the front connector and block diagram

The figure below shows the terminal assignment of the front connector of the SM 338 and the block diagram.



① Error LED - red
② Connection to CPU ground
③ Twisted-pair cables

Configuring SM 338

Assigning parameters for SM 338

Operating mode

Enabling Fast Mode

Fast Mode

Fast Mode is enabled in the respective check box.

In Fast Mode, the module operates with faster encoder value detection and compressed feedback interface. The Freeze function is not possible.

Isochronous mode

Hardware requirements

To operate SM 338 in isochronous mode, you need:

- a CPU which supports isochronous mode
- a DP master which supports the constant bus cycle
- Interface module IM 153-x, which supports isochronous mode

Properties of the SM 338 in isochronous mode

SM 338 operates in non-isochronous or isochronous mode, depending on system parameters.

In isochronous mode, data are exchanged between the DP master and SM 338 in synchronism to the PROFIBUS DP cycle.

In isochronous mode, all 16 bytes of the feedback interface are consistent.

If synchronism is lost due to disturbance or failure/delay of Global Control (GC), the SM 338 resumes isochronous mode at the next cycle without error reaction.

If synchronism is lost, the feedback interface is not updated.

Note

For basic information on isochronous mode, refer to the Isochronous mode function manual (http://support.automation.siemens.com/WW/view/en/15218045).

Enable

Enabling diagnostic interrupt

Diagnostic interrupt

The diagnostic interrupts are not preset - in other words, they are inhibited without appropriate enabling.

Diagnostic interrupts are enabled in the respective check box.

If you have enabled diagnostic interrupts, the incoming error events (initial occurrence of the error) and outgoing error events (message after troubleshooting) are reported by means of interrupts (Page 10427).

The CPU interrupts user program execution, and executes diagnostic interrupt OB.

The diagnostic information is consistent until such time as diagnostic interrupt OB is exited. When diagnostic interrupt OB is exited, the diagnostic interrupt is acknowledged on the module.

Encoder input n

Selecting the encoder type

Encoder type

The SM 338 supports the following encoder types:

- Absolute value encoder (SSI), frame length 13 bit
- Absolute value encoder (SSI), frame length 21 bit
- Absolute value encoder (SSI), frame length 25 bit

The encoder type can be found in the technical data of the absolute value encoder used.

The selection "No encoder" switches off the encoder input.

Selecting the code type

Code type

The parameter "Code type" is used to determine the handling of the code supplied by the encoder:

- When "Gray" is set, the encoder value supplied by the absolute encoder in gray code is converted to dual code.
- When "Binary" is set, the supplied encoder value is not converted.

Selecting the transmission rate

Transmission rate

With the "Transmission rate", you can determine the data transmission rate from the SSI encoder to the SM 338.

Select the desired transmission rate in the drop-down list.

Observe the relationship between line length and transmission rate:

- up to 320 m → 125 kHz
- up to 160 m → 250 kHz
- up to 60 m → 500 kHz
- up to 20 m → 1 MHz

Selecting monoflop time

Monoflop time

The monoflop time is the break interval between 2 SSI message frames.

The configured monoflop time must be greater than the monoflop time of the absolute value encoder used (see manufacturer's technical data).

The time 2 × (1 / transmission rate) is added to the configured value. With a transmission rate of 125 kHz and a configured monoflop time of 16 µs, the actual effective monoflop time is 32 µs.

The following limitation applies to the monoflop time of the absolute value encoder:

(1 / transmission rate) < monoflop time of absolute value encoder < 64 µs + 2 × (1 / transmission rate)

Select the corresponding monoflop time in the drop-down list.

The specification of the monoflop time is relevant to freewheeling encoder value acquistion.

Entering number of adjusted bits

Number of adjusted bits [0..12]

SM 338 requires the number of adjusted bits to determine the encoder position and to correctly detect the direction of movement of the encoder.

This value can be found in the technical data of the absolute value encoder used.

Selecting steps/revolution

Steps/revolution

Up to 13 bits are available for the steps/revolution. The resulting number of steps/revolutions is automatically calculated according to the information "Number of adjusted bits (Page 10423)".

Selecting the digital input for the FREEZE function

FREEZE function

The FREEZE function "freezes" the actual encoder values of SM 338. The FREEZE function is coupled to the digital inputs DI 0 and DI 1 of SM 338.

The freeze function is triggered by a signal transition (positive edge) at digital input DI 0 or DI 1. Bit 31 = 1 (output address) identifies a frozen encoder value. You can freeze one, two or three encoder values using one digital input.

To allow their event-triggered evaluation, the encoder values are retained until the FREEZE function is terminated.

Enabling the FREEZE function

The FREEZE function must be enabled during parameter assignment.

To do so, select the digital input at which a rising edge freezes the encoder value:

- off (encoder value is not frozen)
- DI 0
- DI 1

Canceling the FREEZE function

The FREEZE function must be terminated separately at each encoder input. You acknowledge the function in the user program by using the operation T POB "xyz" to set bit 0, 1 or 2 according to the encoder input (see the program example (Page 10424)).

The acknowledgement resets bit 31 of the corresponding encoder value, and initiates a refresh of the encoder values. You can freeze the encoder values again after you cleared the ACK bit at the output address of the module.

In isochronous mode, the acknowledgement is processed at the time T_o. From this point on, you can freeze the encoder values again by setting the digital inputs.

Note

The FREEZE function is automatically acknowledged if you reconfigure the corresponding encoder input with different parameters.
I The freeze function will be unaffected if the parameters are identical.

Addressing SM 338

Data areas for encoder values

The SM 338 inputs and outputs are addressed beginning at the module start address. The input and output address is determined during the configuration of the SM 338.

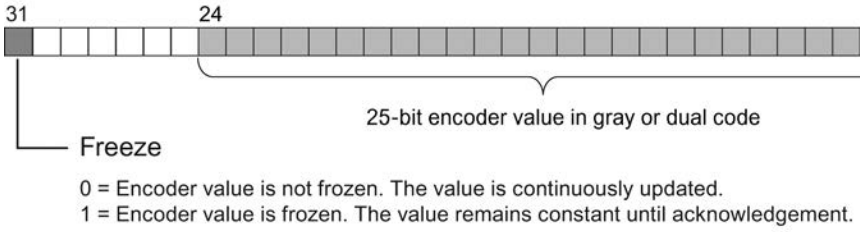
Input addresses

The table below shows the input addresses of the SM 338.

Encoder input	Input address (derived from configuration) + address offset
0	"Module start address"
1	"Module start address" + 4 byte address offset
2	"Module start address" + 8 byte address offset

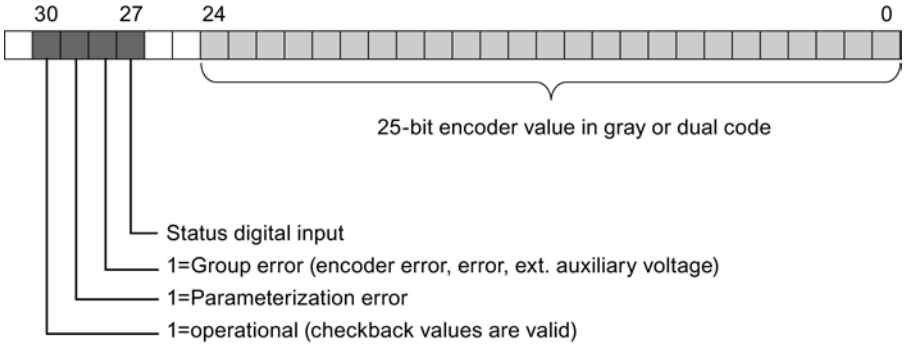
Structure of double data word in Standard Mode

The figure below shows the structure of the double data word for each encoder input in standard mode.



Structure of double data word in Fast Mode

The figure below shows the structure of the double data word for each encoder input in fast mode.

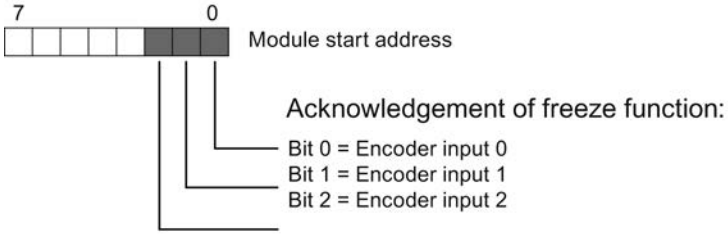


The status of digital input DI0 is reported in bit 27 (digital input status) in the double data word from encoder input 0, and the status of digital input DI1 in the double data word from encoder input 1.

In the double data word from encoder input 2, bit 27 is always = 0.

Output data in standard mode

The figure below shows the structure of the output data in standard mode.



no output data are supported in Fast Mode.

Reading data areas

You can read the data areas in your user program using the operation L PID "xyz."

Program example of access to encoder values and use of the FREEZE function

You want to read and evaluate the values at the encoder inputs. The module start address is 256.

```
STL                 Explanation
L PED256            // Read encoder value in the address area for encoder input 0

T MD 100            // Save encoder value to memory double word
UD W#16#1FFFFFF     // smooth the lower 25 bits to completely hide the upper 7 bits
T MD 120            // Encoder value without FREEZE or other status bits can also
                    be used in fast mode, since only 25 bits are displayed
U M 100.7           // FREEZE state for subsequent acknowledgement
= M 99.0            // determine and save

L PED 260           // Read encoder value in the address area for encoder input 1

T MD 104            // Save encoder value to memory double word
UD W#16#1FFFFFF     // smooth the lower 25 bits to completely hide the upper 7 bits
T MD 124            // Encoder value without FREEZE or other status bits can also
                    be used in fast mode, since only 25 bits are displayed
U M 104.7           // FREEZE state for subsequent acknowledgement
= M 99.1            // determine and save

L PED 264           // Read encoder value in the address area for encoder input 2

T MD 108            // Save encoder value to memory double word
UD W#16#1FFFFFF     // smooth the lower 25 bits to completely hide the upper 7 bits
T MD 128            // Encoder value without FREEZE or other status bits can also
                    be used in fast mode, since only 25 bits are displayed
U M 108.7           // FREEZE state for subsequent acknowledgement
= M 99.2            // determine and save

L MB 99             // Load FREEZE state and
T PAB 256           // acknowledge (SM 338: output address 256)
```

You can then process the encoder values from the bit memory address areas MD 100, MD 104 and MD 108. The encoder value is set in bits 0 to 30 of the memory double word.

SM 338: Online & Diagnostics

Diagnostics of the SM 338

The SM 338 provides diagnostics messages, i.e. it always provides all diagnostics messages without user intervention.

Measures after a diagnostic message

Actions initiated by diagnostic messages:

- The diagnostic message is entered in the diagnosis of the module and forwarded to the CPU.
- The SF LED on the module is lit.
- If you configured "Enable Diagnostic Interrupt (Page 10421)", the system triggers a diagnostic interrupt and calls the diagnostic interrupt OB.

Reading diagnostic messages

You can read detailed diagnostic messages using instructions in the user program.

The error cause can be displayed in the module diagnostics.

Diagnostic message using the SF LED

The SM 338 indicates errors at its SF LED (group error LED.) The SF LED lights up when the SM 338 generates a diagnostic message. It goes dark after all error states are cleared.

The SF LED also lights up to indicate external errors (short circuit at the encoder supply), regardless of the CPU operating state (at POWER ON.)

The SF LED lights up temporarily at startup during the self-test of SM 338.

Diagnostic messages of SM 338

The table below shows the diagnostic messages of SM 338.

Diagnostics message	LED	Scope of diagnostics
Module error	SF	Module
Internal error	SF	Module
External error	SF	Module
Channel error	SF	Module
External auxiliary voltage missing	SF	Module
Module parameters not assigned	SF	Module
Incorrect parameters	SF	Module
Channel information available	SF	Module
Watchdog time-out	SF	Module
Channel error	SF	Channel (encoder input)

Diagnostics message	LED	Scope of diagnostics
Configuration / programming error	SF	Channel (encoder input)
External channel error (encoder error)	SF	Channel (encoder input)

Causes of error and troubleshooting

The table below shows error causes and troubleshooting for diagnostic messages of SM 338.

Diagnostics message	Possible cause of error	To correct or avoid error
Module error	Any, the module has detected an error.	
Internal error	Module has detected an error within the automation system.	
External error	Module has detected an error outside of the automation system.	
Channel error	Indicates that only specific channels are faulty.	
External auxiliary voltage missing	Supply voltage L+ of the module is missing	Connect supply L+
Module parameters not assigned	Module requires information whether it should operate with default system parameters or user parameters.	Message present after POWER ON, until the CPU has completed the transfer of parameters; configure the module as required.
Incorrect parameters	One parameter, or the combination of parameters, is not plausible	Program the module
Channel information available	Channel error; module can provide additional channel information.	
Watchdog time-out	Infrequent high electromagnetic interference	Eliminate interference
Channel error	Any, the module has detected an error at the encoder input.	
Configuration / programming error	Illegal parameter transferred to module	Program the module
External channel error (encoder error)	Wire-break at encoder cable, encoder cable not connected, or encoder defective.	Check connected encoder

11.4.2.3 Using ET 200S 1SSI

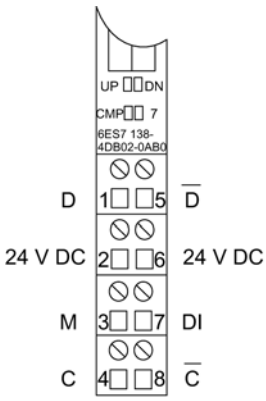
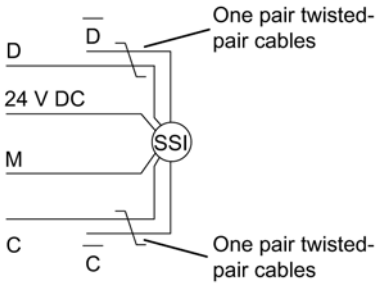
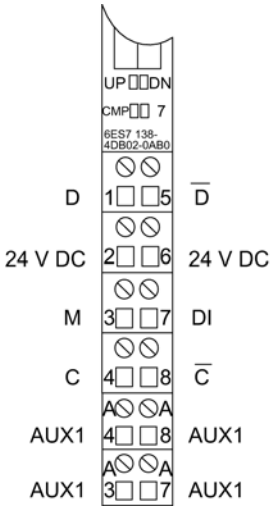
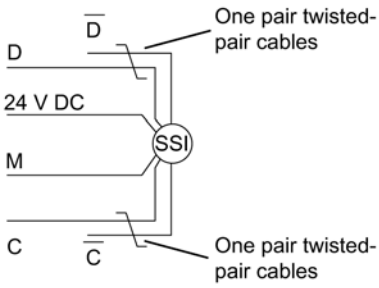
Terminal Assignment of the 1SSI

Wiring Rules

The cables (terminals 1 and 5 and terminals 4 and 8) must be shielded, twisted-pair cables. The shield must be supported at both ends. To do this, use the shield contact (see the appendix of the ET 200S Distributed I/O System (http://support.automation.siemens.com/WW/view/en/1144348) operating instructions).

Terminal assignment of the 1SSI

The following table shows the terminal assignment of 1SSI.

View	Terminal assignment	Remarks	
	TM-E15S24-01 and EM 1SSI 	Terminals 1 to 8:	
		1/5:	Data from the SSI encoder [1]
		2/6:	Power supply for absolute value encoder and switch [2]
		3:	Chassis ground
		7:	Digital input latch function
		4/8:	SSI clock (clock wire) [1]
	TM-E15S26-A1 and EM 1SSI 		

[1] It is essential that you maintain the correct polarity when wiring the encoder. If you do not, an absolute encoder error is reported. Signals to RS 422

[2] Short circuit-proof, maximum 0.5 A.

Configuring and assigning parameters to 1SSI

Operation

Selecting an operating mode

Operating mode

You define 1SSI functionality by specifying an operating mode.

Operating mode	Description
Standard Mode (Page 10430)	1SSI standard operating mode with complete functional scope: • Scaling the encoder value • Direction reversal to adapt the direction of motion of the absolute encoder to the axis. • Encoder value acquisition modes: – freewheeling – synchronous to the update rate • Maximum encoder sampling rate (e.g. for ultrasonic encoders) is taken into account in isochrone mode • Lifebeat in isochrone mode • Parity check of encoder value can be performed • Gray code/dual code converter • Latch function for freezing the current encoder value • Compare function for comparing the current encoder value and loadable comparison values This operating mode is suitable for monitoring or detecting position points and for length measurement, edge detection and synchronization with workpieces.
Fast mode (Page 10442)	The 1SSI has compressed functionality in this operating mode. The latch and compare functions are not available. This operating mode is suitable for rapid encoder value acquisition and controller applications such as position controllers with the position as the actual value.

Standard mode parameters

Enabling group diagnostics

Group diagnostics

Channel-specific diagnostics will be carried out if you enable group diagnostics in your parameter assignment.

Selecting encoder value acquisition

Description

The following alternatives are available for encoder value acquisition:

- freewheeling encoder value acquisition
- synchronous encoder value acquisition
- Isochrone encoder value acquisition

You can set cyclic or synchronous encoder value acquisition with the "Detection" parameter in parameter assignment. This parameter only works in non-isochrone mode.

Encoder value acquisition will be isochrone when the 1SSI is operating in isochrones mode. In this case, the "Detection" parameter is not evaluated.

The following table sets out these encoder value acquisition conditions.

Operation	"Detection" parameter	Encoder value acquisition
Non-isochrone mode	freewheeling	freewheeling encoder value acquisition
	Synchronous	synchronous encoder value acquisition
Isochrone mode	- (irrelevant)	Isochrone encoder value acquisition

Cyclic encoder value acquisition

You obtain maximum accuracy when latching the encoder value (Page 10435) during freewheeling encoder value acquisition.

The 1SSI initiates the transmission of a frame each time the parameterized monoflop time (Page 10433) elapses.

The 1SSI processes the acquired encoder value asynchronously to these freewheeling frames in the update rate cycle.

Freewheeling acquisition therefore returns encoder values of different ages. The difference between the maximum and minimum age is the jitter.

Synchronous encoder value acquisition

Synchronous encoder value acquisition gives you maximum accuracy.

The 1SSI initiates the transmission of a frame in the cycle of its update rate.

The 1SSI processes the transferred encoder value synchronously with its update rate.

Selecting encoder type

Encoder type

The 1SSI supports the following encoder types:

- Absolute encoder (SSI) with 13 bits
  to
- Absolute encoder (SSI) with 25 bits

You will find the encoder type in the technical specifications for your absolute encoder.

Selecting "No encoder" deactivates the encoder input.

Selecting Gray code/dual code converter

Gray code/dual code converter

The "Gray code/dual code converter" parameter defines how the codes returned by the encoder are processed:

- When "Gray code" is set, the encoder value returned by the absolute encoder in gray code will be converted to dual code.
- When "Dual code" is set, the values returned by the encoder will remain unchanged.

Selecting the transmission rate

Transmission rate

You define the data transmission rate between the SSI encoder and the 1SSI with the "transmission rate".

Select the desired transmission rate in the drop-down box.

Note that the transmission rate affects how accurate and up-to-date the encoder values are.

The maximum transmission rate depends on the cable length:

- 320 m → 125 kHz
- 160 m → 250 kHz
- 60 m → 500 kHz
- 20 m → 1 MHz
- 10 m → 1.5 MHz
- 8 m → 2 MHz

See also

Selecting monoflop time (Page 10433)

Selecting parity

Parity

With "Parity", you specify whether no parity, even parity or odd parity should be assigned for an absolute encoder.

A configured parity bit is not included in the "Encoder Type (Page 10432)" parameter. If a 25-bit encoder with parity is parameterized, the encoder will read 26 bits.

The bit following the LSB (least significant bit) of the encoder is evaluated as a parity bit. A parity error is reported in the feedback interface (Page 10441) via the bit ERR_SSI.

Selecting monoflop time

Monoflop time

The monoflop time represents the interval between two SSI frames.

The parameterized monoflop time must be greater than the monoflop time of the absolute encoder used (refer to the technical specifications of the manufacturer).

The following restriction applies to the monoflop time of the absolute encoder:

(1/transmission rate) < monoflop time of absolute encoder < 64 µs

Select the desired monoflop time in the drop-down box.

The specification of the monoflop time is relevant for freewheeling encoder value acquisition (Page 10431).

See also

Selecting the transmission rate (Page 10432)

Activating scaling

Encoder value and scaling

The encoder value transferred contains the encoder position of the absolute encoder. In addition to the encoder position, the encoder transfers additional bits located before and after the encoder position, depending on the encoder used.

The 1SSI requires the following information to allow it to detect the encoder position:

- Encoder type (Page 10432)
- Number of adjusted bits (Page 10434)
- Increments per encoder revolution (Page 10435)

Scaling specifies presentation of the encoder value at the feedback interface (Page 10441).

- "Scaling **on**" specifies that trailing, irrelevant bits in the encoder value are to be discounted.
- "Scaling **off**" specifies that adjusted bits are retained and are available for evaluation.

Entering number of adjusted bits

Number of adjusted bits

The 1SSI requires the number of adjusted bits in order to detect the encoder position and correctly detect the encoder's direction of motion.

You will find this value in the technical specifications for your absolute encoder.

Enabling direction reversal

Direction detection and direction reversal

The 1SSI needs the following information to detect the direction of movement of the encoder correctly:

- Encoder type (Page 10432)
- Number of adjusted bits (Page 10434)
- Increments per encoder revolution (Page 10435)

The direction of motion determined is displayed in the feedback interface (Page 10441) and by the LEDs.

- LED UP: Encoder position change from lower to higher value
- LED DN: Encoder position change from higher to lower value

Direction reversal adapts the encoder direction of motion to the axis.

Two settings are possible:

- **Off:** The direction of the transferred encoder position is maintained.
- **On:** The direction of the transferred encoder position is reversed. This means that although the encoder sends ascending values, descending values are displayed.
  The reversal relates to the increments/encoder revolution set in the parameters.

Example of Direction Reversal

Presettings:

You are using a single-turn encoder with 2^{10} (corresponds to 10 bits) = 1024 increments/encoder revolution (resolution/360°) with the following parameter assignment:

- Encoder type: SSI-13 bit
- Number of adjusted bits: 3 places
- Direction reversal: On
- Increments per encoder revolution: 1024

Encoder value before direction reversal: cyclically recorded encoder position 1023

Encoder value after direction reversal: displayed encoder position 0

Entering increments per encoder revolution

Increments per encoder revolution

The "Increments per encoder revolution" parameter defines the number of increments an encoder returns per revolution.

You will find this value in the technical specifications for your absolute encoder.

Valid value ranges

The limits depend on the individual encoder models. Only values to the power of 2 are permitted.

- 13-bit encoder type: 16 to 8192
- 14-bit encoder type: 16 to 16384
- 15-bit encoder type: 16 to 32768
- 16-bit encoder type: 16 to 65536
- 17-bit encoder type: 16 to 131072
- 18-bit encoder type: 16 to 262144
- 19-bit encoder type: 16 to 524288
- 20-bit encoder type: 16 to 1048576
- 21-bit encoder type: 16 to 2097152
- 22-bit encoder type: 16 to 4194304
- 23-bit encoder type: 16 to 8388608
- 24-bit encoder type: 16 to 16777216
- 25-bit encoder type: 16 to 33554432

Selecting encoder value latching

Latching the encoder value

You use the latch function to "freeze" the current encoder value of the 1SSI with an edge at the digital input (DI).

The encoder value can thus be evaluated on an event-dependent basis.

A frozen encoder value is identified by the set bit 31 and is retained until the latch function is exited.

The frozen encoder value is entered at the feedback interface (Page 10441) at the position of the otherwise cyclically recorded value and assigned the identifier "Bit 31 set".

Note

Direction determination, comparison, and error monitoring also take place when the encoder value is frozen.

Prerequisites for Using the Latch Function

You specify which edge at the digital input is to freeze the encoder value using parameter assignment.

- Falling edge at DI
- Rising edge at DI
- Both edges at DI

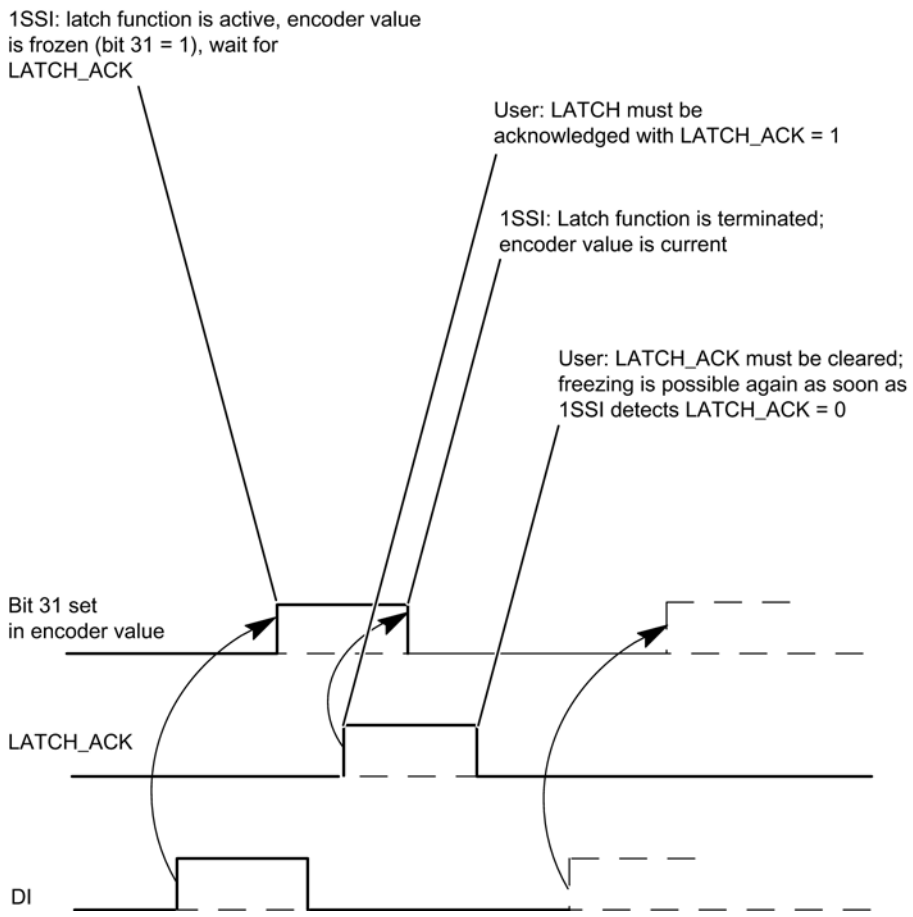
Selecting an edge activates the latch function coupled to the digital input.

The encoder value cannot be frozen if "Inactive" is selected.

Terminating the Latch Function

The latch function must be acknowledged. When the controller program acknowledges the acceptance of the encoder value, bit 31 is deleted and the encoder value is updated again. Freezing is then possible again.

The figure below shows the latch function process.



Selecting comparator behavior

Setting comparators

The encoder position detected can be compared with up to two loadable values. Both comparison results are stored in the feedback interface (Page 10441). The appropriate comparator becomes active only after the comparison value is loaded.

You set the two comparators in the "Comparator 1" and "Comparator 2" parameters:

Setting	Effect on the result of comparison (CMPx)
inactive	The encoder value is not compared. The feedback bit CMPx = 0
UP	The encoder value is compared in the up direction (UP). • If the encoder value ≥ comparison value, feedback bit CMPx = 1. • If the encoder value < comparison value, feedback bit CMPx = 0. • If the direction is down, the feedback bit CMPx remains unchanged. • If no change is detected in the encoder value, the feedback bit CMPx will remain unchanged.

Setting	Effect on the result of comparison (CMPx)
Down	The encoder value is compared in the down direction (DN). • If the encoder value ≤ comparison value, feedback bit CMPx = 1. • If the encoder value > comparison value, feedback bit CMPx = 0. • If the direction is up, the feedback bit CMPx will remain unchanged. • If no change is detected in the encoder value, the feedback bit CMPx will remain unchanged.
In both directions	The encoder value is compared in both directions. If the direction is up, the following conditions apply: • If the encoder value ≥ comparison value, feedback bit CMPx = 1. • If the encoder value < comparison value, feedback bit CMPx = 0. • If no change is detected in the encoder value, the feedback bit CMPx will remain unchanged. If the direction is down, the following conditions apply: • If the encoder value ≤ comparison value, feedback bit CMPx = 1. • If encoder value > comparison value, feedback bit CMPx = 0. • If no change is detected in the encoder value, the feedback bit CMPx will remain unchanged.

The comparison result is deleted and entered in accordance with the directional setting as soon as you load a comparison value.

Note

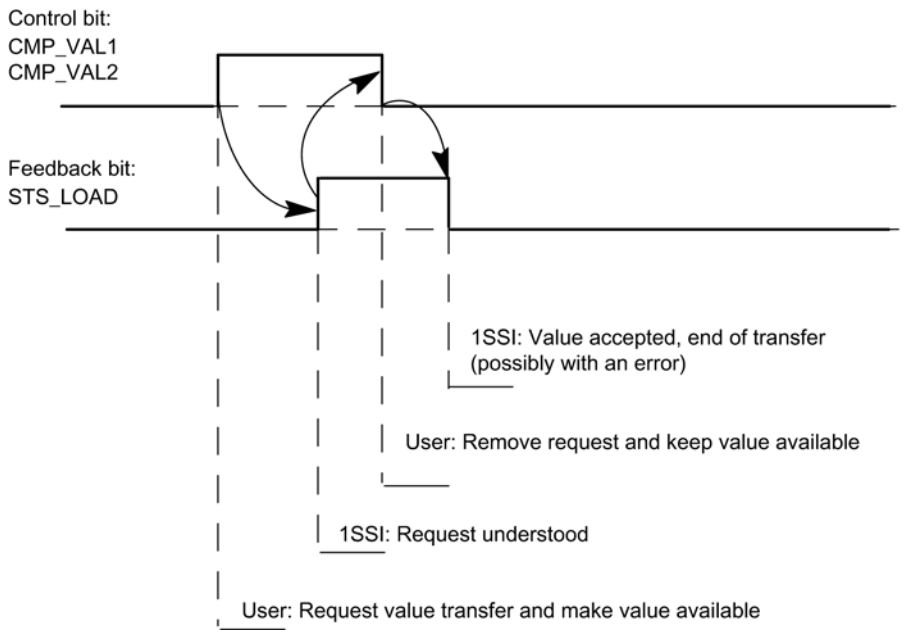
Only one control bit can be set at a particular time:

CMP_VAL1 or CMP_VAL2).

Otherwise, the ERR_LOAD error will be reported until both control bits are deleted.

Loading the Comparison Value

The figure below illustrates the value transfer process:



Comparator in Isochronous Mode

In isochronous mode, the comparison values are loaded at time T_o and are effective as of time T_i in the same bus cycle.

Enabling lifebeat (default mode)

Lifebeat

This parameter is only active in isochrone mode.

When lifebeat is activated, the lifebeat bit is toggled each time an encoder value is read in in isochronous mode, i.e. the last value sent is inverted. If a reduction is assigned in the "Encoder Sampling Rate (Page 10439)" parameter, the value is only toggled if an encoder value has actually been read in.

The lifebeat bit can be found in the feedback interface (Page 10441) in byte 4 / bit 7.

Setting the encoder sampling rate

Encoder sampling rate

This parameter is only active in isochrone mode.

Any encoder sampling rate to be taken into account is set here. This allows slower encoders (such as ultrasound encoders) to be used, even with a fast processing cycle.

An integer reduction n is calculated using the set frequency. The encoder is then only read in again every nth clock cycle.

Example

- Processing cycle 500 µs
- Encoder sampling rate: 1.2 kHz (approximately every 833 µs)

This produces a reduction n = 2.

The encoder is only read in again every second processing cycle, in other words each millisecond.

Control and feedback interface for standard mode

Standard mode feedback interface

Note

The following control interface data is consistent for 1SSI:

- Bytes 0 to 3
- Bytes 4 to 7

Control interface assignment

The table below shows the assignment of the 1SSI control interface (outputs) in standard mode.

Address	Assignment		Name
Bytes 0 to 3	Comparison value 1 or 2 (double word)		
Byte 4	Bit 7	Error Acknowledgment	EXTF_ACK
	Bit 6	Acknowledgement latch function	LATCH_ACK
	Bit 5	Reserved = 0	
	Bit 4	Reserved = 0	
	Bit 3	Reserved = 0	
	Bit 2	Reserved = 0	
	Bit 1	Load comparison value 2	CMP_VAL2
	Bit 0	Load comparison value 1	CMP_VAL1
Byte 5	Reserved = 0		
Bytes 6 to 7	Reserved = 0		

Notes on the control bits

Bit	Explanation
CMP_VAL1	Load comparison value 1
CMP_VAL2	Load comparison value 2
EXTF_ACK	Error acknowledgement for the "Absolute encoder error" ERR_SSI and "Encoder supply short circuit" ERR_24V.
LATCH_ACK	Acknowledgement for latch function

Standard mode feedback interface

Note

The following feedback interface data is consistent for 1SSI:

- Bytes 0 to 3
- Bytes 4 to 7

Feedback interface assignment

The table below shows the assignment of the 1SSI feedback interface (inputs) in standard mode.

Address	Assignment		Name
Bytes 0 to 3	Encoder value (double word) (bit 31 set, encoder value frozen)		
Byte 4	Bit 7	Reserved = 0 or lifebeat	Lifebeat
	Bit 6	Ready	RDY
	Bit 5	Parameter assignment error	ERR_PARA
	Bit 4	Absolute encoder error	ERR_SSI
	Bit 3	Encoder supply short circuit	ERR_24V
	Bit 2	DI status	STS_DI
	Bit 1	Status DN	STS_DN
	Bit 0	Status UP	STS_UP
Byte 5	Bit 7	Reserved = 0	
	Bit 6	Reserved = 0	
	Bit 5	Reserved = 0	
	Bit 4	Reserved = 0	
	Bit 3	Comparison value 2 reached	CMP2
	Bit 2	Comparison value 1 reached	CMP
	Bit 1	Load function error	ERR_LOAD
	Bit 0	Load function is running	STS_LOAD
Bytes 6 to 7	Reserved = 0		

Notes on the feedback bits

Bit	Explanation
CMP	Comparison result of comparator 1
CMP2	Comparison result of comparator 2
ERR_24V	There is a short circuit in the encoder supply. ERR_24V is reset when the short circuit is eliminated and acknowledged with the EXTF_ACK control bit.
ERR_LOAD	Error while loading the comparison values because both control bits CMP_VAL1 and CMP_VAL2 are set.
ERR_PARA	Incorrect parameter assignment for the ET 200S station. Cause: Increments per encoder revolution not in the valid value range for the encoder type. The parameter bit is cleared when correct parameter assignment is transmitted.
ERR_SSI	The 1SSI detects an "Absolute encoder error" if the frames at the SSI interface are interrupted. Causes: No encoder connected; wire break in the encoder cable; encoder supply under-voltage; encoder type, transmission rate, monoflop time do not correspond to the connected encoder; programmable encoders do not correspond to the settings on the 1SSI; encoder is defective or faults or parity errors exist. ERR_SSI is reset when the cause of the error is eliminated and acknowledged with the EXTF_ACK control bit.
Lifebeat	This parameter is only active in isochrone mode. When lifebeat is on, the lifebeat bit is toggled each time an encoder value is read in in isochronous mode, i.e. the last value sent is inverted. If a reduction is assigned in the "Encoder Sampling Rate" parameter, the value is only toggled if an encoder value has actually been read in.
STS_DI	The bit displays the status of the digital input DI.
STS_DN	Status direction down; for encoder value change from larger to smaller encoder positions (including zero pass)
STS_LOAD	Feedback bit for CMP_VAL1 and CMP_VAL2. The 1SSI uses this bit to indicate that a comparison value is loaded.
STS_UP	Status direction down; for encoder value change from smaller to larger encoder positions (including zero pass)
RDY	1SSI parameter assignment is correct, and the module is executing its functions. The feedback displayed is valid. For absolute value encoder errors, ERR_SSI is also set.

Fast mode parameters

Enabling group diagnostics

Group diagnostics

Channel-specific diagnostics will be carried out if you enable group diagnostics in your parameter assignment.

Selecting encoder value acquisition

Description

The absolute encoder transfers its encoder values in frames to the 1SSI. The transmission of frames is initiated by the 1SSI. The following alternatives are available for encoder value acquisition:

- freewheeling encoder value acquisition
- synchronous encoder value acquisition
- Isochrone encoder value acquisition

You can set freewheeling or synchronous encoder value acquisition with the "Detection" parameter in parameter assignment. This parameter only works in non-isochrone mode.

Encoder value acquisition will be isochrone when the 1SSI is operating in isochrones mode. In this case, the "Detection" parameter is not evaluated.

The following table sets out these encoder value acquisition conditions.

Operation	"Detection" parameter	Encoder value acquisition
Non-isochrone mode	freewheeling	freewheeling encoder value acquisition
	Synchronous	synchronous encoder value acquisition
Isochrone mode	- (irrelevant)	Isochrone encoder value acquisition

Cyclic encoder value acquisition

The 1SSI initiates the transmission of a frame each time the parameterized monoflop time (Page 10445) elapses.

The 1SSI processes the acquired encoder value asynchronously to these freewheeling frames in the update rate cycle.

Freewheeling acquisition therefore returns encoder values of different ages. The difference between the maximum and minimum age is the jitter.

Synchronous encoder value acquisition

Synchronous encoder value acquisition gives you maximum accuracy.

The 1SSI initiates the transmission of a frame in the cycle of its update rate.

The 1SSI processes the transferred encoder value synchronously with its update rate.

Isochrone encoder value acquisition

Isochrone encoder value acquisition is automatically set if the bus system and I/O modules are synchronized.

The 1SSI initiates the transmission of a frame in each bus cycle at time T_i as long as the configured maximum encoder sampling rate does not result in a reduction.

The 1SSI processes the transmitted encoder value isochronously to the bus cycle.

Selecting encoder type

Encoder type

The 1SSI supports the following encoder types:

- Absolute encoder (SSI) with 13 bits
  to
- Absolute encoder (SSI) with 25 bits

You will find the encoder type in the technical specifications for your absolute encoder.

Selecting "No encoder" deactivates the encoder input.

Selecting Gray code/dual code converter

Gray code/dual code converter

The "Gray code/dual code converter" parameter defines how the codes returned by the encoder are processed:

- When "Gray code" is set, the encoder value returned by the absolute encoder in gray code will be converted to dual code.
- When "Dual code" is set, the values returned by the encoder will remain unchanged.

Selecting the transmission rate

Transmission rate

You define the data transmission rate between the SSI encoder and the 1SSI with the "transmission rate".

Select the desired transmission rate in the drop-down box.

Note that the transmission rate affects how accurate and up-to-date the encoder values are.

The maximum transmission rate depends on the cable length:

- 320 m → 125 kHz
- 160 m → 250 kHz
- 60 m → 500 kHz
- 20 m → 1 MHz
- 10 m → 1.5 MHz
- 8 m → 2 MHz

See also

Selecting monoflop time (Page 10445)

Selecting parity

Parity

"Parity" allows you to specify whether no parity, even parity or odd parity is to be assigned for an absolute encoder.

A configured parity bit is not included in the "Encoder Type (Page 10444)" parameter. If a 25-bit encoder with parity is parameterized, the encoder will read 26 bits.

The bit following the LSB (least significant bit) of the encoder is evaluated as a parity bit. A parity error is reported in the feedback interface (Page 10448) via the bit EXTF.

Selecting monoflop time

Monoflop time

The monoflop time represents the interval between two SSI frames.

The parameterized monoflop time must be greater than the monoflop time of the absolute encoder used (refer to the technical specifications of the manufacturer).

The following restriction applies to the monoflop time of the absolute encoder:

(1/transmission rate) < monoflop time of absolute encoder < 64 µs

Select the desired monoflop time in the drop-down box.

The specification of the monoflop time is relevant for freewheeling encoder value acquisition (Page 10443).

See also

Selecting the transmission rate (Page 10444)

Activating scaling

Encoder value and scaling

The encoder value transferred contains the encoder position of the absolute encoder. In addition to the encoder position, the encoder transfers additional bits located before and after the encoder position, depending on the encoder used.

The 1SSI requires the following information to allow it to detect the encoder position:

- Encoder type (Page 10444)
- Number of adjusted bits (Page 10446)
- Increments per encoder revolution (Page 10447)

Scaling specifies presentation of the encoder value at the feedback interface (Page 10448).

- "Scaling **on**" specifies that trailing, irrelevant bits in the encoder value are to be discounted.
- "Scaling **off**" specifies that adjusted bits are retained and are available for evaluation.

Entering number of adjusted bits

Number of adjusted bits

The 1SSI requires the number of adjusted bits in order to detect the encoder position and correctly detect the encoder's direction of motion.

You will find this value in the technical specifications for your absolute encoder.

Enabling direction reversal

Direction detection and direction reversal

The 1SSI needs the following information to detect the direction of movement of the encoder correctly:

- Encoder type (Page 10444)
- Number of adjusted bits (Page 10446)
- Increments per encoder revolution (Page 10447)

The direction of motion determined is displayed in the feedback interface (Page 10448) and by the LEDs.

- LED UP: Encoder position change from lower to higher value
- LED DN: Encoder position change from higher to lower value

Direction reversal adapts the encoder direction of motion to the axis.

Two settings are possible:

- **Off:** The direction of the transferred encoder position is maintained.
- **On:** The direction of the transferred encoder position is reversed. This means that although the encoder sends ascending values, descending values are displayed.
  The reversal relates to the increments/encoder revolution set in the parameters.

Example of Direction Reversal

Presettings:

You are using a single-turn encoder with 2^{10} (corresponds to 10 bits) = 1024 increments/encoder revolution (resolution/360°) with the following parameter assignment:

- Encoder type: SSI-13 bit
- Number of adjusted bits: 3 places
- Direction reversal: On
- Increments per encoder revolution: 1024

Encoder value before direction reversal: cyclically recorded encoder position 1023

Encoder value after direction reversal: displayed encoder position 0

Entering increments per encoder revolution

Increments per encoder revolution

The "Increments per encoder revolution" parameter defines the number of increments an encoder returns per revolution.

You will find this value in the technical specifications for your absolute encoder.

Valid value ranges

The limits depend on the individual encoder models. Only values to the power of 2 are permitted.

- 13-bit encoder type: 16 to 8192
- 14-bit encoder type: 16 to 16384
- 15-bit encoder type: 16 to 32768
- 16-bit encoder type: 16 to 65536
- 17-bit encoder type: 16 to 131072
- 18-bit encoder type: 16 to 262144
- 19-bit encoder type: 16 to 524288
- 20-bit encoder type: 16 to 1048576
- 21-bit encoder type: 16 to 2097152
- 22-bit encoder type: 16 to 4194304
- 23-bit encoder type: 16 to 8388608
- 24-bit encoder type: 16 to 16777216
- 25-bit encoder type: 16 to 33554432

Enabling lifebeat (Fast mode)

Lifebeat

This parameter is only active in isochrone mode.

When lifebeat is activated, the lifebeat bit is toggled each time an encoder value is read in in isochronous mode, i.e. the last value sent is inverted. If a reduction is assigned in the "Encoder sampling rate (Page 10448)" parameter, the value is only toggled if an encoder value has actually been read in.

The lifebeat bit can be found in the feedback interface (Page 10448) in bytes 0 to 3 / bit 31.

Setting the encoder sampling rate

Encoder sampling rate

This parameter is only active in isochrone mode.

Any encoder sampling rate to be taken into account is set here. This allows slower encoders (such as ultrasound encoders) to be used, even with a fast processing cycle.

An integer reduction n is calculated using the set frequency. The encoder is then only read in again every nth clock cycle.

Example

- Processing cycle 500 µs
- Encoder sampling rate: 1.2 kHz (approximately every 833 µs)

This produces a reduction n = 2.

The encoder is only read in again every second processing cycle, in other words each millisecond.

Fast mode feedback interface

Feedback interface assignment

The table below shows the assignment of the 1SSI feedback interface (inputs) in fast mode.

Address	Assignment		Name
Bytes 0 to 3	Bit 31	Reserved = 0 or lifebeat	Lifebeat
	Bit 30	Ready for operation (feedback is valid)	RDY
	Bit 29	Parameter assignment error	ERR_PARA
	Bit 28	Group error , absolute encoder or encoder supply short circuit	EXTF
	Bit 27	DI status	STS_DI
	Bit 26	Status DN	STS_DN
	Bit 25	Status UP	STS_UP
	Bits 0 to 24	Encoder value	

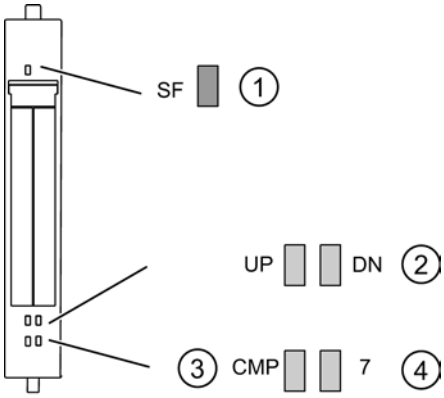
Notes on the feedback bits

Bit	Explanation
ERR_PARA	Incorrect parameter assignment for the ET 200S station. Cause: Increments per encoder revolution not in the valid value range for the encoder type. The parameter bit is cleared when correct parameter assignment is transmitted.
EXTF	Group error absolute encoder or encoder supply short circuit Causes: • The encoder supply has short-circuited • No encoder connected; wire break in the encoder cable; encoder type, transmission rate, monoflop time do not correspond to the connected encoder; programmable encoders do not correspond to the settings on the 1SSI; encoder is defective or faults or parity errors exist. EXTF is reset when the causes of the errors have been eliminated.
Lifebeat	This parameter is only active in isochrone mode. When lifebeat is on, the lifebeat bit is toggled each time an encoder value is read in in isochronous mode, i.e. the last value sent is inverted. If a reduction is assigned in the "Encoder Sampling Rate" parameter, the value is only toggled if an encoder value has actually been read in.
STS_DI	The bit displays the status of the digital input DI.
STS_DN	Status direction down; for encoder value change from larger to smaller encoder positions (including zero pass)
STS_UP	Status direction down; for encoder value change from smaller to larger encoder positions (including zero pass)
RDY	1SSI parameter assignment is correct, and the module is executing its functions. The feedback displayed is valid. For absolute value encoder errors, EXTF is also set.

Diagnostics of the 1SSI

Diagnostics using the LED display

LED display on the 1SSI



① Group error (red)
② Status display for change in sensor value (green)
③ Status display for result of comparison
④ Status display for digital input (green)

Status and error displays by means of LEDs on the 1SSI

The table below shows the status and error displays on the 1SSI.

Event (LED)					Cause	Remedy
SF	UP	DN	CMP	7		
On					No parameter assignment. There is a diagnostic message.	Check the parameter assignment. Evaluate the diagnostics data.
	On				At value change from smaller to larger sensor values (including zero-crossing)	
		On			At value change from larger to smaller sensor values (including zero-crossing)	
			On		Set for comparison result CMP 1	
				On	DI (latch) is activated.	

Error types

For information on the structure of the channel-related diagnostics, refer to the manual on the interface module used in your ET 200S station.

1SSI error types

The following table shows the error types on the 1SSI.

Error class		Meaning	Remedy
1_D	00001: Short circuit	Short circuit of the supply to the absolute encoder.	Correct the process wiring.
9_D	01001: Error	Internal module error occurred.	Replace the module.
		Load voltage from the power module is too low.	Correct the process wiring. Check the load voltage.
16_D	10000: Parameter assignment error	Parameters have not been assigned to the module.	Correct the parameter assignment.
26_D	11010: External error	Start/Stop bit error (error absolute value encoder) Wire break in the encoder cable or encoder cable is not connected. Encoder type, transmission rate, monoflop time do not correspond to the connected encoder Programmable encoders do not correspond to the settings on the 1SSI Sensor is defective or there are faults.	Replace the encoder Correct the process wiring Correct the parameter assignment.

11.4.2.4 Basics of positioning with function modules

What is positioning?

Positioning

Positioning is taking up a specific spatial position, in particular a position defined by coordinates.

In SIMATIC technology, positioning means adjusting and positioning mechanical axes. A range of difference procedures are used:

- Controlled positioning with rapid traverse/creep speed
- Controlled positioning
- Positioning with stepper motor

Positioning can control linear axes with a limited traversing range and continuous rotary axes.

Controlled positioning with rapid traverse/creep speed

Controlled positioning

Each positioning process is characterized by:

- A starting position
- The target of the positioning
- Parameters that determine the sequence of the positioning

The target is initially approached at a higher velocity (rapid traverse). At a specified distance from the target, the velocity switches to creep speed. The drive is switched off shortly before the axis reaches the target - also at a specified distance from the target. In doing so, the module monitors the target approach.

The drive is controlled via digital outputs with rapid traverse or creep speed and in the appropriate direction.

Ranges and switching points of the positioning module

Target

The target is the absolute or relative position on the axis that is approached during a positioning process.

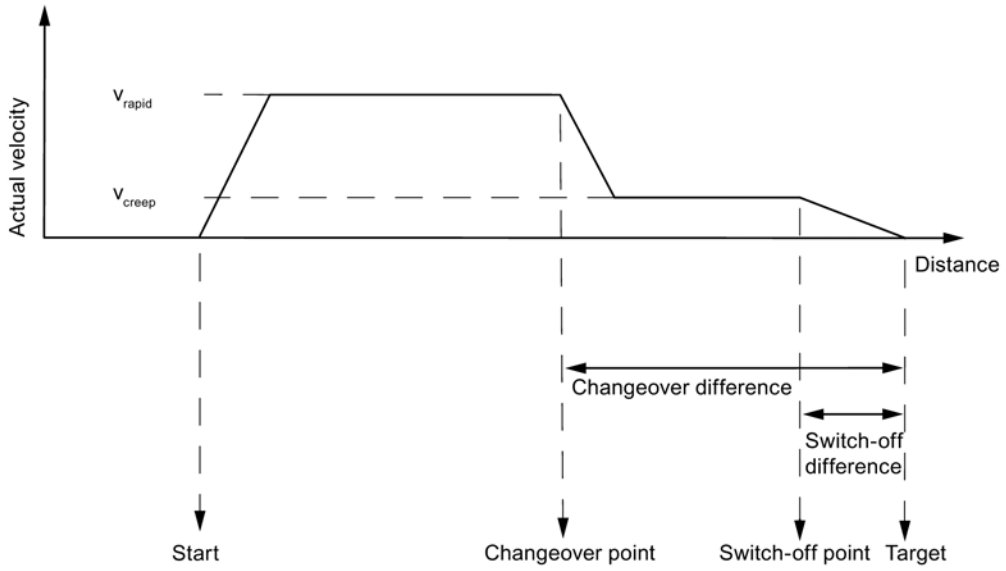
Definition of the switching points and switching ranges

The following ranges and positions can be parameterized for controlled positioning:

Range / Position	Description
Operating range	defines the range you set for your job by means of the software limit switches or the end of the rotary axis.
Switch-over difference	defines the distance from the target at which the drive changes from rapid traverse to creep traverse.
Switch-over point	Defines the position at which the drive changes from rapid traverse to creep speed.
Switch-off difference	defines the distance from the target at which the drive is switched off.
Switch-off point	Defines the position at which the drive is switched off. The positioning module then adopts the monitoring functions from this point on.
Target range	defines the positioning precision for your application and surrounds the target symmetrically.
Standstill range	Defines a symmetrical range around the target to be monitored by the positioning module.

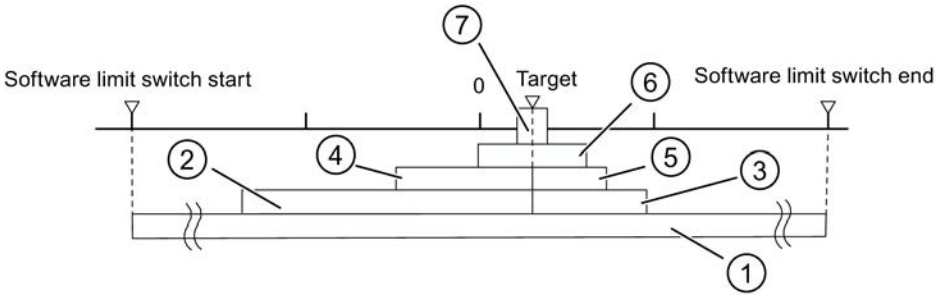
Switching points and switching differences

The following figure shows how the switching points and switching differences can be arranged for positioning. In the interests of simplicity, it is assumed that the actual speed changes linearly over the distance traversed. The emerging ramps can be explained by mechanical inertia or by the parameterization possibilities of the power section.



Switching ranges

The figure below shows how the switching ranges can be arranged around the target.



① Operating range
② Changeover difference in direction plus
③ Changeover difference in direction minus
④ Switch-off difference in plus traverse direction
⑤ Switch-off difference in minus traverse direction
⑥ Standstill range
⑦ Target range

Encoder signals

Supported encoders

Incremental encoders supported

Incremental encoders (Page 10455) with two pulses 90° out of phase with or without zero marks are supported:

- Encoders with asymmetrical output signals with 24 V level
  - Limit frequency = 50 kHz
  - Cable up to 100 m long, shielded
- Encoders with symmetrical output signals with 5 V differential interface in accordance with RS 422
  - Limit frequency = 400 kHz
  - With 5 V voltage supply: Cable up to 32 m long, shielded
  - With 24 V voltage supply: Cable up to 100 m long, shielded

Note

If the 5 V encoder does not output a zero mark signal and wire-break monitoring is enabled, you must interconnect the zero mark signal inputs N and /N externally so that the inputs have different signal levels (for example, N to 5 V, /N to ground).

Absolute encoders supported

Absolute encoders (Page 10457) with serial interface are supported. Position data are transferred synchronously using the SSI protocol (**S**ynchronous**S**erial**I**nterface). FM x51 only supports GRAY code. The data formats 25-bit (fir tree) and 13-bit (half fir tree) result from the arrangement of the data bits in the frames.

Absolute encoders are divided into the categories:

- Single-turn encoder
  The total range of single-turn encoders is scaled to one revolution.
- Multiturn encoder
  The total range of multiturn encoders is scaled to several revolutions.

Encoder type	Frame length
Single-turn encoder	13 bits
Single-turn encoder	25 bits
Multiturn encoder	13 bits … 25 bits

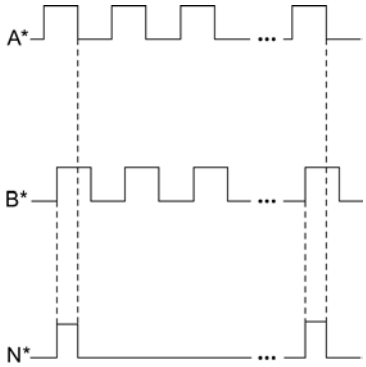
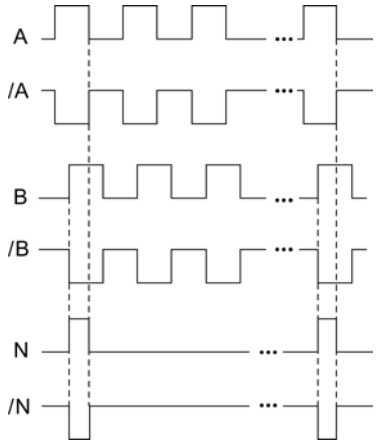
The transmission rate for data transmission depends on the cable length:

- Max. 188 kHz for 200 m of cable, shielded
- Max. 375 kHz for 100 m of cable, shielded
- Max. 750 kHz for 40 m of cable, shielded
- Max. 1.5 MHz for 12 m of cable, shielded

Incremental encoder

Signal forms

The table below illustrates the signal forms of encoders with asymmetrical and symmetrical output signals.

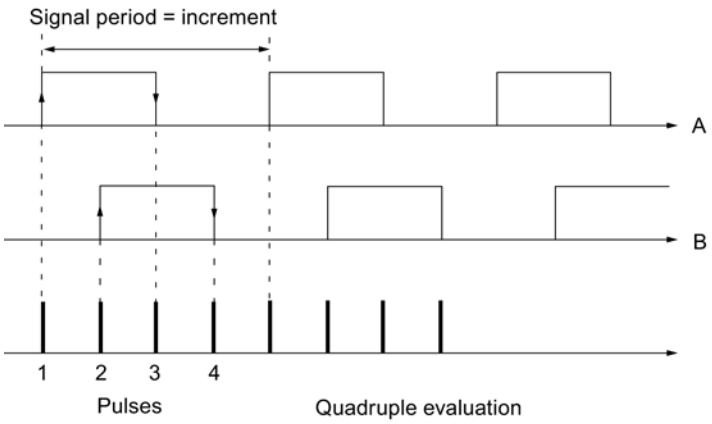
asymmetrical	symmetrical
	

Signal evaluation

Increments

An increment identifies a signal period of the two encoder signals A and B. This value is specified on the rating plate of the encoder and/or in the encoder specifications.

The figure below shows the ratio of increments to pulses.



Pulses

The positioning module evaluates all 4 edges of signals A and B in each increment (quadruple evaluation).

Pulses
1 increment (encoder specification) = 4 pulses (evaluation in the module)

Response times

The positioning module has the following response times for connected incremental encoders:

Response times
Response time = switching cycle of the connected switching elements

Note

You can compensate for the minimum response time by assigning the changeover difference and switch-off difference accordingly.

Indecision

The indecision influences the precision of the positioning. In the case of incremental encoders the indecision is negligible.

Absolute encoder

Pulse evaluation of absolute encoders

Pulse evaluation of absolute encoders
1 increment (encoder specification) = 1 pulse (FM evaluation)

Response times

The FM x51 has the following response times for absolute encoders:

Response times
Minimum response time = message frame time + switching cycle of the connected switching elements
Maximum reaction time = 2 x frame runtime + monoflop time + switching time of the connected switching elements
With programmable absolute encoders: Maximum reaction time = message frame time + monoflop time + switching cycle of the connected switching elements +1/ max. step sequence frequency

Monoflop time

The monoflop time is 64 µs.

Encoders with values greater than the limits specified here are not permitted.

Message frame times

The message frame times depend on the transmission rate:

Transmission rate	Message frame time with 13 bit	Message frame time with 25 bit
0.188 MHz	75 µs	139 µs
0.375 MHz	38 µs	70 µs
0.750 MHz	19 µs	35 µs
1.500 MHz	10 µs	18 µs

Example of response times

This example shows you how to calculate the minimum and maximum reaction time. The example does not use a programmable encoder.

- Hardware switching cycle: approx. 150 µs
- Message frame time: 18 µs at 1.5 MHz transmission rate (25 Bit message frames)
- Monoflop time: 64 µs

Minimum response time = 18 µs + 150 µs = 168 µs

Maximum reaction time = 2 x 18 µs + 64 µs + 150 µs = 250 µs

Note

You can compensate for the minimum response time by assigning the changeover and switch-off differences accordingly.

Indecision

The indecision is the difference between the maximum and the minimum response time. With an absolute encoder this amounts to

Indecision
Indecision = message frame time + monoflop time
With programmable absolute encoders: Indecision = message frame time + monoflop time + 1/max. step sequence frequency

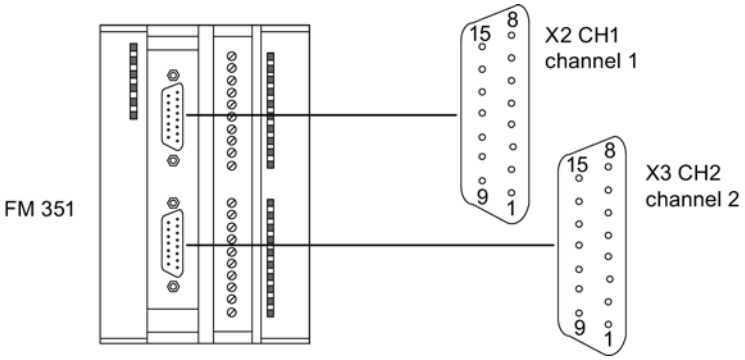
11.4.2.5 Using FM 351

Terminal assignment of FM 351 front connector

Description of the encoder interface

Location of the sub D sockets

The figure shows the position and designation of the connectors on the module. You can connect an incremental encoder or absolute encoder (SSI) to the two sub D sockets.



Assignment of encoder interfaces (X2 and X3)

Pin	Name	Incremental encoder (24 V)	Incremental encoder (5 V)	Absolute encoder
1	A*	Encoder signal A	---	---
2	CLS	---	---	SSI shift clock
3	/CLS	---	---	SSI shift clock inverse
4	B*	Encoder signal B	---	---
5	24 VDC	Encoder supply	Sensor supply	Sensor supply
6	5.2 VDC	---	Sensor supply	Sensor supply
7	M	Chassis ground	Chassis ground	Chassis ground
8	N*	Zero mark signal	---	---
9	RE	Current sourcing/current sinking[1)]	---	---
10	N	---	Zero mark signal	---
11	/N	---	Zero mark signal, inverse	---
12	/B	---	Encoder signal B, inverse	---
13	B	---	Encoder signal B	---
14	/A / /DAT	---	Encoder signal A, inverse	SSI data inverse
15	A / DAT	---	Encoder signal A	SSI data

[1)]**Note:** If you are connecting an incremental encoder from a third party in series-mode switching (current sourcing/current sinking), the following must be observed:

- Sourcing output: Connect RE (9) with ground (7).
- Sinking output: Connect RE (9) with +24 V (5).

Terminal assignment of the front connector

Front connector

Connect the following to the 20-pin front connector:

- the digital inputs and outputs assigned to the channels
- the supply voltage of the encoders and digital outputs

Terminal assignment of front connector (X1)

The table below shows the terminal assignment of the front connector of FM 351.

Wiring	Name	Function	Incremental encoder	Absolute encoder
1	1L+	24 V DC auxiliary voltage for the encoder supply		
2	1M	Ground for sensor power supply		
3	1I0	Channel 1: Digital input 0	Reference point switch	Not used
4	1I1	Channel 1: Digital input 1	Reversing switch	Not used
5	1I2	Channel 1: Digital input 2	Enable input	

Wiring	Name	Function	Incremental encoder	Absolute encoder
6	1I3	Channel 1: Digital input 3	Not used	
7	2I0	Channel 2: Digital input 0	Reference point switch	Not used
8	2I1	Channel 2: Digital input 1	Reversing switch	Not used
9	2I2	Channel 2: Digital input 2	Enable input	
10	2I3	Channel 2: Digital input 3	Not used	
11	1Q0	Channel 1: Digital output 0		
12	1Q1	Channel 1: Digital output 1		
13	1Q2	Channel 1: Digital output 2		
14	1Q3	Channel 1: Digital output 3		
15	2Q0	Channel 2: Digital output 0		
16	2Q1	Channel 2: Digital output 1		
17	2Q2	Channel 2: Digital output 2		
18	2Q3	Channel 2: Digital output 3		
19	2L+	24 V DC auxiliary voltage for the load current supply		
20	2M	Mass load power supply		

Additional information about design and wiring can be found in the Operating Instructions (http://support.automation.siemens.com/WW/view/en/2103043) of the positioning module FM 351.

Auxiliary voltage for the encoder supply (1L+, 1M)

Here, you connect a 24 V DC auxiliary voltage for the encoder supply. The reference potential of this supply (1M) is not connected to the ground of the load power supply (2M) in the FM 351.

The 24 V DC auxiliary voltage for the encoder supply is monitored for undervoltage and ground wire break.

This supplies 24 V DC and 5.2 V DC at the encoder interface (sub D sockets X2 and X3) to the various types of encoders.

CAUTION

Ensure the correct polarity of the 24 V DC auxiliary voltage for the encoder supply (1L+, 1M).

If you reverse the polarity of the 24 V DC auxiliary voltage for the encoder supply, the module will be damaged and will have to be replaced.

Auxiliary voltage for the load power supply (2L+, 2M)

You must connect a 24 V DC auxiliary voltage for the load power supply of the digital outputs to the terminals 2L+ and 2M.

⚠ CAUTION
Ensure the correct polarity of the 24 V DC auxiliary voltage for the load power supply (2L+, 2M). If you reverse the polarity of the 24 V DC auxiliary voltage for the load current supply, the module will be damaged and will have to be replaced.

Wiring information for 24 V DC

When wiring, note that the terminals 1L+, 1M and 2L+, 2M must be interconnected so that the module will operate error-free.

If you connect the 1L+, 1M and 2L+, 2M to separate voltage supplies, after a voltage loss on the auxiliary voltage for the load power supply (2L+, 2M) the synchronization of the axes is maintained.

Load power supplies

The DC load current supply must meet the following requirements:

Only a safe, isolated extra-low voltage of ≤ 60 V DC may be used as the load current supply. Safe isolation can be implemented, for example, by adhering to the specifications

- VDE 0100 Part 410 / HD 384-4-41 / IEC 364-4-41
  (as protective extra low voltage) and/or
- VDE 0805 / EN 60950 / IEC 950
  (as safety low voltage SELV) or
- VDE 0106 Part 101.

8 digital inputs (1I0 to 2I3)

The FM 351 has 4 digital inputs per channel.

You can connect bounce-free switches (24 V current sourcing) or non-contact sensors (2- or 3-wire proximity switches) to the 8 digital inputs.

The digital inputs are not monitored for short circuit or wire break and are electrically isolated from the encoder supply ground and the CPU ground.

The state of each input can be read off from the respective LED.

8 digital outputs (1Q0 to 2Q3)

The FM 351 has 4 digital outputs per channel.

The power unit is controlled via the digital outputs. The function of the digital outputs depends on the control mode (Page 10499).

The digital outputs are not monitored for short circuit or wire break and are electrically isolated from the encoder supply ground and the CPU ground.

The state of each output can be read off from the respective LED.

Functions of the FM 351

Overview of functions

You define module functionality by setting the mode, job or function switch. The table below gives an overview of the functionality of the module.

Functionality	Description
Writing and reading machine data (Page 10463)	You can use the machine data to adapt the positioning module to the axis and the encoder.
Writing and reading incremental dimension tables (Page 10465)	The incremental dimensions define the targets controlled by the positioning module with "Relative / absolute incremental approach" mode.
End of a positioning operation (Page 10467)	The end of a positioning operation is achieved in three different ways: • Target approach • Shutdown • Cancel
"Jog" mode (Page 10473)	In "Jog" mode you can move the drive in one direction by pressing a button.
"Reference point approach" mode (Page 10476)	The "Reference point approach" mode can be used to synchronize the axis based on a recurring external event.
"Incremental approach" mode (Page 10482)	In "Incremental approach" mode, the positioning module can move the drive • toward absolute targets, • relatively by an incremental distance in a specified direction.
"Set actual value" / "Undo set actual value" job (Page 10488)	With the "Set actual value" job, you assign a new coordinate to the actual encoder value. With the "Undo set actual value" job, you reset all the operating range shifts that were generated by means of the "Set actual value" job.
"Set reference point" job (Page 10490)	With the "Set reference point" job, you synchronize the axis.
"Loop traverse" function switch (Page 10491)	You use "Loop traverse" to specify the direction in which a target is approached with a frictional connection.
"Enable input" function switch (Page 10494)	The enable input is an external input with which positioning can be enabled by an external event.
"Read position data" job (Page 10494)	With the "read position data" job you can read the incremental dimension, residual distance and velocity at the current time.
"Read encoder data" job (Page 10495)	With the "read encoder data" job you can read the current data of the encoder as well as the value for the absolute encoder adjustment.

Functionality	Description
Checkback signals for positioning (Page 10495)	The "Checkback signals for positioning" inform you of the current status of the positioning.
Checkback signals for diagnostics (Page 10496)	The "Checkback signals for diagnostics" informs you of occurring diagnostic events.

Writing and reading machine data

Writing, enabling and reading machine data

You can use the machine data to adapt the positioning module to the axis and the encoder.

You can change and read parameters during operation using the user program.

All parameters are stored in the parameter DB:

- Machine data is located in the parameter DB at the addresses 4.0 to 116.0.

Enter the parameter DB number in the corresponding channel DB.

You can enter the parameters with the DB Editor or, simply and easily, enter them in the "Drive", "Axis", "Encoder" and "Increments" dialogs before writing them to the parameter DB with the "Export" function.

You can use the "Import" function to import parameters from an existing parameter DB to the dialogs.

Initial parameter assignment

If the channel does not yet contain any machine data, proceed as follows for initial parameter assignment without the parameter assignment dialogs:

1. Enter the new values in the parameter DB and then save them.
2. Download the parameter DB to the CPU.
3. Set the following trigger bit in the channel DB for the job:
   – Write machine data (MDWR_EN = 1).
4. Call the ABS_CTRL (Page 7125) instruction in the cyclic user program.

Modifying machine data

Proceed as follows to change machine data using the user program:

1. Enter the new values in the parameter DB.
2. Set the trigger bits in the channel DB for the jobs:
   – Write machine data (MDWR_EN = 1)
   – Activate machine data (MD_EN = 1)
3. Call the ABS_CTRL (Page 7125) instruction in the cyclic user program.

If you set the trigger bits for these jobs all at once, the ABS_CTRL instruction will ensure the jobs are processed in the correct order.

Otherwise change the machine data always in the following sequence:

- Write machine data
- Enable machine data

Reading machine data

In order to read the current machine data from a channel, proceed as follows:

1. Set the following trigger bit in the channel DB:
   – Reading machine data (MDRD_EN = 1)
2. Call the ABS_CTRL (Page 7125) instruction in the cyclic user program.

The current machine data is then stored in the parameter DB on the CPU.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
35.0	MDWR_EN	BOOL	FALSE	1 = write machine data
35.1	MD_EN	BOOL	FALSE	1 = enable machine data
36.5	MDRD_EN	BOOL	FALSE	1 = read machine data

Note

If synchronization-relevant parameters have been changed, the module will carry out the following actions for the respective channel when the machine data is enabled:

- Synchronization will be deleted.
- The function switches and zero offset will be reset.
- All previous machine data and incremental dimension tables will become invalid.

Parameters relevant to synchronization:

- Axis type
- End of rotary axis
- Signal type
- Distance per encoder revolution
- Increments per encoder revolution
- Number of revolutions
- Reference point coordinate
- Absolute encoder adjustment
- Type of reference point approach
- Count direction

Writing and reading incremental dimension tables

Writing and reading incremental dimension tables

The incremental dimensions define the targets controlled by the positioning module with "Relative / absolute incremental approach" mode.

You can change and read parameters during operation using the user program.

All parameters are stored in the parameter DB:

- Increment tables are located in the parameter DB at the addresses 120.0 to 516.0.

The number of the parameter DB must be entered in the respective associated channel DB.

You can enter the parameters with the DB Editor or, simply and easily, enter them in the "Incremental dimensions" dialog before writing them to the parameter DB with the "Export" function.

You can use the function "Import" to import parameters in an existing parameter DB to the dialogs.

Initial parameter assignment

If the channel does not yet contain any increment tables, proceed as follows for initial parameter assignment without the parameter assignment dialog:

1. Enter the new values in the parameter DB and then save them.
2. Download the parameter DB to the CPU.
3. Set the trigger bits in the channel DB for the jobs:
   – Writing incremental dimension table 1 (TRGL1WR_EN = 1) and / or writing incremental dimension table 2 (TRGL2WR_EN = 1).
4. Call the ABS_CTRL (Page 7125) instruction in the cyclic user program.

Changing increment tables

Proceed as follows to change increment tables using the user program:

1. Enter the new values in the parameter DB.
2. Set the trigger bits in the channel DB for the jobs:
   – Writing incremental dimension table 1 (TRGL1WR_EN = 1) and / or writing incremental dimension table 2 (TRGL2WR_EN = 1).
3. Call the ABS_CTRL (Page 7125) instruction in the cyclic user program.

Reading increment tables

In order to read the increment tables from a channel, proceed as follows:

1. Set the trigger bits in the channel DB for the jobs:
   – Reading incremental dimension table 1 (TRGL1RD_EN = 1) and / or writing incremental dimension table 2 (TRGL2RD_EN = 1).
2. Call the ABS_CTRL (Page 7125) instruction in the cyclic user program.

The increment tables are then stored in the parameter DB on the CPU.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
35.4	TRGL1WR_EN	BOOL	FALSE	1 = write increment table 1 (1 ... 50)
35.5	TRGL2WR_EN	BOOL	FALSE	1 = write increment table 2 (51 ... 100)
36.6	TRGL1RD_EN	BOOL	FALSE	1 = read increment table 1 (1 ... 50)
36.7	TRGL2RD_EN	BOOL	FALSE	1 = read increment table 2 (51 ... 100)

Note

If synchronization-relevant parameters have been changed, the module will carry out the following actions for the respective channel when the machine data is enabled:

- Synchronization will be deleted.
- The function switches and zero offset will be reset.
- All previous machine data and incremental dimension tables will become invalid.

Parameters relevant to synchronization:

- Axis type
- End of rotary axis
- Signal type
- Distance per encoder revolution
- Increments per encoder revolution
- Number of revolutions
- Reference point coordinate
- Absolute encoder adjustment
- Type of reference point approach
- Count direction

End of a positioning operation

Definition

The end of a positioning operation is displayed by the checkback signal WORKING = 0. It can be reached in three different ways:

- Target approach
- End
- Cancel

Monitoring

The following monitoring functions are active towards the end of a positioning operation:

- Monitoring time
  The monitoring time is retriggered at the switch-off point for the last time and then loses its validity with the end of positioning.
  The end of the positioning operation must be reached within this time. If it is not, the outputs will be switched off and the operating error "Error during target approach" (error number 5) issued.
- Monitoring the target range
  The FM x51 sets a symmetrical range around each target, thereby defining the positioning precision of your application. The axis must come to a standstill within this range during a target approach. A value 0 switches off the tolerance during the target approach.
- Monitoring the standstill velocity
  The standstill velocity is used determine whether the drive comes to a standstill within the target range. The FM x51 checks whether the actual drive speed is less than the stationary drive speed when the switch-off point has been reached.
  It must be less than the standstill velocity within the target range. The FM x51 will otherwise report the operating error "Target range overrun" (error number 10).
  Falling short of the standstill velocity is monitored only once per target approach.
  Note that the velocity can briefly fall below the standstill velocity for the velocity determination of the module if the axis moves at a very slow positioning velocity (less than 2 pulses per 8 ms).
- Monitoring the stationary range
  Once a positioning operation ends, the drive is monitored to see whether it remains at an approached target position or drifts off.
  The standstill range is monitored
  - after the FM x51 has reported the checkback signal "PR"
  - if the monitoring time is exceeded
  - if the velocity falls below the standstill velocity.

  If the standstill range is exited without a valid traversing job, the FM x51 reports the "Stationary range exited" error (error number 6).

Target approach

The target approach in the "Absolute incremental approach" and "Incremental approach" modes starts when the switch-off point has been reached. As of this point the drive is switched off and the FM x51 adopts the monitoring functions.

Depending on the monitoring you have parameterized, there are a number of different scenarios in which the checkback signal "PR (POS_RCD)" is generated. The positioning operation is cancelled if no "PR (POS_RCD)" checkback signal is generated.

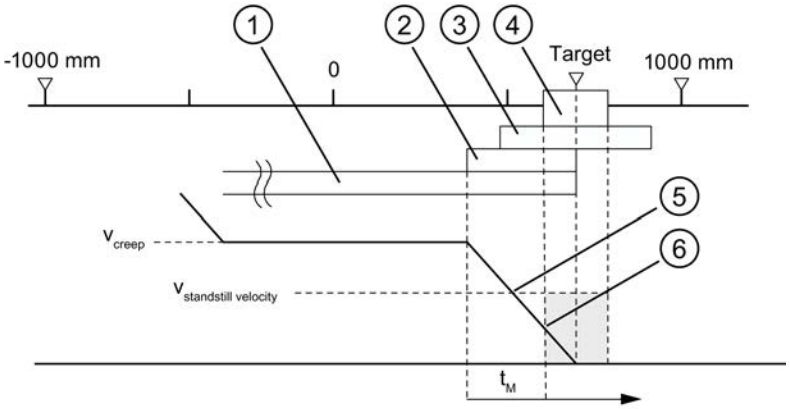
Scenario 1: You have assigned parameters as follows:

- Target range (TRG_RANGE) > 0
- Standstill velocity (ZSPEED_L) > 0
- Monitoring time (MON_TIME) > 0

PR is generated when the velocity falls below the standstill velocity and the target range has been reached. It does not matter which condition is fulfilled first.

PR is not generated if the actual value has not reached the target range within the monitoring time or the target range has been overrun without the velocity falling below the standstill velocity.

The figure below shows the target approach in incremental approach mode with a target range > 0 and standstill velocity > 0.



①	Changeover difference plus
②	Switch-off difference plus
③	Stationary range
④	Target range
⑤	Standstill velocity reached
⑥	Target range with V_{still} reached: PR is set
t_M	Monitoring time

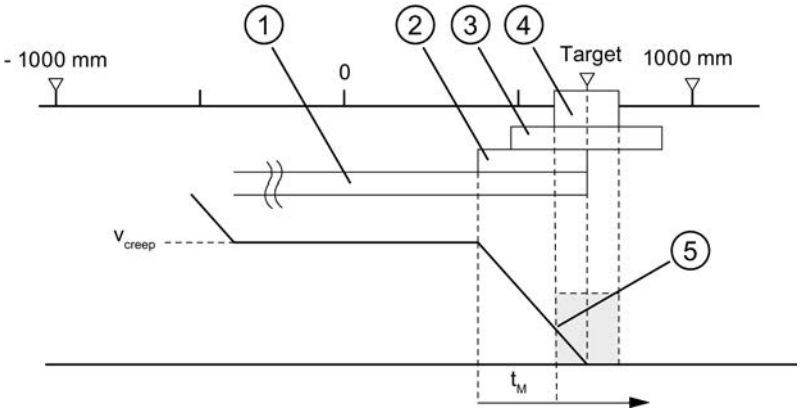
Scenario 2: You have assigned parameters as follows:

- Target range (TRG_RANGE) > 0
- Standstill velocity (ZSPEED_L) = 0
- Monitoring time (MON_TIME) > 0

PR is generated when the target range has been reached.

PR is not generated if the actual value does not reach the target range within the monitoring time.

The figure below shows the target approach in incremental approach mode with a target range > 0 and standstill velocity = 0.



①	Changeover difference plus
②	Switch-off difference plus
③	Stationary range
④	Target range
⑤	Target range with V_{still} reached: PR is set
t_M	Monitoring time

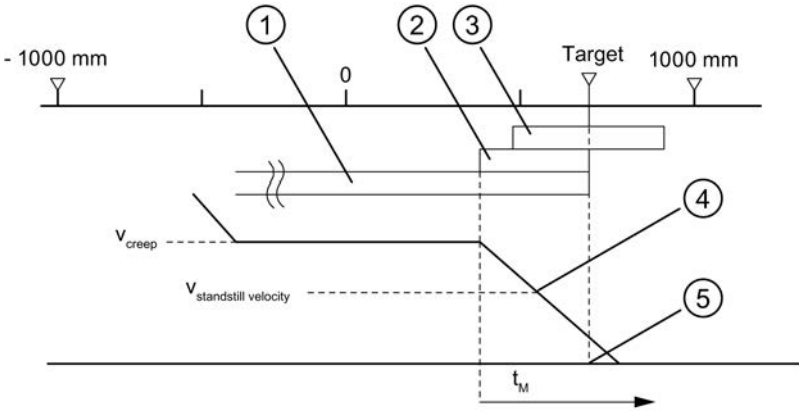
Scenario 3: You have assigned parameters as follows:

- Target range (TRG_RANGE) = 0
- Standstill velocity (ZSPEED_L) > 0
- Monitoring time (MON_TIME) > 0

PR is generated if the velocity falls below the standstill velocity and the target is then reached.

PR is not generated if the actual value has not reached the target within the monitoring time or the target range has been overrun without the velocity falling below the standstill velocity.

The figure below shows the target approach in incremental approach mode with a target range = 0 and standstill velocity > 0.



① Changeover difference plus
② Switch-off difference plus
③ Stationary range
④ Standstill velocity reached
⑤ Target reached: PR is set
t_M Monitoring time

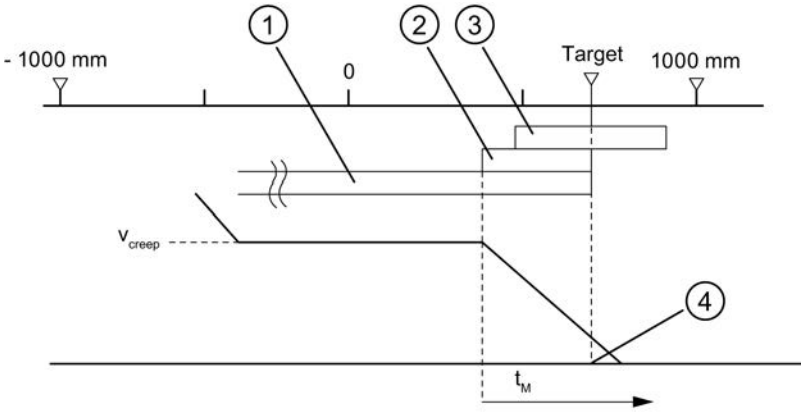
Scenario 4: You have assigned parameters as follows:

- Target range (TRG_RANGE) = 0
- Standstill velocity (ZSPEED_L) = 0
- Monitoring time (MON_TIME) > 0

PR is generated when the target has been reached.

PR is not generated if the actual value does not reach the target within the monitoring time.

The figure below shows the target approach in incremental approach mode with a target range = 0 and standstill velocity = 0.



① Changeover difference plus
② Switch-off difference plus
③ Stationary range
④ Target reached: PR is set
t_M Monitoring time

Scenario 5: You have assigned parameters as follows:

- Target range (TRG_RANGE) ≥ 0
- Standstill velocity (ZSPEED_L) ≥ 0
- Monitoring time (MON_TIME) = 0

If, in this case, the drive comes to a standstill before the target range, the end of the positioning operation will not be detected. PR will not be generated and the checkback signal WORKING will remain set. You can only cancel the positioning operation only by deleting the drive enable (DRV_EN = 0).

Shutdown without specified target

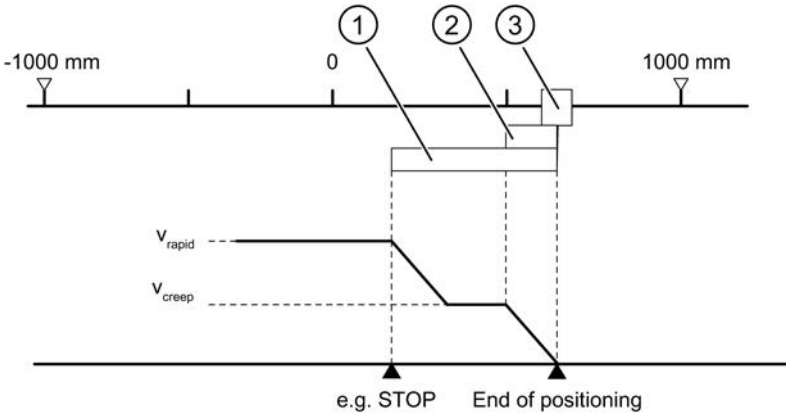
Shutdown means: The positioning operation is purposefully ended whilst complying with the differences between rapid traverse and creep speed.

The positioning operation is ended if

- The FM x51 receives a STOP signal (STOP = 1)
- The operating modes "Jog" and "Reference point approach" are ended.
- Operating errors or the operating error "Target overrun" (error number 9) occur.

The checkback signal "PR (POS_RCD)" is not set. The order of events are analogous to those of the target approach.

The figure below shows the end of a positioning operation.



① Changeover difference plus
② Switch-off difference plus
③ Target range

Cancel

Shutdown means: The positioning process is ended immediately after standstill without the use of the changeover and switch-off differences between rapid traverse and creep speed. In addition, all the relevant outputs of the respective control mode are immediately switched off and also:

- Increment = actual value
- Distance-to-go = zero

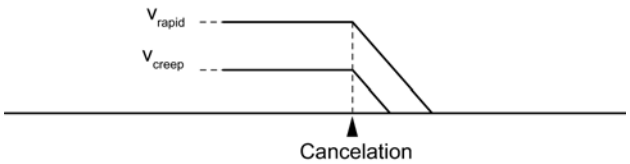
The positioning operation is cancelled if

- The drive enable signal is deleted (DRV_EN = 0).
- The CPU goes into STOP mode.
- Diagnostic errors or all operating errors apart from the "Target overrun" operating error (error number 9) occur.

The checkback signal "PR (POS_RCD)" is not be set in "Incremental approach" mode.

If the standstill velocity is assigned, the standstill monitoring will become active as soon as the velocity falls below the standstill velocity. If the standstill velocity is not assigned, the standstill monitoring will be active when the outputs have been switched off.

The figure below shows the cancellation of a positioning operation.



Data used in the parameter DB

Address	Name	Type	Initial value	Comment
76.0	TRG_RANGE	DINT	L#1000	Target range
80.0	MON_TIME	DINT	L#2000	Monitoring time
84.0	ZSPEED_R	DINT	L#1000	Stationary range
88.0	ZSPEED_L	DINT	L#30000	Standstill velocity

Checkback signals in the channel DB

Address	Name	Type	Initial value	Comment
23.1	WORKING	BOOL	FALSE	1 = Positioning operation is running
25.7	POS_RCD	BOOL	FALSE	1 = position reached

"Jog" mode

Definition

In "Jog" mode you can move the drive in one direction by pushing a button. You must install a button for both directions (plus and minus) respectively. You can use the "Jog" mode both for a synchronized as well as for an unsynchronized axis.

In the case of an unsynchronized axis, jogging is a positioning in the specified direction.

For a synchronized axis (linear axis), jogging involves positioning to the software limit switches.

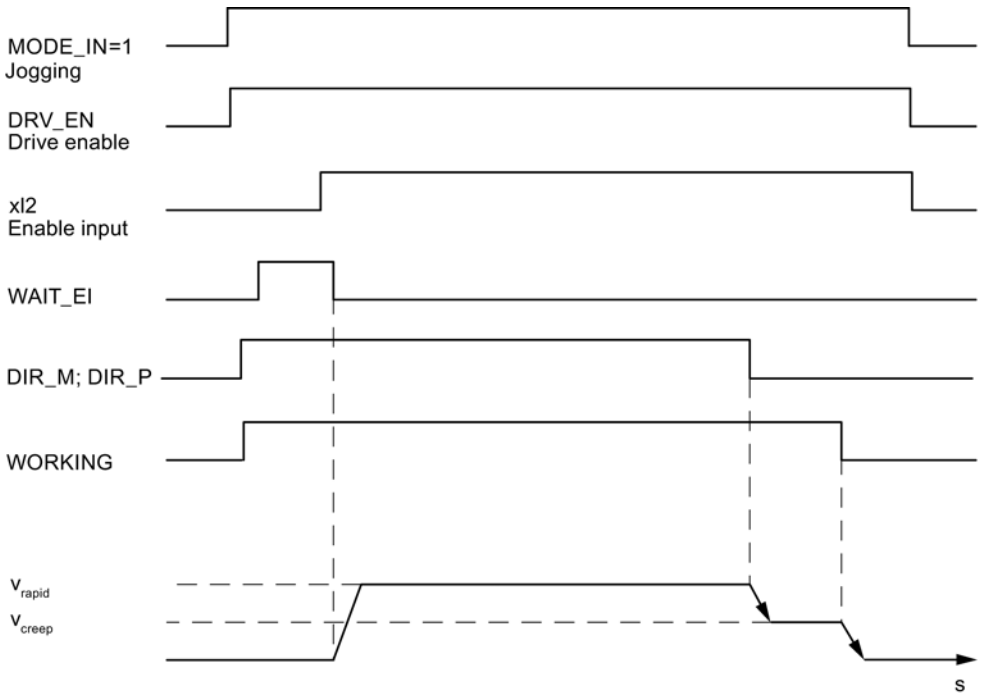
Requirement

The axis parameters must be assigned.

Sequence of events for "Jog" mode

1. Set the control signal for the "Jog" mode (MODE_IN = 1).
2. Set the control signal for the drive enable (DRV_EN = 1).
3. Set the function switch for "Do not evaluate enable input" (EI_OFF = 1) or wire the enable input for the corresponding channel.
4. Enter the start speed.
   - Rapid traverse (MODE_TYPE = 1)
   - Creep speed (MODE_TYPE = 0)
5. Set the control signal for traversing direction plus or minus (DIR_P = 1 or DIR_M = 1).
6. Call the ABS_CTRL (Page 7125) instruction.

The following diagram shows an example of "Jog" mode.



Data used in the channel DB

Address	Name	Type	Initial value	Comment
15.2	DIR_M	BOOL	FALSE	1 = Minus direction
15.3	DIR_P	BOOL	FALSE	1 = Plus direction
15.7	DRV_EN	BOOL	FALSE	1 = Turn on drive enable
16.0	MODE_IN	BYTE	B#16#0	1 = jog
17.0	MODE_TYPE	BYTE	B#16#0	1 = rapid traverse 0 = creep speed
23.0	ST_ENBLD	BOOL	FALSE	1 = Start enabled
23.1	WORKING	BOOL	FALSE	1 = Positioning operation is running
23.2	WAIT_EI	BOOL	FALSE	1 = Axis waiting for external enable
34.2	EI_OFF	BOOL	FALSE	1 = Do not evaluate enable input

Shutting down jogging

The "Jog" mode is shut down if

- You reopen the switch used for jogging (DIR_M = 0 or DIR_P = 0),
- The FM 351 receives a STOP signal (STOP = 1)
- the process value with a synchronized linear axis reaches the limit of the operating range. The axis can only continue moving in the opposite direction.

After traversing is shut down, it can continue in any direction.

Canceling jogging

"Jog" mode is cancelled if

- The drive enable signal is deleted (DRV_EN = 0)
- A traversing range limit has been overrun for a linear axis

Monitoring functions

The following monitoring functions are not active at the end of positioning in "Jog" mode:

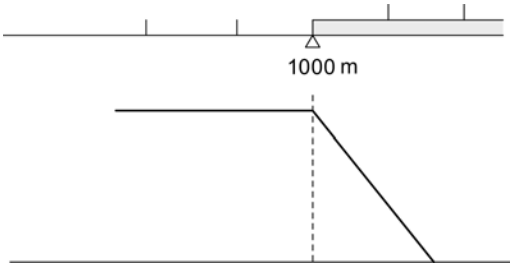
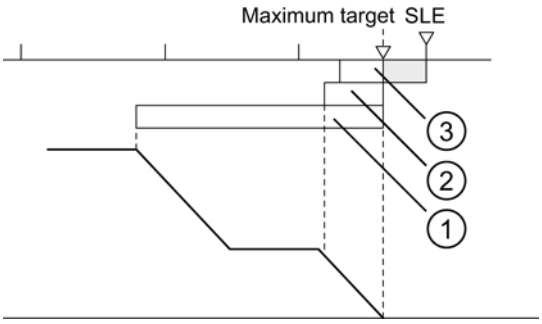
- Monitoring the stationary range
- Monitoring the target range
- Monitoring the standstill velocity

Jogging does not result in proper positioning at the end of which the "Position reached" signal is set.

Operating range limit for a linear axis

The limits for the "Jog" mode differ between a synchronized and an unsynchronized axis.

The following table shows "Jog" with synchronized and non-synchronized axes.

Axis is unsynchronized	Axis is synchronized
If, when jogging, the traversing range limit is overrun, • the process value display is no longer valid. • the positioning process is cancelled.  The process value indicated is no longer valid	Jog involves positioning to targets that are removed from the software limit switches by a distance equal to the whole target range. The operating range limits are determined from • **SLE - ½ target range** for the end of the linear axis in the plus direction • **SLE + ½ target range** for the end of the linear axis in the plus direction If you do not open the switch beforehand, the FM 351 shuts down at a target point that is positioned half the target range away from the respective software limit switch. All ranges that are necessary for proper shutdown are set by the FM 351 relative to this target point.  = Part of operating range in which a target position cannot be set ① = Changeover difference plus ② = Switch-off difference plus ③ = ½ target range

"Reference point approach" mode

Definition

The "Reference point approach" mode can be used to synchronize the axis based on a recurring external event.

Requirements

- An incremental encoder with zero mark.
- The axis parameters must be assigned:

Connections	Channel 1	Channel 2
Reference point switch	Digital input 1I0	Digital input 2I0
	The reference point switch should be such that the drive can safely brake from rapid traverse to creep speed in the range of the switch.	
Reversing switch	Digital input 1I1	Digital input 2I1
	When assigning parameters, ensure that the start of the reference point approach is assigned in the direction of the reversing switch. Only in this way can you guarantee that the reference point switch will always be found.	
Enable input	Digital input 1I2	Digital input 2I2

Sequence of events for "Reference point approach" mode

The keys for the positive and negative traversing direction must be installed for each channel.

1. Set the value for the reference point coordinate in the parameter DB REFPT.
2. Enter the type of "Reference point approach" in the parameter DB.
   The following options are available:

Start in direction ...	That leads to synchronization ...	
plus	the first zero mark after leaving the reference point switch in direction **plus**	REFPT_TYPE = 0
plus	the first zero mark after leaving the reference point switch in direction **minus**	REFPT_TYPE = 1
minus	the first zero mark after leaving the reference point switch in direction **plus**	REFPT_TYPE = 2
minus	the first zero mark after leaving the reference point switch in direction **minus**	REFPT_TYPE = 3

3. Enter the start speed:
   - Rapid traverse (REFPT_SPD = 0)
   - Creep speed (REFPT_SPD = 1)
4. Write and enable the machine data.
5. Set the control signal for the "Reference point approach" mode (MODE_IN = 3).
6. Set the control signal for the drive enable (DRV_EN = 1).
7. Set the function switch for "Do not evaluate enable input" (EI_OFF = 1) or wire the enable input for the corresponding channel.
8. Set the control signal for traversing direction plus or minus or the start signal (DIR_P = 1, DIR_M = 1 or START = 1)
9. Call the ABS_CTRL (Page 7125) instruction.

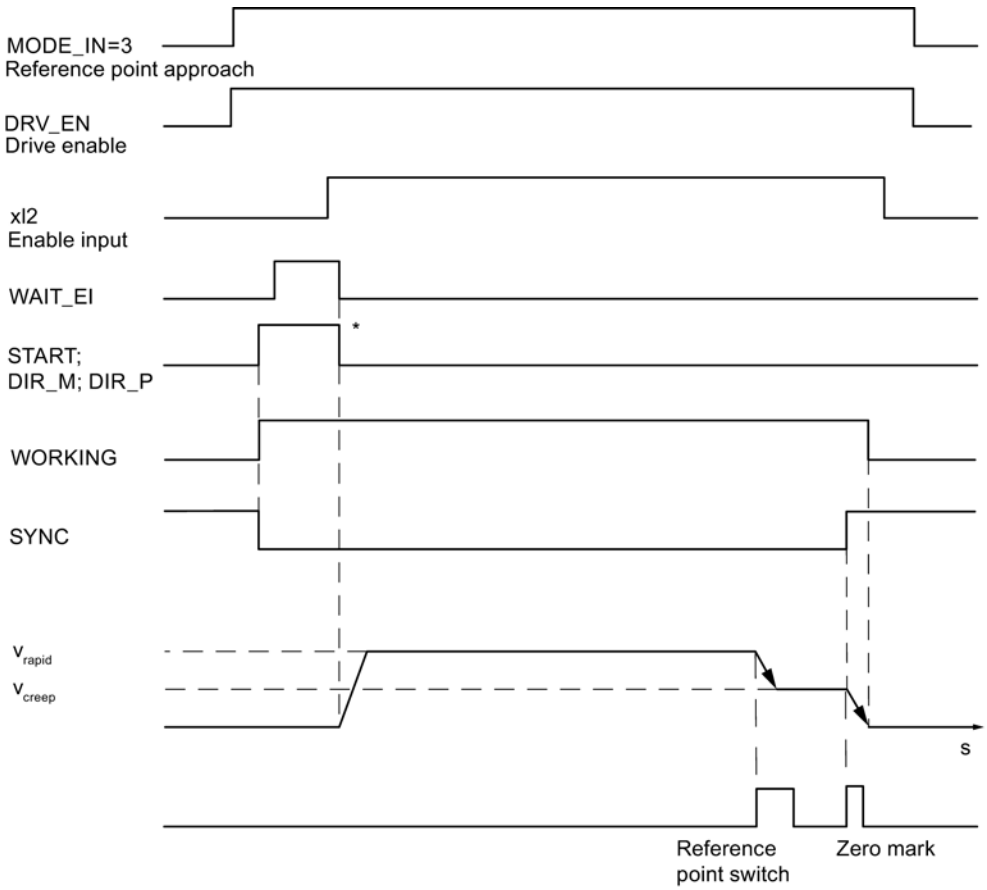
The following table shows the start commands for a reference point approach.

Start command	Job	Comment
DIR_P	The drive starts in the direction with positive values. It therefore moves in the direction of the traversing range end.	The FM 351 will signal an operating error if a negative direction is entered in the machine data. No reference point approach is executed.
DIR_M	The drive starts in the direction with negative values. It therefore moves in the direction of the traversing range start.	If a positive direction is entered in the machine data, the FM 351 will signal an operating error. No reference point approach is executed.
START	The drive starts in the direction that is entered in the machine data.	

Note

The following applies to the rotary axis: Reproducibility of the reference point is only ensured if the ratio of the **End of rotary axis** value and the **Distance per encoder revolution** value is an integer.

The following figure shows an example of "Reference point approach" mode.



* The start signals are reset by the ABS_CTRL (Page 7125) instruction.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
15.0	START	BOOL	FALSE	1 = Start positioning operation
15.2	DIR_M	BOOL	FALSE	1 = Minus direction
15.3	DIR_P	BOOL	FALSE	1 = Plus direction
15.7	DRV_EN	BOOL	FALSE	1 = Turn on drive enable
16.0	MODE_IN	BYTE	B#16#0	3 = reference point approach
23.0	ST_ENBLD	BOOL	FALSE	1 = Start enabled
23.1	WORKING	BOOL	FALSE	1 = Positioning operation is running
23.2	WAIT_EI	BOOL	FALSE	1 = Axis waiting for external enable
25.0	SYNC	BOOL	FALSE	1 = axis is synchronized
34.2	EI_OFF	BOOL	FALSE	1 = Do not evaluate enable input

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
44.0	REFPT	DINT	L#0	Reference point coordinate
52.0	REFPT_TYPE	DINT	L#0	Type of reference point approach
99.0	REFPT_SPD	BOOL	TRUE	Start velocity for the reference point approach 0 = rapid traverse 1 = creep speed

Effects of the mode

- Synchronization is canceled when traversing starts.
- The actual position is set to the value of the reference point coordinate when the SYNC feedback signal is set.
- The operating range is established on the axis.
- The individual points within the operating range maintain their original value, but are located at different positions.

Canceling the reference point approach

The "Reference point approach" mode is cancelled if

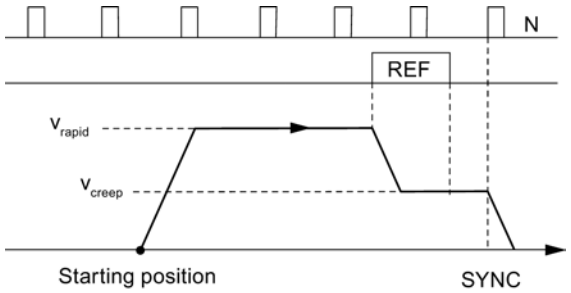
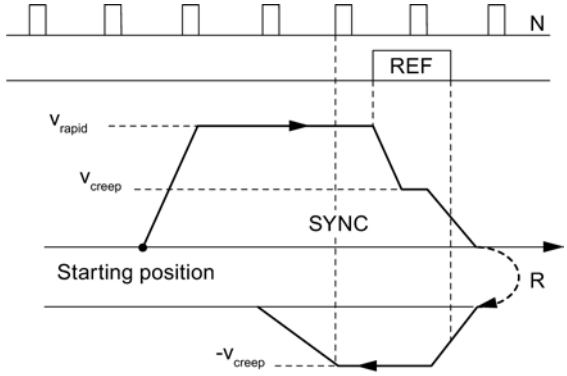
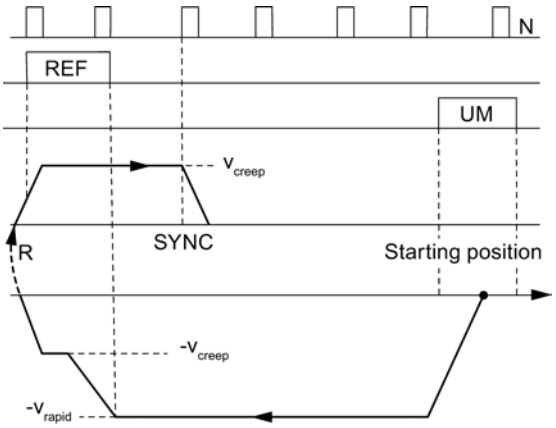
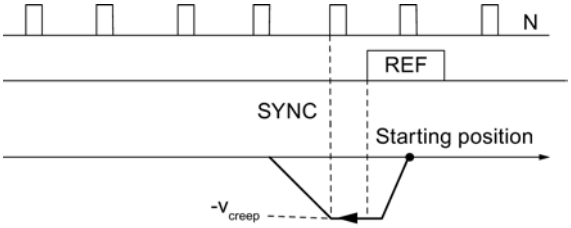
- The drive enable signal is deleted (DRV_EN = 0)
- A traversing range limit has been overrun for a linear axis

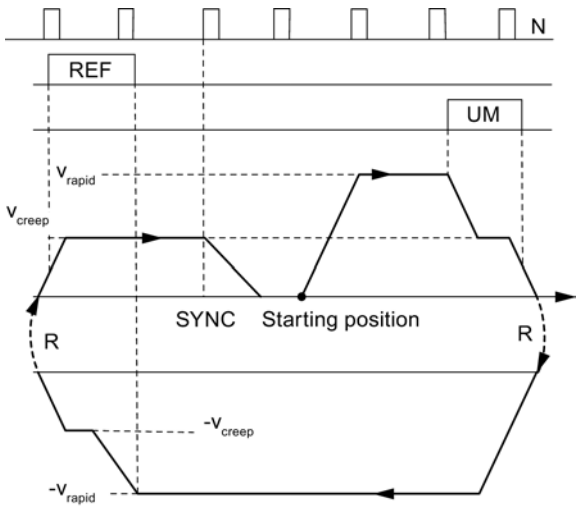
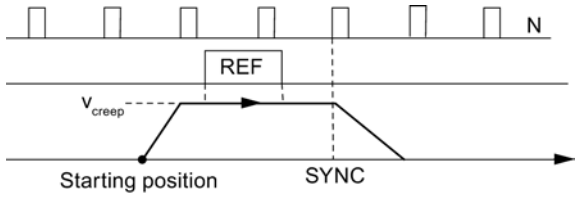
Reference point approach in dependence on the start position

In the case of a reference point approach, you must distinguish between different cases that are dependent on

- the location of the drive at the time a reference point approach is started,
- the assigned start direction,
- the assigned location of the zero mark relative to the reference point switch.

The following table shows the REFPT_TYPE = 0 and REFPT_TYPE = 1 cases. The figures apply accordingly to REFPT_TYPE = 2 and REFPT_TYPE = 3.

Conditions of the reference point approach	Sequence of events for the reference point approach
Example of reference point approach (REFPT_TYPE = 0): • Start direction is plus. • The location of the zero mark of the reference point switch is assigned in the plus direction.	
Example of reference point approach (REFPT_TYPE = 1): • Start direction is plus. • The location of the zero mark of the reference point switch is assigned in the minus direction.	
Example of reference point approach (REFPT_TYPE = 0): • The start direction must be assigned in the plus direction. • The location of the zero mark of the reference point switch is assigned in the plus direction. • The reversing switch is more positive than the reference point switch.	
Example of reference point approach (REFPT_TYPE = 1): • Start direction is plus. • The location of the zero mark of the reference point switch is assigned in the minus direction. • The start position of the reference point approach is on the reference point switch.	

Conditions of the reference point approach	Sequence of events for the reference point approach
Example of reference point approach (REFPT_TYPE = 0): • Start direction is plus. • The location of the zero mark of the reference point switch is assigned in the plus direction. • The reversing switch is more positive than the reference point switch.	
Example of reference point approach (REFPT_TYPE=0): • Start direction is plus. • The location of the zero mark of the reference point switch is assigned in the plus direction. • Start velocity = creep speed	
R = Direction reversal REF = Reference point switch UM = Reversing switch N = Zero mark of the encoder SYNC = Synchronization has been achieved	

"Incremental approach" mode

Definition

In "Incremental approach" mode, the FM 351 can move the drive.

- toward **absolute** targets,
- **relatively** by an incremental distance in a specified direction.

You specify the target or the relative distances as incremental dimensions of the FM 351. You can enter up to 100 incremental dimensions in a table; these are valid for both "Relative incremental approach" mode and "Absolute incremental approach" mode. Independently of the incremental dimension table, you can specify the distance using incremental dimension 254 (Page 10526) and incremental dimension 255 (Page 10526).

Requirements

- The axis parameters must be assigned.
- The axis must be synchronized.
- The increments must be available on the module.

Interpretation of the increments

The FM 351 interprets the specifications differently depending on the incremental approach you select.

- Absolute incremental approach: The increments will be interpreted as an absolute target position.
- Relative incremental approach: The increments will be interpreted as a relative distance from the start position.

Note

Only positive increments are permissible for the "Relative incremental approach" mode.

The incremental dimensions result from the DIR_P or DIR_M direction specifications.

"Incremental approach" mode sequence with incremental dimension number 1 to 100

Step	Absolute incremental approach	Relative incremental approach
	Incremental dimension number 1 to 100	
1	Set the control signal for the "Absolute incremental approach" mode (MODE_IN = 5).	Set the control signal for the "Relative incremental approach" mode (MODE_IN = 4).
2	Enter the incremental dimensions in the tables (TRGL1; TRGL2).	
3	Write the incremental dimension table (TRGL1/2WR_EN = 1).	
4	Set the control signal for the drive enable (DRV_EN = 1).	
5	Set the function switch for "Do not evaluate enable input" (EI_OFF = 1) or wire the enable input for the corresponding channel.	
6	Enter the incremental dimension number (MODE_TYPE = 1 … 100).	
7	Set the control signal: • Linear axis: – START; the direction is determined exactly by the target and the current process value. • Rotary axis: – START; the target is approached by the shortest route. – DIR_P; start in plus direction – DIR_M; start in minus direction	Set the control signal: • Linear axis: – DIR_P; start in plus direction – DIR_M; start in minus direction • Rotary axis: – DIR_P; start in plus direction – DIR_M; start in minus direction

Step	Absolute incremental approach	Relative incremental approach
	Incremental dimension number 1 to 100	
8	Call the ABS_CTRL (Page 7125) instruction.	
Steps 2 and 3 are only required if no increments are given or the existing increments are to be changed.		

Sequence of events for the "incremental approach" mode with increment number 254

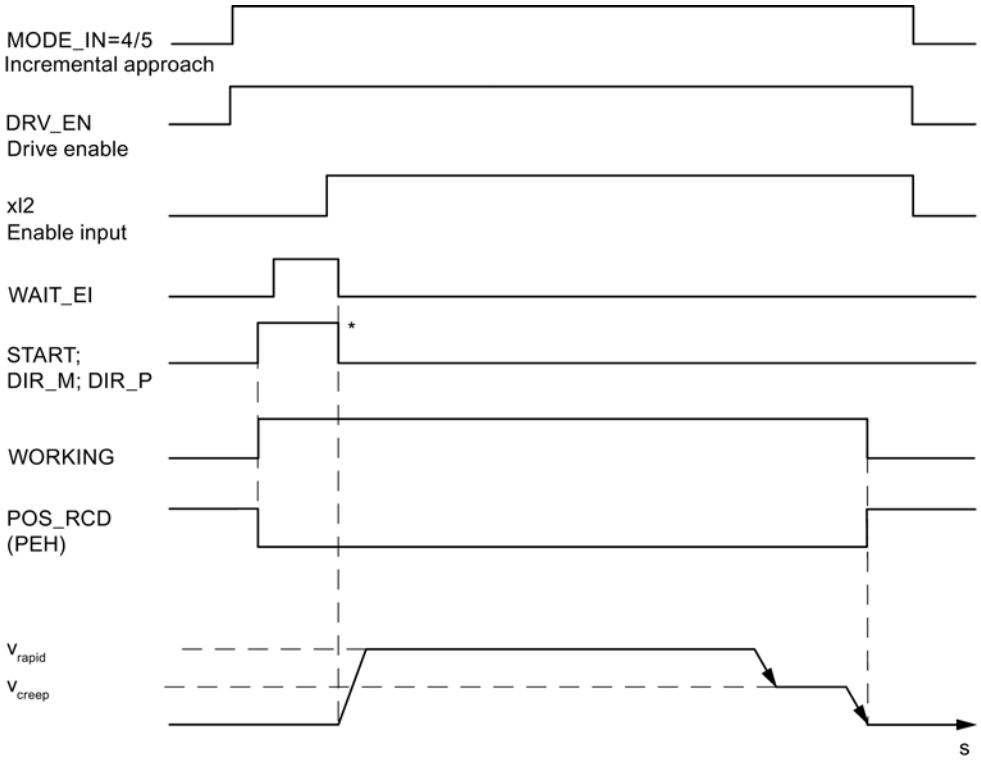
Step	Absolute incremental approach	Relative incremental approach
	Increment number 254	
1	Set the control signal for the "Absolute incremental approach" mode (MODE_IN = 5).	Set the control signal for the "Relative incremental approach" mode (MODE_IN = 4).
2	Set the control signal for the drive enable (DRV_EN = 1).	
3	Set the function switch for "Do not evaluate enable input" (EI_OFF = 1) or wire the enable input for the corresponding channel.	
4	Enter the incremental dimension number (MODE_TYPE = 254).	
5	Enter the incremental dimension for the incremental dimension number 254 (TRG252_254).	
6	Set the trigger bit for writing the incremental dimension (TRG252_254_EN = 1).	
7	Set the control signal: • Linear axis: – START; the direction is determined exactly by the target and the current process value. • Rotary axis: – START; the target is approached by the shortest route. – DIR_P; start in plus direction – DIR_M; start in minus direction	Set the control signal: • Linear axis: – DIR_P; start in plus direction – DIR_M; start in minus direction • Rotary axis: – DIR_P; start in plus direction – DIR_M; start in minus direction
8	Call the ABS_CTRL (Page 7125) instruction.	

Sequence of events for the "incremental approach" mode with increment number 255

Step	Absolute incremental approach	Relative incremental approach
	Increment number 255	
1	Set the control signal for the "Absolute incremental approach" mode (MODE_IN = 5).	Set the control signal for the "Relative incremental approach" mode (MODE_IN = 4).
2	Set the control signal for the drive enable (DRV_EN = 1).	
3	Set the function switch for "Do not evaluate enable input" (EI_OFF = 1) or wire the enable input for the corresponding channel.	
4	Enter the incremental dimension number (MODE_TYPE = 255).	
5	Enter the incremental dimension for the incremental dimension number 255 (TRG255).	
6	Enter the value for the changeover difference of the incremental dimension number 255 (CHGDIF255).	

Step	Absolute incremental approach	Relative incremental approach
	Increment number 255	
7	Enter the value for the changeover difference of the incremental dimension number 255 (CUTDIF255).	
8	Set the trigger bit for writing the incremental dimension, switch-off difference, and changeover difference (TRG255_EN = 1).	
9	Set the control signal: • Linear axis: – START; the direction is determined exactly by the target and the current process value. • Rotary axis: – START; the target is approached by the shortest route. – DIR_P; start in plus direction – DIR_M; start in minus direction	Set the control signal: • Linear axis: – DIR_P; start in plus direction – DIR_M; start in minus direction • Rotary axis: – DIR_P; start in plus direction – DIR_M; start in minus direction
10	Call the ABS_CTRL (Page 7125) instruction.	

The following figure shows an example of "Incremental approach" mode.



* The start signals are reset by the ABS_CTRL (Page 7125) instruction.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
15.0	START	BOOL	FALSE	1 = start positioning
15.2	DIR_M	BOOL	FALSE	1 = Minus direction
15.3	DIR_P	BOOL	FALSE	1 = Plus direction
15.7	DRV_EN	BOOL	FALSE	1 = Turn on drive enable
16.0	MODE_IN	BYTE	B#16#0	4 = relative incremental approach 5 = absolute incremental approach
17.0	MODE_TYPE	BYTE	B#16#0	Incremental dimension number 1 … 100, 254 or 255
23.0	ST_ENBLD	BOOL	FALSE	1 = Start enabled
23.1	WORKING	BOOL	FALSE	1 = positioning is active
23.2	WAIT_EI	BOOL	FALSE	1 = Axis waiting for external enable
25.7	POS_RCD	BOOL	FALSE	1 = Position reached
34.2	EI_OFF	BOOL	FALSE	1 = Do not evaluate enable input
36.2	TRG252_254_EN	BOOL	FALSE	1 = write increment for increment number 254
36.3	TRG255_EN	BOOL	FALSE	1 = write increment for increment number 255
35.4	TRGL1WR_EN	BOOL	FALSE	1 = Write incremental dimension table 1 (incremental dimension number 1...50)
35.5	TRGL2WR_EN	BOOL	FALSE	1 = Write incremental dimension table 2 (incremental dimension number 51...100)
96.0	TRG252_254	DINT	L#0	Increment for increment number 254
100.0	TRG255	DINT	L#0	Increment for increment number 255
104.0	CHGDIF_255	DINT	L#0	Changeover difference for increment number 255
108.0	CUTDIF_255	DINT	L#0	Switch-off difference for increment number 255

Data used in the parameter DB

Address	Name	Type	Initial value	Comment	
100.0	CHGDIF_P	DINT	L#5000	Changeover difference plus	
104.0	CHGDIF_M	DINT	L#5000	Changeover difference minus	
108.0	CUTDIF_P	DINT	L#2000	Switch-off difference plus	
112.0	CUTDIF_M	DINT	L#2000	Switch-off difference minus	
120.0	TRGL1.TRG[1]	DINT	L#0	Increment number 1	Increment table 1
. . .	. . .	. . .	. . .	. . .	
316.0	TRGL1.TRG[50]	DINT	L#0	Increment number 50	
320.0	TRGL2.TRG[51]	DINT	L#0	Increment number 51	Increment table 2
. . .	. . .	. . .	. . .	. . .	
516.0	TRGL2.TRG[100]	DINT	L#0	Increment number 100	

Distance-to-go

The residual distance is the difference between the target (increment) and process value. The residual distance is

- positive if the target has not yet been reached
- negative if the target has already been overrun.

In the case of a rotary axis, the displayed distance-to-go cannot be used.

Shutting down the Incremental approach

The "Incremental approach" mode is shut down when the FM 351 receives a STOP signal (STOP = 1).

After traversing is shut down, the distance-to-go remains.

The remaining distance-to-go with a "relative incremental approach" can be traveled to the end if

- the mode is unchanged, and
- the increment number is unchanged, and
- the direction is unchanged, and
- the remaining distance-to-go is greater than the assigned switch-off difference.

You travel the distance-to-go by starting the "relative incremental approach" without any changes.

Canceling the incremental approach

The "Incremental approach" mode is cancelled when the "Drive enable" signal is deleted (DRV_EN = 0).

Delete residual distance

With the job "Delete residual distance" you can delete any awaiting residual distance.

You also delete the awaiting residual distance by calling a different mode or by starting the mode in the other direction.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
35.2	DELDIST_EN	BOOL	FALSE	1 = Delete residual distance

"Set actual value" / "Undo set actual value" job

Definition

The job "Set actual value" assigns a new coordinate to the current encoder state. The operating range is projected on a different area of the axis.

You determine the shift in the operating range with (ACT_{new}- $ACT_{current}$).

- $ACTUAL_{new}$ is the default value
- $ACTUAL_{current}$ is the actual value at the time of the execution

Requirements

- The axis parameters must be assigned.
- The axis must be synchronized.

Job sequence

1. Enter the coordinate AVAL for the actual value (ACT_{new}) again.
   - Linear axis:
     You must select the specified actual value so that the software limit switches are still within the permissible traversing range after the job is called.
     The amount of the shift resulting from (ACT_{new} - $ACT_{current}$) must be less than or equal to the amount of the permissible traversing range (max. 100 m or 1000 m).
   - Rotary axis:
     Rule for the defined actual value:
     0 ≤ actual value < end of rotary axis
2. Set the relevant trigger bit (AVAL_EN = 1).

If issued during a positioning, the "Set actual value" job is held until the end of positioning and only carried out after the subsequent block call.

Data used in the channel DB

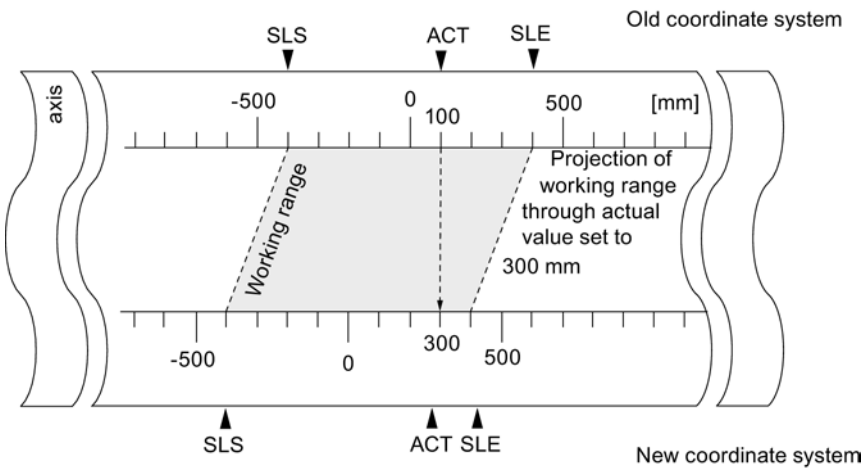
Address	Name	Type	Initial value	Comment
35.7	AVAL_EN	BOOL	FALSE	1 = set actual value
84.0	AVAL	DINT	L#0	Coordinate for "Set actual value"

Effects of the job

In the example "set actual value " to 300 mm you can see how this job projects the operating range onto a specified position of the axis. This has the following results:

- The actual position is set on the value of the actual value coordinate.
- The operating range is shifted on the axis.
- The individual points (e.g., software limit switch end) within the operating range maintain their original value, but have new positions.

The following table shows the shift in the axis operating range by "set reference point".

Set actual value	SLS	ACT UAL	SLE
	-400	100	400
	-400	300	400

Withdrawing the job ("Undo set actual value")

The job "Undo set actual value" resets all the operating range shifts that were created by means of "Set actual value".

The total of all operating range shifts must not exceed the traversing range, so that this job can be correctly executed.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
35.3	AVALREM_EN	BOOL	FALSE	1 = cancel set actual value

"Set reference point" job

Definition

You use the "set reference point" job to synchronize the axes. The job shifts the operating range. All the shifts that were created with "Set actual value" are retained.

The setting projects the operating range on the axis. Therefore, by entering different values, the operating range can be moved to any position within the physical area of the axis.

Prerequisites

- The positioning must have ended.
- The axis parameters must be assigned.

Job sequence

1. Enter the value REFPT for the reference point coordinate.
   - Linear axis:

     The reference point coordinates must not lie outside the software limit switches. This also applies to the reference point coordinate in a shifted coordinate system.
   - Rotary axis:
     Rule for the reference point coordinate:
     0 ≤ reference point coordinate < end of rotary axis
2. Set the relevant trigger bit (REFPT_EN = 1).

Data used in the channel DB

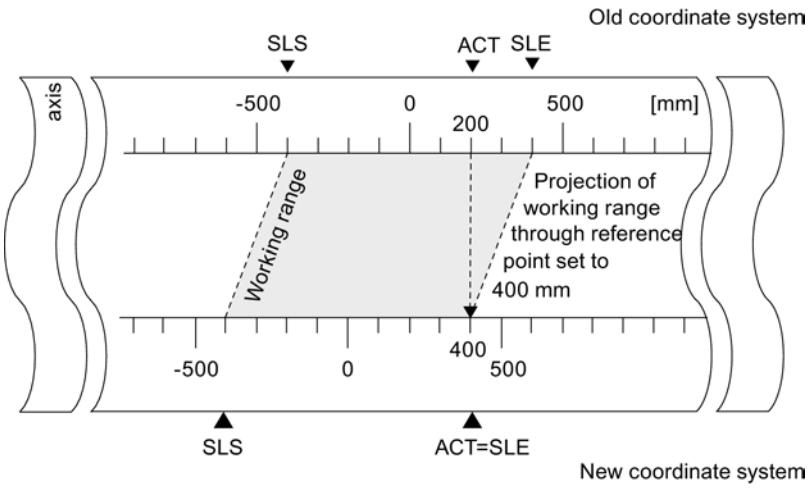
Address	Name	Type	Initial value	Comment
25.0	SYNC	BOOL	FALSE	1 = axis is synchronized
35.6	REFPT_EN	BOOL	FALSE	1 = set reference point
92.0	REFPT	DINT	L#0	Reference point coordinate

Effects of the job

In the example "set reference point" to 400 mm you can see how this job projects the operating range onto a specific, physical position of the axis. This has the following results:

- The actual position is set on the value of the reference point coordinate.
- The operating range is shifted on the axis.
- The individual points (e.g. software limit switch end) maintain their original value, but are located at new positions.
- The SYNC bit in the checkback signals is set.

The following table shows how "Set reference point" shifts the operating range on the axis.

Set reference point	SLS	ACTUAL	SLE
	-400	200	400
	-400	400	400

Special features of the absolute encoder

This job is necessary for an absolute encoder adjustment (Page 10513).

"Loop traverse" function switch

Definition

With "loop approach" you specify the direction in which a target is approached with a frictional connection. You can use loop approach if a frictional connection between the motor and axis can only be ensured in one direction.

A target that is approached in the direction opposite the specified direction is initially overrun. Then, the module performs a direction reversal and approaches the target in the specified direction.

Requirements

- The axis parameters must be assigned.
- The axis must be synchronized.
- In the case of a loop approach in the opposite direction to that of the traversing direction of the target the maximum target position is:
  - traversing direction plus
    Target < SLE - ½ Target range - Switch-off difference plus - Changeover difference minus
  - traversing direction minus
    Target < SLE - ½ Target range + Switch-off difference minus + Changeover difference minus

- A loop approach will not be carried out if the target is being approached in the direction of the loop approach. In this case, an incremental approach without an about-turn in direction will be carried out.
- The sequence of the "incremental approach" (Page 10482) operating mode must be known.

Sequence of events for the loop approach

1. Set the control signal for the "absolute/relative incremental approach" operating mode (MODE_IN = 5/4).
2. Set the control signal for the drive enable (DRV_EN = 1).
3. Set the "Enable input" (EI_OFF = 1) function switch or wire the enable input for the corresponding channel.
4. Enter the incremental dimension number (MODE_TYPE = 1 … 100, 254, 255).
5. Set the "Loop traverse" function switch (PLOOP_ON / MLOOP_ON = 1).
6. Start the incremental approach.

Data used in the channel DB

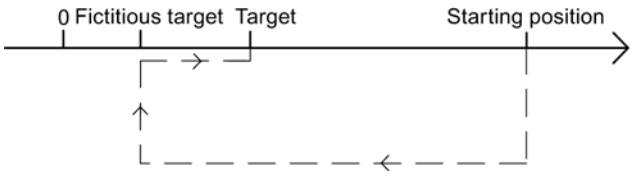
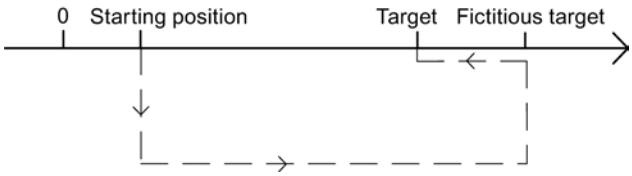
Address	Name	Type	Initial value	Comment
15.0	START	BOOL	FALSE	1 = start positioning
15.2	DIR_M	BOOL	FALSE	1 = minus direction
15.3	DIR_P	BOOL	FALSE	1 = plus direction
15.7	DRV_EN	BOOL	FALSE	1 = switch on drive enable
16.0	MODE_IN	BYTE	B#16#0	4/5 = relative / absolute incremental approach
17.0	MODE_TYPE	BYTE	B#16#0	Incremental dimension number 1 … 100, 254 or 255
34.0	PLOOP_ON	BOOL	FALSE	1 = loop approach in plus direction
34.1	MLOOP_ON	BOOL	FALSE	1 = loop approach in minus direction
34.2	EI_OFF	BOOL	FALSE	1 = do not evaluate enable input

Fictitious target

If you start a positioning to a target that is located in the direction opposite to that of the assigned loop traverse, the module determines a fictitious target for this target. The module performs a direction reversal at the fictitious target and then approaches the target in the proper direction.

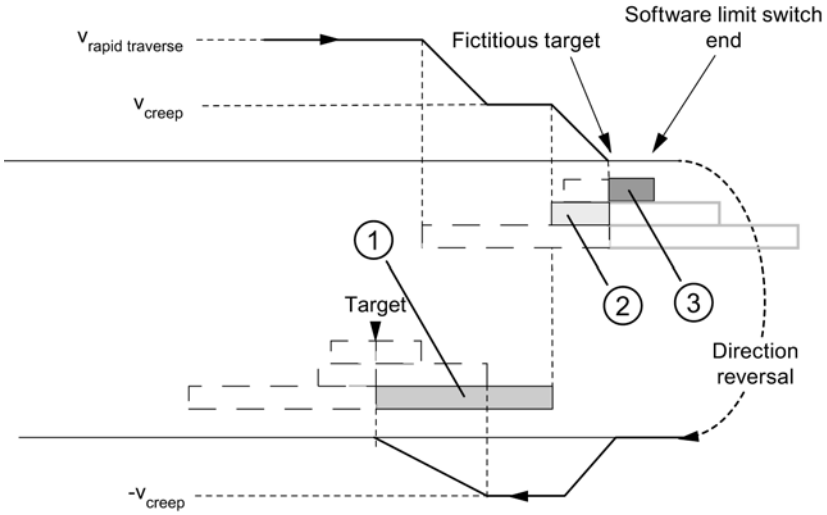
This fictitious target **must** be located by at least half the target range before the respective software limit switch.

The distance of the fictitious target from the parameterized target is established depending on the direction: The following table shows the calculation of the location of the fictitious target with a loop traverse.

Guidelines	Location of fictitious target
Parameter assignment: Loop + (frictional connection plus) and approach in minus direction.	The fictitious target (**$target_f$**) has the value: **$target_f$ = target - switch-off difference minus - changeover difference plus** 
Parameter assignment: Loop - (frictional connection minus) and approach in plus direction.	The fictitious target (**$target_f$**) has the value: **$target_f$ = target + switch-off difference plus + changeover difference minus** 

Example

Using a positioning with loop approach minus to a maximum target we can illustrate the location of the fictitious target.



① Changeover difference negative

② Switch-off difference positive

③ ½ target range

"Enable input" function switch

Definition

The enable input is an external input with which a positioning can be enabled by an external event.

Evaluate enable input (EI_OFF = 0)

The respective enable input (xI2) must be wired for the channel.

With this you have the possibility of preparing the start of a positioning. Start the positioning independently of the program flow of your user program by setting a "1" signal on the enable input.

Traversing begins when you create a "1" signal at the enable input and is ended when you create a "0" signal at the enable input.

Do not evaluate enable input (EI_OFF = 1)

If you switch off the evaluation of the enable input, an operating mode starts immediately after the start signal is detected. Then it is not possible to prepare an operating mode and start it at a defined later time.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
34.2	EI_OFF	BOOL	FALSE	1 = do not evaluate enable input

"Read position data" job

Definition

With the "read position data" job you can read the increment, distance-to-go and speed at the current time.

Job sequence

1. Set the trigger bit at the DB channel (ACTSPD_EN = 1).
2. The data are then stored in the channel DB.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
37.1	ACTSPD_EN	BOOL	FALSE	1 = read position data
112.0	ACTSPD	DINT	L#0	Current speed
116.0	DIST_TO_GO	DINT	L#0	Distance-to-go
120.0	ACT_TRG	DINT	L#0	Current increment

"Read encoder data" job

Definition

With the "read encoder data" job you can read the current data of the encoder as well as the value for the absolute encoder adjustment.

Requirements

You can read the value for the absolute encoder adjustment (Page 10513) after you have carried out the "set reference point" (Page 10490) job.

Job sequence

1. Set the trigger bit at the DB channel (ENCVAL_EN = 1).
2. The data is stored in the channel DB.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
37.2	ENCVAL_EN	BOOL	FALSE	1 = read encoder values
124.0	ENCVAL	DINT	L#0	Actual encoder value (internal representation)
128.0	ZEROVAL	DINT	L#0	Last zero mark value (internal representation)
132.0	ENC_ADJ	DINT	L#0	Absolute encoder adjustment

Querying checkback signals for the positioning

Definition

The "checkback signals for the positioning" inform you of the current status of the positioning.

Sequence

1. The data is stored in the channel DB each time the ABS_CTRL (Page 7125) instruction is called.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
23.0	ST_ENBLD	BOOL	FALSE	1 = Start enabled
23.1	WORKING	BOOL	FALSE	1 = positioning is active
23.2	WAIT_EI	BOOL	FALSE	1 = Axis waiting for external enable
23.4	SPEED_OUT	BOOL	FALSE	0 = creep speed 1 = rapid traverse
23.5	ZSPEED	BOOL	FALSE	1 = axis is located in the standstill range
23.6	CUTOFF	BOOL	FALSE	1 = axis is located in the switch-off range
23.7	CHGOVER	BOOL	FALSE	1 = axis is located in the changeover range
24.0	MODE_OUT	BYTE	B#16#0	Active operating mode
25.2	GO_M	BOOL	FALSE	1 = axis moves in minus direction
25.3	GO_P	BOOL	FALSE	1 = axis moves in plus direction
25.7	POS_RCD	BOOL	FALSE	1 = Position reached
26.0	ACT_POS	DINT	L#0	Current process value (current position of the axis)

Querying checkback signals for diagnostics

Definition

The "checkback signals for the diagnostics" informs you of occurring diagnostic events.

Sequence

1. If the module enters a new event in the diagnostic buffer, it sets the DIAG bit in all the channels in the feedback interface. Error (Page 10538) events are logged in the diagnostics buffer. The DIAG bit is also set when the diagnostic buffer is deleted.
2. If it is not possible to call an operating mode or control an active operating mode, or if this has been carried out incorrectly, the module sets an OT_ERR operator error in the feedback interface. The cause of error is then entered in the diagnostic buffer. While the operator error exists, you can neither start a new operating mode nor continue with the halted one. You acknowledge an existing operating error with OT_ERR_A = 1.
3. The module sets the DATA_ERR bit in the feedback interface when it detects incorrect data in a write job. The cause of error is then entered in the diagnostic buffer.
4. The checkback signals are stored in the channel DB.
5. When the diagnostic buffer is read, the module sets the DIAG bit back to 0 in all channels.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
22.2	DIAG	BOOL	FALSE	1 = delete diagnostic buffer
22.3	OT_ERR	BOOL	FALSE	1 = operator error
22.4	DATA_ERR	BOOL	FALSE	1 = data error

Configuring FM 351

Parameter assignment FM 351

Specifying channels

Selecting system of units

Selecting system of units

For the input and output of the data, you can select from the following systems of units:

- Millimeter (default setting)
- Inches
- Degrees

Note

If you change the system of units in the "Specify channels" screen, the values will be converted into the new system. This can cause round-off errors.

If you change the system of units via the machine date, the values are not automatically converted.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
8.0	UNITS	DINT	L#1	**System of units** Valid values: • 1 = 10^{-3} mm • 2 = 10^{-4} inch • 3 = 10^{-4} degrees • 4 = 10^{-2} degrees • 6 = 10^{-3} degrees

Standard system of units in the online help

To specify the minimum and maximum values we use the mm system of units. To define the limits in other systems of units, convert the values as shown below:

To convert from		calculate
mm → inch		Limit value(inches) = limit value (mm) × 0.1 [1]
mm → degrees	10^{-4} (4 post-decimal places)	Limit value (degrees) = limit value (mm) x 0.1
	10^{-3} (3 post-decimal places)	Limit value (degrees) = limit value (mm) x 1
	10^{-2} (2 post-decimal places)	Limit value (degrees) = limit value (mm) x 10
inches → degrees		Limit (degrees) = limit (inches) .

[1] When set to the maximum value, the number of decimal places affects the number of digits before the decimal point. In the inch measurement system, 4 decimal places are used; you can therefore enter a maximum of 100,000.0000 inches. The millimeter measurement system uses 3 decimal places; a maximum of 1,000,000.000 mm can be entered.

Connection between increments and system of units

The encoder signals of a connected encoder will be evaluated by the module and converted into the current system of units. The resolution (Page 10519) is used for the conversion.

If the positioning module

- has counted 10 increments and
- a resolution of 100 µm per increment is assigned in the encoder data,

this means that the axis will be moved by a distance of 1 mm.

Copying channels

Copying channels

You can copy the parameters of one channel to another.

1. Select the channel whose parameters are to be copied (source channel).
2. Select the channel to which the parameters are to be copied (target channel).
3. Press the "Copy" button.

Channel n drive

Selecting control mode

Overview of control modes

Overview

The control mode specifies how a motor is operated at the four digital outputs (Q) of a channel by means of the power module. There are 4 control modes per channel for the positioning module:

- Control mode 1 (Page 10500)
- Control mode 2 (Page 10500)
- Control mode 3 (Page 10501)
- Control mode 4 (Page 10502)

The following table shows the states of the 4 digital outputs for each control mode, x stands for channel 1 and 2.

	Rapid traverse		Creep speed		PR Position reached stop
	Direction +	Direction -	Direction +	Direction -	
Control mode 1					
xQ0	1	1	0	0	-
xQ1	0	0	1	1	-
xQ2	1	0	1	0	-
xQ3	0	1	0	1	-
Control mode 2					
xQ0	1	1	0	0	0
xQ1	0	0	0	0	1
xQ2	1	0	1	0	0
xQ3	0	1	0	1	0
Control mode 3					
xQ0	1	1	0	0	-
xQ1	1	1	1	1	-
xQ2	1	0	1	0	-
xQ3	0	1	0	1	-
Control mode 4					
xQ0	1	0	0	0	-
xQ1	1	0	1	0	-
xQ2	0	1	0	0	-
xQ3	0	1	0	1	-

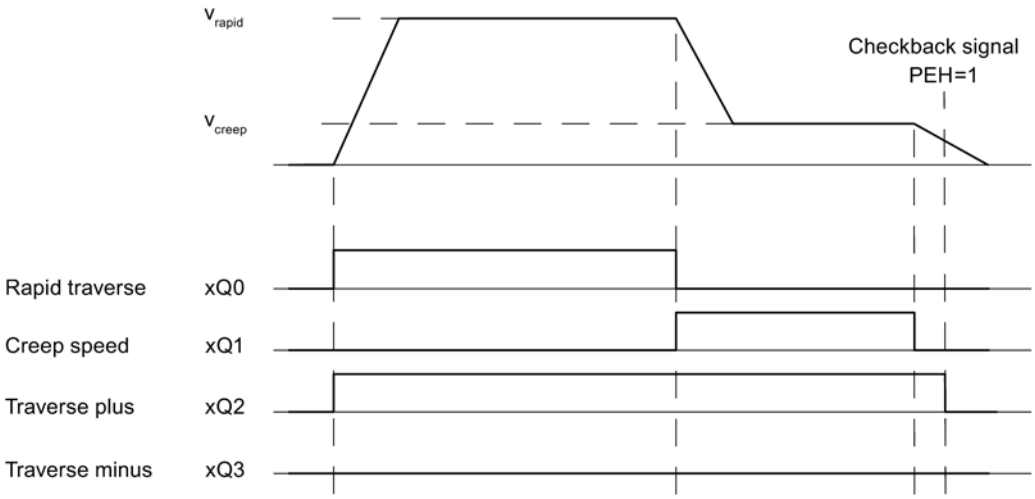
Control mode 1

Control mode 1

The following settings apply for control mode 1 (x stands for channel n):

- xQ0: Rapid traverse
- xQ1 Creep speed
- xQ2 Travel plus
- xQ3 Travel minus

The following image shows the function of the digital outputs with control mode 1.



Data used in the parameter DB

Address	Name	Type	Initial value	Comment
92.0	CTRL_TYPE	DINT	L#1	1 = Control mode 1

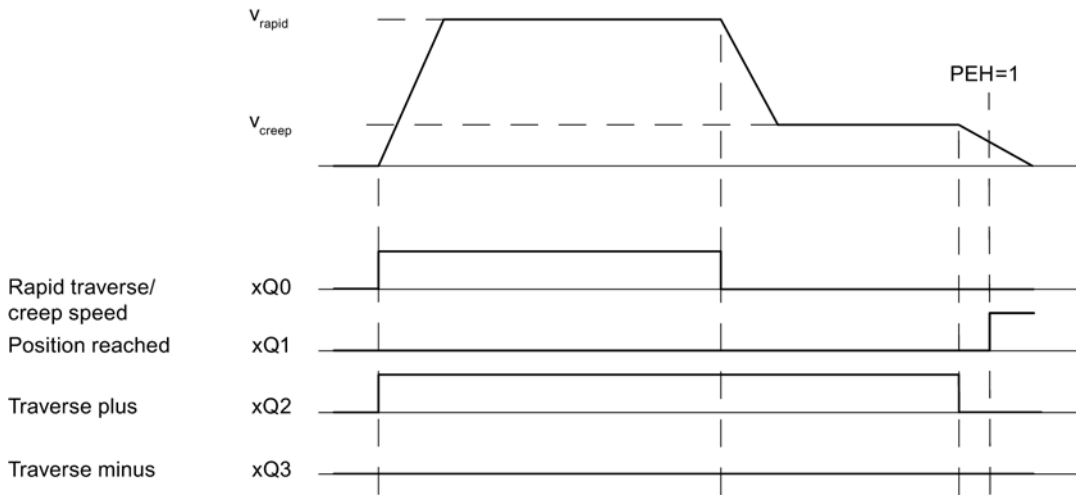
Control mode 2

Control mode 2

The following settings apply for control mode 2 (x stands for channel n):

- xQ0: Rapid traverse/creep speed
- xQ1 Position reached
- xQ2 Travel plus
- xQ3 Travel minus

The following image shows the function of the digital outputs with control mode 2.



Data used in the parameter DB

Address	Name	Type	Initial value	Comment
92.0	CTRL_TYPE	DINT	L#1	2 = Control mode 2

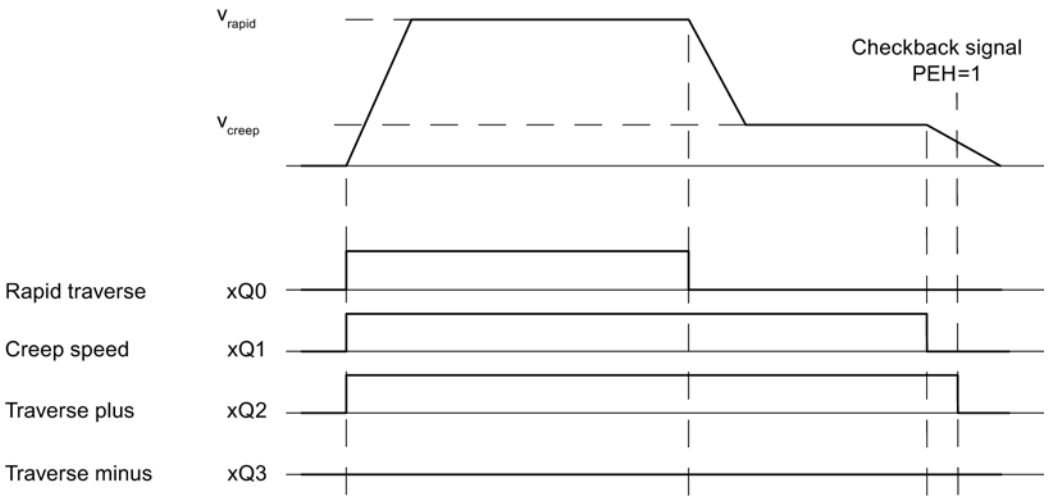
Control mode 3

Control mode 3

The following settings apply for control mode 3 (x stands for channel n):

- xQ0: Rapid traverse
- xQ1 Creep speed
- xQ2 Travel plus
- xQ3 Travel minus

The following image shows the function of the digital outputs with control mode 3.



Data used in the parameter DB

Address	Name	Type	Initial value	Comment
92.0	CTRL_TYPE	DINT	L#1	3 = Control mode 3

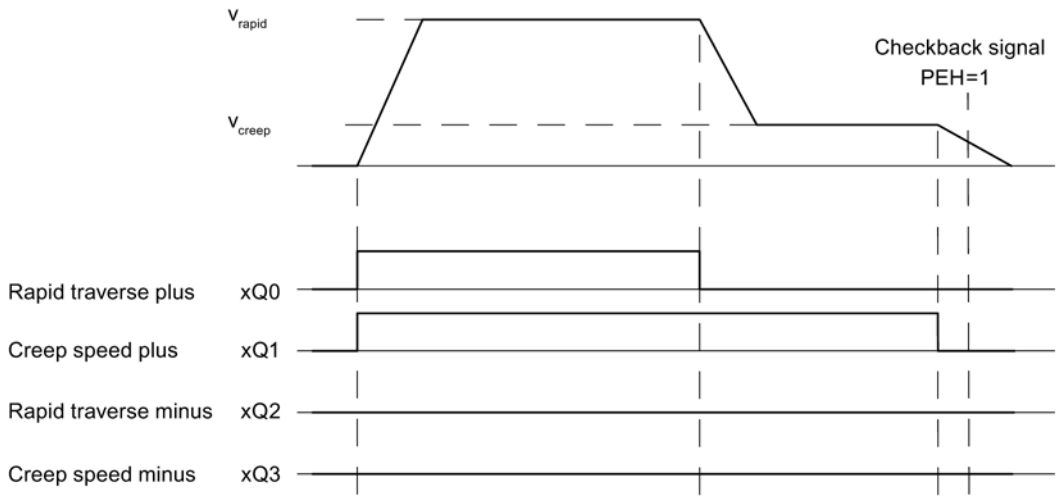
Control mode 4

Control mode 4

The following settings apply for control mode 4 (x stands for channel n):

- xQ0: Rapid traverse plus
- xQ1 Creep speed plus
- xQ2 Rapid traverse minus
- xQ3 Creep speed minus

The following image shows the function of the digital outputs with control mode 4.



Data used in the parameter DB

Address	Name	Type	Initial value	Comment
92.0	CTRL_TYPE	DINT	L#1	4 = Control mode 4

Specifying changeover difference

Changeover difference

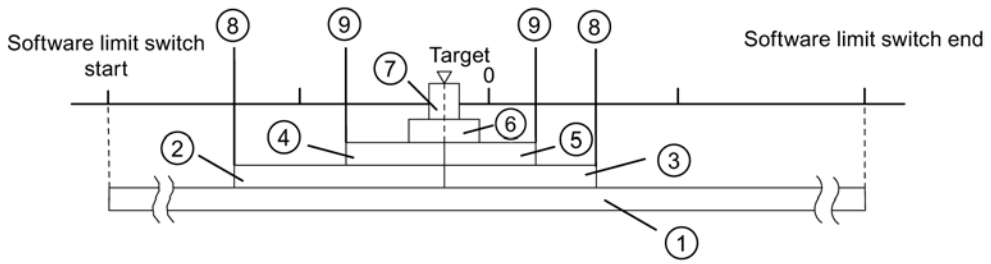
The changeover difference defines the distance from the destination at which the drive is switched over from rapid feed to creep feed.

The specified values apply to all targets approached by the positioning module.. An incremental approach with the incremental dimension 255 is an exception. A special changeover difference is specified for this.

Rules for changeover difference

- The changeover difference plus and the changeover difference minus do not have to be arranged symmetrically around the target. The values can differ.
- The changeover difference must be greater than the switch-off difference.
- The changeover difference must lie within the operating range.
- The changeover difference must be less than the end of rotary axis.
- Select a sufficient distance between the changeover point and the switch-off point to ensure that the drive can safely switch from rapid traverse to creep speed.
- The changeover point and the switch-off point must be separated from each other by a time interval of at least 8 ms.

The following image shows the location of the switch areas around the target.



① Operating range
② Changeover difference plus / minus
③
④ Switch-off difference plus / minus
⑤
⑥ Standstill range
⑦ Target range
⑧ Changeover point
⑨ Switch-off point

For more information regarding the arrangement of the ranges, please see "Ranges and switching points of the positioning modules".

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
100.0	CHGDIF_P	DINT	L#5000	**Changeover difference plus**
104.0	CHGDIF_M	DINT	L#5000	**Changeover difference minus** Valid value range: • 1 µm to 1 000 000 000 µm at resolution ≥ 1 µm/pulse • 1 µm to 100 000 000 µm at resolution <1 µm/pulse

Specifying switch-off difference

Switch-off difference

The "switch-off difference" defines the distance to the destination at which the drive (creep speed) is switched off. The positioning module then adopts the monitoring functions from this point on.

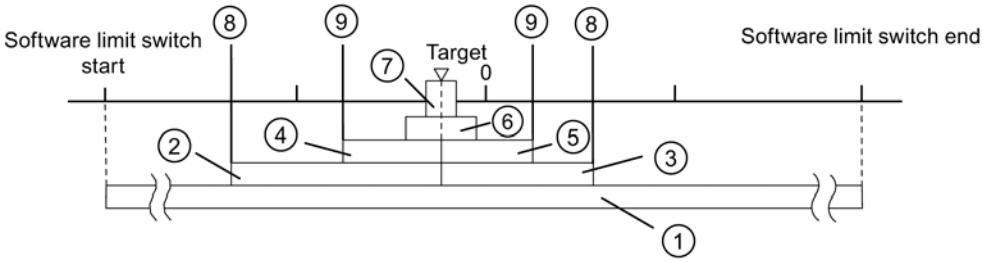
The specified values apply to all targets approached by the positioning module.. An incremental approach with the incremental dimension 255 is an exception. A special switch-off difference is specified for this.

Rules for switch-off difference

- The switch-off difference plus and the switch-off difference minus do not have to be arranged symmetrically around the target. The values can differ.
- The switch-off difference must be greater than half the target range.

- Select the distance between the switch-off point and the target to ensure that the drive comes to a standstill within the target range.
- The switch-off point and the start of the target range must be separated from each other by a time interval of at least 8 ms.

The following image shows the location of the switch areas around the target.



①	Operating range	⑦	Target range
② ③	Changeover difference plus / minus	⑧	Changeover point
④ ⑤	Switch-off difference plus / minus	⑨	Switch-off point
⑥	Standstill range		

For more information regarding the arrangement of the ranges, please see "Ranges and switching points of the positioning modules".

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
108.0	CUTDIF_P	DINT	L#2000	**Switch-off difference plus**
112.0	CUTDIF_M	DINT	L#2000	**Switch-off difference minus** Valid value range: • 1 µm to 1 000 000 000 µm at resolution ≥ 1 µm/pulse • 1 µm to 100 000 000 µm at resolution <1 µm/pulse

Specifying target approach

Specifying target range

Target range

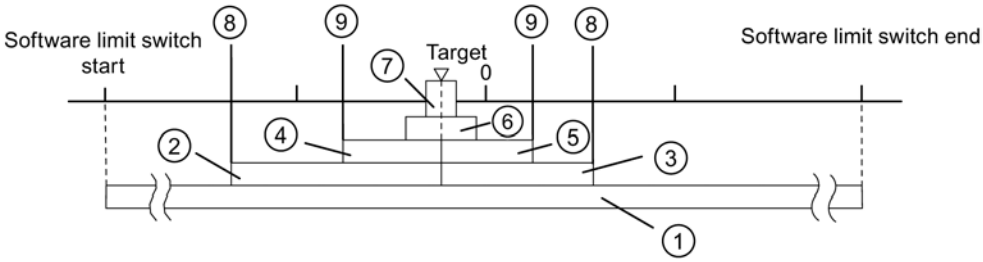
The positioning module sets a symmetrical range around each target, the target range. The actual value must lie within this range with a target approach.

A value specification of "0" switches off the tolerance during the target approach. The speed when reaching the target must then be less than the assigned standstill velocity (Page 10508) parameter.

Rules for the target range

- The target range lies symmetrically around the target.
- A value specification of "0" switches off the monitoring of the target range.

The following image shows the location of the ranges around the target.



①	Operating range	⑦	Target range
②	Changeover difference plus / minus	⑧	Changeover point
③			
④	Switch-off difference plus / minus	⑨	Switch-off point
⑤			
⑥	Standstill range		

Additional information on target approach can be found under "Ranges and switching points of the positioning modules" as well as under "End of a positioning operation (Page 10467)".

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
76.0	TRG_RANGE	DINT	L#1000	**Target range** Valid value range: • 0 = no monitoring • 1 µm to 1 000 000 000 µm at resolution ≥ 1 µm/pulse • 1 µm to 100 000 000 µm at resolution <1 µm/pulse

Specifying monitoring time

Monitoring time

The module monitors the following during the monitoring time:

- the movement of the axis up to the switch-off point.
  The monitoring time starts with the start of a positioning and is retriggered with each actual value change in the traversing direction.
  Within the monitoring time, the actual value must have changed by at least one increment (the distance of the resolution).
- the target approach.
  The monitoring time is retriggered when the switch-off difference is reached for the last time.
  The target range must be reached within the monitoring time.

Additional information on target approach can be found under "End of a positioning operation (Page 10467)".

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
80.0	MON_TIME	DINT	L#2000	**Monitoring time** Valid value range: • 0 = no monitoring • 1 to 100 000 ms
Actual monitoring time You can specify all values from the range of values for the monitoring time. The module rounds the specified time to a multiple of 8 ms (module cycle).				

Specifying standstill range

Standstill range

The standstill range lies symmetrically around the respective target. In this range, monitoring is performed to determine whether the drive remains on an approached target position or drifts away from it. The monitoring begins when the positioning module has reported "Position reached".

If the standstill range is exited without a valid traversing job, the positioning module signals an error.

Rules for standstill range

- The standstill range lies symmetrically around the respective target.
  (Target – ½ × standstill range) ≤ target ≤ (target + ½ × standstill range)
- The standstill range must be greater than or equal to the target range.

- The coordinate values for the standstill range must lie in the operating range.
- A setting of "0" deactivates the standstill monitoring function.

Additional information on target approach can be found under "Ranges and switching points of the positioning modules" as well as under "End of a positioning operation (Page 10467)".

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
84.0	ZSPEED_R	DINT	L#1000	**Standstill range** Valid value range: • 0 = no monitoring • 1 µm to 1 000 000 000 µm at resolution ≥ 1 µm/pulse • 1 µm to 100 000 000 µm at resolution <1 µm/pulse

Specifying standstill velocity

Standstill velocity

The standstill velocity serves as a reference speed. If, during a positioning operation, the actual speed reaches the standstill velocity and the drive is located in the target range, then "Position reached" (PR) is set. The positioning operation is therefore successfully completed.

The precision of the positioning increases when the value of the standstill velocity is reduced. A setting of "0" deactivates the standstill monitoring function.

Additional information on target approach can be found under "End of a positioning operation (Page 10467)".

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
88.0	ZSPEED_L	DINT	L#30000	**Standstill velocity** Valid value range: • 0 = no monitoring • 1 µm/min to 100 000 µm/min

Channel n axis

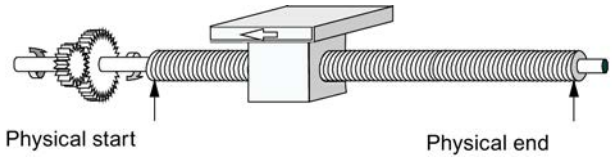
Specifying axis type

Selecting axis type

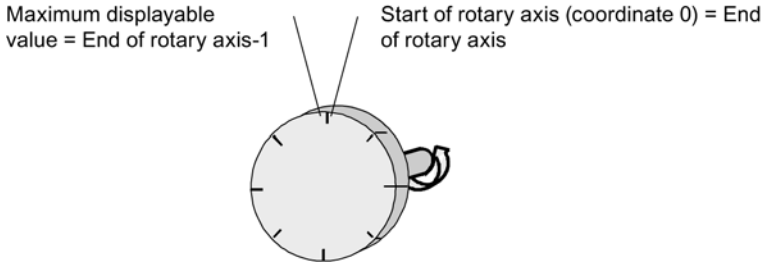
Axis types

The positioning module can control the following axis types:

- Linear axis
  A linear axis is an axis which has a limited physical traversing range.



- Rotary axis
  The rotary axis is an axis whose traversing range is not restricted by mechanical stops.



Data used in the parameter DB

Address	Name	Type	Initial value	Comment
12.0	AXIS_TYPE	DINT	L#0	**Axis type** Valid values: • Linear axis = 0 • Rotary axis = 1

Specifying software limit switch

Software limit switch

Software limit switches only exist for the linear axis.

The operating range is the range limited by the software limit switches.

The software limit switch start must always be smaller than the software limit switch end.

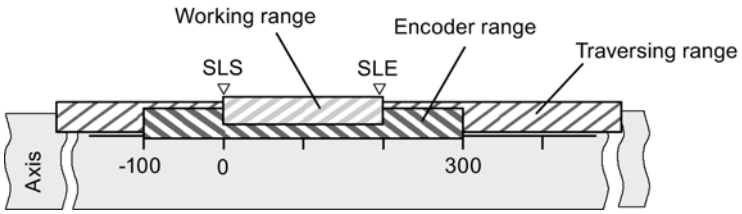
The software limit switches are monitored after the axis is synchronized.

- When using an incremental encoder:
  The synchronization of the axis is achieve by a reference point approach or by setting a reference point.
- When using an absolute encoder (SSI):
  The axis is synchronized when the module has received a complete, error-free frame for the associated channel.
  The absolute encoder must cover at least the operating range including the software limit switches.

Relationship between the operating range, encoder range and traversing range

- The operating range is the range which you specify for your job using the software limit switches.
- The encoder range represents the range explicitly covered by the encoder. For a linear axis, the module assigns the encoder range symmetrically across across the operating range. In other words, the module shifts the encoder range in order to equalize the distances between the software limit switches and the ends of the encoder range (see figure below).
- This traversing range represents the value range the positioning module can process. The traversing range is dependent on the resolution.

The following figure shows the relationship between software limit switches, the operating range, the encoder range and the traversing range.



Data used in the parameter DB

Address	Name	Type	Initial value	Comment
64.0	SSW_STRT	DINT	L#-100000000	**Software limit switch start**
68.0	SSW_END	DINT	L#100000000	**Software limit switch end** Valid value range: • -1,000,000,000 µm to 1 000 000 000 µm at resolution ≥ 1 µm/pulse • -100,000,000 µm to 100,000,000 µm at resolution <1 µm/pulse

Specifying end of rotary axis

End of rotary axis

The value "end of rotary axis" is the theoretical maximum value that the actual value can reach. The theoretically highest value is however never displayed, because it physically identifies the same position as the start of the rotary axis (0).

The maximum value that will be displayed for a rotary axis has the value:

End of rotary axis [µm] - resolution [µm / pulse] x 1 [pulse]

Example: End of rotary axis 1000 mm, resolution 1000 µm / pulse

The display jumps:

- with a positive rotational direction from 999 mm to 0 mm
- with a negative rotational direction from 0 mm to 999 mm.

Rotary axis with absolute encoders

The rotary axis range (0 to end of rotary axis) of a rotary axis with an absolute encoder must correspond exactly to the encoder range of the absolute encoder:
End of the rotary axis [µm] = number of encoder revolutions x distance/encoder revolutions [µm]

Data used in the DB parameter

Address	Name	Type	Initial value	Comment
16.0	ENDROTAX	DINT	L#100000	**End of rotary axis** Valid value range: • 1 µm to 1 000 000 000 µm at resolution ≥ 1 µm/pulse • 1 µm to 100 000 000 µm at resolution <1 µm/pulse

Specifying reference point approach

Selecting type of reference point approach

Type of reference point approach

With "Type of reference point approach", you determine the conditions for the axis synchronization.

- The first specification defines the start direction in which the reference point approach is starting.
- The second specification defines the location of the zero mark, which leads to synchronization, as far as the reference point switch is concerned.

Detailed information can be found under "Reference point approach" operating mode (Page 10476).

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
52.0	REFPT_TYPE	DINT	L#0	**Type of reference point approach** Valid values: • 0 = plus, reference point switch in plus direction • 1 = plus, reference point switch in minus direction • 2 = minus, reference point switch in plus direction • 3 = minus, reference point switch in minus direction

Specifying reference point coordinate

Reference point coordinate

- **Incremental encoders:**
  You require the reference point coordinate for the "Reference point approach" mode. If the axis is not synchronized following writing and activating of machine data, the actual value is set to the value of the reference point coordinate.
- **Absolute encoder (SSI)**
  You require the reference point coordinates for the mechanical adjustment of the encoder. For more information, refer to the description of absolute encoder adjustment (Page 10513) which explains the interaction of the absolute encoder adjustment with other data.

The value of the reference point coordinate must lie within the operating range:

- **Linear axis**
  Including the software limit switches.
- **Rotary axis**
  Greater than or equal to 0 and less than the "end of rotary axis" value (0 ≤ reference point coordinate < end of rotary axis).

Data used in the DB parameter

Address	Name	Type	Initial value	Comment
44.0	REFPT	DINT	L#0	**Reference point coordinate** Valid value range: • -1 000 000 000 µm to 1 000 000 000 µm at resolution ≥1 µm/pulse • -100 000 000 µm to 100 000 000 µm at resolution < 1 µm/pulse

Specifying absolute encoder adjustment

Determining the absolute encoder adjustment

Absolute encoder adjustment

Absolute encoder adjustment is only necessary for an absolute encoder.

The absolute encoder adjustment and reference point coordinate map the encoder's range of values explicitly to the axis coordinate system.

Additional information can be found under "Absolute encoder adjustment procedure (Page 10513)" and in an example (Page 10514).

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
44.0	REFPT	DINT	L#0	**Reference point coordinate** Valid value range: • -1 000 000 000 µm to 1 000 000 000 µm at resolution ≥ 1 µm/pulse • -100,000,000 µm to 100 000 000 µm at resolution <1 µm/pulse
48.0	ENC_ADJ	DINT	L#0	**Absolute encoder adjustment** Valid value range: • 0 to (2^{25} - 1)

Absolute encoder adjustment procedure

Determining the correct absolute encoder adjustment

After initial parameter assignment, further steps must be taken to ensure that a correct relationship is created between the encoder and coordinate system. The procedure shown in the next section is based on the use of the parameter assignment dialogs.

1. Traverse the axis to a defined, reproducible point to which a unique coordinate is assigned. This can, for example, be the "software limit switch end".
2. Call the job "Set reference point (Page 10490)" with the coordinate of the point defined under 1.
   The positioning module now determines an encoder value for the reference point coordinate entered in the DB channel (REFPT in channel DB), namely the absolute encoder adjustment. You can read this value with the "Read encoder data (Page 10495)" job.
3. Enter the value read in the "absolute encoder adjustment" field.
4. Save your parameter assignment to the corresponding parameter DB using the export function.

5. Download the data in the HWCN to the CPU.
6. Restart the CPU (cold restart) to apply the data.

Note

You perform this adjustment just once during the commissioning. After a configuration, the position module is synchronized during start-up as soon as a complete, faultless message frame of the encoder is received after start-up.

Otherwise: Mechanical adjustment of the encoder

A correct relationship between the coordinate system and the encoder can be achieved as follows:

1. Move the axis to a defined, reproducible point to which a unique coordinate is assigned. This can, for example, be the "software limit switch start".
2. Enter this coordinate value in the machine data as "reference point coordinate".
3. Read the encoder value displayed at this position with the "Read encoder data (Page 10495)" job.
4. Enter this value in the machine data as the "absolute encoder adjustment".

After the parameter assignment a correct actual value will always be shown.

Instead of steps 3 and 4, you can also set the encoder to zero using "Reset", if available. Then enter the value "0" in the machine data as the absolute encoder adjustment.

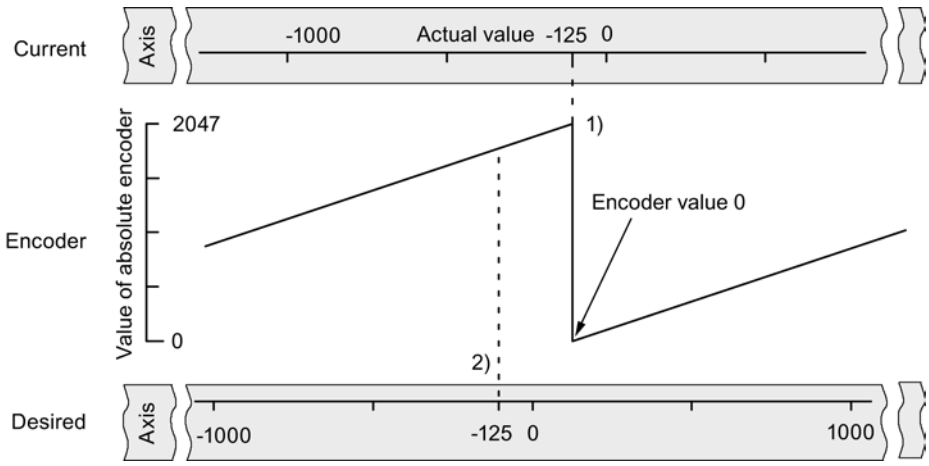
Example of absolute encoder adjustment

Example of an absolute encoder adjustment

The following assumptions apply to the example:

- Reference point coordinate = -125 mm
- Operating range from SSW_STRT = -1000 mm to SSW_END = 1000 mm
- Absolute encoder adjustment = 0
- Encoder range = 2048 increments with a resolution of 1 mm/pulse
- The absolute encoder used cannot be adjusted mechanically with precision and it also is not possible to set the encoder value specifically.

The following figure shows an example of absolute encoder adjustment.



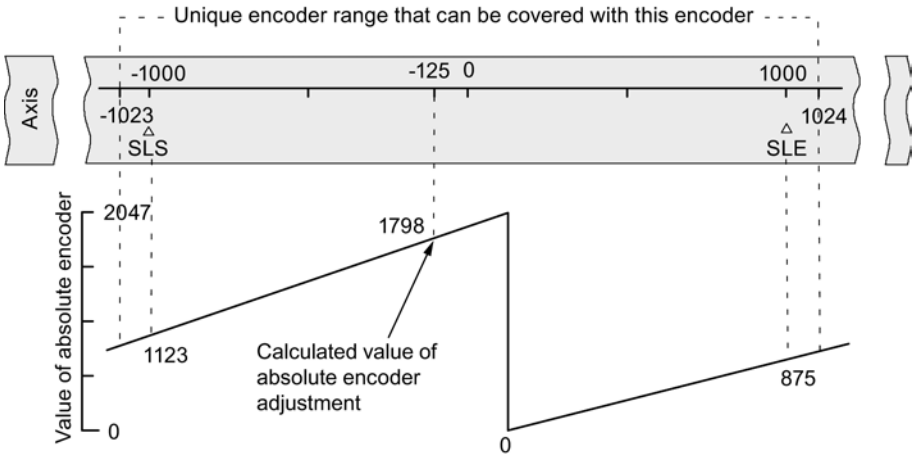
1) Assignment of the coordinate system to encoder values, based on the absolute encoder adjustment. The encoder value 0 corresponds to the actual value -125.

2) Required assignment of the coordinate system to the encoder. At this position the coordinate should be -125.

Result in accordance with "Set reference point"

After "Set reference point" the relationship looks as follows:

The reference point coordinate on the axis (-125) is assigned to the encoder value (1798) determined from absolute encoder adjustment. This assignment is set out in the figure below.



The encoder supplies 2048 unique values. The operating range is defined by the software limit switches. The selected resolution of 1 mm per pulse means the encoder can, however, cover a greater operating range than the one set by the software limit switches.

With the set resolution, the operating range is already covered with 2001 values. Therefore, in the example, 47 pulses "remain" and these are symmetrically positioned around the operating range.

Selecting start velocity

Start speed in the case of a reference point approach

With this data you select the speed for the start of the reference point approach.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
99.0	REFPT_SPD	BOOL	TRUE	**Start speed for reference point approach** Valid values: • 0 = rapid traverse • 1 = creep speed

Channel n encoder

Selecting the signal type

Signal type

The encoder provides distance information to the module. The module then evaluates this information and converts this with the resolution into an actual value.

You can select the signal type from one of the following encoders:

- 5 V incremental encoder
  Select the signal type "5 V differential signal" to use a 5 V incremental encoder with differential signal A, /A, B, /B and N, /N.
- 24 V incremental encoder
  Select signal type "24 V track A+B phase shifted" to use a 24 V incremental encoder with signals A*, B* and N*.
- Absolute encoder (SSI)
  Select the "SSI absolute" signal type if you use an absolute encoder with a serial interface.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
20.0	ENC_TYPE	DINT	L#1	**Signal type** Valid values: • 1 = 5 V differential signal • 2 = 24 V track A+B phase shifted • 3 = SSI 13-bit frame length [1)] • 4 = SSI 25-bit frame length [1)] • 5 = SSI 14-bit frame length [1)] • 6 = SSI 15-bit frame length [1)] • 7 = SSI 16-bit frame length [1)] • : • : • 14 = SSI 23-bit frame length [1)] • 15 = SSI 24-bit frame length [1)]

[1)] The frame length is selected in the "SSI" dialog box.

Selecting the count direction

Count direction

With the machine data "Count direction" you can adapt the direction of the position measuring to the direction of movement of the axis. You must also take into account the rotational direction of all transmission elements (for example, couplings and gears).

- Normal = incremental count pulses (incremental encoder) or encoder values (absolute encoder) represent increasing actual position values
- Inverted = incremental count pulses (incremental encoder) or encoder values (absolute encoder) correspond to decreasing actual position values

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
59.0	CNT_DIR	BOOL	FALSE	**Count direction** Valid values: • 0 = normal • 1 = inverted

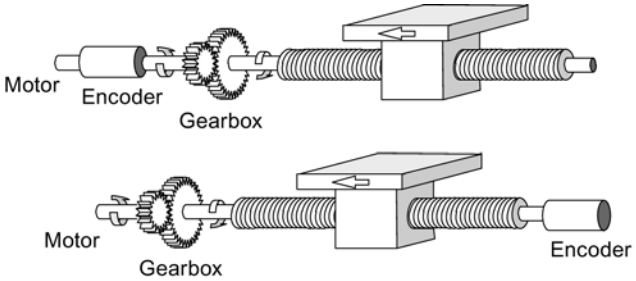
Specifying resolution

Specify distance per encoder revolution

Distance per encoder revolution

With the machine data "Distance per encoder revolution" you can inform the positioning module which distance the drive system should cover per encoder revolution.

The "Distance per encoder revolution" value depends on the structure of the axis and the way in which the encoder is mounted. You must take into account all the transfer elements like clutches or gearboxes. The following figure shows an example of two differently-mounted encoders.



The relationship between the "Distance per encoder revolution" and "Increments per encoder revolution" machine data is described under resolution (Page 10519).

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
24.0	DISP_REV	DINT	L#80000	**Distance per encoder revolution** Valid value range: • 1 µm to 1 000 000 000 µm

Specifying increments per encoder revolution

Increments per encoder revolution

The machine data "Increments per encoder revolution" indicates the number of increments that an encoder makes for each revolution. The positioning module calculates the resolution (Page 10519) based on this value and the "Distance per encoder revolution" machine date.

Incremental encoder

Any value within the range of values can be entered. The module evaluates the increments four times.

Absolute encoder

For the limits you have to differentiate between the individual encoder models. Only values to the power of 2 are permitted as the entered value.

- Single-turn encoder (number of revolutions = 1) with 13-bit frame length:
  - minimum value = 4
  - maximum value = 8192
- Multiturn encoder (number of revolutions > 1) with 25-bit message frame length:
  - Minimum value = 4
  - Maximum value = 8192
- Single-turn encoder (number of revolutions = 1) with 25-bit frame length:
  - Minimum value = 4
  - maximum value = 2^{25}

Linear scales will be configured as multiturn encoders as follows:

- Increments per encoder revolution = 8192
- Number of revolutions x 8192 ≥ number of steps in the linear scale

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
32.0	INC_REV	DINT	L#500	**Increments per encoder revolution** Valid value range: • 1 to 2^{25}

Resolution

Resolution

The resolution indicates the traversing distance corresponding to one pulse. It is a measurement for the precision of the positioning and also determines the possible maximum traversing range of the positioning module.

The resolution (RES) can be calculated as in the following table:

	Incremental encoder	Absolute encoder
Input values	• Distance per encoder revolution • Increments per encoder revolution: – Quadruple evaluation – 1 increment = 4 pulses	• Distance per encoder revolution • Increments per encoder revolution: – 1 increment = 1 pulse
Calculation	$RES = \frac{\frac{Distance}{Encoder\ revolution}}{\frac{Pulses}{Encoder\ revolution}}$	

Note

All position information is rounded up to the integral multiple of the resolution. This way the entered values differentiate from the used values.

Range of values of the resolution

The chosen system of units determines the range of values of the resolution:

System of units	Information in ...	Range of values of the resolution
mm	10^{-3} mm	$0{,}1 \times 10^{-3}$ … 1000×10^{-3} mm/pulse
inch	10^{-4} inch	$0{,}1 \times 10^{-4}$ … 1000×10^{-4} inch/pulse
degrees	10^{-4} degrees 10^{-3} degrees 10^{-2} degrees	$0{,}1 \times 10^{-4}$ … 1000×10^{-4} degrees/pulse $0{,}1 \times 10^{-3}$ … 1000×10^{-3} degrees/pulse $0{,}1 \times 10^{-2}$ … 1000×10^{-2} degrees/pulse

Example

- An incremental encoder has the following data:
  - Increments per encoder revolution: 5000
  - Distance per encoder revolution: 1000 mm
  - 1 increment = 4 pulses

  This results in the resolution (4 x evaluation):

$$\text{Resolution} = \frac{1000\ \text{mm}}{5000\ \text{increments}} = 0.2000\ \frac{\text{mm}}{\text{Increment}} = 0.2000\ \frac{\text{mm}}{4\ \text{pulses}}$$

$$= 0.0500\ \frac{\text{mm}}{\text{Pulse}}$$

- An SSI encoder has the following data:
  - Increments per encoder revolution: 4096
  - Distance per encoder revolution: 1000 mm
  - 1 increment = 1 pulse

  This results in the resolution:

$$\text{Resolution} = \frac{1000\ \text{mm}}{4096\ \text{increments}} = 0.2441\ \frac{\text{mm}}{\text{Increment}} = 0.2441\ \frac{\text{mm}}{\text{Pulse}}$$

Relationship between the traversing range and resolution

The traversing range is restricted by the numerical representation in the position module. This numerical representation varies depending on the resolution. You should thus make sure that your specifications are always within the permitted limits.

The maximum traversing range is illustrated in the following table:

Resolution (RES) lies in the range	Maximum traversing range
0.1 µm/pulse ≤ RES < 1 µm/pulse	-10^8 µm to 10^8 µm (-100 m to +100 m)
1 µm/pulse ≤ RES ≤ 1000 µm/pulse	-10^9 µm to 10^9 µm (-1000 m to +1000 m)

Selecting data for absolute encoder (SSI)

Number of encoder revolutions

Number of encoder revolutions

The machine date "Number of encoder revolutions" only applies to absolute encoders. As a result, enter the number of revolutions that are possible with this encoder.

The **overall number of steps of the encoder** is not machine data. It is calculated as follows:

Overall number of steps = increments/encoder revolution x number of revolutions

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
36.0	NO_REV	DINT	L#1	**Number of encoder revolutions** Valid value range: • 1 (single-turn encoder) • 2 to 4096 in powers of 2 (multiturn encoder)

Transmission rate

Transmission rate

With the machine data "Transmission rate" you can determine the data transmission rate from the SSI encoder to the positioning module.

The maximum Transmission rate is dependent on the cable length:

- 200 m → 188 kHz
- 100 m → 375 kHz
- 40 m → 750 kHz
- 12 m → 1500 kHz

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
40.0	BAUDRATE	DINT	L#0	**Transmission rate** Valid values: • 0 = 188 kHz • 1 = 375 kHz • 2 = 750 kHz • 3 = 1500 kHz

Frame length

Frame length

With "frame" you define the clock frame output by the positioning module for absolute encoder.

A frame between 13 bits and 25 bits is permitted.

Data used in the parameter DB

The frame length selected here is stored in the parameter DB under the ENC_TYPE (Page 10516) date in the parameter DB.

Parity

Parity

With "Parity", you specify whether no parity, even parity or odd parity should be assigned for an absolute encoder.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
28.0	PARITY	DINT	L#0	**Parity** Valid values: • 0 = none • 1 = even parity • 2 = odd parity

Selecting monitoring

Monitoring

Wire break

When monitoring is activated, the positioning module monitors all cables for a 5 V incremental encoder and an absolute encoder. The monitoring detects:

- Wire break
- Short circuit on individual cables
- Edge distance of counting pulses (also for 24 V incremental encoder)

For monitoring for a 24 V incremental encoder, you must set MON_TIME > 0 as the monitoring time (Page 10507).

With 5 V incremental encoders with zero marks, you must disable the wire break monitoring or interconnect the N and /N signals externally.

Frame error

The module monitors the message frame of an absolute encoder (SSI) for:

- Start and stop bit errors

You cannot disable the message frame error monitoring with absolute encoders (SSI).

Missing pulses (incremental encoder)

An incremental encoder must always return the same number of increments between two successive zero marks.

The positioning module checks to see if the zero mark of an incremental encoder is occurring at the correct encoder state.

For encoders without zero marks you must switch off the monitoring for missing pulses.

You must also disable the wire-break monitoring or interconnect the N and /N zero mark inputs externally.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
				Monitoring:
63.0	MON_WIRE	BOOL	TRUE	1 = wire break
63.1	MON_FRAME	BOOL	TRUE	1 = message frame error (must always be 1)
63.2	MON_PULSE	BOOL	TRUE	1 = missing pulses

Channel n incremental dimensions

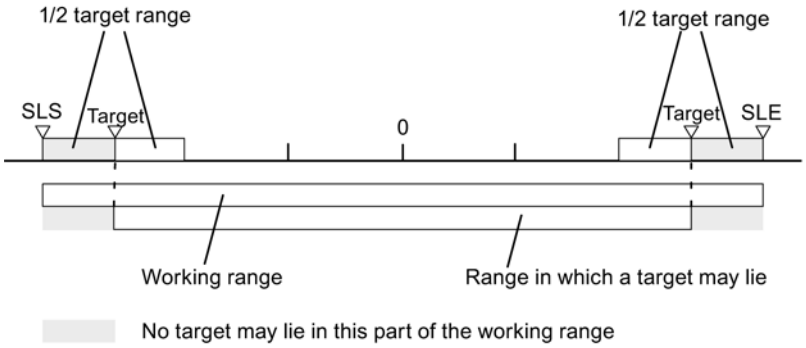
Specifying incremental dimensions

Definition

Incremental dimensions are target specifications that can be controlled by the positioning module with the "relative incremental approach" or rather "absolute incremental approach" mode.

Possible range for incremental dimensions

The target that is to be approached must be at a distance of at least one-half the target range in front of the respective software limit switch. The following figure shows the limits for specifying incremental dimensions.



Increment number 1 to 100

Incremental dimension number 1 to 100

You have the possibility of entering up to 100 incremental dimensions in a table that are valid both for the "Relative incremental approach" mode as well as for the "Absolute incremental approach" mode.

Note that the positioning module does not permit negative values for the "relative incremental approach". The values will be interpreted by the positioning module - depending on the direction of movement - as either a positive or a negative difference.

Note

The entry is made in the unit of the system of units set. Note the decimal places in particular here.

Numerical example:

- Incremental dimension: 800 mm
- System of units: 10^{-3} mm
- Entry in the DB parameter: 800000

Tip: Define separate areas for relative and absolute incremental dimensions in the incremental dimension table.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
35.4	TRGL1WR_EN	BOOL	FALSE	1 = write increment table 1 (increments 1 ... 50)
35.5	TRGL2WR_EN	BOOL	FALSE	1 = write increment table 2 (increments 51 ... 100)
36.6	TRGL1RD_EN	BOOL	FALSE	1 = read increment table 1 (increments 1 ... 50)
36.7	TRGL2RD_EN	BOOL	FALSE	1 = read increment table 2 (increments 51 ... 100)

Data used in the parameter DB

Address	Name	Type	Initial value	Comment	
120.0	TRGL1.TRG[1]	DINT	L#0	Incremental dimension number 1	Increment table 1
. . .	. . .	. .	. . .	. . .	
316.0	TRGL1.TRG[50]	DINT	L#0	Incremental dimension number 50	
320.0	TRGL2.TRG[51]	DINT	L#0	Incremental dimension number 51	Increment table 2
. . .	. . .	. . .	. . .	. . .	
516.0	TRGL2.TRG[100]	DINT	L#0	Incremental dimension number 100	

Increment number 254

Incremental dimension number 254

Independently of the increment table you can use the increment number 254 as an additional distance specification. For the changeover and switch-off differences, the entries from the parameter DB are valid for this increment.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
36.2	TRG252_254_EN	BOOL	FALSE	1 = write increment for increment number 254
96.0	TRG252_254	DINT	L#0	Increment for increment number 254

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
100.0	CHGDIF_P	DINT	L#5000	Changeover difference plus
104.0	CHGDIF_M	DINT	L#5000	Changeover difference minus
108.0	CUTDIF_P	DINT	L#2000	Switch-off difference plus
112.0	CUTDIF_M	DINT	L#2000	Switch-off difference minus

Increment number 255

Incremental dimension number 255

The incremental dimension number 255 gives you an additional guideline for the path.

The switch-off differences and the changeover differences are presented together with the increment. Unlike the situation with the other increments, the increment 255 uses the values established in the channel DB for the switch-off and changeover differences. The entries from the machine data have no validity for this increment.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
36.3	TRG255_EN	BOOL	FALSE	1 = write increment for increment number 255
100.0	TRG255	DINT	L#0	Increment for increment number 255
104.0	CHGDIF255	DINT	L#0	Changeover differences for increment number 255
108.0	CUTDIF255	DINT	L#0	Switch-off differences for increment number 255

Control and feedback interface of the FM 351

Application

In special applications or in an alarm level, a particularly fast access to checkback and control signals could be required. You can reach this data directly via the input and output areas of the module.

To coordinate startup following each module startup, for example, after plugging a module or after a STOP-RUN transition of the CPU, you must call the ABS_CTRL (Page 7125) instruction until the end of the startup is indicated by RETVAL = 0. Then, do not use the ABS_CTRL anymore.

Note

It is not possible to use the ABS_CTRL together with a write access.

Reading checkback signals by means of direct access

The byte addresses must be specified relative to the start address of the outputs of the respective channel. The names of the parameters correspond to the names in the channel DB (Page 7133).

Start address channel 1 = start address of the module

Start address channel 2 = start address of the module + 8

In STL, you access the data using the commands PEB (read 1 byte), PEW (read 2 bytes) and PED (read 4 bytes).

The following table shows the assignment of the feedback interface (inputs) of FM 351.

Address	Bit number							
	7	6	5	4	3	2	1	0
Byte 0	PARA	Internal	Internal	DATA_ERR	OT_ERR	DIAG	Internal	Internal
Byte 1	CHGOVER	CUTOFF	ZSPEED	SPEED_OUT	0	WAIT_EI	WORKING	ST_ENBLD
Byte 2	MODE_OUT							
Byte 3	POS_RCD	0	0	0	GO_P	GO_M	0	SYNC
Byte 4	ACT_POS							
Byte 5								
Byte 6								
Byte 7								

Example: Position process value ACT_POS

The start address of the module is 512.

STL	Description
L PED 516	//Read the current position process value (ACT_POS) from channel 1
	//with direct access:
	//Start address of the channel + 4

Writing control signals using direct access

The byte addresses must be specified relative to the start address of the inputs of the respective channel. The names of the parameters correspond to the names in the channel DB (Page 7133).

Start address channel 1 = start address of the module

Start address channel 2 = start address of the module + 8

In STL, you access the data using the commands PAB (write 1 byte), PAW (write 2 bytes) and PAD (write 4 bytes).

The following table shows the assignment of the control interface (outputs) of FM 351.

Address	Bit number							
	7	6	5	4	3	2	1	0
Byte 0	0	0	0	0	OT_ERR_A	0	0	0
Byte 1	DRV_EN	0	0	0	DIR_P	DIR_M	STOP	START
Byte 2	MODE_IN							
Byte 3	MODE_TYPE							
Byte 4	Reserved							
Byte 5								
Byte 6								
Byte 7								

Example: START signals of channel 2

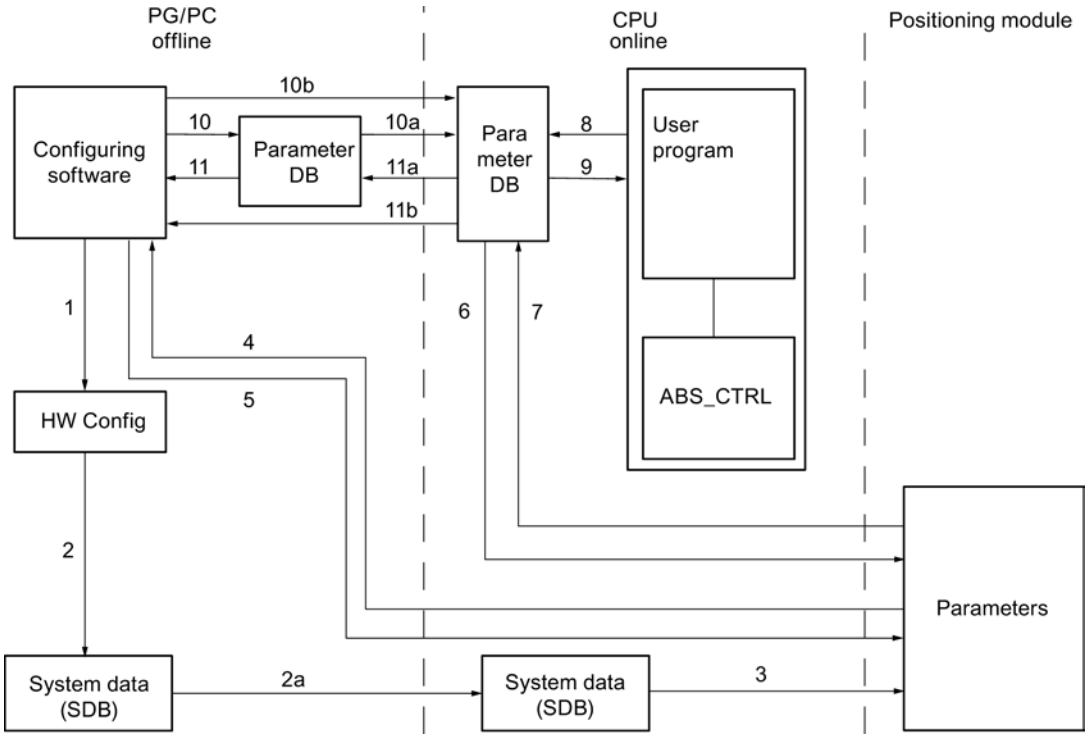
The start address of the module is 512.

STL	Description
L 2#10001000	//Set DRV_EN and DIR_P to 1
T PAB 521	//Write signals with direct access for channel 2:
	//Start address of the module + 8 + 1

Parameter transmission paths

Parameters include the following machine data and increments.

The following figure shows the parameter transmission paths of the FM x51.



1 Saving parameters in the configuration software.

2 Save and compile the HW configuration.

2a Download the HW configuration to the CPU. The CPU automatically carries out step 3.

3 The CPU writes the parameters when system parameter assignments are carried out for the module.

4 With the command "download target system to PG" download the parameters of a channel of the module to the PG.

5 With the command "download target system" download the parameters from the configuration software into a channel of the module.

6 Write parameters using jobs of the user program in a channel of the module.

7 Read parameters using jobs of the user program from a channel of the module.

8 Store parameters from the user program into the online DB.

9 Read in parameters from the online DB into the user program.

10 Export parameters from the configuration software to the offline DB.

10a Download the offline DB to the CPU.

10b Export parameters from the configuration software to the online DB.

11 Import parameters from the offline DB into the configuration software.

11a Load parameters from the online DB to the PG.

11b Import parameters from the online DB into the configuration software.

Some use cases for the transfer of parameters

Use case	Steps
You process the parameters using the configuration software. The channels of the module should then be automatically assigned upon start-up.	Carry out steps 1, 2, and 2a.
You change the parameters during commissioning in test mode in the configuration software.	Carry out steps 4 and 5.
The parameters edited during commissioning should be downloaded automatically during startup.	Carry out steps 1, 2, and 2a.
You create the parameters using the configuration software. The channels of the module should be assigned upon startup only by the user program via the data blocks.	Carry out steps 10, 10a and 6 or 10b and 6.
You would like to create conveniently stored data for recipes.	Carry out steps 10 and 10a.
You create the parameters using the configuration software. These should be made available to the user program for temporary changes.	Carry out steps 1, 2 and 2a for the automatic parameter assignment.
	Carry out the steps 10, 10a, 9, 8 and 6 for access by the user program.
You change existing parameters solely via the user program.	Carry out steps 7, 9, 8 and 6.
You would like to use the configuration software to view the data that has been changed with the user program.	Carry out steps 11a and 11 or just 11b.
Parameters modified by the user program should also be loaded automatically during startup.	Carry out steps 11b or 11a, 11 and then 1, 2, 2a.

Commissioning the FM 351

Using FM 351 from the user program

Integrating the FM 351 into a user program

Instructions are available for integrating the FM 351 into a user program. These enable you to handle the desired functions with ease.

The instructions ABS_INIT and ABS_CTRL can be called in the CPU user program. They ensure communication between the CPU and the FM 351. The channel DB (Page 7133) is created automatically as the instance DB for the ABS_CTRL instruction when you insert the ABS_CTRL instruction into the project.

All the parameter data associated with the channel of the module (machine data, incremental dimension table) is located in the parameter DB (Page 7139) for the module. Use the command "Program blocks > Add new block" to add a ABS_PARATYPE type data block to your project.

The instruction ABS_DIAG is also available for FM 351. This operation allows you to transfer the diagnostic information to the diagnostic DB. The diagnostic DB (Page 7141) is created automatically as the instance DB for the ABS_DIAG instruction when you insert the ABS_DIAG instruction into the project.

Commissioning the module

The module can be initially tested using the "Commissioning" dialog, even when no user program is available. Note the following warning in particular in this case.

WARNING

Injury and material damage could occur.

To prevent personal injury and material damage, observe the following points:

- Install an emergency stop switch in the vicinity of the computer.
  This is the only way you can ensure that the plant can be safely shut down in the event of computer or software failure.
- Install hardware limit switches, which act directly on the power units of all drives.
- Make sure that no one has access to the area of the plant in which moving parts are present.
- When the FM 351 is controlled and monitored simultaneously by your program and the "Commissioning" dialog, conflicts can occur, the effects of which are not clear. Therefore, always set the CPU to STOP mode or disable your program when you work with the "Commissioning" dialog.

See also

ABS_INIT (Page 7124)

ABS_CTRL (Page 7125)

ABS_DIAG (Page 7127)

FM 351: Online & Diagnostics

Current module information

Default diagnostics

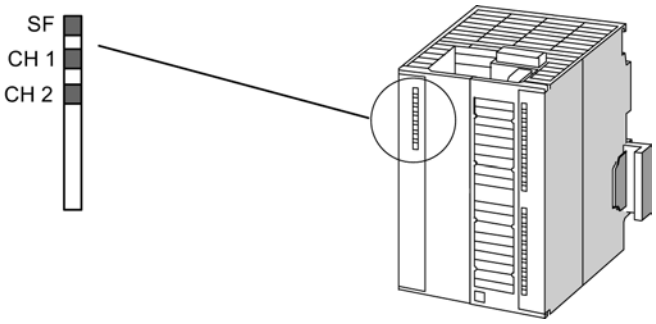
Error display through error LEDs of FM 351

Where is the error displayed?

If the red group error LED is lit, an error occurred either at the module (internal error) or at the cable ports (external error).

The red LEDs CH 1 and CH 2 display a channel error.

The following figure shows the location of the error LEDs on the FM 351.



Which errors are displayed?

The table below shows the error types indicated by the error LEDs.

LED	Meaning	Explanations
SF	Group errors	This LED indicates the following error state of the FM 351: Diagnostic interrupt (internal or external (channel) errors) To remove the error, see Error classes (Page 10538).
CH 1 CH 2	Channel error 1 Channel error 2	These LEDs show a channel error on channel 1 or on channel 2. • Encoder wire break • Absolute encoder error • Missing pulses of Incremental encoder • Operating error • Programming error

Channel-specific diagnostics

Triggering diagnostic interrupts

Processing interrupts

The positioning module can trigger a diagnostic interrupt. These interrupts are processed in an interrupt OB. If an interrupt is triggered without the corresponding OB being loaded, the CPU switches to STOP.

Enable the processing of the diagnostic interrupts as follows:

1. Select the module in HWCN.
2. Select "Edit" > Object Properties > Basic Parameters" to enable the diagnostic interrupt.
3. Save and compile the configuration.
4. Download the configuration to the CPU.

Overview of diagnostic interrupts

The following events and errors trigger a diagnostic interrupt:

- Operating error
- Machine data error
- Diagnostics errors

These errors are explained in detail in "Error classes (Page 10538)".

Reaction of the positioning module to an error with diagnostic interrupt

- The positioning is canceled.
- The synchronization is deleted in the case of the following diagnostic interrupts:
  - Front connector missing, external auxiliary voltage for encoder supply missing
  - Zero mark error detected, cable fault (5 V encoder signals)
  - Traversing range exceeded (indicated by an operating error)
  - Set process value cannot be executed (indicated by an operating error).
- With one exception, control signals START, DIR_P und DIR_M are no longer processed.
  Exception:
  In the case of an operating error it is still possible to jog in the direction of the operating range.
- Function switches and jobs will no longer be processed.

Positioning module detects an error ("incoming")

A diagnostic interrupt is "incoming", if at least one error is given. If not all errors are remedied, the remaining existing errors will be reported again as "incoming".

Sequence:

1. The positioning module detects one or more errors and generates a diagnostic interrupt. The "SF" LED and, depending on error, the "CH1" / "CH2" LEDs illuminate. The error is then entered in the diagnostic buffer.
2. The CPU operating system calls the diagnostic interrupt OB, e.g. OB 82.
3. You can evaluate the start info of OB 82.
4. The OB82_MOD_ADDR parameter indicates which module triggered the interrupt.
5. For additional information, call the ABS_DIAG (Page 7127) instruction.

The positioning module detects the transition to the error-free state ("outgoing")

A diagnostic interrupt is then only "outgoing" if the last error on the module has been remedied.

Sequence:

1. The positioning module detects that all errors have been eliminated and triggers a diagnostic interrupt. The "SF" LED is not lit. The diagnostic buffer is not changed.
2. The CPU operating system calls the diagnostic interrupt OB, e.g. OB 82.
3. The OB82_MOD_ADDR parameter indicates which module triggered the interrupt.
4. Evaluate the OB82_MDL_DEFECT bit.
   If this bit is "0", there are no errors on the module. You can end your evaluation here.

Diagnostic interrupts in dependence on the CPU status

- Diagnostic interrupts are blocked by the positioning module when the CPU is in STOP mode.
- If not all of the pending errors are eliminated while the CPU is in STOP mode, the positioning module reports the remaining errors as "incoming" events again at the next CPU transition to RUN.
- If all pending errors are eliminated while the CPU is in STOP mode, the module does **not** report its error-free state with a diagnostic interrupt at the next CPU transition to RUN.

Error evaluation in the user program

Error response in the user program

In the user program you can respond selectively to errors. The following methods can be used:

- The RETVAL return values of the integrated standard FB:
  This value is determined again each time the block is called.
  RETVAL = -1 is a group display for a synchronous error in a job or when communicating with the module.
- An _ERR error bit belongs to each job. This is a group display for an error in the job or one of its predecessors in the job chain:
  The error bit is set for a write job and its subsequent jobs when a data error is reported by the module or a communication error occurs.
  With reading jobs the error bit is set for the job in question when a communication error occurs.
  The ABS_CTRL (Page 7125) instruction resets the error bits once a job has been processed. These should however be cancelled by the user program in the case of an error assessment.
- The DATA_ERR checkback signal as a group display for an error that the module has detected during a write job. The signal will be established again with the next write job.
- The OT_ERR checkback signal (operator error) as a group display for an error that the module has detected when starting a traverse. The error must be acknowledged after the cause has been acknowledged with OT_ERR_A = 1.
- The DIAG checkback signal is set when the contents of the diagnostic buffer have changed. This signal can come along later than the signals DATA_ERR und OT_ERR.
- The communication error JOB_ERR includes the error code (Page 10537) in the case of a communication problem between the instruction and the module. The value is determined again after a job has been processed and is stored in the channel DBI for the ABS_CTRL operation and in the diagnostic DB for the ABS_DIAG operation.

- The ABS_DIAG (Page 7127) instruction for reading the diagnostic buffer of the module. Here you can discover the error causes for synchronous and asynchronous events.
- Diagnostic interrupts for the fast response to events in the diagnostic interrupt OB (e.g. OB 82).

Evaluation of a diagnostic interrupt in the user program

The positioning module sets the following entries in the local data of the diagnostic interrupt OB (e.g. OB 82). The errors will also be entered in the diagnostic buffer (error class 128 (Page 10545)).

Address	Name	Type	Comment
0.0	OB82_EV_CLASS	BYTE	Event class and identifiers: B#16#38: Outgoing event B#16#39: Incoming event
1.0	OB82_FLT_ID	BYTE	Error code (B#16#42)
2.0	OB82_PRIORITY	BYTE	Priority class: B#16#1A in RUN B#16#1C in START-UP
3.0	OB82_OB_NUMBR	BYTE	OB number (82)
4.0	OB82_RESERVED_1	BYTE	Reserved
5.0	OB82_IO_FLAG	BYTE	Input module: B#16#54
6.0	OB82_MDL_ADDR	INT	Logical base address of the module in which the error has occurred
8.0	OB82_MDL_DEFECT	BOOL	Module fault
8.1	OB82_INT_FAULT	BOOL	Internal error
8.2	OB82_EXT_FAULT	BOOL	External error
8.3	OB82_PNT_INFO	BOOL	Channel error
8.4	OB82_EXT_VOLTAGE	BOOL	Missing external auxiliary supply
...	Not used		
10.3	OB82_WTCH_DOG_FLT	BOOL	Time monitoring has responded
...	Not used		
12.0	OB82_DATE_TIME	DATE_AND_TIME	Date and time at which the OB was requested

Diagnostics buffer of the module

Diagnostic events

The module´s diagnostic buffer includes maximum 9 diagnostic events and is organized as a ring buffer.

A diagnostic event is written in the buffer if an "incoming" (error) message is detected. This can be a message, a synchronous error (data error, operator error) or even an asynchronous error (operational error and diagnostics error). From a cause of error several entries can occur as follow-up errors. Outgoing messages do not create any entries in the diagnostic buffer.

For each diagnostic event the following is given:

- Status (always incoming)
- Internal error
- External error
- Error class
- Error number
- Channel number
- Incremental dimension number (in the case of increment table errors)

When a diagnostic event is written in the diagnostic buffer, the DIAG = 1 checkback signal is set in all assigned channels.

Using the ABS_DIAG (Page 7127) instruction, the diagnostic buffer can be transferred as a whole to a data block (diagnostic DB) or displayed by the error evaluation dialog of the configuration software. If the diagnostic buffer is read, the module sets the checkback signal DIAG = 0.

Note

If the diagnostic buffer is read simultaneously from the ABS_DIAG operation and the error evaluation dialog, it is possible that a new diagnostic event not be detected by the program.

Data and structure of the diagnostic DB

Data and structure of the diagnostic DB

Note

You must not change any data that is not listed in this table.

The following table shows the structure of the diagnostic DB:

Address	Name	Type	Initial value	Comment
0.0	MOD_ADDR	INT	0	Module address
256.0	JOB_ERR	INT	0	Error number of the communication error
258.0	JOBBUSY	BOOL	FALSE	1 = job active
258.1	DIAGRD_EN	BOOL	FALSE	1 = definitely read diagnostics buffer
260.0	DIAG_CNT	INT	0	Number of valid entries in the list
262.0	DIAG[1]	STRUCT		Diagnostics data - latest entry
272.0	DIAG[2]	STRUCT		Diagnostics data - second entry
282.0	DIAG[3]	STRUCT		Diagnostics data - third entry
292.0	DIAG[4]	STRUCT		Diagnostics data - fourth entry
302.0	DIAG[5]	STRUCT		Diagnostics data - fifth entry

Address	Name	Type	Initial value	Comment
312.0	DIAG[6]	STRUCT		Diagnostics data - sixth entry
322.0	DIAG[7]	STRUCT		Diagnostics data - seventh entry
332.0	DIAG[8]	STRUCT		Diagnostics data - eighth entry
342.0	DIAG[9]	STRUCT		Diagnostics data - ninth entry

The diagram below shows the structure of a DIAG[n] diagnostic entry.

Address	Name	Type	Initial value	Comment
+0.0	STATE	BOOL	FALSE	0 = incoming event 1 = outgoing event
+0.1	INTF	BOOL	FALSE	1 = internal error
+0.2	EXTF	BOOL	FALSE	1 = external error
+2.0	FCL	INT	0	Error class: 1: Operating error 2: Operator errors 4: Data error 5: Machine data error 6: Incremental dimension table error 15: Alarms 128: Diagnostics errors
+4.0	FNO	INT	0	Error number
+6.0	CH_NO	INT	0	Channel number
+8.0	TRG_NO	INT	0	Incremental dimension number

List of JOB_ERR messages

JOB_ERR alarms

JOB_ERR (Hex)	JOB_ERR (Dec)	JOB_ERR (Int)	Meaning
80A0	32928	-32608	Negative acknowledgement when reading from the module. The module has been removed during the read operation - or the module is faulty.
80A1	32929	-32607	Negative acknowledge when writing to the module. The module has been removed during the write operation - or the module is faulty.
80A2 [1)]	32930	-32606	Protocol error at layer 2
80A3 [1)]	32931	-32605	Protocol error involving user interface/user
80A4 [1)]	32932	-32604	Communication on C bus interrupted
80B0	32944	-32592	Data record / job unknown.
80B1	32945	-32591	Incorrect length specification. FM_TYPE parameter in the channel DB is not correctly set for the module being used.
80B2	32946	-32590	The configured slot is not assigned.
80B3	32947	-32589	Actual module type dissimilar to the set module type.
80C0 [1)]	32960	-32576	The module does not yet have ready the data to be read.

JOB_ERR (Hex)	JOB_ERR (Dec)	JOB_ERR (Int)	Meaning
80C1 [1]	32961	-32575	The data of a same-named write job is not yet ready on the module.
80C2 [1]	32962	-32574	The module is currently processing the possible maximum of jobs.
80C3 [1]	32963	-32573	Required equipment (memory etc.) is currently in use.
80C4 [1]	32964	-32572	Communication error
80C5 [1]	32965	-32571	Distributed I/O unavailable.
80C6 [1]	32966	-32570	Priority class cancellation (restart or background)
8522	34082	-31454	Channel DB or parameter DB too short. The data cannot be read from the DB. (write job)
8532	34098	-31438	DB number of the parameter DB too high (write job)
853A	34106	-31430	Parameter DB does not exist. (write job)
8544	34116	-31420	Error with the n-th (n > 1) read access on a DB after an error has occurred. (write job)
8723	34595	-30941	Channel DB or parameter DB too short. The data cannot be written in the DB. (read job)
8730	34608	-30928	Parameter DB in the CPU is write-protected. The data cannot be written in the DB (read job).
8732	34610	-30926	DB number of the parameter DB too high (read job)
873A	34618	-30918	Parameter DB does not exist. (read job)
8745	34629	-30907	Error with the n-th (n > 1) write access on a DB after an error has occurred. (read job)

[1] Errors 80A2 to 80A4 and 80Cx are temporary, i.e., they can be rectified automatically after a waiting time.

Error classes

Operating error

Error class 1: Operating error

Operating errors are detected asynchronously to an operation / control. The operating errors bring about a cancellation of the positioning, apart from with error number 9, which causes the positioning to shut down.

The following table shows the possible operating errors and their respective causes:

No.	Meaning		Diagnostic interrupt
1	Software limit switch start overrun		Yes
	Cause	The actual value lies outside of the operating range.	
2	Software limit switch end overrun		Yes
	Cause	The actual value lies outside of the operating range.	
3	Traversing range start overrun		Yes
	Cause	Traversing range limit overrun (coordinates of the traversing range limits belong with the traversing range).	

No.	Meaning		Diagnostic interrupt
4	Traversing range end overrun		Yes
	Cause	Traversing range limit overrun (coordinates of the traversing range limits also belong to the traversing range)	
5	Error on target approach		Yes
	Cause	Target range has not been reached within the monitoring time.	
6	Standstill range exited		Yes
	Cause	The actual value lies outside of the standstill range.	
7	Positive feedback		Yes
	Cause	Actual value change > ½ standstill range in the wrong direction.	
8	Missing or too slight actual value change		Yes
	Cause	No actual value change or an actual value change against the set direction within the monitoring time.	
9	Target overrun (FM 451)		Yes
	Cause	The target has been overrun with "Set actual value on-the-fly".	
10	Target range overrun		Yes
	Cause	Target range has been overrun after target approach.	
11	Changeover point switched incorrectly		Yes
	Cause	Axis swinging in changeover point.	
12	Switch-off point switched incorrectly		Yes
	Cause	Axis swinging between switch-off / return point.	
13	Start of target range switched incorrectly		Yes
	Cause	Axis oscillating in the target range.	
14	Change greater than half the rotary axis range		Yes
	Cause	The velocity/frequency is too high or there are faulty actual value jumps.	
15	Change greater than the rotary axis range		Yes
	Cause	The velocity/frequency is too high or there are faulty actual value jumps.	
16	Increment for increment number 252 cannot be transferred (FM 451)		Yes
	Cause	The increment cannot be transferred.	
17	Increment for increment number 252 cannot be approached (FM 451)		Yes
	Cause	The distance between the current actual position and the specified increment is less than the changeover difference or switch-off difference.	
18	Incorrect increment for increment number 252 (FM 451)		Yes
	Cause	The increment lies outside of the operating range.	

Operator errors

Error class 2: Operator errors

Operator errors have been detected when changing the control signals in the user data area. The operator errors cause the positioning to shut down.

The following table shows the possible operator errors and their respective causes:

No.	Meaning		Diagnostic interrupt
1	**Impermissible operating mode**		No
	Cause	The chosen operating mode is inadmissible.	
3	**Inadmissible interface job**		No
	Cause	The selected signal is inadmissible in this operating mode.	
4	**Incorrect operating mode parameter**		No
	Cause	• In "jog" mode the velocity specification is not equal to the rapid traverse or creep speed. • In "incremental approach" operating mode, the incremental dimension is a value other than 1 - 100, 252, 254, or 255	
5	**Start enable not available**		No
	Cause	Start enable not available upon start.	
7	**Target / target range lies outside of the operating range**		No
	Cause	Specified or calculated target is outside the software limit switches.	
8	**Axis parameters not assigned**		No
	Cause	Incorrect machine data or no machine data has been assigned for the axis.	
9	**Axis not synchronized**		No
	Cause	The "incremental approach" mode is only possible with an already synchronized axis.	
10	**Target/incremental distance cannot be positioned**		No
	Cause	The distance between the current actual position and the specified target is less than the switch-off difference.	
17	**Reference point approach not possible**		No
	Cause	An SSI encoder is connected.	
18	**Relative or absolute incremental approach is not possible**		No
	Cause	The increment is invalid.	
19	**Switch-off difference is less than ½ the target range with increment number 255**		No
	Cause	The switch-off difference for the increment number 255 is less than half the target range.	
20	**Inadmissible travel in the specified direction**		No
	Cause	The distance to the software limit switch is insufficient	

Data error

Error class 4: Data error

Operating errors are detected synchronously to an operation / control. Data errors do not result in an error response.

The following table shows the possible data errors and their respective causes:

No.	Meaning		Diagnostic interrupt
6	Specified increment too great		No
	Cause	The value lies outside of ±100 m or ±1000 m. The increment/target must not be greater than the traversing range. For a rotary axis, the coordinate must be greater than or equal to 0 and less than the end of rotary axis.	
10	Faulty zero offset (FM 451)		No
	Cause	The zero offset is greater than ±100 m or ±1000 m. The software limit switches are positioned after the zero offset outside of the traversing range (100 m to +100 m or -1000 m to +1000 m). Rotary axis: The zero offset amount is greater than the end of the rotary axis.	
11	Incorrect actual value		No
	Cause	Linear axis: The coordinate lies outside the current (possibly shifted) software limit switches. Rotary axis: The coordinate is < 0 or greater than the end of rotary axis.	
12	Incorrect reference point		No
	Cause	Linear axis: The coordinate lies outside the current (possibly shifted) software limit switches. Rotary axis: The coordinate is < 0 or greater than the end of rotary axis.	
20	Not permissible to activate machine data		No
	Cause	There are no new, error-free machine data on the module.	
27	Unauthorized bit-coded setting		No
	Cause	Bits that are not used and not described here are not equal to 0.	
29	Inadmissible bit coding		No
	Cause	Unused and, in this case, unwritten bits are not 0.	
34	Not possible to cancel set actual value		No
	Cause	With an SSI encoder and a linear axis, the location of the actual value would lie outside the operating range after making the setting.	
36	Incorrect changeover difference with the increment number 255		No
	Cause	The value is outside the permitted numeric range of ±100 m or ±1000 m. For a rotary axis, the coordinate must be greater than or equal to 0 and less than the end of rotary axis.	
37	Incorrect switch-off difference with the increment number 255		No
	Cause	The value is outside the permitted numeric range of ±100 m or ±1000 m. The switch-off difference must be less than the changeover difference.	
107	Axis parameters not assigned		No
	Cause	On the axis there is either no machine data or it has not been enabled.	
108	Axis not synchronized		No
	Cause	One of the "set actual value" and "set actual value on-the-fly" jobs has been triggered even though the axis is not synchronized.	

Machine data error

Error class 5: Machine data error

The diagnostic interrupt is only triggered in the event of a faulty system data block (SDB). Machine data errors do not result in an error response.

The following table shows the possible machine data errors and their respective causes:

No.	Meaning		Diagnostic interrupt
5	**Error in hardware interrupt setting**		Yes
	Cause	You have attempted to select a hardware interrupt that the module does not support.	
6	**Incorrect minimum edge distance (FM 451)**		Yes
	Cause	You have entered a value < 0 or > 10^9 µm as the minimum edge-to-edge distance.	
7	**Incorrect system of units**		Yes
	Cause	The value for the system of units lies outside of the permissible range of 1 to 4 and 6.	
8	**Incorrect axis type**		Yes
	Cause	You have entered neither 0 nor 1 as an axis type.	
9	**Incorrect end of rotary axis**		Yes
	Cause	The value for the end of the rotary axis is outside the valid range from 1 to 10^9 µm or 1 to 10^8 µm (depending on the resolution).	
10	**Incorrect encoder type**		Yes
	Cause	The value for the encoder type lies outside of the permissible range of 1 to 4.	
11	**Incorrect distance per encoder revolution**		Yes
	Cause	The value for distance per encoder revolution lies outside the permissible range of 1 to 10^9 µm (irrespective of the resolution).	
13	**Incorrect number of increments per encoder revolution** (see "Increments per encoder revolution (Page 10518)")		Yes
14	**Incorrect number of revolutions** (see "Number of encoder revolutions (Page 10521)")		Yes
15	**Incorrect transmission rate**		Yes
	Cause	You have specified a value for the transmission rate that lies outside the permissible range of 0 to 3.	
16	**Incorrect reference point coordinate**		Yes
	Cause	The coordinate lies outside the range from -100 m to +100 m or -1000 m to +1000 m (depending on the resolution). Linear axis: The coordinate lies outside of the operating range. Rotary axis: The coordinate is greater than the end of the rotary axis or < 0.	
17	**Incorrect absolute encoder adjustment**		Yes
	Cause	SSI navigator: The value of the absolute encoder adjustment is not in the encoder range: (increments per encoder revolution x number of revolutions - 1).	
18	**Incorrect type of reference point approach**		Yes
	Cause	You have specified a value outside of the permissible value quantities of 0, 1, 2 and 3.	
19	**Incorrect counting direction**		Yes
	Cause	You have specified a value outside of the permissible value quantities of 0 and 1.	

No.	Meaning		Diagnostic interrupt
20	It is not possible to monitor the hardware		Yes
	Cause	You have set frame error monitoring to "FALSE" in the parameter DB. The monitoring of missing pulses is not possible with encoders being used. Deactivate the MON_PULSE parameter.	
21	Incorrect software limit switch start		Yes
	Cause	Linear axis: The software limit switch start is outside the traversing range (-100 m to +100 m or 1,000 m to +1000 m, depending on the resolution). Linear axis: The software limit switch start (including any existing zero offset) is less than -100 m or -1000 m (depending on the resolution).	
22	Incorrect software limit switch end		Yes
	Cause	Linear axis: The software limit switch end is outside the traversing range (-100 m to +100 m or 1,000 m to +1000 m, depending on the resolution) or is less than the software limit switch start. Linear axis: The software limit switch end (including any existing zero offset) is greater than +100 m or +1000 m (depending on the resolution).	
23	Incorrect maximum velocity		Yes
	Cause	The data that is not listed in the parameter DB must be 0.	
24	Incorrect target range		Yes
	Cause	Linear axis: Range between 0 and 100 m or 0 and 1000 m depending on resolution. Rotary axis: Range greater than the end of rotary axis.	
25	Incorrect monitoring time		Yes
	Cause	The value for the monitoring time lies outside the permissible range from 0 to 100 000 ms.	
26	Incorrect standstill range		Yes
	Cause	Linear axis: Range between 0 and 100 m or 0 and 1000 m, depending on the resolution. Rotary axis: Range greater than the end of rotary axis.	
127	Incorrect standstill velocity		Yes
	Cause	The value for the standstill velocity lies outside the permissible range from 0 to 100 000 µm/min.	
128	Incorrect control mode		Yes
	Cause	You have specified a value for the control mode that lies outside the permissible range of 1 to 4.	
129	Incorrect start velocity for the reference point approach		Yes
	Cause	You have entered neither 0 nor 1 as a start velocity.	
130	Incorrect changeover difference in the direction +		Yes
	Cause	Linear axis: Range between 0 and 100 m or 0 and 1000 m (depending on resolution). Rotary axis: Range greater than the end of rotary axis and smaller than ½ of the target range.	
131	Incorrect changeover difference in the - direction		Yes
	Cause	Linear axis: Range between 0 and 100 m or 0 and 1000 m (depending on the resolution). Rotary axis: Range greater than the end of rotary axis and smaller than ½ of the target range.	
132	Incorrect switch-off difference in the direction +		Yes

No.	Meaning		Diagnostic interrupt
	Cause	The switch-off difference is greater than the switch-over difference plus, less than ½ of the target range, or is outside the permissible range between 0 to 100 m or 1000 m (depending on the resolution)	
133	**Incorrect switch-off difference in the - direction**		Yes
	Cause	The switch-off difference is greater than the switch-over difference minus, less than ½ of the target range, or is outside the permissible range between 0 to 100 m or 1000 m (depending on the resolution)	
200	**Incorrect resolution**		Yes
	Cause	You have specified a resolution < 0.1 µm/pulse or >1000 µm/pulse. You have specified a distance per encoder revolution and a number of pulses per encoder revolution that results in a resolution of < 0.1 or > 1000.	
201	**Encoder does not fit the operating range / rotary axis range**		Yes
	Cause	SSI encoders and rotary axes: The encoder does not precisely cover the rotary axis range. Linear axis: The encoder does not cover at least the operating range (including software limit switches).	

Incremental dimension table error

Error class 6: Incremental dimension table error

The incremental dimension table errors do not result in an error response.

The following table shows the possible incremental dimension errors and their respective causes:

No.	Meaning		Diagnostic interrupt
6	**Specified increment in the increment table too great**		No
	Cause	The value lies outside of ±100 m or ±1000 m. The increment/target must not be greater than the traversing range. For a rotary axis, the coordinate must be greater than or equal to 0 and less than the end of rotary axis.	

Alarms

Error class 15: Alarms

Alarms do not result in an error response.

The following table shows the possible alarms and their respective causes:

No.	Meaning		Diagnostic interrupt
1	**Start of parameter assignment**		No
	Cause	The module has detected a parameter assignment via a system data block.	
2	**End of parameter assignment**		No
	Cause	The module has processed the parameter assignment via a system data block without errors.	
11	**Distance to change-over point too short**		No
	Cause	The hardware response times cannot be adhered to because the distance between the switching points is too short.	
12	**Distance to return point too short**		No
	Cause	The hardware response times cannot be adhered to because the distance between the switch points is too short	
14	**Distance to cut-off point too short**		No
	Cause	The hardware response times cannot be adhered to because the distance between the switch points is too short	
15	**Distance to start of target range too short**		No
	Cause	The hardware response times cannot be adhered to because the distance between the switch points is too short	

Diagnostic errors

Error class 128: Diagnostic errors

The following table shows the possible diagnostic errors and their respective causes, effects and remedies.

No.	Meaning		Diagnostic interrupt
4	**Missing external auxiliary supply**		Yes
	Cause	• External 24 V auxiliary voltage is not connected or has failed • The fuse on the module is defective • Undervoltage • Ground wire break • Short circuit (e.g. at the encoder connected)	
	Effect	• The positioning is cancelled on all channels • Output switch-off • Synchronization is cancelled for incremental encoders when the auxiliary voltage for the encoder supply is missing • Parameters have not been assigned to the positioning module • Start enable is deleted	
	Remedy	Make sure that the 24 V connection is correct (if the 24 V connection is correct, then the module is defective)	
5	**Front connector is missing (FM 451)**		Yes
	Cause	Front connector of the positioning module is not inserted	

No.	Meaning		Diagnostic interrupt
	Effect	• No external 24 V auxiliary supply • Module is not ready	
	Correction	Plug the front connector into the positioning module	
51	**Time monitoring activated (watchdog)**		Yes
	Cause	• Strong interference on the positioning module • Error in the positioning module	
	Effect	• Module is reset • Turn off all outputs • If, after resetting the module, no module fault is detected, the module is ready for operation again • The module reports the expired watchdog with "incoming" and "outgoing"	
	Correction	• Remedying interferences • Contact the relevant sales department which will require details of the exact circumstances leading to the error • Replace the positioning module	
144	**Encoder wire break**		Yes
	Cause	• The encoder cable has sheared off or is not plugged in • Encoders without cross signals • Incorrect pin assignment • Cable length too long • Encoder signals short-circuited • Encoder signal edge fault • Maximum input frequency of the encoder input has been exceeded • Sensor voltage supply failure	
	Effect	• The positioning is canceled • Output switch-off • Delete the synchronization with incremental encoders • Start enable is deleted	
	Correction	• Check encoder cable • Comply with encoder specification • Monitoring can be temporarily hidden at the responsibility of the operator using the parameter assignment dialog • Comply with the technical specifications of the module	
145	**Absolute encoder frame error**		Yes

No.	Meaning		Diagnostic interrupt
	Cause	The frame traffic between the positioning module and the absolute encoder (SSI) is incorrect or interrupted: • Encoder cable cut or not plugged in • Incorrect encoder type • The encoder is incorrectly set (programmable encoder) • The message length has been incorrectly specified • The encoder is supplying incorrect values (encoder is faulty) • Interference on measuring system cable • Chosen transmission rate is too high • Monoflop time of the encoder is greater than 64 µs	
	Effect	• The positioning is canceled • Output switch-off • Start enable is deleted	
	Correction	• Check encoder cable • Check encoder • Check the frame traffic between the encoder and positioning module	
146	**Missing pulses of incremental encoder**		Yes
	Cause	• Encoder monitoring has established a missing pulse • Incorrect entry of the number of increments per encoder revolution • Encoder faulty: does not return the specified number of pulses • incorrect or no zero mark • Crosstalk on the encoder cable	
	Effect	• The positioning is canceled • Output switch-off • Start enable is deleted	
	Correction	• Enter the correct number of increments per encoder revolution (programming dialog) • Check the encoder and encoder cable • Comply with the shielding and grounding directives • Monitoring can be temporarily hidden at the responsibility of the operator using the parameter assignment dialog	

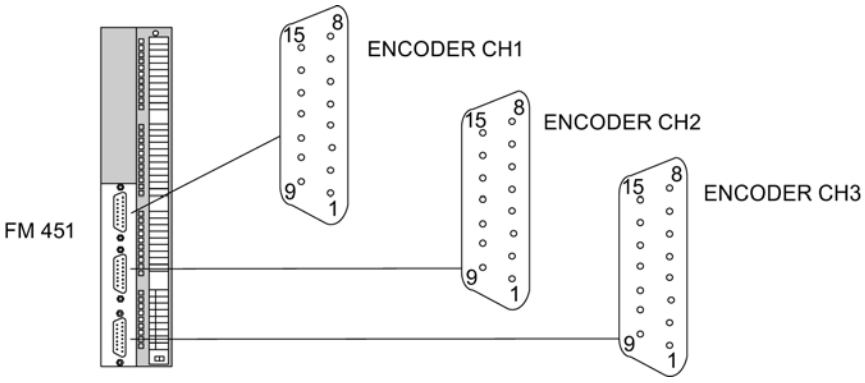
11.4.2.6 Using FM 451

Terminal assignment of FM 451 front connector

Description of the encoder interface

Location of the sub D sockets

The figure shows the position and designation of the connectors on the module. You can connect an incremental encoder or absolute encoder (SSI) to the sub D sockets.



CH = Channel

Assignment of the encoder interfaces (CH1, CH2, and CH3 encoders)

Pin	Name	Incremental encoder (24 V)	Incremental encoder (5 V)	Absolute encoder
1	A*	Encoder signal A	---	---
2	CLS	---	---	SSI shift clock
3	/CLS	---	---	SSI shift clock inverse
4	B*	Encoder signal B	---	---
5	24 VDC	Sensor supply	Sensor supply	Sensor supply
6	5.2 VDC	---	Sensor supply	Sensor supply
7	M	Chassis ground	Chassis ground	Chassis ground
8	N*	Zero mark signal	---	---
9	RE	Current sourcing/current sinking[1)]	---	---
10	N	---	Zero mark signal	---
11	/N	---	Zero mark signal, inverse	---
12	/B	---	Encoder signal B, inverse	---
13	B	---	Encoder signal B	---
14	/A / /DAT	---	Encoder signal A, inverse	SSI data inverse

Pin	Name	Incremental encoder (24 V)	Incremental encoder (5 V)	Absolute encoder
15	A / DAT	---	Encoder signal A	SSI data
[1]**Note:** If you are connecting an incremental encoder from a third party in series-mode switching (current sourcing/current sinking), the following must be observed: • Sourcing output: Connect RE (9) with ground (7). • Sinking output: Connect RE (9) with +24 V (5).				

Terminal assignment of the front connector

Front connector

Connect the following to the 48-pin front connector:

- the digital inputs and outputs assigned to the channels
- the supply voltage of the encoders and digital outputs

Terminal assignment of front connector (X1)

The table below shows the terminal assignment of the front connector of FM 451.

Wiring	Name	Function	Incremental encoder	Absolute encoder
1		Assigned; contains cable jumpers for detection of inserted front connector.		
2				
3	1L+	24 V DC auxiliary voltage for the encoder supply		
4 to 7		Not used		
8	1I0	Channel 1: Digital input 0	Reference point switch	Not used
9	1I1	Channel 1: Digital input 1	Reversing switch	Not used
10	1I2	Channel 1: Digital input 2	Enable input	
11	1I3	Channel 1: Digital input 3	Input for "Set process value on-the-fly"/ length measurement/edge detection/ external event for increment number 252	
12		Not used		
13	2L+	24 V DC auxiliary voltage for the load current supply		
14	2L+	Both terminals are internally connected on the module.		
15	2I0	Channel 2: Digital input 0	Reference point switch	Not used
16	2I1	Channel 2: Digital input 1	Reversing switch	Not used
17	2I2	Channel 2: Digital input 2	Enable input	
18	2I3	Channel 2: Digital input 3	Input for "Set process value on-the-fly"/ length measurement/edge detection/ external event for increment number 252	
19	3I0	Channel 3: Digital input 0	Reference point switch	Not used
20	3I1	Channel 3: Digital input 1	Reversing switch	Not used
21	3I2	Channel 3: Digital input 2	Enable input	

Wiring	Name	Function	Incremental encoder	Absolute encoder
22	3I3	Channel 3: Digital input 3	Input for "Set process value on-the-fly"/ length measurement/edge detection/ external event for increment number 252	
23 to 24		Not used		
25	3L+	24 V DC auxiliary voltage for the load current supply		
26	3L+	Both terminals are internally connected on the module.		
27	1Q0	Channel 1: Digital output 0		
28	1Q1	Channel 1: Digital output 1		
29	1Q2	Channel 1: Digital output 2		
30	1Q3	Channel 1: Digital output 3		
31	2Q0	Channel 2: Digital output 0		
32	2Q1	Channel 2: Digital output 1		
33	2Q2	Channel 2: Digital output 2		
34	2Q3	Channel 2: Digital output 3		
35 to 36		Not used		
37	4L+	24 V DC auxiliary voltage for the load current supply		
38	4L+	Both terminals are internally connected on the module.		
39	3Q0	Channel 3: Digital output 0		
40	3Q1	Channel 3: Digital output 1		
41	3Q2	Channel 3: Digital output 2		
42	3Q3	Channel 3: Digital output 3		
43 to 47		Not used		
48	M	Ground for auxiliary voltages 1L+, 2L+, 3L+, and 4L+		

Additional information about design and wiring can be found in the Operating Instructions (http://support.automation.siemens.com/WW/view/en/1118070) of the positioning module FM 451.

Auxiliary voltage for the encoder supply (1L+, M)

Here, you connect a 24 V DC auxiliary voltage for the encoder supply. The reference potential of this supply (M) is connected to the ground of the load current supply (M) in the FM 451.

The 24 V DC auxiliary voltage for the encoder supply is monitored for undervoltage and wire break.

This supplies 24 V DC and 5.2 V DC at the encoder interface (sub D sockets encoder CH1, CH2 and CH3) to the various types of encoders.

CAUTION

Ensure the correct polarity of the 24 V DC auxiliary voltage for the encoder supply (1L+, M).

If you reverse the polarity of the 24 V DC auxiliary voltage for the encoder supply, the module will be damaged and will have to be replaced.

Auxiliary voltage for the load current supply (2L+, 3L+, 4L+, M)

You must connect the 24 V DC auxiliary voltages for the load current supply of the digital outputs to the 2L+, 3L+, 4L+, and M terminals.

⚠ CAUTION
Verify the correct polarity of the 24 V DC auxiliary voltage for the load current supply (2L+, 3L+, 4L+, M). If you reverse the polarity of the 24 V DC auxiliary voltage for the load current supply, the module will be damaged and will have to be replaced.

Wiring information for 24 V DC

When wiring, note that the terminals 1L+ to 4L+ must be interconnected so that the module will operate error-free.

Start by wiring the 24 V DC to terminal 38 and connect the auxiliary voltage of terminal 37 to terminal 26. Use the same procedure for the other terminals.

Alternatively, you can also connect separate voltage supplies to the 1L+ (encoder), 2L+ (channel 1), 3L+ (channel 2), and 4L+ (channel 3) connections. Make sure that all voltage supplies have a common ground potential. In this case, synchronization of the axes will be retained following a power failure in the auxiliary voltages for the load current supply.

Load current supplies

The DC load current supply must meet the following requirements:

Only a safe, isolated extra-low voltage of ≤ 60 V DC may be used as the load current supply. Safe isolation can be implemented, for example, by adhering to the specifications

- VDE 0100 Part 410 / HD 384-4-41 / IEC 364-4-41
  (as protective extra low voltage) and/or
- VDE 0805 / EN 60950 / IEC 950
  (as safety low voltage SELV) or
- VDE 0106 Part 101.

12 digital inputs (1I0 to 3I3)

The FM 451 has 4 digital inputs per channel.

You can connect bounce-free switches (24 V current sourcing) or non-contact sensors (2- or 3-wire proximity switches) to the 12 digital inputs.

The digital inputs are not monitored for short circuits or wire break and have a non-isolated connection to the encoder supply ground.

A separate LED indicates the state of each input.

12 digital outputs (1Q0 to 3Q3)

The FM 451 has 4 digital outputs per channel.

The power unit is controlled via the digital outputs. The function of the digital outputs depends on the control mode (Page 10596).

The digital outputs are not monitored for short circuits or wire break and have a non-isolated connection to the encoder supply ground.

The associated LED indicates the state of each output.

Functions of the FM 451

Overview of functions

You define module functionality by setting the mode, job or function switch. The table below gives an overview of the functionality of the module.

Functionality	Description
Writing and reading machine data (Page 10553)	You can use the machine data to adapt the positioning module to the axis and the encoder.
Writing and reading increment tables (Page 10555)	The incremental dimensions define the targets which are controlled by the positioning module with the "Relative / absolute incremental approach" mode.
End of a positioning operation (Page 10557)	The end of a positioning operation is achieved in three different ways: • Target approach • Shutdown • Cancel
"Inching" mode (Page 10564)	In "Inching" mode you can move the drive in one direction at the press of a button.
"Reference point approach" mode (Page 10567)	The "Reference point approach" mode can be used to synchronize the axis based on a recurring external event.
"Incremental approach" mode (Page 10573)	With the "Incremental approach" mode, the positioning module can move the drive • toward absolute targets, • relatively by an incremental distance in a specified direction.
The "Set actual value" / Set actual value on-the-fly / "Undo Set Actual Value" job (Page 10580)	With the "Set actual value" or "Set actual value on-the-fly", you assign a new coordinate to the actual encoder value. With the "Undo set value" job, you reset all the operating range shifts that were generated by means of the "Set actual value" job.
"Zero offset" job (Page 10582)	The "zero offset" job lets you shift the zero point in the coordinate system by a defined value.
"Set reference point" job (Page 10584)	With the "Set reference point" job you synchronize the axis.
"Length measurement" / "Edge detection" function switches (Page 10585)	"Length measurement" and "Edge detection" let you determine the length of a workpiece.

Functionality	Description
"Loop traverse" function switch (Page 10588)	You use "Loop traverse" to specify the direction in which a target is approached with a frictional connection.
"Enable input" function switch (Page 10591)	The enable input is an external input with which positioning can be enabled by an external event.
"Read position data" job (Page 10592)	With the "read position data" job you can read the incremental dimension, residual distance and velocity at the current time.
"Read encoder data" job (Page 10592)	With the "read encoder data" job you can read the current data of the encoder as well as the value for the absolute encoder adjustment.
Checkback signals for positioning (Page 10593)	The "Feedback signals for positioning" inform you of the current status of the positioning.
Checkback signals for diagnostics (Page 10593)	The "Feedback signals for diagnostics" informs you of occurring diagnostic events.

Writing and reading machine data

Writing, enabling and reading machine data

You can use the machine data to adapt the positioning module to the axis and the encoder.

You can change and read parameters during operation using the user program.

All parameters are stored in the parameter DB:

- Machine data is located in the parameter DB at the addresses 4.0 to 116.0.

Enter the parameter DB number in the corresponding channel DB.

You can enter the parameters with the DB Editor or also easily using the "Drive", "Axis", "Encoder" and "Increments" dialogs and then write in the parameter DB with the "Export" function.

You can use the "Import" function to import parameters from an existing parameter DB to the dialogs.

First parameter assignment

If the channel still contains no machine data, proceed as with a first parameter assignment without the parameter assignment dialogs in the following way:

1. Enter the new values in the parameter DB and then save these.
2. Download the parameter DB to the CPU.
3. Set the following trigger bit in the channel DB for the job:
   – Write machine data (MDWR_EN = 1).
4. Call the ABS_CTRL (Page 7125) or ABS_CTRL_451 (Page 7129) instruction in the cyclic user program.

Modifying machine data

Proceed as follows to change machine data using the user program:

1. Enter the new values in the parameter DB.
2. Set the trigger bits in the channel DB for the jobs:
   – Write machine data (MDWR_EN = 1)
   – Activate machine data (MD_EN = 1)
3. Call the ABS_CTRL (Page 7125) or ABS_CTRL_451 (Page 7129) instruction in the cyclic user program.

If you set the trigger bits for these jobs all at once, the ABS_CTRL instruction (or ABS_CTRL_451) makes sure the jobs are processed in the correct order.

Otherwise change the machine data always in the following sequence:

- Write machine data
- Enable machine data

Reading machine data

In order to read the current machine data from a channel, proceed as follows:

1. Set the following trigger bit in the channel DB:
   – Reading machine data (MDRD_EN = 1)
2. Call the ABS_CTRL (Page 7125) or ABS_CTRL_451 (Page 7129) instruction in the cyclic user program.

The current machine data is then stored in the parameter DB on the CPU.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
35.0	MDWR_EN	BOOL	FALSE	1 = Write machine data
35.1	MD_EN	BOOL	FALSE	1 = Enable machine data
36.5	MDRD_EN	BOOL	FALSE	1 = read machine data

Note

If synchronization-relevant parameters have been changed, when the machine data is enabled the module will carry out the following actions for the respective channel:

- Synchronization will be deleted.
- The function switches and zero offset will be reset.
- All previous machine data and incremental dimension tables will become invalid.

Parameters relevant to synchronization:

- Axis type
- End of rotary axis
- Signal type
- Distance per encoder revolution
- Increments per encoder revolution
- Number of revolutions
- Reference point coordinate
- Absolute encoder adjustment
- Type of reference point approach
- Count direction

Writing and reading incremental dimension tables

Writing and reading incremental dimension tables

The incremental dimensions define the targets which are controlled by the positioning module with the "Relative / absolute incremental approach" mode.

You can change and read parameters during operation using the user program.

All parameters are stored in the parameter DB:

- Increment tables are located in the parameter DB at the addresses 120.0 to 516.0.

The number of the parameter DB must be entered in the respective associated channel DB.

You can enter the parameters with the DB Editor or also easily using the "Incremental dimensions" dialog and then write in the parameter DB with the "Export" function.

You can use the function "Import" to import parameters that are in an already available parameter DB into the dialogs.

First parameter assignment

If the channel still contains no increment tables, proceed as with a first parameter assignment without the parameter assignment dialog in the following way:

1. Enter the new values in the parameter DB and then save these.
2. Download the parameter DB to the CPU.
3. Set the trigger bits in the channel DB for the jobs:
   – Writing incremental dimension table 1 (TRGL1WR_EN = 1) and / or writing incremental dimension table 2 (TRGL2WR_EN = 1).
4. Call the ABS_CTRL (Page 7125) or ABS_CTRL_451 (Page 7129) instruction in the cyclic user program.

Changing increment tables

Proceed as follows to change increment tables using the user program:

1. Enter the new values in the parameter DB.
2. Set the trigger bits in the channel DB for the jobs:
   – Writing incremental dimension table 1 (TRGL1WR_EN = 1) and / or writing incremental dimension table 2 (TRGL2WR_EN = 1).
3. Call the ABS_CTRL (Page 7125) or ABS_CTRL_451 (Page 7129) instruction in the cyclic user program.

Reading increment tables

In order to read the increment tables from a channel, proceed as follows:

1. Set the trigger bits in the channel DB for the jobs:
   – Reading incremental dimension table 1 (TRGL1RD_EN = 1) and / or writing incremental dimension table 2 (TRGL2RD_EN = 1).
2. Call the ABS_CTRL (Page 7125) or ABS_CTRL_451 (Page 7129) instruction in the cyclic user program.

The increment tables are then stored in the parameter DB on the CPU.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
35.4	TRGL1WR_EN	BOOL	FALSE	1 = write increment table 1 (1 ... 50)
35.5	TRGL2WR_EN	BOOL	FALSE	1 = write increment table 2 (51 ... 100)
36.6	TRGL1RD_EN	BOOL	FALSE	1 = read increment table 1 (1 ... 50)
36.7	TRGL2RD_EN	BOOL	FALSE	1 = read increment table 2 (51 ... 100)

Note

If synchronization-relevant parameters have been changed, when the machine data is enabled the module will carry out the following actions for the respective channel:

- Synchronization will be deleted.
- The function switches and zero offset will be reset.
- All previous machine data and incremental dimension tables will become invalid.

Parameters relevant to synchronization:

- Axis type
- End of rotary axis
- Signal type
- Distance per encoder revolution
- Increments per encoder revolution
- Number of revolutions
- Reference point coordinate
- Absolute encoder adjustment
- Type of reference point approach
- Count direction

End of a positioning operation

Definition

The end of a positioning operation is displayed by the checkback signal WORKING = 0. It can be reached in three different ways:

- Target approach
- End
- Cancel

Monitoring

The following monitoring functions are active towards the end of a positioning operation:

- Monitoring time
  The monitoring time is retriggered at the switch-off point for the last time and then loses its validity with the end of positioning.
  The end of the positioning operation must be reached within this time. If it is not, the outputs will be switched off and the operating error "Error during target approach" (error number 5) issued.
- Monitoring the target range
  The FM x51 sets a symmetrical range around each target, thereby defining the positioning precision of your application. The axis must come to a standstill within this range during a target approach. A value 0 switches off the tolerance during the target approach.
- Monitoring the standstill velocity
  The standstill velocity is used determine whether the drive comes to a standstill within the target range. The FM x51 checks whether the actual drive speed is less than the stationary drive speed when the switch-off point has been reached.
  It must be less than the standstill velocity within the target range. The FM x51 will otherwise report the operating error "Target range overrun" (error number 10).
  Falling short of the standstill velocity is monitored only once per target approach.
  Note that the velocity can briefly fall below the standstill velocity for the velocity determination of the module if the axis moves at a very slow positioning velocity (less than 2 pulses per 8 ms).
- Monitoring the stationary range
  Once a positioning operation ends, the drive is monitored to see whether it remains at an approached target position or drifts off.
  The standstill range is monitored
  - after the FM x51 has reported the checkback signal "PR"
  - if the monitoring time is exceeded
  - if the velocity falls below the standstill velocity.

  If the standstill range is exited without a valid traversing job, the FM x51 reports the "Stationary range exited" error (error number 6).

Target approach

The target approach in the "Absolute incremental approach" and "Incremental approach" modes starts when the switch-off point has been reached. As of this point the drive is switched off and the FM x51 adopts the monitoring functions.

Depending on the monitoring you have parameterized, there are a number of different scenarios in which the checkback signal "PR (POS_RCD)" is generated. The positioning operation is cancelled if no "PR (POS_RCD)" checkback signal is generated.

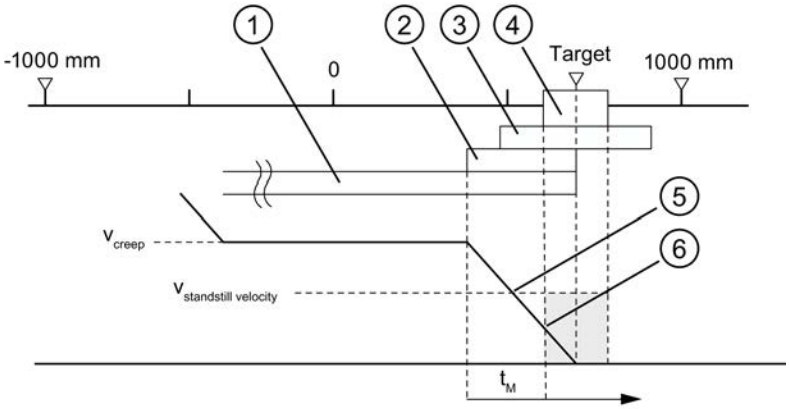
Scenario 1: You have assigned parameters as follows:

- Target range (TRG_RANGE) > 0
- Standstill velocity (ZSPEED_L) > 0
- Monitoring time (MON_TIME) > 0

PR is generated when the velocity falls below the standstill velocity and the target range has been reached. It does not matter which condition is fulfilled first.

PR is not generated if the actual value has not reached the target range within the monitoring time or the target range has been overrun without the velocity falling below the standstill velocity.

The figure below shows the target approach in incremental approach mode with a target range > 0 and standstill velocity > 0.



① Changeover difference plus
② Switch-off difference plus
③ Stationary range
④ Target range
⑤ Standstill velocity reached
⑥ Target range with V_{still} reached: PR is set
t_M Monitoring time

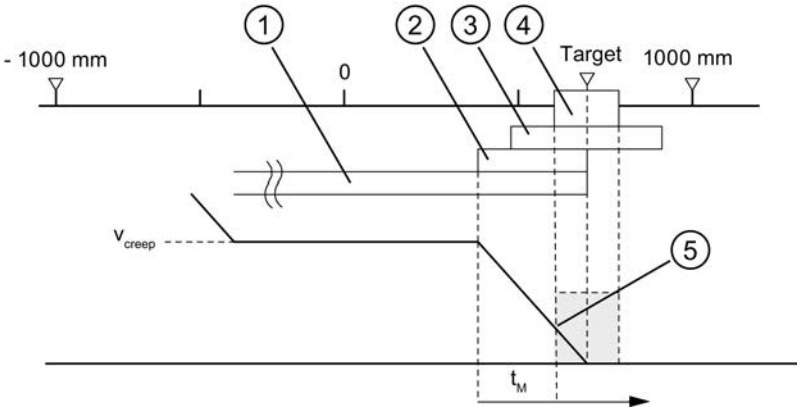
Scenario 2: You have assigned parameters as follows:

- Target range (TRG_RANGE) > 0
- Standstill velocity (ZSPEED_L) = 0
- Monitoring time (MON_TIME) > 0

PR is generated when the target range has been reached.

PR is not generated if the actual value does not reach the target range within the monitoring time.

The figure below shows the target approach in incremental approach mode with a target range > 0 and standstill velocity = 0.



① Changeover difference plus
② Switch-off difference plus
③ Stationary range
④ Target range
⑤ Target range with V_{still} reached: PR is set
t_M Monitoring time

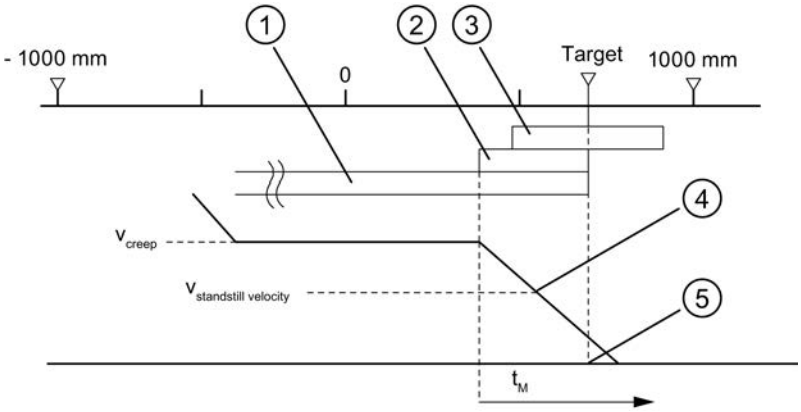
Scenario 3: You have assigned parameters as follows:

- Target range (TRG_RANGE) = 0
- Standstill velocity (ZSPEED_L) > 0
- Monitoring time (MON_TIME) > 0

PR is generated if the velocity falls below the standstill velocity and the target is then reached.

PR is not generated if the actual value has not reached the target within the monitoring time or the target range has been overrun without the velocity falling below the standstill velocity.

The figure below shows the target approach in incremental approach mode with a target range = 0 and standstill velocity > 0.



① Changeover difference plus
② Switch-off difference plus
③ Stationary range
④ Standstill velocity reached
⑤ Target reached: PR is set
t_M Monitoring time

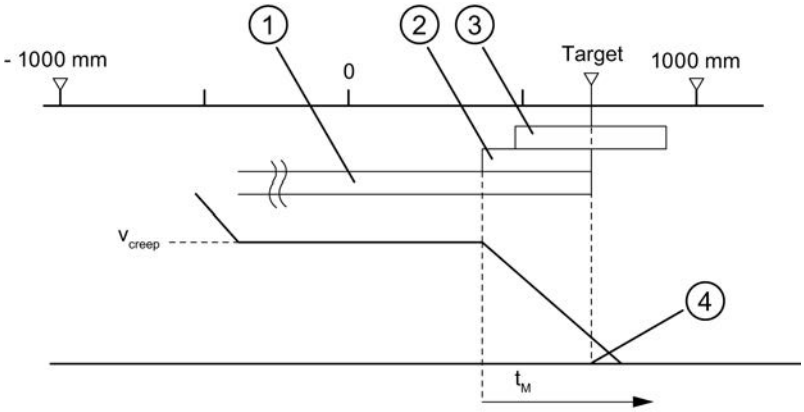
Scenario 4: You have assigned parameters as follows:

- Target range (TRG_RANGE) = 0
- Standstill velocity (ZSPEED_L) = 0
- Monitoring time (MON_TIME) > 0

PR is generated when the target has been reached.

PR is not generated if the actual value does not reach the target within the monitoring time.

The figure below shows the target approach in incremental approach mode with a target range = 0 and standstill velocity = 0.



① Changeover difference plus
② Switch-off difference plus
③ Stationary range
④ Target reached: PR is set
t_M Monitoring time

Scenario 5: You have assigned parameters as follows:

- Target range (TRG_RANGE) ≥ 0
- Standstill velocity (ZSPEED_L) ≥ 0
- Monitoring time (MON_TIME) = 0

If, in this case, the drive comes to a standstill before the target range, the end of the positioning operation will not be detected. PR will not be generated and the checkback signal WORKING will remain set. You can only cancel the positioning operation only by deleting the drive enable (DRV_EN = 0).

Shutdown without specified target

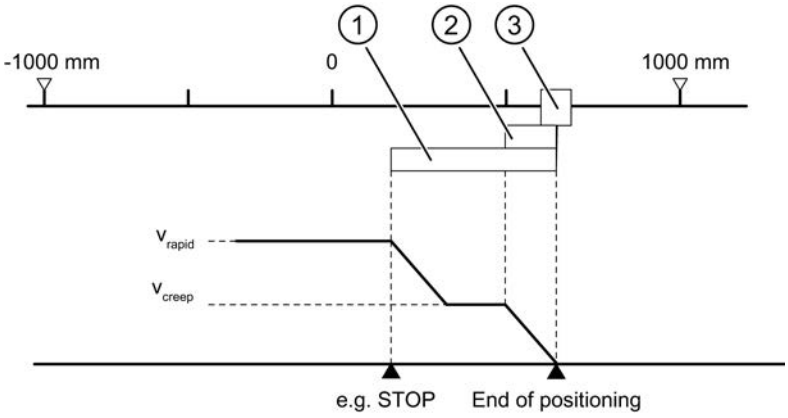
Shutdown means: The positioning operation is purposefully ended whilst complying with the differences between rapid traverse and creep speed.

The positioning operation is ended if

- The FM x51 receives a STOP signal (STOP = 1)
- The operating modes "Jog" and "Reference point approach" are ended.
- Operating errors or the operating error "Target overrun" (error number 9) occur.

The checkback signal "PR (POS_RCD)" is not set. The order of events are analogous to those of the target approach.

The figure below shows the end of a positioning operation.



① Changeover difference plus
② Switch-off difference plus
③ Target range

Cancel

Shutdown means: The positioning process is ended immediately after standstill without the use of the changeover and switch-off differences between rapid traverse and creep speed. In addition, all the relevant outputs of the respective control mode are immediately switched off and also:

- Increment = actual value
- Distance-to-go = zero

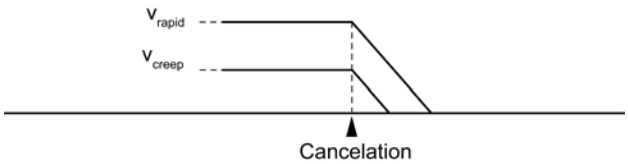
The positioning operation is cancelled if

- The drive enable signal is deleted (DRV_EN = 0).
- The CPU goes into STOP mode.
- Diagnostic errors or all operating errors apart from the "Target overrun" operating error (error number 9) occur.

The checkback signal "PR (POS_RCD)" is not be set in "Incremental approach" mode.

If the standstill velocity is assigned, the standstill monitoring will become active as soon as the velocity falls below the standstill velocity. If the standstill velocity is not assigned, the standstill monitoring will be active when the outputs have been switched off.

The figure below shows the cancellation of a positioning operation.



Data used in the parameter DB

Address	Name	Type	Initial value	Comment
76.0	TRG_RANGE	DINT	L#1000	Target range
80.0	MON_TIME	DINT	L#2000	Monitoring time
84.0	ZSPEED_R	DINT	L#1000	Stationary range
88.0	ZSPEED_L	DINT	L#30000	Standstill velocity

Checkback signals in the channel DB

Address	Name	Type	Initial value	Comment
23.1	WORKING	BOOL	FALSE	1 = Positioning operation is running
25.7	POS_RCD	BOOL	FALSE	1 = position reached

"Jog" mode

Definition

In "Jog" mode, you can move the drive in one direction by pushing a button. You must install a button for both directions (plus and minus) respectively. You can use the "Jog" mode both for a synchronized as well as for an unsynchronized axis.

In the case of an unsynchronized axis, jogging is a positioning in the specified direction.

For a synchronized axis (linear axis), jogging involves a positioning to the software limit switches.

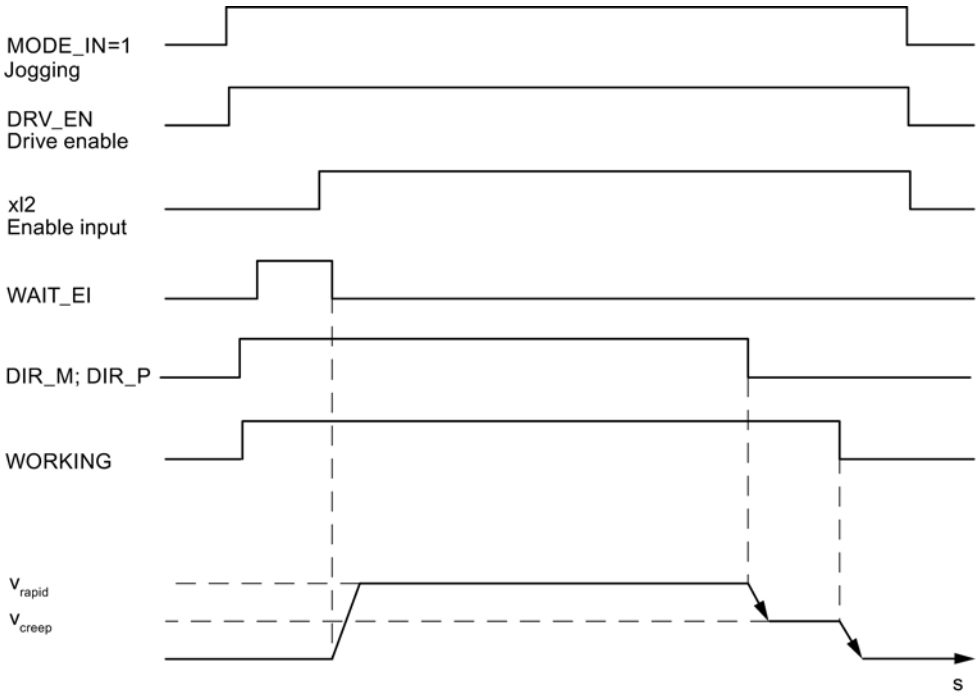
Requirement

The axis parameters must be assigned.

Sequence of events for "Jog" mode

1. Set the control signal for the "Jogging" mode (MODE_IN = 1).
2. Set the control signal for the drive enable (DRV_EN = 1).
3. Set the function switch for "Do not evaluate enable input" (EI_OFF = 1) or wire the enable input for the corresponding channel.
4. Enter the start velocity.
   - Rapid traverse (MODE_TYPE = 1)
   - Creep speed (MODE_TYPE = 0)
5. Set the control signal for traversing direction plus or minus (DIR_P = 1 or DIR_M = 1).
6. Call the ABS_CTRL (Page 7125) or ABS_CTRL_451 (Page 7129) instruction.

The following diagram shows an example for the "Jogging" mode.



Data used in the channel DB

Address	Name	Type	Initial value	Comment
15.2	DIR_M	BOOL	FALSE	1 = Minus direction
15.3	DIR_P	BOOL	FALSE	1 = Plus direction
15.7	DRV_EN	BOOL	FALSE	1 = Turn on drive enable
16.0	MODE_IN	BYTE	B#16#0	1 = jog
17.0	MODE_TYPE	BYTE	B#16#0	1 = Rapid traverse 0 = Creep speed
23.0	ST_ENBLD	BOOL	FALSE	1 = Start enabled
23.1	WORKING	BOOL	FALSE	1 = Positioning process is running
23.2	WAIT_EI	BOOL	FALSE	1 = Axis waiting for external enable
34.2	EI_OFF	BOOL	FALSE	1 = Do not evaluate enable input

Shutting down jogging

The "Jog" mode is shut down if

- You open the button used for jogging (DIR_M = 0 or DIR_P = 0),
- The FM 451 receives a STOP signal (STOP = 1)
- the process value with a synchronized linear axis reaches the limit of the operating range. The axis can only continue moving in the opposite direction.

After traversing is shut down, it can continue in any direction.

Canceling jogging

"Jog" mode is cancelled if

- The drive enable signal is deleted (DRV_EN = 0)
- A traversing range limit has been overrun for a linear axis

Monitoring functions

In the "Jogging" mode, the following monitoring functions are not active at the end of the positioning:

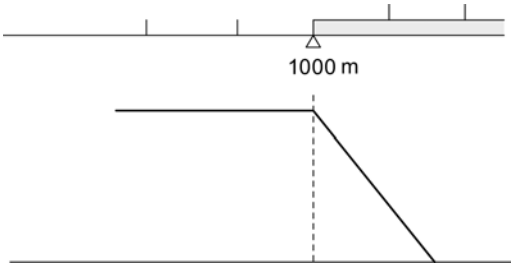
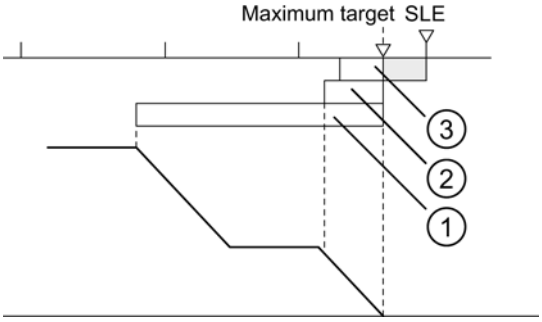
- Monitoring of standstill range
- Monitoring of target range
- Monitoring of standstill velocity

Jogging does not result in proper positioning for which the "Position reached" signal is set at the end.

Operating range limit for a linear axis

The limits for the "Jogging" mode differ between a synchronized and an unsynchronized axis.

The following table shows "Jogging" with synchronized and non-synchronized axes.

Axis is unsynchronized	Axis is synchronized
If, when jogging, the traversing range limit is overrun, • the process value display is no longer valid. • the positioning process is aborted.  The process value indicated is no longer valid	Jogging involves a positioning to targets that are removed from the software limit switches by a distance equal to the whole target range. The operating range limits are determined from • **SEE - ½ target range** for the end of the linear axis in the plus direction • **SEE + ½ target range** for the end of the linear axis in the plus direction If you do not open the button beforehand, the FM 451 shuts down at a target point that is positioned half the target range away from the respective software limit switch. All ranges that are necessary for proper shutdown are set by the FM 451 relative to this target point.  = Part of operating range in which a target position cannot be set ① = Changeover difference plus ② = Switch-off difference plus ③ = ½ target range

"Reference point approach" mode

Definition

The "Reference point approach" mode can be used to synchronize the axis based on a recurring external event.

Requirements

- An incremental encoder with zero mark.
- The axis parameters must be assigned:

Connections	Channel 1	Channel 2	Channel 3
Reference point switch	Digital input 1I0	Digital input 2I0	Digital input 3I0
	The reference point switch should be furnished such that the drive can safely brake from rapid traverse to creep speed in the range of the switch.		
Reversing switch	Digital input 1I1	Digital input 2I1	Digital input 3I1
	When assigning parameters, ensure that the start of the reference point approach is assigned in the direction of the reversing switch. Only in this way can you guarantee that the reference point switch will always be found.		
Enable input	Digital input 1I2	Digital input 2I2	Digital input 3I2

Sequence of events for "Reference point approach" mode

The keys for the positive and negative traversing direction must be installed for each channel.

1. Set the value for the reference point coordinate at the parameter DB REFPT.
2. Enter the type of "Reference point approach" in the parameter DB.
   The following options are available:

Start in direction ...	That leads to synchronization ...	
plus	the first zero mark after leaving the reference point switch in direction **plus**	REFPT_TYPE = 0
plus	the first zero mark after leaving the reference point switch in direction **minus**	REFPT_TYPE = 1
minus	the first zero mark after leaving the reference point switch in direction **plus**	REFPT_TYPE = 2
minus	the first zero mark after leaving the reference point switch in direction **minus**	REFPT_TYPE = 3

3. Enter the start velocity.
   - Rapid traverse (REFPT_SPD = 0)
   - Creep speed (REFPT_SPD = 1)
4. Write and enable the machine data.
5. Set the control signal for the "Reference point approach" mode (MODE_IN = 3).
6. Set the control signal for the drive enable (DRV_EN = 1).
7. Set the function switch for "Do not evaluate enable input" (EI_OFF = 1) or wire the enable input for the corresponding channel.
8. Set the control signal for traversing direction plus or minus or the start signal (DIR_P = 1, DIR_M = 1 or START = 1)
9. Call the ABS_CTRL (Page 7125) or ABS_CTRL_451 (Page 7129) instruction.

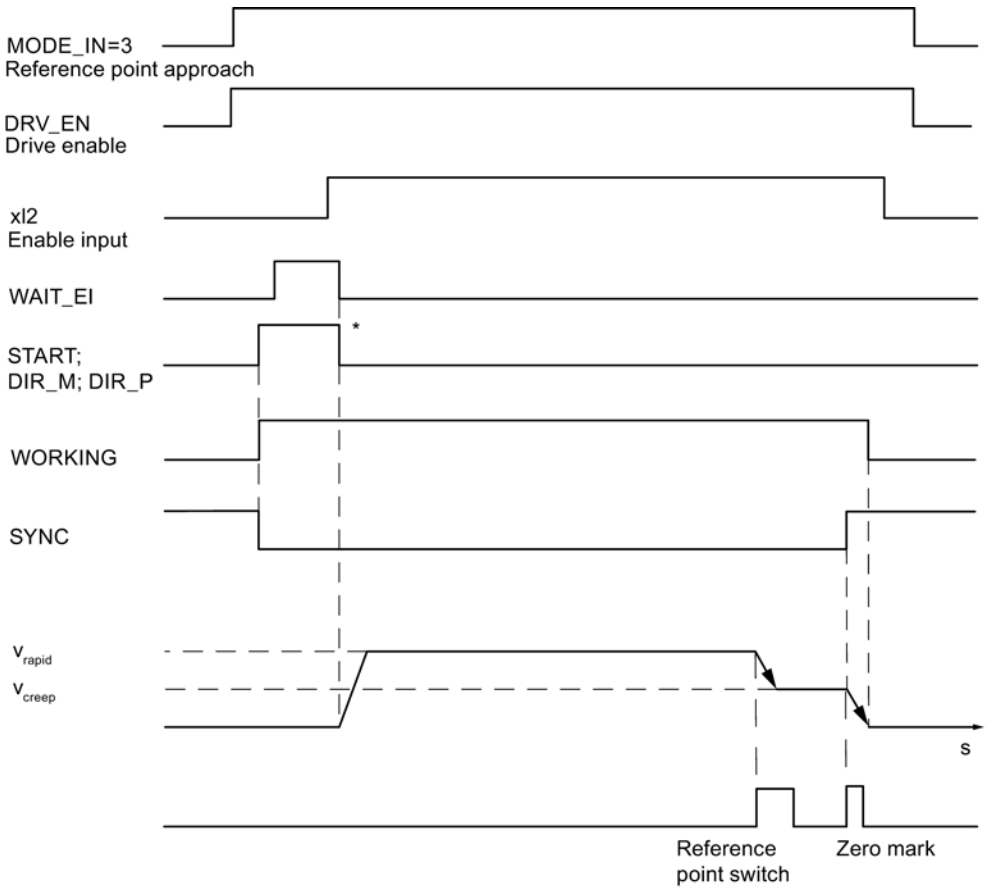
The following table shows the start commands for a reference point approach.

Start command	Job	Comment
DIR_P	The drive starts in the direction with positive values. It therefore moves in the direction of the traversing range end.	If a negative direction is entered in the machine data, the FM 451 signals an operator error. No reference point approach is executed.
DIR_M	The drive starts in the direction with negative values. It therefore moves in the direction of the traversing range start.	If a positive direction is entered in the machine data, the FM 451 signals an operator error. No reference point approach is executed.
START	The drive starts in the direction that is entered in the machine data.	

Note

The following applies to the rotary axis: Reproducibility of the reference point is only ensured if the ratio of the **End of rotary axis** value and the **Distance per encoder revolution** value is an integer.

The following figure shows an example for the "Reference point approach" mode.



* The start signals are reset by the ABS_CTRL (Page 7125) or ABS_CTRL_451 (Page 7129) instruction.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
15.0	START	BOOL	FALSE	1 = Start positioning process
15.2	DIR_M	BOOL	FALSE	1 = Minus direction
15.3	DIR_P	BOOL	FALSE	1 = Plus direction
15.7	DRV_EN	BOOL	FALSE	1 = Turn on drive enable
16.0	MODE_IN	BYTE	B#16#0	3 = reference point approach
23.0	ST_ENBLD	BOOL	FALSE	1 = Start enabled
23.1	WORKING	BOOL	FALSE	1 = Positioning process is running
23.2	WAIT_EI	BOOL	FALSE	1 = Axis waiting for external enable
25.0	SYNC	BOOL	FALSE	1 = axis is synchronized
34.2	EI_OFF	BOOL	FALSE	1 = Do not evaluate enable input

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
44.0	REFPT	DINT	L#0	Reference point coordinate
52.0	REFPT_TYPE	DINT	L#0	Type of reference point approach
99.0	REFPT_SPD	BOOL	TRUE	Start velocity for the reference point approach 0 = rapid traverse 1 = creep speed

Effects of the mode

- Synchronization is canceled when traversing starts.
- The actual position is set to the value of the reference point coordinate when the SYNC feedback signal is set.
- The operating range is established on the axis.
- The individual points within the operating range maintain their original value, but are located at different positions.

Canceling the reference point approach

The "Reference point approach" mode is cancelled if

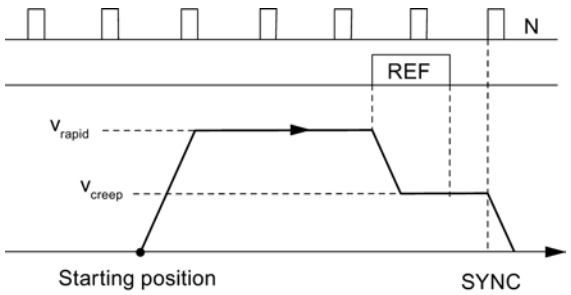
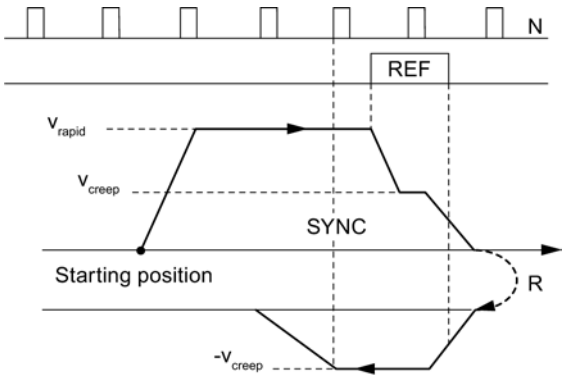
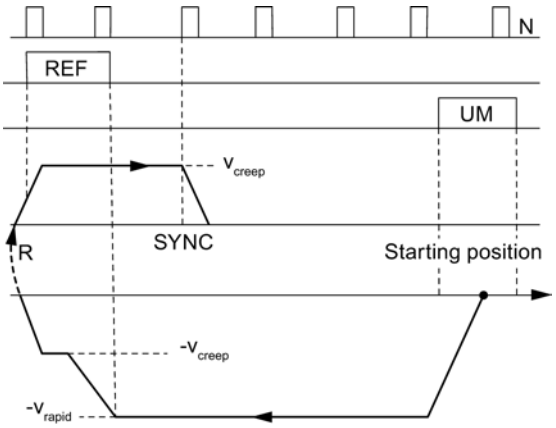
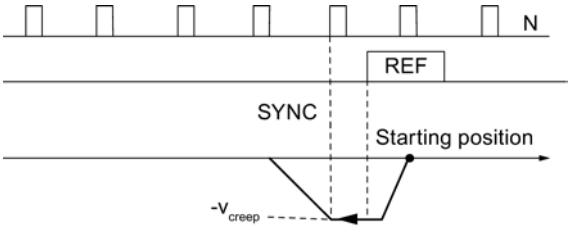
- The drive enable signal is deleted (DRV_EN = 0)
- A traversing range limit has been overrun for a linear axis

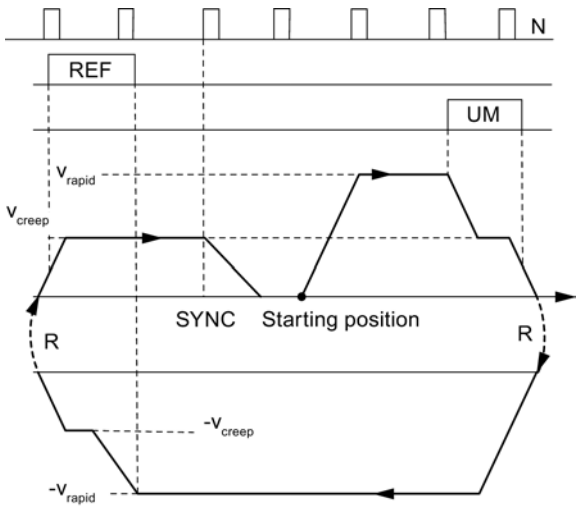
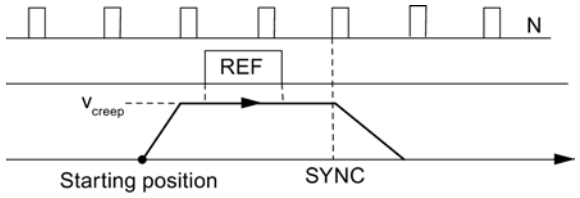
Reference point approach in dependence on the start position

In the case of a reference point approach, you must distinguish between different cases that are dependent on

- the location of the drive at the time a reference point approach is started,
- the assigned start direction,
- the assigned location of the zero mark relative to the reference point switch.

The following table shows the cases for REFPT_TYPE = 0 and REFPT_TYPE = 1. The figures apply correspondingly to REFPT_TYPE = 2 and REFPT_TYPE = 3.

Conditions of the reference point approach	Sequence of events for the reference point approach
Example of reference point approach (REFPT_TYPE = 0): • Start direction is plus. • The location of the zero mark of the reference point switch is assigned in the plus direction.	
Example of reference point approach (REFPT_TYPE = 1): • Start direction is plus. • The location of the zero mark of the reference point switch is assigned in the minus direction.	
Example of reference point approach (REFPT_TYPE = 0): • The start direction must be assigned in the plus direction. • The location of the zero mark of the reference point switch is assigned in the plus direction. • The reversing switch is more positive than the reference point switch.	
Example of reference point approach (REFPT_TYPE = 1): • Start direction is plus. • The location of the zero mark of the reference point switch is assigned in the minus direction. • The start position of the reference point approach is on the reference point switch.	

Conditions of the reference point approach	Sequence of events for the reference point approach
Example of reference point approach (REFPT_TYPE = 0): • Start direction is plus. • The location of the zero mark of the reference point switch is assigned in the plus direction. • The reversing switch is more positive than the reference point switch.	
Example of reference point approach (REFPT_TYPE=0): • Start direction is plus. • The location of the zero mark of the reference point switch is assigned in the plus direction. • Start velocity = creep speed	
R = Direction reversal REF = Reference point switch UM = Reversing switch N = Zero mark of the encoder SYNC = Synchronization has been achieved	

"Incremental approach" mode

Definition

With the "Incremental approach" mode, the FM 451 can move the drive.

- toward **absolute** targets,
- **relatively** by an incremental distance in a specified direction.

You specify the target or the relative distances as increment dimensions of the FM 451. You have the possibility of entering up to 100 incremental dimensions in a table that are valid both for the "Relative incremental approach" mode as well as for the "Absolute incremental approach" mode. Independently of the incremental dimension table, you can specify the distance using increment dimension 254 (Page 10623) and incremental dimension 255 (Page 10624)

The incremental dimension 252 (Page 10623) gives you the option of moving the drive initially in a specified direction without a target (continuous traversing). When an external event occurs (xI3), the programmed target is then positioned upon.

Requirements

- The axis parameters must be assigned.
- The axis must be synchronized.
- The increments must be available on the module.

Interpretation of the increments

The FM 451 interprets the specifications differently, depending on the incremental approach you select.

- Absolute incremental approach: The incremental dimensions are interpreted as absolute target position.
  With a rotary axis, the end of the axis may not be passed.
- Relative incremental approach: The incremental dimensions are interpreted as distance difference from the start position.
  With a rotary axis, the end of the axis may not be passed (multiple revolutions are permitted).

Note

Only positive increments are permissible for the "Relative incremental approach" mode.

The incremental dimensions result from the DIR_P or DIR_M direction specifications.

"Incremental approach" mode sequence with incremental dimension number 1 to 100

Step	Absolute incremental approach	Relative incremental approach
	Incremental dimension number 1 to 100	
1	Set the control signal for the "Absolute incremental approach" mode (MODE_IN = 5).	Set the control signal for the "Relative incremental approach" mode (MODE_IN = 4).
2	Enter the incremental dimensions in the tables (TRGL1; TRGL2).	
3	Write the incremental dimension table (TRGL1/2WR_EN = 1).	
4	Set the control signal for the drive enable (DRV_EN = 1).	
5	Set the function switch for "Do not evaluate enable input" (EI_OFF = 1) or wire the enable input for the corresponding channel.	
6	Enter the incremental dimension number (MODE_TYPE = 1 … 100).	

Step	Absolute incremental approach	Relative incremental approach
	Incremental dimension number 1 to 100	
7	Set the control signal: • Linear axis: – START; the direction is determined explicitly by the target and the current process value. • Rotary axis: – START; the target is approached by the shortest route. – DIR_P; start in plus direction – DIR_M; start in minus direction	Set the control signal: • Linear axis: – DIR_P; start in plus direction – DIR_M; start in minus direction • Rotary axis: – DIR_P; start in plus direction – DIR_M; start in minus direction
8	Call the ABS_CTRL (Page 7125) or ABS_CTRL_451 (Page 7129) instruction.	
Steps 2 and 3 are only required if no increments are given or the existing increments are to be changed.		

"Incremental approach" mode with incremental dimension number 252

Step	Absolute incremental approach	Relative incremental approach
	Incremental dimension number 252	
1	Set the control signal for the "Absolute incremental approach" mode (MODE_IN = 5).	Set the control signal for the "Relative incremental approach" mode (MODE_IN = 4).
2	Set the control signal for the drive enable (DRV_EN = 1).	
3	Set the function switch for "Do not evaluate enable input" (EI_OFF = 1) or wire the enable input for the corresponding channel.	
4	Enter the incremental dimension number (MODE_TYPE = 252).	
5	Enter the start velocity. • Rapid traverse (SPEED252 = 1) • Creep speed (SPEED252 = 0)	
6	Enter the incremental dimension for the incremental dimension number 252 (TRG252_254).	
7	Set the trigger bit for writing the incremental dimension (TRG252_254_EN = 1).	
8	Set the control signal: • Linear axis / rotary axis: – DIR_P; start in plus direction – DIR_M; start in minus direction	
9	Call the ABS_CTRL (Page 7125) or ABS_CTRL_451 (Page 7129) instruction.	
10	When an external event occurs (digital input xI3), the positioning to the assigned target (increment) is performed.	
The actions listed in steps 6 and 7 can also only be performed during traversing. In this case, the FM 451 module controls a target up to the incremental dimension specification. This target is located a distance in front of the respective software limit switch equal to half the target range.		

Sequence of events for the "incremental approach" mode with increment number 254

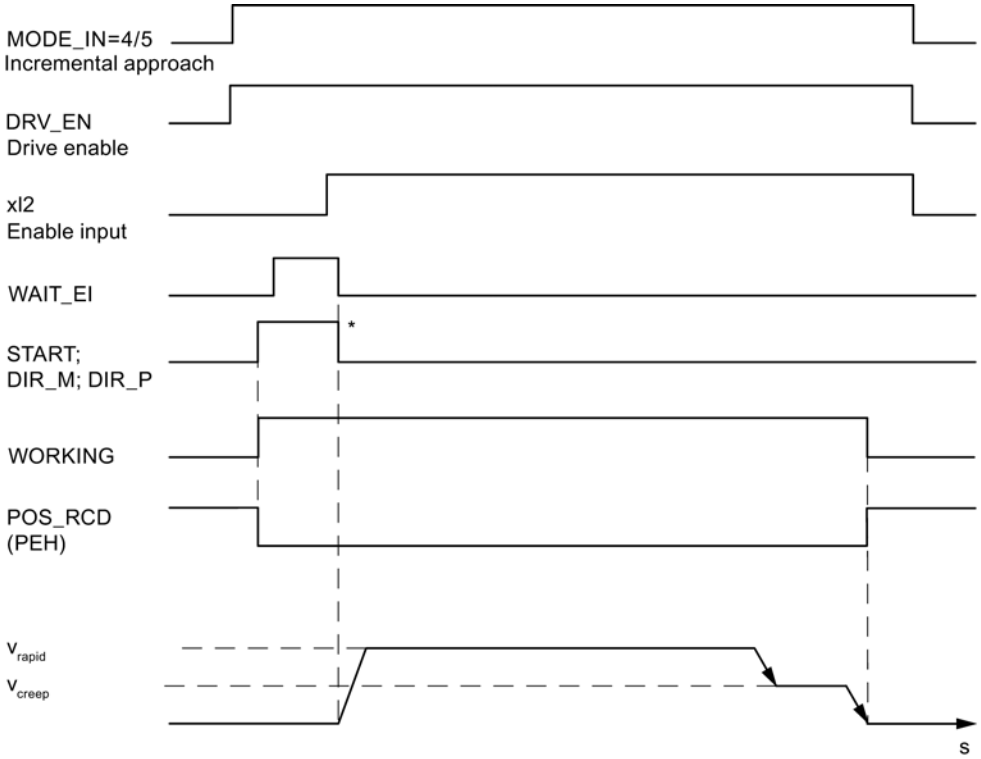
Step	Absolute incremental approach	Relative incremental approach
	Increment number 254	
1	Set the control signal for the "Absolute incremental approach" mode (MODE_IN = 5).	Set the control signal for the "Relative incremental approach" mode (MODE_IN = 4).
2	Set the control signal for the drive enable (DRV_EN = 1).	
3	Set the function switch for "Do not evaluate enable input" (EI_OFF = 1) or wire the enable input for the corresponding channel.	
4	Enter the incremental dimension number (MODE_TYPE = 254).	
5	Enter the incremental dimension for the incremental dimension number 254 (TRG252_254).	
6	Set the trigger bit for writing the incremental dimension (TRG252_254_EN = 1).	
7	Set the control signal: • Linear axis: – START; the direction is determined explicitly by the target and the current process value. • Rotary axis: – START; the target is approached by the shortest route. – DIR_P; start in plus direction – DIR_M; start in minus direction	Set the control signal: • Linear axis: – DIR_P; start in plus direction – DIR_M; start in minus direction • Rotary axis: – DIR_P; start in plus direction – DIR_M; start in minus direction
8	Call the ABS_CTRL (Page 7125) or ABS_CTRL_451 (Page 7129) instruction.	

Sequence of events for the "incremental approach" mode with increment number 255

Step	Absolute incremental approach	Relative incremental approach
	Increment number 255	
1	Set the control signal for the "Absolute incremental approach" mode (MODE_IN = 5).	Set the control signal for the "Relative incremental approach" mode (MODE_IN = 4).
2	Set the control signal for the drive enable (DRV_EN = 1).	
3	Set the function switch for "Do not evaluate enable input" (EI_OFF = 1) or wire the enable input for the corresponding channel.	
4	Enter the incremental dimension number (MODE_TYPE = 255).	
5	Enter the incremental dimension for the incremental dimension number 255 (TRG255).	
6	Enter the value for the switch-over difference of the incremental dimension number 255 (CHGDIF255).	
7	Enter the value for the switch-over difference of the incremental dimension number 255 (CUTDIF255).	
8	Set the trigger bit for writing the incremental dimension, switch-off difference, and switch-over difference (TRG255_EN = 1).	

Step	Absolute incremental approach	Relative incremental approach
	Increment number 255	
9	Set the control signal: • Linear axis: – START; the direction is determined explicitly by the target and the current process value. • Rotary axis: – START; the target is approached by the shortest route. – DIR_P; start in plus direction – DIR_M; start in minus direction	Set the control signal: • Linear axis: – DIR_P; start in plus direction – DIR_M; start in minus direction • Rotary axis: – DIR_P; start in plus direction – DIR_M; start in minus direction
10	Call the ABS_CTRL (Page 7125) or ABS_CTRL_451 (Page 7129) instruction.	

The following figure shows an example for the "Incremental approach" mode.



* The start signals are reset by the ABS_CTRL (Page 7125) or ABS_CTRL_451 (Page 7129) instruction.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
15.0	START	BOOL	FALSE	1 = start positioning
15.2	DIR_M	BOOL	FALSE	1 = Minus direction
15.3	DIR_P	BOOL	FALSE	1 = Plus direction
15.6	SPEED252	BOOL	FALSE	Start velocity for incremental approach with increment number 252 • 0 = Creep speed • 1 = Rapid traverse
15.7	DRV_EN	BOOL	FALSE	1 = Turn on drive enable
16.0	MODE_IN	BYTE	B#16#0	4 = relative incremental approach 5 = absolute incremental approach
17.0	MODE_TYPE	BYTE	B#16#0	Incremental dimension 1 … 100, 252, 254 or 255
23.0	ST_ENBLD	BOOL	FALSE	1 = Start enabled
23.1	WORKING	BOOL	FALSE	1 = positioning is active
23.2	WAIT_EI	BOOL	FALSE	1 = Axis waiting for external enable
25.7	POS_RCD	BOOL	FALSE	1 = in position
34.2	EI_OFF	BOOL	FALSE	1 = Do not evaluate enable input
36.2	TRG252_254_EN	BOOL	FALSE	1 = Write incremental dimension for increment dimension number 252 or 254
36.3	TRG255_EN	BOOL	FALSE	1 = write increment for increment number 255
35.4	TRGL1WR_EN	BOOL	FALSE	1 = Write incremental dimension table 1 (incremental dimension number 1...50)
35.5	TRGL2WR_EN	BOOL	FALSE	1 = Write incremental dimension table 2 (incremental dimension number 51...100)
96.0	TRG252_254	DINT	L#0	Write incremental dimension for incremental dimension number 252 or 254
100.0	TRG255	DINT	L#0	Increment for increment number 255
104.0	CHGDIF_255	DINT	L#0	Changeover difference for increment number 255
108.0	CUTDIF_255	DINT	L#0	Switch-off difference for increment number 255

Data used in the parameter DB

Address	Name	Type	Initial value	Comment	
100.0	CHGDIF_P	DINT	L#5000	Changeover difference plus	
104.0	CHGDIF_M	DINT	L#5000	Changeover difference minus	
108.0	CUTDIF_P	DINT	L#2000	Switch-off difference plus	
112.0	CUTDIF_M	DINT	L#2000	Switch-off difference minus	
120.0	TRGL1.TRG[1]	DINT	L#0	Increment number 1	Increment table 1
. . .	. . .	. . .	. . .	. . .	
316.0	TRGL1.TRG[50]	DINT	L#0	Increment number 50	

Address	Name	Type	Initial value	Comment	
320.0	TRGL2.TRG[51]	DINT	L#0	Increment number 51	Increment table 2
. . .	. . .	. . .	. . .	. . .	
516.0	TRGL2.TRG[100]	DINT	L#0	Increment number 100	

Distance-to-go

The distance-to-go is the difference between the target (increment) and process value. The distance-to-go is

- positive, if the target has not yet been reached
- negative, if the target has already been passed.

In the case of a rotary axis, the displayed distance-to-go cannot be used.

Shutting down the Incremental approach

The "Incremental approach" mode is shut down when the FM 451 receives a STOP signal (STOP = 1).

After traversing is shut down, the distance-to-go remains.

The distance-to-go for "Relative incremental approach" and "Absolute incremental approach with incremental dimension number 252" can be traversed to the end if

- the mode is unchanged, and
- the increment number is unchanged, and
- the direction is unchanged, and
- the remaining distance-to-go is greater than the assigned switch-off difference.

You traverse the residual distance by starting "Relative incremental approach" or "Absolute incremental approach with incremental dimension number 252" again without making any changes.

Canceling the incremental approach

The "Incremental approach" mode is cancelled when the "Drive enable" signal is deleted (DRV_EN = 0).

Delete distance-to-go

With the job "Delete distance-to-go" you can delete any awaiting distance-to-go.

With "Incremental approach with incremental dimension number 252", traversing restarts with continuous traversing after "Delete residual distance".

You also delete the awaiting distance-to-go by calling a different mode or by starting the mode in the other direction.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
35.2	DELDIST_EN	BOOL	FALSE	1 = delete distance-to-go

The "Set actual value" / Set actual value on-the-fly / "Undo Set Actual Value" job

Definition

You use the "Set actual value" or "Set actual value on-the-fly" jobs in order to assign a new coordinate to the current encoder value. The operating range is projected on a different area of the axis.

You determine the shift in the operating range with (ACT_{new}- $ACT_{current}$).

- $ACTUAL_{new}$ is the default value
- $ACTUAL_{current}$ is the actual value at the time of the execution

Requirements

- The axis parameters must be assigned.
- The axis must be synchronized.

For "Set actual value", the positioning process must be finished.

Job sequence

1. Enter the coordinate AVAL or FVAL for the actual value (ACT_{new}).
   - Linear axis:
     You must select the specified actual value so that the software limit switches are still within the permissible traversing range after the job is called.
     The amount of the shift resulting from (ACT_{new} - $ACT_{current}$) must be less than or equal to the amount of the permissible traversing range (max. 100 m or 1000 m).
   - Rotary axis:
     Rule for the defined actual value:
     0 ≤ actual value < end of rotary axis
2. Set the relevant trigger bit (AVAL_EN = 1 or FVAL_EN = 1).

"Set actual value" is executed immediately.

"Set actual value on-the-fly" is executed at the next positive edge at digital input xI3.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
35.7	AVAL_EN	BOOL	FALSE	1 = set actual value
36.0	FVAL_EN	BOOL	FALSE	1 = Set actual value on-the-fly

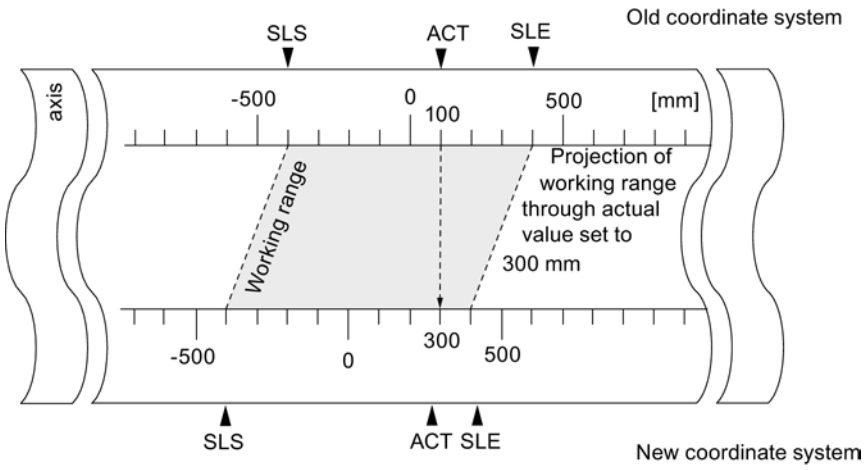
Address	Name	Type	Initial value	Comment
84.0	AVAL	DINT	L#0	Coordinate for "Set actual value"
88.0	FVAL	DINT	L#0	Coordinate for "Set actual value on-the-fly"

Effects of the job

In the example "set actual value " to 300 mm you can see how this job projects the operating range onto a specified position of the axis. This produces the following effects:

- The actual position is set on the value of the actual value coordinate.
- The operating range is shifted on the axis.
- The individual points (e.g., software limit switch end) within the operating range maintain their original value, but have new positions.

The following table shows the shift of the axis operating range by "set reference point".

Set actual value	SLS	ACTUAL	SLE
	-400	100	400
	-400	300	400

Withdrawing the job ("Undo set actual value")

The "Undo set actual value" job can be used to reset the **most recent** operating range shift caused by "Set actual value".

Once "Set actual value on-the-fly" has been initiated, it can no longer be deleted before execution by a positive edge at digital input xI3. However, it can be overwritten by a new "Set actual value on-the-fly". These jobs are reset the next time the module starts up.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
35.3	AVALREM_EN	BOOL	FALSE	1 = cancel set actual value

"Zero offset" job

Definition

The "zero offset" job lets you shift the zero point in the coordinate system by a defined value.

The sign determines the offset direction in the coordinate system.

Determining a new coordinate

All values of the shifted coordinate system can be calculated according to the following formula:

$\mathbf{Coordinate_{new} = Coordinate_{old} - (ZPO_{new} - ZPO_{old})}$

$\mathbf{ZPO_{old}}$ designates an existing zero offset.

If no zero offset was active prior to the call, set the value 0 for $\mathbf{ZPO_{old}}$.

You can use this to determine the coordinate values for the software limit switches, for example.

Requirements

- The positioning must be ended.
- The axis parameters must be assigned.

Job sequence

1. Enter the value ZOFF for the zero offset.
   - Linear axis:
     The zero offset must be selected so that the software limit switches remain within the permissible numerical range after the job is called.
   - Rotary axis:
     Rule for the zero offset:
     value of zero offset ≤ end of rotary axis
2. Set the relevant trigger bit (ZOFF_EN = 1).

Data used in the channel DB

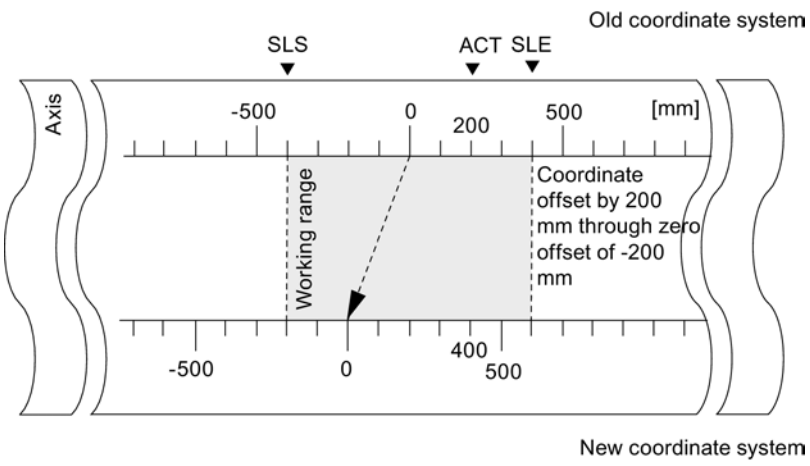
Address	Name	Type	Initial value	Comment
36.1	ZOFF_EN	BOOL	FALSE	1 = Set zero offset
80.0	ZOFF	DINT	L#0	Zero offset

Effects of the job on a linear axis

Based on the example of a zero offset of -200 mm, you can see that this job shifts the coordinate system positively. This produces the following effects:

- The operating range is **not** physically shifted.
- The individual points (such as the software limit switches) are assigned new coordinate values.

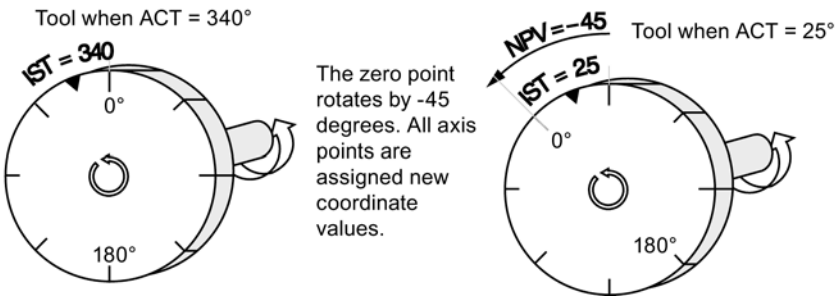
The following table shows the offset of the coordinate system by "Zero offset".

Zero offset	SLS	ACTUAL	SLE
	-400	200	400
	-200	400	600

Effects of the job on a rotary axis

Based on the example of a zero offset by -45°, you can see how this job **rotates** the coordinate system:

The following figure shows the revolution of the coordinate system by a zero offset.



Including a **ZPO_{old} = 0**, the result is a new value of 385°.

Because the actual value starts over at 0 at the end of the rotary axis with a positive direction of rotation, the actual value of 25° is calculated from:

$Coordinate_{new}$ = $Coordinate_{old}$ - (ZPO_{new} - ZPO_{old}) - end of rotary axis

Loss of synchronization

If synchronization is lost due to an error or is reset due to "reference point approach", a zero offset **remains** active.

Canceling the job

A zero offset of 0 mm resets any existing zero offset.

"Set reference point" job

Definition

You use the "set reference point" job to synchronize the axes. The job shifts the operating range. All the shifts that were created with "Set actual value" are retained.

The setting projects the operating range on the axis. Therefore, by entering different values, the operating range can be moved to any position within the physical area of the axis.

Prerequisites

- The positioning must have ended.
- The axis parameters must be assigned.

Job sequence

1. Enter the value REFPT for the reference point coordinate.
   - Linear axis:

     The reference point coordinates must not lie outside the software limit switches. This also applies to the reference point coordinate in a shifted coordinate system.
   - Rotary axis:
     Rule for the reference point coordinate:
     0 ≤ reference point coordinate < end of rotary axis
2. Set the relevant trigger bit (REFPT_EN = 1).

Data used in the channel DB

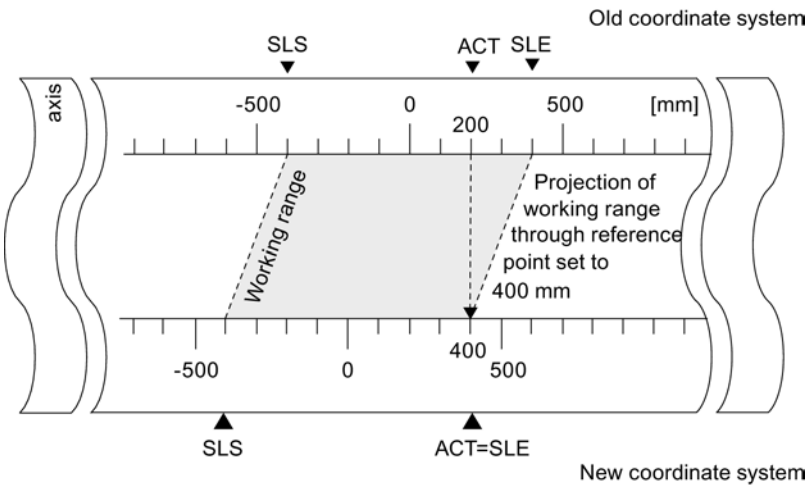
Address	Name	Type	Initial value	Comment
25.0	SYNC	BOOL	FALSE	1 = axis is synchronized
35.6	REFPT_EN	BOOL	FALSE	1 = set reference point
92.0	REFPT	DINT	L#0	Reference point coordinate

Effects of the job

In the example "set reference point" to 400 mm you can see how this job projects the operating range onto a specific, physical position of the axis. This has the following results:

- The actual position is set on the value of the reference point coordinate.
- The operating range is shifted on the axis.
- The individual points (e.g. software limit switch end) maintain their original value, but are located at new positions.
- The SYNC bit in the checkback signals is set.

The following table shows how "Set reference point" shifts the operating range on the axis.

Set reference point	SLS	ACTUAL	SLE
	-400	200	400
	-400	400	400

Special features of the absolute encoder

This job is necessary for an absolute encoder adjustment (Page 10610).

"Length measurement" / "Edge detection" function switches

Definition

The "length measurement" and "edge detection" let you determine the length of a workpiece.

The "length measurement" or "edge detection" function switch is active and remains active until you disable it or select the other measuring method. If you select both measuring methods in parallel, ABS_CTRL (Page 7125) or ABS_CTRL_451 (Page 7129) enables length measurement.

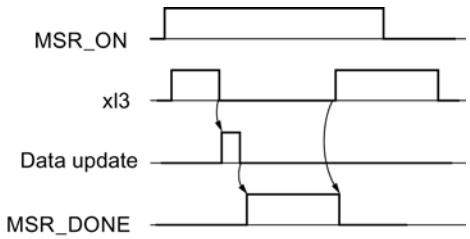
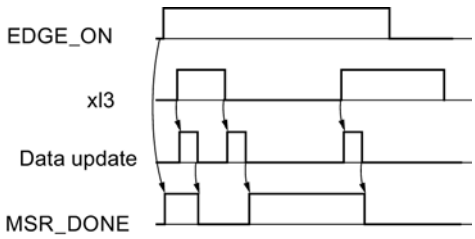
Requirements

- The positioning must be ended.
- The axis parameters must be assigned.
- The axis must be synchronized.
- A **bounce-free** switch must be connected to input xI3.
- If these function switches are selected during a positioning, "Length measurement" and "Edge detection" can only be executed with the next positioning.

Measuring sequence

Depending on the type of measurement, the FM 451 updates the data on the module at different times. FM 451 reports each update at a parameter on the feedback interface.

The following table shows the measuring sequence.

Sequence	Length measurement	Edge detection
		
1	Set the function switch for "Length measurement" (MSR_ON = 1).	Enter a value EDGEDIST for the minimum edge-to-edge distance in the parameter DB. Write and enable machine data.
2	Start a positioning.	Set the "Edge detection" function switch (EDGE_ON = 1). This sets the MSR_DONE parameter.
3	A positive edge at input xI3 starts the length measurement.	Start a positioning.
4	A negative edge at input xI3 stops the current measurement. The FM 451 updates the initial value, end value, and length data.	A positive edge at input xI3 starts the measurement. After the "minimum edge distance" is traversed, the measurement is valid. This is signaled by MSR_DONE = 0. The results of the measurement are updated and can be read out, and the start value of the measurement is entered; the end value and length assume the value -1.
5	When parameter MSR_DONE is set, FM 451 reports the data update. The parameter indicates completion of the measurement. The results of the measurement can be read out.	After the update, FM 451 reports the change by resetting the MSR_DONE parameter.
6	The start of the next measurement at a positive edge at xI3 resets the MSR_DONE parameter.	A negative edge at input xI3 stops the current measurement. The FM 451 updates the data for the end measurement and length values.

Sequence	Length measurement	Edge detection
7	-	After the update, FM 451 reports the change by setting the MSR_DONE parameter. The results of the measurement can be read out.
8	-	The start of the next measurement at a positive edge at xI3 resets the MSR_DONE parameter.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
25.1	MSR_DONE	BOOL	FALSE	1 = Length measurement/edge detection completed
34.3	EDGE_ON	BOOL	FALSE	1 = Edge detection ON
34.4	MSR_ON	BOOL	FALSE	1 = Length measurement ON
35.0	MDWR_EN	BOOL	FALSE	1 = Write machine data
35.1	MD_EN	BOOL	FALSE	1 = Enable machine data
37.0	MSRRD_EN	BOOL	FALSE	1 = Read measured values
136.0	BEG_VAL	DINT	L#0	Start value of length measurement/edge detection
140.0	END_VAL	DINT	L#0	End value of length measurement/edge detection
144.0	LEN_VAL	DINT	L#0	Length

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
4.0	EDGEDIST	DINT	L#0	Minimum edge distance for edge detection
The measurement will be discarded if the measuring operation ends within this range (BEG_VAL, END_VAL and LEN_VAL = -1).				

Boundary conditions for a length measurement

- The CPU program requires an interval of sufficient length between the off and on edges at input xI3 in order to correctly evaluate the result of the measurement before a new measurement starts.
- The minimum interval between the positive and negative edges at input xI3 and between the negative edge and the next positive edge at input xI3 must be greater than 8 ms.

Faulty measurement

The FM 451 returns the length value -1 in the event of faulty length measurement / edge detection. A length measurement is faulty if:

- The length measured for a rotary axis is greater than 2^{31}.
- the on and off edges are detected simultaneously by the FM 451 (for example, due to switch bounce).

"Loop traverse" function switch

Definition

With "loop approach" you specify the direction in which a target is approached with a frictional connection. You can use loop approach if a frictional connection between the motor and axis can only be ensured in one direction.

A target that is approached in the direction opposite the specified direction is initially overrun. Then, the module performs a direction reversal and approaches the target in the specified direction.

Requirements

- The axis parameters must be assigned.
- The axis must be synchronized.
- In the case of a loop approach in the opposite direction to that of the traversing direction of the target the maximum target position is:
  - traversing direction plus
    Target < SLE - ½ Target range - Switch-off difference plus - Changeover difference minus
  - traversing direction minus
    Target < SLE - ½ Target range + Switch-off difference minus + Changeover difference minus
- A loop approach will not be carried out if the target is being approached in the direction of the loop approach. In this case, an incremental approach without an about-turn in direction will be carried out.
- The sequence of the "incremental approach" (Page 10573) operating mode must be known.

Sequence of events for the loop approach

1. Set the control signal for the "absolute/relative incremental approach" operating mode (MODE_IN = 5/4).
2. Set the control signal for the drive enable (DRV_EN = 1).
3. Set the "Enable input" (EI_OFF = 1) function switch or wire the enable input for the corresponding channel.
4. Enter the incremental dimension number (MODE_TYPE = 1 … 100, 254, 255).
5. Set the "Loop traverse" function switch (PLOOP_ON / MLOOP_ON = 1).
6. Start the incremental approach.

Data used in the channel DB

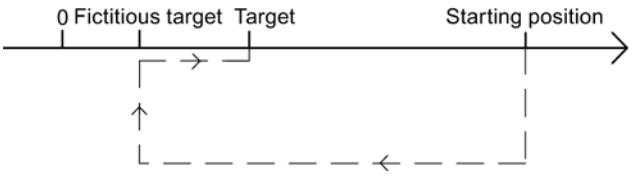
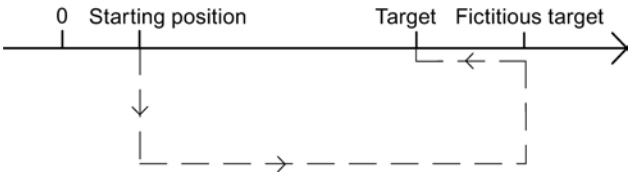
Address	Name	Type	Initial value	Comment
15.0	START	BOOL	FALSE	1 = start positioning
15.2	DIR_M	BOOL	FALSE	1 = minus direction
15.3	DIR_P	BOOL	FALSE	1 = plus direction
15.7	DRV_EN	BOOL	FALSE	1 = switch on drive enable
16.0	MODE_IN	BYTE	B#16#0	4/5 = relative / absolute incremental approach
17.0	MODE_TYPE	BYTE	B#16#0	Incremental dimension number 1 … 100, 254 or 255
34.0	PLOOP_ON	BOOL	FALSE	1 = loop approach in plus direction
34.1	MLOOP_ON	BOOL	FALSE	1 = loop approach in minus direction
34.2	EI_OFF	BOOL	FALSE	1 = do not evaluate enable input

Fictitious target

If you start a positioning to a target that is located in the direction opposite to that of the assigned loop traverse, the module determines a fictitious target for this target. The module performs a direction reversal at the fictitious target and then approaches the target in the proper direction.

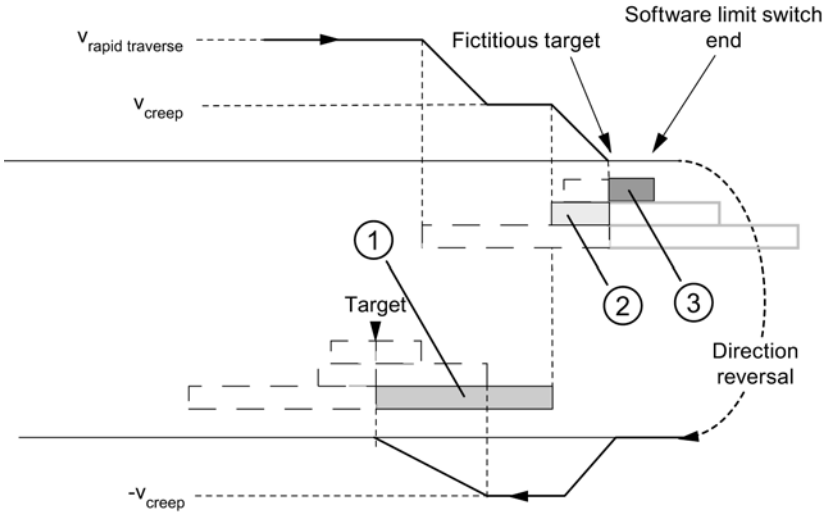
This fictitious target **must** be located by at least half the target range before the respective software limit switch.

The distance of the fictitious target from the parameterized target is established depending on the direction: The following table shows the calculation of the location of the fictitious target with a loop traverse.

Guidelines	Location of fictitious target
Parameter assignment: Loop + (frictional connection plus) and approach in minus direction.	The fictitious target (**$target_f$**) has the value: **$target_f$ = target - switch-off difference minus - changeover difference plus** 
Parameter assignment: Loop - (frictional connection minus) and approach in plus direction.	The fictitious target (**$target_f$**) has the value: **$target_f$ = target + switch-off difference plus + changeover difference minus** 

Example

Using a positioning with loop approach minus to a maximum target we can illustrate the location of the fictitious target.



① Changeover difference negative
② Switch-off difference positive
③ ½ target range

"Enable input" function switch

Definition

The enable input is an external input with which positioning can be enabled by an external event.

Evaluate enable input (EI_OFF = 0)

The relevant enable input (xI2) must be wired for the channel.

You have the option to prepare for the start of a positioning. You start the positioning independent of the program sequence of your user program by applying a "1" signal at the enable input.

You have the following options for evaluating the enable inputs:

- level-controlled (EI_TYPE = 0)
  The traverse begins when you set a "1" signal on the enable input and is shut down when you set a "0" signal on the enable input.
- edge-controlled (EI_TYPE = 1)
  Traversing begins with detection of a rising edge at the enable input. The additional signal characteristic at this input no longer affects the course of the motion that was started.

Do not evaluate enable input (EI_OFF = 1)

If you disable the evaluation of the enable input, an operating mode starts immediately after the start signal is detected. In this case, you cannot prepare an operating mode and start it at a future specified time.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
34.2	EI_OFF	BOOL	FALSE	1 = Do not evaluate enable input

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
99.1	EI_TYPE	BOOL	FALSE	0 = Level-controlled 1 = Edge-controlled

"Read position data" job

Definition

With the "read position data" job you can read the increment, distance-to-go and speed at the current time.

Job sequence

1. Set the trigger bit at the DB channel (ACTSPD_EN = 1).
2. The data are then stored in the channel DB.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
37.1	ACTSPD_EN	BOOL	FALSE	1 = read position data
112.0	ACTSPD	DINT	L#0	Current speed
116.0	DIST_TO_GO	DINT	L#0	Distance-to-go
120.0	ACT_TRG	DINT	L#0	Current increment

"Read encoder data" job

Definition

With the "read encoder data" job you can read the current data of the encoder as well as the value for the absolute encoder adjustment.

Requirements

You can read the value for the absolute encoder adjustment (Page 10610) after you have carried out the "set reference point" (Page 10584) job.

Job sequence

1. Set the trigger bit at the DB channel (ENCVAL_EN = 1).
2. The data is stored in the channel DB.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
37.2	ENCVAL_EN	BOOL	FALSE	1 = read encoder values
124.0	ENCVAL	DINT	L#0	Actual encoder value (internal representation)

Address	Name	Type	Initial value	Comment
128.0	ZEROVAL	DINT	L#0	Last zero mark value (internal representation)
132.0	ENC_ADJ	DINT	L#0	Absolute encoder adjustment

Checkback signals for the positioning

Definition

The "checkback signals for the positioning" inform you of the current status of the positioning.

Sequence

The data is stored in the channel DB each time the ABS_CTRL (Page 7125) or ABS_CTRL_451 (Page 7129) instruction is called.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
23.0	ST_ENBLD	BOOL	FALSE	1 = Start enabled
23.1	WORKING	BOOL	FALSE	1 = positioning is active
23.2	WAIT_EI	BOOL	FALSE	1 = Axis waiting for external enable
23.4	SPEED_OUT	BOOL	FALSE	0 = Creep speed 1 = Rapid traverse
23.5	ZSPEED	BOOL	FALSE	1 = axis is located in the standstill range
23.6	CUTOFF	BOOL	FALSE	1 = axis is located in the switch-off range
23.7	CHGOVER	BOOL	FALSE	1 = axis is located in the changeover range
24.0	MODE_OUT	BYTE	B#16#0	Active operating mode
25.2	GO_M	BOOL	FALSE	1 = axis moves in minus direction
25.3	GO_P	BOOL	FALSE	1 = axis moves in plus direction
25.5	FVAL_DONE	BOOL	FALSE	1 = Set process value on-the-fly completed
25.7	POS_RCD	BOOL	FALSE	1 = in position
26.0	ACT_POS	DINT	L#0	Current process value (current position of the axis)

Querying checkback signals for diagnostics

Definition

The "checkback signals for the diagnostics" informs you of occurring diagnostic events.

Sequence

1. If the module enters a new event in the diagnostic buffer, it sets the DIAG bit in all the channels in the feedback interface. Error (Page 10637) events are logged in the diagnostics buffer. The DIAG bit is also set when the diagnostic buffer is deleted.
2. If it is not possible to call an operating mode or control an active operating mode, or if this has been carried out incorrectly, the module sets an OT_ERR operator error in the feedback interface. The cause of error is then entered in the diagnostic buffer. While the operator error exists, you can neither start a new operating mode nor continue with the halted one. You acknowledge an existing operating error with OT_ERR_A = 1.
3. The module sets the DATA_ERR bit in the feedback interface when it detects incorrect data in a write job. The cause of error is then entered in the diagnostic buffer.
4. The checkback signals are stored in the channel DB.
5. When the diagnostic buffer is read, the module sets the DIAG bit back to 0 in all channels.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
22.2	DIAG	BOOL	FALSE	1 = delete diagnostic buffer
22.3	OT_ERR	BOOL	FALSE	1 = operator error
22.4	DATA_ERR	BOOL	FALSE	1 = data error

Configuring FM 451

Parameter assignment FM 451

Specifying channels

Selecting system of units

Selecting system of units

For the input and output of the data, you can select from the following systems of units:

- Millimeter (default setting)
- Inches
- Degrees

Note

If you change the system of units in the "Specify channels" screen, the values will be converted into the new system. This can cause round-off errors.

If you change the system of units via the machine date, the values are not automatically converted.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
8.0	UNITS	DINT	L#1	System of units Valid values: • 1 = 10^{-3} mm • 2 = 10^{-4} inch • 3 = 10^{-4} degrees • 4 = 10^{-2} degrees • 6 = 10^{-3} degrees

Standard system of units in the online help

To specify the minimum and maximum values we use the mm system of units. To define the limits in other systems of units, convert the values as shown below:

To convert from		calculate
mm → inch		Limit value(inches) = limit value (mm) × 0.1 [1]
mm → degrees	10^{-4} (4 post-decimal places)	Limit value (degrees) = limit value (mm) x 0.1
	10^{-3} (3 post-decimal places)	Limit value (degrees) = limit value (mm) x 1
	10^{-2} (2 post-decimal places)	Limit value (degrees) = limit value (mm) x 10
inches → degrees		Limit (degrees) = limit (inches) .

[1] When set to the maximum value, the number of decimal places affects the number of digits before the decimal point. In the inch measurement system, 4 decimal places are used; you can therefore enter a maximum of 100,000.0000 inches. The millimeter measurement system uses 3 decimal places; a maximum of 1,000,000.000 mm can be entered.

Connection between increments and system of units

The encoder signals of a connected encoder will be evaluated by the module and converted into the current system of units. The resolution (Page 10616) is used for the conversion.

If the positioning module

- has counted 10 increments and
- a resolution of 100 µm per increment is assigned in the encoder data,

this means that the axis will be moved by a distance of 1 mm.

Copying channels

Copying channels

You can copy the parameters of one channel to another.

1. Select the channel whose parameters are to be copied (source channel).
2. Select the channel to which the parameters are to be copied (target channel).
3. Press the "Copy" button.

Channel n drive

Selecting control mode

Overview of control modes

Overview

The control mode specifies how a motor is operated at the four digital outputs (Q) of a channel by means of the power module. There are 4 control modes per channel for the positioning module:

- Control mode 1 (Page 10597)
- Control mode 2 (Page 10598)
- Control mode 3 (Page 10598)
- Control mode 4 (Page 10599)

The following table shows the states of the 4 digital outputs for each control mode, x stands for channel 1, 2 and 3.

	Rapid traverse		Creep speed		PR Position reached stop
	Direction +	Direction -	Direction +	Direction -	
Control mode 1					
xQ0	1	1	0	0	-
xQ1	0	0	1	1	-
xQ2	1	0	1	0	-
xQ3	0	1	0	1	-
Control mode 2					
xQ0	1	1	0	0	0
xQ1	0	0	0	0	1
xQ2	1	0	1	0	0
xQ3	0	1	0	1	0
Control mode 3					
xQ0	1	1	0	0	-

	Rapid traverse		Creep speed		PR Position reached stop
	Direction +	Direction -	Direction +	Direction -	
xQ1	1	1	1	1	-
xQ2	1	0	1	0	-
xQ3	0	1	0	1	-
Control mode 4					
xQ0	1	0	0	0	-
xQ1	1	0	1	0	-
xQ2	0	1	0	0	-
xQ3	0	1	0	1	-

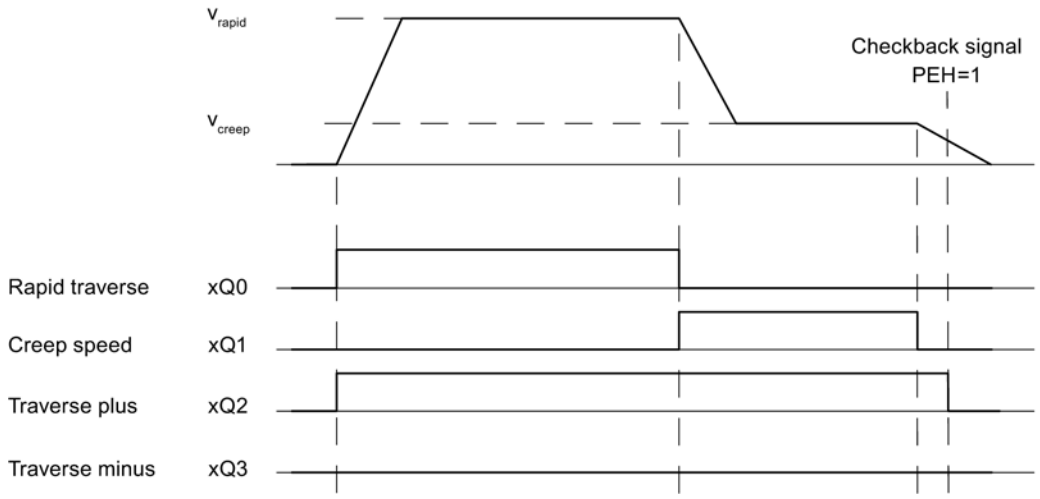
Control mode 1

Control mode 1

The following settings apply for control mode 1 (x stands for channel n):

- xQ0: Rapid traverse
- xQ1 Creep speed
- xQ2 Travel plus
- xQ3 Travel minus

The following image shows the function of the digital outputs with control mode 1.



Data used in the parameter DB

Address	Name	Type	Initial value	Comment
92.0	CTRL_TYPE	DINT	L#1	1 = Control mode 1

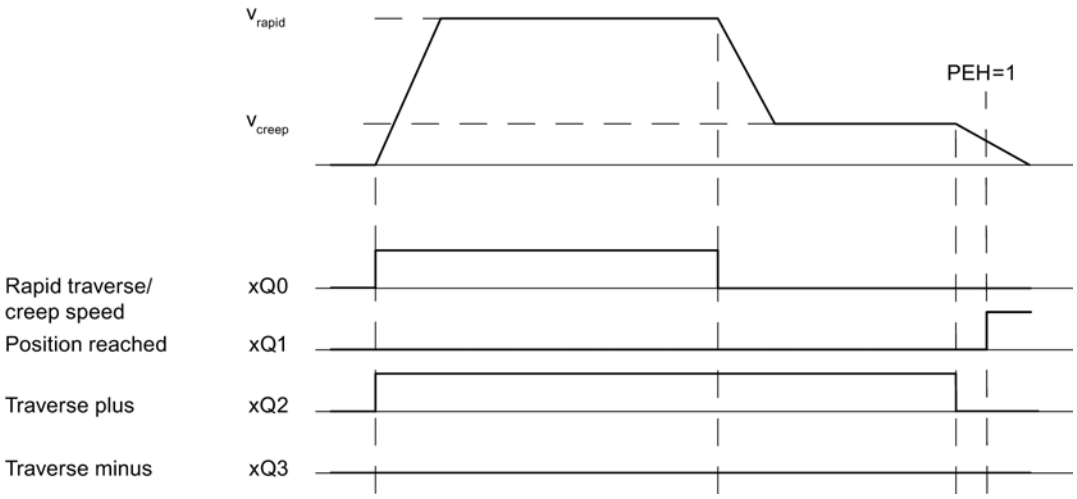
Control mode 2

Control mode 2

The following settings apply for control mode 2 (x stands for channel n):

- xQ0: Rapid traverse/creep speed
- xQ1 Position reached
- xQ2 Travel plus
- xQ3 Travel minus

The following image shows the function of the digital outputs with control mode 2.



Data used in the parameter DB

Address	Name	Type	Initial value	Comment
92.0	CTRL_TYPE	DINT	L#1	2 = Control mode 2

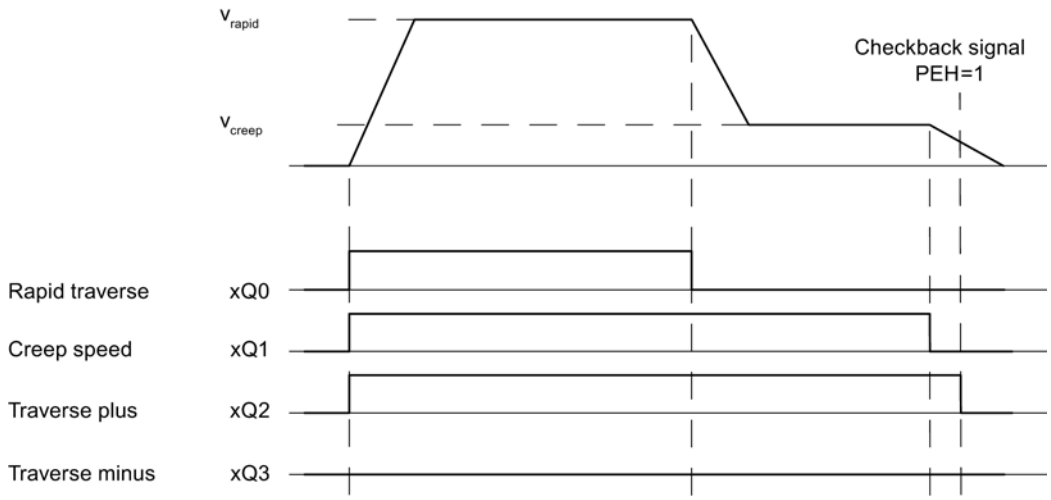
Control mode 3

Control mode 3

The following settings apply for control mode 3 (x stands for channel n):

- xQ0: Rapid traverse
- xQ1 Creep speed
- xQ2 Travel plus
- xQ3 Travel minus

The following image shows the function of the digital outputs with control mode 3.



Data used in the parameter DB

Address	Name	Type	Initial value	Comment
92.0	CTRL_TYPE	DINT	L#1	3 = Control mode 3

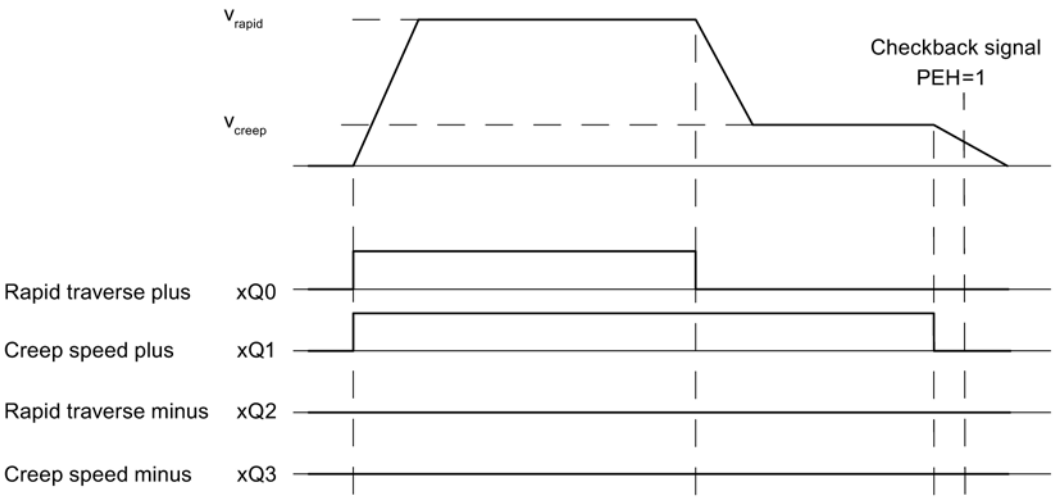
Control mode 4

Control mode 4

The following settings apply for control mode 4 (x stands for channel n):

- xQ0: Rapid traverse plus
- xQ1 Creep speed plus
- xQ2 Rapid traverse minus
- xQ3 Creep speed minus

The following image shows the function of the digital outputs with control mode 4.



Data used in the parameter DB

Address	Name	Type	Initial value	Comment
92.0	CTRL_TYPE	DINT	L#1	4 = Control mode 4

Specifying changeover difference

Changeover difference

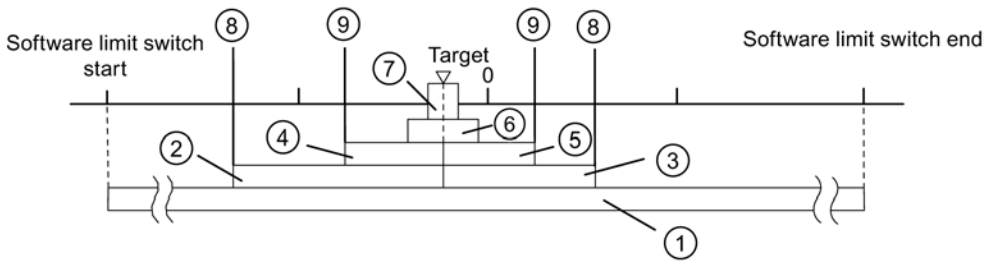
The changeover difference defines the distance from the destination at which the drive is switched over from rapid feed to creep feed.

The specified values apply to all targets approached by the positioning module.. An incremental approach with the incremental dimension 255 is an exception. A special changeover difference is specified for this.

Rules for changeover difference

- The changeover difference plus and the changeover difference minus do not have to be arranged symmetrically around the target. The values can differ.
- The changeover difference must be greater than the switch-off difference.
- The changeover difference must lie within the operating range.
- The changeover difference must be less than the end of rotary axis.
- Select a sufficient distance between the changeover point and the switch-off point to ensure that the drive can safely switch from rapid traverse to creep speed.
- The changeover point and the switch-off point must be separated from each other by a time interval of at least 8 ms.

The following image shows the location of the switch areas around the target.



①	Operating range	⑦	Target range
②	Changeover difference plus / minus	⑧	Changeover point
③			
④	Switch-off difference plus / minus	⑨	Switch-off point
⑤			
⑥	Standstill range		

For more information regarding the arrangement of the ranges, please see "Ranges and switching points of the positioning modules".

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
100.0	CHGDIF_P	DINT	L#5000	**Changeover difference plus**
104.0	CHGDIF_M	DINT	L#5000	**Changeover difference minus** Valid value range: • 1 µm to 1 000 000 000 µm at resolution ≥ 1 µm/pulse • 1 µm to 100 000 000 µm at resolution <1 µm/pulse

Specifying switch-off difference

Switch-off difference

The "switch-off difference" defines the distance to the destination at which the drive (creep speed) is switched off. The positioning module then adopts the monitoring functions from this point on.

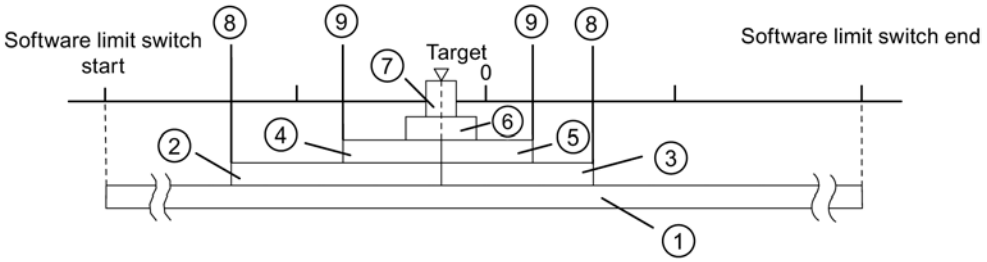
The specified values apply to all targets approached by the positioning module.. An incremental approach with the incremental dimension 255 is an exception. A special switch-off difference is specified for this.

Rules for switch-off difference

- The switch-off difference plus and the switch-off difference minus do not have to be arranged symmetrically around the target. The values can differ.
- The switch-off difference must be greater than half the target range.

- Select the distance between the switch-off point and the target to ensure that the drive comes to a standstill within the target range.
- The switch-off point and the start of the target range must be separated from each other by a time interval of at least 8 ms.

The following image shows the location of the switch areas around the target.



①	Operating range	⑦	Target range
② ③	Changeover difference plus / minus	⑧	Changeover point
④ ⑤	Switch-off difference plus / minus	⑨	Switch-off point
⑥	Standstill range		

For more information regarding the arrangement of the ranges, please see "Ranges and switching points of the positioning modules".

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
108.0	CUTDIF_P	DINT	L#2000	**Switch-off difference plus**
112.0	CUTDIF_M	DINT	L#2000	**Switch-off difference minus** Valid value range: • 1 µm to 1 000 000 000 µm at resolution ≥ 1 µm/pulse • 1 µm to 100 000 000 µm at resolution <1 µm/pulse

Specifying target approach

Specifying target range

Target range

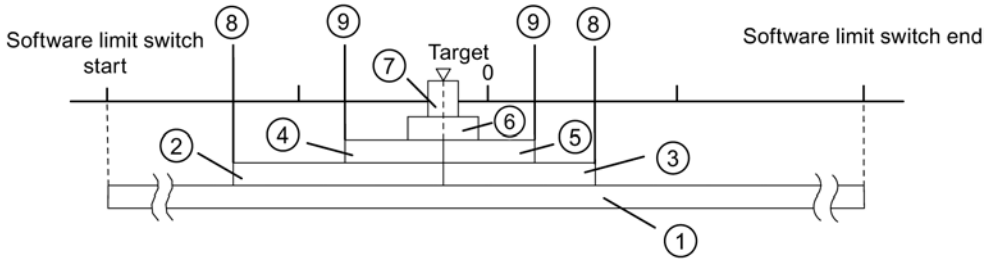
The positioning module sets a symmetrical range around each target, the target range. The actual value must lie within this range with a target approach.

A value specification of "0" switches off the tolerance during the target approach. The speed when reaching the target must then be less than the assigned standstill velocity (Page 10605) parameter.

Rules for the target range

- The target range lies symmetrically around the target.
- A value specification of "0" switches off the monitoring of the target range.

The following image shows the location of the ranges around the target.



①	Operating range	⑦	Target range
②	Changeover difference plus / minus	⑧	Changeover point
③			
④	Switch-off difference plus / minus	⑨	Switch-off point
⑤			
⑥	Standstill range		

Additional information on target approach can be found under "Ranges and switching points of the positioning modules" as well as under "End of a positioning operation (Page 10557)".

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
76.0	TRG_RANGE	DINT	L#1000	**Target range** Valid value range: • 0 = no monitoring • 1 µm to 1 000 000 000 µm at resolution ≥ 1 µm/pulse • 1 µm to 100 000 000 µm at resolution <1 µm/pulse

Specifying monitoring time

Monitoring time

The module monitors the following during the monitoring time:

- the movement of the axis up to the switch-off point.
  The monitoring time starts with the start of a positioning and is retriggered with each actual value change in the traversing direction.
  Within the monitoring time, the actual value must have changed by at least one increment (the distance of the resolution).
- the target approach.
  The monitoring time is retriggered when the switch-off difference is reached for the last time. The target range must be reached within the monitoring time.

Additional information on target approach can be found under "End of a positioning operation (Page 10557)".

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
80.0	MON_TIME	DINT	L#2000	**Monitoring time** Valid value range: • 0 = no monitoring • 1 to 100 000 ms
Actual monitoring time You can specify all values from the range of values for the monitoring time. The module rounds the specified time to a multiple of 8 ms (module cycle).				

Specifying standstill range

Standstill range

The standstill range lies symmetrically around the respective target. In this range, monitoring is performed to determine whether the drive remains on an approached target position or drifts away from it. The monitoring begins when the positioning module has reported "Position reached".

If the standstill range is exited without a valid traversing job, the positioning module signals an error.

Rules for standstill range

- The standstill range lies symmetrically around the respective target.
  (Target – ½ × standstill range) ≤ target ≤ (target + ½ × standstill range)
- The standstill range must be greater than or equal to the target range.

- The coordinate values for the standstill range must lie in the operating range.
- A setting of "0" deactivates the standstill monitoring function.

Additional information on target approach can be found under "Ranges and switching points of the positioning modules" as well as under "End of a positioning operation (Page 10557)".

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
84.0	ZSPEED_R	DINT	L#1000	**Standstill range** Valid value range: • 0 = no monitoring • 1 µm to 1 000 000 000 µm at resolution ≥ 1 µm/pulse • 1 µm to 100 000 000 µm at resolution <1 µm/pulse

Specifying standstill velocity

Standstill velocity

The standstill velocity serves as a reference speed. If, during a positioning operation, the actual speed reaches the standstill velocity and the drive is located in the target range, then "Position reached" (PR) is set. The positioning operation is therefore successfully completed.

The precision of the positioning increases when the value of the standstill velocity is reduced. A setting of "0" deactivates the standstill monitoring function.

Additional information on target approach can be found under "End of a positioning operation (Page 10557)".

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
88.0	ZSPEED_L	DINT	L#30000	**Standstill velocity** Valid value range: • 0 = no monitoring • 1 µm/min to 100 000 µm/min

Channel n axis

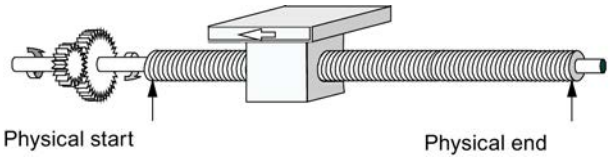
Specifying axis type

Selecting axis type

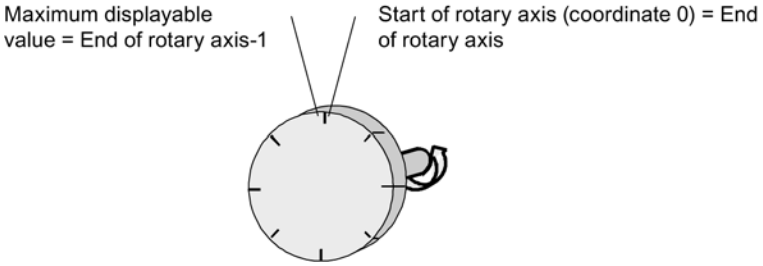
Axis types

The positioning module can control the following axis types:

- Linear axis
  A linear axis is an axis which has a limited physical traversing range.



- Rotary axis
  The rotary axis is an axis whose traversing range is not restricted by mechanical stops.



Data used in the parameter DB

Address	Name	Type	Initial value	Comment
12.0	AXIS_TYPE	DINT	L#0	**Axis type** Valid values: • Linear axis = 0 • Rotary axis = 1

Specifying software limit switch

Software limit switch

Software limit switches only exist for the linear axis.

The operating range is the range limited by the software limit switches.

The software limit switch start must always be smaller than the software limit switch end.

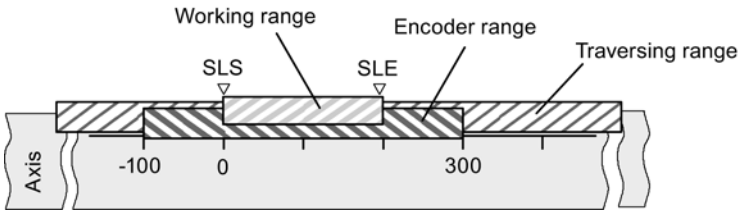
The software limit switches are monitored after the axis is synchronized.

- When using an incremental encoder:
  The synchronization of the axis is achieve by a reference point approach or by setting a reference point.
- When using an absolute encoder (SSI):
  The axis is synchronized when the module has received a complete, error-free frame for the associated channel.
  The absolute encoder must cover at least the operating range including the software limit switches.

Relationship between the operating range, encoder range and traversing range

- The operating range is the range which you specify for your job using the software limit switches.
- The encoder range represents the range explicitly covered by the encoder. For a linear axis, the module assigns the encoder range symmetrically across across the operating range. In other words, the module shifts the encoder range in order to equalize the distances between the software limit switches and the ends of the encoder range (see figure below).
- This traversing range represents the value range the positioning module can process. The traversing range is dependent on the resolution.

The following figure shows the relationship between software limit switches, the operating range, the encoder range and the traversing range.



Data used in the parameter DB

Address	Name	Type	Initial value	Comment
64.0	SSW_STRT	DINT	L#-100000000	**Software limit switch start**
68.0	SSW_END	DINT	L#100000000	**Software limit switch end** Valid value range: • -1,000,000,000 µm to 1 000 000 000 µm at resolution ≥ 1 µm/pulse • -100,000,000 µm to 100,000,000 µm at resolution <1 µm/pulse

Specifying end of rotary axis

End of rotary axis

The value "end of rotary axis" is the theoretical maximum value that the actual value can reach. The theoretically highest value is however never displayed, because it physically identifies the same position as the start of the rotary axis (0).

The maximum value that will be displayed for a rotary axis has the value:

End of rotary axis [µm] - resolution [µm / pulse] x 1 [pulse]

Example: End of rotary axis 1000 mm, resolution 1000 µm / pulse

The display jumps:

- with a positive rotational direction from 999 mm to 0 mm
- with a negative rotational direction from 0 mm to 999 mm.

Rotary axis with absolute encoders

The rotary axis range (0 to end of rotary axis) of a rotary axis with an absolute encoder must correspond exactly to the encoder range of the absolute encoder:
End of the rotary axis [µm] = number of encoder revolutions x distance/encoder revolutions [µm]

Data used in the DB parameter

Address	Name	Type	Initial value	Comment
16.0	ENDROTAX	DINT	L#100000	**End of rotary axis** Valid value range: • 1 µm to 1 000 000 000 µm at resolution ≥ 1 µm/pulse • 1 µm to 100 000 000 µm at resolution <1 µm/pulse

Specifying reference point approach

Selecting type of reference point approach

Type of reference point approach

With "Type of reference point approach", you determine the conditions for the axis synchronization.

- The first specification defines the start direction in which the reference point approach is starting.
- The second specification defines the location of the zero mark, which leads to synchronization, as far as the reference point switch is concerned.

Detailed information can be found under "Reference point approach" operating mode (Page 10567).

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
52.0	REFPT_TYPE	DINT	L#0	**Type of reference point approach** Valid values: • 0 = plus, reference point switch in plus direction • 1 = plus, reference point switch in minus direction • 2 = minus, reference point switch in plus direction • 3 = minus, reference point switch in minus direction

Specifying reference point coordinate

Reference point coordinate

- **Incremental encoders:**
  You require the reference point coordinate for the "Reference point approach" mode. If the axis is not synchronized following writing and activating of machine data, the actual value is set to the value of the reference point coordinate.
- **Absolute encoder (SSI)**
  You require the reference point coordinates for the mechanical adjustment of the encoder. For more information, refer to the description of absolute encoder adjustment (Page 10610) which explains the interaction of the absolute encoder adjustment with other data.

The value of the reference point coordinate must lie within the operating range:

- **Linear axis**
  Including the software limit switches.
- **Rotary axis**
  Greater than or equal to 0 and less than the "end of rotary axis" value (0 ≤ reference point coordinate < end of rotary axis).

Data used in the DB parameter

Address	Name	Type	Initial value	Comment
44.0	REFPT	DINT	L#0	**Reference point coordinate** Valid value range: • -1 000 000 000 µm to 1 000 000 000 µm at resolution ≥1 µm/pulse • -100 000 000 µm to 100 000 000 µm at resolution < 1 µm/pulse

Specifying absolute encoder adjustment

Determining the absolute encoder adjustment

Absolute encoder adjustment

Absolute encoder adjustment is only necessary for an absolute encoder.

The absolute encoder adjustment and reference point coordinate map the encoder's range of values explicitly to the axis coordinate system.

Additional information can be found under "Absolute encoder adjustment procedure (Page 10610)" and in an example (Page 10611).

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
44.0	REFPT	DINT	L#0	**Reference point coordinate** Valid value range: • -1 000 000 000 µm to 1 000 000 000 µm at resolution ≥ 1 µm/pulse • -100,000,000 µm to 100 000 000 µm at resolution <1 µm/pulse
48.0	ENC_ADJ	DINT	L#0	**Absolute encoder adjustment** Valid value range: • 0 to (2^{25} - 1)

Absolute encoder adjustment procedure

Determining the correct absolute encoder adjustment

After initial parameter assignment, further steps must be taken to ensure that a correct relationship is created between the encoder and coordinate system. The procedure shown in the next section is based on the use of the parameter assignment dialogs.

1. Traverse the axis to a defined, reproducible point to which a unique coordinate is assigned. This can, for example, be the "software limit switch end".
2. Call the job "Set reference point (Page 10584)" with the coordinate of the point defined under 1.
   The positioning module now determines an encoder value for the reference point coordinate entered in the DB channel (REFPT in channel DB), namely the absolute encoder adjustment. You can read this value with the "Read encoder data (Page 10592)" job.
3. Enter the value read in the "absolute encoder adjustment" field.
4. Save your parameter assignment to the corresponding parameter DB using the export function.

5. Download the data in the HWCN to the CPU.
6. Restart the CPU (cold restart) to apply the data.

Note

You perform this adjustment just once during the commissioning. After a configuration, the position module is synchronized during start-up as soon as a complete, faultless message frame of the encoder is received after start-up.

Otherwise: Mechanical adjustment of the encoder

A correct relationship between the coordinate system and the encoder can be achieved as follows:

1. Move the axis to a defined, reproducible point to which a unique coordinate is assigned. This can, for example, be the "software limit switch start".
2. Enter this coordinate value in the machine data as "reference point coordinate".
3. Read the encoder value displayed at this position with the "Read encoder data (Page 10592)" job.
4. Enter this value in the machine data as the "absolute encoder adjustment".

After the parameter assignment a correct actual value will always be shown.

Instead of steps 3 and 4, you can also set the encoder to zero using "Reset", if available. Then enter the value "0" in the machine data as the absolute encoder adjustment.

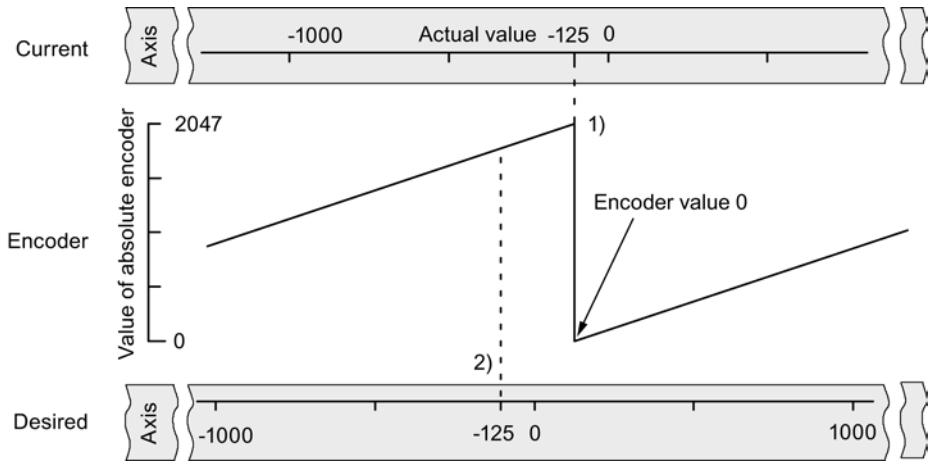
Example of absolute encoder adjustment

Example of an absolute encoder adjustment

The following assumptions apply to the example:

- Reference point coordinate = -125 mm
- Operating range from SSW_STRT = -1000 mm to SSW_END = 1000 mm
- Absolute encoder adjustment = 0
- Encoder range = 2048 increments with a resolution of 1 mm/pulse
- The absolute encoder used cannot be adjusted mechanically with precision and it also is not possible to set the encoder value specifically.

The following figure shows an example of absolute encoder adjustment.



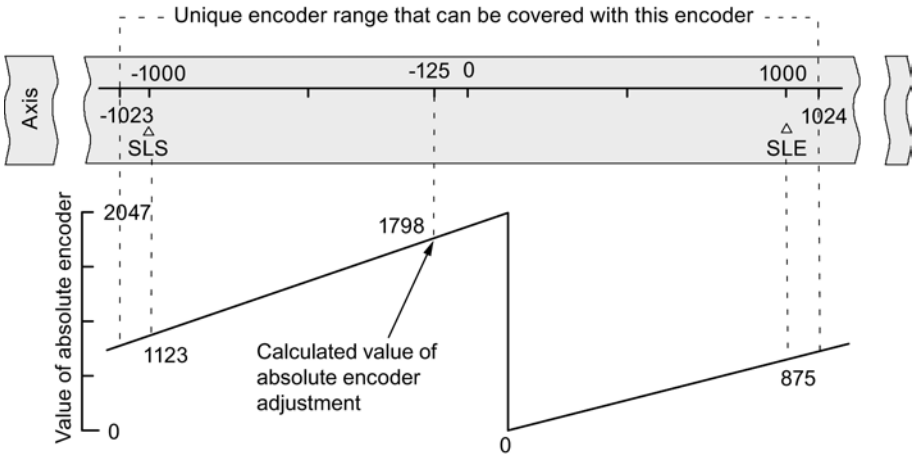
1) Assignment of the coordinate system to encoder values, based on the absolute encoder adjustment. The encoder value 0 corresponds to the actual value -125.

2) Required assignment of the coordinate system to the encoder. At this position the coordinate should be -125.

Result in accordance with "Set reference point"

After "Set reference point" the relationship looks as follows:

The reference point coordinate on the axis (-125) is assigned to the encoder value (1798) determined from absolute encoder adjustment. This assignment is set out in the figure below.



The encoder supplies 2048 unique values. The operating range is defined by the software limit switches. The selected resolution of 1 mm per pulse means the encoder can, however, cover a greater operating range than the one set by the software limit switches.

With the set resolution, the operating range is already covered with 2001 values. Therefore, in the example, 47 pulses "remain" and these are symmetrically positioned around the operating range.

Selecting start velocity

Start speed in the case of a reference point approach

With this data you select the speed for the start of the reference point approach.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
99.0	REFPT_SPD	BOOL	TRUE	**Start speed for reference point approach** Valid values: • 0 = rapid traverse • 1 = creep speed

Specifying minimum edge-to-edge distance

Minimum edge distance

With the machine data "Minimum edge-to-edge distance", you define a range after the start of a measurement is detected with an edge detection. The measurement will be discarded if the measuring operation ends within this range.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
4.0	EDGEDIST	DINT	L#0	**Minimum edge distance** Valid value range: • 0 to 1 000 000 000 µm at resolution ≥1 µm/pulse • 0 to 100 000 000 µm at resolution <1 µm/pulse

Channel n encoder

Selecting the signal type

Signal type

The encoder provides distance information to the module. The module then evaluates this information and converts this with the resolution into an actual value.

You can select the signal type from one of the following encoders:

- 5 V incremental encoder
  Select the signal type "5 V differential signal" to use a 5 V incremental encoder with differential signal A, /A, B, /B and N, /N.
- 24 V incremental encoder
  Select signal type "24 V track A+B phase shifted" to use a 24 V incremental encoder with signals A*, B* and N*.
- Absolute encoder (SSI)
  Select the "SSI absolute" signal type if you use an absolute encoder with a serial interface.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
20.0	ENC_TYPE	DINT	L#1	**Signal type** Valid values: • 1 = 5 V differential signal • 2 = 24 V track A+B phase shifted • 3 = SSI 13-bit frame length [1)] • 4 = SSI 25-bit frame length [1)] • 5 = SSI 14-bit frame length [1)] • 6 = SSI 15-bit frame length [1)] • 7 = SSI 16-bit frame length [1)] • : • : • 14 = SSI 23-bit frame length [1)] • 15 = SSI 24-bit frame length [1)]

[1)] The frame length is selected in the "SSI" dialog box.

Selecting the count direction

Count direction

With the machine data "Count direction" you can adapt the direction of the position measuring to the direction of movement of the axis. You must also take into account the rotational direction of all transmission elements (for example, couplings and gears).

- Normal = incremental count pulses (incremental encoder) or encoder values (absolute encoder) represent increasing actual position values
- Inverted = incremental count pulses (incremental encoder) or encoder values (absolute encoder) correspond to decreasing actual position values

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
59.0	CNT_DIR	BOOL	FALSE	**Count direction** Valid values: • 0 = normal • 1 = inverted

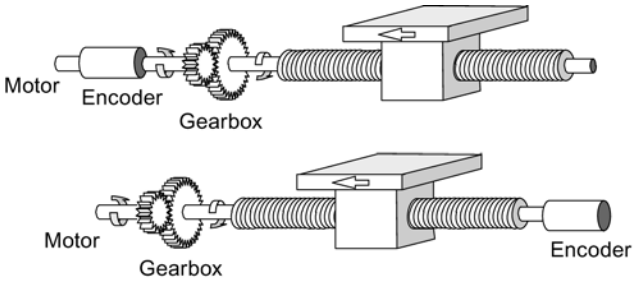
Specifying resolution

Specify distance per encoder revolution

Distance per encoder revolution

With the machine data "Distance per encoder revolution" you can inform the positioning module which distance the drive system should cover per encoder revolution.

The "Distance per encoder revolution" value depends on the structure of the axis and the way in which the encoder is mounted. You must take into account all the transfer elements like clutches or gearboxes. The following figure shows an example of two differently-mounted encoders.



The relationship between the "Distance per encoder revolution" and "Increments per encoder revolution" machine data is described under resolution (Page 10616).

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
24.0	DISP_REV	DINT	L#80000	**Distance per encoder revolution** Valid value range: • 1 µm to 1 000 000 000 µm

Specifying increments per encoder revolution

Increments per encoder revolution

The machine data "Increments per encoder revolution" indicates the number of increments that an encoder makes for each revolution. The positioning module calculates the resolution (Page 10616) based on this value and the "Distance per encoder revolution" machine date.

Incremental encoder

Any value within the range of values can be entered. The module evaluates the increments four times.

Absolute encoder

For the limits you have to differentiate between the individual encoder models. Only values to the power of 2 are permitted as the entered value.

- Single-turn encoder (number of revolutions = 1) with 13-bit frame length:
  - minimum value = 4
  - maximum value = 8192
- Multiturn encoder (number of revolutions > 1) with 25-bit message frame length:
  - Minimum value = 4
  - Maximum value = 8192
- Single-turn encoder (number of revolutions = 1) with 25-bit frame length:
  - Minimum value = 4
  - maximum value = 2^{25}

Linear scales will be configured as multiturn encoders as follows:

- Increments per encoder revolution = 8192
- Number of revolutions x 8192 ≥ number of steps in the linear scale

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
32.0	INC_REV	DINT	L#500	**Increments per encoder revolution** Valid value range: • 1 to 2^{25}

Resolution

Resolution

The resolution indicates the traversing distance corresponding to one pulse. It is a measurement for the precision of the positioning and also determines the possible maximum traversing range of the positioning module.

The resolution (RES) can be calculated as in the following table:

	Incremental encoder	Absolute encoder
Input values	• Distance per encoder revolution • Increments per encoder revolution: – Quadruple evaluation – 1 increment = 4 pulses	• Distance per encoder revolution • Increments per encoder revolution: – 1 increment = 1 pulse
Calculation	$RES = \frac{\frac{\text{Distance}}{\text{Encoder revolution}}}{\frac{\text{Pulses}}{\text{Encoder revolution}}}$	

Note

All position information is rounded up to the integral multiple of the resolution. This way the entered values differentiate from the used values.

Range of values of the resolution

The chosen system of units determines the range of values of the resolution:

System of units	Information in ...	Range of values of the resolution
mm	10^{-3} mm	$0{,}1 \times 10^{-3}$ … 1000×10^{-3} mm/pulse
inch	10^{-4} inch	$0{,}1 \times 10^{-4}$ … 1000×10^{-4} inch/pulse
degrees	10^{-4} degrees 10^{-3} degrees 10^{-2} degrees	$0{,}1 \times 10^{-4}$ … 1000×10^{-4} degrees/pulse $0{,}1 \times 10^{-3}$ … 1000×10^{-3} degrees/pulse $0{,}1 \times 10^{-2}$ … 1000×10^{-2} degrees/pulse

Example

- An incremental encoder has the following data:
  - Increments per encoder revolution: 5000
  - Distance per encoder revolution: 1000 mm
  - 1 increment = 4 pulses

  This results in the resolution (4 x evaluation):

$$\text{Resolution} = \frac{1000\ \text{mm}}{5000\ \text{increments}} = 0.2000\ \frac{\text{mm}}{\text{Increment}} = 0.2000\ \frac{\text{mm}}{4\ \text{pulses}}$$

$$= 0.0500\ \frac{\text{mm}}{\text{Pulse}}$$

- An SSI encoder has the following data:
  - Increments per encoder revolution: 4096
  - Distance per encoder revolution: 1000 mm
  - 1 increment = 1 pulse

  This results in the resolution:

$$\text{Resolution} = \frac{1000\ \text{mm}}{4096\ \text{increments}} = 0.2441\ \frac{\text{mm}}{\text{Increment}} = 0.2441\ \frac{\text{mm}}{\text{Pulse}}$$

Relationship between the traversing range and resolution

The traversing range is restricted by the numerical representation in the position module. This numerical representation varies depending on the resolution. You should thus make sure that your specifications are always within the permitted limits.

The maximum traversing range is illustrated in the following table:

Resolution (RES) lies in the range	Maximum traversing range
0.1 µm/pulse ≤ RES < 1 µm/pulse	-10^8 µm to 10^8 µm (-100 m to +100 m)
1 µm/pulse ≤ RES ≤ 1000 µm/pulse	-10^9 µm to 10^9 µm (-1000 m to +1000 m)

Date for absolute encoder

Number of encoder revolutions

Number of encoder revolutions

The machine date "Number of encoder revolutions" only applies to absolute encoders. As a result, enter the number of revolutions that are possible with this encoder.

The **overall number of steps of the encoder** is not machine data. It is calculated as follows:

Overall number of steps = increments/encoder revolution x number of revolutions

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
36.0	NO_REV	DINT	L#1	**Number of encoder revolutions** Valid value range: • 1 (single-turn encoder) • 2 to 4096 in powers of 2 (multiturn encoder)

Transmission rate

Transmission rate

With the machine data "Transmission rate" you can determine the data transmission rate from the SSI encoder to the positioning module.

The maximum Transmission rate is dependent on the cable length:

- 320 m → 125 kHz
- 160 m → 250 kHz
- 63 m → 500 kHz
- 20 m → 1,000 kHz

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
40.0	BAUDRATE	DINT	L#0	**Transmission rate** Valid values: • 0 = 125 kHz • 1 = 250 kHz • 2 = 500 kHz • 3 = 1,000 kHz

Frame length

Frame length

With "frame" you define the clock frame output by the positioning module for absolute encoder.

A frame between 13 bits and 25 bits is permitted.

Data used in the parameter DB

The frame length selected here is stored in the parameter DB under the ENC_TYPE (Page 10613) date in the parameter DB.

Parity

Parity

With "Parity", you specify whether no parity, even parity or odd parity should be assigned for an absolute encoder.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
28.0	PARITY	DINT	L#0	**Parity** Valid values: • 0 = none • 1 = even parity • 2 = odd parity

Selecting monitoring

Monitoring

Wire break

When monitoring is activated, the positioning module monitors all cables for a 5 V incremental encoder and an absolute encoder. The monitoring detects:

- Wire break
- Short circuit on individual cables
- Edge distance of counting pulses (also for 24 V incremental encoder)

For monitoring for a 24 V incremental encoder, you must set MON_TIME > 0 as the monitoring time (Page 10604).

With 5 V incremental encoders with zero marks, you must disable the wire break monitoring or interconnect the N and /N signals externally.

Frame error

The module monitors the message frame of an absolute encoder (SSI) for:

- Start and stop bit errors

You cannot disable the message frame error monitoring with absolute encoders (SSI).

Missing pulses (incremental encoder)

An incremental encoder must always return the same number of increments between two successive zero marks.

The positioning module checks to see if the zero mark of an incremental encoder is occurring at the correct encoder state.

For encoders without zero marks you must switch off the monitoring for missing pulses.

You must also disable the wire-break monitoring or interconnect the N and /N zero mark inputs externally.

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
				Monitoring:
63.0	MON_WIRE	BOOL	TRUE	1 = wire break
63.1	MON_FRAME	BOOL	TRUE	1 = message frame error (must always be 1)
63.2	MON_PULSE	BOOL	TRUE	1 = missing pulses

Channel n incremental dimensions

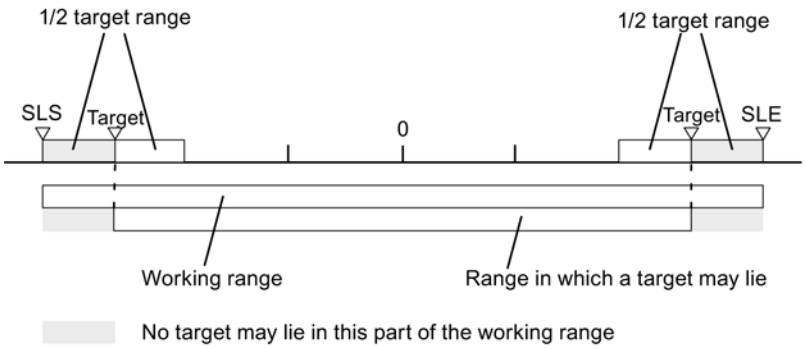
Specifying incremental dimensions

Definition

Incremental dimensions are target specifications that can be controlled by the positioning module with the "relative incremental approach" or rather "absolute incremental approach" mode.

Possible range for incremental dimensions

The target that is to be approached must be at a distance of at least one-half the target range in front of the respective software limit switch. The following figure shows the limits for specifying incremental dimensions.



Increment number 1 to 100

Incremental dimension number 1 to 100

You have the possibility of entering up to 100 incremental dimensions in a table that are valid both for the "Relative incremental approach" mode as well as for the "Absolute incremental approach" mode.

Note that the positioning module does not permit negative values for the "relative incremental approach". The values will be interpreted by the positioning module - depending on the direction of movement - as either a positive or a negative difference.

Note

The entry is made in the unit of the system of units set. Note the decimal places in particular here.

Numerical example:

- Incremental dimension: 800 mm
- System of units: 10^{-3} mm
- Entry in the DB parameter: 800000

Tip: Define separate areas for relative and absolute incremental dimensions in the incremental dimension table.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
35.4	TRGL1WR_EN	BOOL	FALSE	1 = write increment table 1 (increments 1 ... 50)
35.5	TRGL2WR_EN	BOOL	FALSE	1 = write increment table 2 (increments 51 ... 100)
36.6	TRGL1RD_EN	BOOL	FALSE	1 = read increment table 1 (increments 1 ... 50)
36.7	TRGL2RD_EN	BOOL	FALSE	1 = read increment table 2 (increments 51 ... 100)

Data used in the parameter DB

Address	Name	Type	Initial value	Comment	
120.0	TRGL1.TRG[1]	DINT	L#0	Incremental dimension number 1	Increment table 1
. . .	. . .	. .	. . .	. . .	
316.0	TRGL1.TRG[50]	DINT	L#0	Incremental dimension number 50	
320.0	TRGL2.TRG[51]	DINT	L#0	Incremental dimension number 51	Increment table 2

Address	Name	Type	Initial value	Comment	
. . .	. . .	. . .	. . .	. . .	
516.0	TRGL2.TRG[100]	DINT	L#0	Incremental dimension number 100	

Increment number 252

Increment number 252

The increment number 252 gives you the option of moving the drive initially in a specified direction without a target (continuous traversing) and positioning the drive at the assigned target (increment) on occurrence of an external event at digital input xI3.

The entries from the parameter DB for this increment are applicable to the changeover and switch-off differences.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
36.2	TRG252_254_EN	BOOL	FALSE	1 = Write incremental dimension for incremental dimension number 252 or 254
96.0	TRG252_254	DINT	L#0	Incremental dimension for incremental dimension number 252 or 254

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
100.0	CHGDIF_P	DINT	L#5000	Changeover difference plus
104.0	CHGDIF_M	DINT	L#5000	Changeover difference minus
108.0	CUTDIF_P	DINT	L#2000	Switch-off difference plus
112.0	CUTDIF_M	DINT	L#2000	Switch-off difference minus

Increment number 254

Increment number 254

Independently of the increment table you can use the increment number 254 as an additional distance specification. For the changeover and switch-off differences, the entries from the parameter DB are valid for this increment.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
36.2	TRG252_254_EN	BOOL	FALSE	1 = Write incremental dimension for incremental dimension number 252 or 254
96.0	TRG252_254	DINT	L#0	Write incremental dimension for incremental dimension number 252 or 254

Data used in the parameter DB

Address	Name	Type	Initial value	Comment
100.0	CHGDIF_P	DINT	L#5000	Changeover difference plus
104.0	CHGDIF_M	DINT	L#5000	Changeover difference minus
108.0	CUTDIF_P	DINT	L#2000	Switch-off difference plus
112.0	CUTDIF_M	DINT	L#2000	Switch-off difference minus

Increment number 255

Incremental dimension number 255

The incremental dimension number 255 gives you an additional guideline for the path.

The switch-off differences and the changeover differences are presented together with the increment. Unlike the situation with the other increments, the increment 255 uses the values established in the channel DB for the switch-off and changeover differences. The entries from the machine data have no validity for this increment.

Data used in the channel DB

Address	Name	Type	Initial value	Comment
36.3	TRG255_EN	BOOL	FALSE	1 = write increment for increment number 255
100.0	TRG255	DINT	L#0	Increment for increment number 255
104.0	CHGDIF255	DINT	L#0	Changeover differences for increment number 255
108.0	CUTDIF255	DINT	L#0	Switch-off differences for increment number 255

Control and feedback interface of the FM 451

Application

In special applications or in an alarm level, a particularly fast access to checkback and control signals could be required. You can reach this data directly via the input and output areas of the module.

To coordinate startup following each module startup, for example, after plugging a module or after a STOP-RUN transition of the CPU, you must call the ABS_CTRL (Page 7125) instruction until the end of the startup is indicated by RETVAL = 0. Then, do not use the ABS_CTRL anymore.

Note

It is not possible to use the ABS_CTRL together with a write access.

Reading checkback signals by means of direct access

The byte addresses must be specified relative to the start address of the outputs of the respective channel. The names of the parameters correspond to the names in the channel DB (Page 7133).

Start address channel 1 = start address of the module

Start address channel 2 = start address of the module + 8

Start address channel 3 = start address of the module + 16

In STL, you access the data using the commands PEB (read 1 byte), PEW (read 2 bytes) and PED (read 4 bytes).

The following table shows the assignment of the feedback interface (inputs) of FM 451.

Address	Bit number							
	7	6	5	4	3	2	1	0
Byte 0	PARA	Internal	Internal	DATA_ERR	OT_ERR	DIAG	Internal	Internal
Byte 1	CHGOVER	CUTOFF	ZSPEED	SPEED_OUT	0	WAIT_EI	WORKING	ST_ENBLD
Byte 2	MODE_OUT							
Byte 3	POS_RCD	0	FVAL_DONE	0	GO_P	GO_M	MSR_DONE	SYNC
Byte 4	ACT_POS							
Byte 5								
Byte 6								
Byte 7								

Example: Position process value ACT_POS

The start address of the module is 512.

STL	Description

L PED 516	//Read the current position process value (ACT_POS) from channel 1
	//with direct access:
	//Start address of the channel + 4

Writing control signals using direct access

The byte addresses must be specified relative to the start address of the inputs of the respective channel. The names of the parameters correspond to the names in the channel DB (Page 7133).

Start address channel 1 = start address of the module

Start address channel 2 = start address of the module + 8

Start address channel 3 = start address of the module + 16

In STL, you access the data using the commands PAB (write 1 byte), PAW (write 2 bytes) and PAD (write 4 bytes).

The following table shows the assignment of the control interface (outputs) of FM 451.

Address	Bit number							
	7	6	5	4	3	2	1	0
Byte 0	0	0	0	0	OT_ERR_A	0	0	0
Byte 1	DRV_EN	SPEED252	0	0	DIR_P	DIR_M	STOP	START
Byte 2	MODE_IN							
Byte 3	MODE_TYPE							
Byte 4	Reserved							
Byte 5								
Byte 6								
Byte 7								

Example: START signals of channel 2

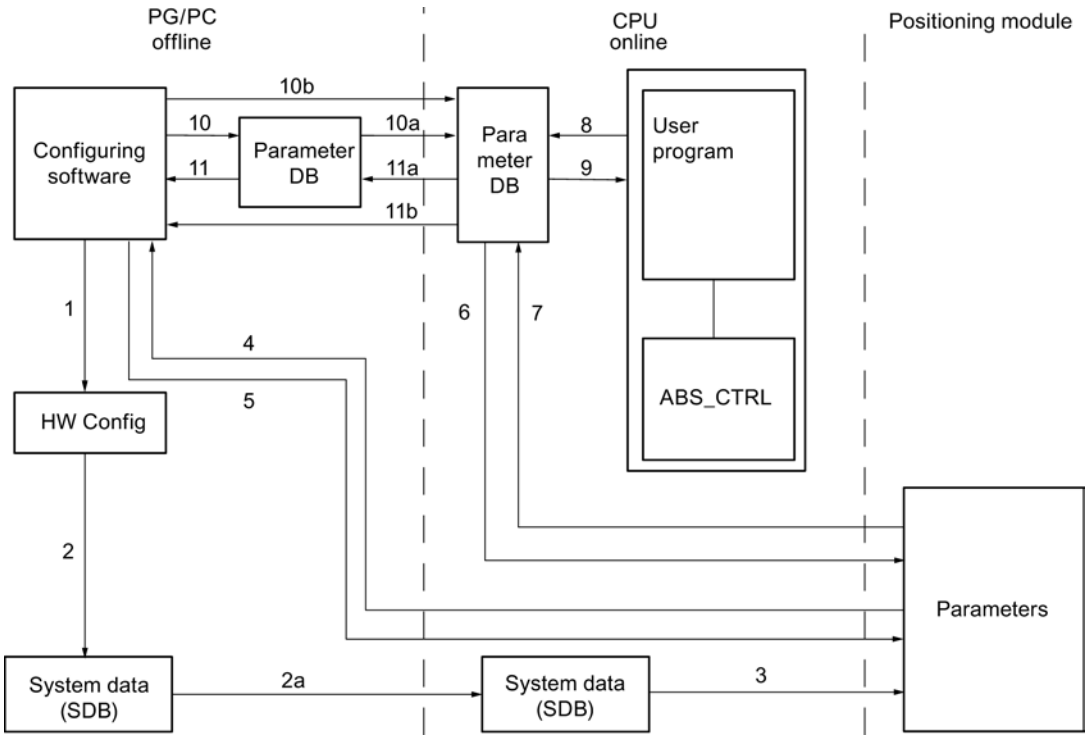
The start address of the module is 512.

```
STL                 Description
L 2#10001000        //Set DRV_EN and DIR_P to 1
T PAB 521           //Write signals with direct access for channel 2:
                    //Start address of the module + 8 + 1
```

Parameter transmission paths

Parameters include the following machine data and increments.

The following figure shows the parameter transmission paths of the FM x51.



1 Saving parameters in the configuration software.

2 Save and compile the HW configuration.

2a Download the HW configuration to the CPU. The CPU automatically carries out step 3.

3 The CPU writes the parameters when system parameter assignments are carried out for the module.

4 With the command "download target system to PG" download the parameters of a channel of the module to the PG.

5 With the command "download target system" download the parameters from the configuration software into a channel of the module.

6 Write parameters using jobs of the user program in a channel of the module.

7 Read parameters using jobs of the user program from a channel of the module.

8 Store parameters from the user program into the online DB.

9 Read in parameters from the online DB into the user program.

10 Export parameters from the configuration software to the offline DB.

10a Download the offline DB to the CPU.

10b Export parameters from the configuration software to the online DB.

11 Import parameters from the offline DB into the configuration software.

11a Load parameters from the online DB to the PG.

11b Import parameters from the online DB into the configuration software.

Some use cases for the transfer of parameters

Use case	Steps
You process the parameters using the configuration software. The channels of the module should then be automatically assigned upon start-up.	Carry out steps 1, 2, and 2a.
You change the parameters during commissioning in test mode in the configuration software.	Carry out steps 4 and 5.
The parameters edited during commissioning should be downloaded automatically during startup.	Carry out steps 1, 2, and 2a.
You create the parameters using the configuration software. The channels of the module should be assigned upon startup only by the user program via the data blocks.	Carry out steps 10, 10a and 6 or 10b and 6.
You would like to create conveniently stored data for recipes.	Carry out steps 10 and 10a.
You create the parameters using the configuration software. These should be made available to the user program for temporary changes.	Carry out steps 1, 2 and 2a for the automatic parameter assignment.
	Carry out the steps 10, 10a, 9, 8 and 6 for access by the user program.
You change existing parameters solely via the user program.	Carry out steps 7, 9, 8 and 6.
You would like to use the configuration software to view the data that has been changed with the user program.	Carry out steps 11a and 11 or just 11b.
Parameters modified by the user program should also be loaded automatically during startup.	Carry out steps 11b or 11a, 11 and then 1, 2, 2a.

Commissioning the FM 451

Using FM 451 from the user program

Integrating the FM 451 into a user program

Instructions are available for integrating the FM 451 into a user program. These enable you to handle the desired functions with ease.

You call the ABS_INIT and ABS_CTRL or ABS_CTRL_451 instructions in the CPU user program. They ensure communication between the CPU and the FM 451.

The channel DB (Page 7133) is created automatically as the instance DB for the ABS_CTRL instruction when you insert the ABS_CTRL instruction into the project.
When you insert the ABS_CTRL_451 instruction into the project, use the command "Program blocks > Add new block" to add a ABS_CHANTYPE type data block to your project as the channel DB.

All the parameter data associated with the channel of the module (machine data, incremental dimension table) is located in the parameter DB (Page 7139) for the module. Use the command "Program blocks > Add new block" to add a ABS_PARATYPE type data block to your project.

The ABS_DIAG or ABS_DIAG_451 instruction is also available for FM 451. This instruction lets you transfer the diagnostic information to the diagnostic DB.

The diagnostic DB (Page 7141) is created automatically as the instance DB for the ABS_DIAG instruction when you insert the ABS_DIAG instruction into the project.
When you insert the ABS_DIAG_451 instruction into the project, use the command "Program blocks > Add new block" to add a ABS_DIAGTYPE type data block to your project as the diagnostic DB.

Commissioning the module

The module can be initially tested using the "Commissioning" dialog, even when no user program is available. Note the following warning in particular in this case.

WARNING

Injury and material damage could occur.

To prevent personal injury and material damage, observe the following points:

- Install an emergency stop switch in the vicinity of the computer.
  This is the only way you can ensure that the plant can be safely shut down in the event of computer or software failure.
- Install hardware limit switches, which act directly on the power units of all drives.
- Make sure that no one has access to the area of the plant in which moving parts are present.
- When the FM 451 is controlled and monitored simultaneously by your program and the "Commissioning" dialog, conflicts can occur, the effects of which are not clear. Therefore, always set the CPU to STOP mode or disable your program when you work with the "Commissioning" dialog.

See also

ABS_INIT (Page 7124)

ABS_CTRL (Page 7125)

ABS_DIAG (Page 7127)

ABS_CTRL_451 (Page 7129)

ABS_DIAG_451 (Page 7131)

FM 451: Online & Diagnostics

Current module information

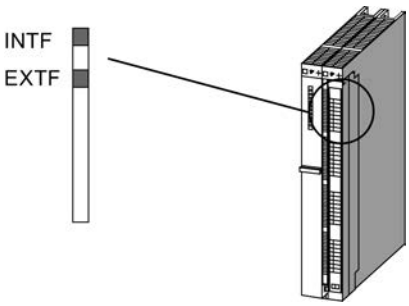
Default diagnostics

Error display through error LEDs of FM 451

Where is the error displayed?

The red error LEDs on the module display different error states. The LEDs are lit, even with errors that occur briefly, for at least 3 seconds.

The following figure shows the location of the error LEDs on FM 451.



Which errors are displayed?

The table below shows the error types indicated by the error LEDs.

LED	Meaning	Explanations
INTF	Group error for internal error	This LED indicates the following error states of FM 451: • Module defect • Internal (channel) error • The FM 451 is not configured. Configure the module. To remove the error, see Error classes (Page 10637).
EXTF	Group error for external error	This LED indicates an external (channel) error.

Channel-specific diagnostics

Triggering diagnostic interrupts

Processing interrupts

The positioning module can trigger a diagnostic interrupt. These interrupts are processed in an interrupt OB. If an interrupt is triggered without the corresponding OB being loaded, the CPU switches to STOP.

Enable the processing of the diagnostic interrupts as follows:

1. Select the module in HWCN.
2. Select "Edit" > Object Properties > Basic Parameters" to enable the diagnostic interrupt.
3. Save and compile the configuration.
4. Download the configuration to the CPU.

Overview of diagnostic interrupts

The following events and errors trigger a diagnostic interrupt:

- Operating error
- Machine data error
- Diagnostics errors

These errors are explained in detail in "Error classes (Page 10637)".

Reaction of the positioning module to an error with diagnostic interrupt

- The positioning is canceled.
- The synchronization is deleted in the case of the following diagnostic interrupts:
  - Front connector missing, external auxiliary voltage for encoder supply missing
  - Zero mark error detected, cable fault (5 V encoder signals)
  - Traversing range exceeded (indicated by an operating error)
  - Set process value cannot be executed (indicated by an operating error).
- With one exception, control signals START, DIR_P und DIR_M are no longer processed.
  Exception:
  In the case of an operating error it is still possible to jog in the direction of the operating range.
- Function switches and jobs will no longer be processed.

Positioning module detects an error ("incoming")

A diagnostic interrupt is "incoming", if at least one error is given. If not all errors are remedied, the remaining existing errors will be reported again as "incoming".

Sequence:

1. The positioning module detects one or more errors and generates a diagnostic interrupt. Either the "INTF" or "EXTF" LED is illuminated. The error is then entered in the diagnostic buffer.
2. The CPU operating system calls the diagnostic interrupt OB, e.g. OB 82.
3. You can evaluate the start info of OB 82.
4. The OB82_MOD_ADDR parameter indicates which module triggered the interrupt.
5. For additional information, call the ABS_DIAG (Page 7127) or ABS_DIAG_451 (Page 7131) instruction.

The positioning module detects the transition to the error-free state ("outgoing")

A diagnostic interrupt is then only "outgoing" if the last error on the module has been remedied.

Sequence:

1. The positioning module detects that all errors have been eliminated and triggers a diagnostic interrupt. The "INTF" or "EXTF" LED is no longer illuminated. The diagnostic buffer is not changed.
2. The CPU operating system calls the diagnostic interrupt OB, e.g. OB 82.
3. The OB82_MOD_ADDR parameter indicates which module triggered the interrupt.
4. Evaluate the OB82_MDL_DEFECT bit.
   If this bit is "0", there are no errors on the module. You can end your evaluation here.

Diagnostic interrupts in dependence on the CPU status

- Diagnostic interrupts are blocked by the positioning module when the CPU is in STOP mode.
- If not all of the pending errors are eliminated while the CPU is in STOP mode, the positioning module reports the remaining errors as "incoming" events again at the next CPU transition to RUN.
- If all pending errors are eliminated while the CPU is in STOP mode, the module does **not** report its error-free state with a diagnostic interrupt at the next CPU transition to RUN.

Error evaluation in the user program

Error response in the user program

In the user program you can respond selectively to errors. The following methods can be used:

- The RET_VAL return values of the integrated operations:
  This value is determined again each time the instruction is called.
  RET_VAL = -1 is a group display for a synchronous error in a job or when communicating with the module.
- An _ERR error bit belongs to each job. This is a group display for an error in the job or one of its predecessors in the job chain:
  The error bit is set for a write job and its subsequent jobs when a data error is reported by the module or a communication error occurs.
  With reading jobs the error bit is set for the job in question when a communication error occurs.
  The ABS_CTRL (Page 7125) or ABS_CTRL_451 (Page 7129) instruction resets the error bits once a job has been executed. These should however be cancelled by the user program in the case of an error assessment.
- The DATA_ERR checkback signal as a group display for an error that the module has detected during a write job. The signal will be established again with the next write job.
- The OT_ERR checkback signal (operator error) as a group display for an error that the module has detected when starting a traverse. The error must be acknowledged after the cause has been acknowledged with OT_ERR_A = 1.
- The DIAG checkback signal is set when the contents of the diagnostic buffer have changed. This signal can come along later than the signals DATA_ERR und OT_ERR.
- The communication error JOB_ERR includes the error code (Page 10636) in the case of a communication problem between the instruction and the module. The value is determined again after a job has been processed and is stored in the channel DB for the ABS_CTRL (Page 7125) or ABS_CTRL_451 (Page 7129) instruction in the channel DB and for the ABS_DIAG (Page 7127) or ABS_DIAG_451 (Page 7131) instruction in the diagnostics DB.
- The ABS_DIAG (Page 7127) or ABS_DIAG_451 (Page 7131) instruction for reading the diagnostic buffer of the module. Here you can discover the error causes for synchronous and asynchronous events.
- Diagnostic interrupts for the fast response to events in the diagnostic interrupt OB (e.g. OB 82).

Evaluation of a diagnostic interrupt in the user program

The positioning module sets the following entries in the local data of the diagnostic interrupt OB (e.g. OB 82). The errors will also be entered in the diagnostic buffer (error class 128 (Page 10644)).

Address	Name	Type	Comment
0.0	OB82_EV_CLASS	BYTE	Event class and identifiers: B#16#38: Outgoing event B#16#39: Incoming event
1.0	OB82_FLT_ID	BYTE	Error code (B#16#42)
2.0	OB82_PRIORITY	BYTE	Priority class: B#16#1A in RUN B#16#1C in START-UP
3.0	OB82_OB_NUMBR	BYTE	OB number (82)
4.0	OB82_RESERVED_1	BYTE	Reserved
5.0	OB82_IO_FLAG	BYTE	Input module: B#16#54
6.0	OB82_MDL_ADDR	INT	Logical base address of the module in which the error has occurred
8.0	OB82_MDL_DEFECT	BOOL	Module fault
8.1	OB82_INT_FAULT	BOOL	Internal error
8.2	OB82_EXT_FAULT	BOOL	External error
8.3	OB82_PNT_INFO	BOOL	Channel error
8.4	OB82_EXT_VOLTAGE	BOOL	Missing external auxiliary supply
...	Not used		
10.3	OB82_WTCH_DOG_FLT	BOOL	Time monitoring has responded
...	Not used		
12.0	OB82_DATE_TIME	DATE_AND_TIME	Date and time at which the OB was requested

Diagnostics buffer of the module

Diagnostic events

The module´s diagnostic buffer includes maximum 9 diagnostic events and is organized as a ring buffer.

A diagnostic event is written in the buffer if an "incoming" (error) message is detected. This can be a message, a synchronous error (data error, operator error) or even an asynchronous error (operational error and diagnostics error). From a cause of error several entries can occur as follow-up errors. Outgoing messages do not create any entries in the diagnostic buffer.

For each diagnostic event the following is given:

- Status (always incoming)
- Internal error
- External error
- Error class

- Error number
- Channel number
- Increment number (in the case of increment table errors)

When a diagnostic event is written in the diagnostic buffer, the DIAG = 1 checkback signal is set in all assigned channels.

With the ABS_DIAG (Page 7127) or ABS_DIAG_451 (Page 7131) instruction, the diagnostic buffer can be transferred as a whole into a data block (diagnostic DB) or displayed via the evaluation dialog of the configuration software. If the diagnostic buffer is read, the module sets the checkback signal DIAG = 0.

Note

If the diagnostic buffer is read simultaneously from the ABS_DIAG or ABS_DIAG_451 instruction and the error evaluation dialog, it is possible that a new diagnostic event is not detected by the program.

Data and structure of the diagnostic DB

Data and structure of the diagnostic DB

Note

You must not change any data that is not listed in this table.

The following table shows the structure of the diagnostic DB:

Address	Name	Type	Initial value	Comment
0.0	MOD_ADDR	INT	0	Module address
256.0	JOB_ERR	INT	0	Error number of the communication error
258.0	JOBBUSY	BOOL	FALSE	1 = job active
258.1	DIAGRD_EN	BOOL	FALSE	1 = definitely read diagnostics buffer
260.0	DIAG_CNT	INT	0	Number of valid entries in the list
262.0	DIAG[1]	STRUCT		Diagnostics data - latest entry
272.0	DIAG[2]	STRUCT		Diagnostics data - second entry
282.0	DIAG[3]	STRUCT		Diagnostics data - third entry
292.0	DIAG[4]	STRUCT		Diagnostics data - fourth entry
302.0	DIAG[5]	STRUCT		Diagnostics data - fifth entry
312.0	DIAG[6]	STRUCT		Diagnostics data - sixth entry
322.0	DIAG[7]	STRUCT		Diagnostics data - seventh entry
332.0	DIAG[8]	STRUCT		Diagnostics data - eighth entry
342.0	DIAG[9]	STRUCT		Diagnostics data - ninth entry

The diagram below shows the structure of a DIAG[n] diagnostic entry.

Address	Name	Type	Initial value	Comment
+0.0	STATE	BOOL	FALSE	0 = incoming event 1 = outgoing event
+0.1	INTF	BOOL	FALSE	1 = internal error
+0.2	EXTF	BOOL	FALSE	1 = external error
+2.0	FCL	INT	0	Error class: 1: Operating error 2: Operator errors 4: Data error 5: Machine data error 6: Incremental dimension table error 15: Alarms 128: Diagnostics errors
+4.0	FNO	INT	0	Error number
+6.0	CH_NO	INT	0	Channel number
+8.0	TRG_NO	INT	0	Incremental dimension number

List of JOB_ERR messages

JOB_ERR alarms

JOB_ERR (Hex)	JOB_ERR (Dec)	JOB_ERR (Int)	Meaning
80A0	32928	-32608	Negative acknowledgement when reading from the module. The module has been removed during the read operation - or the module is faulty.
80A1	32929	-32607	Negative acknowledge when writing to the module. The module has been removed during the write operation - or the module is faulty.
80A2 [1)]	32930	-32606	Protocol error at layer 2
80A3 [1)]	32931	-32605	Protocol error involving user interface/user
80A4 [1)]	32932	-32604	Communication on C bus interrupted
80B0	32944	-32592	Data record / job unknown.
80B1	32945	-32591	Incorrect length specification. FM_TYPE parameter in the channel DB is not correctly set for the module being used.
80B2	32946	-32590	The configured slot is not assigned.
80B3	32947	-32589	Actual module type dissimilar to the set module type.
80C0 [1)]	32960	-32576	The module does not yet have ready the data to be read.
80C1 [1)]	32961	-32575	The data of a same-named write job is not yet ready on the module.
80C2 [1)]	32962	-32574	The module is currently processing the possible maximum of jobs.
80C3 [1)]	32963	-32573	Required equipment (memory etc.) is currently in use.
80C4 [1)]	32964	-32572	Communication error
80C5 [1)]	32965	-32571	Distributed I/O unavailable.
80C6 [1)]	32966	-32570	Priority class cancellation (restart or background)
8522	34082	-31454	Channel DB or parameter DB too short. The data cannot be read from the DB. (write job)

JOB_ERR (Hex)	JOB_ERR (Dec)	JOB_ERR (Int)	Meaning
8532	34098	-31438	DB number of the parameter DB too high (write job)
853A	34106	-31430	Parameter DB does not exist. (write job)
8544	34116	-31420	Error with the n-th (n > 1) read access on a DB after an error has occurred. (write job)
8723	34595	-30941	Channel DB or parameter DB too short. The data cannot be written in the DB. (read job)
8730	34608	-30928	Parameter DB in the CPU is write-protected. The data cannot be written in the DB (read job).
8732	34610	-30926	DB number of the parameter DB too high (read job)
873A	34618	-30918	Parameter DB does not exist. (read job)
8745	34629	-30907	Error with the n-th (n > 1) write access on a DB after an error has occurred. (read job)

[1] Errors 80A2 to 80A4 and 80Cx are temporary, i.e., they can be rectified automatically after a waiting time.

Error classes

Operating error

Error class 1: Operating error

Operating errors are detected asynchronously to an operation / control. The operating errors bring about a cancellation of the positioning, apart from with error number 9, which causes the positioning to shut down.

The following table shows the possible operating errors and their respective causes:

No.	Meaning		Diagnostic interrupt
1	**Software limit switch start overrun**		Yes
	Cause	The actual value lies outside of the operating range.	
2	**Software limit switch end overrun**		Yes
	Cause	The actual value lies outside of the operating range.	
3	**Traversing range start overrun**		Yes
	Cause	Traversing range limit overrun (coordinates of the traversing range limits belong with the traversing range).	
4	**Traversing range end overrun**		Yes
	Cause	Traversing range limit overrun (coordinates of the traversing range limits also belong to the traversing range)	
5	**Error on target approach**		Yes
	Cause	Target range has not been reached within the monitoring time.	
6	**Standstill range exited**		Yes
	Cause	The actual value lies outside of the standstill range.	
7	**Positive feedback**		Yes

No.	Meaning		Diagnostic interrupt
	Cause	Actual value change > ½ standstill range in the wrong direction.	
8	**Missing or too slight actual value change**		Yes
	Cause	No actual value change or an actual value change against the set direction within the monitoring time.	
9	**Target overrun (FM 451)**		Yes
	Cause	The target has been overrun with "Set actual value on-the-fly".	
10	**Target range overrun**		Yes
	Cause	Target range has been overrun after target approach.	
11	**Changeover point switched incorrectly**		Yes
	Cause	Axis swinging in changeover point.	
12	**Switch-off point switched incorrectly**		Yes
	Cause	Axis swinging between switch-off / return point.	
13	**Start of target range switched incorrectly**		Yes
	Cause	Axis oscillating in the target range.	
14	**Change greater than half the rotary axis range**		Yes
	Cause	The velocity/frequency is too high or there are faulty actual value jumps.	
15	**Change greater than the rotary axis range**		Yes
	Cause	The velocity/frequency is too high or there are faulty actual value jumps.	
16	**Increment for increment number 252 cannot be transferred (FM 451)**		Yes
	Cause	The increment cannot be transferred.	
17	**Increment for increment number 252 cannot be approached (FM 451)**		Yes
	Cause	The distance between the current actual position and the specified increment is less than the changeover difference or switch-off difference.	
18	**Incorrect increment for increment number 252 (FM 451)**		Yes
	Cause	The increment lies outside of the operating range.	

Operator errors

Error class 2: Operator errors

Operator errors have been detected when changing the control signals in the user data area. The operator errors cause the positioning to shut down.

The following table shows the possible operator errors and their respective causes:

No.	Meaning		Diagnostic interrupt
1	**Impermissible operating mode**		No
	Cause	The chosen operating mode is inadmissible.	
3	**Inadmissible interface job**		No
	Cause	The selected signal is inadmissible in this operating mode.	
4	**Incorrect operating mode parameter**		No

No.	Meaning		Diagnostic interrupt
	Cause	• In "jog" mode the velocity specification is not equal to the rapid traverse or creep speed. • In "incremental approach" operating mode, the incremental dimension is a value other than 1 - 100, 252, 254, or 255	
5	**Start enable not available**		No
	Cause	Start enable not available upon start.	
7	**Target / target range lies outside of the operating range**		No
	Cause	Specified or calculated target is outside the software limit switches.	
8	**Axis parameters not assigned**		No
	Cause	Incorrect machine data or no machine data has been assigned for the axis.	
9	**Axis not synchronized**		No
	Cause	The "incremental approach" mode is only possible with an already synchronized axis.	
10	**Target/incremental distance cannot be positioned**		No
	Cause	The distance between the current actual position and the specified target is less than the switch-off difference.	
17	**Reference point approach not possible**		No
	Cause	An SSI encoder is connected.	
18	**Relative or absolute incremental approach is not possible**		No
	Cause	The increment is invalid.	
19	**Switch-off difference is less than ½ the target range with increment number 255**		No
	Cause	The switch-off difference for the increment number 255 is less than half the target range.	
20	**Inadmissible travel in the specified direction**		No
	Cause	The distance to the software limit switch is insufficient	

Data error

Error class 4: Data error

Operating errors are detected synchronously to an operation / control. Data errors do not result in an error response.

The following table shows the possible data errors and their respective causes:

No.	Meaning		Diagnostic interrupt
6	**Specified increment too great**		No
	Cause	The value lies outside of ±100 m or ±1000 m. The increment/target must not be greater than the traversing range. For a rotary axis, the coordinate must be greater than or equal to 0 and less than the end of rotary axis.	
10	**Faulty zero offset (FM 451)**		No

No.	Meaning		Diagnostic interrupt
	Cause	The zero offset is greater than ±100 m or ±1000 m. The software limit switches are positioned after the zero offset outside of the traversing range (100 m to +100 m or -1000 m to +1000 m). Rotary axis: The zero offset amount is greater than the end of the rotary axis.	
11	Incorrect actual value		No
	Cause	Linear axis: The coordinate lies outside the current (possibly shifted) software limit switches. Rotary axis: The coordinate is < 0 or greater than the end of rotary axis.	
12	Incorrect reference point		No
	Cause	Linear axis: The coordinate lies outside the current (possibly shifted) software limit switches. Rotary axis: The coordinate is < 0 or greater than the end of rotary axis.	
20	Not permissible to activate machine data		No
	Cause	There are no new, error-free machine data on the module.	
27	Unauthorized bit-coded setting		No
	Cause	Bits that are not used and not described here are not equal to 0.	
29	Inadmissible bit coding		No
	Cause	Unused and, in this case, unwritten bits are not 0.	
34	Not possible to cancel set actual value		No
	Cause	With an SSI encoder and a linear axis, the location of the actual value would lie outside the operating range after making the setting.	
36	Incorrect changeover difference with the increment number 255		No
	Cause	The value is outside the permitted numeric range of ±100 m or ±1000 m. For a rotary axis, the coordinate must be greater than or equal to 0 and less than the end of rotary axis.	
37	Incorrect switch-off difference with the increment number 255		No
	Cause	The value is outside the permitted numeric range of ±100 m or ±1000 m. The switch-off difference must be less than the changeover difference.	
107	Axis parameters not assigned		No
	Cause	On the axis there is either no machine data or it has not been enabled.	
108	Axis not synchronized		No
	Cause	One of the "set actual value" and "set actual value on-the-fly" jobs has been triggered even though the axis is not synchronized.	

Machine data error

Error class 5: Machine data error

The diagnostic interrupt is only triggered in the event of a faulty system data block (SDB). Machine data errors do not result in an error response.

The following table shows the possible machine data errors and their respective causes:

No.	Meaning		Diagnostic interrupt
5	**Error in hardware interrupt setting**		Yes
	Cause	You have attempted to select a hardware interrupt that the module does not support.	
6	**Incorrect minimum edge distance (FM 451)**		Yes
	Cause	You have entered a value < 0 or > 10^9 µm as the minimum edge-to-edge distance.	
7	**Incorrect system of units**		Yes
	Cause	The value for the system of units lies outside of the permissible range of 1 to 4 and 6.	
8	**Incorrect axis type**		Yes
	Cause	You have entered neither 0 nor 1 as an axis type.	
9	**Incorrect end of rotary axis**		Yes
	Cause	The value for the end of the rotary axis is outside the valid range from 1 to 10^9 µm or 1 to 10^8 µm (depending on the resolution).	
10	**Incorrect encoder type**		Yes
	Cause	The value for the encoder type lies outside of the permissible range of 1 to 4.	
11	**Incorrect distance per encoder revolution**		Yes
	Cause	The value for distance per encoder revolution lies outside the permissible range of 1 to 10^9 µm (irrespective of the resolution).	
13	**Incorrect number of increments per encoder revolution** (see "Increments per encoder revolution (Page 10616)")		Yes
14	**Incorrect number of revolutions** (see "Number of encoder revolutions (Page 10618)")		Yes
15	**Incorrect transmission rate**		Yes
	Cause	You have specified a value for the transmission rate that lies outside the permissible range of 0 to 3.	
16	**Incorrect reference point coordinate**		Yes
	Cause	The coordinate lies outside the range from -100 m to +100 m or -1000 m to +1000 m (depending on the resolution). Linear axis: The coordinate lies outside of the operating range. Rotary axis: The coordinate is greater than the end of the rotary axis or < 0.	
17	**Incorrect absolute encoder adjustment**		Yes
	Cause	SSI navigator: The value of the absolute encoder adjustment is not in the encoder range: (increments per encoder revolution x number of revolutions - 1).	
18	**Incorrect type of reference point approach**		Yes
	Cause	You have specified a value outside of the permissible value quantities of 0, 1, 2 and 3.	
19	**Incorrect counting direction**		Yes
	Cause	You have specified a value outside of the permissible value quantities of 0 and 1.	
20	**It is not possible to monitor the hardware**		Yes
	Cause	You have set frame error monitoring to "FALSE" in the parameter DB. The monitoring of missing pulses is not possible with encoders being used. Deactivate the MON_PULSE parameter.	
21	**Incorrect software limit switch start**		Yes

No.	Meaning		Diagnostic interrupt
	Cause	Linear axis: The software limit switch start is outside the traversing range (-100 m to +100 m or 1,000 m to +1000 m, depending on the resolution). Linear axis: The software limit switch start (including any existing zero offset) is less than -100 m or -1000 m (depending on the resolution).	
22	Incorrect software limit switch end		Yes
	Cause	Linear axis: The software limit switch end is outside the traversing range (-100 m to +100 m or 1,000 m to +1000 m, depending on the resolution) or is less than the software limit switch start. Linear axis: The software limit switch end (including any existing zero offset) is greater than +100 m or +1000 m (depending on the resolution).	
23	Incorrect maximum velocity		Yes
	Cause	The data that is not listed in the parameter DB must be 0.	
24	Incorrect target range		Yes
	Cause	Linear axis: Range between 0 and 100 m or 0 and 1000 m depending on resolution. Rotary axis: Range greater than the end of rotary axis.	
25	Incorrect monitoring time		Yes
	Cause	The value for the monitoring time lies outside the permissible range from 0 to 100 000 ms.	
26	Incorrect standstill range		Yes
	Cause	Linear axis: Range between 0 and 100 m or 0 and 1000 m, depending on the resolution. Rotary axis: Range greater than the end of rotary axis.	
127	Incorrect standstill velocity		Yes
	Cause	The value for the standstill velocity lies outside the permissible range from 0 to 100 000 µm/min.	
128	Incorrect control mode		Yes
	Cause	You have specified a value for the control mode that lies outside the permissible range of 1 to 4.	
129	Incorrect start velocity for the reference point approach		Yes
	Cause	You have entered neither 0 nor 1 as a start velocity.	
130	Incorrect changeover difference in the direction +		Yes
	Cause	Linear axis: Range between 0 and 100 m or 0 and 1000 m (depending on resolution). Rotary axis: Range greater than the end of rotary axis and smaller than ½ of the target range.	
131	Incorrect changeover difference in the - direction		Yes
	Cause	Linear axis: Range between 0 and 100 m or 0 and 1000 m (depending on the resolution). Rotary axis: Range greater than the end of rotary axis and smaller than ½ of the target range.	
132	Incorrect switch-off difference in the direction +		Yes
	Cause	The switch-off difference is greater than the switch-over difference plus, less than ½ of the target range, or is outside the permissible range between 0 to 100 m or 1000 m (depending on the resolution)	
133	Incorrect switch-off difference in the - direction		Yes

No.	Meaning		Diagnostic interrupt
	Cause	The switch-off difference is greater than the switch-over difference minus, less than ½ of the target range, or is outside the permissible range between 0 to 100 m or 1000 m (depending on the resolution)	
200	Incorrect resolution		Yes
	Cause	You have specified a resolution < 0.1 µm/pulse or >1000 µm/pulse. You have specified a distance per encoder revolution and a number of pulses per encoder revolution that results in a resolution of < 0.1 or > 1000.	
201	Encoder does not fit the operating range / rotary axis range		Yes
	Cause	SSI encoders and rotary axes: The encoder does not precisely cover the rotary axis range. Linear axis: The encoder does not cover at least the operating range (including software limit switches).	

Incremental dimension table error

Error class 6: Incremental dimension table error

The incremental dimension table errors do not result in an error response.

The following table shows the possible incremental dimension errors and their respective causes:

No.	Meaning		Diagnostic interrupt
6	Specified increment in the increment table too great		No
	Cause	The value lies outside of ±100 m or ±1000 m. The increment/target must not be greater than the traversing range. For a rotary axis, the coordinate must be greater than or equal to 0 and less than the end of rotary axis.	

Alarms

Error class 15: Alarms

Alarms do not result in an error response.

The following table shows the possible alarms and their respective causes:

No.	Meaning		Diagnostic interrupt
1	Start of parameter assignment		No
	Cause	The module has detected a parameter assignment via a system data block.	
2	End of parameter assignment		No
	Cause	The module has processed the parameter assignment via a system data block without errors.	

No.	Meaning		Diagnostic interrupt
11	**Distance to change-over point too short**		No
	Cause	The hardware response times cannot be adhered to because the distance between the switching points is too short.	
12	**Distance to return point too short**		No
	Cause	The hardware response times cannot be adhered to because the distance between the switch points is too short	
14	**Distance to cut-off point too short**		No
	Cause	The hardware response times cannot be adhered to because the distance between the switch points is too short	
15	**Distance to start of target range too short**		No
	Cause	The hardware response times cannot be adhered to because the distance between the switch points is too short	

Diagnostic errors

Error class 128: Diagnostic errors

The following table shows the possible diagnostic errors and their respective causes, effects and remedies.

No.	Meaning		Diagnostic interrupt
4	**Missing external auxiliary supply**		Yes
	Cause	• External 24 V auxiliary voltage is not connected or has failed • The fuse on the module is defective • Undervoltage • Ground wire break • Short circuit (e.g. at the encoder connected)	
	Effect	• The positioning is cancelled on all channels • Output switch-off • Synchronization is cancelled for incremental encoders when the auxiliary voltage for the encoder supply is missing • Parameters have not been assigned to the positioning module • Start enable is deleted	
	Remedy	Make sure that the 24 V connection is correct (if the 24 V connection is correct, then the module is defective)	
5	**Front connector is missing (FM 451)**		Yes
	Cause	Front connector of the positioning module is not inserted	
	Effect	• No external 24 V auxiliary supply • Module is not ready	
	Correction	Plug the front connector into the positioning module	
51	**Time monitoring activated (watchdog)**		Yes

No.	Meaning		Diagnostic interrupt
	Cause	• Strong interference on the positioning module • Error in the positioning module	
	Effect	• Module is reset • Turn off all outputs • If, after resetting the module, no module fault is detected, the module is ready for operation again • The module reports the expired watchdog with "incoming" and "outgoing"	
	Correction	• Remedying interferences • Contact the relevant sales department which will require details of the exact circumstances leading to the error • Replace the positioning module	
144	**Encoder wire break**		Yes
	Cause	• The encoder cable has sheared off or is not plugged in • Encoders without cross signals • Incorrect pin assignment • Cable length too long • Encoder signals short-circuited • Encoder signal edge fault • Maximum input frequency of the encoder input has been exceeded • Sensor voltage supply failure	
	Effect	• The positioning is canceled • Output switch-off • Delete the synchronization with incremental encoders • Start enable is deleted	
	Correction	• Check encoder cable • Comply with encoder specification • Monitoring can be temporarily hidden at the responsibility of the operator using the parameter assignment dialog • Comply with the technical specifications of the module	
145	**Absolute encoder frame error**		Yes
	Cause	The frame traffic between the positioning module and the absolute encoder (SSI) is incorrect or interrupted: • Encoder cable cut or not plugged in • Incorrect encoder type • The encoder is incorrectly set (programmable encoder) • The message length has been incorrectly specified • The encoder is supplying incorrect values (encoder is faulty) • Interference on measuring system cable • Chosen transmission rate is too high • Monoflop time of the encoder is greater than 64 µs	
	Effect	• The positioning is canceled • Output switch-off • Start enable is deleted	

No.	Meaning		Diagnostic interrupt
	Correction	• Check encoder cable • Check encoder • Check the frame traffic between the encoder and positioning module	
146	**Missing pulses of incremental encoder**		Yes
	Cause	• Encoder monitoring has established a missing pulse • Incorrect entry of the number of increments per encoder revolution • Encoder faulty: does not return the specified number of pulses • incorrect or no zero mark • Crosstalk on the encoder cable	
	Effect	• The positioning is canceled • Output switch-off • Start enable is deleted	
	Correction	• Enter the correct number of increments per encoder revolution (programming dialog) • Check the encoder and encoder cable • Comply with the shielding and grounding directives • Monitoring can be temporarily hidden at the responsibility of the operator using the parameter assignment dialog	

11.4.2.7 Using ET 200S 1PosU

Terminal Assignment of the 1PosU

Wiring rules

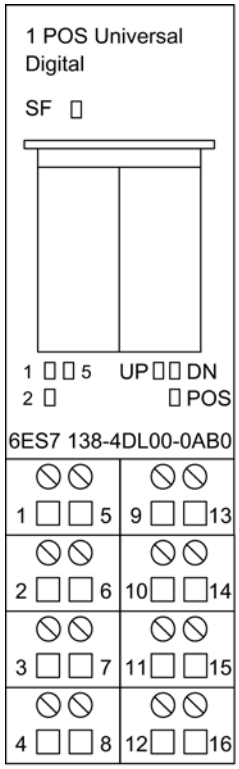
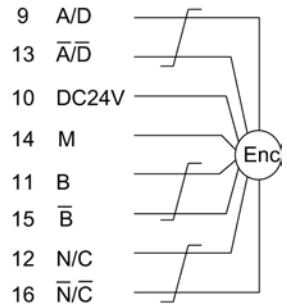
If a position encoder with 5 V differential signals is used, the wires to the terminals 9 and 13, the terminals 12 and 16, as well as at incremental encoders the wires to the terminals 11 and 15 have to be in twisted pairs and shielded.

If an incremental encoder with 24 V signals is used, the wires to the terminals 9, 11 and 12 have to be shielded.

The shield must be supported at both ends. To do this, use the shield contact (see the appendix of the ET 200S Distributed I/O System (http://support.automation.siemens.com/WW/view/en/1144348) operating instructions).

Terminal assignment

The following table shows the terminal assignment of 1PosU.

View	Terminal assignment	Remarks			
		Connection of the Switches and the Drive: Terminals 1 to 8		**Connection of the position encoder with 5 V differential signals or 24 V signals: Terminals 9 to 16**	
		1: IN0	Limit switch minus	9: A / D	Track A / Data from the SSI encoder
		5: IN1	Limit switch plus	13: /A / /D	
		2: IN2	Reducing cam; latch signal	10: 24 V DC	Power supply for the position encoder
		6: 24 V DC	Supply for the switches	14: M	
		3: OUT0	Travel minus or rapid feed	11: B	Track B
		7: 2L+	Load voltage infeed for OUT0, OUT1 and OUT2	15: /B	
		4: OUT1	Travel plus or creep feed	12: N / C	Track N / SSI clock (clock line)
		8: OUT2	Rapid/creep feed and travel plus/ minus	16: /N / /C	

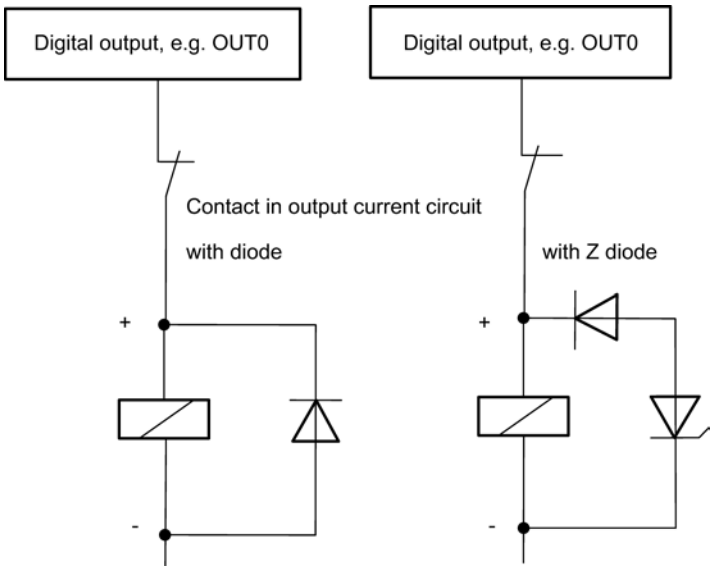
Connection of Relays and Contactors to the Digital Outputs

Note

Direct connection of inductivities (such as relays and contactors) is possible without external circuiting. If SIMATIC output circuits can be deactivated by additionally installed contacts (for example relay contacts), you have to provide additional overvoltage protection devices at inductivities (see the following example for overvoltage protection).

Overvoltage Protection Example

The following figure shows an output circuit that requires additional overvoltage protection devices. Direct-current coils are wired with diodes or Zener diodes.



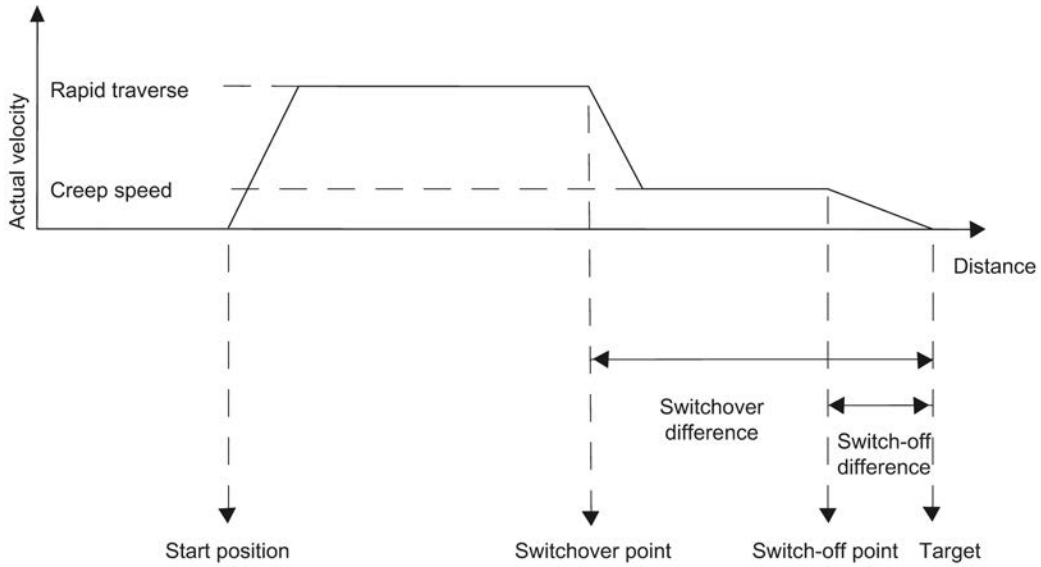
Basics of controlled positioning by means of rapid/creep feed

Positioning process

From the start position, the target is approached at high velocity (rapid traverse). At a preset distance from the target (changeover point), there is a change to a lower velocity (creep speed). Shortly before the axis reaches the target, again at a preset distance from the target, the drive is switched off (switch-off point).

The drive is controlled via digital outputs for rapid traverse or creep speed and the appropriate direction.

The following figure shows the switch points and switch differences. To facilitate understanding, the change in velocity is illustrated over the path traversed.



Definitions

The following tables contains the basic terms for controlled positioning using rapid traverse / creep speed.

Term	Description
Working range	defines the range which you set for a particular job by means of the hardware limit switches. At an SSI encoder, the operating range is also limited by the range covered by the SSI encoder. You enter the encoder range in the parameters for: • Pulses per encoder revolution and • revolutions. Encoder range = pulses per encoder revolution × revolutions Maximum operating range: • Linear axis max. 0 to (encoder range – 1) • Rotary axis from 0 to (encoder range - 1) At an incremental encoder the operating range is limited to: • Max. 0 to 16,777,215 increments at a linear axis • 0 to the programmed end of the rotary axis
Changeover difference	defines the distance from the destination at which the drive is switched over from rapid traverse to creep speed.
Changeover point	defines the position at which the drive is switched over from rapid traverse to creep speed.
Switch-off difference	defines the distance from the destination at which the drive is switched off. If the switch-off difference ≥ the changeover difference, there is no changeover point. There is no change from rapid traverse to creep speed.
Switch-off point	defines the position at which the drive is switched off. The 1PosU reports the end of the traverse at this point.

Term	Description
Start position	defines the position of the drive within the operating range from which the run is started. If the start position is within the switch-off difference, the drive is not triggered. The 1PosU reports the end of the traverse at this point. If the start position is within the changeover difference, the traverse is only executed at creep speed.
Target	defines the absolute or relative position of the axis approached during positioning. The target is the position to be reached on an axis during a traverse. In the case of an absolute traverse, you specify the target directly by means of your control program. In the case of a relative traverse, the destination is calculated from the start position and the distance specified in the control program. If you want to find out how accurately you have reached the destination, you have to compare the actual value with the position specified.
Linear axis	defines the axis type with a limited operating range. It is limited by the following: • the encoder range • the displayable numeric range (0 to 16 777 215 increments) • hardware limit switches
Rotary axis	defines the axis type with an infinite operating range. This includes resetting the axis position to 0 after one rotation (assigned parameter end of a rotary axis at an incremental encoder or parameterized encoder range at an SSI encoder).
Minus direction	If the drive moves in the minus direction, the actual value displayed is decreased.
Plus direction	If the drive moves in the plus direction, the actual value displayed is increased.

Functions of 1PosU

Overview of the Functions

Overview

The 1PosU offers you the following functions for moving your axis:

- Stop (Page 10656)
- Inching (Page 10656)
- Referance point approach (Page 10659)
- Relative Positioning (Page 10664)
- Absolute Positioning (Page 10666)

In addition to the different types of motion, the 1PosU also offers functions for:

- Cancelling JOB processing (Page 10669)
- Setting of Actual Value (Page 10670)
- Move Encoder Range (Page 10672)
- Change Switch-Off Difference (Page 10674)

- Change Changeover Difference (Page 10675)
- Reference Signal Evaluation (Page 10676)
- Latch Function (Page 10678)
- Setting the Monitoring of the Direction of Rotation (Page 10679)
- Display Current Values (Page 10681)
- Error detection / diagnostics (Page 10683)
- Behavior in CPU/Master-STOP (Page 10689)

Parameters

Define the variables that depend on the drive, axis, and encoder uniquely in the parameters.

Dosing mode

With incremental encoders, you can use the 1PosU for dosing. You set the dosing mode in the parameters. In dosing mode the 1PosU only evaluates the encoder signal A (/A). The actual value is incremented at each positive edge.

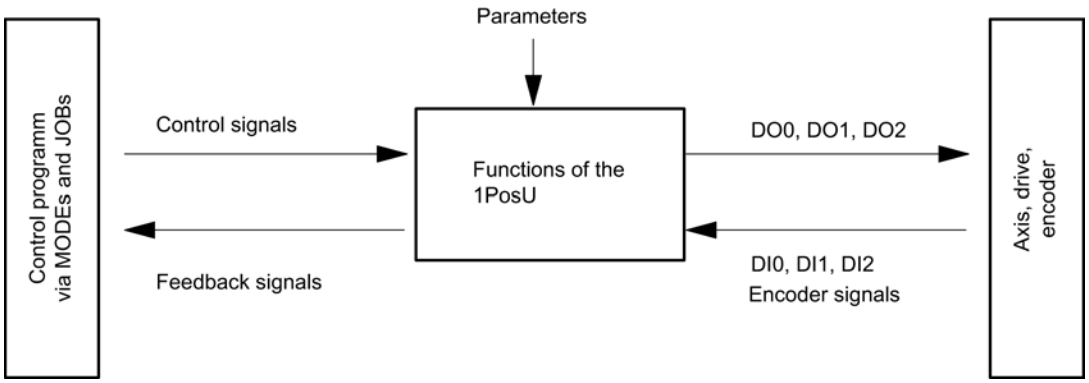
In dosing mode, only the "Inching" and "Relative positioning" functions are available for controlling the digital outputs.

The dosing function itself is triggered using the "Relative positioning" function. The dosing quantity is specified during starting by means of the control interface (distance).

At every start the actual value is set to 0 and the digital outputs are controlled as a function of the changeover and switch-off difference.

How the 1PosU works

The following figure shows how the 1PosU works.



Interfaces to the Control Program and the Axis

To execute the function, the 1PosU has digital inputs as an interface to the axis, encoder signals for the connection to an encoder and digital outputs to control the drive.

Starting motion types (MODEs)

You can control and monitor the types of motion (MODEs) with your control program using control signals and checkback signals.

Starting MODEs

The following table shows how you start MODEs and how the 1PosU reacts to them.

What You Do	1PosU reactions
Provide the control interface (Page 10707) with the data required depending on the MODE. Check whether the feedback bit (Page 10709) is POS_ACK = 0.	
Switch the START control bit from 0 to 1.	The 1PosU sets the POS_ACK = 1 and POS_DONE = 0 feedback bit. You can tell by this that the start has been detected by 1PosU and when POS_ERR = 0, the MODE is executed. When POS_ERR = 1, the MODE is not executed.
Switch the START control bit from 1 to 0.	The 1PosU sets the POS_ACK = 0 feedback bit.
	In the case of stopping, the reference point traverse and absolute and relative positioning, the 1PosU sets the POS_DONE = 1 feedback bit when the MODE is terminated without errors. When POS_ERR = 1, the MODE is terminated with errors.
You can start a new MODE only if POS_ACK = 0. If you start when a MODE is running, the 1PosU takes on the new motion and executes a change of direction, if necessary.	

The following figure shows the interaction of the control signals and checkback signals when starting the MODEs.

Activating further functions JOBs)

You can control and monitor further functions (JOBs) with your control program using control signals and checkback signals.

Activating JOBs

The following table shows how you start JOBs and how the 1PosU reacts to them.

What You Do	1PosU reactions
Provide the control interface (Page 10707) with the data required depending on the JOB. Check whether the feedback bit (Page 10709) is JOB_ACK = 0.	
Switch the JOB_REQ control bit from 0 to 1.	The 1PosU sets the JOB_ACK = 1 feedback bit. You can tell by this that the start has been detected by 1PosU and when JOB_ERR = 0, the JOB is executed. • With "Evaluate reference signal", the 1PosU sets the SYNC = 0 feedback bit simultaneously. • In the case of the latch function, the 1PosU sets the LATCH_DONE = 0 feedback bit simultaneously. • All the other JOBs are then executed. When JOB_ERR = 1, the JOB is not executed.
Switch the JOB_REQ control bit from 1 to 0.	The 1PosU sets the JOB_ACK = 0 feedback bit.
	When a reference signal is evaluated, the 1PosU sets the SYNC = 1 feedback bit when the function has been executed. With "Latch", the 1PosU sets the LATCH_DONE = 1 feedback bit when the function has been executed.
You can only activate a new JOB if JOB_ACK = 0.	

The following figure shows the interaction of the control signals and checkback signals when starting the JOBs.

Axis, Drive and Encoder

Evaluation of the Encoder Signals

The 1PosU evaluates the signals supplied by the encoder differently depending on the encoder type:

- SSI encoder:
  The 1PosU evaluates the encoder value supplied by the SSI encoder directly in increments and forms the actual value in increments (actual value = encoder value).
  The actual value lies in the encoder range from 0 ... (number of rotations * number of increments) – 1. The 1PosU generates an high limit violated / low limit violated of the actual value at the limits of the encoder operating range.
- Incremental encoder:
  The 1PosU evaluates the pulses supplied by the position encoder four times and adds them up direction-specifically to form the actual value. You must take the quadruple evaluation into account when you make settings for paths in the parameters and in the control and feedback interfaces:
  1 pulse of the incremental encoder corresponds to 4 increments of the 1PosU.
  The actual value lies in the operating range of 0 - 16 777 215 increments. The 1PosU generates an high limit violated or low limit violated of the actual value at the limits of the operating range.
- Incremental encoder in dosing mode:
  The 1PosU only evaluates the positive edges of the encoder A signal and adds them up to form the actual value.
  The actual value lies in the operating range of 0 - 16 777 215 increments. The 1PosU creates an high limit violated of the actual value at the high limit of the operating range.

Controlling the Drive

The drive is controlled using the 3 digital outputs of the 1PosU.

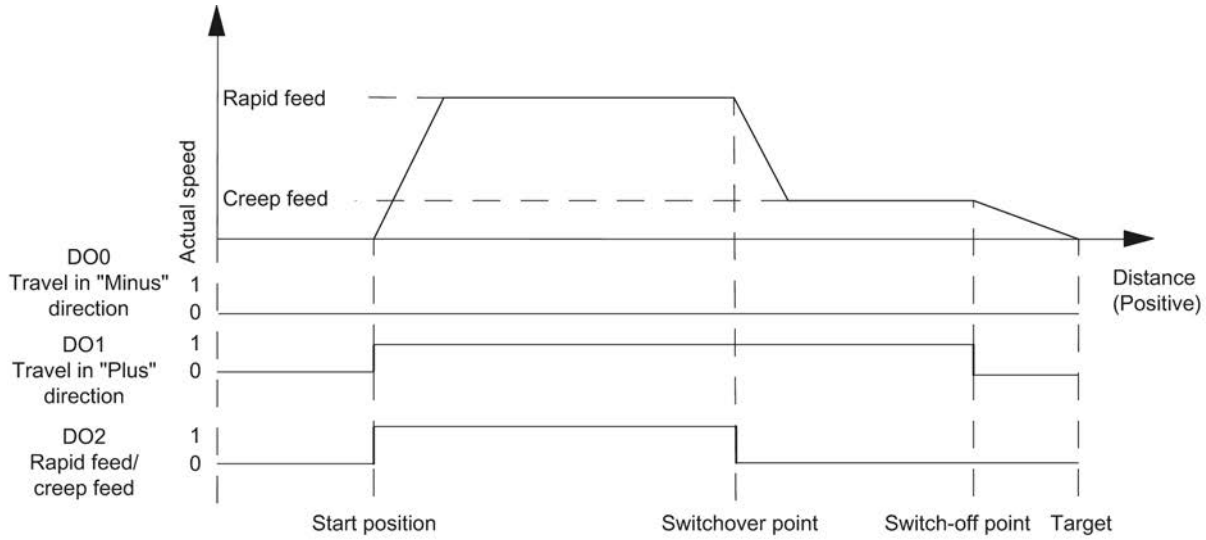
You can select the velocity with the SPEED control bit (SPEED = 0 is creep speed; SPEED = 1 is rapid traverse). You can also change the speed during the run.

You can bring about a change in direction (Page 10696) with the "T_{min} direction change" parameter.

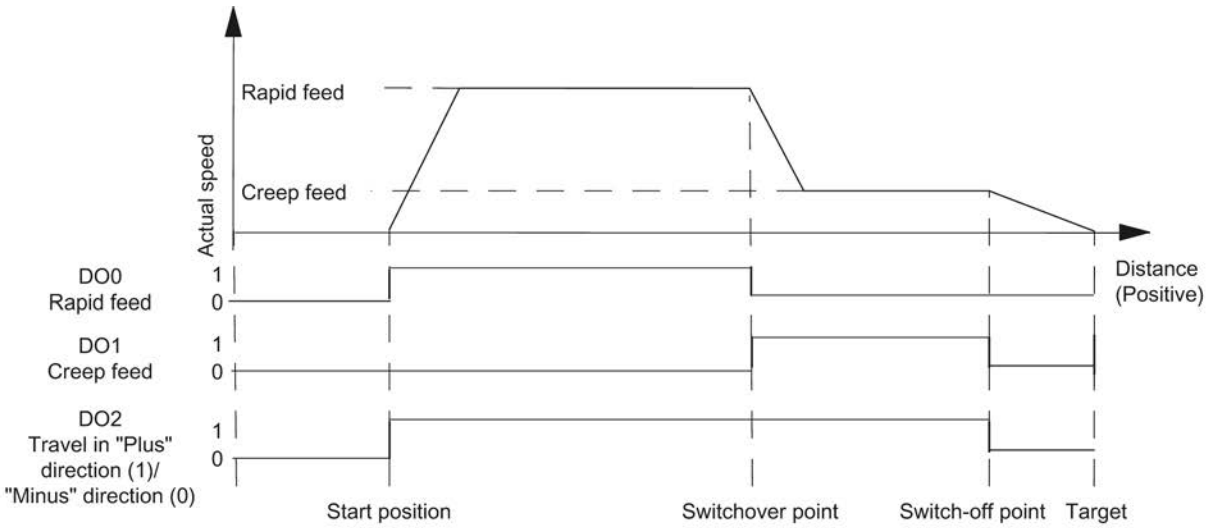
You can read the status of the respective output from the feedback interface (Page 10709) (DO 0, DO 1 and DO 2).

The function of the digital outputs depends on the control mode:

The following figure shows the states of the digital outputs with **control mode 0**.



The following figure shows the states of the digital outputs with **control mode 1**.



Effect of the Directional Enables

Description

You enable the digital outputs directionally using the DIR_M and DIR_P control bits.

- With DIR_M = 1, you can traverse in a minus direction.
- With DIR_P = 1, you can traverse in a plus direction.

Interrupting and Continuing the Run

If you reset the relevant directional enable during a run, the motion of the axis is halted, all 3 digital outputs are set to 0, and the run is interrupted.

If you set the relevant directional enable again, the run is continued.

Stop (MODE 0)

Stop

If you start MODE 0, then the 1PosU stops the current traverse, all 3 digital outputs are set to "0" and the traverse is terminated (POS_ERR = 0, POS_DONE = 1).

A traverse terminated with MODE 0 cannot be continued. To put the axis into motion again, you start a new MODE.

Control signals

The following table shows the data in the control interface which is relevant for "Stop".

Address	Assignment					
Byte 0	Bit 0.7 … 0.4:					
	Bit	7	6	5	4	
		0	0	0	0	MODE 0 = Stop
	Bit 0: START					

Checkback signals

The following table shows the data in the control interface (Page 10709) relevant for "Stop".

Address	Assignment
Byte 0	Bit 2: POS_DONE Bit 1: POS_ERR Bit 0: POS_ACK

See also

1PosU control interface (Page 10707)

Inching (MODE 1)

Inching

You use "Inching" to control the drive directly in a particular direction using the DIR_M or DIR_Pcontrol bits (Page 10655).

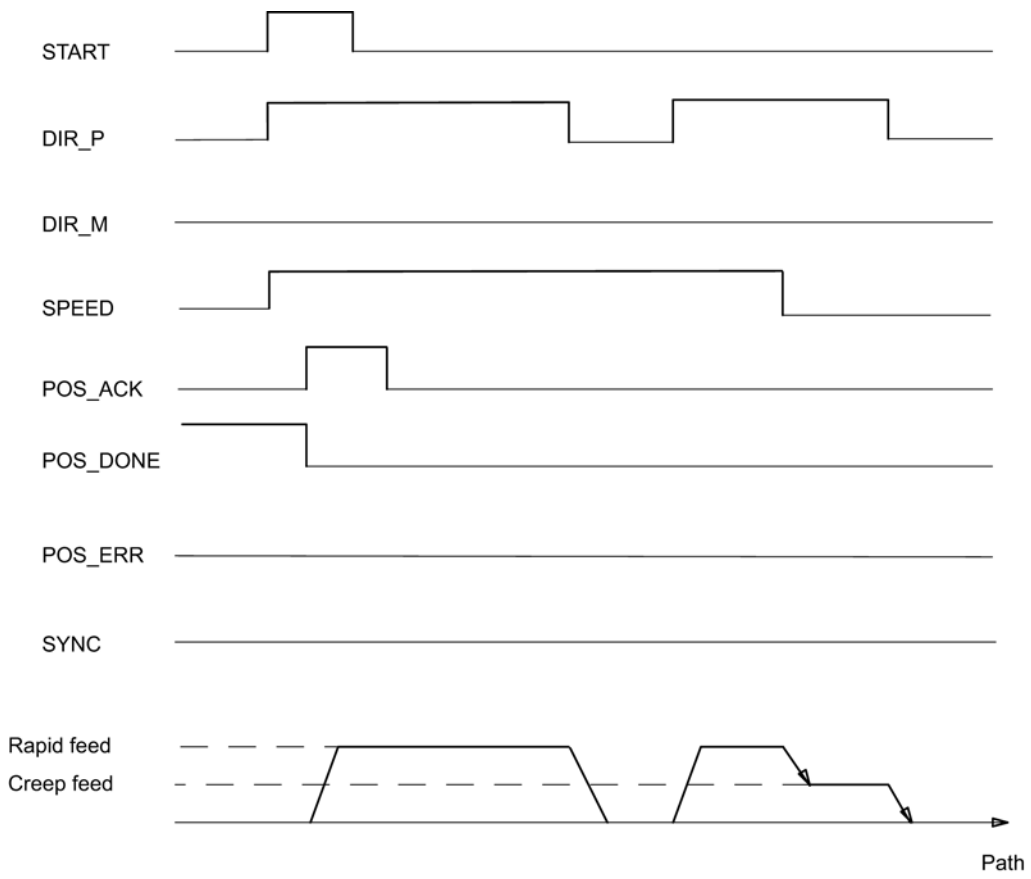
If you start MODE 1 then the 1PosU moves the drive with the specified velocity (SPEEDcontrol bit (Page 10707)) in the specified direction (DIR_M or DIR_Pcontrol bits (Page 10707)).

You stop the drive by setting the DIR_M = 0 and DIR_P = 0 control bits.

A change of direction (Page 10696) is executed after the time T_{min} elapses.

You can also activate jogging on an unsynchronized axis (feedback bit SYNC = 0) or when there is a pending encoder error (feedback bit ERR_ENCODER = 1) or without an encoder connected.

The figure below shows the sequence of the "Inching" motion type.



Control signals

The following table shows the data in the control interface (Page 10707) which is relevant for "Inching".

Address	Assignment					
Byte 0	Bit 0.7 … 0.4:					
	Bit	7	6	5	4	MODE 1 = Inching

Address	Assignment					
		0	0	0	1	
	Bit 3: SPEED (SPEED = 0 is creep speed; SPEED = 1 is rapid traverse) Bit 2: DIR_M Bit 1: DIR_P Bit 0: START					

Checkback signals

The following table shows the data in the control interface (Page 10709) which is relevant for "Inching".

Address	Assignment
Byte 0	Bit 2: POS_DONE Bit 1: POS_ERR Bit 0: POS_ACK
Bytes 1 to 3	Actual value: • at the incremental encoder: – Linear axis: 0 … 16 777 215 – Rotary axis: 0 … end of the rotary axis – 1 • at the SSI encoder: – 0 … encoder range – 1

Causes of the POS_ERR error

You must find out the causes of errors with JOB 15 (Display current values (Page 10681)).

The following table shows the error numbers relevant for "Inching" with the causes of the errors and possible remedies.

Error Number	Cause	Remedy
2	ERR_2L+ is displayed	Check the load voltage (2L+) at terminal 7.
5	The limit switch that lies in the direction in which the drive is moved is active	Check your switches and the wiring as well as the "DI0 limit switch minus (Page 10693)" and "DI1 limit switch plus (Page 10693)" parameters.
7	Inching: DIR_P = 1 and DIR_M = 1	
13	Direction of rotation of the drive and the encoder varies	Check the wiring of the drive and the encoder as well as the "Reversal of the direction of rotation (Page 10692)" parameter.
15	In dosing mode: DIR_M = 1	

Reference point approach (MODE 3)

Reference point approach (MODE 3)

Reference point approach

A reference point run can only be started at incremental encoders and non-activated dosing mode.

With reference point approach, you synchronies the axis by using external referencing signals. You can use either the 3 digital inputs or the zero mark (Page 10661) as a reference signal.

You can assign parameters to the digital inputs DI0 (minus limit switch), DI1 (plus limit switch) and DI2 (reducing cam) as NC or NO contacts.

Provide the control interface (Page 10707) with the reference point coordinates and start MODE 3. The 1PosU sets the SYNC = 0 checkback signal, moves the drive at the set velocity (SPEED control bit) in the programmed start direction and searches for the reference signal. The 1PosU automatically executes the required change of direction at the limit switches or the reducing cam.

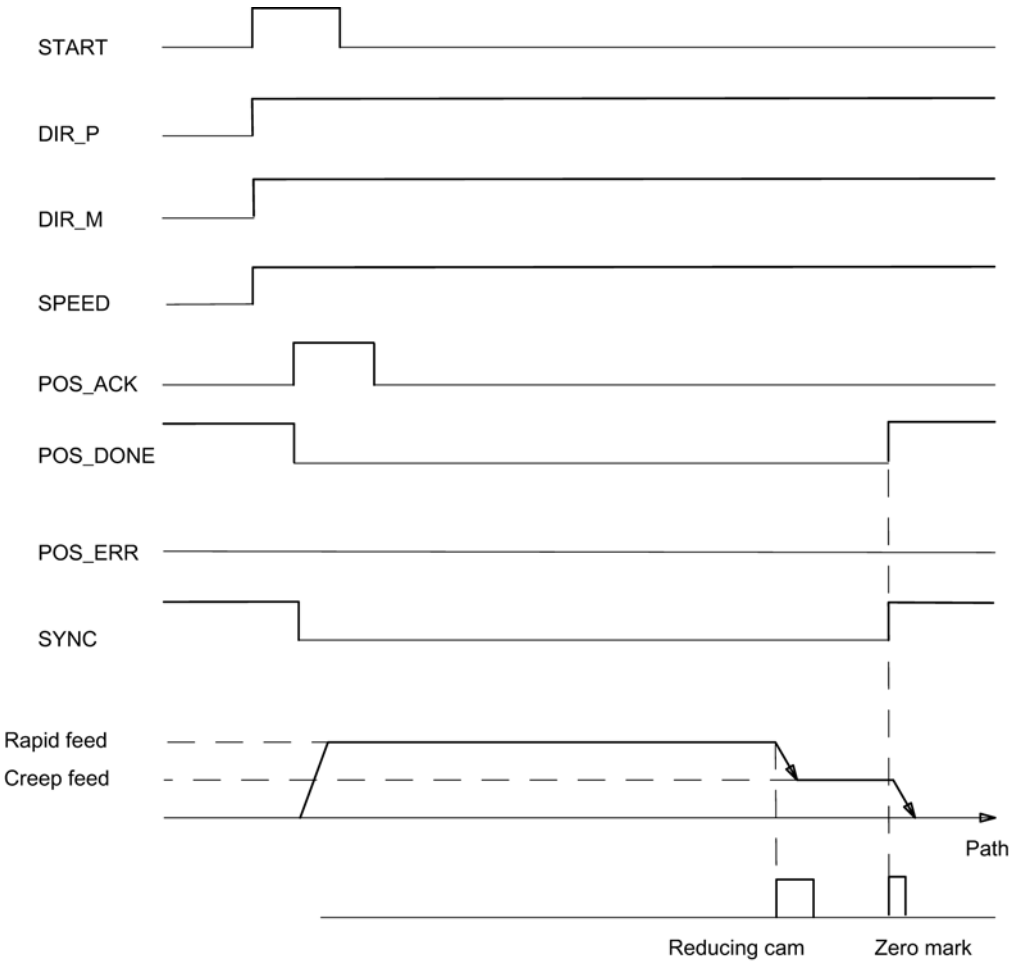
Set the necessary directional enables (Page 10655) (DIR_M, DIR_P) to ensure that the drive is controlled.

When the 1PosU detects the assigned parameter reference signal, it then controls the drive at creep speed in the referencing direction. This is controlled by the "Reference signal" and "Reference switch" parameters.

<table>
<tr><th></th><th>Reference switch: Reduction cam towards minus</th><th>Reference switch: Reduction cam towards plus</th><th>Reference switch: Limit switch minus</th><th>Reference switch: Limit switch plus</th></tr>
<tr><td>Reference signal: Reference switch and zero mark</td><td rowspan="2">Minus referencing direction</td><td rowspan="2">Plus referencing direction</td><td rowspan="2">Plus referencing direction</td><td rowspan="2">Minus referencing direction</td></tr>
<tr><td>Reference signal: Reference switch</td></tr>
<tr><td>Reference signal: Zero mark</td><td colspan="4">The referencing direction is not defined. The axis is synchronized at the next zero mark.</td></tr>
</table>

After the reference signal has been traversed, the axis is synchronized. The 1PosU sets the checkback signal SYNC = 1 and assigns the reference point coordinates to the actual value.

The following figure shows the sequence of the "Inching" motion type.



Control signals

The following table shows the data in the control interface (Page 10707) which is relevant for "Reference point approach".

Address	Assignment					
Byte 0	Bit 0.7 … 0.4:					
	Bit	7	6	5	4	
		0	0	1	1	MODE 3 = Reference point approach
	Bit 3: SPEED (SPEED = 0 is creep speed; SPEED = 1 is rapid traverse) Bit 2: DIR_M Bit 1: DIR_P Bit 0: START					
Bytes 1 to 3	Reference point coordinates: • Linear axis: 0 … 16 777 215 • Rotary axis: 0 … end of the rotary axis – 1					

Checkback signals

The following table shows the data in the control interface (Page 10709) which is relevant for "Reference point approach".

Address	Assignment
Byte 0	Bit 3: SYNC Bit 2: POS_DONE Bit 1: POS_ERR Bit 0: POS_ACK
Bytes 1 to 3	Actual value: • Linear axis: 0 … 16 777 215 • Rotary axis: 0 … end of the rotary axis – 1

Causes of the POS_ERR error

You must find out the causes of errors with JOB 15 (Display current values (Page 10681)).

The following table shows the error numbers relevant for "Reference point approach" with the causes of the errors and possible remedies.

Error number	Cause	Remedy
1	Impermissible MODE in dosing operation	
2	ERR_2L+ is displayed	Check the load voltage (2L+) at terminal 7.
3	ERR_ENCODER is displayed	Check the encoder wiring
10	Reference point approach: Reference point coordinate ≥ end of the rotary axis	
11	No reference signal found up to the limit switch or between the limit switches	Check your switches, the encoder and the wiring
13	Direction of rotation of the drive and the encoder varies	Check the wiring of the drive and the encoder as well as the "Reversal of the direction of rotation (Page 10692)" parameter.

Profile of a reference point approach

Execution of a Reference Point Run Depending on Parameterization and Start Position

The profile of a reference point approach depends on the assigned parameters and the start position.

Therefore, for a reference point approach, there are different scenarios which are dependent on

- The start position of the drive at the start of the reference point run
- The assigned parameter start direction

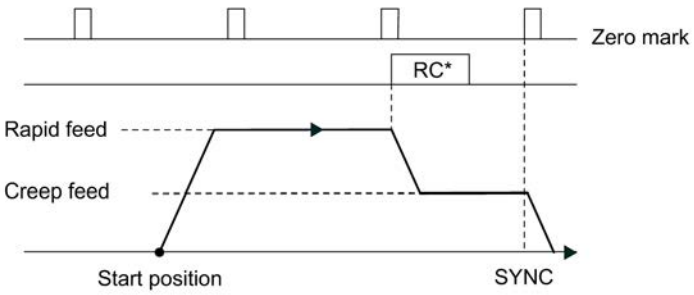
- The assigned parameter reference signal
- The assigned parameter reference switch.

Example 1: Reference point approach with reduction cam and zero mark

Start position and parameters assigned:

- Start position: between the minus limit switch and the reducing cam
- Start direction: Plus
- Reference signal: Reference switch and zero mark
- Reference switch: Reduction cam towards plus

The following figure shows a reference point approach with reducing cams and zero mark.



*RC = reducing cam

You can also carry out synchronization using the reducing cam without a zero mark.

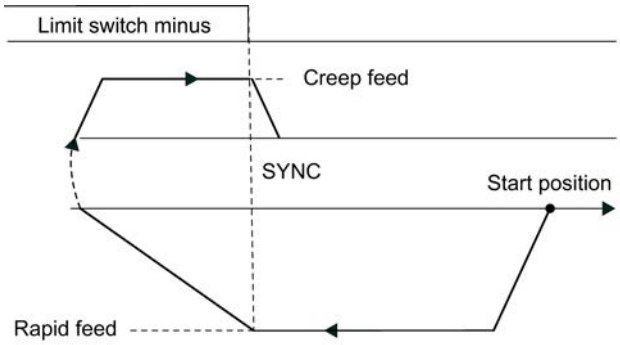
If the start position is on the reducing cam, the 1PosU controls the drive directly at creep speed in the referencing direction.

Example 2: Reference Point Run with Minus Limit Switch

Start position and parameters assigned:

- Start position: between the minus limit switch and the plus limit switch
- Start direction: Minus
- Reference signal: Reference switch
- Reference switch: Limit switch minus

The following figure shows a reference point approach with minus limit switch.



You can also carry out synchronization at the limit switch with the following zero mark.

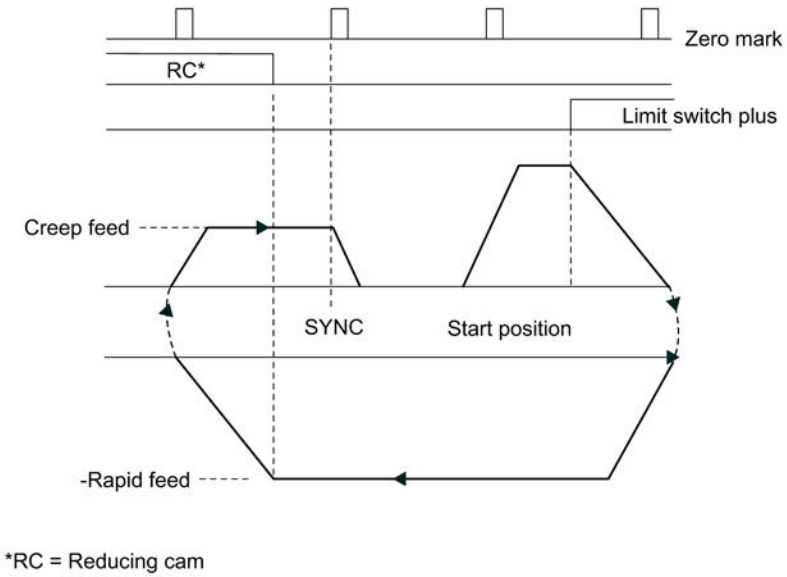
If the start position is on the limit switch, the 1PosU controls the drive directly at creep speed in the referencing direction.

Example 3: Reference Point Run with Reversal of Direction at the Plus Limit Switch

Start position and parameters assigned:

- Start position: between the minus limit switch and the reducing cam
- Start direction: Plus
- Reference signal: Reference switch and zero mark
- Reference switch: Reduction cam towards plus

The following figure shows a reference point approach with a direction reversal at the plus limit switch.



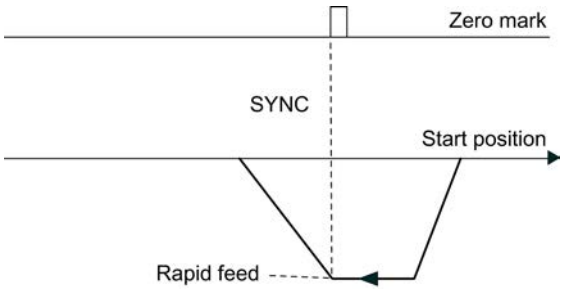
If the start position is at the plus limit switch, the 1PosU controls the drive directly at rapid traverse in the opposite direction to the assigned parameter start direction.

Example 4: Reference Point Run Only with Zero Mark

Start position and parameters assigned:

- Start position: between the minus limit switch and the plus limit switch
- Start direction: Minus
- Reference signal: Zero mark
- Reference switch: irrelevant

The following figure shows a reference point approach just with zero mark.



Relative positioning (MODE 4)

Relative Positioning

In relative positioning the 1PosU moves the drive from the start position in a specified direction for a certain preset distance.

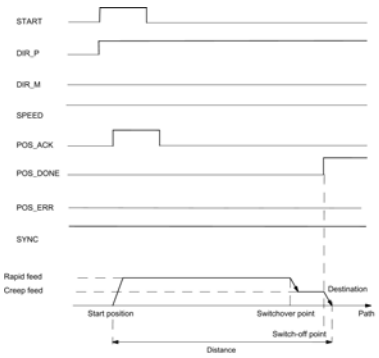
Provide the control interface (Page 10707) with the distance, and start MODE 4, specifying the direction (DIR_M and DIR_P). The 1PosU moves the drive at the preset velocity (SPEED control bit) for that distance. At the switch-over point the 1PosU switches from rapid traverse to creep speed, and at the switch-off point the 1PosU terminates the traverse.

If you start during an active traverse, the 1PosU executes the necessary change in direction (Page 10696) after the time T_{min} has elapsed.

The preset distance is not checked by the 1PosU. This means that more than one revolution may be involved with rotary axes.

In dosing mode (Page 10650), "Relative positioning" is only possible in the plus direction. The actual value is set to 0 at every start.

The figure below shows the sequence of the "Relative positioning" motion type.



Control signals

The following table shows the data in the control interface (Page 10707) which is relevant for "Relative positioning".

Address	Assignment					
Byte 0	Bit 0.7 … 0.4:					
	Bit	7	6	5	4	
		0	1	0	0	MODE 4 = Relative positioning
	Bit 3: SPEED (SPEED = 0 is creep speed; SPEED = 1 is rapid traverse) Bit 2: DIR_M Bit 1: DIR_P Bit 0: START					
Bytes 1 to 3	distance: • Linear axis: 0 … 16 777 215 • Rotary axis: 0 … 16 777 215					

Checkback signals

The following table shows the data in the feedback interface (Page 10709) which is relevant for "Relative positioning".

Address	Assignment
Byte 0	Bit 3: SYNC Bit 2: POS_DONE Bit 1: POS_ERR Bit 0: POS_ACK
Bytes 1 to 3	Actual value: • at the incremental encoder: – Linear axis: 0 … 16 777 215 – Rotary axis: 0 … end of the rotary axis – 1 • at the SSI encoder: – 0 … encoder range – 1

Causes of the POS_ERR error

You must find out the causes of errors with dem JOB 15 (Display current values (Page 10681)).

The following table shows the error numbers relevant for "Relative positioning" with the causes of the errors and possible remedies.

Error number	Cause	Remedy
2	ERR_2L+ is displayed	Check the load voltage (2L+) at terminal 7.
3	ERR_ENCODER is displayed	Check the encoder wiring
5	The limit switch that lies in the direction in which the drive is moved is active	Check your switches and the wiring as well as the "DI0 limit switch minus (Page 10693)" and "DI1 limit switch plus (Page 10693)" parameters.
7	Relative positioning: Start withDIR_P = 0 and DIR_M = 0 or DIR_P = 1 and DIR_M = 1	
13	Direction of rotation of the drive and the encoder varies	Check the wiring of the drive and the encoder as well as the "Reversal of the direction of rotation (Page 10692)" parameter.
15	In dosing mode: DIR_M = 1	

Absolute positioning (MODE 5)

Absolute Positioning

With absolute positioning, the 1PosU moves the drive toward absolute targets. To do this, the axis must be synchronized.

"Absolute positioning" is not possible in activated dosing mode.

Provide the control interface (Page 10707) with the target, and start MODE 5 with the necessary directional enable (Page 10655) (DIR_M, DIR_P). The 1PosU moves the drive at the preset velocity (control bit SPEED) toward the target. At the switch-over point the 1PosU switches from rapid traverse to creep speed, and at the switch-off point the 1PosU terminates the traverse.

If you start during an active traverse, the 1PosU executes the necessary change in direction (Page 10696) after the time T_{min} has elapsed.

Linear axis

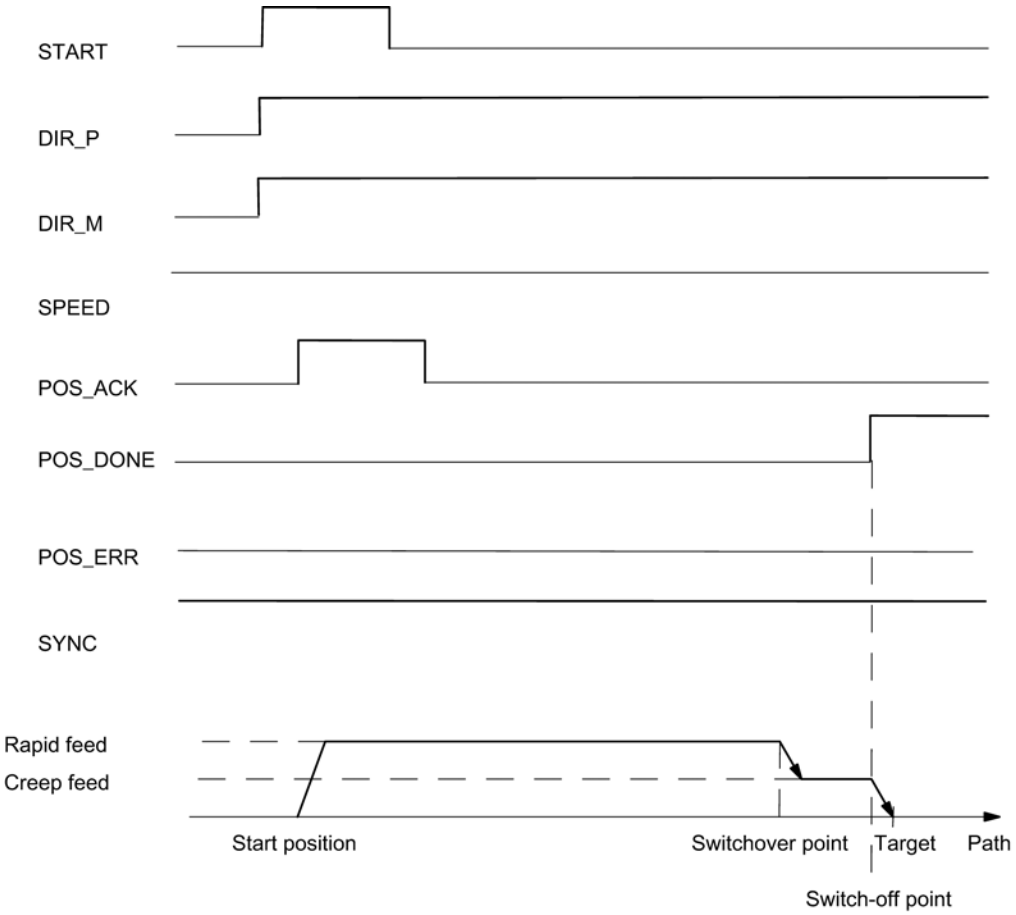
The 1PosU determines the direction the destination is to be approached from. You must set the necessary directional enable (DIR_M, DIR_P) to start. You can also set both enables.

Rotary axis

You determine the direction in which the target is approached by selecting the directional enable (DIR_M, DIR_P):

DIR_P and DIR_M control bits	Direction
DIR_P = 1 DIR_M = 0	The destination is approached in the plus direction.
DIR_P = 0 DIR_M = 1	The destination is approached in the minus direction.
DIR_P = 1 DIR_M = 1	The destination is approached by the shortest route. The 1PosU determines the direction the destination is to be approached from. If the resulting distance to be traversed is smaller than the switch-off difference, no traverse is started (POS_DONE = 1).

The figure below shows the sequence of the "Absolute positioning" motion type.



Control signals

The following table shows the data in the control interface (Page 10707) which is relevant for "Absolute positioning".

<table>
<tr><th>Address</th><th>Assignment</th></tr>
<tr><td rowspan="3">Byte 0</td><td colspan="6">Bit 0.7 … 0.4:</td></tr>
<tr><td rowspan="2">Bit</td><td>7</td><td>6</td><td>5</td><td>4</td><td rowspan="2">MODE 5 = Absolute positioning</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td colspan="6">Bit 3: SPEED (SPEED = 0 is creep speed; SPEED = 1 is rapid traverse)
Bit 2: DIR_M
Bit 1: DIR_P
Bit 0: START</td></tr>
<tr><td>Bytes 1 to 3</td><td colspan="6">Target:
• at the incremental encoder:
– Linear axis: 0 … 16 777 215
– Rotary axis: 0 … end of the rotary axis – 1
• at the SSI encoder:
– 0 … encoder range – 1</td></tr>
</table>

Checkback signals

The following table shows the data in the feedback interface (Page 10709) which is relevant for "Absolute positioning".

Address	Assignment
Byte 0	Bit 3: SYNC Bit 2: POS_DONE Bit 1: POS_ERR Bit 0: POS_ACK
Bytes 1 to 3	Actual value: • at the incremental encoder: – Linear axis: 0 … 16 777 215 – Rotary axis: 0 … end of the rotary axis – 1 • at the SSI encoder: – 0 … encoder range – 1

Causes of the POS_ERR error

You must find out the causes of errors with JOB 15 (Display current values (Page 10681)).

The following table shows the error numbers relevant for "Absolute positioning" with the causes of the errors and possible remedies.

Error number	Cause	Remedy
1	Impermissible MODE in dosing operation	
2	ERR_2L+ is displayed	Check the load voltage (2L+) at terminal 7.
3	ERR_ENCODER is displayed	Check the encoder wiring
4	Axis is not synchronized (SYNC = 0)	You can synchronize the axis with: • Reference point run • Reference Signal Evaluation • Setting Actual Value
5	The limit switch in the direction of which the drive is controlled is active	Check your switches and the wiring as well as the "DI0 limit switch minus (Page 10693)" and "DI1 limit switch plus (Page 10693)" parameters.
7	Start with DIR_P = 0 and DIR_M = 0 or relevant DIR_P = 0 or DIR_M = 0 control bit	
8	Target ≥ end of rotary axis (at incremental encoders) or target ≥ encoder range (at SSI encoders)	
9	Absolute positioning was terminated because JOB 9 was initiated (only at incremental encoders)	
13	Direction of rotation of the drive and the encoder varies	Check the wiring of the drive and the encoder as well as the "Reversal of the direction of rotation (Page 10692)" parameter.

Canceling JOB Processing (JOB 0)

Cancelling JOB processing

When you activate JOB 0, then the 1PosU reacts as follows:

- it cancels the current JOB 9 "Evaluate referencing signal",
- it cancels the current JOB 10 "Latch function",
- it sets a pending JOB_ERR = 0.

You can activate JOB 0 whatever the state of the axis.

Effect on the MODEs

MODEs are not affected by the JOB 0.

Control signals

The following table shows the data in the control interface (Page 10707) which is relevant for "Cancel JOB processing".

Address	Assignment					
Byte 4	Bit 4.7 … 4.4:					
	Bit	7	6	5	4	
		0	0	0	0	JOB 0 = JOB-Cancel processing
	Bit 0: JOB_REQ					

Checkback signals

The following table shows the data in the feedback interface (Page 10709) which is relevant for "Cancel JOB processing".

Address	Assignment
Byte 4	Bit 1: JOB_ERR Bit 0: JOB_ACK

Setting the Actual Value (JOB 1)

Setting actual value

"Set actual value" assigns new coordinates to the actual value displayed. This moves the operating range to another part of the axis.

The axis is synchronized at incremental encoders and non-activated dosing mode.

Provide the control interface (Page 10707) with the new actual value coordinates and activate JOB 1.

The 1PosU sets the preset actual value coordinates to the actual value displayed in the feedback interface (Page 10709) and sets the feedback bit SYNC = 1.

Effect on the MODEs

The following table shows the effects of JOB 1 on the various MODEs.

MODE	What happens?
Reference point approach	At incremental encoders and in non-activated dosing mode, ensure that, when the reference point run is evaluated, the SYNC = 1 feedback bit is set immediately. The reference point run still continues to run.
Inching	-

MODE	What happens?
Absolute positioning	The following reactions are possible: • Distance to the target ≤ switch-off difference The switch-off point is reached or overshot; positioning is switched off immediately, and the run is terminated with POS_DONE = 1. In this case, the destination is sometimes overshot. • Distance to the destination ≤ the changeover difference The changeover point is reached or overshot; there is an immediate reduction from rapid feed to creep feed. In this case, the distance covered at creep speed is less than (switch-over difference - switch-off difference). • Distance to the destination > the changeover difference The drive is moved using rapid traverse, even if it was switched over to creep speed beforehand.
Relative Positioning	The preset distance continues to be traversed.

Control signals

The following table shows the data in the control interface (Page 10707) which is relevant for "Set actual value".

<table>
<tr><th>Address</th><th colspan="6">Assignment</th></tr>
<tr><td rowspan="3">Byte 4</td><td colspan="6">Bit 4.7 … 4.4:</td></tr>
<tr><td rowspan="2">Bit</td><td>7</td><td>6</td><td>5</td><td>4</td><td rowspan="2">JOB 1 = Setting actual value</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td colspan="6">Bit 0: JOB_REQ</td></tr>
<tr><td>Bytes 5 to 7</td><td colspan="6">Actual value coordinates:
• at the incremental encoder:
– Linear axis: 0 … 16 777 215
– Rotary axis: 0 … end of the rotary axis – 1
• at the SSI encoder:
– 0 … encoder range – 1</td></tr>
</table>

Checkback signals

The following table shows the data in the feedback interface (Page 10709) which is relevant for "Set actual value".

Address	Assignment
Byte 0	Bit 3: SYNC
Bytes 1 to 3	Actual value: • at the incremental encoder: – Linear axis: 0 … 16 777 215 – Rotary axis: 0 … end of the rotary axis – 1 • at the SSI encoder: – 0 … encoder range – 1
Byte 4	Bit 1: JOB_ERR Bit 0: JOB_ACK

Causes of the JOB_ERR error

The following table shows the error numbers relevant for "Set actual value" with the causes of the errors and possible remedies.

Error number	Meaning	Remedy
23	ERR_ENCODER is displayed	Check the encoder wiring
34	Actual value coordinates ≥ end of rotary axis (at incremental encoders) or actual value coordinate ≥ encoder range (at SSI encoders)	

Shifting the encoder range (JOB 2)

Move Encoder Range

The "Shift encoder range" function can only be executed at SSI encoders.

"Shift encoder range" adjusts the encoder value so that the actual value displayed corresponds to the real actual value. Before this can be done, any active run must be terminated.

Provide the control interface (Page 10707) with the offset and activate JOB 2.

You calculate the offset as follows:

- Offset = displayed actual value – real actual value

If the offset is negative, proceed as follows:

- Offset = displayed actual value - real actual value + (number of rotations × number of increments)

The 1PosU accepts the preset offset and displays the real actual value coordinates at the feedback interface (Page 10709).

Effect on the MODEs

MODEs are not affected by the JOB 2.

Control signals

The following table shows the data in the control interface (Page 10707) which is relevant for "Shift encoder range"

Address	Assignment					
Byte 4	Bit 4.7 … 4.4:					
	Bit	7	6	5	4	
		0	0	1	0	JOB 2 = Shift encoder range
	Bit 0: JOB_REQ					
Bytes 5 to 7	Offset (0 to encoder range)					

Checkback signals

The following table shows the data in the feedback interface (Page 10709) which is relevant for "Shift encoder range".

Address	Assignment
Bytes 1 to 3	Actual value: • 0 … encoder range – 1
Byte 4	Bit 1: JOB_ERR Bit 0: JOB_ACK

Causes of the JOB_ERR error

The following table shows the error numbers relevant for "Shift encoder range" with the causes of the errors and possible remedies.

Error number	Meaning	Remedy
21	Impermissible JOB at the incremental encoder:	
23	ERR_ENCODER is displayed	Check the encoder wiring
26	JOB 2 "Shift encoder range" cannot be initiated because there is an active traverse	
33	with JOB 2: Offset not in encoder range	

Changing the switch-off difference (JOB 3)

Change Switch-Off Difference

"Change switch-off difference" allows you to adjust the drive control to adapt to any changes in the load and mechanical conditions.

Provide the control interface (Page 10707) with the new switch-off difference and activate JOB 3.

The 1PosU accepts the specified switch-off difference.

The switch-off difference remains valid until the parameter assignment (Page 10689) of the 1PosU is changed.

Effect on the MODEs

The following table shows the effects of JOB 3 on the various MODEs.

MODE	What happens?
Reference point approach	-
Inching	
Absolute positioning	Distance to the target ≤ switch-off difference The switch-off point is reached or overshot; positioning is switched off immediately, and the run is terminated with POS_DONE = 1. In this case, the destination is sometimes overshot.
Relative Positioning	

Control signals

The following table shows the data in the control interface (Page 10707) which is relevant for "Change switch-off difference".

Address	Assignment					
Byte 4	Bit 4.7 … 4.4:					
	Bit	7	6	5	4	
		0	0	1	1	JOB 3 = Change switch-off difference
	Bit 0: JOB_REQ					
Bytes 5 to 7	Switch-off difference • Linear axis: 0 … 16 777 215 • Rotary axis: 0 … 16 777 215					

Checkback signals

The following table shows the data in the feedback interface (Page 10709) which is relevant for "Change switch-off difference".

Address	Assignment
Byte 4	Bit 0: JOB_ACK

Changing the changeover difference (JOB 4)

Change Changeover Difference

"Change switch-over difference" allows you to adjust the drive control to adapt to any changes in the load and mechanical conditions.

Provide the control interface (Page 10707) with the new switch-over difference and activate JOB 4.

The 1PosU accepts the specified switch-over difference.

The switch-over difference remains valid until the parameter assignment (Page 10689) of the 1PosU is changed.

Effect on the MODEs

The following table shows the effects of JOB 4 on the various MODEs.

MODE	What happens?
Reference point approach	-
Inching	
Absolute positioning	The following reactions are possible: • Distance to the destination ≤ the changeover difference. The changeover point is reached or overshot; there is an immediate reduction from rapid feed to creep feed. In this case, the distance covered at creep speed is less than (switch-over difference - switch-off difference). • Distance to the destination > the changeover difference The drive is moved using rapid feed, even if it was switched over to creep feed beforehand.
Relative Positioning	

Control signals

The following table shows the data in the control interface (Page 10707) which is relevant for "Change switch-over difference".

Address	Assignment				
Byte 4	Bit 4.7 … 4.4:				
	Bit	7	6	5	4

Address	Assignment					
		0	1	0	0	JOB 4 = Change switch-over difference
	Bit 0: JOB_REQ					
Bytes 5 to 7	Switch-over difference: • Linear axis: 0 … 16 777 215 • Rotary axis: 0 … 16 777 215					

Checkback signals

The following table shows the data in the feedback interface (Page 10709) which is relevant for "Change changeover difference".

Address	Assignment
Byte 4	Bit 0: JOB_ACK

Evaluating the reference signal (JOB 9)

Evaluating the reference signal

The "Evaluate reference signal" function is only available at incremental encoders and in non-active dosing mode.

"With "Evaluate reference signal" you can synchronize the axis using an external reference signal during a current traverse in "Jogging" and "Relative positioning" MODEs. You can use either the 3 digital inputs or the zero mark (Page 10661) as a reference signal.

You can assign parameters to the digital inputs DI0 (minus limit switch), DI1 (plus limit switch) and DI2 (reducing cam) as NC or NO contacts.

Provide the control interface (Page 10707) with the reference point coordinates and activate JOB 9. The 1PosU sets the SYNC = 0 checkback signal.

If the 1PosU detects the passing of the assigned parameter reference signal in the reference direction, the axis is synchronized. The 1PosU sets the checkback signal SYNC = 1 and assigns the reference point coordinates to the actual value.

The reference direction is controlled by the "Reference signal" and "Reference switch" parameters.

	Reference switch: Reduction cam towards minus	Reference switch: Reduction cam towards plus	Reference switch: Limit switch minus	Reference switch: Limit switch plus
Reference signal: Reference switch and zero mark	Minus referencing direction	Plus referencing direction	Plus referencing direction	Minus referencing direction
Reference signal: Reference switch				
Reference signal: Zero mark	The referencing direction is not defined. The axis is synchronized at the next zero mark.			

Effect on the MODEs

The following table shows the effects of JOB 9 on the various MODEs.

MODE	What happens?
Reference point approach	The reference coordinates transferred with JOB 9 are valid.
Inching	-
Absolute positioning	Traverse is cancelled with POS_ERR = 1, as SYNC us deleted
Relative positioning	-

Control signals

The following table shows the data in the control interface (Page 10707) which is relevant for "Evaluate reference signal".

Address	Assignment					
Byte 4	Bit 4.7 … 4.4:					
	Bit	7	6	5	4	
		1	0	0	1	JOB 9 = Evaluating the reference signal
	Bit 0: JOB_REQ					
Bytes 5 to 7	Reference point coordinates: • Linear axis: 0 … 16 777 215 • Rotary axis: 0 … end of the rotary axis – 1					

Checkback signals

The following table shows the data in the control interface (Page 10709) which is relevant for "Evaluate reference signal".

Address	Assignment
Byte 0	Bit 3: SYNC
Bytes 1 to 3	Actual value: • Linear axis: 0 … 16 777 215 • Rotary axis: 0 … end of the rotary axis – 1
Byte 4	Bit 1: JOB_ERR Bit 0: JOB_ACK

Causes of the JOB_ERR error

The following table shows the error numbers relevant for "Evaluate the reference signal" with the causes of the errors and possible remedies.

Error number	Meaning	Remedy
21	Impermissible JOB at SSI encoders or in dosing operation	
23	ERR_ENCODER is displayed	Check the encoder wiring
30	Reference point coordinate ≥ end of the rotary axis	

Latch Function (JOB 10)

Latch Function

The "Latch function" allows you to store the actual value once at an edge at the DI2 digital input. You can use this function, for example, to detect edges or measure lengths.

Provide the control interface (Page 10707) with the desired edge and activate JOB 10.

If the 1PosU detects the preset edge at the DI2 digital input, it stores the associated actual value, displays it as a feedback value (Page 10709) and sets the LATCH_DONE = 1 feedback bit.

You can then activate the "Latch function" again.

Latch function and reference point approach or "Evaluate the reference signal"

If the 1PosU synchronizes at the same edge, it stores the actual value before it assigns the reference point coordinates.

Effect on the MODEs

MODEs are not affected by the JOB 10.

Control signals

The following table shows the data in the control interface (Page 10707) which is relevant for "Latch function".

Address	Assignment					
Byte 4	Bit 4.7 … 4.4:					
	Bit	7	6	5	4	
		1	0	1	0	JOB 10 = Latch function
	Bit 0: JOB_REQ					
Byte 5	Bit 1: Latch at negative edge at DI2 Bit 0: Latch at positive edge at DI2					

Checkback signals

The following table shows the data in the feedback interface (Page 10709) which is relevant for "Latch function".

Address	Assignment
Byte 4	Bit 2: LATCH_DONE Bit 1: JOB_ERR Bit 0: JOB_ACK
Bytes 5 to 7	Feedback value: Actual value at the edge at the DI2 • at the incremental encoder: – Linear axis: 0 … 16 777 215 – Rotary axis: 0 … end of the rotary axis – 1 • at the SSI encoder: – 0 … encoder range – 1

Causes of the JOB_ERR error

The following table shows the error numbers relevant for "Latch function" with the causes of the errors and possible remedies.

Error number	Meaning	Remedy
23	ERR_ENCODER is displayed	Check the encoder wiring
36	Edge selection unknown	

Defining the settings for monitoring the direction of rotation (JOB 11)

Setting the Monitoring of the Direction of Rotation

The "Define the settings for monitoring the direction of rotation" function is not available in dosing mode.

With "Define the settings for monitoring the direction of rotation" you can adjust the monitoring of the 1PosU's direction of rotation to suit your load and mechanical conditions.

Monitoring of the direction of rotation is always active. The 1PosU detects whether the direction of rotation of the drive and the encoder is the same. Direction of rotation monitoring will tolerate different directions for the drive and the encoder up to the preset path difference. If the specified distance is exceeded then the 1PosU reports POS_ERR = 1.

Unless you have activated JOB 11, double the switch-off difference is used from the parameters as the path difference. JOB 3 "Change switch-off difference" does not affect the distance difference for the purpose of monitoring the direction of rotation.

Provide the control interface (Page 10707) with the new distance difference and activate JOB 11.

The 1PosU accepts the preset distance difference for the monitoring of the direction of rotation.

The distance difference for the monitoring of the direction of rotation remains valid until the parameter assignment (Page 10689) of the 1PosU is changed.

Disabling the Monitoring of the Direction of Rotation

Monitoring of the direction of rotation is disabled when the path difference is 0.

Effect on the MODEs

MODEs are not affected by the JOB 11.

Control signals

The following table shows the data in the control interface (Page 10707) which is relevant for "Define the settings for monitoring the direction of rotation".

Address	Assignment					
Byte 4	Bit 4.7 … 4.4:					
	Bit	7	6	5	4	JOB 11 = Define the settings for monitoring the direction of rotation
		1	0	1	1	
	Bit 0: JOB_REQ					
Byte 5	0					
Bytes 6, 7	Distance difference for monitoring the direction of rotation • 0 … 65 535					

Checkback signals

The following table shows the data in the feedback interface (Page 10709) which is relevant for "Define the settings for monitoring the direction of rotation".

Address	Assignment
Byte 4	Bit 1: JOB_ERR Bit 0: JOB_ACK

Causes of the JOB_ERR error

The following table shows the error numbers relevant for "Define the settings for monitoring the direction of rotation" with the causes of the errors and possible remedies.

Error number	Meaning	Remedy
21	Impermissible JOB in dosing mode	
38	Monitoring of the direction of rotation path difference > 65 535	

Displaying current values (JOB 15)

Display Current Values

You can display the following values in the feedback interface as feedback values:

- Residual distance
- Actual speed
- Causes of errors POS_ERR and JOB_ERR

The residual distance is set by the 1PosU as the default for the feedback value.

The 1PosU continuously displays the actual value in the feedback interface irrespective of the selected feedback value.

Provide the control interface (Page 10707) with the desired feedback value and activate JOB 15.

The selected feedback value remains valid until the parameter assignment (Page 10689) of the 1PosU is changed.

"Display current values and latch function

If you activate the latch function, the 1PosU sets a feedback value of 0 and displays the actual value at the edge at the DI2 digital input.

You can only activate JOB 15 again after the latch function has terminated.

Residual distance

The 1PosU calculates the distance to the target as the residual distance in the "Absolute positioning" and "Relative positioning" MODEs. As long as the actual value is before the destination, the residual distance remains positive. It becomes negative when the destination is overshot. The residual distance is 0 in the other MODEs.

The 1PosU displays the residual distance with a sign between -8 388 608 and 8 388 607 increments. Negative values are displayed in twos complement. If the actual residual distance is beyond these limits, the limit value is displayed.

Actual speed

The 1PosU calculates the actual velocity as an encoder value change in increments per 10 ms. It displays these between 0 and 16 777 215.

Causes of errors POS_ERR and JOB_ERR

The 1PosU displays the causes of errors for POS_ERR and JOB_ERR as well as the MODE and JOB entered in the control interface (Page 10707).

Effect on the MODEs

MODEs are not affected by the JOB 15.

Control signals

The following table shows the data in the control interface (Page 10707) which is relevant for "Display current values".

Address	Assignment					
Byte 4	Bit 4.7 … 4.4:					
	Bit	7	6	5	4	
		1	1	1	1	JOB 15 = Display current values
	Bit 0: JOB_REQ					
Byte 5	0: Residual distance 1: Actual speed 2: Causes of errors POS_ERR and JOB_ERR					

Checkback signals

The following table shows the data in the feedback interface (Page 10709) which is relevant for "Display current values".

Address	Assignment
Byte 4	Bit 1: JOB_ERR Bit 0: JOB_ACK
Bytes 5 to 7	In accordance with the selected feedback value: • With a residual distance of: –8 388 608 … 8 388 607 • With an actual speed of: 0 … 16 777 215 • with causes of POS_ERR and JOB_ERR errors: – Byte 5: Causes of (Page 10687)POS_ERR error – Byte 6: Causes of (Page 10688)JOB_ERR error – Bit 7.3 … 7.0: MODE (= bits 0.7 to 0.4 from the control signals) – Bit 7.7 … 7.4: JOB (= bits 4.7 to 4.4 from the control signals)

Causes of the JOB_ERR error

The following table shows the error numbers relevant for "Display current values" with the causes of the errors and possible remedies.

Error number	Meaning	Remedy
35	Display current values: Selection unknown	
37	Display current values: JOB 15 cannot be activated with the latch function running.	

Error detection / diagnostics

Overview of errors with 1PosU

Various errors can occur when working with the 1PosU. The following types of error are differentiated between:

- Parameter assignment error (Page 10683)
- External Errors (Page 10684)
- Errors in the Control of MODEs and JOBs (Page 10686)

You must acknowledge (Page 10686) the "Load voltage missing", "Short circuit of the encoder supply" and "Wire break/short circuit of the encoder signals" errors.

If errors occur when starting MODEs or activating JOBs, you can find more information here:

- Causes of POS_ERR error (Page 10687)
- Causes of JOB_ERR error (Page 10688)

Parameter assignment error

Parameter assignment error

The following table shows 1PosU parameter assignment errors.

Parameter assignment error	1PosU reactions
Causes: • The 1PosU cannot identify existing parameters as its own. • The 1PosU slot you have configured does not match the structure. Only at SSI encoder: • Impermissible value for "Pulses per encoder revolution" parameter. • Impermissible value for "Revolutions" parameter. • Pulses per encoder revolution × revolutions is greater than 4096 × 4096. Only for dosing operation: • Activated "Encoder signal diagnostics" • Activated "Reversal of the direction of rotation" What to Do: • Check the configuration and structure.	• The 1PosU is not configured and cannot execute its functions. • Generating channel-specific diagnostics

External errors

External errors

The following table shows 1PosU external errors.

Load Voltage 2L+ Missing	Reactions of the 1PosU
Causes: • Load voltage 2L+ not present or too low at terminal 7 Remedy: • Check the wiring. Eliminate a possible short-circuit. • Acknowledge (Page 10686) the error with the EXTF_ACK control bit.	• The current traverse is halted; it is not possible to start a new traverse. – All 3 digital outputs are set to 0. – POS_ERR = 1 feedback bit – POS_DONE = 0 feedback bit • ERR_2L+ = 1 feedback bit • generate channel-specific diagnostics • waits for EXTF_ACK error acknowledgment
Short circuit of the encoder supply	**Reactions of the 1PosU**
Causes: • Short circuit of the encoder supply made available at terminals 6 and 10 Remedy: • Check the wiring and correct the short circuit. • Acknowledge (Page 10686) the error with the EXTF_ACK control bit.	• The current "Reference point approach", "Relative positioning" and "Absolute positioning" MODEs are stopped; it is not possible to start a new traverse in these MODEs. – All 3 digital outputs are set to 0. – POS_ERR = 1 feedback bit – POS_DONE = 0 feedback bit • ERR_ENCODER = 1 feedback bit • SYNC = 0 feedback bit • generate channel-specific diagnostics • waits for EXTF_ACK error acknowledgment • Jogging MODE is not affected by this error. • The current "Evaluate referencing signal" JOB is cancelled.

Wire break/short circuit of the encoder signals	Reactions of the 1PosU
If an SSI encoder is used: Requirements: • To detect errors at the encoder signals, you must enable the "Encoder signal diagnostics" parameter. Causes: • Wire break or short circuit of the encoder signals at terminals 9 and 13 or 12 and 16. • The parameters for the SSI encoder do not correspond to the encoder connected. Remedy: • Check the wiring and correct the short circuit. • Compare the configuration with the technical specifications of the encoder. • Acknowledge (Page 10686) the error with the EXTF_ACK control bit. **The following applies when using an incremental encoder:** Requirement: • You must enable the "Encoder signal diagnostics" parameter for error detection at the A, /A and B, /B signals with 5V differential signal or at the A* and B* with 24V differential signal. • The "Zero marker diagnostics" parameter must be enabled in order to allow error recognition at the N, /N signals with 5V differential signal. If you use an encoder without a zero mark, switch off error detection. If dosing mode is activated, "Zero marker signal diagnostics" is not possible. Causes: • For 5V differential signal: Wire break or short circuit of the encoder signals at terminals 9 and 13 or 11 and 15 or 12 and 16. • Edge error of the encoder signals recognized so that 1PosU cannot carry out clear direction recognition. Remedy: • Check the wiring and correct the short circuit. • Acknowledge (Page 10686) the error with the EXTF_ACK control bit.	• The current "Reference point approach", "Relative positioning" and "Absolute positioning" MODEs are stopped; it is not possible to start a new traverse in these MODEs. – All 3 digital outputs are set to 0. – POS_ERR = 1 feedback bit – POS_DONE = 0 feedback bit • ERR_ENCODER = 1 feedback bit • SYNC = 0 feedback bit • generate channel-specific diagnostics • waits for EXTF_ACK error acknowledgment • Jogging MODE is not affected by this error. • The current "Evaluate referencing signal" JOB is cancelled.

Errors in the control of MODEs and JOBs

Errors in the control of MODEs and JOBs

The following tables shows errors in the control of MODEs and JOBs with the 1PosU.

POS_ERR	Reactions of the 1PosU
Causes: • Certain requirements or conditions have not been met at the start of a MODEs	• The MODE started is not executed. • The current traverse is stopped. – All 3 digital outputs are set to 0. – POS_ERR = 1 feedback bit – POS_DONE = 0 feedback bit
JOB_ERR	**Reactions of the 1PosU**
Causes: • Certain requirements or conditions have not been met at the start of a JOBs	• The activated JOB is not executed. – JOB_ERR = 1 feedback bit

Error acknowledgment EXTF_ACK

EXTF_ACK error acknowledgment

You must acknowledge (Page 10684) the "Load voltage missing", "Short circuit of the encoder supply" and "Wire break/short circuit of the encoder signals" corrected external errors. The following table shows the sequence necessary for this.

What You Do	1PosU reactions
	• ERR_2L+ = 1 feedback bit and/or • ERR_ENCODER = 1 feedback bit
Your control program detects the set ERR_2L+ or ERR_ENCODER feedback bit. Execute your application-specific error response. Eliminate the cause of the error.	
Switch the EXTF_ACKcontrol bit (Page 10707) from 0 to 1	The 1PosU sets the ERR_2L+ = 0 and ERR_ENCODER = 0feedback bits (Page 10709). This tells you that the cause has been eliminated and acknowledged. If ERR_2L+ = 1 and/or ERR_ENCODER = 1, the cause of the error has not yet been eliminated.
Switch the EXTF_ACKcontrol bit (Page 10707) from 1 to 0.	
In the case of constant error acknowledgment (EXTF_ACK = 1) or at CPU/Master Stop, the 1PosU reports the errors as soon as they are detected and deletes them as soon as they have been eliminated.	

Causes of POS_ERR error

Causes of the POS_ERR error

The following table shows the causes and possible remedies for the POS_ERR error.

Error number	Cause	Remedy
1	MODE unknown or impermissible	Permissible MODEs are: • MODE 0 • MODE 1 • MODE 3 (not possible at SSI encoders or in dosing mode) • MODE 4 • MODE 5 (not possible in dosing mode)
2	ERR_2L+ is displayed	Check the load voltage (2L+) at terminal 7.
3	ERR_ENCODER is displayed	Check the encoder wiring
4	Axis is not synchronized (SYNC = 0)	SSI encoder: • Eliminate the encoder error. The axis can be synchronized at incremental encoders and non-activated dosing mode with: • Reference point approach • Evaluating the reference signal • Set actual value
5	The limit switch in the direction of which the drive is controlled is active	Check your switches and the wiring as well as the "DI0 limit switch minus (Page 10693)" and "DI1 limit switch plus (Page 10693)" parameters.
7	Inching: DIR_P = 1 and DIR_M = 1 Absolute positioning: Start with DIR_P = 0 and DIR_M = 0 or relevant DIR_P = 0 or DIR_M = 0 control bit Relative positioning: Start withDIR_P = 0 and DIR_M = 0 or DIR_P = 1 and DIR_M = 1	
8	Absolute positioning : Target ≥ end of rotary axis at incremental encoders or target ≥ encoder range at SSI encoders	
9	Absolute positioning was terminated because JOB 9 was initiated	
10	Reference point approach: Reference point coordinate ≥ end of the rotary axis	
11	Reference point approach: No reference signal found up to the limit switch or between the limit switches	Check your switches, the encoder and the wiring

Error number	Cause	Remedy
13	Direction of rotation of the drive and the encoder varies	Check the wiring of the drive and the encoder as well as the "Reversal of the direction of rotation (Page 10692)" parameter.
15	Jogging, relative positioning: In dosing mode DIR_M = 1	

Causes of JOB_ERR error

Causes of the JOB_ERR error

The following table shows the causes and possible remedies for the POS_ERR error.

Error number	Meaning	Remedy
21	JOB unknown or impermissible	Permissible JOBs are: • JOB 0 • JOB 1 • JOB 2 (only possible at SSI encoders) • JOB 3 • JOB 4 • JOB 9 (not possible at SSI encoders or in dosing mode (Page 10650)) • JOB 10 • JOB 11 (not possible in dosing mode) • JOB 15
23	ERR_ENCODER is displayed	Check the encoder wiring
26	JOB 2 "Shift encoder range" cannot be initiated because there is an active traverse	
29	Evaluating the Reference Signal: Reference point coordinate ≥ end of the rotary axis	
34	Setting an Actual Value: Actual value coordinates ≥ encoder range	
35	Display current values: Selection unknown	
36	Latch Function: Edge selection unknown	
37	Display current values: JOB 15 cannot be activated with the latch function running.	
38	Monitoring of the direction of rotation path difference > 65 535	

CPU/Master STOP and RESET state

Behavior in CPU/Master-STOP

The following table shows the behavior of 1PosU in CPU/Master-STOP.

Behavior in CPU/Master-STOP	1PosU reactions
• due to POWER-OFF of the CPU / bus master or • due to POWER-OFF of the IM 151 / IM 151 FO or • due to failure of bus transmission or • Due to change from RUN to STOP	• The current traverse is stopped. • All 3 digital outputs are set to 0. • POS_ERR = 0feedback bit (Page 10709) • POS_DONE = 1feedback bit (Page 10709)

Exiting the CPU/Master STOP status

The following table shows the behavior of 1PosU after exiting the CPU/Master-STOP status.

Exiting the CPU/Master STOP status	1PosU reactions
• with POWER-ON of the CPU / bus master or • with POWER-OFF of the IM 151 / IM 151 FO or • after failure of the bus transmission or • After a change from STOP to RUN	• The feedback interface (Page 10709) of the 1PosU remains current. • The axis remains synchronized, and the actual value is current. • The moved encoder range remains valid. • The changed switch-off and changeover differences and the path difference for the monitoring of the direction of rotation remain valid. • An initiated "Evaluate referencing signal (Page 10676)" JOB 9 and "Latch function (Page 10678)" JOB 10 remain active. • The feedback value (Page 10681) selected with JOB 15is current.

RESET state of the 1PosU

The following table shows the behavior of the 1PosU in RESET state.

RESET state of the 1PosU and changing the 1PosU parameters	1PosU reactions
• due to changing the parameters of the 1PosU and downloading the parameter assignment or configuration of the ET 200S station to the CPU / bus master or • due to POWER-ON at the power module of the 1PosU or • inserting the 1PosU in an energized state	• The axis is synchronized, and the actual value corresponds to the current encoder value. • The encoder range has not been moved. • The switch-off and changeover difference is accepted from the parameters. • The path difference for the monitoring of the direction of rotation is set at double the switch-off difference. • "Evaluate referencing signal (Page 10676)" JOB 9 and "Latch function (Page 10678)" JOB 10 are not active. • The residual distance is displayed as a feedback value (Page 10681).

Configuring and assigning parameters to 1PosU

Operation

Selecting the encoder interface

Encoder interface

You define the 1PosU signal type you want to use by setting the encoder interface.

Encoder interface	Description
5V differential signal (Page 10691)	The 1PosU receives the signals from the incremental encoders with 5V differential signal.
24 V differential signal (Page 10696)	The 1PosU receives the signals from the incremental encoders with 24V signals.
SSI absolute (Page 10702)	The 1PosU receives the signals from the absolute encoders (SSI).

Parameters for incremental encoders with 5V differential signal

Enables

Enabling group diagnostics

Group diagnostics

You enable group diagnostics by ticking the corresponding options box.

When group diagnostics is enabled, an encoder error (ERR_ENCODER), no load voltage (ERR_2L+) or a parameter assignment error will result in a channel-specific diagnostics.

Enabling encoder signal diagnostics

Encoder signal diagnostics

You enable encoder signal diagnostics by ticking the corresponding options box.

With "Encoder signal diagnostics" enabled, encoder signals A, /A and B, /B are monitored for short circuit and wire break. With non-enabled dosing mode, the signal sequence is also monitored. A simultaneous edge change at signals A and B causes an error.

Enable zero mark signal diagnostics

Zero mark signal diagnostics

You enable zero mark signal diagnostics by ticking the corresponding options box.

With "Zero mark signal diagnostics" enabled, zero mark signals N, /N are monitored for short circuit and wire break.

Encoder and axes

Enabling dosing mode

Dosing mode

You enable dosing mode by ticking the corresponding options box.

In dosing mode (Page 10650), 1PosU only evaluates encoder signals A and /A. As a result, the distance can only be changed in the positive direction. The actual value is incremented at each positive edge.

Enabling reversal of direction of rotation

Reversal of direction of rotation

You enable reversal of direction of rotation by ticking the corresponding options box.

You can use the "Reversal of direction of rotation" parameter to adapt the direction of rotation of the encoder to that of the drive and the axis.

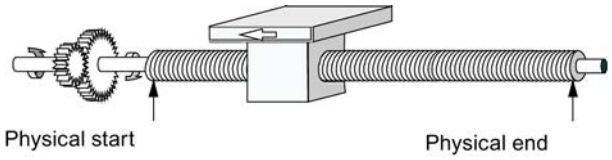
In dosing mode (Page 10650), reversal of the direction of rotation is not possible.

Selecting axis type

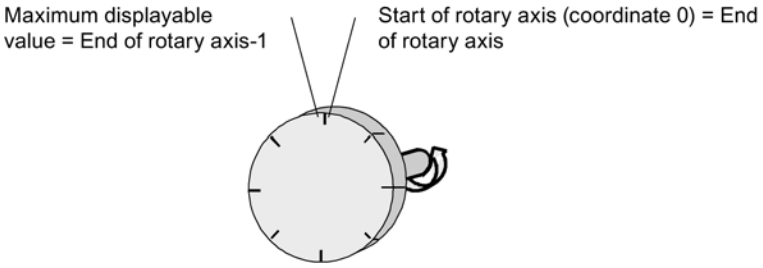
Axis types

The 1PosU positioning module can control the following axis types:

- Linear axis
  A linear axis is an axis which has a limited physical traversing range.



- Rotary axis
  The rotary axis is an axis whose traversing range is not restricted by mechanical stops.



Specifying end of rotary axis

End of rotary axis

This parameter is only relevant for the "Rotary axis" axis type.

The value "end of rotary axis" is the theoretical maximum value that the actual value can reach. The theoretically highest value is however never displayed, because it physically identifies the same position as the start of the rotary axis (0).

The displayed value jumps at

- high limit violated of (end of rotary axis – 1) to 0
- low limit violated of 0 to (end of rotary axis – 1)

Permitted value range for the end of the rotary axis: 1 … 16 777 215
Specifying "0" causes a configuration error.

Digital inputs

Selecting minus limit switch and plus limit switch

Function of the hardware limit switches

The two digital inputs (DI0 and DI1) are evaluated by the 1PosU as limit switches:

- DI0 is the minus limit switch and limits the operating range in the minus direction.
- DI1 is the plus limit switch and limits the operating range in the plus direction.

You can assign parameters to the hardware limit switches **separately as NC or NO contacts**.

The hardware limit switches are evaluated with linear axes and rotary axes.

Only the hardware limit switch that lies in the direction in which the drive is being moved is evaluated.

This enables you to move away from a hardware limit switch without additional error acknowledgment by moving in the other direction if you reach or overrun a hardware limit switch.

The current signal level of the digital inputs is displayed in the feedback interface, delayed by the update rate.

The following table shows how the hardware limit switches work in the individual MODEs.

MODE	Function of the hardware limit switches
Reference point approach	The 1PosU executes an automatic direction reversal at the hardware limit switch.
Inching	The motion of the axis is halted on the hardware limit switch, all 3 digital outputs are set to 0, and the POS_ERR feedback bit is reported.
Absolute Positioning	
Relative Positioning	

Starting on the hardware limit switch

The following table shows the reaction of the 1PosU when starting MODEs on the hardware limit switch.

Direction	1PosU reactions
Starting into the operating range	The 1PosU starts the specified MODE.
Starting away from the operating range	The POS_ERR = 1 feedback bit is set.

Selecting reduction cam

Function of the reduction cam

The digital input DI2 is evaluated as a reduction cam or latch input by 1PosU:

- as a reduction cam with MODE 3 "Reference point approach (Page 10659)" and with JOB 9 "Evaluate referencing signal (Page 10676)"
- as a latch input with JOB 10 "Latch function (Page 10678)"

You can assign parameters to the hardware limit switches separately as **NC or NO contacts**.

Reference point approach and evaluation of the reference signal

Selecting reference signal

Reference signal

The "Reference signal" parameter defines the relevant switch or combination of switches and zero marks for MODE 3 "Reference point approach (Page 10659)" and JOB 9 "Evaluate reference signal (Page 10676)". The following are selectable:

- Reference switch and zero mark
- Reference switch
- Zero mark

You select the type of reference switch with the "Reference switch (Page 10694)" parameter.

Selecting reference switch

Reference switch

The "Reference switch" parameter defines the relevant switch and the reference direction in which the switch must be overrun. The following are selectable:

- Reduction cam towards minus
- Reduction cam towards plus
- Limit switch minus
- Limit switch plus

These parameters are only relevant if you have selected it with "reference signal (Page 10694)"

- Reference switch and zero mark
  or
- Reference switch

Selecting start direction

Start direction

The "Start direction" parameter defines the start direction for the MODE 3 "Reference point approach (Page 10659)". The following are selectable:

- Plus
- Minus

Drive

Selecting control mode

Control mode

The"Control mode" parameter defines the function of the digital outputs for the control of the drive (Page 10654). The following are selectable:

- Control mode 0:
  - DO0 travel minus
  - DO1 travel plus
  - DO2 rapid/creep feed
- Control mode 1:
  - DO0 rapid feed
  - DO1 creep feed (rapid feed is 0)
  - DO2 traverse plus (1)/ traverse minus (0)

Entering the switch-off difference

Switch-off difference

Switch-off difference (Page 10648) defines the distance from the target at which the drive is slowed down from creep speed to 0.

If the switch-off difference ≥ the changeover difference, there is no changeover point. There is no deceleration from rapid traverse to creep speed; the response is executed directly at the switch-off point.

Valid value range:

- 0 to 65,535

You can use JOB 3 (Page 10674) to change the switch-off difference.

Entering the changeover difference

Changeover difference

Changeover difference (Page 10648) defines the distance from the target at which the drive is slowed down from rapid traverse to creep speed.

Valid value range:

- 0 to 65,535

You can use JOB 4 (Page 10675) to change the changeover difference.

Entering the delay time for the direction change

T_{min} direction change

The "T_{min} direction change" parameter defines the delay time for the direction change. The digital outputs are switched off, and a change of direction by T_{min} is executed with a delay. T_{min} is effective at each change of direction during a run.

T_{min} does not work when starting after POS_DONE = 1 or POS_ERR = 1.

Valid value range:

- 0 to 255

Your input value is multiplied by 10. You thus specify T_{min} in increments of 10 ms (for example, 0 ms, 10 ms or 2550 ms).

Parameters for incremental encoders with 24V signals

Enables

Enabling group diagnostics

Group diagnostics

You enable group diagnostics by ticking the corresponding options box.

When group diagnostics is enabled, an encoder error (ERR_ENCODER), no load voltage (ERR_2L+) or a parameter assignment error will result in a channel-specific diagnostics.

Enabling encoder signal diagnostics

Encoder signal diagnostics

You enable encoder signal diagnostics by ticking the corresponding options box.

A simultaneous edge change of the signals A* and B* with "Encoder signal diagnostics" enabled leads to an error.

Encoder and axes

Enabling dosing mode

Dosing mode

You enable dosing mode by ticking the corresponding options box.

In dosing mode (Page 10650) the 1PosU only evaluates the encoder signal A*. As a result, the distance can only be changed in the positive direction. The actual value is incremented at each positive edge.

Selecting encoder inputs

Encoder inputs

Different sensors can be connected to the encoder inputs:

- Sourcing output/push pull
- Sinking output

Note

If you have selected the "Sinking output" setting with for the "Encoder inputs" parameter, you must use the M-switching sensors.

Select the output circuit of the encoder in use for the signals A*, B* and N* from the drop-down list box.

Enabling reversal of direction of rotation

Reversal of direction of rotation

You enable reversal of direction of rotation by ticking the corresponding options box.

You can use the "Reversal of direction of rotation" parameter to adapt the direction of rotation of the encoder to that of the drive and the axis.

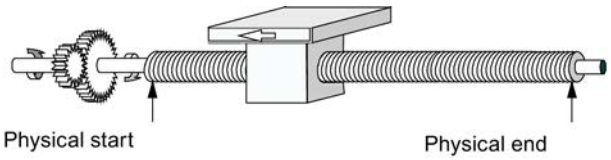
In dosing mode (Page 10650), reversal of the direction of rotation is not possible.

Selecting axis type

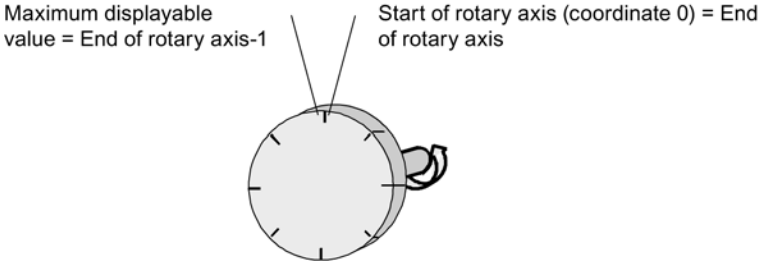
Axis types

The 1PosU positioning module can control the following axis types:

- Linear axis
  A linear axis is an axis which has a limited physical traversing range.



- Rotary axis
  The rotary axis is an axis whose traversing range is not restricted by mechanical stops.



Specifying end of rotary axis

End of rotary axis

This parameter is only relevant for the "Rotary axis" axis type.

The value "end of rotary axis" is the theoretical maximum value that the actual value can reach. The theoretically highest value is however never displayed, because it physically identifies the same position as the start of the rotary axis (0).

The displayed value jumps at

- high limit violated of (end of rotary axis – 1) to 0
- low limit violated of 0 to (end of rotary axis – 1)

Permitted value range for the end of the rotary axis: 1 … 16 777 215
Specifying "0" causes a configuration error.

Digital inputs

Selecting minus limit switch and plus limit switch

Function of the hardware limit switches

The two digital inputs (DI0 and DI1) are evaluated by the 1PosU as limit switches:

- DI0 is the minus limit switch and limits the operating range in the minus direction.
- DI1 is the plus limit switch and limits the operating range in the plus direction.

You can assign parameters to the hardware limit switches **separately as NC or NO contacts**.

The hardware limit switches are evaluated with linear axes and rotary axes.

Only the hardware limit switch that lies in the direction in which the drive is being moved is evaluated.

This enables you to move away from a hardware limit switch without additional error acknowledgment by moving in the other direction if you reach or overrun a hardware limit switch.

The current signal level of the digital inputs is displayed in the feedback interface, delayed by the update rate.

The following table shows how the hardware limit switches work in the individual MODEs.

MODE	Function of the hardware limit switches
Reference point approach	The 1PosU executes an automatic direction reversal at the hardware limit switch.
Inching	The motion of the axis is halted on the hardware limit switch, all 3 digital outputs are set to 0, and the POS_ERR feedback bit is reported.
Absolute Positioning	
Relative Positioning	

Starting on the hardware limit switch

The following table shows the reaction of the 1PosU when starting MODEs on the hardware limit switch.

Direction	1PosU reactions
Starting into the operating range	The 1PosU starts the specified MODE.
Starting away from the operating range	The POS_ERR = 1 feedback bit is set.

Selecting reduction cam

Function of the reduction cam

The digital input DI2 is evaluated as a reduction cam or latch input by 1PosU:

- as a reduction cam with MODE 3 "Reference point approach (Page 10659)" and with JOB 9 "Evaluate referencing signal (Page 10676)"
- as a latch input with JOB 10 "Latch function (Page 10678)"

You can assign parameters to the hardware limit switches separately as **NC or NO contacts**.

Reference point approach and evaluation of the reference signal

Selecting reference signal

Reference signal

The "Reference signal" parameter defines the relevant switch or combination of switches and zero marks for MODE 3 "Reference point approach (Page 10659)" and JOB 9 "Evaluate reference signal (Page 10676)". The following are selectable:

- Reference switch and zero mark
- Reference switch
- Zero mark

You select the type of reference switch with the "Reference switch (Page 10700)" parameter.

Selecting reference switch

Reference switch

The "Reference switch" parameter defines the relevant switch and the reference direction in which the switch must be overrun. The following are selectable:

- Reduction cam towards minus
- Reduction cam towards plus
- Limit switch minus
- Limit switch plus

These parameters are only relevant if you have selected it with "reference signal (Page 10700)"

- Reference switch and zero mark
  or
- Reference switch

Selecting start direction

Start direction

The "Start direction" parameter defines the start direction for the MODE 3 "Reference point approach (Page 10659)". The following are selectable:

- Plus
- Minus

Drive

Selecting control mode

Control mode

The"Control mode" parameter defines the function of the digital outputs for the control of the drive (Page 10654). The following are selectable:

- Control mode 0:
  - DO0 travel minus
  - DO1 travel plus
  - DO2 rapid/creep feed
- Control mode 1:
  - DO0 rapid feed
  - DO1 creep feed (rapid feed is 0)
  - DO2 traverse plus (1)/ traverse minus (0)

Entering the switch-off difference

Switch-off difference

Switch-off difference (Page 10648) defines the distance from the target at which the drive is slowed down from creep speed to 0.

If the switch-off difference ≥ the changeover difference, there is no changeover point. There is no deceleration from rapid traverse to creep speed; the response is executed directly at the switch-off point.

Valid value range:

- 0 to 65,535

You can use JOB 3 (Page 10674) to change the switch-off difference.

Entering the changeover difference

Changeover difference

Changeover difference (Page 10648) defines the distance from the target at which the drive is slowed down from rapid traverse to creep speed.

Valid value range:

- 0 to 65,535

You can use JOB 4 (Page 10675) to change the changeover difference.

Entering the delay time for the direction change

T_{min} direction change

The "T_{min} direction change" parameter defines the delay time for the direction change. The digital outputs are switched off, and a change of direction by T_{min} is executed with a delay. T_{min} is effective at each change of direction during a run.

T_{min} does not work when starting after POS_DONE = 1 or POS_ERR = 1.

Valid value range:

- 0 to 255

Your input value is multiplied by 10. You thus specify T_{min} in increments of 10 ms (for example, 0 ms, 10 ms or 2550 ms).

Parameters for SSI encoders

Enables

Enabling group diagnostics

Group diagnostics

You enable group diagnostics by ticking the corresponding options box.

When group diagnostics is enabled, an encoder error (ERR_ENCODER), no load voltage (ERR_2L+) or a parameter assignment error will result in a channel-specific diagnostics.

Enabling encoder signal diagnostics

Encoder signal diagnostics

You enable encoder signal diagnostics by ticking the corresponding options box.

With "Encoder signal diagnostics" enabled, encoder signals D, /D are monitored for short circuit and wire break. Monitoring of the frame is carried out additionally (start bit and stop bit).

Encoder and axes

Selecting encoder type

Encoder type

Select the encoder in use from the drop-down list:

- Single-turn encoder (SSI 13 bit)
- Multi-turn encoder (SSI 25 bit)

Selecting the transmission rate

Transmission rate

With the "transmission rate" parameter, you define the data transmission rate between the SSI encoder and the positioning module.

The maximum transmission rate is dependent on the cable length:

- 320 m → 125 kHz
- 160 m → 250 kHz
- 60 m → 500 kHz
- 20 m → 1 MHz

Selecting increments per encoder revolution

Increments per encoder revolution

The "Increments per encoder revolution" parameter defines the number of increments an absolute encoder returns per revolution.

For the limits you have to differentiate between the individual encoder models. Only values to the power of 2 are selectable.

- Single-turn encoder (number of revolutions = 1) with 13-bit frame length:
  - minimum value = 4
  - maximum value = 8192
- Multiturn encoder (number of revolutions > 1) with 25-bit message frame length:
  - Minimum value = 4
  - Maximum value = 8192

Selecting the number of encoder revolutions

Number of encoder revolutions

This parameter is used to define the maximum number of revolutions of this encoder.

The entry is only relevant for multi-turn encoders. For single-turn encoders, the 1PosU sets the number of revolutions to 1.

Valid value range:

- 2 to 4096 in powers of 2

Enabling reversal of direction of rotation

Reversal of direction of rotation

You enable reversal of direction of rotation by ticking the corresponding options box.

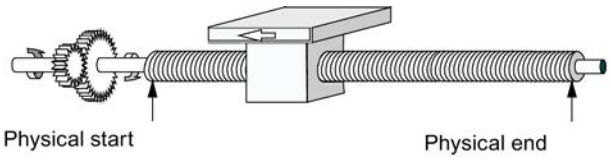
You can use the "Reversal of direction of rotation" parameter to adapt the direction of rotation of the encoder to that of the drive and the axis.

Selecting axis type

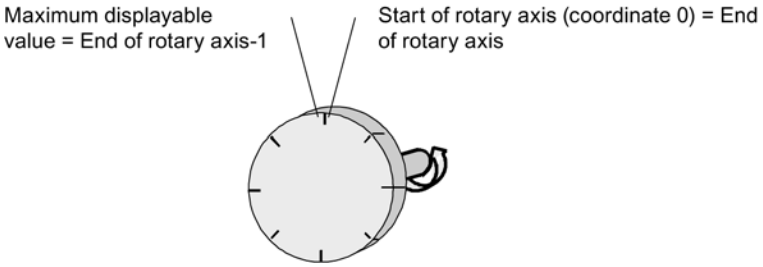
Axis types

The 1PosU positioning module can control the following axis types:

- Linear axis
  A linear axis is an axis which has a limited physical traversing range.



- Rotary axis
  The rotary axis is an axis whose traversing range is not restricted by mechanical stops.



Digital inputs

Selecting minus limit switch and plus limit switch

Function of the hardware limit switches

The two digital inputs (DI0 and DI1) are evaluated by the 1PosU as limit switches:

- DI0 is the minus limit switch and limits the operating range in the minus direction.
- DI1 is the plus limit switch and limits the operating range in the plus direction.

You can assign parameters to the hardware limit switches **separately as NC or NO contacts**.

The hardware limit switches are evaluated with linear axes and rotary axes.

Only the hardware limit switch that lies in the direction in which the drive is being moved is evaluated.

This enables you to move away from a hardware limit switch without additional error acknowledgment by moving in the other direction if you reach or overrun a hardware limit switch.

The current signal level of the digital inputs is displayed in the feedback interface, delayed by the update rate.

The following table shows how the hardware limit switches work in the individual MODEs.

MODE	Function of the hardware limit switches
Reference point approach	The 1PosU executes an automatic direction reversal at the hardware limit switch.
Inching	The motion of the axis is halted on the hardware limit switch, all 3 digital outputs are set to 0, and the POS_ERR feedback bit is reported.
Absolute Positioning	
Relative Positioning	

Starting on the hardware limit switch

The following table shows the reaction of the 1PosU when starting MODEs on the hardware limit switch.

Direction	1PosU reactions
Starting into the operating range	The 1PosU starts the specified MODE.
Starting away from the operating range	The POS_ERR = 1 feedback bit is set.

Selecting reduction cam

Function of the reduction cam

The digital input DI2 is evaluated as a reduction cam or latch input by 1PosU:

- as a reduction cam with MODE 3 "Reference point approach (Page 10659)" and with JOB 9 "Evaluate referencing signal (Page 10676)"
- as a latch input with JOB 10 "Latch function (Page 10678)"

You can assign parameters to the hardware limit switches separately as **NC or NO contacts**.

Drive

Selecting control mode

Control mode

The"Control mode" parameter defines the function of the digital outputs for the control of the drive (Page 10654). The following are selectable:

- Control mode 0:
  - DO0 travel minus
  - DO1 travel plus
  - DO2 rapid/creep feed
- Control mode 1:
  - DO0 rapid feed
  - DO1 creep feed (rapid feed is 0)
  - DO2 traverse plus (1)/ traverse minus (0)

Entering the switch-off difference

Switch-off difference

Switch-off difference (Page 10648) defines the distance from the target at which the drive is slowed down from creep speed to 0.

If the switch-off difference ≥ the changeover difference, there is no changeover point. There is no deceleration from rapid traverse to creep speed; the response is executed directly at the switch-off point.

Valid value range:

- 0 to 65,535

You can use JOB 3 (Page 10674) to change the switch-off difference.

Entering the changeover difference

Changeover difference

Changeover difference (Page 10648) defines the distance from the target at which the drive is slowed down from rapid traverse to creep speed.

Valid value range:

- 0 to 65,535

You can use JOB 4 (Page 10675) to change the changeover difference.

Entering the delay time for the direction change

T_{min} direction change

The "T_{min} direction change" parameter defines the delay time for the direction change. The digital outputs are switched off, and a change of direction by T_{min} is executed with a delay. T_{min} is effective at each change of direction during a run.

T_{min} does not work when starting after POS_DONE = 1 or POS_ERR = 1.

Valid value range:

- 0 to 255

Your input value is multiplied by 10. You thus specify T_{min} in increments of 10 ms (for example, 0 ms, 10 ms or 2550 ms).

1PosU control and feedback interface

1PosU control interface

Assignment of the Control Interface

The table below shows the assignment of the 1PosU control interface (outputs).

Address	Assignment					
Byte 0	Bits 0.7 to 0.4 stand for the MODEs					
	Bit	7	6	5	4	
		0	0	0	0	MODE 0 = Stop
		0	0	0	1	MODE 1 = Inching
		0	0	1	1	MODE 3 = Reference point approach
		0	1	0	0	MODE 4 = Relative positioning
		0	1	0	1	MODE 5 = Absolute positioning

<table>
<tr><th>Address</th><th>Assignment</th></tr>
<tr><td></td><td>Bit 3: SPEED (SPEED = 0 is creep speed; SPEED = 1 is rapid traverse)
Bit 2: DIR_M
Bit 1: DIR_P
Bit 0: START</td></tr>
<tr><td rowspan="3">Bytes 1 to 3</td><td>with MODE 3 = Reference point approach: reference point coordinates</td></tr>
<tr><td>with MODE 4 = Relative positioning: distance</td></tr>
<tr><td>with MODE 5 = Absolute positioning: target</td></tr>
<tr><td rowspan="13">Byte 4</td><td>Bits 4.7 to 4.4 stand for the JOBs
<table>
<tr><td>Bit</td><td>7</td><td>6</td><td>5</td><td>4</td><td></td></tr>
<tr><td></td><td>0</td><td>0</td><td>0</td><td>0</td><td>JOB 0 = Cancel JOB processing</td></tr>
<tr><td></td><td>0</td><td>0</td><td>0</td><td>1</td><td>JOB 1 = Set actual value</td></tr>
<tr><td></td><td>0</td><td>0</td><td>1</td><td>0</td><td>JOB 2 = Shift encoder range (only at SSI encoders)</td></tr>
<tr><td></td><td>0</td><td>0</td><td>1</td><td>1</td><td>JOB 3 = Change switch-off difference</td></tr>
<tr><td></td><td>0</td><td>1</td><td>0</td><td>0</td><td>JOB 4 = Change changeover difference</td></tr>
<tr><td></td><td>1</td><td>0</td><td>0</td><td>1</td><td>JOB 9 = Evaluate the reference signal</td></tr>
<tr><td></td><td>1</td><td>0</td><td>1</td><td>0</td><td>JOB 10 = Latch function</td></tr>
<tr><td></td><td>1</td><td>0</td><td>1</td><td>1</td><td>JOB 11 = Define the settings for monitoring the direction of rotation</td></tr>
<tr><td></td><td>1</td><td>1</td><td>1</td><td>1</td><td>JOB 15 = Display current values</td></tr>
</table></td></tr>
<tr><td>Bit 3: EXTF_ACK
Bit 2: Reserved = 0
Bit 1: Reserved = 0
Bit 0: JOB_REQ</td></tr>
<tr><td>Bytes 5 to 7</td><td>In accordance with the selected JOBs:
• with JOB 1 = Actual value coordinates:
• with JOB 2 = Offset:
• with JOB 3 = Switch-off difference
• with JOB 4 = Changeover difference
• withJOB 9 = Reference point coordinates
• with JOB 10
– Byte 5: Bit 0 = latch at positive edge at DI2
– Byte 5: Bit 1 = latch at negative edge at DI2
• with JOB 11 = Distance difference for monitoring of the direction of rotation
• with JOB 15
– Byte 5: 0 = Residual distance
– Byte 5: 1 = Actual speed
– Byte 5: 2 = Error information</td></tr>
</table>

1PosU feedback interface

Assignment of the Feedback Interface

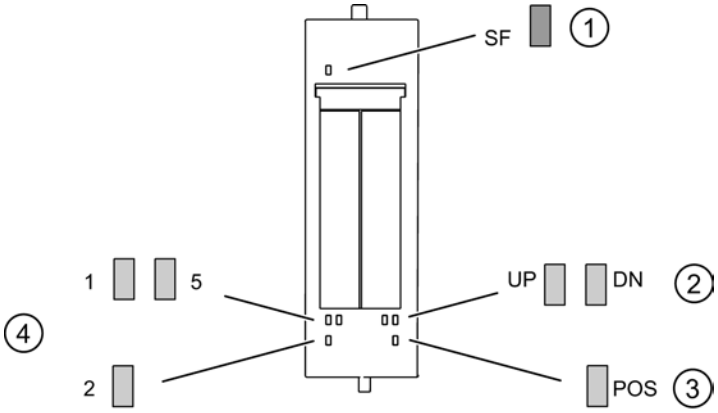
The following table shows the assignment of the 1PosU feedback interface (inputs).

Address	Assignment
Byte 0	Bit 7: ERR_ENCODER Bit 6: STATUS DO 2 Bit 5: STATUS DO 1 Bit 4: STATUS DO 0 Bit 3: SYNC Bit 2: POS_DONE Bit 1: POS_ERR Bit 0: POS_ACK
Bytes 1 to 3	Actual value
Byte 4	Bit 7: ERR_2L+ Bit 6: STATUS DI 2 Reduction cam Bit 5: STATUS DI 1 Limit switch plus Bit 4: STATUS DI 0 Limit switch minus Bit 3: Reserved = 0 Bit 2: LATCH_DONE Bit 1: JOB_ERR Bit 0: JOB_ACK
Bytes 5 to 7	Feedback value

Diagnostics of 1PosU

Diagnostics using the LED display

LED display on the 1PosU



① Group error (red)
② Status display for a change in an actual value (green)
③ Positioning underway (green)
④ Status displays for digital inputs (green)

Status and error displays by means of LEDs on the 1PosU

The table below shows the status and error displays on the 1PosU.

Event (LED)							Cause	Remedy
SF	1	5	2	UP	DN	POS		
On							No parameter assignment. Diagnostic message.	Check the parameter settings. Analyze the diagnostic data.
	On						DI 0 is activated.	
		On					DI 1 is activated.	
			On				DI 2 is activated.	
				On			In the case of actual value change from lower to higher values	
					On		In the case of actual value change from higher to lower values	
						On	Positioning is running and one of the 3 digital outputs is set.	

Error types

For information on the structure of the channel-related diagnostics, refer to the manual on the interface module used in your ET 200S station.

1PosU error types

The following table shows the error types on the 1PosU.

Error type		Meaning	Remedy
1_D	00001: Short circuit	Short circuit of the sensor supply.	Check the wiring to the sensor. Correct the process wiring.
16_D	10000: Parameter assignment error	Parameters have not been assigned to the module.	Correct the parameter assignment.
17_D	10001: Load voltage 2L + missing	Power supply voltage not present or too low.	Correct the process wiring. Check the supply voltage.
26_D	11010: External error	Wire break/short circuit of the sensor signals. Wire break in the sensor cable, or sensor cable is not connected. Sensor is defective or there are faults. Sensor type, transmission rate, and monoflop time do not correspond to the sensor connected; programmable sensors do not correspond to the settings on the module.	Correct the process wiring. Correct the parameter assignment. Replace the sensor.

11.4.2.8 Using ET 200S 1STEP 5V

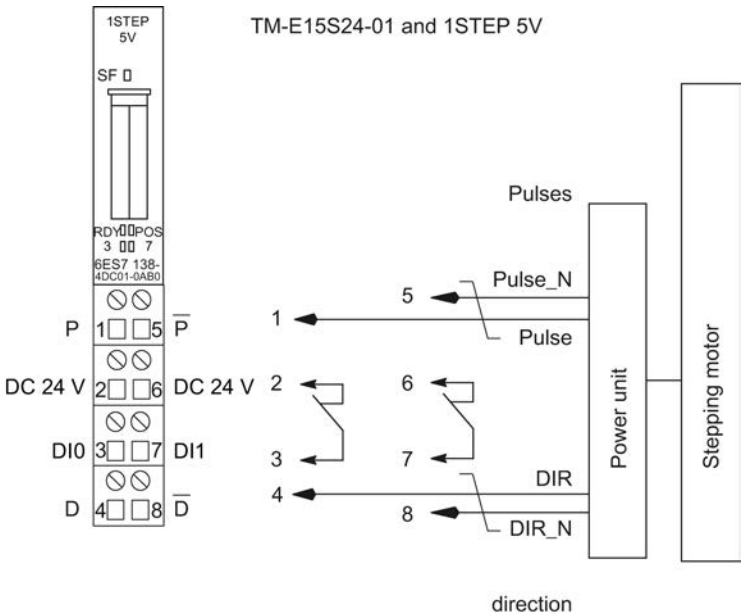
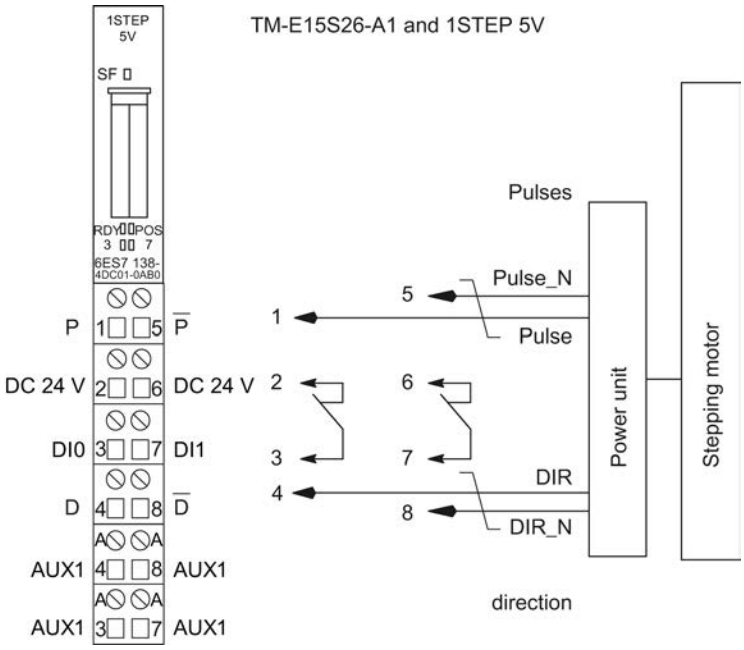
Terminal assignment of the 1STEP 5V

Wiring rules

The cables (terminals 1 and 5 and terminals 4 and 8) to the power unit must be shielded, twisted-pair cables. The shield must be supported at both ends. To do this, use the shield contact (see the appendix of the ET 200S Distributed I/O System (http://support.automation.siemens.com/WW/view/en/1144348) operating instructions).

Terminal Assignment

The following table shows the terminal assignment of 1STEP 5V.

View	Terminal Assignment	Remarks
		24 V DC: Load voltage DI0: external enable pulse, external STOP or limit switch in forward/reverse direction DI1: Reference switch or reference switch and limit switch in forward/ reverse direction P, /P and D, /D are signals to RS 422.
		24 V DC: Load voltage DI0: external enable pulse, external STOP or limit switch in forward/reverse direction DI1: Reference switch or reference switch and limit switch in forward/ reverse direction P, /P and D, /D are signals to RS 422.

Fundamentals of positioning with stepper motor

Interaction of components

Below is a description of how individual components for positioning with stepping motor affect each other.

1STEP 5V

The 1STEP 5V generates pulses and a directional signal for the power units of stepping motors. The number of pulses emitted determines the distance traversed. The pulse frequency determines the velocity. The change in the pulse frequency per time unit (second) is a measure for the acceleration or deceleration. 1STEP 5V is influenced by its parameters and settings.

Power unit for stepping motors

The power unit is the link between the 1STEP 5V and the stepping motor. The 1STEP 5V sends RS 422 differential signals for frequency and direction. These signals are converted in the power unit into motor currents that control the movements of the motor with a very high degree of precision.

Stepping motors

Stepping motors are used to position axes. They represent the simple and cost-effective solution for precision positioning jobs in wide performance ranges.

A stepping motor shaft turns by a certain angle with every pulse. During rapid pulse sequences, this stepping movement becomes a continuous turning motion.

Parameters and Settings

Required Information

To ensure optimum interplay between the individual components, you must provide the 1STEP 5V with information:

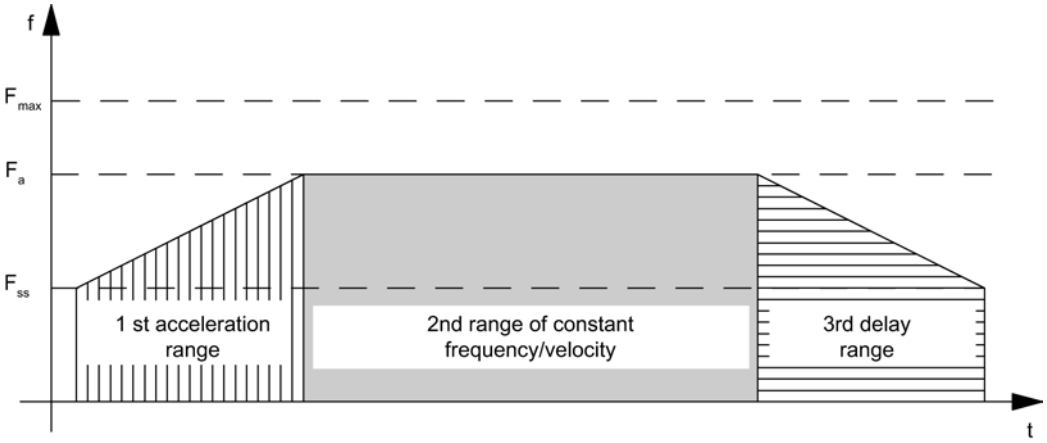
- **One time: during parameter configuration using your configuration software**
  - Base frequency (Page 10717) F_b
  - Multiplier n (Page 10714) for setting the start-stop frequency F_{ss}
  - Multiplier i (Page 10714) for setting the acceleration / delay
  - Function and behavior of the digital inputs (Page 10733)
  - Traversing range (Page 10729)
- **In operation: movement of the motor by means of a traversing job in your user program**
  - Multiplier G (Page 10714) for the velocity/output frequency F_a
  - Reduction factor R (Page 10717)
  - Distance, position or frequency
  - Operating mode
  - Direction specification (traversing job) to the start
- **In operation: to adjust to different load conditions as a parameter assignment request in your user program**
  - Base frequency (Page 10717) F_b
  - Multiplier n (Page 10714) for setting the start-stop frequency F_{ss}
  - Multiplier i (Page 10714) for setting the acceleration / delay

Traversal curve of the 1STEP 5V

Traversal curve

An incremental mode is carried out by the 1STEP 5V in accordance with the following traversal curve. The 1STEP 5V forms the fundamental parameters (start-stop frequency, output

frequency, and acceleration/delay) of the traversal curve with a base frequency (Page 10717) that you select.



F_{ss} = start-stop frequency
F_a = output frequency
F_{max} = maximum frequency of the stepping motor

Start-Stop Frequency F_{ss}

The start-stop frequency F_{ss} is the frequency to which the motor can be accelerated under load from a standstill.

The size of F_{ss} depends on the load inertia. The best way to work out the load inertia is by trial and error.

The start-stop frequency F_{ss} is simultaneously the minimum output frequency F_a needed to move the stepping motor.

Setting the Start-Stop Frequency F_{ss}

Through parameter assignment, the 1STEP 5V permits the start-stop frequency F_{ss} to be set in steps. To do so, select the multiplier n between 1 and 255, which is multiplied with the base frequency F_b. You can lower the start-stop frequency F_{ss} again with the reduction factor R (1 or 0.1) in the traversing job.

The start-stop frequency is calculated according to equation:

$F_{ss} = F_b \times n \times R$

Additional information is available in the following table (Page 10717) with the ranges for start-stop frequency, starting frequency and acceleration.

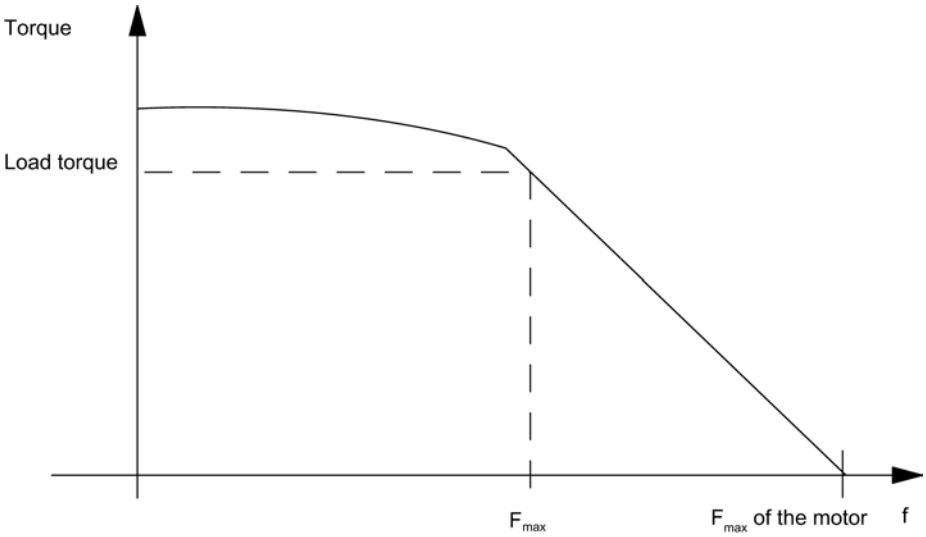
Maximum frequency / velocity of the Axis F_{max}

When you choose a stepping motor, remember the following:

The maximum frequency/velocity is determined by your application. At this frequency, the motor must reach a torque high enough to move its load.

Note this does not mean the highest possible frequency that the motor or the power unit can tolerate.

You can work out the maximum frequency F_{max} with the corresponding characteristic curve. The following figure shows the torque characteristic curve of a stepping motor.



Output frequency / velocity F_a

The output frequency can be set differently for each run.

When selecting the output frequency, also observe the minimum pulse duration of your power unit (see Formula and table (Page 10717) with the ranges for start-stop frequency, output frequency and acceleration).

If the selected output frequency is lower than the set start-stop frequency F_{ss}, the 1STEP 5V output frequency is set to the start-stop frequency F_{ss}.

F_a must always be smaller than F_{max}.

Setting of the output frequency / Velocity F_a

The 1STEP 5V permits the output frequency F_a to be set in steps. To do so, select the multiplier G between 1 and 255, which is multiplied with the base frequency F_b. You can lower the output frequency F_a again with the reduction factor R (1 or 0.1) in the traversing job.

The output frequency is calculated according to the equation:

$F_a = F_b \times G \times R$

Additional information is available in the following table (Page 10717) with the ranges for start-stop frequency, starting frequency and acceleration.

Acceleration / delay a

The maximum permitted acceleration / delay depends on the load to be moved.

The motor must reach a torque high enough to accelerate or delay the load without loss of step.

Depending on the application, you must also take into account additional criteria for setting the acceleration/delay, such as smooth starting and stopping.

Setting the acceleration / delay a

Through parameter assignment, the 1STEP 5V permits the acceleration / delay to be set in steps by means of the multiplier i.

During the acceleration phase, the frequency is increased continuously starting from the start-stop frequency F_{ss} until the output frequency F_a has been reached.

The time interval for the continuous increase in frequency can be set in steps. To do so you select a multiplier i between 1 and 255.

In the delay phase, the output frequency is reduced in the same way.

You can lower the acceleration / delay a again with the reduction factor R (1 or 0.1) in the traversing job.

The acceleration / deceleration is calculated according to the equation:

$a = F_b \times R / (i \times 0.128\ ms)$

Additional information is available in the following table (Page 10717) with the ranges for start-stop frequency, starting frequency and acceleration.

Setting the Base Frequency

Introduction

Through parameter assignment, the 1STEP 5V permits the base frequency to be set in steps.

The base frequency sets the range for the start-stop frequency, the output frequency, and the acceleration.

Procedure

1. Depending on the priority of your requirements select a suitable range either of the start-stop frequency F_{ss} and of the starting frequency F_a or of the acceleration a from the following table in accordance with the following criteria:
   - Range for the start-stop frequency F_{ss},
     for example, for starting and stopping as soon as possible
   - Range for the output frequency F_a,
     for example, for a velocity setting that is as precise as possible
   - Range for the acceleration a,
     for example, for the fastest possible positioning operations
2. Use the table to determine the base frequency F_b.
   To optimize the base frequency F_b, proceed as follows:

3. Check whether the other corresponding values meet your requirements. If necessary, select another base frequency F_b, which meets your requirements better.
4. Define the multipliers required to set the output frequency F_a, the acceleration/delay a, and the start-stop frequency F_{ss}.
5. Determine the corresponding reduction factor R from the table.

The following table shows the ranges for the start-stop frequency, output frequency, and acceleration.

Base frequency F_b: in Hz	Reduction factor R	Range Start-stop frequency F_{ss} Starting frequency F_a in Hz	Range Acceleration a in Hz/ms
		Equation: $F_{ss} = F_b \times n \times R$ $F_a = F_b \times G \times R$	Equation: $a = F_b \times R / (i \times 0.128\ ms)$
4	0,1	0,4 … 102	0,01 … 3,13
8	0,1	0,8 … 204	0,02 … 6,25
20	0,1	2 … 510	0,06 … 15,6
4	1	4 … 1020	0,12 … 31,3
8	1	8 … 2040	0,25 … 62,5
20	1	20 … 5100	0,61 … 156
40	1	40 … 10200	1,23 … 313
80	1	80 … 20400	2,45 … 625
200	1	200 … 51000	6,13 … 1563
400	1	400 … 102000	12,25 … 3125
800	1	800 … 204000	24,51 … 6250
2000 [1]	1	2000 … 510000	61,27 … 15625

F_b = Base frequency
F_{ss} = Start-Stop frequency
F_a = Output frequency
a = Acceleration / delay
R = Reduction factor
n = Multiplier for setting the start-stop frequency in steps
G = Multiplier for setting the output frequency in steps
i = Multiplier for setting the acceleration / delay in steps

[1] as of 6ES7138-4DD01-0AB0

The minimum pulse duration results from the set starting frequency F_a and is calculated using the equation $T_{Pulse} = 1 / (2 \times F_a)$.

Functions of 1STEP 5V

Overview of functions

Overview

The job of the 1STEP 5V is to position a drive on certain predefined targets (incremental modes) and to travel continuously with specifiable frequencies (speed-control mode).

The following functions are available to you to this purpose:

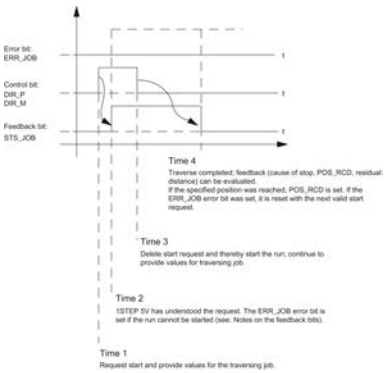
- Reference point approach (Page 10720): The axis is synchronized to a reference point
- Set reference point (Page 10724) (as of 6ES7138-4DC01-0AB0): A value is assigned to the current position.
- Relative incremental approach (Page 10724) (relative positioning): The axis is moved by a predefined distance.
- Absolute incremental approach (Page 10725) (absolute positioning) (as of 6ES7138-4DC01-0AB0): The axis is traveled to a predefined position.
- Speed mode (Page 10726) (as of 6ES7138-4DC01-0AB0): The drive is moved with a speed that can be specified flexibly (pulse frequency).
- Hold traversing job (Page 10728)
- Changing Parameters during Operation (Page 10732)

The description of the control interface (Page 10741) provides information about how to control the functions.

Starting a traversing job

Starting the traversing job

The figure below shows the process from start request for a traversing job up to the end of the traversing.



Evaluating the ERR_JOB error bit

As soon as the feedback bit (Page 10744)STS_JOB is cleared at time stamp 4, evaluate the ERR_JOB error bit. Note that the feedback bit STS_JOB is only cleared if the control bits (Page 10741)DIR_P, DIR_M and C_PAR are cleared.

Search for Reference

Description of the function

The home position marks the point of reference of your drive system (reference cam) for the following traversing jobs. You can determine the home position by, for example, installing an initiator on the reference cam and connecting it to the DI1 digital input.

The 1STEP 5V ensures the reference point can be reproduced accurately in that it is always approached from the same direction. You can specify this direction by always starting the search for reference in the same direction.

Traversing job for reference point approach

The traversing job contains the following information:

- Multiplier G for the velocity/output frequency F_a
- Reduction factor R for the assigned parameters base frequency F_b
- Reference point position
- Mode = 1 for reference point approach
- Stop at the reference cam (Page 10728)
- Direction specification as start (Page 10741)

Note

The 1STEP 5V checks the set position for limits (minimum 0 and maximum 16777215). The full-scale value can be configured.

Note

If you have configured the behavior of the digital input (Page 10733) DI1 (7) as a "Reference switch and limit switch", the 1STEP 5V automatically selects the starting direction toward the limit switch, irrespective of the direction specified in the traversing job.

Status bit SYNC

The status bit (Page 10744)SYNC informs you that the axis has been synchronized, that is, after the correct homing procedure, this status bit is set and deleted during the run.

The SYNC status bit is deleted

- After parameter assignment of your ET 200S station
- After deletion of the pulse enable
- After a CPU/Master STOP

In these cases it is advisable to carry out a search for reference.

POS and POS_RCD status bits

While reference point approach is active, it is indicated by the set feedback bit (Page 10744)POS.

On completion of a reference point approach, the set POS_RCD feedback bit indicates that the position has been reached.

If the reference point approach is interrupted, the POS_RCD feedback bit remains reset.

Residual distance, position, frequency

The residual distance reported (Page 10744) is irrelevant during the reference point approach.

Note

In order for the 1STEP 5V to approach the home position with repeated precision, the period duration of the start-stop frequency has to be greater than the runtime of a single step from the 1STEP 5V to the stepping motor and via the reference cam back to 1STEP 5V (input delay of digital inputs: 4 ms).

When stopping at the reference cam or at one of the limit switches during the acceleration phase, the 1STEP 5V continues to send pulses for a maximum of 50 ms at the frequency already reached before it starts braking. This avoids abrupt changes in frequency, which can lead to step losses.

Sequence of Execution of the Search for Reference

Steps of the Search for Reference

A search for reference consists of a maximum of three sections.

In the **first section** (1) and **second section** (2), the system ensures that the reference cam is found.

Both these sections are traversed at the defined output frequency F_a.

In the **third section** (3), the reference cam is approached with start-stop frequency F_{ss} in the selected direction up to the reference point ⊕ with reproducible accuracy.

Note

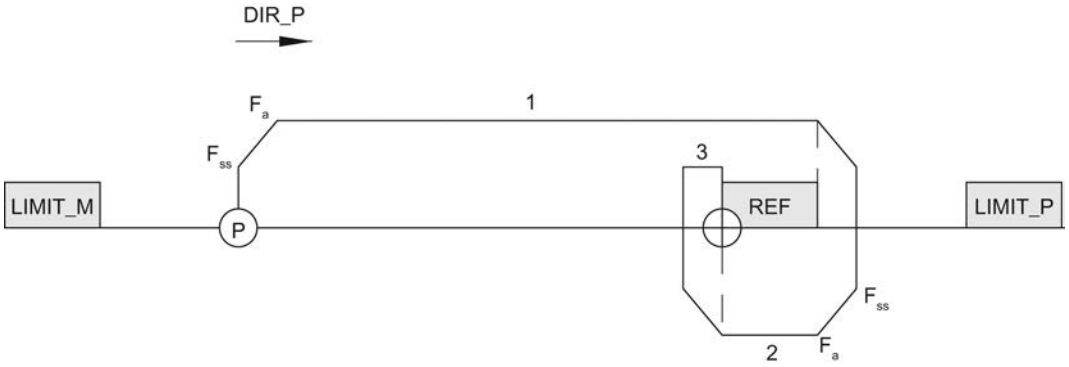
The maximum number of output pulses of a section is the configured length of the traversing range minus 1.

Different Execution Sequences

Depending on the position (P) at the start of the reference point approach, there are different execution patterns for the run (REF is the reference cam, which is wired to the DI1 digital input). The illustration applies to the forward starting direction (DIR_P).

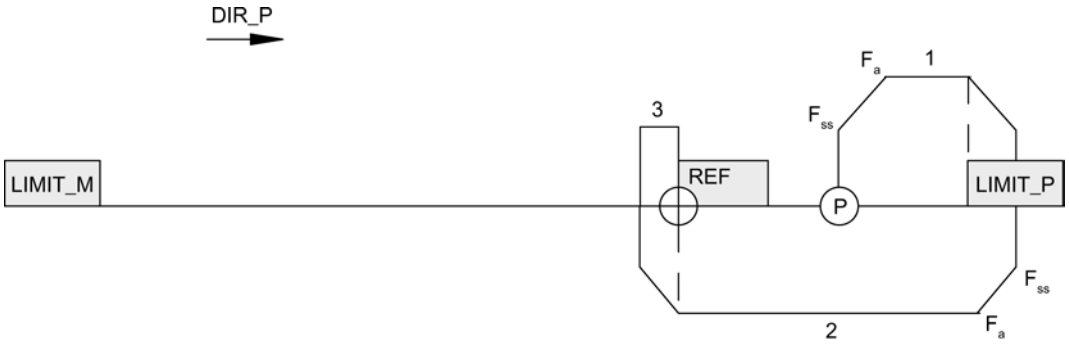
- **Start before REF or at limit switch LIMIT_M**

The figure below shows a reference point approach with start before the REF.



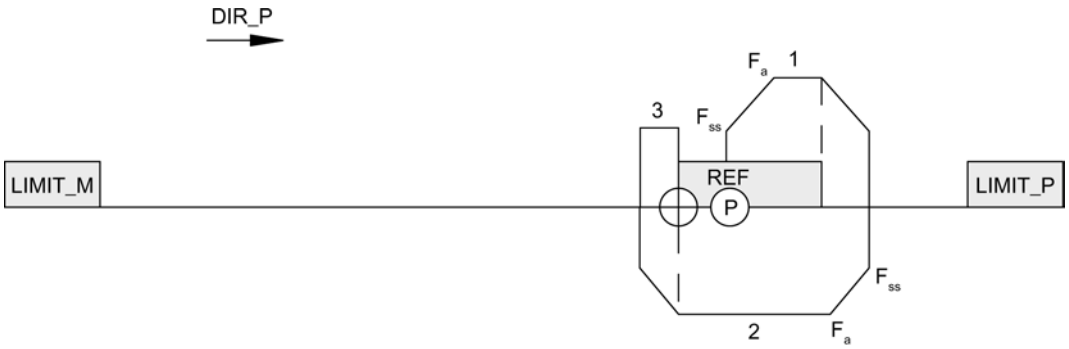
- **Start after REF**

The figure below shows a reference point approach with start after the REF.



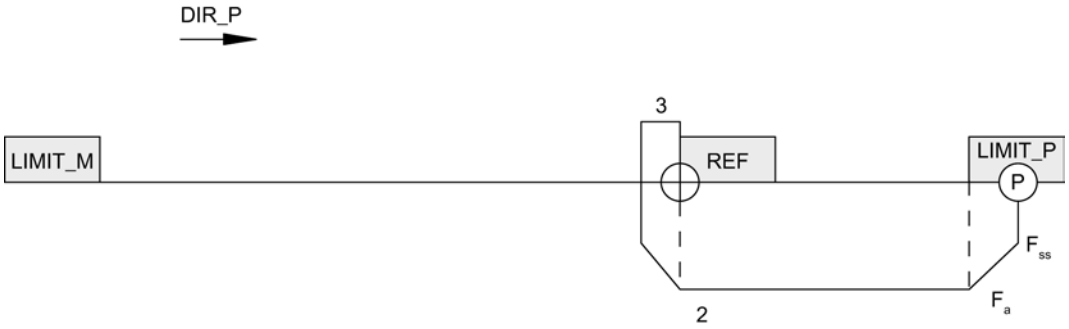
- **Start on the REF**

The figure below shows a reference point approach with start on the REF.



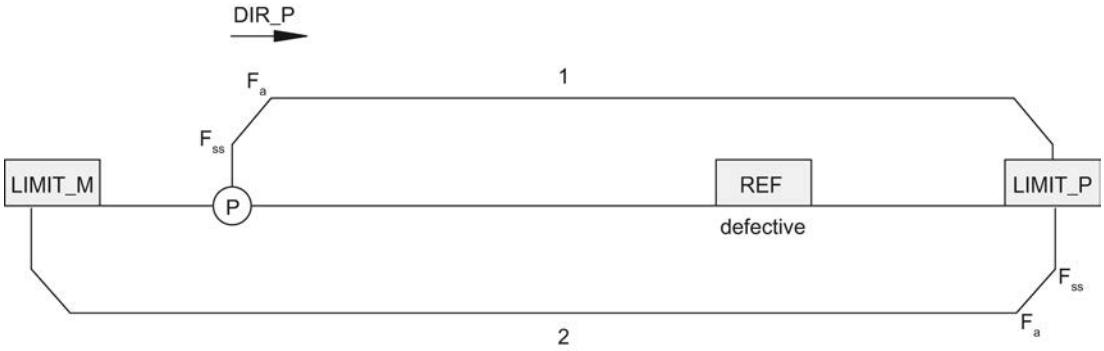
- **Start at the limit switch in start direction**

The figure below shows a reference point approach with start on the limit switch in start direction.

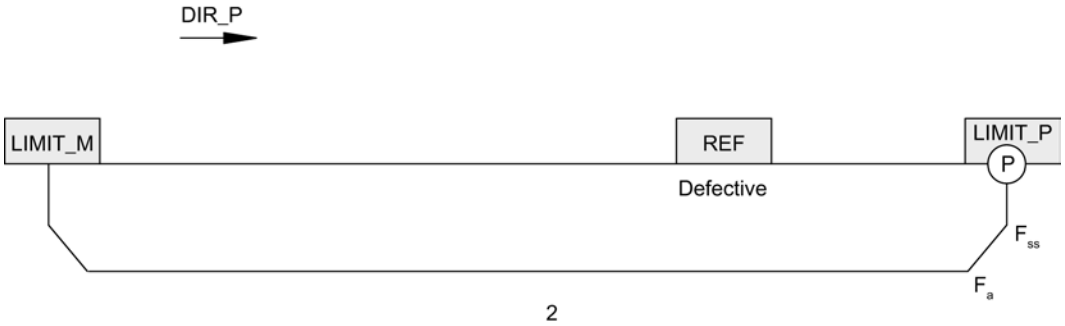


Response to defective reference cam with limit switch (interruption of traversing)

The figure below shows a reference point approach with defective reference cam and start before the REF.



The figure below shows a reference point approach with defective reference cam and start on the LIMIT_P.



Behavior in the case of a constantly set reference cam without limit switch

At the end of the first section, after 16777215 pulses have been output, traversing is terminated with cleared SYNC and POS_RCD status bits.

Response to failure of the reference cam without limit switch

All three sections of traversing are executed, each with output of 16777215 pulses. Afterwards, the search is interrupted with cleared SYNC and POS_RCD status bits.

Set reference point (as of 6ES7138-4DC01-0AB0)

Description of functions

The home position marks the reference point of your drive system which the subsequent absolute incremental modes and the position value in the feedback interface reference. You define the home position by specifying the absolute position value for the current position of the stepping motor.

Job for setting the home position

A job for setting the home position is a virtual job without traversing movement. It contains the following information:

- Position of the home position
- Mode = 4 for setting home position
- Any direction specification as start (Page 10741)

Note

The 1STEP 5V checks the set position for limits (minimum 0 and maximum 16777215). The full-scale value can be configured.

Feedback messages

Correct execution of the job is indicated by the set feedback bits (Page 10744)SYNC and POS_RCD.

Relative incremental mode (relative positioning)

Description of functions

You can use the relative incremental mode to move the stepping motor a defined distance and thus approach a specified position.

You can determine the direction of traversing and the velocity at the start.

Traversing job for relative incremental mode

The traversing job contains the following information:

- Distance (number of pulses to be emitted)
- Multiplier G for the velocity/output frequency F_a
- Reduction factor R for the assigned parameters base frequency F_b
- Mode = 0 for incremental mode, relative
- Stop at the reference cam (Page 10728)
- Direction specification as start (Page 10741)

Note

The 1STEP 5V checks the specified distance for limits (minimum 1 and maximum 16777215 pulses). The distance to the limit switch is not checked by the 1STEP 5V. Traversing is stopped at the latest when the limit switch is reached.

Feedback messages

The feedback bit (Page 10744)POS_RCD is reset at the beginning of incremental mode.

While the incremental mode is active, it is indicated by the set POS feedback bit.

After incremental mode has been correctly executed, the set POS_RCD feedback bit indicates that the position has been reached.

If the incremental mode is interrupted, the POS_RCD feedback bit remains reset. After incremental mode has been stopped, the distance still to be traversed is displayed if the feedback value (Page 10744) is set to "Residual distance".

Absolute incremental approach (absolute positioning) (as of 6ES7138-4DC01-0AB0)

Description of functions

You can use the absolute mode to move the stepping motor to a defined position and thus approach a specified position.

The velocity is specified at the start. The direction and the distance of traversing are determined automatically by the 1STEP 5V on the basis of the starting position (actual position value). You can also specify the direction for a modulo axis.

Note

If you set Forward start and Backward start (DIR_P and DIR_M) simultaneously at a modulo axis, the 1STEP 5V then automatically selects the shortest distance to reach (Page 10729) the target position.

Traversing job for absolute incremental mode

The traversing job contains the following information:

- Target position
- Multiplier G for the velocity/output frequency F_a
- Reduction factor R for the assigned parameters base frequency F_b
- Mode = 2 for incremental mode
- Any direction specification as start (Page 10741)

Note

The 1STEP 5V checks the set position for limits (minimum 0 and maximum 16777215). The full-scale value can be configured.

The traversing job is only executed if you have determined or specified the position of the home position beforehand (the SYNC bit has to be set, see "Reference point approach (Page 10720)" or "Set reference point (Page 10724)").

The control signal (Page 10741) "Stop at reference cam" is not taken into consideration.

Feedback messages

The feedback bit (Page 10744)POS_RCD is reset at the beginning of incremental mode.

While the incremental mode is active, it is indicated by the set POS feedback bit.

After incremental mode has been correctly executed, the set POS_RCD feedback bit indicates that the position has been reached.

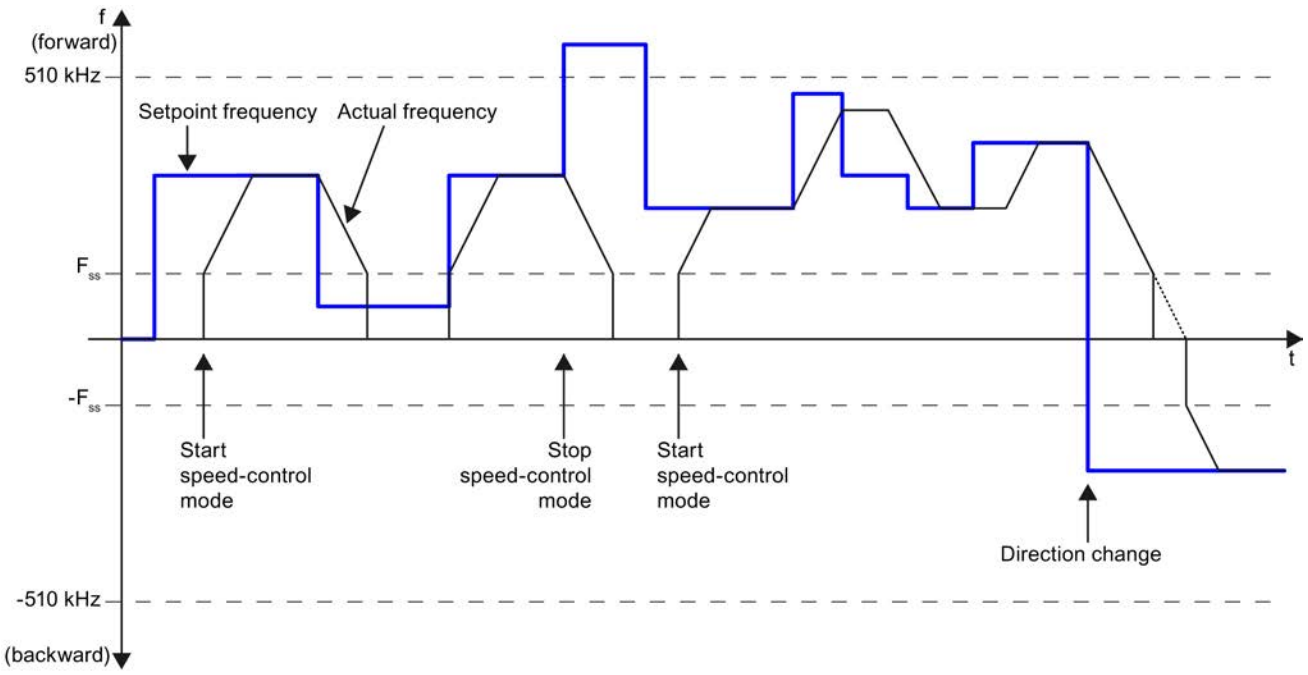
If the incremental mode is interrupted, the POS_RCD feedback bit remains reset. After incremental mode has been stopped, the distance still to be traversed is displayed if the feedback value (Page 10744) is set to "Residual distance".

Speed mode (as of 6ES7138-4DC01-0AB0)

Description of functions

In this operating mode you specify the frequency with which the pulses (steps) are output. When you change the frequency, the pulses are output with the new frequency after an acceleration or deceleration phase. The output is carried out continuously until you stop the traversing job or a traversing range is reached at a linear axis.

The figure below shows the speed-control mode with a modulo axis.



Traversing job for speed-control mode

The traversing job contains the following information:

- Setpoint frequency as 32-bit value (STEP 7 data type REAL)
- Direction specification by the sign of the setpoint frequency (positive: forward)
- Mode = 3 for speed-control mode
- Any direction specification as start (Page 10741)

Note

The 1STEP 5V checks the set position for limits (minimum -510,0 kHz and maximum +510,0 kHz).

The specified frequency is approached with the configured acceleration a under consideration of the start-stop frequency F_{ss}. No pulse output is emitted at frequencies that are less than F_{ss}.

The continuous output of the frequency is terminated at the following events:

- Reaching of the limits of the configured traversing range (0 in the direction backward) unless a modulo axis is configured
- Other aborting conditions for traversing jobs (Page 10728)

Feedback messages

While the traversing job is active, it is indicated by the set feedback bit (Page 10744)POS.

When a new frequency is specified, the POS_RCD feedback bit is cleared. When the new frequency has been reached after the acceleration or deceleration phase, POS_RCD is set again.

The current frequency is displayed in the feedback interface as a 32-bit value (STEP 7 data type REAL) if the feedback value (Page 10744) is set to "Frequency".

Hold traversing job

Specific holding of the traversing job

- Caused by	Displayed by Feedback Bit
STOP by control bit	-
External STOP at digital input	STOP_EXT
Limit switch in the forward direction reached (LIMIT_P or digital input)	STOP_LIMIT_P
Limit switch in the backward direction reached (LIMIT_M or digital input)	STOP_LIMIT_M
STOP at the reference cam	STOP_REF

Note

Remember that the limit switches are used in the reference point approach mode also to search for the reference cam.

If a traversing job to LIMIT_P was stopped, then you can move the axis away from the limit switch using a new traversing job with DIR_M. When stopping at LIMIT_M, you can move the axis away from the limit switch with DIR_P.

Stop at the reference cam

If the "Hold at reference cam" function is selected (the control bit STOP_REF_EN is set) at the start of traversing and the reference cam is detected during traversing, the stepping motor is halted and traversing is terminated.

Holding the traversing job in exceptional circumstances

In the following cases the traversing job is halted with loss of the synchronization:

- Incorrect operation in the control interface during an active traversing job
- External error ERR_24V through overload of the encoder supply (e.g. short circuit)
- CPU/Master STOP
- At linear axis: Reaching of the limit of the traversing range

Effects

If one of the above reasons for holding the current positioning operation occurs, it is terminated by a deceleration ramp.

The return value continues to be updated even when the traversing job is halted in exceptional cases. This enables you to traverse the residual distance after holding by means of a new traversing job in the "Relative incremental mode".

Limit Switches and External STOP

By assigning parameters, you can choose to wire normally open or normally closed contacts for the external STOP and the limit switches.

- **Normally closed contact means:**The external STOP and the effect of the limit switches are triggered by a 0 signal. When the limit switches are reached, delete the associated control bit.
- **Normally open contact means:**The external STOP and the effect of the limit switches are triggered by a 1 signal. When the limit switches are reached, set the associated control bit.

Note

In case of holding during the acceleration phase the 1STEP 5V continues to send pulses for a maximum of 50 ms at the frequency already reached before it starts braking. This avoids abrupt changes in frequency, which can lead to step losses.

Axis type and traversing range (as of 6ES7138-4DC01-0AB0)

Overview

During configuration you specify the axis type to be controlled by the stepping motor controlled by 1STEP 5V. You have the choice of the following axis types:

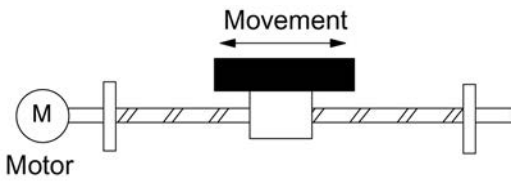
- Linear axis
- Modulo axis

Description of the function

Linear axis

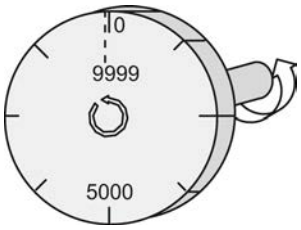
The traversing range of a linear axis can be set. The low limit is always 0, the high limit is configured and has a value range from 1 to 16777215.
The traversing range can be restricted further using limit switches (operating range).

The following figure shows a linear axis.



Modulo axis

A modulo axis is a particular form of the rotary axis. The following figure shows a modulo axis.



End of the modulo axis

The "Traversing range" parameter is used to specify the end of the modulo axis.

The actual position value cannot reach the traversing range value, because this highest value lies physically at the same position as the start of the modulo axis (0).

Example:

You specify the value 10000 as the traversing range, see figure above.

During a forward movement the position value jumps in the feedback interface from 9999 to 0, during a backward movement from 0 to 9999.

Reference point approach

If you have selected the modulo axis during the configuration and have assigned a reference cam to your drive system, you can carry out a reference point approach (Page 10720).

Traversing is aborted unsuccessfully if the reference cam is not found after the output of a number of pulses that corresponds to the configured traversing range. The status bits (Page 10744)SYNC and POS_RCD will then remain deleted.

Set home position

You may only specify values from 0 to the end of the configured end of the traversing range – 1 for the position of the home position.

Relative positioning

The end of the traversing range (end of the modulo axis) may be exceeded in both directions.

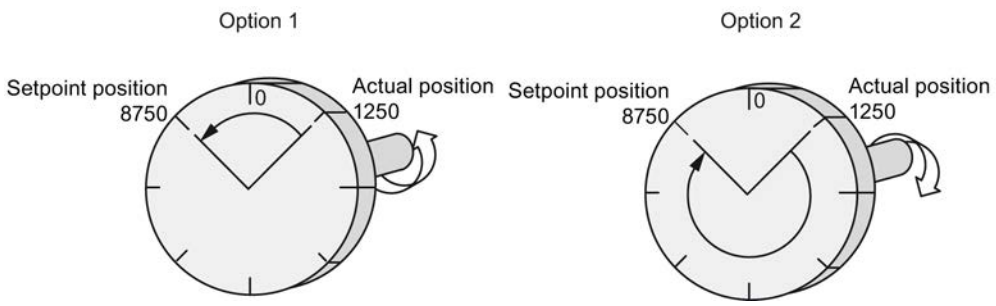
Absolute positioning

If you have selected the modulo axis during the configuration, you may only specify values from 0 to the configured end of the traversing range – 1 for the target position.

In contrast to the linear axis, specify on which path the target position is to be reached by means of the direction specification (Page 10741) when the traversing job is started:

- Backward start (DIR_M): The 1STEP 5V approaches the target position in the direction of lower actual position values (Option 1 in the following figure).
- Forward start (DIR_P): The 1STEP 5V approaches the target position in the direction of higher actual position values (Option 2 in the following figure).
- Forward start and backward start simultaneously (DIR_P and DIR_M): The 1STEP 5V automatically selects the shortest path for reaching the target position (Option 1 in the following figure).

The figure below shows absolute positioning with a modulo axis.



Pulse Enable

Description of the function

Pulse enable permits the output of pulses from the 1STEP 5V to the power unit. A run is not possible without pulse enable.

Activating Pulse Enable

You can activate pulse enable by one of the following methods:

- Through the digital input DI0 when "Function DI0" is configured as an external enable pulse (Page 10733)

or

- Through the control bit DRV_EN when the "Function DI0" is configured as an external STOP (Page 10733) or limit switch forward or backward (Page 10733)

You can recognize the assigned pulse enable through the fact that

- The LED (Page 10747)RDY at the 1STEP 5V lights up in the case of correct configuration.
- The feedback bit (Page 10744)STS_DRV_EN is set.

Deleting the Pulse Enable

Deleting the pulse enable during a run terminates the run immediately because no more pulses are emitted to the power unit. The residual distance and actual position value are no longer valid. The synchronization of the axis by means of the reference point is lost. The SYNC feedback bit and the RDY LED are deleted.

Deleting the pulse enable when the motor is at standstill deletes the SYNC feedback bit and the RDY LED.

In this case, it may be necessary to carry out a reference point approach (Page 10720).

Changing Parameters during Operation

Introduction

You can change several of the 1STEP 5V parameters during operation without having to reassign the parameters of the whole ET 200S station.

Parameters That Can Be Changed

The following parameters can be changed:

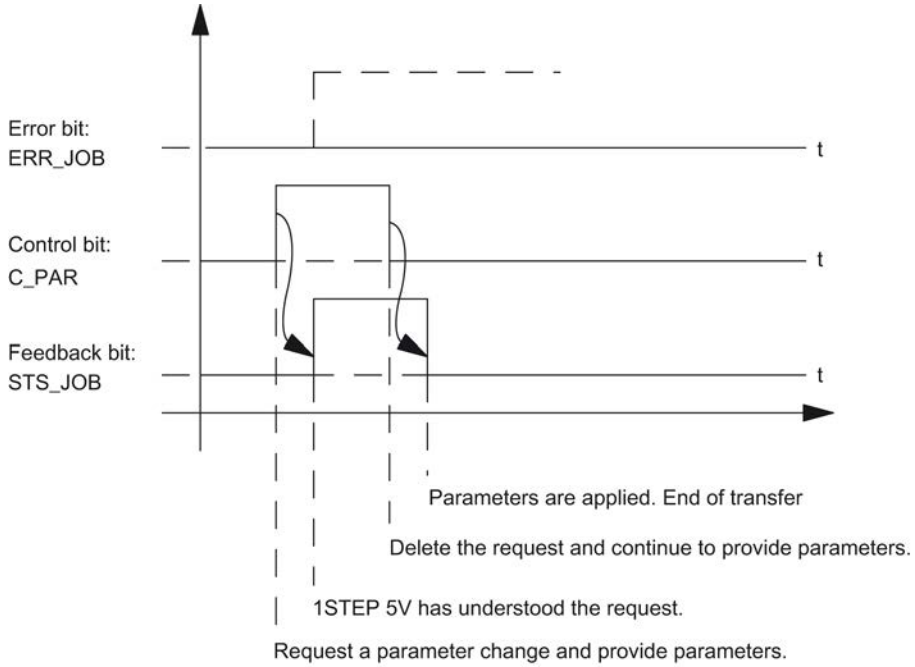
- Base Frequency $F_{b:}$
- Multiplier n for start-stop frequency F_{ss}
- Multiplier i for acceleration / delay
- Feedback value in the feedback interface

When you start changing parameters by means of the control bit (Page 10741)C_PAR, the parameters are checked for permitted values. If you have set invalid values, the feedback bit (Page 10744)ERR_JOB is set.

Only the feedback bits for the ERR_JOB and STS_JOB job processing are affected by the configuration job.

Carrying Out a Parameter Change

The following figure shows the process for a parameter change.



Note

Only one of the following control bits can be set at a particular time:

DIR_Por DIR_Mor C_PAR.

Otherwise, the ERR_JOB error is reported. The job error message is deleted by the start of the next job.

Behavior of the Digital Inputs

Introduction

You can configure the function and the behavior (active level) of the digital inputs DI0 (3) and DI1 (7) . These parameters cannot be changed using the user program.

Digital input DI0 (3)

You can configure the function of the digital input DI0 (3) as:

- External enable pulse
- External STOP

- Limit switch plus: In the forward direction
- Limit switch minus: In the backward direction

You can also configure the behavior of the digital input DI0 (3) as:

- Normally closed contact
- Normally open contact

Digital input DI0 (3) as an external enable pulse

The input must be set (Page 10731) (activated) during operation. If the input is set and the configuration correct, the 1STEP 5V is ready for operation.

Digital input DI0 (3) as an external STOP

With this input function, you can halt (Page 10728) a current traversing job using an external signal.

Digital input DI0 (3) as limit switch plus or minus

With these input functions, you can limit the traversing range in the forward or backward direction through an external signal. The signal has the same effect as one of the two control bits LIMIT_P or LIMIT_M in the control interface (Page 10741).

Digital input DI1 (7)

You can configure the function of the digital input DI1 (7) as:

- Reference switch: Reference cam
- Reference and limit switch plus: In the forward direction
  This parameter selection is only possible if "Function DI0" is not configured as a "Limit switch plus".
- Reference and limit switch minus: In the backward direction
  This parameter selection is only possible if "Function DI0" is not configured as a "Limit switch minus".

You can also configure the behavior of the digital input DI1 (7) as:

- Normally closed contact
- Normally open contact

Digital input DI1 (7) as a reference switch

You can wire a switch to this input for the reference cam.

You need a reference cam for the following:

- For a reference point approach
- For an incremental mode with hold on the reference cam.

Digital input DI1 (7) as a reference switch and limit switch plus or minus

With these input functions, you additionally limit the traversing range in the forward or backward direction through the reference cam. The signal also has the same effect as one of the two control bits LIMIT_P or LIMIT_M in the control interface (Page 10741).

If you have configured the behavior of the digital input DI1 (7) as a "Reference and limit switch", then 1STEP 5V automatically selects the starting direction toward the limit switch, regardless of the direction specified in the traversing job.

Reaction to CPU/Master STOP

Introduction

The 1STEP 5V detects the CPU/Master STOP. It responds to it by stopping the active traversing job (Page 10728).

Exiting the CPU/Master STOP status

ET 200S station	1STEP 5V
Without reconfiguration of the ET 200S station	• The feedback interface of the 1STEP 5V remains current. • The values changed by means of parameter assignment job are maintained. • If a control bit was set (DIR_P, DIR_M, C_PAR) when the CPU/Master STOP occurred, the bits STS_JOB and ERR_JOB are set when the CPU/ Master STOP status is exited. Delete the control bit. Traversing / the parameter assignment job is not executed. You can then start a new traversing by means of the control bit. • After the delay ramp, the pulse enable, the RDY LED, and the SYNC status bit are deleted.
With reconfiguration of the ET 200S station	• Information on previous searches and parameter assignment jobs is reset. • If pulse enable was activated by means of the control bit DRV_EN at the time of the CPU/Master STOP, the pulse enable, the RDY LED, and the SYNC status bit are deleted after the delay ramp.

Reconfiguration of the ET 200S station

Reconfiguration of the ET 200S station is carried out by your CPU/ DP master at:

- POWER ON of the CPU / DP master
- POWER ON of the IM 151 / IM 151 FO
- After failure of the DP transmission

- Upon loading changed parameters or configuration of the ET 200S station into the CPU / DP master
- When the 1STEP 5V is plugged
- Upon power on or inserting of the appropriate power module

See also

Pulse Enable (Page 10731)

Behavior of the digital inputs (-4DC00-)

Digital input DI 3

You can configure the function of the digital input DI 3 as:

- External enable pulse
- External STOP

You can also configure the behavior of the digital input DI 3 as:

- Normally closed contact
- Normally open contact

Digital input DI 3 as an external enable pulse

The input must be set (Page 10736) (activated) during operation. If the input is set and the configuration correct, the 1STEP 5V/204kHz is ready for operation.

Digital input DI 3 as an external STOP

With this input function, you can hold an active traversing job using an external signal.

Digital input REF

You can wire a switch to this input for the reference cam.

You need a reference cam for the following:

- For a reference point approach
- For an incremental mode with hold on the reference cam.

Pulse enable (-4DC00-)

Description of the function

Pulse enable permits the output of pulses from the 1STEP 5V/204kHz to the power unit. A run is not possible without pulse enable.

Activating Pulse Enable

You can activate pulse enable by one of the following methods:

- Through the digital input DI 3 when "Function DI" is configured as an external enable pulse (Page 10736)

or

- Through the control bit DRV_EN when the "Function DI" is configured as an external STOP (Page 10736)

You can recognize the assigned pulse enable through the fact that

- The LED (Page 10747)RDY at the 1STEP 5V/204kHz lights up in the case of correct configuration.
- The STS_DRV_EN feedback bit is set.

Deleting the Pulse Enable

Deleting the pulse enable during a run terminates the run immediately because no more pulses are emitted to the power unit. The stepper motor runs without pulse control. This means the residual distance is no longer valid. The synchronization of the axis by means of the reference point is lost. The SYNC feedback bit and the RDY LED are deleted.

Deleting the pulse enable when the motor is at standstill deletes the SYNC feedback bit and the RDY LED.

In this case, it may be necessary to perform a reference point approach.

1STEP 5V configuration and parameter assignment

Parameters

Enabling group diagnostics

Group diagnostics

Group diagnostics are enabled in the respective check box.

A "Short circuit encoder supply" ERR_24V or a "Parameter assignment error" ERR_PARA lead to channel-based diagnostics if group diagnostics is enabled.

Selecting the base frequency

Base frequency F_b:

The base frequency F_b is the base value for setting the start-stop frequency, output frequency and acceleration/delay.

Select the base frequency suitable for your application.

Detailed information can be found under Setting the base frequency (Page 10717).

Specifying multiplier n

Multiplier n

With multiplier n, you can set the start-stop frequency in increments:

$F_{ss} = F_b \times n$

Permitted range of values: 1 … 255

See also

Setting the Base Frequency (Page 10717)

Specifying time interval i

Time interval i

With multiplier i, you can set the acceleration/delay in increments:

$a = F_b / (i \times 0.128 \text{ ms})$

Permitted range of values: 1 … 255

See also

Setting the Base Frequency (Page 10717)

Selecting the feedback value

Feedback value

Select the feedback value that is supplied in the feedback interface (Page 10744):

- Residual distance
- Position
- Frequency

Selecting the function DI0

Function DI0

Select which function the digital input DI0 (3) (Page 10733) should have:

- An external enable pulse
- An external STOP
- Limit switch in the forward direction
- Limit switch in the backward direction

Selecting the function DI1

Function DI1

Select which function the digital input DI1 (7) (Page 10733) should have:

- A reference switch (reference cam)
- Reference switch and limit switch in the forward direction
  This parameter selection is only possible if "Function DI0" is not configured as a "Limit switch forward".
- Reference switch and limit switch in the backward direction
  This parameter selection is only possible if "Function DI0" is not configured as a "Limit switch backward".

Selecting the behavior of inputs DI0 and DI1

Input DI0, input DI1

Select the behavior of digital inputs DI0 (3) and DI1 (7):

- Normally closed contact
- Normally open contact

See also

Behavior of the Digital Inputs (Page 10733)

Hold traversing job (Page 10728)

Selecting the behavior of the limit switches in the control interface

Limit switches in the control interface

Select the behavior of the limit switches in the control interface (Page 10741):

- Normally closed contact
- Normally open contact

See also

Hold traversing job (Page 10728)

Enabling the modulo axis

Modulo axis

The axis type "linear axis" is preconfigured at the 1STEP 5V. If you want to control a "modulo axis", you have to enable the "Modulo axis" check box.

See also

Axis type and traversing range (as of 6ES7138-4DC01-0AB0) (Page 10729)

Specifying the traversing range

Traversing range

The "Traversing range" parameter is used to specify the end of the modulo axis.

Permitted range of values: 1 … 16777216

See also

Axis type and traversing range (as of 6ES7138-4DC01-0AB0) (Page 10729)

1STEP 5V control and feedback interface

Control interface of 1STEP 5V

Note

For the 1STEP 5V, the following data of the control interface are consistent:

- Bytes 0 to 3
- Bytes 4 to 7

Assignment of the control interface

The following table shows the assignment of the control interface (outputs) of 1STEP 5V.

Address	Assignment	
Bytes 0 to 3	**Relative incremental mode, absolute incremental mode**	
	Byte 0	Multiplier G: $F_a = F_b \times R \times G$ (value range 1 to 255)
	Byte 1	Distance or position bit 23 to bit 16
	Byte 2	Distance or position bit 15 to bit 8
	Byte 3	Distance or position bit 7 to bit 0
	Reference point approach	
	Byte 0	Multiplier G: $F_a = F_b \times R \times G$ (value range 1 to 255)
	Byte 1	Position bit 23 to bit 16
	Byte 2	Position bit 15 to bit 8
	Byte 3	Position bit 7 to bit 0
	Set home position	
	Byte 0	Reserve = 0
	Byte 1	Position bit 23 to bit 16
	Byte 2	Position bit 15 to bit 8
	Byte 3	Position bit 7 to bit 0
	Speed-control mode	
	Bytes 0 to 3	Frequency as STEP 7 data type REAL
	Parameter Assignment Request	
	Byte 0	Reserve = 0
	Byte 1	Multiplier i: $a = F_b \times R / (i \times 0.128\ ms)$ (value range 1 to 255)
	Byte 2	Multiplier n: $F_{ss} = F_b \times n \times R$ (value range 1 to 255)

Address	Assignment		
	Byte 3	Base frequency F_b: • 0 = 800 Hz • 1 = 400 Hz • 2 = 200 Hz • 3 = 80 Hz • 4 = 40 Hz • 5 = 20 Hz • 6 = 8 Hz • 7 = 4 Hz • 8 = 2000 Hz	
Byte 4	Bit 7	Reduction factor • 0 = Factor 1.0 (no reduction) • 1 = Factor 0.1	R
	Bit 6	Hold traversing job	STOP
	Bit 5	Backward start	DIR_M
	Bit 4	Forward start	DIR_P
	Bit 3	Reserve = 0	
	Bit 2 to Bit 0	Operating mode • 0 = Relative incremental mode (relative positioning) • 1 = Reference point approach • 2 = Absolute incremental mode (absolute positioning) • 3 = Speed-control mode • 4 = Set home position	MODE
Byte 5	Bit 7	Diagnostics error acknowledgment	EXTF_ACK
	Bit 6	Change parameter	C_PAR
	Bit 5 to Bit 4	Feedback value in the feedback interface • 00 = Residual distance • 01 = Position • 10 = Frequency • 11 = Reserved	FEEDBACK
	Bit 3	Stop at the reference cam	STOP_REF_EN
	Bit 2	Pulse enable	DRV_EN
	Bit 1	Limit switch in the forward direction	LIMIT_P
	Bit 0	Limit switch in the backward direction	LIMIT_M
Byte 6	Reserve = 0		
Byte 7	Reserve = 0		

Notes on the Control Bits

Bits	Explanation
Base frequency $F_{b:}$	Coding for setting the base frequency in steps: • 0 = 800 Hz • 1 = 400 Hz • 2 = 200 Hz • 3 = 80 Hz • 4 = 40 Hz • 5 = 20 Hz • 6 = 8 Hz • 7 = 4 Hz • 8 = 2000 Hz
Operating mode	Coding for operating mode: • 0 = Relative incremental mode (relative positioning) • 1 = Reference point approach • 2 = Absolute incremental mode (absolute positioning) • 3 = Speed-control mode • 4 = Set home position
C_PAR	A parameter change is requested with this bit.
DIR_M	This bit requests and starts a traversing job in the backward direction.
DIR_P	This bit requests and starts a traversing job in the forward direction.
DRV_EN	If you use the DI0 (3) digital input as an external STOP, this bit is interpreted as a pulse enable.
Limit switch LIMIT_M	This limit switch limits the travel range in the backward direction. You set or delete this bit in your user program.
Limit switch LIMIT_P	This limit switch limits the travel range in the forward direction. You set or delete this bit in your user program.
EXTF_ACK	Acknowledgment bit for diagnostic message
FEEDBACK	Coding for the feedback value in the feedback interface: • 00 = Residual distance • 01 = Position • 10 = Frequency • 11 = Reserved
Frequency	A 32-bit value (STEP 7 data type REAL) that contains the pulse frequency to be output.
Multiplier G	Factor for setting the velocity / output frequency in steps
Multiplier i	Factor for setting the acceleration / deceleration in steps
Multiplier n	Factor for setting the start-stop frequency in steps
Position	24-bit value that contains the target position to be approached
Reduction factor R	The base frequency F_b is multiplied by 0.1 if the bit is set. This reduces the output frequency F_a, the start-stop frequency F_{ss}, and the acceleration / deceleration a by the same amount.
STOP	With this bit, you can stop a traversing job with a delay ramp at any time (see also "Stop traversing job (Page 10728)").

Bits	Explanation
STOP_REF_EN	When the bit is set, the "Stop at the reference cam" function is active. When the reference cam is recognized, the traversing job is stopped with a delay ramp (see also "Stop traversing job (Page 10728)").
Distance	A 24-bit value that contains (without signs) the number of pulses that have to be traversed.

Feedback interface of 1STEP 5V

Note

For the 1STEP 5V, the following data of the feedback interface are consistent:

- Bytes 0 to 3
- Bytes 4 to 7

Assignment of the feedback interface

The following table shows the assignment of the feedback interface (inputs) of 1STEP 5V.

Address	Assignment		
Bytes 0 to 3	Residual distance (bit 23 to bit 0 of 32 bits) / Position (bit 23 to bit 0 of 32 bits) / Frequency (32 bits, STEP 7 data type REAL)		
Byte 4	Bit 7	Short-circuit at encoder supply	ERR_24V
	Bit 6	Reserve = 0	
	Bit 5	Programming error	ERR_PARA
	Bit 4	Determining the home position	SYNC
	Bit 3	Reserve = 0	
	Bit 2	Position reached	POS_RCD
	Bit 1	Error during job transfer	ERR_JOB
	Bit 0	Job transfer running	STS_JOB
Byte 5	Bit 7	Traversing job running	POS
	Bit 6	Limit switch forward is cause for stop	STOP_LIMIT_P
	Bit 5	Limit switch backward is cause for stop	STOP_LIMIT_M
	Bit 4	External STOP is cause for stop	STOP_EXT
	Bit 3	Reference cam is cause for stop	STOP_REF
	Bit 2	Status DI0	STS_DI0
	Bit 1	Status DI1	STS_DI1
	Bit 0	Status pulse enable active	STS_DRV_EN
Byte 6	Error number at an error during job transfer		
Byte 7	Reserve = 0		

Notes on the Feedback Bits

Table 11-13 Feedback Bits

Bits	Explanation
Frequency	A 32-bit value (STEP 7 data type REAL) that contains the current pulse frequency. The pulse frequency can take on positive (forward direction) and negative (backward direction) values.
ERR_JOB	This bit is set if the job is not clear or not possible. The error cause is specified in more detail by the returned error number (see the following table "Error number in the feedback interface").
ERR_PARA	Incorrect parameter assignment for the ET 200S station. The error cause is specified in more detail by the returned error number (see the following table "Error number in the feedback interface"). The parameter error bit is deleted when a correct parameter assignment is transmitted.
ERR_24V	The encoder supply has been overloaded (e.g. by a short circuit) and is now switched off. ERR_24V is reset if it has been acknowledged with the EXTF_ACK control bit. After the overload has been eliminated, the encoder supply is switched on again and ERR_24V remains cleared.
Error number	Specifies the error cause if ERR_JOB or ERR_PARA is set (see table below "Error numbers in the feedback interface").
POS	Traversing: This bit is set as long as the traversing job is running.
POS_RCD	POS_RCD is cleared at the start of an incremental mode or at specification of a new setpoint frequency in speed-control mode. POS_RCD is set after a correctly executed incremental mode or when the setpoint frequency has been reached in speed-control mode. If traversing was interrupted (if the traversing job has stopped (Page 10731) or the pulse enable is deleted (Page 10728)), POS_RCD remains cleared.
Position	A 24-bit value that contains the current absolute position (without signs). Byte 0 of the feedback interface is 0.
Residual distance	A 24-bit value that contains the number of pulses that still have to be traversed (without signs). Byte 0 of the feedback interface is 0.
STOP_EXT	Cause for stop: This bit is set if the traversing job has been stopped by an external STOP.
STOP_LIMIT_M	Cause for stop: This bit is set if the traversing job has been stopped by reaching of the limit switch backward.
STOP_LIMIT_P	Cause for stop: This bit is set if the traversing job has been stopped by reaching of the limit switch forward.
STOP_REF	Cause for stop: This bit is set if the traversing job has been stopped by reaching of the reference cam.
STS_DI0	The bit displays the status of the DI0 (3) digital input.
STS_DI1	The bit displays the status of the DI1 (7) digital input.
STS_DRV_EN	This bit is set when one of the following occurs, depending on the assigned parameter function of the digital input DI0: • The external enable pulse is set. or • The DRV_EN control bit is set for the pulse enable.
STS_JOB	This bit is set as feedback when a job request for a traversing or parameter assignment job is detected and then reset when the job has been executed.
SYNC	This bit is set after a correct reference point approach or after manual specification of the home position has been set. The SYNC bit is cleared after parameter assignment with new ET 200S station parameters or after deletion of the pulse enable.

Error number

If an error with the job transfer (ERR_JOB is set) or an error in the basic parameter assignment (ERR_PARA is set) is displayed in the feedback interface, the error cause is specified in more detail by means of an error number.

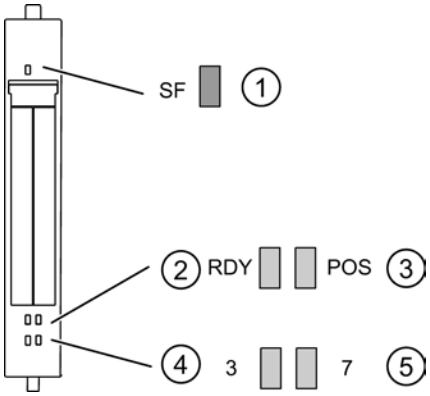
The table below shows the error numbers of the feedback interface.

Error number	Meaning
General error causes	
0	No error (then ERR_JOB or ERR_PARA is also not set)
1	Combination of the control bits (DIR_P, DIR_M, C_PAR) is invalid
2	Another job is still running.
Causes of errors with a traversing job	
16	Start forward (DIR_P) at limit switch forward (LIMIT_P) active
17	Start backward (DIR_M) at limit switch backward (LIMIT_M) active
18	Start with set control bit STOP
19	Start at external STOP active
20	Start at a missing pulse enable (internal or external)
21	Start with set STOP_REF_EN with active reference cam
22	Start without reference (at absolute incremental mode)
23	Start with diagnostic error present
24	Start was interrupted by CPU/Master STOP
25	Start with incorrect operating mode (not identical with requirement)
26	Distance or position specification is invalid
27	Multiplier G for the velocity is zero
28	Frequency is invalid at speed-control mode
Error causes at a parameter assignment job or for the basic parameter assignment	
32	Specification for the base frequency is invalid
33	Multiplier n for start-stop frequency is zero
34	Multiplier i for acceleration / delay is zero
35	Feedback value for the feedback interface is invalid
36	Combination of the functions of DI0 and DI1 is invalid (limit switches)
37	Specification for the end of the traversing range is invalid

1STEP 5V diagnostics

Diagnostics using the LED display

LED display on the 1STEP 5V



① Group error (red)
② Ready for traversing job (green)
③ Positioning underway (green)
④ Status display for Digital input 0 (green)
⑤ Status display for Digital input 1 (green)

Status and error displays by means of LEDs on the 1STEP 5V

The table below shows the status and error displays on the 1STEP 5V.

Event (LED)					Cause	Remedy
SF	RDY	POS	3	7		
On					No configuration or incorrect module plugged in. There is a diagnostic message.	Check the parameter assignment. Evaluate the diagnostics data.
	On				If parameters were assigned to the module correctly and pulse enable has been activated	
		On			When traversing job is running	
			On		DI 0 is activated.	
				On	DI 1 is activated.	

Error types

For information on the structure of the channel-related diagnostics, refer to the manual on the interface module used in your ET 200S station.

1STEP 5V error types

The following table shows the error types on the 1STEP 5V.

Error type		Meaning	Remedy
1_D	00001: Short circuit	Short circuit of the sensor supply.	Check the connection to the switches. Correct the process wiring.
9_D	01001: Error	Internal module error occurred.	Replace the module.
16_D	10000: Parameter assignment error	Parameters have not been assigned to the module.	Correct the parameter assignment.

11.4.2.9 Using the IM 174

Overview

Interface module for analog drives and stepper motors

An IM 174 module (interface module) is an interface module suitable for running up to 4 drives with an analog setpoint interface and with one TTL or SSI encoder per axis on the isochronous PROFIBUS.

Communication between the controller and the IM 174 is performed exclusively using PROFIBUS, via an IM 174-specific message frame type which, in addition to digital input/output data, also contains a message frame type (standard message frame 3 and 81) for each drive specified according to a PROFIDrive profile. As part of cyclic DP communication, the actual drive values (encoder values) are transferred from the IM 174 module to the controller via the isochronous PROFIBUS, and the speed setpoints calculated by the controller are transferred to the IM 174 module.

The transferred speed setpoints are then output from the IM 174 module to the drives as analog values or pulses.

Field of application

IM 174 can be used for the following control jobs:

- controlling analog drives
- controlling stepper motors
- reading in encoder data

Configuring and programming IM174

Configuring IM174

Steps required - Overview

You configure the IM 174 in the Inspector window under "Properties". You must carry out the following steps:

1. Add a PROFIBUS module to the hardware configuration
2. Assign parameters for isochronous PROFIBUS
3. Specify drive and encoder parameters for the axes

Assign parameters for isochronous PROFIBUS

1. Select the PROFIBUS subnetwork in the "PROFIBUS address" section and assign a PROFIBUS address.

   Note

   PROFIBUS address at the DIP switch

   The PROFIBUS address is set via DISourcing outputs on the PROFIBUS module IM 174. A power on/off / is necessary on the IM 174 to activate a PROFIBUS address set on the DIP switch. The assigned PROFIBUS address must be identical to the PROFIBUS address set at the DIP switch.

2. Select the check box "Synchronize DP slave to equidistant DP cycle" in the "Isochronous mode" section and set time Ti/To.
3. Select the frame type (Page 10758) in the "Encoder and drives" section.
4. Specify the address ranges for the input and output parameters of the axes and assign a process display in the "Addresses" section.

Specify drive and encoder parameters for the axes

1. Select the drive type (Page 10749) for each axis and set the drive parameters (Page 10750) in the "Encoder and drives" section.
2. Select the encoder type (Page 10749) for each axis and if necessary set the required encoder parameters (Page 10754) in the "Encoder and drives" section.
3. You can set the parameters (Page 10756) for switch-off behavior in the "Parameters" section.

Encoders and Drives

You can use the "Encoders and Drives" property in the area navigation to configure the drives and encoders that are connected to the IM 174.

Drive types

You can select the following drive types:

- Servo (analog drive)
- Stepper (stepper drive)

Encoder types for analog drives

The following encoder types can be selected for analog drives

- Encoder type not available
- Encoder type TTL
- Encoder type SSI

Encoder types for stepper drives

The following encoder types can be selected for stepper drives

- Encoder type not available
- Encoder type TTL
- Encoder type SSI
- Encoder type stepper

Drive parameters

Drive types

You can select the following drive types:

- Servo (analog drive)
- Stepper (stepper drive)

Servo drive type

If you have selected the servo drive type, you can switch the voltage range of the analog output voltage via the "Unipolar" check box.

Unipolar not selected

If the "Unipolar" option is **not** selected, an analog voltage in the range of **-10 V** to **+10 V** will be output as the setpoint. The drive can then be traversed in two directions.

Unipolar selected

If the "Unipolar" option **selected**, an analog voltage in the range of **0V** to **+10 V** will be output as the setpoint. The direction of rotation is then output from the IM 174, depending on the current speed setpoint via a digital output of the IM 174:

- Direction of rotation signal for Axis 1 -> Digital output X11, Pin 13
- Direction of rotation signal for Axis 2 -> Digital output X11, Pin 15
- Direction of rotation signal for Axis 3 -> Digital output X11, Pin 17
- Direction of rotation signal for Axis 4 -> Digital output X11, Pin 19

Alt.DrvRdy

Depending on the drive you use, the Drive Ready signal returns different status messages:

- The drive signals "Ready for switching on" with Signal Drive Ready. An enable from the controller is still required to switch the drive on.
- The drive signals "Ready" with the Signal Drive Ready signal. The drive is ready for control in this state and directly follows a setpoint.

You should **not**activate the "Alt.DrvRdy" function, if your drive reports a "ready to switch on" state upon a Drive-Ready signal (mostly in analogue drives).

Activate the "Alt.DrvRdy" function when the drive signals "Ready" with a Drive Ready signal (mostly with servo drives).

Information on which status message the drive you use returns can be found in the product information for your drive.

Stepper drive type

You must enter drive-specific or process-specific frequency values [Hz] in the fields "Max. frequency [Hz]" and "Stand. frequency [Hz]" for the "Stepper" drive type. The necessary power electronics (e.g. FM Stepdrive) and the stepper motor are required for the operation of a stepper motor on the IM 174.

The basic structure of an IM 174 with stepper drives and without encoders is shown in the following figure:

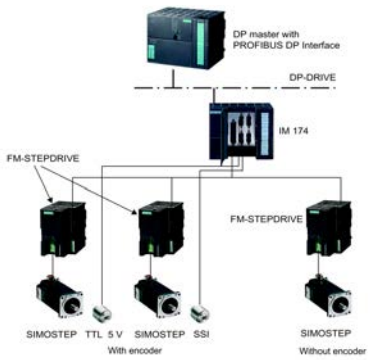


Figure 11-11 Basic structure of an IM 174 with FM STEPDRIVE and SIMOSTEP motor

Alt.DrvRdy

Depending on the drive you use, the Drive Ready signal returns different status messages:

- The drive signals "Ready" with the Drive Ready signal. An enable from the controller is still required to switch the drive on.
- The drive signals "Ready" with the Signal Drive Ready signal. The drive is ready for control in this state and directly follows a setpoint.

You should **not**activate the "Alt.DrvRdy" function, if your drive reports a "ready to switch on" state upon a Drive Ready signal (mostly in analogue drives).

Activate the "Alt.DrvRdy" function when the drive signals "Ready" with a Drive Ready signal (mostly with servo drives).

Information on which status message the drive you use returns can be found in the product information for your drive.

Max. frequency [Hz] and stand. frequency [Hz]

You must enter drive-specific or process-specific frequency values [Hz] in the fields "Max. frequency [Hz]" (maximum frequency) and "Stand. frequency [Hz]" (rated frequency) for the "Stepper" drive type.

Calculation of the stand. frequency

The stand. frequency is calculated according to the following formula:

Stand. frequency [Hz] = n [rpm] / 60 * resolution on the stepper motor

n [rpm]:	Speed of the stepper motor (characteristic values lie between 500 and 1000 rpm)
Resolution on the stepper motor:	Number of increments on the stepper motor (characteristic values: 500, 1000, 5000)

Calculation of the max. frequency

The max. frequency is calculated according to the following formula:

Maximum frequency [Hz] = n_{max} [rpm] / 60 * resolution on stepping motor

n_{max} [rpm]:	Maximum speed of the stepper motor (characteristic values lie between 500 and 1000 rpm)
Resolution on the stepper motor:	Number of increments on the stepper motor (characteristic values: 500, 1000, 5000)

The maximum speed n_{max} is determined by the operating characteristics of stepper motors. Hence, a maximum motor speed nmax, corresponding to a maximum axis speed, is determined from the technology required torque M_t.

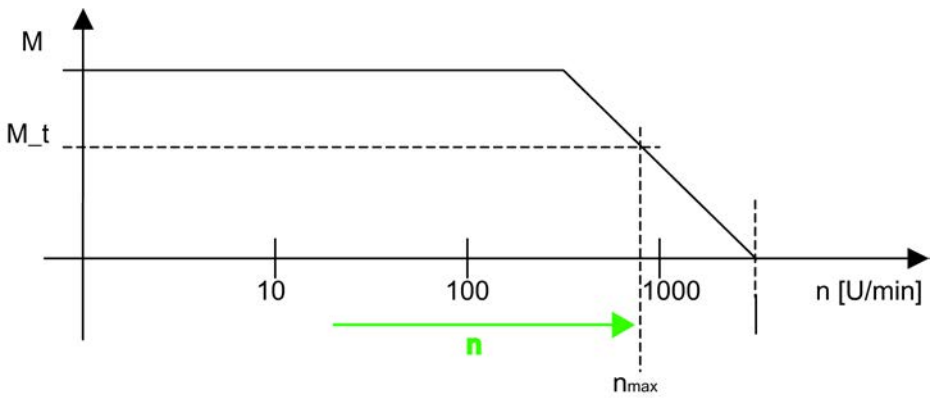


Figure 11-12 Characteristic operational features of a stepper motor

Note

Overload range

To avoid operating in the overload range, use the drive only at normal frequency!

Description of the behavior of a stepper motor

The positioning accuracy, the speed n as well as the torque M produced by the motor are very important for the process-specific requirements. For the optimum determination of these values, the behavior of the stepper motor must be considered. From a given speed of rotation of the stepper motor (around 500 rpm), the torque produced by the motor drops logarithmically, and approaches 0 at a maximum speed (around 3,000 rpm). The specific data can be found in the data sheet for the motor used.

Consequences of operation in the overload range

If the stepper motor is ever unable to produce the requested torque, it loses synchronization with the predefined frequency and its speed drops suddenly. This can lead to standstill. Motion can only be resumed in this state once a setpoint of 0 has been entered. For a position axis without additional encoders, the traversing position and thus synchronization of the axis are lost.

Note

Stepper drive with "Stepper" encoder type (PULSE REFEED operation)

During operation with a "Stepper" encoder type (PULSE REFEED operation), the stepper motor may vibrate at standstill.

This occurs with the default parameterization of a project when the stepper is traversed in a preceding traversing motion to a position value which is exactly between two natural step divisions of the stepper.

Stepper drive with TTL/SSI encoders

When using TTL/SSI encoders, the ratio of the encoder resolution (incl. fine resolution) to the step resolution of the stepper must be taken into account.

To prevent vibration between two position values at standstill, the stepper increment distance must be less than the distance representation in the encoder feedback (i.e. the increment of the stepper must be greater than the encoder resolution). If necessary, encoders with a lower resolution must be used or a higher step resolution set on the stepper.

Encoder parameters

Encoder types for analog drives

The following encoder types can be selected for analog drives:

- Encoder type not available
- Encoder type TTL
- Encoder type SSI

Encoder types for stepper drives

The following encoder types can be selected for stepper drives:

- Encoder type not available
- Encoder type TTL
- Encoder type SSI
- Encoder type stepper

Encoder type "not available"

The following applies to **"Servo" drive types**: Axis x is not available or should not be operated. Useful data transmitted for this axis in the PROFIBUS message frame are empty.

The following applies to**"Stepper" drive types**: The stepper is operated without an encoder.

If the stepper is configured for operation with encoder, it may only be used as a speed-controlled axis.

If no encoder (SSI/TTL) is used, the encoder type "Stepper" can be selected to enable positioning even if no encoder is connected.

Encoder type TTL

In the case of encoder type "TTL" you can set the following encoder parameters:

- **Resolution**: Encoder resolution in encoder pulses per encoder revolution
- **Speed calculation enabled**: The IM 174 calculates the speed if the check box is enabled.
- **Normalization speed**: This input box is only visible if the "Speed calculation enabled" option is enabled. Enter the rated speed of the motor in revolutions per minute. The speed with the least noise is obtained when the speed is calculated by the IM174.
- **Reserved bits for fine resolution**: Setting 0 - 15
  The "Reserved bits for fine resolution" parameter specifies the desired pulse multiplication of the encoder increments transmitted in actual encoder values G1_XIST1 and G1_XIST2. The setting corresponds to a pulse multiplication between 2^0 = 1to 2^{15} = 32,768.

Note

Minimum fine resolution for TTL encoders

TTL encoders always require considering and setting a minimum fine resolution of 2^2 (= 4).

Transmitted actual encoder value G1_XIST1 or G1_XIST2

31		0
Overflows	Current encoder increments	Reserved bits for fine resolution

Encoder type SSI

You can set the following parameters for encoder type "SSI":

- **Parity**: Select this option if the encoder data are to be transmitted from the encoder to the IM 174 with a parity bit.
- **MsgLength**: Setting 0 - 25
  Number of useful data bits transmitted by the encoder
- **Encoding**: The following encoder codes are supported:
  - Binary
  - Gray

- **Transmission rate**: The transmission rate setting must be identical for all SSI encoders. If transmission rate settings are different, the transmission rate of the SSI encoder with the highest encoder number is assigned.
  The following transmission rates are supported:
  - 187.5 Kbps
  - 375 Kbps
  - 750 Kbps
  - 1.5 Mbps
  - 3.0 Mbps

  Note

  Configure a maximum of 187.5 Kbps. It is not practical to set a quicker transmission rate for this module.

- **Reserved bits for fine resolution**: Setting 0 - 15
  The "Reserved bits for fine resolution" parameter specifies the desired pulse multiplication of the encoder increments transmitted in actual encoder values G1_XIST1 and G1_XIST2. The setting corresponds to a pulse multiplication between 2^0 = 1 to 2^{15} = 32,768.

Encoder type stepper

Stepper drive in PULSE REFEED mode

An Axis x with the "Stepper" drive type can be operated with a "Stepper" encoder type. In this mode, the setpoints are signaled back as process values from the IM 174 to the controller.

You can set the following encoder parameters for encoder type "Stepper":

- **Motor monitoring**: If the "Motor monitoring" check box is activated, the number of steps of a stepper motor must be constantly monitored for a preset reference point distance (see "BERO distance"). In the case of encoderless operation, the stepper drive used simultaneously signals the process values (pulse refeed).
- **Bero distance**: In the case of "BERO distance" the number of steps (for the stepper drive used) between two BERO signals is entered. The distance between the two BERO signals is the reference point distance.
- **Bero tolerance**: Setting 0 - 65535
  In the case of "BERO tolerance" the permissible deviation of the steps is entered. The resulting step range lies within the following tolerance:
  resulting step range = Bero distance ± 1 / 2 * Bero tolerance
- **Reserved bits for fine resolution**: Setting 0 - 15
  The "Reserved bits for fine resolution" parameter specifies the desired pulse multiplication of the encoder increments transmitted in actual encoder values G1_XIST1 and G1_XIST2. The setting corresponds to a pulse multiplication between 2^0 = 1to 2^{15} = 32,768.

Parameters

Important parameters for the IM 174 are specified under the "Parameters" property in the area navigation.

shutdown ramp [ms]

The value range is 0 - 65535 ms.

The "Shutdown ramp" parameter specifies a function that is linear with respect to time. If an error is detected, all drives that are connected to the IM 174 are decelerated down to setpoint 0 in accordance with this function.

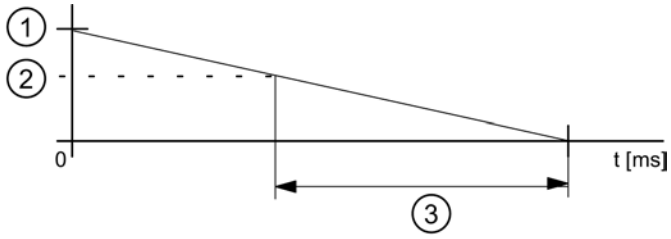
The shutdown ramp takes effect for the following errors:

- Temperature alarm of the module (approx. 90 °C ON, approx. 85 °C OFF)
- Lifebeat error
- Synchronization error between master and slave

Note

Parameter value 0

A parameter value of 0 causes the drive to coast to a standstill when an error occurs.



1) Maximum setpoint
2) Current setpoint
3) Parameter value: Shutdown ramp

Figure 11-13 Parameters: Shutdown ramp

Shutdown delay

The value range is 0 - 65535 s.

The "Shutdown delay" parameter can be used to specify a time after which, after a **temperature alarm** has occurred, the shutdown ramp is activated.

The shutdown delay is started when a temperature of 90 °C is exceeded in the module. If the temperature drops to approx. 85 °C while the delay is active, the shutdown ramp is not activated and the shutdown delay time is reset.

Tolerable lifebeat errors

The value range is 0 - 15.

The "Tolerable lifebeat error" parameter specifies the number of lifebeat errors tolerated for the DP master. The shutdown ramp is activated when the parameterized number is exceeded. The value range is

611U-compliant mode

In 611U-compliant mode, the signal source for homing of axes is no longer specified using the PROFIDrive standard message frame (STD3, encoder control word G1_STW), but rather using the additional digital output word in the PROFIBUS message frame of the IM 174 (see table "Message frame structure" in the "Message frame type" section).

611U-compliant mode:

- Not selected
  The signal source for homing is specified via the encoder control word Gx_STW in the PROFIDrive standard message frame.
- Selected
  The signal source for homing is specified via the additional digital output word in the PROFIBUS message frame.

Selecting the message frame type

Message frame type

The IM 174 DP slave is operated with two specific message frame types: Four axes, each with one encoder (standard message frame 3 and 81) and I/O data

Note

Standard message frame 81

Standard message frame 81 can only be used for encoders.

Message frame structure

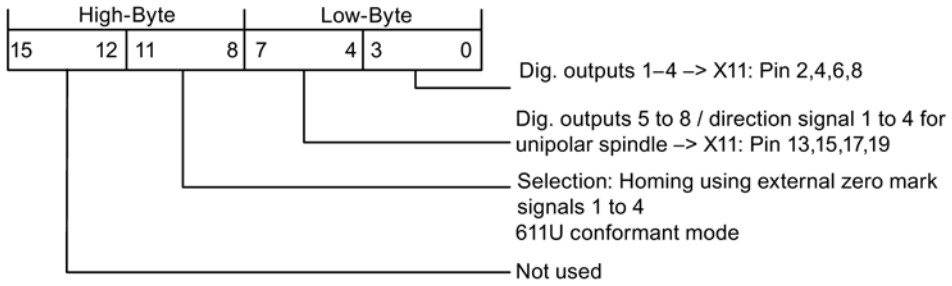
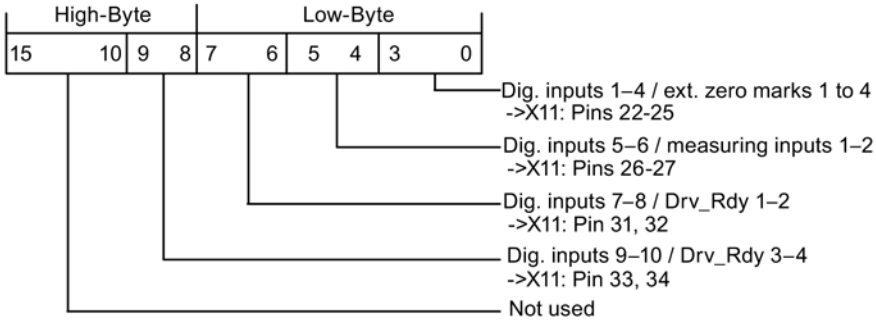
Message frame structure

The overview below explains the frame structure.

Table 11-14

Message frame type	Description
4 axes, each with one encoder, standard message frame 3 + IO, PZD-5/9 O/I 1/1 or message frame 81 + IO, PZD-2/6 + I/O 1/1	4 x Standard message frame 3 or 81 and 1 PZD word each for digital I/O data
PZD x/y Number of process data words, x: Setpoint, y: Actual value, for example, PZD-5 / 9: 5 process data words to desired values 9 process data words for actual values	

Message frame type	Description
IM174 message frame structure Axis1: STD 3 or STD 81; Axis2: STD 3; Axis3: STD 3; Axis4: STD 3; I/O: O word — Setpoints (master -> slave) Low — Axis1: STD 3 or STD 81; Axis2: STD 3; Axis3: STD 3; Axis4: STD 3; I/O: I word — High — Actual values (slave -> master) STD 3: standard message frame 3 per PROFIDrive V3.0 Q word: digital output data (16 bits) I word: digital input data (16 bits)	
Standard message frame 3: speed setpoint interface 32 bits with 1 encoder PZD1: STW1; PZD1–PZD3: NSOLL_B; PZD4: STW2; PZD5: G1_STW — Setpoint (master -> slave) Low — High PZD1: ZSW1; PZD2–PZD3: NIST_B; PZD4: ZSW2; PZD5: G1_ZSW; PZD6–PZD7: G1_XIST1 PZD8–PZD9: G1_XIST2 — Actual value (slave -> master)	
Standard message frame 81: speed setpoint interface 32 bits with 1 encoder PZD1: STW2; PZD2: G1_STW1 — Setpoint (master -> slave) PZD1: ZSW2; PZD2: G1_ZSTW1; PZD3–PZD4: G1_XIST1; PZD5–PZD6: G1_XIST2 — Actual value (slave -> master)	

Message frame type	Description
O word (dig. output data 16 bits)	
	
I word (dig. input data 16 bits)	
	

Note

The message frame type setting for the IM 174 DP slave must agree with the message frame type setting in the controller. There is no automatic adjustment.

Encoder control word Gx_STW

The overview below explains the encoder control word (extract). The overview is based on the following requirements:

- Find reference mark
- On-the-fly measurement
- Encoder error

Bit	Identifier		Signal state/description		
0	Find reference mark or On-the-fly measurement	Functions	Bit 7 = 0 => Request: Find reference mark		
			Bit	Meaning	Homing using:
			0	Function 1:	Encoder zero mark (except in "611 U conform mode")
			1	Function 2:	Not used
			2	Function 3:	Not used
			3	Function 4:	Not used
1			Bit 7 = 1 => Request: On-the-fly measurement		
			Bit	Meaning	
			0	Function 1:	Measuring input 1 rising edge
			1	Function 2:	Measuring input 1 falling edge
			2	Function 3:	Measuring input 2 rising edge
			3	Function 4:	Measuring input 2 falling edge
2			Note • Bit x = 1 Function requested Bit x = 0 Function not requested • If more than one function is enabled, the values for all functions cannot be read until all functions have ended and this has been signaled via the relevant status bit (G1_ZSW, Bit 0 - Bit 3 = 0). • On-the-fly measurement The rising and falling edges of the measuring input can be enabled simultaneously. The measuring input signal is detected according to the direction of the signal change. The measured values are read out consecutively.		
3					
4		Command	Bits 6, 5, 4	Meaning	
			000	--	
5			001	Enable function x	
6			010	Read value x	
			011	Cancel function x	
7		Mode	0	Find reference mark	
			1	On-the-fly measurement	
15	Acknowledge encoder error		0 1	-- Acknowledge encoder error	

Note

Measurement on rising or falling edge

IM 174 only supports measurement on a rising or falling edge.

Additional encoder actual value Gx_XIST2

The overview below sets out the error codes in Gx_XIST2 where G1_ZSW, Bit 15 == 1:

Table 11-15 Error codes in Gx_XIST2

G1_XIST2	Meaning	Possible causes / description
1_{Hex}	Encoder sum error	The encoder signal levels are too low, faulty (inadequate shielding) or open-circuit monitoring has been tripped.
2_{Hex}	Zero mark monitoring	A fluctuation in the measured rotor position has arisen between 2 encoder zero marks (encoder pulses may be lost).

Status word ZSW_1 - Bit 11 to Bit 14

The following overview describes the status word ZSW_1 with regard to the module-specific messages Bit 11 to Bit 14:

Table 11-16 Status word ZSW_1

Bit	Name	Description
11	Temperature error	A specific temperature in the housing of module IM 174 has been exceeded.
12	PLL synchronization error	The IM 174 cannot be synchronized with the global check-back signal.
13	Master life sign error	The IM 174 cannot be synchronized with the master life sign signal.
14	Drive error	A drive error exists, for example the Drive Ready signal is missing.

Zero mark monitoring

Method of functioning of the zero reference mark monitoring

Zero mark monitoring compares the recognized encoder increments between two zero marks with the configured encoder resolution. If a difference is found, an encoder error is reported.

Track A and Track B as well as Track Z of the encoder have to be on the high level simultaneously so that zero mark monitoring is carried out correctly.

Restrictions:

- The failure of a zero mark in itself does not inevitably result in an error, since checking is not started until a zero mark is reached.
- The result of the counted encoder increments MOD 16 has to be nonzero so that an error is reported.
- The result of the counted encoder increments MOD 10 has to be nonzero so that an error is reported.

Example:

Encoder configuration: TTL 2048 pulses

Counted encoder increments 2049 -> encoder error

Counted encoder increments 1024 -> no encoder error

Programming the IM 174

We recommend the "Easy Motion Control" add-on package for programming the IM 174.

Instructions for IM 174

The following instructions are available for IM 174 in the Easy Motion Control instruction list:

- EncoderIM174 - Use of IM 174 as position feedback module for incremental and absolute encoders
- OutputIM174 - Use of a channel of module IM 174 as output module

11.4.2.10 Using Easy Motion Control

Positioning with Easy Motion Control

Product overview

Overview

Easy Motion Control allows you to control the positioning of drives. The movement is controlled solely by software (instructions) in the CPU. **No special hardware** is required for connection to the drives, although corresponding I/O must of course be available in order to record actual positions and output speed setpoint values.

The instructions of Easy Motion Control were created based on the "Technical Specification Function blocks for motion control V1.0" of **PLCopen**.

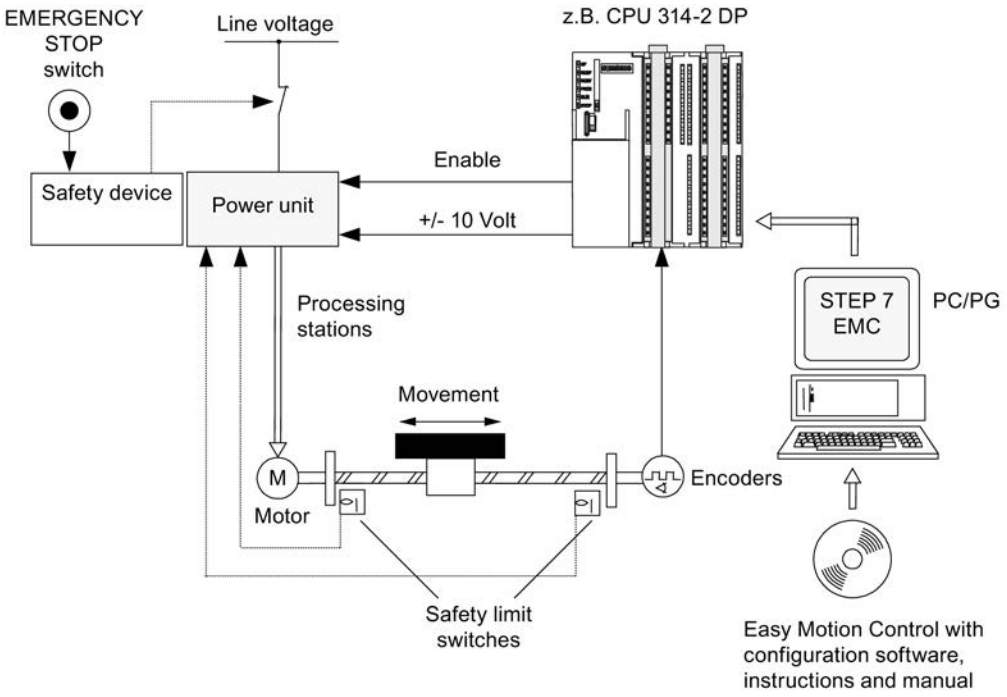
You assign parameters using configuration software that also supports commissioning using test functions. All parameters of an axis are stored in a single data block, namely the axis DB.

You can connect the drives to central I/O, or to distributed I/O via a DP master / IO controller that is integrated in the CPU.

You execute the movement by interconnecting the corresponding instructions and running them in the user program.

Components for Positioning

The following figure shows a configuration model with Easy Motion Control.



Safety device

When the safety device is actuated, the emergency STOP button disconnects the power supply.

Power unit

The power unit controls the motor. The safety limit switches disconnect the power unit.

Motor

The motor is controlled by the power unit and drives the axis.

Encoder

The encoder provides information on distance and direction.

Easy Motion Control

Easy Motion Control provides the following functions:

- Jogging
- Reference point approach
- Absolute/relative positioning
- Electronic gear
- Set reference point
- All common monitoring functions for positioning
- Velocity override
- Closed-loop position control
- Simulation
- Preconfigured input drivers for recording encoder values, or output drivers for analog output modules

Modules for position detection and setpoint output

These modules form the interface between the CPU and the machine. They read axis positions and output setpoint values to the power unit.

CPU

The CPU executes the user program that contains the embedded instructions of Easy Motion Control.

PC/PG

A PC/PG is used for the following functions:

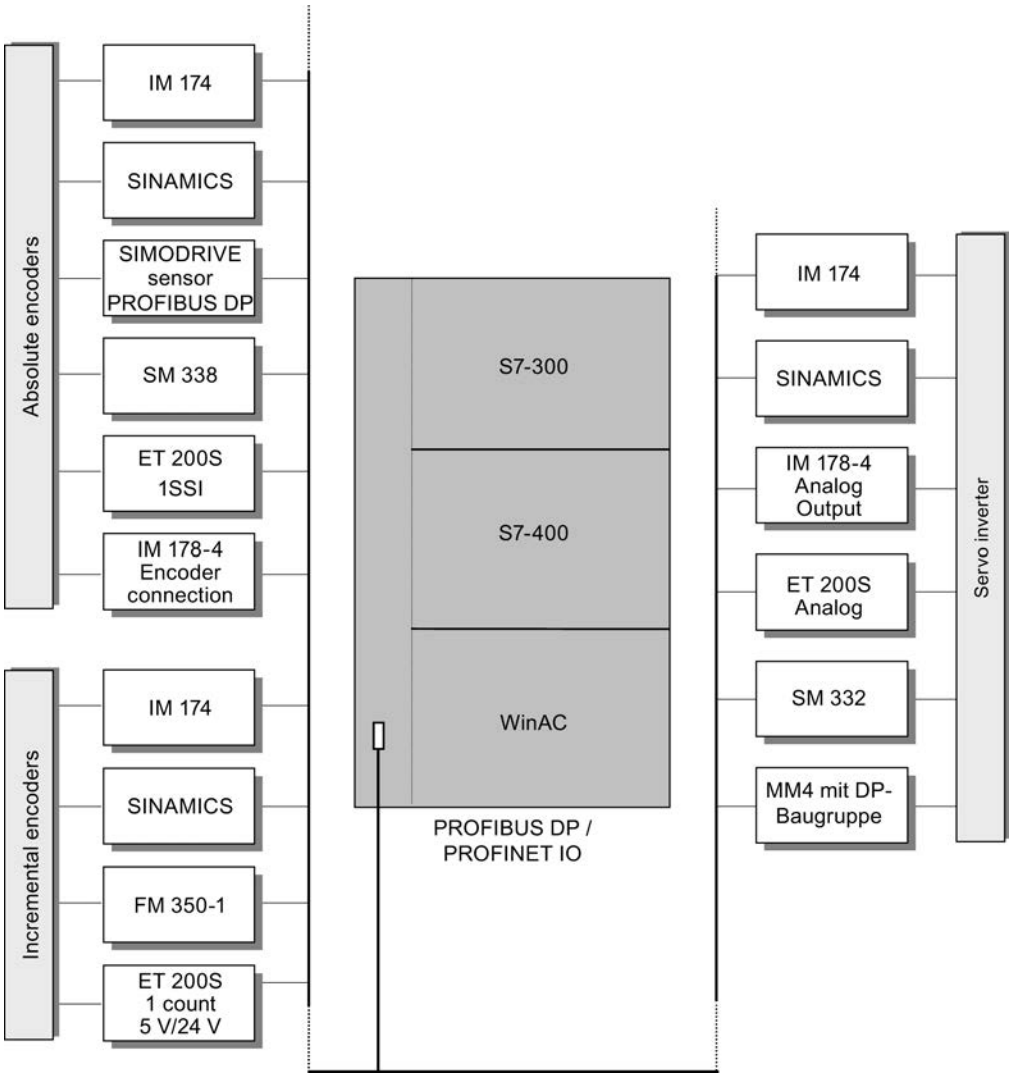
- Parameter assignment: You assign parameters to Easy Motion Control using the configuration software, or directly in the axis DB (see Configuring the axis TO (Page 10775)).
- Programming: You integrate the Easy Motion Control instructions directly in your program.

Supported hardware configurations

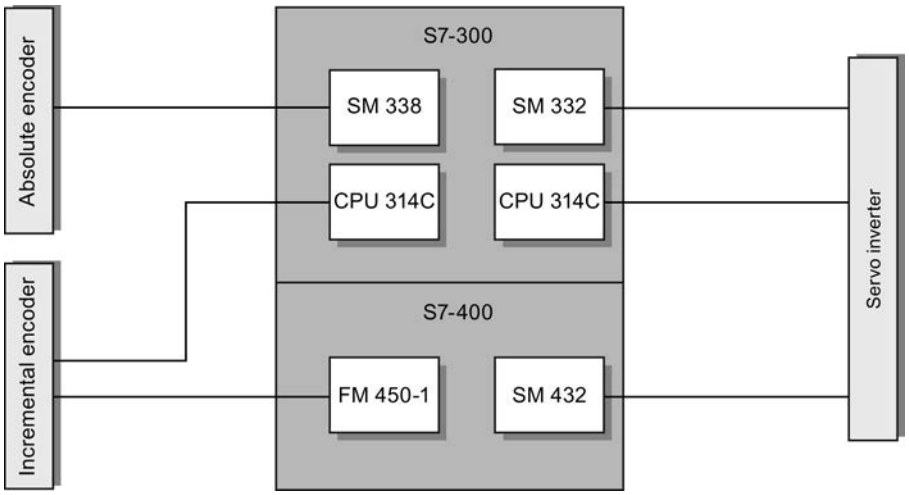
The following hardware configurations are conveniently supported by driver instructions. These are supplied with Easy Motion Control.

You can adapt configurations that are not directly supported using the open driver blocks that are also provided in your package.

The following figure shows **distributed** configurations for Easy Motion Control.



The following figure shows **centralized** configurations for Easy Motion Control.

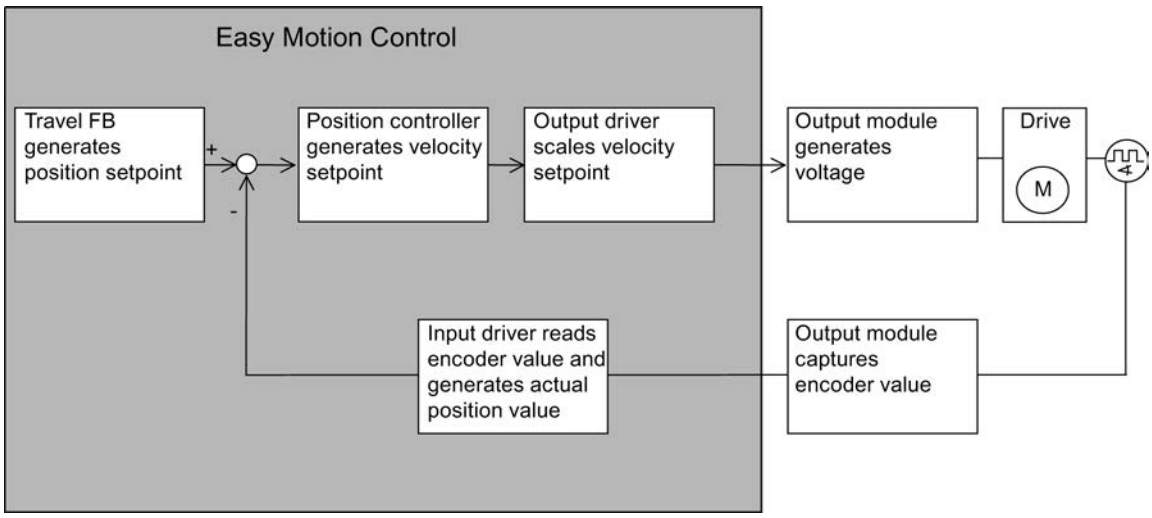


Principles of Positioning with Easy Motion Control

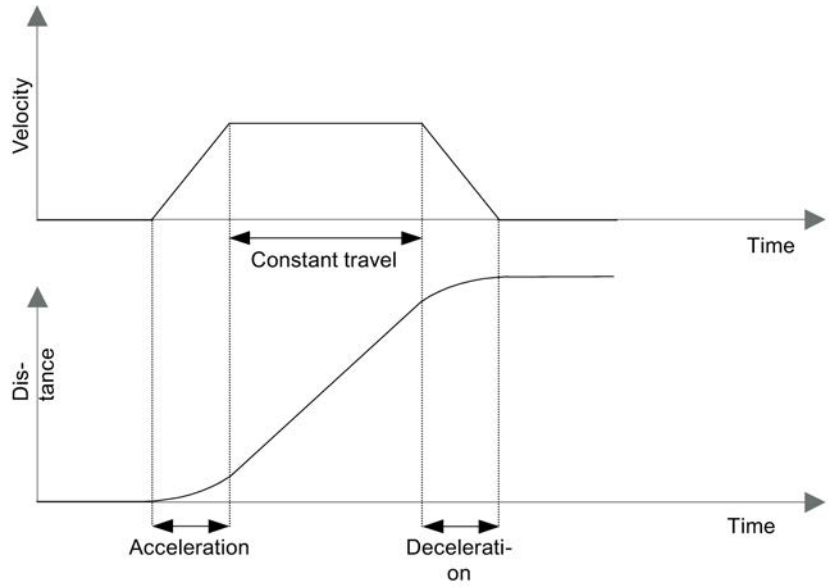
Setting up closed loop position control

With controlled positioning, the current position of the axis is compared with the setpoint position. Any deviations are compensated using the position controller both during a travel movement as well as at a standstill.

The following figure shows the schematic representation of controlled positioning with Easy Motion Control. A travel instruction (Page 10799) specifies the position setpoints. The input driver (Page 7058) records the actual value of the axis position. The position controller (Page 10786) compares the actual position value with the position setpoint and generates a velocity setpoint value from the difference. The output driver (Page 7081) outputs the scaled velocity setpoint.



The following figure shows the relationship between the axis velocity and the distance covered during a traversing movement.



Concept Definitions

Target

The target is the absolute or relative position of the axis that is approached during positioning.

Traversing movement

Traversing movements are triggered by calling instructions, which are also referred to as travel instructions in the following.

Residual distance

Residual distance is the difference between target position and current position setpoint.

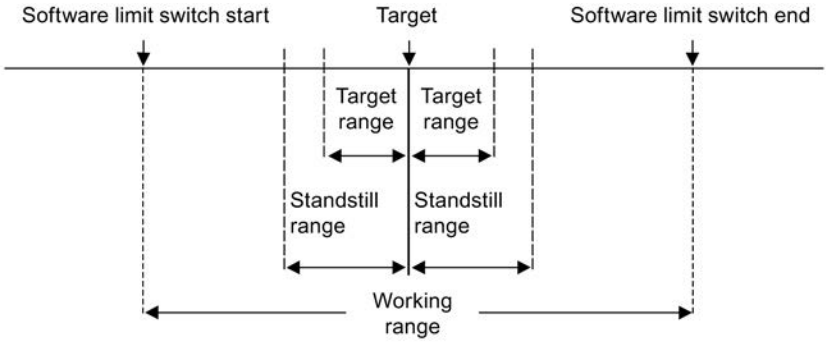
Following distance / following error

During a travel movement, a difference builds up between the current position setpoint and the actual position value due to axial inertia. This is called the following error. If the following distance exceeds an assignable value, a following error occurs.

Monitoring ranges

- Work range:
  Defines the permitted traversing range of your axis, which you determine for your task using the software limit switches (Page 10769), or the end of the rotary axis (Page 10770).
- Standstill range:
  Defines a configurable range that the axis must not leave when at a standstill. The standstill range lies to the left and right of the target.
- Target range:
  If the setpoint has reached the target at the end of a motion, the approach of the axis into the target range is monitored. The target range is used to monitor the positioning accuracy of your application and lies to the left and right of the target.
  Standstill range monitoring begins as soon as the axis reaches the target range. For this reason, the standstill range should be greater than the target range.

The following figure shows the position of the ranges in the target area.



Linear axis

Description

The **working range** extends from the start of the software limit switch to the end of the software limit switch and is limited to 1 × 2^{24} length units (Cartesian coordinates between -2^{24} and +2^{24}).

The **physical axis length** is limited by the mechanical stops of your axis.

The **numerical range** of Easy Motion Control is only relevant if work range monitoring is switched off. It is limited to 2 × 2^{24} length units (from -2^{24}+6 to +2^{24}-6).

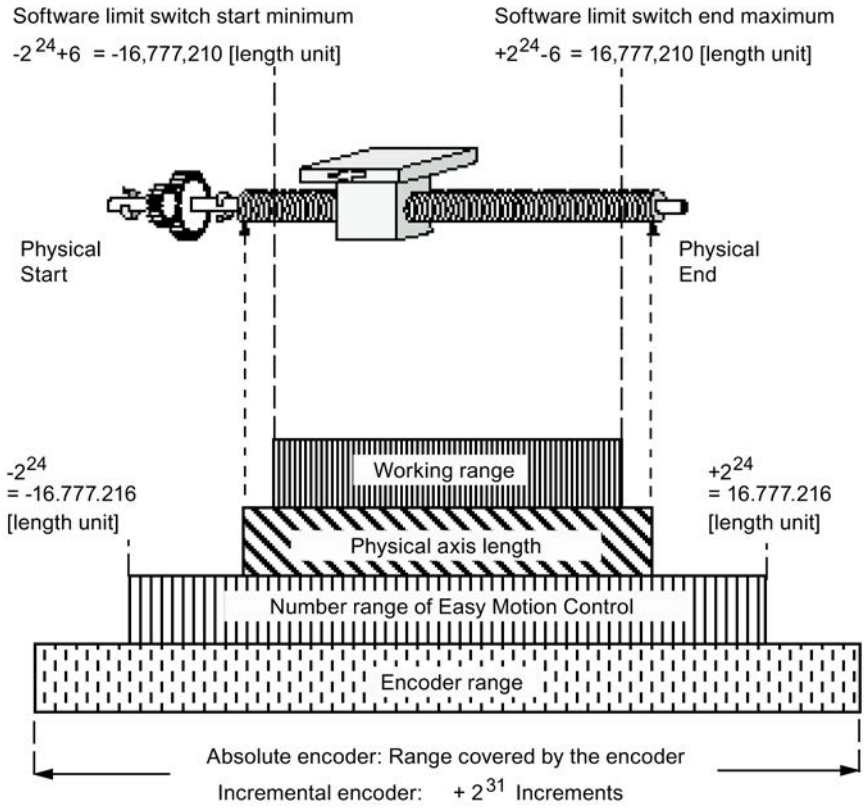
The **encoder range** is:

- The range covered by an absolute encoder
- The distance from -2^{31} to +2^{31} steps covered by an incremental encoder

With linear axes, the **work range** (= travel range) is limited as follows:

Working range < physical axis length < numerical range of Easy Motion Control < encoder range

The following figure shows the various ranges of a linear axis.



Rotary axis

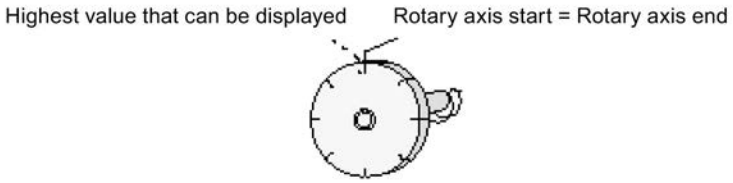
Description

The **working range** extends from the start of the rotary axis to the end of the rotary axis and is limited to 1 × 2^{24} length units (Cartesian coordinates between -2^{24} and $+2^{24}$).

The **encoder range** is:

- Arbitrary for an incremental encoder
- Determined for an absolute encoder based on the rule:
  Encoder range = working range × n
  where:
  - n = 1 with single-turn encoders
  - n > 1 and integer with multi-turn encoders

The actual value of a rotary axis is reset to the start of the rotary axis after every revolution. Rotary axes therefore have an infinite traversing range that extends beyond the start or end of the rotary axis. The axis position is always displayed as a value between the start and end of the rotary axis.



Functions with Easy Motion Control

Overview

Easy Motion Control is used to position an axis on a target with programmable acceleration, velocity, and deceleration or move it at constant velocity. The necessary traversing movements are triggered by calling travel instructions.

At the start of a traversing job, the target or distance of the traversing movement and possibly the required acceleration, velocity, or deceleration, are configured at the travel instruction. The system of units (length unit) can be selected freely for these values (for example, mm).

Jogging

The jogging function allows you to move the drive in the positive or negative direction. No target is specified.

The jogging function is realized using the MC_MoveJog (Page 7040) instruction.

Synchronization

In order to assign a reproducible encoder value to the real position, a reference (synchronization) must be established between the axis position and the encoder value.

The synchronization function is realized using the MC_Home (Page 7043) instruction.

Synchronization with an Incremental Encoder

An incremental encoder does not give any information about the immediate position of the axis when the system is switched on. Instead, it starts counting at the current position of the axis.

Synchronization is always necessary when the system is switched on again after occurrence of errors or after replacing the encoder.

An incremental encoder can be synchronized by:

- Reference point approach
- Set reference point

Synchronization with absolute encoder

When commissioning absolute encoders, you must synchronize the value indicated by the encoder with the actual axis position.

Resynchronization is not necessary when the system is switched back on or after errors have occurred. After the encoder has been replaced, however, resynchronization is necessary.

An absolute encoder can only be synchronized by means of reference point setting (also called absolute encoder adjustment).

Reference point approach

The axis is moved in the direction of the reference point. The axis is synchronized after it has passed the reference point, i.e. a position value is assigned to the numerical value of the incremental encoder. A new coordinate value can be assigned to the reference point with every reference point approach.

A reference point approach is only possible with incremental encoders.

Set reference point

With reference setting, a new coordinate value is assigned to the current position.

Reference point setting is supported for incremental and absolute encoders.

Absolute or Relative Positioning

Easy Motion Control can move the axis

- To **absolute** targets
- By a given distance in a specific direction with **relative** positioning

The absolute and relative positioning functions are realized using MC_MoveAbsolute (Page 7035) and MC_MoveRelative (Page 7038) instructions.

Gear

Easy Motion Control realizes an electronic gear between two axes using the MC_GearIn (Page 7051) instruction.

Stopping Travel

You can stop current travel at any time.

The Stop function is realized using the MC_StopMotion (Page 7047) instruction.

Overriding Travel

A currently active travel instruction can always

- Be overridden by starting a different travel instruction
- Be canceled by calling instruction MC_StopMotion (see Sequence of traversing movements (Page 10805)).

Tracking

With tracking, the position setpoint is continuously adjusted (tracked) in the position controller to the actual position value. During tracking, the axis is not monitored for leaving the standstill range.

This function is activated by clearing the "EnableDrive" signal at instruction MC_Control (Page 7049), or after an error has occurred.

Velocity override

Velocity override enables dynamic changes to the velocity that is configured in a travel instruction. Acceleration and deceleration values are not affected.

For more information on velocity override, refer to Specifying velocity override (Page 10791).

Simulation

In simulation mode, you can test your travel program without moving your axis.

Simulation is realized using the MC_Simulation (Page 7050) instruction.

Error Responses and Diagnosis

All errors detected by the instructions are displayed in the axis DB by

- Setting a group error and
- Error-specific display

With errors requiring acknowledgment, an error acknowledgment must be set after rectifying the error.

If a travel instruction (for example, MC_MoveJog, or MC_MoveAbsolute) has caused or detected the error, the "Error" output is set at the corresponding block.

However, if the error was detected by a driver block, the exact cause of the error is indicated in the static data of the block (see Input drivers (Page 7058)).

For detailed information on error response, error display, and error acknowledgment, see Error Displays and Error Acknowledgment in the axis DB (Page 10820).

Configuring Easy Motion Control

Guideline for using Easy Motion Control

This guideline shows you the basic procedures for using Easy Motion Control in combination with an S7-300/400 CPU.

Requirement

To use the "Axis" technology object (TO), you need to create a project with a CPU.

Procedure

Proceed based on the sequence recommended below to use Easy Motion Control with the S7-300/400 CPU. Use the following links for this purpose:

1. Adding an axis technology object (Page 10774)
2. Working with the configuration dialog (Page 10775)
3. Download to CPU (Page 10797)
4. Function test of the axis in the commissioning window
5. Programming Easy Motion Control (Page 10799)
6. Diagnostics of the axis control

Adding an axis technology object

Inserting an axis TO in the project navigator

When you add a technology object, an instance DB of this TO will be generated. The axis configuration is stored in this instance DB.

Requirement

A project with a CPU has been created.

Procedure

Proceed as follows to insert an axis TO in the project navigator:

1. Open the CPU folder in the project tree.
2. Open the "Technology objects" folder.
3. Double-click "Add new object".
   The "Add new object" dialog opens.
4. Select the "Motion" technology.
5. Open the "Motion Control" folder.
6. Open the "Easy Motion Control" folder.
7. Select the instruction for the technology object: in this case, AXIS_REF.
8. Type in an individual name for the axis in the "Name" text box.
9. Select the "Manual" option if you want to change the proposed data block number of the instance DB.
10. Click "More information" if you want to supplement user information on the TO.
11. Click "OK" to add the TO.
    Click "Cancel" if you want to discard your entries.

Result

The new technology object has been created and stored in the project tree in the "Technology objects" folder. The technology object is used if the instructions for this TO are called in a cyclic interrupt or synchronous cycle interrupt OB.

For more information, refer to "User Program Setup (Page 10799)".

Note

You can select the "Add new and open" check box at the bottom of the dialog box. This opens the configuration of the technology object after adding has been completed.

Configuring the axis TO

Working with the configuration dialog

You configure the properties of the technology object in the configuration window. To open the configuration window of the technology object, follow these steps:

1. Open the group of the selected technology object in the project tree.
2. Double-click the "Configuration" object.

The configuration is divided into the following categories:

- **Basic parameters**
  The basic parameters contain all the parameters which must be configured for a functioning axis.
- **Advanced parameters**
  The advanced parameters include parameters for adapting to your drive or plant.

Icons of the configuration window

Icons in the area navigation of the configuration show additional details about the status of the configuration:

	The configuration contains default values and is complete. The configuration contains only default values. With these default values you can use the technology object without additional changes.
	The configuration contains values set by the user and is complete. All input fields of the configuration contain valid values and at least one preset value has changed.
	The configuration is incomplete or incorrect At least one input field or drop-down list contains an invalid value. The corresponding field or the drop-down list is displayed on a red background. Click the roll-out error message to indicate the cause of error.
	The configuration is valid but contains warnings Only one hardware limit switch is configured. Depending on the plant, the lacking configuration of a hardware limit switch may result in a hazard. The corresponding field or the drop-down list is displayed on a yellow background.

Basic parameters

Naming the axis

Axis name

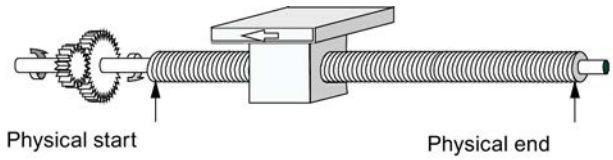
Define the name of the axis, or the name of the "Axis" technology object, in this field. The technology object is listed under this name in the project navigator.

Selecting axis type

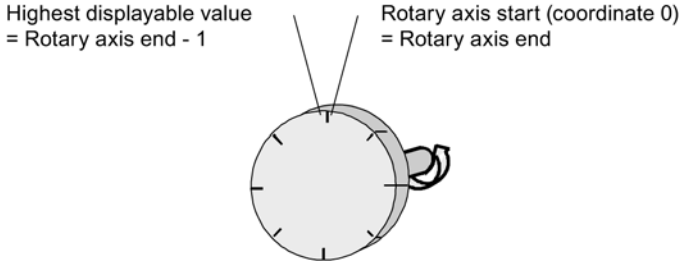
Axis types

Easy Motion Control is able to control the following axis types:

- Linear axis
  A linear axis is an axis which has a limited physical traversing range.



- Rotary axis
  The rotary axis is an axis whose traversing range is not restricted by mechanical stops.



Data used in the axis DB

Address	Name	Type	Initial value	Comment	
4.0	AxisType	BOOL	FALSE	**Axis type** Valid values: • FALSE = Linear axis • TRUE = Rotary axis	I

Selecting length units

Length unit

Select the length unit for the axis unit system from the drop-down list. The selected unit will be used for further configuration of the "Axis" TO and for visualization of the current axis data.

You can select a length unit for data input and output from the following:

- Millimeter (default setting)
- Degree
- Inch

The values at the input parameters (Position, Distance, Velocity, ...) of the Motion Control instructions also refer to this unit.

Note

A change of the length unit will not affect the values of the parameters. There, is no conversion.

Data used in the axis DB

Address	Name	Type	Initial value	Comment
196.0	LengthUnit	STRING [4]	mm	**Length unit** for configuration software Valid values: • mm (default) • Degree • Inch

Activating simulation mode

Simulation mode

Select the check box if you want to test your traversing program without traversing the axis.

Data used in the axis DB

Address	Name	Type	Initial value	Comment
4.2	Sim	BOOL	FALSE	**Simulation mode** Valid values: • FALSE = Deactivated • TRUE = Activated

Advanced parameters

Driver interfaces

Input driver

Selecting an input module

Input module

Select the input module to which your encoder is connected from the drop-down list.

Based on your selection, the "Instruction" field displays the name of the instruction that you have to call for actual value acquisition.

Data used in the axis DB

Address	Name	Type	Initial value	Comment
202.0	InputModuleType	STRING [24]	IM 174	Input driver **for the module**. It is not allowed to overwrite this parameter in manual mode.

Selecting the encoder type

Encoder type

Select the check box for the encoder used.

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
4.1	EncoderType	BOOL	FALSE	**Encoder type** Valid values: • FALSE = Incremental encoder • TRUE = Absolute encoder	I

Input module: Module address inputs

Module address inputs

Enter the value specified for the input module in your hardware configuration in the text box.

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
58.0	InputModuleInAddr	INT	0	**Start address of the inputs of the position decoder module** Range: • Determined by the hardware configuration	I

Input module: Module address outputs

Module address outputs

The parameter is used only for modules for which it is possible to set the "Module address outputs" separately from the "Module address inputs" in the hardware configuration.

Enter the value specified for the input module in your hardware configuration in the text box.

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
60.0	InputModuleOutAddr	INT	0	**Start address of the outputs of the position decoder module** Range: • Determined by the hardware configuration	I

Input module: Channel number

Channel number

- Single-channel modules: The default value "0" that is set in the text box cannot be edited.
- Multi-channel modules: Enter the number of the channel used in this text box. Enter "0" for the first channel, etc.

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
62.0	InputChannelNo	INT	0	**Channel number of the position input module** Range: • Depends on the module	I

Output drivers

Selecting output modules

Output module

Select the output module to which your drive is connected from the drop-down list.

Based on your selection, the "Instruction" field displays the name of the instruction that you have to call for speed setpoint output.

Data used in the axis DB

Address	Name	Type	Initial value	Comment
228.0	OutputModuleType	STRING [24]	IM 174	Output driver **for the module**. It is not allowed to overwrite this parameter in manual mode.

Output module: Module address inputs

Module address inputs

The parameter is used only for modules for which it is possible to set the "Module address inputs" separately from the "Module address outputs" in the hardware configuration.

Enter the value specified for the output module in your hardware configuration in the text box.

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
82.0	OutputModuleInAddr	INT	0	**Start address of the inputs of the output module** Range: • Determined by the hardware configuration	I

Output module: Module address outputs

Module address outputs

Enter the value specified for the output module in your hardware configuration in the text box.

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
80.0	OutputModuleOutAddr	INT	0	**Start address of the outputs of the module outputs** Range: • Determined by the hardware configuration	I

Output module: Channel number

Channel number

- Single-channel modules: The default value "0" that is set in the text box cannot be edited.
- Multi-channel modules: Enter the number of the channel used in this text box. Enter "0" for the first channel, etc.

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
84.0	OutputChannelNo	INT	0	**Channel number of the output module** Range: • Depends on the module	I

Mechanics

Encoders

Specifying the steps per encoder revolution

Steps per encoder revolution

In the text box, enter the number of steps that your position decoder module outputs per encoder revolution.

The number of steps is converted to the length unit based on this value and the "Axis distance per encoder revolution" parameter (axis DB address 68.0 (Page 10784)).

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
64.0	StepsPerRev	DINT	L#1	**Steps per encoder revolution** Valid range of values: • ≥ 1 [steps/encoder revolution]	I

Incremental encoder

If the incremental encoder is read using a module that evaluates the pulses in a two-step or four-step operation, this has to be taken into account when entering the parameters.

Example:

- Encoder pulses per revolution: 500
- 4-step evaluation

In the "Steps per encoder revolution" text box, enter: 2000

Absolute encoder

The number of steps per encoder revolution is usually a power of two for absolute encoders.

Common values are:

- With single-turn encoders: 4096 and 8192
- With multi-turn encoders (12 × 12 bits in the 25-bit frame): 4096

If there, is any doubt, the correct value can be determined by moving the encoder one revolution and observing the change of the encoder value in the configuration software.

With a linear scale, enter the total number of steps for "Steps per encoder revolution" corresponding to the length of your linear scale.

Specifying the number of encoder revolutions

Number of encoder revolutions

In the text box, enter the number of encoder revolutions for an absolute encoder.

Enter "1" for a single-turn encoder.

With a **linear scale**, enter "1" for "number of encoder revolutions".

This parameter is irrelevant for incremental encoders.

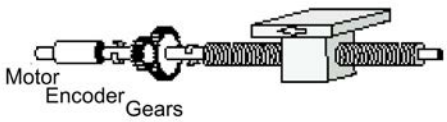
Data used in the axis DB

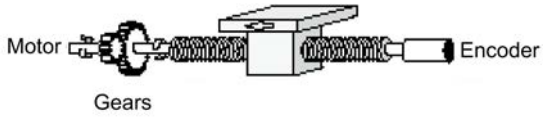
Address	Name	Type	Initial value	Comment	
72.0	NumberRevs	INT	1	**Number of revolutions** of an absolute encoder Valid range of values: • > 0	I

Specifying the axis distance per encoder revolution

Axis distance per encoder revolution

In the text box, enter the distance traversed by the axis that is equivalent to one encoder revolution. This value depends on the structure of the axis and on the encoder mounting position. You must make allowances for all transmission elements, such as couplings or gears.





For a **linear scale**, enter the length of your linear scale for "Axis travel per encoder revolution".

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
68.0	DisplacementPerRev	REAL	1.0	**Axis distance per encoder revolution** Valid range of values: • > 0.0 [length unit/encoder revolution]	I

Special features with rotary axes

The encoder must perform an integer number of steps per rotary axis revolution:

$$\frac{\text{(Rotary axis end - rotary axis start)}}{\text{Axis distance per encoder revolution}} \times \text{Steps per encoder revolution} = \text{integer}$$

Special features for rotary axes with absolute encoders

To ensure the reproducibility of all positions, the axis traversing distance covered by the encoder must be an integer multiple of the distance that the rotary axis covers in one rotation:

$$\frac{\text{(Number of encoder revolutions) x (Axis distance per encoder revolution)}}{\text{(Rotary axis end - Rotary axis start)}} = \text{Integer}$$

Example:

24-bit absolute encoder

=> Number of encoder revolutions = 4096

Rotary axis start = 0.0°

Rotary axis end = 360.0°

=> Rotary axis end - rotary axis start = 360.0°

Axis distance per encoder revolution = 45°

$$\frac{4096 \times 45°}{360°} = 512 \text{ (integer)}$$

Selecting the encoder polarity

Set encoder polarity

Here, you adjust the position feedback direction to the movement direction of the axis:

- Positive = Incremental count pulses (incremental encoder), or encoder values (absolute encoder) represent increasing actual position values
- Negative = Incremental count pulses (incremental encoder), or encoder values (absolute encoder) represent descending actual position values.

Select the corresponding option from the drop-down list.

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
74.0	PolarityEncoder	INT	1	**Set encoder polarity** Valid values: • +1 or -1	I

Controller

Specifying controller gain

Controller gain

The optimum controller gain can be determined empirically on the axis.

Increase controller gain in increments of 1.0 until the axis begins to oscillate while traversing, or while it is at a standstill. If this is the case, reduce controller gain until this tendency to oscillate is no longer perceptible.

Enter the corresponding value in the text box.

Data used in the axis DB

Address	Name	Type	Initial value	Comment
44.0	FactorP	REAL	1.0	**Controller gain** Valid range of values: • > 0.0 [1/s]

Specifying the velocity setpoint in manual mode

Velocity setpoint in manual mode

Enter a velocity setpoint for manual mode in this text box.

When manual mode is activate, the controller outputs the value in "Manual setpoint velocity" as a manipulated variable and limits it to "MaxVelocity". The position setpoint is tracked to the actual position value.

Data used in the axis DB

Address	Name	Type	Initial value	Comment
48.0	ManVelocity	REAL	0.0	**Velocity setpoint in manual mode** Valid range of values: • ≤ ± MaxVelocity (Page 10791) [length unit/s]
52.0	ManEnable	BOOL	FALSE	**Activating manual mode** Valid values: • FALSE = Deactivated • TRUE = Activated

Motor

Reference value for 100 % speed

Reference value for 100 % speed

This parameter is irrelevant when you are operating power units with a control voltage of ±10 V.

If using the OutputMM4_DP output driver, set the control value that is needed to accelerate the drive to 100 % of its speed at this parameter.

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
96.0	DriveInputAt100	REAL	10.0	**Reference value for 100 % speed** Valid range of values: • -10.0 to +10.0 [V]	I

Example

- Rated speed (100 %) of the motor: = 1380 rpm
- Motor is synchronized to: = 1380 rpm at 50 Hz

=> Enter "Reference value for 100% speed = 50 Hz

Specifying the reference value for maximum axis velocity

Reference value for maximum axis velocity

If using power unit with a control voltage of **±10 V**, enter the voltage value at which the drive reaches "Maximum velocity" in the text box (see Axis DB address 14.0 (Page 10791)).

Example:

- Rated speed of the motor: = 3000 rpm
- Motor is synchronized to: = 9 V at 3000 rpm
- Maximum speed to reach: = 1500 rpm

=> Enter the following value for "Reference value for maximum axis speed": = 4.5 V

If using the **OutputMM4_DP** output driver, enter the voltage value at which the drive reaches "Maximum speed" in the text box (see Axis DB address 14.0 (Page 10791)).

Example:

- Rated speed of the motor: = 1380 rpm
- Motor is synchronized to: = 1380 rpm at 50 Hz
- Maximum speed to reach: = 690 rpm

=> Enter the following value for "Reference value for maximum axis speed": = 25 Hz

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
88.0	DriveInputAtMaxVel	REAL	9.0	**Reference value for max. axis velocity [V]** Valid range of values: • Depends on the power unit	

Specifying zero offset compensation

Zero offset compensation

Use this parameter to implement zero offset compensation if your drive does not provide this option. The value configured here is added to the output value.

The OutputMM4_DP output driver ignores this parameter.

Data used in the axis DB

Address	Name	Type	Initial value	Comment
92.0	OffsetCompensation	REAL	0.0	**Zero offset compensation** Valid range of values: • -10.0 to +10.0 [V]

Selecting the drive polarity

Set drive polarity

Here, you select the direction of drive movement:

- Positive = Positive velocity setpoints initiate movements towards the positive direction.
- Negative = positive velocity setpoints initiate movements towards the negative direction.

Select the corresponding option from the drop-down list.

If you change "Set drive polarity", you must also change "Set encoder polarity" (Axis DB address 74.0 (Page 10785)).

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
86.0	PolarityDrive	INT	1	**Steps per encoder revolution** Valid values: • +1 or -1	

Limit switches

Activating software limit switches

Software limit switches

Software limit switches are only available for linear axes.

Monitoring begins as soon as the axis has been synchronized and software limit switch monitoring has been activated.

When software limit switch monitoring is deactivated, the "Software limit switch start" and "Software limit switch end" parameters (Axis DB addresses 6.0 and 10.0 (Page 10789)) are of no significance.

Data used in the axis DB

Address	Name	Type	Initial value	Comment
42.1	SWLimitEnable	BOOL	TRUE	**Monitoring software limit switches** Valid values: • TRUE = Monitoring activated • FALSE = Monitoring deactivated

Specifying software limit switches

Software limit switches

Software limit switches are only available for linear axes.

Linear axis: Software limit switch start/end

Monitoring begins as soon as the axis has been synchronized and software limit switch monitoring has been activated (see "Activating software limit switches (Page 10789)"). The working range is limited by means of software limit switches.

The value in "Software limit switch start" must always be less than the "Software limit switch end" value.

Incremental encoder

Initially, the axis is not synchronized after CPU startup. The axis is not synchronized and the software limit switches are not monitored unless a reference point approach or reference point setting has been completed.

Absolute encoder

Once reference point setting has been completed, the setting will not be lost at the next CPU restart. The software limit switches can be monitored.
The encoder range of the absolute encoder must cover at least the working range.

Rotary axis: Rotary axis start / Rotary axis end

"Rotary axis start" and "Rotary axis end" are used to define the range of axis position values to be displayed during one rotation of the axis.

The "Rotary axis end" value is theoretically the maximum an actual value can reach:

$$\text{Highest displayable value} = \text{Rotary axis end} - \frac{\text{Distance per encoder revolution}}{\text{Increments per encoder revolution}}$$

However, the theoretical maximum value is never displayed, as it is physically identical to the rotary axis start position (= zero).

Example:

- AxisLimitMin = 0.0°
- AxisLimitMax = 360.0°

With positive direction, the actual value counts ...359.8, 359.9, 0.0, 0.1, etc.

Pay attention to the "Axis distance per encoder revolution (Page 10784)" parameter when using absolute encoders.

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
6.0	AxisLimitMax	REAL	$1.0e^{+6}$	Linear axis: **Software limit switch end** Rotary axis: **Rotary axis end** Valid range of values: • $-2^{24}+6 \leq$ AxisLimitMin < AxisLimitMax $\leq +2^{24}-6$ [length unit]	I
10.0	AxisLimitMin	REAL	$-1.0e^{+6}$	Linear axis: **Software limit switch start** Rotary axis: **Rotary axis start** Valid range of values: • $-2^{24}+6 \leq$ AxisLimitMin < AxisLimitMax $\leq +2^{24}-6$ [length unit]	I

Dynamic response

General

Specifying the maximum axis velocity

Maximum axis velocity

Here, you configure the maximum axis velocity if the motor is controlled with the value entered in the "Reference value for maximum axis velocity" parameter (Axis DB address 88.0 (Page 10787)).

The "Maximum axis velocity" parameter is also used as a factor for calculating the value of the signal output to the drive, as well as for limiting the axis velocity.

Special features for rotary axes with absolute encoders

To enable the recording of all the rotary axis rotations, the maximum axis velocity must be limited so that the distance traversed in each scan cycle is less than half of the encoder range of the absolute encoder.

Example for single-turn encoders:

Condition => one axis rotation is equivalent to one encoder rotation.

Traversing distance / encoder range = 360.0°

Traversing distance / half of encoder range = 180.0°

Scan time = 0.01 s

$$\Rightarrow \text{"Maximum velocity"} - \frac{\text{Travel distance/half the encoder range}}{\text{Scan time}}$$

=> "Maximum velocity" < 180.0° / 0.01s = 18000°/s is equivalent to 50 axis rev/s

Data used in the axis DB

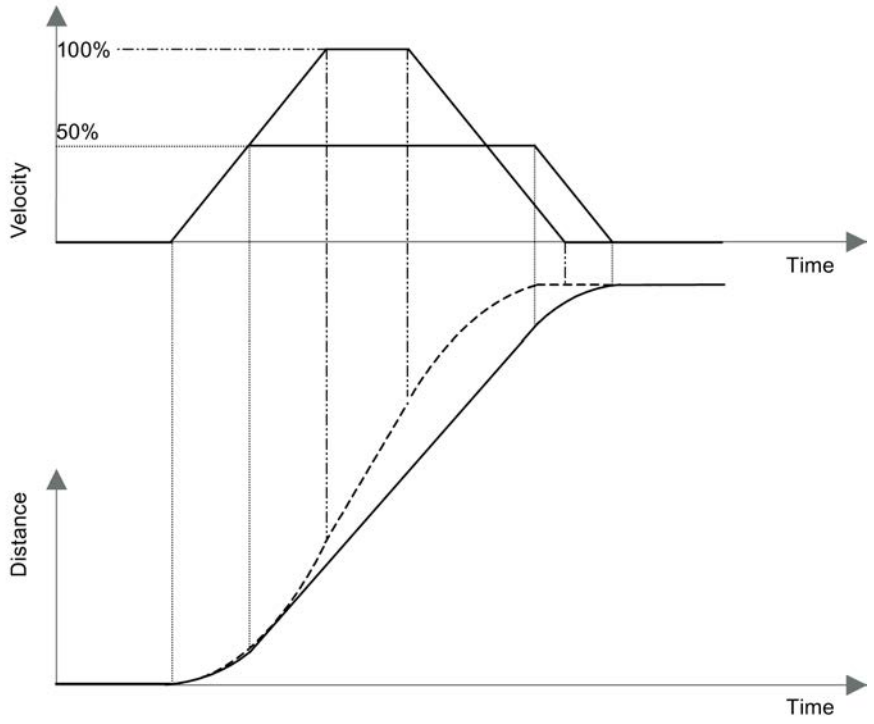
Address	Name	Type	Initial value	Comment	
14.0	MaxVelocity	REAL	0.1	**Maximum axis velocity** Valid range of values: • > 0.0 [unit of length/s]	I

Specifying velocity override

Velocity override

"Velocity override" has a sustained effect on the configured velocity of a travel instruction. Acceleration and deceleration values are not affected.

Example: Travel trends with 50 % and 100 % override



The velocity is doubled if you switch from 50 % to 100 %. As acceleration and deceleration are not affected by the "Velocity override" parameter, however, the positioning time is **not** halved.

Data used in the axis DB

Address	Name	Type	Initial value	Comment
100.0	Override	REAL	100.0	**Velocity override** Valid range of values: • 0.0 to +100.0 [%]

Specifying the maximum axis velocity

Maximum axis acceleration

The gradient of the acceleration ramp of the axis is limited to this value in controlled mode.

The maximum possible acceleration can be determined empirically.

It should only be high enough so that, when accelerating to maximum velocity (under load), it is just below the current limit of the drive.

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
18.0	MaxAcceleration	REAL	0.1	**Maximum axis acceleration** Valid range of values: • > 0.0 [length unit/s^2]	I

Specifying maximum axis deceleration

Maximum axis deceleration

The slope of the deceleration ramp of the axis is restricted to this value in controlled mode.

The maximum possible deceleration can be determined empirically.

It should only be high enough so that, when slowing down from maximum velocity (under load), it is just below the current limit of the drive.

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
22.0	MaxDeceleration	REAL	0.1	**Maximum axis deceleration** Valid range of values: • > 0.0 [length unit/s^2]	I

Specifying the scan time

Scan time

To ensure correct functioning of position control, it is important that all functions of one axis are processed in a fixed time frame. This time pattern is determined by calling all instructions of an axis in the same cyclic interrupt or synchronous cycle interrupt OB. Select the time pattern of this OB in the hardware configuration of the CPU.

For the "Scan time", enter the time pattern of the OB in which you call all instructions of an axis.

Example:

Call the program using the Easy Motion Control instructions in OB 35. You have set the cyclic interrupt time pattern of OB 35 to 10 ms in the hardware configuration.

Enter 0.010 as the "scan time".

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
0.0	Sample_T	REAL	0.01	**Scan time** of the axis instructions Recommended: • S7 300 to CPU 316 ≥ 0.020 [s] • S7 400 and as of CPU 317 ≥ 0.004 [s]	I

Displaying the ramp-up time

Rampup time

The ramp-up time denotes the time an axis needs to accelerate from a standstill to maximum velocity.

The ramp-up time is calculated based on the following parameters:

Rampup time = Maximum axis velocity × velocity override / maximum acceleration

The output field displays the calculated ramp-up time.

Displaying the ramp-down time

Ramp-down time

The ramp-down time denotes the time required to decelerate an axis without jerk limitation from maximum velocity to a standstill.

The ramp-down time is calculated based on the following parameters:

Ramp-down time = Maximum axis velocity × velocity override / maximum deceleration

The output field displays the calculated ramp-down time.

Emergency stop

Specifying emergency stop deceleration

Deceleration for emergency stop (hard stop)

At this parameter you define the slope of the time-controlled deceleration ramp used to decelerate the drive when an error with hard stop occurs.

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
54.0	EmergencyDec	REAL	1.0	**Emergency stop deceleration** Valid range of values: • > 0.0 [length unit/s^2]	I

Displaying the ramp-down time

Ramp-down time

The ramp-down time denotes the time required to decelerate an axis in emergency stop mode from maximum velocity to a standstill.

The ramp-down time is calculated based on the following parameters:

Ramp-down time = Maximum axis velocity / emergency stop deceleration

The output field displays the calculated ramp-down time.

Monitoring

Specifying the target range

Target range

The "target range" lies to the left and right of the target (see Concept Definitions (Page 10768)).

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
30.0	TargetRange	REAL	1.0	**Target range** Valid range of values: • > 0.0 [length unit]	I

Specifying the standstill range

Standstill range

When at a standstill, the axis is monitored to determine whether it is remaining at a given position or drifting away from it. The "Outside standstill range" error bit (Axis DB address 130.3 (Page 10825)) is set as soon as the axis leaves the "standstill range" without an active traversing job. The "standstill range" lies to the left and right of the target (see Concept Definitions (Page 10768)).

Recommended: The "standstill range" should be greater than the "target range" (Axis DB address 30.0 (Page 10795)).

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
34.0	StandstillRange	REAL	1.5	**Standstill range** Valid range of values: • > 0.0 [length unit]	I

Target approach monitoring

Monitoring the target approach

When target range monitoring is activated, a movement is terminated as soon as the actual position value has reached the "target range" (axis DB address 30.0) within the "Monitoring time for target approach" (axis DB address 26.0).

If target range monitoring is deactivated, axis movement is terminated as soon as the position setpoint has reached the target (see Terminology (Page 10768)).

Data used in the axis DB

Address	Name	Type	Initial value	Comment
42.0	MonitorTargetAppr	BOOL	TRUE	**Target approach monitoring** Valid values: • TRUE = Monitoring activated • FALSE = Monitoring deactivated

Specifying the monitoring time for target approach

Monitoring time for target approach

The "Monitoring time for target approach" starts as soon as the traversing axis has reached the **position setpoint** in the target area, provided the "Monitor target approach" bit has been

set (Axis DB address 42.0 (Page 10796)). If the **actual position value** has not reached the range defined in the "target area" parameter (Axis DB address 30.0 (Page 10795)) on expiration of this time, the "Target approach error" bit is set in the axis DB (Axis DB address 130.4 (Page 10825)).

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
26.0	MonTimeTargetAppr	REAL	1.0	**Monitoring time for target approach** Valid range of values: • > 2 × Sample_T [s] (scan time (Page 10793))	I

Specifying the valid maximum following error

Maximum permitted following distance

The following error monitoring function is used to check whether the actual position during a motion remains inadmissibly far behind the position setpoint. If the maximum following error has been exceeded, the "Following error exceeded" error bit is set in the axis DB" (Axis DB address 130.2 (Page 10825)).

Recommended setting: ("maximum velocity" / "controller gain") × 1.1 (see axis DB addresses 14.0 (Page 10791) and 44.0 (Page 10786))

Data used in the axis DB

Address	Name	Type	Initial value	Comment	
38.0	MaxFollowingDist	REAL	5.0	**Maximum permitted following distance** Valid range of values: • > 0.0 [length unit]	I

Download to CPU

New or modified project data must be downloaded to the CPU.

Downloading the program blocks and the TO configuration to the CPU

Proceed as follows for downloading:

1. Select the "Program blocks" object, or the "Technology objects" object in the project navigator.
2. Select the "Online > Download to device" command.

Note

Whenever you download blocks to the CPU, the TIA Portal always verifies consistency between the offline project blocks and the online blocks in the CPU.

Regardless of the marking in the project tree, all new and modified offline blocks and technology objects are downloaded to the CPU.

On each loading operation, all process values of the data blocks are reset to their initial values.

Downloading the hardware configuration to the CPU

Proceed as follows when downloading the hardware configuration:

1. Select the CPU object in the project navigator.
2. Select the "Download > Hardware configuration" command from the shortcut menu.

Downloading the program blocks and the TO configuration to the CPU

Proceed as follows for downloading:

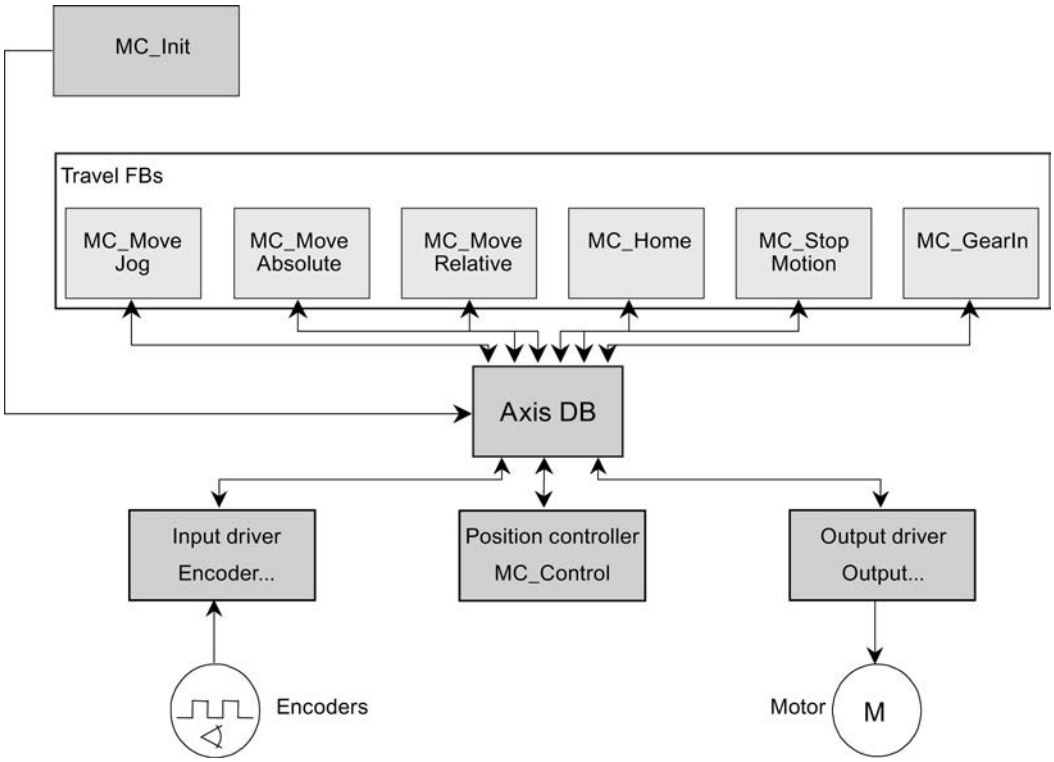
1. Select the CPU object in the project navigator.
2. Select the "Online > Download to device" command.

Programming Easy Motion Control

User Program Setup

Interaction of the instructions

The following figure demonstrates how instructions interact in Easy Motion Control.



instruction MC_Init serves to **initialize** the instructions.

The **input driver** (Encoder… instruction) reads the actual position value of an axis from the I/O module and scales this value in accordance with the length unit that you have selected.

One of the **travel instructions** generates the position setpoint.

The **position controller** (MC_Control instruction) compares the actual position value with the position setpoint and generates a velocity setpoint value from the difference.

The **output driver** (Output… instruction) scales the velocity setpoint to the suitable format and writes it to the connected I/O module.

The program for a **closed-loop control circuit** consists of the input driver (Encoder…), at least one travel instruction, the controller (MC_Control), and the output driver (Output…).

All instructions of an axis work with the same **Axis DB**. This axis DB contains all axis data needed to operate the axis and is available for each axis. Read the information on Loading the axis DB (Page 10809).

Call environment

To ensure correct functioning of position control, it is important that all functions of one axis are processed in a fixed **time frame**. This time pattern is determined by calling the instructions of Easy Motion Control in the **same** execution level. The following two methods can be applied, depending on the CPU and type of I/O connection used:

Cyclic interrupt OB (OB 30 to OB 38)

Used for central integration of the I/O, or for distributed integration via DP bus system without constant bus cycle time. The cyclic interrupt time corresponds to the position-control cycle.

Synchronous cycle interrupt OBs (OB 61 to OB 64)

The position control cycle on a continuously isochronous DP bus for I/O integration corresponds to the value you have selected in the bus configuration for "Constant bus cycle time".

This value must be entered at the axis DB in both cases.

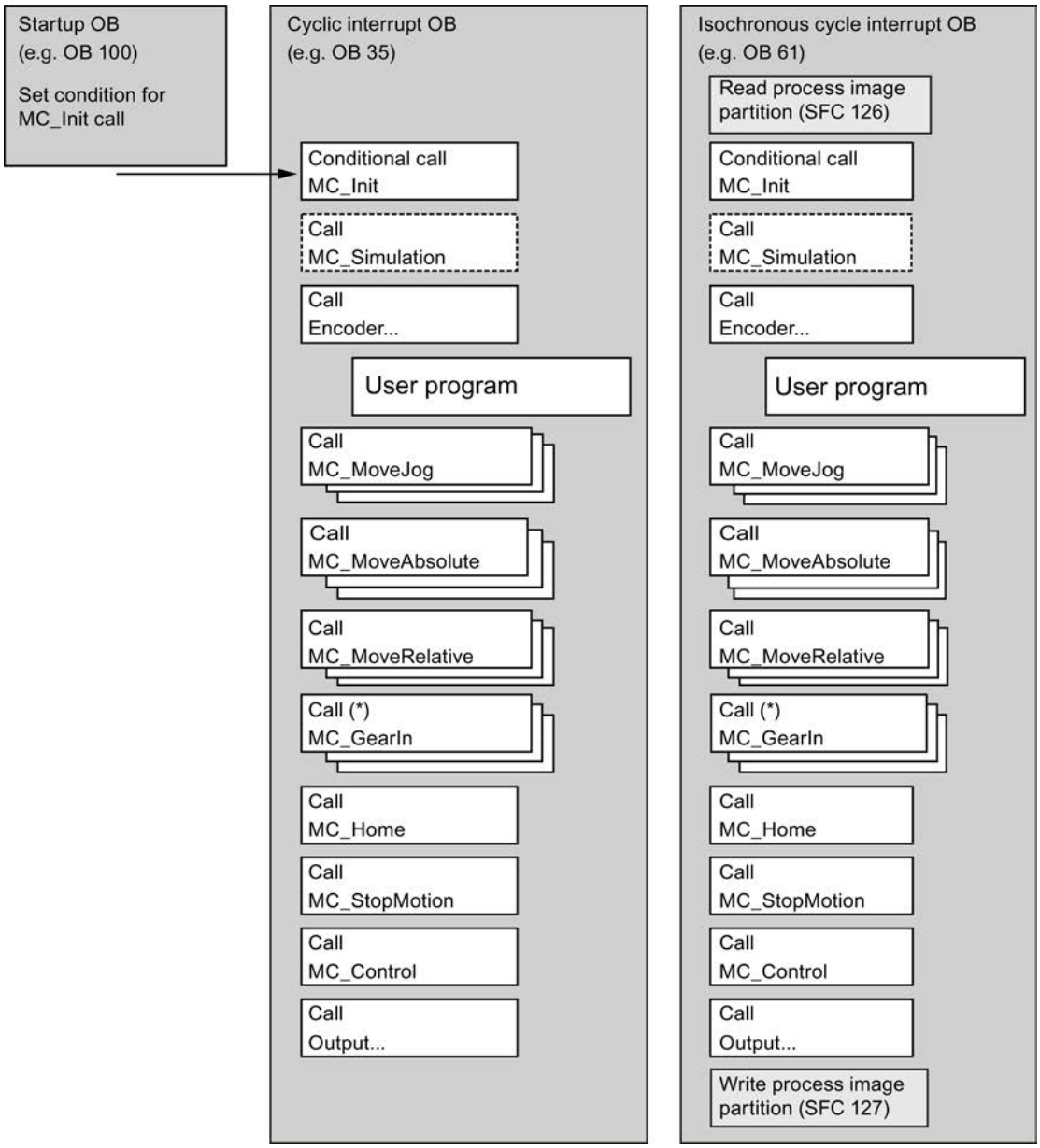
For general information on the isochronous mode, refer to the TIA Portal online help and to the Isochronous Mode Function Manual (http://support.automation.siemens.com/WW/view/en/15218045).

Please note the following special features for Easy Motion Control:

- When configuring the I/O bus and the DP slaves / IO devices (including their modules, if applicable), you must select and configure the "constant bus cycle time".
- The I/O addresses of an axis used for position detection or setpoint output must be placed in the same process image partition (PIP). When using gear axes, both gear axes must be placed into the same PIP.
- When configuring the CPU, this PIP is assigned to the synchronous cycle interrupt used.
- You need to update this PIP in the interrupt OB before and after each call of the EMC blocks using the system instructions "SYNC_PI" / "SYNC_PO".
- Evaluate the feedback from these system instructions in your program and respond accordingly. (If an error has occurred, the PIP update has possibly failed and the actual value read was therefore incorrect.)

Recommended Call Sequence

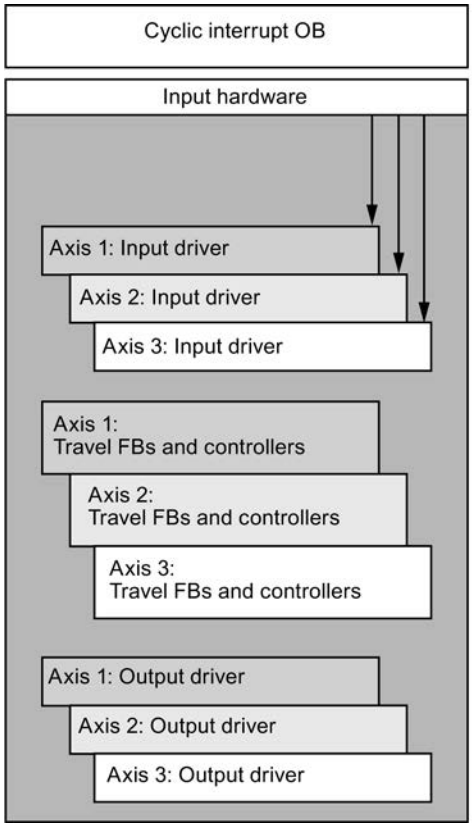
The following figure shows the call sequence recommended for the instructions of Easy Motion Control.

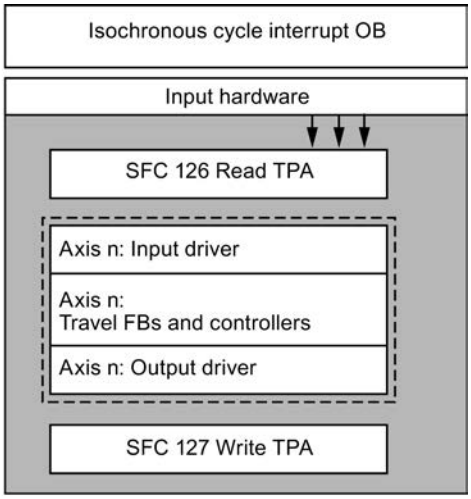


(*) A call of the gear instruction MC_GearIn is only appropriate for following axes.

Multiple Axes

We recommend the following user program structure for controlling multiple axes in one CPU:





Recommended for the cyclic interrupt OB.	Recommended for the synchronous cycle OB.
Advantage: Calling the input drivers at the start of the OB produces a minimum time offset and minimum jitter when reading the I/O.	**Advantage:** It is possible to group *all* instructions of an axis in a *single* multi-instance instruction. Due to the PIP, which is required for the synchronous cycle interrupt OB, the read I/O data are consistent irrespective of the driver call.
Disadvantage: It is not possible to group *all* instructions of an axis in a *single* multi-instance instruction.	

All interacting axes must be called in the same OB.

All axes of an isochronous application must be placed into the same process image partition.

Handling Instance Data

You can store the **instance data** for each instruction in its own DB, or in a single multi-instance DB for all instructions.

On completion of the download of the instance data of a instruction to the module, it is necessary to initialize the instruction. Set the "Init" input of the instruction accordingly. See also Initialization and Parameter Changes (Page 10804) and Saving and Loading the Axis Data Block (Page 10809).

Interconnecting the instructions

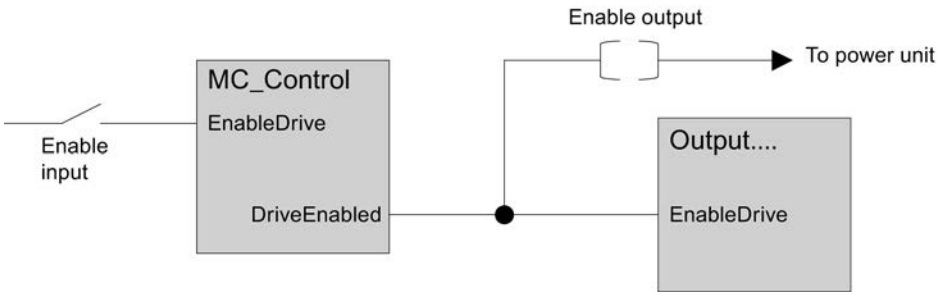
The **connections** between the instructions and the axis DB displayed in the above figure under Interaction of the instructions are handled using the following parameters:

- Axis
- Init

(See Significance of the common parameters (Page 10804))

The following interconnections are also necessary:

- The "EnableDrive" input of instruction MC_Control must be interconnected with an enable switch. instruction MC_Control will only output a speed setpoint if input "EnableDrive" = TRUE.
- Input "EnableDrive" of instruction Output… must be interconnected with output "DriveEnabled" of instruction MC_Control. "EnableDrive" = TRUE must be set in order to output a value not equal to zero at instruction Output….
- You should interconnect the "DriveEnabled" output of instruction MC_Control with the enable input of your drive.



Influences on Position Control in the CPU

Recommended for optimal position control:

- Scan time must be adhered to as closely as possible.
  - In the hardware configuration parameters for the CPU, assign the highest possible priority to the cyclic interrupt OB that calls the instructions.
  - Minimize the communication load on the cycle as far as possible with appropriate parameter settings in the hardware configuration of the CPU.
  - Use an isochronous environment to avoid having to take additional measures.
- When integrating encoder value acquisition via distributed I/O (PROFIBUS DP / PROFINET IO), set the maximum possible transmission rate for the I/O bus (for example, 12 Mbps).

Importance of Shared Parameters

The following table lists the parameters that are used in all instructions.

Parameter	Declaration	Data type	Description
EN	INPUT	BOOL	Enable input of LAD/instructionD box
ENO	OUTPUT	BOOL	Enable output of LAD/instructionD box
Axis	IN-OUT	STRUCT	Interconnect this parameter with the axis DB. For a DB of the type AXIS_REF, the interconnection is: • Axis := DBname.Ax For a DB that contains a variable of the type AXIS_REF, the interconnection is: • Axis := DBname.Variable name.Ax
Init	IN-OUT	BOOL	Interconnect these parameter with an initialization bit in the axis DB that is not used by any other instruction ("Init.I0" … "Init.I31"; refer to Bit array for the initialization function (Page 10812)). With Init = TRUE, the instruction completes its initialization and then resets the bit. For a DB of the type AXIS_REF, the interconnection is: • Init := DBname.Ax.Init.Ix For a DB that contains a variable of the type AXIS_REF, the interconnection is: • Init := DBname.Variablename.Ax.Init.Ix when Ix = I0 … I31

Initialization and Parameter Changes

The functionality of Easy Motion Control instructions is based on the data stored in the axis DB. This is where parameters describing the basic axis itself can also be found. If these parameters are changed, **all** Easy Motion Control instructions called in your user program must run their initialization **once** following the change. These parameters are identified in configuration sections (Page 10775) by the following symbol:

Bit array "Init.Ix" is defined for initialization in the axis DB and all instructions are assigned the "Init" parameter. The instructions run their initialization routine if their "Init" input is set and then reset the interconnected initialization bit.

The bits in the "Init.Ix" bit array are set in the axis DB:

- At every call of instruction MC_Init.
- By the configuration software in the event of any change in corresponding parameters of the axis DB

Interconnect the "Init" parameter with an initialization bit in the axis DB that is not in use by any other instruction for all Easy Motion Control instructions called in your user program. This interconnection ensures that the instructions run their initialization routine whenever this becomes necessary as a result of parameter changes or CPU restart ("Init.I0" … "Init.I31"; refer to Bit array for the initialization function (Page 10812)).

Call instruction MC_Init once in the following situations:

- After every startup of the CPU
- If you have changed one of the parameters identified by the following symbol in the configuration sections (Page 10775) without using the configuration software:

For additional information on the initialization instruction, refer to MC_Init (Page 7034).

WARNING
Parameters identified by the following symbol in the configuration section (Page 10775) may only be changed **when the axis is at a standstill**:
If you do not observe this rule, controlled axis movement cannot be guaranteed.

WARNING
During initialization, specific auxiliary values are calculated and saved to the static parameters of the instructions and to the axis DB. For this reason, you are only permitted to overwrite the instance data blocks and the axis DB when the axis is at a standstill.
If you do not observe this rule, controlled axis movement cannot be guaranteed.

Sequence of Travel Movements

Start of a travel order

The input parameters of travel instructions are interpreted once at the start of a travel order. Changing these values while an order is in progress does not affect current travel.

Maximum Travel Distance

The travel distance per movement is limited to the distance that is equivalent to 2^{24} steps of

$$\text{Maximum traversing distance} = 2^{24} \times \frac{\text{"Axis distance per encoder revolution"}}{\text{"Steps per encoder revolution"}}$$

Example:

Axis distance per encoder revolution	= 2.5 mm
Steps per encoder revolution	= 4096
Maximum distance = 2^{24} x 2.5 mm / 4096	= 10240 mm

Exception:

The MC_MoveJog and MC_GearIn instructions do not limit the traversing distance of a rotary axis.

Overriding travel

A currently active travel instruction can always be

- Overridden by starting a different travel instruction
- Cancelled by the start of instruction MC_StopMotion
- Aborted by setting manual mode

When overridden by another travel instruction, the axis is decelerated or accelerated to the velocity specified in the overriding instruction and then approaches the new target without intermediate stop. The overridden travel instruction sets the "CommandAborted" output.

If the new motion requires a change in direction of the axis, the axis is first stopped and then moved in the opposite direction (reversal).

For safety reasons, the deceleration of the two travel instructions involved in the override must coincide. This ensures that on reversing direction, the **original** target of the movement is not overtraveled.

Multiple use of a travel instruction

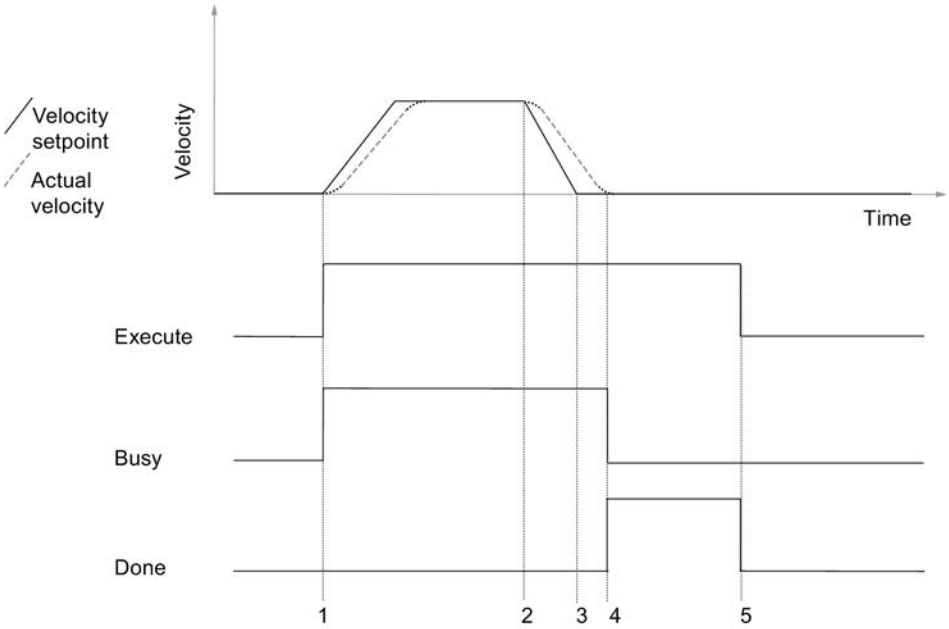
A travel instruction can be called several times for an axis in your program if necessary.

Error Response

All errors that occur are indicated in the axis DB, where they must be acknowledged by the user (see Error Displays and Error Acknowledgment in the axis DB (Page 10820)). Each error will abort any travel order that is currently active. A new travel order can only start if there is no longer any error pending.

Signal Flow Diagrams

The following diagram shows the signal flow of a movement without error.



1 Start of the travel FB

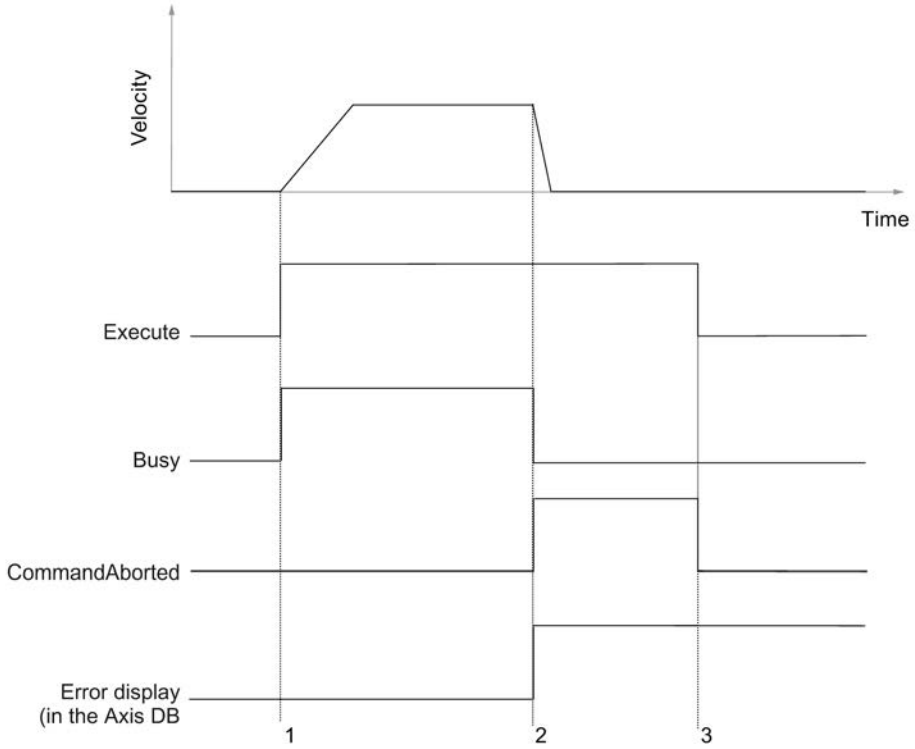
2 Brake application point

3 Setpoint has reach target position

4 Axis has reached the target range

5 Reset of "Execute" resets "Done". The process is completed. If "Execute" is reset before the target position has been reached, "Done" is set for one cycle after the target position has been reached and is then reset again.

The following diagram shows the signal flow of a movement with error.



1 Start of the travel FB

2 Error occurred

3 Reset of "Execute" resets "CommandAborted" feedback. If "Execute" is reset before the error has occurred, "MotionAborted" is set for one cycle after the error has occurred and is then reset again.

Axis data block

Function

The axis DB is the central database of Easy Motion Control. It contains:

- Parameters that describe the basic axis (for example, I/O addresses)
- Position control parameters that are calculated based on these basic parameters during initialization of the instructions of Easy Motion Control.
- Actual values of the position control (for example, position, error states, etc.).
- Initialization bits for coordinating the start of the instructions

Interaction of these parameter types is ensured if you take one of the following measures:

- Edit and transfer the axis DB using the configuration software.
- After having changed specific parameters, ensure that instruction MC_Init (Page 7034) is called.

In both cases, the plausibility of the entered parameters is checked wherever possible, and the values internally required by Easy Motion Control are computed.

The configuration software also analyzes the type of changes and uses this information to control the loading processes.

Saving and Loading the Axis Data Block

WARNING

The entire axis DB can only be overwritten when the axis is at a standstill.

You may **NOT** download the axis DB in RUN mode.

Correct movement of the axis cannot be ensured if you download the axis DB to the module while ignoring the notes in the following sections.

Offline

After specific parameters have been changed, all instructions of Easy Motion Control must complete an initialization routine.

These parameters are identified in the description of the axis DB by the following symbol:

Changing and loading these parameters:

- Is only possible when the axis is at a standstill
- Must include setting the initialization bits (see Initialization and Parameter Changes (Page 10804)).

Online

Transferring parameters that were changed online (for example, the coordinate of the reference point for absolute encoders) to the offline management is not supported.

Axis parameters

All information on length, velocity, and deceleration provided in the "Configuring Easy Motion Control (Page 10774)" section refers to an arbitrary, yet uniform length unit. All times are specified in seconds.

In the tables of that section, the parameter name from the axis DB is specified in the "Name" column, while the parameter name used in the configuration software is specified in the "Comments" column.

Note

Parameters not listed are reserved for internal use by the Easy Motion Control instructions and must not be altered.

Actual values

Overview

The following parameters in the axis DB show information on the current status of the moving axis. You must not change the contents of these parameters.

Actual position value

Displays the actual position value.

Address	Name	Type	Initial value	Comment
104.0	ActPosition	REAL	0.0	**Actual position value** Range of values: Floating point number [length unit]

Following error

Following error = current position setpoint - actual position value

Address	Name	Type	Initial value	Comment
108.0	FollowingDistance	INT	0	**Following error** Range of values: Floating point number [length unit]

Residual distance

Indicates the distance remaining to the target position during movements with defined target (MC_MoveAbsolute, MC_MoveRelative).

Travel without target (MC_MoveJog, MC_Home) and errors with hard stop reset the remaining distance to 0.

Address	Name	Type	Initial value	Comment
112.0	RemainingDistance	REAL	0.0	**Distance remaining** to target Range of values: Floating point number [length unit]

Setpoint velocity

Displays the velocity setpoint of the axis calculated by the travel instruction during a movement.

Address	Name	Type	Initial value	Comment
116.0	NomVelocity	REAL	0.0	**Setpoint velocity** Range of values: Floating point number [length unit/s]

Actual velocity

The actual velocity of the axis is displayed here.

Address	Name	Type	Initial value	Comment
120.0	ActVelocity	REAL	0.0	**Actual velocity** Range of values: Floating point number [length unit/s]

Axis synchronized

When using an incremental encoder:

- FALSE: After CPU startup or detection of an encoder error
- TRUE: After a reference point approach or reference setting

When using an absolute encoder:

- FALSE: Before initial reference setting
- TRUE: After reference setting

Address	Name	Type	Initial value	Comment
124.0	Sync	BOOL	FALSE	**Axis synchronized** FALSE = Not synchronized TRUE = Synchronized

Current encoder value

The current encoder value is displayed here.

Address	Name	Type	Initial value	Comment
136.0	EncoderValue	DINT	L#0	**Current encoder value** Range of values: Decimal [steps]

Bit Field for the Initialization Function

Bit array function

All Easy Motion Control instructions must be initialized after every CPU startup and after specific parameters have been changed.

For this purpose, the "Init.Ix" bit array is defined for initialization in the axis DB and all instructions are assigned the "Init" parameter. The instructions run their initialization routine if their "Init" input is set. On completion, the instructions reset the interconnected initialization bit.

You must interconnect the "Init" parameter with a bit of the bit array at each call of a instruction. This bit may not be used by another instruction (refer to Initialization and parameter changes (Page 10804)).

Whenever the MC_Init instruction is called, or if you change a parameter in the configuration software, which is not activated unless the instructions have completed their initialization, all bits of this array is set simultaneously to TRUE. This allows all instructions to carry out an initialization with their next call.

Address	Name	Type	Initial value	Comment
278.0	Init	STRUCT	TRUE	Bit array for initialization of the instructions

User Program as a Multi-instance FB

If you want to save all parameters of your travel instructions to a single DB, or use the same program for several axes, create the user program in a multi-instance FB.

Place the axis data in this multi-instance DB, as well. To do this, declare a static variable of the type AXIS_REF from the "Easy Motion Control" library in your multi-instance FB.

Interconnect the Axis and Init parameters of the instructions of Easy Motion Control with this variable.

Call instruction MC_Init conditionally as the first instruction in your multi-instance block. This condition must be set in the startup OB (for example, OB100) and reset after instruction MC_Init has been called.

New multi-instance DBs can only be created using the TIA Portal, or the LAD/STL/instructionD Editor.

Commissioning Easy Motion Control

Safety Information

Introduction

Read the following notes before you start commissioning.

> **WARNING**
>
> **The commissioning functions result in a movement of the axis.**
>
> Uncontrolled travel motions of the axis during commissioning and operation can cause serious injury and material damage.
>
> To avoid injury to persons and damage to objects, take the following precautions:
>
> - Install an EMERGENCY STOP switch in the vicinity of the computer. This is the only way to ensure that the system is reliably switched off in the event of a hardware or software failure.
> - Install safety limit switches for direct control of the power units of all drives.
> - Ensure that nobody has access to the area of the system where there are moving parts.

Commissioning the axis technology object

The commissioning editor helps you with commissioning and the function test for the axis technology object. You can run the following tests and measurements:

- Wiring test (Page 10814)
- Velocity measurement (Page 10814)
- Homing the axis (Page 10815)
- Distance measurement (Page 10816)

Procedure

To open the commissioning editor of the axis technology object, follow these steps:

1. Open the "Technology objects" folder in the project tree.
2. Open the corresponding instance of the AXIS_REF technology object (e.g. EMC_AXIS_1) in the project tree.
3. Double-click on the "Commissioning" object.

Remedy in the case of errors

If errors occur during commissioning, you will find information about possible remedies in section Error profiles and remedies during commissioning of the user program (Page 10817).

Additional information

The corresponding tooltips include detailed information on the parameters and buttons of the commissioning user interfaces.

Testing the wiring

The Wiring test is used to determine the direction of rotation of the axis.

Procedure

To perform the Wiring test, follow these steps:

1. Click on "Wiring test" in the configuration view.
2. Click the "Activate" button.
   A query regarding activation of manual control is displayed.
3. Click "OK".
   An online connection is created between STEP 7 (TIA Portal) and the CPU.
4. Enter a velocity as a percentage of the maximum velocity in the "Specified velocity" entry field.
5. Enter a value in the "Ramp-up time" entry field.
6. Make sure that the start position permits a movement of the axis in both directions.
7. Click the "Jog backwards" or "Jog forwards" button to move the axis.
8. In "Result", specify whether the axis has moved in the requested direction.
9. If you have selected "Yes" as the result, click "OK" in the window that appears.
10. If you have selected "No" as the result, click "Yes" in the window that appears.
    The parameters "Encoder polarity" and "Drive polarity" are changed.

Measuring the axis velocity

The velocity measurement determines the maximum axis velocity.

Procedure

To perform velocity measurement, follow these steps:

1. Click on "Velocity measurement" in the configuration view.
2. Click the "Activate" button.
   A query regarding activation of manual control is displayed.

3. Click "OK".
   An online connection is created between STEP 7 (TIA Portal) and the CPU.
4. Enter a velocity as a percentage of the maximum velocity in the "Specified velocity" entry field.
5. Enter a value in the "Ramp-up time" entry field.
6. Make sure that the start position permits a sufficiently long travel distance in at least one direction.
7. Click the "Jog backwards" or "Jog forwards" button and hold it until the value in the "Actual output" output field has reached the "Setpoint output".
   Easy Motion Control calculates the maximum axis velocity and displays it in the "Determined value" output field.
8. If you want to save the calculated axis velocity in the project, click the "Apply" button.
   The calculated value overwrites the "Maximum velocity" in the online/offline data management of the technology object.

Set reference point

A reference point is set for absolute encoders, and this is referred to as absolute encoder adjustment in this case.

Procedure

To set a reference point for an absolute encoder, proceed as follows:

1. Click on "Homing the axis" in the configuration view.
2. Click the "Activate" button.
   A query regarding activation of manual control is displayed.
3. Click "OK".
   An online connection is created between STEP 7 (TIA Portal) and the CPU.
4. Enter a velocity as a percentage of the maximum velocity in the "Specified velocity" entry field.
5. Enter a value in the "Ramp-up time" entry field.
6. Click the "Jog backwards" or "Jog forwards" button until the axis has traveled to the reference point.
   Alternatively:
   Move the axis to the reference point manually.
7. In the "Input of reference position" input field, enter a position value as reference value.
   The offset between the reference position and the physical axis position is determined and saved in the axis DB.
8. In the window that appears, confirm whether you wish to save the offset for the absolute encoder adjustment in the offline data management of the technology object.

Note

If you replace the absolute encoder, you will have to set the reference point again.

Measuring the axis distance

The distance measurement determines the axis distance per encoder revolution.

Procedure

Proceed as follows to perform the distance measurement:

1. Click on "Distance measurement" in the configuration view.
2. Click the "Activate" button.
   A query regarding activation of manual control is displayed.
3. Click "OK".
   An online connection is created between STEP 7 (TIA Portal) and the CPU.
4. Enter a velocity as a percentage of the maximum velocity in the "Specified velocity" entry field.
5. Enter a value in the "Ramp-up time" entry field.
6. Click the "Jog backwards" or "Jog forwards" button until the axis has traveled to the start position.
   Alternatively:
   Move the axis to the start position manually.
7. Enter a position value as the start position in the "Input of start position" entry field.
8. Click the "Jog backwards" or "Jog forwards" button until the axis has traveled to the end position.
   Alternatively:
   Move the axis to the end position manually.
9. Enter a position value as the end position in the "Input of end position" entry field.
   Easy Motion Control calculates the axis distance per encoder revolution and displays it in the output field "Determined value".
10. If you want to save the calculated axis distance per encoder revolution in the project, click "Apply".
    The calculated value is entered in the online/offline data management of the technology object.

Error profiles and remedies during commissioning of the user program

Axis does not Move Despite Travel Order

If the axis is not moving irrespective of a pending travel command, check the following conditions:

Error	Remedies
Wiring of the drive is faulty	• Check whether the "DriveEnabled" output of the MC_Control instruction is continuously interconnected with the enable input of the power unit. • Check whether the "DriveEnabled" output of the MC_Control instruction is interconnected with the "EnableDrive" input of the output driver.
I/O addresses or channel numbers are incorrect	Set the correct I/O addresses and channel numbers: In the axis DB, the **start** addresses of the modules are expected. From the start address and channel number (counting method: from channel 0), the driver blocks calculate the module-specific I/O addresses for setpoint and actual values.
Simulation mode switched on	Switch off simulation mode ("Sim" = FALSE). If the simulation mode is switched on ("Sim" = TRUE), **no** I/O values are read or written.
Manual mode switched on with missing setpoint velocity	If manual mode is switched on ("ManEnable" = TRUE), specify a "manual setpoint velocity" ("ManVelocity" <> 0.0 [length unit/s]).
Drive enable not set	Set the "EnableDrive" input of the MC_Control instruction to TRUE.
Axis error pending	If the "group error" bit ("Error") is set in the axis DB, the type of axis error is visible in the bit arrays for the error display: • After the MC_Init instruction is called, the "Stop status requiring acknowledgment" error is always set ("Err.StoppedMotion"). Acknowledge the error by setting the "Group acknowledgment" "ErrorAck" = TRUE. • After you have rectified incorrect values in the case of a parameter assignment error, you need to restart MC_Init. • In the case of an encoder error, check whether the position input module is fully configured (with the help of the module configuration software, if necessary). • Other errors: Remedy and acknowledge the error by setting the "Group acknowledgment" "ErrorAck" = TRUE.
"Group acknowledgment" is not reset	If a "Group acknowledgment" ("ErrorAck") you have set is not detected and reset by Easy Motion Control , then: • Prevent the Easy Motion Control instructions from remaining in their initialization branch due to the constant call of the MC_Init instruction. • Check whether the position decoder module responds to the error acknowledgment.
"Velocity override" not 0%	Configure the "Velocity override" = 0% ("Override").
Travel command not processed by program	Set the parameter "Busy" = TRUE in the travel instruction.
"Parameter of travel instruction inadmissible" ("Err.DataErr") or "Target outside travel range" ("Err.TargetErr")	If you download the axis DB to the CPU without using the Easy Motion Control configuration software, you must subsequently call the MC_Init instruction. MC_Init stores the auxiliary values in the axis DB.

Axis Moves without Travel Order

Possible causes of axis movement without a travel command are:

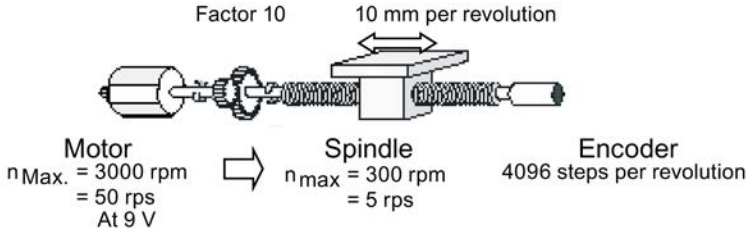
Error	Causes	Remedies
The axis starts moving very quickly.	The direction of control action has been set incorrectly.	Adjust the "Encoder polarity" ("PolarityEncoder") and "Drive polarity" ("PolarityDrive") parameters with the help of the "Wiring test" wizard. In this way, you adapt the instructions to your hardware installation.
The axis is drifting slowly.	The power unit of the drive is enabled and the axis is **not** in control (MC_Control instruction outputs "DriveEnabled" = FALSE).	—
	The analog output is not correctly wired with the power unit of the drive.	Correct the wiring between the analog output and the power unit of the drive.
	Either the input driver or the output driver is not accessing the modules belonging to the drive because the module address or the channel number in the axis DB is set incorrectly.	Adjust the parameters with the help of the "wiring test" wizard.

Axis Velocity Other than Expected or Axis Oscillates

Are the parameters describing the axis correct?

There are certain relations between these parameters as shown in the following example.

- "Controller gain" ("FactorP"):
  The axis may oscillate if the selected "Controller gain" ("FactorP") factor is too high.
- "Maximum axis velocity" ("MaxVelocity")
- "Reference value for maximum axis velocity" ("DriveInputAtMaxVel"):
  Example of a ±10 V control:



The following relationship holds:
With an output voltage of 9 V at the output module, the drive rotates at a speed of 3000 rpm. This speed is proportional to a spindle speed of 300 rpm or 5 rps and the slide moves by a distance of 10 mm per spindle revolution. The maximum velocity of the slide is thus 10 mm/s • 300 U/min = 50 mm/s.
In other words, set:

– "Maximum axis velocity" = 50 mm/s and "Reference value for maximum axis velocity" = 9 V.

- "Steps per encoder revolution" ("StepsPerRev"):
  You can check whether the number of steps produced by the encoder coincides with the configured value by observing the parameter "Encoder value" with the configuration software, while you turn the **encoder once** by hand.
  Example: The "encoder value" changes by 4096 steps per spindle revolution.
- "Axis distance per encoder revolution" ("DisplacementPerRev"):
  This parameter assigns a specific distance to one revolution of the encoder. You can check whether this is correct by observing the parameter "Actual position" with the configuration software, while you turn the **encoder once** by hand.
  Example: The "actual position" changes by 10 mm per spindle revolution.
- "Scan time" ("Sample_T"):
  Scan time is used together with the distance covered since the last scan in order to calculate axis velocity The time must correspond to the call time of the OB in which the instructions of Easy Motion Control are called.
  "Sample_T" must be entered with the second unit in the axis DB.

Positioning Not Accurate Enough

If positioning lacks precision:

- Check the settings in the "Steps per encoder revolution" parameter ("StepsPerRev").
- Calculate the "Axis distance per encoder revolution" parameter ("DisplacementPerRev") with maximum possible precision ("Distance measurement" wizard).

See also

Measuring the axis distance (Page 10816)

Maximum Following Distance Exceeded

Overshoot of the maximum following error has different causes:

- In order for the "maximum axis velocity" ("MaxVelocity") to be reached at all, the following rules must apply:

$$„MaxFollowingDistance“ \quad >= \frac{„MaxVelocity“}{„FactorP“}$$

- The configured acceleration cannot be reached by the real axis.
- Incorrect settings for the "Maximum axis velocity" ("MaxVelocity") and "Reference value for maximum axis velocity" ("DriveInputAtMaxVel") parameters, or of the "Reference value for 100 % speed" ("DriveInputAt100") (refer to Axis velocity not as expected, or axis oscillates (Page 10818)).

The MC_GearIn instruction displays neither "InGear" nor "Coupled"

The following causes prevent the MC_GearIn instruction from displaying "InGear" or "Coupled":

- The slave axis is about to couple to a moving master axis without having (yet) reached the gear speed.
- The slave axis has released the coupling due to an error.

MC_GearIn instruction displays "InGear", but not "Coupled"

instruction MC_GearIn displays "InGear", but not "Coupled" due to the following causes:

- The slave axis may have reached the gear speed for (possibly only for one cycle), but has lost this speed:
  - The acceleration or deceleration of the master axis exceeds the capability of the slave axis based on its configuration and the gear ratio.
  - The slave axis cannot reach the velocity that is determined by the velocity setpoint of the master axis and by the gear ratio.

Diagnostics

Error Displays and Error Acknowledgement in the Axis DB

Error displays and error acknowledgment

All errors detected by the instructions are displayed in the axis DB as follows:

- By setting the group error "Error"
- By means of error-specific display "Err.xxxx"

With errors requiring acknowledgment, an error acknowledgment must be set after the error has been eliminated (parameter "ErrorAck" in the axis DB; see below).

There are three types of errors with different error responses:

- Errors where perfect functioning of position control is **no longer** guaranteed lead to a **hard stop**:
  instruction MC_Control brings the axis to a standstill along the deceleration ramp stored in "EmergencyDec". instruction MC_Control will then reset its "DriveEnabled" to "0".
  Once the error has been eliminated, set the error acknowledgment in the axis DB.
  "DriveEnabled" is then set again.
- Errors where position control still functions lead to a **soft stop**:
  The axis is slowed down at the deceleration that is configured in the active travel instruction.
  The axis retains controlled operation and "DriveEnabled" is **not** reset.
  In this situation, the movement can only be overridden by means of MC_StopMotion.
  Once the error has been eliminated, set the error acknowledgment in the axis DB.
- Error in the axis parameters (**parameter error**):
  instruction MC_Init checks the parameter entered in the Axis Db as far as possible. In the event of an error, it sets "Err.ConfigErr" bit and enters the precise cause of error in "Config.Err_xxx".
  Parameter errors **cannot** be rectified by acknowledgment. Approach of the axis is prevented. After having corrected the invalid parameter entry, you must once again call instruction MC_Init. Parameter error displays are then reset once there is no longer any error.

Note

The following applies to errors with hard and soft stop:

No travel order is possible as long as one or several unacknowledged errors are pending.

Exception:

If the axis only overshoots a software limit switch, a travel order for reversing the movement is possible. This error is automatically acknowledged once the axis has cleared the software limit switch.

Note

If a travel block (for example, MC_MoveJog, or MC_MoveAbsolute) has detected the error, the "Error" output is set at the corresponding instruction.

If a driver instruction has detected the error, the precise cause of error is indicated at this instruction (see Input drivers (Page 7058)).

Error display in the axis DB

The group error is TRUE as long as there is at least one error.

Address	Name	Type	Initial value	Comment
124.1	Error	BOOL	FALSE	**Group error** FALSE = No error TRUE = Group error

Error acknowledgment in the axis DB

When group acknowledgment is set, all acknowledgeable error displays are deleted.

Group acknowledgment is only effective when the axis is at a standstill and after initialization.

On completion of the evaluation of error acknowledgment, "ErrorAck" is reset automatically.

Address	Name	Type	Initial value	Comment
124.2	ErrorAck	BOOL	FALSE	**Group acknowledgment** for all errors FALSE = No acknowledgment TRUE = Acknowledge error

Axis error

Error with soft stop

Description

The following rules apply to the errors described below: TRUE = Error is pending.

Errors requiring acknowledgment are identified by the following symbol:

The following table lists all errors with soft stop.

Address	Name: Err.xxx	Description		
128.0	SWLimitMinExceeded	**Software limit switch start exceeded**		—
		Cause:	• The actual position value is outside the working range. • Replacement of the absolute encoder.	
		Remedy:	• Move the axis out of the range of the limit switch. • Deactivate the software limit switch ("SWLimitEnable" = FALSE), synchronize the axis and then reactivate the software limit switch.	
128.1	SWLimitMaxExceeded	**Software limit switch end exceeded**		—

Address	Name: Err.xxx	Description		Q
		Cause:	• The actual position value is outside the working range. • Replacement of the absolute encoder.	
		Remedy:	• Move the axis out of the range of the limit switch. • Deactivate the software limit switch ("SWLimitEnable" = FALSE), synchronize the axis and the reactivate the software limit switch.	
128.2	TargetErr	**Target outside of travel range**		Q
		Cause:	• **MC_MoveAbsolute**: – The target is in the range of a software limit switch. – The target is < "Rotary axis start", or ≥ "Rotary axis end". – The calculated traversing distance is greater than the distance that is equivalent to 2^{24} steps of your encoder. – Auxiliary variables were not calculated in the axis DB, because instruction MC_Init was not called. • **MC_MoveRelative**: – The travel distance entered leads to a target on a software limit switch. – The specified travel distance is greater than the distance that is equivalent to 2^{24} steps of your encoder. – Auxiliary variables were not calculated in the axis DB, because instruction MC_Init was not called. • **MC_Home**: – The reference point coordinate is in the range of a software limit switch. – The reference point coordinate is < "Rotary axis start", or ≥ "Rotary axis end". – Auxiliary variables were not calculated in the axis DB, because instruction MC_Init was not called.	
		Remedy:	• Correct the parameters. • Call the MC_Init instruction.	
128.3	NoSync	**Axis not synchronized**		Q
		Cause:	instruction MC_MoveAbsolute was started with unsynchronized axis.	
		Remedy:	Synchronize the axis (reference point approach, reference setting).	
128.4	DirectionErr	**Invalid travel direction entered**		Q
		Cause:	Further movement into the range of a software limit switch is not allowed.	
		Remedy:	Start a travel order that moves the axis out of the range of the software limit switch.	
128.5	DataErr	**Invalid parameter in travel instruction**		Q

Address	Name: Err.xxx	Description		Q
		Cause:	**MC_MoveHome**: • "Velocity" < 0 • "Velocity" > 0 and "Acceleration" ≤ 0 • "Velocity" > 0 and "Deceleration" ≤ 0 • "Velocity" > 0 with absolute encoder • "Position" lies outside the numerical range on a linear axis. • Auxiliary variables were not calculated in the axis DB, because instruction MC_Init was not called. **MC_MoveAbsolute**, **MC_MoveRelative** and **MC_MoveJog**: • "Velocity" ≤ 0 • "Acceleration" ≤ 0 • "Deceleration" ≤ 0 • The entered or calculated target coordinate is outside the numerical range (for MC_MoveAbsolute and MC_MoveRelative). • Gear ratio denominator (RatioDenominator") ≤ 0 (MC_GearIn) • Gear ratio numerator ("RatioNumerator") = 0 (MC_GearIn) • A travel instruction was canceled and "Deceleration" of the overriding instruction is inconsistent with the deceleration configured at the canceled instruction (see Overriding a movement (Page 10773)). • Auxiliary variables were not calculated in the axis DB, because instruction MC_Init was not called.	
		Remedy:	• Specify admissible values. • Initialize the instructions. • Call the MC_Init instruction.	
128.6	StartErr	**Start from current axis status not possible**		Q
		Cause:	• In a travel order, or at MC_StopMotion, the Execute input was set while the axis was in error state or manual mode ("ManEnable" = TRUE), or while a MC_StopMotion instruction was active. • instruction MC_Home was not started while the axis was at a standstill. • Both directions are set in instruction MC_MoveJog. • Invalid status at the leading axis for instruction MC_GearIn: – Error with hard stop – MC_Home is active – Manual mode ("ManEnable" = TRUE)	
		Remedy:	• Eliminate any errors. • Make sure that the master axis is in a permitted state.	
128.7	DistanceErr	**Overtravel**		Q

Address	Name: Err.xxx	Description		
		Cause:	A movement without target (MC_Home, MC_MoveJog, MC_GearIn) has covered a distance that is equivalent to more than 2^{24} encoder steps. The axis is braked at about this point. A MC_MoveJog or MC_GearIn instruction has decelerated a synchronized linear axis onto a software limit switch. Moreover, "SWLimitMinExceeded" or "SWLimitMaxExceeded" is set as well in most cases.	
		Remedy:	Make sure the travel distance does not exceed the limit specified above (see Maximum Travel Distance (Page 10805)).	
129.0	MasterErr	**Master axis in invalid state**		
		Cause:	The master axis was set to an invalid state during gear coupling: • Error with hard stop • MC_Home is active • Manual mode ("ManEnable" = TRUE)	
		Remedy:	Correct the invalid state of the master axis.	

Error with hard stop

Description

The following rules apply to the errors described below: TRUE = Error is pending.

Errors requiring acknowledgment are identified by the following symbol:

The following table lists all errors with hard stop.

Address	Name: Err.xxx	Description		
130.0	StoppedMotion	**Axis is in stop state requiring acknowledgment**		
		Cause:	The axis is in a stop state requiring acknowledgment: • After initialization by calling instruction MC_Init • As the result of an error listed in this table	
		Remedy:	Rectify any errors that might possibly be present.	
130.1	EnableDriveErr	**Drive enable missing**		
		Cause:	"EnableDrive" is not set at instruction MC_Control.	
		Remedy:	Set "EnableDrive".	
130.2	FollowingDistErr	**Following error exceeded**		

Address	Name: Err.xxx	Description		Q
		Cause:	The axis is not following the setpoints or is too slow in following.	
		Remedy:	• Switch on the drive. • Optimize "Controller gain" (Axis DB address 44.0 (Page 10786)). • Adjust "Set encoder polarity" (Axis DB address 74.0 (Page 10785)) to "Set drive polarity" (Axis DB address 86.0 (Page 10788)). • Increase the valid maximum following error (Axis DB address 38.0 (Page 10797)).	
130.3	StandstillErr	**Outside standstill range**		Q
		Cause:	• The axis has moved out of "Standstill range" without a travel order. • "Standstill range" is less than "Target range". • "Set encoder polarity" or "Set drive polarity" is incorrectly configured.	
		Remedy:	• Check the connection to the setpoint input of the power unit. • Adjust the parameters "Standstill range"/"Target range" parameters (Axis DB address 34.0 (Page 10796) / 30.0 (Page 10795)) • During first commissioning: Test the wiring (Axis DB address 74.0 (Page 10785) / 86.0 (Page 10788)).	
130.4	TargetApproachErr	**Error on target approach**		Q
		Cause:	The actual value has not reached the "Target range" within the "Monitoring time for target approach".	
		Remedy:	• Optimize the "target range" and "Monitoring time for target approach" (Axis DB address 30.0 (Page 10795) / 26.0 (Page 10796)). • Check the drive and axis.	
130.5	EncoderErr	**Encoder error**		Q
		Cause:	• The input driver has detected an encoder error. The instance DB of the driver DB indicates the precise cause of error. See Input drivers (Page 7058). • Input "EncErr" was set at the input driver (see Input drivers (Page 7058), error handling section).	
		Remedy:	Eliminate the cause of the error.	
130.6	OutputErr	**Error at output driver**		Q
		Cause:	Input "OutErr" was set at the output driver (see Output drivers (Page 7081), error handling section).	
		Remedy:	Eliminate the cause of the error.	
130.7	ConfigErr	**Group error: Axis data incorrectly configured**		**Acknowledgment not possible**
		Cause:	The parameters in the axis DB are defective. For exact cause of error, see Parameter errors (Page 10827).	
		Remedy:	Adjust the faulty parameter and make sure that instruction MC_Init is called.	

Address	Name: Err.xxx	Description		Q
131.0	DriveErr	**Drive error**		**Acknowledgment not possible**
		Cause:	The output driver has detected an error at the power unit / drive. The instance DB of the driver DB indicates the precise cause of error. See Output drivers (Page 7081).	
		Remedy:	Eliminate the cause of the error.	

Parameter errors

Parameter errors

The following table shows the parameter errors that are likely to occur in Easy Motion Control.

The "Address" column contains the address of the error display in the axis DB. The "Parameter address" column points to the address of the faulty parameter in the axis DB (see Structure of the axis data block (Page 7093)).

Address	Name Config.xxx	Parameter error	Parameter address
132.0	Err_AxisLimit	"Software limit switch start" < -2^{24} [length unit] "Software limit switch end" > $+2^{24}$ [length unit] "Software limit switch end" ≤ "Software limit switch start"	6.0 (Page 10789) , 10.0 (Page 10789)
132.1	Err_MaxVelocity	"Maximum axis velocity" ≤ 0.0 [length unit/s]	14.0 (Page 10791)
132.2	Err_MaxAcceleration	"Maximum acceleration" ≤ 0.0 [length unit/s^2]	18.0 (Page 10792)
132.3	Err_MaxDeceleration	"Maximum deceleration" ≤ 0.0 [length unit/s^2]	22.0 (Page 10793)
132.4	Err_MonTimeTargetAppr	"Monitoring time for target approach" ≤ 0.0 [s]	26.0 (Page 10796)
132.5	Err_TargetRange	"Target range" ≤ 0.0 [length unit]	30.0 (Page 10795)
132.6	Err_StandstillRange	"Standstill range" ≤ 0.0 [length unit]	34.0 (Page 10796)
132.7	Err_MaxFollowingDist	"Maximum permitted following distance" ≤ 0.0 [length unit]	38.0 (Page 10797)
133.0	Err_EmergencyDec	"Deceleration for hard stop" ≤ 0.0 [length unit/s^2]	54.0 (Page 10794)
133.1	Err_StepsPerRev	"Steps per encoder revolution" < 1	64.0 (Page 10782)
133.2	Err_DisplacementPerRev	"Axis distance per encoder revolution" ≤ 0.0 [length unit]	68.0 (Page 10784)
133.3	Err_NumberRevs	"Number of encoder revolutions" < 1 for absolute encoder	72.0 (Page 10783)
133.4	Err_PolarityEncoder	"Encoder polarity" not equal to ±1	74.0 (Page 10785)
133.5	Err_PolarityDrive	"Drive polarity" not equal to ±1	86.0 (Page 10788)
133.6	Err_DriveInputAtMaxVel	"Reference value for maximum axis velocity" ≤ 0.0, or > "DriveInputAt100"	88.0 (Page 10787) , 96.0 (Page 10787)
133.7	Err_AxisLength	Axis length > 2^{24} [length unit], or axis length > (2^{31} -1) [steps]	6.0 (Page 10789) , 10.0 (Page 10789)
134.0	Err_EncoderRange	Encoder range does not match axis length (see "Axis Distance per Encoder Revolution", Axis DB address 68.0 (Page 10784))	6.0 (Page 10789) , 10.0 (Page 10789) , 68.0 (Page 10784) , 72.0 (Page 10783)

Address	Name Config.xxx	Parameter error	Parameter address
134.1	Err_MaxVelRotaryAxis	Maximum velocity and scan time do not match rotary axis length (see "Maximum axis velocity" Axis DB address 14.0 (Page 10791))	14.0 (Page 10791) , 0.0 (Page 10793) , 68.0 (Page 10784) , 72.0 (Page 10783)
134.2	Err_DriveInputAt100	"Reference value for 100 % speed" ≤ 0	96.0 (Page 10787)

JOB_ERR interrupts

JOB_ERR interrupts

JOB_ERR (Hex)	JOB_ERR (Dec)	JOB_ERR (Int)	Meaning
80A0	32928	-32608	Negative acknowledgment on reading from module. Module removed during reading process or module defective
80A1	32929	-32607	Negative acknowledgment on writing to module. Module removed during write process or module defective
80A2 [1)]	32930	-32606	Protocol error at layer 2
80A3 [1)]	32931	-32605	Protocol error at user interface / user
80A4 [1)]	32932	-32604	Communication faulty at K-bus
80B0	32944	-32592	Data record / order unknown
80B1	32945	-32591	Length data incorrect. FM_TYPE parameter in the channel DB is not correctly set for the module being used.
80B2	32946	-32590	Configured slot is not assigned.
80B3	32947	-32589	Actual module type not same as setpoint module type.
80C0 [1)]	32960	-32576	Module has not yet prepared the data to be read.
80C1 [1)]	32961	-32575	Data of a similar write order has not yet been processed on the module.
80C2 [1)]	32962	-32574	Module is currently processing the maximum possible number of orders.
80C3 [1)]	32963	-32573	Required resources (memory, etc.) are currently occupied.
80C4 [1)]	32964	-32572	Communication error
80C5 [1)]	32965	-32571	Distributed I/O not available.
80C6 [1)]	32966	-32570	Priority class aborted (restart or background)
8522	34082	-31454	Channel DB or parameter DB too short. Data cannot be read from the DB (write order).
8532	34098	-31438	DB number of parameter DB too high (write order)
853A	34106	-31430	Parameter DB not available (write order).
8544	34116	-31420	Error at nth (n > 1) read access to a DB after an error has occurred (write order).
8723	34595	-30941	Channel DB or parameter DB too short. Data cannot be written to the DB (read order).
8730	34608	-30928	Parameter DB in CPU is write-protected. Data cannot be written to the DB (read order).
8732	34610	-30926	DB number of parameter DB too high (read order).

JOB_ERR (Hex)	JOB_ERR (Dec)	JOB_ERR (Int)	Meaning
873A	34618	-30918	Parameter DB not available (read order).
8745	34629	-30907	Error at nth (n > 1) write access to a DB after an error has occurred (read order).

1) Errors 80A2 to 80A4 and 80Cx are temporary, i.e., they can be rectified automatically after a waiting time.

Using Team Engineering 12

12.1 Shared commissioning of projects

12.1.1 Basics for shared commissioning

Introduction

As part of Team Engineering , you have the option to perform shared commissioning of projects. Several editors can access one CPU at the same time with up to five engineering systems (ES) in this process.

A special advantage is that parts of a master project can be edited independently and offline at the same time during the commissioning phase. The changes of the other editors are displayed in the dialog "Software synchronization" during loading and synchronized automatically, if possible.

Certain online functions can also be executed at the same time from several engineering systems on the shared CPU, for example:

- Monitoring blocks on the CPU
- Controlling blocks on the CPU
- Trace functions

The following online functions cannot be executed at the same time:

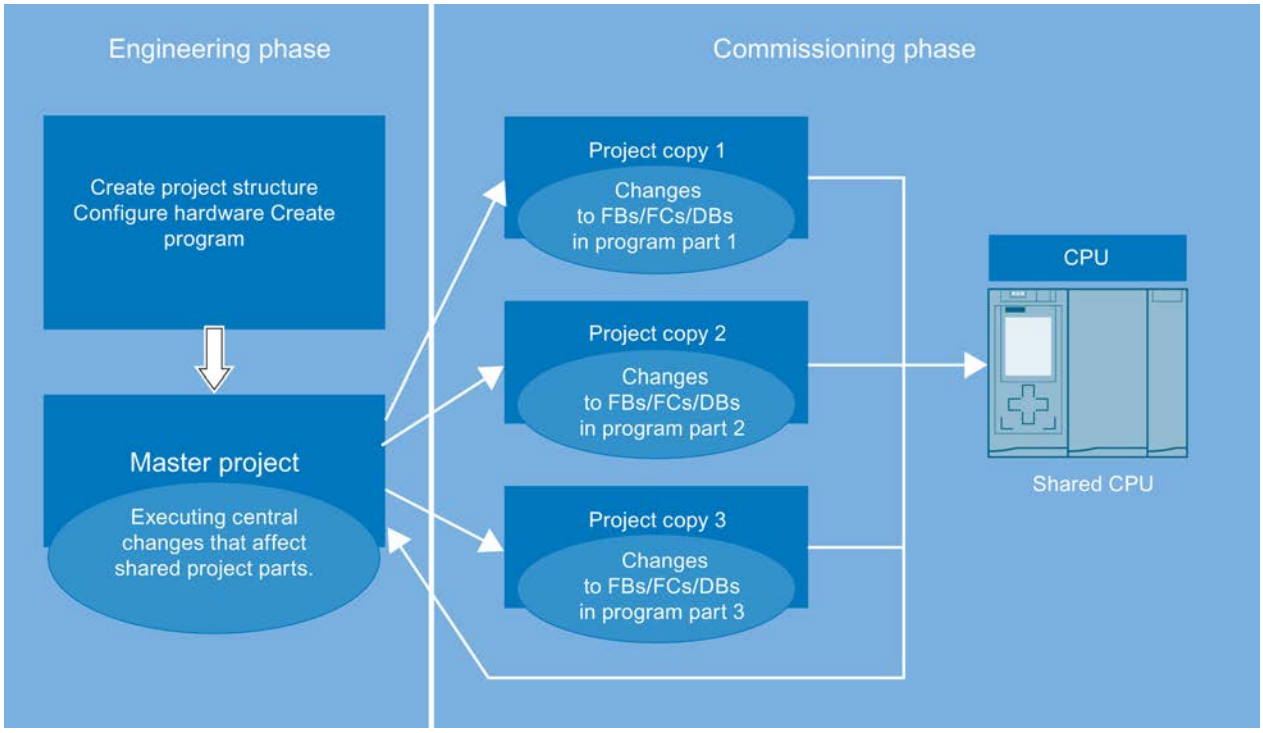
- Forcing: Only one ES can force on the CPU.
- Loading: Only one ES can load to the CPU.

You can also find additional information on the topic of Team Engineering at Service and support in Siemens Industry online support (http://support.automation.siemens.com/WW/view/en/82142829).

Creating a master project

A "master project" structured according to the specified rules (Page 10837), which already contains the complete configured hardware configuration with all required tags and blocks, is used as the basis for the shared commissioning. This project is loaded in the jointly used CPU

and then distributed as "master project" by means of project copies to up to five participating engineering systems.



Working with project copies

Each engineering system only processes the parts it was assigned within the specific project copy. This is especially important to avoid conflicts during loading and unintentional overwriting of blocks that were already modified by other engineering systems.

Each ES loads its parts to the shared CPU after editing. The TIA Portal helps you during download to the CPU. The changes made since the last download are displayed in the dialog "Software synchronization before loading to a device" (hereafter referred to as "synchronization dialog"), by means of an online-offline comparison. You are provided recommendations during download to the CPU about how you can synchronize the offline data you have edited with the existing online data.

If there are "competing changes", you may have to synchronize them manually.

Competing changes are:

- when different editors edit the same block in different project copies at the same time.
- when PLC tags (I, Q, M, T, C) are added, changed or deleted in the project copies.
- when blocks that have relations to the hardware configuration are edited in the project copies.
- when F blocks are edited in the project copies.

Each editor loads their changes only after all conflicts have been removed.

The procedure for editing and loading of project copies can be repeated as often as necessary by the individual engineering systems until the shared commissioning is complete.

Integrating project copies into the master project once again

The individual project copies must once again be integrated into the master project at the end of commissioning at the same time. This way you also back up the data that only exist in the offline projects and have not been downloaded to the CPU. These include, for example, the project languages and text lists created in the project copies.

To do this, open the master project and one project copy as reference object. You determine the differences between the two projects by performing an offline/offline comparison using the comparison editor. Copy the edited objects from the respective project copy to the master project. Repeat this step for all project copies.

When the shared commissioning is completed, you have an executable master project with all created project data.

The integration of the project data into the master project is required even if you determine during the commissioning phase that the shared central elements need to be expanded or corrected, for example, the PLC tag table needs to be expanded, or a new hardware component needs to be added.

Notes on compatibility

The following compatibility rules apply to shared commissioning:

- If an engineering system with TIA Portal V13 or later is online on the CPU and performs a download, a second ES cannot establish an online connection regardless of the version.
- If an engineering system with TIA Portal < V13 is online on the CPU, a second ES cannot establish an online connection regardless of the version.

See also

Requirements for shared commissioning (Page 10833)

Procedure for shared commissioning (Page 10834)

12.1.2 Requirements for shared commissioning

Software and hardware requirements

For shared commissioning of projects, the minimum software and hardware requirements for the installation of TIA Portal V13 or later must be met.

The following requirements also apply:

Software:

- You have installed TIA Portal V13 or later and the "SIMATIC STEP 7 Professional" software package on the involved engineering systems.
- The same software version must be installed on all engineering systems.
- The master project must be one created with V13 or later.

Hardware:

- You have access to a fully configured CPU S7-1500 as of firmware version V1.5.
- The involved engineering systems (max. 5) can establish an online connection to this CPU.

Requirements for shared, parallel working on the project

The following requirements are in effect for shared commissioning of projects:

- You have created a project with complete hardware configuration and fully programmed user program that can be commissioned.
- The project was downloaded to the CPU and defined as "master project".
- Multiple engineering systems have access to this CPU and can establish an online connection to this CPU.
- You are familiar with the defined rules and procedures for working together on one CPU.

Note on compatibity mode

The functions for shared commissioning are not available in compatibility mode.

See also

Software and hardware requirements STEP 7 (Page 91)

Basics for shared commissioning (Page 10831)

Procedure for shared commissioning (Page 10834)

Rules for shared commissioning (Page 10837)

12.1.3 Procedure for shared commissioning

Introduction

If you are performing shared commissioning a project as part of team engineering, it is very important that all editors of the project observe a defined procedure.

Only when the specified procedure is observed are changes and corrections in the project automatically synchronized and applied, and changes of individual editors not unintentionally overwritten during loading or even lost in case of competing changes.

Procedure for creating the master project

The master project is created in the following steps:

1. Create a master project that includes the entire project structure.
2. Fully configure the hardware for the master project.
3. Define a language as project language, which must also be used exclusively by all participating engineering systems.
4. Create all tags and blocks that are required in the master project.
5. Create your own folders and groups for the blocks to be edited by the individual engineering systems.
6. Create a fully programmed and executable user program.
7. Download the master project to the shared CPU.
8. Save the master project following every download.

For this, also observe the Rules for shared commissioning (Page 10837).

Procedure for shared commissioning

Shared commissioning takes place with the following steps:

1. Download the master project to the shared CPU.
2. Create copies of the master project and distribute them to the editors involved in the project on the associated engineering systems.
3. Inform all editors who is to edit the specific parts of the project copies and who may download to the CPU.
4. Have the project copies edited in the individual engineering systems.
5. After editing, the individual engineering systems download the changes one after the other to the CPU.
6. All changes are automatically detected during loading by means of an online-offline comparison. The displayed synchronization dialog offer you recommendations for synchronizing the modified data when possible. You may first have to download blocks that were edited by other editors to your ES before you can download your changes to the CPU. This may be needed to update the corrections of the other editors that have already been downloaded to the CPU in your project copy.
   The following synchronization options are available:
   - Download block: There are edited blocks on the CPU that must be updated in your project copy.
   - Download block: There are new blocks on the CPU that must be downloaded to your project copy.
   - Blocks with competing changes: System-supported synchronization is not possible in this case; the conflict must be resolved manually.
7. Manually resolve any conflicts that may occur due to competing changes or changes to central objects.
8. Download your project copy to the CPU when all conflicts have been removed.

9. Save your project copy following every download.
10. Repeat editing of project copies and download to the CPU as often as necessary until shared commissioning is complete.
11. Integrate the completely edited project copies into the master project once again in order to also back up the project data that only exist offline.

You should also observe the Rules for shared commissioning (Page 10837).

Procedure for manual synchronization of competing changes

Manual synchronization of competing changes is performed in the following steps:

1. Start the comparison editor to manually resolve a conflict displayed in the synchronization dialog during download to the CPU. Select the shared CPU in the project tree and select the "Compare > Offline/Online" command in the shortcut menu. You can define certain actions depending on the status of the objects. Note, however, that you can only perform actions in one direction during synchronizing.
2. First, select the action "Upload from device" for all blocks that were changed by other editors on the CPU and which you want to apply.
3. If necessary, perform a detailed comparison of the blocks to identify differences between your offline blocks and the online blocks loaded on the CPU.
4. Manually correct the competing changes to the blocks.
5. Then download the affected blocks to the CPU using the "Continue without synchronization" command.
6. Save your project or project copy following every download.

For this, also observe the Rules for shared commissioning (Page 10837).

Note

To prevent online changes occurring again while an engineering system is being manually synchronized, no downloading to the shared CPU should be performed by another participating engineering system during the manual synchronizing.

Procedure for integrating project copies into the master project

You integrate project copies into the master project with the following steps:

1. Open the master project and the project copy to be integrated as a reference object.
2. Copy the program parts you have edited from the respective project copy to the master project and confirm overwriting the existing objects. You can also use the comparison editor to apply your program parts to the master project.
3. Save the master project and download it to the shared CPU with the "Continue without synchronization" command.
4. Save the master project following every download.

For this, also observe the Rules for shared commissioning (Page 10837).

Procedure for modifying central objects within the master project

Modification of central objects that have an effect on all program parts is performed with the following steps:

1. Stop further processing of the project copies.
2. Integrate all existing project copies one after the other into the master project as described above.
3. Download the master project to the shared CPU with the "Continue without synchronization" command.
4. Make the required modification on the shared objects such as in the hardware configuration, for example, or in the PLC tag table. The procedure is the same for modification of other central objects such as technology objects, F blocks, OBs, etc.
5. Download the master project into the CPU once again when modification is complete.
6. Save the master project following every download.
7. Distribute the updated project copies to the respective engineering systems for further processing.

For this, also observe the Rules for shared commissioning (Page 10837).

See also

Basics for shared commissioning (Page 10831)

Requirements for shared commissioning (Page 10833)

Rules for shared commissioning (Page 10837)

12.1.4 Rules for shared commissioning

Introduction

If you are performing shared commissioning a project as part of team engineering, it is very important that all editors of the project observe predefined rules for successful cooperation.

Rules for the master project

The following rules are to be observed:

- Create a master project that is suitable for shared editing.
- Divide the user program into program parts that are independent of each other.
- Use "Groups" to visually separate the program parts from each other.
- Use a main OB and a central FC for each program part which calls the functions of the part program.
- If possible, create a separate PLC tag table for each part.
- In the master project, specify a project language which cannot be changed in the project copies.
- If you want to exchange data between the program parts, use the FC and FB interface parameters (IN, OUT, INOUT) or global data blocks.
- Use global data blocks to save the data for the individual program parts, no bit memories.
- Do not assign different names for blocks with identical addresses.
- Do not assign identical names for blocks with different addresses.

Example of the program structure in the master project

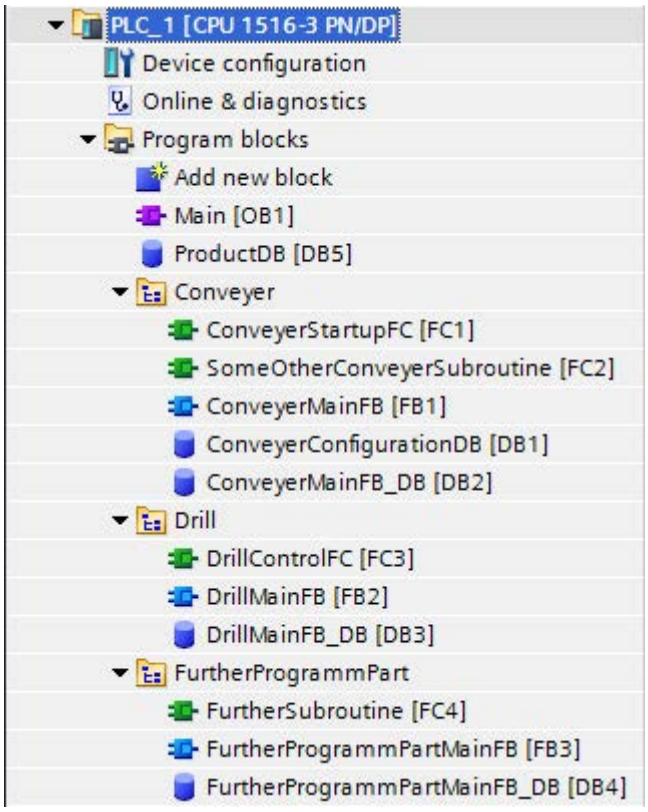
Here is an example of a team-capable, structured master project that is suitable for parallel editing during shared commissioning.

The program parts edited by the individual Engineering Systems are:

- Program part 1: "Conveyer"
- Program part 2: "Drill"
- Program part 3: "FurtherProgramPart"

Each program part contains a "main FB", which calls the specific lower-lever functions for this program part.

Example: in the "Conveyer" program part, the "ConveyerMainFB" calls the "ConveyerStartup" function.



In team engineering, this subdivision within the project structure allows the parallel editing of individual program parts and the automatic synchronization of changes during the shared commissioning phase.

Rules for shared working on one CPU

Observe the following rules for shared working:

- Editor only edits the blocks they have been assigned in the project copy in the assigned groups.
- Only FBs, DBs, FCs and UDTs may be changed in the project copies, i.e. the following program elements should **not** be edited in the project copies:
  - Hardware configurations
  - PLC tag table
  - Organization blocks
  - Technological objects
  - Text lists and project language
  - F blocks
- Global data blocks should be used in programming instead of bit memories.

- IEC timers and counters should be used during programming instead of SIMATIC timers and counters.
- The project language specified in the master project may not be changed in the project copies.

Rules for editing shared, central objects

Observe the following rules for shared central objects:

- Changes to shared central objects always have to be made by means of the master project. These objects include:
  - Hardware configurations
  - PLC tag table
  - Organization blocks
  - Technological objects
  - Text lists and project language
  - F blocks
- Before the shared central objects can be modified in the master project, the various working versions within the project copies must first be integrated into the master project once again.
- The modification can then be made within the master project, and the master project can then be downloaded to the shared CPU.
- Once modification is complete, new project copies can be created and distributed to the engineering systems for further processing.

Rules for downloading to the CPU

Observe the following rules for downloading:

- Editors may only download the project copies one after the other; only one engineering system can download at any given time.
- No other engineering system may be connected online with the CPU during download.
- Perform the synchronization that is offered automatically in the synchronization dialog. This ensures that changes from other editors are not unintentionally overwritten.
- Correct the name conflicts that are shown in the synchronization dialog, which you rename in your offline block before downloading to the CPU. A name conflict is indicated if you want to download to the CPU a block that already exists with identical name on the CPU.

- So-called "competing changes" must be corrected manually. Competing changes come about when two editors work on the same block on two engineering systems at the same time. The first editor can still download his or her changes to the CPU without any problems. However, if the second editor wants to download the same block with the changes the editor has made, the synchronization dialog displays a conflict that cannot be removed automatically. The previous changes made to this block would be undone by overwriting during download. The editors have to decide in this case which change is to be applied and which one is to be discarded. Alternatively, you can manually merge the changes with the help of the detailed comparison of the blocks in the comparison editor.
  Such changes should always be avoided by implementing a good project structure and corresponding agreements between the editors.
- Changes to central objects are displayed in the synchronization dialog but cannot be synchronized automatically. Changes to central objects always have to be made by means of the master project.
- Save your project or project copy following every download.

Note

Creating and subsequent deleting of objects in the project copies

Please note that a software synchronization is also required before downloading when you create a new object in a project copy and then delete this again immediately afterwards.

The creation of an object causes a change in the internal data management which cannot be undone by the subsequent deleting of the object. As a result it is necessary to perform a synchronization before the next loading if the newly created object was deleted immediately thereafter and no change in the existing object is visible within the project copy.

Rules for online functions

The following rules are to be observed when using online functions:

Monitoring and controlling:

- Up to five Engineering Systems can be monitored and controlled simultaneously on the CPU blocks.
- A specific code block can only be monitored and controlled by one Engineering System at a time.
- However, additional Engineering Systems can simultaneously monitor and control other code blocks.

Force

- Forcing can only be implemented on the CPU from one Engineering System at a time.
- The Engineering Systems that starts the forcing has taken over the forcing job exclusively. Although the other Engineering Systems receive the information that a forcing job is running, they cannot access this job and cannot also see any results.
- As soon as the Engineering System to which the force job belongs ends the online connection, the force job can be applied by another ES that establishes an online connection to the CPU and changes the existing force job.

Trace functions:

- Up to four Engineering Systems can perform trace functions simultaneously.
- The Engineering Systems that starts the tracing has taken over the trace job exclusively. Although the other Engineering Systems can see this job in the trace editor, they cannot access it.
- As soon as the Engineering System to which the trace job belongs closes the trace editor, the trace job can be taken over by another ES that re-opens the trace editor.

See also

Basics for shared commissioning (Page 10831)

Requirements for shared commissioning (Page 10833)

Procedure for shared commissioning (Page 10834)

12.2 Exchanging data with Inter Project Engineering (IPE)

12.2.1 Basics of Inter Project Engineering (IPE)

Introduction to Inter Project Engineering (IPE)

The functionality of Inter Project Engineering, hereinafter referred to as IPE, is used to exchange the controller data in a source project between different projects with the help of a device proxy.

You can then transfer this data to other projects, for visualization in HMI, for example, and use it there for further configuration.

Inter Project Engineering gives you a simple way of providing controller data from a PLC programming across multiple projects for an HMI configuration.

Using the object "Device Proxy Data", you exchange controller data across multiple projects consistently and easily without redundant configuration overheads.

This makes it possible for HMI project engineers to work on a project without the hardware configuration having to be present in their project. This means they can work on the PLC project and on the HMI project at the same time.

You can exchange the following controller data between projects via the device proxy:

- Program blocks
- Technological objects
- PLC tags
- PLC alarms

The following data is automatically included in the exchange:

- Controller interfaces
- Configured communication processors and communication modules

This automatic data exchange ensures that the interfaces and the configured communication processors and modules are transferred along with the data you selected. This data is required to allow you to continue working consistently and without problems in your target project.

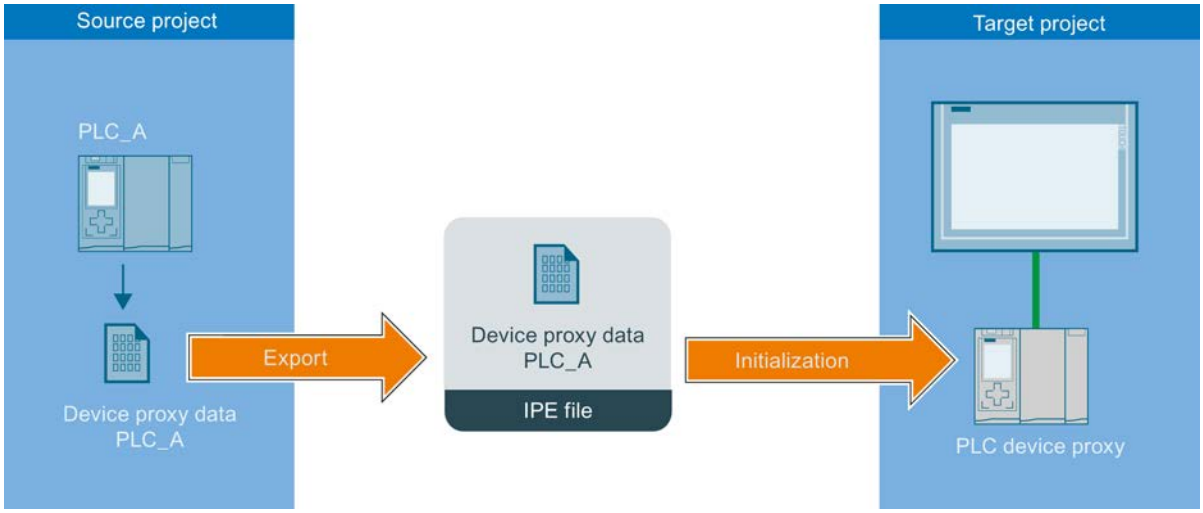
Cross-project data exchange

The following options are available for cross-project data exchange with Inter Project Engineering:

- Exchanging controller data via IPE file
- Exchanging controller data via project file

Exchanging controller data via IPE file

The figure below shows cross-project data exchange via an IPE file.

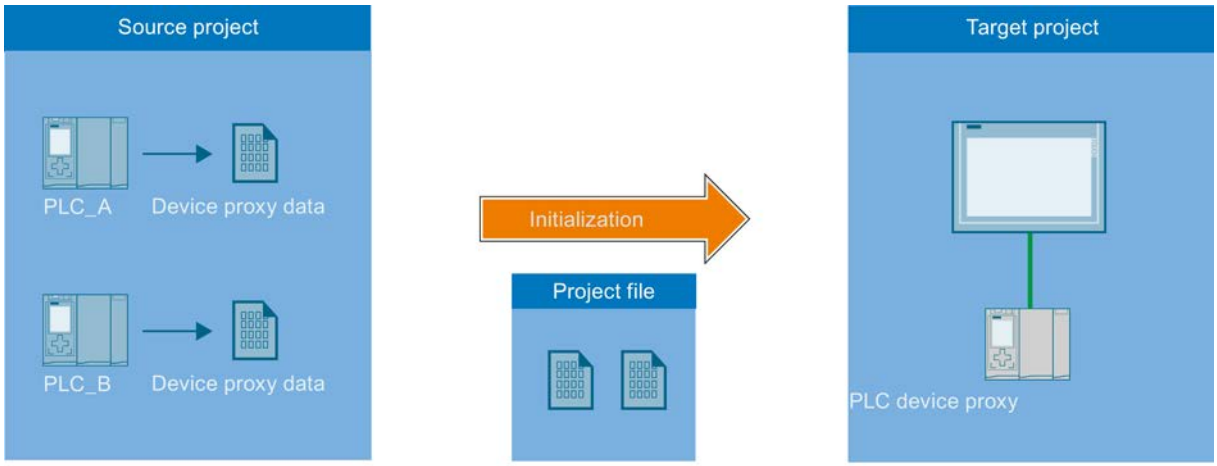


The PLC controller data from the source project is transferred to the PLC in the target project via the object "Device Proxy Data".

Controller data that is exchanged via an IPE file must originate from TIA Portal Projects V13 or higher.

Exchanging controller data via a project file

The figure below shows cross-project data exchange via a project file.



The PLC controller data from the source project is transferred to the PLC in the target project via a project file.

You can easily transfer and continue to use controller data from older TIA Portal projects as well as STEP 7 Classic projects that are not integrated in TIA Portal in your current TIA Portal projects.

See also

Requirements for Inter Project Engineering (IPE) (Page 10844)

Procedure for working with Inter Project Engineering (IPE) (Page 10845)

12.2.2 Requirements for Inter Project Engineering (IPE)

Software and hardware requirements

To use the functionality of Inter Project Engineering, the minimum software and hardware requirements for the installation of TIA Portal V13 or higher must be met.

The following requirements also apply:

- You have installed TIA Portal V13 or higher and the "SIMATIC STEP 7 Professional" software package.
- You are using a CPU of the S7-1200/1500 or S7-300/400 series.
- The same software version must be installed on all involved controllers.

Requirements for IPE

The requirements for accepting controller data from projects are as follows:

- You have created a project with hardware configuration and PLC data that can be transferred to another project.
- You have made sure that the project is consistent before you transfer the controller data by means of the device proxy data.
- You are familiar with the defined procedures for Inter Project Engineering.

Notes on compatibility mode

The functions for Inter Project Engineering are not available in compatibility mode.

See also

Basics of Inter Project Engineering (IPE) (Page 10842)

Procedure for working with Inter Project Engineering (IPE) (Page 10845)

12.2.3 Procedure for working with Inter Project Engineering (IPE)

Exchanging controller data across projects

You can use the functionality of Inter Project Engineering to exchange existing controller data between different projects with the help of a device proxy and to continue using this data in another project. The device proxy provides you with a consistent and convenient way of exchanging data between projects without redundant configuration overheads.

Procedure for exchanging controller data via an IPE file

To exchange data via an IPE file, follow these steps:

1. Using TIA Portal V13 or higher, create a project with all required controller data and an executable user program.
2. Configure the hardware required for the project.
3. Compile the project to ensure consistency in your project.
4. In your project, create the object "Device Proxy Data" below the required CPU by clicking the corresponding command "Add device proxy data" in the "Device Proxy Data" folder.
   Result: The "Device Proxy Data" object has been created.
5. Select the required device proxy data in the project tree and double-click the object to open it in the editor.

6. Enter the required data for the device proxy data in the editor under "General" or apply the default setting. You can change the name and enter a comment for export of the controller data.
   In the "Define content" area of the editor select which of the existing controller data you wish to export as IPE file and click "Export device proxy data".
7. In the subsequent dialog, enter the name and storage location for the IPE file to be created and click "Save". You will be notified as soon as the export has been completed successfully.
   Result: The selected controller data are exported as IPE file.
8. In your target project, create a device proxy to be able to import the controller data contained in the IPE file. To do so, click the command "Add new device" and select the required device proxy in the dialog that opens.
9. Click the newly created device proxy and select the "Initialize device proxy" command from the shortcut menu.
10. Select the previously created IPE file to be used for the initialization, and confirm the selection in the following dialog with "OK". This starts the initialization.
    Result: The controller data from the source project contained in the IPE file is transferred to the device proxy in the target project.

After the successful exchange of controller data, you can continue with the configuration in your target project and, for example, connect PLC tags with HMI tags. The controller data can be updated as often as required when it is changed in the source project.

Procedure for exchanging controller data via a project file

When exchanging controller data via a project file, device proxy data can either be predefined in the source project or it can be defined and selected from the target project.

Follow these steps to exchange data via a project file:

1. Using TIA Portal V13 or higher, create a project with all required controller data and an executable user program.
2. Configure the hardware required for the project.
3. Compile the project to ensure consistency in your project.
4. Open the TIA Portal project from which you want to import the controller data into the target project.
5. In your project, create the object "Device Proxy Data" below the required CPU by clicking the corresponding command "Add device proxy data" in the "Device Proxy Data" folder.
   Result: The "Device Proxy Data" object has been created.
6. Enter the required data for the device proxy data in the editor under "General" or apply the default setting. You can change the name and enter a comment for the device proxy data.
7. In the "Define content" area of the editor select which of the existing controller data you want to provide by means of the device proxy data.
8. Click "Save" to save the changes in the project.
   Result: You have successfully saved the selected controller data in your device proxy data.

9. In your target project, create a device proxy to import the controller data contained in the project file. To do so, click the command "Add new device" and select the required device proxy in the dialog that opens.
10. Click the newly created device proxy and select the "Initialize device proxy" command from the shortcut menu.
11. Select the previously created project file with the prepared device proxy data to be used for the initialization, and confirm the selection in the following dialog with "OK". This starts the initialization.
    Result: The controller data from the source project contained in the project file is transferred to the device proxy in the target project.

After the successful exchange of controller data, you can continue with the configuration in your target project and, for example, connect PLC tags with HMI tags. The controller data can be updated as often as required when it is changed in the source project.

Procedure for updating already transferred controller data

System requirements for the update:

The following requirements must be met for updating already transferred controller data with the help of the already created "Device Proxy Data" object:

- You are using the same project for the update as for the previous transfer of the controller data.
- You are using the already created "Device Proxy Data" for transfer of the controller data.
- You have not made any changes to the hardware configuration or the communication interfaces in the meantime. You can add new hardware components.

Updating procedure:

1. Open the TIA Portal project from which you want to update the controller data in the target project.
2. Double-click the existing "Device Proxy Data" object and select under content which controller data you want to make available for the update.
3. Save the project or export the IPE file once again.
4. Open the required target project in the TIA Portal.
5. Click the existing device proxy in the target project and select the "Update device proxy" command from the shortcut menu.
6. Select the source which is to be used for the update, and confirm the selection.
7. In the following dialog, select under "Device" the CPU that was previously used as source, from which the controller data is to be re-imported.
8. Under "Defined device proxy data", select the required object and exit the dialog with "OK".
9. The update is then started and the selected device proxy data is transferred from the source project to the device proxy in the target project.

After the successful update of the controller data, you can continue with the configuration in your target object and, for example, connect PLC tags with HMI tags. The controller data can be updated as often as required when it is changed in the source project.

See also

Basics of Inter Project Engineering (IPE) (Page 10842)

Requirements for Inter Project Engineering (IPE) (Page 10844)

Creating device proxy data (Page 3079)

Opening device proxy data (Page 3080)

Exchange of controller data via IPE file (Page 3080)

Using online and diagnostics functions 13

13.1 Displaying accessible devices

Accessible devices

Accessible devices are all devices connected to an interface of the programming device / PC and that are turned on. Devices that allow only restricted configuration using the currently installed products or that cannot be configured at all can also be displayed.

Displaying accessible devices on an interface of the programming device / PC in the project tree

To display accessible devices on a single interface of the programming device / PC, follow these steps:

1. Open the "Online access" folder in the project tree.
2. Click on the arrow to the left of the interface to show all the objects arranged below the interface.
3. Double-click on the "Update accessible devices" command below the interface.
   All devices that are accessible over this interface are displayed in the project tree. When there is a large number of connected devices, the update process may take some time. You can see the progress of the update in the status bar. If you have found the desired device before the update is completed, you can cancel the update of accessible devices. To do this, click on the cross to the right of the progress bar.

Displaying accessible devices in a list

To display the accessible devices on all available interfaces in an overview list, follow these steps:

1. Select the "Accessible devices" command in the "Online" menu.
   The "Accessible devices" dialog is displayed.
2. Select the type of interface from the "Type of the PG/PC interface" drop-down list. The "PG/PC interface" drop-down list then shows only the interfaces of the programming device / PC that match the selected interface type.
3. Select the required interface of the programming device / PC from the "PG/PC interface" drop-down list, for example an Industrial Ethernet adapter.
   If no devices are available on an interface, an unbroken connecting line is displayed between the programming device / PC and the device. If devices are accessible, an unbroken connecting line is shown and the devices accessible on the selected interface of the programming device / PC are displayed in a list.

4. If you have connected a new device in the meantime, click the "Refresh" button to refresh the list of accessible devices.
5. To go to a device in the project tree, select the device from the list of accessible devices and click the "Show" button.
   The interface to which the selected device is connected is shown as selected in the project tree.

Displaying additional information about the accessible devices in the project tree

To display additional information on the accessible devices in the project tree, follow these steps:

1. Click on the arrow to the left of one of the accessible devices in the project tree.
   All data available online, for example blocks and system data, is displayed for known devices. Objects that you cannot edit directly at this point are grayed out. If additional editing options are available for a device, for example, downloading using the shortcut menu, the device is shown in black text.

See also

Changing the device configuration online (Page 10850)

13.2 Changing the device configuration online

You can assign parameters to some devices, preferably in small hardware setups, directly online. You do not need to create a project or have offline data to do this. In this way, you can quickly and easily change the device configuration. You do not need to compile the hardware configuration or perform a download. Depending on the device, either all changes become active immediately or they are written to the device only after confirmation.

Requirement

- The device must support online parameter assignment. You can learn whether or not your specific devices support this function in the device manual.
- The device must be connected to the PG/PC and available in the list of accessible devices.

Procedure

To change the device configuration online, follow these steps:

1. Show the accessible devices on the interface via which the device is connected. To learn how to show the accessible devices, see the previous chapter "Showing accessible devices (Page 10849)".
2. Expand the device to display the lower-level elements.
3. Double-click the "Parametrize device" item.
   A configuration page for the device opens in the work area.

4. Make all required settings.
   With some devices, the new settings take effect immediately.
5. Optionally, depending on the device: Click on the "Upload to device" button.
   The settings are transferred to the device.

13.3 Connecting devices online

13.3.1 General information about online mode

Online mode

In online mode, there is an online connection between your programming device / PC and one or more devices.

An online connection between the programming device/PC and the device is required, for example, for the following tasks:

- Testing user programs
- Displaying and changing the operating mode of the CPU
- Displaying and setting the date and time of day of the CPU
- Displaying module information
- Comparing blocks
- Hardware diagnostics

Before you can establish an online connection, the programming device/PC and the device must be physically or remotely connected. As an alternative, some devices support a simulation mode. In this case, a connection to the device is simulated via the PLCSIM virtual interface.

After establishing a connection, you can use the Online and Diagnostics view or the "Online tools" task card to access the data on the device. The current online status of a device is indicated by an icon to the right of the device in the project tree. You will find the meaning of the individual status icons in the relevant tooltip.

Note

Some online functions depend on the scope of the installed software or whether a project is open.

Standby or hibernation of the programming device / PC

If the programming device / PC is changed to the standby or hibernation mode when there is an online connection, all online connections are terminated. When the programming device / PC wakes up from hibernation, the online connections are not automatically re-established.

Note that suddenly terminating an online connection can lead to loss of data or a connected device may interrupt program execution.

Performing an LED flash test

In many online dialogs you can perform an LED flash test, if the device connected online supports this feature. If you select the "Flash LED" check box, an LED flashes on the currently selected device. This feature is useful, for example, when you are not sure which device in the hardware configuration corresponds to the station currently selected in the software.

Read any additional information and learn about the possible limitations to the LED flash test in the respective device documentation.

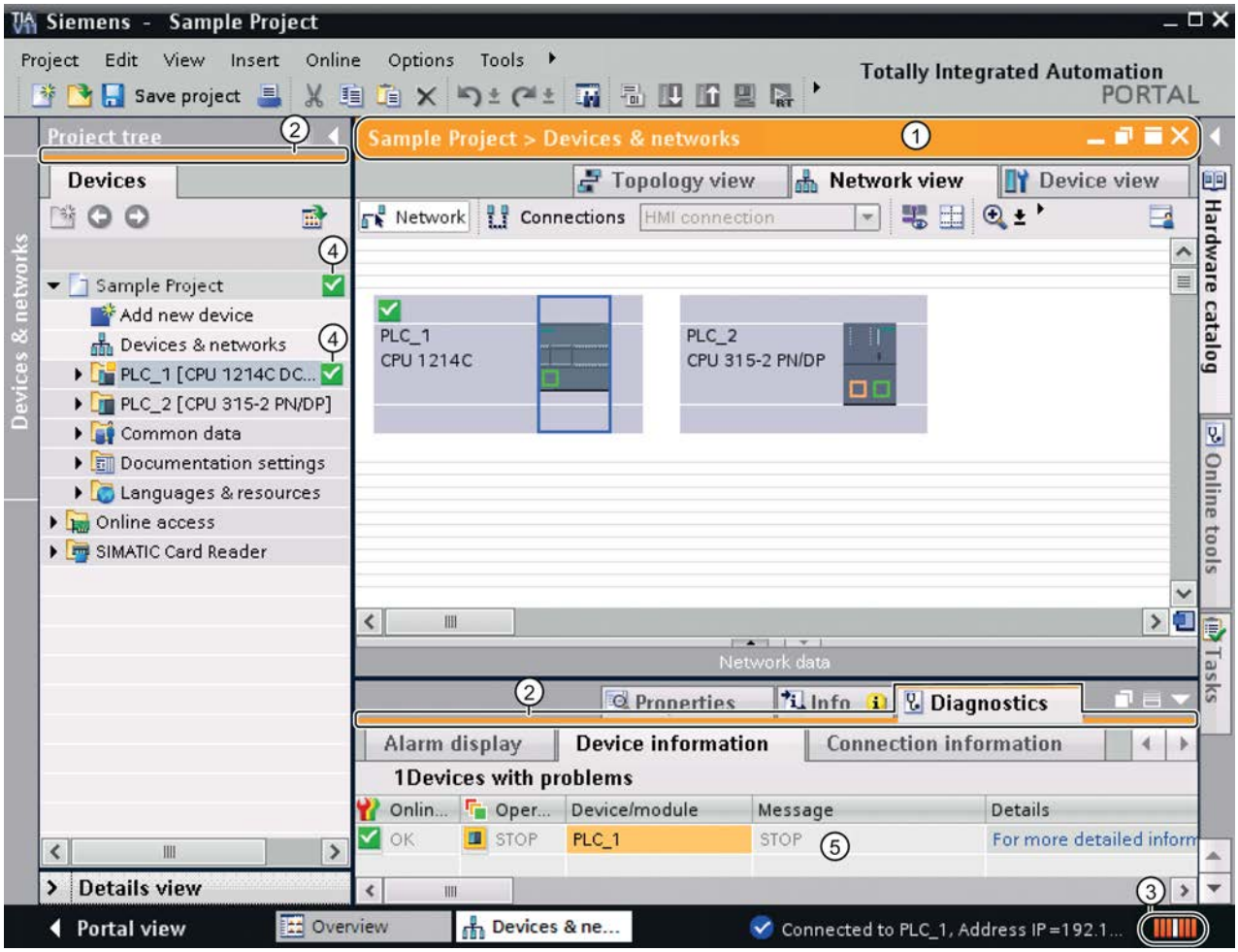
See also

View in online mode (Page 10853)

13.3.2 View in online mode

Online displays

After the online connection has been established successfully, the user interface changes. If a device is unavailable, this is indicated by a symbol. The following figure shows a device connected online and the corresponding user interface:



① The title bar of the active window gets an orange background as soon as at least one of the devices currently displayed in the editor has been successfully connected online. If one or more devices are unavailable, a symbol for a broken connection appears in the title bar of the editor.

② The title bars of inactive windows for the relevant station now have an orange line below them.

③ An orange, pulsing bar appears at the right-hand edge of the status bar. If the connection has been established but is functioning incorrectly, an icon for an interrupted connection is displayed instead of the bar. You will find more information on the error in "Diagnostics" in the Inspector window.

④ Operating mode symbols or diagnostics symbols for the stations connected online and their underlying objects are shown in the project tree. A comparison of the online and offline status is also made automatically. Differences between online and offline objects are also displayed in the form of symbols.

⑤ The "Diagnostics > Device information" area is brought to the foreground in the Inspector window.

Online connection abort

The online mode and its display are retained as long as at least one device is connected online. If the online connection to one or more devices aborts, the TIA Portal remains in online mode. The display of the TIA Portal changes to offline mode only when there is no longer an online connection to any device.

See also

General information about online mode (Page 10851)

Basics of project data comparison (Page 363)

13.3.3 Establishing and canceling an online connection

Requirement

At least one PG/PC interface is installed and is physically connected to a device, for example with an Ethernet cable. As an alternative, it is also possible to establish a virtual connection using PLCSIM.

Go online

To establish an online connection, follow these steps:

1. In the project tree, select one or mote devices to which you want an online connection to be established.
2. Select the "Go online" command in the "Online" menu.
   If the device was already connected to a specific PG/PC interface, the online connection is automatically established to the previous PG/PC interface. In this case, you can ignore the following steps. If there was no previous connection, the "Go online" dialog opens.
3. Select the type of interface from the "Type of the PG/PC interface" drop-down list. The "PG/PC interface" drop-down list then shows only the interfaces of the programming device / PC that match the selected interface type.
4. Select the required interface of the programming device / PC from the "PG/PC interface" drop-down list, for example an Industrial Ethernet adapter.

5. In the "Connection to subnet" drop-down list, select the interface via which the device is connected to the programming device or PC. In this case, a direct connection is established to the device, without a network node, for example, an interposed switch. Alternatively, select the appropriate subnet for the connection to the programming device or PC if the device can be accessed via a network node. If you do not know how the device is connected to the programming device or PC, choose the "Try all interfaces" entry.
   If you selected an MPI or PROFIBUS subnet, the bus parameters configured in the programming device/PC interface are applied at this point.
6. If the device is accessible via a gateway, select the gateway that connects the two subnets involved in the "1st gateway" drop-down list.
   If no devices are available on the interface, a broken connecting line is displayed between the programming device / PC and the device. If devices are accessible, an unbroken connecting line is shown and the devices accessible on the selected interface of the programming device/PC are displayed in the "Compatible devices in target subnet" list.
7. Optional: Click the "Update" button to update the "Compatible devices in target subnet" list.
8. Optional: Select the "Flash LED" check box to the left of the graphic to run an LED flash test. With this function, you can check that you have selected the correct device. The LED flash test is not supported by all devices.
9. Select your device in the "Compatible devices in target subnet" table, and confirm your selection with "Go online".
   The online connection to the selected target device is established.

Result

After the online connection has been established, the title bars of the editors change to orange. An orange activity bar is also shown in the title bar of an editor and in the status bar. In the project tree, status symbols show the difference between online and offline objects.

The connection path is stored for future connection attempts. It is no longer necessary to open the "Go online" dialog unless you want to select a new connection path.

Note

If no accessible device is displayed, select a different network access for the PG/PC interface or check the settings of the interface.

Canceling an online connection

To disconnect the existing online connection, follow these steps:

1. Select the device you want to disconnect from in the project tree.
2. Select the "Go offline" command in the "Online" menu.

See also

Connecting online with several devices (Page 10856)

View in online mode (Page 10853)

Assigning a temporary IP address (Page 10868)

Influence of user rights (Page 318)

13.3.4 Connecting online with several devices

You can establish an online connection to several devices at the same time without needing to select individual devices previously in the network view.

Requirement

- No device must be selected
- At least one PG/PC interface is installed and is physically connected to a device, for example with an Ethernet cable. As an alternative, it is also possible to establish a virtual online connection using PLCSIM or a remote connection.

Procedure

To establish an online connection to several devices at the same time, follow these steps:

1. Select the project in the project tree.
2. Select the "Go online" command in the "Online" menu.
   The "Select device for opening the online connection" dialog opens with a table of all available devices.
3. Select the devices to which you want to establish an online connection in the "Go online" column.
4. Click the "Go online" button.

Result

Without any further prompt for confirmation, a connection is established to all selected devices if a connection was already established to the selected devices at least once. If there was no previous online connection, the "Go online" dialog opens. In this case, first configure the online connection as described in the section "Go online and Go offline (Page 10854)".

See also

Establishing and canceling an online connection (Page 10854)

Assigning a temporary IP address (Page 10868)

13.3.5 Disconnecting online connections of multiple devices

You can disconnect the online connections to multiple devices at one time without needing to select individual devices beforehand in the network view.

Requirement

- No device is selected.
- There is currently an online connection to at least one device.

Procedure

To terminate the online connections to multiple devices at one time, follow these steps:

1. Select the "Go offline" command in the "Online" menu.
   The "Select devices" dialog opens with a table of all available devices.
2. Select the device for which you want to terminate the online connection in the "Go offline" column.
3. Click the "Go offline" button.

Result

The online connection to the all the selected devices is terminated.

13.4 Backing up the software and hardware configuration of a device

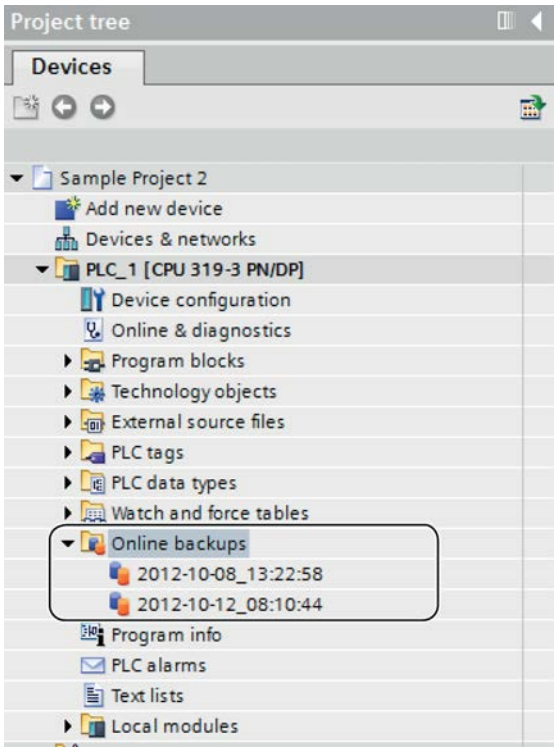
13.4.1 Creating a backup of a device

Backing up the software and hardware configuration of an S7-300/400 CPU

If you have already downloaded a configuration to an S7-300/400 CPU, it is advisable to make a backup. You may have modified the configuration and want to test the new configuration. Before you download the new configuration to the CPU, you can create a backup of the current device state and then restore the current configuration at a later date. The backup is performed with the current values of the CPU. In the case of S7-400 CPUs with fail-safe function, the initial values are backed up.

You can create as many backups as you want and store a variety of configurations for a CPU. The backups are named with the name of the CPU and the time and date of the backup. You can find the backup in the project tree under the CPU in the "Online backups" folder.

The following figure shows an S7-319 CPU for which two backups were created:



See also

Restoring the software and hardware configuration of a device (Page 10859)

Backing up a device configuration (Page 10858)

General information on loading (Page 355)

13.4.2 Backing up a device configuration

You can back up the configuration of a S7-300/400 CPU in the TIA Portal. So you can download and test a new configuration to a device without any risk. If needed, you can restore the initial configuration of the CPU.

Requirement

- The CPU must already be created in the project.
- It must be an S7-300/400 CPU.
- The CPU must be online. If there is no online connection, an online connection is established during the backup.

Procedure

To create a backup of the current configuration of a CPU, follow these steps:

1. Select the CPU in the project tree.
2. Select the "Backup from online device" command in the "Online" menu.

Result:

A backup of the entire hardware configuration and software is created. The backup is stored in the project tree in the "Name of the CPU > Online backups" folder. The backup is assigned the name of the CPU with the time and date of the backup. You can rename the backup, but you cannot make any changes to the contents of the backup.

See also

Establishing and canceling an online connection (Page 10854)

Restoring the software and hardware configuration of a device (Page 10859)

Creating a backup of a device (Page 10857)

13.4.3 Restoring the software and hardware configuration of a device

If you have backed up the configuration of a device at an earlier point in time, you can transfer the backup back to the device. The saved configuration is then restored on the device.

Requirement

You must have previously configure the device and stored a backup of the device in the project.

Procedure

To restore older software and hardware state on a device, follow these steps:

1. Open up the folder of the device in the project tree to display the lower-level objects.
2. Open the "Online backups" folder.
3. Select the backup you want to restore.
4. In the "Online" menu, select the "Download to device" command.
   - If you had previously established an online connection, the "Load preview" dialog opens. This dialog displays messages and proposes actions necessary for downloading.
   - If you had not previously established an online connection, the "Extended download to device" dialog opens, and you must first select the interfaces via which you want to establish the online connection to the device.
     See also: Establishing and terminating an online connection (Page 10854)

5. Check the messages in the "Load preview" dialog, and select the actions in the "Action" column, if necessary.

   Note

   Performing the proposed actions while the plant is in operation can cause serious bodily injury and property damage in the event of malfunctions or program errors.

6. As soon as loading becomes possible, the "Load" button is enabled.
7. Click the "Load" button.
   The backup is transferred to the device and device is restored. The "Load results" dialog then opens. In this dialog, you can check whether or not the loading operation was successful and take any further action that may be necessary.
8. Click the "Finish" button.

See also

Creating a backup of a device (Page 10857)

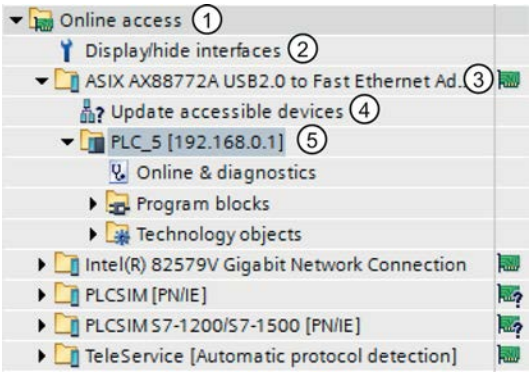
13.5 Configuring the PG/PC interface

13.5.1 Online access

Online access of the project

In the "Online access" folder of the project tree, you will find all active interfaces of your programming device/PC. Each interface icon provides you with information on the status of the interface. You can also display the accessible devices and display and edit the properties of an interface using the shortcut menu.

The following figure shows the "Online access" folder in the project tree:



① "Online access" folder in the project tree
All interfaces installed in the programming device/PC are displayed in the "Online access" folder.

② Show/hide interfaces
Use the function "Show/hide interfaces" to show or hide individual interfaces.

③ Status display for the interfaces
The current status of an interface is indicated by an icon to the right of the name. You can see the meaning of the icon in the tooltip.

④ Updating the list of accessible devices.
This function is available for each hardware interface of the programming device/PC. Software interfaces, such as a remote connection, do not offer this function.

⑤ Devices connected via the respective interface with the programming device/PC
The type of the respective device and its status are displayed by the preceding icon.

Displaying or updating accessible devices

You have the following options if you want to display all devices accessible online on your programming device/PC:

- Display of the accessible devices on a single interface of the programming device / PC in the project tree. In the project tree, you can also display additional information about the individual accessible devices.
- Display of the accessible devices of all interfaces in a list.

See also: Displaying accessible devices

Overview of icons for accessible devices

The accessible devices are identified with an icon according to their type and status. The following is an overview of all icons and their meaning.

	Icon for unidentified modules This icon is displayed whenever the identification of a module is not yet complete or when the identification of a module was not successful, for example, because the required online data could not be read.
	Icon for the following device types: • PLCs • SIMOCODE pro devices • IE/PB links • CPs of PC systems • SCALANCE head modules • S7-300 and S7-400 CPs • PROFINET IO devices and PROFINET CPs • SCALANCE modules and gateways that could not be identified
	PROFINET IO devices, encoders, switchgear, sensors and identification systems that were replaced by similar devices because these could not be identified
	Icon for the following device types: • HMI devices • PROFINET IO devices of the HMI type if these could not be identified and were therefore replaced by a similar device
	PROFINET IO devices of the drive type that could not be identified and were therefore replaced by a similar device
	PROFINET IO devices of the type development kit and network components that could not be identified and were therefore replaced by a similar device
	PROFINET IO devices of the type TeleService adapter that could not be identified and were therefore replaced by a similar device

See also

Displaying and modifying interface properties (Page 10863)

13.5.2 Basics of assigning parameters for the PG/PC interface

Options for connecting to target systems

If the devices of the project are connected via different subnets, you assign a suitable network access to each PG/PC interface to be able to establish online connections to the target systems. The following interfaces are automatically supported:

- MPI
- PROFIBUS
- Industrial Ethernet (ISO and TCP/IP)

You can make various settings for the interfaces. The following sections explain the parameter settings you can make.

Note

Note that changes to interface parameters have a direct influence on the operating system and the programming device / PC. Remember that some parameter settings can only be changed if you have adequate user rights.

See also

Setting parameters for the Industrial Ethernet interface (Page 10865)

Setting parameters for the MPI and PROFIBUS interfaces (Page 10869)

13.5.3 Showing or hiding interfaces

To improve clarity, it is possible to hide individual interfaces of the PG/PC in the project tree and to show them again when necessary.

Procedure

To show or hide individual interfaces, follow the steps below:

1. Open the "Online access" folder in the project tree.
2. Double click on the "Display/hide interfaces" icon.
   The "Display/hide interfaces" dialog opens.
3. Enable or disable the required interfaces in the "Display in the project tree" column.
4. Click the "Apply" button.
   The changes are applied. The view of the interfaces in the "Online access" folder is refreshed.

13.5.4 Displaying and modifying interface properties

Introduction

For each interface, you can display and, in some cases, modify properties, for example the network type, address, and status.

Procedure

To open the properties, follow these steps:

1. Right-click on the required interface below "Online access" in the project tree.
2. Select the "Properties" command from the shortcut menu.
   A dialog containing the properties of the interface opens. On the left of the dialog, you will see the area navigation. You can view the current parameter settings in the individual entries in the area navigation and, if necessary, change them.

13.5.5 Adding interfaces

You have the option of installing additional interfaces after installation of the TIA Portal.

Procedure

To install an interface at a later time and add it to the TIA Portal, follow these steps:

1. Install or update the drivers in the operating system once you have installed the interface hardware.
2. Close the TIA Portal if it is still open.
3. Open the Windows control panel.
4. Open the entry "Setting the PG/PC Interface" in the Control Panel.
   The "Setting the PG/PC Interface" dialog opens.
5. Make any necessary changes to the interface configuration and confirm them with "OK".
   You have to click "OK", even if you have not made any changes.
6. Restart the TIA Portal.

Result

The newly installed interface is now displayed in the project tree under the "Online access" folder.

13.5.6 Setting parameters for the Ethernet interface

13.5.6.1 Setting parameters for the Industrial Ethernet interface

Options in the parameter settings for the Industrial Ethernet interface

When setting parameters for the Industrial Ethernet interface, you have the following options:

- Parameters dependent on the operating system
  The Industrial Ethernet interface has parameters that are set in the operating system and are valid for all connected devices. These parameter settings are only displayed here, they can, however, be changed in the network settings of the operating system.
- Parameters that can be set in the software

Note

Note that changes to interface parameters have a direct influence on the operating system and the programming device / PC. Remember that some parameter settings can only be changed if you have adequate user rights.

Parameters for the Industrial Ethernet interface

The following table contains an overview of the parameters of the Industrial Ethernet interface that are set by the operating system and can be changed by the user.

Parameter settings that cannot be changed	Parameters that can be set
MAC address	Fast acknowledge at the IE-PG access and for TCP/IP
DHCP server activated/deactivated	Timeout at the IE-PG access and for TCP/IP
APIPA activated/deactivated	LLDP
IP address	Additional, dynamic IP addresses for the network adapter
Subnet mask	-
DNS addresses	-
DHCP addresses	-

See also

Basics of assigning parameters for the PG/PC interface (Page 10862)

Displaying operating system parameters (Page 10866)

Connecting the PG/PC interface to a subnet (Page 10866)

Setting parameters for the Ethernet interface (Page 10867)

Assigning a temporary IP address (Page 10868)

Managing temporary IP addresses (Page 10868)

Influence of user rights (Page 318)

13.5.6.2 Displaying operating system parameters

The Ethernet interface is part of the operating system. All parameters of the network adapter can therefore be adapted in the network settings of the operating system.

You can display the following parameters in the software:

- Physical address of the network adapter
- Assignment of the IP address by a DHCP server activated or deactivated
- Assignment of a private IP address by the operating system activated or deactivated
- Current static IP address
- Assigned subnet mask
- DNS addresses
- DHCP addresses

If you want to modify the parameter settings, please refer to the documentation of the operating system or the network adapter.

Displaying current parameters of the Ethernet interface

To display the current parameters of the Ethernet interface, follow these steps:

1. Select the Ethernet interface in the project tree in "Online access".
2. Select the "Properties" command in the shortcut menu of the interface.
   The dialog for configuring the interface opens.
3. Select "Configurations > Industrial Ethernet" in the area navigation.

See also

Setting parameters for the Ethernet interface (Page 10867)

13.5.6.3 Connecting the PG/PC interface to a subnet

If you have created several subnets, you can specify the subnet to which the Ethernet interface is connected.

Procedure

To select the subnet to which the Ethernet interface is connected, follow these steps:

1. Select the Ethernet interface in the project tree in "Online access".
2. Select the "Properties" command in the shortcut menu of the interface.
   The dialog for configuring the interface opens.

3. Go to "General > Assignment" and select the subnet to which you want to connect the Ethernet interface of the programming device / PC in the "Connection to subnet" drop-down list.
4. Close the dialog with "OK".

13.5.6.4 Setting parameters for the Ethernet interface

You can adapt some parameter settings relating to the network protocol directly in the software.

Requirement

You must have adequate user rights.

See also: Influence of user rights (Page 318).

Procedure

To change parameter settings relating to the network protocol, follow these steps:

1. Select the Ethernet interface in the project tree in "Online access".
2. Select the "Properties" command in the shortcut menu of the interface.
   The dialog for configuring the interface opens.
3. Select "Configurations > IE-PG access" to adapt the protocol settings relevant to network management.
   - Select the "Fast acknowledge" check box to achieve faster reaction times with smaller network packets.
   - From the "Timeout" drop-down list, select the maximum time that can elapse before a network node is detected.
4. To activate the LLDP protocol and discover the network topology more accurately, set the "LLDP active" check box in "Configurations > LLDP".
5. Select "Configurations > TCP/IP" to adapt the TCP/IP protocol for network traffic during runtime.
   - Select the "Fast acknowledge" check box to achieve faster reaction times with smaller network packets.
   - From the "Timeout" drop-down list, select the maximum time that can elapse before there is a timeout during communication with a network node.

See also

Influence of user rights (Page 318)

Displaying operating system parameters (Page 10866)

13.5.6.5 Assigning a temporary IP address

Adding a dynamic IP address

If the IP address of a device is located in a different subnet from the IP address of the network adapter, you will first need to assign an additional IP address with the same subnet address as the device. Only then is communication between the device and the programming device / PC possible.

The assignment of an additional temporary IP address is also proposed automatically if you want to perform an online action and the current IP address of the programming device/PC is not yet in the correct subnet.

A temporarily assigned IP address remains valid until the next time the programming device/ PC is restarted or until you delete it manually.

Note

You require adequate permissions to be able to assign a temporary IP address.

See also: Influence of user rights (Page 318)

See also

Managing temporary IP addresses (Page 10868)

13.5.6.6 Managing temporary IP addresses

If the IP address of a device is located in a different subnet from the current static IP address of the network adapter, the network adapter temporarily assigns a suitable IP address from the subnet of the device.

You can display all temporarily assigned addresses and delete them. Note that IP addresses that you manually assigned in the operating system are not displayed in the TIA Portal.

Requirement

To delete, you require adequate permissions.

Procedure

To display and delete temporarily assigned addresses, follow these steps:

1. Select the Ethernet interface in the project tree in "Online access".
2. Select the "Properties" command in the shortcut menu of the interface.
   The dialog for configuring the interface opens.

3. Select "Configurations > IE-PG access".
   A table with the assigned IP addresses is displayed.
4. Click the "Delete project-specific IP addresses" button to delete all the IP addresses at one time.

See also

Influence of user rights (Page 318)

13.5.6.7 Resetting the TCP/IP configuration

If you have changed the TCP/IP protocol settings, you can reset them to the defaults.

Procedure

To restore the TCP/IP configuration to the default settings, follow these steps:

1. Select the Ethernet interface in the project tree in "Online access".
2. Select the "Properties" command in the shortcut menu of the interface.
   The dialog for configuring the interface opens.
3. Select "Configurations > TCP/IP".
4. Click the "Standard" button to reset all the settings.

13.5.7 Setting parameters for the MPI and PROFIBUS interfaces

13.5.7.1 Setting parameters for the MPI and PROFIBUS interfaces

Possible parameter settings for the MPI and PROFIBUS interfaces

The following parameter settings can be made for the MPI and PROFIBUS interfaces:

- Automatic configuration: You can use automatic detection functions to find out whether a device is connected to the PG/PC interface over PROFIBUS or MPI.
- Selecting a default configuration for PROFIBUS or MPI that can be adapted later.

Device- and network-related settings for MPI and PROFIBUS

You can set device- and network-related parameters for MPI and PROFIBUS interfaces. Device-related parameters are local settings for the interface. Network-related parameters, on the other hand, must match up on all devices.

MPI interface parameters you can modify

You can adapt the following default parameters for the MPI interface:

Device-related parameters	Network-related parameters
Is the only master	Highest address
Own address	Transmission rate
Timeout	

PROFIBUS interface parameters you can modify

You can adapt the following default parameters for the PROFIBUS interface:

Device-related parameters	Network-related parameters
Is the only master	Highest address
Own address	Transmission rate
Timeout	Profile
	Bus parameters
	Number of masters on bus
	Number of slaves on bus

See also

Basics of assigning parameters for the PG/PC interface (Page 10862)

13.5.7.2 Setting MPI or PROFIBUS interface parameters automatically

Setting up automatic bus parameter detection

If you select an interface with automatic detection of the bus parameters (for example CP 5611 (Auto)), you can connect the programming device or PC to MPI or PROFIBUS without needing to set bus parameters. At a transmission speed lower than 187.5 Kbps, you may, however, have waiting times of up to one minute.

Requirement

- Masters that distribute bus parameters cyclically are connected to the bus.
- In PROFIBUS networks, the cyclic distribution of the bus parameters must be enabled.

Procedure

To enable automatic bus parameter detection, follow these steps:

1. Select the interface in the project tree.
2. Select the "Properties" command in the shortcut menu of the interface.
   The dialog for configuring the interface opens.
3. Go to "General > Configurations > Active configuration" and select the setting "Automatic protocol detection".
4. Go to "Configurations > Auto configuration > Local settings" and select the address of the PG/PC interface in the "Own address" drop-down list.
5. If you then want to display the current bus settings, click the "Network detection" button.

See also

Setting parameters for the MPI interface (Page 10871)

Setting parameters for the PROFIBUS interface (Page 10873)

13.5.7.3 Setting parameters for the MPI interface

Changing the parameter settings of the MPI interface

The network-related parameters and bus parameters for the MPI network can be adapted. You should first select a default setting and then adapt this to the specific situation.

Setting defaults for the MPI interface

To adapt the parameters of the MPI interface, follow these steps:

1. Select the interface in the project tree.
2. Select the "Properties" command in the shortcut menu of the interface.
   The dialog for configuring the interface opens.

3. Go to "General > Assignment" and select the subnet with which you want to connect the interface in the "Connection to subnet" drop-down list.
4. Under "General > Configuration", select a default for the device and network-related parameters. The defaults are suitable for most configurations. Select one of the following settings:
   - Automatic protocol detection
     You can connect the programming device to MPI or PROFIBUS without having to set bus parameters. At a transmission speed lower than 187.5 Kbps, you may, however, have waiting times of up to one minute. Prerequisite for the automatic detection is a connection to the bus master, which distributes the bus parameters cyclically. With PROFIBUS subnets, cyclic distribution of bus parameters may not be deactivated (default PROFIBUS network setting).
   - MPI
     The "MPI" transmission protocol is selected. Typical parameters are set that are adequate for most configurations. You can change the parameters to your needs, however.
   - PROFIBUS
     The "PROFIBUS" transmission protocol is selected. Typical parameters are set that are adequate for most configurations. You can change the parameters to your needs, however.

Changing the default parameter settings

To adapt the default settings to your requirements, change the parameter setting where necessary in "Configurations > MPI".

You can set the following device-related parameters:

- Is only master
  An additional verification function to prevent bus disruptions when connecting the PG/PC to the network is disabled because the programming device or PC is the only master on the bus.
  - Do not enable this option unless you have only connected slaves to your programming device or PC.
  - If the "Is only master" check box is enabled, it is not possible to identify the directly connected device in the "Accessible devices" window.
- Own address
  This setting relates to the programming device or PC on which you call up the parameter settings of the interface. Set the local device address of your programming device or PC here.
  - This address must be unique throughout the network.
  - The programming device or PC is addressed using this address in the MPI network.

- Check
  This enables an additional safety function to prevent bus disruptions when connecting the PG/PC to the network. The driver checks whether the local address is already being used by another station. Active as well as passive stations are taken into consideration in this case. The driver monitors this on the PROFIBUS. The connection of the PG/PC to the network will take longer with the automatic check. To use the check, the driver must support the function. Furthermore, the "Is only master" option must not be selected.
- Timeout
  Set a higher timeout value if, for example, you have problems with long response times on the network.

You can set the following network-related parameters:

- Highest address:
  Select the configured highest device address. Make sure that the same highest device address is set for all devices of a PROFIBUS or MPI network.
- Transfer rate:
  Here, you select the transmission speed to be used on the MPI network.

See also

Setting MPI or PROFIBUS interface parameters automatically (Page 10870)

13.5.7.4 Setting parameters for the PROFIBUS interface

Changing the parameter settings of the PROFIBUS interface

The network-related parameters and bus parameters for the PROFIBUS network can be adapted more precisely. You should first select a default setting and then adapt this to the specific situation.

Setting defaults for the PROFIBUS interface

To adapt the parameters of the PROFIBUS interface, follow these steps:

1. Select the interface in the project tree.
2. Select the "Properties" command in the shortcut menu of the interface.
   The dialog for configuring the interface opens.

3. Go to "General > Assignment" and select the subnet with which you want to connect the interface in the "Connection to subnet" drop-down list.
4. Under "General > Configuration", select a default for the device and network-related parameters. The defaults are suitable for most configurations. Select one of the following settings:
   - Automatic protocol detection
     You can connect the programming device to MPI or PROFIBUS without having to set bus parameters. At a transmission speed lower than 187.5 Kbps, you may, however, have waiting times of up to one minute. Prerequisite for the automatic detection is a connection to the bus master, which distributes the bus parameters cyclically. With PROFIBUS subnets, cyclic distribution of bus parameters may not be deactivated (default PROFIBUS network setting).
   - MPI
     The "MPI" transmission protocol is selected. Typical parameters are set that are adequate for most configurations. You can change the parameters to your needs, however.
   - PROFIBUS
     The "PROFIBUS" transmission protocol is selected. Typical parameters are set that are adequate for most configurations. You can change the parameters to your needs, however.

Changing the default parameter settings

To adapt the default settings to your requirements, change the parameter setting where necessary in "Configurations > PROFIBUS".

You can set the following device-related parameters:

- Is only master
  An additional verification function to prevent bus disruptions when connecting the PG/PC to the network is disabled because the programming device or PC is the only master on the bus.
  - Do not enable this option unless you have only connected slaves to your programming device or PC.
  - If the "Is only master" check box is enabled, it is not possible to identify the directly connected device in the "Accessible devices" window.
- Own address
  This setting relates to the programming device or PC on which you call up the parameter settings of the interface. Set the local device address of your programming device or PC here.
  - This address must be unique throughout the network.
  - The programming device or PC is addressed using this address in the PROFIBUS network.

- Check
  This enables an additional safety function to prevent bus disruptions when connecting the PG/PC to the network. The driver checks whether the local address is already being used by another station. Active as well as passive stations are taken into consideration in this case. The driver monitors this on the PROFIBUS. The connection of the PG/PC to the network will take longer with the automatic check. To use the check, the driver must support the function. Furthermore, the "Is only master" option must not be selected.
- Timeout
  Set a higher timeout value if, for example, you have problems with long response times on the network.

You can set the following network-related parameters:

- Highest address:
  Select the configured highest device address. Make sure that the same highest station address is set for all devices of a PROFIBUS network.
- Transfer rate:
  Here, you select the transmission speed to be used on the PROFIBUS network.
- Profile:
  You have a choice of four alternatives for the PROFIBUS settings. "DP", "Standard" and "Universal (DP/FMS)" are predefined settings that you cannot change. If you select "User-defined", you can adapt the bus parameters yourself.
  - If you have selected "User-defined", go to "Configurations > PROFIBUS > Bus parameters" in area navigation.
  - If you have selected one of the defaults (DP, Standard or Universal (DP/FMS)), you should select the "Include" check box in "Configurations > PROFIBUS > Bus parameters > Additional parameters". You can then set the number of masters and slaves on the bus. This allows a more precise calculation of the bus parameters and potential bus disruptions can be prevented. The option cannot be selected with a user-defined profile.

See also

Overview of the bus parameters for PROFIBUS (Page 10875)

Setting MPI or PROFIBUS interface parameters automatically (Page 10870)

13.5.7.5 Overview of the bus parameters for PROFIBUS

Introduction

The PROFIBUS subnet will only function problem-free if the parameters for the bus profile are matched to one another. You should therefore only change the default values if you are familiar with how to configure the bus profile for PROFIBUS.

It may be possible for the bus parameters to be adjusted depending on the bus profile. The offline values of the bus parameters are always shown even if you are online and linked to the target system.

The displayed parameters are valid for the entire PROFIBUS subnet.

Meaning of the individual parameters

- Tslot: Wait-to-receive time (slot time)
  The wait-to-receive time (slot time) defines the maximum time the sender will wait to receive a response from the addressed partner.
- Max. Tsdr: Maximum protocol processing time (max. station delay responder)
  The maximum protocol processing time defines the time after which the responding device must have processed the protocol.
- Min. Tsdr: Minimum protocol processing time (min. station delay responder)
  The minimum protocol processing time specifies the minimum time required by the responding device to process the protocol.
- Tset: Trigger time (setup time)
  The trigger time is the time that may lapse between the reception of a data frame frame and the reaction to it.
- Tqui: Quiet time for modulator
  The quiet time for modulator specifies the time required to change from sending to receiving.
- GAP factor: GAP update factor (GAP factor)
  The GAP factor specifies the number of token rotations before a new device is included in the token ring.
- Retry limit: Maximum number of repeated call attempts (retry limit)
  This parameter defines the maximum number of attempts made to reach a device.
- Trdy: Ready time
  The ready time is the time for an acknowledgment or response.
- Tid1: Idle time 1
  Idle time 1 specifies the delay time after receiving a response.
- Tid2: Idle time 2
  Idle time 2 specifies the delay time after sending a call without a response.
- Ttr: Target rotation time
  The target rotation time is the maximum time made available for a token rotation. During this time, all active devices (masters) receive the token once. The difference between the desired token round-trip time and the actual token round-trip time decides how much time is left for masters to send data frames to the slaves.
  As the minimum target rotation time (Ttr), select a value = 5000 times the HSA (Highest Station Address).
- Watchdog: Watchdog
  The watchdog time specifies the time after which a device must be addressed.
  As the minimum watchdog time, select a value = 6250 times the HSA.

Note

If you want to create a user-defined bus profile, please note that the minimum target rotation time (Ttr) should be 5000 times the HSA (highest PROFIBUS address). The minimum watchdog time should also be 6250 times the HSA.

See also

Setting parameters for the PROFIBUS interface (Page 10873)

13.5.7.6 Resetting the MPI or PROFIBUS configuration

If you have changed the MPI or PROFIBUS protocol settings, you can reset them to the defaults.

Procedure

To restore the MPI or PROFIBUS configuration to the default settings, follow these steps:

1. Select the MPI/PROFIBUS interface in the project tree in "Online access".
2. Select the "Properties" command in the shortcut menu of the interface.
   The dialog for configuring the interface opens.
3. Select "Configurations > MPI" or "Configurations > PROFIBUS", depending on the interface properties you want to reset.
4. Click the "Standard" button to reset all the settings.

13.6 Using the trace and logic analyzer function

Preface

Purpose of the documentation

The diagnostics options available with the trace and logic analyzer function are described in this documentation. Depending on the device used, the recording options can vary.

Required basic knowledge

In order to understand this documentation, the following knowledge is required:

- General knowledge in the field of automation
- Knowledge about the use of Windows-based computers
- S7-1200/1500 CPUs
  - Knowledge of working with the SIMATIC industrial automation system
  - Knowledge of working with TIA Portal >= V12.0
- SINAMICS G120
  - Knowledge of working with the drive

Validity of the documentation

This documentation applies to all products of the S7-1200, S7-1500 and SINAMICS G120 product family.

Conventions

This documentation contains pictures of the devices described. The pictures may differ slightly from the devices supplied.

Please also observe notes marked as follows:

Note

A note contains important information on the product described in the documentation, on the handling of the product and on the section of the documentation to which particular attention should be paid.

Further support

- The range of technical documentation for the individual SIMATIC products and systems is available on the Internet (http://www.siemens.com/simatic-tech-doku-portal).
- The online catalog and the online ordering system is available on the Internet (http://mall.automation.siemens.com).

13.6.1 Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mind, Siemens' products and solutions undergo continuous development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Third-party products that may be in use should also be considered. You can find more information about industrial security on the Internet (http://www.siemens.com/industrialsecurity).

To stay informed about product updates as they occur, sign up for a product-specific newsletter. You can find more information on the Internet (http://support.automation.siemens.com).

13.6.1 Description

13.6.1.1 Supported hardware

The following devices (Page 10904) support the trace and logic analyzer function:

- SIMATIC S7-1200 CPUs (as of firmware version V4.0)
- SIMATIC S7-1500 CPUs
- SINAMICS G120

13.6.1.2 Recording of measured values with the trace function

Introduction

The trace and logic analyzer function can be called in the device folder in the project navigator under the name "Traces" (Page 10882).

You record device tags and evaluate the recordings with the trace and logic analyzer function. Tags are, for example, drive parameters or system and user tags of a CPU. The maximum recording duration is limited by the memory size. How much memory is available for the recording depends on the hardware used.

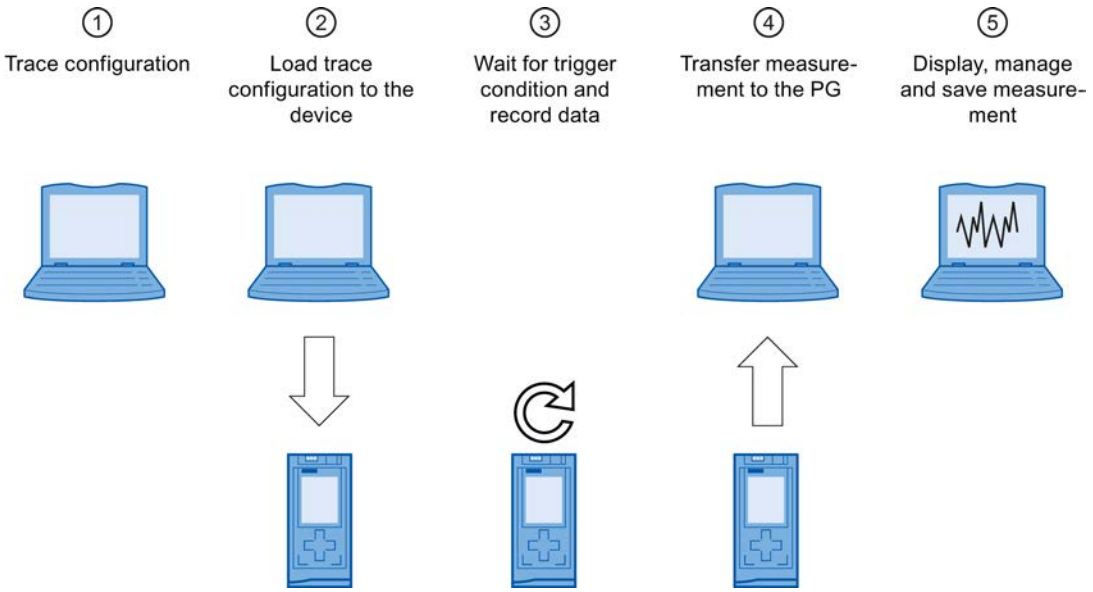
The recordings are saved on the device and, when required, can be read out with the engineering system (ES) and saved permanently. The trace and logic analyzer function is therefore suitable for monitoring highly dynamic processes. The recorded values are overwritten when "Installed traces" is activated again.

Active recordings from the axis control panel or from the PID are displayed at "Installed traces". These recordings can be added to the measurements in the curve diagram of the axis control panel or the PID via a shortcut menu command.

Depending on the device (Page 10904) used, the recording options can vary.

A quick start (Page 10892) for working with the trace and logic analyzer function can be found in the Operation section.

The following figure shows the method of operation of the "Traces":



① Trace configuration on the programming device (PG) in the TIA Portal

You can specify the signals to be recorded, the duration of the recording and the trigger condition in the trace configuration. The trace configuration depends on the device and is described at the respective device (Page 10904).

② Transferring the trace configuration from the PG to the device

You can transfer the complete trace configuration (Page 10899) to the device when an online connection is established.

③ Waiting for the recording

If the installed trace configuration is activated (Page 10899), then the recording is performed independently of the PG. The recording is started as soon as the trigger condition is satisfied.

④ Transferring the measurement from the device to the PG

The saving of the measurement in the project (Page 10901) stores the measurement in the opened project of the TIA Portal. The measurement can be saved at any time after completing the recording, irrespective of the time of the measurement.

⑤ Evaluating, managing and saving the measurement

Numerous options are available for the evaluation of the measurement in the curve diagram and in the signal table (Page 10900). Various display types are possible, for example, a bit representation for binary signals.
Measurements can also be exported and imported as a file.
With the saving of the project (Page 10901) in the TIA Portal, the measurements transferred to the project are also saved.

13.6.1.3 Trace configuration, recording and measurement

This section explains the meaning of the terms: trace configuration, recording and measurement.

Trace configuration

You can make the following settings in the trace configuration :

- Signals to be recorded
- Recording conditions
  - Sampling
  - Trigger

Recording

A recording is performed in the device. There is only one recording for each installed trace configuration. When a new recording is started, the old recording is overwritten.

An installed recording is not retentive (it is lost when the device is switched off/on) and can be saved permanently in the project as a measurement.

Installed traces

An installed trace consists of a trace configuration and optionally a recording. The maximum number of installed traces depends on the device.

The status of an installed trace can be viewed online.

Measurement

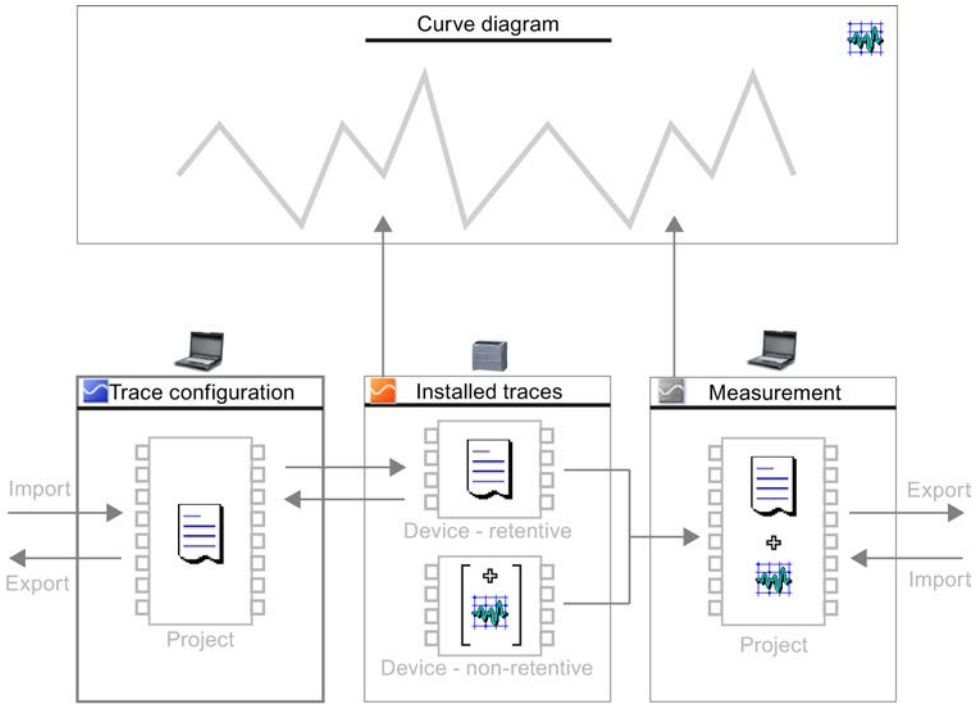
A measurement always consists of a trace configuration with an associated recording. If an installed trace contains a recording, it can be saved in the project as a measurement.

The recording of a measurement can be viewed offline.

13.6.1.4 Data storage

The trace handling and the curve diagram also enable the transfer of the trace configuration and the viewing of the recording.

The following figure is a schematic diagram of the data storage:



Note

Saving the trace configuration and measurement

You save the trace configuration and measurement with the project in the TIA Portal.

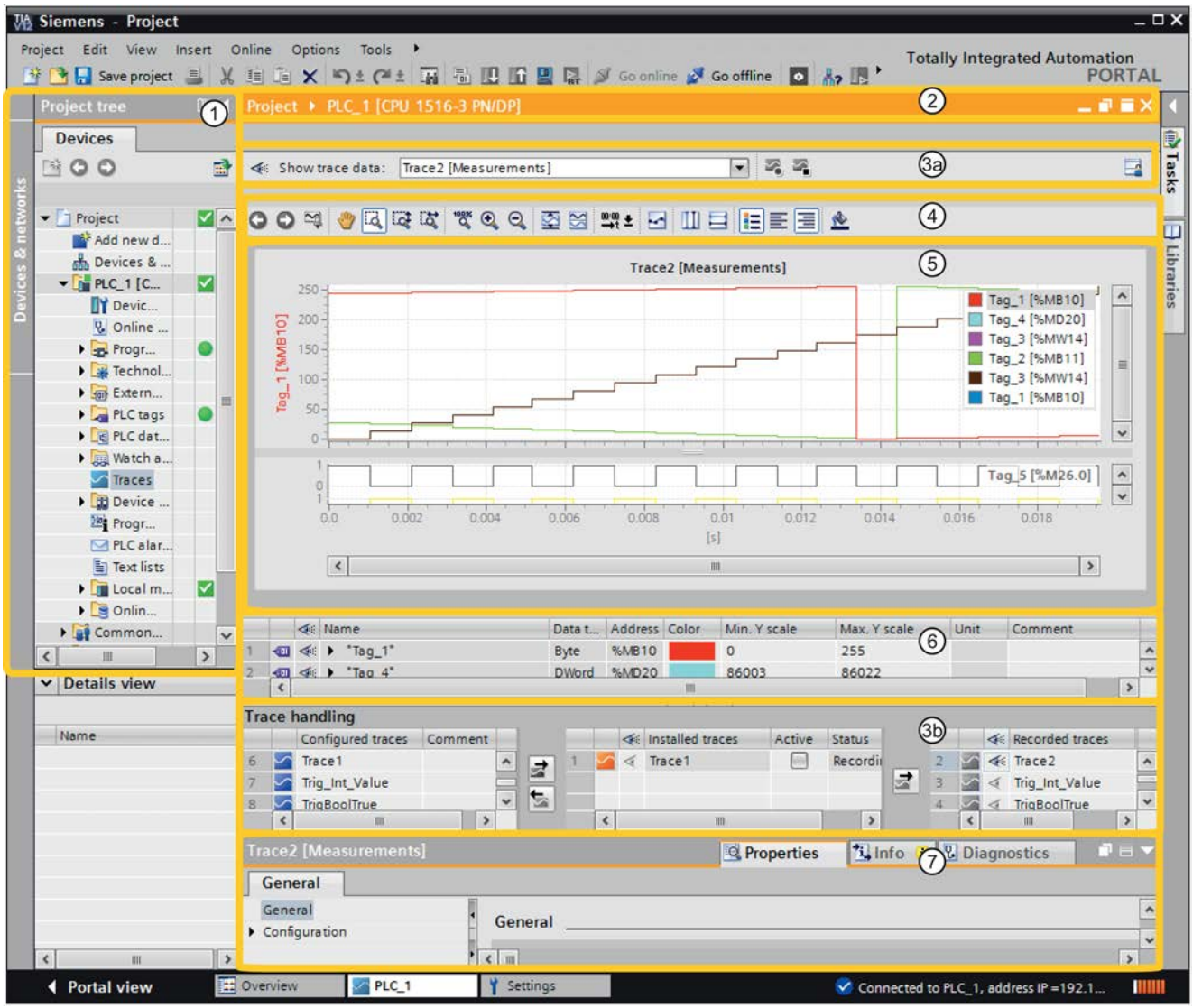
If you close the project without saving, the trace configurations and the measurements transferred to the project are discarded. The trace editor can be closed and reopened without loss of data until the project is closed.

13.6.2 Software user interface

Display areas

The user interface of the trace and logic analyzer function consists of several areas.

The example in the figure below shows the division of the user interface in the TIA Portal:



1	Project navigator	
	Working area	
2		Title bar of the working area Shows the device to which the current display belongs.
3a		Toolbar of the trace handling (Page 10888) Selection of recordings for display in the curve diagram and buttons for activation/ deactivation of installed traces.
4		Toolbar of the curve diagram (Page 10884) Tools for editing the measurement and for adapting the display in the curve diagram.
5		Curve diagram (Page 10884) Display of the recorded values.
6		Signal table (Page 10886) Signals of the shown measurement.

3b		Trace handling (Page 10888) Creation of new trace configurations and the handling of the measurements.
	Device-specific area (see Devices (Page 10904))	
7		Properties area in the Inspector window Configuration dialog boxes for the recording duration, trigger condition and signal selection.

13.6.2.1 Project navigator

If a device supports the trace and logic analyzer function, "Traces" is offered for selection in the project navigator below the device.

Double-clicking "Traces" opens the trace editor.

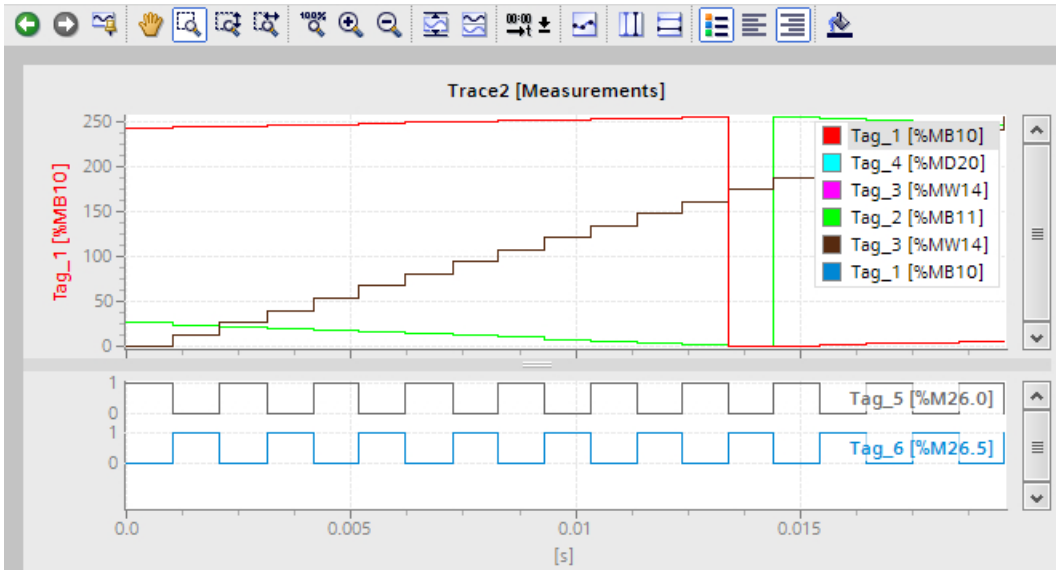
13.6.2.2 Working area

User interface - curve diagram

The curve diagram displays the selected signals of a recording. Bits are shown in the lower diagram as a bit track. You can adjust the display of the signals in the signal table (Page 10886) and with the aid of the toolbar of the curve diagram.

Setting options and displays in the curve diagram

The following figure shows an example of the display in the TIA Portal:



The scale in the diagram applies to the selected (highlighted in gray) signal in the legend. The legend can be moved laterally with the mouse.

Shortcut menu commands

The following table shows the shortcut menu commands in the curve diagram:

Shortcut menu command	Description
"Save diagram as image"	Exports the current display as a bitmap.
"Copy image to clipboard"	Copies the current display to the clipboard.
"Center measurement cursors"	Positions the activated measurement cursors at a central point in the current display.
"Add to measurements" (only axis control panel and PID)	Adds the shown recording to the "Measurements" table.

Toolbar of the curve diagram

Tools are available for adapting the display via buttons.

The following table shows the functions of the buttons:

Icon	Function	Description
	Undo zoom	Undoes the zoom function executed last. If several zoom functions have been executed, they can be undone step-by-step.
	Redo zoom	Redoes the last undone zoom function. If several zoom functions have been undone, they can be redone step-by-step.
	Standard view	Uses the current view as standard for this recording. If the trace recording is shown again later, the standard view is restored.
	Move view	Moves the display with the mouse button pressed.
	Zoom selection	Selection of an arbitrary range with the mouse button pressed. The display is scaled to the range selection.
	Vertical zoom selection	Selection of a vertical range with the mouse button pressed. The display is scaled to the range selection.
	Horizontal zoom selection	Selection of a horizontal range with the mouse button pressed. The display is scaled to the range selection.
	Display all	Scales the display of the available data so that the entire time range and all values are displayed. **Note** The dynamic progress indicator of a running trace is stopped when a zoom function is activated. This button activates the progress indicator again.
	Zoom in	Enlargement of the display. The ranges of the time axis and value axis are reduced every time the button is clicked. The curves are displayed larger.
	Zoom out	Reduction of the display. The ranges of the time axis and value axis are increased every time the button is clicked. The curves are displayed smaller.
	Scale automatically	Scaling of the display so that all values are displayed for the currently displayed time range.

Icon	Function	Description
	Arrange in tracks	Arranges signals one beneath the other without overlaps.
	Unit changeover of the time axis	Changeover of the unit between time and cycles.
	Display samples	The samples are displayed as small circles on the curves.
	Display vertical measurement cursors	Display of the vertical measurement cursors. The vertical position of the two measurement cursors can be moved with the mouse. The associated measured values and the difference of the measurement cursors corresponding to the position are shown in the signal table.
	Display horizontal measurement cursors	Display of the horizontal measurement cursors. The horizontal position of the two measurement cursors can be moved with the mouse.
	Display chart legend	Display or hiding of the chart legend in the curve diagram.
	Align the chart legend to the left	Display of the chart legend on the left side of the curve diagram.
	Align the chart legend to the right	Display of the chart legend on the right side of the curve diagram.
	Change background color	Changeover between various background colors.

User interface - signal table

The signal table lists the signals of the selected measurement and provides setting options for some properties. If recording data of "Installed traces" is displayed and the settings are changed in the signal table, these settings are retained until there is a change to the offline mode. If the "Installed trace" is added to the measurements, the current settings of the signal table are saved in the measurement.

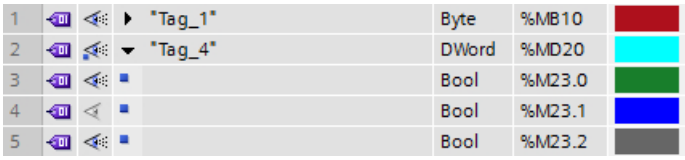
The signals can be sorted using drag-and-drop. The bits of a signal can be resorted within a signal.

Setting options and displays in the signal table

The following figure shows an example of the display in the TIA Portal:

		Name	Data type	Address	Color	Min. Y scale	Max. Y scale	Y(t1)	Y(t2)	DeltaY	Unit	Comment
1		"Tag_1"	Byte	%MB10		1	20	7	13	6		
2		"Tag_4"	DWord	%MD20		1	20	7	13	6		
3		"Tag_3"	Word	%MW...		3	60	21	39	18		

The following table shows the settings and displays of the recorded signals:

Column	Description
	Static display of the signal icon
	Selection for the display in the curve diagram The point indicates that at least one bit has been selected for display as bit track for the signal in the bit selection.
"Name"	Display of the signal name A click on the name of a displayed signal updates the scale in the curve diagram.
	Open bit selection Individual bits can also be selected for the following data types for display as a bit track in the lower curve diagram. • Byte, Word, DWord, LWord • SInt, USInt, Int, UInt, DInt, UDInt, LInt, ULInt Example of an opened bit selection for the DWORD data type:  Select or deselect the relevant bit for display by clicking the icon.
"Data type"	Display of the data type
"Address"	Display of the address (not for symbolic tags)
"Color"	Display and setting option for the color of the signal
"Min. Y-scale"	Display or input of the minimum value for the scaling of the signal
"Max. Y-scale"	Display or input of the maximum value for the scaling of the signal
"Signal group"	Display or input of the signal group name for one signal group The Y-scales are scaled identically for all signals of one signal group. Enter an identical signal group name for those signals that are to be scaled identically. Signals can be removed from the signal group by deleting the signal group name. The signal groups can be saved via the function "Use current view as standard" (button).
"Y(t1)"	Display of the value at the position of the first measurement cursor
"Y(t2)"	Display of the value at the position of the second measurement cursor
"ΔY"	Display of the value difference between the first and the second measurement cursor
"Unit"	Display of the unit (e.g. for technology objects)
"Comment"	Display and input option for a comment about the signal

Shortcut menu commands

The following table shows the shortcut menu commands of the signal table:

Shortcut menu command	Description
"Cut"	Cannot be selected.
"Copy"	Copies the contents of the selected lines to the clipboard.

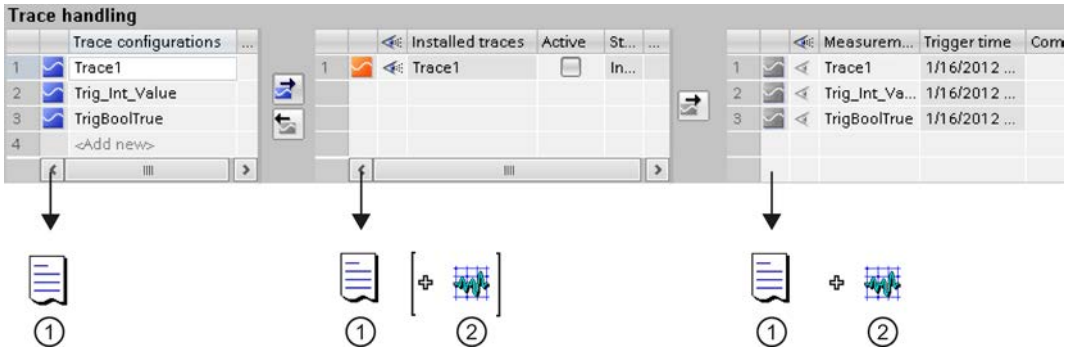
Shortcut menu command	Description
"Paste"	Cannot be selected.
"Show/hide signal"	Shows/hides the signal in the curve diagram.
"Bring to foreground"	Displays the selected signal in the foreground. The other signals are hidden.

User interface - trace management

The trace handling enables the configuration and handling of trace configurations.

Setting options and displays in the trace handling

The following figure shows an example of the display in the TIA Portal:



① Trace configuration

② Recording (recorded values)

Toolbar of the trace handling

A recording can be selected for display via a drop-down list and buttons, and activated or deactivated.

The example in the figure below shows the toolbar of the trace handling

The following table describes the functions of the buttons:

Icon	Description
	Static icon display of the drop-down list selection in the curve diagram
	Selection of an "Installed trace" or a "Measurement" for display in the curve diagram
	Activation of the recording The button is available for "Installed traces".
	Deactivation of the recording The button is available for "Installed traces".

"Trace configurations" table

The table contains all the trace configurations of the project that can be transferred to the device. Each trace configuration contains the tags to be recorded and the required recording settings. The data is displayed in offline and online mode.

The following table shows the settings and displays:

Column	Icon	Description
-		Display of the icon for the trace configuration Incorrect trace configurations are shown with a red border and cannot be transferred to the device. Check the configuration in the Inspector window. Double-clicking the icon brings the Inspector window with the properties to the foreground.
"Trace configuration"	-	Name of the trace configuration. The name can be changed after double-clicking the field.
"Comment"	-	Input field for a comment.

Shortcut menu commands

The following table shows the shortcut menu commands of the "Trace configurations" table when a line is selected:

Shortcut menu command	Description
"Cut"	Cannot be selected.
"Copy"	Copies the contents of the selected lines to the clipboard.
"Paste"	Pastes the contents of the clipboard.
"Delete"	Deletes the selected lines from the table or deletes the name of the selected cell.
"Rename"	Switches the selected cell to the editing mode.
"Transfer trace configuration to device"	Transfers the trace configuration selected in the "Trace configurations" table to the device. The button is active when all the required settings in the device-specific configuration have been configured correctly in the Inspector window and there is an online connection.
"Export trace configuration"	Exports a trace configuration as a file with the "*.ttcfgx" file extension. For compatibility reasons, the "*.ttcfg" file extension will be supported in V12, but does not contain any information about the device family.
"Import trace configuration"	Imports a trace configuration from a file.
"Properties"	Brings the Inspector window with the device-specific properties to the foreground.

"Installed traces" table

The table contains the trace configurations already transferred to the device and their current status. The data is displayed in the online mode.

The following table shows the settings and displays:

Column	Icon	Description
-		Static display of the online icon Double-clicking the icon brings the Inspector window with the device-specific properties to the foreground.
		Selection for the display in the curve diagram A click on the icon shows/hides the recording in the curve diagram.
"Installed traces"	-	Name of the trace configuration.
"Active"		Activation of the recording and feedback of the status. The checkbox is reset at the end of the recording. If the recording of an installed trace is repeated, then the settings relevant for the display (curve diagram and signal table) are also retained for the new recording.
"Status"	-	Status display of the trace configuration The following states are possible: • "Installing" • "Waiting for trigger" • "Recording" • "Recording completed" • "Aborted" • "Fault"
"Comment"	-	Display of the trace configuration comment

Shortcut menu commands

The following table shows the shortcut menu commands of the "Installed traces" table when a line is selected:

Shortcut menu command	Description
"Cut"	Cannot be selected.
"Copy"	Copies the contents of the selected lines to the clipboard.
"Paste"	Cannot be selected.
"Delete"	Deletes the selected lines from the table.
"Rename"	Cannot be selected.
"Add to trace configurations"	Transfers the installed trace configuration to the "Trace configurations" table.
"Add to measurements"	Adds the selected "Installed trace" to the "Measurements" table. **Note** The "Installed trace" data can be saved as a measurement after being displayed once in the curve diagram. The recording data is loaded from the device for the display.
"Activate"	Activates the recording and shows the status in the table
"Show/hide in curve diagram"	Shows/hides the recording in the curve diagram.
"Properties"	Brings the Inspector window with the device-specific properties to the foreground.

"Measurements" table

The table contains the trace configurations with recordings. The data is displayed in offline and online mode.

Measurements can also be exported and imported, see Exporting and importing measurements (Page 10901).

The following table shows the settings and displays:

Column	Icon	Description
-		Static display of the icon for a measurement Double-clicking the icon brings the Inspector window with the device-specific properties to the foreground.
		Selection for the display in the curve diagram A click on the icon shows/hides the recording in the curve diagram.
"Measurements"	-	Name of the measurement The name can be changed after double-clicking the field.
"Trigger time"	-	The meaning of the time stamp depends on the device Examples: • SIMATIC S7-1200/1500 CPUs The absolute time of the controller is entered at the start of the recording. • SINAMICS G120 The transfer time is entered with the transfer of the trace configuration to the device.
"Comment"	-	Input field for a comment.

Shortcut menu commands

The following table shows the shortcut menu commands of the "Measurements" table when a line is selected:

Shortcut menu command	Description
"Cut"	Cannot be selected.
"Copy"	Copies the contents of the selected lines to the clipboard.
"Paste"	Pastes the contents of the clipboard.
"Delete"	Deletes the selected lines from the table or deletes the name of the selected cell.
"Rename"	Switches the selected cell to the editing mode.
"Add to trace configurations"	Transfers the trace configuration of the measurement to the "Trace configurations" table.
"Show/hide in curve diagram"	Shows/hides the recording in the curve diagram.
"Export measurement"	Exports a measurement as a file with the "*.ttrecx" or "*.csv" file extension. For compatibility reasons, the "*.ttrec" file extension will be supported in V12, but does not contain any information about the device family.

Shortcut menu command	Description
"Import measurement"	Imports a measurement from a file with the "*.ttrecx" file extension. For compatibility reasons, the "*.ttrec" file extension will be supported in V12, but does not contain any information about the device family.
"Properties"	Brings the Inspector window with the device-specific properties to the foreground.

Buttons

The following table shows the functions available via buttons:

Icon	Description
	Button to transfer a selected trace configuration to the device The selected trace configuration in the "Trace configurations" table is transferred to the device. The button is active when all the required settings in the configuration have been correctly configured in the Inspector window and there is an online connection.
	Button to transfer a selected trace configuration from the device The trace configuration selected in the "Installed traces" table is transferred from the device to the "Trace configurations" table.
	Button to transfer a selected measurement from the device to the project The selection in the "Installed traces" table is added to the "Measurements" table. **Note for S7 CPUs** In order to be able to save the "Installed trace" data as a measurement, it must first have been displayed once in the curve diagram. The recording data is loaded from the device when displayed.

13.6.2.3 Device-specific area

The device-specific area enables the trace configuration.

The signals to be recorded and the trigger condition are specified here. These and other configuration options are described in the device-specific part (Page 10904) for the individual devices.

13.6.3 Operation

13.6.3.1 Quick start

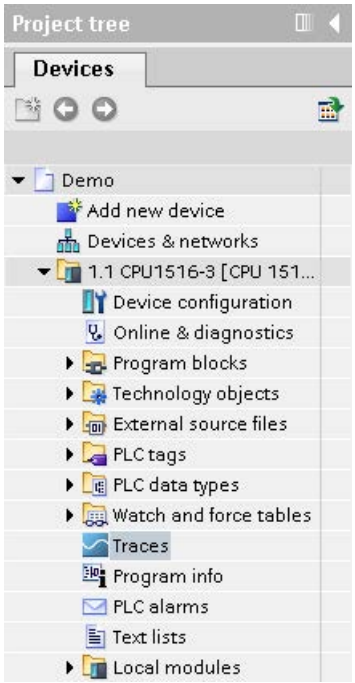
This description shows the steps for a recording of the S7-1500 CPU as an example. The displayed settings can differ depending on the device.

Requirement

A device is configured that supports the trace and logic analyzer function.

Calling the trace editor

The following figure shows the project navigator with "Traces" below the device:

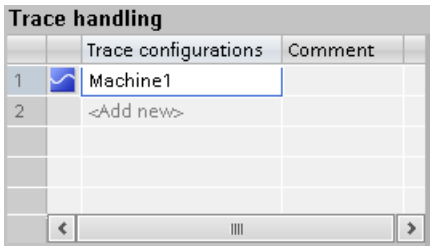


Procedure:

1. Open the trace editor by double-clicking the "Traces" entry.

Creating a trace configuration

The following figure shows the "Trace configurations" table:

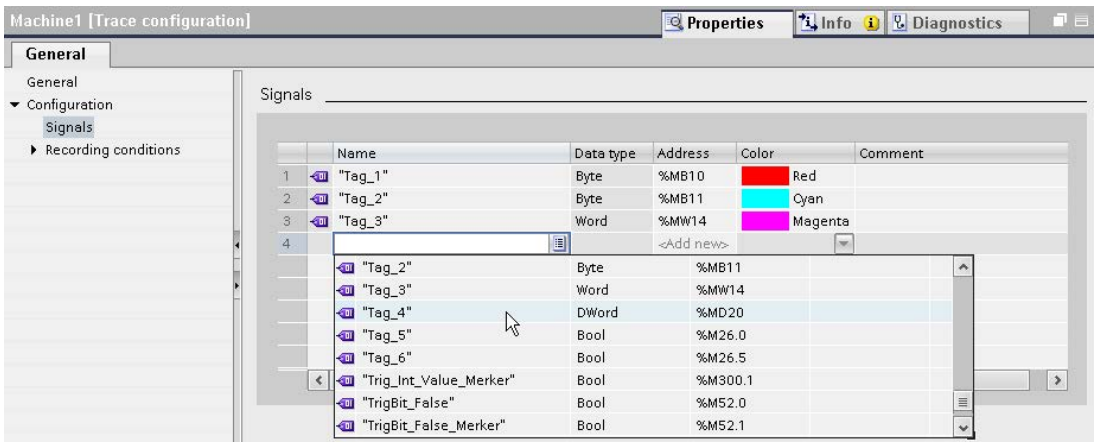


Procedure:

1. Enter the name of the trace configuration in the "Trace configurations" table.

Selecting signals

The following figure shows the configuration of the signals:



Procedure:

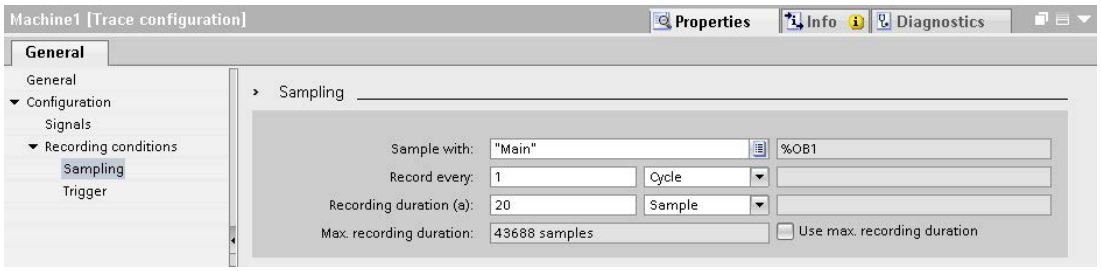
1. Select the signals to be recorded in the "Signals" area of the Inspector window.

Or:

1. Drag a signal to the table using drag-and-drop.

Configuring the recording cycle

The following figure shows the configuration of the sampling:

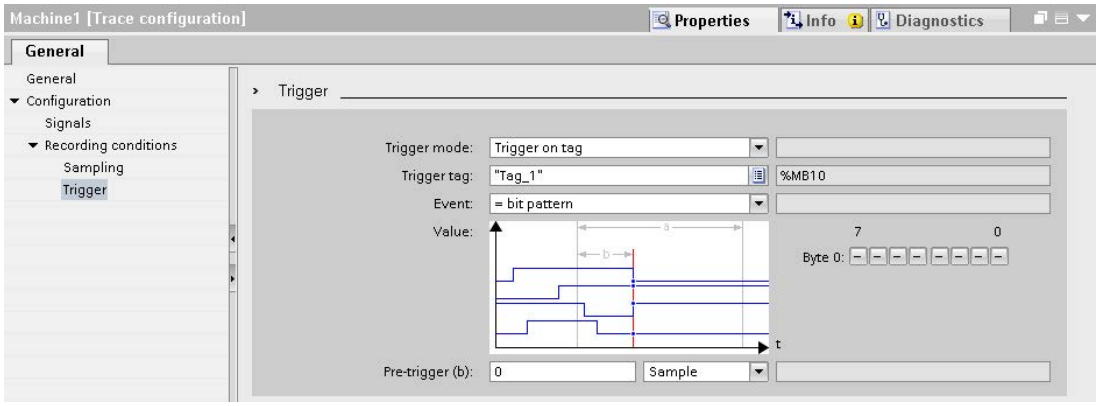


Procedure:

1. Configure the sampling.

Configuring the trigger

The following figure shows the configuration of the trigger:

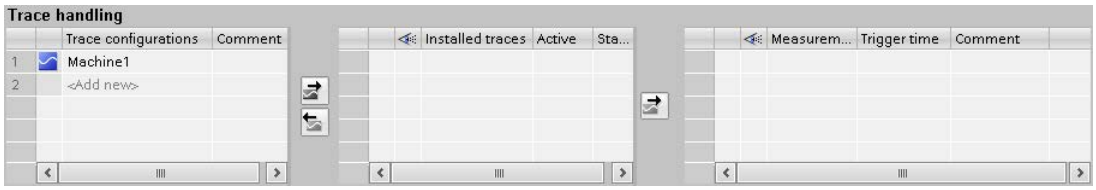


Procedure:

1. Configure the trigger mode and the condition for the selected trigger.

Transferring the trace configuration to the device

The following figure shows the trace handling in online mode:

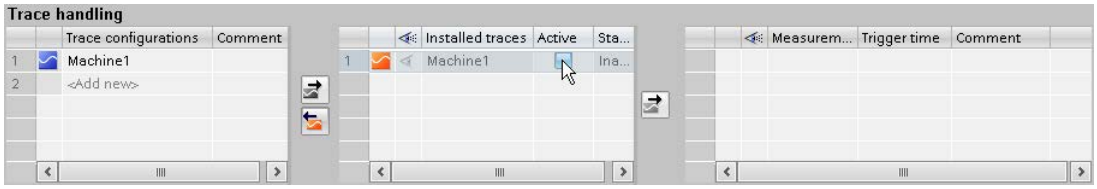


Procedure:

1. Check whether the configuration is correct based on the icon in the "Trace configurations" table.
   The icon indicates a correct trace configuration.
   The icon indicates a faulty trace configuration.
   Only correct trace configurations can be transferred to the device.
2. Establish an online connection to the device.
3. Select the trace configuration in the "Trace configurations" table and transfer it to the device with the button.

Activating a recording

The following figure shows the trace handling with an installed trace configuration:

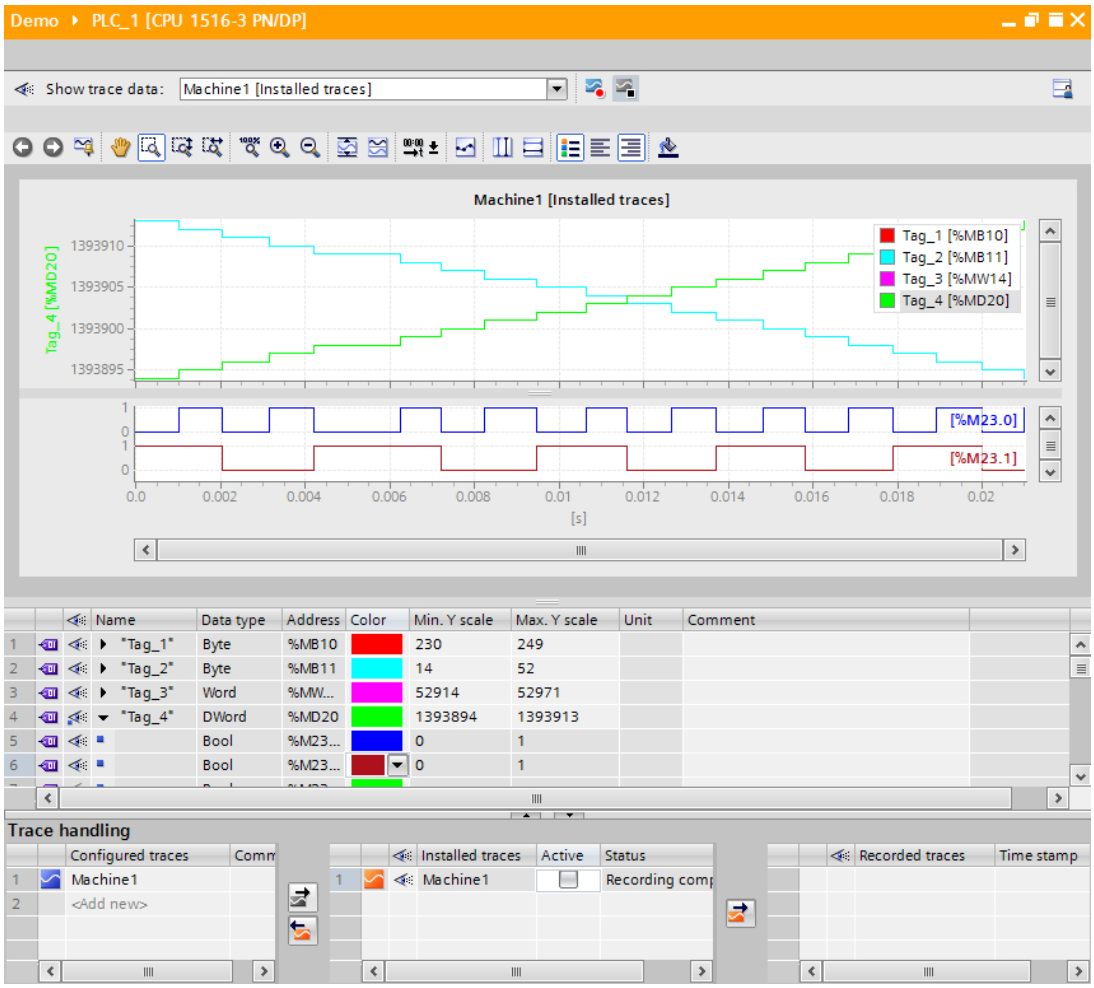


Procedure:

1. Click the checkbox in the "Active" column in the "Installed traces" table.

Displaying the recording

The following figure shows the curve diagram with the recording of the "Installed trace":

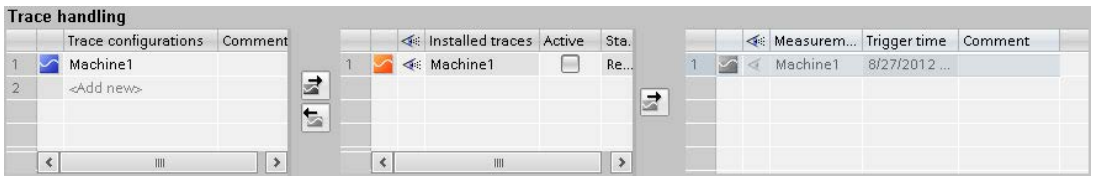


Procedure:

1. Wait until the "Recording" or "Recording completed" status is displayed in the "Installed traces" table.
2. Click the icon in the "Installed traces" table to display the recording in the curve diagram.
3. Click the icon of a signal in the signal table.
   The individual bits of the signal are offered for display as a bit track.
4. In the signal table, select or deselect the individual signals and bits for display with the icon.

Saving the measurement in the project

The following figure shows the measurement saved in the project:



Procedure:

1. Select the measurement in the "Installed traces" table and transfer it to the project with the button.

13.6.3.2 Using the trace function - overview

Requirement

A device is configured in the TIA Portal that supports the trace and logic analyzer function and to which an online connection has been established.

Procedure

The following table shows a procedural overview with typical steps when working with the trace and logic analyzer function.

Step	Description
1	Calling the trace editor (Page 10898)
2	Configuring the trace (Page 10898)
3	Transferring the trace configuration to the device (Page 10899)
4	Activating/deactivating an installed trace (Page 10899)
5	Monitoring the recording (Page 10900)
6	Saving measurements in the project (Page 10901)
7	Displaying the recording (Page 10900)

13.6.3.3 Calling the trace editor

A trace editor can be opened for each device in the project navigator.

The following instructions describe how you call the trace editor in the work area of the TIA Portal.

Requirement

A device is configured that supports the trace and logic analyzer function.

Procedure

To call the trace editor, proceed as follows:

1. In the project navigator, double-click the device.
   The device folder opens.
2. Double-click "Traces".

13.6.3.4 Trace handling

Configuring the trace

Requirement

The trace editor is open.

Creating and deleting the trace configuration

To create a new trace configuration, proceed as follows:

1. Click the "<Add>" entry in the "Trace configurations" table in the Trace handling (Page 10888).
2. Enter the name for the trace configuration.
   The new trace configuration is created.

To delete a trace configuration, proceed as follows:

1. Right-click a trace configuration in the "Trace configurations" table and select the shortcut menu command "Delete".
   The trace configuration is deleted.

Configuring the trace

In the configuration, you specify the recording and trigger conditions and select the signals to be recorded.

See Section "Configuration" below the respective device (Page 10904).

Note

Saving the trace configuration

You save the trace configuration with the project in the TIA Portal.

If you close the project without saving, the configuration is discarded. The trace editor can be closed and reopened without loss of data until the project is closed.

Transferring the trace configuration to the device

Requirement

- A valid trace configuration is located in the "Trace configurations" table.
- The maximum number of "Installed traces" has not been reached yet.
- There is an online connection to the device.

Procedure

To transfer a trace configuration to the device, proceed as follows:

1. Select a trace configuration in the "Trace configurations" table.
2. Click the button to transfer the trace configuration to the device.

Result

The trace configuration is transferred to the device and displayed in the "Installed traces" table.

Activating/deactivating an installed trace

Requirement

A trace configuration is installed and there is an online connection to the device.

Activating an installed trace

To activate the recording for an installed trace, proceed as follows:

1. Activate the checkbox of the "Active" column in the "Installed traces" table.
   The installed trace is activated and starts the recording according to the configured trigger condition. The trigger condition is device-specific and described in Section "Configuration" below the respective device (Page 10904).

Note

When a recording is restarted, the previously recorded values are lost.

To save the recorded values, save the measurement in the project (Page 10901) before you activate the recording again.

Deactivating an installed trace

To deactivate an activated installed trace, proceed as follows:

1. Deactivate the checkbox of the "Active" column in the "Installed traces" table.
   The installed trace is deactivated.

Displaying the recording

Requirement

A trace configuration with recording is in the device and there is an online connection to the device or there is an entry in the "Measurements" table.

Procedure

To display the recording, proceed as follows:

1. Select an entry in the "Installed traces" table.
2. Click in the column for the selected entry.

Or:

1. Select an entry in the "Measurements" table.
2. Click in the column for the selected entry.

Result

The recording is displayed in the curve diagram and the signal table.

Saving measurements in the project

Requirement

A trace configuration with recording is installed and there is an online connection to the device.

The "Installed trace" data must have been displayed at least once in the curve diagram. The recording data is loaded from the device for the display.

Procedure

To save a recording in the project, proceed as follows:

1. Click an entry in the "Installed traces" table in the trace handling.
2. Click the button to transfer the selected entry.
   The selected entry in the "Installed traces" table is added to the "Measurements" table.
3. Save the project in the TIA Portal.

Exporting and importing measurements

Requirement

There is at least one measurement in the "Measurements" table.

Exporting measurements

To export a measurement, proceed as follows:

1. Right-click an entry in the "Measurements" table and select the shortcut menu command "Export measurement".
2. Select a folder, a file name and a data type to save the measurement.
3. Click the "Save" button.

Importing measurements

To import a measurement, proceed as follows:

1. Right-click in the "Measurements" table and select the shortcut menu command "Import measurement".
2. Select the file of the "*.ttrecx" file type with the measurement to be imported.
3. Click the "Open" button.
   The imported measurement is displayed with the file name in the "Measurements" table.

Transferring the trace configuration from the device to the project

Requirement

A trace configuration is installed and there is an online connection to the device.

Procedure

To transfer a trace configuration to the project, proceed as follows:

1. Select an entry in the "Installed traces" table.
2. Click the button to transfer the trace configuration from the device.

Result

The configuration of the "installed trace" is transferred to the "Trace configurations" table.

Deleting installed traces

Requirement

A trace is installed and there is an online connection to the device.

Deleting an installed trace

To delete an installed trace, proceed as follows:

1. Select one or more lines to be deleted in the "Installed traces" table.
2. Press <Del> to delete the "Installed traces".

13.6.3.5 Signal table

Use of the signal table

The signal table shows the signals of an installed trace or a measurement. You can show or hide individual signals for the display in the table and adapt the properties for the display. Individual bits can be selected for some data types and displayed as a bit track.

The following operating instructions describe the operation of the signal table.

Requirement

- An entry is selected in the column in the "Installed traces" or "Measurements" table.
- For the display of individual bits as a bit track:
  At least one recorded signal supports the display as a bit track.

Selecting individual signals in the signal table and changing the format

To adapt the display to suit your requirements, proceed as follows:

1. Click the icon of the respective signal in the column to select or deselect it for the display.
2. Click in the "Color" column for the respective signal to change the default color of the signal.
3. Click the "Bring to foreground" entry in the shortcut menu of a signal to display it in the foreground.

Selecting individual bits in the signal table for display as a bit track

To display individual bits as a bit track in the lower curve diagram, proceed as follows:

1. Click the icon of a signal in the signal table.
2. Click the icon in the opened bit selection of the signal to select or deselect individual bits for the display.

13.6.3.6 Curve diagram

Use of the curve diagram

The curve diagram shows the signals of a recording selected in the signal table. Select the recording in the "Installed traces" or "Measurements" table of the trace handling.

The display area can be zoomed as required. Measurement cursors can be used to select individual values for display in the signal table.

The following operating instructions describe the use of the measurement cursors.

Requirement

A recording has been selected for display.

Evaluation of a certain instant of a recording

To display the values for a specific sample, proceed as follows:

1. Display the vertical measurement cursors via the button.
2. Move a measurement cursor with the mouse to the required position in the recording. The values of the signals are displayed in the signal table. The time or sample for the measurement cursors is displayed in the lower area of the curve diagram.

Evaluation of the difference between two samples

To display the difference, proceed as follows:

1. Display the vertical measurement cursors via the button.
2. Move both measurement cursors with the mouse to the required samples in the recording. The values of the signals and the difference are displayed in the signal table. The time or sample for the measurement cursors is displayed in the lower area of the curve diagram.

Using horizontal measurement cursors

To check whether a certain value has been reached, proceed as follows:

1. Display the horizontal measurement cursors via the button.
2. Move a measurement cursor with the mouse to the required value of the recording.
   The values of the measurement cursor for the selected signal are displayed in the lower area of the curve diagram.

Printing a recording

The curve diagram supports the saving of the display as a bitmap and the copying of the display to the clipboard. Also use these functions (Page 10884) for printing.

13.6.4 Devices

13.6.4.1 S7-1200/1500 CPUs

Recordable variables

Device-dependent recording of tags

The following list shows the operand areas from which tags can be recorded:

- Process image input
- Process image output
- Bit memory
- Data blocks

Data types

Elementary data types can be recorded. The availability of the individual data types depends on the device used:

The following table lists the elementary data types:

Data types	Note
Binary numbers	
BOOL	-
Bit strings	
BYTE	-
WORD	-
DWORD	-
LWORD	Symbolic name required

Data types	Note
Integers	
SINT	-
USINT	-
INT	-
UINT	-
DINT	-
UDINT	-
LINT	Symbolic name required
ULINT	Symbolic name required
Floating-point numbers	
REAL	-
LREAL	Symbolic name required

Lifetime of the installed trace configuration and recorded values

Installed trace configurations are retained after POWER OFF. The recording is activated again after the restart of the CPU.

Recorded values are lost during the restart.

Note

Downloading a configuration to the device in the "STOP" operating state

Note that after downloading a configuration in the "STOP" operating state, you must check the installed traces and, if required, reactivate them or transfer them again.

Note

If signals or trigger tags that affect the address are changed, the trace configuration must also be transferred to the device again.

This is the case for example, when a data block is shortened or extended or the data type is changed.

Recording levels

The following list shows the execution levels that can be selected for the recording cycle:

- Program cycle - OB 1
- Time-of-day interrupt - OB 1x
- Time-delay interrupt - OB 2x
- Watchdog interrupt - OB 3x
- Synchronized processing cycles - OB 6x, not OB 60

- Servo task - OB 91 (S7-1500)
- IPO - OB 92 (S7-1500)

Note

The measured values are recorded at the end of the OB after the processing of the user program.

Quantity structure

The following table shows the maximum quantity structure that can be recorded with the trace and logic analyzer function:

Device	Maximum number of "Installed traces"	Maximum number of signals per trace configuration
S7-1200 (as of firmware version V4.0)	2	16
S7-1500	At least 4 (depending on the CPU type)	16

Example of S7-1516

- Maximum of 3854 samples for 16 long tags
- Maximum of 21844 samples for 16 binary tags
- Maximum of 58250 samples for 1 binary tag

Refer to respective manual for more detailed information.

CPU load through trace recording

A trace recording increases the runtime of the respective recording level that can result in an execution level overflow with high utilization of the CPU.

Remedy

- **Change the trace configuration**
  1) Configure fewer tags and signals.
  2) Then increase the number of tags and signals up to the maximum number of signals step-by-step without an execution level overflow.
- **Select a slower recording level**

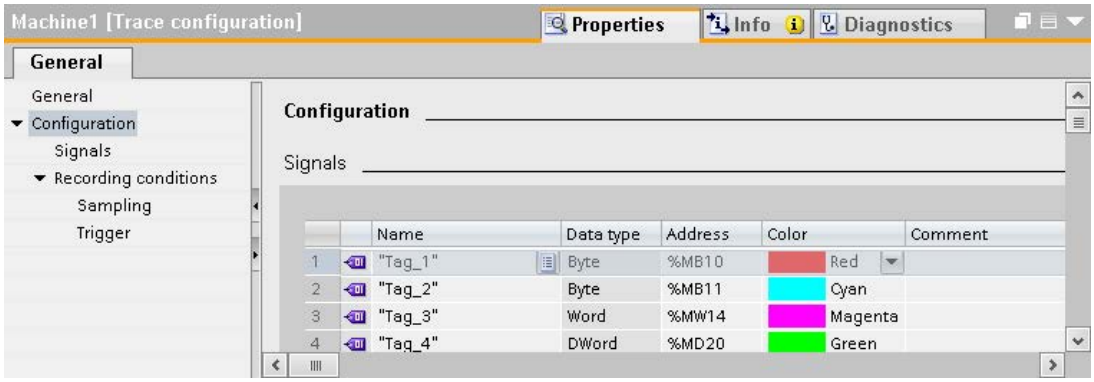
Software user interface of the device-specific area

Structure of the user interface

Display areas in the Properties window of the Inspector

The settings options differ depending on the configured device.

The following figure shows an example of the display in the TIA Portal:



The area navigation provides the following entries for selection:

- General
- Configuration
  - Signals (Page 10908)
  - Recording conditions (Page 10909)

Display of the properties of a trace configuration

The trace configuration is selected in the trace handling:

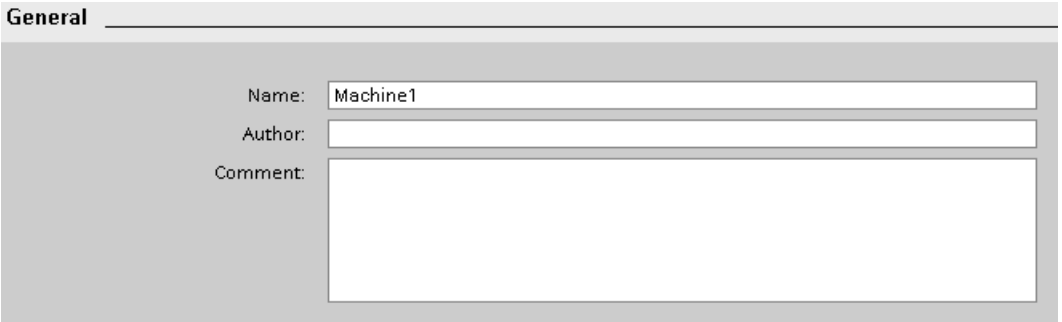
- Selection in the "Trace configurations" table
  The properties are displayed in the Inspector window and can be changed.
- Selection in the the "Installed traces" table or "Measurements" table
  The properties are displayed write-protected in the Inspector window.

User interface - General

The "General" area shows the name of the trace configuration and input fields for the author and a comment.

Input options and displays in General

The following figure shows an example of the display in the TIA Portal:



The following table shows the input fields and displays:

Column	Icon	Description
"Name"	-	Name of the trace configuration
"Author"	-	Author of the trace configuration
"Comment"	-	Input field for a comment

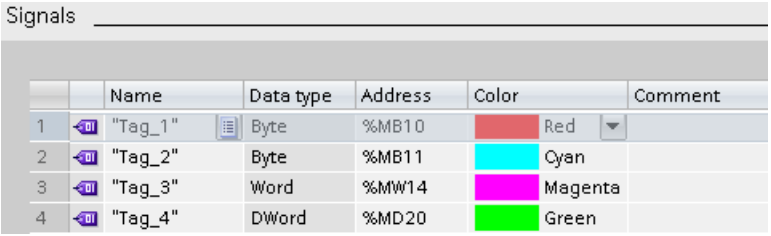
User interface - Signals

The "Signals" area shows a table in which the signals to be recorded are configured for the selected trace configuration.

Signals can also be inserted in the table using drag-and-drop.
The signals can be sorted using drag-and-drop.

Setting options and displays in "Signals"

The following figure shows an example of the display in the TIA Portal:



The following table shows the settings and displays:

Column	Icon	Description
		Display of the signal icon for a selected signal.
"Name"	-	Input field for the name or address of the signal. **Examples:** • M0.0 • DB1.DBW3 • "Data_block_1".pressure
-		Button to open the signal selection table. The button is displayed when the table line is selected. Clicking the icon opens a table which offers possible signals for selection. The selected signal is displayed in the input field.
"Data type"	-	Text field with display of the data type for the signal.
"Address"	-	Input field for the address of the signal. With purely symbolic signals the field remains empty.
"Color"	-	Text field for display and selection of the color. Click the selected field to display the signal color. Click the signal color to open the color selection dialog.
"Comment"		Input field for a comment on the signal.

Shortcut menu commands

The following table shows the shortcut menu commands of the table:

Shortcut menu command	Description
"Cut"	Cannot be selected.
"Copy"	Copies the contents of the selected lines to the clipboard.
"Paste"	Pastes the contents of the clipboard to the selected line. The existing contents are overwritten.
"Delete"	Deletes the selected lines from the table or deletes the name of the selected cell.
"Rename"	Switches the selected cell to the editing mode.

Recording conditions

Supported data types

The following table shows the supported data types for the trigger tag:

Memory requirement and format of the number	Data type
1 byte	BOOL
8-bit integers	SINT, USINT, BYTE
16-bit integers	INT, UINT, WORD
32-bit integers	DINT, UDINT, DWORD
64-bit integers	LINT, ULINT, LWORD (device-dependent)

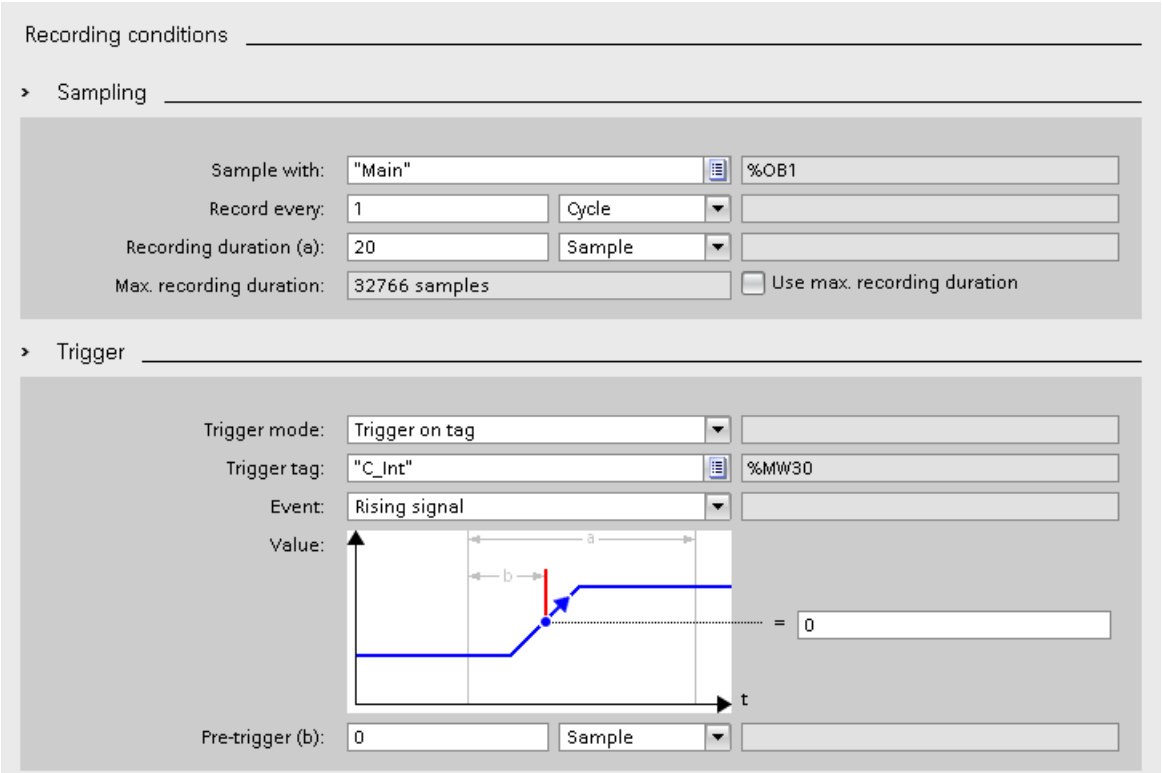
Memory requirement and format of the number	Data type
32-bit floating-point numbers	REAL
64-bit floating-point numbers	LREAL

User interface - Recording conditions

The "Recording conditions" area shows the trigger condition for the selected trace configuration and in which cycle, how fast and how long the recording is made. The configuration is only possible if the trace configuration has been selected in the "Trace configurations" table of the trace handling.

Setting options and displays in "Recording conditions"

The following figure shows an example of the display in the TIA Portal:



Setting/display		Description
"Recording time"		
	Drop-down list	Selection of the recording time. See Recording levels (Page 10905)
	Text field	Detailed information on the selected recording time.
"Record every"		
	Input field	Input of the reduction in relation to the cycles.

Setting/display		Description
	Drop-down list	Selection of the reduction factor. The following settings are possible: • "Cycle"
	Text field	Display of the sampling time, taking into account the configured reduction.
"Recording duration"		
	Input field	Input of the recording duration in relation to the samples. If the "Recording duration = max. recording duration" checkbox is activated, entries are overwritten by the value displayed in "Max. recording duration".
	Drop-down list	Select the time unit for the recording duration. The following settings are possible: • "Samples"
	Text field	Displays the calculated recording duration.
"Max. recording duration"		
	Text field	Displays the calculated maximum recording duration. The "Max. recording duration" depends on how many signals are recorded and the data type of these signals.
	"Use max. recording duration"	Set the recording duration to the maximum value. When the checkbox is activated, the recording duration is set to the maximum possible recording duration. The set reduction in the "Record every" input field is taken into account. The recording duration is also adapted when additional signals are added.
"Trigger mode"		Selection of the trigger mode.
	Drop-down list	The following settings are possible: • "Record immediately" The recording is made immediately after the device is activated. • "Trigger on tag" The recording is made as soon as the installed trace is activated and the configured trigger condition is fulfilled.
	Text field	Display of the trigger type.
"Trigger tag"		The "Trigger tag" specifies a signal that is used to trigger the recording.
	Input field	Enter a signal. **Examples:** • M0.0 • DB1.DBW3 • "DataBlock_1".Temperature
		Opens the signal selection table. Clicking the icon opens a table which offers possible signals for selection as trigger tag. The selected signal is displayed in the input field.
	Text field	Display of the trigger tag address.
"Event"		The events that can be used on this trigger tag are offered for selection according to the data type of the trigger tag. The event can only be configured when a valid signal has been entered as trigger tag.

Setting/display		Description
	Drop-down list	Event selection for which the trigger tag is checked. The entries in the drop-down list are described in Section Trigger event (Page 10912).
	Text field	Display of the comparison type.
"Value"		Configuration of the selected event. The configuration options differ depending on the format of the trigger tag and the selected event. See Trigger event (Page 10912).
"Pre-trigger"		"Pre-trigger" defines the time during which the signals are already recorded before the actual trigger condition is fulfilled. If the trigger event occurs immediately or shortly after the activation of the recording, this may result in a shorter recording duration. Example of "recording duration" = 20 samples and "pre-trigger" = 5 samples: • Trigger event occurs 50 samples after activation of the recording Actual recording duration = 20 samples • Trigger event occurs 2 samples after activation of the recording Actual recording duration = 17 samples
	Input field	Input of the duration in relation to the selection in the drop-down list.
	Drop-down list	Select the time unit The following settings are possible: • "Samples"
	Text field	Displays the calculated "pre-trigger" period

Trigger event

Depending on the selection in the drop-down list box, the further settings differ for the "event".

The individual events are described below.

"=TRUE"

Supported data types: Bit (Page 10909)

The recording starts when the state of the trigger is TRUE.

"=FALSE"

Supported data types: Bit (Page 10909)

The recording starts when the state of the trigger is FALSE.

"Rising edge"

Supported data types: Bit (Page 10909)

The recording is started when the trigger state changes from FALSE to TRUE.
After activation of the installed trace, at least two cycles are required to identify the edge.

"Rising signal"

Supported data types: Integers and floating-point numbers (Page 10909)

The recording is started when the rising value of the trigger reaches or exceeds the value configured for this event.
After activation of the installed trace, at least two cycles are required to identify the edge.

"Falling edge"

Supported data types: Bit (Page 10909)

The recording is started when the trigger state changes from TRUE to FALSE.
After activation of the installed trace, at least two cycles are required to identify the edge.

"Falling signal"

Supported data types: Integers and floating-point numbers (Page 10909)

The recording is started when the falling value of the trigger reaches or falls below the value configured for this event.
After activation of the installed trace, at least two cycles are required to identify the edge.

"In the range"

Supported data types: Integers and floating-point numbers (Page 10909)

The recording starts as soon as the value of the trigger is in the value range configured for this event.

"Outside of the range"

Supported data types: Integers and floating-point numbers (Page 10909)

The recording starts as soon as the value of the trigger is outside the value range configured for this event.

"= value"

Supported data types: Integers (Page 10909)

The recording starts when the value of the trigger is equal to the value configured for this event.

"<> value"

Supported data types: Integers (Page 10909)

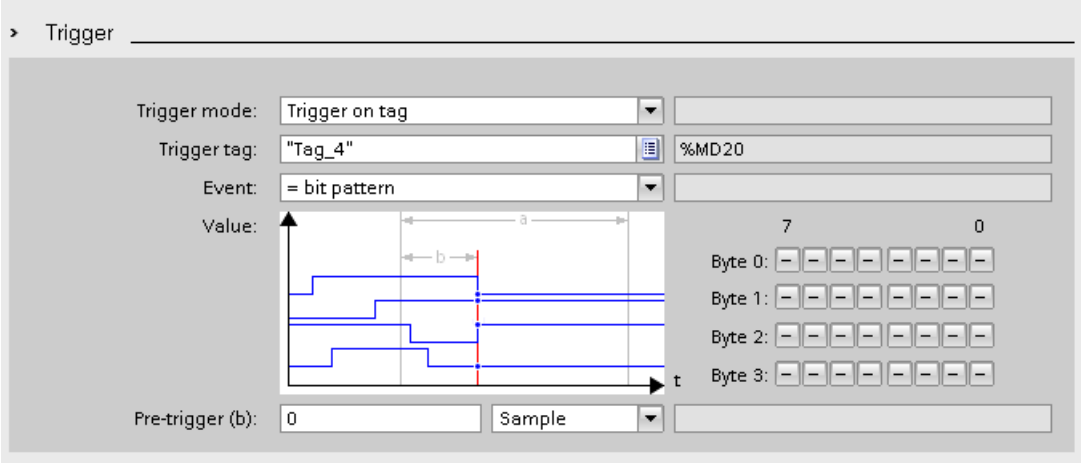
The recording starts when the value of the trigger is not equal to the value configured for this event.

"= bit pattern"

Supported data types: Integers (Page 10909)

The recording starts when the value of the trigger matches the bit pattern configured for this event.

The following figure shows the setting options for a "bit pattern":



It is possible to switch between the icons by clicking the respective button.

The following table shows the icons:

Icon	Description
–	Bit is not evaluated
0	Bit is checked for FALSE
1	Bit is checked for TRUE

"<> bit pattern"

Supported data types: Integers (Page 10909)

The recording starts when the value of the trigger does not match the bit pattern configured for this event.

See also

Configuring the trigger conditions (Page 10916)

Configuration

Trace configuration - overview

The configuration of the recording conditions and the signals to be recorded is device-specific.

Requirement

A trace configuration has been created and selected in the "Trace configurations" table of the trace handling.

Procedure

The following table shows the procedure for configuring.

Step	Description
1	Documentation of the configuration (optional) Enter a comment and an author for the configuration.
2	Selecting signals (Page 10915) Select the signals to be recorded in the "Signals" area.
3	Configuring the recording cycle and duration Select a recording time, a cycle and the duration in the "Recording conditions" area.
4	Configuring the trigger conditions (Page 10916) In the "Recording conditions" area, select whether the recording is to be performed immediately or depending on a trigger condition.

Selecting signals

Requirement

- A trace configuration has been created and selected in the "Trace configurations" table of the trace handling.
- The "Signals" area is open in the Inspector window.

Procedure

To configure the signals to be recorded, proceed as follows:

1. Click in the first empty line of the table.
2. Click in the first empty line of the "Name" column.
3. Select a signal. The following options are available:
   - In the "Name" column, click the button and select a tag.
   - Enter the symbolic tag name in the cell in the "Name" column.
   - Enter the address directly in the "Address" column.
   - Drag a signal to the table using drag-and-drop.
4. Click in the "Display color" column and select a color for the display of the signal.
5. Click in the "Comment" column and enter a comment for the signal.
6. Repeat the procedure from step 1 until all the signals to be recorded have been entered in the table.

Configuring the recording cycle and duration

Requirement

- A trace configuration has been created and selected in the "Trace configurations" table of the trace handling.
- The "Recording conditions" area is open in the Inspector window.

Procedure

To configure the cycle and the duration of a recording, proceed as follows:

1. Click the button for the recording time.
2. Select an OB for the recording time (Page 10905).
3. Select a unit for the reduction factor in the drop-down list box for "Record every".
4. Enter the factor for the reduction in the input field for "Record every".
5. Select a unit in the drop-down list box for "Recording duration".
6. Specify the recording duration.
   The following options are available:
   – Enter a value for the duration in the input field for "Recording duration".
   – Activate the "Use max. recording duration" checkbox.

Configuring the trigger conditions

Requirement

- A trace configuration has been created and selected in the "Trace configurations" table of the trace handling.
- The "Recording conditions" area is open in the Inspector window.

"Record immediately" trigger condition

To start the recording immediately, proceed as follows:

1. Select the "Record immediately" entry in the drop-down list for "Trigger mode".
   The input fields for the trigger tag are hidden.

"Trigger on tag" trigger condition

To start the recording depending on a condition, proceed as follows:

1. Select the "Trigger on tag" entry in the drop-down list for "Trigger mode".
2. Select a trigger tag. The following options are available:
   – Click the button for the trigger tag and select a tag.
   – Enter the address or the symbolic name of the tag directly in the input field for the trigger tag.

   A drop-down list with events and input fields is displayed. The display depends on the data type of the tag.
3. Configure the event.
4. Select a unit for the pre-trigger in the drop-down list for "Pre-trigger".
5. In order to record a period before the trigger event, enter a value greater than 0 in the input field for the pre-trigger.

Note

The trigger condition is checked in every cycle irrespective of the setting in "Record every". To reliably identify the trigger, the trigger signal must be present for at least one full cycle.

13.7 Establishing a remote connection with TeleService

13.7.1 Basics of working with TeleService

13.7.1.1 Introduction to TeleService

Introduction

TeleService gives your controller telecommunication capability. You can manage, control and monitor distributed plants centrally by means of remote connections.

Scope of functions

TeleService allows you to use the range of TIA portal functions via a telephone network or an Internet connection by establishing a remote connection to a remote plant. The online connection allows you to edit a remote plant as usual with the TIA Portal.

Advantages

Using TeleService offers the following advantages:

- You can easily access even remote sections of plants and include them in a complete system.
- You can offer rapid help and support in the event of faults in a remote system without having to go there yourself.
- You can employ your resources effectively.
- It significantly reduces costs.
- It can significantly reduce plant downtimes.
- It improves the efficiency of your plant.

See also

TeleService functionality (Page 10918)

13.7.1.2 TeleService functionality

TeleService fields of application

TeleService offers the following the fields of application:

- **Access to remote systems (remote maintenance):**
  You can manage, control, and monitor remote systems centrally by means of remote connections.
  This is possible with an S7-300/400 CPU, an S7-1200 CPU and an S7-1500 CPU and a TS Adapter MPI or a TS Adapter IE.
- **Establishing connections from and to remote systems (PG-AS remote link):**
  You can use PRODAVE MPI V5.0 and newer to establish a remote connection to a remote system, and the communications instruction "PG_DIAL" to establish a remote connection from a remote system.
  This is possible with an S7-300/400 CPU and a TS Adapter MPI.
- **Data exchange between systems (AS-AS remote link):**
  The communications instruction "AS_DIAL" allows two automation systems to exchange process data over the telephone network.
  This is possible with an S7-300/400 CPU and a TS Adapter MPI.

- **Sending an text message from a system:**
  An automation system can send a message (SMS) via a GSM wireless modem using the communications instruction "SMS_SEND" ".
  This is possible with an S7-300/400 CPU and a TS Adapter MPI.
- **Sending an email from a system**
  An automation system can also send an email with the following communications instructions and a TS Adapter IE .
  - S7-300/400 CPUs (CPU S7-31x-2PN/DP or CPU 41x-3PN/D) use the instruction "AS_MAIL"
  - S7-1200 CPUs use the instruction "TM_MAIL"
  - S7-1500 CPUs use the instruction "TMAIL_C"

13.7.1.3 Telephone book at TeleService

Introduction

By double clicking the "phone book" folder in the project tree you open the phone book editor displaying the TeleService phone book.

Each version of the TIA Portal has its own "global phone book". If a global phone book from an earlier version of the TIA Portal is found in a more recent version of the TIA Portal, you will be asked once if you want to import this phone book.

This gives you the advantage of being able to import the system data from the previous version into the more recent version of the TIA Portal.

Global phone book properties

The global phone book is used in TeleService to manage specific system data that are required to establish a remote connection.

When you open the phone book for the first time, you will see an empty phone book with all available columns; in all other cases, the last phone book edited is displayed.

You can enter any number of systems in a phone book. Systems contain the data required for establishing a remote connection, for example, the name and location of the device and the phone number to be dialed, along with all country-specific details. With VPN connections, you can enter an IP address or a DNS name instead of the phone number.

The TS adapters used for establishing the connection are distinguished by color, depending on whether a TS Adapter MPI or a TS Adapter IE is used for establishing the connection.

See also

Working with the phone book (Page 10920)

13.7.2 Working with the phone book

13.7.2.1 Basics on working with the phone book

Working with the phone book

You have the following options when working with a phone book:

- Open phone book
- Save phone book
- Import phone book data
- Export phone book data
- Printing the phone book
- Use phone book data to establish a remote connection

You can implement these functions simply and easily by using the buttons in the phone book toolbar.

Note

Access to phone books

The phonebook is user specific in TeleService. However, it is not possible to access the global phone book with more than one instance of the TIA portal at the same time.

See also

Open phone book (Page 10922)

Saving phone book (Page 10923)

Exporting phone book data (Page 10925)

Printing the phone book (Page 10926)

Structure of the phone book (Page 10920)

13.7.2.2 Structure of the phone book

Introduction

A global phone book in TeleService is used to manage the data you require for establishing a remote connection. Once you have created the connection data and saved it in the phone book, you can access it each time you want to establish a remote connection.

Structure of the phone book

The integrated global phone book in TeleService contains the following columns:

Column name	Meaning
System name	Enter a name for your system.
Adapter type	Select the TS Adapter type used in the drop-down list: TS Adapter MPI or TS Adapter IE.
Connection type	Select the required connection type: Dial-up connection or VPN connection.
Area code	Enter the required area code. This column is only active with dial-up connections. This column cannot be edited for VPN connections.
Phone number/remote address	Enter the required connection data for establishing the remote connection. For dial-up connections, this can be a phone number, and for VPN connections a DNS name or an IP address.
Fingerprint	Enter the corresponding fingerprint for establishing a VPN connection to the TS Adapter IE Advanced.
Country	Enter the country code. This column is only active with dial-up connections. This column cannot be edited for VPN connections.
User name	Enter the user name you have logged on under.
Password	Enter the password for this user name.
Group	Enter the group if you have carried out grouping.
Company	Enter the company to be contacted.
Department	Enter the relevant department.
Street	Enter the street.
Town/City	Enter the town or city to which the remote connection is to be established.
Comment	Enter a comment if required.

Showing or hiding columns

You can show or hide the individual columns. To do so, select the required column header and open the shortcut menu with the right mouse button.

13.7.2.3 Symbols in the phone book

Meaning of the TeleService icons

The following table shows the meaning of the TeleService icons:

Symbol	Meaning
	Open the global phone book
	Imports a phone book
	Exports a phone book

Symbol	Meaning
	Establishes a remote connection
	Terminates the active remote connection
	Establishes a remote connection or terminates it
	Displays the connection to a TS Adapter IE in the phone book
	Displays the connection to a TS Adapter MPI in the phone book
	Adds a new row to the phone book
	Inserts a new row in the phone book

13.7.2.4 Manage phone book

Open phone book

Opening phone books

To open the phone book, follow these steps:

1. In the project tree, double-click on the "Phone book" folder under "Online access" > "TeleService".
2. The phone book opens so that you can enter or edit the required system data.

Inserting rows in the phone book

Inserting rows in the phone book

To insert a new row in the phone book, follow these steps:

1. Select the row in front of which you want to insert a new row.
2. Click the "Insert row" button in the toolbar.

Result

A new row is inserted in the phone book above the selected row.

Showing and hiding columns in the phone book

Showing and hiding columns

To show or hide columns in the phone book, follow these steps:

1. Click a column header.
2. In the shortcut menu, select the "Show/hide columns" command.
   The selection of available columns is displayed.
3. To show a column, select the column's check box.
4. To hide a column, clear the column's check box.

Result

The respective columns are shown or hidden when displaying the phone book.

Saving phone book

Saving phone books

When you exit the phone book editor or leave the TIA portal, you are asked whether you want to save the global phone book.

Click "Yes" to save the phone book.

Importing phone book data

Introduction

It is possible to import the phone book data from an external file or from an older version of the TIA Portal.

Requirement

You have already created an import-capable phone book file.

You have already created a phone book with an older version of the TIA Portal.

Importing phone book data from a phone book file

To import phone book data from a phone book file, follow these steps:

1. Open the "TeleService" folder under "Online access" in the project tree.
2. Double-click on the "Phone book" folder.
3. Click the "Import" button in the toolbar.

4. Confirm the prompt asking if you want to save the current state of the phone book with "Yes", if applicable, and specify the location for storing the phone book in the dialog that follows.
5. If you do not want to save the current state of the phone book, answer the prompt with "No". In the subsequent dialog, select the phone book file to which the current phone book should be stored.
6. Close the dialog box with "OK".

Result

The imported phone book data is displayed in the global phone book.

Note

Defining dialing rules

Dialing rules must be defined before the "area code" and "country" columns in the imported phone book can be edited for dial-up connections.

To define dialing rules, follow the link in the history.

Importing phone book data from an older version of the TIA Portal

To import phone book data from an older version of the TIA Portal, follow these steps:

1. Open the "TeleService" folder under "Online access" in the project tree.
2. Double-click on the "Phone book" folder.
3. Click the "Import" button in the toolbar.
4. Confirm the prompt asking if you want to save the current state of the phone book with "Yes", if applicable, and specify the location for storing the phone book in the dialog that follows.
5. If you do not want to save the current state of the phone book, answer the prompt with "No".
6. Enter the following path in the dialog that follows: "%appdata%\siemens\automation \TeleService\GlobalTeleServicePhoneBook.tel".
7. Close the dialog box with "OK".

Result

The phone book data imported from the older version of the TIA Portal is displayed in the global phone book.

Note

Defining dialing rules

Dialing rules must be defined before the "area code" and "country" columns in the imported phone book can be edited for dial-up connections.

To define dialing rules, follow the link in the history.

See also

Defining dialing rules (Page 10926)

Exporting phone book data

Introduction

It is possible to export phone book data to an external file.

Requirement

You have already created a phone book under TeleService with the corresponding system data.

Procedure

Proceed as follows to export the phone book data:

1. Open the "TeleService" folder in project tree.
2. Double-click on the "Phone book" folder.
3. Click the "Export" button in the toolbar.
4. In the next dialog box, select where the current phone book is to be exported.
5. Close the dialog box with "OK".

Result

The exported phone book data are saved in the specified export file.

Printing the phone book

Printing phone books

You can print out all or some of the data in a phone book.

Proceed as follows:

1. Open the phone book.
2. Select the **Phone book > Print** menu command or click on the appropriate button in the toolbar. The "Print" dialog will open.
3. Specify whether you wish to print the complete phone book or just part of it and set all other options.
4. Start the print job with "OK".

Result

The phone book data is printed on the default printer. If the printout is more than one page long, an identifier is printed after the page number at the bottom right corner of the page to indicate that there is another page. The last page does not have this symbol, indicating that no more pages are to follow.

Defining dialing rules

Procedure

To define location-specific dialing rules, follow these steps:

1. Open the folder "Online access" in the TIA portal and select the "TeleService" folder.
2. Use the shortcut menu to access the "TeleService" properties.
3. In the dialog that follows, open the "Advanced settings" tab and activate the "Use dialing rules" option.
4. Click the "Adapt" button.
5. In the dialog that follows, enter the required location information for connection establishment.
   - For new locations, enter the country or region and the corresponding area code.
   - If required, set the correct destination code and pre-dial line for local/long-distance calls.
   - Select your chosen dialing mode for the location.
6. Exit the dialog by clicking "OK".
7. In the next dialog, select the location from which you wish to dial, and click "OK".

8. In the TeleService dialog that follows, check that the correct location appears under "Location".
9. Confirm your entries by clicking "OK".

Note

Modem operation on a local loop or extension

You do not need to specify a pre-dial line if you operate your modem on a local loop (main telephone line). The fields for the pre-dial lines for local calls and long-distance calls must be empty.
If you operate your modem on an extension, you should enter the pre-dial lines which need to be called to access a local loop.

Example of the use of dialing rules

The following example illustrates the use of location-specific dialing rules for the city of Karlsruhe in Germany.

- In the TeleService phone book, you enter the phone number "1234567", without the area code and without the country code.
  Result:
  The phone number "1234567" is always dialed, irrespective of the location of the caller.
- In addition to the phone number "1234567", you enter the location-specific dial parameters for the Karlsruhe area code "0721" and the country code "+49" for Germany in the TeleService phone book.
  Result:
  "1234567" is dialed from Karlsruhe.
  "07211234567" is dialed from other towns in Germany.
  "00497211234567" is dialed from other countries.

Note

Display in the phone book

The "area code" and "country" columns in the TeleService phone book can only be edited once you have defined the dialing rules as detailed above.

13.7.3 Remote connections as dial-up connections

13.7.3.1 Basics for establishing a dial-up connection

Using a TS Adapter for dial-up connections

A TS Adapter is needed for establishing a remote dial-up connection using TeleService.

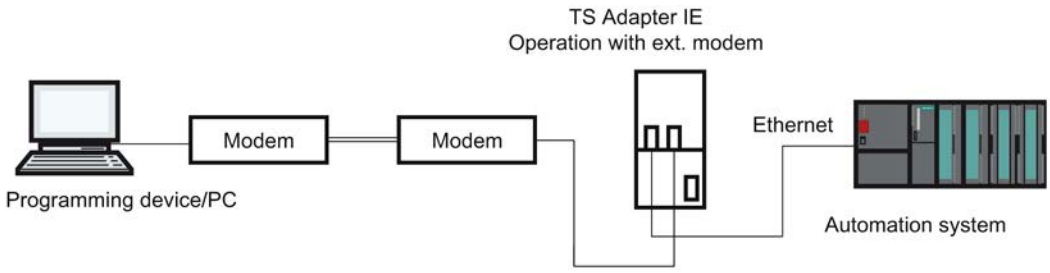
The TS Adapter is used to prepare an automation system for use with TeleService by connecting it to a telephone network via a modem. The TS Adapter has an integrated parameter memory in which a parameter set for TeleService operation is stored.

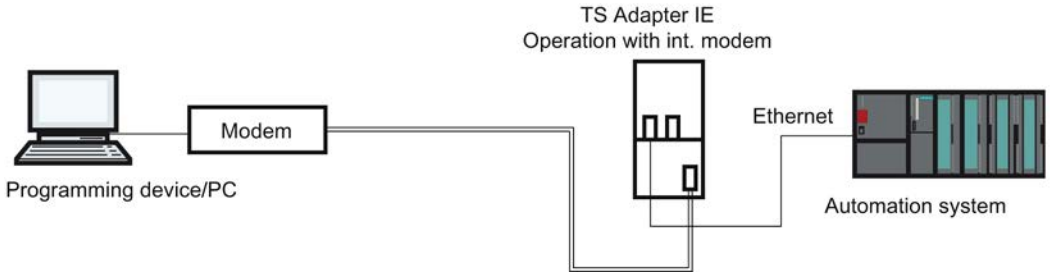
With the function "export adapter parameter", different parameter sets can be saved in external files, and reloaded in the TS Adapter via the function "import adapter parameter".

Establishing a remote dial-up connection

You can choose between a number of different TS Adapters, each of which offers a different functionality and different connection options.

The figure below shows two possible configurations for establishing a dial-up connection to a system with the TS Adapter IE.





Overview of possible TS Adapter:

The TS Adapter comes in the following versions:

- TS Adapter II (also designated "TS Adapter MPI")
- TS Adapter IE Standard (also designated "TS Adapter IE")
- TS Adapter IE Basic (also designated "TS Adapter IE")

Designation "TS Adapter"

In the following pages, the designation "TS Adapter" stands for all versions. The relevant product designation is listed beside information which only applies to a specific version, for example, "TS Adapter II" or "TS Adapter IE Standard".

Note

For more detailed information on your TS Adapter, please refer to the documentation supplied with the TS Adapter.

See also

Short description of the TS adapter MPI (Page 10937)

Short description of the TS adapter IE (Page 10944)

Exporting adapter parameters (Page 10943)

Importing adapter parameters (Page 10944)

13.7.3.2 Telephone networks and modems

Supported telephone networks and modems

Telephone networks which can be used

TeleService can be used with digital networks (ISDN), analog networks and wireless networks (with GSM technology). This version supports a remote connection to a TS Adapter.

Modem support

TeleService has been implemented to be independent of the modem. This means that all standard modems which can be installed in the Windows Control Panel and are visible as modems can also be used by TeleService.

The choice of modem type is determined primarily by the existing hardware of the programming device/PC and the telephone network to be used.

The following modem types/media are supported:

- Modems (external modems at the COM interface, internal modems and PCMCIA cards)
- External ISDN adapter at the COM interface or USB interface
- Internal ISDN adapter with virtual COM interface (for example AVM CAPI port)
- External ISDN modems (ISDN adapter with integrated analog modem functionality) at the COM interface or USB interface
- Radio network modems with GSM technology, PCMCIA adapter card or data cable and mobile phone

Gateways

Gateways between the various telephone networks are in principle possible. Remote connections from an ISDN adapter to an analog modem and vice versa only function with special ISDN telephone adapters.

Performance in telephone networks

The data throughput of a remote connection depends on the modem and telephone network used and the quality of the telephone line.

Installing the local modem

Introduction

If you have already installed a modem for data transfer in your operating system, this modem can also be used for TeleService.

If a modem has not yet been installed for your operating system, one must be installed before you can establish a remote connection with TeleService.

Procedure

Proceed as follows:

1. Make sure your programming device/PC and the modem are switched off.
2. Physically connect the external modem to a COM or USB interface on your programming device/PC. You can also install an internal modem or a PCMCIA card in accordance with the manufacturer's specifications.
3. Now switch on the modem and then the programming device or the PC.

Result

Plug-and-play modems are recognized and installed automatically by the operating system. Dialogs will take you through the installation procedure.

Note

Modems without plug-and-play

If your modem is not recognized automatically when switched on, you will have to install it yourself using the Control Panel.

Please refer to the information in the documentation supplied with your modem.

Connecting and configuring the remote modem

Introduction

A modem must also be connected to the remote system before you can work with TeleService. This modem is designated the "remote modem".

Configuring the remote modem

The modem receives all parameters required for operation from the TS Adapter connected. These include data for initializing the modem and settings for serial transmission between the TS Adapter and the modem.

The data required for the remote modem is specified during the configuration of the TS Adapter.

The modem may be internal or external depending on the TS Adapter used.

How to connect a TS Adapter with an internal modem

1. Switch off the TS Adapter.
2. Connect the TS Adapter to the automation system.
3. Connect the TS Adapter to the telephone line.
4. Switch on the TS Adapter.

How to connect a TS Adapter with an external modem

1. Switch off the modem.
2. Connect the TS Adapter to the automation system.
3. Connect the TS Adapter to the modem using a modem cable.
4. Connect the modem to the telephone line.
5. Switch on the modem.
6. Switch on the TS Adapter.

Note

Please note the following information on configuring the remote modem:

- The default parameters for the modem and the serial port set in the TS Adapter should in most cases ensure successful operation; changes in the parameter assignment will only be needed in rare cases.
- You need only change the parameter assignment of the TS Adapter if a modem connection is not established or if factory settings are to be adapted or optimized.
- TS Adapter parameter assignment can be changed via either a direct or a remote connection.

13.7.3.3 Access protection for dial-up connections

Access protection information

Introduction

When you assign the parameters for your TS adapter, you can restrict access to the parameters of the TS adapter and access to remote systems.

Scope of access protection

Access protection only exists for remote connections; TS adapter parameter assignment can be accessed at any time in direct connection mode.

Access protection also exists in direct connection for the TS adapter IE.

Access protection information

The TS Adapter MPI is not delivered with access protection activated. There is a default password for the TS adapter IE.

The first user who assigns the parameters for this adapter can therefore activate access protection by defining the password for a user and/or a callback number.

This is a multi-level access protection with several users, each with or without administrator rights. For the TS adapter MPI there is only one adminstrator and no more than two users.

For a modem connection, only an administrator can define the two users and change and, if necessary, delete their settings. Those logged in as users can only change their own passwords and their own callback numbers. However, with the TS adapter MP you can access the process of assigning parameters of the TS adapter in direct connection, without restriction.

Advantages

Access protection offers the following advantages:

- Unauthorized access by persons outside the system is almost impossible.
- The plant operator bears most of the telephone costs.

TeleService callback options

Callback variants

The costs of a telephone connection are normally borne by the caller who establishes the dial-up connection.

TeleService can, however, be used so that after a short initial connection the modem connection is established again in the opposite direction, in other words initiated by the TS Adapter (callback). In this case, the plant operator bears the costs of the callback.

There are two callback variants in TeleService:

1. Callback to a number specified during connection establishment.
2. Callback to a number stored on the TS Adapter.

Levels of protection

Introduction

You can set up one of two possible levels of access protection for TeleService access to the TS Adapter. Different options are available with each protection level.

Access protection options

Access protection level 1:

The TS Adapter is protected by the user name and password. You can access the TS Adapter via any telephone line and specify any callback number during connection establishment.

Access protection level 2:

The TS Adapter is protected by the user name, password, and the callback number. You can only access the TS Adapter from one telephone connection per user.

The table below sets out the above conditions for the various protection levels:

Level of access protection	Administrator/User password	Callback number
1	enter	do not enter
2	enter	enter

Logging on to TS Adapter

When you log on to the TS Adapter and after you have set up access protection, enter your user name, the corresponding password and, if desired, a callback number:

Level of access protection	Administrator/User password	Callback number
1	enter	do not enter or enter any callback number
2	enter	do not enter

If you have entered a callback number during connection establishment (access protection level 1) or stored a callback number in the TS Adapter (access protection level 2), the modem connection will be terminated and the TS Adapter will call back the given number.

Setting up access protection and callback number for the TS adapter

Introduction

During the parameter assignment for the TS adapter MPI in TeleService, you can set up access protection and a callback number for the parameter assignment of the adapter and connection to the remote system. The following describes the parameter assignment for a TS adapter MPI. The parameter assignment of a TS adapter IE is carried out in analog. The specific method is described in the web help of this adapter.

Requirement

A TS Adapter MPI is connected to your computer and is displayed in the project tree under "Accessible nodes".

Procedure

To set up access protection for the TS adapter, proceed as follows:

1. Click on the command "Assign TS Adapter MPI parameters" in the project tree.
2. Open the "Access security" tab.
3. Enter a password for your user name and/or number that you want the modem to call back following logon.
   - If you are an administrator, you can change all the settings for administrators and users, and delete or create users.
   - If you are logged on as a user, you can only change your own settings (password and callback number).
4. Confirm all entries before exiting the dialog with "OK".
5. Click the "Yes" button to confirm the following query.

Result

The parameter assignment for the access protection and the callback number is saved in the non-volatile memory of the TS adapter MPI.

Note

Important points to note when setting up access protection:

- The settings in the "Modem" tab must correspond to the conditions at the plant if callback functionality is to be guaranteed.
- Entering an incorrect callback number in the role of "ADMIN" user will mean you are no longer able to access the TS Adapter MPI over a remote connection!
- Test the callback number before you enter it as the "ADMIN" user by calling the given callback number during connection establishment (access protection level 1).

Complete a callback in TeleService

Callback options

Two different callback variants can be set up in TeleService.

The following callback options are available:

- Callback to a number specified during connection establishment.
- Callback to a number stored on the TS Adapter

Callback to a number specified during connection establishment

1. In the project tree of the TIA portal, open the "Online access" folder.
2. Click on the "TeleService" folder it contains.
3. Double-click the "Set up/close remote connection" entry. The "Set up remote connection to the remote system" dialog opens.
4. In the "Adapter type" drop-down list, select the adapter type used.
5. Select the "dial-up connection" under "Connection type" if it is not already selected.
6. Select the modem you are using under "Local Settings".
7. Enter the phone number to be dialed in the appropriate box or open the phone book by clicking on the button behind it and take the desired phone number from the phone book.
8. Enter your user name and associated password of the TS adapter.
9. If you want a "Connection setup with callback", select the appropriate option button.

10. Click the "Connect" button to establish the required remote connection. This button only becomes active when you have entered all the parameters needed to establish a remote connection. Any remote connection is displayed under "Status".
11. Enter the desired callback number in the dialog that follows.

Result

The remote connection to the desired system is made with callback.

The connected system is shown with the corresponding icon in the project tree.

Note

This procedure is useful if the costs of the modem connection are to be borne by the plant operator and if the actual callback number is not fixed, i.e. callback is not always to the same receiver. It is particularly useful for mobile users.

Callback to a number stored on the TS Adapter

1. Assign the parameters for the desired callback number in the TS adapter.
2. Establish a connection to the TS adapter as described above, and observe the following features:
   - Enter the user name and password for which the callback number parameters are assigned in the TS adapter.
   - The check box "Establish a connection with callback" does not have to be selected, since the callback number is already known by the TS adapter.

Result

Callback to a number stored on the TS adapter has been established. If a remote connection is established, the callback occurs from the remote system.

Note

This procedure offers the highest level of access protection. However, it does pose a risk: if the callback number stored on the TS Adapter is not correct, it will no longer be possible to access the TS Adapter over a modem connection. The device can in such a case only be put back into operation by changing the parameter settings on site.

13.7.3.4 TS adapter MPI

Short description of the TS adapter MPI

TS Adapter MPI:

The designation "TS Adapter MPI" is a collective term for allTS Adapter with an MPI/DP interface.

The TS Adapter MPI comes in the following versions:

- As TS Adapter I (parameters cannot be assigned via the TIA Portal)
- as TS Adapter II

The table below provides a short description of the functionalities. For detailed information on your TS Adapter, please refer to the documentation supplied with your TS Adapter.

TS Adapter II:	
Direct connection	Connection via the Universal Serial Bus (USB). The firmware can be replaced. The modem is integrated or can also be connected as external modem. The TS Adapter II switches automatically between the modems. As long as no external modem is connected, the adapter will use the internal modem.
There are two variants	• With internal analog modem. An external modem can also be connected to the RS232 port. • With internal ISDN adapter. An external modem can also be connected to the RS232 port.

Use of the designation "TS Adapter"

For TeleService the designation "TS Adapter" is the generalization for all versions. The relevant product designation is listed beside information which only applies to a specific version of a TS Adapter, for example, "TS Adapter II", "TS Adapter IE Standard" or "TS Adapter IE Basic".

How the TS adapter MPI works

How the TS Adapter MPI Works

In line with the configuration, the TS Adapter MPI connects the serial port or USB port of your programming device/personal computer (direct connection) or the serial port of a modem (modem connection) to the MPI/PROFIBUS network of your automation system.

The TS Adapter MPI has a non-volatile memory. Parameters for the following functions are stored in this memory:

- The MPI/PROFIBUS network (network parameters)
- The mode of the modem used

- The serial port to the modem
- Access protection

Default parameter assignment

The TS Adapter comes with default parameter assignment. The parameters can be set and saved to the non-volatile memory of the TS adapter in a parameter assignment session.

When "Direct connection" is configured, the TS Adapter will only use the network parameters for access to the MPI/PROFIBUS network.

In the "Modem connection" configuration, all the parameters stored on the TS Adapter will be activated.

Note

For more detailed information on the configuration of your TS Adapter, please refer to the documentation supplied with it.

Operating a TS adapter MPI in direct connection mode

Direct connection with TS Adapter MPI

The direct connection is used to assign the parameters of the TS Adapter MPI. The same configuration also allows you to go online in the TIA Portal and thereby check the configured MPI/PROFIBUS parameters for bus compatibility. This means that (as with a PC adapter) SIMATIC S7/C7 systems can be accessed via the MPI/DP interface without an MPI/PROFIBUS module occupying a slot for a programming device/PC.

Access protection for the TS Adapter is not active in direct connection configuration. This means that the parameter assignment of the TS Adapter can be changed without any problems, for example by importing adapter parameters.

Note

Display of the TS Adapter MPI in the TIA Portal

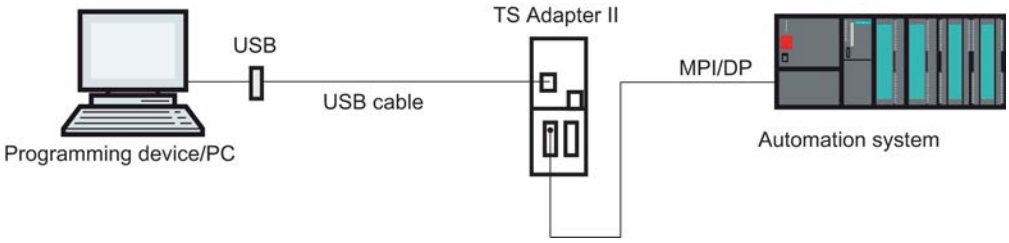
As soon as you have connected a TS Adapter MPI with the PG/PC using the USB port, the "TS Adapter" folder is displayed in the project tree of the TIA Portal.

When you open the folder, you can assign the parameters of the connected TS Adapter MPI via the following dialog.

Establishing the direct connection for TS Adapter MPI

Direct connection mode means there is a direct connection via the TS Adapter MPI between the programming device/personal computer on which TeleService is installed and the automation system. No modem is required.

The figure below shows the configuration of the TS Adapter MPI with a direct connection.



Operating a TS adapter MPI in modem connection mode

Introduction to the modem connection with TS Adapter MPI

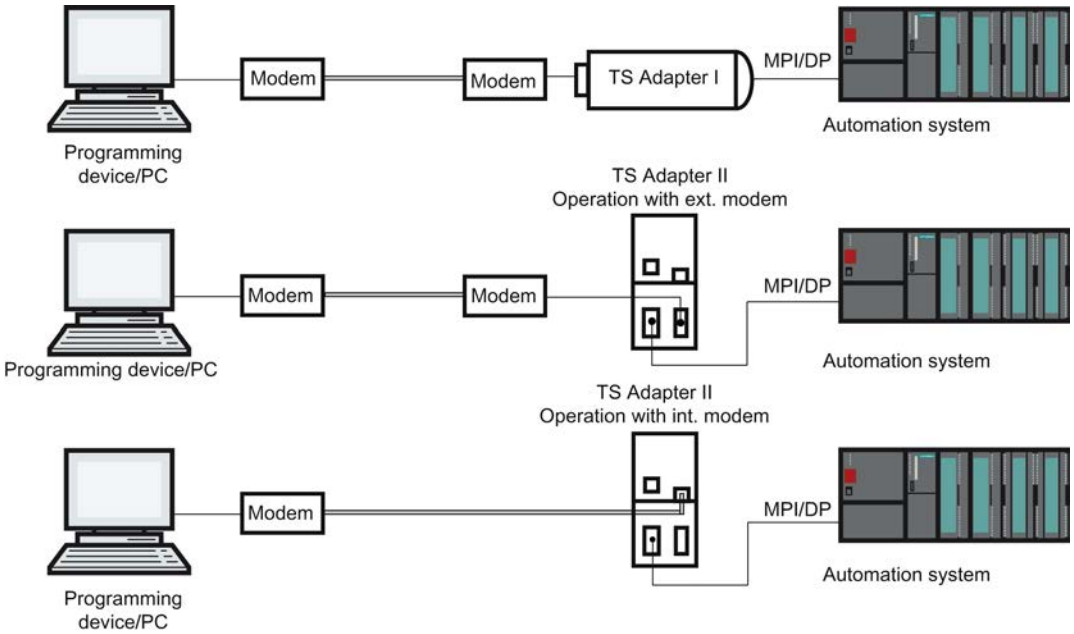
This configuration allows you to dial into a remote system. To do this, you establish a remote connection to a remote system using TeleService on a telephone network. Over the established modem connection, you can then work with the selected plant as usual with the TIA Portal.

Establishing a modem connection with TS Adapter MPI

This connection between the programming device or PC on which TeleService is installed and the automation system in which the TS Adapter MPI is inserted in the MPI/DP interface is made through a modem route.

The configuration therefore includes the programming device or PC via the telephone network and the TS Adapter MPI on the MPI/DP interface of the automation system.

The figure below shows the structure of the modem connection.



Note

Parallel operation between direct and modem connection

The TS Adapter II has two connections for communication with PG/PC, both of which can be connected at the same time. At the same time, connect the USB interface with the PG/PC and the modem interface with the telephone network.

In this configuration you can either use the direct or the modem connection.

A parallel operation is **not** possible!

TS adapter MPI configuration options

Useful information on configuring the TS Adapter MPI

The TS Adapter MPI can be configured in both direct connection mode and via an existing remote connection.

The following parameter assignment options are available:

- Reconfiguration (Page 10941)
- Restoring default parameter assignment (Page 10942)
- Importing adapter parameters (Page 10944)
- Exporting adapter parameters (Page 10943)
- Setting up access protection (Page 10934)

Parameter assignment

Configure your TS Adapter in accordance with the documentation supplied with the TS Adapter. It will detail the exact procedure for parameter assignment.

Note

Please note the following when configuring the TS Adapter MPI

- If you change the current parameter settings when there is an established remote connection, there is a risk it will not subsequently be possible to establish a modem connection with the modified parameters. The TS Adapter MPI can in this case only be configured in direct connection mode.
- This means that either parameter assignment must be carried out with a programming device/personal computer at the plant location or the TS Adapter MPI must be brought to the location of the local programming device/personal computer in order to be configured.

Positive acknowledgement

During parameter assignment, the data is written to the non-volatile memory of the TS Adapter MPI. The parameter assignment process is not acknowledged positively until all precautions have been taken to ensure that parameter changes have been carried out correctly and will thus survive a power failure.

Changes become effective for the TS Adapter MPI as follows:

- The serial parameters, the modem parameters and the parameters for access protection are activated once the remote connection has been terminated.
- The modified network parameters are activated immediately.

Configuring TS adapter MPI

Introduction

The TS Adapter MPI can be configured in both direct connection mode and via an existing remote connection in modem connection mode.

The following describes the method for assigning parameters.

Requirement

A TS Adapter MPI is connected to your computer and the folder "TS Adapter" is displayed in the project tree under "Online access".

Procedure

To assign the parameters for the TS Adapter MPI im Direktanschluss please proceed as follows:

1. Double-click the "Online access" folder in the project tree.
2. Open the required folder:
   - The "TS Adapter" folder for a direct connection.
   - For an existing remote connection, the "TeleService" folder followed by the folder with the required system name.
3. Select the command "Assign TS Adapter MPI parameters". The "Assign TS Adapter MPI parameters" dialog opens.
4. Set the required parameters in the individual tabs of the dialog.
5. Confirm your entries with "OK".

Result

The configured parameters are saved in the non-volatile memory of the TS Adapter MPI. Parameter assignment is then complete.

Restoring default parameter assignment for TS adapter MPI

Introduction

You can restore the default, factory state parameters of the TS Adapter MPI.

Requirement

A TS Adapter MPI is connected to your computer and is displayed in the project tree under "Online access" in the "TeleService" folder.

Procedure

Proceed as follows to restore default parameters for the TS Adapter MPI:

1. Open the "TeleService" folder in project tree.
2. Double-click the "TS Adapter MPI" folder.
3. Select the command "Assign TS Adapter MPI parameters". The "Assign TS Adapter MPI parameters" dialog opens.
4. Click the "Reset" button under "General".
5. Confirm your entries with "OK".

Result

The TS Adapter MPI default parameters set on delivery are restored.

See also

TS adapter MPI configuration options (Page 10940)

Exporting adapter parameters

Introduction

You can export the configuration of a TS Adapter MPI to an external file. The configuration saved in this file can be imported in turn into any number of TS Adapter MPI.

This can for example be useful if you want to assign identical parameters to multipleTS Adapter MPI or if you want save, document or distribute the parameter set.

Requirement

A TS Adapter MPI is connected to your computer and is displayed in the project tree under "Online access" in the "TeleService" folder.

Procedure

To export the adapter parameters of a TS Adapter MPI:

1. Open the "TeleService" folder in project tree.
2. Double-click the "TS Adapter MPI" folder.
3. Select the command "Assign TS Adapter MPI parameters". The "Assign TS Adapter MPI parameters" dialog opens.
4. Click the "Export" button.
5. A window will open in which you can select the file to which you wish to export the configuration of the TS Adapter MPI.
6. Confirm with "Save".

Result

The parameters of the TS Adapter MPI are saved in the specified file (*.tap). The export of the adapter parameters is now complete.

Importing adapter parameters

Introduction

You can import the configuration of a TS Adapter MPI from a previously created export file (*.tap).

The configuration saved in this file can be imported into any number of TS Adapter. This can for example be useful if you want to assign identical parameters to multiple TS Adapter MPI.

You can import parameters locally in direct connection mode or via an existing remote connection in modem connection mode.

Requirement

A TS Adapter MPI is connected to your computer and is displayed in the project tree under "Online access" in the "TeleService" folder.

Procedure

To import the adapter parameters of a TS Adapter MPI:

1. Open the "TeleService" folder in project tree.
2. Double-click the "TS Adapter MPI" folder.
3. Select the command "Assign TS Adapter MPI parameters". The "Assign TS Adapter MPI parameters" dialog opens.
4. Click the "Import" button.
5. A dialog will open in which you can select the file to which you wish to import the configuration of the TS Adapter MPI.
6. Confirm the next dialog with "Yes".

Result

The parameters selected are saved in the non-volatile memory of the TS Adapter MPI. Adapter parameter import is then complete.

13.7.3.5 TS adapter IE

Short description of the TS adapter IE

TS Adapter IE

The designation "TS Adapter IE" is a collective term for allTS Adapter with an Ethernet port.

The TS Adapter IE comes in the following versions:

- as TS Adapter IE Standard
- as TS Adapter IE Basic

The tables below provide a short description of the functionalities. For detailed information on your TS Adapter, please refer to the documentation supplied with it.

TS Adapter IE Standard:
Direct connection by means of Industrial Ethernet (IE). Firmware update possible. Modem integrated or external. The TS Adapter IE cannot automatically switch between modems like the TS Adapter II. Parameters are assigned via a Web interface.
There are 2 variants:
• With internal analog modem. An external modem can also be connected to the RS232 port.
• With internal ISDN adapter. An external modem can also be connected to the RS232 port.

TS Adapter IE Basic:
Direct connection by means of Industrial Ethernet (IE). Firmware update possible. Plug-in modules. Parameters are assigned via a Web interface.
There are 4 variants:
• TS Adapter IE Basic MODEM: Basic device TS Adapter IE Basic with TS Module MODEM for operation on the analog telephone network.
• TS Adapter IE Basic ISDN: Basic device TS Adapter IE Basic with TS Module ISDN for operation on ISDN telephone systems.
• TS Adapter IE Basic GSM: Basic device TS Adapter IE Basic with TS Module GSM for operation on the GSM radio network.
• TS Adapter IE Basic RS232: Basic device TS Adapter IE Basic with TS Module RS232 for connecting an external modem.

Use of the designation "TS Adapter"

"TS Adapter" is used in the TeleService online help as a general designation for all versions. The relevant product designation is listed beside information which only applies to a specific version of a TS Adapter, e.g. "TS Adapter I", "TS Adapter II", "TS Adapter IE Standard" or "TS Adapter IE Basic".

How the TS adapter IE works

How the TS Adapter IE works

The TS Adapter IE connects the telephone network or the serial port of a modem with the Industrial Ethernet of your automation system.

The TS Adapter IE has a non-volatile memory. Parameters for the following functions are stored in this memory:

- The mode of the modem used
- The serial port to the modem
- Access protection

Default parameter assignment

The TS Adapter IE comes with default parameter assignment. The parameters can be set and saved to the non-volatile memory of the TS adapter in a parameter assignment session.

Note

For more detailed information on the configuration of your TS Adapter, please refer to the documentation supplied with it.

Connection Types

Connection types of the TS Adapter IE Basic

The following diagrams show the connection types possible with the TS Adapter IE Basic.

Direct connection

In the direct connection to the PG/PC, you can set the TS Adapter IE Basic through Ethernet.

Note

The operation of the TS Adapter IE Basic without a TS module is not permitted.

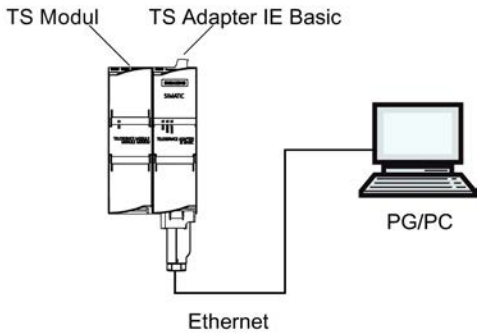


Figure 13-1 Direct connection

Connection to the telephone network

In order to have a direct connection to the telephone network, you must connect the TS Adapter IE Basic together with one of the following TS modules:

- TS Module Modem
- TS Module ISDN

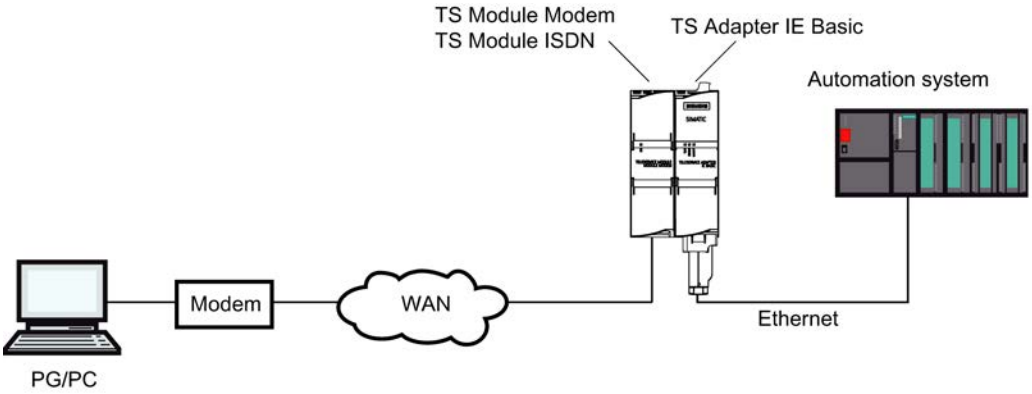


Figure 13-2 Direct connection to the telephone network

More information about the TS modules can be found in the *TS Adapter modular* manual.

Connection to the GSM network

In order to connect to the GSM network, you must operate the TS Adapter IE Basic together with this TS module:

- TS Module GSM

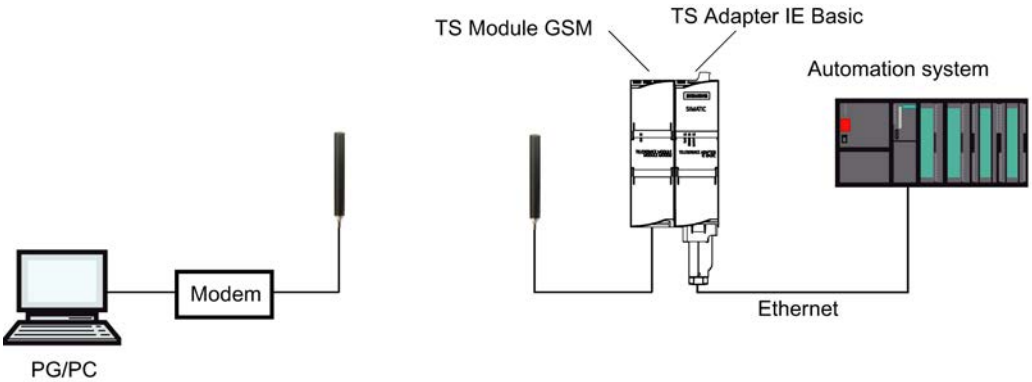


Figure 13-3 Connection to the GSM network

More information about the TS modules can be found in the *TS Adapter modular* manual.

Connection to the telephone network through an external modem

For the connection to an external modem, you must operate the TS Adapter IE Basic together with this module:

- TS Module RS232

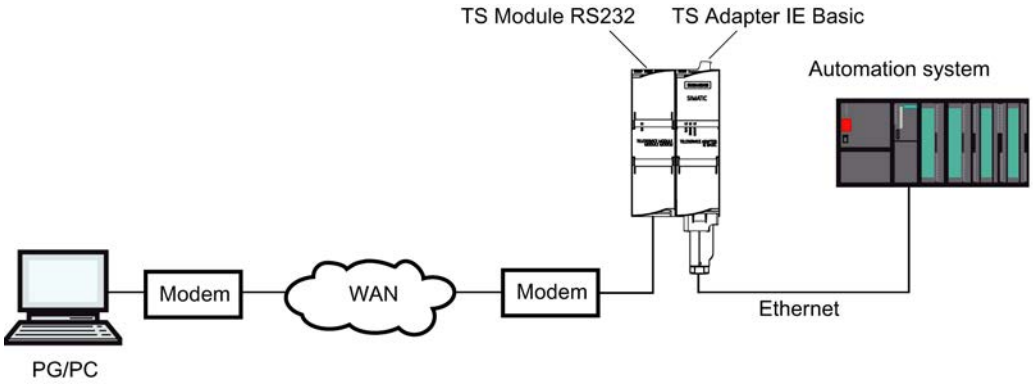


Figure 13-4 Connection to an external modem

More information about the TS modules can be found in the *TS Adapter modular* manual.

TS Adapter IE parameter assignment options

Useful information for configuring the TS Adapter IE

The TS Adapter IE is configured via a Web interface.

Web help associated with the parameter assignment interface is made available for the configuring the TS Adapter IE.

The following parameter assignment options are available:

- Reconfiguration
- Restoring default parameter assignment
- Importing adapter parameters
- Exporting adapter parameters

Note

Parameter assignment

Configure your TS Adapter in accordance with the documentation supplied with the TS Adapter. It will detail the exact procedure for parameter assignments.

Parameter assignment for TS Adapter IE

Introduction

The TS Adapter IE can be configured in both direct connection mode and via an existing remote connection in modem connection mode.

Both parameter assignment options are described below.

Specific details for assigning parameters of the TS Adapter IE can be obtained from the TS Adapter IE documentation.

Parameter assignment of the TS Adapter IE in direct connection

Requirement

There is a LAN connection to your TS Adapter IE .

The TS Adapter IE Basic is connected to the power supply.

Procedure

To assign the parameters for the TS Adapter IE , proceed as follows:

1. In the project tree of the TIA portal, open the "Online access" folder.
2. Double-click on the Ethernet port of your computer.
3. Double-click on the "Display accessible nodes" command. The TS Adapter IE is then displayed.
4. Double-click on the <TS Adapter IE> folder and then on "Online and diagnostic", and assign the desired IP address to the TS Adapter IE in the following dialogs. Please note that the IP address of the PG/PC interface card is located in the same subnet as the IP address that you issue for the TS Adapter IE.
5. Update the view in the project tree for the "Accessible nodes", so that the TS Adapter IE is displayed with the newly allocated IP address.
6. Open the folder <TS Adapter IE> in the device list.
7. Double-click the command "Assign TS Adapter IE parameters". The allocated web interface opens for assigning the TS Adapters IE parameters.
8. Complete the "logon" for the web interface.
9. Set the required parameters in the individual tabs of the dialog.
10. Confirm your entries with "Save settings".

Result

The configured parameters are saved in the non-volatile memory of the TS Adapter IE. Parameter assignment is then complete.

Parameter assignment of the TS Adapter IE using a remote connection

Requirement

There is an established remote connection to a TS Adapter IE .

Procedure

1. In the project tree of the TIA portal, open the "Online access" folder.
2. Open the "TeleService" folder followed by the required system folder.
3. Double-click the command "Assign TS Adapter IE parameters". The associated web interface for assignment of the TS Adapter IE parameters opens. The "logon" for the web interface takes place automatically with the login data of the remote connection.
4. Set the required parameters in the individual tabs of the dialog.
5. Confirm your entries with "Save settings".

Result

The configured parameters are saved in the non-volatile memory of the TS Adapter IE. Parameter assignment is then complete.

13.7.3.6 Establishing a dial-up connection to a remote system

Establishing a dial-up connection

Introduction to establishing a remote dial-up connection

A dial-up connection is established when you use TeleService to dial into a remote system via a telephone network. The programming device/personal computer is connected to the telephone network with TeleService via a modem. At the other end, the automation system is connected to the telephone line via a configured TS Adapter and a modem.

Requirements

A local modem is installed and configured.

The TS Adapter is located in the remote system.

A remote modem is installed and parameters have been assigned.

Proceed as follows:

1. In the project tree of the TIA portal, open the "Online access" folder.
2. Click on the "TeleService" folder it contains.
3. Double-click the "Set up/close remote connection" entry. The "Set up remote connection to the remote system" dialog opens.
4. In the "Adapter type" drop-down list, select the adapter type used.
5. Under "Connection type", select "Dial-up connection".
6. Select the modem you are using under "Local settings".
7. Enter the phone number to be dialed in the appropriate box or open the phone book by clicking on the button behind it and take the desired phone number from the phone book.
8. Enter your user name and the corresponding password.
9. If you want a "Connection setup with callback", select the appropriate option button.
10. Click the "Connect" button to establish the required remote connection. This button only becomes active when you have entered all the parameters needed to establish a remote connection. Any remote connection is displayed under "Status".

Result

The dial-up connection to the desired system is established.

The dialog closes as soon as the dial-up connection is established. The following message appears in the TIA Portal status bar: "Remote connection is established". You can now use the remote connection with TIA Portal and communicate with the automation system.

Connection cannot be established

If the connection cannot be established, try to find the cause using the "Notes on troubleshooting".

Terminating the connection

Once you have finished editing the remote system, exit the remote connection in the project tree by double-clicking on the entry "Establish/disconnect remote connection".

By exiting the TIA Portal you are also terminating the remote connection.

Terminating a dial-up connection

Terminating an active dial-up connection

Note

Terminating the dial-up connection

You should go offline in the TIA Portal before you terminate the remote connection.

Proceed as follows:

1. Double-click the "Set up/close remote connection" entry in the TIA portal.
2. Confirm the prompt in the dialog that follows with "Yes".

Result

The connection is terminated.

13.7.4 Remote VPN connections

13.7.4.1 Basics for establishing a VPN connection

Using a TS Adapter IE Advanced for VPN connections

A TS Adapter IE Advanced is required to establish a VPN connection using TeleService.

VPN definition

VPN stands for "Virtual Private Network". This is a private network of computers based on a public network infrastructure.

VPN is used to create a connection that is as secure as possible between devices in a private network at different locations and the gateway of another network.

A VPN device has direct access to the gateway in the other network over an encrypted connection.

VPN connections

When a VPN connection is established with TeleService, a CA certificate and a unique fingerprint generated from it are used for the unique identification of the TS Adapter IE Advanced. The VPN connection allows a communication that is secure from tapping and manipulation between the remote PC and the TS Adapter.

If you want to establish a VPN connection to multiple remote networks, you will require a separate TS Adapter IE Advanced as an interface for each network. The networks connected by the TS Adapter can used identical IP addresses internally. Only the external IP addresses (WAN) of the TS Adapters need to be different.

Only one VPN connection to a TS Adapter is possible at any one time.

Note

Authentication

Please not that the CA certificate exclusively authenticates the TS Adapter IE Advanced.

As in the case of dial-up connections, users are authenticated via the login (user name and password).

Therefore, use only a secure password for the login.

See also

- Establishing VPN connections (Page 10958)
- Terminating VPN connections (Page 10960)

13.7.4.2 Basics of CA certificates

Introduction

To establish a secure VPN connection with TeleService, you need to generate a CA certificate with a unique fingerprint when you configure the TS Adapter IE Advanced.

You can install this CA certificate as follows on each PC which is to access the TS Adapter IE Advanced:

- Using the automatic download in the TeleService connection dialog, by entering the fingerprint for the CA certificate
- Manually by using the Microsoft® Management Console.

Definition of CA certificate

A CA certificate is a digital certificate issued by a "certificate authority" or "certification authority", hereinafter referred to as "CA". In the case of the TS Adapter IE Advanced, self-signed certificates are used; the certification authority in this case is the TS Adapter IE Advanced itself.

Certificates for "SSTP" (Secure Socket Tunneling Protocol) and "HTTPS" (Hypertext Transfer Protocol Secure) are derived from the CA certificate.

CA certificates contain a "key" and additional information used for authentication and for encrypting and decrypting confidential data. Additional information includes the validity period, references to certificate revocation lists, etc. which are included in the certificate by the CA.

Using CA certificates with TeleService

A CA certificate with a unique fingerprint is generated by the TS Adapter to uniquely identify the TS Adapter IE Advanced as connection partner for the remote PC.

This CA certificate must be saved in the Windows certificate store of your programming device/ PC before you can establish a VPN connection. When you access the Web server using a direct connection, a security warning will appear if no CA certificate exists. In such cases, however, you can choose to ignore the warning and allow the connection.

Note

Handling CA certificates

Specialist knowledge of the operating system is required for handling CA certificates. CA certificates should only be managed by trained personnel!

You require administrator rights to manage CA certificates.

Definition of fingerprint

The fingerprint is a 20-byte hexadecimal expression. It provides a unique value for a CA certificate and is used to identify a specific CA certificate.

A fingerprint is calculated dynamically using the SHA-1 algorithm and is not physically contained in the CA certificate.

Use of fingerprints with TeleService

The CA certificate is used to uniquely identify the TS Adapter IE Advanced as connection partner. A unique 20-byte fingerprint of this certificate is generated each time a CA certificate is generated by the TS Adapter IE Advanced. It is calculated automatically by the TS Adapter IE Advanced when the certificate is generated. Each certificate has a specific, unique fingerprint. This fingerprint must be transferred securely to your computer, for example by phone or in an encrypted e-mail. You need to enter this fingerprint in the connection dialog when establishing a VPN connection with TeleService, unless the CA certificate is already saved on your PC in the Windows certificate store.

You will find the fingerprint for your TS Adapter IE Advanced in the Web interface for the TS Adapter IE Advanced. To open the Web interface, double-click the command "Assign TS Adapter IE Advanced parameters" in the device list in the TIA portal. Carry out the Web login to view the fingerprint in the "Security > Certificates" tab.

CA certificate download during connection establishment

When the connection is established, TeleService checks whether a suitable CA certificate is installed in the Windows certificate store of your programming device/PC. If an appropriate certificate is found, the VPN connection is established as an SSTP connection (Secure Socket Tunneling Protocol).

If no appropriate CA certificate is found, the CA certificate is first loaded by the corresponding TS Adapter IE Advanced. The TS Adapter IE Advanced is called by the remote address entered in the connection dialog. If this download of the CA certificate is successful, the fingerprint of the downloaded CA certificate is calculated and compared with the fingerprint entered in the connection dialog. If the two fingerprints match, a dialog opens and you are prompted to save the CA certificate in the Windows certificate store of your programming device/PC. You require administrator rights to save the CA certificate.

The VPN connection is then established.

See also

- Installing CA certificates for VPN connections (Page 10955)
- Deleting CA certificates for VPN connections (Page 10958)

13.7.4.3 Installing CA certificates for VPN connections

Installing CA certificates

To establish a secure VPN connection between your programming device/PC and a remote system with TeleService, you require a valid CA certificate generated by the TS Adapter IE Advanced. This must be saved in the Windows certificate store of your programming device/PC.

A CA certificate can be installed manually or using an automatic download.

CA certificates are managed with the Microsoft® Management Console.

Note

Handling CA certificates

Specialist knowledge of the operating system is required for handling CA certificates. CA certificates should only be managed by trained personnel!

You require administrator rights to manage CA certificates!

Requirement

A CA certificate has yet to be installed in the Windows certificate store on your computer.

Installing CA certificates using the automatic download

Proceed as follows:

1. Log on to the system as administrator.
2. Transfer the fingerprint for the CA certificate from the TS Adapter IE Advanced to your computer using a "secure route", for example by phone or in an encrypted e-mail.
   You will find the fingerprint for your TS Adapter IE Advanced in the Web interface for the TS Adapter IE Advanced. To open the Web interface, double-click the command "Assign TS Adapter IE parameters" in the device list in the TIA portal. Perform the Web login to view the fingerprint in the "Security > Certificates" tab.
3. In the project tree of the TIA portal, open the "Online access" folder.
4. Click on the "TeleService" folder it contains.
5. Double-click the "Set up/close remote connection" entry. The "Set up remote connection to the remote system" dialog opens.
6. Select the "TS Adapter IE Advanced" as Adapter type from the drop-down list.
7. Under "Connection type", select "VPN".
8. In the relevant input box, enter the IP address / DNS name for the TS Adapter IE Advanced to be contacted. Alternatively, you can apply any existing data from the phone book by clicking the corresponding button.
9. Enter your user name and the corresponding password.
10. Copy the fingerprint displayed in the Web interface for the TS Adapter IE Advanced to the "Fingerprint" column.
11. Click the "Connect" button to establish the required remote connection. This button only becomes active when you have entered all the parameters needed to establish a remote connection.
12. No valid CA certificate is found, because no CA certificate has been installed on your PC yet.
    A "normal" connection (not a VPN connection) is therefore established and the certificate required is downloaded from the TS Adapter IE Advanced to the working memory of your computer. The fingerprint is then calculated (SHA-1 algorithm) and compared with the fingerprint entered in the connection dialog. If the two fingerprints match, a dialog opens and you are prompted to save the CA certificate in the Windows certificate store of your programming device/PC.
13. Confirm that you want to save the CA certificate.

Result

The VPN connection to the required TS Adapter IE Advanced is established. The dialog closes once the connection is established.

Manual installation of CA certificates

Proceed as follows:

1. Log on to the system as administrator.
2. Open the Windows Certificate Manager on your programming device/PC with the Microsoft® Management Console.
   Click "Start", enter "mmc" in the search field and press ENTER.
   The console will open.
3. Open the "File" menu and click "Add/remove snap-in...".
   The "Add/remove snap-in..." dialog will open.
4. Double-click "Certificates" in the "Snap-in" list and select "Computer account" in the dialog that opens.
5. Select "Local computer" in the next dialog and click "Finish" and "OK".
   The console root opens and the "Certificates (local computer)" folder appears.
6. Open the displayed "Certificates (local computer)" folder and click "Trusted certification authorities".
7. Click the "Certificates " folder and use the shortcut menu to access the command "All tasks" > "Import...".
8. Note the information that appears in the "Certificate - Import wizard" dialog and click "Next".
9. In the dialog that follows, click "Browse ..." and select the required CA certificate.
10. Then click "Next" twice and subsequently "Finish" to install the CA certificate.

Result

The selected CA certificate is installed at the specified location in the Windows certificate memory.

Note

Additional information ...

... on installing CA certificates can be found using the "F1" button in the online help of your operating system.

13.7.4.4 Deleting CA certificates for VPN connections

Deleting CA certificates

Proceed as follows:

1. Log on to the system as administrator.
2. Open the Windows Certificate Manager on your programming device/PC with the Microsoft® Management Console.
   Click "Start", enter "mmc" in the search field and press the ENTER KEY.
   The console will open.
3. Open the "File" menu and click "Add/remove snap-in...".
   The snap-in selection dialog will open.
4. Double-click "Certificates" in the "Snap-in" list and select "Computer account" in the dialog that opens.
5. Select "Local computer" in the next dialog and click "Finish" and "OK".
   The console root opens and the "Certificates (local computer)" folder appears.
6. Open the displayed "Certificates (local computer)" folder and click "Trusted certification authorities".
7. Open the "Certificates" folder, select the required CA certificate and click "Delete" in the shortcut menu.
8. Confirm the next prompt with "Yes".

Result

The selected CA certificate is deleted from the list of available certificates.

13.7.4.5 Establishing a VPN connection to a remote system

Establishing VPN connections

Introduction to establishing VPN connections

A VPN connection is established when you use TeleService to connect to a remove system over the Internet.

To do this, your programming device/PC with installed TIA Portal is connected to the Internet at one end. At the other end, the automation system is connected to the Internet via the WAN (Wide Area Network) interface of the configured TS Adapter IE Advanced.

Requirement

Your programming device/PC is connected to the Internet.

There is a TS Adapter IE Advanced in the remote system.

The TS Adapter IE Advanced has been configured and is connected to the Internet.

The CA certificate required for identifying the TS Adapter has been generated and installed in the Windows certificate store of your programming device/PC.

Proceed as follows:

1. In the project tree of the TIA portal, open the "Online access" folder.
2. Click on the "TeleService" folder it contains.
3. Double-click the "Set up/close remote connection" entry. The "Set up remote connection to the remote system" dialog opens.
4. Select the "TS Adapter IE Advanced" as "Adapter type" from the drop-down list.
5. Enter "VPN" as "connection type".
6. Enter the IP address / DNS name for the TS Adapter IE Advanced to be contacted in the relevant field. Alternatively, you can apply any existing data from the phone book by clicking the corresponding button.
7. Enter your user name and the corresponding password.
8. Click the "Establish" button to establish the required VPN connection. This button only becomes active once you have entered all the parameters needed for establishing the remote connection.

Result

The VPN connection to the required system is established. "Status" indicates the progress in establishing the connection. The dialog closes once the VPN connection is established. The following message appears in the TIA Portal status bar: "Remote connection is established". You can now use the remote connection with TIA Portal and communicate with the automation system.

Connection cannot be established

If the connection cannot be established, try to find the cause using the "Notes on troubleshooting".

Note

Rules for IP addresses

- Only use IP addresses which have not yet been assigned in the system network.
- If the IP address configured for the TS Adapter IE Advanced has already been assigned in the system network, the TS Adapter IE Advanced can only be addressed by its MAC address.

See also

Installing CA certificates for VPN connections (Page 10955)

Terminating VPN connections

Terminating an active VPN connection

Note
Terminating the VPN connection
You should go offline in the TIA portal before you terminate the VPN connection.

Proceed as follows:

1. Double-click the "Set up/close remote connection" entry in the TIA portal.
2. Confirm the prompt in the dialog that follows with "Yes".

Result

The VPN connection is terminated.

13.7.4.6 TS Adapter IE Advanced

Short description of the TS Adapter IE Advanced

TS Adapter IE Advanced

The TS Adapter IE Advanced has the following properties:

- Direct connection over Industrial Ethernet (IE), 2 ports
- WAN (Wide Area Network) interface for VPN connections
- Firmware update supported
- Plug-in modules
- Parameters are assigned via a Web interface.

Note

Further information on the TS Adapter IE Advanced

For detailed information on your TS Adapter, please refer to the documentation supplied with your TS Adapter.

Connection types

TS Adapter IE Advanced connection types

The following diagrams show the types of connection possible with the TS Adapter IE Advanced.

Direct connection

In the direct connection to the programming device/PC, you can set the TS Adapter IE Advanced parameters over the Ethernet.

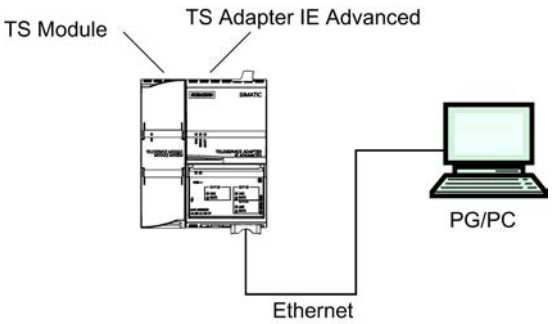


Figure 13-5 TS Adapter IE Advanced - direct connection

Connection to the Internet (DSL modem/router)

In order to connect to the Internet, you must operate the DSL modem/router at the WAN connection of the TS Adapter IE Advanced.

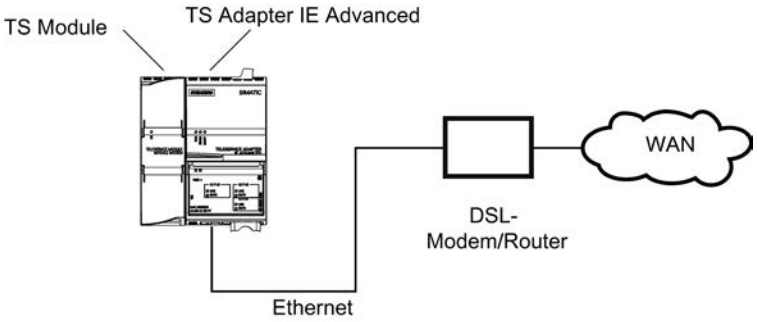


Figure 13-6 TS Adapter IE Advanced - connection to the Internet

Connection to the company network (Intranet)

In order to connect to the Intranet, you must operate system network (Ethernet) at one of the two LAN connections of the TS Adapter IE Advanced.

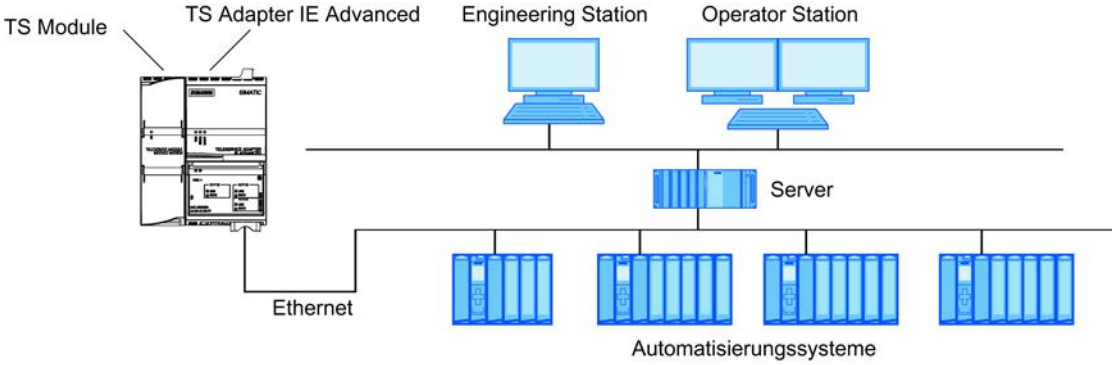


Figure 13-7 TS Adapter IE Advanced - connection to company network (Intranet)

Additional information

For more detailed information on the TS modules, please refer to the documentation supplied with your TS Adapter IE Advanced.

TS Adapter IE Advanced parameter assignment options

Basics on configuring the TS Adapter IE

The TS Adapter IE Advanced is configured via a Web interface.

Web help in the Web interface is available for assigning TS Adapter IE Advanced parameters.

Parameter assignment options include:

- Reconfiguration
- Restoring default parameter assignment
- Importing adapter parameters
- Exporting adapter parameters

Note

Parameter assignment

Configure your TS Adapter IE Advanced in accordance with the documentation supplied. This details the exact procedure for parameter assignment.

Parameter assignment for TS Adapter IE Advanced

Introduction

The TS Adapter IE Advanced can be configured in both direct connection mode and via an existing remote connection.

Both parameter assignment options are described below.

Specific details on assigning parameters for the TS Adapter IE Advanced are available in the TS Adapter IE Advanced documentation.

Assigning parameters for the TS Adapter IE Advanced in direct connection

Requirement

There is a LAN connection to your TS Adapter IE Advanced.

Procedure

To assign the parameters for the TS Adapter IE Advanced, proceed as follows:

1. In the project tree of the TIA portal, open the "Online access" folder.
2. Double-click on the Ethernet port of your computer.
3. Double-click on the "Display accessible nodes" command. The TS Adapter IE Advanced is then displayed.
4. Double-click on the "TS Adapter IE Advanced" folder and then on "Online and diagnostics", and assign the desired IP address to the TS Adapter in the dialogs that follow. Please note that the IP address of the programming device/PC interface card must be located in the same subnet as the IP address that you assign for the TS Adapter IE Advanced.
5. Update the view in the project tree for the "Accessible nodes", so that the TS Adapter IE Advanced is displayed with the newly assigned IP address.

6. Open the folder "TS Adapter IE Advanced" in the device list.
7. Double-click the command "Assign TS Adapter IE parameters". The corresponding Web interface opens for assigning TS Adapter IE Advanced parameters.
8. Complete the "logon" for the web interface.
9. Set the required parameters in the individual tabs of the dialog.
10. Confirm your entries with "Save settings".

Result

The configured parameters are saved in the non-volatile memory of the TS Adapter IE Advanced. Parameter assignment is then complete.

Assigning parameters for the TS Adapter IE Advanced using a remote connection

Requirement

There is an established remote connection to a TS Adapter IE Advanced..

Procedure

1. In the project tree of the TIA Portal, open the "Online access" folder.
2. Open the "TeleService" folder followed by the required plant folder.
3. Double-click the command "Assign TS Adapter IE parameters". The associated web interface for assignment of the TS Adapter IE parameters opens. The "logon" for the web interface takes place automatically with the login data of the remote connection.
4. Set the required parameters in the individual tabs of the dialog.
5. Confirm your entries with "Save settings".

Result

The configured parameters are saved in the non-volatile memory of the TS Adapter IE Advanced. Parameter assignment is then complete.

13.7.5 CPU controlled TeleService remote connections

13.7.5.1 Overview of CPU controlled remote connections

Introduction

TeleService offers a range of options for establishing remote connections; these differ according to the CPU used. The initiative for establishing a connection starts from the CPU. The communications instructions given below are used for the individual connection options.

Connection establishment with S7-300/400 CPUs

The following communications instructions are available:

- Communications instruction "PG_DIAL": Establish remote connection to programming device/PC
- Communications instruction "SMS_SEND": Send text message (SMS)
- Communications instruction "AS_DIAL": Establish remote connection to AS
- Communications instruction "AS_MAIL": Transfer email

Connection establishment with S7-1200 CPUs

The following communications instruction is available:

- Communications instruction "TM_MAIL": Transfer email

Connection establishment with S7-1500 CPUs

The following communications instruction is available:

- Communications instruction "TMAIL_C": Transfer email

Note

Description of individual communications instructions

More detailed information on the available communications instructions can be found in the information system of the TIA Portal in the directory "References > Communication > TeleService".

See also

PG_DIAL: Establish remote connection to programming device/PC (Page 7933)

AS_DIAL: Establish remote connection to AS (Page 7937)

SMS_SEND: Send text (SMS) message (Page 7943)

AS_MAIL: Transfer email (Page 7949)

TM_MAIL: Transfer email (Page 5360)

TS Adapter IE parameter assignment options (Page 10948)

13.7.5.2 Establishing a connection from and to remote systens (PG-AS-remote coupling)

Remote plant access to a programming device/personal computer

Introduction

You can establish a remote connection to and communicate with a remote system using the application TeleService and a TS Adapter MPI. The initiative for establishing the remote connection comes from the programming device/personal computer.

However, events which require rapid intervention often occur at a remote system. In such cases, the automation system can initiate a remote connection to a programming device/ personal computer if an asynchronous event occurs.

The graphic below shows the components which are required for establishing a connection from a plant to a programming device/personal computer.

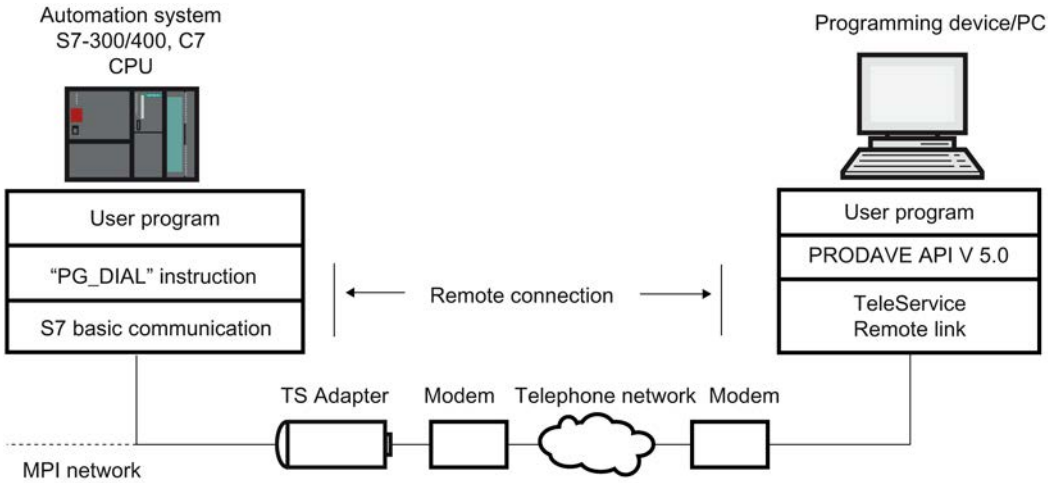


Figure 13-8 How the "PG_DIAL" communications instruction works

Requirements for establishing a connection

Introduction

Certain hardware and software requirements must be fulfilled if a remote system is to establish a remote connection to a programming device/personal computer. These requirements are detailed below.

Hardware requirements:

The only hardware required for establishing a remote connection from a remote system to a programming device/personal computer is that needed for accessing the remote system from the programming device/personal computer.

Your user program calls the communication instruction "PG_DIAL" to establish the connection. This can only be executed on an S7-300 or S7-400 CPU on which S7 basic communication is implemented.

A TS Adapter I , version 5.0 or later, or a TS Adapter II must be used.

Software requirements at the plant end:

Communication instruction "PG_DIAL" is included in the TeleService scope of delivery and is installed when the TIA portal is installed. You will find the communication instructions installed in the "Communication > TeleService" folder of the block editor task card.

If a remote system is to establish a remote connection to a programming device/personal computer, the plant user program must call the "PG_DIAL" function block.

Software requirements for the programming device/personal computer

You require a software product in the programming device/personal computer which, with TeleService, waits for a call from a remote system, recognizes this call and informs your user program.

13.7.5.3 Data exchange between remote systems (AS-AS-remote coupling)

AS-AS remote link basics

Introduction

The AS-AS remote link allows two automation systems to exchange process data via the telephone network.

Requirement

Communication instruction "AS_DIAL" is available if you use a CPU from the S7-300/400 family.

Definition: Local and remote automation system

- The automation system from which the initiative to establish the remote connection originates is described as **local**.
- The automation system to which the remote connection is to be established is described as **remote**.

Data exchange over the AS-AS remote link

Data exchange is carried out using specific communication instructions for non-configured S7 connections. Use the communication instruction "AS_DIAL" to establish a remote connection to the automation system.

More detailed information on establishing the connection can be found in the information system in the directory "References > Communication > TeleService".

The following graphic shows the components required for establishing a connection from a local to a remote automation system.

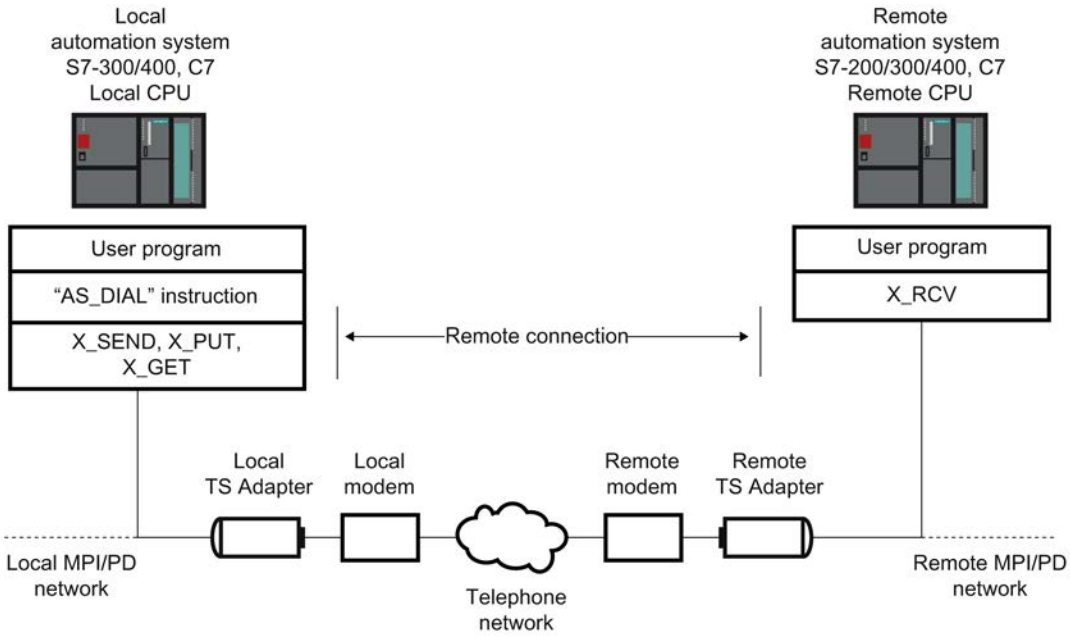


Figure 13-9 Data exchange over the AS-AS remote link

Hardware and software requirements for AS-AS remote link

Introduction

Certain hardware and software requirements must be fulfilled before a local automation system can establish a remote connection to a remote automation system. These requirements are detailed below.

Hardware requirements

The only hardware you need for transferring process data from a local to a remote automation system is that also needed for accessing the respective automation system from the programming device/personal computer.

To establish and terminate the remote connection, the TIA Portal user program of the local CPU calls a communication instruction. This communication instruction can be executed on an S7-300/400 CPU or a C7 CPU. The communication instruction requires S7 basic

communication to be implemented on the CPU. The remote CPU must also support S7 basic communication.

A TS Adapter I, version V5.1 or later, or a TS Adapter II must be used.

Software requirements

The "AS_DIAL" communication instruction is included in the product package of TeleService, and is integrated into the library of the TIA Portal during the installation in the communication instructions folder of the Task Card under TeleService. In order to establish and terminate a remote connection to a remote automation system from a local automation system, call the communication instruction "AS_DIAL" in the TIA Portal user program of the local CPU.

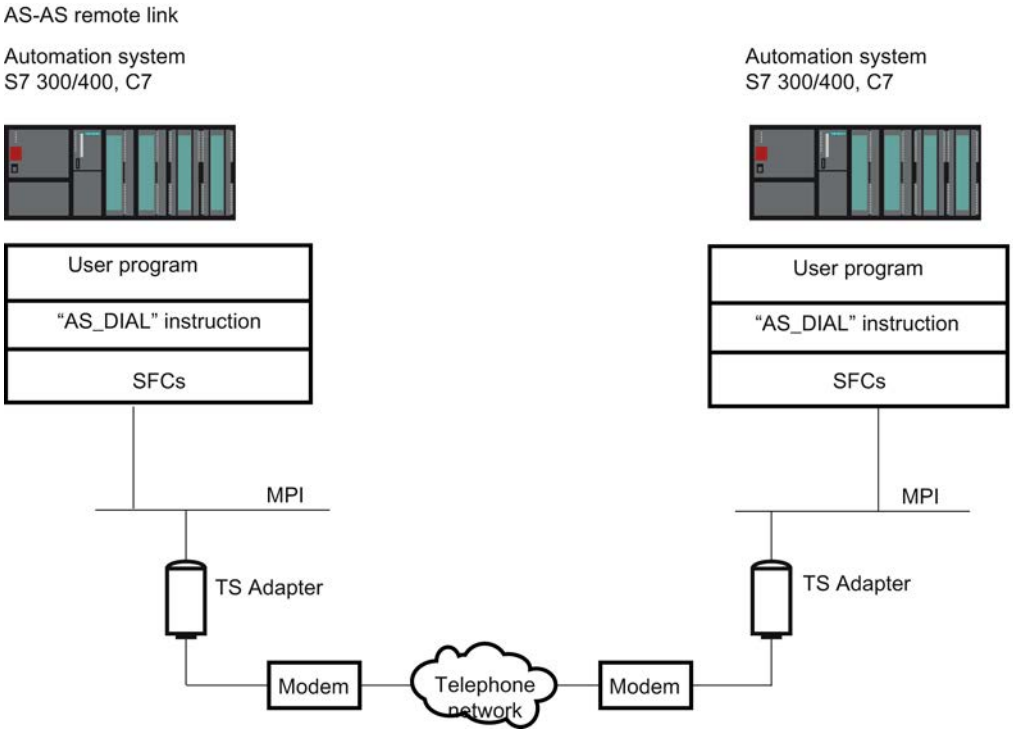


Figure 13-10 Hardware and software requirements for AS-AS remote link

13.7.5.4 Send SMS from a system

Requirements for sending an SMS

Introduction

Certain hardware and software requirements must be fulfilled before a system can send an SMS. These requirements are detailed below.

Hardware requirements

To send an SMS from a system, you will require a GSM wireless modem and a TS Adapter MPI.

A TS Adapter I, version V5.2 or later, or a TS Adapter II must be used.

Software requirements at the system end

The "SMS_SEND" communications instruction is included in the scope of delivery of TeleService, and is integrated into the library of the TIA Portal during the installation in the communications instructions folder of the Task Card under TeleService. If a system is to send an SMS, the user program of the system must call the communications instruction "SMS_SEND".

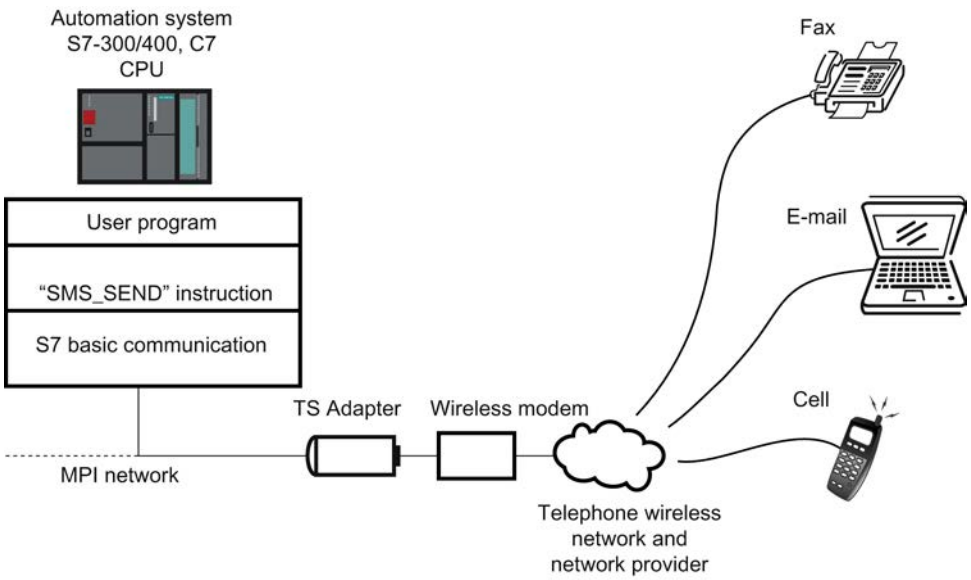


Figure 13-11 How the "SMS_SEND" communications instruction works

Note

You may be able to send an SMS not only to a cell phone but also to an email address or a fax device by using additional services offered by the cell phone service provider.

13.7.5.5 Send an email from a system

Requirements for sending e-mails

Introduction

The following hardware and software requirements must be fulfilled if a system is to send an email:

Hardware requirements

You need a TS Adapter IE to send an email from a system and one of the CPUs listed below:

- a CPU 31x2 PN/DP as of firmware version V2.5
- a CPU 41x-3 PN/DP
- a CPU of the S7-1200 series
- a CPU of the S7-1500 series

Software requirements at the system end

Different communications instructions are included in the scope of delivery of TeleService depending on the CPU that are integrated into the library of the TIA Portal during the installation in the communications instructions folder of the Task Card under TeleService.

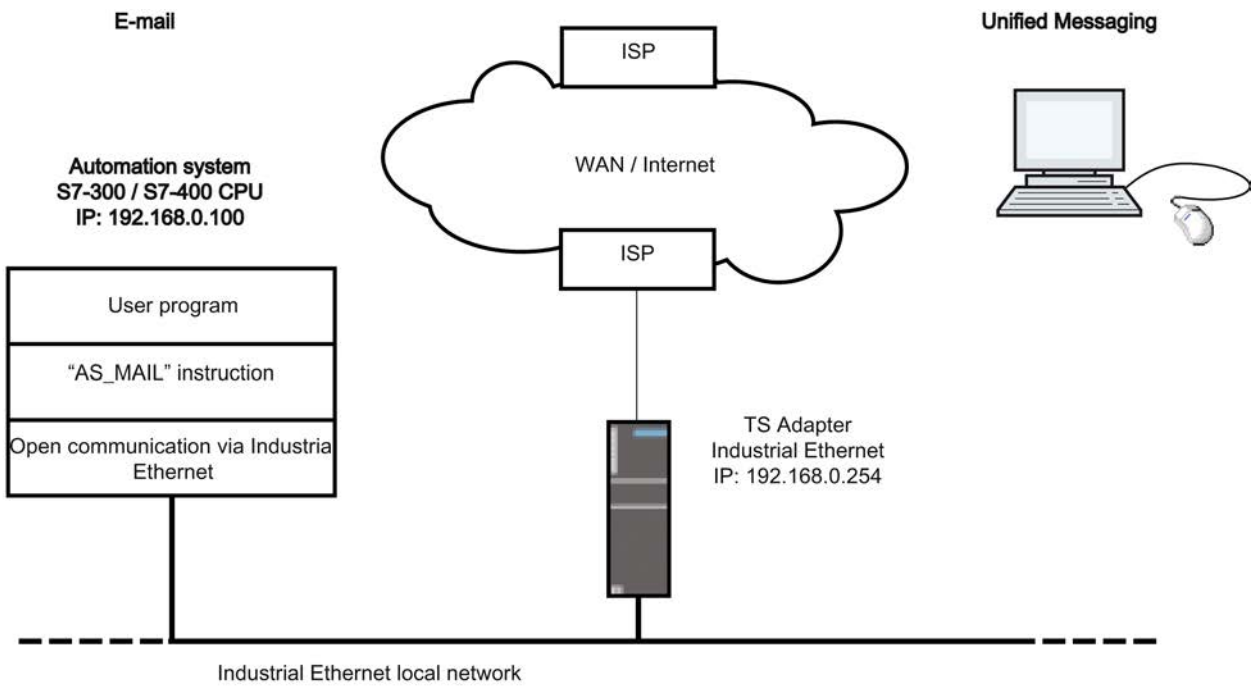
If a system is to send an email, the user program of the system must call the corresponding communications instruction for the CPU.

The following communications instructions are available for sending an email:

- S7-300/400 CPUs: use the communications instruction "AS_MAIL": Transfer email
- S7-1200 CPU V 2.0 and V 3.0: use the communications instruction "TM_MAIL": Transfer email
- S7-1200 CPU >V3.0: use the communications instruction "TMAIL_C": Transfer email
- S7-1500 CPUs: use the communications instruction "TMAIL_C": Transfer email

The respective communications instruction transfers an email from a CPU to a mail server by means of the Simple Mail Transfer Protocol (SMTP) with the "LOGIN" authentication method. The data is transferred unencrypted with this SMTP process.

The figure below shows an example with the communications instruction "AS_MAIL":



The "Use gateway/router" property must also be set for the Ethernet interface in the configuration of the CPU on which the communications instruction "AS_MAIL" runs. (available in the device configuration under Ethernet addresses and there under IP protocol)The IP address of the Ethernet interface of the TS Adapter IE is to be specified as "Address".

Note

You can find further information in the task card in the "Communications instructions" folder under TeleService.

13.7.6 Notes on troubleshooting

13.7.6.1 General information on troubleshooting for modem problems

Introduction

The information below should help you establish and eliminate the causes of any modem problems.

1. Enable "Record log file" for data traffic between PG/PC and modem. The entries in this file can provide valuable information for determining the cause of errors.
2. Switch on the loudspeaker on your local modem. Select a volume loud enough to be clearly heard.
   You can then hear whether:
   – there is a dial tone at the connection
   – the modem called is busy
   – the modem called accepts the call.

Common modem problems

Modem connection problems are among the most common modem problems:

- Modem connection is not established
- Modem connection is interrupted

The topics below contain tables detailing possible causes and providing information on eliminating the error in question.

See also

Dial-up connection to the TS Adapter is not established (Page 10974)

Dial-up connection from the TS Adapter is not established (Page 10975)

Modem connection is interrupted (Page 10976)

Modem alarms (Page 10977)

Recording a log file for the modem (Page 10973)

13.7.6.2 Recording a log file for the modem

Introduction

It is advisable to record a log file as this makes it easier to find the causes of faults in a modem.

Procedure:

Proceed as follows:

1. Activate the properties dialog of the modem used via the "Phone and modem options" option in the Control Panel.
2. Check the settings of the "Log" option in the "Diagnostics" tab and if necessary change the log file settings so that the file is recorded.

Result:

Activity between the programming device/personal computer and the modem are entered in the log file. If there are problems establishing the connection, you can evaluate the data recorded in the log file to determine the cause of the error.

13.7.6.3 Dial-up connection to the TS Adapter is not established

Dial-up connection to TS Adapter is not established

The table below sets out possible causes and how to eliminate them if no remote connection to the TS Adapter can be established.

Possible cause	Check/Remedy
Cabling faulty	• Are all the connecting cables connected correctly? • Are the connectors loose?
Dial parameters for main telephone line and extension incorrectly set	• Do the properties and dial parameters of your modem match the phone connection to main phone line or extension? • Do not specify a dial-out code in the "Dial parameters" dialog if you operate your modem on a local loop (main telephone line). The fields for the dial-out code for local calls and long-distance calls must be empty.
Dialing mode incorrectly set	• Is the correct dialing mode (tone/pulse) set in the dialog for the dial parameters of your modem? • Use a connected telephone to check the connection on which you want to operate the modem. You should hear crackling noises on the telephone during pulse dialing and tones of varying pitches during tone dialing. Set the corresponding dialing mode in the modem dial parameters.

Possible cause	Check/Remedy
Dial disable active	• The dial disable function is a country-specific modem property which, depending on the modem, becomes effective after one or more unsuccessful attempts to establish a connection. • If your modem still does not respond after several attempts to dial, the dial disable function may be active. Characters are still sent to the modem after the dial command but the modem does not start the dialing process. The driver receives a general error message. • Refer to the modem documentation for information on how the dial disable function is implemented for your modem. • Create a log file (Page 10973) (modemlog.txt) in which the activities between the programming device/personal computer and modem are recorded. Then check whether the file contains an entry caused by dial disable (e.g. DELAYED).
Phone connection defective or busy	• Connect a phone and check whether a dial tone can be heard on this connection. • Any analog phone connected on the same connection must be hung up. You cannot establish an additional modem connection on this connection if there is an existing phone connection.
Serial parameters set incorrectly	• Are the correct values entered in the "Settings" tab for modem properties (8 data bits, no parity, 1 stop bit)? Is the correct COM interface set in the "General" tab for the modem properties?
Initialization string of the TS Adapter is not suitable for the modem.	• Familiarize yourself with the modem initialization string requirements and set the string accordingly. Procedure for configuring the TS Adapter IE (Page 10945)
Settings for error correction between the modem at the TS Adapter and the modem at the PC/programming device are not compatible.	• Adapt the modem settings. Useful information on configuring the TS Adapter MPI (Page 10940) Restoring the default parameter assignment of a TS Adapter MPI (Page 10942) Procedure for configuring the TS Adapter IE (Page 10941)

13.7.6.4 Dial-up connection from the TS Adapter is not established

No callback from TS Adapter

The table below sets out possible causes and how to eliminate them if there is no callback from the TS Adapter.

Possible cause	Check/Remedy
Errors in the location or call settings in the TS Adapter	Check the TS Adapter parameter assignment: • Are the dialing mode and dial-out code set correctly for your phone connection? • Does the modem at the TS Adapter support the characters configured for the dial-out code? • Is "Wait for dial tone before dialing" deactivated for an extension?
Initialization of modem insufficient	Check the string for modem initialization: • The modem may require a further initialization in order to establish a remote connection. • Properties of the modem initialization string for the TS Adapter MPI
Callback number is incorrect	Check the configuration of the callback number you assigned.

No call from TS Adapter MPI

The table below sets out possible causes and and how to remedy them if there is no call from the TS Adapter MPI.

Possible cause	Check/Remedy
Phone number is incorrect	Is the required number being transferred to the communication instruction "PG_DIAL" ?
TS Adapter MPI parameter assignment incorrect	Check the TS Adapter MPI parameter assignment: • Are the dialing mode and dial-out code set correctly for your phone connection? • Does the modem at the TS Adapter MPI support the characters configured for the dial-out code? • Is "Wait for dial tone before dialing" deactivated for an extension?

13.7.6.5 Modem connection is interrupted

Modem connection is interrupted

The table below sets out possible causes and and how to remedy them if the modem connection is interrupted.

Possible cause:	Check / Remedy:
Metering pulses in the line	Metering pulses will be generated if you have applied to the phone company for a metering clock. This may mean that the modem no longer recognizes the carrier signal and switches off. • Set a longer waiting or switch-off time at the modem. • Have the metering pulse deactivated by the phone company.
Shielding	• Are the connection cables used shielded sufficiently? • Make sure that the modem cables do not run next to power cables and that they are as far as possible from power supply units and monitors.
Protocol timeout	• Set fixed monitoring times.
Automatic connection termination	• Deactivate the option that terminates an existing connection automatically after a specified time without data transfer ("Terminate after idle of ...").
Data flow control deactivated	• Click on the "Extended" button in the "Settings" tab of the modem properties and activate the following options in the dialog displayed (if available and not yet set): – Data flow control – Hardware (RTS/CTS) – Data compression – Error control
Initialization string of the TS Adapter is not suitable for the modem	• Set the modem initialization string in accordance with the following requirements. See also: TS Adapter IE parameter assignment options (Page 10948)

See also

TS adapter MPI configuration options (Page 10940)

13.7.6.6 Checklist for troubleshooting the modem

Introduction

The following list should help you establish the potential cause of any problems with the modem. The help topics below set out how and in which dialogs you define the relevant settings.

Modem connection cannot be established:

- Check the cabling and the connections.
- Check whether the correct dialing mode (tone/pulse) is set.
- If your modem does not react after several attempts to dial, a dial disable function may be active. Familiarize yourself with dial disable on your modem.
- Are you operating your modem on a main telephone line or on an extension line? Configure the properties and dialing parameters of the modem accordingly.
- Enable the log file option in the advanced properties. The next attempt to establish a connection will then be recorded in a file in the Windows directory.
- Ensure that the ISDN TAs used work with the same B and D channel protocol.

The modem connection is terminated:

- Metering pulses can have a negative affect on a connection. Have the pulses deactivated by your telephone company.
- Set fixed monitoring times.
- Deactivate the option that terminates an existing connection automatically after a specified time without data transfer (idle).
- Make sure that you have activated RTS/CTS for data flow control.

13.7.6.7 Modem alarms

Information in the log file

The modem alarms are entered in a log file if you have activated the recording function.

The log file contains the following information:

Alarm:	Possible cause:	Remedy:
NO DIALTONE	A phone call may currently be being carried out on this line.	• Repeat the process once the phone call is over.
NO CARRIER	The device dialed is not ready, is not a modem or cannot establish a connection in the set operating mode.	• Check the numbers and the settings.

Alarm:	Possible cause:	Remedy:
BUSY	The device dialed is busy.	• Try again later.
DELAYED: ...	Dial disable	• Refer to the modem documentation for information on how the dial disable function is implemented for your modem and if necessary remove it.

13.7.6.8 Possible error messages with VPN connections

VPN connection to TS Adapter IE Advanced is not established

The table below sets out possible errors and how to eliminate them if no VPN connection to the TS Adapter IE Advanced can be established.

Possible errors	Check/Remedy
The error message "The webpage cannot be displayed" appears when the Web interface is accessed.	• Note: This problem occurs when the Windows certificate store contains a CA certificate with the same name as the CA certificate which has just been generated. • If a CA certificate for this TS Adapter is already installed in the Windows certificate store and you have generated a new CA certificate on the TS Adapter, you need to remove the old CA certificate from the Windows certificate store and install the newly generated CA certificate. • See also: Installing CA certificates for VPN connections (Page 10955) • See also: Deleting CA certificates for VPN connections (Page 10958)
Error message: "Unexpected error in application".	• Check whether other users have been saved in the TS Adapter IE Advanced user database. • The user "Administrator" cannot establish a remote connection.
Error message: "No corresponding CA certificate was found in the Windows certificate store".	• Use the automatic certificate download. Enter the fingerprint in the corresponding field in the connection dialog. • See also: Establishing VPN connections (Page 10958) Or • Export the CA certificate from the Web interface of the TS Adapter IE Advanced and install it manually in the Windows certificate store. • See also: Installing CA certificates for VPN connections (Page 10955)
Error message: "The remote address specified does not correspond to the remote address of the TS Adapter".	• You must use the remote address which you specified in the TS Adapter IE Advanced to establish the connection. • You cannot use the IP address if you have entered the DNS name (and vice versa).
Error message: "The signature of the certificate could not be verified".	• If you have generated a new CA certificate on the TS Adapter IE Advanced, you need to delete the old CA certificate from the Windows certificate store and install the new one. • See also: Deleting CA certificates for VPN connections (Page 10958) • See also: Installing CA certificates for VPN connections (Page 10955)
Error message: "Protocol error"	• Check to make sure that the IP address of the service PC is set in the configuration of the TS Adapter IE Advanced (Parameter > Plant Network)

Possible errors	Check/Remedy
Error message: "Connection setup error."	• Check to make sure that the LAN IP-address of the TS Adapter IE Advanced and the IP address of your network cards are in the same subnet. If this is the case, deactivate this network card before you establish the remote connection.
An e-mail cannot be sent.	• Check whether there is a network/Internet connection at the WAN port. • Check whether the mail server is accessible. • Check whether the outgoing connections in the TS Adapter IE Advanced are valid. • Check whether the SMTP port in the firewall is open for outgoing connections.

Hardware documentation 14

14.1 General information on the hardware documentation

Additional information on the available hardware

The TIA Portal can be used to configure a wide variety of hardware, depending on the installed products. You can find the available hardware in the hardware catalog. You can find all the current manuals, operating instructions and FAQs as well as updates for your devices in the Siemens Industry Online Support (https://support.automation.siemens.com/).

To help you locate the appropriate documents for your hardware in the Siemens Industry Online Support, the following sections list all modules and module families that are available in the current scope of the installation of the TIA Portal. You will find a link for each module that takes you directly to the relevant manuals and operating instructions in the Siemens Industry Online Support.

14.2 PLC

14.2.1 SIMATIC S7-1200

14.2.1.1 CPU

CPU 1211C (6ES7 211-1xxx-0XB0)

Information on the CPUs is available here:

- CPU 1211C AC/DC/Rly (6ES7 211-1BD30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72111BD300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1211C AC/DC/Rly (6ES7 211-1BE31-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72111BE310XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1211C AC/DC/Rly (6ES7 211-1BE40-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72111BE400XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

- CPU 1211C DC/DC/DC (6ES7 211-1AD30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72111AD300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1211C DC/DC/DC (6ES7 211-1AE31-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72111AE310XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1211C DC/DC/DC (6ES7 211-1AE40-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72111AE400XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1211C DC/DC/Rly (6ES7 211-1HD30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72111HD300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1211C DC/DC/Rly (6ES7 211-1HE31-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72111HE310XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1211C DC/DC/Rly (6ES7 211-1HE40-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72111HE400XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

CPU 1212C (6ES7 212-1xxx-0XB0)

Information on the CPUs is available here:

- CPU 1212C AC/DC/Rly (6ES7 212-1BD30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72121BD300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1212C AC/DC/Rly (6ES7 212-1BE31-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72121BE310XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1212C AC/DC/Rly (6ES7 212-1BE40-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72121BE400XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

- CPU 1212C DC/DC/DC (6ES7 212-1AD30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72121AD300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1212C DC/DC/DC (6ES7 212-1AE31-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72121AE310XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1212C DC/DC/DC (6ES7 212-1AE40-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72121AE400XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1212C DC/DC/Rly (6ES7 212-1HD30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72121HD300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1212C DC/DC/Rly (6ES7 212-1HE31-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72121HE310XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1212C DC/DC/Rly (6ES7 212-1HE40-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72121HE400XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

CPU 1214C (6ES7 214-1xxx-0XB0)

Information on the CPUs is available here:

- CPU 1214C AC/DC/Rly (6ES7 214-1BE30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72141BE300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1214C AC/DC/Rly (6ES7 214-1BG31-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72141BG310XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1214C AC/DC/Rly (6ES7 214-1BG40-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72141BG400XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

- CPU 1214C DC/DC/DC (6ES7 214-1AE30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72141AE300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1214C DC/DC/DC (6ES7 214-1AG31-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72141AG310XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1214C DC/DC/DC (6ES7 214-1AG40-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72141AG400XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1214C DC/DC/Rly (6ES7 214-1HE30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72141HE300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1214C DC/DC/Rly (6ES7 214-1HG31-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72141HG310XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1214C DC/DC/Rly (6ES7 214-1HG40-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72141HG400XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

CPU 1215C (6ES7 215-1xxx-0XB0)

Information on the CPUs is available here:

- CPU 1215C DC/DC/DC (6ES7 215-1BG31-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72151BG310XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1215C DC/DC/DC (6ES7 215-1BG40-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72151BG400XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1215C AC/DC/Rly (6ES7 215-1AG31-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72151AG310XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

- CPU 1215C AC/DC/Rly (6ES7 215-1AG40-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72151AG400XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1215C DC/DC/Rly (6ES7 215-1HG31-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72151HG310XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- CPU 1215C DC/DC/Rly (6ES7 215-1HG40-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72151HG400XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

CPU 1217C (6ES7 217-1xxx-0XB0)

Information on the 1217C DC/DC/DC CPU is available here (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72171AG400XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

14.2.1.2 Signal boards

Signal boards (6ES7 2xx-xxx30-0XB0)

Information on signal boards for S7-1200 is available here:

- DI 4x24VDC (6ES7 221-3BD30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72233BD300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=viewB))
- DI 4x5VDC (6ES7 221-3AD30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72213AD300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- DQ 4x24VDC (6ES7 222-1BD30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72221BD300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=viewB))
- DQ 4x5VDC (6ES7 222-1AD30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72221AD300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=viewB))

- DI 2/DQ 2x24VDC (6ES7 223-0BD30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72230BD300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=viewB))
- DI 2/DQ 2x24VDC (6ES7 223-3BD30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72213BD300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- DI 2/DQ 2x5VDC (6ES7 223-3AD30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72233AD300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=viewB))
- AI 1x12BIT (6ES7 231-4HA30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72314HA300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=viewB))
- AI 1xRTD (6ES7 231-5PA30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72315PA300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=viewB))
- AI 1xTC (6ES7 231-5QA30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72315QA300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=viewB))
- AQ 1x12BIT (6ES7 232-4HA30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72324HA300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=viewB))

14.2.1.3 Communication boards

Point-to-point

CB 1241 (6ES7 241-1CH30-1XB0)

Information on the CB 1241 communication board is available here (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72411CH301XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

14.2.1.4 Battery boards

BB 1297 (6ES7 297-0AX30-0XA0)

Information on the BB 1297 communication board is available here (http://support.automation.siemens.com/WW/llisapi.dll?aktprim=100&lang=en&referer=%2fWW%2f&func=cslib.cssearch&nodeid0=41886045&viewreg=WW&siteid=csius&extranet=standard&groupid=4000002&objaction=cssearch&content=adsearch%2Fadsearch%2Easpx).

14.2.1.5 Digital input modules

Digital input modules (6ES7 221-1Bx3x-0XB0)

Information on the digital input modules is available here:

- DI8 x 24VDC (6ES7 221-1BF30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72211BF300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- DI8 x 24VDC (6ES7 221-1BF32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72211BF320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- DI16 x 24VDC (6ES7 221-1BH30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72211BH300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- DI16 x 24VDC (6ES7 221-1BH32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72211BH320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

14.2.1.6 Digital output modules

Digital output modules (6ES7 222-1xx3x-0XB0)

Information on the digital output modules is available here:

- DQ 8x24VDC (6ES7 222-1BF30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72221BF300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- DQ 8x24VDC (6ES7 222-1BF32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72221BF320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- DQ 16x24VDC (6ES7 222-1BH30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72221BH300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- DQ 16x24VDC (6ES7 222-1BH32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72221BH320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- DQ 16xRelay (6ES7 222-1HH30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72221HH300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- DQ 16xRelay (6ES7 222-1HH32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72221HH320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- DQ 8xRelay (6ES7 222-1HF30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72221HF300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- DQ 8xRelay (6ES7 222-1HF32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72221HF320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

- DQ 8xNO/NC Relay (6ES7 222-1XF30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72221XF300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- DQ 8xNO/NC Relay (6ES7 222-1XF32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72221XF320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

14.2.1.7 Digital input and digital output modules

Digital input and output modules (6ES7 223-1xx3x-0XB0)

Information on the digital input and output modules is available here:

- DI 8/DQ 8x24VDC (6ES7 223-1BH30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72231BH300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- DI 8/DQ 8x24VDC (6ES7 223-1BH32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72231BH320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- DI 16/DQ 16x24VDC (6ES7 223-1BL30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72231BL300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- DI 16/DQ 16x24VDC (6ES7 223-1BL32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72231BL320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- DI 8x24VDC/DQ 8xRelay (6ES7 223-1PH30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72231PH300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- DI 8x24VDC/DQ 8xRelay (6ES7 223-1PH32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72231PH320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- DI 16x24VDC/DQ 16xRelay (6ES7 223-1PL30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72231PL300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

- DI 16x24VDC/DQ 16xRelay (6ES7 223-1PL32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72231PL320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- DI/DO 8x120VAC/DQ 8xRelay (6ES7 223-1QH30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72231QH300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- DI/DO 8x120VAC/DQ 8xRelay (6ES7 223-1QH32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72231QH320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

14.2.1.8 Analog input modules

Analog input modules (6ES7 231-xxx3x-0XB0)

Information on the analog input modules is available here:

- AI 4x13BIT (6ES7 231-4HD30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72314HD300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- AI 4x13BIT (6ES7 231-4HD32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72314HD320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- AI 4x16BIT (6ES7 231-5ND30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72315ND300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- AI 4x16BIT (6ES7 231-5ND32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72315ND320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- AI 8x13BIT (6ES7 231-4HF30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72314HF300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- AI 8x13BIT (6ES7 231-4HF32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72314HF320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

- AI 4xRTD (6ES7 231-5PD30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72315PD300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- AI 4xRTD (6ES7 231-5PD32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72315PD320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- AI 8xRTD (6ES7 231-5PF30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES723145PF300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- AI 8xRTD (6ES7 231-5PF32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72315PF320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- AI 4xTC (6ES7 231-5QD30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72315QD300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- AI 4xTC (6ES7 231-5QD32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72315QD320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- AI 8xTC (6ES7 231-5QF30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72315QF300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- AI 8xTC (6ES7 231-5QF32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72315QF320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

14.2.1.9 Analog output modules

Analog output modules (6ES7 234-4Hx3x-0XB0)

Information on the analog output modules is available here:

- AQ 2x14BIT (6ES7 232-4HB30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72344HB300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- AQ 2x14BIT (6ES7 232-4HB32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72344HB320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- AQ 4x14BIT (6ES7 232-4HD30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72344HD300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- AQ 4x14BIT (6ES7 232-4HD32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72344HD320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

14.2.1.10 Analog input and analog output modules

Analog input and output module (6ES7 234-4HE3x-0XB0)

Information on the analog input and output modules is available here:

- AI 4x13BIT/AQ 2x14BIT (6ES7 234-4HE30-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72324HB300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))
- AI 4x13BIT/AQ 2x14BIT (6ES7 234-4HE32-0XB0 (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72324HB320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view))

14.2.1.11 Communications modules

Industrial Remote Communication

CP 1242-7 (6GK7 242-7KX30-0XE00)

Information on the CP 1242-7 Telecontrol communication module is available here (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6GK72427KX300XE0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

CP 1243-1 (6GK7 243-1JX30-0XE0)

Information on the CP 1243-1 Ethernet communication module is available here (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6GK72431JX300XE0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

PROFIBUS

CM 1242-5 (6GK7 242-5DX3x-0XE0)

Information on the CM 1242-5 PROFIBUS communication module is available here (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6GK72425DX300XE0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

CM 1243-5 (6GK7 243-5DX3x-0XE0)

Information on the CM 1243-5 PROFIBUS communication module is available here (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6GK72435DX300XE0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

Point-to-point

CM 1241 RS232 (6ES7 241-1AH30-0XB0)

Information on the CM 1241 (RS232) communication module for point-to-point connections is available here (http://support.automation.siemens.com/WW/llisapi.dll?

func=cslib.csinfo&lang=en&objid=6ES72411AH300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

CM 1241 RS232 (6ES7 241-1AH32-0XB0)

Information on the CM 1241 (RS232) communication module for point-to-point connections is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72411AH320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

CM 1241 RS485 (6ES7 241-1CH30-0XB0)

Information on the CM 1241 (RS485) communication module for point-to-point connections is available here (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72411CH300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

CM 1241 RS422/485 (6ES7 241-1CH31-0XB0)

Information on the CM 1241 (RS422/485) communication module for point-to-point connections is available here (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72411CH310XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

CM 1241 RS422/485 (6ES7 241-1CH32-0XB0)

Information on the CM 1241 (RS422/485) communication module for point-to-point connections is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72411CH320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

Identification systems

RF120C (6GT2 002-0LA00)

Information on the communication module for Ident technology RFC120C is available here (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6GT20020LA00&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

AS-Interface

AS-Interface CM 1243-2 (3RK7 243-2AA30-0XB0)

Information on the CM 1243-2 AS-i communication module is available here (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=3RK72432AA300XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

14.2.1.12 Technology modules

SIWAREX WP231 (7MH4 960-2AA01)

Information on the SIWAREX WP231 weighing module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=7MH4960-2AA01&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

SIWAREX WP241 (7MH4 960-4AA01)

Information on the SIWAREX WP241 weighing module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=7MH4960-4AA01&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

4SI IO-Link (6ES7 278-4BD32-0XB0)

Information on the 4SI IO-Link IO-Link Master technology module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES72784BD320XB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

14.2.2 SIMATIC S7-1500

14.2.2.1 CPU

CPU 1511-1 PN (6xxx 511-1AK00-xAB0)

Information on the CPU 1511-1 PN (including SIPLUS variant) is available here (https://support.automation.siemens.com/WW/llisapi.dll?

func=cslib.csinfo&lang=en&objid=6ES75111AK000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

CPU 1513-1 PN (6xxx 513-1AL00-xAB0)

Information on the CPU 1513-1 PN (including SIPLUS variant) is available here (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75131AL000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

CPU 1515-2 PN (6ES7 515-2AM00-0AB0)

Information on the CPU 1515-2 PN is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75152AM000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

CPU 1516-3 PN/DP (6xxx 516-3AN00-xAB0)

Information on the CPU 1516-3 PN/DP (including SIPLUS variant) is available here (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75163AN000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

CPU 1518-3 PN/DP (6ES7 518-3AN00-0AB0)

Information on the CPU 1518-3 PN/DP is available here (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75183AN000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

CPU 1518-4 PN/DP (6ES7 518-4AP00-0AB0)

Information on the CPU 1518-4 PN/DP is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75184AP000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

CPU 1511F-1 PN (6ES7 511-1FK00-0AB0)

Information on the CPU 1511F-1 PN is available here (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75111FK000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

CPU 1513F-1 PN (6ES7 513-1FL00-0AB0)

Information on the CPU 1513F-1 PN is available here (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75131FL000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

CPU 1515F-2 PN (6ES7 515-2FM00-0AB0)

Information on the CPU 1515F-2 PN is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75152FM000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

CPU 1516F-3 PN/DP (6ES7 516-3FN00-0AB0)

Information on the CPU 1516F-1 PN/DP is available here (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75163FN000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

CPU 1518F-4 PN/DP (6ES7 518-4FP00-0AB0)

Information on the CPU 1518F-4 PN/DP is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75184FP000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

14.2.2.2 Additional modules for S7-1500

The modules of the S7-1500 can also be used for the ET 200MP distributed I/O module. Therefore, refer to the information on the expansion modules in section "ET 200MP (Page 10998)".

See also

ET 200MP (Page 10998)

14.3 Distributed I/O

14.3.1 ET 200MP

14.3.1.1 Interface modules

PROFINET

IM 155-5 PN ST (6ES7 155-5AA00-0AB0)

Information on the distributed I/O module IM 155-5 PN ST is available here (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71555AA000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

IM 155-5 PN HF (6ES7 155-5AA00-0AC0)

Information on the IM 155-5 PN HF distributed I/O module is available here (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71555AA000AC0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

14.3.1.2 Digital input modules

DI 16x24VDC BA (6ES7 521-1BH10-0AA0)

Information on the DI 16x24VDC BA digital input module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75211BH100AA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DI 16x24VDC HF (6ES7 521-1BH00-0AB0)

Information on the digital input module DI 16x24VDC HF is available here (https://support.automation.siemens.com/WW/llisapi.dll?

func=cslib.csinfo&lang=en&objid=6ES75211BH000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DI 16x24VDC SRC BA (6xxx 521-1BH50-xAA0)

Information on the DI 16x24VDC HF digital input module (including SIPLUS variant) is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75211BH500AA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DI 16x230VAC BA (6xxx 521-1FH00-xAA0)

Information on the DI 16x24VDC HF digital input module (including SIPLUS variant) is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75211FH000AA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DI 32x24VDC HF (6ES7 521-1BL00-0AB0)

Information on the digital input module DI 16x24VDC HF is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75211BL000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DI 32x24VDC BA (6ES7 521-1BL10-0AA0)

Information on the DI 16x24VDC BA digital input module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75211BL100AA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

14.3.1.3 Digital output modules

DQ 8x24VDC/2A HF (6xxx 522-1BF00-xAB0)

Information on the DQ 8x24VDC/2A HF digital output module (including SIPLUS variant) is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75221BF000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DQ 8x230VAC/2A ST (6xxx 522-5FF00-xAB0)

Information on the DQ 8x230VAC/2A ST digital output module (including SIPLUS variant) is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75225FF000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DQ 8x230VAC/5A ST (6xxx 522-5HF00-xAB0)

Information on the 8x230VAC/5A ST digital output module (including SIPLUS variant) is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75225HF000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DQ 16x24VDC/0.5A BA (6ES7 522-1BH10-0AA0)

Information on the DQ 16x24VDC/0.5A BA digital output module is available here (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75221BH100AA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DQ 16x24VDC/0.5A ST (6ES7 522-1BH00-0AB0)

Information on the digital output module DQ 16x24VDC/0.5A ST is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75221BH000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DQ 32x24VDC/0.5A BA (6ES7 522-1BL10-0AA0)

Information on the DQ 32x 24VDC/0.5A BA digital output module is available here (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75221BL100AA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DQ 32x24VDC/0.5A ST (6ES7 522-1BL00-0AB0)

Information on the digital output module DQ 32x 24VDC/0.5A is available here (https://support.automation.siemens.com/WW/llisapi.dll?

func=cslib.csinfo&lang=en&objid=6ES75221BL000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

14.3.1.4 Digital input and digital output modules

DI16/DO 16x24VDC (6ES7 523-1BL00-0AA0)

Information on the DI 16/DO 16x24VDC digital input and output module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75231BL000AA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

14.3.1.5 Analog input modules

AI 4xU/I/RTD/TC ST (6ES7 531-7QD00-0AB0)

Information on the AI 4xU/I/RTD/TC ST analog input module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75317QD00AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

AI 8xU/I/RTD/TC ST (6ES7 531-7KF00-0AB0)

Information on the analog input module AI 8xU/I/RTD/TC ST is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75317KF000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

AI 8xU/I HS (6xxx 531-7NF10-xAB0)

Information on the AI 8xU/I HS analog input module (including SIPLUS variant) is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75317NF100AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

14.3.1.6 Analog output modules

AQ 2xU/I ST (6ES7 532-5NB00-0AB0)

Information on the AQ 2xU/I ST analog output module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75325NB000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

AQ 4xU/I ST (6ES7 532-5HD00-0AB0)

Information on the analog output module AQ 4xU/I ST is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75325HD000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

AQ 8xU/I HS (6xxx 532-5HF00-xAB0)

Information on the AQ 8xU/I HS analog output module (including SIPLUS variant) is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75325HF000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

14.3.1.7 Analog input and analog output modules

AI/AQ 4xU/I/RTD/TC / 2xU/I ST (6ES7 534-7QE00-0AB0)

Information on the AI/AQ 4xU/I/RTD/TC / 2xU/I ST analog input and output module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75347QE000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

14.3.1.8 Communications modules

PROFINET/Ethernet

CM 1542-1 (6GK7 542-1AX00-0XE)

Information on the CM 1542-1 PROFINET/Ethernet communication module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6GK75421AX000XE0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

CP 1543-1 (6GK7 543-1AX00-0XE0)

Information on the CP 1543-1 PROFINET/Ethernet communications processor is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6GK75431AX000XE0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

PROFIBUS

CM 1542-5 (6GK7 542-5DX00-0XE0)

Information on the CM 1542-5 PROFIBUS communication module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6GK75425DX000XE0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

Point-to-point

RS232

CM PtP RS232 BA (6xxx 540-1AD00-xAA0)

Information on the CM PtP RS232 BA point-to-point communication module (including SIPLUS variant) is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75401AD000AA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

CM PtP RS232 HF (6ES7 541-1AD00-0AB0)

Information on the CM PtP RS232 HF point-to-point communication module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75411AD000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

RS422/485

CM PtP RS422/485 BA (6xxx 540-1AB00-xAA0)

Information on the CM PtP RS422/485 BA point-to-point communication module (including SIPLUS variant) is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75401AB000AA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

CM PtP RS422/485 HF (6xxx 541-1AB00-xAB0)

Information on the CM PtP RS422/485 HF point-to-point communication module (including SIPLUS variant) is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75411AB000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

14.3.1.9 Power supply module

PS 25W 24VDC (6ES7 505-0KA00-0AB0)

Information on the PS 25W 24VDC power supply module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75050KA000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

PS 60W 24/48/60VDC (6xxx 505-0RA00-xAB0)

Information on the PS 60W 24/48/60VDC power supply module (including SIPLUS variant) is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75050RA000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

PS 60W 120/230VAC/DC (6xxx 507-0RA00-xAB0)

Information on the power supply module PS 60W 120/230VAC/DC (including SIPLUS variant) is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75070RA000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

14.3.1.10 Technology modules

TM Count 2x24V (6ES7 550-1AA00-0AB0)

Information on the counter module TM Count 2x24V is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75501AA000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

TM PosInput 2 (6ES7 551-1AB00-0AB0)

Information on the positioning module TM PosInput 2 is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75511AB000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

CM AS-i Master (3RK7 137-6SA00-0BC1)

Information on the CM AS-i Master module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=3RK71376SA000BC1&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

F-CM AS-i (3RK7 136-6SC00-0BC1)

Information on the F-CM AS-i module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=3RK71366SC000BC1&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

14.3.2 ET 200SP

14.3.2.1 CPU

CPU 1510SP-1 (6ES7 510-1DJ00-0AB0)

Information on the 1510SP-1 CPU is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75101DJ000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

CPU 1512SP-1 (6ES7 512-1DK00-0AB0)

Information on the 1512SP-1 CPU is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES75121DK000AB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

14.3.2.2 Interface modules

PROFINET

IM 155-6 PN ST (6ES7 155-6AU00-0BN0)

You can find information on the interface module IM 155-6 PN ST here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71556AU000BN0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

IM 155-6 PN HF (6ES7 155-6AU00-0CN0)

You can find information on the interface module IM 155-6 PN HF here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71556AU000CN0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

PROFIBUS

IM 155-6 DP HF (6ES7 155-6BU00-0CN0)

Information on the IM 155-6 DP HF interface module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71556BU000CN0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

Other fieldbuses

IM 155-6 Receive (6ES7 155-6DU00-0BN0)

Information on the IM 155-6 Receive interface module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71556DU000BN0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

14.3.2.3 Digital input modules

DI 4x120..230VAC ST (6ES7 131-6FD00-0BB1)

Information on the DI 4x120..230VAC ST digital input module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71316FD000BB1&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DI 8x24VDC ST (6ES7 131-6BF00-0BA0)

Information on the digital input module DI 8x24VDC ST is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71316BF000BA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DI 8x24VDC HF (6ES7 131-6BF00-0CA0)

Information on the digital input module DI 8x24VDC HF is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71316BF000CA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DI 8xNAMUR HF (6ES7 131-6TF00-0CA0)

Information on the DI 8xNAMUR HF digital input module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71316TF000CA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DI 8x24VDC SRC BA (6ES7 131-6BF60-0AA0)

Information on the DI 8x24VDC SRC BA digital input module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71316BF6000A0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DI 16x24VDC ST (6ES7 131-6BH00-0BA0)

Information on the digital input module DI 16x24VDC ST is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71316BH000BA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

14.3.2.4 Digital output modules

DQ 4x24VDC/2A ST (6ES7 132-6BD20-0BA0)

Information on the digital output module DQ 4x24VDC/2A ST is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71326BD200BA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DQ 4x24..230VAC/2A ST (6ES7 132-6FD00-0BB1)

Information on the DQ 4x24..230VAC/2A ST digital output module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71326FD000BB1&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DQ 4x24VDC/2A HF (6ES7 132-6BD20-0CA0)

Information on the DQ 4x24VDC/2A HF digital output module is available here (https://support.automation.siemens.com/WW/llisapi.dll?

func=cslib.csinfo&lang=en&objid=6ES71326BD200CA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

RQ 4x120..230VUC/5A NO ST (6ES7 132-6HD00-0BB0)

Information on the RQ 4x120..230VUC/5A NO ST relay output module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71326HD000BB0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

RQ 4x24VUC/2A CO ST (6ES7 132-6GD50-0BA0)

Information on the RQ 4x24VUC/2A CO ST relay output module is available here.

DQ 8x24VDC/0.5A ST (6ES7 132-6BF00-0BA0)

Information on the digital output module DQ 8x24VDC/0.5A ST is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71326BF000BA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DQ 8x24VDC/0.5A SNK BA (6ES7 132-6BF60-0AA0)

Information on the DQ 8x24VDC/0.5A SNK BA digital output module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71326BF600AA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DQ 8x24VDC/0.5A HF (6ES7 132-6BF00-0CA0)

Information on the digital output module DQ 8x24VDC/0.5A HF is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71326BF000CA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

DQ 16x24VDC/0.5A ST (6ES7 132-6BH00-0BA0)

Information on the digital output module DQ 16x24VDC/0.5A ST is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71326BH000BA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

14.3.2.5 Analog input modules

AI 2xU/I 2-/4-wire HF (6ES7 134-6HB00-0CA1)

Information on the AI 2xU/I 2-/4-wire HF analog input module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71346HB000CA1&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

AI 2xU/I 2-/4-wire HS (6ES7 134-6HB00-0DA1)

Information on the AI 2xU/I 2-/4-wire HS analog input module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71346HB000DA1&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

AI 4xI 2-/4-wire ST (6ES7 134-6GD00-0BA1)

Information on the analog input module AI 4xI 2-/4-wire ST is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71346GD000BA1&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

AI 4xU/I 2-wire ST (6ES7 134-6HD00-0BA1)

Information on the analog input module AI 4xU/I 2-wire ST is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71346HD000BA1&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

AI 4xRTD/TC 2-/3-/4-wire HF (6ES7 134-6JD00-0CA1)

Information on the analog input module AI 4xRTD/TC 2-/3-/4-wire HF is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71346JD000CA1&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

AI 8xRTD/TC 2-wire HF (6ES7 134-6JF00-0CA1)

Information on the AI 8xRTD/TC 2-wire HF analog input module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71346JF000CA1&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

AI EnergyMeter ST (6ES7 134-6PA00-0BD0)

Information on the AI EnergyMeter ST relay output module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71346PA000BD0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

14.3.2.6 Analog output modules

AQ 2xU/I HF (6ES7 135-6HB00-0CA1)

Information on the AQ 2xU/I HF analog output module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71356HB000CA1&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

AQ 2xU/I HS (6ES7 135-6HB00-0DA1)

Information on the analog output module AQ 2xU/I HS is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71356HB000DA1&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

AQ 4xU/I ST (6ES7 135-6HD00-0BA1)

Information on the analog output module AQ 4xU/I ST is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71356HD000BA1&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

14.3.2.7 Communication modules

CM PtP (6ES7 137-6AA00-0BA0)

Information on the CM PtP communication module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71376AA000BA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

14.3.2.8 Power modules

PM 120..230VAC/10A ST (6ES7 133-6AA00-0BC0)

Information on the PM 120..230VAC/10A ST power module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71336AA000BC0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

14.3.2.9 Special modules

Server module (6ES7 193-6PA00-0AA0)

Information on the server module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71936PA000AA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

BusAdapter BA 2xSCRJ (6ES7 193-6AP00-0AA0)

Information on the BA 2xSCRJ BusAdapter is available here.

See also

Link to the device manual (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71936AP000AA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view)

BusAdapter BA 2xRJ45 (6ES7 193-6AR00-0AA0)

Information on the BusAdapter BA 2xRJ45 is available here (https://support.automation.siemens.com/WW/llisapi.dll?

func=cslib.csinfo&lang=en&objid=6ES71936AR000AA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

BusAdapter FC (6ES7 193-6AF00-0AA0)

Information on the BusAdapter FC is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71936AF000AA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

BusAdapter Send (6ES7 193-6AS00-0AA0)

Information on the BusAdapter Send is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71936AS000AA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

BusAdapter Receive (6ES7 193-6AE00-0AA0)

Information on the BusAdapter Receive is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71936AE000AA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

14.3.2.10 Technology modules

TM Count 1x24V (6ES7 138-6AA00-0BA0)

Information on the relay output module TMCount 1x24V is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71386AA000BA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

TM PosInput 1 (6ES7 138-6BA00-0BA0)

Information on the positioning module TM PosInput 1 is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71386BA000BA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

IO-Link Master Module (6ES7 137-6BD00-0BA0)

Information on the relay output module IO-Link Master Module is available here (https://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6ES71376BD000BA0&objaction=csviewmlfbbeitraege&subtype=133300&caller=view).

Index

-

-, 2682
-D, 3966, 5969
-I, 3961, 5965
-R, 3975, 5979

add external graphic , 8842
add graphic to graphics library, 8841
6ES7 155-6AU00-0CN0, 11006
Absolute value, 3278, 3574, 4352, 5448, 5660, 6288
ADM_ZONE
In/out parameters, 7303
Output parameters, 7302
Static variables, 7303
Call
EncoderIM178, 7060
Character string
Converting with S_CONV, 6441
Connection configuration
General, 676
Count periodically, 9085
External source file
Editing, 2868
Opening, 2868
FMT_PID
In/out parameters, 7231
INVD, 4018, 6020
INVI, 4017, 6019
LP_ZONE
In/out parameters, 7300
Input parameters, 7300
Output parameters, 7300
Static variables, 7300
Master control relay (MCR)
Functionality, 2471
Measuring interval, 9009, 9012
Module
Inserting, 508
Parameter
EncoderIM178, 7060
PID_PAR
Output parameters, 7215
PULSEGEN
Input parameters, 6811
Output parameters, 6812
Quality code, 2063
READ_455
Output parameters, 7288
Remainder of division, 3973, 5976
Send_P2P (send point-to-point data)
LENGH and BUFFER parameters, 5197, 7484
SetBacklightColor, 8333
SIMATIC PC station
Load, 74
STARTUP
Warm restart, 1566, 1642, 1834
User permissions , 1938
Warm restart, 1642
Zero offset
Canceling, 10584
Zoom in/out, (See Zoom)

"

"Interrupt" gate function, 9103
"IP access protection" tab, 1934
"IP configuration" tab, 1924
"OPC properties" tab, 660
"Port parameters" tab, 1931
"Target computer configuration" dialog box, 1998
"WWW" instruction, 928, 963

#

#, 2552

&

&, 2686, 2687

(

(), 2687

*

*, 2682
**, 2682
*.bmp, 8010

*.cer, 809
*.dat, 809
*.emf, 8010
*.gif, 8010
*.ico, 8010
*.jpeg, 8010
*.jpg, 8010
*.p12, 737, 809
*.tif, 8010
*.wmf, 8010
*D, 3967, 5970
*I, 3962, 5966
*R, 3976, 5980

.

.Net service packs, 32
.Net versions, 32

/

/, 2682
/D, 3968, 5971
/FC, 2453
/I, 3963, 5967
/R, 3977, 5981

:

:=, 2687, 2688
:P, 1577, 1845

+

+, 2682
+AR1, 4107, 6108
+AR2, 4109, 6110
+D, 3965, 5968
+I, 3960, 5964
+R, 3974, 5978

<

<, 2684
<=, 2684
<>, 2684, 2687

=

=, 2684
==, 2687

>

>, 2684
>=, 2684

1

1Count24V, 9219
1Count5V, 9281
1PosU
- Absolute positioning (MODE 5), 10666
- Activating JOBs, 10653
- Axis type, 10692, 10698, 10704
- Cancel JOB processing (JOB 0), 10669
- Causes of JOB_ERR error, 10688
- Causes of POS_ERR error, 10687
- Changeover difference, 10696, 10702, 10707
- Changing the changeover difference (JOB 4), 10675
- Changing the switch-off difference (JOB 3), 10674
- Control interface, 10707
- Controlling the drive, 10654
- Controlling the Drive, 10695, 10701, 10706
- CPU/Master STOP, 10689
- Defining the settings for monitoring the direction of rotation (JOB 11), 10679
- directional enables, 10655
- Displaying current values (JOB 15), 10681
- Dosing mode, 10651, 10691, 10697
- Encoder inputs, 10697
- Encoder type, 10703
- End of rotary axis, 10692, 10698
- Errors in the control of MODEs and JOBs, 10686
- Evaluating the reference signal (JOB 9), 10676
- Evaluation of the encoder signals, 10654
- External errors, 10684
- EXTF_ACK error acknowledgment, 10686
- Feedback interface, 10709
- Functions, 10650
- Hardware limit switches, 10693, 10699, 10705
- Inching (MODE 1), 10656
- Increments per encoder revolution, 10703
- Number of encoder revolutions, 10704
- Parameter assignment error, 10683

Reduction cam, 10694, 10700, 10706
Reference point approach (MODE 3), 10659
Reference signal, 10694, 10700
Reference switch, 10694, 10700
Relative positioning (MODE 4), 10664
RESET state, 10690
Reversal of direction of rotation, 10692, 10697, 10704
Setting actual value (JOB 1), 10670
Shifting the encoder range (JOB 2), 10672
Start direction for reference point approach, 10695, 10701
Starting MODEs, 10652
Stop (MODE 0), 10656
Switch-off difference, 10695, 10701, 10706
Tmin direction change, 10696, 10702, 10707
Transmission rate, 10703
1SSI
Encoder value acquisition, 10416, 10431, 10443
1STEP 5V
Absolute incremental mode, 10725
Acceleration / delay a, 10716
Axis type, 10729
Base frequency, 10717
Changing Parameters during Operation, 10732
Digital input DI0 (3), 10733
Digital input DI1 (7), 10734
Hold traversing job, 10728
Maximum frequency/velocity of the axis Fmax, 10715
Modulo axis, 10730
Output frequency/velocity Fa, 10716
Pulse enable, 10731
Reaction to CPU/Master STOP, 10735
Reference point approach, 10720
Reference point approach, process, 10721
Relative incremental mode, 10724
Set home position, 10724
Speed-control mode, 10726
Starting a traversing job, 10719
Start-stop frequency Fss, 10715
Traversal curve, 10714

2

24 V encoder signals, 9106
Input filter, 9124, 9125, 9200, 9201
2PULSE
Direct control of the DO, 9989
Frequency output, 9982
Measurement of current, 9988
On/Off-delay, 9974
Parallel connection of both channels, 9992
Pulse output, 9952
Pulse train, 9968
Pulse width modulation, 9958
Sequence counter, 9988

3

32-bit down counter, 2028
3964(R)
Receiving data, 998
Sending data, 997
3964(R) procedure, 995, 1052
Block check character, 1053
Control characters, 996, 1052
Priority, 996, 1053
receiving data, 1057
Sending Data, 1054
3964(R) protocol (CPU 31xC), 1732
3964R procedure
Block check character, 996
3DES, 787
3RK7 136-6SC00-0BC1, 11006
3RK7 137-6SA00-0BC1, 11005
3RK7 243-2AA30-0XB0, 10995

5

5 V encoder signals, 9105
Monitoring, 9106

6

611U-compliant mode, 10758
6AG1 505-0RA00-0AB0, 11005
6AG1 507-0RA00-0AB0, 11005
6AG1 511-1AK00-2AB0, 10996
6AG1 513-1AL00-2AB0, 10996
6AG1 516-3AN00-2AB0, 10996
6AG1 521-1BH50-7AA0, 10999
6AG1 521-1FH00-7AA0, 10999
6AG1 522-1BF00-7AB0, 11000
6AG1 522-5FF00-7AB0, 11000
6AG1 522-5HF00-2AB0, 11000
6AG1 531-7NF10-7AB0, 11002
6AG1 532-5HF00-7AB0, 11002
6AG1 540-1AB00-7AA0, 11004
6AG1 540-1AD00-7AA0, 11004
6AG1 541-1AB00-7AB0, 11004
6ES7 131-6BF00-0BA0, 11007
6ES7 131-6BF00-0CA0, 11007

6ES7 131-6BF60-0AA0, 11008
6ES7 131-6BH00-0BA0, 11008
6ES7 131-6FD00-0BB1, 11007
6ES7 131-6TF00-0CA0, 11008
6ES7 132-6BD20-0BA0, 11008
6ES7 132-6BD20-0CA0, 11009
6ES7 132-6BF00-0BA0, 11009
6ES7 132-6BF00-0CA0, 11009
6ES7 132-6BF60-0AA0, 11009
6ES7 132-6BH00-0BA0, 11010
6ES7 132-6FD00-0BB1, 11008
6ES7 132-6GD50-0BA0, 11009
6ES7 132-6HD00-0BB0, 11009
6ES7 133-6AA00-0BC0, 11012
6ES7 134-6GD00-0BA1, 11010
6ES7 134-6HB00-0CA1, 11010
6ES7 134-6HB00-0DA1, 11010
6ES7 134-6HD00-0BA1, 11010
6ES7 134-6JD00-0CA1, 11010
6ES7 134-6JF00-0CA1, 11011
6ES7 134-6PA00-0BD0, 11011
6ES7 135-6HB00-0CA1, 11011
6ES7 135-6HB00-0DA1, 11011
6ES7 135-6HD00-0BA1, 11011
6ES7 137-6AA00-0BA0, 11012
6ES7 137-6BD00-0BA0, 11014
6ES7 138-6AA00-0BA0, 11013
6ES7 138-6BA00-0BA0, 11014
6ES7 155-5AA00-0AB0, 10998
6ES7 155-5AA00-0AC0, 10998
6ES7 155-6AU00-0BN0, 11006
6ES7 155-6BU00-0CN0, 11007
6ES7 155-6DU00-0BN0, 11007
6ES7 193-6AE00-0AA0, 11013
6ES7 193-6AF00-0AA0, 11013
6ES7 193-6AP00-0AA0, 11012
6ES7 193-6AR00-0AA0, 11013
6ES7 193-6AS00-0AA0, 11013
6ES7 193-6PA00-0AA0, 11012
6ES7 211-1AD30-0XB0, 10982
6ES7 211-1AE31-0XB0, 10982
6ES7 211-1AE40-0XB0, 10982
6ES7 211-1BD30-0XB0, 10982
6ES7 211-1BE31-0XB0, 10982
6ES7 211-1BE40-0XB0, 10982
6ES7 211-1HD30-0XB0, 10982
6ES7 211-1HE31-0XB0, 10982
6ES7 211-1HE40-0XB0, 10982
6ES7 212-1AD30-0XB0, 10983
6ES7 212-1AE31-0XB0, 10983
6ES7 212-1AE40-0XB0, 10983
6ES7 212-1BD30-0XB0, 10983
6ES7 212-1BE31-0XB0, 10983
6ES7 212-1BE40-0XB0, 10983
6ES7 212-1HD30-0XB0, 10983
6ES7 212-1HE31-0XB0, 10983
6ES7 212-1HE40-0XB0, 10983
6ES7 214-1AE30-0XB0, 10984
6ES7 214-1AG31-0XB0, 10984
6ES7 214-1AG40-0XB0, 10984
6ES7 214-1BE30-0XB0, 10984
6ES7 214-1BG31-0XB0, 10984
6ES7 214-1BG40-0XB0, 10984
6ES7 214-1HE30-0XB0, 10984
6ES7 214-1HG31-0XB0, 10984
6ES7 214-1HG40-0XB0, 10984
6ES7 215-1AG31-0XB0, 10985
6ES7 215-1AG40-0XB0, 10985
6ES7 215-1BG31-0XB0, 10985
6ES7 215-1BG40-0XB0, 10985
6ES7 215-1HG31-0XB0, 10985
6ES7 215-1HG40-0XB0, 10985
6ES7 217-1AG40-0XB0, 10985
6ES7 221-1BF30-0XB0, 10988
6ES7 221-1BF32-0XB0, 10988
6ES7 221-1BH30-0XB0, 10988
6ES7 221-1BH32-0XB0, 10988
6ES7 221-3AD30-0XB0, 10986
6ES7 221-3BD30-0XB0, 10986
6ES7 222-1AD30-0XB0, 10986
6ES7 222-1BD30-0XB0, 10986
6ES7 222-1BF30-0XB0, 10989
6ES7 222-1BF32-0XB0, 10989
6ES7 222-1BH30-0XB0, 10989
6ES7 222-1BH32-0XB0, 10989
6ES7 222-1HF30-0XB0, 10989
6ES7 222-1HF32-0XB0, 10989
6ES7 222-1HH30-0XB0, 10989
6ES7 222-1HH32-0XB0, 10989
6ES7 222-1XF30-0XB0, 10989
6ES7 222-1XF32-0XB0, 10989
6ES7 223-0BD30-0XB0, 10986
6ES7 223-1BH30-0XB0, 10990
6ES7 223-1BH32-0XB0, 10990
6ES7 223-1BL30-0XB0, 10990
6ES7 223-1BL32-0XB0, 10990
6ES7 223-1PH30-0XB0, 10990
6ES7 223-1PH32-0XB0, 10990
6ES7 223-1PL30-0XB0, 10990
6ES7 223-1PL32-0XB0, 10990
6ES7 223-1QH30-0XB0, 10990
6ES7 223-1QH32-0XB0, 10990
6ES7 223-3AD30-0XB0, 10986
6ES7 223-3BD30-0XB0, 10986

6ES7 231-4HA30-0XB0, 10986
6ES7 231-4HD30-0XB0, 10992
6ES7 231-4HD32-0XB0, 10992
6ES7 231-4HF30-0XB0, 10992
6ES7 231-4HF32-0XB0, 10992
6ES7 231-5ND30-0XB0, 10992
6ES7 231-5ND32-0XB0, 10992
6ES7 231-5PA30-0XB0, 10986
6ES7 231-5PD30-0XB0, 10992
6ES7 231-5PD32-0XB0, 10992
6ES7 231-5PF30-0XB0, 10992
6ES7 231-5PF32-0XB0, 10992
6ES7 231-5QA30-0XB0, 10986
6ES7 231-5QD30-0XB0, 10992
6ES7 231-5QD32-0XB0, 10992
6ES7 231-5QF30-0XB0, 10992
6ES7 231-5QF32-0XB0, 10992
6ES7 232-4HA30-0XB0, 10986
6ES7 232-4HB30-0XB0, 10992
6ES7 232-4HB32-0XB0, 10992
6ES7 232-4HD30-0XB0, 10992
6ES7 232-4HD32-0XB0, 10992
6ES7 234-4HE30-0XB0, 10993
6ES7 234-4HE32-0XB0, 10993
6ES7 241-1AH30-0XB0, 10994
6ES7 241-1AH32-0XB0, 10994
6ES7 241-1CH30-0XB0, 10994
6ES7 241-1CH30-1XB0, 10986, 10987
6ES7 241-1CH31-0XB0, 10994
6ES7 241-1CH32-0XB0, 10994
6ES7 278-4BD32-0XB0, 10995
6ES7 505-0KA00-0AB0, 11004
6ES7 505-0RA00-0AB0, 11004
6ES7 507-0RA00-0AB0, 11005
6ES7 510-1DJ00-0AB0, 11006
6ES7 511-1AK00-0AB0, 10996
6ES7 511-1FK00-0AB0, 10997
6ES7 512-1DK00-0AB0, 11006
6ES7 513-1AL00-0AB0, 10996
6ES7 513-1FL00-0AB0, 10997
6ES7 515-2AM00-0AB0, 10996
6ES7 515-2FM00-0AB0, 10997
6ES7 516-3AN00-0AB0, 10996
6ES7 516-3FN00-0AB0, 10997
6ES7 518-3AN00-0AB0, 10996
6ES7 518-4AP00-0AB0, 10996
6ES7 518-4FP00-0AB0, 10997
6ES7 521-1BH00-0AB0, 10999
6ES7 521-1BH10-0AA0, 10998
6ES7 521-1BL10-0AA0, 10999
6ES7 522-1BF00-0AB0, 10999
6ES7 522-1BH00-0AB0, 11000
6ES7 522-1BH10-0AA0, 11000
6ES7 522-1BL00-0AB0, 11001
6ES7 522-1BL10-0AA0, 11000
6ES7 522-5FF00-0AB0, 11000
6ES7 522-5HF00-0AB0, 11000
6ES7 523-1BL00-0AA0, 11001
6ES7 531-7KF00-0AB0, 11001
6ES7 531-7NF10-0AB0, 11001
6ES7 531-7QD00-0AB0, 11001
6ES7 532-5HD00-0AB0, 11002
6ES7 532-5HF00-0AB0, 11002
6ES7 532-5NB00-0AB0, 11002
6ES7 534-7QE00-0AB0, 11002
6ES7 540-1AB00-0AA0, 11004
6ES7 540-1AD00-0AA0, 11003
6ES7 541-1AB00-0AB0, 11004
6ES7 541-1AD00-0AB0, 11004
6ES7 550-1AA00-0AB0, 11005
6GK7 242-5DX30-0XE0, 10993
6GK7 242-5DX31-0XE0, 10993
6GK7 242-7KX30-0XE0, 10993
6GK7 243-1JX30-0XE0, 10993
6GK7 243-5DX30-0XE0, 10993
6GK7 243-5DX31-0XE0, 10993
6GK7 542-1AX00-0XE0, 11003
6GK7 542-5DX00-0XE0, 11003
6GK7 543-1AX00-0XE0, 11003
6GT2 002-0LA00, 10995

7

7MH4 960-2AA01, 10995
7MH4 960-4AA01, 10995

A

A, 3899, 5903
A(, 3909, 5913
A_DEAD_B
  Input parameters, 6947
  Instruction, 6944
  Output parameters, 6948
AB, 2477
Ability to read back connection parameters, 695
ABS, 3278, 3574, 3978, 4162, 4352, 5448, 5660, 5982, 6154, 6288
ABS_CTRL, 7125
  Call, 7126
  Control signals, 7126
  Data blocks used, 7126
  Description, 7125, 7129

Error response, 7127, 7131
Function switches, 7143
Job status, 7144
Jobs, 7142
Operating principle, 7125
Return values, 7126
ABS_CTRL_451, 7129
Call, 7129
Control signals, 7130
Data blocks used, 7130
Function switches, 7145
Jobs, 7144
Operating principle, 7129
Parameters, 7130
Return values, 7130
ABS_DIAG, 7127
Call, 7127
Commands, 7128
Data blocks used, 7127
Description, 7127
Diagnostics data, 7128
Error response, 7128
Operating principle, 7127
Return values, 7128
Startup, 7128
ABS_DIAG_451, 7131
Call, 7131
Data blocks used, 7131
Description, 7131
Diagnostics data, 7132
Error response, 7132
Jobs, 7132
Operating principle, 7131
Parameters, 7132
Return values, 7132
Startup, 7132
ABS_INIT, 7124
Call, 7125
Data blocks used, 7125
Description, 7124
Parameters, 7125
Return values, 7125
Absolute actual value, 10220
Absolute Addressing
of a tag, 8112, 8414
Absolute encoder, 8981, 9349, 10454
Frame run time, 9353
Increments per encoder revolution, 10518, 10616
Listen-in, 9352
Message frame times, 10457
Monoflop time, 9353, 10457
Pulse evaluation, 9352, 10457
Reaction times, 9352
Response times, 10457
Absolute encoder adjustment, 9390, 9465, 10513, 10610
Determining, 9390, 9466, 10513, 10610
Example, 9391, 9467, 10514, 10611
Mechanical adjustment, 9391, 9466, 10514, 10611
Absolute positioning
Easy Motion Control, 10772
Absolute value, 3978, 4162, 4352, 5982, 6154, 6288
Absolute value adjustment, 10238
Accelerating startup (PROFINET), 1509
Access
direct ~ to the I/O, 1743
Access control, 1294
Automatic learning, 1294
Access coordination, 7728
Access protection, 1889, 8265
Access protection level 1, 10933
Access protection level 2, 10933
Advantages, 10933
Configuring, 8269
Factory state, 10932
User administration, 8264
Validity, 10932
Access to operands, 2209, 2210, 2211, 2213, 2214, 2216, 2218
Accessible devices, 1461, 8442, 8491, 8544
Accompanying signals, 975
Accumulator, 4104, 4105, 4106, 6102, 6103, 6104, 6105, 6107
Accumulator instructions, 2650
Accumulators, 1741
ACK, 8946
Acknowledge, 8152, 8361
Alarm, 8948
Error alarm, 8948
AcknowledgeAlarm, 8316
Acknowledgment, 8154
Configuring, 8177, 8178, 8179
Acknowledgment concept
Alarm with simple acknowledgment, 8156
Alarm without acknowledgment, 8156
Acknowledgment model, 8156
ACL, 1294
ACOS, 3296, 3594, 3987, 4178, 4368, 5465, 5677, 5991, 6168, 6303
Acquisition cycle
Area pointer, 8419
Tag, 8128, 8135, 8145

Acquisition mode
Tag, 8128
ACT_TINT, 1680, 4703
Action
Basics, 2728
Block call, 2735
Changing, 2797
Copying, 2798
Counter operations, 2737
Cutting, 2798
Deleting, 2799
Event-dependent actions, 2732
Inserting, 2792, 2798
Introduction, 2726
Moving, 2799
Standard actions, 2730
Timers, 2736
Actions
Basics of redoing actions, 402
Basics of undoing, 402
Redoing, 406
Undoing, 405
activate, 8353
Activate
Project language, 8830
ActivateCleanScreen, 8290
ActivatePreviousScreen, 8291
ActivateScreen, 8288
ActivateScreenByNumber, 8289
Activating and deactivating IO devices, 1505
Active
Area pointer, 8419
Active bus module (ET 200M), 2061
Active homing
with digital input, 10231
with zero mark, 10230
with zero mark and proximity switch, 10226
Active nodes, 791
Actual encoder value
Additional, 10762
Actual value change, 9361, 9436
Actual values, 10220
Easy Motion Control, 10810
Actuator, 10211
AD, 2477, 4081, 6080
Adapting a project
For a different HMI device, 8900
Add, 3267, 3563, 3960, 3965, 3969, 3974, 5652, 5964, 5968, 5973, 5978
Adding timers with T_ADD, 6409
Objects to the group, 8016
ADD, 3267, 3563, 5440, 5652
Add empty line, 10062
Add Motion Control instructions, 10323
Add separator line, 10062
Adding
Adding timers with T_ADD, 4504
Additional encoder actual value, 10762
Additional field devices (PROFIBUS and PROFINET), 1383, 1435, 1464
Address, (see Module start address)
Area pointer, 8419
Mitsubishi, 8683
Omron Host Link, 8739
Address assignment, 881
User data with DPV1, 1392
Address assignment (value status), 1848
Address assignment rules, 881
address conversion
Instructions for, 4807
Address multiplexing
with absolute addresses, 8136
with symbolic addresses, 8136
Address overview, 515, 888
Address packing, 1359, 1427
Address parameter
Broadcast, 651
Address range, 761
Changing, 882, 939
Address register, 2225, 2227, 2233, 2234, 4107, 4109, 6108, 6110
Addresses, 2005
Assigning, 894, 940
Determining a module with GEO_LOG, 4816, 6783
DP interfaces, 1391
DP slave, 1393
Interrupt with packed, 1365, 1433
Interrupts on the I-slave, 1370
Pack, 1359, 1427
Read station address with GetStationInfo, 4750
System information, 1392
Unpack, 1360, 1428
User data with DPV1, 1391
Addressing, 1577, 1845
Addressing tags indirectly, 8132, 8133
Allen-Bradley, 8643, 8661
Allen-Bradley Ethernet IP, 8639
Changing, 882, 939
Ethernet/IP, 8643, 8661
General, 880, 881, 937
MPI, 8560
Multiplexing, 8132

Addressing operands, 2209, 2210, 2211, 2213, 2214, 2216, 2218, 2220, 2221, 2222, 2224, 2225, 2227, 2229, 2230, 2231, 2232, 2233, 2234
AdjustContrast, 8287
adjustment
    Dynamic, 9346
ADM_ZONE
    Instruction, 7301
Admin, 8941
Administrator, 739
Advanced Encryption Standard (AES), 787
Advanced mode
    Global firewall rules, 747
AES, 787
AES-128, 782
AG_CNTEX, 7734
AG_CNTRL, 7746
AG_LOCK, 7729
AG_LRECV, 7862
AG_LSEND, 7855
AG_RECV / AG_LRECV / AG_SRECV, 7724
AG_SEND / AG_LSEND / AG_SSEND, 7719
AG_UNLOCK, 7731
Aggressive mode, 787
Aging time, 1298
AG-RECV, 7862
AG-SEND, 7855
Alarm
    Acknowledge, 8948
    Alarm data type, 3052
    Components, 3048, 8157
    Configuring, 8163, 8169
    Displaying, 8090
    Editing, 8948
    Exporting, 8800
    Formal parameter, 3051
    Importing, 8801
    in Runtime, 8942
    In Runtime, 8180
    Inspector window, 63
    System function, 8185
    Time-stamped transferring with TIMESTMP, 6434
Alarm blocks, 3048
Alarm buffer
    in Runtime, 8943
    In Runtime, 8181
Alarm buffer overflow, 8360
Alarm class, 8157
    Copying, 3073
    Creating, 3073
    Editing, 3073
    in Runtime, 8942
    In Runtime, 8180
    Layout, 8182, 8945
Alarm classes, 8153
    Custom, 8153
    Name change through migration, 153
    Predefined, 8153
    Use, 8153
Alarm configuration
    Additional text, 3069
    Deleting an associate value, 3061
    Entering text, 3068
    Info text, 3069
    Introduction, 3047
    Locking an attribute, 3070
    Locking text, 3069
    Unlocking an attribute, 3070
    Unlocking text, 3069
Alarm display, 2074
    "Active alarms" view, 2076
    Acknowledging an alarm, 2077
    Archive view, 2074
    Clear archive, 2076
    Export archive, 2076
    Ignoring alarms, 2078
    Layout of the alarms in the "Active alarms" view, 2076
    Layout of the alarms in the archive view, 2075
    Receiving alarms, 2075
    Using the keyboard, 2078
Alarm editor
    Structure, 3053
Alarm event
    Acknowledge, 8152
    Incoming, 8152
    Outgoing, 8152
Alarm group, 8156, 8158
    Configuring, 8161
    Creating, 8161
    Migration, 152
Alarm indicator, 62, 8095, 8172, 8183, 8946
    Alarm classes, 8095
    Configuring, 8175
    Events, 8095
    in Runtime, 8944
    In Runtime, 8181
    Layout, 8095, 8183
    Operation, 8183
Alarm log
    in Runtime, 8943

Alarm message
Acknowledgment by the PLC, 8469, 8519, 8577, 8616, 8628, 8671, 8698, 8727, 8744
Acknowledgment on the HMI device, 8469, 8520, 8577, 8617, 8629, 8671, 8699, 8728, 8744
Configure acknowledgment, 8469, 8519, 8577, 8616, 8628, 8670, 8698, 8727, 8743
Alarm number, 8157
Assigning, 3048
Alarm settings, 1860
Alarm status, 8158
Acknowledged, 8152
Incoming, 8152
Outgoing, 8152
Alarm system, 8149
Alarm text, 8158
Formatting, 8167
Output fields, 68
Removing format settings, 8167
Special characters, 68
Alarm type, 3050
Editing, 3056, 3064
Alarm types, 8150
Alarm view, 62, 2770, 8090, 8092, 8172, 8183, 8946
Alarm text window, 8183, 8946
Column, 8091
Configuring, 8173
Configuring the layout, 8174
Control element, 8091
Layout, 8090, 8182, 8945
Operation, 8183, 8946
Alarm window, 62, 8172
Configuring, 8175
in Runtime, 8944
In Runtime, 8181
Alarm with simple acknowledgment, 8156
Alarm without acknowledgment, 8156
Alarms
Configuring, 8518, 8576, 8615, 8627
Data types, 8518, 8576, 8615, 8627
Logging on for receiving from CPU, 2159
Modicon Modbus, 8726
Non-integrated connection, 8726
Output of a tag value, 8166
Restriction, 8698
AlarmViewAcknowledgeAlarm, 8315
AlarmViewEditAlarm, 8314
AlarmViewShowOperatorNotes, 8315
Align
Object flush, 7990
Allen-Bradley, 8630, 8651, 8652
Allen-Bradley DF1 communication driver, 8651, 8652
Analog alarm, 8670
Basic Panel, 8425
Communication drivers, 8631
Data type, 8670
DF1, 8631
EtherNet/IP, 8631
Mitsubishi, 8697
Allen-Bradley
Area pointer, 8668
Allen-Bradley DF1
Configuring a connection, 8647
Connection, 8647, 8650
Connection parameters, 8648
CPU type, 8661
KF2 module, 8651, 8652
KF3 module, 8651, 8652
Migrating data types, 166
Valid data type, 8659
Allen-Bradley Ethernet IP
Address multiplexing, 8643
Addressing, 8639
Addressing type, 8641
Migrating data types, 166
Allen-Bradley EtherNet/IP
Configuring a connection, 8632
Connection, 8632, 8636
Connection parameters, 8634
Data type, 8637
Alternative branch
Creating, 2787
Introduction, 2722
AN, 3900, 5905
AN(, 3910, 5914
ANALOG, 7322
Analog alarm, 8150
Allen-Bradley, 8670
Configuring, 8163, 8170
Limit value exceeded, 8126
Mitsubishi, 8697
Omron, 8742
Analog alarm types, 8150
Analog alarms
Configuring, 8159
Analog module
Resetting to factory settings, 2122
Analog value, 3362, 3365, 3662, 3664, 3827, 3829, 4245, 4247, 4416, 4419
Analog value processing, 3362, 3365, 3662, 3664, 3827, 3829, 4245, 4247, 4416, 4419

Analyzing
Diagnostic information with DPV1, 1389
Interrupt information with DPV1, 1389
AND, 2686, 2687, 3399, 3445, 3447, 3699, 5505, 5581, 5583, 5717
Animation
Configuring, 8046
Diagonal movement, 8049
Direct movement, 8050
Green arrow in Overview, 8046
Horizontal movement, 8048
Multiple selection, 8053
Object group, 8052
Overview, 8046
Vertical movement, 8049
Antivirus programs, 83, 93
ANY, 2275
Appearance
Dynamization of an object, 8047
Applet, 741
Application, 1550
Button, 8931
Date/time field, 8923
Graphic IO field, 8927
Graphic view, 8927
IO field, 8924
Recipe view, 8097
Switch, 8932
Symbolic IO field, 8933
Trend view, 8929
User view, 8077
Application cycle, 1486
Application options, 1077
Application options of the communication module, 1079
Arccosine, 3296, 3594, 3987, 4178, 4368, 5465, 5677, 5991, 6168, 6303
Archiving projects, 348, 349
Arcsine, 3294, 3592, 3985, 4176, 4366, 5464, 5676, 5988, 6167, 6302
Arctangent, 3990, 4179, 5993, 6169
Arctangent value, 3297, 3595, 4369, 5467, 5679, 6304
Area of unplugged modules, 501
Area pointer, 8417, 8452, 8501, 8561, 8600, 8745
Acquisition cycle, 8419
Active, 8419
Address, 8419
Allen-Bradley, 8668
Basic Panel, 8428
Comment, 8419
Connection, 8417, 8422
Connections editor, 8418
Coordination, 8749
Data record, 8216, 8460, 8511, 8568, 8608, 8753
Date/time, 8502, 8503
Job mailbox, 8458, 8509, 8566, 8606, 8751
Length, 8419
PLC tag, 8419
Pointer name, 8419
Project ID, 8457, 8508, 8565, 8605, 8750
Screen number, 8453, 8507, 8561, 8601, 8746
Tab, 8419
Trends, 8468, 8625
Area pointers
Configuring, 8420
Coordination, 8456, 8505, 8564, 8604
Creating, 8420
Date/time, 8453, 8562, 8601, 8747
Date/time PLC, 8455, 8563, 8602, 8748
Length, 8421
Mitsubishi, 8695
Modicon Modbus, 8724
Omron, 8741
ARP, 841
Diagnostics, 841
Table, 841
Arrange
Object in the screen, 7982
Arrangement of byte sequence, 3629, 4198
Array, 67, 254, 2267, 2268, 8142, 8144
Creating, 8144
Indirect addressing, 8132, 8142
Move uninterruptible, 5820
Reset bit, 5542, 5853, 6230, 6357
Reset byte, 5545, 6233, 6360
Reset in the I/O, 6231
Reset in the I/O area, 5544, 6358
see ARRAY, 2553, 2834
Set bit, 5537, 6226, 6352
Set byte, 6229
Set byte array, 5540, 6355
Set in the I/O, 6227
Set in the I/O area, 5538, 6353
ARRAY, 2222, 2232
Addressing, 2214
Declaration in global data blocks, 2834
Declaration in PLC data types, 2857
Declaration in the block interface, 2553
Example, 2270, 2271
Format, 2267, 2268
ARRAY component, 46

Array data block, 3332, 3334, 3337, 3340, 3631, 3633, 3635, 3639, 3806, 3809, 3811, 3814, 4199, 4201, 4203, 4206, 4390, 4392, 4394, 4397
ARRAY data block, 2172, 2175, 2211, 2214, 2503, 2826, 2829, 2850, 2914
Array element
    Location of use of HMI tag, 67
    Name, 67
Array elements, 8142
ARRAY limits, 2689
Array tag, 8142
    Char, 67
AS interface, 1437
ASCII code table, 697
ASCII driver, 1040
    Receive buffer, 1047
    receiving data, 1042, 1076
    RS232C accompanying signals, 1048, 1073
    Sending Data, 1041
ASCII protocol, 989
ASCII protocol (CPU 31xC), 1725
ASCII TSAP, 697
Asian, 31
Asian characters
    Input on the HMI device, 8914
    Interpretation, 8914
    Memory requirements, 8914
Asian languages
    Configuration, 8914
    Font size, 8914
    Text field length, 8914
Asian operating system, 8828
ASIN, 3294, 3592, 3985, 4176, 4366, 5464, 5676, 5988, 6167, 6302
Assembling manuals, 329
Assign
    Object of a layer, 8070
Assigning
    a function to a function key, 8061
    a graphic to a function key, 8064
    Function key, 8056, 8058, 8060
Assigning access rights to specific OPC items, 1972
Assigning an IP address
    Basics, 2114, 2163
    from the project context, 2115, 2165
    Using "Accessible devices", 2114, 2163
Assigning connection path, 600
Assigning global data blocks , 49
Assigning parameters
    A module with PARM_MOD, 6596
Assigning symbol, 894, 940
Assigning tag, 894, 940
Assignment, 2688, 3159, 3452, 3917, 5377, 5588, 5921
    Diagnostic data record DS0, 9912
    Diagnostic data record DS1 of the FM 355-2, 9913
    Negate, 3160, 3453
Assignment list, 1839, 1841
    Enabling the display of retentive bit memories, 2945
Assignment list
    Meaning of symbols, 2939
    Defining filter, 2942
    Delete filter, 2942
    Displaying, 2940
    Example for displaying bit memory, 2938
    Example for displaying inputs and outputs, 2938
    Filter options, 2942
    Filtering, 2943
    Introduction, 2937
    Setting view options, 2941
    Structure, 2938
Associated value
    Example, 3067
    Inserting, 3065
    Settings, 3066
Asynchronous
    Transferring data, 8461, 8512, 8569, 8613, 8754
Asynchronous data transmission, 979
Asynchronous error
    Delaying with DIS_AIRT, 6628
    Disabling with DIS_IRT, 6624
    Enabling with EN_AIRT, 6628
    Enabling with EN_IRT, 6626
    Event ID, 1818
    Instructions, 6623
    Settings, 1679
Asynchronous error event
    Delaying with DIS_AIRT, 4721
    Disabling with DIS_IRT, 4718
    Enabling with EN_AIRT, 4722
    Enabling with EN_IRT, 4720
AT , 2218
ATAN, 3297, 3595, 3990, 4179, 4369, 5467, 5679, 5993, 6169, 6304
ATH, 4537, 6443
ATTACH, 1583, 4691
ATTR_DB, 4805
Audit log, 805, 807
Authentication, 1236, 1329
Authentication methods, 785, 786
Authorization, 8935
    Assignment, 8249, 8271

Changing the name, 8254
Configuring, 8263
Creating, 8248, 8269
Deleting, 8254
Managing, 8253
Autocompletion
Function, 2540
Insert tag, 2540, 2541
Inserting an instruction, 2541
Automatic commissioning, 1461
Automatic mode of the PROFINET IO controller, 1989
Automatic operation of accompanying signals, 987
Automatically determining connection path, 605
Automation system, 8374, 8398
Local, 10967
Remote, 10967
Setting up, 8374
Autonegotiation, 1460
Disabled, 8441, 8490, 8543
Availability
Object for Basic Panels, 8075
Availability for specific devices
Screen, 7963
AW, 2477, 4075, 6074
AWP command, 916, 917, 951, 952
AWP_In_Variable, 955
AWP_Out_Variable, 918, 953
AX, 2477
axis
Axis type, 10509, 10606
Axis
Absolute encoder adjustment, 9390, 9465, 10513, 10610
Axis types, 9386, 9461
End of rotary axis, 10511, 10608
Hysteresis, 9393, 9468
Minimum edge distance, 10613
Minimum edge-to-edge distance, 9395, 9471
Reference point coordinate, 9389, 9464, 10512, 10609
Simulation speed, 9394, 9470
Software limit switch, 9386, 9462, 10509, 10606
Start speed for reference point approach, 10516, 10613
Type of reference point approach, 10511, 10608
Axis and command table technology object
List of ErrorIDs and ErrorInfo, 10132
Axis data block
Loading, 10809
Saving, 10809
Axis DB
Content, 7093
Function, 7093, 10808
Axis does not move
Easy Motion Control, 10817
Axis moves without travel command
Easy Motion Control, 10818
Axis type, 10509, 10606
Axis types, 9386, 9461

B

Background color
Dynamization, 8047
Background OB (OB 90) (S7-300, S7-400), 1806
Backing up, 8936
HMI device data, 8910
Backup, 1659
Data of the HMI device, 8894
Deleting, 8879
Rename, 8879
Backup from online device, 10859
Bandwidth (IRT communication), 1477
Bar, 8076, 8922
Color transition, 8076
Display limit lines, 8076
Bar segment
Defining, 8077
Base time
Calculate base time from local time with LT_BT, 6425
Basic mode, 2995
Basic Panel
Area pointer, 8428
Communication drivers, 8425
Display and operating object, 8075
ETHERNET, 8427
IF1B, 8427
Interface, 8427
Mitsubishi, 8425
Modicon Modbus, 8425
Omron, 8425
OPC, 8425
Runtime start, 8886, 8907
Basic parameters
High_Speed_Counter, 9029
Technology module, 9046
Basic submodule, 1848
Basics
Migration, 143
BCC, 996
BCD, 4012, 6015

BCDCPL, 3443, 3744, 3896, 4325, 4487, 5579, 5793, 5901, 6265, 6395
BE, 4066, 6066
BEC, 4068, 6067
Behavior
Button, 8931
Date/time field, 8923
Graphic IO field, 8928
IO field, 8925
Simple recipe view, 8234, 8954
Switch, 8933
Symbolic IO field, 8934
Behavior of a stepper motor, 10753
Behavior of the digital outputs
Active on reaching the comparison value for pulse duration down, 9135, 9210
Active on reaching the comparison value for pulse duration up, 9135, 9210
Active on reaching the comparison value for pulse duration up/down, 9134, 9209
Basic conditions, 9137
Boundary conditions, 9211
Default setting, 9132
in counting modes, 9130
in isochrone mode, 9132
in measuring modes, 9141
No comparison, 9133, 9208
On when counter value <= comparison value, 9134, 9209
On when counter value >= comparison value, 9133, 9209
Switch at comparison values, 9136
Behavior of the Digital Outputs
Default setting, 9176
in counting modes, 9207
Benefits of using TeleService, 10918
BEU, 4069, 6069
Bias time Ti, 1400
Bidirectional data traffic, 979
Binary result, 2453, 2466
Bit (0, 1)
Graphics list, 8039
Text list, 8031
Bit array
Reset, 5752, 5853, 6230, 6357
Reset a bit array in the I/O area, 5544, 5754, 5854, 6358
Reset in the I/O area, 6231
Resetting, 5542
Set, 5747, 5847, 6226, 6352
Set in the I/O area, 5538, 5848, 6227, 6353
Set in the I/O area , 5748
Setting, 5537
Bit field
Reset, 3457
Resetting, 3163
Set, 3456
Setting, 3162
Bit logic operation
AND, 3899, 3900, 3909, 3910, 5903, 5905, 5913, 5914
EXCLUSIVE OR, 3905, 3906, 3913, 3914, 5909, 5911, 5918, 5919
OR, 3902, 3903, 3908, 3911, 3912, 5906, 5908, 5912, 5916, 5917
Bit logic operations
AND, 3445, 3447, 5581, 5583
EXCLUSIVE OR, 3450, 5586
Insert input, 3451, 5586
OR, 3447, 3448, 5583, 5584
Bit mask, 3434, 3735, 3881, 4316, 4473, 5563, 5773, 5878, 6250, 6378
Bit memories
Enabling the display of retentive bit memories, 2945
Bit memory, 1842
Bit number (0 - 31)
Graphics list , 8041
Text list, 8032
Bit string, 2240, 2241, 2242
64-bit, 2243
Bits
Count, 3444, 4326, 4488, 5580, 6266, 6396
Counting, 3746, 3897, 5795, 5902
BITSUM, 3444, 3746, 3897, 4326, 4488, 5580, 5795, 5902, 6266, 6396
Bit-triggered trends, 8468, 8625
BLD, 4110, 6118
BLKMOV, 50, 3347, 3646, 3818, 4230, 4401, 5470, 5683, 5817, 6171, 6308
Block
Block, 3646, 5683
Call, 5527, 5738
Call conditionally, 4072, 6071
Changing passwords for know-how protected blocks, 2935
Closing, 2519
Comparing, 2873
Comparing code blocks, 2871
Comparing data blocks, 2872
Comparison, 2869
Compiling, 2898
Compiling in the program editor, 2902
Compiling in the project tree, 2901

Consistency check, 2898, 2950, 2956
Consistency check in the call structure, 2951
Copying, 2506, 2509
Deleting offline, 2521
Deleting online, 2521
Displaying properties, 2517
Download to device, 2903, 2907
Downloading blocks to a memory card, 2927
Downloading to a memory card, 2912
Downloading to device, 2921, 2922
Downloading to device in RUN operating mode, 2906
Editing properties, 2517
Entering a comment, 2511
Entering a title, 2510
exporting to an external source file, 2865
Fill, 3326, 3352, 3625, 3652, 3802, 3824, 4194, 4235, 4384, 4406, 5476, 5689, 5823, 6177, 6315
Fill uninterruptible, 3328, 3627, 3804, 4196, 4386
Find and open, 2518
Inserting, 2506
Know-how protection, 2929
Leave, 4271, 6202
Loading to device in RUN operating state, 2922
Move, 3616, 3793, 3818, 4187, 4230, 4376, 4401, 5817, 6171, 6308
Move uninterruptible, 3324, 3350, 3623, 3649, 3800, 3821, 4192, 4233, 4382, 4404, 5474, 5686, 6174, 6312
Moving, 3318, 3347, 5470
Opening, 2518
Opening know-how protected blocks, 2933
Optimized access, 2175, 2177
Pasting, 2510
Printing know-how protected blocks, 2934
Properties, 2512
Removing copy protection, 2931
Renaming, 2520
Reset bit, 5752
Reset byte, 5755, 5856
Reset in the I/O area, 5754, 5854
Saving, 2519
Set bit, 5747, 5847
Set byte array, 5750, 5850
Set in the I/O area, 5848
Setting up copy protection, 2931
Time stamp, 2515
Types, 2169
Upload from device, 2903, 2921
Uploading blocks from a memory card, 2913, 2928
Uploading blocks from device, 2911, 2926
Using a library, 2504
BLOCK, 2279
Block access
Data block, 2839
Block call
Basics, 2180
Calling as single instance or multiple instance, 2182
Correcting the call type, 2590, 2631, 2675
GRAPH, 2735
Inserting, 2587, 2629, 2673, 2706, 2707, 2709, 2710, 2711, 2712, 2812, 2813
Multi-instance, 2183
Nesting depth, 2181
Pasting, 2671
Single instance, 2183
Updating, 2589, 2630, 2674, 2714, 2814
Block check character, 996, 1053
Block comment
Hiding, 2532
Showing, 2532
Block comparison
Basics, 2869
Changing the block, 2893
Comparing code blocks, 2871
Comparing data blocks, 2872
Detailed comparison, 2875, 2876, 2880, 2883, 2886
Execute action, 377
Navigation, 2892
Synchronize scrolling, 2893
Updating comparison results, 374, 2894
Block consistency
Checking, 2951
Checking in the dependency structure, 2956
Block end, 4066, 4068, 4069, 6066, 6067, 6069
Block folder, 48, 2500
Block interface
Declaring ARRAY, 2553
Declaring PLC data type, 2555
Declaring STRUCT, 2554
Declaring tags, 2550, 2552, 2556
GRAPH, 2741
Hiding, 2532
Importing and exporting tags, 2567
Multi-instance, 2557
Purpose of tag declaration, 2544
Retentivity, 2561
Showing, 2532
Structure, 2544
Tag properties, 2560, 2562
Updating, 2557

Valid data types, 2548, 2549
Block parameters, 2185, 2187, 2189, 2192, 2193, 2194, 2195, 2196
Basics, 1629
Block interface, 2544
Block property
Displaying, 2517
Editing, 2517
Function, 2512
Overview, 2512
Block stack, 1658, 1665
BLOCK_DB_TO_WORD, 2447
Blocking control loop calls, 7031
Bookmarks
Deleting, 2697
Function, 2695
Navigating, 2696
Setting, 2695
BOOL, 2240, 2289, 2322, 2375, 2391, 2420, 2430
BOOL_TO_, 2322, 2391, 2430
Borders
Placeholder for document information, 393
Specifying the print area, 392
Bounce-free switch, 10586
Boundaries, 1461, 8442, 8491, 8544
Boundary reached, 8361
BR, 2453, 2466, 3923, 5528, 5739, 5928
Brake cam track, 9344
Branch, 3915, 4258, 4260, 5920, 6190, 6192
Closing, 2606, 2789
Definition, 2604, 2646
Deleting, 2607, 2647, 2790
Inserting, 2606, 2647
Rules, 2605
Rules , 2646
BRCV, 5048
Breakpoint
Deleting, 2976
Disable, 2973
Enable, 2973
Navigating to the breakpoints, 2971
Resume program execution, 2974
Run to cursor, 2974
Setting, 2970
Setting the call environment, 2972
Step into, 2974
Step over, 2974
Testing in single step mode, 2969
Broadcast, 622, 662, 761, 769, 1003, 1087, 1300
With UDP, 650
BSEND, 5046
BTD, 4012, 6015
BTI, 4009, 6012
BUFFER parameter, Send_P2P, 5197, 7484
Bus cycles, 1395
BUSY for asynchronous instructions, 5367
BUSY signal, 1076
Button, 8099, 8931
Application, 8931
Behavior, 8931
Configuring, 8269
Configuring access protection, 8265
Define hotkey, 8101
Graphic, 8100
Layout, 8931
Mode, 8100
Operation, 8931
Text, 8100
Buttons and Switches
Library, 8766
BY, 4262
BYTE, 2240, 2289, 2323, 2375, 2392, 2421, 2431
BYTE_TO_, 2323, 2392, 2431
Bytes
Swap, 4198
Swap, 3330, 3629, 4388, 6305

C

CA certificate, 733, 736, 737, 10953
CA group certificate, 737
Cabling rules (PROFINET), 1460, 8441, 8490, 8543
CAD, 4023, 6026
CALC, 3264, 3560
Calculate, 2590, 2632, 3264, 3560
CALCULATE, 2590, 2632
CalibrateTouchScreen, 8302
Calibrating, 2120
Overview, 2119
Call, 4070, 4072, 4073, 5527, 5738, 6070, 6071, 6073
ABS_CTRL, 7126
ABS_CTRL_451, 7129
ABS_DIAG, 7127
ABS_DIAG_451, 7131
ABS_INIT, 7125
CAM_CTRL, 7157
CAM_CTRL_452, 7160
CAM_DIAG, 7159
CAM_DIAG_452, 7162
CAM_INIT, 7156
CAM_MSRM_452, 7163
CNT_CTL1, 7098
CNT_CTL2, 7101

CNT_CTRL, 7096
CNT2_CTR, 7112
CNT2RDPN, 7111
CNT2WRPN, 7111
DIAG_INF, 7098
DIAG_RD, 7113
EncoderAbsSensorDP, 7066
EncoderCPU314C, 7073
EncoderET200S1Count, 7069
EncoderET200S1SSI, 7064
EncoderFM350, 7071
EncoderFM450, 7062
EncoderIM174, 7076
EncoderSINAMICS, 7078
EncoderSM338, 7067
EncoderUniversal, 7080
High_Speed_Counter, 4920
MC_Control, 7050
MC_GearIn, 7056
MC_Home, 7046
MC_Init, 7034
MC_MoveAbsolute, 7037
MC_MoveJog, 7042
MC_MoveRelative, 7039
MC_Simulation, 7051
MC_StopMotion, 7048
OutputCPU314C, 7084
OutputET200S2AO, 7083
OutputIM174, 7088
OutputIM178, 7082
OutputMM4_DP, 7087
OutputSINAMICS, 7092
OutputSM332, 7085
OutputSM432, 7083
OutputUniversal, 7086
CALL, 4070, 5527, 5738, 6070
Call data, 6944
Call environment
Basics, 2968
Setting for blocks, 2969
Setting for breakpoints, 2972
Call hierarchy, 2181
Call structure, 1839, 1841, 2945
Meaning of symbols, 2947
Displaying, 2949
Introduction, 2945
Setting view options, 2950
Structure, 2948
Callback to a number specified during connection establishment, 10935
Calling FAQs for a module, 491
Calling manuals for a module, 491
Cam
Effective direction, 9412, 9488
Hardware interrupt, 9414, 9490
Switching characteristics, 9413, 9489
Cam cycle time, 9406, 9482
Cam data
Reading, 9357
Writing, 9356
Cam tracks, 9342
CAM_CTRL, 7157
Call, 7157
Data blocks used, 7157
Description, 7157
Error response, 7158
Function switches, 7176
Job status, 7176
Jobs, 7175
Operating principle, 7157
Return values, 7158
CAM_CTRL_452, 7160
Call, 7160
Command status, 7177
Data blocks used, 7160
Description, 7160
Error response, 7161
Function switches, 7177
Jobs, 7176
Operating principle, 7160
Parameters, 7161
Return values, 7161
CAM_DIAG
Call, 7159
Commands, 7159
Data blocks used, 7159
Description, 7158
Error response, 7159
Operating principle, 7158
Return values, 7159
Startup, 7159
CAM_DIAG_452
Call, 7162
Data blocks used, 7162
Description, 7162
Error response, 7163
Jobs, 7162
Operating principle, 7162
Parameters, 7162
Return values, 7162
Startup, 7163
CAM_INIT
Call, 7156
Description, 7156

Parameters, 7156
Return values, 7157
CAM_MSRM_452, 7163
Call, 7163
Data blocks used, 7163
Description, 7163
Error response, 7164
Parameters, 7163
Return values, 7164
Startup, 7164
Cams
Activation time, 9409, 9485
Cam "valid", 9407, 9483
Cam data, 9406, 9482
Cam end for position-based cams, 9408, 9484
Cam end for time-based cams, 9409, 9485
Cam start for position-based cams, 9408, 9484
Cam start for time-based cams, 9409, 9485
Cam type, 9408, 9484
Directional reversal, 9346
Dynamic adjustment, 9412, 9487
Lead time, 9410, 9486
Number of cams, 9406, 9482
Track number, 9407, 9483
CAN_DINT, 4708
CAN_TINT, 4702
Cancel, 10472, 10563
Incremental approach, 10487, 10579
Jog, 10475
Jogging, 10566
Reference point approach, 10480, 10571
Cancel gate function, 9103
Cancel set actual value, 10489, 10581
Data in the channel DB, 10581
Canceling
Zero offset , 10584
Canceling a calibration, 2122
Canceling printing, 398
Cancelling the function
Undo set process value, 9363, 9439
Capture, 8981, 8982, 8987, 8989, 9035, 9058, 9064
CAR, 4003, 6006
Cascaded counting function, 2028
CASE, 4260, 6192
Catalog, (see Hardware catalog)
CAW, 4022, 6025
CC 0, 2453, 2458, 2459, 2460, 2462, 2466, 2470
CC 1, 2453, 2458, 2459, 2460, 2462, 2466, 2470
CD, 3237, 3534, 3954, 5430, 5644, 5957
CDB, 4061, 6062
CEIL, 3358, 3657, 4241, 4413, 5482, 5694, 6182, 6320
Central racks
Selecting from the hardware catalog, 872
Certificate, 734, 786
Exporting, 733
Importing, 733
Renewing, 736
Replace, 737
Replacing, 737
self-signed, 736
signed by certificate authority, 736
Certificate authority, 733, 734
Certificate manager, 734
CH_DIAG
Block diagram, 7211
Input parameters, 7209
Instruction, 7208
Output parameters, 7209
CH_DIAG_455
Block diagram, 7291
Input parameters, 7289
Instruction, 7288
Output parameters, 7289
Change, 8354
Change cam edges, 9368, 9443
Canceling, 9369, 9445
Data used in the channel DB, 9368, 9444
Definition, 9368, 9443
Effects, 9369, 9444
Requirements, 9368, 9443
Steps, 9368, 9444
Change word order
Modicon MODBUS TCP/IP, 8704
ChangeConnection, 8344
Changeover contact, 2064
Changeover contact sensor type, 2063
Changeover difference, 10503, 10600
Changeover difference minus, 10504, 10601
Changeover difference plus, 10504, 10601
Changing
Displayed name of user group, 8254
Incremental dimension tables, 10465, 10556
Logoff time in runtime, 8261
Machine data, 10463, 10554
Name of the user, 8260
Object size, 7987
Password, 8252
User group in runtime, 8261
Users, 8940
Changing a port interconnection
Graphic view, 718
Changing and displaying operating mode (example), 8281

Changing connection partners, 617
Changing connections interactively, 618
Changing from DPV0 to DPV1, 1388
Changing partner port, 1505
Changing partner ports during operation, 1505
Changing the configuration online, 10850
Changing the connection path in the Inspector window, 619
Changing the device configuration online, 10850
Changing the name
    Authorization, 8254
    User group, 8253
    Users, 8252
Changing the path of a configured connection, 618
Channel address, (see Module start address)
Channel data block, 7164
Channel DB
    Configuration, 7133, 7165
    Content, 7134, 7165
    Purpose, 7133, 7164
Channel diagnostics, 2157
    Diagnostic data, 6633
Channel setting
    FM 350-2, 9166
Channel-specific diagnostics, 2157
Char
    Array tag, 67
CHAR, 2263, 2317, 2367, 2389, 2415, 2429, 2445
CHAR_TO_, 2367, 2415, 2445
Character, 2263
Character delay time, 992
Character delay time CDT, 1003, 1088
Character set
    in message texts, 1073
Character string, 2264, 2265
    Combine character strings with CONCAT, 6447
    Combining character strings with CONCAT, 4542
    Comparing string tags with S_COMP, 4519, 6440
    Converting from a hexadecimal number with HTA, 4539, 6445
    Converting number to character string with VAL_STRG, 4524
    Converting numeric character string with STRG_VAL, 4522
    Converting to a hexadecimal number with ATH, 4537, 6443
    Converting with S_CONV, 4520
    Copying characters to character string with Chars_TO_Strg, 4529
    Deleting characters with DELETE, 4545, 6450
    Determine maximum length with MAX_LEN, 4530
    Determining length with LEN, 4541, 6446
    Finding a character with FIND, 4547
    Finding characters with FIND, 6452
    Inserting characters with INSERT, 4546, 6450
    Move operation, 6441
    Reading out left character with LEFT, 4543
    Reading out middle character with MID, 4544
    Reading out right character with RIGHT, 4543
    Reading the left character with LEFT, 6448
    Reading the middle character with MID, 6449
    Reading the right character with RIGHT, 6448
    Replacing character with REPLACE, 4546
    Replacing characters with REPLACE, 6451
Characters, 31
Chars_TO_Strg, 4529
Check whether project can be migrated, 126
Checkback signals for the diagnostics, 10496, 10593
    Data in the channel DB, 10497, 10594
    Sequence, 10496, 10594
Checkback signals for the positioning, 10495, 10593
    Data in the channel DB, 10496, 10593
    Sequence, 10496, 10593
Checkback signals in the channel DB
    End of a positioning operation, 10473, 10564
CHECKBACK_SIGNALS
    CNT2_CTR, 7120
Checking
    Device version, 8876
Checking the connection, 5117
Checklist for troubleshooting the modem, 10977
CiR, 5503, 5715, 5840, 6210, 6336
CiR process
    Controlling, 5715, 5840, 6210
Circle, 8086
    Radius, 8086
CJ_T_PAR
    Input parameters, 7218
    Instruction, 7218
    Output parameters, 7219
CJ_T_PAR_455
    Input parameters, 7297
    Instruction, 7297
    Output parameters, 7298
CJ1, 8738
CJ2, 8738
Class of Service, 1264
Cleaning project for migration, 128
ClearAlarmBuffer, 8312
ClearAlarmBufferProTool, 8313
ClearAlarmBufferProtoolLegacy, 8313
cleared, 8353
Click, 8359

Click when flashing, 8359
Clock function, 1882, 1883
Clock parameters, 1685
Reading the time, 1881
Setting the time, 1881
Clock memory, 1586, 1608, 1686, 1860, 1884
Clock parameters, 1882, 1883, 1967
Close branch, 2723
Closing
Runtime, 8909
CLR, 3922, 5927
CMP <, 3249, 3545, 5438, 5651
CMP <=, 3245, 3542, 5436, 5648
CMP <>, 3241, 3538, 5433, 5646
CMP ==, 3238, 3536, 5432, 5645
CMP >, 3247, 3544, 4494, 4495, 4500, 4501, 5437, 5650, 6400, 6401, 6406
CMP >=, 3243, 3540, 5435, 5647
CNT_CTL1, 7098
Call, 7098
Description, 7098
Error response, 7100
Jobs, 7099
Operating principle, 7098
Parameter, 7098
Startup characteristics, 7100
Value transfer, 7107
CNT_CTL2, 7100
Call, 7101
Description, 7100
Operating principle, 7100
CNT_CTRL, 7096
Call, 7096
Description, 7096
Error response, 7097
Jobs, 7097
Operating principle, 7096
Parameter, 7096
Startup characteristics, 7097
CNT2_CTR, 7112
Call, 7112
CHECKBACK_SIGNALS, 7120
CONTROL_SIGNALS, 7120
Description, 7112
Operating principle, 7112
Parameters, 7112
CNT2RDPN, 7111
Call, 7111
Description, 7111
Operating principle, 7111
Parameters, 7111
Read job, 7123
Status of the write job, 7124
CNT2WRPN, 7110
Call, 7111
Description, 7110
Operating principle, 7110
Parameters, 7111
Status of the write job, 7122
Write command, 7121
Code block, 1839
Code transparency, 994
Coil, 3159, 5377
Negate, 3160
Cold restart, 1642
Cold restart (S7-300, S7-400), 1807
Color
Diagnostics of Ethernet cables, 2087
Diagnostics of ports, 2087
Color transition
Bar, 8076
Column
Alarm view, 8091
Comfort Panel
S7-1200, 74
Command management, 7125
Command message frame, 1061
Command status
CAM_CTRL_452, 7177
Command table technology object
Tools, 10055
Add new object, 10056
Basic parameters, 10057
Command table configuration, 10059
Command.Command[1...32] tags, 10161
Configuration window icons, 10057
Configuring activate warnings, 10059
Configuring duration, 10061
Configuring position / travel path, 10060
Configuring the command type, 10059
Configuring the next step, 10061
Configuring the step code, 10061
Configuring use axis parameters of, 10059
Configuring velocity, 10060
Extended parameters, 10057
General configuration, 10058
Usage, 10055
Commands
ABS_DIAG, 7128
CAM_DIAG, 7159
Comment
Area pointer, 8419
Comment section, 2676

Comments
Inserting in SCL program, 2715
Commissioning
High_Speed_Counter, 9042
HMI device, 8903
Technology module, 9072
communication
Status data, 6709
Communication, 1613, 1888, 8373
Basics, 8374
Cycle load, 1870
Definition, 8373
Open User Communication, 599, 7410
Query architecture, 5181, 7469
S7 1200, 8477
S7 1500, 8429
S7 200, 8591
S7 300, 8531
S7 400, 8531
SIMATIC LOGO!, 8618
Third-party drivers, 8630
using area pointers, 8380
Communication connection, 563
Communication diagnostics, 2157
Communication drivers, 8380
Allen-Bradley, 8631
Basic Panel, 8425
Mitsubishi, 8672
Modicon Modbus, 8700
Omron, 8728
Communication events, 1826
Communication interfaces
Programming, 5179, 7467
Communication load, 1586, 1860, 1870
Communication module (CM)
Data reception, 5197, 7484
Programming, 5179, 7467
Communication network, 8398
PROFINET, 8399
Communication partners
HMI device, 8373
Networking, 8402
PLC, 8373
SIMATIC S7, 8477, 8531, 8588, 8597, 8600
Communication properties of a module, 2150
Communication types
Allen-Bradley, 8636
Connectable PLCs, 8707, 8718
Enabled, 8677, 8689, 8706, 8717, 8733
Ethernet/IP, 8636
Restriction, 8706, 8717
Communication via PUT/GET instructions
Basic information on PUT/GET instructions, 700
Creating and assigning parameters to a connection, 705
Deleting an interconnection, 706
Overview of connection configuration, 701
Requirements, 700
Starting connection parameter assignment, 704
Communications
Connection resources, 1688
Cycle load, 1591
Communications error OB (OB 87) (S7-300, S7-400), 1802
Communications instruction
"AS_DIAL", 10967, 10968
"AS_MAIL", 10971
"PG_DIAL", 10966, 10967
"SMS_SEND", 10969
Communications load, 1591
Communications module (CM), 5297
Communications modules, 1599
Properties, 1599
Communications port
Configuring, 1600
Communications processor
SIMATIC NET PC software, 1969
STEP 7 V5 + STEP 7 Professional, 1992
Communications protocol
Defining, 1603
Overview, 1602
Compact, 8709, 8722
CompactLogix, 8639
Comparator operations
Time tags, 4502, 6407
Compare
Bit mask, 3434, 3735, 3881, 4316, 4473, 5563, 5773, 5878, 6250, 6378
Character strings with S_COMP, 4519, 6440
Equal, 3238, 3536, 5432, 5645
Floating-point numbers, 3958, 5962
Greater or equal, 3243, 3540, 5435, 5647
Greater than, 3247, 3544, 4494, 4495, 4500, 4501, 5437, 5650, 6400, 6401, 6406
Integers (16 bit), 3955, 5959
Integers (32 bit), 3957, 5960
Less or equal, 3245, 3542, 5436, 5648
Less than, 3249, 3545, 5438, 5651
Not equal, 3241, 3538, 5433, 5646
Compare editor
Changing the view, 378
Filtering the view, 373
Overview, 366

Showing and hiding columns, 372
Specifying actions, 376
Compare offline/online
Automatic device assignment, 722
Compare scan matrix, 3436, 3737, 3886, 4318, 4477, 5565, 5775, 5883, 6252, 6380
Comparing, 50
Comparing library objects, 416
Comparison value, 8998, 9037, 9060, 9066, 9131, 9207
Counting, 9037, 9066
Measuring, 9060
Compatibility, 39
Backward compatibility of projects, 333
Global libraries from older product versions, 429
Projects from older program versions, 333
Projects with add-on software, 334
S7-1200 CPU firmware versions, 1637
WinCC version, 8871
Compile
Block, 2901
Compiling
Address parameters, 73
Blocks, 2898
Consistency check, 2898
Correcting compilation errors, 2902
Migrated project, 146
Project, 8882
Components
Alarm, 8157
Commissioning, 8645, 8666, 8684, 8694, 8711, 8723, 8740
Compress, 5709, 6329
COMPRESS, 5497, 5709, 5833, 6204, 6329
Compressing, 5497, 5833, 6204
Memory in the Online and Diagnostics view, 2160
Computer connection RK512, 1061
Command message frame, 1061, 1062
Fetching data, 1067
Response message frame, 1061
Sending Data, 1064
CONCAT, 4542
Condition
Inserting a permanent instruction, 2809
Introduction, 2739
Programming a supervision, 2801
Programming an interlock, 2800
Programming transitions, 2802
CONF_DB see configuration data block, 7817
Configuration
Distributed I/O, 1339
DP slave, 1340, 1353, 1423
DP slave, simple, 1343, 1414
Hardware, 494
I-slave, 1345
Loading to device, 1553
Loading to PG/PC, 1554
Modules, 864
Tag, 8121
Configuration control, 885, 896, 902, 2042, 2054
Configuration data block, 7817
Configuration file, 1160
Configuration mode, 1220
Configuration phase, 8902
Configuration rights, 741
Configure a PROFIBUS subnet
Matching parameters to one another, 529
Setting bus parameters, 529
Configure acknowledgment
Alarm message, 8469, 8519, 8577, 8616, 8628, 8670, 8698, 8727, 8743
Configure alarms
Allen-Bradley, 8669
Data types, 8697
Non-integrated connection, 8697
Special considerations, 8697
Configured
Loop-in-alarm, 8167
Configuring, 8163, 8171
Access protection, 8269
Alarm acknowledgment, 8177, 8178, 8179
Alarm group, 8161
Alarm view, 8173
Alarms, 8518, 8615, 8627
Analog alarm, 8163, 8170
Analog alarms, 8159
Authorization, 8263
Connections, 8579, 8580, 8582, 8591, 8619
Controller alarms, 8159
CP as I-slave, 1373
CPU as I-slave, 1373
Diagnostic addresses on the I-slave, 1369
Direct data exchange, 1377
Discrete alarm, 8161, 8169
Discrete alarms, 8159
DP slave as I-slave, 1374
Event-driven tasks, 8165
HART variable, 2062
I-slave, 1368, 1424
Layout of the alarm view, 8174
Multiple tags, 8121
PC station, offline, 1160
PC station, online, 1158
Recipe view, 8230

Rectangle, 8024
System events, 8159
Trend view for values from the PLC, 8147
Configuring a connection
Allen-Bradley DF1, 8647
Allen-Bradley EtherNet/IP, 8632
Mitsubishi FX, 8685
Mitsubishi MC TCP/IP, 8673
Modicon Modbus RTU, 8712
Modicon Modbus TCP, 8700
Omron Host Link, 8729
Configuring a PROFIBUS subnet
Meaning of the bus parameters, 530
Configuring connections, 564
Dialog guided, 600, 608
Graphic, 600
Configuring internal subnets manually, 836
Configuring IP network nodes manually, 835
Configuring MAC network nodes manually, 836
Configuring the network with Ethernet, 534, 8485, 8540
Creating private subnets, 536, 8486, 8541
Linking networks, 536, 8486, 8541
Relationship between IP address and subnet mask, 535, 8485, 8540
Setting the IP address, 535, 8485, 8540
Setting the subnet mask, 535, 8485, 8540
Configuring the remote modem, 10931
Connecting
PLC, 8636, 8676, 8690, 8705, 8733
Connecting a TS Adapter with an external modem, 10931
Connecting a TS Adapter with an internal modem, 10931
Connecting cable
6XV1440 - 2P for Mitsubishi PG protocol, 8690
Allen-Bradley cable 1784-CP10, 8656
Mitsubishi FX, 8690
Connecting cable 9-pole
Sub D, RS 422, 8655
Connecting cables
Point-to-point cable 1,
Point-to-point cable 2, 8720
Connection, 541, 554, 601, 8379
Address details, 2136
Allen-Bradley DF1, 8647, 8650
Allen-Bradley EtherNet/IP, 8632
Area pointer, 8417, 8422
canceling to partner outside of the local S7 station with X_ABORT, 7931
canceling to partner within the local S7 station with I_ABORT, 7916
Configuring, 564, 8403, 8407, 8632, 8647, 8673, 8685, 8700, 8712, 8729
Configuring a connection when there is no or no clear network assignment, 546, 608
Configuring without networking, 613
Creating, 600, 682, 8403, 8407
Deleting, 685
Diagnosing with C_DIAG, 6743
Displaying assigned and reserved, 2154
E-mail, 563, 568, 571, 573
Establishing with TCON, 7431
FDL, 563, 568, 571, 573
Highlighting, 8405
Influence of the networking - response when networking is missing, 602
Integrated, 8403
integrated connection, 8112, 8414
ISO transport, 563, 568, 571, 573
ISO-on-TCP, 563, 568, 571, 573
Mitsubishi MC TCP/IP, 8673
Modicon, 8716
Modicon Modbus RTU, 8712
Modicon Modbus TCP, 8700
offline, 8458, 8508, 8566, 8605
Omron communication drivers, 8741
Omron Host Link, 8729
Parameters, 8634, 8648, 8674, 8687, 8702, 8714, 8731
Password, 8438, 8450, 8487, 8500
Programmed, 597, 599
Querying status with C_CNTRL, 7381
S7 200, 8591, 8619
Table, 8402
TCP, 563, 568, 571, 573
to a specific station in any project, 596
UDP, 563, 568, 571, 573
Unspecified, 601
without specifying a port, 596
Connection cable
Allen-Bradley cable 1747-CP3, 8657
Allen-Bradley cable 1761-CBL-PM02, 8658
Modicon, 8716
Multipoint cable 1:MP/TP/PC, 8735
Multipoint cable 2:RS422, MP/TP/PC, 8736
Omron Host Link, 8733
Point-to-point cable PP2 for Omron, 8736, 8737
Connection configuration
Connection parameters, 678
Overview, 677
Starting, 681
Connection configuration with an OPC server, 595

Connection description
Changing parameter values, 695
Data block , 688, 691, 692
Structure, 689, 691, 693
Connection details, 2135
Connection diagnostics
Detailed, 2133
Overview, 2131
Connection display in the network view, 604
Connection ID, 7822
Possible values, 7822
Connection information, 2134
Diagnostics, 8387
Connection mechanisms, 1613, 1888
Connection modes, 1166
Connection parameter assignment of PUT/GET instructions, 704
Connection parameters
Allen-Bradley DF1, 8648
Allen-Bradley EtherNet/IP, 8634
Mitsubishi FX, 8687
Mitsubishi MC TCP/IP, 8674
Modicon Modbus RTU, 8714
Modicon Modbus TCP, 8702
Omron Host Link, 8731
Connection parameters of PUT/GET instructions, 703
Connection path
Highlighting in the network view, 614
Connection resource, 541, 554
Connection resources, 579, 2154
Online, 2134, 2135
Connection rules, 851, 860
Connection status
Displaying using icons, 2132
Connection table, 604, 616
Connection types
Parameter fields for, 7822
Connections
Allen-Bradley DF1, 8651, 8652
Configuring, 8579, 8580, 8582, 8591, 8619
Configuring graphically, 605
Editor, 8579, 8580, 8582
Consistency
Slot rules, 496
Consistency check, 832, 2950, 2951, 2956
Introduction, 2950, 2956
local, 732
project-wide, 733
Consistency check before migration, 127
Constant
Data types, 2206
Definition, 2204
Global constant, 2205, 2206, 2208, 2488, 2489, 2492
Local constant, 2205, 2206, 2208, 2544, 2550, 2560
Non-typed constant, 2204
Symbolic constant, 2205
Typed constant, 2204
Constant bus cycle time, 1397
Activation in the PROFIBUS network, 1406
Composition of bus cycles, 1395
Definition, 1395
Requirements, 1396
Time for constant bus cycle time DP cycle, 1396
Constants, 3969, 5973
CONT_C
Block diagram, 6796
Input parameters, 6796
Mode of operation, 6794
Output parameters, 6798
CONT_S
Block diagram, 6801
Input parameters, 6801
Instruction, 6799
Mode of operation, 6799
Output parameters, 6803
Context filter, 497
Continuation FETCH frame, 1069
Continuation message frame, 1061
Continuation SEND message frame, 1065
CONTINUE, 4267, 6199
Control characters, 1053
in message texts, 1073
Control data record, 896
Control element
Alarm view, 8091
simple alarm view, 8945
Simple alarm view, 8182
Trend view, 8088
Control interface, 9075
1Count24V for counting modes, 9241
1Count24V for measuring modes, 9259
1Count24V for position feedback, 9270
1Count5V for counting modes, 9302
1Count5V for measuring modes, 9317
1Count5V for position feedback, 9328
1SSI in standard mode, 10440, 10441
1STEP 5V, 10741
2PULSE (long), 10000
2PULSE (short), 9999
FM 350-1 for counter modes, 9146
FM 350-1 for measuring modes, 9151

FM 351, 10528
FM 352, 9416
FM 451, 10626
FM 452, 9492
Control jobs, 10748
Control mode 1, 10500, 10597
Control mode 2, 10500, 10598
Control mode 3, 10501, 10598
Control mode 4, 10502, 10599
Control signals
ABS_CTRL, 7126
ABS_CTRL_451, 7130
Writing, 7125, 7129
CONTROL_SIGNALS
CNT2_CTR, 7120
Controlled positioning
analog with ANALOG, 7322
Digital with DIGITAL, 7332
Controller alarms
Configuring, 8159
Controller data, 8784
Initializing, 8783, 8785
Controlling counter from user program with COUNT, 7341
ControlLogix, 8639
Control-timer alarm, 3426, 3430, 3728, 3731, 3873, 3877, 4309, 4312, 4466, 4469, 5556, 5559, 5766, 5769, 5871, 5874, 6243, 6246, 6371, 6374
Conversion, 2286, 2372, 2417
Explicit, 2322, 2323, 2329, 2332, 2335, 2337, 2339, 2341, 2344, 2346, 2349, 2351, 2352, 2354, 2355, 2357, 2358, 2360, 2361, 2362, 2364, 2365, 2367, 2368, 2369, 2371, 2391, 2392, 2398, 2400, 2401, 2403, 2405, 2407, 2408, 2410, 2411, 2412, 2413, 2414, 2415, 2416, 2430, 2431, 2436, 2438, 2439, 2440, 2441, 2443, 2444, 2445, 2446, 2447
Explicit , 2325, 2327, 2393, 2395, 2432, 2434
Implicit, 2289, 2292, 2293, 2294, 2295, 2296, 2297, 2298, 2300, 2301, 2302, 2304, 2305, 2306, 2307, 2308, 2310, 2311, 2312, 2313, 2314, 2316, 2317, 2318, 2319, 2320, 2375, 2376, 2377, 2378, 2379, 2380, 2381, 2382, 2383, 2384, 2385, 2386, 2387, 2388, 2389, 2390, 2420, 2421, 2422, 2423, 2424, 2425, 2426, 2427, 2428, 2429, 2430
Conversion operations
Converting timers with T_CONV, 4503, 6408
Convert, 3355, 3654, 4009, 4011, 4013, 4014, 4015, 4238, 4410, 5479, 6012, 6014, 6016, 6017, 6018, 6180, 6318
CONVERT, 3355, 3654, 4238, 4410, 5479, 5692, 6180, 6318
Convert BCD to integer, 4012, 6015
Convert value, 5692
Converted name (PROFINET), 1447
Converting
A character string from a hexadecimal number with HTA, 6445
A character string to a hexadecimal number with ATH, 4537, 6443
Character string in array with Ctrl_TO_Chars, 4527
Character strings with S_CONV, 4520, 6441
Converting number to numeric character string with VAL_STRG, 4524
Converting numeric character string to number with STRG_VAL, 4522
Converting timers with T_CONV, 4503, 6408
Copying characters to numerical character string with Chars_TO_Strg, 4529
Hexadecimal number to character string with HTA, 4539
Converting an unspecified CPU, 176
Cookie, 921, 956
Coordinated transfer, 8216
With PLC, 8216
Coordination, 8380
Copy
Screen, 7968
Tag, 8120
Template, 7974
Copy protection, 2931
Copying
Adjusting screen size, 8865
Alarm indicator, 62
Alarm view, 62
Alarm window, 62
Color, 8864
Excel format, 8796
Font, 8864
Function key, 8865
Hardware component, 511
HMI device, 57
Invalid object, 8864
Linked objects, 8866, 8867
Principle, 8863
Screen, 8865
User-defined folders, 8864
Correction distance, 9346
CoS, 1264
Queue, 1264
COS, 3292, 3590, 3986, 4174, 4364, 5461, 5673, 5990, 6165, 6299
CoS (Class of Service), 1181

Cosine, 3292, 3590, 3986, 4174, 4364, 5461, 5673, 5990, 6165, 6299
Count
  Down, 3223, 3233, 3237, 3520, 3530, 3534, 3769, 4147, 4155, 4345, 5418, 5426, 5430, 5631, 5639, 5806, 6143, 6150, 6282
  Up, 3220, 3231, 3236, 3517, 3528, 3766, 4144, 4153, 4342, 5416, 5424, 5429, 5629, 5637, 5642, 6141, 6148, 6280
  Up and down, 3226, 3229, 3523, 3772, 4149, 4157, 4347, 5420, 5422, 5633, 5809, 6152, 6284
  Up and Down, 3526, 5635, 6146
COUNT, 7341
Count (8 DI NAMUR), 2026
Count (CPU 31xC), 1693
Count continuously, 9083
Count once, 9084
COUNTER, 2279, 3229, 4157, 5422, 6152
Counter cam track, 9344
Counter DB
  Configuration, 7101
  Contents, 7102, 7113
  Purpose, 7101, 7113
Counter input, 1595
Counter limits, 8980
Counter mode, 1595
Counter operations
  Control high-speed counters, 4918
  Enable, 3946, 5950
  High-speed counters, 4918
  Reset, 3950, 5954
  Set, 3951, 5955
Counter, high-speed, 1594
Counting
  Comparison value, 9037, 9066
  Continuous, 9083
  Down, 3954, 5644, 5957
  Periodic, 9085
  Single, 9084
  Up, 3533, 3953, 5804, 5956
Counting functions, 8980
Counting limits, 8982, 9033
Counting range, 9118, 9197
CountOfElements, 3346, 3645, 4229
Cover page
  Placeholder for document information, 393
CP
  As I-slave, 1373
  Point-to-point connection, 1027
  With separate MPI address, 881
CP 1242-7, 42, 1164
CP 1613, 33
CP 1623, 33
CP 343-2, 41
CP address, 7850
CP1, 8738
C-PLUG
  Formatting, 1254
CPM, 8738
CPR_OUT
  Input parameters, 6951
  Instruction, 6950
  Output parameters, 6951
CPU
  As I-slave, 1373
  Assigned and reserved connections, 2154
  Displaying the current LED status, 2097
  Fill level of all types of memory, 2098, 2099
  Identifying with the system status list, 6698
  Inserting a signal board, 942
  Logging on for alarms, 2159
  Not assigned connection resources, 2154
  Parameter, 1672
  Properties, 1585
  Reading characteristics with the system status list, 6693
  Reading date and time-of-day with RD_SYS_T, 4509
  Reading out a diagnostics buffer, 2101
  Reading the date and time-of-day with RD_SYS_T, 6413
  Relationship of protocol variant and data length, 7422
  Selecting from the hardware catalog, 503
  Setting the time-of-day with WR_SYS_T, 6412
  Setting time-of-day with WR_SYS_T, 4508
  Startup, 7127, 7131, 7158, 7161
  Switching operating mode, 2107
CPU 1516-3 PN/DP, 1871
CPU 1518-4 PN/DP, 1871
CPU control panel
  Display area, 2098, 2153
CPU data block
  Definition, 2174
  Deleting, 2522
CPU display, (See display)
CPU firmware, 1637
CPU hardware fault OB (OB 84) (S7-300, S7-400), 1790
CPU load through trace, 10906
CPU memory area
  Displaying, 2960
CPU memory reset, 1651
CPU messages - OPC server, 1988

CPU properties, 1586, 1860
CPU registers, 1741
CPU RUN, 1014, 1153
CPU STOP, 1014, 1153
CPU type
Allen-Bradley DF1, 8661
Allen-Bradley EtherNet/IP, 8639
FX3 series, 8682
Mitsubishi FX, 8692
Mitsubishi MC TCP/IP, 8682
Modicon Modbus RTU, 8722
Modicon Modbus TCP, 8709
Omron Host Link, 8738
CPU warm restart, 1153
CPU-STOP, 9046
CRC, 1003, 1088
Create
Connection, 8479
Global library, 8771
Group, 8014
Template, 7973
Users, 8250
Users in runtime, 8258
Create bit pattern for seven-segment display, 3894, 4485
CREATE_DB, 4798
Creating, 8223
Alarm group, 8161
Array, 8144
Connection, 8403, 8407, 8431, 8533
External tag, 8115
Infotext, 8163, 8165
Internal tag, 8117
Recipe, 8223
Recipe data record, 8955
Recipe data record on the HMI device, 8236, 8237
Screen, 7967
User group, 8248, 8271
Users, 8272, 8937, 8939
Creating a new type
Style, 8782
Creating a print preview, 394
Creating a route, 829
Creating a type
Screen, 8781
Script, 8780
Creating a watch table, 2998
Creating custom documentation, 329
Creating labels, 399
Crossing
Definition, 2607
Deleting, 2609
Inserting, 2608
rearranging, 2608
Cross-reference
Inspector window, 63
Cross-reference list, (See cross-references), (See cross-references)
Displaying, 2964, 8820
Overview, 2964, 8820
Settings, 2962, 8821
Sorting columns, 2962, 8821
Structure, 2962, 8821
Views, 2962, 8821
Cross-references
Displaying, 2964, 8820, 8823
Introduction, 2961, 8820
Linking, 8823
Replace, 8823
Uses, 2961, 8820
CRP_IN
Input parameters, 6949
Instruction, 6948
Output parameters, 6949
CS1, 8738
CSV file, 8796, 8798, 8807
CTD, 3223, 3520, 3769, 4147, 4345, 5418, 5631, 5806, 6143, 6282
CTRL_HSC, 4918
CTRL_PWM, 4766
CTS, 982
CTU, 3220, 3517, 3766, 4144, 4342, 5416, 5629, 5804, 6141, 6280
CTUD, 3226, 3523, 3772, 4149, 4347, 5420, 5633, 5809, 6146, 6284
CU, 3236, 3533, 3953, 3954, 5429, 5642, 5956, 5957
Cu10 sensor, 2050
Custom alarm classes, 8153
Cycle load, 1591, 1870
Cycle load due to communication, 1676
Cycle monitoring time, 3847, 4277, 4432, 5835, 6205, 6331
Cycle OB (S7-1500), 1891
Cycle time, 1586, 1590, 1860, 1869
display configured, 2091
display measured, 2096
Cyclic
Continuous, 8129
In operation, 8129
Cyclic behavior, 1675
Cyclic interrupt, 1621
OB 30 to OB 38 (S7-300, S7-400), 1760

Settings, 1682
Cyclic interrupt (S7-1500), 1901
Cyclic interrupt OB
Assigning parameters, 1630
Description, 1621
Parameter assignment, 1917
Querying parameters with QRY_CINT, 4696
Setting parameters with SET_CINT, 4694
Cyclic interrupt OBs (S7-1500), 1901
Cyclic operation, 8590, 8593, 8624
Cyclic program (OB 1) (S7-300, S7-400), 1754
Cyclic program (S7-1500), 1891
Cyclic program execution
Options for interrupting, 1614
Programming, 1614

D

D_ACT_DP, 885, 4589
D_ACT_DP (activating and deactivating IO devices), 1505
Data areas
Area pointer, 8420, 8746
Reading, 8420, 8746
Writing, 8420, 8746
Data backup, 101
HMI device, 8894, 8910
Data bit (DBX), 1572
Data block, 1839, 1844
Adjusting data values during commissioning, 2838, 2850, 2851, 2852, 2853
ARRAY data block, 2172, 2175, 2211, 2214, 2503, 2826, 2829, 2850, 2914
Based on a PLC data type , 2836
CONF_DB, 7452
CPU data block, 2174
Creating, 2503, 2829
Creating with CREA_DB, 6754
Creating with CREA_DBL, 6756
Creating with CREAT_DB, 6753
Creating with CREATE_DB, 4798
Declaration table, 2827
Declaring ARRAY, 2834
Declaring STRUCT, 2835
Default value, 2836
Deleting with DEL_DB, 6751
Deleting with DELETE_DB, 4806
Displaying values, 2849
Global data block, 2172, 2826
Importing and exporting tags, 2848
Instance data block, 2174, 2826
Loading block changes without reinitialization, 2831
Monitoring data values online, 2849
Open, 4059, 4060, 5495, 5496, 5707, 5708, 6060, 6061
Optimized access, 2175, 2177
Programming, 2826, 2833
Reading attributes with ATTR_DB, 4805
Reading from load memory with READ_DBL, 4801, 6747
Retentive behavior, 2839, 2840
Rule DB WS_RULES, 6439
Start value, 2836, 2837
Tag properties, 2841, 2843, 2845
Testing with TEST_DB, 6758
Updating, 2830
Using setting values, 2838, 2850, 2851, 2852, 2853
Writing to load memory with WRIT_DBL, 4803
Writing to the load memory with WRIT_DBL, 6749
Data block area
Fetching with FETCH_RK, 7904
Sending with SEND_PTP, 7894
Sending with SEND_RK, 7900
Data block register, 4061, 6062
Data blocks used
ABS_CTRL, 7126
ABS_CTRL_451, 7130
ABS_DIAG, 7127
ABS_DIAG_451, 7131
ABS_INIT, 7125
CAM_CTRL, 7157
CAM_CTRL_452, 7160
CAM_DIAG, 7159
CAM_DIAG_452, 7162
CAM_MSRM_452, 7163
Data byte (DBB), 1572
Data check, 9371, 9446
by the module, 9371, 9446
Data consistency, 5033, 7358
Data double word (DBD), 1572
Data Encryption Standard (DES), 788
Data error, 9161, 9218
Data exchange, 2650, 8373
DP slave, 1343, 1414
I-slave - DP master, 1345, 1415
Tags, 8379
Trends, 8468, 8625
using area pointers, 8380
Data exchange broadcast, (See Direct Data Exchange)

Data exchange over the AS-AS remote link
Communication instruction "AS_DIAL", 10968
Data flow, 8952
Data flow control, 975, 982, 984, 1601
Hardware, 985
Software, 985
Data in the channel DB
Cancel set actual value, 10489, 10581
Checkback signals for the diagnostics, 10497, 10594
Checkback signals for the positioning, 10496, 10593
Edge detection, 10587
Enable input, 10494, 10591
Encoder data, 10495, 10592
Incremental approach, 10486, 10487, 10578, 10580
Incremental dimension tables, 10466, 10556
Jog, 10474
Jogging, 10565
Length measurement, 10587
Loop traverse, 10492, 10589
Machine data, 10464, 10554
Position data, 10495, 10592
Reference point approach, 10479, 10570
Set actual value, 10488, 10580
Set reference point, 10490, 10584
Setting actual value on the fly, 10580
Zero offset, 10582
Data in the parameter DB
Edge detection, 10587
Enable input, 10591
End of a positioning operation, 10473, 10564
Incremental approach, 10486, 10578
Reference point approach, 10480, 10571
Data log, 947
Closing with DataLogClose, 4791
Creating with DataLogCreate, 4777
Creating with DataLogNewFile, 4795
Creating with DataLogTypedNewFile, 4796
Deleting on memory card with S7-1500, 38
Deleting with DataLogDelete, 4793
Empty with DataLogClear, 4788
Opening with DataLogOpen, 4784
Opening with DataLogTypedOpen, 4785
Writing with DataLogWrite, 4789
Data mailbox
For recipes, 8216
Data record, 8380
Asynchronously reading data record of a module with RD_DPARA, 4685, 6595
Dynamically transferring parameters with WR_PARM, 6600
Making available on I-device with PRVREC, 4606, 6506
Reading, 8237
Reading from configured system data with RD_DPARM, 4688, 6599
Reading of a module with RD_DPAR, 4683, 6593
Reading with RD_REC, 4594, 6491
Reading with RDREC, 4557, 6459
Receiving on I-device with RCVREC, 4603, 6503
Transfer, 8238
Transferring with WRREC, 4559, 6461
Writing and reading data records, 4681, 6591
Writing predefined parameters with WR_DPARM, 4689, 6601
Writing with WR_REC, 4599, 6496
Data record list, 8219
Data transmission rates, 975
Data transmission with the ET 200S Modbus master, 1098
Data transmission, trigger, 5195, 7482
Data type
Convert explicit, 3654, 4238, 4410
Convert explicitly, 6180
Explicit conversion, 3355
Allen-Bradley, 8670
Allen-Bradley EtherNet/IP, 8637
ANY, 2275
ARRAY, 2267, 2268
BOOL, 2240, 2289, 2322, 2375, 2391, 2420, 2430
BYTE, 2240, 2289, 2323, 2375, 2392, 2421, 2431
CHAR, 2263, 2317, 2367, 2389, 2415, 2429, 2445
Conversion, 2286, 2292, 2372, 2377, 2386, 2387, 2388, 2417, 2422, 2425, 2427, 2428
DATE, 2258, 2313, 2362, 2388, 2412, 2427, 2442
DINT, 2247, 2298, 2341, 2382, 2405, 2424, 2438
Discrete alarm, 8670, 8697, 8742
DT, 2260, 2310, 2358, 2428, 2444
DTL, 2262, 2312, 2361, 2387, 2414
DWORD, 2242, 2292, 2327, 2377, 2395, 2422, 2434

Explicit conversion, 2322, 2323, 2325, 2327, 2329, 2332, 2335, 2337, 2339, 2341, 2344, 2346, 2349, 2351, 2352, 2354, 2355, 2357, 2358, 2360, 2361, 2362, 2364, 2365, 2367, 2368, 2369, 2371, 2391, 2392, 2393, 2395, 2398, 2400, 2401, 2403, 2405, 2407, 2408, 2410, 2411, 2412, 2413, 2414, 2415, 2416, 2430, 2431, 2432, 2434, 2436, 2438, 2439, 2440, 2441, 2442, 2443, 2444, 2445, 2446
GRAPH instruction, 2795
Implicit conversion, 2289, 2290, 2293, 2294, 2295, 2296, 2297, 2298, 2300, 2301, 2302, 2304, 2305, 2306, 2307, 2308, 2310, 2311, 2312, 2313, 2314, 2316, 2317, 2318, 2319, 2320, 2375, 2376, 2378, 2379, 2380, 2381, 2382, 2383, 2384, 2385, 2387, 2389, 2390, 2420, 2421, 2422, 2423, 2424, 2425, 2426, 2428, 2429, 2430
INT, 2245, 2296, 2337, 2380, 2401, 2423, 2436
Internal tag, 8114, 8416
LDT, 2261, 2311, 2360
LINT, 2249, 2301, 2346
LREAL, 2253, 2305, 2352, 2385, 2410
LTIME, 2257, 2308, 2357
LTIME_OF_DAY, 2259
LTOD, 2316, 2365
LWORD, 2243, 2293, 2329
Mitsubishi, 8697
Mitsubishi FX, 8691
Mitsubishi MC TCP/IP, 8681
Modicon Modbus RTU, 8721
Modicon Modbus TCP, 8708
Omron, 8742
Omron Host Link, 8737
Pip, 892
PLC data type, 2280, 2854
POINTER, 2273
REAL, 2252, 2304, 2351, 2384, 2408, 2425, 2439
S5TIME, 2255, 2306, 2354, 2426, 2441
SCL instruction, 2700, 2701, 2702
SINT, 2244, 2294, 2332, 2378, 2398
STRING, 2264, 2319, 2369, 2390, 2416, 2430, 2446
STRUCT, 2272
TIME, 2257, 2307, 2355, 2386, 2411, 2425, 2440
TIME_OF_DAY, 2259
TOD, 2314, 2364, 2387, 2413, 2428, 2443
UDINT, 2248, 2300, 2344, 2383, 2407
UINT, 2246, 2297, 2339, 2381, 2403
ULINT, 2250, 2302, 2349
USINT, 2245, 2295, 2335, 2379, 2400
Valid, 8637, 8681, 8691, 8708, 8721, 8737
Validity, 2236
VARIANT, 2277
WCHAR, 2263, 2318, 2368
WORD, 2241, 2290, 2325, 2376, 2393, 2422, 2432
WSTRING, 2265, 2320, 2371
Data type conversion, 46
Data types, 245
Migration, 165
S7 1500, 8471
S7 200, 8618
S7 300/400, 8578
S7-1200 V1, 8522
Trends, 8468, 8627
Valid, 8471, 8522, 8578, 8618
Data word (DBW), 1572
DataLogClear, 4788
DataLogClose, 4791
DataLogCreate, 4777
DataLogDelete, 4793
DataLogNewFile, 4795
DataLogOpen, 4784
DataLogTypedNewFile, 4796
DataLogTypedOpen, 4785
DataLogWrite, 4789
Date, 2258, 2260, 2261, 2262
DATE, 2258, 2313, 2362, 2388, 2412, 2427, 2442
Date/time, 8380
Area pointer, 8502
Date/time field, 8078
Application, 8923
Behavior, 8923
Display system time, 8079
Format, 8079
Layout, 8923
Operation, 8923
Using tags, 8079
Date/time PLC, 8380
DATE_AND_LTIME, 2261
DATE_AND_TIME, 2260
DATE_TO_, 2362, 2412, 2442
Daylight Saving Time/Standard Time
Adjusting time-of-day status with SET_SW_S, 6431
Switching with SET_SW, 6429
DB, 2503, 2829
DB 101 to DB 104
In/out parameters, 7226
Input parameters, 7220
Output parameters, 7223
DB 101 to DB 116
In/out parameters, 7321
Input parameters, 7316

Output parameters, 7319
DB variable
Definition, 2203
DB_ANY_TO_VARIANT, 3834, 4252
DCAT, 3426, 3728, 3873, 4309, 4466, 5556, 5766, 5871, 6243, 6371
DCD, 982
DCP (Primary Setup Tool), 754
DCP server, 1220
Deactivate, 8356
Dead peer detection (DPD), 794
DEAD_T
Input parameters, 6953
Instruction, 6951
Output parameters, 6953
Shared data block, 6953
Deadband
Setting, 8165
DEADBAND
Input parameters, 6955
Instruction, 6953
Output parameters, 6955
DEC, 3276, 3573, 3972, 5975
Decimal places, 34, 3298, 3597, 4180, 4370, 8919, 8920
Declaring ready for communication
- passive connection establishment, 596
DECO, 3405, 3706, 3860, 4289, 4442, 5511, 5723, 5843, 6212, 6340
Decode, 3405, 3706, 3860, 4289, 4442, 5511, 5723, 5843, 6212, 6340
DecreaseFocusedValue, 8342
DecreaseTag, 8343
Decrement, 3276, 3573, 3972, 5975
Default domain, 1476
Default font, 8849, 8912
Default for the dynamics, 10303
Default setting
Diagnostic interrupt, 9911
Default settings, 1083
Default style
Defining, 7978
Defaults for the dynamics, 10287
Defective devices, 2080
Defective license, 89, 102
Define
Reference object, 7989
Define hotkey, 8101
Defining
For the assignment list, 2942
Definition
Diagnostic interrupt, 9911
Delay time
Isochronous mode, 1405
Setting, 8165
Delete
Template, 7974
WinAC RTX, 38
DELETE, 4545
Delete residual distance, 10487, 10579
DELETE_DB, 4806
Deleting
Authorization, 8254
Filter in the assignment list, 2942
Hardware component, 510
Object, 7984
Recipe data record, 8957
Screen, 7968
Tag, 8120
User group, 8254
Users, 8253, 8941
Deleting a connection, 180
Deleting CA certificates, 10958
Deleting receive buffer with RES_RCVB, 7899
Deleting WinAC RTX, 38
Demultiplex, 3413, 3713, 4295, 4449
DEMUX, 3413, 3713, 4295, 4449
Dependencies of rights, 743
Dependency structure, 2952
Introduction, 2952
Meaning of symbols, 2954
Setting view options, 2955
Structure, 2953
Dependency structure
Displaying, 2954
DES, 782, 788
Description
ABS_CTRL, 7125
ABS_CTRL_451, 7129
ABS_DIAG, 7127
ABS_DIAG_451, 7131
ABS_INIT, 7124
CAM_CTRL, 7157
CAM_CTRL_452, 7160
CAM_DIAG, 7158
CAM_DIAG_452, 7162
CAM_INIT, 7156
CAM_MSRM_452, 7163
CNT_CTL1, 7098
CNT_CTL2, 7100
CNT_CTRL, 7096
CNT2_CTR, 7112
CNT2RDPN, 7111
CNT2WRPN, 7110

DIAG_INF, 7097
DIAG_RD, 7112
EncoderAbsSensorDP, 7065
EncoderCPU314C, 7073
EncoderET200S1Count, 7069
EncoderET200S1SSI, 7064
EncoderFM350, 7070
EncoderFM450, 7061
EncoderIM174, 7074
EncoderIM178, 7059
EncoderSINAMICS, 7077
EncoderSM338, 7067
EncoderUniversal, 7080
MC_Control, 7049
MC_GearIn, 7051
MC_Home, 7043
MC_Init, 7034
MC_MoveAbsolute, 7035
MC_MoveJog, 7040
MC_MoveRelative, 7038
MC_Simulation, 7050
MC_StopMotion, 7047
OutputCPU314C, 7084
OutputET200S2AO, 7083
OutputIM174, 7087
OutputIM178, 7082
OutputMM4_DP, 7086
OutputSINAMICS, 7089
OutputSM332, 7085
OutputSM432, 7083
OutputUniversal, 7085
Deserialization, 3785, 4181, 4371
Deserialize, 3307, 3605, 3785, 4181, 4371
Designation
TS Adapter, 10929, 10937, 10945
Designing a background, 7995
Designing a border, 7995
Destruction of license keys, 88
DETACH, 4693
Detailed comparison
Performing, 374
Starting, 2875
Visualization of the comparison result for GRAPH, 2886
Visualization of the comparison result for LAD/FBD, 2876
Visualization of the comparison result for SCL, 2883
Visualization of the comparison result for STL, 2880
Details of connection establishment, 631, 637, 643
Details view, 299
Determine diagnostics status, 2100
Determining a coordinate
Zero offset, 9364, 9440
Determining time of day, 2108
Device
Adding to a hardware configuration , 504
Copying, 57, 511
Defective devices, 2080
Deleting, 510
Inserting, 8375
Moving, 512
Renaming, 721
Device address, 881, 937
Device dependency
S7-1200 V1, 8520
S7-1200 V2, 8520
S7-1200 V3, 8520
S7-1200 V4, 8520
SIMATIC S7-1500 V1.0, 8470
Device information, 384, 2080
Diagnostics, 8387
Device name, 1441, 1447, 1549
Device name, automatic commissioning (PROFINET), 1443
Device names
PROFINET, 1465
Device number, 1442
PROFINET, 1466
Device overview
Address range, 880, 881, 937
Device proxy
Initializing, 8790
Updating, 8788, 8793
Device replacement, 513
Device status
Querying a remote partner with STATUS, 7405
Receiving change with USTATUS, 7408
Device Tool, 1551, 1552
Device version, 8884
Checking, 8876
Switching, 8875
Updating, 8896
Device view
Area of unplugged modules, 501
Edit parameters, 514
Edit properties, 514
Hardware and network editor, 480, 8389
Inserting a signal board, 941
Inserting module, 507
Racks, 498, 876
Device wizard, 8760

Devices
Connecting, 8379
Networking, 8378, 8402
Devices & Networks, 8378, 8402
Devices and networks, 8378, 8402
Connections, 8403
HMI connections, 8403
DeviceStates, 4753
DHCP
Client, 1233
Server configuration, 830
DHCP server, 831
DIAG_INF, 7097
Call, 7098
Description, 7097
Operating principle, 7097
Parameter, 7098
DIAG_RD, 7112
Call, 7113
Description, 7112
Operating principle, 7112
Parameters, 7113
Diagnose repeater, 2128
Diagnostic address, 881
Configuring, 1369
I-slave, 1368
Diagnostic buffer, 903
Diagnostic data
Reading out with GET_DIAG, 4761
Diagnostic data record, 9863, 9911, 9912, 9944
Diagnostic data record DS0
Assignment, 9863, 9912, 9944
Diagnostic data record DS1 of the FM 355-2
Assignment, 9913
Diagnostic DB
Configuration, 9423, 9501
Data, 9423, 9501, 10536, 10635
Structure, 10536, 10635
Diagnostic error interrupt, 1625
Diagnostic error interrupt (S7-1500), 1908
Diagnostic events, 9423, 9500, 10535, 10634
Diagnostic functions, 1014, 1154
Diagnostic information
Analyzing with DPV1, 1389
Diagnostic interrupt, 9017, 9158, 9184, 9215
Default setting, 9862, 9911, 9943
Definition, 9862, 9911
enable, 9862, 9911, 9943
Enable, 9047
In event of error , 9943
OB 82, 9862, 9911, 9943
trigger, 9862, 9910, 9943
triggered, 9911
What is a diagnostic interrupt?, 9911
Diagnostic Interrupt
What is a diagnostic interrupt?, 9943
Diagnostic interrupt OB, 1625
Diagnostic interrupt OB (OB 82) (S7-300, S7-400), 1783
Diagnostic interrupt OB (S7-1500), 1908
Diagnostic interrupts, 9420, 9498, 10532, 10631
Evaluation, 10535, 10634
Diagnostic record DS1
Assignment, 9864, 9913, 9945
Diagnostics, 797, 1014, 1154, 2037
Writing a diagnostic event to the diagnostics buffer with WR_USMSG, 6665
Channel-specific, 2157
Connection, 8387
Connection information, 8387
Connections with C_DIAG, 6743
Device, 8387
Device information, 8387
Diagnostic data, 6631
Diagnostic data record, 6737
Diagnostic data, channel-specific, 6633
DP slave, reading diagnostic data with DPNRM_DG, 4614
DP slave, reading diagnostics data with DPNRM_DG, 6523
Easy Motion Control, 10773, 10820
Errors in Motion Control instructions, 10353
External encoder, 10362
High_Speed_Counter, 9043
Module diagnostic information, 6738
Module, system status list, 6736
Motion Control, 10349
Motion Control diagnostics concept, 10349
Positioning axis, 10357
PROFINET, 1472
PROFINET IO components, 2158
Speed axis, 10353
Technology alarms, 10350
Technology module, 9071
Diagnostics address
Assigning, 1386
Diagnostics buffer
Basics, 1576, 1666, 2123
Organization, 1666, 2123
System status list, 6735
Diagnostics data, 1668
ABS_DIAG, 7128
ABS_DIAG_451, 7132

Diagnostics DB
Configuration, 7174
Content, 7174
Purpose, 7174
Diagnostics DB FM x51
Configuration, 7141
Content, 7141
Purpose, 7141
Diagnostics event
for modules, 1827
Diagnostics status
Determining and displaying online, 2081
Diagnostics text, 9863, 9912, 9944
Diagnostics user, 739
Diagonal movement
Animation, 8049
Dialing rules in phone books, 10926
Dialog ", 1998
dial-up connection
Establishing, 10950
Terminating, 10952
DIF
Input parameters, 6957
Output parameters, 6957
Differentiator, 6955
Diffie-Hellman key agreement, 787
DIGITAL, 7332
Digital input
Functions, 9035, 9058, 9064, 9065
Digital inputs
Input filter, 9125
Input filters, 9201
Digital output
Comparison value, 8998, 9037, 9060, 9066
Functions, 9037, 9060, 9066
Digital outputs
Behavior in counting modes, 9130, 9207
Behavior in measuring modes, 9141
Enable, 9131, 9141
Enabling, 9208
Setting and resetting, 9131
DINT, 2247, 2298, 2382, 2424
DINT , 2341, 2405, 2438
DINT_TO_, 2341, 2405, 2438
Direct access to the I/O, 1577, 1743, 1845
Direct data exchange
Configuring, 1377
DP slave - DP master, 1344
DP slave - I-slave, 1344, 1346
Principle of operation, 1375
PROFIBUS, 1376
Setting up, 1377
Direct homing, 10237
Direct key
configuring, 8066
Direction of rotation signal, 10751
Direction output and travel direction
relation, 10011
Direction reversal, 10492, 10589
Direction reversal at the hardware limit switch, 10237
DIS_AIRT, 4721
DIS_IRT, 4718
Disabling
Project language, 8830
Discrete alarm, 8150, 8151
Allen-Bradley, 8670
Configuring, 8161, 8169
Mitsubishi, 8697
Omron, 8742
Discrete alarm types, 8150
Discrete alarms
Configuring, 8159
display
Tooltip, 8921
Display, 1868
Communication instructions for non-configured S7 connections, 7352
HMI backup, 8878
Display (CPU), 1889
Display class, 1688
Display classes for messages, 912
Display object
Availability for Basic Panels, 8075
Display PDF files, 85
Display system time, 8079
Display Welcome Tour, 85
Displaying
Call structure, 2949
Assignment list , 2940
CPU memory area, 2960
Cross-reference, 8823
Cross-references, 2964, 8820
Dependency structure , 2954
Load memory, 2961
Maximum load memory available, 2961
Program information, 2936
Displaying a force table, 3026
Displaying alarms, 2074
Displaying memory types of a CPU, 2098, 2099
Displaying or hiding tag information, 2603, 2645, 2668, 2702, 2810
Displaying support for a module, 491
Displaying the online status, 10855

Displaying values
As a trend, 8146
Distance measurement
Easy Motion Control, 10816
Distance-to-go, 10487, 10579
Distribute
Objects evenly, 7991
Distributed I/O, 1340, 1413, 2035
Configuring, 1339
Slot numbers, 1340
DIV, 3271, 3568, 5444, 5656
Divide, 3271, 3568, 3963, 3968, 3977, 5444, 5656, 5967, 5971, 5981
DLE, 996
DM 370 Dummy, 865
DMSK_FLT, 4716
DNS client, 1224
DNS configuration, 1933
DO, 4262, 4265, 6197
Docking unit, 1505, 1507
Docking Unit, 1507
Documentation editor
Layout, 389
Documentation settings
Using frames and cover pages, 387
Dosing, 9098
Double evaluation, 9108
Double word, 2242
double-click, 8357
Download, (See Loading to device)
Error messages, 8899
Project, 8885, 8903
Download firmware via Web, 1948
Downloading
Downloading blocks to a memory card, 2912, 2927
Downloading blocks to device, 2908, 2909, 2923
Uploading blocks from a memory card, 2928
Downloading data to the PLC
Error message, 73
DP, 8499, 8552
DP cycle
Time for constant bus cycle time DP cycle, 1396
DP interface, 1347, 1418, 2151
Addresses, 1391
Changeover, 1350
DP master, 863, 2022
Add DP master system, 1350, 1420
Devices and modules, 1347, 1418
Disconnection from DP master system, 1350, 1420
Display on DP slave, 1348, 1419
DP interface, 1347, 1418
DPV1 functionality, 1385
DP master system
Add DP slave, 1352, 1422
Creating, 1340, 1346, 1353, 1413, 1416, 1417
Determining topology with DP_TOPOL, 4617, 6526
Disconnect DP slave, 1352, 1422
Disconnection from subnet, 1349, 1420
Edit properties, 1351, 1421
Highlight, 1348, 1419
Information, system status list, 6717
More than one, 1343
Node disconnection, 1349, 1420
Setting up, 1379
DP slave, 863
Activating and deactivating with D_ACT_DP, 4589, 6486
Add to DP master system, 1352, 1422
Addresses, 1393
As I-slave, 1374
Assign DP master system, 1352, 1422
configuring, 1423
Configuring, 1340, 1343, 1353, 1414
Data exchange, 1343, 1414
Disconnect from DP master system, 1352, 1422
DP master display, 1348, 1419
Hardware catalog, 1341, 1413
Intelligent, (See I-slave), (See I-slave)
Networking, 1353, 1416
Reading diagnostic data with DPNRM_DG, 4614
Reading diagnostics data with DPNRM_DG, 6523
Setting the isochronous mode, 1408
Slot rules, 1341, 1351
SYNC/FREEZE groups, 1355
Synchronizing groups with DP_SYC_FR, 4608, 6517
Types, 1351, 1421
With preprocessing, (See I-slave), (See I-slave)
DP standard slave, 1383, 1435
DP standard slaves
Reading a portion of the inputs with GETIO_PART, 4562, 6465
Reading all inputs with GETIO, 4560, 6463
Reading consistent data with DPRD_DAT, 4600, 6498
Writing a part of the outputs with SETIO_PART, 4563, 6467
Writing all outputs with SETIO, 4561, 6463
Writing consistent data with DPWR_DAT, 4601, 6500

DP/DP coupler, 1342, 1414
DP_CTRL, 7884
DP_DIAG, 7877
DP_RECV, 7871
DP_SEND, 7868
DP_TOPOL, 4617
DPNRM_DG, 4614
DPRD_DAT, 885, 4600
DPSYC_FR, 885, 4608
DPV0/DPV1, 1384
DPV1, 1683
  Analyzing diagnostic information, 1389
  Analyzing interrupt information, 1389
  Configuring an ET 200S, 1365, 1433
  OBs for interrupts, 1386
  Slot model, 1391
DPWR_DAT, 885, 4601
Drive
  Changeover difference, 10503, 10600
  Control mode 1, 10500, 10597
  Control mode 2, 10500, 10598
  Control mode 3, 10501, 10598
  Control mode 4, 10502, 10599
  Machine data, 10499, 10596
  Monitoring time, 10507, 10604
  Standstill range, 10507, 10604
  Standstill velocity, 10508, 10605
  Switch-off difference, 10504, 10601
  Target range, 10505, 10602
Drive and encoder connection, 10216
Drive connection for Motion Control, 10216
Drive type, 10750, 10751, 10757
DRUM, 3423, 3725, 3865, 4306, 4459, 5549, 5759, 5859, 6236, 6363
DRUM_X, 5552, 5762, 5867, 6239, 6366
DSC (Dynamic Servo Control), 10240
DSCP, 1264
DSR, 982
DST
  Daylight saving time, 1238
DT, 2260, 2310, 2358, 2428, 2444
DT_TO_, 2358, 2444
DTB, 4014, 6017
DTL, 2262, 2312, 2361, 2387, 2414
DTL_TO_, 2361, 2414
DTR, 982, 4015, 6018
Dual code, 9017
Dummy module, 865
Duration, 2255
DWORD, 2242, 2292, 2327, 2377, 2395, 2422, 2434
DWORD_TO_, 2327, 2395, 2434
DxB, (See Direct Data Exchange)
Dynamic adjustment, 9346, 9412, 9487
Dynamic reference temperature, 2057
Dynamic Servo Control (DSC), 10240
Dynamically allocated system resources
  Releasing with DEL_SI, 6677
Dynamization
  Color of an object, 8047
  Flashing, 8047
  Object, 8042
Dynamize
  Appearance of an object, 8047
  Direct movement, 8050
  Green arrow in Overview, 8046
  Movement of an object, 8048

E

Eastern characters
  Input on the HMI device, 8914
Easy Motion Control
  Absolute encoder adjustment, 10772
  Absolute positioning, 10772
  Actual values, 10810
  Axis DB, 10808
  Axis does not move, 10817
  Axis moves without travel command, 10818
  Axis oscillates, 10818
  Axis parameters, 10810
  Bit array for initialization, 10812
  Call environment, 10800
  Call sequence of the instructions, 10801
  Common parameters for all instructions, 10804
  Diagnostics, 10773, 10820
  Distance measurement, 10816
  EMERGENCY STOP switch, 10813
  Error with hard stop, 10825
  Error with soft stop, 10822
  Fault reactions, 10773
  Following error, 10768, 10819
  Gear, 10773
  Hardware configurations, 10765
  Interaction of the instructions, 10799
  Jogging, 10771
  Linear axis, 10769
  Monitoring ranges, 10769
  Multi-instance FB, 10812
  Overriding travel, 10773, 10806
  Parameter error, 10821
  Parameter errors, 10827
  Positioning, 10767
  Reference point approach, 10772
  Relative positioning, 10772

Residual distance, 10768
Rotary axis, 10770
Set reference point, 10772
Simulation, 10773
Standstill range, 10769
Start of a travel order, 10805
Stopping, 10773
Synchronization, 10771
Target, 10768
Target range, 10769
Tracking, 10773
Traversing range, 10769
User program configuration, 10799
Velocity measurement, 10814
Velocity of axis, 10818
Velocity override, 10773
Wiring test, 10814
Working range, 10769
EB, 2477
EC31-HMI/RTX
Visualization, 1162
EC31-RTX, 41
ED, 2477
Edge
Negative, 3168, 3171, 3173, 3463, 3465, 3468, 3924, 4114, 5384, 5387, 5595, 5598, 5929
Positive, 3167, 3169, 3172, 3461, 3464, 3467, 3926, 4113, 5383, 5386, 5594, 5597, 5930
Edge detection, 10585
Data in the channel DB , 10587
Data in the parameter DB, 10587
faulty, 10587
Requirement, 10586
Sequence, 10586
Edit
Object within a group, 8018
Edit networking
Disconnecting from a network, 526
EditAlarm, 8293
Editing
Alarm, 8948
Folder link, 8010
Function list, 8280
Recipe data record, 8956
Editing a version of a type in testing, 453
Editing language, 340, 8826
Selecting, 8830
Editing networking
Copying a subnet, 526
Copying subnets and devices, 527
Editor
Connection, 8407
Connections, 8579, 8580, 8582
Devices and networks, 8405
Graphics, 8840
Effect of Ti
Isochronous mode, 1483
Effect of To
Isochronous mode, 1484
Effects
Reference point approach, 10480, 10571
Set actual value, 10489, 10581
Set reference point, 10490, 10585
Zero offset , 10583
Element list, 8219
Ellipse
Horizontal radius, 8082
Vertical radius, 8082
Ellipse , 8081
ELSE, 4258, 4260, 6190, 6192
ELSIF, 4258, 6190
E-mail
Sending, 667
Transferring e-mail with TM_MAIL, 5360
Transferring e-mail with TMAIL_C, 5073
E-mail connection, 1933
Configuring, 665
Parameter field for, 7826
E-mail DB, 668
E-mail server, 667
Embedded controller, 1163
Load:Download, 1163
Emergency stop deceleration, 10250, 10288, 10304
Emergency stop ramp, 10250
EMERGENCY STOP switch
Easy Motion Control, 10813
Empty box
Inserting a LAD element, 2582
Inserting an FBD element, 2624
EN/ENO mechanism
Basics, 2447
FBD example, 2450
LAD example, 2450
SCL example, 2451
STL example, 2452
EN_AIRT, 4722
EN_IRT, 4720
Enable
Alarms with EN_MSG, 6671
Diagnostic interrupt, 9017, 9911
Hardware interrupt, 9017
Enable autonegation, 1459, 8440, 8489, 8542
Enable input, 10477, 10494, 10568, 10591
Data in the channel DB, 10494, 10591

Data in the parameter DB, 10591
Do not evaluate, 10494, 10591
Evaluate, 10494, 10591
Enable output ENO, 250
Enabling
Machine data, 9354, 10463, 10554
Enabling the firewall
SCALANCE S, 821, 823
Enabling tunneled communication
CP x43-1 Adv., 844
SCALANCE S, 821, 823
ENCO, 3407, 3707, 3862, 4290, 4444, 5513, 5725, 5844, 6214, 6341
Encode, 3407, 3707, 3862, 4290, 4444, 5513, 5725, 5844, 6214, 6341
Encoder
Count direction, 9401, 9476, 10517, 10614
Distance per encoder revolution, 9397, 9472, 10518, 10615
Frame length, 9403, 9479, 10522, 10619
Increments per encoder revolution, 9397, 9473, 10518, 10616
Initiator direction, 9404, 9479
Monitoring, 9401, 9477, 10523, 10620
Number of encoder revolutions, 9402, 9478, 10521, 10618
Parity, 10522, 10620
Selecting the signal type, 9121, 9172, 9197
Signal type, 9396, 9471, 10516, 10613
Transmission rate, 9403, 9478, 10521, 10619
Encoder configuration:, 10222
Encoder connection for Motion Control, 10216
Encoder control word, 10761
Encoder data, 10495, 10592
Data in the channel DB, 10495, 10592
Process when reading, 10495, 10592
Requirement, 10495, 10592
Encoder machine data
Data in the parameter DB, 9398, 9474
Encoder type, 10750, 10754, 10755
SSI, 10755
TTL, 10755
Encoder type not available, 10750, 10754, 10755
Encoder value acquisition
freewheeling, 10431, 10443
Freewheeling, 10416
isochrone, 10417, 10443
Synchronous, 10431, 10443
EncoderAbsSensorDP, 7065
Call, 7066
Description, 7065
Error handling, 7067
Parameter, 7067
Synchronization, 7067
EncoderCPU314C, 7073
Call, 7073
Description, 7073
Error handling, 7074
Parameter, 7073
Synchronization, 7073
EncoderET200S1Count, 7069
Call, 7069
Description, 7069
Error handling, 7070
Parameter, 7069
Synchronization, 7070
EncoderET200S1SSI, 7064
Call, 7064
Description, 7064
Error handling, 7065
Parameter, 7065
Synchronization, 7065
EncoderFM350, 7070
Call, 7071
Description, 7070
Error handling, 7072
Parameter, 7071
Synchronization, 7072
EncoderFM450, 7061
Call, 7062
Description, 7061
Error handling, 7063
Parameter, 7062
Synchronization, 7063
EncoderIM174, 7074
Call, 7076
Configuration of IM 174 (hardware configuration), 7074
Description, 7074
Error acknowledgment, 7077
Error handling, 7077
Parameter, 7076
Synchronization, 7076
EncoderIM178, 7059
Call, 7060
Configuration of IM 178-4 (hardware configuration), 7060
Description, 7059
Error acknowledgment, 7061
Error handling, 7061
Parameter, 7060
Synchronization, 7060
Encoders
Connectable, 9105

Incremental, 9105
Pulse, 9105
EncoderSINAMICS, 7077
Call, 7078
Configuration of SINAMICS (hardware configuration), 7077
Description, 7077
Error acknowledgment, 7080
Error handling, 7080
Parameter, 7078
Synchronization, 7079
EncoderSM338, 7067
Call, 7067
Description, 7067
Error handling, 7068
Parameter, 7068
Synchronization, 7068
EncoderUniversal, 7080
Call, 7080
Description, 7080
Error handling, 7081
Operating principle, 7080
Parameter, 7080
Synchronization, 7081
Encryption, 727
Encryption software, 93, 97
End, 10471, 10562
Positioning operation, 10467, 10557
End criterion, 1043
After character delay time elapses, 1043
End delimiter, 1044
Fixed message frame length, 1046
End delimiter, 1077
End detection indicator of a receive message frame, 1077
End of a positioning operation
Checkback signals in the channel DB, 10473, 10564
Data in the parameter DB, 10473, 10564
End of detection of accessible devices, 1461
End of discovery of accessible devices, 8442, 8491, 8544
End of rotary axis, 10511, 10608
End of sync domain, 1461, 8442, 8491, 8544
End of topology discovery, 1461, 8442, 8491, 8544
End sequence, 992
END_CASE, 4260, 6192
END_FOR, 4262
END_IF, 4258, 6190
END_REPEAT, 4266, 6198
END_WHILE, 4265, 6197
Endian, 4218, 4220, 4222, 4224
ENDIS_PW, 3382, 3682, 3842, 4272, 4427
Engineering system
Performance features, 8959
Enhanced Write Filter (EWF), 1163
ENO, 250
ENSET_DN, 9126, 9202
ENSET_UP, 9126, 9202
ENT, 6105
Entering
Message frame type, 10758
Enumeration types, 923, 958
Enumerations
Web server, 924, 960
EQ_ElemType, 3258, 3554
EQ_Type, 3256, 3552
Equal, 3261, 3556
EQUAL ARRAY, 3263, 3559, 4161
ERPC tag, 1941
ERR_MON
Input parameters, 6959
Instruction, 6957
Output parameters, 6959
Error
external, 9861, 9910, 9942
Internal, 9861, 9910, 9942
Error alarm
Acknowledge, 8948
Error analysis with RET_VAL, 3152, 5370
Error classes
Alarms, 9431, 9508, 10544, 10643
Cam data error, 9430, 9507
Data error, 9426, 9504, 10540, 10639
Diagnostic errors, 9431, 9509, 10545, 10644
Incremental dimension table error, 10544, 10643
Machine data error, 9428, 9505, 10542, 10640
Operating error, 9425, 9503, 10538, 10637
Operator errors, 10539, 10638
Error codes
Gx_XIST2, 10762
Error display, 9861, 9910, 9942
Error evaluation
User program, 9422, 9500, 10534, 10633
Error handling, 2825
Basics, 2821
GetError, 3388, 3688, 3849, 4278, 4433
GetErrorID, 3392, 3692, 3852, 4282, 4437
Local error handling, 2823
Error ID, 10405
Error information, 2823, 3152, 3388, 3392, 3688, 3692, 3849, 3852, 4278, 4282, 4433, 4437, 5370
Error LEDs, 9942

Error message
Downloading data to the PLC, 73
Error messages
Download, 8899
Error response
ABS_CTRL, 7127, 7131
ABS_DIAG, 7128
ABS_DIAG_451, 7132
CAM_CTRL, 7158
CAM_CTRL_452, 7161
CAM_DIAG, 7159
CAM_DIAG_452, 7163
CAM_MSRM_452, 7164
CNT_CTL1, 7100
CNT_CTRL, 7097
High_Speed_Counter, 4923
Error status, 1199
Error with hard stop
Easy Motion Control, 10825
Error with soft stop
Easy Motion Control, 10822
ErrorID, 4926
List of ErrorIDs, 10405
Errors in Motion Control instructions, 10353
ESP protocol, 788
Establishing a connection from a remote system ("PG_DIAL"), 10966
Establishing a remote connection, 10950
Procedure, 10951
ET 200eco, 2020
ET 200eco PN, 2020, 2035
ET 200iSP, 2019
ET 200iSP distributed I/O station
Definition, 2021
ET 200L, 2019
ET 200M, 1409, 2019, 2061
Definition, 2060
Slot rules, 1341
ET 200MP, 2052
ET 200pro, 2020
ET 200R, 2020
ET 200S, 1409, 2019
DPV1 mode, 1365, 1433
Option handling, 1361, 1364, 1429, 1432
Positioning module, 41
Reference junctions, 1357, 1425
Slot rules, 1356, 1425
ET 200S 1SI, 1027
ET 200S COMPACT, 2019
ET 200SP, 2041
Application area, 2041
Operation, 39
ET 200M distributed IO device, 2060
Ethernet, 534, 536
Parameters, 8474, 8528, 8586, 8595, 8623
ETHERNET
Basic Panel, 8427
Ethernet interface
Displaying parameters, 10866
Ethernet interfaces
EC31-HMI/RTX, 1162
EC31-RTX, 1162
EC31-RTX F, 1162
Ethernet module
Removal/insertion, 38
Ethernet non IP frames, 746
Ethernet Statistics
Frame length, 1208
Frame type, 1209
Interface statistics, 1207
Packet Errors, 1209, 1210
ETX, 996
Evaluating interrupt information with DPV1, 1389
Evaluation
Diagnostic interrupts, 10535, 10634
Event, 1816
Acknowledge, 8361
activate, 8353
Alarm buffer overflow, 8360
Assigning organization block (OB) with ATTACH, 4691
Boundary reached, 8361
Canceling assignment to organization block (OB) with DETACH, 4693
Change, 8354
cleared, 8353
Click, 8359
Click when flashing, 8359
Deactivate, 8356
double-click, 8357
Execute, 8354
For function lists, 8134
Free space critically low, 8363
Incoming, 8359
Input finished, 8358
Inspector window, 63
Loop-In-Alarm, 8360
Low free storage space, 8363
Outgoing, 8358
Overflow, 8362
Press, 8357
Press ESC twice, 8358
Press key, 8362
release, 8360

Release key, 8362
Runtime Stop, 8361
Screen change, 8356
Selection changed, 8355
Switch OFF, 8363
Switch ON, 8363
Tags, 8134
Time expired, 8364
User change, 8356
Value change, 8363
When dialog is closed, 8356
When dialog is opened, 8355
When high limit is exceeded, 8355
When low limit is undershot, 8355
Event class, 1816
Event class 1 - standard OB events, 1817
Event class 2 - synchronous errors, 1817
Event class 3 - synchronous errors, 1818
Event class 4 - stop events and other mode changes, 1821
Event class 5 - status runtime events, 1825
Event class 6 - communication events, 1826
Event class 8 - diagnostics events for modules, 1827
Event class 9 - standard user events, 1829
Event classes A and B - free user events, 1830
Event ID, 1816
Event name, 1631
Event trigger, 8369, 8372
Event-driven tasks
Configuring:event-driven tasks, 8163
Events
Configuration, 1229
Log table, 1198
Severity filter, 1231
Events and event ID, 1816
EW, 2477
EX, 2477
Example
Application for alarm classes, 8153
Calculating an equation, 3104, 3121, 3131, 3139
Changing and displaying operating mode, 8281
Controlling a conveyor belt, 3082, 3099, 3110, 3115, 3134, 3145
Controlling room temperature, 3088, 3097, 3122, 3132
Detecting the direction of a conveyor belt, 3084, 3092, 3100, 3111, 3117, 3126, 3135, 3146
Detecting the fill level of a storage area, 3085, 3093, 3102, 3112, 3118, 3127, 3136, 3147
For displaying bit memory in the assignment list, 2938
For displaying inputs and outputs in the assignment list, 2938
For entering force values in the force table, 3029
Heating an oven, 3105, 3140
Modify values in the watch table, 3002
Specification of a substitute value, 1815
System alarm, 8150
Example:
Discrete alarm, 8151
Examples
Controlling a conveyor belt, 3090, 3124
Exchangeable medium, 1461
Exchanging, 8784
EXCLUSIVE OR, 3449, 3450, 5585, 5586
Bit logic operations, 3449, 5585
Execute, 8354
Execution cycle, 1397
Exit
User program, 3387, 3687, 3848, 4278, 4433, 5499, 5711, 5836, 6207, 6332
EXIT, 4268, 6200
EXP, 3289, 3587, 3981, 4172, 4362, 5458, 5671, 5985, 6163, 6297
Expandable racks, 877
Expanded mode, 2995
Expanding and collapsing sections of code, 2694
Expansion rack
ER1 (S7-400), 878
IM 46x, 878
Rules, 879
Exponential value, 3289, 3587, 3981, 4362, 5458, 5671, 5985, 6297
Exponentiate, 3300, 3598
Export
Alarm, 8800
Excel format, 8796
Project texts, 8817, 8837
Recipe, 8796
Tag, 8807
Text list, 8812
Export of labels, 399
Exporting an NTP server, 778
Expression
Basics, 2681
Logical expression, 2686
Relational expressions, 2684
Expression
Arithmetic expression, 2682
EXPT, 3300, 3598
Extended download to device, 74
Extended status information, 2689
External, 46

External encoder
Configuration, 10305
Tags, 10395
External encoder technology object, 10214
External errors, 9861, 9910, 9942
External graphic
Edit folder, 8009
Link folder, 8010
Removing the folder link, 8009
Rename folder, 8009
External image file
add to graphics library, 8842
Managing, 7982
Storing in the image browser, 8011
External source file
Basics, 2863
Exporting blocks, 2865
Import organization block, 48
inserting, 2866
Linking file types to an editor, 2867
Rules for programming, 2865
External sources, 46
External tags
Data exchange, 8379

F

F_TRIG, 3176, 3470, 3748, 4114
Facility, 767
Factory settings
Resetting to, 2111, 2112
Resetting to , 8897
Fail-safe modules
Selecting from the hardware catalog, 874
FALSE, 4113, 4114
FAQs, 8397
Fast cam parameter change
Canceling, 9371, 9447
Data check, 9371, 9446
Data used in the channel DB, 9370, 9445
Definition, 9370, 9445
Effects, 9371, 9446
Requirements, 9370, 9445
Steps, 9370, 9445
Fast startup (PROFINET IO), (See Prioritized startup)
Fault monitoring
Connection status change, 1250
Power supply, 1249
Redundancy, 1250
Fault reactions
Easy Motion Control, 10773
Favorites
Adding, 2585, 2627, 2669, 2703, 2772
Hiding, 2532
Removing, 2587, 2628, 2670, 2705, 2773
Showing, 2532
Using, 2586, 2627, 2670, 2704, 2773
FB, 2171
FBD, 2611, 3090, 3092, 3093, 3124, 3126, 3127
Detailed comparison, 2876
Migration, 135
FBD element
Copying, 2636
Cutting, 2637
Deleting, 2642
Inserting, 2623, 2624
Inserting an operand , 2643
pasting from the clipboard, 2637
Replacing, 2638
Rules for inserting, 2621
Selecting, 2635
FBD network
Branch, 2646
Deleting a branch, 2647
Inserting a block call, 2587, 2629
Inserting a branch, 2647
Program status display, 2988
Rules for branches, 2646
FC, 2170
FDL connection
Free layer 2 access, 662
Unspecified , 662
With broadcast, 662
With multicast, 664
Feedback interface, 9077
1Count24V for counting modes, 9243
1Count24V for fast mode, 9279
1Count24V for measuring modes, 9261
1Count24V for position feedback, 9272
1Count5V for counting modes, 9304
1Count5V for fast mode, 9337
1Count5V for measuring modes, 9319
1Count5V for position feedback, 9330
1SSI in fast mode, 10448
1STEP 5V, 10744
2PULSE, 10002
FM 350-1 for counter modes, 9149
FM 350-1 for measuring modes, 9154
FM 351, 10527
FM 352, 9415
FM 451, 10625
FM 452, 9492

Feedback signals
Reading, 7125, 7129
FETCH frame, 1061
FETCH/WRITE, 1613, 1888
PDU structure, 1961
FETCH/WRITE configuration, 1960
FETCH/WRITE services
Configuration for ISO transport, 655, 1959
FETCH_RK, 1030, 7904
Fetching data
RK512, 1067
Fictitious target, 10492, 10589
Location, 10492, 10589
Field device, 1383, 1435, 1464
Field of application, 10748
FieldRead, 3314, 3612
Fields of application, 10748
FieldWrite, 3316, 3614
File allocation table, 1942, 1951
File browser, 903
File Browser, 947
Fill
Block, 3326, 3352, 3625, 3652, 3802, 3824, 4194, 4235, 4384, 4406, 5476, 5689, 5823, 6177, 6315
Block uninterruptible, 3328, 3627, 3804, 4196, 4386
FILL, 50, 3352, 3652, 3824, 4235, 4406, 5476, 5689, 5823, 6177, 6315
FILL_BLK, 3326, 3625, 3802, 4194, 4384
Filter
Defining for the assignment list, 2942
Delete, 2942
Hardware catalog, 497
In the assignment list, 2942
Selecting, 2943
Filter frequency, 9031, 9056
Filters
Assignment list , 2943
FIND, 4547
Find and replace, 406
Additional options for searching, 407
Replacing search keys, 408
Start search, 408
Using the search function, 407
Find reference mark, 10761
Firewall, 906
Creating service groups, 752, 754
Defining ICMP services, 751
Firewall rules, 746
Managing service groups, 752, 755
Firmware, 1637
Updating, 8896
Firmware card, 1570
Firmware download for S7-300/400 PROFIBUS CPs via CP 5612 / CP 5622, 1992
Firmware update, 1860, 2109
Firmware update memory card, 1570, 1838
Firmware version V4, 31
First scan, 2453
Fixed frame length, 992
Fixed reference temperature, 2057
Flash test, 2127
Flashing, 8047
Flexible memory capacity, 1659
Flip
Object, 7982
Flip-flop
Reset/set, 3165, 3460, 5382, 5593
Set/reset, 3164, 3459, 5380, 5591
Floating-point number, 2252, 2253
Check invalidity, 3255, 3551
Check validity, 3254, 3550
Floating-point numbers, 2254, 2459
Invalid , 2254
FLOOR, 3359, 3659, 4242, 4414, 5483, 5696, 6183, 6321
Flutter monitoring, 2064
FM
With separate MPI address, 881
FM 350-1, 9109
FM 350-2, 9163
Channel setting, 9166
FM 351, 10458
Startup, 7127
FM 352, 9358
Startup, 7158, 7161
FM 355, 9836
FM 355-2, 9868
FM 450-1, 9190
FM 451, 10548
Startup, 7131
FM 452, 9433
Startup, 7158, 7161
FM 455, 9916
FMT_CJ_T
In/out parameters, 7266
Input parameters , 7265
Instruction, 7265
Output parameters, 7266
FMT_DS1
In/out parameters, 7252
Input parameters , 7251
Instruction, 7251
Output parameters, 7252

FMT_PAR
In/out parameters, 7248
Input parameters , 7248
Instruction, 7247
Output parameters, 7248
FMT_PID
Block diagram, 7238
Input parameters , 7230
Instruction, 7226
Mode of operation, 7228
Output parameters, 7230
Static variables, 7231, 7233, 7235
FMT_PV
In/out parameters, 7264
Input parameters , 7263
Instruction, 7259
Output parameters, 7264
FMT_TUN
In/out parameters, 7256
Input parameters , 7255
Instruction, 7254
Output parameters, 7255
Static variables, 7256
FN, 3924, 5929
Folder link
Editing, 8010
Removing, 8010
Renaming, 8010
Following error, 10243
Following error monitoring, 10242, 10243
Font settings
Migrate, 150
Font size
Asian languages , 8914
FOR, 4262, 6194
Force
Force all, 3043
Stop forcing, 3045, 3046
Force job, 1837
On SD card, 33
Force table
Meaning of the icons, 3025
"Monitor once and now" command for tags, 3037
Basic mode, 3024
Display, 3026
Expanded mode, 3024
Functionality, 3021
Layout, 3023
Meaning of the columns, 3023
Monitoring and modifying modes, 3009
Opening, 3026
Overview of the display formats, 3030
Overview of the test options, 3021
Permitted operands, 3028
Permitted operands for force values, 3029
Saving, 3027
Switching between basic mode and expanded mode, 3024
Syntax check, 3027
Test options, 3021
Force value
Permitted operands, 3029
FORCE355
Input parameters, 7205
Instruction, 7204
Output parameters, 7206
FORCE455
Input parameters, 7286
Instruction, 7284
Output parameters, 7286
Forcing tags
Safety precautions, 3023, 3039
Foreground color
Dynamization, 8047
Form exponential value, 4172, 6163
Form square, 4358
Form square root, 4360
Format, 8079
Formatting
Alarm text, 8167
FP, 3926, 5930
FR, 3928, 3946, 5932, 5950
FRAC, 3298, 3597, 4180, 4370
Fragment, 925, 930, 960, 966
Frame configuration
Instructions, 5180, 7467
Frame error, 9401, 9477, 10523, 10620
Frame size IO device, 1932
frame structure, 1002
Frames for Motion Control, 10217
Free layer 2 access, 662
Free space critically low, 8363
Free user events, 1830
Free-form comments
Deleting, 2594, 2635
Editing, 2592, 2634
Inserting, 2592, 2633
Introduction, 2591, 2633
Freeport
Code transparency, 994
End criteria, 992
Message end, 990
Message start, 990
Receive buffer, 995

Receiving data, 1008
Sending data, 1008
Start criteria, 991
Freeport protocol, 989
FREEZE, 1356
FREQUENC, 7345
Frequency measurement, 9008, 9093
Frequency measurement (CPU 31xC), 1699
Frequency measurement with FREQUENC, 7345
Frequency meter, 2032, 2035
FTP, 741, 743, 1613, 1888, 5134
FTP client
S7-1500, 1957
FTP configuration, 1942
File allocation table, 1942
FTP connection
Parameter field for, 7827
FTP server
S7-1500, 1950
FTP_CMD, 5134, 7829
FTPS, 5134
FTPS certificates, 733
Full-duplex mode, 1033
Full-duplex operation, 979
Function
Assigning to a function key, 8061
Change cam edges, 9368, 9443
Set process value, 9361, 9437
Set process value on-the-fly, 9361, 9437
Set reference point, 9366, 9442
Undo set process value, 9361, 9437
Zero offset, 9364, 9439
Function (FC)
Creating, 2502
Definition, 2170
exporting to an external source file, 2865
Function block (FB)
Creating, 2502
Definition, 2171
exporting to an external source file, 2865
Instance data block, 2171
Function Block Diagram, 2611
Function key, 8056
Assigning a function, 8061
Assigning a graphic, 8064
Global assignment, 8058
Global screen, 7972
Local assignment, 7972, 8060
protect with password, 8063
Use global assignment, 7972
used for screen navigation, 8068
Function list, 8134, 8276, 8368
Asynchronous execution, 8281
Execution, 8278
Execution in Runtime, 8281
Synchronous execution, 8281
System function, 8278
User-defined functions, 8278
Function lists
Editing, 8280
Function switches
ABS_CTRL, 7143
ABS_CTRL_451, 7145
CAM_CTRL, 7176
CAM_CTRL_452, 7177
Functional description
High_Speed_Counter, 4920
Functional scope
ProSave, 8894
Functionality
PROFIBUS DPV0/DPV1, 1385
Functionality of the force table, 3021
Functions
Updating tag value, 8111, 8413
FUZ_355
In/out parameters, 7204
Input parameters, 7203
Instruction, 7202
Output parameters, 7203
FUZ_455
In/out parameters, 7284
Input parameters, 7283
Instruction, 7282
Output parameters, 7284
FX1 series, 8692
FX2 series, 8692
FX3
PLC, 8678
FX3 series, 8682

G

GADR_LGC, 4820
Gate
Hardware, 9100
Internal, 9101
Software, 9100
Gate control, 8980, 8984, 9008, 9035, 9058
in isochrone mode, 9104
gate function
cancel, 9103
interrupt, 9103
Gate stop function, 9101
Gateway, 534

Gear
Easy Motion Control, 10773
GEN_DIAG, 4758
General information on troubleshooting, 10973
General rules, 2186
Generate off-delay, 4337
Generate on-delay, 4334
Generate pulse, 4331
Generating diagnostics information, 4758
Generation of packed addresses, 1360, 1428
GEO_LOG, 4816
GEO2LOG, 4808
Geographic coordinates, 1222
GET, 5038
Get_AlarmState, 4725
GET_DIAG, 4761
Get_Features, 978, 1009
Get_IM_Data, 4742
GET_IM_DATA, 50
GET_NAME, 31, 4746
GetBlockName, 4549
GetError, 2824, 2825, 3388, 3688, 3849, 4278, 4433
GetError , 2823
GetErrorID, 2823, 2824, 2825, 3392, 3692, 3852, 4282, 4437
GetGroupNumber, 8309
GetInstanceName, 4549
GETIO, 4560
GETIO_PART, 4562
GetPassword, 8309
GetPLCMode, 8310
GetStationInfo, 4750
GetSymbolName, 4548
GetUserName, 8308
Gigabit address, 736
Global assignment
Of a function key, 8056, 8058
Global data block, 2172
Configuring, 8421
Creating, 8421
Retentive behavior, 2840
Global firewall rules, 747
Assigning, 749
Global libraries, 416
Archiving, 437
Backward compatibility with older product versions, 430
Compatibility mode V12 SP1, 429
Retrieving, 438
Global library, 8762
Adding types, 460
Archiving, 437
Cleaning up, 470
Closing, 435
Create, 8771
Creating, 428
Creating folders, 441
Deleting, 436
Displaying logs, 434, 8773
Migrating, 157
Open, 8772
Opening, 430
Overview of the USS protocol, 5216, 7503
Save, 8772
Saving, 434
Showing properties, 433
Updating the project, 462
Upgrade, 8875
Using filter view, 443
Using the element view, 419
Using types, 461
Global packet filter rules, 749
Global screen, 7971
Function key, 7972
Template, 7973
Global softkey, 8058
Global tag, (See PLC tag)
GMRP, 1298
Go online, 1871, 2125
connecting several devices, 10856, 10857
Multiple TIA Portal instances, 37
Going offline, 2127
GOTO, 4269, 6201
GoToEnd, 8298
GoToHome, 8297
GPRS, 1165
GRAPH, 46
Acknowledging supervision errors, 2985
Actions, 2728, 2730, 2732, 2735, 2736, 2737, 2792, 2797, 2798, 2799
Alarm view, 2770
Alternative branch, 2722, 2787
Block calls, 2812, 2813, 2814
Block interface, 2741, 2743, 2746, 2749, 2751, 2752
Changing the settings, 2757
Close branch, 2723, 2789
Conditions, 2739, 2800, 2801, 2802, 2809
Data type basics, 2795
Declaring internal parameters as retentive, 2761
Declaring internal parameters as visible and accessible from HMI, 2761
Deleting a branches, 2790
Detailed comparison, 2886

Disable all steps, 2984
Initialize sequencer, 2984
Interlock, 2740
Jump to step, 2723, 2790, 2791, 2792
Migration, 136
Operating modes, 2981, 2985
Permanent instructions, 2739
Permanent post-instructions, 2769
Permanent pre-instructions, 2764
Program status display, 2991
Programming language, 2718
Programming window, 2762
Resetting registration requirements, 49
Sequence end, 2723
Sequence view, 2765
Sequencers, 2720, 2721, 2774, 2775, 2776, 2777, 2778
Settings, 2756
Simultaneous branch, 2721, 2788
Single step view, 2767
Specific operands, 2727
Specifying the interface parameter set, 2759
Specifying the memory space model, 2760
Specifying the programming language, 2758
Step, 2724, 2725, 2779, 2780, 2781, 2783, 2784, 2785, 2786
Supervision, 2740
System synchronization, 2982, 2986
Test settings, 2982, 2983
Testing with program status, 2981
Transitions, 2726, 2779, 2781, 2783, 2785, 2786
Graphic
add to graphics library, 8841
Alignment, 7999
Assigning to a function key, 8064
Button, 8100
Distance, 8000
Graphic view, 8083
Inserting, 7982
managing, 8009
Size adjustment, 7999
Stretching, 8083, 8085
Using from the graphic browser, 8009
With transparent background, 8009
Graphic browser, 8009
Graphic browser , 8842
Graphic I/O field, 8084
Output graphics list, 8042
Graphic IO field, 8927
Application, 8927
Behavior, 8928
Operation, 8928
Graphic view, 8082, 8927
Application, 8927
Layout, 8927
Operation, 8927
transparent color, 8083
Graphics, 8011
Editor, 8840
Graphics list
Application, 8033
Bit (0, 1), 8039
Bit number (0 - 31), 8041
Creating, 8034
Graphic I/O field, 8042
Outputting configuration data, 8042
Range (0 - 31), 8035
Range (... - ...), 8038
Gray code, 9017
Greater than step activation time, 4494, 4500, 6400, 6406
Greater than uninterrupted step activation time, 4495, 4501, 6401, 6406
Gross volume of data, 1932
group
Cancel, 8015
Edit, 8014
Ungroup, 8015
Group
Adding objects, 8016
Cancel, 8015
Creating, 8014
Editing, 8014
Removing an object, 8017
Ungroup, 8015
Group error LED, 9861, 9910
FM 350-1, 9157
FM 351, 10531
FM 352, 9420
SM 338, 10427
Group name, 750, 753
Group properties, 786
GSD files
Configuring devices (PROFIBUS), 1383, 1435
GSD revisions (PROFIBUS), 1382, 1434
Installation, 1383, 1435
GSD files (PROFINET), 1462
Changing revision, 1464
Installation, 1464
GSDML, (See GSD files (PROFINET))
GSM network, 1165
Gx_STW, 10761, 10762
Gx_XIST2, 10762

H

Half-duplex mode, 1033
Half-duplex operation, 979
Handshaking, 984, 1602
Hardware, 50
  Comparing, 31
  Configuring and assigning parameters, 494
  Detection, 42
  Edit parameters, 514
  Edit properties, 514
Hardware and network editor
  Device view, 480, 8389
  Network view, 477, 8383
  Topology view, 482, 8391
Hardware and software limit switches
  Function, 10015
Hardware catalog
  Adding device, 505
  Browsing, 497
  DP slave, 1341, 1413
  DP/DP coupler, 1342, 1414
  I slave, 1414
  I-slave, 1342
  Selecting CPUs, 872
  Selecting fail-safe modules, 874
  Selecting the hardware component, 503
  Task card, 489, 8395
Hardware compatibility, 1860
Hardware configuration, 1839
  Adding device, 504
  Adding module, 507
Hardware configuration for Motion Control S7-1200, 10006
Hardware data flow control, 985
Hardware data types, 2283
Hardware detection, 507
Hardware diagnostics, 2079
Hardware documentation, 10981
Hardware editor
  Components, 476, 8381
  Function, 475, 8381
  Hardware catalog, 489, 8395
  Inspector window, 8393
  Inspector window , 487
Hardware gate, 8980, 8984, 9035, 9058, 9100
  edge-triggered opening and closing, 9103
  level-triggered opening and closing, 9102
Hardware identifier, 881, 883, 886, 937, 2043, 2054, 5153, 5156, 5159, 5163, 5168, 5169, 5170, 5171, (See Hardware identifier), (See Hardware identifier)
  Determining with LOG2MOD, 4811
Hardware interrupt, 1623, 9017, 9143, 9180, 9211
  Enable, 9052
  enabling, 9144, 9180
  Enabling, 9211
  Example, 1856
  How it works, 1583
  I-slave, 1394
  lost, 9051
  OB 40 to OB 47 (S7-300, S7-400), 1762
  Settings, 1678
Hardware interrupt (S7-1500), 1902
Hardware interrupt OB
  Description, 1623
  Parameter assignment, 1631, 1918
Hardware interrupt OBs (S7-1500), 1902
hardware interrupts
  Enable, 9414, 9490
Hardware limit switches , 10245
Hardware requirements, 91
  Communications instruction "AS_MAIL", 10971
  Communications instruction "PG_DIAL", 10967
  Communications instruction "SMS_SEND", 10970
Hardware requirements for AS-AS remote link
  Communication instruction "AS_DIAL", 10968
Hardware RTS always ON, 986
Hardware RTS always ON, ignore DTR/DSR, 986
Hardware RTS always switched, 987
Hardware support package
  Installing, 107
Hardware version, 1196
Hardware-controlled data flow control, 1601
Harmonizing
  Object size, 7987
HART, 2023
HART variables
  Configuring, 2062
  Quality code, 2063
  Structure, 2062
Help
  Browsing, 324
  Call topic from favorites, 325
  Delete topic from favorites, 326
  Find keywords, 324
  Full-text search, 324
  Identification of help topics, 321
  Open tooltip cascades automatically, 327
  Opening, 323
  Printing help topics, 326
  Save topics in favorites, 325
  Use index, 324

Help indicator, 8085
Hidden input, 8081
Hidden parameters, 2603, 2644
High count limit, 9114, 9170, 9240, 9301
High limit, 9140, 9258, 9316
High_Speed_Counter, 4920, 9024
  Basic parameters, 9029
  Call, 4920
  Commissioning, 9042
  Configuring, 9027
  Description, 4920
  Diagnostics, 9043
  Error response, 4923
  Functional description, 4920
  Input parameters, 4924
  Output parameters, 4925
  Programming, 9041
  Static tags, 4927
Highlighting
  Connection, 8405
High-speed counter
  Configuring, 1597
  General, 1594
  How it works, 1594
HMI backup
  Deleting, 8879
  Display, 8878
  Rename, 8879
HMI connection, 177, 554, 1613, 1888, 8379
  Creating, 8431, 8479, 8533
  MPI, 8554
  MPI parameters, 8556
  Password, 8438, 8450, 8487, 8500
  PROFIBUS, 8443, 8546
  PROFIBUS parameters, 8446, 8494, 8548
  PROFINET, 8430, 8431, 8478, 8479, 8532, 8533
  PROFINET parameters, 8433, 8480, 8535
  S7 1200, 8479
  S7 1500, 8431
  S7 300/400, 8533
HMI connections
  Devices & Networks, 8403
HMI device, 100
  Changing the device type, 8900
  Commissioning, 8903
  Configuration, 8876
  Data backup, 8892, 8894, 8895, 8910
  Image, 8892
  Load, 8892
  MPI parameters, 8558
  Performance features, 8961, 8965
  PROFIBUS parameters, 8448, 8496, 8549
  PROFINET parameters, 8434, 8482, 8536
  Recommissioning, 8903
  Reloading the operating system, 8897
  Reset to factory settings, 8898
  Restoring data, 8894, 8896, 8910
  Software, 8892
  System limits, 8961, 8965
HMI device configuration, 8876
HMI device licensing
  Non-PC-based, 100
HMI device on a PLC
  Commissioning, 8645, 8666, 8684, 8694, 8711, 8723, 8740
HMI device replacement, 57
HMI device type
  Changing, 8900
HMI device version, 8875, 8876, (Device version), (Device version)
HMI device wizard, 8760
HMI style, 8782
HOLD, 1650
Home position, 10225
Home position offset, 10225
Homing, 10223
  Absolute value adjustment, 10224
  Active homing, 10224
  Direct homing, 10224
  Homing mode, 10224
  Homing modes, 10017
  Homing on the fly, 10224
  Homing status, 10223
  Passive homing, 10224
Homing mark, 10224
Homing mark position, 10225
Homing mode, 10225
Horizontal radius, 8082
Hot restart, 1643, 1674
  with RESUME, 7404
Hot restart (S7-300, S7-400), 1807
Hot restart of the CPU, 1153
How the frequency meter works, 2033
HSC, 1594
HSP, (See support package)
HTA, 4539, 6445
HTTP, 750
  Load/save, 1225
HTTPS, 822, 912, 946, 1687
HTTPS server only, 1219
HW ID, (See Hardware identifier)
  See Hardware identifier, 937
Hysteresis, 8981, 8982, 9014, 9015, 9039, 9067, 9143, 9173, 9393, 9468

Position-based cams, 9393, 9468
Time-based cams, 9394, 9469
Hysteresis range, 9345

I

I address, 881, 937
I device, 1492
I device diagnostics, 1498
I slave
Hardware catalog, 1414
I&M data: read out with Get_IM_Data, 4742
I/O, 39, 52
Direct access to, 1577, 1845
Direct access to ~, 1743
I/O access error, 1575, 1661, 1662
I/O access error (S7-1500), 1914
I/O access error OB, 47
I/O access error OB (S7-1500), 1914
I/O access error OB (S7-300, S7-400), 1814
I/O address, 881, 937, 1577, 1845
I/O addressing, 1670
I/O data area
Access, 1667
Data records of the module, 1669
Diagnostics data, 1668
Parameter data, 1668
User data, 1668
I/O field, 8079
Data format, 8080
Decimal format, 64
Format, 64
Hidden input, 8081
Mode, 8080
I/O input, 1572
I/O output, 1572
I/Os, 2210
I_ABORT, 7916
IB, 2477
ICMP, 756
Icon
For comparison, 2085
For comparison status, 2133
for connection status, 2133
For hardware diagnostics, 2084
for operating state, 2086
For software diagnostics, 2085
For value comparison, 10032, 10058
Overlay icon, 2086
Icons
in TeleService, 10921
Icons in the force table, 3025
Icons in the watch table, 2997
ID, 2477
Identifiers of elements and blocks, 1745
I-device, 1450
Idle line, 991
IE/AS-i link PN IO, 1437
IE/PB Link PN IO, 2003
IEC check, 2372
Setting, 2374
IEC Check
Setting, 2288, 2420
IEC counters, 249
IEC timer, 2257
IEC timers, 249
IEC V2.2, 1873
IEC V2.3, 1873
IEEE 802.3, 746
IEEE tag, 2022
IEEE-compliant LLDP, 1873
IF, 4258, 6190
IF1B
Basic Panel, 8427
IGMP, 1298
IGMP querier, 1298
IKE settings, 786
IM 155-6 DP HF, 39
IM 174
Configuring, 10749
Image, 8875, 8883, 8884, 8896
HMI device, 8892
Image file
Storing in the image browser, 8011
Storing in the image browser , 8842
IMC, 3434, 3735, 3881, 4316, 4473, 5563, 5773, 5878, 6250, 6378
Import
Alarm, 8801
Analog alarm structure, 8802
Project texts, 8819, 8839
Recipe, 8798
Structure of discrete alarms, 8805
Structure of recipe data, 8799
Tag, 8809
Text list, 8814
Importing a recipe
Structure for the import, 8799
Importing analog alarms
Structure for the import, 8802
Importing discrete alarms
Structure for the import, 8805
Importing NTP servers, 778

in the screen
Arrange object, 7991
IN_RANGE, 3251, 3547
INC, 3275, 3572, 3971, 5974
Incoming, 8152, 8359
IncreaseFocusedValue, 8296
IncreaseTag, 8297
Increment, 3275, 3572, 3971, 5974
Increment number 1-100, 10483, 10574
Increment number 254, 10484, 10576
Increment number 255, 10484, 10576
Incremental actual value, 10220
Incremental approach, 10482, 10573
Absolute, 10482, 10573
Cancel, 10487, 10579
Data in the channel DB, 10486, 10487, 10578, 10580
Data in the parameter DB, 10486, 10578
Increment number 1-100:, 10483, 10574
Increment number 254 , 10484, 10576
Increment number 255, 10484, 10576
Incremental dimension number 252 , 10575
Relative, 10482, 10573
Requirement, 10483, 10574
Shutdown, 10487, 10579
Incremental approach with increment number 254
Sequence, 10484, 10576
Incremental approach with increment number 255
Sequence, 10484, 10576
Incremental approach with incremental dimension number 252
Sequence, 10575
Incremental dimension number 1 to 100, 10524, 10622
Incremental dimension number 252, 10575, 10623
Incremental dimension number 254, 10526, 10623
Incremental dimension number 255, 10526, 10624
Incremental dimension tables, 10465, 10555
changing, 10556
Changing, 10465
Data in the channel DB, 10466, 10556
reading, 10556
Reading, 10466
Writing, 10465, 10556
Incremental encoder, 9348, 10454
Increment, 9350, 10455
Increments per encoder revolution, 10518, 10616
Pulse, 9350, 10456
Reaction times, 9351
Response times, 10456
Signal forms, 10455
Signal shapes, 9350
Increments, 10524, 10621
Indenting and outdenting
Lines, 2693
Index of a PC station, 1155
Index tag, 8132, 8133
Indirect addressing, 246, 2220, 2221, 2222, 2224, 2225, 2229, 2230, 2231, 2232, 2233, 2234
Indirect addressing , 8132
Industrial Ethernet, 8398
Information
ARP table, 1197
Ethernet Statistics, 1208
Event log, 1198
Frame type, 1209
Packet Errors, 1209, 1210
Spanning Tree, 1200
Versions, 1195
Information system
Components of the information system, 320
Roll-out, 322
Tooltip, 322
Infotext, 8158
Creating, 8163, 8165
INIT_RD, 3394, 3695, 3855, 4285, 4439
Initial state of receive line, 1081
Initial step
Canceling, 2780
Specifying, 2780
Initialization, 1153
IPE file, 8784
Project file, 8790
Initialization (user-defined web pages), 914, 949
Initialize all retain data, 3394, 3695, 3855, 4285, 4439
Initializing
Controller data, 8783
IPE file, 8785
Project file, 8790
Initiator, 9349, 9352
Input
Insert, 5586
Inserting, 2598, 3451
remove, 2599, 2640
Input (I), 1572
Input byte (IB), 1572
Input driver
Call, 7058
Error handling, 7059
Function, 7058
Initialization, 7058
Simulation mode, 7059
Input filter, 9125, 9201

Input finished, 8358
Input parameter PID_ES, 6922
Input parameters
High_Speed_Counter, 4924
LP_SCHED_S, 7033
Input parameters of GRAPH function block, 2743
Input word (IW), 1572
INSERT, 4546
Insert empty line, 10062
Insert input, 2639
Insert separator line, 10062
Inserting
Graphic, 7982
Library object, 8779
Object, 7981, 7982
Rectangle, 8023
Inserting a comment section, 4271, 6203
Inserting/removing the USB network adapter, 1988
Inspector window
Cross-reference, 63
Event, 63
Hardware and network editor, 8393
Layout, 293
Reducing automatically, 288
Inspector window
Diagnostics tab, 2080
Hardware and network editor, 487
Install update, 107
Installation
Displaying software, 108
Licenses, 84
Log, 103
Modifying products, 109
ProSave, 73
Repairing products, 111
run automatically, 83
Smartdrive, 59
Starting, 104
Support package, 107
System requirements, 85
Target directory, 83
Updates, 107
Updating products, 109
Installing CA certificate, 10955
Installing license keys, 88
Installing support packages, 107
Installing the local modem, 10930
Instance data block
Creating, 2174
Definition, 2174
Modifying the data types of IEC timers and IEC counters, 2702
Retentive behavior, 2839
Instance DB
Creating, 3056, 3064
Instruction, 7850
AG_SEND / AG_LSEND / AG_SSEND, 7719
Copying, 2717
Cutting, 2717
Deleting, 2717
Inserting, 2717
Rules, 2697
Specify data type, 2583, 2584, 2624, 2626, 2666
Versions, 2534
Instruction profile
Activating and deactivating, 2538
Basics, 2535
Creating, 2536
Deleting, 2539
Editing, 2537
Opening, 2537
Instructions
AG_LOCK / AG_UNLOCK, 7728
AG_RECV / AG_LRECV / AG_SRECV, 7724
for configured connections, 7715
for DP mode, 7867
For S5S5 connection, 7853
General information, 7850
P3964_Config (protocol configuration), 5193, 7480
Port_Config (port configuration), 5184, 7472
Receive_Config (receive configuration), 5188, 7476
Receive_P2P (receive point-to-point data), 5197, 7484
Receive_Reset (reset receiver), 5199, 7486
S7-400 Automation System, 7637
Send_Config (send configuration), 5187, 7474
Send_P2P (send point-to-point data), 5195, 7482
Signal_Get (get RS232 signals), 5199, 7486
Signal_Get (set RS232 signals), 5200, 7487
Symbolic name, 7850
USS_Drive_Control, 5223, 7510
USS_Port_Scan , 5220, 7507
USS_Read_Param, 5225, 7512
USS_Write_Param, 5227, 7514
Instructions for access coordination with FETCH/WRITE, 7728
Instructions for configuring a remote modem, 10932
INT, 2245, 2296, 2337, 2380, 2401, 2423, 2436
INT_TO_, 2337, 2401, 2436
INTEG
Input parameters, 6963
Instruction, 6960

Output parameters, 6963
Integer, 3361, 3660, 4025, 4243, 4415, 5485, 5697, 6028, 6184, 6323
16-bit, 2245, 2246
32-bit, 2247, 2248
64-bit, 2249, 2250
8-bit, 2244, 2245
Integers, 2458
Integrated
Connection, 8403, 8407
Integrated connection, 8112, 8414
Integrated project
Converting an unspecified CPU, 176
Creating an integrated HMI connection, 177
Deleting an unspecified connection, 180
Linking HMI tags, 179
Migrating, 161, 172
Post-editing, 175
Integrating texts from text lists in alarms, 3072
Example, 3072
Integration time, 9088
Intelligent DP slave, (See I-slave), (See I-slave)
Intelligent IO device, 1492
Inter Project Engineering
Creating "Device Proxy Data" in the source project, 3079
Exchange of controller data via "IPE file", 3080
Exchanging controller data across projects, 3077, 10845
Opening "Device Proxy Data", 3080
Procedure for exchanging controller data via a project file, 3078, 10846
Procedure for exchanging controller data via an IPE file, 3077, 10845
Procedure for updating already transferred controller data, 3078, 10847
Requirements, 3076, 10845
Software and hardware requirements, 3076, 10844
Inter Project Engineering (IPE)
Basics, 3074, 10842
Exchanging controller data via IPE file, 3075, 10843
Exchanging controller data via project file, 3075, 10843
Interconnecting, 536
Interconnecting ports
Graphic view, 718, 720
Table view, 719, 721
Interface, (See block interface)
Adding, 10864
Basic Panel, 8427
Displaying, 2092, 2150
Renaming, 721
Interface configuration
Instructions, 5180, 7467
Interface module, 876
Interface parameter set, 2741, 2743, 2746, 2759
Interface properties, 2092, 2150
Interfaces, 31, 974
Interfaces (PROFINET), 1871
Interference frequency suppression, 1357, 2056
Interlock
Introduction, 2725, 2740
Programming, 2800
Internal
Ethernet port, 8681
Internal errors, 9861, 9910, 9942
Internal Ethernet port
Open settings, 8681
Internal Gate, 9101
Internal network nodes
Configuring, 792
Diagnostics, 804
Internal parameters, 46
declaring as retentive, 2761
Declaring as visible and accessible from HMI, 2761
Internal reference junction, 2057
Internal tags
Data exchange, 8379
Internet Key Exchange (IKE), 787
Inter-Project_Engineering
IPE, 8783
Interrupt
Delaying with DIS_AIRT, 6628
Disabling with DIS_IRT, 6624
Enabling with EN_AIRT, 6628
Enabling with EN_IRT, 6626
Sending in the I-slave, 1370
Interrupt event
Delaying with DIS_AIRT, 4721
Disabling with DIS_IRT, 4718
Enabling with EN_AIRT, 4722
Enabling with EN_IRT, 4720
Interrupt stack, 1664
Interrupts
Activating time-of-day interrupt with ACT_TINT, 4703, 6607
Canceling time-delay interrupt with CAN_DINT, 4708, 6612
Canceling time-of-day interrupt with CAN_TINT, 4702

Canceling time-of-day interrupts with CAN_TINT, 6607
Cyclic interrupt (S7-300, S7-400), 1760
Diagnostic interrupt (S7-300, S7-400), 1783
Hardware interrupt (S7-300, S7-400), 1762
OBs for DPV1, 1386
Pull/plug interrupt (S7-300, S7-400), 1786
Querying time-delay interrupt with QRY_DINT, 4708, 6613
Querying time-of-day interrupt with QRY_TINT, 4704, 6608
Reading interrupt status with the system status list, 6704
Receiving from I/O module with RALRM, 4564
Receiving from the I/O module with RALRM, 6468
Sending an I-slave with SALRM, 6508
Setting time-of-day interrupt with SET_TINT, 4697, 6605
Setting time-of-day interrupt with SET_TINTL, 4699
Starting time-delay interrupt with SRT_DINT, 4706, 6611
Time-delay (S7-300, S7-400), 1759
Time-of-day interrupt (S7-300, S7-400), 1755
Triggering process interrupt with DP_PRAL, 6515
With packed addresses, 1365, 1433
Intrusion Detection System (IDS), 93, 97
INV, 3404, 3704, 4441, 5510, 5722, 6339
Invalid license
With a time zone change, 89, 102
InverseLinearScaling, 8301
Invert, 3158, 3404, 3452, 3704, 5376, 5510, 5587, 5722
InvertBit, 8299
InvertBitInTag, 8300
Inverting, 3920, 4017, 4018, 5925, 6019, 6020
InvertLinearScaling, 8301
IO device
Networking, 1453
Update time, 1456
Watchdog time, 1457
IO field, 8924
Application, 8924
Behavior, 8925
Changing between input fields, 8925
IO field, 8926
Layout, 8925
Operation, 8925
Touch operation, 8926
IO routing, 1471
IO system, 1453, 1540, 1543, 1546
Creating, 1453
IO2MOD, 4812
IO-Link, 1552
IP access control list, 742
IP access protection, 598, 1933, 1936
IP access control list, 1935
IP address, 33, 534, 536, 1441, (See display)
IP address parameters, 1442, 1450
IP addresses
PROFINET, 1465
IP addressing using DHCP, 596
IP configuration
Change parameters from the user program, 5122, 7449
IP packet filter rules, 756
IP parameters, 2092
IP protocol, 8483
IP rule sets, 747
IP services, 750
IP subnet, 536
IP subnet address, 1871
IP_CONFIG, 7812
IP-ACL (IP access control list), 1934
iPad, 937
IPE, 8783
File, 8784, 8785
IPE data
Updating, 8788, 8793
IPE file, 8784
Initialization, 8784
Initializing, 8785
IPsec settings, 786
IPv4 addresses, 761
IPv6
E-mail, 2073
FETCH/WRITE, 2073
FTP client, 2073
FTP server, 2073
Notation, 2073
SNMP, 2073
Use with the CP 1543-1, 2073
IPv6 addresses, 761
IQ sense, 2065
iQ series, 8682
IRT, 1477
IRT (Isochronous Realtime Ethernet), 1472, 1476, 1480
IS_ARRAY, 3263, 3559, 4161
IS_NULL, 3261, 3556
ISAKMP, 796
I-slave, 1345, 1367, 1415, 1423
Configuring, 1345, 1368, 1424

Configuring a DP slave, 1374
Configuring CP, 1373
Configuring CPU, 1373
Data access, 1424
Data exchange, 1345, 1415
Devices, 1342
Diagnostic addresses, 1368
Direct data exchange, 1346
DP master, 1345
Hardware catalog, 1342
Sending interrupt with SALRM, 1370, 6508
Slot model of the DP slaves, 1391
Triggering hardware interrupt, 1394
Triggering process interrupt with DP_PRAL, 6515
ISO protocol, 834
Isochronous mode, 1486, 1553, 1559, 9024
Combination of DP with and without isochronous mode, 1403
Configuring, 1403, 1487, 1488
CPU properties, 1405
Creating a user program, 1408
Delay time, 1405
DP cycle time and Ti/To values, 1407
Effect of Ti, 1483
Effect of To, 1484
Example: Capture of multiple process tags, 1401
Not selectable, 1406
Process image partitions, 1405
Properties on PROFIBUS, 1406
Rules, requirements, and supplemental conditions, 1402, 1490
Same Ti/To times for all slaves, 1408
Sequence, 1482
Setting on the subnet, 1489
Settings on the DP slave, 1408
System reserves, 1412
Ti/To overlap, 1411
Why?, 1480
Isochronous mode , 1397
Isochronous mode interrupt OB, 1400
Assigning parameters, 1919
OB 61 to OB 64 (S7-300, S7-400), 1776
Isochronous mode interrupt OBs (S7-1500), 1903
Isochronous PROFIBUS, (See Isochronous mode)
ISO-on-TCP
Characteristics, 687
TSAP, 696
ISO-on-TCP connection
Parameter field for, 7825
ITB, 4011, 6014
ITD, 4013, 6016
IW, 2477
IX, 229, 2477

J

JBI, 4037, 6039
JC, 4032, 6034
JCB, 4034, 6036
JCN, 4033, 6035
Jerk limit, 10249, 10285, 10287, 10301, 10303
Function, 10016
JL, 4055, 6057
JM, 4048, 6050
JMP, 3372, 3672, 5491, 5703
JMP_LIST, 3375, 3675
JMPN, 3373, 3673, 5492, 5704
JMZ, 4052, 6053
JN, 4044, 6046
JNB, 4036, 6038
JNBI, 4038, 6040
JO, 4040, 6041
Job
Cancel set actual value, 10489, 10581
Set actual value, 10488, 10580
Set reference point, 10490, 10584
Setting actual value on the fly, 10580
Zero offset, 10582
Job buffer, 662, 664, 7853
Job execution
Order, 7143, 7145
Job header, 662, 664, 7718, 7853, 7855
Job mailbox, 8380
Transferring data, 8463, 8514, 8572, 8610, 8757
Job management, 7129
Job status, 7147
ABS_CTRL, 7144
CAM_CTRL, 7176
Jobs
ABS_CTRL, 7142
ABS_CTRL_451, 7144
ABS_DIAG_451, 7132
CAM_CTRL, 7175
CAM_CTRL_452, 7176
CAM_DIAG_452, 7162
CNT_CTL1, 7099
CNT_CTRL, 7097
Jog, 10473
Cancel, 10475
Data in the channel DB, 10474
Monitoring functions, 10475
Operating range limit, 10475
Requirement, 10473
Sequence, 10473

Shutdown, 10474
Jogging, 10564
Cancel, 10566
Data in the channel DB, 10565
Easy Motion Control, 10771
Monitoring functions, 10566
Operating range limit, 10566
Requirement, 10564
Sequence, 10564
Shutdown, 10565
JOS, 4041, 6043
JP, 4046, 6048
JPZ, 4050, 4052, 6051, 6053
jQuery, 937
JU, 4030, 6032
Jump, 3372, 3373, 3374, 3375, 3377, 3380, 3672, 3673, 3674, 3675, 3677, 3680, 4029, 4055, 5491, 5492, 5493, 5494, 5703, 5704, 5705, 5706, 6031, 6057
Ending sequencers, 2777
if BR = 0, 4038, 6040
if BR = 1, 4037, 6039
if OS = 1, 4041, 6043
if OV = 1, 4040, 6041
if RLO = 0, 4033, 4036, 6035, 6038
if RLO = 1, 4032, 4034, 6034, 6036
If the result is greater than or equal to zero, 4050, 6051
if the result is greater than zero, 4046, 6048
if the result is invalid, 4053, 6055
if the result is less than or equal to zero, 4052, 6053
if the result is less than zero, 4048, 6050
if the result is not zero, 4044, 6046
if the result is zero, 4043, 6044
Unconditional, 4030, 6032
Jump distributor, 3377, 3677
Jump label, 3374, 3674, 4029, 4269, 5493, 5705, 6031, 6201
Jump list, 3375, 3675, 4055, 6057
Jump to step
Changing the jump destination, 2791
Deleting, 2792
Inserting, 2790
Introduction, 2723
Navigating to the jump destination, 2791
JUO, 4053, 6055
JZ, 4043, 6044

K

Key operation
Date/time field, 8924
IO field, 8926
Keyboard operation, 492
Basic functions of the TIA Portal, 308
Customizing editors, 313
Editing objects, 314
Editing texts, 315
Navigation in the TIA Portal, 311
Online functions, 317
Project editing, 310
Selecting objects, 314
Tables, 316
Window, 310
KEY-PLUG, 1252, 1254
Formatting, 1254
Keyword highlighting, 2689
Keywords, 2197
Know-how protection
Changing a password, 2935
Introduction, 2929
Opening blocks, 2933
Printing block, 2934
remove, 2935
Setting up, 2932

L

L, 3929, 3947, 3991, 5934, 5951, 5994
L DBLG, 4062, 6063
L DBNO, 4063, 6064
L DILG, 4064, 6064
L DINO, 4065, 6065
L STW, 3993, 5996
LABEL, 3374, 3674, 5493, 5705
LACP, 1290
LAD, 2568, 3084, 3117
Detailed comparison, 2876
Migration, 135
LAD element
Copying, 2595
Cutting, 2596
Deleting, 2601
Inserting, 2581, 2582
Inserting an operand, 2601
pasting from the clipboard, 2596
Replacing, 2597
Rules for inserting, 2579

Selecting, 2594
LAD network
Branch, 2604
Closing a branch, 2606
Crossing, 2607
Deleting a branch, 2607
Deleting a crossing, 2609
Insert crossing, 2608
Inserting a block call, 2587, 2629
Inserting a branch, 2606
Program status display, 2987
Prohibited interconnections, 2581
Rearranging a crossing, 2608
Rules for simultaneous branches, 2605
Ladder Logic, 2568
LAG1ST
Input parameters, 6967
Instruction, 6963
Output parameters, 6967
LAG2ND
Input parameters, 6970
Output parameters, 6970
Language
Activate project language, 8830
Asian languages, 8828
Asian operating system, 8828
Disabling the project language, 8830
Editing language, 8830
Language support, 8828
Language-dependent format, 8827
Language-specific graphic, 8840
Log, 8850
multilingual project , 8832
Reference language, 8830
Regional format of the date, time, currency, and numbers, 8827
Language (web server), 910, 911
Language abbreviation, 970
Language abbreviations, 934
Language behavior
On-screen keyboard, 75
Language files
For diagnostics display, 1948
Language switching, 934, 970, 8844, 8913
Log, 8850
Runtime language, 8847
Languages
Migrating, 155
LAR1, 3994, 5997
LAR1 <D>, 3995, 5998
LAR1 AR2, 3997, 6000
LAR2, 3998, 6001
LAR2 <D>, 3999, 6002
Latch/Retrigger, 9117
Latching, 9115
Layer
Assigning objects to a layer, 8070
Layer 2, 746
Layer 3, 746, 1254
Layout
Alarm indicator, 8095, 8183
Alarm view, 8090, 8093, 8182, 8945
Bar, 8076
Button, 8100, 8931
Circle, 8086
Date/time field, 8078, 8923
Ellipse, 8082
Graphic I/O field, 8084
Graphic view, 8083, 8927
I/O field, 8080
IO field, 8925
Line, 8089
Rectangle, 8096
Simple recipe view, 8233, 8953
Simple user view, 8936
Switch, 8099, 8932
Symbolic I/O field, 8101
Text field, 8104
Trend view, 8087, 8929
User view, 8078
Layout of the force table, 3023
LC, 3931, 3949, 5936, 5953
LDB configuration, 1995
LDT, 2261, 2311, 2360
LDT_TO_, 2360
Lead and lag algorithm, 3438, 3740, 3891, 4320, 4482, 5567, 5778, 5888, 6254, 6382
Lead time, 9346
LEAD_LAG, 3438, 3740, 3891, 4320, 4482, 5567, 5778, 5888, 6254, 6382
Leadscrew, 10222
Learning functionality, 792
Learning mode, 833
LEAVE, 6107
LED, 4740
LED ACK, 8946
LED status, 2153
Reading out with LED, 4740
Reading the system status list, 6715
LEFT, 4543
LEN, 4541
Length
Area pointer, 8419
Area pointers, 8421

Length measurement, 10585
Data in the channel DB , 10587
Edge distance, 10587
faulty, 10587
Requirement, 10586
Sequence, 10586
LENGTH parameter, Send_P2P, 5197, 7484
LGC_GADR, 4821
Library, 8762
Adding a master copy, 442
Adding and using a block, 2504
Basics, 415
Buttons and Switches, 8766
Cleaning up, 470
Comparing library elements, 471
Copying a library object, 8765
Copying types to the clipboard, 465
Creating folders, 441
Cutting library elements, 465
Cutting master copies, 465
Cutting types, 465
Deleting instances, 466
Deleting types and versions, 466
Displaying the properties of a type, 449
Harmonizing names and path structure, 469
Master copies, 415
Master copy, 8765
Moving library elements, 466
Open, 8772
Opening a released type version, 448
Pasting master copies from the clipboard, 466
Pasting types from the clipboard, 465
PLC data types generated by the system, 49
Released type version, 447
Save, 8772
Storing an object, 8778
Style, 8782
Task card, 417
Type, 8765
Type version in progress, 446
Type version in test, 446
Types, 415
Using filter view, 443
Using master copies, 441, 444
Using the element view, 419
Using types, 444
Versioning of types, 445
Library for the USS protocol
General information on drive setup, 5228, 7515
Overview, 5216, 7503
Requirements for use, 5218, 7505
USS_Drive_Control, 5223, 7510
USS_Port_Scan , 5220, 7507
USS_Read_Param, 5225, 7512
USS_Write_Param, 5227, 7514
Library management
Displaying cross references of an instance, 427
Displaying instances of type versions, 427
Displaying relations between library objects, 428
Filtering, 8770
Filtering by non-released types, 426
Opening, 425, 8769
Overview, 423, 8767
Library object, 8762, 8763
Inserting, 8779
Library view
Exit, 423
Opening, 423
Overview, 420, 8763
License
Defective, 89, 102
Defective license, 89, 103
Starting without a valid license, 99
Starting without valid license, 87
License key, 88, 101
Handling license keys, 88
Working with license keys, 101
License Manager Panel Plugin, 100
Licenses, 84
Life of certificates, 785
Lifebeat failures, 10757
LIMALARM
Input parameters, 6972
Instruction, 6970
Output parameters, 6972
LIMIT, 3283, 3580, 3780, 4167, 4356, 5452, 5665, 5815, 6158, 6292
Limit and enable password legitimation, 3382, 3682, 3842, 4272, 4427
Limit monitoring, 9091
Limit range
Tag, 8125
Limit value, 2047, 3283, 3580, 4167, 6158, 8158
Tag, 8126
Limit value check, 8918, 8920
Limit value exceeded
Analog alarm, 8126
LIMITER
Input parameters, 6974
Instruction, 6972
Output parameters, 6974
Limits for dynamics, 10248, 10285, 10301
Line, 8089
Design, 7995

Line end, 7995, 8089
Line start, 8089
Line break, 991, 2038
Line comment, 2676
Line end
Line, 8089
Line start
Line, 8089
Linear axis
Easy Motion Control, 10769
Linear axis (CPU 31xC), 1715, 1723
Linear programming, 2168
LinearScaling, 8311
Link folder
External graphic, 8010
Link Layer Discovery Protocol (LLDP), 1873
Linked objects
Copying, 8866, 8867
Linking, 8823
Linking HMI tags, 179
Linking racks, 876, 878
LINT, 2249, 2301, 2346
LINT_TO_, 2346
LLDP, 742, 743, 1461
LLDP (Link Layer Discovery Protocol), 1461, 8442, 8491, 8544
LLDP (see Link Layer Discovery Protocol), 1873
LMNGEN_C
In/out parameters, 6979
Input parameters, 6978
Instruction, 6974
Output parameters, 6978
Static variables, 6979
LMNGEN_S
In/out parameters, 6986
Input parameters, 6985
Instruction, 6979
Output parameters, 6985
Static variables, 6986
LN, 3288, 3586, 3982, 4171, 4361, 5457, 5669, 5986, 6162, 6296
Load
Blocks in the device, 2922
Extended download to device, 74
HMI device, 8892
In RUN operating state, 2922
Loading block changes without reinitialization, 2914, 2916, 2918, 2919, 2920
Loading blocks to device, 2924
SIMATIC PC station, 74
Load gear, 10222
Load memory, 1572, 1655, 1659, 1839, 2957
Displaying, 2961
Load value, 9100, 9125, 9201
directly, 9100
in preparation, 9100
Load window layout
Loading additional window layouts, 305
Loading via quick access, 305
Loading, 3991, 3993, 3994, 3995, 3997, 3998, 3999, 4062, 4063, 4064, 4065, 5994, 5996, 5997, 5998, 6000, 6001, 6002, 6063, 6064, 6065
Axis data block, 10809
Blocks to the device, 2906
Configurations with PROFIBUS, 1558
Configurations with PROFINET, 1559
Configurations with web server, 1558
Counter value, 3947, 3949, 5951, 5953
Device configurations, 1556
Distributed I/O, 1557
Downloading project data to the device, 357
Downloading to a memory card, 359
from a device, 39, 47, 48, 360
General information, 355
Going online, 1563
GSDML, 1560
HMI devices, 1562
I-device, 1560
IE/PB-Link, 1561
in RUN operating mode, 2906
Isochronous applications, 1553, 1559
Loading block changes without reinitialization, 2917
Module comments, 41
Placeholder, 1560
Shared Devices, 1562
Subnets, 1557
Timer value, 3929, 3931, 5934, 5936
to PG/PC, 1554
to the device, 39, 1553
Uploading blocks from a memory card, 2913
Loading projects
with connected HMI device, 8883
Loading with S7 routing, 8884
Local access protection, 1889
Local assignment
Of a function key, 8056, 8060
Local data bit (L), 1572
Local data byte (LB), 1572
Local data double word (LD), 1572
Local data stack, 1662
Local data word (LW), 1572
Local error handling, 2823
Error information, 2823

Instructions, 2823
Priorities, 2824
Local logging, 798
Local tag, 2544
Local time, 1586, 1860
Calculate local time from base time with BT_LT, 6424
Calculating with SET_TIMEZONE, 4513
Reading and calculating with LOC_TIME, 6423
Reading out with RD_LOC_T, 4510
Set time-of-day interrupt with S_LTINT, 6427
Writing with WR_LOC_T, 4511
Location identifier, 1672
Log
Language switching, 8850
Runtime language, 8850
Sending data with AR_SEND, 6669
Log file
Modem alarms , 10977
Log language, 8850
Log table
Event log, 1198
LOG_GEO, 4817
LOG2GEO, 4810
LOG2MOD, 4811
Logarithm, 3288, 3586, 3982, 4171, 4361, 5457, 5669, 5986, 6162, 6296
Logging, 797
CP x43-1 Adv., 844
SCALANCE S, 821, 823
Logic analyzer function, 10879
Logic operation
AD, 4081, 6080
AND, 3399, 3699, 5505, 5717
AW, 4075, 6074
EXCLUSIVE OR, 3402, 3703, 5508, 5720
OD, 4083, 6082
OR, 3401, 3701, 5507, 5719
OW, 4077, 6076
XOD , 4085, 6084
XOW, 4079, 6078
Logic path
Deleting, 2649
Inserting, 2649
Use, 2648
Logical address
Determining a module with GADR_LGC, 4820, 6788
Determining a module with RD_LGADR, 4818, 6787
Determining associated slot with LGC_GADR, 4821, 6790
Determining associated slot with LOG_GEO, 4817, 6784
Logoff, 8287
Users, 8938
Logoff time, 8935
Changing, 8253
Changing in runtime, 8261
Logon, 907, 8292
Logon failed, 8263
Reporting, 8266
User, 8262
Logon dialog
Configuring access protection, 8265
Logon for web server, 944
Logon web server, 944
Long word, 2243
Loop, 4057, 4258, 4262, 4265, 4266, 4267, 4268, 6058, 6190, 6194, 6197, 6198, 6199, 6200
LOOP, 4057, 6058
Loop traverse, 10491, 10588
Data in the channel DB, 10492, 10589
Requirements, 10491, 10588
Sequence , 10492, 10588
Loop-in-alarm
Configured, 8167
Loop-In-Alarm, 8360
Loss of
Synchronization, 10584
Loss of synchronization, 9366, 9442
Low free storage space, 8363
Low limit, 9140, 9258, 9316
LP_SCHED
Input parameters, 6943
LP_ZONE
Instruction, 7298
LREAL, 2253, 2305, 2352, 2385, 2410
LREAL_TO_, 2352, 2410
LSAP configuration, 1993
LTIME, 2257, 2308, 2357
LTIME_TO_, 2357
LTOD, 2259, 2316, 2365
LTOD_TO_, 2365
LWORD, 2243, 2293, 2329
LWORD_TO_, 2329

M

MAC packet filter rules, 759
MAC rule sets, 747
MAC services, 753
Machine data, 10463, 10553
changing, 10463

Changing, 10554
Data in the channel DB, 10464, 10554
Drive, 10499, 10596
Enable input, 9405, 9481
Enabling, 9354, 10463, 10554
Reading, 9356, 10464, 10554
write, 10463
Writing, 9354, 10553
Main, 1614
Main (S7-1500), 1891
Main count direction, 9086, 9119, 9171, 9234, 9294
Main mode, 787
managing
Graphic, 8009
Managing
User group, 8253
Users, 8252
Users in runtime, 8260
Manual fragment, 930, 966
Manual hot restart, 1674
Manuals, 8397
Master, 1083
Master Control Relay, 5529, 5532, 5534, 5535, 5739, 5742, 5744, 5745, 6111, 6113, 6115, 6116
Master control relay (MCR)
Formal parameter access, 2472
Notes on use, 2472
Parameter transfer, 2472
Master copy
Library, 8765
Math functions
CALCULATE, 2590, 2632
MAX, 3281, 3578, 3778, 4165, 4355, 5451, 5663, 5813, 6157, 6290
Max. frequency, 10753
MAX_LEN, 4530
Maximum, 3281, 3578, 3778, 4165, 4355, 5451, 5663, 5813, 6157, 6290
Maximum configuration of an S7-1500, 868
Maximum cycle time, 1569, 1590, 1591, 1836, 1869, 3387, 3687, 5498, 5710
Maximum length
Tag, 75
Maximum load memory available
Displaying, 2961
Maximum number of characters, 992
MB, 2477
MB_CLIENT, 5323, 5341
MB_COMM_LOAD, 5307
MB_MASTER, 5310
MB_SERVER, 5331, 5352
MB_SLAVE, 5318
MC Interpolator, 10251
MC Servo, 10251
MC_ChangeDynamic
Instruction, 4851
Parameter, 4851
Parameter, 4854
MC_CommandTable
Instruction, 4849
Parameter, 4850
MC_Control, 7049
Call, 7050
Control mode, 7049
Description, 7049
Follow-up mode, 7049
Manual mode, 7050
Operating principle, 7050
Parameter, 7050
MC_GearIn, 7051
Call, 7056
Description, 7051
Operating principle, 7051
Parameter, 7056
MC_Halt, 4912, 4915
Function chart, 4834
Instruction, 4832
Parameter, 4866
Parameters, 4833
MC_Home, 4890, 7043
Call, 7046
Description, 7043
Instruction, 4829
Operating principle, 7043
Parameter, 4830, 7047
Reference point approach, 7043
Set reference point, 7044
MC_Init, 7034
Call, 7034
Description, 7034
Operating principle, 7034
MC_MoveAbsolute, 4908, 4911, 7035
Parameter, 4836
Call, 7037
Description, 7035
Function chart, 4837
Instruction, 4835
Operating principle, 7035
Parameter, 7037
MC_MoveJog, 4894, 4898, 7040
Call, 7042
Description, 7040
Function chart, 4848
Instruction, 4846

Operating principle, 7040
Parameter, 4847, 7042
MC_MoveRelative, 4904, 4907, 7038
Call, 7039
Description, 7038
Function chart, 4841
Instruction, 4838
Operating principle, 7038
Parameter, 4839, 7039
MC_MoveVelocity, 4899, 4903
Function chart, 4845
Instruction, 4842
MC_Power, 4885, 4889
Function chart, 4826
Instruction, 4823
Parameters, 4823
MC_Reset, 4916
MC_Simulation, 7050
Call, 7051
Description, 7050
Operating principle, 7051
MC_StopMotion, 7047
Call, 7048
Description, 7047
Operating principle, 7047
Parameter, 7049
Stopping a motion, 7048
MCAT, 3430, 3731, 3877, 4312, 4469, 5559, 5769, 5874, 6246, 6374
MC-Interpolator, 1906
MC-Interpolator OB, 1906
MCR, 5529, 5532, 5534, 5535, 5739, 5742, 5744, 5745, 6111, 6113, 6115, 6116
MCRA, 5534, 5744, 6115
MCRD, 5535, 5745, 6116
MC-Servo, 1905
Assigning parameters, 1921
MC-Servo OB, 1905
MD, 2477
MD5, 782, 788
MDM, 329
Meaning of the columns in the force table, 3023
Measure
Frequency, 9093
Rotary speed, 9094
Measure program runtime, 3397, 3697, 3858, 4287
Measure times
between two edges, 9130
Measuring
Comparison value, 9060
Period, 9096
Measuring functions, 9008
Measuring range, 9009, 9012
Measuring range violation (overrange), 9865, 9914, 9947
Measuring range violation (underrange), 9865, 9914, 9947
Measuring system
Selecting, 9384, 9460
Measuring time, 9088
Measuring times, 9130
Measuring transducer fault, 9865, 9914, 9947
Measuring window (GATE), 2035
Media redundancy (MRP), 1512, 1514, 1516, 1518
Memory, 1839, 1842
Command area, 2160
compress, 5497
Compress, 5709, 6329
Compressing, 2160, 5833, 6204
Copying from RAM to ROM, 2160
Memory area
Distribution, 1655
Load memory, 1572
of the inputs and outputs, 1742
PI and PQ, 1743
Reading the module with the system status list, 6695
Retentive memory areas, 1574
Work memory, 1572
Memory areas
Basis for the calculation, 1658
Load memory, 1655
System memory, 1655
System memory, reading with the system status list, 6696
Work memory, 1656
Memory bit (M), 1572
Memory byte (MB), 1572
Memory card, 33, 1570, 1653, 1837, 1839, (See Memory Card), (See Memory Card)
Accessing, 414
Adding a card reader, 413
Deleting data logs with S7-1500, 38
Formatting, 2113
Introduction, 412
Removal/insertion, 38
Showing properties, 414
Memory cards, 34
Memory concept of the S7-400, 1657
Memory double word (MD), 1572
Memory requirements
Recipe, 8970
Memory reserve, 49, 1845
Memory reset, 1570, 1651, 1837, 1842

Making, 2108
Memory size, 1659
Memory space model, 2741, 2760
Memory word (MW), 1572
Menu command
Simple recipe view, 8234, 8954
Message
Sending, 1605
Specifying the end, 1606
Specifying the start, 1605
Message frame header
Format of a RK512 command message frame, 1062
Message frame structure, 1087
Message frame structure , 10758
Message frame type
Entering, 10758
Message length from message, 992
Message texts, 1073
Character conversion table, 1073
Control characters, 1073
Control characters table, 1073
Page layout, 1073
Message timeout, 992
MIB, 742
Micro, 8709
Micro memory card, 1651
MicroLogix, 8639, 8661
MID, 4544
Migrated project
Compiling, 146
Migrating PLC programs to S7-1500, 190, 192, 198, 200, 206, 211, 228, 229, 231
Migrating projects
Procedure, 122
Requirements, 121
Migration, 33, 132, 135, 136, 137, 190, 192, 198, 200, 206, 211, 228, 229, 231
Adaptations beforehand, 147
Alarm groups, 152
Allen-Bradley DF1 data types, 166
Allen-Bradley Ethernet IP data types, 166
Basics, 143
Data types Modicon Modbus, 167
Data types Modicon Modbus TCP/IP, 168
Data types Omron Hostlink/Multilink, 168
Data types SIMATIC S7 200, 169
Data types SIMATIC S7 300/400, 169
Displaying a log file, 123
Displaying history, 123
Font settings, 150
Global library, 157
Including the hardware configuration, 118
Integrated project, 172
Introduction, 142
Introduction to migration, 117
Migrating an integrated project, 161, 172
Migrating PLC programs to S7-1500, 180, 182, 184, 202, 204, 205, 213, 214, 215, 216, 218, 219, 221, 222, 223, 224, 226, 233, 235, 236
Migrating projects, 145, 163
Mitsubishi FX data types, 167
of external tags, 165
of languages, 155
of project text, 155
Project library, 157
Recipe data, 158
Runtime data, 158
Supported HMI devices, 147
Supported products, 117
Tags, 151
Text, 155
The migration process, 117
User administration, 158
WinCC V7.0 SP3, 61
Migration of IP access protection lists when activating Security, 44
Migration of projects with Ethernet CPs and Security, 43
Migration of STEP 7 projects
Alarms, 132
CBA for IO controllers, 129
Device names of PROFINET IO devices, 128
GSD and GSDML devices, 130
I slaves and I-devices, 129
Introduction, 124
Isochronous mode with PROFINET IO, 129
Media redundancy, 129
Migratable types of connection, 130
Multiproject, 131
Names of modules and racks, 128
PROFIBUS isochronous mode, 130
Requirements, 125
Requirements for the project, 125
Scope of migration, 124
Shared devices, 129
System error messages, 132
Transfer areas, 129
Update groups PROFINET IO, 129
Migration tool, 144
Creating a migration file, 121
Distribution and sources, 114
Including the hardware configuration, 119
Removing, 115

System requirements, 114
Using the migration tool, 118
MIN, 3279, 3576, 3776, 4163, 4353, 5449, 5661, 5812, 6156, 6289
Minimum, 3279, 3576, 3776, 4163, 4353, 5449, 5661, 5812, 6156, 6289
Minimum cycle time, 1569, 1590, 1836, 1869
Minimum edge distance, 10613
Minimum edge-to-edge distance, 9395, 9471
Minimum pulse width, 9125, 9201
Mirroring, 1271, 1273
Missing pulses incremental encoder, 9402, 9477
Missing pulses of incremental encoder, 10523, 10620
Missing supply voltage, 2047, 2055
Mitsubishi, 8630
Address, 8683
Analog alarm, 8697
Basic Panel, 8425
Communication drivers, 8672
Connection interruption, 70
FX, 8672
TCP/IP, 8672
Mitsubishi FX
Configuring a connection, 8685
Connection, 8685, 8690
Connection parameters, 8687
CPU type, 8692
Data type, 8691
Migrating data types, 167
Mitsubishi MC TCP/IP, 8677, 8689
Mitsubishi MC TCP/IP
Configuring a connection, 8673
Connection, 8673, 8676
Connection parameters, 8674
CPU type, 8682
Data type, 8681
Mixed mode, 789
MOD, 2682, 3273, 3569, 3973, 5445, 5657, 5976
MODB_341, 1031
MODB_441, 1031
Modbus
End of frame, 1003
Exception code, 1003
Modbus_Comm_Load, 5246, 7533
Modbus_Slave, 5249, 5254, 7536, 7541
RS232 signals, 987
Modbus communication, 1001
as a slave with MB_SLAVE, 5318
as Modbus master with MB_MASTER, 5310
Configuring a port with MB_COMM_LOAD, 5307
Modbus instructions, 1013
Modbus master, 7653
Modbus_Comm_Load, 978, 1013, 5246, 7533
Modbus_Master, 978, 1013
Modbus_Slave, 978, 1013, 5249, 5254, 7536, 7541
Mode, 8100
Graphic I/O field, 8084
I/O field, 8080
Mode selector switch, 2153
Modem
Local, 10930
Problem, 10973
Remote, 10931
Modem alarms
Log file, 10977
Modem connection cannot be established, 10977
Modem connection is closed, 10977
Modem connection is interrupted, 10976
Causes and solutions, 10976
Modems without plug-and-play, 10931
Modicon
Approved communication with Modbus RTU, 8717
Connection, 8716
Connection cable, 8716
Restrictions with Modbus RTU, 8717
Modicon M340, 8709
Modicon Modbus, 8630
Basic Panel, 8425
Communication drivers, 8700
Migrating data types, 167
RTU, 8700
TCP, 8700
Modicon Modbus connection
Data types, 8726
Modicon Modbus RTU, 8722
Configuring a connection, 8712
Connection, 8712
Connection parameters, 8714
Data type, 8721
Modicon Modbus TCP, 8709
Configuring a connection, 8700
Connection, 8700, 8705
Connection parameters, 8702
Data type, 8708
Modicon MODBUS TCP/IP
Change word order, 8704
Modicon Modbus TCP/IP
Migrating data types, 168
Modify
High-speed counters, 4918
Modify value
Permitted operands, 3002

Modifying mode, 3009
Module
Module status information using the system status list, 6718
Addressing, 880, 881, 937
Assigning parameters with PARM_MOD, 6596
Asynchronously reading data record with RD_DPARA, 4685, 6595
Communication properties, 2150
Communication Status, 803
Configuring and assigning, 37
Copying, 511
Deleting, 510
Determine diagnostics status, 2100
Determining logical address of the slot with LOG_GEO, 4817, 6784
Determining logical address with GADR_LGC, 4820, 6788
Determining logical addresses with RD_LGADR, 4818, 6787
Determining slot of a logical address with LGC_GADR, 4821, 6790
Determining start address with GEO_LOG, 4816, 6783
Diagnostic data, 6631
Diagnostics information, system status list, 6736
Identification via system status list, 6692
Inserting, 38, 509
Module class, 6690
Moving, 512
Reading data record with RD_DPAR, 4683, 6593
Removing, 38
Replacing, 513
Select, 500
Time of day on a module, 2108
Module addressing, 1577, 1670
Module arrangement, 497
Module comments, 41
Module replacement, 176
Module rights, 741
Module start address, 1846, (see Module start address)
Module titles, 499
Module version
Updating, 516
Module-internal shared input (MSI), 1536
Module-internal shared output (MSO), 1536
Modules
Configuration overview, 864
Parameters in the user program, 876
Properties of the CPUs, 875
ModuleStates, 4756
Modulo, 10215
Software limit switch, 10247
Modulo axes, 10215
Momentum, 8709, 8722
Monitor, 1459, 8440, 8489, 8542
Monitoring
"Monitor all" command, 3011, 3036
"Monitor now" command, 3037
"Monitor once and now" command, 3012
Standstill velocity, 10467, 10558
Target range, 10467, 10558
Monitoring 5 V encoder signals, 9106
Monitoring and modifying modes, 3009
Monitoring functions, 10467, 10558
Jog, 10475
Jogging, 10566
Monitoring mode, 3009
Monitoring time, 10467, 10468, 10507, 10558, 10559, 10604
Monitoring window, 2064
Mono-master system, 1343, 1415
Motion Control, 10248
Execution system, 10251
External encoder, 10214
Linear axis, 10214
Positioning axis, 10212
Principle of operation, 10208
Rotary axis, 10214
Speed-controlled axis, 10212
Motion Control control structure, 10241
Motion Control CPU S7-1200
Guidelines, 10018
Motion Control instructions
Programming, 10319
Motion Control jobs
Ending Motion Control jobs, 10335
Starting Motion Control jobs, 10324
Tracking running jobs, 10326
Move
Block, 3318, 3347, 3616, 3646, 3793, 3818, 4187, 4230, 4376, 4401, 5470, 5683, 5817, 6171, 6308
Block uninterruptible, 3324, 3350, 3623, 3649, 3800, 3821, 4192, 4233, 4382, 4404, 5474, 5686, 5820, 6174, 6312
Character string in array with Ctrl_TO_Chars, 4527
Screen, 7968
Value, 3301, 3600, 3782, 5468, 5681
MOVE, 252, 3301, 3600, 3782, 5468, 5681
Move block, 3320, 3618, 3795, 4189, 4377
Move view
Keyboard operation, 493

Overview navigation, 478, 481, 483
MOVE_BLK, 3318, 3616, 3793, 4187, 4376
MOVE_BLK_VARIANT, 3320, 3618, 3795, 4189, 4377
MoveVelocity
Parameter, 4843
Moving
Hardware components, 512
Moving the view
Overview navigation, 8384, 8390, 8392
MPI, 8398
Address assignment, 881
Addressing, 8560
Connection, 8582
HMI connection, 8554
Network, 8400
Network architecture, 8400
Parameters, 8556, 8558, 8559
S7 200, 8589, 8598
S7 300/400, 8553, 8554
MPI address
S7 300, 8560
S7 400, 8561
MPI communication
S7 300/400, 8553
MPI interface, 2151
MPI parameters
S7 300/400, 8556, 8558, 8559
MPI/DP interface, 2151
MRP (Media Redundancy Protocol), 1512, 1514
MRP domain, 1516, 1518, 2095
MRP domain properties, 2095
MSI, 1847
MSI (module-internal shared input), 1848
MSI submodule, 1848
MSI/MSO, 1536
MSK_FLT, 4715
MSO, 1847
MSTP, 1285
MUL, 3270, 3566, 5443, 5655
Multicast, 622, 664, 769, 1296
With UDP, 650, 651
Multicomputing interrupt
Triggering with MP_ALM, 6629
Multi-instance
Declaring, 2557
Definition, 2183
Multi-instance FB
Easy Motion Control, 10812
Multi-key operation, 78
Multi-master systems, 1343
Multiple instance, 2213
Correcting the call type, 2590, 2631, 2675
Multiple selection, 500, 7988
Multiple Spanning Tree, 1285
Multiple use IO system, 1540, 1543, 1546, 1549
Multiplex, 3410, 3710, 4293, 4447, 6216
Multiplex/synchronous mode, 2069
Multiplexing, 8132
Address multiplexing, 8136
Address multiplexing Allen-Bradley Ethernet IP, 8643
with absolute addresses, 8136
with symbolic addresses, 8136
Multiply, 3270, 3566, 3962, 3967, 3976, 5443, 5655, 5966, 5970, 5980
Multipoint, 1082
Multi-point connection
Allen-Bradley DF1, 8654, 8655
MUX, 3410, 3710, 4293, 4447, 6216
MW, 2477
MX, 2477
My Documentation Manager, 329

N

N, 3168, 3171, 3463, 3465, 5384, 5595
N_TRIG, 3173, 3468, 5387, 5598
NAK, 996
Name
Array element, 67
Names of alarm classes
Change through migration, 153
NAMUR encoder, 9167
NAT/NAPT
Routing, 768
NCM connection
Connection details, 2165
NE_ElemType, 3259, 3555
NE_Type, 3257, 3553
NEG, 3274, 3570, 4351, 5447, 5659, 6286
Negate, 3274, 3570, 4351, 5447, 5659, 6286
Negating, 4019, 4020, 4021, 6021, 6022, 6024
NEGD, 4020, 6022
NEGI, 4019, 6021
Negotiation, 1246
NEGR, 4021, 6024
Nesting depth, 2181
Net volume of data, 1932
Network, 45, 8373
Closing, 2575, 2617, 2659, 2806
Copying, 2574, 2616, 2658, 2805
Deleting, 2574, 2617, 2658, 2806
Entering a comment, 2577, 2620, 2661

Ethernet, 8399
Insert title, 2576, 2619, 2660, 2808
Inserting, 2572, 2574, 2615, 2616, 2656, 2658, 2804, 2805
MPI, 8400
Navigating blocks, 2578, 2621, 2662, 2809
Opening, 2575, 2617, 2659, 2806
PPI, 8401
PROFIBUS, 8399
PROFINET, 8399
Selecting, 2573, 2616, 2657, 2804
Using, 2572, 2615, 2656, 2803
Network access
Opening the properties, 10863
Network adapter, 843
Network architecture
PPI, 8401
Network comments
Hiding, 2532
Showing, 2532
Network drive, 33
Network editor
Components, 476, 8381
Function, 475, 8381
Hardware catalog, 489, 8395
Inspector window, 8393
Inspector window , 487
Network ID, 829
Network Load Balancing Manager (NLB), 1986
Network overview
Basic functions, 523
Basic functions for editing the network overview table, 523
Network Syslog, 798
Network view, 8402
Adding device, 505
Hardware and network editor, 477, 8383
Meaning, 613
Networking
Communication partners, 8402
Devices, 8402
Networking devices
Basics of configuring networks, 519
Editing interface parameters, 525
Editing network parameters, 525
Networking several interfaces at the same time, 521
Networking with an existing subnet, 522
Networking without an existing subnet, 520
Networks within a project, 519
Requirements, 524
Types of communication, 519
Networking in the device view
Networking options, 523
Procedure, 524
New data types, 245
Nodes with an unknown IP address, 790
Non-integrated
Connection, 8407
Non-licensed mode
Engineering System, 99
ES, 87
HMI devices, 100
NONLIN
Input parameters, 6989
Instruction, 6986
Output parameters, 6989
NOP 0, 4111, 6119
NOP 1, 4112, 6120
NORM
Input parameters, 6991
Instruction, 6990
Output parameters, 6991
NORM_X, 3365, 3664, 3829, 4247, 4419
Normalize, 3365, 3664, 3829, 4247, 4419
Normally closed contact, 3157, 5375
Normally open contact, 3156, 5374
NOT, 2686, 2687, 3158, 3920, 4441, 5376, 5925, 6339
Not equal, 3262, 3558
NOT_NULL, 3262, 3558
NOT_OK, 3255, 3551
NTP, 1296
NTP (secure), 778
NTP server, 778
NULL, 2273
Null instruction, 4110, 4111, 4112, 6118, 6119, 6120

O

O, 3902, 3908, 5906, 5912, 10851
O(, 3911, 5916
OB
Event source, 1854
Events and OBs, 1578, 1853
Overview, 1578, 1853
OB 1, 1614, 1754
OB 1 PI (OB 1 process image), 1742
OB 10 to OB 17 (S7-300, S7-400), 1755
OB 100
OB 101 and OB 102 (S7-300, S7-400), 1807
OB 121 (S7-300, S7-400), 1811
OB 122 (S7-300, S7-400), 1814
OB 20 to OB 23 (S7-300, S7-400), 1759

OB 30 to OB 38 (S7-300, S7-400), 1760
OB 40 to OB 47 (S7-300, S7-400), 1762
OB 61 (S7-300, S7-400), 1776
OB 62 (S7-300, S7-400), 1776
OB 63 (S7-300, S7-400), 1776
OB 64 (S7-300, S7-400), 1776
OB 80, 1624
OB 80 (S7-300, S7-400), 1777
OB 81 (S7-300, S7-400), 1780
OB 82, 1625
    Diagnostic interrupt, 9862, 9911, 9943
OB 82 (S7-300, S7-400), 1783
OB 83 (S7-300, S7-400), 1786
OB 84 (S7-300, S7-400), 1790
OB 85, 1662
OB 85 (S7-300, S7-400), 1791
OB 86 (S7-300, S7-400), 1796
OB 87 (S7-300, S7-400), 1802
OB 88 (S7-300, S7-400), 1804
OB for manufacturer-specific interrupt, 1619
OB for manufacturer-specific interrupt (S7-1500), 1898
OB for manufacturer-specific interrupts (OB 57) (S7-300, S7-400), 1772
OB for profile-specific interrupt, 1619
OB for profile-specific interrupt (S7-1500), 1898
OB priority, 1854
OB start event, 1853
OB start information, 1890
OB 61 to OB 64, 1400
OB 83, 1626
object
    Group, 8014
    Paste, 8006
    Rotate, 7993
Object
    Arrange, 7982, 7991
    Assigning to a layer, 8070
    Availability for Basic Panels, 8075
    Creating a new OLE object, 7982
    Creating an OLE object from a file, 7982
    Deleting, 7984
    Design, 7995
    Designing the background color, 7997
    Designing the fill pattern, 7998
    Dynamization, 8042
    Dynamization of the appearance, 8047
    Dynamize direct movement, 8050
    editing within a group, 8018
    Evenly distributing, 7991
    Flip, 7982
    Flush alignment, 7990
    Grouping, 8014
    Inserting, 7981, 7982
    Inserting of the same type, 7982
    Inserting several times, 8006
    Multiplying, 8006
    outside the area, 7992
    Repositioning and resizing multiple objects, 7982, 8008
    Repositioning objects, 7985
    Resize, 7987
    Rotate, 7982, 7993
    Select multiple objects, 7988
    Selecting multiple objects, 7982
    Storing in a library, 8778
    Tab sequence, 7982
    Transparency, 7998
Object group
    Animation, 8052
    Editing an object within a group, 8018
    Removing an object, 8017
Object list
    Tag, 8116
Object names
    Uniqueness, 143
Object size
    Harmonizing, 7987
Obtaining user rights
    Logging on to the operating system with administrator privileges, 319
    With Windows user account control, 319
OD, 4083, 6082
Odd send clock, 1468
OF, 4260, 6192
Off delay, 3182, 3191, 3209, 3218, 3477, 3504, 3514, 3756, 3944, 4122, 4142, 4337, 5393, 5405, 5414, 5604, 5616, 5626, 5801, 5948, 6126, 6139, 6277
Offline, 727
OK, 3254, 3550
OLE object
    create from a file, 7982
    Recreate, 7982
    Save in Graphics, 8011
    Storing in the image browser , 8842
Omron, 8630
    Analog alarm, 8742
    Basic Panel, 8425
    Communication drivers, 8728
    Host Link, 8728
Omron Host Link, 8738
    Address, 8739
    Configuring a connection, 8729
    Connection, 8729, 8733

Connection cable, 8733
Connection parameters, 8731
Data type, 8737
Omron Hostlink/Multilink
Migrating data types, 168
ON, 3903, 5908
On delay, 3179, 3189, 3204, 3215, 3474, 3499, 3510, 3753, 3939, 4119, 4137, 4334, 5391, 5400, 5411, 5602, 5611, 5622, 5799, 5943, 6123, 6134, 6275
Retentive, 3207, 3217, 3502, 3512, 3942, 4139, 5403, 5412, 5614, 5624, 5946, 6136
On demand, 8129
ON(, 3912, 5917
Ones complement, 3404, 3704, 4017, 4018, 5510, 5722, 6019, 6020
Online, 31, 727
Hardware detection, 507
Online access, 10860
Online and Diagnostics view, 2079
Online connection, 10851
Multiple TIA Portal instances, 37
Online diagnostics, 799
Online displays
Orange color, 2088
Online mode, 10851
Canceling an online connection, 10855
Display of the online mode, 10853
Displaying accessible devices on an interface, 10849
Displaying all accessible devices as a list, 10849
Go online, 10854
Standby or hibernation of the programming device / PC, 10851
Online operation, 32, 37
Online Tools, 2079, 2089
On-the-fly measurement, 10761
OP, 5495
OP 73
Baud rate on PROFIBUS, 8475, 8529
OP 77A
Baud rate on PROFIBUS, 8475, 8529
OP 73
Baud rate on PROFIBUS, 8587, 8589, 8596, 8598, 8599
OP 77A
Baud rate on PROFIBUS, 8587, 8589, 8596, 8598, 8599
OP73
Loading a project, 74
OP77A
Loading a project, 74
OPC
Basic Panel, 8425
OPC configuration, 1993
OPC properties, 635, 641, 646
OPC server
Central communication service provider, 1156
Slot rules, 1157
Using as SMTP server, 666
OPC server - CPU messages, 1988
OPC server functions
Data record functions, 1994
For DP master class 2, 1994
OPC UA client, 1985
OPC UA redundancy, 1985
OPC UA server, 1985
Open
Global library, 8772
Open a project, 334
Compatibility mode V12 SP1, 333
Open log file
Packet filter events, 805, 807
Security events, 805, 807
System events, 804, 807
Open User Communication, 50, 599, 7410
Changing parameter values, 695
Sending data with TSEND, 7436
Ability to read back, 695
Changing IP configuration parameters with IP_CONF, 7449
Changing IP configuration parameters with T_CONFIG, 5122
Configuring connections with TCP and ISO on TCP, 7414
Configuring the local communications access point with UDP, 7419
Connection configuration, 676, 677
Connection description, 688, 691, 692
Connection establishment, 675
Connection parameters, 678
Creating a connection, 682
Deleting an interconnection, 685
Establishing a communication connection with TCON, 5083, 5086
Establishing a connection and reading data with TRCV_C, 5061, 5066
Establishing a connection and sending data with TSEND_C, 5050, 5054
Form of the address information for the remote partner with UDP, 7421
Functional description of instructions, 7411
FW_IOT, 7459
FW_TCP, 7455

General, 675
Instructions, 675
Port numbers, 694
Protocols used, 686
Receiving data via UDP with TURCV, 5112, 7446
Receiving data with TRCV, 5099, 5103, 7439
Sending data via UDP with TUSEND, 5109, 7443
Sending data with TSEND, 5093, 5096
Starting connection parameter assignment, 681
TCON, 7431
TCON_IP_RFC, 693
TCON_IP_v4, 691
TCON_Param, 689
Terminating a communications connection with TDISCON, 5091
Terminating a local communication access point with TDISCON, 7434
TSAP, 696
via Industrial Ethernet, 599, 7410
Opening a force table, 3026
Opening a watch table, 2999
Opening phone books, 10922
Opening Siemens Industry Online Support, 491
Operand, 2197, 2203, 2204, 2205, 2206, 2208, 2209, 2210, 2211, 2213, 2214, 2216, 2218
Inserting, 2601, 2643
Pasting, 2667
Operand area, 1572, 1659
Displaying, 2152
Operating behavior
Adjusting and loading parameters, 1671
Default settings, 1671
Enabling system memory, 1607, 1883
Module parameters, 1671
Using clock memory, 1608, 1686, 1884
Using time-of-day functions, 1593, 1684, 1881
Using timers and counters, 1667
Operating mode, 843, 9048
Count periodically, 9085
Count continuously, 9083
Count once, 9084
Dosing, 9098
For testing GRAPH programs, 2981
Frequency measurement, 9093
HOLD, 1650
Incremental approach, 10482, 10573
Introduction, 1563, 1831
Jog, 10473
Jogging, 10564
Measuring, 9054
Overview, 1639
Period measurement, 9096
Priority, 1641
Reference point approach, 10476, 10567
Rotational speed measurement, 9094
RUN, 1569, 1650, 1836
Runtime events, 1825
STARTUP, 1565, 1833
STOP, 1569, 1641, 1836
Testing in automatic mode, 2985
Testing in manual mode, 2985
Testing in semiautomatic mode, 2985
Transitions, 1564, 1640, 1821, 1832
Operating mode transitions, 1014, 1153
Operating Modes of Motion Control, 10255
Operating object
Availability for Basic Panels, 8075
Operating principle
ABS_CTRL, 7125
ABS_CTRL_451, 7129
ABS_DIAG, 7127
ABS_DIAG_451, 7131
CAM_CTRL, 7157
CAM_CTRL_452, 7160
CAM_DIAG, 7158
CAM_DIAG_452, 7162
CNT_CTL1, 7098
CNT_CTL2, 7100
CNT_CTRL, 7096
CNT2_CTR, 7112
CNT2RDPN, 7111
CNT2WRPN, 7110
DIAG_INF, 7097
DIAG_RD, 7112
MC_Control, 7050
MC_GearIn, 7051
MC_Home, 7043
MC_Init, 7034
MC_MoveAbsolute, 7035
MC_MoveJog, 7040
MC_MoveRelative, 7038
MC_Simulation, 7051
MC_StopMotion, 7047
Operating range, 10452
Operating range limit
Jog, 10475
Jogging, 10566
Operating recipes
Creating a recipe data record, 8236
Creating recipe data records, 8236, 8237
Read recipe data record, 8237
Transfer data record, 8238
Operating system
Asian language setting, 8828

Setting to Western, 8828
Updating, 8897
Operating system , 2167
Operation
Alarm indicator, 8183
Button, 8931
Date/time field, 8923
Graphic IO field, 8928
Graphic view, 8927
IO field, 8925
simple alarm view, 8945
Simple alarm view, 8182
Simple recipe view, 8233, 8953
Simple user view, 8937
Switch, 8932
Symbolic IO field, 8934
Trend view, 8930
Operation in runtime
Multi-key operation, 78
Operation when expected and actual configurations do not match, 1357
Operator control
Alarm view, 8093
Trend view, 8930
Operator error, 9162, 9219
Operator input in the recipe view
Transferring data, 8513, 8571, 8609, 8756
OPN, 4059, 5707, 6060
OPNDI, 4060, 6061
OPNI, 5496, 5708
Optimized access, 2175, 2177
Optimized block access, 244
Optimizing the configuration
Allen-Bradley DF1, 8646, 8667, 8684, 8694, 8711, 8723, 8740
Option handling, 1361, 1364, 1429, 1432, (See Configuration control (ET 200SP)), (See Configuration control (ET 200MP))
Option handling (see Configuration control), 896
OR, 2453, 2687, 3401, 3447, 3448, 3701, 5507, 5583, 5584, 5719
Order
Job execution, 7143, 7145
Order number, 489, 1196
Order numbers, 974
Organization block (OB)
Activating time-of-day interrupt organization blocks from user program, 6607
Assigning event with ATTACH, 4691
Background OB (OB 90) (S7-300, S7-400), 1806
Canceling event assignment with DETACH, 4693
Cold restart OB (OB 102) (S7-300, S7-400), 1807
Communications error OB (OB 87) (S7-300, S7-400), 1802
Complete overview of OBs, 1752
CPU hardware fault OB (OB 84) (S7-300, S7-400), 1790
Creating, 2501
Cyclic interrupt OB, querying parameters with QRY_CINT, 4696
Cyclic interrupt OB, setting parameters with SET_CINT, 4694
Cyclic interrupt OBs (OB 30 to OB 38) (S7-300, S7-400), 1760
Cyclic program (OB 1) (S7-300, S7-400), 1754
Determining program runtime with OB_RT, 6739
Diagnostic interrupt OB (OB 82) (S7-300, S7-400), 1783
Displaying, 2152
exporting to an external source file, 2865
Function, 2170
Hardware interrupt OBs (OB 40 to OB 47) (S7-300, S7-400), 1762
Hot restart OB (OB 101) (S7-300, S7-400), 1807
I/O access error OB (OB 122) (S7-300, S7-400), 1814
Isochronous mode interrupt OB (OB 61 to OB 64) (S7-300, S7-400), 1776
OB for manufacturer-specific interrupts (OB 57) (S7-300, S7-400), 1772
Power supply error OB (OB 81) (S7-300, S7-400), 1780
Processing abort OB (OB 88) (S7-300, S7-400), 1804
Program execution error OB (OB 85) (S7-300, S7-400), 1791
Programming error OB (OB 121) (S7-300, S7-400), 1811
Pull/plug interrupt OB (OB 83) (S7-300, S7-400), 1786
Rack failure OB (OB 86) (S7-300, S7-400), 1796
Reading start information with RD_SINFO, 4729, 6679
Start information, 2170
Status interrupt OB (OB 55) (S7-300, S7-400), 1764
Time error OB (OB 80) (S7-300, S7-400), 1777
Time-delay interrupt OBs (OB 20 to OB 23) (S7-300, S7-400), 1759
Time-of-day interrupt OBs (OB 10 to OB 17) (S7-300, S7-400), 1755
Update interrupt OB (OB 56) (S7-300, S7-400), 1768

Warm restart OB (OB 100) (S7-300, S7-400), 1807
Organization blocks (OB)
Activating time-delay interrupt organization blocks from user program, 6611
Canceling time-delay interrupt organization blocks from user program, 6612
Querying time-delay interrupt organization blocks from user programs, 6613
OS, 2453, 2458, 2459, 2460, 2466
OSPF
Area range, 1319
Areas, 1187, 1318
Configuration, 1316
Interfaces, 1320
Link State Advertisement, 1187
OSPFv2 Interfaces, 1213
OSPFv2 LSDB (information), 1217
OSPFv2 neighbors, 1214
OSPFv2 virtual neighbors, 1216
Router, 1187
Router status, 1187
Virtual Links, 1322
Other PLCs
Addressing, 8631
Data types, 8631
Special features, 8631
OUT_RANGE, 3252, 3548
Outgoing, 8152, 8358
Output
Inserting, 2598, 2639
remove, 2599, 2640
Output (O), 1572
Output byte (QB), 1572
Output double word (QD), 1572
Output driver
Call, 7081
Error handling, 7081
Function, 7081
Initialization, 7081
Simulation mode, 7082
Output fields
Multiplex tag, 68
Output parameters
High_Speed_Counter, 4925
Output parameters of GRAPH function block, 2746
Output word (QW), 1572
OutputCPU314C, 7084
Call, 7084
Description, 7084
Parameter, 7084
OutputET200S2AO, 7083
Call, 7083
Description, 7083
Parameter, 7084
OutputIM174, 7087
Call, 7088
Configuration of IM 174 (hardware configuration), 7087
Description, 7087
Parameter, 7088
OutputIM178, 7082
Call, 7082
Description, 7082
Digital outputs, 7082
Parameter, 7082
OutputMM4_DP, 7086
Call, 7087
Description, 7086
Parameter, 7087
OutputSINAMICS, 7089
Call, 7092
Configuration of the SINAMICS in HWCN, 7089
Description, 7089
Parameter, 7092
OutputSM332, 7085
Call, 7085
Description, 7085
Parameter, 7085
OutputSM432, 7083
Call, 7083
Description, 7083
Parameter, 7083
OutputUniversal, 7085
Call, 7086
Description, 7085
Operating principle, 7085
Parameter, 7086
OV, 2453, 2458, 2459, 2460, 2462, 2466, 2470
Overflow, 2039, 2047, 2453, 2466, 8362
Overlay icon, 2086
Overlaying
Tag, 2218
Overlaying tags, 2556
Overload behavior, 1855
Overload range, 10753, 10754
OVERRIDE
Input parameters, 6993
Instruction, 6991
Output parameters, 6993
Overriding travel
Easy Motion Control, 10773, 10806
Overview
using the display formats in the force table, 3030

via the display formats in the watch table, 3004
Overview navigation, 478, 481, 483, 8384, 8390, 8392
Overview of the cross-reference list, 2964
Overview of TS Adapters that can be used, 10929
Overview window, 58, 300
Comparing objects, 302
Display all blocks, 303
Showing and hiding columns, 303
Sorting details view, 303
Overwriting the PROFINET device name, 1549
OW, 4077, 6076

P

P, 3167, 3169, 3461, 3464, 5383, 5594
P_PRINT, 1029
P_PRINT341, 1029
P_RCV, 1029
P_RCV_RK, 1029, 7653
P_RESET, 1029
P_SEND, 1029
P_SND_RK, 1029, 7653
P_TRIG, 3172, 3467, 5386, 5597
P3964_Config, 977, 1008
P3964_Config (protocol configuration), 5193, 7480
Packet filter log, 805, 807
Page layout
in message texts, 1073
PageDown, 8328
PageUp, 8327
Pane mode, 297
Panning, (See move view)
PARA_CLT
Input parameters, 6994
Output parameters, 6995
PARA_CTL
Instruction, 6993
Parameter
CNT_CTL1, 7098
CNT_CTRL, 7096
DIAG_INF, 7098
EncoderAbsSensorDP: , 7067
EncoderCPU314C, 7073
EncoderET200S1Count, 7069
EncoderET200S1SSI, 7065
EncoderFM350, 7071
EncoderFM450, 7062
EncoderIM174, 7076
EncoderSINAMICS, 7078
EncoderSM338, 7068
EncoderUniversal, 7080
ErrorID, 4926
For CPU, 1586
Hidden parameters, 2603, 2644
MC_Control, 7050
MC_GearIn, 7056
MC_Home, 7047
MC_MoveAbsolute, 7037
MC_MoveJog, 7042
MC_MoveRelative, 7039
MC_StopMotion, 7049
OutputCPU314C, 7084
OutputET200S2AO, 7084
OutputIM174, 7088
OutputIM178, 7082
OutputMM4_DP, 7087
OutputSINAMICS, 7092
OutputSM332, 7085
OutputSM432, 7083
OutputUniversal, 7086
Predefined - writing with WR_DPARM, 4689, 6601
SIMATIC LOGO!, 8621
Parameter (CPU), 1860
Parameter assignment, 2186
Technology module, 8979, 9053
Parameter assignment parameters
Hardware, 494
Parameter configuration
LENGH and BUFFER for Send_P2P, 5197, 7484
Parameter data, 1668
Parameter DB
Configuration, 7171
Content, 7172
Purpose, 7171
Parameter DB FM x51
Configuration, 7139
Content, 7139
Purpose, 7139
Parameter error
Easy Motion Control, 10821
Parameter errors
Easy Motion Control, 10827
Parameter passing
Activating via registers, 2654
Parameter transfer, 2185, 2187, 2189, 2192, 2193, 2194, 2195, 2196
Parameter types, 2279
Parameters, 7351
ABS_CTRL_451, 7130
ABS_DIAG_451, 7132
ABS_INIT, 7125
CAM_CTRL_452, 7161

CAM_DIAG_452, 7162
CAM_INIT, 7156
CAM_MSRM_452, 7163
CNT2_CTR, 7112
CNT2RDPN, 7111
CNT2WRPN , 7111
Connection, 8634, 8648, 8674, 8687, 8702, 8714, 8731
DIAG_RD, 7113
Dynamically - transferring with WR_PARM, 6600
Internal - Changing settings, 2761
MPI, 8556, 8558, 8559
PROFIBUS, 8446, 8448, 8449, 8494, 8496, 8497, 8548, 8549, 8551
S7 1200, 8526
S7 1500, 8472
S7 200, 8584, 8589, 8594, 8598, 8599
S7 basic communication, 7351
Part number, (see Order number)
Passive homing
with digital input, 10236
with zero mark, 10235
with zero mark and proximity switch, 10234
Password, 1889, 8935
Backing up, 8936
Changing, 8252
For CPU access protection, 1586, 1860
for function key, 8063
Hierarchy level, 8246
HMI connection, 8438, 8450, 8487, 8500
Password aging, 8247
Password complexity, 8247
Restoring, 8936
Password legitimation, 3382, 3682, 3842, 4272, 4427
Password list, 8935
Password protection, 944, 1586, 1609, 1610, 1690, 1860, 1884
Protection concept, 378
Revoking access rights, 379
Pasting
Adjusting screen size, 8865
Color, 8864
Font, 8864
Function key, 8865
Invalid object, 8864
Principle, 8863
PC CP, 726
PC station
Configuring offline, 1160
Configuring online, 1158
Connections, 595
OPC configuration, 1988, 1993
Operating mode, 1155
Plug component, 1157
Plugging submodules, 1157
Slot rules, 1157
Terms, 1155
XDB configuration, 2000
PDU structure with FETCH/WRITE, 1961
PE_CMD, 4634, 6546
PE_DS3_Write_ET200S, 4640, 6551
PE_End_RSP, 4671, 6581
PE_Error_RSP, 4668, 6578
PE_Get_Mode_RSP, 4673, 6583
PE_I_DEV, 4664, 6574
PE_List_Modes_RSP, 4672, 6582
PE_Measurement_List_RSP, 4678, 6588
PE_Measurement_Value_RSP, 4680, 6590
PE_Mode_ID, 7779, 7804
PE_PEM_Status_RSP, 4675, 6585
PE_Start_End, 4630, 6542
PE_Start_RSP, 4669, 6579
PE_WOL, 50, 4641, 6552
PEEK, 4209
Read memory address, 4209
PEEK_BOOL, 4211
Read memory bit, 4211
Perfect Forward Secrecy, 788
Performance, 243
Performance data, 2152
Performance features
Engineering system, 8959
HMI device, 8961, 8965
Period
Resolution, 9120, 9256, 9314
Period (CPU 31xC), 1705
Period measurement, 9008, 9096
Period resolution, 9120, 9256, 9314
Permanent connection, 1166
Permanent instruction
Inserting, 2809
Introduction, 2739
Permanent station, 1166
Permit access with PUT/GET communication from remote partners, 1613, 1888
Permit overwriting of device name, 1461
Permitted data types
Trends, 8468, 8627
Permitted operands for the force table, 3028
Permitted operands for the watch table, 3001
PG/PC interface
Go online, 10855
Modifiable MPI interface parameters, 10870

Modifiable PROFIBUS interface parameters, 10870
Phase shift (S7-1500), 1901
PID
Input parameters, 6998
Instruction, 6995
Operating principle, 6996
Output parameters, 6999
PID_3Step
In/out parameters, 4985
Input parameters, 4981, 5010
Instruction, 4974, 5004
Output parameters, 4983, 5011
Static tags, 5012
PID_Compact
Input parameters, 4936, 4958
In/out parameters, 4938
Instruction, 4955
Output parameters, 4937, 4959
Static tags, 4939, 4960
PID_CP
Output parameters, 6897
Description, 6880
In/out parameters, 6898
Input parameters, 6895
PID_ES
Output parameters, 6924
PID_FM
Block diagrams, 7195
In/out parameters, 7192
Input parameters, 7189
Instruction, 7185
Mode of operation, 7187
Output parameters, 7189
PID_FM_455
Block diagrams, 7276
In/out parameters, 7272
Input parameters, 7269
Instruction, 7266
Mode of operation, 7267
PID_PAR
Changeable parameters, 7215
Input parameters, 7214
Instruction, 7212
PID_PAR_455
Alterable parameters, 7294
Input parameters, 7294
Instruction, 7292
Output parameters, 7294
Pin assignment
6XV1440 - 2P for Mitsubishi PG protocol, 8690
Allen-Bradley, 8655
Allen-Bradley cable 1747-CP3, 8657
Allen-Bradley cable 1761-CBL-PM02, 8658
Allen-Bradley cable 1784-CP10, 8656
Front connector of the FM 350-1, 9109
Multipoint cable 1:MP/TP/PC, 8735
Multipoint cable 2:RS422, MP/TP/PC, 8736
Point-to-point cable 1,
Point-to-point cable 2, 8720
Point-to-point cable PP1 for Omron, 8736
Point-to-point cable PP2 for Omron, 8737
Ping, 1256
Ping command, 7734
PING command, 7735
PIP, (See process image partition), (See process image partition), (See process image partition)
Pip (data type), 892
Placeholder device, 1560
Plant designation, 1672
PLC
Connecting, 8636, 8676, 8690, 8705, 8733
MPI parameters, 8559
PROFIBUS parameters, 8449, 8497, 8551
PROFINET parameters, 8436, 8483, 8538
Reading recipe data record, 8957
Tag, 8132
Transferring recipe data record, 8958
PLC alarms with instructions
Acknowledgeable with ALARM_DQ, 6662
Always acknowledged with ALARM_D, 6660
Creating , 6635
Determining acknowledgement status with ALARM_SC, 6664
Disabling with DIS_MSG, 6673
Enabling with EN_MSG, 6671
Error behavior of instructions, 6652
Overview, 6653
Reading dynamically allocated system resources with READ_SI, 6675
Releasing dynamically allocated system resources with DEL_SI, 6677
Startup characteristics of instructions, 6652
System resources with ALARM_SQ and ALARM_S, 6655, 6657
With acknowledgement display with ALARM, 6649
With associated values for eight signals with ALARM_8P, 6643
Without acknowledgement display with NOTIFY, 6646
Without acknowledgement display with NOTIFY_8P, 6639

Without associated values for eight signals with ALARM_8, 6641
PLC data, 8784
PLC data type
Addressing, 2215
Comparing, 2897
Creating, 2856
Declaration in data blocks, 2836
Declaration in the block interface, 2555
Declaration table for PLC data types, 2855
Declaring, 2859
Declaring ARRAY, 2857
Declaring STRUCT, 2858
Definition, 2280, 2854
Deleting, 2856
Offline/offline comparison, 2872
PLC data types generated by the system in libraries, 49
Programming the structure, 2857
Tag properties, 2860, 2861
PLC data types, 45, 245
PLC migration
Instruction profile, 38
PLC tag
Importing and exporting, 2496, 2497, 2498
Automatically filling in cells, 2496
Comparing, 2895
Comparing a PLC tag table, 2872
Comparison, 2869
Copying, 2494
Declaring, 2481, 2482, 2485, 2487
Definition, 2203
Monitor, 2493
Offline/offline comparison, 2872
Permissible addresses and data types, 2479
PLC tag table, 2473, 2474, 2479, 2480, 2481
Properties, 2490, 2491
Reconnecting, 8124
Retentive behavior, 2483, 2484
Rules, 2479
Sorting rows, 2495
Updating comparison results, 374
PLC tags, 917, 918, 919, 952, 953, 955
PLC5, 8661
PLCSIM, 38, 47
PLUG, 1254
C-PLUG, (C-PLUG)
KEY-PLUG, (KEY-PLUG)
Plug component
PC station, 1157
Plugging submodules
PC station, 1157
PN/PN couplers
Grouping, 1550
Linking Ethernet subnets, 1550
PNIO_ALARM instruction
Formal parameter, 7776
PNIO_RECV, 7763
PNIO_RECV instruction
Formal parameter, 7764
PNIO_RW_REC instruction
Formal parameter, 7772
PNIO_SEND, 7756
PNIO_SEND instruction
Formal parameter, 7757
Pointer
ANY, 2275
POINTER, 2273
VARIANT, 2277
POINTER, 2273
Pointer name
Area pointer, 8419
point-to-point, 1185
Point-to-point, 1082
Point-to-point communication (PtP), 1599
Freeport protocol, 1599
Point-to-point connection, 976
Allen-Bradley DF1, 8653
Configuring communication parameters with PORT_CFG, 5277
Configuring receive parameters with RCV_CFG, 5281
Configuring serial transmission parameters with SEND_CFG, 5280
Deleting receive buffer with RCV_RST, 5291
Enable receiving with RCV_PTP, 5290
Initiating data transfer with SEND_PTP, 5288
Querying the signal state with SGN_GET, 5292
Setting output signals with SGN_SET, 5293
Point-to-point connection ((3964(R), CPU 31xC)), 1732
Point-to-point connection ((ASCII), CPU 31xC), 1725
Point-to-point connection ((RK 512), CPU 31xC)), 1737
Point-to-point connection (CPs, ET 200S 1SI), 1027
Point-to-point programming, 5179, 7467
POKE, 4213
Write memory address, 4213
POKE_BLK, 4216
Write memory area, 4216
POKE_BOOL, 4214
Write memory bit, 4214
POP, 4106, 6104
Port, 536

102 (S7 protocol - TCP), 750
123 (NTP), 770, 777
161 (SNMP), 770
20/21 (FTP), 750
443 (HTTPS), 770
4500 (IPsec), 770
500 (IPsec), 770
500 (ISAKMP - UDP), 796
514 (Syslog), 770
80 (HTTP), 750
8448 (security diagnostics), 856
Configuration, 1247
Interconnecting, 1455
Port configuration, 1246, 1249
Renaming, 721
Port configuration, 1249
Port diagnostics
SFP diagnostics, 1259
Port numbers, 694
Port options, 1460, 8441, 8490, 8543
Enable autonegation, 1459, 8440, 8489, 8542
Monitor, 1459, 8440, 8489, 8542
Transmission medium/duplex, 1458
Transmission rate / duplex, 8440, 8489, 8542
PORT_CFG, 5277
Port_Config, 977, 1008
Port_Config (port configuration), 5184, 7472
Portal view, 277
Position
Editing multiple objects, 8008
Of objects, 7985
Position data, 10494, 10592
Data in the channel DB, 10495, 10592
Process when reading, 10494, 10592
Position feedback (CPU 31xC), 1710, 1719
Position input
with motion control, 9017
with SSI absolute encoder, 8981
Position-based cams, 9340, 9341
Positioning (CPU 31xC), 1709
Positioning axis
Configuration, 10266
Tags, 10365
Positioning axis technology object, 10212
Configuring approach/homing direction, 10050
Configuring distance per motor revolution, 10036
Configuring end of reference point switch, 10049
Configuring maximum velocity / start/stop velocity, 10041
Configuring passive homing, 10048
Configuring the homing speed, 10051
Drive enable configuration, 10035
Drive ready configuration, 10036
PTO and HSC configuration, 10033
User unit configuration, 10035
Active homing, 10053
ActiveHoming tags, 10145
Actor tags, 10143
Add new object, 10031
Axis name configuration, 10033
Basic parameters, 10032
Changing the configuration parameters for dynamics in the user program, 10047
Changing the configuration parameters for homing in the user program, 10054
Commissioning overview, 10030
Configuration overview, 10030
Configuration window icons, 10032
Configuring acceleration, 10042
Configuring active homing, 10049
Configuring deceleration, 10042
Configuring end of reference point switch, 10051
Configuring home position offset, 10051
Configuring invert direction signal, 10036
Configuring jerk limiter, 10043
Configuring permit auto reverse at the hardware limit switch, 10050
Configuring pulses per motor revolution, 10036
Configuring ramp-down time, 10042
Configuring ramp-up time, 10042
Configuring reference point position, 10049, 10051
Configuring reference point switch input, 10049, 10050
Configuring smoothing time, 10043
Configuring the velocity limiting unit configuration, 10041
Diagnostics overview, 10030
Drive signal configuration, 10035
DynamicDefaults tags, 10150
DynamicLimits tags, 10149
Emergency stop deceleration configuration, 10044
ErrorBits tags, 10159
Extended parameters, 10032
General dynamics configuration, 10041
Hardware and software components, 10026
Hardware interface configuration, 10033
Homing tags, 10153
Mechanics configuration, 10036
Mechanics tag, 10148
Passive homing, 10051
PassiveHoming tags, 10146
Position tag, 10142

PositionLimitsHW tags, 10152
PositionLimitsSW tags, 10151
Response when jerk limiter is activated, 10046
StatusBits tags, 10155
StatusPositioning tags, 10154
Tools, 10029
Units tag, 10148
Updating the tags, 10160
Velocity tag, 10143
Positioning monitoring, 10242
Positioning operation
End, 10467, 10557
Possible cause of error
Transferring data, 8465, 8516, 8574, 8612, 8759
Power budget, 2053
Power segment, 868
Power supply error OB (OB 81) (S7-300, S7-400), 1780
Power supply modules, redundant, 866
PPI, 8398
Network, 8401
Network architecture, 8401
S7 200, 8599
Predefined alarm classes, 8153, 8154
Predefined firewall rules
CP 1543-1, 856
CP 1628, 847
CP x43-1 Adv., 845
SCALANCE S, 821, 823
Predefined styles, 8005
Premium, 8709
Preparing migration, 128
Prerequisites for establishing a VPN connection, 10958
Preshared keys, 786
Press, 8357
Press ESC twice, 8358
Press key, 8362
PressButton, 8324
Principle of operation
Direct data exchange, 1375
Interrupt generation on the I-slave, 1370
Print scope, 381
Printable contents, 381
Printer driver
BUSY signal, 1076
Examples, 1074
Format string, 1074
Handshake procedure, 1075
Message text output, 1075
Message texts, 1073
Software handshake, 1075
Variables, 1073
Printing
Changing the settings, 382
Creating a cover page, 387
Creating frames, 386
Documentation settings, 380
Elements in the library, 385
Elements in the project tree, 385
Non-printable contents, 381
Non-printable objects, 382
Print contents, 384
Printing device view, 484
Printing network view, 484
Process a cover page, 388
Process borders, 388
Specify layout, 383
Specify print layout, 383
Specifying the print area, 392
Structure of printout, 380
Use borders, 385
Use borders and cover pages, 380
Use cover page, 383
Printing a project, 397
Printing labels, 401
Determining the correction value for the print shift, 402
Printing of alarms
Configuring:printing of alarms, 8171
Priorities
Local error handling, 2824
Prioritized startup, 1505
Prioritized startup (PROFINET), 1509
Priority class, 1758, 1821
Process control phase, 8902
Process data area
I/O addressing, 1670
Module addressing, 1577, 1670
Module start address, 1577
Process image, 889, 891, 892, 1577, 1845
Basics, 1575, 1661
I/O access error, 1575, 1661
of inputs and outputs, 1742
Reading inputs with GETIO, 4560, 6463
Reading the process image area with GETIO_PART, 4562, 6465
Synchronizing the inputs with SYNC_PI, 4553, 6455
Synchronizing the outputs with SYNC_PO, 4555, 6457
Transferring the process image area with SETIO_PART, 4563, 6467
Updating, 1575, 1661

Updating inputs with UPDAT_PI, 4550, 6452
Updating the outputs with UPDAT_PO, 4552, 6454
Writing outputs with SETIO, 4561, 6463
Process image partition, 890, 892, 893, 1405
Process image partition "OB Servo PIP", 10253
Process interrupt
Triggering for DP master with DP_PRAL, 6515
Process mode, 1692
Processing abort OB (OB 88) (S7-300, S7-400), 1804
Product support, 8397
PROFIBUS, 39, 531, 1397, 8398, 8499, 8552
Connection with PROFINET, 1437
Different functionality, 1386
Direct data exchange, 1376
DPV0/DPV1, 1384
ET 200S in DPV1 mode, 1365, 1433
HMI connection, 8443, 8493, 8546
Network, 8399
OP 73, 8475, 8529
OP 77A, 8475, 8529
OP 73, 8587, 8589, 8596, 8598, 8599
OP 77A, 8587, 8589, 8596, 8598, 8599
Parameters, 8446, 8448, 8449, 8475, 8494, 8496, 8497, 8529, 8548, 8549, 8551, 8587, 8596
S7 1200, 8492
S7 1500, 8445
S7 300/400, 8443, 8545, 8546
S7-1200, 8493
Setting up a DP master system, 1379
Standard, 8499, 8552
Switch over DP interface, 1350
Universal, 8499, 8552
PROFIBUS cable configuration
Optical ring, 533
PROFIBUS communication
S7 1200, 8492
S7 1500, 8445
S7 300/400, 8545
PROFIBUS DP, 1353, 1416, 8399
Connection, 8580
PROFIBUS DPV1, 1683
PROFIBUS parameters
S7 1200, 8494, 8496, 8497
S7 1500, 8446, 8448, 8449
S7 300/400, 8548, 8549, 8551
PROFIBUS profile, 531, 8499, 8552
Different profiles on the same subnet, 531, 8499, 8552
Effect on the transmission rate, 531, 8499, 8552
Meaning of profiles, 532, 8499, 8553
PROFIdrive, 10216
PROFIdrive mode, 1995
PROFIenergy, 1520
Control PROFIenergy commands in the I-Device, 4664, 6574
Description, 4628, 6540
Generate answer to command at end of pause, 4671, 6581
Generate answer to command at start of pause, 4669, 6579
Generate list of supported measured values as answer, 4678, 6588
Generate negative answer to command, 4668, 6578
Generate PEM status as answer, 4675, 6585
Generate queried energy data as answer, 4673, 6583
Generate queried energy savings modes as answer, 4672, 6582
Generate queried measured values as answer, 4680, 6590
Generate supported PROFIenergy commands as answer, 4677, 6587
PE_CMD, 4634, 6546
PE_DS3_Write_ET200S, 4640, 6551
PE_End_RSP, 4671, 6581
PE_Error_RSP, 4668, 6578
PE_Get_Mode_RSP, 4673, 6583
PE_I_DEV, 4664, 6574
PE_Identify_RSP, 4677, 6587
PE_List_Modes_RSP, 4672, 6582
PE_Measurement_List_RSP, 4678, 6588
PE_Measurement_Value_RSP, 4680, 6590
PE_PEM_Status_RSP, 4675, 6585
PE_Start_End, 4630, 6542
PE_Start_RSP, 4669, 6579
Reading out status information, 4634, 6546
Set power module switching behavior, 4640, 6551
Start and exit energy-saving mode, 4630, 4634, 6546
Starting and stopping energy-saving mode, 6542
Wake on LAN, 4641, 6552
PROFIenergy controller, 7779
PROFIenergy device, 7779
Profile, (See bus profile)
Profiles, 1619
Profiles (S7-1500), 1898
PROFINET, 536, 834, 1468, 1472, 1476, 1498, 1516, 1518, 1873, 8398
Connection, 8579
Connection with PROFIBUS, 1437

Device names, 1465
Device number, 1466
Device replacement without exchangeable medium, 1461
Diagnostics, 1472
HMI connection, 8430, 8431, 8478, 8479, 8532, 8533
IP addresses, 1465
S7 1200, 8478
S7 1500, 8430
S7 300/400, 8532
PROFINET address, 736
PROFINET CBA
Updating inputs of the program interface with PN_IN, 7920
Updating interconnections with PN_DP, 7922
Updating outputs of the PROFINET interface with PN_OUT, 7921
PROFINET device name, 1447, 1549, (See display)
PROFINET interface, 1450, 1586, 1860, 1871, 2092, 2151
PROFINET interface parameters
Reset, 2162
PROFINET IO, 1436, 1441, 1442, 1452, 1453, 1492
Port options, 1458, 8440, 8489, 8542
PROFINET IO device
Assigning a name, 2116
Name assignment in the dialog, 2118
Name assignment in the online and diagnostics view, 2117
PROFINET IO devices
Reading a portion of the inputs with GETIO_PART, 4562, 6465
Reading consistent data with DPRD_DAT, 4600, 6498
Writing a part of the outputs with SETIO_PART, 4563, 6467
Writing all outputs with SETIO, 4561, 6463
Writing consistent data with DPWR_DAT, 4601, 6500
PROFINET IO proxy, 2003
PROFINET IO system, 1453
PROFINET IRT, 1480
PROFINET parameters
HMI connection, 8433, 8480, 8535
HMI device, 8434, 8482, 8536
PLC, 8436, 8483, 8538
S7 1200, 8480, 8483
S7 1500, 8433, 8434, 8436
S7 300/400, 8535, 8536, 8538
PROFINET RT class, 1437
PROFINET CBA component
Updating with instructions, 7918
Program
Cyclic (S7-300, S7-400), 1754
Determining runtime with OB_RT, 6739
Program alarm
Creating, 3054, 3062
Deleting, 3056, 3064
Editing, 3054, 3055, 3062, 3063
generating with Program_Alarm, 4722
Program card, 1570
Program control operations, 3387, 3687, 3848, 4278, 4433, 5499, 5711, 5836, 6207, 6332
Program cycle OB
Description, 1614
Program display, 4110, 6118
Program editor
"Instructions" task card, 2525
Block interface, 2525
Enlarging the programming window, 2531
Favorites, 2525
Function, 2523
General settings, 273, 275, 276, 2542
Programming window, 2525
Structure, 2524
Testing task card, 2525
Toolbar, 2525
Using the keyboard, 2527
Program execution
Cyclic, 1614
Program execution error OB (OB 85) (S7-300, S7-400), 1791
Program information, 1841
Displaying, 2936
In the assignment list, 2937
In the call structure, 2945
In the dependency structure, 2952
in the tab , 2957
Views, 2936
Program loop, 4057, 6058
Program resources, 48, 2500
Program status
Call environment, 2968, 2969
Disable all steps, 2984
Editing blocks, 2979
FBD, 2988
Function, 2967
GRAPH, 2981, 2984, 2985, 2991
Initialize sequencer, 2984
LAD, 2987
Modify tag, 2979
Operating modes, 2981, 2985

SCL, 2990
STL , 2989
Switching off, 2977
Switching on, 2977
System synchronization, 2982, 2986
Program_Alarm, 4722
Programming, 10315
Linear, 2168
Modbus, 1012
Motion Control instructions, 10319
PtP, 1006
PtP instructions, 5179, 7467
Structured, 2168
USS, 1010
Programming error (S7-1500), 1912
Programming error OB (S7-1500), 1912
Programming error OB (S7-300, S7-400), 1811
Programming language
Change over, 2758
Changeover, 2820
FBD, 2611
GRAPH, 2718
LAD, 2568
Rules for changing, 2820
Programming recommendations, 246
Programming window
Alarm view, 2770
Customizing, 2692
GRAPH, 2762
Permanent post-instructions, 2769
Permanent pre-instructions, 2764
SCL, 2691
Sequence view, 2765
Single step view, 2767
Project
Basics, 331
Carrying out an online/offline comparison, 364
Carrying out offline/offline comparisons, 364
Closing, 339
Comparing data, 38, 363
compile (overview), 8880
Compiling , 8882
Compiling project data, 354
Creating, 332
Deleting, 339
Detailed comparison, 374
Download, 8885, 8903
download (overview), 8880
Harmonizing, 469
Migrate, 163
Migrating, 145
multilingual, 8832
Multiple connections, 8458, 8508, 8566, 8605
Saving, 338
Showing properties, 338
Simulation with a tag simulator, 8889
Upgrade, 8874
Project archives, 348
Project data
Compiling, 354
Downloading to a memory card, 359
Export alarm, 8800
Export text list, 8812
Exporting a recipe, 8796
Exporting tag, 8807
Importing alarms, 8801
Information on compiling, 353
Information on loading, 355
IO-Link master module, 39
Loading, 357
Uploading from a device, 360
Project file
Initialization, 8790
Initializing, 8790
Project ID, 8380
Project language, 340, 8826
Activate, 8830
Disabling, 8830
Project languages
Changing the editing language, 343
Specifying project languages, 342
System texts, 8833
Use, 340
User texts, 8833
Project library, 58, 416, 8762
Adding types, 449
Assigning a common version to types, 463
Cleaning up, 470
Creating folders, 441
Discarding all versions, 455
Discarding the type version, 455
Migrating, 157
Performing a consistency check for a type version, 454
Release all versions, 457
Release the type versions, 456
Updating the project, 458
Using filter view, 443
Using the element view, 419
Using types, 450
Project text
Exporting, 8817, 8837
Project texts
Application example, 347

Changing the text of selected objects, 344
Displaying reference text, 344, 8836
Exporting all texts, 346
Exporting individual texts, 345
Importing, 346, 8819, 8839
Migrating, 155
Translating individual texts, 8834
Translating project texts, 343
Translation into project languages, 340, 8833
Project tree
Adding device, 505
Function, 282
Hiding, 2531
Reducing automatically, 288
Showing, 2531
Project version
Upgrade, 8874
Project view, 279
Properties (CPU), 1586, 1860
Properties of the VPN group, 786
ProSave, 8893
Installation, 73
Protect
Function key with password, 8063
PROTECT, 5500, 5712, 5837, 6208, 6333
Protection concept, 1610, 1613, 1885, 1888, 1889
Introduction, 378
Revoking access rights, 379
Protection level, 1609, 1610, 1690, 1884
Changing, 5500, 5712, 5837, 6208, 6333
Revoking access rights, 379
Protocol, 750
Protocol parameters, 1077
Protocols of the communication modules, 976
Proximity switch, 10225
Proxy ARP, 841
PRVREC, 4606
PT, 3491
PTC resistor, 2050
PTO, 48
PtP communication
Programming, 5179, 7467
PtP error classes, 5184, 7471
PtP instructions, 1008
PtP instructions return values, 5183, 7470
Pull or plug of modules (S7-1500), 1909
Pull/plug interrupt (ET 200M), 2061
Pull/plug interrupt OB, 1626
Pull/plug interrupt OB (S7-1500), 1909
Pull/plug interrupt OB (S7-300, S7-400), 1786
Pulling and plugging Ethernet PC CPs, 1988
Pulse, 3177, 3187, 3198, 3212, 3472, 3493, 3507, 3750, 3935, 4116, 4131, 4331, 5389, 5395, 5408, 5600, 5606, 5619, 5796, 5939, 6120, 6129, 6272
Extended, 3201, 3213, 3496, 3508, 3937, 4134, 5397, 5409, 5608, 5620, 5941, 6131
PULSE, 7348
Pulse duration, 9142, 9206
Default setting, 9142, 9206
Range of values, 9142, 9206
Pulse extension, 2046
Pulse generator
Enabling/disabling with CTRL_PWM, 4766
Pulse interface
Principle, 10011
PULSE REFEED, 10754
Pulse stretching, 2064
Pulse width modulation with PULSE, 7348
PULSEGEN
Instruction, 6803
Mode of operation, 6804
PULSEGEN_M
Input parameters, 7006
Instruction, 6999
Operating principle, 7000
Output parameters, 7007
Pulse-width modulation (CPU 31xC), 1704
PUSH, 4105, 6103
PUT, 5040
PUT / GET, 1613, 1888
PUT frame, 1061
PWM, 48

Q

Q
PLC, 8679
Q address, 881, 937
Q series, 8682
Q0xUDEH CPU
PLC, 8681
QB, 2477
QD, 2477
QI (quality information), (See value status)
QRY_CINT, 4696
QRY_DINT, 4708
QRY_TINT, 4704
Quadruple evaluation, 9109
Quantum, 8709
Query architecture, 5181, 7469
Query architecture master, 5182, 7469
Query architecture slave, 5182, 7470
QW, 2477

QX, 2477

R

R, 531, 3160, 3454, 3918, 3933, 3950, 5378, 5589, 5923, 5938, 5954
R_TRIG, 3174, 3469, 3747, 4113
Rack error OB, 1627
Rack failure OB (OB 86) (S7-300, S7-400), 1796
Rack failure OB (S7-1500), 1910
Rack or station failure, 1627
Rack or station failure (S7-1500), 1910
Racks, 498, 876
    Inserting a module, 508
    Maximum number, 865, 866
    S7-300 rules for arranging components, 865
    S7-400 rules for arranging components, 866
Radius, 8086
RADIUS, 1328
RALRM, 4564
Ramp-down time, 10285, 10287, 10301, 10303
Ramp-up time, 10285, 10287, 10301, 10303
Range
    Value outside range, 3252, 3548
    Value within range, 3251, 3547
Range (... - ...)
    Graphics list, 8038
RCV_CFG, 5281
RCV_PTP, 1030, 5290, 7896
RCV_RST, 5291
RCVREC, 4603
RD_ADDR , 4813
RD_DPAR, 4683
RD_DPARA, 4685
RD_DPARM, 4688
RD_LGADR, 4818
RD_LOC_T, 4510
RD_REC, 4594
RD_SINFO, 4729
RD_SYS_T, 4509
RDREC, 885, 4557
RE_TRIGR, 3387, 3687, 3847, 4277, 4432, 5498, 5710, 5835, 6205, 6331
Reaction to CPU STOP, 9046
Reaction to CPU/Master STOP
    1STEP 5V, 10735
Read
    Feedback signals, 7125
Read continuously
    Tag, 8129
Read field, 3314, 3612
Read job
    CNT2RDPN, 7123
Read jobs, 7143, 7146
Read memory address, 4209
Read memory bit, 4211
Read system time with TIME_TCK, 4516, 6414
READ_355
    Input parameters, 7207
    Instruction, 7206
    Output parameters, 7208
READ_455
    Input parameters, 7287
    Instruction, 7287
READ_BIG, 4222
READ_DBL, 51, 4801
READ_ERR, 4717
READ_LITTLE, 4218
ReadFromArrayDB, 3332, 3631, 3806, 4199, 4390
ReadFromArrayDBL, 3337, 3635, 3811, 4203, 4394
Reading
    A data record with RD_REC, 4594, 6491
    Asynchronously- data record of a module with RD_DPARA, 4685, 6595
    Cam data, 9357
    Data record of a module with RD_DPAR, 4683, 6593
    Dynamically allocated system resources with READ_SI, 6675
    Feedback signals, 7129
    from a data block in load memory with READ_DBL, 4801, 6747
    Incremental dimension tables, 10466, 10556
    Local time with RD_LOC_T, 4510
    Machine data, 9356, 10464, 10554
    Reading the local time with LOC_TIME, 6423
    Reading the runtime meter with READ_RTM, 6422
    SZL partial list with RDSYSST, 6681
Reading data
    From DP standard slaves/PROFINET IO devices with DPRD_DAT, 4600, 6498
    From remote CPU with GET, 5038
    outside of the local S7 station with X_GET, 7928
    within the local S7 station with I_GET, 7913
Reading out
    LED status with LED, 4740
    Recipe data record, 8957
Reading out a diagnostics buffer, 2101
Reading out service data, 906
Reading tags, 918, 953
Read-only, 1609
ReadPLCMode, 8310

REAL, 2252, 2304, 2351, 2384, 2408, 2425, 2439
REAL_TO_, 2351, 2408, 2439
Real-time clock, 2152
Receive buffer, 995, 1047, 1085
Receive buffer size, 975
Receive line initial state, 983
Receive_Config, 977, 1008
Receive_Config (receive configuration), 5188, 7476
Receive_P2P, 977, 1008
Receive_P2P (receive point-to-point data) , 5197, 7484
Receive_Reset, 977, 1008
Receive_Reset (reset receiver), 5199, 7486
Receiver module, 876
receiving data
    3964(R) procedure, 1057
    ASCII driver, 1042, 1076
Receiving data
    and storing with RCV_PTP, 7896
    and supplying with SERVE_RK, 7909
    from outside of the local S7 station with X_RCV, 7924
    on I-device with RCVREC, 4603, 6503
Receiving SMS messages, 5162
Recipe, 947, 8211, 8212, 8213, 8799, 8949
    Application example: Machine parameter assignment, 8212
    Application example:Batch production, 8212
    Basics, 8211, 8212
    Control, 8951
    Creating new, 8223
    Data flow, 8952
    Data record, 8213, 8950
    Entry, 8213
    Exporting, 8796
    Exporting to CSV file, 4768
    Importing, 8798
    Importing into data block, 4770
    Memory requirements, 8970
    Use, 8212
    Use of text lists, 8218
Recipe data
    Migration, 158
Recipe data record
    Create on the HMI device, 8236, 8237
    Creating, 8955
    Creating new, 8223
    Creating on the HMI device, 8236
    Deleting, 8957
    Editing, 8956
    Reading from PLC, 8957
    Synchronize with PLC, 8956
    Transferring the project, 79
    Transferring to PLC, 8958
    Use of text lists, 8218
Recipe element
    Creating new, 8223
Recipe list, 8219
Recipe view, 8219
    Application, 8097
    Behavior with screen change, 8220
    Configuring, 8230
    Layout, 8097
    Operation using the function key, 8220
    Recipe data record, 8214, 8221
    Simple, 8219
    Toolbar, 8097
    Updating, 8220
RecipeExport, 4768
RecipeImport, 4770
RecipeViewBack, 8321
RecipeViewClearDataRecord, 8317
RecipeViewGetDataRecordFromPLC, 8317
RecipeViewMenu, 8318
RecipeViewNewDataRecord, 8317
RecipeViewOpen, 8318
RecipeViewRenameDataRecord, 8320
RecipeViewSaveAsDataRecord, 8320
RecipeViewSaveDataRecord, 8320
RecipeViewSetDataRecordToPLC, 8319
RecipeViewShowOperatorNotes, 8321
Recommissioning
    HMI device, 8903
Reconfiguration via user program, 896
Reconnecting
    Tag, 8124
Reconnecting a PLC tag, 8124
Recording a log file for the modem, 10973
Rectangle
    Configuring, 8024
    Inserting, 8023
    Inserting and configuring, 8022
    Radius corners X, 8096
    Radius corners Y, 8096
RECV_440, 1030
Redoing actions
    Basics of redoing actions, 402
    Redoing actions, 406
Reduction of process reaction time, 1409
Reduction of the system cycle, 1409
Redundancy group, 1985
Redundancy port, 1985
Redundant power supply modules, 866
Redundant sync master, 1479

Reference channel, 2047
Reference channel error, 2055
Reference channel of the module, 2057
Reference junction, 1358, 1426, 2024, 2039, 2047, 2057
Reference language, 340, 8826
  Selecting, 8830
Reference object
  Defining, 7989
Reference point approach, 10476, 10567
  Cancel, 10480, 10571
  Data in the channel DB, 10479, 10570
  Data in the parameter DB, 10480, 10571
  Effects, 10480, 10571
  Requirement, 10477, 10568
  Start speed, 10516, 10613
  Synchronization, 10476, 10567
  Types, 10480, 10571
Reference point coordinate, 9389, 9464, 10512, 10609
Reference point switch, 10477, 10568
Reference project
  Basics, 351
  Closing, 351
  Comparing, 352
  Opening, 351
Reference temperature, 2047, 2057
Reference values for Motion Control, 10218
References, 246
Regional language for the web server, 910, 911
Reinitialization of technology objects, 10336
Relative positioning
  Easy Motion Control, 10772
release, 8360
Release key, 8362
ReleaseButton, 8325
Remainder of division, 3273, 3569, 5445, 5657
Remote access user, 739
Remote connection
  as CPU-controlled remote connection, 10965
  as dial-up connection, 10928
  As VPN connection, 10952
Remote connection cannot be established, 10951
Remote connection from the TS Adapter is not established, 10975
Remote connection to the TS Adapter is not established, 10974
Remote desktop, 32
Remote system access to a programming device/ personal computer, 10966
Remove
  Formatting in the alarm text, 8167
  Object from the group, 8017
Removing
  Folder link, 8010
Rename
  Screen, 7968
  Tag, 8120
  Template, 7974
Renewing the CA group certificate, 791
REPEAT, 4266, 6198
REPL_VAL, 5548, 5758, 5858, 6235, 6362
Replace
  Cross-references, 8823
REPLACE, 4546
Replace modules during operation, 2061
Replacing
  Module, 513
Replacing a module, 7713, 7851
Replacing devices, 513
  Limitations on connections, 8854
  Principle, 8854
Report system errors, (See Report System Errors)
Report System Errors, 909, (See System diagnostics)
  Created error OBs, 2140
Reporting system errors, (See System diagnostics)
Requirement
  Edge detection, 10586
  Encoder data, 10495, 10592
  Incremental approach, 10483, 10574
  Jog, 10473
  Jogging, 10564
  Length measurement, 10586
  Loop traverse, 10491, 10588
  Reference point approach, 10477, 10568
  Set actual value, 10488, 10580
  Set reference point, 10490, 10584
  Setting actual value on the fly, 10580
  Zero offset , 10582
Requirements for establishing a remote connection, 10950
Requirements for sending an SMS, 10969
Requirements for sending emails, 10971
RES_RCVB, 1030, 7899
RES_RECV, 1030
Reserved event classes, 1831
Reset, 1225
  Bit array, 5542, 5752, 5853, 6230, 6357
  Bit field, 3163, 3457
  Byte array, 5545, 5856, 6233, 6360
  Counter operations, 3950, 5954
  I/O area, 5544, 5754, 5854, 6231, 6358
  IEC timer, 3195, 3762, 4128
  of PROFINET interface parameters, 2162

Operand, 3160, 3165, 3454, 3460, 3918, 5378, 5382, 5589, 5593, 5923
RLO, 3922, 5927
Set byte array, 5755
Timer, 3933, 5938
To factory settings, 2111, 2112
to factory settings , 8897
RESET, 5542, 5752, 5853, 6230, 6357
Reset timer, 3490
Reset to factory settings, 1842
RESET_BF, 3163, 3457
ResetBit, 8322
ResetBitInTag, 8323
RESETI, 5545, 5755, 5856, 6233, 6360
RESETP, 5544, 5754, 5854, 6231, 6358
Resetting
to factory settings, 8897
Resetting the connection, 5115
Resetting the MPI/PROFIBUS settings, 10877
Resetting the TCP/IP settings, 10869
Resetting the user interface layout, 307
Resolution, 9398, 9474, 10519, 10616
Range of values, 10520, 10617
Value range, 9399, 9475
Resources, 1841, 2957
Code work memory, 2957
Data work memory, 2957
Introduction, 2957
Load memory, 2957
Retentive memory, 2957
Retentive memory data, 2957
Work memory, 2957
Resources tab
Structure, 2959
Response message frame, 1061
Structure and contents, 1063
Response timeout, 992
Restart, 1225, 1674
Restart of technology objects, 10336
Restore
Data of the HMI device, 8896
Restore process
License key, 101
Restore window layout, 305
Restoring, 8936
Data of the HMI device, 8894
HMI device data, 8910
Restoring a backup, 10859
Restoring a device, 10859
Restoring data
HMI device, 8894, 8896, 8910
Restriction
Modicon Modbus RTU, 8725
Modicon Modbus TCP/IP, 8725
Restrictions due to user rights, 318, 319
Recognizing, 319
Result of logic operation, 2453, 2466
RET, 3380, 3680, 5494, 5706
RET_VAL for asynchronous instructions, 5367
RET_VALU messages, 9866, 9867, 9915, 9916, 9948, 9949
Retain, 2177
Retentive, 46
Retentive address area
Load memory, work memory, system memory, 1656
Retentive behavior, 1677
Bit memory, timers, counters, 2484
Retentive behavior after power failure, 1644
Retentive bit memories
Enabling the display, 2945
Retentive memory, 1656, 1839, 2957
Retentivity, 1574, 1842, 2175
Bit memory, timers, counters, 2483
Block interface, 2561
Data block, 2839
Data block , 2840
PLC tag, 2483, 2484
Retentivity of IP address parameters, 1450
Retrieving projects, 350
Retrigger reference point, 9388, 9464
RETURN, 4271, 6202
Return values
ABS_CTRL, 7126
ABS_CTRL_451, 7130
ABS_DIAG, 7128
ABS_DIAG_451, 7132
ABS_INIT , 7125
CAM_CTRL, 7158
CAM_CTRL_452, 7161
CAM_DIAG, 7159
CAM_DIAG_452, 7162
CAM_INIT, 7157
CAM_MSRM_452, 7164
PtP instructions, 5183, 7470
Return values receive runtime, 5197, 7484
Reversing cam, 10225, 10237
Reversing switch, 10477, 10568
RI, 982
RIGHT, 4543
Ring port, 1516, 1518
Ring Redundancy, 1278

RIP
RIPv2 statistics, 1219
RIPv2
Configuration, 1325
Interfaces, 1325
RK 512 protocol (CPU 31xC), 1737
RLD, 4098, 6096
RLDA, 4102, 6100
RLO, 2453, 2466
Invert, 3158, 3452, 5376, 5587
Inverting, 3920, 5925
Reset to 0, 3922, 5927
set to 1, 3921, 5926
RMON
History, 1303
Statistics, 1302
RMP_SOAK
Input parameters, 7013
Instruction, 7007
Output parameters, 7013
RND, 4024, 6027
RND-, 4028, 6030
RND+, 4027, 6029
ROC_LIM
Input parameters, 7020
Instruction, 7014
Output parameters, 7020
ROL, 3421, 3723, 4304, 4457, 5521, 5733, 6224, 6350
Role name, 740
Roles, 739
System-defined, 739
User-defined, 740
Root certification authorities, 735
Root, square, 4170, 6161
ROR, 3419, 3721, 4301, 4455, 5519, 5731, 6222, 6348
Rotary axis
Easy Motion Control, 10770
Rotary axis (CPU 31xC), 1716, 1723
Rotate
Left, 3421, 3723, 4098, 4100, 4102, 4304, 4457, 5521, 5733, 6096, 6098, 6100, 6224, 6350
Object, 7982
Right, 3419, 3721, 4103, 4301, 4455, 5519, 5731, 6101, 6222, 6348
Rotational speed measurement, 9094
Round, 3357, 3358, 3359, 3361, 3656, 3657, 3659, 3660, 4024, 4025, 4027, 4028, 4240, 4241, 4242, 4411, 4413, 4414, 4415, 5481, 5482, 5483, 5485, 5693, 5694, 5696, 5697, 6027, 6028, 6029, 6030, 6181, 6182, 6183, 6319, 6320, 6321, 6323
ROUND, 3357, 3656, 4240, 4411, 5481, 5693, 6181, 6319
Router, 534
Router IP address, 829
Routing, 1186
Routing table, 1212
Static routes, 1186
VRRP, 1186
Routing mode, 767
RRD, 4100, 6098
RRDA, 4103, 6101
RS, 3165, 3460, 5382, 5593
RS232 accompanying signals
Automatic use, 987
RS232 mode, 981
RS232 signals, 982
RS-232/PIP multi-master cable, 1603
RS232C accompanying signals, 1048
Automatic use, 1049
RS422 mode, 983
RS485 mode, 984
RT, 3195, 3490, 3762, 4128
RT class, 1437
RT_INFO, 4738
RTM, 2283, 4517
RTS, 982
Rules
STL network, 2663
Rules for configuring MPI networks
Rules for assigning the MPI address, 527
Rules for network configuration for PROFIBUS networks
Assigning device addresses, 527, 528
RUN, 1569, 1648, 1650, 1836
Rung
Deleting, 2611
Inserting, 2611
Use, 2609
Runtime
Execution of the function list, 8281
Mouse wheel, 77
Simulating, 8887
Start screen, 7969
RUNTIME, 3397, 3697, 3858, 4287
Measure program runtime, 3397, 3697, 3858, 4287
Runtime data
Migration, 158
Runtime language, 8826, 8844, 8911
Font, 8849, 8912
Log, 8850
Order for language switching, 8847

Selecting, 8845, 8911
Runtime meter
Display information, 2156
Run-time meter, 1686
Runtime meters
Handling with RTM, 4517, 6419
Properties, 6418
Range of values, 6418
Reading with READ_RTM, 6422
Setting with SET_RTM, 6420
Setting, starting, stopping and reading with RTM, 6419
Starting and stopping with CTRL_RTM, 6421
Runtime settings
User administration, 7970, 8245, 8261, 8262
Runtime start
Basic Panel, 8886, 8907
Runtime Stop, 8361
Runtime version, 8881
RW_REC, 7771

S

S, 3161, 3455, 3919, 3951, 5379, 5590, 5924, 5955
S_AVERZ, 3209, 5405
S_CD, 3233, 3530, 5426, 5639
S_COMP, 4519
S_CONV, 4520
S_CU, 3231, 3528, 4153, 5424, 5637, 6148
S_CUD, 3526, 5635
S_EVERZ, 3204, 5400
S_MODB, 1031
S_MOVE, 4519
S_ODT, 3204, 3499, 4137, 5400, 5611, 6134
S_ODTS, 3207, 3502, 4139, 5403, 5614, 6136
S_OFFDT, 3209, 3504, 4142, 5405, 5616, 6139
S_PEXT, 3201, 3496, 4134, 5397, 5608, 6131
S_PULSE, 3198, 3493, 4131, 5395, 5606, 6129
S_RCV, 1031
S_RTS, 1031
S_SEND, 1031, 1098
S_SEVERZ, 3207, 5403
S_USSI, 1031
S_USSR, 1031
S_USST, 1031
S_V24, 1031
S_VSET, 1031
S_VSTAT, 1031
S_XON, 1031
S5 addressing mode, 1960
S5TIME, 2255, 2306, 2354, 2426, 2441
S5TIME_TO_, 2354, 2441
S7 1200
Communication, 8477
Connection parameters, 8526
HMI connection, 8479
Parameters, 8526
PROFIBUS, 8492
PROFIBUS parameters, 8494, 8496, 8497
PROFINET, 8478
S7 1500
Communication, 8429
Connection parameters, 8472
Data types, 8471
HMI connection, 8431
Parameters, 8472
PROFIBUS, 8443, 8445
PROFIBUS parameters, 8446, 8448, 8449
PROFINET, 8430
PROFINET parameters, 8433, 8434, 8436
S7 200
Communication, 8591
Connection, 8591, 8619
Connection parameters, 8594
Data types, 8618
MPI, 8589, 8598
Parameters, 8589, 8594, 8598, 8599
PPI, 8599
S7 300
Communication, 8531
MPI address, 8560
S7 300/400
Connection parameters, 8584
Data types, 8578
HMI connection, 8533
MPI, 8553, 8554
MPI parameters, 8556, 8558, 8559
Parameters, 8584
PROFIBUS, 8545, 8546
PROFIBUS parameters, 8548, 8549, 8551
PROFINET, 8532
PROFINET parameters, 8535, 8536, 8538
S7 400
Communication, 8531
MPI address, 8561
S7 addressing mode, 1960
S7 basic communication
Common parameters, 7351
I_ABORT, 7916
I_PUT, 7915
X_ABORT, 7931
X_GET, 7928
X_PUT, 7930
X_RCV, 7924

X_SEND, 7923
S7 communication
Performing a warm or cold restart with START, 7400
Change in operating mode, 7357
Data exchange , 7356
Determining the connection status with CONTROL, 7391
Error behavior, 7364
Overview of instructions, 7356
Parameter, 5035
Parameters, 7359
Performing a hot restart with RESUME, 7404
Putting the device in the STOP operating mode with STOP, 7402
Query connection, 7357
Query operating mode, 7357
Querying connection status with C_CNTRL, 7381
Querying the device status with STATUS, 7405
Read data from a remote CPU with GET_S, 7383
Read data from remote CPU with GET, 7365
Receive data from a remote partner instruction with BRCV, 5048, 7378
Receive data from a remote partner instruction with URCV, 5044, 7373, 7389
Receiving device status change with USTATUS, 7408
Send and receive parameters, 5036, 7361
Send data to a printer with PRINT, 7394
Send data to a remote partner instruction with BSEND, 5046, 7375
Send data to a remote partner instruction with USEND, 5043, 7370
Send data to a remote partner instruction with USEND_S, 7387
Startup characteristics, 7362
User data size, 5037, 7361
Write data to a remote CPU with PUT_S, 7385
Write data to remote CPU with PUT, 7368
S7 connection, 541
TSAP, 552, 629
S7 CPU
Work memory, 1572
Load memory, 1572
Memory areas, 1655
Operand area, 1572, 1659
S7 routing
Loading, 8884
S7-1200, 40
PROFIBUS, 8493
S7-1200 modules, 40
S7-1200 V2
Data types, 8522
S7-1500
Application area, 867
Definition, 867
Power segment, 868
Properties, 867
S7-300
Slot rules, 865
S7-400
Slot rules, 866
S7-400 CPUs
Memory types, 1657
S7-CP, 726
S7-modular Embedded Controller, 1161
Configuring, 1162
S7-PCT, 1552
S7-Port Configuration Tool, 40
SA lifetime, 788
Safety guidelines, 327
Safety instruction
Changing the recipe data record in the background, 8220
Switching tooltip, 8921
Safety precautions when forcing tags, 3023, 3039
SALRM, 1370
Sample program
OB 122, 1815
OB 81 (S7-300, S7-400), 1782
Save
Data of the HMI device, 8895
Global library, 8772
RLO in BR bit, 3923, 5528, 5739, 5928
SAVE, 3923, 5528, 5739, 5928
Save message text, 1073
Save table layout, 304
Save user interface
Save layout in editors, 304
Save window layout, 304
Save window layout, 304
Saving
Axis data block, 10809
Saving a force table, 3027
Saving a watch table, 3000
Saving phone books, 10923
Saving the user interface, 304
SC, 3235, 3532, 5428, 5641
SCALANCE S, 725
Scale, 3362, 3367, 3662, 3667, 3827, 3836, 4245, 4254, 4416, 4421, 5486, 5699, 5827, 6186, 6324
SCALE, 3367, 3667, 3836, 4254, 4421, 5486, 5699, 5827, 6186, 6324

SCALE_M
Input parameters, 7022
Instruction, 7021
Output parameters, 7022
SCALE_X, 3362, 3662, 3827, 4245, 4416
Scaling
Applying linear scaling to a tag, 8129, 8131
Scan operand for negative signal edge, 3168, 3463, 5384, 5595
Scan operand for positive signal edge, 3167, 3461, 5383, 5594
Scan RLO for negative signal edge, 3173, 3468, 5387, 5598
Scan RLO for positive signal edge, 3172, 3467, 5386, 5597
Scheduler, 8364
Deactivated task, 8369
Event trigger, 8369, 8372
Function list, 8368
Triggers, 8368
Work area, 8367
SCL, 46, 2680, 3110, 3111, 3145, 3146
Detailed comparison, 2883
Insert comment, 4271, 6203
Migration, 137
Programming window, 2691
SCL instruction
Changing the data type, 2701, 2702
Data type basics, 2700
Inserting, 2698, 2699
Rules, 2697
Selecting, 2716
SCL program
Inserting a block call, 2706, 2707, 2709, 2710, 2711, 2712
Inserting an instruction, 2698, 2699
Inserting comments, 2715
Program status display, 2990
Selecting an instruction, 2716
Showing and hiding the parameter list, 2714
Screen
Availability for specific devices, 7963
Copy, 7968
Copying, 8865
Creating, 7967
Creating a type, 8781
Deleting, 7968
Font, 7963
Inserting, 7968
Library, 8781
Move, 7968
Rename, 7968
Use template, 7976
Working steps in creating, 7967
Zooming, 7966
Screen change, 8356
Screen keyboard
Alphanumerical, 8918, 8920
Key assignment, 8918, 8919
KTP400 Basic, 8917
KTP600, KTP1000, TP1500 Basic, 8919
Language behavior, 75
Numerical, 8918, 8920
Screen number, 8380
Screen object
Dynamic movement, 8048
Screen template, 8058
ScreenObjectCursorDown, 8294
ScreenObjectCursorUp, 8294
ScreenObjectPageDown, 8295
ScreenObjectPageUp, 8295
Script
Creating a type, 8780
Library, 8780
Updating tag value, 8111, 8413
SD, 3213, 3215, 3510, 3937, 3939, 5409, 5411, 5622, 5941, 5943
SD card
Open force job, 33
SE, 3213, 3508, 3937, 5409, 5620, 5941
Search
Hardware catalog, 497
Secondary sync master, (see Redundant sync master)
Security, 45
Security (web server), 906
Security Configuration Tool
Configuration views, 727
Security configuration: Symbolic names, 44
Security events, 805, 807
Security module, 726
Security online diagnostics of S7 CPs, 44
Security program, 93, 97
Security settings, 1234
Security system, 8935
SEG, 3441, 3743, 3894, 4323, 4485, 5577, 5792, 5899, 6263, 6393
Segment diagnostics
Graphic display, 2130
Icons, 2129
Text display, 2130
SEL, 3408, 3708, 3863, 4291, 4445, 5514, 5726, 5845, 6215, 6342

Select, 3408, 3708, 3863, 4291, 4445, 5514, 5726, 6215, 6342
Multiple objects, 7988
Selecting, 5845
Multiple objects, 7982
Selecting gate control, 9085, 9086, 9093, 9094, 9096, 9098
Selecting specific connection path, 607
Selecting the Gate Contrtol, 9084
Selection changed, 8355
Send clock, 1468, 1477
Send clock adaptation, 1468
Send clock cycle, 1456
SEND frame, 1061, 1065
SEND/PUT message frame, 1061, 1062, 1065
SEND_440, 1030
SEND_CFG, 5280
Send_Config, 977, 1008
Send_Config (send configuration), 5187, 7474
Send_P2P, 977, 1008
Send_P2P (send point-to-point data), 5195, 7482
SEND_PTP, 1030, 5288, 7894
SEND_RK, 1030, 7900
Sender module, 876
Sending data
outside of the local S7 station with X_SEND, 7923
With TSEND, 7436
Sending Data
3964(R) procedure, 1054
ASCII driver, 1041, 1075
RK512, 1064
Sending SMS messages, 5158
Sensor, 10211
Sensor type, 9031, 9056
Sequence
Checkback signals for the diagnostics, 10496, 10594
Checkback signals for the positioning, 10496, 10593
Edge detection, 10586
Incremental approach with increment number 1-100, 10483, 10574
Incremental approach with increment number 254, 10484, 10576
Incremental approach with increment number 255, 10484, 10576
Incremental approach with incremental dimension number 252, 10575
Jog, 10473
Jogging, 10564
Length measurement, 10586
Loop traverse, 10492, 10588
Read encoder data, 10495, 10592
Read position data, 10494, 10592
Set actual value, 10488, 10580
Set reference point, 10490, 10584
Setting actual value on the fly, 10580
Zero offset , 10582
Sequence end
Ending sequencers, 2777
Introduction, 2723
Sequence view, 2765
Sequencer, 3423, 3725, 3865, 4306, 4459, 5549, 5552, 5759, 5762, 5859, 5867, 6236, 6239, 6363, 6366
Adding comments, 2775
Basics, 2720
Deleting, 2778
Elements, 2721
Ending, 2777
Initializing, 2984
Inserting, 2774
Inserting new elements, 2776
Renaming, 2775
Serial data transmission, 978
Serial number, 1196
Serialization, 3609, 3789, 4184, 4373
Serialize, 3310, 3609, 3789, 4184, 4373
SERVE_RK, 1030, 7909
Service & Support, 8397
Service data
Saving, 2106
Service groups, 752, 754
Service_Request_ID, 7789
Servo drive type, 10750
Servo motor, 10006
Set
Bit array, 5537, 5747, 5847, 6226, 6352
Bit field, 3162, 3456
Byte array, 5540, 5850, 6229, 6355
Counter initial value, 5428
Counter operations, 3951, 5955
Counter start value, 3235, 3532, 5641
I/O area, 5538, 5848, 6227, 6353
Operand, 3161, 3164, 3455, 3459, 3919, 5379, 5380, 5590, 5591, 5924
RLO, 3921, 5926
Set byte array, 5750
SET, 3921, 5537, 5747, 5847, 5926, 6226, 6352, 9125
Set actual value, 10488, 10580
Data in the channel DB, 10488, 10580
Effects, 10489, 10581

Requirement, 10488, 10580
Sequence, 10488, 10580
Set limit value, 3283, 3580, 3780, 4167, 4356, 5452, 5665, 5815, 6158, 6292
Set operand on negative signal edge, 3171, 3465
Set operand on positive signal edge, 3169, 3464
Set process value, 9361, 9437
Data in the channel DB, 9362, 9438
Effect, 9363, 9438
Requirement, 9362, 9437
Sequence, 9362, 9437
Set process value on-the-fly, 9361, 9437
Data in the channel DB, 9362, 9438
Requirement, 9362, 9437
Sequence, 9362, 9437
Set reference point, 9366, 9442, 10490, 10584
Data in the channel DB, 10490, 10584
Data used in the channel DB, 9367, 9442
Definition, 9366, 9442
Effects, 9367, 9443, 10490, 10585
Requirement, 10490, 10584
Requirements, 9367, 9442
Sequence, 10490, 10584
Special features of the absolute encoder, 9368, 9443
Steps, 9367, 9442
Synchronization, 10490, 10584
Set tag on negative signal edge, 3176, 3470, 3748, 4114
Set tag on positive signal edge, 3174, 3469, 3747, 4113
SET_BF, 3162, 3456
SET_CINT, 4694
SET_CLKS, 6415
Set_Features, 978, 1009
SET_TIMEZONE, 4513
SET_TINT, 4697
SET_TINTL, 4699
SetBacklightColor, 8333
SetBit, 8329
SetBitInTag, 8330
SetBitWhileKeyPressed, 8332
SetConnectionMode, 8337
SetDeviceMode, 8328
SETI, 5540, 5750, 5850, 6229, 6355
SETIO, 4561
SETIO_PART, 4563, 6467
SetLanguage, 8335
SETP, 5538, 5748, 5848, 6227, 6353
SetPLCMode, 8334
SetTag, 8336
Setting
Deadband, 8165
Delay time, 8165
Language, 8922
Languages in the operating system, 8827
Style, 8005
View options for the assignment list, 2941
View options for the dependency structure, 2955
Setting actual value on the fly, 10580
Data in the channel DB, 10580
Requirement, 10580
Sequence, 10580
Setting AS-i network configuration parameters, 540
Setting ENO automatically, 2689
Setting language, 8922
Setting parameters for the Ethernet interface, 10867
Adding a dynamic IP address, 10868
Connecting to a subnet, 10866
Deleting dynamic IP addresses, 10868
Modifiable parameters, 10865
Options in the parameter settings, 10865
Setting the communication load, 534
Setting the counter, 9125, 9201
With an external signal, 9126, 9202
With digital input 1I2, 9202
With digital input I2, 9126
With the zero mark, 9128, 9204
Setting the font, size and color , 2692
Setting the mnemonics, 2533
Setting the tab spacing , 2692
Setting time-of-day status with SET_CLKS, 6415
Settings
Changing, 276, 2543, 2571, 2614, 2655, 2690, 2757
FBD, 2571, 2614
General, 273, 275, 276, 2542
GRAPH, 2756
LAD, 2571, 2614
SCL, 2689
STL, 2655
Settings for the PG/PC interface, 10862
Automatic bus parameter detection, 10870
Setting parameters for the MPI interface, 10871
Setting parameters for the PROFIBUS interface, 10873
Setup
Direct data exchange, 1377
Seven-segment display, 3441, 3743, 4323, 5577, 5792, 5899, 6263, 6393
Severity, 767
SF, 3218, 3514, 3944, 5414, 5626, 5948
SFP diagnostics, 1259

SGN_GET, 5292
SGN_SET, 5293
SHA algorithm, 1235
SHA1, 782, 788
Shadow memory, 7918
Shared device
    Associated IO controllers, 2094
Shared Device, 1521, 1536, 1848
Shared PtP parameter errors, 5184, 7471
Shift
    Left, 3417, 3719, 4091, 4094, 4300, 4453, 5517, 5729, 6089, 6093, 6220, 6346
    Right, 3415, 3717, 4087, 4089, 4093, 4096, 4298, 4451, 5515, 5727, 6086, 6088, 6091, 6095, 6218, 6344
Shift register, 5572, 5575, 5782, 5788, 5893, 5896, 6258, 6261, 6387, 6390
SHL, 3417, 3719, 4300, 4453, 5517, 5729, 6220, 6346
Short-circuit to ground, 2045
Short-circuit to L+, 2046
Show
    Limit lines on the bar, 8076
Show line numbers, 2692
Show/hide interfaces, 10863
ShowAlarmWindow, 8347
Showing and hiding absolute operands, 2693
Showing and hiding columns, 372, 2496, 2567, 2848, 2863
ShowLogonDialog, 8345
ShowOperatorNotes, 8346
SHR, 3415, 3717, 4298, 4451, 5515, 5727, 6218, 6344
SHRB, 5575, 5788, 5896, 6261, 6390
Shutdown
    Incremental approach, 10487, 10579
    Jog, 10474
    Jogging, 10565
Shutdown delay, 10757
SiClock, 754
Sign, 3278, 3574, 3978, 4352, 5448, 5660, 5982, 6288
Signal board, 941
    Inserting, 942
Signal evaluation, 9030, 9055, 9108
Signal N, 9031, 9056
Signal type, 9029, 9055, 9062
Signal_Get, 977, 1009
Signal_Get (get RS232 signals), 5199, 7486
Signal_Get (set RS232 signals), 5200, 7487
Signal_Set, 978, 1009
SIMATIC LOGO!
    Communication, 8618
    Connection parameters, 8621
    Parameter, 8621
SIMATIC memory card, 1839
    Formatting, 2113
SIMATIC NET PC software for communications processors, 1969
SIMATIC S7
    Communication partners, 8477, 8531, 8588, 8597, 8600
SIMATIC S7 Embedded Controller, 41
SIMATIC S7 200
    Migrating data types, 169
SIMATIC S7 300/400
    Migrating data types, 169
simple alarm view, 8944
    Control element, 8945
    Operation, 8945
    Use, 8944
Simple alarm view
    Application, 8181
    Control element, 8182
    Operation, 8182
simple alarm window
    Application, 8181
    Use, 8944
Simple recipe view, 8219
    Behavior, 8234, 8954
    Layout, 8233, 8953
    Menu command, 8234, 8953, 8954
    Operation, 8233, 8953
Simple user view, 8936
    Layout, 8936
    Operation, 8937
    Use, 8936
Simulate hardware, 472
Simulate modules, 472
Simulate software, 472
SimulateSystemKey, 8339
SimulateTag, 8340
Simulating
    Project with tag simulator, 8889
Simulating devices, 472
Simulation, 79
    Easy Motion Control, 10773
    PLC connection, 79
    Runtime, 8887
Simulation speed, 9394, 9470
Simultaneous branch
    Creating, 2788
    Introduction, 2721

SIN, 3290, 3588, 3984, 4173, 4363, 5460, 5672, 5987, 6164, 6298
Sine, 3290, 3588, 3984, 4173, 4363, 5460, 5672, 5987, 6164, 6298
Single evaluation, 9108
Single instance
    Correcting the call type, 2590, 2631, 2675
    Definition, 2183
    Example, 2183
Single step mode
    Basics, 2969
    Deleting breakpoints, 2976
    Disable breakpoint, 2973
    Enable breakpoint., 2973
    Exit test, 2975
    Navigating to the breakpoints, 2971
    Resume program execution, 2974
    Run to cursor, 2974
    Setting breakpoints, 2970
    Setting the call environment for breakpoints, 2972
    Step into, 2974
    Step over, 2974
Single step view, 2767
Sinking output, 9031, 9056
SINT, 2244, 2294, 2332, 2378, 2398
SINT_TO_, 2332, 2398
Size
    Editing multiple objects, 8008
Size of the user data IO device, 1932
Slave, 1083
SLC, 8661
SLD, 4094, 6093
Slice access, 2216
Slicing, 246
Slot
    Determining logical address with LGC_GADR, 4821, 6790
    Determining logical address with LOG_GEO, 4817, 6784
    Racks, 499
    Select, 500
    Slot model, 1391
    Slot model with DPV1, 1391
    User data with DPV1, 1391
Slot model with DPV1, 1391
Slot rules, 496
    DP slave, 1341, 1351
    DPV1, 1391
    ET 200M, 1341
    ET 200S, 1356, 1425
    OPC server, 1157
    PC station, 1157
    S7-1500, 868
    S7-300, 865
    S7-400, 866
SLW, 4091, 6089
SM 338
    Addressing, 10424
    Diagnostic interrupt, 10421
    Diagnostic messages, 10427
    Isochronous mode, 10421
    Transmission rate, 10422
Smartdrive
    Installation, 59
SMC, 3436, 3737, 3886, 4318, 4477, 5565, 5775, 5883, 6252, 6380
Smoothing, 2038, 2056
SMS text, 5160, 5164, 5170
SMTP Client, 1231
SNC_RTCB, 4515
SNMP, 742, 1182, 1220, 1234, 1950
    Enable, 1950
    Groups, 1234
    SNMP trap, 1234
    SNMPv1, 1182
    SNMPv2c, 1182
    SNMPv3, 1182
    Users, 1235
SNMPv1, 782
SNMPv1 agent, 1950
SNMPv3, 782
Softkey, 8056, 8060
    global, 8058
SOFTNET Security Client
    Configuring in the project, 809
    Creating a configuration file, 809
    Database, 809
    Using, 808
SOFTNET Security Client, 725
Software
    HMI device, 8892
Software change, 1844
Software controller
    Configuring, 9532
Software data flow control, 985
Software gate, 8980, 8984, 9100
    Opening and closing, 9102
Software handshake, 1075
Software limit switch, 9386, 9462, 10247, 10509, 10606
Software requirements at the plant end
    Communications instruction "PG_DIAL", 10967

Software requirements at the system end
Communications instruction "AS_MAIL", 10971
Communications instruction "SMS_SEND", 10970
Software requirements for AS-AS remote link
Communication instruction "AS_DIAL", 10969
Software requirements for the programming device/ personal computer
Communications instruction "PG_DIAL", 10967
Software version, 1196
Software-controlled data flow control, 1602
Sourcing output, 9031, 9056
SP, 3212, 3507, 3935, 5408, 5619, 5939
SP_GEN
Input parameters, 7024
Instruction, 7022
Output parameters, 7025
Spanning Tree
Information, 1200
Rapid Spanning Tree, 1185
Special characters
Alarm text, 68
Special considerations
Omron Host Link, 8741
Special tag (web server), 921, 956
Special tracks, 9343
Speed axis
Configuration, 10298
Tags, 10385
Speedy Splitter, 477, 480, 482, 492
SPLT_RAN
Input parameters, 7027
Instruction, 7025
Output parameters, 7027
SQR, 3979, 4169, 5983, 6160
SQR , 3285, 3583, 4358, 5454, 5667, 6294
SQRT, 3286, 3584, 3980, 4170, 4360, 5456, 5668, 5984, 6161, 6295
Square, 3285, 3583, 3979, 4169, 5454, 5667, 5983, 6160, 6294
Square root, 3286, 3584, 3980, 4170, 5456, 5668, 5984, 6161, 6295
SR, 3164, 3459, 5380, 5591
SRD, 4096, 6095
SRT_DINT, 4706
SRW, 4093, 6091
SS, 3217, 3512, 3942, 5412, 5624, 5946
SSD, 4089, 6088
SSH server, 1219
SSI, 4087, 6086
SSI absolute encoder, 9061, 9062
SSL certificate, 736
STA, 2453
Stamping, 8006
Stand. frequency, 10752
Standard
PROFIBUS, 8499, 8552
Standard device, 1462
Standard machine project, 1540, 1543, 1546, 1549
Standard OB events, 1817
Standard router, 829
Standard slave, 1383, 1435
Standard user, 739
Standard user events, 1829
Standby module
Substitute value, 1362, 1430
Standstill range, 10452, 10507, 10604
Standstill velocity, 10508, 10605
Monitoring, 10467, 10558
Start address, 880, 881, 937, 1577, 1846
Start character, 991
Start information, 1890, 2170
As for S7-300 and S7-400, 1890, 1891, 1892, 1895, 1896, 1897, 1899, 1900, 1902, 1903, 1904, 1907, 1908, 1909, 1911, 1912, 1915
optimized, 1890, 1891, 1892, 1895, 1896, 1897, 1899, 1900, 1902, 1903, 1904, 1907, 1908, 1909, 1911, 1912, 1915
Start of a travel order
Easy Motion Control, 10805
Start off-delay timer, 3486
Start on-delay timer, 3484
Start pulse timer, 3482
Start screen, 7969
Start sequences, 991
Start speed for reference point approach, 10516, 10613
Start the online and diagnostics view, 2088
Start the simulation, 473
Start value, 48, 1839, 8980, 9034
Tag, 8127, 8128
Starting firmware loader, 2162
Starting NCM S7 diagnostics, 2159
Starting removal, 112
Starting the migration tool, 120
Starting the topology view, 711
Startup, 1614, 1842
ABS_DIAG, 7128
ABS_DIAG_451, 7132
CAM_DIAG, 7159
CAM_DIAG_452, 7163
CAM_MSRM_452, 7164
Complete overview, 1648
CPU, 7127, 7131, 7158, 7161

FM 351, 7127
FM 352, 7158, 7161
FM 451, 7131
FM 452, 7158, 7161
Organization blocks, 1614
STARTUP, 1641
Automatic restart (warm restart), 1642
Automatic restart (warm restart) without battery backup, 1642
Cancellation, 1646
Function, 1565, 1833
Instructions for generating PLC alarms, 6652
Manual restart (warm restart), 1642
Retentive behavior, 1644
STARTUP activities, 1567, 1645, 1834
Startup (S7-1500), 1891
Startup characteristics, 1153
CNT_CTL1, 7100
CNT_CTRL, 7097
Startup OB
Description, 1614
Startup OBs (S7-1500), 1891
Startup OBs OB 100, OB 101, OB 102 (S7-300, S7-400), 1807
Startup of IO devices, 1505
Startup parameters, 1673
Start-up parameters, 1568, 1835
Startup program (S7-1500), 1891
Startup routine, 1614
Startup type, 1586, 1860
Stateful packet inspection, 746
Static parameters of GRAPH function block, 2749, 2751, 2752
Static tags
High_Speed_Counter, 4927
Station
Copying, 511
Deleting, 510
Moving, 512
Read information with GetStationInfo, 4750
Renaming, 721
Station failure, (Rack error)
Station name, (See device name)
Station_Info, 885
Statistical data, 2156
Status, 1618, 2453
Module defective, 2101
Status bit, 2453, 2458, 2459, 2460, 2462, 2466, 2470
Query, 2463, 2467, 5736
Status bits
Query, 2464, 2469
Status interrupt, 1618
Status interrupt (S7-1500), 1896
Status interrupt OB, 1618
Status interrupt OB (OB 55) (S7-300, S7-400), 1764
Status interrupt OB (S7-1500), 1896
Status of the online connection, 2125
Status of the write job
CNT2RDPN, 7124
CNT2WRPN, 7122
Status symbols, 1999
Status word, 2453, 2458, 2459, 2460, 2462, 2463, 2466, 2467, 2470
Step
Actions, 2726
Adding comments, 2784
Basics, 2724
Copying and inserting, 2785
Deleting, 2786
Disable all steps, 2984
Elements, 2725
Inserting, 2779
Interlock, 2725
Renaming, 2783
Renumbering, 2781
Skipping, 2786
Static parameters, 2749
Supervision, 2725
Stepper drive, 10751
Stepper drive type, 10751
Stepper drive with Stepper encoder type PULSE REFEED operation, 10754
Stepper motor, 10006
STL, 2650, 3100, 3135
Detailed comparison, 2880
Migration, 135
Parameter passing, 2654
STL instruction
Copying, 2678
Cutting, 2678
Deleting, 2680
Entering, 2664
Inserting an operand, 2667
paste from Clipboard, 2679
Pasting, 2665
Select, 2677
STL program
Comment section, 2676
Entering an instruction, 2664
Inserting a block call, 2671, 2673
Inserting an instruction, 2665
Line comment, 2676
Program status display, 2989

Rules, 2663
Select instructions, 2677
STL source
Generating blocks, 2868
STOP, 1569, 1641, 1648, 1836
Stop events, 1821
Stop forcing, 3045, 3046
Stopping
Easy Motion Control, 10773
StopRuntime, 8341
Storing
External graphic, 8011
STP, 3387, 3687, 3848, 4278, 4433, 5499, 5711, 5836, 6207, 6332
Strg_TO_Chars, 4527
STRG_VAL, 4522
String
Comparing tags with S_COMP, 4519, 6440
STRING, 2224, 2264, 2319, 2369, 2390, 2416, 2430, 2446
Addressing, 2215
STRING_TO_, 2369, 2416, 2446
STRUCT
Addressing, 2215
Declaration in global data blocks, 2835
Declaration in the block interface, 2554
Declaring in a PLC data type, 2858
Structure, 2272
Structure, 2041, 2053
Call structure, 2948
of the cross-reference list, 2962, 8821
Of the dependency structure, 2953
Resources tab, 2959
STRUCT, 2272
Structured programming, 2168
STX, 996
Style
Changing, 7978
Creating a new type, 8782
Deleting, 7977
Edit, 7978
Setting, 8005
SUB, 3269, 3565, 5441, 5653
Subfield types, 7819
Submodule, 1536
Submodule configuration, 1848
Subnet, 536
Subnet mask, 534, 1442
Substitute value, 5548, 5758, 5858, 6235, 6362, 9133, 9179
Standby module, 1362, 1430
Subtract, 3269, 3565, 3961, 3966, 3975, 5441, 5653, 5965, 5969, 5979
Subtract timers with T_SUB, 6410
Subtracting
Subtracting timers with T_SUB, 4505
Summary of bit, byte, word, and double word identifiers, 1744
Supervision
Acknowledging supervision errors, 2985
Introduction, 2725, 2740
Programming, 2801
Supply voltage, 2047
Supported hardware compatibility, 1860
SW_GATE, 8984
SWAP, 3330, 3629, 4198, 4388, 6305
Swap bytes, 3330, 4388, 6305
Swapping, 4061, 6006, 6025, 6026, 6062
Switch, 8098, 8932
Application, 8932
Behavior, 8933
Layout, 8932
Operation, 8932
Type, 8099
SWITCH, 3377, 3677
Input parameters, 7028
Instruction, 7027
Output parameters, 7028
Switch buffer, 8467, 8626
Switch OFF, 8363
Switch ON, 8363
Switch points, 10452
Switching, 4003, 4022, 4023
Switching between basic mode and expanded mode in the force table, 3024
Switching between basic mode and expanded mode in the watch table, 2996
Switching characteristics
of time-based cams, 9361, 9436
Switching operating mode, 2107
Switching ranges, 10452
Switch-off difference, 10452, 10504, 10601
Switch-off difference minus, 10505, 10602
Switch-off difference plus, 10505, 10602
Switch-off point, 10452
Switch-over difference, 10452
Switch-over point, 10452
Switchover time, 1604
Switchport, 1460, 8441, 8490, 8543
Symbol
Alarm classes, 8173
For value comparison, 9538
Symbolic addressing, 247

of a tag, 8113, 8415
Symbolic I/O field, 8101
Mode, 8102
Text list, 8032, 8102
Symbolic IO field, 8933
Application, 8933
Behavior, 8934
Operation, 8934
Symbolic programming
Displaying absolute addresses, 2534
Symbolism, 244
Symbols
In the assignment list, 2939
In the call structure, 2947
In the dependency structure, 2954
Symmetrical encoders, 9106
SYNC, 1356
Sync domain, 1461, 1476, 1477, 2094, 8442, 8491, 8544
Sync domain properties, 2094
Sync master, 1479
SYNC/FREEZE control command, 1355
SYNC_PI, 1683, 4553
SYNC_PO, 1683, 4555
Synchronization, 8981, 8991, 9035, 9058
at signal N, 8996, 9035, 9058
by digital input, 8994
Easy Motion Control, 10771
From DP slaves with DP_SYC_FR, 4608, 6517
loss of, 9366, 9442
Loss of, 10584
Reference point approach, 10476, 10567
Set reference point, 10490, 10584
Slave clocks with SNC_RTCB, 4515, 6416
Synchronization (user-defined web pages), 914, 949
Synchronize, 8111, 8413
Synchronized execution cycles, 1397
Synchronizing a PLC tag, 8124
Synchronizing the time, 1685, 1882
Synchronizing user-defined web pages, 5132
Synchronous
Transferring data, 8462, 8570, 8613, 8755
Synchronous cycle (S7-1500), 1903
Synchronous cycle interrupt, 1683
Synchronous error
OB 121 (S7-300, S7-400), 1811
OB 122 (S7-300, S7-400), 1814
Synchronous errors, 1817
Masking with MSK_FLT, 4715, 6620
Querying the error register with READ_ERR, 4717, 6622
Unmasking with DMSK_FLT, 4716, 6621
Syntax
Addressing, 8641
Syntax errors
Basics, 2818
Finding errors, 2819
Syntax for AWP commands, 917, 952
Syntax for constants, 1744
Syslog Client, 1246
Syslog server, 798
System
General information, 1221
System configuration, 1219
System alarm, 8150
Meaning, 8186, 8187, 8191, 8196
System block
Displaying, 2152
System blocks folder, 48, 2500
System cycle, 1481
System data
Parameter field for, 7818
System diagnostics, 8207
Created blocks, 2140
Detail view, 8208
Device view, 8208
Diagnostic buffer view, 8208
Diagnostics status DB, 2142
Introduction, 2136, 2147
System diagnostics view, 8207
System diagnostics view
Configuring, 8210
Detail view, 8103
Device view, 8103
Diagnostic buffer view, 8103
System event, 8150
Meaning, 8186, 8187, 8190, 8191, 8193, 8194, 8195, 8196, 8197, 8198, 8200, 8202, 8203
Parameters, 8186
System events, 804, 807
Configuring, 8159
System function
Application, 8276
Function list, 8278
In the function list, 8276
Usage, 8275
Use, 8276
System functions, 8274
AcknowledgeAlarm, 8316
ActivateCleanScreen, 8290
ActivatePreviousScreen, 8291
ActivateScreen, 8288
ActivateScreenByNumber, 8289
AdjustContrast, 8287

AlarmViewAcknowledgeAlarm, 8315
AlarmViewEditAlarm, 8314
AlarmViewShowOperatorNotes, 8315
Available, 8284
CalibrateTouchScreen, 8302
ChangeConnection, 8344
ClearAlarmBuffer , 8312
ClearAlarmBufferProTool , 8313
DecreaseFocusedValue, 8342
DecreaseTag, 8343
EditAlarm, 8293
GetGroupNumber, 8309
GetPassword, 8309
GetUserName, 8308
GoToEnd, 8298
GoToHome, 8297
IncreaseFocusedValue, 8296
IncreaseTag, 8297
InvertBit, 8299
InvertBitInTag, 8300
InvertLinearScaling, 8301
LinearScaling, 8311
Logoff, 8287
Logon, 8292
PageDown, 8328
PageUp, 8327
Parameters in the user program, 876
PressButton, 8324
RecipeViewBack, 8321
RecipeViewClearDataRecord, 8317
RecipeViewGetDataRecordFromPLC, 8317
RecipeViewMenu, 8318
RecipeViewNewDataRecord, 8317
RecipeViewOpen, 8318
RecipeViewRenameDataRecord, 8320
RecipeViewSaveAsDataRecord, 8320
RecipeViewSaveDataRecord, 8320
RecipeViewSetDataRecordToPLC, 8319
RecipeViewShowOperatorNotes, 8321
ReleaseButton, 8325
ResetBit, 8322
ResetBitInTag, 8323
ScreenObjectCursorDown, 8294
ScreenObjectCursorUp, 8294
ScreenObjectPageDown, 8295
ScreenObjectPageUp, 8295
SetBit, 8329
SetBitInTag, 8330
SetBitWhileKeyPressed, 8332
SetConnectionMode, 8337
SetDeviceMode, 8328
SetLanguage, 8335
SetTag, 8336
ShowAlarmWindow, 8347
ShowLogonDialog, 8345
ShowOperatorNotes, 8346
SimulateSystemKey, 8339
SimulateTag, 8340
StopRuntime, 8341
TraceUserChange, 8342
TrendViewBackToBeginning, 8307
TrendViewCompress, 8305
TrendViewExtend, 8304
TrendViewRulerLeft, 8306
TrendViewRulerRight, 8305
TrendViewScrollBack, 8304
TrendViewScrollForward, 8303
TrendViewSetRulerMode, 8306
TrendViewStartStop, 8307
UpdateTag, 8292
System information
Addresses, 1392
System limits
HMI device, 8961, 8965
System log, 804, 807
System memory, 1586, 1607, 1655, 1860, 1883
Access via FETCH/WRITE, 655, 1959
Block stack, 1665
Diagnostics buffer, 1576, 1666, 2123
Interrupt stack, 1664
Local data stack, 1662
Operand areas, 1572, 1659
process image input/output, 1575, 1661
System of units
Selecting, 10497, 10594
System power supply, 2053
System requirement for WinCC Basic, 94
System requirements STEP 7 Professional, 91
System status list
SZL-ID W#16#xy90 - DP master system information, 6717
SZL-ID W#16#xy9C - Tool changeover information (PROFINET IO), 6732
Definition, 6688
Partial list extract SZL ID W#16#0132 index W#16#0005, 6710
Partial list extract SZL ID W#16#0132 index W#16#0008, 6711
Partial list extract SZL ID W#16#0132 index W#16#000B, 6713
Partial list extract SZL ID W#16#0132 index W#16#000C, 6713
Partial list extract SZL ID W#16#0232 index W#16#0004, 6714

Partial lists, overview, 6691
Reading partial SZL excerpt with RDSYSST, 6681
Structure of a partial list, 6689
SZL ID W#16#xy13 - user memory areas, 6695
SZL ID W#16#xy25 - assignment of process image partitions to OBs, 6707
SZL ID, structure, 6690
SZL-ID W#16#00B1 - module diagnostic information, 6736
SZL-ID W#16#00B2 - diagnostic data record 1 with physical address, 6737
SZL-ID W#16#00B3 - module diagnostic data with logical base address, 6738
SZL-ID W#16#00B4 - diagnostic data of a DP slave, 6738
SZL-ID W#16#0x94 - status information for rack/ station, 6726
SZL-ID W#16#xy11 - module Identification, 6692
SZL-ID W#16#xy12 - CPU characteristics, 6693
SZL-ID W#16#xy14 - system areas, 6696
SZL-ID W#16#xy15 - block types, 6697
SZL-ID W#16#xy1C - component identification, 6698
SZL-ID W#16#xy22 - interrupt status, 6704
SZL-ID W#16#xy32 - communication status data, 6709
SZL-ID W#16#xy74 - status of the module LEDs, 6715
SZL-ID W#16#xy91 - module status information, 6718
SZL-ID W#16#xy92 - rack / station status information, 6723
SZL-ID W#16#xy95 - extended DP master system information, 6728
SZL-ID W#16#xy96 - PROFINET IO and PROFIBUS DP module status information, 6729
SZL-ID W#16#xyA0 - diagnostics buffer, 6735
System synchronization
For testing GRAPH programs, 2982
Performing, 2986
System texts, 341
System Time
NTP Client, 1241
PTP Client, 1243
SNTP Client, 1240
System-defined role
Administrator, 739
diagnostics, 739
Remote access, 739
standard, 739
System-defined text lists
Editing, 411
Modifying texts, 412
System-relevant information
Manufacturer´s specifications, 2149
Module information, 2089
Module specifications, 2149
Vendor information, 2089

T

T, 4000, 6003
T STW, 4002, 6005
T_ADD, 4504
T_COMBINE, 4507
T_COMP, 4502
T_CONFIG, 5122
T_CONV, 4503
T_DIAG, 5117
T_DIFF, 4506
T_RESET, 5115
T_SUB, 4505
Table
Adding values with ATT, 6760
Calculating the standard deviation with DEV, 6774
Collecting / distributing data with PACK, 6780
Comparing table values with CDT, 6776
Connection, 8402
Copying values with TBL_WRD, 6769
Executing operations with TBL, 6767
Finding values with TBL_FIND, 6763
Linking values logically and storing with WRD_TBL, 6771
Linking with TBL_TBL, 6778
Outputting the first value with FIFO, 6761
Outputting the last value with LIFO, 6765
Table:Designing a border, 8001
Tag
absolute addressing, 8112, 8414
Acquisition cycle, 8128, 8135, 8145
Acquisition mode, 8128
Addressing, 8115
Changing the PLC, 8132
Comment, 8117
Configuration, 8121
Configuring several tags, 8121
Connection to PLC, 8115
Copy, 8120
Creating external tags, 8115
Creating internal tags, 8117
Data type, 8117

Deleting, 8120
Displaying or hiding tag information, 2603, 2645, 2668, 2702, 2810
Displaying values, 8146
Event, 8134
Exporting, 8807
External tag, 8110, 8412
Importing, 8809
in Runtime, 8128
Index tag, 8132, 8133
Indirect addressing, 8132, 8133
Internal tag, 8114, 8416
Length, 8117
Limit range, 8125
Limit value, 8126
Linear scaling, 8129, 8131
Maximum length, 75
Multiplexing, 8132
Name, 8117
Negative, 3176, 3470, 3748
Object list, 8116
Output in alarm, 8166
Overlaying, 2218
Overview, 2203
PLC tag, 2473
PLC tags and DB tags, 2203
Positive, 3174, 3469, 3747
Read continuously, 8129
Reconnecting, 8124
Rename, 8120
Start value, 8127, 8128
symbolic addressing, 8113, 8415
Update, 8129, 8135
Tag data
Structure for the import, 8810, 8815
Tag declaration
Automatically filling in cells, 2496, 2567, 2847, 2863
Based on a PLC data type, 2836
Block interface, 2544
Declaring ARRAY, 2553
Declaring PLC data type, 2555
Declaring STRUCT, 2554
Declaring tags, 2550, 2552, 2556
Deleting a tag, 2495, 2566, 2847, 2862
Importing and exporting tags, 2567, 2848
Inserting a table row, 2493, 2565, 2846, 2861
Inserting table rows at the end, 2566, 2846, 2862
Multi-instance, 2557
Overlaying tags, 2556
Purpose of tag declaration, 2544
Reserved key words, 2197
Retentivity, 2561
Showing and hiding columns , 2496, 2567, 2848, 2863
Sorting rows, 2495
Tag properties, 2560, 2562, 2841, 2843, 2845
Updating the block interface, 2557
Valid data types, 2548, 2549
Tag import
Tag data structure, 8810, 8815
Tag list
Indirect addressing, 8132, 8133
Tag simulator, 8890
Tag table
Default, 8109, 8411
for HMI devices, 8109, 8411
user-defined, 8109, 8411
Web server, 907
Tag value
Output, 8146
Tags, 1940, 1941
Basics, 8108, 8410
Data exchange, 8379
Migrating, 151
Migration, 165
Monitor all, 3011, 3036
Monitor now, 3037
Monitor once and now, 3012
Tags of the positioning axis technology object, 10365
Tags of the technology object external encoder, 10395
Tags of the technology object SpeedAxis, 10385
TAK, 4104, 6102
Tan, 4175, 6166
TAN, 3293, 3591, 3988, 4365, 5462, 5675, 5992, 6301
Tangent, 3293, 3591, 3988, 4175, 4365, 5462, 5675, 5992, 6166, 6301
TAR1, 4004, 6007
TAR1 <D>, 4005, 6008
TAR1 AR2, 4006, 6009
TAR2, 4007, 6010
TAR2 <D>, 4008, 6011
Target, 10452
Target approach, 10468, 10558
Target range, 10452, 10505, 10602
Monitoring, 10467, 10558
Task, 8366
Deactivate, 8369
Task card, 7964
Changing the pane mode, 297
Function, 295

Hardware catalog, 489, 8395
Reducing automatically, 288
Task Card
Online Tools, 2089
Tools, 7978
TCI (Tool Calling Interface), 1551
TCON, 5083, 5086
TCON_IP_RFC, 693
TCON_IP_v4, 691
TCON_Param, 689
TCONT_CP
In/out parameters, 6826
Input parameters, 6825
Instruction, 6812
Mode of operation, 6813
Output parameters, 6826
Static tags, 6827
TCONT_S
In/out parameters, 6839
Input parameters, 6838
Instruction, 6833
Operating principle, 6834
Output parameters, 6839
Static tags, 6839
TCP, 750
Characteristics, 687
Port numbers, 694
TCP connection
Parameter field for, 7823
TCP port, 1960
TDISCON, 5091
TDP (System cycle), 1398
Team Engineering
Creating a master project, 10831
Example of the program structure in the master project, 10838
Integrating project copies into the master project once again, 10833
Introduction to shared commissioning, 10831
Notes on compatibility, 10833
Procedure for creating the master project, 10835
Procedure for integrating project copies into the master project, 10836
Procedure for manual synchronization of competing changes, 10836
Procedure for modifying central objects within the master project, 10837
Procedure for shared commissioning, 10835
Requirements for shared, parallel working on the project, 10834
Rules for downloading to the CPU, 10840
Rules for editing shared central objects, 10840
Rules for online functions, 10841
Rules for shared working on one CPU, 10839
Rules for the master project, 10838
Software and hardware requirements, 10833
Working with project copies, 10832
Technology alarms, 10350
List of the technology alarms, 10403
Technology data block, 10210, 10315
Change restart-relevant data, 10318
Evaluate StatusWord, ErrorWord and WarningWord, 10317
Evaluating the technology data block, 10315
Tags of the positioning axis technology object, 10365
Tags of the technology object external encoder, 10395
Tags of the technology object SpeedAxis, 10385
Technology module
Applications, 8978
Basic parameters, 9046
Control interface, 9075
Feedback interface, 9077
Hardware configuration, 9045
Online & Diagnostics, 9071
Parameter assignment, 9053
Performance features, 8977
Technology object
External encoder, 10210, 10305
High_Speed_Counter, 9024
Positioning axis, 10210, 10266
Speed axis, 10298
Speed-controlled axis, 10210, 10212
Technology object command table: Shortcut menu commands, 10062
Technology objects
CONT_C, 9622
CONT_S, 9627
PID_3Step, 9589
PID_Compact, 9559
PID_CP, 9653
PID_ES, 9703
TCONT_CP, 9630
TCONT_S, 9650
Telecontrol server, 1165
TELECONTROL SERVER BASIC, 1165
TeleService
Meaning of the icons, 10921
Phone book properties, 10919
Access to phone books, 10920
AS_DIAL, 7937, 10967
AS_MAIL, 7949
Callback variants, 10933

Communications instruction:"AS_DIAL", 10918
Communications instruction:"AS_MAIL", 10919
Communications instruction:"PG_DIAL", 10918
Communications instruction:"SMS_SEND", 10919
Communications instruction:"TM_MAIL", 10919
Communications instruction:"TMAIL_C", 10919
Connection establishment options:S7-1200 CPUs, 10965
Connection establishment options:S7-1500 CPUs, 10965
Connection establishment options:S7-300/400 CPUs, 10965
Defining dialing rules, 10926
Establish connection to AS, 7937, 10967
Establish connection to PG/PC, 7933
Export phone book, 10925
Functionality, 10918
Gateways, 10930
Import phone book, 10923
Inserting rows in the phone book, 10922
Modem support, 10929
Modem types / media, 10930
Open phone book, 10922
Performance in telephone networks, 10930
PG_DIAL, 7933
Phone book, 10919
Printing the phone book, 10926
Save phone book, 10923
Send text (SMS) message, 7943
Sending e-mails, 7949
Showing or hiding columns in the phone book, 10923
SMS_SEND, 7943
Structure of the phone book, 10921
Transferring e-mail with TM_MAIL, 5360
Working with the phone book, 10920
TeleService callback options, 10935
TeleService phone book, 10919
TeleService via GPRS, 1169
Establish connection, 1170
Temperature coefficient, 2037, 2056
Temperature compensation, 2024
template
Template, 7974
Template, 8056
Copy, 7974
Creating, 7973
Deleting, 7974
Global screen, 7973
Inserting, 7974
Move, 7974
Rename, 7974
Use in screen, 7976
temporary, 1166
Temporary connection, 1166
Temporary station, 1166
Temporary tags (TEMP)
in cyclic program (S7-300, S7-400), 1754
required for OBs, 1814
Tens complement, 3443, 3744, 3896, 4325, 4487, 5579, 5793, 5901, 6265, 6395
Terminal assignment
Encoder interface of FM 351, 10459
Encoder interface of FM 352, 9358
Encoder interface of FM 451, 10548
Front connector of FM 351, 10459
Front connector of FM 352, 9359
Front connector of FM 451, 10549
Front connector of FM 452, 9433
Front connector of the FM 350-2, 9163
Front connector of the FM 450-1, 9190
Front connector of the SM 338, 10420
Terminal modules and electronic modules, 2022
Terminating the remote connection, 10951
Test mode, 1692
Test options in the force table, 3021
Test settings for GRAPH
Changing, 2983
Overview, 2982
Text
Button, 8100
Text field, 8104
Text field, 8104
Size, 8105
Text field length
Asian languages , 8914
Text list
Application, 8026
Bit (0, 1), 8031
Bit number (0 - 31), 8032
Creating, 8027
Exporting, 8812
Importing, 8814
Symbolic I/O field, 8032, 8102
Value/Range, 8029
Text lists
Introduction, 409
System-defined, 409
Use in recipe data records, 8218
User-defined, 409
Text lists for alarms, 3071
Texts
Migrating, 155

TFTP
Load/save, 1227
THEN, 4258, 6190
Third-party drivers
Communication, 8630
Special features, 8631
THIS, 2211
Ti (input data time read-in), 1409
Ti (read in time of input data), 1398, 1489
Ti/To overlap
Operating principle, 1410
Requirement, 1409
Rules, 1412
Setting, 1411
TIA environment, 10748
TIA Portal
Exiting, 273
Starting, 273
Time, 1220, 1586, 1860, 2255, 2257
Switching from daylight saving time/standard time with SET_SW, 6429
TIME, 2257, 2307, 2355, 2386, 2411, 2425, 2440
Time accumulator, 3184, 3479, 3488, 3759, 4125, 4339, 5570, 5781, 5891, 6257, 6385
Time delay, 3396, 3696, 3857, 4286, 4440, 5500, 5712, 5836, 6206, 6332
Time delay interrupt, 1620
Time duration, 3196, 3491, 3764, 4129
Time error interrupt, 1624
Time error interrupt (S7-1500), 1907
Time error OB, 1624
Threshold mechanism, 1856
Time error OB (OB 80) (S7-300, S7-400), 1777
Time error OB (S7-1500), 1907
Time expired, 8364
Time of day
Precision Time Protocol, 1243
Setting the time of day in the Online and Diagnostics view, 2108
SNTP (Simple Network Time Protocol), 1240
Synchronization, 2153
System Time, 1237
Time-of-day synchronization, 1240
Time of day (S7-1500), 1893
Time stamp, 8158
Transferring alarms with TIMESTMP, 6434
Time stamping, 2025
Time synchronization, 778, 1586, 1860
TIME_TCK, 4516
TIME_TO_, 2355, 2411, 2440
Time-based cam
Activating, 9361, 9436
Influence of functions, 9361, 9436
Switching characteristics, 9361, 9436
Time-based cams, 9340, 9341
Time-delay interrupt
Canceling with CAN_DINT, 4708, 6612
Instructions, 4705, 6610
OB 20 to OB 23 (S7-300, S7-400), 1759
Querying with QRY_DINT, 4708, 6613
Settings, 1679
Starting with SRT_DINT, 4706, 6611
Time-delay interrupt (S7-1500), 1899
Time-delay interrupt OB, 1620
Time-delay interrupt OBs (S7-1500), 1899
Time-of-day, 2259, 2260, 2261, 2262
Calculate local time with BT_LT, 6424
Calculating base time with LT_BT, 6425
Calculating local time with SET_TIMEZONE, 4513
Reading the system time of the CPU with TIME_TCK, 4516, 6414
Reading the time-of-day and date of the CPU with RD_SYS_T, 6413
Reading time-of-day and date of CPU with RD_SYS_T, 4509
Set time-of-day interrupt with S_LTINT, 6427
Setting for CPU with WR_SYS_T, 4508
Setting the time-of-day and time-of-day status with SET_CLKS, 6415
Setting the time-of-day of the CPU with WR_SYS_T, 6412
Synchronizing between slaves and master, 6416
Time-of-day function, 1967
Basics, 1593, 1684, 1881
Clock parameters, 1594
Reading the time of day, 1684
Reading the time-of-day, 1593
Run-time meter, 1686
Setting the time of day, 1684
Setting the time-of-day, 1593
Time-of-day format, 1593, 1684, 1881
Time-of-day interrupt, 1680
Activating with ACT_TINT, 4703, 6607
Cancel, 1616
Canceling with CAN_TINT, 4702, 6607
Function, 1616
Instructions, 6602
OB 10 to OB 17 (S7-300, S7-400), 1755
Properties of SET_TINT, CAN_TINT, ACT_TINT, QRY_TINT, 6603
Querying with QRY_TINT, 4704, 6608
Rules, 1616
Set and activate, 1616

Setting with SET_TINT, 4697, 4699, 6605
Status query, 1616
Time-of-day interrupt (S7-1500), 1893
Time-of-day interrupt OB
Parameter assignment, 1629, 1916
Time-of-day interrupt OB (S7-1500), 1893
Time-of-day status
Adjusting with SET_SW_S, 6431
Time-of-day synchronization, 1925, 1964
PROFIBUS CP, 1964
Timer, 2257, 3198, 3493, 5395, 5606
Enable, 3928, 5932
Recording, 3193
Reset, 3933, 5938
TIMER, 2279
Timers
Adding with the instruction T_ADD, 4504, 6409
Combine date and time with T_COMBINE, 4507
Combining with T_COMBINE, 6411
Comparing time tags with T_COMP, 4502, 6407
Converting with T_CONV, 4503, 6408
Determining difference with T_DIFF, 4506, 6411
Subtracting with T_SUB, 4505, 6410
Time-triggered trends, 8468, 8625
TM, 1471
TM_MAIL, 5360
TMAIL_C, 50, 5073
TO, 4262
To (output data time read-in), 1398, 1409
To (output time of output data), 1489
TO_PositioningAxis, 10031
TOD, 2259, 2314, 2364, 2387, 2413, 2428, 2443
TOD_TO_, 2364, 2413, 2443
TOF, 3182, 3191, 3477, 3486, 3756, 4122, 4337, 5393, 5604, 5801, 6126, 6277
Toggle
Between runtime languages, 8913
Between Runtime languages, 8844
Tolerable lifebeat failures, 10757
TON, 3179, 3189, 3474, 3484, 3753, 4119, 4334, 5391, 5602, 5799, 6123, 6275
TONR, 3184, 3193, 3479, 3488, 3759, 4125, 4339, 5570, 5781, 6257, 6385
TONR_X, 5891
Tool Calling Interface (TCI), 1551
Tool changeover, (See Docking Station)
Toolbar
Order, 7982, 7991
Tools, 7978, (See Docking unit), (See Docking unit), (See Docking Unit)
Tooltip
display, 8921
Topologies, 1082
Topology discovery, 1461, 8442, 8491, 8544
Topology display, 1948
Topology editor, 536
Topology view
Adding device, 505
Adopt devices identified online, 725
Adopt port interconnections identified online, 724
Configured topology, 711, 712
Diagnostics status in the graphic view, 714
Diagnostics status in the table view, 714
Differences compared with the network view, 710
Functions, 709
Hardware and network editor, 482, 8391
Interconnecting ports, 717
Offline/online comparison, 715
Touch operation
IO field, 8926
TP, 3177, 3187, 3472, 3482, 3750, 4116, 4331, 5389, 5600, 5796, 6120, 6272
TP177A
Loading a project, 74
Trace, 10879
Bit track, 10884
CPU load, 10906
Curve diagram, 10884, 10903
Data storage, 10881
Installed traces, 10889
Lifetime of the values, 10905
Measurement, 10881, 10891, 10901
Measurement cursor, 10903
Pre-trigger, 10912
Printing, 10904
Project navigator, 10884
Quantity structure, 10906
Quick start, 10892
Recordable tags, 10904
Recording, 10881, 10900
Recording conditions, 10909, 10910, 10912
Recording cycle, 10894, 10916
Recording duration, 10916
Recording levels, 10905
Reduction, 10910
Sampling, 10894
Saving the trace configuration, 10899
Signal table, 10886, 10903
Signals, 10908, 10915
Supported devices, 10879
Trace configuration, 10881, 10889, 10898, 10899, 10902, 10915
Trace editor, 10898
Trace handling, 10888

Trigger, 10917
Trigger mode, 10911
Trigger tag, 10909, 10911
Trigger time, 10891
User interface, 10882, 10907
Trace function, 10879
Trace S7-1200/1500, 10904
TraceUserChange, 8342
Tracking
Easy Motion Control, 10773
Tracks
Brake cam track, 9405, 9480
Counter cam track, 9405, 9480
enabling, 9343
External enable, 9343
High count value for counter cam track, 9405, 9481
Setting a track signal, 9343
Special tracks, 9405, 9480
Track output control, 9404, 9480
Track result, 9342
Transfer, 4000, 4002, 4004, 4005, 4006, 4007, 4008, 6003, 6005, 6007, 6008, 6009, 6010, 6011
Transfer area, 1391
Transfer card, 1570, (See Memory Card)
Transfer card; see Transfer card, 1570
Transfer module mapping, 1471
Transferring
Recipe data record, 8958
Transferring data
Area pointer, 8417, 8452, 8501, 8561, 8600, 8745
Area pointer screen number, 8453, 8507, 8561, 8601, 8746
Coordination area pointer, 8456, 8505, 8564, 8604, 8749
Data record are pointer, 8753
Data record area pointer, 8460, 8511, 8568, 8608
Date/time area pointer, 8453, 8502, 8503, 8562, 8601, 8747
Date/time PLC area pointer, 8455, 8563, 8602, 8748
Job mailbox, 8463, 8514, 8572, 8610, 8757
Job mailbox area pointer, 8458, 8509, 8566, 8606, 8751
Operator input in the recipe view, 8513, 8571, 8609, 8756
Possible cause of error, 8465, 8516, 8574, 8612, 8759
Project ID area pointer, 8457, 8508, 8565, 8605, 8750
Triggering by means of a configured function, 8464, 8515, 8573, 8611, 8758
With synchronization, 8462, 8570, 8613, 8755
Without synchronization, 8461, 8512, 8569, 8613, 8754
Transferring the project
HMI device, 79
Recipe data record, 79
Transition
Copying and inserting, 2785
Deleting, 2786
Inserting, 2779
Introduction, 2726
Programming, 2802
Renaming, 2783
Renumbering, 2781
Static parameters, 2751
Translate
Editor, 8832
Translating texts, 340
Transmission integrity, 1037
with ASCII driver, 1039
With RK512, 1040
Transmission medium/duplex, 1458
Transmission rate, 10432, 10444
Transmission rate / duplex, 8440, 8489, 8542
Transmission security, 979
for Modbus and USS, 981
with 3964(R), 981
with Freeport, 980
Transparency
In graphic, 8009
transparent color, 8083
Display on panels, 8083
Display on the panel HMI, 8927
Traversing range limitation, 10245
TRCV, 5099, 5103
TRCV_C, 5061, 5066
Trend request
Trend transfer, 8467, 8626
Trend request area, 8467, 8626
Trend transfer area, 8467, 8626
Trend transfer area 1
Trend transfer area 2, 8467, 8626
Trend view, 8087, 8929
Application, 8929
Button, 8088
Configuring for values from the PLC, 8147
Layout, 8929
Operation, 8930
Operator control, 8930
TrendViewBackToBeginning, 8307

TrendViewCompress, 8305
TrendViewExtend, 8304
TrendViewRulerLeft, 8306
TrendViewRulerRight, 8305
TrendViewScrollBack, 8304
TrendViewScrollForward, 8303
TrendViewSetRulerMode, 8306
TrendViewStartStop, 8307
Trigger tag, 8158
Triggering
    Diagnostic interrupt, 9911
Triggering by means of a configured function
    Transferring data, 8464, 8515, 8573, 8611, 8758
Triggers, 8368
    Event trigger, 8369, 8372
TRUE, 4113, 4114
TRUNC, 3361, 3660, 4025, 4243, 4415, 5485, 5697, 6028, 6184, 6323
TS adapter, 79
TS Adapter IE Advanced
    Brief description, 10960
    Connection types, 10961
    Parameter assignment, 10963
    Parameter assignment options, 10962
TS Adapter MPI
    Brief description, 10937
    Default parameter assignment, 10938
    Direct connection, 10938
    Establishing a modem connection, 10939
    Establishing the direct connection, 10938
    Exporting adapter parameters, 10943
    Importing adapter parameters, 10944
    Modem connection, 10939
    parameter Assignment, 10941
    Parameter assignment , 10941
    Parameter assignment options, 10940
    Principle of operation, 10937
    Restoring the default parameter assignments, 10942
    Setting up access protection, 10934
TS Adapter IE
    Brief description, 10944
    Default parameter assignment, 10946
    Parameter assignment, 10949, 10963
    Principle of operation, 10945
TS Adapter IE Basic
    Connection to the GSM network, 10947
    Connection to the telephone network, 10947
    Connection to the telephone network through an external modem, 10948
    Connection types, 10946
TSAP, 1960
    ASCII code table, 697
    Structure, 552, 629, 696
TSAP assignment
    Examples, 698
TSEND, 5093, 5096
TSEND_C, 5050, 5054
TUN_EC
    In/out parameters, 6854
    Input parameters, 6851
    Instruction, 6842
    Operating principle, 6845
    Output parameters, 6852
TUN_ES
    In/out parameters, 6873
    Input parameters, 6869
    Instruction, 6861
    Operating principle, 6864
    Output parameters, 6871
TURCV, 5112
TUSEND, 5109
Twos complement, 3274, 3570, 4019, 4020, 4021, 4351, 5447, 5659, 6021, 6022, 6024, 6286
Type
    Library, 8765
Type of reference point approach, 10511, 10608
TypeOf, 4159
TypeOfElements, 4160
Types
    Reference point approach, 10480, 10571
Types of DP slave, 1351, 1421

U

UA redundancy server, 1983
UART data transmission, 1598
UBLKMOV, 50, 3350, 3649, 3821, 4233, 4404, 5474, 5686, 5820, 6174, 6312
UC, 4073, 6073
UDINT, 2248, 2300, 2344, 2383, 2407
UDINT_TO_, 2344, 2407
UDP, 750, 756, 777
    Characteristics, 688
    Port numbers, 694
UDP connection
    Parameter field for, 7824
UDP connection / FDL connection
    Unspecified, 622
UDT, 45, 245, 256
UFILL_BLK, 3328, 3627, 3804, 4196, 4386
UINT, 2246, 2297, 2339, 2381, 2403
UINT_TO_, 2339, 2403
ULINT, 2250, 2302, 2349

ULINT_TO_, 2349
UMOVE_BLK, 3324, 3623, 3800, 4192, 4382
Underflow, 2039, 2047
Undo set actual value
    Data in the channel DB, 10489
Undo set process value, 9361, 9437
    AVALREM_EN, 9363, 9439
Undoing actions
    Basics of undoing, 402
    Undoing last action, 405
    Undoing multiple actions, 405
Unidirectional/bidirectional data traffic, 982, 1033, 1049
Uninstalling license keys, 88
Uniqueness
    of object names, 143
Units of measure, 10215
Universal
    PROFIBUS, 8499, 8552
Universal Serial Interface Protocol, (See USS protocol)
Universal symbolism, 244
Unknown peers, 790
Unlock
    User, 8261
Unplugged module, 501
Unscale, 3369, 3670, 3839, 4256, 4424, 5489, 5701, 5830, 6188, 6326
UNSCALE, 3369, 3670, 3839, 4256, 4424, 5489, 5701, 5830, 6188, 6326
Unspecified connection
    Characteristics, 623
    Configuring graphically, 620
    Configuring interactively, 621
Unspecified connections
    In PROFIBUS, 596
    On Industrial Ethernet, 596
Unspecified CPU, 506
UNTIL, 4266, 6198
UPDAT_PI, 4550
UPDAT_PO, 4552
Update, 1619
    DP interconnections, 7918
    PROFINET CBA components, 7918
    Tag, 8129, 8135
Update cycle, 8145
Update interrupt, 1619
Update interrupt (S7-1500), 1897
Update interrupt OB, 1619
Update interrupt OB (OB 56) (S7-300, S7-400), 1768
Update interrupt OB (S7-1500), 1897
Update time, 1456, 9008
Update Time, 9088
UpdateTag, 8292
Updating
    Device proxy, 8788, 8793
    IPE data, 8788, 8793
    Operating system, 8897
Updating a library, 467, 8774
Updating automatically, 891
Updating DP interconnections with PN_DP, 7922
Updating the device version, 8896
Updating the firmware , 8896
Updating the operating system, 8896
Updating types to the latest version, 467, 8774
Upgrade
    Global library, 8875
    Project, 8874
    Project version, 8874
Upgrading a global library, 432
Upload, 1554
URCV, 5044
USB card readers, 34
Use, 8212
    Of recipes, 8212
    Screen navigation function keys, 8068
    simple alarm view, 8944
    simple alarm window, 8944
    Simple user view, 8936
    System function, 8276
Use global assignment
    Function key, 7972
Useful information on configuring the TS Adapter MPI, 10940
USEND, 5043
User
    Assigning roles, 741
    Changing, 8260
    Creating roles, 739
    Deleting in runtime, 8261
    Logon, 8262
    Setting up, 738
    Unlock, 8261
User administration, 1688, 8241
    Migration, 158
    Object with access protection, 8264
    Runtime settings, 7970, 8245, 8261, 8262
    Setting up, 8267
User change, 8356, 8371
User constant, 892
User data, 1668
    Addresses, 1391
    Area, 1577, 1670, 1845

Backing up, 8936
Restoring, 8936
Slots, 1391
User diagnostic alarm
Creating, 3057
Deleting, 3058
Editing, 3057
User group, 8935
Administer authorizations, 8253
Assigning, 8272
Assigning users, 8251
Change displayed name, 8254
Changing in runtime, 8261
Changing the name, 8253
Creating, 8271
Deleting, 8254
Managing, 8253
Unauthorized, 8263
User interface
"Reference projects" palette, 297
Details view, 299
Inspector window, 293
Maximizing the work area, 287
Minimizing the work area, 287
Overview window, 300
Portal view, 277
Project tree, 282
Project view, 279
Task card, 295
Views, 277
Work area, 285
User interface language, 8826
Selecting, 8829
User interface language for the web server, 911
User management, 1938
User Management
Web server, 911
User name, 739
User program
Error evaluation, 9422, 9500, 10534, 10633
Finding errors, 2819
Function, 2167
Testing, 2966
User texts, 341
User view, 8255
Application, 8077
Complex user view , 8257
Configuring, 8258
Layout, 8078
Number of lines, 8078
Simple user view, 8255
User-defined, 31
User-defined functions
In function lists, 8278
User-defined roles, 740
User-defined text lists
Creating, 410
Editing, 411
Editing value ranges and texts, 411
User-defined web page, 937
User-defined web pages, 913, 916, 924, 925, 927, 928, 948, 951, 960, 962, 963
Users, 8935
Admin, 8941
Assigning a user group, 8251
Changing the name, 8252
Changing, touch operation, 8940
Create, 8250
Creating, 8272, 8937
Creating in runtime, 8258
Creating, touch operation, 8937, 8939
Deleting, 8253, 8941
Logging logons, 8266
Logoff, 8938
Managing, 8252, 8260
Updating after change of user, 8371
Using a TS Adapter for TeleService, 10928, 10952
Using frames and cover pages from the library, 387
Using tags, 8079
Using the keyboard
Editing texts, 2527
FBD, 2527
GRAPH, 2527
LAD, 2527
Program editor, 2527
SCL, 2527
STL, 2527
Using the on-screen keyboard, 318
Using UDTs, 256
USINT, 2245, 2295, 2335, 2379, 2400
USINT_TO_, 2335, 2400
USS communication, 999
Controlling a transmission to a drive with USS_PORT, 5299
Data exchange with drives via USS_DRIVE, 5300
Modifying parameters in the drive with USS_WPM, 5304
Reading parameters from the drive with USS_RPM, 5303
USS instructions, 1011
USS master
Overview of functions, 1001, 1151
Sequence of function calls, 1151
USS protocol, 999, 1149

USS protocol: Data field, 1000
USS protocol: Network data block, 1150
USS protocol:Data encryption, 1000, 1150
USS protocol:Data transmission procedure, 1000, 1150
USS protocol:frame structure, 999
USS protocol:Message frame structure, 1149
USS protocol
General structure of the data block:Parameter area (PKW), 1000
General structure of the data block:Process data area (PZD), 1000
General structure of the network data block:Parameter area (PKW), 1150
General structure of the network data block:Process data area (PZD), 1150
Instructions, 5295
USS status codes, 5305
USS_DRIVE, 5300
USS_Drive_Control, 978, 1011, 5217, 5223, 7504, 7510
USS_PORT, 5299
USS_Port_Scan, 978, 1011, 5217, 5220, 7504, 7507
USS_Read_Param, 978, 1011, 5217, 5225, 7504, 7512
USS_RPM, 5303
USS_WPM, 5304
USS_Write_Param, 978, 1011, 5217, 5227, 7504, 7514

V

V24_SET, 1029
V24_SET_340, 1029
V24_SET_441, 1030
V24_STAT, 1029
V24_STAT_340, 1029
V24_STAT_441, 1030
VAL_STRG, 4524
Valid
Data type, 8637, 8681, 8691, 8708, 8721, 8737
Data types, 8471, 8522, 8578, 8618
Valid data type
Allen-Bradley DF1, 8659
Value, 3278, 3574, 3978, 5448, 5660, 5982
Value assignment, 2688
Value change, 8363
Value ranges for the IP address, subnet mask and address of the gateway, 1929
Value status, 1536, 1846, 1848
Value transfer
CNT_CTL1, 7107
DB parameters for count modes, 7108
DB parameters for measuring modes, 7110
Value/Range
Text list, 8029
Values
Comparing, 9538, 10032, 10058
Variable index, 254
VARIANT, 263, 2277, 3256, 3257, 3258, 3259, 3343, 3344, 3346, 3552, 3553, 3554, 3555, 3642, 3643, 3645, 3832, 3834, 4159, 4160, 4226, 4227, 4229, 4250, 4252
VARIANT_TO_DB_ANY, 3832, 4250
VariantGet, 3343, 3642, 4226
VariantPut, 3344, 3643, 4227
Velocity measurement, 9008
Easy Motion Control, 10814
Velocity override
Easy Motion Control, 10773
Velocity profile, 10249
Vendor ID, 1196
Version overview, 1637
Versioning of types
Released version, 447
Version in progress, 446
Version in test, 446
Versions in WinCC, 8870
Vertical movement
Animation, 8049
Vertical radius, 8082
View options
For the assignment list, 2941
Setting the call structure, 2950
Setting the dependency structure, 2955
Views of the cross-reference list, 2962, 8821
Virtual Machine (VM)
Supported virtualization platforms, 92, 96
Virus scanners, 97
Visualization, 1162
VLAN, 1179
Day, 1269
Port VID, 1269
Priority, 1269
VLAN ID, 1182
VLAN tag, 1181
VLAN operation, 786
VLAN tagging, 786
VMware - PC CPs, 43
VOID, 2279
VPN connection
Deleting CA certificates, 10958
Establishing, 10958
Installing CA certificate, 10955

Terminating, 10960
VPN connection is not established
Check and remedy, 10978
VPN group, 784
VRRP
Address overview, 1315
Backup router, 1186
Configuration, 1314
Master router, 1186
Router, 1313
Virtual router, 1186
VRRP router, 1186
VRRP Statistics, 1202

W

WAIT, 3396, 3696, 3857, 4286, 4440, 5500, 5712, 5836, 6206, 6332
Wake on LAN, 4641, 6552
Wake-up call, 1166
Wake-up SMS, 1166
WAN IP address
Specifying, 795
Warm restart, 1566, 1834
Warm restart (S7-300, S7-400), 1807
Watch table
Meaning of the icons, 2997
Basic mode, 2995
Copying, 2999
Creating, 2998
Example of filling out a watch table, 3000
Expanded mode, 2995
Layout, 2995
Loading data blocks during an active control job, 51
Meaning of the columns, 2995
Monitoring and modifying modes, 3009
Multiple access to the same CPU, 51
Opening, 2999
Overview of the display formats, 3004
Overview of the test options, 2994
Permitted operands, 3001
Permitted operands for modify values, 3002
Possible applications, 2994
Saving, 3000
Switching between basic mode and expanded mode, 2996
Syntax check, 3000
Testing wiring, 2994
Web server, 907
Watchdog time, 1457
WCHAR, 2263, 2318, 2368
WCHAR_TO_, 2368
Web access, 907, 908
Web application, 913, 948
Web Control DB, 929, 964
Web data blocks, 927, 962
Web diagnostics, 1948, 1949
Web pages
Update interval, 1949
Web pages in browser, 933, 969
Web server, 862, 903, 910, 911, 913, 916, 927, 942, 944, 948, 951, 962, 1687
Display classes for messages, 912
Enable, 908, 909, 945
Enumerations, 924, 960
Fragment, 925, 960
HTTPS, 912, 946
Managing access, 908
User Management, 911
When dialog is closed, 8356
When dialog is opened, 8355
When high limit is exceeded, 8355
When low limit is undershot, 8355
WHILE, 4265, 6197
WinCC flexible project
Migrate, 163
Migrating, 145
WinCC V7.0 SP3, 61
WinCC version
Compatibility, 8871
WinCC flexible Runtime
Closing, 8909
Window layouts
Changing order, 306
Deleting window layouts, 306
Wire break, 2038, 2055, 9047, 9401, 9477, 9914, 10523, 10620
Wire breakage, 9865
Wire-break, 9947
Wireless LAN/PB link, 1437
Wiring test
Easy Motion Control, 10814
Wiring tests, 2994
Wizard
Device wizard, 8760
WORD, 2241, 2290, 2325, 2376, 2393, 2422, 2432
Word logic operations, 2470
WORD_TO_, 2325, 2393, 2432
WORD_TO_BLOCK_DB, 2447
Work area
Embedding floating elements, 290
Floating elements, 290
Function, 285

Maximizing, 287
Maximizing elements, 292
Minimizing, 287
Minimizing elements, 292
Saving a layout of editors and tables, 306
Splitting, 289
Switching between elements, 292
Using grouped elements, 290
Work memory, 1572, 1656, 1659, 1839, 2957
Working step
to create screens, 7967
WR_DPARM, 4689
WR_LOC_T, 4511
WR_REC, 4599
WR_SYS_T, 4508
WRIT_DBL, 51, 4803
Write
Incremental dimension tables, 10556
Machine data, 10463
Write command
CNT2WRPN, 7121
Write field, 3316, 3614
Write jobs, 7143, 7145
Write memory address, 4213
Write memory area, 4216
Write memory bit, 4214
WRITE_BIG, 4224
WRITE_LITTLE, 4220
WriteToArrayDB, 3334, 3633, 3809, 4201, 4392
WriteToArrayDBL, 3340, 3639, 3814, 4206, 4397
Writing
A data record with WR_REC, 4599, 6496
Cam data, 9356
Control signals, 7125, 7129
Incremental dimension tables, 10465
Machine data, 9354, 10553
Writing data
in remote CPU with PUT, 5040, 7368
outside of the local S7 station with X_PUT, 7930
To DP standard slaves/PROFINET IO devices
with DPWR_DAT, 4601, 6500
within the local S7 station with I_PUT, 7915
Writing tags, 955
WRREC, 885, 4559
WSR, 5572, 5782, 5893, 6258, 6387
WSTRING, 2224, 2265, 2320, 2371
Addressing, 2215
WSTRING_TO_, 2371
WWW (instruction), 928, 963
WWW Synchronizing user-defined web pages, 5132

X

X, 3905, 5909
X(, 3913, 5918
X27 (RS422/485) interface, 1081
XDB configuration, 1160, 2000
xlsx file, 8801, 8808, 8812, 8814
XN, 3906, 5911
XN(, 3914, 5919
XOD, 4085, 6084
XON/XOFF, 985
XOR, 2686, 2687, 3402, 3449, 3450, 3703, 5508, 5585, 5586, 5720
XOW, 4079, 6078

Z

Zero mark, 10224
Zero offset, 10582
Canceling, 9366, 9442
Data in the channel DB, 10582
Data used in the channel DB, 9365, 9440
Definition, 9364, 9439
Effects, 10583
on a linear axis, 9365, 9440
on a rotary axis, 9366, 9441
Requirement, 10582
Sequence, 9364, 9440, 10582
ZOFF_EN, 9364, 9439
Zero pass, 9119, 9197
ZONE
In/out parameters, 7306
Input parameters, 7305
Instruction, 7303
Static variables, 7306, 7310
Zoom
Adjusting the zoom setting, 478, 480, 483, 8384, 8390, 8392
Keyboard operation, 493
Zooming
Screen, 7966